Red Hat OpenShift Service on AWS 4

Applications de construction

Configurer Red Hat OpenShift Service sur AWS pour vos applications

Last Updated: 2025-05-27

Red Hat OpenShift Service on AWS 4 Applications de construction

Configurer Red Hat OpenShift Service sur AWS pour vos applications

Notice légale

Résumé

Ce document fournit des informations sur la configuration du service OpenShift Red Hat sur AWS (ROSA) pour les déploiements de votre application. Cela inclut la configuration de domaines wildcard personnalisés.

Table des matières

CHAPITRE 1. APERÇU DES APPLICATIONS DE CONSTRUCTION

En utilisant Red Hat OpenShift Service sur AWS, vous pouvez créer, éditer, supprimer et gérer des applications à l'aide de la console Web ou de l'interface de ligne de commande (CLI).

1.1. LE TRAVAIL SUR UN PROJET

En utilisant des projets, vous pouvez organiser et gérer des applications isolées. Il est possible de gérer l'ensemble du cycle de vie du projet, y compris la création, la visualisation et la suppression d'un projet dans Red Hat OpenShift Service sur AWS.

Après avoir créé le projet, vous pouvez accorder ou révoquer l'accès à un projet et gérer les rôles de cluster pour les utilisateurs en utilisant la perspective Développeur. En outre, vous pouvez modifier la ressource de configuration du projet tout en créant un modèle de projet qui est utilisé pour le provisionnement automatique de nouveaux projets.

En tant qu'utilisateur disposant d'autorisations d'administrateur dédiées, vous pouvez choisir d'empêcher un groupe d'utilisateurs authentifié de fournir automatiquement de nouveaux projets.

1.2. LE TRAVAIL SUR UNE APPLICATION

1.2.1. Créer une application

Afin de créer des applications, vous devez avoir créé un projet ou avoir accès à un projet avec les rôles et autorisations appropriés. Il est possible de créer une application en utilisant soit la perspective Développeur dans la console Web, les Opérateurs installés, soit l'OpenShift CLI (oc). Les applications à ajouter au projet peuvent être obtenues à partir des fichiers Git, JAR, devfiles ou du catalogue de développeurs.

Il est également possible d'utiliser des composants qui incluent du code source ou binaire, des images et des modèles pour créer une application à l'aide de l'OpenShift CLI (oc). Avec le Red Hat OpenShift Service sur la console web AWS, vous pouvez créer une application à partir d'un opérateur installé par un administrateur de cluster.

1.2.2. Le maintien d'une application

Après avoir créé l'application, vous pouvez utiliser la console Web pour surveiller vos métriques de projet ou d'application. Il est également possible d'éditer ou de supprimer l'application à l'aide de la console Web.

Lorsque l'application est en cours d'exécution, toutes les ressources des applications ne sont pas utilisées. En tant qu'administrateur de cluster, vous pouvez choisir d'activer ces ressources évolutives pour réduire la consommation de ressources.

1.2.3. Déploiement d'une application

Il est possible de déployer votre application à l'aide d'objets Deployment ou DeploymentConfig et de les gérer à partir de la console Web. Il est possible de créer des stratégies de déploiement qui aident à réduire les temps d'arrêt lors d'un changement ou d'une mise à niveau vers l'application.

Il est également possible d'utiliser Helm, un gestionnaire de paquets logiciels qui simplifie le déploiement d'applications et de services à Red Hat OpenShift Service sur les clusters AWS.

1.3. EN UTILISANT LE RED HAT MARKETPLACE

Le Red Hat Marketplace est un marché de cloud ouvert où vous pouvez découvrir et accéder à des logiciels certifiés pour les environnements basés sur des conteneurs qui fonctionnent sur les nuages publics et sur site.

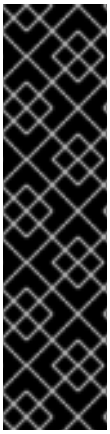
CHAPITRE 2. LES PROJETS

2.1. COLLABORER AVEC DES PROJETS

Le projet permet à une communauté d'utilisateurs d'organiser et de gérer leur contenu indépendamment des autres communautés.

NOTE

Les projets commençant par openshift- et kube- sont des projets par défaut. Ces projets hébergent des composants de cluster qui fonctionnent comme des pods et d'autres composants d'infrastructure. En tant que tel, Red Hat OpenShift Service sur AWS ne vous permet pas de créer des projets commençant par openshift- ou kube- à l'aide de la commande oc new-project. Les administrateurs de cluster peuvent créer ces projets à l'aide de la commande oc adm new-project.



IMPORTANT

Évitez d'exécuter des charges de travail ou de partager l'accès aux projets par défaut. Les projets par défaut sont réservés à l'exécution de composants de cluster de base.

Les projets par défaut suivants sont considérés comme hautement privilégiés: par défaut, kube-public, kube-system, openshift, openshift-infra, openshift-node, et d'autres projets créés par système qui ont l'étiquette openshift.io / run-level définie à 0 ou 1. La fonctionnalité qui repose sur des plugins d'admission, tels que l'admission de sécurité pod, les contraintes de contexte de sécurité, les quotas de ressources de cluster et la résolution de référence d'image, ne fonctionne pas dans des projets hautement privilégiés.

2.1.1. Créer un projet

Il est possible d'utiliser le service Red Hat OpenShift sur la console web AWS ou l'OpenShift CLI (oc) pour créer un projet dans votre cluster.

2.1.1.1. Créer un projet en utilisant la console web

Le Red Hat OpenShift Service vous permet de créer un projet dans votre cluster.

NOTE

Les projets commençant par openshift- et kube- sont considérés comme critiques par Red Hat OpenShift Service sur AWS. En tant que tel, Red Hat OpenShift Service sur AWS ne vous permet pas de créer des projets commençant par openshift- à l'aide de la console Web.

Conditions préalables

- Assurez-vous d'avoir les rôles et autorisations appropriés pour créer des projets, des applications et d'autres charges de travail dans Red Hat OpenShift Service sur AWS.

Procédure

- Lorsque vous utilisez la perspective de l'administrateur:

a. Accédez à Home → Projets.

b. Cliquez sur Créer un projet:

i. Dans la boîte de dialogue Créer un projet, entrez un nom unique, tel que myproject, dans le champ Nom.

ii. Facultatif: Ajoutez le nom de l'affichage et les détails de la description pour le projet.

iii. Cliquez sur **Create**.
Le tableau de bord de votre projet est affiché.

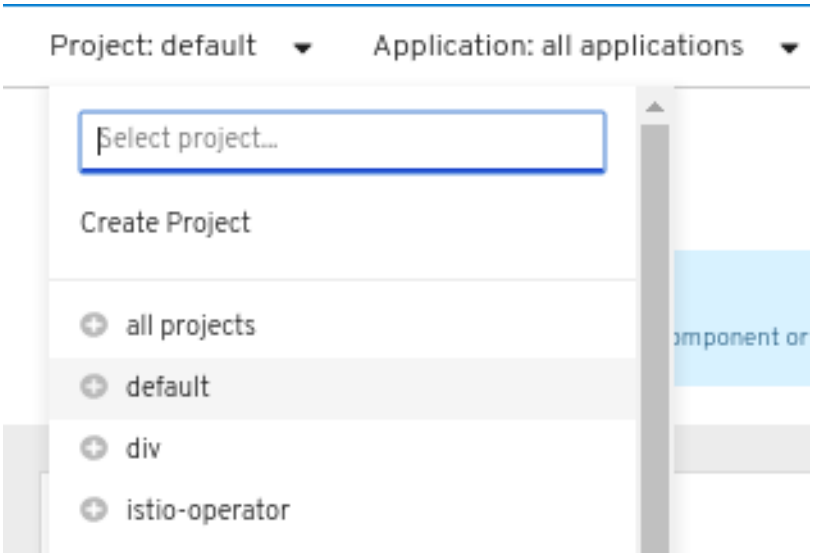
c. Facultatif: Sélectionnez l'onglet Détails pour afficher les détails du projet.

d. Facultatif : Si vous disposez d'autorisations adéquates pour un projet, vous pouvez utiliser l'onglet Accès au projet pour fournir ou révoquer les privilèges d'administration, d'édition et de visualisation du projet.

- Lorsque vous utilisez la perspective Développeur:

a. Cliquez sur le menu Projet et sélectionnez Créer un projet:

Figure 2.1. Créer un projet



i. Dans la boîte de dialogue Créer un projet, entrez un nom unique, tel que myproject, dans le champ Nom.

ii. Facultatif: Ajoutez le nom de l'affichage et les détails de la description pour le projet.

iii. Cliquez sur **Create**.

b. Facultatif : Utilisez le panneau de navigation de gauche pour accéder à la vue Projet et voir le tableau de bord de votre projet.

c. Facultatif: Dans le tableau de bord du projet, sélectionnez l'onglet Détails pour afficher les détails du projet.

d. Facultatif : Si vous disposez d'autorisations adéquates pour un projet, vous pouvez utiliser l'onglet Accès au projet du tableau de bord du projet pour fournir ou révoquer les privilèges d'administration, d'édition et d'affichage du projet.

Ressources supplémentaires

Ressources supplémentaires

- Personnalisation des rôles de cluster disponibles à l'aide de la console Web

2.1.1.2. Créer un projet en utilisant le CLI

Lorsque votre administrateur de cluster l'autorise, vous pouvez créer un nouveau projet.

NOTE

Les projets commençant par openshift- et kube- sont considérés comme critiques par Red Hat OpenShift Service sur AWS. En tant que tel, Red Hat OpenShift Service sur AWS ne vous permet pas de créer des projets commençant par openshift- ou kube- à l'aide de la commande oc new-project. Les administrateurs de cluster peuvent créer ces projets à l'aide de la commande oc adm new-project.

Procédure

- Cours d'exécution:

```
$ oc new-project <project_name> \
    --description="<description>" --display-name="<display_name>"
```

À titre d'exemple:

```
$ oc new-project hello-openshift \
    --description="This is an example project" \
    --display-name="Hello OpenShift"
```

NOTE

Le nombre de projets que vous êtes autorisé à créer peut être limité par l'administrateur système. Après que votre limite est atteinte, vous devrez peut-être supprimer un projet existant afin d'en créer un nouveau.

2.1.2. Affichage d'un projet

Le service Red Hat OpenShift est disponible sur la console web AWS ou sur l'OpenShift CLI (oc) pour afficher un projet dans votre cluster.

2.1.2.1. Affichage d'un projet à l'aide de la console Web

Consultez les projets auxquels vous avez accès en utilisant le service Red Hat OpenShift sur la console web AWS.

Procédure

- Lorsque vous utilisez la perspective de l'administrateur:

  a. Accédez à Home → Projets dans le menu de navigation.

  b. Choisissez un projet à afficher. L'onglet Aperçu comprend un tableau de bord pour votre projet.

c. Choisissez l'onglet Détails pour afficher les détails du projet.

d. Choisissez l'onglet YAML pour afficher et mettre à jour la configuration YAML pour la ressource du projet.

e. Choisissez l'onglet Charges de travail pour afficher les charges de travail dans le projet.

f. Choisissez l'onglet RoleBindings pour afficher et créer des liens de rôle pour votre projet.

- Lorsque vous utilisez la perspective Développeur:

a. Accédez à la page Projet dans le menu de navigation.

b. Choisissez tous les projets dans le menu déroulant Projet en haut de l'écran pour énumérer tous les projets de votre cluster.

c. Choisissez un projet à afficher. L'onglet Aperçu comprend un tableau de bord pour votre projet.

d. Choisissez l'onglet Détails pour afficher les détails du projet.

e. Lorsque vous disposez d'autorisations adéquates pour un projet, sélectionnez l'onglet d'accès au projet et mettez à jour les privilèges pour le projet.

2.1.2.2. Affichage d'un projet à l'aide du CLI

Lorsque vous visualisez des projets, vous êtes limité à voir uniquement les projets auxquels vous avez accès en fonction de la politique d'autorisation.

Procédure

1. Afin de consulter une liste de projets, exécutez:

```
$ oc get projects
```

2. Il est possible de passer du projet actuel à un projet différent pour les opérations de CLI. Le projet spécifié est ensuite utilisé dans toutes les opérations ultérieures qui manipulent le contenu à portée de projet:

```
$ oc project <project_name>
```

2.1.3. Fournir des autorisations d'accès à votre projet en utilisant la perspective Développeur

Dans la perspective Développeur, vous pouvez utiliser la vue Projet pour accorder ou révoquer les autorisations d'accès à votre projet.

Conditions préalables

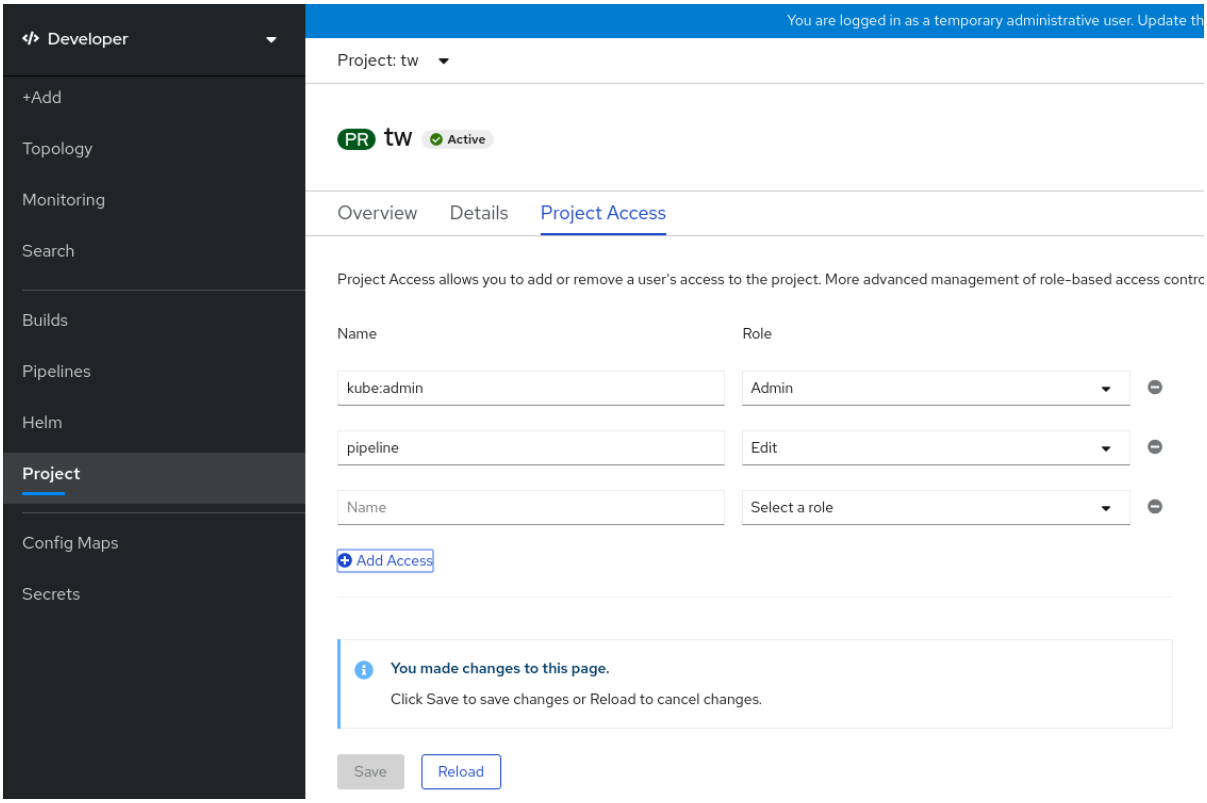
- « vous avez créé un projet.

Procédure

Ajouter des utilisateurs à votre projet et leur fournir un accès Admin, Edit ou Affichage:

1. Dans la perspective Développeur, accédez à la page Projet.
2. Choisissez votre projet dans le menu Projet.
3. Cliquez sur l'onglet Accès au projet.
4. Cliquez sur Ajouter un accès pour ajouter une nouvelle ligne d'autorisations aux autorisations par défaut.

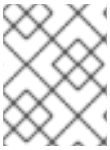
Figure 2.2. Autorisations de projet



5. Entrez le nom d'utilisateur, cliquez sur la liste déroulante Sélectionner un rôle et sélectionnez un rôle approprié.
6. Cliquez sur Enregistrer pour ajouter les nouvelles autorisations.

Il est également possible d'utiliser:

- La liste déroulante Sélectionner un rôle, pour modifier les autorisations d'accès d'un utilisateur existant.
- L'icône Supprimer l'accès, pour supprimer complètement les autorisations d'accès d'un utilisateur existant au projet.



NOTE

Le contrôle d'accès avancé basé sur les rôles est géré dans la perspective des rôles et des rôles dans la perspective de l'administrateur.

2.1.4. Personnalisation des rôles de cluster disponibles à l'aide de la console Web

Dans la perspective Développeur de la console web, la page d'accès Project → Project permet à un administrateur de projet d'accorder des rôles aux utilisateurs dans un projet. Les rôles de cluster

disponibles qui peuvent être accordés aux utilisateurs d'un projet sont l'administration, l'édition et la visualisation.

En tant qu'administrateur de cluster, vous pouvez définir quels rôles de cluster sont disponibles dans la page d'accès au projet pour tous les projets à l'échelle du cluster. Il est possible de spécifier les rôles disponibles en personnalisant l'objet spec.customization.projectAccess.availableClusterRoles dans la ressource de configuration de la console.

Conditions préalables

- En tant qu'utilisateur, vous avez accès au cluster avec le rôle cluster-admin.

Procédure

1. Dans la perspective de l'administrateur, accédez aux paramètres Administration → Cluster.
2. Cliquez sur l'onglet Configuration.
3. Dans la liste des ressources de configuration, sélectionnez Console operator.openshift.io.
4. Accédez à l'onglet YAML pour afficher et modifier le code YAML.
5. Dans le code YAML sous Spéc, personnalisez la liste des rôles de cluster disponibles pour l'accès au projet. L'exemple suivant spécifie les rôles d'administrateur, d'édition et de visualisation par défaut:

```
apiVersion: operator.openshift.io/v1
kind: Console
metadata:
  name: cluster
# ...
spec:
  customization:
    projectAccess:
      availableClusterRoles:
      - admin
      - edit
      - view
```

6. Cliquez sur Enregistrer pour enregistrer les modifications apportées à la ressource de configuration de la console.

La vérification

1. Dans la perspective Développeur, accédez à la page Projet.
2. Choisissez un projet dans le menu Projet.
3. Cliquez sur l'onglet Accès au projet.
4. Cliquez sur le menu dans la colonne Rôle et vérifiez que les rôles disponibles correspondent à la configuration que vous avez appliquée à la configuration de ressource de la console.

2.1.5. Ajouter à un projet

Ajoutez des éléments à votre projet en utilisant la page +Ajouter dans la perspective Développeur.

Conditions préalables

- « vous avez créé un projet.

Procédure

1. Dans la perspective Développeur, accédez à la page +Ajouter.
2. Choisissez votre projet dans le menu Projet.
3. Cliquez sur un élément de la page +Ajouter, puis suivez le flux de travail.

NOTE

Il est également possible d'utiliser la fonction de recherche dans la page Add* pour trouver d'autres éléments à ajouter à votre projet. Cliquez * sous Ajouter en haut de la page et tapez le nom d'un composant dans le champ de recherche.

2.1.6. Contrôle de l'état du projet

Le service Red Hat OpenShift peut être utilisé sur la console web AWS ou sur l'OpenShift CLI (oc) pour visualiser l'état de votre projet.

2.1.6.1. Contrôle de l'état du projet à l'aide de la console Web

Il est possible d'examiner l'état de votre projet à l'aide de la console web.

Conditions préalables

- « vous avez créé un projet.

Procédure

- Lorsque vous utilisez la perspective de l'administrateur:
  a. Accédez à Home → Projets.
  b. Choisissez un projet dans la liste.
  c. Examinez l'état du projet dans la page Aperçu.
- Lorsque vous utilisez la perspective Développeur:
  a. Accédez à la page Projet.
  b. Choisissez un projet dans le menu Projet.
  c. Examinez l'état du projet dans la page Aperçu.

2.1.6.2. Contrôle de l'état du projet en utilisant le CLI

Consultez l'état d'avancement de votre projet à l'aide de l'OpenShift CLI (oc).

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- « vous avez créé un projet.

Procédure

1. Basculez vers votre projet:

```
$ oc project <project_name> 1
```

1 <project_name> par le nom de votre projet.

2. Avoir un aperçu de haut niveau du projet:

```
$ oc status
```

2.1.7. La suppression d'un projet

Il est possible d'utiliser le service Red Hat OpenShift sur la console web AWS ou l'OpenShift CLI (oc) pour supprimer un projet.

Lorsque vous supprimez un projet, le serveur met à jour l'état du projet vers Terminating from Active. Ensuite, le serveur efface tout le contenu d'un projet qui est dans l'état de terminaison avant de finalement supprimer le projet. Bien qu'un projet soit en état de terminaison, vous ne pouvez pas ajouter de nouveau contenu au projet. Les projets peuvent être supprimés du CLI ou de la console web.

2.1.7.1. La suppression d'un projet en utilisant la console web

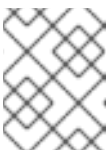
Il est possible de supprimer un projet en utilisant la console web.

Conditions préalables

- « vous avez créé un projet.
- Les autorisations requises pour supprimer le projet sont requises.

Procédure

- Lorsque vous utilisez la perspective de l'administrateur:
  a. Accédez à Home → Projets.
  b. Choisissez un projet dans la liste.
  c. Cliquez sur le menu déroulant Actions du projet et sélectionnez Supprimer le projet.



NOTE

L'option Supprimer le projet n'est pas disponible si vous n'avez pas les autorisations requises pour supprimer le projet.

1. Dans le volet Supprimer le projet?, confirmez la suppression en saisissant le nom de votre projet.

2. Cliquez sur Supprimer.

- Lorsque vous utilisez la perspective Développeur:

  a. Accédez à la page Projet.

  b. Choisissez le projet que vous souhaitez supprimer dans le menu Projet.

  c. Cliquez sur le menu déroulant Actions du projet et sélectionnez Supprimer le projet.

NOTE

Dans le cas où vous n'avez pas les autorisations requises pour supprimer le projet, l'option Supprimer le projet n'est pas disponible.

1. Dans le volet Supprimer le projet?, confirmez la suppression en saisissant le nom de votre projet.

2. Cliquez sur Supprimer.

2.1.7.2. La suppression d'un projet en utilisant le CLI

Il est possible de supprimer un projet en utilisant l'OpenShift CLI (oc).

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- « vous avez créé un projet.
- Les autorisations requises pour supprimer le projet sont requises.

Procédure

1. Effacer votre projet:

```
$ oc delete project <project_name> 1
```

1 <project_name> par le nom du projet que vous souhaitez supprimer.

2.2. CONFIGURATION DE LA CRÉATION DE PROJET

Dans Red Hat OpenShift Service sur AWS, les projets sont utilisés pour regrouper et isoler des objets connexes. Lorsqu'une demande est faite pour créer un nouveau projet à l'aide de la console Web ou de la commande oc new-project, un point de terminaison dans Red Hat OpenShift Service sur AWS est utilisé pour fournir le projet en fonction d'un modèle, qui peut être personnalisé.

En tant qu'administrateur de cluster, vous pouvez autoriser et configurer la façon dont les développeurs et les comptes de services peuvent créer, ou auto-provisionner, leurs propres projets.

2.2.1. À propos de la création de projet

Le Red Hat OpenShift Service sur le serveur AWS API fournit automatiquement de nouveaux projets en fonction du modèle de projet identifié par le paramètre projectRequestTemplate dans la ressource de configuration de projet du cluster. Dans le cas où le paramètre n'est pas défini, le serveur API crée un modèle par défaut qui crée un projet avec le nom demandé et attribue l'utilisateur demandeur au rôle d'administrateur de ce projet.

Lorsqu'une demande de projet est soumise, l'API remplace les paramètres suivants dans le modèle:

Tableau 2.1. Les paramètres du modèle de projet par défaut

Le paramètre	Description
AJOUTER AU PANIER PROJECT_NAME	Le nom du projet. C'est nécessaire.
LE PROJET_DISPLAYNAME	Le nom d'affichage du projet. C'est peut-être vide.
LE PROJET_DESCRIPTION	La description du projet. C'est peut-être vide.
LE PROJET_ADMIN_USER	Le nom d'utilisateur de l'utilisateur administrant.
ACCUEIL PROJET_REQUESTING_USER	Le nom d'utilisateur de l'utilisateur demandeur.

L'accès à l'API est accordé aux développeurs ayant le rôle d'auto-proviseur et le rôle de cluster d'auto-fournisseurs. Ce rôle est disponible pour tous les développeurs authentifiés par défaut.

2.2.2. La modification du modèle pour les nouveaux projets

En tant qu'administrateur de cluster, vous pouvez modifier le modèle de projet par défaut afin que de nouveaux projets soient créés en utilisant vos exigences personnalisées.

Créer votre propre modèle de projet personnalisé:

Conditions préalables

- Grâce à un compte doté d'autorisations d'administration dédiées, vous avez accès à un service Red Hat OpenShift sur AWS.

Procédure

1. Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.
2. Générer le modèle de projet par défaut:

```
$ oc adm create-bootstrap-project-template -o yaml > template.yaml
```

3. Créez un éditeur de texte pour modifier le fichier template.yaml généré en ajoutant des objets ou en modifiant des objets existants.

4. Le modèle de projet doit être créé dans l'espace de noms openshift-config. Chargez votre modèle modifié:

```
$ oc create -f template.yaml -n openshift-config
```

5. Éditez la ressource de configuration du projet à l'aide de la console Web ou du CLI.

   - En utilisant la console web:

     i. Accédez à la page Administration → Paramètres du cluster.

     ii. Cliquez sur Configuration pour afficher toutes les ressources de configuration.

     iii. Cherchez l'entrée pour Projet et cliquez sur Modifier YAML.

   - En utilisant le CLI:

     i. Editez la ressource project.config.openshift.io/cluster:

     ```
     $ oc edit project.config.openshift.io/cluster
     ```

6. Actualisez la section Spécifications pour inclure les paramètres projectRequestTemplate et nom, et définissez le nom de votre modèle de projet téléchargé. Le nom par défaut est project-request.

   Configuration du projet ressource avec modèle de projet personnalisé

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestTemplate:
    name: <template_name>
# ...
```

7. Après avoir enregistré vos modifications, créez un nouveau projet pour vérifier que vos modifications ont été appliquées avec succès.

2.2.3. Désactivation de l'auto-provisionnement du projet

Il est possible d'empêcher un groupe d'utilisateurs authentifié d'auto-provisionner de nouveaux projets.

Procédure

1. Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.

2. Affichez l'utilisation de la liaison de rôle du cluster auto-fournisseur en exécutant la commande suivante:

```
$ oc describe clusterrolebinding.rbac self-provisioners
```

   Exemple de sortie

```
Name:  self-provisioners
Labels:  <none>
Annotations: rbac.authorization.kubernetes.io/autoupdate=true
Role:
  Kind: ClusterRole
  Name: self-provisioner
Subjects:
  Kind Name    Namespace
  ---- ----    ---------
  Group system:authenticated:oauth
```

Examiner les sujets dans la section des autoprovisionnaires.

3. Enlevez le rôle de cluster d'auto-fournisseur du système de groupe:authenticated:oauth.

   - Lorsque la liaison de rôle du cluster auto-fournisseur lie uniquement le rôle d'auto-fournisseur au système:authenticated:oauth group, exécutez la commande suivante:

     ```
     $ oc patch clusterrolebinding.rbac self-provisioners -p '{"subjects": null}'
     ```

   - Lorsque la liaison de rôle de cluster auto-fournisseur lie le rôle d'auto-fournisseur à plus d'utilisateurs, de groupes ou de comptes de service que le groupe system:authenticated:oauth, exécutez la commande suivante:

     ```
     $ oc adm policy \
         remove-cluster-role-from-group self-provisioner \
         system:authenticated:oauth
     ```

4. Éditez la liaison du rôle de cluster d'auto-fournisseurs pour empêcher les mises à jour automatiques du rôle. Les mises à jour automatiques réinitialisent les rôles de cluster à l'état par défaut.

   - De mettre à jour la liaison de rôle à l'aide de l'ICC:

     i. Exécutez la commande suivante:

     ```
     $ oc edit clusterrolebinding.rbac self-provisioners
     ```

     ii. Dans la liaison de rôle affichée, définissez la valeur du paramètre rbac.authorization.kubernetes.io/autoupdate sur false, comme indiqué dans l'exemple suivant:

     ```
     apiVersion: authorization.openshift.io/v1
     kind: ClusterRoleBinding
     metadata:
       annotations:
         rbac.authorization.kubernetes.io/autoupdate: "false"
     # ...
     ```

   - De mettre à jour la liaison de rôle en utilisant une seule commande:

     ```
     $ oc patch clusterrolebinding.rbac self-provisioners -p '{ "metadata": { "annotations": {
     "rbac.authorization.kubernetes.io/autoupdate": "false" } } }'
     ```

5. Connectez-vous en tant qu'utilisateur authentifié et vérifiez qu'il ne peut plus auto-provisionner un projet:

```
$ oc new-project test
```

Exemple de sortie

```
Error from server (Forbidden): You may not request a new project via this API.
```

Envisagez de personnaliser ce message de demande de projet pour fournir des instructions plus utiles spécifiques à votre organisation.

2.2.4. Personnalisation du message de demande de projet

Lorsqu'un développeur ou un compte de service qui n'est pas en mesure d'auto-provisionner des projets fait une demande de création de projet à l'aide de la console Web ou du CLI, le message d'erreur suivant est retourné par défaut:

```
You may not request a new project via this API.
```

Les administrateurs de clusters peuvent personnaliser ce message. Envisagez de le mettre à jour pour fournir d'autres instructions sur la façon de demander un nouveau projet spécifique à votre organisation. À titre d'exemple:

- Afin de demander un projet, communiquez avec votre administrateur système à projectname@example.com.
- Afin de demander un nouveau projet, remplissez le formulaire de demande de projet situé à https://internal.example.com/openshift-project-request.

Afin de personnaliser le message de demande de projet:

Procédure

1. Éditez la ressource de configuration du projet à l'aide de la console Web ou du CLI.
   - En utilisant la console web:
     i. Accédez à la page Administration → Paramètres du cluster.
     ii. Cliquez sur Configuration pour afficher toutes les ressources de configuration.
     iii. Cherchez l'entrée pour Projet et cliquez sur Modifier YAML.
   - En utilisant le CLI:
     i. Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.
     ii. Editez la ressource project.config.openshift.io/cluster:

```
$ oc edit project.config.openshift.io/cluster
```

2. Actualisez la section Spécifications pour inclure le paramètre projectRequestMessage et définissez la valeur à votre message personnalisé:

Configuration du projet ressource avec message personnalisé de demande de

Configuration du projet ressource avec message personnalisé de demande de projet

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestMessage: <message_string>
# ...
```

À titre d'exemple:

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestMessage: To request a project, contact your system administrator at
projectname@example.com.
# ...
```

3. Après avoir enregistré vos modifications, essayez de créer un nouveau projet en tant que développeur ou compte de service qui n'est pas en mesure d'auto-provisionner des projets pour vérifier que vos modifications ont été appliquées avec succès.

CHAPITRE 3. CRÉATION D'APPLICATIONS

3.1. EN UTILISANT DES MODÈLES

Les sections suivantes fournissent un aperçu des modèles, ainsi que la façon de les utiliser et de les créer.

3.1.1. Comprendre les modèles

Le modèle décrit un ensemble d'objets qui peuvent être paramétrés et traités pour produire une liste d'objets à créer par Red Hat OpenShift Service sur AWS. Le modèle peut être traité pour créer tout ce que vous avez la permission de créer dans un projet, par exemple des services, des configurations de construction et des configurations de déploiement. Le modèle peut également définir un ensemble d'étiquettes à appliquer à chaque objet défini dans le modèle.

Il est possible de créer une liste d'objets à partir d'un modèle à l'aide du CLI ou, si un modèle a été téléchargé dans votre projet ou dans la bibliothèque globale de modèles, à l'aide de la console Web.

3.1.2. Chargement d'un modèle

Lorsque vous avez un fichier JSON ou YAML qui définit un modèle, vous pouvez télécharger le modèle vers des projets à l'aide du CLI. Cela permet d'enregistrer le modèle au projet pour une utilisation répétée par tout utilisateur ayant un accès approprié à ce projet. Des instructions sur la rédaction de vos propres modèles sont fournies plus tard dans ce sujet.

Procédure

- Envoyez un modèle en utilisant l'une des méthodes suivantes:
  - Envoyez un modèle dans la bibliothèque de modèles de votre projet actuel, passez le fichier JSON ou YAML avec la commande suivante:

    ```
    $ oc create -f <filename>
    ```

  - Envoyez un modèle vers un autre projet en utilisant l'option -n avec le nom du projet:

    ```
    $ oc create -f <filename> -n <project>
    ```

Le modèle est maintenant disponible pour la sélection à l'aide de la console Web ou du CLI.

3.1.3. Créer une application à l'aide de la console Web

Il est possible d'utiliser la console Web pour créer une application à partir d'un modèle.

Procédure

1. Choisissez Développeur dans le sélecteur de contexte en haut du menu de navigation de la console Web.
2. Dans le projet souhaité, cliquez sur +Ajouter
3. Cliquez sur Tous les services dans la tuile du catalogue des développeurs.

4. Cliquez sur Builder Images sous Type pour voir les images du constructeur disponibles.

NOTE

Les balises de flux d'images qui ont la balise constructeur listée dans leurs annotations apparaissent dans cette liste, comme démontré ici:

```
kind: "ImageStream"
apiVersion: "image.openshift.io/v1"
metadata:
  name: "ruby"
  creationTimestamp: null
spec:
# ...
  tags:
    - name: "2.6"
      annotations:
        description: "Build and run Ruby 2.6 applications"
        iconClass: "icon-ruby"
        tags: "builder,ruby" 1
        supports: "ruby:2.6,ruby"
        version: "2.6"
# ...
```

1 Inclure le constructeur ici s'assure que cette balise de flux d'images apparaît dans la console Web en tant que constructeur.

5. Modifiez les paramètres dans le nouvel écran de l'application pour configurer les objets pour prendre en charge votre application.

3.1.4. Création d'objets à partir de modèles en utilisant le CLI

Il est possible d'utiliser le CLI pour traiter les modèles et utiliser la configuration générée pour créer des objets.

3.1.4.1. Ajout d'étiquettes

Les étiquettes sont utilisées pour gérer et organiser des objets générés, tels que des pods. Les étiquettes spécifiées dans le modèle sont appliquées à chaque objet généré à partir du modèle.

Procédure

- Ajouter des étiquettes dans le modèle à partir de la ligne de commande:

```
$ oc process -f <filename> -l name=otherLabel
```

3.1.4.2. Liste des paramètres

La liste des paramètres que vous pouvez remplacer sont listées dans la section paramètres du modèle.

Procédure

1. Il est possible de répertorier les paramètres avec le CLI en utilisant la commande suivante et en spécifiant le fichier à utiliser:

```
$ oc process --parameters -f <filename>
```

Alternativement, si le modèle est déjà téléchargé:

```
$ oc process --parameters -n <project> <template_name>
```

À titre d'exemple, ce qui suit montre la sortie lors de la liste des paramètres de l'un des modèles de démarrage rapide dans le projet openshift par défaut:

```
$ oc process --parameters -n openshift rails-postgresql-example
```

Exemple de sortie

```
NAME                     DESCRIPTION
GENERATOR          VALUE
SOURCE_REPOSITORY_URL       The URL of the repository with your application source
code                                    https://github.com/sclorg/rails-ex.git
SOURCE_REPOSITORY_REF       Set this to a branch name, tag or other ref of your
repository if you are not using the default branch
CONTEXT_DIR                 Set this to the relative path to your project if it is not in the root of
your repository
APPLICATION_DOMAIN          The exposed hostname that will route to the Rails service
rails-postgresql-example.openshiftapps.com
GITHUB_WEBHOOK_SECRET       A secret string used to configure the GitHub webhook
expression          [a-zA-Z0-9]{40}
SECRET_KEY_BASE             Your secret key for verifying the integrity of signed cookies
expression          [a-z0-9]{127}
APPLICATION_USER            The application user that is used within the sample application
to authorize access on pages                    openshift
APPLICATION_PASSWORD        The application password that is used within the sample
application to authorize access on pages                    secret
DATABASE_SERVICE_NAME       Database service name
postgresql
POSTGRESQL_USER             database username
expression          user[A-Z0-9]{3}
POSTGRESQL_PASSWORD         database password
expression          [a-zA-Z0-9]{8}
POSTGRESQL_DATABASE         database name
root
POSTGRESQL_MAX_CONNECTIONS   database max connections
10
POSTGRESQL_SHARED_BUFFERS    database shared buffers
12MB
```

La sortie identifie plusieurs paramètres qui sont générés avec un générateur d'expression régulier lorsque le modèle est traité.

3.1.4.3. Générer une liste d'objets

En utilisant le CLI, vous pouvez traiter un fichier définissant un modèle pour retourner la liste des objets à la sortie standard.

Procédure

1. Le processus d'un fichier définissant un modèle pour retourner la liste des objets à la sortie standard:

```
$ oc process -f <filename>
```

   Alternativement, si le modèle a déjà été téléchargé dans le projet actuel:

```
$ oc process <template_name>
```

2. Créez des objets à partir d'un modèle en traitant le modèle et en comblant la sortie pour créer:

```
$ oc process -f <filename> | oc create -f -
```

   Alternativement, si le modèle a déjà été téléchargé dans le projet actuel:

```
$ oc process <template> | oc create -f -
```

3. Il est possible de remplacer les valeurs de paramètres définies dans le fichier en ajoutant l'option -p pour chaque paire <nom>=<valeur> que vous souhaitez remplacer. La référence de paramètre apparaît dans n'importe quel champ de texte à l'intérieur des éléments de modèle. Dans les paramètres POSTGRESQL_USER et POSTGRESQL_DATABASE suivants, les paramètres POSTGRESQL_DATABASE d'un modèle sont dépassés pour produire une configuration avec des variables d'environnement personnalisées:

   a. Création d'une liste d'objets à partir d'un modèle

```
$ oc process -f my-rails-postgresql \
    -p POSTGRESQL_USER=bob \
    -p POSTGRESQL_DATABASE=mydatabase
```

   b. Le fichier JSON peut être redirigé vers un fichier ou appliqué directement sans télécharger le modèle en supprimant la sortie traitée vers la commande oc create:

```
$ oc process -f my-rails-postgresql \
    -p POSTGRESQL_USER=bob \
    -p POSTGRESQL_DATABASE=mydatabase \
    | oc create -f -
```

   c. Lorsque vous disposez d'un grand nombre de paramètres, vous pouvez les stocker dans un fichier, puis passer ce fichier au processus oc:

```
$ cat postgres.env
POSTGRESQL_USER=bob
POSTGRESQL_DATABASE=mydatabase
```

```
$ oc process -f my-rails-postgresql --param-file=postgres.env
```

   d. Il est également possible de lire l'environnement à partir de l'entrée standard en utilisant "-" comme argument pour --param-file:

```
$ sed s/bob/alice/ postgres.env | oc process -f my-rails-postgresql --param-file=-
```

-

3.1.5. La modification des modèles téléchargés

Il est possible d'éditer un modèle qui a déjà été téléchargé sur votre projet.

Procédure

- De modifier un modèle qui a déjà été téléchargé:

```
$ oc edit template <template>
```

3.1.6. Écrire des modèles

Il est possible de définir de nouveaux modèles pour faciliter la recréation de tous les objets de votre application. Le modèle définit les objets qu'il crée ainsi que certaines métadonnées pour guider la création de ces objets.

Ce qui suit est un exemple d'une définition simple d'objet modèle (YAML):

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  name: redis-template
  annotations:
    description: "Description"
    iconClass: "icon-redis"
    tags: "database,nosql"
objects:
- apiVersion: v1
  kind: Pod
  metadata:
    name: redis-master
  spec:
    containers:
    - env:
      - name: REDIS_PASSWORD
        value: ${REDIS_PASSWORD}
      image: dockerfile/redis
      name: master
      ports:
      - containerPort: 6379
        protocol: TCP
parameters:
- description: Password used for Redis authentication
  from: '[A-Z0-9]{8}'
  generate: expression
  name: REDIS_PASSWORD
labels:
  redis: master
```

3.1.6.1. Écrire la description du modèle

La description du modèle vous informe de ce que le modèle fait et vous aide à le trouver lors de la recherche dans la console Web. Des métadonnées supplémentaires au-delà du nom du modèle sont

facultatives, mais utiles à avoir. En plus des informations descriptives générales, les métadonnées comprennent également un ensemble de balises. Les balises utiles incluent le nom du langage auquel le modèle est lié par exemple, Java, PHP, Ruby, etc.

Ce qui suit est un exemple de métadonnées de description de modèle:

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: cakephp-mysql-example 1
  annotations:
    openshift.io/display-name: "CakePHP MySQL Example (Ephemeral)" 2
    description: >-
      An example CakePHP application with a MySQL database. For more information
      about using this template, including OpenShift considerations, see
      https://github.com/sclorg/cakephp-ex/blob/master/README.md.


      WARNING: Any data stored will be lost upon pod destruction. Only use this
      template for testing." 3
    openshift.io/long-description: >-
      This template defines resources needed to develop a CakePHP application,
      including a build configuration, application DeploymentConfig, and
      database DeploymentConfig.  The database is stored in
      non-persistent storage, so this configuration should be used for
      experimental purposes only. 4
    tags: "quickstart,php,cakephp" 5
    iconClass: icon-php 6
    openshift.io/provider-display-name: "Red Hat, Inc." 7
    openshift.io/documentation-url: "https://github.com/sclorg/cakephp-ex" 8
    openshift.io/support-url: "https://access.redhat.com" 9
message: "Your admin credentials are ${ADMIN_USERNAME}:${ADMIN_PASSWORD}" 10
```

1. Le nom unique du modèle.
2. Bref, nom convivial, qui peut être utilisé par les interfaces utilisateur.
3. Description du modèle. Incluez suffisamment de détails pour que les utilisateurs comprennent ce qui est déployé et toutes les mises en garde qu'ils doivent savoir avant de le déployer. Il devrait également fournir des liens vers des informations supplémentaires, comme un fichier README. Les nouvelles lignes peuvent être incluses pour créer des paragraphes.
4. Description de modèle supplémentaire. Cela peut être affiché par le catalogue de services, par exemple.
5. Balises à associer au modèle pour la recherche et le regroupement. Ajoutez des tags qui l'incluent dans l'une des catégories de catalogue fournies. Consultez l'id et la catégorieAliases dans CATALOG_CATEGORIES dans le fichier constants de la console.
6. Icône à afficher avec votre modèle dans la console Web.

Exemple 3.1. Icônes disponibles

- **icône-3scale**

- **icône-aerogear**
- **icône-amq**
- **icône-angularjs**
- **icône-ansible**
- **icône-apache**
- **icône-beaker**
- **icône-camel**
- **icône-capedwarf**
- **icône-cassandra**
- **icône-catalog-icon**
- **icône-clojure**
- **icône-codeigniter**
- **icône-cordova**
- **icône-datagrid**
- **icône-datavirt**
- **icône-debian**
- **icône-decisionserver**
- **icône-django**
- **icône-dotnet**
- **icône-drupal**
- **icône-eap**
- **icône-élastique**
- **icône-erlang**
- **icône-fedora**
- **icône-freebsd**
- **icône-git**
- **icône-github**
- **icône-gitlab**
- **icône-verre de poisson**

- **icône-go-gopher**
- **icône-golang**
- **icône-grails**
- **icône-hadoop**
- **icône-haproxy**
- **icône-helm**
- **icône-infinispan**
- **icône-jboss**
- **icône-jenkins**
- **icône-jetée**
- **icône-joomla**
- **icône-jruby**
- **icône-js**
- **icône-knative**
- **icône-kubevirt**
- **icône-laravel**
- **icône-charge-balanceur**
- **icône-mariadb**
- **icône-mediawiki**
- **icône-memcached**
- **icône-mongodb**
- **icône-mssql**
- **icône-mysql-base de données**
- **icône-nginx**
- **icône-nodejs**
- **icône-openjdk**
- **icône-openliberty**
- **icône-openshift**
- **icône-openstack**

- **icône-autre-linux**
- **icône-autre-inconnu**
- **icône-perl**
- **icône-phalcon**
- **icône-php**
- **jeu d'icônes**
- **icônepostgresql**
- **icône-processserver**
- **icône-python**
- **icône-quarkus**
- **icône-rabbitmq**
- **icône-rails**
- **icône-redhat**
- **icône-redis**
- **icône-rh-intégration**
- **icône-rh-spring-boot**
- **icône-rh-tomcat**
- **icône-ruby**
- **icône-scala**
- **icône-serverlessfx**
- **icône-shadowman**
- **icône-spring-boot**
- **icône-spring**
- **icône-sso**
- **icône-stackoverflow**
- **icône-suse**
- **icône-symfony**
- **icône-tomcat**
- **icône-ubuntu**

- **icône-vertx**
- **icône-wildfly**
- **icône-fenêtres**
- **icône-motpress**
- **icône-xamarin**
- **icône-zend**

7 Le nom de la personne ou de l'organisation fournissant le modèle.

8 D'une URL faisant référence à d'autres documents pour le modèle.

9 D'une URL où le support peut être obtenu pour le modèle.

10 C'est un message d'instruction qui s'affiche lorsque ce modèle est instancié. Ce champ devrait indiquer à l'utilisateur comment utiliser les ressources nouvellement créées. La substitution de paramètres est effectuée sur le message avant d'être affichée de sorte que les informations d'identification générées et d'autres paramètres puissent être inclus dans la sortie. Inclure des liens vers toute documentation suivante que les utilisateurs devraient suivre.

3.1.6.2. Écrire des étiquettes de modèle

Les modèles peuvent inclure un ensemble d'étiquettes. Ces étiquettes sont ajoutées à chaque objet créé lorsque le modèle est instancié. Définir une étiquette de cette façon permet aux utilisateurs de trouver et de gérer facilement tous les objets créés à partir d'un modèle particulier.

Ce qui suit est un exemple d'étiquettes d'objets de modèle:

```
kind: "Template"
apiVersion: "v1"
...
labels:
  template: "cakephp-mysql-example" 1
  app: "${NAME}" 2
```

1 Étiquette appliquée à tous les objets créés à partir de ce modèle.

2 Étiquette paramétrée qui est également appliquée à tous les objets créés à partir de ce modèle. L'expansion des paramètres est effectuée à la fois sur les clés d'étiquette et sur les valeurs.

3.1.6.3. Écrire des paramètres de modèle

Les paramètres permettent qu'une valeur soit fournie par vous ou générée lorsque le modèle est instancié. Ensuite, cette valeur est remplacée partout où le paramètre est référencé. Les références peuvent être définies dans n'importe quel champ dans le champ de liste d'objets. Ceci est utile pour générer des mots de passe aléatoires ou vous permettre de fournir un nom d'hôte ou une autre valeur spécifique à l'utilisateur qui est nécessaire pour personnaliser le modèle. Les paramètres peuvent être référencés de deux manières:

- En tant que valeur de chaîne en plaçant des valeurs dans le formulaire ${PARAMETER_NAME} dans n'importe quel champ de chaîne du modèle.

- En tant que valeur JSON ou YAML en plaçant des valeurs dans la forme ${PARAMETER_NAME}} à la place de n'importe quel champ dans le modèle.

Lors de l'utilisation de la syntaxe ${PARAMETER_NAME}, plusieurs références de paramètres peuvent être combinées dans un seul champ et la référence peut être intégrée dans des données fixes, telles que "http://${PARAMETER_1}${PARAMETER_2}". Les deux valeurs de paramètres sont substituées et la valeur résultante est une chaîne citée.

Lors de l'utilisation de la syntaxe ${{PARAMETER_NAME}} seule une seule référence de paramètre est autorisée et les caractères menant et traînant ne sont pas autorisés. La valeur résultante n'est pas citée à moins que, après la substitution, le résultat n'est pas un objet JSON valide. Lorsque le résultat n'est pas une valeur JSON valide, la valeur résultante est citée et traitée comme une chaîne standard.

Il peut être référencé plusieurs fois dans un modèle et il peut être référencé à l'aide des deux syntaxes de substitution au sein d'un seul modèle.

La valeur par défaut peut être fournie, qui est utilisée si vous ne fournissez pas une valeur différente:

Ce qui suit est un exemple de définition d'une valeur explicite comme valeur par défaut:

```
parameters:
  - name: USERNAME
    description: "The user name for Joe"
    value: joe
```

Les valeurs de paramètre peuvent également être générées en fonction des règles spécifiées dans la définition des paramètres, par exemple en générant une valeur de paramètre:

```
parameters:
  - name: PASSWORD
    description: "The random user password"
    generate: expression
    from: "[a-zA-Z0-9]{12}"
```

Dans l'exemple précédent, le traitement génère un mot de passe aléatoire de 12 caractères, composé de toutes les lettres et chiffres de l'alphabet majuscule et minuscule.

La syntaxe disponible n'est pas une syntaxe d'expression régulière complète. Cependant, vous pouvez utiliser \w, \d, \a et \A modificateurs:

- [\W]{10} produit 10 caractères de l'alphabet, des nombres et des accents. Ceci suit la norme PCRE et est égal à [a-zA-Z0-9_]{10}.

- [\d]{10} produit 10 nombres. Ceci est égal à [0-9]{10}.

- [\a]{10} produit 10 caractères alphabétiques. Ceci est égal à [a-zA-Z]{10}.

- [\A]{10} produit 10 caractères de ponctuation ou de symbole. Ceci est égal à [~!@#$%\^&*()\-_+={}\\\\|<,>.?/"';:']{10}.

NOTE

En fonction de si le modèle est écrit dans YAML ou JSON, et le type de chaîne dans laquelle le modificateur est intégré, vous devrez peut-être échapper au backslash avec un second backslash. Les exemples suivants sont équivalents:

Exemple de modèle YAML avec un modificateur

```
parameters:
- name: singlequoted_example
  generate: expression
  from: '[\A]{10}'
- name: doublequoted_example
  generate: expression
  from: "[\\A]{10}"
```

Exemple de modèle JSON avec un modificateur

```
{
  "parameters": [
    {
      "name": "json_example",
      "generate": "expression",
      "from": "[\\A]{10}"
    }
  ]
}
```

En voici un exemple d'un modèle complet avec des définitions de paramètres et des références:

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: my-template
objects:
  - kind: BuildConfig
    apiVersion: build.openshift.io/v1
    metadata:
      name: cakephp-mysql-example
      annotations:
        description: Defines how to build the application
    spec:
      source:
        type: Git
        git:
          uri: "${SOURCE_REPOSITORY_URL}" (1)
          ref: "${SOURCE_REPOSITORY_REF}"
        contextDir: "${CONTEXT_DIR}"
  - kind: DeploymentConfig
    apiVersion: apps.openshift.io/v1
    metadata:
      name: frontend
    spec:
      replicas: "${{REPLICA_COUNT}}" (2)
```

```
parameters:
  - name: SOURCE_REPOSITORY_URL 3
    displayName: Source Repository URL 4
    description: The URL of the repository with your application source code 5
    value: https://github.com/sclorg/cakephp-ex.git 6
    required: true 7
  - name: GITHUB_WEBHOOK_SECRET
    description: A secret string used to configure the GitHub webhook
    generate: expression 8
    from: "[a-zA-Z0-9]{40}" 9
  - name: REPLICA_COUNT
    description: Number of replicas to run
    value: "2"
    required: true
message: "... The GitHub webhook secret is ${GITHUB_WEBHOOK_SECRET} ..." 10
```

1. Cette valeur est remplacée par la valeur du paramètre SOURCE_REPOSITORY_URL lorsque le modèle est instancié.
2. Cette valeur est remplacée par la valeur non citée du paramètre REPLICA_COUNT lorsque le modèle est instancié.
3. Le nom du paramètre. Cette valeur est utilisée pour référencer le paramètre dans le modèle.
4. Le nom convivial du paramètre. Ceci est affiché pour les utilisateurs.
5. Description du paramètre. Fournir des informations plus détaillées aux fins du paramètre, y compris toute contrainte sur la valeur attendue. Les descriptions doivent utiliser des phrases complètes pour suivre les normes de texte de la console. Il ne s'agit pas d'un duplicata du nom d'affichage.
6. La valeur par défaut du paramètre qui est utilisé si vous ne remplacez pas la valeur lors de l'instanciation du modèle. Évitez d'utiliser des valeurs par défaut pour des choses comme les mots de passe, utilisez plutôt les paramètres générés en combinaison avec des secrets.
7. Indique que ce paramètre est requis, ce qui signifie que vous ne pouvez pas le remplacer par une valeur vide. Dans le cas où le paramètre ne fournit pas de valeur par défaut ou générée, vous devez fournir une valeur.
8. C'est un paramètre qui a sa valeur générée.
9. L'entrée au générateur. Dans ce cas, le générateur produit une valeur alphanumérique de 40 caractères incluant des caractères majuscules et minuscules.
10. Les paramètres peuvent être inclus dans le message de modèle. Cela vous informe des valeurs générées.

3.1.6.4. Écrire la liste d'objets du modèle

La partie principale du modèle est la liste des objets qui est créé lorsque le modèle est instancié. Cela peut être n'importe quel objet API valide, tel qu'une configuration de build, une configuration de déploiement ou un service. L'objet est créé exactement comme défini ici, avec des valeurs de paramètres substituées avant la création. La définition de ces objets peut référencer les paramètres définis précédemment.

Ce qui suit est un exemple d'une liste d'objets:

```
kind: "Template"
apiVersion: "v1"
metadata:
  name: my-template
objects:
  - kind: "Service" 1
    apiVersion: "v1"
    metadata:
      name: "cakephp-mysql-example"
      annotations:
        description: "Exposes and load balances the application pods"
    spec:
      ports:
        - name: "web"
          port: 8080
          targetPort: 8080
      selector:
        name: "cakephp-mysql-example"
```

1 La définition d'un service, qui est créé par ce modèle.

NOTE

Lorsqu'une métadonnées de définition d'objet comprend une valeur de champ d'espace de noms fixe, le champ est retiré de la définition lors de l'instanciation du modèle. Lorsque le champ namespace contient une référence de paramètre, une substitution de paramètre normale est effectuée et l'objet est créé dans n'importe quel espace de noms auquel le paramètre a résolu la valeur, en supposant que l'utilisateur ait l'autorisation de créer des objets dans cet espace de noms.

3.1.6.5. Marquer un modèle comme liable

Le Template Service Broker annonce un service dans son catalogue pour chaque objet de modèle dont il est conscient. Chacun de ces services est annoncé par défaut comme étant liable, ce qui signifie qu'un utilisateur final est autorisé à se lier contre le service fourni.

Procédure

Les auteurs de modèles peuvent empêcher les utilisateurs finaux de se lier contre les services fournis à partir d'un modèle donné.

- Empêcher l'utilisateur final de se lier contre les services fournis à partir d'un modèle donné en ajoutant l'annotation template.openshift.io/bindable: "faux" au modèle.

3.1.6.6. Exposer les champs d'objets du modèle

Les auteurs de modèles peuvent indiquer que des champs d'objets particuliers dans un modèle doivent être exposés. Le Template Service Broker reconnaît les champs exposés sur les objets ConfigMap, Secret, Service et Route, et retourne les valeurs des champs exposés lorsqu'un utilisateur lie un service soutenu par le courtier.

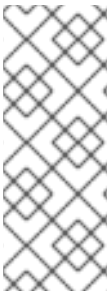
Afin d'exposer un ou plusieurs champs d'un objet, ajoutez des annotations préfixées par template.openshift.io/expose- ou template.openshift.io/base64-expose- à l'objet dans le modèle.

Chaque clé d'annotation, avec son préfixe supprimé, est passée à travers pour devenir une clé dans une réponse de liaison.

Chaque valeur d'annotation est une expression Kubernetes JSONPath, qui est résolue au moment de la liaison pour indiquer le champ objet dont la valeur doit être retournée dans la réponse de liaison.

NOTE

Les paires clé-valeur de réponse peuvent être utilisées dans d'autres parties du système comme variables d'environnement. Il est donc recommandé que chaque clé d'annotation avec son préfixe soit un nom de variable d'environnement valide - commençant par un caractère A-Z, a-z, ou _, et étant suivi de zéro ou plus caractères A-Z, a-z, 0-9, ou _.



NOTE

À moins de s'échapper avec un backslash, l'implémentation JSONPath de Kubernetes interprète des caractères tels que ., @, et d'autres comme métacaractères, quelle que soit leur position dans l'expression. Donc, par exemple, pour se référer à un datum ConfigMap nommé my.key, l'expression JSONPath requise serait {.data['my\.key']}. En fonction de la façon dont l'expression JSONPath est alors écrite en YAML, un backslash supplémentaire peut être requis, par exemple "{.data['my\\\.key']}.

Ce qui suit est un exemple de champs d'objets différents exposés:

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: my-template
objects:
- kind: ConfigMap
  apiVersion: v1
  metadata:
    name: my-template-config
    annotations:
      template.openshift.io/expose-username: "{.data['my\\.username']}"
  data:
    my.username: foo
- kind: Secret
  apiVersion: v1
  metadata:
    name: my-template-config-secret
    annotations:
      template.openshift.io/base64-expose-password: "{.data['password']}"
  stringData:
    password: <password>
- kind: Service
  apiVersion: v1
  metadata:
    name: my-template-service
    annotations:
      template.openshift.io/expose-service_ip_port: "{.spec.clusterIP}:{.spec.ports[?
(.name==\"web\")].port}"
```

```
  spec:
    ports:
    - name: "web"
      port: 8080
- kind: Route
  apiVersion: route.openshift.io/v1
  metadata:
    name: my-template-route
    annotations:
      template.openshift.io/expose-uri: "http://{.spec.host}{.spec.path}"
  spec:
    path: mypath
```

Exemple de réponse à une opération de liaison étant donné le modèle partiel ci-dessus:

```
{
  "credentials": {
    "username": "foo",
    "password": "YmFy",
    "service_ip_port": "172.30.12.34:8080",
    "uri": "http://route-test.router.default.svc.cluster.local/mypath"
  }
}
```

Procédure

- L'annotation template.openshift.io/expose- renvoie la valeur du champ en tant que chaîne de caractères. C'est pratique, bien qu'il ne traite pas de données binaires arbitraires.
- Lorsque vous souhaitez retourner des données binaires, utilisez l'annotation template.openshift.io/base64-expose- pour coder les données avant qu'elles ne soient retournées.

3.1.6.7. En attente d'un modèle de préparation

Les auteurs de modèles peuvent indiquer que certains objets d'un modèle doivent être attendus avant qu'une instanciation de modèle par le catalogue de service, Template Service Broker ou l'API TemplateInstance soit considérée comme complète.

Afin d'utiliser cette fonctionnalité, marquez un ou plusieurs objets de type Build, BuildConfig, Deployment, DeploymentConfig, Job ou StatefulSet dans un modèle avec l'annotation suivante:

```
"template.alpha.openshift.io/wait-for-ready": "true"
```

L'instanciation du modèle n'est pas terminée tant que tous les objets marqués du rapport d'annotation ne sont pas prêts. De même, si l'un des rapports d'objets annotés a échoué, ou si le modèle ne parvient pas à devenir prêt dans un délai fixe d'une heure, l'instanciation du modèle échoue.

Aux fins de l'instanciation, la préparation et l'échec de chaque type d'objet sont définis comme suit:

Je suis gentille.	L'état de préparation	Échec

Je suis gentille.	L'état de préparation	Échec
Construire	La phase des rapports d'objets est terminée.	Les rapports d'objets ont été annulés, une erreur ou un échec.
BuildConfig	Les derniers rapports d'objets associés sont terminés.	Les derniers rapports d'objets de construction associés ont été annulés, erreurs ou échoués.
Déploiement	L'objet signale un nouvel ensemble de répliques et un déploiement disponible. Cela honore les sondes de préparation définies sur l'objet.	L'état d'avancement des rapports d'objet est faux.
DéploiementConfig	L'objet signale le nouveau contrôleur de réplication et le déploiement disponible. Cela honore les sondes de préparation définies sur l'objet.	L'état d'avancement des rapports d'objet est faux.
Emploi	Les rapports d'objet sont terminés.	L'objet signale qu'un ou plusieurs échecs ont eu lieu.
ÉtatfulSet	L'objet signale toutes les répliques prêtes. Cela honore les sondes de préparation définies sur l'objet.	Ce n'est pas applicable.

Ce qui suit est un exemple d'extrait de modèle, qui utilise l'annotation prête à attendre. D'autres exemples peuvent être trouvés dans le Red Hat OpenShift Service sur les modèles de démarrage rapide AWS.

```
kind: Template
apiVersion: template.openshift.io/v1
metadata:
  name: my-template
objects:
- kind: BuildConfig
  apiVersion: build.openshift.io/v1
  metadata:
    name: ...
    annotations:
      # wait-for-ready used on BuildConfig ensures that template instantiation
      # will fail immediately if build fails
      template.alpha.openshift.io/wait-for-ready: "true"
  spec:
    ...
- kind: DeploymentConfig
  apiVersion: apps.openshift.io/v1
  metadata:
    name: ...
    annotations:
      template.alpha.openshift.io/wait-for-ready: "true"
  spec:
```

```
    ...
- kind: Service
  apiVersion: v1
  metadata:
    name: ...
  spec:
    ...
```

Autres recommandations

- Définissez la mémoire, le CPU et les tailles par défaut de stockage pour vous assurer que votre application dispose de suffisamment de ressources pour fonctionner en douceur.
- Évitez de faire référence à la dernière balise à partir d'images si cette balise est utilisée dans les versions principales. Cela peut causer des applications en cours d'exécution à casser lorsque de nouvelles images sont poussées vers cette balise.
- Après le déploiement du modèle, un bon modèle construit et déploie proprement sans nécessiter de modifications.

3.1.6.8. Créer un modèle à partir d'objets existants

Au lieu d'écrire un modèle entier à partir de zéro, vous pouvez exporter des objets existants de votre projet dans le formulaire YAML, puis modifier le YAML à partir de là en ajoutant des paramètres et d'autres personnalisations en tant que formulaire de modèle.

Procédure

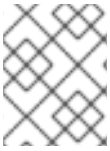
- Exporter des objets dans un projet sous forme YAML:

  ```
  $ oc get -o yaml all > <yaml_filename>
  ```

  Il est également possible de substituer un type de ressource particulier ou plusieurs ressources au lieu de toutes. Exécutez oc get -h pour plus d'exemples.

  Les types d'objets inclus dans oc get -o yaml tous sont:

  - **BuildConfig**
  - **Construire**
  - **DéploiementConfig**
  - **ImageStream**
  - **La pod**
  - **Contrôleur de réplication**
  - **Itinéraire**
  - **Le service**



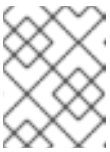
NOTE

L'utilisation de tous les alias n'est pas recommandée car le contenu peut varier selon différents clusters et versions. Au lieu de cela, spécifiez toutes les ressources requises.

3.2. CRÉER DES APPLICATIONS EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

La perspective Développeur dans la console Web vous fournit les options suivantes de la vue +Ajouter pour créer des applications et des services associés et les déployer sur Red Hat OpenShift Service sur AWS:

- Démarrage des ressources: Utilisez ces ressources pour vous aider à démarrer avec la console de développement. Il est possible de cacher l'en-tête à l'aide du menu Options.
    - Création d'applications à l'aide d'échantillons: Utilisez des échantillons de code existants pour commencer avec la création d'applications sur le service OpenShift Red Hat sur AWS.
    - Construire avec la documentation guidée: Suivez la documentation guidée pour créer des applications et vous familiariser avec les concepts clés et les terminologies.
    - Explorez les nouvelles fonctionnalités du développeur : explorez les nouvelles fonctionnalités et ressources dans la perspective du développeur.
- Catalogue des développeurs: Découvrez le catalogue des développeurs pour sélectionner les applications, les services ou les sources nécessaires pour créer des images, puis l'ajouter à votre projet.
    - All Services: Parcourez le catalogue pour découvrir les services de Red Hat OpenShift Service sur AWS.
    - Base de données: Sélectionnez le service de base de données requis et ajoutez-le à votre application.
    - L'opérateur Backed: Sélectionnez et déployez le service géré par l'opérateur requis.
    - Graphique de helm: Sélectionnez le graphique Helm requis pour simplifier le déploiement des applications et des services.
    - Devfile: Sélectionnez un devfile dans le registre Devfile pour définir de manière déclarative un environnement de développement.
    - Evénement Source: Sélectionnez une source d'événement pour enregistrer l'intérêt pour une classe d'événements à partir d'un système particulier.



NOTE

L'option Services gérés est également disponible si l'opérateur RHOAS est installé.

- Dépôt Git: Importez une base de code existante, Devfile ou Dockerfile à partir de votre dépôt Git en utilisant respectivement les options From Git, From Devfile ou From Dockerfile pour construire et déployer une application sur Red Hat OpenShift Service sur AWS.
- Images conteneur: Utilisez des images existantes à partir d'un flux d'images ou d'un registre pour les déployer sur le service Red Hat OpenShift sur AWS.

- Pipelines: Utilisez le pipeline Tekton pour créer des pipelines CI/CD pour votre processus de livraison de logiciels sur le service OpenShift Red Hat sur AWS.
- Découvrez les options Serverless pour créer, construire et déployer des applications apatrides et sans serveur sur le service Red Hat OpenShift sur AWS.
    - Canal: Créez un canal Knative pour créer une couche de transmission d'événements et de persistance avec des implémentations en mémoire et fiables.
- Échantillons : explorez les exemples d'applications disponibles pour créer, construire et déployer rapidement une application.
- Démarrage rapide : explorez les options de démarrage rapide pour créer, importer et exécuter des applications avec des instructions et des tâches étape par étape.
- De la machine locale: explorez la tuile de la machine locale pour importer ou télécharger des fichiers sur votre machine locale pour créer et déployer facilement des applications.
    - Importer YAML : Téléchargez un fichier YAML pour créer et définir des ressources pour la création et le déploiement d'applications.
    - Envoyer un fichier JAR : Téléchargez un fichier JAR pour créer et déployer des applications Java.
- Le partage de mon projet : Utilisez cette option pour ajouter ou supprimer des utilisateurs à un projet et leur fournir des options d'accessibilité.
- Dépôts de graphiques helm: Utilisez cette option pour ajouter des référentiels Helm Chart dans un espace de noms.
- La ré-commandation des ressources : Utilisez ces ressources pour réorganiser les ressources épinglées ajoutées à votre volet de navigation. L'icône glisser-déposer s'affiche sur le côté gauche de la ressource épinglée lorsque vous le survolez dans le volet de navigation. La ressource traînée ne peut être supprimée que dans la section où elle réside.

Il est à noter que l'option Pipelines n'est affichée que lorsque l'opérateur de pipelines OpenShift est installé.

3.2.1. Conditions préalables

Créer des applications en utilisant la perspective Développeur s'assure que:

- Connectez-vous à la console web.

3.2.2. Créer des exemples d'applications

Les exemples d'applications peuvent être utilisés dans le flux +Ajouter de la perspective Développeur pour créer, construire et déployer rapidement des applications.

Conditions préalables

- Connectez-vous au service Red Hat OpenShift sur la console web AWS et vous êtes dans la perspective Développeur.

Procédure

1. Dans la vue +Ajouter, cliquez sur la tuile Échantillons pour voir la page Échantillons.

2. Dans la page Échantillons, sélectionnez l'une des applications types disponibles pour voir le formulaire Créer une demande d'échantillon.

3. Dans le formulaire de demande d'échantillon:

   - Dans le champ Nom, le nom de déploiement est affiché par défaut. Il est possible de modifier ce nom au besoin.

   - Dans la version d'image du constructeur, une image de constructeur est sélectionnée par défaut. Cette version d'image peut être modifiée à l'aide de la liste déroulante de la version déroulante de Builder Image.

   - L'URL du référentiel Git est ajoutée par défaut.

4. Cliquez sur Créer pour créer l'exemple d'application. L'état de construction de l'application de l'échantillon est affiché sur la vue Topology. Après la création de l'exemple d'application, vous pouvez voir le déploiement ajouté à l'application.

3.2.3. Création d'applications en utilisant Quick Starts

La page Démarrage rapide vous montre comment créer, importer et exécuter des applications sur Red Hat OpenShift Service sur AWS, avec des instructions et des tâches étape par étape.

Conditions préalables

- Connectez-vous au service Red Hat OpenShift sur la console web AWS et vous êtes dans la perspective Développeur.

Procédure

1. Dans la vue +Ajouter, cliquez sur le lien Démarrer les ressources → Construire avec la documentation guidée → Voir tous les liens de démarrage rapide pour afficher la page Démarrage rapide.

2. Dans la page Démarrage rapide, cliquez sur la tuile pour le démarrage rapide que vous souhaitez utiliser.

3. Cliquez sur Démarrer pour commencer le démarrage rapide.

4. Effectuez les étapes qui sont affichées.

3.2.4. Importer une base de code à partir de Git pour créer une application

La perspective Développeur peut être utilisée pour créer, construire et déployer une application sur Red Hat OpenShift Service sur AWS à l'aide d'une base de code existante dans GitHub.

La procédure suivante vous guide à travers l'option From Git dans la perspective Développeur pour créer une application.

Procédure

1. Dans la vue +Ajouter, cliquez sur À partir de Git dans la tuile Git Repository pour voir le formulaire Importer à partir de git.

2. Dans la section Git, entrez l'URL du référentiel Git pour la base de code que vous souhaitez utiliser pour créer une application. Entrez par exemple l'URL de cet exemple d'application Node.js https://github.com/sclorg/nodejs-ex. L'URL est ensuite validée.

3. Facultatif: Vous pouvez cliquer sur Afficher les options avancées de Git pour ajouter des détails tels que:

    - Git Référence au point de code d'une branche, d'une balise ou d'un engagement spécifique pour être utilisé pour construire l'application.

    - Context Dir pour spécifier le sous-répertoire du code source de l'application que vous souhaitez utiliser pour construire l'application.

    - Créer un nom secret avec des informations d'identification pour tirer votre code source d'un référentiel privé.

4. Facultatif : Vous pouvez importer un fichier Devfile, un Dockerfile, une image Builder ou une fonction sans serveur via votre référentiel Git pour personnaliser davantage votre déploiement.

    - Lorsque votre dépôt Git contient un Devfile, un Dockerfile, une image Builder ou un func.yaml, il est automatiquement détecté et peuplé sur les champs de chemin respectifs.

    - Lorsqu'un Devfile, un Dockerfile ou une image Builder sont détectés dans le même référentiel, le Devfile est sélectionné par défaut.

    - Lorsque func.yaml est détecté dans le référentiel Git, la stratégie d'importation change de fonction sans serveur.

    - Alternativement, vous pouvez créer une fonction sans serveur en cliquant sur Créer la fonction Serverless dans la vue +Ajouter à l'aide de l'URL du référentiel Git.

    - Afin d'éditer le type d'importation de fichier et de sélectionner une stratégie différente, cliquez sur Modifier la stratégie d'importation.

    - Lorsque plusieurs Devfiles, Dockerfiles, ou un Builder Images sont détectés, pour importer une instance spécifique, spécifiez les chemins respectifs par rapport au répertoire contextuel.

5. Après la validation de l'URL Git, l'image de constructeur recommandée est sélectionnée et marquée d'une étoile. Lorsque l'image du constructeur n'est pas détectée automatiquement, sélectionnez une image de constructeur. Dans le cas de l'URL https://github.com/sclorg/nodejs-ex Git, l'image du constructeur Node.js est sélectionnée par défaut.

    a. Facultatif : Utilisez le déroulant de la version déroulante de Builder Image pour spécifier une version.

    b. Facultatif: Utilisez la stratégie Modifier l'importation pour sélectionner une stratégie différente.

    c. Facultatif: Pour l'image du constructeur Node.js, utilisez le champ de commande Exécuter pour remplacer la commande pour exécuter l'application.

6. Dans la section générale:

    a. Dans le champ Application, entrez un nom unique pour le groupement d'applications, par exemple myapp. Assurez-vous que le nom de l'application est unique dans un espace de noms.

b. Le champ Nom pour identifier les ressources créées pour cette application est automatiquement rempli en fonction de l'URL du référentiel Git s'il n'y a pas d'applications existantes. Dans le cas d'applications existantes, vous pouvez choisir de déployer le composant dans une application existante, de créer une nouvelle application ou de garder le composant non affecté.

NOTE

Le nom de la ressource doit être unique dans un espace de noms. Modifiez le nom de la ressource si vous obtenez une erreur.

7. Dans la section Ressources, sélectionnez:

   - Déploiement, pour créer une application dans le style Kubernetes.
   - Déploiement Config, pour créer un service Red Hat OpenShift sur l'application de style AWS.

8. Dans la section Pipelines, sélectionnez Ajouter un pipeline, puis cliquez sur Afficher la visualisation des pipelines pour voir le pipeline pour l'application. Le pipeline par défaut est sélectionné, mais vous pouvez choisir le pipeline que vous souhaitez dans la liste des pipelines disponibles pour l'application.

NOTE

La case à cocher Ajouter un pipeline est cochée et Configure PAC est sélectionnée par défaut si les critères suivants sont remplis:

- L'opérateur de pipeline est installé
- les pipelines-as-code sont activés
- le répertoire .Tekton est détecté dans le référentiel Git

9. Ajoutez un webhook à votre référentiel. Lorsque Configurez PAC est vérifié et que l'application GitHub est configurée, vous pouvez voir les options Utiliser l'application GitHub et configurer un webhook. Dans le cas où l'application GitHub n'est pas configurée, vous ne pouvez voir que l'option Configuration d'un webhook:

   a. Allez dans Paramètres → Webhooks et cliquez sur Ajouter webhook.

   b. Définissez l'URL Charge utile sur les pipelines en tant qu'URL publique du contrôleur de code.

   c. Choisissez le type de contenu comme application/json.

   d. Ajoutez un secret de webhook et notez-le dans un autre endroit. Avec Opensl installé sur votre machine locale, générez un secret aléatoire.

   e. Cliquez sur Laissez-moi sélectionner des événements individuels et sélectionnez ces événements: Commiter les commentaires, émettre des commentaires, Pull request et Pushes.

   f. Cliquez sur Ajouter webhook.

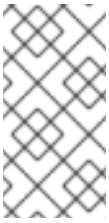
10. Facultatif : Dans la section Options avancées, le port cible et l'itinéraire Créer un itinéraire vers l'application sont sélectionnés par défaut afin que vous puissiez accéder à votre application à l'aide d'une URL accessible au public.
    Lorsque votre application n'expose pas ses données sur le port public par défaut, 80, effacer la case à cocher et définir le numéro de port cible que vous souhaitez exposer.

11. Facultatif: Vous pouvez utiliser les options avancées suivantes pour personnaliser davantage votre application:

    Le routage

    En cliquant sur le lien de routage, vous pouvez effectuer les actions suivantes:

    - Personnalisez le nom d'hôte pour l'itinéraire.
    - Indiquez le chemin que le routeur montre.
    - Choisissez le port cible pour le trafic dans la liste déroulante.
    - Assurez votre itinéraire en sélectionnant la case à cocher Route sécurisée. Choisissez le type de terminaison TLS requis et définissez une stratégie pour le trafic non sécurisé à partir des listes déroulantes respectives.

    NOTE

    Dans le cas des applications sans serveur, le service Knative gère toutes les options de routage ci-dessus. Cependant, vous pouvez personnaliser le port cible pour le trafic, si nécessaire. Lorsque le port cible n'est pas spécifié, le port par défaut de 8080 est utilisé.

    Bilans de santé

    Cliquez sur le lien Health Checks pour ajouter des sondes de préparation, de vie et de démarrage à votre application. L'ensemble des sondes ont des données par défaut prépopulées; vous pouvez ajouter les sondes avec les données par défaut ou les personnaliser au besoin.
    Afin de personnaliser les sondes de santé:

    - Cliquez sur Ajouter la sonde de préparation, si nécessaire, modifiez les paramètres pour vérifier si le conteneur est prêt à traiter les demandes et sélectionnez la marque de cocher pour ajouter la sonde.
    - Cliquez sur Ajouter une sonde de vie, si nécessaire, modifiez les paramètres pour vérifier si un conteneur est toujours en cours d'exécution et sélectionnez la coche pour ajouter la sonde.
    - Cliquez sur Ajouter une sonde de démarrage, si nécessaire, modifiez les paramètres pour vérifier si l'application dans le conteneur a commencé et sélectionnez la coche pour ajouter la sonde.
      Dans chacune des sondes, vous pouvez spécifier le type de requête - HTTP GET, Container Command ou TCP Socket, à partir de la liste déroulante. Le formulaire change selon le type de demande sélectionné. Ensuite, vous pouvez modifier les valeurs par défaut pour les autres paramètres, tels que les seuils de succès et de défaillance de la sonde, le nombre de secondes avant d'effectuer la première sonde après le démarrage du conteneur, la fréquence de la sonde et la valeur d'expiration.

    Construire la configuration et le déploiement

Cliquez sur les liens Configuration et déploiement pour voir les options de configuration respectives. Certaines options sont sélectionnées par défaut; vous pouvez les personnaliser davantage en ajoutant les déclencheurs et les variables d'environnement nécessaires.
Dans le cas des applications sans serveur, l'option Déploiement n'est pas affichée car la ressource de configuration Knative maintient l'état souhaité pour votre déploiement au lieu d'une ressource DeploymentConfig.

La mise à l'échelle

Cliquez sur le lien Scaling pour définir le nombre de pods ou d'instances de l'application que vous souhaitez déployer initialement.
Lorsque vous créez un déploiement sans serveur, vous pouvez également configurer les paramètres suivants:

- Les Pods min déterminent la limite inférieure pour le nombre de gousses qui doivent fonctionner à tout moment pour un service Knative. Ceci est également connu sous le nom de paramètre minScale.
- Les Max Pods déterminent la limite supérieure pour le nombre de gousses pouvant fonctionner à tout moment pour un service Knative. Ceci est également connu sous le nom de réglage maxScale.
- La cible de concurrence détermine le nombre de demandes simultanées souhaitées pour chaque instance de la demande à un moment donné.
- La limite de concurrence détermine la limite pour le nombre de demandes simultanées autorisées pour chaque instance de la demande à un moment donné.
- L'utilisation de la concurrence détermine le pourcentage de la limite de demandes simultanées qui doit être respectée avant que Knative n'évolue de nouveaux pods pour gérer le trafic supplémentaire.
- La fenêtre Autoscale définit la fenêtre de temps sur laquelle les métriques sont moyennes pour fournir une entrée pour les décisions de mise à l'échelle lorsque l'autoscaler n'est pas en mode panique. Le service est réduit à zéro si aucune demande n'est reçue pendant cette fenêtre. La durée par défaut de la fenêtre autoscale est 60s. Ceci est également connu sous le nom de fenêtre stable.

Limite de ressources

Cliquez sur le lien Limite de ressources pour définir la quantité de ressources CPU et mémoire qu'un conteneur est garanti ou autorisé à utiliser lors de l'exécution.

Étiquettes

Cliquez sur le lien Labels pour ajouter des étiquettes personnalisées à votre application.

12. Cliquez sur Créer pour créer l'application et une notification de succès est affichée. L'état de construction de l'application est affiché dans la vue Topology.

3.2.5. Créer des applications en déployant l'image de conteneur

Il est possible d'utiliser un registre d'images externe ou une balise de flux d'images à partir d'un registre interne pour déployer une application sur votre cluster.

Conditions préalables

- Connectez-vous au service Red Hat OpenShift sur la console web AWS et vous êtes dans la perspective Développeur.

Procédure

1. Dans la vue +Ajouter, cliquez sur Container images pour afficher la page Deploy Images.
2. Dans la section Image:
   a. Choisissez le nom de l'image dans le registre externe pour déployer une image à partir d'un registre public ou privé, ou sélectionnez Image Stream tag dans le registre interne pour déployer une image à partir d'un registre interne.
   b. Choisissez une icône pour votre image dans l'onglet de l'icône Runtime.
3. Dans la section générale:
   a. Dans le champ Nom de l'application, entrez un nom unique pour le groupement d'applications.
   b. Dans le champ Nom, entrez un nom unique pour identifier les ressources créées pour ce composant.
4. Dans la section Type de ressource, sélectionnez le type de ressource à générer:
   a. Choisissez Déploiement pour activer les mises à jour déclaratives pour les objets Pod et ReplicaSet.
   b. Choisissez DéploymentConfig pour définir le modèle d'un objet Pod et gérer le déploiement de nouvelles images et sources de configuration.
5. Cliquez sur **Create**. Il est possible d'afficher l'état de construction de l'application dans la vue Topology.

3.2.6. Déploiement d'une application Java en téléchargeant un fichier JAR

La perspective Développeur de la console Web vous permet de télécharger un fichier JAR en utilisant les options suivantes:

- Accédez à la vue +Ajouter de la perspective Développeur, puis cliquez sur Télécharger le fichier JAR dans la tuile From Local Machine. Recherchez et sélectionnez votre fichier JAR, ou faites glisser un fichier JAR pour déployer votre application.
- Accédez à la vue Topology et utilisez l'option Upload JAR, ou faites glisser un fichier JAR pour déployer votre application.
- Dans la vue Topology, utilisez le menu in-contexte, puis utilisez l'option Upload JAR pour télécharger votre fichier JAR pour déployer votre application.

Conditions préalables

- L'opérateur d'échantillons de cluster doit être installé par un utilisateur ayant le rôle d'administrateur dédié.
- Accès au service Red Hat OpenShift sur la console web AWS et vous êtes dans la perspective Développeur.

Procédure

1. Dans la vue Topologie, faites un clic droit n'importe où pour afficher le menu Ajouter au projet.

2. Survolez le menu Ajouter au projet pour voir les options de menu, puis sélectionnez l'option Télécharger le fichier JAR pour afficher le formulaire de fichier JAR. Alternativement, vous pouvez glisser le fichier JAR dans la vue Topology.

3. Dans le champ fichier JAR, recherchez le fichier JAR requis sur votre machine locale et téléchargez-le. Alternativement, vous pouvez faire glisser le fichier JAR sur le champ. En haut à droite, une alerte toast est affichée si un type de fichier incompatible est glisser dans la vue Topology. L'erreur de champ s'affiche si un type de fichier incompatible est déposé sur le champ dans le formulaire de téléchargement.

4. L'icône d'exécution et l'image du constructeur sont sélectionnées par défaut. Lorsqu'une image de constructeur n'est pas détectée automatiquement, sélectionnez une image de constructeur. Au besoin, vous pouvez modifier la version à l'aide de la liste déroulante de la version déroulante de Builder Image.

5. Facultatif: Dans le champ Nom de l'application, entrez un nom unique pour votre application à utiliser pour l'étiquetage des ressources.

6. Dans le champ Nom, entrez un nom de composant unique pour les ressources associées.

7. Facultatif: Utilisez la liste déroulante type de ressource pour modifier le type de ressource.

8. Dans le menu Options avancées, cliquez sur Créer un itinéraire vers l'application pour configurer une URL publique pour votre application déployée.

9. Cliquez sur Créer pour déployer l'application. Il est indiqué qu'une notification de toast vous informe que le fichier JAR est en cours de téléchargement. La notification de toast comprend également un lien pour voir les journaux de construction.

NOTE

Lorsque vous tentez de fermer l'onglet du navigateur pendant que la construction est en cours d'exécution, une alerte Web s'affiche.

Après le téléchargement du fichier JAR et le déploiement de l'application, vous pouvez afficher l'application dans la vue Topology.

3.2.7. En utilisant le registre Devfile pour accéder aux devfiles

Dans le flux +Add de la perspective Développeur, vous pouvez utiliser les devfiles pour créer une application. Le flux +Add fournit une intégration complète avec le registre communautaire devfile. Devfile est un fichier YAML portable qui décrit votre environnement de développement sans avoir besoin de le configurer à partir de zéro. En utilisant le registre Devfile, vous pouvez utiliser un devfile préconfiguré pour créer une application.

Procédure

1. Accédez à Perspective des développeurs → +Ajouter → Catalogue des développeurs → Tous les services. La liste de tous les services disponibles dans le catalogue des développeurs est affichée.

2. Dans Type, cliquez sur Devfiles pour parcourir les devfiles qui prennent en charge un langage ou un framework particulier. Alternativement, vous pouvez utiliser le filtre de mots clés pour rechercher un devfile particulier en utilisant leur nom, leur balise ou leur description.

3. Cliquez sur le fichier devfile que vous souhaitez utiliser pour créer une application. La tuile devfile affiche les détails du devfile, y compris le nom, la description, le fournisseur et la documentation du devfile.

4. Cliquez sur Créer pour créer une application et afficher l'application dans la vue Topology.

3.2.8. En utilisant le catalogue des développeurs pour ajouter des services ou des composants à votre application

Le catalogue des développeurs vous permet de déployer des applications et des services basés sur des services soutenus par l'opérateur tels que les bases de données, les images Builder et les graphiques Helm. Le catalogue des développeurs contient une collection de composants d'applications, de services, de sources d'événements ou de constructeurs de sources à image que vous pouvez ajouter à votre projet. Les administrateurs de clusters peuvent personnaliser le contenu mis à disposition dans le catalogue.

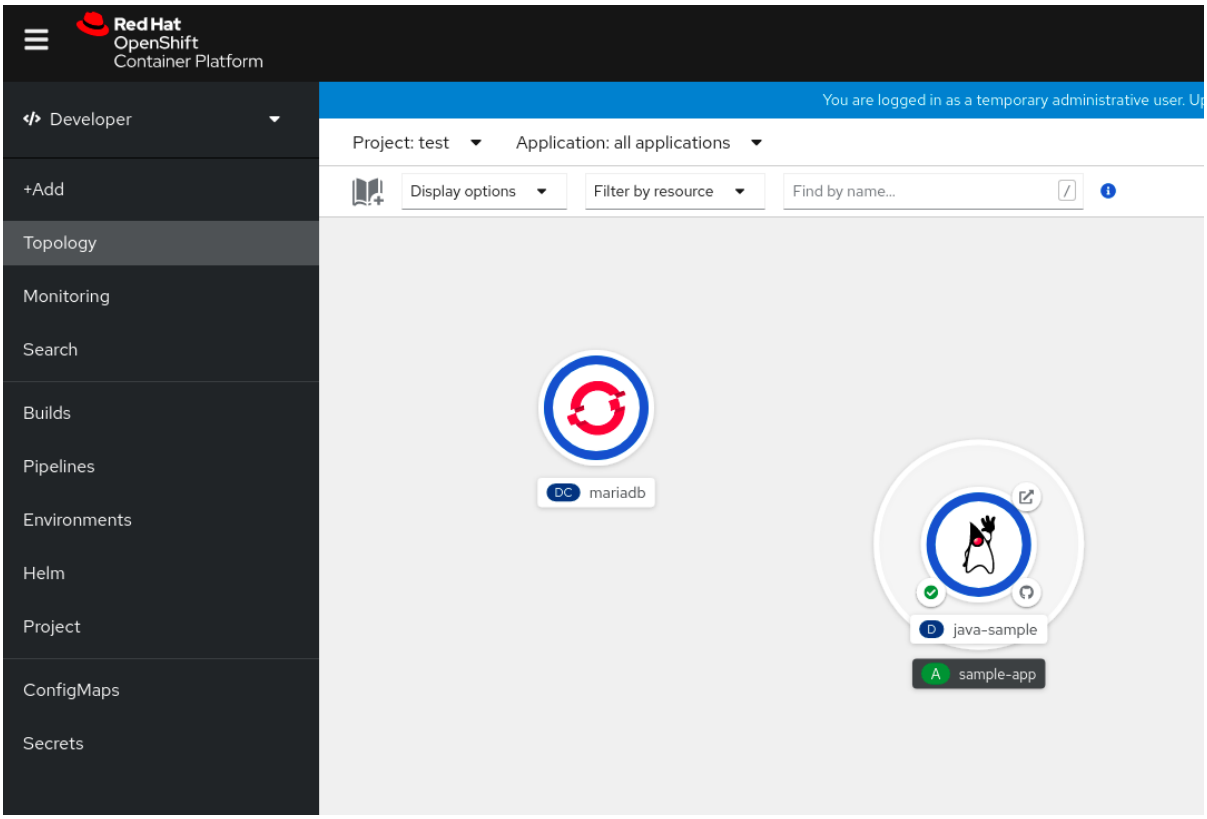
Procédure

1. Dans la perspective Développeur, accédez à la vue +Ajouter et à partir de la tuile du catalogue des développeurs, cliquez sur Tous les services pour afficher tous les services disponibles dans le catalogue des développeurs.

2. Dans Tous les services, sélectionnez le type de service ou le composant que vous devez ajouter à votre projet. Dans cet exemple, sélectionnez Bases de données pour répertorier tous les services de base de données, puis cliquez sur MariaDB pour voir les détails du service.

3. Cliquez sur Instantiate Template pour voir un modèle automatiquement peuplé avec des détails pour le service MariaDB, puis cliquez sur Créer pour créer et afficher le service MariaDB dans la vue Topology.

Figure 3.1. La MariaDB en topologie



3.2.9. Ressources supplémentaires

- En savoir plus sur les paramètres de routage Knative pour OpenShift Serverless, voir Routage.
- En savoir plus sur les paramètres de cartographie de domaine pour OpenShift Serverless, consultez Configurer un domaine personnalisé pour un service Knative.
- En savoir plus sur Knative Autoscaling Paramètres pour OpenShift Serverless, voir Autoscaling.
- En savoir plus sur l'ajout d'un nouvel utilisateur à un projet, voir Travailler avec des projets.
- En savoir plus sur la création d'un référentiel Helm Chart, voir Créer des dépôts Helm Chart.

3.3. CRÉATION D'APPLICATIONS À PARTIR D'OPÉRATEURS INSTALLÉS

Les opérateurs sont une méthode d'emballage, de déploiement et de gestion d'une application Kubernetes. Il est possible de créer des applications sur Red Hat OpenShift Service sur AWS à l'aide d'opérateurs installés par un administrateur de clusters.

Ce guide guide les développeurs à travers un exemple de création d'applications à partir d'un opérateur installé à l'aide du Red Hat OpenShift Service sur la console web AWS.

3.3.1. Création d'un cluster etcd à l'aide d'un opérateur

Cette procédure passe par la création d'un nouveau cluster etcd à l'aide de l'opérateur etcd, géré par Operator Lifecycle Manager (OLM).

Conditions préalables

- Accès à un service Red Hat OpenShift sur AWS cluster.
- L'opérateur etcd déjà installé à l'échelle du cluster par un administrateur.

Procédure

1. Créez un nouveau projet dans le Red Hat OpenShift Service sur la console web AWS pour cette procédure. Cet exemple utilise un projet appelé my-etcd.
2. Accédez à la page Opérateurs installés → Opérateurs installés. Les opérateurs qui ont été installés sur le cluster par l'administrateur dédié et qui sont disponibles pour utilisation sont présentés ici sous la forme d'une liste de versions de services de cluster (CSV). Les CSV sont utilisés pour lancer et gérer le logiciel fourni par l'opérateur.

   ASTUCE

   Cette liste peut être obtenue à partir du CLI en utilisant:

   ```
   $ oc get csv
   ```

3. Dans la page Opérateurs installés, cliquez sur l'opérateur etcd pour voir plus de détails et les actions disponibles.
   Comme indiqué dans les API fournies, cet opérateur met à disposition trois nouveaux types de ressources, dont un pour un cluster etcd (la ressource EtcdCluster). Ces objets fonctionnent comme les Kubernetes natifs intégrés, tels que Déploiement ou ReplicaSet, mais contiennent une logique spécifique à la gestion etcd.
4. Créer un nouveau cluster etcd:
   a. Dans la zone API de cluster etcd, cliquez sur Créer une instance.
   b. La page suivante vous permet d'apporter des modifications au modèle de démarrage minimal d'un objet EtcdCluster, comme la taille du cluster. Cliquez pour l'instant sur Créer pour finaliser. Cela déclenche l'opérateur pour démarrer les pods, services et autres composants du nouveau cluster etcd.
5. Cliquez sur le cluster exemple etcd, puis cliquez sur l'onglet Ressources pour voir que votre projet contient maintenant un certain nombre de ressources créées et configurées automatiquement par l'opérateur.
   Assurez-vous qu'un service Kubernetes a été créé qui vous permet d'accéder à la base de données à partir d'autres pods de votre projet.
6. L'ensemble des utilisateurs ayant le rôle d'édition dans un projet donné peuvent créer, gérer et supprimer des instances d'application (un cluster etcd, dans cet exemple) gérées par des opérateurs qui ont déjà été créés dans le projet, de manière en libre-service, tout comme un service cloud. Lorsque vous souhaitez activer d'autres utilisateurs avec cette capacité, les administrateurs de projet peuvent ajouter le rôle à l'aide de la commande suivante:

   ```
   $ oc policy add-role-to-user edit <user> -n <target_project>
   ```

Il y a maintenant un cluster etcd qui va réagir aux défaillances et rééquilibrer les données à mesure que les pods deviennent malsains ou sont migrés entre les nœuds du cluster. Le plus important, les administrateurs dédiés ou les développeurs avec un accès approprié peuvent désormais facilement utiliser la base de données avec leurs applications.

3.4. CRÉER DES APPLICATIONS EN UTILISANT LE CLI

Il est possible de créer un service Red Hat OpenShift sur l'application AWS à partir de composants qui incluent du code source ou binaire, des images et des modèles à l'aide du service Red Hat OpenShift sur AWS CLI.

L'ensemble d'objets créés par la nouvelle application dépend des artefacts passés en entrée : référentiels sources, images ou modèles.

3.4.1. Création d'une application à partir du code source

Avec la commande new-app, vous pouvez créer des applications à partir du code source dans un référentiel Git local ou distant.

La commande new-app crée une configuration de build, qui crée elle-même une nouvelle image d'application à partir de votre code source. La commande new-app crée généralement un objet Déploiement pour déployer la nouvelle image, et un service pour fournir un accès équilibré à la charge au déploiement exécutant votre image.

Le service OpenShift Red Hat sur AWS détecte automatiquement si la stratégie de construction de pipeline, source ou docker doit être utilisée et, dans le cas de la construction source, détecte une image de constructeur de langage appropriée.

3.4.1.1. Au niveau local

Créer une application à partir d'un référentiel Git dans un répertoire local:

```
$ oc new-app /<path to source code>
```

NOTE

Lorsque vous utilisez un référentiel Git local, le référentiel doit avoir une origine nommée à distance qui pointe vers une URL accessible par le service Red Hat OpenShift sur le cluster AWS. En l'absence d'une télécommande reconnue, l'exécution de la commande new-app créera une construction binaire.

3.4.1.2. À distance

Créer une application à partir d'un référentiel Git distant:

```
$ oc new-app https://github.com/sclorg/cakephp-ex
```

Créer une application à partir d'un référentiel Git privé distant:

```
$ oc new-app https://github.com/youruser/yourprivaterepo --source-secret=yoursecret
```

NOTE

Lorsque vous utilisez un dépôt Git privé distant, vous pouvez utiliser le drapeau --source-secret pour spécifier un secret de clone source existant qui sera injecté dans votre configuration de construction pour accéder au référentiel.

Il est possible d'utiliser un sous-répertoire de votre référentiel de code source en spécifiant un drapeau --context-dir. Créer une application à partir d'un référentiel Git distant et d'un sous-répertoire contextuel:

```
$ oc new-app https://github.com/sclorg/s2i-ruby-container.git \
    --context-dir=2.0/test/puma-test-app
```

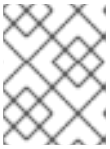
En outre, lorsque vous spécifiez une URL distante, vous pouvez spécifier une branche Git à utiliser en joignant #<branch_name> à la fin de l'URL:

```
$ oc new-app https://github.com/openshift/ruby-hello-world.git#beta4
```

3.4.1.3. Construire la détection de stratégie

Le service Red Hat OpenShift sur AWS détermine automatiquement la stratégie de création à utiliser en détectant certains fichiers:

- Lorsqu'un fichier Jenkins existe dans le répertoire racine ou spécifié du référentiel source lors de la création d'une nouvelle application, Red Hat OpenShift Service sur AWS génère une stratégie de construction de pipelines.



NOTE

La stratégie de construction de pipelines est dépréciée; envisagez d'utiliser les pipelines Red Hat OpenShift à la place.

- Lorsqu'un Dockerfile existe dans le répertoire racine ou spécifié du référentiel source lors de la création d'une nouvelle application, Red Hat OpenShift Service sur AWS génère une stratégie de création de docker.
- En cas de détection d'un fichier Jenkins ni d'un Dockerfile, Red Hat OpenShift Service sur AWS génère une stratégie de création de sources.

Remplacez la stratégie de construction détectée automatiquement en définissant le drapeau --stratégie à docker, pipeline ou source.

```
$ oc new-app /home/user/code/myapp --strategy=docker
```

NOTE

La commande oc exige que les fichiers contenant des sources de build soient disponibles dans un dépôt Git distant. Dans toutes les versions de source, vous devez utiliser git Remote -v.

3.4.1.4. Détection de la langue

Lorsque vous utilisez la stratégie de création de source, new-app tente de déterminer le constructeur de langage à utiliser par la présence de certains fichiers dans le répertoire racine ou spécifié du répertoire contextuel du référentiel:

Tableau 3.1. Langues détectées par new-app

Langue	Fichiers
JEE	**le POM.xml**
à propos de NodeJS	App.json, package.json
à propos de Perl	cpanfile, index.pl
à propos de PHP	compositeur.json, index.php
à propos de Python	conditions.txt, setup.py
♪ **Ruby** ♪	Gemfile, Rakefile, config.ru
à propos de Scala	**construire.sbt**
Golang	Godeps, main.go

Après avoir détecté une langue, une nouvelle application recherche le service OpenShift Red Hat sur le serveur AWS pour trouver des balises de flux d'images qui ont une annotation de support correspondant à la langue détectée, ou un flux d'images qui correspond au nom de la langue détectée. En cas d'absence de correspondance, la nouvelle application effectue une recherche dans le registre Docker Hub pour trouver une image correspondant à la langue détectée en fonction du nom.

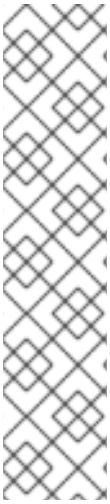
Il est possible de remplacer l'image que le constructeur utilise pour un référentiel source particulier en spécifiant l'image, soit un flux d'image ou une spécification de conteneur, et le référentiel avec un ~ comme séparateur. A noter que si cela est fait, construire la détection de stratégie et la détection du langage ne sont pas effectués.

À titre d'exemple, utiliser le flux d'images myproject/my-ruby avec la source dans un référentiel distant:

```
$ oc new-app myproject/my-ruby~https://github.com/openshift/ruby-hello-world.git
```

D'utiliser le flux d'images openshift/ruby-20-centos7: dernier conteneur avec la source dans un référentiel local:

```
$ oc new-app openshift/ruby-20-centos7:latest~/home/user/code/my-ruby-app
```



NOTE

La détection de la langue nécessite que le client Git soit installé localement afin que votre dépôt puisse être cloné et inspecté. Lorsque Git n'est pas disponible, vous pouvez éviter l'étape de détection du langage en spécifiant l'image du constructeur à utiliser avec votre référentiel avec la syntaxe <image>~<répository>.

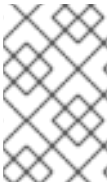
L'invocation -i <image> <repository> exige que la nouvelle application tente de cloner le référentiel pour déterminer quel type d'artefact il est, de sorte que cela échoue si Git n'est pas disponible.

L'invocation -i <image> --code <repository> nécessite un dépôt de clones new-app pour déterminer si l'image doit être utilisée comme constructeur pour le code source, ou déployée séparément, comme dans le cas d'une image de base de données.

3.4.2. Créer une application à partir d'une image

Il est possible de déployer une application à partir d'une image existante. Les images peuvent provenir de flux d'images dans le service OpenShift Red Hat sur le serveur AWS, d'images dans un registre spécifique ou d'images dans le serveur Docker local.

La commande new-app tente de déterminer le type d'image spécifié dans les arguments qui lui ont été transmis. Cependant, vous pouvez explicitement dire à new-app si l'image est une image conteneur à l'aide de l'argument --docker-image ou un flux d'image en utilisant l'argument -i|--image-stream.



NOTE

Lorsque vous spécifiez une image de votre dépôt Docker local, vous devez vous assurer que la même image est disponible pour le service Red Hat OpenShift sur les nœuds de cluster AWS.

3.4.2.1. Docker Hub MySQL image

Créez une application à partir de l'image Docker Hub MySQL, par exemple:

```
$ oc new-app mysql
```

3.4.2.2. Image dans un registre privé

Créez une application à l'aide d'une image dans un registre privé, spécifiez la spécification complète de l'image du conteneur:

```
$ oc new-app myregistry:5000/example/myimage
```

3.4.2.3. Flux d'images existants et balise optionnelle de flux d'images

Créez une application à partir d'un flux d'images existant et d'une balise optionnelle de flux d'images:

```
$ oc new-app my-stream:v1
```

3.4.3. Créer une application à partir d'un modèle

Il est possible de créer une application à partir d'un modèle précédemment stocké ou à partir d'un fichier de modèle, en spécifiant le nom du modèle en tant qu'argument. À titre d'exemple, vous pouvez stocker un exemple de modèle d'application et l'utiliser pour créer une application.

Envoyez un modèle d'application dans la bibliothèque de modèles de votre projet actuel. L'exemple suivant télécharge un modèle d'application à partir d'un fichier appelé example/sample-app/application-template-stibuild.json:

```
$ oc create -f examples/sample-app/application-template-stibuild.json
```

Créez ensuite une nouvelle application en faisant référence au modèle d'application. Dans cet exemple, le nom du modèle est ruby-helloworld-sample:

```
$ oc new-app ruby-helloworld-sample
```

Afin de créer une nouvelle application en faisant référence à un fichier modèle dans votre système de fichiers local, sans le stocker d'abord dans Red Hat OpenShift Service sur AWS, utilisez l'argument -f|-fichier. À titre d'exemple:

```
$ oc new-app -f examples/sample-app/application-template-stibuild.json
```

3.4.3.1. Les paramètres du modèle

Lors de la création d'une application basée sur un modèle, utilisez l'argument -p|--param pour définir les valeurs de paramètres définies par le modèle:

```
$ oc new-app ruby-helloworld-sample \
    -p ADMIN_USERNAME=admin -p ADMIN_PASSWORD=mypassword
```

Il est possible de stocker vos paramètres dans un fichier, puis d'utiliser ce fichier avec --param-file lors de l'instanciation d'un modèle. Lorsque vous souhaitez lire les paramètres à partir de l'entrée standard, utilisez --param-file=-. Ce qui suit est un fichier d'exemple appelé helloworld.params:

```
ADMIN_USERNAME=admin
ADMIN_PASSWORD=mypassword
```

Faites référence aux paramètres dans le fichier lors de l'instanciation d'un modèle:

```
$ oc new-app ruby-helloworld-sample --param-file=helloworld.params
```

3.4.4. Création d'applications

La nouvelle commande d'application génère Red Hat OpenShift Service sur les objets AWS qui construisent, déploient et exécutent l'application créée. Habituellement, ces objets sont créés dans le projet actuel et les noms attribués qui sont dérivés des référentiels source d'entrée ou des images d'entrée. Cependant, avec la nouvelle application, vous pouvez modifier ce comportement.

Tableau 3.2. les objets de sortie de nouvelle application

L'objet	Description

L'objet	Description
BuildConfig	L'objet BuildConfig est créé pour chaque référentiel source spécifié dans la ligne de commande. L'objet BuildConfig spécifie la stratégie à utiliser, l'emplacement source et l'emplacement de sortie de construction.
ImageStreams	Dans le cas de l'objet BuildConfig, deux flux d'images sont généralement créés. La première représente l'image d'entrée. Avec les builds source, c'est l'image du constructeur. Avec Docker builds, c'est l'image FROM. Le second représente l'image de sortie. Lorsqu'une image conteneur a été spécifiée comme entrée dans la nouvelle application, un flux d'images est également créé pour cette image.
DéploiementConfig	L'objet DeploymentConfig est créé soit pour déployer la sortie d'une build, soit pour une image spécifiée. La commande new-app crée des volumes videDir pour tous les volumes Docker spécifiés dans les conteneurs inclus dans l'objet DeploymentConfig résultant.
Le service	La commande new-app tente de détecter les ports exposés dans les images d'entrée. Il utilise le port exposé numérique le plus bas pour générer un service qui expose ce port. Afin d'exposer un port différent, une fois la nouvelle application terminée, il suffit d'utiliser la commande oc expose pour générer des services supplémentaires.
Autres	D'autres objets peuvent être générés lors de l'instanciation des modèles, selon le modèle.

3.4.4.1. Spécification des variables d'environnement

Lorsque vous générez des applications à partir d'un modèle, d'une source ou d'une image, vous pouvez utiliser l'argument -e|-env pour transmettre des variables d'environnement au conteneur d'application au moment de l'exécution:

```
$ oc new-app openshift/postgresql-92-centos7 \
    -e POSTGRESQL_USER=user \
    -e POSTGRESQL_DATABASE=db \
    -e POSTGRESQL_PASSWORD=password
```

Les variables peuvent également être lues à partir du fichier en utilisant l'argument --env-file. Ce qui suit est un fichier d'exemple appelé postgresql.env:

```
POSTGRESQL_USER=user
POSTGRESQL_DATABASE=db
POSTGRESQL_PASSWORD=password
```

Lisez les variables du fichier:

```
$ oc new-app openshift/postgresql-92-centos7 --env-file=postgresql.env
```

De plus, des variables d'environnement peuvent être données sur l'entrée standard en utilisant --env-file=-:

```
$ cat postgresql.env | oc new-app openshift/postgresql-92-centos7 --env-file=-
```

NOTE

Les objets BuildConfig créés dans le cadre du traitement de nouvelles applications ne sont pas mis à jour avec les variables d'environnement passées avec l'argument -e|--env ou --env-file.

3.4.4.2. Définir les variables d'environnement de construction

Lorsque vous générez des applications à partir d'un modèle, d'une source ou d'une image, vous pouvez utiliser l'argument --build-env pour transmettre des variables d'environnement au conteneur de construction au moment de l'exécution:

```
$ oc new-app openshift/ruby-23-centos7 \
    --build-env HTTP_PROXY=http://myproxy.net:1337/ \
    --build-env GEM_HOME=~/.gem
```

Les variables peuvent également être lues à partir d'un fichier en utilisant l'argument --build-env-file. Ce qui suit est un exemple de fichier appelé ruby.env:

```
HTTP_PROXY=http://myproxy.net:1337/
GEM_HOME=~/.gem
```

Lisez les variables du fichier:

```
$ oc new-app openshift/ruby-23-centos7 --build-env-file=ruby.env
```

De plus, des variables d'environnement peuvent être données sur l'entrée standard en utilisant --build-env-file=-:

```
$ cat ruby.env | oc new-app openshift/ruby-23-centos7 --build-env-file=-
```

3.4.4.3. Spécification des étiquettes

Lorsque vous générez des applications à partir de sources, d'images ou de modèles, vous pouvez utiliser l'argument -l|--label pour ajouter des étiquettes aux objets créés. Les étiquettes facilitent la sélection, la configuration et la suppression des objets associés à l'application.

```
$ oc new-app https://github.com/openshift/ruby-hello-world -l name=hello-world
```

3.4.4.4. Affichage de la sortie sans création

Afin de voir une exécution à sec de la commande new-app, vous pouvez utiliser l'argument -o|--sortie avec une valeur yaml ou json. Ensuite, vous pouvez utiliser la sortie pour prévisualiser les objets créés ou rediriger vers un fichier que vous pouvez éditer. Après avoir été satisfait, vous pouvez utiliser oc create pour créer le service OpenShift Red Hat sur les objets AWS.

Afin de produire des artefacts de nouvelle application dans un fichier, exécutez ce qui suit:

```
$ oc new-app https://github.com/openshift/ruby-hello-world \
   -o yaml > myapp.yaml
```

Éditer le fichier:

```
$ vi myapp.yaml
```

Créez une nouvelle application en faisant référence au fichier:

```
$ oc create -f myapp.yaml
```

3.4.4.5. Création d'objets avec différents noms

Les objets créés par la nouvelle application sont normalement nommés d'après le référentiel source, ou l'image utilisée pour les générer. Il est possible de définir le nom des objets produits en ajoutant un drapeau --name à la commande:

```
$ oc new-app https://github.com/openshift/ruby-hello-world --name=myapp
```

3.4.4.6. Créer des objets dans un projet différent

Habituellement, new-app crée des objets dans le projet actuel. Cependant, vous pouvez créer des objets dans un projet différent en utilisant l'argument -n|--namespace:

```
$ oc new-app https://github.com/openshift/ruby-hello-world -n myproject
```

3.4.4.7. Créer plusieurs objets

La commande new-app permet de créer plusieurs applications spécifiant plusieurs paramètres à new-app. Les étiquettes spécifiées dans la ligne de commande s'appliquent à tous les objets créés par la commande unique. Les variables d'environnement s'appliquent à tous les composants créés à partir de sources ou d'images.

Créer une application à partir d'un référentiel source et d'une image Docker Hub:

```
$ oc new-app https://github.com/openshift/ruby-hello-world mysql
```

NOTE

Lorsqu'un référentiel de code source et une image de constructeur sont spécifiés comme arguments distincts, new-app utilise l'image du constructeur comme constructeur pour le référentiel de code source. Dans le cas contraire, spécifiez l'image de constructeur requise pour la source à l'aide du séparateur ~.

3.4.4.8. Groupement d'images et source en un seul pod

La commande new-app permet de déployer plusieurs images ensemble dans un seul pod. Afin de spécifier les images à regrouper, utilisez le séparateur +. L'argument de ligne de commande --group peut également être utilisé pour spécifier les images qui doivent être regroupées. Afin de regrouper l'image construite à partir d'un référentiel source avec d'autres images, spécifiez son image constructeur dans le groupe:

```
$ oc new-app ruby+mysql
```

Déployer une image construite à partir d'une source et d'une image externe ensemble:

```
$ oc new-app \
    ruby~https://github.com/openshift/ruby-hello-world \
    mysql \
    --group=ruby+mysql
```

3.4.4.9. La recherche d'images, de modèles et d'autres entrées

Afin de rechercher des images, des modèles et d'autres entrées pour la commande oc new-app, ajoutez les drapeaux --search et --list. À titre d'exemple, pour trouver toutes les images ou modèles qui incluent PHP:

```
$ oc new-app --search php
```

3.4.4.10. Définir le mode d'importation

Afin de définir le mode d'importation lors de l'utilisation d'oc new-app, ajoutez le drapeau --import-mode. Ce drapeau peut être ajouté avec Legacy ou PreserveOriginal, qui offre aux utilisateurs la possibilité de créer des flux d'images en utilisant un seul sous-manifeste, ou tous les manifestes, respectivement.

```
$ oc new-app --image=registry.redhat.io/ubi8/httpd-24:latest  --import-mode=Legacy --name=test
```

```
$ oc new-app --image=registry.redhat.io/ubi8/httpd-24:latest  --import-mode=PreserveOriginal --
name=test
```

3.5. CRÉATION D'APPLICATIONS À L'AIDE DE RUBY ON RAILS

Le Ruby on Rails est un framework web écrit en Ruby. Ce guide couvre l'utilisation de Rails 4 sur Red Hat OpenShift Service sur AWS.

AVERTISSEMENT

Consultez l'ensemble du tutoriel pour avoir une vue d'ensemble de toutes les étapes nécessaires pour exécuter votre application sur le Service OpenShift Red Hat sur AWS. Lorsque vous rencontrez un problème, essayez de lire l'ensemble du tutoriel, puis retournez à votre problème. Il peut également être utile de revoir vos étapes précédentes pour vous assurer que toutes les étapes ont été exécutées correctement.

3.5.1. Conditions préalables

- Connaissances de base Ruby et Rails.

- La version installée localement de Ruby 2.0.0+, Rubygems, Bundler.
- Connaissances de base sur Git.
- Instance d'exécution de Red Hat OpenShift Service sur AWS 4.
- Assurez-vous qu'une instance de Red Hat OpenShift Service sur AWS est en cours d'exécution et est disponible. Assurez-vous également que votre client CLI oc est installé et que la commande est accessible depuis votre shell de commande, de sorte que vous pouvez l'utiliser pour vous connecter à l'aide de votre adresse e-mail et mot de passe.

3.5.2. Configuration de la base de données

Les applications ferroviaires sont presque toujours utilisées avec une base de données. Dans le cas du développement local, utilisez la base de données PostgreSQL.

Procédure

1. Installer la base de données:

```
$ sudo yum install -y postgresql postgresql-server postgresql-devel
```

2. Initialiser la base de données:

```
$ sudo postgresql-setup initdb
```

   Cette commande crée le répertoire /var/lib/pgsql/data, dans lequel les données sont stockées.

3. Démarrer la base de données:

```
$ sudo systemctl start postgresql.service
```

4. Lorsque la base de données est en cours d'exécution, créez votre utilisateur de rails:

```
$ sudo -u postgres createuser -s rails
```

   A noter que l'utilisateur créé n'a pas de mot de passe.

3.5.3. Écrire votre candidature

Lorsque vous démarrez votre application Rails à partir de zéro, vous devez d'abord installer le gemme Rails. Ensuite, vous pouvez procéder à la rédaction de votre demande.

Procédure

1. Installez le gemme Rails:

```
$ gem install rails
```

Exemple de sortie

```
Successfully installed rails-4.3.0
1 gem installed
```

2. Après avoir installé le gemme Rails, créez une nouvelle application avec PostgreSQL comme base de données:

```
$ rails new rails-app --database=postgresql
```

3. Changez dans votre nouveau répertoire d'applications:

```
$ cd rails-app
```

4. Lorsque vous avez déjà une application, assurez-vous que le bijou pg (postgresql) est présent dans votre fichier Gemfile. Dans le cas contraire, modifiez votre Gemfile en ajoutant le gemme:

```
gem 'pg'
```

5. Générez un nouveau Gemfile.lock avec toutes vos dépendances:

```
$ bundle install
```

6. En plus d'utiliser la base de données postgresql avec le gemme pg, vous devez également vous assurer que la config/database.yml utilise l'adaptateur postgresql.
Assurez-vous de mettre à jour la section par défaut dans le fichier config/database.yml, de sorte qu'il ressemble à ceci:

```
default: &default
  adapter: postgresql
  encoding: unicode
  pool: 5
  host: localhost
  username: rails
  password: <password>
```

7. Créez les bases de données de développement et de test de votre application:

```
$ rake db:create
```

Cela crée une base de données de développement et de test dans votre serveur PostgreSQL.

3.5.3.1. Créer une page de bienvenue

Étant donné que Rails 4 ne sert plus une page publique statique/index.html en production, vous devez créer une nouvelle page racine.

Avoir une page d'accueil personnalisée doit faire les étapes suivantes:

- Créez un contrôleur avec une action d'index.
- Créez une page d'affichage pour l'action d'index du contrôleur de bienvenue.
- Créez un itinéraire qui sert la page racine des applications avec le contrôleur créé et la vue.

Rails offre un générateur qui complète toutes les étapes nécessaires pour vous.

Procédure

1. Générateur de rails d'exécution:

```
$ rails generate controller welcome index
```

L'ensemble des fichiers nécessaires sont créés.

2. éditer la ligne 2 dans le fichier config/routes.rb comme suit:

```
root 'welcome#index'
```

3. Exécutez le serveur de rails pour vérifier la page est disponible:

```
$ rails server
```

Consultez votre page en visitant http://localhost:3000 dans votre navigateur. Lorsque vous ne voyez pas la page, vérifiez les journaux qui sont affichés sur votre serveur pour déboguer.

3.5.3.2. Configuration de l'application pour Red Hat OpenShift Service sur AWS

Afin que votre application communique avec le service de base de données PostgreSQL exécuté dans Red Hat OpenShift Service sur AWS, vous devez modifier la section par défaut de votre config/database.yml pour utiliser des variables d'environnement, que vous devez définir ultérieurement lors de la création du service de base de données.

Procédure

- Éditez la section par défaut dans votre config/database.yml avec des variables prédéfinies comme suit:

Exemple de fichier YAML de configuration/base de données

```
<% user = ENV.key?("POSTGRESQL_ADMIN_PASSWORD") ? "root" :
ENV["POSTGRESQL_USER"] %>
<% password = ENV.key?("POSTGRESQL_ADMIN_PASSWORD") ?
ENV["POSTGRESQL_ADMIN_PASSWORD"] : ENV["POSTGRESQL_PASSWORD"] %>
<% db_service = ENV.fetch("DATABASE_SERVICE_NAME","").upcase %>

default: &default
  adapter: postgresql
  encoding: unicode
  # For details on connection pooling, see rails configuration guide
  # http://guides.rubyonrails.org/configuring.html#database-pooling
  pool: <%= ENV["POSTGRESQL_MAX_CONNECTIONS"] || 5 %>
  username: <%= user %>
  password: <%= password %>
  host: <%= ENV["#{db_service}_SERVICE_HOST"] %>
  port: <%= ENV["#{db_service}_SERVICE_PORT"] %>
  database: <%= ENV["POSTGRESQL_DATABASE"] %>
```

3.5.3.3. Stocker votre application dans Git

Construire une application dans Red Hat OpenShift Service sur AWS nécessite généralement que le code source soit stocké dans un référentiel git, vous devez donc installer git si vous ne l'avez pas déjà.

Conditions préalables

- Installez git.

Procédure

1. Assurez-vous d'être dans votre répertoire d'application Rails en exécutant la commande ls -1. La sortie de la commande devrait ressembler à:

```
$ ls -1
```

Exemple de sortie

```
app
bin
config
config.ru
db
Gemfile
Gemfile.lock
lib
log
public
Rakefile
README.rdoc
test
tmp
vendor
```

2. Exécutez les commandes suivantes dans votre répertoire d'application Rails pour initialiser et commettre votre code à git:

```
$ git init
```

```
$ git add .
```

```
$ git commit -m "initial commit"
```

Après que votre application est engagée, vous devez la pousser vers un référentiel distant. Compte GitHub, dans lequel vous créez un nouveau référentiel.

3. Définissez la télécommande qui pointe vers votre référentiel git:

```
$ git remote add origin git@github.com:<namespace/repository-name>.git
```

4. Appuyez sur votre application vers votre référentiel git distant.

```
$ git push
```

3.5.4. Déploiement de votre application dans Red Hat OpenShift Service sur AWS

Il est possible de déployer votre application dans Red Hat OpenShift Service sur AWS.

Après avoir créé le projet d'application rails, vous êtes automatiquement passé au nouvel espace de noms du projet.

Le déploiement de votre application dans Red Hat OpenShift Service sur AWS comporte trois étapes:

- Création d'un service de base de données à partir de Red Hat OpenShift Service sur l'image PostgreSQL d'AWS.
- Créer un service frontend à partir de Red Hat OpenShift Service sur l'image du constructeur Ruby 2.0 d'AWS et votre code source Ruby on Rails, qui sont câblés avec le service de base de données.
- Créer un itinéraire pour votre application.

3.5.4.1. Création du service de base de données

Procédure

L'application Rails s'attend à un service de base de données en cours d'exécution. Ce service utilise l'image de base de données PostgreSQL.

Afin de créer le service de base de données, utilisez la commande oc new-app. À cette commande, vous devez passer quelques variables d'environnement nécessaires qui sont utilisées à l'intérieur du conteneur de base de données. Ces variables d'environnement sont nécessaires pour définir le nom d'utilisateur, le mot de passe et le nom de la base de données. Les valeurs de ces variables d'environnement peuvent être modifiées par ce que vous souhaitez. Les variables sont les suivantes:

- AJOUTER AU PANIER POSTGRESQL_DATABASE
- AJOUTER AU PANIER POSTGRESQL_USER
- AJOUTER AU PANIER POSTGRESQL_PASSWORD

La définition de ces variables garantit:

- Il existe une base de données avec le nom spécifié.
- Il existe un utilisateur avec le nom spécifié.
- L'utilisateur peut accéder à la base de données spécifiée avec le mot de passe spécifié.

Procédure

1. Créer le service de base de données:

```
$ oc new-app postgresql -e POSTGRESQL_DATABASE=db_name -e
POSTGRESQL_USER=username -e POSTGRESQL_PASSWORD=password
```

   Afin de définir également le mot de passe pour l'administrateur de la base de données, ajouter à la commande précédente avec:

```
-e POSTGRESQL_ADMIN_PASSWORD=admin_pw
```

2. Découvrez les progrès:

```
$ oc get pods --watch
```

-

3.5.4.2. Création du service frontend

Afin d'apporter votre application à Red Hat OpenShift Service sur AWS, vous devez spécifier un référentiel dans lequel votre application vit.

Procédure

1. Créez le service frontend et spécifiez les variables d'environnement liées à la base de données qui ont été configurées lors de la création du service de base de données:

```
$ oc new-app path/to/source/code --name=rails-app -e POSTGRESQL_USER=username -e
POSTGRESQL_PASSWORD=password -e POSTGRESQL_DATABASE=db_name -e
DATABASE_SERVICE_NAME=postgresql
```

   Avec cette commande, Red Hat OpenShift Service sur AWS récupère le code source, configure le constructeur, construit l'image de votre application et déploie l'image nouvellement créée avec les variables d'environnement spécifiées. L'application est nommée Rails-app.

2. Les variables d'environnement ont été ajoutées en consultant le document JSON de la configuration de déploiement de l'application rails:

```
$ oc get dc rails-app -o json
```

   Consultez la section suivante:

   Exemple de sortie

```
env": [
    {
        "name": "POSTGRESQL_USER",
        "value": "username"
    },
    {
        "name": "POSTGRESQL_PASSWORD",
        "value": "password"
    },
    {
        "name": "POSTGRESQL_DATABASE",
        "value": "db_name"
    },
    {
        "name": "DATABASE_SERVICE_NAME",
        "value": "postgresql"
    }

],
```

3. Consultez le processus de construction:

```
$ oc logs -f build/rails-app-1
```

4. Après la construction est terminée, regardez les pods en cours d'exécution dans Red Hat OpenShift Service sur AWS:

```
$ oc get pods
```

Il faut voir une ligne commençant par myapp-<numéro>-<hash>, et c'est votre application qui s'exécute dans Red Hat OpenShift Service sur AWS.

5. Avant que votre application ne soit fonctionnelle, vous devez initialiser la base de données en exécutant le script de migration de la base de données. Il y a deux façons de le faire:

   - À partir du conteneur frontal en cours d'exécution:
     - Exec dans le conteneur frontal avec commande rsh:

       ```
       $ oc rsh <frontend_pod_id>
       ```

     - Exécutez la migration à partir de l'intérieur du conteneur:

       ```
       $ RAILS_ENV=production bundle exec rake db:migrate
       ```

       Lorsque vous exécutez votre application Rails dans un environnement de développement ou de test, vous n'avez pas à spécifier la variable d'environnement RAILS_ENV.
   - En ajoutant des crochets de cycle de vie avant déploiement dans votre modèle.

3.5.4.3. Créer un itinéraire pour votre application

Il est possible d'exposer un service pour créer un itinéraire pour votre application.

AVERTISSEMENT

Assurez-vous que le nom d'hôte que vous spécifiez résout dans l'adresse IP du routeur.

CHAPITRE 4. AFFICHAGE DE LA COMPOSITION DE L'APPLICATION À L'AIDE DE LA VUE TOPOLOGY

La vue Topology dans la perspective Développeur de la console Web fournit une représentation visuelle de toutes les applications d'un projet, de leur statut de construction et des composants et services qui leur sont associés.

4.1. CONDITIONS PRÉALABLES

Afin de visualiser vos applications dans la vue Topologie et d'interagir avec elles, assurez-vous que:

- Connectez-vous à la console web.
- « vous êtes dans la perspective Développeur.

4.2. CONSULTER LA TOPOLOGIE DE VOTRE APPLICATION

Dans la perspective Développeur, vous pouvez accéder à la vue Topology en utilisant le panneau de navigation de gauche. Après avoir déployé une application, vous êtes dirigé automatiquement vers la vue Graphique où vous pouvez voir l'état des pods d'application, accéder rapidement à l'application sur une URL publique, accéder au code source pour la modifier et voir l'état de votre dernière construction. Il est possible de zoomer vers l'intérieur et l'extérieur pour voir plus de détails pour une application particulière.

La vue Topology vous offre la possibilité de surveiller vos applications à l'aide de la vue Liste. Cliquez sur l'icône d'affichage de la liste () pour afficher une liste de toutes vos applications et utilisez l'icône Graph view () pour revenir à la vue graphique.

Il est possible de personnaliser les vues au besoin en utilisant les éléments suivants:

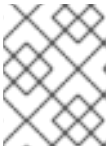
- Cliquez sur le champ Rechercher par nom pour trouver les composants requis. Les résultats de recherche peuvent apparaître en dehors de la zone visible; cliquez sur Fit to Screen de la barre d'outils inférieure gauche pour redimensionner la vue Topology pour afficher tous les composants.
- La liste déroulante Options d'affichage permet de configurer la vue Topology des différents groupes d'applications. Les options sont disponibles en fonction des types de composants déployés dans le projet:
    - Développer le groupe
        - Les machines virtuelles: basculer pour afficher ou cacher les machines virtuelles.
        - Groupements d'applications: Effacer pour condenser les groupes d'applications en cartes avec une vue d'ensemble d'un groupe d'applications et des alertes qui y sont associées.
        - Helm Releases: Clear pour condenser les composants déployés en tant que Helm Release dans les cartes avec un aperçu d'une version donnée.
        - Groupements d'opérateurs : Effacer pour condenser les composants déployés avec un opérateur en cartes avec un aperçu du groupe donné.
    - Afficher les éléments basés sur Pod Count ou Labels

- Compte de pod: Sélectionnez pour afficher le nombre de pods d'un composant dans l'icône du composant.
- Étiquettes: Appuyez pour afficher ou masquer les étiquettes des composants.

4.3. INTERAGIR AVEC LES APPLICATIONS ET LES COMPOSANTS

Dans la vue Topologie dans la perspective Développeur de la console Web, la vue Graph offre les options suivantes pour interagir avec les applications et les composants:

- Cliquez sur Ouvrir l'URL () pour voir votre application exposée par l'itinéraire sur une URL publique.
- Cliquez sur Modifier le code source pour accéder à votre code source et le modifier.



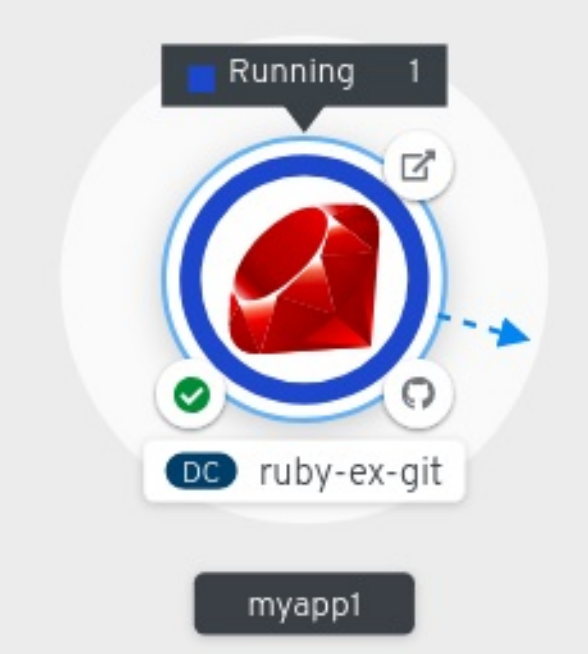
NOTE

Cette fonctionnalité n'est disponible que lorsque vous créez des applications à l'aide des options From Git, From Catalog et From Dockerfile.

- Hissez votre curseur sur l'icône en bas à gauche sur la gousse pour voir le nom de la dernière version et son statut. L'état de la construction de l'application est indiqué comme nouveau ( ), en attente ( ), en cours d'exécution ( ), complété ( ), échoué ( ) et annulé ( ).
- L'état ou la phase du pod est indiqué par différentes couleurs et infobules comme:
  - Exécution (): Le pod est lié à un nœud et tous les conteneurs sont créés. Au moins un conteneur est toujours en cours d'exécution ou est en cours de démarrage ou de redémarrage.
  - Les pods qui exécutent plusieurs conteneurs, tous les conteneurs ne sont pas prêts.
  - Avertissement( ): Les conteneurs dans les gousses sont résiliés, mais la résiliation n'a pas réussi. Certains conteneurs peuvent être d'autres états.
  - Échec(): Tous les conteneurs dans la gousse terminée, mais au moins un conteneur s'est terminé en panne. C'est-à-dire que le conteneur est sorti avec un statut non nul ou a été résilié par le système.
  - En attente(): Le pod est accepté par le cluster Kubernetes, mais un ou plusieurs des conteneurs n'ont pas été mis en place et prêts à fonctionner. Cela inclut le temps qu'un pod passe à attendre d'être programmé ainsi que le temps passé à télécharger des images de conteneurs sur le réseau.
  - Réussi(): Tous les conteneurs dans le pod se sont terminés avec succès et ne seront pas redémarrés.
  - Terminaison(): Lorsqu'un pod est supprimé, il est affiché comme Terminant par certaines commandes kubectl. L'état de terminaison n'est pas l'une des phases de la pod. La pod bénéficie d'un délai de résiliation gracieux, qui par défaut est de 30 secondes.
  - Inconnu(): L'état de la gousse n'a pas pu être obtenu. Cette phase se produit généralement en raison d'une erreur dans la communication avec le nœud où le pod doit être en cours d'exécution.

- Après avoir créé une application et qu'une image est déployée, l'état est affiché en attente. Après la construction de l'application, elle est affichée sous la forme Running.

Figure 4.1. Application topologie



Le nom de la ressource de l'application est joint avec des indicateurs pour les différents types d'objets de ressources comme suit:

- CJ: CronJob
- D: Déploiement
- DC: DéploiementConfig
- DS: DaemonSet
- J: Job
- Catégorie: Pod
- Catégorie: StatefulSet
- (Knative) : Une application sans serveur

NOTE

Les applications sans serveur prennent un certain temps pour se charger et s'afficher sur la vue Graphique. Lorsque vous déployez une application sans serveur, il crée d'abord une ressource de service, puis une révision. Après cela, il est déployé et affiché sur la vue Graphique. Lorsqu'il s'agit de la seule charge de travail, vous pouvez être redirigé vers la page Ajouter. Après le déploiement de la révision, l'application sans serveur s'affiche sur la vue Graphique.

4.4. DIMENSIONNEMENT DES PODS D'APPLICATION ET VÉRIFICATION DES CONSTRUCTIONS ET DES ITINÉRAIRES

La vue Topology fournit les détails des composants déployés dans le panneau Aperçu. Les onglets Aperçu et détails peuvent être utilisés pour mettre à l'échelle les pods de l'application, vérifier l'état de la construction, les services et les itinéraires comme suit:

- Cliquez sur le nœud du composant pour voir le panneau Aperçu à droite. Cliquez sur l'onglet Détails pour:
  - Faites évoluer vos gousses en utilisant les flèches haut et bas pour augmenter ou diminuer manuellement le nombre d'instances de l'application. Dans le cas des applications sans serveur, les pods sont automatiquement réduits à zéro lorsqu'ils sont inutilisés et mis à l'échelle en fonction du trafic du canal.
  - Consultez les étiquettes, les annotations et l'état de l'application.
- Cliquez sur l'onglet Ressources pour:
  - Consultez la liste de tous les pods, visualisez leur statut, accédez aux journaux et cliquez sur le pod pour voir les détails de la gousse.
  - Consultez les versions, leur statut, les journaux d'accès et démarrez une nouvelle version si nécessaire.
  - Consultez les services et itinéraires utilisés par le composant.

  Dans le cas des applications sans serveur, l'onglet Ressources fournit des informations sur la révision, les itinéraires et les configurations utilisées pour ce composant.

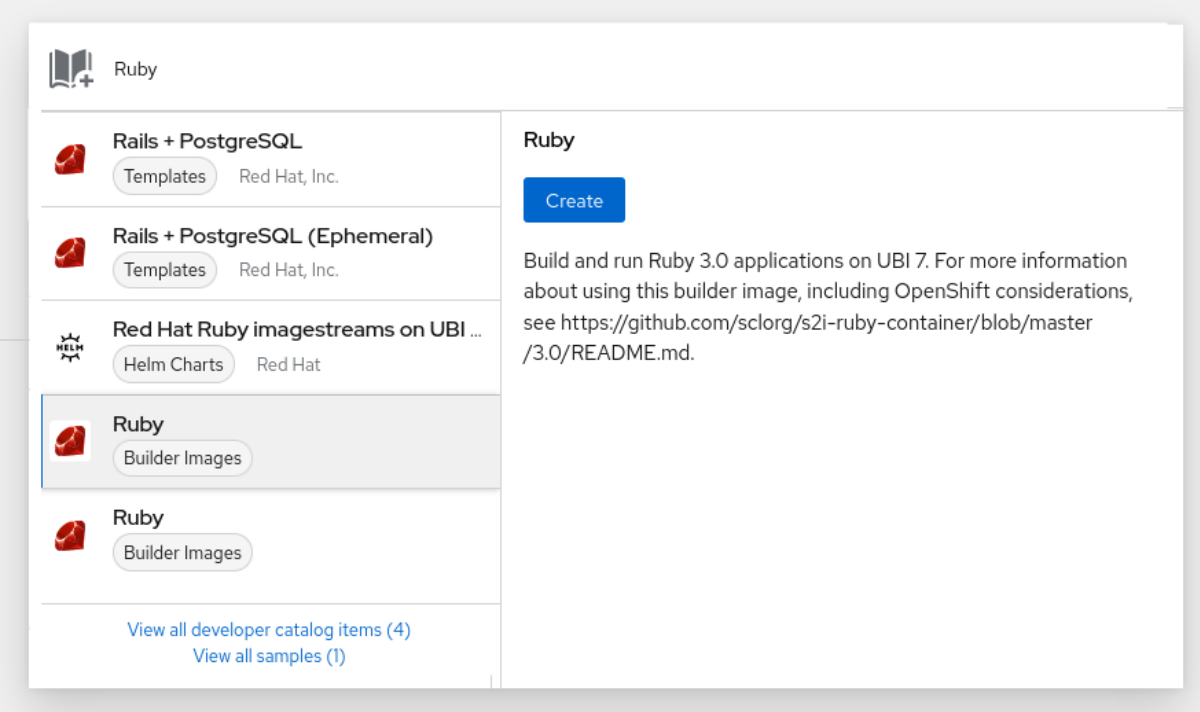
4.5. AJOUT DE COMPOSANTS À UN PROJET EXISTANT

Il est possible d'ajouter des composants à un projet.

Procédure

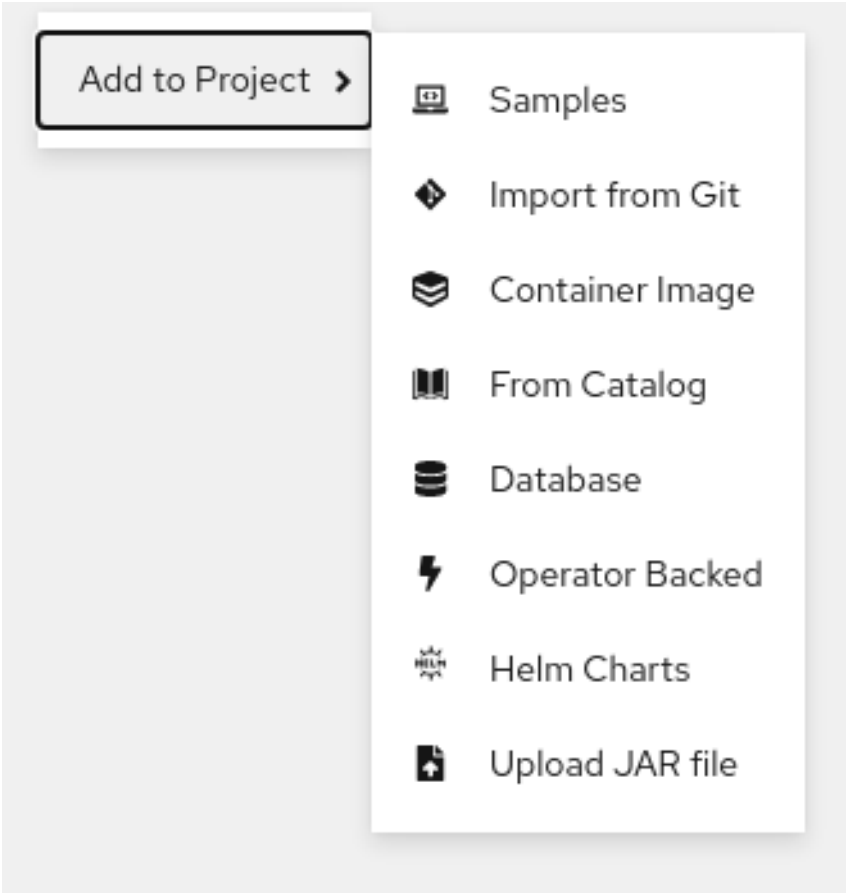
1. Accédez à la vue +Ajouter.
2. Cliquez sur Ajouter au projet () à côté du volet de navigation gauche ou appuyez sur CtrlSpace
3. Cherchez le composant et cliquez sur le bouton Démarrer/Créer/Installer ou cliquez sur Entrée pour ajouter le composant au projet et le voir dans la vue graphique de topologie.

Figure 4.2. Ajout d'un composant via une recherche rapide



Alternativement, vous pouvez également utiliser les options disponibles dans le menu contextuel, telles que l'importation à partir de Git, l'image de conteneur, la base de données, à partir du catalogue, l'opérateur sauvegardé, les graphiques Helm, les échantillons ou le téléchargement du fichier JAR, en cliquant avec le bouton droit dans la vue de topologie Graphique pour ajouter un composant à votre projet.

Figure 4.3. Le menu contextuel pour ajouter des services



4.6. REGROUPER PLUSIEURS COMPOSANTS AU SEIN D'UNE APPLICATION

La vue +Add permet d'ajouter plusieurs composants ou services à votre projet et d'utiliser la vue de topologie Graph pour regrouper les applications et les ressources au sein d'un groupe d'applications.

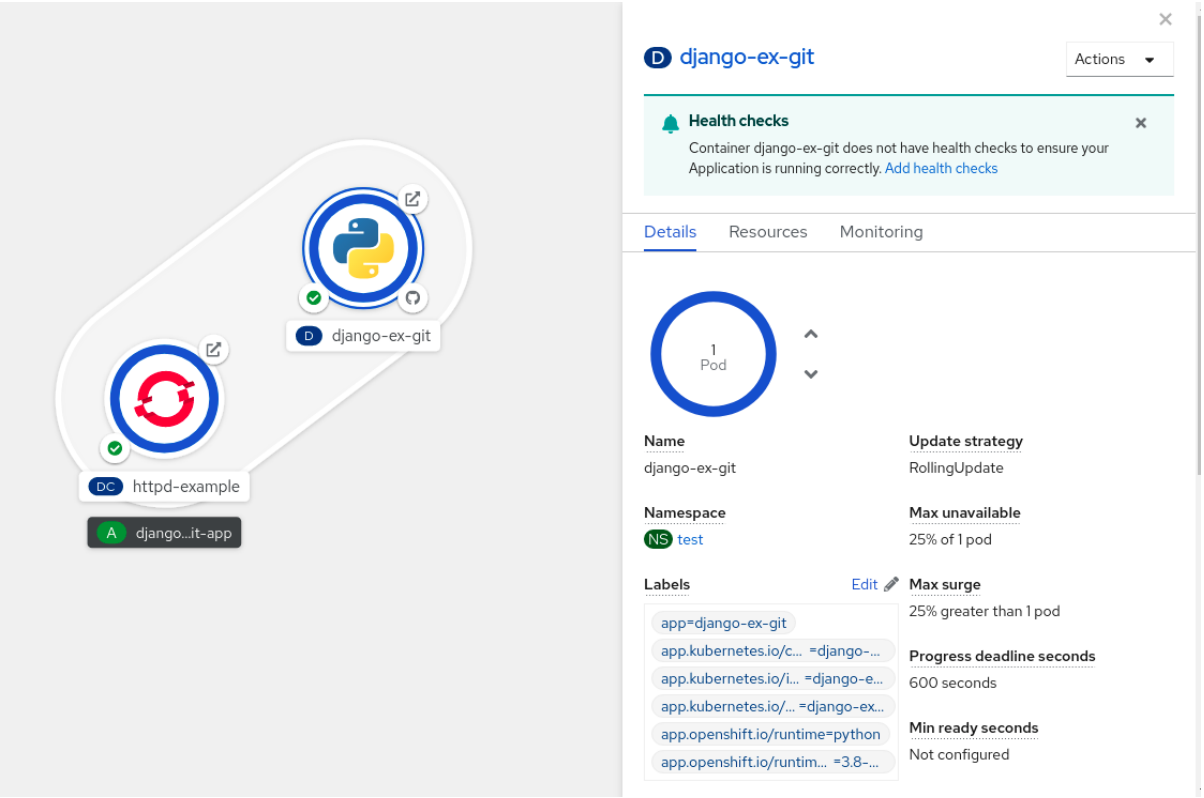
Conditions préalables

- En utilisant la perspective Développeur, vous avez créé et déployé au minimum deux composants ou plus sur Red Hat OpenShift Service sur AWS.

Procédure

- Afin d'ajouter un service au groupe d'applications existant, appuyez sur Shift+ vers le groupe d'applications existant. Faire glisser un composant et l'ajouter à un groupe d'applications ajoute les étiquettes requises au composant.

Figure 4.4. Groupement d'applications



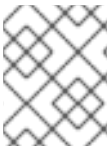
Alternativement, vous pouvez également ajouter le composant à une application comme suit:

1. Cliquez sur le module de service pour voir le panneau Aperçu à droite.
2. Cliquez sur le menu déroulant Actions et sélectionnez Modifier le groupe d'applications.
3. Dans la boîte de dialogue Modifier le groupement d'applications, cliquez sur la liste déroulante Application et sélectionnez un groupe d'application approprié.
4. Cliquez sur Enregistrer pour ajouter le service au groupe d'applications.

Il est possible de supprimer un composant d'un groupe d'applications en sélectionnant le composant et en utilisant Shift+ glisser pour le faire glisser hors du groupe d'applications.

4.7. AJOUT DE SERVICES À VOTRE APPLICATION

Afin d'ajouter un service à votre application, utilisez les actions +Add en utilisant le menu contextuel dans la vue graphique de topologie.



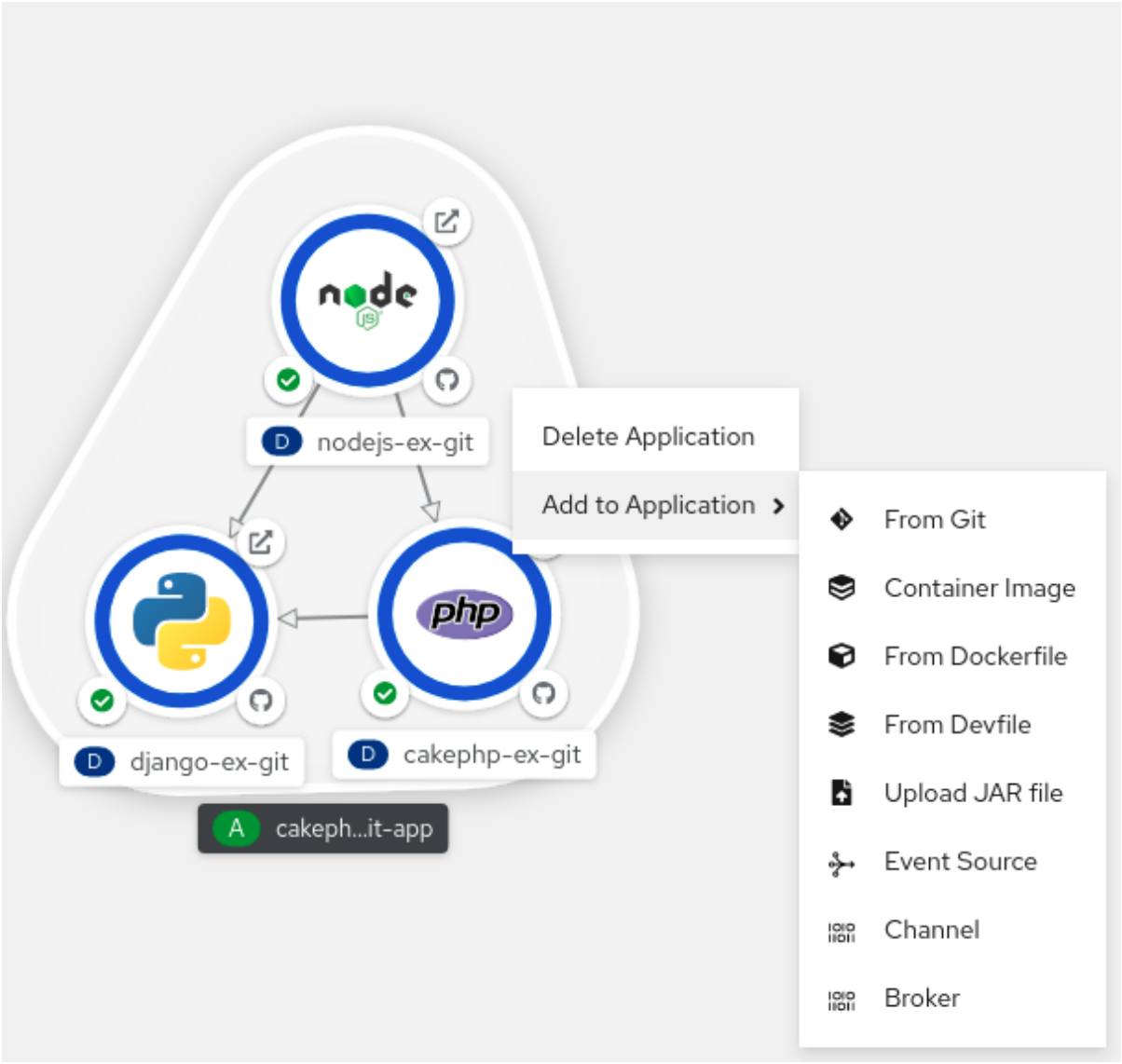
NOTE

En plus du menu contextuel, vous pouvez ajouter des services en utilisant la barre latérale ou en planant et en faisant glisser la flèche dangling du groupe d'applications.

Procédure

1. Faites un clic droit sur un groupe d'applications dans la vue topologie Graphique pour afficher le menu contextuel.

Figure 4.5. Ajouter le menu contextuel des ressources



2. Ajouter à l'application pour sélectionner une méthode d'ajout d'un service au groupe d'applications, telles que From Git, Container Image, From Dockerfile, From Devfile, Upload JAR file, Event Source, Channel ou Broker.

3. Complétez le formulaire pour la méthode que vous choisissez et cliquez sur Créer. À titre d'exemple, pour ajouter un service basé sur le code source de votre référentiel Git, choisissez la méthode From Git, remplissez le formulaire Importer à partir de Git et cliquez sur Créer.

4.8. LA SUPPRESSION DES SERVICES DE VOTRE APPLICATION

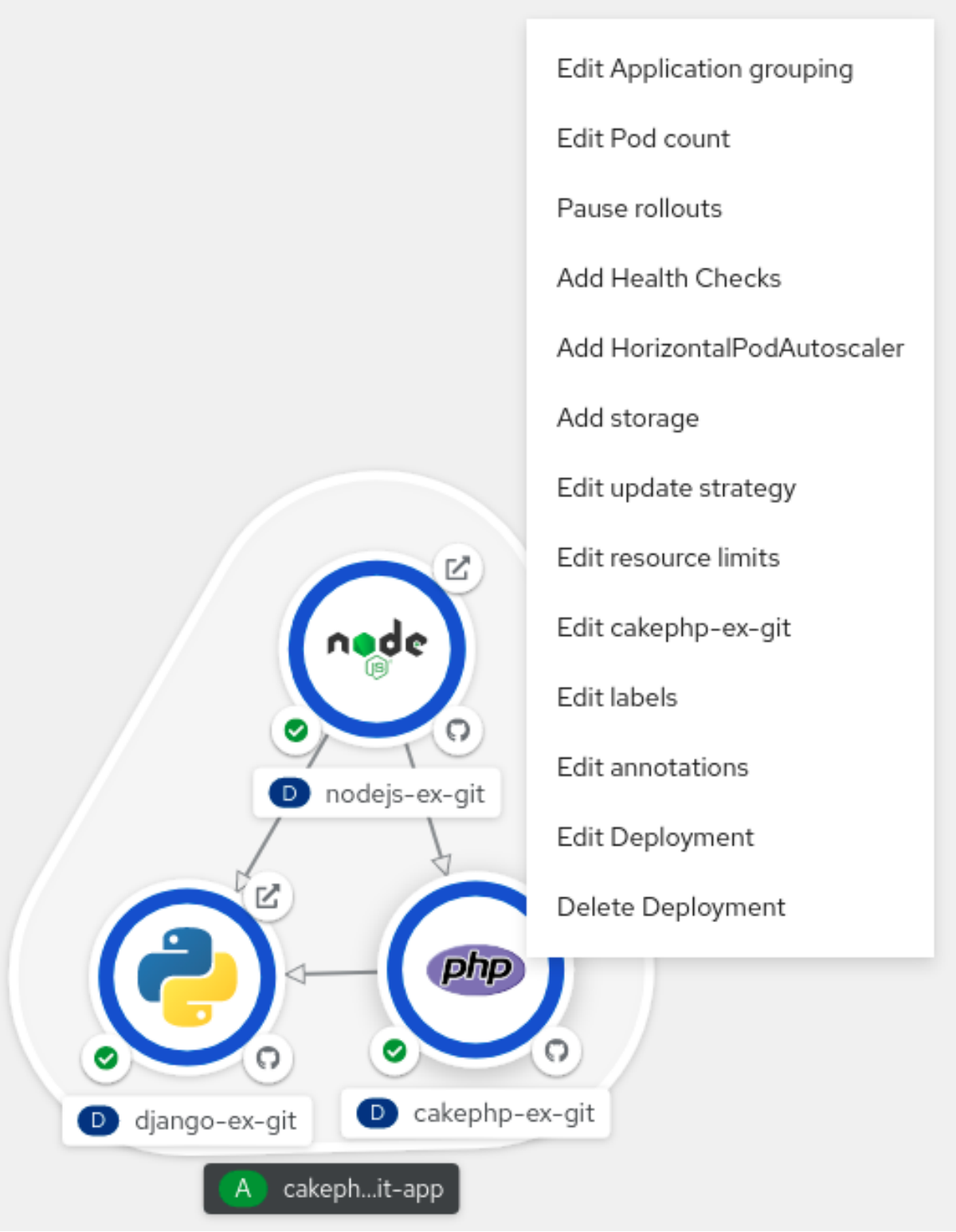
Dans la vue de topologie Graph, supprimez un service de votre application à l'aide du menu contextuel.

Procédure

1. Faites un clic droit sur un service dans un groupe d'applications dans la vue topologie Graphique pour afficher le menu contextuel.

2. Choisissez Supprimer le déploiement pour supprimer le service.

Figure 4.6. La suppression de l'option de déploiement



4.9. ÉTIQUETTES ET ANNOTATIONS UTILISÉES POUR LA VUE TOPOLOGY

La vue Topology utilise les étiquettes et les annotations suivantes:

Icône affichée dans le nœud

Les icônes dans le nœud sont définies en recherchant des icônes correspondantes à l'aide de l'étiquette app.openshift.io/runtime, suivie de l'étiquette app.kubernetes.io/name. Cette correspondance est effectuée à l'aide d'un ensemble prédéfini d'icônes.

Lien vers l'éditeur de code source ou la source

L'annotation app.openshift.io/vcs-uri est utilisée pour créer des liens vers l'éditeur de code source.

Connecteur de nœud

L'app.openshift.io/connects-to annotation est utilisé pour connecter les nœuds.

Groupement d'applications

L'étiquette app.kubernetes.io/part-of=<appname> est utilisée pour regrouper les applications, les services et les composants.

Des informations détaillées sur les étiquettes et annotations Red Hat OpenShift Service sur les applications AWS doivent être utilisées, voir les Lignes directrices pour les étiquettes et les annotations pour les applications OpenShift.

4.10. RESSOURCES SUPPLÉMENTAIRES

- Consultez Importer une base de code à partir de Git pour créer une application pour plus d'informations sur la création d'une application à partir de Git.

CHAPITRE 5. EN TRAVAILLANT AVEC HELM CHARTS

5.1. COMPRENDRE HELM

Helm est un gestionnaire de paquets logiciels qui simplifie le déploiement d'applications et de services à Red Hat OpenShift Service sur les clusters AWS.

Helm utilise un format d'emballage appelé graphiques. Le graphique Helm est une collection de fichiers qui décrit le service OpenShift Red Hat sur les ressources AWS.

Créer un graphique dans un cluster crée une instance en cours d'exécution du graphique connu sous le nom de version.

Chaque fois qu'un graphique est créé, ou qu'une version est mise à jour ou retournée, une révision incrémentale est créée.

5.1.1. Caractéristiques clés

Helm fournit la capacité de:

- Faites une recherche à travers une grande collection de graphiques stockés dans le référentiel de graphiques.
- C) Modifier les graphiques existants.
- Créez vos propres graphiques avec Red Hat OpenShift Service sur les ressources AWS ou Kubernetes.
- Emballez et partagez vos applications sous forme de graphiques.

5.1.2. Certification Red Hat des cartes Helm pour OpenShift

Choisissez de vérifier et de certifier vos graphiques Helm par Red Hat pour tous les composants que vous déployez sur le service Red Hat Red Hat OpenShift sur AWS. Les graphiques passent par un flux de travail automatisé de certification Red Hat OpenShift qui garantit la conformité à la sécurité ainsi que la meilleure intégration et expérience avec la plate-forme. La certification assure l'intégrité du graphique et garantit que le graphique Helm fonctionne de manière transparente sur les clusters Red Hat OpenShift.

5.1.3. Ressources supplémentaires

- Cliquez ici pour plus d'informations sur la façon de certifier vos graphiques Helm en tant que partenaire Red Hat, consultez la certification Red Hat des graphiques Helm pour OpenShift.
- En savoir plus sur les guides de certification OpenShift et Container pour les partenaires Red Hat, consultez le Guide des partenaires pour la certification OpenShift et Container.
- Dans la liste des graphiques, consultez le fichier de l'index Red Hat Helm.
- Les cartes disponibles sont disponibles sur la place de marché Red Hat. En savoir plus, voir Using the Red Hat Marketplace.

5.2. INSTALLATION DE HELM

La section suivante décrit comment installer Helm sur différentes plates-formes en utilisant le CLI.

En cliquant sur l'icône ? dans le coin supérieur droit et en sélectionnant Outils de ligne de commande, vous pouvez également trouver l'URL des derniers binaires du service OpenShift Red Hat sur la console web AWS.

Conditions préalables

- Go, version 1.13 ou supérieure.

5.2.1. À propos de Linux

1. Installez le binaire Linux x86_64 ou Linux amd64 Helm et ajoutez-le à votre chemin:

```
# curl -L https://mirror.openshift.com/pub/openshift-v4/clients/helm/latest/helm-linux-amd64 -
o /usr/local/bin/helm
```

2. Faire le fichier binaire exécutable:

```
# chmod +x /usr/local/bin/helm
```

3. Consultez la version installée:

```
$ helm version
```

Exemple de sortie

```
version.BuildInfo{Version:"v3.0",
GitCommit:"b31719aab7963acf4887a1c1e6d5e53378e34d93", GitTreeState:"clean",
GoVersion:"go1.13.4"}
```

5.2.2. Avec Windows 7/8

1. Téléchargez le dernier fichier .exe et mettez dans un répertoire de vos préférences.
2. Faites un clic droit sur Démarrer et cliquez sur Panneau de configuration.
3. Choisissez Système et Sécurité, puis cliquez sur Système.
4. Dans le menu à gauche, sélectionnez Paramètres des systèmes avancés et cliquez sur Variables d'environnement en bas.
5. Choisissez Path dans la section Variable et cliquez sur Modifier.
6. Cliquez sur Nouveau et tapez le chemin vers le dossier avec le fichier .exe dans le champ ou cliquez sur Parcourir et sélectionnez le répertoire, puis cliquez sur OK.

5.2.3. Avec Windows 10

1. Téléchargez le dernier fichier .exe et mettez dans un répertoire de vos préférences.
2. Cliquez sur Rechercher et tapez env ou environnement.

3. Choisissez Modifier les variables d'environnement pour votre compte.
4. Choisissez Path dans la section Variable et cliquez sur Modifier.
5. Cliquez sur Nouveau et tapez le chemin vers le répertoire avec le fichier exe dans le champ ou cliquez sur Parcourir et sélectionnez le répertoire, puis cliquez sur OK.

5.2.4. À propos de MacOS

1. Installez le binaire Helm et ajoutez-le à votre chemin:

```
# curl -L https://mirror.openshift.com/pub/openshift-v4/clients/helm/latest/helm-darwin-amd64
-o /usr/local/bin/helm
```

2. Faire le fichier binaire exécutable:

```
# chmod +x /usr/local/bin/helm
```

3. Consultez la version installée:

```
$ helm version
```

Exemple de sortie

```
version.BuildInfo{Version:"v3.0",
GitCommit:"b31719aab7963acf4887a1c1e6d5e53378e34d93", GitTreeState:"clean",
GoVersion:"go1.13.4"}
```

5.3. CONFIGURATION DE RÉFÉRENTIELS DE GRAPHIQUES HELM PERSONNALISÉS

Le catalogue des développeurs, dans la perspective Développeur de la console Web, affiche les graphiques Helm disponibles dans le cluster. Il répertorie par défaut les graphiques Helm du référentiel du graphique Red Hat OpenShift Helm. Dans la liste des graphiques, consultez le fichier de l'index Red Hat Helm.

En tant qu'administrateur de cluster, vous pouvez ajouter plusieurs référentiels de graphiques Helm, séparés du référentiel Helm par défaut, et afficher les graphiques Helm à partir de ces référentiels dans le catalogue des développeurs.

En tant qu'utilisateur régulier ou membre du projet avec les autorisations appropriées de contrôle d'accès basé sur les rôles (RBAC), vous pouvez ajouter plusieurs référentiels de graphiques Helm à portée de noms, à l'exception du référentiel Helm par défaut, et afficher les graphiques Helm de ces référentiels dans le catalogue des développeurs.

Dans la perspective Développeur de la console Web, vous pouvez utiliser la page Helm pour:

- Créez Helm Releases et Repositories à l'aide du bouton Créer.
- Créez, mettez à jour ou supprimez un référentiel de graphiques Helm.

- Consultez la liste des référentiels de graphiques Helm existants dans l'onglet Repositories, qui peuvent également être facilement distingués en tant que cluster scoped ou namespace scoped.

5.3.1. Créer des versions Helm en utilisant la perspective Développeur

Il est possible d'utiliser la perspective Développeur dans la console Web ou le CLI pour sélectionner et créer une version à partir des graphiques Helm listés dans le catalogue des développeurs. Il est possible de créer des versions Helm en installant des graphiques Helm et de les voir dans la perspective Développeur de la console Web.

Conditions préalables

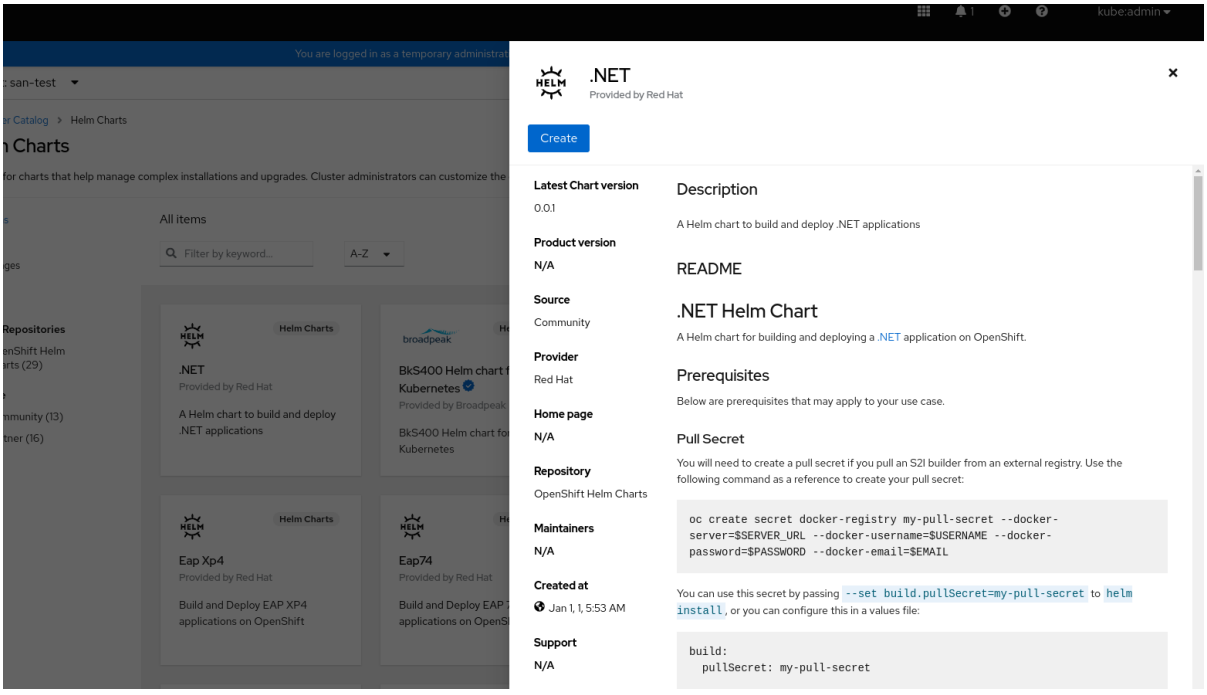
- Connectez-vous à la console Web et passez à la perspective Développeur.

Procédure

Créer des versions Helm à partir des graphiques Helm fournis dans le catalogue des développeurs:

1. Dans la perspective Développeur, accédez à la vue +Ajouter et sélectionnez un projet. Cliquez ensuite sur l'option Helm Chart pour voir tous les graphiques Helm dans le catalogue des développeurs.
2. Choisissez un graphique et lisez la description, README, et d'autres détails sur le graphique.
3. Cliquez sur **Create**.

Figure 5.1. Graphiques helm dans le catalogue des développeurs



4. Dans la page Créer Helm Release:
   a. Entrez un nom unique pour la version dans le champ Nom de libération.
   b. Choisissez la version graphique requise dans la liste déroulante de la version graphique.
   c. Configurez votre graphique Helm à l'aide de la vue de formulaire ou de la vue YAML.

NOTE

Lorsque disponible, vous pouvez basculer entre la vue YAML et la vue de formulaire. Les données sont persistantes lors de la commutation entre les vues.

d. Cliquez sur Créer pour créer une version Helm. La console Web affiche la nouvelle version dans la vue Topology.
Lorsqu'un graphique Helm a des notes de sortie, la console Web les affiche.

Lorsqu'un graphique Helm crée des charges de travail, la console Web les affiche sur la page de détails de la publication Topology ou Helm. Les charges de travail sont DaemonSet, CronJob, Pod, Deployment et DeploymentConfig.

e. Consultez la version de Helm nouvellement créée dans la page Helm Releases.

Il est possible de mettre à niveau, de faire reculer ou de supprimer une version Helm en utilisant le bouton Actions du panneau latéral ou en faisant un clic droit sur une version Helm.

5.3.2. En utilisant Helm dans le terminal web

Il est possible d'utiliser Helm en accédant au terminal web dans la perspective Développeur de la console Web.

5.3.3. Création d'un graphique Helm personnalisé sur Red Hat OpenShift Service sur AWS

Procédure

1. Créer un nouveau projet:

```
$ oc new-project nodejs-ex-k
```

2. Cliquez ici pour télécharger un exemple Node.js qui contient Red Hat OpenShift Service sur les objets AWS:

```
$ git clone https://github.com/redhat-developer/redhat-helm-charts
```

3. Allez dans le répertoire avec l'exemple de graphique:

```
$ cd redhat-helm-charts/alpha/nodejs-ex-k/
```

4. Modifiez le fichier Chart.yaml et ajoutez une description de votre graphique:

```
apiVersion: v2 (1)
name: nodejs-ex-k (2)
description: A Helm chart for OpenShift (3)
icon: https://static.redhat.com/libs/redhat/brand-assets/latest/corp/logo.svg (4)
version: 0.2.1 (5)
```

(1) La version de l'API graphique. Il devrait être v2 pour les graphiques Helm qui nécessitent au moins Helm 3.

2. Le nom de votre carte.
3. La description de votre tableau.
4. L'URL d'une image à utiliser comme icône.
5. La version de votre graphique selon la spécification de version sémantique (SemVer) 2.0.0.

5. Assurez-vous que le graphique est correctement formaté:

```
$ helm lint
```

Exemple de sortie

```
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
```

6. Accédez au niveau précédent du répertoire:

```
$ cd ..
```

7. Installez le graphique:

```
$ helm install nodejs-chart nodejs-ex-k
```

8. Assurez-vous que le graphique s'est installé avec succès:

```
$ helm list
```

Exemple de sortie

```
NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION
nodejs-chart nodejs-ex-k 1 2019-12-05 15:06:51.379134163 -0500 EST deployed nodejs-
0.1.0  1.16.0
```

5.3.4. Filtrer les graphiques Helm par leur niveau de certification

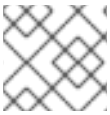
Il est possible de filtrer les graphiques Helm en fonction de leur niveau de certification dans le catalogue des développeurs.

Procédure

1. Dans la perspective Développeur, accédez à la vue +Ajouter et sélectionnez un projet.
2. Dans la tuile Catalogue des développeurs, sélectionnez l'option Helm Chart pour voir tous les graphiques Helm dans le catalogue des développeurs.
3. Les filtres se trouvent à gauche de la liste des graphiques Helm pour filtrer les graphiques requis:
   - Le filtre Repositories de graphique permet de filtrer les graphiques fournis par les

Le filtre Repositories de graphique permet de filtrer les graphiques fournis par les graphiques de certification Red Hat ou OpenShift Helm Charts.

- Le filtre Source permet de filtrer les graphiques provenant des partenaires, de la communauté ou du chapeau rouge. Les graphiques certifiés sont indiqués avec l'icône ().



NOTE

Le filtre Source ne sera pas visible lorsqu'il n'y a qu'un seul type de fournisseur.

Désormais, vous pouvez sélectionner le graphique requis et l'installer.

5.4. EN TRAVAILLANT AVEC LES VERSIONS DE HELM

Dans la console Web, vous pouvez utiliser la perspective Développeur pour mettre à jour, réduire ou supprimer une version Helm.

5.4.1. Conditions préalables

- Connectez-vous à la console Web et passez à la perspective Développeur.

5.4.2. Amélioration d'une version Helm

Il est possible de mettre à niveau une version Helm pour passer à une nouvelle version graphique ou de mettre à jour votre configuration de version.

Procédure

1. Dans la vue Topologie, sélectionnez la version Helm pour voir le panneau latéral.
2. Cliquez sur Actions → Mettre à niveau Helm Release.
3. Dans la page Mise à niveau Helm Release, sélectionnez la version graphique vers laquelle vous souhaitez mettre à niveau, puis cliquez sur Mise à niveau pour créer une autre version de Helm. La page Helm Releases affiche les deux révisions.

5.4.3. Faire reculer une libération de Helm

En cas d'échec d'une version, vous pouvez retourner la version Helm à une version précédente.

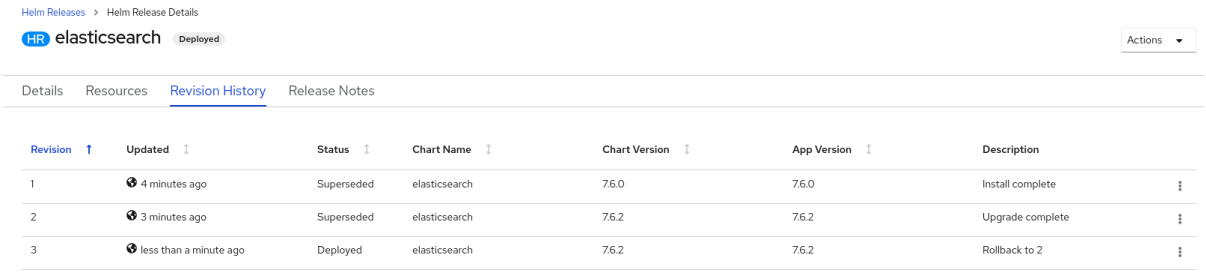
Procédure

Faire reculer une version en utilisant la vue Helm:

1. Dans la perspective Développeur, accédez à la vue Helm pour voir les versions Helm dans l'espace de noms.
2. Cliquez sur le menu Options adjacentes à la version listée, puis sélectionnez Retourner.
3. Dans la page Rollback Helm Release, sélectionnez la révision à laquelle vous souhaitez revenir et cliquez sur Retourner.
4. Dans la page Helm Releases, cliquez sur le graphique pour voir les détails et les ressources de cette version.

5. Allez dans l'onglet Historique des révisions pour voir toutes les révisions du graphique.

Figure 5.2. Historique de révision de la barre



6. Au besoin, vous pouvez utiliser le menu Options attenant à une révision particulière et sélectionner la révision vers laquelle revenir.

5.4.4. La suppression d'une version Helm

Procédure

1. Dans la vue Topologie, cliquez avec le bouton droit sur la version Helm et sélectionnez Supprimer Helm Release.

2. Dans l'invite de confirmation, entrez le nom du graphique et cliquez sur Supprimer.

CHAPITRE 6. DÉPLOIEMENTS

6.1. DOMAINES PERSONNALISÉS POUR LES APPLICATIONS

AVERTISSEMENT

À partir de Red Hat OpenShift Service sur AWS 4.14, l'opérateur de domaine personnalisé est obsolète. Gérer Ingress dans Red Hat OpenShift Service sur AWS 4.14, utilisez l'opérateur Ingress. La fonctionnalité est inchangée pour Red Hat OpenShift Service sur AWS 4.13 et versions antérieures.

Il est possible de configurer un domaine personnalisé pour vos applications. Les domaines personnalisés sont des domaines génériques spécifiques qui peuvent être utilisés avec Red Hat OpenShift Service sur les applications AWS.

6.1.1. Configuration de domaines personnalisés pour les applications

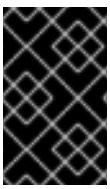
Les domaines de premier niveau (TLD) appartiennent au client qui exploite le service OpenShift Red Hat sur AWS cluster. L'opérateur de domaines personnalisés met en place un nouveau contrôleur d'entrée avec un certificat personnalisé en tant qu'opération de deuxième jour. L'enregistrement DNS public pour ce contrôleur d'entrée peut ensuite être utilisé par un DNS externe pour créer un enregistrement CNAME wildcard pour une utilisation avec un domaine personnalisé.

NOTE

Les domaines API personnalisés ne sont pas pris en charge car Red Hat contrôle le domaine API. Cependant, les clients peuvent changer leurs domaines d'application. Dans le cas des domaines personnalisés privés avec un contrôleur privé IngressController, définissez .spec.scope sur Interne dans le CustomDomain CR.

Conditions préalables

- Compte utilisateur doté de privilèges dédiés à l'administration
- Domaine unique ou générique, tel que *.apps.<company_name>.io
- Certificat personnalisé ou certificat sur mesure, tel que CN=*.apps.<company_name>.io
- Accès à un cluster avec la dernière version du CLI oc installé



IMPORTANT

Il ne faut pas utiliser les noms réservés par défaut ou les applications*, telles que les applications ou les applications2, dans la section métadonnées/nom: section CustomDomain CR.

Procédure

1. Créez un nouveau secret TLS à partir d'une clé privée et d'un certificat public, où fullchain.pem et privkey.pem sont vos certificats wildcard publics ou privés.

   Exemple :

   ```
   $ oc create secret tls <name>-tls --cert=fullchain.pem --key=privkey.pem -n <my_project>
   ```

2. Créer une nouvelle ressource personnalisée CustomDomain (CR):

   Exemple <company_name>-custom-domain.yaml

   ```
   apiVersion: managed.openshift.io/v1alpha1
   kind: CustomDomain
   metadata:
    name: <company_name>
   spec:
    domain: apps.<company_name>.io (1)
    scope: External
    loadBalancerType: Classic (2)
    certificate:
     name: <name>-tls (3)
     namespace: <my_project>
    routeSelector: (4)
     matchLabels:
      route: acme
    namespaceSelector: (5)
     matchLabels:
      type: sharded
   ```

   (1) Le domaine personnalisé.

   (2) Le type d'équilibreur de charge pour votre domaine personnalisé. Ce type peut être le classique par défaut ou la NLB si vous utilisez un équilibreur de charge réseau.

   (3) Le secret créé à l'étape précédente.

   (4) Facultatif: Filtre l'ensemble des itinéraires desservis par l'entrée CustomDomain. En l'absence de valeur, la valeur par défaut n'est pas de filtrage.

   (5) Facultatif: Filtre l'ensemble d'espaces de noms desservis par l'entrée CustomDomain. En l'absence de valeur, la valeur par défaut n'est pas de filtrage.

3. Appliquer le CR:

   Exemple :

   ```
   $ oc apply -f <company_name>-custom-domain.yaml
   ```

4. Bénéficiez du statut de votre CR nouvellement créé:

   ```
   $ oc get customdomains
   ```

   Exemple de sortie

```
NAME ENDPOINT DOMAIN STATUS
<company_name> xxrywp.<company_name>.cluster-01.opln.s1.openshiftapps.com
*.apps.<company_name>.io Ready
```

5. À l'aide de la valeur du point de terminaison, ajoutez un nouvel enregistrement générique CNAME à votre fournisseur DNS géré, tel que Route53.

   Exemple :

   ```
   *.apps.<company_name>.io -> xxrywp.<company_name>.cluster-
   01.opln.s1.openshiftapps.com
   ```

6. Créez une nouvelle application et exposez-la:

   Exemple :

   ```
   $ oc new-app --docker-image=docker.io/openshift/hello-openshift -n my-project
   ```

   ```
   $ oc create route <route_name> --service=hello-openshift hello-openshift-tls --hostname
   hello-openshift-tls-my-project.apps.<company_name>.io -n my-project
   ```

   ```
   $ oc get route -n my-project
   ```

   ```
   $ curl https://hello-openshift-tls-my-project.apps.<company_name>.io
   Hello OpenShift!
   ```

Résolution de problèmes

- Erreur de création de TLS secret
- Dépannage: CustomDomain dans l'état NotReady

6.1.2. Le renouvellement d'un certificat pour les domaines personnalisés

Il est possible de renouveler les certificats auprès de l'opérateur de domaines personnalisés (CDO) en utilisant l'outil oc CLI.

Conditions préalables

- La dernière version de l'outil oc CLI est installée.

Procédure

1. Créer un nouveau secret

   ```
   $ oc create secret tls <secret-new> --cert=fullchain.pem --key=privkey.pem -n <my_project>
   ```

2. Correction CustomDomain CR

   ```
   $ oc patch customdomain <company_name> --type='merge' -p '{"spec":{"certificate":
   {"name":"<secret-new>"}}}'
   ```

3. Effacer le vieux secret

```
$ oc delete secret <secret-old> -n <my_project>
```

Résolution de problèmes

- Erreur de création de TLS secret

6.2. COMPRENDRE LES DÉPLOIEMENTS

Les objets API Déploiement et DéploiementConfig dans Red Hat OpenShift Service sur AWS fournissent deux méthodes similaires mais différentes pour une gestion fine sur les applications utilisateur courantes. Ils sont composés des objets API distincts suivants:

- L'objet Déploiement ou DéploiementConfig décrit l'état souhaité d'un composant particulier de l'application comme un modèle de pod.
- Les objets de déploiement impliquent un ou plusieurs ensembles de répliques, qui contiennent un enregistrement point dans le temps de l'état d'un déploiement en tant que modèle de pod. De même, les objets DeploymentConfig impliquent un ou plusieurs contrôleurs de réplication, qui ont précédé les ensembles de répliques.
- C) un ou plusieurs pods, qui représentent une instance d'une version particulière d'une application.

Les objets de déploiement, sauf si vous avez besoin d'une fonctionnalité ou d'un comportement spécifique fournis par les objets DeploymentConfig.

IMPORTANT

Depuis Red Hat OpenShift Service sur AWS 4.14, les objets DeploymentConfig sont obsolètes. Les objets DeploymentConfig sont toujours pris en charge, mais ne sont pas recommandés pour les nouvelles installations. Il n'y aura que des problèmes critiques et liés à la sécurité.

Au lieu de cela, utilisez des objets de déploiement ou une autre alternative pour fournir des mises à jour déclaratives pour les pods.

6.2.1. Éléments constitutifs d'un déploiement

Les déploiements et les configurations de déploiement sont activés par l'utilisation d'objets API Kubernetes natives ReplicaSet et ReplicationController, respectivement, comme blocs de construction.

Les utilisateurs n'ont pas à manipuler des ensembles de répliques, des contrôleurs de réplication ou des pods appartenant aux objets Deployment ou DeploymentConfig. Les systèmes de déploiement s'assurent que les changements sont propagés de manière appropriée.

ASTUCE

Lorsque les stratégies de déploiement existantes ne sont pas adaptées à votre cas d'utilisation et que vous devez exécuter des étapes manuelles pendant le cycle de vie de votre déploiement, alors vous devriez envisager de créer une stratégie de déploiement personnalisée.

Les sections suivantes fournissent de plus amples détails sur ces objets.

6.2.1.1. Ensembles de répliques

A ReplicaSet est un objet API Kubernetes natif qui garantit qu'un nombre spécifié de répliques de pod s'exécute à un moment donné.

NOTE

Il suffit d'utiliser des ensembles de répliques si vous avez besoin d'orchestration de mise à jour personnalisée ou si vous n'avez pas besoin de mises à jour du tout. Autrement, utilisez des déploiements. Les ensembles de répliques peuvent être utilisés indépendamment, mais sont utilisés par des déploiements pour orchestrer la création de pod, la suppression et les mises à jour. Les déploiements gèrent automatiquement leurs ensembles de répliques, fournissent des mises à jour déclaratives aux pods et n'ont pas à gérer manuellement les ensembles de répliques qu'ils créent.

Ce qui suit est un exemple de définition de ReplicaSet:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend-1
  labels:
    tier: frontend
spec:
  replicas: 3
  selector: 1
    matchLabels: 2
      tier: frontend
    matchExpressions: 3
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - image: openshift/hello-openshift
        name: helloworld
        ports:
        - containerPort: 8080
          protocol: TCP
      restartPolicy: Always
```

1. Il s'agit d'une requête d'étiquette sur un ensemble de ressources. Le résultat de matchLabels et matchExpressions sont logiquement associés.
2. Le sélecteur basé sur l'égalité pour spécifier les ressources avec des étiquettes qui correspondent au sélecteur.
3. Définir le sélecteur pour filtrer les touches. Cela sélectionne toutes les ressources avec clé égale à niveau et valeur égale à frontend.

6.2.1.2. Contrôleurs de réplication

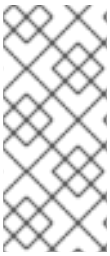
À l'instar d'un ensemble de répliques, un contrôleur de réplication s'assure qu'un nombre spécifié de répliques d'un pod s'exécute à tout moment. Lorsque les pods sortent ou sont supprimés, le contrôleur de réplication instancie davantage jusqu'au nombre défini. De même, s'il y a plus de fonctionnement que souhaité, il supprime autant que nécessaire pour correspondre au montant défini. La différence entre un ensemble de répliques et un contrôleur de réplication est qu'un ensemble de répliques prend en charge les exigences de sélecteur basées sur des ensembles alors qu'un contrôleur de réplication ne prend en charge que les exigences de sélecteur basées sur l'égalité.

La configuration du contrôleur de réplication consiste en:

- Le nombre de répliques souhaitées, qui peuvent être ajustées au moment de l'exécution.
- Définition de Pod à utiliser lors de la création d'un pod répliqué.
- D'un sélecteur pour identifier les gousses gérées.

Le sélecteur est un ensemble d'étiquettes attribuées aux pods qui sont gérées par le contrôleur de réplication. Ces étiquettes sont incluses dans la définition de Pod que le contrôleur de réplication instantanée. Le contrôleur de réplication utilise le sélecteur pour déterminer combien d'instances de la gousse sont déjà en cours d'exécution afin de s'ajuster au besoin.

Le contrôleur de réplication n'effectue pas de mise à l'échelle automatique basée sur la charge ou le trafic, car il ne suit pas non plus. Cela nécessite plutôt que son nombre de répliques soit ajusté par un auto-scaleur externe.



NOTE

Faites appel à un DeploymentConfig pour créer un contrôleur de réplication au lieu de créer des contrôleurs de réplication directement.

Lorsque vous avez besoin d'orchestration personnalisée ou que vous n'avez pas besoin de mises à jour, utilisez des ensembles de répliques au lieu de contrôleurs de réplication.

Ce qui suit est une définition d'un contrôleur de réplication:

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: frontend-1
spec:
  replicas: 1 (1)
  selector: (2)
    name: frontend
  template: (3)
    metadata:
      labels: (4)
        name: frontend (5)
    spec:
      containers:
      - image: openshift/hello-openshift
        name: helloworld
        ports:
        - containerPort: 8080
          protocol: TCP
      restartPolicy: Always
```

1. Le nombre d'exemplaires de la gousse à exécuter.
2. Le sélecteur d'étiquette de la gousse à exécuter.
3. C'est un modèle pour le pod que le contrôleur crée.
4. Les étiquettes sur la gousse doivent inclure celles du sélecteur d'étiquettes.
5. La longueur maximale du nom après avoir étendu n'importe quel paramètre est de 63 caractères.

6.2.2. Déploiements

Kubernetes fournit un type d'objet API native de première classe dans Red Hat OpenShift Service sur AWS appelé Déploiement. Les objets de déploiement décrivent l'état souhaité d'un composant particulier d'une application comme un modèle de pod. Les déploiements créent des ensembles de répliques, qui orchestrent les cycles de vie des pod.

À titre d'exemple, la définition de déploiement suivante crée une réplique pour mettre en place une pod hello-openshift:

Définition du déploiement

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-openshift
  template:
    metadata:
      labels:
        app: hello-openshift
    spec:
      containers:
      - name: hello-openshift
        image: openshift/hello-openshift:latest
        ports:
        - containerPort: 80
```

6.2.3. Déploiement des objetsConfig

IMPORTANT

Depuis Red Hat OpenShift Service sur AWS 4.14, les objets DeploymentConfig sont obsolètes. Les objets DeploymentConfig sont toujours pris en charge, mais ne sont pas recommandés pour les nouvelles installations. Il n'y aura que des problèmes critiques et liés à la sécurité.

Au lieu de cela, utilisez des objets de déploiement ou une autre alternative pour fournir des mises à jour déclaratives pour les pods.

En s'appuyant sur des contrôleurs de réplication, Red Hat OpenShift Service sur AWS ajoute une prise en charge étendue du développement logiciel et du cycle de vie du déploiement avec le concept d'objets DeploymentConfig. Dans le cas le plus simple, un objet DeploymentConfig crée un nouveau contrôleur de réplication et permet de démarrer des pods.

Cependant, le service OpenShift Red Hat sur les déploiements AWS à partir d'objets DeploymentConfig offre également la possibilité de passer d'un déploiement existant d'une image à une nouvelle et de définir également des crochets à exécuter avant ou après la création du contrôleur de réplication.

Le système de déploiement DeploymentConfig fournit les capacités suivantes:

- L'objet DeploymentConfig, qui est un modèle pour l'exécution d'applications.
- Déclencheurs qui génèrent des déploiements automatisés en réponse à des événements.
- Des stratégies de déploiement personnalisables par l'utilisateur pour passer de la version précédente à la nouvelle version. La stratégie s'exécute à l'intérieur d'un pod communément appelé processus de déploiement.
- Ensemble de crochets (hameçons du cycle de vie) pour exécuter un comportement personnalisé dans différents points pendant le cycle de vie d'un déploiement.
- La version de votre application pour prendre en charge les redémarrages manuellement ou automatiquement en cas de défaillance de déploiement.
- L'échelle manuelle de réplication et l'autoscaling.

Lorsque vous créez un objet DeploymentConfig, un contrôleur de réplication est créé représentant le modèle de pod de l'objet DeploymentConfig. En cas de changement de déploiement, un nouveau contrôleur de réplication est créé avec le dernier modèle de pod, et un processus de déploiement s'exécute pour réduire l'ancien contrôleur de réplication et faire évoluer le nouveau.

Les instances de votre application sont automatiquement ajoutées et supprimées des équilibreurs de charge de service et des routeurs au fur et à mesure qu'ils sont créés. Aussi longtemps que votre application prend en charge l'arrêt gracieux lorsqu'elle reçoit le signal TERM, vous pouvez vous assurer que les connexions utilisateur en cours d'exécution ont une chance de compléter normalement.

L'objet Red Hat OpenShift sur AWS DeploymentConfig définit les détails suivants:

1. Les éléments d'une définition de RéplicationController.
2. Déclencheurs pour créer un nouveau déploiement automatiquement.
3. La stratégie de transition entre les déploiements.
4. Des crochets de cycle de vie.

Chaque fois qu'un déploiement est déclenché, que ce soit manuellement ou automatiquement, un pod de déploiement gère le déploiement (y compris la mise à l'échelle de l'ancien contrôleur de réplication, la mise à l'échelle du nouveau et l'exécution de crochets). Le pod de déploiement reste pour une durée indéterminée après avoir terminé le déploiement pour conserver ses journaux de déploiement. Lorsqu'un déploiement est remplacé par un autre, le contrôleur de réplication précédent est conservé pour permettre un retour facile si nécessaire.

Exemple DeploymentConfig Définition

```
apiVersion: apps.openshift.io/v1
```

```
kind: DeploymentConfig
metadata:
  name: frontend
spec:
  replicas: 5
  selector:
    name: frontend
  template: { ... }
  triggers:
  - type: ConfigChange (1)
  - imageChangeParams:
      automatic: true
      containerNames:
      - helloworld
      from:
        kind: ImageStreamTag
        name: hello-openshift:latest
    type: ImageChange (2)
  strategy:
    type: Rolling (3)
```

1. Le déclencheur d'un changement de configuration se traduit par un nouveau contrôleur de réplication chaque fois que des modifications sont détectées dans le modèle de pod de la configuration de déploiement.

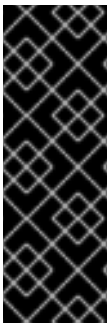
2. Le déclenchement d'un changement d'image provoque la création d'un nouveau déploiement chaque fois qu'une nouvelle version de l'image de support est disponible dans le flux d'images nommé.

3. La stratégie Rolling par défaut permet une transition sans temps d'arrêt entre les déploiements.

6.2.4. Comparaison des objets Déploiement et DéploiementConfig

Les objets de déploiement Kubernetes et le service OpenShift Red Hat sur les objets DeploymentConfig fournis par AWS sont pris en charge dans Red Hat OpenShift Service sur AWS; cependant, il est recommandé d'utiliser des objets de déploiement à moins que vous ayez besoin d'une fonctionnalité ou d'un comportement spécifique fourni par les objets DeploymentConfig.

Les sections suivantes donnent plus de détails sur les différences entre les deux types d'objets pour vous aider à décider quel type utiliser.



IMPORTANT

Depuis Red Hat OpenShift Service sur AWS 4.14, les objets DeploymentConfig sont obsolètes. Les objets DeploymentConfig sont toujours pris en charge, mais ne sont pas recommandés pour les nouvelles installations. Il n'y aura que des problèmes critiques et liés à la sécurité.

Au lieu de cela, utilisez des objets de déploiement ou une autre alternative pour fournir des mises à jour déclaratives pour les pods.

6.2.4.1. Conception

Les propriétés du théorème CAP que chaque conception a choisi pour le processus de déploiement

constituent une différence importante entre le déploiement et le déploiement des objetsConfig. Les objets DeploymentConfig préfèrent la cohérence, tandis que les objets de déploiement prennent la disponibilité plutôt que la cohérence.

Dans le cas des objets DeploymentConfig, si un nœud exécutant un pod de déploiement diminue, il ne sera pas remplacé. Le processus attend jusqu'à ce que le nœud revienne en ligne ou soit supprimé manuellement. La suppression manuelle du nœud supprime également le pod correspondant. Cela signifie que vous ne pouvez pas supprimer la gousse pour décoller le déploiement, car le kubelet est responsable de la suppression de la gousse associée.

Cependant, les déploiements de déploiement sont conduits par un gestionnaire de contrôleur. Le gestionnaire de contrôleur fonctionne en mode haute disponibilité sur les maîtres et utilise des algorithmes d'élection leader pour valoriser la disponibilité par rapport à la cohérence. En cas d'échec, il est possible pour d'autres maîtres d'agir sur le même déploiement en même temps, mais ce problème sera réconcilié peu de temps après l'échec.

6.2.4.2. Caractéristiques spécifiques au déploiement

Le roulement
Le processus de déploiement des objets de déploiement est piloté par une boucle de contrôleur, contrairement aux objets DeploymentConfig qui utilisent des pods de déploiement pour chaque nouveau déploiement. Cela signifie que l'objet Déploiement peut avoir autant d'ensembles de répliques actives que possible, et finalement le contrôleur de déploiement réduira tous les anciens ensembles de répliques et mettra à l'échelle le plus récent.

Les objets DeploymentConfig peuvent avoir au plus un pod de déploiement en cours d'exécution, sinon plusieurs déployeurs pourraient entrer en conflit lorsqu'ils tentent de développer ce qu'ils pensent être le plus récent contrôleur de réplication. De ce fait, seuls deux contrôleurs de réplication peuvent être actifs à tout moment. En fin de compte, cela entraîne des déploiements rapides plus rapides pour les objets de déploiement.

Échelle proportionnelle
Étant donné que le contrôleur de déploiement est la seule source de vérité pour les tailles de nouveaux et anciens ensembles de répliques appartenant à un objet de déploiement, il peut faire évoluer les déploiements en cours. Des répliques supplémentaires sont distribuées proportionnellement en fonction de la taille de chaque ensemble de répliques.

Les objets DeploymentConfig ne peuvent pas être mis à l'échelle lorsqu'un déploiement est en cours, car le contrôleur aura des problèmes avec le processus de déploiement de la taille du nouveau contrôleur de réplication.

La pause à mi-démarrage
Les déploiements peuvent être mis en pause à tout moment, ce qui signifie que vous pouvez également mettre en pause les déploiements en cours. Cependant, vous ne pouvez actuellement pas mettre en pause les pods de déploiement; si vous essayez de mettre un terme à un déploiement au milieu d'un déploiement, le processus de déploiement n'est pas affecté et se poursuit jusqu'à ce qu'il soit terminé.

6.2.4.3. DeploymentConfig caractéristiques spécifiques à l'objet

Les retours automatiques
Actuellement, les déploiements ne prennent pas en charge le retour automatique à la dernière réplique déployée avec succès en cas d'échec.

Déclencheurs
Les déploiements ont un déclencheur de modification de configuration implicite en ce sens que chaque changement dans le modèle de pod d'un déploiement déclenche automatiquement un nouveau

déploiement. Lorsque vous ne voulez pas de nouveaux déploiements sur les modifications de modèles de pod, arrêtez le déploiement:

```
$ oc rollout pause deployments/<name>
```

Crochets de cycle de vie
Les déploiements ne prennent pas encore en charge les crochets du cycle de vie.

Des stratégies personnalisées
Les déploiements ne prennent pas en charge les stratégies de déploiement personnalisées spécifiées par l'utilisateur.

6.3. GESTION DES PROCESSUS DE DÉPLOIEMENT

6.3.1. Gestion des objets de déploiementConfig

IMPORTANT

Depuis Red Hat OpenShift Service sur AWS 4.14, les objets DeploymentConfig sont obsolètes. Les objets DeploymentConfig sont toujours pris en charge, mais ne sont pas recommandés pour les nouvelles installations. Il n'y aura que des problèmes critiques et liés à la sécurité.

Au lieu de cela, utilisez des objets de déploiement ou une autre alternative pour fournir des mises à jour déclaratives pour les pods.

Les objets DeploymentConfig peuvent être gérés à partir du service Red Hat OpenShift sur la page Charges de travail de la console web AWS ou en utilisant le CLI oc. Les procédures suivantes montrent l'utilisation de CLI sauf indication contraire.

6.3.1.1. Démarrage d'un déploiement

Démarrez un déploiement pour commencer le processus de déploiement de votre application.

Procédure

1. Afin de démarrer un nouveau processus de déploiement à partir d'un objet DeploymentConfig existant, exécutez la commande suivante:

```
$ oc rollout latest dc/<name>
```

NOTE

Lorsqu'un processus de déploiement est déjà en cours, la commande affiche un message et un nouveau contrôleur de réplication ne sera pas déployé.

6.3.1.2. Affichage d'un déploiement

Il est possible d'afficher un déploiement pour obtenir des informations de base sur toutes les révisions disponibles de votre application.

Procédure

1. Afin d'afficher des détails sur tous les contrôleurs de réplication récemment créés pour l'objet DeploymentConfig fourni, y compris tout processus de déploiement en cours d'exécution, exécutez la commande suivante:

```
$ oc rollout history dc/<name>
```

2. Afin d'afficher les détails spécifiques à une révision, ajouter le drapeau --révision:

```
$ oc rollout history dc/<name> --revision=1
```

3. Afin d'obtenir des informations plus détaillées sur un objet DeploymentConfig et sa dernière révision, utilisez la commande de description oc:

```
$ oc describe dc <name>
```

6.3.1.3. La réessayer d'un déploiement

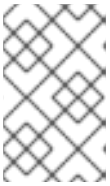
Lorsque la révision actuelle de votre objet DeploymentConfig n'a pas été déployée, vous pouvez redémarrer le processus de déploiement.

Procédure

1. De redémarrer un processus de déploiement défaillant:

```
$ oc rollout retry dc/<name>
```

Lorsque la dernière révision de celle-ci a été déployée avec succès, la commande affiche un message et le processus de déploiement n'est pas récupéré.



NOTE

La réessayer d'un déploiement redémarre le processus de déploiement et ne crée pas de nouvelle révision du déploiement. Le contrôleur de réplication redémarré a la même configuration qu'il avait quand il a échoué.

6.3.1.4. Faire reculer un déploiement

Les redémarrages retournent une application à une révision précédente et peuvent être effectués à l'aide de l'API REST, du CLI ou de la console Web.

Procédure

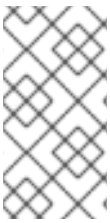
1. Afin de revenir à la dernière révision réussie de votre configuration:

```
$ oc rollout undo dc/<name>
```

Le modèle de l'objet DeploymentConfig est retourné pour correspondre à la révision de déploiement spécifiée dans la commande d'annulation, et un nouveau contrôleur de réplication est lancé. Dans le cas où aucune révision n'est spécifiée avec --to-révision, la dernière révision déployée avec succès est utilisée.

2. Les déclencheurs de changement d'image sur l'objet DeploymentConfig sont désactivés dans le cadre du rollback afin d'éviter le démarrage accidentel d'un nouveau processus de déploiement peu après la fin du retour.
   Afin de réactiver les déclencheurs de changement d'image:

```
$ oc set triggers dc/<name> --auto
```



NOTE

Les configurations de déploiement prennent également en charge le retour automatique à la dernière révision réussie de la configuration au cas où le dernier processus de déploiement échouerait. Dans ce cas, le dernier modèle qui n'a pas réussi à déployer reste intact par le système et c'est aux utilisateurs de fixer leurs configurations.

6.3.1.5. Exécution des commandes à l'intérieur d'un conteneur

Il est possible d'ajouter une commande à un conteneur, ce qui modifie le comportement de démarrage du conteneur en annulant l'ENTRYPOINT de l'image. Ceci est différent d'un crochet de cycle de vie, qui peut être exécuté une fois par déploiement à un moment spécifié.

Procédure

1. Ajoutez les paramètres de commande au champ spec de l'objet DeploymentConfig. Il est également possible d'ajouter un champ args, qui modifie la commande (ou l'ENTRYPOINT si la commande n'existe pas).

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
spec:
  template:
# ...
    spec:
     containers:
     - name: <container_name>
       image: 'image'
       command:
         - '<command>'
       args:
         - '<argument_1>'
         - '<argument_2>'
         - '<argument_3>'
```

   À titre d'exemple, pour exécuter la commande java avec les arguments -jar et /opt/app-root/springboots2idemo.jar:

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
spec:
```

```
  template:
# ...
    spec:
      containers:
        - name: example-spring-boot
          image: 'image'
          command:
            - java
          args:
            - '-jar'
            - /opt/app-root/springboots2idemo.jar
# ...
```

6.3.1.6. Affichage des journaux de déploiement

Procédure

1. Diffuser les journaux de la dernière révision pour un objet de déploiement donnéConfig:

```
$ oc logs -f dc/<name>
```

   En cas d'exécution ou d'échec de la dernière révision, la commande renvoie les journaux du processus responsable du déploiement de vos pods. En cas de succès, il renvoie les journaux d'un pod de votre application.

2. En outre, vous pouvez afficher les journaux des anciens processus de déploiement défaillants, si et seulement si ces processus (anciens contrôleurs de réplication et leurs pods de déploiement) existent et n'ont pas été élavés ou supprimés manuellement:

```
$ oc logs --version=1 dc/<name>
```

6.3.1.7. Déclencheurs de déploiement

L'objet DeploymentConfig peut contenir des déclencheurs, ce qui entraîne la création de nouveaux processus de déploiement en réponse aux événements à l'intérieur du cluster.

AVERTISSEMENT

Lorsqu'aucun déclencheur n'est défini sur un objet DeploymentConfig, un déclencheur de modification de configuration est ajouté par défaut. Lorsque les déclencheurs sont définis comme un champ vide, les déploiements doivent être démarrés manuellement.

Configurer les déclencheurs de déploiement de changement
Le déclencheur de modification de configuration se traduit par un nouveau contrôleur de réplication chaque fois que des modifications de configuration sont détectées dans le modèle de pod de l'objet DeploymentConfig.

NOTE

Lorsqu'un déclencheur de modification de configuration est défini sur un objet DeploymentConfig, le premier contrôleur de réplication est automatiquement créé peu de temps après la création de l'objet DeploymentConfig et il n'est pas mis en pause.

Déclencheur de déploiement de changement de configuration

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
spec:
# ...
  triggers:
    - type: "ConfigChange"
```

Déclencheurs de déploiement de changement d'image

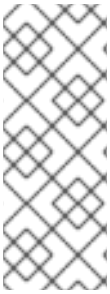
Le déclencheur de changement d'image entraîne un nouveau contrôleur de réplication chaque fois que le contenu d'une balise de flux d'image change (lorsqu'une nouvelle version de l'image est poussée).

Déclencheur de déploiement de changement d'image

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
spec:
# ...
  triggers:
    - type: "ImageChange"
      imageChangeParams:
        automatic: true (1)
        from:
          kind: "ImageStreamTag"
          name: "origin-ruby-sample:latest"
          namespace: "myproject"
        containerNames:
          - "helloworld"
```

(1) Lorsque le champ imageChangeParams.automatic est défini sur false, le déclencheur est désactivé.

Avec l'exemple ci-dessus, lorsque la dernière valeur de balise du flux d'image de l'échantillon d'origine change et que la nouvelle valeur de l'image diffère de l'image actuelle spécifiée dans le conteneur helloworld de l'objet DeploymentConfig, un nouveau contrôleur de réplication est créé à l'aide de la nouvelle image pour le conteneur helloworld.



NOTE

Lorsqu'un déclencheur de changement d'image est défini sur un objet DeploymentConfig (avec un déclencheur de modification de configuration et automatique=false, ou avec automatic=true) et que la balise de flux d'image pointée par le déclencheur de changement d'image n'existe pas encore, le processus de déploiement initial démarre automatiquement dès qu'une image est importée ou poussée par une construction vers la balise de flux d'image.

6.3.1.7.1. Définir les déclencheurs de déploiement

Procédure

1. Il est possible de définir les déclencheurs de déploiement d'un objet DeploymentConfig à l'aide de la commande déclencheurs oc. À titre d'exemple, pour définir un déclencheur de changement d'image, utilisez la commande suivante:

```
$ oc set triggers dc/<dc_name> \
    --from-image=<project>/<image>:<tag> -c <container_name>
```

6.3.1.8. Définition des ressources de déploiement

Le déploiement est complété par un pod qui consomme des ressources (mémoire, CPU et stockage éphémère) sur un nœud. Les pods consomment par défaut des ressources de nœuds non liés. Cependant, si un projet spécifie les limites de conteneurs par défaut, les pods consomment des ressources jusqu'à ces limites.

NOTE

La limite de mémoire minimale pour un déploiement est de 12 Mo. En cas de démarrage d'un conteneur en raison d'un événement de Pod de mémoire impossible, la limite de mémoire est trop faible. Augmentez ou supprimez la limite de mémoire. La suppression de la limite permet aux gousses de consommer des ressources de nœuds non limitées.

Il est également possible de limiter l'utilisation des ressources en spécifiant les limites de ressources dans le cadre de la stratégie de déploiement. Les ressources de déploiement peuvent être utilisées avec les stratégies de déploiement, de recréation, de roulement ou de déploiement personnalisées.

Procédure

1. Dans l'exemple suivant, chacune des ressources, cpu, mémoire et stockage éphémère est facultative:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hello-openshift
# ...
spec:
# ...
  type: "Recreate"
  resources:
    limits:
```

```
      cpu: "100m" (1)
      memory: "256Mi" (2)
      ephemeral-storage: "1Gi" (3)
```

(1) CPU est en unités CPU: 100m représente 0,1 unité CPU (100 * 1e-3).

(2) la mémoire est en octets: 256Mi représente 268435456 octets (256 * 2 ^ 20).

(3) le stockage éphémère est en octets: 1Gi représente 1073741824 octets (2 ^ 30).

Cependant, si un quota a été défini pour votre projet, l'un des deux éléments suivants est requis:

- D'une section de ressources assortie d'une demande explicite:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hello-openshift
# ...
spec:
# ...
  type: "Recreate"
  resources:
    requests: (1)
      cpu: "100m"
      memory: "256Mi"
      ephemeral-storage: "1Gi"
```

(1) L'objet demande contient la liste des ressources qui correspondent à la liste des ressources dans le quota.

- Limite définie dans votre projet, où les valeurs par défaut de l'objet LimitRange s'appliquent aux pods créés au cours du processus de déploiement.

Afin de définir les ressources de déploiement, choisissez l'une des options ci-dessus. Autrement, déployer la création de pod échoue, invoquant un défaut de satisfaire les quotas.

6.3.1.9. Evolution manuelle

En plus des retours en arrière, vous pouvez exercer un contrôle à grains fins sur le nombre de répliques en les évoluant manuellement.

NOTE

Les pods peuvent également être mis à l'échelle automatique à l'aide de la commande oc autoscale.

Procédure

1. Afin de mettre à l'échelle manuellement un objet DeploymentConfig, utilisez la commande oc scale. À titre d'exemple, la commande suivante définit les répliques de l'objet frontend DeploymentConfig à 3.

```
$ oc scale dc frontend --replicas=3
```

Le nombre de répliques finit par se propager à l'état souhaité et actuel du déploiement configuré par le frontend objet DeploymentConfig.

6.3.1.10. Accès aux référentiels privés à partir des objets DeploymentConfig

Ajoutez un secret à votre objet DeploymentConfig afin qu'il puisse accéder aux images d'un référentiel privé. Cette procédure montre la méthode Red Hat OpenShift Service sur la console web AWS.

Procédure

1. Créez un nouveau projet.
2. Accédez aux charges de travail → Secrets.
3. Créez un secret qui contient des informations d'identification pour accéder à un référentiel privé d'images.
4. Accédez à Workloads → DeploymentConfigs.
5. Créer un objet DeploymentConfig.
6. Dans la page de l'éditeur d'objet DeploymentConfig, définissez le secret Pull et enregistrez vos modifications.

6.3.1.11. Exécuter un pod avec un compte de service différent

Il est possible d'exécuter un pod avec un compte de service autre que le compte par défaut.

Procédure

1. Éditer l'objet DeploymentConfig:

```
$ oc edit dc/<deployment_config>
```

2. Ajoutez les paramètres serviceAccount et serviceAccountName au champ spec, et spécifiez le compte de service que vous souhaitez utiliser:

```
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
  name: example-dc
# ...
spec:
# ...
  securityContext: {}
  serviceAccount: <service_account>
  serviceAccountName: <service_account>
```

6.4. EN UTILISANT DES STRATÉGIES DE DÉPLOIEMENT

Les stratégies de déploiement sont utilisées pour modifier ou mettre à niveau les applications sans temps d'arrêt afin que les utilisateurs remarquent à peine un changement.

Étant donné que les utilisateurs accèdent généralement aux applications via une route gérée par un routeur, les stratégies de déploiement peuvent se concentrer sur les fonctionnalités d'objet DeploymentConfig ou les fonctionnalités de routage. Les stratégies qui se concentrent sur l'objet DeploymentConfig ont un impact sur tous les itinéraires qui utilisent l'application. Les stratégies qui utilisent le routeur comprennent des itinéraires individuels ciblés.

La plupart des stratégies de déploiement sont prises en charge par l'objet DeploymentConfig, et certaines stratégies supplémentaires sont prises en charge par le biais de fonctionnalités de routeur.

6.4.1. Choisir une stratégie de déploiement

Considérez ce qui suit lors du choix d'une stratégie de déploiement:

- Les connexions de longue durée doivent être gérées gracieusement.
- Les conversions de bases de données peuvent être complexes et doivent être faites et retournées avec l'application.
- Lorsque l'application est un hybride de microservices et de composants traditionnels, des temps d'arrêt pourraient être nécessaires pour terminer la transition.
- Il faut avoir l'infrastructure nécessaire pour le faire.
- Lorsque vous disposez d'un environnement de test non isolé, vous pouvez casser les versions nouvelles et anciennes.

La stratégie de déploiement utilise des vérifications de préparation pour déterminer si un nouveau pod est prêt à être utilisé. En cas d'échec d'une vérification de préparation, l'objet DeploymentConfig se répète pour exécuter le pod jusqu'à ce qu'il soit éteint. Le timeout par défaut est 10m, une valeur définie dans TimeoutSeconds dans dc.spec.strarategy.*params.

6.4.2. La stratégie de roulement

Le déploiement mobile remplace lentement les instances de la version précédente d'une application par des instances de la nouvelle version de l'application. La stratégie de roulement est la stratégie de déploiement par défaut utilisée si aucune stratégie n'est spécifiée sur un objet DeploymentConfig.

En général, un déploiement roulant attend que les nouveaux pods deviennent prêts via un contrôle de préparation avant de réduire les anciens composants. En cas de problème important, le déploiement roulant peut être avorté.

Lors de l'utilisation d'un déploiement roulant:

- Lorsque vous voulez ne pas prendre de temps d'arrêt lors d'une mise à jour de l'application.
- Lorsque votre application prend en charge l'ancien code et le nouveau code en même temps.

Le déploiement continu signifie que vous avez des versions anciennes et nouvelles de votre code en même temps. Cela nécessite généralement que votre application gère la compatibilité N-1.

Définition d'une stratégie de roulement

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
```

```
# ...
spec:
# ...
  strategy:
    type: Rolling
    rollingParams:
      updatePeriodSeconds: 1 (1)
      intervalSeconds: 1 (2)
      timeoutSeconds: 120 (3)
      maxSurge: "20%" (4)
      maxUnavailable: "10%" (5)
      pre: {} (6)
     post: {}
```

(1) Le temps d'attendre entre les mises à jour individuelles des pod. En cas d'absence de précision, cette valeur par défaut à 1.

(2) Le temps d'attendre entre le sondage de l'état du déploiement après la mise à jour. En cas d'absence de précision, cette valeur par défaut à 1.

(3) Le temps d'attendre un événement de mise à l'échelle avant d'abandonner. Facultatif; la valeur par défaut est 600. Ici, abandonner signifie retourner automatiquement au déploiement complet précédent.

(4) le maxSurge est facultatif et par défaut à 25% s'il n'est pas spécifié. Consultez les informations ci-dessous la procédure suivante.

(5) le maxUnavailable est optionnel et par défaut à 25% s'il n'est pas spécifié. Consultez les informations ci-dessous la procédure suivante.

(6) les pré et post sont tous deux des crochets du cycle de vie.

La stratégie de roulement:

1. Exécute n'importe quel crochet de cycle de vie.
2. Augmente le nouveau contrôleur de réplication en fonction du nombre de surtensions.
3. Réduit l'ancien contrôleur de réplication en fonction du nombre maximum indisponible.
4. Il répète cette mise à l'échelle jusqu'à ce que le nouveau contrôleur de réplication ait atteint le nombre de répliques souhaité et que l'ancien contrôleur de réplication ait été mis à l'échelle à zéro.
5. Exécute n'importe quel crochet après le cycle de vie.

IMPORTANT

Lors de la mise à l'échelle, la stratégie de roulement attend que les pods deviennent prêts afin qu'il puisse décider si une autre mise à l'échelle affecterait la disponibilité. En cas de mise à l'échelle des pods ne deviennent jamais prêts, le processus de déploiement finira par s'achever et entraînera une défaillance du déploiement.

Le paramètre maxUnavailable est le nombre maximum de pods qui peuvent être indisponibles pendant

la mise à jour. Le paramètre maxSurge est le nombre maximum de gousses qui peuvent être programmées au-dessus du nombre original de gousses. Les deux paramètres peuvent être définis sur un pourcentage (par exemple, 10 %) ou une valeur absolue (par exemple, 2). La valeur par défaut pour les deux est de 25%.

Ces paramètres permettent de régler le déploiement pour la disponibilité et la vitesse. À titre d'exemple:

- le maxUnavailable*=0 et maxSurge*=20% assurent le maintien de la pleine capacité pendant la mise à jour et la mise à l'échelle rapide.
- * maxUnavailable*=10% et maxSurge*=0 effectue une mise à jour sans capacité supplémentaire (une mise à jour en place).
- le maxUnavailable*=10% et maxSurge*=10% augmentent et diminuent rapidement avec un certain potentiel de perte de capacité.

Généralement, si vous voulez des déploiements rapides, utilisez maxSurge. Dans le cas où vous devez prendre en compte le quota de ressources et accepter l'indisponibilité partielle, utilisez maxUnavailable.

AVERTISSEMENT

Le paramètre par défaut pour maxUnavailable est 1 pour tous les pools de configuration de la machine dans Red Hat OpenShift Service sur AWS. Il est recommandé de ne pas modifier cette valeur et de mettre à jour un nœud de plan de contrôle à la fois. Il ne faut pas changer cette valeur à 3 pour le pool de plan de contrôle.

6.4.2.1. Déploiements des Canaries

Les déploiements roulants dans Red Hat OpenShift Service sur AWS sont des déploiements canaris ; une nouvelle version (le canari) est testée avant que toutes les anciennes instances ne soient remplacées. Lorsque la vérification de préparation ne réussit jamais, l'instance canari est supprimée et l'objet DeploymentConfig sera automatiquement redéployé.

La vérification de préparation fait partie du code d'application et peut être aussi sophistiquée que nécessaire pour s'assurer que la nouvelle instance est prête à être utilisée. Lorsque vous devez implémenter des vérifications plus complexes de l'application (comme l'envoi de véritables charges de travail utilisateur à la nouvelle instance), envisagez de mettre en œuvre un déploiement personnalisé ou d'utiliser une stratégie de déploiement bleu-vert.

6.4.2.2. Création d'un déploiement mobile

Les déploiements roulants sont le type par défaut dans Red Hat OpenShift Service sur AWS. Il est possible de créer un déploiement mobile à l'aide du CLI.

Procédure

1. Créer une application basée sur l'exemple d'images de déploiement trouvées dans Quay.io:

```
$ oc new-app quay.io/openshifttest/deployment-example:latest
```

NOTE

Cette image n'expose aucun port. Lorsque vous souhaitez exposer vos applications sur un service externe LoadBalancer ou activer l'accès à l'application via Internet public, créez un service en utilisant la commande oc expose dc/déploiement-exemple --port=<port> après avoir terminé cette procédure.

2. Lorsque vous avez installé le routeur, rendez l'application disponible via un itinéraire ou utilisez l'IP du service directement.

```
$ oc expose svc/deployment-example
```

3. Accédez à l'application à l'exemple de déploiement.<project>.<router_domain> pour vérifier que vous voyez l'image v1.

4. Faites évoluer l'objet DeploymentConfig jusqu'à trois répliques:

```
$ oc scale dc/deployment-example --replicas=3
```

5. Déclenchez automatiquement un nouveau déploiement en étiquetant une nouvelle version de l'exemple comme la dernière balise:

```
$ oc tag deployment-example:v2 deployment-example:latest
```

6. Dans votre navigateur, actualisez la page jusqu'à ce que vous voyez l'image v2.

7. Lors de l'utilisation du CLI, la commande suivante indique combien de pods sont sur la version 1 et combien sont sur la version 2. Dans la console web, les pods sont progressivement ajoutés à v2 et supprimés de v1:

```
$ oc describe dc deployment-example
```

Au cours du processus de déploiement, le nouveau contrôleur de réplication est progressivement mis à l'échelle. Après que les nouveaux pods soient marqués comme prêts (en passant leur vérification de préparation), le processus de déploiement se poursuit.

Dans le cas où les pods ne deviennent pas prêts, le processus s'absente et le déploiement revient à sa version précédente.

6.4.2.3. Éditer un déploiement en utilisant la perspective Développeur

En utilisant la perspective Développeur, vous pouvez modifier la stratégie de déploiement, les paramètres d'image, les variables d'environnement et les options avancées pour votre déploiement.

Conditions préalables

- Dans la perspective Développeur de la console web.
- C'est vous qui avez créé une application.

Procédure

1. Accédez à la vue Topology.

2. Cliquez sur votre application pour voir le panneau Détails.

3. Dans le menu déroulant Actions, sélectionnez Modifier le déploiement pour afficher la page Modifier le déploiement.

4. Il est possible d'éditer les options avancées suivantes pour votre déploiement:

    a. Facultatif: Vous pouvez mettre fin aux déploiements en cliquant sur les déploiements de Pause, puis en sélectionnant les déploiements de Pause pour cette case à cocher de déploiement.
    En mettant en pause les déploiements, vous pouvez apporter des modifications à votre application sans déclencher un déploiement. À tout moment, vous pouvez reprendre les déploiements.

    b. Facultatif: Cliquez sur Évoluer pour modifier le nombre d'instances de votre image en modifiant le nombre de répliques.

5. Cliquez sur **Save**.

6.4.2.4. Démarrage d'un déploiement mobile en utilisant la perspective Développeur

Il est possible de mettre à niveau une application en démarrant un déploiement mobile.

Conditions préalables

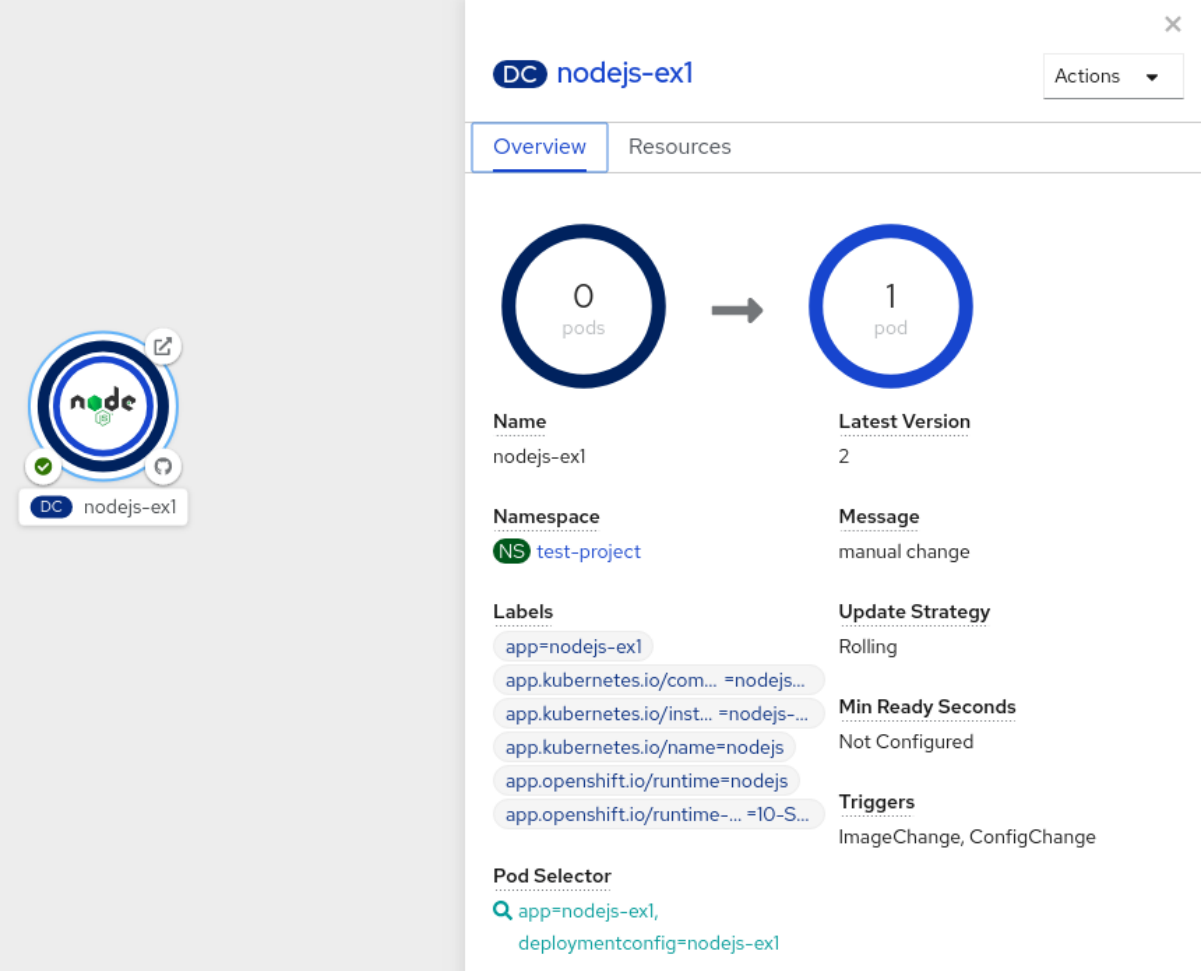
- Dans la perspective Développeur de la console web.
- C'est vous qui avez créé une application.

Procédure

1. Dans la vue Topologie, cliquez sur le nœud de l'application pour voir l'onglet Aperçu dans le panneau latéral. Il est à noter que la stratégie de mise à jour est définie sur la stratégie de roulement par défaut.

2. Dans le menu déroulant Actions, sélectionnez Démarrer le déploiement pour lancer une mise à jour continue. Le déploiement roulant fait tourner la nouvelle version de l'application, puis met fin à l'ancienne.

Figure 6.1. La mise à jour continue



Ressources supplémentaires

- Création et déploiement d'applications sur Red Hat OpenShift Service sur AWS en utilisant la perspective Développeur
- Afficher les applications de votre projet, vérifier leur statut de déploiement et interagir avec elles dans la vue Topology

6.4.3. Créer une stratégie

La stratégie de recréation a un comportement de déploiement de base et prend en charge les crochets du cycle de vie pour injecter du code dans le processus de déploiement.

Exemple recréer la définition de stratégie

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hello-openshift
# ...
spec:
# ...
  strategy:
    type: Recreate
    recreateParams: 1
```

```
    pre: {} 2
    mid: {}
    post: {}
```

1 les recréerParams sont facultatifs.

2 avant, le milieu et le post sont des crochets du cycle de vie.

La stratégie de recréer:

1. Exécute n'importe quel crochet de cycle de vie.
2. Réduit le déploiement précédent à zéro.
3. Exécute n'importe quel crochet de cycle de vie moyen.
4. Augmente le nouveau déploiement.
5. Exécute n'importe quel crochet après le cycle de vie.

IMPORTANT

Lors de la mise à l'échelle, si le nombre de répliques du déploiement est supérieur à un, la première réplique du déploiement sera validée pour la préparation avant d'augmenter complètement le déploiement. En cas d'échec de la validation de la première réplique, le déploiement sera considéré comme un échec.

Lors de l'utilisation d'un déploiement recréé:

- Lorsque vous devez effectuer des migrations ou d'autres transformations de données avant le début de votre nouveau code.
- Lorsque vous ne supportez pas d'avoir des versions nouvelles et anciennes de votre code d'application en même temps.
- Lorsque vous souhaitez utiliser un volume RWO, qui n'est pas pris en charge étant partagé entre plusieurs répliques.

Le déploiement recrée entraîne des temps d'arrêt parce que, pendant une courte période, aucune instance de votre application n'est en cours d'exécution. Cependant, votre ancien code et votre nouveau code ne s'exécutent pas en même temps.

6.4.3.1. Éditer un déploiement en utilisant la perspective Développeur

En utilisant la perspective Développeur, vous pouvez modifier la stratégie de déploiement, les paramètres d'image, les variables d'environnement et les options avancées pour votre déploiement.

Conditions préalables

- Dans la perspective Développeur de la console web.
- C'est vous qui avez créé une application.

Procédure

1. Accédez à la vue Topology.
2. Cliquez sur votre application pour voir le panneau Détails.
3. Dans le menu déroulant Actions, sélectionnez Modifier le déploiement pour afficher la page Modifier le déploiement.
4. Il est possible d'éditer les options avancées suivantes pour votre déploiement:
    a. Facultatif: Vous pouvez mettre fin aux déploiements en cliquant sur les déploiements de Pause, puis en sélectionnant les déploiements de Pause pour cette case à cocher de déploiement.
    En mettant en pause les déploiements, vous pouvez apporter des modifications à votre application sans déclencher un déploiement. À tout moment, vous pouvez reprendre les déploiements.
    b. Facultatif: Cliquez sur Évoluer pour modifier le nombre d'instances de votre image en modifiant le nombre de répliques.
5. Cliquez sur **Save**.

6.4.3.2. Démarrage d'un déploiement recréé en utilisant la perspective Développeur

La stratégie de déploiement peut passer de la mise à jour par défaut à une mise à jour recréée en utilisant la perspective Développeur dans la console Web.

Conditions préalables

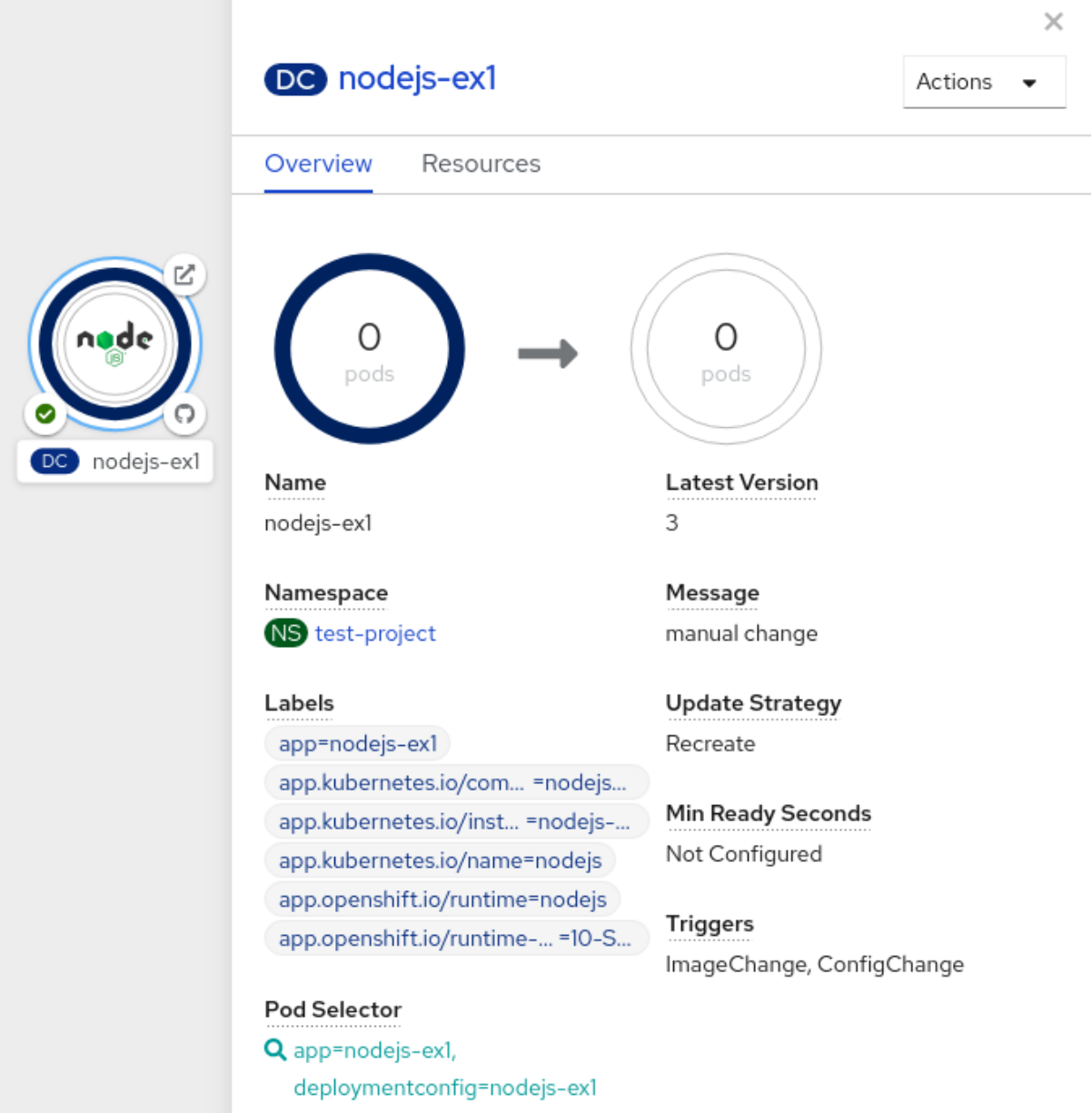
- Assurez-vous que vous êtes dans la perspective Développeur de la console Web.
- Assurez-vous d'avoir créé une application à l'aide de la vue Ajouter et de la voir déployée dans la vue Topology.

Procédure

De passer à une stratégie de recréer la mise à jour et de mettre à niveau une application:

1. Cliquez sur votre application pour voir le panneau Détails.
2. Dans le menu déroulant Actions, sélectionnez Edit Deployment Config pour voir les détails de configuration de déploiement de l'application.
3. Dans l'éditeur YAML, modifiez spec.strarategy.type pour Recréer et cliquez sur Enregistrer.
4. Dans la vue Topologie, sélectionnez le nœud pour voir l'onglet Aperçu dans le panneau latéral. La stratégie de mise à jour est maintenant définie sur Recreate.
5. Dans le menu déroulant Actions, sélectionnez Démarrer le déploiement pour lancer une mise à jour à l'aide de la stratégie de recréation. La stratégie de recréation termine d'abord les pods pour l'ancienne version de l'application, puis tourne les pods pour la nouvelle version.

Figure 6.2. Créer une mise à jour



Ressources supplémentaires

- Création et déploiement d'applications sur Red Hat OpenShift Service sur AWS en utilisant la perspective Développeur
- Afficher les applications de votre projet, vérifier leur statut de déploiement et interagir avec elles dans la vue Topology

6.4.4. La stratégie personnalisée

La stratégie personnalisée vous permet de fournir votre propre comportement de déploiement.

Exemple de définition de stratégie personnalisée

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
```

```
spec:
# ...
  strategy:
    type: Custom
    customParams:
      image: organization/strategy
      command: [ "command", "arg1" ]
      environment:
        - name: ENV_1
          value: VALUE_1
```

Dans l'exemple ci-dessus, l'image du conteneur d'organisation/stratégie fournit le comportement de déploiement. Le tableau de commande optionnel remplace toute directive CMD spécifiée dans le Dockerfile de l'image. Les variables d'environnement optionnelles fournies sont ajoutées à l'environnement d'exécution du processus de stratégie.

De plus, Red Hat OpenShift Service sur AWS fournit les variables d'environnement suivantes au processus de déploiement:

La variable d'environnement	Description
AJOUTER AU PANIER OPENSHIFT_DEPLOYMENT_NAME	Le nom du nouveau déploiement, un contrôleur de réplication.
DESCRIPTION DU PRODUIT OPENSHIFT_DEPLOYMENT_NAMESPACE	L'espace nom du nouveau déploiement.

Le nombre de répliques du nouveau déploiement sera initialement nul. La responsabilité de la stratégie est de rendre le nouveau déploiement actif en utilisant la logique qui répond le mieux aux besoins de l'utilisateur.

Alternativement, utilisez l'objet CustomParams pour injecter la logique de déploiement personnalisée dans les stratégies de déploiement existantes. Fournissez une logique de script shell personnalisé et appelez le binaire openshift-déploy. Les utilisateurs n'ont pas à fournir leur image de conteneur de déploiement personnalisé; dans ce cas, le service par défaut Red Hat OpenShift sur l'image de déploiement AWS est utilisé à la place:

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: example-dc
# ...
spec:
# ...
  strategy:
    type: Rolling
    customParams:
      command:
      - /bin/sh
      - -c
      - |
```

```
    set -e
    openshift-deploy --until=50%
    echo Halfway there
    openshift-deploy
    echo Complete
```

Il en résulte un déploiement suivant:

```
Started deployment #2
--> Scaling up custom-deployment-2 from 0 to 2, scaling down custom-deployment-1 from 2 to 0
(keep 2 pods available, don't exceed 3 pods)
    Scaling custom-deployment-2 up to 1
--> Reached 50% (currently 50%)
Halfway there
--> Scaling up custom-deployment-2 from 1 to 2, scaling down custom-deployment-1 from 2 to 0
(keep 2 pods available, don't exceed 3 pods)
    Scaling custom-deployment-1 down to 1
    Scaling custom-deployment-2 up to 2
    Scaling custom-deployment-1 down to 0
--> Success
Complete
```

Lorsque le processus de stratégie de déploiement personnalisé nécessite l'accès au service Red Hat OpenShift sur AWS API ou à l'API Kubernetes, le conteneur qui exécute la stratégie peut utiliser le jeton de compte de service disponible à l'intérieur du conteneur pour l'authentification.

6.4.4.1. Éditer un déploiement en utilisant la perspective Développeur

En utilisant la perspective Développeur, vous pouvez modifier la stratégie de déploiement, les paramètres d'image, les variables d'environnement et les options avancées pour votre déploiement.

Conditions préalables

- Dans la perspective Développeur de la console web.
- C'est vous qui avez créé une application.

Procédure

1. Accédez à la vue Topology.
2. Cliquez sur votre application pour voir le panneau Détails.
3. Dans le menu déroulant Actions, sélectionnez Modifier le déploiement pour afficher la page Modifier le déploiement.
4. Il est possible d'éditer les options avancées suivantes pour votre déploiement:
    a. Facultatif: Vous pouvez mettre fin aux déploiements en cliquant sur les déploiements de Pause, puis en sélectionnant les déploiements de Pause pour cette case à cocher de déploiement.
    En mettant en pause les déploiements, vous pouvez apporter des modifications à votre application sans déclencher un déploiement. À tout moment, vous pouvez reprendre les déploiements.

b. Facultatif: Cliquez sur Évoluer pour modifier le nombre d'instances de votre image en modifiant le nombre de répliques.

5. Cliquez sur **Save**.

6.4.5. Crochets de cycle de vie

Les stratégies de roulement et de recréation prennent en charge les crochets du cycle de vie, ou les crochets de déploiement, qui permettent d'injecter le comportement dans le processus de déploiement à des points prédéfinis dans la stratégie:

Exemple de crochet pré cycle de vie

```
pre:
  failurePolicy: Abort
  execNewPod: {} 1
```

1 execNewPod est un crochet de cycle de vie à base de pod.

Chaque crochet a une politique d'échec, qui définit l'action que la stratégie doit prendre lorsqu'une défaillance de crochet est rencontrée:

Avort	Le processus de déploiement sera considéré comme un échec si le crochet échoue.
Essayez de réessayer	L'exécution du crochet doit être récupérée jusqu'à ce qu'elle réussisse.
Ignorer	Les pannes de crochet doivent être ignorées et le déploiement doit se poursuivre.

Les crochets ont un champ spécifique qui décrit comment exécuter le crochet. Actuellement, les crochets à base de pod sont le seul type de crochet pris en charge, spécifié par le champ execNewPod.

Crochet de cycle de vie à base de pod

Les crochets de cycle de vie à base de pod exécutent le code de crochet dans une nouvelle pod dérivée du modèle dans un objet DeploymentConfig.

L'exemple de déploiement simplifié suivant utilise la stratégie de roulement. Les déclencheurs et d'autres détails mineurs sont omis pour la brièveté:

```
kind: DeploymentConfig
apiVersion: apps.openshift.io/v1
metadata:
  name: frontend
spec:
  template:
    metadata:
      labels:
        name: frontend
    spec:
      containers:
        - name: helloworld
          image: openshift/origin-ruby-sample
```

```
replicas: 5
selector:
  name: frontend
strategy:
  type: Rolling
  rollingParams:
    pre:
      failurePolicy: Abort
      execNewPod:
        containerName: helloworld (1)
        command: [ "/usr/bin/command", "arg1", "arg2" ] (2)
        env: (3)
          - name: CUSTOM_VAR1
            value: custom_value1
        volumes:
          - data (4)
```

1. Le nom helloworld fait référence à spec.template.spec.containers[0].name.
2. Cette commande remplace toute ENTRYPOINT définie par l'image openshift/origin-ruby-sample.
3. ENV est un ensemble optionnel de variables d'environnement pour le conteneur de crochet.
4. les volumes sont un ensemble facultatif de références de volume pour le récipient à crochets.

Dans cet exemple, le crochet pré sera exécuté dans un nouveau pod en utilisant l'image openshift/origin-ruby-sample du conteneur helloworld. La gousse de crochet a les propriétés suivantes:

- La commande crochet est /usr/bin/command arg1 arg2.
- Le conteneur de crochet a la variable d'environnement CUSTOM_VAR1=custom_value1.
- La politique d'échec du crochet est Abort, ce qui signifie que le processus de déploiement échoue si le crochet échoue.
- La gousse de crochet hérite du volume de données de la pod objet DeploymentConfig.

6.4.5.1. Réglage des crochets du cycle de vie

Il est possible de définir des crochets de cycle de vie, ou des crochets de déploiement, pour un déploiement à l'aide du CLI.

Procédure

1. La commande oc set déploiement-hook permet de définir le type de crochet que vous souhaitez: --pre, --mid, ou --post. À titre d'exemple, pour définir un crochet de pré-déploiement:

```
$ oc set deployment-hook dc/frontend \
    --pre -c helloworld -e CUSTOM_VAR1=custom_value1 \
    --volumes data --failure-policy=abort -- /usr/bin/command arg1 arg2
```

6.5. EN UTILISANT DES STRATÉGIES DE DÉPLOIEMENT BASÉES SUR LA ROUTE

Les stratégies de déploiement permettent à l'application d'évoluer. Certaines stratégies utilisent des objets de déploiement pour apporter des modifications qui sont vues par les utilisateurs de tous les itinéraires qui se résolvent à l'application. D'autres stratégies avancées, telles que celles décrites dans cette section, utilisent des fonctions de routeur en conjonction avec des objets de déploiement pour avoir un impact sur des itinéraires spécifiques.

La stratégie la plus courante basée sur la route est d'utiliser un déploiement bleu-vert. La nouvelle version (la version verte) est présentée pour les tests et l'évaluation, tandis que les utilisateurs utilisent toujours la version stable (la version bleue). Lorsqu'ils sont prêts, les utilisateurs sont passés à la version verte. En cas de problème, vous pouvez revenir à la version bleue.

Alternativement, vous pouvez utiliser une stratégie de versions A/B dans laquelle les deux versions sont actives en même temps. Avec cette stratégie, certains utilisateurs peuvent utiliser la version A et d'autres utilisateurs peuvent utiliser la version B. Cette stratégie permet d'expérimenter des changements d'interface utilisateur ou d'autres fonctionnalités afin d'obtenir des commentaires des utilisateurs. Il peut également être utilisé pour vérifier le bon fonctionnement dans un contexte de production où les problèmes affectent un nombre limité d'utilisateurs.

Le déploiement canari teste la nouvelle version, mais lorsqu'un problème est détecté, il revient rapidement à la version précédente. Cela peut être fait avec les deux stratégies ci-dessus.

Les stratégies de déploiement basées sur la route n'évoluent pas le nombre de pods dans les services. Afin de maintenir les caractéristiques de performance souhaitées, les configurations de déploiement pourraient devoir être mises à l'échelle.

6.5.1. Éclats de proxy et fractionnement du trafic

Dans les environnements de production, vous pouvez contrôler avec précision la distribution du trafic qui atterrit sur un fragment particulier. Lorsque vous traitez d'un grand nombre d'instances, vous pouvez utiliser l'échelle relative des fragments individuels pour implémenter le trafic basé sur le pourcentage. Cela se combine bien avec un shard proxy, qui transmet ou divise le trafic qu'il reçoit à un service ou une application séparé fonctionnant ailleurs.

Dans la configuration la plus simple, le proxy transmet les requêtes inchangées. Dans les configurations plus complexes, vous pouvez dupliquer les requêtes entrantes et envoyer à la fois à un cluster séparé ainsi qu'à une instance locale de l'application, et comparer le résultat. D'autres modèles incluent la conservation des caches d'une installation DR au chaud, ou l'échantillonnage du trafic entrant à des fins d'analyse.

Chaque proxy TCP (ou UDP) pourrait être exécuté sous le shard désiré. La commande oc scale permet de modifier le nombre relatif d'instances qui servent des requêtes dans le shard proxy. Afin de gérer le trafic plus complexe, envisagez de personnaliser le service OpenShift Red Hat sur le routeur AWS avec des capacités d'équilibrage proportionnelles.

6.5.2. Compatibilité N-1

Les applications qui ont un nouveau code et un ancien code en même temps doivent veiller à ce que les données écrites par le nouveau code puissent être lues et traitées (ou gracieusement ignorées) par l'ancienne version du code. Cela s'appelle parfois l'évolution du schéma et est un problème complexe.

Cela peut prendre plusieurs formes: données stockées sur disque, dans une base de données, dans un cache temporaire, ou qui fait partie de la session du navigateur d'un utilisateur. Bien que la plupart des applications Web puissent prendre en charge les déploiements roulants, il est important de tester et de

concevoir votre application pour la gérer.

Dans certaines applications, la période pendant laquelle l'ancien code et le nouveau code s'exécutent côte à côte est courte, de sorte que les bogues ou certaines transactions utilisateur échouées sont acceptables. Dans d'autres cas, le schéma d'échec peut entraîner la non-fonctionnalité de l'ensemble de l'application.

La validation de la compatibilité N-1 consiste à utiliser un déploiement A/B: exécuter l'ancien code et le nouveau code en même temps de manière contrôlée dans un environnement de test, et vérifier que le trafic qui s'écoule vers le nouveau déploiement ne provoque pas de défaillances dans l'ancien déploiement.

6.5.3. Résiliation gracieuse

Le service OpenShift Red Hat sur AWS et Kubernetes donne aux instances d'application le temps de s'arrêter avant de les retirer des rotations d'équilibrage de charge. Cependant, les applications doivent s'assurer qu'elles arrêtent proprement les connexions utilisateur avant leur sortie.

Lors de l'arrêt, Red Hat OpenShift Service sur AWS envoie un signal TERM aux processus dans le conteneur. Code de l'application, sur réception SIGTERM, arrêtez d'accepter de nouvelles connexions. Cela garantit que les balanceurs de charge acheminent le trafic vers d'autres instances actives. Le code de l'application attend ensuite jusqu'à ce que toutes les connexions ouvertes soient fermées, ou gracieusement mettre fin aux connexions individuelles à la prochaine occasion, avant de sortir.

Après l'expiration du délai de résiliation gracieux, un processus qui n'est pas sorti est envoyé le signal KILL, qui met immédiatement fin au processus. L'attribut terminaisonGracePeriodSeconds d'un modèle pod ou pod contrôle le délai de résiliation gracieux (par défaut 30 secondes) et peut être personnalisé par application si nécessaire.

6.5.4. Déploiements bleu-vert

Les déploiements bleu-vert impliquent l'exécution de deux versions d'une application en même temps et le déplacement du trafic de la version en production (la version bleue) à la version la plus récente (la version verte). Il est possible d'utiliser une stratégie mobile ou de changer de service dans un itinéraire.

Étant donné que de nombreuses applications dépendent de données persistantes, vous devez avoir une application qui prend en charge la compatibilité N-1, ce qui signifie qu'elle partage des données et implémente la migration en direct entre la base de données, le stockage ou le disque en créant deux copies de la couche de données.

Considérez les données utilisées pour tester la nouvelle version. Lorsqu'il s'agit des données de production, un bug dans la nouvelle version peut casser la version de production.

6.5.4.1. Configuration d'un déploiement bleu-vert

Les déploiements bleu-vert utilisent deux objets de déploiement. Les deux sont en cours d'exécution, et celui en production dépend du service que l'itinéraire spécifie, chaque objet de déploiement étant exposé à un service différent.

NOTE

Les itinéraires sont destinés au trafic Web (HTTP et HTTPS), de sorte que cette technique convient le mieux aux applications Web.

Il est possible de créer un nouvel itinéraire vers la nouvelle version et de le tester. Lorsque vous êtes prêt, modifiez le service dans la voie de production pour pointer vers le nouveau service et la nouvelle version (verte) est en direct.

Au besoin, vous pouvez revenir à l'ancienne version (bleue) en reconnectant le service à la version précédente.

Procédure

1. Créez deux composants d'application indépendants.
   a. Créez une copie de l'application d'exemple exécutant l'image v1 sous le service example-blue:

      ```
      $ oc new-app openshift/deployment-example:v1 --name=example-blue
      ```

   b. Créez une deuxième copie qui utilise l'image v2 sous le service example-green:

      ```
      $ oc new-app openshift/deployment-example:v2 --name=example-green
      ```

2. Créez un itinéraire qui pointe vers l'ancien service:

   ```
   $ oc expose svc/example-blue --name=bluegreen-example
   ```

3. Accédez à l'application à bluegreen-example-<project>.<router_domain> pour vérifier que vous voyez l'image v1.

4. Modifiez l'itinéraire et modifiez le nom du service en exemple-green:

   ```
   $ oc patch route/bluegreen-example -p '{"spec":{"to":{"name":"example-green"}}}'
   ```

5. Afin de vérifier que l'itinéraire a changé, actualisez le navigateur jusqu'à ce que vous voyez l'image v2.

6.5.5. Déploiements a/B

La stratégie de déploiement A/B vous permet d'essayer une nouvelle version de l'application de manière limitée dans l'environnement de production. Il est possible de spécifier que la version de production reçoit la plupart des requêtes de l'utilisateur alors qu'une fraction limitée des requêtes vont à la nouvelle version.

Étant donné que vous contrôlez la partie des requêtes à chaque version, à mesure que le test progresse, vous pouvez augmenter la fraction des requêtes vers la nouvelle version et finalement arrêter d'utiliser la version précédente. Au fur et à mesure que vous ajustez la charge de requête sur chaque version, le nombre de pods dans chaque service peut devoir être mis à l'échelle aussi bien pour fournir les performances attendues.

En plus de mettre à niveau le logiciel, vous pouvez utiliser cette fonctionnalité pour expérimenter des versions de l'interface utilisateur. Comme certains utilisateurs obtiennent l'ancienne version et d'autres la nouvelle, vous pouvez évaluer la réaction de l'utilisateur aux différentes versions pour éclairer les décisions de conception.

Afin d'être efficace, les anciennes et les nouvelles versions doivent être suffisamment similaires pour que les deux puissent fonctionner en même temps. Ceci est courant avec les versions de correction de

bogues et lorsque les nouvelles fonctionnalités n'interfèrent pas avec l'ancienne. Les versions nécessitent une compatibilité N-1 pour fonctionner correctement ensemble.

Le Red Hat OpenShift Service sur AWS prend en charge la compatibilité N-1 via la console Web ainsi que le CLI.

6.5.5.1. Équilibrage de charge pour les essais A/B

L'utilisateur met en place un itinéraire avec plusieurs services. Chaque service gère une version de l'application.

Chaque service se voit attribuer un poids et la partie des requêtes à chaque service est le service_weight divisé par le sum_of_weights. La pondération pour chaque service est répartie aux points de terminaison du service de sorte que la somme des pondérations du point de terminaison soit le poids du service.

L'itinéraire peut avoir jusqu'à quatre services. Le poids pour le service peut être compris entre 0 et 256. Lorsque le poids est 0, le service ne participe pas à l'équilibrage de charge, mais continue de desservir les connexions persistantes existantes. Lorsque le poids de service n'est pas 0, chaque point final a un poids minimum de 1. De ce fait, un service avec beaucoup de points finaux peut se retrouver avec un poids plus élevé que prévu. Dans ce cas, réduisez le nombre de gousses pour obtenir le poids attendu de l'équilibre de charge.

Procédure

La mise en place de l'environnement A/B:

1. Créez les deux applications et donnez-leur des noms différents. Chacun crée un objet de déploiement. Les applications sont des versions du même programme; l'une est généralement la version de production actuelle et l'autre la nouvelle version proposée.

   a. Créez la première application. L'exemple suivant crée une application appelée ab-example-a:

   ```
   $ oc new-app openshift/deployment-example --name=ab-example-a
   ```

   b. Créer la deuxième application:

   ```
   $ oc new-app openshift/deployment-example:v2 --name=ab-example-b
   ```

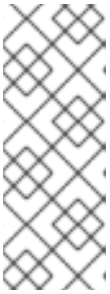
   Les deux applications sont déployées et des services sont créés.

2. Faites en sorte que l'application soit disponible à l'extérieur via un itinéraire. À ce stade, vous pouvez exposer non plus. Il peut être pratique d'exposer d'abord la version de production actuelle et de modifier plus tard l'itinéraire pour ajouter la nouvelle version.

   ```
   $ oc expose svc/ab-example-a
   ```

   Accédez à l'application ab-example-a.<project>.<router_domain> pour vérifier que vous voyez la version attendue.

3. Lorsque vous déployez l'itinéraire, le routeur équilibre le trafic en fonction des poids spécifiés pour les services. À ce stade, il y a un seul service avec le poids par défaut=1 de sorte que toutes les requêtes vont à elle. L'ajout de l'autre service en tant que backends alternatif et l'ajustement des poids donnent vie à la configuration A/B. Cela peut être fait par la commande oc set route-backends ou en modifiant l'itinéraire.



NOTE

Lorsque vous utilisez des backends alternatifs, utilisez également la stratégie d'équilibrage de la charge de rotation pour s'assurer que les demandes sont distribuées comme prévu aux services en fonction du poids. la rondrobine peut être définie pour un itinéraire à l'aide d'une annotation de route. Consultez la section Ressources supplémentaires pour plus d'informations sur les annotations d'itinéraire.

La définition d'oc set route-backend à 0 signifie que le service ne participe pas à l'équilibrage de charge, mais continue de desservir les connexions persistantes existantes.

NOTE

Les modifications apportées à l'itinéraire ne font que changer la portion du trafic vers les différents services. Il vous faudra peut-être mettre à l'échelle le déploiement pour ajuster le nombre de pods pour gérer les charges prévues.

Afin de modifier l'itinéraire, exécutez:

```
$ oc edit route <route_name>
```

Exemple de sortie

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-alternate-service
  annotations:
    haproxy.router.openshift.io/balance: roundrobin
# ...
spec:
  host: ab-example.my-project.my-domain
  to:
    kind: Service
    name: ab-example-a
    weight: 10
  alternateBackends:
  - kind: Service
    name: ab-example-b
    weight: 15
# ...
```

6.5.5.1.1. Gérer les poids d'un itinéraire existant à l'aide de la console Web

Procédure

1. Accédez à la page Networking → Routes.
2. Cliquez sur le menu Options à côté de l'itinéraire que vous souhaitez modifier et sélectionnez Modifier l'itinéraire.
3. Éditez le fichier YAML. Actualisez le poids pour être un entier entre 0 et 256 qui spécifie le poids

relatif de la cible par rapport aux autres objets de référence cibles. La valeur 0 supprime les requêtes dans ce back end. La valeur par défaut est 100. Exécutez oc expliquer routes.spec.alternateBackends pour plus d'informations sur les options.

4. Cliquez sur **Save**.

6.5.5.1.2. Gérer les poids d'une nouvelle route à l'aide de la console Web

1. Accédez à la page Networking → Routes.
2. Cliquez sur Créer un itinéraire.
3. Entrez le nom de l'itinéraire.
4. Choisissez le Service.
5. Cliquez sur Ajouter un service alternatif.
6. Entrez une valeur pour le poids et le poids de service alternatif. Entrez un nombre compris entre 0 et 255 qui représente un poids relatif par rapport à d'autres cibles. La valeur par défaut est 100.
7. Choisissez le port cible.
8. Cliquez sur **Create**.

6.5.5.1.3. Gérer les poids à l'aide du CLI

Procédure

1. Afin de gérer les services et les poids correspondants de charge équilibrés par l'itinéraire, utilisez la commande route-backends oc set:

```
$ oc set route-backends ROUTENAME \
    [--zero|--equal] [--adjust] SERVICE=WEIGHT[%] [...] [options]
```

A titre d'exemple, ab-exemple-a est le service principal avec poids=198 et ab-example-b comme premier service de rechange avec un poids = 2:

```
$ oc set route-backends ab-example ab-example-a=198 ab-example-b=2
```

Cela signifie que 99% du trafic est envoyé au service ab-exemple-a et 1% pour le service ab-example-b.

Cette commande ne met pas à l'échelle le déploiement. Il se peut que vous soyez obligé de le faire pour disposer de suffisamment de pods pour gérer le chargement de la demande.

2. Exécutez la commande sans drapeaux pour vérifier la configuration actuelle:

```
$ oc set route-backends ab-example
```

Exemple de sortie

```
NAME                KIND    TO          WEIGHT
routes/ab-example   Service ab-example-a 198 (99%)
routes/ab-example   Service ab-example-b 2   (1%)
```

3. Afin de remplacer les valeurs par défaut de l'algorithme d'équilibrage de charge, ajustez l'annotation sur l'itinéraire en réglant l'algorithme sur roundrobin. Dans le cas d'un itinéraire sur Red Hat OpenShift Service sur AWS, l'algorithme d'équilibrage de charge par défaut est défini sur des valeurs aléatoires ou source.
Afin de définir l'algorithme sur Roundrobin, exécutez la commande:

```
$ oc annotate routes/<route-name> haproxy.router.openshift.io/balance=roundrobin
```

La valeur par défaut est source pour les routes de transmission de la sécurité des couches de transport (TLS). Dans tous les autres itinéraires, la valeur par défaut est aléatoire.

4. Afin de modifier le poids d'un service individuel par rapport à lui-même ou au service principal, utilisez le drapeau --ajuster. La spécification d'un pourcentage ajuste le service par rapport à la primaire ou à la première alternative (si vous spécifiez la primaire). En cas d'autres backends, leurs poids sont maintenus proportionnels au changement.
L'exemple suivant modifie le poids des services ab-exemple-a et ab-exemple-b:

```
$ oc set route-backends ab-example --adjust ab-example-a=200 ab-example-b=10
```

Alternativement, modifier le poids d'un service en spécifiant un pourcentage:

```
$ oc set route-backends ab-example --adjust ab-example-b=5%
```

En spécifiant + avant la déclaration de pourcentage, vous pouvez ajuster une pondération par rapport au paramètre actuel. À titre d'exemple:

```
$ oc set route-backends ab-example --adjust ab-example-b=+15%
```

Le drapeau égal fixe le poids de tous les services à 100:

```
$ oc set route-backends ab-example --equal
```

Le drapeau --zero définit le poids de tous les services à 0. Ensuite, toutes les requêtes retournent avec une erreur 503.

NOTE

Les routeurs ne supportent pas tous les backends multiples ou pondérés.

6.5.5.1.4. 1 service, plusieurs objets de déploiement

Procédure

1. Créez une nouvelle application, en ajoutant une étiquette ab-example=true qui sera commune à tous les fragments:

```
$ oc new-app openshift/deployment-example --name=ab-example-a --as-deployment-
config=true --labels=ab-example=true --env=SUBTITLE\=shardA
```

```
$ oc delete svc/ab-example-a
```

L'application est déployée et un service est créé. C'est le premier shard.

2. Rendre l'application disponible via un itinéraire, ou utiliser le service IP directement:

```
$ oc expose deployment ab-example-a --name=ab-example --selector=ab-example\=true
```

```
$ oc expose service ab-example
```

3. Accédez à l'application ab-example-<project_name>.<router_domain> pour vérifier que vous voyez l'image v1.

4. Créez un second fragment basé sur la même image source et étiquette que le premier shard, mais avec une version différente et des variables d'environnement uniques:

```
$ oc new-app openshift/deployment-example:v2 \
    --name=ab-example-b --labels=ab-example=true \
    SUBTITLE="shard B" COLOR="red" --as-deployment-config=true
```

```
$ oc delete svc/ab-example-b
```

5. À ce stade, les deux ensembles de gousses sont desservis sous la route. Cependant, étant donné que les deux navigateurs (en laissant une connexion ouverte) et le routeur (par défaut, via un cookie) tentent de préserver votre connexion à un serveur back-end, vous pourriez ne pas voir les deux fragments vous être retournés.
Forcer votre navigateur à l'un ou l'autre shard:

   a. La commande oc scale permet de réduire les répliques d'ab-exemple-a à 0.

```
$ oc scale dc/ab-example-a --replicas=0
```

      Actualisez votre navigateur pour afficher v2 et shard B (en rouge).

   b. Échelle ab-exemple-a à 1 réplique et ab-exemple-b à 0:

```
$ oc scale dc/ab-example-a --replicas=1; oc scale dc/ab-example-b --replicas=0
```

      Actualisez votre navigateur pour afficher v1 et shard A (en bleu).

6. Lorsque vous déclenchez un déploiement sur l'un ou l'autre des fragments, seuls les pods de ce fragment sont affectés. Il est possible de déclencher un déploiement en modifiant la variable d'environnement SUBTITLE dans l'un ou l'autre objet de déploiement:

```
$ oc edit dc/ab-example-a
```

   a) ou

```
$ oc edit dc/ab-example-b
```

6.5.6. Ressources supplémentaires

- Annotations spécifiques à la route.

CHAPITRE 7. QUOTAS

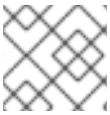
7.1. QUOTAS DE RESSOURCES PAR PROJET

Le quota de ressources, défini par un objet ResourceQuota, fournit des contraintes qui limitent la consommation globale de ressources par projet. Il peut limiter la quantité d'objets qui peuvent être créés dans un projet par type, ainsi que la quantité totale de ressources de calcul et de stockage qui pourraient être consommés par les ressources de ce projet.

Ce guide décrit comment fonctionnent les quotas de ressources, comment les administrateurs de clusters peuvent définir et gérer les quotas de ressources par projet, et comment les développeurs et les administrateurs de clusters peuvent les voir.

7.1.1. Les ressources gérées par les quotas

Ce qui suit décrit l'ensemble des ressources de calcul et des types d'objets qui peuvent être gérés par un quota.



NOTE

Le pod est dans un état terminal si Status.phase dans (Failed, Succeed) est vrai.

Tableau 7.1. Calcul des ressources gérées par quota

Le nom de la ressource	Description
CPU	La somme des requêtes CPU dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. CPU et request.cpu sont la même valeur et peuvent être utilisés de manière interchangeable.
la mémoire	La somme des requêtes de mémoire dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. la mémoire et les requêtes.memory sont la même valeur et peuvent être utilisées de manière interchangeable.
demandes.cpu	La somme des requêtes CPU dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. CPU et request.cpu sont la même valeur et peuvent être utilisés de manière interchangeable.
demandes.memory	La somme des requêtes de mémoire dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. la mémoire et les requêtes.memory sont la même valeur et peuvent être utilisées de manière interchangeable.
limites.cpu	La somme des limites CPU sur toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur.
Limites.memory	La somme des limites de mémoire à travers tous les pods dans un état non terminal ne peut pas dépasser cette valeur.

Tableau 7.2. Les ressources de stockage gérées par quota

Le nom de la ressource	Description
demandes.stockage	La somme des demandes de stockage sur toutes les revendications de volume persistant dans n'importe quel état ne peut pas dépasser cette valeur.
revendications persistantes du volume	Le nombre total de revendications de volume persistant qui peuvent exister dans le projet.
<storage-class-name>.storageclass.storage.k8s.io/requests.storage	La somme des demandes de stockage pour toutes les revendications de volume persistantes dans n'importe quel état ayant une classe de stockage correspondante ne peut pas dépasser cette valeur.
<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims	Le nombre total de revendications de volume persistant avec une classe de stockage correspondante qui peut exister dans le projet.
le stockage éphémère	La somme des demandes de stockage éphémères locales dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur. le stockage éphémère et les requêtes.ephemeral-stockage sont la même valeur et peuvent être utilisés de manière interchangeable.
demandes.ephemeral-stockage	La somme des demandes de stockage éphémères dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur. le stockage éphémère et les requêtes.ephemeral-stockage sont la même valeur et peuvent être utilisés de manière interchangeable.
limites.ephemeral-stockage	La somme des limites de stockage éphémères dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur.

Tableau 7.3. Comptages d'objets gérés par quota

Le nom de la ressource	Description
les gousses	Le nombre total de pods dans un état non terminal qui peut exister dans le projet.
contrôleurs de réplication	Le nombre total de RéplicationControllers qui peuvent exister dans le projet.
quotas de ressources	Le nombre total de quotas de ressources pouvant exister dans le projet.
les services	Le nombre total de services pouvant exister dans le projet.
balanceurs de charge Services.	Le nombre total de services de type LoadBalancer qui peuvent exister dans le projet.

Le nom de la ressource	Description
les services.nodeports	Le nombre total de services de type NodePort pouvant exister dans le projet.
les secrets	Le nombre total de secrets qui peuvent exister dans le projet.
ConfigMaps	Le nombre total d'objets ConfigMap pouvant exister dans le projet.
revendications persistantes du volume	Le nombre total de revendications de volume persistant qui peuvent exister dans le projet.
informations sur OpenShift.io/imagestreams	Le nombre total de flux d'images pouvant exister dans le projet.

7.1.2. Champ d'application des quotas

Chaque quota peut avoir un ensemble de portées associées. Le quota ne mesure l'utilisation d'une ressource que s'il correspond à l'intersection des périmètres énumérés.

L'ajout d'une portée à un quota limite l'ensemble des ressources auxquelles ce quota peut s'appliquer. La spécification d'une ressource en dehors de l'ensemble autorisé entraîne une erreur de validation.

Champ d'application	Description
BestEffort	Assortir des gousses qui ont le meilleur effort de qualité de service pour le cpu ou la mémoire.
À propos de NotBestEffort	Assortir des gousses qui n'ont pas le meilleur effort de qualité de service pour le cpu et la mémoire.

La portée de BestEffort limite un quota à la limitation des ressources suivantes:

- **les gousses**

La portée NotBestEffort limite un quota au suivi des ressources suivantes:

- **les gousses**
- **la mémoire**
- **demandes.memory**
- **Limites.memory**
- **CPU**
- **demandes.cpu**
- **limites.cpu**

7.1.3. Application des quotas

Après la création d'un quota de ressources pour un projet, le projet restreint la possibilité de créer de nouvelles ressources qui pourraient violer une contrainte de quota jusqu'à ce qu'il ait calculé des statistiques d'utilisation mises à jour.

Après la création d'un quota et la mise à jour des statistiques d'utilisation, le projet accepte la création de nouveaux contenus. Lorsque vous créez ou modifiez des ressources, votre utilisation de quota est incrémentée immédiatement à la demande de créer ou de modifier la ressource.

Lorsque vous supprimez une ressource, votre utilisation des quotas est décrémentée lors de la prochaine recalcul complet des statistiques des quotas pour le projet. Le temps configurable détermine combien de temps il faut pour réduire les statistiques d'utilisation des quotas à la valeur actuelle du système observé.

Lorsque les modifications de projet dépassent une limite d'utilisation de quota, le serveur refuse l'action, et un message d'erreur approprié est renvoyé à l'utilisateur expliquant la contrainte de quota violée, et quelles sont les statistiques d'utilisation actuellement observées dans le système.

7.1.4. Demandes par rapport aux limites

Lors de l'attribution des ressources de calcul, chaque conteneur peut spécifier une requête et une valeur limite chacune pour le stockage CPU, mémoire et éphémère. Les quotas peuvent restreindre l'une de ces valeurs.

Lorsque le quota a une valeur spécifiée pour request.cpu ou request.memory, il exige que chaque conteneur entrant fasse une demande explicite pour ces ressources. Lorsque le quota a une valeur spécifiée pour limits.cpu ou limits.memory, il exige que chaque conteneur entrant spécifie une limite explicite pour ces ressources.

7.1.5. Exemples de définitions des quotas de ressources

Core-object-counts.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: core-object-counts
spec:
  hard:
    configmaps: "10" 1
    persistentvolumeclaims: "4" 2
    replicationcontrollers: "20" 3
    secrets: "10" 4
    services: "10" 5
    services.loadbalancers: "2" 6
```

1. Le nombre total d'objets ConfigMap pouvant exister dans le projet.
2. Le nombre total de revendications en volume persistant (PVC) qui peuvent exister dans le projet.
3. Le nombre total de contrôleurs de réplication pouvant exister dans le projet.
4. Le nombre total de secrets qui peuvent exister dans le projet.

5 Le nombre total de services pouvant exister dans le projet.

6 Le nombre total de services de type LoadBalancer qui peuvent exister dans le projet.

ajouter au panier OpenShift-object-counts.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: openshift-object-counts
spec:
  hard:
    openshift.io/imagestreams: "10" 1
```

1 Le nombre total de flux d'images pouvant exister dans le projet.

calcul-resources.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "4" 1
    requests.cpu: "1" 2
    requests.memory: 1Gi 3
    limits.cpu: "2" 4
    limits.memory: 2Gi 5
```

1 Le nombre total de pods dans un état non terminal qui peut exister dans le projet.

2 Dans tous les pods dans un état non terminal, la somme des requêtes CPU ne peut pas dépasser 1 cœur.

3 Dans tous les pods dans un état non terminal, la somme des requêtes de mémoire ne peut pas dépasser 1Gi.

4 Dans toutes les gousses dans un état non terminal, la somme des limites CPU ne peut pas dépasser 2 cœurs.

5 Dans toutes les gousses dans un état non terminal, la somme des limites de mémoire ne peut pas dépasser 2Gi.

le meilleur effort.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: besteffort
spec:
```

```
  hard:
    pods: "1" 1
  scopes:
  - BestEffort 2
```

1 Le nombre total de pods dans un état non terminal avec la qualité de service BestEffort qui peut exister dans le projet.

2 Limite le quota aux seuls pods assortis qui ont une qualité de service BestEffort pour la mémoire ou le CPU.

calcul-ressources-long-running.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources-long-running
spec:
  hard:
    pods: "4" 1
    limits.cpu: "4" 2
    limits.memory: "2Gi" 3
  scopes:
  - NotTerminating 4
```

1 Le nombre total de gousses dans un état non terminal.

2 Dans toutes les gousses dans un état non terminal, la somme des limites CPU ne peut pas dépasser cette valeur.

3 Dans toutes les gousses dans un état non-terminal, la somme des limites de mémoire ne peut pas dépasser cette valeur.

4 Limite le quota à des pods correspondant uniquement lorsque spec.activeDeadlineSeconds est fixé à zéro. Les pods de construction relèvent de Non-Terminating à moins que la politique de redémarrage ne soit appliquée.

calcul-resources-time-bound.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources-time-bound
spec:
  hard:
    pods: "2" 1
    limits.cpu: "1" 2
    limits.memory: "1Gi" 3
  scopes:
  - Terminating 4
```

1. Le nombre total de pods dans un état de terminaison.
2. Dans tous les pods dans un état de terminaison, la somme des limites CPU ne peut pas dépasser cette valeur.
3. Dans tous les pods dans un état de terminaison, la somme des limites de mémoire ne peut pas dépasser cette valeur.
4. Limite le quota à des pods correspondant uniquement lorsque spec.activeDeadlineSeconds >=0. À titre d'exemple, ce quota coûte des pods de construction ou de déploiement, mais pas des pods en cours d'exécution comme un serveur Web ou une base de données.

conservation-consommation.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-consumption
spec:
  hard:
    persistentvolumeclaims: "10" (1)
    requests.storage: "50Gi" (2)
    gold.storageclass.storage.k8s.io/requests.storage: "10Gi" (3)
    silver.storageclass.storage.k8s.io/requests.storage: "20Gi" (4)
    silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5" (5)
    bronze.storageclass.storage.k8s.io/requests.storage: "0" (6)
    bronze.storageclass.storage.k8s.io/persistentvolumeclaims: "0" (7)
    requests.ephemeral-storage: 2Gi (8)
    limits.ephemeral-storage: 4Gi (9)
```

1. Le nombre total de réclamations en volume persistant dans un projet
2. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée ne peut pas dépasser cette valeur.
3. Dans toutes les revendications de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage aurifère ne peut pas dépasser cette valeur.
4. Dans toutes les revendications de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage d'argent ne peut pas dépasser cette valeur.
5. Dans toutes les revendications en volume persistant dans un projet, le nombre total de revendications dans la classe de stockage d'argent ne peut pas dépasser cette valeur.
6. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage de bronze ne peut pas dépasser cette valeur. Lorsque cela est réglé à 0, cela signifie que la classe de stockage en bronze ne peut pas demander de stockage.
7. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage de bronze ne peut pas dépasser cette valeur. Lorsque cela est réglé à 0, cela signifie que la classe de stockage de bronze ne peut pas créer de revendications.
8. Dans toutes les gousses dans un état non terminal, la somme des demandes de stockage éphémère ne peut pas dépasser 2Gi.

9 Dans toutes les gousses dans un état non terminal, la somme des limites de stockage éphémères ne peut pas dépasser 4Gi.

7.1.6. Créer un quota

Il est possible de créer un quota pour limiter l'utilisation des ressources dans un projet donné.

Procédure

1. Définissez le quota dans un fichier.
2. À utiliser le fichier pour créer le quota et l'appliquer à un projet:

```
$ oc create -f <file> [-n <project_name>]
```

À titre d'exemple:

```
$ oc create -f core-object-counts.yaml -n demoproject
```

7.1.6.1. Création de quotas de comptage d'objets

Il est possible de créer un quota de nombre d'objets pour tous les types de ressources standard d'espacement de noms sur Red Hat OpenShift Service sur AWS, tels que les objets BuildConfig et DeploymentConfig. Le nombre de quotas d'objets place un quota défini sur tous les types de ressources standard en espace de noms.

Lors de l'utilisation d'un quota de ressources, un objet est facturé sur le quota lors de la création. Ces types de quotas sont utiles pour protéger contre l'épuisement des ressources. Le quota ne peut être créé que s'il y a suffisamment de ressources de rechange dans le projet.

Procédure

Configurer un quota de nombre d'objets pour une ressource:

1. Exécutez la commande suivante:

```
$ oc create quota <name> \
    --hard=count/<resource>.<group>=<quota>,count/<resource>.<group>=<quota> 1
```

1 La variable <resource> est le nom de la ressource, et <group> est le groupe API, le cas échéant. Utilisez la commande oc api-ressources pour une liste de ressources et leurs groupes API associés.

À titre d'exemple:

```
$ oc create quota test \
    --
hard=count/deployments.extensions=2,count/replicasets.extensions=4,count/pods=3,count/secr
ets=4
```

Exemple de sortie

```
resourcequota "test" created
```

Cet exemple limite les ressources répertoriées à la limite dure de chaque projet du cluster.

2. Assurez-vous que le quota a été créé:

```
$ oc describe quota test
```

Exemple de sortie

```
Name:                       test
Namespace:                    quota
Resource                    Used  Hard
--------                   ----  ----
count/deployments.extensions  0     2
count/pods                    0     3
count/replicasets.extensions  0     4
count/secrets                 0     4
```

7.1.6.2. Fixation d'un quota de ressources pour les ressources étendues

Le surengagement des ressources n'est pas autorisé pour les ressources étendues, vous devez donc spécifier les demandes et les limites pour la même ressource étendue dans un quota. Actuellement, seuls les éléments de quota avec les demandes de préfixe. est autorisé pour les ressources étendues. Ce qui suit est un scénario d'exemple de la façon de définir le quota de ressources pour la ressource GPU nvidia.com/gpu.

Procédure

1. Déterminez combien de GPU sont disponibles sur un nœud dans votre cluster. À titre d'exemple:

```
# oc describe node ip-172-31-27-209.us-west-2.compute.internal | egrep
'Capacity|Allocatable|gpu'
```

Exemple de sortie

```
                    openshift.com/gpu-accelerator=true
Capacity:
 nvidia.com/gpu:  2
Allocatable:
 nvidia.com/gpu:  2
  nvidia.com/gpu  0           0
```

Dans cet exemple, 2 GPU sont disponibles.

2. Créez un objet ResourceQuota pour définir un quota dans l'espace de noms nvidia. Dans cet exemple, le quota est 1:

Exemple de sortie

```
apiVersion: v1
kind: ResourceQuota
```

```
metadata:
  name: gpu-quota
  namespace: nvidia
spec:
  hard:
    requests.nvidia.com/gpu: 1
```

3. Créer le quota:

```
# oc create -f gpu-quota.yaml
```

Exemple de sortie

```
resourcequota/gpu-quota created
```

4. Assurez-vous que l'espace de noms a le bon jeu de quotas:

```
# oc describe quota gpu-quota -n nvidia
```

Exemple de sortie

```
Name:                   gpu-quota
Namespace:              nvidia
Resource                Used  Hard
--------                ----  ----
requests.nvidia.com/gpu  0     1
```

5. Définissez un pod qui demande un seul GPU. Le fichier de définition d'exemple suivant est appelé gpu-pod.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  generateName: gpu-pod-
  namespace: nvidia
spec:
  restartPolicy: OnFailure
  containers:
  - name: rhel7-gpu-pod
    image: rhel7
    env:
      - name: NVIDIA_VISIBLE_DEVICES
        value: all
      - name: NVIDIA_DRIVER_CAPABILITIES
        value: "compute,utility"
      - name: NVIDIA_REQUIRE_CUDA
        value: "cuda>=5.0"
    command: ["sleep"]
    args: ["infinity"]
    resources:
      limits:
        nvidia.com/gpu: 1
```

6. Créer le pod:

```
# oc create -f gpu-pod.yaml
```

7. Assurez-vous que le pod est en cours d'exécution:

```
# oc get pods
```

Exemple de sortie

```
NAME            READY    STATUS     RESTARTS   AGE
gpu-pod-s46h7   1/1      Running    0          1m
```

8. Assurez-vous que le compteur utilisé est correct:

```
# oc describe quota gpu-quota -n nvidia
```

Exemple de sortie

```
Name:                   gpu-quota
Namespace:              nvidia
Resource                Used  Hard
--------                ----  ----
requests.nvidia.com/gpu  1     1
```

9. Essayez de créer un deuxième pod GPU dans l'espace de noms nvidia. Ceci est techniquement disponible sur le nœud car il dispose de 2 GPU:

```
# oc create -f gpu-pod.yaml
```

Exemple de sortie

```
Error from server (Forbidden): error when creating "gpu-pod.yaml": pods "gpu-pod-f7z2w" is
forbidden: exceeded quota: gpu-quota, requested: requests.nvidia.com/gpu=1, used:
requests.nvidia.com/gpu=1, limited: requests.nvidia.com/gpu=1
```

Ce message d'erreur interdit est attendu parce que vous avez un quota de 1 GPU et ce pod a essayé d'allouer un second GPU, qui dépasse son quota.

7.1.7. Affichage d'un quota

Les statistiques d'utilisation liées à toute limite dure définie dans le quota d'un projet peuvent être affichées en naviguant dans la console Web vers la page Quota du projet.

Il est également possible d'utiliser le CLI pour afficher les détails des quotas.

Procédure

1. Demandez la liste des quotas définis dans le projet. À titre d'exemple, pour un projet appelé démoproject:

```
$ oc get quota -n demoproject
```

Exemple de sortie

```
NAME                         AGE    REQUEST
LIMIT
besteffort                   4s     pods: 1/2
compute-resources-time-bound   10m    pods: 0/2
limits.cpu: 0/1, limits.memory: 0/1Gi
core-object-counts           109s   configmaps: 2/10, persistentvolumeclaims: 1/4,
replicationcontrollers: 1/20, secrets: 9/10, services: 2/10
```

2. Décrivez le quota qui vous intéresse, par exemple le quota des comptes d'objets de base:

```
$ oc describe quota core-object-counts -n demoproject
```

Exemple de sortie

```
Name:   core-object-counts
Namespace:  demoproject
Resource  Used Hard
--------  ---- ----
configmaps  3 10
persistentvolumeclaims 0 4
replicationcontrollers 3 20
secrets   9 10
services  2 10
```

7.1.8. Configuration des quotas de ressources explicites

Configurez des quotas de ressources explicites dans un modèle de demande de projet pour appliquer des quotas de ressources spécifiques dans de nouveaux projets.

Conditions préalables

- Accès au cluster en tant qu'utilisateur avec le rôle cluster-admin.
- Installez le OpenShift CLI (oc).

Procédure

1. Ajouter une définition de quota de ressources à un modèle de demande de projet:
   - Lorsqu'un modèle de demande de projet n'existe pas dans un cluster:
     a. Créer un modèle de projet bootstrap et le produire dans un fichier appelé template.yaml:

```
$ oc adm create-bootstrap-project-template -o yaml > template.yaml
```

     b. Ajoutez une définition de quota de ressource à template.yaml. L'exemple suivant définit un quota de ressources nommé « stockage-consommation ». La définition doit être ajoutée avant les paramètres: section dans le modèle:

```
- apiVersion: v1
```

```
kind: ResourceQuota
metadata:
  name: storage-consumption
  namespace: ${PROJECT_NAME}
spec:
  hard:
    persistentvolumeclaims: "10" (1)
    requests.storage: "50Gi" (2)
    gold.storageclass.storage.k8s.io/requests.storage: "10Gi" (3)
    silver.storageclass.storage.k8s.io/requests.storage: "20Gi" (4)
    silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5" (5)
    bronze.storageclass.storage.k8s.io/requests.storage: "0" (6)
    bronze.storageclass.storage.k8s.io/persistentvolumeclaims: "0" (7)
```

1. Le nombre total de revendications en volume persistant dans un projet.
2. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée ne peut pas dépasser cette valeur.
3. Dans toutes les revendications de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage aurifère ne peut pas dépasser cette valeur.
4. Dans toutes les revendications de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage d'argent ne peut pas dépasser cette valeur.
5. Dans toutes les revendications en volume persistant dans un projet, le nombre total de revendications dans la classe de stockage d'argent ne peut pas dépasser cette valeur.
6. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage de bronze ne peut pas dépasser cette valeur. Lorsque cette valeur est définie à 0, la classe de stockage bronze ne peut pas demander de stockage.
7. Dans toutes les demandes de volume persistantes dans un projet, la somme de stockage demandée dans la classe de stockage de bronze ne peut pas dépasser cette valeur. Lorsque cette valeur est définie à 0, la classe de stockage de bronze ne peut pas créer de revendications.

c. Créer un modèle de demande de projet à partir du fichier template.yaml modifié dans l'espace de noms openshift-config:

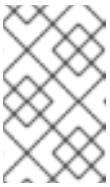
```
$ oc create -f template.yaml -n openshift-config
```

NOTE

Afin d'inclure la configuration en tant qu'annotation de configuration kubectl.kubernetes.io/last-applied-configuration, ajoutez l'option --save-config à la commande oc create.

Le modèle s'appelle par défaut project-request.

- Lorsqu'un modèle de demande de projet existe déjà au sein d'un cluster:



NOTE

Lorsque vous gérez des objets de manière déclarative ou impérative au sein de votre cluster en utilisant des fichiers de configuration, modifiez plutôt le modèle de demande de projet existant via ces fichiers.

a. Liste des modèles dans l'espace de noms openshift-config:

```
$ oc get templates -n openshift-config
```

b. Éditer un modèle de demande de projet existant:

```
$ oc edit template <project_request_template> -n openshift-config
```

c. Ajoutez une définition de quota de ressource, telle que l'exemple précédent de stockage-consommation, dans le modèle existant. La définition doit être ajoutée avant les paramètres : section dans le modèle.

2. Lorsque vous avez créé un modèle de demande de projet, faites-le référence dans la ressource de configuration de projet du cluster:

a. Accédez à la ressource de configuration du projet pour l'édition:

- En utilisant la console web:

i. Accédez à la page Administration → Paramètres du cluster.

ii. Cliquez sur Configuration pour afficher toutes les ressources de configuration.

iii. Cherchez l'entrée pour Projet et cliquez sur Modifier YAML.

- En utilisant le CLI:

i. Editez la ressource project.config.openshift.io/cluster:

```
$ oc edit project.config.openshift.io/cluster
```

b. Actualisez la section Spécification de la ressource de configuration du projet pour inclure les paramètres projectRequestTemplate et nom. L'exemple suivant fait référence au nom de modèle de demande de projet par défaut:

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestTemplate:
    name: project-request
```

3. Assurez-vous que le quota de ressources est appliqué lors de la création des projets:

a. Créer un projet:

```
$ oc new-project <project_name>
```

b. Énumérez les quotas de ressources du projet:

```
$ oc get resourcequotas
```

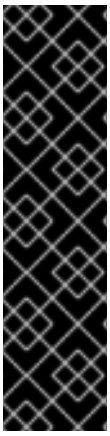
c. Décrivez en détail le quota de ressources:

```
$ oc describe resourcequotas <resource_quota_name>
```

7.2. QUOTAS DE RESSOURCES POUR PLUSIEURS PROJETS

Le quota multiprojet, défini par un objet ClusterResourceQuota, permet de partager les quotas entre plusieurs projets. Les ressources utilisées dans chaque projet sélectionné sont agrégées et cet agrégat est utilisé pour limiter les ressources pour tous les projets sélectionnés.

Ce guide décrit comment les administrateurs de clusters peuvent définir et gérer les quotas de ressources sur plusieurs projets.



IMPORTANT

Évitez d'exécuter des charges de travail ou de partager l'accès aux projets par défaut. Les projets par défaut sont réservés à l'exécution de composants de cluster de base.

Les projets par défaut suivants sont considérés comme hautement privilégiés: par défaut, kube-public, kube-system, openshift, openshift-infra, openshift-node, et d'autres projets créés par système qui ont l'étiquette openshift.io / run-level définie à 0 ou 1. La fonctionnalité qui repose sur des plugins d'admission, tels que l'admission de sécurité pod, les contraintes de contexte de sécurité, les quotas de ressources de cluster et la résolution de référence d'image, ne fonctionne pas dans des projets hautement privilégiés.

7.2.1. Choisir plusieurs projets lors de la création de quotas

Lorsque vous créez des quotas, vous pouvez sélectionner plusieurs projets en fonction de la sélection d'annotation, de la sélection d'étiquettes ou des deux.

Procédure

1. Afin de sélectionner des projets basés sur des annotations, exécutez la commande suivante:

```
$ oc create clusterquota for-user \
    --project-annotation-selector openshift.io/requester=<user_name> \
    --hard pods=10 \
    --hard secrets=20
```

Cela crée l'objet ClusterResourceQuota suivant:

```
apiVersion: quota.openshift.io/v1
kind: ClusterResourceQuota
metadata:
  name: for-user
spec:
```

```
  quota: (1)
    hard:
      pods: "10"
      secrets: "20"
  selector:
    annotations: (2)
      openshift.io/requester: <user_name>
    labels: null (3)
status:
  namespaces: (4)
  - namespace: ns-one
    status:
      hard:
        pods: "10"
        secrets: "20"
      used:
        pods: "1"
        secrets: "9"
  total: (5)
    hard:
      pods: "10"
      secrets: "20"
    used:
      pods: "1"
      secrets: "9"
```

(1) L'objet ResourceQuotaSpec qui sera appliqué sur les projets sélectionnés.

(2) C'est un simple sélecteur de valeur clé pour les annotations.

(3) Le sélecteur d'étiquettes qui peut être utilisé pour sélectionner des projets.

(4) Carte per-namespace qui décrit l'utilisation actuelle des quotas dans chaque projet sélectionné.

(5) L'utilisation globale de tous les projets sélectionnés.

Ce document de quota multi-projets contrôle tous les projets demandés par <user_name> à l'aide du point de terminaison de requête par défaut. Il est limité à 10 pods et 20 secrets.

2. De même, pour sélectionner des projets basés sur des étiquettes, exécutez cette commande:

```
$ oc create clusterresourcequota for-name \ (1)
    --project-label-selector=name=frontend \ (2)
    --hard=pods=10 --hard=secrets=20
```

(1) Les deux clusterssourcequota et clusterquota sont des alias de la même commande. for-name est le nom de l'objet ClusterResourceQuota.

(2) Afin de sélectionner des projets par étiquette, fournissez une paire clé-valeur en utilisant le format --project-label-selector=key=value.

Cela crée la définition d'objet ClusterResourceQuota suivante:

```
apiVersion: quota.openshift.io/v1
kind: ClusterResourceQuota
metadata:
  creationTimestamp: null
  name: for-name
spec:
  quota:
    hard:
      pods: "10"
      secrets: "20"
  selector:
    annotations: null
    labels:
      matchLabels:
        name: frontend
```

7.2.2. Affichage des quotas de ressources de cluster applicables

L'administrateur de projet n'est pas autorisé à créer ou modifier le quota multiprojets qui limite son projet, mais l'administrateur est autorisé à consulter les documents de quotas multiprojets qui sont appliqués à son projet. L'administrateur de projet peut le faire via la ressource AppliedClusterResourceQuota.

Procédure

1. Afin de voir les quotas appliqués à un projet, exécutez:

```
$ oc describe AppliedClusterResourceQuota
```

Exemple de sortie

```
Name:   for-user
Namespace:  <none>
Created:  19 hours ago
Labels:   <none>
Annotations:  <none>
Label Selector: <null>
AnnotationSelector: map[openshift.io/requester:<user-name>]
Resource  Used  Hard
--------  ----  ----
pods      1     10
secrets   9     20
```

7.2.3. Granularité de sélection

En raison du blocage de la demande d'allocations de quotas, le nombre de projets actifs sélectionnés par un quota multiprojets est une considération importante. La sélection de plus de 100 projets dans le cadre d'un seul quota multi-projets peut avoir des effets néfastes sur la réactivité des serveurs API dans ces projets.

CHAPITRE 8. CONFIGURATION DES CARTES AVEC DES APPLICATIONS

Les cartes de configuration vous permettent de découpler les artefacts de configuration du contenu de l'image pour garder les applications conteneurisées portables.

Les sections suivantes définissent les cartes de configuration et comment les créer et les utiliser.

8.1. COMPRENDRE LES CARTES DE CONFIGURATION

De nombreuses applications nécessitent une configuration en utilisant une combinaison de fichiers de configuration, d'arguments de ligne de commande et de variables d'environnement. Dans Red Hat OpenShift Service sur AWS, ces artefacts de configuration sont découplés du contenu de l'image pour garder les applications conteneurisées portables.

L'objet ConfigMap fournit des mécanismes pour injecter des conteneurs avec des données de configuration tout en gardant les conteneurs agnostiques de Red Hat OpenShift Service sur AWS. La carte de configuration peut être utilisée pour stocker des informations fines telles que des propriétés individuelles ou des informations grossières comme des fichiers de configuration entiers ou des blobs JSON.

L'objet ConfigMap contient des paires clés-valeur de données de configuration qui peuvent être consommées dans des pods ou utilisées pour stocker des données de configuration pour des composants système tels que des contrôleurs. À titre d'exemple:

Définition d'objet de ConfigMap

```
kind: ConfigMap
apiVersion: v1
metadata:
  creationTimestamp: 2016-02-18T19:14:38Z
  name: example-config
  namespace: my-namespace
data: 1
  example.property.1: hello
  example.property.2: world
  example.property.file: |-
    property.1=value-1
    property.2=value-2
    property.3=value-3
binaryData:
  bar: L3Jvb3QvMTAw 2
```

1 Contient les données de configuration.

2 Indique un fichier qui contient des données non-UTF8, par exemple un fichier binaire Java keystore. Entrez les données du fichier dans la base 64.

NOTE

Lorsque vous créez une carte de configuration à partir d'un fichier binaire, vous pouvez utiliser le champ de données binaires, comme une image.

Les données de configuration peuvent être consommées en pods de diverses manières. La carte de configuration peut être utilisée pour:

- Peupler les valeurs variables d'environnement dans les conteneurs
- Définir les arguments de ligne de commande dans un conteneur
- Populer des fichiers de configuration dans un volume

Les utilisateurs et les composants système peuvent stocker les données de configuration dans une carte de configuration.

La carte de configuration est similaire à un secret, mais conçue pour prendre en charge plus facilement le travail avec des chaînes qui ne contiennent pas d'informations sensibles.

Configuration des restrictions de carte

Il faut créer une carte de configuration avant que son contenu puisse être consommé en pods.

Les contrôleurs peuvent être écrits pour tolérer les données de configuration manquantes. Consultez les composants individuels configurés en utilisant des cartes de configuration au cas par cas.

Les objets ConfigMap résident dans un projet.

Ils ne peuvent être référencés que par des pods dans le même projet.

Le Kubelet ne prend en charge que l'utilisation d'une carte de configuration pour les pods qu'il obtient du serveur API.

Cela inclut tous les pods créés en utilisant le CLI, ou indirectement à partir d'un contrôleur de réplication. Il n'inclut pas les pods créés en utilisant le service Red Hat OpenShift sur le drapeau --manifest-url d'AWS node, son drapeau --config ou son API REST parce que ce ne sont pas des moyens courants de créer des pods.

Ressources supplémentaires

- Création et utilisation de cartes de configuration

8.2. CAS D'UTILISATION: CONSOMMER DES CARTES DE CONFIGURATION DANS DES PODS

Les sections suivantes décrivent certains cas d'utilisation lors de la consommation d'objets ConfigMap dans des pods.

8.2.1. Populer des variables d'environnement dans les conteneurs en utilisant des cartes de configuration

Il est possible d'utiliser des cartes de configuration pour remplir des variables d'environnement individuelles dans des conteneurs ou pour remplir des variables d'environnement dans des conteneurs à partir de toutes les clés qui forment des noms de variables d'environnement valides.

À titre d'exemple, considérez la carte de configuration suivante:

ConfigMap avec deux variables d'environnement

```
apiVersion: v1
```

```
kind: ConfigMap
metadata:
  name: special-config 1
  namespace: default 2
data:
  special.how: very 3
  special.type: charm 4
```

1 Le nom de la carte de configuration.

2 Le projet dans lequel réside la carte de configuration. Les cartes de configuration ne peuvent être référencées que par des pods dans le même projet.

3 4 Les variables d'environnement à injecter.

ConfigMap avec une variable d'environnement

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-config 1
  namespace: default
data:
  log_level: INFO 2
```

1 Le nom de la carte de configuration.

2 Environnement variable à injecter.

Procédure

- Il est possible de consommer les clés de ce ConfigMap dans un pod à l'aide des sections configMapKeyRef.

  ### Exemple de spécification de Pod configurée pour injecter des variables d'environnement spécifiques

  ```
  apiVersion: v1
  kind: Pod
  metadata:
    name: dapi-test-pod
  spec:
    securityContext:
      runAsNonRoot: true
      seccompProfile:
        type: RuntimeDefault
    containers:
      - name: test-container
        image: gcr.io/google_containers/busybox
        command: [ "/bin/sh", "-c", "env" ]
        env: 1
          - name: SPECIAL_LEVEL_KEY 2
  ```

```
      valueFrom:
        configMapKeyRef:
          name: special-config 3
          key: special.how 4
    - name: SPECIAL_TYPE_KEY
      valueFrom:
        configMapKeyRef:
          name: special-config 5
          key: special.type 6
          optional: true 7
  envFrom: 8
    - configMapRef:
        name: env-config 9
  securityContext:
    allowPrivilegeEscalation: false
    capabilities:
      drop: [ALL]
restartPolicy: Never
```

1. Il permet d'extraire les variables d'environnement spécifiées à partir d'un ConfigMap.
2. Le nom d'une variable d'environnement de pod dans laquelle vous injectez la valeur d'une clé.
3. 5 Le nom du ConfigMap pour tirer des variables d'environnement spécifiques.
4. 6 Environnement variable à tirer du ConfigMap.
7. Rend l'environnement variable optionnel. En option, le pod sera démarré même si le ConfigMap spécifié et les clés n'existent pas.
8. Il permet d'extraire toutes les variables d'environnement d'un ConfigMap.
9. Le nom du ConfigMap pour tirer toutes les variables d'environnement.

Lorsque ce pod est exécuté, les logs de pod incluront la sortie suivante:

```
SPECIAL_LEVEL_KEY=very
log_level=INFO
```

NOTE

Le code SPECIAL_TYPE_KEY=charm n'est pas listé dans la sortie de l'exemple car optionnel: true est défini.

8.2.2. Définition des arguments de ligne de commande pour les commandes de conteneur avec des cartes de configuration

Il est possible d'utiliser une carte de configuration pour définir la valeur des commandes ou des arguments dans un conteneur à l'aide de la syntaxe de substitution de Kubernetes $(VAR_NAME).

À titre d'exemple, considérez la carte de configuration suivante:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  special.how: very
  special.type: charm
```

Procédure

- Afin d'injecter des valeurs dans une commande dans un conteneur, vous devez consommer les clés que vous souhaitez utiliser comme variables d'environnement. Ensuite, vous pouvez vous référer à eux dans la commande d'un conteneur en utilisant la syntaxe $(VAR_NAME).

  Exemple de spécification de la gousse configurée pour injecter des variables d'environnement spécifiques

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "echo $(SPECIAL_LEVEL_KEY) $(SPECIAL_TYPE_KEY)" ]
(1)
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.how
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.type
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          drop: [ALL]
  restartPolicy: Never
```

(1) Injectez les valeurs dans une commande dans un conteneur en utilisant les clés que vous souhaitez utiliser comme variables d'environnement.

Lorsque ce pod est exécuté, la sortie de la commande écho s'exécute dans le conteneur test-conteneur comme suit:

```
very charm
```

8.2.3. Injecter du contenu dans un volume en utilisant des cartes de configuration

Il est possible d'injecter du contenu dans un volume en utilisant des cartes de configuration.

Exemple de ressource personnalisée ConfigMap (CR)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  special.how: very
  special.type: charm
```

Procédure

Il existe plusieurs options différentes pour injecter du contenu dans un volume en utilisant des cartes de configuration.

- La façon la plus basique d'injecter du contenu dans un volume en utilisant une carte de configuration est de remplir le volume avec des fichiers où la clé est le nom du fichier et le contenu du fichier est la valeur de la clé:

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "cat", "/etc/config/special.how" ]
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          drop: [ALL]
  volumes:
    - name: config-volume
      configMap:
        name: special-config 1
  restartPolicy: Never
```

1 Fichier contenant la clé.

Lorsque ce pod est exécuté, la sortie de la commande chat sera:

```
very
```

- Il est également possible de contrôler les chemins dans le volume où sont projetées les touches cartographiques de configuration:

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "cat", "/etc/config/path/to/special-key" ]
      volumeMounts:
      - name: config-volume
        mountPath: /etc/config
      securityContext:
        allowPrivilegeEscalation: false
        capabilities:
          drop: [ALL]
  volumes:
    - name: config-volume
      configMap:
        name: special-config
        items:
        - key: special.how
          path: path/to/special-key 1
  restartPolicy: Never
```

1 Chemin vers la configuration de la clé map.

Lorsque ce pod est exécuté, la sortie de la commande chat sera:

```
very
```

CHAPITRE 9. CONTRÔLE DES MÉTRIQUES DE PROJET ET D'APPLICATION EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

La vue Observer dans la perspective Développeur fournit des options pour surveiller vos métriques de projet ou d'application, telles que CPU, mémoire, et l'utilisation de la bande passante, et les informations liées au réseau.

9.1. CONDITIONS PRÉALABLES

- Des applications ont été créées et déployées sur Red Hat OpenShift Service sur AWS.
- Connectez-vous à la console Web et passez à la perspective Développeur.

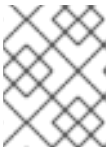
9.2. LE SUIVI DE VOS MÉTRIQUES DE PROJET

Après avoir créé des applications dans votre projet et les avoir déployées, vous pouvez utiliser la perspective Développeur dans la console Web pour voir les métriques de votre projet.

Procédure

1. Allez à Observer pour voir le tableau de bord, les métriques, les alertes et les événements pour votre projet.
2. Facultatif: Utilisez l'onglet Tableau de bord pour voir les graphiques représentant les paramètres d'application suivants:
   - L'utilisation du CPU
   - L'utilisation de la mémoire
   - Consommation de bande passante
   - Informations liées au réseau telles que le taux de paquets transmis et reçus et le taux de paquets abandonnés.

   Dans l'onglet Tableau de bord, vous pouvez accéder aux tableaux de bord des ressources de calcul Kubernetes.

   NOTE

   Dans la liste Tableau de bord, le tableau de bord Kubernetes / Compute Resources / Namespace (Pods) est sélectionné par défaut.

   Les options suivantes permettent d'en savoir plus:

   - Choisissez un tableau de bord dans la liste des tableaux de bord pour voir les métriques filtrées. Les tableaux de bord produisent des sous-menus supplémentaires lorsqu'ils sont sélectionnés, à l'exception de Kubernetes / Compute Resources / Namespace (Pods).
   - Choisissez une option dans la liste de l'intervalle de temps pour déterminer le délai pour les données saisies.

- Définissez une plage de temps personnalisée en sélectionnant la plage de temps personnalisée dans la liste Time Range. Il est possible d'entrer ou de sélectionner les dates et heures de From and To. Cliquez sur Enregistrer pour enregistrer la plage de temps personnalisée.

- Choisissez une option dans la liste d'intervalle de rafraîchissement pour déterminer la période après laquelle les données sont actualisées.

- Déplacez votre curseur sur les graphiques pour voir les détails spécifiques de votre pod.

- Cliquez sur Inspecter situé dans le coin supérieur droit de chaque graphique pour voir les détails du graphique. Les détails du graphique apparaissent dans l'onglet Metrics.

3. Facultatif: Utilisez l'onglet Metrics pour interroger la métrique de projet requise.

Figure 9.1. Indicateurs de surveillance

a. Dans la liste Sélectionner la requête, sélectionnez une option pour filtrer les détails requis pour votre projet. Les métriques filtrées pour tous les pods d'application de votre projet sont affichées dans le graphique. Les pods de votre projet sont également listés ci-dessous.

b. À partir de la liste des gousses, effacer les cases carrées colorées pour supprimer les métriques pour les gousses spécifiques afin de filtrer davantage le résultat de votre requête.

c. Cliquez sur Afficher PromQL pour voir la requête Prometheus. En outre, vous pouvez modifier cette requête à l'aide d'invites pour personnaliser la requête et filtrer les métriques que vous souhaitez voir pour cet espace de noms.

d. La liste déroulante vous permet de définir une plage de temps pour les données affichées. Cliquez sur Réinitialiser Zoom pour le réinitialiser dans la plage de temps par défaut.

e. Facultatif: Dans la liste Sélectionner la requête, sélectionnez la requête personnalisée pour créer une requête Prometheus personnalisée et filtrer les métriques pertinentes.

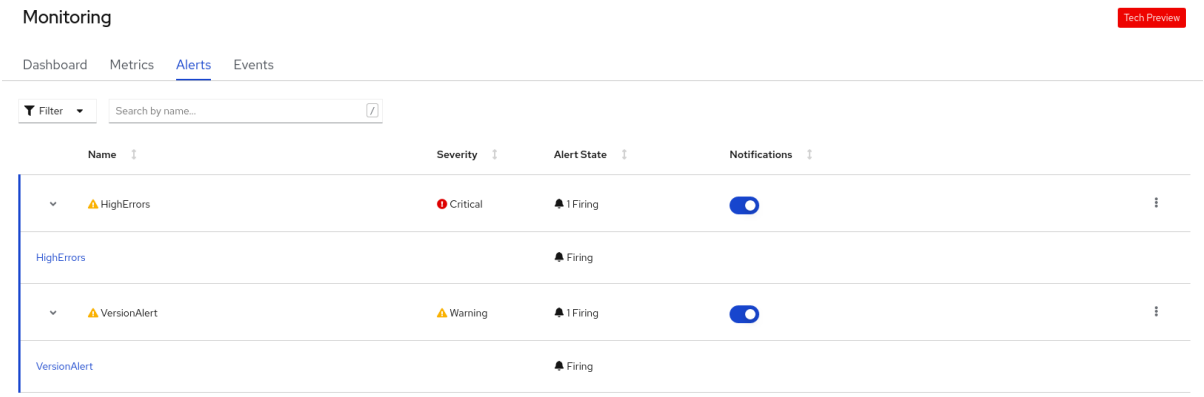
4. Facultatif: Utilisez l'onglet Alertes pour effectuer les tâches suivantes:

- Consultez les règles qui déclenchent des alertes pour les applications de votre projet.

- Identifiez les alertes qui se déclenchent dans le projet.

- Faites taire ces alertes si nécessaire.

Figure 9.2. Alertes de surveillance

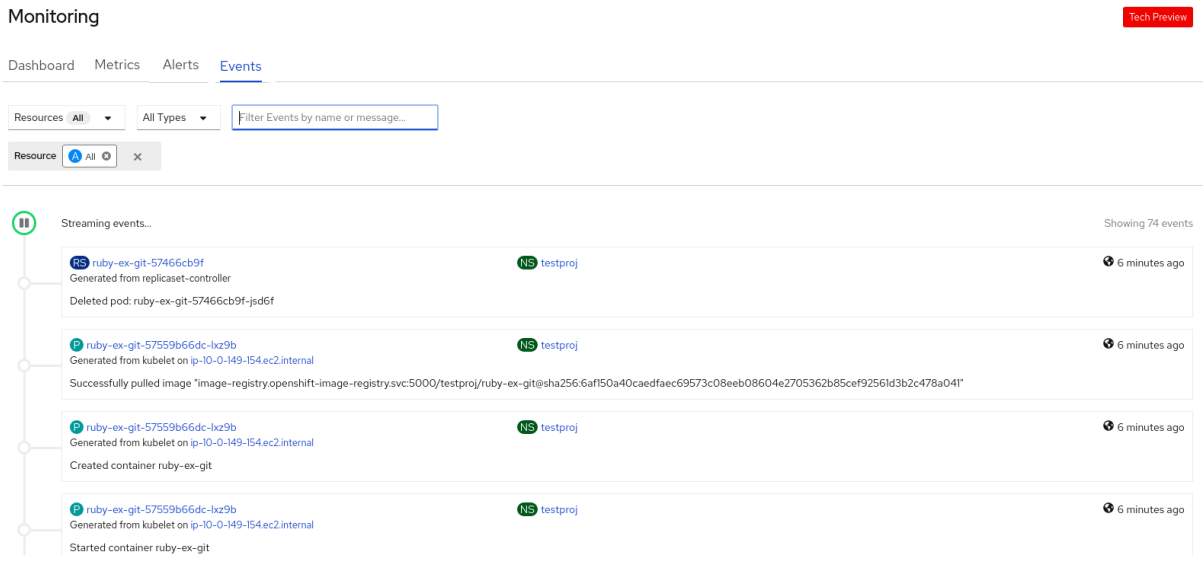


Les options suivantes permettent d'en savoir plus:

- La liste des filtres permet de filtrer les alertes par leur état d'alerte et leur gravité.
- Cliquez sur une alerte pour accéder à la page de détails de cette alerte. Dans la page Détails des alertes, vous pouvez cliquer sur Afficher les métriques pour voir les métriques de l'alerte.
- Cliquez sur les notifications pour activer une règle d'alerte pour réduire au silence toutes les alertes pour cette règle, puis sélectionnez la durée pour laquelle les alertes seront réduites au silence dans la liste Silence. Il faut avoir les autorisations d'éditer les alertes pour voir les notifications basculer.
- Dans le menu Options adjacentes à une règle d'alerte, consultez les détails de la règle d'alerte.

5. Facultatif: Utilisez l'onglet Événements pour voir les événements de votre projet.

Figure 9.3. Le suivi des événements



Il est possible de filtrer les événements affichés à l'aide des options suivantes:

- Dans la liste Ressources, sélectionnez une ressource pour voir les événements de cette ressource.
- Dans la liste Tous les types, sélectionnez un type d'événement pour voir les événements pertinents à ce type.

- Cherchez des événements spécifiques à l'aide du champ Filtrer par noms ou messages.

9.3. CONTRÔLE DE VOS MÉTRIQUES D'APPLICATION

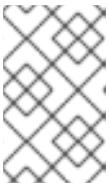
Après avoir créé des applications dans votre projet et les avoir déployées, vous pouvez utiliser la vue Topology dans la perspective Développeur pour voir les alertes et les métriques de votre application. Les alertes critiques et d'avertissement pour votre application sont indiquées sur le nœud de charge de travail dans la vue Topology.

Procédure

Afin de voir les alertes pour votre charge de travail:

1. Dans la vue Topologie, cliquez sur la charge de travail pour voir les détails de la charge de travail dans le panneau de droite.

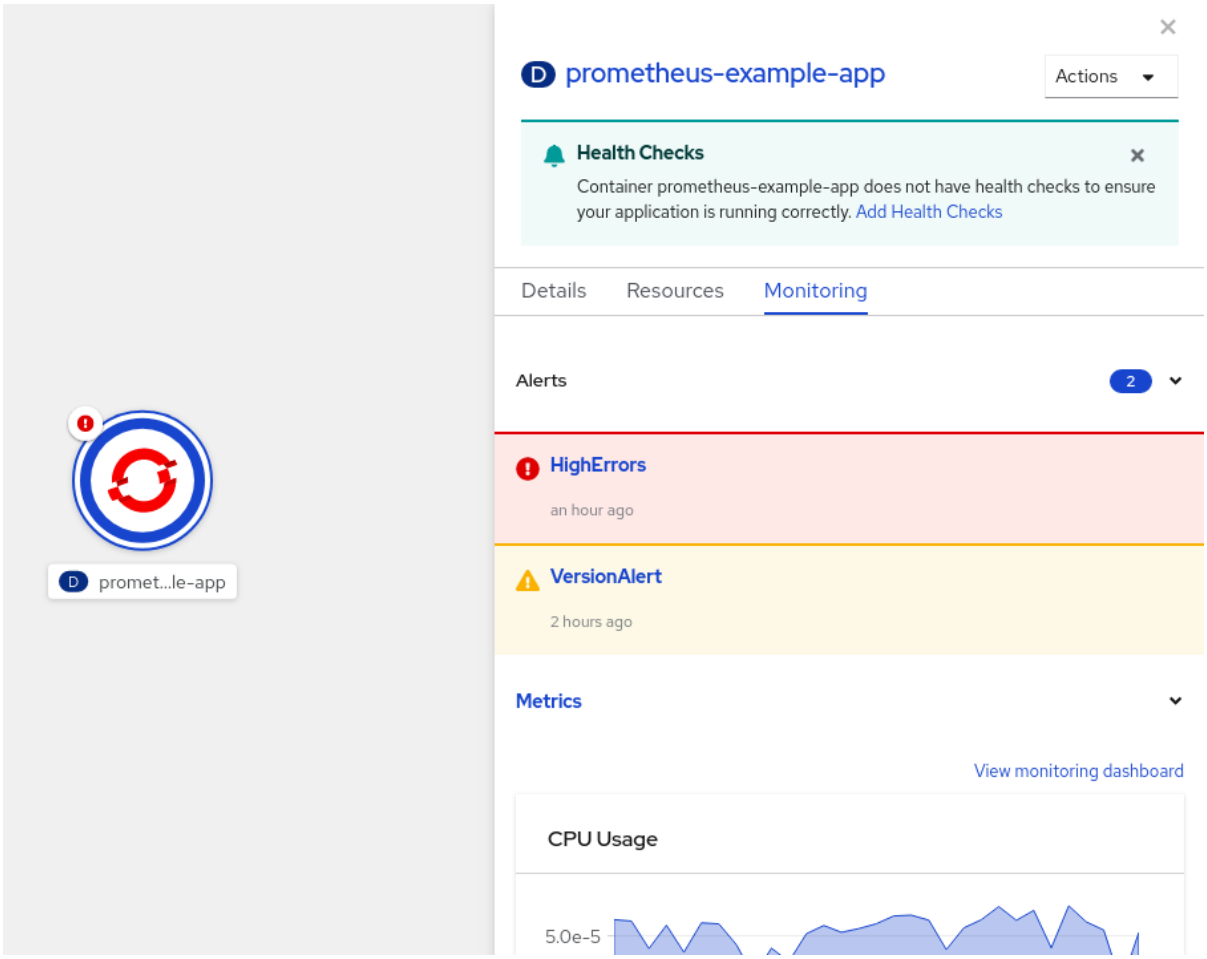
2. Cliquez sur l'onglet Observer pour voir les alertes critiques et d'avertissement pour l'application; graphiques pour les métriques, telles que CPU, mémoire et utilisation de bande passante; et tous les événements pour l'application.



NOTE

Les alertes critiques et les alertes d'avertissement dans l'état de Firing sont affichées dans la vue Topologie. Les alertes dans les états Silenced, En attente et Non Firing ne sont pas affichées.

Figure 9.4. Contrôle des paramètres d'application



a. Cliquez sur l'alerte listée dans le panneau de droite pour voir les détails de l'alerte dans la page Détails de l'alerte.

b. Cliquez sur n'importe lequel des graphiques pour aller à l'onglet Metrics pour voir les métriques détaillées de l'application.

c. Cliquez sur Afficher le tableau de bord de surveillance pour voir le tableau de bord de surveillance de cette application.

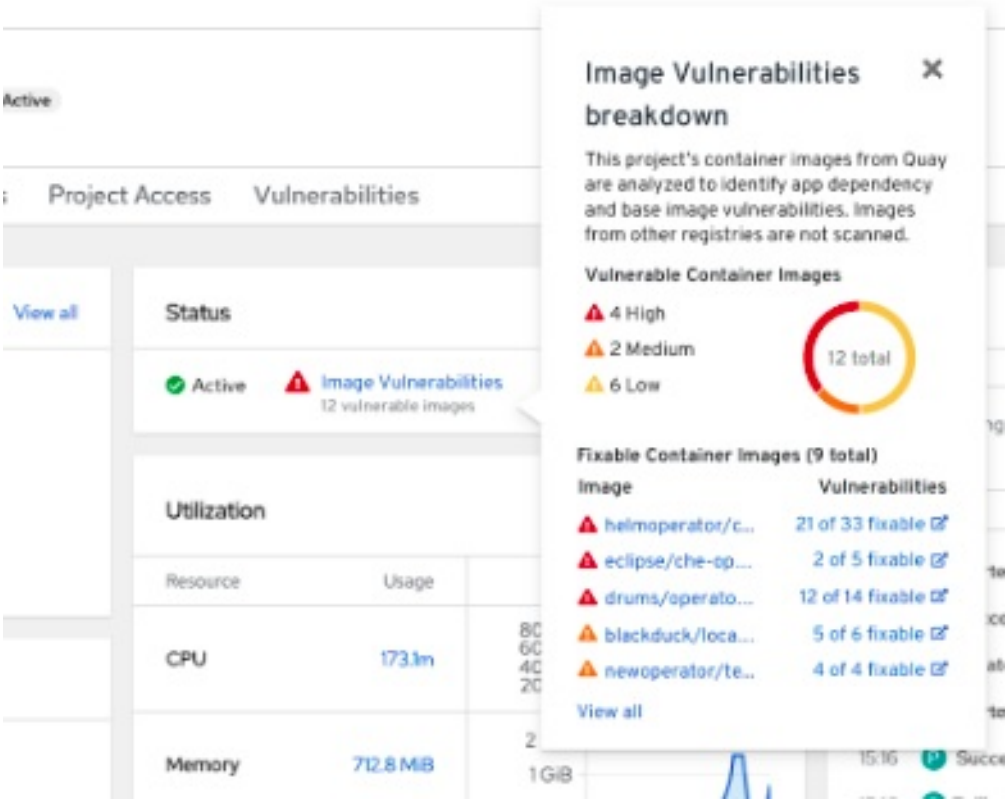
9.4. DÉCOMPRESSION DES VULNÉRABILITÉS DE L'IMAGE

Dans la perspective Développeur, le tableau de bord du projet affiche le lien Vulnérabilités d'image dans la section État. À l'aide de ce lien, vous pouvez afficher la fenêtre de décompression des vulnérabilités d'image, qui comprend des détails concernant les images de conteneur vulnérables et les images de conteneur fixes. La couleur de l'icône indique la gravité:

- Rouge : Priorité élevée. Fixez-vous immédiatement.
- L'orange : priorité moyenne. Il peut être corrigé après des vulnérabilités hautement prioritaires.
- Jaune : Faible priorité. Il peut être corrigé après des vulnérabilités à priorité élevée et moyenne.

En fonction du niveau de gravité, vous pouvez prioriser les vulnérabilités et les corriger de manière organisée.

Figure 9.5. Affichage des vulnérabilités de l'image



9.5. CONTRÔLE DES PARAMÈTRES DES VULNÉRABILITÉS DE VOTRE APPLICATION ET DE L'IMAGE

Après avoir créé des applications dans votre projet et les avoir déployées, utilisez la perspective Développeur dans la console Web pour voir les paramètres des vulnérabilités de dépendance de votre

application dans votre cluster. Les métriques vous aident à analyser en détail les vulnérabilités d'image suivantes:

- Comptage total des images vulnérables dans un projet sélectionné
- Comptage basé sur la gravité de toutes les images vulnérables dans un projet sélectionné
- Forage en sévérité pour obtenir les détails, tels que le nombre de vulnérabilités, le nombre de vulnérabilités fixables et le nombre de pods affectés pour chaque image vulnérable

Conditions préalables

- L'opérateur Red Hat Quay Container Security est installé sur le hub de l'opérateur.

NOTE

L'opérateur Red Hat Quay Container Security détecte les vulnérabilités en scannant les images qui se trouvent dans le registre du quay.

Procédure

1. Dans le panneau de navigation de la perspective Développeur, cliquez sur Projet pour voir le tableau de bord du projet.
2. Cliquez sur Vulnérabilités d'image dans la section État. La fenêtre qui s'ouvre affiche des détails tels que les images de conteneurs vulnérables et les images de conteneurs fixes.
3. Cliquez sur l'onglet Vulnérabilités sur le tableau de bord du projet pour obtenir un aperçu détaillé des vulnérabilités.
    a. Cliquez sur son nom pour obtenir plus de détails sur une image.
    b. Affichez le graphique par défaut avec tous les types de vulnérabilités dans l'onglet Détails.
    c. Facultatif: Cliquez sur le bouton basculer pour afficher un type spécifique de vulnérabilité. Cliquez par exemple sur Apppendance pour voir les vulnérabilités spécifiques à la dépendance des applications.
    d. Facultatif : Vous pouvez filtrer la liste des vulnérabilités en fonction de leur gravité et de leur type ou les trier par Severity, Package, Type, Source, Version actuelle et Fixée dans la version.
    e. Cliquez sur une vulnérabilité pour obtenir les détails associés:
        - Les vulnérabilités de l'image de base affichent des informations provenant d'un Red Hat Security Advisory (RHSA).
        - Les vulnérabilités de dépendance des applications affichent les informations de l'application de sécurité Snyk.

9.6. RESSOURCES SUPPLÉMENTAIRES

- Aperçu du suivi

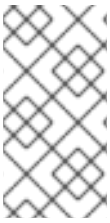
CHAPITRE 10. CONTRÔLE DE LA SANTÉ DES APPLICATIONS EN UTILISANT DES CONTRÔLES DE SANTÉ

Dans les systèmes logiciels, les composants peuvent devenir malsains en raison de problèmes transitoires tels que la perte de connectivité temporaire, les erreurs de configuration ou les problèmes de dépendances externes. Le service OpenShift Red Hat sur les applications AWS dispose d'un certain nombre d'options pour détecter et gérer des conteneurs malsains.

10.1. COMPRENDRE LES CONTRÔLES DE SANTÉ

Le bilan de santé effectue périodiquement des diagnostics sur un conteneur en cours d'exécution en utilisant n'importe quelle combinaison de la préparation, de la vivacité et des contrôles de santé au démarrage.

Il est possible d'inclure une ou plusieurs sondes dans la spécification de la gousse qui contient le contenant que vous souhaitez effectuer les contrôles de santé.



NOTE

Lorsque vous souhaitez ajouter ou modifier des contrôles de santé dans un pod existant, vous devez modifier l'objet Pod DeploymentConfig ou utiliser la perspective Développeur dans la console Web. Il n'est pas possible d'utiliser le CLI pour ajouter ou modifier des contrôles de santé pour un pod existant.

La sonde de préparation

La sonde de préparation détermine si un conteneur est prêt à accepter les demandes de service. En cas de défaillance de la sonde de préparation pour un conteneur, le kubelet retire la gousse de la liste des terminaux de service disponibles.
Après une défaillance, la sonde continue d'examiner la gousse. Lorsque la gousse devient disponible, le kubelet ajoute le pod à la liste des points de terminaison de service disponibles.

Contrôle de l'état de vie

La sonde de vivacité détermine si un conteneur est toujours en cours d'exécution. En cas d'échec de la sonde de vivacité en raison d'une condition telle qu'une impasse, le kubelet tue le conteneur. Le pod répond ensuite en fonction de sa politique de redémarrage.
À titre d'exemple, une sonde de vivacité sur une gousse avec redémarragePolicy of Always or OnFailure tue et redémarre le conteneur.

La sonde de démarrage

La sonde de démarrage indique si l'application dans un conteneur est lancée. Les autres sondes sont désactivées jusqu'à ce que la startup réussisse. Lorsque la sonde de démarrage ne réussit pas dans un délai spécifié, le kubelet tue le conteneur, et le conteneur est soumis à la police de redémarrage. Certaines applications peuvent nécessiter un temps de démarrage supplémentaire lors de leur première initialisation. Il est possible d'utiliser une sonde de démarrage avec une sonde de vivacité ou de préparation pour retarder cette sonde assez longtemps pour gérer le temps de démarrage long en utilisant les paramètres de panneThreshold et periodSeconds.

A titre d'exemple, vous pouvez ajouter une sonde de démarrage, avec un seuil de 30 défaillances et une périodeDeuxièmes de 10 secondes (30 * 10s = 300s) pour un maximum de 5 minutes, à une sonde de vivacité. Après que la sonde de démarrage réussit la première fois, la sonde de vivacité prend le relais.

Configurez des sondes de vivacité, de préparation et de démarrage avec l'un des types de tests suivants:

- HTTP GET: Lors de l'utilisation d'un test HTTP GET, le test détermine la santé du conteneur en utilisant un crochet Web. Le test est réussi si le code de réponse HTTP se situe entre 200 et 399.
  Il est possible d'utiliser un test HTTP GET avec des applications qui renvoient les codes d'état HTTP lorsqu'ils sont complètement initialisés.

- Commande de conteneur : Lors de l'utilisation d'un test de commande de conteneur, la sonde exécute une commande à l'intérieur du conteneur. La sonde est réussie si le test sort avec un statut 0.

- Douille TCP: Lors de l'utilisation d'un test de prise TCP, la sonde tente d'ouvrir une prise sur le conteneur. Le récipient n'est considéré comme sain que si la sonde peut établir une connexion. Il est possible d'utiliser un test de socket TCP avec des applications qui ne commencent pas à écouter jusqu'à ce que l'initialisation soit terminée.

Configurez plusieurs champs pour contrôler le comportement d'une sonde:

- initialDelaySeconds: Le temps, en secondes, après le démarrage du conteneur avant que la sonde puisse être programmée. La valeur par défaut est 0.

- le délai, en secondes, entre l'exécution des sondes. La valeur par défaut est 10. Cette valeur doit être supérieure au timeoutSeconds.

- chronométrageSeconds: Le nombre de secondes d'inactivité après quoi la sonde est sortie et le conteneur est supposé avoir échoué. La valeur par défaut est 1. Cette valeur doit être inférieure à periodSeconds.

- le nombre de fois que la sonde doit signaler le succès après un échec à réinitialiser l'état du conteneur pour réussir. La valeur doit être de 1 pour une sonde de vivacité. La valeur par défaut est 1.

- failThreshold: Le nombre de fois où la sonde est autorisée à échouer. La valeur par défaut est 3. Après les tentatives spécifiées:
  - dans le cas d'une sonde de vivacité, le conteneur est redémarré
  - dans le cas d'une sonde de préparation, le pod est marqué
  - dans le cas d'une sonde de démarrage, le conteneur est tué et est soumis au redémarrage du podPolicy

Exemples de sondes

Ce qui suit sont des échantillons de sondes différentes comme ils apparaîtraient dans une spécification d'objet.

Échantillon de sonde de préparation avec une sonde de préparation de commande de conteneur dans un pod spec

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: health-check
  name: my-application
```

```
# ...
spec:
  containers:
  - name: goproxy-app 1
    args:
    image: registry.k8s.io/goproxy:0.1 2
    readinessProbe: 3
      exec: 4
        command: 5
        - cat
        - /tmp/healthy
# ...
```

1. Le nom du conteneur.
2. L'image du conteneur à déployer.
3. C'est une sonde de préparation.
4. Essai de commande de conteneur.
5. Les commandes à exécuter sur le conteneur.

Exemple de sonde de démarrage de commande de conteneur et sonde de vivacité avec des tests de commande de conteneur dans un pod spec

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: health-check
  name: my-application
# ...
spec:
  containers:
  - name: goproxy-app 1
    args:
    image: registry.k8s.io/goproxy:0.1 2
    livenessProbe: 3
      httpGet: 4
        scheme: HTTPS 5
        path: /healthz
        port: 8080 6
        httpHeaders:
        - name: X-Custom-Header
          value: Awesome
    startupProbe: 7
      httpGet: 8
        path: /healthz
        port: 8080 9
      failureThreshold: 30 10
      periodSeconds: 10 11
# ...
```

1. Le nom du conteneur.
2. Indiquez l'image du conteneur à déployer.
3. C'est une sonde de vivacité.
4. Le test HTTP GET.
5. Le schéma Internet : HTTP ou HTTPS. La valeur par défaut est HTTP.
6. Le port sur lequel le conteneur est à l'écoute.
7. La sonde de démarrage.
8. Le test HTTP GET.
9. Le port sur lequel le conteneur est à l'écoute.
10. Le nombre de fois pour essayer la sonde après une défaillance.
11. Le nombre de secondes pour effectuer la sonde.

Échantillon de sonde de vivacité avec un test de commande de conteneur qui utilise un timeout dans un pod spec

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: health-check
  name: my-application
# ...
spec:
  containers:
  - name: goproxy-app 1
    args:
    image: registry.k8s.io/goproxy:0.1 2
    livenessProbe: 3
      exec: 4
        command: 5
        - /bin/bash
        - '-c'
        - timeout 60 /opt/eap/bin/livenessProbe.sh
      periodSeconds: 10 6
      successThreshold: 1 7
      failureThreshold: 3 8
# ...
```

1. Le nom du conteneur.
2. Indiquez l'image du conteneur à déployer.
3. La sonde de vivacité.

4. Le type de sonde, ici une sonde de commande de conteneur.
5. La ligne de commande à exécuter à l'intérieur du conteneur.
6. Combien de fois en secondes pour effectuer la sonde.
7. Le nombre de succès consécutifs nécessaires pour montrer le succès après un échec.
8. Le nombre de fois pour essayer la sonde après une défaillance.

Échantillon de sonde de préparation et sonde de vivacité avec un test de prise TCP dans un déploiement

```
kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    test: health-check
  name: my-application
spec:
# ...
  template:
    spec:
      containers:
        - resources: {}
          readinessProbe: 1
            tcpSocket:
              port: 8080
            timeoutSeconds: 1
            periodSeconds: 10
            successThreshold: 1
            failureThreshold: 3
          terminationMessagePath: /dev/termination-log
          name: ruby-ex
          livenessProbe: 2
            tcpSocket:
              port: 8080
            initialDelaySeconds: 15
            timeoutSeconds: 1
            periodSeconds: 10
            successThreshold: 1
            failureThreshold: 3
# ...
```

1. La sonde de préparation.
2. La sonde de vivacité.

10.2. CONFIGURATION DES CONTRÔLES DE SANTÉ À L'AIDE DU CLI

Afin de configurer la préparation, la vivacité et les sondes de démarrage, ajoutez une ou plusieurs sondes à la spécification de la gousse qui contient le conteneur que vous souhaitez effectuer les contrôles de santé

NOTE

Lorsque vous souhaitez ajouter ou modifier des contrôles de santé dans un pod existant, vous devez modifier l'objet Pod DeploymentConfig ou utiliser la perspective Développeur dans la console Web. Il n'est pas possible d'utiliser le CLI pour ajouter ou modifier des contrôles de santé pour un pod existant.

Procédure

Ajouter des sondes pour un conteneur:

1. Créez un objet Pod pour ajouter une ou plusieurs sondes:

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: health-check
  name: my-application
spec:
  containers:
  - name: my-container 1
    args:
    image: registry.k8s.io/goproxy:0.1 2
    livenessProbe: 3
      tcpSocket: 4
        port: 8080 5
      initialDelaySeconds: 15 6
      periodSeconds: 20 7
      timeoutSeconds: 10 8
    readinessProbe: 9
      httpGet: 10
        host: my-host 11
        scheme: HTTPS 12
        path: /healthz
        port: 8080 13
    startupProbe: 14
      exec: 15
        command: 16
        - cat
        - /tmp/healthy
      failureThreshold: 30 17
      periodSeconds: 20 18
      timeoutSeconds: 10 19
```

1. Indiquez le nom du conteneur.
2. Indiquez l'image du conteneur à déployer.
3. Facultatif: Créez une sonde Liveness.
4. Indiquez un test à effectuer, ici un test TCP Socket.

5. Indiquez le port sur lequel le conteneur est à l'écoute.
6. Indiquez l'heure, en quelques secondes, après le démarrage du conteneur avant que la sonde puisse être programmée.
7. Indiquez le nombre de secondes pour effectuer la sonde. La valeur par défaut est 10. Cette valeur doit être supérieure au timeoutSeconds.
8. Indiquez le nombre de secondes d'inactivité après lesquelles la sonde est supposée avoir échoué. La valeur par défaut est 1. Cette valeur doit être inférieure à periodSeconds.
9. Facultatif: Créez une sonde de préparation.
10. Indiquez le type de test à effectuer, ici un test HTTP.
11. Indiquez une adresse IP hôte. Lorsque l'hôte n'est pas défini, le PodIP est utilisé.
12. Indiquez HTTP ou HTTPS. Lorsque le schéma n'est pas défini, le schéma HTTP est utilisé.
13. Indiquez le port sur lequel le conteneur est à l'écoute.
14. Facultatif: Créez une sonde de démarrage.
15. Indiquez le type d'essai à effectuer, ici une sonde d'exécution de conteneur.
16. Indiquez les commandes à exécuter sur le conteneur.
17. Indiquez le nombre de fois pour essayer la sonde après une défaillance.
18. Indiquez le nombre de secondes pour effectuer la sonde. La valeur par défaut est 10. Cette valeur doit être supérieure au timeoutSeconds.
19. Indiquez le nombre de secondes d'inactivité après lesquelles la sonde est supposée avoir échoué. La valeur par défaut est 1. Cette valeur doit être inférieure à periodSeconds.

NOTE

Lorsque la valeur initialeDelaySeconds est inférieure à la valeur de la périodeSeconds, la première sonde de préparation se produit à un moment donné entre les deux périodes dues à un problème avec les minuteries.

La valeur TimeoutSeconds doit être inférieure à la valeur periodSeconds.

2. Créer l'objet Pod:

```
$ oc create -f <file-name>.yaml
```

3. Vérifier l'état de la dose de contrôle de santé:

```
$ oc describe pod my-application
```

Exemple de sortie

```
Events:
```

```
  Type    Reason     Age   From                                Message
  ----    ------     ----  ----                                -------
  Normal  Scheduled  9s    default-scheduler                   Successfully assigned openshift-
logging/liveness-exec to ip-10-0-143-40.ec2.internal
  Normal  Pulling    2s    kubelet, ip-10-0-143-40.ec2.internal  pulling image
"registry.k8s.io/liveness"
  Normal  Pulled     1s    kubelet, ip-10-0-143-40.ec2.internal  Successfully pulled image
"registry.k8s.io/liveness"
  Normal  Created    1s    kubelet, ip-10-0-143-40.ec2.internal  Created container
  Normal  Started    1s    kubelet, ip-10-0-143-40.ec2.internal  Started container
```

Ce qui suit est la sortie d'une sonde défaillante qui a redémarré un conteneur:

Échantillon de sortie de vérification de la vie avec un conteneur malsain

```
$ oc describe pod pod1
```

Exemple de sortie

```
....

Events:
  Type     Reason          Age                From                                    Message
  ----     ------          ----               ----                                    -------
  Normal   Scheduled       <unknown>                                                  Successfully
assigned aaa/liveness-http to ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
  Normal   AddedInterface  47s                multus                                  Add eth0
[10.129.2.11/23]
  Normal   Pulled          46s                kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Successfully pulled image "registry.k8s.io/liveness" in 773.406244ms
  Normal   Pulled          28s                kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Successfully pulled image "registry.k8s.io/liveness" in 233.328564ms
  Normal   Created         10s (x3 over 46s)  kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Created container liveness
  Normal   Started         10s (x3 over 46s)  kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Started container liveness
  Warning  Unhealthy       10s (x6 over 34s)  kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-
snzrj  Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing         10s (x2 over 28s)  kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Container liveness failed liveness probe, will be restarted
  Normal   Pulling         10s (x3 over 47s)  kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Pulling image "registry.k8s.io/liveness"
  Normal   Pulled          10s                kubelet, ci-ln-37hz77b-f76d1-wdpjv-worker-b-snzrj
Successfully pulled image "registry.k8s.io/liveness" in 244.116568ms
```

10.3. LA SURVEILLANCE DE LA SANTÉ DES APPLICATIONS EN UTILISANT LA PERSPECTIVE DES DÉVELOPPEURS

La perspective Développeur vous permet d'ajouter trois types de sondes de santé à votre conteneur pour vous assurer que votre application est saine:

- La sonde Readiness permet de vérifier si le conteneur est prêt à traiter les demandes.
- La sonde Liveness permet de vérifier si le conteneur est en cours d'exécution.

- La sonde Startup permet de vérifier si l'application à l'intérieur du conteneur a commencé.

Il est possible d'ajouter des contrôles de santé lors de la création et du déploiement d'une application, ou après avoir déployé une application.

10.4. AJOUT DE CONTRÔLES DE SANTÉ EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

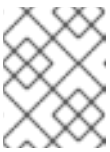
La vue Topology vous permet d'ajouter des contrôles de santé à votre application déployée.

Conditions préalables

- La perspective Développeur est passée à la console Web.
- En utilisant la perspective Développeur, vous avez créé et déployé une application sur Red Hat OpenShift Service sur AWS.

Procédure

1. Dans la vue Topologie, cliquez sur le nœud de l'application pour voir le panneau latéral. Dans le cas où le conteneur n'a pas de contrôle de santé ajouté, une notification Health Checks est affichée avec un lien pour ajouter des contrôles de santé.
2. Dans la notification affichée, cliquez sur le lien Ajouter des vérifications de santé.
3. Alternativement, vous pouvez également cliquer sur la liste Actions et sélectionner Ajouter des bilans de santé. A noter que si le conteneur a déjà des contrôles de santé, vous verrez l'option Modifier les vérifications de santé au lieu de l'option add.
4. Dans le formulaire Ajouter des vérifications de santé, si vous avez déployé plusieurs conteneurs, utilisez la liste des conteneurs pour s'assurer que le conteneur approprié est sélectionné.
5. Cliquez sur les liens de sonde de santé requis pour les ajouter au conteneur. Les données par défaut pour les contrôles de santé sont préremplies. Il est possible d'ajouter les sondes avec les données par défaut ou de personnaliser les valeurs, puis de les ajouter. À titre d'exemple, ajouter une sonde de préparation qui vérifie si votre conteneur est prêt à traiter les demandes:
   a. Cliquez sur Ajouter la sonde de préparation, pour voir un formulaire contenant les paramètres de la sonde.
   b. Cliquez sur la liste Type pour sélectionner le type de demande que vous souhaitez ajouter. Dans ce cas, par exemple, sélectionnez Container Command pour sélectionner la commande qui sera exécutée à l'intérieur du conteneur.
   c. Dans le champ Commande, ajoutez un chat d'argument, de même, vous pouvez ajouter plusieurs arguments pour la vérification, par exemple, ajouter un autre argument /tmp/healthy.
   d. Conserver ou modifier les valeurs par défaut pour les autres paramètres au besoin.



NOTE

La valeur Timeout doit être inférieure à la valeur Période. La valeur par défaut Timeout est 1. La valeur par défaut de la Période est 10.

e. Cliquez sur la coche en bas du formulaire. Le message de Readiness Probe Added est affiché.

6. Cliquez sur Ajouter pour ajouter le bilan de santé. Il est redirigé vers la vue Topology et le conteneur est redémarré.

7. Dans le panneau latéral, vérifiez que les sondes ont été ajoutées en cliquant sur le pod déployé sous la section Pods.

8. Dans la page Détails de Pod, cliquez sur le conteneur listé dans la section Conteneurs.

9. Dans la page Détails du conteneur, vérifiez que la sonde Readiness - Exec Command cat /tmp/healthy a été ajoutée au conteneur.

10.5. ÉDITION DES CONTRÔLES DE SANTÉ EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

La vue Topology permet de modifier les contrôles de santé ajoutés à votre application, de les modifier ou d'ajouter d'autres contrôles de santé.

Conditions préalables

- La perspective Développeur est passée à la console Web.
- En utilisant la perspective Développeur, vous avez créé et déployé une application sur Red Hat OpenShift Service sur AWS.
- Des contrôles de santé ont été ajoutés à votre demande.

Procédure

1. Dans la vue Topologie, faites un clic droit sur votre application et sélectionnez Modifier les vérifications de santé. Alternativement, dans le panneau latéral, cliquez sur la liste déroulante Actions et sélectionnez Modifier les vérifications de santé.

2. Dans la page Modifier les vérifications de santé:

   - Afin de supprimer une sonde de santé précédemment ajoutée, cliquez sur l'icône Supprimer à côté.
   - D'éditer les paramètres d'une sonde existante:

     a. Cliquez sur le lien Modifier la sonde à côté d'une sonde précédemment ajoutée pour voir les paramètres de la sonde.

     b. Modifiez les paramètres au besoin et cliquez sur la coche pour enregistrer vos modifications.

   - Afin d'ajouter une nouvelle sonde de santé, en plus des contrôles de santé existants, cliquez sur les liens de la sonde. À titre d'exemple, ajouter une sonde Liveness qui vérifie si votre conteneur est en cours d'exécution:

     a. Cliquez sur Ajouter une sonde de vie, pour voir un formulaire contenant les paramètres de la sonde.

     b. Éditez les paramètres de la sonde au besoin.

NOTE

La valeur Timeout doit être inférieure à la valeur Période. La valeur par défaut Timeout est 1. La valeur par défaut de la Période est 10.

c. Cliquez sur la coche en bas du formulaire. Le message de Liveness Probe Ajouté est affiché.

3. Cliquez sur Enregistrer pour enregistrer vos modifications et ajouter les sondes supplémentaires à votre conteneur. Il est redirigé vers la vue Topology.

4. Dans le panneau latéral, vérifiez que les sondes ont été ajoutées en cliquant sur le pod déployé sous la section Pods.

5. Dans la page Détails de Pod, cliquez sur le conteneur listé dans la section Conteneurs.

6. Dans la page Détails du conteneur, vérifiez que la sonde Liveness - HTTP Get 10.129.4.65:8080/ a été ajoutée au conteneur, en plus des sondes existantes antérieures.

10.6. CONTRÔLE DES ÉCHECS DES CONTRÔLES DE SANTÉ À L'AIDE DE LA PERSPECTIVE DÉVELOPPEUR

En cas d'échec d'un contrôle de santé d'application, vous pouvez utiliser la vue Topology pour surveiller ces violations du bilan de santé.

Conditions préalables

- La perspective Développeur est passée à la console Web.
- En utilisant la perspective Développeur, vous avez créé et déployé une application sur Red Hat OpenShift Service sur AWS.
- Des contrôles de santé ont été ajoutés à votre demande.

Procédure

1. Dans la vue Topologie, cliquez sur le nœud de l'application pour voir le panneau latéral.
2. Cliquez sur l'onglet Observer pour voir les échecs du contrôle de santé dans la section Événements (Warning).
3. Cliquez sur la flèche vers le bas des événements attenants (Warning) pour voir les détails de l'échec du bilan de santé.

Ressources supplémentaires

- Des détails sur l'ajout de contrôles de santé lors de la création et du déploiement d'une application, consultez Options avancées dans les applications Créer des applications à l'aide de la section perspective Développeur.

CHAPITRE 11. ÉDITION DES APPLICATIONS

Il est possible d'éditer la configuration et le code source de l'application que vous créez à l'aide de la vue Topology.

11.1. CONDITIONS PRÉALABLES

- En utilisant la perspective Développeur, vous avez créé et déployé une application sur Red Hat OpenShift Service sur AWS.
- Connectez-vous à la console Web et passez à la perspective Développeur.

11.2. ÉDITER LE CODE SOURCE D'UNE APPLICATION EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

Dans la perspective Développeur, vous pouvez utiliser la vue Topology pour modifier le code source de votre application.

Procédure

- Dans la vue Topologie, cliquez sur l'icône Modifier le code source, affichée en bas à droite de l'application déployée, pour accéder à votre code source et le modifier.

NOTE

Cette fonctionnalité n'est disponible que lorsque vous créez des applications à l'aide des options From Git, From Catalog et From Dockerfile.

11.3. ÉDITER LA CONFIGURATION DE L'APPLICATION EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

Dans la perspective Développeur, vous pouvez utiliser la vue Topology pour modifier la configuration de votre application.

NOTE

Actuellement, seules les configurations d'applications créées à l'aide des options From Git, Container Image, From Catalog ou From Dockerfile dans le flux de travail Ajouter de la perspective Développeur peuvent être modifiées. Les configurations d'applications créées à l'aide de l'option CLI ou YAML à partir du flux de travail Ajouter ne peuvent pas être modifiées.

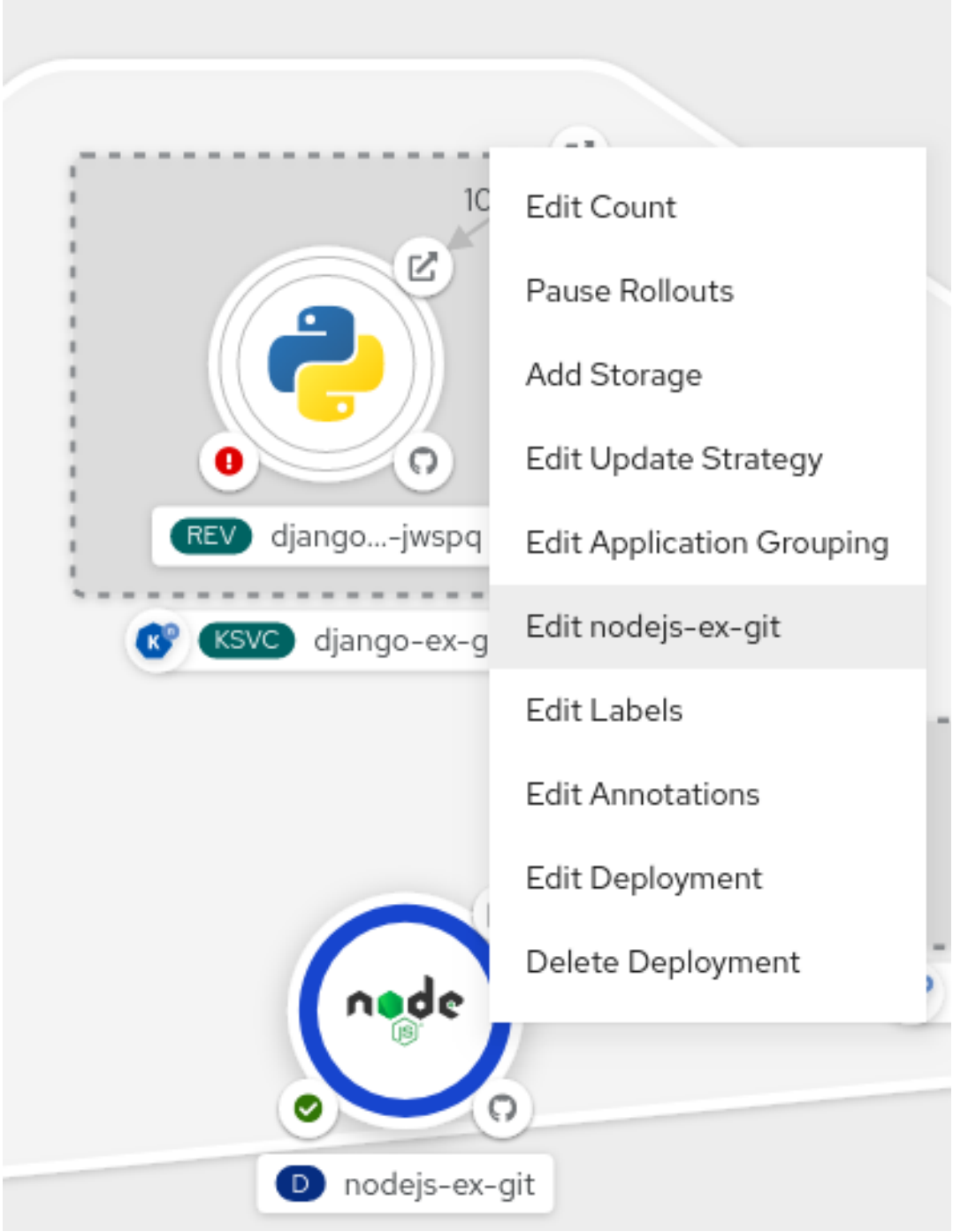
Conditions préalables

Assurez-vous que vous avez créé une application à l'aide des options From Git, Container Image, From Catalog ou From Dockerfile dans le workflow Ajouter.

Procédure

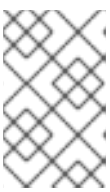
1. Après avoir créé une application et qu'elle est affichée dans la vue Topology, faites un clic droit sur l'application pour voir les options d'édition disponibles.

Figure 11.1. Édition de l'application



2. Cliquez sur Modifier le nom de l'application pour voir le flux de travail Ajouter que vous avez utilisé pour créer l'application. Le formulaire est prérempli avec les valeurs que vous avez ajoutées lors de la création de l'application.

3. Éditez les valeurs nécessaires pour l'application.

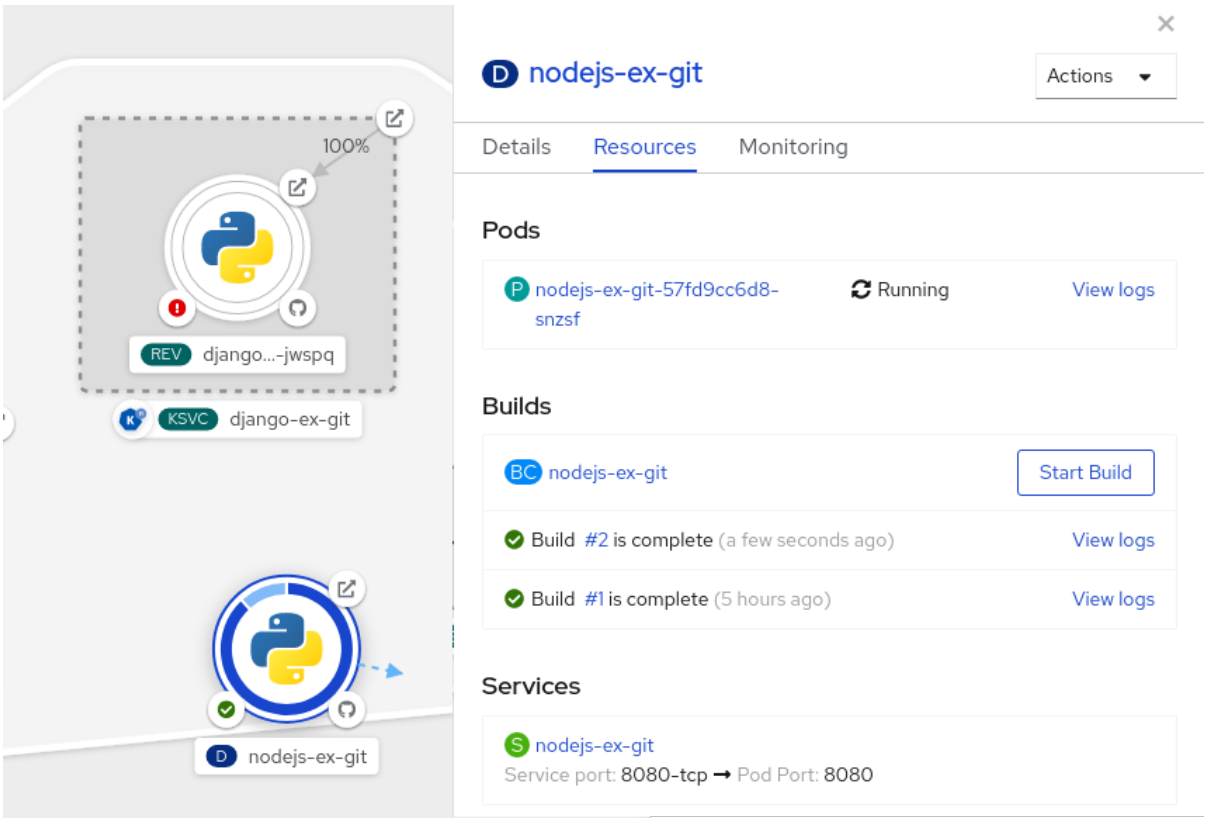


NOTE

Dans la section Options avancées, vous ne pouvez pas modifier le champ Nom dans la section Généralité, les pipelines CI/CD ou Créer un itinéraire vers le champ d'application.

4. Cliquez sur Enregistrer pour redémarrer la construction et déployer une nouvelle image.

Figure 11.2. Éditer et redéployer l'application



CHAPITRE 12. LE TRAVAIL AVEC LES QUOTAS

Le quota de ressources, défini par un objet ResourceQuota, fournit des contraintes qui limitent la consommation globale de ressources par projet. Il peut limiter la quantité d'objets qui peuvent être créés dans un projet par type, ainsi que la quantité totale de ressources de calcul et de stockage qui peuvent être consommés par les ressources de ce projet.

Le nombre de quotas d'objets place un quota défini sur tous les types de ressources standard en espace de noms. Lors de l'utilisation d'un quota de ressources, un objet est facturé sur le quota s'il existe dans le stockage du serveur. Ces types de quotas sont utiles pour protéger contre l'épuisement des ressources de stockage.

Ce guide décrit comment fonctionnent les quotas de ressources et comment les développeurs peuvent travailler et les visualiser.

12.1. AFFICHAGE D'UN QUOTA

Les statistiques d'utilisation liées à toute limite dure définie dans le quota d'un projet peuvent être affichées en naviguant dans la console Web vers la page Quota du projet.

Il est également possible d'utiliser le CLI pour afficher les détails des quotas.

Procédure

1. Demandez la liste des quotas définis dans le projet. À titre d'exemple, pour un projet appelé démoproject:

```
$ oc get quota -n demoproject
```

Exemple de sortie

```
NAME                         AGE    REQUEST
LIMIT
besteffort                   4s     pods: 1/2
compute-resources-time-bound   10m    pods: 0/2
limits.cpu: 0/1, limits.memory: 0/1Gi
core-object-counts           109s   configmaps: 2/10, persistentvolumeclaims: 1/4,
replicationcontrollers: 1/20, secrets: 9/10, services: 2/10
```

2. Décrivez le quota qui vous intéresse, par exemple le quota des comptes d'objets de base:

```
$ oc describe quota core-object-counts -n demoproject
```

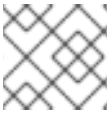
Exemple de sortie

```
Name:   core-object-counts
Namespace:  demoproject
Resource  Used Hard
--------  ---- ----
configmaps  3 10
persistentvolumeclaims 0 4
```

```
replicationcontrollers 3 20
secrets   9 10
services  2 10
```

12.2. LES RESSOURCES GÉRÉES PAR LES QUOTAS

Ce qui suit décrit l'ensemble des ressources de calcul et des types d'objets qui peuvent être gérés par un quota.



NOTE

Le pod est dans un état terminal si Status.phase dans (Failed, Succeed) est vrai.

Tableau 12.1. Calcul des ressources gérées par quota

Le nom de la ressource	Description
CPU	La somme des requêtes CPU dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. CPU et request.cpu sont la même valeur et peuvent être utilisés de manière interchangeable.
la mémoire	La somme des requêtes de mémoire dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. la mémoire et les requêtes.memory sont la même valeur et peuvent être utilisées de manière interchangeable.
demandes.cpu	La somme des requêtes CPU dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. CPU et request.cpu sont la même valeur et peuvent être utilisés de manière interchangeable.
demandes.memory	La somme des requêtes de mémoire dans tous les pods dans un état non terminal ne peut pas dépasser cette valeur. la mémoire et les requêtes.memory sont la même valeur et peuvent être utilisées de manière interchangeable.
limites.cpu	La somme des limites CPU sur toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur.
Limites.memory	La somme des limites de mémoire à travers tous les pods dans un état non terminal ne peut pas dépasser cette valeur.

Tableau 12.2. Les ressources de stockage gérées par quota

Le nom de la ressource	Description
demandes.stockage	La somme des demandes de stockage sur toutes les revendications de volume persistant dans n'importe quel état ne peut pas dépasser cette valeur.
revendications persistantes du volume	Le nombre total de revendications de volume persistant qui peuvent exister dans le projet.

Le nom de la ressource	Description
<storage-class-name>.storageclass.storage.k8s.io/requests.storage	La somme des demandes de stockage pour toutes les revendications de volume persistantes dans n'importe quel état ayant une classe de stockage correspondante ne peut pas dépasser cette valeur.
<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims	Le nombre total de revendications de volume persistant avec une classe de stockage correspondante qui peut exister dans le projet.
le stockage éphémère	La somme des demandes de stockage éphémères locales dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur. le stockage éphémère et les requêtes.ephemeral-stockage sont la même valeur et peuvent être utilisés de manière interchangeable.
demandes.ephemeral-stockage	La somme des demandes de stockage éphémères dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur. le stockage éphémère et les requêtes.ephemeral-stockage sont la même valeur et peuvent être utilisés de manière interchangeable.
limites.ephemeral-stockage	La somme des limites de stockage éphémères dans toutes les gousses dans un état non terminal ne peut pas dépasser cette valeur.

Tableau 12.3. Comptages d'objets gérés par quota

Le nom de la ressource	Description
les gousses	Le nombre total de pods dans un état non terminal qui peut exister dans le projet.
contrôleurs de réplication	Le nombre total de RéplicationControllers qui peuvent exister dans le projet.
quotas de ressources	Le nombre total de quotas de ressources pouvant exister dans le projet.
les services	Le nombre total de services pouvant exister dans le projet.
balanceurs de charge Services.	Le nombre total de services de type LoadBalancer qui peuvent exister dans le projet.
les services.nodeports	Le nombre total de services de type NodePort pouvant exister dans le projet.
les secrets	Le nombre total de secrets qui peuvent exister dans le projet.
ConfigMaps	Le nombre total d'objets ConfigMap pouvant exister dans le projet.

Le nom de la ressource	Description
revendications persistantes du volume	Le nombre total de revendications de volume persistant qui peuvent exister dans le projet.
informations sur OpenShift.io/imagestreams	Le nombre total de flux d'images pouvant exister dans le projet.

12.3. CHAMP D'APPLICATION DES QUOTAS

Chaque quota peut avoir un ensemble de portées associées. Le quota ne mesure l'utilisation d'une ressource que s'il correspond à l'intersection des périmètres énumérés.

L'ajout d'une portée à un quota limite l'ensemble des ressources auxquelles ce quota peut s'appliquer. La spécification d'une ressource en dehors de l'ensemble autorisé entraîne une erreur de validation.

Champ d'application	Description
BestEffort	Assortir des gousses qui ont le meilleur effort de qualité de service pour le cpu ou la mémoire.
À propos de NotBestEffort	Assortir des gousses qui n'ont pas le meilleur effort de qualité de service pour le cpu et la mémoire.

La portée de BestEffort limite un quota à la limitation des ressources suivantes:

- **les gousses**

La portée NotBestEffort limite un quota au suivi des ressources suivantes:

- **les gousses**
- **la mémoire**
- **demandes.memory**
- **Limites.memory**
- **CPU**
- **demandes.cpu**
- **limites.cpu**

12.4. APPLICATION DES QUOTAS

Après la création d'un quota de ressources pour un projet, le projet restreint la possibilité de créer de nouvelles ressources qui pourraient violer une contrainte de quota jusqu'à ce qu'il ait calculé des statistiques d'utilisation mises à jour.

Après la création d'un quota et la mise à jour des statistiques d'utilisation, le projet accepte la création de nouveaux contenus. Lorsque vous créez ou modifiez des ressources, votre utilisation de quota est incrémentée immédiatement à la demande de créer ou de modifier la ressource.

Lorsque vous supprimez une ressource, votre utilisation des quotas est décrémentée lors de la prochaine recalcul complet des statistiques des quotas pour le projet. Le temps configurable détermine combien de temps il faut pour réduire les statistiques d'utilisation des quotas à la valeur actuelle du système observé.

Lorsque les modifications de projet dépassent une limite d'utilisation de quota, le serveur refuse l'action, et un message d'erreur approprié est renvoyé à l'utilisateur expliquant la contrainte de quota violée, et quelles sont les statistiques d'utilisation actuellement observées dans le système.

12.5. DEMANDES PAR RAPPORT AUX LIMITES

Lors de l'attribution des ressources de calcul, chaque conteneur peut spécifier une requête et une valeur limite chacune pour le stockage CPU, mémoire et éphémère. Les quotas peuvent restreindre l'une de ces valeurs.

Lorsque le quota a une valeur spécifiée pour request.cpu ou request.memory, il exige que chaque conteneur entrant fasse une demande explicite pour ces ressources. Lorsque le quota a une valeur spécifiée pour limits.cpu ou limits.memory, il exige que chaque conteneur entrant spécifie une limite explicite pour ces ressources.

CHAPITRE 13. ÉLAGAGE DES OBJETS POUR RÉCUPÉRER DES RESSOURCES

Au fil du temps, les objets API créés dans Red Hat OpenShift Service sur AWS peuvent s'accumuler dans le stockage de données etcd du cluster grâce à des opérations utilisateur normales, telles que la création et le déploiement d'applications.

L'utilisateur ayant le rôle d'administrateur dédié peut prélever périodiquement les anciennes versions d'objets du cluster qui ne sont plus nécessaires. En tailleant des images, par exemple, vous pouvez supprimer des images et des couches plus anciennes qui ne sont plus utilisées, mais qui prennent encore de l'espace disque.

13.1. LES OPÉRATIONS D'ÉLAGAGE DE BASE

Le CLI regroupe les opérations de taille sous une commande parente commune:

```
$ oc adm prune <object_type> <options>
```

Ceci spécifie:

- Le <object_type> pour effectuer l'action sur, tels que des groupes, des builds, des déploiements ou des images.
- Les <options> supportées pour tailler ce type d'objet.

13.2. GROUPES D'ÉLAGAGE

Afin de prélever des enregistrements de groupes à partir d'un fournisseur externe, les administrateurs peuvent exécuter la commande suivante:

```
$ oc adm prune groups \
    --sync-config=path/to/sync/config [<options>]
```

Tableau 13.1. drapeaux OC adm prunes

Les options	Description
--confirmer	Indiquer que l'élagage devrait se produire, au lieu d'effectuer une course à sec.
--liste noire	Chemin d'accès au fichier de liste noire du groupe.
--liste blanche	Chemin d'accès au fichier de liste blanche du groupe.
--sync-config	Chemin d'accès au fichier de configuration de synchronisation.

Procédure

1. Afin de voir les groupes que la commande prune supprime, exécutez la commande suivante:

   ```
   $ oc adm prune groups --sync-config=ldap-sync-config.yaml
   ```

2. Afin d'effectuer l'opération de pruneaux, ajoutez le drapeau --confirmer:

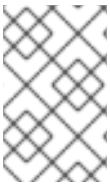
```
$ oc adm prune groups --sync-config=ldap-sync-config.yaml --confirm
```

13.3. ÉLAGAGE DES RESSOURCES DE DÉPLOIEMENT

En raison de l'âge et du statut, vous pouvez tailler les ressources associées aux déploiements qui ne sont plus requis par le système.

Les contrôleurs de réplication de commande suivants sont associés aux objets DeploymentConfig:

```
$ oc adm prune deployments [<options>]
```



NOTE

Afin de tailler également les ensembles de répliques associés aux objets de déploiement, utilisez le drapeau --replica-sets. Ce drapeau est actuellement une fonctionnalité d'aperçu technologique.

Tableau 13.2. drapeaux OC adm prunes

L'option	Description
--confirmer	Indiquer que l'élagage devrait se produire, au lieu d'effectuer une course à sec.
--maintenez-complete=<N>	Dans l'objet DeploymentConfig, gardez les derniers contrôleurs de réplication N qui ont un statut de comptage complet et réplique de zéro. La valeur par défaut est 5.
--maintenir l'échec =<N>	Dans l'objet DeploymentConfig, gardez les derniers contrôleurs de réplication N ayant un statut d'échec et de nombre de répliques de zéro. La valeur par défaut est 1.
--maintenir-jenger-que=<durée>	Aucun contrôleur de réplication n'est plus jeune que <duration> par rapport à l'heure actuelle. Les unités de mesure valides comprennent les nanosecondes (ns), les microsecondes (nous), les millisecondes (ms), les secondes (s), les minutes (m) et les heures (h). La valeur par défaut est de 60m.
--les orphelins	Eliminez tous les contrôleurs de réplication qui n'ont plus d'objet DeploymentConfig, ont le statut de Complet ou d'échec, et ont un compte de réplique de zéro.

Procédure

1. Afin de voir ce qu'une opération d'élagage supprimerait, exécutez la commande suivante:

```
$ oc adm prune deployments --orphans --keep-complete=5 --keep-failed=1 \
    --keep-younger-than=60m
```

2. Afin d'effectuer l'opération de pruneaux, ajoutez le drapeau --confirmer:

```
$ oc adm prune deployments --orphans --keep-complete=5 --keep-failed=1 \
    --keep-younger-than=60m --confirm
```

13.4. CONSTRUCTIONS D'ÉLAGAGE

En raison de l'âge et de l'état, les administrateurs peuvent exécuter la commande suivante pour pruner des builds qui ne sont plus requis par le système en raison de leur âge et de leur statut:

```
$ oc adm prune builds [<options>]
```

Tableau 13.3. le pruneur OC adm construit des drapeaux

L'option	Description
--confirmer	Indiquer que l'élagage devrait se produire, au lieu d'effectuer une course à sec.
--les orphelins	Eliminer toutes les versions dont la configuration de construction n'existe plus, l'état est complet, échoué, erreur ou annulé.
--maintenez-complete=<N>	Dans chaque configuration de construction, conservez les dernières versions N dont l'état est complet. La valeur par défaut est 5.
--maintenir l'échec =<N>	Dans chaque configuration de build, conservez les dernières versions N dont le statut est échoué, erreur ou annulé. La valeur par défaut est 1.
--maintenir-jenger-que=<durée>	Il ne faut pas tailler un objet plus jeune que <duration> par rapport à l'heure actuelle. La valeur par défaut est de 60m.

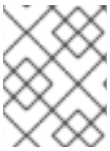
Procédure

1. Afin de voir ce qu'une opération d'élagage supprimerait, exécutez la commande suivante:

```
$ oc adm prune builds --orphans --keep-complete=5 --keep-failed=1 \
    --keep-younger-than=60m
```

2. Afin d'effectuer l'opération de pruneaux, ajoutez le drapeau --confirmer:

```
$ oc adm prune builds --orphans --keep-complete=5 --keep-failed=1 \
    --keep-younger-than=60m --confirm
```



NOTE

Les développeurs peuvent activer l'élagage automatique de la construction en modifiant leur configuration de construction.

13.5. ÉLAGAGE AUTOMATIQUE DES IMAGES

Les images du registre d'images OpenShift qui ne sont plus requises par le système en raison de l'âge, de l'état ou des limites supérieures sont automatiquement taillées. Les administrateurs de clusters peuvent configurer la ressource personnalisée de Pruning ou la suspendre.

Conditions préalables

- Grâce à un compte doté d'autorisations d'administration dédiées, vous avez accès à un service Red Hat OpenShift sur AWS.
- Installez le CLI oc.

Procédure

- Assurez-vous que l'objet nommé imagepruners.imageregistry.operator.openshift.io/cluster contient les champs de spécifications et d'état suivants:

```
spec:
  schedule: 0 0 * * * (1)
  suspend: false (2)
  keepTagRevisions: 3 (3)
  keepYoungerThanDuration: 60m (4)
  keepYoungerThan: 3600000000000 (5)
  resources: {} (6)
  affinity: {} (7)
  nodeSelector: {} (8)
  tolerations: [] (9)
  successfulJobsHistoryLimit: 3 (10)
  failedJobsHistoryLimit: 3 (11)
status:
  observedGeneration: 2 (12)
  conditions: (13)
  - type: Available
    status: "True"
    lastTransitionTime: 2019-10-09T03:13:45
    reason: Ready
    message: "Periodic image pruner has been created."
  - type: Scheduled
    status: "True"
    lastTransitionTime: 2019-10-09T03:13:45
    reason: Scheduled
    message: "Image pruner job has been scheduled."
  - type: Failed
    staus: "False"
    lastTransitionTime: 2019-10-09T03:13:45
    reason: Succeeded
    message: "Most recent image pruning job succeeded."
```

(1) horaire: CronJob formaté horaire. Il s'agit d'un champ optionnel, par défaut est quotidien à minuit.

(2) suspendre: Si défini à true, l'élagage en cours d'exécution CronJob est suspendu. Il s'agit d'un champ optionnel, par défaut est false. La valeur initiale des nouveaux clusters est fausse.

3. KeepTagRevisions: Le nombre de révisions par balise à conserver. Il s'agit d'un champ optionnel, par défaut 3. La valeur initiale est 3.

4. KeepYoungerThanDuration: Retenez des images plus jeunes que cette durée. Il s'agit d'un champ facultatif. Lorsqu'une valeur n'est pas spécifiée, KeepYoungerThan ou la valeur par défaut 60m (60 minutes) sont utilisées.

5. KeepYoungerThan: Déprécié. Le même que KeepYoungerThanDuration, mais la durée est spécifiée comme un entier en nanosecondes. Il s'agit d'un champ facultatif. Lorsque keepYoungerThanDuration est défini, ce champ est ignoré.

6. les ressources : Demandes et limites de ressources de pod standard. Il s'agit d'un champ facultatif.

7. affinité: affinité standard de la pod. Il s'agit d'un champ facultatif.

8. nodeSelector: Sélecteur de nœud de pod standard. Il s'agit d'un champ facultatif.

9. les tolérances: tolérances standard des pods. Il s'agit d'un champ facultatif.

10. avec succèsJobsHistoryLimit: Le nombre maximum d'emplois réussis à conserver. Doit être >= 1 pour s'assurer que les métriques sont rapportées. Il s'agit d'un champ optionnel, par défaut 3. La valeur initiale est 3.

11. failJobsHistoryLimit: Le nombre maximum d'emplois échoués à conserver. Doit être >= 1 pour s'assurer que les métriques sont rapportées. Il s'agit d'un champ optionnel, par défaut 3. La valeur initiale est 3.

12. génération observée : la génération observée par l'opérateur.

13. conditions: Les objets de condition standard avec les types suivants:

    - Disponible : Indique si le travail d'élagage a été créé. Les raisons peuvent être prêtes ou erronées.
    - Indique si le prochain travail d'élagage a été programmé. Les raisons peuvent être programmées, suspendues ou erreur.
    - Échec : Indique si le dernier travail d'élagage a échoué.

IMPORTANT

Le comportement de l'opérateur de registre d'images pour gérer le tailleur est orthogonal à l'état de gestion spécifié sur l'objet ClusterOperator de l'opérateur de registre d'images. Dans le cas où l'opérateur de registre d'images n'est pas dans l'état géré, le tailleur d'image peut toujours être configuré et géré par la ressource personnalisée de Pruning.

Cependant, l'état de gestion de l'opérateur de registre d'images modifie le comportement de la tâche de tailleur d'image déployée:

- Géré: le drapeau --prune-registry pour l'élageur d'image est défini sur true.
- Supprimé: l'indicateur --prune-registry pour l'élageur d'image est défini sur false, ce qui signifie qu'il n'élimine que les métadonnées d'image en etcd.

13.6. EMPLOIS POUR TAILLER CRON

Les travaux Cron peuvent effectuer l'élagage d'emplois réussis, mais pourraient ne pas gérer correctement les tâches échouées. L'administrateur du cluster doit donc effectuer un nettoyage régulier des tâches manuellement. Ils devraient également restreindre l'accès aux emplois cron à un petit groupe d'utilisateurs de confiance et fixer un quota approprié pour empêcher l'emploi cron de créer trop d'emplois et de pods.

Ressources supplémentaires

- Quotas de ressources pour plusieurs projets

CHAPITRE 14. APPLICATIONS AU RALENTI

Les administrateurs de cluster peuvent inactiver les applications pour réduire la consommation de ressources. Ceci est utile lorsque le cluster est déployé sur un cloud public où le coût est lié à la consommation de ressources.

En l'absence de ressources évolutives, Red Hat OpenShift Service sur AWS les découvre et les ralentit en évoluant leurs répliques à 0. La prochaine fois que le trafic réseau est dirigé vers les ressources, les ressources sont désactivées par la mise à l'échelle des répliques, et le fonctionnement normal se poursuit.

Les applications sont faites de services, ainsi que d'autres ressources évolutives, telles que les configurations de déploiement. L'action du ralenti d'une application implique le ralenti de toutes les ressources associées.

14.1. APPLICATIONS AU RALENTI

Le ralenti d'une application consiste à trouver les ressources évolutives (configurations de déploiement, contrôleurs de réplication et autres) associées à un service. Au ralenti, une application trouve le service et le marque comme inactif, réduisant les ressources à zéro répliques.

La commande oc oisle permet d'activer un seul service ou d'utiliser l'option --resource-names-file pour activer plusieurs services.

14.1.1. Au ralenti d'un seul service

Procédure

1. Afin d'inactiver un seul service, exécutez:

```
$ oc idle <service>
```

14.1.2. Des services multiples au ralenti

Le ralenti de plusieurs services est utile si une application couvre un ensemble de services au sein d'un projet, ou lorsque vous ralentissez plusieurs services en conjonction avec un script pour inactiver plusieurs applications en vrac dans le même projet.

Procédure

1. Créer un fichier contenant une liste des services, chacun sur sa propre ligne.
2. Inactivez les services en utilisant l'option --resource-names-file:

```
$ oc idle --resource-names-file <filename>
```

NOTE

La commande oisive est limitée à un seul projet. Dans le cas d'applications au ralenti à travers un cluster, exécutez la commande oisive pour chaque projet individuellement.

14.2. APPLICATIONS D'UNIDLING

Les services d'application deviennent de nouveau actifs lorsqu'ils reçoivent du trafic réseau et sont réduits à la hausse de leur état précédent. Cela inclut à la fois le trafic vers les services et le trafic passant par les routes.

Les applications peuvent également être désactivées manuellement en augmentant les ressources.

Procédure

1. Afin de mettre à l'échelle un DeploymentConfig, exécutez:

```
$ oc scale --replicas=1 dc <dc_name>
```

NOTE

Le déroulage automatique par un routeur n'est actuellement pris en charge que par le routeur HAProxy par défaut.

CHAPITRE 15. LA SUPPRESSION DES APPLICATIONS

Il est possible de supprimer les applications créées dans votre projet.

15.1. LA SUPPRESSION DES APPLICATIONS EN UTILISANT LA PERSPECTIVE DÉVELOPPEUR

Dans la perspective Développeur, vous pouvez supprimer une application et tous ses composants associés en utilisant la vue Topology:

1. Cliquez sur l'application que vous souhaitez supprimer pour voir le panneau latéral avec les détails des ressources de l'application.

2. Cliquez sur le menu déroulant Actions affiché en haut à droite du panneau, puis sélectionnez Supprimer l'application pour afficher une boîte de dialogue de confirmation.

3. Entrez le nom de l'application et cliquez sur Supprimer pour la supprimer.

Cliquez avec le bouton droit de la souris sur l'application que vous souhaitez supprimer et cliquez sur Supprimer l'application pour la supprimer.

CHAPITRE 16. EN UTILISANT LE RED HAT MARKETPLACE

Le Red Hat Marketplace est un marché de cloud ouvert qui facilite la découverte et l'accès à des logiciels certifiés pour les environnements basés sur des conteneurs qui fonctionnent sur les nuages publics et sur site.

16.1. CARACTÉRISTIQUES RED HAT MARKETPLACE

Les administrateurs de clusters peuvent utiliser le Red Hat Marketplace pour gérer les logiciels sur Red Hat OpenShift Service sur AWS, donner aux développeurs un accès en libre-service pour déployer des instances d'application et corréler l'utilisation des applications par rapport à un quota.

16.1.1. Connectez Red Hat OpenShift Service sur les clusters AWS à la Marketplace

Les administrateurs de clusters peuvent installer un ensemble commun d'applications sur Red Hat OpenShift Service sur les clusters AWS qui se connectent au Marketplace. Ils peuvent également utiliser le Marketplace pour suivre l'utilisation des clusters par rapport aux abonnements ou aux quotas. Les utilisateurs qu'ils ajoutent en utilisant le Marketplace ont suivi l'utilisation de leur produit et facturé à leur organisation.

Au cours du processus de connexion de cluster, un opérateur Marketplace est installé qui met à jour le secret du registre des images, gère le catalogue et signale l'utilisation de l'application.

16.1.2. Installer des applications

Les administrateurs de clusters peuvent installer des applications Marketplace depuis OperatorHub dans Red Hat OpenShift Service sur AWS, ou à partir de l'application Web Marketplace.

Depuis la console Web, vous pouvez accéder aux applications installées en cliquant sur Opérateurs > Opérateurs installés.

16.1.3. Déployer des applications sous différentes perspectives

Les applications Marketplace peuvent être déployées du point de vue de l'administrateur et du développeur de la console Web.

La perspective des développeurs
Les développeurs peuvent accéder aux capacités nouvellement installées en utilisant la perspective Développeur.

Après l'installation d'un opérateur de base de données, un développeur peut créer une instance à partir du catalogue dans son projet. L'utilisation de la base de données est agrégée et signalée à l'administrateur du cluster.

Cette perspective n'inclut pas l'installation de l'opérateur et le suivi de l'utilisation des applications.

La perspective de l'administrateur
Les administrateurs de clusters peuvent accéder aux informations d'installation et d'utilisation des applications de l'opérateur du point de vue de l'administrateur.

Ils peuvent également lancer des instances d'application en parcourant les définitions de ressources personnalisées (CRD) dans la liste des opérateurs installés.