OpenShift Container Platform 4.12

レジストリー

OpenShift Container Platform のレジストリーの設定

Last Updated: 2024-09-28

OpenShift Container Platform 4.12 レジストリー

OpenShift Container Platform のレジストリーの設定

法律上の通知

概要

このドキュメントでは、OpenShift Container Platform の内部レジストリーを設定し、管理する方法を説明します。また、OpenShift Container Platform に関連付けられたレジストリーの概要も提供します。

目次

第1章 OPENSHIFT イメージレジストリーの概要

OpenShift Container Platform はイメージをソースコードからビルドし、そのイメージをデプロイしてライフサイクルを管理できます。OpenShift Container Platform は、OpenShift Container Platform 環境にデプロイできる内部の統合コンテナーイメージレジストリーを提供しており、ここからイメージをローカルで管理できます。この概要には、OpenShift イメージレジストリーに重点を置いた、OpenShift Container Platform で一般的に使用されるレジストリーの参照情報およびリンクが含まれます。

1.1. OPENSHIFT イメージレジストリーの共通用語集

この用語集では、レジストリーコンテンツで使用される一般的な用語を定義しています。

コンテナー

ソフトウェアとそのすべての依存関係を構成する軽量で実行可能なイメージ。コンテナーはオペレーティングシステムを仮想化するため、データセンター、パブリッククラウドまたはプライベートクラウド、ローカルホストでコンテナーを実行できます。

Image Registry Operator

Image Registry Operator は **openshift-image-registry** namespace で実行し、その場所のレジストリーインスタンスを管理します。

イメージリポジトリー

イメージリポジトリーは、関連するコンテナーイメージおよびイメージを特定するタグのコレクションです。

ミラーレジストリー

ミラーレジストリーは、OpenShift Container Platform イメージのミラーを保持するレジストリーです。

namespace

namespace は、単一クラスター内のリソースのグループを分離します。

pod

Pod は、Kubernetes における最小の論理単位です。Pod は、ワーカーノードで実行される 1 つ以上のコンテナーで構成されます。

プライベートレジストリー

レジストリーは、コンテナーイメージレジストリー API を実装するサーバーです。プライベートレジストリーは、ユーザーがそのコンテンツにアクセスするのに認証が必要なレジストリーです。

公開レジストリー

レジストリーは、コンテナーイメージレジストリー API を実装するサーバーです。公開レジストリーは、その内容を公に提供するレジストリーです。

Quay.io

Red Hat により提供および維持されるパブリックな Red Hat Quay Container Registry インスタンスであり、ほとんどのコンテナーイメージと Operator を OpenShift Container Platform クラスターに提供します。

OpenShift イメージレジストリー

OpenShift イメージレジストリーは、イメージを管理するために OpenShift Container Platform により提供されるレジストリーです。

レジストリー認証

プライベートイメージリポジトリーとの間でイメージをプッシュおよびプルするには、レジストリーで認証情報を使用してユーザーを認証する必要があります。

ルート

サービスを公開して、OpenShift Container Platform インスタンス外のユーザーおよびアプリケーションから Pod へのネットワークアクセスを許可します。

スケールダウン

レプリカの数を減らすことを意味します。

スケールアップ

レプリカの数を増やすことを意味します。

サービス

サービスは、一連の Pod で実行中のアプリケーションを公開します。

1.2. 統合 OPENSHIFT イメージレジストリー

OpenShift Container Platform は、クラスター上の標準ワークロードとして実行される組み込みコンテナーイメージレジストリーを提供します。このレジストリーはインフラストラクチャー Operator によって設定され、管理されます。また、追加設定なしで使用できる、ワークロードを実行するイメージの管理を目的とするソリューションを提供し、既存のクラスターインフラストラクチャーの上部で実行されます。このレジストリーは、他のクラスターワークロードのようにスケールアップまたはスケールダウンでき、特定のインフラストラクチャーのプロビジョニングを必要としません。さらに、クラスターのユーザー認証および認可システムに統合されるため、イメージを作成および取得するためのアクセスは、イメージリソースでユーザーのパーミッションを定義することで制御できます。

通常、レジストリーはクラスター上にビルドされたイメージの公開ターゲットとして、またクラスター上で実行されるワークロードのイメージのソースとして使用されます。新規イメージがレジストリーにプッシュされると、その旨がクラスターに通知されます。他のコンポーネントは、更新されたイメージに対して応答したり、それを使用したりできます。

イメージデータは 2 つの場所に保存されます。実際のイメージデータは、クラウドストレージまたはファイルシステムボリュームなどの設定可能なストレージの場所に格納されます。標準のクラスター API によって公開され、アクセス制御の実行に使用されるイメージメタデータは、標準的な API リソース、特にイメージおよびイメージストリームとして保存されます。

関連情報

- OpenShift Container Platform の Image Registry Operator

1.3. サードパーティーレジストリー

OpenShift Container Platform はサードパーティーレジストリーからのイメージを使用してコンテナーを作成できますが、これらのレジストリーは統合 OpenShift イメージレジストリーと同じイメージ通知をサポートする可能性はほぼありません。このため、OpenShift Container Platform はイメージストリームの作成時にリモートレジストリーからタグをフェッチします。フェッチされたタグを更新するには、**oc import-image <stream>** を実行します。新規イメージが検出されると、記述したビルドとデプロイメントの応答が生じます。

1.3.1. 認証

OpenShift Container Platform はユーザーが指定する認証情報を使用してプライベートイメージリポジトリーにアクセスするためにレジストリーと通信できます。これにより、OpenShift Container Platform はイメージのプッシュ/プルをプライベートリポジトリーへ/から実行できます。

1.3.1.1. Podman を使用したレジストリー認証

一部のコンテナーイメージレジストリーではアクセス認証が必要です。Podman は、コンテナーおよびコンテナーイメージを管理し、イメージレジストリーと対話するためのオープンソースツールです。Podman を使用して、認証情報の認証、レジストリーイメージのプル、ローカルファイルシステムへのローカルイメージの保存を行なえます。以下は、Podman でレジストリーを認証する一般的な例です。

手順

1. Red Hat Ecosystem Catalog を使用して Red Hat リポジトリーから特定のコンテナーイメージを検索し、必要なイメージを選択します。
2. **Get this image** をクリックして、コンテナーイメージのコマンドを見つけます。
3. 次のコマンドを実行してログインし、ユーザー名とパスワードを入力して認証を受けます。

```
$ podman login registry.redhat.io
 Username:<your_registry_account_username>
 Password:<your_registry_account_password>
```

4. 以下のコマンドを実行してイメージをダウンロードし、ローカルに保存します。

```
$ podman pull registry.redhat.io/<repository_name>
```

1.4. RED HAT QUAY レジストリー

エンタープライズ向けの高品質なコンテナーイメージレジストリーが必要な場合、Red Hat Quay をホストされたサービスとして、また独自のデータセンターやクラウド環境にインストールするソフトウェアとして使用できます。Red Hat Quay の高度な機能には、geo レプリケーション、イメージのスキャニング、およびイメージのロールバック機能が含まれます。

Quay.io サイトにアクセスし、独自のホストされた Quay レジストリーアカウントをセットアップします。その後、Quay チュートリアルに従って Quay レジストリーにログインし、イメージの管理を開始します。

Red Hat Quay レジストリーへのアクセスは、任意のリモートコンテナーイメージレジストリーと同様に OpenShift Container Platform から実行できます。

関連情報

- Red Hat Quay 製品ドキュメント

1.5. 認証が有効な RED HAT レジストリー

Red Hat Ecosystem Catalog のコンテナーイメージのセクションで利用可能なすべてのコンテナーイメージはイメージレジストリーの **registry.redhat.io** でホストされます。

レジストリー **registry.redhat.io** では、イメージおよび OpenShift Container Platform でホストされるコンテンツへのアクセスに認証が必要です。新規レジストリーへの移行後も、既存レジストリーはしばらく利用可能になります。

注記

OpenShift Container Platform はイメージを **registry.redhat.io** からプルするため、これを使用できるようにクラスターを設定する必要があります。

新規レジストリーは、以下の方法を使用して認証に標準の OAuth メカニズムを使用します。

- **認証トークン**。管理者によって生成されるこれらのトークンは、コンテナーイメージレジストリーに対する認証機能をシステムに付与するサービスアカウントです。サービスアカウントはユーザーアカウントの変更による影響を受けないため、トークンを使用する認証方法の信頼性は高く、復元力もあります。これは、実稼働クラスター用にサポートされている唯一の認証オプションです。

- **Web ユーザー名およびパスワード**。これは、**access.redhat.com** などのリソースへのログインに使用する標準的な認証情報のセットです。OpenShift Container Platform でこの認証方法を使用することはできますが、これは実稼働デプロイメントではサポートされません。この認証方法の使用は、OpenShift Container Platform 外のスタンドアロンのプロジェクトに制限されます。

ユーザー名およびパスワード、または認証トークンのいずれかの認証情報を使用して **podman login** を使用し、新規レジストリーのコンテンツにアクセスします。

すべてのイメージストリームは、インストールプルシークレットを使用して認証を行う新規レジストリーを参照します。

認証情報は以下のいずれかの場所に配置する必要があります。

- **openshift namespace**。**openshift** namespace のイメージストリームがインポートできるように、認証情報は **openshift** namespace に配置してください。

- **ホスト**。Kubernetes でイメージをプルする際にホストの認証情報を使用するため、認証情報はホスト上に配置してください。

関連情報

- Registry service accounts

第2章 OPENSHIFT CONTAINER PLATFORM の IMAGE REGISTRY OPERATOR

2.1. クラウドプラットフォームおよび OPENSTACK のイメージレジストリー

Image Registry Operator は、OpenShift イメージレジストリーの単一インスタンスをインストールし、レジストリーストレージのセットアップを含むすべてのレジストリー設定を管理します。

注記

ストレージは、AWS、Azure、GCP、IBM または OpenStack に installer-provisioned infrastructure クラスターをインストールする場合にのみ自動的に設定されます。

installer-provisioned infrastructure クラスターを AWS、Azure、GCP、IBM、または OpenShift でインストールまたはアップグレードする場合、イメージレジストリー Operator は **spec.storage.managementState** パラメーターを **Managed** に設定します。**spec.storage.managementState** パラメーターが **Unmanaged** に設定されている場合、Image Registry Operator はストレージに関連するアクションを実行しません。

コントロールプレーンのデプロイ後、Operator はクラスターで検出される設定に基づいてデフォルトの **configs.imageregistry.operator.openshift.io** リソースインスタンスを作成します。

完全な **configs.imageregistry.operator.openshift.io** リソースを定義するために利用できる十分な情報がない場合、不完全なリソースが定義され、Operator は不足分を示す情報を使用してリソースのステータスを更新します。

Image Registry Operator は **openshift-image-registry** namespace で実行し、その場所のレジストリーインスタンスも管理します。レジストリーのすべての設定およびワークロードリソースはその namespace に置かれます。

重要

プルーナーを管理するための Image Registory Operator の動作は、Image Registory Operator の **ClusterOperator** オブジェクトで指定される **managementState** とは独立しています。Image Registory Operator が **Managed** の状態ではない場合、イメージプルーナーは **Pruning** カスタムリソースで設定および管理できます。

ただし、Image Registory Operator の **managementState** は、デプロイされたイメージプルーナージョブの動作を変更します。

- **Managed**: イメージプルーナーの **--prune-registry** フラグは **true** に設定されます。
- **Removed**: イメージプルーナーの **--prune-registry** フラグは **false** に設定されます。つまり、これは etcd のイメージメタデータのみのプルーニングを実行します。

2.2. ベアメタル、NUTANIX、および VSPHERE 上のイメージレジストリー

2.2.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、Image Registry Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。これが完了したら、ストレージを設定する必要があります。

2.3. アベイラビリティーゾーン全体での IMAGE REGISTRY OPERATOR のディストリビューション

Image Registry Operator のデフォルト設定は、イメージレジストリー Pod をトポロジーゾーン全体に分散し、すべての Pod が影響を受ける完全なゾーンに障害が発生した場合のリカバリー時間を防ぎます。

Image Registry Operator は、ゾーン関連のトポロジー制約でデプロイされる場合に、デフォルトで以下に設定されます。

ゾーン関連のトポロジー制約を使用してデプロイされた Image Registry Operator

```
topologySpreadConstraints:
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: kubernetes.io/hostname
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: node-role.kubernetes.io/worker
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: topology.kubernetes.io/zone
  whenUnsatisfiable: DoNotSchedule
```

Image Registry Operator は、ベアメタルおよび vSphere インスタンスに適用されるゾーン関連のトポロジー制約なしでデプロイされた場合、デフォルトで次のようになります。

ゾーン関連のトポロジー制約を使用せずにデプロイされた Image Registry Operator

```
topologySpreadConstraints:
- labelSelector:
    matchLabels:
      docker-registry: default
  maxSkew: 1
  topologyKey: kubernetes.io/hostname
  whenUnsatisfiable: DoNotSchedule
- labelSelector:
    matchLabels:
      docker-registry: default
```

```
maxSkew: 1
topologyKey: node-role.kubernetes.io/worker
whenUnsatisfiable: DoNotSchedule
```

クラスター管理者は、**configs.imageregistry.operator.openshift.io/cluster** 仕様ファイルを設定することで、デフォルトの **topologySpreadConstraints** をオーバーライドできます。その場合、指定した制約のみが適用されます。

2.4. 関連情報

- Pod トポロジー分散制約の設定

2.5. IMAGE REGISTRY OPERATOR の設定パラメーター

configs.imageregistry.operator.openshift.io リソースは以下の設定パラメーターを提供します。

パラメーター	説明
managementState	**Managed**: Operator は、設定リソースが更新されるとレジストリーを更新します。 **Unmanaged**: Operator は設定リソースへの変更を無視します。 **Removed**: Operator はレジストリーインスタンスを取り除き、Operator がプロビジョニングしたすべてのストレージを削除します。
logLevel	レジストリーインスタンスの **logLevel** を設定します。デフォルトは **Normal** です。 **logLevel** の次の値がサポートされています。 ● **Normal** ● **Debug** ● **Trace** ● **TraceAll**
httpSecret	デフォルトで生成されるアップロードのセキュリティーを保護するためにレジストリーで必要な値。

パラメーター	説明
operatorLogLevel	**operatorLogLevel** 設定パラメーターは、Operator 自体にインテントベースのロギングを提供し、Operator が自分で解釈する必要がある粗粒度のロギング選択を管理する簡単な方法を提供します。この設定パラメーターのデフォルトは **Normal** です。きめ細かい制御はできません。 次の **operatorLogLevel** の値がサポートされています。 ● **Normal** ● **Debug** ● **Trace** ● **TraceAll**
proxy	マスター API およびアップストリームレジストリーの呼び出し時に使用されるプロキシーを定義します。
affinity	**affinity** パラメーターを使用して、Image Registry Operator Pod の Pod スケジューリングの設定と制約を設定できます。 アフィニティー設定では、**podAffinity** または **podAntiAffinity** 仕様を使用できます。どちらのオプションでも、**preferredDuringSchedulingIgnoredDuringExecution** ルールまたは **requiredDuringSchedulingIgnoredDuringExecution** ルールのいずれかを使用できます。
storage	**Storagetype**: レジストリーストレージを設定するための詳細。たとえば、S3 バケットの位置情報 (coordinate) など。通常はデフォルトで設定されます。
readOnly	レジストリーインスタンスが新規イメージのプッシュや既存イメージの削除の試行を拒否するかどうかを示します。
requests	API 要求の制限の詳細。指定されたレジストリーインスタンスが追加リソースをキューに入れる前に処理する並列要求の数を制御します。
defaultRoute	外部ルートがデフォルトのホスト名を使用して定義されるかどうかを決定します。これが有効にされている場合、ルートは re-encrypt 暗号を使用します。デフォルトは **false** です。
routes	作成する追加ルートの配列。ルートにホスト名および証明書を指定します。
rolloutStrategy	イメージレジストリーのデプロイメントのロールアウトストラテジーを定義します。デフォルトは **RollingUpdate** です。
replicas	レジストリーのレプリカ数。
disableRedirect	バックエンドにリダイレクトするのではなく、レジストリーを介してすべてのデータをルーティングするかどうかを制御します。デフォルトは **false** です。

パラメーター	説明
spec.storage.managementState	Image Registry Operator は、AWS または Azure の installer-provisioned infrastructure を使用してクラスターの新規インストールまたはアップグレード時に **spec.storage.managementState** パラメーターを **Managed** に設定します。 • **Managed**: Image Registry Operator が基礎となるストレージを管理することを判別します。Image Registry Operator の **managementState** が **Removed** に設定されている場合、ストレージは削除されます。 ○ **managementState** が **Managed** に設定されている場合、Image Registry Operator は基礎となるストレージユニットにいくつかのデフォルト設定を適用しようとします。たとえば、**Managed** に設定されている場合、Operator はこれをレジストリーで利用可能にする前に S3 バケットで暗号を有効にしようとします。デフォルト設定を提供しているストレージに適用しないと、**managementState** は **Unmanaged** に設定されていることを確認します。 • **Unmanaged**: Image Registry Operator はストレージ設定を無視します。Image Registry Operator の **managementState** が **Removed** に設定されている場合、ストレージは削除されません。バケットまたはコンテナー名などの基礎となるストレージユニット設定を指定し、**spec.storage.managementState** がまだいずれの値にも設定されていない場合、Image Registry Operator はこれを **Unmanaged** に設定します。

2.6. イメージレジストリーのデフォルトルートをカスタムリソース定義 (CUSTOM RESOURCE DEFINITION) で有効にする

OpenShift Container Platform では、**Registry** Operator は OpenShift イメージレジストリー機能を制御します。Operator は、**configs.imageregistry.operator.openshift.io** カスタムリソース定義 (CRD) で定義されます。

イメージレジストリーのデフォルトルートを自動的に有効にする必要がある場合は、Image Registry Operator CRD のパッチを適用します。

手順

- Image Registry Operator CRD にパッチを適用します。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":
{"defaultRoute":true}}'
```

2.7. イメージレジストリーアクセス用の追加トラストストアの設定

image.config.openshift.io/cluster カスタムリソースには、イメージレジストリーのアクセス時に信頼される追加の認証局が含まれる config map への参照を含めることができます。

前提条件

- 認証局 (CA) は PEM でエンコードされている。

手順

openshift-config namespace で config map を作成し、 **image.config.openshift.io** カスタムリソースの **AdditionalTrustedCA** でその名前を使用して、外部レジストリーにアクセスするときに信頼する必要がある追加の CA を提供できます。

config map のキーは、この CA を信頼するポートがあるレジストリーのホスト名であり、値は各追加レジストリー CA が信頼する証明書のコンテンツです。

イメージレジストリー CA の config map の例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-registry-ca
data:
  registry.example.com: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
  registry-with-port.example.com..5000: | (1)
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

(1) レジストリーにポートがある場合 (例: **registry-with-port.example.com:5000**)、**:** は **..** に置き換える必要があります。

以下の手順で追加の CA を設定できます。

1. 追加の CA を設定するには、以下を実行します。

```
$ oc create configmap registry-config --from-file=<external_registry_address>=ca.crt -n
openshift-config
```

```
$ oc edit image.config.openshift.io cluster
```

```
spec:
  additionalTrustedCA:
    name: registry-config
```

2.8. IMAGE REGISTRY OPERATOR のストレージ認証情報の設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースの他にも、 **openshift-image-registry** namespace 内の別のシークレットリソースによってストレージの認証情報の設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator が使用するデフォルト認証情報をオーバーライドします。

手順

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=KEY1=value1 --from-literal=KEY2=value2 --namespace openshift-image-registry
```

2.9. 関連情報

- AWS の user-provisioned infrastructure のレジストリーの設定
- GCP の user-provisioned infrastructure のレジストリーの設定
- Azure user-provisioned infrastructure のレジストリーの設定
- ベアメタルのレジストリーの設定
- vSphere のレジストリーの設定
- RHOSP のレジストリーの設定
- Red Hat OpenShift Data Foundation のレジストリーの設定
- Nutanix 用レジストリーの設定

第3章 レジストリーのセットアップおよび設定

3.1. AWS の USER-PROVISIONED INFRASTRUCTURE のレジストリーの設定

3.1.1. Image Registry Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、 **openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator が使用するデフォルト認証情報をオーバーライドします。

AWS ストレージの S3 の場合、シークレットには以下のキーが含まれることが予想されます。

- **REGISTRY_STORAGE_S3_ACCESSKEY**
- **REGISTRY_STORAGE_S3_SECRETKEY**

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=myaccesskey --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=mysecretkey --namespace openshift-image-registry
```

3.1.2. user-provisioned infrastructure を使用した AWS のレジストリーストレージの設定

インストール時に、Amazon S3 バケットを作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合、以下の手順により S3 バケットを作成し、ストレージを設定することができます。

前提条件

- user-provisioned infrastructure を使用した AWS 上にクラスターがある。
- Amazon S3 ストレージの場合、シークレットには以下のキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_S3_ACCESSKEY**
  - **REGISTRY_STORAGE_S3_SECRETKEY**

手順

レジストリー Operator が S3 バケットを作成できず、ストレージを自動的に設定する場合は、以下の手順を使用してください。

1. バケットライフサイクルポリシー を設定し、1 日以上経過している未完了のマルチパートアップロードを中止します。

2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
```

警告

AWS でレジストリーイメージのセキュリティーを保護するには、S3 バケットに対して パブリックアクセスのブロック を実行します。

3.1.3. AWS S3 の Image Registry Operator 設定パラメーター

以下の設定パラメーターは AWS S3 レジストリーストレージで利用できます。

イメージレジストリーの **spec.storage.s3** 設定パラメーターには、バックエンドストレージに AWS S3 サービスを使用するようにレジストリーを設定するための情報が保持されます。詳細は、S3 ストレージドライバーのドキュメント を参照してください。

パラメーター	説明
bucket	バケットは、レジストリーのデータを保存するバケット名です。これはオプションであり、指定されていない場合は生成されます。
region	リージョンはバケットが存在する AWS リージョンです。これはオプションであり、インストール済みの AWS リージョンに基づいて設定されます。
regionEndpoint	RegionEndpoint は、S3 互換のストレージサービスのエンドポイントです。これは、指定されるリージョンに応じてオプションおよびデフォルトになります。
virtualHostedStyle	VirtualHosted は、カスタム RegionEndpoint で S3 仮想ホストスタイルのバケットパスの使用を有効にします。これはオプションであり、デフォルトは false です。 このパラメーターを設定して、OpenShift Container Platform を非表示のリージョンにデプロイします。

パラメーター	説明
encrypt	encrypt は、イメージが暗号化された形式で保存されるかどうかを指定します。これはオプションであり、デフォルトは false です。
keyID	KeyID は、暗号化に使用する KMS キー ID です。これはオプションです。encrypt は true である必要があります。そうでない場合、このパラメーターは無視されます。
cloudFront	CloudFront は Amazon Cloudfront をレジストリーでストレージミドルウェアとして設定します。これはオプションです。
trustedCA	**trustedCA** によって参照される設定マップの namespace は **openshift-config** です。config map 内のバンドルのキーは **ca-bundle.crt** です。これはオプションです。

注記

regionEndpoint パラメーターの値を Rados Gateway の URL に設定する場合、明示的なポートを指定してはなりません。以下に例を示します。

```
regionEndpoint: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc.cluster.local
```

3.2. GCP の USER-PROVISIONED INFRASTRUCTURE のレジストリーの設定

3.2.1. Image Registry Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、**openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator が使用するデフォルト認証情報をオーバーライドします。

GCP ストレージ上の GCS の場合、シークレットには、GCP が提供する認証情報ファイルの内容に相当するキーが含まれることが予想されます。

- **REGISTRY_STORAGE_GCS_KEYFILE**

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-file=REGISTRY_STORAGE_GCS_KEYFILE=<path_to_keyfile> --namespace openshift-image-registry
```

3.2.2. user-provisioned infrastructure を使用して GCP のレジストリーストレージを設定する

Registry Operator が Google Cloud Platform (GCP) バケットを作成できない場合は、ストレージメディアを手動でセットアップし、レジストリーのカスタムリソース (CR) で設定を行う必要があります。

前提条件

- user-provisioned infrastructure を持つ GCP 上のクラスター。
- GCP のレジストリーストレージを設定するには、レジストリー Operator クラウド認証情報を指定する必要があります。
- GCP ストレージ上の GCS の場合、シークレットには、GCP が提供する認証情報ファイルの内容に相当するキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_GCS_KEYFILE**

手順

1. 経過した未完了のマルチパートアップロードを中止するための Object Lifecycle Management ポリシー を設定します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
# ...
storage:
  gcs:
    bucket: <bucket-name>
    projectID: <project-id>
    region: <region-name>
# ...
```

警告

public access prevention を設定することにより、Google Cloud Storage バケットを使用するレジストリーイメージを保護できます。

3.2.3. GCP GCS の Image Registry Operator 設定パラメーター。

以下の設定パラメーターは、GCP GCS レジストリーストレージに利用できます。

パラメーター	説明
bucket	バケットは、レジストリーのデータを保存するバケット名です。これはオプションであり、指定されていない場合は生成されます。
region	リージョンは、バケットが存在する GCS の場所です。これはオプションであり、インストールされている GCS リージョンに基づいて設定されます。
projectID	ProjectID は、このバケットが関連付けられる必要がある GCP プロジェクトのプロジェクト ID です。これはオプションです。
keyID	KeyID は、暗号化に使用する KMS キー ID です。バケットは GCP でデフォルトで暗号化されているため、これはオプションになります。これにより、カスタム暗号化キーを使用できます。

3.3. OPENSTACK USER-PROVISIONED INFRASTRUCTURE のレジストリーの設定

独自の Red Hat OpenStack Platform (RHOSP) インフラストラクチャーで実行されるクラスターのレジストリーを設定できます。

3.3.1. Swift ストレージを信頼する Image Registry Operator の設定

Image Registry Operator を、Red Hat OpenStack Platform (RHOSP) Swift ストレージを信頼するように設定する必要があります。

手順

- コマンドラインから次のコマンドを入力して、**config.imageregistry** オブジェクトの **spec.disableRedirect** フィールドの値を **true** に変更します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"disableRedirect":true}}'
```

3.3.2. Image Registry Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、**openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator が使用するデフォルト認証情報をオーバーライドします。

Red Hat OpenStack Platform (RHOSP) ストレージ上の Swift の場合、シークレットには次の 2 つのキーが含まれている必要があります。

- **REGISTRY_STORAGE_SWIFT_USERNAME**
- **REGISTRY_STORAGE_SWIFT_PASSWORD**

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_SWIFT_USERNAME=<username> --from-literal=REGISTRY_STORAGE_SWIFT_PASSWORD=<password> -n openshift-image-registry
```

3.3.3. user-provisioned infrastructure での RHOSP のレジストリーストレージ

Registry Operator が Swift バケットを作成できない場合は、ストレージメディアを手動でセットアップし、レジストリーのカスタムリソース (CR) で設定を行う必要があります。

前提条件

- user-provisioned infrastructure を備えた Red Hat OpenStack Platform (RHOSP) 上のクラスター。
- RHOSP のレジストリーストレージを設定するには、レジストリー Operator クラウド認証情報を指定する必要があります。
- RHOSP ストレージ上の Swift の場合、シークレットには次の 2 つのキーが含まれている必要があります。
  - **REGISTRY_STORAGE_SWIFT_USERNAME**
  - **REGISTRY_STORAGE_SWIFT_PASSWORD**

手順

- **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
# ...
storage:
  swift:
    container: <container-id>
# ...
```

3.3.4. RHOSP Swift の Image Registry Operator 設定パラメーター

以下の設定パラメーターは Red Hat OpenStack Platform (RHOSP) Swift レジストリーストレージで利用できます。

パラメーター	説明
authURL	認証トークンを取得するための URL を定義します。この値はオプションです。

パラメーター	説明
authVersion	RHOSP の Auth バージョンを指定します (例: **authVersion: "3"**)。この値はオプションです。
container	レジストリーデータを保存する Swift コンテナーの名前を定義します。この値はオプションです。
domain	Identity v3 API の RHOSP ドメイン名を指定します。この値はオプションです。
domainID	Identity v3 API の RHOSP ドメイン ID を指定します。この値はオプションです。
tenant	レジストリーで使用される RHOSP テナント名を定義します。この値はオプションです。
tenantID	レジストリーで使用される RHOSP テナント ID を定義します。この値はオプションです。
regionName	コンテナーが存在する RHOSP リージョンを定義します。この値はオプションです。

3.4. AZURE USER-PROVISIONED INFRASTRUCTURE のレジストリーの設定

3.4.1. Image Registry Operator のシークレットの設定

configs.imageregistry.operator.openshift.io および ConfigMap リソースのほかにも、 **openshift-image-registry** namespace 内の別のシークレットリソースによって設定が Operator に提供されます。

image-registry-private-configuration-user シークレットは、ストレージのアクセスおよび管理に必要な認証情報を提供します。これは、デフォルト認証情報が見つからない場合に Operator が使用するデフォルト認証情報をオーバーライドします。

Azure レジストリーストレージの場合、シークレットには、Azure が提供する認証情報ファイルの内容に相当する値を持つキーが含まれることが予想されます。

- **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

手順

- 必要なキーが含まれる OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_AZURE_ACCOUNTKEY=<accountkey> --namespace openshift-image-registry
```

3.4.2. Azure の場合のレジストリーストレージの設定

インストール時に、Azure Blob Storage を作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

前提条件

- user-provisioned infrastructure での Azure 上のクラスター。
- Azure のレジストリーストレージを設定するには、レジストリー Operator クラウド認証情報を指定する必要があります。
- Azure ストレージの場合、シークレットには 1 つのキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

手順

1. Azure ストレージコンテナー を作成します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  azure:
    accountName: <storage-account-name>
    container: <container-name>
```

3.4.3. Azure Government の場合のレジストリーストレージの設定

インストール時に、Azure Blob Storage を作成するにはクラウド認証情報を使用でき、レジストリー Operator がストレージを自動的に設定します。

前提条件

- Government リージョンの user-provisioned infrastructure での Azure 上のクラスター。
- Azure のレジストリーストレージを設定するには、レジストリー Operator クラウド認証情報を指定する必要があります。
- Azure ストレージの場合、シークレットには 1 つのキーが含まれることが予想されます。
  - **REGISTRY_STORAGE_AZURE_ACCOUNTKEY**

手順

1. Azure ストレージコンテナー を作成します。
2. **configs.imageregistry.operator.openshift.io/cluster** にストレージ設定を入力します。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

設定例

```
storage:
  azure:
    accountName: <storage-account-name>
    container: <container-name>
    cloudName: AzureUSGovernmentCloud (1)
```

(1) **cloudName** は、適切な Azure API エンドポイントで Azure SDK を設定するために使用できる Azure クラウド環境の名前。デフォルトで **AzurePublicCloud** に設定されます。また、適切な認証情報を使用して **cloudName** を **AzureUSGovernmentCloud**、**AzureChinaCloud**、または **AzureGermanCloud** に設定することもできます。

3.5. RHOSP のレジストリーの設定

3.5.1. RHOSP で実行されるクラスター上のカスタムストレージを使用したイメージレジストリーの設定

Red Hat OpenStack Platform (RHOSP) にクラスターをインストールした後に、特定のアベイラビリティーゾーンにある Cinder ボリュームをレジストリーストレージとして使用できます。

手順

1. YAML ファイルを作成して、使用するストレージクラスとアベイラビリティーゾーンを指定します。以下に例を示します。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: custom-csi-storageclass
provisioner: cinder.csi.openstack.org
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  availability: <availability_zone_name>
```

注記

OpenShift Container Platform では、選択したアベイラビリティーゾーンが存在するかどうかは確認されません。設定を適用する前に、アベイラビリティーゾーンの名前を確認してください。

2. コマンドラインから設定を適用します。

```
$ oc apply -f <storage_class_file_name>
```

出力例

```
storageclass.storage.k8s.io/custom-csi-storageclass created
```

3. ストレージクラスと **openshift-image-registry** namespace を使用する永続ボリュームクレーム (PVC) を指定する YAML ファイルを作成します。以下に例を示します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-imageregistry
  namespace: openshift-image-registry (1)
  annotations:
    imageregistry.openshift.io: "true"
spec:
  accessModes:
  - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 100Gi (2)
  storageClassName: <your_custom_storage_class> (3)
```

1. **openshift-image-registry** namespace を入力します。この namespace により、クラスターイメージレジストリーオペレーターは PVC を使用できます。
2. オプション: ボリュームサイズを調整します。
3. 作成されるストレージクラスの名前を入力します。

4. コマンドラインから設定を適用します。

```
$ oc apply -f <pvc_file_name>
```

出力例

```
persistentvolumeclaim/csi-pvc-imageregistry created
```

5. イメージレジストリー設定の元の永続ボリューム要求は、新しい要求に置き換えます。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type 'json' -p='[{"op":
"replace", "path": "/spec/storage/pvc/claim", "value": "csi-pvc-imageregistry"}]'
```

出力例

```
config.imageregistry.operator.openshift.io/cluster patched
```

数分すると、設定が更新されます。

検証

レジストリーが定義したリソースを使用していることを確認するには、以下を実行します。

1. PVC クレーム値が PVC 定義で指定した名前と同じであることを確認します。

```
$ oc get configs.imageregistry.operator.openshift.io/cluster -o yaml
```

出力例

```
...
status:
  ...
  managementState: Managed
  pvc:
    claim: csi-pvc-imageregistry
...
```

2. PVC のステータスが **Bound** であることを確認します。

```
$ oc get pvc -n openshift-image-registry csi-pvc-imageregistry
```

出力例

```
NAME                    STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS            AGE
csi-pvc-imageregistry   Bound    pvc-72a8f9c9-f462-11e8-b6b6-fa163e18b7b5   100Gi      RWO
custom-csi-storageclass   11m
```

3.6. ベアメタルのレジストリーの設定

3.6.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、Image Registry Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。これが完了したら、ストレージを設定する必要があります。

3.6.2. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"managementState":"Managed"}}'
```

3.6.3. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定に関する手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

3.6.3.1. ベアメタルおよび他の手動インストールの場合のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- ベアメタルなどの、手動でプロビジョニングされた Red Hat Enterprise Linux CoreOS (RHCOS) ノードを使用するクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

> **重要**
>
> OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

> **注記**
>
> 共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resources found in openshift-image-registry namespace
```

> **注記**
>
> 出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim:
```

claim フィールドを空のままにし、**image-registry-storage** PVC の自動作成を可能にします。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME            VERSION            AVAILABLE   PROGRESSING   DEGRADED   SINCE
MESSAGE
image-registry   4.12               True        False         False      6h50m
```

5. イメージのビルドおよびプッシュを有効にするためにレジストリーが managed に設定されていることを確認します。

- 以下を実行します。

```
$ oc edit configs.imageregistry/cluster
```

次に、行を変更します。

```
managementState: Removed
```

次のように変更してください。

```
managementState: Managed
```

3.6.3.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

警告

実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

3.6.3.3. ベアメタルの場合のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時にブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1 つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
```

```
resources:
  requests:
    storage: 100Gi (4)
```

1. **PersistentVolumeClaim** オブジェクトを表す一意の名前。
2. **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。
3. 永続ボリューム要求のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。
4. 永続ボリューム要求のサイズ。

b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

1. カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

3.6.3.4. Red Hat OpenShift Data Foundation で Ceph RGW ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Ceph RGW ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。

- オブジェクトストレージおよび Ceph RGW オブジェクトストレージを提供するために OpenShift Data Foundation Operator をインストールしている。

手順

1. **ocs-storagecluster-ceph-rgw** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、**route** ホストを取得します。

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/router-certs-default  -n openshift-ingress  --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Ceph RGW オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.6.3.5. Red Hat OpenShift Data Foundation で Noobaa ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Noobaa ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- OpenShift Data Foundation Operator をインストールして、オブジェクトストレージと Noobaa オブジェクトストレージを提供しました。

手順

1. **openshift-storage.noobaa.io** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、ルートホストを取得します。

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Nooba オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.6.4. Red Hat OpenShift Data Foundation で CephFS ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、CephFS ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

注記

CephFS は persistent volume claim (PVC) ストレージを使用します。Ceph RGW や Noobaa など、他のオプションが利用可能な場合は、イメージレジストリーストレージに PVC を使用することは推奨しません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージと CephFS ファイルストレージを提供するために、OpenShift Data Foundation Operator をインストールしました。

手順

1. **cephfs** ストレージクラスを使用する PVC を作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 次のコマンドを入力して、CephFS ファイルシステムストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.6.5. 関連情報

- 設定可能な推奨のストレージ技術
- OpenShift Data Foundation を使用するためのイメージレジストリーの設定

3.7. VSPHERE のレジストリーの設定

3.7.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、Image Registry Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。これが完了したら、ストレージを設定する必要があります。

3.7.2. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
```

3.7.3. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定に関する手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

3.7.3.1. VMware vSphere のレジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- クラスター管理者のパーミッション。
- VMware vSphere 上のクラスター。
- Red Hat OpenShift Data Foundation など、クラスターのプロビジョニングされた永続ストレージ。

重要

OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- "100Gi" の容量が必要です。

重要

テストにより、NFS サーバーを RHEL でコアサービスのストレージバックエンドとして使用することに関する問題が検出されています。これには、OpenShift Container レジストリーおよび Quay、メトリックストレージの Prometheus、およびロギングストレージの Elasticsearch が含まれます。そのため、コアサービスで使用される PV をサポートするために RHEL NFS を使用することは推奨されていません。

他の NFS の実装ではこれらの問題が検出されない可能性があります。OpenShift Container Platform コアコンポーネントに対して実施された可能性のあるテストに関する詳細情報は、個別の NFS 実装ベンダーにお問い合わせください。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

注記

共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

注記

出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生成されます。ただし、デフォルトのストレージクラスは、RADOS ブロックデバイス (RBD) などの ReadWriteOnce (RWO) ボリュームを提供する可能性があることに注意してください。これは、複数のレプリカに複製するときに問題を引き起こす可能性があります。

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME            VERSION                         AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.7                            True        False         False      6h50m
```

3.7.3.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

3.7.3.3. VMware vSphere のブロックレジストリーストレージの設定

イメージレジストリーがクラスター管理者によるアップグレード時に vSphere Virtual Machine Disk (VMDK) などのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用できます。

重要

ブロックストレージボリュームはサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、**1** つのレプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して VMware vSphere **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

   (3) 永続ボリューム要求のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (4) 永続ボリューム要求のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

正しい PVC を参照するようにレジストリーストレージを設定する手順は、vSphere のレジストリーの設定 を参照してください。

3.7.3.4. Red Hat OpenShift Data Foundation で Ceph RGW ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Ceph RGW ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージおよび Ceph RGW オブジェクトストレージを提供するために OpenShift Data Foundation Operator をインストールしている。

手順

1. **ocs-storagecluster-ceph-rgw** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

1. あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、**route** ホストを取得します。

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/router-certs-default  -n openshift-ingress  --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Ceph RGW オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.7.3.5. Red Hat OpenShift Data Foundation で Noobaa ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Noobaa ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- OpenShift Data Foundation Operator をインストールして、オブジェクトストレージと Noobaa オブジェクトストレージを提供しました。

手順

1. **openshift-storage.noobaa.io** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --namespace openshift-image-registry
```

5. 次のコマンドを入力して、ルートホストを取得します。

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq '.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n openshift-config
```

7. 次のコマンドを入力して、Nooba オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":{"managementState":"Managed","replicas":2,"storage":{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedCA":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.7.4. Red Hat OpenShift Data Foundation で CephFS ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、CephFS ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

注記

CephFS は persistent volume claim (PVC) ストレージを使用します。Ceph RGW や Noobaa など、他のオプションが利用可能な場合は、イメージレジストリーストレージに PVC を使用することは推奨しません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。

- オブジェクトストレージと CephFS ファイルストレージを提供するために、OpenShift Data Foundation Operator をインストールしました。

手順

1. **cephfs** ストレージクラスを使用する PVC を作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 次のコマンドを入力して、CephFS ファイルシステムストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.7.5. 関連情報

- 設定可能な推奨のストレージ技術
- OpenShift Data Foundation を使用するためのイメージレジストリーの設定

3.8. RED HAT OPENSHIFT DATA FOUNDATION のレジストリーの設定

Red Hat OpenShift Data Foundation ストレージを使用するように OpenShift イメージレジストリーをベアメタルおよび vSphere に設定するには、Ceph または Noobaa を使用してイメージレジストリーを設定する必要があります。

3.8.1. Red Hat OpenShift Data Foundation で Ceph RGW ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Ceph RGW ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージおよび Ceph RGW オブジェクトストレージを提供するために OpenShift Data Foundation Operator をインストールしている。

手順

1. **ocs-storagecluster-ceph-rgw** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、**route** ホストを取得します。

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/router-certs-default  -n openshift-ingress  --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Ceph RGW オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.8.2. Red Hat OpenShift Data Foundation で Noobaa ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Noobaa ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- OpenShift Data Foundation Operator をインストールして、オブジェクトストレージと Noobaa オブジェクトストレージを提供しました。

手順

1. **openshift-storage.noobaa.io** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
```

```
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、ルートホストを取得します。

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Nooba オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}' --type=merge
```

3.8.3. Red Hat OpenShift Data Foundation で CephFS ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、CephFS ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

注記

CephFS は persistent volume claim (PVC) ストレージを使用します。Ceph RGW や Noobaa など、他のオプションが利用可能な場合は、イメージレジストリーストレージに PVC を使用することは推奨しません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージと CephFS ファイルストレージを提供するために、OpenShift Data Foundation Operator をインストールしました。

手順

1. **cephfs** ストレージクラスを使用する PVC を作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 次のコマンドを入力して、CephFS ファイルシステムストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.8.4. 関連情報

- OpenShift Data Foundation を使用するためのイメージレジストリーの設定
- Multicloud Object Gateway(NooBaa) のパフォーマンスチューニングガイド

3.9. NUTANIX 用レジストリーの設定

このドキュメントで説明されている手順に従うことで、ユーザーはコンテナーイメージの配布、セキュリティー、およびアクセス制御を最適化し、OpenShift Container Platform 上で Nutanix アプリケーションの堅牢な基盤を実現できます。

3.9.1. インストール時に削除されたイメージレジストリー

共有可能なオブジェクトストレージを提供しないプラットフォームでは、OpenShift Image Registry Operator 自体が **Removed** としてブートストラップされます。これにより、**openshift-installer** がそれらのプラットフォームタイプでのインストールを完了できます。

インストール後に、Image Registry Operator 設定を編集して **managementState** を **Removed** から **Managed** に切り替える必要があります。これが完了したら、ストレージを設定する必要があります。

3.9.2. イメージレジストリーの管理状態の変更

イメージレジストリーを起動するには、Image Registry Operator 設定の **managementState** を **Removed** から **Managed** に変更する必要があります。

手順

- **managementState** Image Registry Operator 設定を **Removed** から **Managed** に変更します。以下に例を示します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"managementState":"Managed"}}'
```

3.9.3. イメージレジストリーストレージの設定

Image Registry Operator は、デフォルトストレージを提供しないプラットフォームでは最初は利用できません。インストール後に、レジストリー Operator を使用できるようにレジストリーをストレージを使用するように設定する必要があります。

実稼働クラスターに必要な永続ボリュームの設定に関する手順が示されます。該当する場合、空のディレクトリーをストレージの場所として設定する方法が表示されます。これは、実稼働以外のクラスターでのみ利用できます。

アップグレード時に **Recreate** ロールアウトストラテジーを使用して、イメージレジストリーがブロックストレージタイプを使用することを許可するための追加の手順が提供されます。

3.9.3.1. Nutanix 用レジストリーストレージの設定

クラスター管理者は、インストール後にレジストリーをストレージを使用できるように設定する必要があります。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- Nutanix にクラスターがある。
- Red Hat OpenShift Data Foundation などのクラスターのプロビジョニングされた永続ストレージがある。

> **重要**
>
> OpenShift Container Platform は、1 つのレプリカのみが存在する場合にイメージレジストリーストレージの **ReadWriteOnce** アクセスをサポートします。**ReadWriteOnce** アクセスでは、レジストリーが **Recreate** ロールアウト戦略を使用する必要もあります。2 つ以上のレプリカで高可用性をサポートするイメージレジストリーをデプロイするには、**ReadWriteMany** アクセスが必要です。

- 100 Gi の容量がある。

手順

1. レジストリーをストレージを使用できるように設定するには、**configs.imageregistry/cluster** リソースの **spec.storage.pvc** を変更します。

> **注記**
>
> 共有ストレージを使用する場合は、外部からアクセスを防ぐためにセキュリティー設定を確認します。

2. レジストリー Pod がないことを確認します。

```
$ oc get pod -n openshift-image-registry -l docker-registry=default
```

出力例

```
No resourses found in openshift-image-registry namespace
```

> **注記**
>
> 出力にレジストリー Pod がある場合は、この手順を続行する必要はありません。

3. レジストリー設定を確認します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

出力例

```
storage:
  pvc:
    claim: (1)
```

1. **image-registry-storage** 永続ボリューム要求 (PVC) の自動作成を許可するには、**claim** フィールドを空白のままにします。PVC は、デフォルトのストレージクラスに基づいて生

4. **clusteroperator** ステータスを確認します。

```
$ oc get clusteroperator image-registry
```

出力例

```
NAME            VERSION                     AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
image-registry   4.13                       True        False         False      6h50m
```

3.9.3.2. 実稼働以外のクラスターでのイメージレジストリーのストレージの設定

Image Registry Operator のストレージを設定する必要があります。実稼働用以外のクラスターの場合、イメージレジストリーは空のディレクトリーに設定することができます。これを実行する場合、レジストリーを再起動するとすべてのイメージが失われます。

手順

- イメージレジストリーストレージを空のディレクトリーに設定するには、以下を実行します。

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":
{"storage":{"emptyDir":{}}}}'
```

> **警告**
>
> 実稼働用以外のクラスターにのみこのオプションを設定します。

Image Registry Operator がそのコンポーネントを初期化する前にこのコマンドを実行する場合、**oc patch** コマンドは以下のエラーを出して失敗します。

```
Error from server (NotFound): configs.imageregistry.operator.openshift.io "cluster" not found
```

数分待機した後に、このコマンドを再び実行します。

3.9.3.3. Nutanix ボリューム用ブロックレジストリーストレージの設定

アップグレード時にイメージレジストリーがクラスター管理者として Nutanix ボリュームなどのブロックストレージタイプを使用できるようにするには、**Recreate** ロールアウトストラテジーを使用します。

重要

ブロックストレージボリューム (または永続ボリューム) はサポートされますが、実稼働クラスターでのイメージレジストリーと併用することは推奨されません。レジストリーに複数のレプリカを含めることができないため、ブロックストレージにレジストリーが設定されているインストールに高可用性はありません。

イメージレジストリーでブロックストレージボリュームを使用することを選択した場合は、ファイルシステムの persistent volume claim (PVC) を使用する必要があります。

手順

1. 次のコマンドを入力してイメージレジストリーストレージをブロックストレージタイプとして設定し、レジストリーにパッチを適用して **Recreate** ロールアウトストラテジーを使用し、1つの (**1**) レプリカのみで実行されるようにします。

```
$ oc patch config.imageregistry.operator.openshift.io/cluster --type=merge -p '{"spec":
{"rolloutStrategy":"Recreate","replicas":1}}'
```

2. ブロックストレージデバイスの PV をプロビジョニングし、そのボリュームの PVC を作成します。要求されたブロックボリュームは ReadWriteOnce (RWO) アクセスモードを使用します。

   a. 以下の内容で **pvc.yaml** ファイルを作成して、Nutanix **PersistentVolumeClaim** オブジェクトを定義します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage (1)
  namespace: openshift-image-registry (2)
spec:
  accessModes:
  - ReadWriteOnce (3)
  resources:
    requests:
      storage: 100Gi (4)
```

   (1) **PersistentVolumeClaim** オブジェクトを表す一意の名前。

   (2) **PersistentVolumeClaim** オブジェクトの namespace (**openshift-image-registry**)。

   (3) 永続ボリューム要求のアクセスモード。**ReadWriteOnce** では、ボリュームは単一ノードによって読み取り/書き込みパーミッションでマウントできます。

   (4) 永続ボリューム要求のサイズ。

   b. 次のコマンドを入力して、ファイルから **PersistentVolumeClaim** オブジェクトを作成します。

```
$ oc create -f pvc.yaml -n openshift-image-registry
```

3. 次のコマンドを入力して、正しい PVC を参照するようにレジストリー設定を編集します。

```
$ oc edit config.imageregistry.operator.openshift.io -o yaml
```

出力例

```
storage:
  pvc:
    claim: (1)
```

(1) カスタム PVC を作成することにより、**image-registry-storage** PVC のデフォルトの自動作成の **claim** フィールドを空のままにできます。

3.9.3.4. Red Hat OpenShift Data Foundation で Ceph RGW ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Ceph RGW ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージおよび Ceph RGW オブジェクトストレージを提供するために OpenShift Data Foundation Operator をインストールしている。

手順

1. **ocs-storagecluster-ceph-rgw** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: rgwbucket
  namespace: openshift-storage (1)
spec:
  storageClassName: ocs-storagecluster-ceph-rgw
  generateBucketName: rgwbucket
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage rgwbucket -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_ACCESS_KEY_ID}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage rgwbucket -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、**route** ホストを取得します。

```
$ route_host=$(oc get route ocs-storagecluster-cephobjectstore -n openshift-storage --
template='{{ .spec.host }}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/router-certs-default  -n openshift-ingress  --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Ceph RGW オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}}' --type=merge
```

3.9.3.5. Red Hat OpenShift Data Foundation で Noobaa ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、Noobaa ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- OpenShift Data Foundation Operator をインストールして、オブジェクトストレージと Noobaa オブジェクトストレージを提供しました。

手順

1. **openshift-storage.noobaa.io** ストレージクラスを使用してオブジェクトバケットクレームを作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaatest
  namespace: openshift-storage (1)
spec:
  storageClassName: openshift-storage.noobaa.io
  generateBucketName: noobaatest
EOF
```

(1) あるいは、**openshift-image-registry** namespace を使用することもできます。

2. 次のコマンドを入力して、バケット名を取得します。

```
$ bucket_name=$(oc get obc -n openshift-storage noobaatest -o
jsonpath='{.spec.bucketName}')
```

3. 次のコマンドを入力して、AWS 認証情報を取得します。

```
$ AWS_ACCESS_KEY_ID=$(oc get secret -n openshift-storage noobaatest -o yaml | grep -w
"AWS_ACCESS_KEY_ID:" | head -n1 | awk '{print $2}' | base64 --decode)
```

```
$ AWS_SECRET_ACCESS_KEY=$(oc get secret -n openshift-storage noobaatest -o yaml |
grep -w "AWS_SECRET_ACCESS_KEY:" | head -n1 | awk '{print $2}' | base64 --decode)
```

4. 次のコマンドを入力して、**openshift-image-registry プロジェクト** の下にある新しいバケットの AWS 認証情報を使用して、秘密の **image-registry-private-configuration-user** を作成します。

```
$ oc create secret generic image-registry-private-configuration-user --from-
literal=REGISTRY_STORAGE_S3_ACCESSKEY=${AWS_ACCESS_KEY_ID} --from-
literal=REGISTRY_STORAGE_S3_SECRETKEY=${AWS_SECRET_ACCESS_KEY} --
namespace openshift-image-registry
```

5. 次のコマンドを入力して、ルートホストを取得します。

```
$ route_host=$(oc get route s3 -n openshift-storage -o=jsonpath='{.spec.host}')
```

6. 次のコマンドを入力して、入力証明書を使用する設定マップを作成します。

```
$ oc extract secret/$(oc get ingresscontroller -n openshift-ingress-operator default -o json | jq
'.spec.defaultCertificate.name // "router-certs-default"' -r) -n openshift-ingress --confirm
```

```
$ oc create configmap image-registry-s3-bundle --from-file=ca-bundle.crt=./tls.crt  -n
openshift-config
```

7. 次のコマンドを入力して、Nooba オブジェクトストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","s3":{"bucket":"\"${bucket_name}\"","region":"us-east-
1","regionEndpoint":"\"https://${route_host}\"","virtualHostedStyle":false,"encrypt":false,"trustedC
A":{"name":"image-registry-s3-bundle"}}}}' --type=merge
```

3.9.4. Red Hat OpenShift Data Foundation で CephFS ストレージを使用するための Image Registry Operator の設定

Red Hat OpenShift Data Foundation は、OpenShift イメージレジストリーで使用できる複数のストレージタイプを統合します。

- Ceph、共有および分散ファイルシステムとオンプレミスオブジェクトストレージ
- NooBaa。Multicloud Object Gateway を提供します。

このドキュメントでは、CephFS ストレージを使用するようにイメージレジストリーを設定する手順の概要を説明します。

注記

CephFS は persistent volume claim (PVC) ストレージを使用します。Ceph RGW や Noobaa など、他のオプションが利用可能な場合は、イメージレジストリーストレージに PVC を使用することは推奨しません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift Container Platform Web コンソールにアクセスできる。
- **oc** CLI がインストールされている。
- オブジェクトストレージと CephFS ファイルストレージを提供するために、OpenShift Data Foundation Operator をインストールしました。

手順

1. **cephfs** ストレージクラスを使用する PVC を作成します。以下に例を示します。

```
cat <<EOF | oc apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: registry-storage-pvc
 namespace: openshift-image-registry
spec:
 accessModes:
 - ReadWriteMany
 resources:
   requests:
     storage: 100Gi
 storageClassName: ocs-storagecluster-cephfs
EOF
```

2. 次のコマンドを入力して、CephFS ファイルシステムストレージを使用するようにイメージレジストリーを設定します。

```
$ oc patch config.image/cluster -p '{"spec":
{"managementState":"Managed","replicas":2,"storage":
{"managementState":"Unmanaged","pvc":{"claim":"registry-storage-pvc"}}}}' --type=merge
```

3.9.5. 関連情報

- 設定可能な推奨のストレージ技術
- OpenShift Data Foundation を使用するためのイメージレジストリーの設定

第4章 レジストリーへのアクセス

ログおよびメトリクスの表示やレジストリーのセキュリティー保護および公開など、レジストリーへのさまざまなアクセス方法については、以下のセクションを参照してください。

レジストリーに直接アクセスし、**podman** コマンドを起動できます。これにより、**podman push** や **podman pull** などの操作で統合レジストリーに対して、もしくは統合レジストリーからイメージを直接プッシュまたはプルすることができます。これを実行するには、**podman login** コマンドを使用してレジストリーにログインしている必要があります。実行できる操作は、以下のセクションで説明されているようにユーザーが持つパーミッションによって異なります。

4.1. 前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- アイデンティティープロバイダー (IDP) を設定している。
- **podman pull** コマンドを使用する場合などにイメージをプルするために、**registry-viewer** ロールがある。このロールを追加するには、以下のコマンドを実行します。

```
$ oc policy add-role-to-user registry-viewer <user_name>
```

- イメージの書き出しやプッシュを実行するために (**podman push** コマンドを使用する場合など)、以下が完了している。
  - ユーザーに **registry-editor** ロールを指定する。このロールを追加するには、以下のコマンドを実行します。

```
$ oc policy add-role-to-user registry-editor <user_name>
```

  - クラスターに、イメージをプッシュできる既存のプロジェクトを用意する。

4.2. クラスターからレジストリーへの直接アクセス

クラスター内からレジストリーにアクセスできる。

手順

内部ルートを使用して、クラスターからレジストリーにアクセスします。

1. ノードの名前を取得してノードにアクセスします。

```
$ oc get nodes
```

```
$ oc debug nodes/<node_name>
```

2. ノードで **oc** や **podman** などのツールへのアクセスを有効にするには、ルートディレクトリーを **/host** に変更します。

```
sh-4.2# chroot /host
```

3. アクセストークンを使用してコンテナーイメージレジストリーにログインします。

```
sh-4.2# oc login -u kubeadmin -p <password_from_install_log> https://api-int.
<cluster_name>.<base_domain>:6443
```

```
sh-4.2# podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-
registry.svc:5000
```

以下のようなログインを確認するメッセージが表示されるはずです。

```
Login Succeeded!
```

注記

ユーザー名には任意の値を指定でき、トークンには必要な情報がすべて含まれます。コロンが含まれるユーザー名を指定すると、ログインに失敗します。

Image Registry Operator はルートを作成するため、**default-route-openshift-image-registry.<cluster_name>** のようになります。

4. レジストリーに対して **podman pull** および **podman push** 操作を実行します。

重要

任意のイメージをプルできますが、**system:registry** ロールを追加している場合は、各自のプロジェクトにあるレジストリーにのみイメージをプッシュすることができます。

次の例では、以下を使用します。

コンポーネント	値
<registry_ip>	**172.30.124.220**
<port>	**5000**
<project>	**openshift**
<image>	**image**
<tag>	省略 (デフォルトは **latest**)

a. 任意のイメージをプルします。

```
sh-4.2# podman pull <name.io>/<image>
```

b. 新規イメージに **<registry_ip>:<port>/<project>/<image>** 形式でタグ付けします。プロジェクト名は、イメージを正しくレジストリーに配置し、これに後でアクセスできるようにするために OpenShift Container Platform のプル仕様に表示される必要があります。

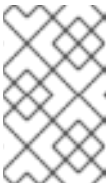
```
sh-4.2# podman tag <name.io>/<image> image-registry.openshift-image-registry.svc:5000/openshift/<image>
```

注記

指定されたプロジェクトについて **system:image-builder** ロールを持っている必要があります。このロールにより、ユーザーはイメージの書き出しやプッシュを実行できます。そうでない場合は、次の手順の **podman push** が失敗します。これをテストするには、新規プロジェクトを作成し、イメージをプッシュできます。

c. 新しくタグ付けされたイメージをレジストリーにプッシュします。

```
sh-4.2# podman push image-registry.openshift-image-registry.svc:5000/openshift/<image>
```



注記

イメージを内部レジストリーにプッシュする場合は、リポジトリー名に **<project>/<name>** 形式を使用する必要があります。リポジトリー名に複数のプロジェクトレベルを使用すると、認証エラーが発生します。

4.3. レジストリー POD のステータスの確認

クラスター管理者は、**openshift-image-registry** プロジェクトで実行されているイメージレジストリー Pod をリスト表示し、それらのステータスを確認できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **openshift-image-registry** プロジェクトの Pod をリスト表示し、それらのステータスを表示します。

```
$ oc get pods -n openshift-image-registry
```

出力例

```
NAME READY STATUS RESTARTS AGE
cluster-image-registry-operator-764bd7f846-qqtpb 1/1 Running 0 78m
image-registry-79fb4469f6-llrln 1/1 Running 0 77m
node-ca-hjksc 1/1 Running 0 73m
node-ca-tftj6 1/1 Running 0 77m
node-ca-wb6ht 1/1 Running 0 77m
node-ca-zvt9q 1/1 Running 0 74m
```

4.4. レジストリーログの表示

oc logs コマンドを使用してレジストリーのログを表示することができます。

手順

- デプロイメントで **oc logs** コマンドを使用して、コンテナーイメージレジストリーのログを表示します。

```
$ oc logs deployments/image-registry -n openshift-image-registry
```

出力例

```
2015-05-01T19:48:36.300593110Z time="2015-05-01T19:48:36Z" level=info
msg="version=v2.0.0+unknown"
2015-05-01T19:48:36.303294724Z time="2015-05-01T19:48:36Z" level=info msg="redis not
configured" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303422845Z time="2015-05-01T19:48:36Z" level=info msg="using
inmemory layerinfo cache" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
2015-05-01T19:48:36.303433991Z time="2015-05-01T19:48:36Z" level=info msg="Using
OpenShift Auth handler"
2015-05-01T19:48:36.303439084Z time="2015-05-01T19:48:36Z" level=info msg="listening
on :5000" instance.id=9ed6c43d-23ee-453f-9a4b-031fea646002
```

4.5. レジストリーメトリックへのアクセス

OpenShift Container レジストリーは、Prometheus メトリック のエンドポイントを提供します。Prometheus はスタンドアロンのオープンソースのシステムモニタリングおよびアラートツールキットです。

メトリックは、レジストリーエンドポイントの **/extensions/v2/metrics** パスに公開されます。

手順

クラスターロールを使用して、メトリクスクエリーを実行すると、メトリクスにアクセスできます。

クラスターロール

1. メトリクスにアクセスするために必要なクラスターロールがない場合は、これを作成します。

```
$ cat <<EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus-scraper
rules:
- apiGroups:
  - image.openshift.io
  resources:
  - registry/metrics
  verbs:
  - get
EOF
```

2. このロールをユーザーに追加し、以下のコマンドを実行します。

```
$ oc adm policy add-cluster-role-to-user prometheus-scraper <username>
```

メトリッククエリー

1. ユーザートークンを取得します。

```
openshift:
$ oc whoami -t
```

2. ノードまたは Pod 内でメトリクスクエリーを実行します。次に例を示します。

```
$ curl --insecure -s -u <user>:<secret> \ (1)
    https://image-registry.openshift-image-registry.svc:5000/extensions/v2/metrics | grep
imageregistry | head -n 20
```

出力例

```
# HELP imageregistry_build_info A metric with a constant '1' value labeled by major, minor,
git commit & git version from which the image registry was built.
# TYPE imageregistry_build_info gauge
imageregistry_build_info{gitCommit="9f72191",gitVersion="v3.11.0+9f72191-135-
dirty",major="3",minor="11+"} 1
# HELP imageregistry_digest_cache_requests_total Total number of requests without scope
to the digest cache.
# TYPE imageregistry_digest_cache_requests_total counter
imageregistry_digest_cache_requests_total{type="Hit"} 5
imageregistry_digest_cache_requests_total{type="Miss"} 24
# HELP imageregistry_digest_cache_scoped_requests_total Total number of scoped
requests to the digest cache.
# TYPE imageregistry_digest_cache_scoped_requests_total counter
imageregistry_digest_cache_scoped_requests_total{type="Hit"} 33
imageregistry_digest_cache_scoped_requests_total{type="Miss"} 44
# HELP imageregistry_http_in_flight_requests A gauge of requests currently being served by
the registry.
# TYPE imageregistry_http_in_flight_requests gauge
imageregistry_http_in_flight_requests 1
# HELP imageregistry_http_request_duration_seconds A histogram of latencies for requests
to the registry.
# TYPE imageregistry_http_request_duration_seconds summary
imageregistry_http_request_duration_seconds{method="get",quantile="0.5"} 0.01296087
imageregistry_http_request_duration_seconds{method="get",quantile="0.9"} 0.014847248
imageregistry_http_request_duration_seconds{method="get",quantile="0.99"} 0.015981195
imageregistry_http_request_duration_seconds_sum{method="get"} 12.260727916000022
```

(1) **<user>** オブジェクトは任意ですが、**<secret>** タグではユーザートークンを使用する必要があります。

4.6. 関連情報

- プロジェクトの Pod が別のプロジェクトのイメージを参照できるようにする方法の詳細は、Pod の複数のプロジェクト間でのイメージの参照を許可する を参照してください。
- **kubeadmin** は削除されるまでレジストリーにアクセスできます。詳細は、kubeadmin ユーザーの削除 を参照してください。

- アイデンティティープロバイダー設定の詳細は アイデンティティープロバイダー設定について を参照してください。

第5章 レジストリーの公開

デフォルトで、OpenShift イメージレジストリーのセキュリティーは、TLS 経由でトラフィックを送信できるようにクラスターのインストール時に保護されます。以前のバージョンの OpenShift Container Platform とは異なり、レジストリーはインストール時にクラスター外に公開されません。

5.1. デフォルトレジストリーの手動での公開

クラスター内からデフォルトの OpenShift イメージレジストリーにログインするのではなく、外部からレジストリーにアクセスできるように、このレジストリーをルートに公開します。この外部アクセスにより、ルートアドレスを使用してクラスターの外部からレジストリーにログインし、ルートホストを使用してイメージにタグを付けて既存のプロジェクトにプッシュできます。

前提条件

- 以下の前提条件が自動的に実行する。
  - Registry Operator をデプロイする。
  - Ingress Operator デプロイする。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

configs.imageregistry.operator.openshift.io リソースで **defaultRoute** パラメーターを使用してルートを公開できます。

defaultRoute を使用してレジストリーを公開するには、以下を実行します。

1. **defaultRoute** を **true** に設定します。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
{"defaultRoute":true}}' --type=merge
```

2. デフォルトのレジストリールートを取得します。

```
$ HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')
```

3. Ingress Operator の証明書を取得します。

```
$ oc get secret -n openshift-ingress  router-certs-default -o go-template='{{index .data
"tls.crt"}}' | base64 -d | sudo tee /etc/pki/ca-trust/source/anchors/${HOST}.crt  > /dev/null
```

4. 以下のコマンドを使用して、クラスターのデフォルト証明書がルートを信頼するようにします。

```
$ sudo update-ca-trust enable
```

5. デフォルトのルートを使用して podman にログインします。

```
$ sudo podman login -u kubeadmin -p $(oc whoami -t) $HOST
```

5.2. セキュアなレジストリーの手動による公開

クラスター内から OpenShift イメージレジストリーにログインするのではなく、外部からレジストリーにアクセスできるように、このレジストリーをルートに公開します。これにより、ルートアドレスを使用してクラスターの外部からレジストリーにログインし、ルートホストを使用してイメージにタグを付けて既存のプロジェクトにプッシュできます。

前提条件

- 以下の前提条件が自動的に実行する。
  - Registry Operator をデプロイする。
  - Ingress Operator デプロイする。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

configs.imageregistry.operator.openshift.io リソースで **DefaultRoute** パラメーターを使用するか、カスタムルートを使用してルートを公開できます。

DefaultRoute を使用してレジストリーを公開するには、以下を実行します。

1. **DefaultRoute** を **True** に設定します。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec":
{"defaultRoute":true}}' --type=merge
```

2. **podman** でログインします。

```
$ HOST=$(oc get route default-route -n openshift-image-registry --template='{{ .spec.host }}')
```

```
$ podman login -u kubeadmin -p $(oc whoami -t) --tls-verify=false $HOST (1)
```

(1) **--tls-verify=false** は、ルートのクラスターのデフォルト証明書が信頼されない場合に必要になります。Ingress Operator で、信頼されるカスタム証明書をデフォルト証明書として設定できます。

カスタムルートを使用してレジストリーを公開するには、以下を実行します。

1. ルートの TLS キーでシークレットを作成します。

```
$ oc create secret tls public-route-tls \
    -n openshift-image-registry \
    --cert=</path/to/tls.crt> \
    --key=</path/to/tls.key>
```

この手順はオプションです。シークレットを作成しない場合、ルートは Ingress Operator からデフォルトの TLS 設定を使用します。

2. Registry Operator では、以下のようになります。

```
$ oc edit configs.imageregistry.operator.openshift.io/cluster
```

```
spec:
  routes:
    - name: public-routes
      hostname: myregistry.mycorp.organization
      secretName: public-route-tls
...
```

注記

レジストリーのルートのカスタム TLS 設定を指定している場合は **secretName** のみ設定します。

トラブルシューティング

- Error creating TLS secret