Red Hat OpenShift GitOps 1.13

액세스 제어 및 사용자 관리

사용자 및 네임스페이스에 대한 사용자 인증 및 액세스 제어 구성

Last Updated: 2024-10-12

Red Hat OpenShift GitOps 1.13 액세스 제어 및 사용자 관리

사용자 및 네임스페이스에 대한 사용자 인증 및 액세스 제어 구성

법적 공지

초록

이 문서에서는 사용자 수준 액세스 및 리소스 요청을 변경 및 관리하는 방법을 설명합니다. 또한 여러 사용자, 권한, Argo CD 리소스 및 클러스터의 인스턴스를 관리하도록 역할 기반 액세스 제어 및 Single Sign-On 인증 공급자를 구성하는 방법도 설명합니다.

차례

1장. ARGO CD RBAC 구성

기본적으로 기본 Argo CD 인스턴스에 로그인한 **kube:admin** 사용자를 제외한 모든 유형의 사용자는 서비스에 액세스할 수 없습니다. 그러나 사용자 지정 Argo CD 인스턴스에 로그인한 사용자는 기본적으로 읽기 전용 사용자입니다.

참고

Red Hat OpenShift GitOps v1.9.0 또는 이전 버전에서는 **kube:admin** 사용자를 제외한 모든 유형의 사용자(RH SSO)를 사용하여 Argo CD에 로그인되어 기본적으로 읽기 전용 사용자입니다.

1.1. 사용자 수준 액세스 구성

사용자 수준 액세스를 관리하고 수정하려면 Argo CD CR(사용자 정의 리소스)에서 RBAC(역할 기반 액세스 제어) 섹션을 구성합니다.

프로세스

1. **argocd** CR을 편집합니다.

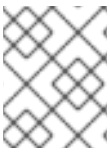
```
$ oc edit argocd [argocd-instance-name] -n [namespace]
```

출력 결과

```
metadata
...
...
  rbac:
    policy: 'g, rbacsystem:cluster-admins, role:admin'
    scopes: '[groups]'
```

2. **rbac** 섹션에 **정책** 구성을 추가하고 사용자의 **이름,이메일** 및 **역할을** 추가합니다.

```
metadata
...
...
rbac:
    policy: <name>, <email>, role:<admin>
    scopes: '[groups]'
```

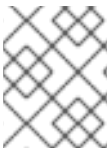


참고

현재 RHSSO는 Red Hat OpenShift GitOps 사용자의 그룹 정보를 읽을 수 없습니다. 따라서 사용자 수준에서 RBAC를 구성합니다.

2장. DEX를 사용하여 ARGO CD에 대한 SSO 구성

Red Hat OpenShift GitOps Operator가 설치되면 Argo CD에서 **admin** 권한이 있는 사용자를 자동으로 생성합니다. 클러스터 관리자는 여러 사용자를 관리하기 위해 Argo CD를 사용하여 SSO(Single Sign-On)를 구성할 수 있습니다.



참고

ArgoCD CR의 **spec.dex** 매개변수는 Red Hat OpenShift GitOps v1.10.0 이상에서 더 이상 지원되지 않습니다. 대신 **.spec.sso** 매개변수를 사용하는 것이 좋습니다.

2.1. OPENSHIFT OAUTH 커넥터를 활성화하는 구성

DEX는 Operator에서 생성한 모든 Argo CD 인스턴스에 대해 기본적으로 설치됩니다. **.spec.sso** 매개변수를 설정하여 Dex를 SSO 인증 공급자로 사용하도록 Red Hat OpenShift GitOps를 구성할 수 있습니다.

DEX는 플랫폼에서 제공하는 **OAuth** 서버를 확인하여 OpenShift Container Platform 내에 정의된 사용자와 그룹을 사용합니다.

프로세스

- Dex를 활성화하려면 Operator의 YAML 리소스에서 **.spec.sso.provider** 매개변수를 **dex** 로 설정합니다.

```
# ...
spec:
  sso:
    provider: dex
    dex:
      openShiftOAuth: true (1)
# ...
```

1. **openShiftOAuth** 속성을 사용하면 값이 **true** 로 설정된 경우 Operator에서 기본 제공 OpenShift Container Platform **OAuth** 서버를 자동으로 구성합니다.

2.1.1. 사용자를 특정 역할에 매핑

Argo CD는 직접 **ClusterRoleBinding** 역할이 있는 경우 사용자를 특정 역할에 매핑할 수 없습니다. OpenShift를 통해 SSO에서 **role:admin** 으로 역할을 수동으로 변경할 수 있습니다.

프로세스

1. **cluster-admins** 라는 그룹을 생성합니다.

```
$ oc adm groups new cluster-admins
```

2. 사용자를 그룹에 추가합니다.

```
$ oc adm groups add-users cluster-admins USER
```

3. **cluster-admin ClusterRole** 을 그룹에 적용합니다.

```
$ oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
```

2.2. .SPEC.SSO를 교체하여 DEX 비활성화

- dex를 비활성화하려면 Argo CD 사용자 정의 리소스에서 **spec.sso** 요소를 제거하거나 다른 SSO 공급자를 지정합니다.

3장. KEYCLOAK을 사용하여 ARGO CD에 대한 SSO 구성

Red Hat OpenShift GitOps Operator가 설치되면 Argo CD에서 **admin** 권한이 있는 사용자를 자동으로 생성합니다. 클러스터 관리자는 여러 사용자를 관리하기 위해 Argo CD를 사용하여 SSO(Single Sign-On)를 구성할 수 있습니다.

3.1. 사전 요구 사항

- Red Hat SSO가 클러스터에 설치되어 있습니다.
- Red Hat OpenShift GitOps Operator가 클러스터에 설치되어 있습니다.
- Argo CD가 클러스터에 설치되어 있습니다.
- 클러스터에서 **DeploymentConfig** API를 사용할 수 있습니다. 자세한 내용은 "DeploymentConfig [apps.openshift.io/v1]"를 참조하십시오.

3.2. KEYCLOAK에서 새 클라이언트 구성

DEX는 Operator에서 생성한 모든 Argo CD 인스턴스에 대해 기본적으로 설치됩니다. 그러나 Dex 구성을 삭제하고 Keycloak을 추가하여 OpenShift 인증 정보를 사용하여 Argo CD에 로그인할 수 있습니다. Keycloak은 Argo CD와 OpenShift 간의 ID 브로커 역할을 합니다.

프로세스

Keycloak을 구성하려면 다음 단계를 따르십시오.

1. Argo CD CR(사용자 정의 리소스)에서 **.spec.sso.dex** 매개변수를 제거하고 CR을 저장하여 Dex 구성을 삭제합니다.

```
dex:
   openShiftOAuth: true
   resources:
    limits:
      cpu:
      memory:
    requests:
      cpu:
      memory:
```

2. Argo CD CR에서 **provider** 매개변수의 값을 **keycloak** 으로 설정합니다.
3. 다음 단계 중 하나를 수행하여 Keycloak을 구성합니다.
   - 보안 연결의 경우 다음 예와 같이 **rootCA** 매개변수 값을 설정합니다.

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  sso:
    provider: keycloak
```

```
    keycloak:
      rootCA: "<PEM-encoded-root-certificate>" (1)
  server:
    route:
      enabled: true
```

(1) Keycloak의 TLS 인증서를 확인하는 데 사용되는 사용자 정의 인증서입니다.

Operator는 **.spec.sso.keycloak.rootCA** 매개변수의 변경 사항을 조정하고 **argocd-cm** 구성 맵에서 PEM 인코딩 루트 인증서로 **oidc.config** 매개변수를 업데이트합니다.

- 비보안 연결의 경우 **rootCA** 매개변수 값을 비워 두고 아래와 같이 **oidc.tls.insecure.skip.verify** 매개변수를 사용합니다.

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  extraConfig:
    oidc.tls.insecure.skip.verify: "true"
  sso:
    provider: keycloak
    keycloak:
      rootCA: ""
```

- 선택 사항: **spec.sso.keycloak** 필드를 사용자 지정하여 **ArgoCD** CR에서 **keycloak** 공급자의 경로 이름을 추가합니다. 이 기능을 사용하여 여러 Ingress 컨트롤러 shard 간에 들어오는 트래픽 로드 밸런싱과 같은 고급 라우팅 사용 사례를 지원합니다.
  - 다음 예제 YAML을 사용하여 **ArgoCD** CR에 **호스트** 매개변수를 추가합니다.

ArgoCD CR의 예

```
apiVersion: argoproj.io/v1alpha1
kind: ArgoCD
metadata:
  name: <resource_name> (1)
  labels:
    example: route
spec:
  sso:
    provider: keycloak
    keycloak:
      host: <hostname> (2)
  server:
    ingress:
      enabled: true
    insecure: true
```

(1) & **lt;resource_name&** gt;을 **ArgoCD** CR의 이름으로 바꿉니다.

2 & **lt;hostname** >을 호스트 키의 이름으로 바꿉니다(예: **sso.test.example.com**).

- **ArgoCD CR** 을 생성하려면 다음 명령을 실행합니다.

```
$ oc create -f <argocd_filename>.yaml -n <your-namespace>
```

- **ArgoCD CR** 을 편집하려면 다음 명령을 실행합니다.

```
$ oc edit -f <argocd_filename>.yaml -n <your_namespace>
```

- 파일을 저장하여 변경 사항을 적용합니다.
- **ArgoCD** CR을 적용하려면 다음 명령을 실행합니다.

```
$ oc apply -f <argocd_filename>.yaml -n <your_namespace>
```

- 다음 명령을 실행하여 **host** 속성이 추가되었는지 확인합니다.

```
$ oc get route keycloak -n <your_namespace> -o yaml
```

출력 예

```
kind: Route
metadata:
  name: keycloak 1
  labels:
    application: keycloak
spec:
  host: sso.test.example.com
status:
  ingress:
    - host: sso.test.example.com 2
```

1 경로 이름을 지정합니다.

2 호스트 키의 이름을 지정합니다.

참고

Keycloak 인스턴스를 설치하고 실행하는 데 2-3분이 걸립니다.

3.3. KEYCLOAK에 로그인

Keycloak 콘솔에 로그인하여 ID 또는 역할을 관리하고 다양한 역할에 할당된 권한을 정의합니다.

사전 요구 사항

- Dex의 기본 구성이 제거됩니다.
- Keycloak SSO 공급자를 사용하도록 Argo CD CR을 구성해야 합니다.

프로세스

1. 로그인할 Keycloak 경로 URL을 가져옵니다.

```
$ oc -n argocd get route keycloak

NAME       HOST/PORT                                                PATH   SERVICES   PORT
TERMINATION   WILDCARD
keycloak   keycloak-default.apps.ci-ln-******.origin-ci-int-aws.dev.**.com          keycloak   <all>
reencrypt     None
```

2. 사용자 이름과 암호를 환경 변수로 저장하는 Keycloak Pod 이름을 가져옵니다.

```
$ oc -n argocd get pods

NAME                  READY   STATUS         RESTARTS   AGE
keycloak-1-2sjcl          1/1     Running          0          45m
```

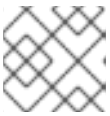
   a. Keycloak 사용자 이름을 가져옵니다.

```
$ oc -n argocd exec keycloak-1-2sjcl -- "env" | grep SSO_ADMIN_USERNAME

SSO_ADMIN_USERNAME=Cqid54Ih
```

   b. Keycloak 암호를 가져옵니다.

```
$ oc -n argocd exec keycloak-1-2sjcl -- "env" | grep SSO_ADMIN_PASSWORD

SSO_ADMIN_PASSWORD=GVXxHifH
```

3. 로그인 페이지에서 로그인 **VIA KEYCLOAK** 를 클릭합니다.



참고

Keycloak 인스턴스가 준비된 후에만 **LOGIN VIA KEYCLOAK** 옵션이 표시됩니다.

4. **OpenShift로 로그인을** 클릭합니다.

참고

kubeadmin 을 사용한 로그인은 지원되지 않습니다.

5. 로그인할 OpenShift 자격 증명을 입력합니다.

6. 선택 사항: 기본적으로 Argo CD에 로그인한 모든 사용자는 읽기 전용 액세스 권한이 있습니다. **argocd-rbac-cm** 구성 맵을 업데이트하여 사용자 수준 액세스를 관리할 수 있습니다.

```
policy.csv:
<name>, <email>, role:admin
```

3.4. KEYCLOAK 설치 제거

Argo CD CR(사용자 정의 리소스) 파일에서 **SSO** 필드를 제거하여 Keycloak 리소스 및 관련 구성을 삭제할 수 있습니다. **SSO** 필드를 제거한 후 파일의 값은 다음과 유사합니다.

```
apiVersion: argoproj.io/v1beta1
kind: ArgoCD
metadata:
  name: example-argocd
  labels:
    example: basic
spec:
  server:
    route:
     enabled: true
```

참고

이 방법을 사용하여 생성된 Keycloak 애플리케이션은 현재 영구적이지 않습니다. Argo CD Keycloak 영역에서 생성된 추가 구성은 서버가 다시 시작되면 삭제됩니다.

3.5. KEYCLOAK 리소스 요청/제한 수정

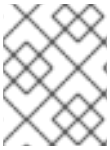
기본적으로 Keycloak 컨테이너는 리소스 요청 및 제한을 사용하여 생성됩니다. 리소스 요청을 변경하고 관리할 수 있습니다.

리소스	요구 사항	제한
CPU	500	1000m
메모리	512 Mi	1024 Mi

프로세스

- Argo CD CR(사용자 정의 리소스)을 패치하는 기본 리소스 요구 사항을 수정합니다.

```
$ oc -n openshift-gitops patch argocd openshift-gitops --type='json' -p='[{"op": "add", "path": "/spec/sso", "value": {"provider": "keycloak", "resources": {"requests": {"cpu": "512m", "memory": "512Mi"}, "limits": {"cpu": "1024m", "memory": "1024Mi"}} }}]'
```



참고

Red Hat OpenShift GitOps가 생성한 Keycloak은 Operator가 변경한 사항만 유지합니다. Keycloak이 다시 시작되면 Keycloak의 관리자가 생성한 추가 구성이 삭제됩니다.