Using Private Repositories and Registries

Using Private Repositories and Registries

WARNING

This chapter build is in progress, expect to change and not working as expected

At the end of this chapter you will be able to :

- Tekton Authentication Secrets
- How to push linux contianer image to external registry
- How to clone from a private Github respository

Overview

In many parctical usecase you might need to pull from private Git repsoitories or might need to push to external container registry such as Quay.io. In both the cases the access requires you to authenticate. With Tekton this is achieved using the Kubernetes Service Account(SA) and Kubernetes Secret.

If you are not in tutorial chapter folder, then navigate to the folder:

```
cd $TUTORIAL_HOME/private_repos_reg
```

Let us create a new namespace called `workspace-auth-demo` and switch to that namespace context.

ASSUMPTIONS

- It is assumed that you have completed Workspaces and the reusing the same namespace that was prepared in that chapter.
- You have container registry account with Quay.io or Docker Hub
- You have GitHub account and a private repository

Pulling from Private Source Repository

In this exercise we will see how to pull i.e. clone the source code from the private source reporsitory. For this example we will use the GitHub as the remote source repository.

To be able to run this exercise you need:

- A remote GitHub Repository, you can pull fork the https://github.com/redhat-scholars/tekton-tutorial-greeter and push that as your private repository
- A GitHub Personal Access Token(PAT) to access the remote repository.

Set the requried environment variables to be used when creating the github-pat-secret:

```
export GITHUB_USERNAME='<your github.com username>'
export TEKTON_TUTORIAL_GITHUB_PAT='<your github.com username personal accessto
```

Create Github PAT Secret

Create the Kubernetes secret that can hold your GitHub.com credentials:

```
kubectl create secret -n workspace-auth-demo generic github-pat-secret --dr
  | yq w - type kubernetes.io/basic-auth \
  | yq w - stringData.username $GITHUB_USERNAME \
  | yq w - stringData.password $TEKTON_TUTORIAL_GITHUB_PAT \
  | kubectl apply -f -
```

```
secret/github-pat-secret created
```

Annotate Secret to be used with GitHub.com

To make Tekton use the Secret `github-pat-secret` with github.com, we need to annotate the Secret with `tekton.dev/git-0=https://github.com`.

```
kubectl annotate -n workspace-auth-demo secret github-pat-secret \
  "tekton.dev/git-0=https://github.com"
```

```
secret/github-pat-secret annotated
```

Verify the github-pat-secret as right type, annoations and credential values:

```
kubectl get -n workspace-auth-demo secret github-pat-secret -o yaml
```

The command should show an output(trimmed for brevity) as shown in the following listing, with your GitHub.com username and PAT.

```
apiVersion: v1
data:
  password: REDACTED
  username: REDACTED
kind: Secret
metadata:
  annotations:
    tekton.dev/git-0: https://github.com
  name: github-pat-secret
  namespace: workspace-auth-demo
type: kubernetes.io/basic-auth
```

Create Service Account

Let us create a Kubernetes ServiceAccount that could be used pull from the private GitHub repository:

```
kubectl create sa -n workspace-auth-demo github-bot
```

```
serviceaccount/github-bot created
```

Patch Service Account

Now patch the `github-bot` service account to use the `github-pat-secret` credentials:

```
kubectl patch serviceaccount github-bot \
  -p '{"secrets": [{"name": "github-pat-secret"}]}'
```

```
serviceaccount/github-bot patched
```

Lets verify if the service account has the secret added:

```
kubectl get sa -n workspace-auth-demo github-bot -o yaml
```

The command should show an output(trimmed for brevity) like:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: github-bot
  namespace: workspace-auth-demo
secrets:
- name: github-pat-secret
- name: build-bot-token-8nl2v
```

Create Pipeline

Create the git clone pipeline which will clone from the private GitHub repo and simply list the contents:

```
kubectl apply -n workspace-auth-demo -f secretworld-app-clone.yaml
```

```
pipeline.tekton.dev/secretworld-app-clone created
```

Run Pipeline

tkn | kubectl

```
tkn pipeline start secretworld-app-clone \
  --namespace=workspace-auth-demo \
  --serviceaccount=github-bot \
  --param private-github-repo-url='https://github.com/redhat-scholars/tek
  --workspace name=source,claimName=tekton-tutorial-sources \
  --use-param-defaults \
  --showlog
```

A successful clone from private repo will show the following output (trimmed brevity):

```
[clone-sources : credential-initializer] {"level":"info","ts":1595922122.90779

[clone-sources : clone] + CHECKOUT_DIR=/workspace/output/
[clone-sources : clone] + '[[' true '==' true ]]
[clone-sources : clone] + cleandir
[clone-sources : clone] + '[[' -d /workspace/output/ ]]
[clone-sources : clone] + rm -rf /workspace/output//Dockerfile /workspace/outp
[clone-sources : clone] + rm -rf /workspace/output//.dockerignore /workspace/o
[clone-sources : clone] + rm -rf '/workspace/output//..?*'
[clone-sources : clone] + test -z
[clone-sources : clone] + test -z
[clone-sources : clone] + test -z
[clone-sources : clone] + /ko-app/git-init -url https://github.com/redhat-scho
[clone-sources : clone] {"level":"info","ts":1595922137.4565356,"caller":"git/
[clone-sources : clone] {"level":"info","ts":1595922137.4990256,"caller":"git/
[clone-sources : clone] + cd /workspace/output/
[clone-sources : clone] + git+ tr -d '\n'
[clone-sources : clone]  rev-parse HEAD
[clone-sources : clone] + RESULT_SHA=5250e1fa185805373e620d1c04a0c48129efd2ee
[clone-sources : clone] + EXIT_CODE=0
[clone-sources : clone] + '[' 0 '!=' 0 ]
[clone-sources : clone] + echo -n 5250e1fa185805373e620d1c04a0c48129efd2ee

[list-cloned-repo : credential-initializer] {"level":"info","ts":1595922139.78


[list-cloned-repo : list-directory] total 44
[list-cloned-repo : list-directory] drwxr-xr-x    4 root     root          409
[list-cloned-repo : list-directory] -rw-r--r--    1 root     root          414
[list-cloned-repo : list-directory] -rwxr-xr-x    1 root     root          660
[list-cloned-repo : list-directory] -rwxr-xr-x    1 root     root         1006
[list-cloned-repo : list-directory] drwxr-xr-x    2 root     root          409
[list-cloned-repo : list-directory] -rw-r--r--    1 root     root           11
[list-cloned-repo : list-directory] -rw-r--r--    1 root     root            8

[list-cloned-repo : show-readme] ? Yay! ?
[list-cloned-repo : show-readme]
[list-cloned-repo : show-readme] You have successfully cloned from private Git
[list-cloned-repo : show-readme]
[list-cloned-repo : show-readme] ? Tekton Rocks!! ?
```

Pushing to external registry

To able push to external container registry, its requried that the Pipline is run with a ServiceAccount that has container registry credentials configured via ServiceAccount secrets. Kubernetes provider a Secret type called `docker-registry`, that can be used to configure the container registry credentials.

Set the requried environment variables to be used when creating the container-registry-secret:

```
export CONTAINER_REGISTRY_SERVER='https://quay.io' ①
export CONTAINER_REGISTRY_USER='<your registry user>'
export CONTAINER_REGISTRY_PASSWORD='<your registry user password>'
```

① The container registry server URL, for Quay.io its https://quay.io and for DockerHub it is https://index.docker.io/v2/

Create Container Registry Secret

```
kubectl create secret -n workspace-auth-demo docker-registry container-regi
  --docker-server=$CONTAINER_REGISTRY_SERVER \
  --docker-username=$CONTAINER_REGISTRY_USER \
  --docker-password=$CONTAINER_REGISTRY_PASSWORD
```

```
secret/container-registry-secret created
```

Create Service Account

```
kubectl create sa -n workspace-auth-demo build-bot
```

```
serviceaccount/build-bot created
```

Patch Service Account

Now patch the `build-bot` service account to use the `container-registry-secret` credentials:

```
kubectl patch serviceaccount build-bot \
  -p '{"secrets": [{"name": "container-registry-secret"}]}'
```

```
serviceaccount/build-bot patched
```

Lets verify if the service account has the secret added:

```
kubectl get sa -n workspace-auth-demo build-bot -o yaml
```

The command should show an output like:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2020-07-28T03:34:32Z"
  name: build-bot
  namespace: auth-demo
  resourceVersion: "53879"
  selfLink: /api/v1/namespaces/auth-demo/serviceaccounts/build-bot
  uid: 628067fd-91d1-4cdd-b6a6-88b4f7280ff0
secrets:
- name: container-registry-secret
- name: build-bot-token-8nl2v
```

Create Pipeline

Create the build and push app pipeline:

```
kubectl apply -n workspace-auth-demo \
  -f greeter-app-build.yaml
```

```
pipeline.tekton.dev/greeter-app-build created
```

Run Pipeline

tkn | kubectl

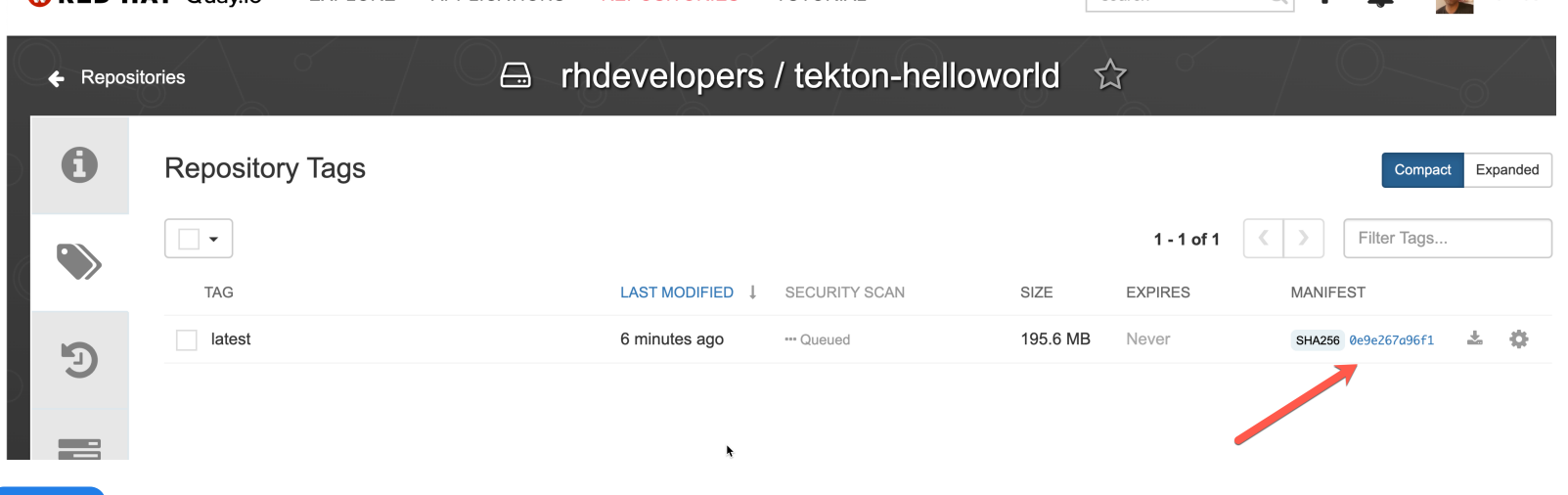
```
tkn pipeline start greeter-app-build \
  --namespace=workspace-auth-demo \
  --serviceaccount=build-bot \
  --workspace name=maven-settings,config=maven-settings \
  --workspace name=source,claimName=tekton-tutorial-sources \
  --use-param-defaults \
  --showlog
```

A successful build and push will show the following output (trimmed brevity):

```
...
[build-java-app-image : push] + buildah --storage-driver=overlay push --tls-ve
[build-java-app-image : push] Getting image source signatures
[build-java-app-image : push] Copying blob sha256:90c2e42f948b524cf98005073e0b
[build-java-app-image : push] Copying blob sha256:869b43e5dca37fa63d84e9bc5886
[build-java-app-image : push] Copying blob sha256:f9ddbcc4e7954a705b700c35c5e5
[build-java-app-image : push] Copying blob sha256:7b08010864ba4c7ce9dfe1b90244
[build-java-app-image : push] Copying config sha256:5bd61d725dc47d1f8b7c225d8d
[build-java-app-image : push] Writing manifest to image destination
[build-java-app-image : push] Copying config sha256:5bd61d725dc47d1f8b7c225d8d
[build-java-app-image : push] Writing manifest to image destination
[build-java-app-image : push] Storing signatures

[build-java-app-image : digest-to-results] + cat /workspace/source/image-diges
[build-java-app-image : digest-to-results] + tee /tekton/results/IMAGE_DIGEST
[build-java-app-image : digest-to-results] sha256:0e9e267a96f1ea48fe00642cd82e
```

A successful pipeline should have pushed the image to the external container registry. The following screen shot shows the image pushed to **rhdevelopers** Quay.io container registry repo:



NOTE

If you noticed the highlighted **sha256** from the log above, is same as that of the **sha256** listed in the contianer registry repository.

Points to Ponder

When need to pull from remote source repository or push to external container registry:

- A Kubernetes Secret to hold the container registry credentials
- A Kubenretes Service Account, with the container registry Secret added to it
- Use the Service Account as `serviceAccountName` in PipelineRuns/TaskRuns
- Annotate Secerts to map which Secret to be used with source repository

You can find more details about using Authentication with Tekton here.

Cleanup

Delete the workspace and its resources:

```
kubectl delete ns workspace-auth-demo
```