The Project Provisioner

(Documented by Simon Reap <[sreap@redhat.com](mailto:sreap@redhat.com)> 18th December 2019)

(Updated 23rd January 2020)

# Description

The “Project provisioner” is code (Written by Alexander) to create new projects and give specific permissions to various OpenShift groups. It is written in Ansible, so uses the ose-ansible-operator base image to create the POD. It uses a YAML configuration file to define the groups and permissions to be created or deleted.

As described below, the code is in <https://xescm.xeop.de/kv082/project-provisioner.git> and the configuraiton in <https://xescm.xeop.de/kv082/provision-test.git>

# Prepare code

## Get the code repository

First clone https://github.com/Havilland/project-provisioner.git to a publically-accessible git repository. The best option might be on xescm.xeop.de, so that "public" is only within Deutsche Boerse. This repository contains the code to build the project-provisioner image, as a series of OpenShift YAML files.

To clone the provisioner to xescm:

- In the https://xescm.xeop.de GUI, create a publically accessible GIT repo called "project-provisioner", but don't add a README.md at this point

- e.g. https://xescm.xeop.de/kv082/project-provisioner

- Ensure that your SSH key is on xescm, so that you can push code directly

- git clone https://github.com/Havilland/project-provisioner.git

- cd project-provisioner

- git branch -a

- Check that the "master" branch has an asterisk next to it, indicating it is the active branch

- git fetch --tags

- git tag

- Check that any expected tags are there - there may not be any

- git branch -a

- Check that master and remotes/origin/master are there

- git checkout master

- Just to make sure - it should already be checked out

- git remote rm origin

- git remote add origin git@xescm.xeop.de:kv082/project-provisioner.git

- git push origin --all

- git push --tags

If you now go back to the xescm GUI and refresh, you should see all of the files.

Select Settings/General on the left-hand menu, and click "Expand" on "visibility, project features, permissions". Make sure that "Project visibility" is "Public".

## Prepare the configuration file

Similarly, create a publically-accessible repository called "provision-test", and select the option to create a README.md (so that you can clone it).

- git clone git@xescm.xeop.de:kv082/provision-test.git

- cd provision-test

- Create a file with the name projects.yaml, with contents describing the projects to be created on the cluster, and the permissions to be applied to those projects. For example, the configuration below creates the "hedgehog-acceptance" project, assigns group-specific permissions to administrate, view and edit it, and also allows kv082 to administer the project.

---

projects:

- name: hedgehog-acceptance

owner: pc543

ownerMail: "jens.kuehnel@deutsche-boerse.com"

access:

- role: admin

subjectName: hedgehog-act-admin

kind: Group

- role: edit

subjectName: hedgehog-act-deployer

kind: Group

- role: view

subjectName: hedgehog-act-operator

kind: Group

- role: admin

subjectName: kv082

kind: User

## Update the Project Provisioner code

Now we edit the project-provisioner repository to reflect various locations:

* In build/buildconfig/enterprise.yaml, update the "uri" field (line 25) to point back to this project-provisioner repository. E.g.
  + uri: '<https://xescm.xeop.de/kv082/project-provisioner.git>'
* In deploy/role\_binding.yaml there are 2 "namespace:" entries with no text - on each, add "project-provisioner". E.g:
  + namespace: project-provisioner
* In deploy/operator-enterprise.yaml, update the name of the image to use. In the original, it uses the "openshift" namespace, and specifies a particular version. E.g.
  + image: "docker-registry.default.svc:5000/project-provisioner/project-provisioner:latest"
* In deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml (note the "\_cr.yaml" at the end, not the "\_crd.yaml" file) we set the "project\_definition\_git" variable to be the "provision-test" repository we created above. E.g.
  + project\_definition\_git: "https://gitlab.com/kv082/provision-test.git"

There is also a bug in the code (it can’t delete empty projects). To fix, edit deploy/role.yaml and add:

- delete

(the leading hyphen is important) to the verbs in the “resources: -namespaces” section (about line 54).

Now “git add” the changed files, “git commit” and “git push” them

* git add build/buildconfig/enterprise.yaml
* git add deploy/role\_binding.yaml
* git add deploy/operator-enterprise.yaml
* git add deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml
* git add deploy/role.yaml
* git commit -m"Update files to point to the correct repositories"
* git push

That completes the edits on the repositories.

# Install the Project Provisioner

Now log on to a machine on which:

* you can "oc login" to the OpenShift Cluster
* you can run the "skopeo" command
* you can clone the "https://xescm.xeop.de/kv082/project-provisioner.git" repository.

Cloned the “project-provisioner” repository, and be in that "project-provisioner" directory.

You will need a login to the Red Hat Customer Portal, to allow access to the image we are trying to copy - below this is given as "user@example.com" with the password "passw0rd".

$ export HTTPS\_PROXY=http://webproxy.deutsche-boerse.de:8080

$ export NO\_PROXY=".svc,xeop.de"

$ oc login https://osr3-cluster.xeop.de:8443

If you don't already have an externally visible endpoint for the internal registry, create one. If you don't do this, you will need to run "skopeo" on a machine which does have access to the internal docker-registry.cluster.svc:5000 address. Run oc get routes -n default and, if docker-registry is not listed, run:

$ oc expose service docker-registry -n default

In the following command, ‘user@example.com:password’ is the username and password for an account with access to Red Hat Customer Portal:

$ skopeo copy docker://registry.redhat.io/openshift4/ose-ansible-operator:v4.1 docker://docker-registry-default.osr3.xeop.de/openshift/ose-ansible-operator:latest --dest-creds $(oc whoami):$(oc whoami -t) --src-creds 'user@example.com:passw0rd' --dest-tls-verify=false

We now set up the new project-provisioner project to hold the provisioner code

$ oc new-project project-provisioner

$ grep "uri:" build/buildconfig/enterprise.yaml

uri: 'https://xescm.xeop.de/kv082/project-provisioner.git'

$ oc create -f build/buildconfig/enterprise.yaml

imagestream.image.openshift.io/project-provisioner created

buildconfig.build.openshift.io/project-provisioner created

$ oc create -f deploy/service\_account.yaml

serviceaccount/project-provisioner created

$ oc create -f deploy/role.yaml

clusterrole.rbac.authorization.k8s.io/project-provisioner created

$ oc create -f deploy/role\_binding.yaml

clusterrolebinding.rbac.authorization.k8s.io/project-provisioner created

clusterrolebinding.rbac.authorization.k8s.io/project-self-provisioner created

$ grep "image:" deploy/operator-enterprise.yaml

image: "docker-registry.default.svc:5000/project-provisioner/project-provisioner:latest"

$ oc create -f deploy/operator-enterprise.yaml

deployment.apps/project-provisioner created

$ oc create -f deploy/crds/projectprovisioner\_v1\_projectprovisioner\_crd.yaml

customresourcedefinition.apiextensions.k8s.io/projectprovisioners.project-provisioner.io created

At this point, check the "Builds" entry in the “Builds” section on the OpenShift console, select the “project-provisioner” project from the dropdown at the top of the page, and verify that "project-provisioner-n" (where “n” is a number) is there. If not, select the “Build Configs” entry, click on the gearwheel next to “project-provisioner” and select “Start Build”. Go back to the “Builds” entry, and you should see the build start and complete. If the build doesn’t work, first check the "Events" to see whether any are reported and fix any problems, and check the "Logs" for errors. One possible problem is access permissions on the code repository - this should be Public.

# Run the Project Provisioner

If all is well, and the project-provisioner/project-provisioner image has been built, start the Operator

$ grep "project\_def" deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml

project\_definition\_git: "https://xescm.xeop.de/kv082/provision-test.git"

$ oc create -f deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml

projectprovisioner.project-provisioner.io/example-projectprovisioner created

Now look at the OpenShift console to see any new projects being created.

If nothing seems to be happening, click on "Pods" on the left-hand menu and select the "project-provisioner" project. Click the small gear wheel to the left of the "project-provisioner-xxxxxx-yyyy" name, and click “Delete Pod”. The POD will be deleted and automatically recreated.

To stop the Project Provisioner consuming a lot of CPU, delete the Custom Resource once all of the required changes have been made to projects.

$ oc delete -f deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml

# Verify the changes

To check whether the changes have been applied, list the rolebindings in the target projects, and compare against the list in the above configuration file. For example, to check the settings in the hedgehog-acceptance project. Some accounts created automatically have been omitted from the text below as it confused the output.

$ oc get rolebindings -n hedgehog-acceptance

NAME ROLE USERS GROUPS

admin /admin system:serv...

hedgehog-act-admin-admin /admin hedgehog-act-admin

hedgehog-act-deployer-edit /edit hedgehog-act-deployer

hedgehog-act-operator-view /view hedgehog-act-operator

kv082-admin /admin kv082

system:image-pullers /system:image-puller

# Update the projects

If changes are made to the provision-test configuration file repository to add a new project, delete an old one, or change the permissions for an existing one, create the Custom Resource again, and delete it after the changes have been made:

$ oc create -f deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml

Wait a few minutes.

$ oc delete -f deploy/crds/projectprovisioner\_v1\_projectprovisioner\_cr.yaml

# Update the Provisioner code

If you make changes to the Project Provisioner code, you will need to rebuild the POD, and redeploy it. See above for instructions on how to start a new build, and how to delete the running POD so that it restarts with the newly built image.