Pipelines

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

- Understand what is a Pipeline ?
- Add Tasks from Catalog
- Create a Pipeline
- Execute a Pipeline to build and deploy a Knative service

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

```
cd $TUTORIAL_HOME/pipelines
```

Ensure the pipeline Resources and Tasks are available:

```
tkn res ls
```

The command above should show an output like:

```
NAME                          TYPE    DETAILS
git-source                    git     url: https://github.com/redhat-scholar
tekton-tutorial-greeter-image  image   url: example.com/rhdevelopers/tekton-t
```

```
tkn task ls
```

The command should show a output like:

```
NAME            AGE
build-app       2 hours ago
source-lister   7 seconds ago
```

If you don't see the output as above please ensure you have completed all the exercises of Chapter 2 and Chapter 3 before proceeding further.

Add Tasks from catalog

The Tekton Pipelines catalog allows you to reuse the catalog from community repositories. Here is list of repositories which from where you can add tasks:

- Tekton Pipelines Catalog
- OpenShift Pipelines Catalog

Since there is no `kubectl` task available in Tekton Pipelines Catalog repository, we can use the OpenShift client task to deploy the app:

```
kubectl create -n tektontutorial \
  -f https://raw.githubusercontent.com/tektoncd/catalog/master/task/openshift-
```

Check the created tasks using the command:

```
tkn task ls
```

The Task list should now list the following two Tasks:

```
NAME              AGE
build-app         2 hours ago
openshift-client  3 seconds ago
source-lister     10 minutes ago
```

Create a Pipeline

Using a Pipeline we can run multiple Task together in a user defined sequence or order.

Let us use the `build-app` task that we created in previous chapter and `openshift-client` task that we deployed in previous step to make Pipeline that will build the application from sources and deploy the built linux container image.

The following snippet shows what a Tekton Pipeline YAML looks like:

svc-deploy.yaml

```
apiVersion: tekton.dev/v1beta1
kind: Pipeline
metadata:
  name: svc-deploy
spec:
  params:
    - name: contextDir
      description: the context directory from where to build the application
  resources:
    - name: appSource
      type: git
    - name: appImage
      type: image
  tasks:
    - name: build-java-app
      taskRef:
        name: build-app
      params:
        - name: contextDir
          value: $(params.contextDir)
      resources:
        inputs:
          - name: source
            resource: appSource
        outputs:
          - name: builtImage
            resource: appImage
    - name: deploy-kubernetes-service
      taskRef:
        name: openshift-client
      runAfter:
        - build-java-app
      resources:
        inputs:
          - name: source
            resource: appSource
      params:
        - name: ARGS
          value:
            - "apply"
            - "-f"
            - "/workspace/source/k8s/deployment.yaml"
            - "-f"
            - "/workspace/source/k8s/service.yaml"
```

Each Pipeline has the following:

- **name** - the unique name using which the Pipeline can be referred
  - **resources** - the pipeline resources that will be used in the Pipeline e.g. appImage, appSource
    - **name** - the name of the input resource using which it can be referenced and bound via Run Pipeline
    - **type** - the type of the input resource, typically the pipeline resource type

tasks has one or more Tasks that needs to be executed as part of the Pipeline. In this example we have two Tasks `build-java-app` and `deploy-kn-service` that will be run to build the application from sources and deploy the built linux container image as knative service.

By default all Tasks of the Pipeline runs in parallel, you can control the execution via `runAfter` attribute. In this example we make the `deploy-kn-service` to run after the `build-java-app` .

Each Task in the Pipeline has

- **taskRef** - the reference to an existing defined task via `name`
- **params** - the Task parameters to define or override
  - **name** - the name of the parameter
  - **value** - the value of the parameter
- **resources** - used to bind the Pipeline inputs and output resource to Task's input and output resource.
- **name** - the local name of the resource
- **resource** - the Pipeline resource (defined under `resources`) name

In this demo the `build-app` Task needs bind two resources namely `source` and `builtImage` . The Pipeline `deploy-kubernetes-service` defines two resources `appSource` and `appImage` that can be configured via Run Pipeline.

The binding between the Pipeline resource and Task resources is done via the task's `resources` attribute. In this demo we bind `appSource` → `source` and `appImage` → `builtImage` .

IMPORTANT

Only pipeline resources of same type can be bound. e.g. resource of type `git` with `git` or `image` with `image`

Deploy Pipeline

The Kubernetes service deployment Pipeline could be created using the command:

```
kubectl apply -n tektontutorial -f svc-deploy.yaml
```

We will use the Tekton cli to inspect the created resources

```
tkn pipeline ls
```

The above command should list one Pipeline as shown below:

```
NAME          AGE             LAST RUN   STARTED   DURATION   STATUS
svc-deploy   4 seconds ago   ---        ---       ---        ---
```

TIP

Use the command **help** via `tkn pipeline --help` to see more options

Run Pipeline

A Kubernetes Service Account is required to deploy applications in to a Kubernetes namespace. The following resource defines a service account called `pipeline` in namespace `tektontutorial` , which will have needed permissions in the `tektontutorial` namespace to perform Tekton tasks.

```
kubectl apply -n tektontutorial -f $TUTORIAL_HOME/kubernetes/pipeline-sa-rc
```

NOTE

OpenShift Pipelines creates and uses the `pipeline` SA by default.

Run the following command to start the pipeline:

```
tkn pipeline start svc-deploy \①
  --resource="appSource=git-source" \②
  --resource="appImage=tekton-tutorial-greeter-image" \③
  --param="contextDir=springboot" \④
  --serviceaccount='pipeline'\⑤
  --showlog
```

① The resources of the Pipeline could be bound via `--resource` option, here we bind the Pipeline `appSource` to pipeline resource `git-source`

② Bind the Pipeline `appImage` to pipeline resource `tekton-tutorial-greeter-image`

③ Set the context directory to build the application sources

④ The service account to use with Pipeline run

It will take few seconds for the PipelineRun to show status as `Running` as it needs to download the container images.

TIP

- Use the command **help** via `tkn pipelinerun --help`
- Use `pr` as shorcut for pipelinerun commands e.g to list pipelinerun run the command `tkn pr ls`

View the pipeline run logs using,

```
tkn pr logs -f -a $(tkn pr ls -n tektontutorial | awk 'NR==2{print $1}')
```

If you see the PipelineRun status as `Failed` or `Error` use the following command to check the reason for error:

```
tkn pipelinerun describe <pipelinerun-name>
```

Invoke Service

Get the service URL,

```
SVC_URL=$(minikube -p tektontutorial -n tektontutorial service greeter --ur
```

NOTE

In OpenShift you can use the routes like:

```
oc expose svc greeter
SVC_URL=$(oc get routes greeter -o yaml | yq r - 'spec.url.host' )
```

Run the service,

```
http --body $SVC_URL
```

The `http` command should return a response containing a line similar to **Meeow!! from Tekton !!**

Cleanup

Delete the pipeline service account and its related permissions:

```
$TUTORIAL_HOME/bin/cleanup.sh
```