# Lab 9.2. Create a Pipeline to Deploy the Workload in the Cluster

## Overview

In this example, we will demonstrate the usage of a **`git clone`** task to clone the repository and a **`deploy to cluster`** task to deploy the workload into the cluster using Helm. We will modify the image name and host address in the `helm/nodejs-welcome/values.yaml` file from the previous example to match our requirements before proceeding with the **`deploy to cluster`** task. Now, let's examine both tasks.

## First Task (Git Clone)

To apply it directly from the Tekton Hub, you can use the following command:

```
kubectl apply -f
https://raw.githubusercontent.com/tektoncd/catalog/main/task/git-clone/0.9/g
it-clone.yaml -n tekton-pipelines
```

This task enables you to clone a GitHub repository by providing its URL and branch name. If your repository is private or requires authentication with a username and password, you can refer to Configuring Authentication for Git in the Installation and Configuration chapter.

## Second Task (Deploy to Cluster)

To create this **`task`**, use the manifest provided below and name it `deploy-to-cluster-task.yaml`.

```
apiVersion: tekton.dev/v1beta1
kind: Task
metadata:
  name: deploy-to-cluster-task
  labels:
    app.kubernetes.io/version: "0.3"
  annotations:
    tekton.dev/pipelines.minVersion: "0.12.1"
    tekton.dev/categories: Deployment
    tekton.dev/tags: helm
    tekton.dev/platforms: "linux/amd64,linux/s390x,linux/ppc64le,linux/arm64"
spec:
  params:
    - name: charts_dir
      description: The directory in source that contains the helm chart
    - name: release_name
      description: The helm release name
      default: <release name>
    - name: release_namespace
      description: The helm release namespace
      default: "default"
    - name: values_file
      description: "The values file to be used"
      default: "values.yaml"
    - name: tag
  workspaces:
    - name: output
  steps:
    - name: upgrade
      image: docker.io/kiwigrid/gcloud-kubectl-helm
      workingDir: /workspace/output
      script: |
        echo current installed helm releases
        helm list --namespace "$(params.release_namespace)"
        helm list -A
        echo installing helm chart...
        helm upgrade --install --wait --values
"$(params.charts_dir)/$(params.values_file)" --set tag="$(params.tag)"
--namespace "$(params.release_namespace)" "$(params.release_name)"
"$(params.charts_dir)" --debug
```

This task requires passing parameters such as **`charts_dir`**, **`release_name`**, **`release_namespace`**, **`values_file`**, and **`tag`**. It utilizes the **`helm`** command to deploy the code into the cluster.

Before applying this task, certain permissions need to be granted for our Service Account.

```
kubectl apply -f deploy-to-cluster-task.yaml -n tekton-pipelines
```

## Pipeline

Let us create a **`pipeline`** using the following manifest and save it as `deploy-workload-to-cluster-pipeline.yaml`:

```
apiVersion: tekton.dev/v1beta1
kind: Pipeline
```

```
metadata:
  name: deploy-workload-to-cluster-pipeline
spec:
  params:
    - name: gitrevision-tag
  workspaces:
    - name: shared-data-dep
  tasks:
    - name: helm-clone
      taskRef:
        name: git-clone
      params:
        - name: url
          value: "<your repo URL>"
        - name: revision
          value: $(params.gitrevision-tag)
      workspaces:
        - name: output
          workspace: shared-data-dep
    - name: deploy-to-cluster
      runAfter: ["helm-clone"]
      taskRef:
        name: deploy-to-cluster-task
      params:
        - name: charts_dir
          value: /workspace/output/<helm chart path>
        - name: release_name
          value: <release name>
        - name: release_namespace
          value: <namespace>
        - name: values_file
          value: values.yaml
        - name: tag
          value: <Tag>
      workspaces:
        - name: output
          workspace: shared-data-dep
```

Apply this pipeline with the following command:

```
kubectl apply -f deploy-workload-to-cluster-pipeline.yaml -n
tekton-pipelines
```

## PipelineRun

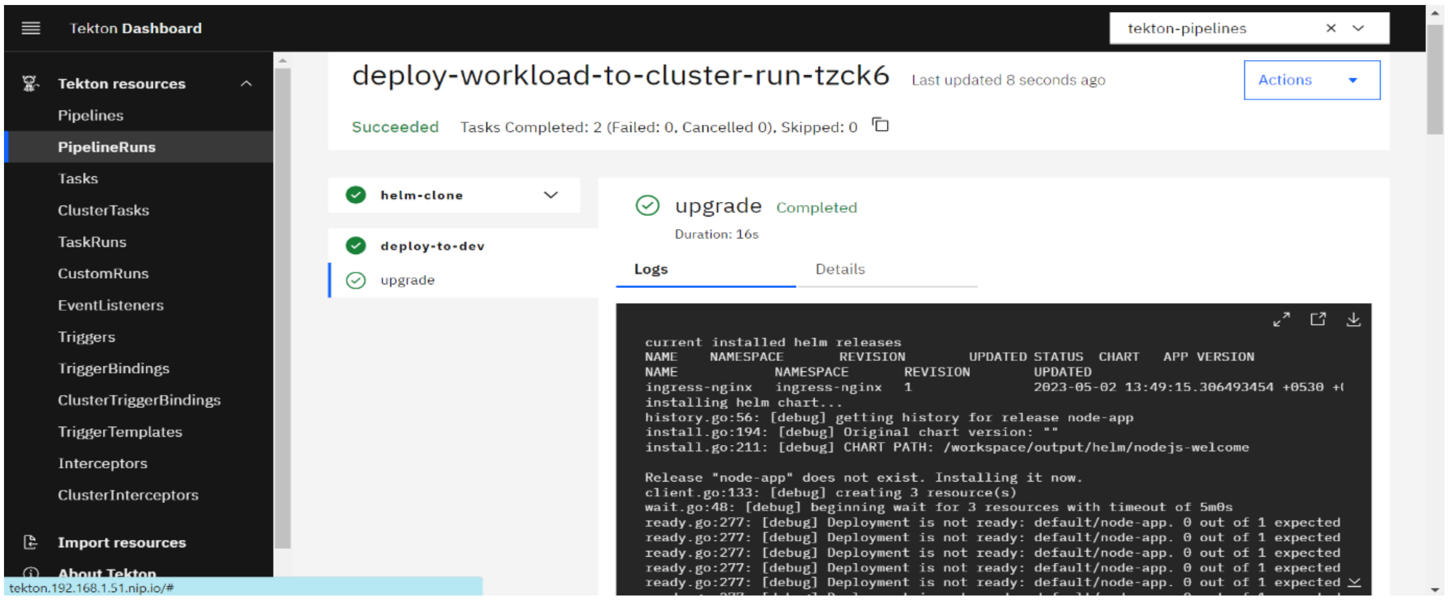
Having created the task and pipeline, we can now proceed to create a `PipelineRun` using the following manifest. Save the manifest with the name `deploy-workload-to-cluster-run.yaml`:

```
apiVersion: tekton.dev/v1beta1
kind: PipelineRun
metadata:
  generateName: deploy-workload-to-cluster-run-
spec:
  serviceAccountName: <service account name>
  pipelineRef:
    name: deploy-workload-to-cluster-pipeline
  podTemplate:
    securityContext:
      fsGroup: 1001
  params:
    - name: gitrevision-tag
      value: <branch name>
  workspaces:
    - name: shared-data-dep
      volumeClaimTemplate:
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 500Mi
```

Apply this manifest with the following command:

```
kubectl create -f deploy-workload-to-cluster-run.yaml -n tekton-pipelines
```

Once you have  successfully completed *deploy workload Pipeline*, your screen should look like the image below.



deploy-workload-to-cluster-run-tzck6 Last updated 8 seconds ago

Succeeded Tasks Completed: 2 (Failed: 0, Cancelled 0), Skipped: 0

upgrade Completed

Duration: 16s

```
current installed helm releases
NAME    NAMESPACE       REVISION        UPDATED STATUS  CHART   APP VERSION
NAME            NAMESPACE       REVISION        UPDATED
ingress-nginx   ingress-nginx   1               2023-05-02 13:49:15.306493454 +0530 +(
installing helm chart...
history.go:56: [debug] getting history for release node-app
install.go:194: [debug] Original chart version: ""
install.go:211: [debug] CHART PATH: /workspace/output/helm/nodejs-welcome

Release "node-app" does not exist. Installing it now.
client.go:133: [debug] creating 3 resource(s)
wait.go:48: [debug] beginning wait for 3 resources with timeout of 5m0s
ready.go:277: [debug] Deployment is not ready: default/node-app. 0 out of 1 expected
ready.go:277: [debug] Deployment is not ready: default/node-app. 0 out of 1 expected
ready.go:277: [debug] Deployment is not ready: default/node-app. 0 out of 1 expected
ready.go:277: [debug] Deployment is not ready: default/node-app. 0 out of 1 expected
ready.go:277: [debug] Deployment is not ready: default/node-app. 0 out of 1 expected
```

**Successfully Completed Deploy Workload Pipeline**