NKS on HCI - Part 3

In this series of posts, we're covering various aspects of getting started with building a CI/CD Pipeline using NKS with NetApp HCI.

In this post, we'll configure the Jenkins deployment that was deployed in part 2, build a pipeline, and run our first builds. For this example I'll be using a small demo application, the repository of which can be found here. This project contains 2 docker images, as well as a Jenkinsfile that will be used to run the pipeline.

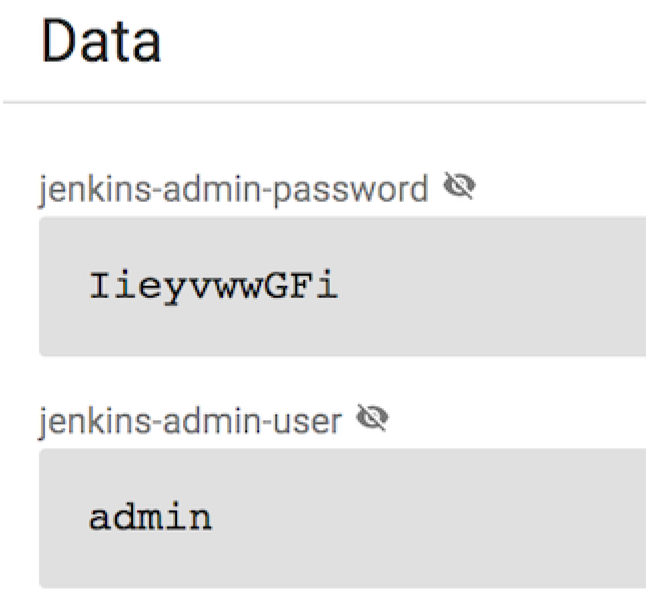
Retrieving the Jenkins Password

By default, Helm will generate a random password for the Jenkins **admin** user during deployment. We'll need to retrieve this password to login to the Jenkins console.

Via the K8S Dashboard

Open the Kubernetes Dashboard from the cluster details page, and navigate to the **jenkins** namespace.

Within this namespace the Jenkins credentials can be found in the `data` section of the `jenkins` Opaque secret:


Data

jenkins-admin-password

IieyvwwGFi

jenkins-admin-user

admin

Via Kubectl

The jenkins credentials can also be retrieved using Kubectl by running the following command:

```
$ kubectl get secrets jenkins  -n jenkins -o 'go-template={{index .data
"jenkins-admin-password"}}' | base64 -d
$ IieyvwwGFi
```

After retrieving the default password, we can log into Jenkins as the **admin** user.

Jenkins Configuration

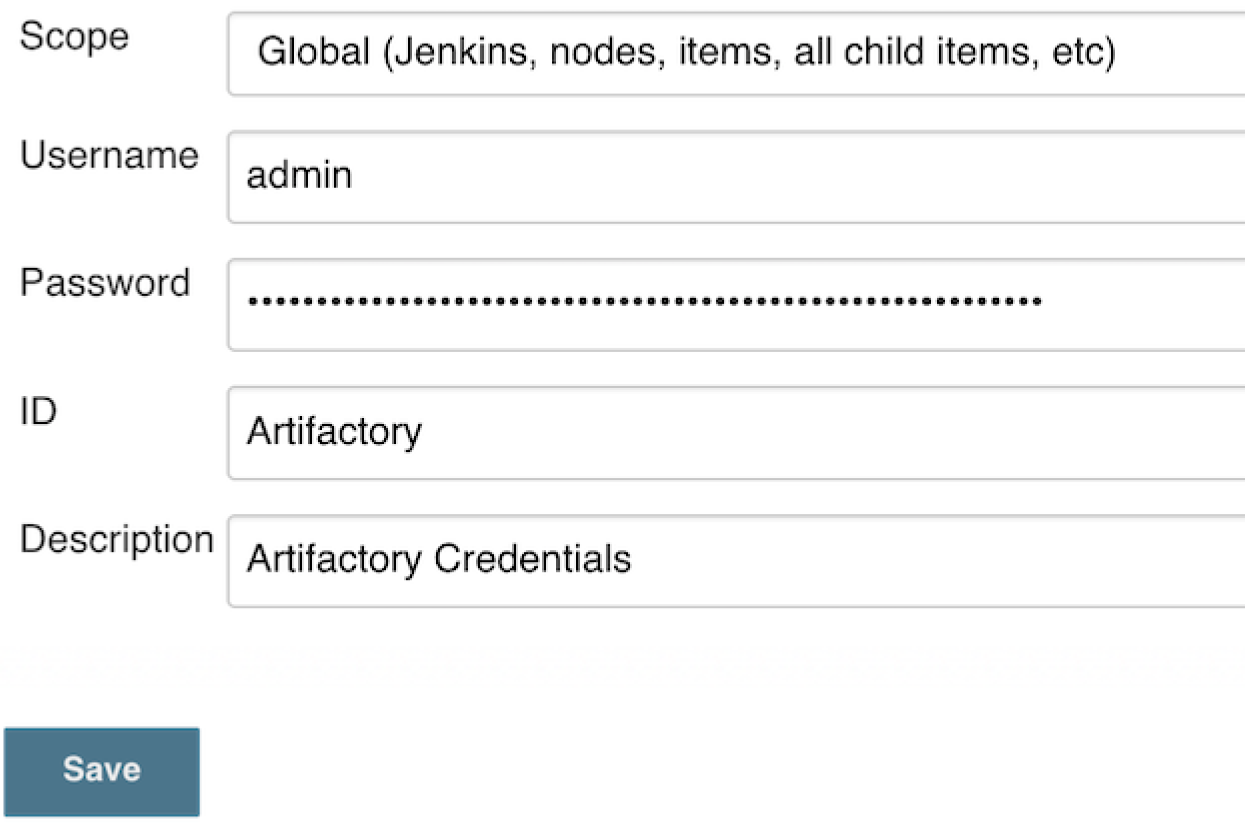
In order to get our pipeline fuctional, we'll need to configure three things:

- Docker Registry Credentials
- Kubernetes Pod Template Spec
- Project Pipeline

Add Docker Registry Credentials

The first thing we'll need is credentials to a Docker registry. These will be referenced in the Pipeline and used to push the application images.

On the left-side column, select `Credentials > Jenkins > Global Credentials`, then click `Add Credentials`.



Scope	Global (Jenkins, nodes, items, all child items, etc)
Username	admin
Password	••
ID	Artifactory
Description	Artifactory Credentials

Configure build container

Next we'll configure the build container. This provides Jenkins with a pod template to use when executing pipeline tasks. We'll use a docker-in-docker (`dind`) image to run our build stages.

- On the left-side column, select `Manage Jenkins > Configure System`
- In the 'Cloud' Section, you'll see a sub-section for `Kubernetes > Kubernetes Pod Template > Container Template`
- In the field labelled 'Raw yaml for the Pod', paste the following:

```
---
apiVersion: v1
```

```
kind: Pod
metadata:
    labels:
    jenkins/kube-default: true
    app: jenkins
    component: agent
spec:
    containers:
    - name: jnlp
        image: jenkins/jnlp-slave:3.10-1
        imagePullPolicy: Always
        env:
        - name: POD_IP
        valueFrom:
            fieldRef:
            fieldPath: status.podIP
        - name: DOCKER_HOST
        value: tcp://localhost:2375
    - name: dind
        image: docker:18.05-dind
        securityContext:
        privileged: true
        volumeMounts:
        - name: dind-storage
            mountPath: /var/lib/docker
    volumes:
    - name: dind-storage
        emptyDir: {}
```

Configure Pipeline

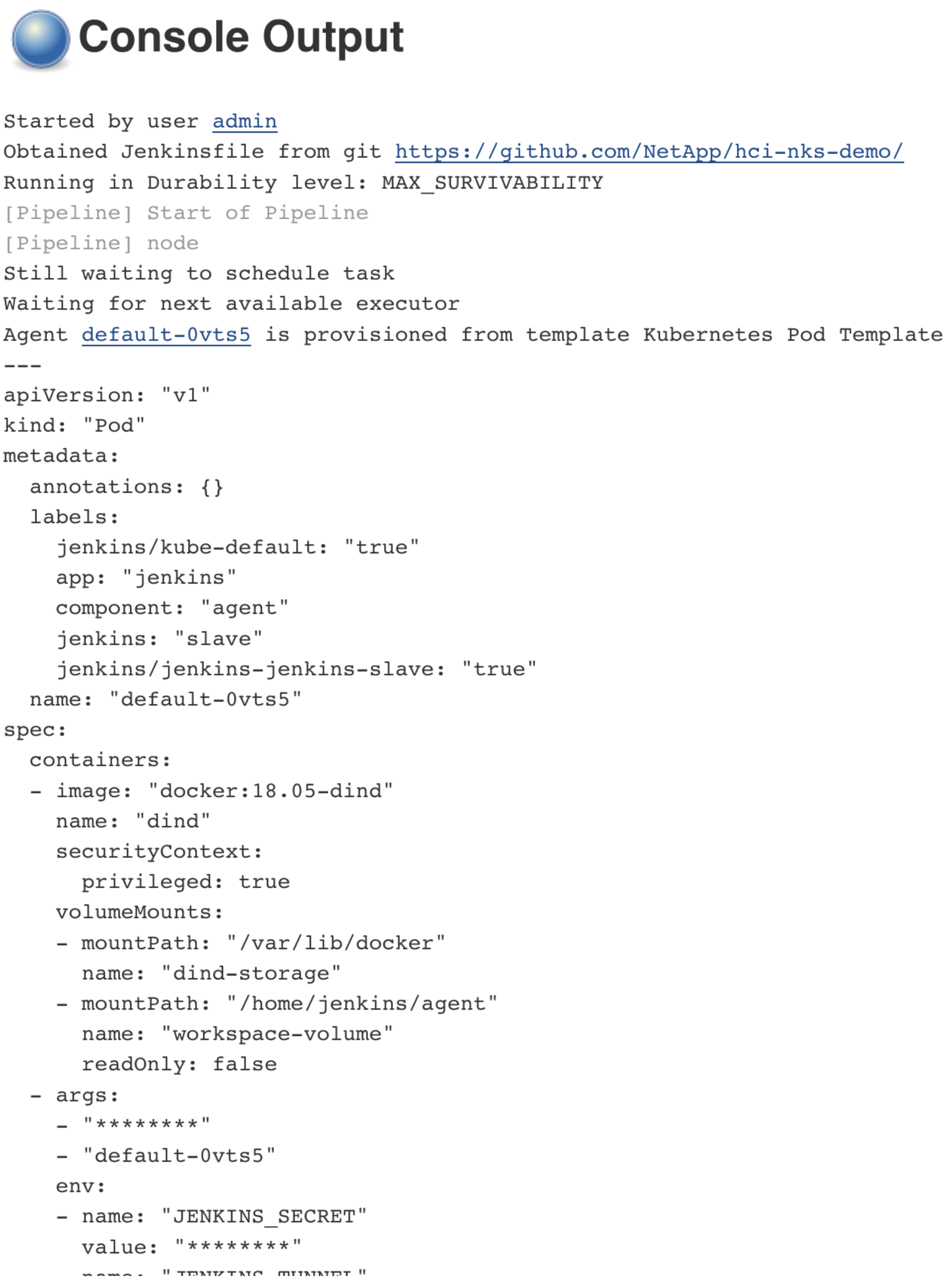
Next we'll configure a simple Jenkins pipeline. This pipeline will clone a remote repository, and run the steps specified in the `Jenkinsfile` contained within.

- Navigate to `Jenkins Home > New item`
- Select `Pipeline`, and give it a name. Click `OK`
- On the Pipeline configuration page, scroll down to the `Pipeline` section.
  - Under **Definition**, select `Pipeline script from SCM`
  - Under **SCM**, choose `Git`
  - In the field for **Repository URL**, enter `https://github.com/NetApp/hci-nks-demo/`
  - In the field for **Branch Specifier**, enter `*/nks-shell`
- Click **Save**

Run Pipeline

On the left-side column, click **Build Now** to run the pipeline. Selecting the job details will display a running log of the build output. This job will perform the following actions:

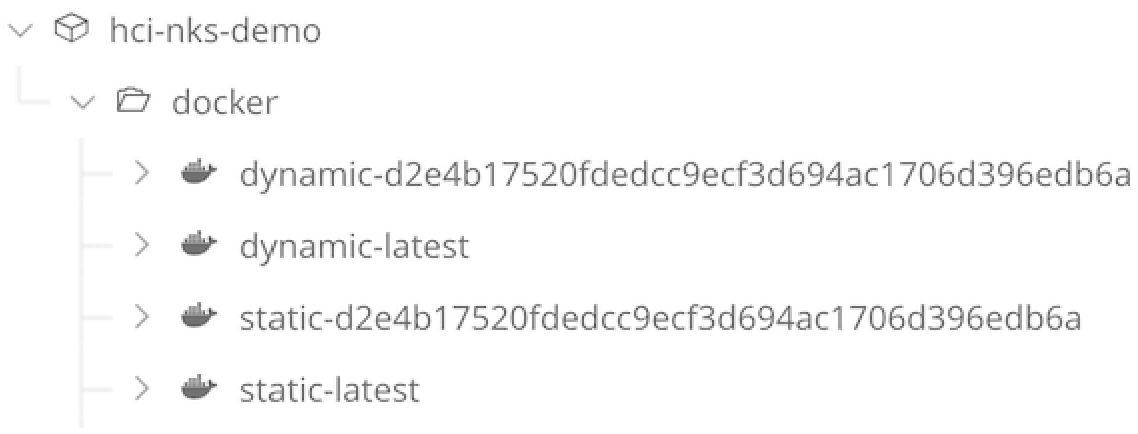
- Cloning the remote repository
- Build/Test application images
- Push images to registry



```
Started by user admin
Obtained Jenkinsfile from git https://github.com/NetApp/hci-nks-demo/
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Still waiting to schedule task
Waiting for next available executor
Agent default-0vts5 is provisioned from template Kubernetes Pod Template
---
apiVersion: "v1"
kind: "Pod"
metadata:
  annotations: {}
  labels:
    jenkins/kube-default: "true"
    app: "jenkins"
    component: "agent"
    jenkins: "slave"
    jenkins/jenkins-jenkins-slave: "true"
  name: "default-0vts5"
spec:
  containers:
  - image: "docker:18.05-dind"
    name: "dind"
    securityContext:
      privileged: true
    volumeMounts:
    - mountPath: "/var/lib/docker"
      name: "dind-storage"
    - mountPath: "/home/jenkins/agent"
      name: "workspace-volume"
      readOnly: false
  - args:
    - "********"
    - "default-0vts5"
    env:
    - name: "JENKINS_SECRET"
      value: "********"
      name: "JENKINS_TUNNEL"
```

View artifacts

Navigating into Artifatory, we can see our images have been stored:



```
hci-nks-demo
  docker
    dynamic-d2e4b17520fdedcc9ecf3d694ac1706d396edb6a
    dynamic-latest
    static-d2e4b17520fdedcc9ecf3d694ac1706d396edb6a
    static-latest
```

Conclusion

In this post, we configured Jenkins with a new pipeline to run jobs in Kubernetes. We also saw an example of how to use the pipeline to build, test, and store the images for a custom application.

In the next post, we'll explore how to deploy our application into a Kubernetes cluster with a custom helm chart using NKS.