**Task 1# Setup & fill postgres database**

1. result/docker/postgresql/Dockerfile

**FROM** docker.io/bitnami/postgresql:11.5.0-debian-9-r84

**COPY** . /docker-entrypoint-initdb.d/

1. Build docker image & push to private container registry

$ docker build -t nexgtech\postgresql:11.5.1

$ docker push nexgtech\postgresql:11.5.1

**Task2# Dockerize the Python API**

[**https://bitbucket.org/devops\_work/titanic-app/src/master/result/docker/**](https://bitbucket.org/devops_work/titanic-app/src/master/result/docker/)

1. result/docker/api/Dockerfile

**FROM** python:3.7

**COPY** . /app

**WORKDIR** /app

**RUN** pip install -r requirements.txt

**EXPOSE** 5000

**ENTRYPOINT** ["python"]

**CMD** ["server.py"]

1. Build docker image & push to private container registry

$ docker build -t nexgtech\titanic-api

$ docker push nexgtech\titanic-api

**Task 3# Deploy to Kubernetes (free, cloud) Using Helm**

1. Create new namespace for installing helm chart in K8s

$ kubectl create namespace test

1. Update the image name and tag number in the chart value.yaml file for both titanic api and postgresql database which added as a dependency chart.

File location: result/helm/api-app/values.yaml



1. Install helm chart

$ helm install titanic . -n test

1. List the chart release

$ helm list -n test

NAME NAMESPACE REVISION UPDATED STATUS CHART APP VERSION

titanic test 4 2021-07-05 19:13:41.950136924 +0000 UTC deployed api-0.1.0 1.16.0

1. After installing helm chart, you can see the following services

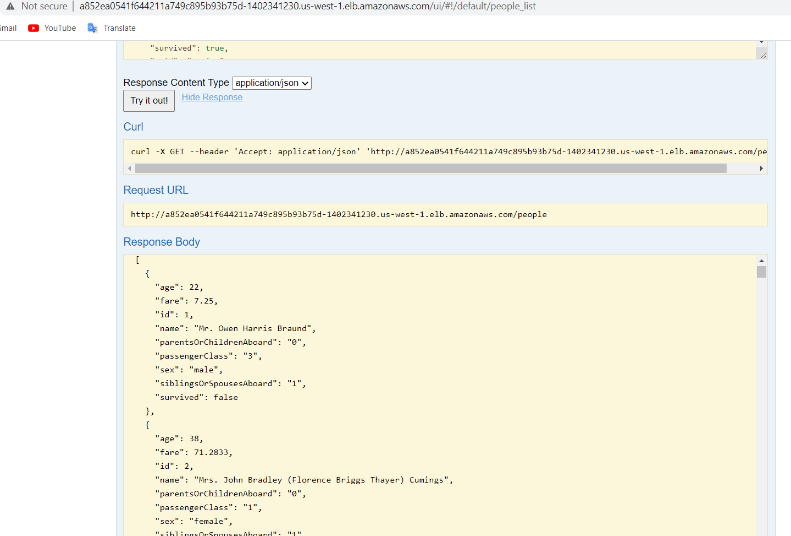
$ kubectl get services -n test

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

titanic-api LoadBalancer 100.71.224.142 elb 80:30111/TCP 17m

titanic-postgresql ClusterIP 100.65.2.224 <none> 5432/TCP 17m

1. Copy the **titanic-api** external url and load the page in the browser to view the application



**Task 4# Create a simple CI/CD pipeline using Jenkins to build and deploy to Kubernetes**

1. The Jenkins-master.yaml file contains all the resources for deploying Jenkins server.

$ kubectl apply -f Jenkins-master.yaml -n test

1. Verify pod is running by using below command.

$ kubectl get pods -n test

NAME READY STATUS RESTARTS AGE

jenkins-master-f49d699c7-fdjv7 1/1 Running 0 3h15m

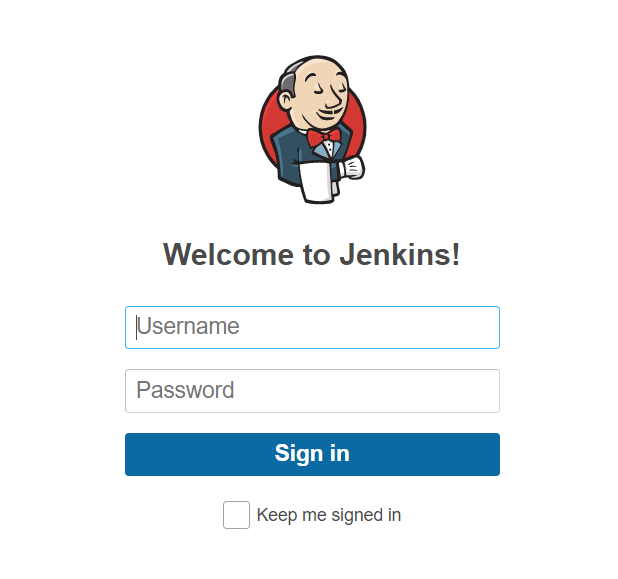
1. Get the load balancer url for the Jenkins service.

$ kubectl get svc -n test

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

jenkins-master LoadBalancer 100.70.99.117 xxx.elb.amazonaws.com 80:32255/TCP,50000:32321/TCP 3h55m

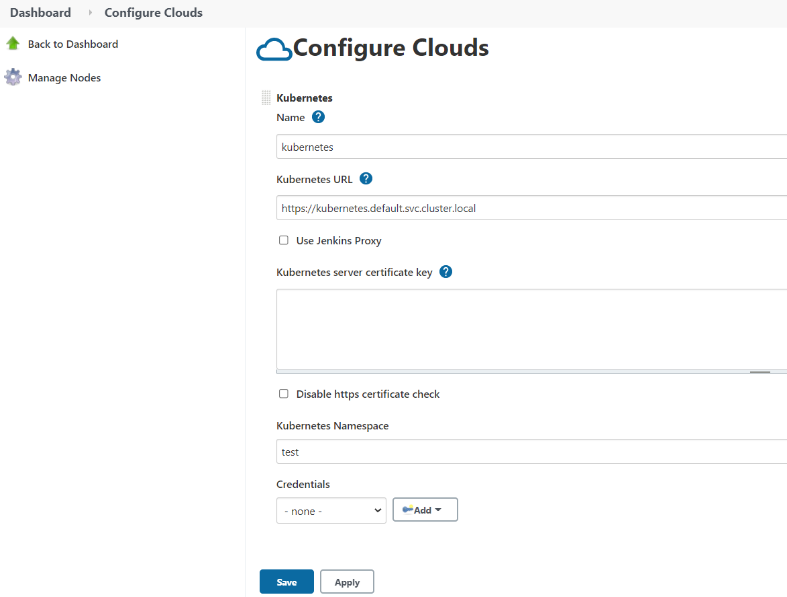
1. Copy the url and load the page in the browser to view the application.



**Note**: Default username & password is “admin”

1. Add Kubernetes cluster as a cloud in Jenkins configuration

Go to Manage Jenkins -> Configure System -> Manage Nodes & Clouds -> Configure Clouds -> Add new Cloud: Kubernetes



Under “Kubernetes Cloud Details” the following values need to be added:

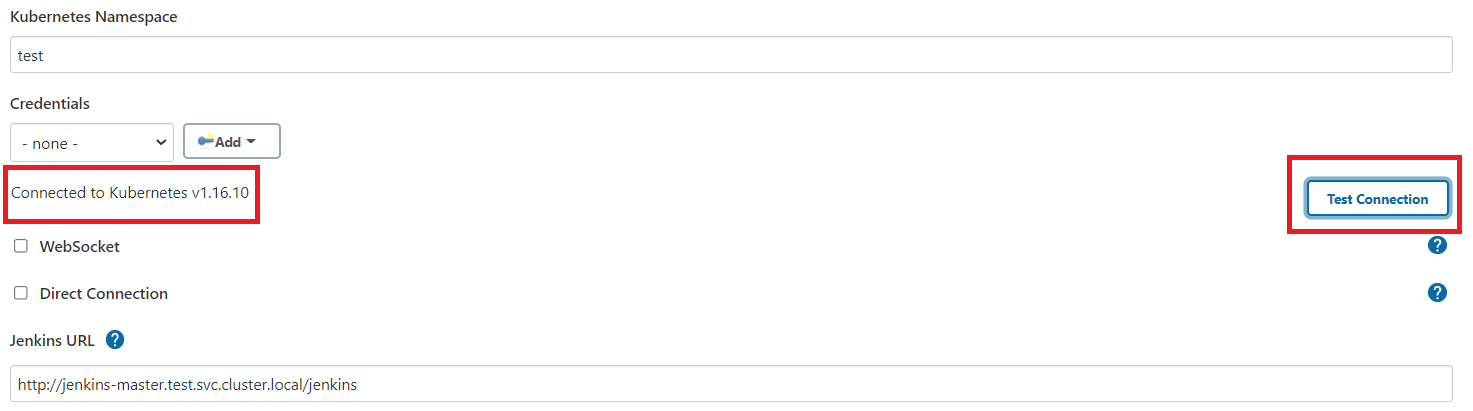
Name: Kubernetes

Kubernetes URL: <https://kubernetes.default.svc.cluster.local>

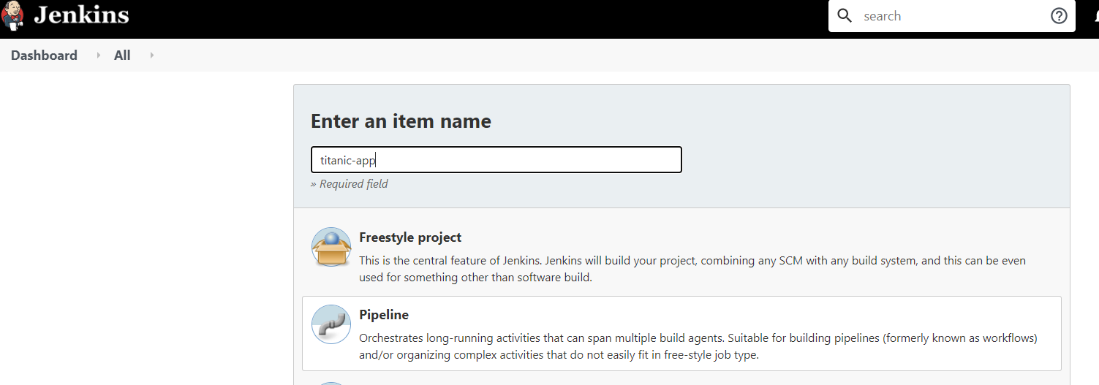
Kubernetes Namespace: test

Jenkins URL: <http://jenkins-master.test.svc.cluster.local/jenkins>

1. Verify the connection by selecting “Test Connection” button and save the configuration

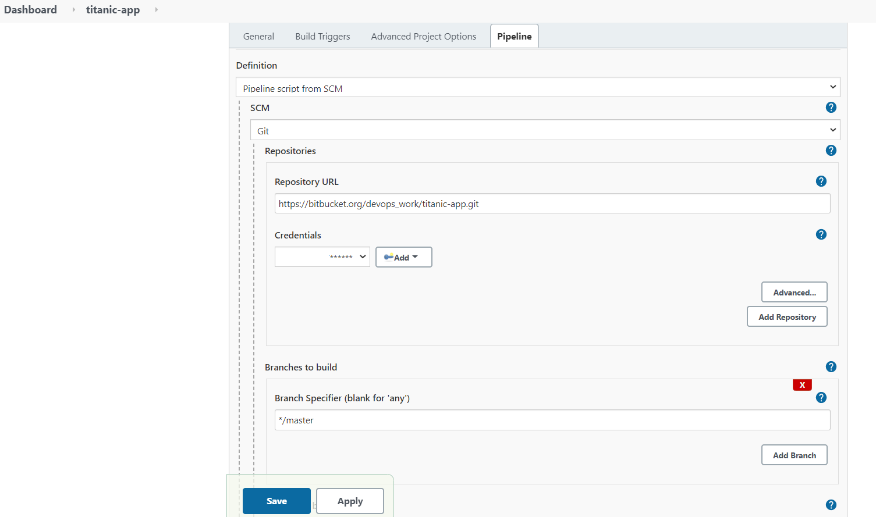


1. To create a pipeline job, click on “**New Item**” in the Jenkins dashboard, select type as “Pipeline” and then provide name of the new pipeline.



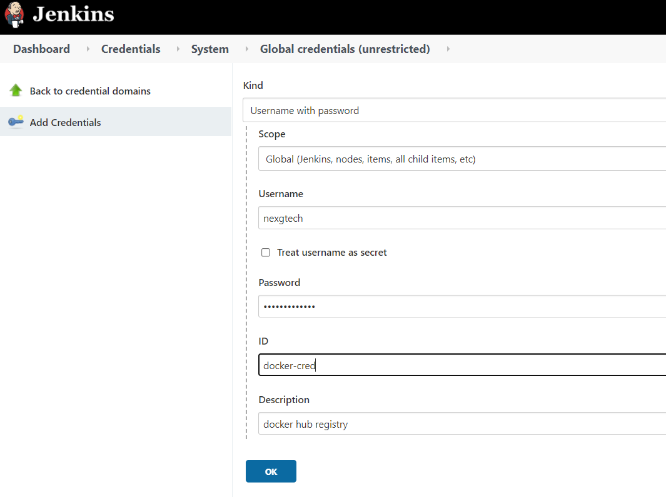
1. In the pipeline section, select the “**Pipeline script from SCM**” option and provide the following details:

* SCM: Git
* Repository Url: Provide git repository url where you have the Jenkinsfile and other source code files.
* Credentials: Configure credentials which used to checkout the code from git repository.
* Branch: \*/master
* Script path: result/Jenkinsfile

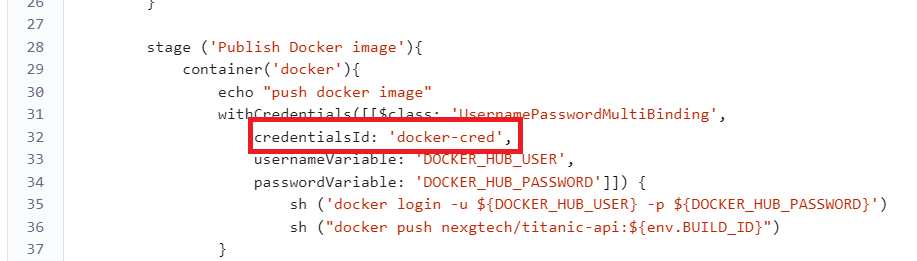


1. Add a private container registry credentials in Jenkins to push & pull the newly created images in the pipeline.

Go to **Manage Jenkins** -> **Manage Credentials** -> select “**global**” link under Store scope -> **Add Credentials** and add following details as shown below



1. Specify the same credential id in the Jenkinfile what you defined earlier in the Managed credentials.



1. Test the pipeline by executing the build.

