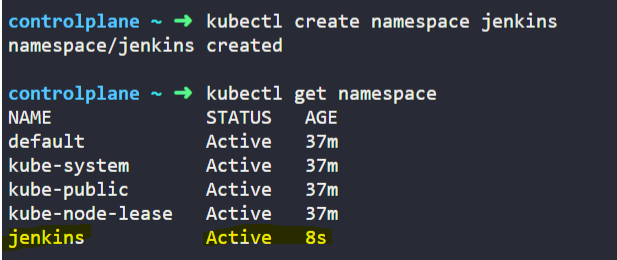
**Deploying an application in Kubernetes cluster using Jenkins pipeline**

**Problem statement:** Deploy an application using Jenkins under Kubernetes on the same cluster.

**Note:** Kindly note that okteto cloud it is not working with my personal github account it is asking for business account – which is not supposed to used by my company account – so whatever I created with the freely available Kubernetes playground platform have shown below in the screenshots. Jenkins server also not able to configure as the playgrounds does not meet the requirements, but have provided step by step process of how we deploy an application.

**Step by step process:**

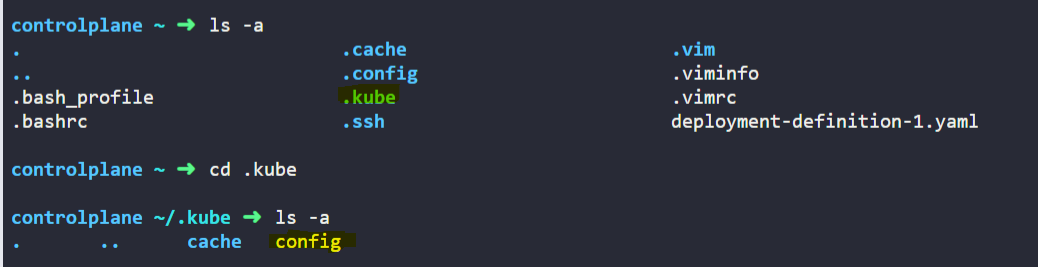
1. Open any of the Kubernetes service platform – if not we can set it up.
2. The first step is to create a Name space in Kubernetes.



We can see that our name space – Jenkins got created

1. We can see our **kubeconfig** file under .kube directory by default, we can also setup multiple kubeconfig files using environment variable as well.

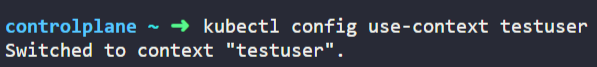
The below is how we can find our kubeconfig file:



We can set **new context** as well as below – kubectl uses a file called kubeconfig file to get the cluster and user details - context is like a mapping between clusters and users.



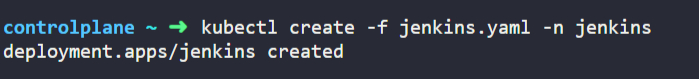
To switch to our created context we can use the below command -

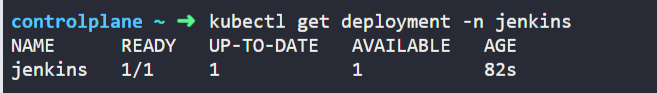


1. Now we need **create the Jenkins deployment file and service file**.

Created – jenkins.yaml file and Jenkins-svc.yml file

1. Create the Jenkins deployment file in the Jenkins name space.

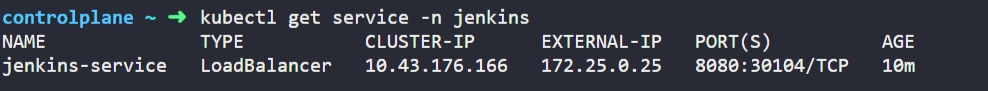


We can check the deployment status as below: 

1. Once the jenkins is in ready we are good to create our service file under the same namespace

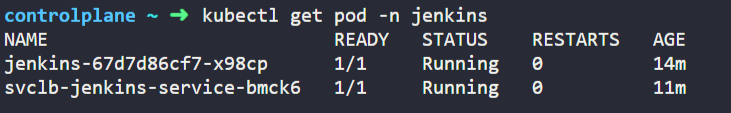


We can check the whether service got created or not by using the below



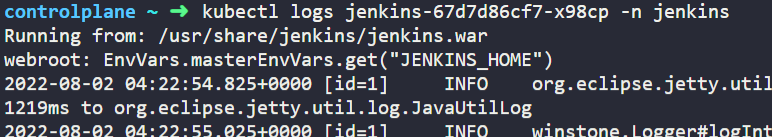
Here the external load balancer ip is 172.25.0.25

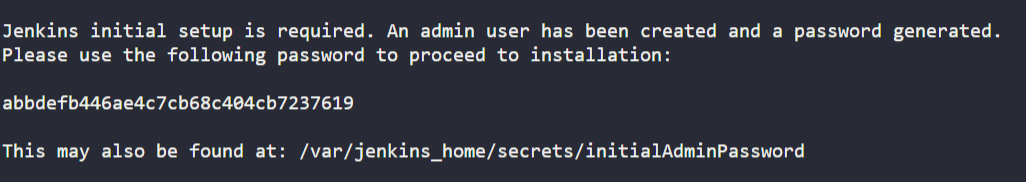
1. We can get the status of our deployment pods and service pods with below command:



1. Now we have to connect to Jenkins server using the LB ip and the port which we defined – i.e., **172.25.0.25:30104**

Once we browse the above url – it will ask us the password to enter – we can get the password from the deployment pod as below:

we will get the lot of logs – under the logs we will find the password for the same as below:



With the help of this password, we can login to Jenkins server.

**Other way to install Jenkins server in docker cluster is running by this command in docker hosted environment:**

docker run -d -p 8080:8080 -p 50000:50000 -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts

1. After Jenkins setup is complete – it will take sometime to install all the required plugins later that we have to setup admin user and password details.
2. The next step is to install plugins – for that – go to manage plugins>go to available and search for Docker pipeline and install(it is mandatory because some of the commands may not work if we do not have this plugin)
3. Next, we need to install Kubernetes continuous deploy plugin of desired version into Jenkins server.
4. Now we can start creating the pipeline in Jenkins:

* Go to create an item name - create a job name as ‘App deploy’ and select pipeline – and then click ok. Our job is ready now.
* Now we can see the General, Build triggers, Advance project options and Pipeline script tabs.
* Here we can set if there is any changes in Git repo the pipeline will trigger automatically – for that we can mention our git repo url.
* Here under the pipeline definition, we can choose Pipeline script from SCM and define our Jenkins pipeline file name there – in our case our - the name is jenkinspipeline.
* Once we completed the above step, now we need to save the credentials of dockerhub in our Jenkins server as we defined to fetch the credentials for pushing the image from docker hub in Jenkins pipeline file.
* For that go to manage Jenkins>manage credentials>click on global>click on add credential – here the kind should be username and password and give the docker hub login credentials and ID as dockerhublogin as we defined this parameter to fetch the credentials and then click on ok.
* And also we need to save our kubeconfigid details in Jenkins server like how we do it in the above step – so that will act as authentication to deploy our app to Kubernetes.
* Again, go to manage credentials>now we need to select kind as Kubernetes configuration and give the ID as Kubernetes i.e., what we defined in our Jenkins pipeline file and for Kubeconfig section we can copy for Kubeconfig content and paste in Kubeconfig serction which we saved in the above #3 and save it.

1. Now we can start our pipeline by clicking on the Build now option in Jenkins server, we can see the step-by-step process of deployment in the console output – it will take sometime according to our application and image we are using and once the deployment is completed, we can see the success message at the end of console, if we have any errors - we can check according to the error message we got.
2. We can also see our deployment status of how much time it took for each step in Checkout source, Build image, Pushing image, Deploying app to Kubernetes.
3. Now we can check our deployment and pod status in Kubernetes console – by using the following commands:

kubectl get deployments -n Jenkins

kubectl get pods -n Jenkins

kubectl get service -n Jenkins

we can see the status as running and ready states.

1. Now finally we can access our application using the external ip address and port which we will get those details from get service command.

**Bonus #1 – Set the email notifications whenever the pipeline build is succeeded or failed in jenkins.**

For any event we can setup the email notification in Jenkins, the following are the steps to be followed:

1. Go to manage Jenkins>click on **Configure system** – we will get the configurations page.
2. Under that go to E-mail notification>under the SMTP server give smtp.gmail.com and click on Advance – here click on the **Use SMTP authentication** –provide our gmail and password, and also click on the **Use SSL** and finally provide the SMTP port as 465 – as it is the port used by gmail.
3. Once all the above is completed click on save and apply – you can test with the help of test configuration button. Now we will receive a test email notification for our gmail.

**Note**: Make sure that gmail account is not in multi factor authentication mode and also allow less secure apps: On in gmail settings.

1. Now we can go to Extended E-mail notification – provide the details as we did before, here under the Advanced settings we can see lot more options than the above one – those are content type we need, Recipient list and also default subject with the format as $PROJECT\_NAME – Build # $BUILD\_NUMBER - $BUILD\_STATUS

Here Project name is our build name, build number is the id of the build and build status is pass/fail.

1. And we have lot of options to choose whenever the email has to be triggered like for 2nd failure or before build and so on and then click on apply and save.
2. So test these email notifications we can go on to our existing Job – in that go configure section, there scroll down to Post-build Actions drop down – here we can choose the editable email notification – here we can see our configuration which we set before and edit whomever and what is the message you want to send.
3. Then we fill get the email according to the build status

**Bonus #2 - Make the application you've deployed highly scalable.**

In order to make the application highly available and scalable – we can achieve this by creating multiple replica pods.

If we define our replicas as 2 in our deployment pipeline – so whenever one of the current pod is down – the Kubernetes clusters identifies that and it will start running the new pod with the same configuration – by this way we can maintain our application to be highly available and scalable based on the application load or the pod health. This can be done by using the readiness and liveliness pod concepts.

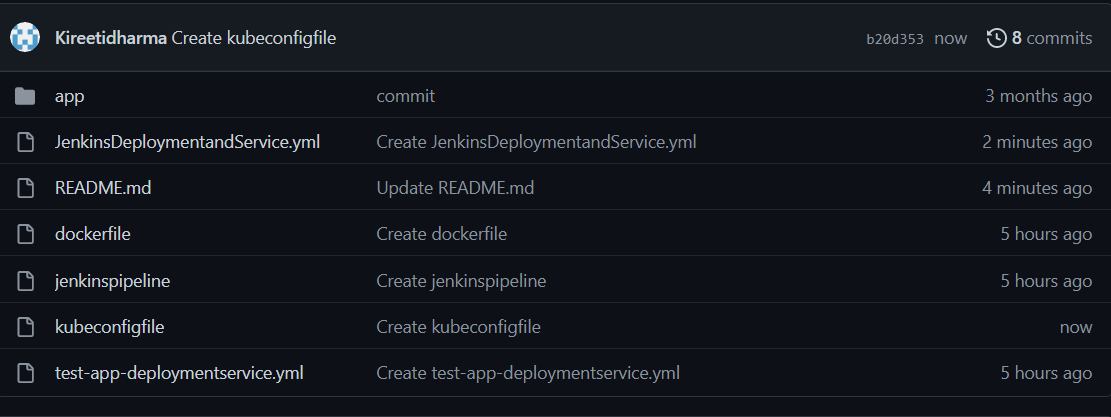
**Bonus #3 - Make Jenkins setup in such a way that whenever pipeline runs it will dynamically provision containers as Jenkins worker nodes and the Jenkins worker must delete automatically when the job is finished**

To achieve this, we need to made some config changes under the following location

Go to Jenkins Home Page => Manage Jenkins => Manage Nodes and Cloud => Configure Clouds => Add a new Cloud – then we need to select the options to configure

In this document not providing more details into this dynamic setup.

**Git Hub files description:**

****

**app –** the folder of the application

**JenkinsDeploymentandService.yml –** This file is to deploy Jenkins server in kubernetes.

**README.md –** provided the documentation steps.

**dockerfile –** File used to pull the image from docker hub

**jenkinspipeline –** File used to deploy the application as stages in Jenkins server.

**kubeconfigfile -** kubernetes playground configuration file with some changes.

**test-app-deploymentservice.yml -** The file used for application deployment and service deployment in Kubernetes.