Contents

Build WebLogic container image using Oracle Container Pipelines (Wercker) 2

Prerequisites 3

Prepare Oracle Container Registry access 3

Import WebLogic Operator Tutorial's source repository into your Github repository 8

Setting Up Tools 18

Install Git and helm 22

Install WeblogicOperator 23

Install Traefik 25

Create PV and PVC 27

Create Domain 28

Access the Weblogic Application 31

Access Sample App 31

Update the sample app image 32

Scaling the pod 36

Install Weblogic Metrics for promethus and Grafana 37

Install Grafana and Prometheus 40

Build WebLogic container image using Oracle Container Pipelines (Wercker)

Oracle Container Pipelines (Wercker) is a Docker-Native CI/CD Automation platform for Kubernetes & Microservice Deployments. Wercker is integrated with Docker containers, which package up application code and can be easily moved from server to server. Each build artifact can be a Docker container. The user can take the container from the Docker Hub or his private registry and build the code before shipping it. Its SaaS platform enables developers to test and deploy code often. They can push software updates incrementally as they are ready, rather than in bundled dumps. It makes it easier for coders to practice continuous integration, a software engineering practice in which each change a developer makes to the codebase is constantly tested in the process so that software doesn't break when it goes live.

Oracle Container Pipelines is based on the concept of pipelines, which are automated workflows. Pipelines take pieces of code and automatically execute a series of steps upon that code.

This tutorial demonstrates how to create Oracle Container Pipelines application (CI/CD) to build/update custom WebLogic container image using official WebLogic image from Docker Store as base source.

The custom WebLogic Domain has the following components configured/deployed:

- Web Application to demonstrate WebLogic Operator features and application life cycle management

The key components of Oracle Container Pipelines:

1. **Step** is self-contained bash script or compiled binary for accomplishing specific automation tasks.
2. **Pipelines** are a series of steps that are triggered on a git push or the completion of another pipeline.
3. **Workflows** are a set of chained and branched pipelines that allow you to form multi-stage, multi-branch complex CI/CD flows that take your project from code to production.

4. All pipelines execute inside a **Docker container** and every build artefact can be a Docker container.

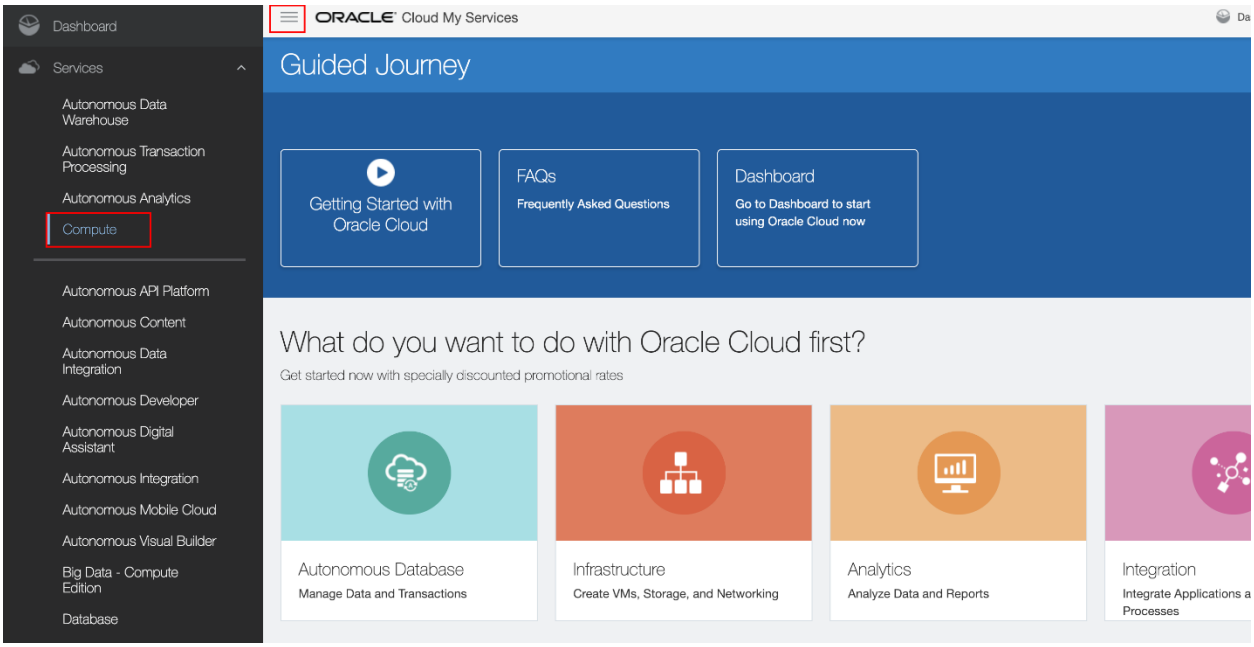
Prerequisites

- Oracle Cloud Infrastructure enabled account.
- Docker account.
- Github account.
- Oracle Container Pipeline Wrecker
- Oracle SSO account

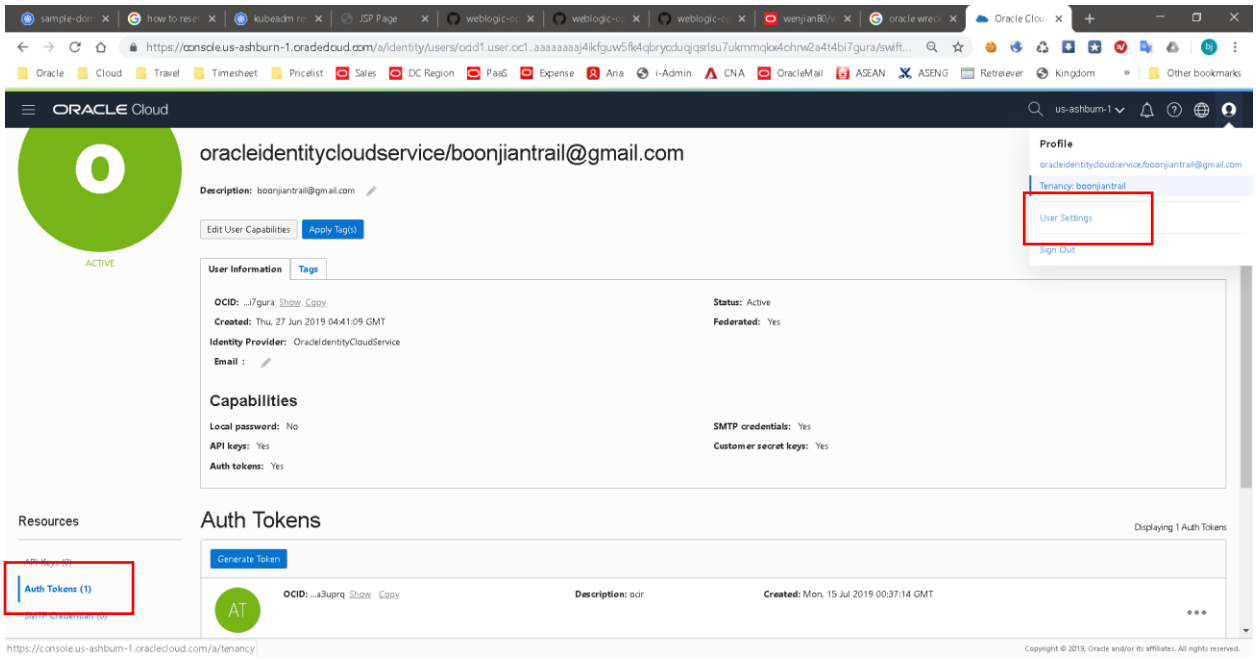
Prepare Oracle Container Registry access

Before you create your build pipeline you need to get your Oracle Container Registry token. Token acts as password to container registry provided by Oracle Cloud Infrastructure.

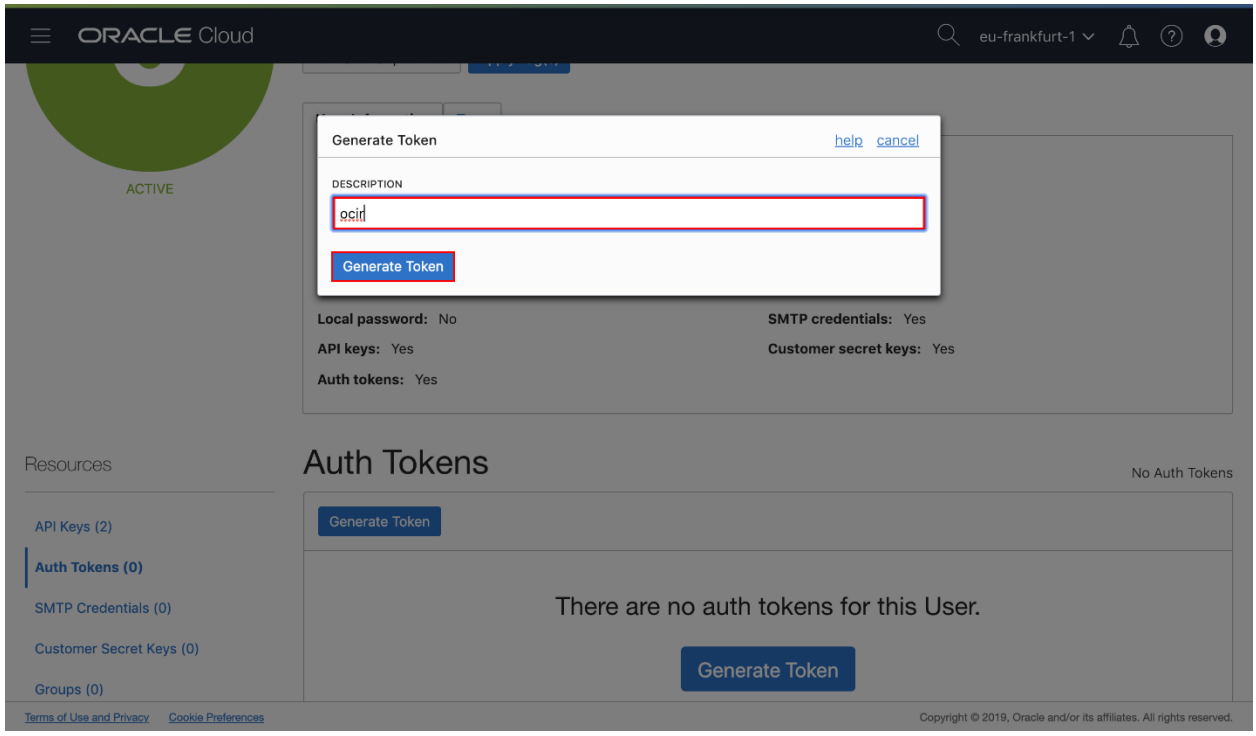
Open your OCI (Oracle Cloud Infrastructure) Console. If necessary Sign in again using your Cloud Services link you got in email during the registration process. Remember on the dashboard you need to click the menu icon at the top left corner and select **Compute** on the left sliding menu.



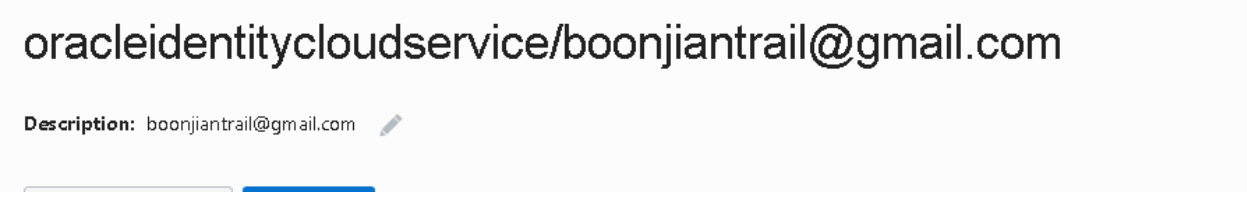
Using the OCI console page click the user icon and select **User Settings**. On the left area of the User details page select the **Auth Tokens** item. Click the **Generate Token** to get a new token.



Enter a description which allows you to easily identify the allocated token later. For example if you want to revoke then you have to find the proper token to delete. For example *ocir*.

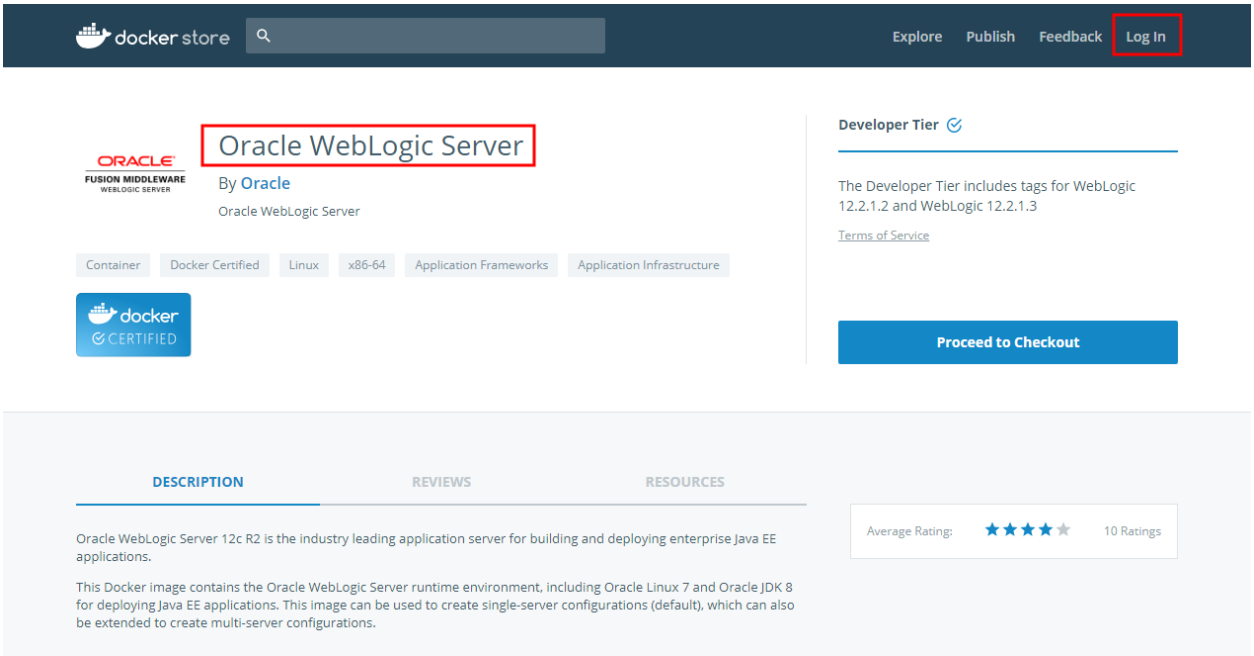


Since you are on the User details page please note the proper user name and token for later usage. You need to use this user name in order to login to OCI Registry for push and pull images.

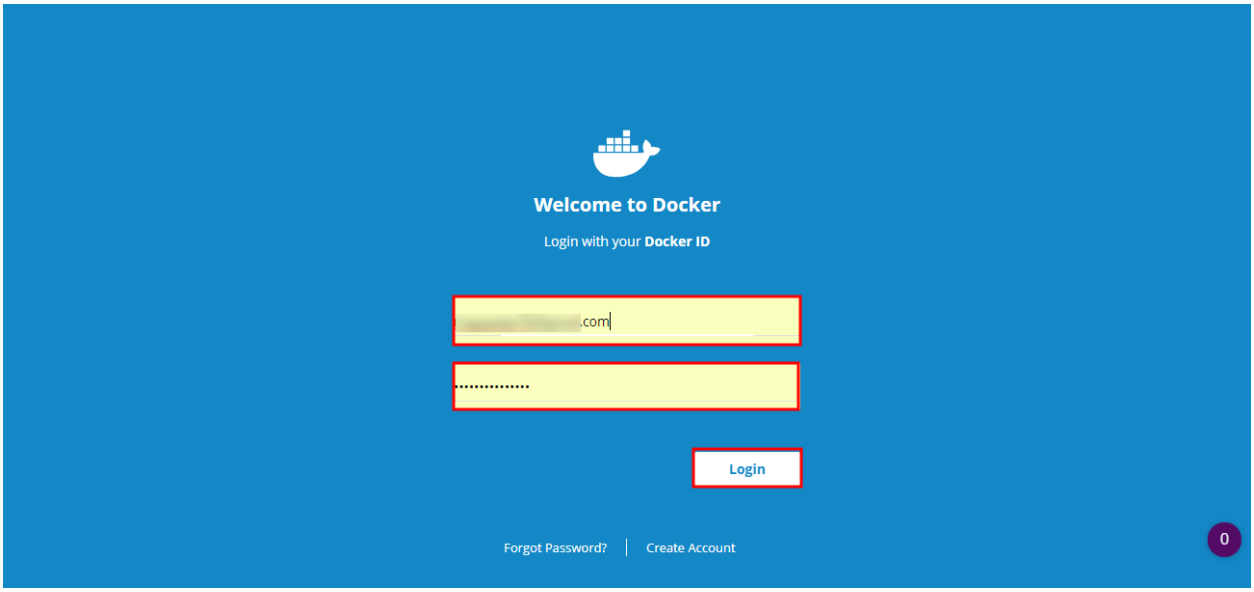


Accept Licence Agreement to use `store/oracle/weblogic:12.2.1.3` image from Docker Store

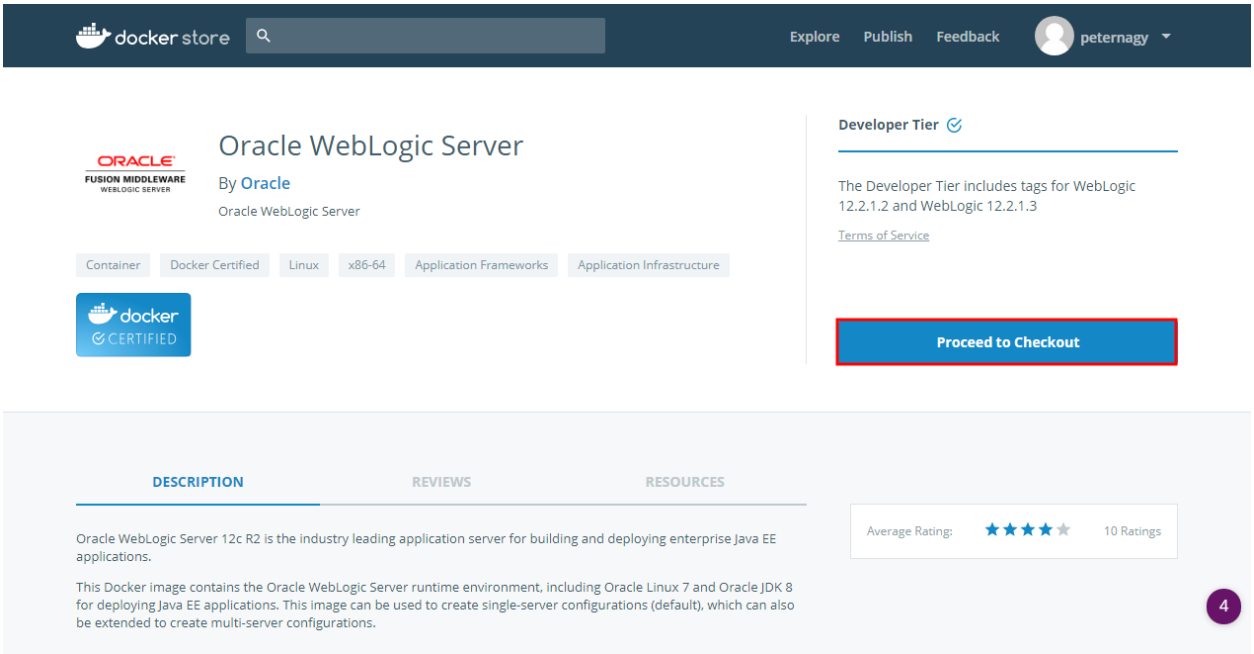
If you have not used the base image `store/oracle/weblogic:12.2.1.3` before, you will need to visit the Docker Store web interface and accept the license agreement before the Docker Store will give you permission to pull that image.
Open https://store.docker.com/images/oracle-weblogic-server-12c in a new browser and click **Log In**.



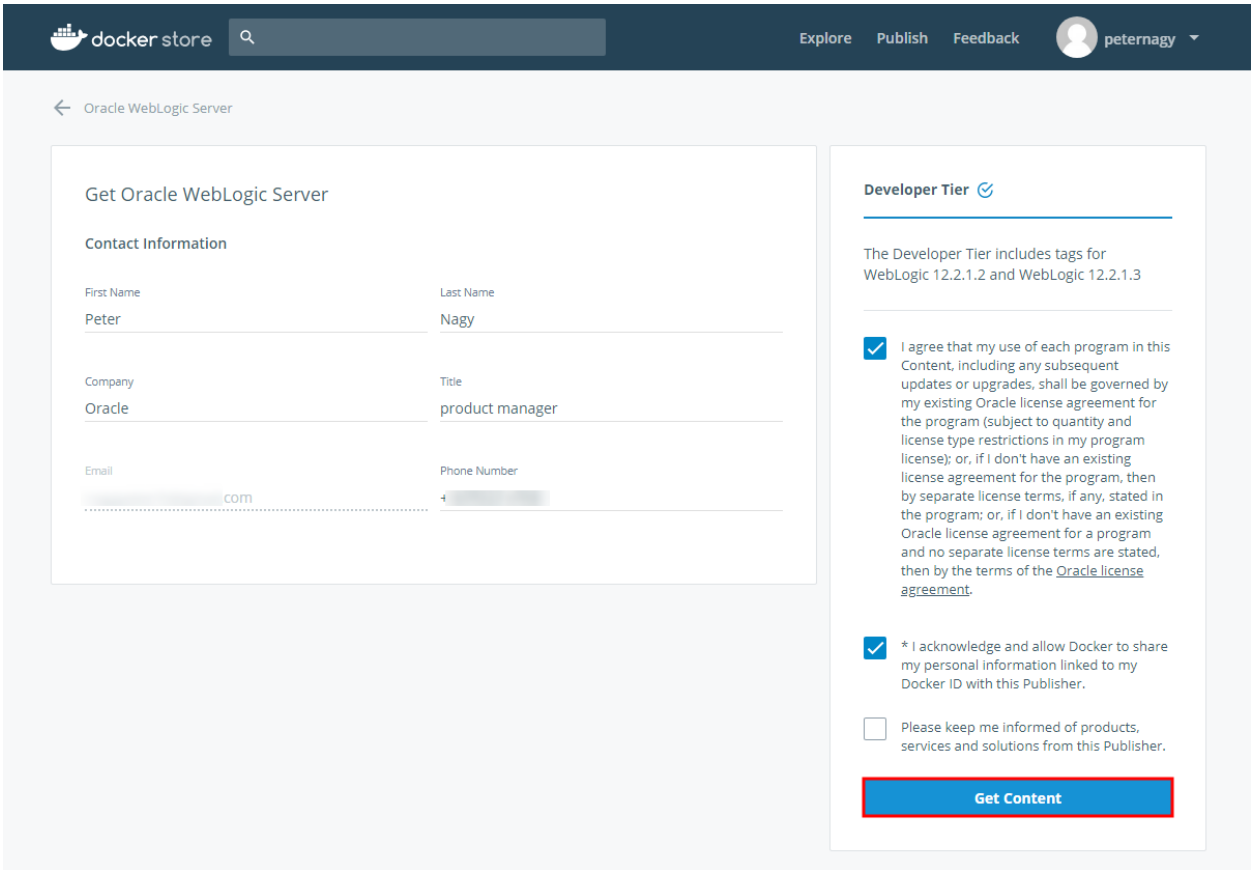
Enter your account details and click **Login**



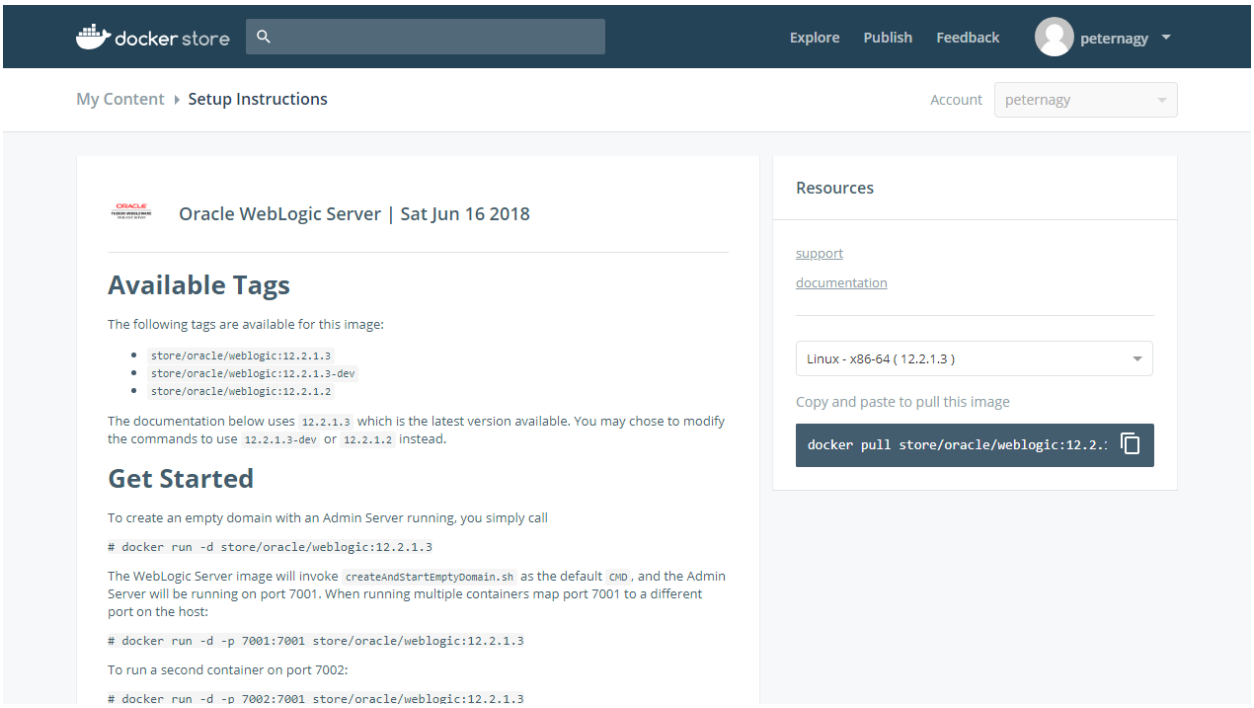
Click **Proceed to Checkout**.



Complete your contact information and accept agreements. Click **Get Content**.



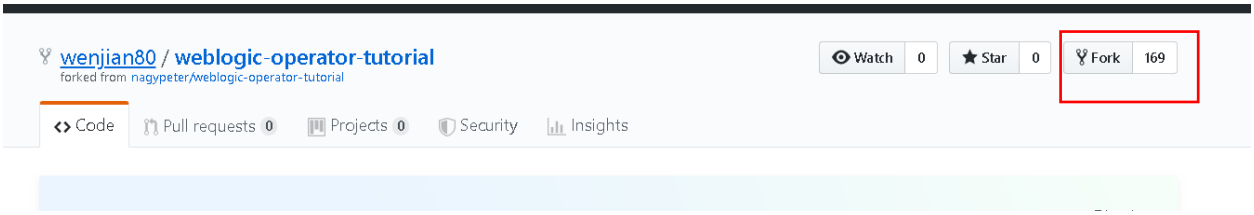
Now you are ready to pull the image on Docker enabled host after authenticating yourself in Docker Hub using your Docker Hub credentials.



Import WebLogic Operator Tutorial's source repository into your Github repository

In this step you will fork the tutorial's source repository. The source repository contains the demo application deployed on top of WebLogic server, configuration yaml to quickly create Oracle Container Pipelines(CI/CD) application to build custom WebLogic image and few additional Kubernetes configuration files to deploy the custom WebLogic image.

Open the https://github.com/wenjian80/weblogic-operator-tutorial.git repository in your browser. Click the **Fork** button at the left top area. Sign in to github.com if necessary.



Wait until the fork process is complete.

Create Oracle Container Pipelines Application to build custom WebLogic Docker container including demo application

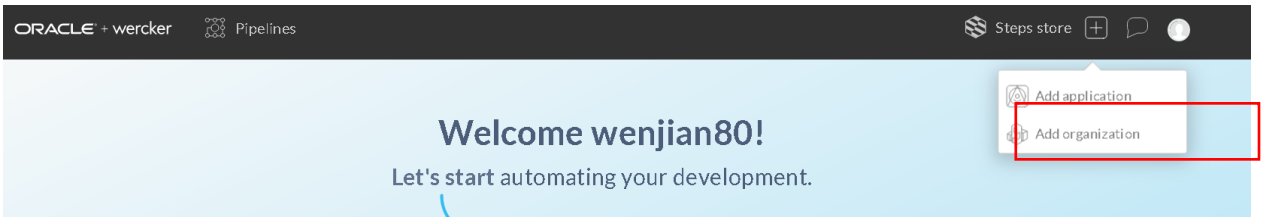
First create your Oracle Container Pipelines application. Oracle Container Pipelines acts as continuous integration tool which will produce WebLogic container image and uploads to Oracle Container Registry.

The following pipelines are predefined in the Oracle Container Pipelines configuration file (wercker.yml):

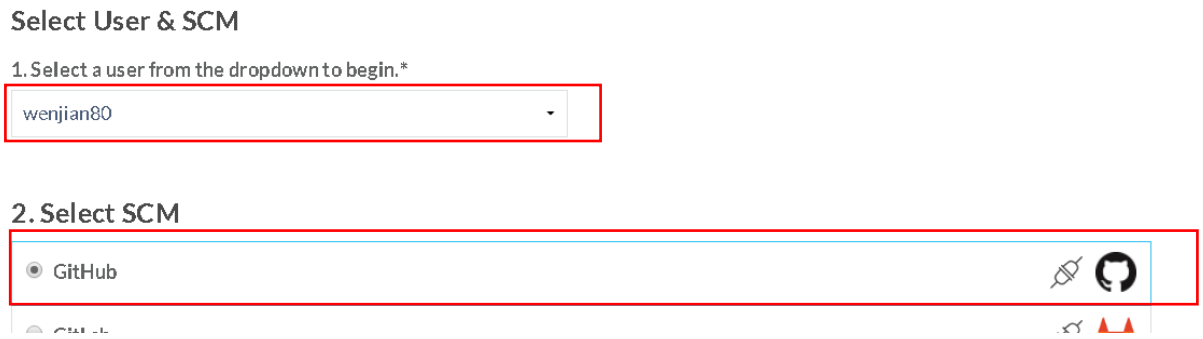
- **build**: Default and mandatory pipeline to start the workflow. It builds the demo Web Application using Maven.
- **build-domain-in-home-image**: Pipeline which runs Docker build to create custom WebLogic container image. First time when no *latest* image available in repository it uses official WebLogic image from Docker Store as base image and runs WLST script to customise the image. Also copies the demo Web Application into the image and deploys using WLST. Once *latest* (tag) of the image is available in the repository then the workflow just builds the Web Application and update the *latest* image with the new application binaries. After the Docker build the pipeline produces a new image and pushes to the image repository. Thus every time when changes happen in the sources and committed to Github. The image tag will be the commit hash tag of the source changes which triggered the new build process. Also the historically latest gets the *latest* tag as well.

Sign in to Oracle Container Pipelines (former Wercker) https://app.wercker.com and click **Create your first application** button or the **+** icon at the top right corner and select *Add Application*.

NOTE! If you need to sign up to Oracle Container Pipelines do it with your Github account. Click the **LOG IN WITH GITHUB**button and authorise Oracle Container Pipelines application for your Github account. You can revoke Oracle Container Pipelines's authorisation request anytime using your Github's profile settings.

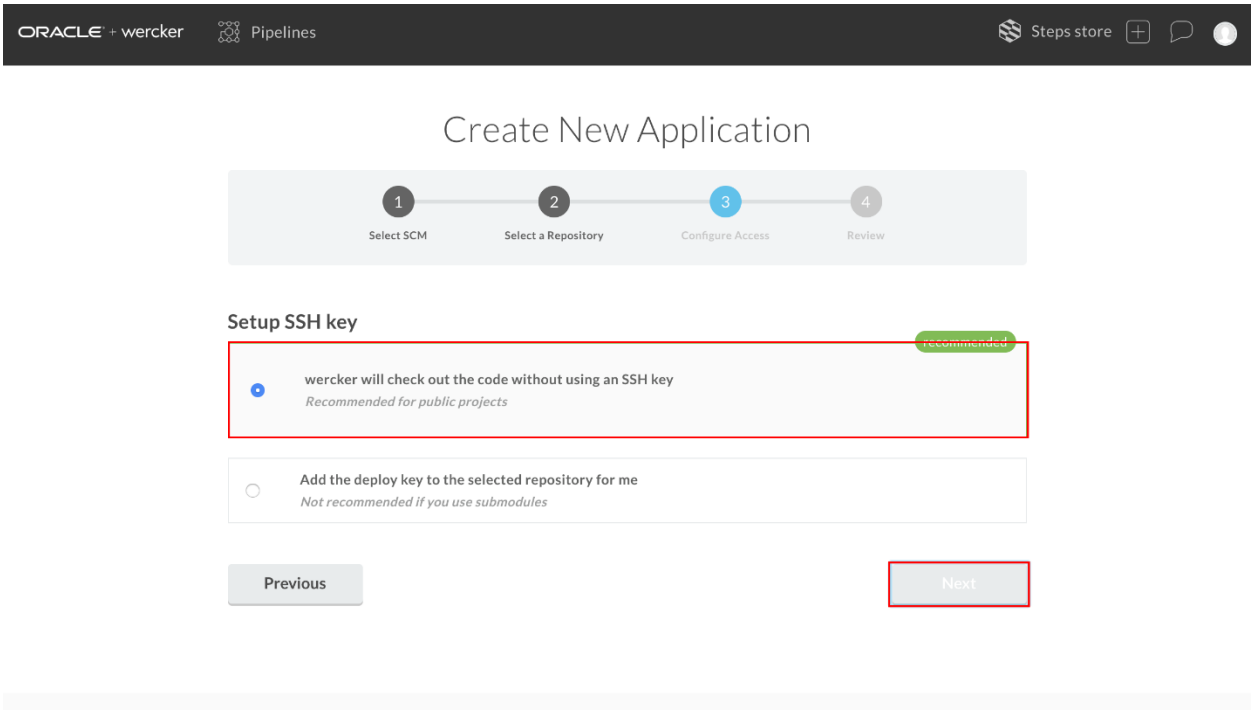


Select the owner of the application. By default it is your Oracle Container Pipelines username, but it can be any organization where you belong to. Make sure the selected SCM is *GitHub*. Click **Next**.



Select *weblogic-operator-tutorial* repository what you imported previously. Click **Next**.

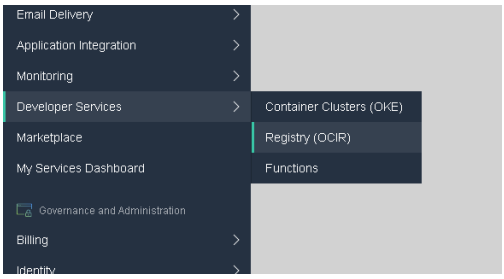
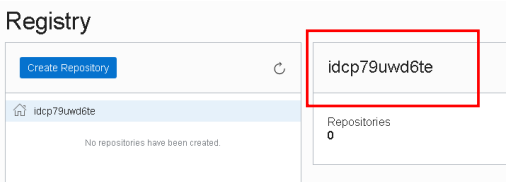
Leave the default repository access without SSH key. Click **Next**.



If you want you can make your application public if you want to share the application's status otherwise leave the default private settings. Click **Create**.

The repository already contains a necessary `wercker.yml` but before the execution provide the following key/value pairs:

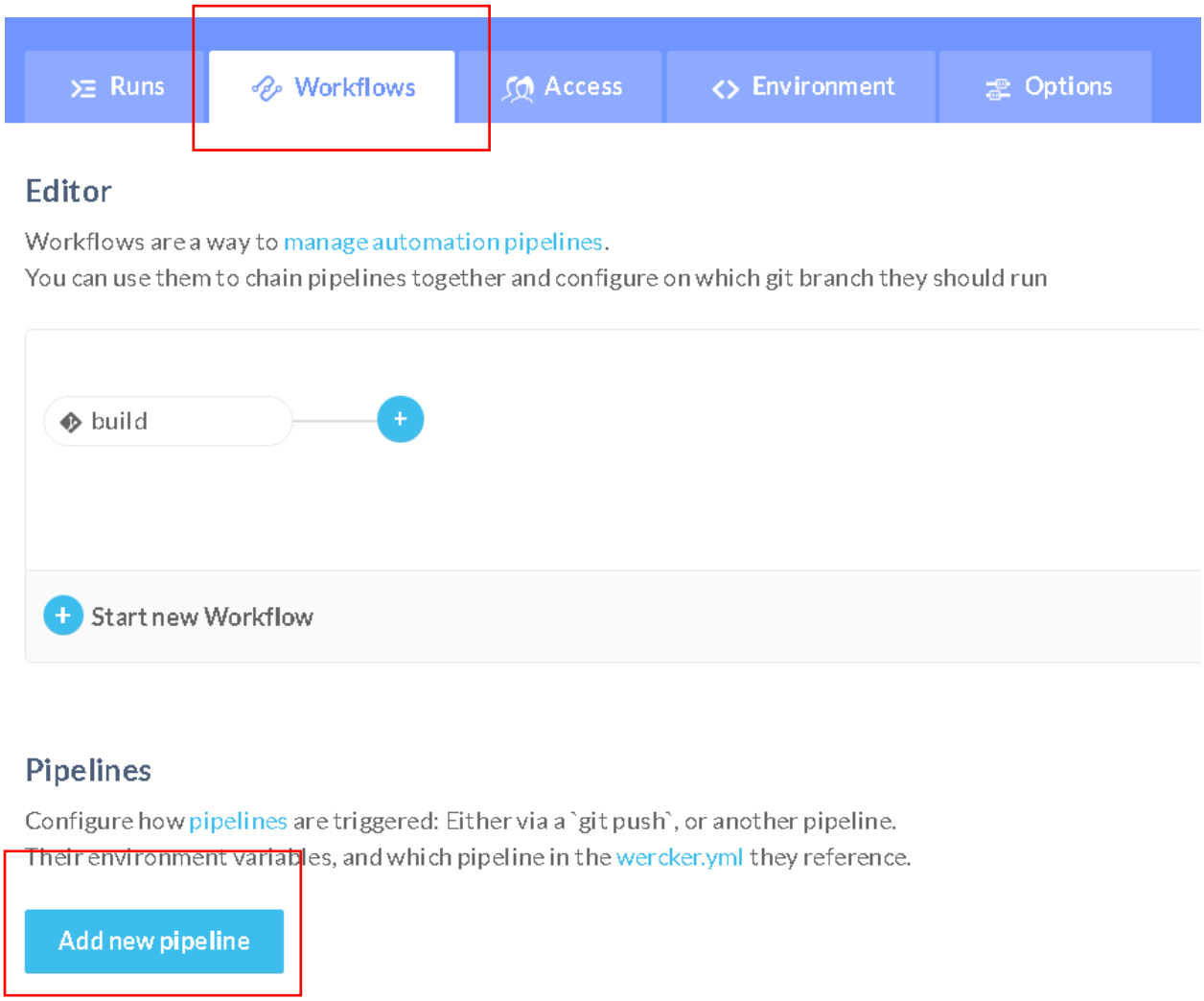
Key	**Value**	**Note**
OCI_REGISTRY_USERNAME	your_cloud_username	The username what you note during user settings. E.g. oracleidentitycloudservice/boonjiantrail@gmail.com
OCI_REGISTRY_PASSWORD	OCIR Auth Token	The Auth Token you generated previous steps. Eg: 8<Q5ISvUXrcY62r.os43 

TENANCY	Name of your tenancy	Check in oci console Go to developer services->OCIR   The name of the tenancy is idcp79uwd6te
REGION	The code of your home region. See the [documentation](#) to get your region code.	Iad for ashburn datacenter For this key in iad
DOCKER_USERNAME	Your Docker Hub username	Necessary to pull official WebLogic Server image from Docker Store Eg: bjlim80
DOCKER_PASSWORD	Your Docker Hub password	Necessary to pull official WebLogic Server image from Docker Store

To define these variables click **<>Environment** tab and enter keys and values. Remember that these values will be visible to anyone to whom you give access to the Oracle Container Pipelines application, therefore select **Protected** for any values that should remain hidden, including all passwords.

Key	Value		
OCI_REGISTRY_USERNAME	oracleidentitycloudservice/ boonjiantrail@gmail.com	Protected	Delete
OCI_REGISTRY_PASSWORD	8<Q5ISvUXrcY62r.os43		Delete
TENANCY	boonjiantrail		Delete
REGION	iad		Delete
DOCKER_USERNAME	bjlim80		Delete
DOCKER_PASSWORD	*Protected*		Delete

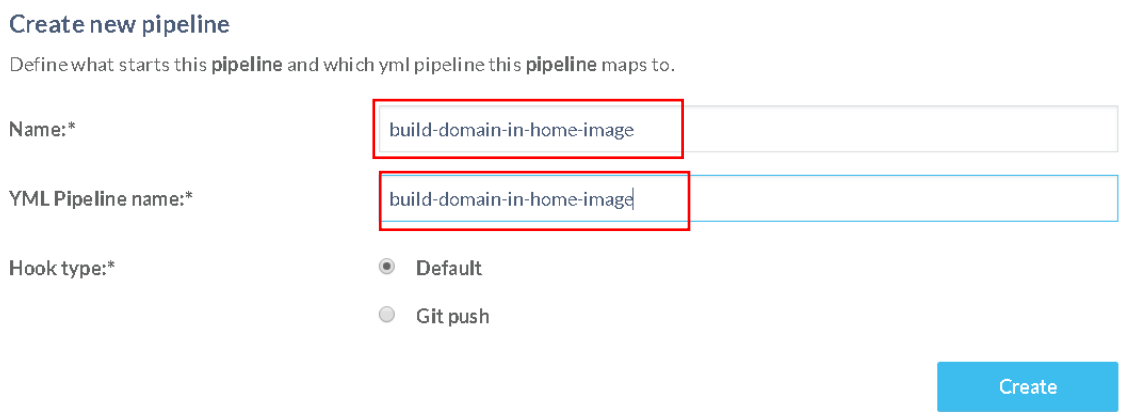
Click the **Worklflow** tab and then **Add new pipeline** to enable pipeline defined in *wercker.yml*.



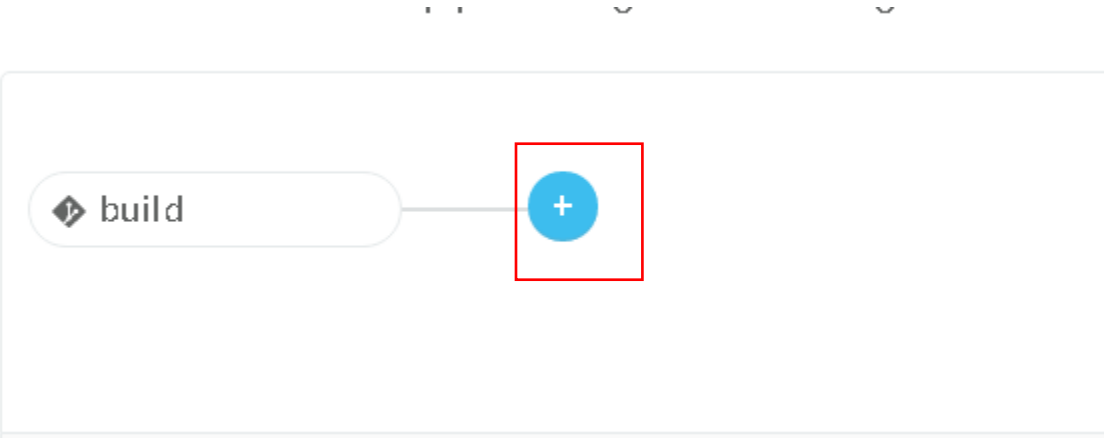
Enter the name of the pipeline and the "YML Pipeline Name" as *build-domain-in-home-image*.

Please enter exactly this name - because this name is hardcoded in the *wercker.yml*.

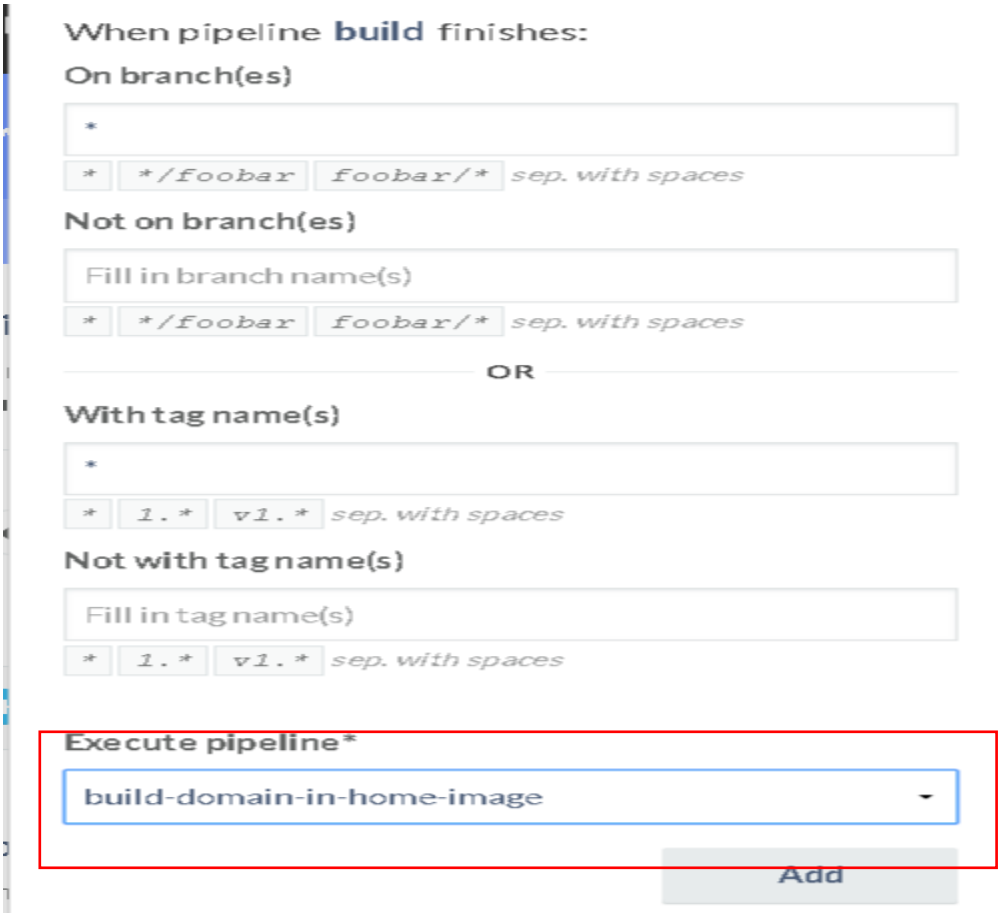
Click **Create**.



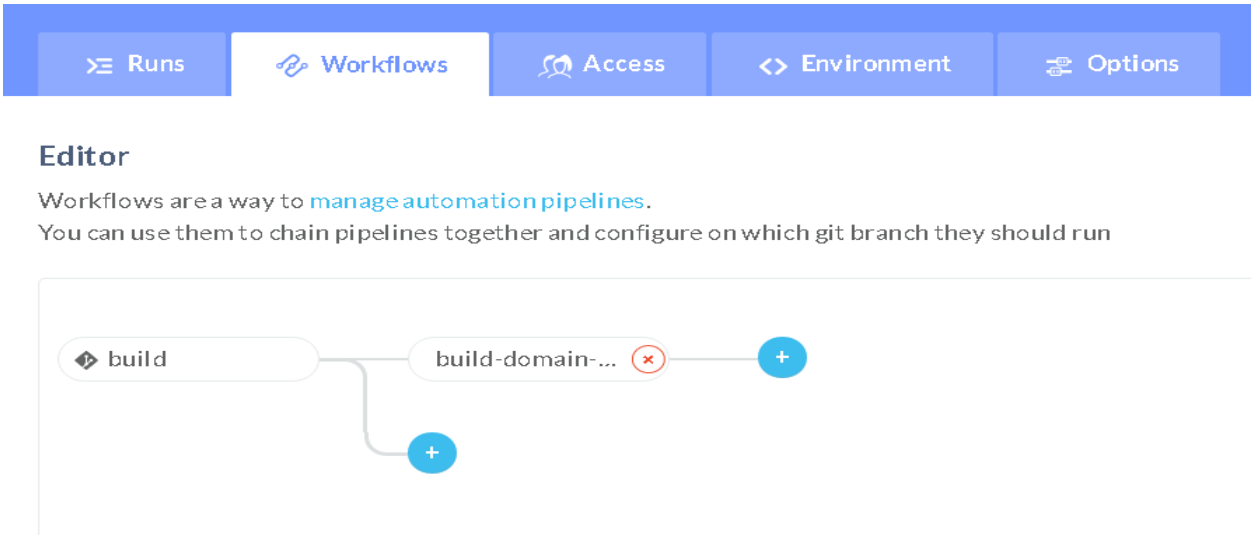
Click again the **Worklflow** tab to get back to the editor page. Click the + sign after the mandatory *build* pipeline.



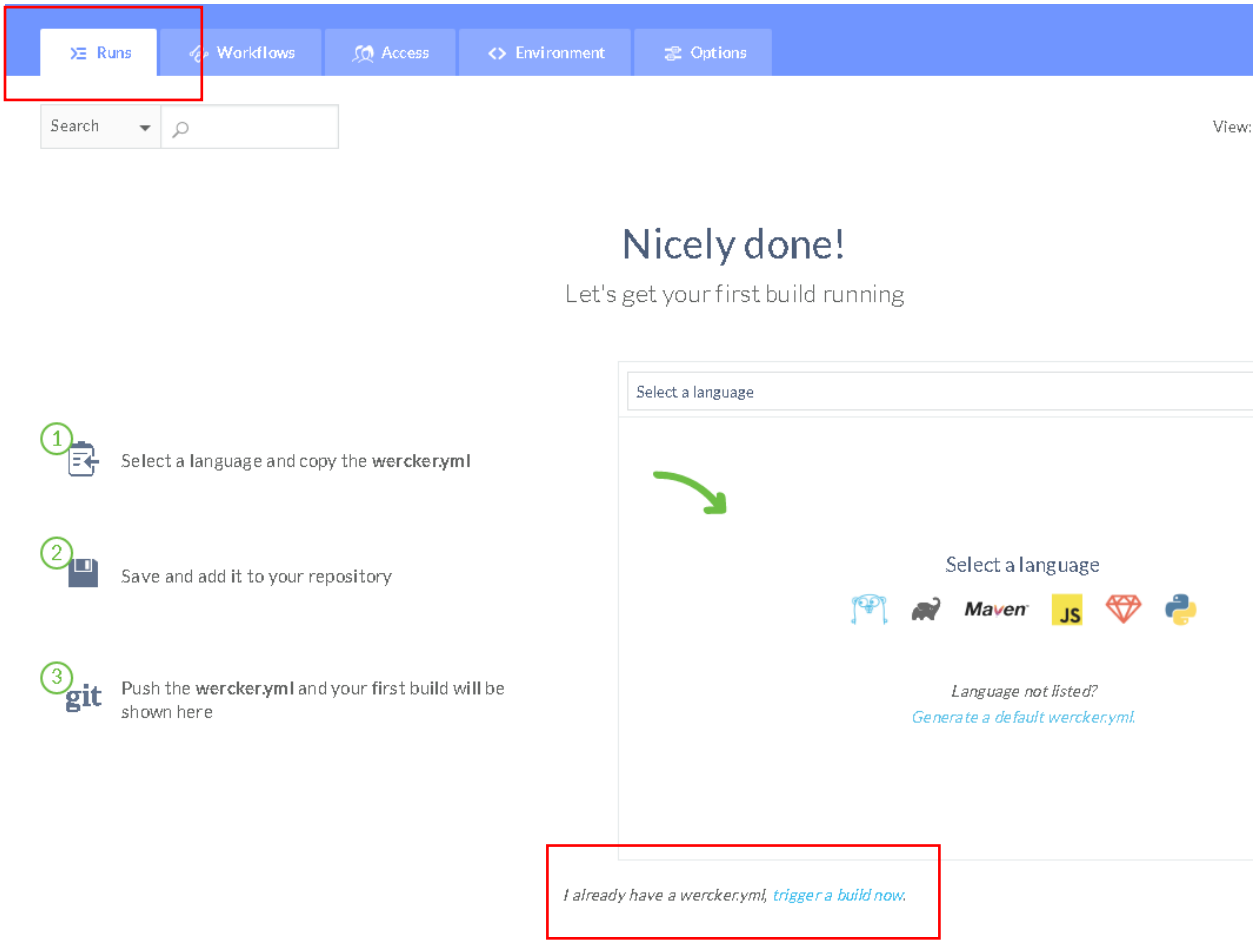
Leave the default branch(es) configuration and select the *build-domain-in-home-image* pipeline.



Your workflow should be similar:

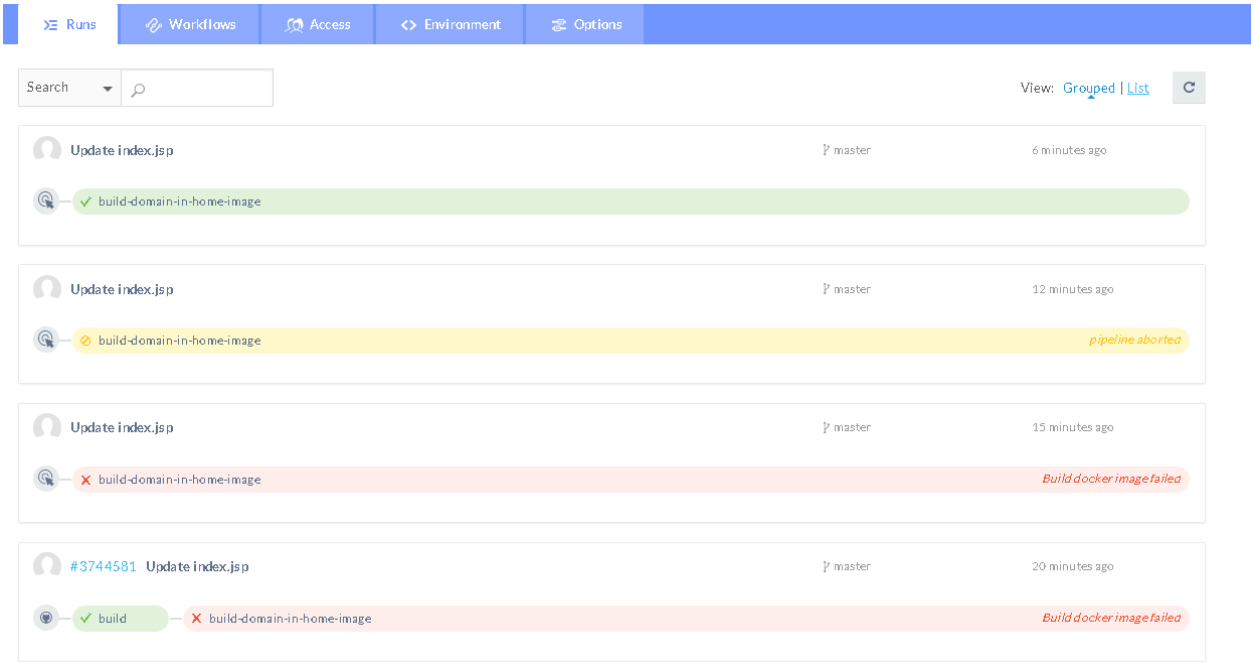


Go to the **Runs** tab and click ***trigger a build now*** link.

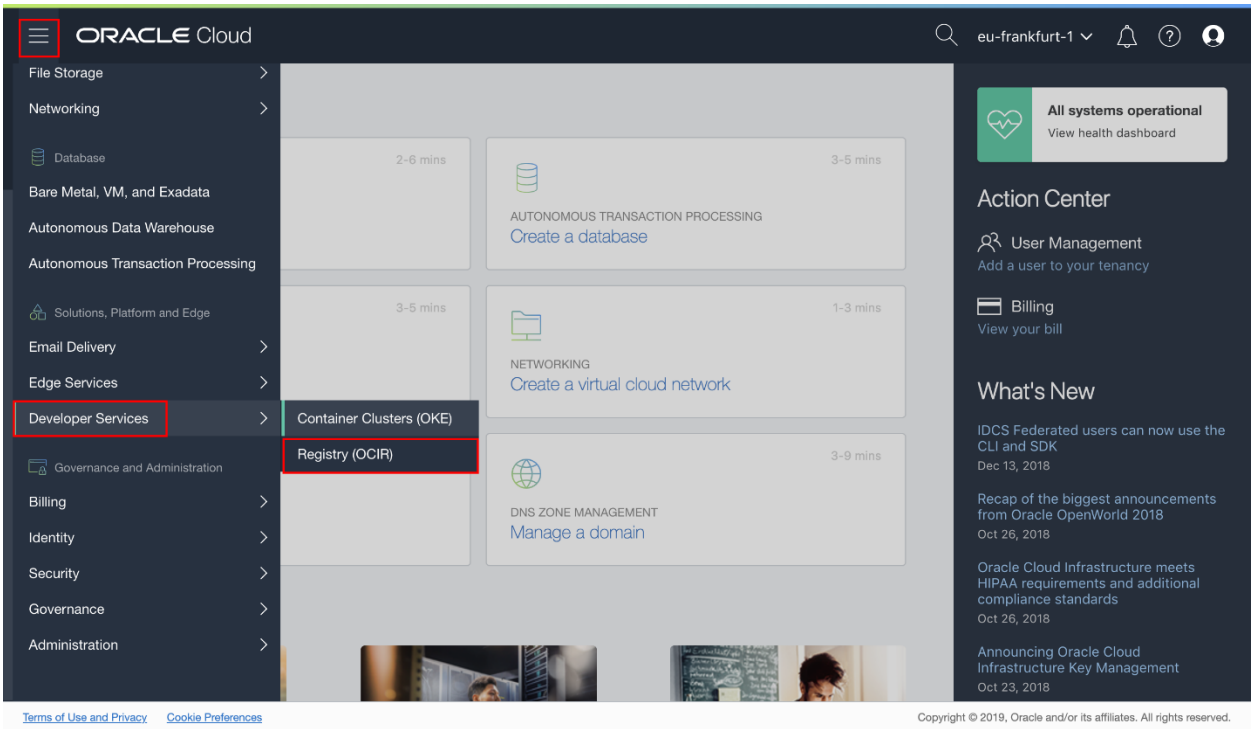


To get more details about the current step click on the pipeline.

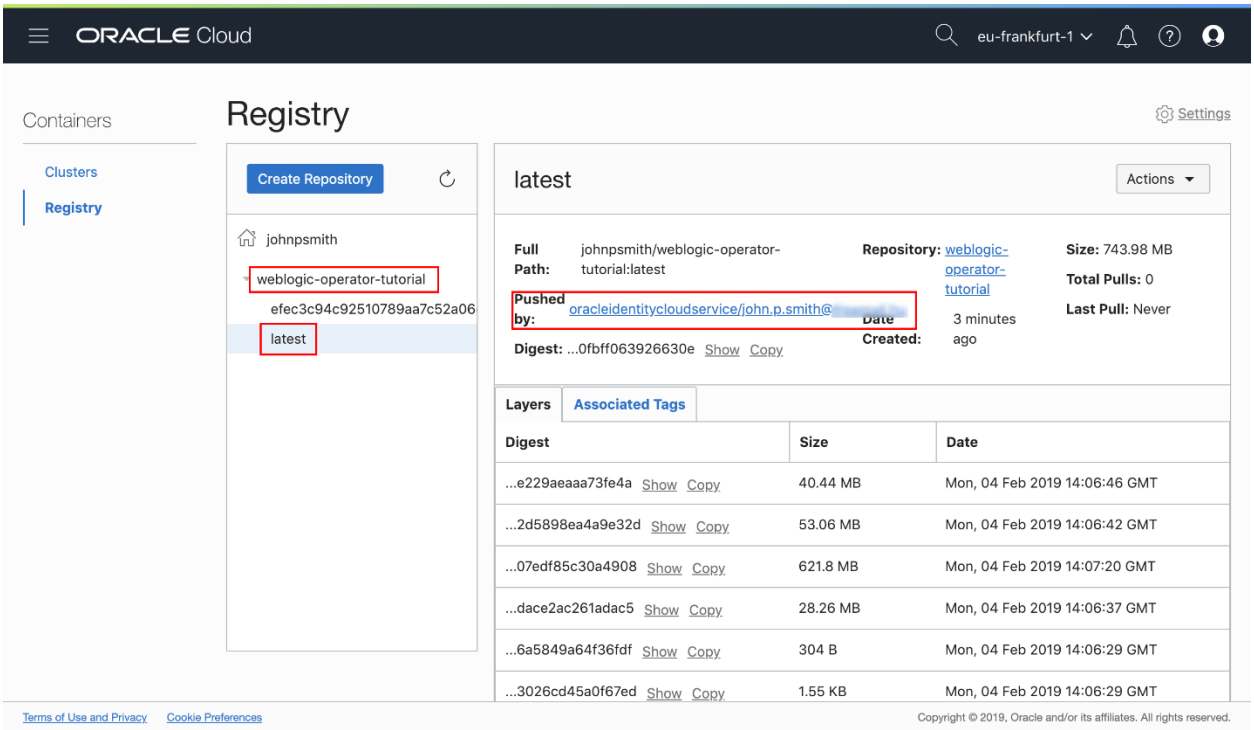
When the workflow is completed the WebLogic image is available in your image repository.



Open the OCI console page and go to the container registry console to check.



In the registry you have to find a repository named like your Oracle Container Pipelines application (e.g. *weblogic-operator-tutorial*). If you open the repository for more details you find two images. Technically the two images are the same, but got two tags. One of them is the git commit hash tag which is uniquely identify the image. The second *latest* tag applied because to have easier access to the historically latest release/image.

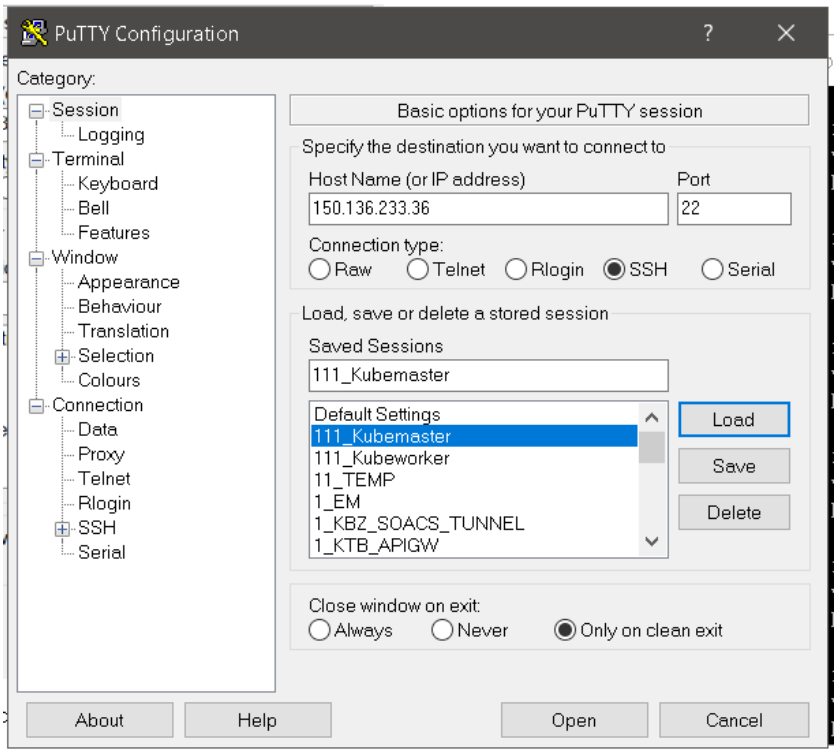


Now the WebLogic domain image is ready to deploy on Kubernetes using WebLogic Operator.

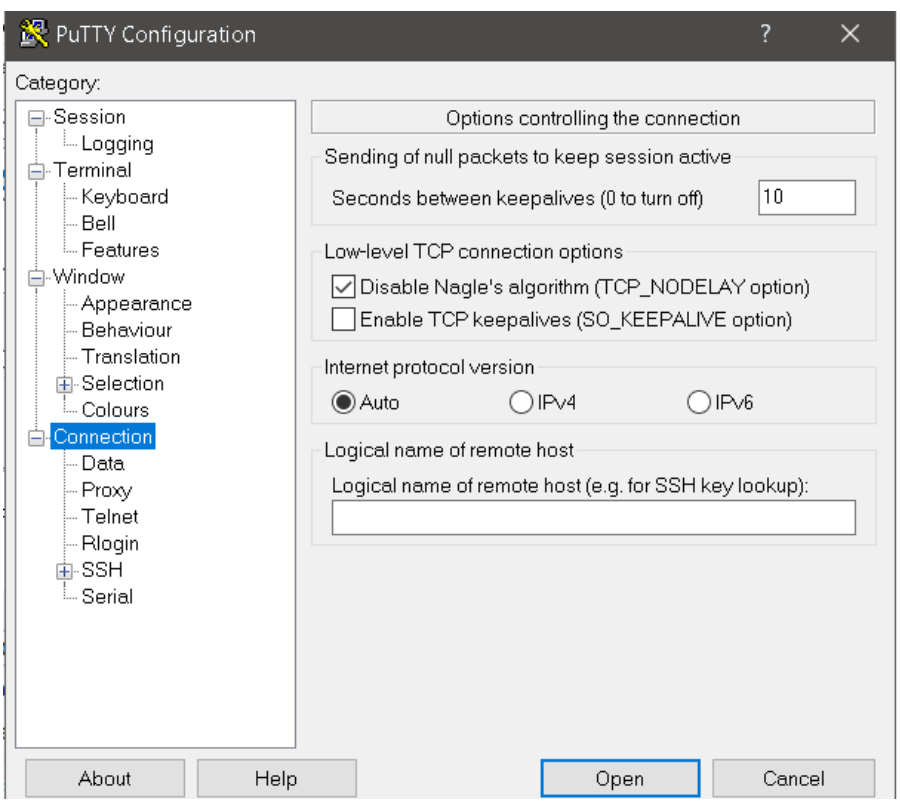
Setting Up Tools

Login into the **master node** using putty

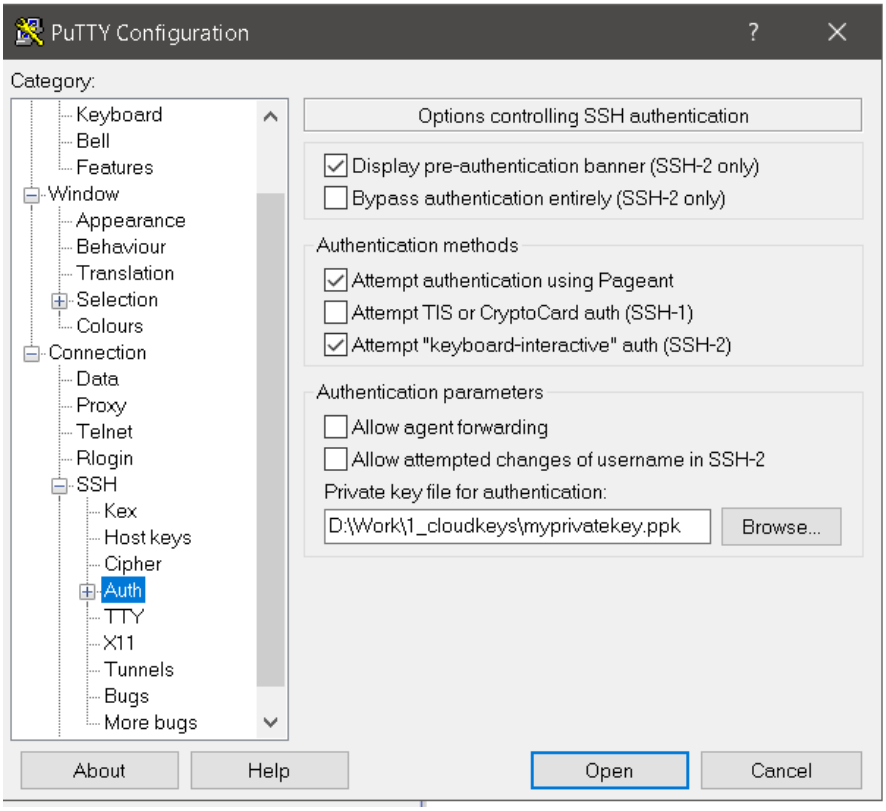
Enter the Ip address of master ip



Set connection greater 10 to make connection live even if we are idle



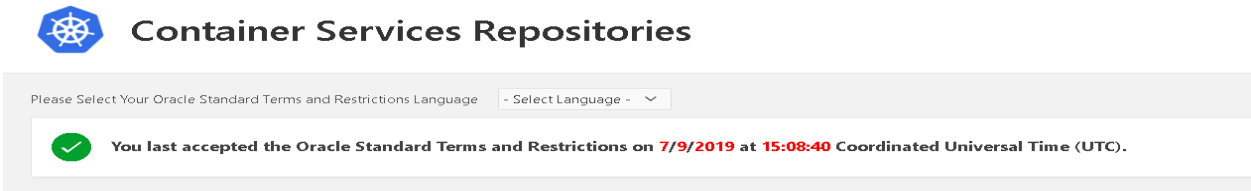
Add private key to login



Open the Session

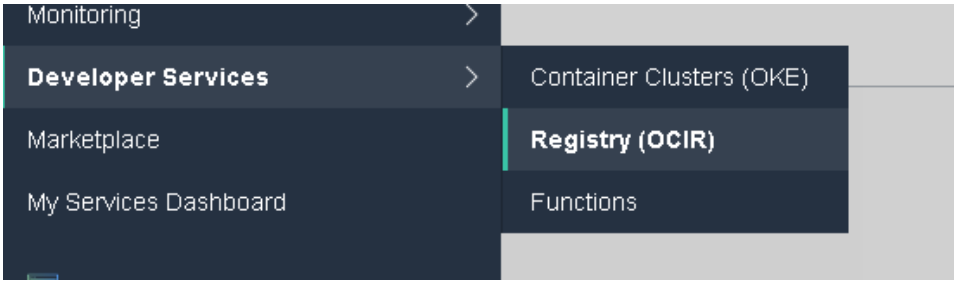
Install Git and Helm and Tiller (Cluster Side)

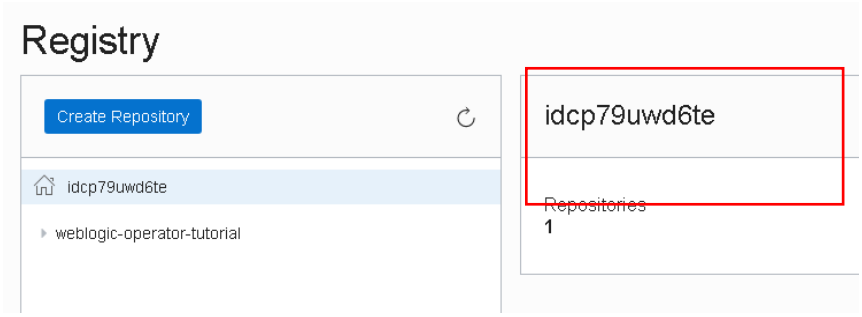
1. Make sure kube_lab has already copied to /home/opc
2. Login to container-registry.oracle.com using your oracle sso account that you have created.
3. Click on
    a. Container Services (Developer) Repositories and make sure you have accepted the agreement.
    b. Container Services Repositories and make sure you have accepted the agreement.



4. Open **9_environment.sh** and amend the values according.
5. Change values in ocir tenacy

```
#add your machine tenancy
#Change to you tenancy of the ocirpass
#Go to developer service->ocir and note down the name
export ocirtenancy=idcp79uwd6te
```





6. Change to your oracle sso account

```
#add your container userName , Password
#Change to your oracle sso account.
#This is the uid/password to login to container-registry.oracle.com to pull down the images.
#Make sure you have accept the agreement in container-registry.oracle.com website
export ocontaineruser=boonjiantrail@gmail.com
export ocontainerpassword=Welc0me1234#
```

7. Change the registry details. If the password has special character such as < append it with \<
Eg the password token variable will be export ocirpass="8\<Q5ISvUXrcY62r.os43" otherwise the variable cannot be set in unix

```
#add your ociruser id
#Change to user email id and token that is created in the lab
#Put a \ if token has a special chracter Eg \<
export ociruser=$ocirtenancy/oracleidentitycloudservice/boon.jian.lim@oracle.com
export ocirpass="\>-6oPo+vAfgsgPJMv.\{L"
```

8. Run **./9_environment.sh** to set the environments as root in the master node

9. Run below to check all variables are set, there should not be blank.

   source /root/.bash

   echo 'ociruser='$ociruser
   echo 'ocirpass='$ocirpass
   echo 'ocontaineruser='$ocontaineruser
   echo 'ocontainerpassword='$ocontainerpassword

```
[root@master kube_lab]# soruce /root/.bashrc
bash: soruce: command not found
[root@master kube_lab]# source /root/.bashrc
[root@master kube_lab]# echo 'ociruser='$ociruser
ociruser=idlicap3m15d/oracleidentitycloudservice/boon.jian.lim@oracle.com
[root@master kube_lab]# echo 'ocirpass='$ocirpass
ocirpass=8C1OuDQluSj({e_<fRQ-
[root@master kube_lab]# echo 'ocontaineruser='$ocontaineruser
ocontaineruser=boonjiantest@gmail.com
[root@master kube_lab]# echo 'ocontainerpassword='$ocontainerpassword
ocontainerpassword=Welc0me1234#
[root@master kube_lab]#
```

Install Git and helm

./**10_tools.sh**

Output

```
The below output
NAME                                READY   STATUS    RESTARTS   AGE
coredns-b7df4d4c4-7zn5g             1/1     Running   0          29h
coredns-b7df4d4c4-fnpql             1/1     Running   0          29h
etcd-kubemaster                     1/1     Running   0          29h
kube-apiserver-kubemaster           1/1     Running   0          29h
kube-controller-manager-kubemaster  1/1     Running   0          29h
kube-flannel-ds-5w7wb               1/1     Running   0          29h
```

```
kube-flannel-ds-7z8cf              1/1     Running   0          29h
kube-proxy-2gzn5                   1/1     Running   0          29h
kube-proxy-v9pp4                   1/1     Running   0          29h
kube-scheduler-kubemaster          1/1     Running   0          29h
kubernetes-dashboard-669df9cb5d-bnpml   1/1     Running   0          29h
tiller-deploy-694dc94c65-x7s6c     1/1     Running   0          39m

git version 1.8.3.1
Client: &version.Version{SemVer:"v2.9+unreleased", GitCommit:"", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.9.1",
GitCommit:"20adb27c7c5868466912eebdf6664e7390ebe710", GitTreeState:"clean"}
```

Update helm repo

./ 11_helmchart.sh

Install WeblogicOperator

```
./12_weblogicoperator.sh
```

Output

```
Cloning into 'weblogic-kubernetes-operator'...
remote: Enumerating objects: 395, done.
remote: Counting objects: 100% (395/395), done.
remote: Compressing objects: 100% (177/177), done.
remote: Total 80055 (delta 155), reused 312 (delta 88), pack-reused 79660
Receiving objects: 100% (80055/80055), 60.70 MiB | 37.15 MiB/s, done.
Resolving deltas: 100% (47057/47057), done.
clusterrolebinding.rbac.authorization.k8s.io/helm-user-cluster-admin-role created
namespace/sample-weblogic-operator-ns created
serviceaccount/sample-weblogic-operator-sa created
NAME:   sample-weblogic-operator
LAST DEPLOYED: Fri Jul 12 17:11:56 2019
NAMESPACE: sample-weblogic-operator-ns
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/Deployment
NAME            DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
```

```
weblogic-operator  1       1       1         0        0s

==> v1/ConfigMap
NAME                DATA  AGE
weblogic-operator-cm  2     0s

==> v1/ClusterRole
NAME                                                    AGE
sample-weblogic-operator-ns-weblogic-operator-clusterrole-general        0s
sample-weblogic-operator-ns-weblogic-operator-clusterrole-domain-admin    0s
sample-weblogic-operator-ns-weblogic-operator-clusterrole-namespace       0s
sample-weblogic-operator-ns-weblogic-operator-clusterrole-operator-admin  0s
sample-weblogic-operator-ns-weblogic-operator-clusterrole-nonresource     0s

==> v1/Role
weblogic-operator-role  0s

==> v1/Service
NAME                        TYPE      CLUSTER-IP     EXTERNAL-IP  PORT(S)   AGE
internal-weblogic-operator-svc  ClusterIP  10.105.197.246  <none>      8082/TCP  0s

==> v1/Pod(related)
NAME                          READY  STATUS           RESTARTS  AGE
weblogic-operator-7cf996998b-n8nlp  0/1    ContainerCreating  0       0s

==> v1/Secret
NAME                   TYPE    DATA  AGE
weblogic-operator-secrets  Opaque  0     0s

==> v1/ClusterRoleBinding
NAME                                                      AGE
sample-weblogic-operator-ns-weblogic-operator-clusterrolebinding-general        0s
sample-weblogic-operator-ns-weblogic-operator-clusterrolebinding-discovery      0s
sample-weblogic-operator-ns-weblogic-operator-clusterrolebinding-auth-delegator  0s
sample-weblogic-operator-ns-weblogic-operator-clusterrolebinding-nonresource     0s

==> v1/RoleBinding
NAME                            AGE
weblogic-operator-rolebinding-namespace  0s
weblogic-operator-rolebinding          0s


NAME                          READY   STATUS           RESTARTS   AGE
weblogic-operator-7cf996998b-n8nlp   0/1     ContainerCreating   0          20s
NAME                      REVISION       UPDATED                    STATUS        CHART
NAMESPACE
sample-weblogic-operator       1            Fri Jul 12 17:11:56 2019       DEPLOYED       weblogic-
operator-2.2.1 sample-weblogic-operator-ns
```

Install Traefik

Run ./**13_traefik.sh**

Output

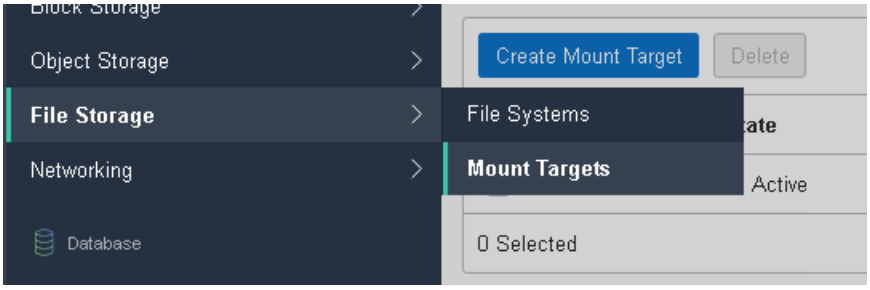
```
NAME:   traefik-operator
LAST DEPLOYED: Fri Jul 12 17:27:51 2019
NAMESPACE: traefik
STATUS: DEPLOYED

RESOURCES:
==> v1/Service
NAME                    TYPE      CLUSTER-IP     EXTERNAL-IP  PORT(S)                AGE
traefik-operator-dashboard  ClusterIP  10.102.214.104  <none>      80/TCP                 0s
traefik-operator          NodePort   10.109.148.238  <none>      80:30305/TCP,443:30443/TCP  0s

==> v1beta1/Ingress
NAME                    HOSTS             ADDRESS  PORTS  AGE
traefik-operator-dashboard  traefik.example.com  80      0s

==> v1/Secret
NAME                      TYPE    DATA  AGE
traefik-operator-default-cert  Opaque  2     0s

==> v1/ConfigMap
NAME                DATA  AGE
traefik-operator      1     0s
traefik-operator-test  1     0s

==> v1/ClusterRoleBinding
NAME            AGE
traefik-operator  0s

==> v1/Deployment
NAME            DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
traefik-operator  1       1       1         0        0s

==> v1/Pod(related)
NAME                        READY  STATUS          RESTARTS  AGE
traefik-operator-789cf75456-gts5g  0/1    ContainerCreating  0       0s
```

```
==> v1/ServiceAccount
NAME            SECRETS  AGE
traefik-operator  1       0s

==> v1/ClusterRole
NAME            AGE
traefik-operator  0s


NOTES:

1. Traefik is listening on the following ports on the host machine:

    http - 30305
    https - 30443

2. Configure DNS records corresponding to Kubernetes ingress resources to point to the
NODE_IP/NODE_HOST

NAME                    TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)                  AGE
traefik-operator        NodePort   10.109.148.238  <none>        80:30305/TCP,443:30443/TCP   20s
traefik-operator-dashboard   ClusterIP   10.102.214.104   <none>        80/TCP                   20s
NAME                    REVISION       UPDATED                   STATUS        CHART
NAMESPACE
sample-weblogic-operator       1            Fri Jul 12 17:11:56 2019       DEPLOYED       weblogic-
operator-2.2.1 sample-weblogic-operator-ns
traefik-operator               1            Fri Jul 12 17:27:51 2019       DEPLOYED       traefik-1.70.2
traefik
```

Create PV and PVC

Get the ip of the of the nfs server which you have created earlier in the labs.

- Login to oracle cloud
- Go to the services -> select file storage -> Mount targets



- Jot down the ip address and you will update the yaml file with this ip later.


```
#Create a directory in shared folder
mkdir -p /shared/logs/sampledomain

#Change the nfs ip to the ip that is given
vi weblogic-sample-pv.yaml
server: 10.0.0.5 – change the ip address of your NFS Server ip

#Run this script
./14_pv-pvc.sh
```

OutPut

```
persistentvolume/weblogic-sample-pv created
persistentvolumeclaim/weblogic-sample-pvc created
NAME                STATUS   VOLUME              CAPACITY   ACCESS MODES   STORAGECLASS
AGE
weblogic-sample-pvc   Bound    weblogic-sample-pv   10Gi       RWX           weblogic-sample-storage-
class   62s
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS                REASON   AGE
```

weblogic-sample-pv	10Gi	RWX	Retain	Bound	sample-domain1-ns/weblogic-sample-pvc	weblogic-sample-storage-class	62s

Create Domain

1. Run "echo $ocirpass" to check password is it set in the environment.

```
[root@master kube_lab]# echo $ocirpass
8<Q5ISvUXrcY62r.os43
```

If password is not set run below command to set the password
If there is any special character, append a "\" in the special character.

```
[root@master kube_lab]#  export ocirpass=8\<Q5ISvUXrcY62r.os43
[root@master kube_lab]# echo $ocirpass
8<Q5ISvUXrcY62r.os43
[root@master kube_lab]#
```

2. Run . /15_prepare_domaininfo.sh to set the necessary secrets

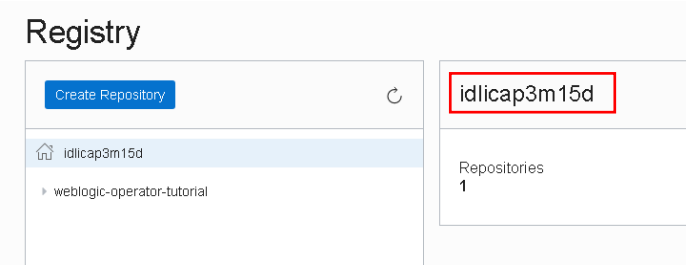
3. We are going to generate a domain home in image based on the below instructions

https://github.com/oracle/weblogic-kubernetes-operator/blob/master/docs-source/content/samples/simple/domains/domain-home-in-image/_index.md

4. Open mysample_domaininputs.yaml in /home/opc/kube_lab and edit the image name the point to your docker tenacy.

```
#WebLogic Server Docker image that the operator uses to start the domain.
#The create domain scripts generate a WebLogic Server Docker image with a domain home in it.
#By default, the scripts tag the generated WebLogic server Docker image as either domain-home-in-image or domain-home-in-image-wdt based on the
#and use it plus the tag that is obtained from the domainHomeImageBase to set the image element in the generated domain YAML file.
#If this property is set, the create domain scripts will use the value specified, instead of the default value, to tag the generated image and
#set the image in the domain YAML file. A unique value is required for each domain that is created using the scripts.
#If you are running the sample scripts from a machine that is remote to the Kubernetes cluster where the domain is going to be running,
#you need to set this property to the image name that is intended to be used in a registry local to that Kubernetes cluster.
#You also need to push the image to that registry before starting the domain using the kubectl create -f or kubectl apply -f command.
image: iad.ocir.io/idlicap3m15d/weblogic-operator-tutorial:latest
```

You can check you docker tenacy by going to Developer Services->OCIR



5. Copy the domain inputs to the folder to generate the config with the following command

cp /home/opc/kube_lab/mysample_domaininputs.yaml /home/opc/kube_lab/weblogic-kubernetes-operator/kubernetes/samples/scripts/create-weblogic-domain/domain-home-in-image

6. Go to the weblogic operator scripts directory

cd /home/opc/kube_lab/weblogic-kubernetes-operator/kubernetes/samples/scripts/create-weblogic-domain/domain-home-in-image

7. Run below command to generate the output

./create-domain.sh -i mysample_domaininputs.yaml -o /home/opc/kube_lab/generateconfig/ -u weblogic -p welcome1

When finished you will see the below output

```
T3 access is available at t3://10.0.0.2:30012
The following files were generated:
  /home/opc/kube_lab/generateconfig//weblogic-domains/sample-domain1/create-domain-inputs.yaml
  /home/opc/kube_lab/generateconfig//weblogic-domains/sample-domain1/domain.yaml
```

8. Run below command to change the rights and copy the domain yaml to /home/opc/kube_lab folder

chown opc:opc /home/opc/kube_lab/generateconfig

chmod -R 777 /home/opc/kube_lab/generateconfig

cp /home/opc/kube_lab/generateconfig/weblogic-domains/sample-domain1/domain.yaml /home/opc/kube_lab

9. Edit /home/opc/kube_lab /domain.yaml to include a version and imagepullpolicy to always

```
  # The in-pod name of the directory to store the domain, nc
  # files in.
  # If not specified or empty, domain log file, server logs,
  # will be stored in the default logHome location of /share
  # serverStartPolicy legal values are "NEVER", "IF_NEEDED",
  # This determines which WebLogic Servers the Operator will
  # - "NEVER" will not start any server in the domain
  # - "ADMIN_ONLY" will start up only the administration ser
  # - "IF_NEEDED" will start all non-clustered servers, incl
  serverStartPolicy: "IF_NEEDED"
  restartVersion: "v1"
```

```
  # imagePullPolicy defaults to "A
  imagePullPolicy: "Always"
  # Identify which Secret contains
```

10. Run script below to create the domain
./15_createdomain.sh

Validate and wait until the admin server and managed server po is up and running.

You can also go to /shared/logs/sampledomain to look a t the logs.

```
kubectl get pod -n sampledomain
NAME                                        READY     STATUS              RESTARTS
AGE
sample-domain1-introspect-domain-job-kcn4n   0/1       ContainerCreating   0
7s
kubectl get po -n sampledomain -o wide
```

```
kubectl get po -n sampledomain -o wide
NAME                              READY     STATUS    RESTARTS   AGE       IP
NODE            NOMINATED NODE
sample-domain1-admin-server       1/1       Running   0          2m
10.244.2.10   130.61.84.41    <none>
sample-domain1-managed-server1    1/1       Running   0          1m
10.244.2.11   130.61.84.41    <none>
```

Access the Weblogic Application

http://<<public_ip_worker_node>>:30305/console/login/LoginForm.jsp to access weblogic application console. The username and password is weblogic/welcome1

Eg

http://132.145.206.79:30305/console

Access Sample App

http:// <<public_ip_worker_node>>:30305/opdemo/

Eg

http://132.145.206.79:30305/opdemo

WebLogic Server on Docker - Request Information

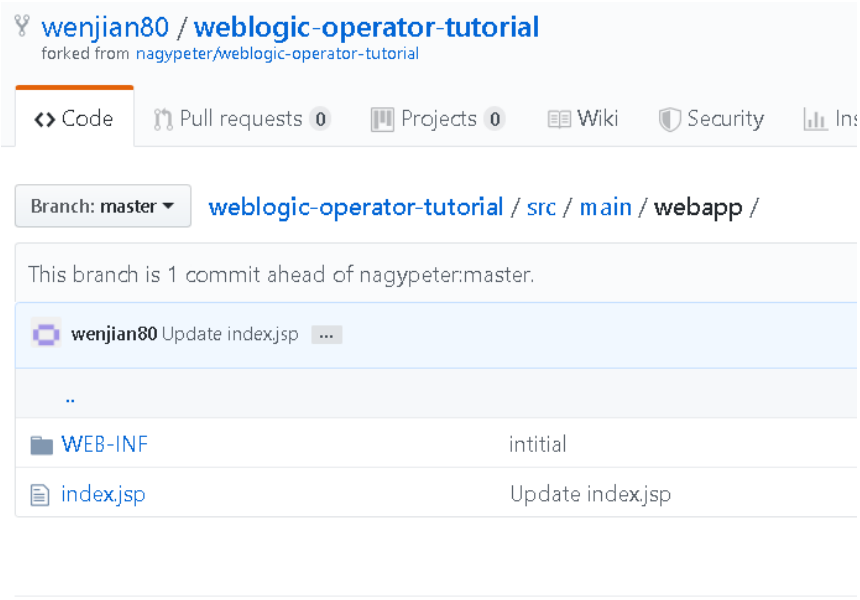
- getVirtualServerName(): managed-server1
- InetAddress.hostname: sample-domain1-managed-server1
- InetAddress.serverAddress: sample-domain1-managed-server1/10.244.1.54
- getLocalAddr(): 10.244.1.54
- getLocalName(): sample-domain1-managed-server1
- getLocalPort(): 8001
- getServerName(): 132.145.206.79
- WLS Server Name: managed-server1
- getIpAddOfCurrSrv():

Update the sample app image

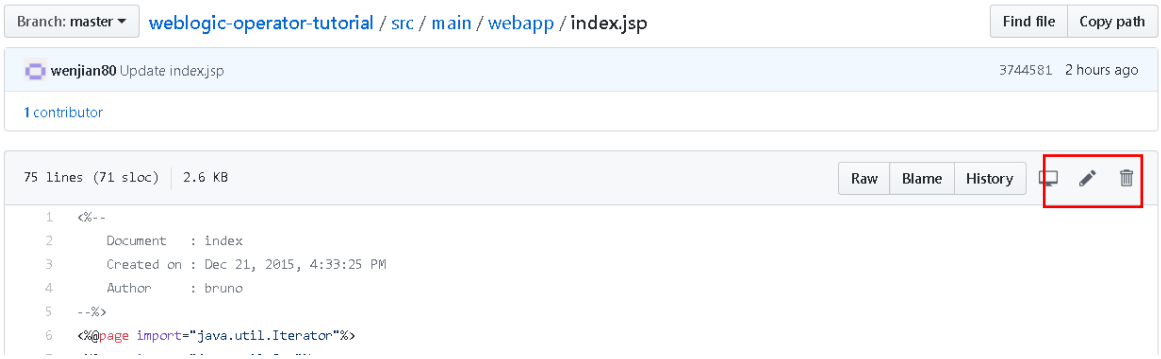
As discuss in the lesson earlier there are 2 deployment approach (traditional approach of deployment and burning the deployment in the image). The below steps will demonstrate how application are burn into the image

The below steps will show how the web application is updated which will trigger the wrecker build and push the images to registry. We will then update the image to be pull from the domain.yaml which will reflect the latest changes.

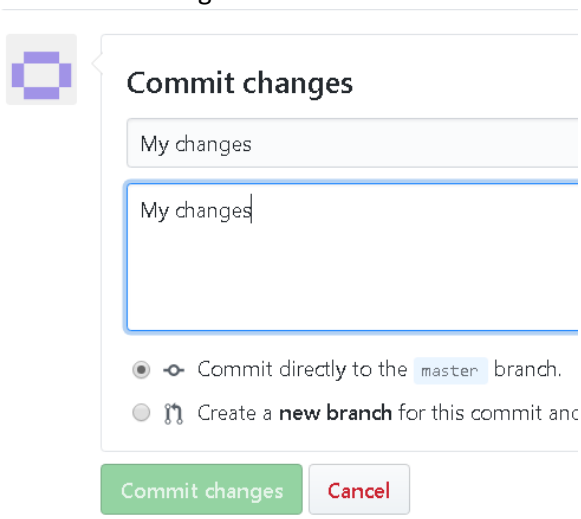
11. Login to your git and navigate to weblogic-operator-tutorial/src/main/webapp/



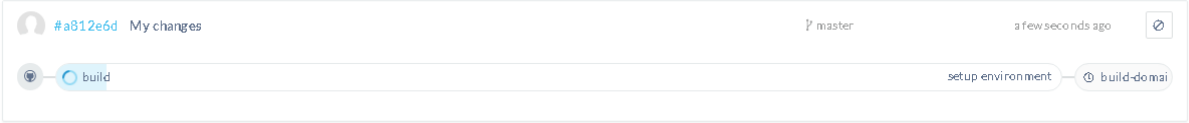
12. Open index.jsp click on edit



13. Commit the changes



14. Login in wrecker and you will see a build being trigger.



15. Login to oracle cloud and a new image is push to the registry. And take note of the tag.

There are various way to update the application, below are the 2 ways.

Change version in domain to update application version

16. Edit the domain.yaml and update the restartVersion from applicationV1 to applicationv2.

```
  # - "NEVER" will not start any server in tl
  # - "ADMIN_ONLY" will start up only the adr
  # - "IF_NEEDED" will start all non-cluster
  replica count
  serverStartPolicy: "IF_NEEDED"
  restartVersion: "applicationV2"
```

17. Run command kubectl apply -f domain.yaml. It will do a rolling restart of admin server and manager server 1. Wait till the managed server 1 is restarted.

18. Access the app again and you will see the new changes reflected with the title new version 2.

Changing the image tag to update the application version

19. Edit the domain.yaml with the image tag

```
  # Update this with the name of the Docker image that will be used to run your domain:
  #image: iad.ocir.io/boonjiantrail/weblogic-operator-tutorial:latest
  image: iad.ocir.io/boonjiantrail/weblogic-operator-tutorial:37445819922f97b0b64662feb75f231cb09db330

  # imagePullPolicy defaults to "Always" if image version is :latest
```

20. Run command kubectl apply -f domain.yaml. It will do a rolling restart of admin server and manager server 1. Wait till the managed server 1 is restarted.

```
[root@master kube_lab]# kubectl get po -n sampledomain
NAME                              READY   STATUS        RESTARTS   AGE
sample-domain1-admin-server       1/1     Terminating   0          31m
sample-domain1-managed-server1    1/1     Running       0          30m
[root@master kube_lab]#
```

```
[root@master kube_lab]# kubectl get po -n sampledomain
NAME                              READY   STATUS        RESTARTS   AGE
sample-domain1-admin-server       1/1     Running       0          79s
sample-domain1-managed-server1    1/1     Terminating   0          33m
[root@master kube_lab]#
```

```
[root@master kube_lab]# kubectl get po -n sampledomain
NAME                                READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server         1/1     Running   0          3m20s
sample-domain1-managed-server1      1/1     Running   0          75s
```

- Access the app again and you will see the new changes reflected with the title new version 2. http:// <<public_ip_worker_node>>:30305/opdemo/

WebLogic Server on Docker - Request Information, new version 2

- getVirtualServerName(): managed-server1
- InetAddress.hostname: sample-domain1-managed-server1
- InetAddress.serverAddress: sample-domain1-managed-server1/10.244.1.56
- getLocalAddr(): 10.244.1.56
- getLocalName(): sample-domain1-managed-server1
- getLocalPort(): 8001
- getServerName(): 132.145.206.79
- WLS Server Name: managed-server1
- getIpAddOfCurrSrv():

Scaling the pod

- Edit domain.yaml and set the replicas to 2

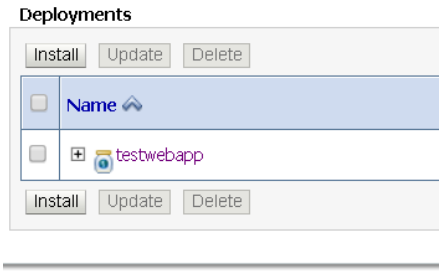
```
clusters:
- clusterName: cluster-1
  serverStartState: "RUNNING"
  replicas: 2
```

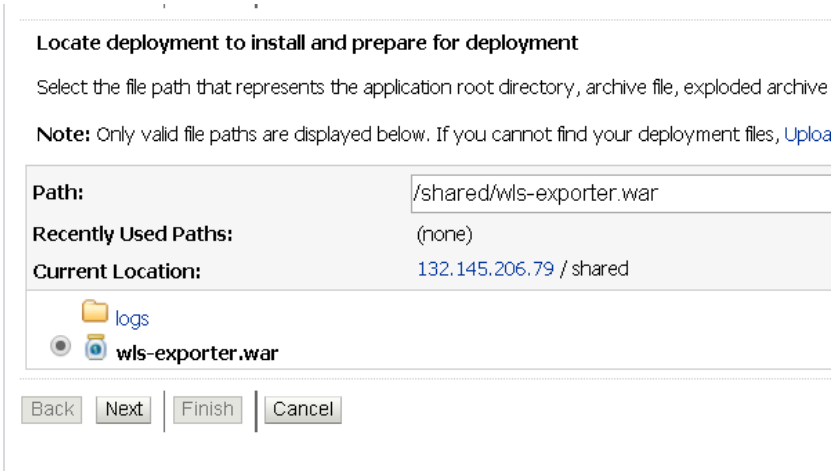
- Run command kubectl apply -f domain.yaml and kubectl get po -n sampledomain. You will see 2 pod running. Alternaitvely you can edit the domain via kubectl edit domain sample-domain1 as well.

```
[root@master kube_lab]#  kubectl apply -f domain.yaml
domain.weblogic.oracle/sample-domain1 configured
[root@master kube_lab]# kubectl get po -n sampledomain
NAME                              READY   STATUS    RESTARTS   AGE
sample-domain1-admin-server       1/1     Running   0          6m8s
sample-domain1-managed-server1    1/1     Running   0          4m3s
sample-domain1-managed-server2    0/1     Running   0          5s
```

Install Weblogic Metrics for promethus and Grafana .

- Run command
    - cp /home/opc/kube_lab/wls-exporter.war /shared
- Login into weblogic console and deployed wls-exporter.war to both admin server and managed server. You will know how to do this after the previous lesson from weblogic labs without much instructions.





Back Next Finish Cancel

Choose installation type and scope

Select if the deployment should be installed as an application or library. Also

The application and its components will be targeted to the same locations. Tl

Install this deployment as an application

Application libraries are deployments that are available for other deployments

Install this deployment as a library

Select a scope in which you want to install the deployment.

Scope:

Back Next Finish Cancel

Back Next Finish Cancel

Select deployment targets

Select the servers and/or clusters to which you want to deploy this ap

Available targets for wls-exporter :

Servers
☑ admin-server

Clusters
☑ cluster-1 ○ All servers in the cluster

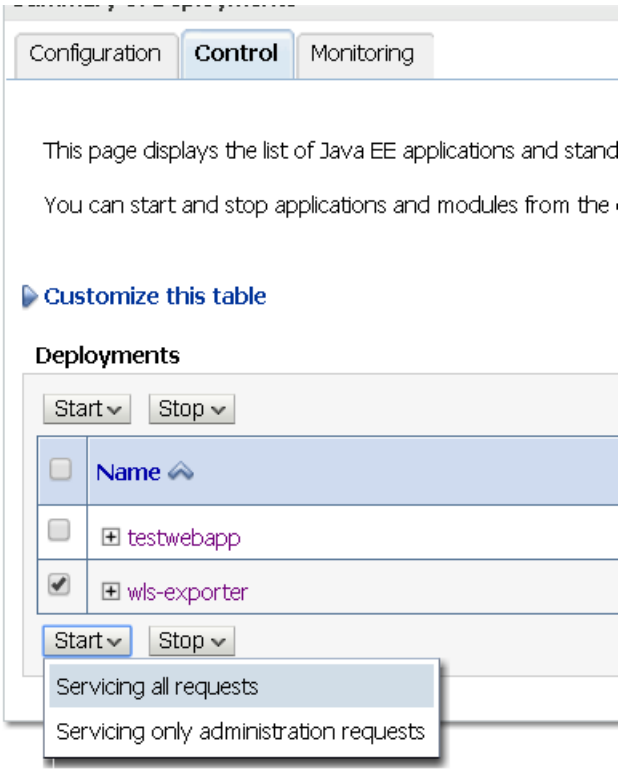
Back Next Finish Cancel

During deployment, the files will be copied automatically to the Managed Servers to which the application is ta

I will make the deployment accessible from the following location

Location: /shared/wls-exporter.war

Provide the location from where all targets will access this application's files. This is often a shared directory. Yo



Access Metrics

- Run kubectl get po -o wide -n sampledomain to get the po ip.
- Run below command to get if metrics is running fine.
- curl http://weblogic:welcome1@10.244.1.56:8001/wls-exporter/metrics | grep -na "cpu"

```
[root@master kube_lab]# kubectl get po -o wide -n sampledomain
NAME                              READY   STATUS    RESTARTS   AGE     IP            NODE     NOMINATED NODE
sample-domain1-admin-server       1/1     Running   0          14m     10.244.1.55   worker   <none>
sample-domain1-managed-server1    1/1     Running   0          12m     10.244.1.56   worker   <none>
sample-domain1-managed-server2    1/1     Running   0          8m49s   10.244.1.57   worker   <none>
[root@master kube_lab]# curl http://weblogic:welcome1@10.244.1.56:8001/wls-exporter/metrics | grep -na "cpu"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0282:wls_jvm_process_cpu_load{name="managed-server1"} 0.024476140
100 53033    0 53033    0     0   156k      0 --:--:-- --:--:-- --:--:--  156k
428:wls_scrape_cpu_seconds{instance="10.244.1.56:8001"} 0.43
```

Install Grafana and Prometheus

```
./16_prograpinstall.sh
```

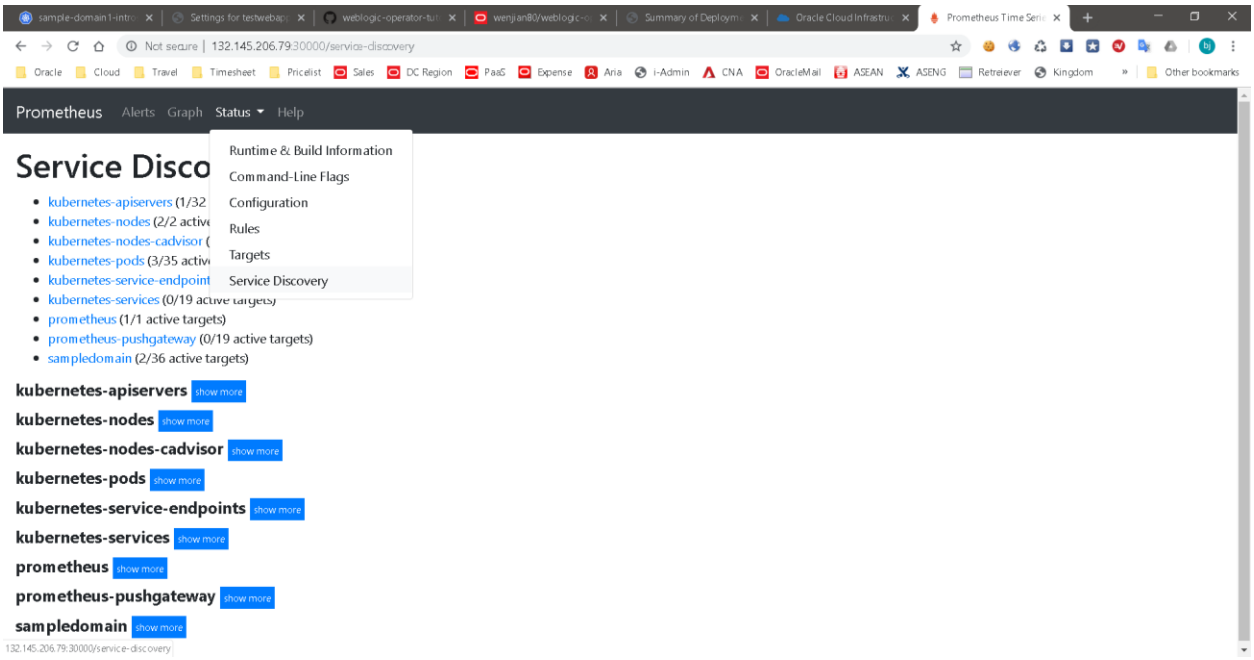
Ouput

```
NAME                          TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)       AGE     SELECTOR
grafana                       NodePort   10.96.146.238  <none>        80:31001/TCP  5m14s   app=grafana,release=grafana
prometheus-kube-state-metrics ClusterIP  None           <none>        80/TCP        5m18s   app=prometheus,component=kube-state-metrics,release=prometheus
prometheus-node-exporter      ClusterIP  None           <none>        9100/TCP      5m18s   app=prometheus,component=node-exporter,release=prometheus
prometheus-server             NodePort   10.106.194.90  <none>        80:30000/TCP  5m18s   app=prometheus,component=server,release=prometheus
```

Login to Prometheus

http://132.145.206.79:30000/graph (worker node ip)

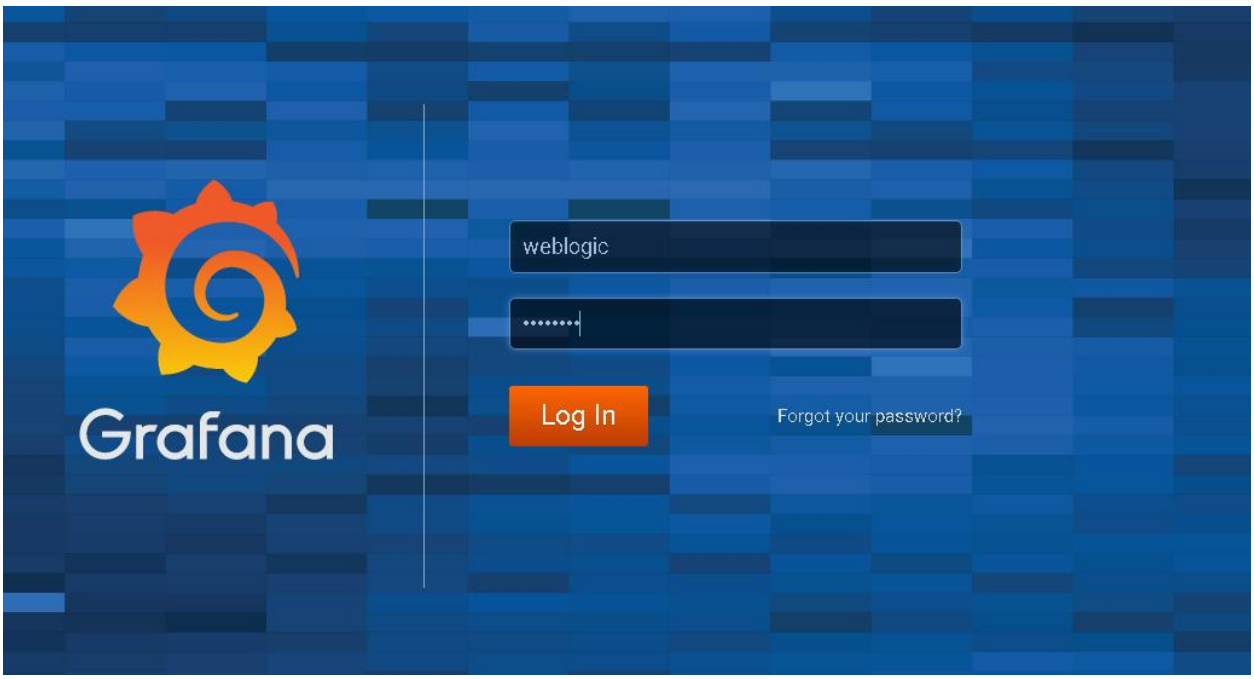
Go to status->Service disvoery. You will see 2 active target



Click on graph and select wls_jvm_process_cpu_load

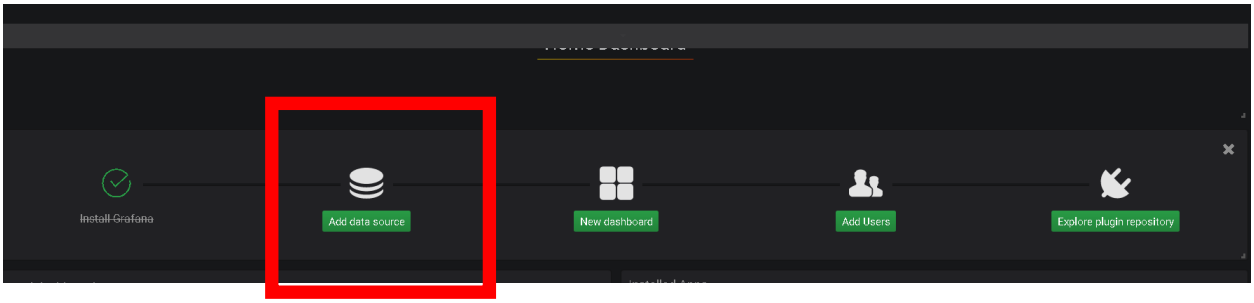
Grafana Login

http://132.145.206.79:31001/login [Access your worker public ip]

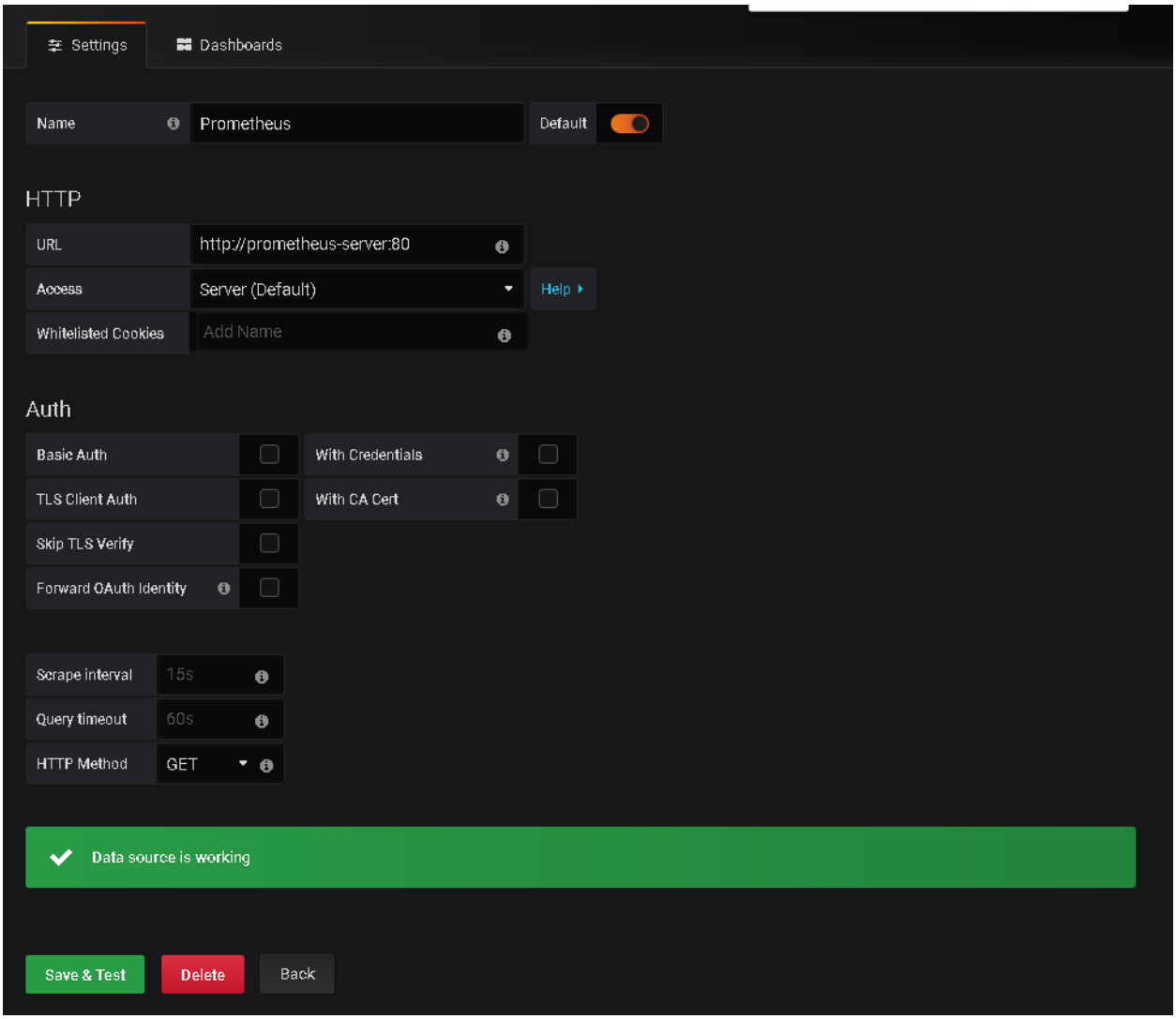


Enter weblogic/welcome1 as user name and password

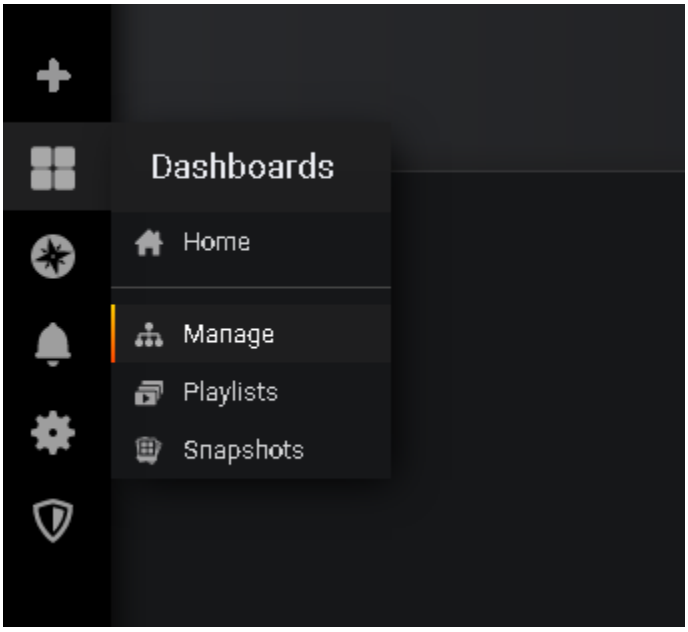
Click an add DataSource



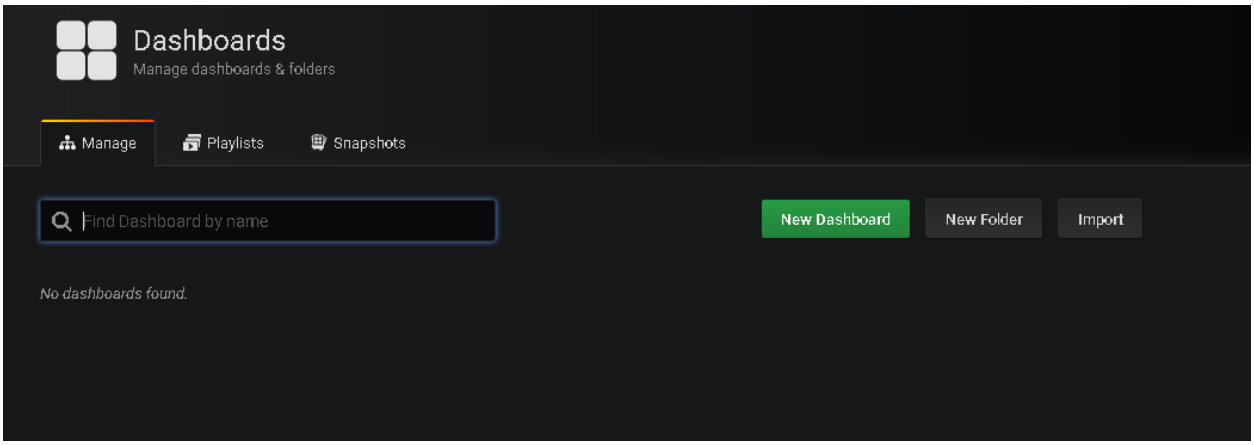
Choose Prometheus enter http://prometheus-server:80 then click save and test

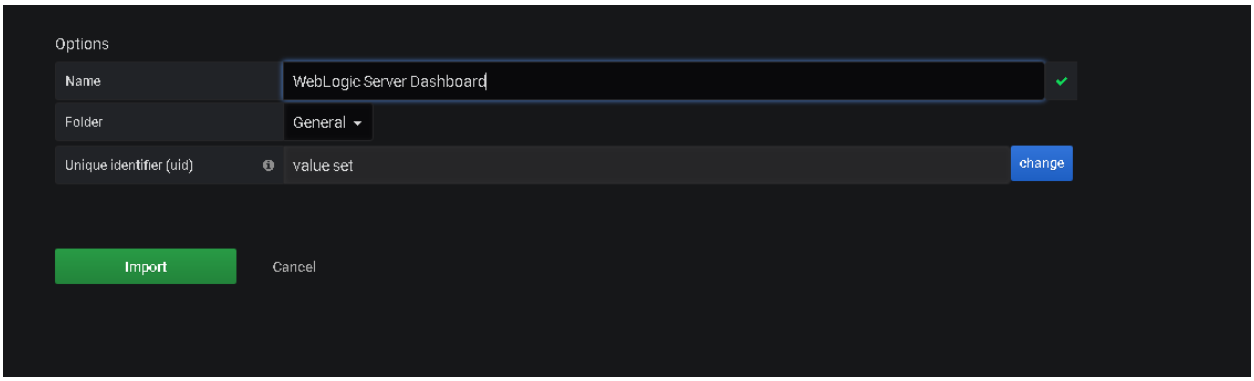


Select managed dashboard.



Click on import to import dashboard json choose weblogic_dashboard.json




Options
Name
WebLogic Server Dashboard
Folder
General
Unique identifier (uid)
value set
change
Import
Cancel


Servers
Running Servers
2
Deployed Applications
7
Server Name
managed-server1
Server Status
RUNNING
Heap Usage
75.0%
Running Time
29.3 min
Open Sockets
12
Server Name
managed-server2
Server Status
RUNNING
Heap Usage
60.0%
Running Time
25.3 min
Open Sockets
13
JVM Heap
250 MB
200 MB
150 MB
100 MB
50 MB
0 B
14:16
14:18
14:20
14:22
14:24
14:26
14:28
14:30
14:32
14:34
14:36
14:38
14:40
14:42
14:44
max
avg
current