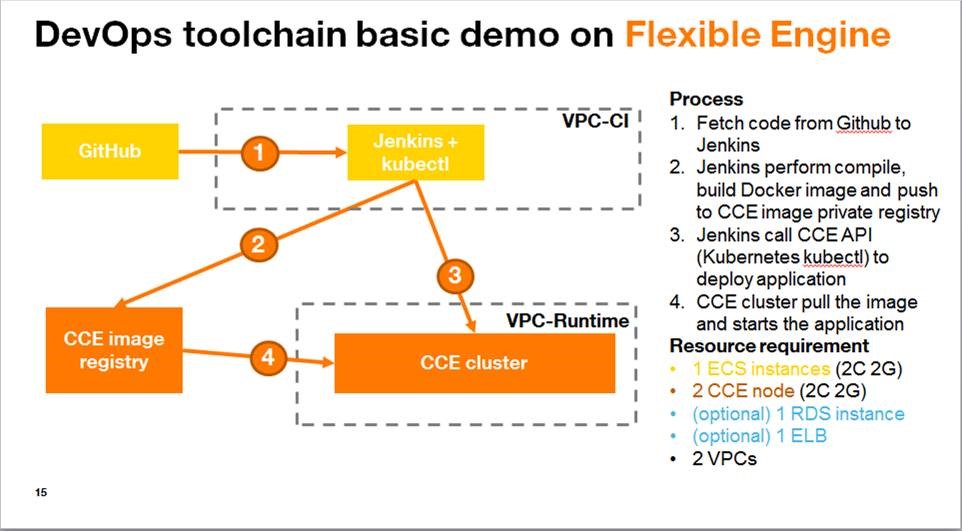
# Abstract

This is a guide on how to integrate CCE with Jenkins to set up a CI/CD Demo environment on FlexibleEngine. And Target of this documentation should have basic understanding of FlexibleEngine especially the services related to the Demo: VPC, ECS, CCE and how to manage these services such as creating a VPC/Subnet, creating an ECS, logging in to an ECS with SSH, creating an CCE cluster etc. So in this documentation it will not describe the details how to do these basic actions on FE and instead just use one short description telling what needs to be done.

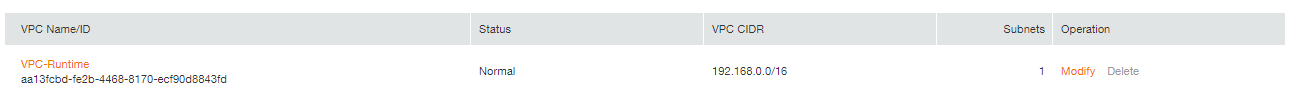
# Architecture

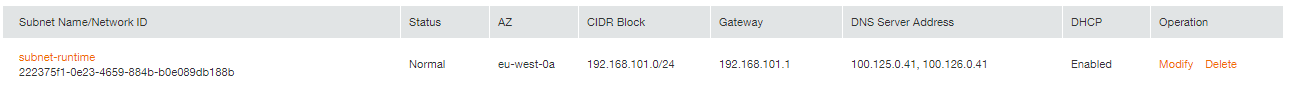


# Prerequisite

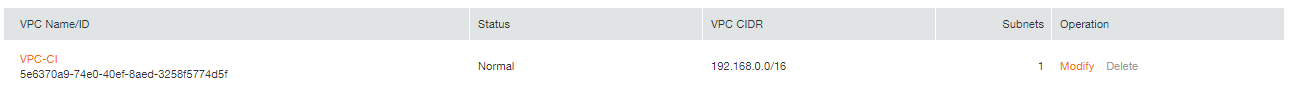
1. A Tenant in FlexibleEngine and a User in this Tenant with Console and API access privilege to manage resources (VPC/ ECS/CCE).
2. Two VPCs are created, one is VPC-CI, another is VPC-Runtime, and in each VPC a subnet is created.

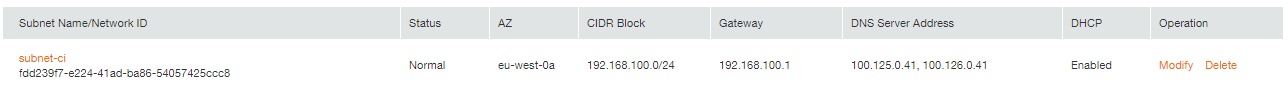
VPC-Runtime and subnet-runtime in this VPC



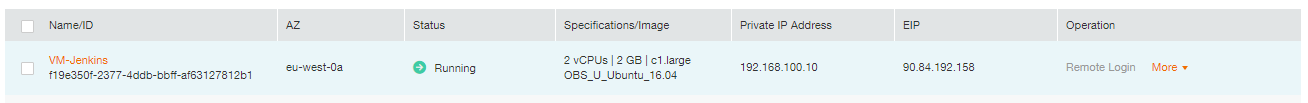


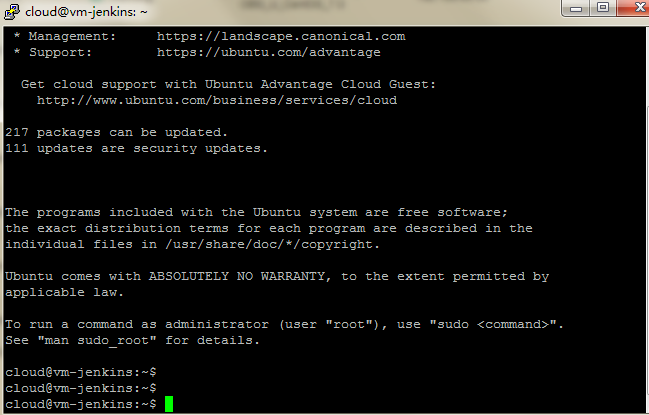
VPC-CI and subnet-ci in this VPC





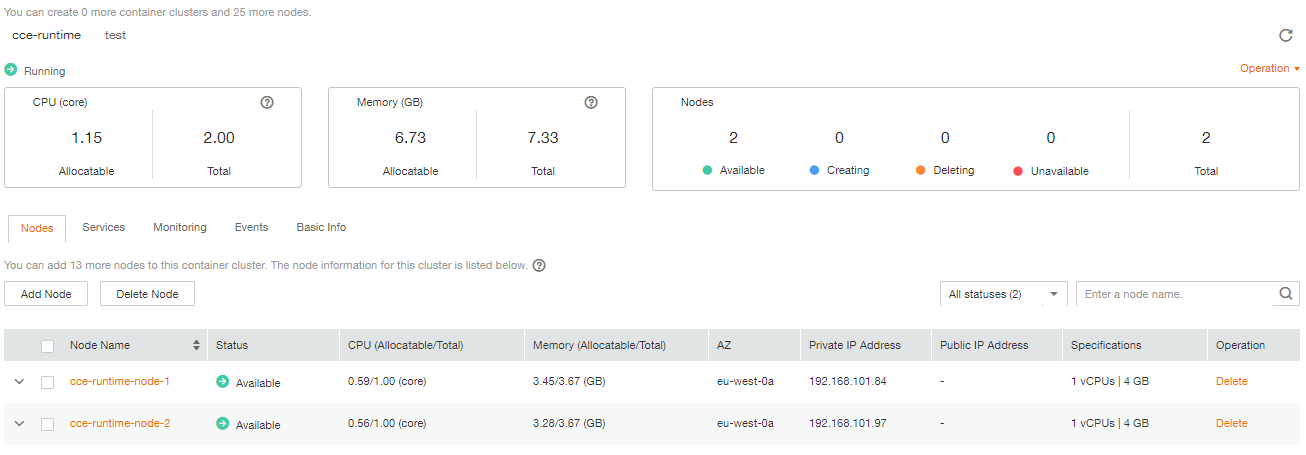
1. An ECS VM (We name it as VM-Jenkins) with OS Ubuntu 16.04 x86-64 is created and attached to subnet-ci and it can be [connected from SSH client](https://docs.prod-cloud-ocb.orange-business.com/en-us/usermanual/ecs/en-us_topic_0017955380.html)





*Note: for all the following operations in ECS vm-jenkins, using a non-root user (for example: cloud) is highly recommended.*

1. A CCE cluster (We name it as cce-runtime) with 2 nodes is created, and the cluster is attached within VPC-Runtime and subnet-runtime. Also in this case, we will attach EIP for each node for the external testing of accessing services that the App opens.



Otherwise, the certificate file is need to be uploaded in order to enable using private container image we are going to build (please follow the guide: <https://docs.prod-cloud-ocb.orange-business.com/en-us/doc/pdf/20180530/20180530115106_9d1333530a.pdf>: 2.2 Uploading an AK/SK File)

# Step 1: Installation

## Set up Github Repository

### Login to [www.github.com](http://www.github.com), (if you don’t have one account, please register one and login)

### [Fork](https://help.github.com/articles/fork-a-repo/) a repository from this repository (<https://github.com/Karajan-project/hellonode>) and [set it as Public repository](https://help.github.com/articles/making-a-public-repository-private/), then you will get your own repository ([https://github.com/<your register account>/hellonode](https://github.com/%3cyour%20register%20account%3e/hellonode)) for this demo. (The reason why you need to have your own repository is because you will need to modify the source code to trigger the build project in Jenkins.)

## Set up Jenkins and Required Plugins

### Login to ECS VM-Jenkins with SSH

### Install Jenkins

*Note: JDK is mandatory to run Jenkins, so before installing Jenkins, please install JDK at first.*

Install JDK:

sudo add-apt-repository ppa:openjdk-r/ppa

sudo apt-get update

sudo apt-get install openjdk-8-jdk

Install Jenkins:

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

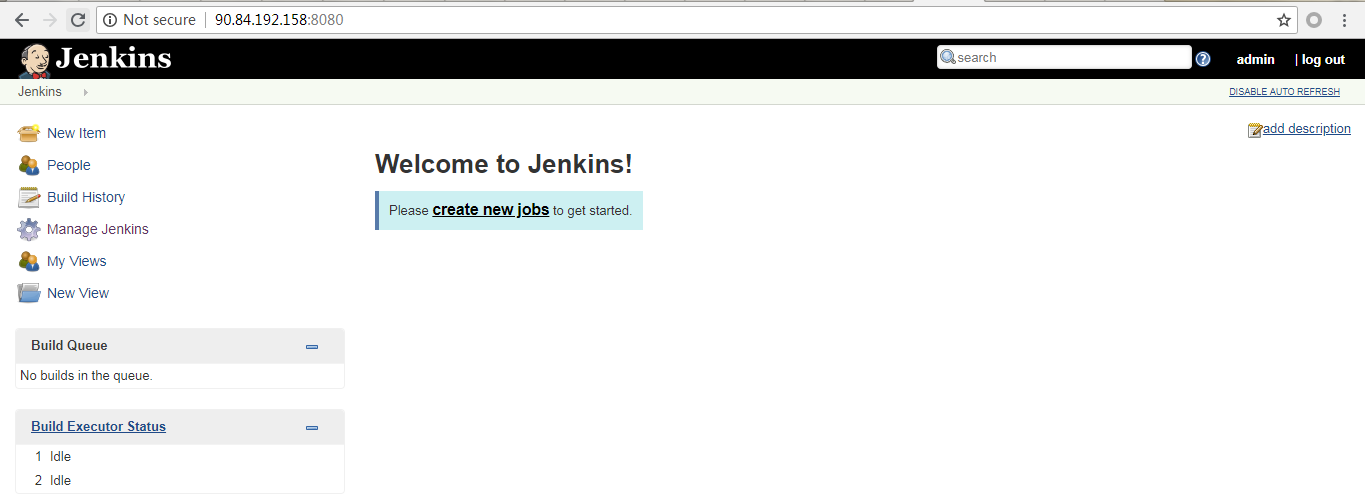
sudo apt-get update

sudo apt-get install Jenkins

*Note: the port Jenkins listens on needs to be added in the security group Inbound* *rule. If you didn’t change the port that Jenkins listens, it will be 8080 default.*



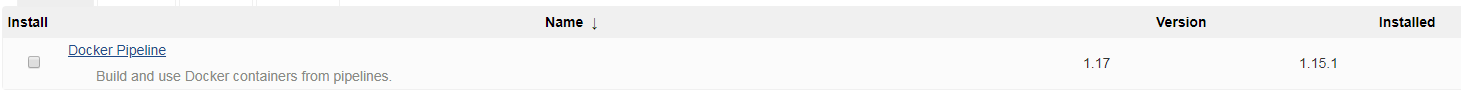
*Check Point: enter << http://<External IP Address> :< Port> >> in browser to see if Jenkins is installed successfully and can be accessed normally.* When logging in successfully, follow the wizard to initialize Jenkins, then you will go to this page.



### Go to Jenkins “go to plugin manager” and install the following plugins:

Refer: <https://jenkins.io/doc/book/managing/plugins>

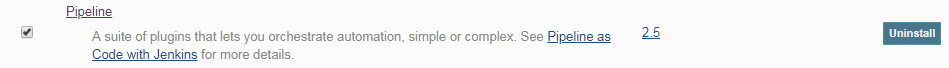
Docker Pipeline



Kubernetes Cli



Pipeline



Github



## Setup docker

### Login to ECS VM-Jenkins with SSH and execute the following CMD:

sudo apt-get remove docker docker-engine docker.io

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

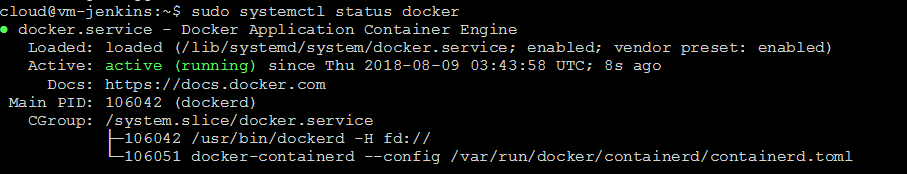
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt-get update

sudo apt-cache policy docker-ce

sudo apt-get install -y docker-ce

*Check Point: execute cmd “sudo systemctl status docker” to check if docker is installed successfully with the following similar output:*



# Step 2: Configure and Create Jenkins Project

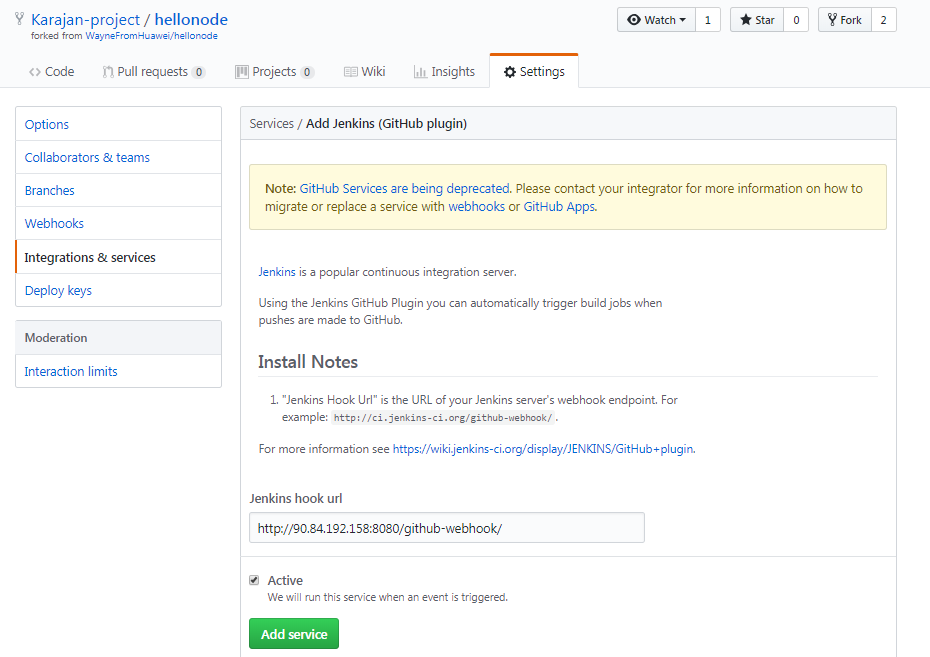
## Configure Github repository

### Automatic building: To make sure that the project in Jenkins can be triggered automatically when there is new commit/push to the source code repository.

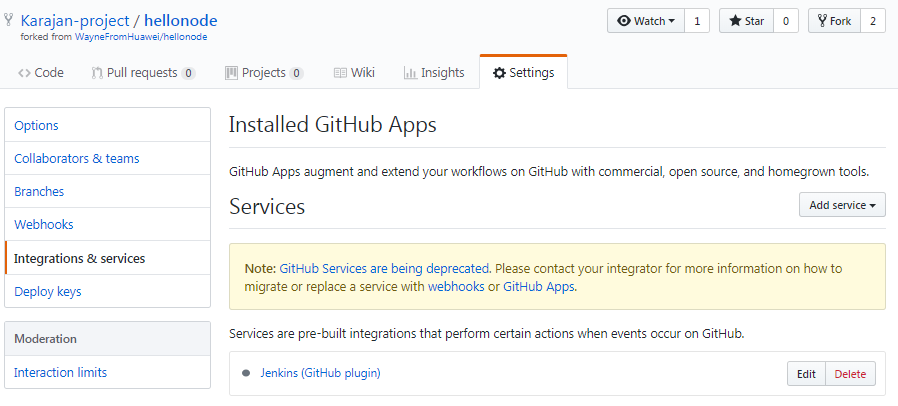
Go to settings -> Integration & Services under the new forked project and install the “Jenkins Github plugin”, and the Jenkins hook url should be:

http://<Jenkins access IP> :< Jenkins access Port>/github-webhook/

As an example:



Final Status after completion:



## Configure Jenkins

### Add user “Jenkins” to group “docker” to make sure jenkins can invoke docker:

sudo usermod -a -G docker Jenkins

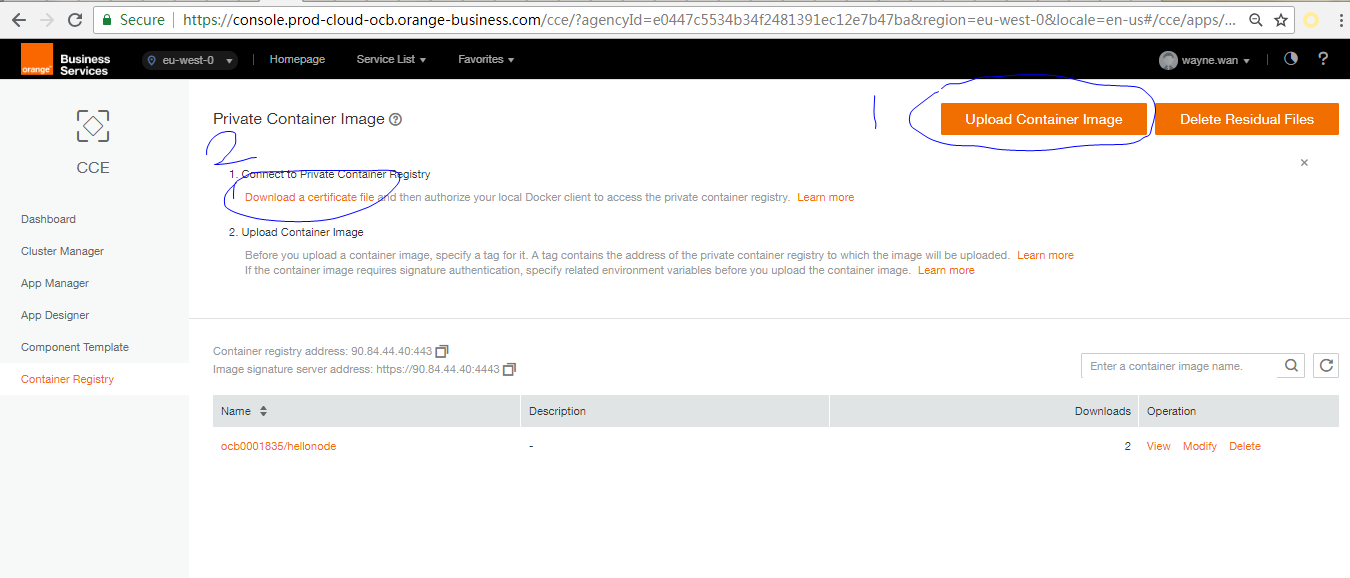
Then restart Jenkins

sudo service jenkins restart

### Create CCE private registry credentials

1. Download a certificate file from Container Registry

With file name “dockercfg.txt”



1. Get auth token

Execute the command:

echo -n {auth}| base64 –d in vm-jenkins,

in this command, the {auth} must be replaced by the value of auth parameter in the dockercfg.txt file. Example command:

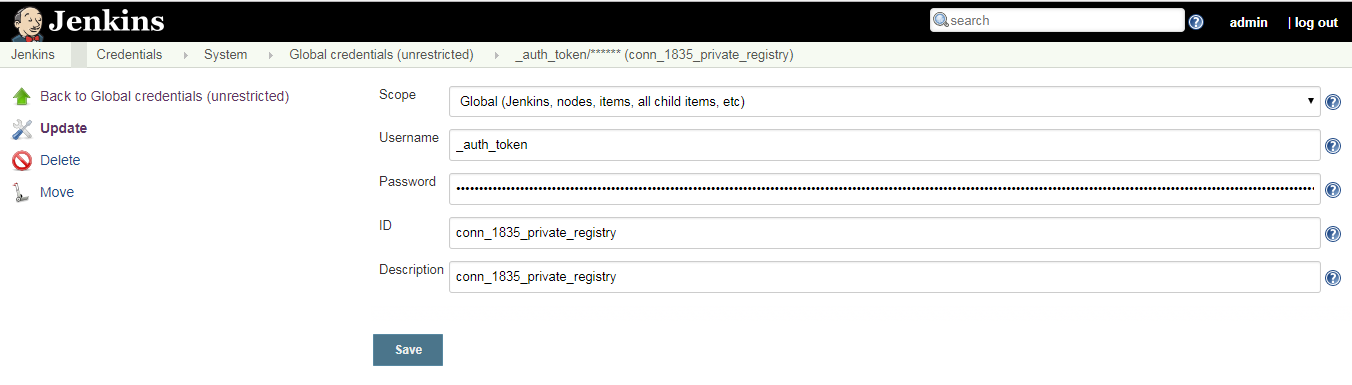
echo -n X2F1dGhfdG9rZW46YTljYWI4YmNiZWJjNGNmMDhjZjkwODI1ODQxYzBhZWItVUdGS1Y4VVlVR 09KSUZRVEw0VUwtMjAxNjA2MTcxODAzNTgtZTc1ZmJiNmFlNTIwYjA3ZTA4ZjY5OThiOGEyZG FiNTJiYjgyNWI4YjRhNDQ4YzMwNjRmNDBiZGI5OWE3NDQxMA==| base64 -d

Example command output:

\_auth\_token:1c1ea38bc3dd4910879170b154ac5eedMO3WATY4WASMHTWDTOV6-20180824102142- c56d5ccc9dcdd8da9345ddfa02b7354425e6e1c0e2a2f7bcbace0ff0c1f20966

While 1c1ea38bc3dd4910879170b154ac5eedMO3WATY4WASMHTWDTOV6-20180824102142- c56d5ccc9dcdd8da9345ddfa02b7354425e6e1c0e2a2f7bcbace0ff0c1f20966 is the token we get and \_auth\_token is the user name.

1. Create credential with the user name and token



### Create Kubernetes CLI Credential

1. Install kubectl in vm-jenkins:

Download kubectl

curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.7.3/bin/linux/amd64/kubectl

chmod +x kubectl

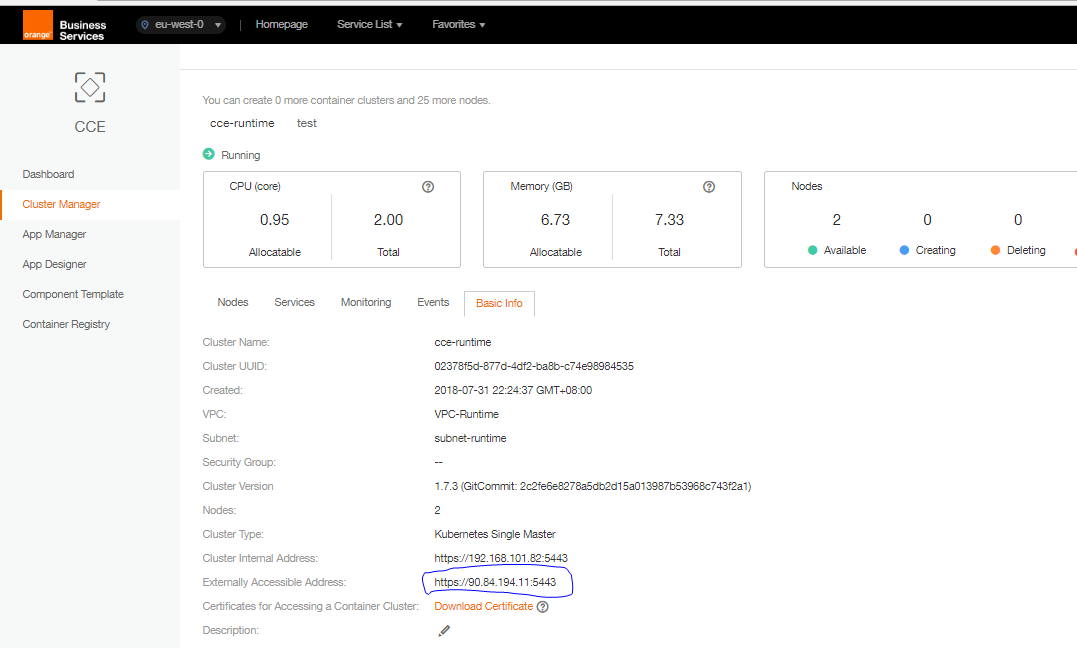
Add kubectl to the PATH env

sudo cp kubectl /usr/local/bin

Append the following line in /etc/hosts:

90.84.194.11 kubernetes.default.svc.cluster.local

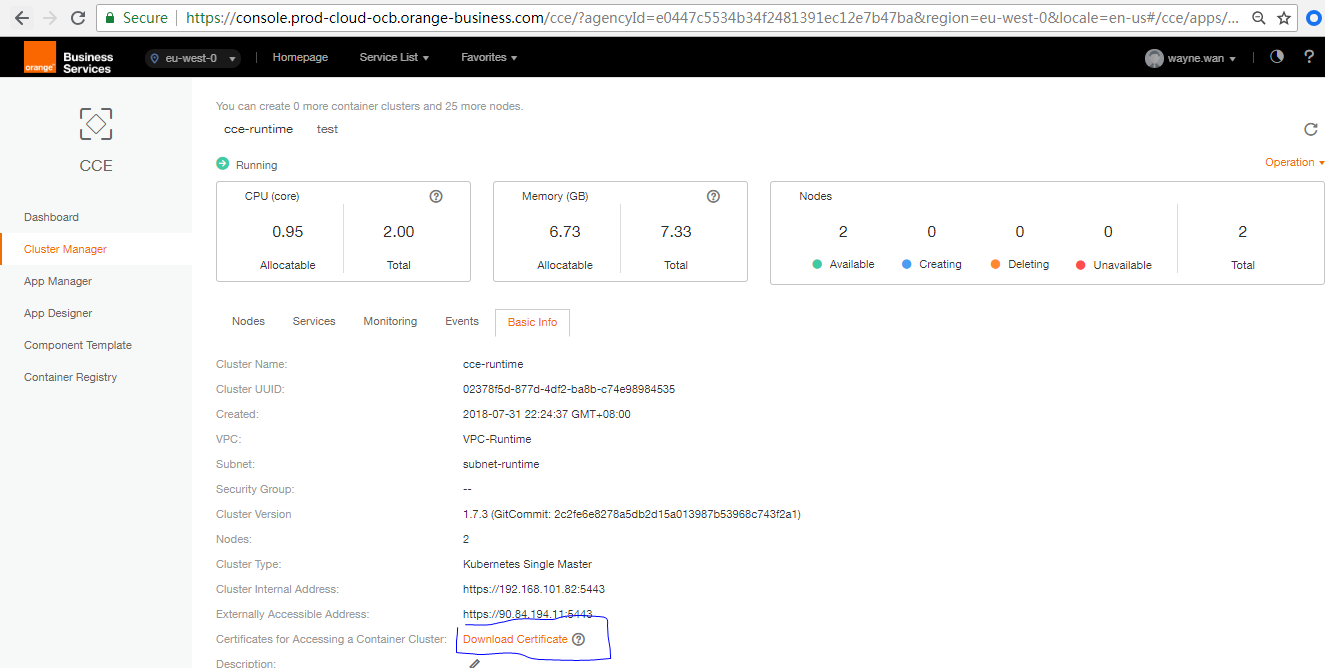
*Note: 90.84.194.11 is the IP Address of the CCE cluster from here illustrated below, you need to replace the IP Address from your own CCE cluster.*



1. setup and connect to the CCE cluster

Download certificate files from CCE cluster console, and you will get three files:

cacrt, clientcrt, clientkey



Upload the above three files to vm-jenkins and execute the following CMDs:

kubectl config set-cluster default-cluster --server=https://kubernetes.default.svc.cluster.local:5443 --certificate-authority=cacrt

kubectl config set-credentials default-admin --certificate-authority=cacrt --client-key=clientkey --client-certificate=clientcrt

kubectl config set-context default-context --cluster=default-cluster --user=default-admin

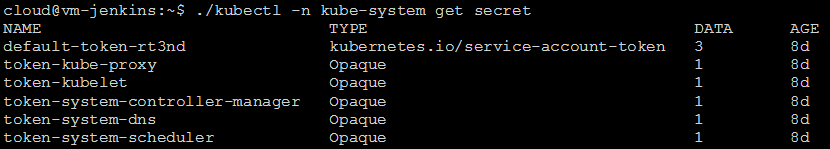
kubectl config set current-context default-context

1. get token content

execute:

kubectl -n kube-system get secret

To get the default token with name “default-token-\*\*\*\*\*”, for example: default-token-

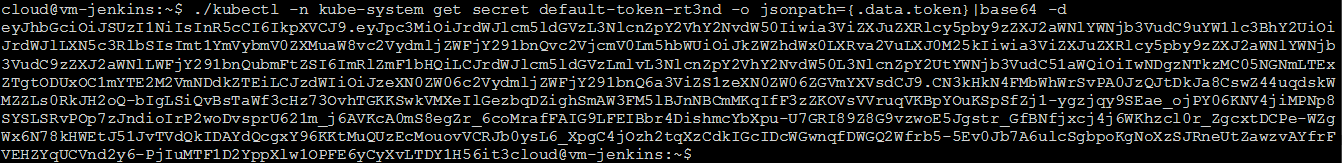
Rt3nd 

Then execute

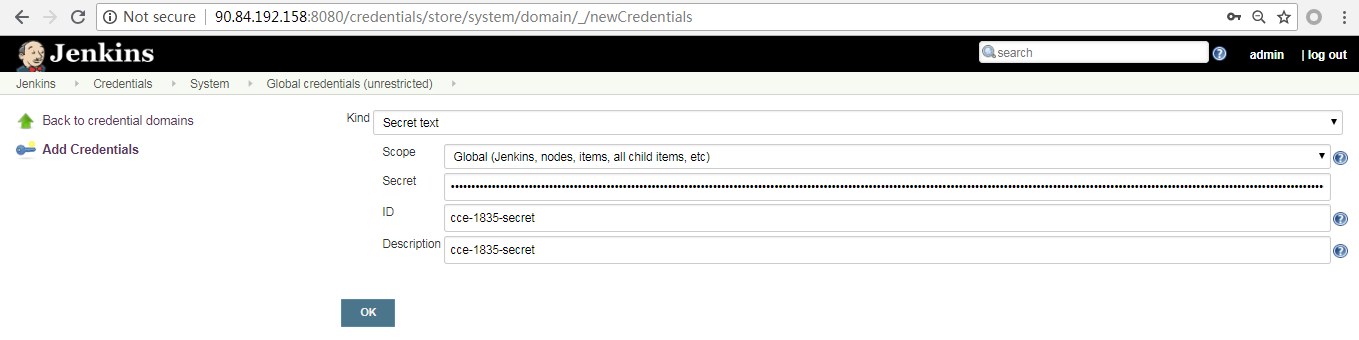
kubectl -n kube-system get secret default-token-rt3nd -o jsonpath={.data.token}|base64 –d

to get the token content:

*Note: replace default-token-rt3nd with the real token name in your environment*

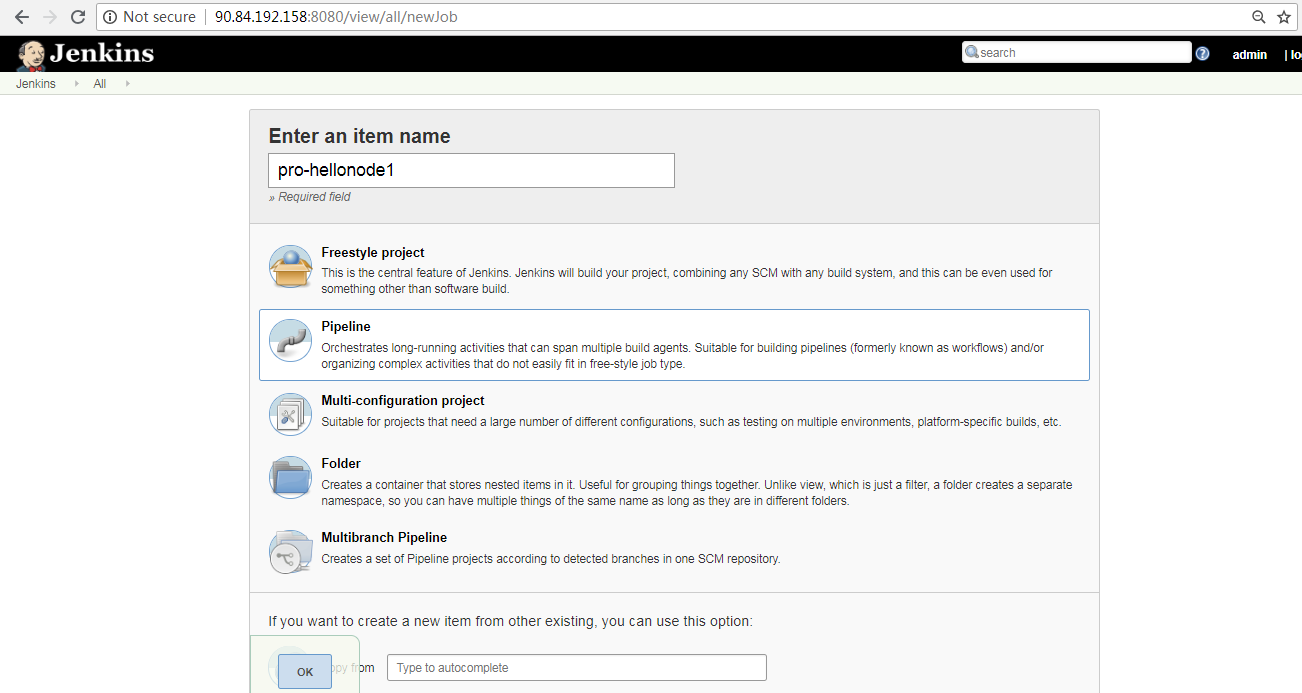


1. create credential in Jenkins with the token:

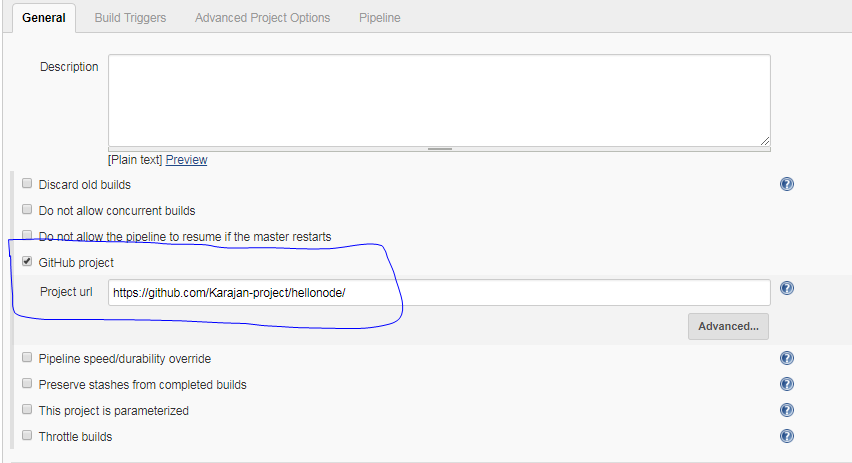


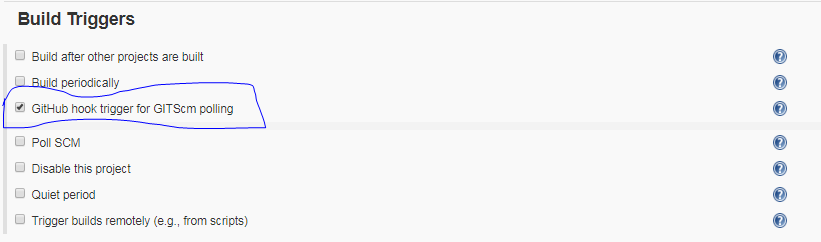
### Jenkins **pipeline** (we name it as pro-hellonode)

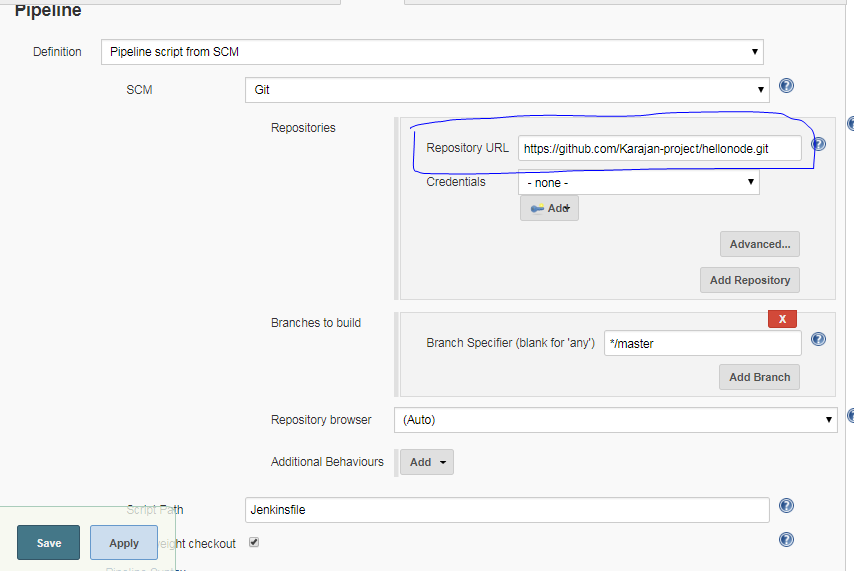
1. Create a pipeline project in jenkins



Refer to the following parameters which are needed to be specified







*Note: Replace the Github project and repository address to the Github repository you just forked.*

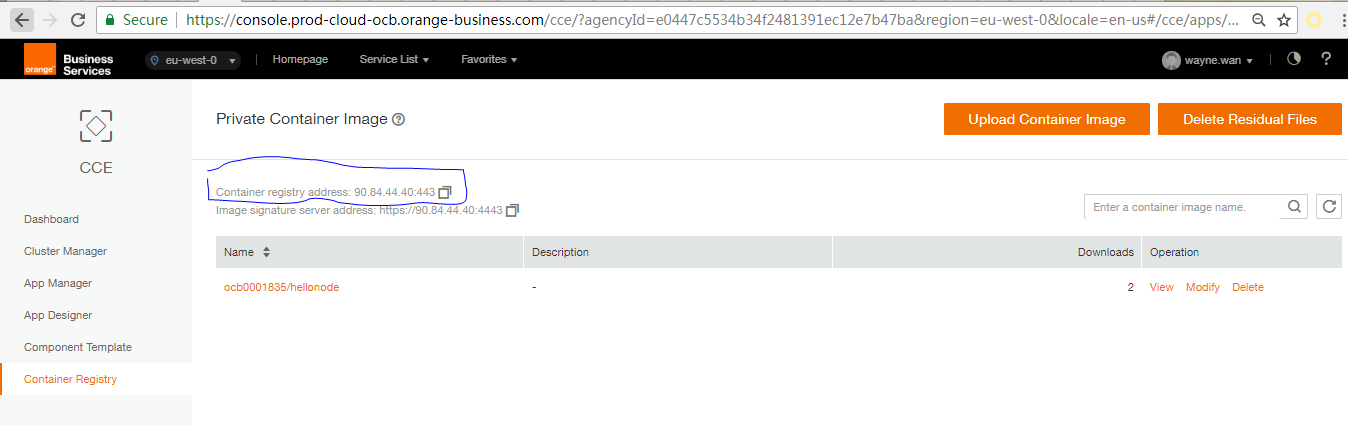
## Configure Docker Client

Add the container registry address (refer to CCE Container Registry console image following) insecure-registries parameter in the /etc/docker/daemon.json file.

{

"insecure-registries": ["{container\_registry\_address}"]

}



Run the following commands to restart Docker:

sudo systemctl daemon-reload

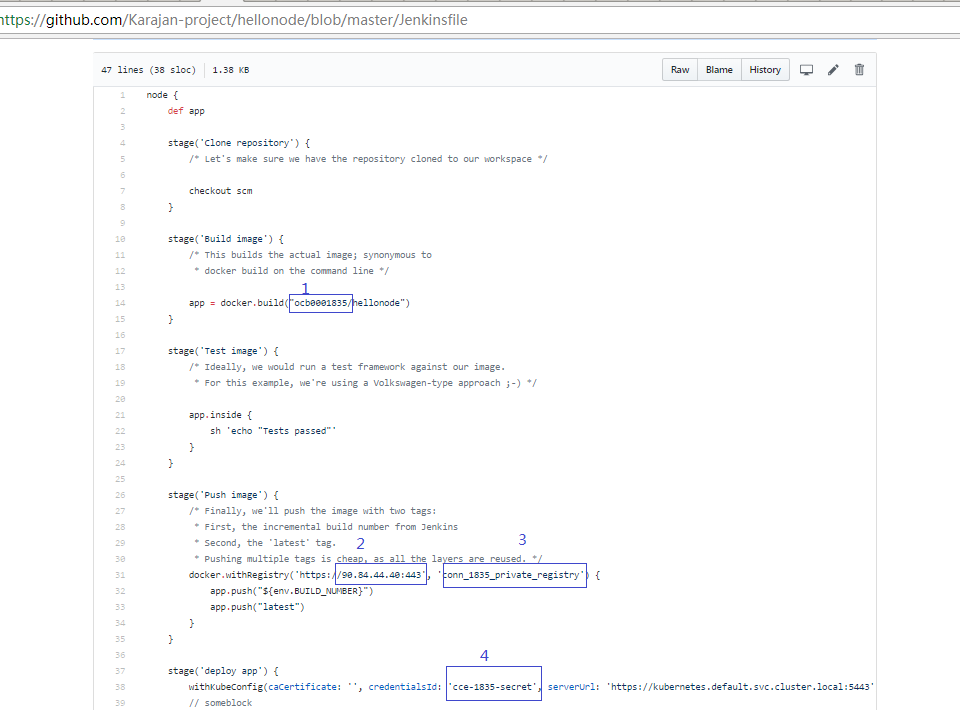
sudo service docker restart

## Modify github repository resource code to adapter your environment

### Jenkinsfile

Go to your github repository, and edit directly the “Jenkinsfile” file on the web portal or you can edit at your local desktop and then push/commit

For the following 4 places:



Number 1 is the tenant id

Number 2 is the CCE container registry address (Refer to Step 2 -> segment 3 Configure Docker Client)

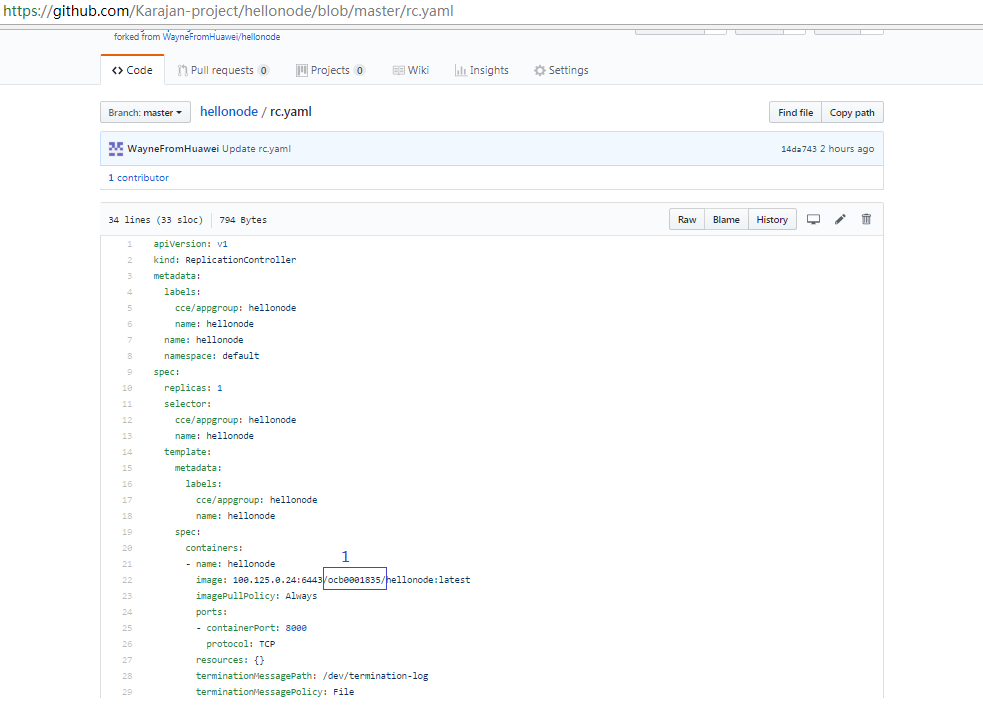
Number 3 is the credential ID for connecting CCE container registry address (refer to Step 2 -> Segment 2 Configure Jenkins -> Create CCE private registry credentials)

Number 4 is the credential ID for connecting CCE cluster (refer to Step 2 -> Segment 2 Configure Jenkins -> Create Kubernetes CLI Credential)

Then commit changes

### rc.yaml

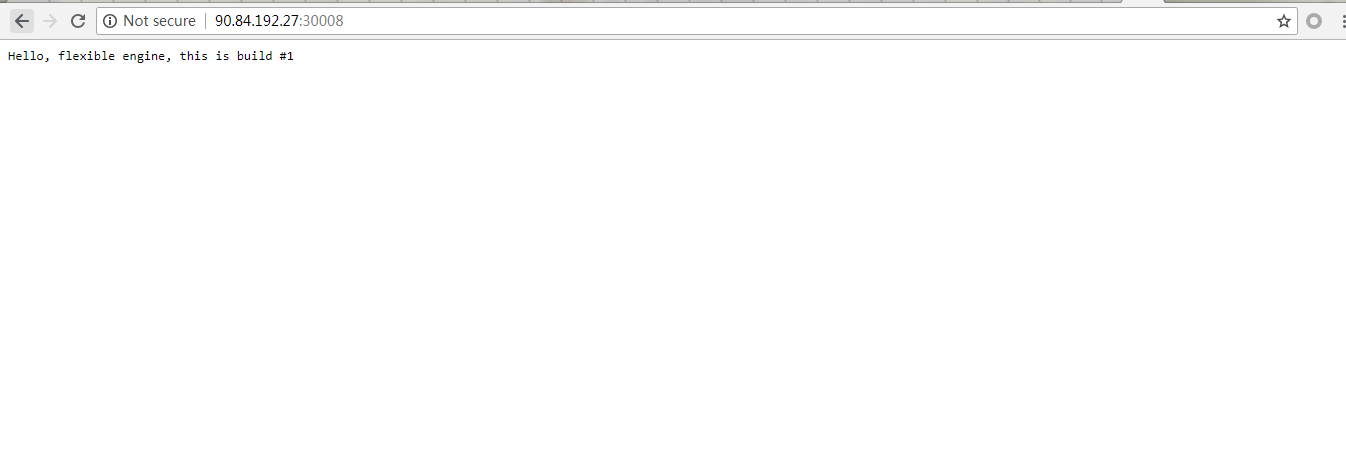
Go to your github repository, and edit directly the “Jenkinsfile” file on the web portal or you can edit at your local desktop and then push/commit



Number 1 is the tenantID needs to be modified.

# Test

1. Go to the source code repository main page and modify the file main.js from “Hello, flexible engine, this is build #1” to “Hello, flexible engine, this is build #2”.
2. Check the pro-hellonode project in jenkins and it should be running automatically.
3. Check App Manager of CCE and the application should be deployed automatically after the pro-hellonode project completes successfully.
4. Go to url http://<containter IP>:<containter Port> and the index page of the application should be displayed like this:



## 