Docker swarm mode in Google cloud Ubuntu VMs

Table of Contents

1 Creation of Ubuntu VM in Google Cloud Console

1.1 Creation of VM

1. Installation of Docker in Ubuntu VM

2.1 Installation of Docker

3 Introduction to Docker Swarm Mode

3.1 What is Swarm Mode?

4 Setting up Docker Swarm mode

4.1 Docker swarm mode

5 Docker compose file in Docker Swarm

5.1 Execution of docker-compose.yml

6 Nodes in Grid Console

6.1 To check nodes in grid console

7 Execution of Selenium Grid code

7.1 Code Execution

8 Execution of Selenium Grid code using Docker file.

8.1 Code Execution using Docker file

9 Execution of Selenium Grid code using Docker file from Jenkins.

9.1 Code Execution From Jenkins.

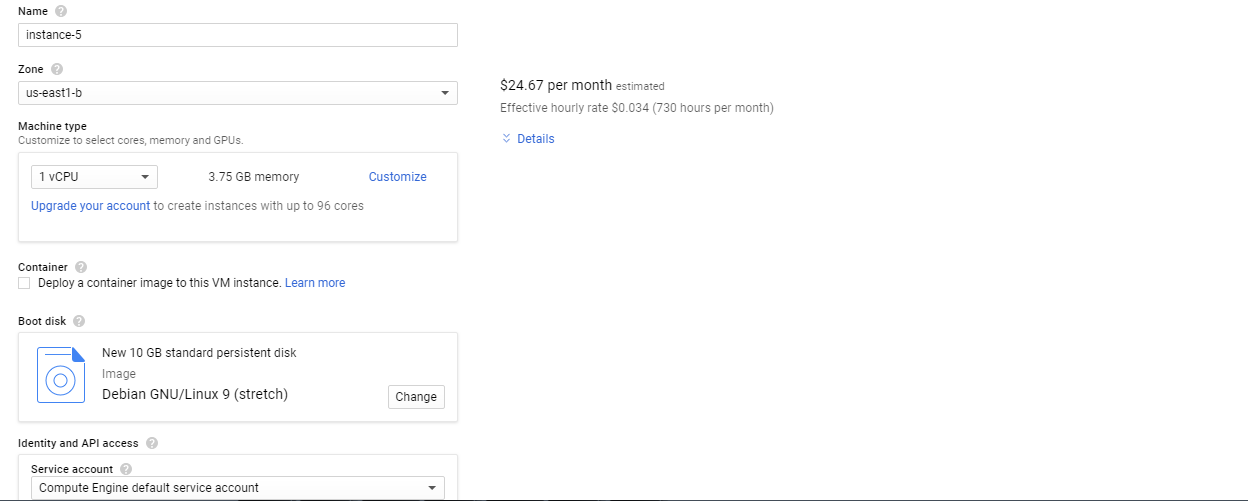
10 Push Docker Image to Docker hub from Jenkins.

10.1 Push image to hub.

1. Creation of Ubuntu VM in Google Cloud Console

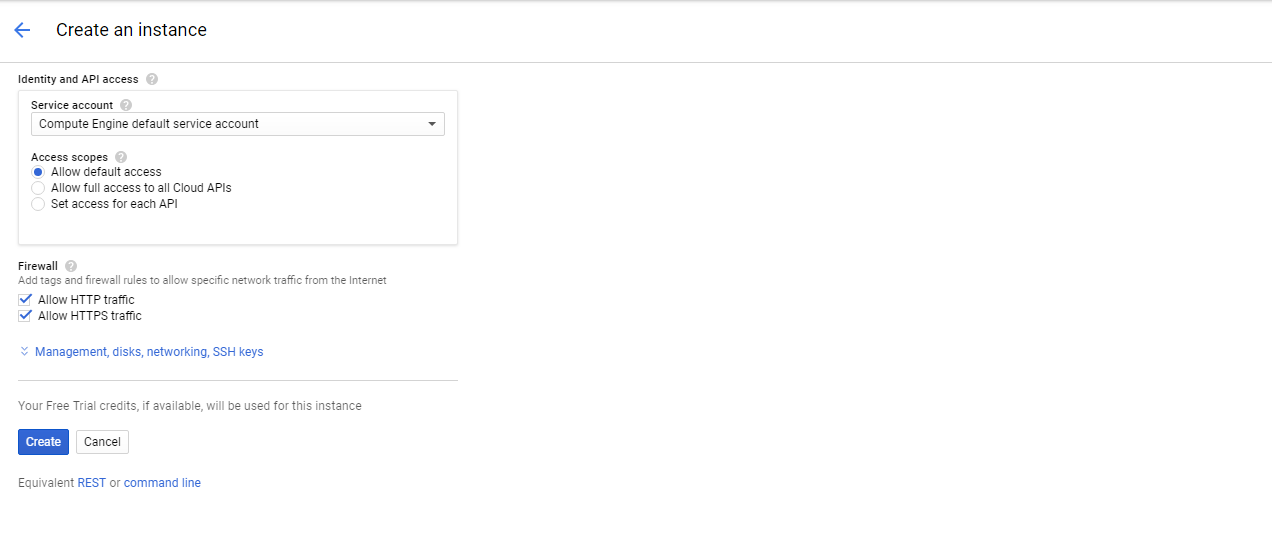
1.1 Creation of VM

1. Open <https://console.cloud.google.com>
2. Navigate to Compute Engine
3. Navigate to VM instances
4. Click on create instance
5. Now you are at

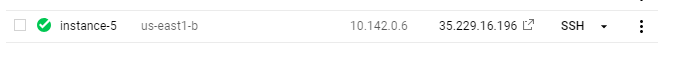


In above window

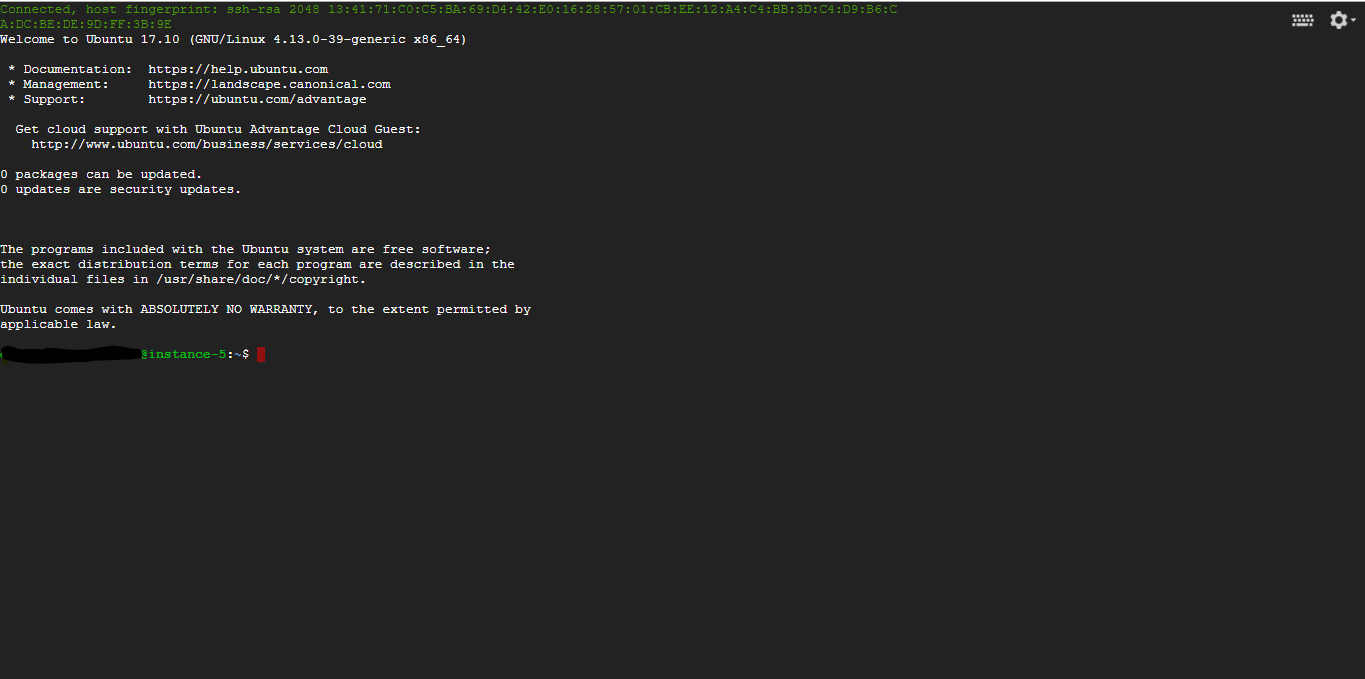
* Click on Change
* Select the OS you want to work on.
* Click on Select
* Now you are at



* Check both the checkbox under Access scopes.
* Then click on Create.
* After few seconds your VM is on



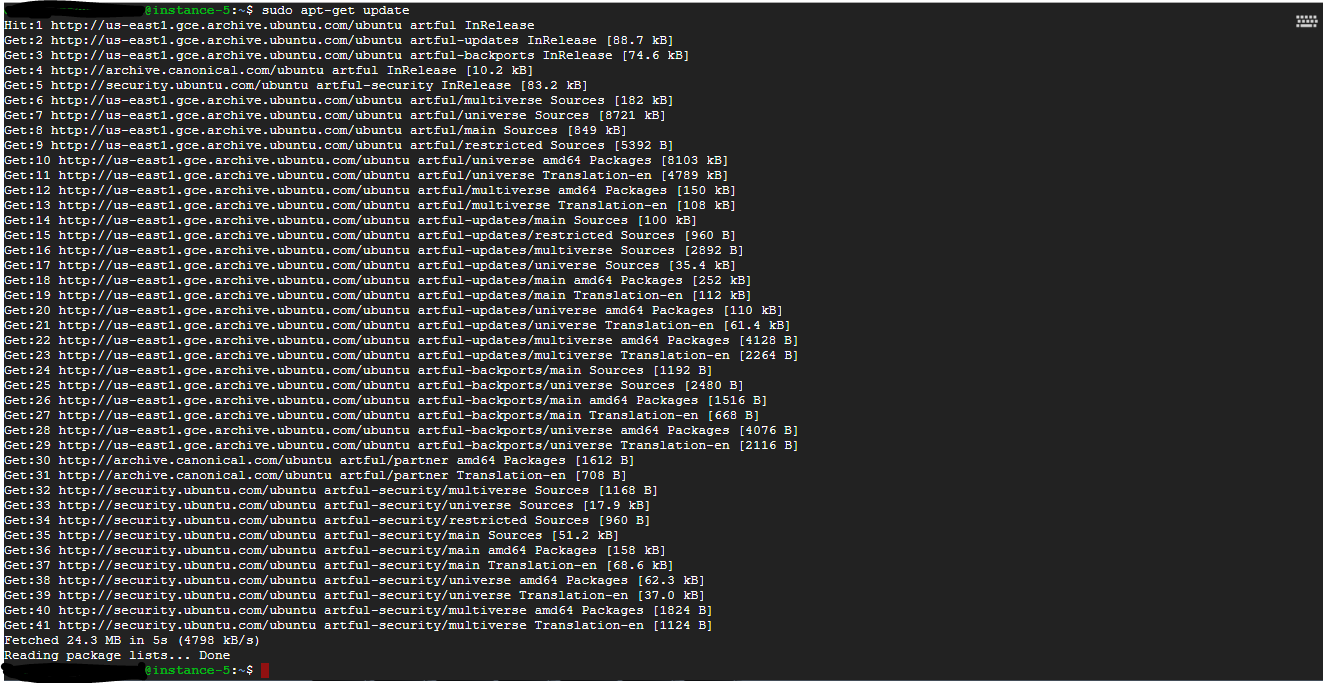
* Now click on the down arrow beside SSH
* Click on Open in Browser Window. (VM might not open if Pop-Ups blocked)
* Click on continue blocking Pop-Up
* After that again click on the down arrow beside SSH
* Now your VM is open. It will take few seconds to get into running mode.



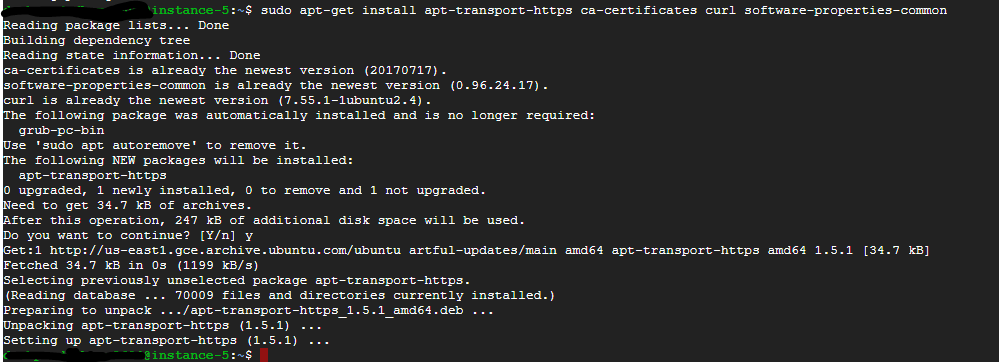
2. Installation of Docker in Ubuntu VM

* 1. Installation of Docker
* To install Docker in Ubuntu VM execute following commands.

1. Execute sudo apt-get update



1. Execute sudo apt-get install apt-transport-https ca-certificates curl software-properties-common



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

C:\Users\kishan.bhavsar\Desktop\Doc\curl.PNG

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

-After executing above command you might get below error.

C:\Users\kishan.bhavsar\Desktop\Doc\error.PNG

-To resolve this error

- Execute sudo nano /etc/apt/sources.list

- Then go inside the sources.list, find the line which is mention in the error and comment it.

- Then press ctrl+o then Enter and then ctrl+x

-Now try to execute sudo apt-get update.

1. Execute sudo apt-get install docker.io (After executing this code, Docker installed in VM.)

-Execute docke info or docker version commands and check the results.

- After executing commands like docker info or docker version you might get below error

C:\Users\kishan.bhavsar\Desktop\Doc\error denied.PNG

* To resolve above error
* Execute sudo usermod -a -G docker $USER
* Then you have to reboot your VM.
* For this execute sudo reboot
* Now execute any commands like docker info or docker version.

3. Introduction to Docker Swarm Mode

* 1. What is Swarm Mode?

A swarm is a group of machines that are running Docker and joined into a cluster. After that has happened, you continue to run the Docker commands you’re used to, but now they are executed on a cluster by a swarm manager. The machines in a swarm can be physical or virtual. After joining a swarm, they are referred to as nodes.

Swarm managers are the only machines in a swarm that can execute your commands, or authorize other machines to join the swarm as workers. Workers are just there to provide capacity and no not have the authority to tell any other machine what it can and cannot do.

* 1. Set up your Swarm.

A swarm is made up of multiple nodes, which can be either physical or virtual machines. The basic concept is simple enough: run dokcer swarm init to enable swarm mode and made your current machine a swarm manager, then run docker swarm join on other machines to have them join the swarm as workers

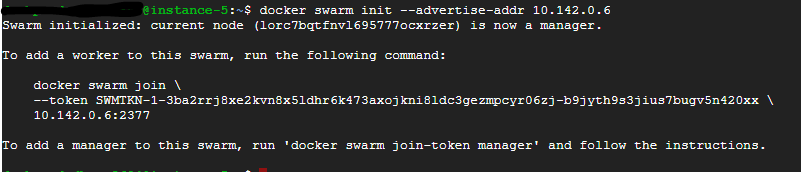
1. Setting up Docker Swarm mode

4.1 Docker swarm mode

- To set up Docker swarm mode you require one more VM. So create one more VM. Go through the Creation of Ubuntu VM in Google Cloud Console.

- Install Docker in the VM.

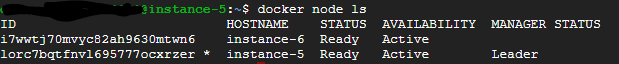
* Now check the internal IP of VM which you want to make Swarm manager.
* Execute docker swarm init –advertise-addr internal IP
* After executing this command your current VM is Swarm Manager.
* You will get one token. This token is useful to join worker VM with Swarm Manager.



* You have one more VM which you made at starting of Docker swarm mode.
* Now copy docker swarm join --token SWMTKN-1-3ba2rrj8xe2kvn8x5ldhr6k473axojkni8ldc3gezmpcyr06zj-b9jyth9s3jius7bugv5n420xx 10.142.0.6:2377
* Past this token into another VM and press Enter. You will get the result of node is joined to Swarm as a worker.

C:\Users\kishan.bhavsar\Desktop\Doc\worker.PNG

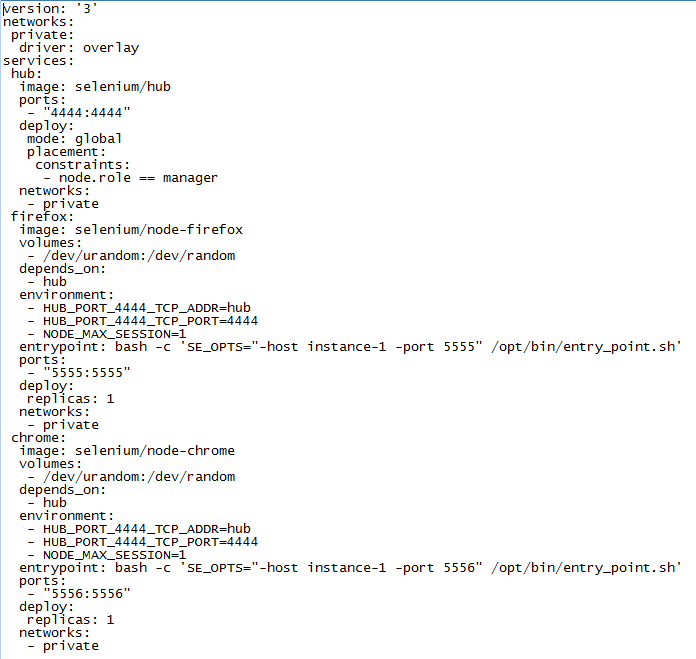
* As you can see your Swarm mode is ready.
* To check whether worker joined with Swarm manager or not
* Execute docker node ls command in Manager VM. You will get below output.



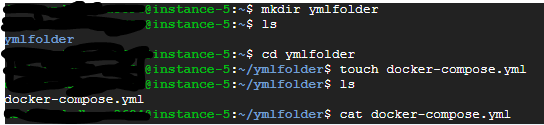
* Here instance-5 is swarm manager, so in manager state it is mention Leader and instance-6 is worker.

1. Docker compose file in Docker Swarm
   1. Execution of docker-compose.yml

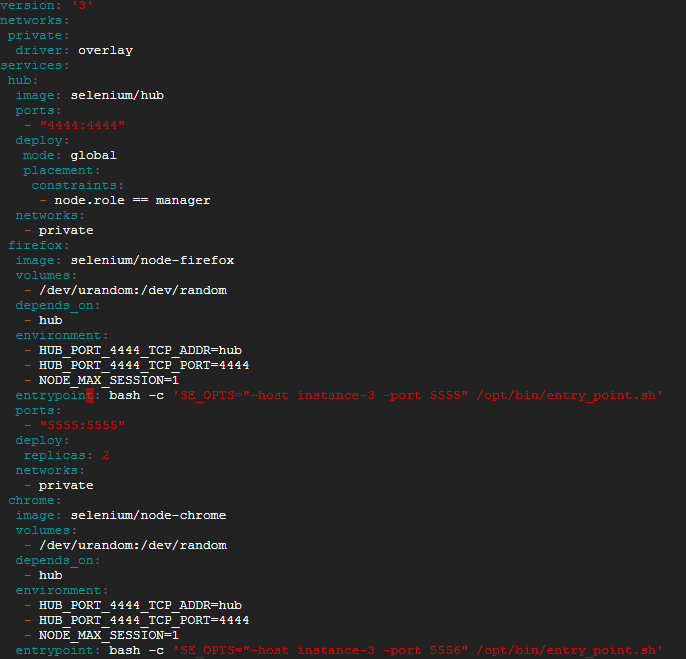
* Here is the docker-compose.yml file for Selenium Grid with Docker Swarm.



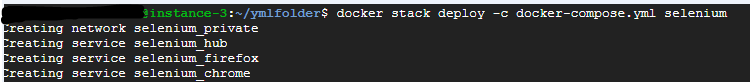
* Now go to the Manager’s VM.
* Create directory using mkdir command.
* Execute mkdir ymlfolder
* To check whether ymlfolder is created or not execute below command.
* Execute ls
* Now to go inside ymlfolder execute below command.
* Execute cd ymlfolder
* Create docker-compose.yml file by using below command.
* Execute touch docker-compose.yml
* To check whether docker-compose.yml file is created or not execute ls.
* Execute ls
* In output you will get docker-compose.yml
* To check, is there any content in docker-compose.yml file execute below command
* Execute cat docker-compose.yml



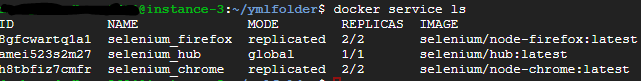
* To add above docker-compose.yml file(Image) content in docker-compose.yml execute below command and then press I and add the content.
* Execute vi docker-compose.yml



* After adding content in docker-compose.yml file. Press :wq to save content in the file.
* To execute above docker-compose.yml file execute below command
* Execute docker stack deploy -c docker-compose.yml selenium
* After executing above command you will get below output.



* To check for the services are running on nodes or not OR replicas are created or not. Execute below code.
* Execute docker service ls
* You will get below output.



* Now that replicas are created and service are up and running.
* You can check the images are there or not. By executing below codes.
* Execute docker images
* You can also check that which node is working on which instance by executing below code.
* Execute docker service ps [name of service]
* For Firefox: Execute docker service ps selenium\_firefox
* You will get below output.

C:\Users\kishan.bhavsar\Desktop\Doc\fire.PNG

* For Chrome: docker service ps selenium\_chrome
* You will get below output.

C:\Users\kishan.bhavsar\Desktop\Doc\chrome1.PNG

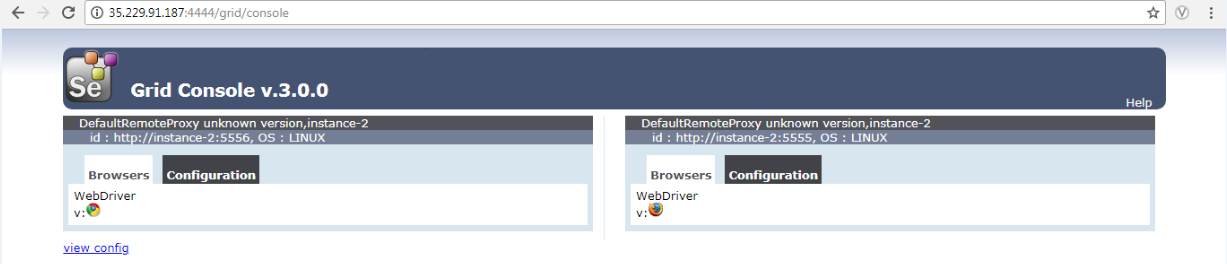
* For Hub: Execute docker service ps selenium\_hub
* You will get below outpu.

C:\Users\kishan.bhavsar\Desktop\Doc\hub.PNG

6. Nodes in Grid Console

6.1 To check nodes in grid console

* After setting up docker swarm mode and executing docker-compose.yml file.
* You can check grid console using both VMs external IP address.
* You can find external IP of your instance in the table of instances in VM instances section.
* Now to check the nodes in grid console.
* Run <http://(externalip):4444/grid/console>
* You can run above link with any of the VMs external IP which are there in Swarm mode.
* You will get same output with VM’s IP which is there in Swarm mode.

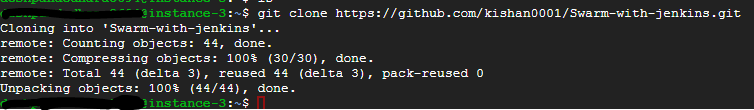




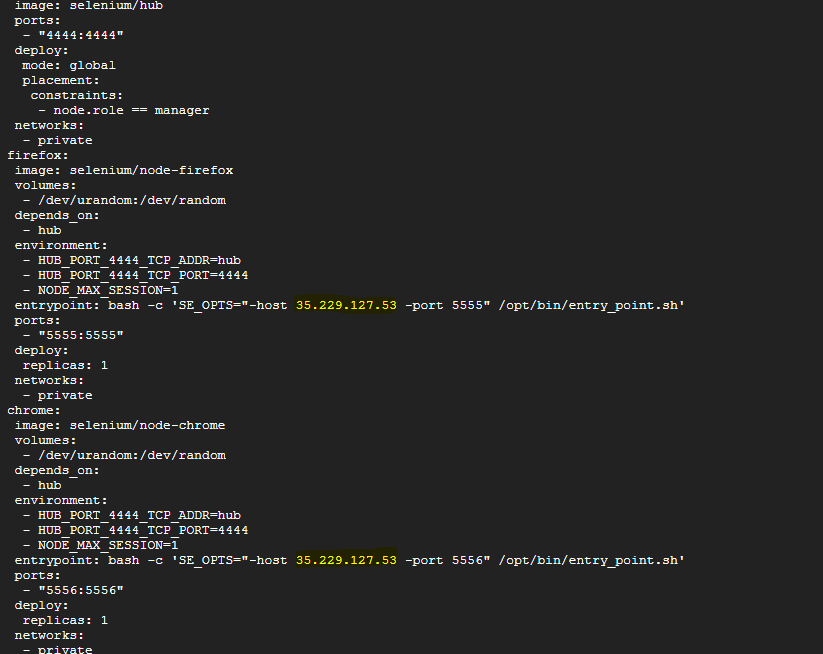
1. Execution of Selenium Grid code
   1. Code Execution

* To execute Selenium Grid code, we required Selenium grid code.
* So I clone the code from github. To clone the code from github do following steps.
* Go inside your Google VM.
* Go into the directory where you want to clone the code from github.
* Now execute below command.

Git clone <url>



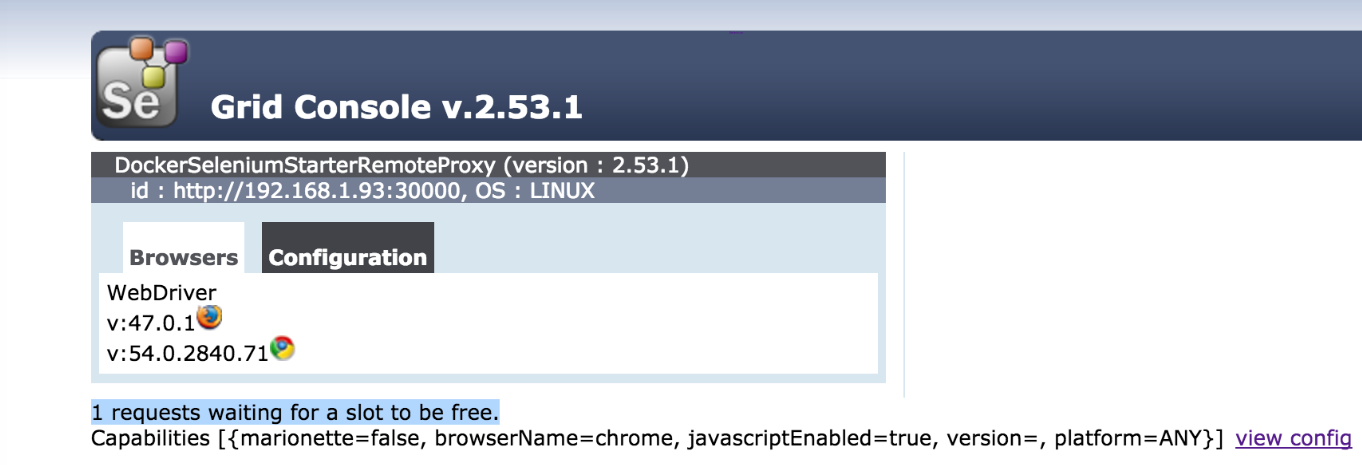
* Now you have all the files which are there in github.
* I used maven project for Selenium gird.
* So I manually install maven, JDK in vm.
* To install maven in VM. Execute below code.
* sudo apt-get install maven
* To check whether maven is installed or not, you can execute below code.
  + mvn --version
* To install JDK in Vm. Execute below codes.
  + sudo apt-get update
  + sudo apt-get install openjdk-8-jdk
* Before execution of code, just go inside the folder where you clone code from the github and check the whole code.
* Now come out from that folder and go inside the folder where your pom.xml is. And execute below codes.
  + mvn clean
  + mvn install
  + mvn test
* Go inside the target folder there you will find some jar files.
* You will find dependencies jar. Now execute that jar. To execute jar file execute below code.
  + Java –jar <jarfilename>
* After executing it you might get below error.
  + “Could not start a new session. Possible causes are invalid address of the remote server or browser start-up failure”
* To overcome this error, you have to do one change in docker-compose.yml.



* + As shown in the image you have to write external ip of the VM instead of name of the VM.
  + Also update the remote url with external IP in selenium code. Which is

driver = new RemoteWebDriver(new URL("http://35.229.127.53:4444/wd/hub"),dcp);

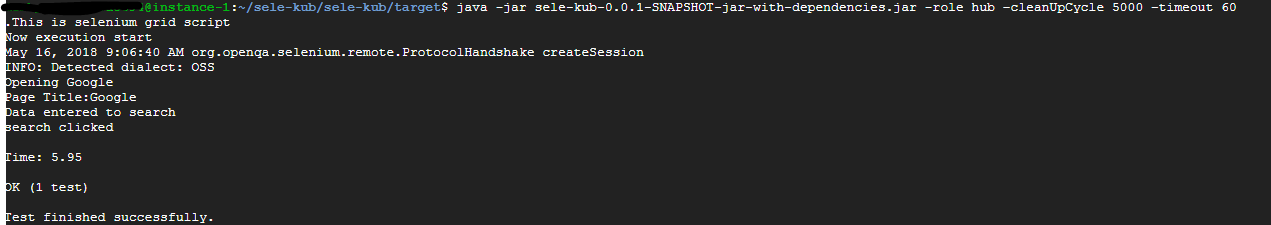
* Error will resolve after following above steps.
* Now you can execute your code again and it will execute successfully.
* But if you execute your code again in a minute you might get an error of “1 request waiting for a slot to be free”.



* It is because your node is still consumed.
* To overcome this error, you have to add command after java –jar <jarfilename> as shown below.

- Java –jar <jarfilename> -role hub –cleanUpCycle 5000 –timeout 60

- So now if you execute your code again and again you will not face any error.

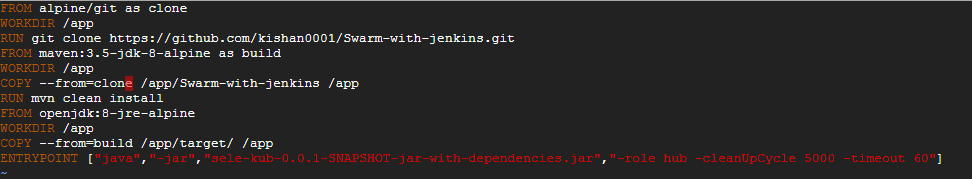


1. Execution of Selenium Grid code using Docker file
   1. Code Execution using Docker file

* To execute Selenium code using dockerfile, you have to first create it as per your requirement.
* So first in your preferred directory create on folder for Docker file.
* Go inside that folder and execute below code.
* touch Dockerfile(Remember you have to create docker file with ‘Dockerfile’)
* As shown below, after executing touch Dockerfile, you will have Dockerfile in the folder.

C:\Users\kishan.bhavsar\Desktop\Docker Work\Doc\dockerfile folder.PNG

* Go inside Dockerfile , to edit dockerfile execute below code.
  + Vi Dockerfile
* Add specific content which you want to execute. As shown below.



* Press Esc then :wq and Enter.
* Now your content is save inside your Dockerfile.
* Overview of Dockerfile
  + First is base image which is git. Which install git
  + Second we created /app.
  + Then pull all the files from my github to /app by using git clone.
  + After that install maven.
  + Then copying all the content from Swarm-with-jenkins to /app
  + Then executed mvn clean install command.
  + Then install jdk.
  + Then copying the content from the target folder into the /app
  + At the end executed the jar file.

- So the above image does each and every thing which you have done in execution of selenium grid manually.

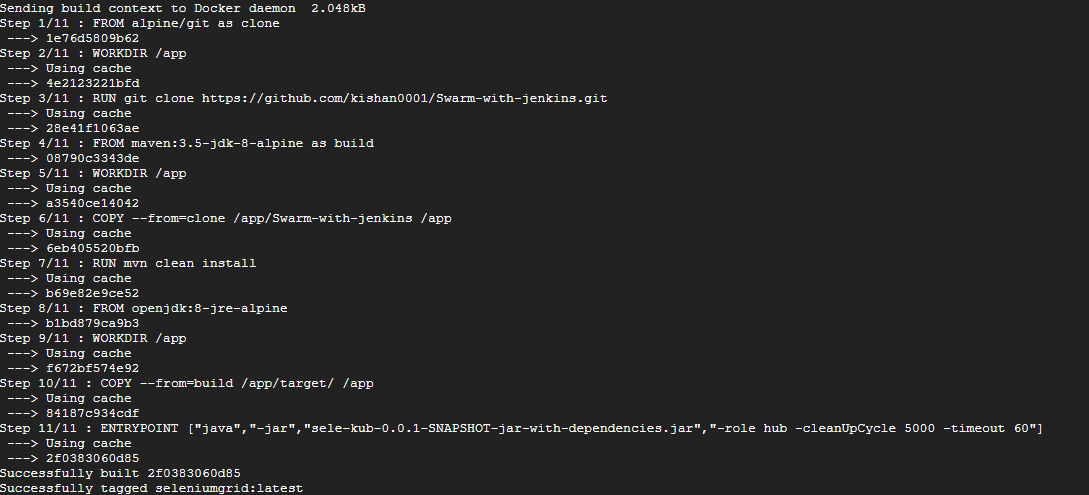
- All the installation of Maven, JDK and all the command execution which you did in Execution of Selenium grid code part is now done by adding some lines in Dockerfile.

- To execute Dockerfile, execute below command.

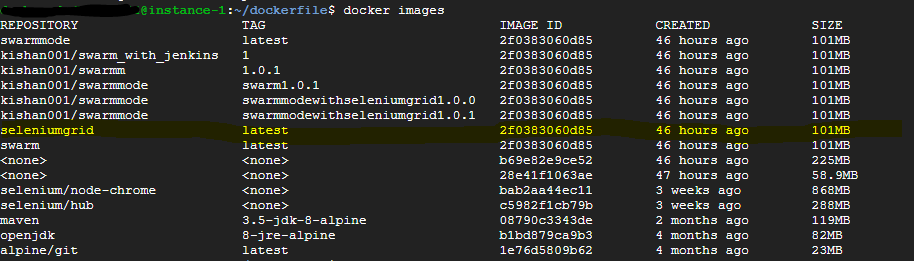
- docker build –t seleniumgrid .

- seleniumgird = Name for image.

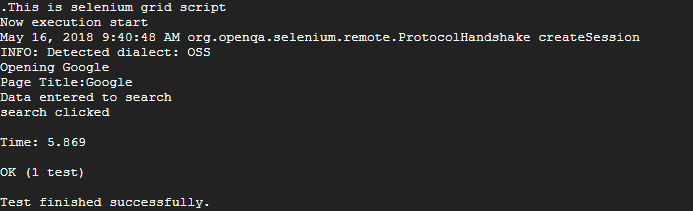
- Above code will build the image. As below



* If image build successfully, Execute docker images command. You will find the image with a name seleniumgrid.

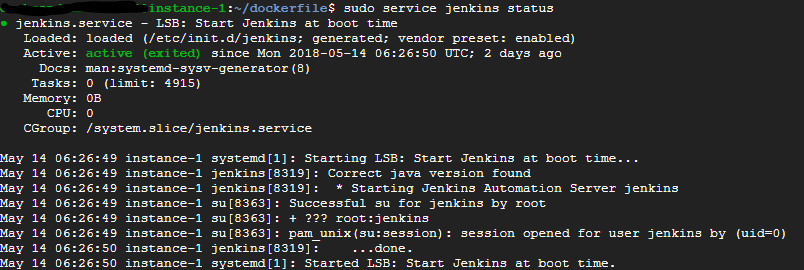


* Now to run this image. Execute below code,
* Docker run seleniumgrid .
* You will find the output.



1. Execution of Selenium Grid code using Docker file from Jenkins.
   1. Code Execution From Jenkins.

* Let’s start with installation of Jenkins in Ubuntu.
* To install Jenkins in Ubuntu VM follow below steps.
  + First add repository key to the system.
  + Execute this command.
  + wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add –
  + Then execute below code.
  + echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list
  + Then execute : sudo apt-get update
  + Then execute : sudo apt-get install Jenkins
  + To check weather Jenkins is installed or not just execute below code.
  + sudo service jenkins status
  + If Jenkins install successfully then you will get below output.



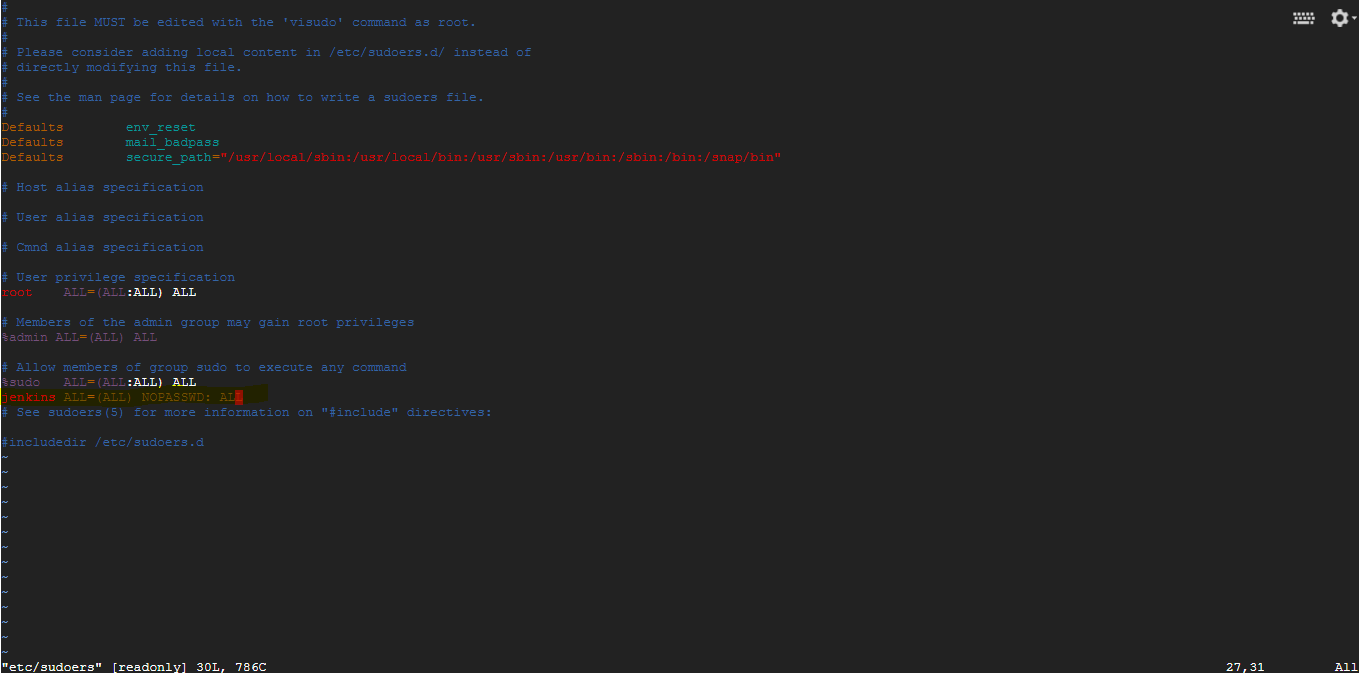
* + So now you can use url : http://<externalip>:8080
  + It will ask for administrator password, just go to the location (Which is mention on the page in browser) and you will get administrator password.
  + So now Jenkins is started. Create your account in it.
* Execution of Selenium Grid code in Jenkins using Dockerfile.
  + Click on New item.
  + Enter your item name.
  + Select freestyle project.
  + Click on ok.
  + Now you are inside configuration part.
  + Go to the Build section.
  + In Add build step, select Execute shell.
  + Now go step by step.
  + First clone the image from github, for that execute below command.
    - git clone <url>
    - ls
    - So now you have all files from github in Jenkins.
    - Go inside the directory
    - cd <directory name>
    - ls
    - now try to execute Dockerfile from Jenkins, by using below command.
    - docker build –t selenium .
    - You might get an error of
      * Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post



* + - To over come this error. Always execute command with sudo.
    - So execute sudo docker build –t selenium .
    - You might get an error of “tty present and no askpass program specified Build step 'Execute shell' marked build as failure”



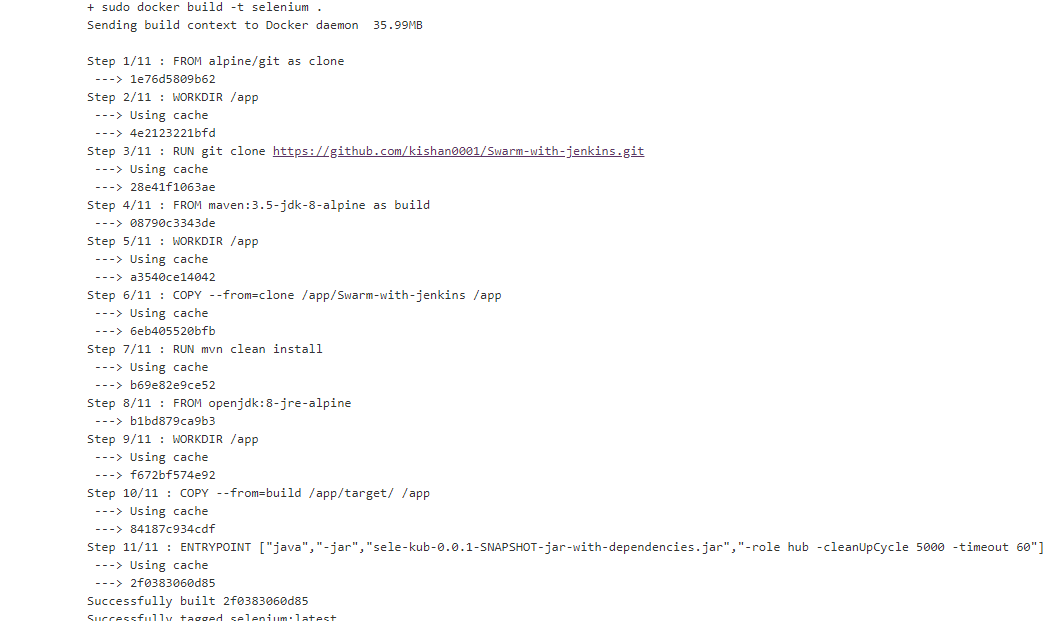
* + - To overcome this error follow below steps
      * Go out from all the directories in VM.
      * Execute below command.
      * sudo vi etc/sudoers
      * Add this line in file : “Jenkins ALL=(ALL) NOPASSWD: ALL”



* + - * Then press Esc, :wq! And Enter
      * The content is successfully saved into the sudoers file.

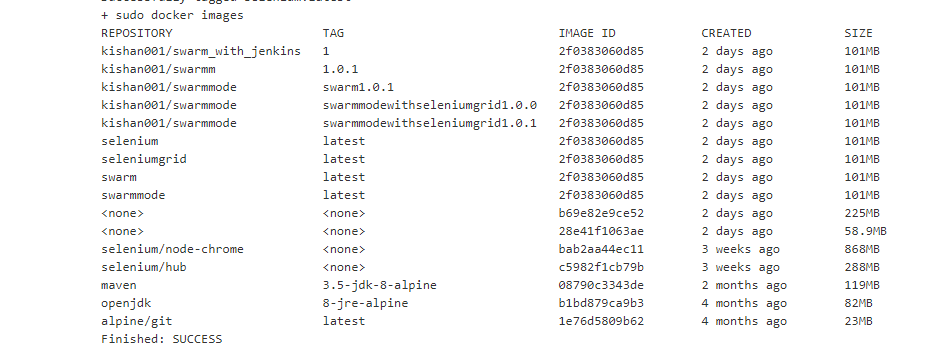
- Now execute code in Jenkins till sudo docker build –t selenium .

- It will build the image.



- Execute sudo docker images.

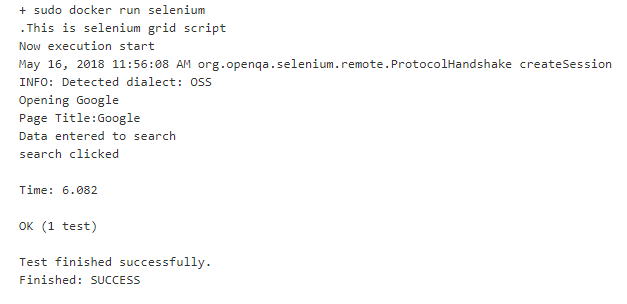
- You will find below output.



- Now execute the image, by executing below code.

- sudo docker run imagename

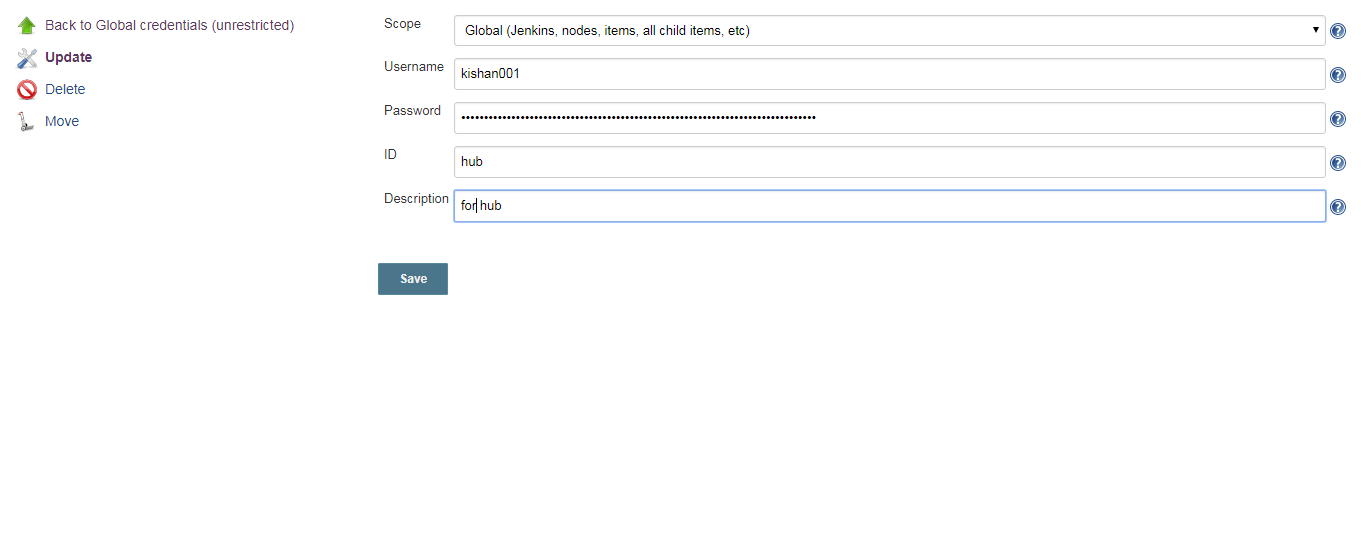
- After executing image you will get below output.



* + This is how you can execute docker file from Jenkins.

1. Push Docker Image to Docker hub from Jenkins.
   1. Push image to hub.

* You can push image from Jenkins to the docker hub.
* First you have to create your repository in hub.docker.com.
  + First sign up into hub.docker.com.
  + Click on create repository.
  + Choose a name and description for your repository and click on create.
* Now use Jenkins to push image to hub.
  + To do that first go to the Jenkins.
  + Click on Manage Jenkins.
  + Click on Manage Plugins.
  + Click on Installed.
  + Check for Credentials Binding Plugin. If it is not there, then install it.
  + Now go to credentials.
  + Click on Jenkins.
  + Click on Global Credentials.
  + Then give your Username.
  + Then give your password.
  + Give ID and Description.



* + Now go to the project.
  + Click on configure.
  + Go to the Build Environment.
  + Click on the Use secret text(s) or file(s) check box.
  + In Bindings, click on the Add dropdown.
  + Click on the Username and password (separated)
  + Give username and password variable name.
  + Click on the specific credentials. And select your credential which you added in Global credentials.



* + After that go to the Build.
  + Click on the Execute Shell.
  + Execute below codes.
    - sudo docker images
    - sudo docker login -u=$username -p=$password
    - sudo docker tag 2f0383060d85 kishan001/swarm\_with\_jenkins:1
    - sudo docker push kishan001/swarm\_with\_jenkins:1

- Here sudo docker login –u=$username –p=$password taking the username and password from Global credentials. So your username and password Is not visible to anyone.

* + Make sure that in, sudo docker tag 2f0383060d85 kishan001/swarm\_with\_jenkins:1, you are using specific tag.
  + Now your image push to the docker hub.