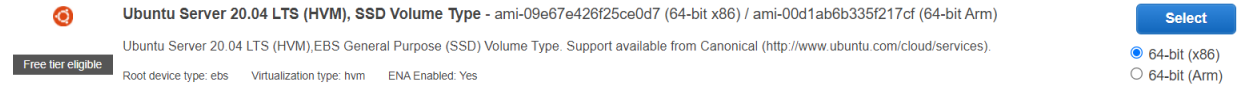
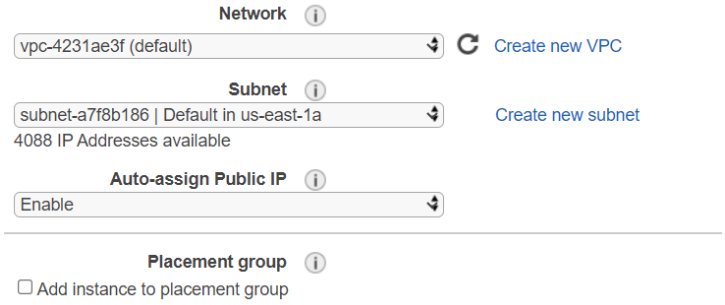
Steps:

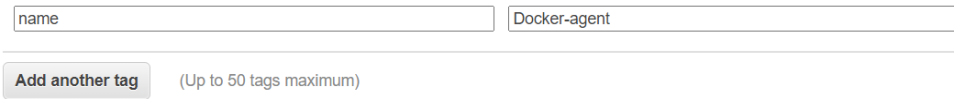
1. Create the ec2 using the Ubuntu Ami



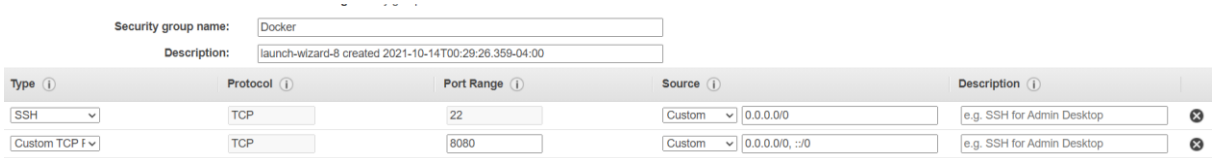
2. Select the subnet for the region you're in and enable it.



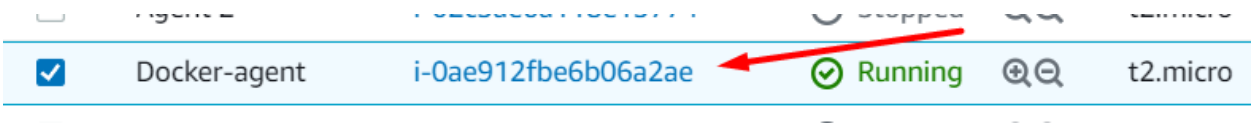
3. Give your ec2 a name.



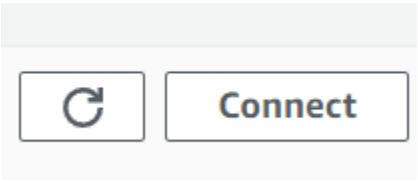
4. Set the security group rules as shown.



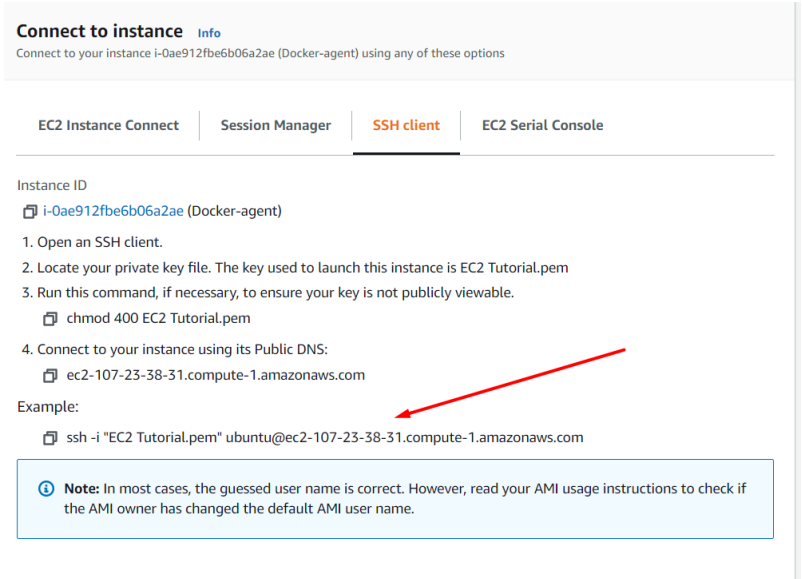
5. Once the ec2 is made and running, click the instanceID.



6. Then click on connect.



7. Ensure that you're in the SSH client section then copy the address.



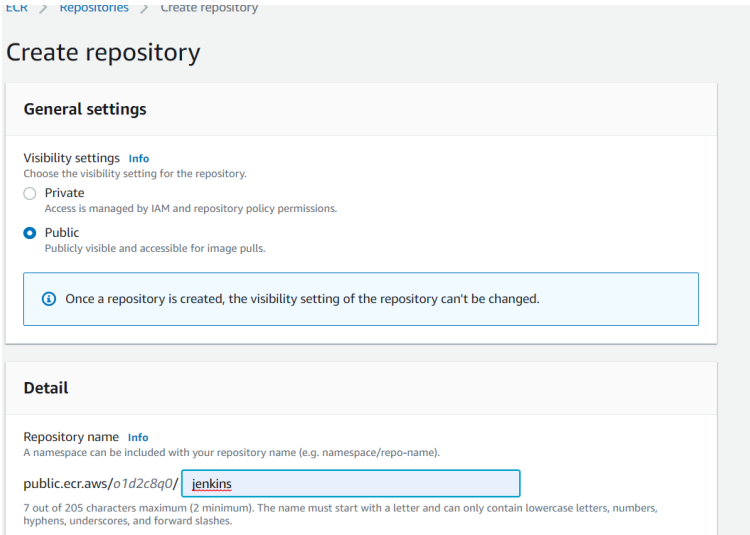
8. Once copied head inside your terminal to ssh into your instance. Ensure that your "pem" is typed correctly or you'll get an error.

```
ssh -i "EC2Tutorial.pem" ubuntu@ec2-107-23-38-31.compute-1.amazonaws.com
```

9. Once successful, follow the steps to in docker onto ubuntu from this site https://serverspace.io/support/help/how-to-install-docker-on-ubuntu-20-04/.
   NOTE: ensure that you use the required username as shown below:

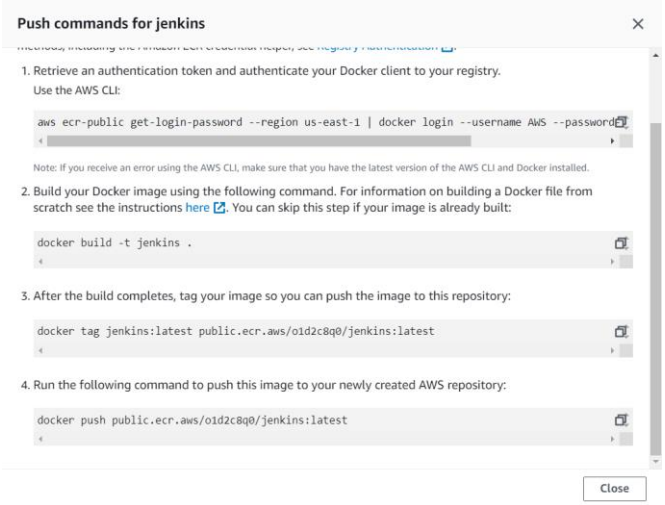
```
sudo usermod -aG docker ubuntu
```

10. Once you're done, use the command ($ sudo docker images) to test if docker was successfully installed.
11. Then use the command ($ curl "https://awscli.amazonaws.com/awscli-exe-linux-86_64.zip" -o "aws.cliv2.zip").
12. Once the zip is downloaded use the command ($ sudo apt-get install unzip).
13. Then use the command ($ sudo unzip aws.cliv2.zip).
14. Then use the commands ($ sudo ./aws/install) and ($ aws –version) to see the version.
15. Once aws is successfully installed used the command ($ aws configure).
16. Fill in the information correctly then use the commands ($ sudo apt update) followed by ($ sudo apt install default-jdk).
17. Then use the command ($ docker pull Jenkins/Jenkins).
18. Then head into aws online and search for the Elastic Container Service(ECS). Once there create your Elastic Container Respository(ECR). Set the visibility to public and give it a name and keep the recommended settings.

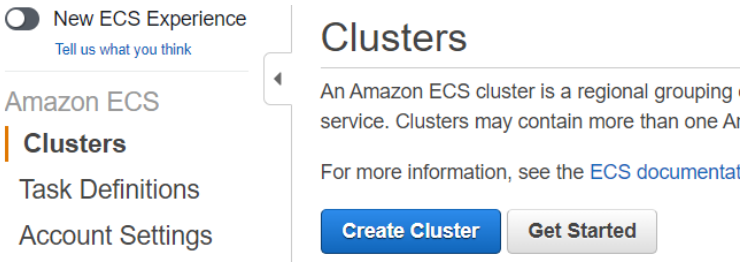


19. Click your new repository and clicked push commands.

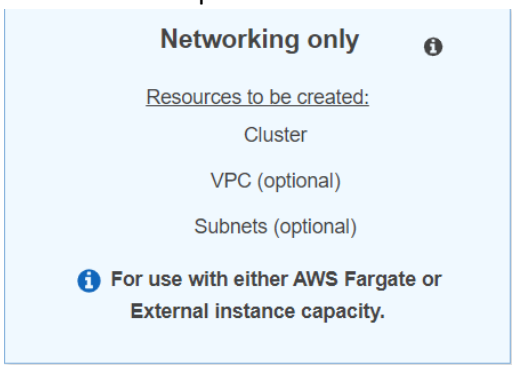
20. From push commands follow steps 1, 3 and 4. Ensure that you get a successful login from step 1 before moving on. After step 3 you can use the command ($ docker images) to see the new repository made.



21. Once done, create a new cluster.

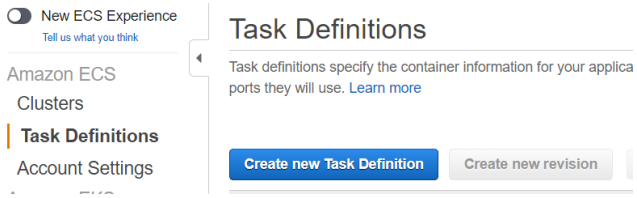


22. Select the first option

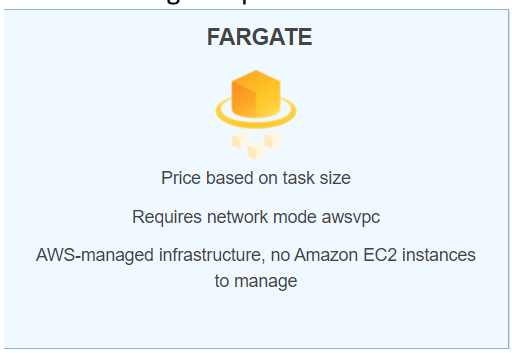


23. Then name your cluster and save changes.

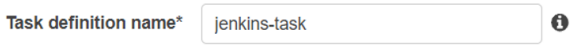
24. Then create a task definition.



25. Select the Fargate option.



26. Name your task definition.



27. Most importantly ensure that you select a reasonable task size. This is important because you'll be using Jenkins and if it's too small the site is going to be extremely slow.

NOTE: fargate charges by the second so ensure that you are very careful and delete once done.

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate la
for the EC2 or External launch type. Container level memory settings are optional when task size is set. Task si
Windows containers.

Task memory (GB) 4GB

The valid memory range for 2 vCPU is: 4GB - 16GB.

Task CPU (vCPU) 2 vCPU

The valid CPU range for 4GB memory is: 0.5 vCPU - 2 vCPU.

28. Add a new container

29. Give the container a name and copy the image made from the repository and paste it.

jenkins public.ecr.aws/o1d2c8q0/jenkins

Standard

Container name* jenkins-container

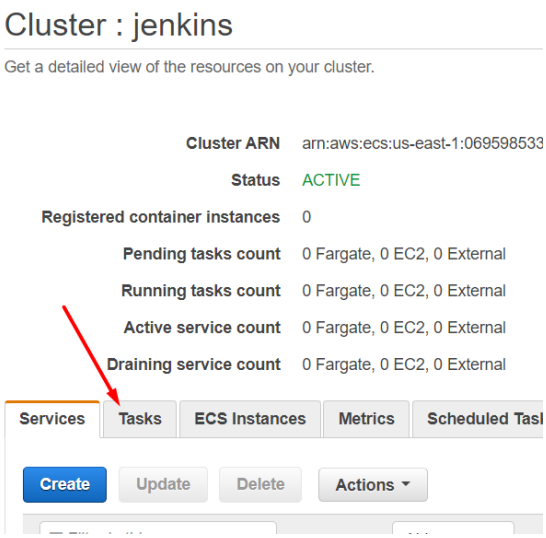
Image* public.ecr.aws/o1d2c8q0/jenkins

30. Set port mappings to 8080.

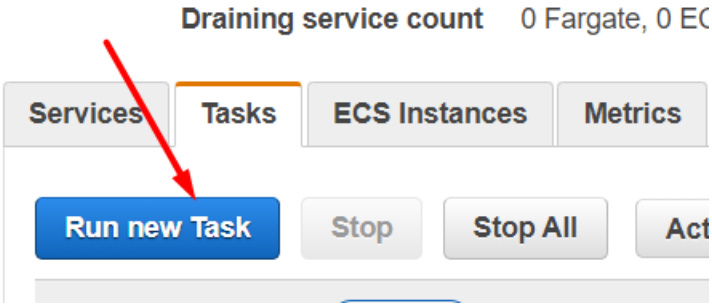
Port mappings *Container port* *Protocol*

8080 tcp

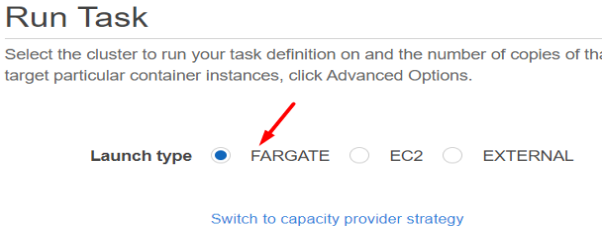
31. Go to the cluster you created and select Tasks.



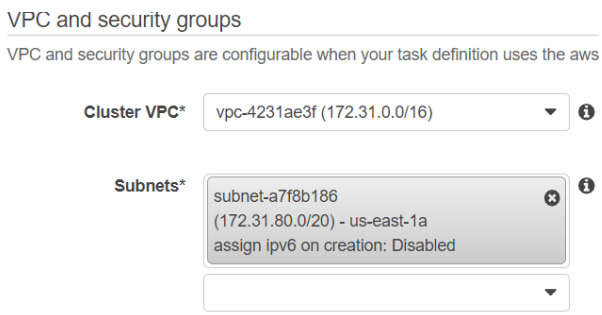
32. Then select Run new tasks.



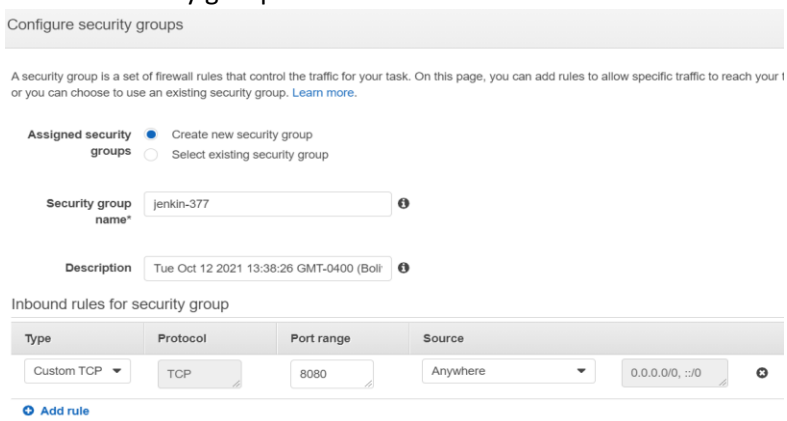
33. Select Fargate



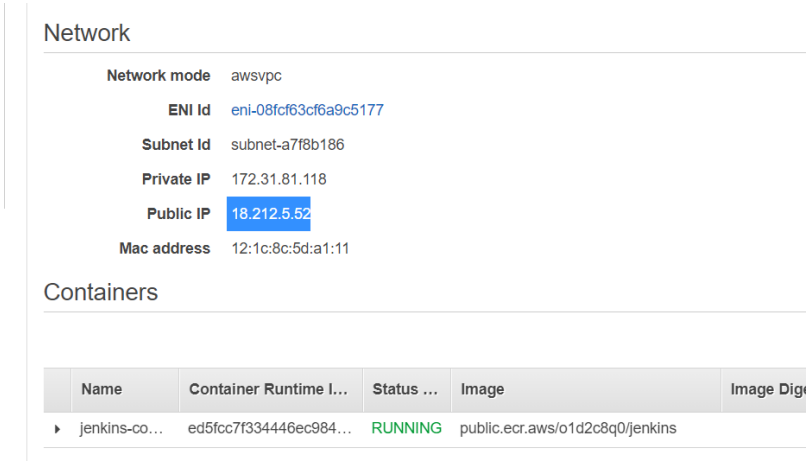
34. Choose a vpc (I used the default) and choose a subnet (I used the one with the region I'm in.



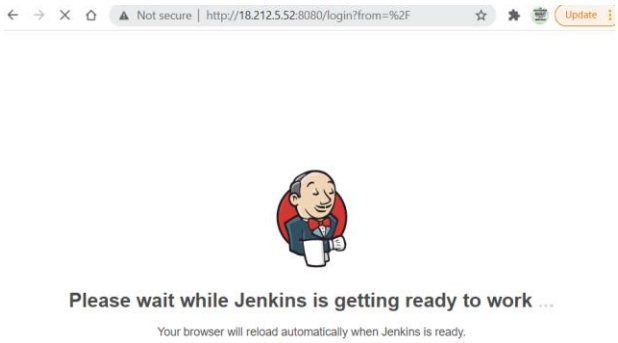
35. Edit the security group as shown:



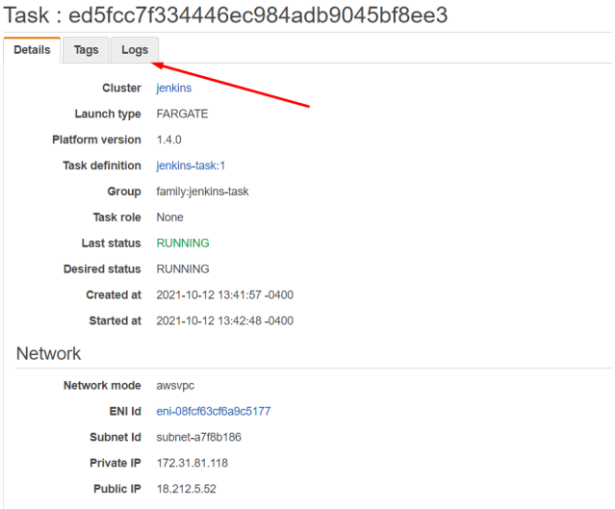
36. Save changes and wait for the task status to change from “pending” to “running”. Then copy and paste the Public IP in a new tab.



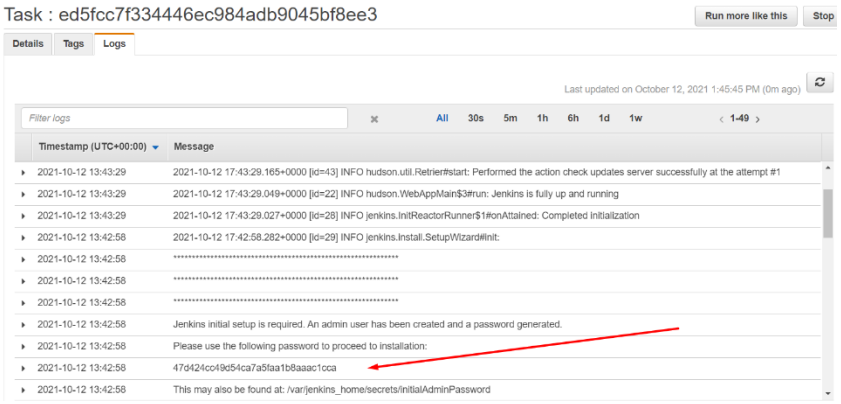
37. Once successful Jenkins will load on the new tab.



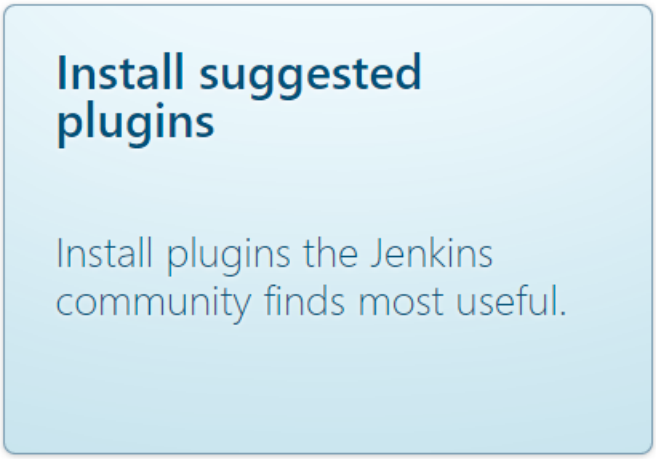
38. Once Jenkins loads successfully, head to the task and click on logs.



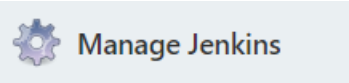
39. Once there, you'll get the password required to log into Jenkins.



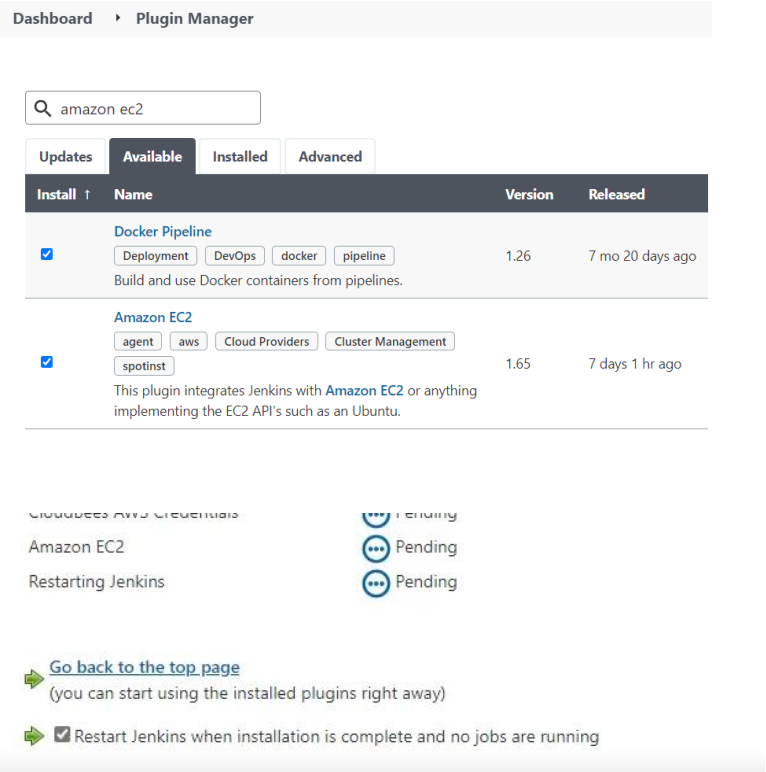
40. Install suggested plugins.



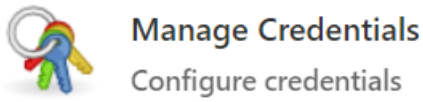
41. Click Manage Jenkins



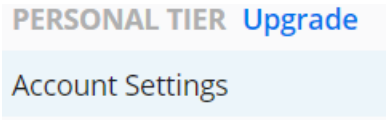
42. Install both Docker pipeline and Amazon EC2 plugins.



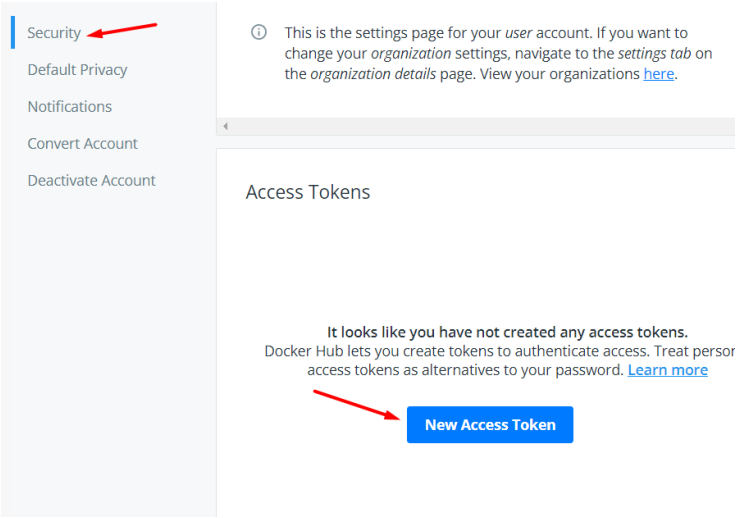
43. Then go to Manage Credentials.



44. When creating your docker credentials got to account settings.

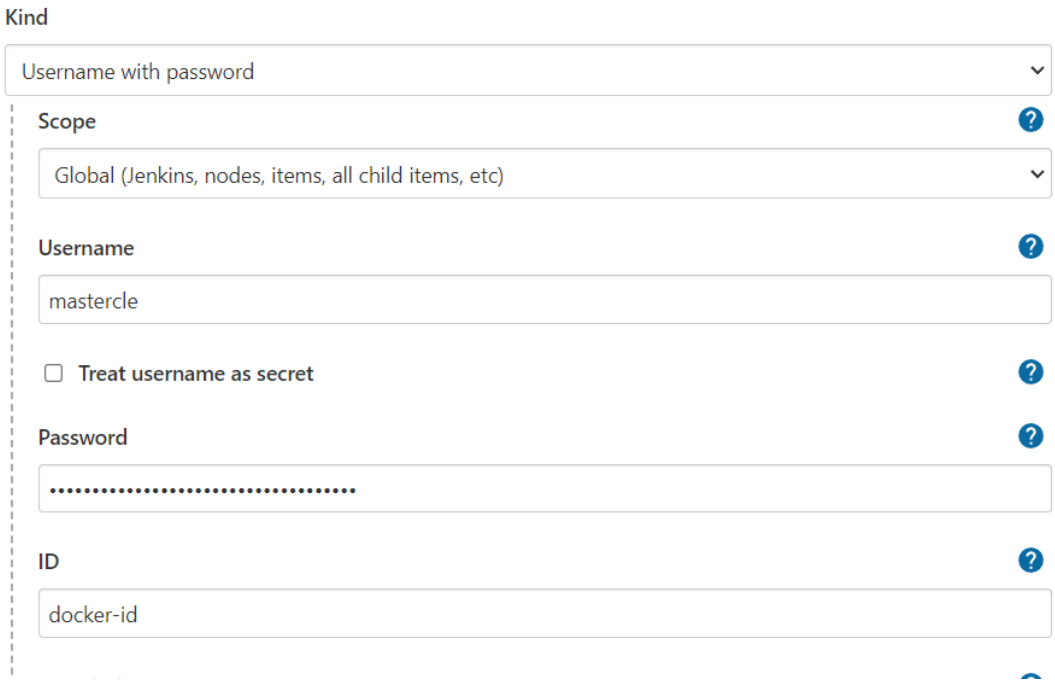


45. Then go to security, there you will create a new access token if you don't already have one.

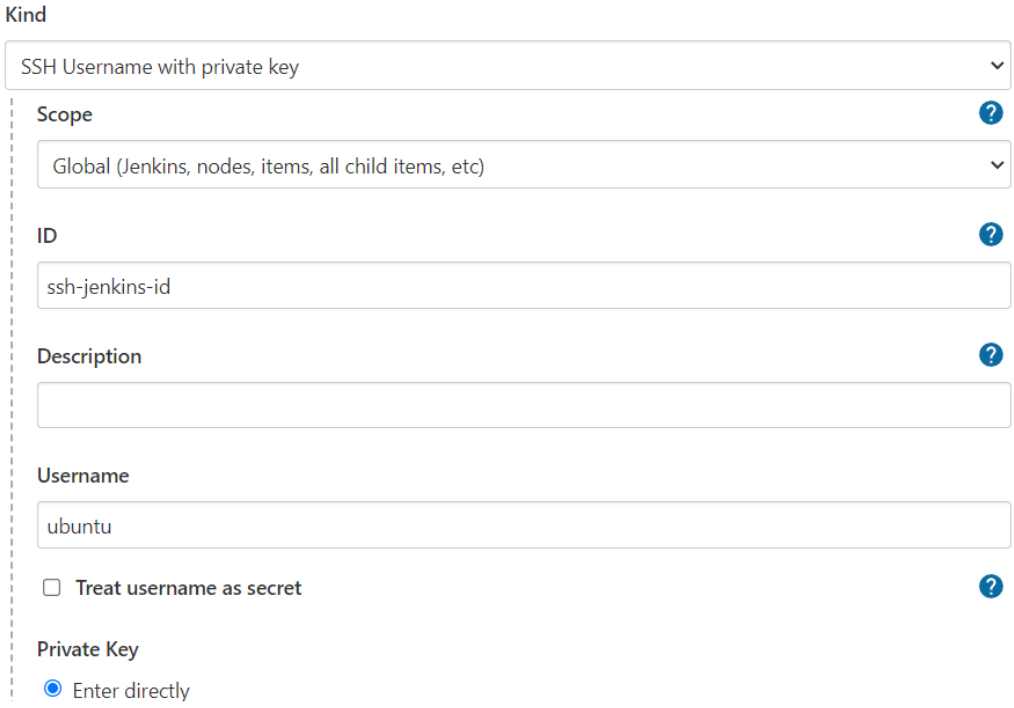


46. Then give your token a name and save it. Ensure that you copy the token and save it for future use.

47. Ensure that you choose the correct kind. Then use the username of your dockerhub account and use the access token for password. Give it an id and save it. Set credentials as shown



48. Then set your Jenkins credentials as shown:



Private Key

Enter directly

Key

Enter New Secret Below


```
YtPj+wKBgQCY7ws39qGMHTVHRWKZamcWVhUq1WbKb1HTqA9Cpy12YtWr7ApA24sy
37T9dB4DTshkRH8HNfVwrEdNlxqpyw8KEZN9xfWK/TQLWEId81W5t82BcsOi6BN5
DfQvhZvFrTH0OwStBKslcXFDd8mKSnKzFpW2WiStm8zGeJ+WmqKpYg==
-----END RSA PRIVATE KEY-----
```


Passphrase

OK

NOTE: for private key, copy and paste your rsa private key.

49. Then add your credentials for github ensuring that the username is the same as your github account and password is your personal access token. Save changes after giving it an id.

Kind

Username with password

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

Mastercle

Treat username as secret

Password

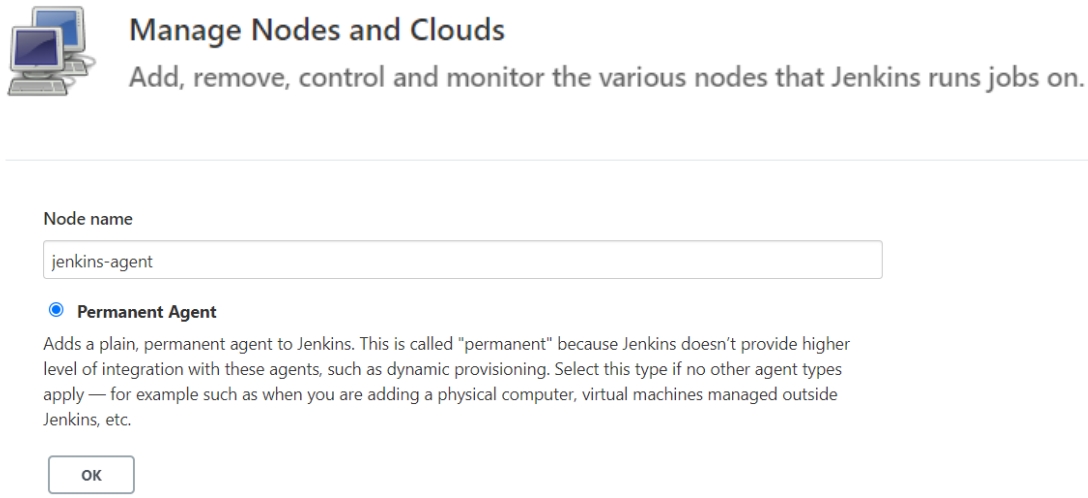
••••••••••••••••••••••••••••••••••••••

ID

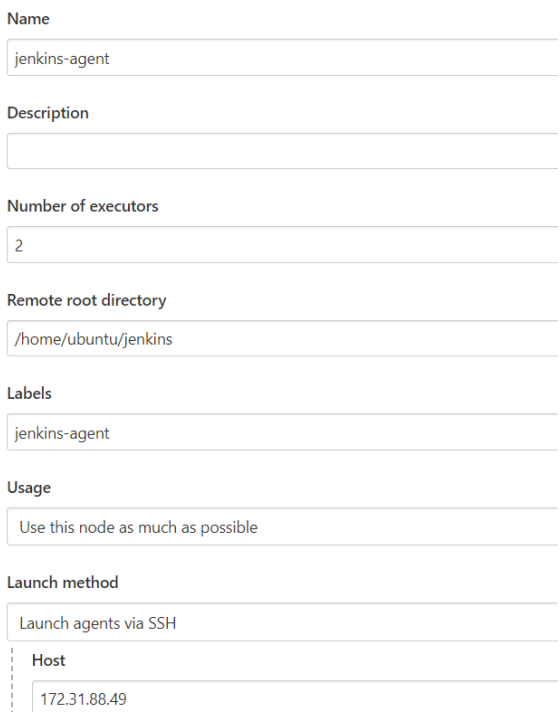
github-id

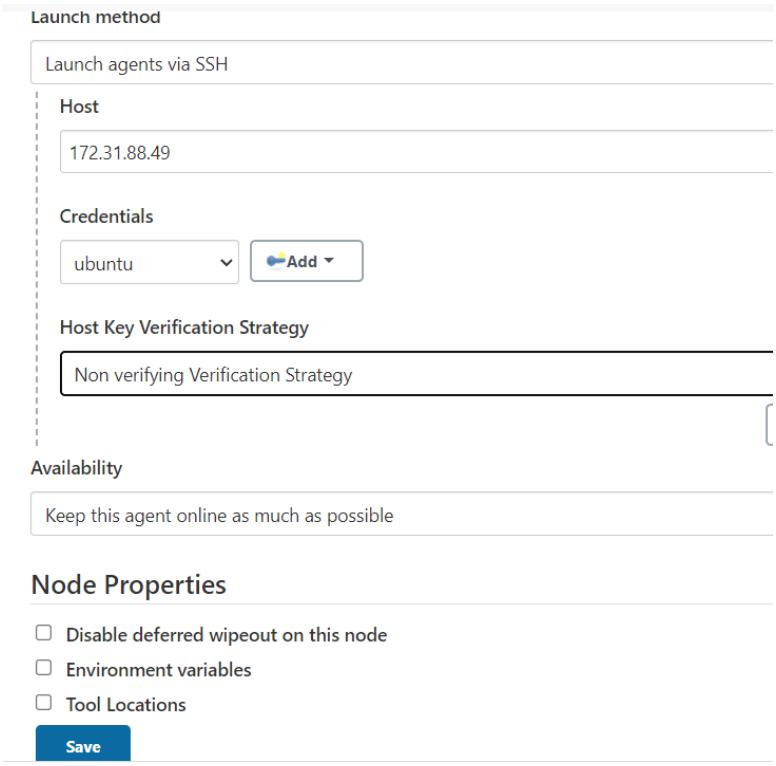
Description

50. Once the credentials are set, time to add a new node.

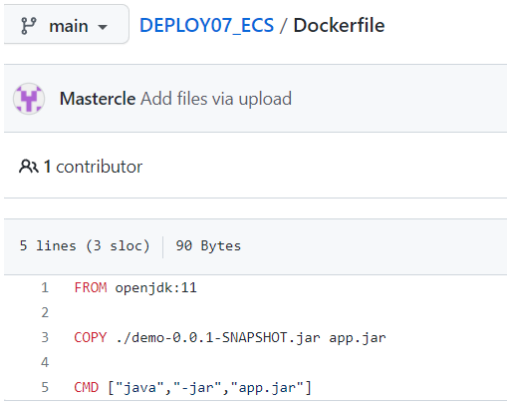


51. Configure your node accordingly giving it a name, correct number of executors (2), root directory, and a label name of your choice. For credentials, use your Jenkins credentials you created.





52. Ensure that your docker file is correct and your Jenkinsfile as well.



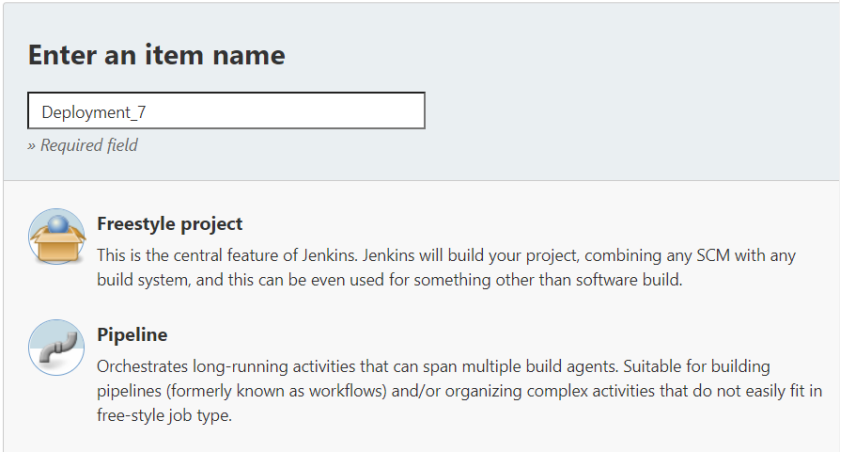
Mastercle Create Jenkinsfile

1 contributor

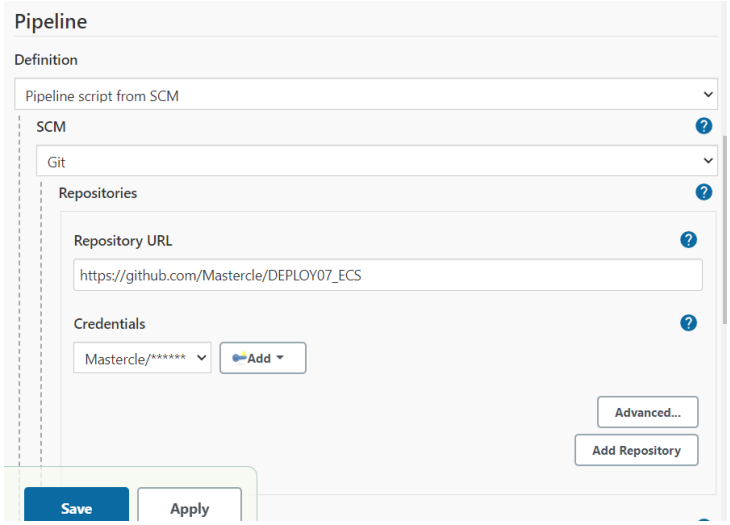
31 lines (28 sloc) | 736 Bytes

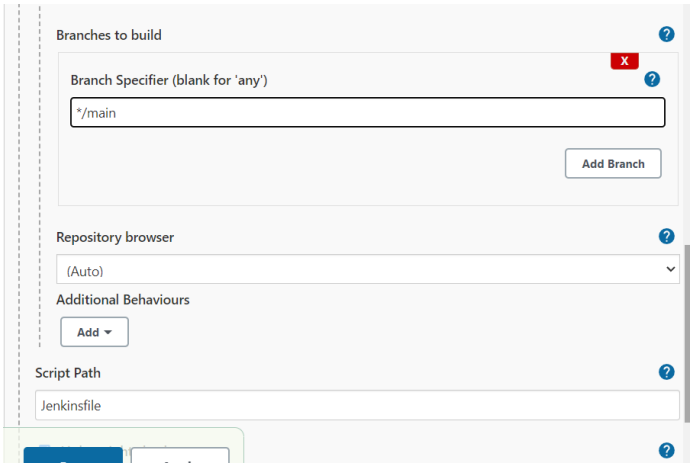
```
pipeline {
    agent {
        label "jenkins-agent"
    }

  environment {
    DOCKERHUB_CREDENTIALS = credentials("docker-id")
}
    stages {
        stage('Build') {
            steps {
                sh 'docker build -t mastercle/javadem .'
                sh 'echo "completed build"'
            }
        }

        stage('Login') {
            steps {
              sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
              sh 'echo "completed login"'
            }
        }

        stage('Push'){
            steps {
                sh 'docker push mastercle/javadem:latest'
                sh 'echo "completed push"'
            }
        }
    }
}
```

53. Then go to your dashboard to create a new item. Give it a name and select pipeline.

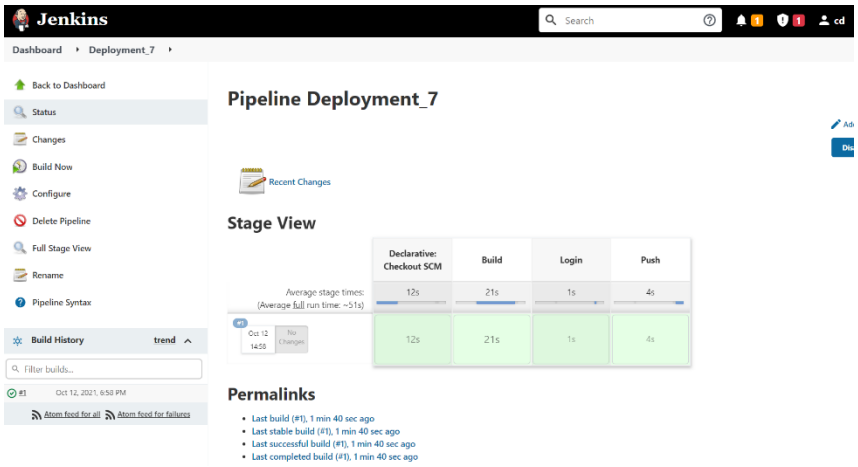


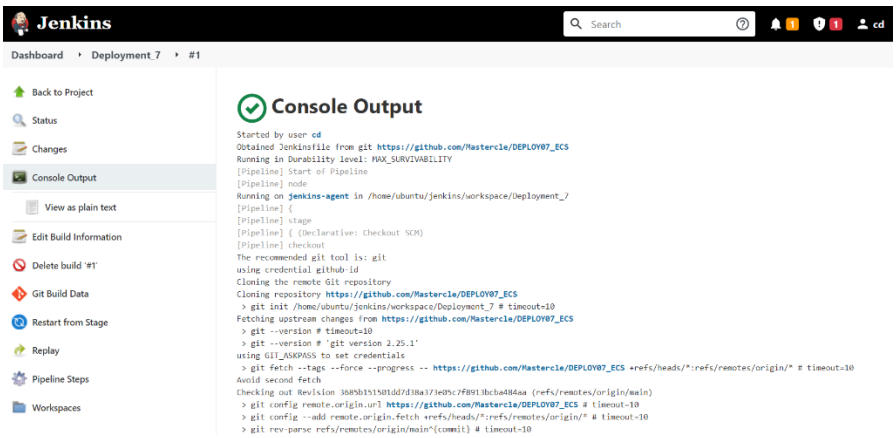
54. Set the pipeline in your Jenkins as shown, directing it to your github containing your Jenkins file. Use the github credentials you made earlier.

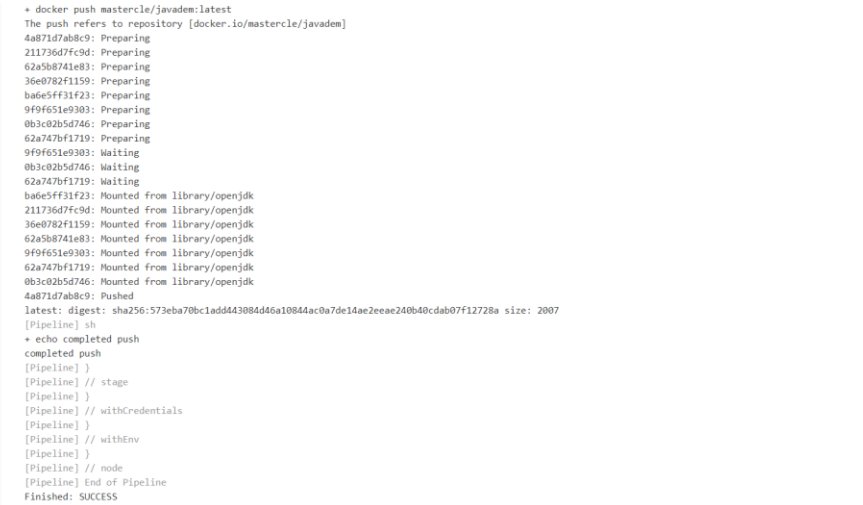




55. Save changes and build your pipeline.







56. Once it's successful head into your dockerhub to see the image made.