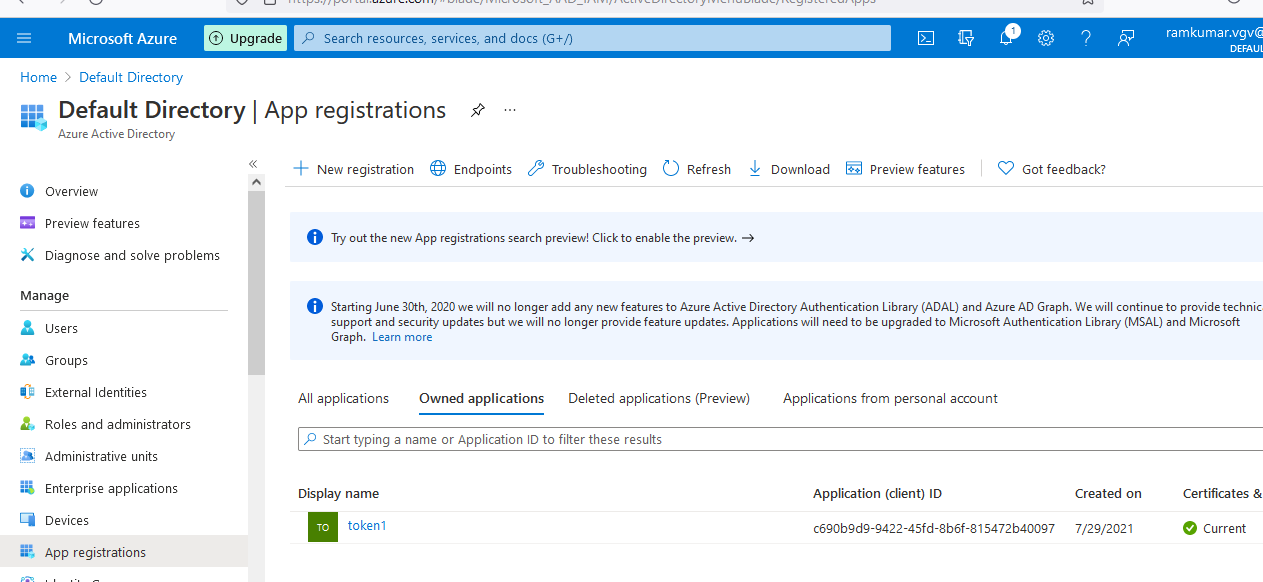
Task 1:

1)use Azure Resouce manger providers -- done

2) create Azure service principles

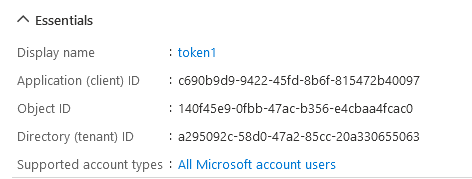
Azure - Active direcotry - App registration-New registraton

name as Token1, select Accounts in any organization and register.



click app registration which has created.

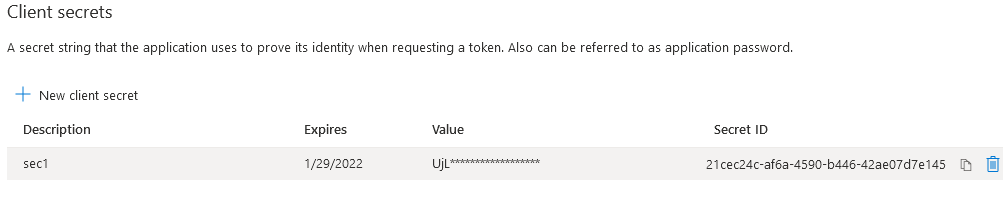
and click overview tab



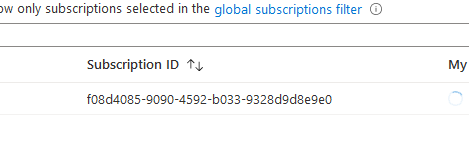
get client id and tenant id.

click certification and secrets tab.

new client secret - name as sec1. copy the secret value and save it.



subscription - free trail - subscription id



Now we have all

subscription\_id = "00000000-0000-0000-0000-000000000000"

client\_id = "00000000-0000-0000-0000-000000000000"

client\_secret = "00000000-0000-0000-0000-000000000000"

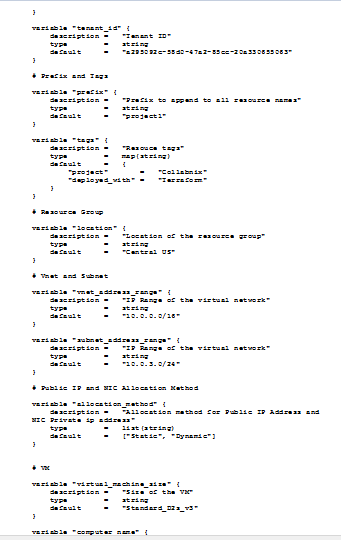
tenant\_id = "00000000-0000-0000-0000-000000000000"

keep this in above format only to use in terraform for azurerm connection.

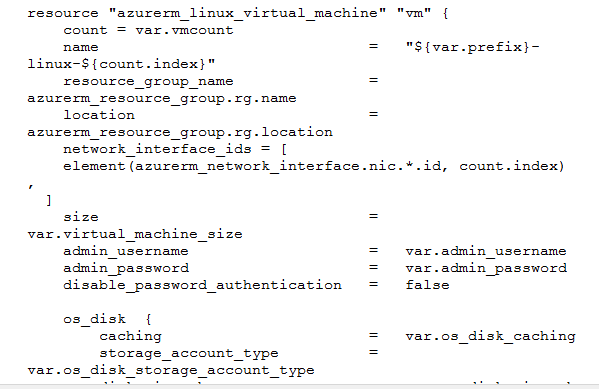
3)create vars.tf file for terrafrom

added all the tokens to connect azurerm. and required fields for location, address, subnet range Operationg system version etc..

gave prefix as project1, vmcount as 2(two vms create, can change)



4)create main.tf (linuxvm.tf)to create two vms

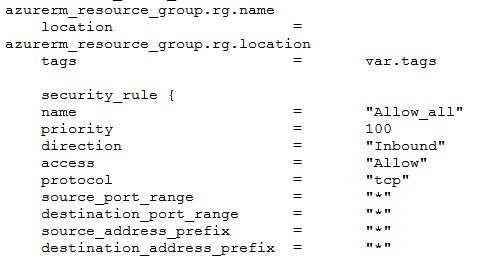


for linux machines, ip address etc.. i have used count which pulls from vars file.

used vmnames as (var.prefix).linux-(count.index)

so created two vms with project-linux-0 and project-linux-1.

added inbound rule as.



5) use terraform provider to install jdk and jenkins on first vm server

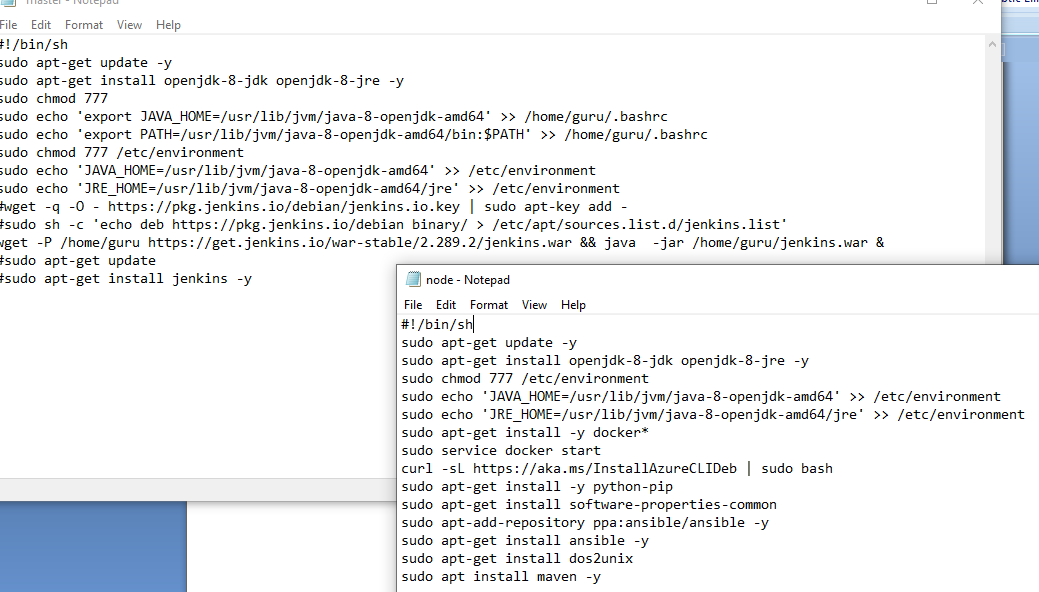
6)use terraform provider to install in second server for ansible, maven etc..

create two null resources to execute configuration and scripts on two servers.



which internal calls two scripts on each servers each one.

on master.sh has given all the required installations including jenkins.



on node.sh for second server added to install python , cli azure, ansible , maven etc...

placed all 4 files (linuxvm.tf, vars.tf, master.sh and node.sh)

7) execute terraform scipts

on powershell installed azure cli and connec-azaccount with ids.

perform below steps.

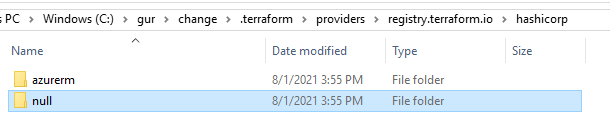
c:\gur\change :

terraform apply

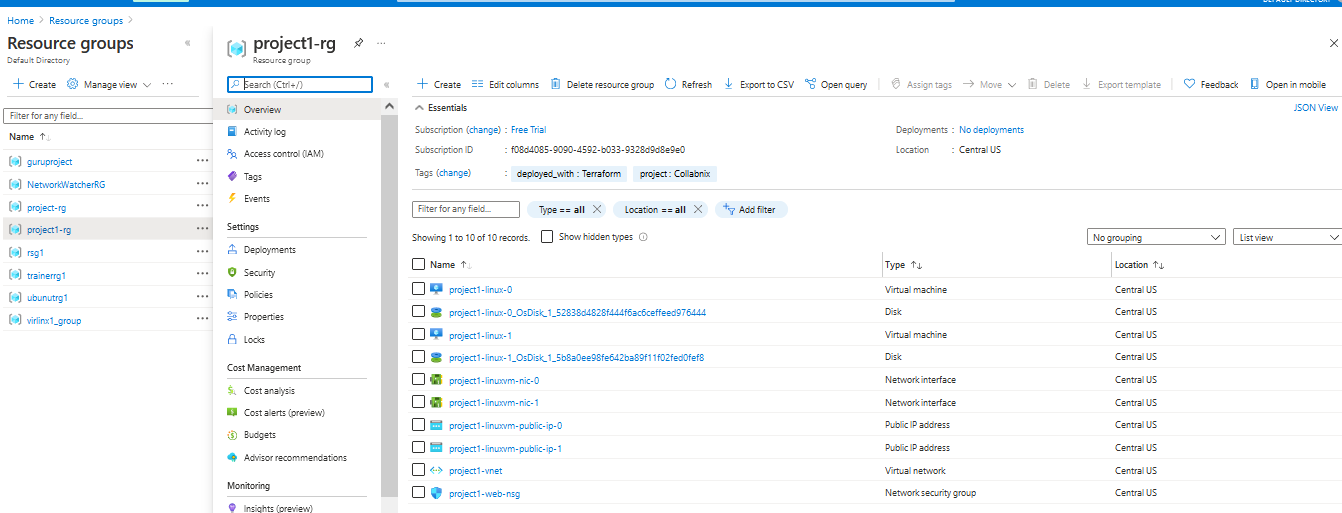
terraform validate

terraform plan

terraform apply



now resource group has creted with 10 records, all taken from vars file names.



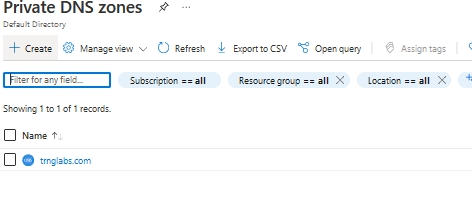
Two vm servers have public ips.

Created private dns zones for the two servers manually.

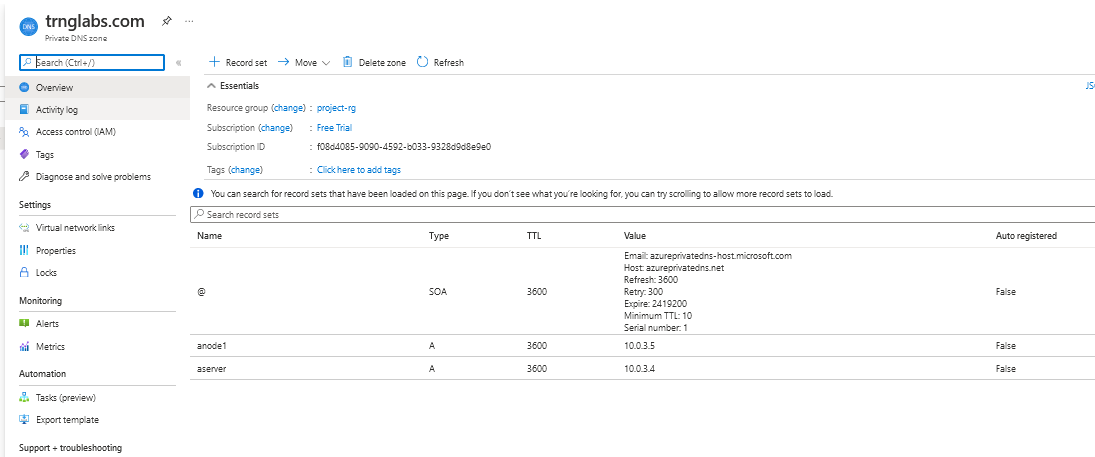
vm 0 -aserver.trnglabs.com

vm1- anode1.trnglabs.com

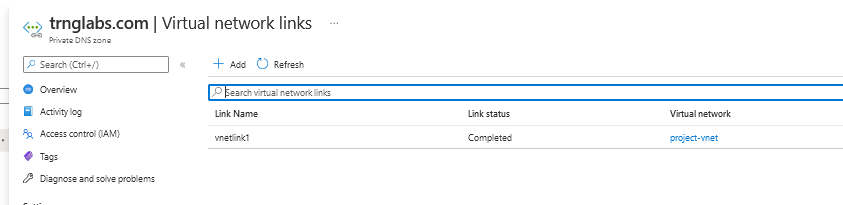
private dns zone - create



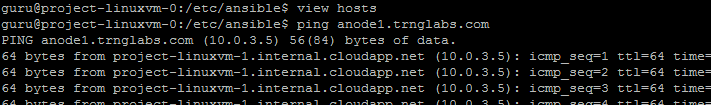
add two servers clicking record set give names as anode1 and aserver. private ips.



virtual networks links add as vnetlink1

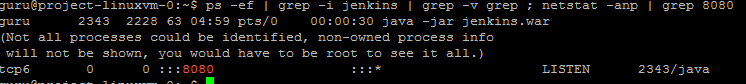


now can able to ping each other server with dns names.



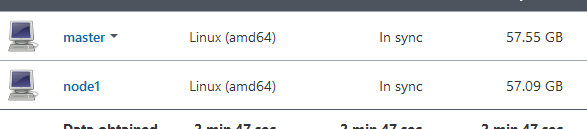
9) Manually start jenkins and required plugin

Jenkins has started with terraform script only.

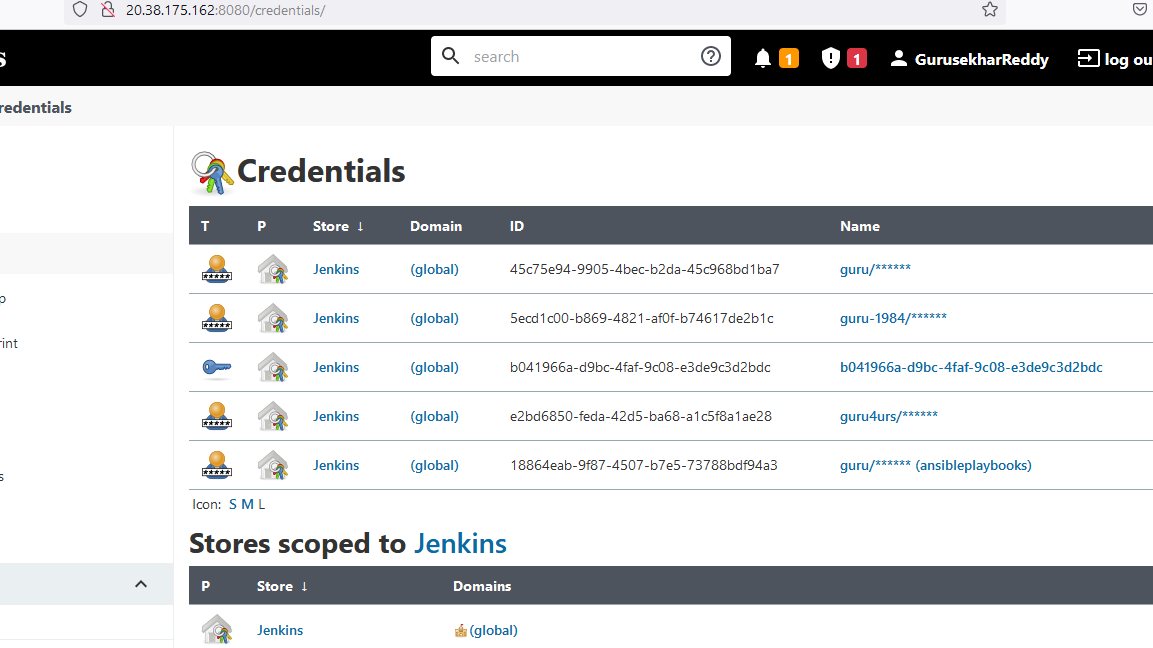


login jenkins with http://aserver.trnglabs.com:8080/login?from=%2F

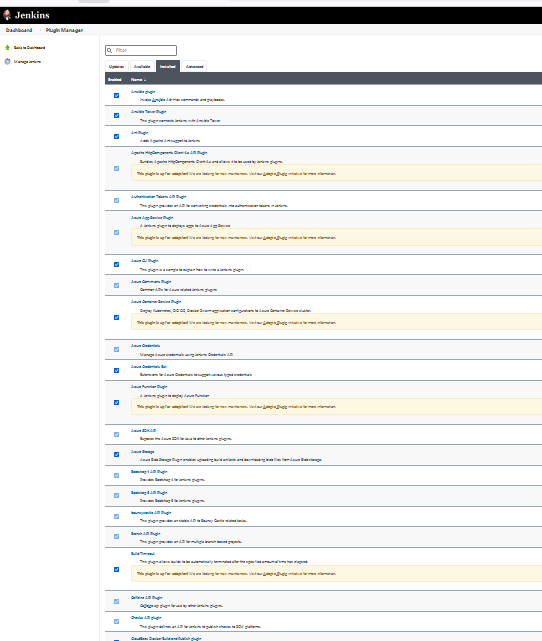
manage jenkin - mange nodes add node1.



credentials added for node2 server , github, dockerhub, azure service principle.



adde all plugins.

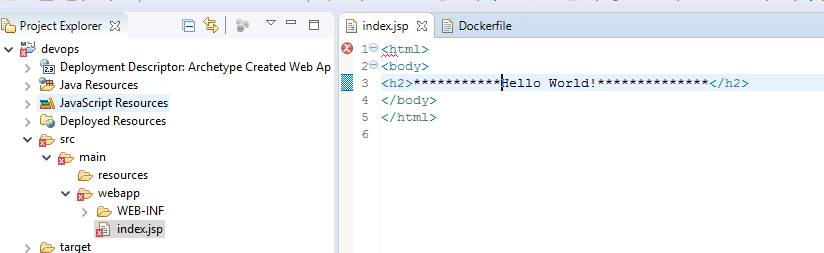


Part2: phase 1

1)create maven project with webarch type webapplication in eclipse

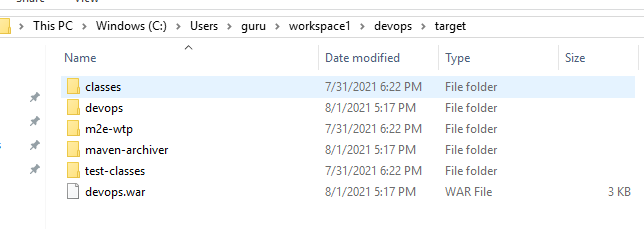
eclipse - new maven project as web app give name as devops

2)modify index.jsp



3)maven install

right click on project run as maven install. new war in.



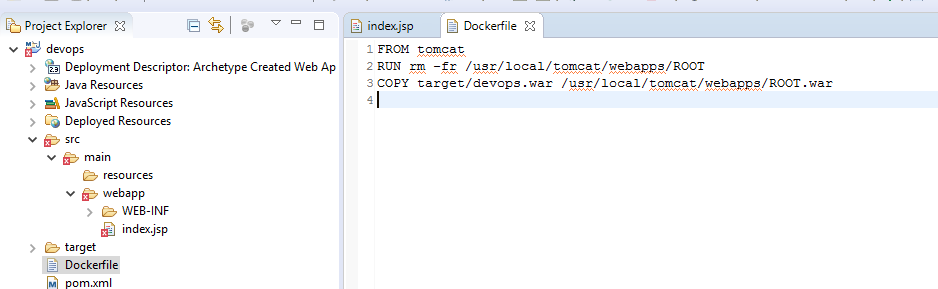
part2 : phase2

1) create docker file under project

right click on project - azure -docker support.

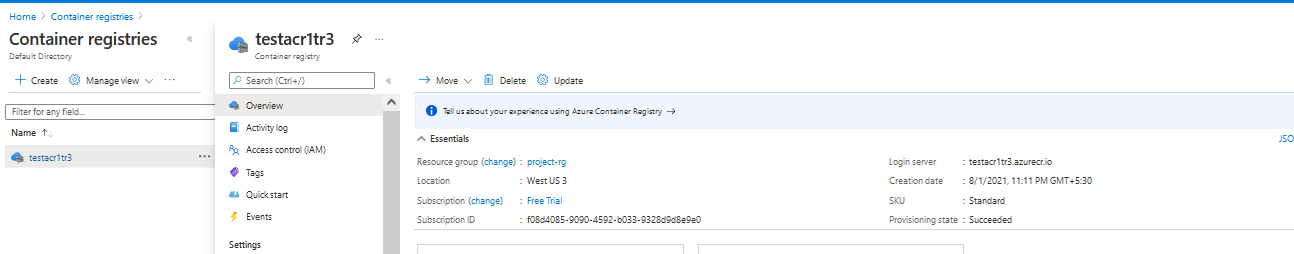
don't see azure then Help - marketplace eclipse - install azure

2)modify docker file.



3)Test the docker file by running docker build and create container.

For this i have used Azure container registry.



az login

go to the directory where workspace war like devops is project name

**PS C:\Users\guru\workspace1> cd devops**

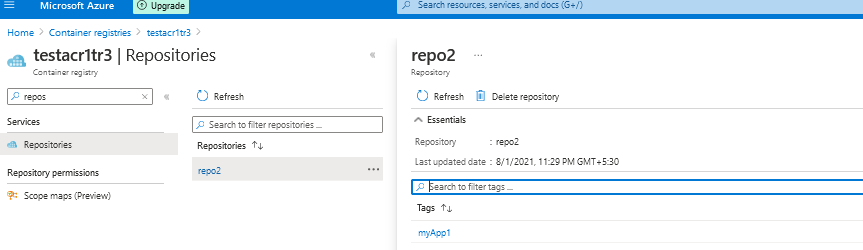
**PS C:\Users\guru\workspace1\devops> az acr build --registry testacr1tr3.azurecr.io --image testacr1tr3.azurecr.io/repo2:myApp1 .**

The login server endpoint suffix '.azurecr.io' is automatically omitted.

Packing source code into tar to upload...

Uploading archived source code from 'C:\Users\guru\AppData\Local\Temp\2\build\_archive\_7bde70b628d84e7e8dfeaa5622a8a0d1.tar.gz'...

Repository added



3)Access the application in container and test it.

on one of linux box,

guru@project-linuxvm-1:~$ docker login testacr1tr3.azurecr.io

Username: testacr1tr3 //continaer reistry- accesskey - enable admin can se this usernmae

Password://password can see in same location

WARNING! Your password will be stored unencrypted in /home/guru/.docker/config.json.

Configure a credential helper to remove this warning. See

https://docs.docker.com/engine/reference/commandline/login/#credentials-store

oject-linuxvm-1:~$ docker pull testacr1tr3.azurecr.io/repo2:myApp1

myApp1: Pulling from repo2

guru@project-linuxvm-1:~$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

testacr1tr3.azurecr.io/repo2 myApp1 98344d5ad8a7 6 minutes ago

guru@project-linuxvm-1:~$ sudo docker run -d -p 8087:8080 --name wor testacr1tr3.azurecr.io/repo2:myApp1

c189ad1a9f1b268c098f80ef65cecad7010ca625ea8245d3162e4674b7d136ae

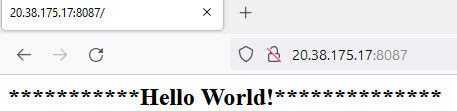
guru@project-linuxvm-1:~$ sudo docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

c189ad1a9f1b testacr1tr3.azurecr.io/repo2:myApp1 "catalina.sh run" 9 seconds ago Up 7 seconds 0.0.0.0:8087->8080/tcp wor

guru@project-linuxvm-1:~$ docker

can access appllicatoin with http://20.38.175.17:8087/



part2: phase3

1)create git repository and copy repo

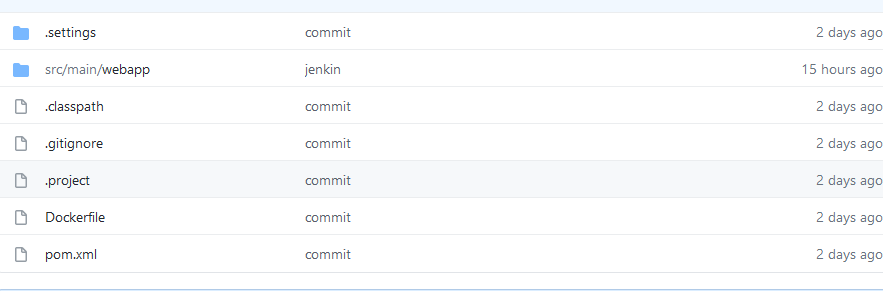
2) eclipse conver app and add in localrepo

ecliipse - local repository

project - team - add- share project

commit and push the repository to github.

3) commit and push to github repositor.



par2: phase4

1)modifty the index.jsp and commit

done

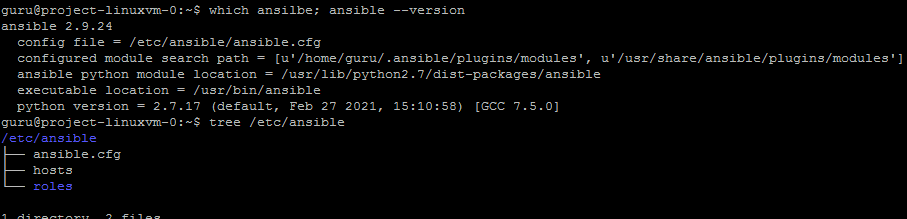
2)check in git hub repo changes there

done.

part2: phase5

1) in build server configur ansible manually

done



2)modify ansible server for hosts as default



3)insttll python pip on ansible sever

sudo apt-get install -y python-pip

pip --version

sudo apt-get install software-properties-common

$ sudo apt-add-repository ppa:ansible/ansible

$ sudo apt-get update

$ sudo apt-get ansible

ansible --version

sudo chown -R guru:guru etc/ansible/

ls -al /etc/ansible/

4) use pip module install azure modules

sudo pip install ansible[azure]

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

5)use the same service principle created for terraform use in jenkins

/home/guru/.jenkins/credentials add it here the 4 keys with out quotes as per shell format.

[default]

subscription\_id=xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

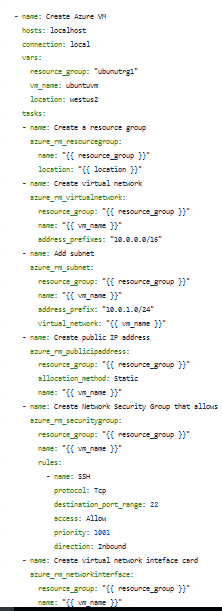
client\_id=xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

secret=xxxxxxxxxxxxxxxxx

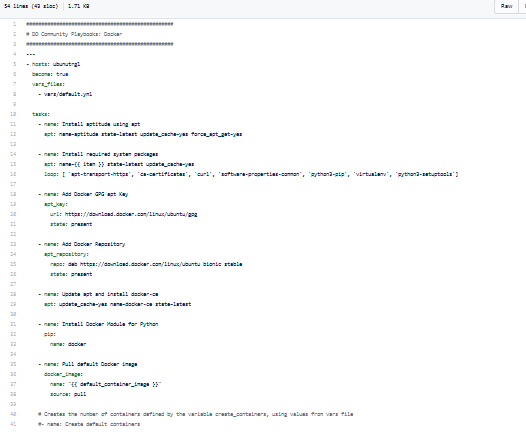
tenant=xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

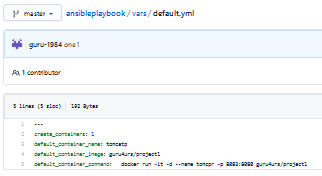


6) create a playbook to create vm1



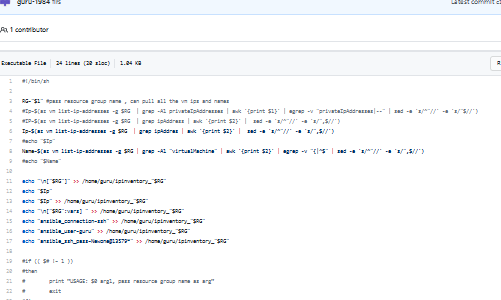
7)update the playbook2 to insall docker engine on vm

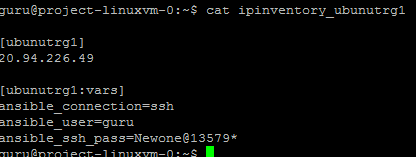




8) create shell to get vmip and update in hosts

i mentioned in script to update the inventory in /home/guru/$(rgname).inventory





part2: phase 6

1) push plyabook1 and playbook2 to git hub

done

2)check the updates in github

done

part2: phase7

1)from the build server run playbook1, playbook2 and vm\_ip.sh scripts.

done

ansilbe-playbook playbook1

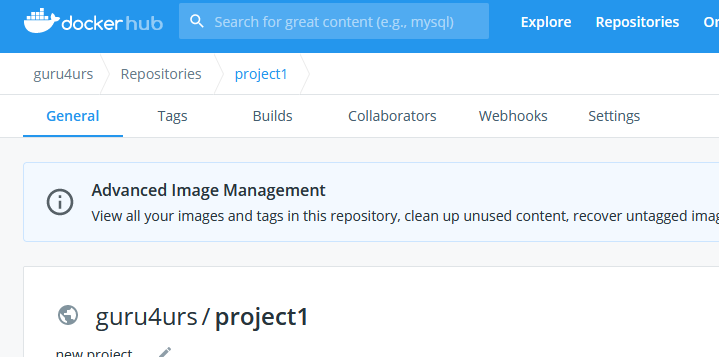
vm\_ip.sh ubunutrg1

ansible playbook playbook2 -i ipinventory\_ubunutrg1

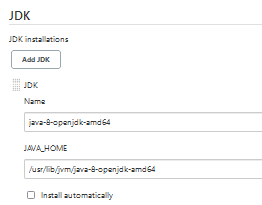
Didn't enable logs cfg so can't paste it here .

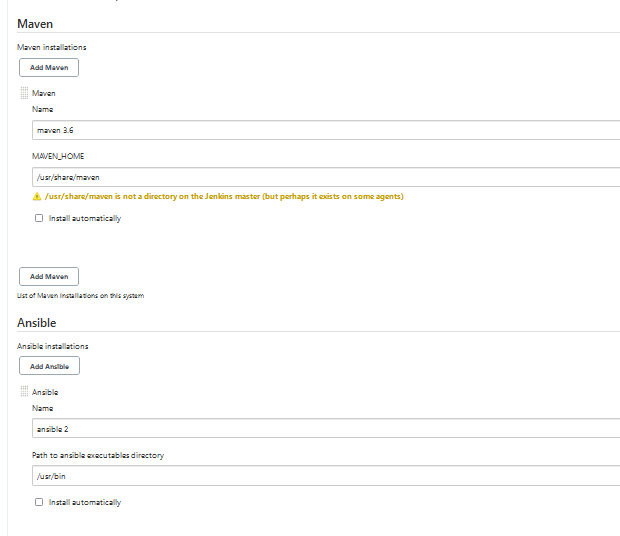
part2: phase8

1)create repo in dockerhub



2)configur global tool configuration jenkins for jdk, docker, maven





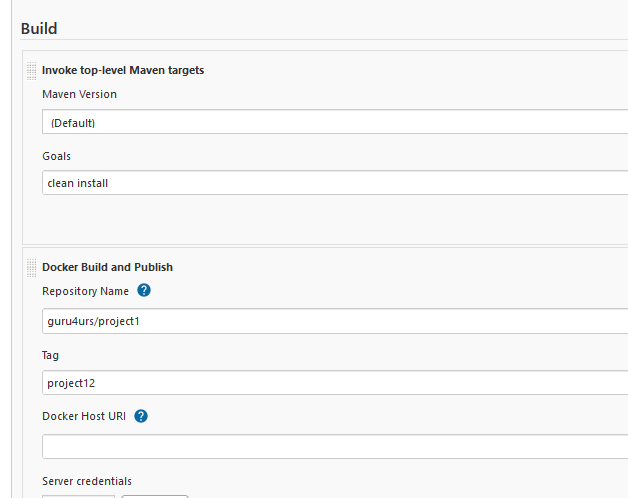
3)configure git credentials , docker , node1 etc.. credentials

4) create freestyle project pipeline1

5)in scm stage pull code from git repo

6) build stage use maven for clean install

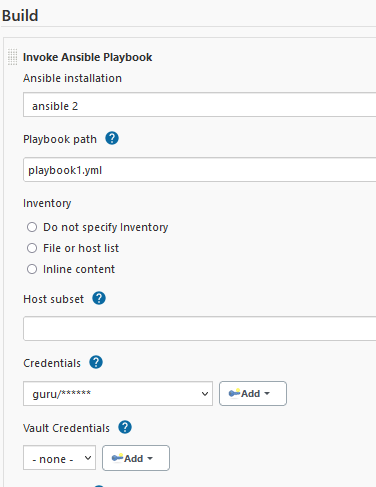
7)build stage step2 use docker to push repository



8)creat pipeline 2 project as free style

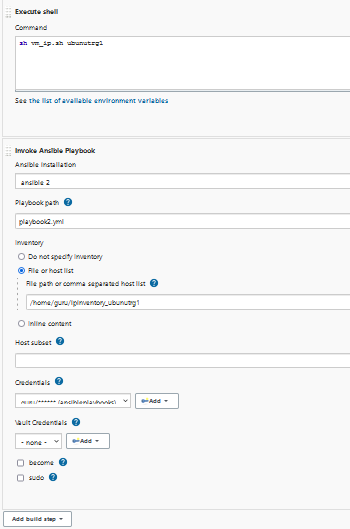
9)build stage pull code from github repo

10) run ansilble playbook



11) build stage step2 call shell script

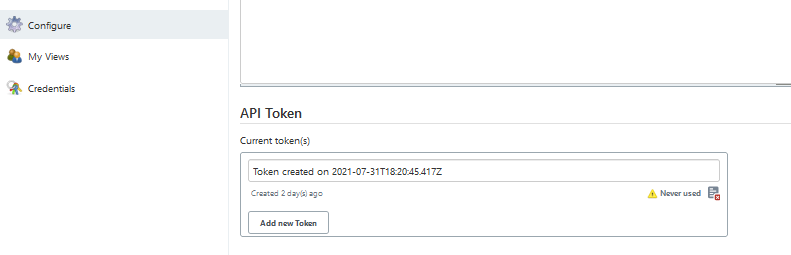
12) build stage step3 run playbook2



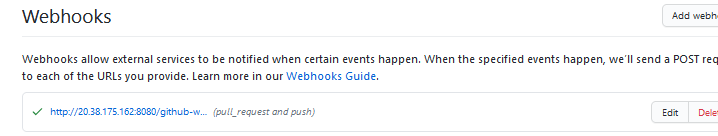
13) post build stage deploy docker container

didn't work this, used plyabook2 for the same.

14)in github repo configure webhook

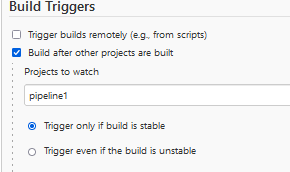


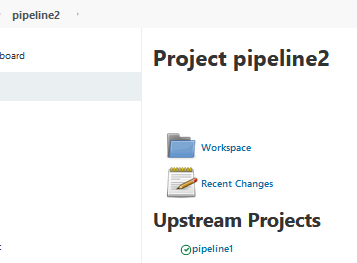
create token use it in webhook

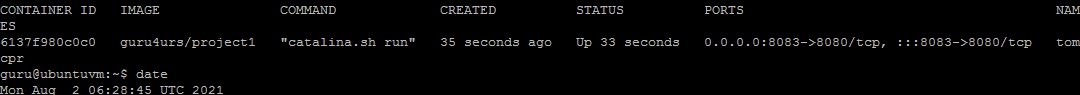


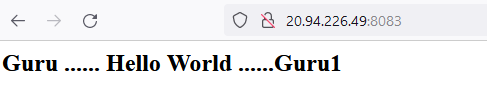


11) for both pipelines configure bujild triggers









9 terraform init

10 terraform validate

11 terraform validate

12 terraform plan

13 terraform apply

14 terraform validate

15 terraform plan

16 terraform apply

PS C:\gur\change> ls

Directory: C:\gur\change

Mode LastWriteTime Length Name

---- ------------- ------ ----

d----- 8/1/2021 3:55 PM .terraform

-a---- 8/1/2021 3:55 PM 2051 .terraform.lock.hcl

-a---- 8/1/2021 4:03 PM 5894 linuxvm.tf

-a---- 8/1/2021 3:42 PM 510 master.sh

-a---- 8/1/2021 3:45 PM 620 node.sh

-a---- 8/1/2021 4:10 PM 23053 terraform.tfstate

-a---- 8/1/2021 4:05 PM 22380 terraform.tfstate.backup

-a---- 8/1/2021 3:03 PM 2979 variables.tf

