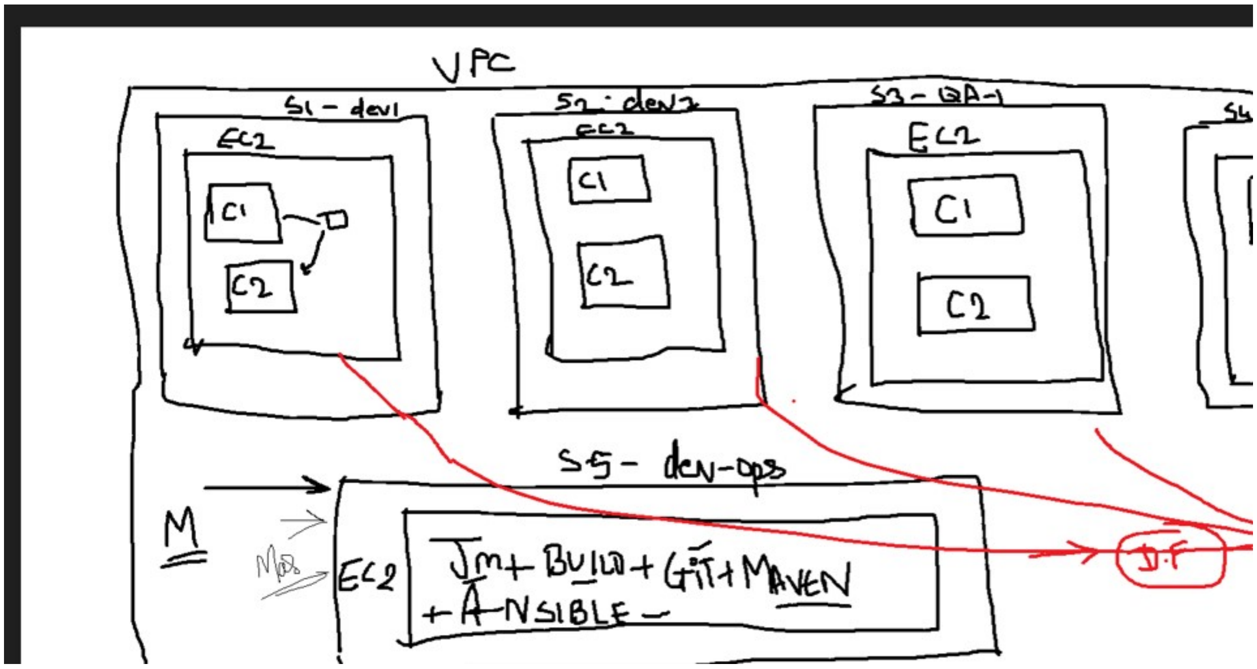
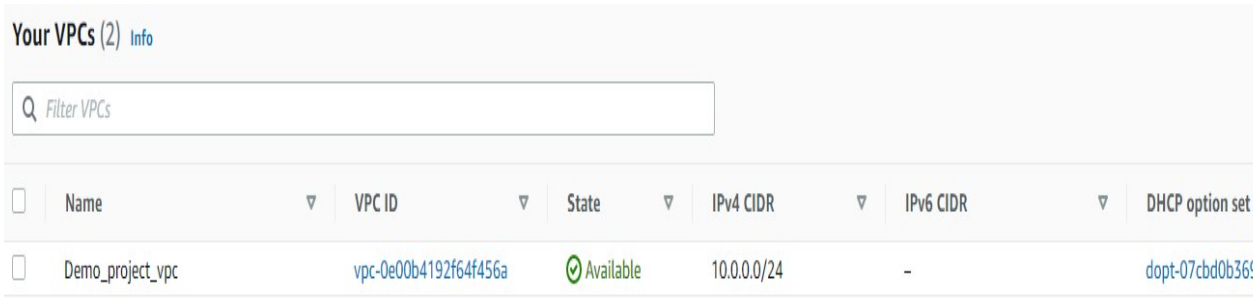
MAIN_PROJECT_USING_ANSIBLE_JENKINS_DOCKER

PROJECT: Need to build Game of life on Master and deploy it on 2container on various env such as Dev1,Dev2,QA1,QA2.



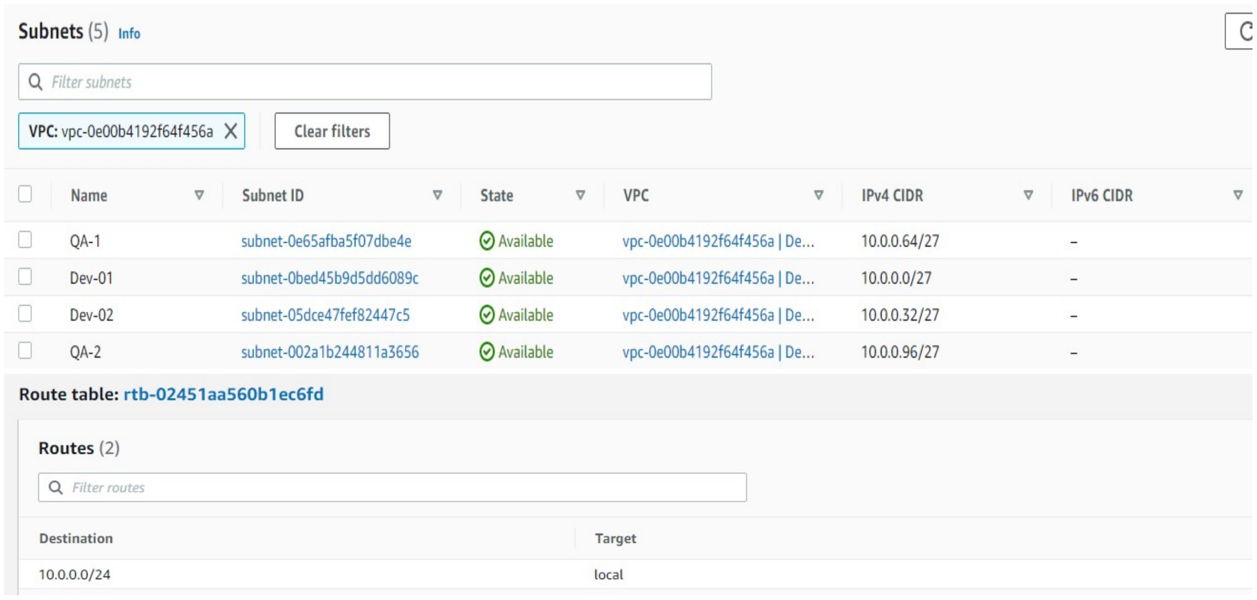
STEP1) CREATE ONE VPC
NAME IT AS DEMO VPC



Your VPCs (2) Info

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set
Demo_project_vpc	vpc-0e00b4192f64f456a	Available	10.0.0.0/24	-	dopt-07cbd0b36

STEP2) CREATE 5 SUBNET (DEV1, DEV2, QA1, QA2 AND DEV_OPS)
ATTACH IGW TO THE SUBNET


Subnets (5) Info

Filter subnets

VPC: vpc-0e00b4192f64f456a Clear filters

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
QA-1	subnet-0e65afba5f07dbe4e	Available	vpc-0e00b4192f64f456a | De...	10.0.0.64/27	–
Dev-01	subnet-0bed45b9d5dd6089c	Available	vpc-0e00b4192f64f456a | De...	10.0.0.0/27	–
Dev-02	subnet-05dce47fef82447c5	Available	vpc-0e00b4192f64f456a | De...	10.0.0.32/27	–
QA-2	subnet-002a1b244811a3656	Available	vpc-0e00b4192f64f456a | De...	10.0.0.96/27	–

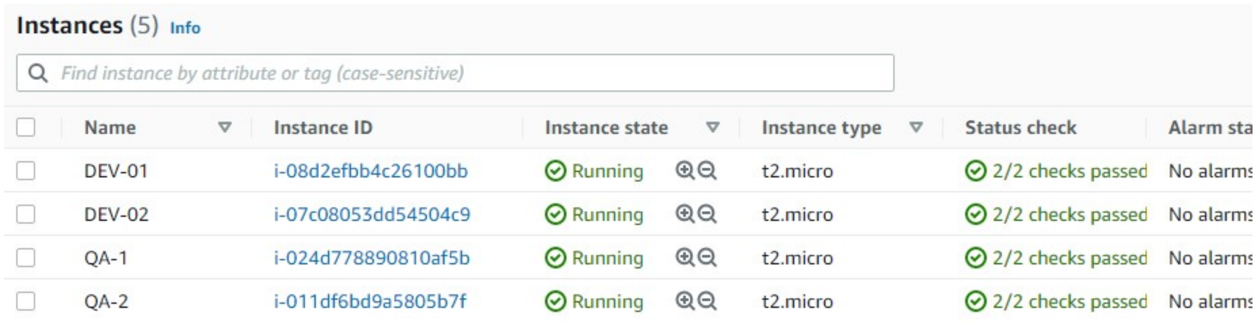
Route table: rtb-02451aa560b1ec6fd

Routes (2)

Filter routes

Destination	Target
10.0.0.0/24	local

STEP3) LAUNCH ONE MACHINE IN EACH SUBNET:
(NAME THE MACHINE AS PER THE SUBNET NAME)


Instances (5) Info

Find instance by attribute or tag (case-sensitive)

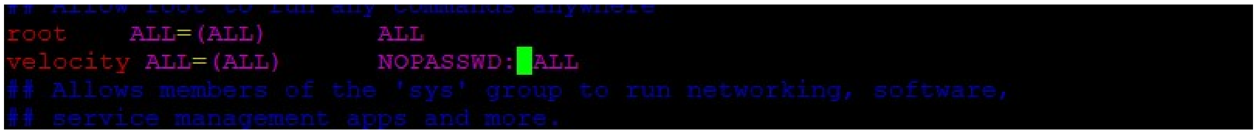
Name	Instance ID	Instance state	Instance type	Status check	Alarm sta
DEV-01	i-08d2efbb4c26100bb	Running	t2.micro	2/2 checks passed	No alarms
DEV-02	i-07c08053dd54504c9	Running	t2.micro	2/2 checks passed	No alarms
QA-1	i-024d778890810af5b	Running	t2.micro	2/2 checks passed	No alarms
QA-2	i-011df6bd9a5805b7f	Running	t2.micro	2/2 checks passed	No alarms

STEP4) DEV_OPS MACHINE WILL BE YOUR MASTER MACHINE IN WHICH WE WILL NEED JENKINS, TOMCAT, ANSIBLE_PLAYBOOK, MAVEN. AS WE ARE GOING TO BUILD THE JOB ON MASTER.

STEP4A)--- CREATE A SERVICE USER NAME AS VELOCITY (IN ALL THE MACHINE DEV1, DEV2, QA1, QA2 AND DEV_OPS)
COMMAND: useradd velocity

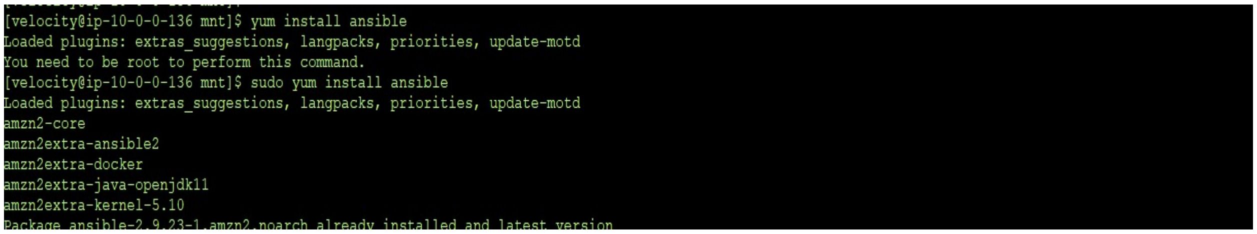
ASSIGN PASSWD TO USER
COMMAND: passwd velocity
(ENTER THE PASSWORD)

NOW GIVE THE SUDO PERMISSON TO VELOCITY USER ON EACH MACHINE (DEV1, DEV2, QA1, QA2 AND DEV_OPS)
COMMAND: visudo


```
root      ALL=(ALL)       ALL
velocity ALL=(ALL)       NOPASSWD: ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
```

NOW AS WE ARE GOING TO INSTALL THE REQ DEPENDENCY USING ANSIBLE WE WILL NEED TO MAKE SOME CHANGES ON EACH MACHINE (DEV1, DEV2, QA1, QA2 AND DEV_OPS)

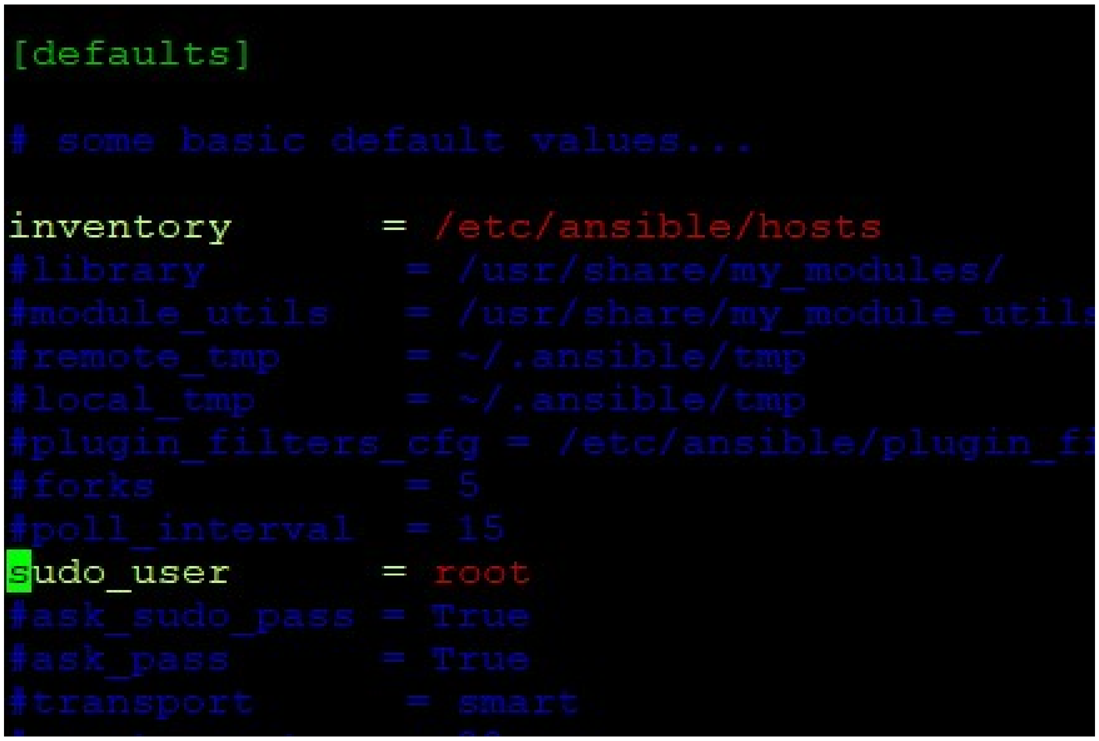
ON MASTER MACHINE (DEV_OPS) INSTALL ANSIBLE
COMMAND: sudo yum install ansible -y


```
[velocity@ip-10-0-0-136 mnt]$ yum install ansible
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
You need to be root to perform this command.
[velocity@ip-10-0-0-136 mnt]$ sudo yum install ansible
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-ansible2
amzn2extra-docker
amzn2extra-java-openjdk11
amzn2extra-kernel-5.10
Package ansible-2.9.23-1.amzn2.noarch already installed and latest version
```

FOR THE MASTER MACHINE (FOR ANSIBLE) WE NEED TO EDIT THE INVENTORY FILE CALLED AS ANSIBLE-CONF IN THE LOCATION /etc/ansible/ansible.cfg

COMMAND: sudo vi /etc/ansible/ansible.cfg

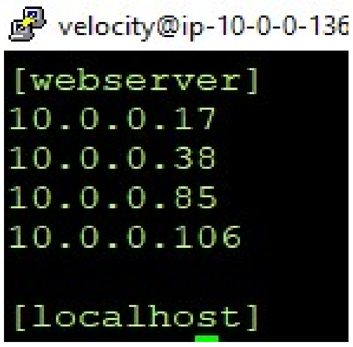
WE NEED TO ENABLE THE INVENTORY AND SUDO USER AS SHOWN BELOW


```
[defaults]

# some basic default values...

inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_fi
#forks          = 5
#poll_interval  = 15
sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
#transport      = smart
```

NOW WE NEED TO EDIT THE HOST FILE FOR ANSIBLE ON MASTER (DEV_OPS)

COMMAND: sudo vi /etc/ansible/hosts

WE NEED TO PROVIDE THE SERVERNAME AND PRIVATE IP ADD OF OTHER HOSTS (DEV1, DEV2, QA1, QA2)


```
velocity@ip-10-0-0-136
[webserver]
10.0.0.17
10.0.0.38
10.0.0.85
10.0.0.106

[localhost]
```

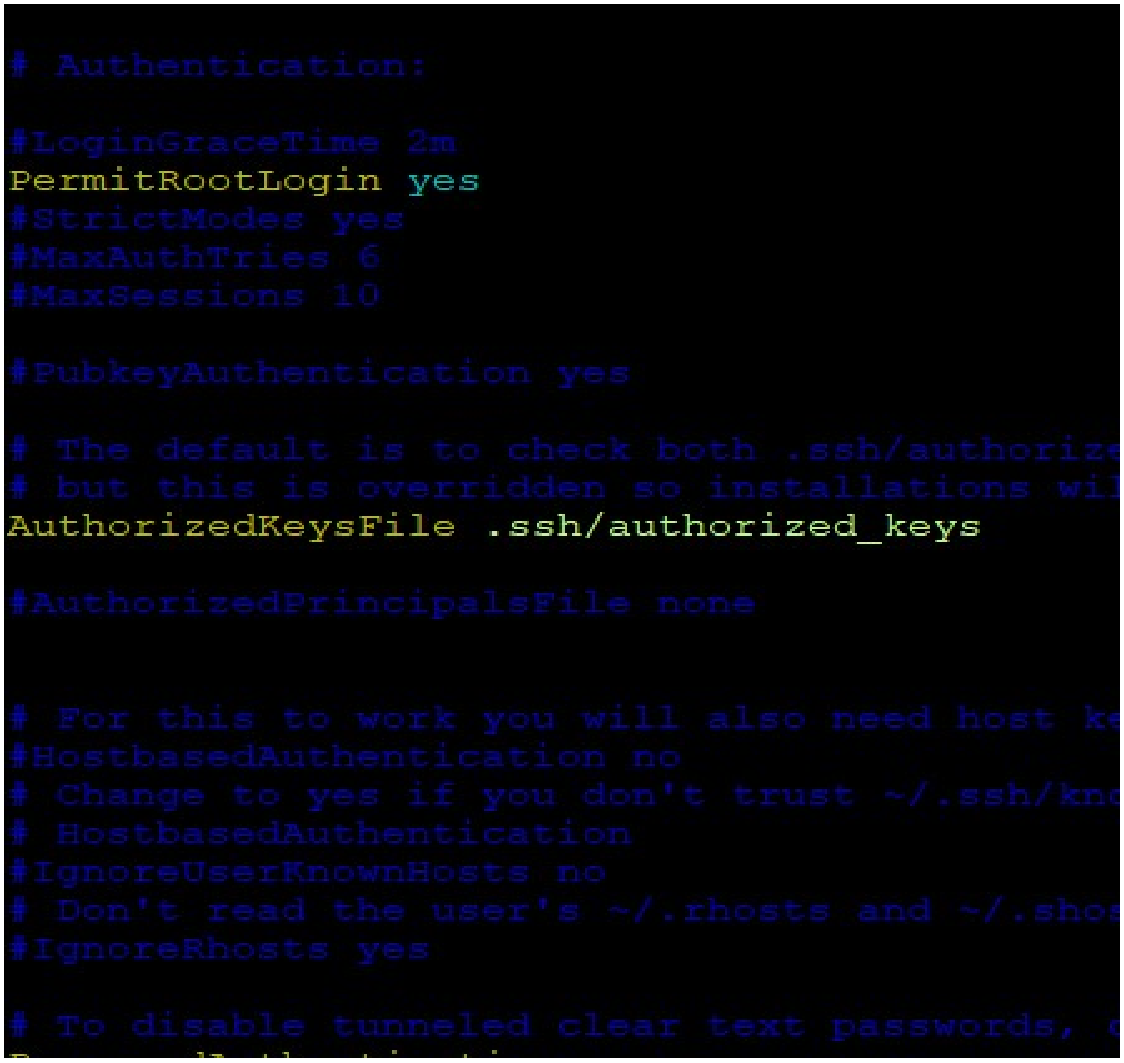
NOW WE HAVE TO MAKE CHANGES IN THE SSHD FILE SO THAT OUR SERVICE USER (VELOCITY) CAN MAKE SSH CONNECTION WITH OTHER HOSTS WITHOUT USING THE PASSWORD

FOR THAT WE NEED TO MAKE CHANGES IN THE SSHD FILE
COMMAND: sudo vi /etc/ssh/sshd_config

MAKE SURE YOU UNCOMMENT THE FOLLOWING

PERMITROTTLOGIN YES
PASSWORDAUTHENTICATION YES

ADD # TO COMMENT
PASSWORDAUTHENTICATION NO



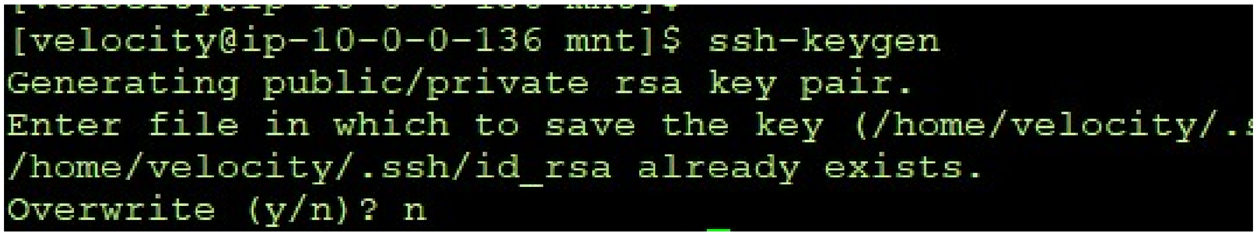
```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorize
# but this is overridden so installations wil
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none


# For this to work you will also need host ke
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/kno
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shos
#IgnoreRhosts yes

# To disable tunneled clear text passwords, c
```

THE ABOVE STEP HAS TO BE EXECUTED AT ALL THE MACHINE (DEV1, DEV2, QA1, QA2 AND DEV_OPS)
MAKE SURE YOU RESTART SSHD SERVICES

COMMAND: service sshd restart

NOW WE HAVE TO GENERATE SSH KEY ON THE MASTER MACHINE SO THAT WE CAN CONNECT WITH HOST MACHINE USING THE SERVICE_USER (VELOCITY)

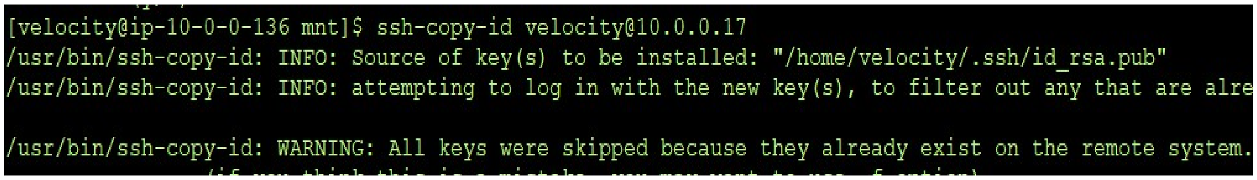
COMMAND: ssh-keygen



```
[velocity@ip-10-0-0-136 mnt]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/velocity/.s
/home/velocity/.ssh/id_rsa already exists.
Overwrite (y/n)? n
```

NOW AS YOUR KEY IS GENERATED WE NEED TO COPY THE KEY TO ALL THE MACHINE USING THE FOLLOWING COMMAND:

COMMAND: ssh-copy-id velocity@10.0.0.17
ssh-copy-id velocity@10.0.0.38
ssh-copy-id velocity@10.0.0.85
ssh-copy-id velocity@10.0.0.106



```
[velocity@ip-10-0-0-136 mnt]$ ssh-copy-id velocity@10.0.0.17
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/velocity/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alre

/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
```

AS WE HAVE ALREADY COPIED SO IT WON'T PROMT FOR THE PASSWORD. THE VERY FIRST TIME YOU COPY THE KEY IT SHOULD ASK YOU FOR PASSWORD OF VELOCITY USER.
ONCE YOU ENTER THE PASSWORD THE NEXT TIME YOU DO SSH IT WON'T ASK YOU FOR KEY.

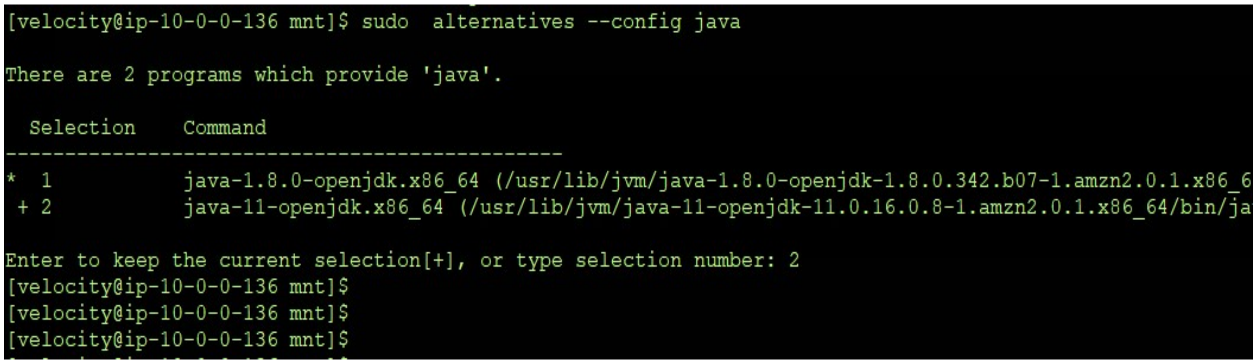
STEP5) ON DEV_OPS MACHINE (MASTER MACHINE) WE WILL INSTALL JAVA.

COMMAND: sudo yum install java-1.8.0_openjdk-devel.x86_64 -y
sudo amazon-linux-extras install java-openjdk11=latest –y

MAKE SURE YOU SELECT THE JAVA 11 AS THE TOP PRORIOTY AS WE ARE GOING TO BUILD JENKINS ON JAVA 11. TO DO SO WE NEED TO USER ALTERNATIVES COMMAND

COMMAND: sudo alternatives –config java

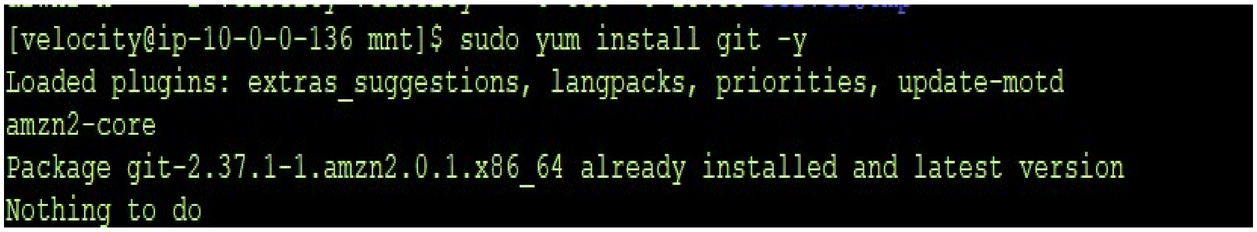
MAKE SURE YOU SELECT THE JAVA 11 OPTION.



```
[velocity@ip-10-0-0-136 mnt]$ sudo  alternatives --config java

There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*  1           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.342.b07-1.amzn2.0.1.x86_6
 + 2           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-openjdk-11.0.16.0.8-1.amzn2.0.1.x86_64/bin/ja

Enter to keep the current selection[+], or type selection number: 2
[velocity@ip-10-0-0-136 mnt]$
[velocity@ip-10-0-0-136 mnt]$
[velocity@ip-10-0-0-136 mnt]$
```

NOW WE NEED TO INSTALL GIT:

COMMAND: yum install git -y

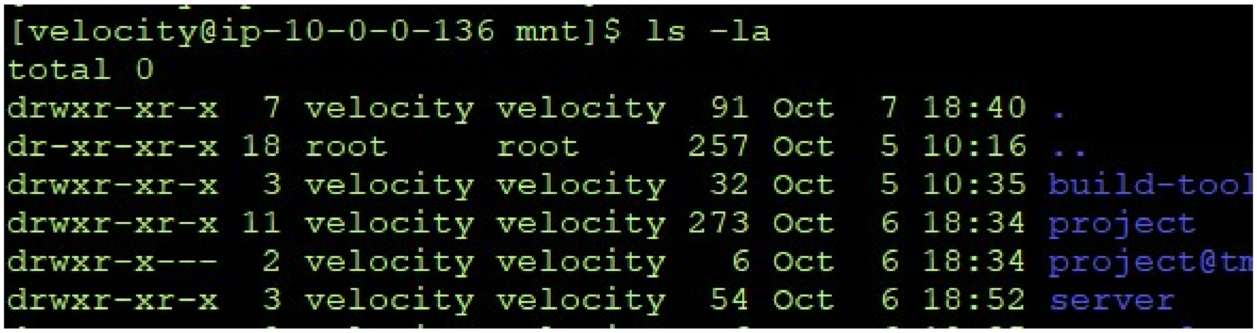


```
[velocity@ip-10-0-0-136 mnt]$ sudo yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Package git-2.37.1-1.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
```

STEP6) NOW WE NEED TO CREATE VARIOUS MOUNT POINTS FOR VARIOUS PACKAGES

BUILD-TOOLS-----IT WILL BE OUR MAVEN HOME DIR
SERVER---- HOME DIR FOR OUR TOMCAT SERVER
PROJECT—WE WILL BUILD OUR PROJECT HERE (ALSO WE WILL CLONE IT HERE USING GIT)

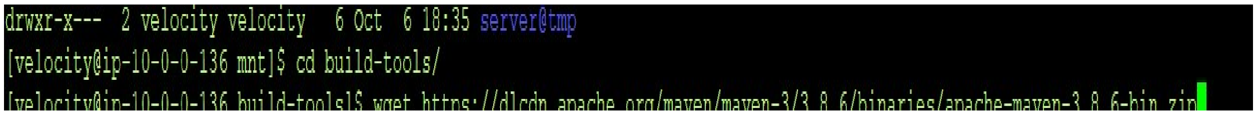
```
COMMAND:      cd
              cd /mnt
              mkdir build-tools
              mkdir projects
              mkdir server
```



```
[velocity@ip-10-0-0-136 mnt]$ ls -la
total 0
drwxr-xr-x  7 velocity velocity  91 Oct  7 18:40 .
dr-xr-xr-x 18 root     root     257 Oct  5 10:16 ..
drwxr-xr-x  3 velocity velocity  32 Oct  5 10:35 build-tool
drwxr-xr-x 11 velocity velocity 273 Oct  6 18:34 project
drwxr-x---  2 velocity velocity   6 Oct  6 18:34 project@tm
drwxr-xr-x  3 velocity velocity  54 Oct  6 18:52 server
```

NOW TO INSTALL MAVEN FIRST CD TO BUILD-TOOLS
COMMAND: cd /mnt/build-tools
GO ON THE MAVEN HOME PAGE AND COPY THE .ZIP LINK TO DOWNLOAD MAVEN

COMMAND: https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.zip



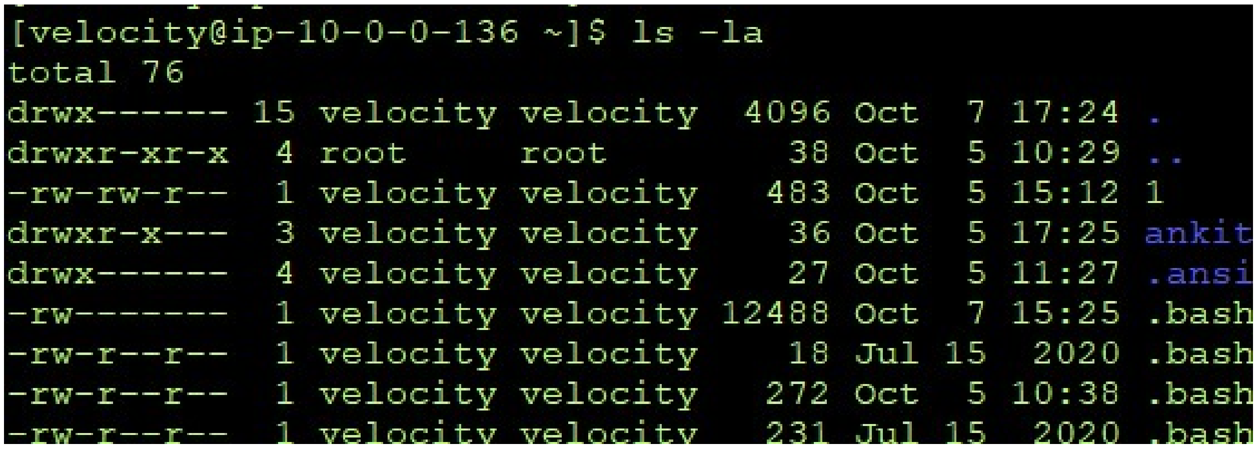
```
drwxr-x---  2 velocity velocity   6 Oct  6 18:35 server@tmp
[velocity@ip-10-0-0-136 mnt]$ cd build-tools/
[velocity@ip-10-0-0-136 build-tools]$ wget https://dlcdn.apache.org/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.zip
```

ONCE YOU HAVE DOWNLOADED IT INTO YOUR REQ DIR WE NEED TO UNZIP IT USING UNZIP
COMMAND : unzip apache-maven-3.8.6

NOW WE NEED TO SET THE ENV VARIABLE FOR MAVEN TO WORK

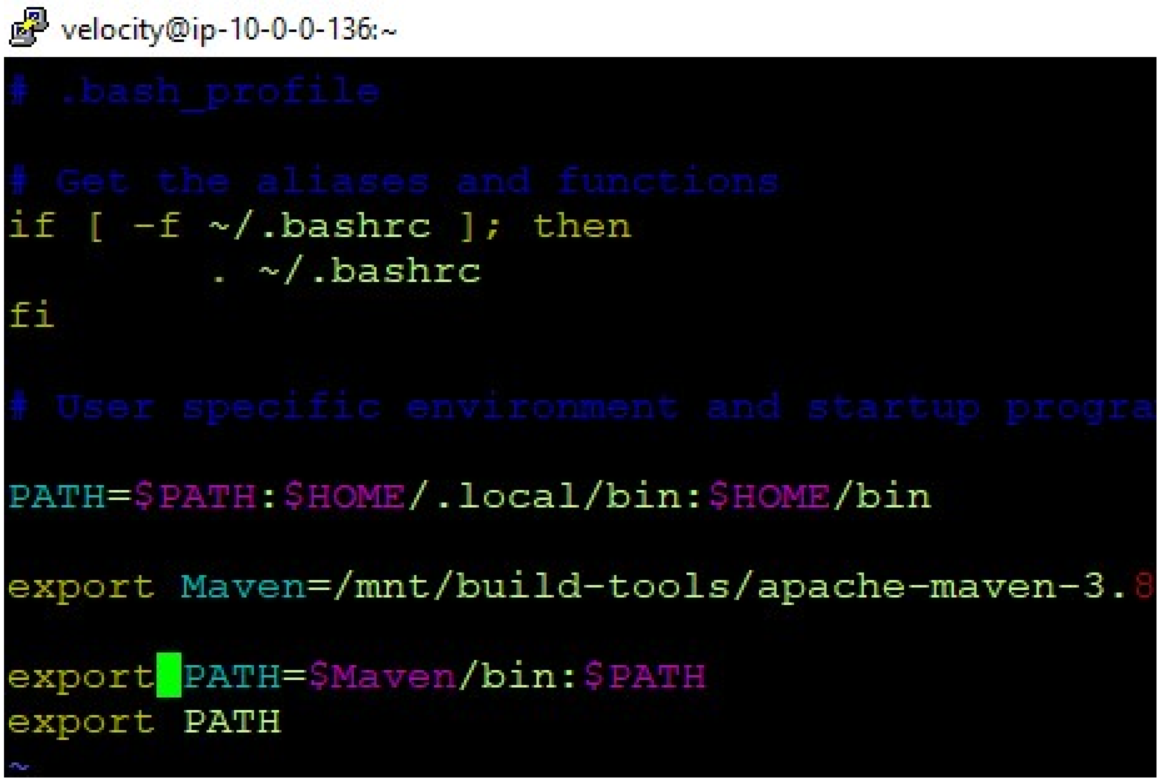
TO DO SO EDIT IT .BASH_PROFILE FILE.

COMMAND:
```
cd
ls  -la
sudo vi .bash_profile
```



```
[velocity@ip-10-0-0-136 ~]$ ls -la
total 76
drwx------ 15 velocity velocity  4096 Oct  7 17:24 .
drwxr-xr-x  4 root     root        38 Oct  5 10:29 ..
-rw-rw-r--  1 velocity velocity   483 Oct  5 15:12 1
drwxr-x---  3 velocity velocity    36 Oct  5 17:25 ankit
drwx------  4 velocity velocity    27 Oct  5 11:27 .ansi
-rw-------  1 velocity velocity 12488 Oct  7 15:25 .bash
-rw-r--r--  1 velocity velocity    18 Jul 15  2020 .bash
-rw-r--r--  1 velocity velocity   272 Oct  5 10:38 .bash
-rw-r--r--  1 velocity velocity   231 Jul 15  2020 .bash
```

ONCE YOU GOT INTO THE FILE WE NEED TO SET MAVEN ENV VAR PATH
REFER TO SCREENSHOT BELOW AND MAKE THE CHANGES AS SHOWN



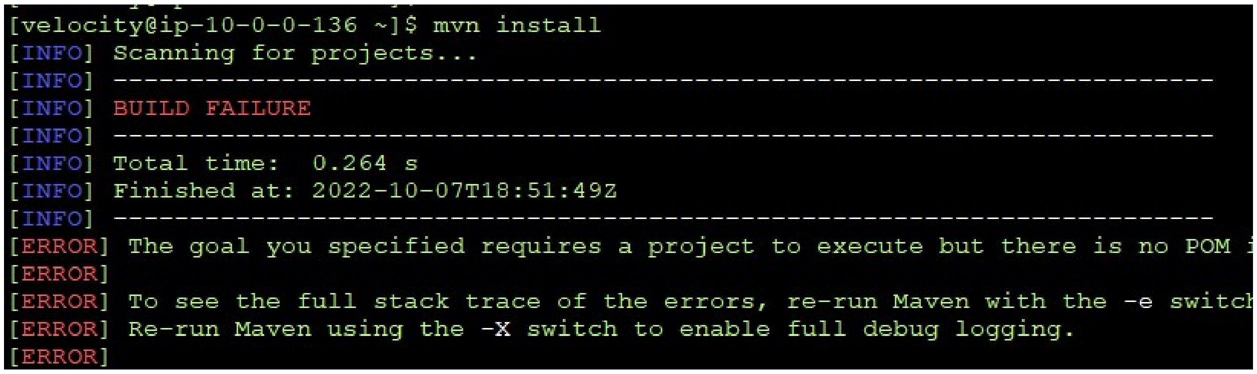
velocity@ip-10-0-0-136:~

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup progra

PATH=$PATH:$HOME/.local/bin:$HOME/bin

export Maven=/mnt/build-tools/apache-maven-3.8

export PATH=$Maven/bin:$PATH
export PATH
~
```

LOUTOUT AND LOGIN BACK FOR CHANGES TO WORK

YOU CAN CHECKIF YOUR ENV VAR IS WORKING BY RUNING THE MVN COMMAND ANYWHERE.

YOU SHOULD SEE SOME ERRORS AS WE DO NOT HAVE THE POM FILE AS OF NOW



```
[velocity@ip-10-0-0-136 ~]$ mvn install
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.264 s
[INFO] Finished at: 2022-10-07T18:51:49Z
[INFO] ------------------------------------------------------------------------
[ERROR] The goal you specified requires a project to execute but there is no POM i
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switc
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
```

NOW TO INSTALL TOMCAT FIRST CD TO SERVER

COMMAND: cd /mnt/server

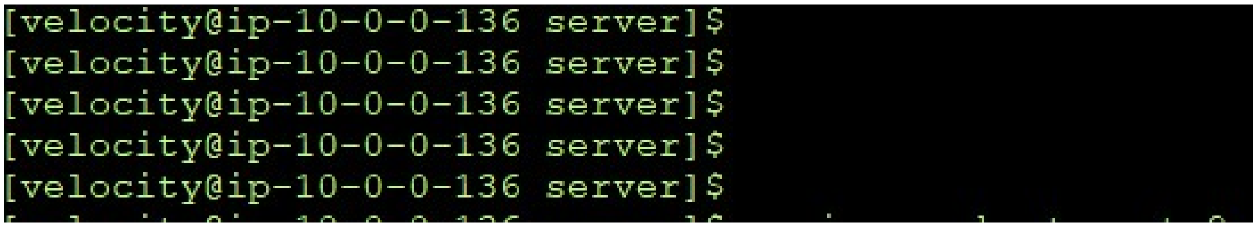
GO ON THE APACHE TOMCAT HOME PAGE AND COPY THE APACHE TOMCAT 9.0 .ZIP LINK TO DOWNLOAD TOMCAT SERVER

COMMAND: https://downloads.apache.org/tomcat/tomcat-9/v9.0.68/bin/apache-tomcat-9.0.68.zip.asc

```
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
```

UNZIP IT USING THE UNZIP COMMAND

COMMAND: unzip



```
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
[velocity@ip-10-0-0-136 server]$
```

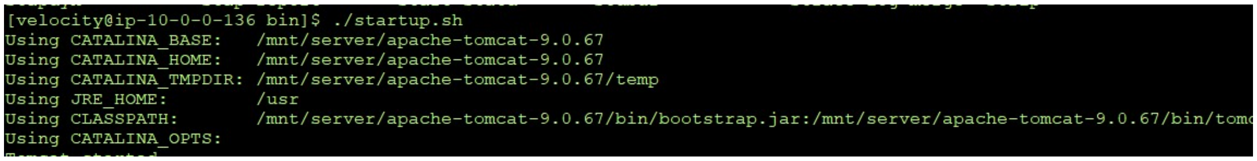
NOW AFYER UNZIP WE NEED TO GIVE (777 PERMISSION TO THE ENTIRE TOMCAT DIR)

COMMAND: sudo chmod –R 777 apache-tomcat-9.0.67.zip

NOW WE NEED TO START THE TOMCAT SERVICES

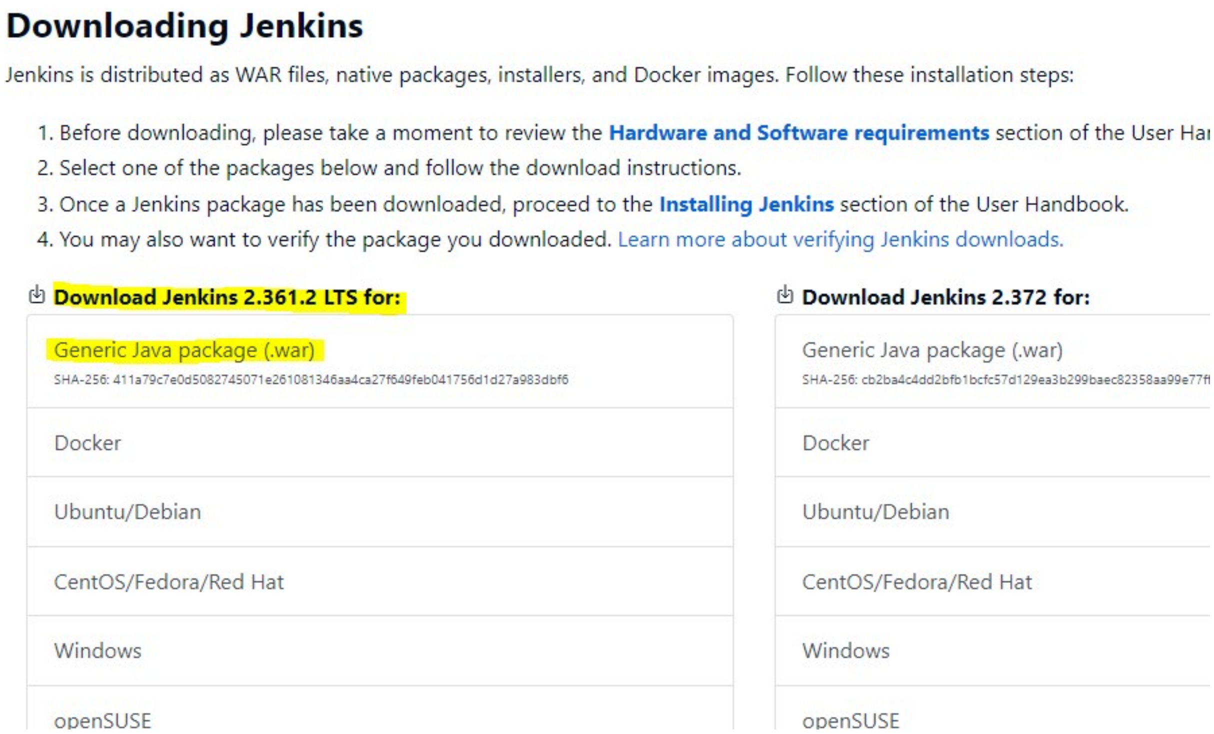
FOR DOING SO WE NEED TO GO INTO /apache-tomcat-9.0.67/bin

HERE WE NEED TO START THE TOMCAT SERVICES BY RUNNING FOLLWING COMMAND
COMMAND: ./startup.sh



```
[velocity@ip-10-0-0-136 bin]$ ./startup.sh
Using CATALINA_BASE:   /mnt/server/apache-tomcat-9.0.67
Using CATALINA_HOME:   /mnt/server/apache-tomcat-9.0.67
Using CATALINA_TMPDIR: /mnt/server/apache-tomcat-9.0.67/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /mnt/server/apache-tomcat-9.0.67/bin/bootstrap.jar:/mnt/server/apache-tomcat-9.0.67/bin/tomc
Using CATALINA_OPTS:
```

NOW THAT YOUR TOMCAT IS UP AND RUNNING WE NEED TO INSTALL JENKINS ONTO IT
FOR THAT WE NEED TO VISIT JENKINS OFFICIAL SITE AND COPY THE .WAR JENKINS TO /apache-tomcat-9.0.67/webapps/



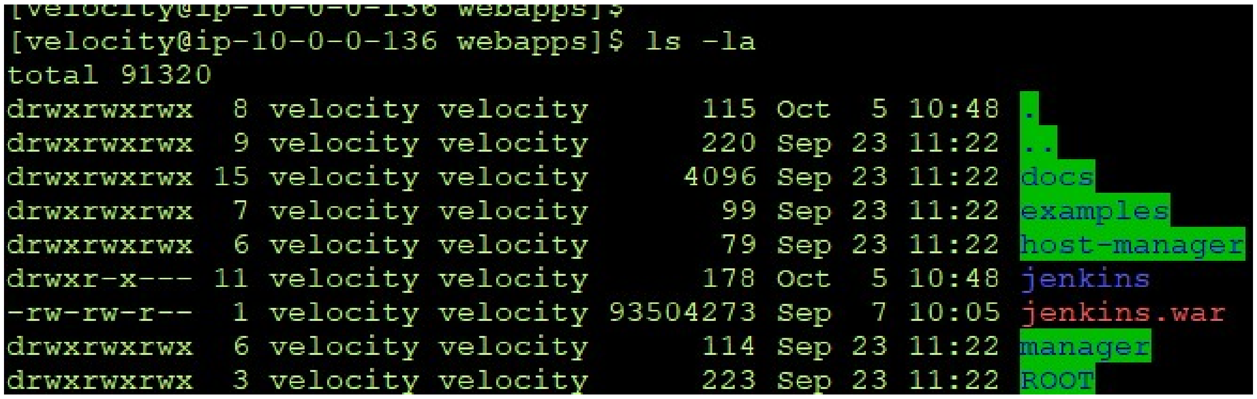
Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the Hardware and Software requirements section of the User Har
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the Installing Jenkins section of the User Handbook.
4. You may also want to verify the package you downloaded. Learn more about verifying Jenkins downloads.

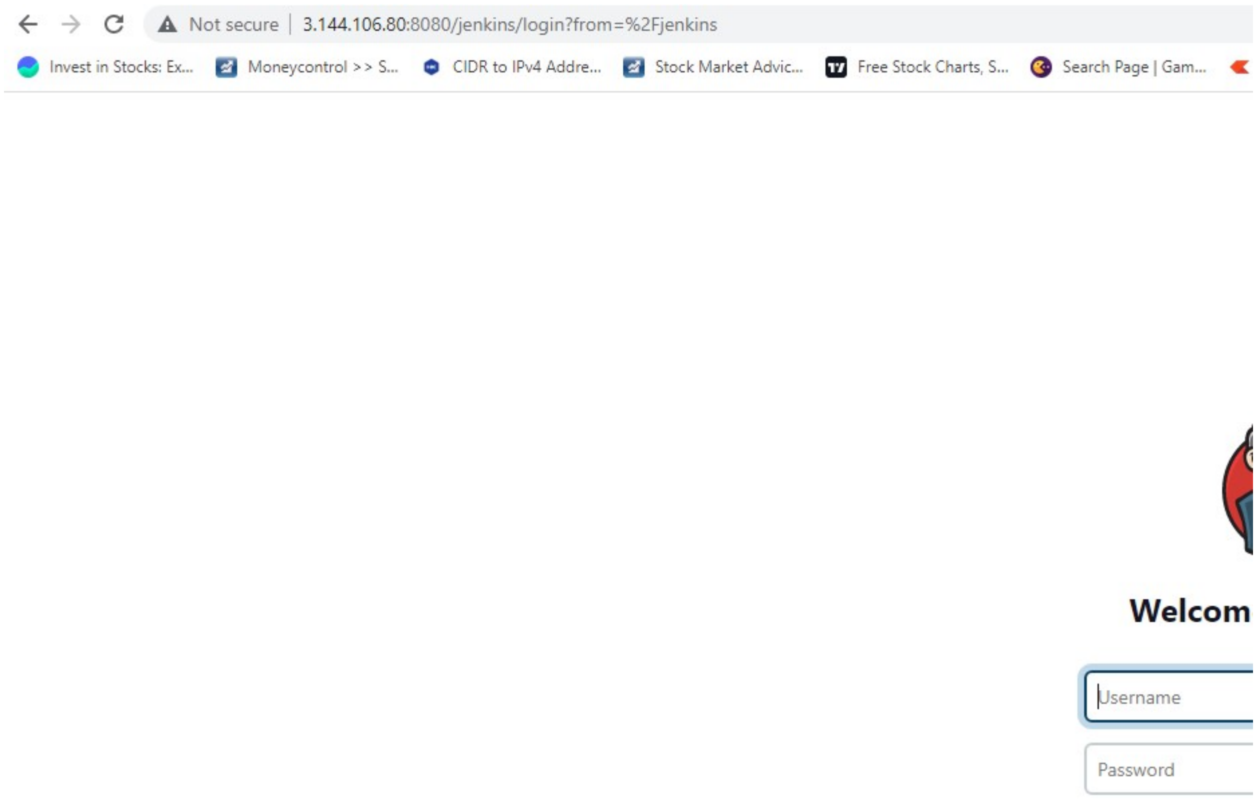
Download Jenkins 2.361.2 LTS for:	Download Jenkins 2.372 for:
Generic Java package (.war) SHA-256: 411a79c7e0d5082745071e261081346aa4ca27f649feb041756d1d27a983dbf6	Generic Java package (.war) SHA-256: cb2ba4c4dd2bfb1bcfc57d129ea3b299baec82358aa99e77ff
Docker	Docker
Ubuntu/Debian	Ubuntu/Debian
CentOS/Fedora/Red Hat	CentOS/Fedora/Red Hat
Windows	Windows
openSUSE	openSUSE

COPY THE URL AND WGET INTO THE WEBAPPS PATH



```
[velocity@ip-10-0-0-136 webapps]$ ls -la
total 91320
drwxrwxrwx  8 velocity velocity      115 Oct  5 10:48 .
drwxrwxrwx  9 velocity velocity      220 Sep 23 11:22 ..
drwxrwxrwx 15 velocity velocity     4096 Sep 23 11:22 docs
drwxrwxrwx  7 velocity velocity       99 Sep 23 11:22 examples
drwxrwxrwx  6 velocity velocity       79 Sep 23 11:22 host-manager
drwxr-x--- 11 velocity velocity      178 Oct  5 10:48 jenkins
-rw-rw-r--  1 velocity velocity 93504273 Sep  7 10:05 jenkins.war
drwxrwxrwx  6 velocity velocity      114 Sep 23 11:22 manager
drwxrwxrwx  3 velocity velocity      223 Sep 23 11:22 ROOT
```

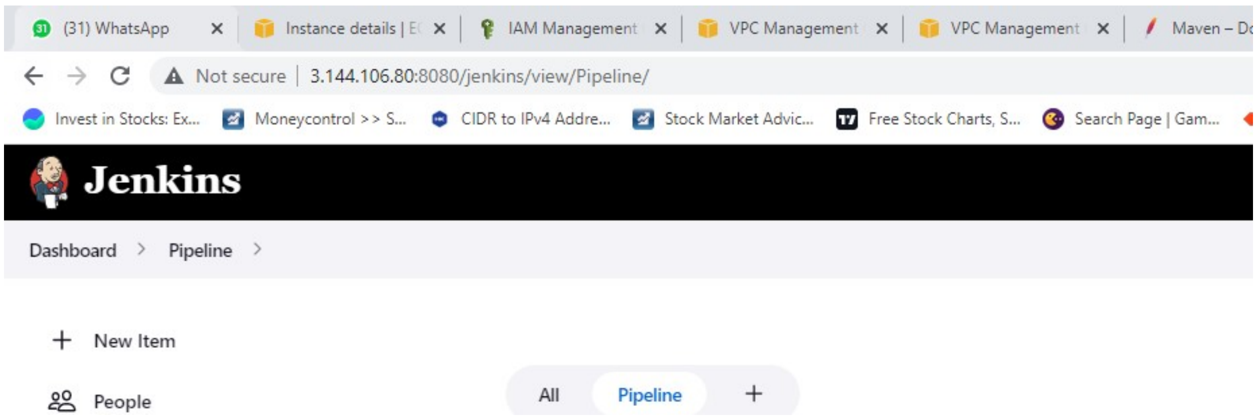
NOW AS TOMCAT IS ALREADY UP AND RUNNING, WE NEED TO JUST GO THE THE INSTANCE (MASTER-DEV_OPS) PUBLIC IP AND ACCESS THE FOLLOWING URL TO INSTALL THE JENKINS



3.144.106.80:8080/jenkins/login?from=%2Fjenkins

MALE SURE TO OPEN THE PORTS ON SECURITY GROUP.
NOW WE HAVE ALREADY INSTALLED AND CONFIGURE JENKINS.

FOR THE FIRST TIME IT WILL ASK YOU TO PROVIDE A PASSWORD FROM YOUR MACHINE AND THEN WILL ASK YOU TO ASIGN USER NAME AND ORTHER DETAILS.
THEN IT WILL ASK YOU FOR THE PLUGIN.
ONCE YOUR ARE DONE WITH IT YOU WILL BE ON YOUR
JENINKS HOMEPAGE



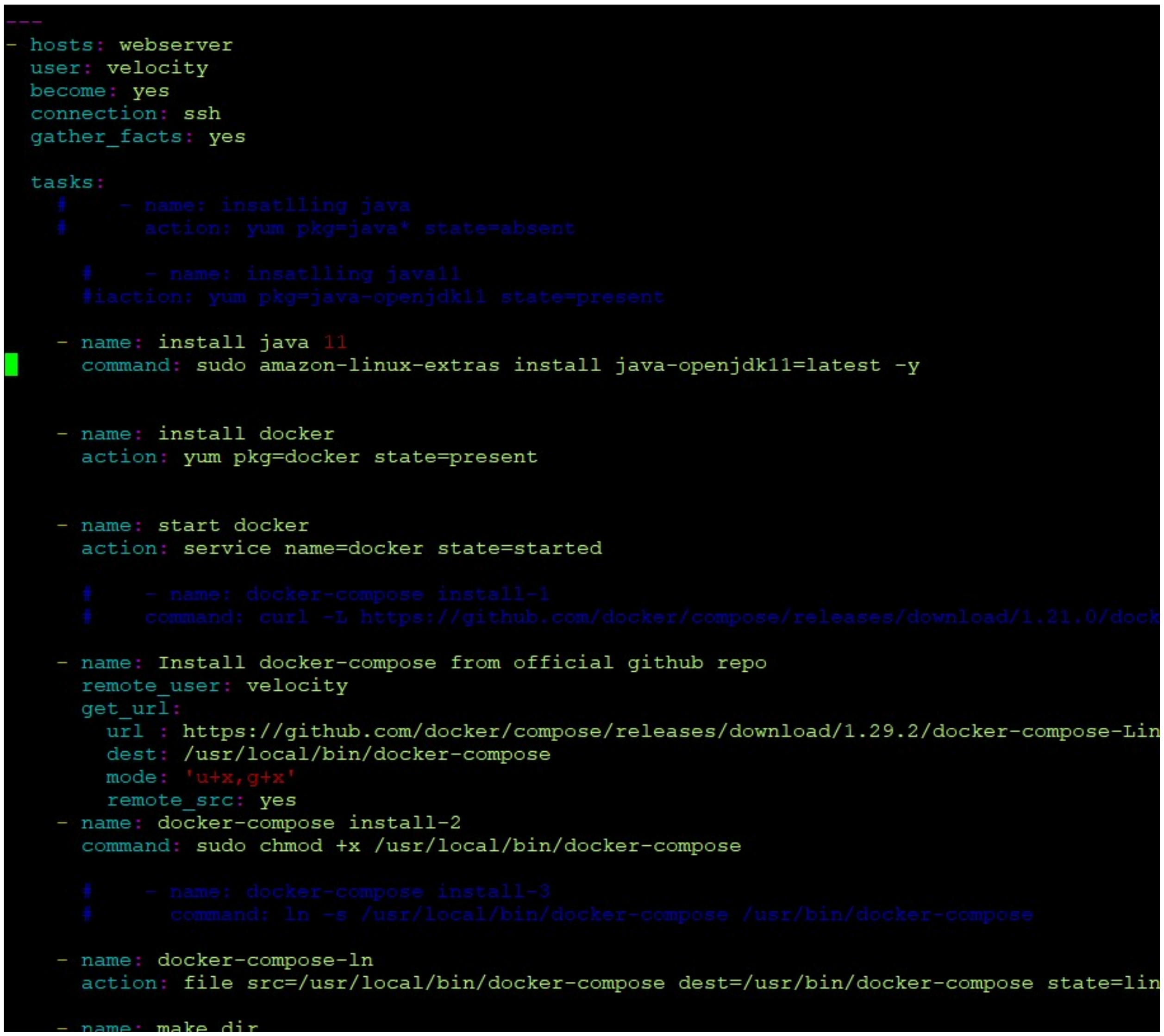
3.144.106.80:8080/jenkins/view/Pipeline/

STEP7) NOW WE NEED TO ANSIBLE-PLAYBOOK ON MASTER TO INSTALL DEPENDENCY ON ALL THE NODE MACHINE (DEV1, DEV2, QA1, QA2)
WE ARE CREATING PLAYBOOK ONTO THE SERVER DIR (WE CAN USE IT FROM GIT HUB AS WELL BUT AS OF NOW WE ARE CREATING IT OVER THIS DIR AND WILL USE IT FROM HERE TO INSTALL THE REQ SOFT ON NODES)

COMMAND:
```
cd /mnt/server
Sudo vi project.yaml
```

```
---
- hosts: webserver
  user: velocity
  become: yes
  connection: ssh
  gather_facts: yes

  tasks:
   #    - name: insatlling java
   #      action: yum pkg=java* state=absent

     #    - name: insatlling java11
     #iaction: yum pkg=java-openjdk11 state=present

   - name: install java 11
     command: sudo amazon-linux-extras install java-openjdk11=latest -y



   - name: install docker
     action: yum pkg=docker state=present



   - name: start docker
     action: service name=docker state=started

     #    - name: docker-compose install-1
     #    command: curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-
compose-`uname -s`-`uname -m` | sudo tee /usr/local/bin/docker-compose > /dev/null

   - name: Install docker-compose from official github repo
     remote_user: velocity
     get_url:
       url : https://github.com/docker/compose/releases/download/1.29.2/docker-compose-Linux-
x86_64
       dest: /usr/local/bin/docker-compose
       mode: 'u+x,g+x'
       remote_src: yes
   - name: docker-compose install-2
     command: sudo chmod +x /usr/local/bin/docker-compose

     #    - name: docker-compose install-3
     #      command: ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

   - name: docker-compose-ln
     action: file src=/usr/local/bin/docker-compose dest=/usr/bin/docker-compose state=link
force=yes

   - name: make dir
     action: file path=/mnt/docky state=directory
```

(NOTE: YOU HAVE TO RUN THE PLAYBOOK HERE ONLY AS WE WILL NEED JAVA 11 ON EACH OF NODE MACHINE TO CONNECT WITH OUR JENKINS SERVER)



```
---
- hosts: webserver
  user: velocity
  become: yes
  connection: ssh
  gather_facts: yes

  tasks:
    #    - name: insatlling java
    #      action: yum pkg=java* state=absent

      #    - name: insatlling java11
      #iaction: yum pkg=java-openjdk11 state=present

    - name: install java 11
      command: sudo amazon-linux-extras install java-openjdk11=latest -y


    - name: install docker
      action: yum pkg=docker state=present


    - name: start docker
      action: service name=docker state=started

      #    - name: docker-compose install-1
      #    command: curl -L https://github.com/docker/compose/releases/download/1.21.0/dock

    - name: Install docker-compose from official github repo
      remote_user: velocity
      get_url:
        url : https://github.com/docker/compose/releases/download/1.29.2/docker-compose-Lin
        dest: /usr/local/bin/docker-compose
        mode: 'u+x,g+x'
        remote_src: yes
    - name: docker-compose install-2
      command: sudo chmod +x /usr/local/bin/docker-compose

      #    - name: docker-compose install-3
      #      command: ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

    - name: docker-compose-ln
      action: file src=/usr/local/bin/docker-compose dest=/usr/bin/docker-compose state=lin

    - name: make dir
```

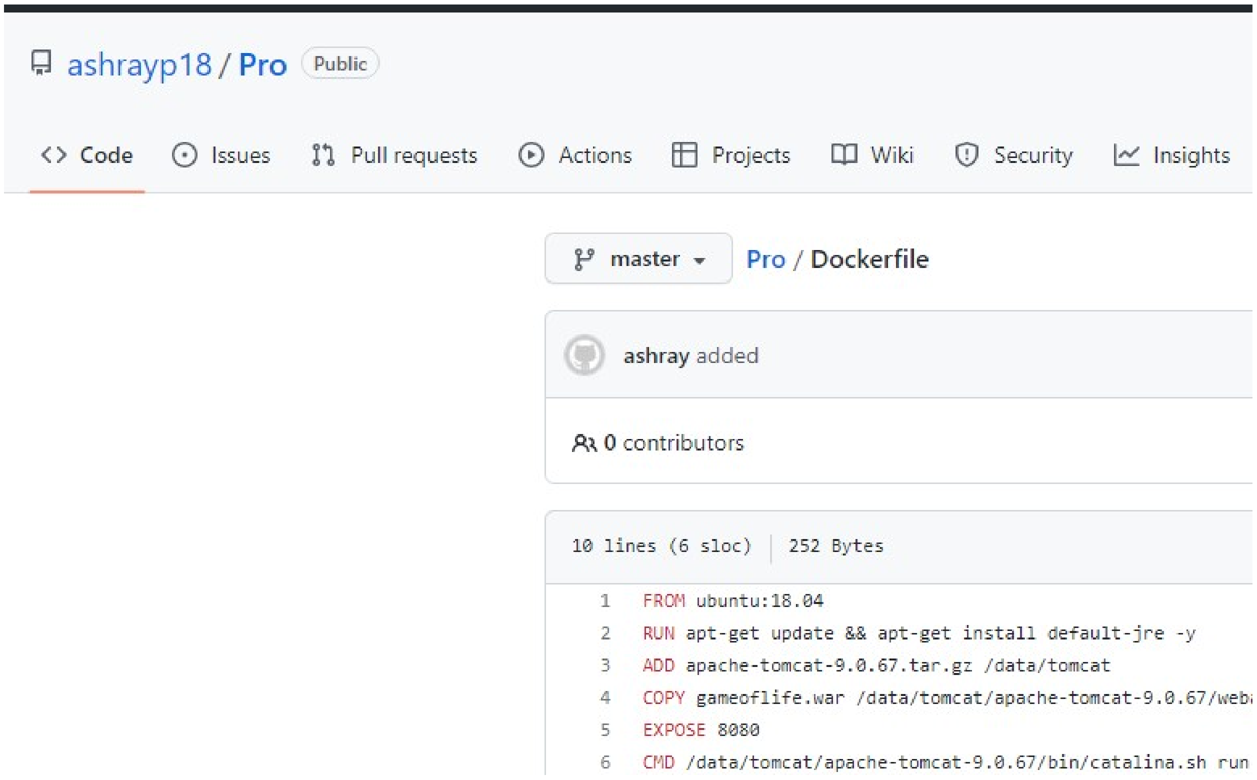
(NOTE : WE WILL INSTALL JAVA 11, DOCKER, START DOCKER SERVICES, DOCKER-COMPOSE INSTALLATION, SETTING THE SL FOR DOCKER-COMPOSE, AND WE WILL ALSO CREATE ONE DIR CALLED AS DOCKY WHICH WE WILL USE LATER TO CLONE DOCKERFILE AND DOCKER-COMPOSE FILE FROM GIT HUB)

STEP8) NOW WE NEED TO CREATE DOCKERFILE AND DOCKER-COMPOSE ON MASTER MACHINE AND PUSH IT OVER THE GITHUB.

COMMAND: vi Dockerfile

```
FROM
ubuntu:18.04
              RUN apt-get update && apt-get install default-jre -y
              ADD apache-tomcat-9.0.67.tar.gz /data/tomcat
```

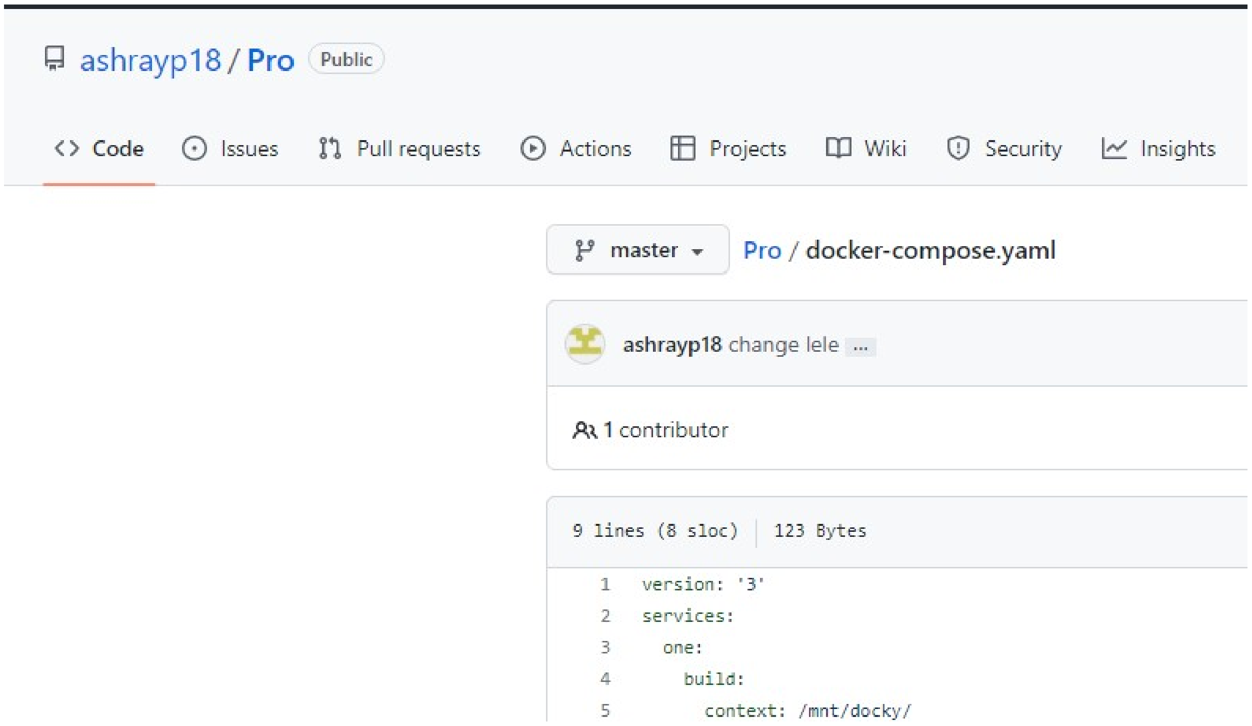
```
COPY gameoflife.war /data/tomcat/apache-tomcat-9.0.67/webapps
EXPOSE 8080
CMD /data/tomcat/apache-tomcat-9.0.67/bin/catalina.sh run
```



```
FROM ubuntu:18.04
RUN apt-get update && apt-get install default-jre -y
ADD apache-tomcat-9.0.67.tar.gz /data/tomcat
COPY gameoflife.war /data/tomcat/apache-tomcat-9.0.67/weba
EXPOSE 8080
CMD /data/tomcat/apache-tomcat-9.0.67/bin/catalina.sh run
```

vi docker-compose.yaml

```
version:'3'
               services:
                 one:
                   build:
                     context: /mnt/docky/
                   image: server-1
                   ports:
                   - "8091-8092:8080"
```


ashrayp18 / Pro Public

Code Issues Pull requests Actions Projects Wiki Security Insights

master Pro / docker-compose.yaml

ashrayp18 change lele ...

1 contributor

9 lines (8 sloc) | 123 Bytes

```
version: '3'
services:
  one:
    build:
      context: /mnt/docky/
```

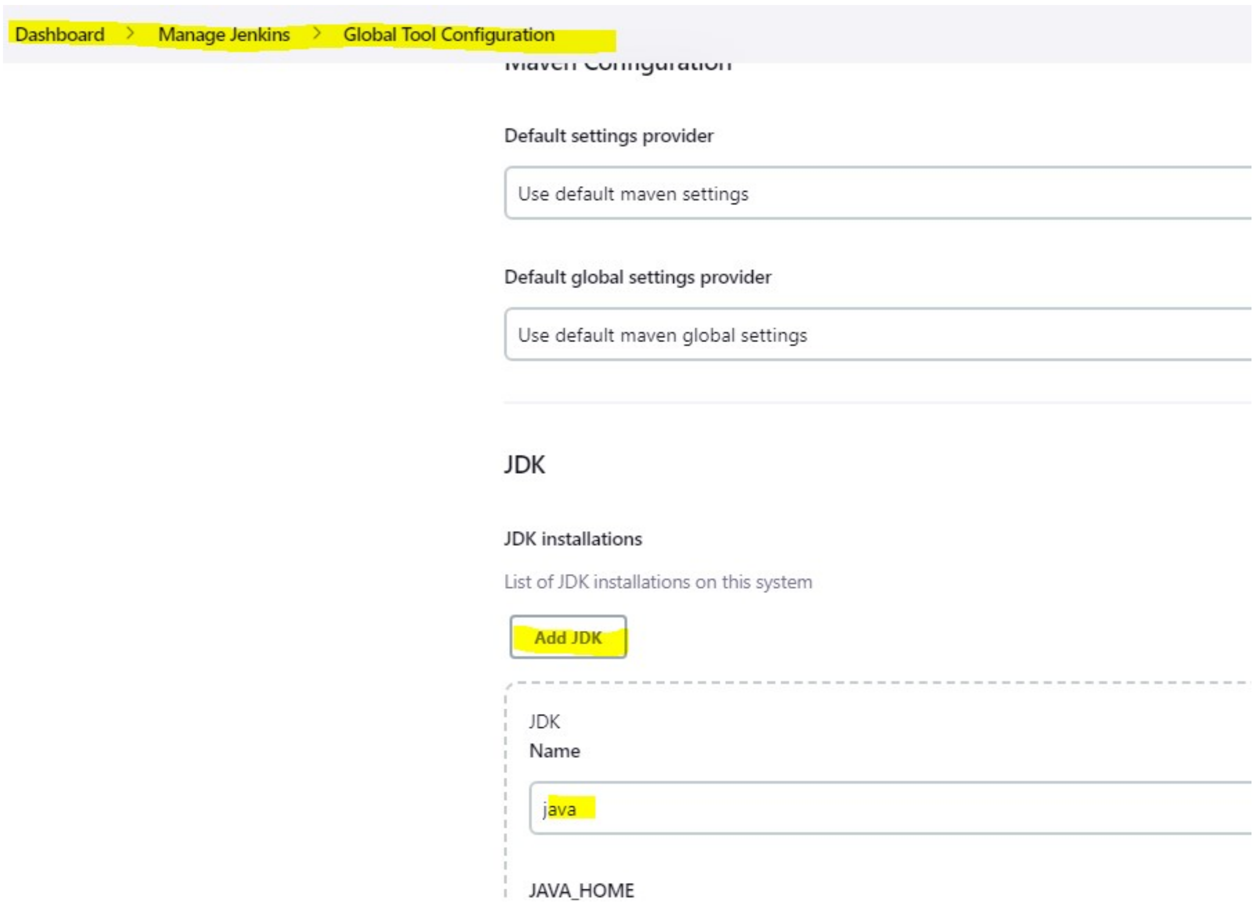
AS YOU CAN SEE WE HAVE ALREADY UPDATED BOTH THE FILE TO OUR GIT REPO CALLED AS PRO

STEP9) NOW WE WILL GO ONTO JENKINS SERVER TO CONFIGURE GLOBAL TOOL SETTING
HERE WE ARE RUNNING JENKINS MASTER ON JAVA 11 AND OUR PROJECT (GAMEOFLIFE) NEEDS JAVA 1.8 TO BUILD. HENCE WE NEED TO DEFINE TOOLS.
TOOLS WILL HELP OUR JENKINS TO RUN ON JAVA 11 BUT AND USE JAVA 1.8 IN THE BACKGROUND TO RUN JOB/STAGE.

SEE THE SETTING AS BELOW.

MAKE SURE YOU REMEMBER THE NAME YOU HAVE PROVIDED UNDER JDK AS WE ARE GOING TO USE IT FURTHER IN OUT JOBS.

PATH: DASHBOARD >> MANAGE JENKINS >> GLOBAL TOOL CONFIGURATION


Dashboard > Manage Jenkins > Global Tool Configuration

Maven Configuration

Default settings provider

Use default maven settings

Default global settings provider

Use default maven global settings

JDK

JDK installations

List of JDK installations on this system

Add JDK

JDK

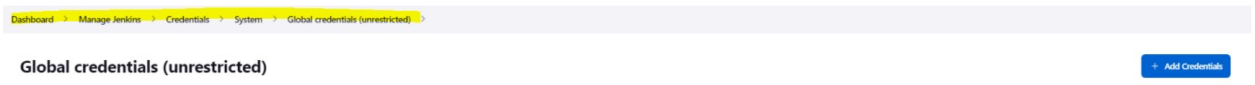
Name

java

JAVA_HOME

STEP10) NOW WE NEED TO CONNECT JENKINS MASTER TO JENKINS HOST
FOR THAT WE FIRST NEED TO CREATE CREDENTIAL SO THAT OUR JENKINS MASTER CAN USE OUR SSH KEYS TO CONNECT TO HOST

FOR THIS WE NEED TO GO ON
MANAGE JENKINS >> CREDENTIALS >> SYSTEM >> GLOCBAL CREDENTIAL>>


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

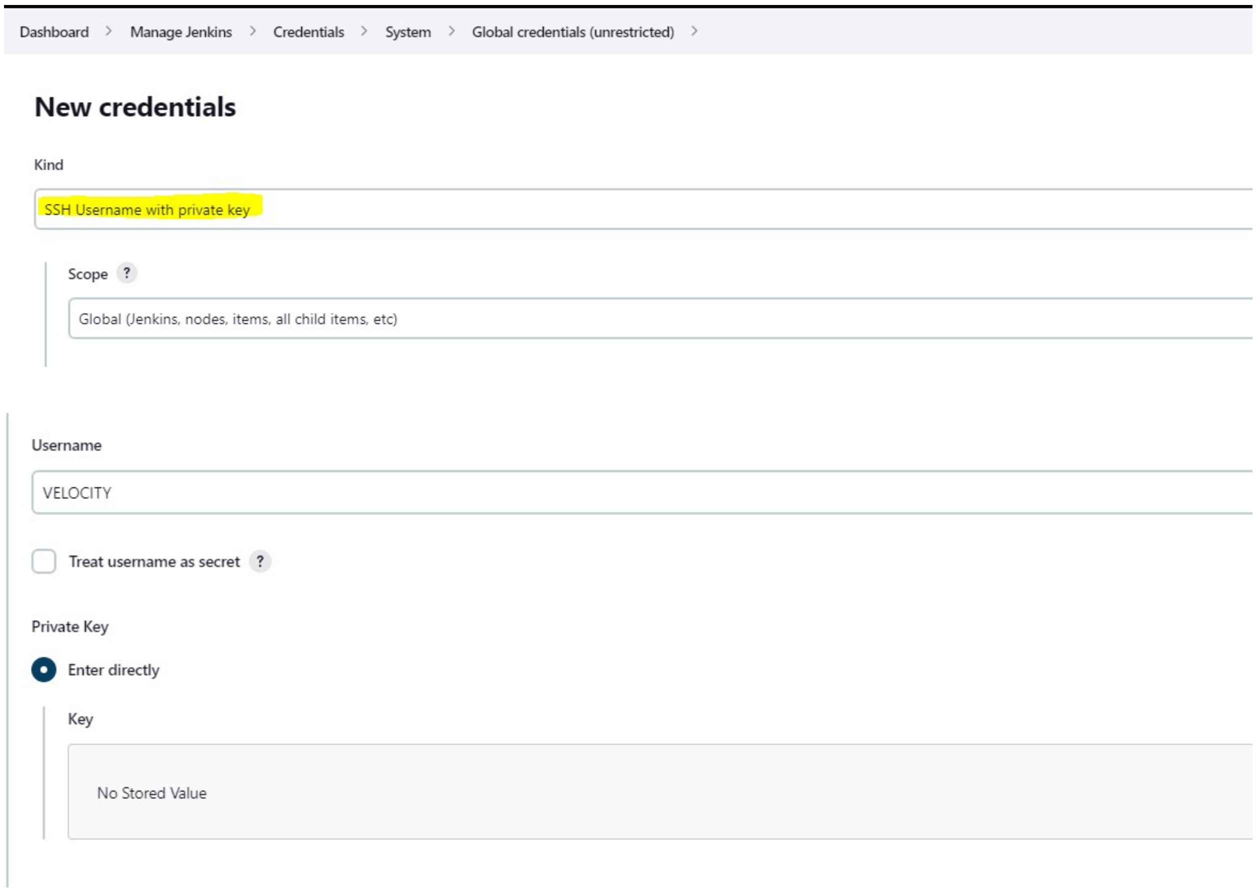
Global credentials (unrestricted)

+ Add Credentials

CLICK ON CREATE CREDENTIAL ON THE RIGHT SIDE.

FILL IN THE DETAILS

SELECT " SSH USERNAME WITH PRIVATE KEY"
ID AS "VELOCITY"
USERNAME AS "VELOCITY"
PASSWORD AS "


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind

SSH Username with private key

Scope

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

Username

VELOCITY

☐ Treat username as secret

Private Key

● Enter directly

Key

No Stored Value

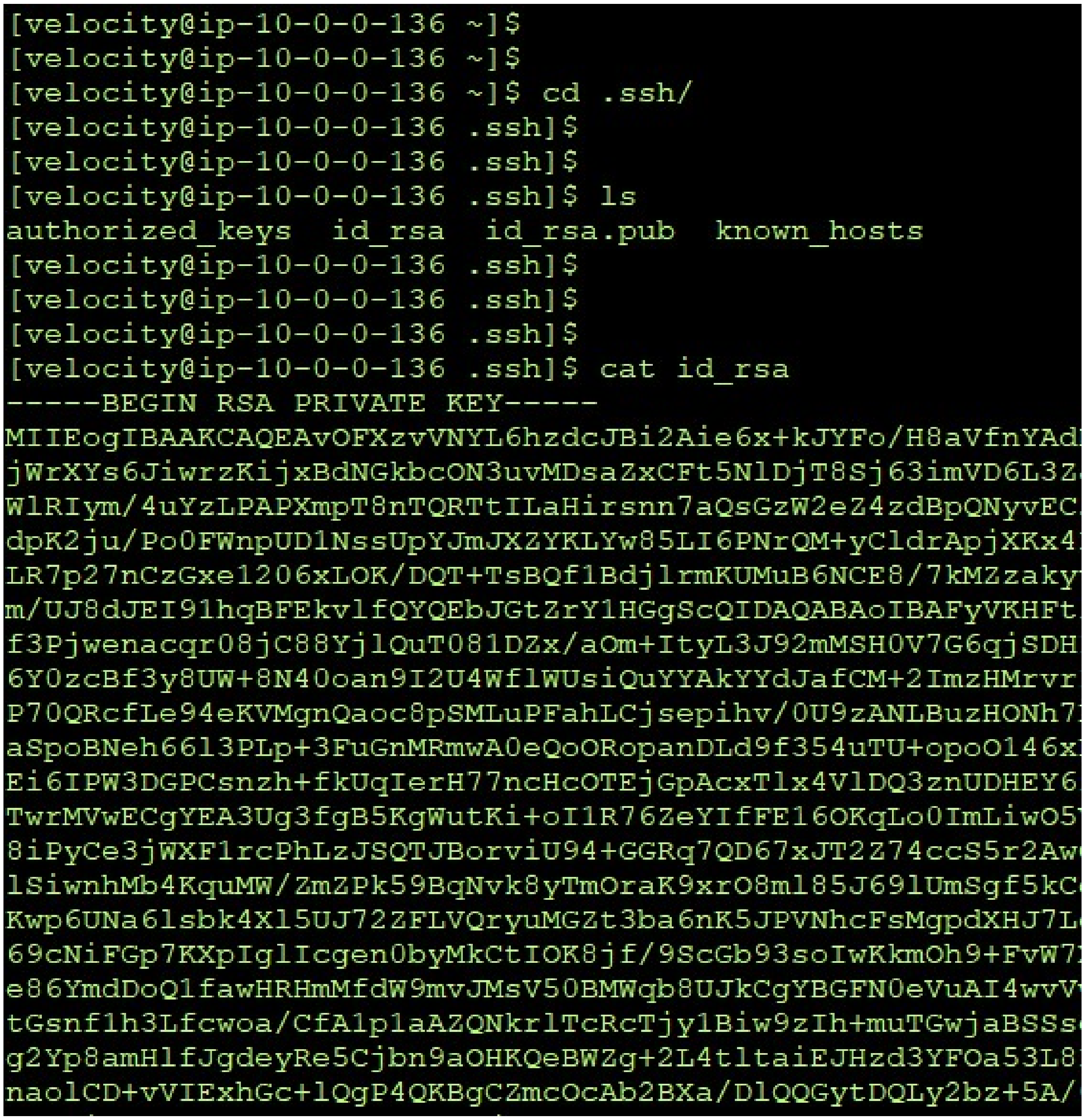
CLICK ON ADD TO ADD THE PRIVATE KEY

SO WE NEED THE KEY NOW FOR USER VELOCITY.

FOR THAT USE THE FOLLOWING COMMAND:

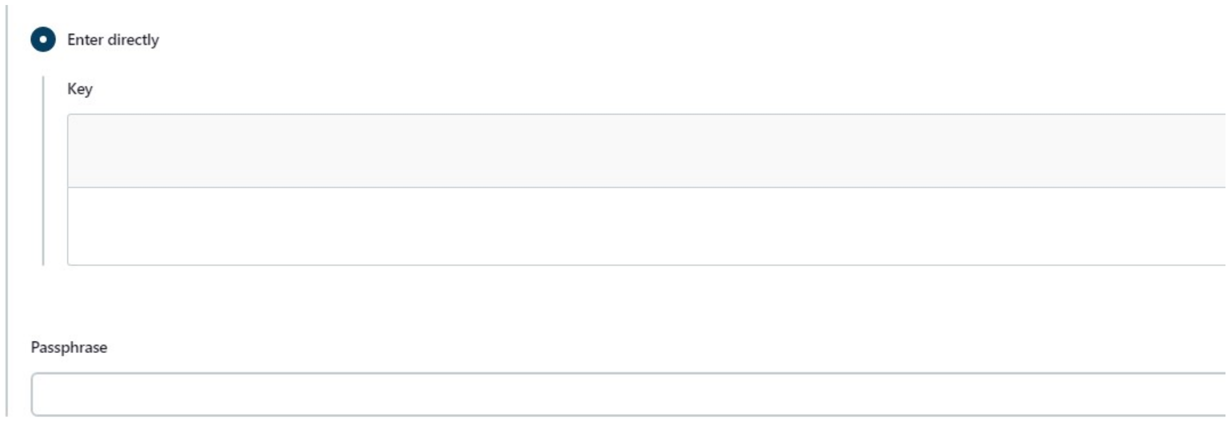
COMMAND:
```
cd
cd .ssh/
ls  -la
```
YOU WILL SEE THE PUB AND PRVATE KEY WHICH YOU HAVE CREATED EARLIER USING THE SSH-KEYGEN COMMAND.

NOW COPY THE CONTENT OF ID_RSA FILE AS SHOWN BELOW



```
[velocity@ip-10-0-0-136 ~]$
[velocity@ip-10-0-0-136 ~]$
[velocity@ip-10-0-0-136 ~]$ cd .ssh/
[velocity@ip-10-0-0-136 .ssh]$
[velocity@ip-10-0-0-136 .ssh]$
[velocity@ip-10-0-0-136 .ssh]$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
[velocity@ip-10-0-0-136 .ssh]$
[velocity@ip-10-0-0-136 .ssh]$
[velocity@ip-10-0-0-136 .ssh]$
[velocity@ip-10-0-0-136 .ssh]$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvOFXzvVNYL6hzdcJBi2Aie6x+kJYFo/H8aVfnYAd
jWrXYs6JiwrzKijxBdNGkbcON3uvMDsaZxCFt5NlDjT8Sj63imVD6L3Z
WlRIym/4uYzLPAPXmpT8nTQRTtILaHirsnn7aQsGzW2eZ4zdBpQNyvEC
dpK2ju/Po0FWnpUD1NssUpYJmJXZYKLYw85LI6PNrQM+yCldrApjXKx4
LR7p27nCzGxe1206xLOK/DQT+TsBQf1BdjlrmKUMuB6NCE8/7kMZzaky
m/UJ8dJEI91hqBFEkvlfQYQEbJGtZrY1HGgScQIDAQABAoIBAFyVKHFt
f3Pjwenacqr08jC88YjlQuT081DZx/aOm+ItyL3J92mMSH0V7G6qjSDH
6Y0zcBf3y8UW+8N40oan9I2U4WflWUsiQuYYAkYYdJafCM+2ImzHMrvr
P70QRcfLe94eKVMgnQaoc8pSMLuPFahLCjsepihv/0U9zANLBuzHONh7
aSpoBNeh66l3PLp+3FuGnMRmwA0eQoORopanDLd9f354uTU+opoO146x
Ei6IPW3DGPCsnzh+fkUqIerH77ncHcOTEjGpAcxTlx4VlDQ3znUDHEY6
TwrMVwECgYEA3Ug3fgB5KgWutKi+oI1R76ZeYIfFE16OKqLo0ImLiwO5
8iPyCe3jWXF1rcPhLzJSQTJBorviU94+GGRq7QD67xJT2Z74ccS5r2Aw
lSiwnhMb4KquMW/ZmZPk59BqNvk8yTmOraK9xrO8ml85J69lUmSgf5kC
Kwp6UNa6lsbk4Xl5UJ72ZFLVQryuMGZt3ba6nK5JPVNhcFsMgpdXHJ7L
69cNiFGp7KXpIglIcgen0byMkCtIOK8jf/9ScGb93soIwKkmOh9+FvW7
e86YmdDoQ1fawHRHmMfdW9mvJMsV50BMWqb8UJkCgYBGFN0eVuAI4wvV
tGsnf1h3Lfcwoa/CfA1p1aAZQNkrlTcRcTjy1Biw9zIh+muTGwjaBSSs
g2Yp8amHlfJgdeyRe5Cjbn9aOHKQeBWZg+2L4tltaiEJHzd3YFOa53L8
naolCD+vVIExhGc+lQgP4QKBgCZmcOcAb2BXa/DlQQGytDQLy2bz+5A/
```

NOW PASTE THE CONTENT OVER HERE UNDER THE KEY OPTION AND CLICK ON CREATE

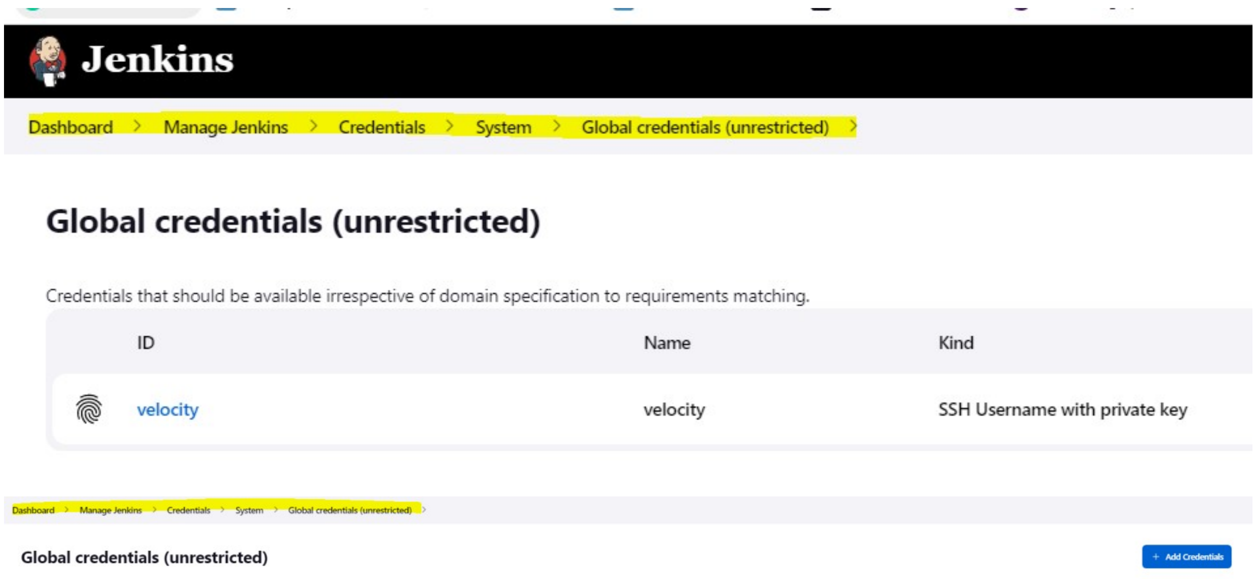


● Enter directly

Key

Passphrase

YOU WILL SEE A CREDENTIAL HAS BEEN CREATED.



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

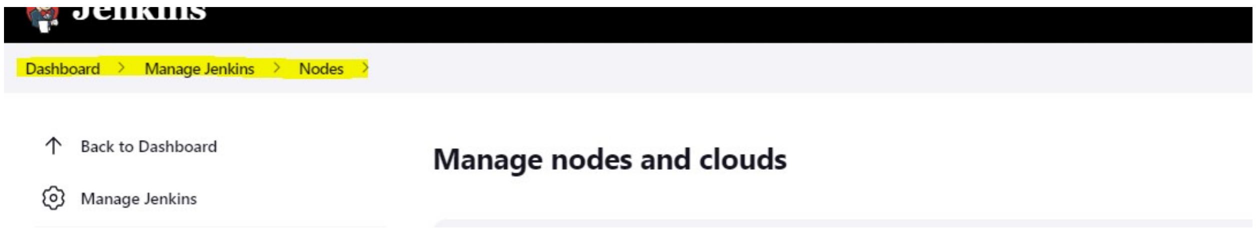
ID	Name	Kind
velocity	velocity	SSH Username with private key

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

STEP11) NOW WE NEED TO ADD THE SLAVE FOR OUR JENKINS MASTER

GO TO DASHBOARD >> MANAGE JENKINS >> NODES>>
CLICK ON +NEW NODE



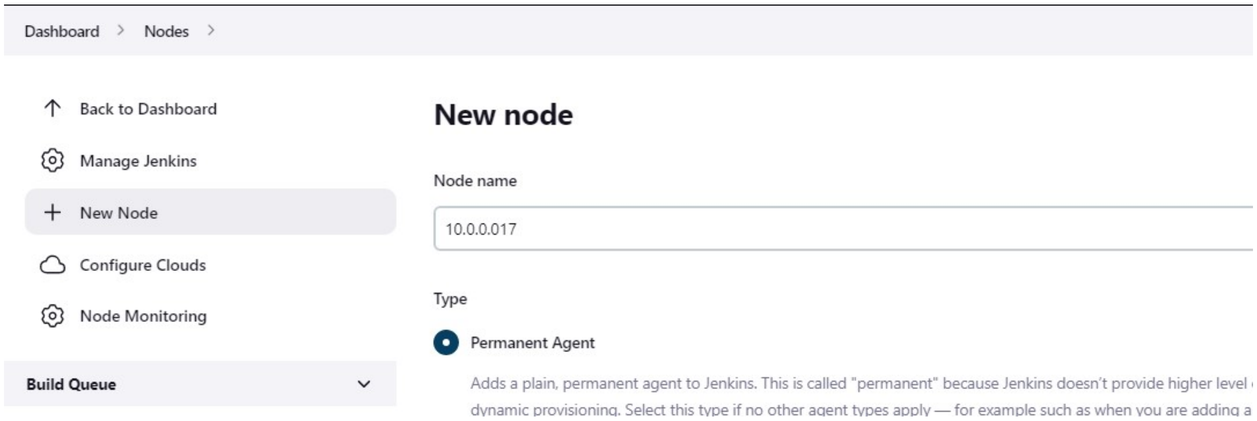
Dashboard > Manage Jenkins > Nodes >

Manage nodes and clouds

ENTER THE NAME AS 10.0.0.17

NOTE (WE ARE ADDING THE IP OF DEV1 MACHINE)

CLICK ON CREATE AFTER ADDING THE NODE NAME



Dashboard > Nodes >

Back to Dashboard

Manage Jenkins

New Node

Configure Clouds

Node Monitoring

Build Queue

New node

Node name

10.0.0.017

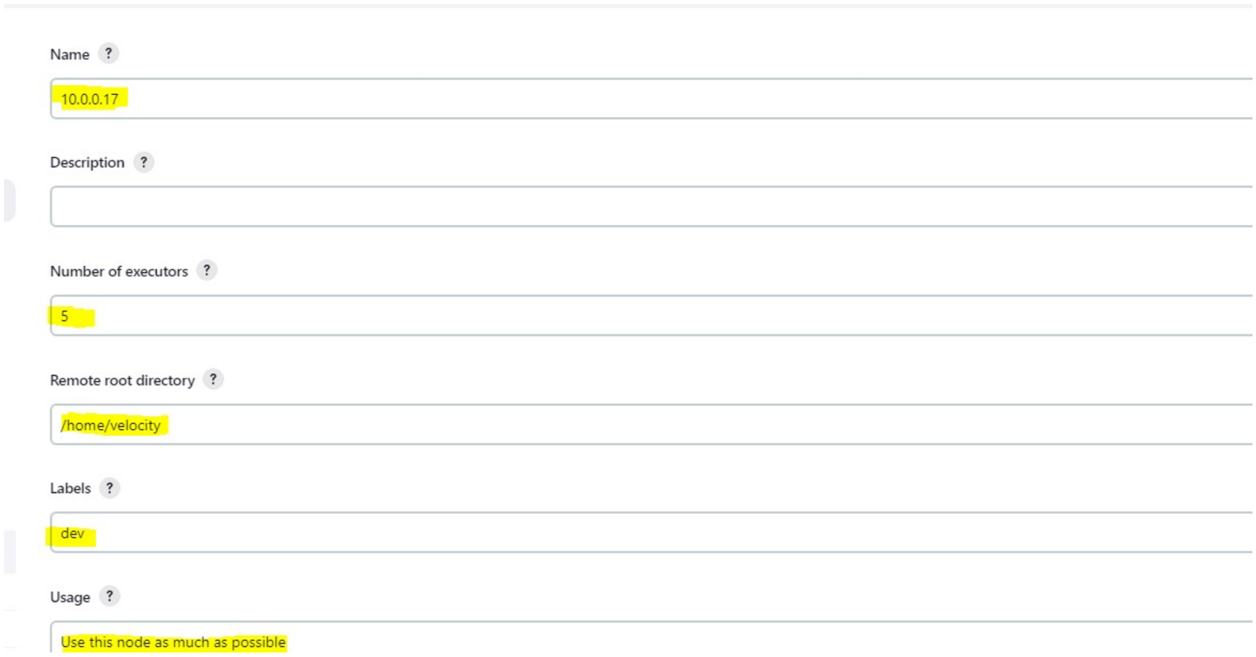
Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level

dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a

CONFIGURE THE NODE AS SHOWN IN PICTURE BELOW



Name ?

10.0.0.17

Description ?

Number of executors ?

5

Remote root directory ?

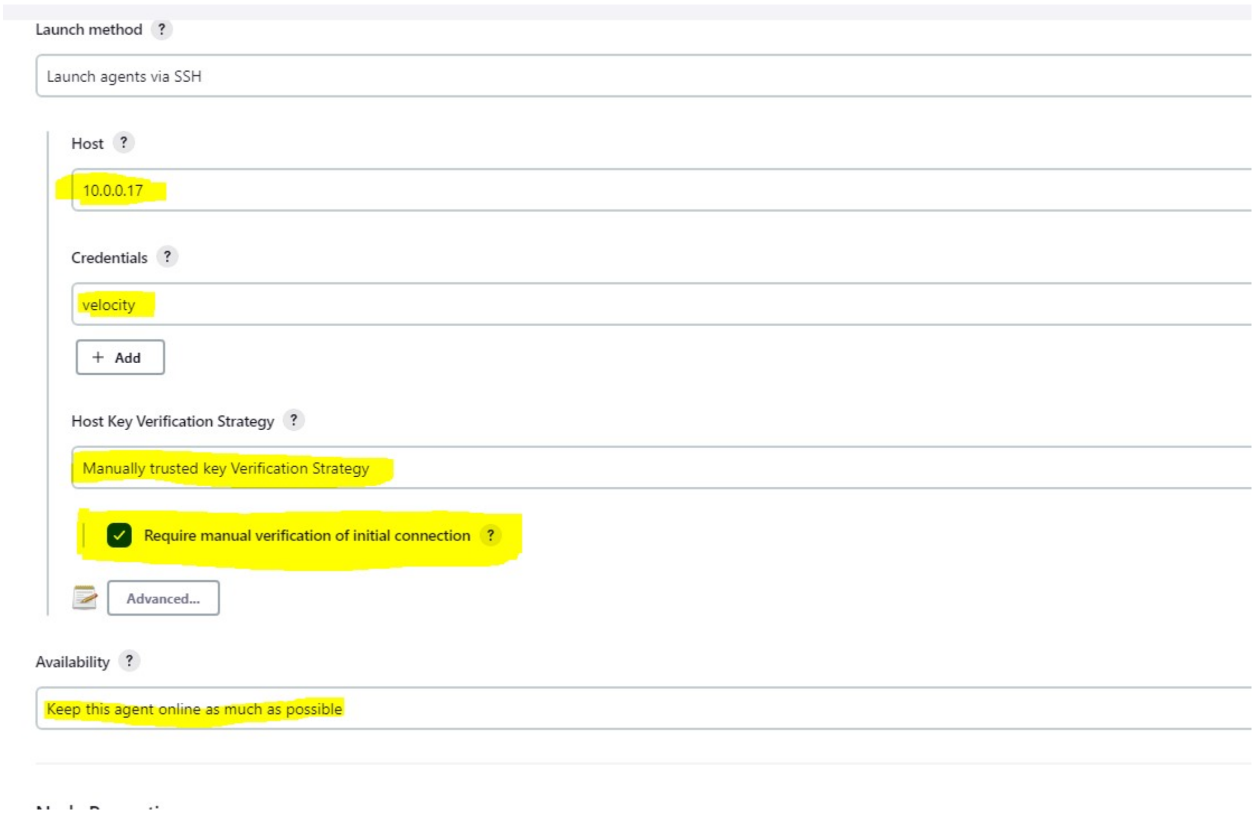
/home/velocity

Labels ?

dev

Usage ?

Use this node as much as possible


Launch method ?

Launch agents via SSH

Host ?

10.0.0.17

Credentials ?

velocity

+ Add

Host Key Verification Strategy ?

Manually trusted key Verification Strategy

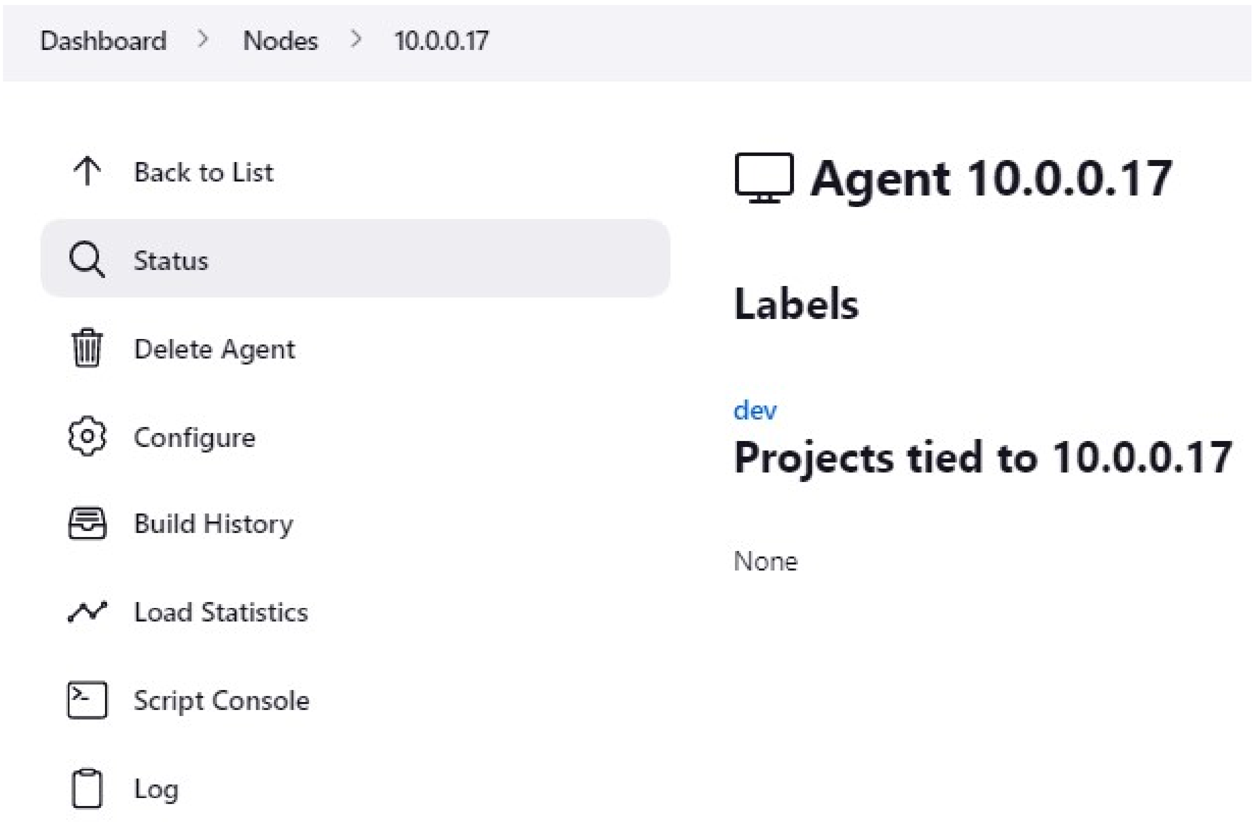
✓ Require manual verification of initial connection ?

Advanced...

Availability ?

Keep this agent online as much as possible

CLICK ON SAVE AND CLICK ON AGENT.


Dashboard > Nodes > 10.0.0.17

Agent 10.0.0.17

Labels

dev

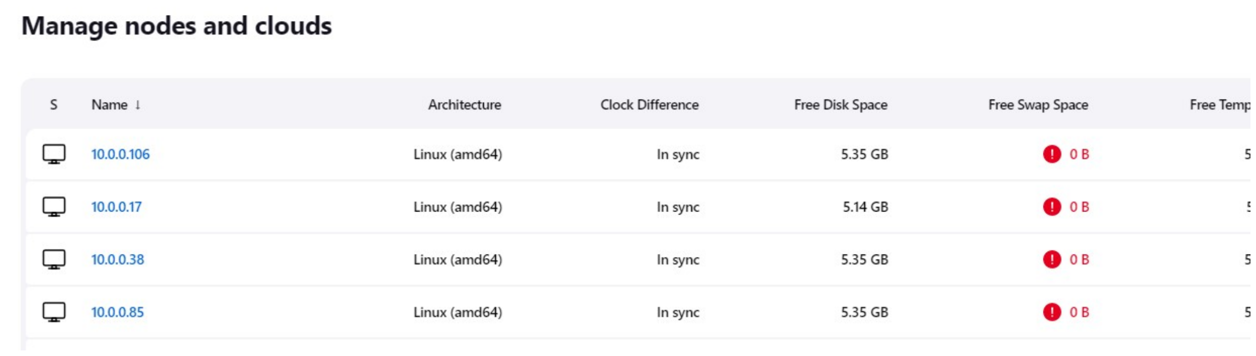
Projects tied to 10.0.0.17

None

NOTE: CURRENTLY WE HAVE ALREADY CONNECTED IT BUT YOU WON'T BE ABLE TO CONNECT IT UNTIL YOU GRANT THE MANNUAL SSH PERMISSION WHICH IT SHOWS BELOW LOG OPTION
ONCE YOU DID THIS THEN YOUR AGENT SHOULD BE ONLINE

(NOTE WE NEED TO HAVE JAVA 11 ON ALL THE NODES BEFORE FOR THE CONNECTION TO BE MADE CONSIDER RUNNING THE PLAYBOOK FIRST FROM THE MASTER SO YOUR CONNECTION WILL BE SUCCESSFUL.)

SO NOW MAKE SUCH CONNECTION WITH EACH NODE (DEV1, DEV2, QA1, Q2)



Manage nodes and clouds

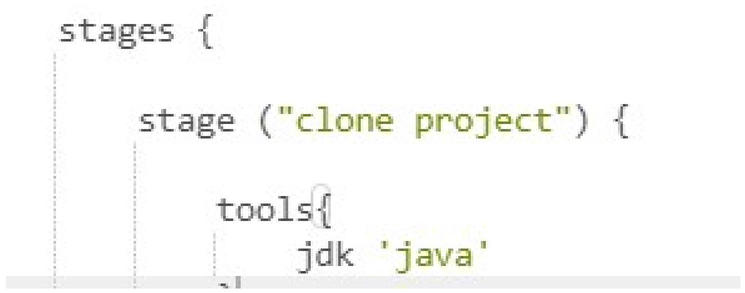
S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp
🖥	10.0.0.106	Linux (amd64)	In sync	5.35 GB	❗ 0 B	5
🖥	10.0.0.17	Linux (amd64)	In sync	5.14 GB	❗ 0 B	5
🖥	10.0.0.38	Linux (amd64)	In sync	5.35 GB	❗ 0 B	5
🖥	10.0.0.85	Linux (amd64)	In sync	5.35 GB	❗ 0 B	5

STEP12) NOW WE WILL CREATE JENKINS JOB:
WE WILL CREATE 3 JENKINS JOB

JOB1--- WE WILL CLONE THE PROJECT FROM GIT HUB AND BUILD IT ON MASTER
JOB2—WE WILL DEOPLY CONATINER USING DOCKER-COMPOSE AND DOCKERFILE ON EACH NODE (DEV1,DEV2, QA1, QA2), ALSO WE WILL COPY THE .WAR FROM OUR MASTER TO EACH SLAVE.
JOB3—HERE WE WILL BUILD JOB1-JOB2 SO THAT IT FIRST TRIGGERS JOB1 AND LATER JOB2.

JOB1:
WE WILL CLONE GAMEOFLIFE PROJECT UNDER /MNT/PROJECT DIR.
NOTE: WE WILL HAVE TO ASSIGN TOOLS WHICH WE HAVE CONFIGURED EARLIER

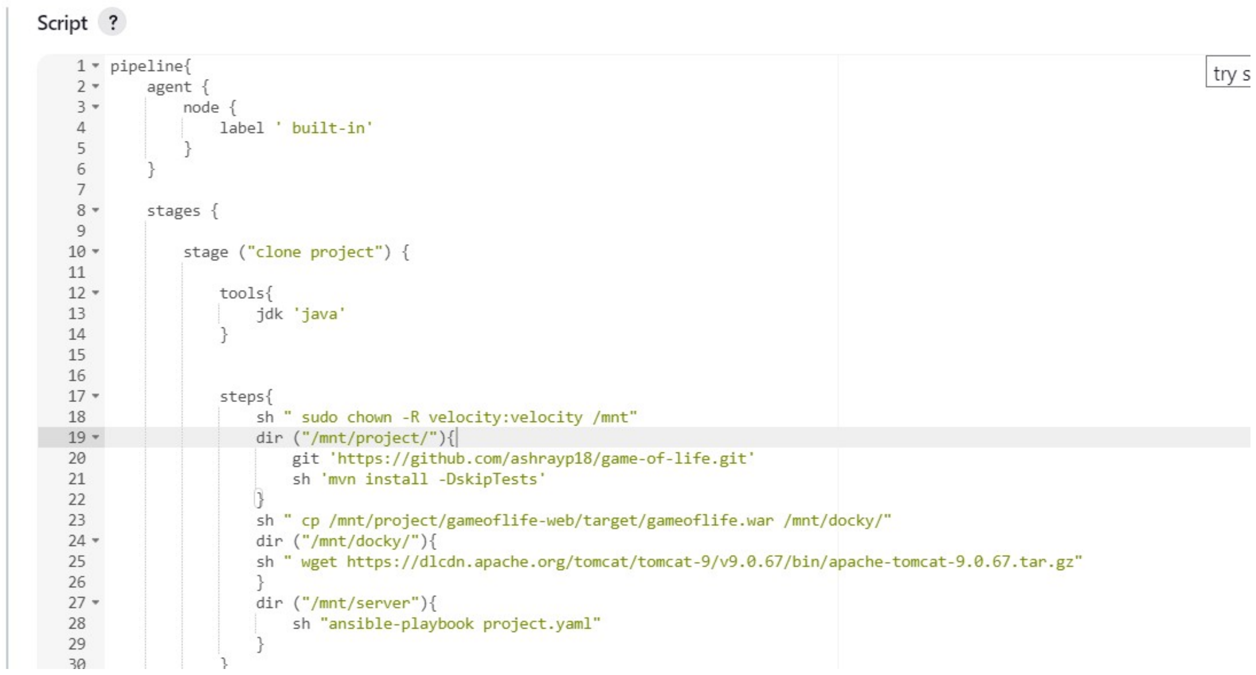
```
tools{
            jdk 'java'
        }
```



```
stages {

    stage ("clone project") {

        tools{
            jdk 'java'
```

ALSO WE WARE RUNNING THE PLAYBOOK FROM JOB1. (BUT WE WILL HAVE TO RUN IN MANUALLY FIRST ON MASTER AFTER CREATING IT ELSE WE WILL NOT BE ABLE TO MAKE NODE CONNECTION DUE TO DEPENDENCY)

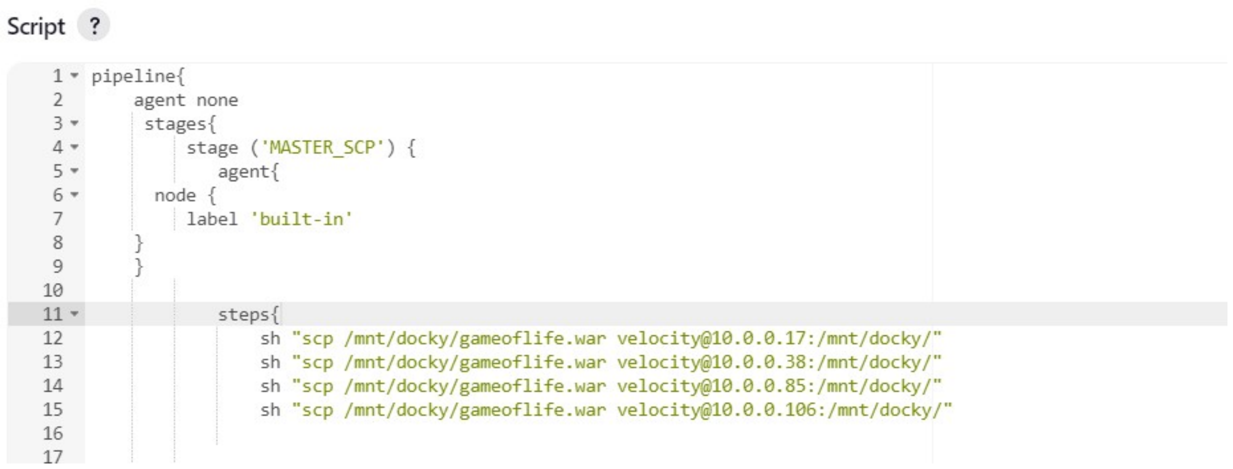
```
pipeline{
   agent {
      node {
         label ' built-in'
      }
   }

   stages {

      stage ("clone project") {

         tools{
            jdk 'java'
         }



         steps{
            sh " sudo chown -R velocity:velocity /mnt"
            dir ("/mnt/project/"){
               git 'https://github.com/ashrayp18/game-of-life.git'
               sh 'mvn install -DskipTests'
            }
            sh " cp /mnt/project/gameoflife-web/target/gameoflife.war /mnt/docky/"
            dir ("/mnt/docky/"){
            sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-
9.0.67.tar.gz"
            }
            dir ("/mnt/server"){
               sh "ansible-playbook project.yaml"
            }
         }
      }


   }
```



```
pipeline{
    agent {
        node {
            label ' built-in'
        }
    }

    stages {

        stage ("clone project") {

            tools{
                jdk 'java'
            }


            steps{
                sh " sudo chown -R velocity:velocity /mnt"
                dir ("/mnt/project/"){
                    git 'https://github.com/ashrayp18/game-of-life.git'
                    sh 'mvn install -DskipTests'
                }
                sh " cp /mnt/project/gameoflife-web/target/gameoflife.war /mnt/docky/"
                dir ("/mnt/docky/"){
                sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.gz"
                }
                dir ("/mnt/server"){
                    sh "ansible-playbook project.yaml"
                }
            }
```

JOB2: CREATE A NEW PIPELINE JOB CALLED AS JOB2.
HERE FIRST WE WILL COPY THE GAMEOFLIFE.WAR FILE WHICH WE HAVE BUILD AND COPIED ON ON JOB1. FOR DOING SO WE WILL USE THE SCP COMMAND
WE WILL NAME THE STAGE AS MASTER-SCP:

```
pipeline{
   agent none
    stages{
       stage ('MASTER_SCP') {
          agent{
     node {
       label 'built-in'
   }
   }


          steps{
             sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.17:/mnt/docky/"
             sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.38:/mnt/docky/"
             sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.85:/mnt/docky/"
             sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.106:/mnt/docky/"


      }
      }
```

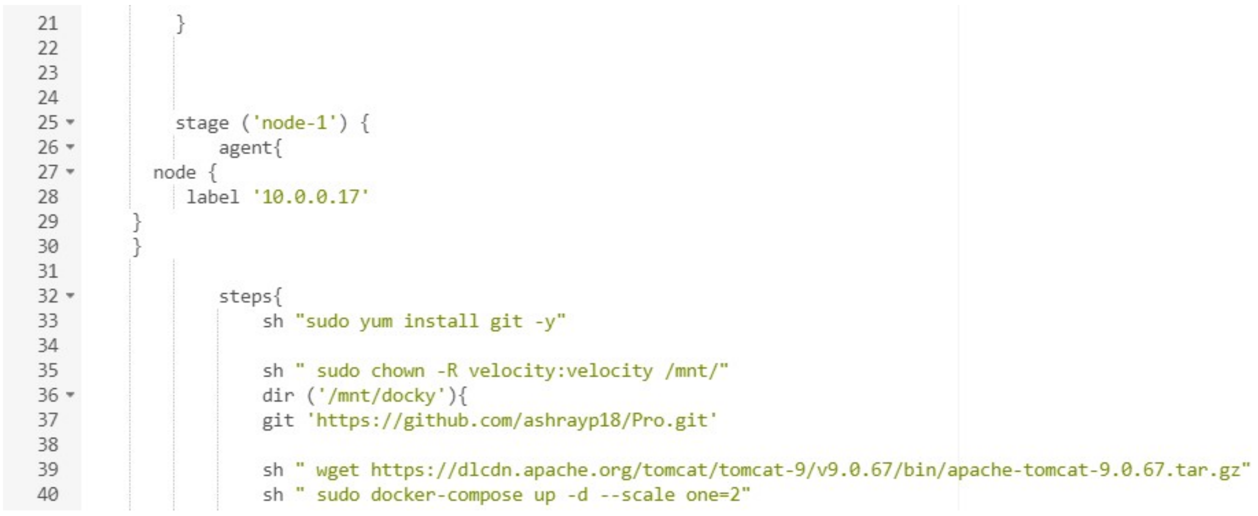


Script ?

```
pipeline{
     agent none
      stages{
          stage ('MASTER_SCP') {
              agent{
        node {
          label 'built-in'
     }
     }

              steps{
                  sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.17:/mnt/docky/"
                  sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.38:/mnt/docky/"
                  sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.85:/mnt/docky/"
                  sh "scp /mnt/docky/gameoflife.war velocity@10.0.0.106:/mnt/docky/"
```

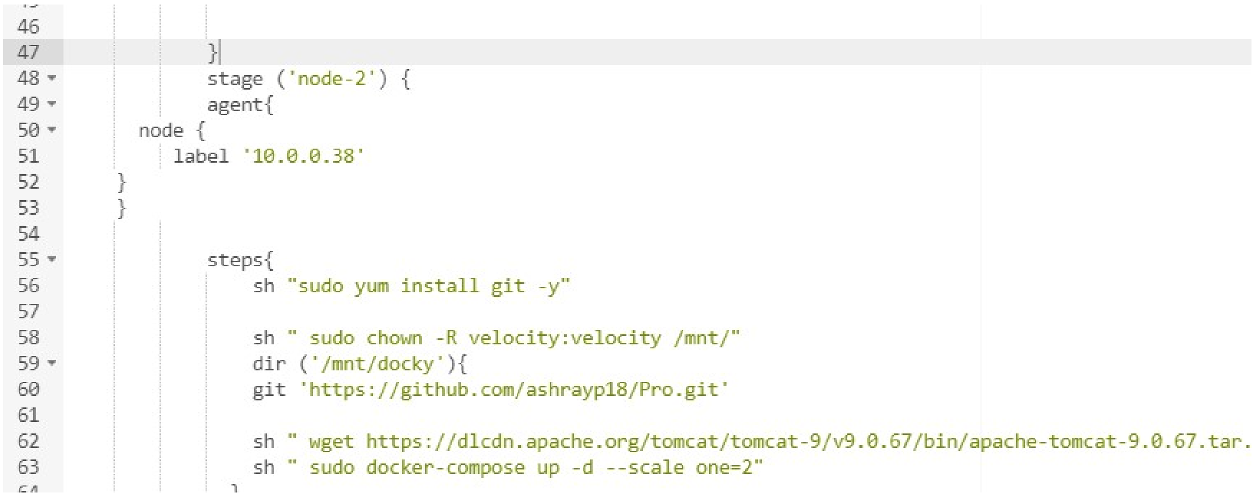
NOW WE WILL WRITE ANOTHER STAGE FOR NODE1
HERE WE WILL CLONE THE DOCKERFILE AND DOCKERCOMEFILE ON A DIR CALLED AS DOCKY.
NOTE: WHILE WRITING DOCKERFILE WE HAVE USED THE COPY AND ADD COMMAND WHICH REQ GAMEOFLIFE.WAR FILE AND APACHE-TOMAT:9 TAR.GZ FILE INTO THE SAME DIR.
SO WE WILL WGET THE TOMCAT FILE AND ALSO INSTALL GIT INTO THE EACH SLAVE AND CLONE OUR GIT REPO (PRO) WHICH HAS DOCKERFILE AND DOCKER-COMPOSE FILE.

```
stage ('node-1') {
            agent{
      node {
         label '10.0.0.17'
   }
   }


            steps{
               sh "sudo yum install git -y"

               sh " sudo chown -R velocity:velocity /mnt/"
               dir ('/mnt/docky'){
               git 'https://github.com/ashrayp18/Pro.git'

               sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-
9.0.67.tar.gz"
               sh " sudo docker-compose up -d --scale one=2"
              }



      }
```

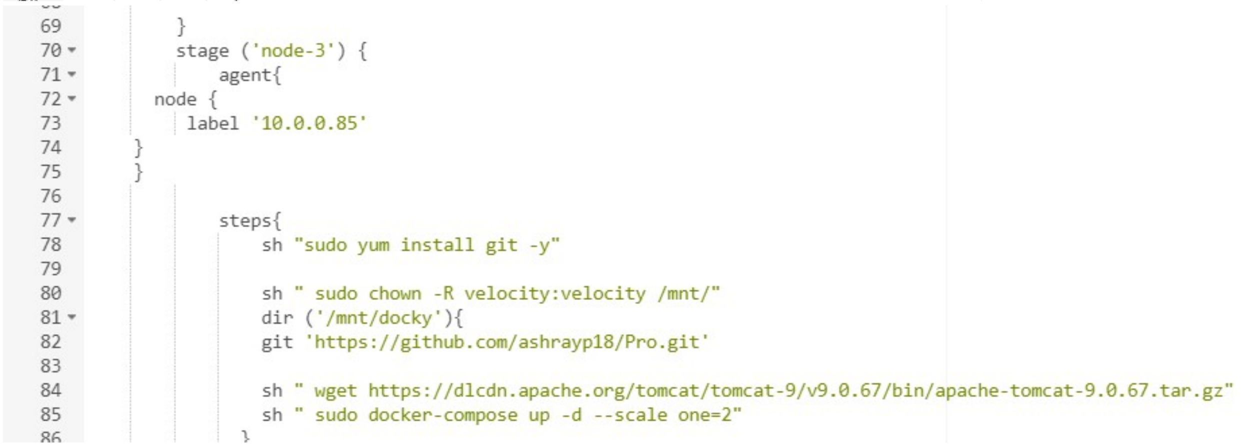


```
        }



        stage ('node-1') {
            agent{
      node {
         label '10.0.0.17'
   }
   }

            steps{
                sh "sudo yum install git -y"

                sh " sudo chown -R velocity:velocity /mnt/"
                dir ('/mnt/docky'){
                git 'https://github.com/ashrayp18/Pro.git'

                sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.gz"
                sh " sudo docker-compose up -d --scale one=2"
```

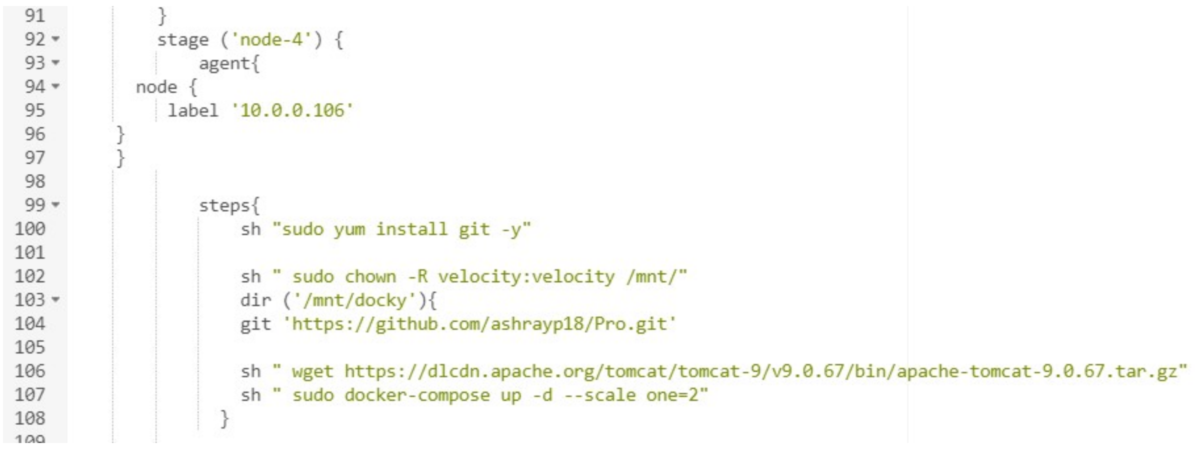
SIMILARLY WRITE STAGES FOR EVERYNODE NODE2, NODE3 AND NODE4)



```

            }
            stage ('node-2') {
            agent{
      node {
         label '10.0.0.38'
   }
   }

            steps{
                sh "sudo yum install git -y"

                sh " sudo chown -R velocity:velocity /mnt/"
                dir ('/mnt/docky'){
                git 'https://github.com/ashrayp18/Pro.git'

                sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.
                sh " sudo docker-compose up -d --scale one=2"
```



```
        }
        stage ('node-3') {
            agent{
      node {
         label '10.0.0.85'
   }
   }

            steps{
                sh "sudo yum install git -y"

                sh " sudo chown -R velocity:velocity /mnt/"
                dir ('/mnt/docky'){
                git 'https://github.com/ashrayp18/Pro.git'

                sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.gz"
                sh " sudo docker-compose up -d --scale one=2"
```

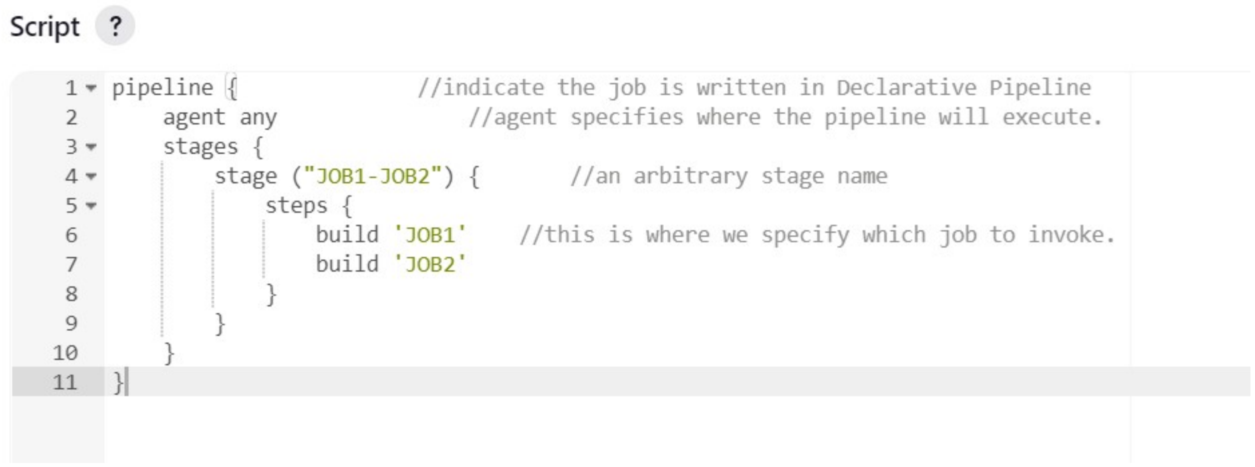


```
        }
        stage ('node-4') {
            agent{
      node {
         label '10.0.0.106'
    }
    }

            steps{
                sh "sudo yum install git -y"

                sh " sudo chown -R velocity:velocity /mnt/"
                dir ('/mnt/docky'){
                git 'https://github.com/ashrayp18/Pro.git'

                sh " wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.gz"
                sh " sudo docker-compose up -d --scale one=2"
              }
```

NOW THAT WE HAVE WRITTEN TWO JOB (JOB1 AND JOB2)

IT IS TIME TO CREATE A NEW JOB CALLED AS JOB3:
WE WILL CALL JOB1 AND JOB2 OVER HERE

```
pipeline {                                    //indicate the job is written in Declarative Pipeline
   agent any                                  //agent specifies where the pipeline will execute.
   stages {
      stage ("JOB1-JOB2") {                   //an arbitrary stage name
         steps {
            build 'JOB1'     //this is where we specify which job to invoke.
            build 'JOB2'
         }
      }
   }
}
```

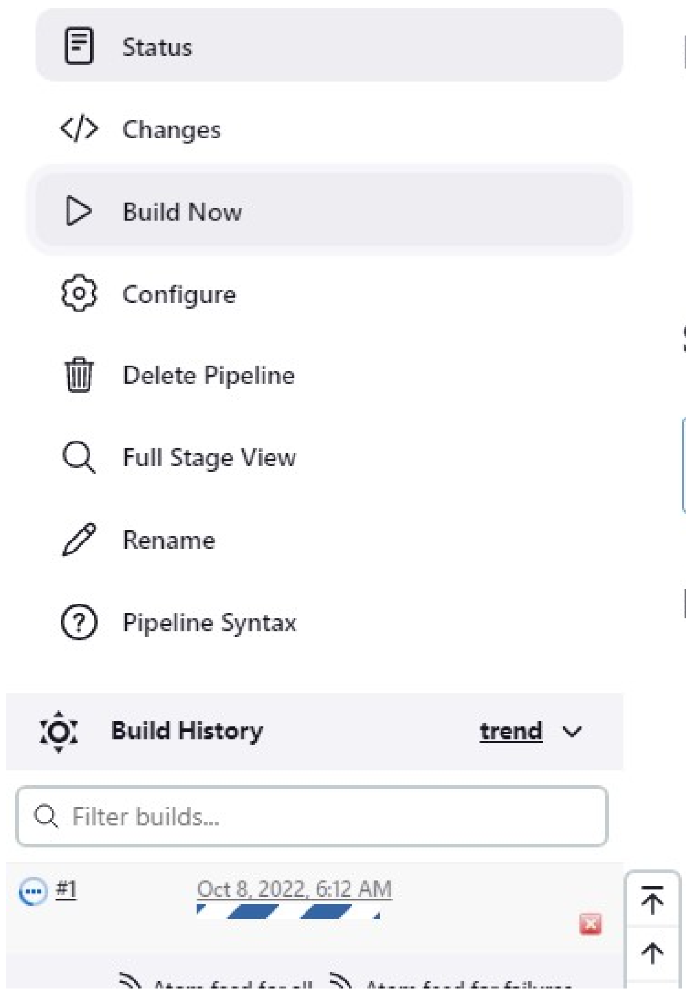


Script ?

```
pipeline {                //indicate the job is written in Declarative Pipeline
    agent any                //agent specifies where the pipeline will execute.
    stages {
        stage ("JOB1-JOB2") {       //an arbitrary stage name
            steps {
                build 'JOB1'    //this is where we specify which job to invoke.
                build 'JOB2'
            }
        }
    }
}
```

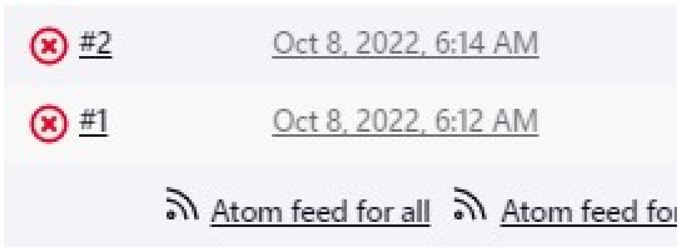
VERY IMP: PLEASE READ THE ENTIRE CODE ATLEAST 2-3 TIMES TO MAKE SURE IF YOU HAVE NOT MADE ANY MISTAKE.

STEP13) RUN JOB3 WHICH WIL TRIGGER JOB1 AND JOB2



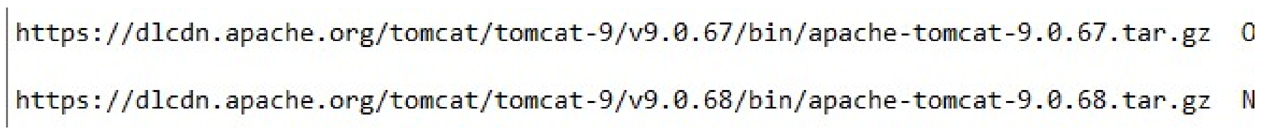
WE WILL WAIT FOR PIPELINE TO FINISH.

WE HAVE ENCOUNTERED BUILD FAIL: TROUBLESHOOT ACCORDINGLY



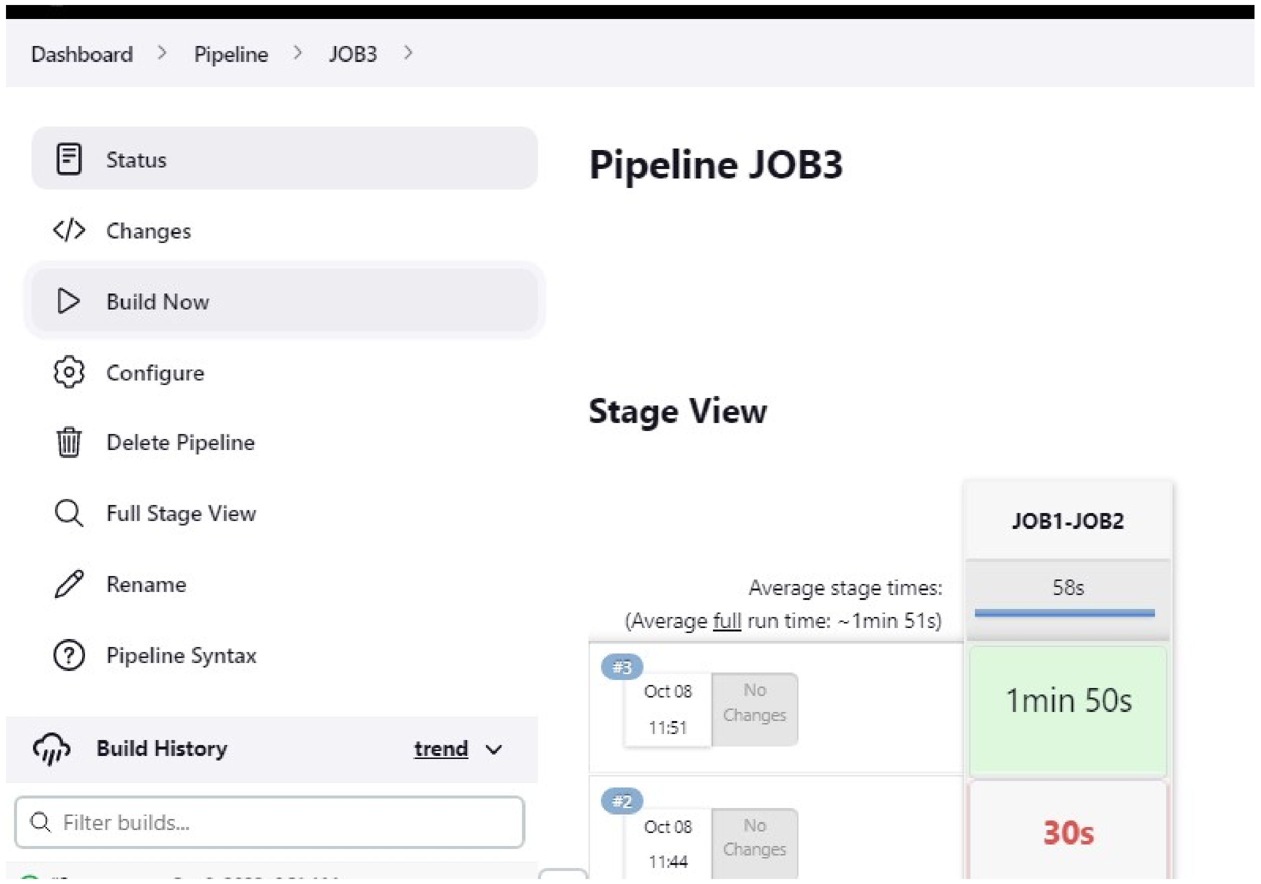
WE GOT ERROR DUR TO APACHE TOMCAT: APACHE TOMCAT HAS NOW RELEASE 9.68 VERSION AND WE ARE REFERRING TO LINK WHICH WAS OF 9.67

SOLVED: UPDATED THE LINK ON BOTH THE JOB (JOB1 AND JOB2)



```
https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.67/bin/apache-tomcat-9.0.67.tar.gz  O

https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.68/bin/apache-tomcat-9.0.68.tar.gz  N
```

LET'S CHECK THE STATUS OF OUR JOB3



Pipeline JOB3

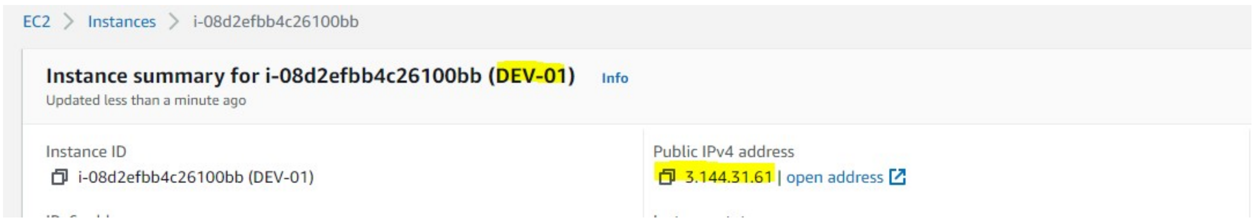
Stage View

	JOB1-JOB2
Average stage times: (Average full run time: ~1min 51s)	58s
#3 Oct 08 11:51 No Changes	1min 50s
#2 Oct 08 11:44 No Changes	30s

BUILD SUCCESS !!!

NOW LETS CHECK DEV1 ENV AND LETS SEE IF OUR GAMEOFLIFE IS RUNNING

DEV1 (OPEN THE PORTS YOU HAVE SPECIFIED ON DOCKERFILE AND CHECK)

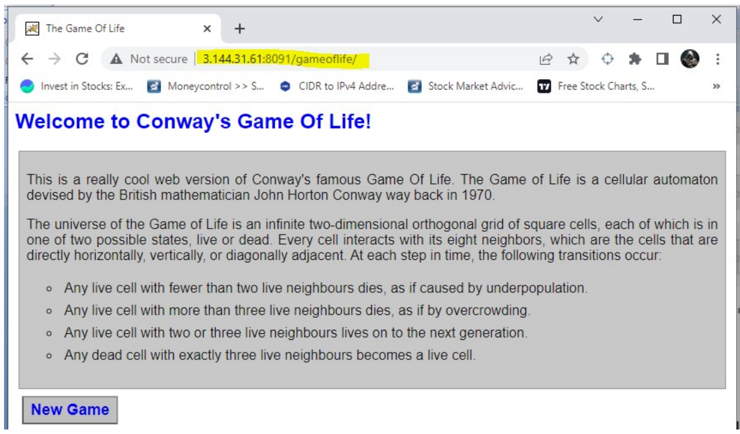


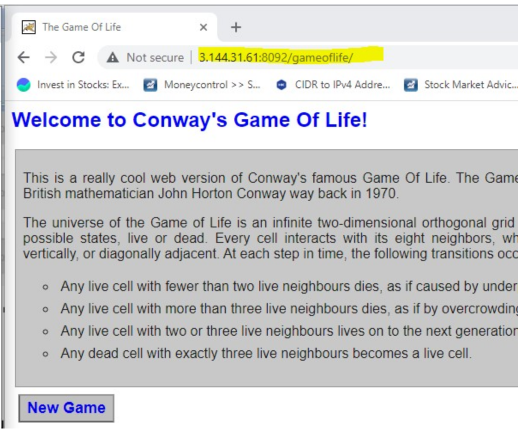
Instance summary for i-08d2efbb4c26100bb (DEV-01)

Updated less than a minute ago

Instance ID: i-08d2efbb4c26100bb (DEV-01)

Public IPv4 address: 3.144.31.61 | open address





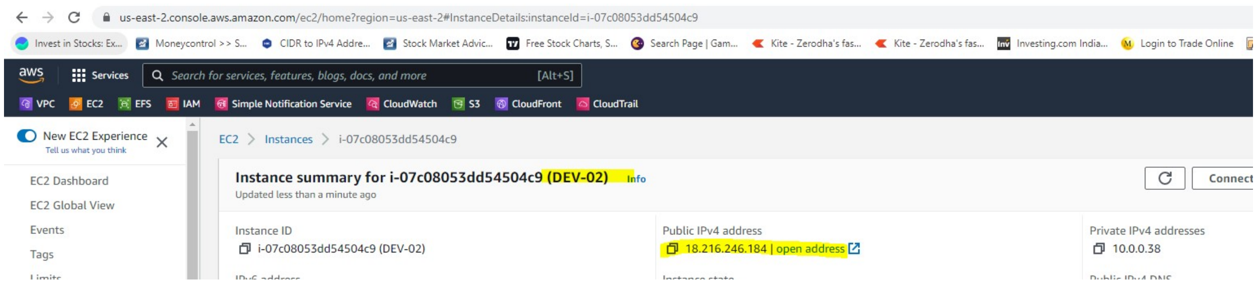
Welcome to Conway's Game Of Life!

This is a really cool web version of Conway's famous Game Of Life. The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway way back in 1970.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.

DEV2:



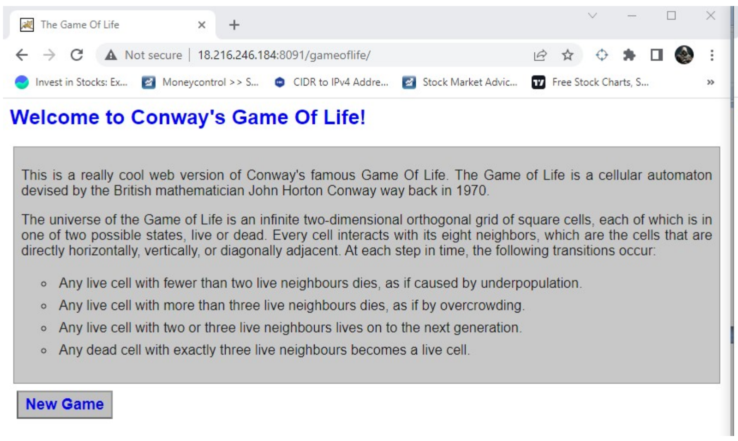
Instance summary for i-07c08053dd54504c9 (DEV-02)

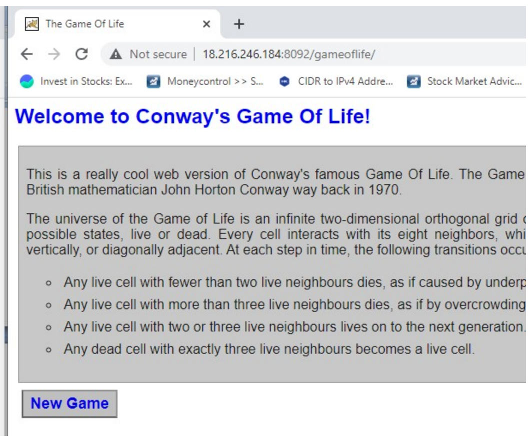
Updated less than a minute ago

Instance ID: i-07c08053dd54504c9 (DEV-02)

Public IPv4 address: 18.216.246.184 | open address

Private IPv4 addresses: 10.0.0.38





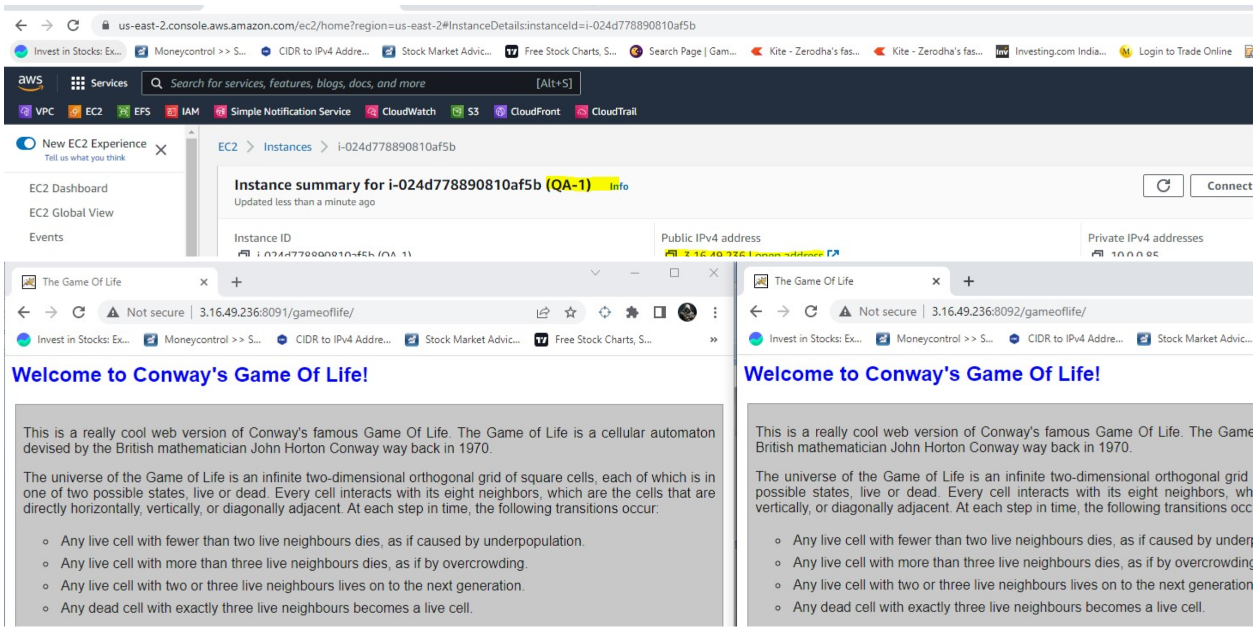
Welcome to Conway's Game Of Life!

This is a really cool web version of Conway's famous Game Of Life. The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway way back in 1970.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.

QA1


us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#InstanceDetails:instanceId=i-024d778890810af5b
Services
Search for services, features, blogs, docs, and more [Alt+S]
VPC EC2 EFS IAM Simple Notification Service CloudWatch S3 CloudFront CloudTrail
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
EC2 > Instances > i-024d778890810af5b
Connect
3.16.49.236:8091/gameoflife/
3.16.49.236:8092/gameoflife/

Instance summary for i-024d778890810af5b (QA-1) Info
Updated less than a minute ago

Instance ID | Public IPv4 address | Private IPv4 addresses

i-024d778890810af5b (QA-1) | 3.16.49.236 | open address | 10.0.0.85

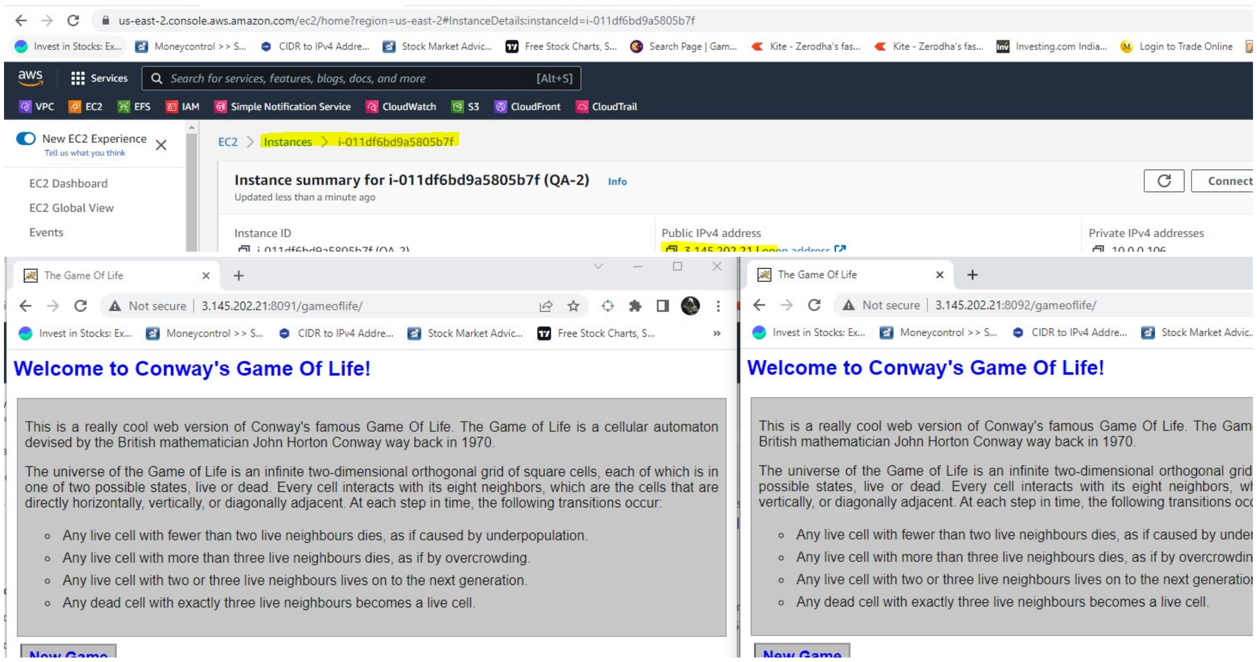
Welcome to Conway's Game Of Life!

This is a really cool web version of Conway's famous Game Of Life. The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway way back in 1970.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.

QA2


us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#InstanceDetails:instanceId=i-011df6bd9a5805b7f
Services
Search for services, features, blogs, docs, and more [Alt+S]
VPC EC2 EFS IAM Simple Notification Service CloudWatch S3 CloudFront CloudTrail
New EC2 Experience
Tell us what you think
EC2 Dashboard
EC2 Global View
Events
EC2 > Instances > i-011df6bd9a5805b7f
Connect
3.145.202.21:8091/gameoflife/
3.145.202.21:8092/gameoflife/
New Game

Instance summary for i-011df6bd9a5805b7f (QA-2) Info
Updated less than a minute ago

Instance ID | Public IPv4 address | Private IPv4 addresses

i-011df6bd9a5805b7f (QA-2) | 3.145.202.21 | open address | 10.0.0.106

Welcome to Conway's Game Of Life!

This is a really cool web version of Conway's famous Game Of Life. The Game of Life is a cellular automaton devised by the British mathematician John Horton Conway way back in 1970.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbors, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
- Any live cell with more than three live neighbours dies, as if by overcrowding.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any dead cell with exactly three live neighbours becomes a live cell.