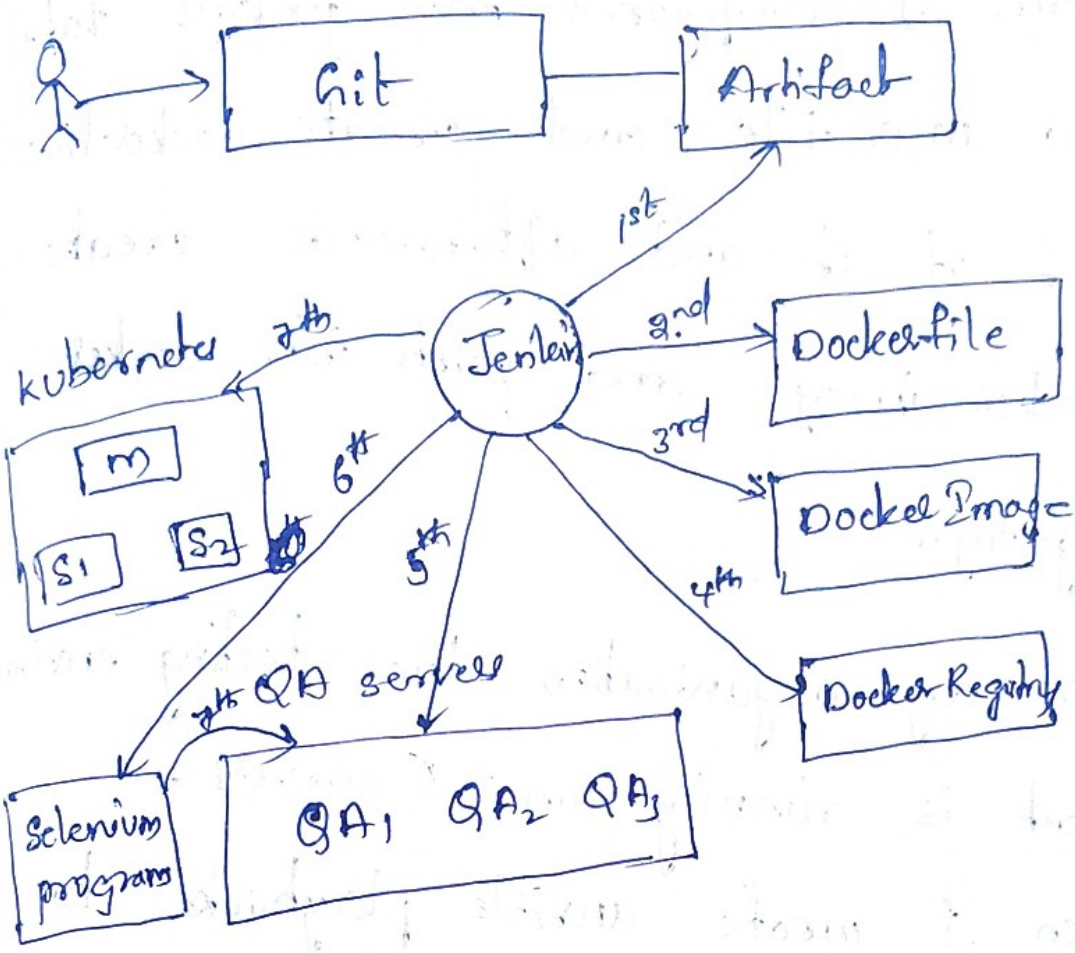
End-to-End Project

End-to-End Project



→ In my organisation, the developers are creating the code and pushing into the github

→ As a Devops engineer, I have configured my jenkins to download the code from github, it is java based application

→ Then with the help of maven, we created a package/artifact i.e war file. this is done by jenkins

→ Then I configured my jenkins take the war file and create dockerfile out of it. and afterwards create docker image and push into docker registry..

→ In my organisation the testing environment is running on 5 servers.

→ Then I wrote ansible playbooks to deploy that application into testing servers this is executed by jenkins.

→ And then I configured my jenkins to pickup the selenium programs written by testers. and deploy into QA environment for testing the application.

→ If the testing is succesful, I have configured my jenkins to deploy it into

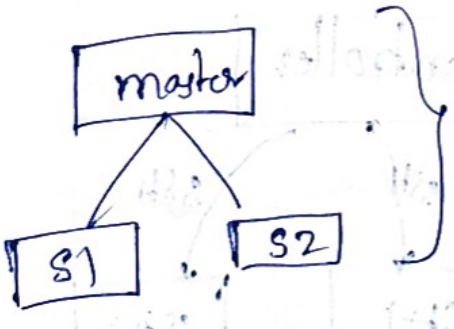
production servers which are running on kubernetes cluster. for that I have designed kubernetes definition files and service files.

Requirements :-

① Jenkins server

② For ansible configuration, we need controller,

③ kubernetes cluster



→ First create ansible controller and two QA servers, and establish password-less SSH connections and in

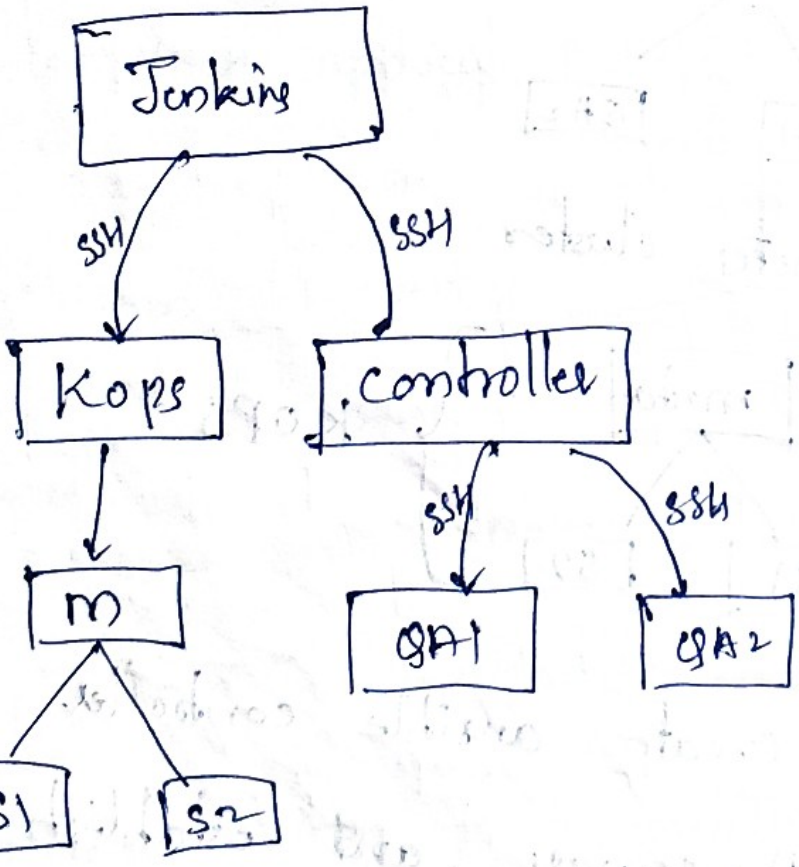
inventroy file save the ip address of QA1 and QA2.

→ Now setup kubernets cluster using KOPS,

→ Now from jenkins to controller make passwdless SSH connections

→ Now from jenkins to kubernetes manager make passwdless SSH connection

→



→ We require 3 servers, to do it.

→ Jenkins, controller, QA1, QA2 are ubuntu

→ For kops we take AWS Linux servers

Installations :-

① Jenkins

- sudo apt-get update
- openjdk-11-jdk
- git, maven
- jenkins

② Ansible → Controller

- sudo apt-get update
- software-properties-common
- sudo apt-add-repository ppa:ansible/ansible
- sudo apt-get update
- sudo apt-get install -y ansible
- edit inventory file

→ make SSH connections to jenkins to controller

③ KOPS → installation

step-1:- Create a declarative pipeline for continous download and continous build.

```
pipeline
{
  agent any
  stages
    {
      stage ('continous download')
        {
          steps
            {
              paste the url from github
            }
        }
      stage ('cont build into artifact')
        {
          steps
            {
              sh 'mvn package'
            }
        }
    }
}
```

step-II :- I have to create dockerfile

→ On jenkins server docker is installed. even though docker is installed jenkins user can't have permissions to access docker and run docker commands.

→ So to have permissions to jenkins, the jenkins user have to be added to docker group as follows:

```
sudo usermod -aG docker $USER
sudo systemctl restart jenkins
```

eg:-
```
sudo usermod -aG docker jenkins
```

→ To check whether it is added or not go into jenkins user and give docker commands

eg:-
→ docker login

→ To run our application we have to design a dockerfile

→ In docker file we have to copy the artifact into the tomcat, this is the requirement, for this we design docker file

→ We have workspace created by our jenkins, in that workspace our project files are there, but we have our artifact (webapp.war) in a folder called target,

→ Then copy the webapp.war file into our workspace, then only jenkins will perform actions on that file.

→ Automatically my jenkins should do copying copying from target folder into workspace

```
cp webapp/target/webapp.war .
```

It will copy that file into workspace.

→ Now for creating dockerfile using jenkins

```
cat > dockerfile << EOF
FROM tomee
COPY webapp.war /usr/local/tomee/webapps
                /testapp.war

EOF
```

→ We must copy into tomcat's webapp's folder to run our application.

→ convert all that code into groovy and paste

```
stage ('Build docker image')
 {
  steps
    {
    sh 'cp webapp/target/webapp.war .'
    sh ''' cat > dockerfile << EOF
    FROM tomee
    COPY webapp.war /usr/local/tomee/webapps/
                    testapp.war
    EOF'''
    sh ' docker build -t sudarshanswt/javaapp'
} } }
```

step-III:- Now upload the image into docker registry.

```
stage ('upload docker image into registry')
    {
     steps
        {
          sh 'docker push sudarshanswat/javaapp'
        }
    }
```

step-IV:- Now this image should be created as container in testing servers and tested with selenium programs.

→ For that we design ansible playbooks

→ Prior to this see that docker is installed on all the QA servers, if it is there just design playbooks for creation of container

→ Create a playbook to install docker on the testing environment and our ansible must be integrated with docker

```
---
- name: Install docker and required s/w's for
        ansible integration.
  hosts: all
  tasks:
    - name: Install pip3
      apt:
        name: python3-pip
        state: present
        update_cache: yes
    - name: Download the docker script and
            execute it and also install
            docker-py
      shell: "{{ item }}"
      with_items:
        - curl -fsSL https://get.docker.com
                -o install-docker.sh
        - sh install-docker.sh
        - pip3 install docker-py
...
```

vim javaapp.yml

```
---
- name : download the image and create
        container on all QA servers

  hosts : all
  tasks :
    - name : create container from the
                image
      docker_container :
          • name : myapp
            image : sudarshanswt/javaapp
            ports :
              - 8181 : 8080
...
```

```
stage ('Deploy the docker image into QA servers')
  {
   steps
     {
      sh ' ssh ubuntu@ip-add-of-controller
      ansible-playbook /home/ubuntu/myplaybooks/
       javaapp.yml : -b '
     }
  }
```

↓ path of the playbook

step-V! = Now download the selenium programs into QA server for testing

```
stage ('Functional testing')
    {
    steps
        {
        git 'Paste url of selenium programs
        sh 'java -jar /var/lib/jenkins/workspace
        /end-to-endProject/testing.jar'
        }
    }
```

step-VI! - Deploy that application into production environment,

→ production environment is running on kubernetes cluster.

→ We design definition files and service definition files in KOPS server.

vim javaapp.yml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
    name: java-app-deployment
    Labels:
        name: java-app
        author: intelliqit
Spec:
    replicas: 2
    selector:
        matchLabels:
            name: java-app
    template:
        metadata:
            name: java-app-pod
            labels:
                name: java-app
        Spec:
            containers:
              - name: my-java-app
                image: sudarshanswt/javaapp
...
```

vim javaappservice.yml

```
---
apiVersion: v1
kind: Service
metadata:
    name: java-app-service
    Labels:
        name: java-app
        author: intelliqit
Spec:
  type: LoadBalancer
  ports:
    - targetPort: 8080
      port: 8080
      nodePort: 30008
  selector:
      name: java-app
```

```
...
stage ('Deploy into K8s cluster-prod environment')
  {
  steps
    {
    sh 'ssh username@ip.add-of-kops server
    kubectl apply -f javaapp.yml'
  } }
```

End-to-end Project - Declarative pipeline code

```
pipeline
    {
    agent any
    stages
        {
        stage('continous download')
            {
            steps
                {
                paste the url from github
                }
            }
stage('Build into artifact')
    {
    steps
        {
        sh 'mvn package'
        }
    }
stage ('Build docker image')
    {
    steps
```

```
sh 'cp webapp/target/webapp.war .'

sh '''cat > dockerfile << EOF
FROM tomee
COPY webapp.war /usr/local/tomee/webapps/
                testapp.war

EOF'''
sh 'docker build -t sudarshanswj/javaapp'

  }

}
stage ('upload docker image into docker
        registry'
  {
  steps
    {
    sh 'docker push sudarshanswj/javaapp'

    }

  }
stage ('Deploy the docker image into OpenShift'
  {
  steps
```

```
sh' ssh username@ip-add -i controller
ansible-playbook /home/ubuntu/myplaybook/
javaapp.yml: -b'
}
}
stage ( 'Functional Testing")
{
steps
{
git 'Paste the url of selenium program
sh' java -jar /var/lib/jenkins/
. workspace/end-to-endProject/testing.jar'
}
}
stage ('Deploy in K8s cluster - prod environme
{
steps
{
```

sh 'ssh username@ ip-add -f KOPS_server
kubectl apply -f javaapp.yml'

}

}

)

Ansible docker Integrations

→ If I want to create containers on 'n' no. of servers, we depend on Ansible,

→ Because ansible is responsible creating any configuration on 'n' no. of servers.

→ To do all these things we need to integrate docker with Ansible, then you can execute docker commands in 'n' no of servers.

Requirements:-

(i) Docker must be installed on all the managed nodes.

(ii) Then pip must be installed (python3-pip)

(iii) docker-py

→ pip is python package management software, via pip we can install python based packages

→ docker-py is used to integrate docker with ansible.

→ Now install all the required s/w's for ansible docker integrations.

vim docker_ansible.yml

```
---
- name : Install all required s/w's
  hosts : all
  tasks :
    - name : Install pip s/w
      apt :
        name : python3-pip
        state : present
        update_cache : yes
    - name : Install docker and docker-py
      shell : "{{ item }}"
      with_items :
        - curl -fsSL https://get.docker.
          com -o install-docker.sh
        - sh install-docker.sh
        - pip3 install docker-py
...
```

→ In ansible we have a module to create containers i.e

docker_container

(*) Create a tomcat container on all the managed nodes

vim ansible_docker.yml

- name : Create a tomcat container
  hosts : all
  tasks :
    - name : Create tomcat
      docker_container :
        name : appserver
        image : tomee

(*) Now create a tomcat container, stop the tomcat container and delete the same container on all the managed node

vim ansible-docker.yml

```
---
- name: Create tomcat container, stop, delete.
  hosts: all
  tasks:
    - name: Create tomcat container
      docker_container:
        name: appserver
        image: tomee
    - name: stop the tomcat container.
      docker_container:
        name: appserver
        state: stopped
    - name: delete the tomcat container
      docker_container:
        name: appserver
        state: absent
```

...

with_sequence :-

→ It is used to create 'n' no. of containers on 'n' no. of servers.

eg:- To create 10 tomcat containers on 1000 servers, this type of scenarios we use with_sequence.

vim ansible_docker.yml

```
---
- name: Create 10 tomcat containers
  hosts: all
  tasks:
    - name: Create 10 tomcat container
      docker_container:
        image: tomee
        name: "container{{item}}"
      with_sequence: count=10
...
```

→ To delete all the containers

```
ansible all -m shell -a 'docker rm -f $(docker ps -aq)'
```

→ To delete all the images

```
ansible all -m shell -a 'docker system
prune -af'
```

→ To know how the arguments are used in the playbooks in docker container

go to

```
ansible-doc docker_container
```

```
ansible-doc docker_image
```

(*) Create a mysql container and link it with wordpress.

```
vim docker_ansible.yml

---
- name: Implementing docker compose using
        ansible
  hosts: all
  tasks:
    - name: Create mysql container
      docker_container:
        image: mysql:5
        name: mydb
```

```
    env:
      MYSQL_ROOT_PASSWORD: intellipaat
  - name: create a wordpress container
    docker_container:
      image: wordpress
      name: mywordpress
      Links:
        - mydb: mysql
      ports:
        - 8888: 80
...
```

(*) Create a jenkins container and (1) link it with qaserver and prodserver for implementing CI-CD.

vim ansible_CICD.yml

```
---
- name: Create jenkins container for CI-CD
    using ansible
  hosts: all
  tasks:
    - name: Create jenkins container
```

```
    docker_container:
        name: myjenkins
        image: jenkins/jenkins
        ports:
          - 5050:8080
- name: Create tomcat container for QA
  docker_container:
        name: qaserver
        image: tomcat
        ports:
          - 6060:8080
        links:
          - myjenkins:jenkins
- name: Create tomcat containe for prod
  docker_container:
        name: prodserve
        image: tomcat
        ports:
          - 7070:8080
        links:
          - myjenkins:jenkins
...
```

(*) Create a ubuntu container and mount volume on it.

```
---
- name: Create an ubuntu container and
        mount /data as a volume
  hosts: all
  tasks:
    - name: Create ubuntu container
      docker_container:
          name: myubuntu
          image: ubuntu
          interactive: true
          tty: true
          volumes:
            - /data
...
```

→ tty :- it is terminal

it → interactive terminal

└ interactive

tty → terminal

8 To download docker images

```
---
- name: download docker images
  hosts: all
  tasks:
    - name: download images from dockerhub
      docker_image:
        name: nginx
        source: pull
...
```

→ To pull the images, the module name is **docker_image**

Syntax:

```
docker_image:
    name:
    source: pull
```

(*) Download tomcat, nginx, ghost images

```
---
- name: Download tomcat, nginx, ghost
  hosts: all
  tasks:
    - name: download images from dockerhub
      docker_image:
        name: "{{item}}"
        source: pull
      with_items:
        - nginx
        - tomcat
        - ghost
...
```

Voting survey application:-

(*) It is developed with ReactJs

→ Build and deploy in the production servers

→ To build this we need "npm" package

step-1:-
clone the project code from github

step-2:-
→ Go inside the cloned project folder

step-3:- Create dockerfile
vim dockerfile in the same projectfolder

```
FROM node:14-alpine
WORKDIR /app
COPY package.json .
RUN npm install
COPY . .
~~EXPOSE~~ RUN npm run build
EXPOSE 3000
CMD ["npm", "start"]
```

docker ~~build~~ -it

docker build -t sudarshanswt/myapp .

docker run --name reactapp -d -p 3000:3000
sudarshanswt/myapp

Sonarqube

step-I :-

Create a docker container for Sonarqube

```
docker run --name mysonarqube -d -p
  8181:9000 sonarqube
```

step-II :-

On next server we install jenkins and configure (install Sonarqube Scanner, Sonarquality gate Plugin, Quality Gates Plugin)

step-III :-

→ Create token in sonarqube and use that in jenkins

step-IV :-

→ In Jenkins → Manage Jenkins → System →

Sonar Qebe server

Name [Enter name of Sonarqube server]

Server url :-

[http:// public ip of sonarqube : port no]

Server authentication token

→ for this click on +add

In that, in kind [secret text]

paste sonarqube token

click on add

step-V:

Manage jenkins → tools → SonarqubeScanner installations →

Name [Sonarqube-scanner]

☑ Install automatically

step-VI:

→ Write pipeline code,

→ We are writing declarative pipeline

```
pipeline
{
 agent any
 environment {
  // Define SonarQube server details
     SONARQUBE_TOKEN = credentials ('sonar')
                                    my token name
 }
 stage ('const download')
   {
     steps
        {
          git ' paste url from github '
        }
   }
 stage ('const build')
    {
      steps
          {
            sh ' mvn package '
          }
    }
stage ('build and SonarQube analysis')
    {
      steps
          {
            withSonarQubeEnv ('sonarqube')
              sh "mvn sonar:sonar"
          }
                                  Sonarqube installation
                                        name
```

```
stage ('Quality Gate')
    {
    steps
        {
        script
            {
            waitForQualityGate abortPipeline:
            false, credentialsId: 'sonar'
            }
        }
    }
}
```

1) terraform init :- It will initialize the the configuration.

2). terraform plan :- It is a dry run. It shows what exactly it is going to do.

3) terraform apply :- It will create infrastructure on the cloud.

4) terraform.tfstate :- It records everything about the infrastructure, what you have created

the life cycle of terraform

terraform init → terraform plan → terraform apply → terraform destroy

Create a VPC, Public subnet, Private subnet, launch two instances in both subnets. and connect from subnet-1 instance to subnet-2 instance, and create AMI of one instance without passwordless connections and launch in the subnets and connect from Public instance to Private instance.

→ VPC
→ subnets (Public, Private)
→ Internet gateway attach to Public subnet.
→ Route tables.
→ Now connect to Public instance.
→ From here connect to Private instance.
→ To connect Public instance we have a key pair in our downloads.
→ Now to connect from Public to private we must have key pair in

Public instance,

→ Now copy keypair from our local machine to Public subnet instance via

```
scp -i keypair source keypair ubuntu@
public-ip -q server:/home/ubuntu
```

→ chmod 400 key-pair

→ Now connect to private instance via

```
ssh -i "my-keypair" ubuntu@pvt-ip-q
@ instance
```

→ To build a python project using jenkins at build stage → select shell script from snippet generator

```
python3 file_name
```

eg: python3 app.py

→ For testing

sh 'python3 -m pytest'

→ To run above stage we must install

python3-pip

Now install pip install pytest

Git Conflicts

→ These conflicts arises when the developers are making some changes in the same file.

→ It happens when the redundancy of the code lines are present.

Solution :-

→ We open the file which is making conflict and sit with the developers and discuss ; which lines are important and which lines are to be removed. and do changes and commit and push.

Scenario :-

* Two developers are there, Dev 1 and Dev 2, the cloned the repository and made some changes and trying to push, but the conflict arises.

→ Dev1 cloned the repo and opened the file and made some changes.

→ first he cloned the repo

```
git clone url
```

→ Developers will not make changes directly on master branch, So he created a branch and made some changes, as below.

→ `git checkout -b newbranch`

```
vim file1

Hello

My name is Developer1

I am from Banglore

I like movies

:wq
```

→ Meanwhile Dev2 cloned the same repo and made changes on the master branch and pushed.

```
git clone url
```

vim file1

Hello

My name is Developer 2

I am from Delhi

I like movies.

:wq ↵

→ git add .

→ git commit -m "updated file1"

→ git push.

→ Now dev1 raised a pull request

```
git pull.
```

→ Now he got the new code with changes made by Developer2.

→ Now he is trying to merge "newbranch" with master.

`git merge newbranch`

→ Now it shows error as Merge conflict

→ Then he opened the file and saw the code

vim file1

```
Hello,
<<<<<<<< HEAD
My name is developer2
I am from Delhi
=======
My name is developer1
I am from Banglore
>>>>>>>> newbranch
I like movies.
```

→ There is a conflict between the two lines, they are

My name is developer2/developer1

I am from banglore/delhi

→ Now ask the developers and which lines has to be kept and which has to be removed, let say's remove developer2 and delhi

vim file1

Hello,

My name is developer1

I am from banglore.

I like movies

:wq!

→ git add .

→ git commit -m "updated file1"

→ git push.