DOCKER

```
pipeline {
    agent any
    stages{
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage ('checkout git repo'){
            steps{
                git branch: 'main', credentialsId: 'gitCred', url:
'https://github.com/Siddeshg672/hello_world_public_war.git'
            }
        }
        stage('sonar scan'){
            steps{
                withSonarQubeEnv('testSonar') {
                    sh "mvn clean install sonar:sonar -Dsonar.projectKey=testnew -Dsonar.url=http://54.88.244.241:9000"
                }
            }
        }
        stage('compile and build binaries'){
            steps{
                sh "mvn clean install"
                sh "cp -R webapp/target/webapp.war ."
            }
        }
        stage('create docker image'){
            steps{
                sh "docker images"
                sh "docker build -t app-image:${BUILD_NUMBER} -f Dockerfile ."
                sh "docker images"
            }
        }
        stage('upload to docker hub') {
            steps{
                withCredentials([usernamePassword(credentialsId: 'dockercred', passwordVariable: 'pass', usernameVariable:
'user')]) {
                    sh "docker login -u ${user} -p ${pass}"
                    sh "docker tag app-image:${BUILD_NUMBER} siddeshg672/app-image:${BUILD_NUMBER}"//To tag a local image
with ID //"0e5574283393" into the "fedora" repository with "version1.0"
                    sh "docker push siddeshg672/app-image:${BUILD_NUMBER}"
                    sh "docker rm -f devops-class"
                    sh "docker run -id --name devops-class -p 8090:8080 siddeshg672/app-image:${BUILD_NUMBER}"
                }
            }
        }
    }
}
```

TOMCAT

```
stage('compile and build binaries'){
            steps{
                sh "mvn clean install"
                sh "cp -R webapp/target/webapp.war ."
                sh "mv webapp.war webapp-${BUILD_NUMBER}.war"
                // sh "mvn validate"
            }
        }
        stage ('Upload file') {
            steps {
                sshagent(credentials: ['jfrog'], ignoreMissing: true) {
                    rtUpload (
                        // Obtain an Artifactory server instance, defined in Jenkins --> Manage Jenkins --> Configure
System:
                        serverId: "jfrogtest",
                        spec: """{
                                "files": [
                                        {
                                            "pattern": "webapp-${BUILD_NUMBER}.war",
                                            "target": "libs-snapshot-local"
                                        }
                                    ]
                                }"""
                    )
                }
            }
        }
        stage ('tomat_deploy') {
            steps {
                sshagent(credentials: ['tomcat_cred'], ignoreMissing: true) {
                // sh 'scp -o StrictHostKeyChecking=no webapp/target/webapp.war ubuntu@35.167.156.24:/opt/tomcat/webapps/'
                sh """
                  scp -o StrictHostKeyChecking=no webapp/target/webapp.war ubuntu@35.167.156.24:/home/ubuntu
                  ssh -o StrictHostKeyChecking=no ubuntu@35.167.156.24 'sudo cp -r /home/ubuntu/*.war
/opt/tomcat/webapps/'
                """

                }
            }
        }
    }
}
```

PLAYBOOK

```
---
- name: nginx webserver
  hosts: webserver
  become: yes
  become_user: root
  tasks:
  - name: Configure nginx server
    apt:
        name: nginx
        state: latest
  - name : start nginx
    service:
        name: nginx
        state: started
```

PLAYBOOK LOOP

```
---
- name: Installing 3 packages
  hosts: webserver
  become: yes
  tasks:
  - name: Install packages
    ansible.builtin.apt:
      name: "{item}"
      state: latest
      with items:
        - nginx
        - httpd
        - git
```

KUBERNETES DEPLOYMENT

```
apiVersion: app/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchlabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerport: 80
---
apiVersion: app/v1
kind: service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocal: TCP
      port: 8080
      targetport: 8080
```

KUBERENETES SECRET

```
apiVersion: app/v1
kind: secret
metadata:
  name: my-secret
type: opaque
data:
  user: asfds544646
  password: asfd=--=+4546

  echo -n 'user' | base64
```

KUBERENETES NODEPORT

```
apiVersion: app/v1
kind: service
metadata:
  name: mynodeport
spec:
  type: nodeport
  selector:
    app: nginx
  ports:
    - nodeport:
      port: 8080
      targetport: 80
```

DOCKER NGINX

```
FROM ubuntu
RUN apt-get -y update && ap-get install -y nginx
COPY default /etc/nginx/site-available/default
EXPOSE 80/tcp
CMD ["/usr/sbin/nginx","-g","deamon off;"]
```

DOCKER HTTPD

```
FROM ubuntu
RUN apt update
RUN apt install –y apache2
RUN apt install –y apache2-utils
RUN apt clean
EXPOSE 80
CMD [“apache2ctl”, “-D”, “FOREGROUND”]
```

DOCKER PYTHON

```
FROM python:3.9
WORKDIR /app
COPY src/requirments.txt ./
RUN pip install -r requirments.txt
COPY src /app
EXPOSE 8080
CMD ["python","server.py"]
```

DOCKER TOMCAT

```
FROM Ubuntu:20.8
RUN apt install -y java
RUN mkdir /opt/tomcat/
WORKDIR /opt/tomcat
RUN curl -O https://  /opt/tomcat
RUN tar xvfz apache*.tar.gz
RUN mv apache/* /opt/tomcat/
EXPOSE 8080
CMD ["/opt/tomcat/bin/catalina.sh", "run"]
```

LAMBDA FUNCTION

```
import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.start_instances(InstanceIds=instances)
    print('started your instances: ' + str(instances))

import boto3
region = 'us-west-1'
instances = ['i-12345cb6de4f78g9h', 'i-08ce9b2d7eccf6d26']
ec2 = boto3.client('ec2', region_name=region)
```

```
def lambda_handler(event, context):
    ec2.stop_instances(InstanceIds=instances)
    print('stopped your instances: ' + str(instances))
```