

You are hired as a DevOps Engineer for Analytics Pvt Ltd. This company is a product based organization which uses Docker for their containerization needs within the company. The final product received a lot of traction in the first few weeks of launch. Now with the increasing demand, the organization needs to have a platform for automating deployment, scaling and operations of application containers across clusters of hosts. As a DevOps Engineer, you need to implement a DevOps lifecycle such that all the requirements are implemented without any change in the Docker containers in the testing environment.

Up until now, this organization used to follow a monolithic architecture with just 2 developers. The product is present on: <https://github.com/hshar/website.git>

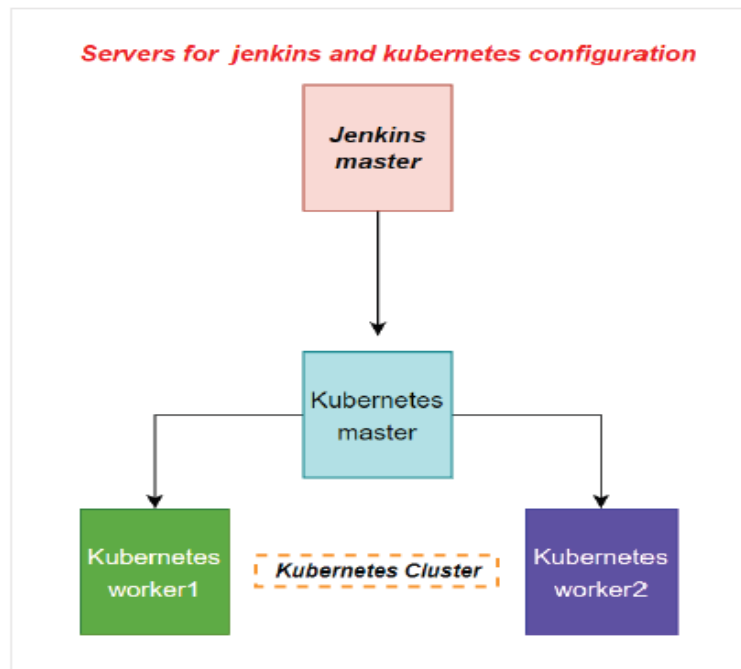
Following are the specifications of the lifecycle:

1. Git workflow should be implemented. Since the company follows a monolithic architecture of development, you need to take care of version control. The release should happen only on the 25th of every month.
2. CodeBuild should be triggered once the commits are made in the master branch.
3. The code should be containerized with the help of the Dockerfile. The Dockerfile should be built every time if there is a push to GitHub. Create a custom Docker image using a Dockerfile.
4. As per the requirement in the production server, you need to use the Kubernetes cluster and the containerized code from Docker Hub should be deployed with 2 replicas. Create a NodePort service and configure the same for port 30008.
5. Create a Jenkins Pipeline script to accomplish the above task.
6. For configuration management of the infrastructure, you need to deploy the configuration on the servers to install necessary software and configurations.
7. Using Terraform, accomplish the task of infrastructure creation in the AWS cloud provider.

Architectural Advice:

Softwares to be installed on the respective machines using configuration management.

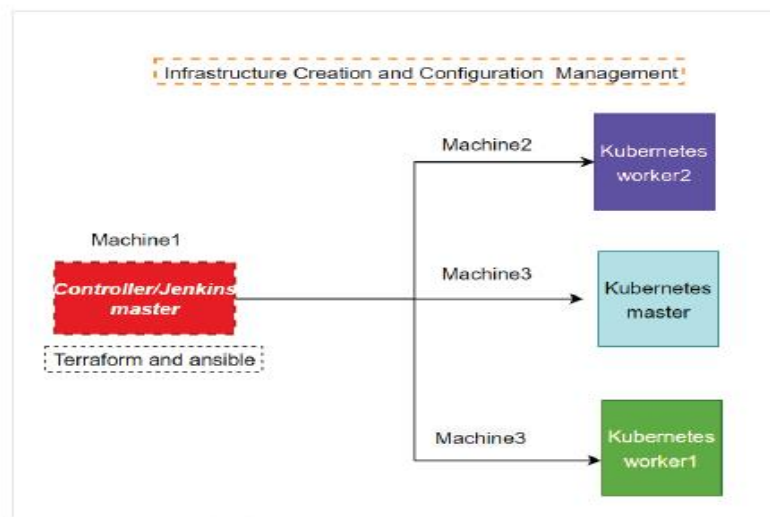
Worker1: Jenkins, Java



Worker2: Docker, Kubernetes

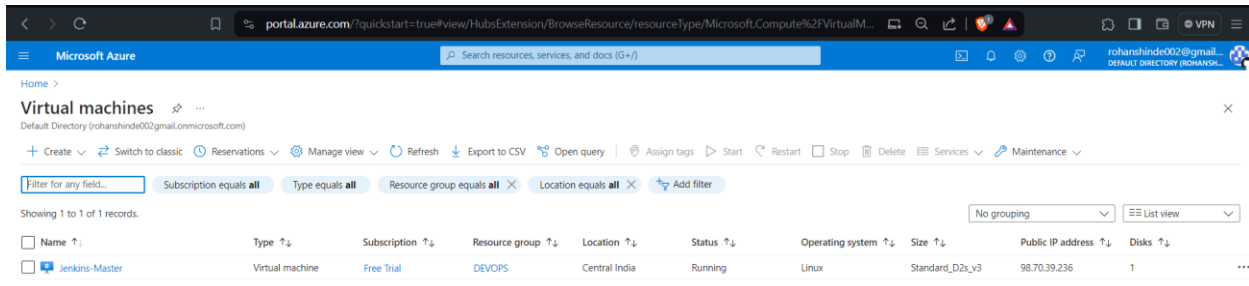
Worker3: Java, Docker, Kubernetes

Worker4: Docker, Kubernetes



SOLUTION

1. Created a Jenkins Master VM and installed terraform and ansible on it



2. Used terraform file for Infrastructure creation and ansible playbook to install java and docker

Main.tf file

```
# Configure the AWS provider
provider "aws" {
  region = "ap-south-1" # Update with your desired region
}

resource "aws_instance" "example" {
  ami = "ami-007020fd9c84e18c7"
  count = 2
  instance_type = "t2.medium"
  key_name = "demo"
  tags = {
    Name = "kubernetes-slave-${count.index}"
  }
  user_data = <<-EOF
```

```
#!/bin/bash
```

```
apt-get update
```

```
apt-get install -y openjdk-21-jdk
```

```
EOF
```

```
}
```

```
resource "aws_instance" "main" {
```

```
  ami = "ami-007020fd9c84e18c7"
```

```
  instance_type = "t2.medium"
```

```
  key_name = "demo"
```

```
  tags = {
```

```
    Name = "kubernetes-master"
```

```
}
```

```
  user_data = <<-EOF
```

```
#!/bin/bash
```

```
apt-get update
```

```
apt-get install -y openjdk-21-jdk
```

```
EOF
```

```
}
```

```
# Output public IP addresses for each instance
```

```
output "Master_Public_IP" {
```

```
  value = "${aws_instance.main.public_ip}"
```

```
}
```

```
output "Worker_public_IP_1" {
```

```
  value = "${aws_instance.example[0].public_ip}" # Access first worker
```

```
}
```

```
output "Worker_public_IP_2" {  
  value = "${aws_instance.example[1].public_ip}" # Access second worker  
}
```

Ansible Playbook

```
---  
  
- hosts: all  
  become: yes  
  tasks:  
    - name: Update apt cache  
      apt: update_cache=yes  
  
    - name: Install OpenJDK Java  
      apt: name={{ item }} state=present  
      with_items:  
        - openjdk-21-jdk # Specify the desired Java version  
  
    - name: Install required packages for Docker  
      apt: name={{ item }} state=present  
      with_items:  
        - apt-transport-https  
        - ca-certificates  
        - curl  
        - software-properties-common  
  
    - name: Add Docker's official GPG key  
      apt_key:
```

```
url: https://download.docker.com/linux/ubuntu/gpg
```

```
state: present
```

```
- name: Add Docker apt repository
```

```
apt_repository:
```

```
repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable
```

```
state: present
```

```
- name: Install Docker
```

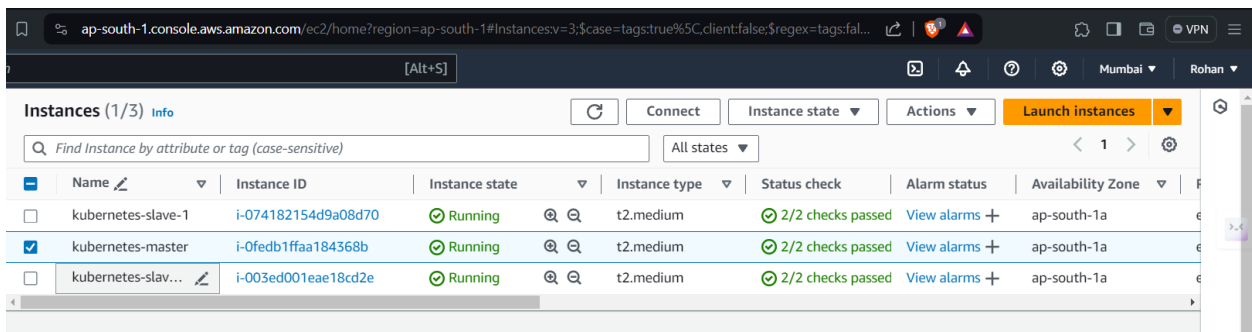
```
apt: name=docker-ce state=present
```

```
- name: Enable Docker service at startup
```

```
service: name=docker state=started enabled=yes
```

Installed kubernetes using .sh files which are in zip folder

Infrastructure created : -



The screenshot shows the AWS Management Console for the 'ap-south-1' region. The 'Instances' page is active, displaying a table of EC2 instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed: 'kubernetes-slave-1', 'kubernetes-master', and 'kubernetes-slave-2'. All three instances are in the 'Running' state and have '2/2 checks passed' for status checks. The 'kubernetes-master' instance is selected with a blue checkmark. The 'Launch instances' button is visible in the top right corner.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
kubernetes-slave-1	i-074182154d9a08d70	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a
kubernetes-master	i-0fedb1ffaa184368b	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a
kubernetes-slave-2	i-003ed001eae18cd2e	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1a

Groovy Script for Jenkins Declarative Pipeline:-

```
pipeline{
```

```
agent none
```

```
environment {
```

```
    DOCKERHUB_CREDENTIALS=credentials('dockercred')
```

```
  }
```

```
  stages{
```

```
    stage('Hello'){
```

```
      agent{
```

```
        label 'kube-master'
```

```
      }
```

```
      steps{
```

```
        echo 'Hello World'
```

```
      }
```

```
    }
```

```
    stage('Git'){
```

```
      agent{
```

```
        label 'kube-master'
```

```
      }
```

```
    steps{
```

```
      git 'https://github.com/sparkyx0022/Jenkins-Case-study.git'
```

```
    }
```

```
  }
```

```
  stage('Docker') {
```

```
    agent {
```

```
      label 'kube-master'
```

```
    }
```

```
  steps {
```

```
sh 'sudo docker build . -t ronn0022/devopsproject02'
```

```
sh 'sudo echo $DOCKERHUB_CREDENTIALS_PSW | sudo docker login -u  
$DOCKERHUB_CREDENTIALS_USR --password-stdin'
```

```
sh 'sudo docker push ronn0022/devopsproject02'
```

```
}
```

```
}
```

```
stage('Kubernetes') {
```

```
agent {
```

```
label 'kube-master'
```

```
}
```

```
steps {
```

```
sh "kubectl create -f deploy.yml"
```

```
sh "kubectl create -f svc.yml"
```

```
}
```

```
}
```

```
}
```

```
}
```


Pipeline view :-

Not secure | 98.70.39.236:8080/job/DevOps_Project_2/

Sorry, that page is missing. Do you want to check if a saved version is available on the Wayback Machine? ☐ Don't ask me again [Check for saved version](#)

Jenkins

Search (CTRL+K) [jenkins](#) [log out](#)

Dashboard > DevOps_Project_2 >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

✎ Rename

🔗 Pipeline Syntax

📅 Build History trend

Filter...

#59 Apr 9, 2024, 9:06 PM

DevOps_Project_2

Stage View

Average stage times:
(Average full run time: ~27s)

	Hello	Git	Docker	Kubernetes
#59 Apr 10 02:36 No Changes	10s	9s	11s	526ms
#58 Apr 10 02:33 No Changes	772ms	4s	17s	328ms
#57	15s	85ms	82ms	73ms
	aborted	aborted	aborted	aborted

Search (CTRL+K) [jenkins](#) [log out](#)

Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☁️	DevOps_Project_2	3 min 23 sec #59	6 min 38 sec #57	27 sec ▶

Icon: S M L [Icon legend](#) [Atom feed for all](#) [Atom feed for failures](#) [Atom feed for just latest builds](#)

After Successful Deployment :-

