

Project 01 – Medicure Web Application

Date Of Submission – 11-02-2025

Date Of Resubmission –

Submitted By – SAMEER A

Description of Medicure Web Application :-

- **Medicure** is a super specialty hospital based in New York, USA and provide world class treatment and surgery including Heart, Liver, Kidney transplants and first robotic surgery center. The chain is owned and managed by Global Health Limited.
- **Medicure** would centrally like to manage all the doctor's and patient's data across the Medicure hospitals in various cities. They have developed an micro service, which offers these services.

Resolution by Company :-

- In order to reduce unnecessary maintenance cost and manual labor, they would like to automate their application build and deployment process using DevOps. They are fine to use any one of the (AWS, Azure, GCP) cloud platform as their primary cloud service provider.

Problems Facing :-

1. Building Complex Monolithic Application is difficult.
2. Manual efforts to test various components/modules of the project.
3. Incremental builds are difficult to manage, test and deploy.
4. It was not possible to scale up individual modules independently.
5. Creation of infrastructure and configure it manually is very time consuming.
6. Continuous manual monitoring the application is quite challenging.

Company Expectations :-

Company Expecting to use the tools :-

1. **Git** - For version control for tracking changes in the code files.
2. **Jenkins** - For continuous integration and continuous deployment.
3. **Docker** - For containerizing applications.
4. **Ansible** - Configuration management tools.
5. **Terraform** - For creation of infrastructure.
6. **Kubernetes** - For running containerized application in managed cluster.

Plan :-

• Using Terraform :- (On AWS Cloud)

- Creating New VPC
- Creating 2 Subnets
- Creating 5 Security Groups
- Creating 2 Route Tables
- Creating 2 Route Tables Associations
- Creating 4 Elastic IPs
- Creating 7 Instances
- Associating Elastic IPs with instance

• Using Ansible :- (On AWS Cloud Servers)

- Configuration of Jenkins Master
- Configuration of Build Server (Maven)
- Configuration of Ansible Controller
- Configuration of Test-Server (Tomcat)
- Configuration of Production-Server (Kubernetes)

• Using Jenkins :-

- Creating a testing pipeline for build, deploy to test server, for testing the web applications
- Creating a production pipeline for build, deploy to production server deployment to main server

- **Using Docker :-**

- Creating and build a Docker image
- Publishing created Docker image to Docker Repository

- **Using Kubernetes :-**

- Pulling Docker image to Production server
- Creating Minimum 4 replicas

- **Using Prometheus & Grafana :-**

- Monitoring Test Server & Production-server

Execution :-

Step - 1.1 :- Creating an Isolated Environment in AWS Cloud Platform Using **Terraform**,

Prerequisite :-

- AWS Account with IAM User with Administrator Access,
- Access-key and Secret-key, to Access AWS using Terraform,

Step - 1.2 :- Creating a terraform script, Create a file “main.tf” and save the following script,

- Authentication and Configuration script

Provider Configuration

```
terraform {  
    required_providers {  
        aws = {  
            source = "hashicorp/aws"  
            version = "~> 5.0"  
        }  
    }  
}
```

Authentication Configuration

```
provider "aws" {
    region    = "ap-northeast-1"
    access_key = var.access_key
    secret_key = var.secret_key
}
```

variable to provide keys in runtime

```
variable "access_key" {
    description = "AWS Access Key"
}
variable "secret_key" {
    description = "AWS Secret Key"
}
```

- Creating a Isolated Environment On AWS Cloud

Creating VPC

```
resource "aws_vpc" "Project-Medicure-vpc" {
    cidr_block = "14.0.0.0/16"
    instance_tenancy = "default"
    tags = {
        Name = "Project-Medicure-VPC"
    }
}
```

- Creating a Subnets (Check the Availability Zone in the region)

Creating 2 Subnets

```
resource "aws_subnet" "Project-pub-sub-a1" {
    vpc_id    = aws_vpc.Project-Medicure-vpc.id
    cidr_block = "14.0.1.0/24"
    availability_zone = "ap-northeast-1a"
    map_public_ip_on_launch = true
    tags = {
        Name = "Project-pub-sub-a1"
    }
}

resource "aws_subnet" "Project-pub-sub-b1" {
    vpc_id    = aws_vpc.Project-Medicure-vpc.id
    cidr_block = "14.0.14.0/24"
    availability_zone = "ap-northeast-1d"
    map_public_ip_on_launch = true
    tags = {
        Name = "Project-pub-sub-b1"
    }
}
```

- Creating Internet Gateway to Access Servers

Creating 1 Internet Gateway

```
resource "aws_internet_gateway" "Project-igw" {
    vpc_id = aws_vpc.Project-Medicure-vpc.id
    tags = {
        Name = "Project-igw"
    }
}
```

- Creating Route Tables For Subnets

Creating 2 Route Tables

```
resource "aws_route_table" "Project-pub-rt1" {
    vpc_id = aws_vpc.Project-Medicure-vpc.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.Project-igw.id
    }
    tags = {
        Name = "Project-pub-rt1"
    }
}

resource "aws_route_table" "Project-pub-rt2" {
    vpc_id = aws_vpc.Project-Medicure-vpc.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.Project-igw.id
    }
    tags = {
        Name = "Project-pub-rt2"
    }
}
```

- Creating a Route Table Associations

Creating 2 Route Table Associations

```
resource "aws_route_table_association" "Project-pub-sub-a1-rt" {
    subnet_id = aws_subnet.Project-pub-sub-a1.id
    route_table_id = aws_route_table.Project-pub-rt1.id
}
```

```
resource "aws_route_table_association" "Project-pub-sub-b1-rt" {
    subnet_id = aws_subnet.Project-pub-sub-b1.id
    route_table_id = aws_route_table.Project-pub-rt2.id
}
```

- Creating a Security Groups For Servers

Creating 5 Security Groups

For Jenkins master node

```
resource "aws_security_group" "Project-SG-JM" {
    name = "Project-SG-JM"
    description = "Allow SSH traffic"
    vpc_id = aws_vpc.Project-vpc.id
    ingress {
        description = "SSH"
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 8080
        to_port = 8080
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
}
```

```
    tags = {
        Name = "Project-SG-JM"
    }
}
```

For Test server

```
resource "aws_security_group" "Project-SG-TS" {
    name = "Project-SG-TS"
    description = "Allow SSH traffic"
    vpc_id = aws_vpc.Project-vpc.id
    ingress {
        description = "SSH"
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 8080
        to_port = 8085
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 9100
        to_port = 9100
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 0
        to_port = 0
    }
}
```

```
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    tags = {
        Name = "Project-SG-TS"
    }
}
```

For kubernetes Master

```
resource "aws_security_group" "Project-SG-KM" {
    name = "Project-SG-KM"
    description = "Allow SSH traffic"
    vpc_id = aws_vpc.Project-vpc.id
    ingress {
        description = "SSH"
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 6443
        to_port = 6443
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 2379
        to_port = 2380
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
}
```

```

ingress {
    description = "HTTP"
    from_port = 10250
    to_port = 10259
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}
egress {
    from_port = 0
    to_port = 0
    protocol = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}
tags = {
    Name = "Project-SG-KM"
}
}

```

For kubernetes Worker Nodes

```

resource "aws_security_group" "Project-SG-KW" {
    name = "Project-SG-KW"
    description = "Allow SSH traffic"
    vpc_id = aws_vpc.Project-vpc.id
    ingress {
        description = "SSH"
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 10250
        to_port = 10256
    }
}

```

```

        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 30000
        to_port = 32767
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        description = "HTTP"
        from_port = 9100
        to_port = 9100
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    tags = {
        Name = "Project-SG-KW"
    }
}

```

Default to allow all traffic

```

resource "aws_security_group" "Project-SG-Default" {
    name = "Project-SG-Default"
    description = "Allow All traffic"
    vpc_id = aws_vpc.Project-vpc.id
    ingress {

```

```

        description = "All traffic"
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 0
        to_port = 0
        protocol = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    tags = {
        Name = "Project-SG-Default"
    }
}

```

- Creating Elastic IPs For Specific Servers

Creating 4 Elastic IPs

For Jenkins Master Node

```

resource "aws_eip" "Project-eip-JM" {
    associate_with_private_ip = true
    depends_on = [aws_internet_gateway.Project-igw]
    tags = {
        Name = "Project-eip-JM"
    }
}

```

For Test Server

```
resource "aws_eip" "Project-eip-TS" {
    associate_with_private_ip = true
    depends_on = [aws_internet_gateway.Project-igw]
    tags = {
        Name = "Project-eip-TS"
    }
}
```

For Kubernetes Worker Node 01

```
resource "aws_eip" "Project-eip-KW1" {
    associate_with_private_ip = true
    depends_on = [aws_internet_gateway.Project-igw]
    tags = {
        Name = "Project-eip-KW1"
    }
}
```

For Kubernetes Worker Node 02

```
resource "aws_eip" "Project-eip-KW2" {
    associate_with_private_ip = true
    depends_on = [aws_internet_gateway.Project-igw]
    tags = {
        Name = "Project-eip-KW2"
    }
}
```

- Create a Key Pair in AWS Cloud For Servers
- Creating Servers, with new user and installing and configuring Ansible Controller and All nodes,

Creating Key Pair

```
resource "tls_private_key" "skmirza_ssh_key" {  
  algorithm = "RSA"  
  rsa_bits = 4096  
}
```

Instance Creating

Ansible master node

```
resource "aws_instance" "Ansible-Master" {  
  ami          = "ami-08f191dd81ec3a3de"  
  instance_type = "t3.medium"  
  subnet_id    = aws_subnet.Project-pub-sub-a1.id  
  vpc_security_group_ids = [aws_security_group.Project-SG-  
  Default.id]  
  associate_public_ip_address = true  
  root_block_device {  
    volume_type = "gp3"  
    volume_size = 10  
  }  
  key_name = "Devops"  
  tags = {  
    Name = "Ansible-Master"  
  }  
  provisioner "remote-exec" {  
    connection {  
      type    = "ssh"  
      user    = "ubuntu"  
      private_key = file("/path/to/Devops.pem")  
      host    = self.public_ip  
    }  
    inline = [  
      "sudo -i apt update -y",  
      "sudo -i apt install software-properties-common",  
      "sudo -i add-apt-repository --yes --update ppa:ansible/ansible",  
    ]  
  }  
}
```

```

"sudo -i apt install ansible -y",
"sudo -i apt install openjdk-17-jre -y",
"sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
"echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
"echo 'PasswordAuthentication yes' | sudo -i tee -a
/etc/ssh/sshd_config",
"sudo service ssh reload",
"echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -a
/etc/sudoers",
"sudo -i chown -R skmirza:skmirza /etc/ansible",
"sudo -i mkdir -p /home/skmirza/.ssh",
"echo '${tls_private_key.skmirza_ssh_key.private_key_pem}' |
sudo -i tee /home/skmirza/.ssh/id_rsa",
"echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}' |
sudo -i tee -a /home/skmirza/.ssh/id_rsa.pub",
"echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}' |
sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
"sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
"sudo -i chmod 700 /home/skmirza/.ssh",
"sudo -i chmod 600 /home/skmirza/.ssh/id_rsa",
"sudo -i chmod 600 /home/skmirza/.ssh/id_rsa.pub",
"sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
]
}
}

```

- Copying Configure Files from Local Host to Ansible Controller,
- You can also create it on the Server

```

resource "null_resource" "Copy_Yaml_Files" {
  provisioner "local-exec" {
    command= "scp -o StrictHostKeyChecking=no -i
    /path/to/Medicure.pem /path/to/*.yaml ubuntu@${
    {aws_instance.Anible-Master.public_ip}:/home/ubuntu/"
  }
}
```

```

provisioner "remote-exec" {
  connection {
    type = "ssh"
    user = "ubuntu"
    private_key = file("/path/to/Medicure.pem")
    host = aws_instance.Ansible-Master.public_ip
  }
  inline = [
    "sudo -i mv /home/ubuntu/*.yaml /etc/ansible/",
    "sudo -i chown -R skmirza:skmirza /etc/ansible/*.yaml"
  ]
}

```

Jenkins master node

```

resource "aws_instance" "Jenkins-Master" {
  ami          = "ami-08f191dd81ec3a3de"
  instance_type = "t3.medium"
  subnet_id    = aws_subnet.Project-pub-sub-a1.id
  vpc_security_group_ids = [aws_security_group.Project-SG-JM.id]
  associate_public_ip_address = true
  root_block_device {
    volume_type = "gp3"
    volume_size = 10
  }
  key_name = "Devops"
  tags = {
    Name = "Jenkins-Master"
  }
  provisioner "remote-exec" {
    connection {
      type    = "ssh"
      user    = "ubuntu"
      private_key = file("/path/to/Devops.pem")
      host    = self.public_ip
    }
  }
}

```

```

    }

    inline = [
        "sudo -i apt update -y",
        "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
        "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
        "echo 'PasswordAuthentication yes' | sudo -i tee -a
        /etc/ssh/sshd_config",
        "sudo -i service ssh reload",
        "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -a
        /etc/sudoers",
        "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
        /home/skmirza/.ssh"',
        "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'
        | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
        "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
        "sudo -i chmod 700 /home/skmirza/.ssh",
        "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
    ]
}

}
}

```

Kubernetes master node

```

resource "aws_instance" "Kubernetes-Master" {
    ami          = "ami-08f191dd81ec3a3de"
    instance_type = "t3.medium"
    subnet_id    = aws_subnet.Project-pub-sub-a1.id
    vpc_security_group_ids = [aws_security_group.Project-SG-KM.id]
    associate_public_ip_address = true
    root_block_device {
        volume_type = "gp3"
        volume_size = 10
    }
    key_name = "Devops"
    tags = {
        Name = "Kubernetes-Master"
    }
}

```

```

}

provisioner "remote-exec" {
  connection {
    type    = "ssh"
    user    = "ubuntu"
    private_key = file("/path/to/Devops.pem")
    host    = self.public_ip
  }
  inline = [
    "sudo -i apt update -y",
    "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
    "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
    "echo 'PasswordAuthentication yes' | sudo -i tee -a
     /etc/ssh/sshd_config",
    "sudo -i service ssh reload",
    "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -
     a /etc/sudoers",
    "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
     /home/skmirza/.ssh"',
    "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'"
     | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
    "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
    "sudo -i chmod 700 /home/skmirza/.ssh",
    "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
  ]
}
}

```

Build Node Maven

```

resource "aws_instance" "Build-Node-mvn" {
  ami        = "ami-08f191dd81ec3a3de"
  instance_type = "t3.medium"
  subnet_id   = aws_subnet.Project-pub-sub-b1.id
  vpc_security_group_ids = [aws_security_group.Project-SG-
Default.id]

```

```

associate_public_ip_address = true
root_block_device {
    volume_type = "gp3"
    volume_size = 16
}
key_name = "Devops"
tags = {
    Name = "Build-Node-mvn"
}
provisioner "remote-exec" {
    connection {
        type    = "ssh"
        user    = "ubuntu"
        private_key = file("/path/to/Devops.pem")
        host    = self.public_ip
    }
    inline = [
        "sudo -i apt update -y",
        "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
        "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
        "echo 'PasswordAuthentication yes' | sudo -i tee -a
        /etc/ssh/sshd_config",
        "sudo -i service ssh reload",
        "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -
        a /etc/sudoers",
        "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
        /home/skmirza/.ssh"',
        "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'"
        | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
        "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
        "sudo -i chmod 700 /home/skmirza/.ssh",
        "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
    ]
}
}

```

Test server

```
resource "aws_instance" "Test-Server" {
    ami          = "ami-08f191dd81ec3a3de"
    instance_type = "t3.medium"
    subnet_id    = aws_subnet.Project-pub-sub-b1.id
    vpc_security_group_ids = [aws_security_group.Project-SG-TS.id]
    associate_public_ip_address = true
    root_block_device {
        volume_type = "gp3"
        volume_size = 16
    }
    key_name = "Devops"
    tags = {
        Name = "Test-Server"
    }
    provisioner "remote-exec" {
        connection {
            type    = "ssh"
            user    = "ubuntu"
            private_key = file("/path/to/Devops.pem")
            host    = self.public_ip
        }
        inline = [
            "sudo -i apt update -y",
            "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
            "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
            "echo 'PasswordAuthentication yes' | sudo -i tee -a
            /etc/ssh/sshd_config",
            "sudo -i service ssh reload",
            "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -
            a /etc/sudoers",
            "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
            /home/skmirza/.ssh'",
```

```

        "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'
        | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
        "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
        "sudo -i chmod 700 /home/skmirza/.ssh",
    "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
    ]
}
}

```

Kubernetes worker node 01

```

resource "aws_instance" "Kubernetes-Worker-01" {
    ami          = "ami-08f191dd81ec3a3de"
    instance_type = "t3.medium"
    subnet_id    = aws_subnet.Project-pub-sub-b1.id
    vpc_security_group_ids = [aws_security_group.Project-SG-KW.id]
    associate_public_ip_address = true
    root_block_device {
        volume_type = "gp3"
        volume_size = 10
    }
    key_name = "Devops"
    tags = {
        Name = "Kubernetes-Worker-01"
    }
    provisioner "remote-exec" {
        connection {
            type    = "ssh"
            user    = "ubuntu"
            private_key = file("/path/to/Devops.pem")
            host    = self.public_ip
        }
        inline = [
            "sudo -i apt update -y",
            "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
            "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",

```

```

    "echo 'PasswordAuthentication yes' | sudo -i tee -a
    /etc/ssh/sshd_config",
    "sudo -i service ssh reload",
    "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -
    a /etc/sudoers",
    "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
    /home/skmirza/.ssh",
    "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'
    | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
    "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
    "sudo -i chmod 700 /home/skmirza/.ssh",
    "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
]
}
}

```

Kubernetes worker node 02

```

resource "aws_instance" "Kubernetes-Worker-02" {
  ami          = "ami-08f191dd81ec3a3de"
  instance_type = "t3.medium"
  subnet_id    = aws_subnet.Project-pub-sub-b1.id
  vpc_security_group_ids = [aws_security_group.Project-SG-KW.id]
  associate_public_ip_address = true
  root_block_device {
    volume_type = "gp3"
    volume_size = 10
  }
  key_name = "Devops"
  tags = {
    Name = "Kubernetes-Worker-02"
  }
  provisioner "remote-exec" {
    connection {
      type    = "ssh"
      user    = "ubuntu"
    }
  }
}

```

```

private_key = file("/path/to/Devops.pem")
host      = self.public_ip
}
inline = [
    "sudo -i apt update -y",
    "sudo -i useradd skmirza -s /bin/bash -m -d /home/skmirza",
    "echo 'skmirza:YOURPASSWD' | sudo -i chpasswd",
    "echo 'PasswordAuthentication yes' | sudo -i tee -a
     /etc/ssh/sshd_config",
    "sudo -i service ssh reload",
    "echo 'skmirza ALL=(ALL) NOPASSWD: ALL' | sudo -i tee -
     a /etc/sudoers",
    "echo 'YOURPASSWD' | su - skmirza -c 'mkdir -p
     /home/skmirza/.ssh"',
    "echo '${tls_private_key.skmirza_ssh_key.public_key_openssh}'
     | sudo -i tee -a /home/skmirza/.ssh/authorized_keys",
    "sudo -i chown -R skmirza:skmirza /home/skmirza/.ssh",
    "sudo -i chmod 700 /home/skmirza/.ssh",
    "sudo -i chmod 600 /home/skmirza/.ssh/authorized_keys"
]
}
}

```

- **Adding all servers to ansible host file**

```

resource "null_resource" "Config_Ansible-Master" {
  provisioner "remote-exec" {
    connection {
      type    = "ssh"
      user    = "ubuntu"
      private_key = file("/path/to/Devops.pem")
      host      = aws_instance.Ansible-Master.public_ip
    }
    inline = [
      "echo '[JenkinsMaster]' | sudo -i tee -a /etc/ansible/hosts",
    ]
  }
}

```

```
"echo 'JenM ansible_ssh_host=${aws_instance.Jenkins-Master.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo " | sudo tee -a /etc/ansible/hosts",
"echo '[BuildNodes]' | sudo -i tee -a /etc/ansible/hosts",
"echo 'BN01 ansible_ssh_host=${aws_instance.Build-Node-mvn.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo " | sudo tee -a /etc/ansible/hosts",
"echo '[TestServers]' | sudo -i tee -a /etc/ansible/hosts",
"echo 'TCS ansible_ssh_host=${aws_instance.Test-Server.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo " | sudo tee -a /etc/ansible/hosts",
"echo '[ProdMaster]' | sudo -i tee -a /etc/ansible/hosts",
"echo 'PM ansible_ssh_host=${aws_instance.Kubernetes-Master.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo " | sudo tee -a /etc/ansible/hosts",
"echo '[ProdWorkers]' | sudo -i tee -a /etc/ansible/hosts",
"echo 'PW01 ansible_ssh_host=${aws_instance.Kubernetes-Worker-01.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo 'PW02 ansible_ssh_host=${aws_instance.Kubernetes-Worker-02.private_ip} ansible_ssh_user=skmirza' | sudo -i tee -a /etc/ansible/hosts",
"echo " | sudo -i tee -a /etc/ansible/hosts"
]
}
}
```

- Attaching Elastic IPs to servers

#Associate EIPs with instances

```
resource "aws_eip_association" "Project-eip-JM" {
    instance_id = aws_instance.Jenkins-Master.id
    allocation_id = aws_eip.Project-eip-JM.id
}

resource "aws_eip_association" "Project-eip-TS" {
    instance_id = aws_instance.Test-Server.id
    allocation_id = aws_eip.Project-eip-TS.id
}

resource "aws_eip_association" "Project-eip-KW1" {
    instance_id = aws_instance.Kubernetes-Worker-01.id
    allocation_id = aws_eip.Project-eip-KW1.id
}

resource "aws_eip_association" "Project-eip-KW2" {
    instance_id = aws_instance.Kubernetes-Worker-02.id
    allocation_id = aws_eip.Project-eip-KW2.id
}
```

- Copy and paste in the “main.tf” file and save it.
- Open it in terminal and run the following commands,
 - terraform init
 - terraform plan
 - terraform apply

```

Thu Feb 6 23:04:52
skmirza014@SkMirza-Ubuntu:~/Documents/USB/Project_Details/Project_Resources

+ dhcp_options_id           = (known after apply)
+ enable_dns_hostnames     = (known after apply)
+ enable_dns_support        = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                         = (known after apply)
+ instance_tenancy          = "default"
+ ipv6_association_id       = (known after apply)
+ ipv6_cidr_block           = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id       = (known after apply)
+ owner_id                  = (known after apply)
+ tags                      = {
    + "Name" = "Project-VPC"
}
+ tags_all                  = {
    + "Name" = "Project-VPC"
}

Plan: 28 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
skmirza014@SkMirza-Ubuntu:~/Documents/USB/Project_Details/Project_Resources$ terraform apply
var.access_key
Enter a value: A
var.secret_key
Enter a value: Ap

```

- In Run time you have to provide Access-key and Secret-key, 28 resources will created in your AWS Cloud,

```

Thu Feb 6 23:09:22
skmirza014@SkMirza-Ubuntu:~/Documents/USB/Project_Details/Project_Resources

aws_instance.Build-Node-mvn: Creating...
aws_instance.Kubernetes-Worker-02: Creating...
aws_instance.Test-Server: Creating...
aws_instance.Kubernetes-Worker-01: Creating...
aws_route_table_association.Project-pub-sub-a1-rt: Creation complete after 1s [id=rtbassoc-0e6cd948c7232ea54]
aws_route_table_association.Project-pub-sub-b1-rt: Creation complete after 0s [id=rtbassoc-073872dd06061deea]
aws_instance.Jenkins-Master: Still creating... [10s elapsed]
aws_instance.Ansible-Master: Still creating... [10s elapsed]
aws_instance.Kubernetes-Master: Still creating... [10s elapsed]
aws_instance.Build-Node-mvn: Still creating... [10s elapsed]
aws_instance.Kubernetes-Worker-02: Still creating... [10s elapsed]
aws_instance.Test-Server: Still creating... [10s elapsed]
aws_instance.Kubernetes-Worker-01: Still creating... [10s elapsed]
aws_instance.Jenkins-Master: Creation complete after 14s [id=i-07ef8a3e30d5f2ae5]
aws_eip_association.Project-eip-JM: Creating...
aws_instance.Ansible-Master: Creation complete after 14s [id=i-06e9c69ee214b6df9]
aws_instance.Kubernetes-Worker-01: Creation complete after 14s [id=i-0bfdad17faf526a05]
aws_instance.Test-Server: Creation complete after 14s [id=i-0b9039ffc7f5e5d4e]
aws_eip_association.Project-eip-KW1: Creating...
aws_eip_association.Project-eip-TS: Creating...
aws_instance.Build-Node-mvn: Creation complete after 14s [id=i-0d5eb8ef79ad40650]
aws_instance.Kubernetes-Master: Creation complete after 14s [id=i-0d179e45ae7d23337]
aws_instance.Kubernetes-Worker-02: Creation complete after 14s [id=i-0da265b1c4ac9be96]
aws_eip_association.Project-eip-KW2: Creating...
aws_eip_association.Project-eip-JM: Creation complete after 2s [id=eipassoc-0a295a064eab4070b]
aws_eip_association.Project-eip-TS: Creation complete after 1s [id=eipassoc-08c4479ffeb52f965]
aws_eip_association.Project-eip-KW1: Creation complete after 2s [id=eipassoc-013247c5d9086dfa2]
aws_eip_association.Project-eip-KW2: Creation complete after 2s [id=eipassoc-0d3bfb4f79a46796b]

Apply complete! Resources: 28 added, 0 changed, 0 destroyed.
skmirza014@SkMirza-Ubuntu:~/Documents/USB/Project_Details/Project_Resources$ 
```

- All resources were created successful, you can verify it in AWS Cloud,

VPC Details:

- VPC ID:** vpc-0539c96f5e845861c
- State:** Available
- DNS resolution:** Enabled
- Main network ACL:** acl-0d314b071441e6384
- IPv6 CIDR (Network border group):** -
- Default VPC:** No
- Network Address Usage metrics:** Disabled
- Block Public Access:** Off
- DHCP option set:** dopt-094745db56345665
- IPv4 CIDR:** 14.0.0.0/16
- Route 53 Resolver DNS Firewall rule groups:** -
- DNS hostnames:** Disabled
- Main route table:** rtb-017891a2c6e02b187
- IPv6 pool:** -
- Owner ID:** 339712990821

Resource Map:

- VPC:** Show details (Your AWS virtual network)
- Subnets (2):** Subnets within this VPC
 - ap-northeast-1a: Project-pub-sub-a1
 - ap-northeast-1d: Project-pub-sub-b1
- Route tables (3):** Route network traffic to resources
 - rtb-017891a2c6e02b187 (associated with Project-pub-rt2 and Project-pub-rt1)
 - Project-pub-rt2
 - Project-pub-rt1
- Network connections (1):** Connections to other networks
 - Project-igw

- Checking Creation of Servers,

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP	IPv6 IPs
Build-Node-mvn	i-0d5eb8ef79ad40650	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1d	-	3.115.12.27	-	-
Kubernetes-Worker-02	i-0da265b1c4ac9be06	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1d	-	18.176.86.169	16	-
Kubernetes-Master	i-0d179e45ae7d23337	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1d	-	35.74.234.245	-	-
Jenkins-Master	i-07ef8a3e30d5f2ae5	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1a	-	52.198.93.221	52	-
Test-Server	i-0b9039ff7c5e5d44	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1d	-	54.95.32.76	54	-
Ansible-Master	i-06e9c69ee214b6df9	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1a	-	54.199.247.91	-	-
Kubernetes-Worker-01	i-0bfdad17f5f526a05	Running	t3.medium	3/3 checks passed	View alarms +	ap-northeast-1d	-	57.182.150.153	57	-

Step – 2.1 :- Now Configuring All the server Using Ansible,

- You can verify the hosts added in the ansible hosts file,

The terminal window shows the following inventory configuration:

```

# Ex4: Multiple hosts arranged into groups such as 'Debian' and 'openSUSE':
## [Debian]
## alpha.example.org
## beta.example.org

## [openSUSE]
## green.example.com
## blue.example.com

[JenkinsMaster]
JenM ansible_ssh_host=192.168.1.101 ansible_ssh_user=skmirza

[BuildNodes]
BN01 ansible_ssh_host=192.168.1.102 ansible_ssh_user=skmirza

[TestServers]
TS ansible_ssh_host=192.168.1.103 ansible_ssh_user=skmirza

[ProdMaster]
PM ansible_ssh_host=192.168.1.104 ansible_ssh_user=skmirza

[ProdWorkers]
PW01 ansible_ssh_host=192.168.1.105 ansible_ssh_user=skmirza
PW02 ansible_ssh_host=192.168.1.106 ansible_ssh_user=skmirza

```

Hosts are grouped into JenkinsMaster, BuildNodes, TestServers, ProdMaster, and ProdWorkers. Each host has its IP address and user specified.

- And try to ping the servers by, “ansible all -m ping”

The terminal window shows the output of the command "ansible all -m ping":

```

Fri Feb 7 09:54:33

skmirza@ip-14-0-1-94:/etc/ansible$ ansible all -m ping
[WARNING]: Platform linux on host KH is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
KH | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host KX is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
KX | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host TS is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
TS | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host BN01 is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
BN01 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}
[WARNING]: Platform linux on host PW02 is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
PW02 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3.10"
    },
    "changed": false,
    "ping": "pong"
}

skmirza@ip-14-0-1-94:/etc/ansible$ 

```

The output shows that all hosts responded with "pong" to the ping command.

- All the servers connected to ansible master,

Step – 2.2 :- Now Configuring Jenkins-Master Node Using Ansible,

- Create a “jenkins-master-config.yaml” file and paste the following script,

```

# yaml file to configure the Jenkins Master using ansible
---
- hosts: JenM # same name as in hosts file
  gather_facts: no
  become: yes
  tasks:
    - name: Update Packages
      shell: sudo apt update -y
    - name: Install Java
      apt:
        name: openjdk-17-jre
        state: present
    - name: Verify Java installation
      shell: java --version
      register: java_version
    - name: Print Java version
      debug:
        msg: "Java Version is: {{ java_version.stdout }}"
    - name: Installing Jenkins
      shell:
        cmd: |
          sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
            https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
          echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
            https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
            /etc/apt/sources.list.d/jenkins.list > /dev/null
          sudo apt-get update
          sudo apt-get install jenkins -y
    - name: Verify Jenkins installation
      shell: jenkins --version
      register: jenkins_installation
    - name: Start and enable Jenkins service
      systemd:
        name: jenkins
        state: started
        enabled: yes
        daemon_reload: yes
    - name: Get Jenkins Status
      shell: systemctl status jenkins
      register: jenkins_status
    - name: Print Jenkins Status
      debug:

```

```

msg: "{{ jenkins_status.stdout }}"
- name: Get Jenkins initial admin password
  shell: cat /var/lib/jenkins/secrets/initialAdminPassword
  register: jenkins_password
- name: Print Jenkins initial admin password
  debug:
    msg: "Jenkins initial admin password: {{ jenkins_password.stdout }}"

```

```

Sun Feb 9 11:12:22
skmirza@ip-14-0-1-84:/etc/ansible
skmirza@ip-14-0-1-84:/etc/ansible$ nano jenkins-master-config.yaml
skmirza@ip-14-0-1-84:/etc/ansible$ cat jenkins-master-config.yaml
# yaml file to configure the Jenkins Master using ansible
...
- hosts: JenM
  gather_facts: no
  become: yes
  tasks:
    - name: Install Java
      apt:
        name: openjdk-17-jre
        state: present
    - name: Verify Java installation
      shell: java --version
      register: java_version
    - name: Print Java version
      debug:
        msg: "Java Version is: {{ java_version.stdout }}"
    - name: Installing Jenkins
      shell:
        cmd: |
          sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
            https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
          echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] " \
            https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
            /etc/apt/sources.list.d/jenkins.list > /dev/null
          sudo apt-get update
          sudo apt-get install jenkins -y
    - name: Verify Jenkins installation
      shell:
        cmd: |
          curl -s http://jenkins:8080

```

- Now execute “ ansible-playbook jenkins-master-config.yaml ”

```

Sun Feb 9 11:24:23
skmirza@ip-14-0-1-84:/etc/ansible
skmirza@ip-14-0-1-84:/etc/ansible$ ansible-playbook jenkins-master-config.yaml
[...]
ok: [JenM]
changed: [JenM]

TASK [Print Jenkins Status] *****
ok: [JenM] => [
  {
    "msg": "● jenkins.service - Jenkins Continuous Integration Server\n  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)\n  Active: active (running) since Sun 2025-02-09 05:54:02 UTC; 6s ago\n    Main PID: 6232 (java)\n      Tasks: 52 (limit: 4586)\n     Memory: 605.4M\n        CPU: 27.524s\n       Group: /system.slice/jenkins.service\n          └─6232 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --http-port=8080\nFeb 09 05:53:53 ip-14-0-1-69 jenkins[6232]: fdbbabcd45749e8942eecdada5378ad\nFeb 09 05:53:53 ip-14-0-1-69 jenkins[6232]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword\nFeb 09 05:53:53 ip-14-0-1-69 jenkins[6232]: *****\nFeb 09 05:53:53 ip-14-0-1-69 jenkins[6232]: *****\nFeb 09 05:54:02 ip-14-0-1-69 jenkins[6232]: 2025-02-09 05:54:02.814+0000 [id=32]      INFO      jenkins.InitReactorRunner$#onAttai ned: Completed initialization\nFeb 09 05:54:02 ip-14-0-1-69 jenkins[6232]: 2025-02-09 05:54:02.841+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle$#onReady: Jenkins is fully up and running\nFeb 09 05:54:02 ip-14-0-1-69 systemd[1]: Started Jenkins Continuous Integration Server.\nFeb 09 05:54:04 ip-14-0-1-69 jenkins[6232]: 2025-02-09 05:54:04.366+0000 [id=48]      INFO      hudson.util.Retrier$#start: Performed the action check updates server successfully at the attempt #1"
}

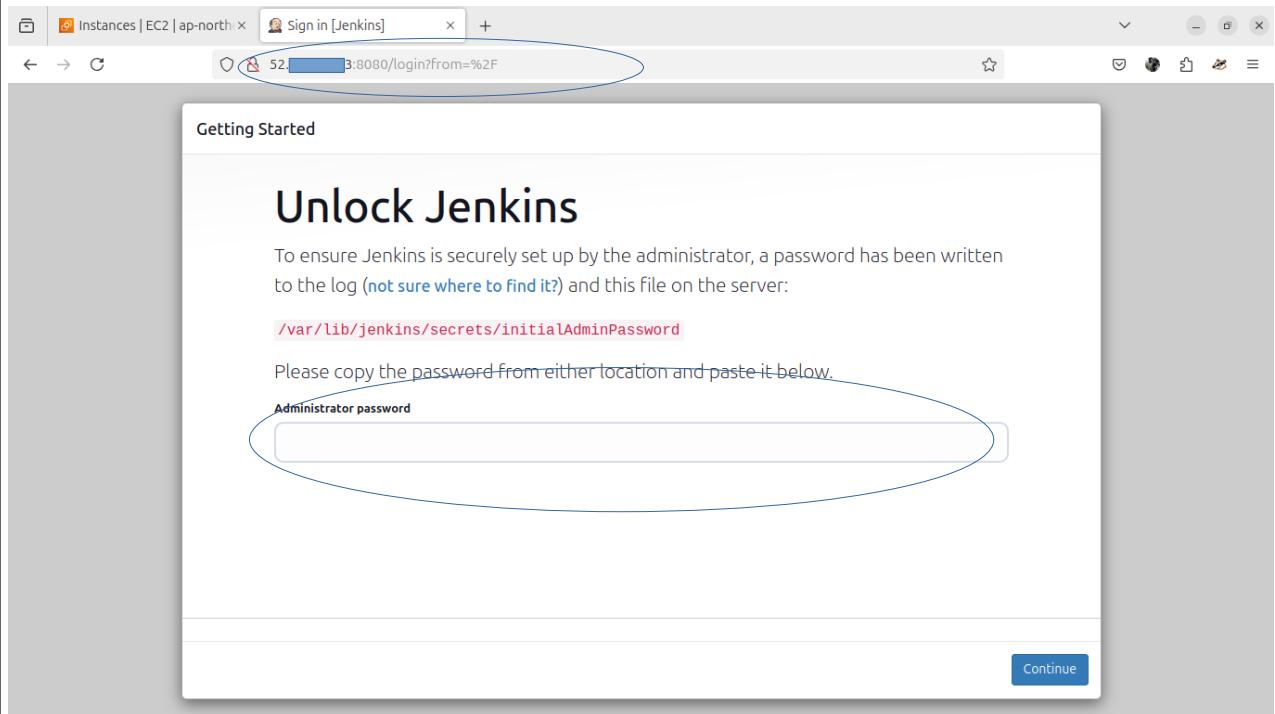
TASK [Get Jenkins initial admin password] *****
changed: [JenM]

TASK [Print Jenkins initial_admin_password] *****
ok: [JenM] => [
  {
    "msg": "Jenkins initial admin password: fdbbabcd45749e8942eecdada5378ad"
  }
]

PLAY RECAP *****
JenM : ok=11   changed=7   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

```

- Jenkins is installed successful in Jenkins-Master Node,
- To verify it, “ <http://<jenkins-public-ip>:8080> ”



- paste the jenkins initial admin password, and config it, install all required pulgins,

Step – 2.3 :- Now Configuring Build-Server Node Using Ansible,

- Create a “build-servers-config.yaml” file and paste the following script,

yaml file to configure the Build Server using ansible

```
---
- hosts: BN01
  gather_facts: no
  become: yes
  tasks:
    - name: Update Packages
      shell: sudo apt update -y
    - name: Install the required packages
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - openjdk-17-jre
        - git
```

```

- maven
- gradle

- name: Installation of Docker
  shell:
    cmd: |
      sudo apt-get update
      sudo apt-get install ca-certificates curl
      sudo install -m 0755 -d /etc/apt/keyrings
      sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
      sudo chmod a+r /etc/apt/keyrings/docker.asc
      echo \
        "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
      $(./etc/os-release && echo "${UBUNTU_CODENAME}:-$VERSION_CODENAME}") stable" | \
      sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
      sudo apt-get update
      sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin
- name: Adding user to docker group
  shell: usermod -aG docker skmirza
- name: Verify Installations
  shell: java --version && git --version && docker --version && mvn --
version && gradle --version
  register: builds_version
- name: Print Java, Git, Docker, Maven & Gradle Version
  debug:
    msg: "Java, Git, Docker, Maven & Gradle Version is:
{{ builds_version.stdout }}"

```

```

Sun Feb 9 11:42:05
skmirza@ip-14-0-1-84:/etc/ansible

skmirza@ip-14-0-1-84:~/Documents/USB/Project_Details/Project_Resources/T... x skmirza@ip-14-0-1-84:/etc/ansible
skmirza@ip-14-0-1-84:/etc/ansible$ nano build-servers-config.yaml
skmirza@ip-14-0-1-84:/etc/ansible$ cat build-servers-config.yaml
---
- hosts: BN01
  gather_facts: no
  become: yes
  tasks:
    - name: Install the required packages
      apt:
        name: "{{ item }}"
        state: present
      with_items:
        - openjdk-17-jre
        - git
        - maven
        - gradle

    - name: Installation of Docker
      shell:
        cmd:
          - sudo apt-get update
          - sudo apt-get install ca-certificates curl
          - sudo install -m 0755 -d /etc/apt/keyrings
          - sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
          - sudo chmod a+r /etc/apt/keyrings/docker.asc
          - echo \
            "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
            ${. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}" stable}" | \
          - sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
          - sudo apt-get update
          - sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
    - name: Adding user to docker group
      shell: usermod -G docker skmirza
    - name: Verify Installations
      shell: java --version && git --version && docker --version && mvn --version && gradle --version
      register: builds_version

```

- Now execute “ ansible-playbook build-servers-config.yaml ”

```

Sun Feb 9 11:47:56
skmirza@ip-14-0-1-84:/etc/ansible

skmirza@ip-14-0-1-84:~/Documents/USB/Project_Details/Project_Resources/T... x skmirza@ip-14-0-1-84:/etc/ansible
skmirza@ip-14-0-1-84:/etc/ansible$ ansible-playbook build-servers-config.yaml
[WARNING]: Platform linux on host BN01 is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.

TASK [Update Packages] *****
[WARNING]: Platform linux on host BN01 is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.

changed: [BN01]

TASK [Install the required packages] *****
changed: [BN01] => (item=openjdk-17-jre)
ok: [BN01] => (item=git)
changed: [BN01] => (item=maven)
changed: [BN01] => (item=gradle)

TASK [Installation of Docker] *****
changed: [BN01]

TASK [Adding user to docker group] *****
changed: [BN01]

TASK [Verify Installations] *****
changed: [BN01]

TASK [Print Java, Git, Docker, Maven & Gradle Version] *****
ok: [BN01] => {
  "msg": "Java, Git, Docker, Maven & Gradle Version is: openjdk 17.0.14 2025-01-21\\nOpenJDK Runtime Environment (build 17.0.14+7-Ubuntu-122.04.1)\\nOpenJDK 64-Bit Server VM (build 17.0.14+7-Ubuntu-122.04.1, mixed mode, sharing)\\ngit version 2.34.1\\ndocker version 27.5.1, build 9f9e405\\n\\u001b[1mApache Maven 3.6.3\\u001b[m\\nhaven-home: /usr/share/maven\\nJava version: 17.0.14, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64\\nDefault locale: en, platform encoding: UTF-8\\noS name: 'linux'\\nversion: '6.8.0-1021-aws', arch: 'amd64', family: 'unix'\\n\\n-----\\nBuild time: 2012-12-21 00:00:00 UTC\\nRevision: none\\nGroovy: 2.4.21\\nAnt: Apache Ant(TM) version 1.10.12 compiled on January 17 1970\\nJVM: 17.0.14 (Ubuntu 17.0.14+7-Ubuntu-122.04.1)\\nOS: Linux 6.8.0-1021-aws amd64"
}

PLAY RECAP *****
BN01 : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

skmirza@ip-14-0-1-84:/etc/ansible$ 

```

- Build tools successful installed in Build Servers,

Step – 2.4 :- Now Configuring Kubernetes-Master Node Using Ansible,

- Create a “kubernetes-master-config.yaml” file and paste the following script,

```

# yaml file to configure the kubernetes Master using ansible
---
- hosts: PM # same name as in hosts file
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Installing and Configuring kubernetes
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-master"

    - name: Turning off swap
      shell:
        cmd: |
          sudo swapoff -a
          sudo sed -i '/ swap / s/^\\(.*)$/#\\1/g' /etc/fstab

    - name: Installing the required packages
      shell:
        cmd: |
          sudo apt-get update
          sudo apt-get install -y ca-certificates curl
          sudo install -m 0755 -d /etc/apt/keyrings
          sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
          sudo chmod a+r /etc/apt/keyrings/docker.asc
          echo \
            "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$ (. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \
          sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
          sudo apt-get update

    - name: Installing Docker
      shell:
        cmd: |
          sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

```

```
- name: Configuring Docker
  shell:
    cmd: |
      sudo modprobe overlay
      sudo modprobe br_netfilter
      cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
      overlay
      br_netfilter
      EOF

- name: Configuring kubernetes
  shell:
    cmd: |
      sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
      net.bridge.bridge-nf-call-ip6tables = 1
      net.bridge.bridge-nf-call-iptables = 1
      net.ipv4.ip_forward = 1
      EOF

- name: Configuring containerd
  shell:
    cmd: |
      sudo sysctl --system
      sudo mkdir -p /etc/containerd
      sudo containerd config default | sudo tee
/etc/containerd/config.toml

- name: Config containerd
  shell: sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/'  

/etc/containerd/config.toml

- name: Restarting containerd
  shell: sudo systemctl restart containerd

- name: Installing pre-requisites dependencies
  shell:
    cmd: |
      echo "Starting installation of pre-requisites"
      sudo apt-get update
      sudo apt-get install -y apt-transport-https ca-certificates curl gpg
- name: Adding Kubernetes repository
  shell:
```

```

cmd: |
  sudo mkdir -p -m 755 /etc/apt/keyrings
  echo "Downloading Kubernetes release key"
  curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | 
sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
  echo "Adding Kubernetes repository"
  echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-
keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.32/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list

- name: Installing kubeadm, kubelet and kubectl
  shell:
    cmd: |
      sudo apt-get update -y
      sudo apt-get install -y kubelet kubeadm kubectl
      sudo apt-mark hold kubelet kubeadm kubectl
      sudo systemctl enable kubelet

- name: Pulling the required images
  shell:
    cmd: |
      sudo kubeadm config images pull

- name: kubeadm initialization
  shell:
    cmd: |
      sudo kubeadm init --pod-network-cidr=10.244.0.0/16
  register: kubeadm_init
- name: Printing the join command
  debug:
    msg: "{{ kubeadm_init.stdout_lines }}"

- name: Creating the .kube directory
  shell:
    cmd: |
      mkdir -p $HOME/.kube
      sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
      sudo chown $(id -u):$(id -g) $HOME/.kube/config
      export KUBECONFIG=/etc/kubernetes/admin.conf

- name: Installing the flannel network
  shell:
    cmd: |

```

```

        kubectl apply -f
https://github.com/coreos/flannel/raw/master/Documentation/kube-
flannel.yml

        kubectl get nodes
register: flannel_install
- name: Printing the status
  debug:
    msg: "{{ flannel_install.stdout_lines }}"

- name: Giving kubectl access to the user
  shell:
    cmd: |
      runuser -l skmirza -c 'mkdir -p $HOME/.kube && sudo cp -i
/etc/kubernetes/admin.conf $HOME/.kube/config && sudo chown $(id -u):
$(id -g) $HOME/.kube/config'

- name: Verifying the access
  shell:
    cmd: |
      runuser -l skmirza -c 'kubectl get nodes'
register: kubectl_access

- name: Printing the status
  debug:
    msg: "{{ kubectl_access.stdout_lines }}"

```

The screenshot shows a terminal window with two tabs. The current tab is titled 'skmirza@ip-14-0-1-84:/etc/ansible' and displays the output of an Ansible playbook. The command history at the top shows:

```

skmirza@ip-14-0-1-84:~/Documents/USB/Project_Details/Project_Resources/Te... x
skmirza@ip-14-0-1-84:/etc/ansible$ nano kubernetes-master-config.yaml
skmirza@ip-14-0-1-84:/etc/ansible$ cat kubernetes-master-config.yaml
# yaml file to configure the kubernetes Master using ansible
---
```

The main content of the terminal shows the execution of the playbook:

```

skmirza@ip-14-0-1-84:/etc/ansible$ nano kubernetes-master-config.yaml
skmirza@ip-14-0-1-84:/etc/ansible$ cat kubernetes-master-config.yaml
# yaml file to configure the kubernetes Master using ansible
---
- hosts: PM
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Installing and Configuring kubernetes
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-master"
    - name: Turning off swap
      shell:
        cmd: |
          sudo swapoff -a
          sudo sed -i '/ swap / s/^.*$/#/g' /etc/fstab
    - name: Installing the required packages
      shell:
        cmd: |
          sudo apt-get update
          sudo apt-get install -y ca-certificates curl
          sudo install -m 0755 -d /etc/apt/keyrings
          sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
          sudo chmod a+r /etc/apt/keyrings/docker.asc
          echo |
            "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
            ${. /etc/os-release && echo "$VERSION_CODENAME"} stable" | \
            sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
          sudo apt-get update
    - name: Installing Docker

```

- Now execute “ ansible-playbook kubernetes-master-config.yaml ”

```

Sun Feb 9 11:58:58
skmirza@ip-14-0-1-84:/etc/ansible

[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster,
[bootstrap-token] Creating the 'cluster-info' ConfigMap in the 'kube-public' namespace,
[kubebundle-finalize] Updating '/etc/kubernetes/kubebundle.conf' to point to a rotatable kubelet client certificate and key,
[addons] Applied essential addon: CoreDNS,
[addons] Applied essential addon: kube-proxy",
",
"Your Kubernetes control-plane has initialized successfully!",
",
"To start using your cluster, you need to run the following as a regular user:",
",
"  mkdir -p $HOME/.kube",
"  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config",
"  sudo chown $(id -u):$(id -g) $HOME/.kube/config",
",
"Alternatively, if you are the root user, you can run:",
",
"  export KUBECONFIG=/etc/kubernetes/admin.conf",
",
"You should now deploy a pod network to the cluster.",
"Run \"kubectl apply -f [podnetwork].yaml\" with one of the options listed at:",
"  https://kubernetes.io/docs/concepts/cluster-administration/addons/",
",
"Then you can join any number of worker nodes by running the following on each as root:",
",
"  kubeadm join 14.0.1.23:6443 --token w23vfo.x8jzl6lkrpiytwgz \\" \
    \t--discovery-token-ca-cert-hash sha256:50d75fb98ae76d907264ac1f59a1b5fb864143064945302d6107213f1c13114d \
"
]

TASK [Creating the .kube directory] *****
changed: [PM]

TASK [Installing the flannel network] *****
changed: [PM]

TASK [Printing the status] *****

```

- Copy and paste the join command for kubernetes worker nodes

```

Sun Feb 9 11:58:13
skmirza@ip-14-0-1-84:/etc/ansible

skmirza014@SkMirza-Ubuntu: ~/Documents/USB/Project_Details/Project_Resources/Te... x skmirza@ip-14-0-1-84:/etc/ansible x

TASK [Installing the flannel network] *****
changed: [PM]

TASK [Printing the status] *****
ok: [PM] => {
  "msg": [
    "namespace/kube-flannel created",
    "clusterrole/rbac.authorization.k8s.io/flannel created",
    "clusterrolebinding/rbac.authorization.k8s.io/flannel created",
    "serviceaccount/flannel created",
    "configmap/kube-flannel-cfg created",
    "daemonset.apps/kube-flannel-ds created",
    "NAME      STATUS   ROLES      AGE     VERSION",
    "kube-master  NotReady  control-plane  6s      v1.32.1"
  ]
}

TASK [Giving kubectl access to the user] *****
changed: [PM]

TASK [Verifying the access] *****
changed: [PM]

TASK [Printing the status] *****
ok: [PM] => {
  "msg": [
    "NAME      STATUS   ROLES      AGE     VERSION",
    "kube-master  NotReady  control-plane  8s      v1.32.1"
  ]
}

PLAY RECAP *****
PM : ok=22  changed=19  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

skmirza@ip-14-0-1-84:/etc/ansible$ 

```

- Kubernetes installed successful in kubernetes-master-node,

Step – 2.5 :- Now Configuring Kubernetes-Workers Node Using Ansible,

- Create a “kubernetes-workers-config.yaml” file and paste the following script,

```

# yaml file to configure the kubernetes Worker using ansible
---
- hosts: ProdWorkers # same name as in hosts file
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Installing and Configuring kubernetes on Worker01
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-worker01"
      when: inventory_hostname == "PW01" # same name as in hosts file
    - name: Installing and Configuring kubernetes on Worker02
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-worker02"
      when: inventory_hostname == "PW02" # same name as in hosts file

    - name: Turning off swap
      shell:
        cmd: |
          sudo swapoff -a
          sudo sed -i '/ swap / s/^.*$/#/g' /etc/fstab

    - name: Installing the required packages
      shell:
        cmd: |
          sudo apt-get update
          sudo apt-get install -y ca-certificates curl
          sudo install -m 0755 -d /etc/apt/keyrings
          sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
          /etc/apt/keyrings/docker.asc
          sudo chmod a+r /etc/apt/keyrings/docker.asc
          echo \
            "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable"
| \
          sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
          sudo apt-get update

```

```
- name: Installing Docker
  shell:
    cmd: |
      sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

- name: Configuring Docker
  shell:
    cmd: |
      sudo modprobe overlay
      sudo modprobe br_netfilter
      cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
      overlay
      br_netfilter
      EOF

- name: Configuring kubernetes
  shell:
    cmd: |
      sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
      net.bridge.bridge-nf-call-ip6tables = 1
      net.bridge.bridge-nf-call-iptables = 1
      net.ipv4.ip_forward = 1
      EOF

- name: Configuring containerd
  shell:
    cmd: |
      sudo sysctl --system
      sudo mkdir -p /etc/containerd
      sudo containerd config default | sudo tee
      /etc/containerd/config.toml

- name: Config containerd
  shell: sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml

- name: Restarting containerd
  shell: sudo systemctl restart containerd

- name: Installing pre-requisites dependencies
  shell:
```

```

cmd: |
  echo "Starting installation of pre-requisites"
  sudo apt-get update
  sudo apt-get install -y apt-transport-https ca-certificates curl gpg
- name: Adding Kubernetes repository
  shell:
    cmd: |
      sudo mkdir -p -m 755 /etc/apt/keyrings
      echo "Downloading Kubernetes release key"
      curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.32/deb/Release.key | 
sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
      echo "Adding Kubernetes repository"
      echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-
keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.32/deb/ ' | sudo tee 
/etc/apt/sources.list.d/kubernetes.list

- name: Installing kubeadm, kubelet and kubectl
  shell:
    cmd: |
      sudo apt-get update -y
      sudo apt-get install -y kubelet kubeadm kubectl
      sudo apt-mark hold kubelet kubeadm kubectl
      sudo systemctl enable kubelet
- name: Joining the worker nodes to the cluster
  shell:
    cmd: |
      sudo kubeadm join 14.0.1.23:6443 --token w23vfo.x8jzl6lkrpiytwgz
\\
      --discovery-token-ca-cert-hash
sha256:50d75fb98ae76d907264ac1f59a1b5fb864143064945302d6107213f
1c13114d # Paste the join command

```

```

Sun Feb 9 12:23:05
skmirza@ip-14-0-1-84:/etc/ansible
skmirza014@SkMirza-Ubuntu: ~/Documents/USB/Project_Details/Project_Resources/Te... x
skmirza@ip-14-0-1-84:/etc/ansible x
skmirza@ip-14-0-1-84:/etc/ansible
# yaml file to configure the kubernetes Worker using ansible
---
- hosts: kubenodes
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Installing and Configuring kubernetes on Worker01
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-worker01"
        when: inventory_hostname == "KubeW01"
    - name: Installing and Configuring kubernetes on Worker02
      shell:
        cmd: |
          sudo hostnamectl set-hostname "kube-worker02"
        when: inventory_hostname == "KubeW02"
    - name: Turning off swap
      shell:
        cmd: |
          sudo swapoff -a
          sudo sed -i '/ swap / s/^.*\$/#\1/g' /etc/fstab
    - name: Installing the required packages
      shell:
        cmd: |
          sudo apt-get update
          sudo apt-get install -y ca-certificates curl
          sudo install -m 0755 -d /etc/apt/keyrings
          sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
          sudo chmod a+r /etc/apt/keyrings/docker.asc
          echo \

```

- Now execute “ ansible-playbook kubernetes-workers-config.yaml ”

```

Sun Feb 9 12:53:59
skmirza@ip-14-0-1-84:/etc/ansible
skmirza014@SkMirza-Ubuntu: ~/Docu... x
skmirza@ip-14-0-1-84:/etc/ansible x
root@kube-worker01:~ x
root@kube-worker02:~ x
changed: [PW02]

TASK [Configuring containerd] *****
changed: [PW01]
changed: [PW02]

TASK [Config containerd] *****
changed: [PW01]
changed: [PW02]

TASK [Restarting containerd] *****
changed: [PW01]
changed: [PW02]

TASK [Installing pre-requisites dependencies] *****
changed: [PW01]
changed: [PW02]

TASK [Adding Kubernetes repository] *****
changed: [PW02]
changed: [PW01]

TASK [Installing kubeadm, kubelet and kubectl] *****
changed: [PW01]
changed: [PW02]

TASK [Joining the worker nodes to the cluster] *****
changed: [PW02]
changed: [PW01]

PLAY RECAP *****
PW01 : ok=14    changed=14   unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
PW02 : ok=14    changed=14   unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

skmirza@ip-14-0-1-84:/etc/ansible$ 

```

- kubernetes worker nodes installed Successfully,
- To verify it execute command in kubernetes master node,
- “ kubectl get nodes ”

```

skmirza@kube-master:~$ kubectl get nodes
NAME      STATUS   ROLES      AGE      VERSION
kube-master   Ready    control-plane   60m    v1.32.1
kube-worker01  Ready    <none>     4m42s   v1.32.1
kube-worker02  Ready    <none>     4m42s   v1.32.1
skmirza@kube-master:~$ 

```

- Configuring kubernetes workers completed successful,

Step - 2.5 :- Now Configuring Test-Server Node Using Ansible,

- Create a “Test-Server-config.yaml” file and paste the following script,

yaml file to configure the test server (tomcat server) using ansible

```

---
- hosts: TCS
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Install Java
      apt:
        name: openjdk-17-jre
        state: present
    - name: Install Tomcat
      shell:
        cmd: |
          wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.98/bin/apache-tomcat-9.0.98.tar.gz
          tar -xvf apache-tomcat-9.0.98.tar.gz

```

```
rm -rf apache-tomcat-9.0.98.tar.gz
mv apache-tomcat-9.0.98 /opt/tomcat
chown -R skmirza:skmirza /opt/tomcat
chmod -R +x /opt/tomcat/bin/*.sh
- name: Creating a systemd service file for Tomcat
  shell:
    cmd: |
      cat <<EOF > /etc/systemd/system/tomcat.service
      [Unit]
      Description=Apache Tomcat Web Application Container
      After=network.target

      [Service]
      Type=forking
      ExecStart=/opt/tomcat/bin/startup.sh
      ExecStop=/opt/tomcat/bin/shutdown.sh
      User=skmirza
      Group=skmirza
      Restart=always
      RestartSec=10

      [Install]
      WantedBy=multi-user.target
      EOF
- name: Start and enable Tomcat service
  systemd:
    name: tomcat
    state: started
    enabled: yes
    daemon_reload: yes
- name: Status of Tomcat service
  shell: systemctl status tomcat
  register: tomcat_status
```

```

Sun Feb 9 18:56:21
skmirza@ip-14-0-1-36:/etc/ansible
skmirza@ip-14-0-1-36:/etc/ansible$ nano Test-Server-Config.yaml
skmirza@ip-14-0-1-36:/etc/ansible$ cat Test-Server-Config.yaml
# yaml file to configure the test server (tomcat server) using ansible
---
- hosts: TCS
  gather_facts: no
  become: yes
  tasks:
    - name: Update Package
      shell: sudo apt update -y
    - name: Install Java
      apt:
        name: openjdk-17-jre
        state: present
    - name: Install Tomcat
      shell:
        cmd: |
          wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.98/bin/apache-tomcat-9.0.98.tar.gz
          tar -xvf apache-tomcat-9.0.98.tar.gz
          rm -rf apache-tomcat-9.0.98.tar.gz
          mv apache-tomcat-9.0.98 /opt/tomcat
          chown -R skmirza:skmirza /opt/tomcat
          chmod -R +x /opt/tomcat/bin/*.sh
    - name: Creating a systemd service file for Tomcat
      shell:
        cmd: |
          cat <<EOF > /etc/systemd/system/tomcat.service
          [Unit]
          Description=Apache Tomcat Web Application Container
          EOF

```

- Now execute “ ansible-playbook test-server-config.yaml ”

```

Sun Feb 9 18:58:08
skmirza@ip-14-0-1-36:/etc/ansible
skmirza@ip-14-0-1-36:/etc/ansible$ ansible-playbook test-server-config.yaml
TASK [Update Package] *****
The authenticity of host '14.0.14.186 (14.0.14.186)' can't be established.
ED25519 key fingerprint is SHA256:vdPd2usfRxGffVImjj0F/3Syy1GKGATcgUvKqaneU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
[WARNING]: Platform linux on host TCS is using the discovered Python interpreter at /usr/bin/python3.10, but future installation of
another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-
core/2.17/reference_appendices/interpreter_discovery.html for more information.
changed: [TCS]

TASK [Install Java] *****
changed: [TCS]

TASK [Install Tomcat] *****
changed: [TCS]

TASK [Creating a systemd service file for Tomcat] *****
changed: [TCS]

TASK [Start and enable Tomcat service] *****
changed: [TCS]

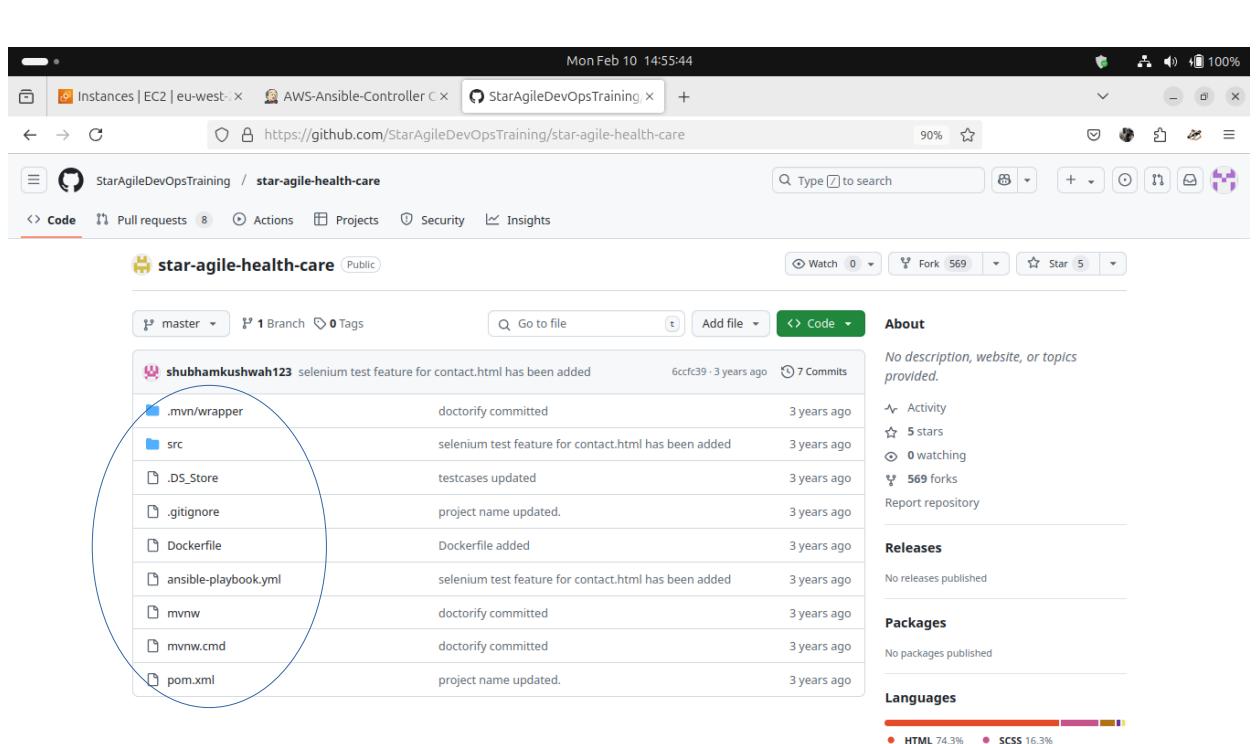
TASK [Status of Tomcat service] *****
changed: [TCS]

PLAY RECAP *****
TCS : ok=6    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

```

- Configuring test server completed successful,
- Hence Creating, Configuring, all servers successfully completed.

Step – 3.1 :- Now Checking the Github Repository of Medicure Web Application,



- Change the Dockerfile, pom.xml, medicuredeploy.yaml files if only needed,
- Create a medicuredeploy.yaml file for kubernetes deployment,
- create script-run-jar.sh file to auto run the jar file in test server in background,

Step - 3.2 :- Change the pom.xml if needed, copy and paste the following,

```
<artifactId>medicure</artifactId>
<version>1.0.1</version>
```

Step - 3.3 :- Change the Dockerfile if needed, copy and paste the following,

```
FROM openjdk:17
COPY ./target/*jar medicure.jar
ENTRYPOINT ["java","-jar","/medicure.jar"]
```

Step - 3.4 :- Change the application.properties if needed, copy and paste the following,

```
spring.jpa.open-in-view=false
server.port=8081
```

Step – 3.5 :- Create a script-run-jar.sh file to run the jar files,

```
#!/bin/bash
# Automatically select the first JAR file in the directory
Wrk_Dir="/opt/tomcat/webapps"
JAR_FILE=$(ls "$Wrk_Dir"/medicure*.jar 2>/dev/null | sort -V |
tail -n 1)
PID_FILE="$Wrk_Dir/app.pid"

# Check if a JAR file exists
if [ -z "$JAR_FILE" ]; then
    echo "No JAR file found in the current directory."
    exit 1
fi

case "$1" in
    start)
        if [ -f "$PID_FILE" ]; then
            echo "Application is already running (PID: $(cat $PID_FILE))."
        else
            echo "Starting the application with $JAR_FILE..."
            nohup java -jar "$JAR_FILE" > output.log 2>&1 &
            echo $! > "$PID_FILE"
            sleep 2
            if ps -p $(cat "$PID_FILE") > /dev/null; then
                echo "Application started successfully (PID: $(cat $PID_FILE))."
            else
                echo "Failed to start the application. Check output.log for errors."
                rm -f "$PID_FILE"
            fi
        fi
    ;;
    stop)
        if [ -f "$PID_FILE" ]; then
            PID=$(cat "$PID_FILE")
            echo "Stopping application with PID $PID..."
            kill $PID
            sleep 2
        fi
    ;;
esac
```

```

        if ps -p $PID > /dev/null; then
            echo "Failed to stop the application. Forcing
termination..."
            kill -9 $PID
        else
            echo "Application stopped successfully."
        fi
        rm -rf "$PID_FILE"
        rm -rf output.log
    else
        echo "No running application found."
    fi
;;
status)
if [ -f "$PID_FILE" ]; then
    PID=$(cat "$PID_FILE")
    if ps -p $PID > /dev/null; then
        echo "Application is running (PID: $PID)."
    else
        echo "PID file exists but application is not running."
        rm -f "$PID_FILE"
    fi
else
    echo "Application is not running."
fi
;;
*)
echo "Usage: $0 {start | stop | status}"
;;
esac

```

Step – 3.6 :- Create a medicuredeploy.yaml file to deploy the web application to production server,

yaml file to Deployment in Production Server

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: skmirza-medicure-deploy
  labels:

```

```

    app: skmirza-medicure-deploy-lb01
spec:
  replicas: 4
  selector:
    matchLabels:
      app: skmirza-medicure-app
template:
  metadata:
    labels:
      app: skmirza-medicure-app
spec:
  containers:
    - name: skmirza-medicure-container
      image: sameer014/skmirza-medicure-img:latest
      ports:
        - containerPort: 8081
---
apiVersion: v1
kind: Service
metadata:
  name: skmirza-medicure-np-service
  labels:
    app: skmirza-medicure-np-app
spec:
  selector:
    app: skmirza-medicure-app

  type: NodePort
  ports:
    - nodePort: 30014
      port: 8081
      targetPort: 8081

```

Step – 3.7 :- Add the Following Environment variables in node level,

#Variable

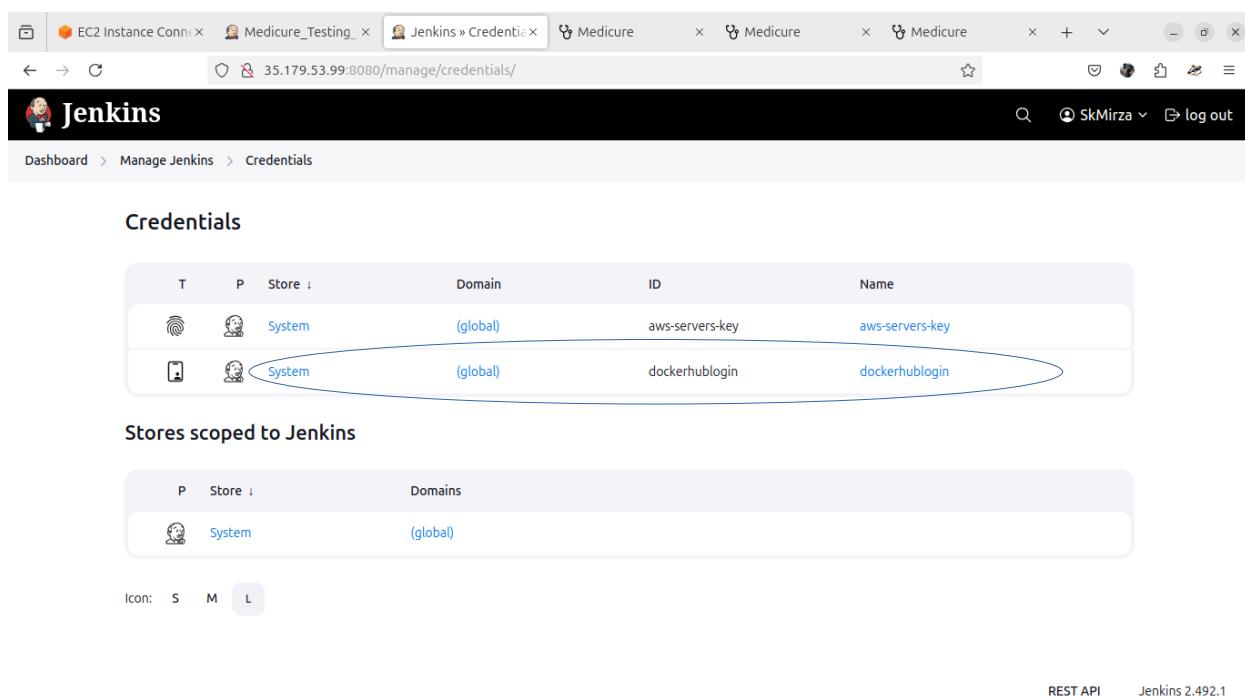
```

App_Name = medicure
Container_Name = ${User_Name}-${App_Name}-container
Default_Ver = 1.0.0
Deploy_Name = ${User_Name}-${App_Name}-deploy
KubeM_Pvt_IP = <Pvt_IP>
KubeW01_Pub_IP = <Pub_IP>

```

KubeW02_Pub_IP = <Pub_IP>
 Prod_Workspace =
 /home/\${User_Name}/workspace/Medicure_Production_Pipeline
 Build01_Pvt_IP = <Pvt_IP>
 BN01_Path = /home/\${User_Name}/workspace/\${JOB_NAME}
 Test_Server_Path = /opt/tomcat/webapps
 Test_Server_Pub_IP = <Pub_IP>
 Test_Server_Pvt_IP = <Pvt_IP>
 User_Name = skmirza

Step – 3.8 :- Add DockerHub Credential in Jenkins,



The screenshot shows the Jenkins 'Credentials' management interface. It lists two entries:

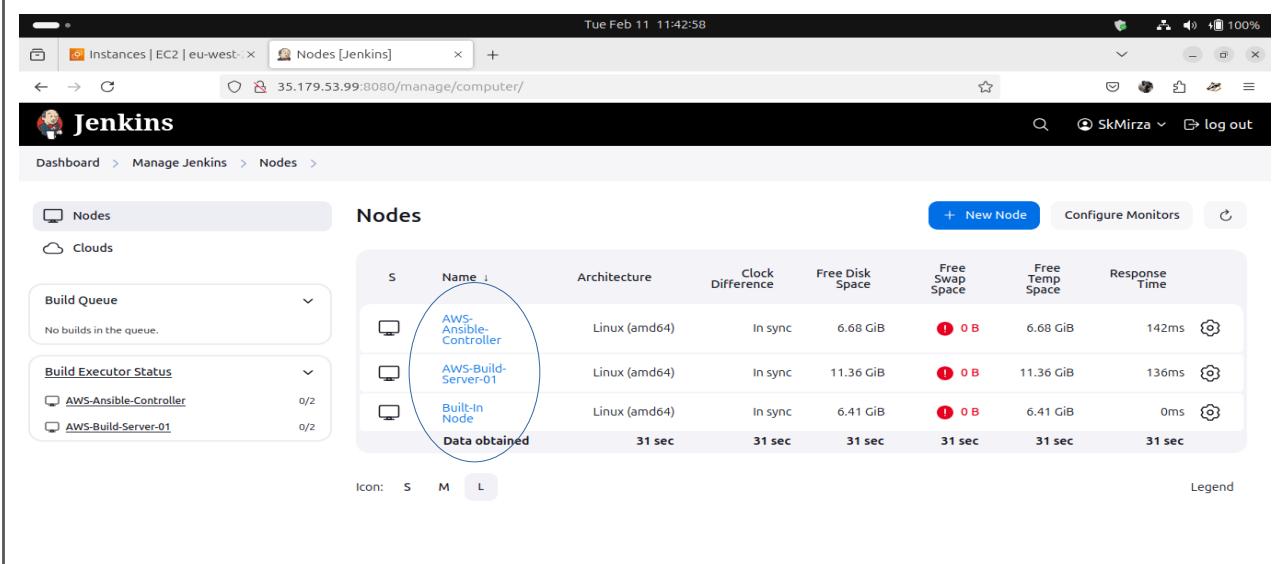
T	P	Store	Domain	ID	Name
aws-servers-key	System	(global)		aws-servers-key	aws-servers-key
dockerhublogin	System	(global)		dockerhublogin	dockerhublogin

Below the table, there is a section titled 'Stores scoped to Jenkins' which contains a single entry:

P	Store	Domains
System	(global)	

At the bottom right, it says 'REST API Jenkins 2.492.1'.

Step – 3.9 :- Add Build-Server And Ansible-Server in Jenkins,



The screenshot shows the Jenkins 'Nodes' management interface. It lists three nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
AWS-Ansible-Controller	Linux (amd64)	In sync	6.68 GiB	! 0 B	6.68 GiB	142ms	142ms
AWS-Build-Server-01	Linux (amd64)	In sync	11.36 GiB	! 0 B	11.36 GiB	136ms	136ms
Built-In Node	Linux (amd64)	In sync	6.41 GiB	! 0 B	6.41 GiB	0ms	0ms

At the bottom right, it says 'Legend'.

Step - 4.1 :- Creating a Testing Pipeline for testing medicure web application in test server,

- Stages of pipelines,
 - Testing Pipeline,
 1. Getting The Version
 2. SCM Checkout (Code)
 3. Build The Code (Using Maven)
 4. Deploying It To Test Server
 5. Verify Deploying By Response Of Web Site
 6. Verify Deploying By the User
 7. Copy Required Files To Production Workspace
 8. Triggering Production Pipeline

Step - 4.2 :- Creating a Production Pipeline for deploying financeme web application in production server,

- Production Pipeline,
 1. Build The Docker Image
 2. Login To DockerHub Account
 3. Push The Docker Image To DockerHub Repository
 4. Deploying The Web Application In Production Servers

- Create a New Job “Medicure_Testing_Pipeline” and paste the following,

```

pipeline {
agent none
environment {
Code_Checkout = false
Maven_Build = false
Deploy_Test = false
Deploy_Verify = false
Verify_User = false
}
stages {
stage ('Get_Version') {
agent {
label 'BN01'
}
steps {
script {
echo "The Default Version is: ${env.Default_Ver}"
try {
timeout(time: 15, unit: 'SECONDS') {
// Prompt user to provide a new version number within 15 seconds
def userVersion = input(
message: 'Please Provide The Version Of The Web Application',
ok: 'Submit',
parameters: [string(defaultValue: env.Default_Ver, description: 'Provide a new version number', name: 'new_ver')]
)
// If input is provided, set it as the new version
env.new_ver = userVersion
echo "The New Version Provided is: ${env.new_ver}"
}
} catch (Exception e) {
}
}
}
}
}

```

```

// On timeout or abort, increment the default version number by
0.0.1
def version = env.Default_Ver.tokenize('.')
version[2] = (version[2] as Integer) + 1
env.new_ver = version.join('.')
echo "No Version Provided So The New Version Will: ${env.new_ver}"
}

}
}
}

stage('SCM_Checkout') {
agent {
label 'BN01'
}
steps {
cleanWs()
echo '##### Checking out the code #####'
git 'https://github.com/sameer-014-Devops/Medicure.git'
}
post {
success {
script {
echo '##### Code Checkout is SUCCESSFUL #####'
Code_Checkout = true
}
}
failure {
script {
echo '##### Code Checkout is FAILED #####'
}
}
}
}

stage('Maven_Build') {
agent {

```

```

label 'BN01'
}
steps {
echo '##### Building the code #####'
sh 'mvn clean package'
echo '##### Packaging the code SUCCESSFULL ######'
echo '##### Deleting the Dependency Installed ######'
sh 'rm -rf ~/.m2/repository'
echo '##### Dependency Deleted in BuildNode #####'
}
post {
success {
script {
echo '##### Maven Build is SUCCESSFUL #####'
Maven_Build = true
}
}
failure {
script {
echo '##### Maven Build is FAILED #####'
}
}
}
stage('Deploy_Test') {
agent {
label 'AC'
}
steps {
echo '##### Deploying the code to Test Server #####'
script {
// Checking and removing the existing sh file in Test Server
def fileExists1 = sh(script: """ssh $User_Name@$Test_Server_Pvt_IP ls $Test_Server_Path/*.sh | | true""", returnStdout: true).trim()
if (fileExists1) {

```

```

echo 'Existing sh file found in the Test Server. Removing the existing sh
file...'
sh """" ssh $User_Name@$Test_Server_Pvt_IP "sh $Test_Server_Path/*.sh
stop" """
sleep 3
sh """ssh $User_Name@$Test_Server_Pvt_IP rm -rf
$Test_Server_Path/*.sh"""
} else {
echo 'No existing sh file found in the Test Server...'
}

// Checking and Removing the existing jar file in Test Server
def fileExists = sh(script: """ssh $User_Name@$Test_Server_Pvt_IP ls
$Test_Server_Path/*.jar || true""", returnStatus: true)
if (fileExists == 0) {
echo 'Existing jar file found in the Test Server. Removing the existing jar
file...'
sh """ssh $User_Name@$Test_Server_Pvt_IP rm -rf
$Test_Server_Path/*.jar"""
} else {
echo 'No existing jar file found in the Test Server...'
}

echo ##### Deploying the code to Test Server #####
// Copy the jar file and sh to the test server by using SCP
sh """ scp $User_Name@$Build01_Pvt_IP:$BN01_Path/target/*.jar
$User_Name@$Test_Server_Pvt_IP:$Test_Server_Path/ """
sh """ scp $User_Name@$Build01_Pvt_IP:$BN01_Path/*.sh
$User_Name@$Test_Server_Pvt_IP:$Test_Server_Path/ """
sh """ ssh $User_Name@$Test_Server_Pvt_IP chmod +x
$Test_Server_Path/*.sh """

// Run the jar file in the test server in using .sh file
sh """ ssh $User_Name@$Test_Server_Pvt_IP "sh $Test_Server_Path/*.sh
start" """
}
}

```

```
post {
success {
script {
Deploy_Test = true
echo ##### Deploy to Test Server is SUCCESSFUL #####
}
}
failure {
script {
echo ##### Deploy to Test Server is FAILED #####
}
}
}
stage('Verify_Deploy_Test') {
agent {
label 'BN01'
}
when {
expression { Deploy_Test == true }
}
steps {
echo *****Verifying Test Deployment*****
//wait for 15 seconds before verifying the deployment
sleep 15
script {
def response = sh(script: """curl -s -o /dev/null -w '%{http_code}' http://$Test_Server_Pub_IP:8081""", returnStdout: true).trim()
if (response == '200') {
Deploy_Verify = true
echo *****Test Deployment is SUCCESSFUL*****
} else {
error *****Test Deployment is FAILED*****
}
}
}
```

```

}

post {
success {
script {
echo '##### Test Deployment Verification is SUCCESSFUL
#####
}
}

failure {
script {
echo '##### Test Deployment Verification is FAILED #####
'}
}

stage('Verification_User'){
agent {
label 'BN01'
}
when{
expression { Deploy_Verify == true }
}
steps{
echo '*****Verifying By User*****'
script{
echo "The Application URL: http://$Test_Server_Pub_IP:8081"
// Prompt user input to verify the deployment by accessing the application URL, within 60 seconds timeout, and proceed based on the user input (Y/N), if not provided, proceed with 'Y' as default input, and if the input is 'N', then only abort the pipeline.
try {
timeout(time: 120, unit: 'SECONDS') {
def userInput = input(
message: 'Please Verify The Deployment By Accessing The Application URL',

```

```
ok: 'Submit',
parameters: [choice(choices: ['Y', 'N'], description: 'Do you want to proceed
with the deployment verification?', name: 'user_input')]
)
if (userInput == 'Y') {
echo 'User Has Confirmed The Deployment Verification So Proceeding
With Docker Build'
Verify_User = true
} else {
echo 'User Has Denied The Deployment Verification So Aborting The
Pipeline'
Verify_User = false
currentBuild.result = 'ABORTED'
error 'Pipeline Aborted by User'
}
}
} catch (hudson.AbortException e) {
// If timeout occurs or any other exception, proceed with the
pipeline
if (e.getMessage().contains("Timeout")) {
Verify_User = true
echo 'User Did Not Provide Any Input Within The Timeout Period,
Proceeding With Default Input (Y)'
} else {
throw e
}
} catch (Exception e) {
// Handle other exceptions
Verify_User = true
echo 'An unexpected error occurred, proceeding with default input (Y)'
}
}
}
post{
success{
```

```

script{
echo ##### User Verification is SUCCESSFUL #####
echo ##### Test Server Deployment is SUCCESSFUL #####
}
}

failure{
script{
echo ##### User Verification is FAILED #####
echo ##### Test Server Deployment is FAILED #####
}
}
}

stage ('Copy_Files_To_Production_Workspace'){
agent {
label 'BN01'
}
when{
expression { Verify_User == true }
}
steps{
sh """mkdir -p $Prod_Workspace"""
sh """cp -R Dockerfile *.yaml target $Prod_Workspace"""
echo *****Copied the Dockerfile, medicuredeploy.yaml, and target files to the Medicure Production workspace*****
cleanWs()
deleteDir()
}
}
}

post {
success {
build job: 'Medicure_Production_Pipeline', parameters: [string(name: 'new_ver', value: env.new_ver)]
echo ##### Production Pipeline Triggered #####
}
}
}

```

```

}

failure {
echo '##### Testing Pipeline is FAILED ####'
}

}

}

```

- Create a other New Job “Medicure_Production_Pipeline” and paste the following,

```

pipeline{
agent none
parameters {
string(name: 'new_ver', defaultValue: 'latest', description: 'Deploy Version from Test Pipeline')
}
environment {
DOCKERHUB_CREDENTIALS = credentials('dockerhublogin')
Docker_Build = false
Docker_Login = false
Docker_Push = false
Deploy_Main = false
}
stages {
stage('Docker_Build') {
agent {
label 'BN01'
}
steps {
echo '*****Building Docker Image*****'

script {
def latestImageExists = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-
img:latest""", returnStdout: true).trim()

```

```

if (latestImageExists) {
echo 'Docker Image with the latest tag already exists. Removing the
existing image...'
sh """docker rmi -f $DOCKERHUB_CREDENTIALS_USR/$User_Name-
$App_Name-img:latest"""
} else {
echo 'Docker Image with the latest tag does not exist...'
}

def versionedImageExists = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:$
{params.new_ver}""", returnStdout: true).trim()

if (versionedImageExists) {
echo 'Docker Image with the specified version already exists. Removing
the existing image...'
sh """docker rmi -f $DOCKERHUB_CREDENTIALS_USR/$User_Name-
$App_Name-img:${params.new_ver}"""
} else {
echo 'Docker Image with the specified version does not exist...'
}

sh 'docker logout'
sh """docker build -t
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:$
{params.new_ver} ."""
sh """docker tag $DOCKERHUB_CREDENTIALS_USR/$User_Name-
$App_Name-img:${params.new_ver}
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-
img:latest"""
sh 'docker images'

def latestImgExists = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-
img:latest""", returnStdout: true).trim()

```

```

if (latestImgExists) {
echo '*****Docker Image Build SUCCESSFUL*****'
Docker_Build = true
} else {
error '*****Docker Image Build FAILED*****'
}
}
}

post {
success {
script {
echo ##### Docker build is SUCCESSFUL #####
}
}

failure {
script {
echo ##### Docker build is FAILED #####
}
}

stage ('Docker_Login'){
agent {
label 'BN01'
}
when{
expression { Docker_Build == true }
}
steps {
echo 'Login to Docker Hub'
sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
script {
if (sh(script: 'docker info | grep -i "Username: $DOCKERHUB_CREDENTIALS_USR"', returnStatus: true) == 0) {

```

```

echo '*****Docker Hub Login SUCCESSFUL*****'
Docker_Login = true
} else {
error '*****Docker Hub Login FAILED*****'
}
}
}
post{
success{
script{
echo ##### Docker Login is SUCCESSFUL #####
}
}
failure{
script{
echo ##### Docker Login is FAILED #####
}
}
}
stage ('Docker_Push'){
agent {
label 'BN01'
}
when{
expression { Docker_Login == true }
}
steps {
echo 'Pushing Docker Image to Docker Hub'
script {
echo "*****Pushing Docker Image With The Version ${params.new_ver}*****"
sh """docker push $DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:${params.new_ver}"""
echo "*****Pushing Docker Image With The Tag 'latest'*****"
}
}
}

```

```

sh """docker push $DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:latest"""
def imagePushed = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:latest""", returnStdout: true).trim()
if (imagePushed) {
echo '*****Docker Image Push SUCCESSFUL*****'
Docker_Push = true
} else {
error '*****Docker Image Push FAILED*****'
}
}
}

post{
success{
script{
echo ##### Docker Push is SUCCESSFUL #####
echo ##### Removing the Docker Images from the Local Machine#####
def latestImageExists1 = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:latest""", returnStdout: true).trim()

if (latestImageExists1) {
echo 'Docker Image with the latest tag already exists. Removing the
existing image...'
sh """docker rmi -f $DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:latest"""
} else {
echo 'Docker Image with the latest tag does not exist...'
}

def versionedImageExists1 = sh(script: """docker images -q
$DOCKERHUB_CREDENTIALS_USR/$User_Name-$App_Name-img:$
{params.new_ver}""", returnStdout: true).trim()
}
}
}
}

```



```

// checking the pod status in the Main Server, if any pod name
contains the User_Name and App_Name, then update the
deployment, else create a new deployment
def podExists = sh(script: """ ssh $User_Name@$KubeM_Pvt_IP kubectl
get pods -n default --no-headers | grep -E ".*$${USER_NAME}.*$"
${APP_NAME}.* | .*$${APP_NAME}.*$${USER_NAME}.*" || true
""",returnStdout: true
).trim()
if (podExists) {
echo '*****Pod is already running, So Updating the
Deployment*****'
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl apply -f
medicureddeploy.yaml"""
sleep 5
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl set image deploy
$Deploy_Name
$Container_Name=$DOCKERHUB_CREDENTIALS_USR/$User_Name-
$App_Name-img:${params.new_ver}"""
sleep 10
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl get pods -o wide"""
} else {
echo '*****Pod is not running, So Creating the
Deployment*****'
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl apply -f
medicureddeploy.yaml"""
sleep 5
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl get pods -o wide"""
}
// if the pod status is errimagepull or pending, then it should
rollout undo the deployment
def podStatuses = sh(script: """ ssh $User_Name@$KubeM_Pvt_IP kubectl
get pods -n default --no-headers | grep -E ".*$${USER_NAME}.*$"
${APP_NAME}.* | .*$${APP_NAME}.*$${USER_NAME}.*" | awk '{print \
\$3}"", returnStdout: true).trim().split('\n')
def rollback = false

```

```

for (status in podStatuses) {
echo "Pod status: ${status}"
if (status == 'ErrImagePull' || status == 'Pending') {
rollback = true
break
}
}
if (rollback) {
echo '*****Pod Status is ErrImagePull or Pending, So Rolling Back the Deployment*****'
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl rollout undo deployment/${APP_NAME}"""
sleep 10
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl get pods -o wide"""
} else {
echo '*****All Pod Statuses are Running, So Deployment is SUCCESSFUL*****'
}
echo '*****Check The Deploy in Main Server*****'
sleep 15
echo "http://${env.KubeW01_Pub_IP}:30014"
echo "http://${env.KubeW02_Pub_IP}:30014"
Deploy_Main = true
}
}
post{
success{
script{
echo ##### Deployment to the Production Environment is SUCCESSFUL #####
cleanWs()
deleteDir()
}
}
failure{

```

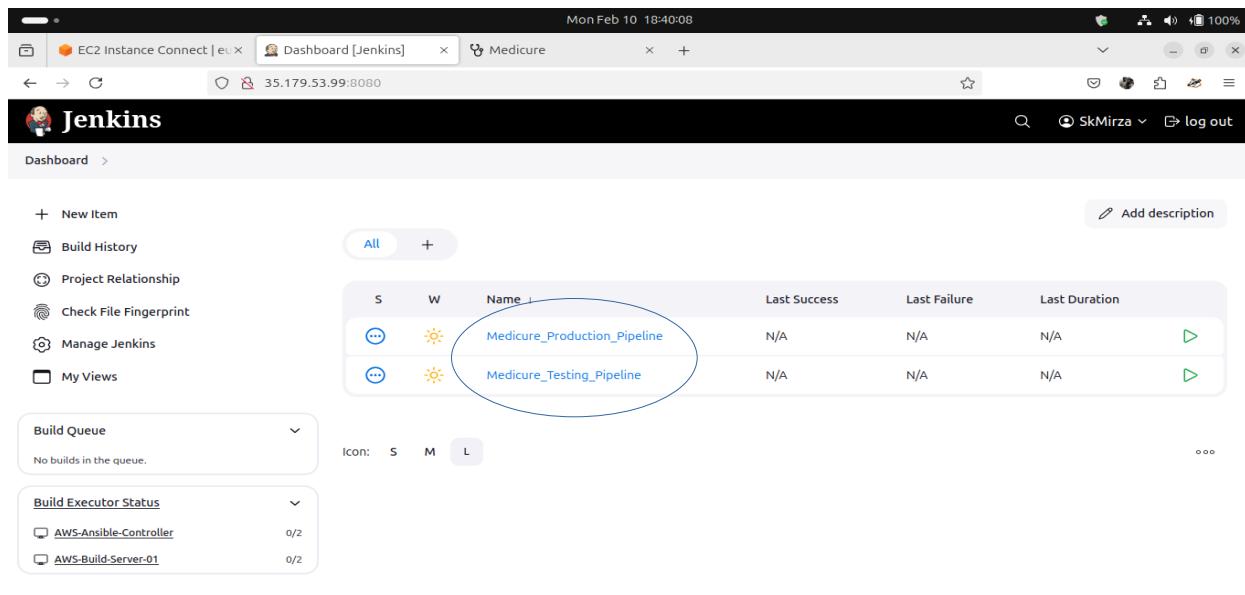
```

script{
echo ##### Deployment to the Production Environment is FAILED#####
}
}
}
}
}

post {
success {
echo ##### Pipeline Completed Successfully #####
}
failure {
echo ##### Pipeline Completed with FAILURE #####
}
}
}
}
}

```

Step - 4.2 :- Running the Testing Pipeline,



The screenshot shows the Jenkins dashboard with the following details:

- Top Bar:** Mon Feb 10 18:40:08, EC2 Instance Connect | el... x, Dashboard [Jenkins] x, Medicure x, +
- User Information:** SkMirza, log out
- Dashboard Header:** Jenkins, Dashboard >
- Left Sidebar:**
 - + New Item
 - Build History
 - Project Relationship
 - Check File Fingerprint
 - Manage Jenkins
 - My Views
- Main Content:**
 - Pipeline List:** All

S	W	Name	Last Success	Last Failure	Last Duration
...	...	Medicure_Production_Pipeline	N/A	N/A	N/A
...	...	Medicure_Testing_Pipeline	N/A	N/A	N/A
 - Build Queue:** No builds in the queue.
 - Build Executor Status:**

AWS-Ansible-Controller	0/2
AWS-Build-Server-01	0/2
- Bottom Right:** Direct API, Jenkins 2.202.4

- Pipelines Created, Now go to Testing Pipeline,

The screenshot shows the Jenkins interface for the 'Medicure_Testing_Pipeline'. On the left, there's a sidebar with various options like Status, Changes, Build Now (which is highlighted with a blue oval), Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The main area is titled 'Medicure_Testing_Pipeline' and 'Stage View'. It displays a message: 'No data available. This Pipeline has not yet run.' Below this is a 'Permalinks' section. At the bottom, there's a 'Builds' section with a message: 'No builds'.

- Now Running The Testing Pipeline, Make sure to run after creating Production Pipeline,
- Because After Testing Pipeline successful it will trigger Production Pipeline,

The screenshot shows the same Jenkins interface after a build has been triggered. In the 'Stage View' section, a stage named 'Get_Version' is shown with an average stage time of 479ms. A tooltip indicates it was paused for 20ms. The 'Builds' section now shows a single build: '#1 1:17 PM' with a status of 'In Progress'. The Jenkins footer indicates version 2.492.1.

- Build Started, Click on #1 for output console,

Mon Feb 10 18:47:41

EC2 Instance Connect | eu-west-1 | Medicure_Testing_Pipeline | Medicure

35.179.53.99:8080/job/Medicure_Testing_Pipeline/1/console

Dashboard > Medicure_Testing_Pipeline > #1

Console Output

```

Started by user SkMirza
[Pipeline] Start of Pipeline
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Get_Version)
[Pipeline] node
Running on AWS-Build-Server-01 in /home/skmirza/workspace/Medicure_Testing_Pipeline
[Pipeline] {
[Pipeline] script
[Pipeline] {
[Pipeline] echo
The Default Version is: 1.0.0
[Pipeline] timeout
Timeout set to expire in 15 sec
[Pipeline] {
[Pipeline] input
Input requested

```

Edit Build Information

Timings

Paused for Input

Pipeline Overview

Pipeline Console

Thread Dump

Pause/resume

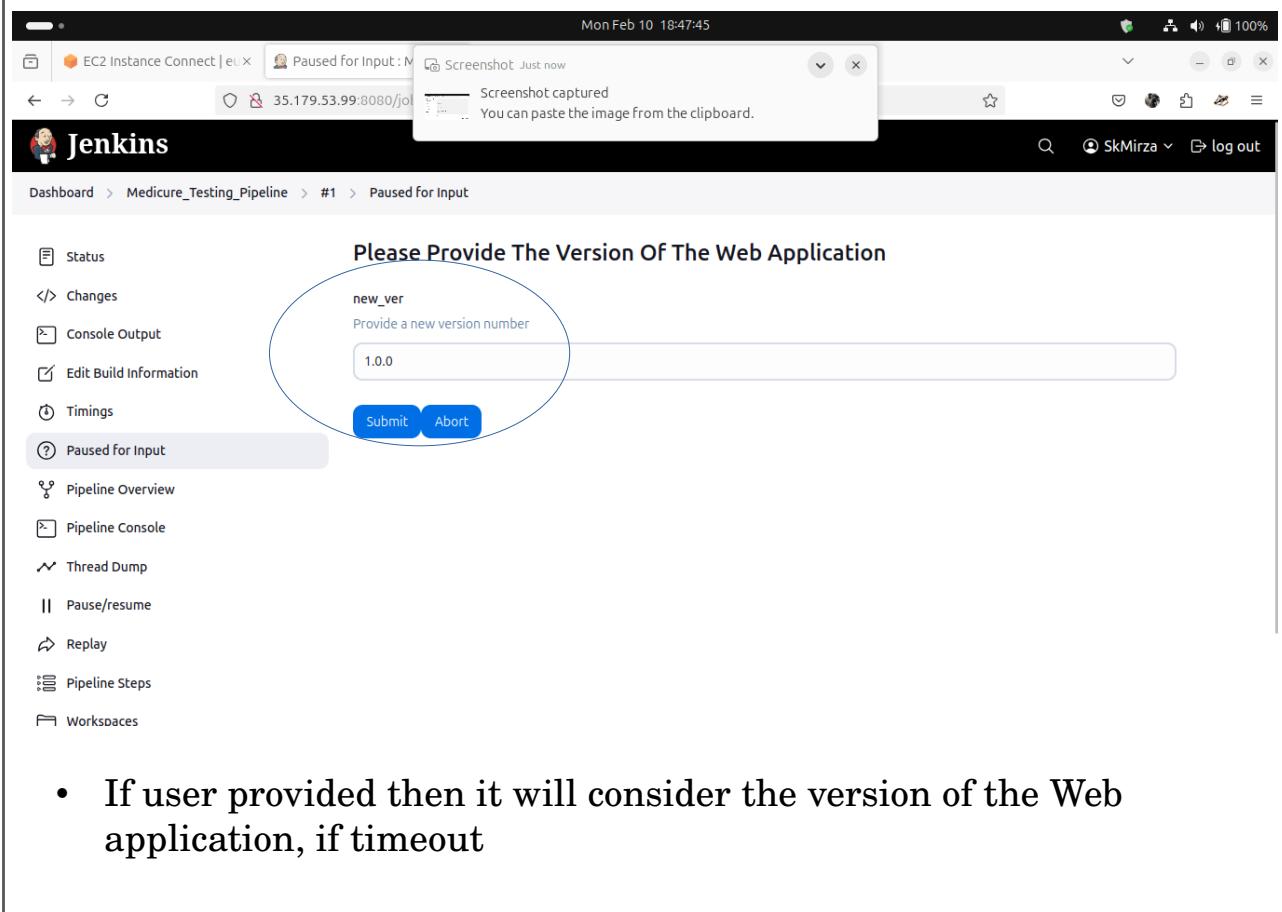
Replay

Pipeline Steps

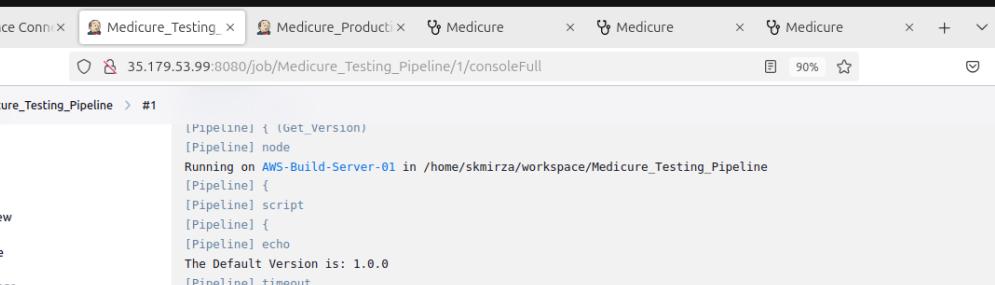
Workspaces

REST API Jenkins 2.492.1

- Here the Default version is: 1.0.0 as set in the environment variable,
- It will Ask For User Input to provide Version of the web application, for 15 Sec timeout



- If user provided then it will consider the version of the Web application, if timeout



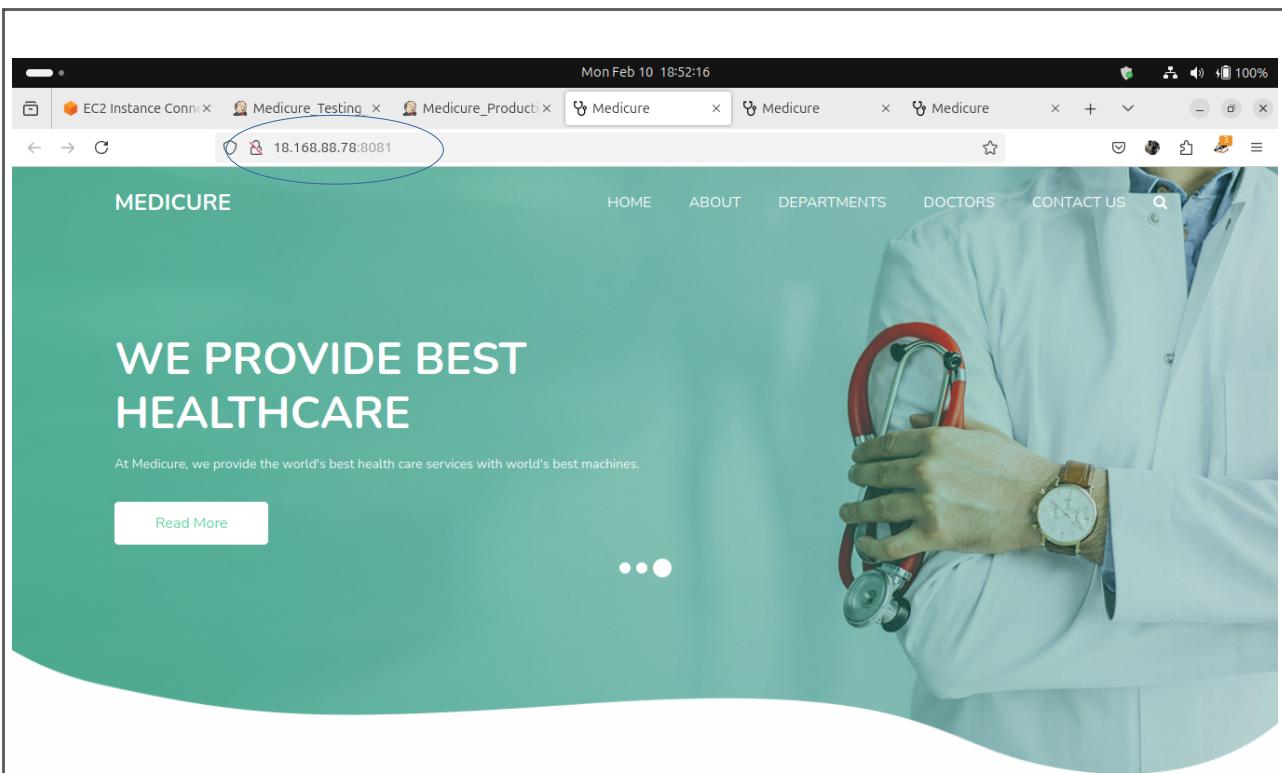
```
Mon Feb 10 19:06:52
EC2 Instance Conn X Medicure_Testing_ X Medicure_Product_ X Medicure X Medicure X Medicure X +
35.179.53.99:8080/job/Medicure_Testing_Pipeline/1/consoleFull
90% ⭐
Dashboard > Medicure_Testing_Pipeline > #1

Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces

(Pipeline) { (Get_Version)
[Pipeline] node
Running on AWS-BUILD-SERVER-01 in /home/skmirza/workspace/Medicure_Testing_Pipeline
[Pipeline] {
[Pipeline] script
[Pipeline] {
[Pipeline] echo
The Default Version is: 1.0.0
[Pipeline] timeout
Timeout set to expire in 15 sec
[Pipeline] {
[Pipeline] input
Input requested
Cancelled nested steps due to timeout
[Pipeline]
[Pipeline] // timeout
[Pipeline] echo
No Version Provided So The New Version Will: 1.0.1
[Pipeline]
[Pipeline] // script
[Pipeline]
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SCM_Checkout)
[Pipeline] node
Running on AWS-BUILD-SERVER-01 in /home/skmirza/workspace/Medicure_Testing_Pipeline
[Pipeline] {
[Pipeline] cleanup
```

- It will Increment the version 1.0.0 to 1.0.1,

- It will Automatically Test the Deployment by response of the web site,



- And wait for the user to verify the Deployment to test server to approve the testing, for 2 min timeout,

The screenshot shows a Jenkins pipeline interface for a Medicure Testing Pipeline. The pipeline has reached a 'Paused for Input' stage. A message on the screen reads: 'Please Verify The Deployment By Accessing The Application URL'. Below this, there is a 'user_input' field with the question 'Do you want to proceed with the deployment verification?'. A dropdown menu shows 'Y' selected. There are two buttons at the bottom: 'Submit' and 'Abort'. On the left sidebar, there are several other pipeline stages listed: 'status', 'Changes', 'Console Output', 'Edit Build Information', 'Timings', 'Paused for Input' (which is currently active), 'Git Build Data', 'Pipeline Overview', 'Pipeline Console', 'Thread Dump', 'Pause/resume', 'Replay', and 'Pipeline Steps'.

- If user provide 'Y' it will continuous the pipeline,
- If user provide 'N' it will abort the pipeline,
- if timeout it will consider as 'Y' because of Previous testing success,

```

Mon Feb 10 19:23:54
[Pipeline] echo
#####
User Verification is SUCCESSFUL #####
[Pipeline] echo
#####
Test Server Deployment is SUCCESSFUL #####
[Pipeline]
[Pipeline] // script
[Pipeline]
[Pipeline] // node
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Copy_Files_To_Production_Workspace)
[Pipeline] node
Running on AWS-Build-Server-01 in /home/skmirza/workspace/Medicure_Testing_Pipeline
[Pipeline] {
[Pipeline] sh
+ mkdir -p /home/skmirza/workspace/Medicure_Production_Pipeline
[Pipeline] sh
+ cp _R_Dockerfile medicuredeploy.yaml target /home/skmirza/workspace/Medicure_Production_Pipeline
[Pipeline] echo
*****Copied the Dockerfile, medicuredeploy.yaml, and target files to the Medicure Production workspace*****
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] deleteDir
[Pipeline]
[Pipeline] // node
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] Build (Building Medicure_Production_Pipeline)
Scheduling project: Medicure_Production_Pipeline
Starting building: Medicure_Production_Pipeline #1
Build Medicure_Production_Pipeline #1 completed: SUCCESS
[Pipeline] echo
#####
Production Pipeline Triggered #####
[Pipeline] }

```

- After Completion of testing pipeline, it will trigger the production pipeline,

```

Mon Feb 10 18:51:56
[Pipeline] sleep
Sleeping for 15 sec
[Pipeline] echo
http://13.43.221.99:30014
[Pipeline] echo
http://3.11.184.22:30014
[Pipeline]
[Pipeline] // script
Post stage
[Pipeline] script
[Pipeline] {
[Pipeline] echo
#####
Deployment to the Production Environment is SUCCESSFUL #####
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] deleteDir
[Pipeline]
[Pipeline] // script
[Pipeline]
[Pipeline] // node
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
#####
Pipeline Completed Successfully #####
[Pipeline] }
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withCredentials
[Pipeline] End of Pipeline
Finished: SUCCESS

```

- Production pipeline completed successful, display the urls of the production servers,
- Image will push to DockerHub Repository,

Screenshot of Docker Hub repository page for `sameer014/skmirza-medicure-img`:

- Tags:** latest (Image, less than 1 day, pushed about 1 hour ago), 1.0.1 (Image, less than 1 day, pushed about 1 hour ago).
- Docker commands:** `docker push sameer014/skmirza-medicure-img:tagname`
- Automated builds:** Available with Pro, Team and Business subscriptions. [Read more about automated builds](#).

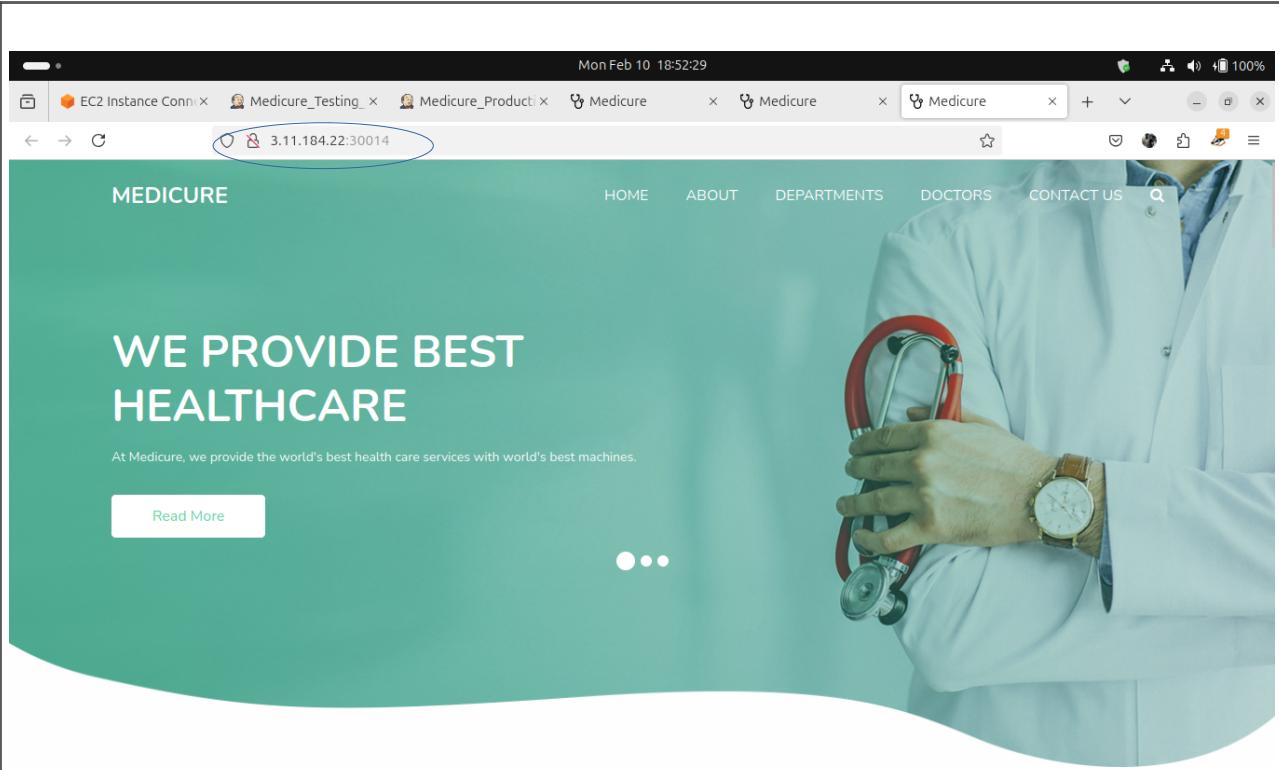
Repository overview (Incomplete)

- Production Server 01

Screenshot of a web browser showing the Medicure website at `13.43.221.99:30014`:

The page displays the Medicure logo and navigation menu (HOME, ABOUT, DEPARTMENTS, DOCTORS, CONTACT US). A large image of a doctor's hand holding a stethoscope is on the right. The main text on the left reads "WE PROVIDE BEST HEALTHCARE". A "Read More" button is visible.

- Production Server 02



- Additionally you can add the loadbalance, and map DNS to the production servers for one url,
- For Continuous Deploying, Your Can Create a webhook in github repository,

```
1> pipeline {  
2>     agent none
```

- Enable Github web hook trigger in Testing Pipeline then,

The screenshot shows the GitHub repository settings page for 'Medicure'. The 'Webhooks' section is active, displaying a configuration for a webhook pointing to 'http://35.179.53.99:8080/github... (push)'. A callout bubble highlights this configuration, indicating it triggers on push events.

- If any Change Done in the GitHub repository it will auto trigger the Testing then Production Pipeline,
- While Deploying to Production Server if any pods failed to deployment it will rollback to previous version,

The screenshot shows the Jenkins console output for the 'Medicure_Production_Pipeline'. The log indicates a successful deployment of the 'medicuredeploy' service. A callout bubble highlights the pod status table, which shows four pods running successfully across three different nodes.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
skmirza-medicure-deploy-69f9f7f985-2hrft	1/1	Running	0	6s	10.244.2.5	kube-worker02
skmirza-medicure-deploy-69f9f7f985-8qh8t	1/1	Running	0	6s	10.244.2.4	kube-worker02
skmirza-medicure-deploy-69f9f7f985-rd79x	1/1	Running	0	6s	10.244.1.5	kube-worker01
skmirza-medicure-deploy-69f9f7f985-w6767	1/1	Running	0	6s	10.244.1.4	kube-worker01

- If any Pods failed it will rollback to previous version,

Step - 5.0 :- For Monitoring Install Prometheus and Grafana in Your local host,

- After Installation, Check the Status of Prometheus and Grafana,

```
skmirza014@SkMirza-Ubuntu: ~
```

```
Mon Feb 10 20:39:04
```

```
skmirza014@SkMirza-Ubuntu: $ sudo systemctl status prometheus.service
```

```
● prometheus.service - Prometheus Server
   Loaded: loaded (/etc/systemd/system/prometheus.service; disabled; preset: enabled)
   Active: active (running) since Mon 2025-02-10 20:30:46 IST; 7min ago
     Docs: https://prometheus.io/docs/introduction/overview/
Main PID: 124844 (prometheus)
   Tasks: 10 (limit: 6783)
  Memory: 81.5M (peak: 117.1M)
    CPU: 1.755s
   CGroup: /system.slice/prometheus.service
           └─124844 /root/prometheus-2.53.3/prometheus --config.file=/root/prometheus-2.53.3/prometheus.yml
```

```
Feb 10 20:30:54 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:00:54.794Z caller=main.go:1391 level=info msg="updated GOGC old=>
Feb 10 20:30:54 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:00:54.794Z caller=main.go:1402 level=info msg="Completed loading >
Feb 10 20:30:54 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:00:54.794Z caller=main.go:1133 level=info msg="Server is ready to >
Feb 10 20:30:54 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:00:54.794Z caller=manager.go:164 level=info component="rule manag>
Feb 10 20:31:01 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:01:778Z caller=compact.go:576 level=info component="tsdb msg="<w>
Feb 10 20:31:01 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:01:779Z caller=head.go:1355 level=info component="tsdb msg="Hea>
Feb 10 20:31:03 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:03:050Z caller=compact.go:576 level=info component="tsdb msg="<w>
Feb 10 20:31:03 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:03:059Z caller=head.go:1355 level=info component="tsdb msg="Hea>
Feb 10 20:31:03 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:03:059Z caller=checkpoint.go:101 level=info component="tsdb msg>
Feb 10 20:31:03 SKMirza-Ubuntu prometheus[124844]: ts=2025-02-10T15:01:03:329Z caller=head.go:1317 level=info component="tsdb msg="WAL>
```

```
skmirza014@SkMirza-Ubuntu: $ sudo systemctl status grafana-server.service
```

```
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; preset: enabled)
   Active: active (running) since Mon 2025-02-10 20:31:40 IST; 7min ago
     Docs: http://docs.grafana.org
Main PID: 125252 (grafana)
   Tasks: 21 (limit: 6783)
  Memory: 207.8M (peak: 260.3M)
    CPU: 13.681s
   CGroup: /system.slice/grafana-server.service
           └─125252 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini - -pidfile=/run/grafana/grafana-server.pid
```

```
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=resource-server t=2025-02-10T20:32:05.248295324+05:30 level=warn msg="failed to >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.255023347+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.257808565+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.260056804+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.263140819+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.269717694+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.276513103+05:30 level=info msg="Adding >
Feb 10 20:32:05 SKMirza-Ubuntu grafana[125252]: logger=grafana-apiserver t=2025-02-10T20:32:05.279658528+05:30 level=info msg="Adding >
```

- Prometheus and Grafana Services Running successful, You can verify in browser,
- Prometheus Web Service,

The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with tabs for EC2 Instance, Medicure_Prod, Jenkins, sameer014/skr, SA2410011 CP, Prometheus T, and Grafana. Below the navigation bar, the URL is set to localhost:9090. The main content area has a dark header with the Prometheus logo and the text "Prometheus". Below the header, there are several checkboxes: "Use local time", "Enable query history", "Enable autocomplete", "Enable highlighting", and "Enable linter". A search bar contains the placeholder "Expression (press Shift+Enter for newlines)". To the right of the search bar is an "Execute" button. Below the search bar, there are two tabs: "Table" (selected) and "Graph". Under the "Table" tab, there are buttons for "Evaluation time" and arrows for navigating through results. A message "No data queried yet" is displayed. At the bottom left, there's an "Add Panel" button.

- **Grafana Web Service,**

The screenshot shows the Grafana web application running in a browser. The URL is `localhost:3000/?orgId=1&from=now-6h&to=now&timezone=browser`. The interface has a dark theme. On the left is a sidebar with links for Home, Dashboards, Explore, and Alerting. The main content area has several panels: 'Basic' (with instructions to finish setup), 'TUTORIAL' (linking to 'DATA SOURCE AND DASHBOARDS' and 'Grafana fundamentals'), 'DATA SOURCES' (linking to 'Add your first data source'), and 'DASHBOARDS' (linking to 'Create your first dashboard'). At the bottom right is a yellow banner with the text 'Why observability needs Grafana'.

- To get data install `node_exporter` on the test servers and production servers,
- Check the status of `node_exporter` in the test servers and production servers,

Step – 5.1 :- Add all targets in prometheus.yaml file,

```

Mon Feb 10 21:42:06
root@SkMirza-Ubuntu:~/prometheus-2.53.3
skmirza@ip-140-1-232:~          root@SkMirza-Ubuntu:~/prometheus-2.53.3
GNU nano 7.2                      prometheus.yaml *

alerting:
  alertmanagers:
    - static_configs:
      - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9690"]
      - targets: ["18.168.88.78:9100"]
      - targets: ["13.43.221.99:9100"]
      - targets: ["3.11.184.22:9100"]

```

The terminal window shows the configuration of the `prometheus.yaml` file. It includes sections for alerting, rule_files, scrape_configs, and static_configs. The `static_configs` section lists four targets: `localhost:9690`, `18.168.88.78:9100`, `13.43.221.99:9100`, and `3.11.184.22:9100`.

- Save the file and restart the prometheus service,
 - To verify execute up in prometheus browser,

The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with links for EC2 Instance C, Medicure_Prod, Jenkins, sameer014/skn, SA2410011 CP, Prometheus, and Grafana. Below the navigation is a header bar with a back/forward button, a search bar containing "localhost:9090/graph?g0.expr=up&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range=" (with a progress bar at 90%), and a user icon.

The main area has tabs for "Table" and "Graph". The "Table" tab is selected, showing a table with four rows of data. Each row represents a metric instance with the label "up". The columns are "Evaluation time" (empty), "Series" (the metric definition), and "Value" (either 1 or 0). The last column also includes a "Remove Panel" link.

At the bottom left, there's a blue "Add Panel" button. On the right side, there are several small icons for navigating between panels.

Step – 5.2 :- Add this data to grafana,

The screenshot shows the Grafana home page. The left sidebar contains navigation links for Home, Bookmarks, Starred dashboards, Dashboards, Playlists, Snapshots, Library panels, Shared dashboards, Explore (Metrics, Logs, Profiles), Alerting (Alert rules, Contact points, Notification policies, Silences, Active notifications), and a general search bar at the top.

The main content area features a "Welcome to Grafana" message. Below it, a "Basic" section provides a quick start guide for new users. To the right, a "TUTORIAL" box covers "DATA SOURCE AND DASHBOARDS" and "Grafana fundamentals". A circular callout "DATA SOURCES" encourages adding a first data source. Another panel "DASHBOARDS" guides users through creating their first dashboard. At the bottom, sections for "Dashboards" and "Latest from the blog" are visible.

- Add the data, and create a dashboard.

➤ For CPU utilization

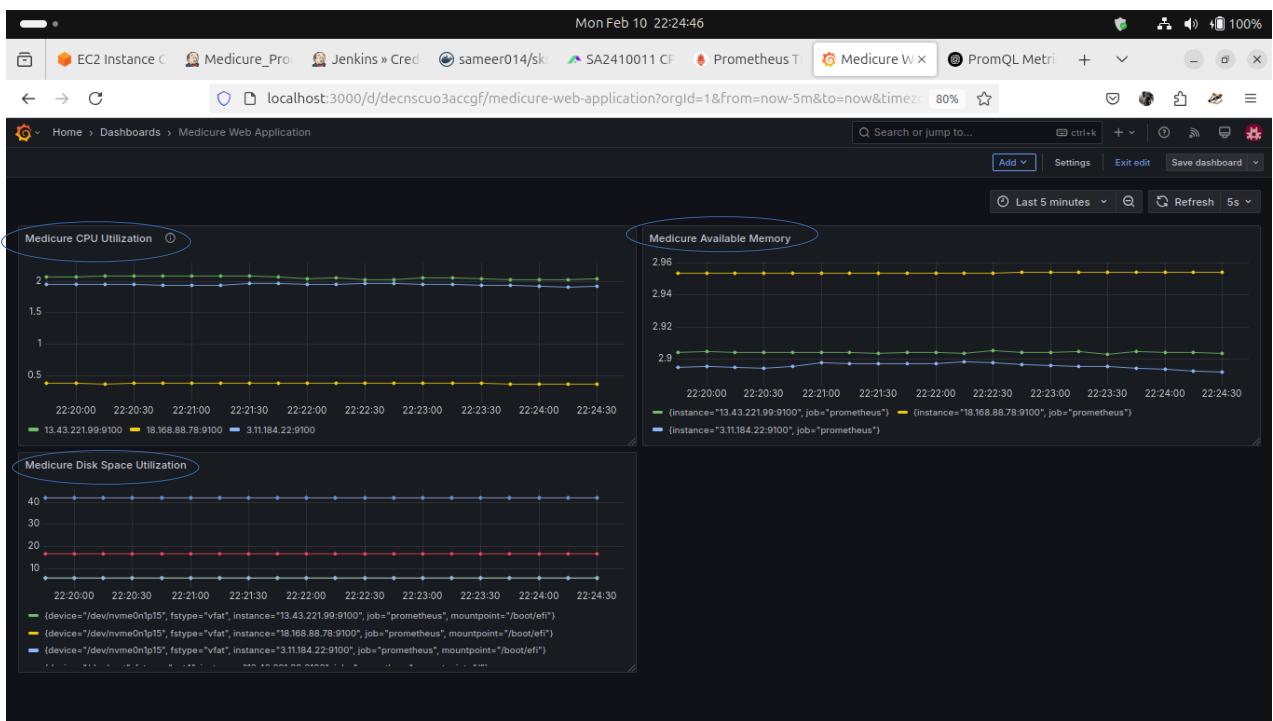
- $100 - (\text{avg by(instance)}(\text{rate(node_cpu_seconds_total\{mode="idle"\}[5m])) * 100)$

➤ Disk Space Utilization

- $100 - ((\text{node_filesystem_avail_bytes\{fstype!~"tmpfs"\}} * 100) / \text{node_filesystem_size_bytes\{fstype!~"tmpfs"\}})$

➤ Total Available Memory

- $(\text{node_memory_MemAvailable_bytes} / 1024 / 1024 / 1024)$



Hence automate web application build and deployment process is Completed Successful with Continuous Deployment / Continuous Delivery & Continues Monitoring.