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. **Docke**r - 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.Ansible-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"
    ]
  }
}
```

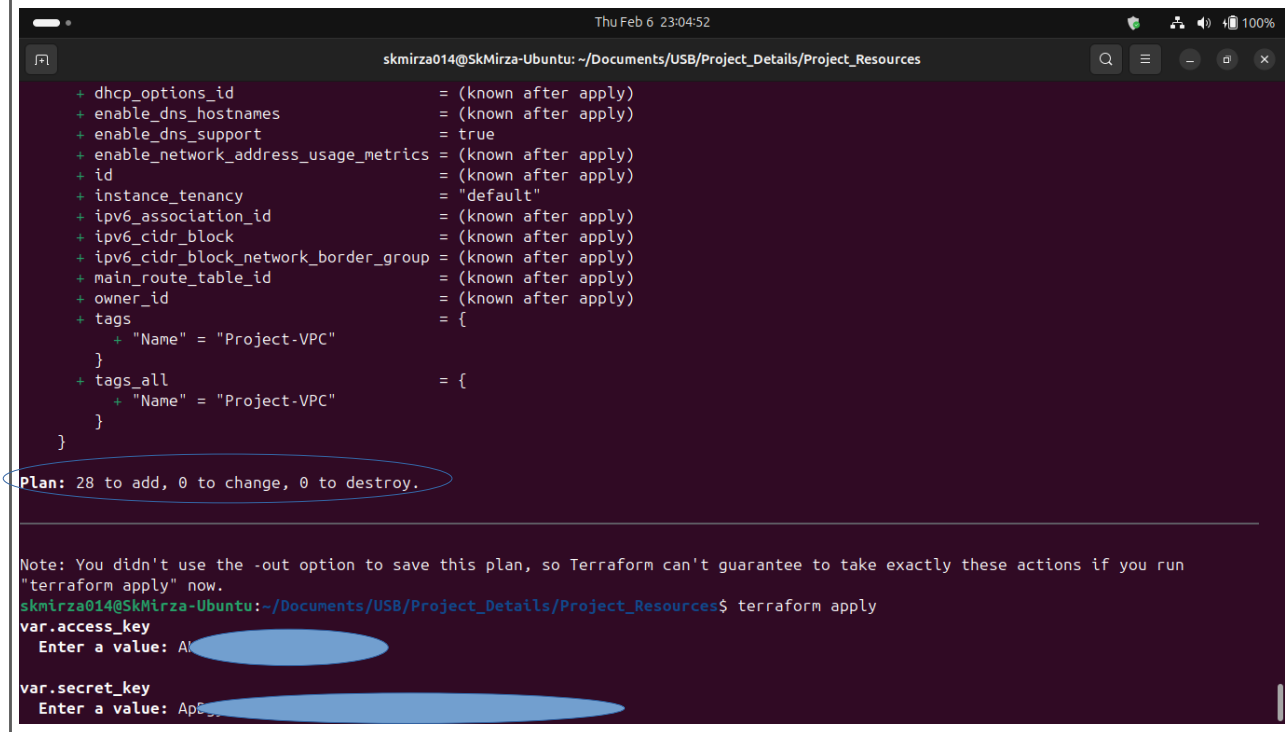
- Attaching Elastic IPs to serevrs

#Associate EIPs with instances

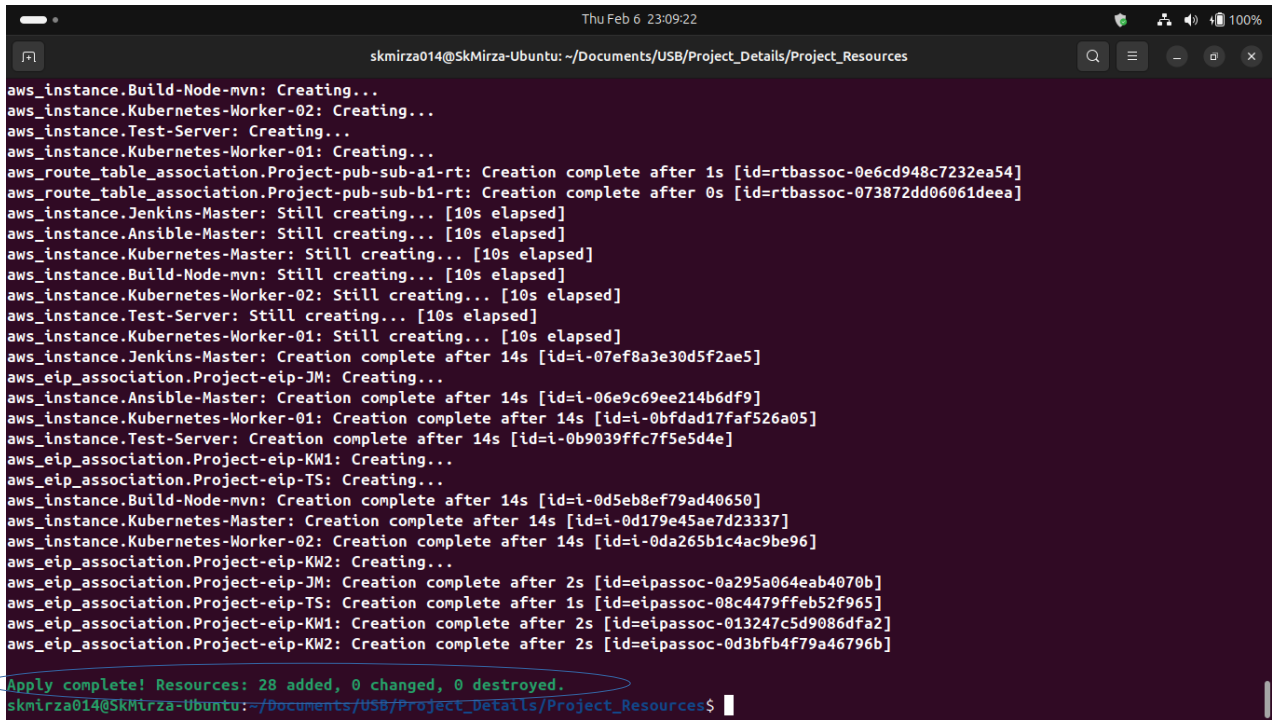
```
resource "aws_eip_association" "Project-eip-JM" {
        instance_id = aws_instance.Jenkins-Master.id
        allocation_id = aws_eip.Project-eip-JM.id
}

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,

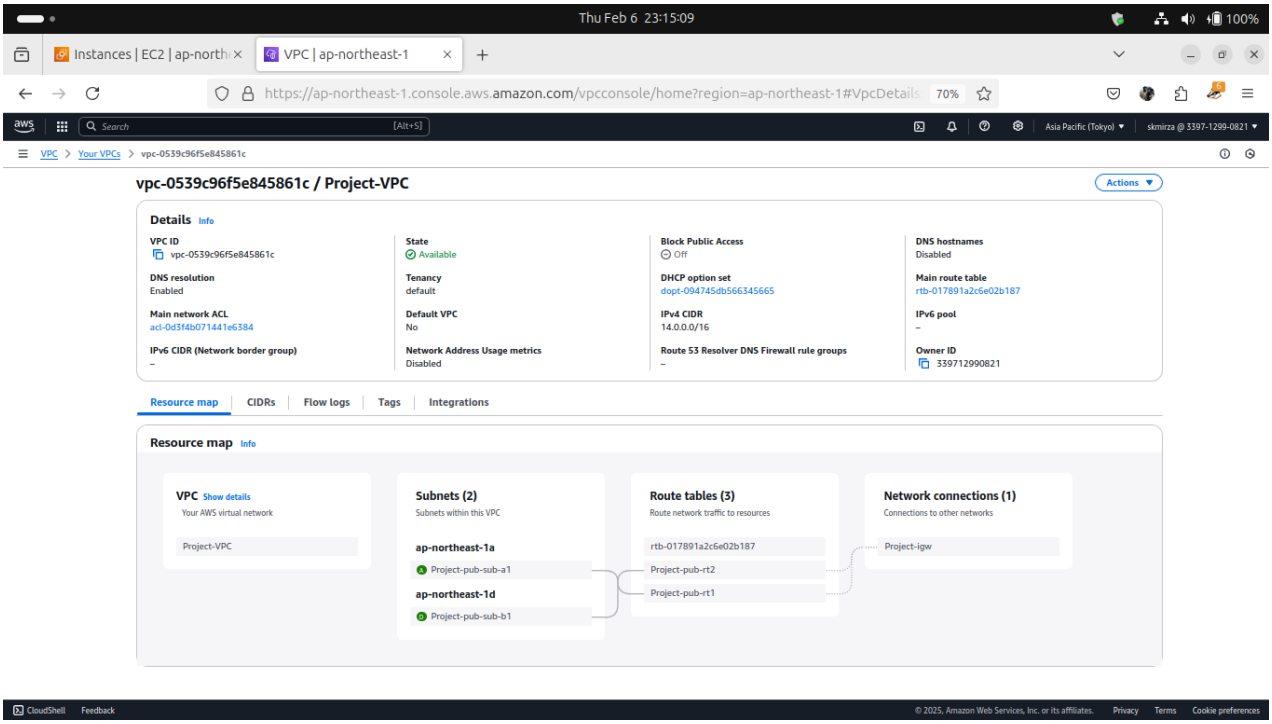
    - terraform init
    - terraform plan
    - terraform apply



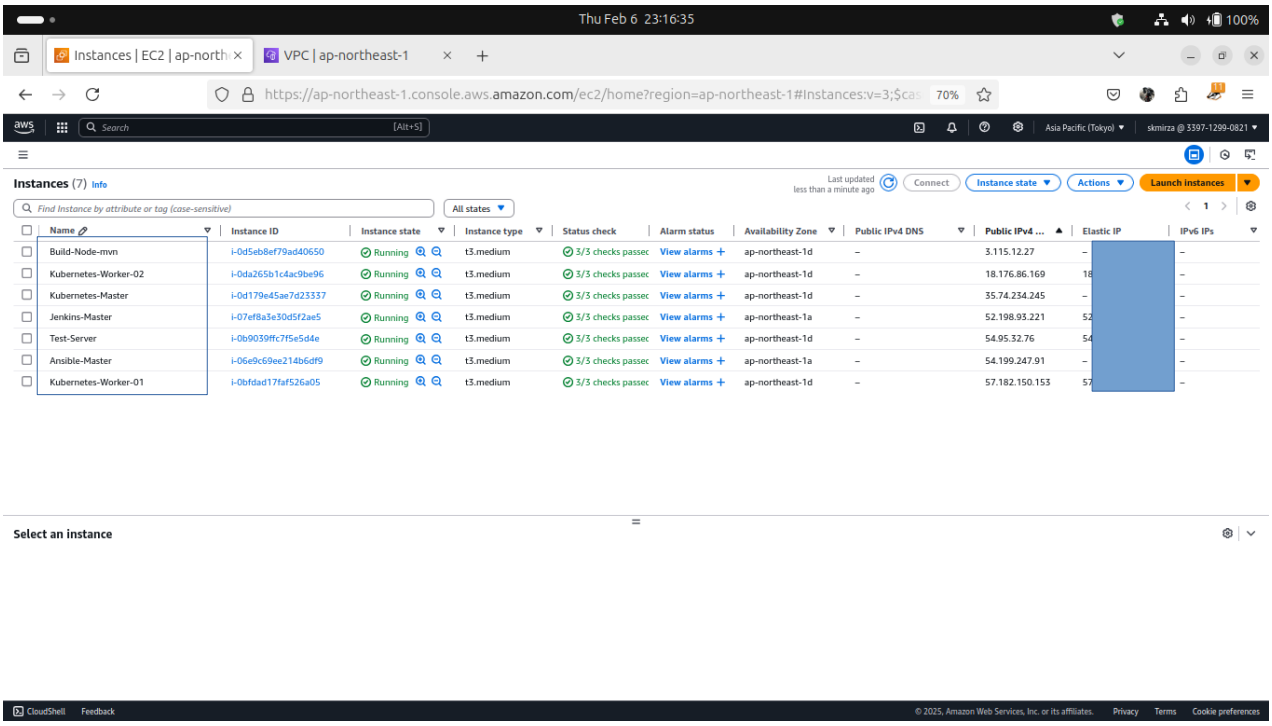
- In Run time you have to provide Access-key and Secret-key, 28 resources will created in your AWS Cloud,



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

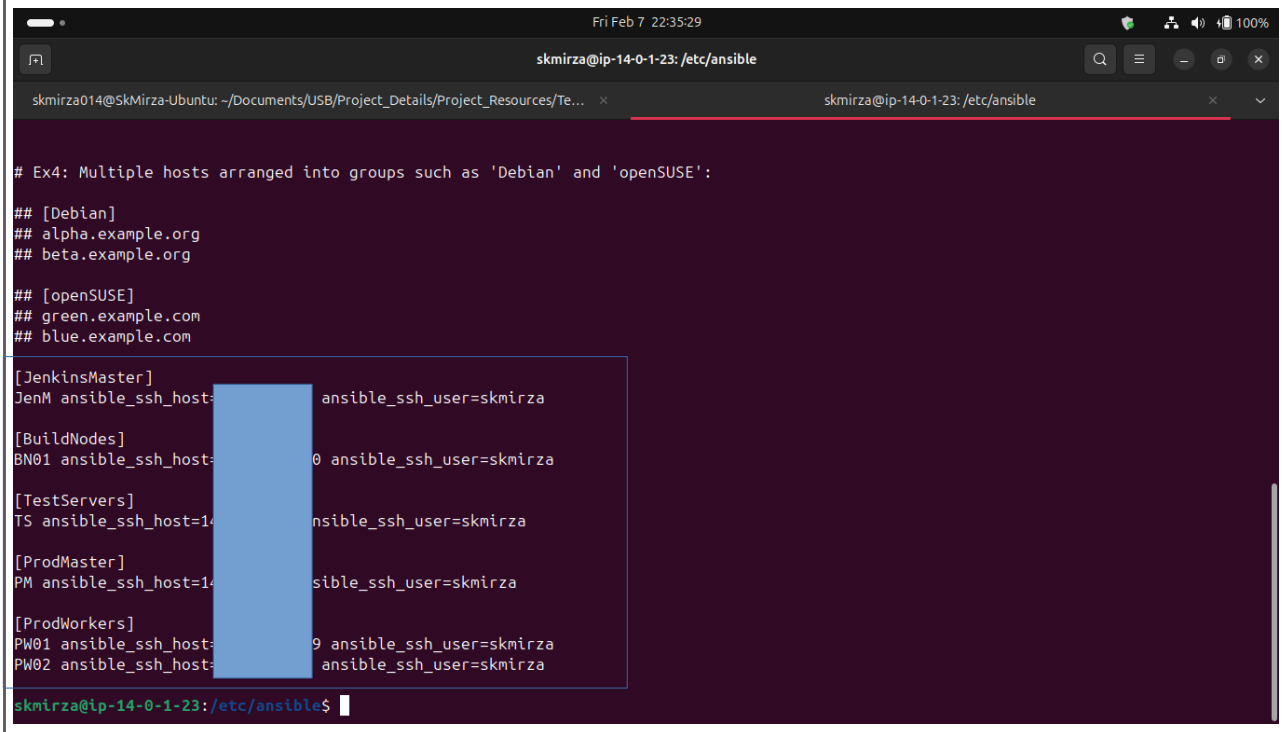


- Checking Creation of Servers,

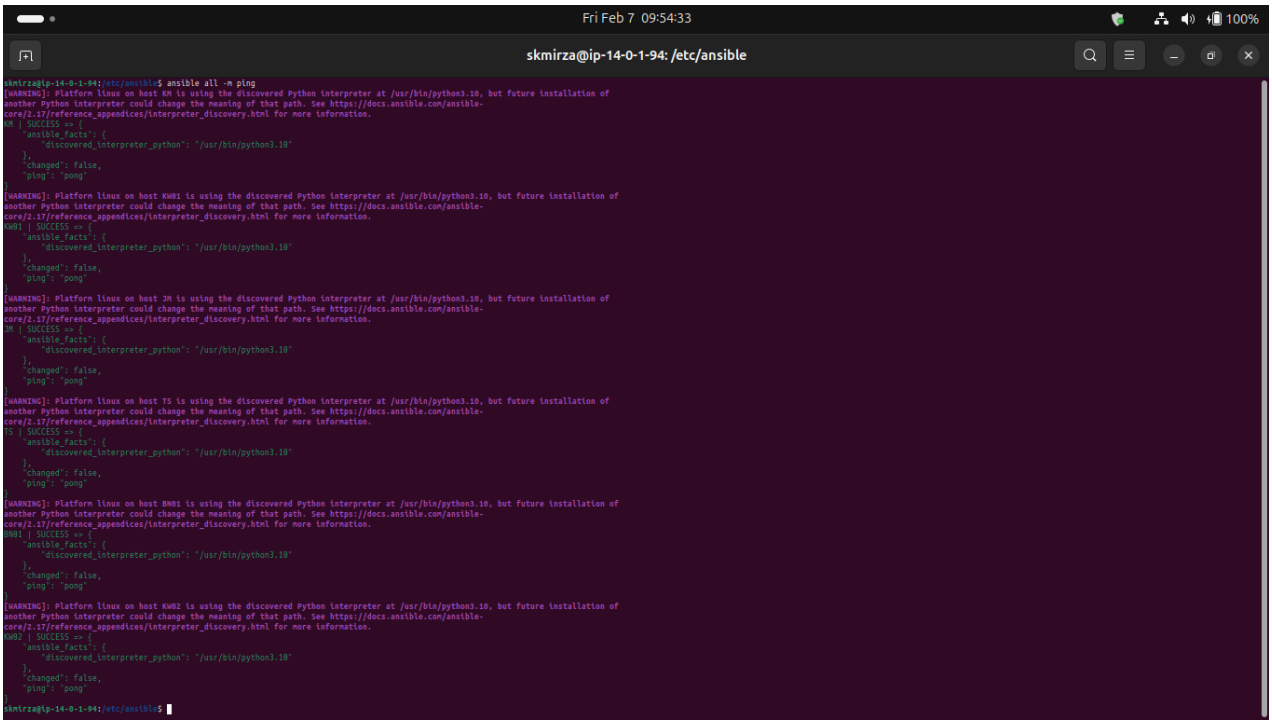


<u>Step – 2.1 :-</u> Now Configuring All the server Using Ansible,

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



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



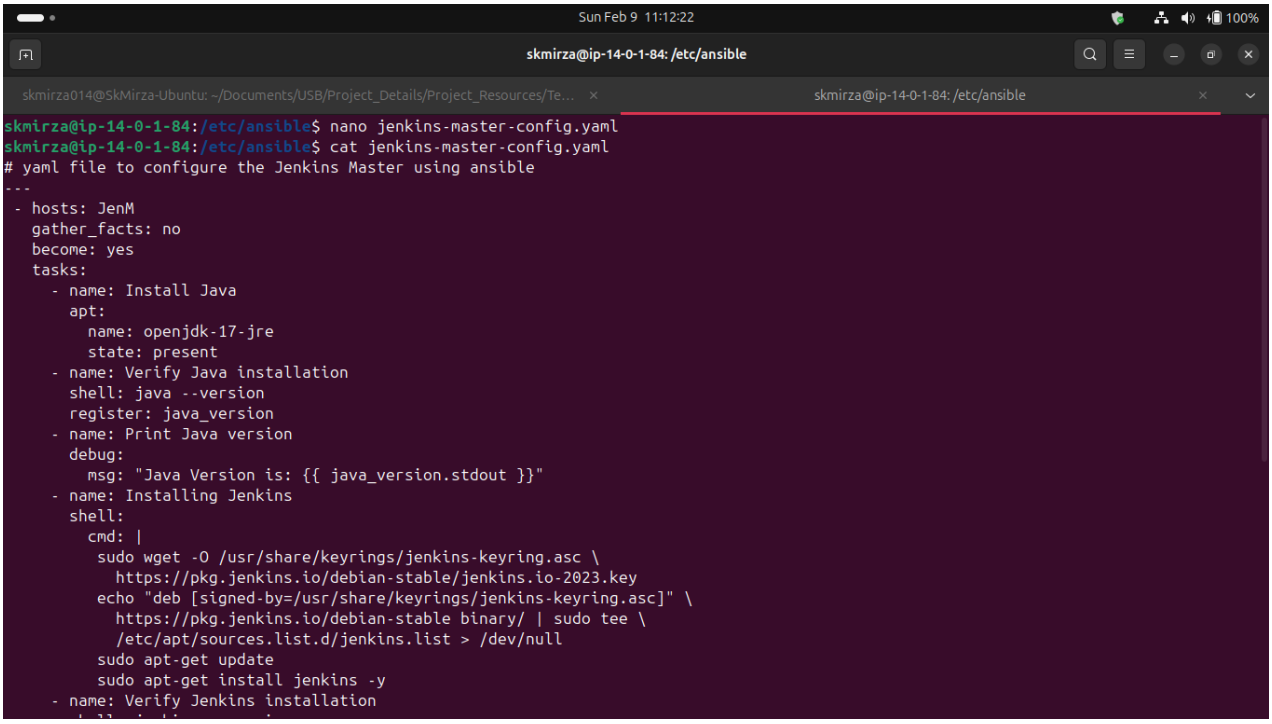
- All the servers connected to ansible master,

<u>Step – 2.2 :-</u> Now Configuring Jenkins-Master Node Using Ansible,

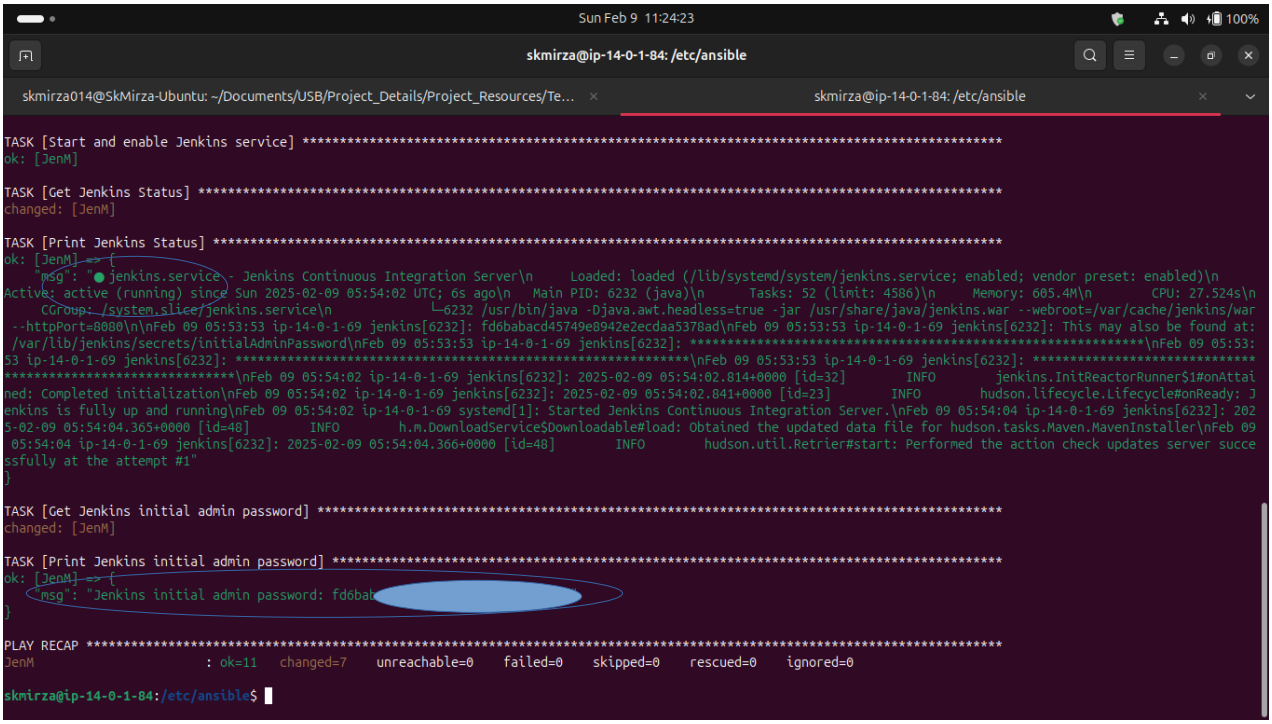
- Create a “jenkins-master-config.yaml” file and paste the following script,

```
# 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:
```

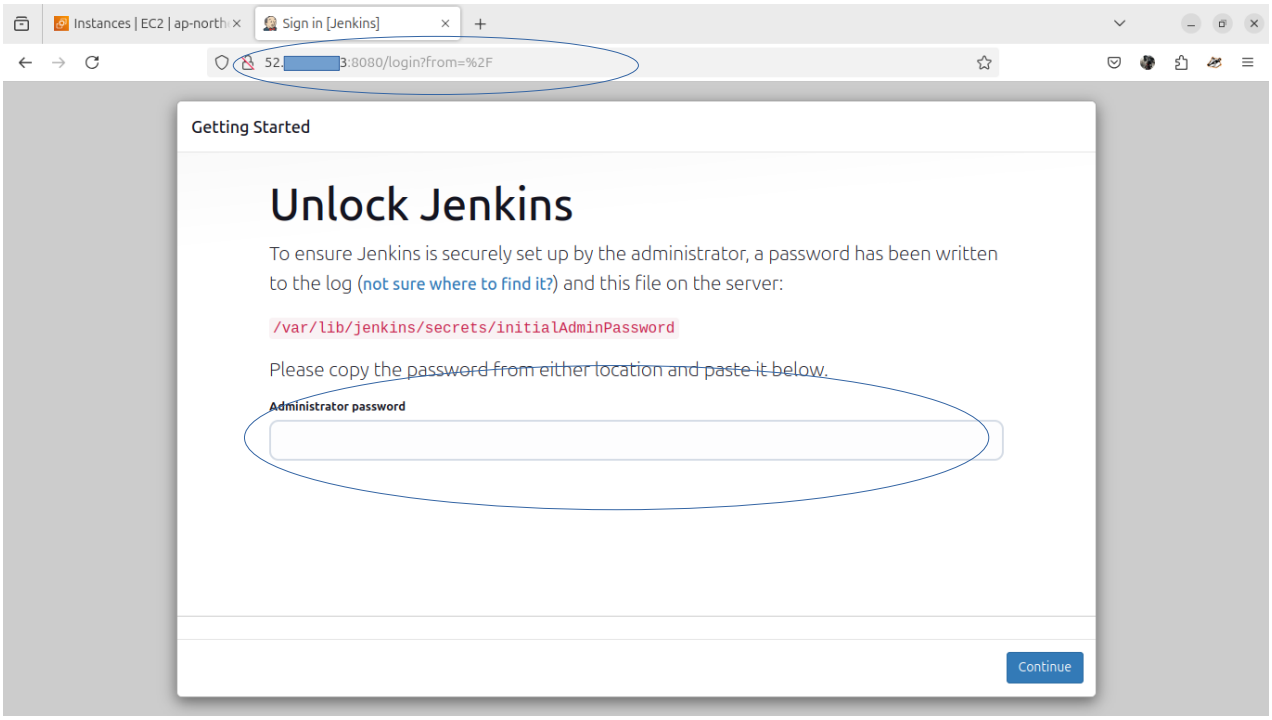
```
    msg: "{{ jenkins_status.stdout }}"
- name: Get Jenkins initial admin password
  shell: cat /var/lib/jenkins/secrets/initialAdminPassword
  register: jenkins_password
- name: Print Jenkins initial admin password
  debug:
    msg: "Jenkins initial admin password: {{ jenkins_password.stdout }}"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



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



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



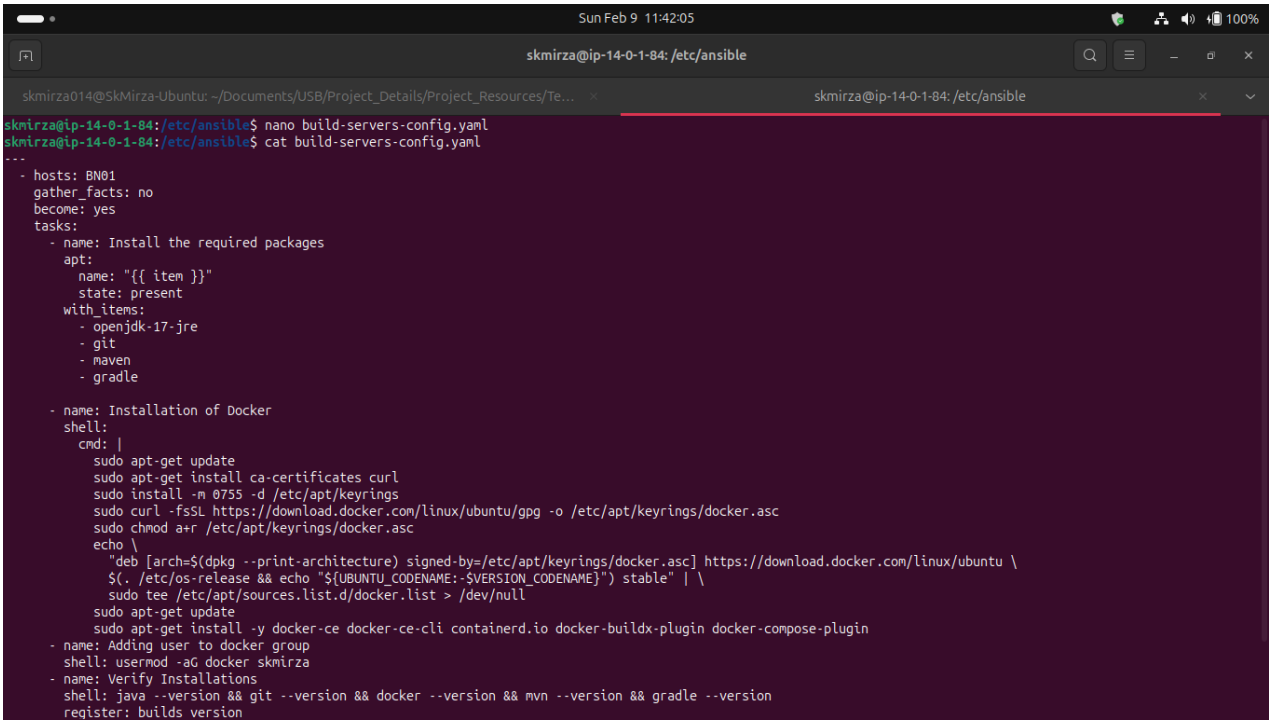
- paste the jenkins initial admin password, and config it, install all requied pulgins,

<u>Step – 2.3 :-</u> Now Configuring Build-Server Node Using Ansible,

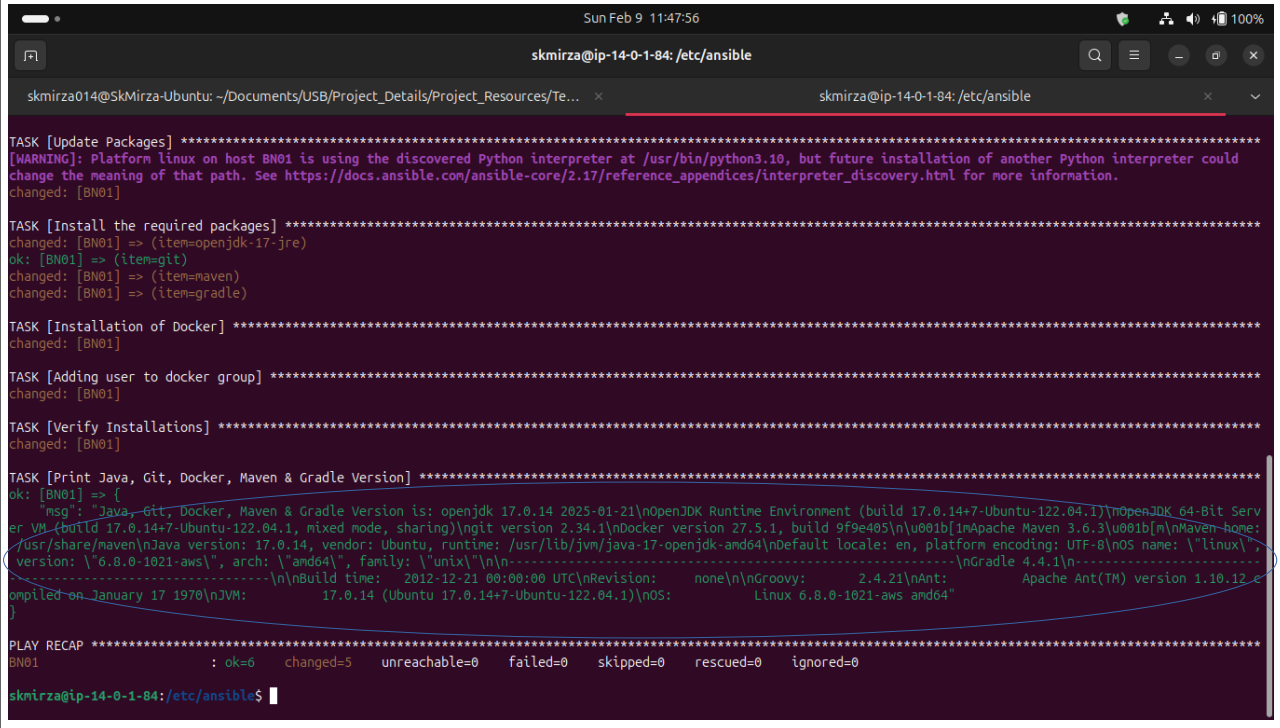
- Create a “build-servers-config.yaml” file and paste the following script,

```
# 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 }}"
```



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



- Build tools successful installed in Build Servers,

<u>Step – 2.4 :-</u> Now Configuring Kubernetes-Master Node Using Ansible,

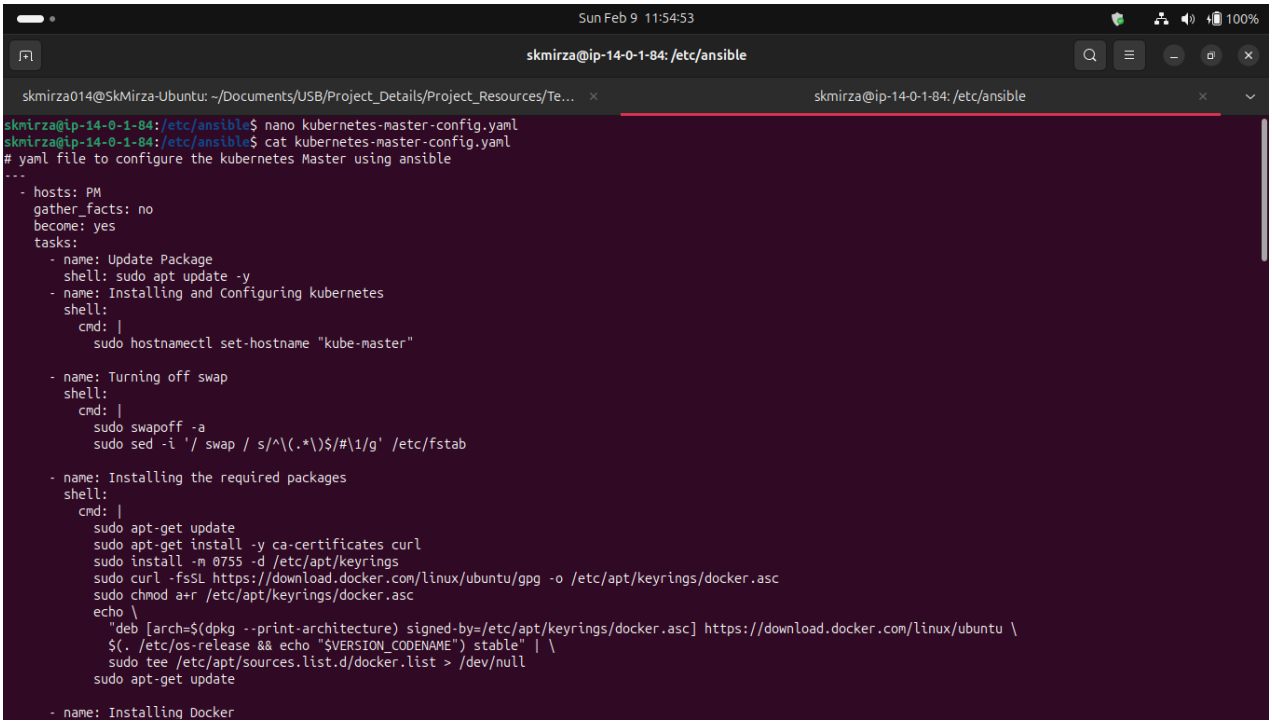
- Create a “kubernetes-master-config.yaml” file and paste the following script,

```
# 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
```

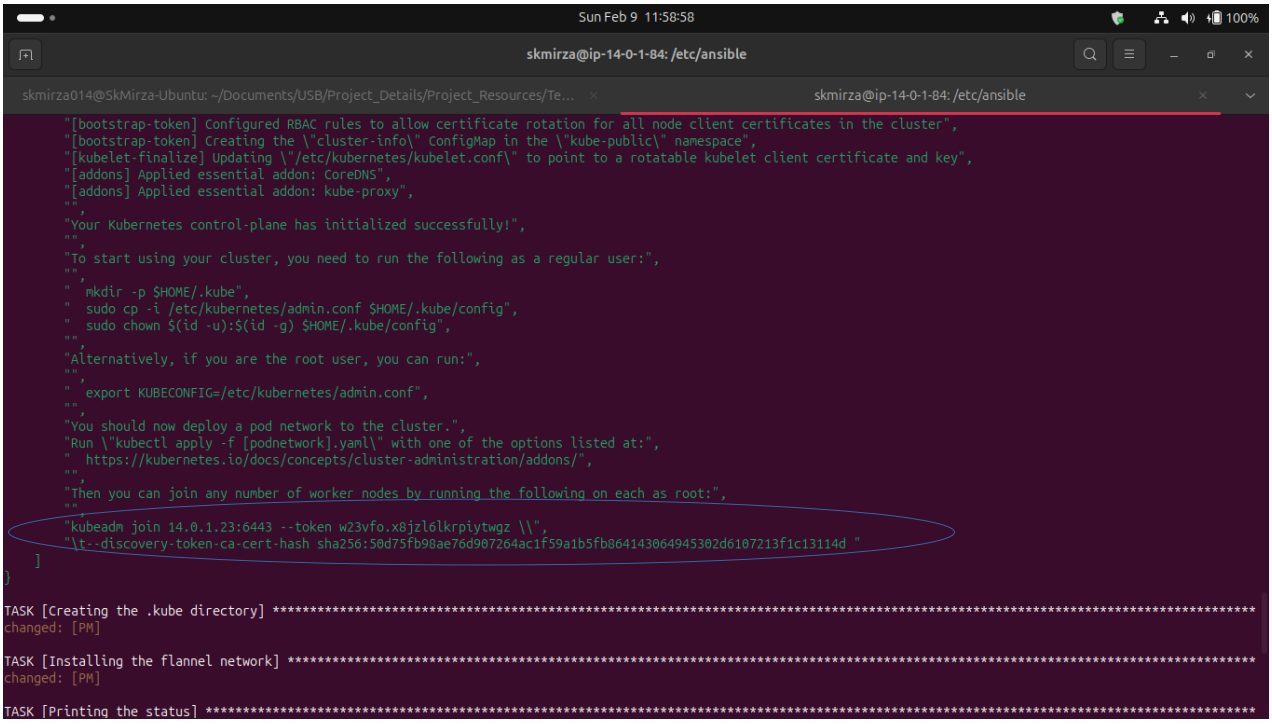
```
    - name: Configuring Docker
      shell:
        cmd: |
          sudo modprobe overlay
          sudo modprobe br_netfilter
          cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
          overlay
          br_netfilter
          EOF

    - 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: |
```

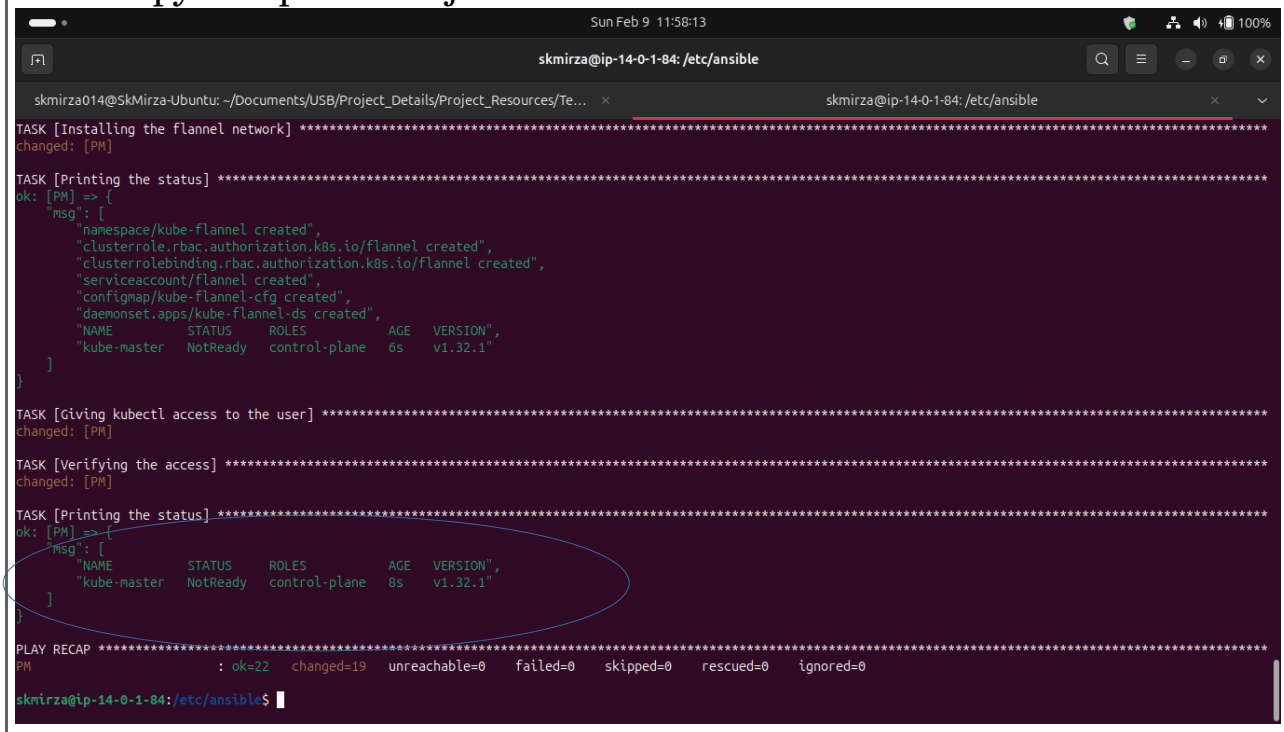
```
        kubectl apply -f
https://github.com/coreos/flannel/raw/master/Documentation/kube-
flannel.yml
        kubectl get nodes
      register: flannel_install
    - name: Printing the status
      debug:
        msg: "{{ flannel_install.stdout_lines }}"

    - 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 }}"
```



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



- Copy and paste the join command for kubernetes worker nodes



- Kubernetes installed successful in kubernetes-master-node,

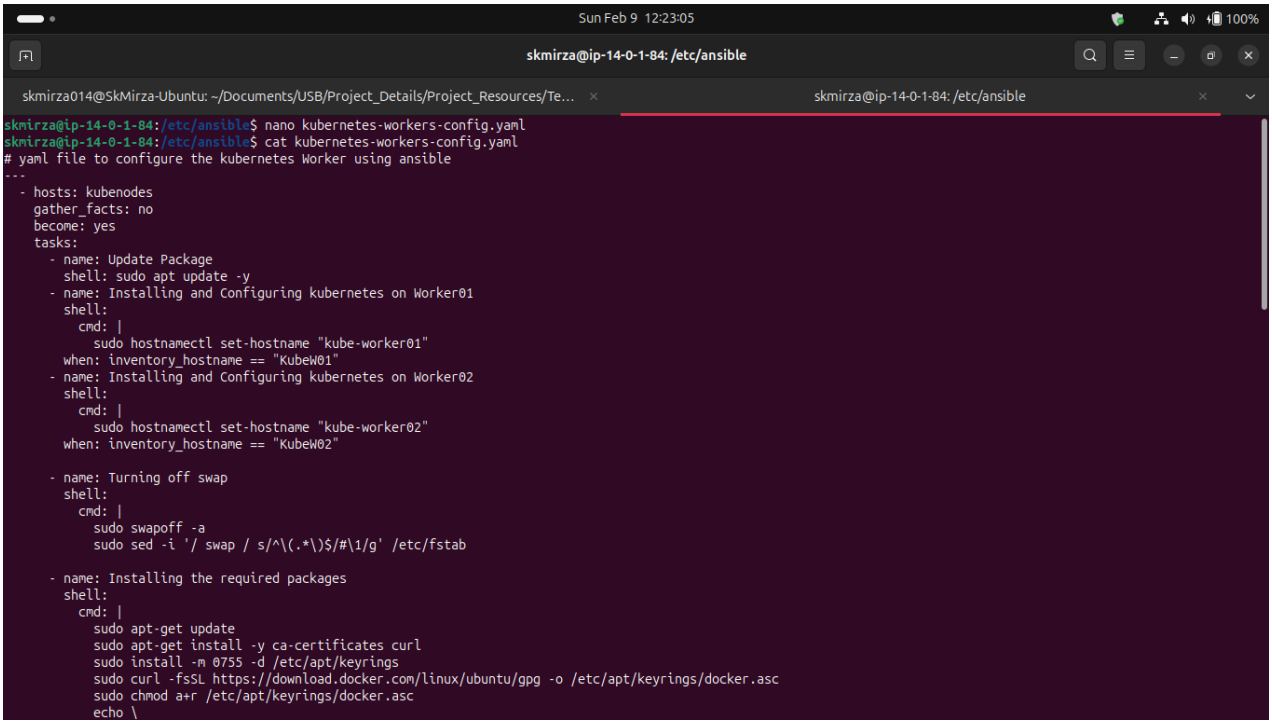
<u>Step – 2.5 :-</u> Now Configuring Kubernetes-Workers Node Using Ansible,

- 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 fileclear
   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/^\(.*\)$/#\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
```

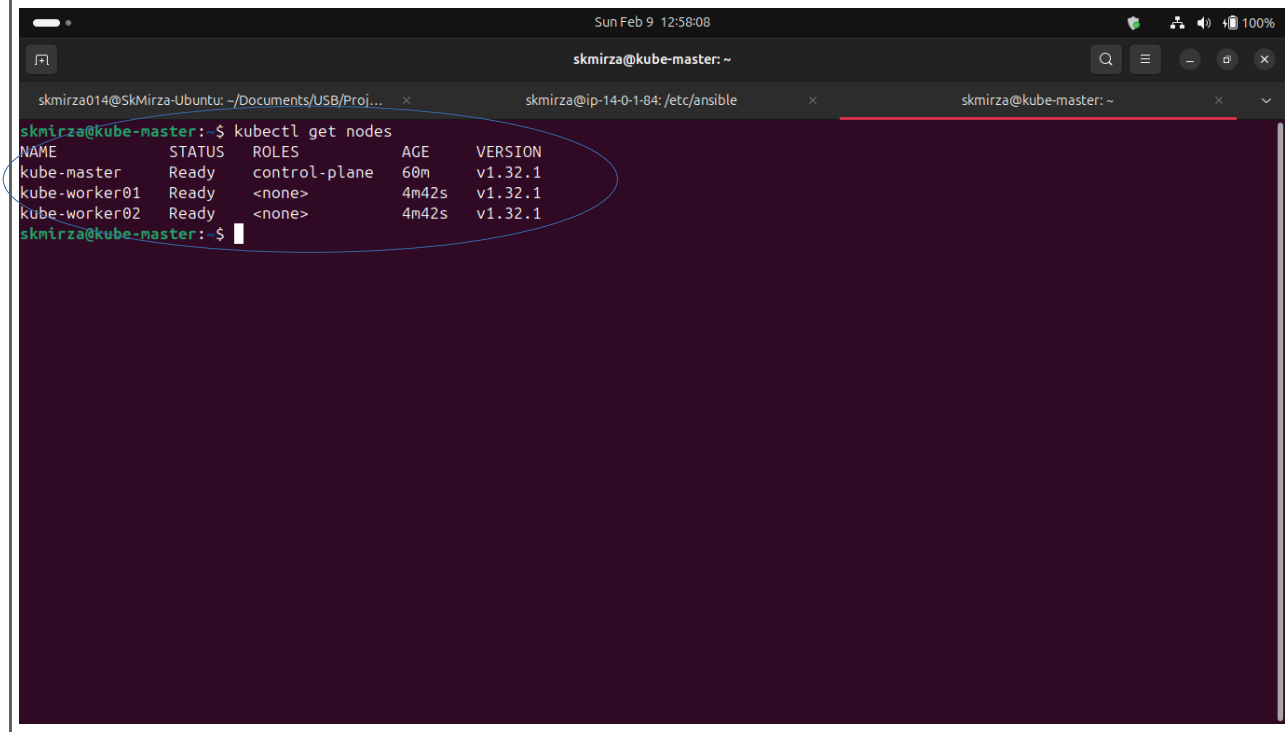
```
    - name: Installing Docker
      shell:
        cmd: |
          sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-
buildx-plugin docker-compose-plugin

    - 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: 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
```



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

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



- Configuring kubernetes workers completed successful,

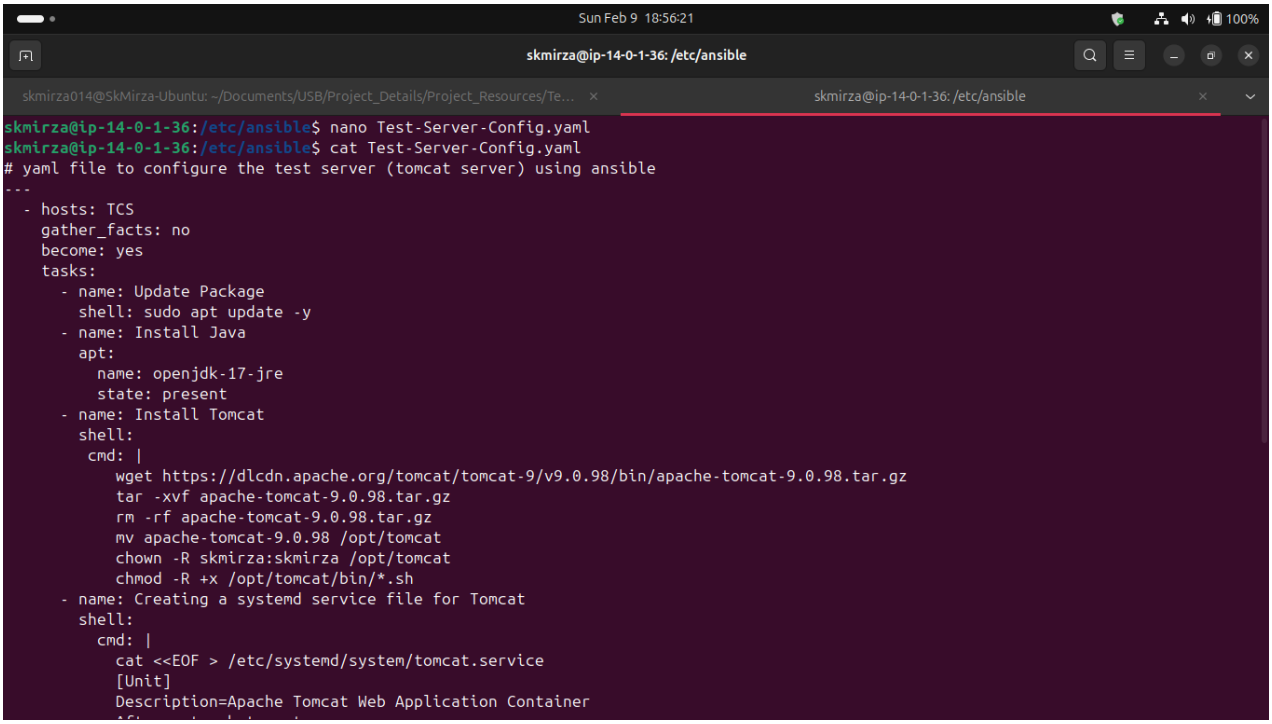
Step – 2.5 :- Now Configuring Test-Server Node Using Ansible,

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

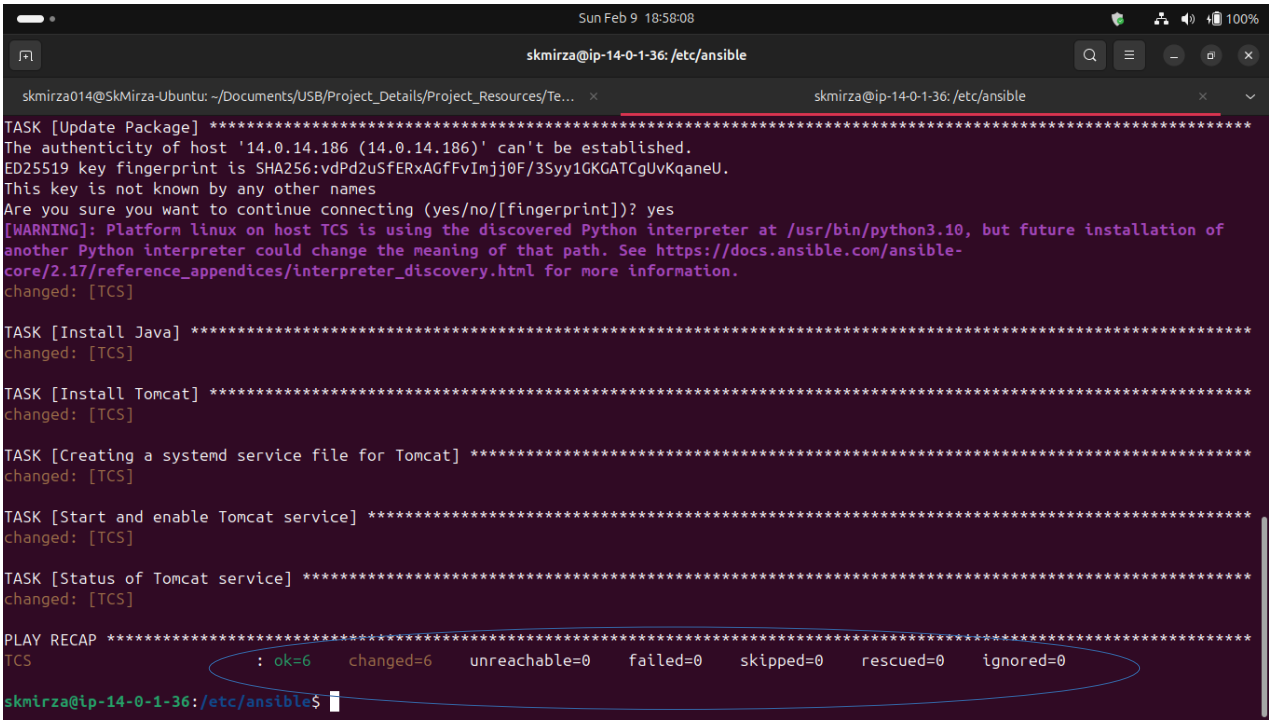
 # yaml file to configure the test server (tomcat server) using ansible

```
---
 - 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
```

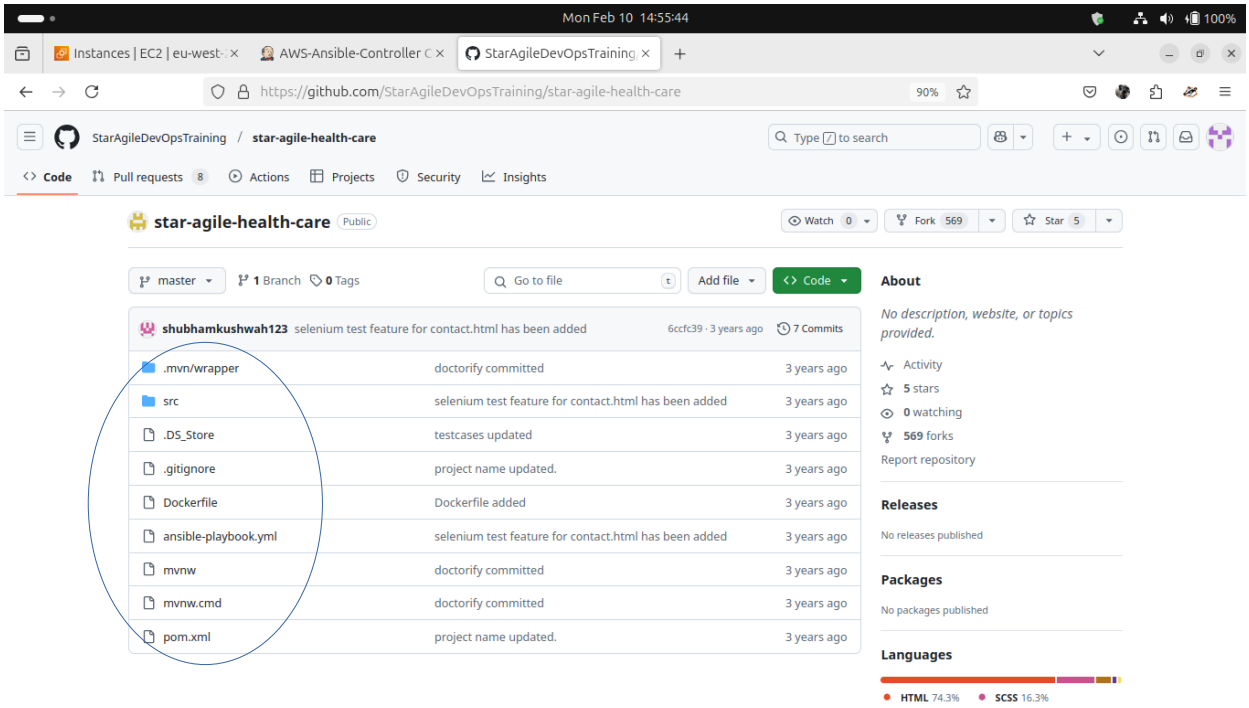


- Now execute “ ansible-playbook test-server-config.yaml ”



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

<u>Step – 3.1 :-</u> 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
```

```
            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
```

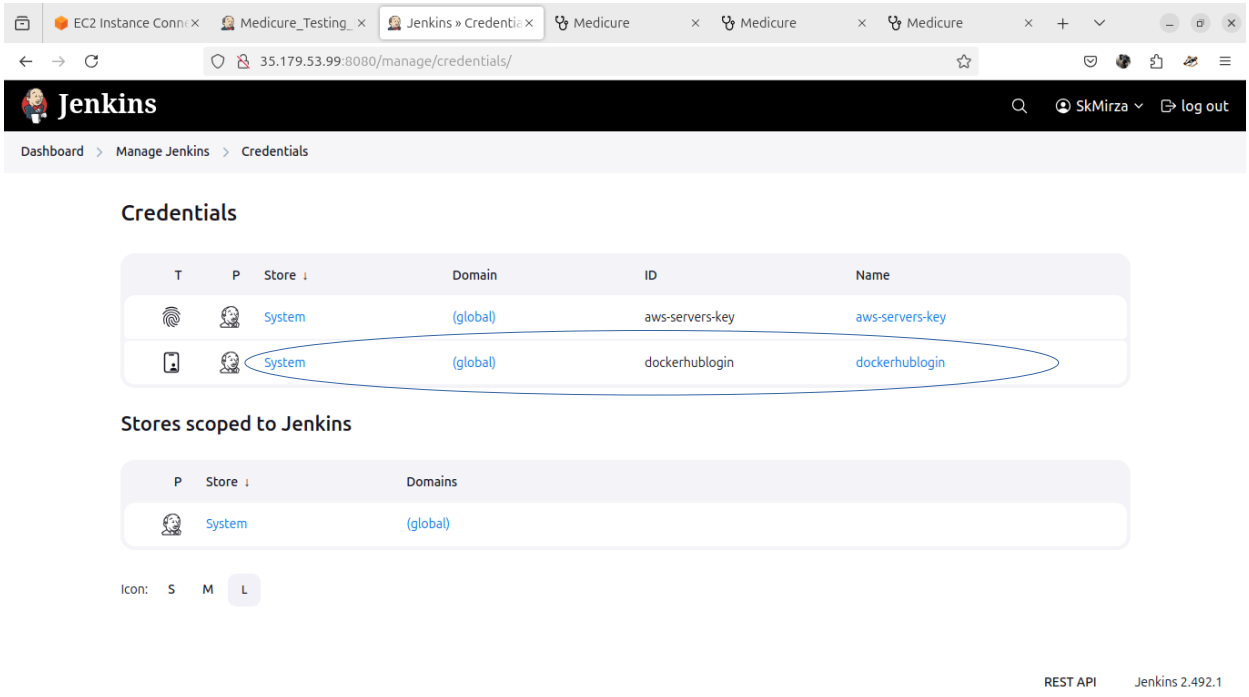
Step – 3.7 :- Add the Following Environment variables in node level,

#Variable

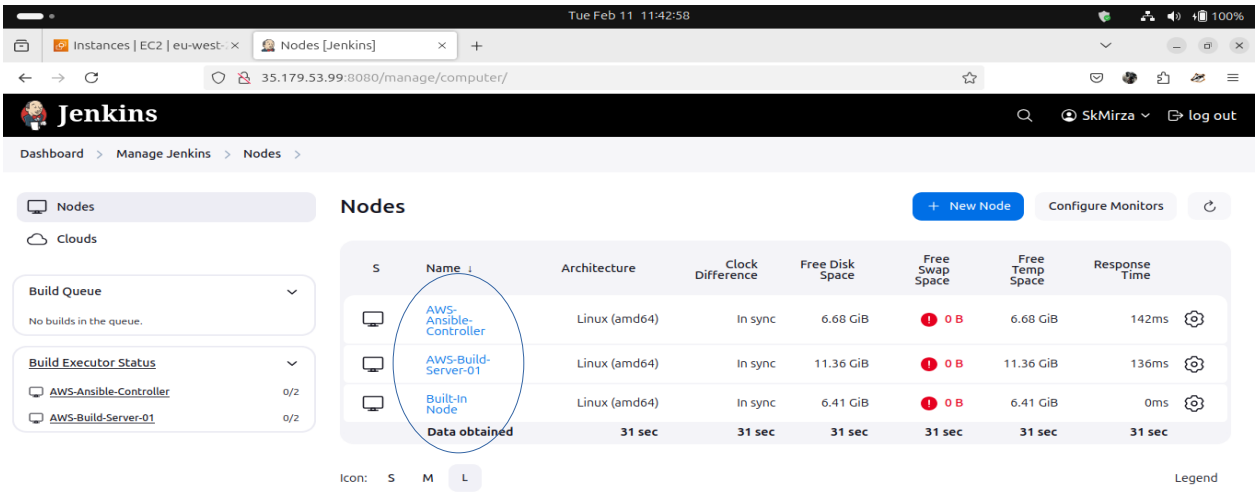
```
App_Name = medicure
Container_Name = ${User_Name}-${App_Name}-container
Default_Ver = 1.0.0
Deploy_Name = ${User_Name}-${App_Name}-deploy
KubeM_Pvt_IP = <Pvt_IP>
KubeW01_Pub_IP = <Pub_IP>
```

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,



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



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()
```

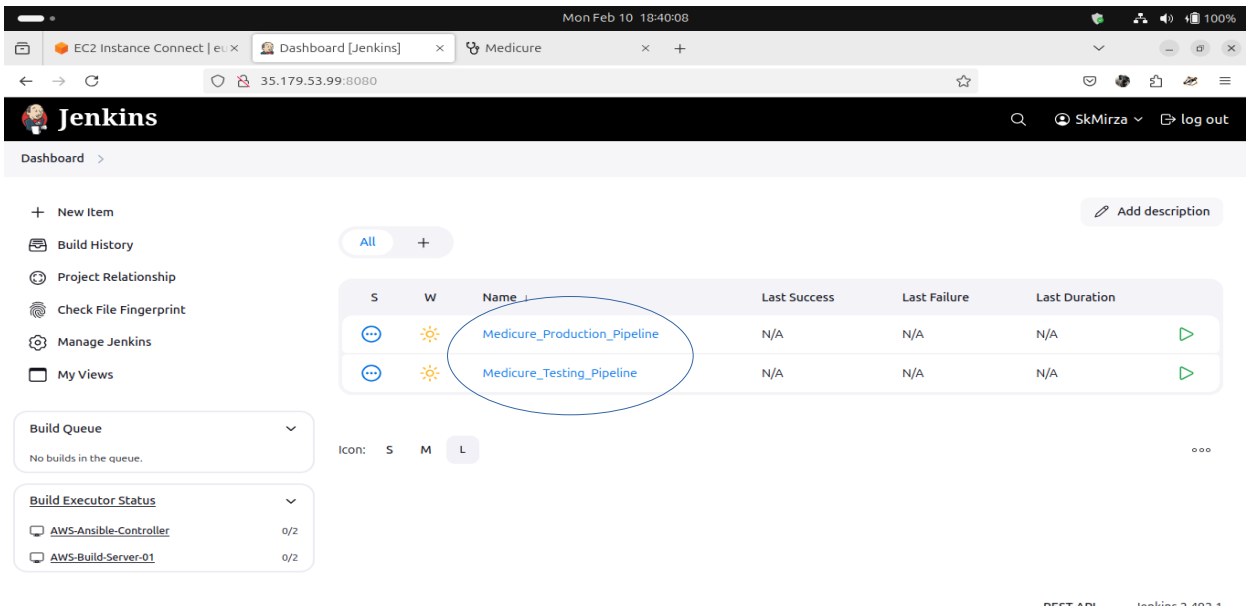
```
if (versionedImageExists1) {
echo 'Docker Image with the specified version already exists. Removing
the existing image...'
sh """docker rmi -f $DOCKERHUB_CREDENTIALS_USR/$User_Name-
$App_Name-img:${params.new_ver}"""
} else {
echo 'Docker Image with the specified version does not exist...'
}
sh 'docker logout'
}
}
failure{
script{
echo '########## Docker Push is FAILED ##########'
}
}
}
}
stage ('Deploy_Main'){
agent {
label 'AC'
}
when{
expression { Docker_Push == true }
}
steps {
echo 'Deploying the Docker Image to the Production Environment'
script {
echo '*********Deploying the Docker Image to the Production
Environment*********'
// Copy the *.yaml file to the Main Server using scp command
sh """ scp $User_Name@$Build01_Pvt_IP:$BN01_Path/*.yaml
$User_Name@$KubeM_Pvt_IP:/home/$User_Name/ """
echo "**********Copyed yaml file to Main Server**********"
```

```
// 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 medicuredeploy.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 medicuredeploy.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
```

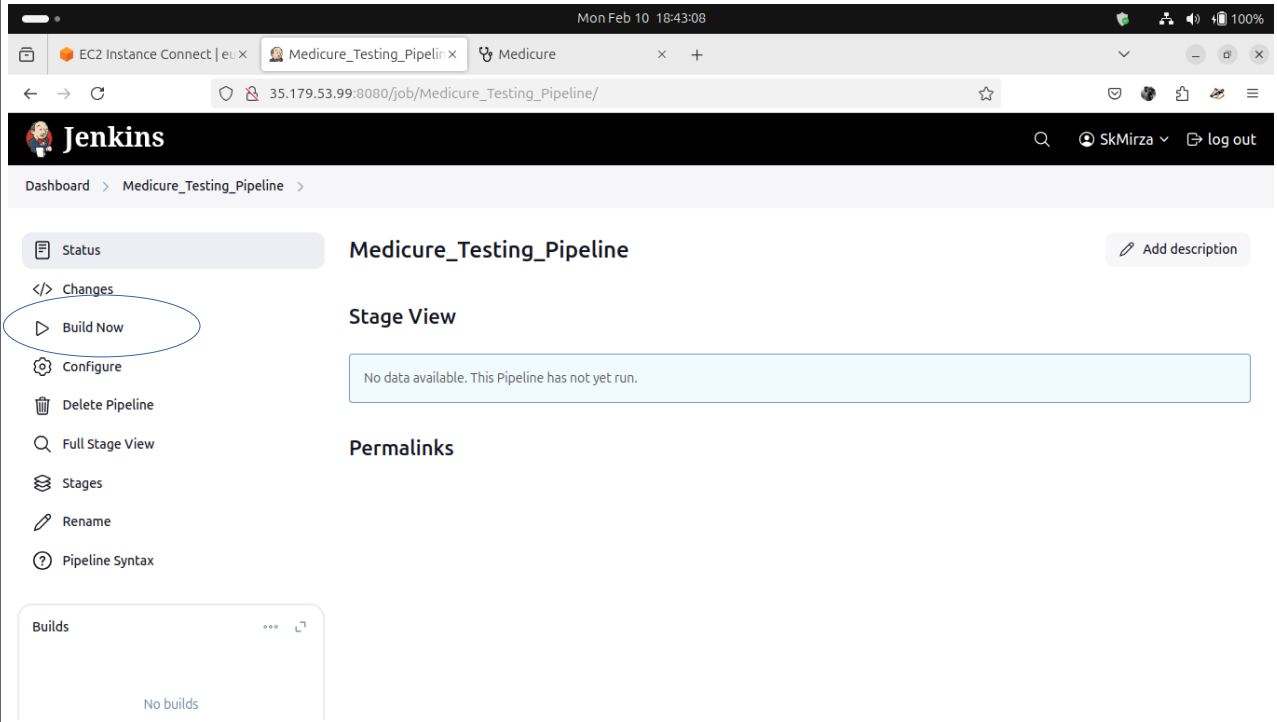
```
for (status in podStatuses) {
echo "Pod status: ${status}"
if (status == 'ErrImagePull' || status == 'Pending') {
rollback = true
break
}
}
if (rollback) {
echo '**********Pod Status is ErrImagePull or Pending, So Rolling Back the Deployment**********'
sh """ssh $User_Name@$KubeM_Pvt_IP kubectl rollout 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 ##########'
}
}
}
```

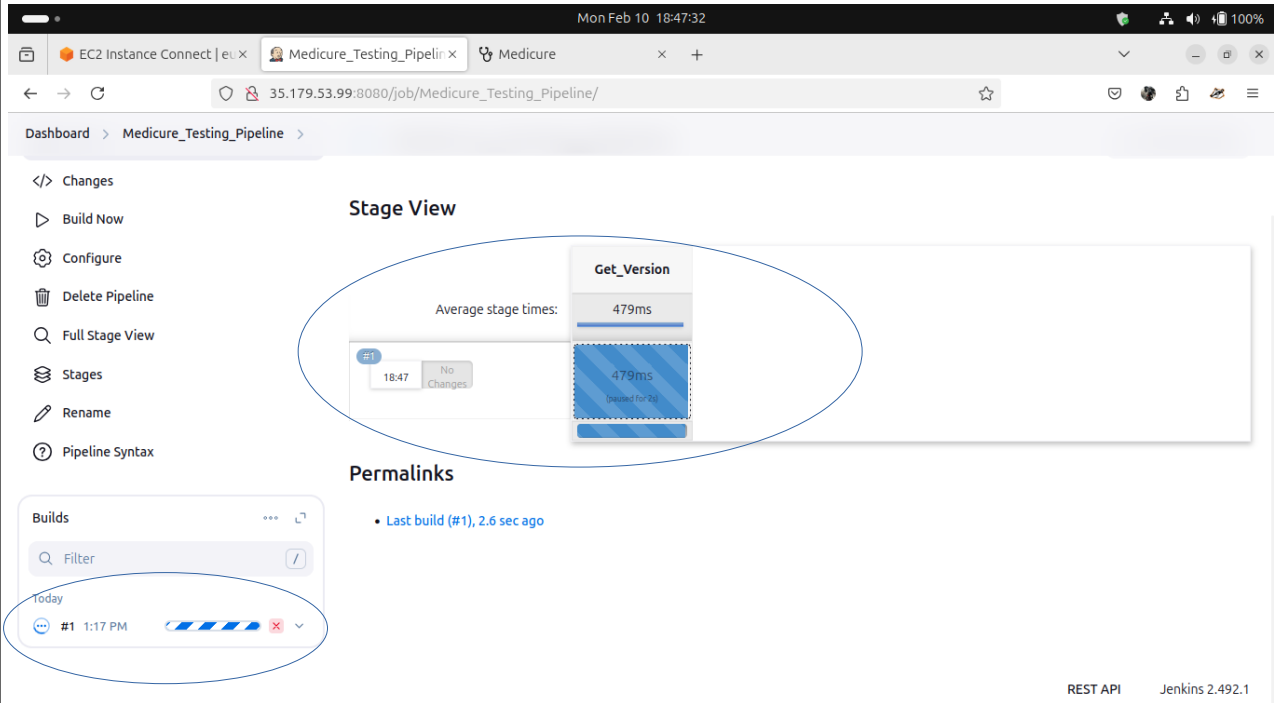
Step – 4.2 :- Running the Testing Pipeline,



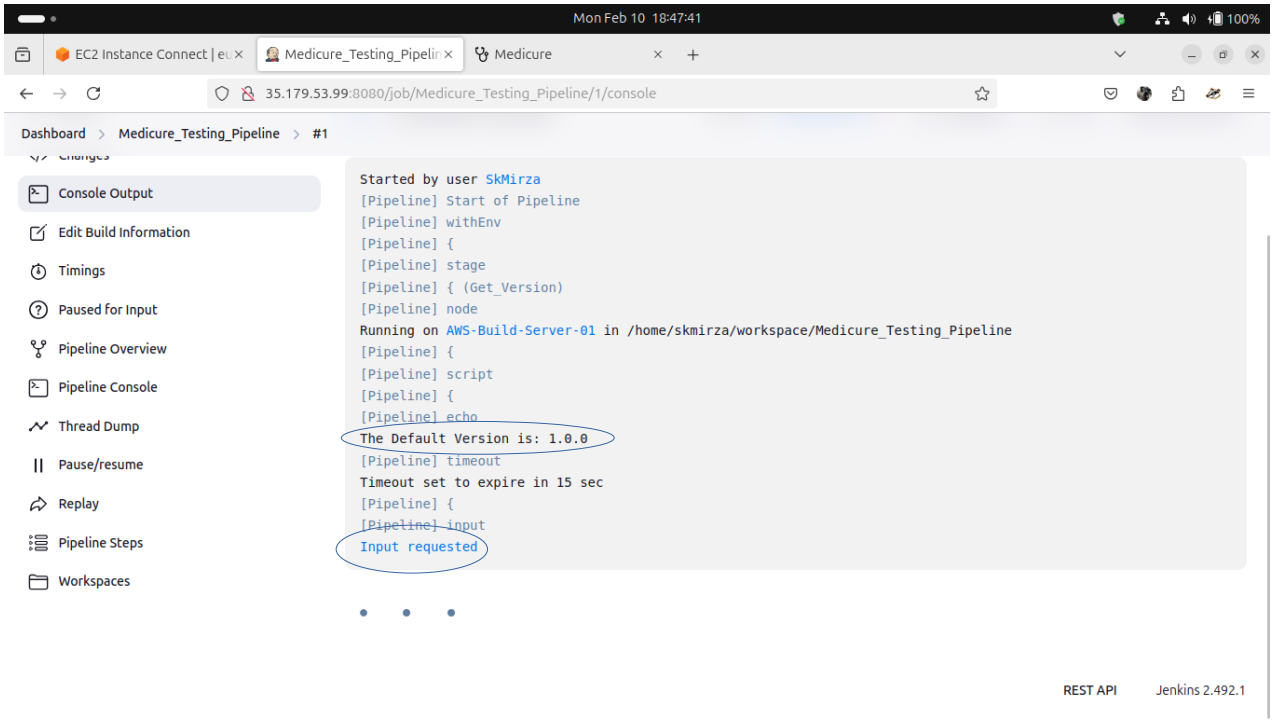
- Pipelines Created, Now go to Testing Pipeline,



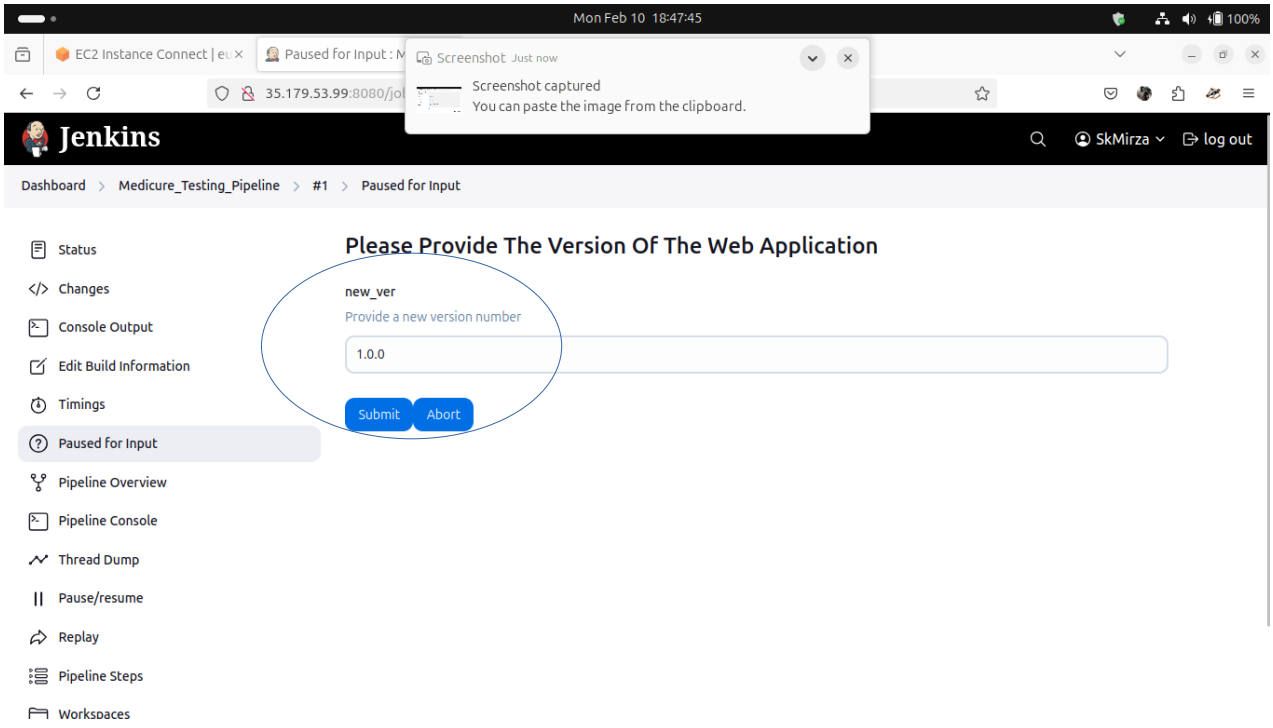
- Now Running The Testing Pipeline, Make sure to run after creating Production Pipeline,
- Because After Testing Pipeline successful it will trigger Production Pipeline,



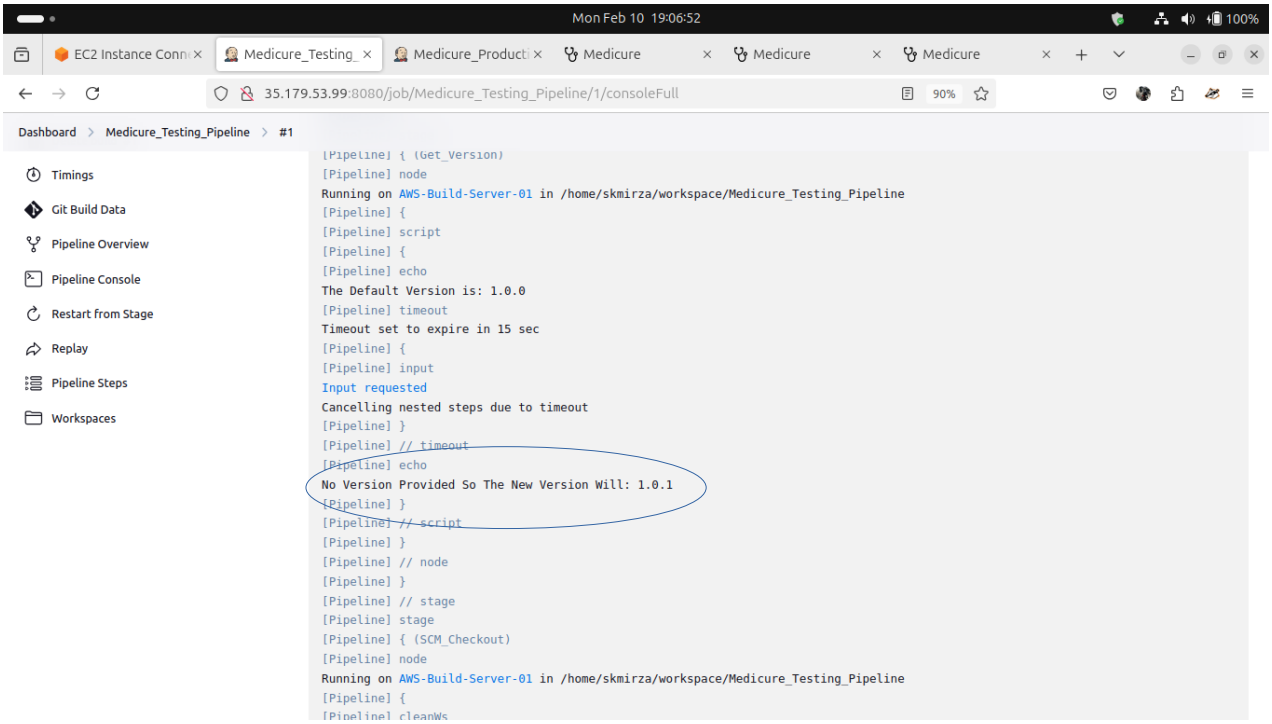
- Build Started, Click on #1 for output console,



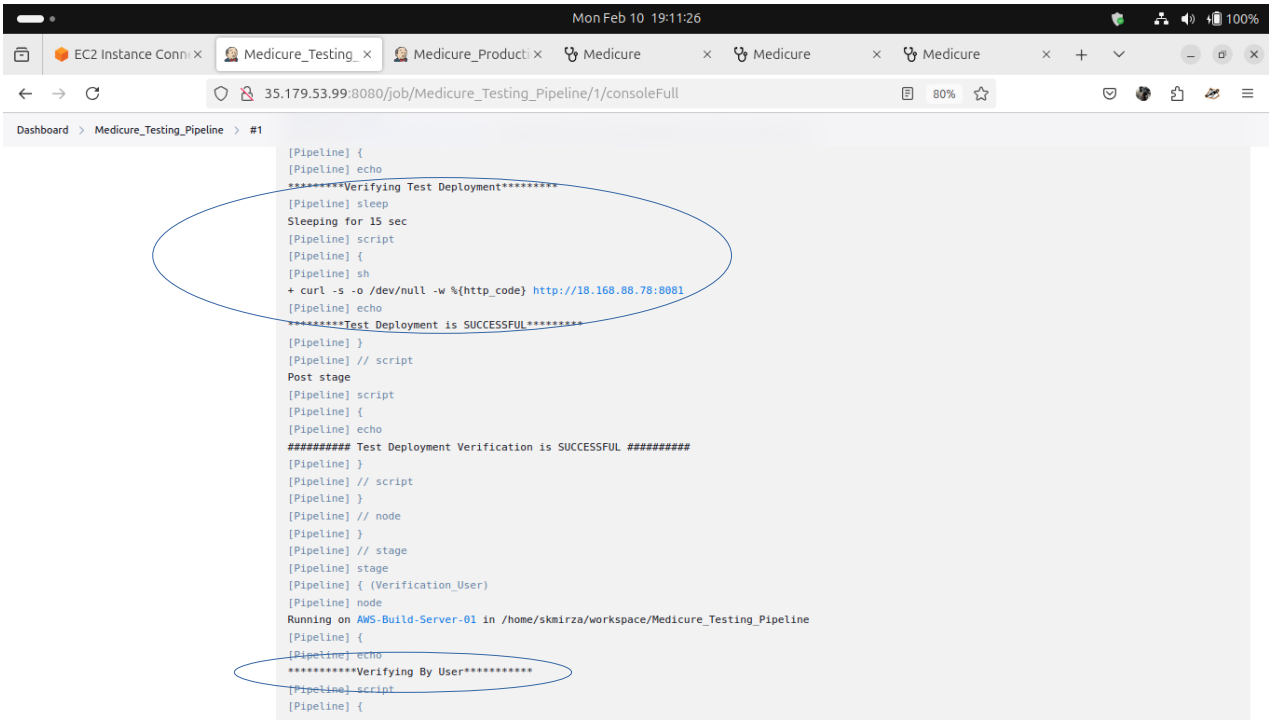
- Here the Default version is: 1.0.0 as set in the environment variable,
- It will Ask For User Input to provide Version of the web application, for 15 Sec timeout



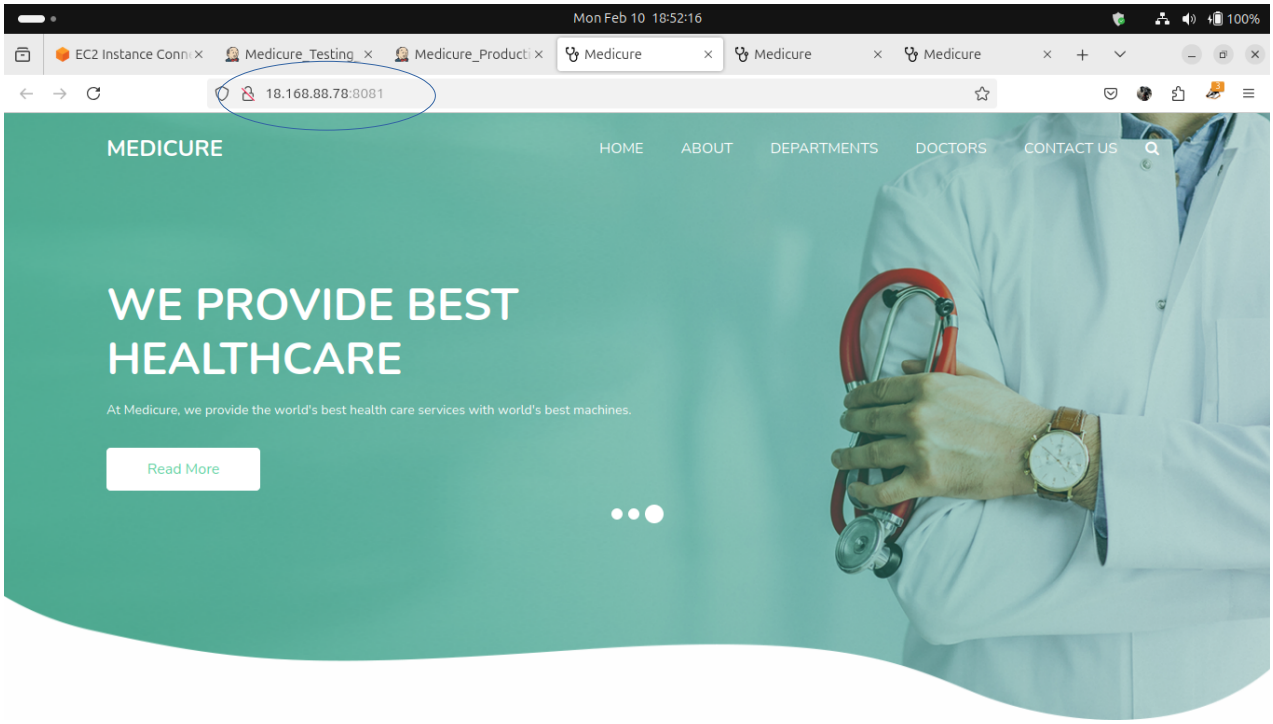
- If user provided then it will consider the version of the Web application, if timeout



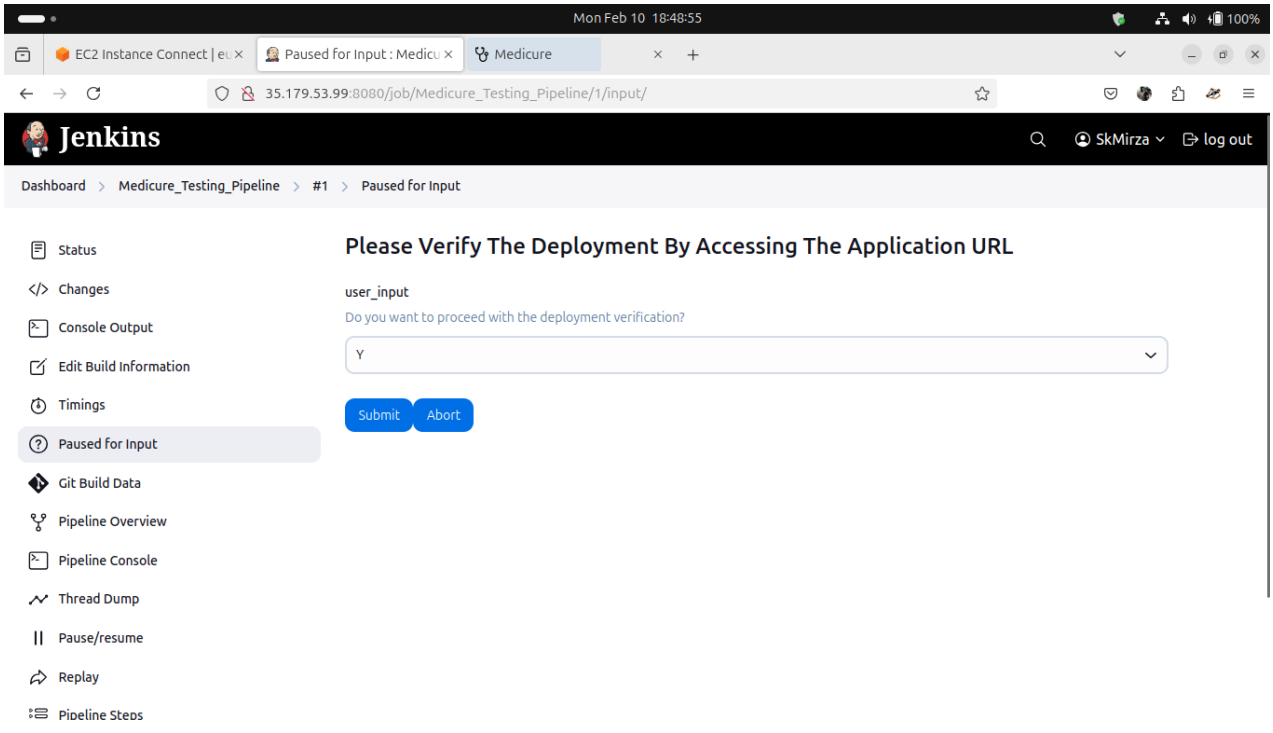
- It will Increment the version 1.0.0 to 1.0.1,



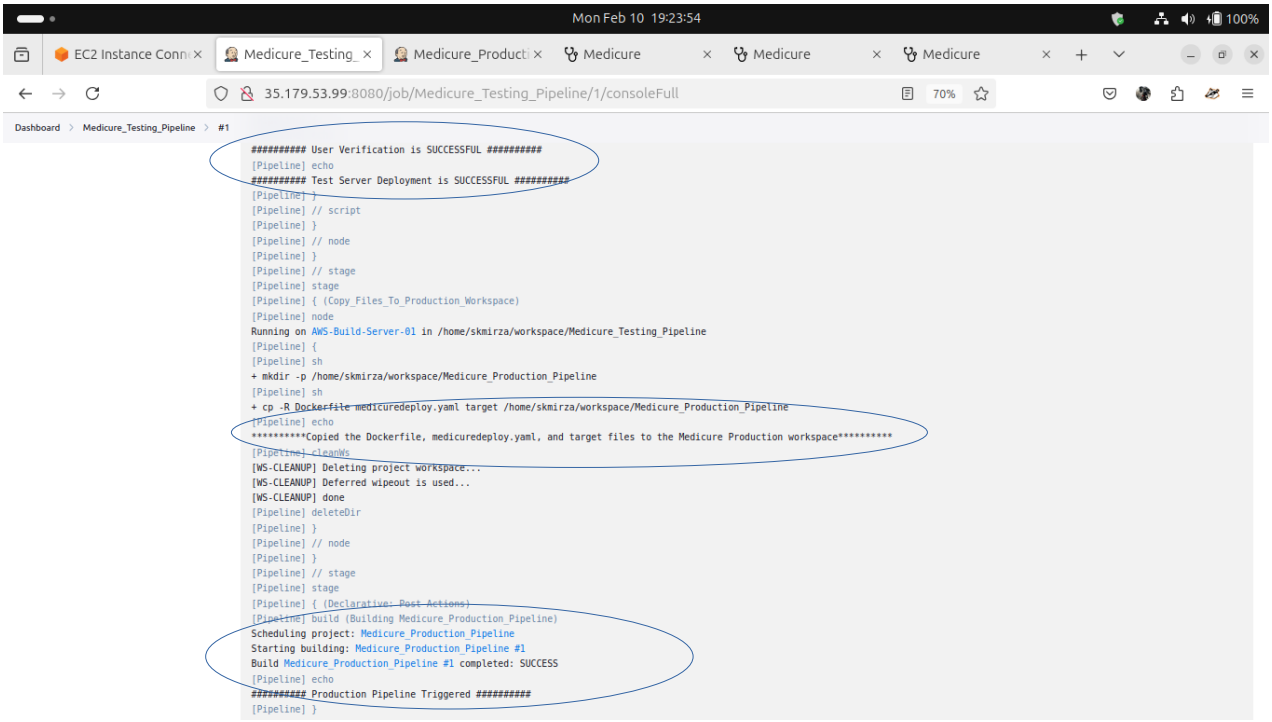
- It will Automatically Test the Deployment by response of the web site,



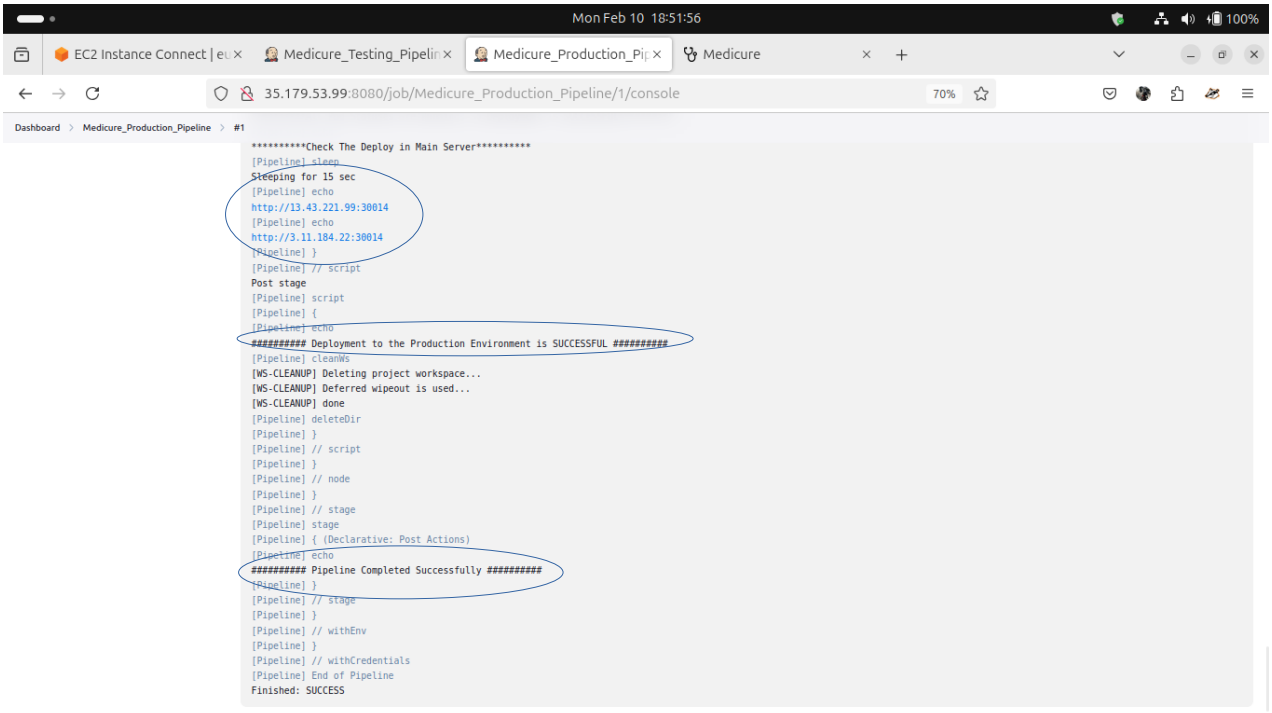
- And wait for the user to verify the Deployment to test server to approve the testing, for 2 min timeout,



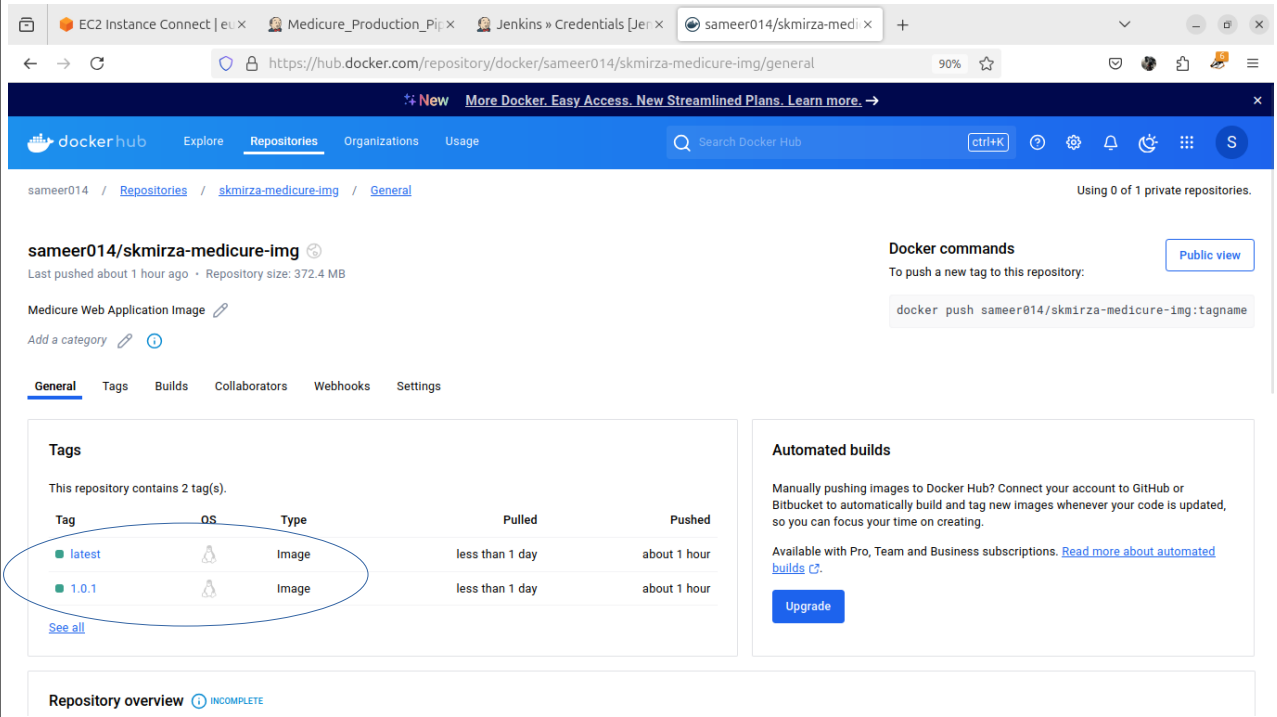
- If user provide ‘Y’ it will continuous the pipeline,
- If user provide ‘N’ it will abort the pipeline,
- if timeout it will consider as ‘Y’ because of Previous testing success,



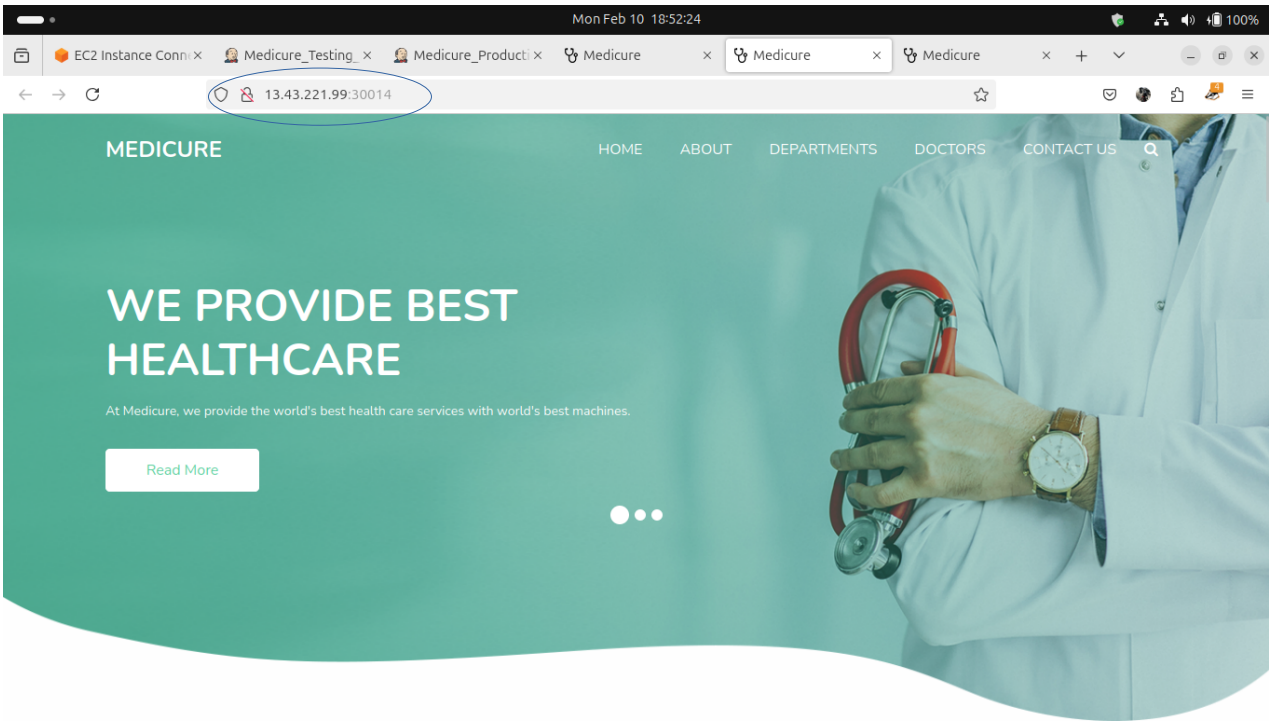
- After Completion of testing pipeline, it will trigger the production pipeline,



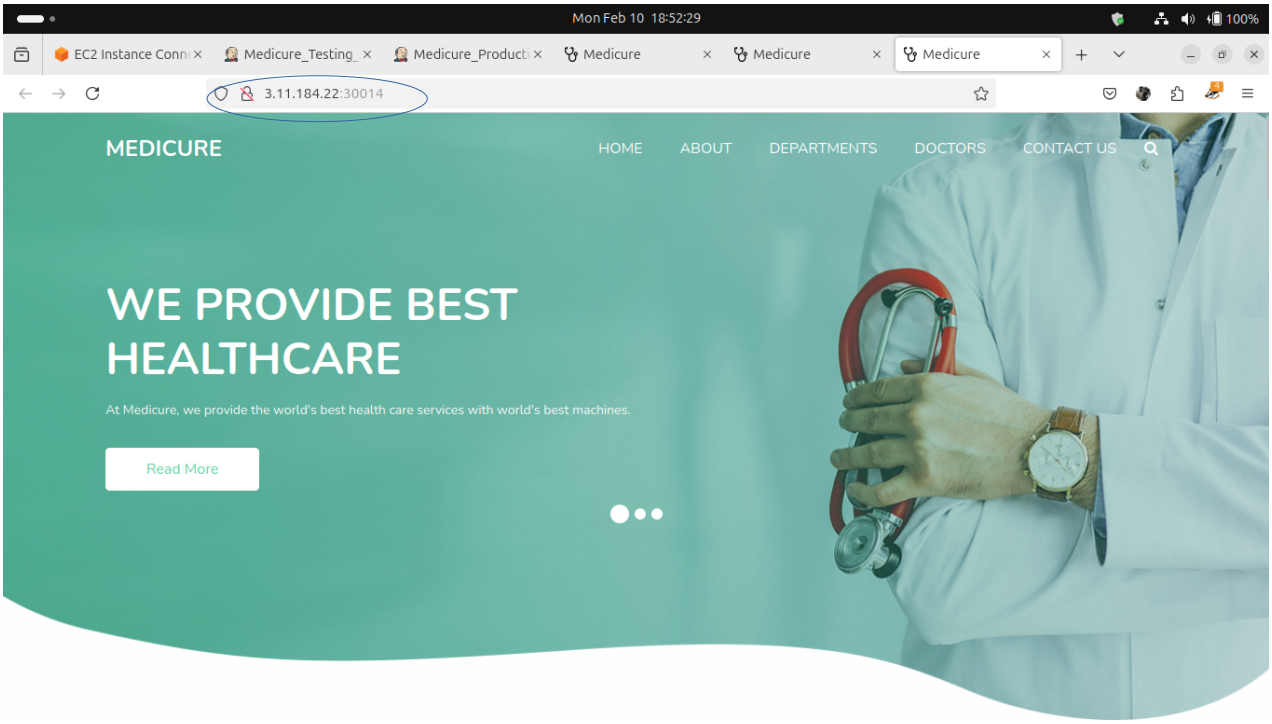
- Production pipeline completed successful, display the urls of the production servers,
- Image will push to DockerHub Repository,



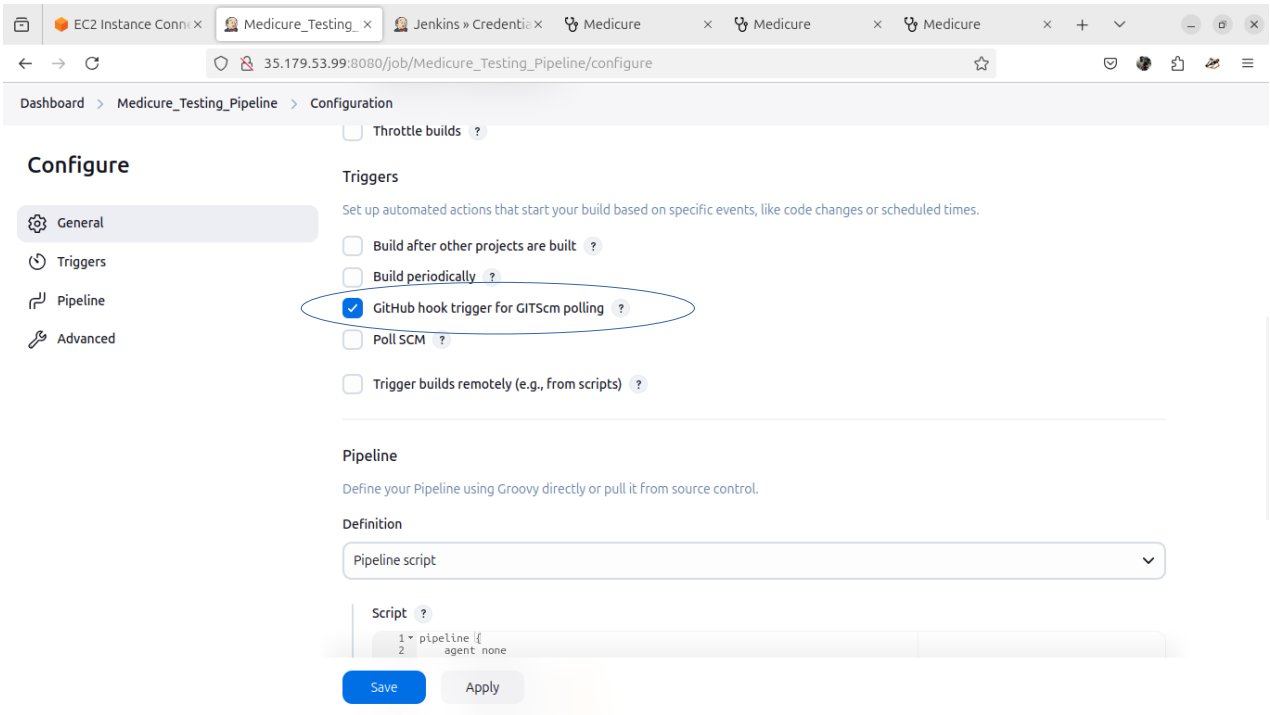
- Production Server 01



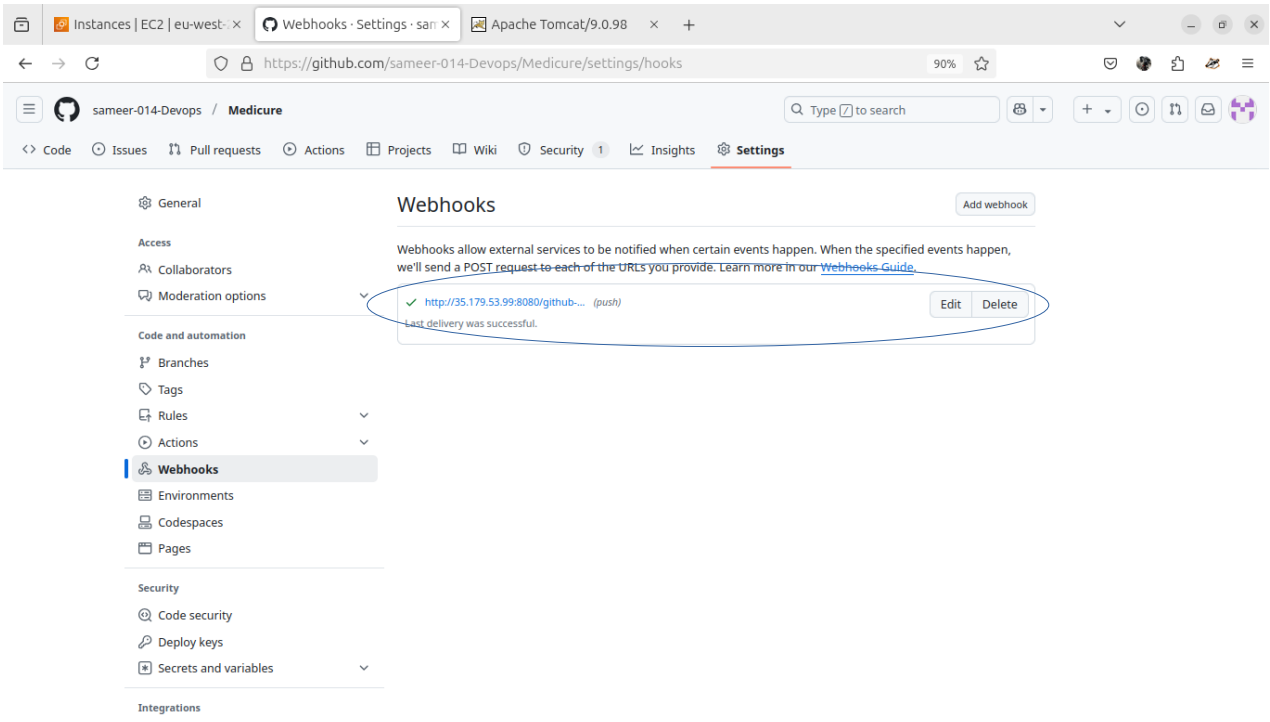
- Production Server 02



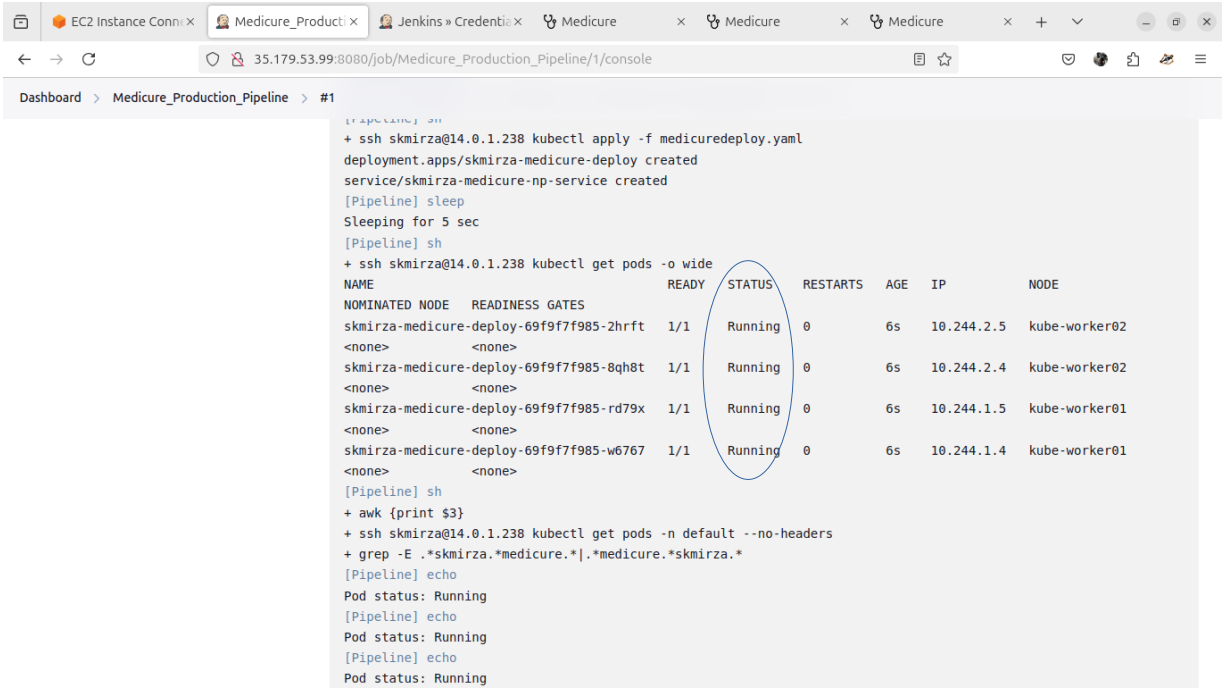
- Additionally you can add the loadbalance, and map DNS to the production servers for one url,
- For Continuous Deploying, Your Can Create a webhook in github repository,



- Enable Github web hook trigger in Testing Pipeline then,



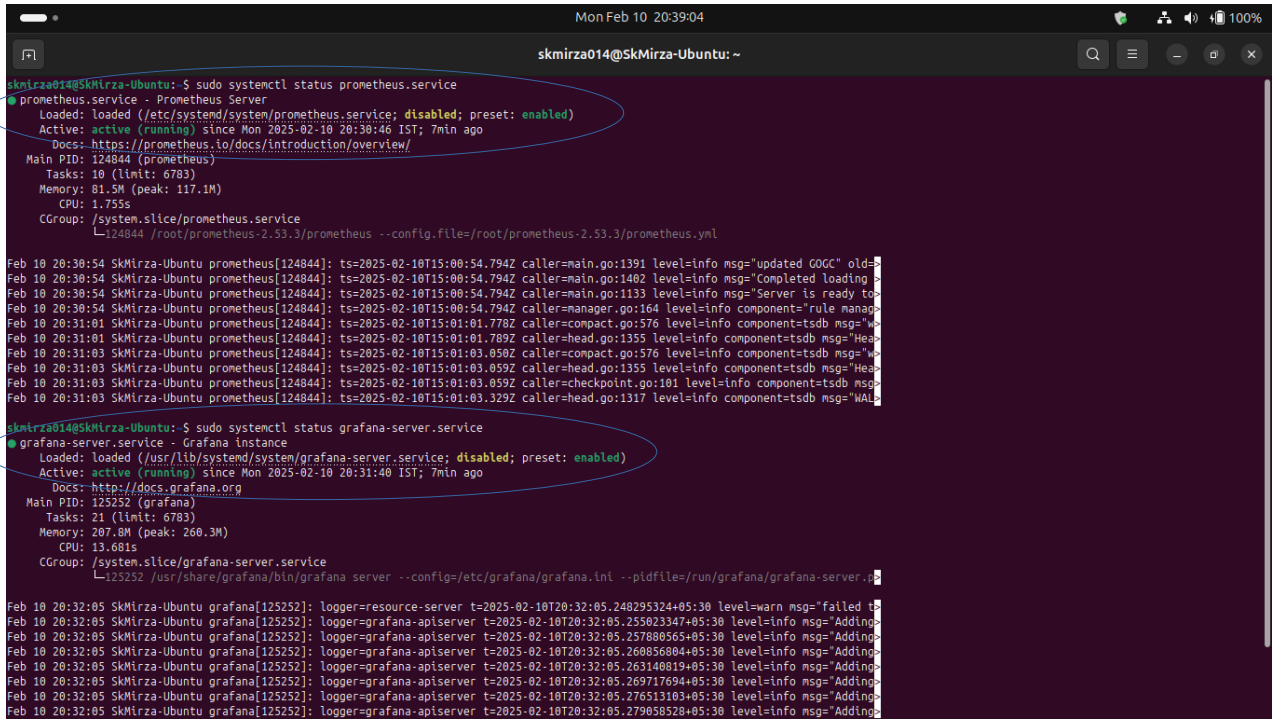
- If any Change Done in the GitHub repository it will auto trigger the Testing then Production Pipeline,
- While Deploying to Production Server if any pods failed to deployment it will rollback to previous version,



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

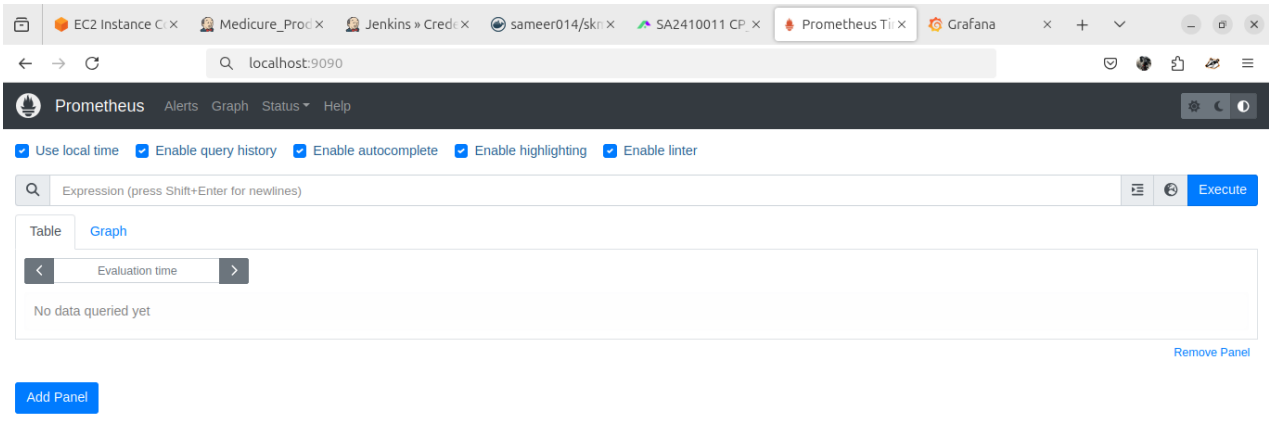
Step – 5.0 :- For Monitoring Install Prometheus and Grafana in Your local host,

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

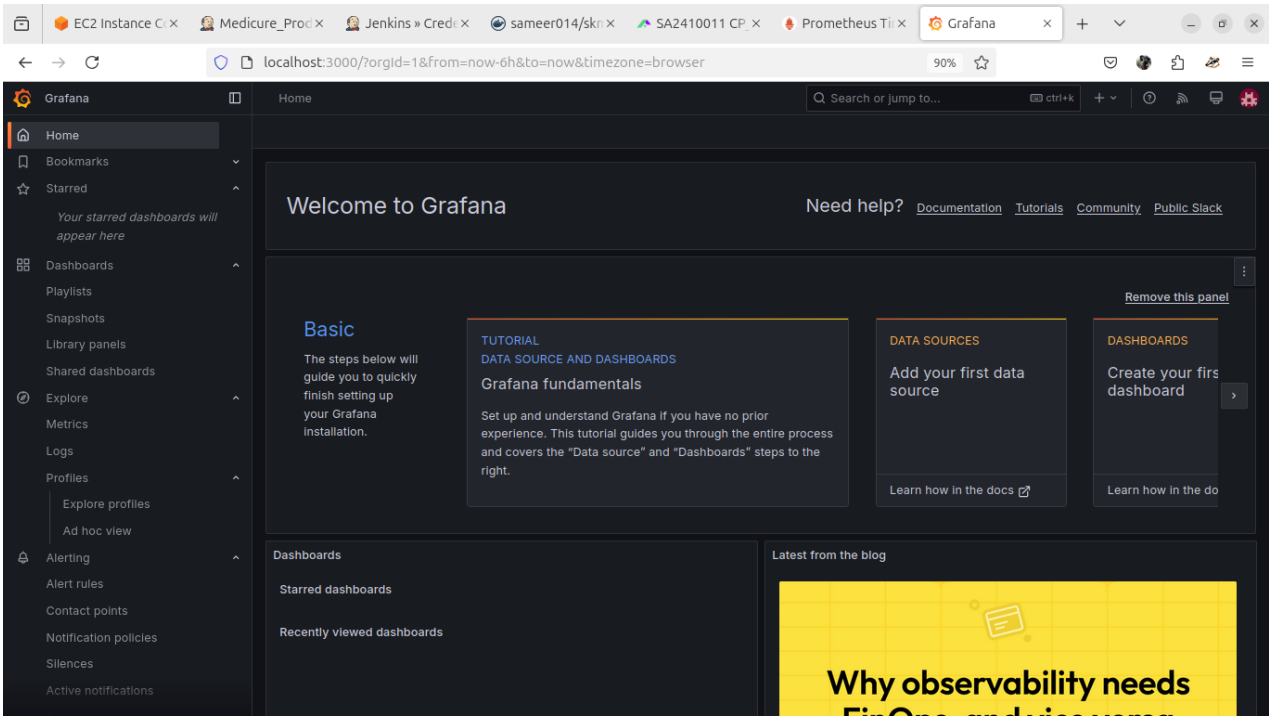


- Prometheus and Grafana Services Running successful, You can verify in browser,

- Prometheus Web Service,

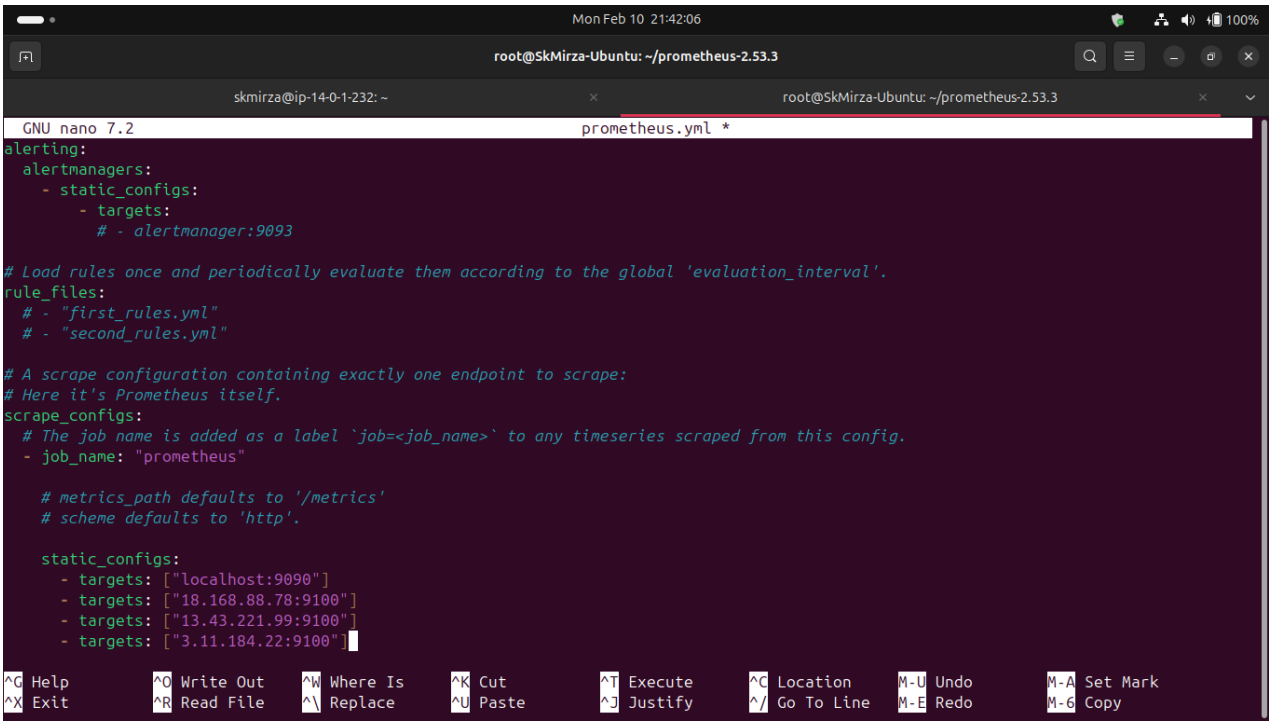


- Grafana Web Service,

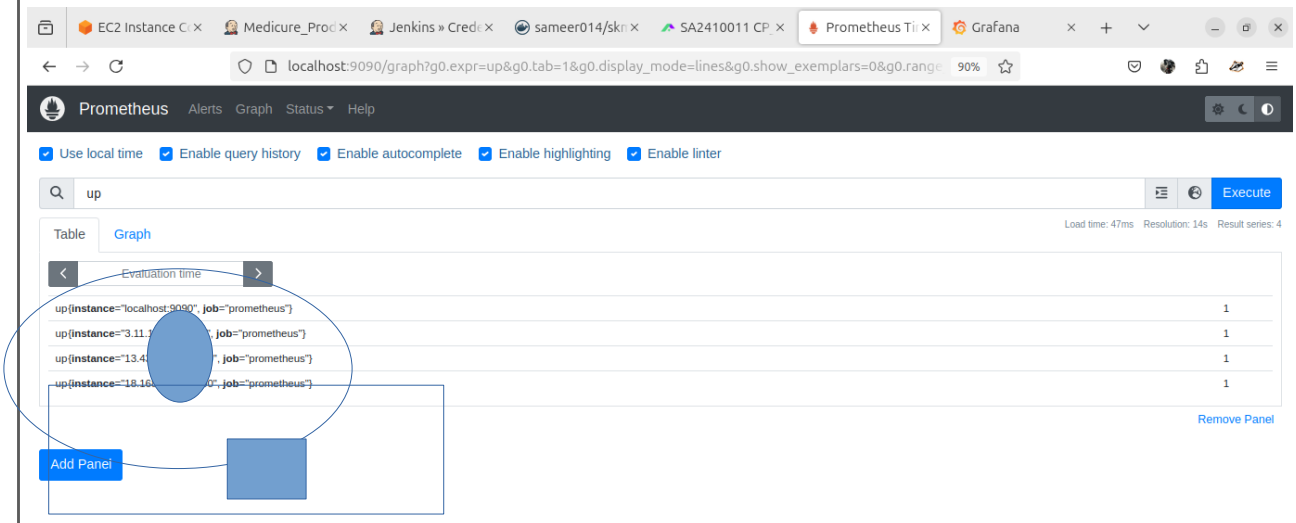


- To get data install node_exporter on the test servers and production servers,
- Check the status of node_exporter in the test servers and production servers,

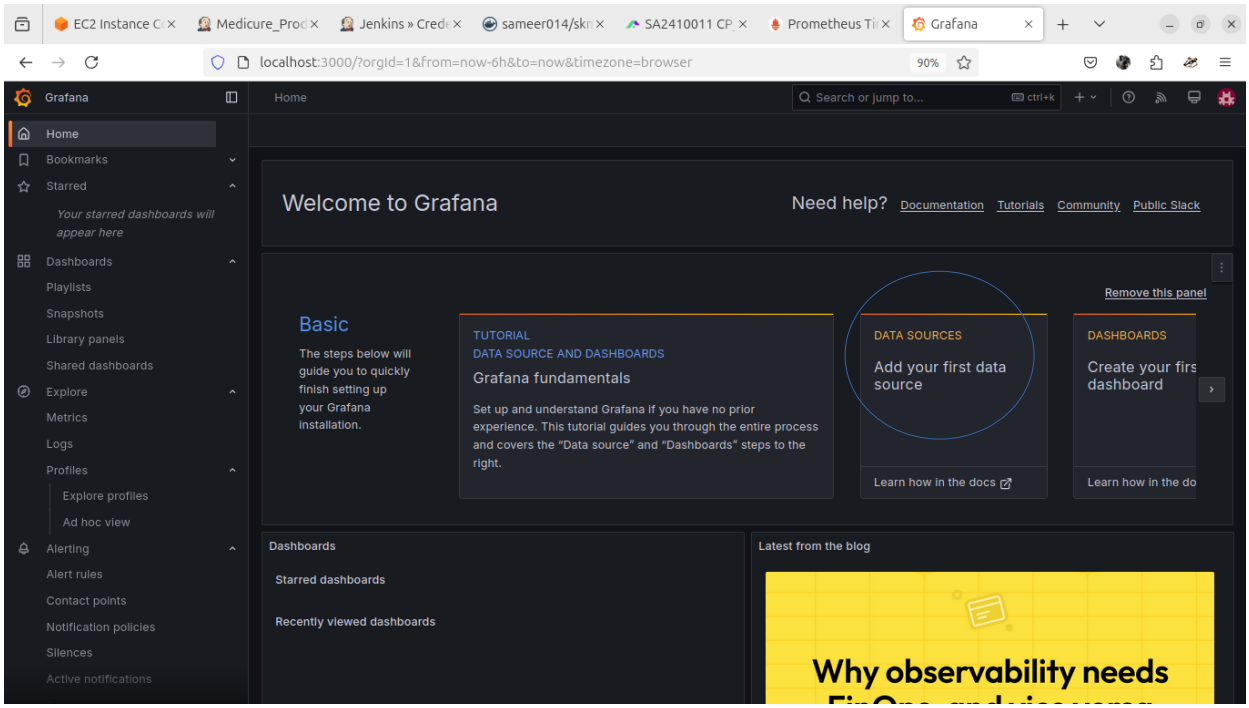
Step – 5.1 :- Add all targets in prometheus.yaml file,



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

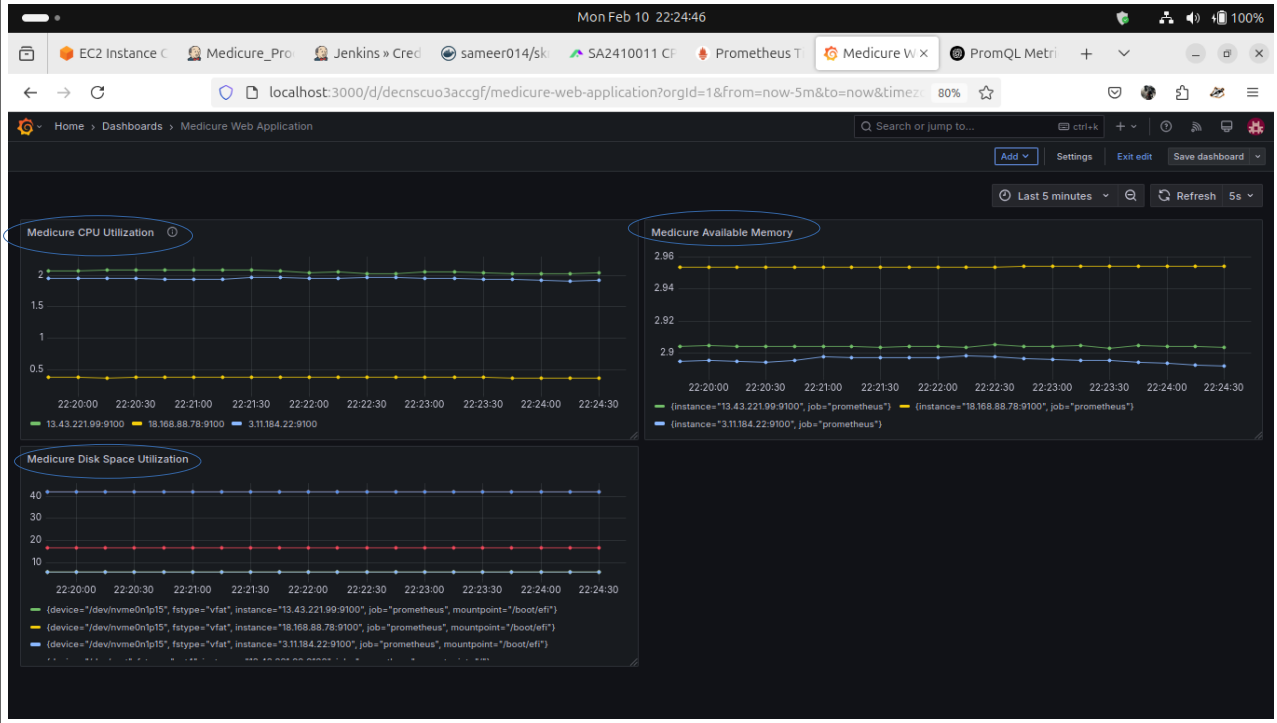


Step – 5.2 :- Add this data to grafana,



- Add the data, and create a dashboard,

- **For CPU utilization**
  - 100 - (avg by(instance) (rate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
- **Disk Space Utilization**
  - 100 - ((node_filesystem_avail_bytes{fstype!~"tmpfs"} * 100) / node_filesystem_size_bytes{fstype!~"tmpfs"})
- **Total Available Memory**
  - (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.