CLOUD MONITORING USING TERRAFORM

CloudWatch, SNS, Prometheus, NodeExporter, Grafana

Mohammed Amir

PROJECT OVERVIEW

Created an automated cloud observability project using **Terraform**, where a secure AWS infrastructure was provisioned with a VPC, public subnet, route table, internet gateway, security groups, and an EC2 instance. SSH access was restricted to a specific CIDR range for enhanced security.

The project showcases two parallel monitoring approaches:

- **AWS CloudWatch Monitoring & Alerting** – A CloudWatch Dashboard was configured to track EC2 performance metrics such as CPU utilization. Additionally, a CloudWatch Alarm was integrated with Amazon SNS to send email notifications whenever CPU utilization reached or exceeded **80%**, ensuring proactive incident response.
- **Prometheus & Grafana Monitoring Stack** – On the EC2 instance, Prometheus, Node Exporter, and Grafana were automatically installed via user data. Prometheus collected system metrics, Node Exporter exposed host-level metrics, and Grafana provided interactive dashboards for real-time visualization and analysis.

This dual setup highlights both AWS-native monitoring and open-source observability tools, delivering a comprehensive and resilient monitoring solution.

KEY SKILLS

- **AWS (VPC, EC2, Security Groups, CloudWatch, SNS):** Designed and deployed secure cloud infrastructure with monitoring dashboards and automated alerting.
- **Terraform (IaC):** Automated provisioning of infrastructure components such as networking, EC2 instances, and monitoring services using Infrastructure-as-Code.
- **Prometheus & Node Exporter:** Implemented open-source monitoring stack to collect and expose system-level metrics.
- **Grafana:** Built interactive dashboards for real-time visualization of system performance and resource usage.
- **Monitoring & Alerting:** Integrated CloudWatch alarms with SNS for proactive incident response.

1 INSTALL TERRAFORM

Terraform was installed on WSL (Linux) to enable infrastructure provisioning. For installation, refer to theofficial documentation page

```
amoomirr@KWid: $ sudo apt-get update && sudo apt-get upgrade -y && \
sudo apt-get install -y gnupg software-properties-common curl && \
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg && \
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /
etc/apt/sources.list.d/hashicorp.list && \
sudo apt-get update && sudo apt-get install terraform -y && \
```

Figure 1: Terraform Installation on WSL

To check if Terraform is installed and verify its version

terraform - -version

Terraform was initialized, plan was checked, and configuration applied.

- **terraform init**: Initializes the Terraform working directory and downloads required provider plugins.

- **terraform plan**: Generates an execution plan showing what actions Terraform will perform.

- **terraform apply**: Applies the changes required to reach the desired state of the configuration.

2 INFRASTRUCTURE SETUP

In this step, the fundamental AWS networking components were provisioned using **Terraform**. The goal was to create a secure and scalable network environment where the monitoring stack could be deployed.

2.1 VPC Creation

A **Virtual Private Cloud (VPC)** named `Monitoring_VPC` was created with the CIDR range `10.0.0.0/16`. This provides an isolated virtual network to host all monitoring resources securely.

2.2 Public Subnet

Inside the VPC, a **public subnet** with CIDR range `10.0.1.0/24` was defined. The subnet is mapped to automatically assign **public IPs** to EC2 instances.

2.3 Internet Connectivity

An **Internet Gateway (IGW)** was attached to the VPC, enabling external connectivity. A **Route Table** was created with a default route (`0.0.0.0/0`) pointing to the IGW. The route table was then associated with the public subnet, ensuring resources inside the subnet could access the internet.

2.4 Security Group (Firewall Rules)

A **Security Group (SG)** named `Monitoring_SG` was created to control inbound and outbound traffic:

- **ICMP (Ping):** Allowed from anywhere for basic connectivity tests.
- **SSH (22):** Restricted to the user's own IP (`var.My_Ip`) for secure remote access.
- **HTTP (80):** Opened to the world to allow web access.
- **Prometheus (9090) & Grafana (3000):** Allowed from anywhere to enable external monitoring and visualization access.
- **Egress:** All outbound traffic allowed for updates and service communication.

2.5 Security Considerations

By restricting **SSH** access only to a specific CIDR (`var.My_Ip`), unauthorized access is prevented. Specific monitoring ports (9090 & 3000) were intentionally opened to support **Prometheus** and **Grafana dashboards**. Using Terraform ensures that all rules are **reproducible, version-controlled, and auditable**.

```
# ------------------ VPC Setup ------------------
resource "aws_vpc" "My_Vpc" {
  cidr_block = "10.0.0.0/16" # Defines VPC CIDR range
  tags = {
    Name = "Monitoring_VPC"
  }
}

# Public Subnet inside VPC
resource "aws_subnet" "Public_Subnet" {
  vpc_id                  = aws_vpc.My_Vpc.id # Links subnet to VPC
  cidr_block              = "10.0.1.0/24"
  availability_zone       = var.aws_az
  map_public_ip_on_launch = true # Enables auto public IP assignment
  tags = {
    Name = "Monitoring-Public-subnet"
  }
}

# Internet Gateway for Internet Access
resource "aws_internet_gateway" "My_IGW" {
  vpc_id = aws_vpc.My_Vpc.id
```

```
  tags = {
    Name = "Monitoring-IGW"
  }
}

# Route Table for Internet Access
resource "aws_route_table" "Public_RT" {
  vpc_id = aws_vpc.My_Vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.My_IGW.id
  }
  tags = {
    Name = "Monitoring-Public-RT"
  }
}

# Associate Route Table to Public Subnet
resource "aws_route_table_association" "PublicRT_Association" {
  subnet_id      = aws_subnet.Public_Subnet.id
  route_table_id = aws_route_table.Public_RT.id
}

# ------------------ Security Group ------------------
resource "aws_security_group" "Monitoring_Sg" {
  name        = "Monitoring"
  description = "Allow ICMP, SSH, HTTP, custom ports"
  vpc_id      = aws_vpc.My_Vpc.id

  ingress {
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [var.My_Ip] # Restricts SSH to your IP
  }

  ingress {
    from_port   = 9090
    to_port     = 9090
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
```

```
    from_port   = 3000
    to_port     = 3000
    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 = "Monitoring_SG"
  }
}
```

3 INSTANCE SETUP

In this step, an **EC2 instance** was provisioned inside the previously created VPC and Public Subnet. The instance was configured to automatically install and start the monitoring stack using the `user_data` script. This ensured a fully automated setup without requiring manual intervention.

3.1 System Preparation

The EC2 instance was updated and upgraded to include the latest security patches before installing monitoring tools.

3.2 Prometheus Installation

- An open-source monitoring and alerting toolkit that collects and stores time-series metrics from applications and systems.
- Prometheus binaries and configuration files were installed.
- A configuration was created to scrape metrics from Prometheus itself (`localhost:9090`) and from Node Exporter (`localhost:9100`).

3.3 Node Exporter Installation

- A lightweight exporter for Prometheus that exposes system-level metrics from Linux servers.
- Node Exporter was installed to expose system-level metrics such as CPU, memory, and disk usage.

- A dedicated `node_exporter` user was created for enhanced security.
- A systemd service was configured to ensure Node Exporter runs automatically on system startup.

3.4 Grafana Installation

- A visualization and analytics platform that integrates with Prometheus (and other data sources) to create interactive dashboards and alerts.
- Grafana was installed from the official repository.
- Grafana server was enabled and started as a systemd service, ensuring persistence across reboots.

3.5 Security Considerations

- The EC2 instance was launched inside the **Monitoring VPC and Security Group**.
- **SSH access (port 22)** is restricted to the user's own IP range (`var.My_Ip`).
- **Prometheus (9090)** and **Grafana (3000)** ports were intentionally exposed for monitoring dashboards.
- Services such as Node Exporter run under dedicated system users to minimize privilege risks.

```
1
2  # ------------------ EC2 Instance------------------
3  resource "aws_instance" "Aws_Vm" {
4    ami                    = "ami-0861f4e788f5069dd" # Amazon Linux 2 AMI
5    instance_type          = "t2.micro"              # Free tier instance
6    key_name               = var.Key_Name
7    subnet_id              = aws_subnet.Public_Subnet.id
8    vpc_security_group_ids = [aws_security_group.Monitoring_Sg.id]
9    availability_zone      = var.aws_az
10
11   tags = {
12     Name = "Ec2-Monitoring"
13   }
14
15 user_data = <<-EOF
16 #!/bin/bash
17 set -e  # Exit on any error
18
19 # -------------------- Update System --------------------
20 sudo yum update -y
21 sudo yum upgrade -y
22
23 # -------------------- Install Prometheus --------------------
24 cd /opt
25
26 # Download and extract Prometheus
```

```
wget https://github.com/prometheus/prometheus/releases/download/v2.1.0/prometheus-2.1.0.linux-amd64.tar.gz
tar -xf prometheus-2.1.0.linux-amd64.tar.gz

# Move Prometheus binaries to /usr/local/bin
sudo mv prometheus-2.1.0.linux-amd64/prometheus /usr/local/bin/
sudo mv prometheus-2.1.0.linux-amd64/promtool /usr/local/bin/

# Create Prometheus directories
sudo mkdir -p /etc/prometheus /var/lib/prometheus

# Move consoles and libraries
sudo mv prometheus-2.1.0.linux-amd64/consoles /etc/prometheus/
sudo mv prometheus-2.1.0.linux-amd64/console_libraries /etc/prometheus/

# Cleanup
rm -rf prometheus-2.1.0.linux-amd64*
# Create Prometheus configuration
cat <<EOT | sudo tee /etc/prometheus/prometheus.yml
global:
  scrape_interval: 10s
scrape_configs:
  - job_name: 'prometheus_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'node_exporter_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9100']
EOT

# -------------------- Install Node Exporter --------------------
cd /opt
sudo curl -LO https://github.com/prometheus/node_exporter/releases/download/v1.7.0/node_exporter-1.7.0.linux-amd64.tar.gz
sudo tar -xvzf node_exporter-1.7.0.linux-amd64.tar.gz
sudo mv node_exporter-1.7.0.linux-amd64/node_exporter /usr/local/bin/

# Create a dedicated user for Node Exporter
sudo useradd --no-create-home --shell /bin/false node_exporter

# Create systemd service for Node Exporter
sudo tee /etc/systemd/system/node_exporter.service > /dev/null <<EOC
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target
EOC

# Enable and start Node Exporter
sudo systemctl daemon-reload
```

```
sudo systemctl enable node_exporter
sudo systemctl start node_exporter

# -------------------- Install Grafana --------------------
sudo tee /etc/yum.repos.d/grafana.repo > /dev/null <<EOC
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
EOC

sudo yum install grafana -y

# Enable and start Grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server

EOF

}
```

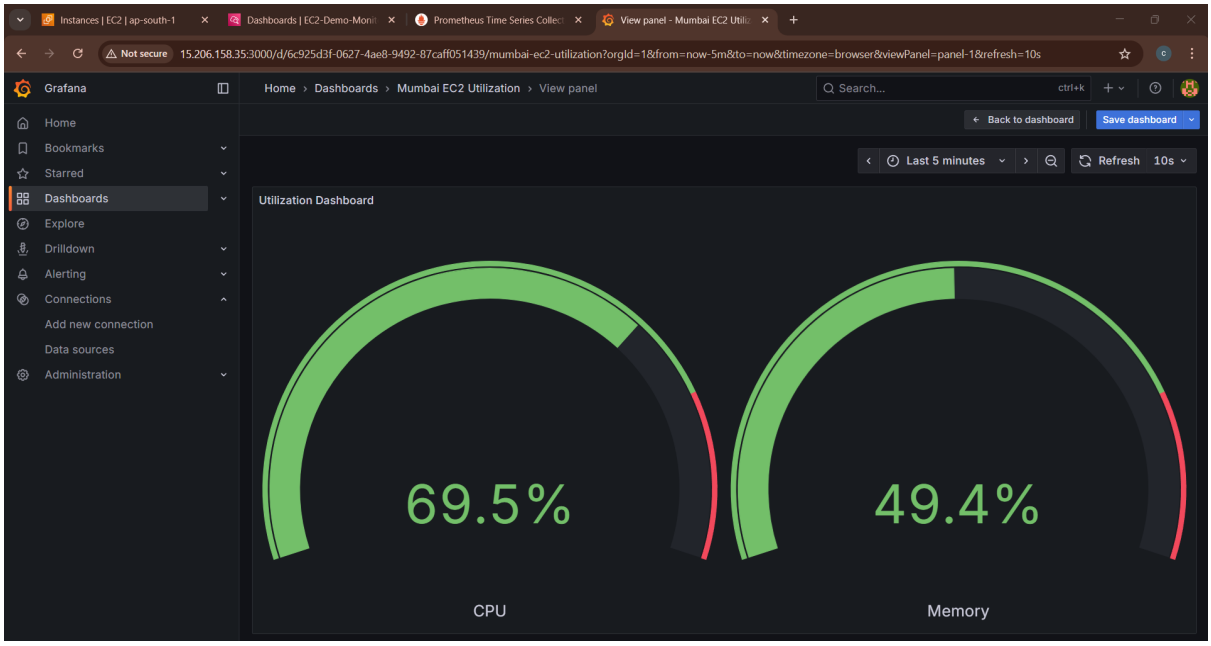

Figure 2: Grafana Dashboard

4 CloudWatch Dashboard and Utilization Alerting with SNS

In this step, **AWS CloudWatch** was used to build a custom monitoring dashboard for the EC2 instance and configure an automated alerting mechanism. The dashboard provides real-time visibility of EC2 performance metrics, while alerts ensure proactive notifications in case of threshold breaches.

4.1 CloudWatch Dashboard

- A dashboard named `EC2-Demo-Monitoring` was created.
- A **title widget** was added to identify the monitored region (Mumbai).
- A **line chart widget** was added to display the average CPU utilization over time.
- A **single value widget** was configured to show the latest CPU utilization percentage at a glance.

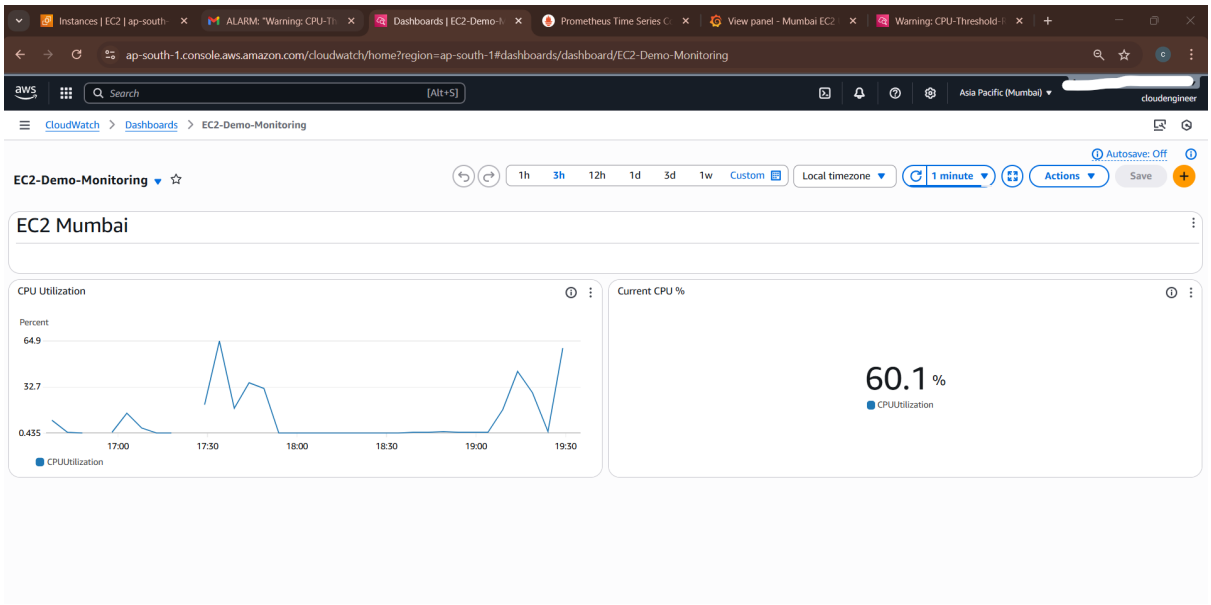

Figure 3: Cloudwatch Dashboard

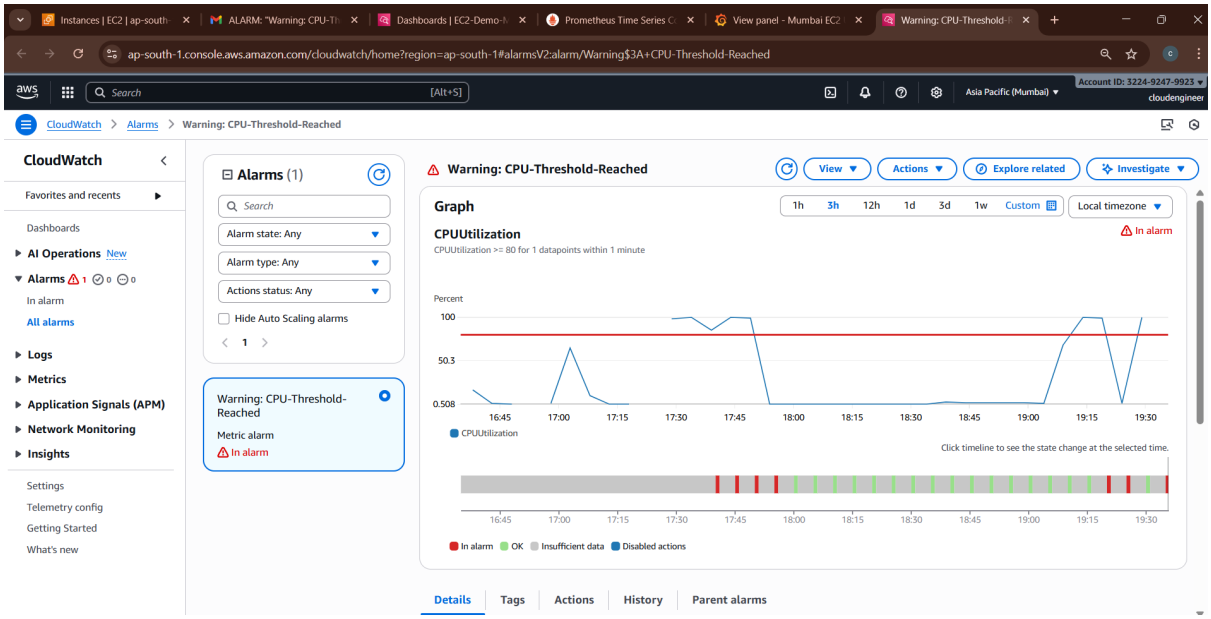

Figure 4: Alert Dashboard

4.2 CloudWatch Alarm and SNS Integration

- A CloudWatch Alarm was created to monitor CPU utilization.
- The alarm threshold was set at **80%**.
- When the threshold was breached, an **SNS Topic** triggered an email notification to subscribed users.

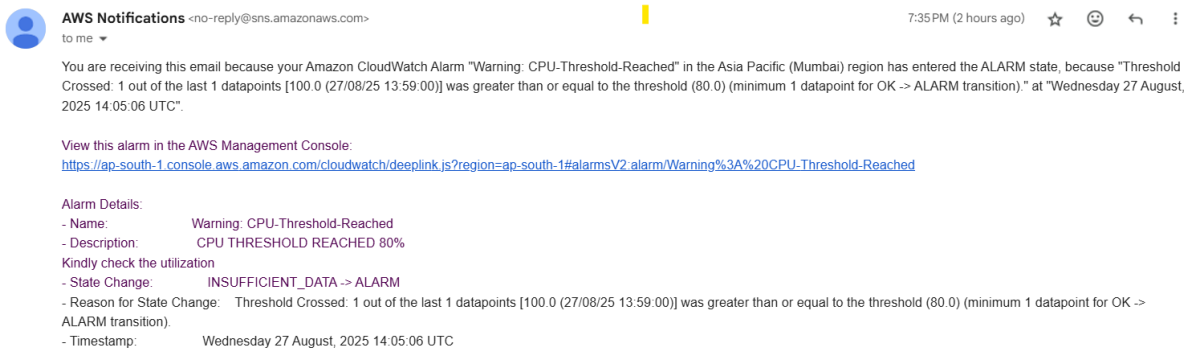

Figure 5: Email Triggered

4.3 Security Considerations

- Alerts were delivered securely via **SNS**.
- The monitoring setup ensures early detection of resource exhaustion and abnormal usage.
- Dashboards provide visibility without exposing sensitive system information.

```
1
2  # ------------------ SNS Topic ------------------
3  resource "aws_sns_topic" "cloudwatch_alarms_topic" {
4    name = "CloudWatch_Alarms_Topic"
5  }
6
7  # ------------------ SNS Subscription ------------------
8  resource "aws_sns_topic_subscription" "email_subscription" {
9    topic_arn = aws_sns_topic.cloudwatch_alarms_topic.arn
10   protocol  = "email"
11   endpoint  = "cloudengineer1282@gmail.com" # Replace with your email
12 }
13
14 # ------------------ CloudWatch Alarm ------------------
15 resource "aws_cloudwatch_metric_alarm" "cpu_high_alarm" {
16   alarm_name          = "High-CPU-Alarm"
17   comparison_operator = "GreaterThanOrEqualToThreshold"
18   evaluation_periods  = 1
19   metric_name         = "CPUUtilization"
20   namespace           = "AWS/EC2"
21   period              = 60
22   statistic           = "Maximum"
23   threshold           = 80
24   alarm_description   = "Alarm triggers if CPU utilization is >= 80% (
       Maximum)."
25   alarm_actions       = [aws_sns_topic.cloudwatch_alarms_topic.arn]
26
27   dimensions = {
28     InstanceId = aws_instance.Aws_Vm.id
29   }
30 }
31
32 # ------------------ CloudWatch Dashboard ------------------
33 resource "aws_cloudwatch_dashboard" "enhanced_dashboard" {
34   dashboard_name = "EC2-Demo-Monitoring"
35
36   # The dashboard_body uses JSON to define widgets
37   dashboard_body = jsonencode({
38     widgets = [
39
40       # Dashboard Title Widget
41       {
42         type       = "text"
43         x          = 0
44         y          = 0
45         width      = 24
46         height     = 2
47         properties = {
48           markdown = "# EC2 Mumbai"
49         }
50       },
51
52       # CPU Utilization Chart
53       {
54         type       = "metric"
55         x          = 0
56         y          = 2
```

```
        width      = 12
        height     = 6
        properties = {
          metrics = [
            ["AWS/EC2", "CPUUtilization", "InstanceId", aws_instance.
               Aws_Vm.id]
          ]
          period = 60
          stat   = "Average"
          title  = "CPU Utilization"
          region = "ap-south-1"
        }
      },

      # Single Value CPU Widget
      {
        type       = "metric"
        x          = 12
        y          = 8
        width      = 12
        height     = 6
        properties = {
          metrics = [
            ["AWS/EC2", "CPUUtilization", "InstanceId", aws_instance.
               Aws_Vm.id]
          ]
          view   = "singleValue"
          stat   = "Average"
          period = 60
          title  = "Current CPU %"
          region = "ap-south-1"
        }
      },
    ]
  }
  )
}
```

5 Restricted SSH Access within VPC

To enhance security, the EC2 instance was configured so that SSH (port 22) access is not open to the public internet. Instead, the security group rules allow SSH connectivity only from other instances within the same VPC.

This ensures that:

- Direct SSH access from external IP addresses is blocked.
- Administrative access can only be performed by first connecting to another instance inside the VPC.
- The attack surface is reduced, as no public SSH exposure exists.

By enforcing this design, the EC2 instance is effectively shielded from unauthorized SSH attempts

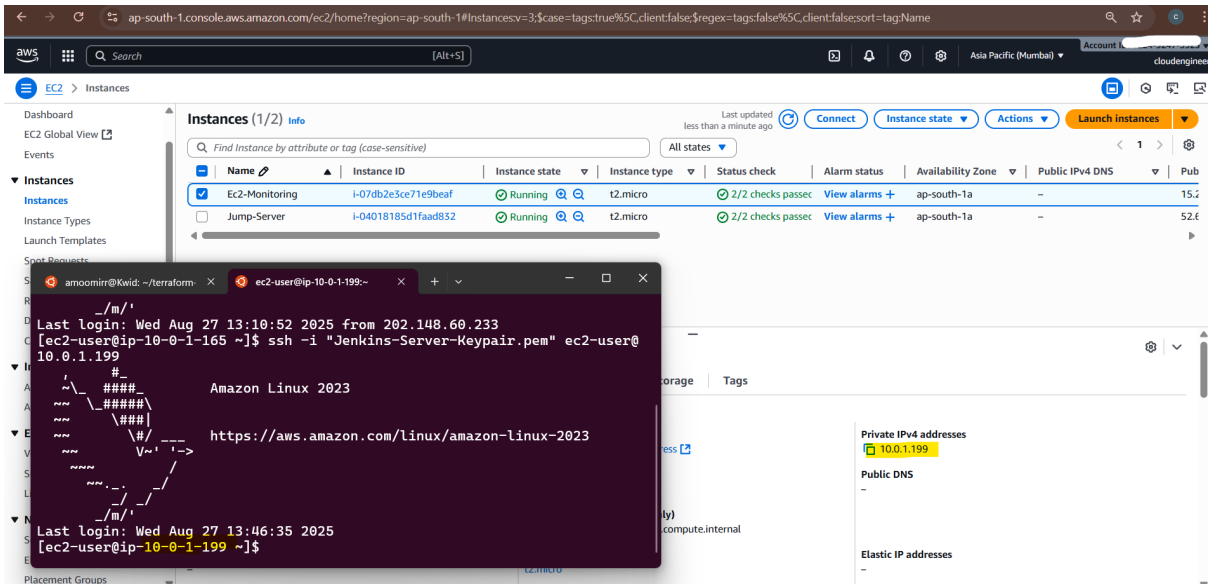

Figure 6: SSH Login

CONCLUSION

This project successfully demonstrated how to build a secure and automated cloud monitoring environment using **Terraform** as the Infrastructure-as-Code (IaC) tool. The solution integrated both **AWS-native monitoring (CloudWatch)** and **open-source observability tools (Prometheus & Grafana)**, showcasing a dual monitoring strategy that ensures resilience, flexibility, and deeper system insights.

By provisioning the complete infrastructure with Terraform — including the VPC, subnet, internet gateway, route tables, security groups, and EC2 instance — the project highlighted the power of automation, reproducibility, and version-controlled infrastructure deployment.

Key achievements:

- **Automated Infrastructure:** Terraform enabled fully reproducible setup of networking and compute resources without manual intervention.
- **Dual Monitoring Setup:** CloudWatch with SNS provided native alerting for CPU thresholds, while Prometheus and Grafana delivered rich visualization and host-level metrics.
- **Security Hardening:** SSH access was restricted to trusted sources and limited within the VPC, reducing the attack surface.
- **Scalability:** The modular design allows easy scaling of monitoring resources or extension to multiple instances.
- **Cost-Efficiency:** Resources could be safely cleaned up using `terraform destroy`, ensuring no unnecessary cloud expenses.

In conclusion, this project demonstrated not only how monitoring can be automated in AWS, but also how combining cloud-native and open-source tools provides a more comprehensive observability solution. It emphasizes the importance of automation, security, and proactive monitoring in modern cloud infrastructure management.