# **CloudWatch Using Terraform**

**What is cloudwatch?**

Amazon CloudWatch is a monitoring and observability service offered by AWS (Amazon Web Services) With Amazon CloudWatch, you can track the resources and application performance, collect and monitor log file details, and enable your resources’ alarms and notifications to be triggered on specific events.

**Key Features of CloudWatch:**

* Metrics Collection
* Logs Monitoring
* Alarms
* Dashboards
* Events and Automated Responses
* Application Insights

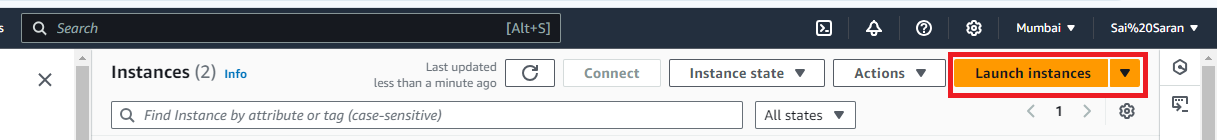
### Use Cases:

* **Monitoring infrastructure** for operational health.
* **Automating responses** to performance changes or failures.
* **Troubleshooting** using logs and traces.
* **Optimizing resource usage** with insights into application performance.

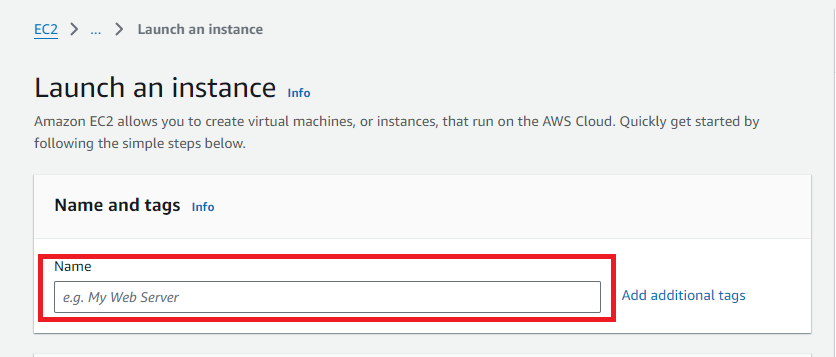
**Steps to create Terraform CloudWatch:**

# Step-1: Create AWS EC2 Instance in AWS Console

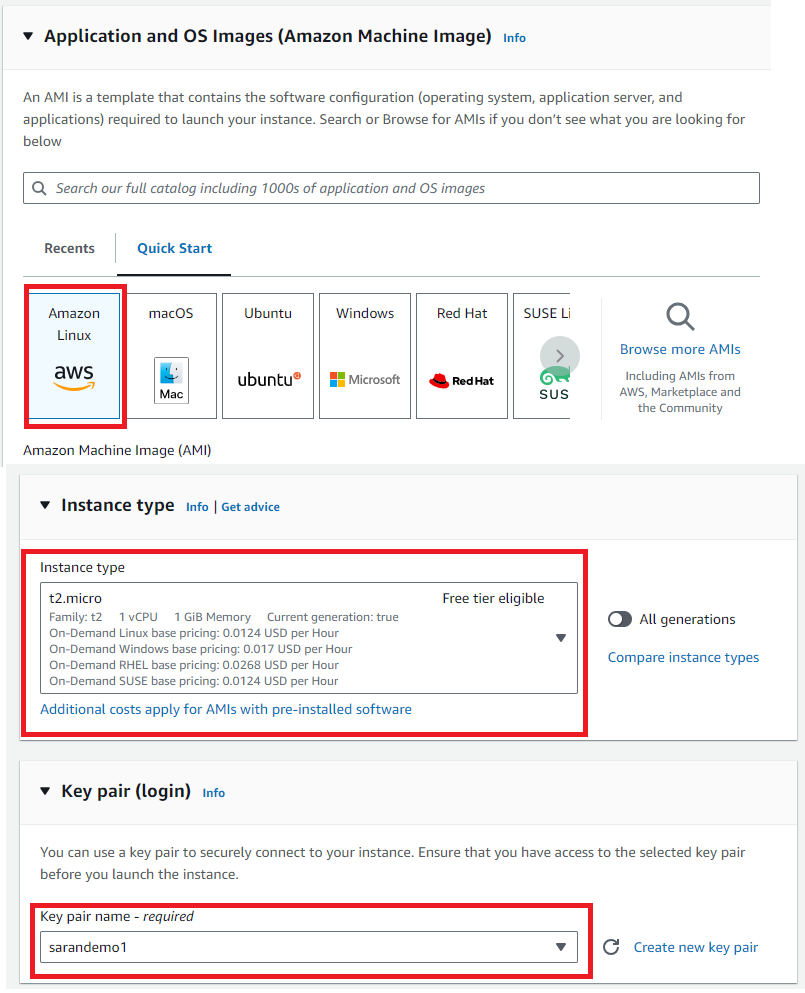
* Go to the AWS console search EC2 Service
* Then Click on launch Instance



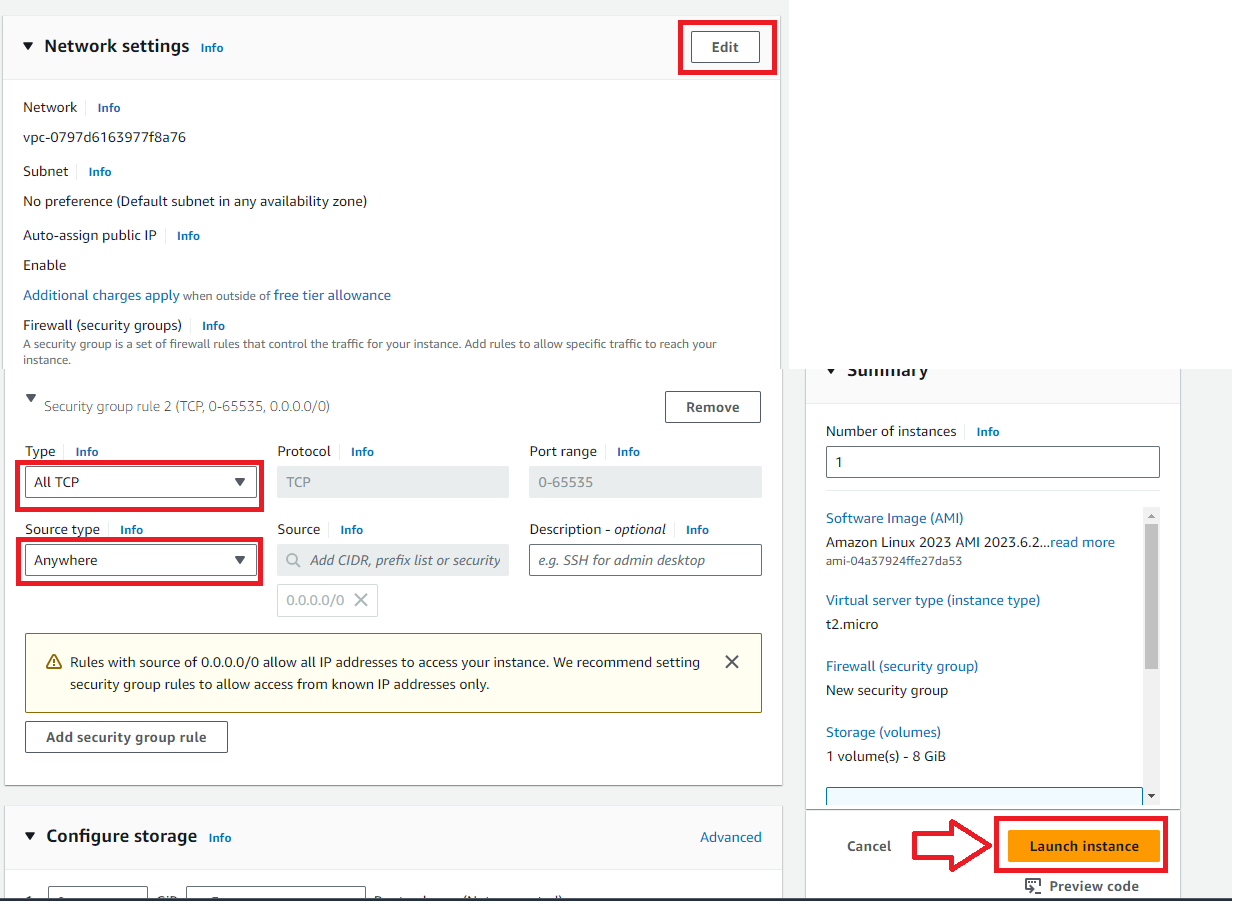
* After click on launch instance provide the name for the Instance



* Then select Amazon Machine Image (AMI)
* After that select Instance type
* Then provide the key pair if existing key pair is not available need to create new key pair by clicking on create new pair

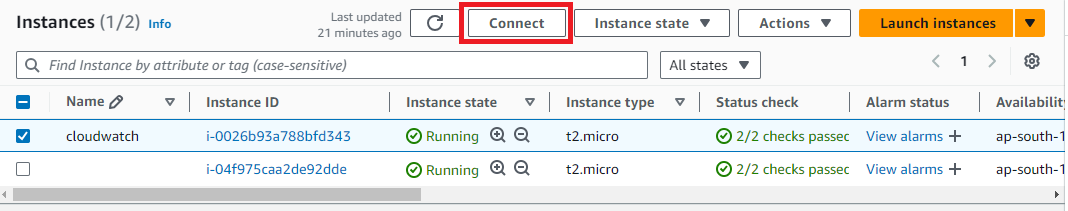


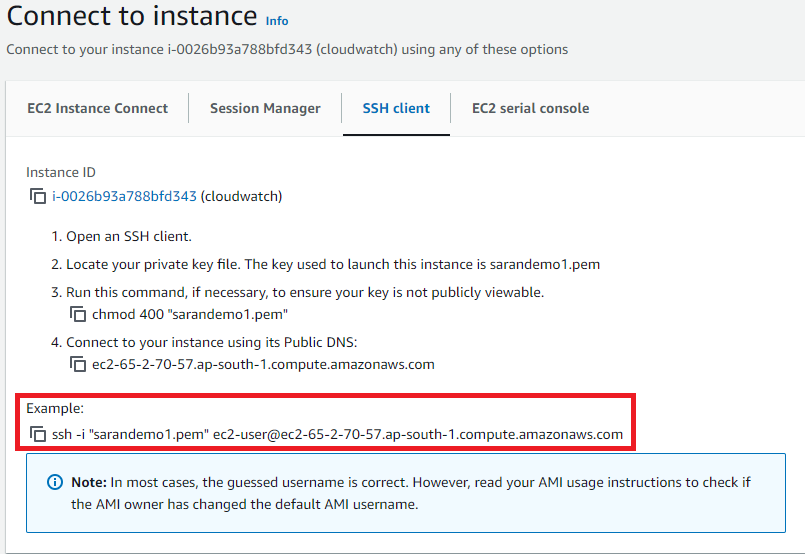
* In Network settings edit network settings and add All TCP
* Then click on Launch Instance



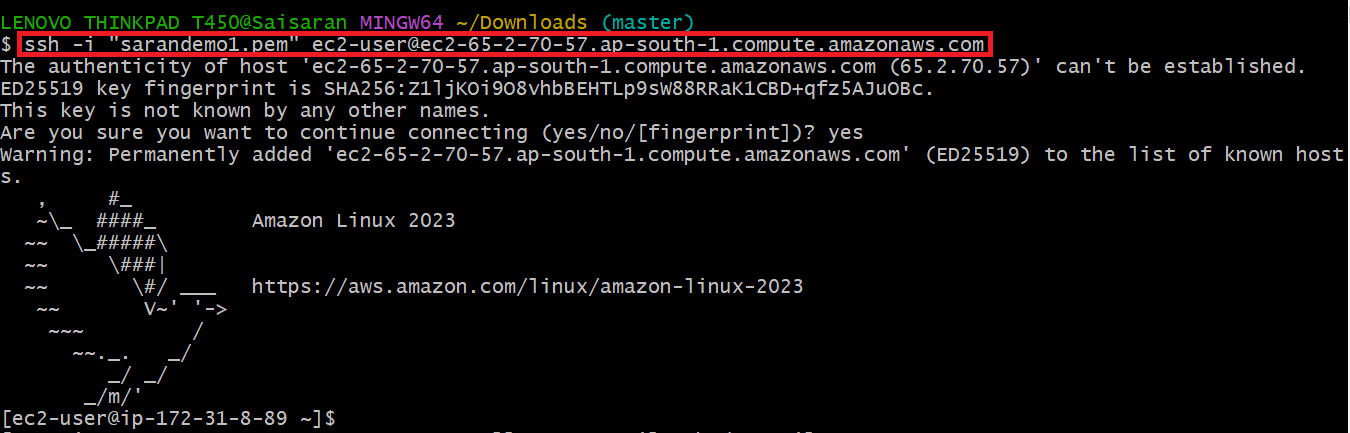
# Step-2: Connect EC2 Instance with Tools

* After creating the Instance need to click on connect on AWS Console
* Then Copy the SSH link





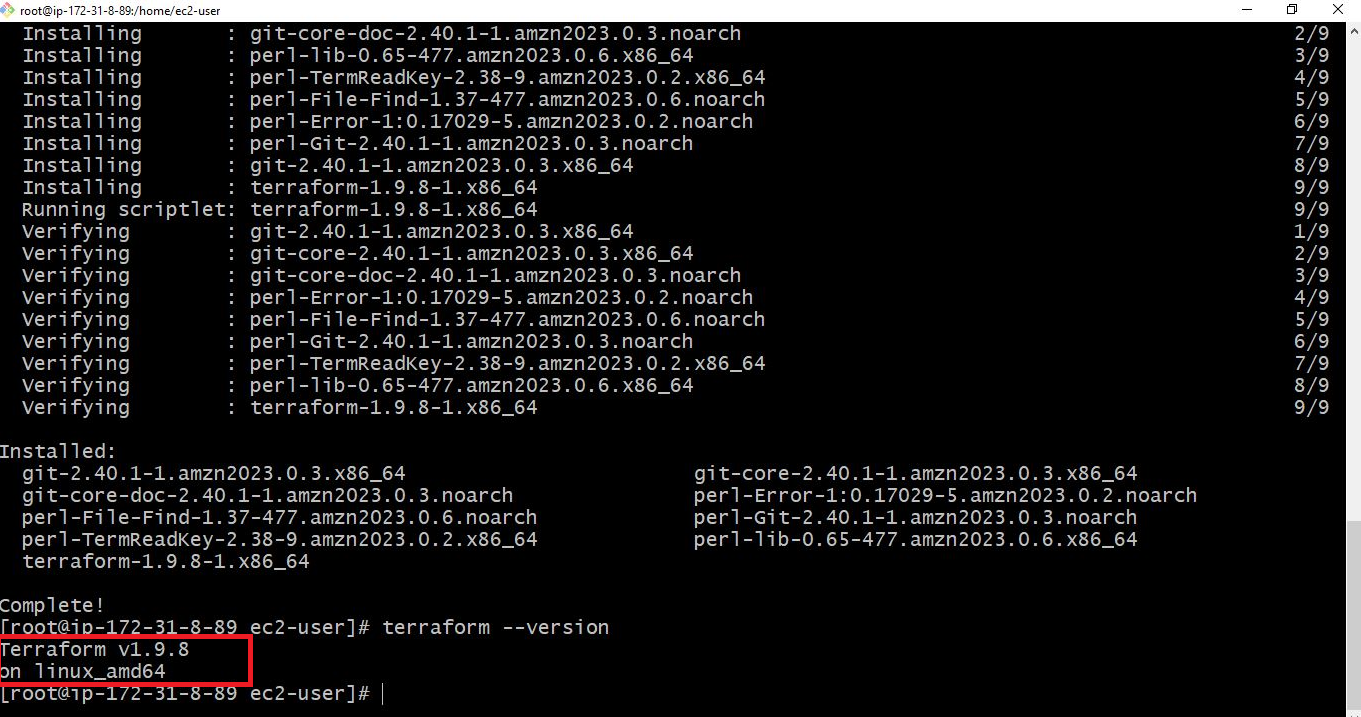
* After copying the SSH link need to connect to GIT,MobaXterm,etc..,
* Here I am connecting with GIT bash for need to got the key pair downloaded path and open GIT bash there and paste the SHH link then click on yes the Linux is connected to tool.



# Step-3:Install Terraform in EC2 Instance

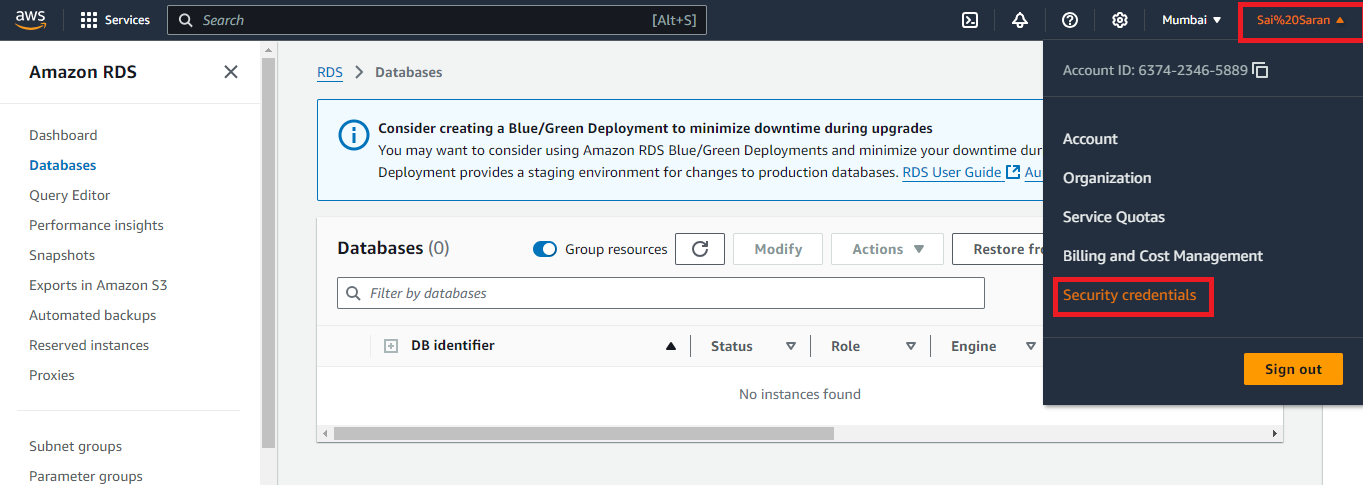
After connecting with Linux need to use below commands

* sudo yum install -y yum-utils shadow-utils
* sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
* sudo yum -y install terraform

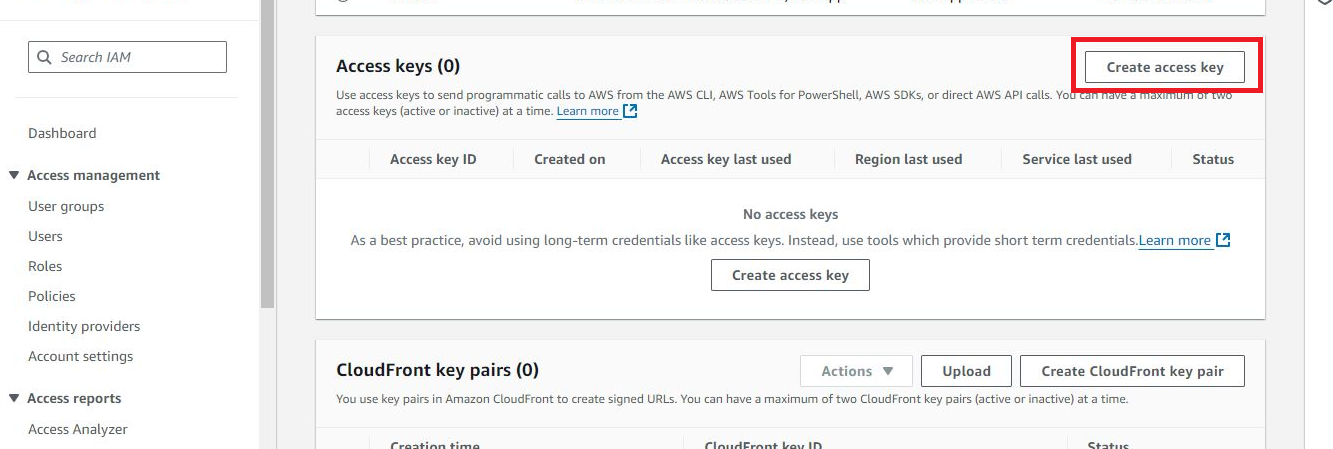


# Step-4:Security credentials creation in AWS Console

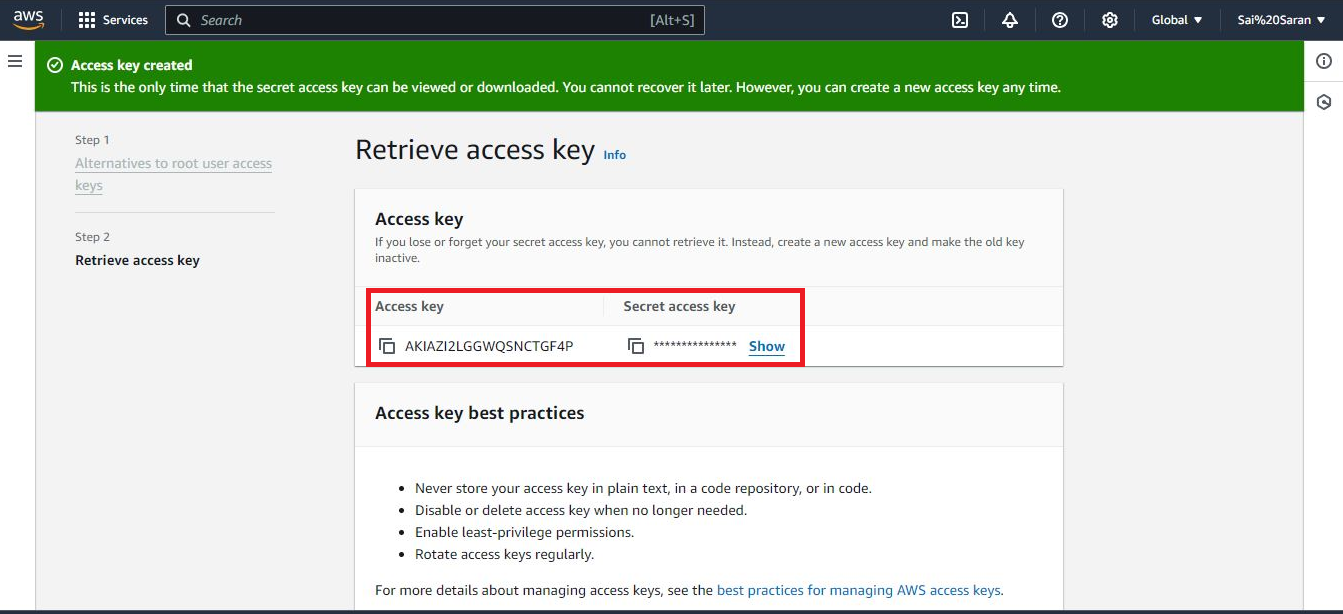
* Open AWS Console on the top right there is profile name in that we can see security credentials



* After opening the security credentials scroll down and check for create access key



* Then Click on next the access was created and we will get access Id and secret key



# Step-5:AWS Configure with Security credentials

After creating the access key in AWS Console then use the command “**aws configure”** to configure the access key with tool

* First provide the access key
* Then give the secret key which was created in AWS Console
* Then provide Region in which we need to reflect the change
* Then provide the output format file type

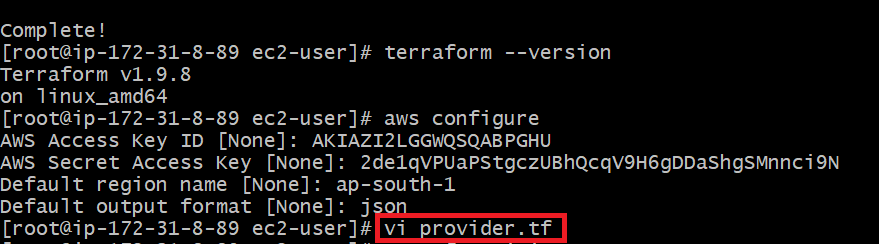


# Step-6: Terraform

* All Terraform files are in the same folder and belong to the same Terraform state file “**.tf” .**
* Make sure to use below commands to avoid unnecessary errors
* **terraform init** command to initialize the Terraform working directory with the AWS plugins
* **terraform validate** to verify your Terraform HCL file
* **terraform plan** to check out the desired changes on every Terraform file creation
* **terraform apply** to create the resources in AWS

# Step-7: AWS Terraform Provider

* First create one file in EC2 instance but using any editor like vim,nano,etc…,



* In the file need to enter the below code and exit from file by using **:wq** command
* In the code need to change the region in which we need to do

**terraform {**

**required\_providers {**

**aws = {**

**source = "hashicorp/aws"**

**version = "3.69.0"**

**}**

**}**

**}**

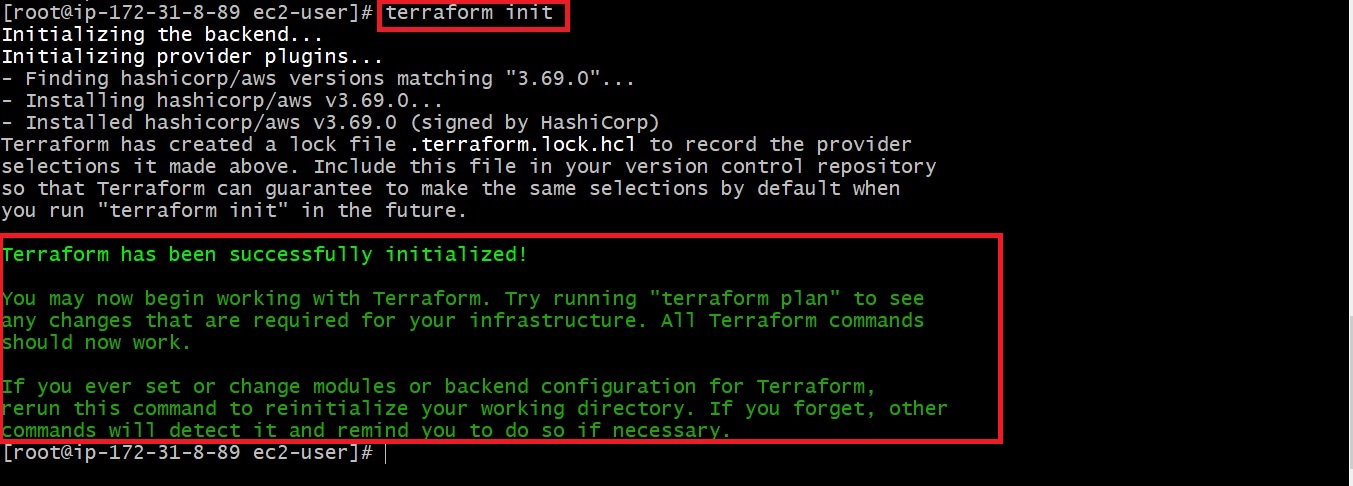
**provider "aws" {**

**profile = "default"**

**region = "us-east-2"**

**}**

* After exit from the file need to use **“terraform init”** commandto initialize the Terraform working directory



# Step-8: Create an AWS CloudWatch dashboard

* After initialize terraform with provider details  we will create a dashboard that monitors the Maximum CPU utilization of all running EC2 instances in your AWS account
* For creating dashboard enter below code in the file

**resource "aws\_cloudwatch\_dashboard" "EC2\_Dashboard" {**

**dashboard\_name = "EC2-Dashboard"**

**dashboard\_body = <<EOF**

**{**

**"widgets": [**

**{**

**"type": "explorer",**

**"width": 24,**

**"height": 15,**

**"x": 0,**

**"y": 0,**

**"properties": {**

**"metrics": [**

**{**

**"metricName": "CPUUtilization",**

**"resourceType": "AWS::EC2::Instance",**

**"stat": "Maximum"**

**}**

**],**

**"aggregateBy": {**

**"key": "InstanceType",**

**"func": "MAX"**

**},**

**"labels": [**

**{**

**"key": "State",**

**"value": "running"**

**}**

**],**

**"widgetOptions": {**

**"legend": {**

**"position": "bottom"**

**},**

**"view": "timeSeries",**

**"rowsPerPage": 8,**

**"widgetsPerRow": 2**

**},**

**"period": 60,**

**"title": "Running EC2 Instances CPUUtilization"**

**}**

**}**

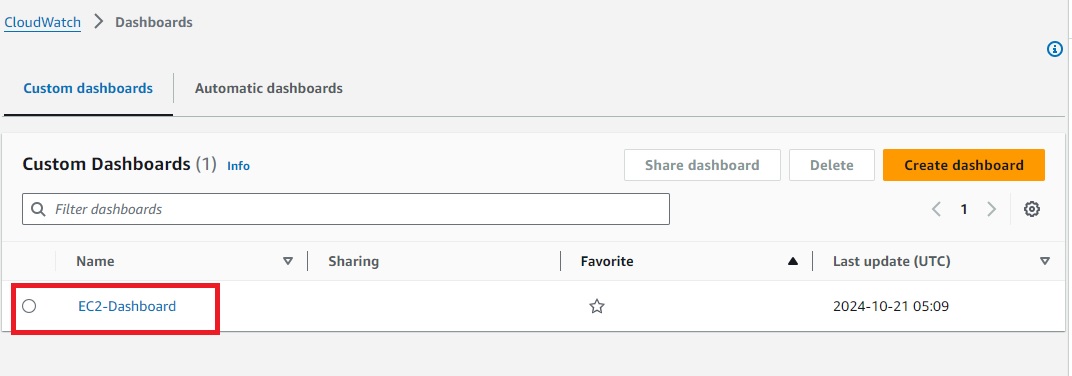
**]**

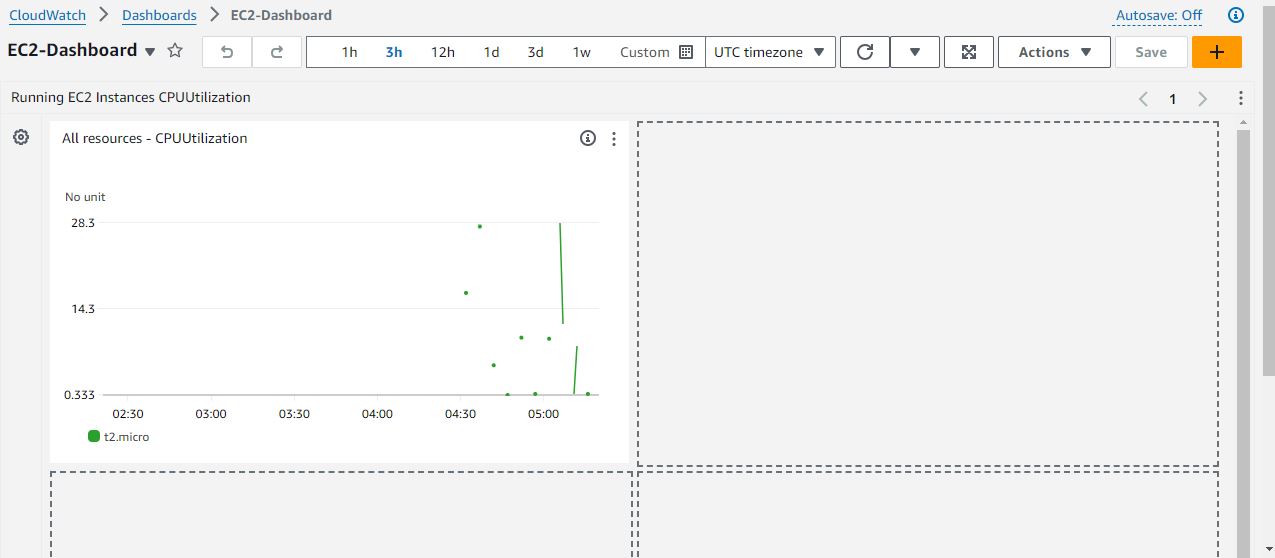
**}**

**EOF**

**}**

* After that run the **terraform plan**  to check out the desired changes on every Terraform file creation
* Then run **terraform apply** to create the resources in AWS
* Then you can log in to the AWS console and access the CloudWatch service dashboard, where you should be able to see the maximum CPU utilization of all the running EC2 instances in your AWS account





### Step-9: **Create an AWS CloudWatch metric alarm**

### To create an AWS CloudWatch metric alarm

### you can use the [aws\_cloudwatch\_metric\_alarm](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudwatch_metric_alarm) resource and pass the required arguments, such as the alarm\_name, comparison\_operator, and evaluation\_periods

* For creating Cloudwatch metric alarm enter below code in the file

**resource "aws\_cloudwatch\_metric\_alarm" "EC2\_CPU\_Usage\_Alarm" {**

**alarm\_name = "EC2\_CPU\_Usage\_Alarm"**

**comparison\_operator = "GreaterThanOrEqualToThreshold"**

**evaluation\_periods = "2"**

**metric\_name = "CPUUtilization"**

**namespace = "AWS/EC2"**

**period = "60"**

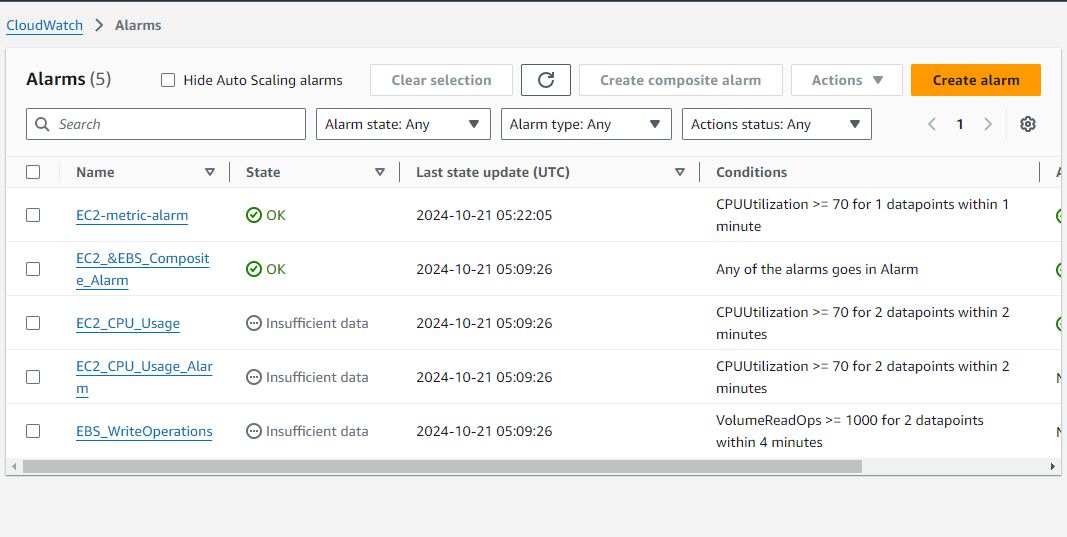
**statistic = "Average"**

**threshold = "70"**

**alarm\_description = "This metric monitors ec2 cpu utilization exceeding 70%"**

**}**

* After that run the **terraform plan**  to check out the desired changes on every Terraform file creation
* Then run **terraform apply** to create the resources in AWS
* Then you can log in to the AWS console and access the CloudWatch service check for all alarms



### Step-10: **Create an AWS CloudWatch metric alarm with EC2 Auto-scaling**

* You can create an AWS CloudWatch metric alarm that triggers multiple actions depending on the defined CloudWatch metric alarm conditions
* In the following example, we shall create an [AWS CloudWatch metric alarm that monitors the average CPU usage and triggers an AWS AutoScaling](https://hands-on.cloud/managing-aws-auto-scaling-using-terraform/" \o "Managing AWS AutoScaling using Terraform" \t "https://hands-on.cloud/terraform-cloudwatch-examples/_blank) policy to spin up more EC2 instances in an AWS AutoScaling group, should the average CPU Utilization average at 70 % for one minute
* For creating CloudWatch metric alarm with EC2 Auto-scaling enter below code in the file
* In code need to change Image Id with our instance ami id and change the availability zone with EC2 instance zone

**resource "aws\_launch\_template" "EC2\_Launch\_Template" {**

**name\_prefix = "EC2-Launch-Template"**

**image\_id = "ami-0f540030bb04d884a"**

**instance\_type = "t2.micro"**

**}**

**resource "aws\_autoscaling\_group" "EC2\_AutoScaling\_Group" {**

**availability\_zones = ["us-east-2b"]**

**desired\_capacity = 1**

**max\_size = 5**

**min\_size = 1**

**launch\_template {**

**id = aws\_launch\_template.EC2\_Launch\_Template.id**

**version = "$Latest"**

**}**

**depends\_on = [**

**aws\_launch\_template.EC2\_Launch\_Template,**

**]**

**}**

**resource "aws\_autoscaling\_policy" "EC2\_AutoScaling\_Policy" {**

**name = "EC2-AutoScaling-Policy"**

**scaling\_adjustment = 2**

**adjustment\_type = "ChangeInCapacity"**

**cooldown = 60**

**autoscaling\_group\_name = aws\_autoscaling\_group.EC2\_AutoScaling\_Group.name**

**depends\_on = [**

**aws\_autoscaling\_group.EC2\_AutoScaling\_Group,**

**]**

**}**

**resource "aws\_cloudwatch\_metric\_alarm" "EC2\_metric\_alarm" {**

**alarm\_name = "EC2-metric-alarm"**

**comparison\_operator = "GreaterThanOrEqualToThreshold"**

**evaluation\_periods = "1"**

**metric\_name = "CPUUtilization"**

**namespace = "AWS/EC2"**

**period = "60"**

**statistic = "Average"**

**threshold = "70"**

**depends\_on = [**

**aws\_autoscaling\_group.EC2\_AutoScaling\_Group,**

**]**

**dimensions = {**

**AutoScalingGroupName = aws\_autoscaling\_group.EC2\_AutoScaling\_Group.name**

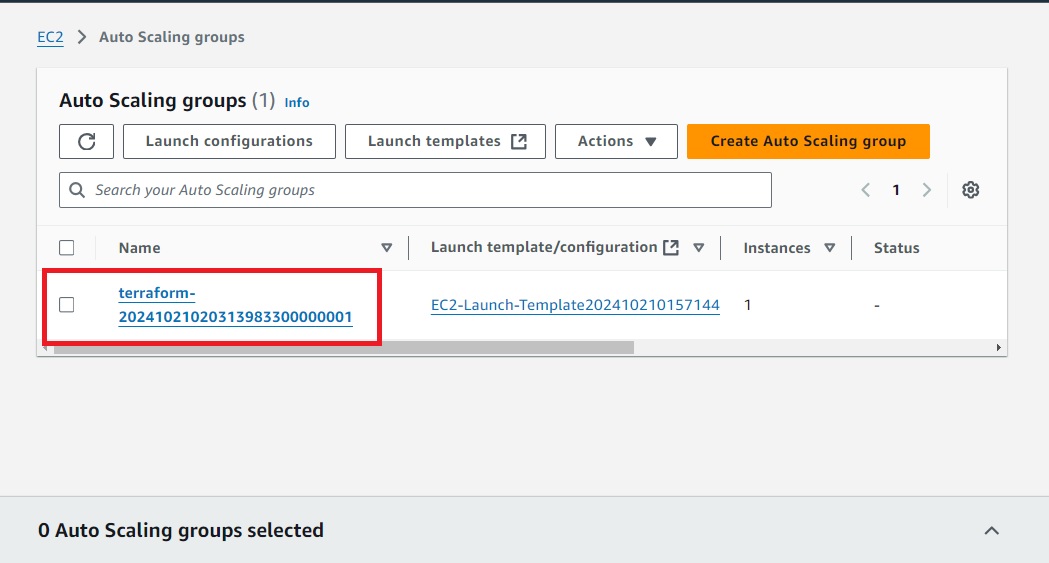
**}**

**alarm\_description = "This metric monitors ec2 cpu utilization"**

**alarm\_actions = [aws\_autoscaling\_policy.EC2\_AutoScaling\_Policy.arn]**

**}**

* After that run the **terraform plan**  to check out the desired changes on every Terraform file creation
* Then run **terraform apply** to create the resources in AWS
* Then you can log in to the AWS console and access the EC2 service check for Autoscaling on left side



### Step-11: **Create an AWS CloudWatch metric stream destination as Amazon S3**

* To create an AWS CloudWatch metric stream, use the aws\_cloudwatch\_metric\_stream and assign the required arguments, such as the firehose\_arn, role\_arn, and output\_format. The include\_filter argument is used to specify the namespaces you want to send metrics to S3. In this case, we are using AWS/EC2 and AWS/EBS.
* Below are the detail explanation of code

1. **CloudWatch Metric Stream** (aws\_cloudwatch\_metric\_stream):
   1. A metric stream named "my-metric-stream".
   2. Sends metrics from namespaces AWS/EC2 and AWS/EBS to the Firehose delivery stream.
   3. The output format is JSON.
2. **IAM Role for Metric Stream** (aws\_iam\_role.metric\_stream\_to\_firehose):
   1. This IAM role allows the CloudWatch Metric Stream service to assume the role and send metrics to Kinesis Firehose.
3. **IAM Policy for Metric Stream to Firehose** (aws\_iam\_role\_policy.metric\_stream\_to\_firehose):
   1. The policy grants the necessary permissions (firehose:PutRecord and firehose:PutRecordBatch) to send data to the Firehose stream.
4. **S3 Bucket** (aws\_s3\_bucket.bucket):
   1. A private S3 bucket is created to store the streamed metric data.
5. **IAM Role for Firehose** (aws\_iam\_role.firehose\_to\_s3):
   1. This role is assumed by Kinesis Firehose to write data into the S3 bucket.
6. **IAM Policy for Firehose to S3** (aws\_iam\_role\_policy.firehose\_to\_s3):
   1. Provides Firehose the necessary permissions to interact with the S3 bucket, such as uploading and managing objects.
7. **Kinesis Firehose Delivery Stream** (aws\_kinesis\_firehose\_delivery\_stream.s3\_stream):
   1. A Kinesis Firehose delivery stream that forwards the metric stream data to the S3 bucket.

* **What is firehose?**
* Amazon Kinesis Data Firehose is a fully managed service used to reliably stream and load real-time data into data lakes, data warehouses, and analytics services. It is designed for scenarios where you need to capture and deliver large volumes of streaming data with minimal configuration and management
* For creating AWS CloudWatch metric stream destination as Amazon S3 enter below code in the file

**resource "aws\_cloudwatch\_metric\_stream" "main" {**

**name = "my-metric-stream"**

**role\_arn = aws\_iam\_role.metric\_stream\_to\_firehose.arn**

**firehose\_arn = aws\_kinesis\_firehose\_delivery\_stream.s3\_stream.arn**

**output\_format = "json"**

**include\_filter {**

**namespace = "AWS/EC2"**

**}**

**include\_filter {**

**namespace = "AWS/EBS"**

**}**

**}**

**resource "aws\_iam\_role" "metric\_stream\_to\_firehose" {**

**name = "metric\_stream\_to\_firehose\_role"**

**assume\_role\_policy = <<EOF**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Action": "sts:AssumeRole",**

**"Principal": {**

**"Service": "streams.metrics.cloudwatch.amazonaws.com"**

**},**

**"Effect": "Allow",**

**"Sid": ""**

**}**

**]**

**}**

**EOF**

**}**

**resource "aws\_iam\_role\_policy" "metric\_stream\_to\_firehose" {**

**name = "default"**

**role = aws\_iam\_role.metric\_stream\_to\_firehose.id**

**policy = <<EOF**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": [**

**"firehose:PutRecord",**

**"firehose:PutRecordBatch"**

**],**

**"Resource": "${aws\_kinesis\_firehose\_delivery\_stream.s3\_stream.arn}"**

**}**

**]**

**}**

**EOF**

**}**

**resource "aws\_s3\_bucket" "bucket" {**

**bucket = "test-metric-stream-bucket"**

**acl = "private"**

**}**

**resource "aws\_iam\_role" "firehose\_to\_s3" {**

**assume\_role\_policy = <<EOF**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Action": "sts:AssumeRole",**

**"Principal": {**

**"Service": "firehose.amazonaws.com"**

**},**

**"Effect": "Allow",**

**"Sid": ""**

**}**

**]**

**}**

**EOF**

**}**

**resource "aws\_iam\_role\_policy" "firehose\_to\_s3" {**

**name = "default"**

**role = aws\_iam\_role.firehose\_to\_s3.id**

**policy = <<EOF**

**{**

**"Version": "2012-10-17",**

**"Statement": [**

**{**

**"Effect": "Allow",**

**"Action": [**

**"s3:AbortMultipartUpload",**

**"s3:GetBucketLocation",**

**"s3:GetObject",**

**"s3:ListBucket",**

**"s3:ListBucketMultipartUploads",**

**"s3:PutObject"**

**],**

**"Resource": [**

**"${aws\_s3\_bucket.bucket.arn}",**

**"${aws\_s3\_bucket.bucket.arn}/\*"**

**]**

**}**

**]**

**}**

**EOF**

**}**

**resource "aws\_kinesis\_firehose\_delivery\_stream" "s3\_stream" {**

**name = "metric-stream-test-stream"**

**destination = "s3"**

**s3\_configuration {**

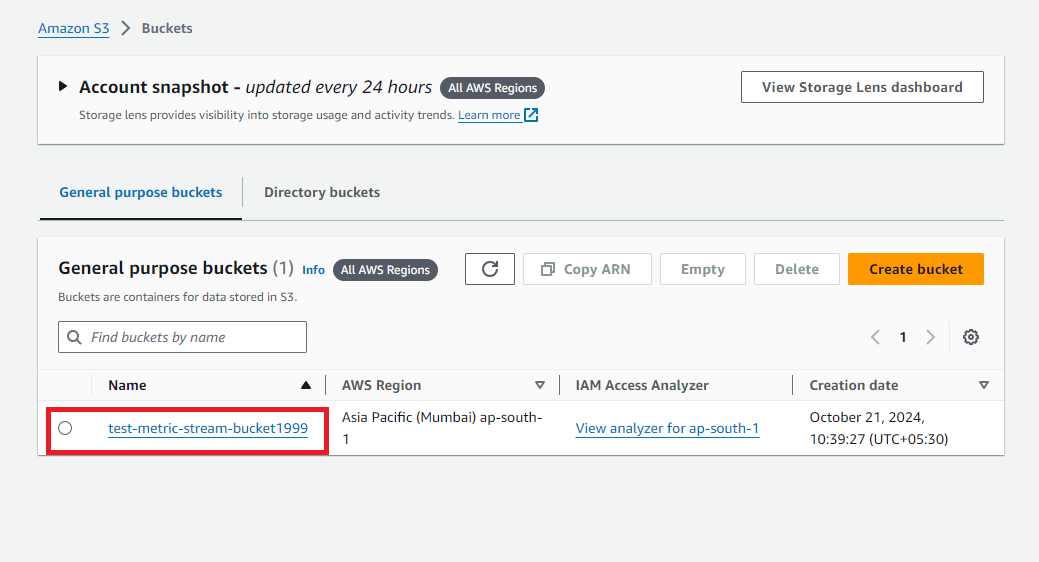
**role\_arn = aws\_iam\_role.firehose\_to\_s3.arn**

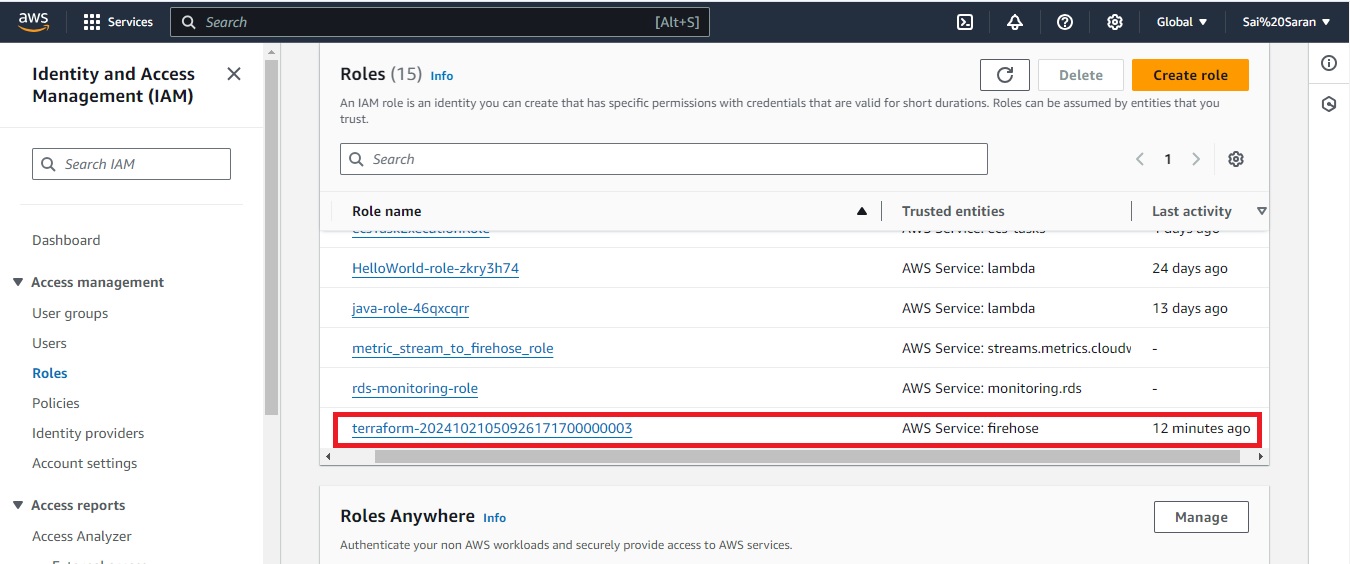
**bucket\_arn = aws\_s3\_bucket.bucket.arn**

**}**

**}**

* After that run the **terraform plan**  to check out the desired changes on every Terraform file creation
* Then run **terraform apply** to create the resources in AWS
* Then you can log in to the AWS console and access the S3 service and IAM Role to check resources are created





### Step-12: **Create an AWS CloudWatch composite alarm**

* AWS[CloudWatch Composite alarms](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Create_Composite_Alarm.html) are used to monitor two or more alarms to determine the alarm state and take action.
* The most common use of Composite alarms is to reduce alarm notifications by grouping multiple alarms and defining which metrics within’ alarms should trigger the different alarm states
* To create a Composite alarm using Terraform, you can use the [aws\_cloudwatch\_composite\_alarm](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudwatch_composite_alarm) resource and pass the required argument, which is the alarm\_name
* The [aws\_sns\_topic](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/sns_topic) and the [aws\_sns\_topic\_subscription](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/sns_topic_subscription) resources are used to configure notifications
* For creating AWS CloudWatch composite alarm enter below code in the file
* In the code need to change Email Id for those who want to receive the notification regarding monitoring

**resource "aws\_cloudwatch\_composite\_alarm" "EC2\_and\_EBS" {**

**alarm\_description = "Composite alarm that monitors CPUUtilization and EBS Volume Write Operations"**

**alarm\_name = "EC2\_&EBS\_Composite\_Alarm"**

**alarm\_actions = [aws\_sns\_topic.EC2\_and\_EBS\_topic.arn]**

**alarm\_rule = "ALARM(${aws\_cloudwatch\_metric\_alarm.EC2\_CPU\_Usage\_Alarm.alarm\_name}) OR ALARM(${aws\_cloudwatch\_metric\_alarm.EBS\_WriteOperations.alarm\_name})"**

**depends\_on = [**

**aws\_cloudwatch\_metric\_alarm.EC2\_CPU\_Usage\_Alarm,**

**aws\_cloudwatch\_metric\_alarm.EBS\_WriteOperations,**

**aws\_sns\_topic.EC2\_and\_EBS\_topic,**

**aws\_sns\_topic\_subscription.EC2\_and\_EBS\_Subscription**

**]**

**}**

**resource "aws\_cloudwatch\_metric\_alarm" "EC2\_CPU\_Usage\_Alarm" {**

**alarm\_name = "EC2\_CPU\_Usage\_Alarm"**

**comparison\_operator = "GreaterThanOrEqualToThreshold"**

**evaluation\_periods = "2"**

**metric\_name = "CPUUtilization"**

**namespace = "AWS/EC2"**

**period = "60"**

**statistic = "Average"**

**threshold = "70"**

**alarm\_description = "This metric monitors ec2 cpu utilization exceeding 70%"**

**}**

**resource "aws\_cloudwatch\_metric\_alarm" "EBS\_WriteOperations" {**

**alarm\_name = "EBS\_WriteOperations"**

**comparison\_operator = "GreaterThanOrEqualToThreshold"**

**evaluation\_periods = "2"**

**metric\_name = "VolumeReadOps"**

**namespace = "AWS/EC2"**

**period = "120"**

**statistic = "Average"**

**threshold = "1000"**

**alarm\_description = "This monitors the average read operations on EBS Volumes in a specified period of time"**

**}**

**resource "aws\_sns\_topic" "EC2\_and\_EBS\_topic" {**

**name = "EC2\_and\_EBS\_topic"**

**}**

**resource "aws\_sns\_topic\_subscription" "EC2\_and\_EBS\_Subscription" {**

**topic\_arn = aws\_sns\_topic.EC2\_and\_EBS\_topic.arn**

**protocol = "email"**

**endpoint = "kelvingalabuzi@gmail.com"**

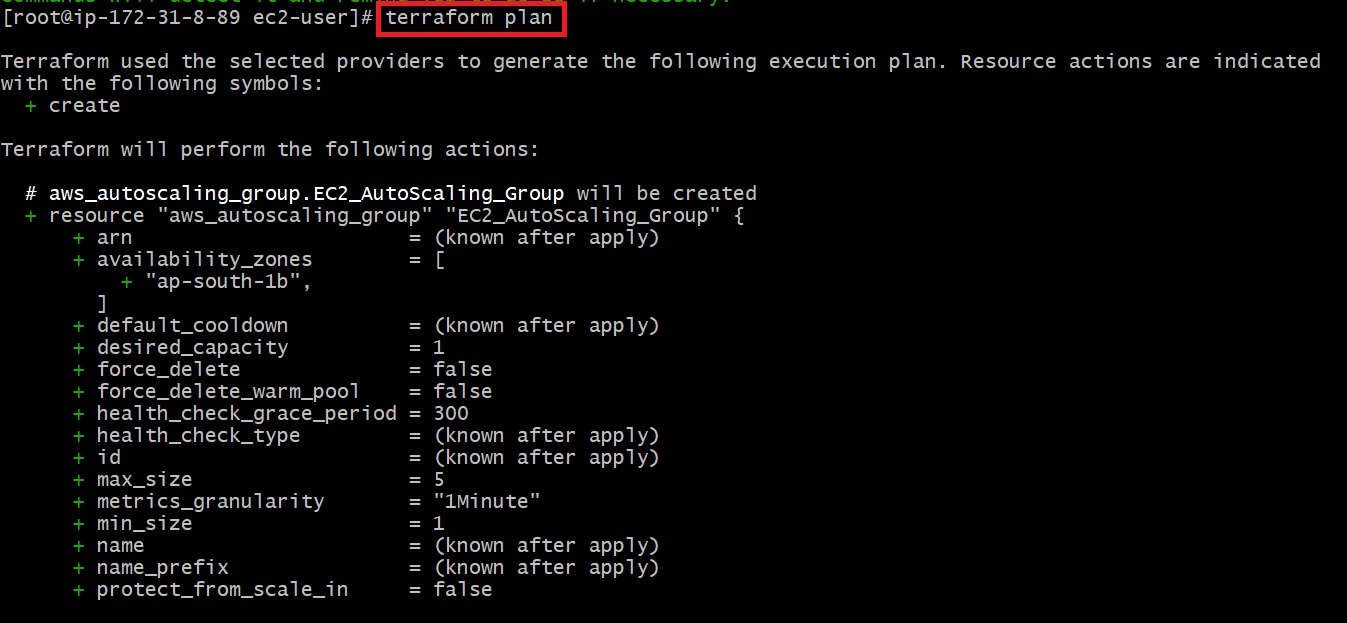
**depends\_on = [**

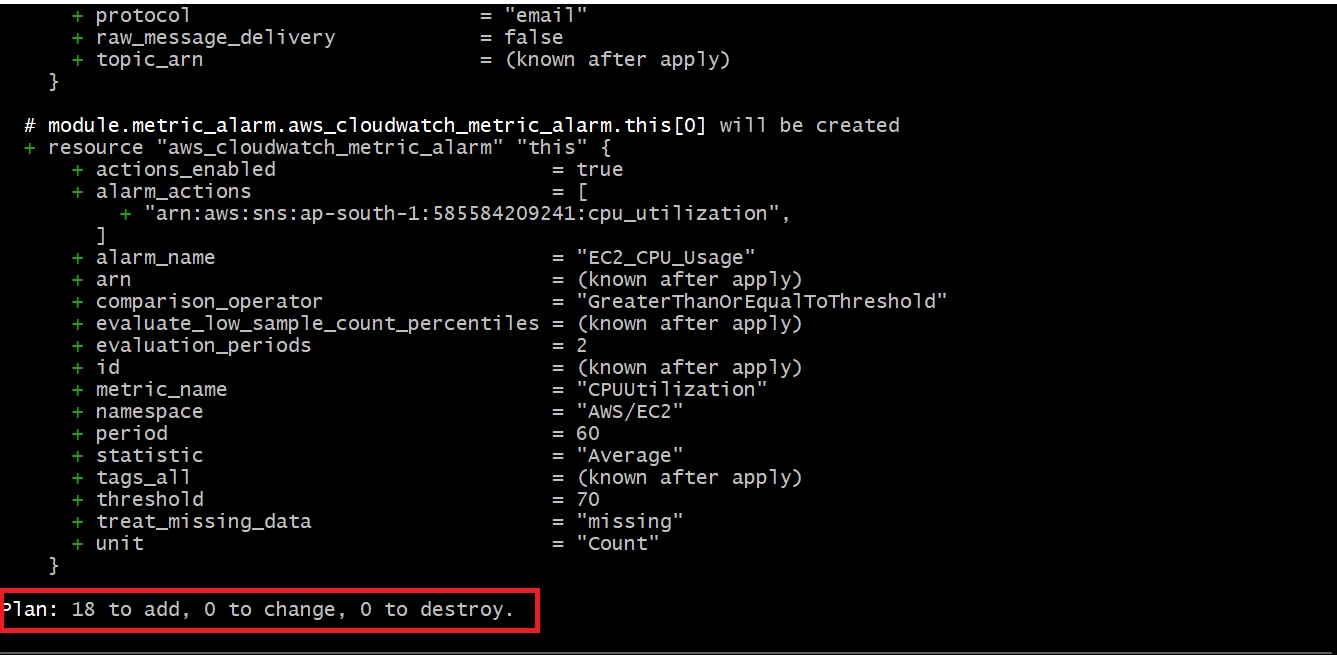
**aws\_sns\_topic.EC2\_and\_EBS\_topic**

**]**

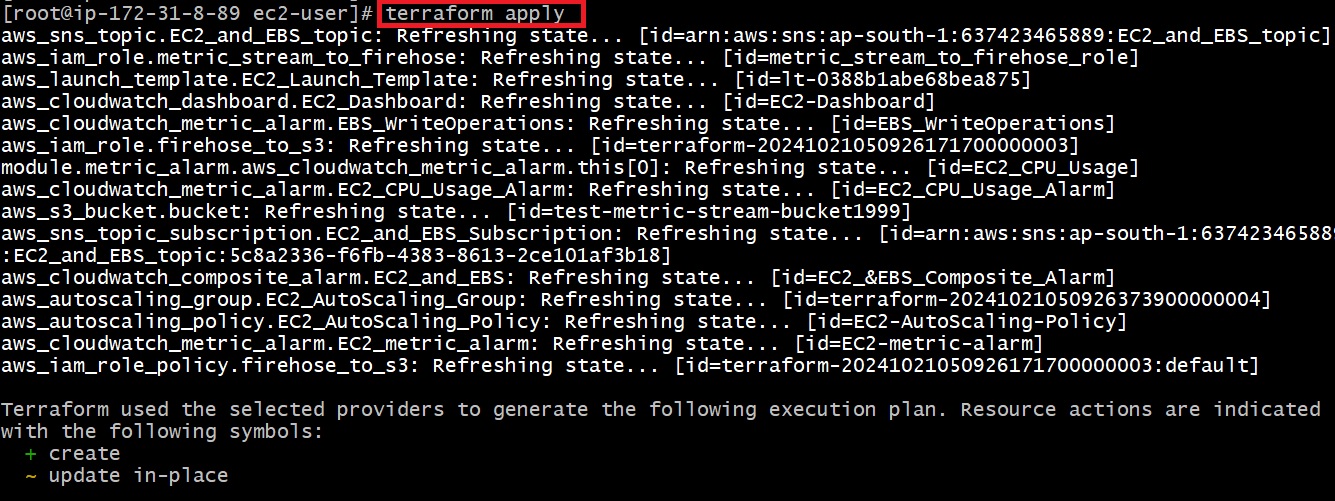
**}**

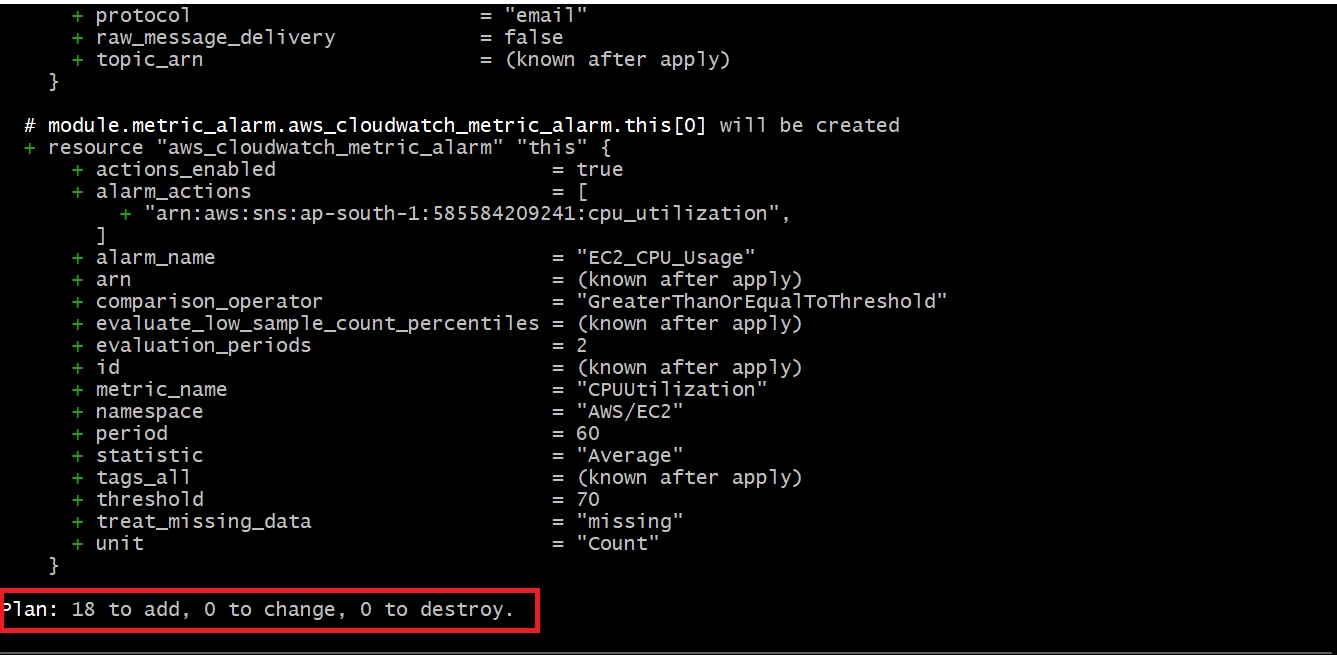
* After that run the **terraform plan**  to check out the desired changes on every Terraform file creation





* Then run **terraform apply** to create the resources in AWS





* Then you can log in to the AWS console and access the SNS and check the Emails for notification.

