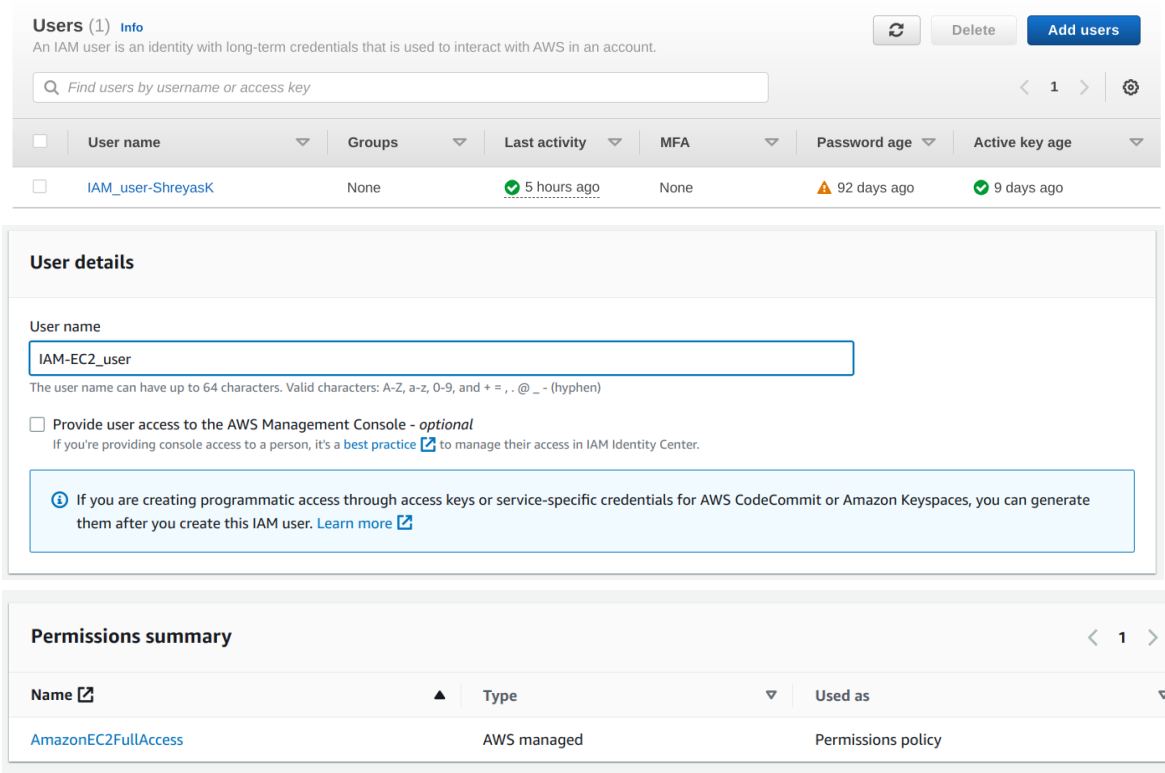
AWS Monitoring

Day 3 - Assignment 27th July 2023

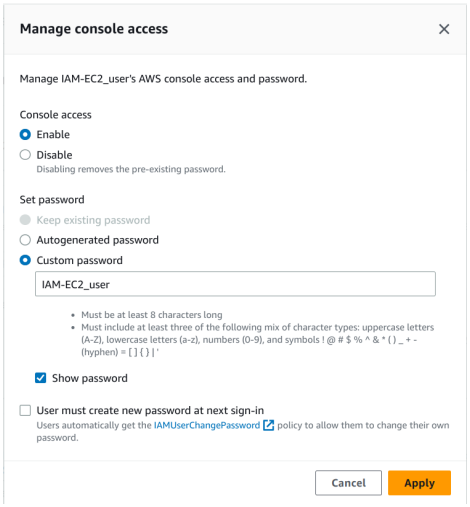
Assignment 1 - Creating multiple profiles and using them on the same server.

Step 1 - Create two IAM users with different permissions assigned to them.

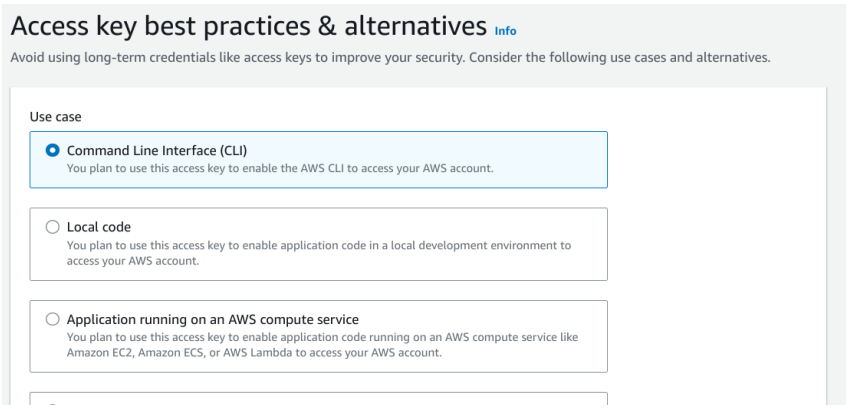
- **IAM user with only EC2 full access permission.**

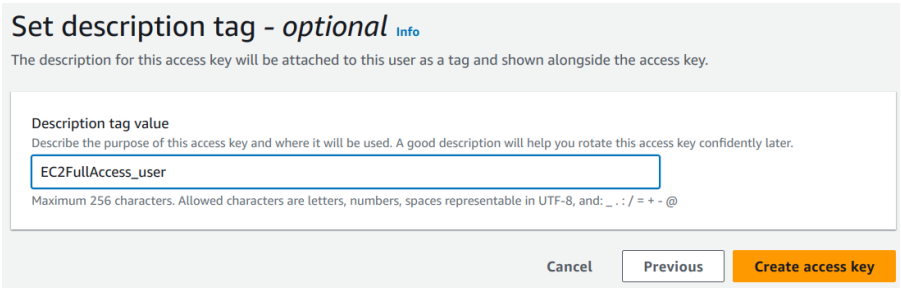


Enable console access for the user and set a password.

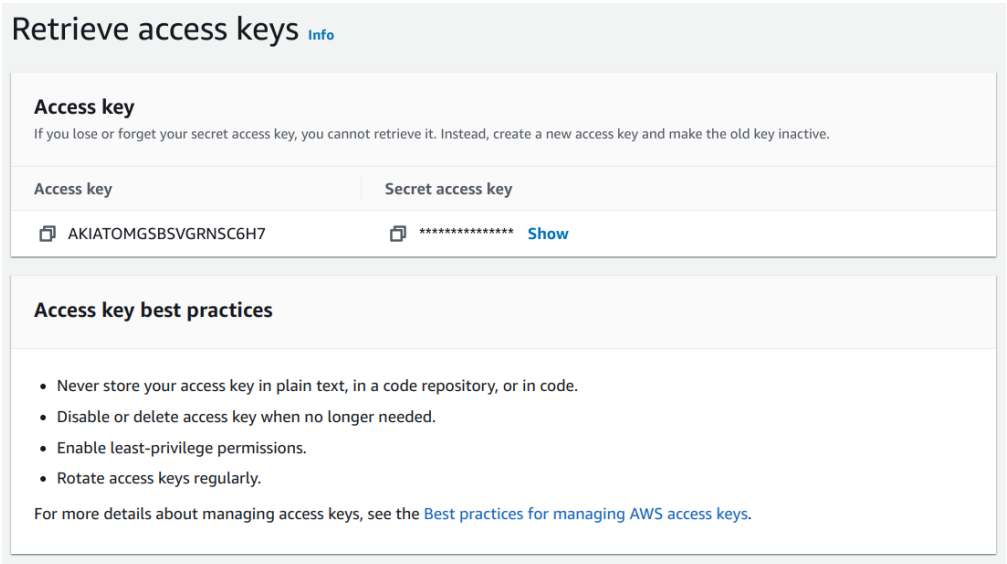


Create an Access Key for the user with command line access.

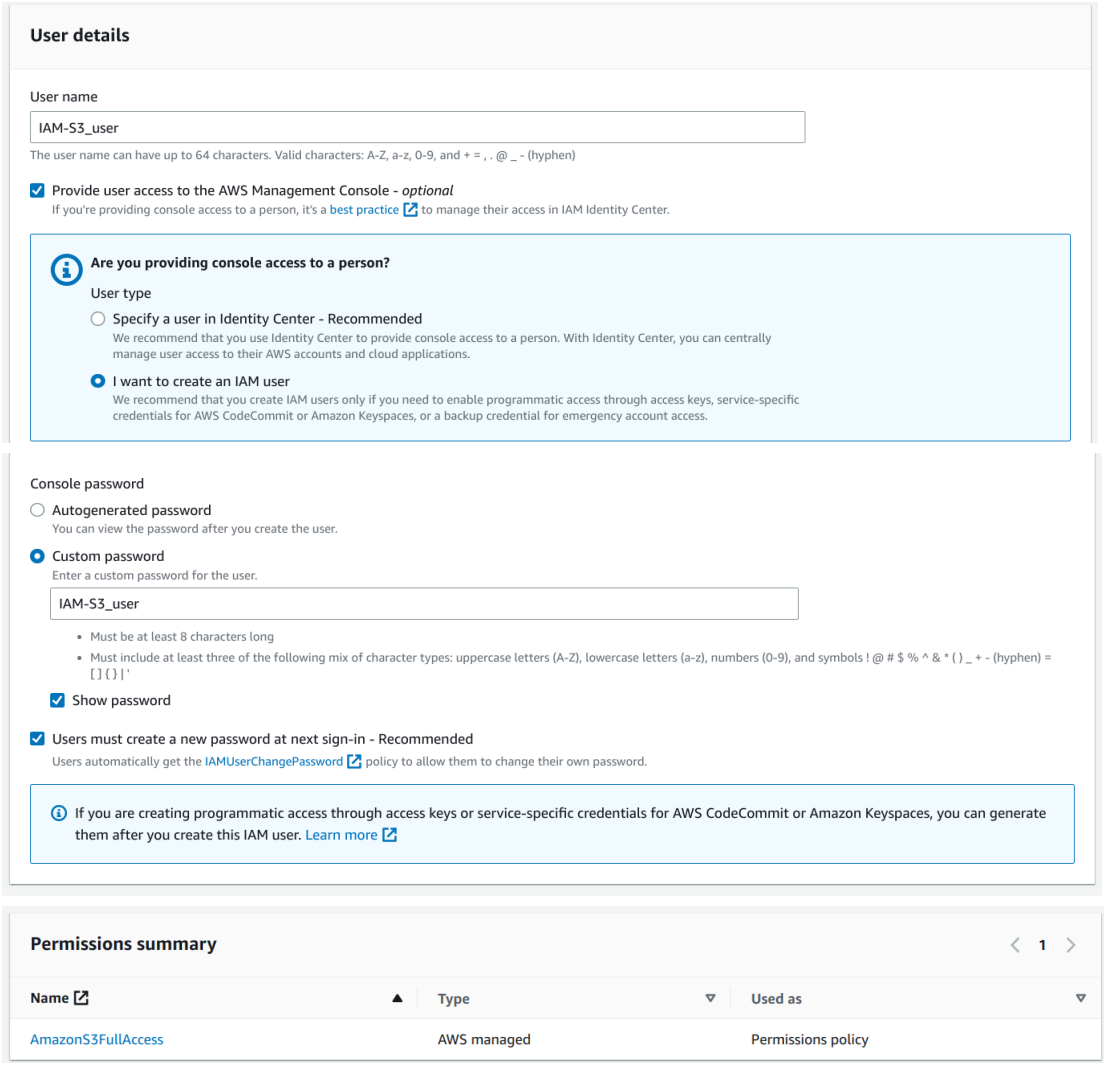




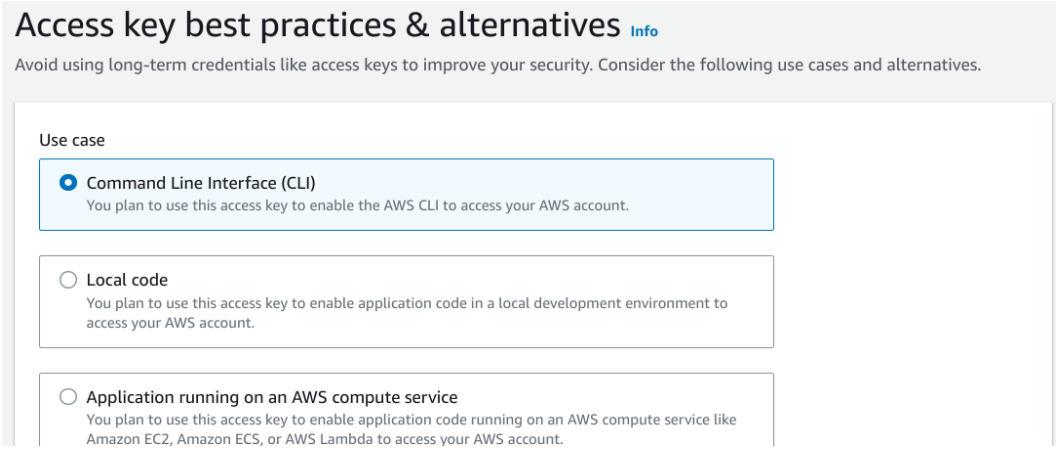
The secret access key should be saved securely and in a durable store as it is retrievable only at the time of creation. Make sure to download the .csv file.

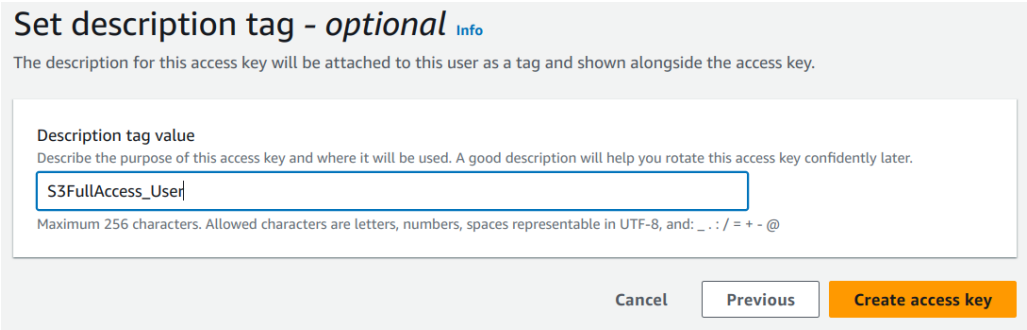


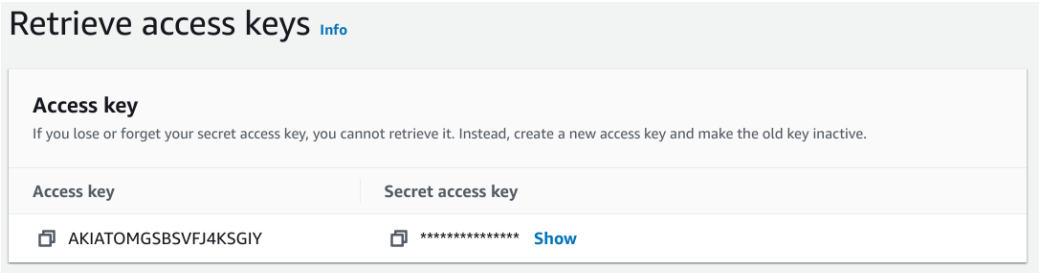
- **IAM user with only S3 full access permission.**



Create an Access Key for the user.

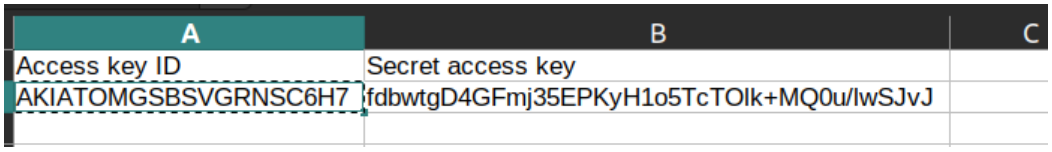




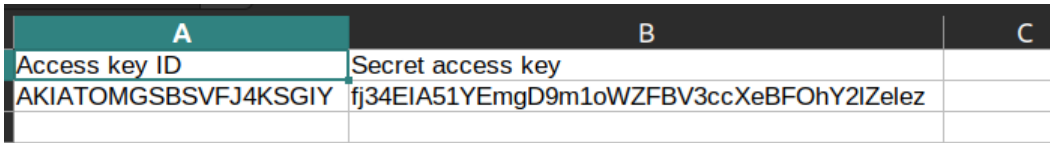


Step 2 - Go to the AWS CLI and run the "aws configure –profile <accountName>" command to set up multiple profiles for different IAM users in a single machine.

The .csv files for both the users' access key and secret access key, downloaded when creating the access keys.



A	B	C
Access key ID	Secret access key	
AKIATOMGSBSVGRNSC6H7	fdbwtgD4GFmj35EPKyH1o5TcTOlk+MQ0u/lwSJvJ	



A	B	C
Access key ID	Secret access key	
AKIATOMGSBSVFJ4KSGIY	fj34EIA51YEmgD9m1oWZFBV3ccXeBFOhY2lZelez	

Run the configure command and enter the access key and secret key values of both the users.

```
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ aws configure --profile ec2_user
AWS Access Key ID [None]: AKIATOMGSBSVGRNSC6H7
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]:
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ aws configure --profile s3_user
AWS Access Key ID [None]: AKIATOMGSBSVFJ4KSGIY
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]:
```

Use the “list-profiles” command to list the profiles.

```
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ awsv2 configure list-profiles
default
ec2_user
s3_user
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$
```

Step 3 - Try to access the S3 buckets with both the profiles and check the output.

With the use of “--profile <profileName>” option with AWS CLI commands we can use multiple profiles or IAM users with different permissions on the same machine.

The profile of “s3_user” can perform actions on S3 buckets as it is the profile of the IAM user with permission of “S3FullAccess”.

But the profile “ec2_user” is not able to perform any action on S3 because the IAM user does not have permissions for S3.

```
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ awsv2 s3 ls --profile s3_user
2023-07-27 10:54:41 aws-cloudtrail-logs-237042273450-300568af
2023-07-11 16:56:55 aws-cloudtrail-logs-237042273450-e4573353
2023-07-11 11:59:34 axcesstestbucket1
2023-07-11 08:59:47 test-bucket-sk
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ awsv2 s3 ls --profile ec2_user

An error occurred (AccessDenied) when calling the ListBuckets operation: Access
Denied
17:39:27 - awscliv2 - ERROR - Command failed with code 254
```

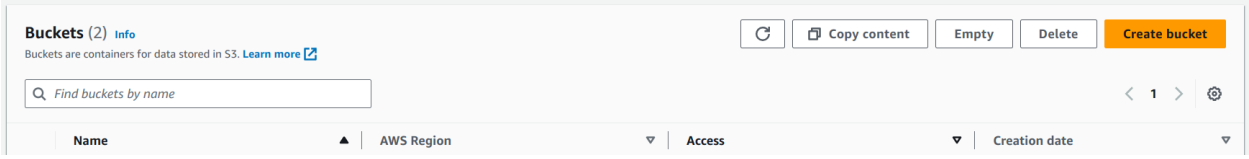
Same with the “ec2_user” as it only has permissions for EC2FullAccess, it cannot access S3 or other services.

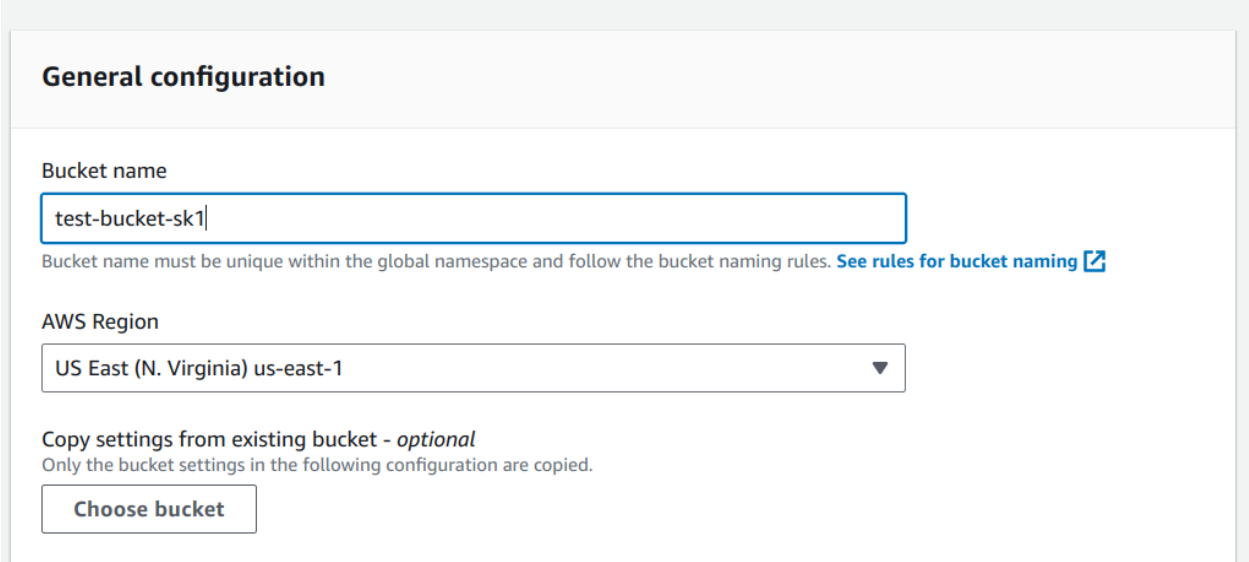
```
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ awsv2 ec2 describe-instances --profile ec2_user
{
    "Reservations": []
}
shreyaskayarkar@rahulraj-TravelMate-P214-53:~$ awsv2 ec2 describe-instances --profile s3_user

An error occurred (UnauthorizedOperation) when calling the DescribeInstances operation: You are not authorized
 to perform this operation.
17:42:25 - awscliv2 - ERROR - Command failed with code 254
```

Assignment 2 - Creating CloudTrail data events for S3.

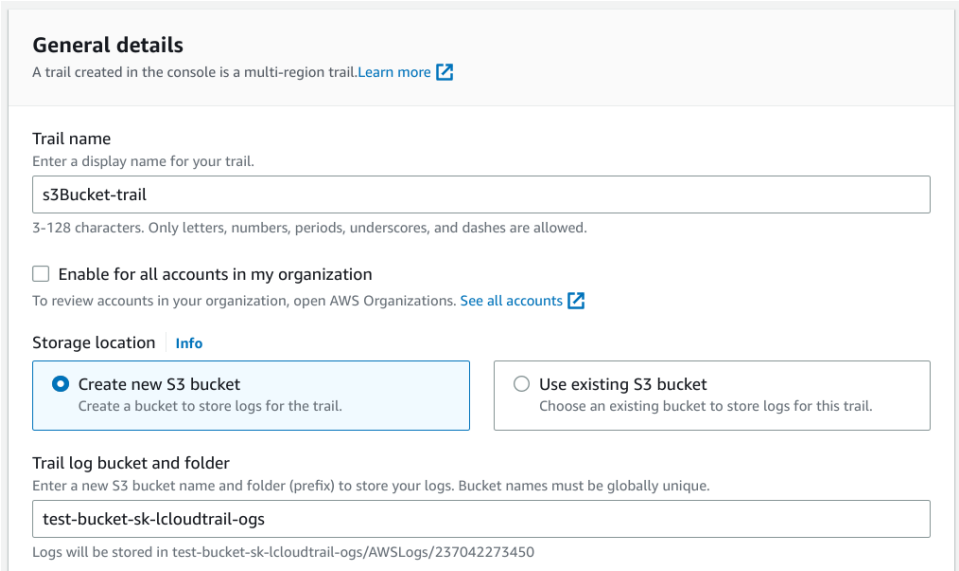
Step 1 - Create a sample S3 bucket.



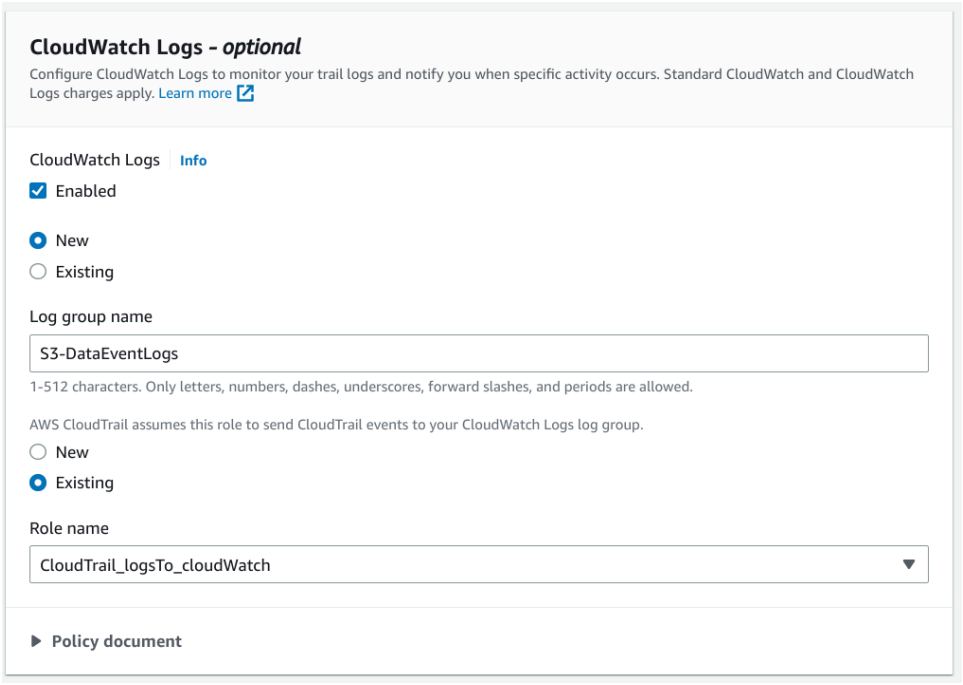


Step 2 - Create a CloudTrail data event for the S3 bucket.

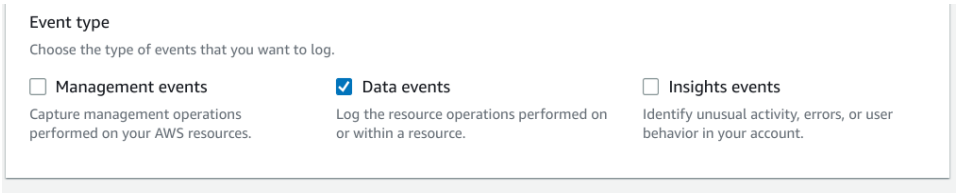
Create a new bucket with a unique and identifiable name. This S3 bucket will store the logs of this trail.



Enable CloudWatch logs, create a new log group which will store the logs coming for this trail.

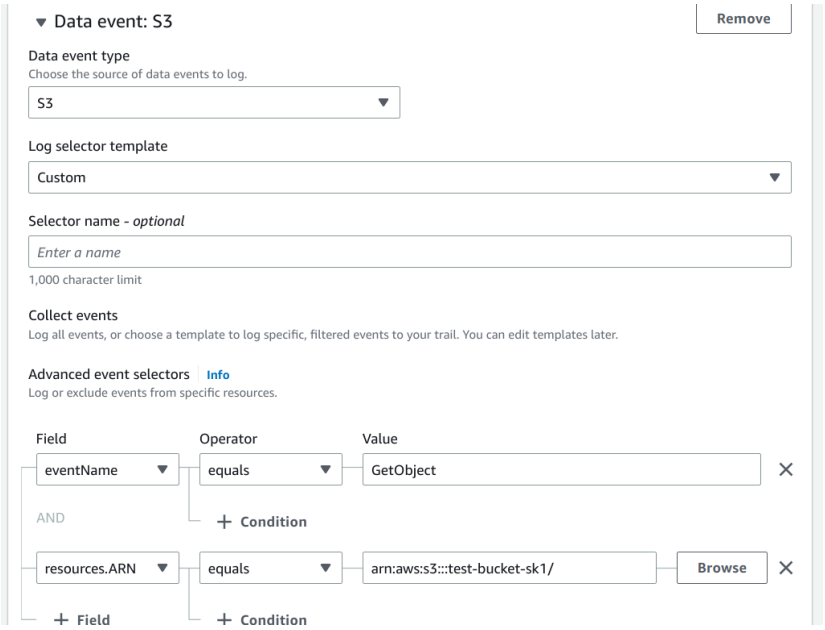


Select “Data events”, which will allow the trail to collect resource level events.

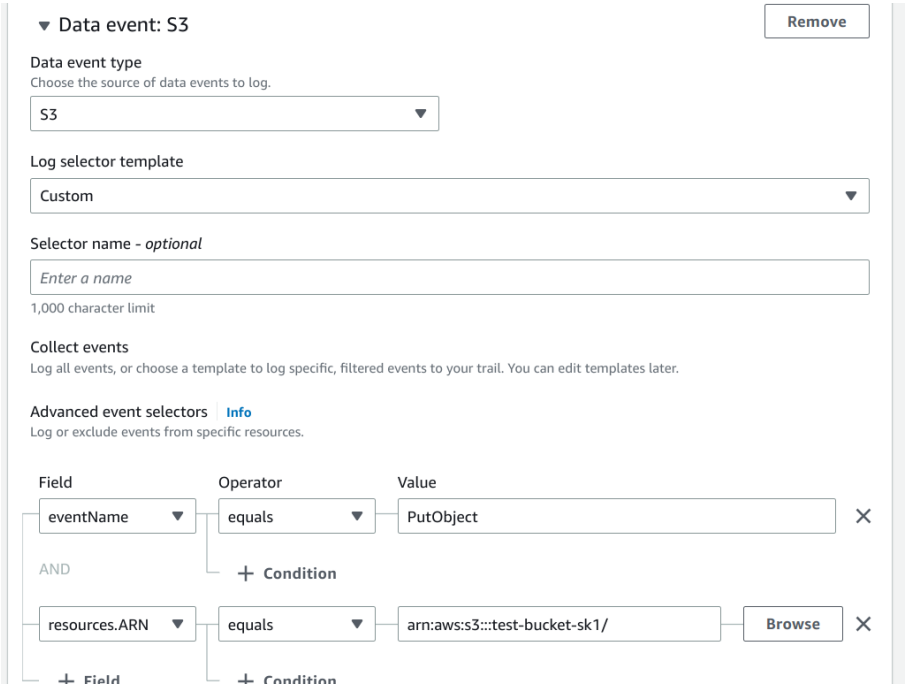


The custom event will allow the trail to collect logs for whenever an object is downloaded from the bucket with the specified ARN.

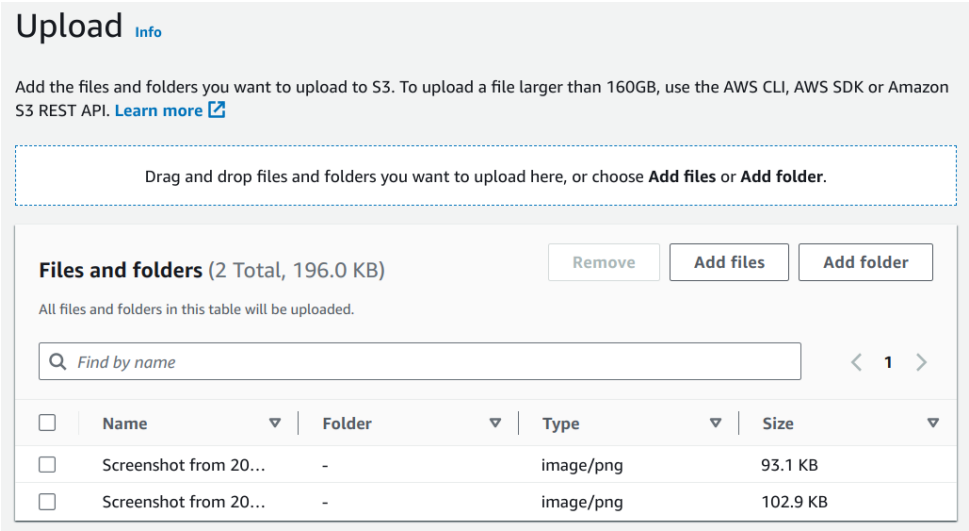
Specifying the resource.ARN field will collect logs only for the put and get object requests with ARN starting from the specified value. It should be ensured that only selected buckets should be selected or the bucket in which the logs are to be stored should be excluded by specifying its resource ARN. It will avoid any logging loop.



This event will collect logs whenever an object is uploaded to the bucket.

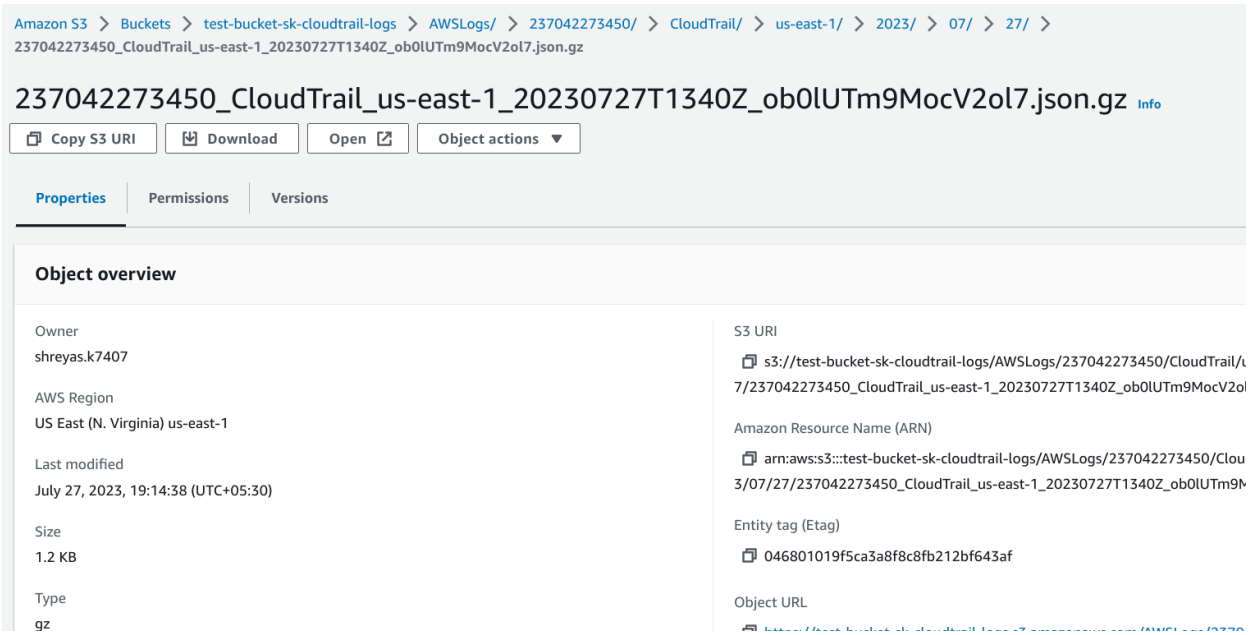


Step 3 - Upload files to the S3 bucket.



Step 4 - Checking the CloudWatch log group for the logs of the “PutObject” event.

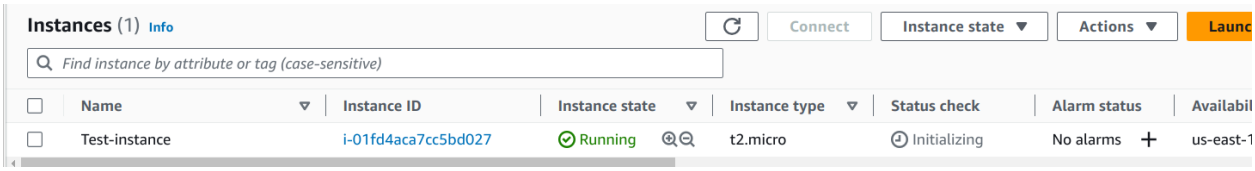
Step 5 - Check the S3 bucket used for storing the logs.



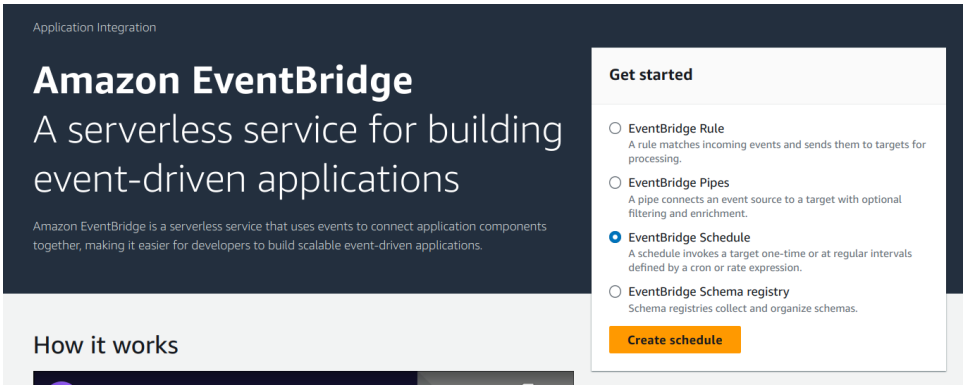

```
{
  "Records": [
    {
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDATOMGSBSVAAKATBQ3F",
        "arn": "arn:aws:iam::237042273450:user/IAM_user-ShreyasK",
        "accountId": "237042273450",
        "accessKeyId": "ASIATOMGSBSVNRT5IPMT",
        "userName": "IAM_user-ShreyasK",
        "sessionContext": {
          "attributes": {
            "creationDate": "2023-07-27T05:47:13Z",
            "mfaAuthenticated": "false"
          }
        }
      },
      "eventTime": "2023-07-27T13:39:34Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "103.6.157.71",
      "userAgent": "[Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.0.0 Safari/537.36]",
      "requestParameters": {
        "X-Amz-Date": "20230727T133932Z",
        "bucketName": "test-bucket-sk1",
        "X-Amz-Algorithm": "AWS4-HMAC-SHA256",
        "x-amz-acl": "bucket-owner-full-control"
```

Assignment 3 - Using event bridge to schedule automatic starting and stopping EC2 instances on a schedule.

Step 1 - Launch an EC2 instance.

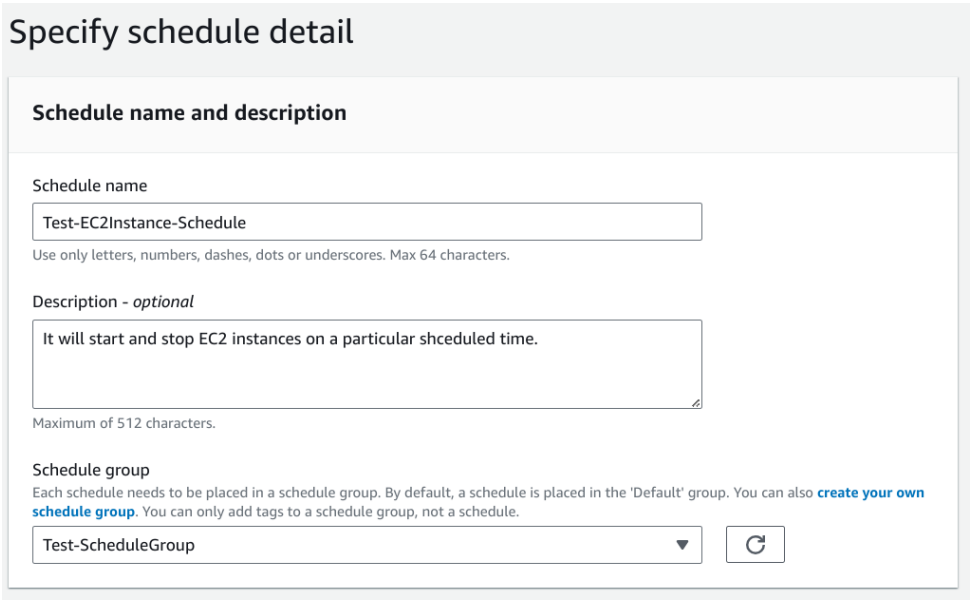


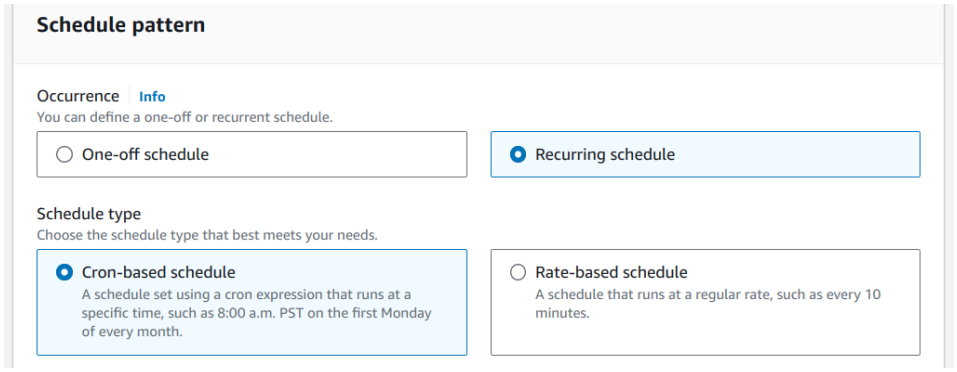
Step 2 - Navigate to the EventBridge console. Select event scheduler and click on “Create schedule”



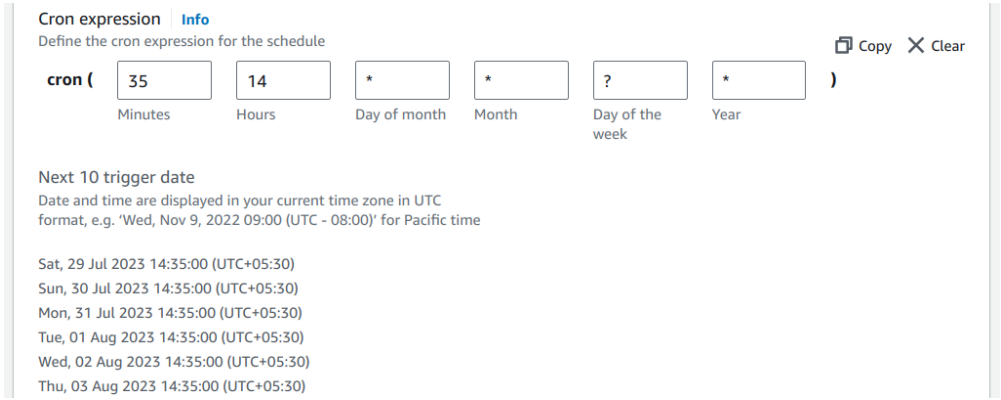
Create a schedule for starting the instance at a scheduled time.

Schedule groups are used to organize the schedules.

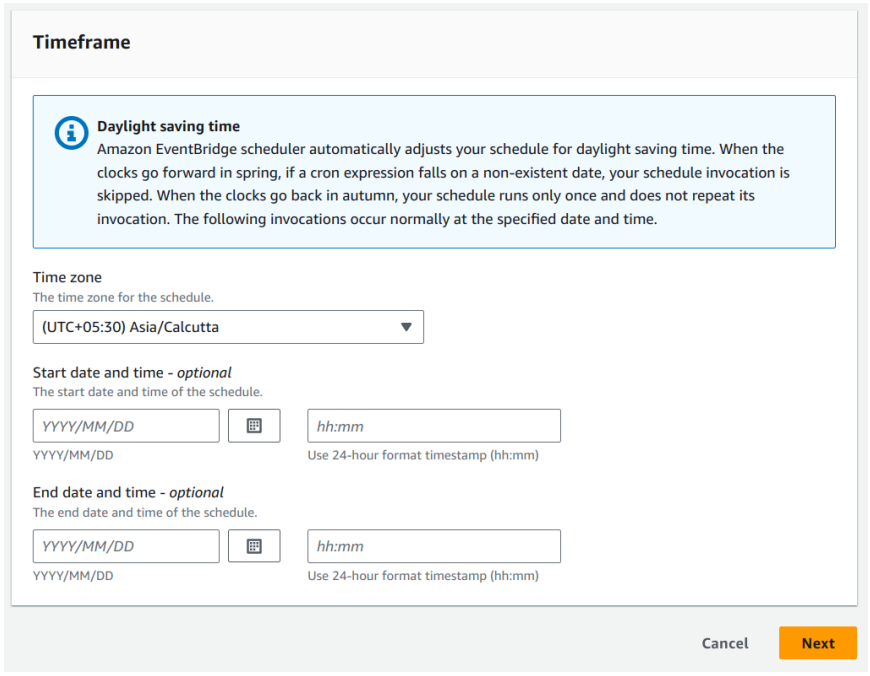


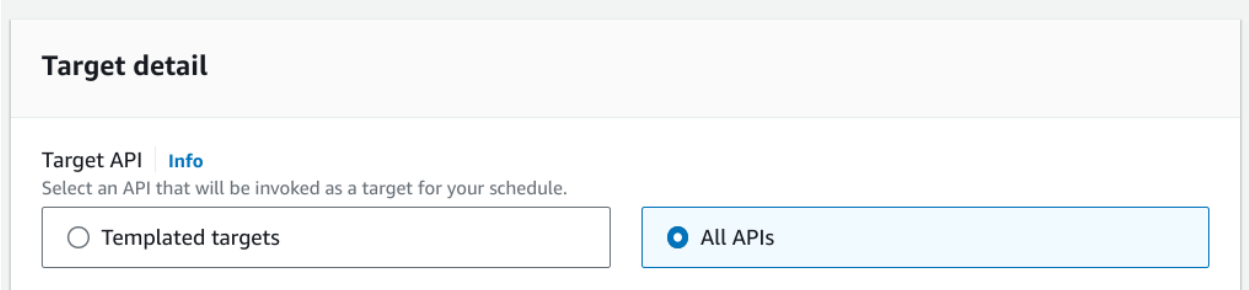


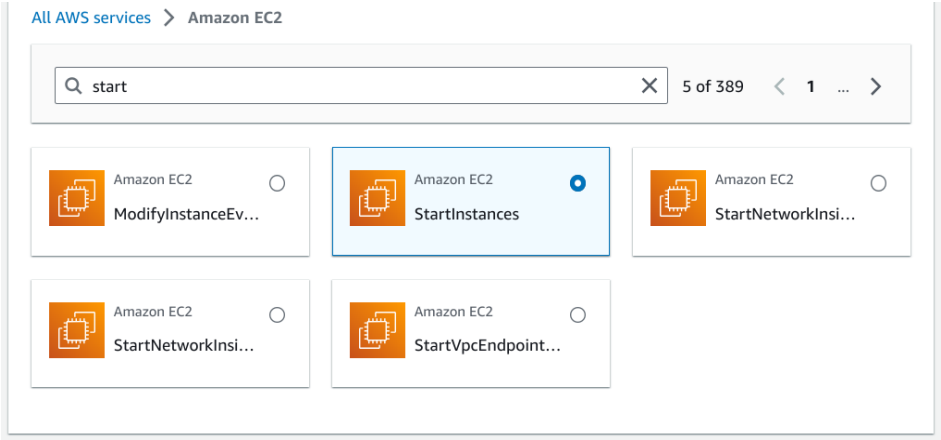
This cron expression will trigger an event everyday at 14:35 hours.



Time will be based on the selected timezone. Here it is set to Asia/Calcutta (IST).

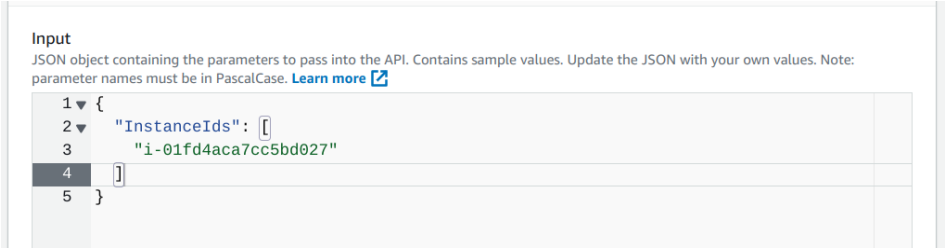




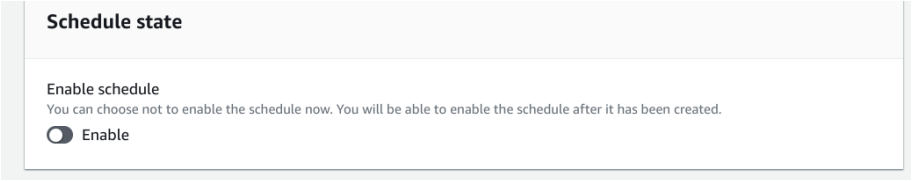


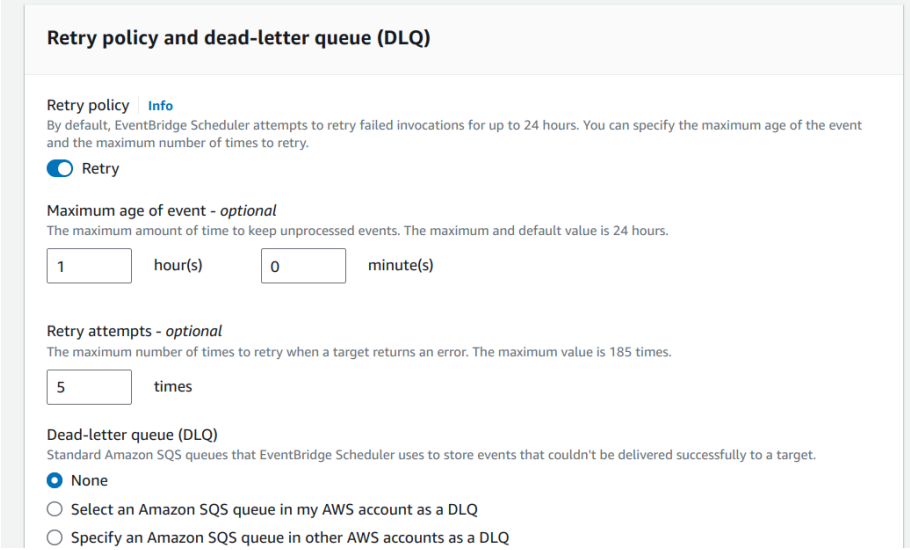
As selected above, the event will be a StartInstance API call for the instance with the instance id specified in the below input section.

Enter the instance id of the instance created in the first step.



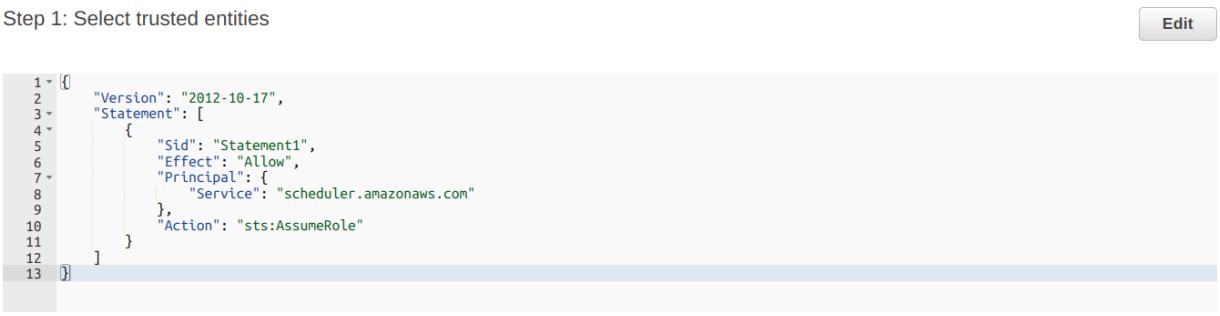
The schedule can be enabled after its creation. Here the retry policy is set for 1 hour after the scheduled time, in which there are 5 retry attempts.



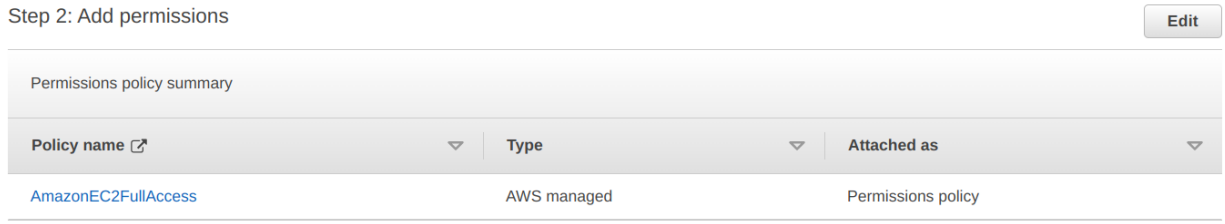


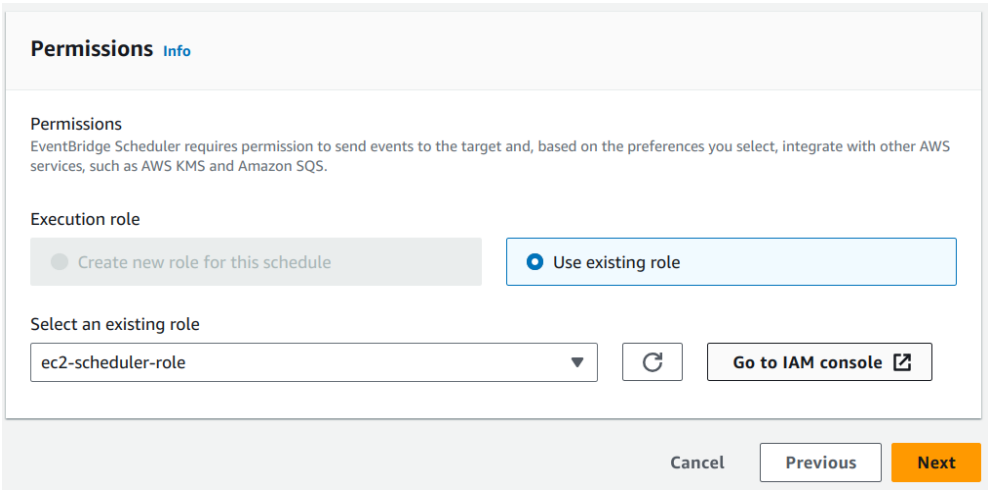
Creating an IAM role for the scheduler to perform actions on the EC2 instance.

Here, a custom trust policy is created which will allow the scheduler to assume the attached role.

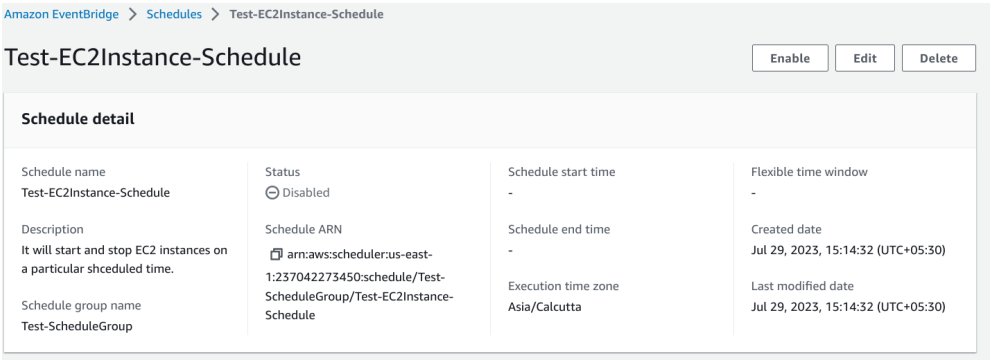


The role has permission policy attached for “EC2FullAccess”.

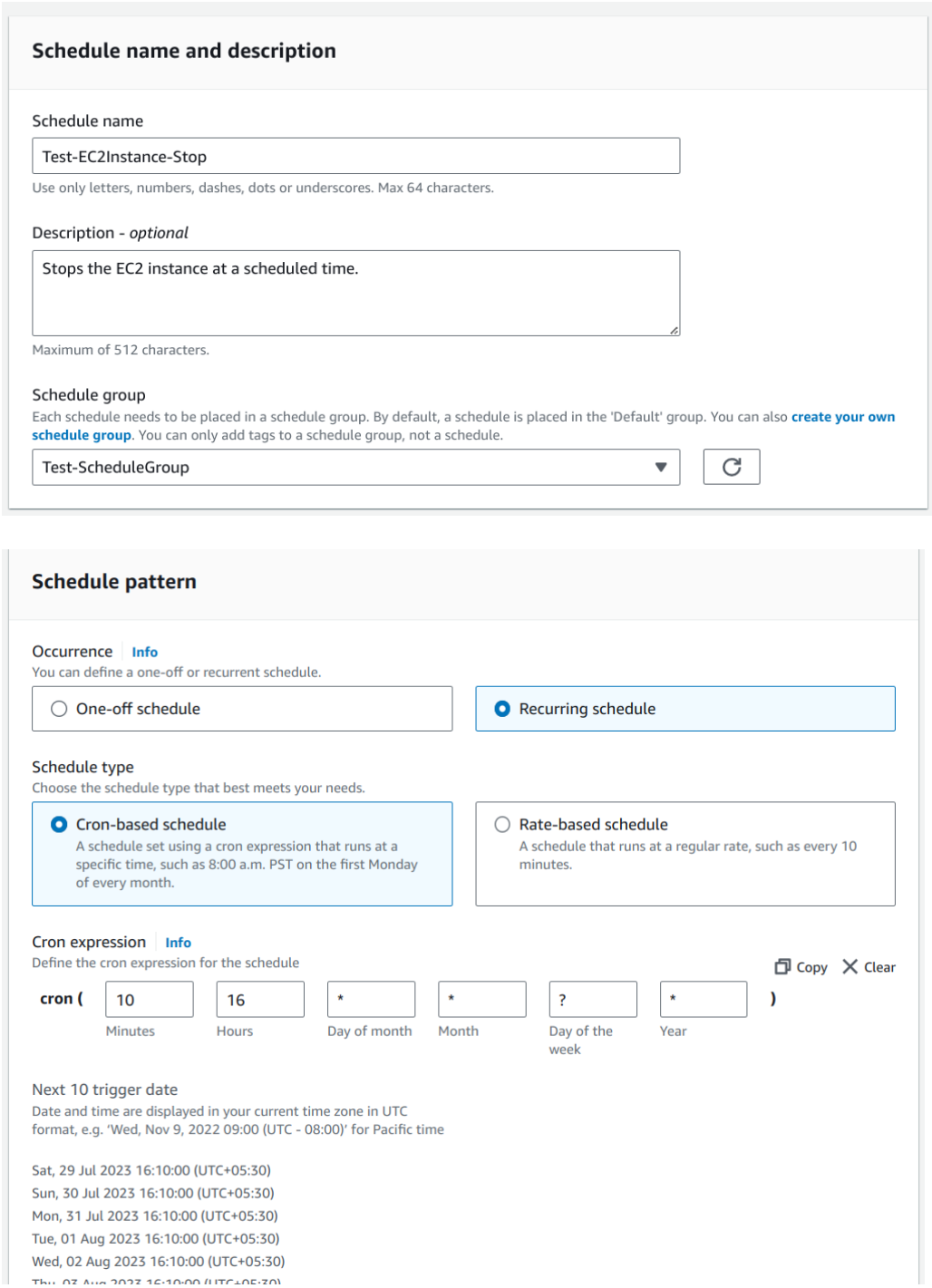




The schedule is created and is initially disabled.



Step 3 - Create a schedule for stopping the instance at a scheduled time.



Schedule name and description

Schedule name

Test-EC2Instance-Stop

Use only letters, numbers, dashes, dots or underscores. Max 64 characters.

Description - *optional*

Stops the EC2 instance at a scheduled time.

Maximum of 512 characters.

Schedule group

Each schedule needs to be placed in a schedule group. By default, a schedule is placed in the 'Default' group. You can also **create your own schedule group**. You can only add tags to a schedule group, not a schedule.

Test-ScheduleGroup

Schedule pattern

Occurrence Info

You can define a one-off or recurrent schedule.

One-off schedule

Recurring schedule

Schedule type

Choose the schedule type that best meets your needs.

Cron-based schedule

A schedule set using a cron expression that runs at a specific time, such as 8:00 a.m. PST on the first Monday of every month.

Rate-based schedule

A schedule that runs at a regular rate, such as every 10 minutes.

Cron expression Info

Define the cron expression for the schedule

Copy Clear

cron (10 16 * * ? *)

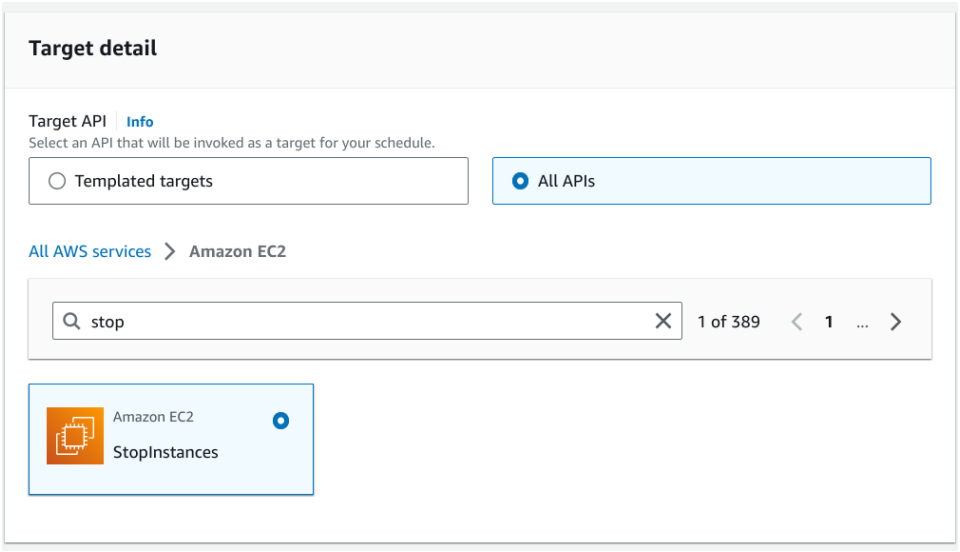
Minutes Hours Day of month Month Day of the week Year

Next 10 trigger date

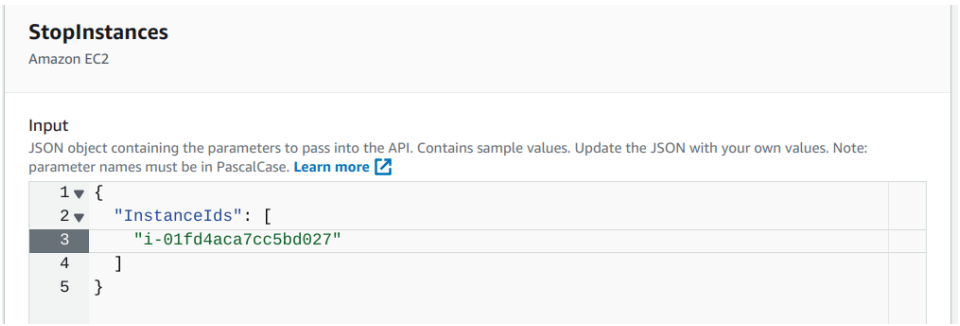
Date and time are displayed in your current time zone in UTC format, e.g. 'Wed, Nov 9, 2022 09:00 (UTC - 08:00)' for Pacific time

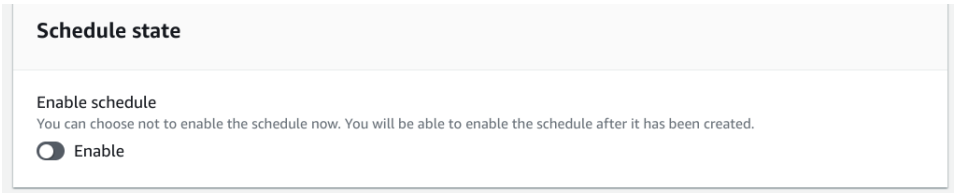
Sat, 29 Jul 2023 16:10:00 (UTC+05:30)
Sun, 30 Jul 2023 16:10:00 (UTC+05:30)
Mon, 31 Jul 2023 16:10:00 (UTC+05:30)
Tue, 01 Aug 2023 16:10:00 (UTC+05:30)
Wed, 02 Aug 2023 16:10:00 (UTC+05:30)

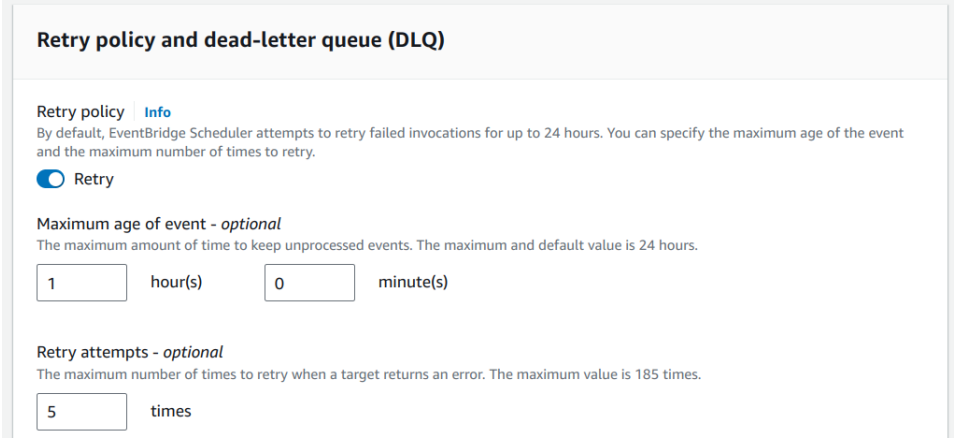
The target API call is to stop the instance.

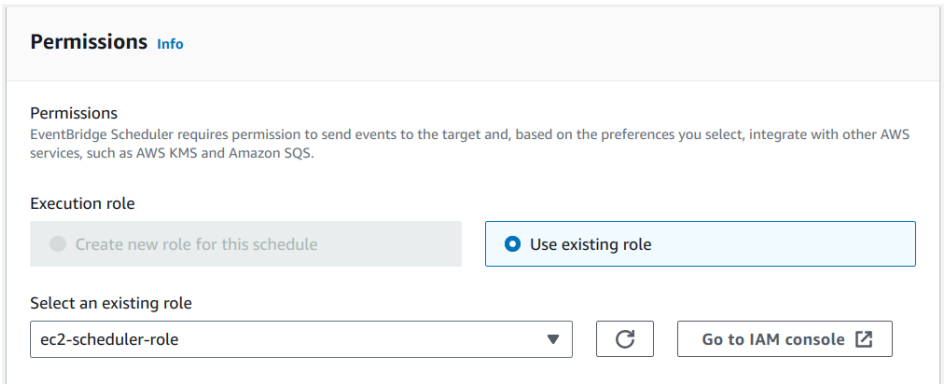


Enter the instance id of the EC2 instance.





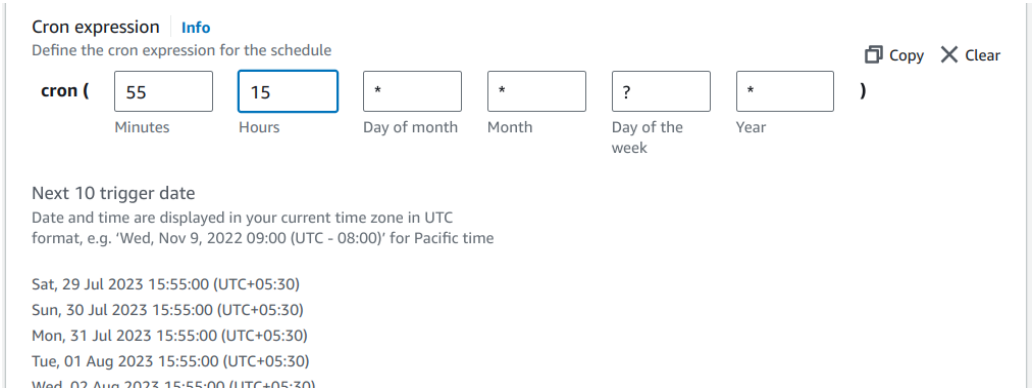




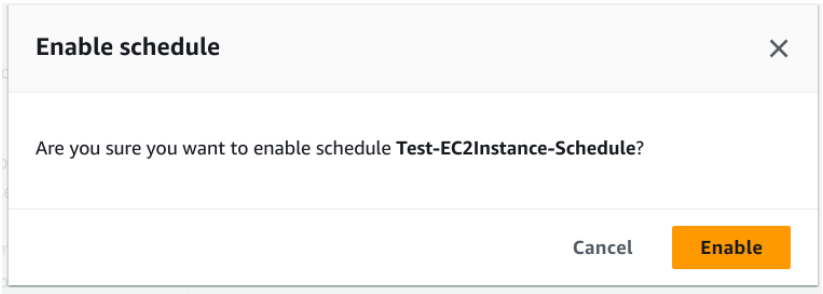
Step 4 - Check if both the schedulers are working as intended.

Stop the instance manually and check if it is started by the scheduler.

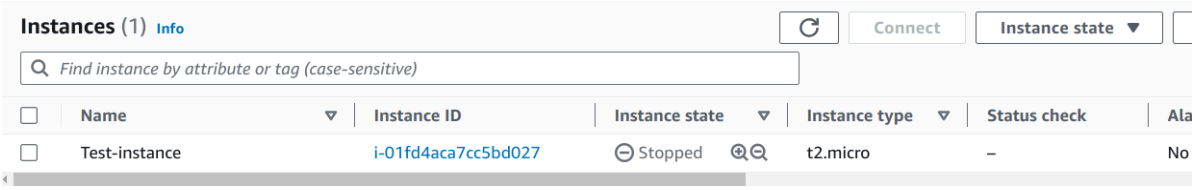
Change the cron expression of the start schedule.

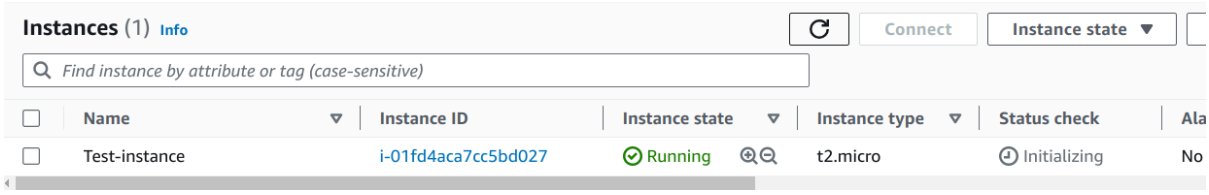


Enable the schedule as it was not enabled when it was created.

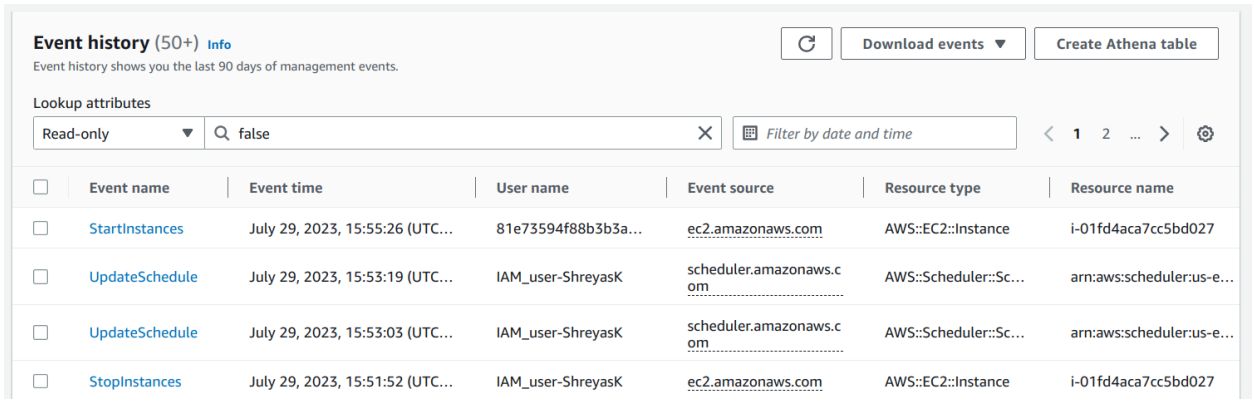


The instance is currently in stopped state.



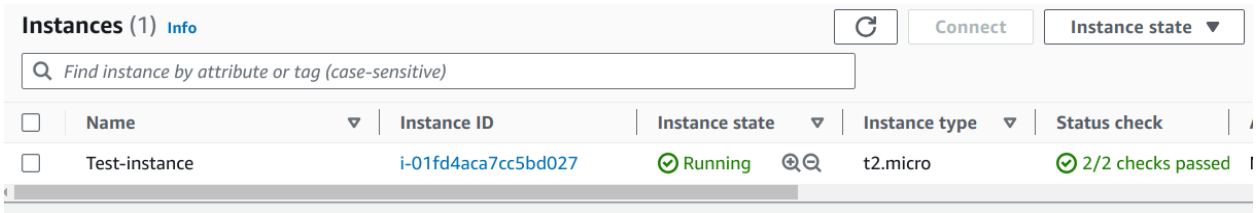


CloudTrail event history shows that the instance was started at 15:55 by the event scheduler.

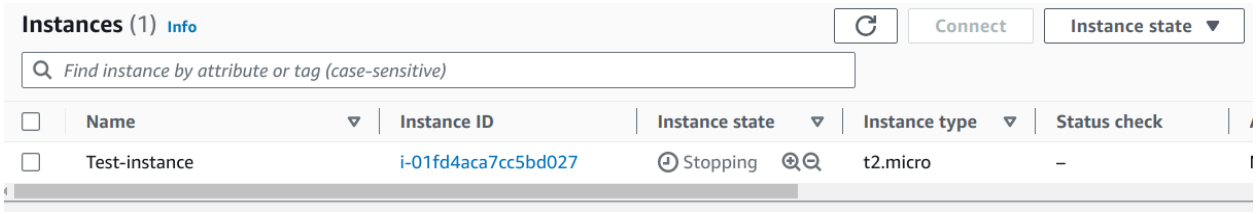


Now checking if the instance stops again at the scheduled time of 16:10.

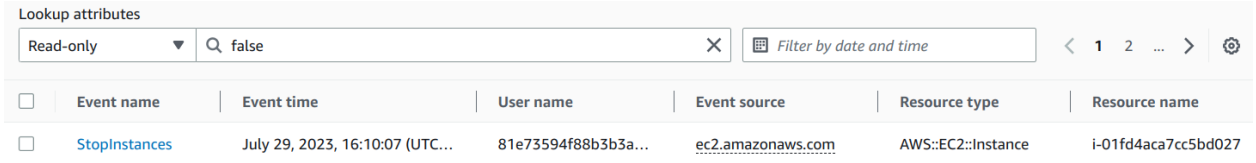
Currently it is in running state.



The instance turns to the stopping state at the scheduled time i.e. 16:10 hours.

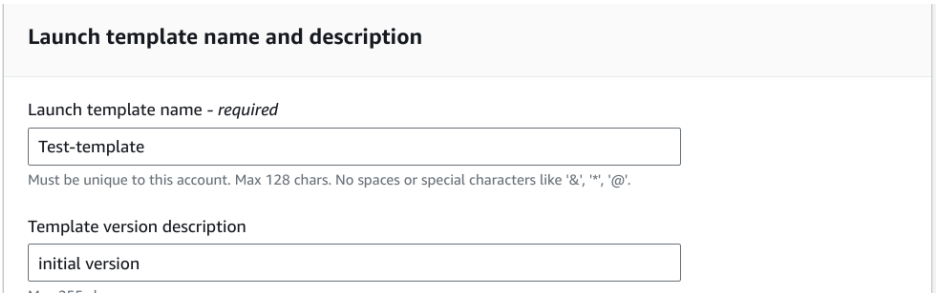


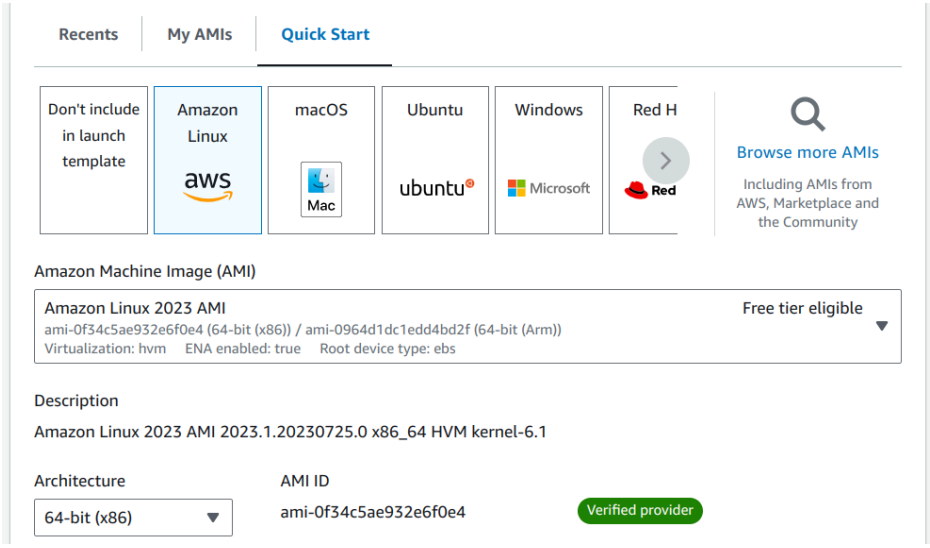
CloudTrail event history shows that the instance was stopped at 16:10 hours by the scheduler.

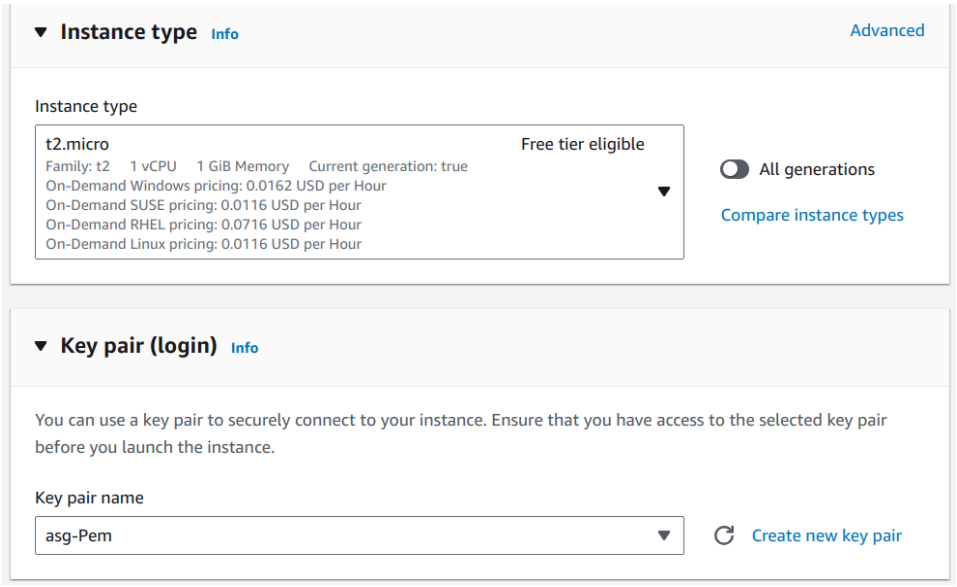


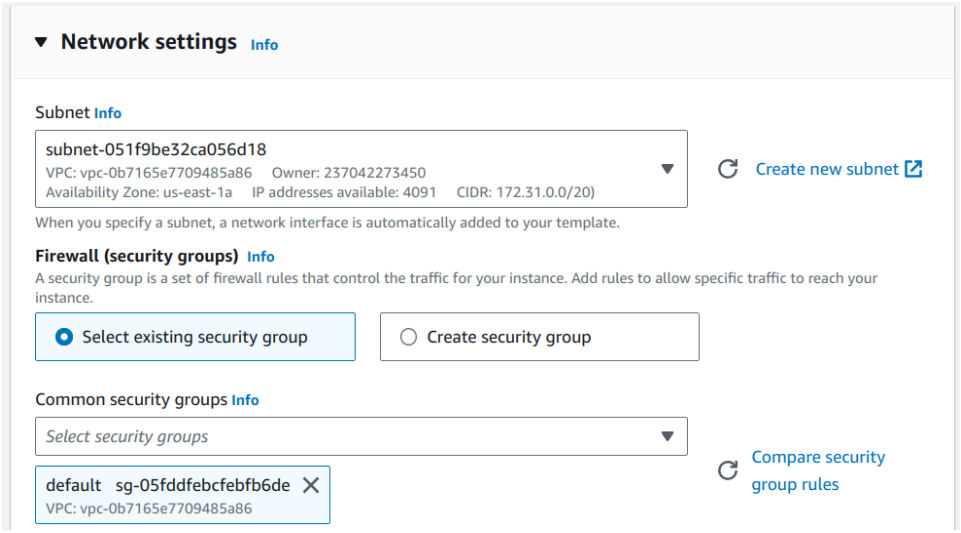
Assignment 4 - Create a SNS alert to send notification if instances in an ASG are manually terminated.

Step 1 - Create a launch template.

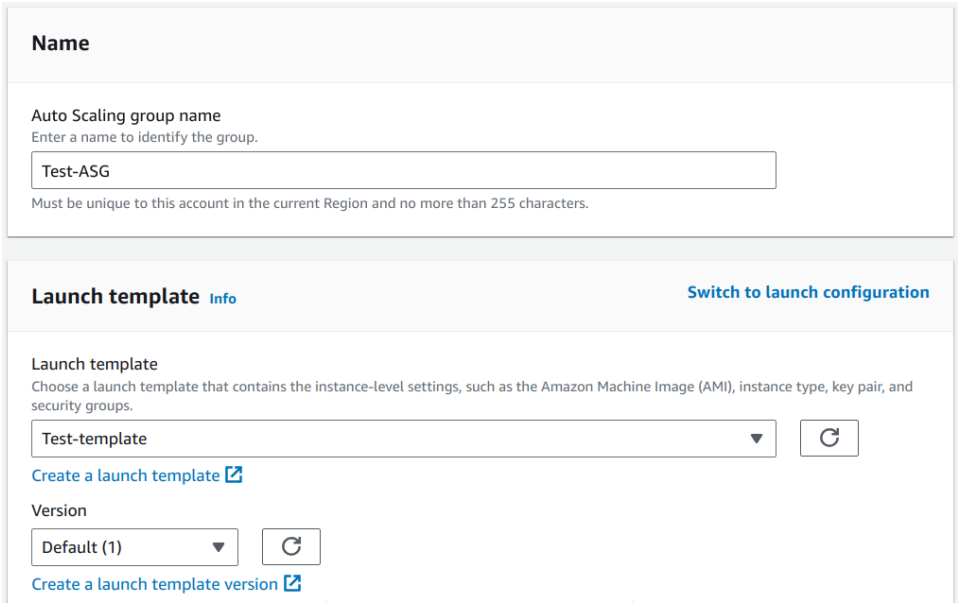


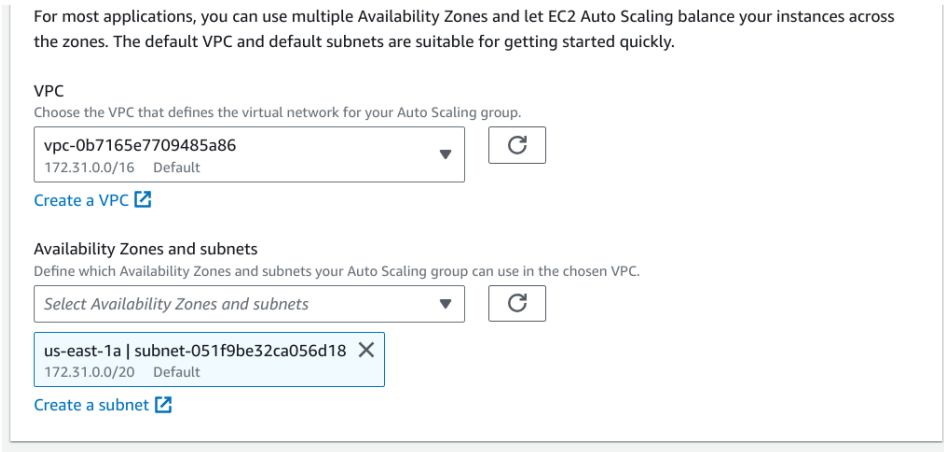


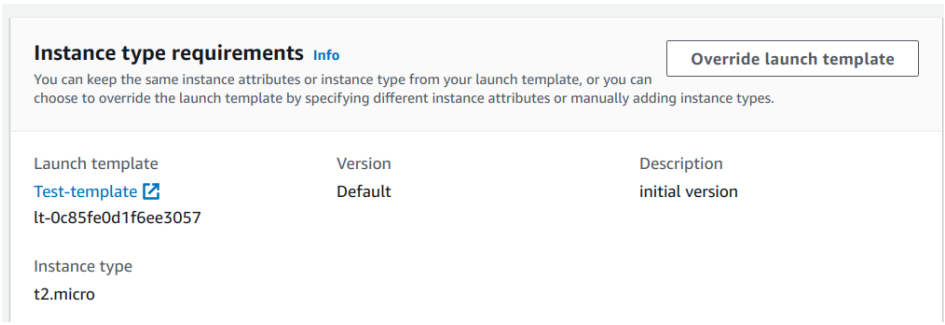


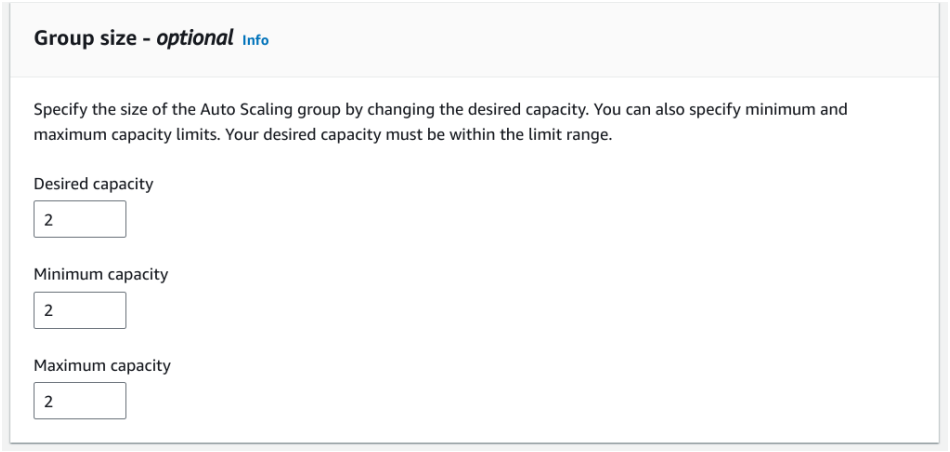


Step 2 - Create an Auto Scaling Group.

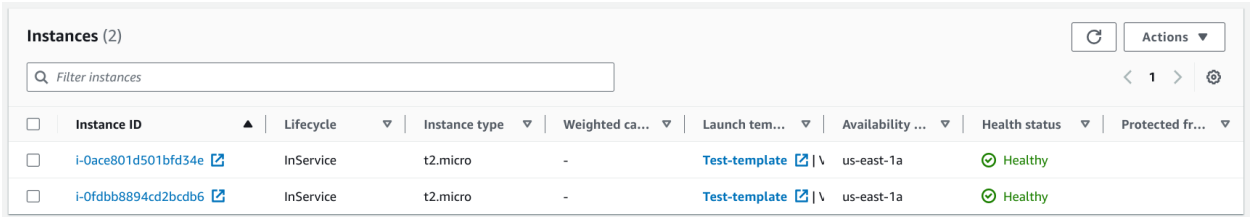






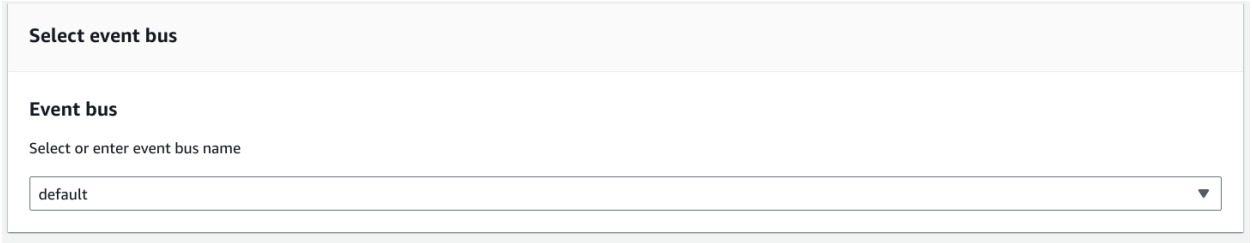


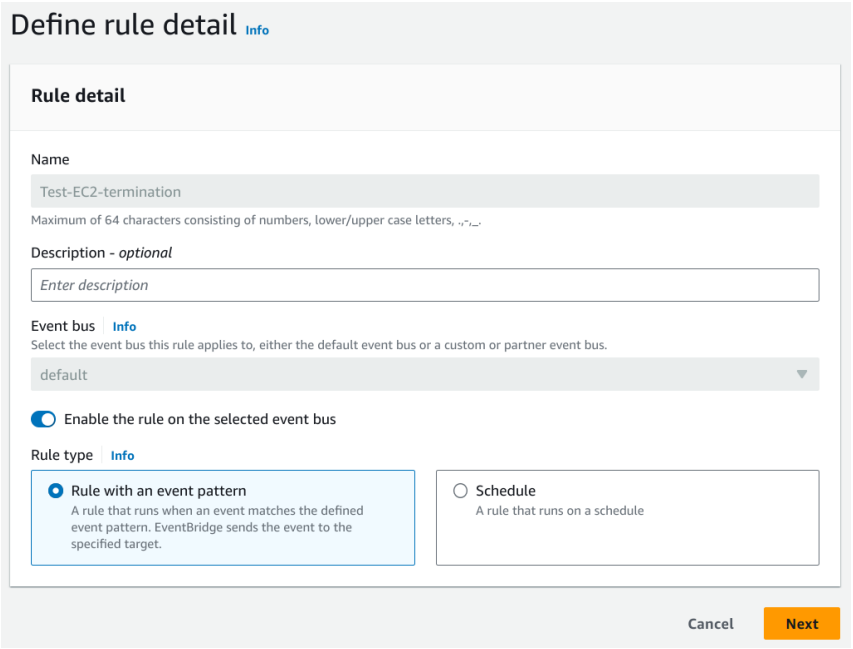
The capacity is updated and two instances are launched.



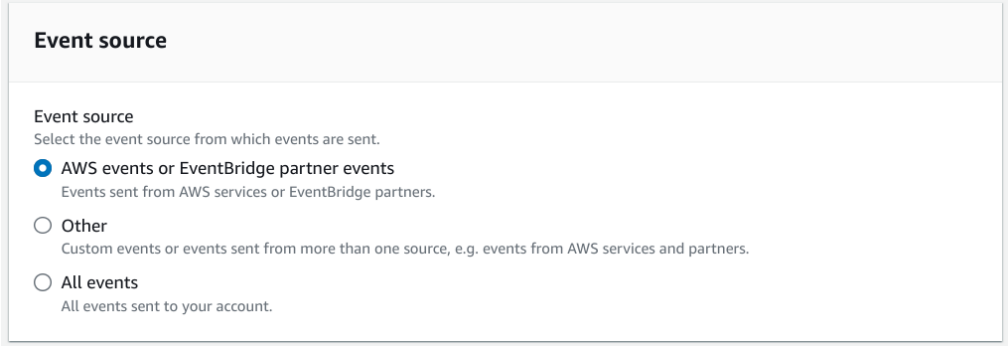
Step 3 - Create a rule.

Select the default event bus.

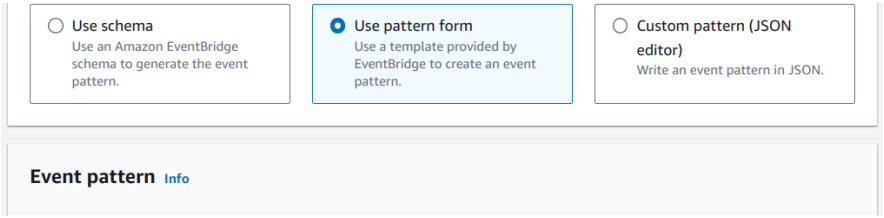


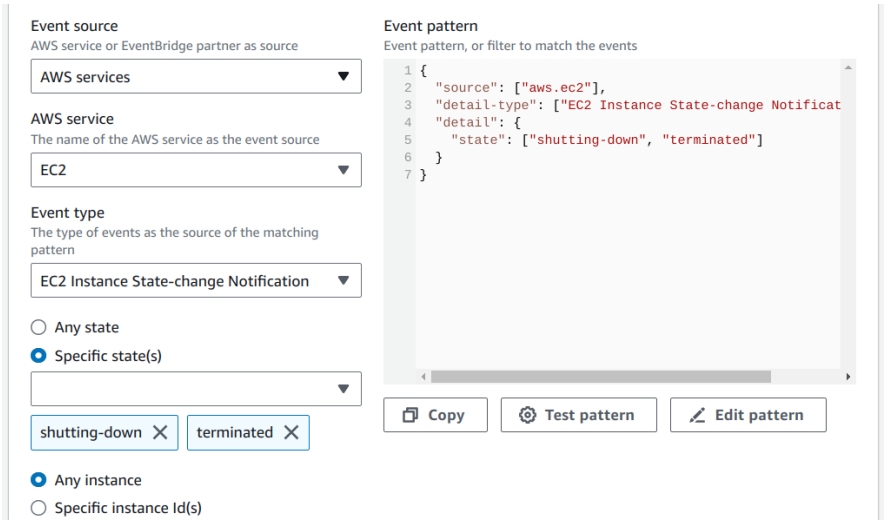


The event source is events sent from AWS services.

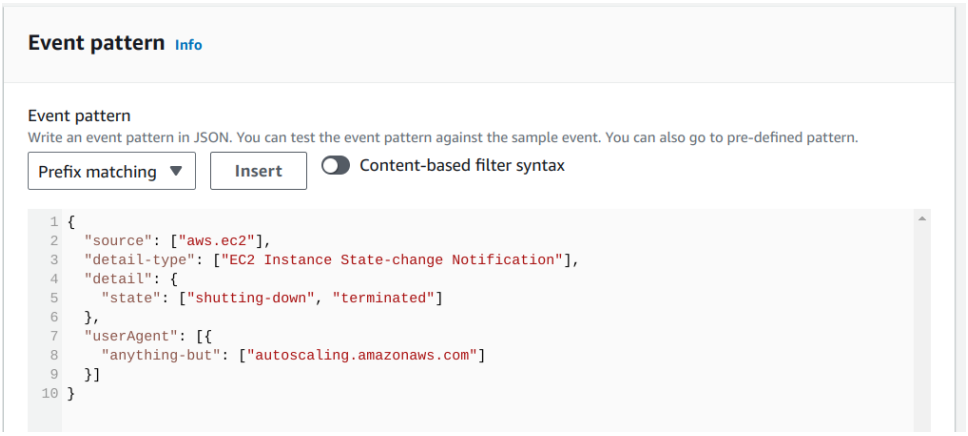


The event pattern will catch the events which match the pattern. Here it will catch EC2 instance manual termination events (source EC2 console).

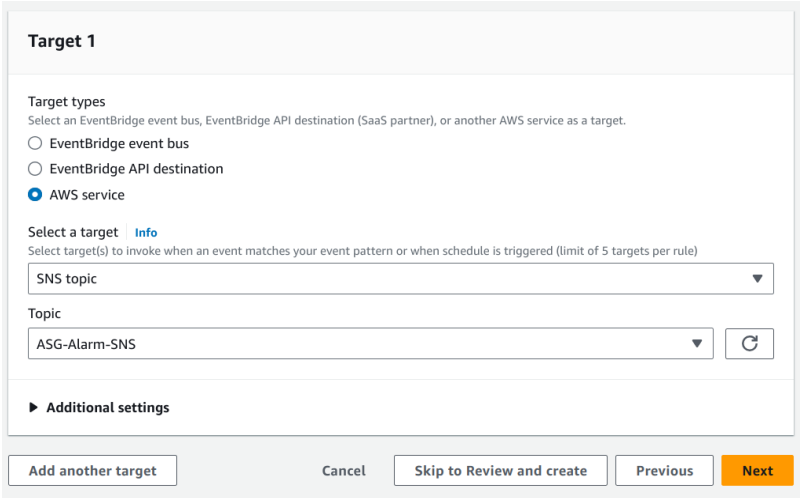




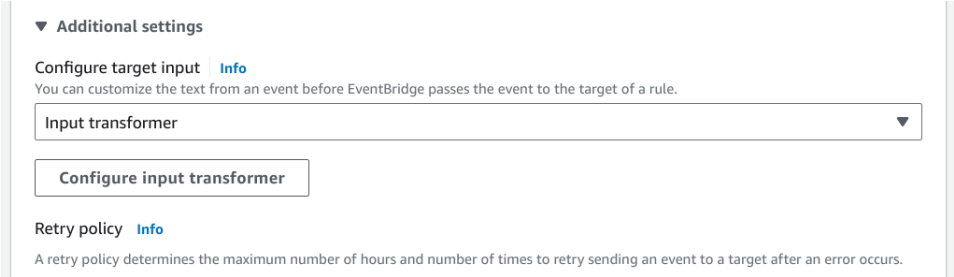
Add a "anything-but matching" pattern to catch events for only manual termination of the instances not by auto scaling group.



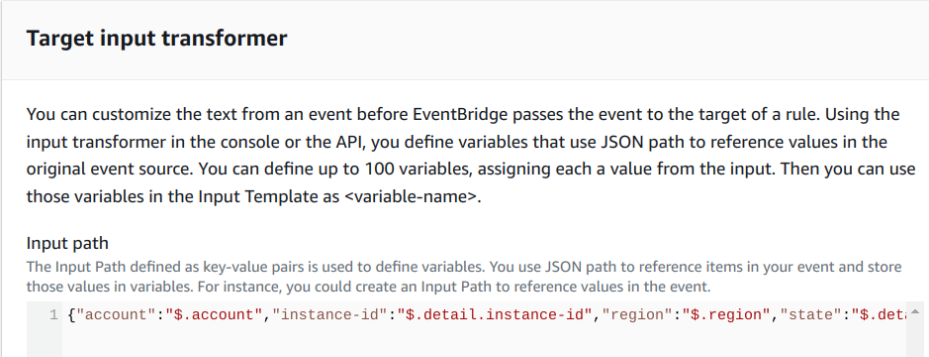
The rule will send an alert through the SNS topic for the event.



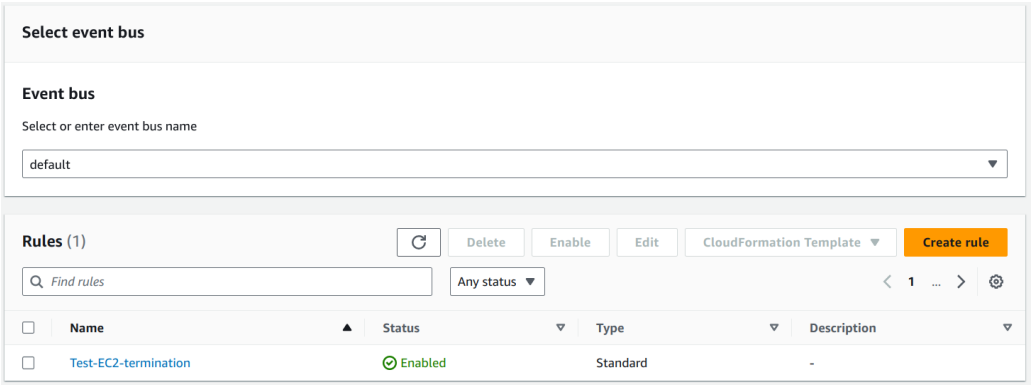
Select Input transformer, it will send a formatted message through the SNS notification. Then click on “Configure input transformer”.



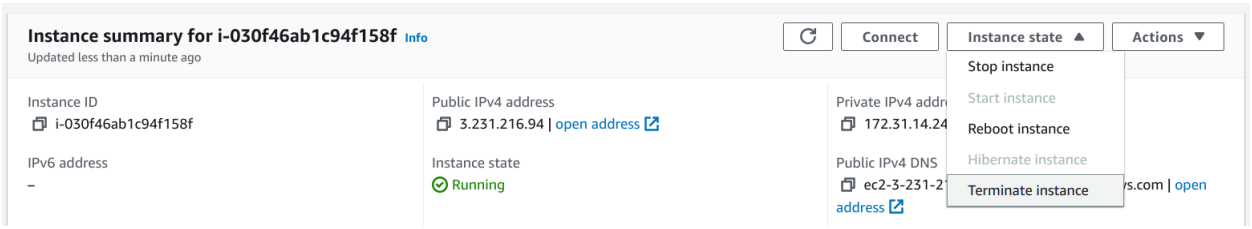
In the input path, create variables that will be used in the message. Here variables are created for account, instance id, region, etc.

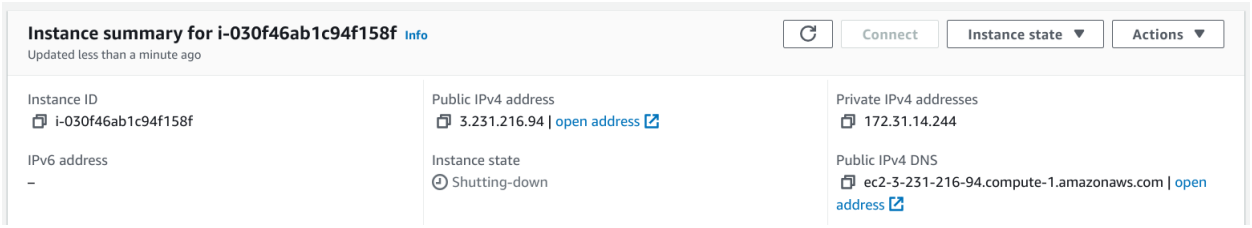


Define the message in the “template” section. Here it is visible that the variables are used in the message.

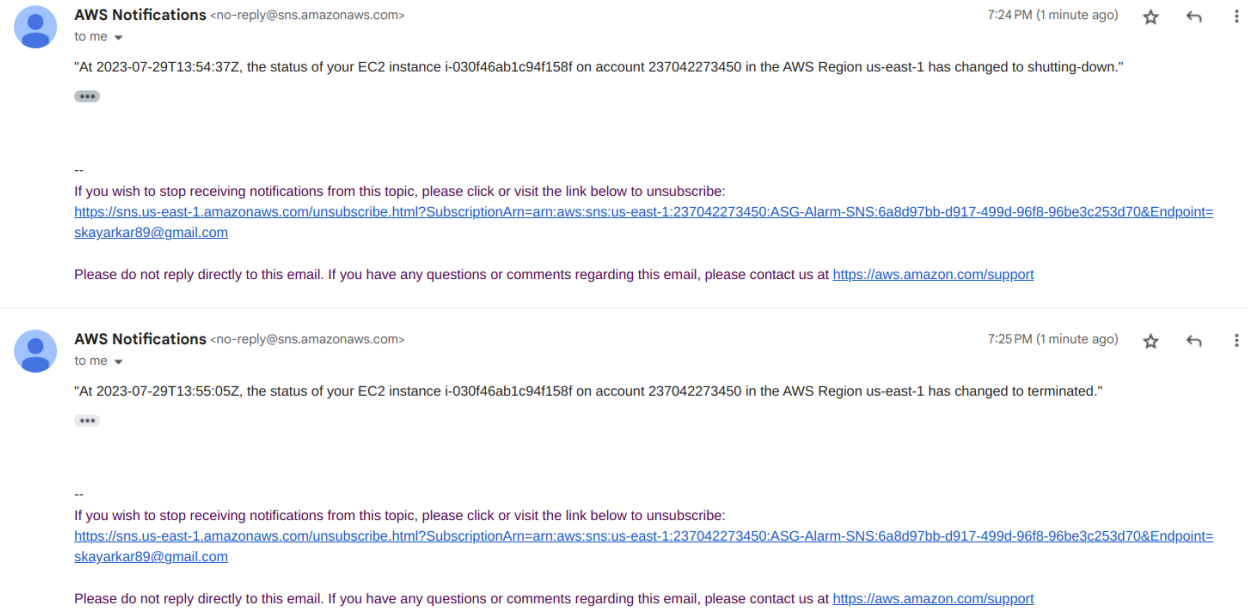


Step 5 - Check if the rule sends an alert, manually terminate one of the EC2 instances in the auto scaling group.





Received alert on for both the state changes, shutting down and terminated.



Assignment 5 - Launch EC2 instance using AWS CLI.

Step 1 - Login to the AWS CLI.

```
[cloudshell-user@ip-10-2-14-12 ~]$ aws sts get-caller-identity
{
    "UserId": "AIDATOMGSBSVAAKATBQ3F",
    "Account": "237042273450",
    "Arn": "arn:aws:iam::237042273450:user/IAM_user-ShreyasK"
}
[cloudshell-user@ip-10-2-14-12 ~]$
```

Step 2 - Create a key pair for the instance.

This command creates a new keypair “linuxKeyPair” and saves it to a file “linuxKeyPair.pem”.

```
[cloudshell-user@ip-10-4-164-203 ~]$ aws ec2 create-key-pair --key-name linuxKeyPair --query 'KeyMaterial' --output text > linuxKeyPair.pem
[cloudshell-user@ip-10-4-164-203 ~]$ ls
ec2-sk-SH.pem  linuxKeyPair.pem
[cloudshell-user@ip-10-4-164-203 ~]$ aws ec2 describe-key-pairs --key-name linuxKeyPair
{
    "KeyPairs": [
        {
            "KeyPairId": "key-0ecc1876ca4134ac0",
            "KeyFingerprint": "d9:e1:f7:f1:d0:c6:3e:97:cb:9c:f6:7d:52:b2:fd:1b:02:86:b2:1c",
            "KeyName": "linuxKeyPair",
            "KeyType": "rsa",
            "Tags": [],
            "CreateTime": "2023-07-29T16:48:34.053000+00:00"
        }
    ]
}
```

Step 3 - Create a security group for the instance.

This command creates a security group “Instance-SG” in the default VPC.

```
[cloudshell-user@ip-10-4-164-203 ~]$ aws ec2 create-security-group --group-name Instance-SG --description "Allows SSH on port 22" --vpc-id vpc-0b7165e7709485a86
{
    "GroupId": "sg-080377966684077b1"
}
[cloudshell-user@ip-10-4-164-203 ~]$ aws ec2 describe-security-groups --group-ids sg-080377966684077b1
{
    "SecurityGroups": [
        {
            "Description": "Allows SSH on port 22",
            "GroupName": "Instance-SG",
            "IpPermissions": [],
            "OwnerId": "237042273450",
            "GroupId": "sg-080377966684077b1",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-0b7165e7709485a86"
        }
    ]
}
```

This command adds an incoming traffic rule to allow SSH on port 22 from anywhere.

```
[cloudshell-user@ip-10-4-38-209 ~]$ aws ec2 authorize-security-group-ingress --group-id sg-080377966684077b1 --protocol tcp --port 22 --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09171e7919ad89037",
            "GroupId": "sg-080377966684077b1",
            "GroupOwnerId": "237042273450",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
[cloudshell-user@ip-10-4-38-209 ~]$
```

```
[cloudshell-user@ip-10-4-38-209 ~]$ aws ec2 describe-security-groups --group-ids sg-080377966684077b1
{
    "SecurityGroups": [
        {
            "Description": "Allows SSH on port 22",
            "GroupName": "Instance-SG",
            "IpPermissions": [
                {
                    "FromPort": 22,
                    "IpProtocol": "tcp",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "ToPort": 22,
                    "UserIdGroupPairs": []
                }
            ],
            "OwnerId": "237042273450",
            "GroupId": "sg-080377966684077b1",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-0b7165e7709485a86"
        }
    ]
}
```

Step 4 - Launch an EC2 instance.

This command launches an EC2 instance with AMI of amazon linux 2023, instance type of t2.micro, attaching the key pair and security group created earlier in the subnet in AZ “us-east-1a” of the default VPC.

```
[cloudshell-user@ip-10-6-186-151 ~]$ aws ec2 run-instances --image-id ami-0f34c5ae932e6f0e4 --count 1 --instance-type t2.micro
{
    "Groups": []
```

```
--key-name linuxKeyPair --security-group-ids sg-080377966684077b1 --subnet-id subnet-051f9be32ca056d18
```

```
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0f34c5ae932e6f0e4",
            "InstanceId": "i-02a791218617c108d",
            "InstanceType": "t2.micro",
            "KeyName": "linuxKeyPair",
            "LaunchTime": "2023-07-29T17:06:10+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-3-207.ec2.internal",
            "PrivateIpAddress": "172.31.3.207",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-051f9be32ca056d18",
            "VpcId": "vpc-0b7165e7709485a86",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "107d962e-2b76-49e2-ba95-5939e93f0cf0",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
```

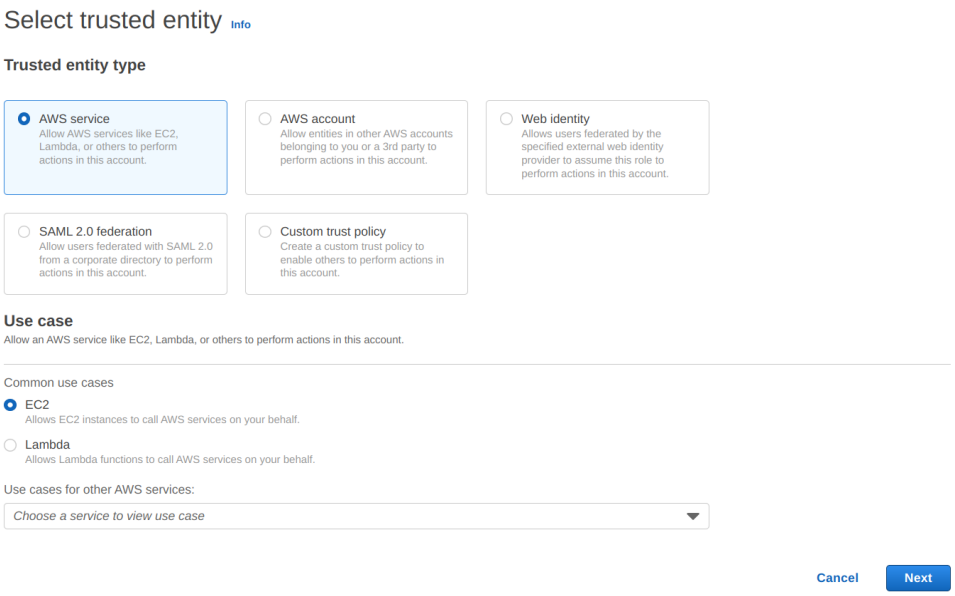
```
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2023-07-29T17:06:10+00:00",
                        "AttachmentId": "eni-attach-0ff9a2bfb051976a9",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching",
                        "NetworkCardIndex": 0
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupName": "Instance-SG",
                            "GroupId": "sg-080377966684077b1"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "02:cb:9e:4d:ab:21",
                    "NetworkInterfaceId": "eni-0a503802a899b45dd",
                    "OwnerId": "237042273450",
                    "PrivateDnsName": "ip-172-31-3-207.ec2.internal",
                    "PrivateIpAddress": "172.31.3.207",
                    "PrivateIpAddresses": [
                        {
                            "Primary": true,
                            "PrivateDnsName": "ip-172-31-3-207.ec2.internal",
                            "PrivateIpAddress": "172.31.3.207"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Status": "in-use",
                    "SubnetId": "subnet-051f9be32ca056d18",
                    "VpcId": "vpc-0b7165e7709485a86",
                    "InterfaceType": "interface"
                }
            ],
```

Step 5 - Connect to the instance.

Assignment 6 - Installing grafana and collecting EC2 metrics.

Step 1 - Create an IAM role with permissions to get and list CloudWatch metrics and describe instances. This role will be attached to the EC2 instance in which grafana will be configured, so that grafana can get metrics from AWS.

Select use case for EC2 as this role will be attached to the EC2 instance.



Create a policy, this policy allows to list metrics, get metric statistics, and get metric data from cloudwatch, and describe tags, instances and regions.

Policy editor

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowReadingMetricsFromCloudWatch",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:GetMetricData"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowReadingTagsInstancesRegionsFromEC2",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeTags",
                "ec2:DescribeInstances",
                "ec2:DescribeRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

Role name
Enter a meaningful name to identify this role.

GetMetrics_EC2Role

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Step 1: Select trusted entities

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Attached the previously created policy to the IAM role.

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
GetMetrics_Policy	Customer managed	Permissions policy

Step 2 - Launch an EC2 instance.

Name and tags Info

Name

Test-Grafana_Instance

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat | SUSE Li

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible
ami-0f34c5ae932e6f0e4 (64-bit (x86)) / ami-0964d1dc1edd4bd2f (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 AMI 2023.1.20230725.0 x86_64 HVM kernel-6.1

Architecture: 64-bit (x86)

AMI ID: ami-0f34c5ae932e6f0e4 — Verified provider

Instance type Info

Instance type

t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour

All generations

Compare instance types

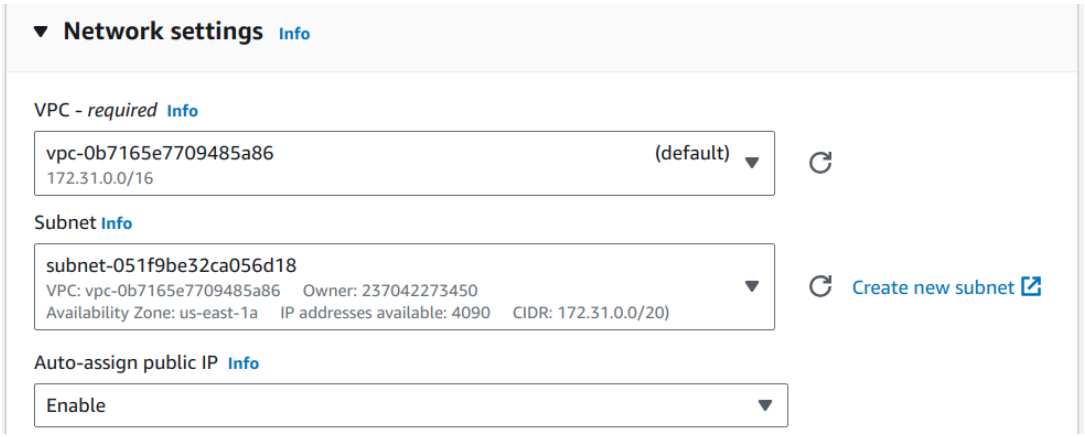
Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

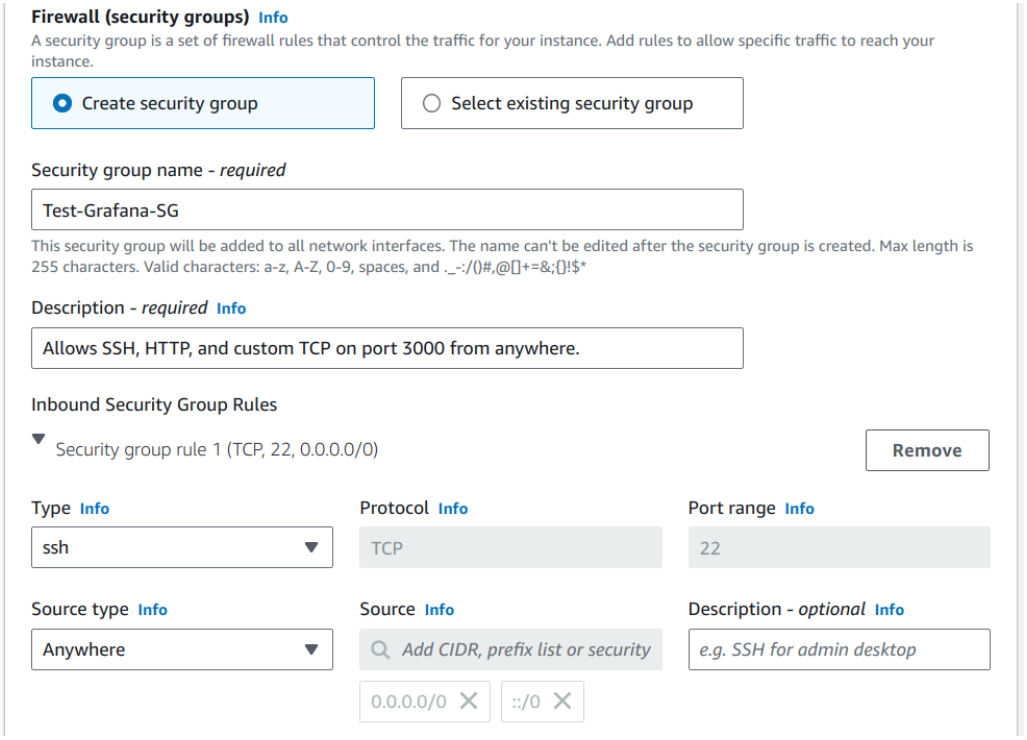
Key pair name - *required*

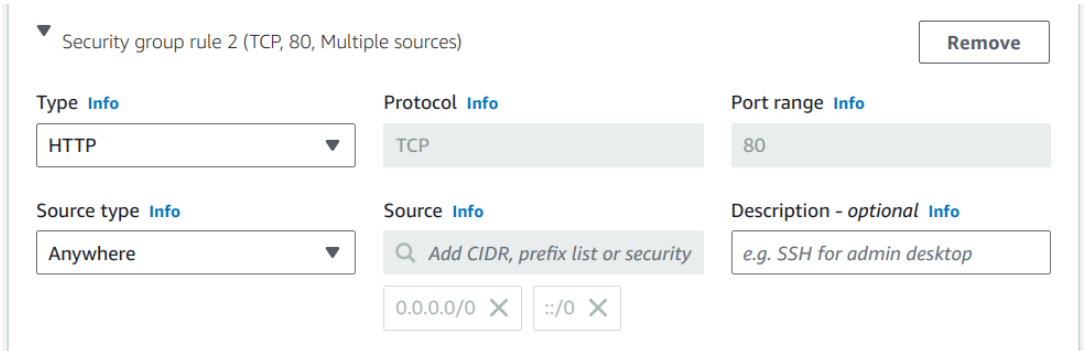
linuxKeyPair

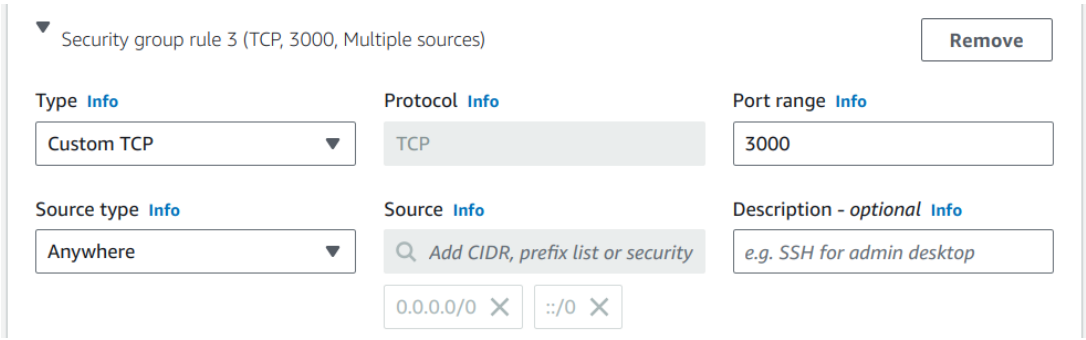
Create new key pair



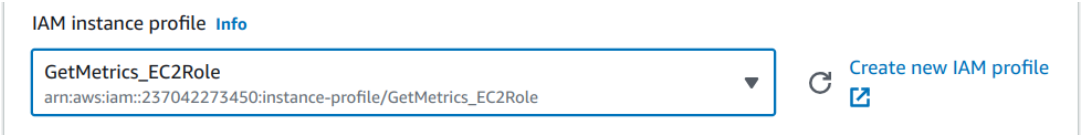
The security group allows incoming traffic for SSH, HTTP, and custom TCP on port 3000 from anywhere.







Attaching the role created in the previous step.



Step 3 - Install grafana in the EC2 instance.

Create a repository for grafana before installing it.

```
[root@ip-172-31-14-3 ec2-user]# vi /etc/yum.repos.d/grafana.repo
```

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
~
~
```

```
[root@ip-172-31-14-3 ec2-user]# yum install grafana -y
grafana
grafana
Importing GPG key 0x2CF3C0C6:
 Userid     : "Grafana Labs <engineering@grafana.com>"
 Fingerprint: 0E22 EB88 E39E 1227 7A77 60AE 9E43 9B10 2CF3 C0C6
 From       : https://packages.grafana.com/gpg.key
grafana
Last metadata expiration check: 0:00:08 ago on Sun Jul 30 10:33:30 2023.
```

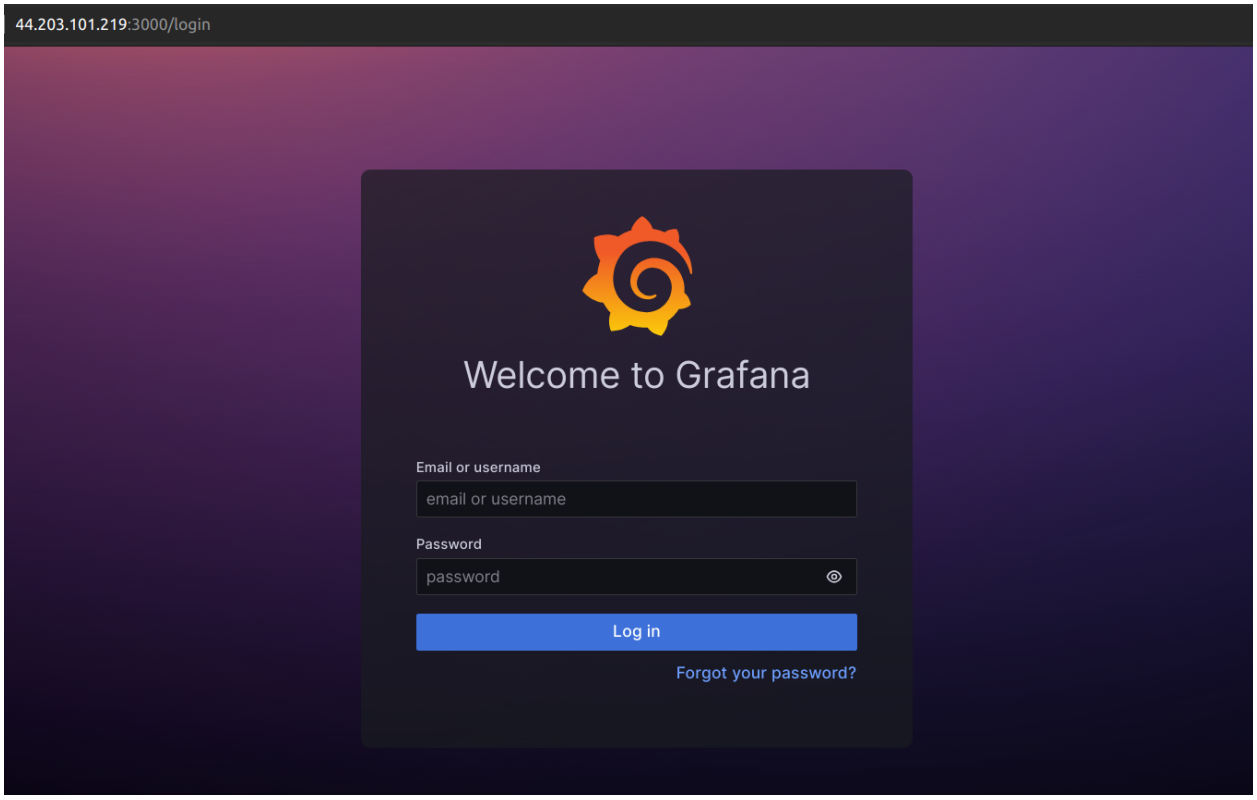
Start the grafana service.

```
[root@ip-172-31-14-3 ec2-user]# systemctl daemon-reload
[root@ip-172-31-14-3 ec2-user]# systemctl start grafana-server
[root@ip-172-31-14-3 ec2-user]#
[root@ip-172-31-14-3 ec2-user]# systemctl status grafana-server
● grafana-server.service - Grafana instance
     Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; disabled; preset: disabled)
     Active: active (running) since Sun 2023-07-30 10:44:51 UTC; 13s ago
       Docs: http://docs.grafana.org
   Main PID: 26686 (grafana)
      Tasks: 9 (limit: 1114)
     Memory: 147.7M
        CPU: 3.331s
     CGroup: /system.slice/grafana-server.service
             └─26686 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/var/r
```

Enable the grafana service so that it will start automatically whenever the instance is restarted.

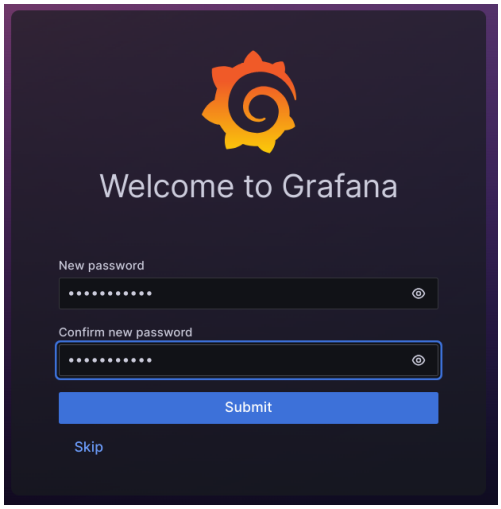
```
[root@ip-172-31-14-3 ec2-user]# systemctl enable grafana-server.service
Synchronizing state of grafana-server.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /usr/lib/systemd/system/grafana-server.service.
```

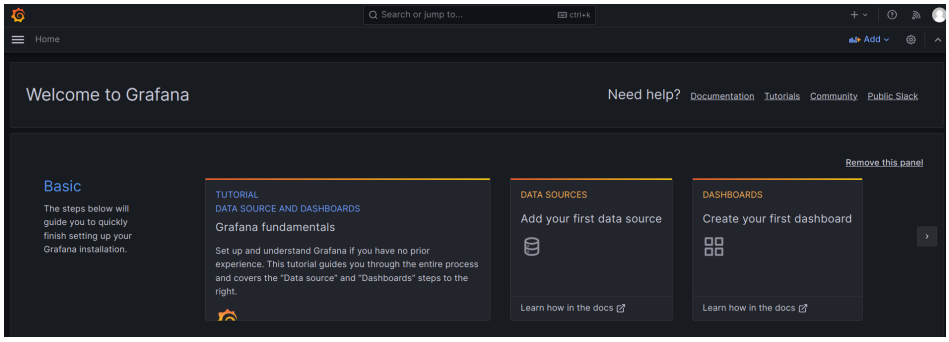
Enter the public IPv4 address of the instance with the custom port number 3000 - “44.203.101.219:3000”



Step 4 - Configure grafana to collect metrics from AWS Cloudwatch.

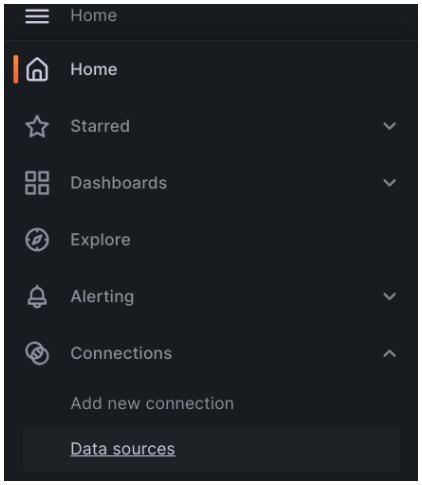
Login to grafana with default username and password (admin). Then reset the password.

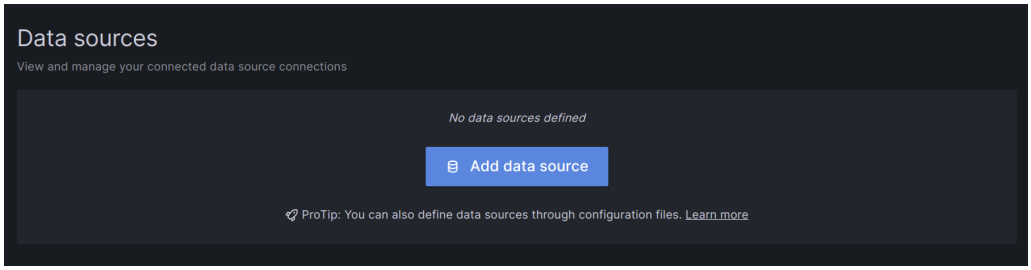


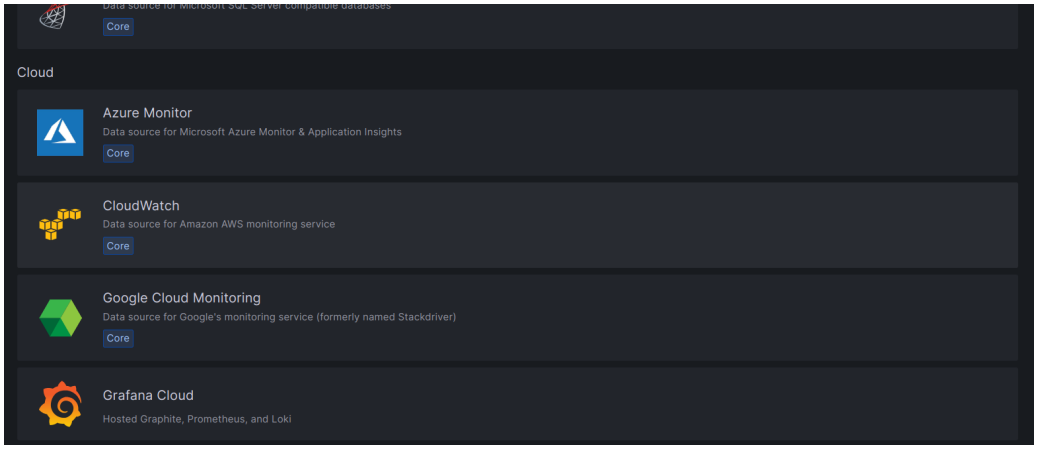


Step 5 - Connect grafana to AWS CloudWatch.

In the left navigation bar, go to the data sources section and choose CloudWatch as the data source.

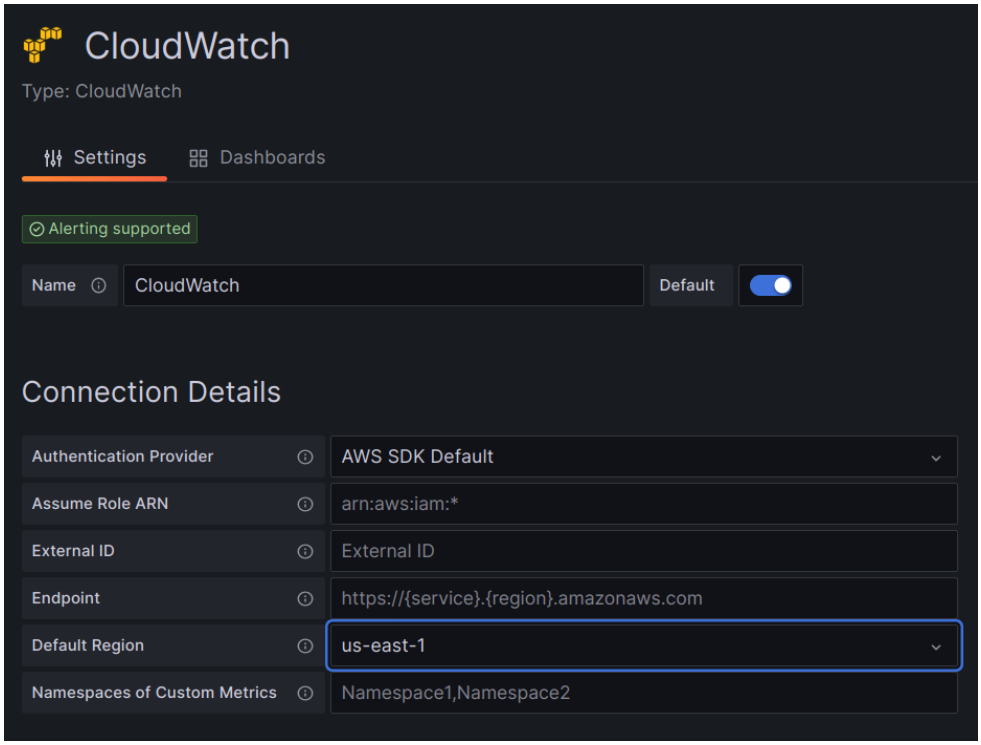






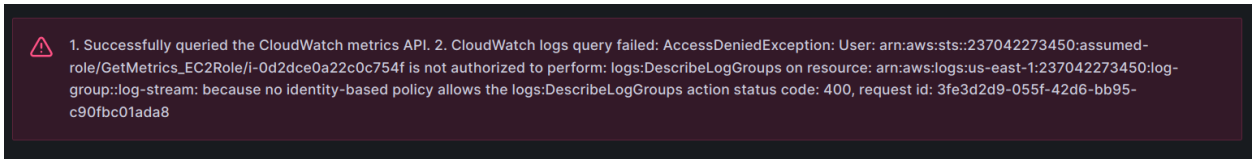
Step 6 - Configure the connection details.

Enter the default region. Here it is “us-east-1”.

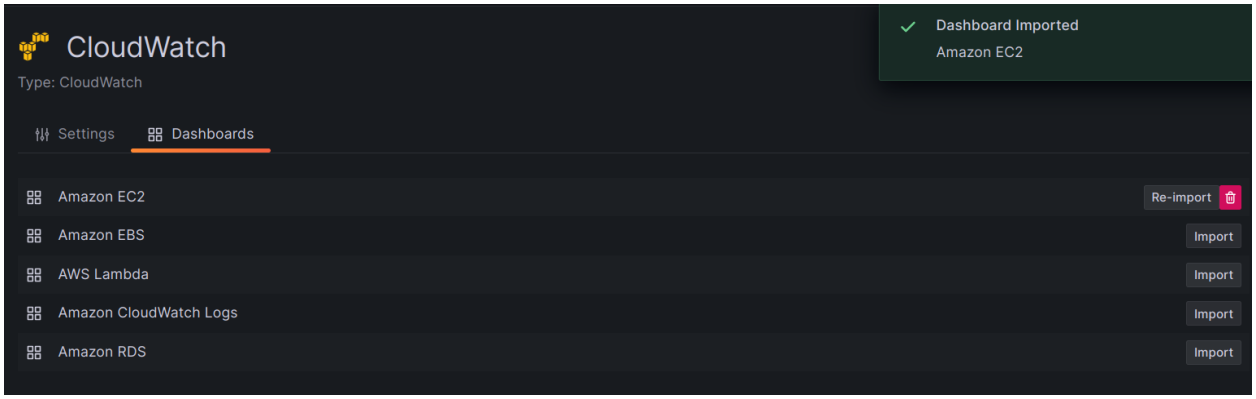


Click on “Test & Save”.

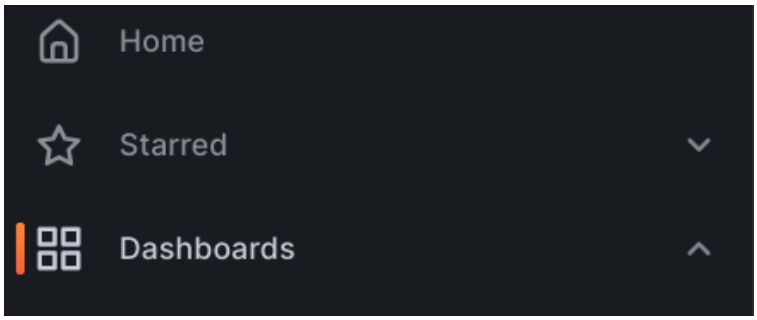
The message shows that metrics are successfully queried but shows error for querying the logs. This is because the IAM role attached to the EC2 instance in which grafana is install, has permissions for only to collect metrics but not the logs.



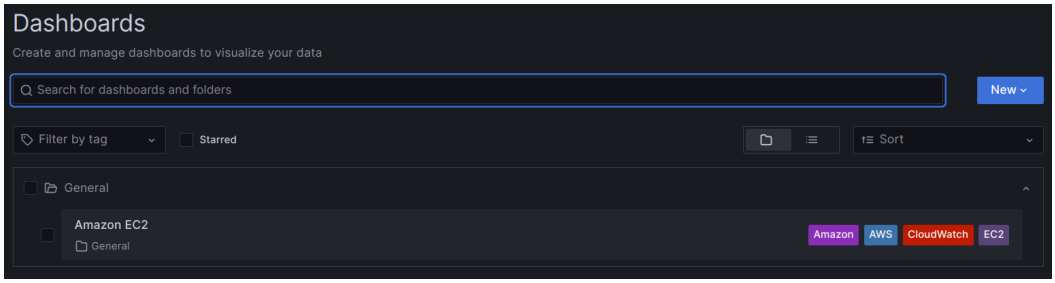
Go to the Dashboards section. Click on import for the service for the which the metrics are required. Here only EC2 metrics are imported.



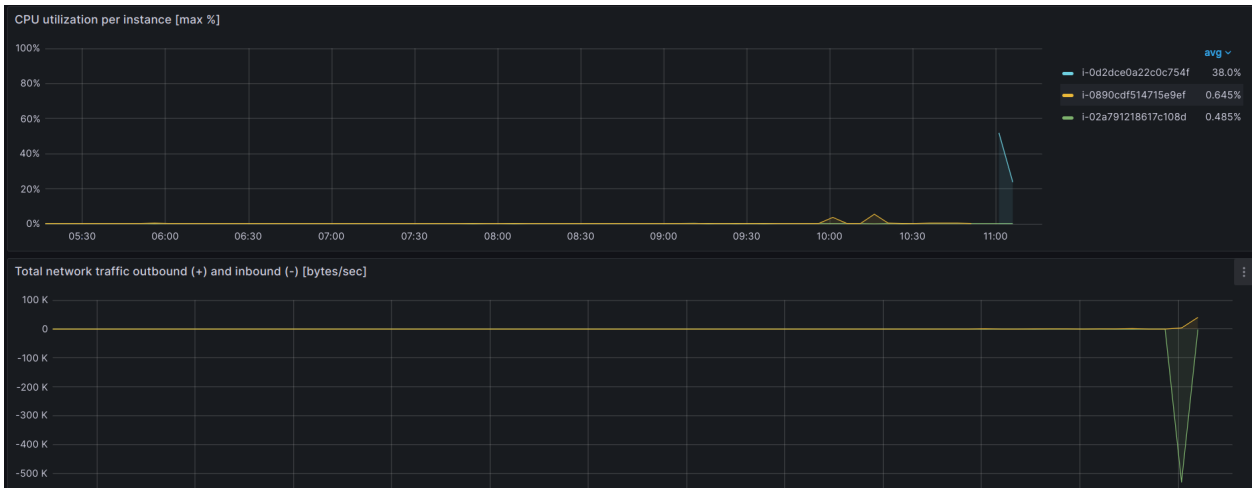
Click on the Dashboard button in the left navigation bar.



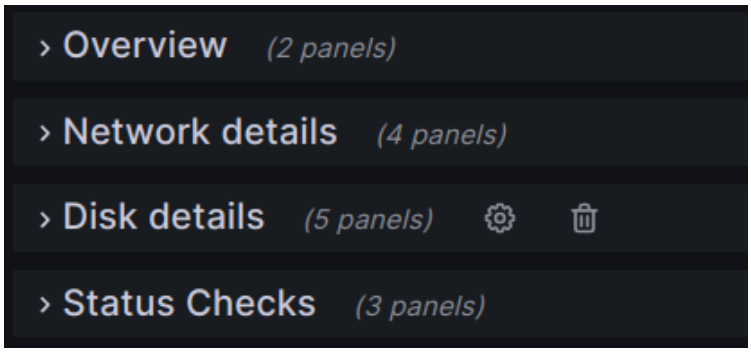
Click on “General” and select the service name. Here EC2.



Here the graphs for CPU utilization and other metrics from the EC2 instances running in the default selected region are displayed.



Here we can see the default metrics published by EC2 instances.



Here we can configure the graphs according to the requirements.