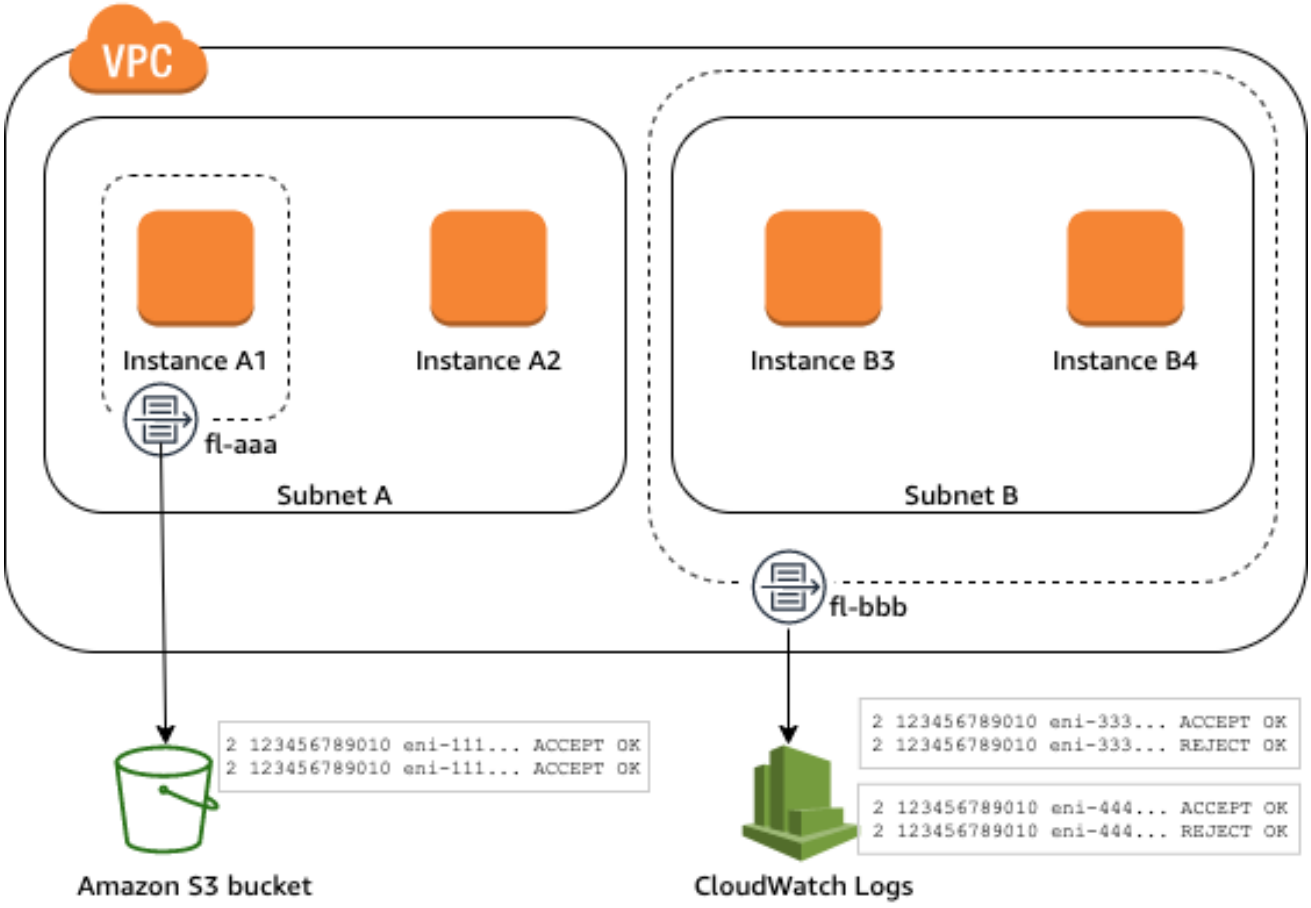
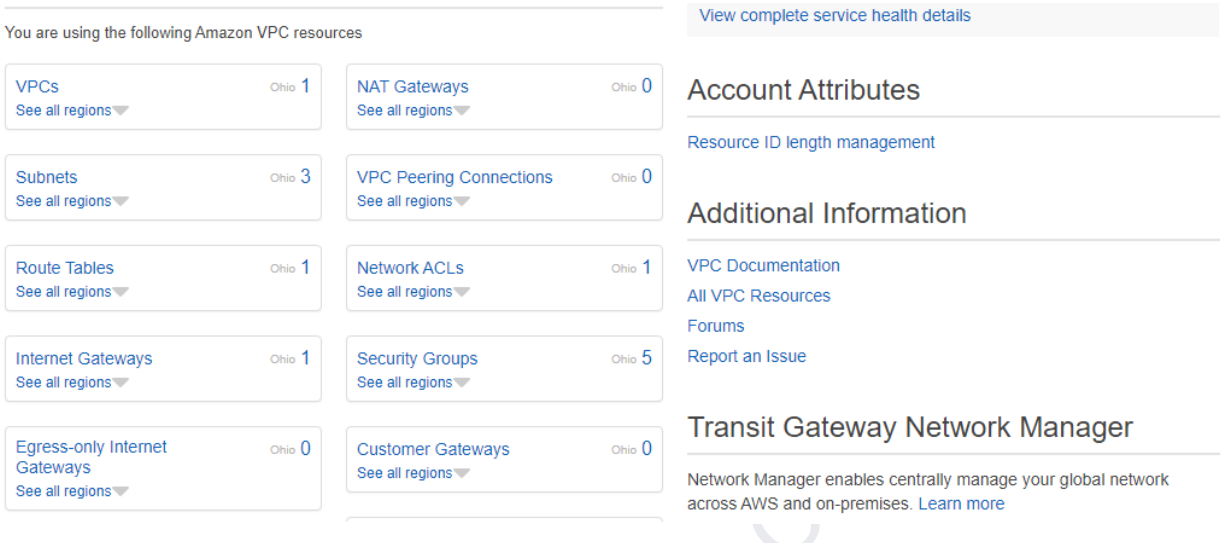
<u>#12 Task:</u> Create VPC Flow Logs, S3 Access Logs, API Gw Logs, Lambda Logs, CloudTrail logs and Install CloudWatch Monitoring Agents on EC2 to capture data for logs. All these Logs can be used further to configure a Cloud Vulnerability Scanning System to maintain a robust & secured Infra Environment.

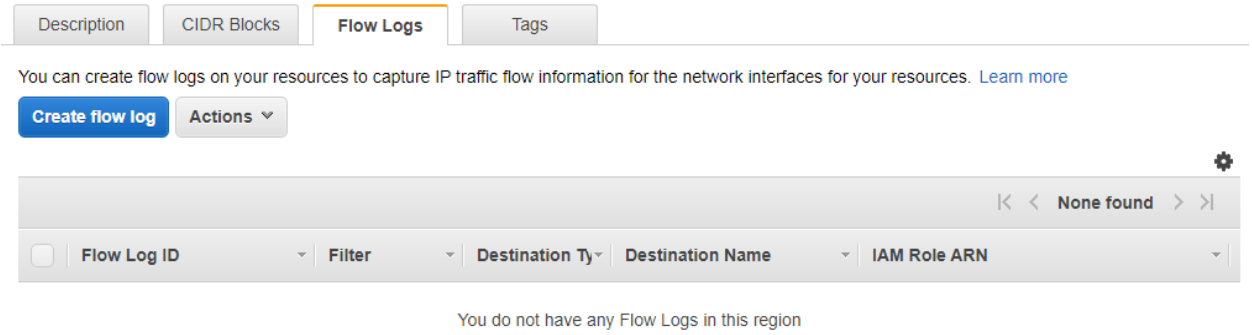
1. Setup VPC Flow Logs To CloudWatch Log Group Or S3 Bucket.



1. Go to your VPC Dashboard on the AWS console

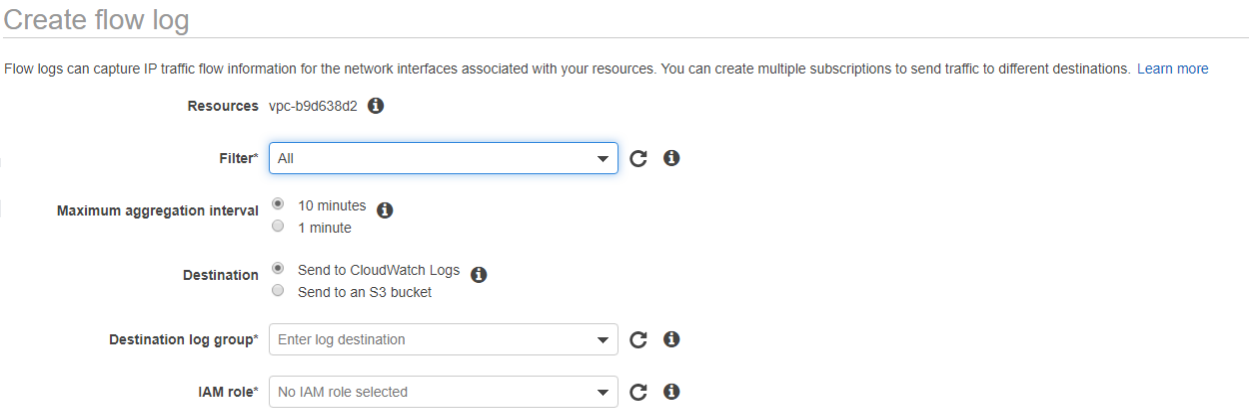


2. Click on Flow logs



Select Flow logs in the VPC Dashboard

3. Click on Create Flow logs

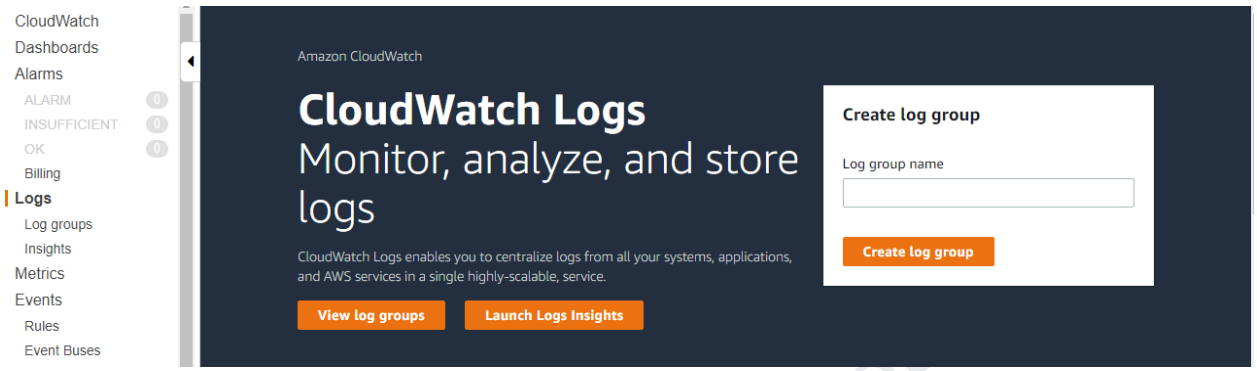


CLOUDSPIKES
Multi-cloud solutions

Create Flow log

If you don't have any Destination log group or IAM Role create one from scratch.

- To Create a Destination log group, Go to the Cloudwatch dashboard on your VPC Console

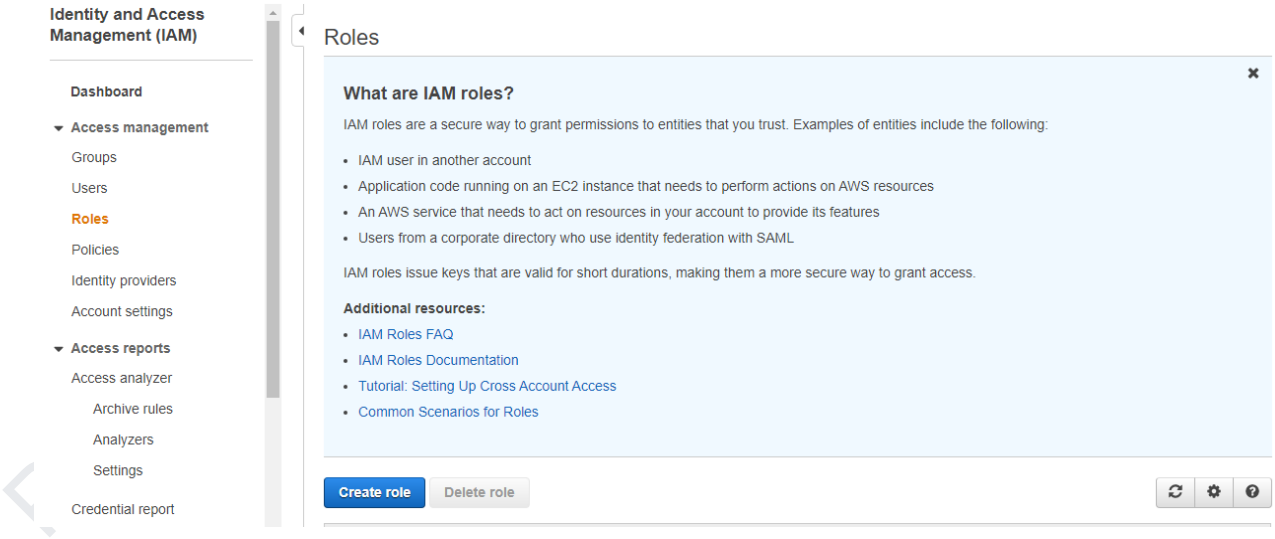


Click on Logs and enter a log group name, click on create log

4. Return back to the VPC Flowlogs dashboard and enter the Log group name created.

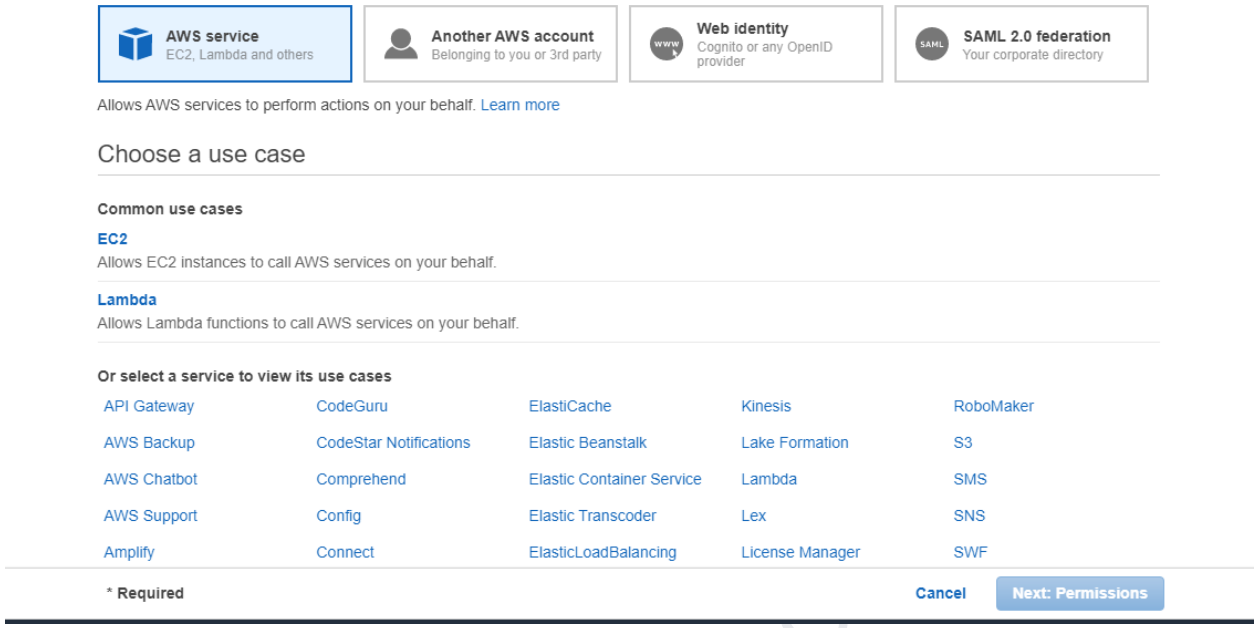
5. Create an IAM role which will allow VPC to Write to the log group

- To create an IAM role, go to the Identity and Access Management dashboard.



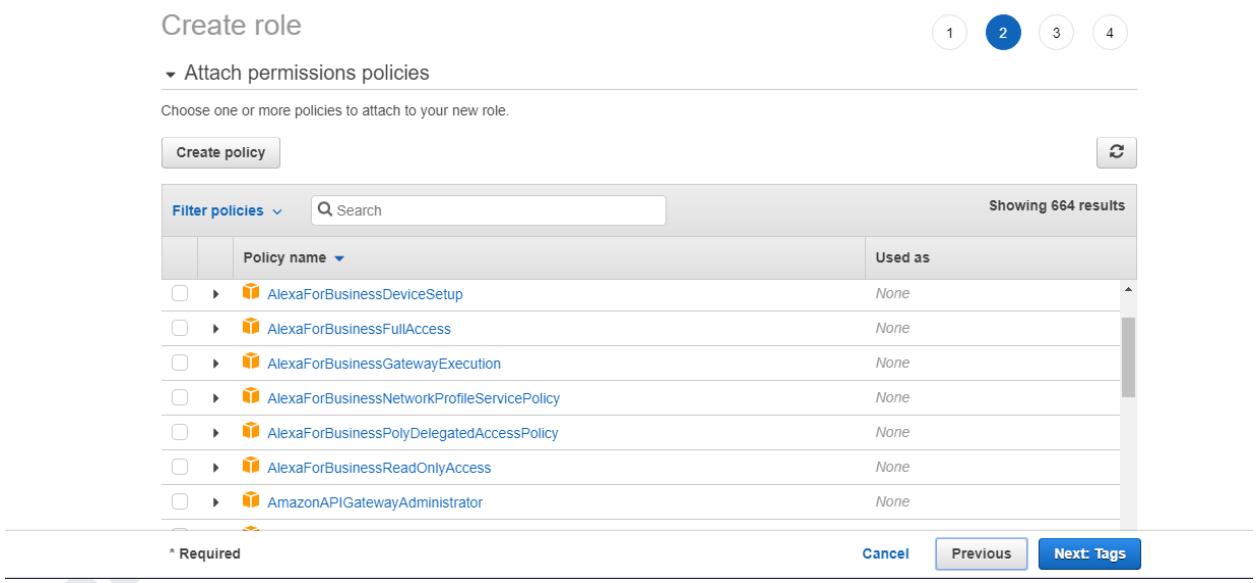
Click on Create a Role.

- The role that will be created will be an EC2 Role.



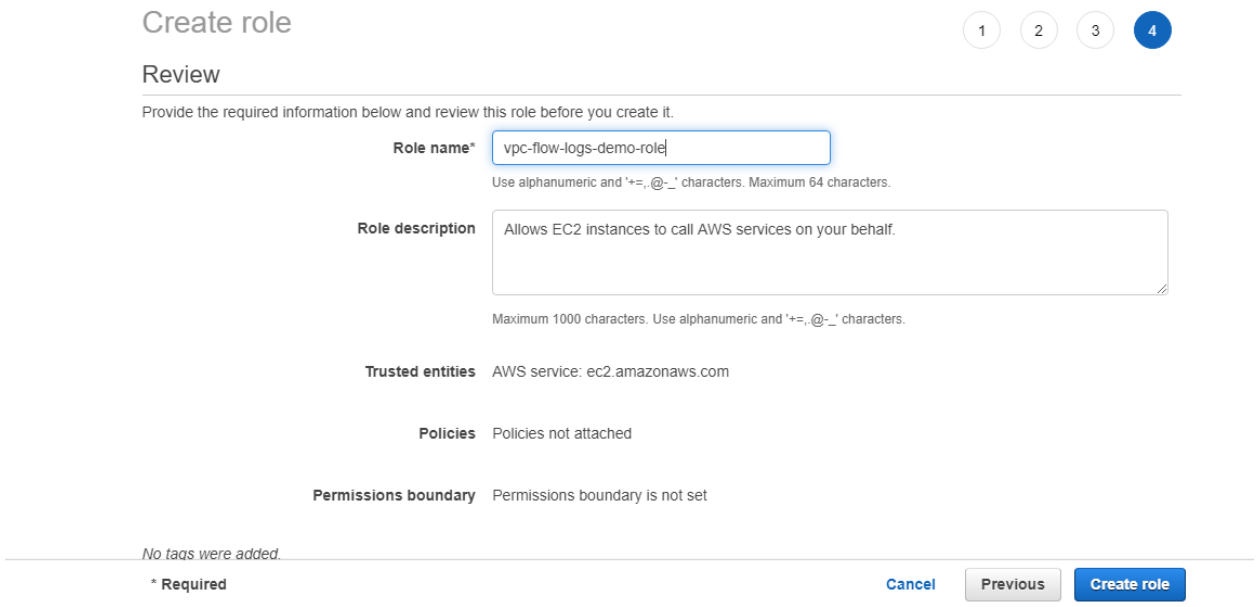
Select EC2 Role.

- Click Next to enter Tag names



Click on Next:Tags

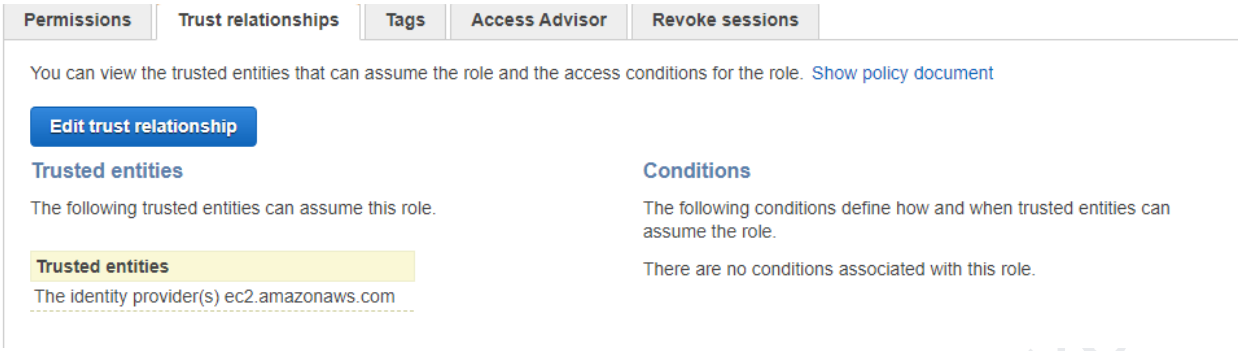
- Enter the Role name and click on create a role



- Attach the IAM roles for publishing flow logs to CloudWatch Logs
- Go to the Role you created and attach the inline policy

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
    "logs:PutLogEvents",
    "logs:DescribeLogGroups",
    "logs:DescribeLogStreams"
   ],
   "Effect": "Allow",
   "Resource": "*"
  }
 ]
}
```

- Ensure that your role has a trust relationship that allows the flow logs service to assume the role. This allows EC2 to write into the Log-group.
- Click on Trust Relationship in the Roles Dashboard

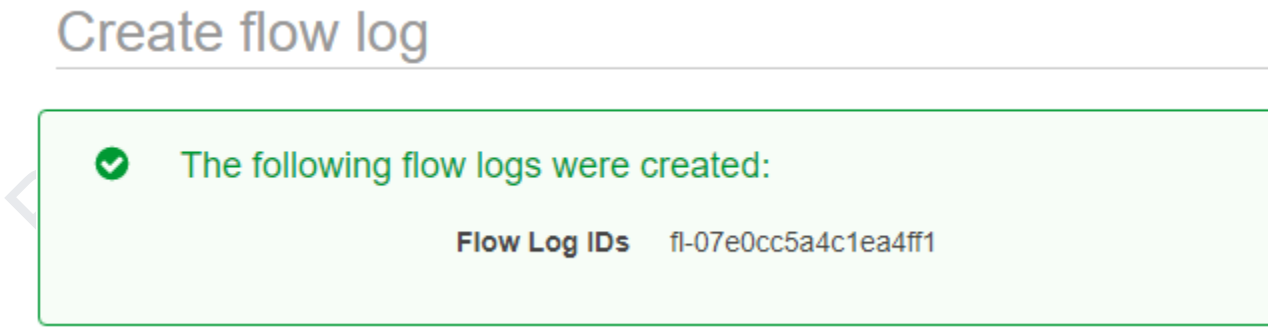


Click on Edit trust relationship

- Paste this there

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

6. Go back to your VPC Flow log dashboard and select the role you created. Click on Create flow logs.



The output of the flow log created

7. Wait for some time, return back to the CloudWatch dashboard and click on log groups you will see the Log stream Traffic.

```
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 40642 6 80 52700 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 18.185.97.81 172.31.35.55 40096 80 6 80 5800 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 18.185.97.81 172.31.35.55 40398 80 6 79 5748 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 41668 6 81 52752 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 18.185.97.81 172.31.35.55 80 40604 6 82 52804 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 41970 80 6 77 5644 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 18.185.97.81 172.31.35.55 80 41930 6 80 52700 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 41094 6 80 52700 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 42022 80 6 80 5800 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 41212 80 6 80 5800 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 40466 6 79 52648 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 41182 6 81 52752 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 80 39788 6 78 52596 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 40188 80 6 80 5800 1534192449 1534192626 ACCEPT OK
2 579807160478 eni-07ec21a29debdea6a 172.31.35.55 18.185.97.81 39204 80 6 79 5748 1534192449 1534192626 ACCEPT OK
```

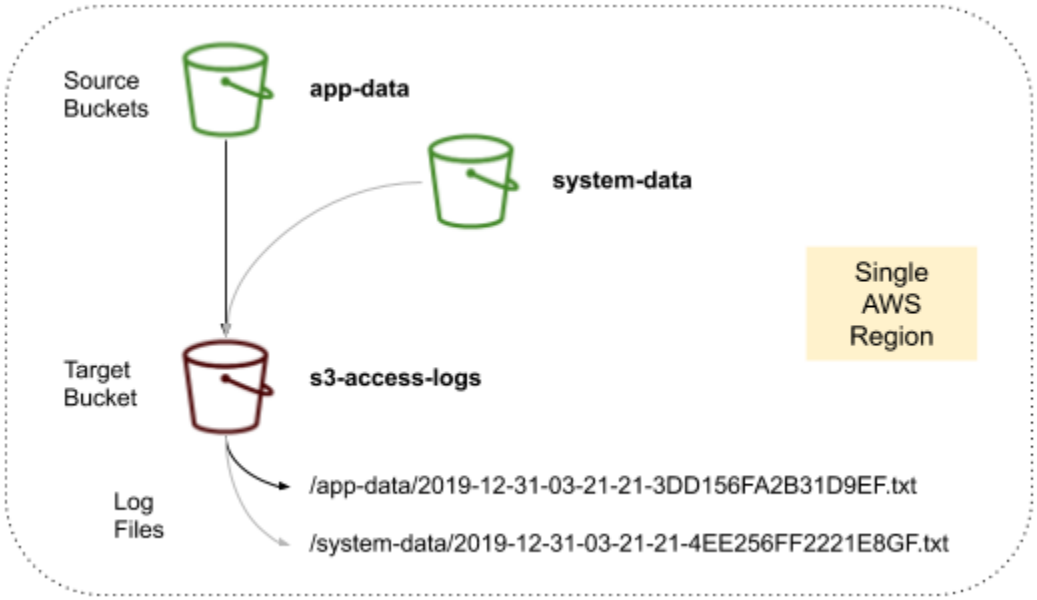
How a typical Flow log looks like

In the Flow log dashboard you can also filter traffic coming from a specific IP address.

Big Ups! You were able to capture information about the IP traffic in your VPC, storing the raw data in Amazon CloudWatch where it can be retrieved and viewed.

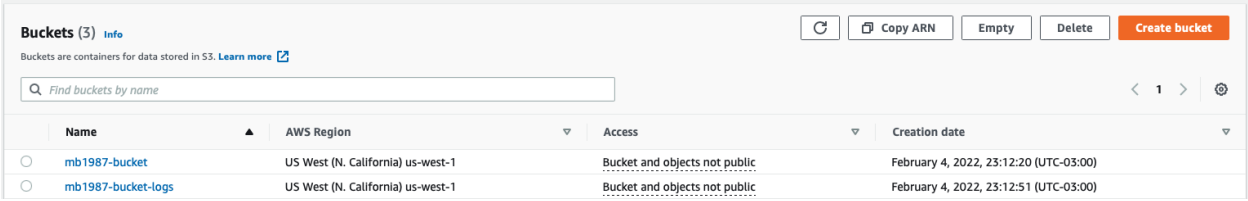
Here is the ***official AWS documentation*** for the same process we gone through above.

2. Setup S3 Access Logs to Log Management S3 Bucket.

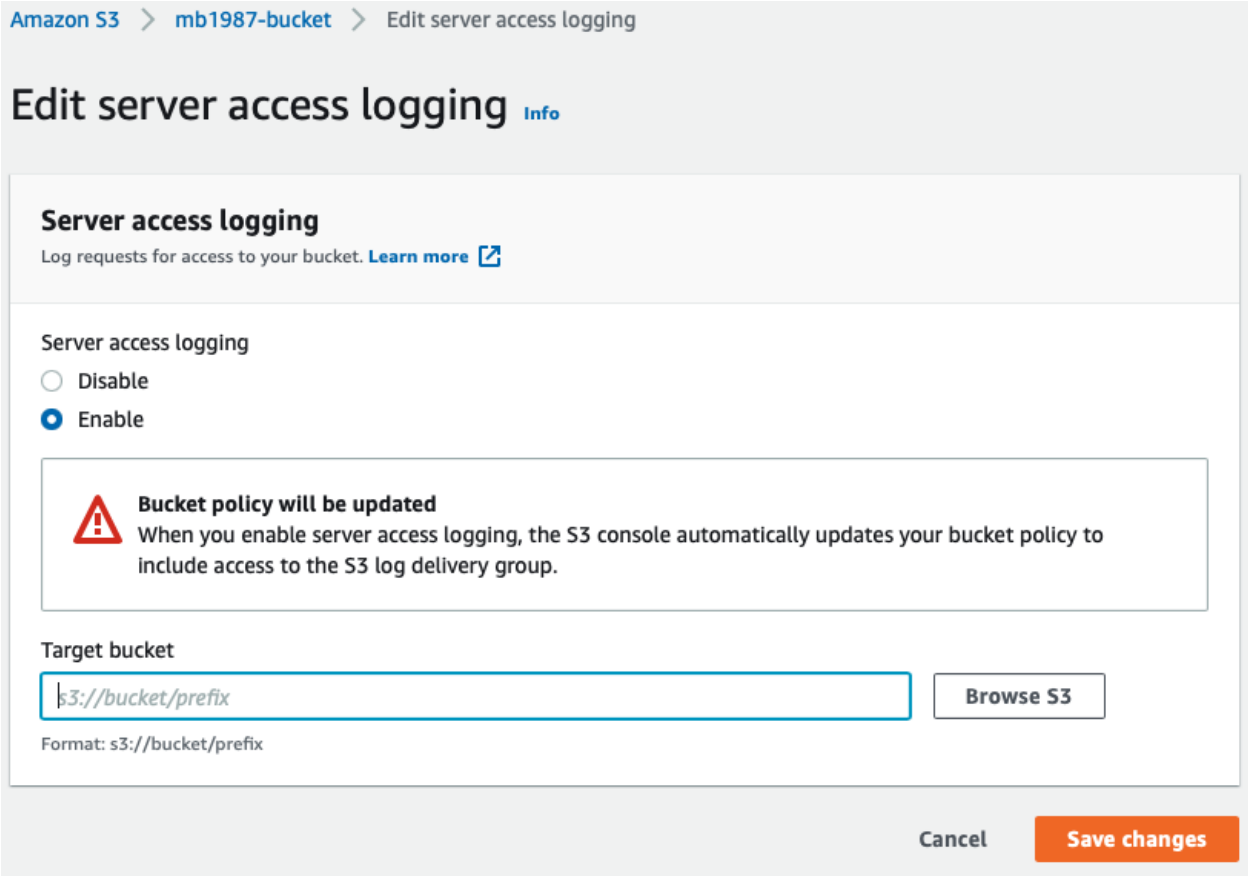


1. Enabling Logging for Bucket Objects

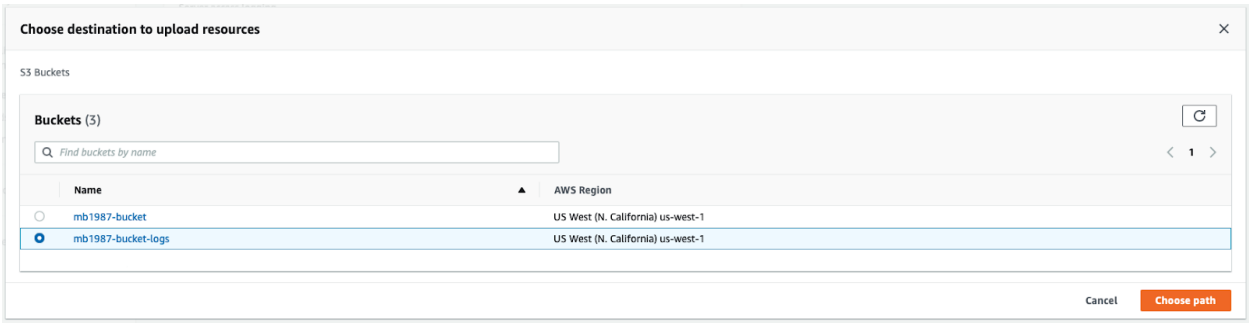
To use S3 logs, you first need to create one bucket to store files (objects) and another to store the logs. This should be created in the same region. It is a good practice not to save the logs in the same bucket because we want to save the logs for the interactions that the bucket receives and if the bucket has a problem the logs may not be able to be saved with the information about what is causing the error.



After you've created the buckets, go to the Properties of the bucket that will store the files to associate it with the bucket for logs. On the Properties page, click on the Edit button in the Server access logging box. In this form, select Enable to allow the bucket to provide log data about stored objects, then click on Browse S3 to select the log bucket.

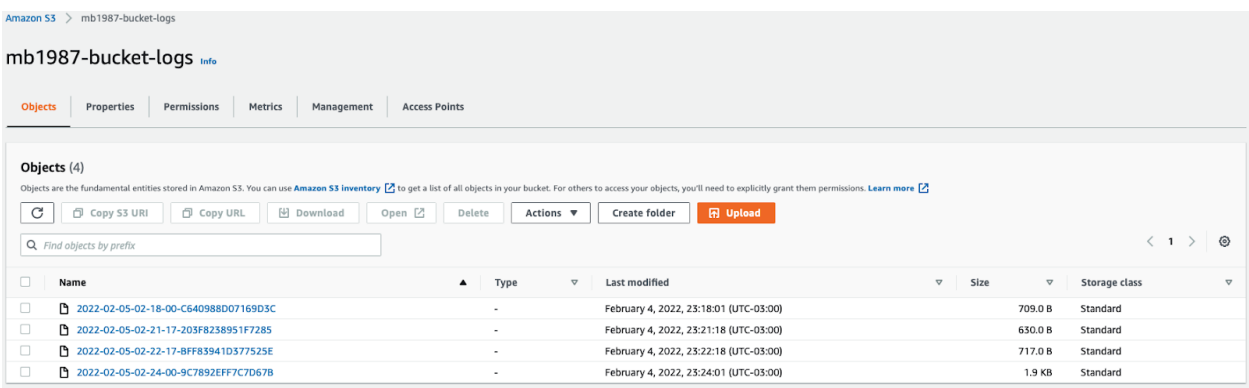


In the modal, select the proper bucket and click on Choose path. Back in the form, click on Save changes to apply the association between the buckets. Clicking that button is all you need to do to start saving object usage logs.



2. Testing Logging

Now, let's try to access the bucket and upload some files. For this test, there is no need to add any other settings. You can open, download, and remove files to generate logs for these actions. Then, you can access the bucket for logs and wait a few minutes to receive the log data about the newly uploaded file. Then, open the logs to see the type of data available in the log information. You will see logs related to actions taken within the bucket to get or remove objects, along with policies and versioning information.



3. Log Data Samples

Let's take a look at some sample logs and their available formats on AWS S3. For example, this is DELETE:

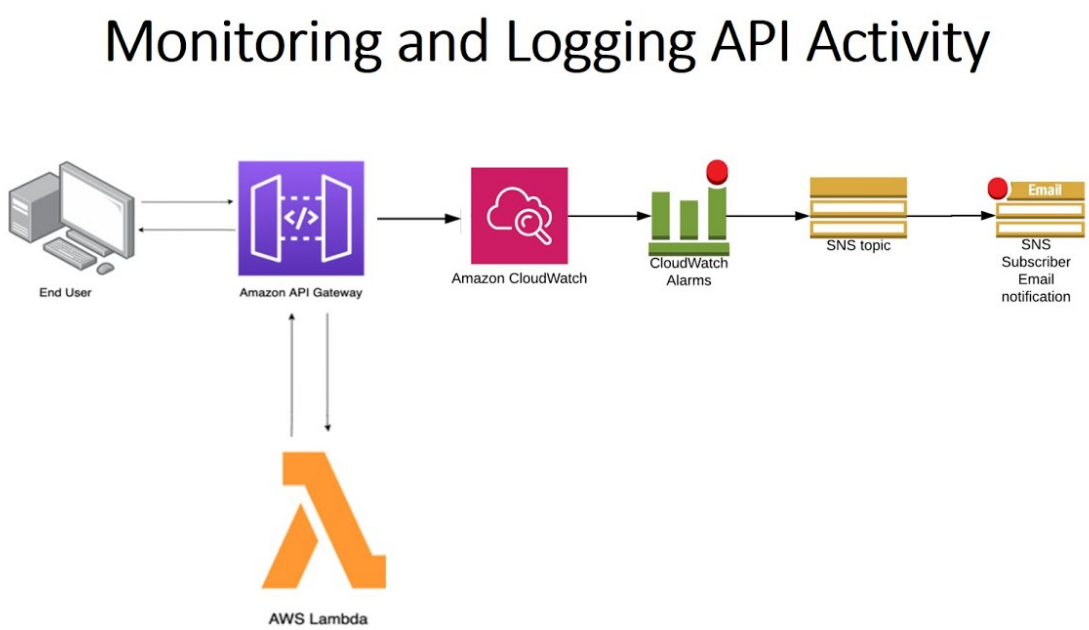
```
ce6f2c543de2b9a3a4fdf21d56e95135af4045032a9157cb2fdb1a4854c73110 mb1987-bucket [05/Feb/2022:01:09:08 +0000]
191.XXX.XXX.216 ce6f2c543de2b9a3a4fdf21d56e95135af4045032a9157cb2fdb1a4854c73110 BW21GJ60HATK1VCR
REST.POST.MULTI_OBJECT_DELETE - "POST /mb1987-bucket?delete= HTTP/1.1" 200 - 5100 - 802 - "-" "S3Console/0.4,
aws-internal/3 aws-sdk-java/1.11.1030 Linux/5.4.172-100.336.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.302-
b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/standard" -
NLZUuMfnLn6Kq3LKEB5GFRNAVEo9cy/BR5bmzhy4dXWD0ogwa6Q71IzUbJKlidFbVwUiRR9jk9w= SigV4 ECDHE-RSA-AES128-GCM-
SHA256 AuthHeader s3-us-west-1.amazonaws.com TLSv1.2 -
```

And this is GET:

```
ce6f2c543de2b9a3a4fdf21d56e95135af4045032a9157cb2fdb1a4854c73110 mb1987-bucket [05/Feb/2022:01:08:58 +0000]
191.XXX.XXX.216 ce6f2c543de2b9a3a4fdf21d56e95135af4045032a9157cb2fdb1a4854c73110 MSAHCXNXGA3NJY9V
REST.GET.BUCKET - "GET /mb1987-bucket?list-type=2&encoding-type=url&max-keys=1&fetch-
owner=true&delimiter=&prefix= HTTP/1.1" 200 - 768 - 106 105 "-" "S3Console/0.4, aws-internal/3 aws-sdk-
java/1.11.1030 Linux/5.4.172-100.336.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.302-b08 java/1.8.0_302
vendor/Oracle_Corporation cfg/retry-mode/standard" -
X4VZjjWLHbLLApCiTeXoulUTQmYYduxisCY8E6se4YZx7FRBX6KsSd4lG+VwfrpgLqQXTdPbTGE= SigV4 ECDHE-RSA-AES128-GCM-
SHA256 AuthHeader s3-us-west-1.amazonaws.com TLSv1.2 -
```

Here is the ***official AWS documentation*** for the same process we have gone through above.

3. CloudWatch API logging using the API Gateway console



To set up CloudWatch API logging, you must have deployed the API to a stage. You must also have configured an appropriate CloudWatch Logs role ARN for your account.

1. Sign in to the API Gateway console at https://console.aws.amazon.com/apigateway
2. Choose a REST API.
3. Choose Settings from the primary navigation panel and enter an ARN of an IAM role with appropriate permissions in CloudWatch log role ARN. You need to do this once.

4. Do one of the following:
    a. Choose an existing API and then choose a stage.
    b. Create an API and deploy it to a stage.
5. Choose Logs/Tracing in the Stage Editor.
6. To enable execution logging:
    a. Choose a logging level from the CloudWatch Logs dropdown menu.

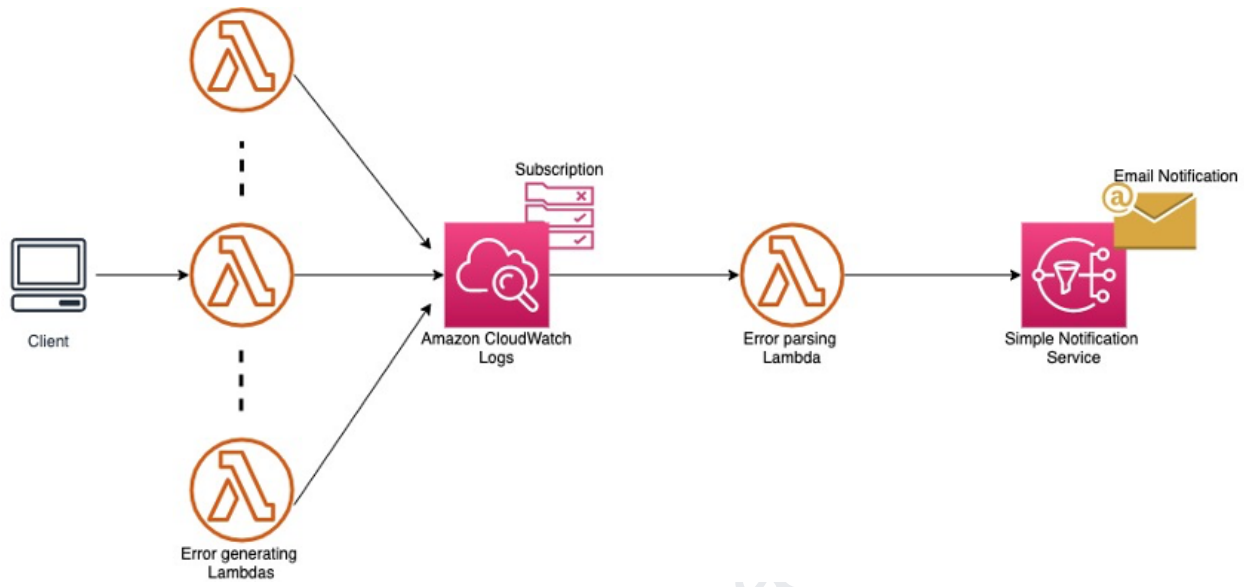
       Warning: *Full Request and Response Logs can be useful to troubleshoot APIs, but can result in logging sensitive data. We recommend not using Full Request and Response Logs for production APIs.*

    b. If desired, choose Enable Detailed CloudWatch Metrics.
7. For more information about CloudWatch metrics, see [Monitoring REST API execution with Amazon CloudWatch metrics](#).
8. To enable access logging:
    a. Choose Enable Access Logging under Custom Access Logging.
    b. Enter the ARN of a log group in Access Log Destination ARN. The ARN format is arn:aws:logs:{region}:{account-id}:log-group:log-group-name.
    c. Enter a log format in Log Format. You can choose CLF, JSON, XML, or CSV to use one of the provided examples as a guide.
9. Choose to Save Changes.

Note: *You can enable execution logging and access logging independently of each other.*

API Gateway is now ready to log requests to your API. You don't need to redeploy the API when you update the stage settings, logs, or stage variables.

4. Monitoring AWS Lambda Function Logs in CloudWatch Stream specific to Lambda Function CloudWatch Log Group.



Prerequisites:

Execution role attached to the Lambda Function needs permission to upload logs to CloudWatch Logs. You can add CloudWatch Logs permissions using the AWSLambdaBasicExecutionRole AWS managed policy provided by Lambda.

To add this policy to your role, run the following AWS CLI command *(Make sure you have configure AWS CLI using Access Key / Secret Key on the machine where you are executing the below AWS CLI command)*:

```
aws iam attach-role-policy --role-name your-role --policy-arn
arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

For more information, see AWS managed policies for Lambda features.

Pricing:

There is no additional charge for using Lambda logs; however, standard CloudWatch Logs charges apply. For more information, see CloudWatch pricing.

Using the Lambda console to view Lambda execution logs in CloudWatch:

To view logs using the Lambda console

1. Open the Functions page of the Lambda console.

2. Choose a function.
3. Choose **Monitor.**
4. Choose **View logs in CloudWatch.**

5. Capturing EC2 instance CPU/Mem/Disk Utilisation metrics by installing CW Agent on the server and monitoring them using a Custom CW Dashboard.

Step 1- Launch an EC2 with Ubuntu AMI 20.04

I believe that we can launch its own so I'm not explaining the process for it.
Used AMI: (Ubuntu 20.04)

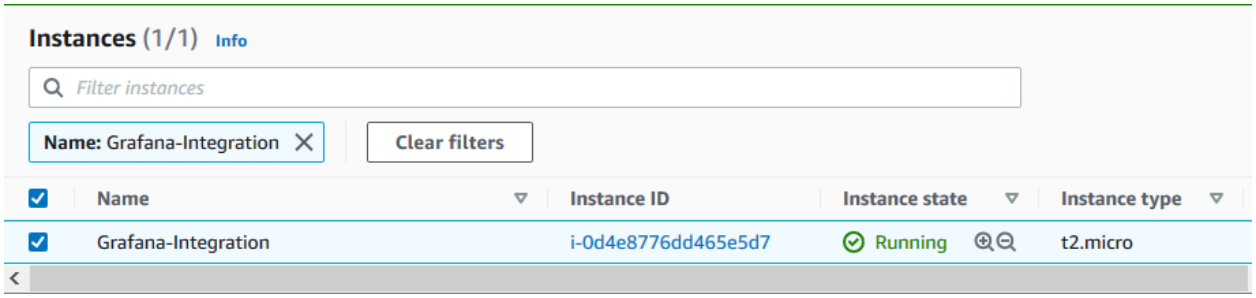
Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-09e67e426f25ce0d7 (64-bit x86) / ami-00d1ab6b335f217cf (64-bit Arm)

Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).

Free tier eligible

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

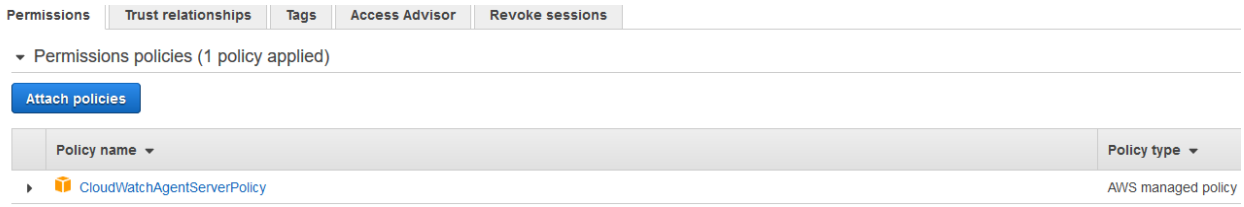
EC2 Instance: (Type: t2.micro)



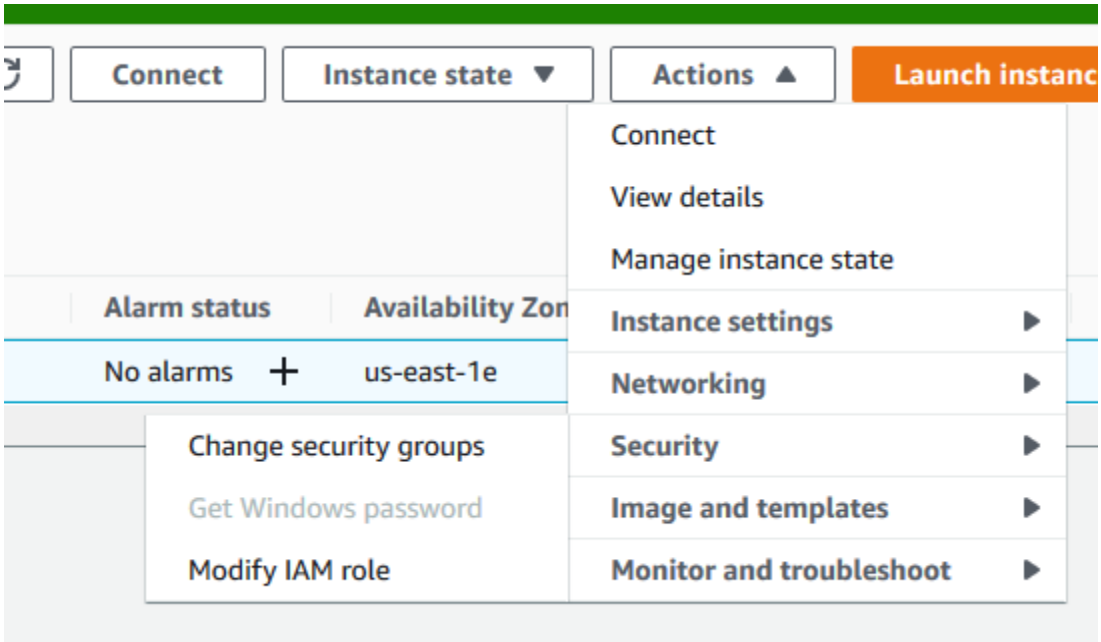
Step 2: Attach an IAM role with EC2 Instance

Access to AWS resources requires permissions. You create an IAM role, an IAM user, or both to grant permissions that the CloudWatch agent needs to write metrics to CloudWatch. If you're going to use the agent on Amazon EC2 instances, you must create an IAM role. If you're going to use the agent on on-premises servers, you must create an IAM user.

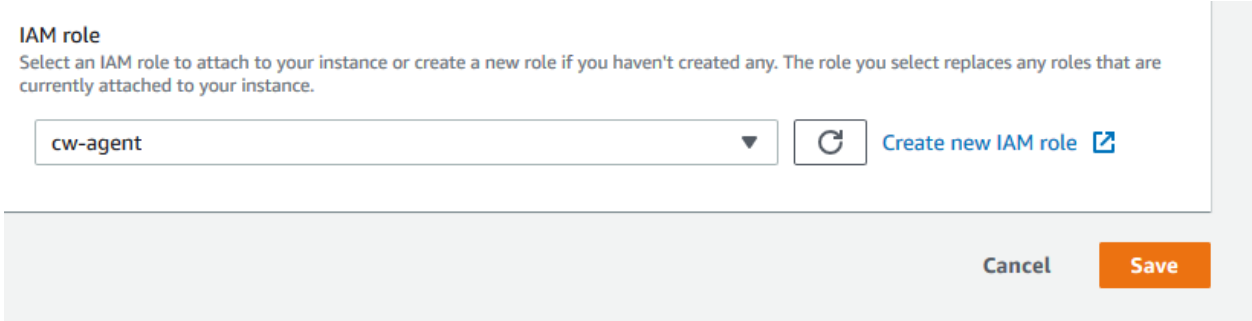
Create an IAM role, Make sure that AWS service is selected under Select type of trusted entity. For Choose a use case, choose EC2 under Common use cases, and Choose Next: Permissions. In the list of policies, select CloudWatchAgentServerPolicy & create an IAM role.



Now attach an IAM role with EC2 Instance. Go to EC2 — Select Instance — Click on Action — Security — Modify IAM role



Choose Create IAM role & save it.



Step 3: Download the CloudWatch Agent Package

Use the following steps to download the CloudWatch agent package, SSH to Instance & Download CW Agent package.

Download the CloudWatch agent:

```
wget
https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb
```

```
root@ip-172-31-52-40:/home/ubuntu# wget https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb
--2021-05-04 05:34:37--  https://s3.amazonaws.com/amazoncloudwatch-agent/ubuntu/amd64/latest/amazon-cloudwatch-agent.deb
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.216.1.27
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.216.1.27|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 54662644 (52M) [application/octet-stream]
Saving to: 'amazon-cloudwatch-agent.deb'

amazon-cloudwatch-agent.deb          100%[===================================================================================>]  52.13M  47.0MB/s    in 1.1s

2021-05-04 05:34:38 (47.0 MB/s) - 'amazon-cloudwatch-agent.deb' saved [54662644/54662644]

root@ip-172-31-52-40:/home/ubuntu#
```

Install the package:

dpkg -i -E ./amazon-cloudwatch-agent.deb

```
root@ip-172-31-52-40:/home/ubuntu# dpkg -i -E ./amazon-cloudwatch-agent.deb
Selecting previously unselected package amazon-cloudwatch-agent.
(Reading database ... 60149 files and directories currently installed.)
Preparing to unpack ./amazon-cloudwatch-agent.deb ...
create group cwagent, result: 0
create user cwagent, result: 0
Unpacking amazon-cloudwatch-agent (1.247347.6b250880-1) ...
Setting up amazon-cloudwatch-agent (1.247347.6b250880-1) ...
root@ip-172-31-52-40:/home/ubuntu#
```

Update Packages & Install collectd: (This will take a few minutes if you haven't updated your available updates prior)

apt-get update && apt-get install collectd

```
0 upgraded, 557 newly installed, 0 to remove and 4 not upgraded.
Need to get 216 MB of archives.
After this operation, 1150 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 libsane-common all 1.0.29-0ubuntu5.2 [277 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 libtalloc2 amd64 2.3.0-3ubuntu1 [29.5 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/main amd64 libtevent0 amd64 0.10.1-4 [35.5 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates/main amd64 libwbclient0 amd64 2:4.11.6+dfsg-0ubuntu1.8 [221 kB]
```

```
aspell-autobuildhash: processing: en [en_US-w_accents-only].
aspell-autobuildhash: processing: en [en_US-wo_accents-only].
Processing triggers for libgdk-pixbuf2.0-0:amd64 (2.40.0+dfsg-3ubuntu0.2) ...
Processing triggers for rygel (0.38.3-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
Processing triggers for dbus (1.12.16-2ubuntu2.1) ...
Processing triggers for systemd (245.4-4ubuntu3.6) ...
Processing triggers for sgml-base (1.29.1) ...
root@ip-172-31-52-40:/home/ubuntu#
```

Step 4: Create the CloudWatch Agent Configuration File

Before running the CloudWatch agent on any servers, you must create a CloudWatch agent configuration file. The agent configuration file is a JSON file that specifies the

metrics and logs that the agent is to collect, including custom metrics. The agent configuration file wizard, amazon-cloud watch-agent-config-wizard, asks for a series of questions.

/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard

We're using Linux machine so option 1

```
root@ip-172-31-52-40:/home/ubuntu# /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard
============================================================
= Welcome to the AWS CloudWatch Agent Configuration Manager =
============================================================
On which OS are you planning to use the agent?
1. linux
2. windows
3. darwin
default choice: [1]:
```

Using EC2 Instance so Option 1

```
Trying to fetch the default region based on ec2 metadata...
Are you using EC2 or On-Premises hosts?
1. EC2
2. On-Premises
default choice: [1]:
```

Select a user

```
Which user are you planning to run the agent?
1. root
2. cwagent
3. others
default choice: [1]:
```

We can skip these two options for memory metric.

```
Do you want to turn on StatsD daemon?
1. yes
2. no
default choice: [1]:
2
Do you want to monitor metrics from CollectD?
1. yes
2. no
default choice: [1]:
2
```

We can choose the below options according to our requirements.

```
Do you want to monitor any host metrics? e.g. CPU, memory, etc.
1. yes
2. no
default choice: [1]:
1
Do you want to monitor cpu metrics per core? Additional CloudWatch charges may apply.
1. yes
2. no
default choice: [1]:
2
Do you want to add ec2 dimensions (ImageId, InstanceId, InstanceType, AutoScalingGroupName) into all of your metrics if the info is available?
1. yes
2. no
default choice: [1]:
2
Would you like to collect your metrics at high resolution (sub-minute resolution)? This enables sub-minute resolution for all metrics, but you
cific metrics in the output json file.
1. 1s
2. 10s
3. 30s
4. 60s
default choice: [4]:
4
Which default metrics config do you want?
1. Basic
2. Standard
3. Advanced
4. None
default choice: [1]:
```

Provide some declarations for config files.

```
Are you satisfied with the above config? Note: it can be manually customized after the wizard completes to add additional items.
1. yes
2. no
default choice: [1]:
1
Do you have any existing CloudWatch Log Agent (http://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AgentReference.html) confi
on?
1. yes
2. no
default choice: [2]:
2
Do you want to monitor any log files?
1. yes
2. no
default choice: [1]:
2
Saved config file to /opt/aws/amazon-cloudwatch-agent/bin/config.json successfully.
Current config as follows:
```

We can check json file under /opt/aws/amazon-cloudwatch-agent/bin/config.json

```
Saved config file to /opt/aws/amazon-cloudwatch-agent/bin/config.json successfully.
Current config as follows:
{
        "agent": {
                "metrics_collection_interval": 60,
                "run_as_user": "root"
        },
        "metrics": {
                "metrics_collected": {
                        "disk": {
                                "measurement": [
                                        "used_percent"
                                ],
                                "metrics_collection_interval": 60,
                                "resources": [
                                        "*"
                                ]
                        },
                        "mem": {
                                "measurement": [
                                        "mem_used_percent"
                                ],
                                "metrics_collection_interval": 60
                        }
                }
        }
}
Please check the above content of the config.
The config file is also located at /opt/aws/amazon-cloudwatch-agent/bin/config.json.
Edit it manually if needed.
Do you want to store the config in the SSM parameter store?
1. yes
2. no
default choice: [1]:
2
Program exits now.
```

Now check the status of CW agent.

/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status

OR

systemctl status amazon-cloudwatch-agent

```
root@ip-172-31-52-40:/home/ubuntu# /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
{
  "status": "stopped",
  "starttime": "",
  "configstatus": "not configured",
  "cwoc_status": "stopped",
  "cwoc_starttime": "",
  "cwoc_configstatus": "not configured",
  "version": "1.247347.6b250880"
}
root@ip-172-31-52-40:/home/ubuntu#
```

Status is stopped now, so start it & check status after that.

/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json

```
root@ip-172-31-52-40:/home/ubuntu# /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -s -c file:/opt/aws/amazon-cloudwatch-age
nt/bin/config.json
****** processing amazon-cloudwatch-agent ******
/opt/aws/amazon-cloudwatch-agent/bin/config-downloader --output-dir /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d --download-source file:/opt/aws/a
mazon-cloudwatch-agent/bin/config.json --mode ec2 --config /opt/aws/amazon-cloudwatch-agent/etc/common-config.toml --multi-config default
Successfully fetched the config and saved in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp
Start configuration validation...
/opt/aws/amazon-cloudwatch-agent/bin/config-translator --input /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.json --input-dir /opt/aws/amazon-cloudwa
tch-agent/etc/amazon-cloudwatch-agent.d --output /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml --mode ec2 --config /opt/aws/amazon-cloudwatch-ag
ent/etc/common-config.toml --multi-config default
2021/05/04 05:59:30 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/file_config.json.tmp ...
Valid Json input schema.
I! Detecting run_as_user...
No csm configuration found.
No log configuration found.
Configuration validation first phase succeeded
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
amazon-cloudwatch-agent has already been stopped
Created symlink /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service → /etc/systemd/system/amazon-cloudwatch-agent.service.
root@ip-172-31-52-40:/home/ubuntu# /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -m ec2 -a status
{
  "status": "running",
  "starttime": "2021-05-04T05:59:31+00:00",
  "configstatus": "configured",
  "cwoc_status": "stopped",
  "cwoc_starttime": "",
  "cwoc_configstatus": "not configured",
  "version": "1.247347.6b250880"
}
root@ip-172-31-52-40:/home/ubuntu#
```

Troubleshooting:

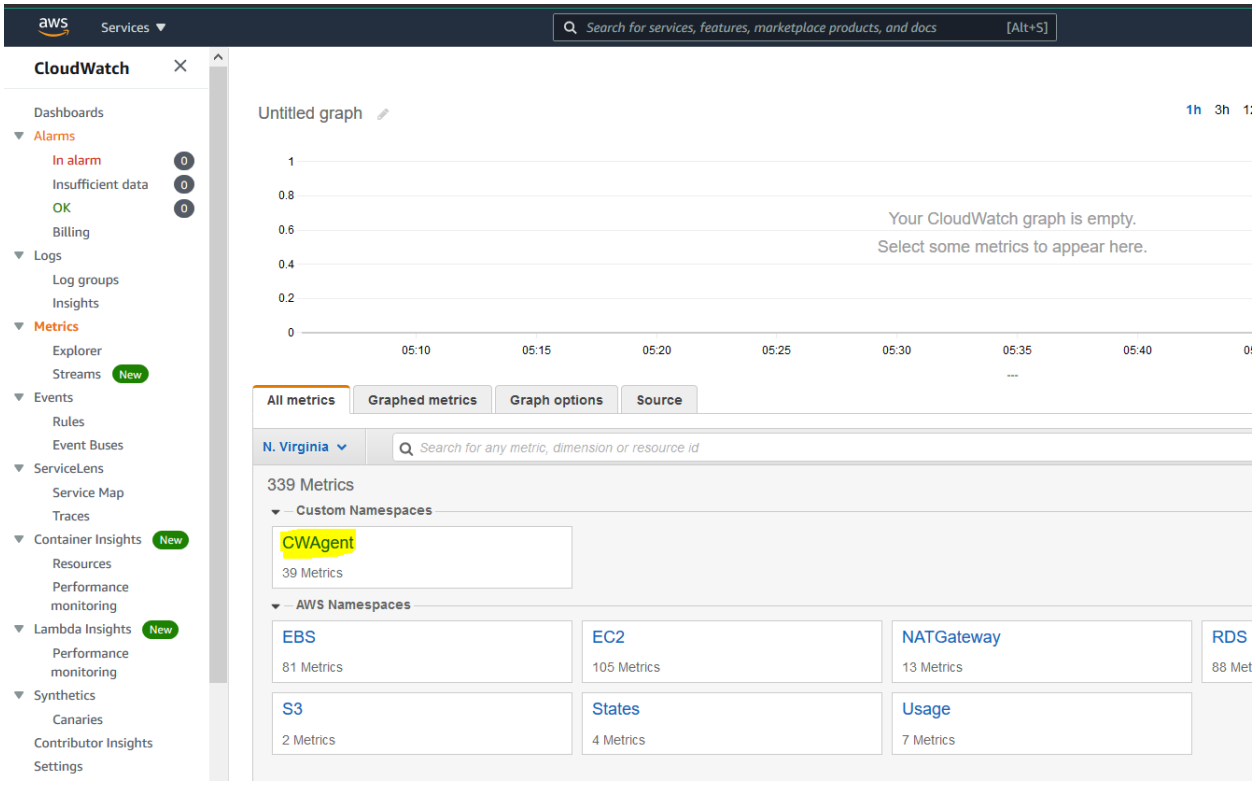
If config file for CW Agents emits error, execute below commands to setup collectD service for CW Agent.

sudo apt install collectd -y
mkdir -p /usr/share/collectd/

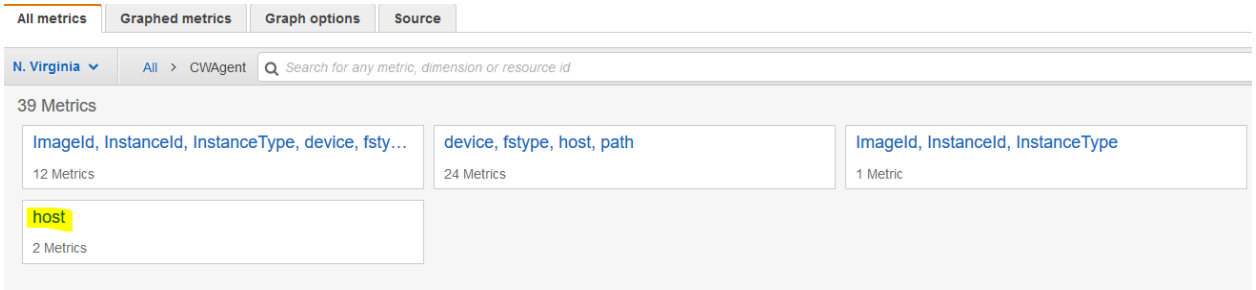
touch /usr/share/collectd/types.db

Step 5: Check Custom Metrics in AWS CloudWatch:

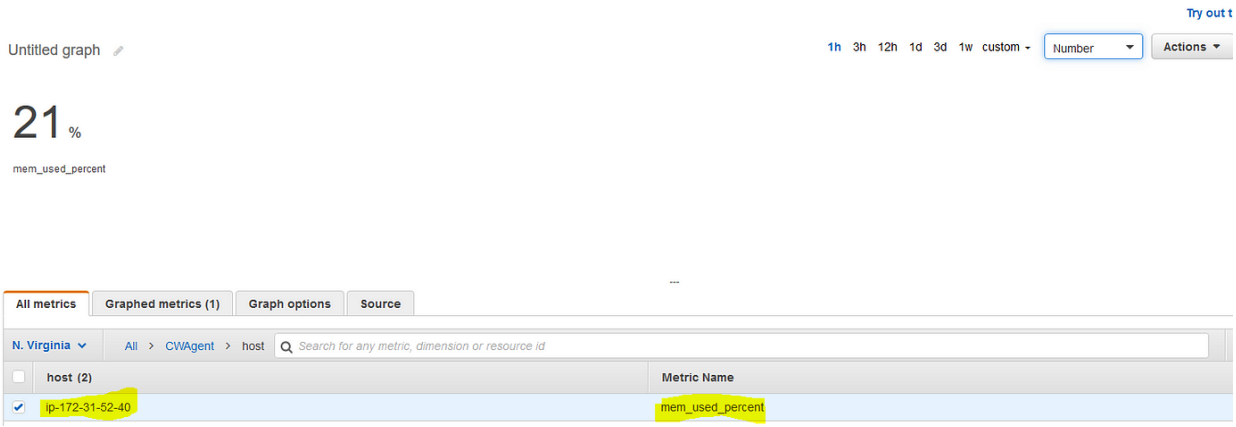
Custom Metrics:



Check inside a host for memory metrics:



Select the host according to the private IP of the instance & here you go with the memory metrics of the instance.



Step 6: Create a custom CW Dashboard and plot metrics on the created dashboard to monitor:

Now, since the metrics are visible on the CW Metrics, you can create a Custom Dashboard with Line graph, Bar graph, Pie chart visualization as per your monitoring requirements.

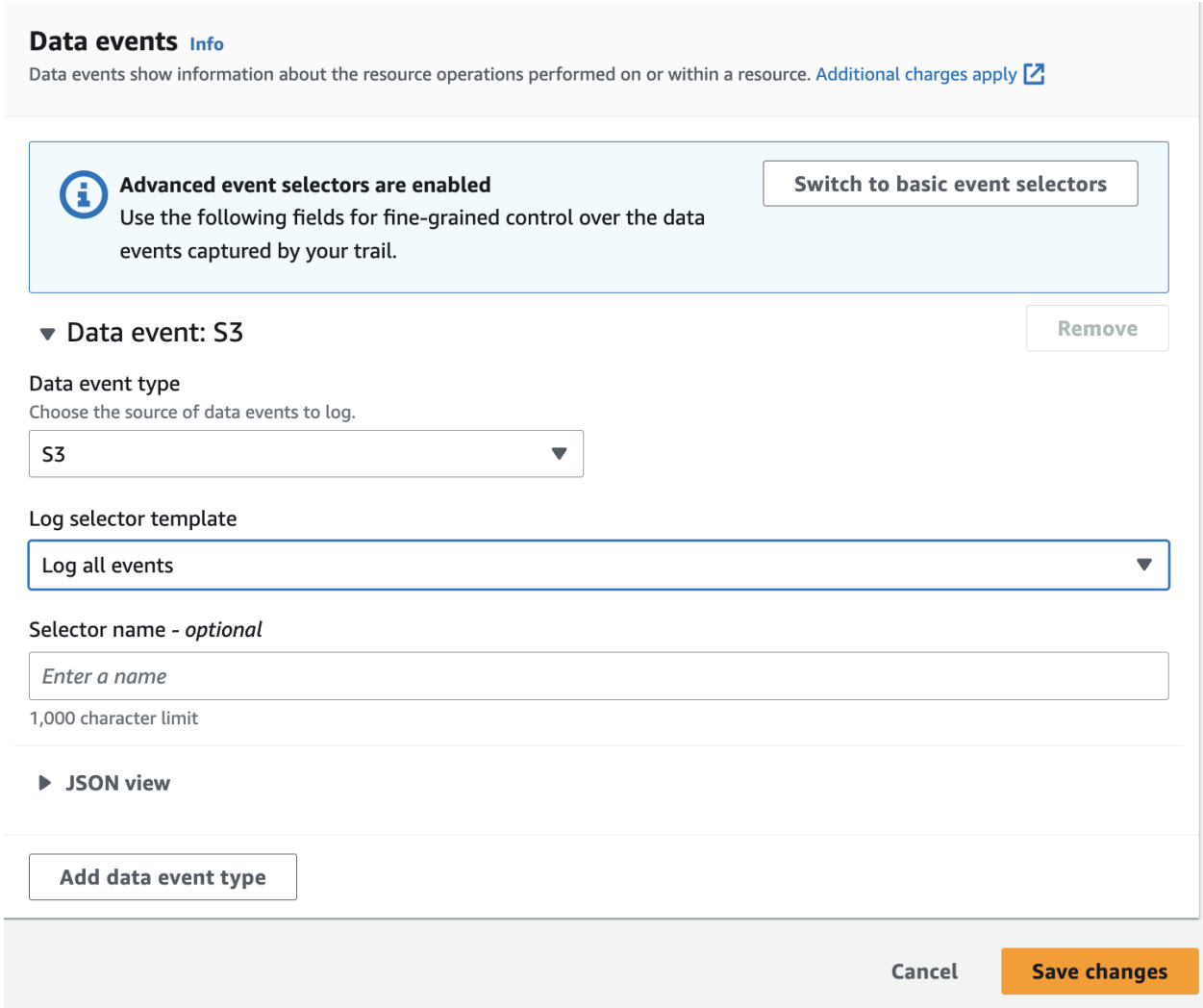
Select the metrics of your choice (which you want to monitor on the custom dashboard) under CWAgent Custom namespaces and click on Actions → Add to Dashboard → Select your custom dashboard where you want to capture and monitor these metrics.

6. Track user activities on your Cloud Environment using CloudTrail on any AWS Cloud Service.

For example, let's assume a scenario: You are given the task to monitor all the user activities performed by the Cloud, DevOps, and Dev team on the AWS S3 bucket service. You can leverage CloudTrail to track API calls made via Console, CLI, SDK, or any other means.

So, all you need to do is just enter CloudTrail in the AWS service search box and go to the CloudTrail service. Click on **Create Trail** and then provide the basic details like Trail name, Storage bucket name, and folder, wether you want to log the user activities in CW Log Groups as well or not, and Tag values.

Once you create the Trail, go to the newly created Trail from the home page of CloudTrail and navigate to the **Data Events** section. Click on the Edit button and you can select as per the requirements such as Event type, Log Selector template (to select logging level), etc., and create a new Data event to capture in the Target Trail S3 bucket.



Finally, after applying these configurations, you can start receiving logs in the target S3 bucket or/and the AWS CW Log Group as per your selection within a few minutes as soon as the configured data event is triggered (In this case we selected S3 All events to capture) in a JSON format as below:

test-bucket Aa ab .* 8 of 28


```
        }
    },
    {
        "eventVersion": "1.08",
        "userIdentity": {
            "type": "Root",
            "principalId": "615311846444",
            "arn": "arn:aws:iam::615311846444:root",
            "accountId": "615311846444",
            "accessKeyId": "ASIAY6Q3RVQWOJTZLX3J",
            "sessionContext": {
                "sessionIssuer": {},
                "webIdFederationData": {},
                "attributes": {
                    "creationDate": "2023-04-28T00:22:24Z",
                    "mfaAuthenticated": "true"
                }
            }
        },
        "eventTime": "2023-04-28T01:27:51Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "GetBucketAcl",
        "awsRegion": "ca-central-1",
        "sourceIPAddress": "99.231.111.68",
        "userAgent": "[S3Console/0.4, aws-internal/3 aws-sdk-java/1.11.1030 Linux/5.4.238-155.347.
        amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/25.362-b10 java/1.8.0_362 vendor/
        Oracle_Corporation cfg/retry-mode/standard]",
        "requestParameters": {
            "bucketName": "test-bucket-2456456",
            "Host": "test-bucket-2456456.s3.ca-central-1.amazonaws.com",
            "acl": ""
```