

AWS Documentation

No:-	Content
1.	CloudWatch Theory
2.	Cloud Dashboard
3.	Creation of custom namespace
4.	Events

CloudWatch Theory:

1. The Notebook's Eyes: Monitoring

- **CloudWatch** has eyes everywhere! It watches your cloud house, checking the health of your servers, applications, and even your coffee machine (well, not really, but almost anything in AWS).
- It looks for things like how much memory your servers are using, how much traffic your website is getting, or if anything is acting funny (errors).

2. Magic Alarms: Alerts

- If **CloudWatch** sees something weird—like your server getting too hot or your website getting too much traffic—it rings a **magic alarm**.
- This alarm can send you a message, so you know something needs fixing before it breaks.

3. Time Machine Pages: Logs

- **CloudWatch** also has time machine pages. It writes down everything that happens so you can go back and look if something goes wrong.

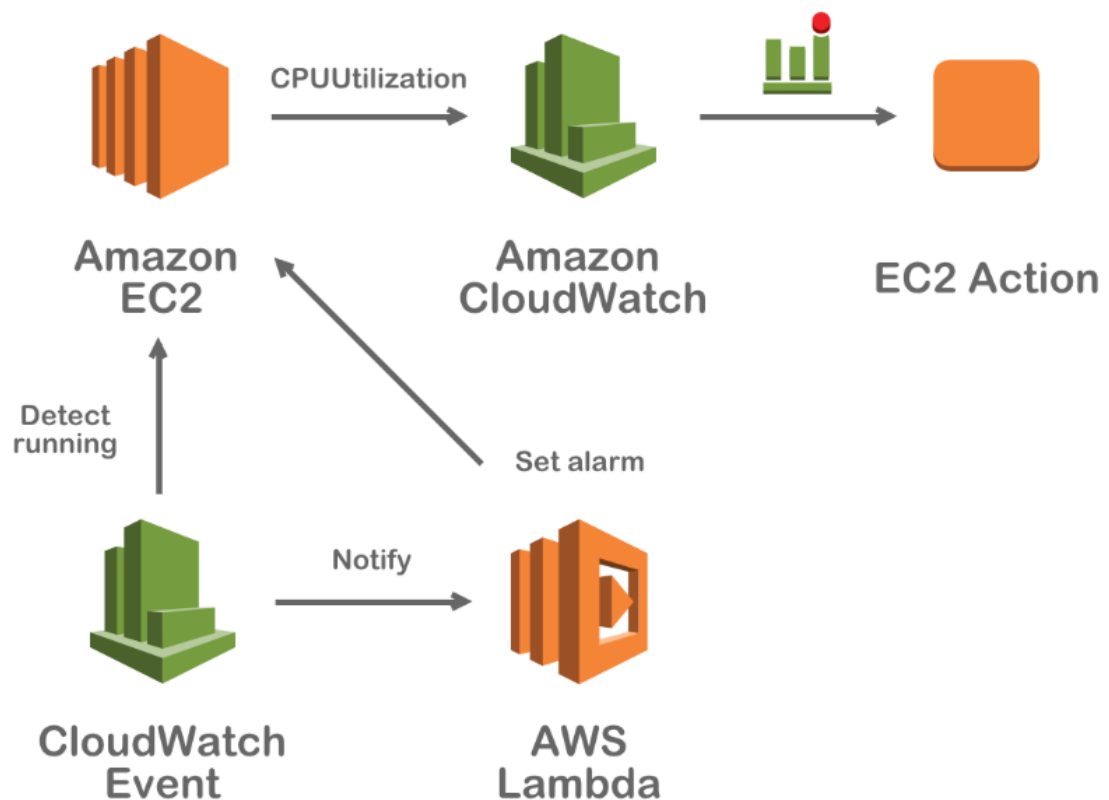
- These pages (logs) help you understand what happened in the past, like why your server stopped working at midnight.

4. Smart Insights: Metrics

- The notebook doesn't just watch and write—it's smart! **CloudWatch** can look at all the data it's collected and give you insights.
- For example, it can show you how your server's performance has changed over time or if your website's traffic is growing.

5. Custom Spells: Custom Metrics

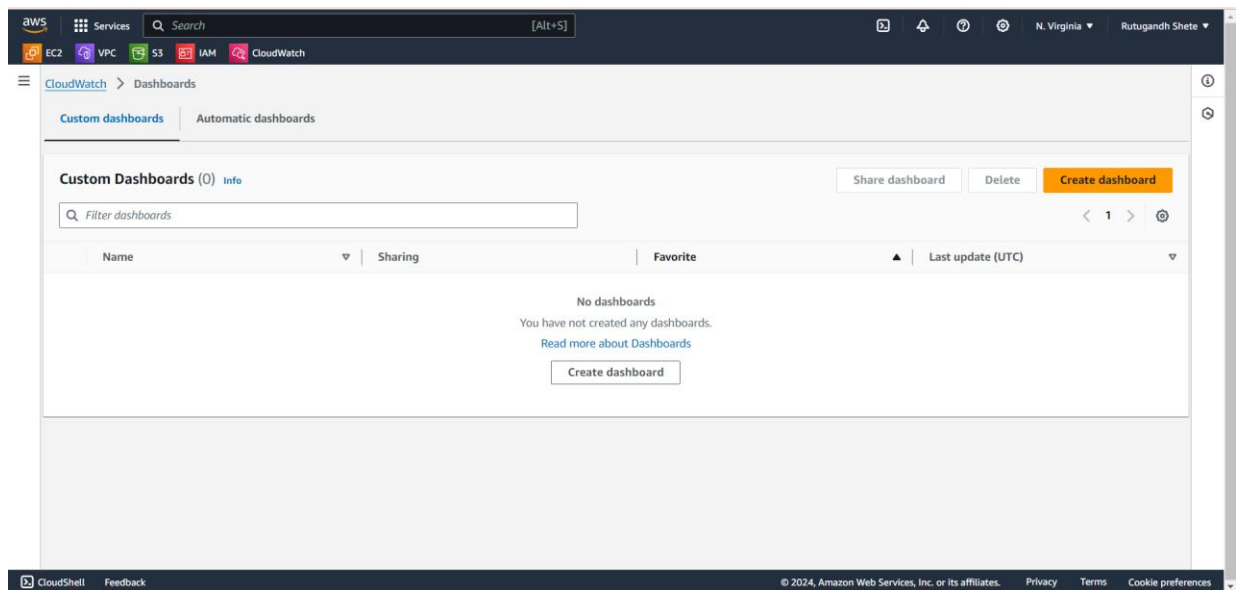
- You can teach **CloudWatch** to look for specific things that matter to you, like how many orders your online store gets every hour.
- These are called **custom metrics**—you create the spells (rules), and **CloudWatch** watches them for you.



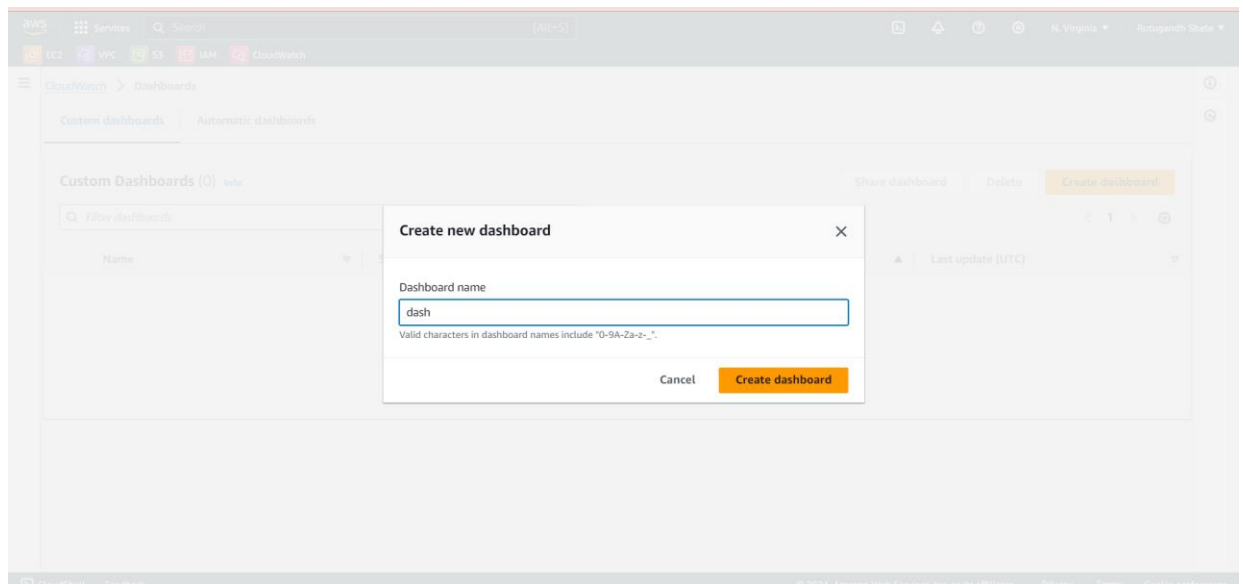
Cloud watch dashboard:

Steps:

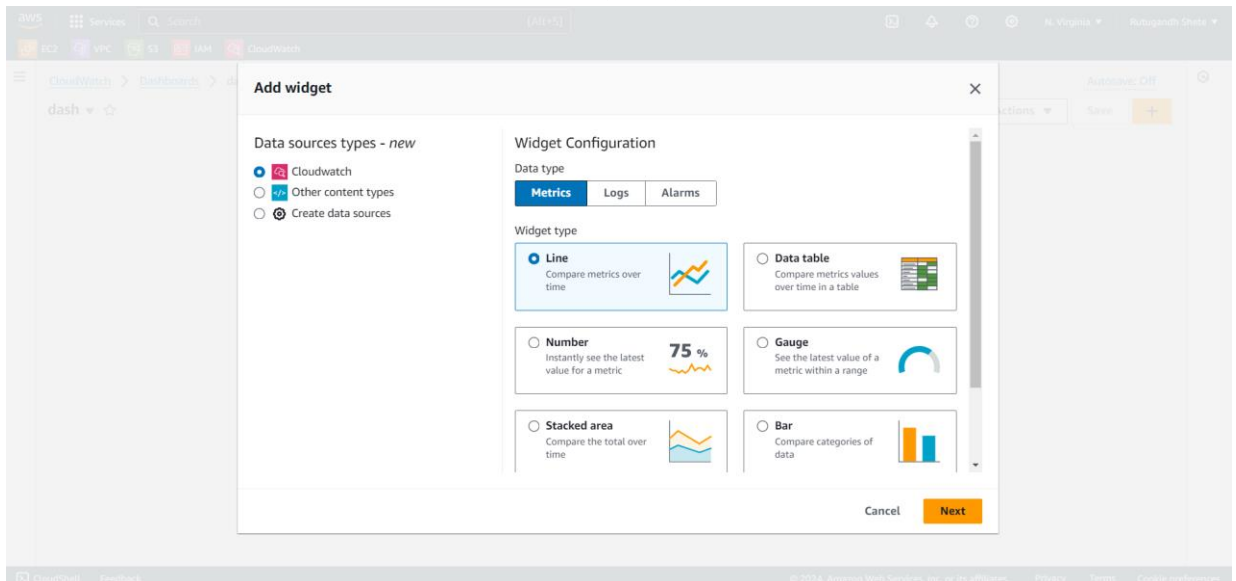
- Go to service → Create dashboard



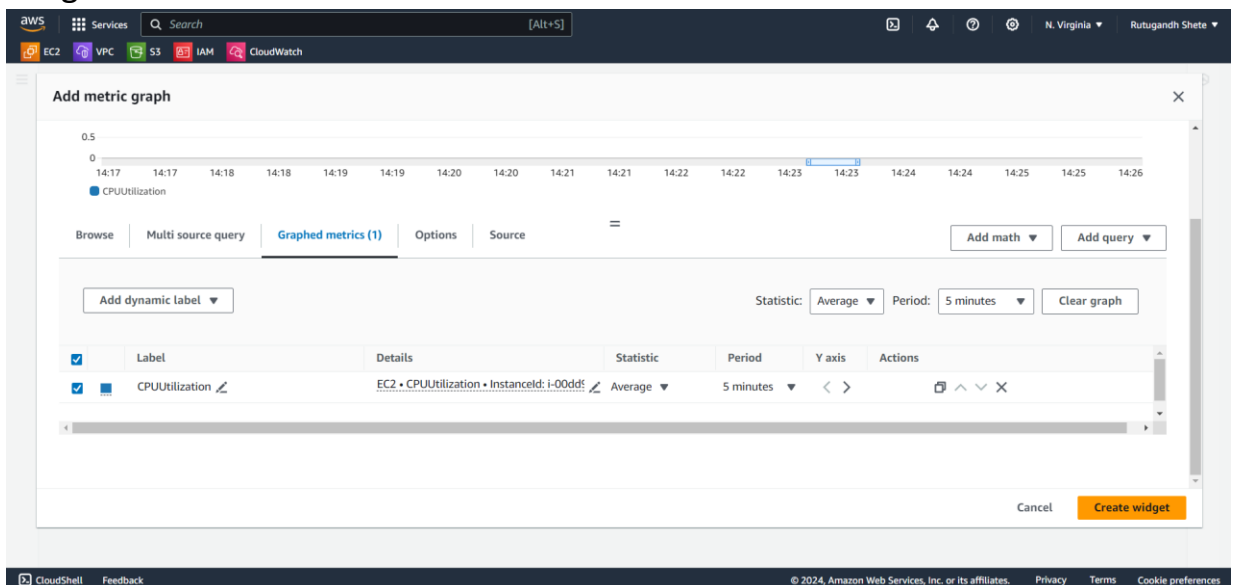
- Give name to dashboard.



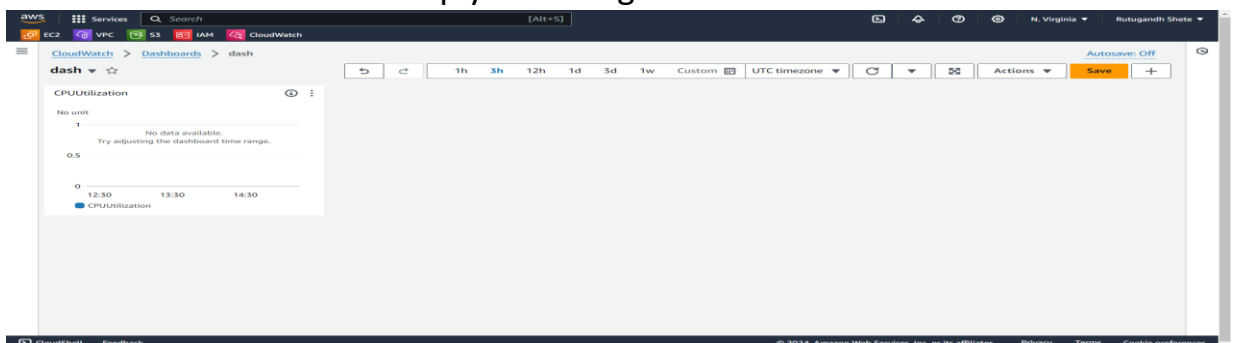
- We can see widget configuration → by choosing data type we can see the results in different forms like into different widget type like line, bars etc.



- Create widget, **Customize Display**: Adjust colors, labels, or thresholds to make the widget clear and useful.



- We can After configuring the widget, click **Add to dashboard**.
You can drag and drop widgets to rearrange them on your dashboard.
Click **Save dashboard** to keep your changes



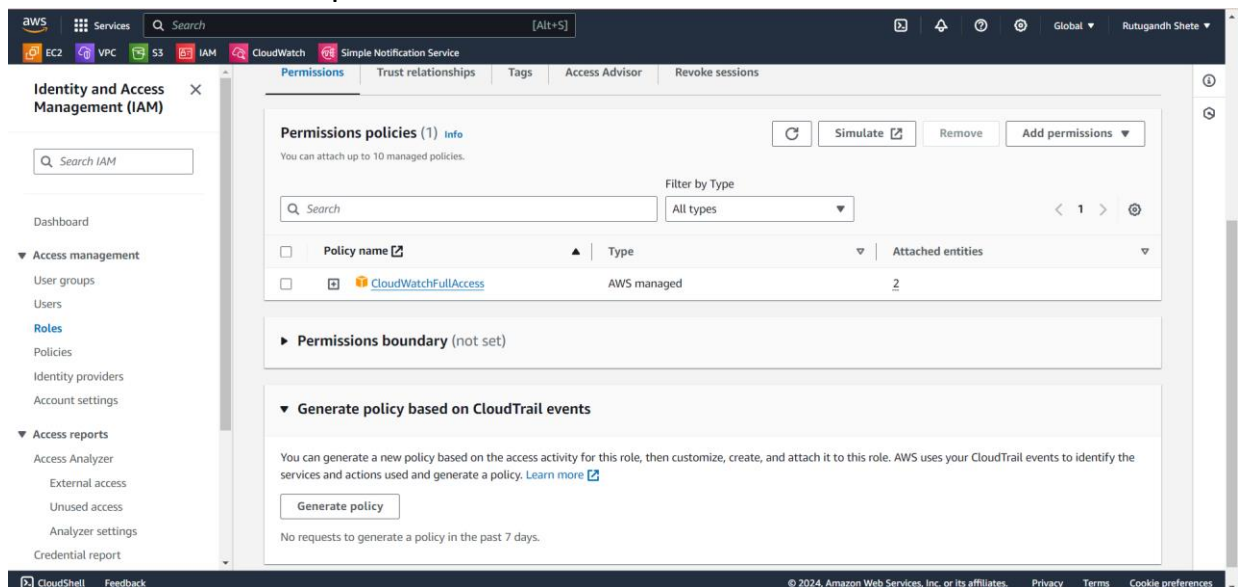
Creation of Custom Namespace:

We create custom namespace because if we want to monitor without default namespace then we use custom namespace.

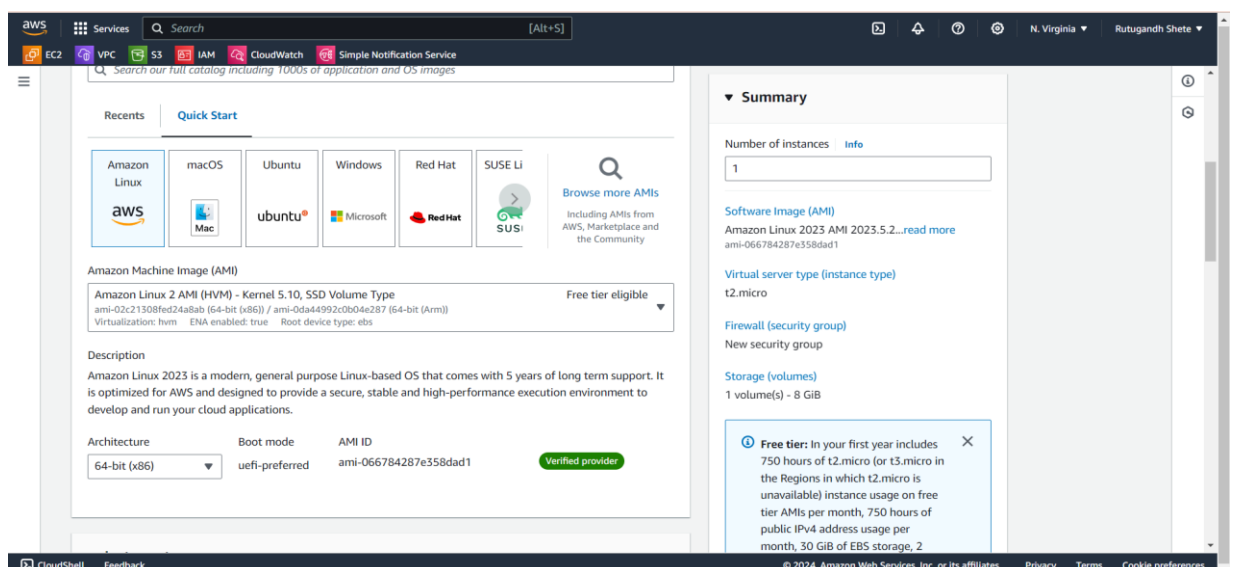
Real Example: - If you have an application that processes customer orders, you could create a custom namespace called MyApp/OrderProcessing and publish metrics like OrderProcessingTime, FailedOrders, or SuccessfulOrders.

Steps:

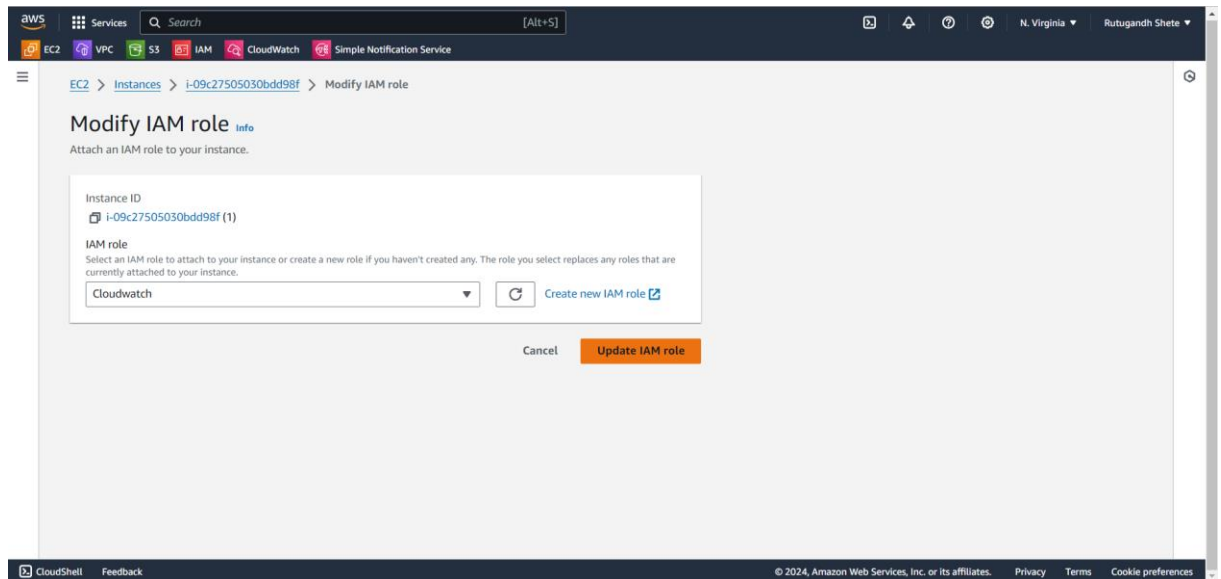
- Create IAM role with permissions “CloudwatchFullaccess”



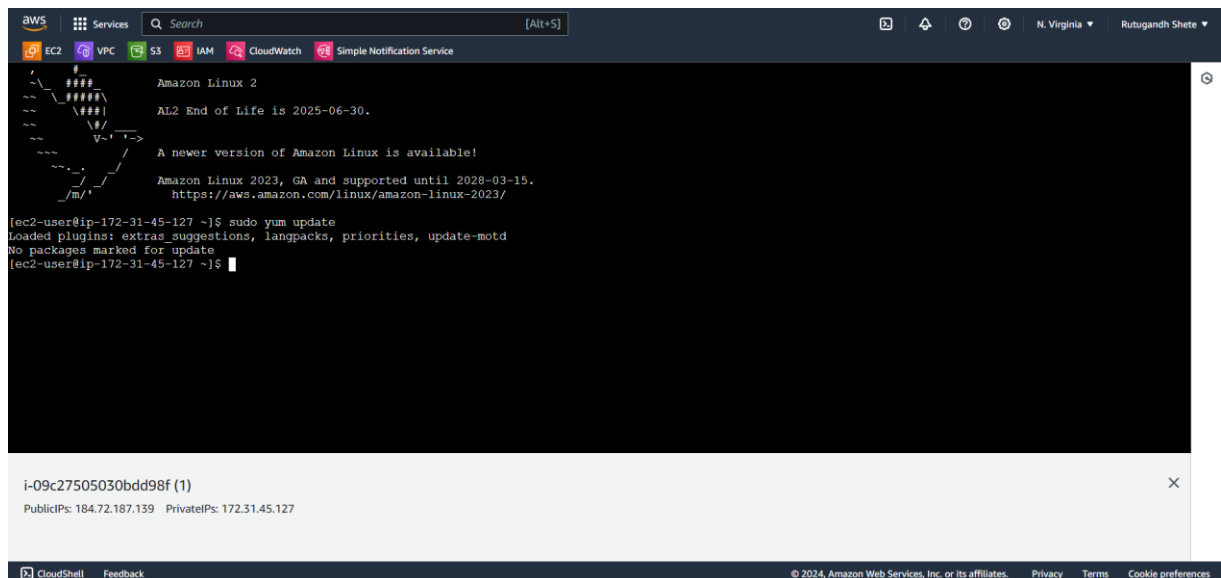
- Create instance but select AMI machine “Amazon machine 2”



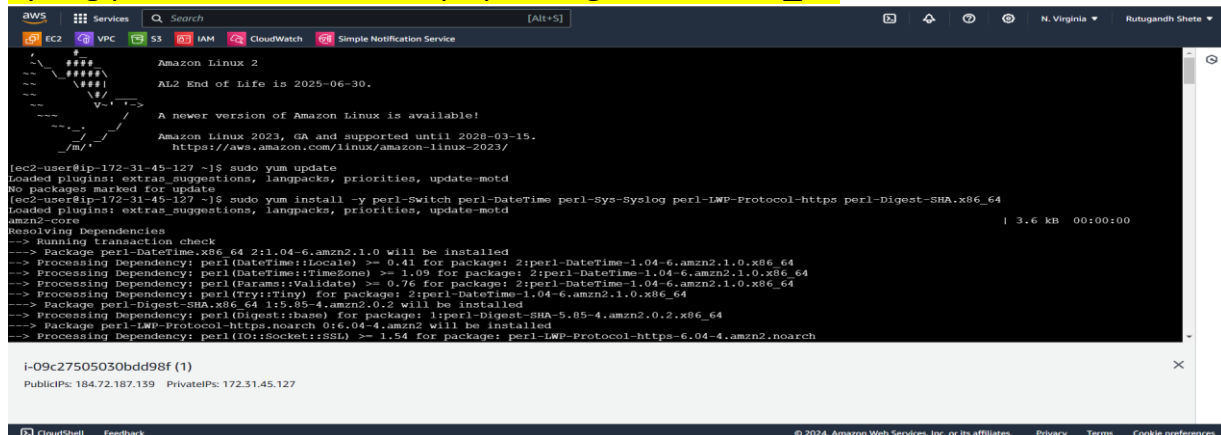
- Attach IAM role to instance.



- Connet instance to terminal.



- **Install Required Perl Packages:** These packages are necessary to run the monitoring scripts. "sudo yum install -y perl-Switch perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https perl-Digest-SHA.x86_64".



- You download a zip file containing the CloudWatch monitoring scripts, unzip it, and navigate to the folder where the scripts are stored. “curl https://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.2.zip -O”

```

AWS CloudShell
perl-Digest-SHA.x86_64 1:5.85-4.amzn2.0.2
perl-Sys-Syslog.x86_64 0:0.33-3.amzn2.0.2
perl-LWP-Protocol-https.noarch 0:6.04-4.amzn2
perl-Business-ISBN.noarch 0:2.06-2.amzn2
perl-Business-ISBN-Data.noarch 0:20120719.001-2.amzn2
perl-Class-Load.noarch 0:0.20-3.amzn2
perl-Class-Singleton.noarch 0:1.4-14.amzn2
perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.amzn2.0.2
perl-Data-Dumper.x86_64 0:2.145-3.amzn2.0.2
perl-Data-OptList.noarch 0:0.107-9.amzn2
perl-Digest-Local.noarch 0:0.45-6.amzn2
perl-Digest-MD5.x86_64 0:2.52-3.amzn2.0.2
perl-File-Find.x86_64 0:1.71-4.amzn2.0.2
perl-File-Listing.noarch 0:6.04-7.amzn2
perl-File-Temp.x86_64 0:0.02-3.amzn2.0.1
perl-HTTP-Daemon.noarch 0:6.01-8.amzn2.0.1
perl-HTTP-Message.noarch 0:6.06-6.amzn2
perl-HTTP-Negotiate.noarch 0:6.01-5.amzn2
perl-IO-Compress.noarch 0:2.061-2.amzn2
perl-IO-HTML.noarch 0:1.00-2.amzn2
perl-IO-Socket-IP.noarch 0:0.21-5.amzn2
perl-IO-Socket-SSL.noarch 0:1.94-7.amzn2.0.1
perl-JSON-PP.noarch 0:2.75-1.amzn2
perl-LWP-MediaTypes.noarch 0:6.02-2.amzn2
perl-Module-Implementation.noarch 0:0.06-6.amzn2
perl-Module-Runtime.noarch 0:0.013-4.amzn2
perl-Mozilla-CA.noarch 0:20130114-5.amzn2
perl-Net-HTTP.noarch 0:6.06-2.amzn2
perl-Net-LibIDN.x86_64 0:0.12-15.amzn2.0.2
perl-Net-SSLeay.x86_64 0:1.01.01-2.amzn2.0.1
perl-Net-URL.noarch 0:1.00-3.amzn2.0.2
perl-Package-Stash.x86_64 0:0.34-2.amzn2
perl-Package-DeprecationManager.noarch 0:0.13-7.amzn2
perl-Params-Util.x86_64 0:1.07-6.amzn2.0.2
perl-Params-Validate.x86_64 0:1.08-4.amzn2.0.2
perl-Sub-Install.noarch 0:0.926-6.amzn2
perl-Term-ANSIColor.noarch 0:5.00-3.amzn2.0.2
perl-Try-Tiny.noarch 0:0.12-2.amzn2
perl-URI.noarch 0:1.60-9.amzn2
perl-WWW-RobotRules.noarch 0:6.02-5.amzn2
perl-libwww-perl.noarch 0:6.05-2.amzn2

Complete!
[ec2-user@ip-172-31-45-127 ~]$ curl https://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.2.zip -O
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 24225 100 24225 0 0 591k 0 --:--:-- --:--:-- --:--:-- 591k
[ec2-user@ip-172-31-45-127 ~]$ ls
CloudWatchMonitoringScripts-1.2.2.zip
[ec2-user@ip-172-31-45-127 ~]$

```

- unzip CloudWatchMonitoringScripts-1.2.2.zip

```

AWS CloudShell
perl-HTTP-Negotiate.noarch 0:6.01-5.amzn2
perl-IO-Compress.noarch 0:2.061-2.amzn2
perl-IO-HTML.noarch 0:1.00-2.amzn2
perl-IO-Socket-IP.noarch 0:0.21-5.amzn2
perl-IO-Socket-SSL.noarch 0:1.94-7.amzn2.0.1
perl-LWP-MediaTypes.noarch 0:6.02-2.amzn2
perl-JSON-PP.noarch 0:2.75-1.amzn2
perl-Module-Implementation.noarch 0:0.06-6.amzn2
perl-Module-Runtime.noarch 0:0.013-4.amzn2
perl-Mozilla-CA.noarch 0:20130114-5.amzn2
perl-Net-HTTP.noarch 0:6.06-2.amzn2
perl-Net-LibIDN.x86_64 0:0.12-15.amzn2.0.2
perl-Net-SSLeay.x86_64 0:1.01.01-2.amzn2.0.1
perl-Net-URL.noarch 0:1.00-3.amzn2.0.2
perl-Package-Stash.x86_64 0:0.34-2.amzn2
perl-Package-DeprecationManager.noarch 0:0.13-7.amzn2
perl-Params-Util.x86_64 0:1.07-6.amzn2.0.2
perl-Params-Validate.x86_64 0:1.08-4.amzn2.0.2
perl-Sub-Install.noarch 0:0.926-6.amzn2
perl-Term-ANSIColor.noarch 0:5.00-3.amzn2.0.2
perl-Try-Tiny.noarch 0:0.12-2.amzn2
perl-URI.noarch 0:1.60-9.amzn2
perl-WWW-RobotRules.noarch 0:6.02-5.amzn2
perl-libwww-perl.noarch 0:6.05-2.amzn2

Complete!
[ec2-user@ip-172-31-45-127 ~]$ curl https://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.2.zip -O
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 24225 100 24225 0 0 591k 0 --:--:-- --:--:-- --:--:-- 591k
[ec2-user@ip-172-31-45-127 ~]$ ls
CloudWatchMonitoringScripts-1.2.2.zip
[ec2-user@ip-172-31-45-127 ~]$ unzip CloudWatchMonitoringScripts-1.2.2.zip
Archive: CloudWatchMonitoringScripts-1.2.2.zip
  extracting: aws-scripts-mon/awscreds.template
  inflating: aws-scripts-mon/AwsSignatureV4.pm
  inflating: aws-scripts-mon/CloudWatchClient.pm
  inflating: aws-scripts-mon/LICENSE.txt
  inflating: aws-scripts-mon/mon-get-instance-stats.pl
  inflating: aws-scripts-mon/mon-put-instance-data.pl
  inflating: aws-scripts-mon/NOTICE.txt
[ec2-user@ip-172-31-45-127 ~]$ ls
aws-scripts-mon  CloudWatchMonitoringScripts-1.2.2.zip
[ec2-user@ip-172-31-45-127 ~]$

```

- cd into unzipped file
- You run a specific script (mon-put-instance-data.pl) to collect memory usage data (like memory utilization, used memory, and available memory) and send this data to CloudWatch. “./mon-put-instance-data.pl --mem-util --mem-used --mem-avail”

```

[ec2-user@ip-172-31-45-127 aws-scripts-mon]$ ./mon-put-instance-data.pl --mem-util --mem-used --mem-avail
Successfully reported metrics to CloudWatch. Reference Id: 16e99746-8903-498c-9b93-0a5a9e1f5b6a
[ec2-user@ip-172-31-45-127 aws-scripts-mon]$

```

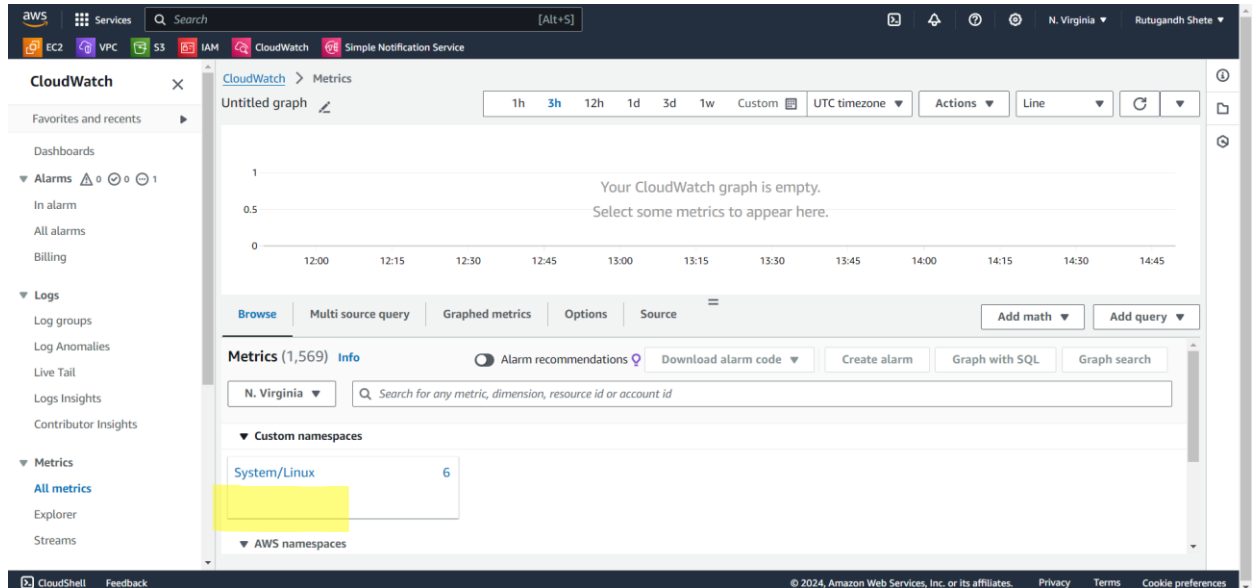
- You can set up a cron job to run this script every 5 minutes, so the memory and disk usage data is regularly sent to CloudWatch. `crontab -e` `* /5 * * *`
`* * ~/aws-scripts-mon/mon-put-instance-data.pl -- mem-used-incl-cache-buff -- mem-util -- disk-space-util -- disk-path=/ -- from-cron`

```

[ec2-user@ip-172-31-45-127 aws-scripts-mon]$ crontab -e
no crontab for ec2-user - using an empty one
crontab: installing new crontab
[ec2-user@ip-172-31-45-127 aws-scripts-mon]$ crontab -l
*/5 * * * * ~/aws-scripts-mon/mon-put-instance-data.pl -- mem-used-incl-cache-buff -- mem-util -- disk-space-util -- disk-path=/ -- from-cron
[ec2-user@ip-172-31-45-127 aws-scripts-mon]$

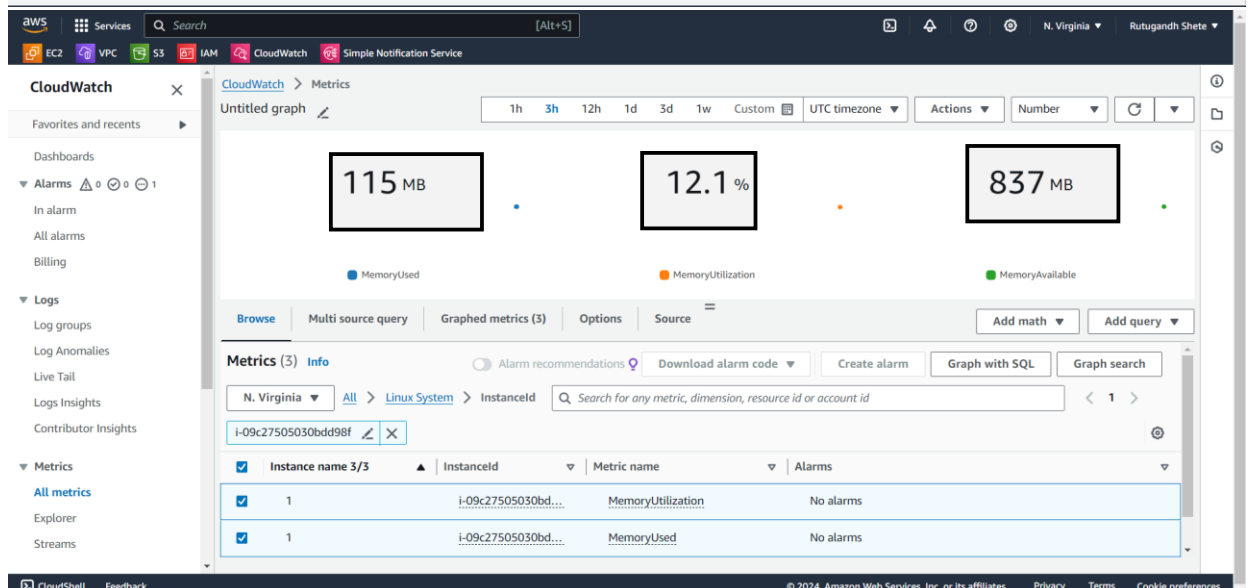
```

- Go to CloudWatch → Metrics → Select `All Metrics` → Select `Custom Namespaces` → Select `System/ Linux` Select `Instance Id`.



- Copy instance ID see the metrics that are available

<input type="checkbox"/>	Instance name 3/3	Instance ID	Metric name	Alarms
<input type="checkbox"/>	1	i-09c2750...	MemoryUtilization	No alarms
<input type="checkbox"/>	1	i-09c27505030bd...	MemoryUsed	No alarms
<input type="checkbox"/>	1	i-09c27505030bd...	MemoryAvailable	No alarms



Simple notification service:

Steps:

- **Create topic** → Name → description

The screenshot shows the 'Create topic' page in the AWS Management Console. The 'Type' section has two options: 'FIFO (first-in, first-out)' (selected) and 'Standard'. The 'Name' field is 'SNS' and the 'Display name' is also 'SNS'. The 'Content-based message deduplication' checkbox is unchecked. The 'Encryption - optional' section is expanded, showing that Amazon SNS provides in-transit encryption by default.

Type [Info](#)
Topic type cannot be modified after topic is created

☒ **FIFO (first-in, first-out)**

- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

☐ **Standard**

- Best-effort message ordering
- At-least once message delivery
- Highest throughput in publishes/second
- Subscription protocols: SQS, Lambda, HTTP, SMS, email, mobile application endpoints

Name

SNS .fifo
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_). FIFO topic names must end with ".fifo".

Display name - optional [Info](#)
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message.

SNS
Maximum 100 characters.

Content-based message deduplication [Info](#)
Enable default message deduplication based on message content. If unchecked, a deduplication ID must be provided for every publish request.

☐ Turn on message deduplication

Encryption - optional
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

- **Create subscription** → provide Protocol → add your Mail ID

The screenshot shows the 'Subscriptions' page in the AWS Management Console. A 'New Feature' banner is at the top. The 'Subscriptions (1)' section shows a table with one subscription. The table has columns for ID, Endpoint, Status, Protocol, and Topic. The subscription is confirmed and uses the EMAIL protocol.

Amazon SNS [New Feature](#)
Amazon SNS now supports in-place message archiving and replay for FIFO topics. [Learn more](#)

Subscriptions (1) [Edit](#) [Delete](#) [Request confirmation](#) [Confirm subscription](#) [Create subscription](#)

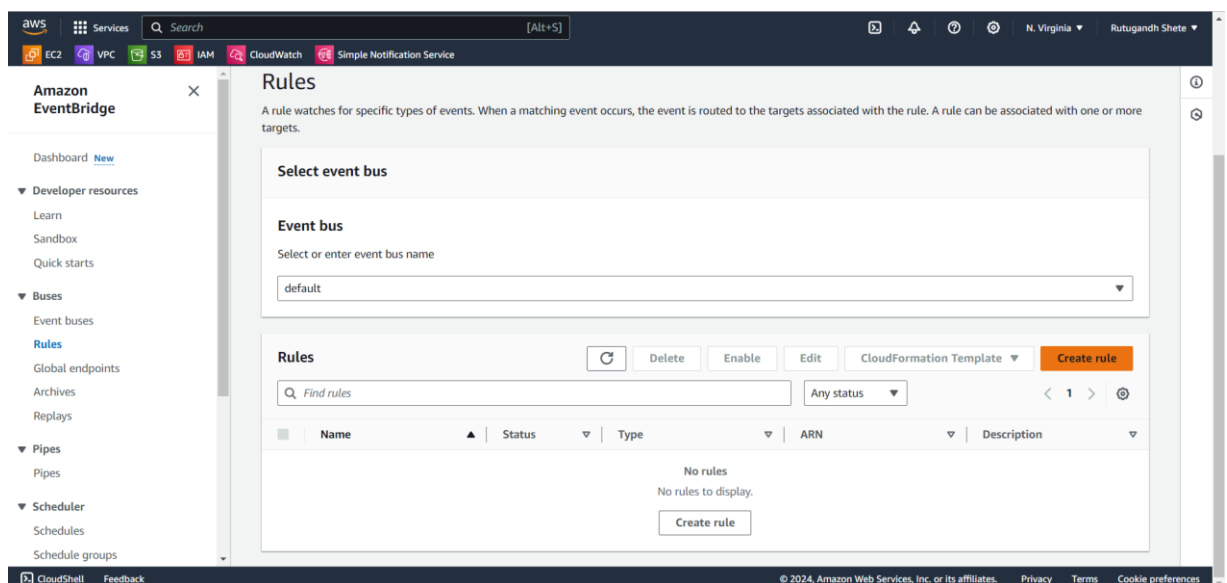
ID	Endpoint	Status	Protocol	Topic
0a826557-ccb3-4710-...	rutugandhshete.skn.co...	Confirmed	EMAIL	notification

Events:

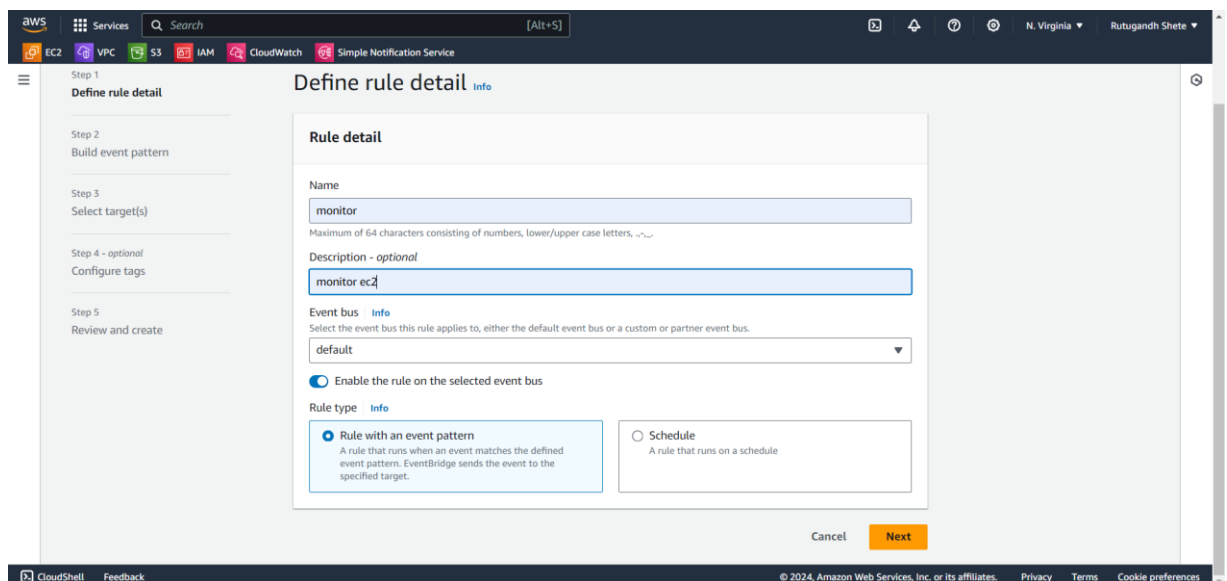
In AWS CloudWatch, **events** refer to actions or changes in your AWS environment that CloudWatch can detect and respond to. CloudWatch Events (now known as Amazon EventBridge) enables you to track these events and automatically trigger responses, such as invoking AWS Lambda functions, starting an EC2 instance, or sending notifications.

Steps:

- Go to Events → Create rule



- Define rule details.



- In Event pattern → select event source → select Aws service → event type.

The screenshot shows the 'Build event pattern' step in the AWS EventBridge console. The 'Event source' is set to 'AWS services', the 'AWS service' is 'EC2', and the 'Event type' is 'All Events'. The 'Event pattern' field contains a JSON snippet: `{ "source": ["aws.ec2"] }`. There are buttons for 'Copy', 'Test pattern', and 'Edit pattern'.

- Add target 1 → SNS service → topic that we have created.

The screenshot shows the 'Select target(s)' step in the AWS EventBridge console. Under 'Target types', 'AWS service' is selected. Under 'Select a target', 'SNS topic' is chosen, and the topic name 'notification' is entered. There are buttons for 'Add another target', 'Cancel', 'Skip to Review and create', 'Previous', and 'Next'.

- Event will get triggered when we perform certain actions on EC2 → we will get notified with SNS service through Mail. Terminate Instance.

