

Cloudwatch service

What is Metrics??

CloudWatch can collect metrics data and provide charts and graphs to visualize that data. Metrics can be collected for AWS resources like EC2 instances, RDS databases, Lambda functions, etc.

What is Alarms??

CloudWatch lets you set alarms that notify you about a certain threshold for a metric. This allows you to monitor your application and resources proactively.

What is Logs??

CloudWatch can collect log files generated by your resources and applications that are running on AWS. It provides storage and dashboards to visualize the log data.

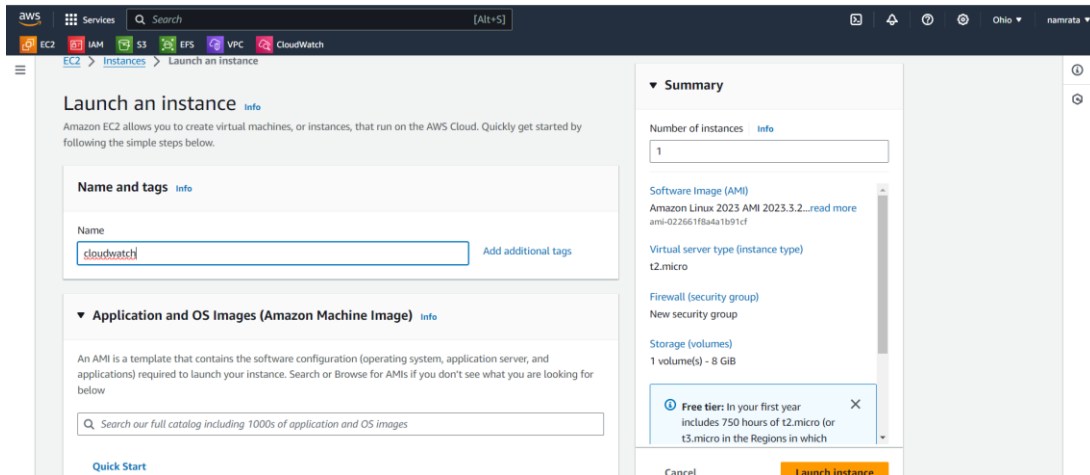
What is Events ??

CloudWatch Events allow you to trigger actions in reaction to changes in your resources or applications.

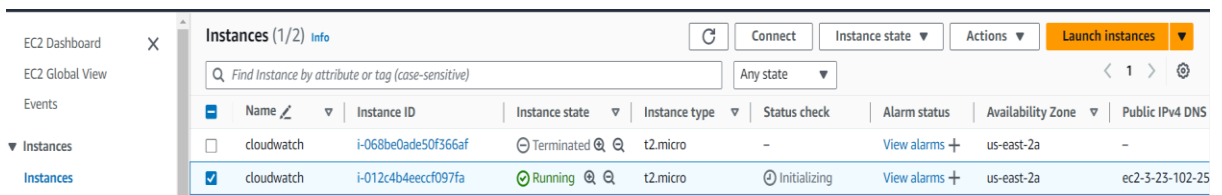
What is Dashboards??

CloudWatch provides fully customizable dashboards where you can add widgets with metrics and log data. This gives you a single view of the health and performance of your applications.

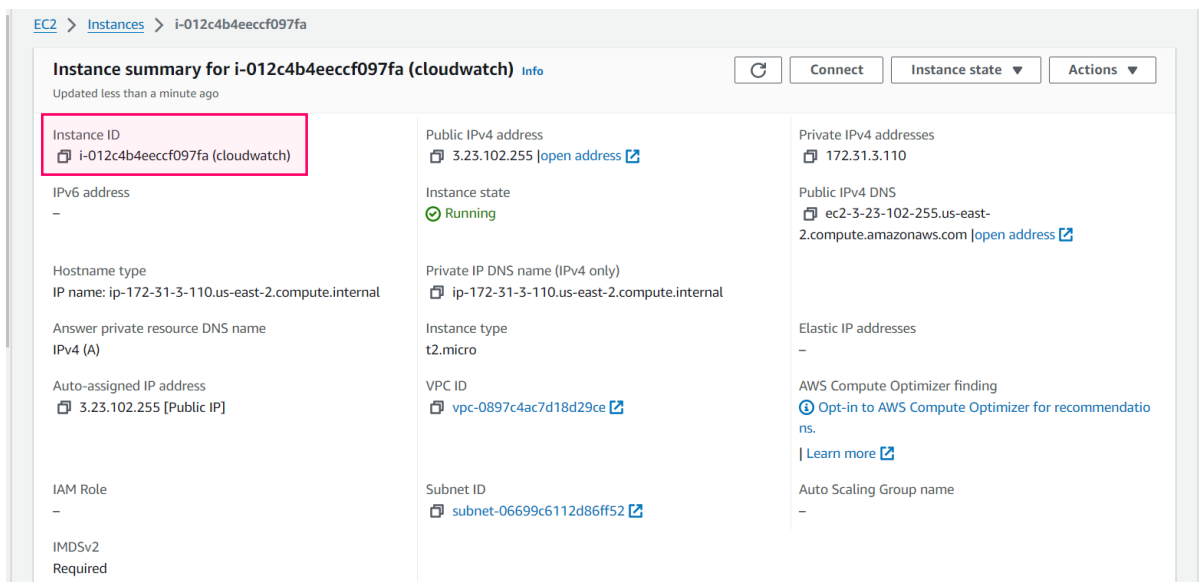
1. For performing this practical we need one public instance



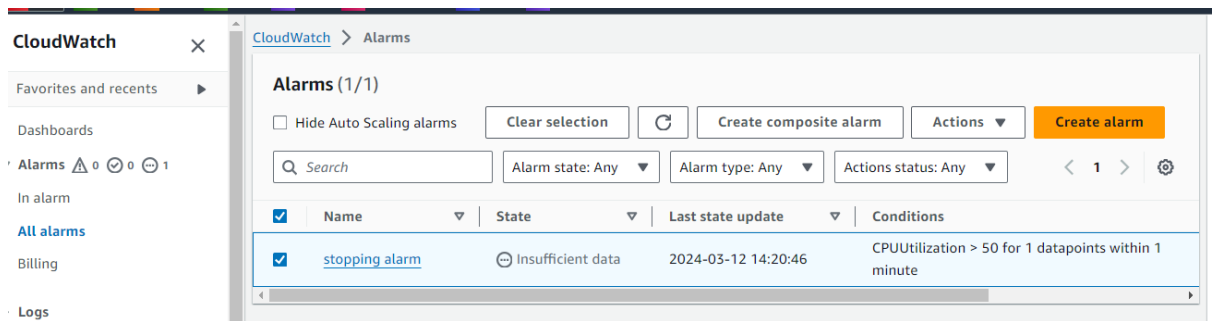
2. Instance Created successfully....



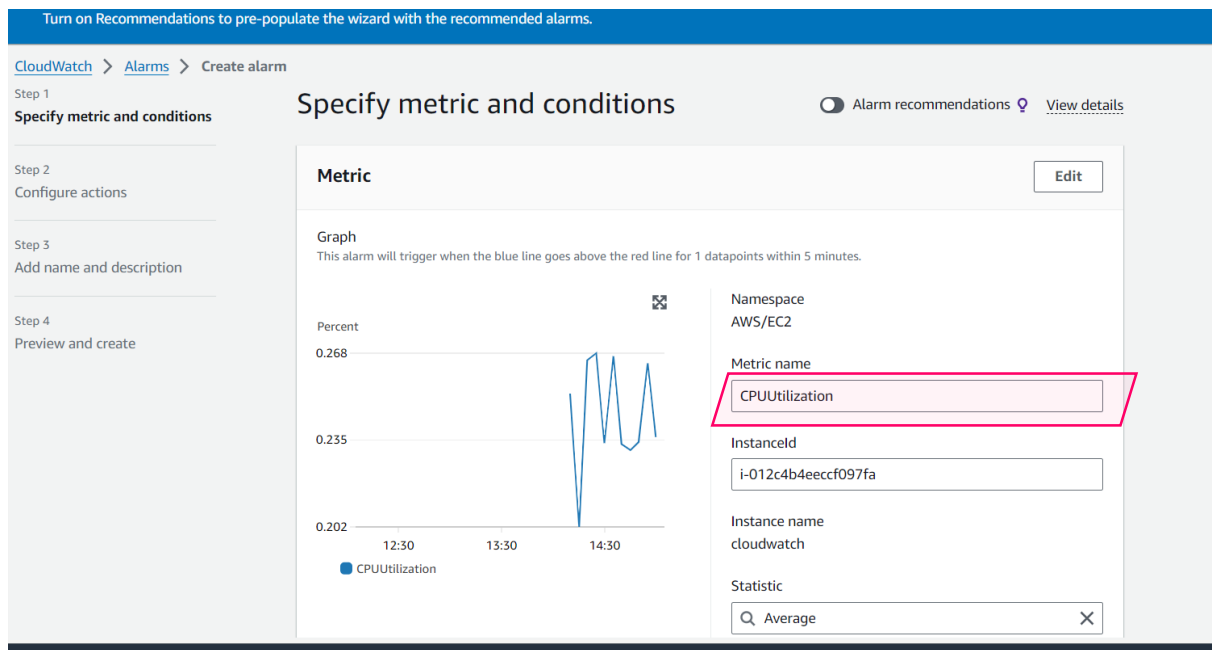
3. Copy instance ID



4. go to cloudwatch service and Under alarm option click on **Create alarm**



5. select the **CPUUtilization** Option



6. Add condition as per your requirement....

Conditions

Threshold type

☒ **Static**
Use a value as a threshold

☐ **Anomaly detection**
Use a band as a threshold

Whenever CPUUtilization is...
Define the alarm condition.

☒ **Greater**
> threshold

☐ **Greater/Equal**
>= threshold

☐ **Lower/Equal**
<= threshold

☐ **Lower**
< threshold

than...
Define the threshold value.

Must be a number

► Additional configuration

7. Click on **Add Ec2 Action**

Auto Scaling action

Add Auto Scaling action

EC2 action

Add EC2 action

Systems Manager action [Info](#)

This action will create an Incident or OpsItem in Systems Manager when the alarm is **In alarm** state.

Add Systems Manager action

Cancel

Previous

Next

8. Give alarm as per your choice and click on next....

The screenshot shows the 'Add name and description' step in the AWS CloudWatch console. On the left, a sidebar lists four steps: Step 1 (Specify metric and conditions), Step 2 (Configure actions), Step 3 (Add name and description), and Step 4 (Preview and create). Step 3 is currently selected. The main area is titled 'Add name and description' and contains a form. The 'Alarm name' field has 'cloudwatch' entered. Below it, the 'Alarm description - optional' field is shown with a 'View formatting guidelines' link. The description field has a preview of markdown formatting: '# This is an H1', '**double asterisks will produce strong character**', and 'This is [an example](https://example.com/) inline link.' A character count at the bottom of the description field shows 'Up to 1024 characters (0/1024)'. At the bottom right of the form, there are 'Cancel', 'Previous', and 'Next' buttons.

9. Summary....

The screenshot shows the 'Step 2: Configure actions' step in the AWS CloudWatch console. A pink box highlights the condition: 'Whenever CPUUtilization is Greater (>) than... 50'. Below this, there is an 'Additional configuration' section. The main area is titled 'Step 2: Configure actions' and contains an 'Edit' button. Under the 'Actions' section, an 'EC2 action' is configured: 'When In alarm, stop this instance (Instance ID: i-012c4b4eccc097fa)'. The bottom of the console shows a 'Next' button.

10. Click on Create alarm

Step 2: Configure actions Edit

Actions

No actions
You don't have any actions for this alarm.

Step 3: Add name and description Edit

Name and description

Name
cloudwatch

Description
-

Cancel Previous **Create alarm**

11. After Performing configuration connect to the instance and give Load using **stress** command....

```
aws | Services | Search [Alt+S]
EC2 IAM S3 EFS VPC CloudWatch

Usage: stress [OPTION [ARG]] ...
  -?, --help            show this help statement
  --version             show version statement
  -v, --verbose         be verbose
  -q, --quiet           be quiet
  -n, --dry-run         show what would have been done
  -t, --timeout N       timeout after N seconds
  --backoff N          wait factor of N microseconds before work starts
  -c, --cpu N           spawn N workers spinning on sqrt()
  -i, --io N            spawn N workers spinning on sync()
  -m, --vm N            spawn N workers spinning on malloc()/free()
  --vm-bytes B          malloc B bytes per vm worker (default is 256MB)
  --vm-stride B         touch a byte every B bytes (default is 4096)
  --vm-hang N           sleep N secs before free (default none, 0 is inf)
  --vm-keep             redirty memory instead of freeing and reallocating
  -d, --hdd N           spawn N workers spinning on write()/unlink()
  --hdd-bytes B         write B bytes per hdd worker (default is 1GB)

Example: stress --cpu 8 --io 4 --vm 2 --vm-bytes 128M --timeout 10s

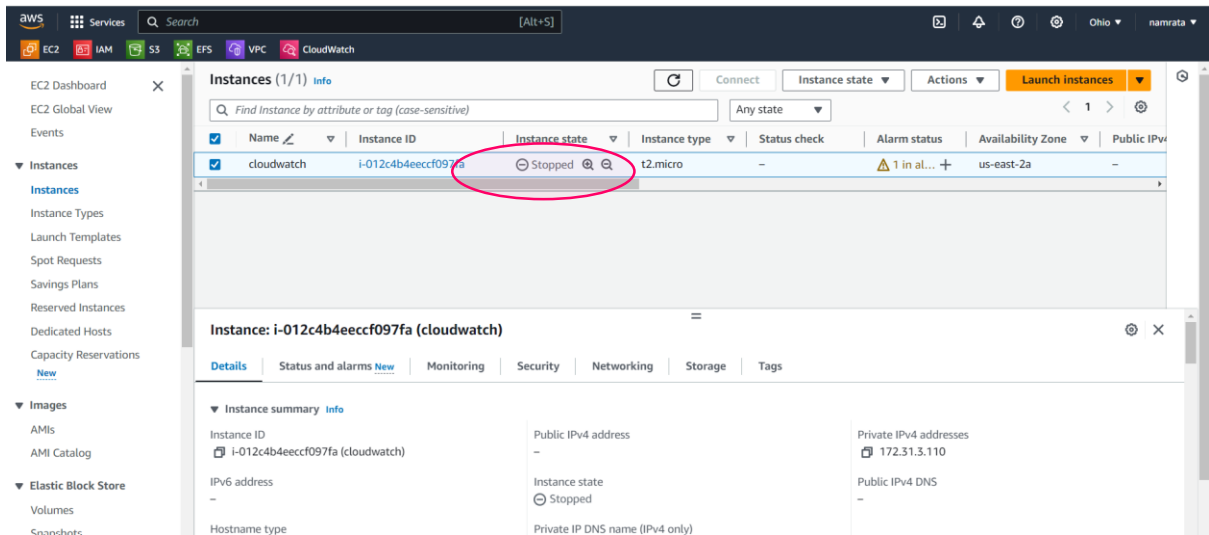
Note: Numbers may be suffixed with s,m,h,d,y (time) or B,K,M,G (size).
[root@ip-172-31-3-110 ec2-user]# ^C
[root@ip-172-31-3-110 ec2-user]# stress --cpu 88 --io 4 --vm 2 --vm-bytes 128M --timeout 10m &
[1] 27267
[root@ip-172-31-3-110 ec2-user]# stress: info: [27267] dispatching hogs: 88 cpu, 4 io, 2 vm, 0 hdd
```

12.Instance is terminated because we apply greater than 50% load.....

```
aws
Services
Search [Alt+S]
EC2 IAM S3 EFS VPC CloudWatch
27472 root 20 0 3512 112 0 D 0.8 0.0 0:00.34 stress
27493 root 20 0 3512 112 0 R 0.8 0.0 0:01.02 stress
27494 root 20 0 3512 112 0 R 0.8 0.0 0:01.02 stress
27268 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27270 root 20 0 134588 15072 264 R 0.5 1.6 0:03.16 stress
27271 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27274 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27276 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27278 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27279 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27280 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27281 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27282 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27283 root 20 0 3512 112 0 R 0.5 0.0 0:03.17 stress
27284 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27285 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
27287 root 20 0 3512 112 0 R 0.5 0.0 0:03.16 stress
Broadcast message from root@localhost (Tue 2024-03-12 15:25:18 UTC):
The system will power off now!
Broadcast message from root@localhost (Tue 2024-03-12 15:25:18 UTC):
The system will power off now!
```

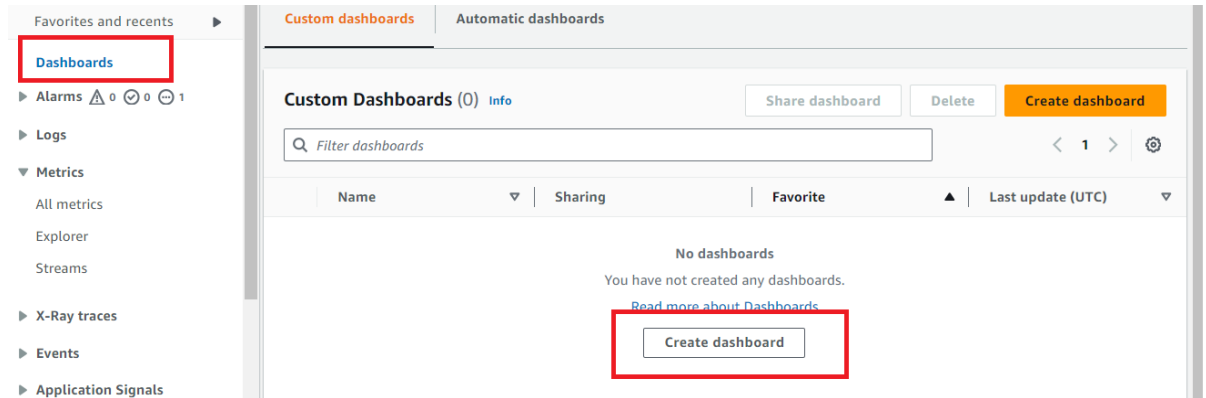
#Instance is terminated because we apply condition...
(Above 50% cpu =stop instance)

13.Instance is terminated successfully....

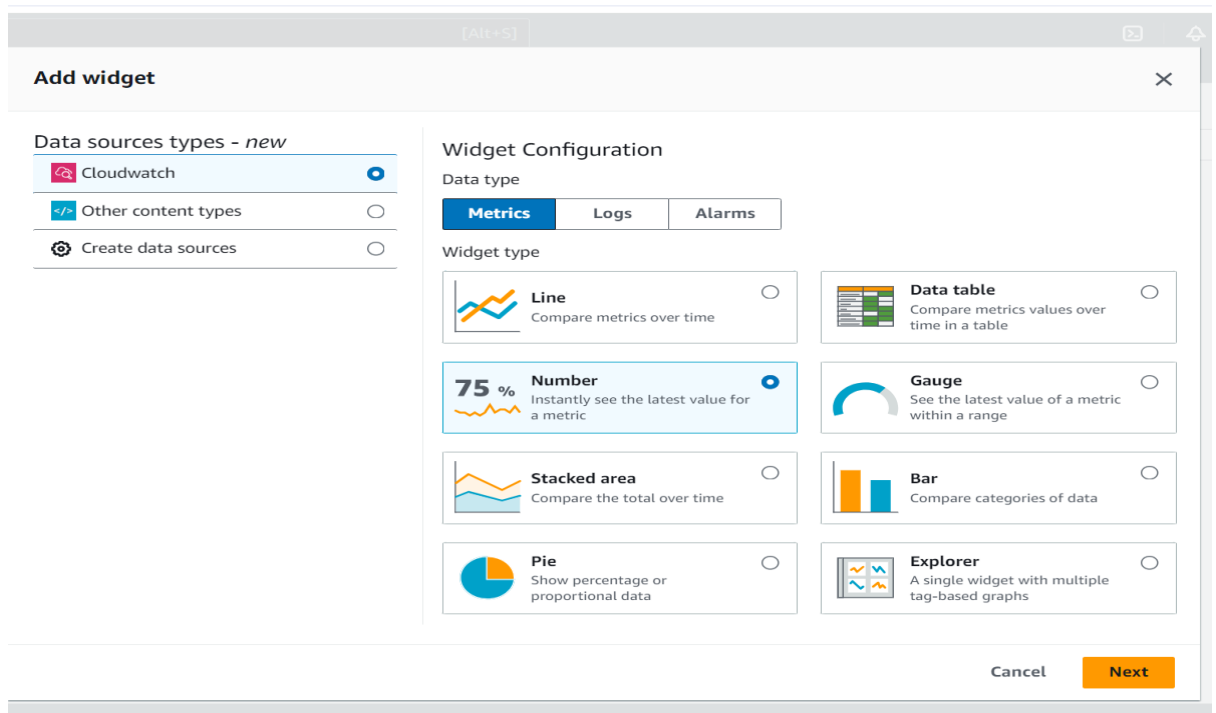


Creating Custom dashboard:-

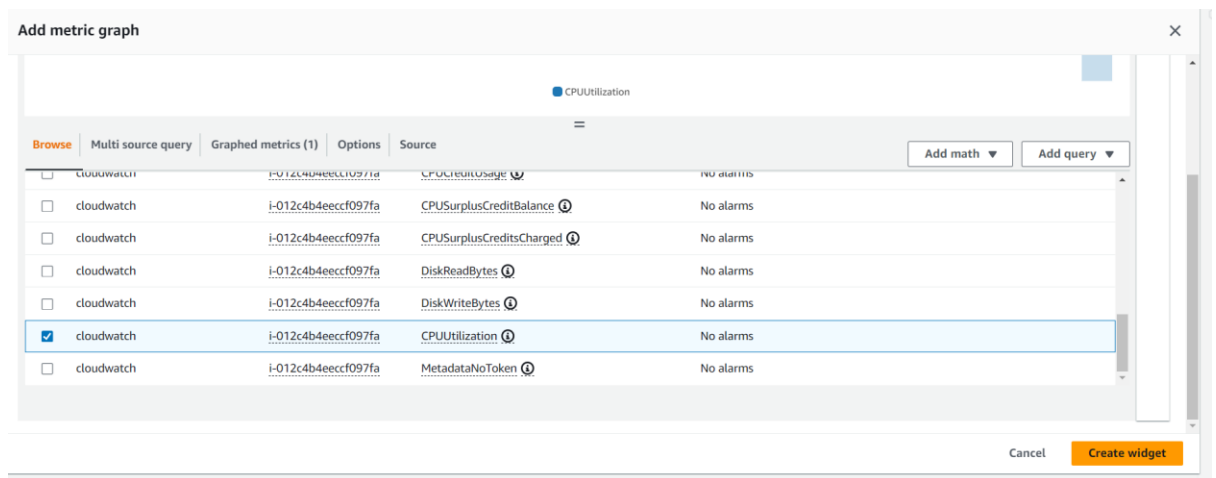
1. Click on Create dashboard Option



2. Select Widget type == Numbers



3. Select the Matrix as per your requirement and click on **Create widget** option



4. Dashboard Created successfully....

