

CloudWatch

CloudWatch service is monitoring your cloud resources. It helps you collect and track metrics, monitor logs, set alarms, and automate actions based on predefined rules. With CloudWatch, you can keep an eye on the performance, availability, and health of your AWS infrastructure and applications, ensuring they run smoothly and efficiently.

Features

Metric Monitoring: CloudWatch allows users to collect and track metrics from various AWS services in real time. These metrics could include CPU utilization, network traffic, disk I/O, and many others. Users can visualize this data through graphs and charts, enabling them to understand resource usage patterns and identify performance bottlenecks.

Log Monitoring: CloudWatch Logs enables users to centralize, monitor, and analyze log data from their AWS resources and applications. By collecting log files in one place, users can easily search, filter, and analyze log events to troubleshoot issues, track changes, and gain insights into system behavior.

Alarms: CloudWatch Alarms enable users to set up notifications based on predefined thresholds for their metrics. When a metric breaches a specified threshold, CloudWatch triggers an alarm, notifying users via various channels such as email, SMS, or Amazon SNS (Simple Notification Service). Alarms can be configured to trigger automated actions, such as scaling AWS resources or executing AWS Lambda functions, helping users respond to incidents promptly.

Dashboards: CloudWatch Dashboards allow users to create customizable dashboards to visualize their metrics data. Users can create multiple dashboards with different sets of metrics, enabling them to monitor the health and performance of their AWS resources and applications at a glance.

Events: CloudWatch Events enables users to respond to changes in their AWS environment by triggering automated actions based on predefined rules. Users can create event-driven workflows to automate tasks such as starting or stopping EC2 instances, resizing RDS databases, or invoking AWS Lambda functions in response to specific events.

Benefits

Proactive Monitoring: CloudWatch enables proactive monitoring of AWS resources and applications, allowing users to detect and address issues before they impact performance or availability.

Operational Efficiency: By centralizing monitoring and automation tasks within CloudWatch, users can streamline operations, reduce manual intervention, and improve resource utilization.

Insights and Analysis: CloudWatch provides valuable insights into resource usage, system behavior, and application performance through metrics visualization, log analysis, and anomaly detection.

Scalability and Flexibility: CloudWatch is designed to scale with the growing needs of AWS customers, supporting large-scale deployments and diverse use cases across various industries and sectors.

CloudWatch Dashboard

In CloudWatch Service, AWS provides an Automatic Dashboard and a Custom Dashboard.

- **Automatic Dashboard**

CloudWatch > Dashboards

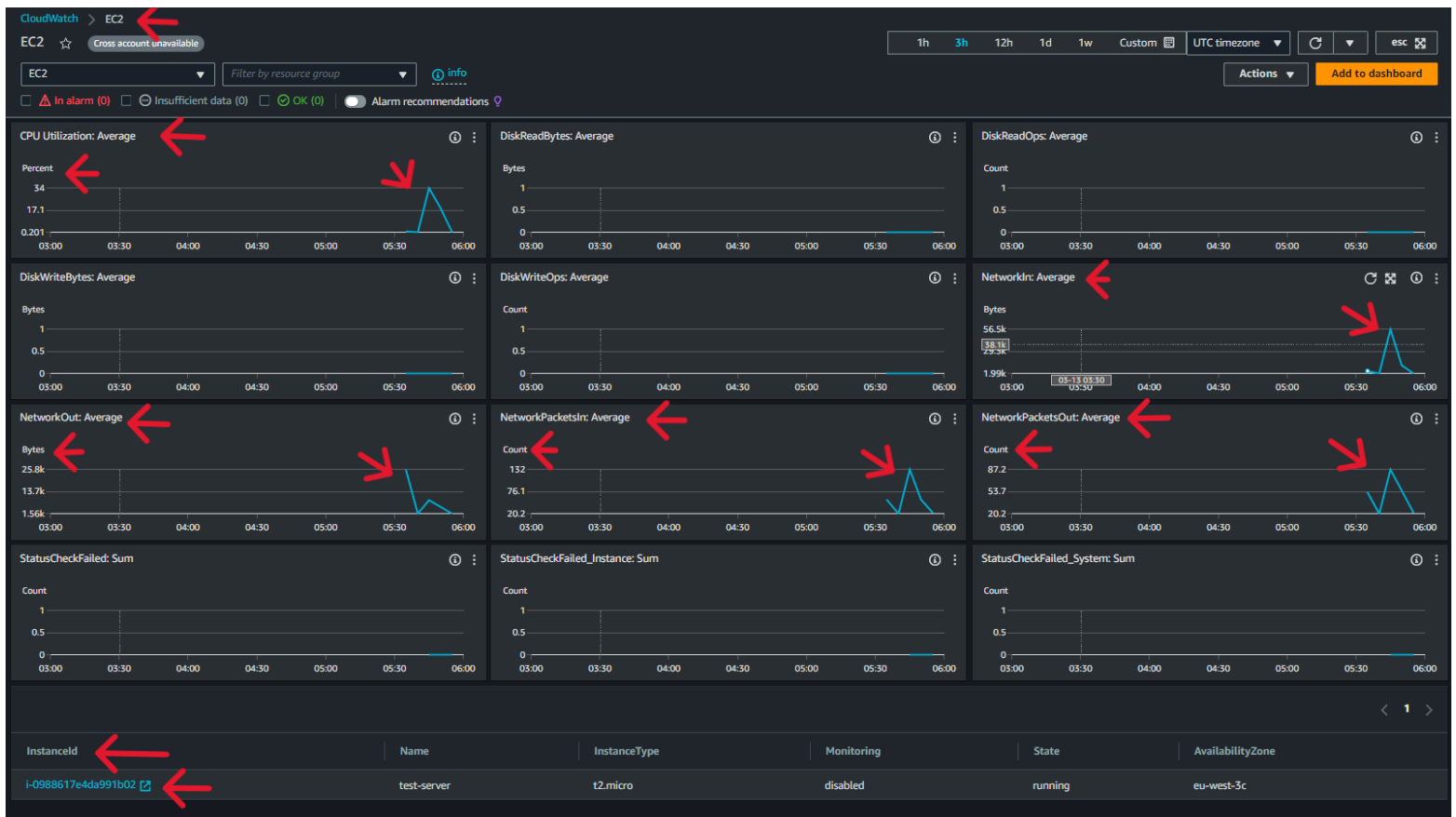
Custom dashboards Automatic dashboards

Automatic dashboards (7) Info

Filter dashboards Filter by resource group

Name	In alarm	Favorite
Elastic Block Store (EBS)		☆
EC2		☆
CloudWatch Events		☆
CloudWatch Logs		☆
CloudWatch Logs Subscriptions		☆
Simple Notification Service		☆
CloudWatch Usage		☆

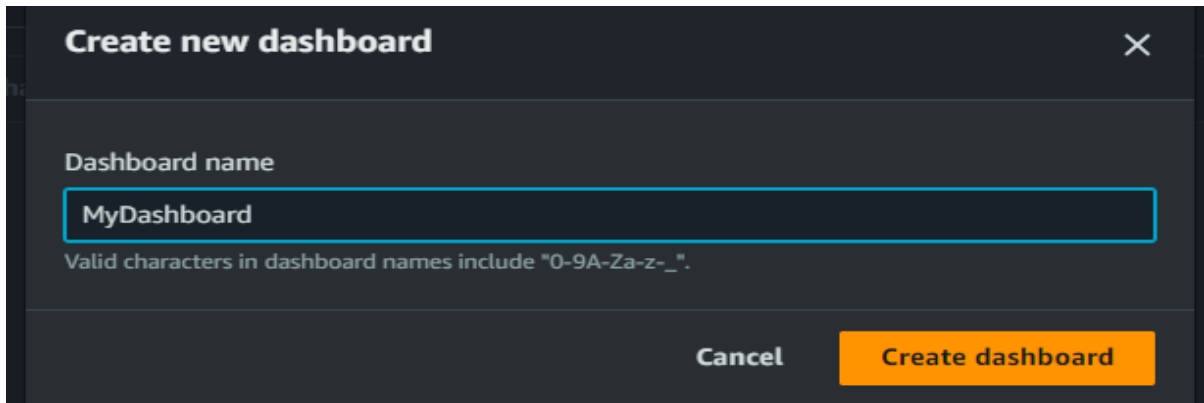
- **Automatic Dashboard for EC2**



CloudWatch Automatic Dashboards provide instant visualizations of your AWS resources' performance metrics without manual setup. They enable you to easily monitor your infrastructure's health and quickly address any issues or anomalies (unusual or unexpected events or occurrences)

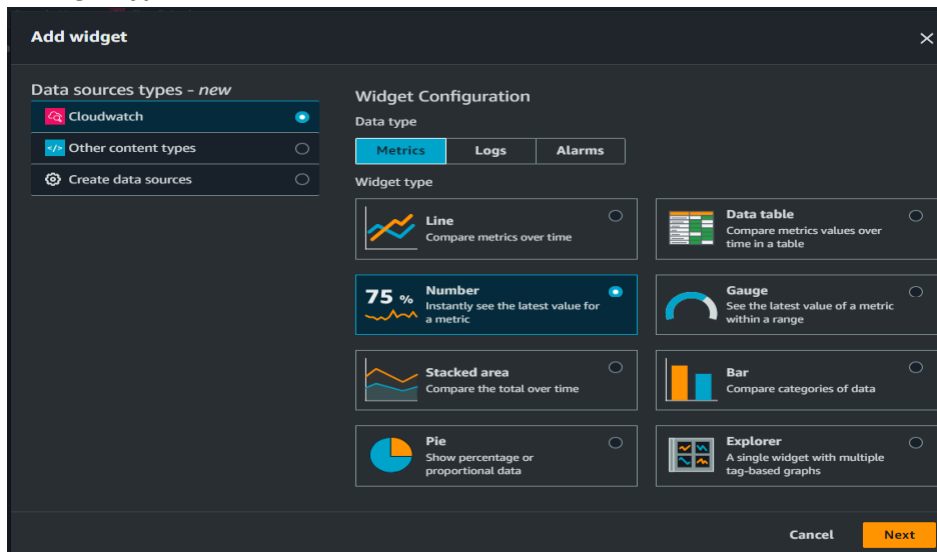
Now we create a Custom Dashboard.

- 1) Go to the Custom Dashboard option and click on the Create Dashboard Button.
- 2) Name the dashboard and click on the Create Dashboard button.



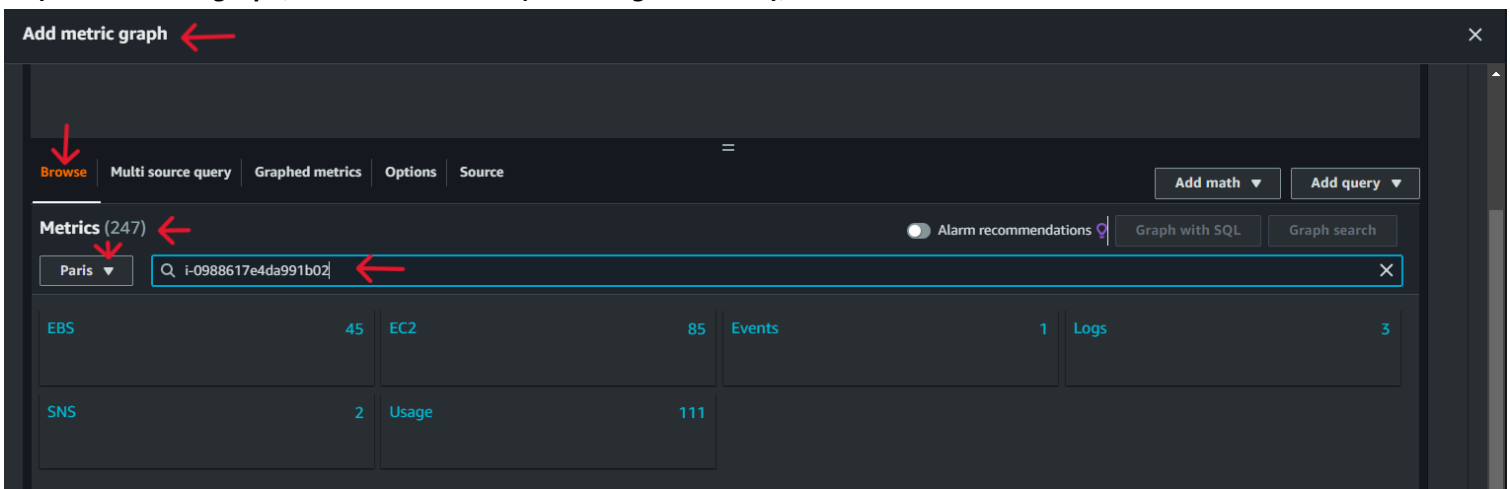
The 'Create new dashboard' dialog box has a title bar with a close button. It contains a 'Dashboard name' label and a text input field with 'MyDashboard'. Below the input field is a note: 'Valid characters in dashboard names include "0-9A-Za-z-_"'. At the bottom are 'Cancel' and 'Create dashboard' buttons.

- 3) Add widget, Select widget type, and click on the next button.



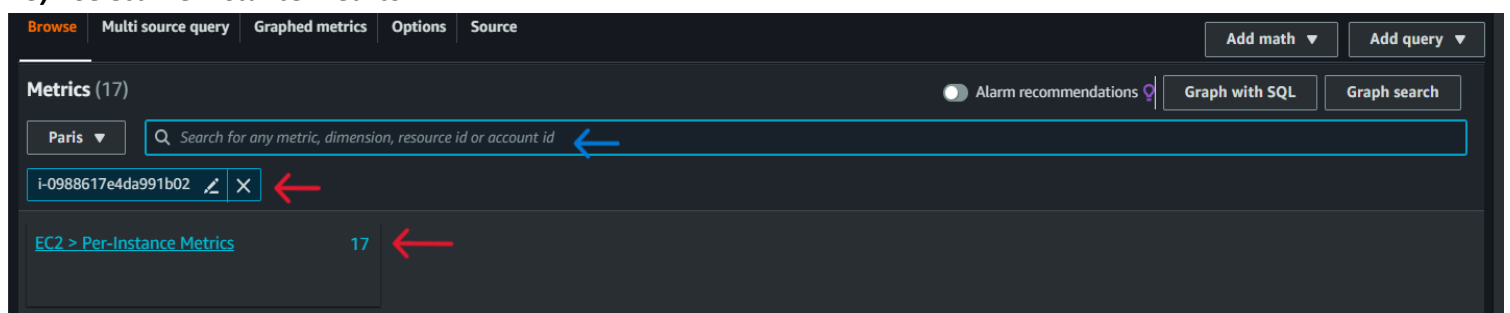
The 'Add widget' dialog box is divided into two main sections. The left section, 'Data sources types - new', lists 'Cloudwatch' (selected), 'Other content types', and 'Create data sources'. The right section, 'Widget Configuration', has tabs for 'Data type' (Metrics, Logs, Alarms) and 'Widget type'. Under 'Widget type', there are eight options: Line, Data table, Number (selected), Gauge, Stacked area, Bar, Pie, and Explorer. Each option has a brief description. At the bottom right are 'Cancel' and 'Next' buttons.

- 4) Add metric graph, enter Resource ID (now we go with ec2), and enter.



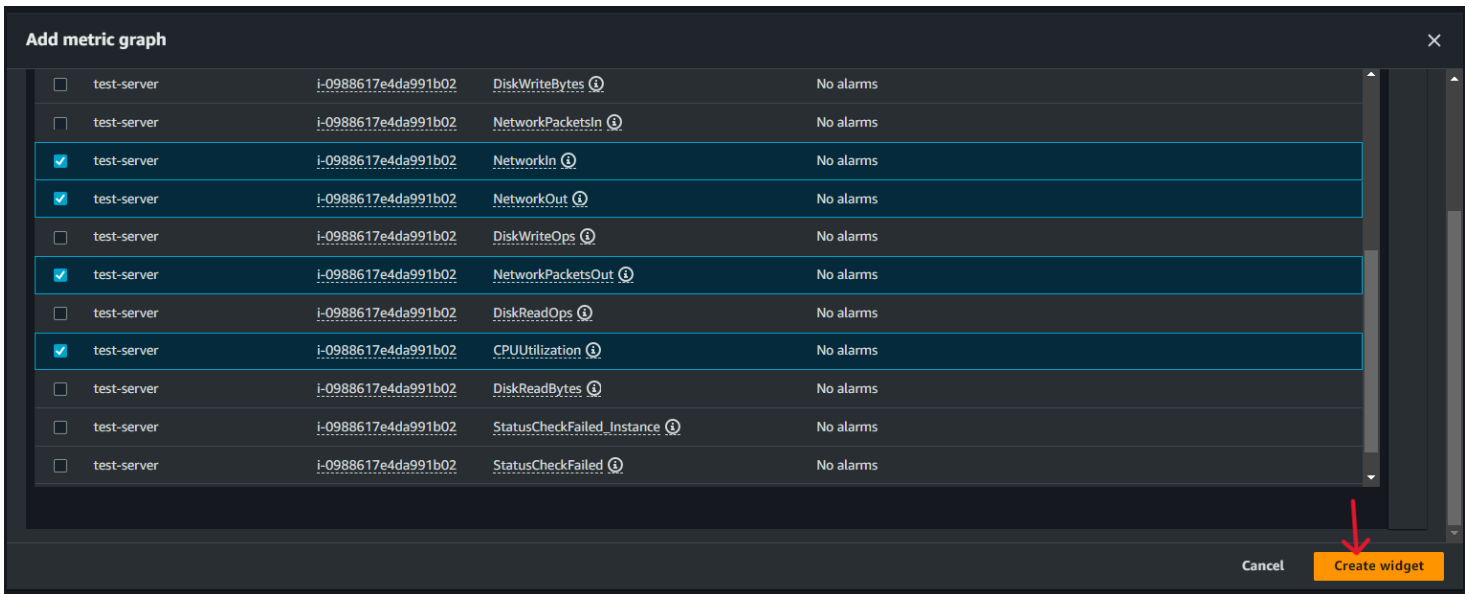
The 'Add metric graph' dialog box has a title bar with a close button. It features a 'Browse' button and tabs for 'Multi source query', 'Graphed metrics', 'Options', and 'Source'. Below the tabs is a search bar with 'Paris' selected and a search icon. A table lists metrics for 'Paris' with columns for metric name, value, and unit. The table has two rows: EBS (45) and EC2 (85) in the first row, and Events (1) and Logs (3) in the second row. Below the table are two more rows: SNS (2) and Usage (111). At the bottom right are 'Add math' and 'Add query' buttons.

- 5) Select Pre-Instance Metrics.

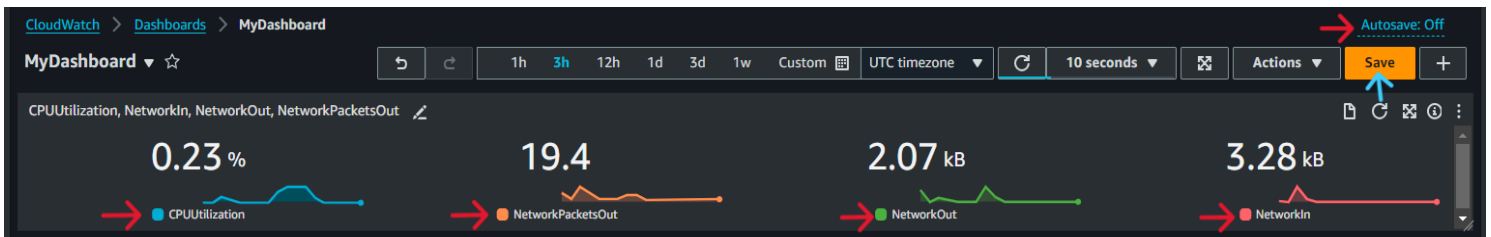


The 'Add metric graph' dialog box shows search results for 'Paris'. The search bar contains 'Search for any metric, dimension, resource id or account id'. Below the search bar is a table with two rows: 'EC2 > Per-Instance Metrics' (17) and 'Usage' (111). At the bottom right are 'Add math' and 'Add query' buttons.

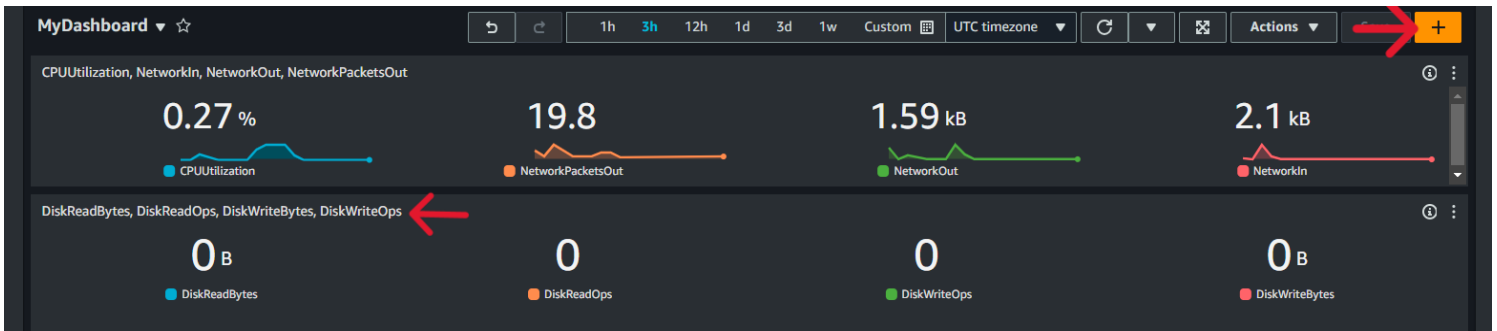
6) Select the Metrics that you want to monitor in your custom dashboard, and click on the Create Widget button.



7) Now our custom dashboard is created, and the Widget of Ec2 Instance is added to it, save it.

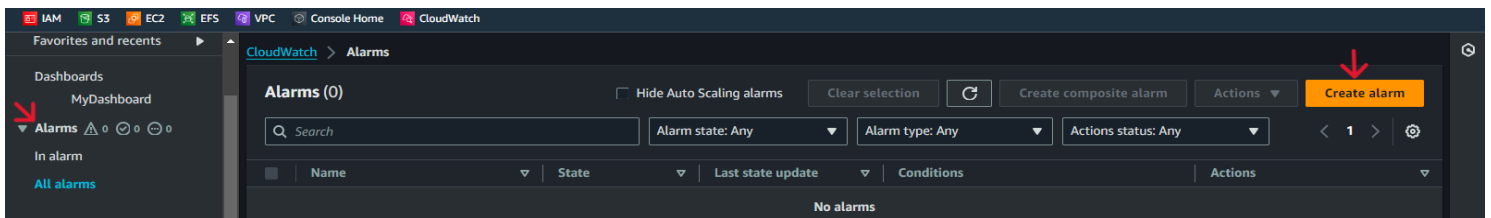


8) Now we can add more Widgets of other resources in our custom dashboard to monitor their metrics, logs, and alarms.

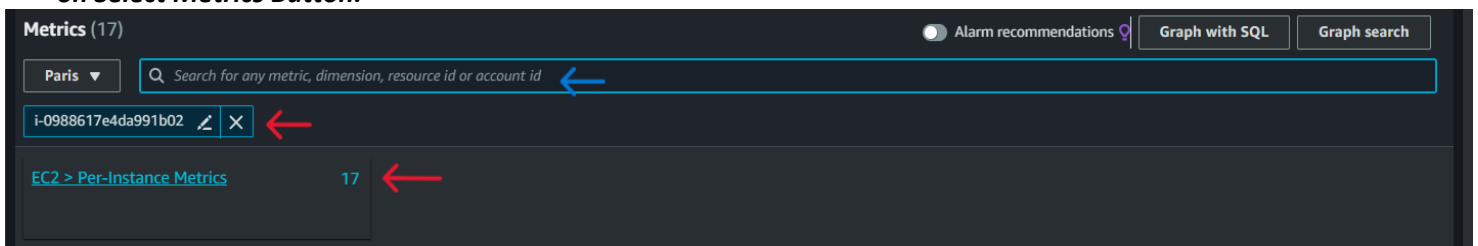


Now we create an Alarm

1) Click on Create Alarm in CloudWatch Service.



2) Click on Add Metrics, Enter Resource ID, Select Pre-Metrics, Pick Metrics CPU utilization (or more as needed), then click on Select Metrics Button.



3) Specify metrics.

Specify metric and conditions

Alarm recommendations View details

Metric Edit

Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.

Percent

99.8

50

0.199

05:30 06:30 07:30

CPUUtilization

Namespace
AWS/EC2

Metric name
CPUUtilization

InstanceId
i-0988617e4da991b02

Instance name
test-server

Statistic
Average

Period
5 minutes

4) Select condition. (The threshold value is 50%.)

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.

50

Must be a number

► **Additional configuration**

5) Configure what action is performed if the threshold is greater than 50%. Now we select Ec2.

EC2 action

Alarm state trigger

Define the alarm state that will trigger this action.

☒ **In alarm**
The metric or expression is outside of the defined threshold.

☐ **OK**
The metric or expression is within the defined threshold.

☐ **Insufficient data**
The alarm has just started or not enough data is available.

Remove

Take the following action...

Define what will happen to the EC2 instance with the Instance ID i-0988617e4da991b02 when this alarm is triggered.

☐ **Recover this instance**
You can only recover certain EC2 instance types. [See documentation](#)

☒ **Stop this instance**
You can only stop an instance if it is backed by an EBS volume. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

☐ **Terminate this instance**
You will not be able to terminate this instance if termination protection is enabled. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

☐ **Reboot this instance**
An instance reboot is equivalent to an operating system reboot. AWS will use the existing Service Linked Role (AWSServiceRoleForCloudWatchEvents) to perform this action. [Show IAM policy document](#)

Add EC2 action

6) Name the alarm, and click on the next button.

Add name and description

Name and description

Alarm name

Stop-server

Alarm description - optional

[View formatting guidelines](#)

Edit

Preview

This is an H1

double asterisks will produce strong character

This is [an example](https://example.com/) inline link.

Up to 1024 characters (0/1024)

7) Preview the alarm, and click on the create alarm button, our alarm will created.

Favorites and recents

Dashboards

MyDashboard

Alarms 0 1 0

In alarm

All alarms

Alarms (1/1)

Alarm state: Any

Alarm type: Any

Actions status: Any

< 1 >

Clear selection

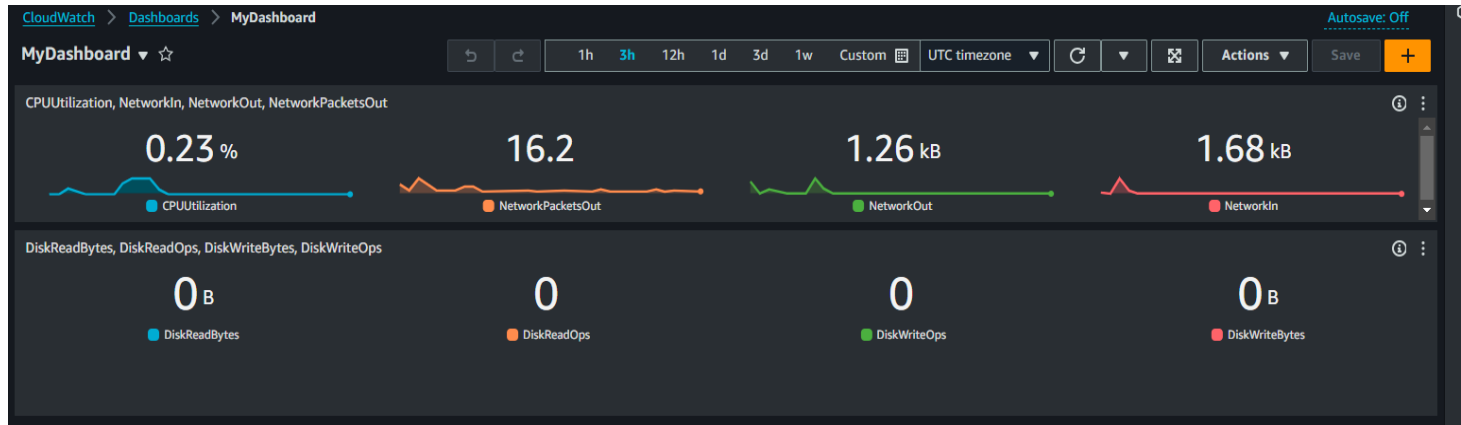
Create composite alarm

Actions

Create alarm

<input checked="" type="checkbox"/>	Name	State	Last state update	Conditions	Actions
<input checked="" type="checkbox"/>	Stop-server	OK	2024-03-13 08:22:47	CPUUtilization > 50 for 1 datapoints within 5 minutes	Actions enabled

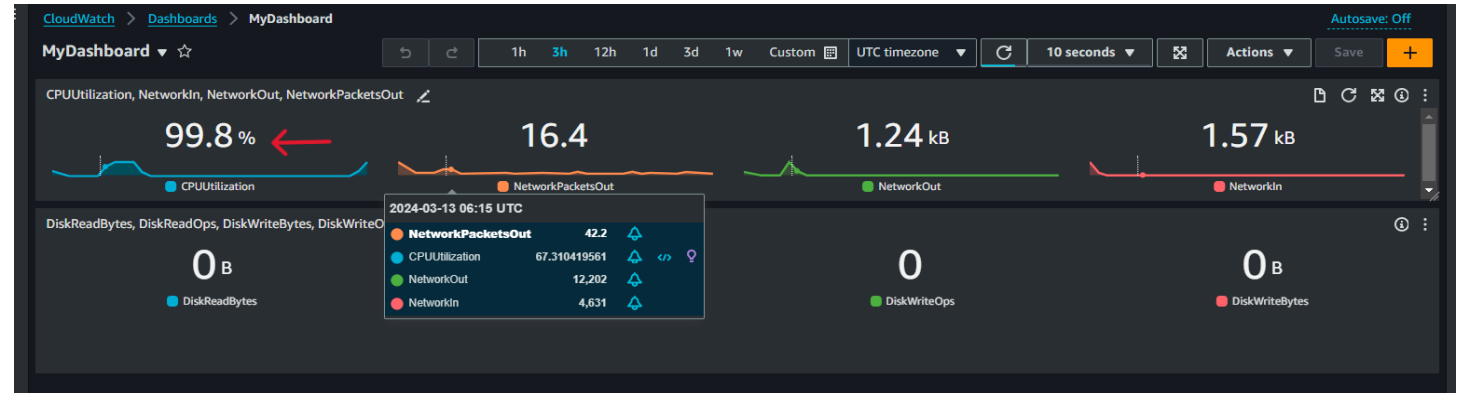
8) Now go the custom dashboard and check the Metrics of our Ec2 Instance.



We can see CPU utilization is at optimal status. So here we will add artificial stress on the CPU to check that there is an alarm trigger and stop the Instance.

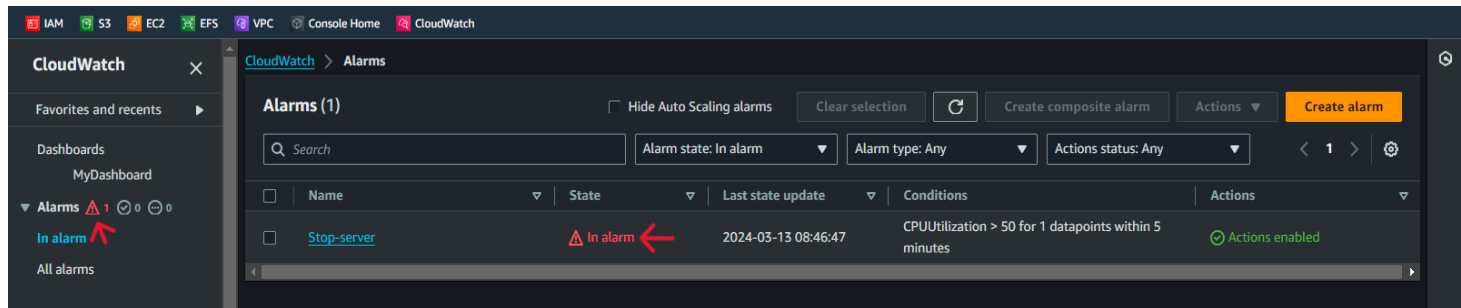
9) Install the Stress tool into the Instance and increase the CPU utilization artificially.

10) Now check the CPU utilization.



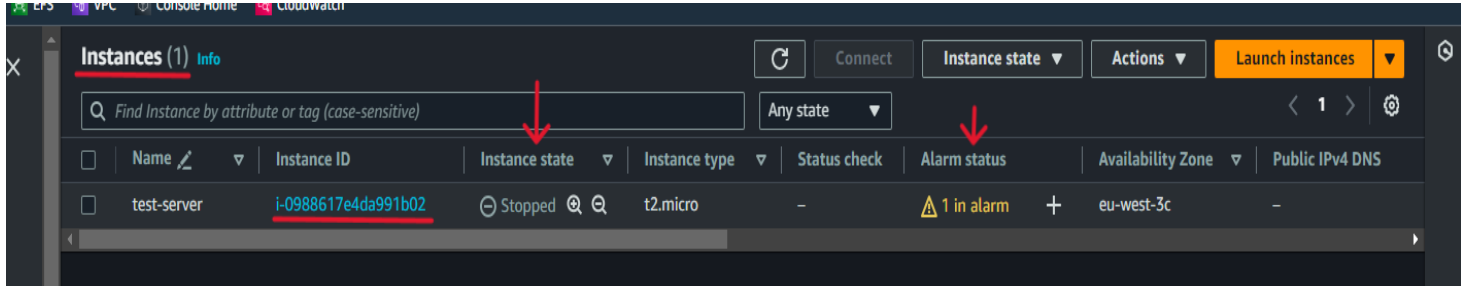
Is artificially increased

11) Now check the alarm status.



The Alarm Status is “In Alarm” which means the alarm has been triggered.

12) Now check the Instance status.



Here we can see our Instance is stopped which means our alarm function is effectively working.