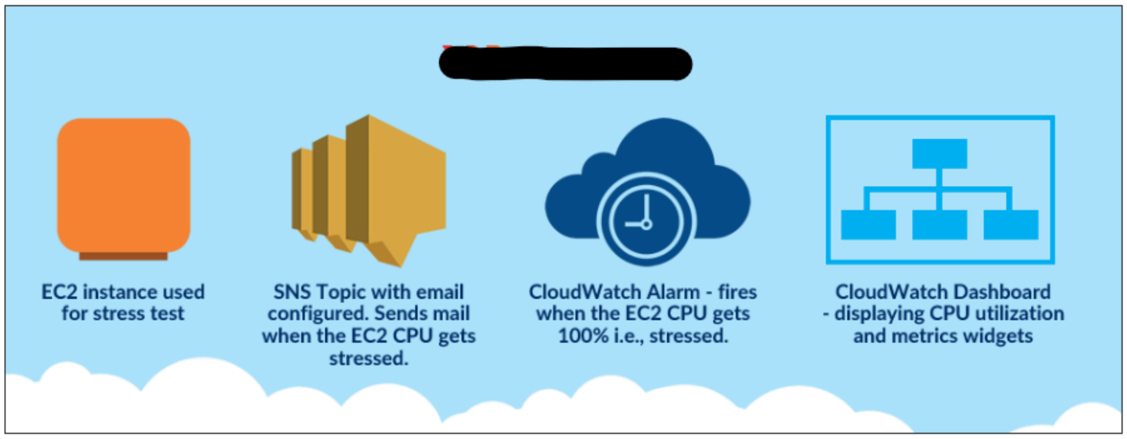
Architecture Diagram


[redacted]
EC2 instance used for stress test
SNS Topic with email configured. Sends mail when the EC2 CPU gets stressed.
CloudWatch Alarm - fires when the EC2 CPU gets 100% i.e., stressed.
CloudWatch Dashboard - displaying CPU utilization and metrics widgets

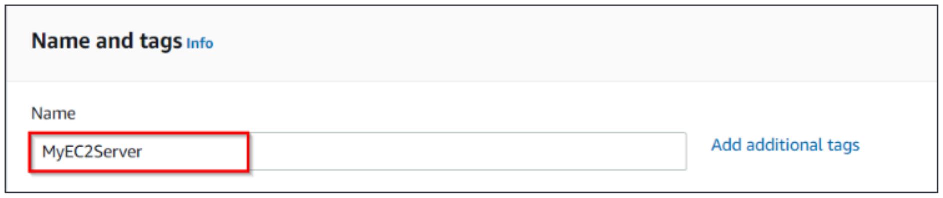
Task Details

1. Sign into the AWS Management Console.
2. Create an EC2 Instance.
3. SSH into EC2 Instance and install necessary Softwares.
4. Create SNS Topic.
5. Subscribe to an SNS Topic.
6. Check EC2 CPU Utilization Metrics in CloudWatch Metrics.
7. Create CloudWatch Alarm.
8. Testing CloudWatch Alarm by Stressing CPU Utilization.
9. Checking For an Email from the SNS Topic.
10. Checking the CloudWatch Alarm Graph.
11. Create a CloudWatch Dashboard.
12. Validation of the lab.

Task 2: Launching an EC2 Instance

In this task, we are going to launch an EC2 Instance that will be used for checking various features in CloudWatch.

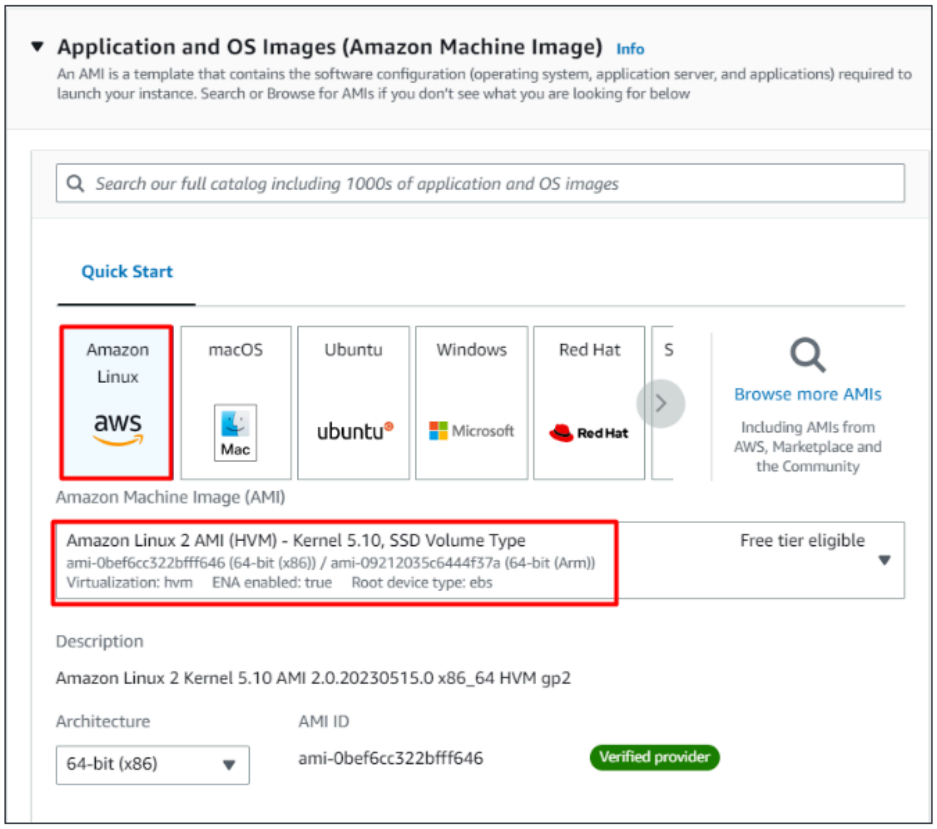
1. Make sure you are in the **N.Virginia** Region.
2. Navigate to **EC2** by clicking on the **Services** menu in the top, then click on **EC2** in the **Compute** section.
3. Navigate to **Instances** from the left side menu and click on **Launch instances** button.
4. Name : Enter **MyEC2Server**



Name: MyEC2Server

5. **For Amazon Machine Image (AMI):** Select **Amazon Linux** and the select **Amazon Linux 2 AMI** from the drop-down.

   Note: if there are two AMI's present for Amazon Linux 2 AMI, choose any of them.


▼ 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
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-0bef6cc322bfff646 (64-bit (x86)) / ami-09212035c6444f37a (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20230515.0 x86_64 HVM gp2

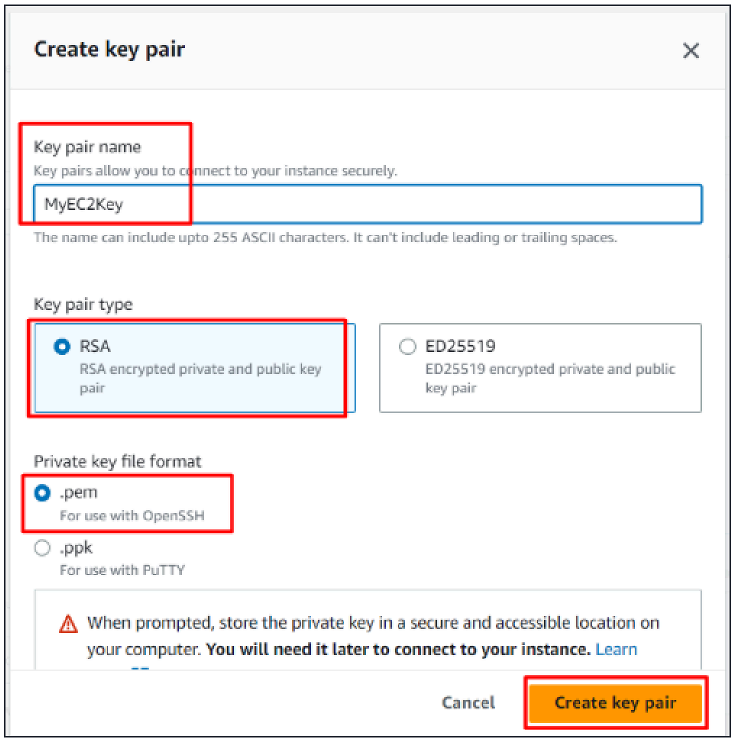
Architecture: 64-bit (x86)

AMI ID: ami-0bef6cc322bfff646 Verified provider

6. For Instance Type: Select **t2.micro**
7. **For Key pair:** Select **Create a new key pair** Button

- Key pair name: **MyEC2Key**
- Key pair type: **RSA**
- Private key file format: **.pem**

8. Select **Create key pair** Button.



Create key pair

Key pair name
Key pairs allow you to connect to your instance securely.
MyEC2Key
The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type
[✓ RSA] RSA encrypted private and public key pair
ED25519 ED25519 encrypted private and public key pair

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** Learn

9. In Network Settings Click on **Edit** button:

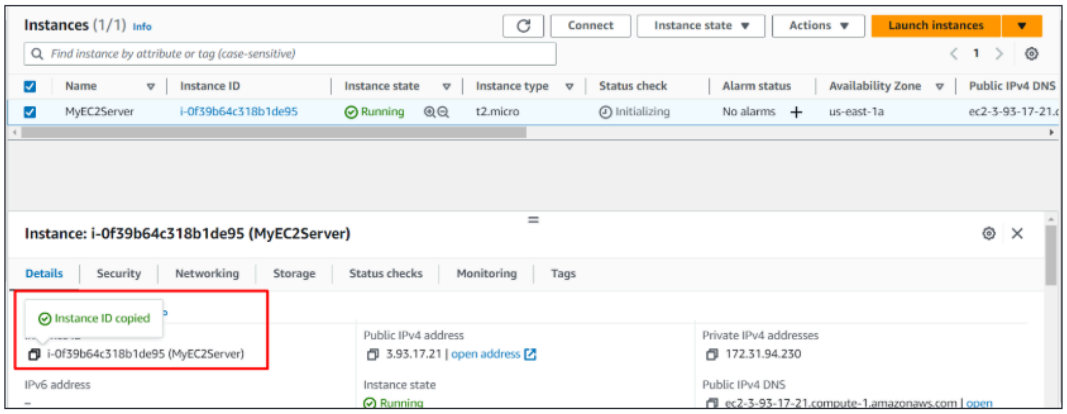
- Auto-assign public IP: **Enable**
- Select **Create new Security group**
- Security group name : Enter **MyEC2Server_SG**
- Description : Enter **Security Group to allow traffic to EC2**
- To add **SSH** :
    - Choose Type: Select **SSH**
    - Source: Select **Anywhere**

10. Keep Rest the things as Default and Click on **Launch Instance** Button.

11. Select **View all Instances** to View the Instance you created.

12. **Launch Status:** Your instance is now launching. Click on the instance ID and wait for complete initialization of the instance (until the status changes to running).

Note: Select the instance and Copy the Instance-ID and save it for later, we need to search the metrics in CloudWatch based on this.



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
MyEC2Server	i-0f39b64c318b1de95	Running	t2.micro	Initializing	No alarms	us-east-1a	ec2-3-93-17-21.

Instance: i-0f39b64c318b1de95 (MyEC2Server)

i-0f39b64c318b1de95 (MyEC2Server)

Public IPv4 address
3.93.17.21 | open address

Private IPv4 addresses
172.31.94.230

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-3-93-17-21.compute-1.amazonaws.com | open

Task 3 : SSH into EC2 Instance and install necessary Softwares

1. Folow the instructions provided in /labs/support-document/ssh-into-ec-instance to SSH into the EC2 instance you created.
2. Once you are logged into the EC2 instance, switch to root user.

```
sudo su
```

3. Update :

```
yum update -y
```

4. Stress Tool : Amazon Linux 2 AMI does not have the stress tool installed by default, we will need to install some packages

```
sudo amazon-linux-extras install epel -y
```

```
yum install stress -y
```

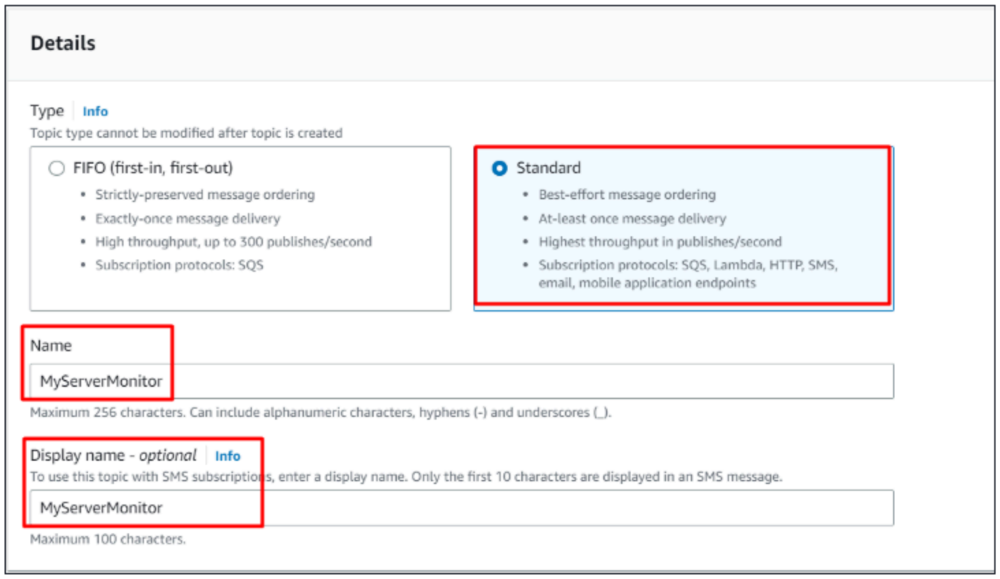
5. Stress tool will be used for simulating EC2 metrics. Once we create the CloudWatch Alarm, we shall come back to SSH and trigger **CPUUtilization** using it.

Task 4: Create SNS Topic

In this task, we are going to create a SNS Topic.

1. Make sure you are in the **N.Virginia** Region.
2. Navigate to **Simple Notification Service** by clicking on the **Services** menu available under the **Application Integration** section.
3. Click on **Topics** in the left panel and then click on **Create topic** button.
4. Under **Details**:

- Type: Select **Standard**
- Name: Enter **MyServerMonitor**
- Display name: Enter **MyServerMonitor**



Details

Type Info
Topic type cannot be modified after topic is created

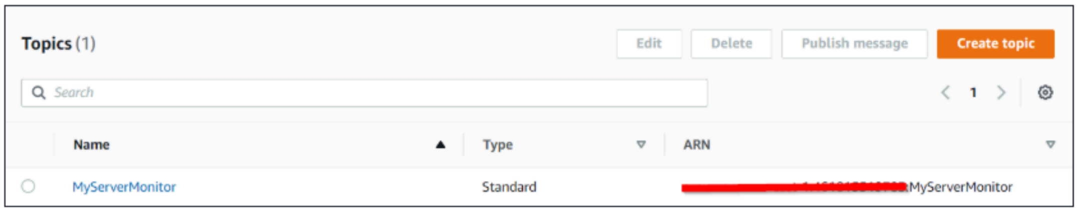
FIFO (first-in, first-out)
- Strictly-preserved message ordering
- Exactly-once message delivery
- High throughput, up to 300 publishes/second
- Subscription protocols: SQS

[✓ 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
MyServerMonitor
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

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.
MyServerMonitor
Maximum 100 characters.

5. Leave other options as default and click on **Create topic** button. A SNS topic will be created.

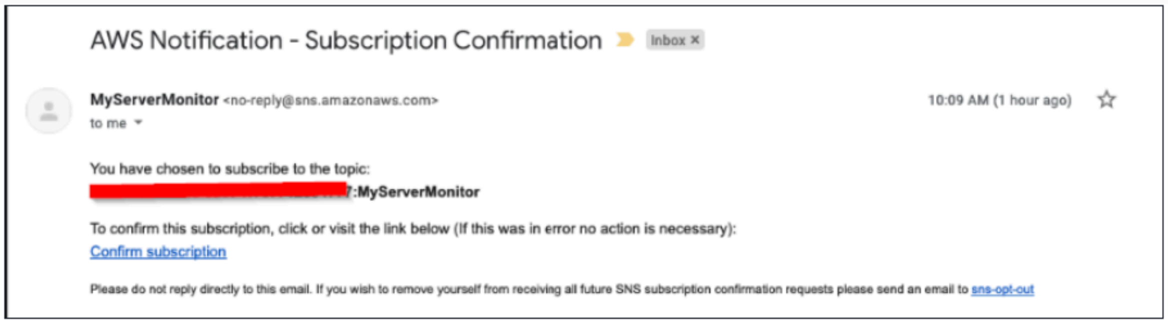


Name	Type	ARN
MyServerMonitor	Standard	[redacted]:MyServerMonitor

Task 5: Subscribe to an SNS Topic

1. Once SNS topic is created, click on SNS topic **MyServerMonitor.**
2. Click on **Create subscription** button.
3. Under Details:

- Protocol : Select **Email**
- Endpoint : Enter your email address
- **Note:** Make sure you give proper email address as this is where your notification will be delivered.


AWS Notification - Subscription Confirmation

MyServerMonitor <no-reply@sns.amazonaws.com>
to me

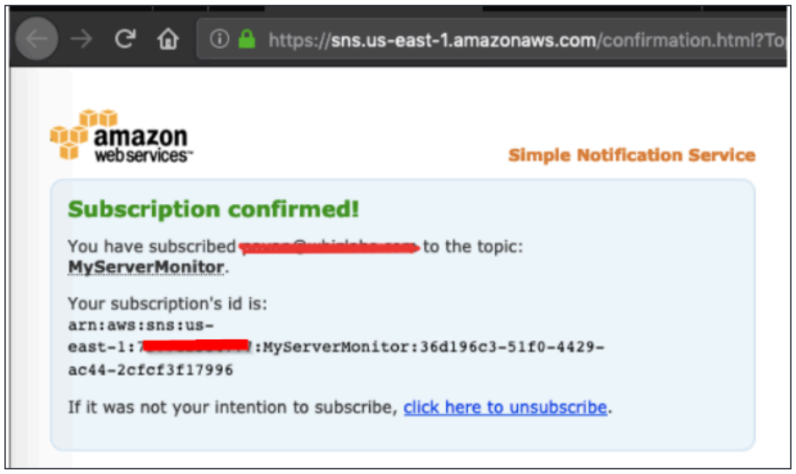
10:09 AM (1 hour ago)

You have chosen to subscribe to the topic:
[redacted]:MyServerMonitor

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):
Confirm subscription

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to sns-opt-out

5. Click on **Confirm subscription**.


https://sns.us-east-1.amazonaws.com/confirmation.html?To

amazon web services

Simple Notification Service

Subscription confirmed!

You have subscribed [redacted] to the topic:
MyServerMonitor.

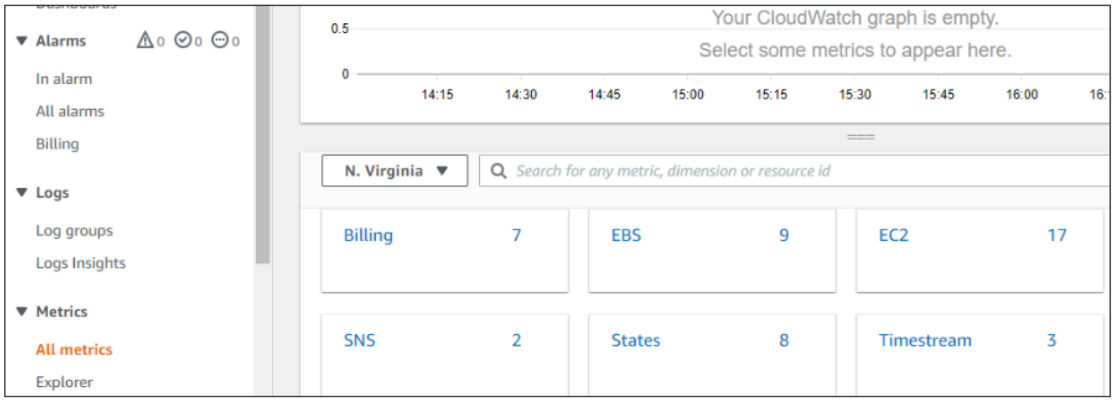
Your subscription's id is:
arn:aws:sns:us-east-1:[redacted]:MyServerMonitor:36d196c3-51f0-4429-ac44-2cfcf3f17996

If it was not your intention to subscribe, click here to unsubscribe.

6. Your email address is now subscribed to SNS Topic **MyServerMonitor**.

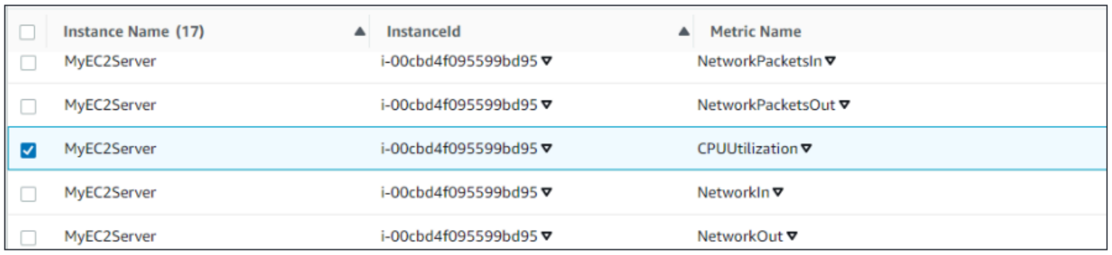
Task 6: Using CloudWatch to Check EC2 CPU Utilization Metrics in CloudWatch Metrics

1. Navigate to **CloudWatch** by clicking on the **Services** menu available under the **Management & Governance** section.
2. Click on **All metrics** under **Metrics** in the Left Panel.
3. You should be able to see **EC2** under **All Metrics.** If EC2 is not visible, please wait for 5-10 minutes, CloudWatch usually takes around 5-10 minutes after the creation of EC2 to start fetching metric details.



4. Click on **EC2**. Select **Per-Instance Metrics.**

5. Here you can see various metrics. Select the CPUUtilization metric to see the graph.



☐	Instance Name (17) ▲	InstanceId ▲	Metric Name
☐	MyEC2Server	i-00cbd4f095599bd95 ▼	NetworkPacketsIn ▼
☐	MyEC2Server	i-00cbd4f095599bd95 ▼	NetworkPacketsOut ▼
✓	MyEC2Server	i-00cbd4f095599bd95 ▼	CPUUtilization ▼
☐	MyEC2Server	i-00cbd4f095599bd95 ▼	NetworkIn ▼
☐	MyEC2Server	i-00cbd4f095599bd95 ▼	NetworkOut ▼

6. Now at the top of the screen, you can see the CPU Utilization graph (which is at zero since we have not stressed the CPU yet).

Task 7: Create CloudWatch Alarm

CloudWatch alarms are used to watch a single CloudWatch metric or the result of a math expression based on CloudWatch metrics.

1. Click on **In alarms** under **Alarms** in the left panel of the CloudWatch dashboard.
2. Click on **Create alarm** available on the top right corner.
3. In the **Specify metric and conditions** page:

    - Click on **Select metric**. It will open the **Select Metrics** page.
    - Scroll down and Select **EC2**.

- Select **Per-Instance Metrics**
- Enter your EC2 **Instance-ID** in the search bar to get metrics for **MyEC2Server**
- Choose the **CPU Utilization** metric.
- Click on **Select metric** button.

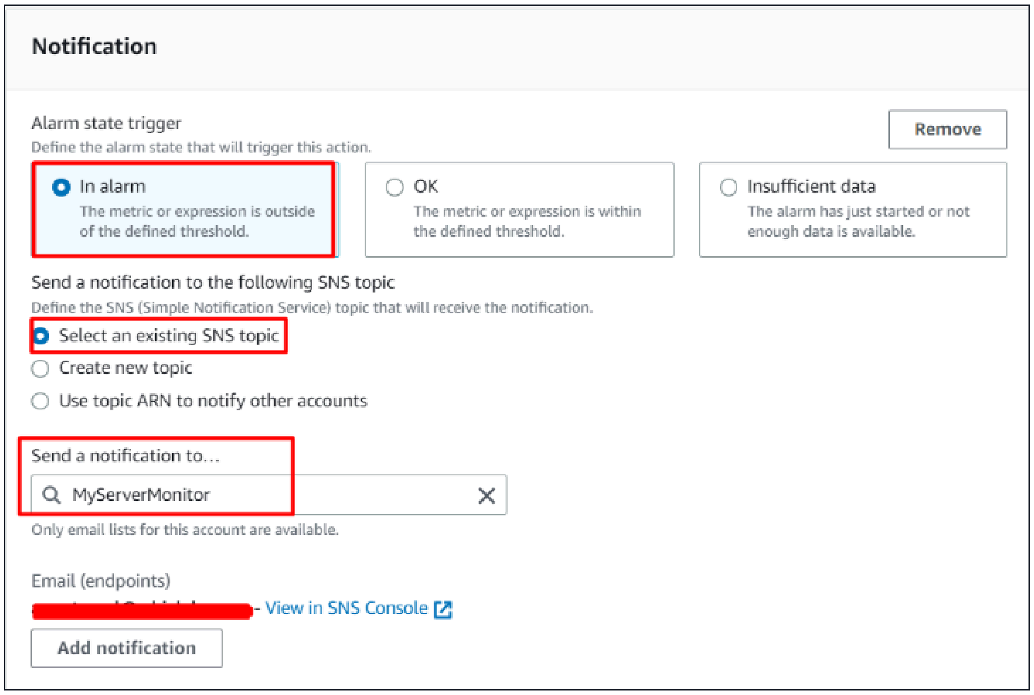
4. Now, configure the alarm with the following details:

- Under **Metrics**
    - Period: Select **1 Minute**
- Under **Conditions**
    - Threshold type: Choose **Static**
    - Whenever CPUUtilization is...: Choose **Greater**
    - than: Enter **30**
- Leave other values as **default** and click on **Next** button.

5. In **Configure actions** page:

- Under **Notification**
    - Alarm state trigger: Choose **In Alarm**

- Select an SNS topic: Choose **Select an existing SNS topic**
- Send a notification to... : Choose **MyServerMonitor** SNS topic which was created earlier.



Notification

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.

Send a notification to the following SNS topic
Define the SNS (Simple Notification Service) topic that will receive the notification.

[✓ Select an existing SNS topic]
Create new topic
Use topic ARN to notify other accounts

Send a notification to...
MyServerMonitor
Only email lists for this account are available.

Email (endpoints)
[redacted] - View in SNS Console

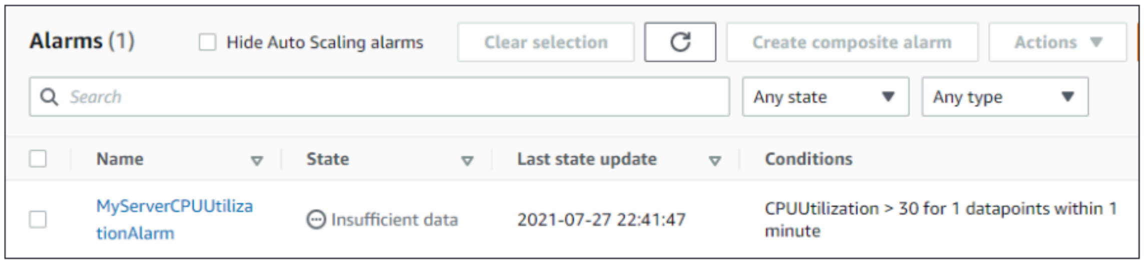
- Leave other fields as default. Click on **Next** button.

6. **In the Add a description** page, (under Name and Description):

- Name: Enter the Name **MyServerCPUUtilizationAlarm**
- Click on **Next** button.

7. A preview of the Alarm will be shown. Scroll down and click on **Create alarm** button.

8. A new CloudWatch Alarm is now created.



	Name	State	Last state update	Conditions
	MyServerCPUUtilizationAlarm	Insufficient data	2021-07-27 22:41:47	CPUUtilization > 30 for 1 datapoints within 1 minute

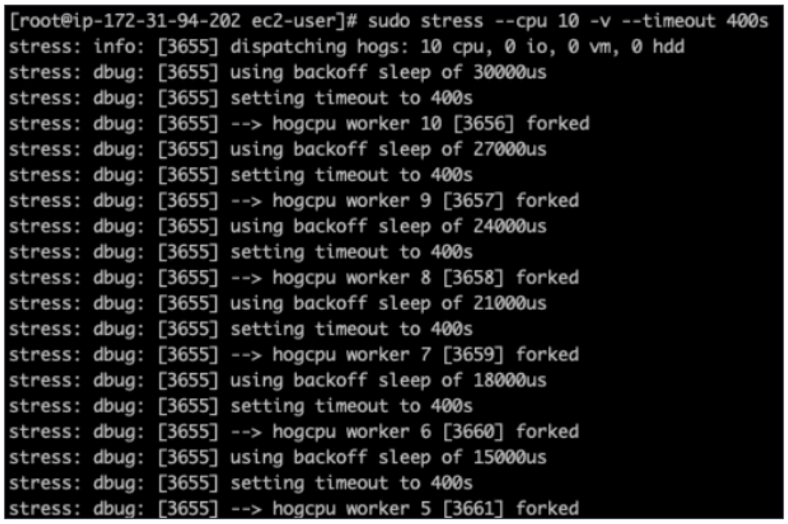
- Whenever the CPU Utilization goes above **30** for **more than 1 minute**, an SNS Notification will be triggered and you will receive an email.

Task 8: Testing CloudWatch Alarm by Stressing CPU Utilization

1. SSH back into the EC2 instance - **MyEC2Server**.
2. The stress tool has already been installed. Lets run a command to increase the CPU Utilization manually.

```
sudo stress --cpu 10 -v --timeout 400s
```

3. This command shall monitor the process created by the stress tool(which we triggered manually). It will run for **6 minutes and 40 seconds**. It will monitor CPU utilization, which should remain very near 100% for that amount of time.

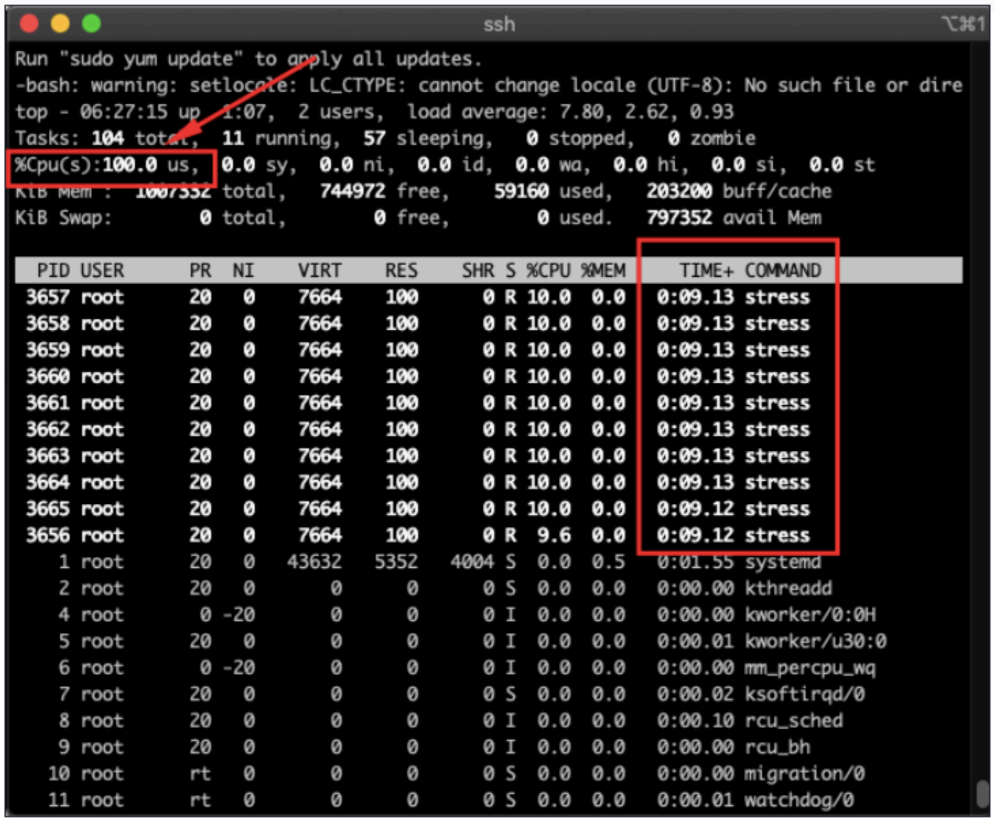


```
[root@ip-172-31-94-202 ec2-user]# sudo stress --cpu 10 -v --timeout 400s
stress: info: [3655] dispatching hogs: 10 cpu, 0 io, 0 vm, 0 hdd
stress: dbug: [3655] using backoff sleep of 30000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 10 [3656] forked
stress: dbug: [3655] using backoff sleep of 27000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 9 [3657] forked
stress: dbug: [3655] using backoff sleep of 24000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 8 [3658] forked
stress: dbug: [3655] using backoff sleep of 21000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 7 [3659] forked
stress: dbug: [3655] using backoff sleep of 18000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 6 [3660] forked
stress: dbug: [3655] using backoff sleep of 15000us
stress: dbug: [3655] setting timeout to 400s
stress: dbug: [3655] --> hogcpu worker 5 [3661] forked
```

4. Open another Terminal on your local machine and SSH back in EC2 instance - **MyEC2Server**.

5. Run this command to see the CPU utilization if you are a MAC or Linux User. For Windows User, you can navigate to Task manager.

```
top
```



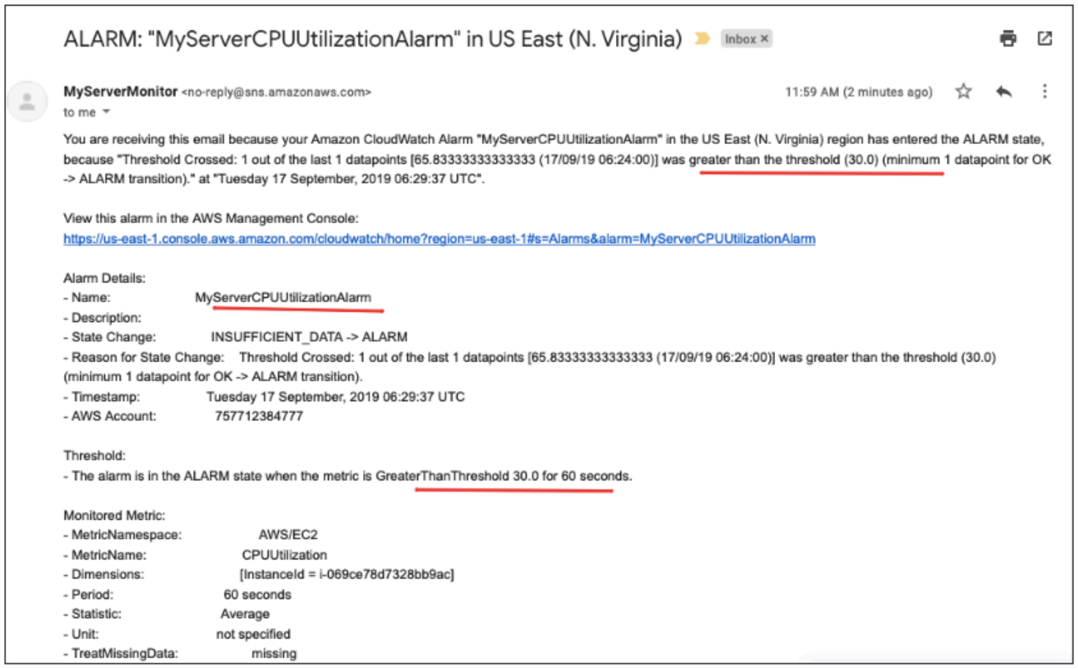
```
Run "sudo yum update" to apply all updates.
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or dire
top - 06:27:15 up 1:07,  2 users,  load average: 7.80, 2.62, 0.93
Tasks: 104 total,  11 running,  57 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1007332 total,   744972 free,    59160 used,   203200 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   797352 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 3657 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3658 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3659 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3660 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3661 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3662 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3663 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3664 root      20   0    7664    100      0 R 10.0  0.0   0:09.13 stress
 3665 root      20   0    7664    100      0 R 10.0  0.0   0:09.12 stress
 3656 root      20   0    7664    100      0 R  9.6  0.0   0:09.12 stress
    1 root      20   0   43632   5352   4004 S  0.0  0.5   0:01.55 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:0H
    5 root      20   0       0      0      0 I  0.0  0.0   0:00.01 kworker/u30:0
    6 root       0 -20       0      0      0 I  0.0  0.0   0:00.00 mm_percpu_wq
    7 root      20   0       0      0      0 S  0.0  0.0   0:00.02 ksoftirqd/0
    8 root      20   0       0      0      0 I  0.0  0.0   0:00.10 rcu_sched
    9 root      20   0       0      0      0 I  0.0  0.0   0:00.00 rcu_bh
   10 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.01 watchdog/0
```

6. You can now see that **%Cpu(s)** is **100**. By running this stress command, we have manually increased the CPU utilization of the EC2 Instance.
7. After 400 Seconds, the %Cpu will reduce back to **0**.

Task 9 : Checking For an Email from the SNS Topic

1. Navigate to your mailbox and refresh it. You should see a new email notification for **MyServerCPUUtilizationAlarm**.



ALARM: "MyServerCPUUtilizationAlarm" in US East (N. Virginia) Inbox ×

MyServerMonitor <no-reply@sns.amazonaws.com> 11:59 AM (2 minutes ago)
to me

You are receiving this email because your Amazon CloudWatch Alarm "MyServerCPUUtilizationAlarm" in the US East (N. Virginia) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [65.83333333333333 (17/09/19 06:24:00)] was greater than the threshold (30.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Tuesday 17 September, 2019 06:29:37 UTC".

View this alarm in the AWS Management Console:
https://us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#s=Alarms&alarm=MyServerCPUUtilizationAlarm

Alarm Details:
- Name: MyServerCPUUtilizationAlarm
- Description:
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [65.83333333333333 (17/09/19 06:24:00)] was greater than the threshold (30.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Tuesday 17 September, 2019 06:29:37 UTC
- AWS Account: 757712384777

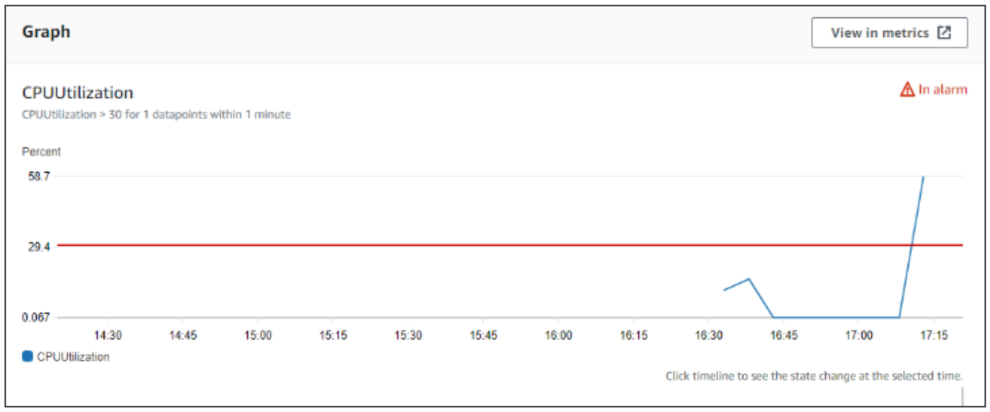
Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 30.0 for 60 seconds.

Monitored Metric:
- MetricNamespace: AWS/EC2
- MetricName: CPUUtilization
- Dimensions: [InstanceId = i-069ce78d7328bb9ac]
- Period: 60 seconds
- Statistic: Average
- Unit: not specified
- TreatMissingData: missing

2. We can see that mail we received contains details about our CloudWatch Alarm,(name of the alarm, when it was triggered, etc.).

Task 10: Checking the CloudWatch Alarm Graph

1. Navigate back to CloudWatch page, Click on Alarms.
2. Click on **MyServerCPUUtilizationAlarm.**
3. On the Graph, you can see places where CPUUtilization has gone above the 30% threshold.



4. We can trigger CPUUtilization multiple times to see the spike on the graph.
5. You have successfully triggered a CloudWatch Alarm for CPUUtilzation.

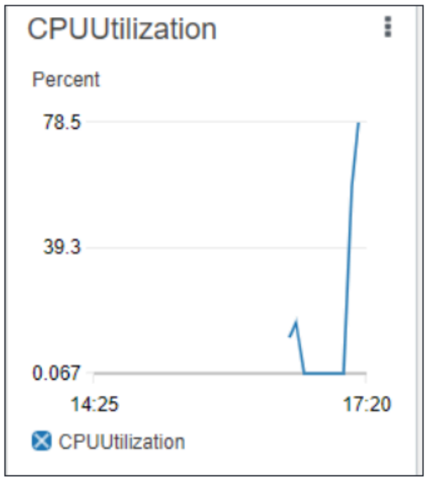
Task 11: Create a CloudWatch Dashboard

We can create a simple Cloudwatch dashboard to see the CPUUtilization and various other metric widgets.

1. Click on Dashboard in the left panel of the CloudWatch page.
2. Click on **Create dashboard** button.

- Dashboard name: Enter **MyEC2ServerDashboard**
- Click on **Create dashboard**
- Add widget: Select **Line** Graph.
- Click on **Next** button.
- Select **Metrics**. Click on **Next** button.
- On the next page, Choose **EC2** under the **Metrics** tab. Choose **Per-Instance Metrics.**
- In the search bar, **enter your EC2 Instance ID**. Select **CPUUtilization**.
- Click on **Create Widget** button.

3. Depending on how many times you triggered the stress command, you will see different spikes in the timeline.



4. Now click on the **Save** button.

5. You can also add multiple Widgets to the same Dashboard by clicking on **Add widget** button.

Do you know?

CloudWatch offers advanced features such as anomaly detection, which uses machine learning algorithms to automatically identify abnormal behavior in your metrics. This helps you to detect and investigate unusual patterns or potential performance bottlenecks in your resources.

Completion and Conclusion

1. You have created an EC2 Instance for which CloudWatch Monitoring will be carried out.
2. You have successfully created an Amazon SNS Topic used by CloudWatch.
3. You have successfully subscribed to SNS topic using your email address.
4. You have used CloudWatch to see CPUUtilization Metrics using CloudWatch Metrics.
5. You have successfully created and triggered a CloudWatch Alarm based on the CPUUtilzation Metric.