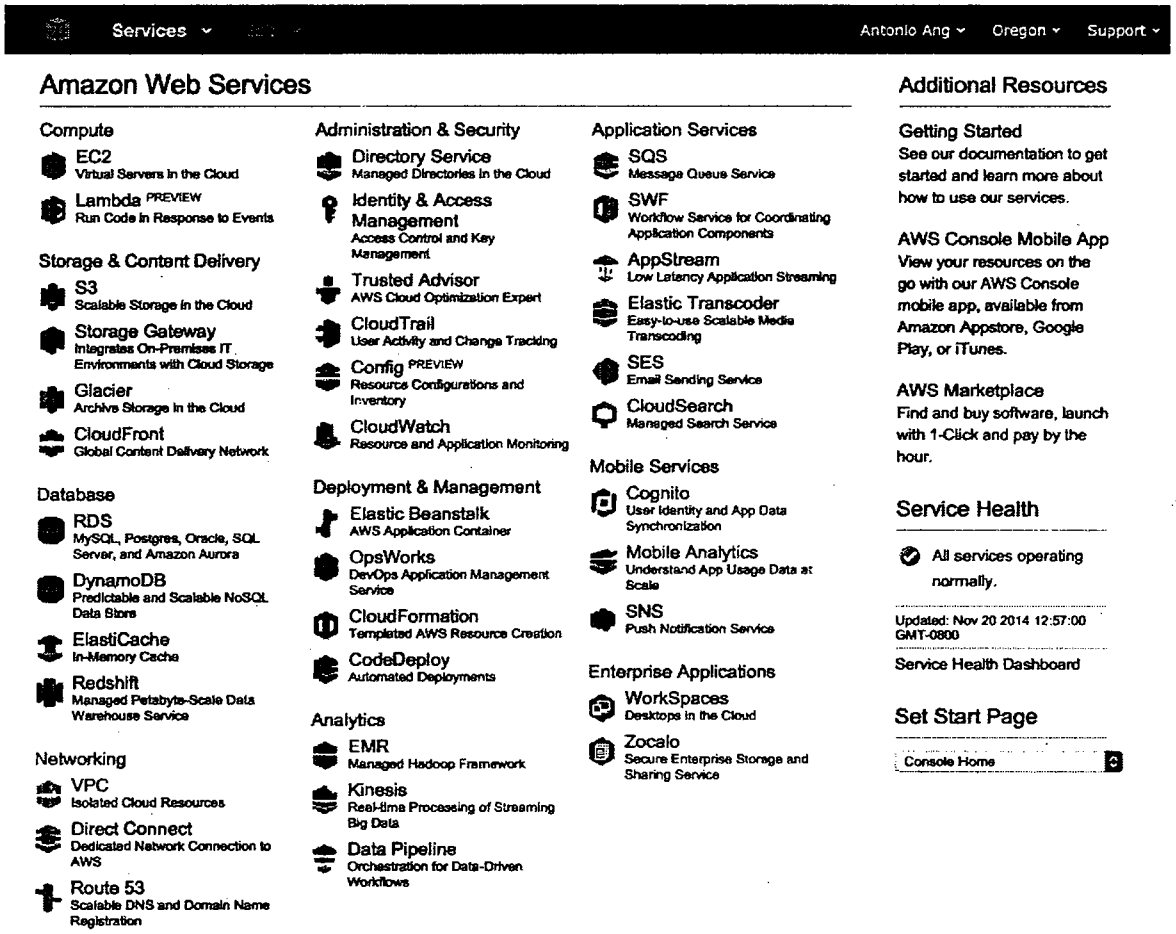
Lab 16

INTRODUCTION TO CLOUDWATCH

STEP 1: Log In to the Amazon Web Service Console

This laboratory experience is about Amazon Web Services and you will use the AWS Management Console in order to complete all the lab steps.



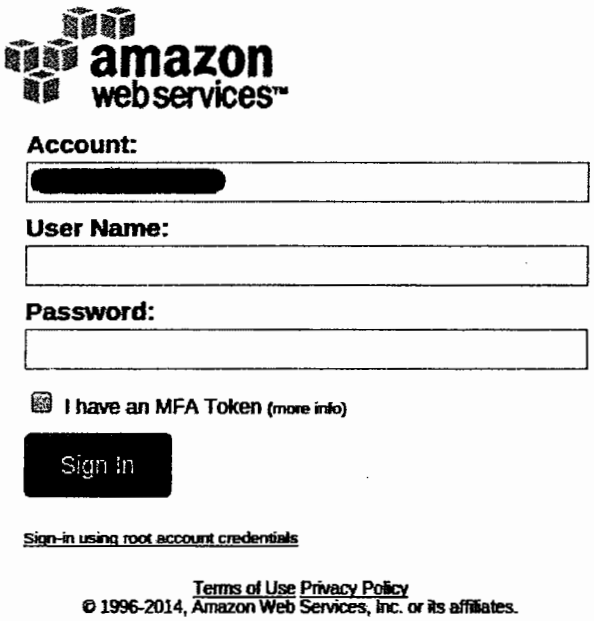
The AWS Management Console is a web control panel for managing all your AWS resources, from EC2 instances to SNS topics. The console enables cloud management for all aspects of the AWS account, including managing security credentials, or even setting up new IAM Users.

Log in to the AWS Management Console

In order to start the laboratory experience, open the Amazon Console by clicking this button:

Open AWS Console

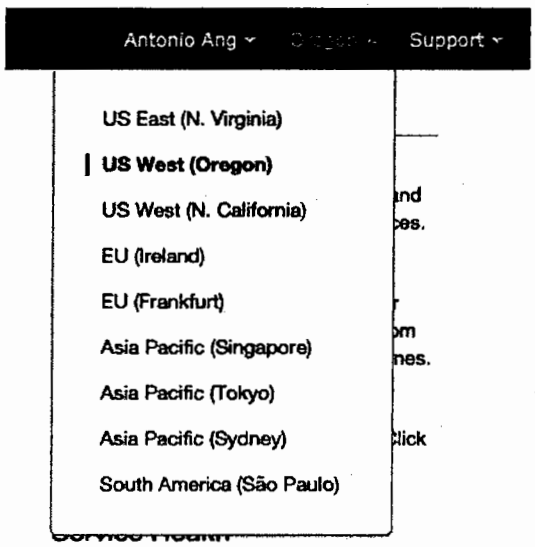
Log in with the username **xxxx** and the password **xxxx**



Select the right AWS Region

Amazon Web Services is available in different regions all over the world, and the console lets you provision resources across multiple regions. You usually choose a region that best suits your business needs to optimize your customer's experience, but you must use the region **US West (Oregon)** for this laboratory.

You can select the **US West (Oregon)** region using the upper right dropdown menu on the AWS Console page.



STEP 2: Explore CloudWatch

Key Concepts
There are a few concepts to define before working with CloudWatch. AWS has done a great job defining these concepts in the documentation, so let's stick with the official version:

Metrics
A metric is the fundamental concept in CloudWatch and represents a time-ordered set of data points. These data points can be either your custom metrics or metrics from other services in AWS. You or AWS products publish metric data points into CloudWatch and you retrieve statistics about those data points as an ordered set of time-series data. Metrics exist only in the region in which they are created.
Think of a metric as a variable to monitor, and the data points represent the values of that variable over time. For example, the CPU usage of a particular Amazon EC2 instance is one metric, and the latency of an Elastic Load Balancing load balancer is another.
CloudWatch stores your metric data for two weeks. You can publish metric data from multiple sources, such as incoming network traffic from dozens of different Amazon EC2 instances, or requested page views from several different web applications. You can request statistics on metric data points that occur within a specified time window.

Namespaces
CloudWatch namespaces are containers for metrics. Metrics in different namespaces are isolated from each other, so that metrics from different applications are not mistakenly aggregated into the same statistics.
Namespace names are strings you define when you create a metric. The names must be valid XML characters, typically containing the alphanumeric characters "0-9A-Za-z" plus "."(period), "-" (hyphen), "_" (underscore), "/" (slash), "#" (hash), and ":" (colon). AWS namespaces all follow the convention AWS/<service>, such as AWS/EC2 and AWS/ELB.
There is no default namespace. You must specify a namespace for each data element you put into CloudWatch.

Dimensions
A dimension is a name/value pair that helps you to uniquely identify a metric. Every metric has specific characteristics that describe it, and you can think of dimensions as categories for those characteristics. Dimensions help you design a structure for your statistics plan. Because dimensions are part of the unique identifier for a metric, whenever you add a unique name/value pair to one of your metrics, you are creating a new metric.

You can take a look at the full version of the documentation in here.

Exploring CloudWatch
Go on the CloudWatch console and make sure you are in the **US West (Oregon)** region.

Management Tools

CloudWatch
Monitor Resources and Applications

CloudFormation
Create and Manage Resources with Templates

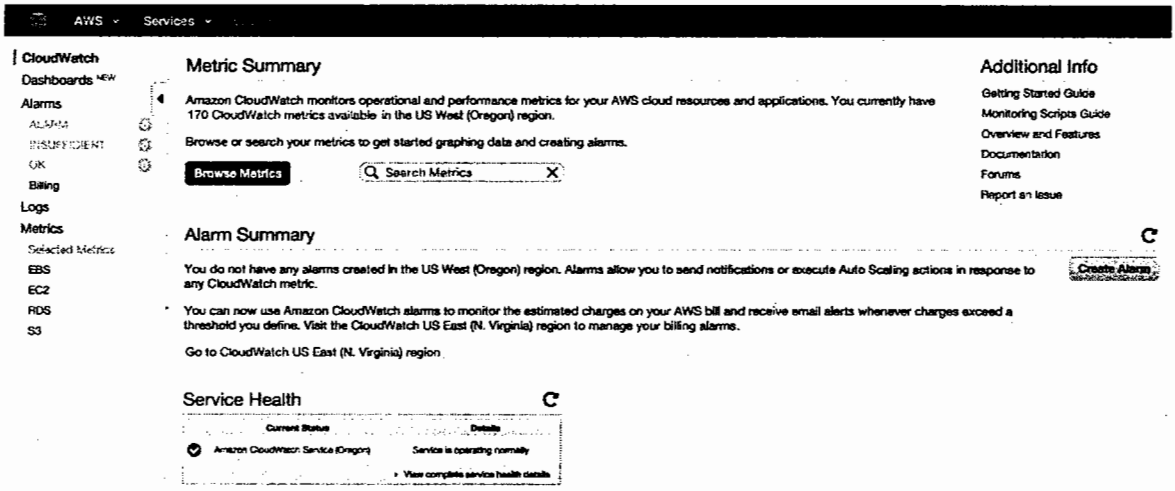
CloudTrail
Track User Activity and API Usage

Config
Track Resource Inventory and Changes

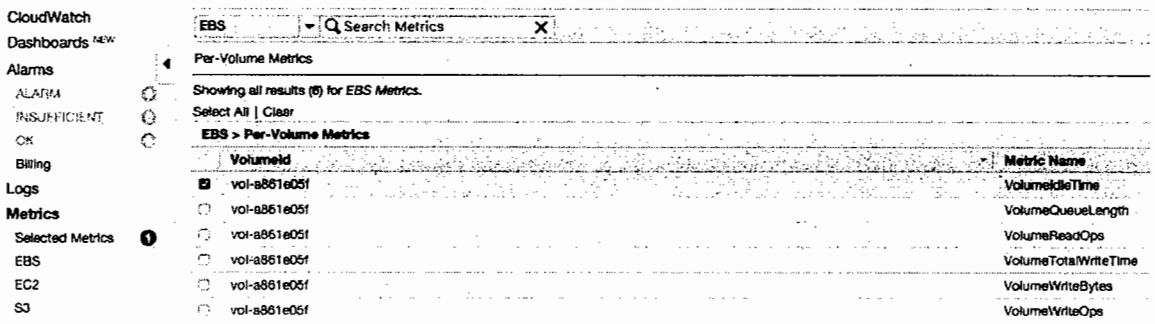
OpsWorks
Automate Operations with Chef

Service Catalog
Create and Use Standardized Products

Trusted Advisor
Optimize Performance and Security



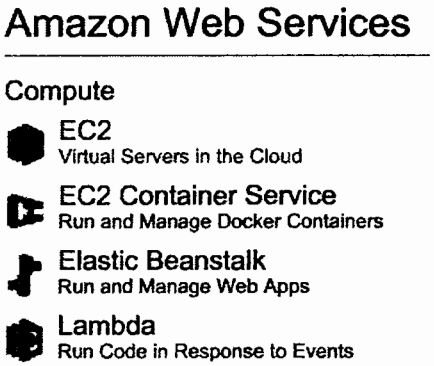
Take time to see what Metrics and Namespaces look like in the CloudWatch console. Simply select a namespace and then a particular metric:

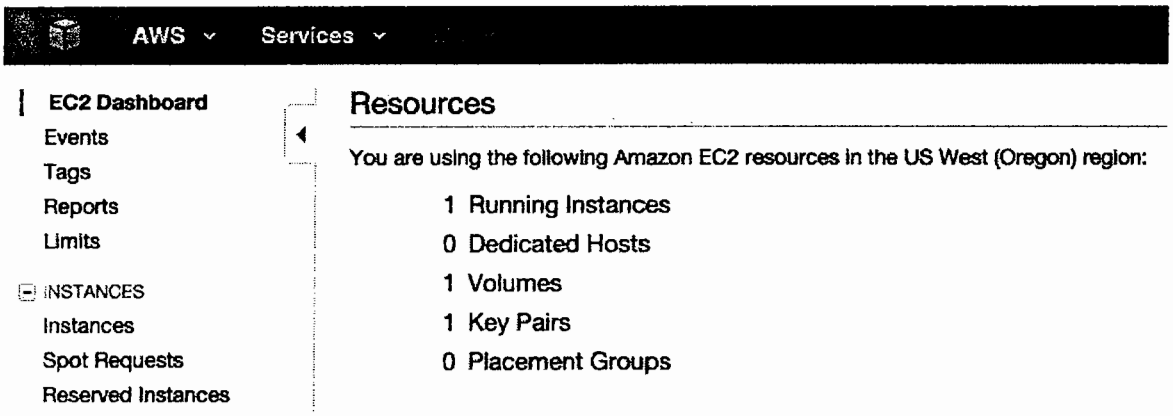


STEP 3: Monitoring EC2 Instances

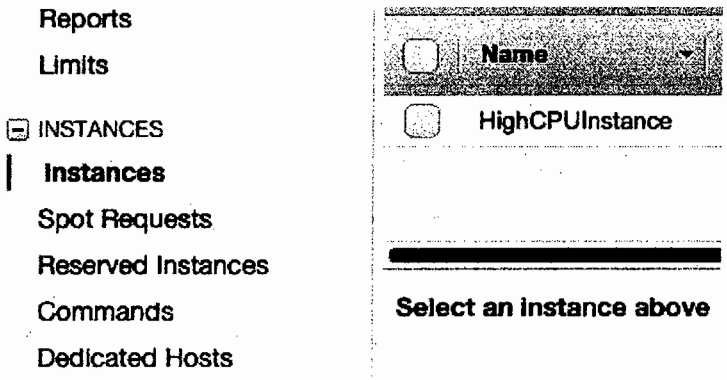
Standard Metrics

CloudWatch doesn't monitor everything out of the box, for EC2 instances, for example, it doesn't monitor things like disk space or memory usage, let's take a look at the metrics that CloudWatch monitors by default for EC2 instances, but this time in the EC2 console:

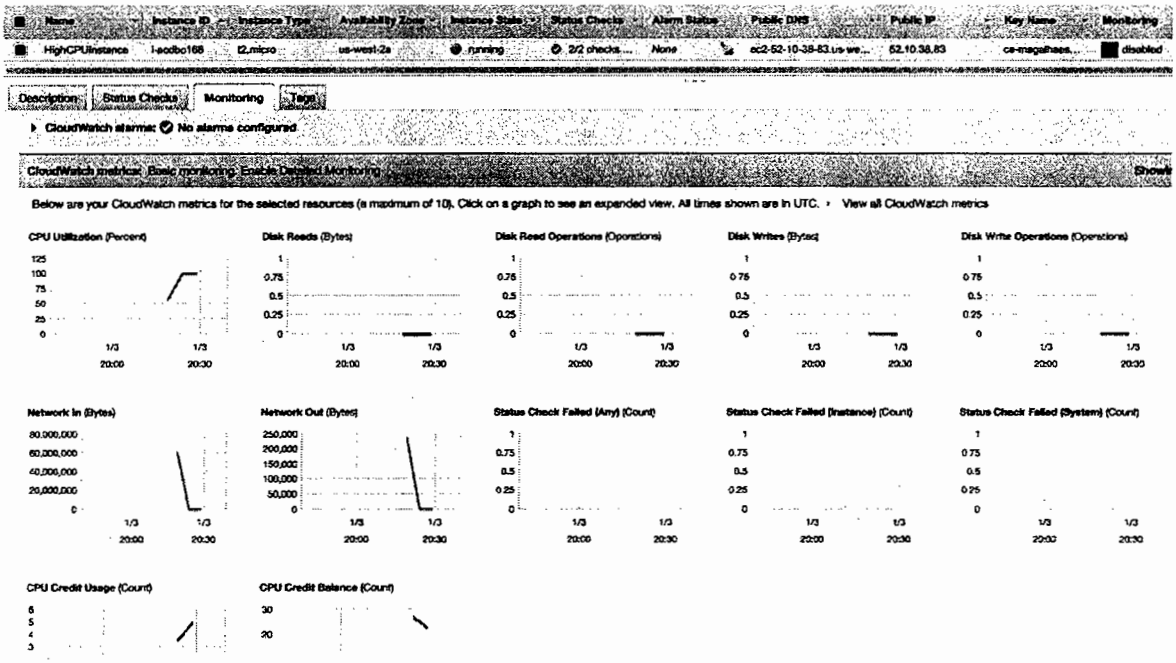




Click on instances to see more details:



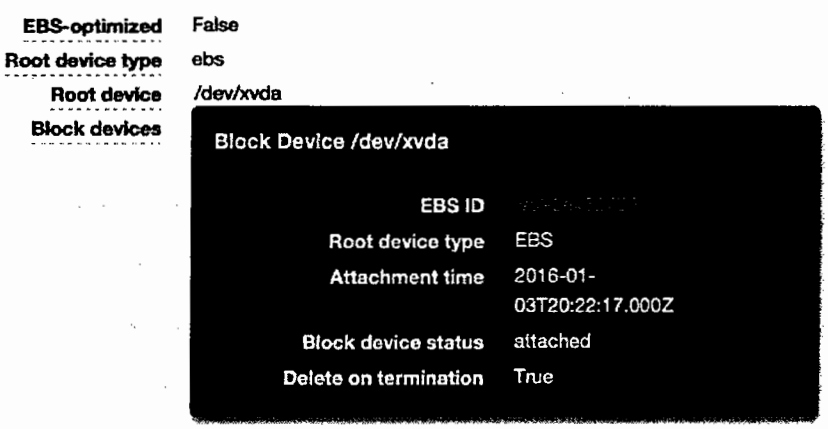
There is already an instance called **HighCPUInstance** running, select this instance and the click in the **Monitoring** tab and take a look at the standard metrics:



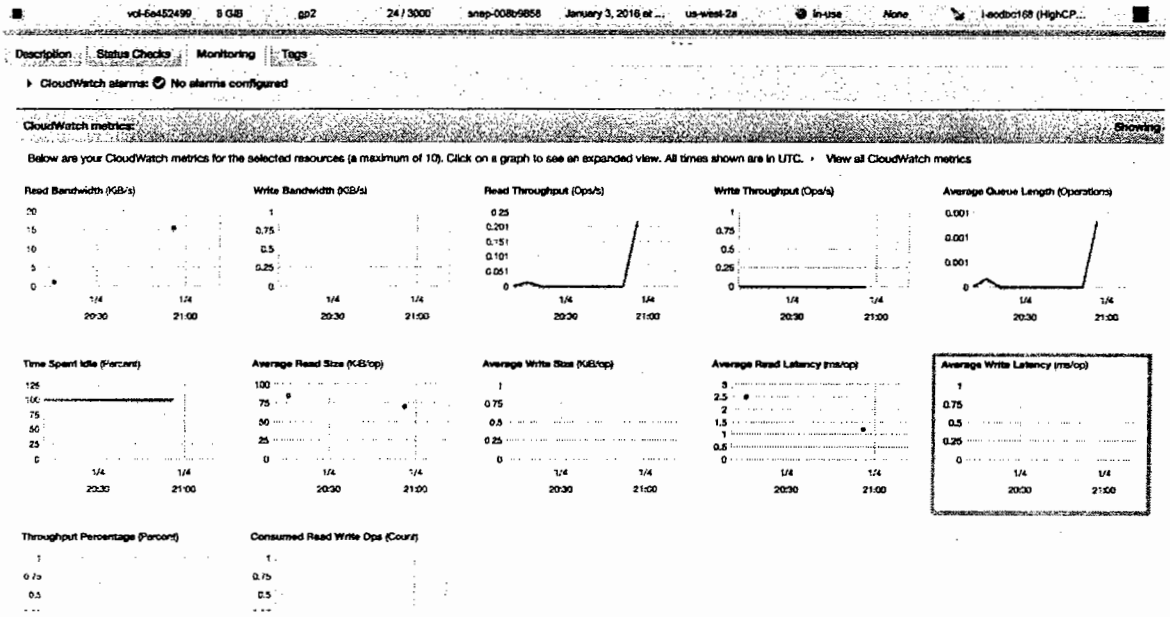
These are the standard metrics that CloudWatch monitors for all your EC2 instances, please refer to the documentation for more details.

You should be aware that all the metrics in this tab related to Disk (Disk Reads, Disk Read Operations, Disk Writes, Disk Write Operations) pertain to ephemeral storage disks. Those metrics will not represent anything if you have launched an EBS backed instance.

To see the metrics related to EBS volumes you need to go somewhere else. Let's take a look at the metrics of the EBS volume from this particular instance. You can do it by selecting the instance, clicking on the **Description** tab, click on the **Root** device then on the EBS ID:



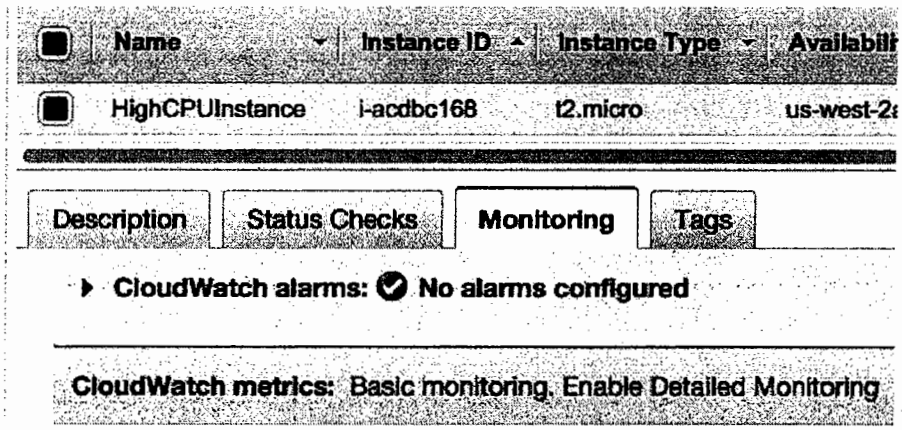
Now with the volume selected, click on the **Monitoring** tab to see the metrics for this EBS volume:



You can learn even more about these metrics by taking a look at the documentation.

Enabling / Disabling Detailed Monitoring

There is also another important thing to consider, CloudWatch typically monitors your EC2 instances every 5 minutes. If you need more frequent monitoring, you can configure CloudWatch to monitor your instances every minute. To make this happen, you need to enable something called **Detailed Monitoring.** You can do this during the instance launch or change it anytime. To enable or disable detailed monitoring after an instance is launched select the instance, click on the **Monitoring** tab, and then on **Enable** (or **Disable**) **Detailed Monitoring**:



For better experience during this lab you should **Enable Detailed Monitoring** for the**HighCPUInstance**.

If you are done exploring and understand what the standard metrics mean, then you should proceed to the next step.

STEP 4: Install the EC2 Monitoring Scripts

In the last step we learned that CloudWatch only monitors a few EC2 metrics per default. That may be enough for many cases, but sometimes you will want more complete monitoring. In this step let's enhance the power of CloudWatch by installing some monitoring scripts in an EC2 instance. These scripts will run every 5 minutes and send custom metrics to CloudWatch.

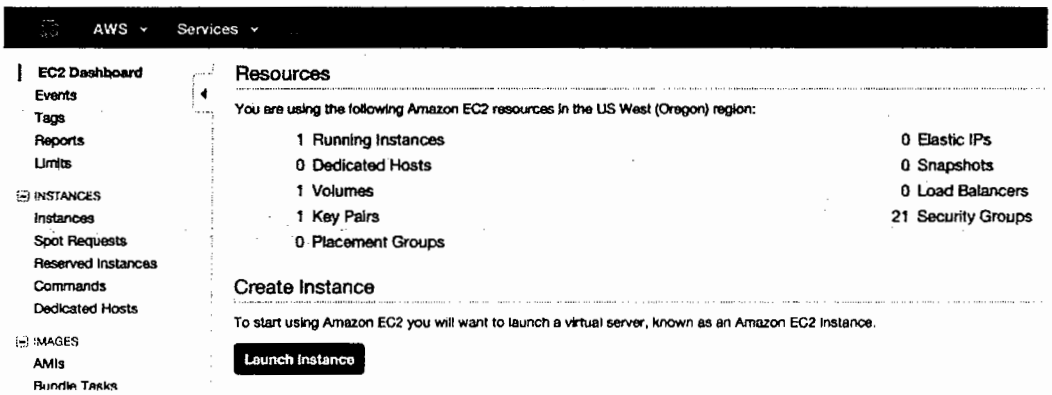
Install the EC2 Monitoring Scripts

Let's launch a new instance and configure the monitoring scripts during the instance's launch by employing User Data.

User Data is a script that you can configure to run once, during the instance launch. This is very useful for bootstraping and with that we can customize our instances without accessing it via SSH or RDP for Windows instances.

Lets launch our instance and see how it works. Go to the EC2 console:

On the Dashboard click on **Launch instance**:



On step 1, select Amazon linux as our AMI:

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag Instance | 6. Configure Security Group | 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

1 to 22 of 22 AMIs

My AMIs

AWS Marketplace

Community AMIs

Free tier only

Amazon Linux

Free tier eligible

Amazon Linux AMI 2015.09.1 (HVM), SSD Volume Type - ami-f0091d91

Select

64-bit

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root device type: ebs Virtualization type: hvm

On step 2, select t2.micro as our instance size:

1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag I

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. I
storage, and networking capacity, and give you the flexibility to choose the appropriate mix of
needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB me

	Family	Type	vCPUs	Mem
☐	General purpose	t2.nano	1	
■	General purpose	t2.micro Free tier eligible	1	

Now, on step 3, is where the magic happens. First of all, you need to expand the advanced area of this step by clicking on the little arrow, just to the left of where you see **Advanced Details**:

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of t
to the instance, and more.

Number of instances	1 — Launch into Auto Scaling Group
Purchasing option	☐ Request Spot instances
Network	vpc-cf7555aa (172.31.0.0/16) (default) — Create new VPC
Subnet	No preference (default subnet in any Availability Zone) — Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
IAM role	None — Create new IAM role
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.

▾ Advanced Details

User data	● As text ○ As file ☐ Input is already base64 encoded (Optional)

Now we will insert the code that we are going to execute during the instance launch, however in order to send metrics to CloudWatch, we need to configure some credentials first, you can use either Access Keys or IAM roles for this task, in this lab we will follow the best practices and use IAM roles. There is an instance role already created in you account, so click on IAM role and select it. It will have a name that looks like **cwinitconf-EC2MonitoringRole-XXXXXXXXXX** :

Auto-assign Public IP	Use subnet setting (Enable)
IAM role	None aws-opsworks-ec2-role ✓ CF-EC2MonitoringRole-1KVQ5IWVBQCSW EMR_EC2_DefaultRole — Create new IAM role

Then copy and paste this code in the User Data text field:

```
#!/bin/bash
yum install -y perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https
wget http://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.1.zip
unzip CloudWatchMonitoringScripts-1.2.1.zip
```

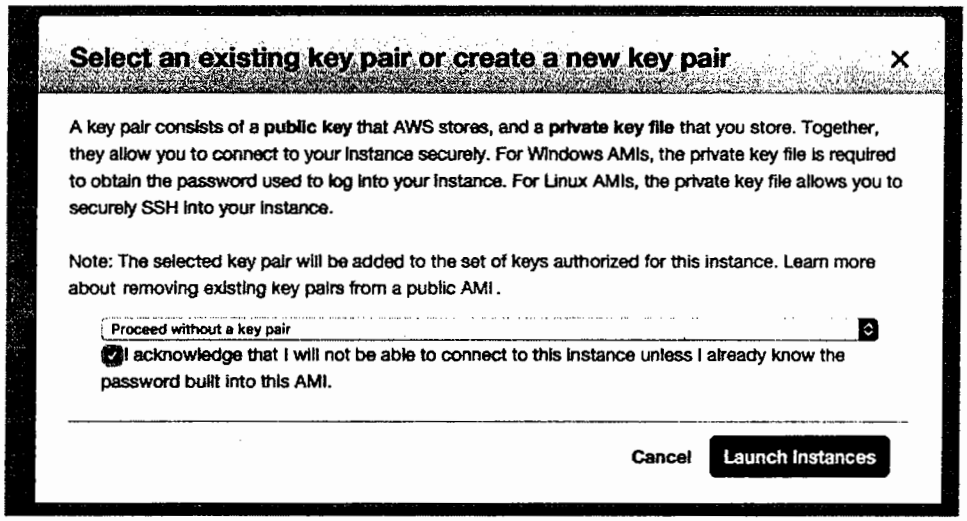
```
rm CloudWatchMonitoringScripts-1.2.1.zip
echo "*/5 * * * * /aws-scripts-mon/mon-put-instance-data.pl --mem-util --disk-space-util --disk-path=/ --from-cron" > monitoring.txt
crontab monitoring.txt
rm monitoring.txt
```

▾ Advanced Details

User data ⓘ ◉ As text ○ As file ☐ Input is already base64 encoded

```
#!/bin/bash
yum install -y perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https
wget http://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.1.zip
```

Now we are ready to proceed. Select **Review and Launch**, then press **Launch**. A dialog box will appear asking for a Key Pair. Select **Proceed without a key pair**, mark the "I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI." checkbox and finally click on **Launch Instances**.



Now we can give a name to our instance, it's not mandatory, but it will help us in the next step:

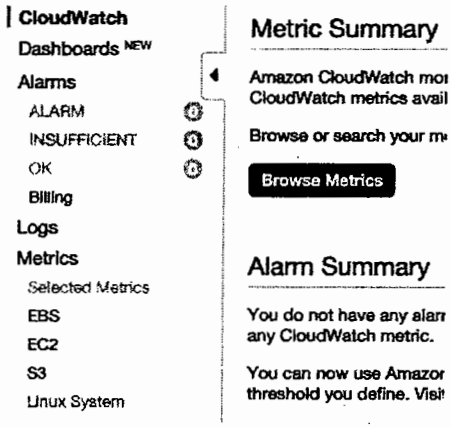
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
HighCPUInstance	i-acdbc168	t2.micro	us-west-2a	running	2/2 checks ...
Monitoring Scripts	i-fa52b43d	t2.micro	us-west-2a	running	2/2 checks ...

Wait untill the Instance State change to "running", then select the instance and go to the monitoring tab. Notice that nothing is different in there. This happens because we are not adding anything special to CloudWatch. We are just configuring our instance to send custom

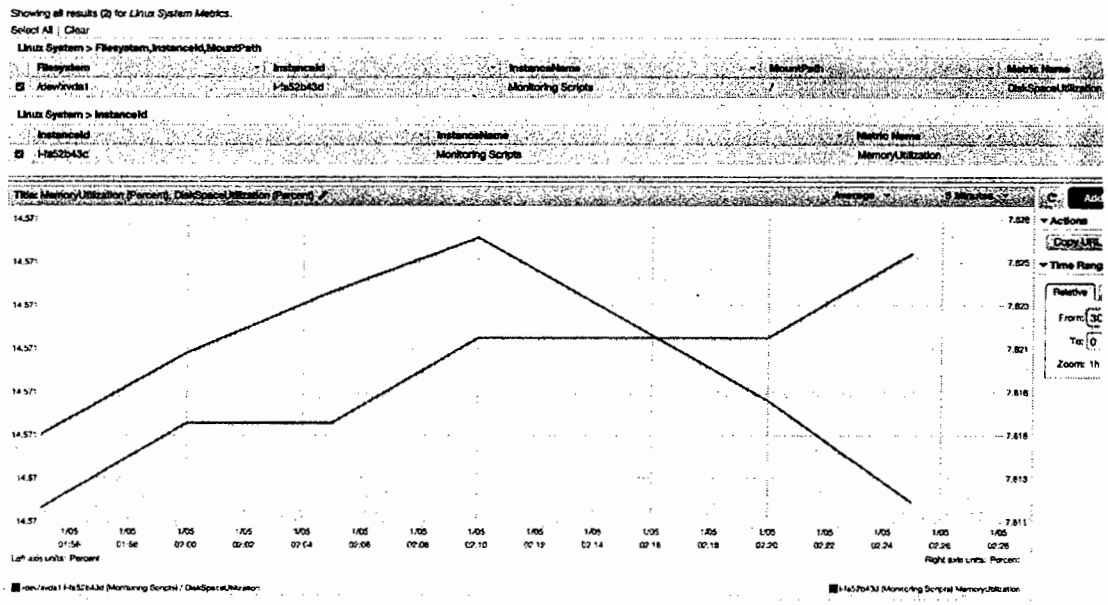
metrics each five minutes to the CloudWatch service, using the credentials associated with the IAM role in the EC2 instance.

To check the results, go to the CloudWatch console. Notice that now we have a new Namespace in here, it is called **Linux System.** This name is configured when you send the custom metrics and this particular name is the standard for these monitoring scripts. If you don't see the new Namespace wait a few minutes and refresh the page because CloudWatch takes some time to display the information in the dashboard.

Once the new Namespace appears you should click on it.



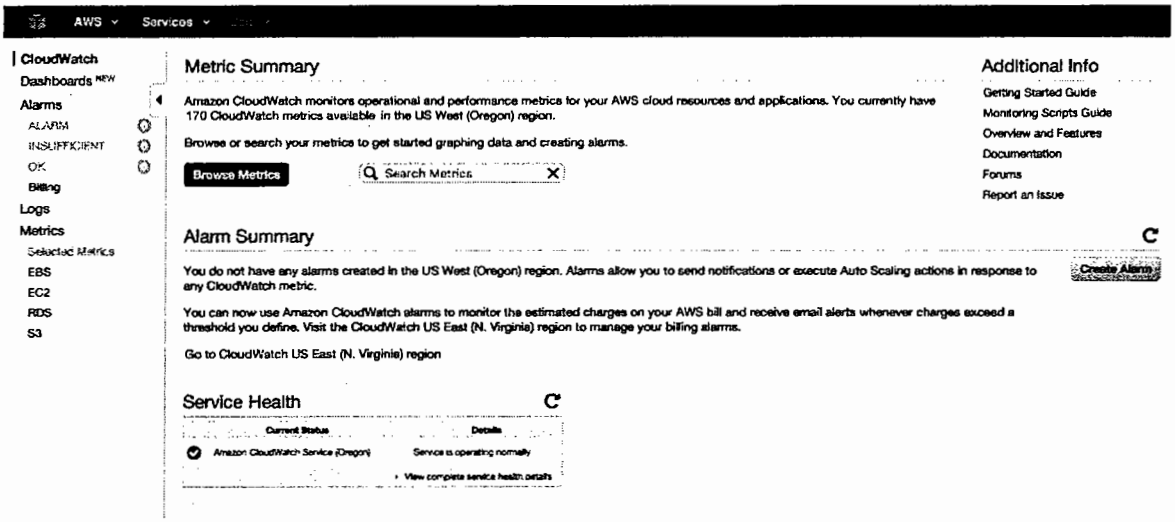
Now we have two new metrics being monitored by CloudWatch, **MemoryUtilization** and**DiskSpaceUtilization**. Select one of them, or both, and see what appears in the graph.



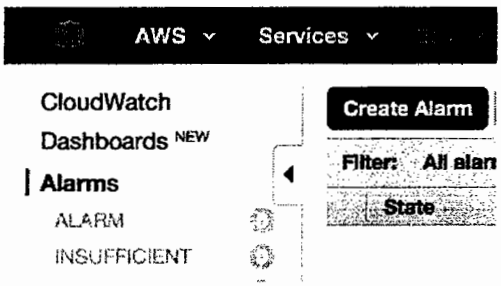
We will continue exploring the Graph tool in a further step.

STEP 5: Create your first CloudWatch alarm

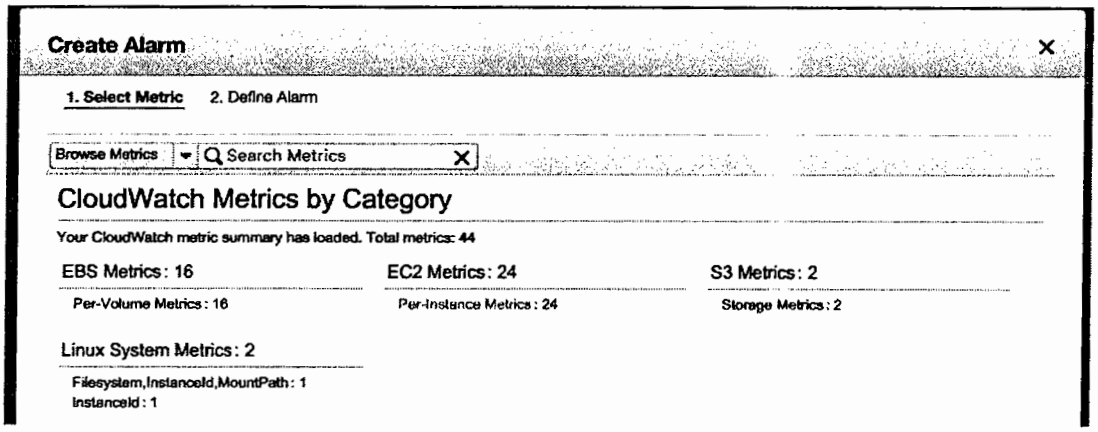
Go to the CloudWatch console and click on **Alarms**:



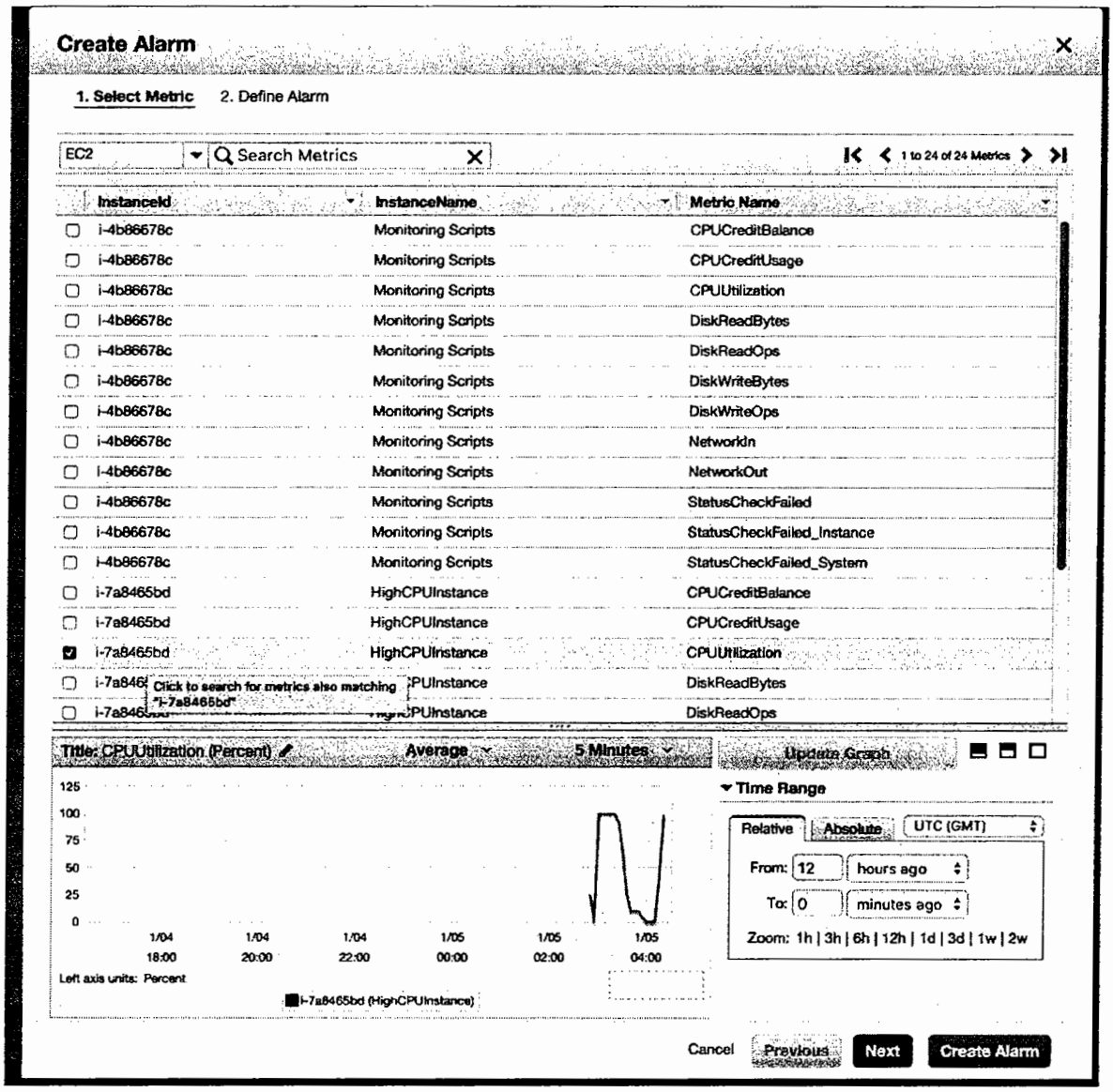
In the Alarms page click **Create Alarm**:



A pop-up will appear on the screen. Now you need to select a particular metric to create the alarm. We will create an alarm for the CPUUtilization metric of the HighCPUInstance, so we'll need to select this particular metric first. To do this, click on **Per-Instance Metrics** right below **EC2 Metrics**:



Then look for the **CPUUtilization** metric related to the **HighCPUInstance:**



And now we are ready to click **Next**.

An Alarm is basically a metric that is beeing monitored over time. Once the metrics meet a customizable rule it will change the alarm's state. There are 3 possibles states for an alarm:

OK—The metric is within the defined threshold
ALARM—The metric is outside of the defined threshold
INSUFFICIENT_DATA—The alarm has just started, the metric is not available, or not enough data is available for the metric to determine the alarm state
We should also set a period of time when the metric will be evaluated. And we can set one or multiple Actions for an alarm. However, it is not mandatory. After filling in all the information needed, we should have something like this:

Create Alarm

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name: High CPU Alarm

Description: When CPU Utilization > 50%

Whenever: CPUUtilization

is: >= 50

for: 1 consecutive period(s)

Actions

Define what actions are taken when your alarm changes state.

Notification Delete

Whenever this alarm: State is ALARM

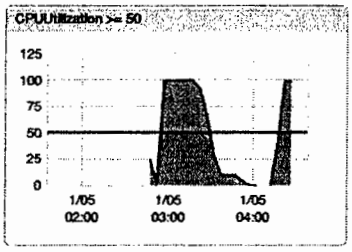
Send notification to: NewTopic Select list

Email list: myPersonalEmail@email.com

+ Notification + AutoScaling Action + EC2 Action

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line for a duration of 5 minutes



Namespace: AWS/EC2

InstanceId: I-7a8465bd

InstanceName: HighCPUInstance

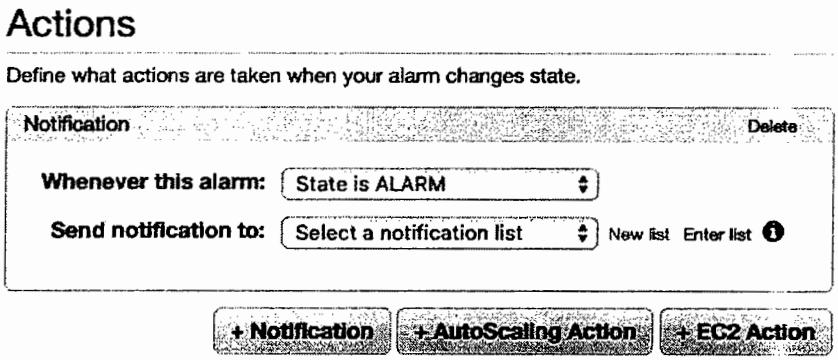
Metric Name: CPUUtilization

Period: 5 Minutes

Statistic: Average

Cancel Previous Next Create Alarm

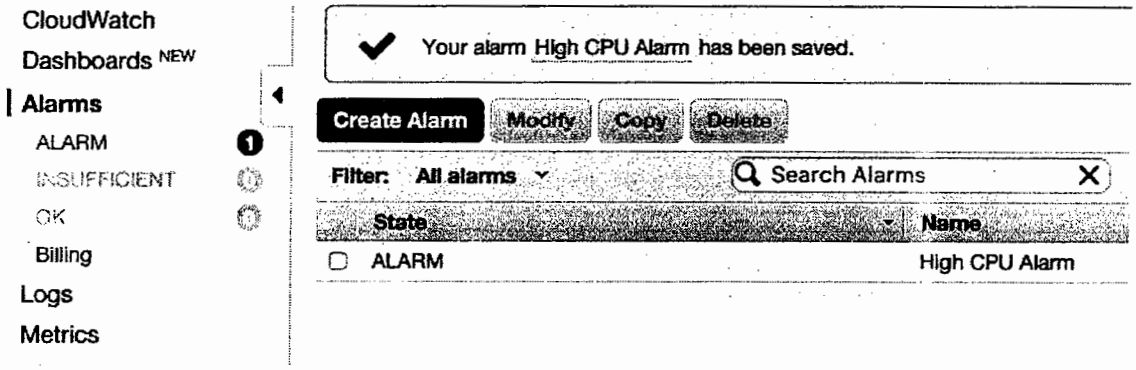
Notice that in the **Actions** area, you must click **New list** in order to change the form. After you must provide a new SNS topic name and also your personal email. The below image is how the screen should appear prior to clicking on **New list**:



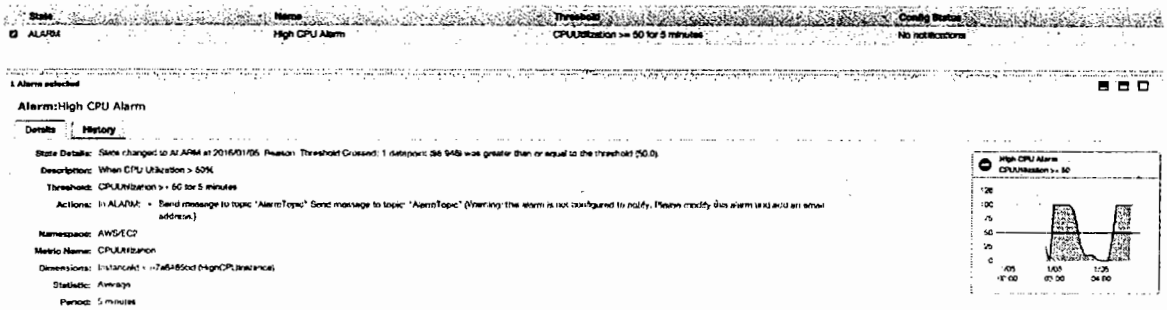
Now we're ready to create the Alarm. Click **Create Alarm**.

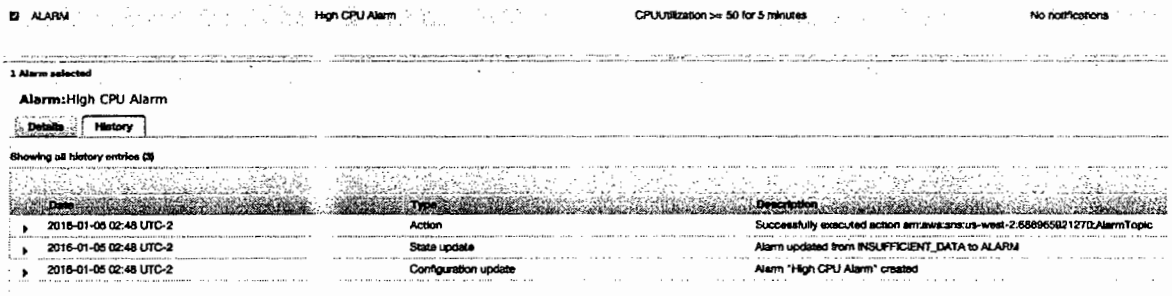
You have just created a new alarm, along with a new SNS topic -- if you have also inserted your email into the Email list. Go to your email account because you should have received an email from AWS. This email is a subscription confirmation. Just click the provided link and you're ready to receive emails from this topic. Every time the state of the alarm switches to**ALARM,** you'll receive a new email.

Now that the alarm is created, you can check it out, don't worry about the state of the alarm right now, we will come back to it soon.



By selecting the alarm you can see some useful information about the alarm itself. In the **Details**tab you have a general overview about the alarm, and in the **History** tab you can see the last 50 states of the alarm:

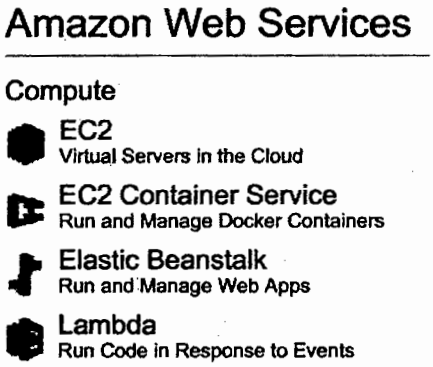




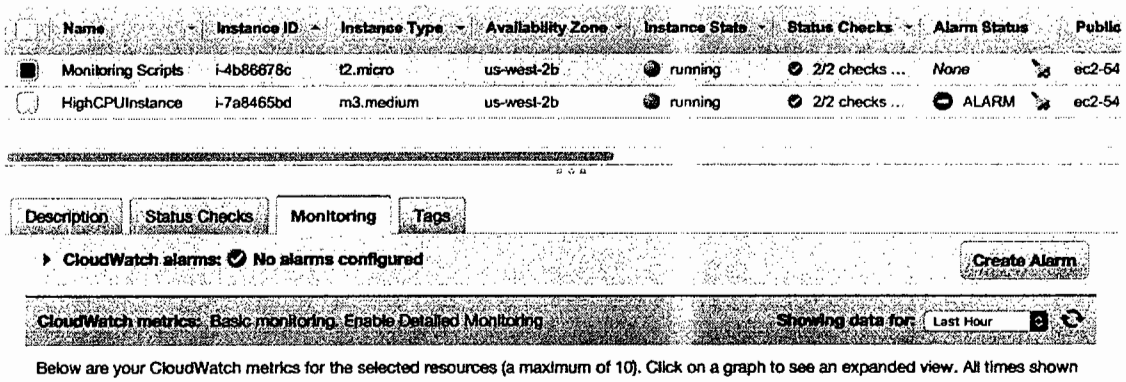
Take some time exploring the latest alarm changes and try to understand what is going on in each change. You can see more details of the changes by clicking the little arrow next to the date/time of the changes.

STEP 6: Create an Alarm using the EC2 consle

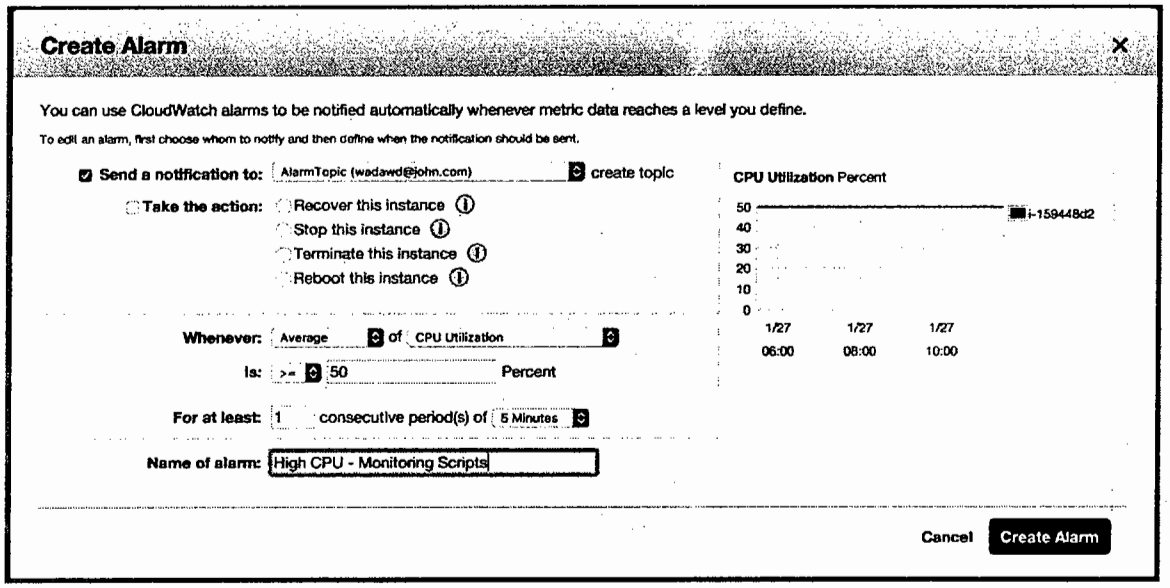
Go to the EC2 console and click on **Instances**:



Select the **Monitoring Scripts** instance. Then go to the **Monitoring** tab and then click **Create Alarm**:



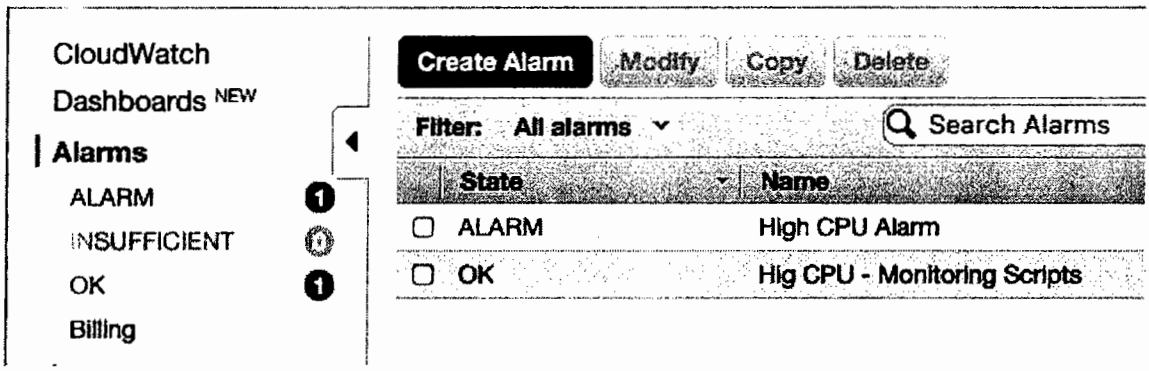
A screen will appear to configure the details of the alarm. It is similar to the screen we saw in the last step and you should feel free to configure the details for yourself. In the end, you should have something similar to it:



Click on **Create Alarm** and a new screen will appear. A message will appear saying that the alarm was created. You can click on **Close**.

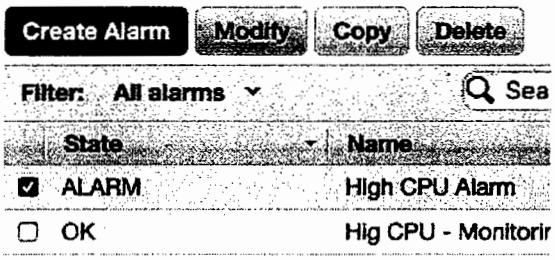
Setting EC2 Actions to the Alarms

Now that we have explored two ways of creating an alarm, let's make sure that they are helping us somehow. Let's go back to the **Alarms** page in the **CloudWatch console** and we can see that the first alarm we created is stuck in the ALARM state:



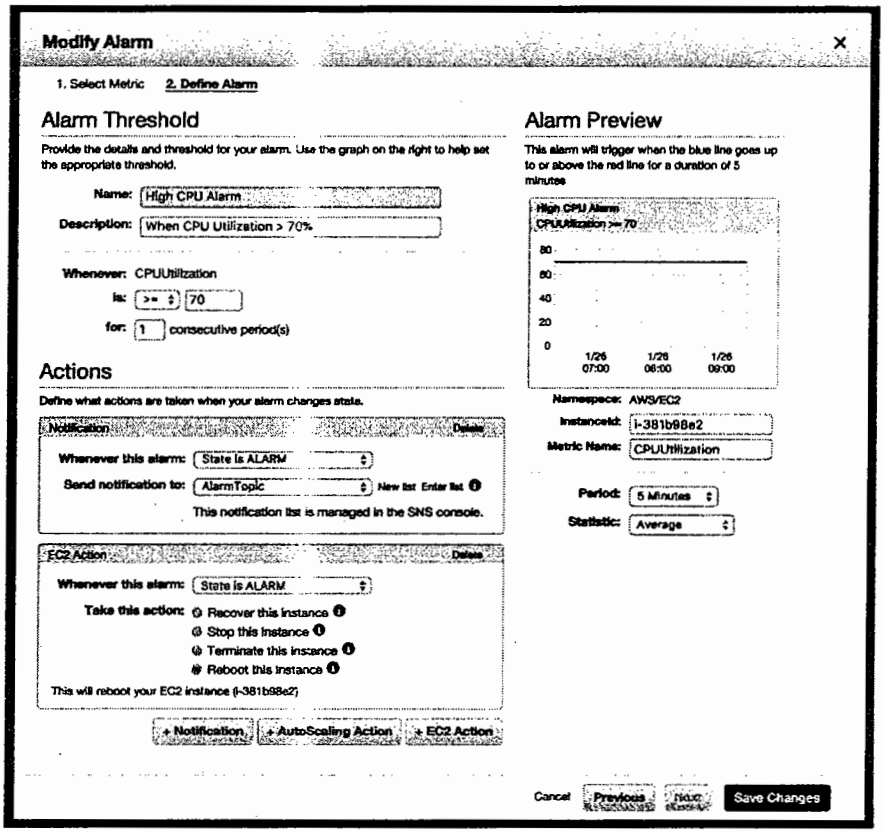
This happens because this instance is running an application that consumes 100% of the CPU utilization. You probably won't have anything like this in a production environment. But, let's imagine that we are managing a production environment and we have an instance that is becoming unavailable intermittantly because of the CPU utilization. We sure would like to receive a notification every time the CPU utilization is high, but this can happen anytime, in the middle of the night, or during a weekend, so it would be nice to have a pre-defined action in these cases -- at least until we find a definitive solution for the problem.

To help us with that, we can set EC2 actions to our alarms. Now, let's modify the **High CPU Alarm** that we created for the **HighCPUInstance.** To do this, just select the alarm that you want to edit, and click on **Modify**:



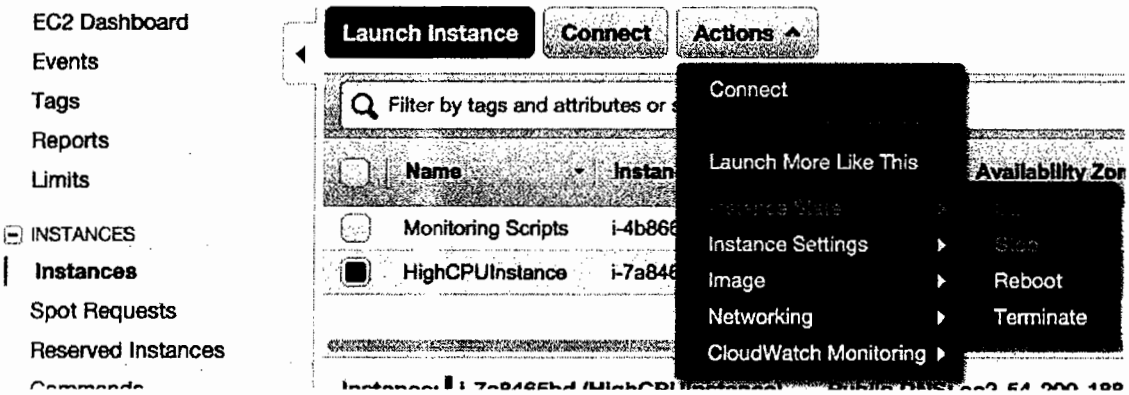
A screen will appear for us. In the **Actions** area, we need to click on **+ EC2 Action** to configure our action.

To make our alarm more suitable to our needs, let's change the threshold from 50% of CPU Utilization to 70% and set a new **EC2 Action** to **Reboot this instance** whenever the state of this alarm is ALARM. Once you fill in all the necessary info, you should have something like this:

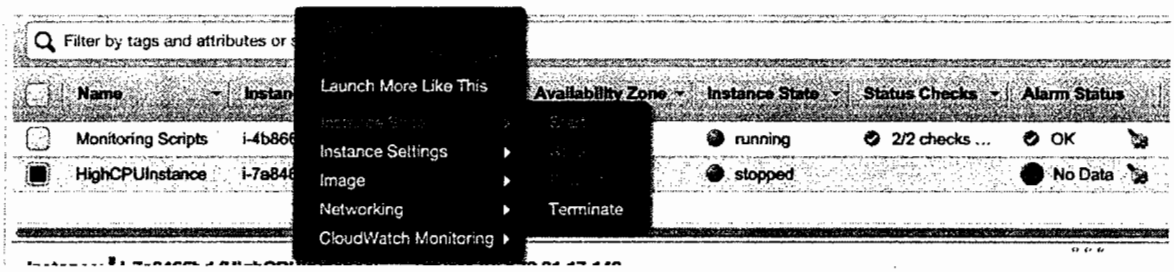


Click **Save Changes**.

Now we need to change the state of the alarm. This is needed because the current state of the alarm is ALARM, so CloudWatch will not take any actiions until the state changes to something else. Let's stop our instance to change the state of the alarm to INSUFICIENT DATA. To do that, go to the EC2 console, and in the Instances page, select the HighCPUInstance and stop it for a few minutes:



Once the **Alarm Status** of the instance changes to **No Data** you can start the instance again:

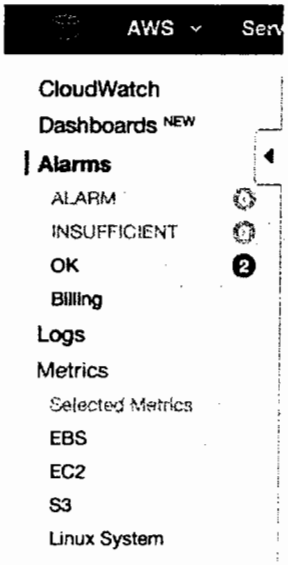


If you feel that it is taking too much time you can delete the current alarm and create a new one from scratch with an email notification and an EC2 Action. That will do the job as well.

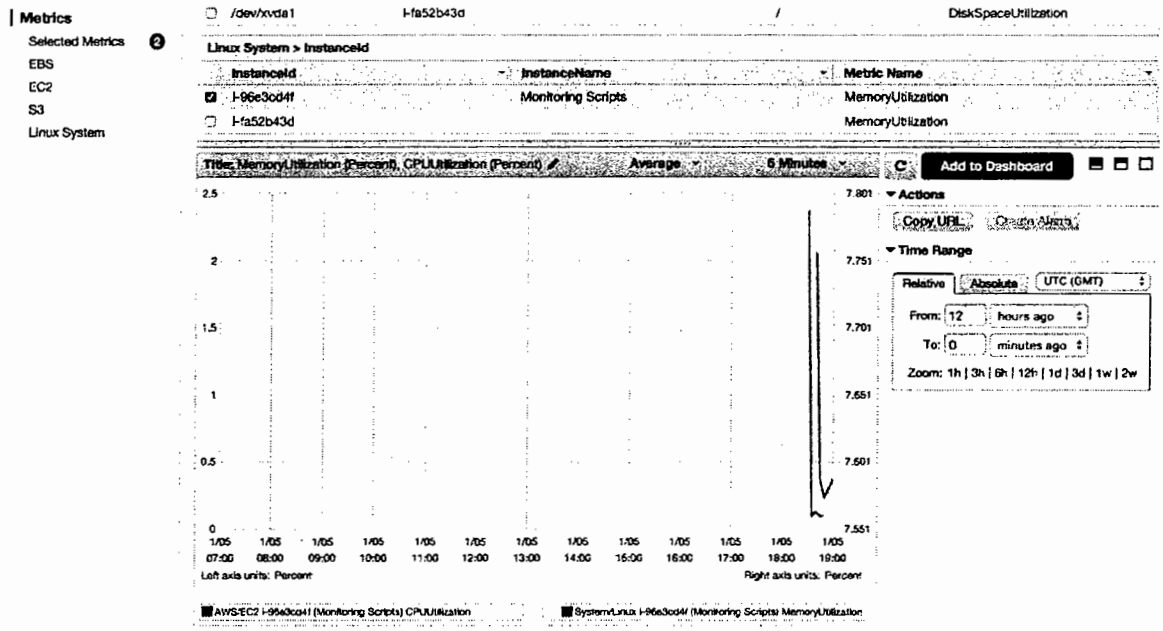
Now you can remain in the **Instances** page in the EC2 console and watch CloudWatch reboot the instance every time the **Alarm Status** changes to ALARM. Later on, you can go to the **Alarms** in the CloudWatch console and see the History of the alarm. Take some time here to watch things and adjust the settings in the alarm and change its behavior. I encourage you to not to limit yourself to these alarms only, create more alarms and test them by yourself.

STEP 7: Analyzing CloudWatch metrics over time

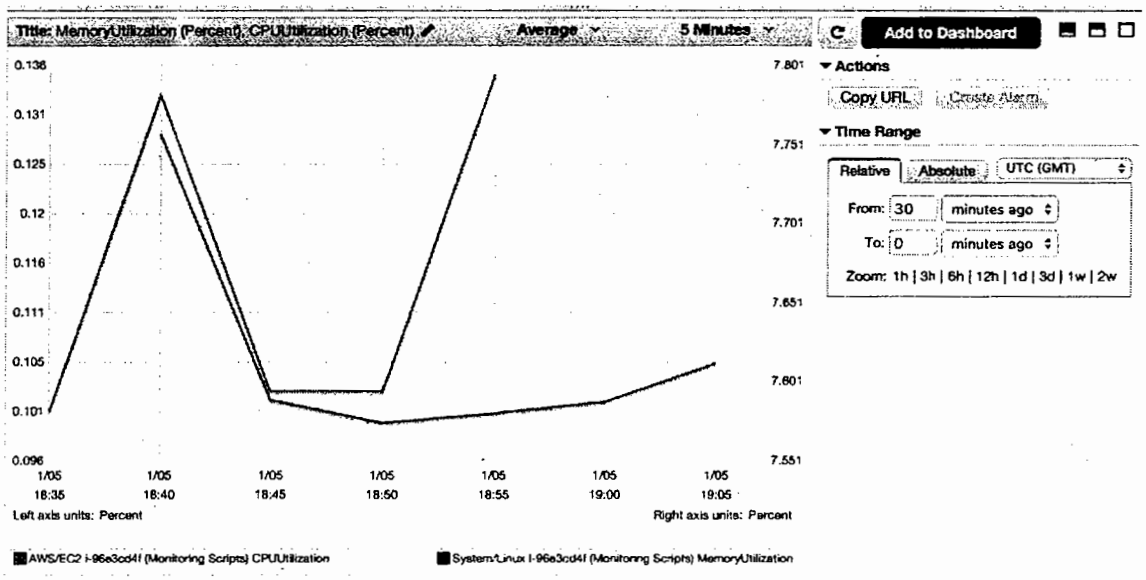
Go to the CloudWatch console and click on Metrics:



Now Select two metrics. In this case I will select the **CPUUtilization** and **MemoryUtilization** of the **Monitoring Scripts** instance. In the end you should have something like this:

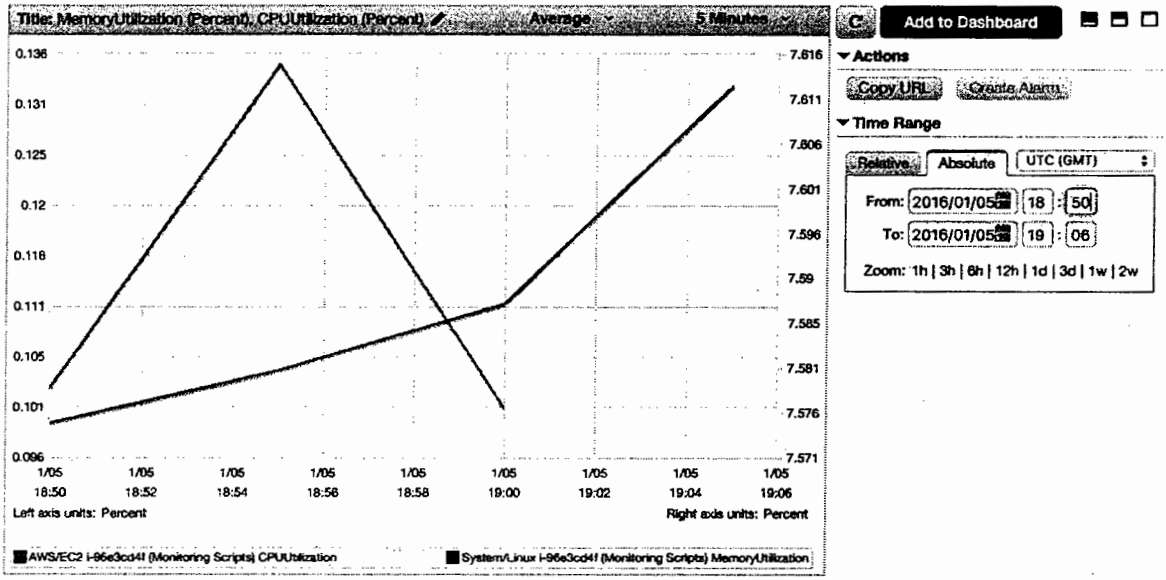


Now let's play with this graph. We will change the Relative time to show what happened in just the last 30 minutes. To do this, simply change the values and hit enter on the number field once you're done:



Now we can see, in greater detail, what happened in the last 30 minutes, and analyze what happens with the memory utilization and the CPU utilization goes up and down.

Another cool feature is the ability to set an Absolute time to check the metrics. This is useful in case we want to troubleshoot what happened during an outage, like in the last step's example, we could use it to examine what happened with a particular instance when the High CPU Alarms changed its state to ALARM. Do this by clicking on the **Absolute** tab and selecting the exact time that you want to review:



As soon as you find something interesting, you can share it with others. To do that click on Copy URL, and notice that you need to be logged in the AWS console in order to see the link. That means you can only share links with people who have access to your AWS account.

Take your time exploring how to use the Graph tool and how to see metrics over time. An interesting thing to watch might be what happens with the EBS volume associated with the HighCPUInstance when the CPUUtilization goes up.