AWS Basics
Auto Scaling (for Linux)

Table of Contents

Introduction

Objectives

This lab introduces the basics of Auto Scaling in Amazon Web Services highlighting multiple Auto Scaling use cases and the command-line tools used for Auto Scaling configuration.

You will learn about:

- Configuring Auto Scaling to automatically launch web server instances
- Building an elastic web farm by integrating Auto Scaling with an Elastic Load Balancer
- Setting CloudWatch alarms to automatically adjust the size of the web farm based on CPU utilization
- Utilizing Auto Scaling to ensure the availability of steady state resources

After completing this lab you will have configured and tested an elastic web farm which automatically scales capacity to accommodate load. In addition you will have explored a steady state use case in which Auto Scaling is used to maintain high availability of critical resources.

Prerequisites

This lab assumes basic knowledge of Amazon EC2 and Linux command-line tools. If you are not comfortable with these prerequisites, please consider preparing for this lab by working through the first lab in this series, *AWS Basics: Creating EC2 Instances (with Linux)*.

Background

What is Auto Scaling?

The Amazon Web Services (AWS) Auto Scaling service automatically adds or removes compute resources allocated for your cloud application, in response to changes in demand. For applications configured to run on a cloud infrastructure, scaling is an important part of cost control and resource management. Scaling is the ability to increase or decrease the compute capacity of your application either by changing the number of servers (horizontal scaling) or changing the size of the servers(vertical scaling).

When you scale using Auto Scaling you can increase the number of servers you're using automatically when the user demand goes up to ensure that performance is maintained, and you can decrease the number of servers when demand goes down to minimize costs. Auto Scaling helps you make efficient use of your compute resources by automatically doing the work of scaling for you. This automatic scaling is the core value of the Auto Scaling service.

Auto Scaling is well suited for applications that experience hourly, daily, or weekly variability in usage and need to automatically scale horizontally to keep up with usage variability. Auto Scaling frees you from having to accurately predict huge traffic spikes and plan for provisioning resources in advance of them. With Auto Scaling you can build a fully scalable and affordable infrastructure on the cloud.

Key Components of Auto Scaling

Auto Scaling Groups

An Auto Scaling group is a representation of multiple Amazon EC2 instances that share similar characteristics, and that are treated as a logical grouping for the purposes of instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase or decrease the number of instances in that group to improve the performance of the application. You can use the Auto Scaling group to automatically scale the number of instances or maintain a fixed number of instances. You create Auto Scaling groups by defining the minimum, maximum, and desired number of running EC2 instances the group must have at any given point of time.

An Auto Scaling group starts by launching the minimum number (or the desired number, if specified) of EC2 instances and then increases or decreases the number of running EC2 instances automatically according to the conditions that you define. Auto Scaling also maintains the current instance levels by conducting periodic health check on all the instances within the Auto Scaling group. If an EC2 instance within the Auto Scaling group becomes unhealthy, Auto Scaling terminates the unhealthy instance and launches a new one to replace the unhealthy instance. This automatic scaling and maintenance of the instance levels in an Auto Scaling group is the core value of the Auto Scaling service.

Launch Configurations

A launch configuration is a template that the Auto Scaling group uses to launch Amazon EC2 instances. You create the launch configuration by including information such as the Amazon machine image ID to use for launching the EC2 instance, the instance type, key pairs, security groups, and block device mappings, among other configuration settings. When you create your Auto Scaling group, you must associate it with a launch configuration. You can attach only one launch configuration to an Auto Scaling group at a time. Launch configurations cannot be modified. They are immutable. If you want to change the launch configuration of your Auto Scaling group, you have to first create a new launch configuration and then update your Auto Scaling group by attaching the new launch configuration. When you attach a new launch configuration to your Auto Scaling group, any new instances will be launched using the new configuration parameters. Existing instances are not affected.

Auto Scaling Policies
An Auto Scaling policy is a set of instructions for Auto Scaling that tells the service how to respond to Amazon CloudWatch alarm messages. The Auto Scaling policy can give instructions to scale in (terminate EC2 instances) or scale out (launch EC2 instances) the Auto Scaling group.

Elastic Load Balancing (ELB)

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance.

Auto Scaling Groups can optionally be configured to automatically register the instances they launch with an Elastic Load Balancer.

CloudWatch

Amazon CloudWatch provides monitoring for AWS cloud resources and the applications customers run on AWS. Developers and system administrators can use it to collect and track metrics, gain insight, and react immediately to keep their applications and businesses running smoothly. Amazon CloudWatch monitors AWS resources such as Amazon EC2 and Amazon RDS DB instances, and can also monitor custom metrics generated by a customer’s applications and services.

Later in this lab you will see how CloudWatch alarms can be configured to trigger execution of Auto Scaling Policies.

Simple Notification Service (SNS)

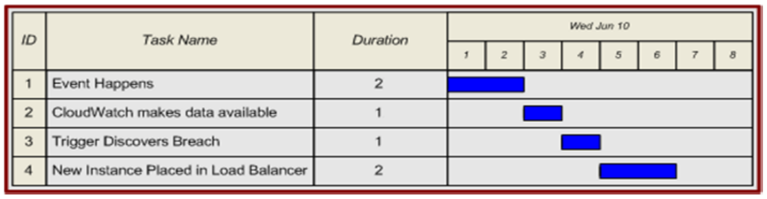
Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud. It provides developers with a highly scalable, flexible, and cost-effective capability to publish messages from an application and immediately deliver them to subscribers or other applications.

Auto Scaling events such as the launch or termination of an EC2 instance can be configured to publish to SNS topics which you will see in the hands-on exercise below.

Timing Matters

Let’s focus on costs related to using Auto Scaling. There are two important concepts that directly affect the cost of AWS and the manner in which your application will scale:

Scaling Takes Time
Consider the graph below. In most situations a considerable amount of time passes between when there is the need for a scaling event, and when the event happens.



ID	Task Name	Duration	Wed Jun 10							
			1	2	3	4	5	6	7	8
1	Event Happens	2	■	■						
2	CloudWatch makes data available	1			■					
3	Trigger Discovers Breach	1				■				
4	New Instance Placed in Load Balancer	2					■	■		

- In this example the rule says that we need to be in a particular condition for at least 2 minutes.
- CloudWatch is the underlying data collection system that monitors statistics such as CPU utilization. It is a polling protocol, and in general takes 60 seconds to aggregate the data.
- Auto Scaling is also a polling system, and it takes another 60 seconds.
- Then there is boot time for your server. A large, complex, server may take many minutes to launch.
- Finally, the load balancer needs to poll the server for a few cycles before it is comfortable that the server is healthy and accepting requests.

The Minimum Unit of Cost for EC2 is One Hour
It does not matter whether an EC2 instance runs for 60 seconds or 60 minutes: AWS bills for the full hour. Accordingly it is very important to avoid a short-cycle situation where a server is added to the fleet for 10 minutes, decommissioned, and then another server is added a few minutes later.

For this reason, we recommend that you configure your Auto Scaling policies to scale up quickly and scale down slowly. This will allow the application to better respond to increased traffic loads after a scale up event, as well as make more economical use of the AWS hourly billing cycle.

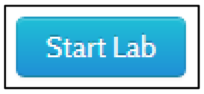
Auto Scaling API Command-Line Tools

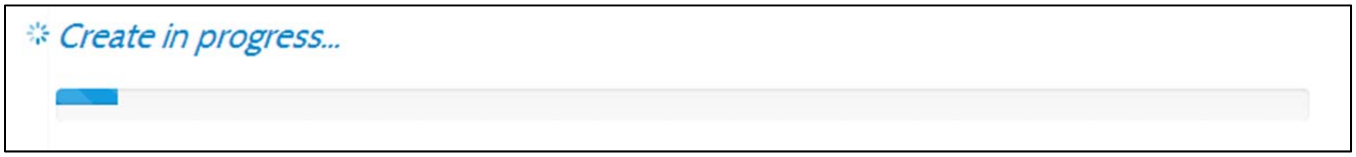
Auto Scaling configuration is not visible in the AWS Management Console, so in order to control Auto Scaling you use the command line tools or API. For this lab we have pre-installed the command-line tools onto an EC2 instance which you can use to configure your Auto Scaling environment.

Hands-On Exercise

Start your *qwikLAB*™

1. Start your *qwikLAB*™
   Use the 'Start Lab' button to start your lab.
   (Hint: If you are prompted for a token, please use one you've been given or have purchased.)





You will see the lab creation in progress.

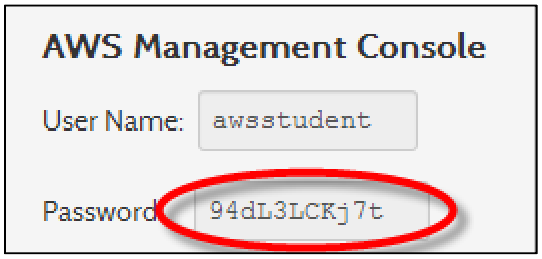
2. Note a few properties of the lab.

   a. Duration - The time the lab will run for before shutting itself down.
   b. Setup Time - The estimated lab creation time on starting the lab.
   c. AWS Region - The AWS Region the lab resources are being created in.

Open the AWS Management Console

3. Copy the Password for the AWS Management Console provided by *qwik*LAB™ for your lab.

   a. Hint: selecting the value shown and using Ctrl (or Command)+C works best

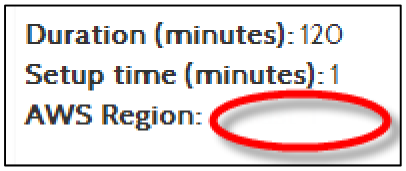


AWS Management Console

User Name: awsstudent

Password: 94dL3LCKj7t

4. Note the AWS Region set for your lab in *qwikLAB*™



Duration (minutes): 120
Setup time (minutes): 1
AWS Region:

5. Click the 'Open Console' button.

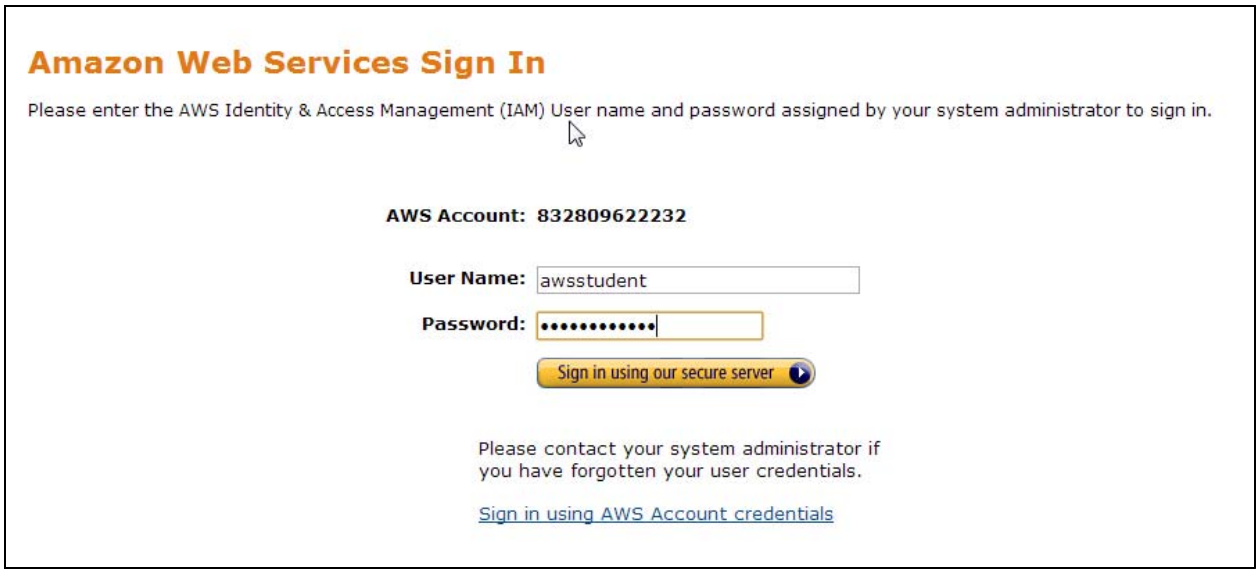
6. Make sure that you are not logged into any other instances of the AWS console (in a student account or your own account), as this may cause conflicts when you open the console and log in below for this lab.

7. Login to the AWS Management Console

   Enter the User Name '**awsstudent**' and paste the password you copied from the lab details in *qwikLAB*™ into the Password field.

   Click on the 'Sign in using our secure server' button.

   In this step you logged into the AWS Management Console using login credentials for a user provisioned via AWS Identity Access Management in an AWS account by *qwikLAB*™.



Amazon Web Services Sign In

Please enter the AWS Identity & Access Management (IAM) User name and password assigned by your system administrator to sign in.

AWS Account: 832809622232

User Name: awsstudent

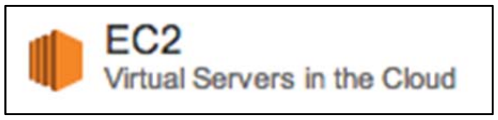
Password: ••••••••••••

Please contact your system administrator if you have forgotten your user credentials.

Sign in using AWS Account credentials

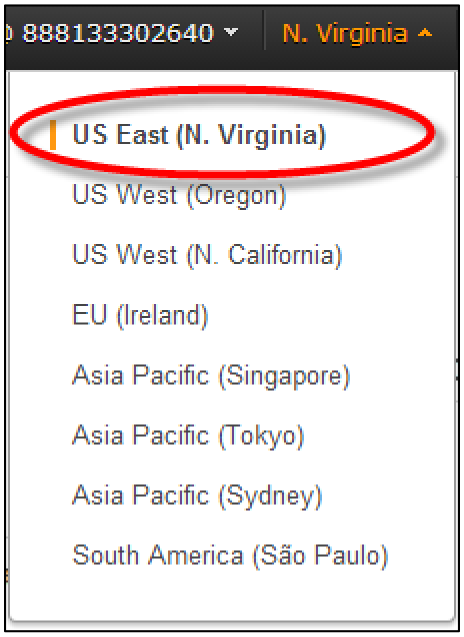
Select the Amazon EC2 Service

8. Select "EC2" from the Console Home.



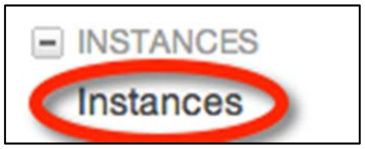
Confirm your AWS Region

9. Select or confirm that the same AWS Region is already set in the AWS Management Console

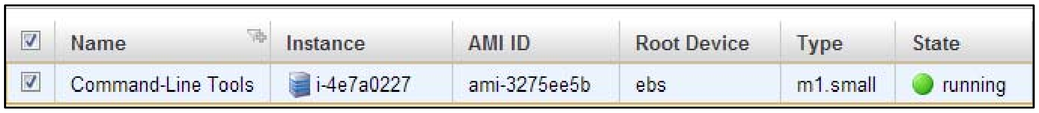


Identify the Command-Line Tools Instance

10. Click on Instances.

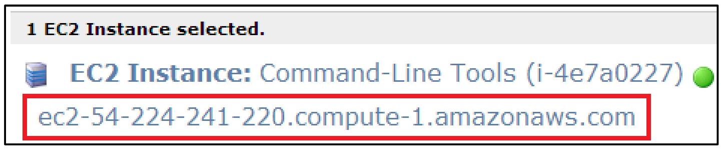


11. Select the m1.small instance named “Command-Line Tools”. This instance has been pre-configured to include the command-line tools necessary to administer Auto Scaling.



✓	Name	Instance	AMI ID	Root Device	Type	State
✓	Command-Line Tools	i-4e7a0227	ami-3275ee5b	ebs	m1.small	● running

12. Copy the public DNS name of the command-line tools instance.



1 EC2 Instance selected.
EC2 Instance: Command-Line Tools (i-4e7a0227)
ec2-54-224-241-220.compute-1.amazonaws.com

Connect to the Command-Line Tools Instance

13. Use the public DNS name of your command-line tools instance to connect via SSH. For instructions, see Appendix A - Connecting to your EC2 Instance via SSH.

Gather Lab Resource Details

*qwik*LAB™ created for you a CloudFormation stack containing an Elastic Load Balancer, EC2 Security Group, and an EC2 Key Pair. You will be using those resources in this lab and referencing them by their names.

14. The names of your resources are in a file '/home/ec2-user/lab-details.txt' in the instance to which you connected. Concatenate those details from the file to your instance standard out using the below command:

```
cat /home/ec2-user/lab-details.txt
ElasticLoadBalancer,stack-136-ElasticL-49ICDQ2I88AE
Ec2SecurityGroup,stack-1366573233-Ec2SecurityGroup-1S7OZJ8S6W73Y
AMIId,ami-ecbe2adc
KeyName,qwiklab-l33-5023
AvailabilityZone,us-west-2b
```

Note the values you see in red will be different when you run this lab. Where red is used later it may denote where you need to use the values from your own lab-details.txt when running commands in this lab.

15. You should copy-paste those details to a plain text editor on your own computer so you can copy-paste the same details as needed into commands you copy-paste from this lab guide into the same text editor so you can edit the command, add appropriate values where needed, and copy-paste them to your terminal (shell) to run them.

Create a Launch Configuration

16. This launch configuration specifies a machine image (Amazon Machine Image or AMI) that will launch when Auto Scaling adds new servers. This should be the value from AMIId in your lab details file.

```
as-create-launch-config --image-id PasteYourAMIIdHere --instance-type
t1.micro --key PasteYourKeyNameHere --group PasteYourSecurityGroupHere --
user-data-file as-bootstrap.sh --launch-config lab-lc

OK-Created launch config
```

The parameters for this command are as follows:

- image-id A 64-bit Amazon Linux AMI
- instance-type An EC2 Instance type. We will use the t1.micro instance type here.
- key The name of an EC2 Key Pair created by *qwik*LAB™ for you.
- group The EC2 Security Group created for you in the lab via CloudFormation.
- user-data-file Your command-line tools instance includes a file /home/ec2-user/ as-bootstrap.sh which will be used to bootstrap the configuration on your Auto Scaling web server instances.
- launch-config The name of this Auto Scaling Launch Configuration. We used **lab-lc**.

Create an Auto Scaling Group

We are going to launch our EC2 Instances into a single AZ in the AWS region you are running this lab in.

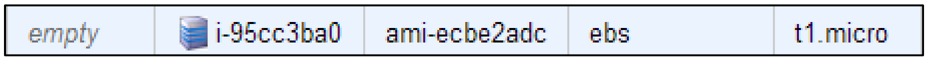
17. Copy-paste the below into your plain-text editor and edit the command to use values you already have from previous commands in this lab and lab-details.txt.

```
as-create-auto-scaling-group lab-as-group --availability-zones
PasteYourAvailabilityZoneHere --launch-configuration lab-lc --load-
balancers PasteYourElasticLoadBalancerHere --max-size 5 --min-size 1

OK-Created AutoScalingGroup
```

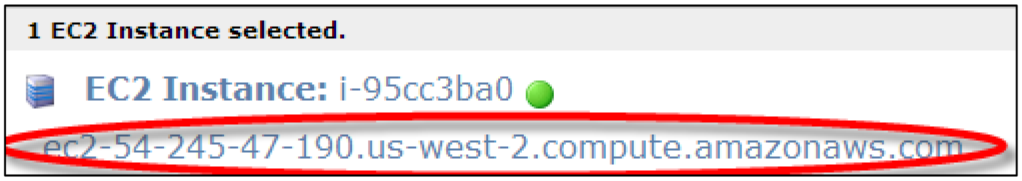
Verify that the Servers Launched

18. Use the AWS Management Console to inspect your EC2 Instances. In addition to your command-line tools instance, you should see a new instance created by the Auto Scaling Group. You may need wait a few minutes or refresh your browser before it appears.
19. Select the new t1.micro instance.



empty	i-95cc3ba0	ami-ecbe2adc	ebs	t1.micro

20. Copy the public DNS name of your new instance into your clipboard.

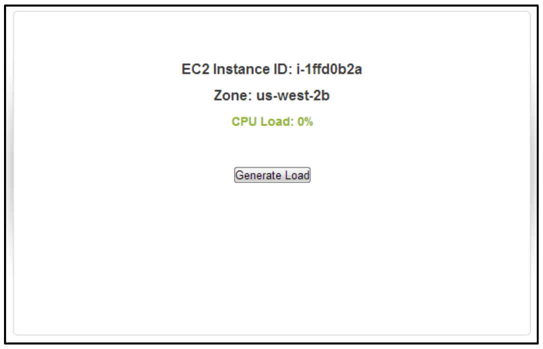


1 EC2 Instance selected.

EC2 Instance: i-95cc3ba0

ec2-54-245-47-190.us-west-2.compute.amazonaws.com

21. Paste the public DNS name into your browser to verify that it is running properly. You should see the application bootstrapped via the launch configuration display information as below:



EC2 Instance ID: i-1ffd0b2a

Zone: us-west-2b

CPU Load: 0%

Verify Auto Scaling Behavior

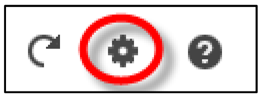
22. Return to the AWS Management Console and terminate the t1.micro web server instance. In a few minutes it will re-appear, because Auto Scaling will notice that the fleet size is below minimum size.

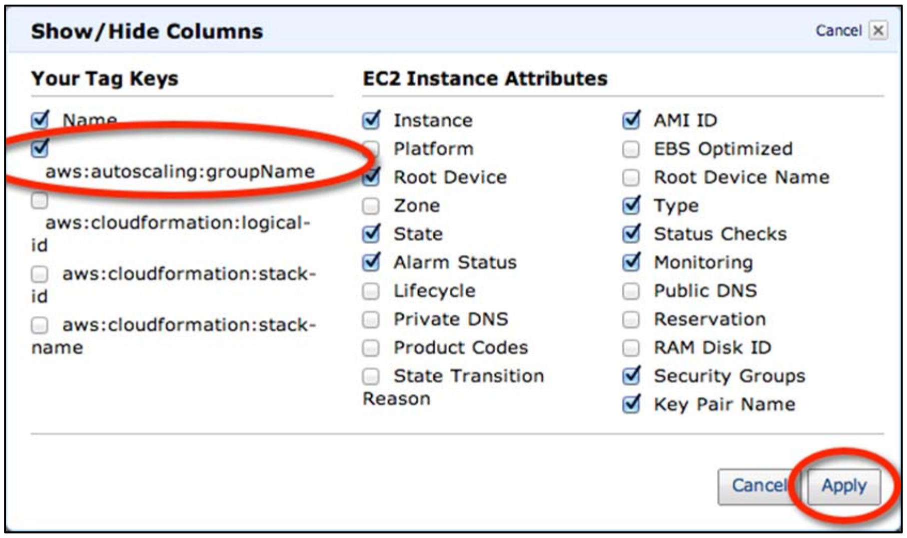
23. Try doing the same thing by shutting down the instance (rather than outright terminating it) either by logging into the new instance and issuing a shutdown command (`sudo shutdown -h now`) or by right-clicking on the instance and selecting **Stop** from the AWS Management Console. Notice that Auto Scaling will detect that the instance is non-responsive and will automatically terminate it and launch a replacement instance for you.

Tagging Auto Scaling Resources

Notice that the Auto Scaling instances are launched without names. There are two ways to help you better identify these instances. The first is by adding a new column to the Management Console.

24. Click on the Show/Hide button ("gear" icon), then select the **aws:autoscaling:groupName** option under **Your Tag Keys**. Click Apply and then click on the Refresh button.





Show/Hide Columns Cancel

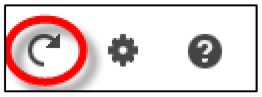
Your Tag Keys

- [✓] Name
- [✓] aws:autoscaling:groupName
- [ ] aws:cloudformation:logical-id
- [ ] aws:cloudformation:stack-id
- [ ] aws:cloudformation:stack-name

EC2 Instance Attributes

- [✓] Instance
- [ ] Platform
- [✓] Root Device
- [ ] Zone
- [✓] State
- [✓] Alarm Status
- [ ] Lifecycle
- [ ] Private DNS
- [ ] Product Codes
- [ ] State Transition Reason
- [✓] AMI ID
- [ ] EBS Optimized
- [ ] Root Device Name
- [✓] Type
- [✓] Status Checks
- [✓] Monitoring
- [ ] Public DNS
- [ ] Reservation
- [ ] RAM Disk ID
- [✓] Security Groups
- [✓] Key Pair Name

Cancel Apply



Auto Scaling automatically creates and populates a tag called aws:autoscaling:groupName for your Auto Scaling instances. The second way you can better identify your Auto Scaling instances is to modify your Auto Scaling group to populate the **Name** tag for you. We could have created a Name tag for the Auto Scaling group on creation by using the `--tag "k=Name, v=AS-Web-Server, p=true"` option.

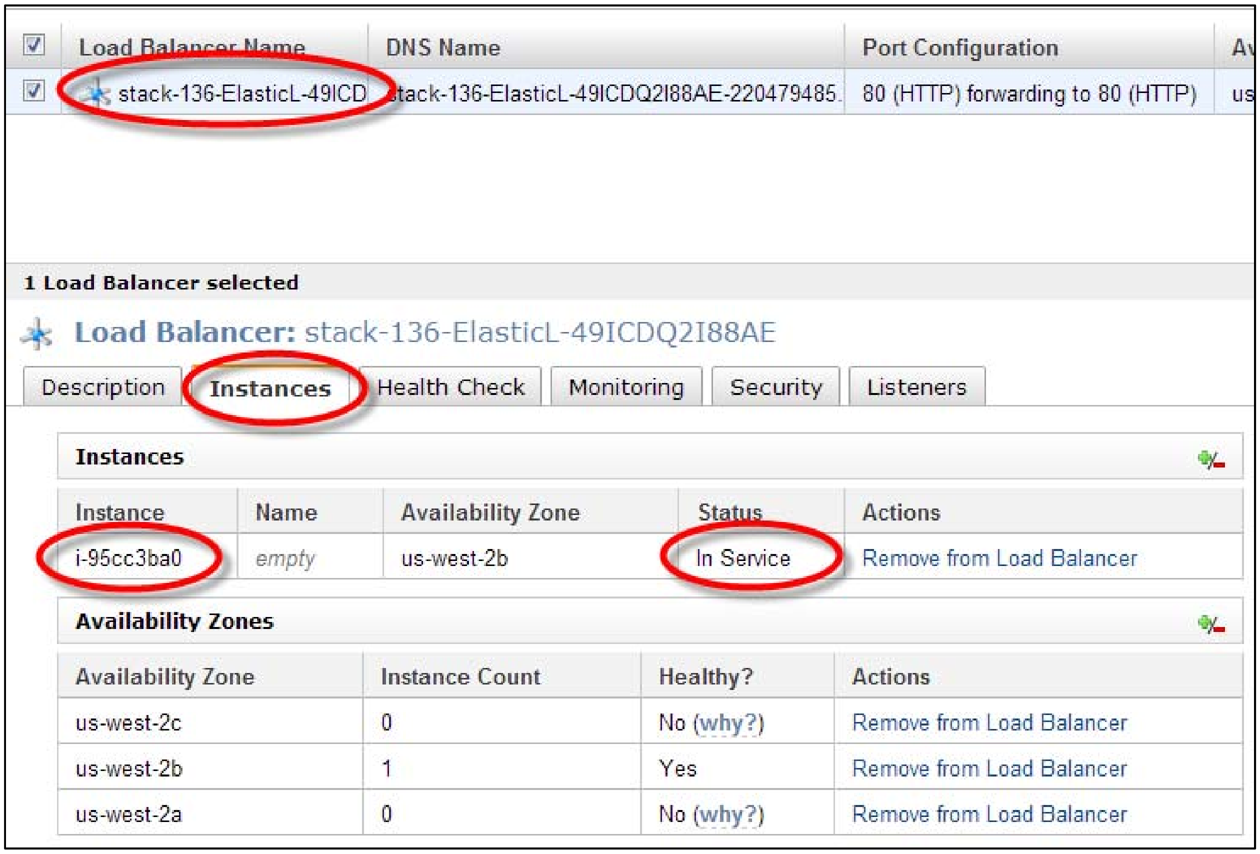
25. Since our Auto Scaling group already exists, let's just modify the existing tags with the following command:

```
as-create-or-update-tags --tag "id=lab-as-group, t=auto-scaling-group,
k=Name, v=AS-Web-Server, p=true"

OK-Created/Updated tags
```

Auto Scaling Integration with Elastic Load Balancing

26. Navigate to the EC2 page (use the menu Services / EC2) in the console and click on the **Load Balancer** link on the left. If you look at your ELB, you will notice that the Auto Scaling instances are also being added to your ELB. This was configured with the `--load-balancers NameOfYourELB` option when creating the Auto Scaling Group.



☑	Load Balancer Name	DNS Name	Port Configuration	Av
☑	stack-136-ElasticL-49ICD	stack-136-ElasticL-49ICDQ2I88AE-220479485	80 (HTTP) forwarding to 80 (HTTP)	us

1 Load Balancer selected

Load Balancer: stack-136-ElasticL-49ICDQ2I88AE

Instances

Instance	Name	Availability Zone	Status	Actions
i-95cc3ba0	*empty*	us-west-2b	In Service	Remove from Load Balancer

Availability Zones

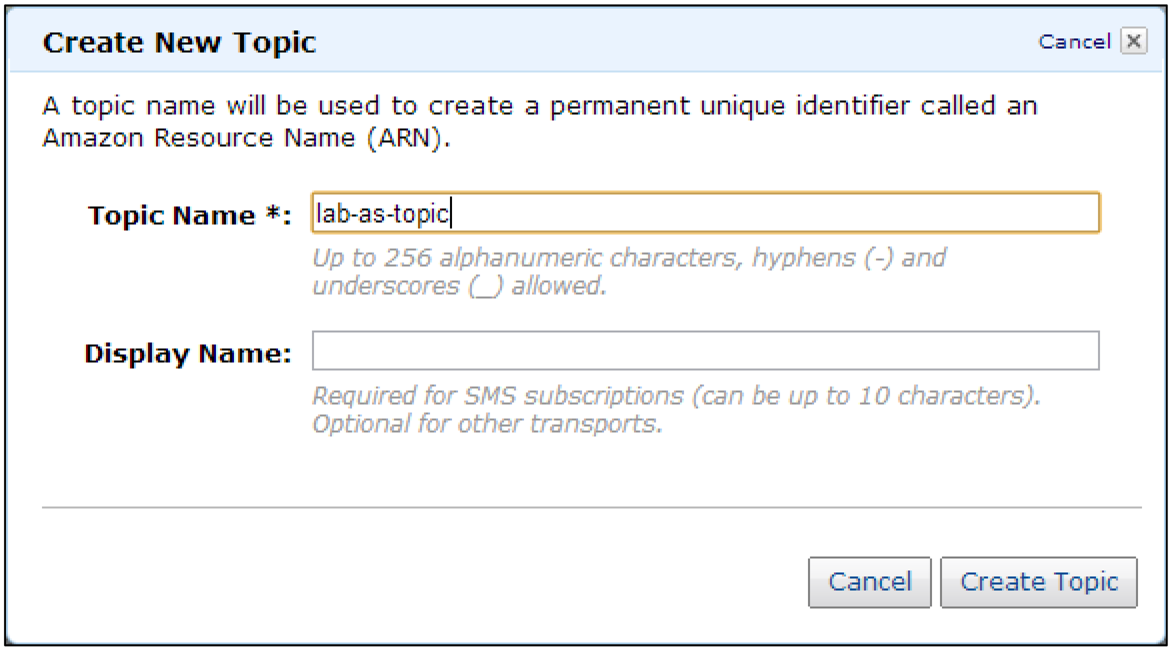
Availability Zone	Instance Count	Healthy?	Actions
us-west-2c	0	No (why?)	Remove from Load Balancer
us-west-2b	1	Yes	Remove from Load Balancer
us-west-2a	0	No (why?)	Remove from Load Balancer

Create SNS Topic

To be notified when Auto Scaling events occur, we need to create an SNS topic that we will use to send SNS notifications.

27. In the **AWS Management Console**, click on the **SNS** tab, and click on the **Create New Topic** button.

28. Create a topic name, in this example we use **lab-as-topic**, and click **Create Topic**.



Create New Topic

A topic name will be used to create a permanent unique identifier called an Amazon Resource Name (ARN).

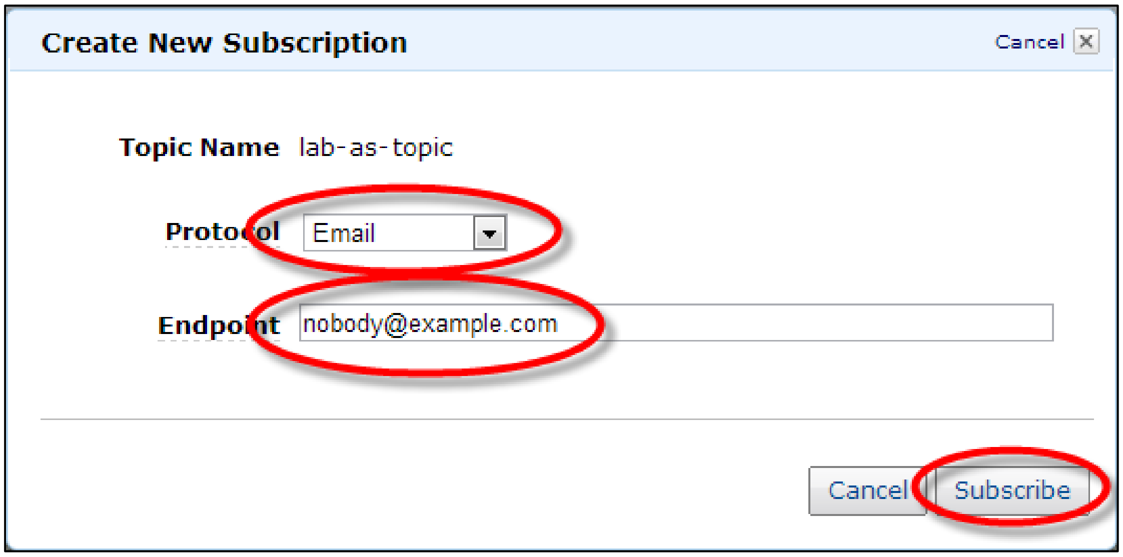
Topic Name *: lab-as-topic

Up to 256 alphanumeric characters, hyphens (-) and underscores (_) allowed.

Display Name:

Required for SMS subscriptions (can be up to 10 characters). Optional for other transports.

29. Click the **Create Subscription** button, select **Email** as the Protocol, enter the email address where you would like to receive email notifications as the **Endpoint**, and click **Subscribe**. Check your email address and click on the appropriate link to verify your email subscription to this topic.

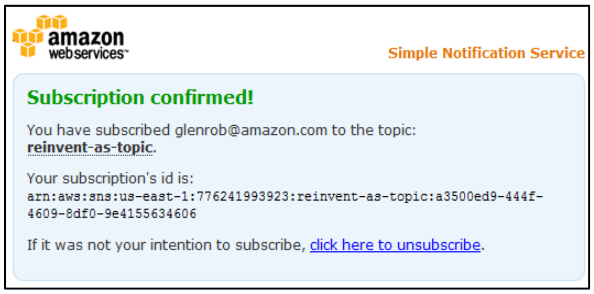


Create New Subscription

Topic Name lab-as-topic

Protocol Email

Endpoint nobody@example.com



Subscription confirmed!

You have subscribed glenrob@amazon.com to the topic: **reinvent-as-topic**.

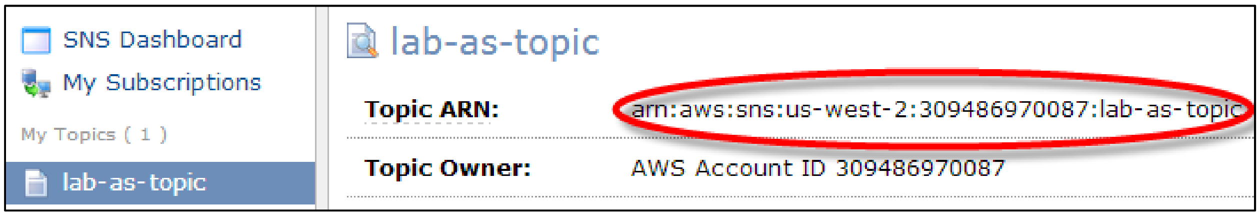
Your subscription's id is:

```
arn:aws:sns:us-east-1:776241993923:reinvent-as-topic:a3500ed9-444f-
4609-8df0-9e4155634606
```

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

Now that SNS is set up, we need the Amazon Resource Name (ARN) for the SNS topic to use with Auto Scaling.

30. Select the SNS topic, locate the Topic ARN, and copy the value for use in the next step.



Topic ARN: arn:aws:sns:us-west-2:309486970087:lab-as-topic
Topic Owner: AWS Account ID 309486970087

Create Auto Scaling Notifications

You can use as-describe-auto-scaling-notification-types to determine the types of Auto Scaling notifications that are supported.

```
as-describe-auto-scaling-notification-types
NOTIFICATION-TYPE autoscaling:EC2_INSTANCE_LAUNCH
NOTIFICATION-TYPE autoscaling:EC2_INSTANCE_LAUNCH_ERROR
NOTIFICATION-TYPE autoscaling:EC2_INSTANCE_TERMINATE
NOTIFICATION-TYPE autoscaling:EC2_INSTANCE_TERMINATE_ERROR
NOTIFICATION-TYPE autoscaling:TEST_NOTIFICATION
```

31. We will use the as-put-notifications-configuration command to create notifications for when instances are launched or terminated by our Auto Scaling group.

```
as-put-notification-configuration lab-as-group --topic-arn
PasteTheTopicARNHere --notification-types
autoscaling:EC2_INSTANCE_LAUNCH, autoscaling:EC2_INSTANCE_TERMINATE

OK-Put Notification Configuration
```

You should receive a test notification message from Auto Scaling confirming this configuration.

Create Auto Scaling Policies

Currently you have an Auto Scaling group that will verify that you have at least one running server. You can modify the number of running servers by manually manipulating the number of minimum servers with the command `as-update-auto-scaling-group lab-as-group --min-size #`. But instead of scaling manually, we are going to configure the Auto Scaling Group to automatically scale up whenever the average CPU of the web server farm is >= 50%.

We will use the `as-put-scaling-policy` command to create two Auto Scaling Policies that will increase the number of servers by 1 for scale up events and decrease the number of servers by 1 for scale down events (and wait 300 "cool down" seconds to let the environment settle before initiating additional scale up/down events).

Scale Up Policy

32. Execute the following command on your command-line tools instance to create a Scaling Policy to increase the desired capacity by one.

```
as-put-scaling-policy lab-scale-up-policy --auto-scaling-group lab-as-group
```

```
--adjustment=1 --type ChangeInCapacity --cooldown 300
```

Scale Down Policy

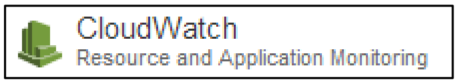
33. Execute the following command on your command-line tools instance to create the inverse Scaling Policy. This command is similar to the previous command, but notice the policy name is different and the adjustment parameter is enclosed in quotes with a negative value.

```
as-put-scaling-policy lab-scale-down-policy --auto-scaling-group lab-as-
group "--adjustment=-1" --type ChangeInCapacity --cooldown 300
```

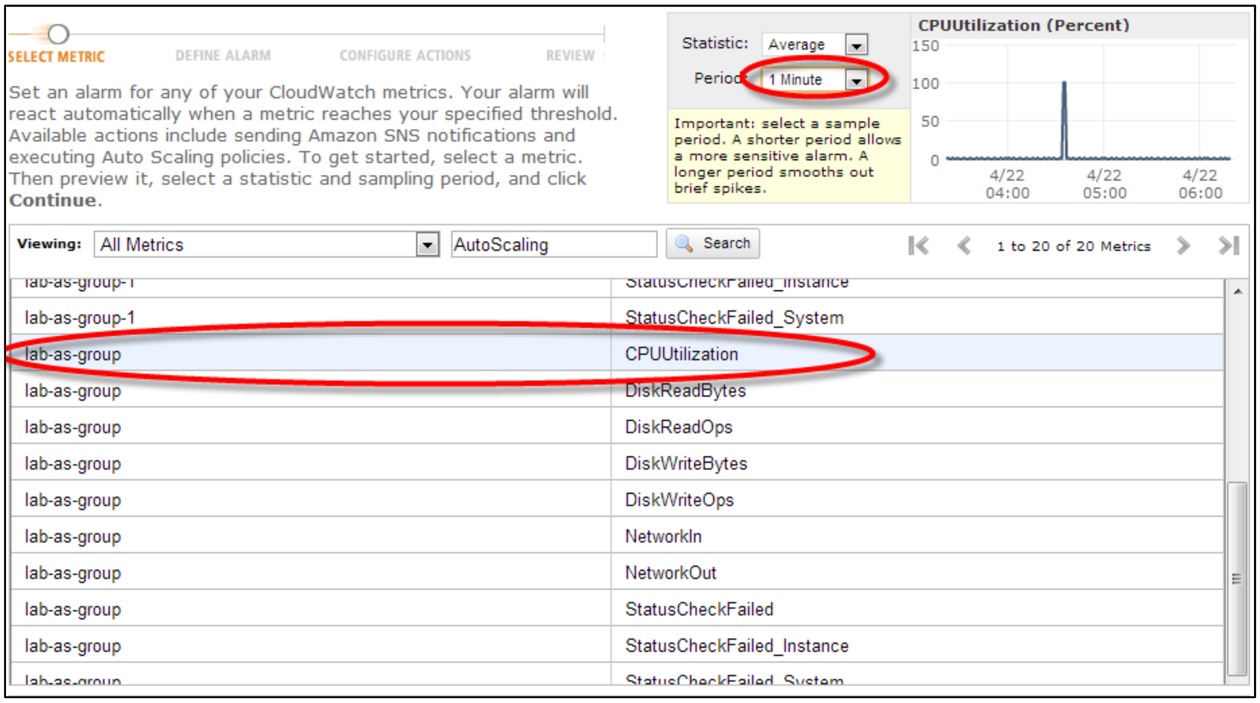
Create CloudWatch High CPU Alert

Now that we have the appropriate Auto Scaling policies, we need to create the appropriate CloudWatch triggers to initiate these policies. In this section we will create a CloudWatch alarm to monitor the aggregate average CPU of the Auto Scaling fleet and trigger the Lab-scale-up-policy.

34. In the **AWS Management Console**, go to the **CloudWatch** tab, and click on the **Create Alarm** button.



35. Perform a search for **AutoScaling**, select the **lab-as-group CPUUtilization** metric, change the Period option to **1 Minute**, and click **Continue**.



Set an alarm for any of your CloudWatch metrics. Your alarm will react automatically when a metric reaches your specified threshold. Available actions include sending Amazon SNS notifications and executing Auto Scaling policies. To get started, select a metric. Then preview it, select a statistic and sampling period, and click **Continue**.

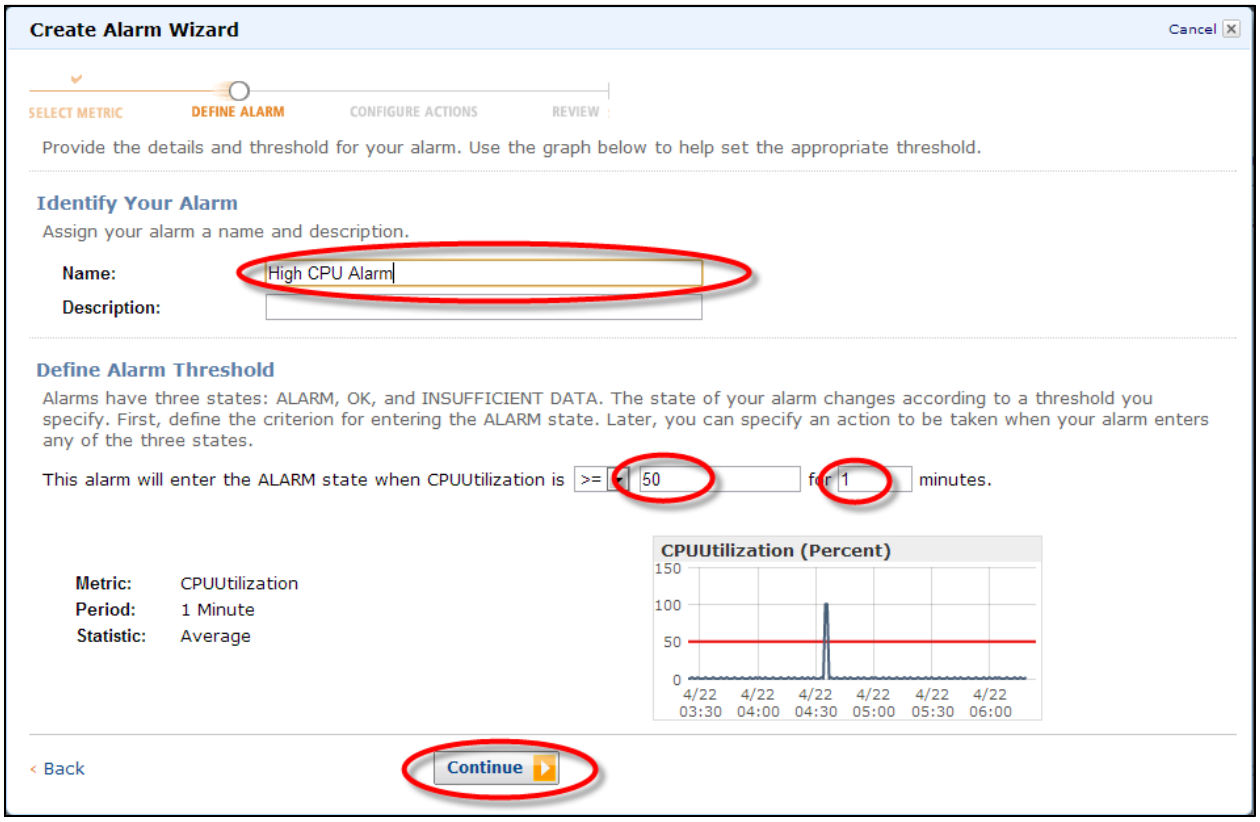
Statistic: Average
Period: 1 Minute

Important: select a sample period. A shorter period allows a more sensitive alarm. A longer period smooths out brief spikes.

Viewing: All Metrics AutoScaling

lab-as-group-1	StatusCheckFailed_Instance
lab-as-group-1	StatusCheckFailed_System
lab-as-group	CPUUtilization
lab-as-group	DiskReadBytes
lab-as-group	DiskReadOps
lab-as-group	DiskWriteBytes
lab-as-group	DiskWriteOps
lab-as-group	NetworkIn
lab-as-group	NetworkOut
lab-as-group	StatusCheckFailed
lab-as-group	StatusCheckFailed_Instance
lab-as-group	StatusCheckFailed_System

36. Give this high CPU alarm a **name** and **description**. Then configure the alarm for CPUUtilization to be **>= 50** for **1** minutes, and click **Continue**.



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

Identify Your Alarm

Assign your alarm a name and description.

Name: High CPU Alarm

Description:

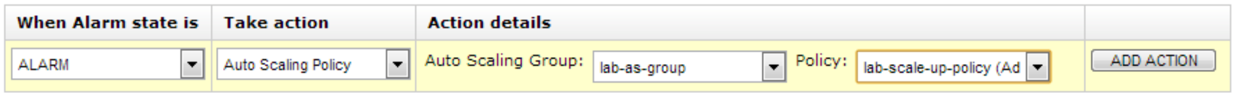
Define Alarm Threshold

Alarms have three states: ALARM, OK, and INSUFFICIENT DATA. The state of your alarm changes according to a threshold you specify. First, define the criterion for entering the ALARM state. Later, you can specify an action to be taken when your alarm enters any of the three states.

This alarm will enter the ALARM state when CPUUtilization is >= 50 for 1 minutes.

Metric:	CPUUtilization
Period:	1 Minute
Statistic:	Average

37. Configure the Take action to **Auto Scaling Policy**, Auto Scaling Group to **lab-as-group**, Policy to **Lab-scale-up-policy**, and click **Continue**. Review your settings and click the **Create Alarm** button.



When Alarm state is	Take action	Action details	
ALARM	Auto Scaling Policy	Auto Scaling Group: lab-as-group Policy: lab-scale-up-policy (Ad	ADD ACTION

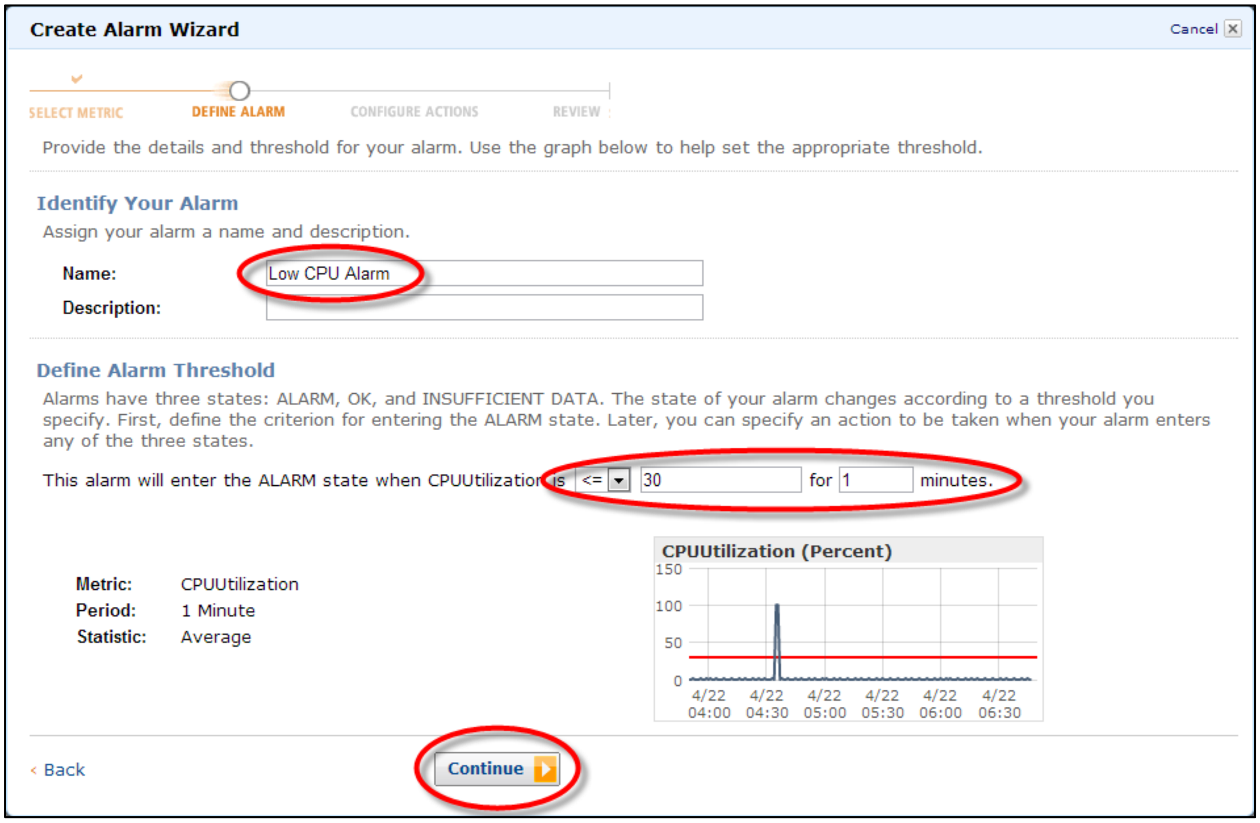
Create CloudWatch Low CPU Alert

Now we need to repeat the steps above, but configure the alert to be <=30% and to trigger the lab-scale-down-policy. The following screenshots only contain what is different from the previous steps.

38. From the **CloudWatch** tab, click on the **Alarm** link, and then click on the **Create Alarms** button.

39. Don't forget to select the **lab-as-group CPUUtilization** metric and then change the Period to **1 minute**. This is important so that we can set the trigger to minute-level granularity rather than 5 minute granularity.

40. Give this low CPU alarm a **Name** and **Description**. Then configure the alarm for CPUUtilization to be **<= 30** for **1** minutes, and click **Continue**.



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

Identify Your Alarm

Assign your alarm a name and description.

Name: Low CPU Alarm

Description:

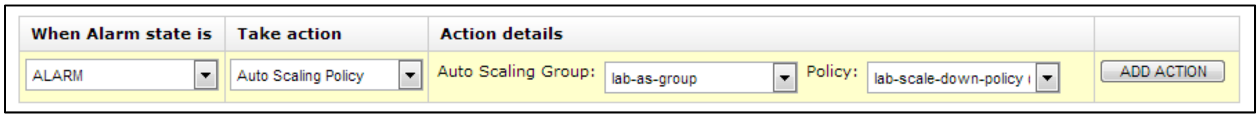
Define Alarm Threshold

Alarms have three states: ALARM, OK, and INSUFFICIENT DATA. The state of your alarm changes according to a threshold you specify. First, define the criterion for entering the ALARM state. Later, you can specify an action to be taken when your alarm enters any of the three states.

This alarm will enter the ALARM state when CPUUtilization is <= 30 for 1 minutes.

Metric:	CPUUtilization
Period:	1 Minute
Statistic:	Average

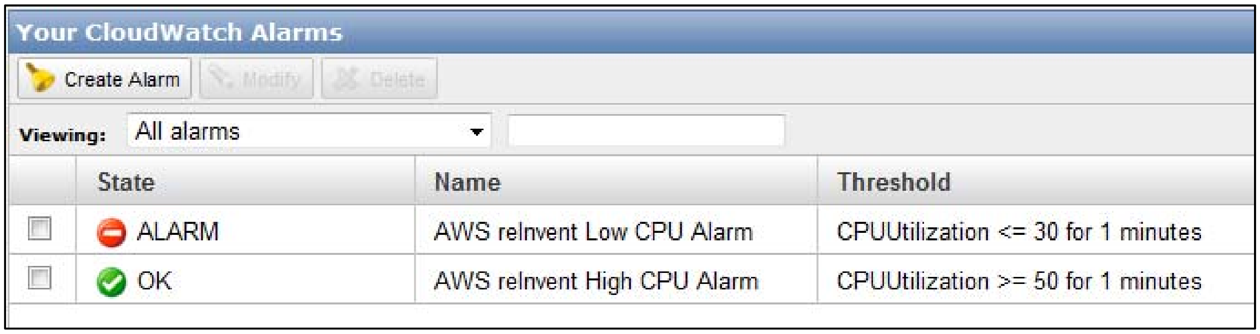
41. Configure Take action to **Auto Scaling Policy**, Auto Scaling Group to **lab-as-group**, Policy to **lab-scale-down-policy**, and click **Continue**.



When Alarm state is	Take action	Action details	
ALARM	Auto Scaling Policy	Auto Scaling Group: lab-as-group Policy: lab-scale-down-policy	ADD ACTION

Test Auto Scaling

Now all the pieces are in place to demonstrate auto scaling based on application usage. In summary, we created an Auto Scaling group with a minimum of 1 instance and a maximum of 5. We created auto scaling policies to increase and decrease the group by 1 instance and CloudWatch alarms to trigger these policies when the aggregate average CPU of the group is >= 50% and < 30% respectively. Currently 1 instance is running because the minimum size is 1 and the group is currently not under any load. Also note that the current CloudWatch alarms are in two different states:



	State	Name	Threshold
☐	ALARM	AWS reInvent Low CPU Alarm	CPUUtilization <= 30 for 1 minutes
☐	OK	AWS reInvent High CPU Alarm	CPUUtilization >= 50 for 1 minutes

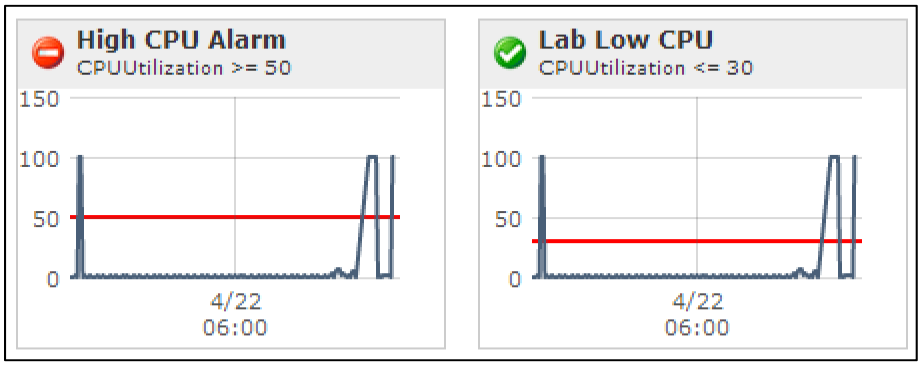
This is because the current group CPU Utilization is < 30%. However Auto Scaling is not removing any instances because the group size is currently at its minimum (1). Also remember that we set the cool down time for our auto scaling polices to 5 minutes (300 seconds), this is important to remember because this will influence how quickly you will be able to see the auto scaling activities.

42. Return to the EC2 service page and select **Load Balancers** in the navigation pane and copy the public DNS name of your ELB into the clipboard.

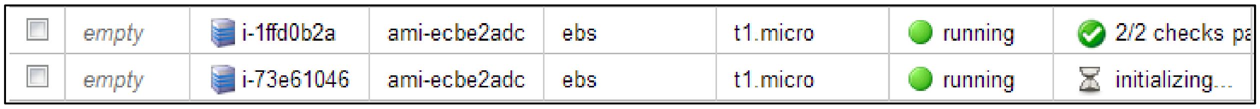
43. Open the ELB public DNS name in a browser window. When you refresh your browser, you should only see a single server. Click the **Generate Load** button and you will see the CPU Load jump up to 100% (you may have to refresh your browser to see the CPU Load increase). This button triggers a simple background process to copy, zip, and unzip ~1GB of nothing (/dev/zero) for 10-20 minutes.

44. Let's head back to the CloudWatch tab and in 3-4 minutes you should see the alarms switch to the Low CPU state changing to **OK** and the High CPU alarm state changing to **Alarm**.

45. Additionally, you should receive an email notification from Auto Scaling informing you that a scale up action was triggered.

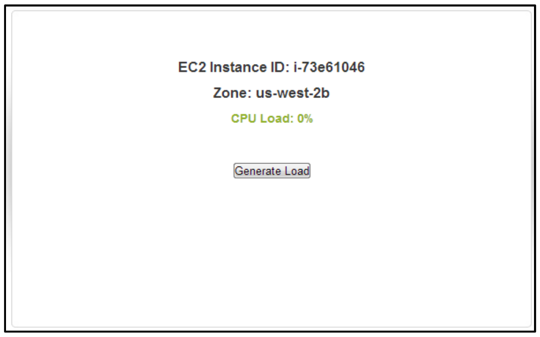


46. Head back over to the **EC2** tab -> **Instances** and you will see a new instance has been added to your group:



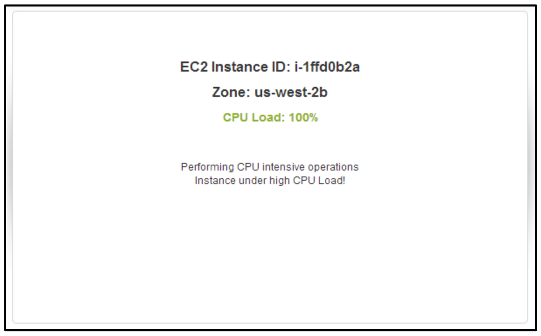
☐	empty	i-1ffd0b2a	ami-ecbe2adc	ebs	t1.micro	● running	✓ 2/2 checks pa
☐	empty	i-73e61046	ami-ecbe2adc	ebs	t1.micro	● running	initializing...

47. Go back to your ELB browser tab and refresh the ELB several times to see one server under heavy load, while the other is not:


EC2 Instance ID: i-73e61046

Zone: us-west-2b

CPU Load: 0%

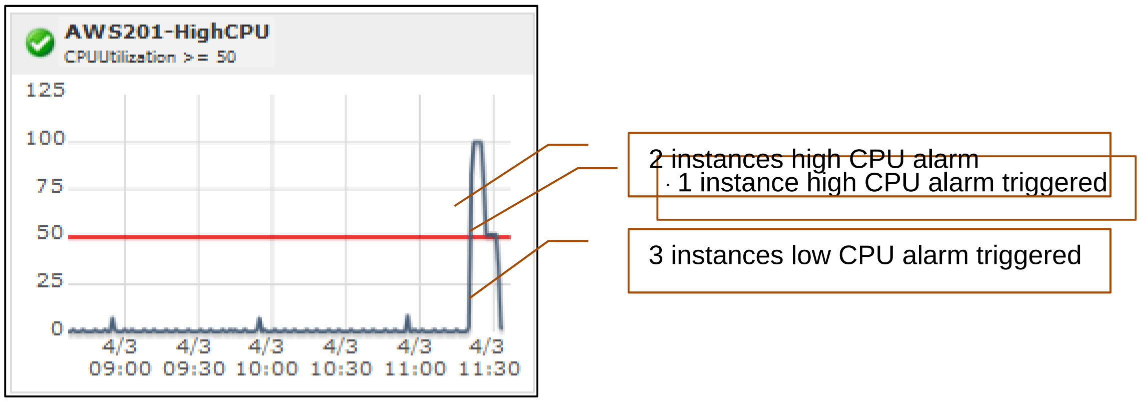

EC2 Instance ID: i-1ffd0b2a

Zone: us-west-2b

CPU Load: 100%

Performing CPU intensive operations
Instance under high CPU Load!

48. Finally, back in the CloudWatch tab, you can see the CPU utilization of the fleet. It is likely that you will actually trigger another Auto Scaling scale up event because your server fleet average CPU will equal 50% (one instance at ~100% and the other at ~0%), and we set the alarm to trigger at >= 50%. You might see something like the following.



49. After 15-20 minutes, your Auto Scaling fleet should have scaled up to 2-3 instances, then scaled back down to 1 instance. Also note that the instances were terminated in launch order, meaning that the “oldest” instances are terminated first.

Viewing Auto Scaling Activities

50. The Auto Scaling API provides a programmatic way to display all of the Auto Scaling activities that have occurred. Let’s use the **as-describe-scaling-activities** command to demonstrate this capability.

```
as-describe-scaling-activities
ACTIVITY 7833fa34-1532-40b7-911b-2daac10e1ec5            lab-as-group
InProgress
ACTIVITY 5efdbe89-e4c6-46b7-bcf0-637433376079 2013-04-22T07:28:58Z lab-as-
group Successful
ACTIVITY 2603ebc2-e207-4a4c-a083-f69631373c21 2013-04-22T07:12:55Z lab-as-
group Successful
```

Steady State Resources

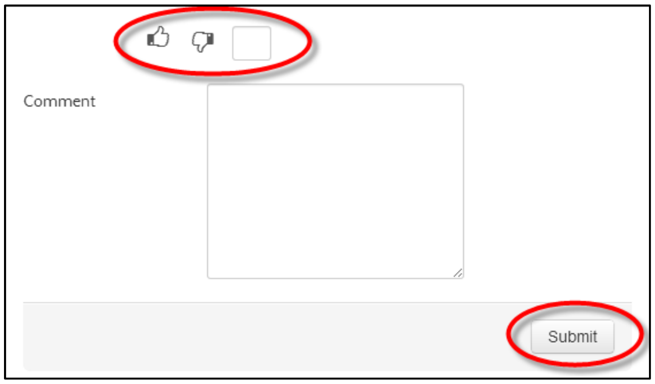
To demonstrate another use case for Auto Scaling Groups, your command-line tools instance has been launched by an Auto Scaling Group with `--min-size 1` and also `--max-size 1`. This means the Auto Scaling Group will never scale out nor scale in, but it will ensure you always have exactly one command-line tools instance available for use. This kind of configuration can be useful to ensure high-availability of steady state resources.

51. To see this in action, terminate your command-line tools instance and you'll see another instance launch to replace it. After a few minutes when the bootstrapping scripts are finished installing the command-line tools, your replacement instance will be ready for use.

End Lab

52. Sign-out of the AWS Management Console.
53. Click the End Lab button in *qwikLAB™*.

54. Give the lab a thumbs-up/down, or enter a comment and click Submit



Errors in this lab can be reported to aws-course-feedback@amazon.com

Summary

Congratulations! You have completed the Auto Scaling Hands-On lab, and now you know the basics of Auto Scaling with Amazon Web Services.

You have configured an elastic web farm which automatically changes size in response to changes in load. You have also explored the use of Auto Scaling Groups to ensure high-availability of steady state resources.

Now you can continue to explore Auto Scaling features such as scheduled actions and hopefully put Auto Scaling to work in your solutions.

For additional information, please visit the Auto Scaling documentation page: http://aws.amazon.com/documentation/autoscaling/.

There's also a quick reference card for the Auto Scaling command line tools which includes additional commands and options: http://awsdocs.s3.amazonaws.com/AutoScaling/latest/as-qrc.pdf.

Appendix A - Connecting to your EC2 Instance via SSH

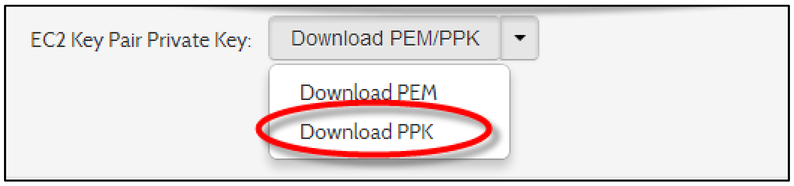
Windows

Download PuTTY

1. Download PuTTY to a location of your choice unless you already have PuTTY. http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

Download your EC2 Key Pair private key file

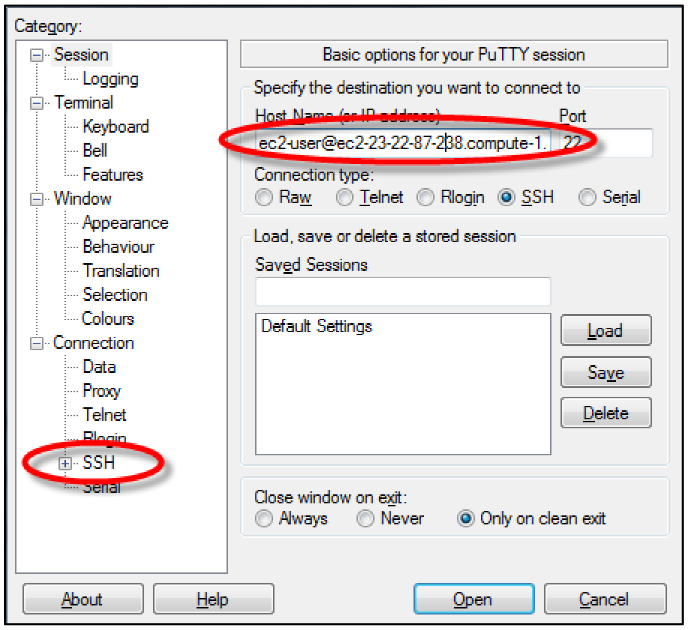
2. Go back to your lab in *qwikLAB*™.
3. Download the *qwikLAB*™ provided EC2 Key Pair private key file in the PuTTY compatible PPK format by clicking on the Download PPK option in the “Download PEM/PPK” drop-down.



4. Save the file to your Downloads directory (or some other directory of your choice.)

Connect to the EC2 Instance using SSH and PuTTY.

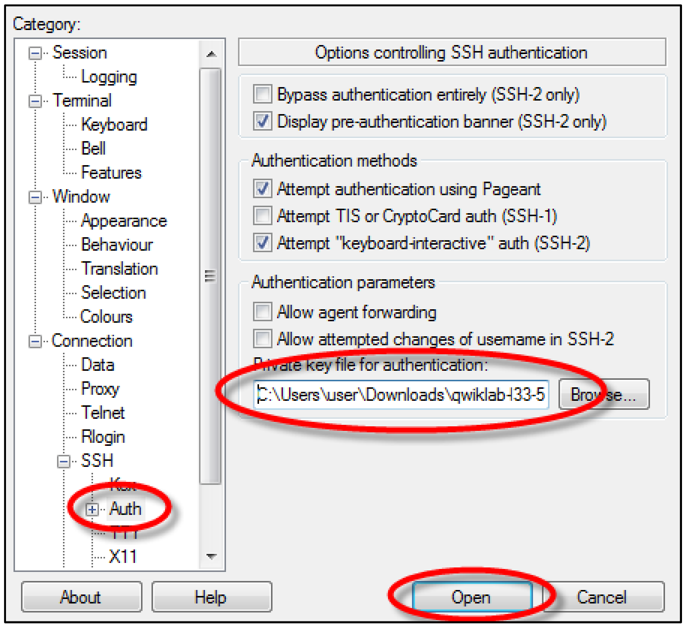
1. Open the putty.exe you downloaded or already had.
2. Enter ec2-user@<your EC2 hostname> into the Host Name input in Putty (Ctrl+v).
3. Expand the SSH category by clicking on it.



Host Name (or IP address): ec2-user@ec2-23-22-87-238.compute-1.
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

4. Select the Auth category by clicking on it (not the + symbol next to it).

5. Click Browse and locate the PPK file (ending in .ppk) in your Downloads directory or whatever other location you chose.
6. Click Open



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

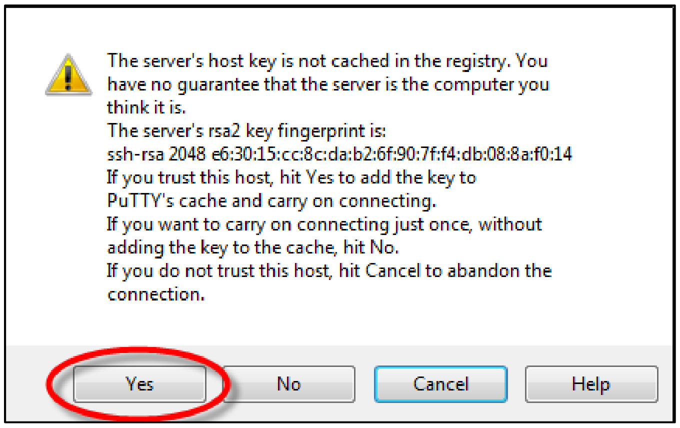
- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:
C:\Users\user\Downloads\qwiklab-l33-5 Browse...

Click Yes when prompted to allow a first connection to this remote SSH server.

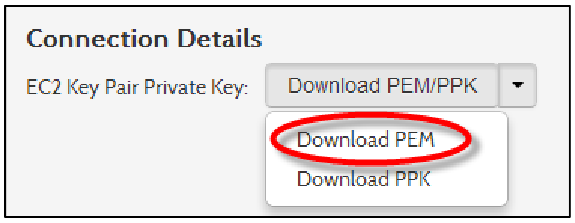


The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 e6:30:15:cc:8c:da:b2:6f:90:7f:f4:db:08:8a:f0:14
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

OS X and Linux

Download your EC2 Key Pair private key file

1. Go back to your lab in *qwikLAB*™.
2. Download the *qwikLAB*™ provided EC2 Key Pair private key file in the PEM format by clicking on the Download PEM option in the "Download PEM/PPK" drop-down.



3. Save the file to your Downloads directory (or some other directory of your choice.)

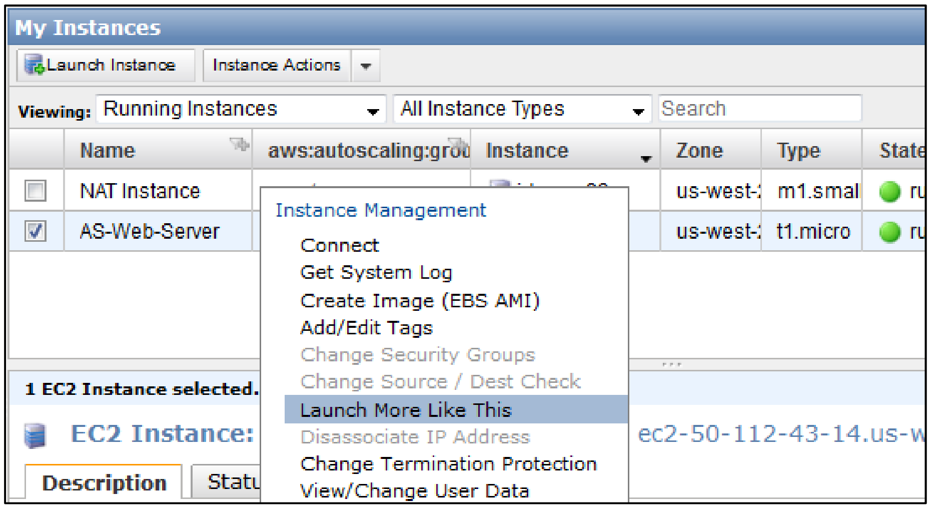
Connect to the EC2 Instance using the OpenSSH CLI client

1. Open the Terminal application.
2. Enter the below commands substituting the path/filename for the .pem file you downloaded from *qiwk*LAB™ and pasting ec2-user@<your EC2 hostname> to substitute the example below.

```
chmod 600 ~/Downloads/qwiklab-l33-5018.pem
ssh –i ~/Downloads/qwiklab-l33-5018.pem ec2-user@ec2-23-22-87-238.compute-1.amazonaws.com
```

Appendix B - Multi-User Auto Scaling Benchmark Example

Apache Bench is an excellent website benchmarking tool and we integrated an example use of this tool for performing a multi-user benchmarking example. First, you need to launch an independent Load Generation instance using the Auto Scaling AMI that you created in this lab. You should be an expert at launching instances now, so we will not repeat all of the steps for launching an EC2 instance. The easiest ways to do this is to **right-click** on an existing AS-Web-Server instance and select **Launch More Like This**.

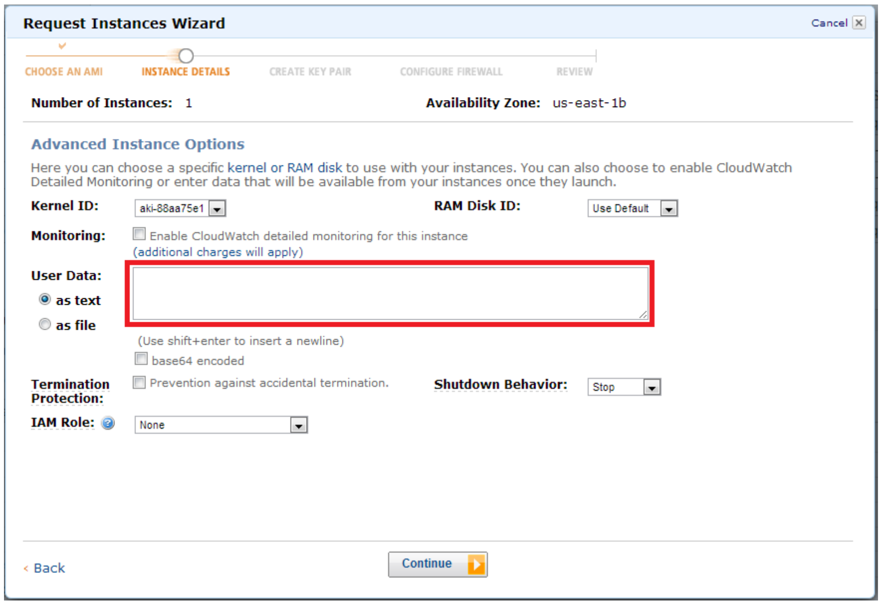


	Name	aws:autoscaling:grou	Instance	Zone	Type	State
☐	NAT Instance			us-west-:	m1.smal	● ru
☑	AS-Web-Server			us-west-:	t1.micro	● ru

1 EC2 Instance selected.

EC2 Instance: ec2-50-112-43-14.us-w

Your web server was launched with the User Data you specified when you created the Auto Scaling Launch Configuration. Remember to paste the contents of as-bootstrap.sh into the User Data field in the Launch Instance wizard. The as-bootstrap.sh file is located in /home/ec2-user/ on your Command-Line Tools instance.



Number of Instances: 1 Availability Zone: us-east-1b

Advanced Instance Options

Here you can choose a specific kernel or RAM disk to use with your instances. You can also choose to enable CloudWatch Detailed Monitoring or enter data that will be available from your instances once they launch.

Kernel ID: aki-88aa75e1 RAM Disk ID: Use Default

Monitoring: ☐ Enable CloudWatch detailed monitoring for this instance (additional charges will apply)

User Data: [✓ as text] as file

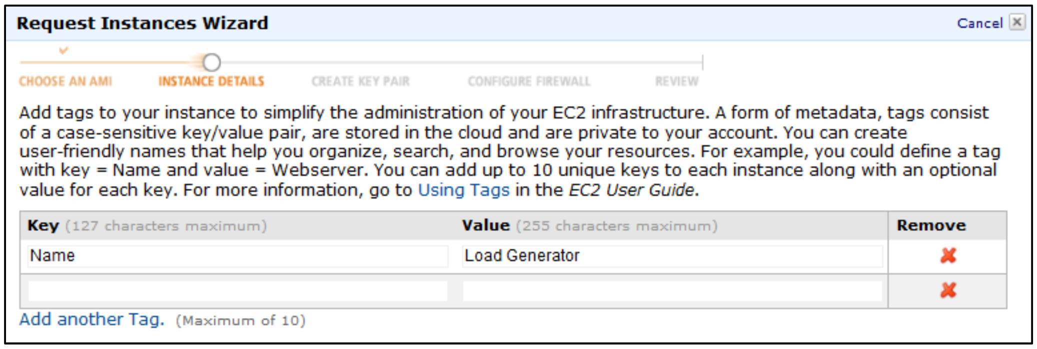
(Use shift+enter to insert a newline)

☐ base64 encoded

Termination Protection: ☐ Prevention against accidental termination. Shutdown Behavior: Stop

IAM Role: None

Select the default instance details and give the Load Generation instance a nice name.


Request Instances Wizard

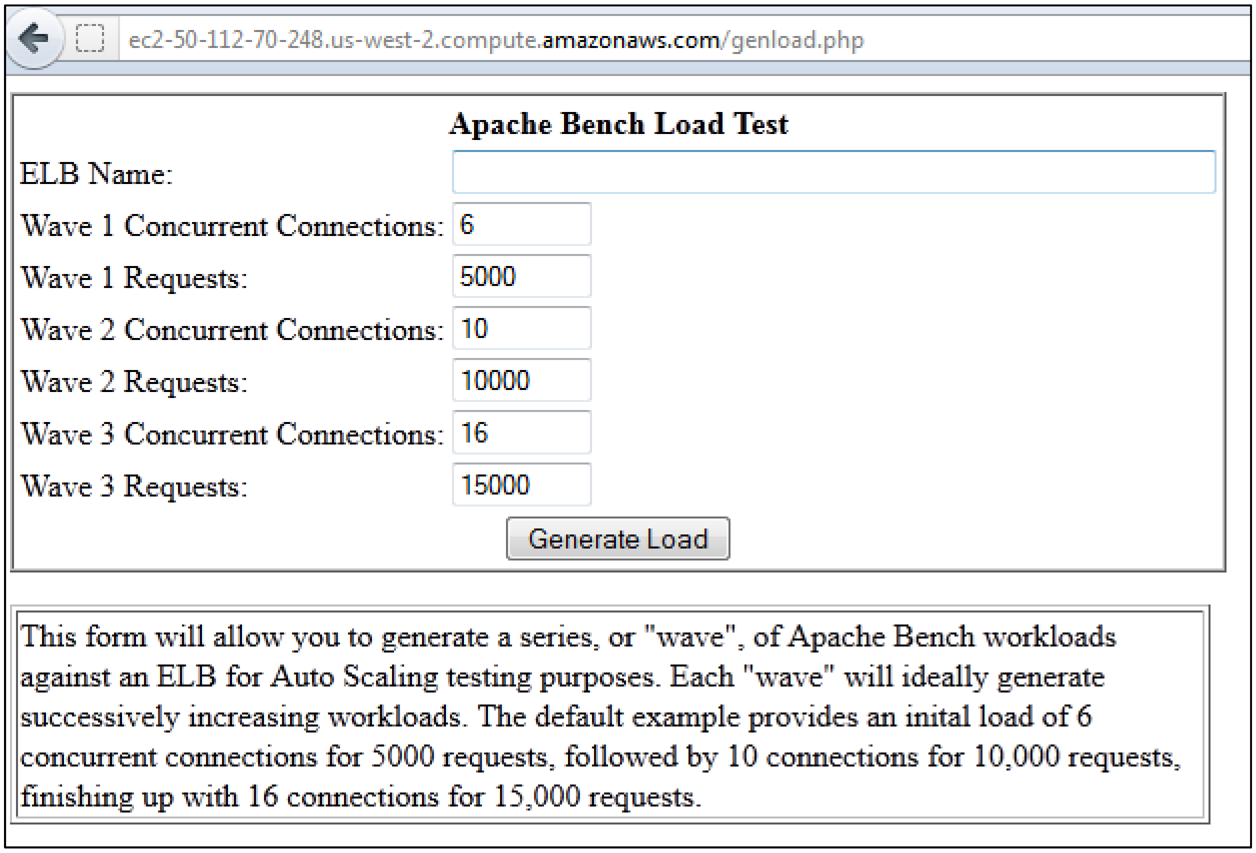
Add tags to your instance to simplify the administration of your EC2 infrastructure. A form of metadata, tags consist of a case-sensitive key/value pair, are stored in the cloud and are private to your account. You can create user-friendly names that help you organize, search, and browse your resources. For example, you could define a tag with key = Name and value = Webserver. You can add up to 10 unique keys to each instance along with an optional value for each key. For more information, go to Using Tags in the *EC2 User Guide*.

Key (127 characters maximum)	**Value** (255 characters maximum)	**Remove**
Name	Load Generator	✗
		✗

Add another Tag. (Maximum of 10)

Use your existing *qwik*LAB™ Key Pair and the **stack-xxxxxxx -Ec2SecurityGroup** security group, and launch the instance.

When the instance has launched and is running, find its DNS name and use another browser tab to connect to the **genload.php** webpage:


ec2-50-112-70-248.us-west-2.compute.amazonaws.com/genload.php

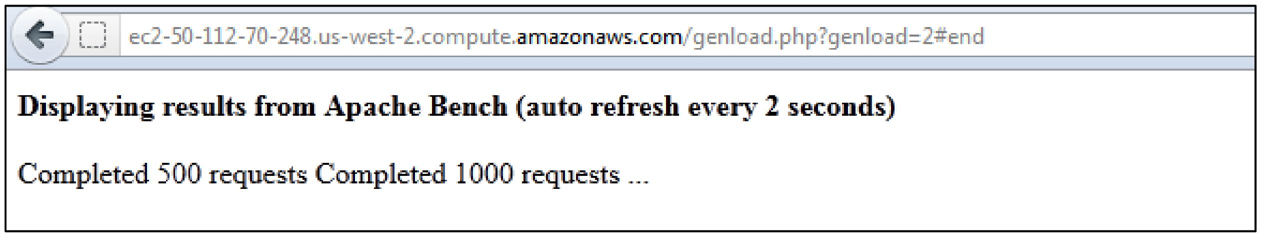
Apache Bench Load Test

ELB Name:	
Wave 1 Concurrent Connections:	6
Wave 1 Requests:	5000
Wave 2 Concurrent Connections:	10
Wave 2 Requests:	10000
Wave 3 Concurrent Connections:	16
Wave 3 Requests:	15000

This form will allow you to generate a series, or "wave", of Apache Bench workloads against an ELB for Auto Scaling testing purposes. Each "wave" will ideally generate successively increasing workloads. The default example provides an inital load of 6 concurrent connections for 5000 requests, followed by 10 connections for 10,000 requests, finishing up with 16 connections for 15,000 requests.

Paste your ELB public DNS name into the **ELB Name** text box and determine what load you want to put on your web site. To demonstrate Auto Scaling, we need to simulate several "waves" of traffic to show how Auto Scaling will increase to accommodate unexpected load, stabilize, and then increase again when the load increases. In this example we provide the ability to create up to 3 "waves" of increasing (or decreasing) web site utilization. We filled in default parameters of increasing concurrent connections and requests that we have found to provide a good example for our lab.

Once you insert your ELB name and click **Generate Load**, your load generator will start sending traffic to your ELB and web servers. Your load generator will continue to refresh showing you results from the performance test:

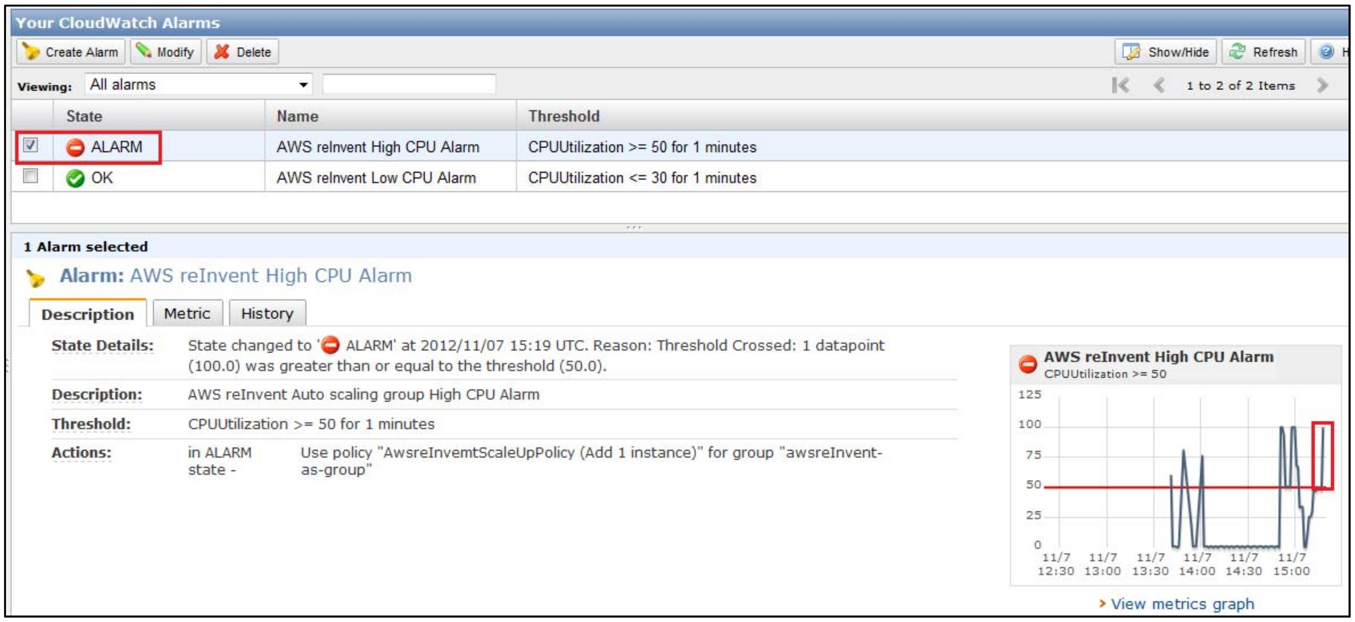


ec2-50-112-70-248.us-west-2.compute.amazonaws.com/genload.php?genload=2#end

Displaying results from Apache Bench (auto refresh every 2 seconds)

Completed 500 requests Completed 1000 requests ...

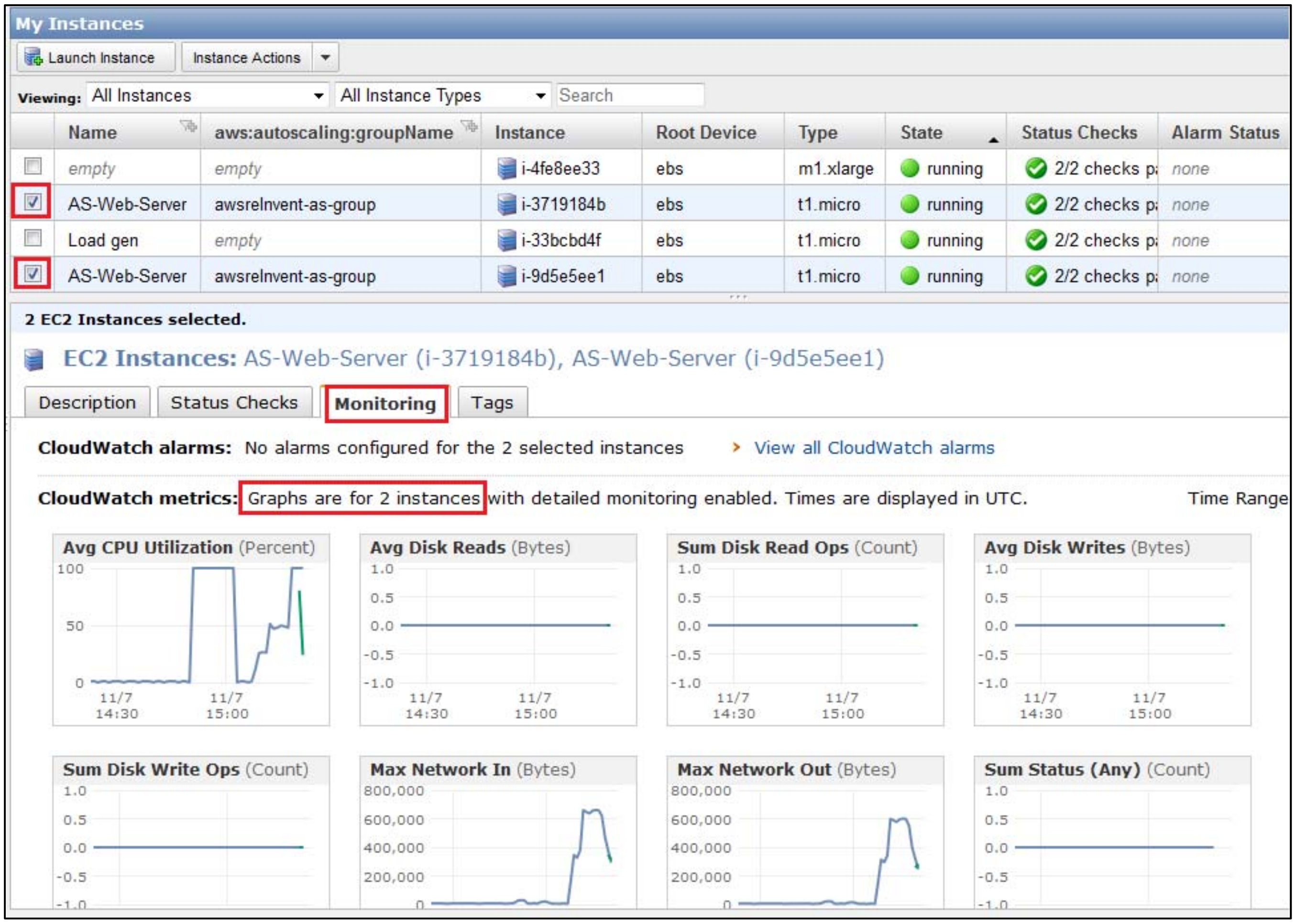
As the test progresses, you can watch what's happening through **CloudWatch** and your **EC2** tabs. First, through the **CloudWatch Alarms** screen, you can see when your **High CPU Alarm** alarm is triggered, resulting in a scale up activity:



State	Name	Threshold
ALARM	AWS reInvent High CPU Alarm	CPUUtilization >= 50 for 1 minutes
OK	AWS reInvent Low CPU Alarm	CPUUtilization <= 30 for 1 minutes

State Details:	State changed to 'ALARM' at 2012/11/07 15:19 UTC. Reason: Threshold Crossed: 1 datapoint (100.0) was greater than or equal to the threshold (50.0).
Description:	AWS reInvent Auto scaling group High CPU Alarm
Threshold:	CPUUtilization >= 50 for 1 minutes
Actions:	in ALARM state - Use policy "AwsreInvemtScaleUpPolicy (Add 1 instance)" for group "awsreInvent-as-group"

Second, through the **EC2 tab**, you can see Auto Scaling launch additional instances, but more importantly, you can select each of these instances and view all of their combined CPU metrics together by clicking on the **Monitoring** tab.



Viewing: All Instances | All Instance Types | Search

	Name	aws:autoscaling:groupName	Instance	Root Device	Type	State	Status Checks	Alarm Status
☐	empty	empty	i-4fe8ee33	ebs	m1.xlarge	● running	✓ 2/2 checks p	none
☑	AS-Web-Server	awsreInvent-as-group	i-3719184b	ebs	t1.micro	● running	✓ 2/2 checks p	none
☐	Load gen	empty	i-33bcbd4f	ebs	t1.micro	● running	✓ 2/2 checks p	none
☑	AS-Web-Server	awsreInvent-as-group	i-9d5e5ee1	ebs	t1.micro	● running	✓ 2/2 checks p	none

2 EC2 Instances selected.

EC2 Instances: AS-Web-Server (i-3719184b), AS-Web-Server (i-9d5e5ee1)

CloudWatch alarms: No alarms configured for the 2 selected instances › View all CloudWatch alarms

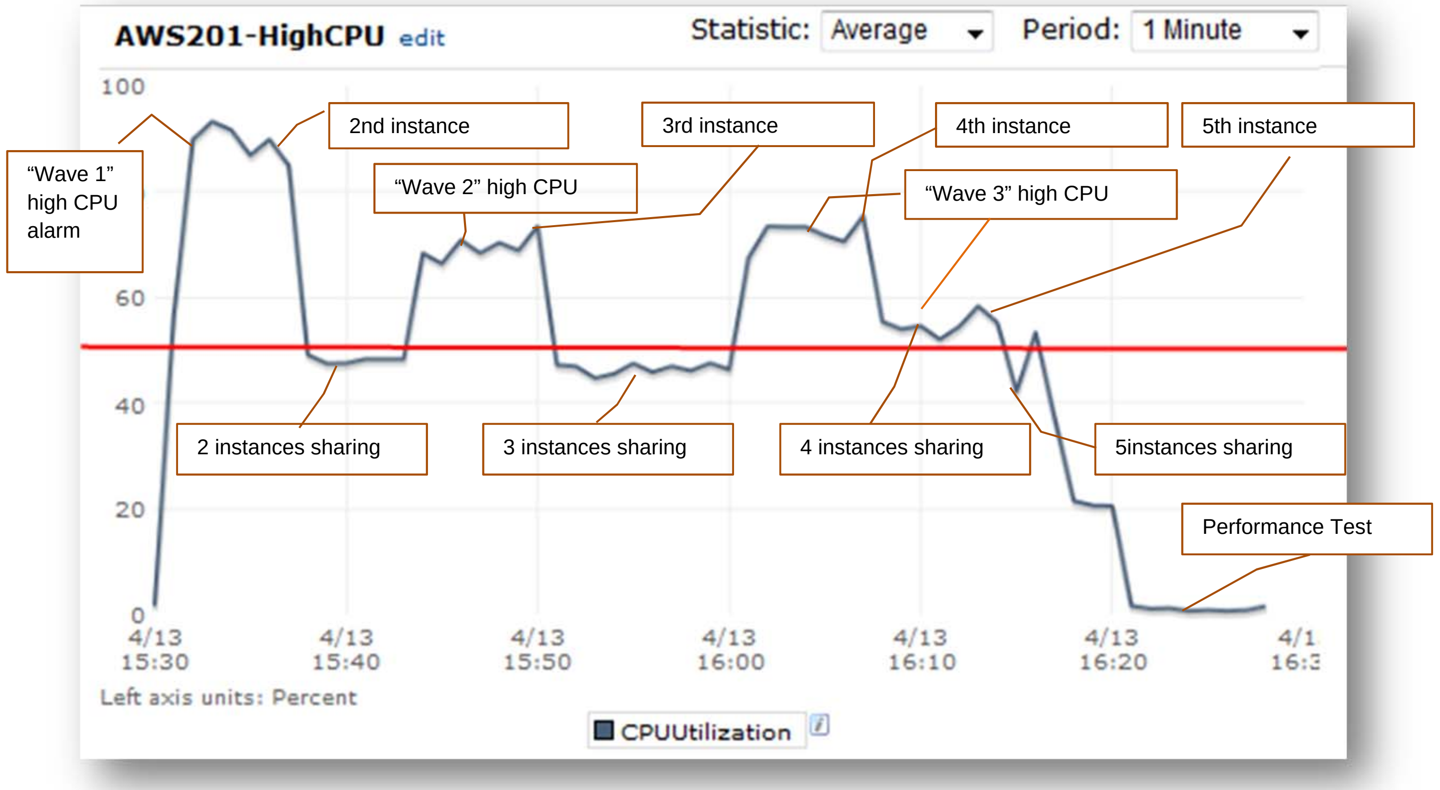
CloudWatch metrics: Graphs are for 2 instances with detailed monitoring enabled. Times are displayed in UTC. Time Range

Explanation of Graphs

CloudWatch High CPU Alarm Graph

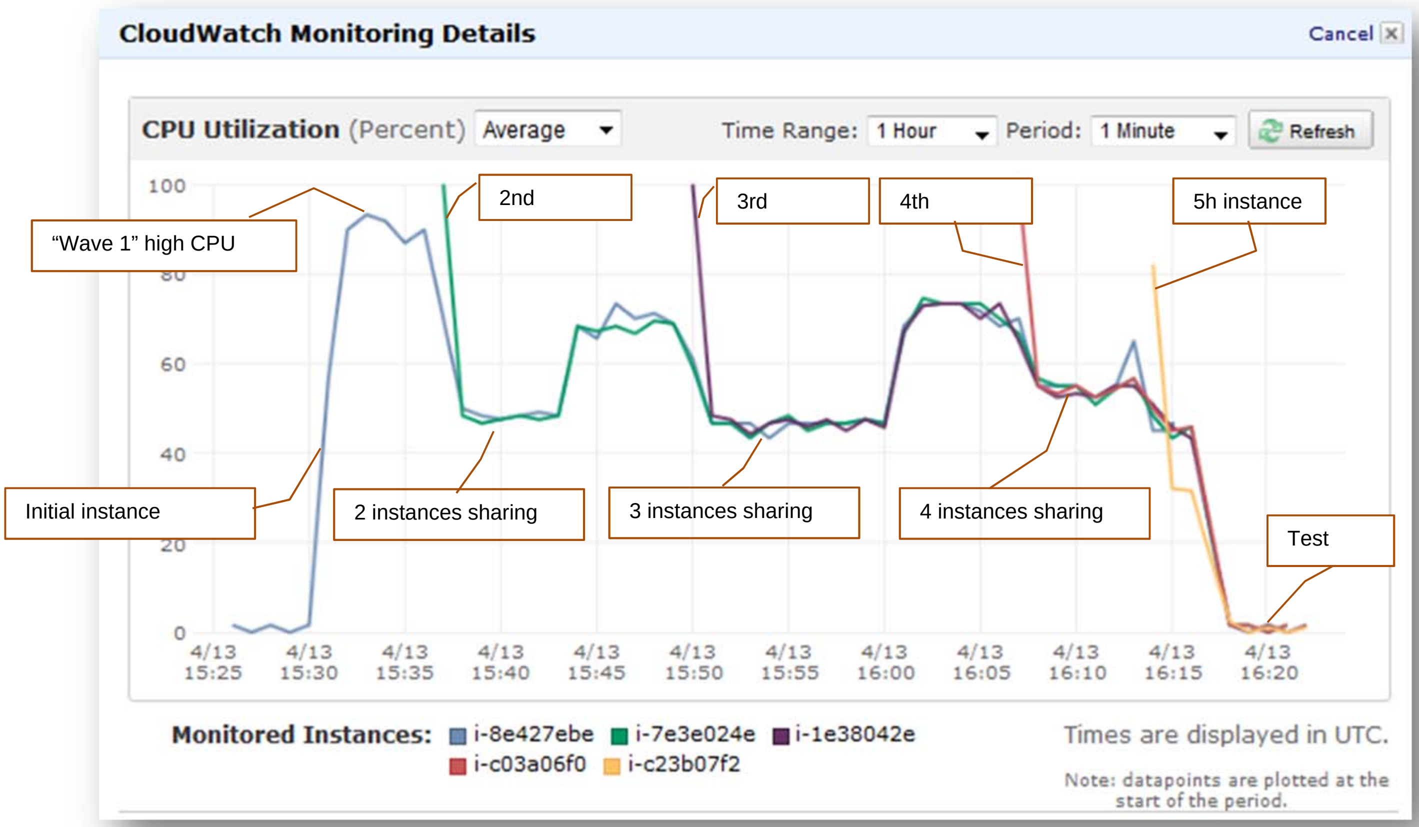
This graph covers the aggregate CPU utilization for the Auto Scaling group as a whole.

- “**Wave 1**” triggered a High CPU alarm and Auto Scaling launched a second instance. Aggregate CPU dropped below 50% with both instances until “Wave 2” brought additional traffic.
- “**Wave 2**” triggered another High CPU alarm and a 3rd instance is launched. 3 instances drop the aggregate CPU to below 50% again until “Wave 3” brought additional traffic.
- “**Wave 3**” causes the 4th and 5th instances to be launched because 4 instances were not able to bring the additional load under the 50% trigger point.



EC2 Combined CloudWatch Graphs

Selecting all Auto Scaling instances in the **EC2** tab, selecting the **Monitoring** tab, and clicking on the CPU utilization chart pulls up the following graph which depicts each individual instance's CPU utilization on the same graph such as the following:



- "**Wave 1**" caused the first instance's CPU to increase to ~90% triggering a High CPU alarm. Auto Scaling launched a second instance and both instance's CPU stabilizes to just under 50% with both instances.
- "**Wave 2**" caused both instances' CPU to increase to ~70% triggering another High CPU alarm and a 3rd instance is launched. All 3 instances stabilize their individual CPU utilization to just under 50% again.
- "**Wave 3**" causes the 4th instance to be launched and all 4 instances stabilize at ~55% CPU utilization. This requires a 5th instance to be launched because 4 instances were not able to bring the additional load under the 50% trigger point.