Auto Scaling

v. 3.0

TABLE OF CONTENTS

Auto Scaling Principles 4

The Four Key Components of Auto Scaling 4

Timing Matters 4

The Minimum Unit of Cost for EC2 is One Hour 5

Scaling Takes Time 5

Auto Scaling API Command-Line Tools 5

Start your *qwikLAB*™ 6

Connect to your EC2 Instance via SSH (Windows) 7

Download PuTTY 7

Download your EC2 Key Pair private key file 7

Connect to the EC2 Instance using SSH and PuTTY. 7

Connect to your EC2 Instance via SSH (OS X and Linux) 11

Download your EC2 Key Pair private key file 11

Connect to the EC2 Instance using the OpenSSH CLI client 11

Gather Lab Resource Details 12

Create a Launch Configuration 12

Create an Auto Scaling Group 13

Open the AWS Management Console 13

Select the EC2 Service 14

Confirm your AWS Region 14

Verify that the Servers Launched 15

Verify How Auto Scaling Works 16

Tagging Auto Scaling Resources 17

Auto Scaling Integration with ELB 18

Auto Scaling Notifications 20
Create SNS Topic 20
Create Auto Scaling Notifications 21
Create Auto Scaling Policies 22
Scale Up Policy 22
Scale Down Policy 22
Create CloudWatch High CPU Alert 23
Create CloudWatch Low CPU Alert 25
Test Auto Scaling 26
Wrapping Up 29
Viewing Auto Scaling Activities 29
Suspending/Resuming Auto Scaling Processes 29
How Large an Auto-Scaling Farm Can I Have? 29
CLI References 30
Appendix A: Multi-User Auto Scaling Benchmark Example 30
Appendix B: Using Proxy Servers 37
End Lab 37

Overview

Auto Scaling represents more than a way to add and subtract servers. It's also a mechanism to handle failures in a manner similar to the way that Load Balancing handles unresponsive servers. This lab will walk the user through configuring Auto Scaling to automatically launch, monitor, and update the Elastic Load Balancer (ELB) associated with your Elastic Compute Cloud (EC2) instances.

AUTO SCALING PRINCIPLES

First, Auto Scaling is a way to set the "cloud temperature". You use rules to "set the thermostat", and under the hood Auto Scaling controls the heat by adding and subtracting EC2 resources on an as-needed basis, in order to maintain the "temperature" (capacity).

Second, Auto Scaling assumes a set of homogeneous servers. That is, Auto Scaling does not know that Server A is a 64-bit XL and more capable than a 32-bit Small instance. In fact, this is a core tenet of Cloud Computing: scale horizontally using a fleet of fungible resources. An individual resource is not important, and accordingly the philosophy is "easy come, easy go".

THE FOUR KEY COMPONENTS OF AUTO SCALING

When you launch a server manually, you provide parameters such as which Amazon Machine Image (AMI), which Instance type, and which Security Group to launch in, etc. Auto Scaling calls this a **Launch Configuration**. It's simply a set of parameters.

Auto Scaling Groups tell the system what to do with an Instance once it launches. This is where you specify which Availability Zones it should launch in, which Elastic Load Balancer it will receive traffic from, and – most importantly this is where you specify the minimum and maximum number of Instances to run at any given time.

You need rules that tell the system when to add or subtract Instances. These are known as **Scaling Policies**, and have rules such as "scale the fleet up by 10%" and "scale down by 1 Instance".

These Scaling Policies can be triggered by you or on a schedule or, most powerfully, by CloudWatch Alarms. Alarms inspect CloudWatch metrics and change state when conditions like "average CPU across servers in the Auto Scaling group drops below 40% for 14 minutes" or "average CPU across Instances in the Auto Scaling group exceeds 65% for 10 minutes" are true or false. An Alarm can be in the Alarm or OK or Insufficient Data state. The last is a special state for when there is no data.

TIMING MATTERS

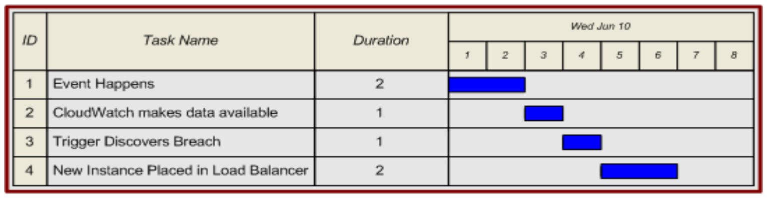
Let's focus on costs related to using Auto Scaling. There are two important concepts that directly affect the cost of AWS, and also the manner in which your application will scale:

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.

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.

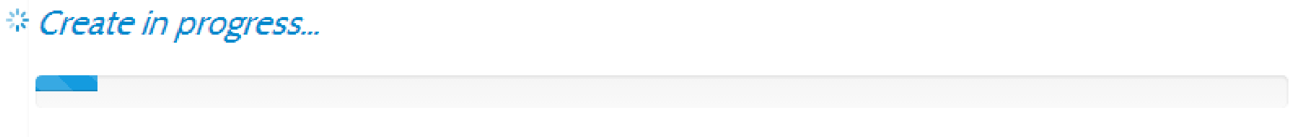
AUTO SCALING API COMMAND-LINE TOOLS

To control Auto Scaling you use the command line tools or API. For this lab we have pre-installed the CLI tools onto an instance, which you can connect to and setup your environment.

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.

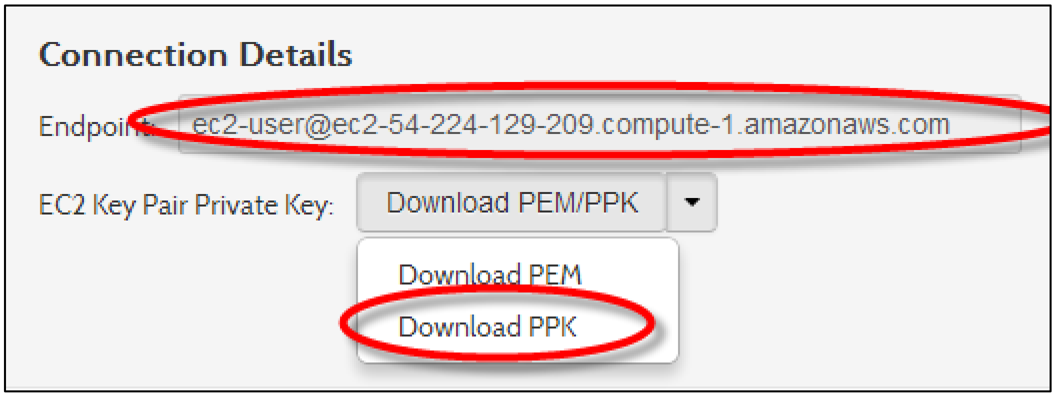
CONNECT TO YOUR EC2 INSTANCE VIA SSH (WINDOWS)

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 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.



Connection Details

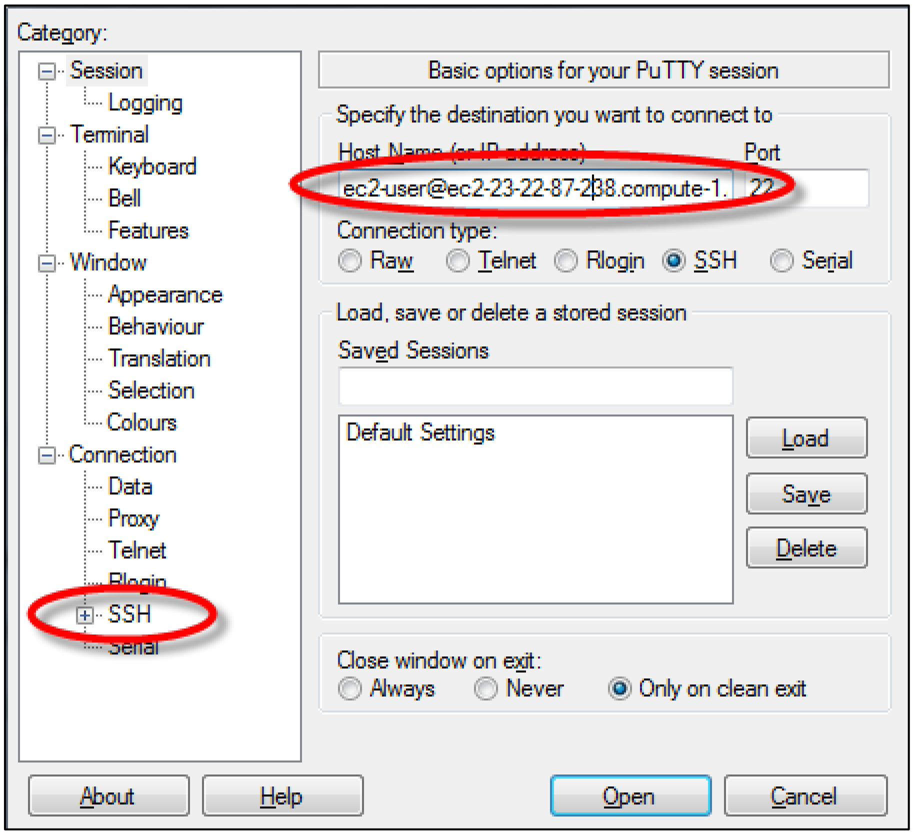
Endpoint: ec2-user@ec2-54-224-129-209.compute-1.amazonaws.com

EC2 Key Pair Private Key: Download PEM/PPK

4. Save the file to your Downloads directory (or some other directory of your choice.)
5. Copy the entire content of Endpoint to your clipboard (select and then Ctrl+c)

CONNECT TO THE EC2 INSTANCE USING SSH AND PUTTY.

1. Open the putty.exe you downloaded or already had.
2. Paste the Endpoint string you copied to your clipboard into the Host Name input in Putty (Ctrl+v).
3. Expand the SSH category by clicking on it.


Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-user@ec2-23-22-87-238.compute-1.

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

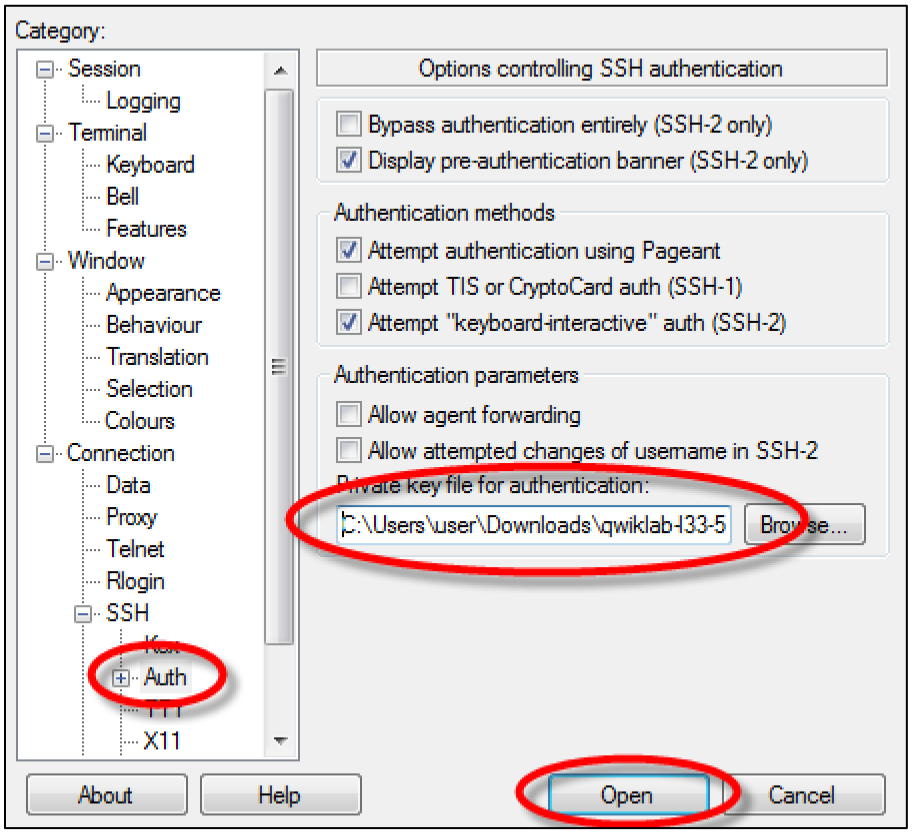
Load, save or delete a stored session

Saved Sessions

Default Settings

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

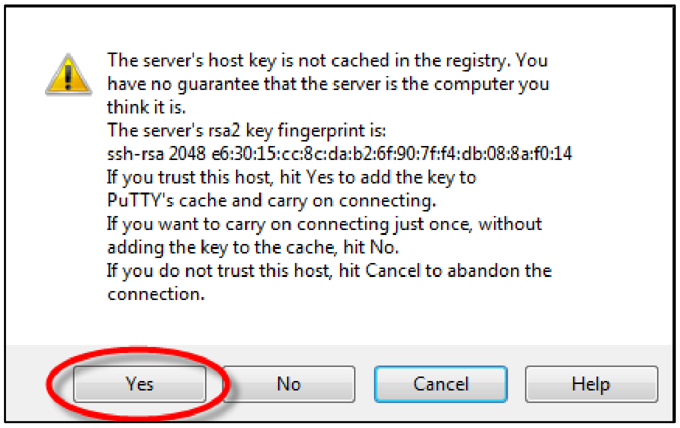
- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

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

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

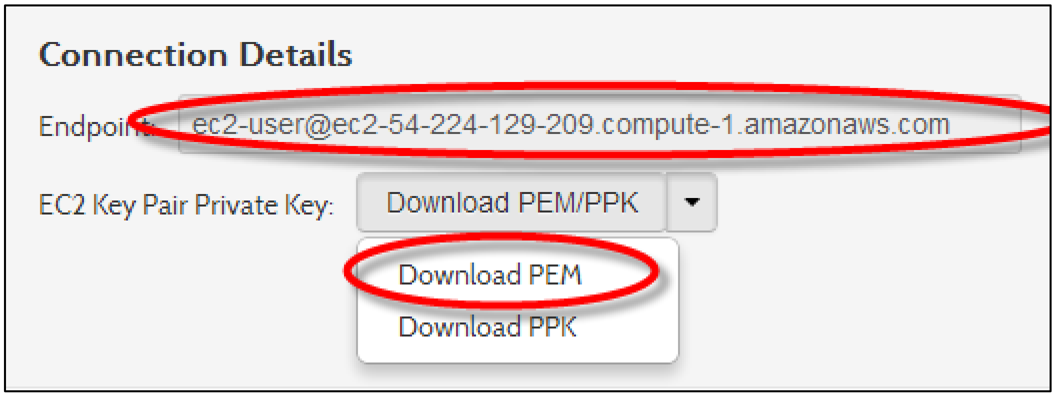


The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 e6:30:15:cc:8c:da:b2:6f:90:7f:f4:db:08:8a:f0:14
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
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.

CONNECT TO YOUR EC2 INSTANCE VIA SSH (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.


Connection Details
Endpoint: ec2-user@ec2-54-224-129-209.compute-1.amazonaws.com
EC2 Key Pair Private Key:
Download PEM/PPK
Download PEM
Download PPK

3. Save the file to your Downloads directory (or some other directory of your choice.)
4. Copy the entire content of Endpoint to your clipboard (select and then Ctrl+c)

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 the username@hostname you copied from Endpoint in *qiwk*LAB.

```
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
```

GATHER LAB RESOURCE DETAILS

*qwik*LAB™ created for you a CloudFormation stack containing an Elastic Load Balancer, EC2 Security Group, and EC2 Instance. *qwik*LAB™ also created an EC2 Key Pair for you. You will be using those resources in this lab and referencing them by their names. Those names are in a file '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 ~/lab-details.txt
ElasticLoadBalancer,stack-136-ElasticL-49ICDQ2I88AE
Ec2SecurityGroup,stack-1366573233-Ec2SecurityGroup-1S7OZJ8S6W73Y
Ec2Instance,i-21f80f14
AMIId,ami-ecbe2adc
KeyName,qwiklab-l33-5023
AvailabilityZone,us-west-2b
```

Note, values in that you see in red are values that will be different for you when you run this lab. Where red is used later it may denote where you may need to replace values with your own when you are constructing and running commands in this lab.

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

This launch configuration specifies a machine image (Amazon Machine Image (or 'AMI')) that will launch when Auto Scaling adds new servers. This image should be the value for 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.
• Launch-config	The name of this Auto Scaling Launch Configuration. We used **lab-lc**.

CREATE AN AUTO SCALING GROUP

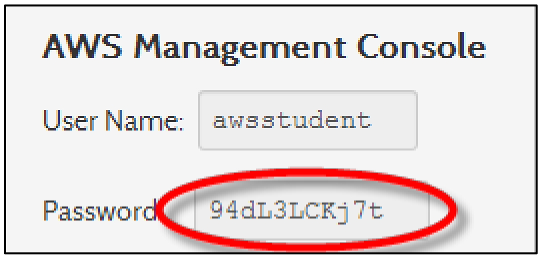
We are going to launch our EC2 Instances into a single AZ in the AWS region you are running this lab in. Copy-paste the below into your plain-text editor and edit the command to use values you already have from previous commands in this lab and lab-details.txt.

```
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
```

OPEN THE AWS MANAGEMENT CONSOLE

1) 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



User Name: awsstudent
Password: 94dL3LCKj7t

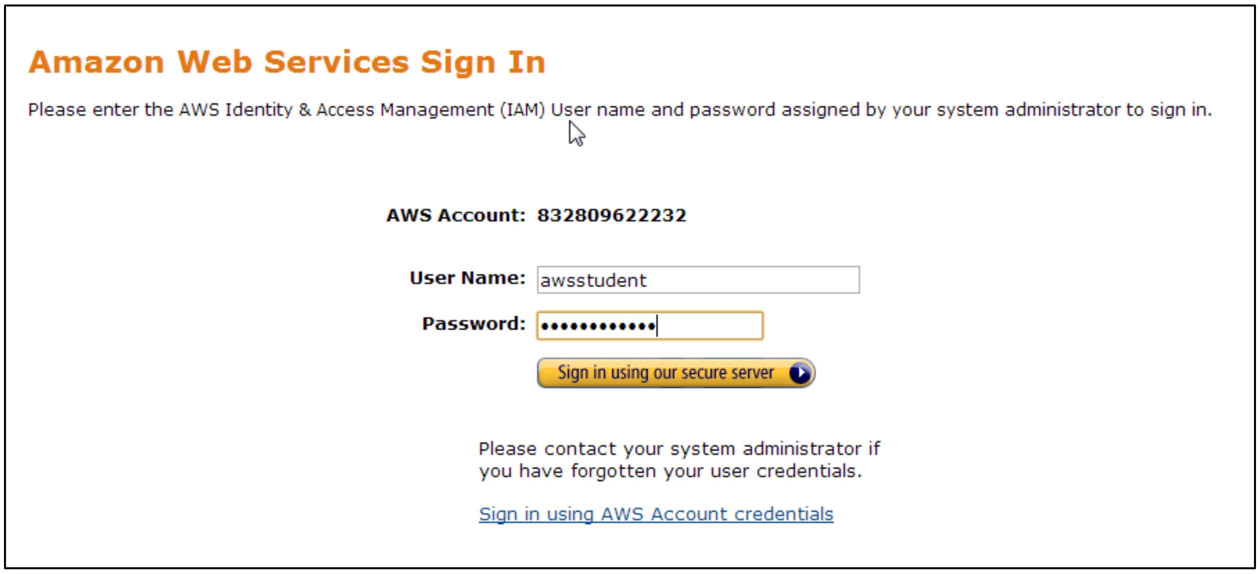
2) Click the 'Open Console' button.

3) 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

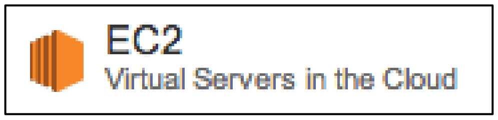
Password: ••••••••••••

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

Sign in using AWS Account credentials

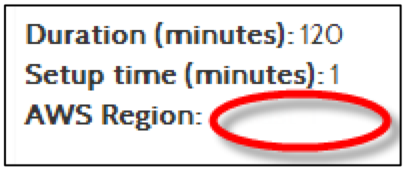
SELECT THE EC2 SERVICE

Select "EC2" from the Console Home.



CONFIRM YOUR AWS REGION

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

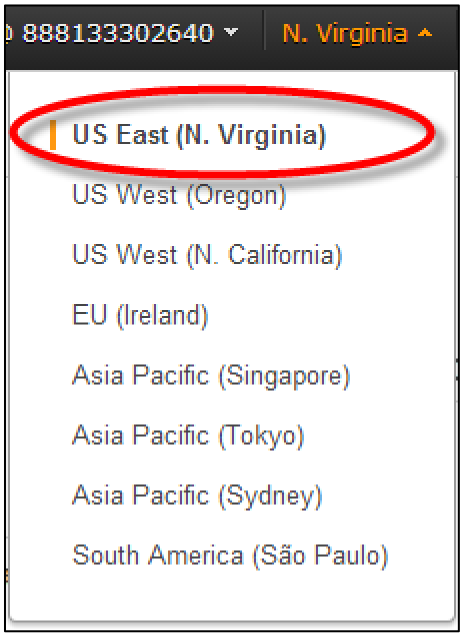


Duration (minutes): 120

Setup time (minutes): 1

AWS Region:

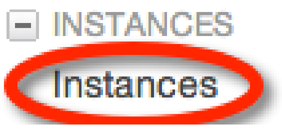
Select or confirm that the same AWS Region is already set in the AWS Management Console



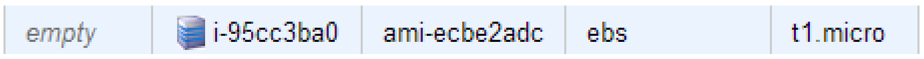
VERIFY THAT THE SERVERS LAUNCHED

Use the AWS Management Console to inspect your Instance count. You should see two Instances in your fleet because you set the minimize size to 1 (you may need wait a few minutes before it appears) and you had one Instance already running.

Click on Instances

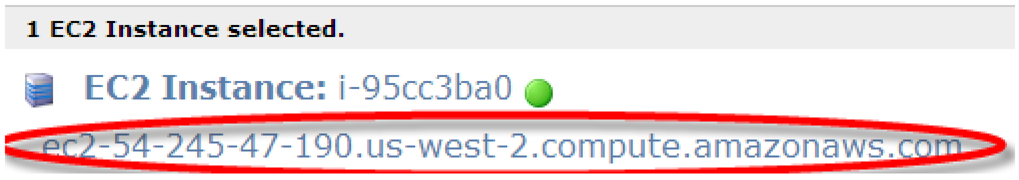


Select the t1.micro type instance.



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

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

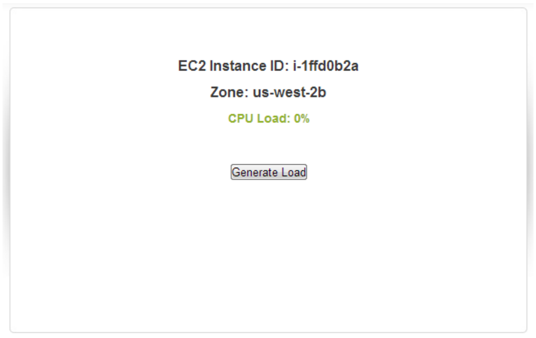


1 EC2 Instance selected.

EC2 Instance: i-95cc3ba0

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

And paste it into your browser to verify that it is running properly. You should see the application bootstrapped via the launch configuration display information as below:

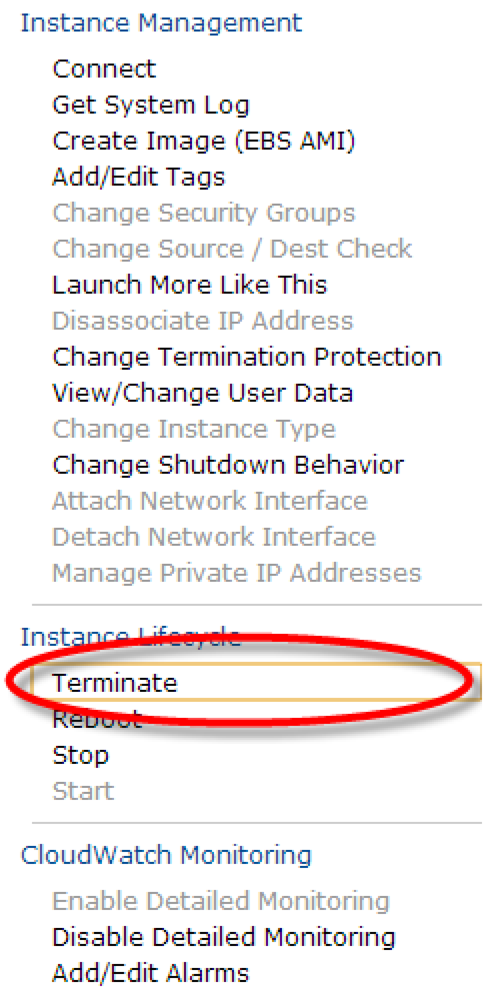

EC2 Instance ID: i-1ffd0b2a

Zone: us-west-2b

CPU Load: 0%

VERIFY HOW AUTO SCALING WORKS

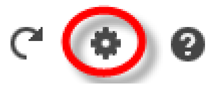
Return to the AWS Management Console and terminate the Instance. In a few minutes it will re-appear, because Auto Scaling will notice that the fleet size is below minimum size.

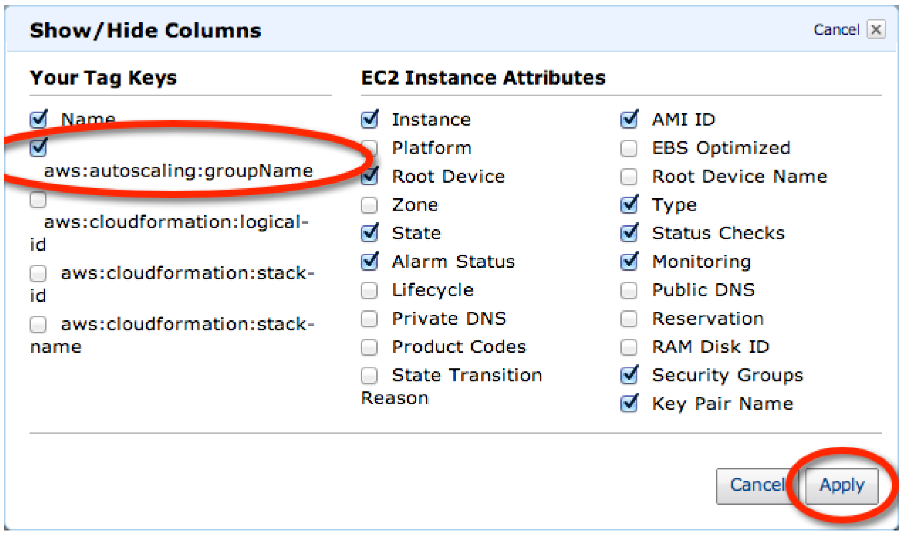


Try doing the same thing by shutting down the instance (rather than outright terminating it) either by logging into the new instance and issuing a shutdown command (sudo shutdown -h now) or by right-clicking on the instance and selecting **Stop** from the AWS Management Console. Notice that Auto Scaling 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. Click on the Show/Hide button, then select the **aws:autoscaling:groupName** option under **Your Tag Keys**. Click Apply and then click on the Refresh button.





Show/Hide Columns

Cancel

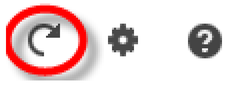
Your Tag Keys

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

EC2 Instance Attributes

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

Cancel Apply

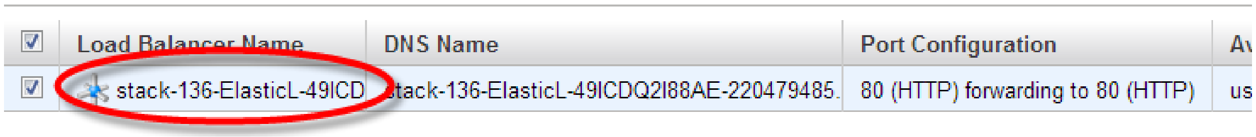


Auto Scaling automatically creates and populates a tag called aws:autoscaling:groupName for your Auto Scaling instances. The second way you can better identify your Auto Scaling instances is to modify your Auto Scaling group to populate the **Name** tag for you. We could have created a Name tag for the Auto Scaling group on creation by using the --tag "k=Name, v=AS Web Server" option. Since our Auto Scaling group already exists, let's just modify the existing tags with the following command:

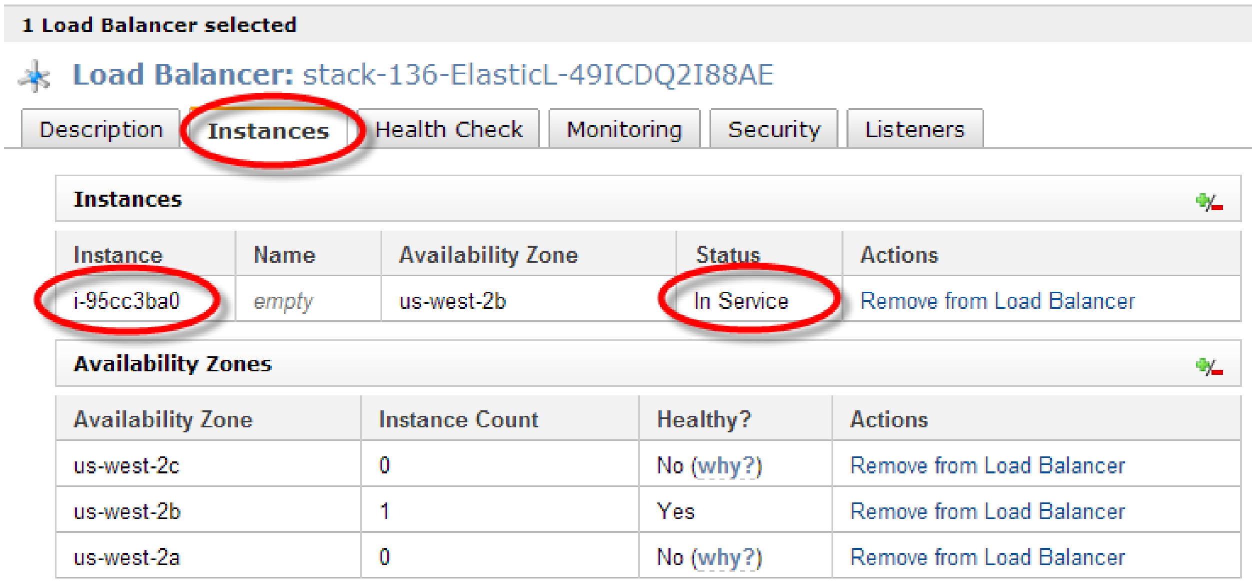
```
as-create-or-update-tags --tag "id=lab-as-group, t=auto-scaling-group, k=Name, v=AS-Web-Server, p=true"


OK-Created/Updated tags
```

AUTO SCALING INTEGRATION WITH ELB

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

AUTO SCALING NOTIFICATIONS

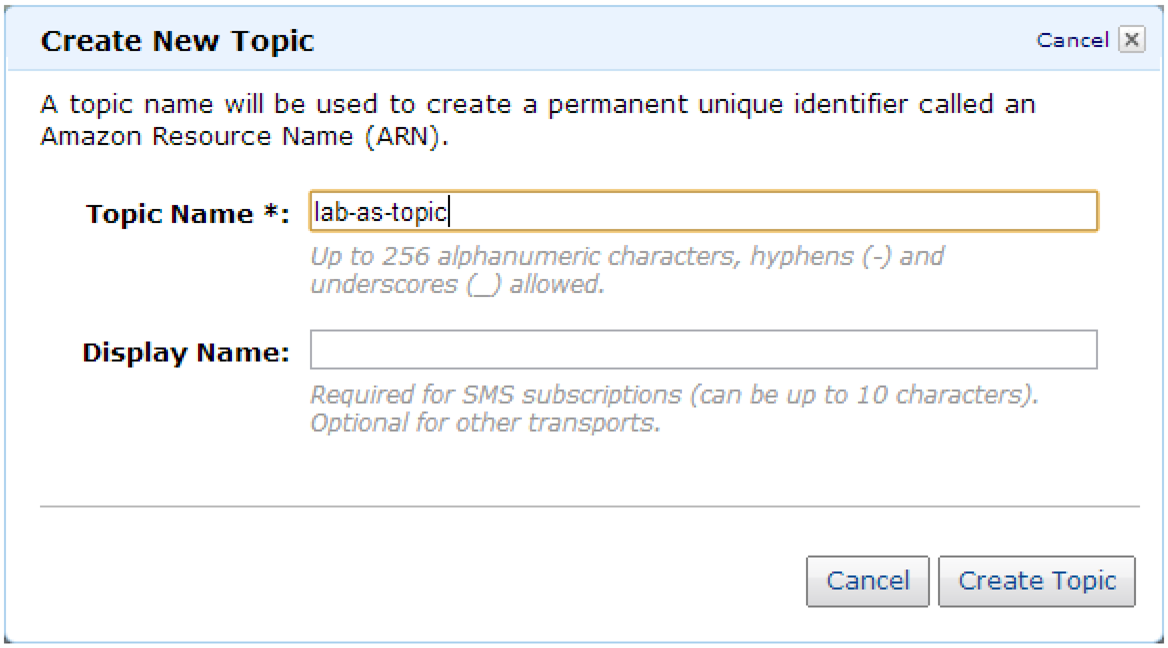
All of these Auto Scaling activities are occurring transparently. Wouldn't it be nice to have Auto Scaling notify you when it automatically creates or terminates instances on your behalf? Auto Scaling has been integrated with Amazon Simple Notification Service (SNS) for precisely this purpose. SNS is 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.

CREATE SNS TOPIC

The first thing we need to do is create an SNS topic that we will use to send SNS notifications. In the **AWS Management Console**, click on the **SNS** tab, and click on the **Create New Topic** button.

SNS
Push Notification Service

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


Create New Topic
Cancel

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

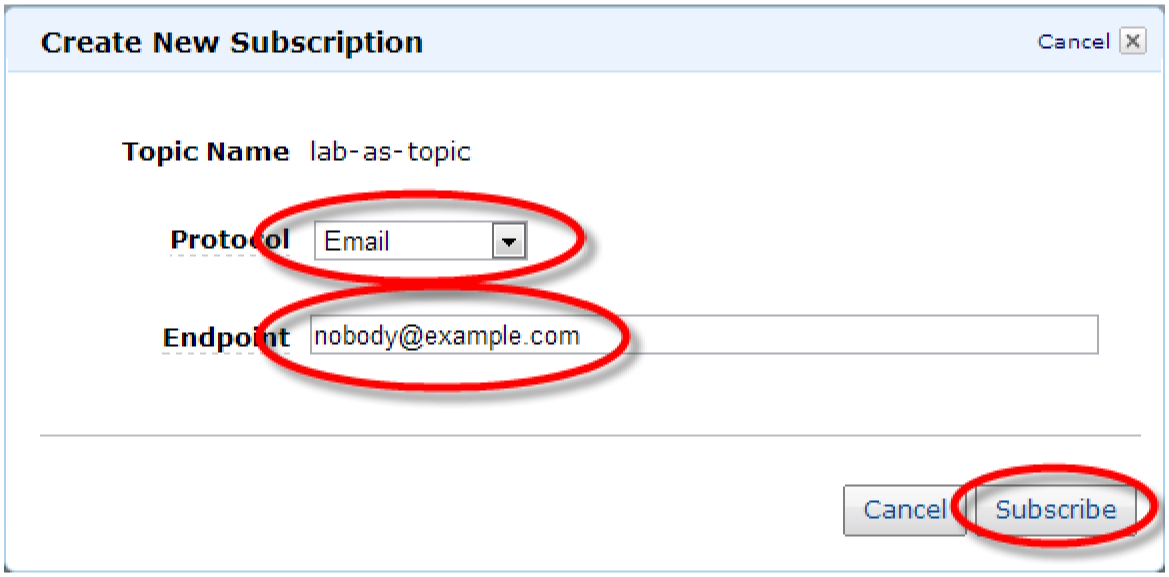
Topic Name *: lab-as-topic

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.

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

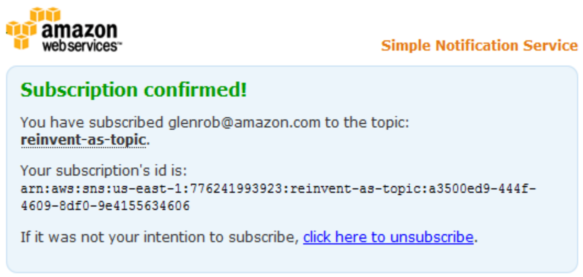


Create New Subscription

Topic Name lab-as-topic

Protocol: Email

Endpoint: nobody@example.com



Simple Notification Service

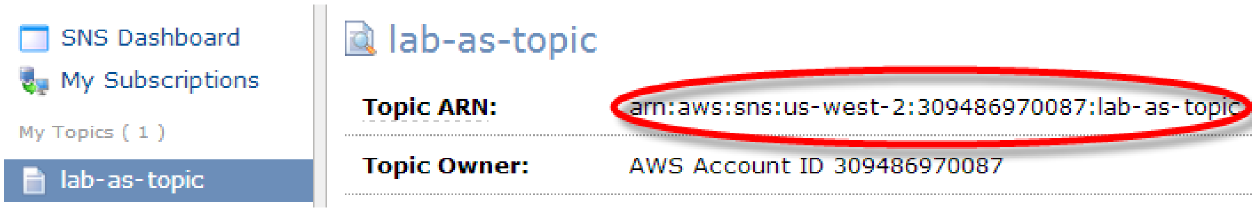
Subscription confirmed!

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 Number (ARN) for the SNS topic to use with Auto Scaling. Select the SNS topic, locate the Topic ARN, and copy the ARN into Notepad for use in the next step.



lab-as-topic

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
```

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 **as-update-auto-scaling-group lab-as-group --min-size #** command. But we are going to go another step farther. 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 fleet is >= 50%.

We will use the **as-put-scaling-policy** command to create two Auto Scaling scale up policies that will increase the number of servers by 1 for scale up events and will 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

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

SCALE DOWN POLICY (notice the policy name change as well as the slight change enclosing the --adjustment=-1 option in quotes, and setting a negative 1 value to decrease the number of instances by 1)

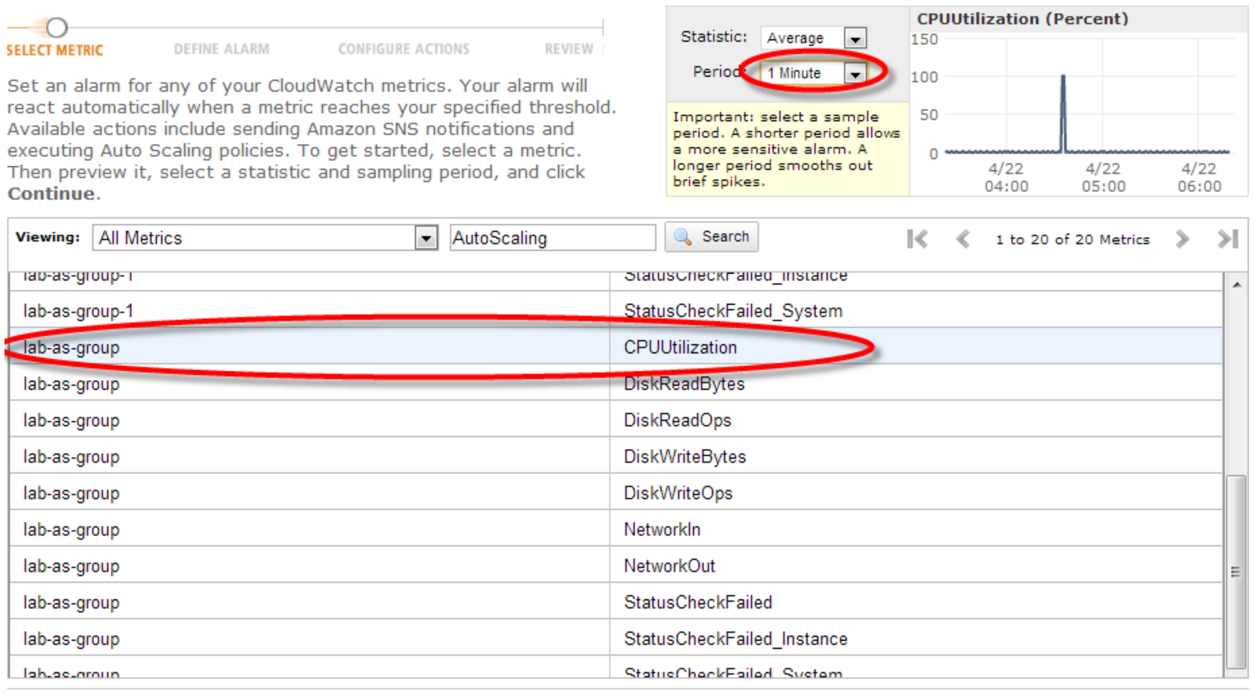
```
as-put-scaling-policy lab-scale-down-policy --auto-scaling-group lab-as-group  "--
adjustment=-1" --type ChangeInCapacity  --cooldown 300
```

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

CloudWatch
Resource and Application Monitoring

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**.

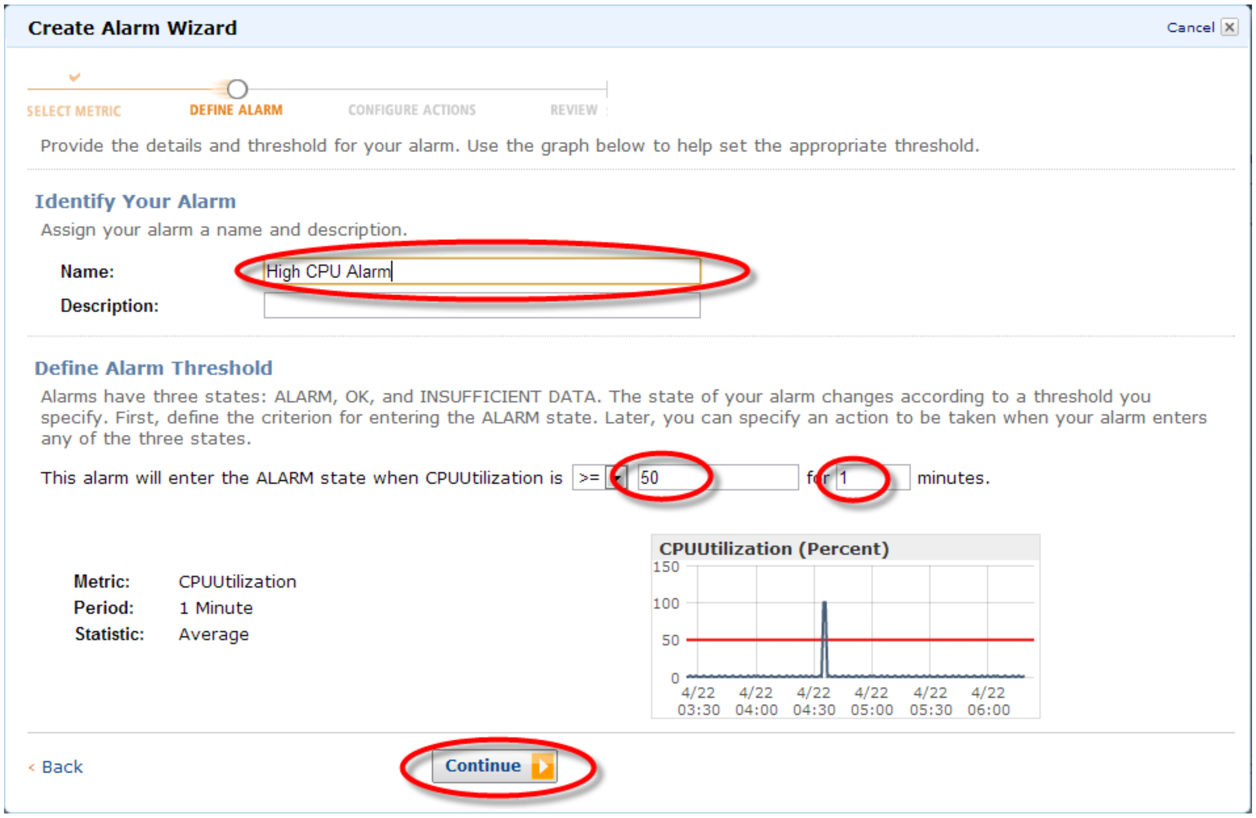
Statistic: Average
Period: 1 Minute

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

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



Create Alarm Wizard

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.

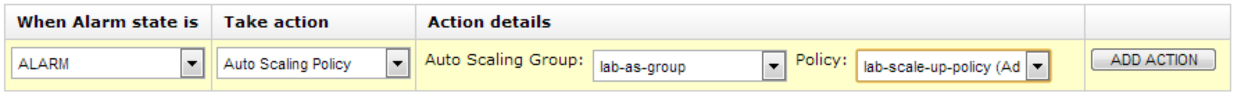
This alarm will enter the ALARM state when CPUUtilization is >= 50 for 1 minutes.

Metric: CPUUtilization

Period: 1 Minute

Statistic: Average

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



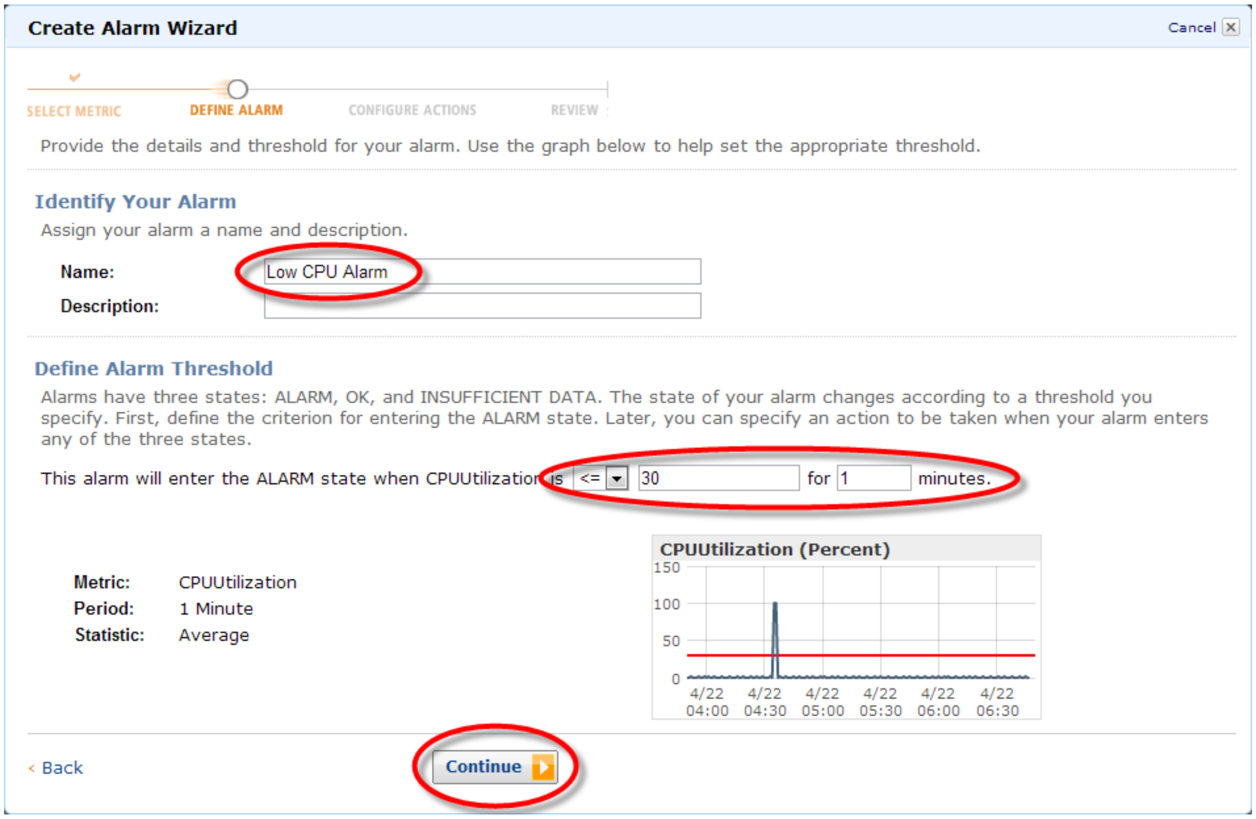
When Alarm state is	Take action	Action details	
ALARM	Auto Scaling Policy	Auto Scaling Group: lab-as-group Policy: lab-scale-up-policy (Ad	ADD ACTION

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

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 (3 minutes) rather than 5 minute granularity.

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:

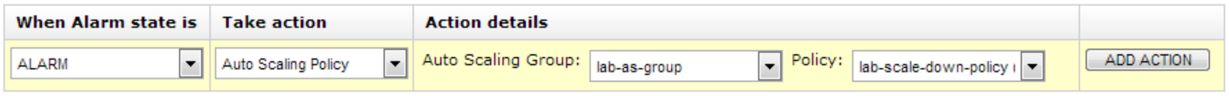
Define Alarm Threshold

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

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

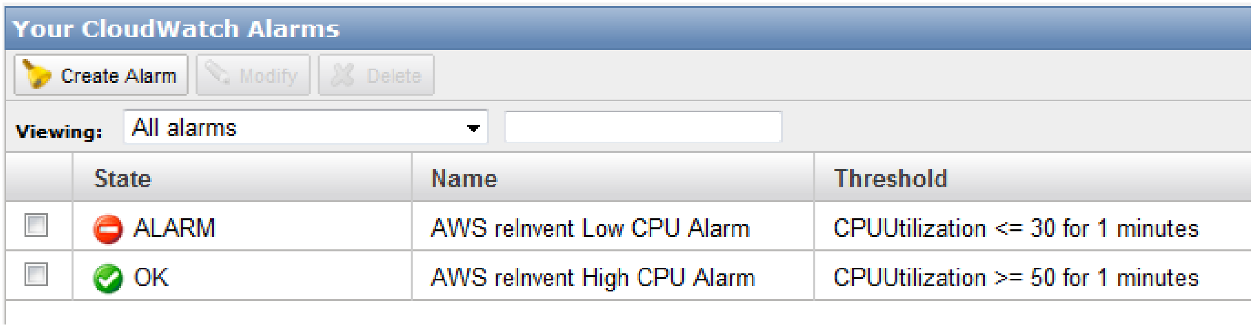


When Alarm state is	Take action	Action details	
ALARM	Auto Scaling Policy	Auto Scaling Group: lab-as-group Policy: lab-scale-down-policy	ADD ACTION

Note: We strongly 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. This lab's example is intentionally short and simplistic. From a billing perspective, it costs no more if the instance is scaled down after 3 minutes than if it is allowed to run for 59 minutes.

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

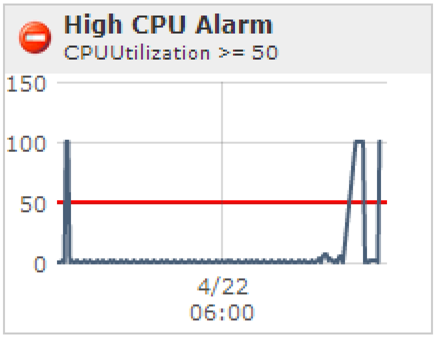
This is because the current group CPU Utilization is < 30%. However Auto Scaling is not removing any instances because the group size is currently at its minimum (1). Also remember that we set the cool down time for our auto scaling polices to 5 minutes (300 seconds), this is important to remember because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

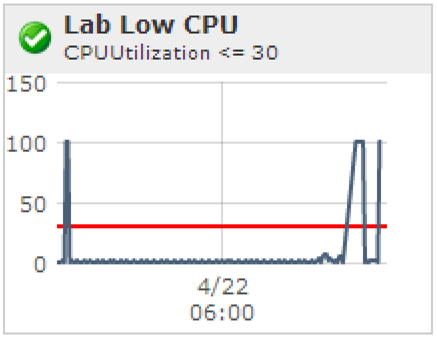
Return to the EC2 service page and select **Load Balancers** in the navigation pane and copy the public DNS name of your ELB into the clipboard.

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.

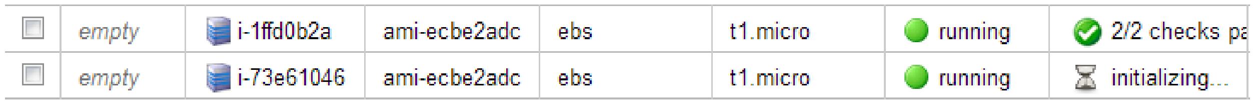
Let's head back to the CloudWatch tab and in 3-4 minutes you should see the alarms switch to the Low CPU state changing to **OK** and the High CPU alarm state changing to **Alarm**.

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



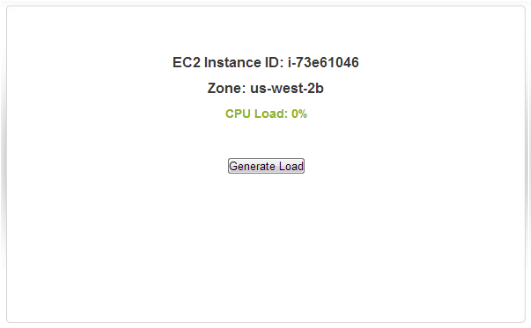


Head back over to the **EC2** tab -> **Instances** and you will see a new instance has been added to your group:



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

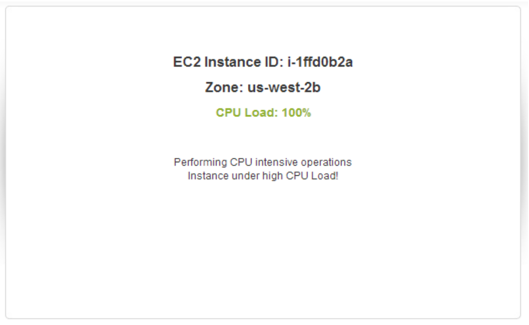
Go back to your ELB browser tab and refresh the ELB several times to see one server under heavy load, while the other is not:



EC2 Instance ID: i-73e61046

Zone: us-west-2b

CPU Load: 0%



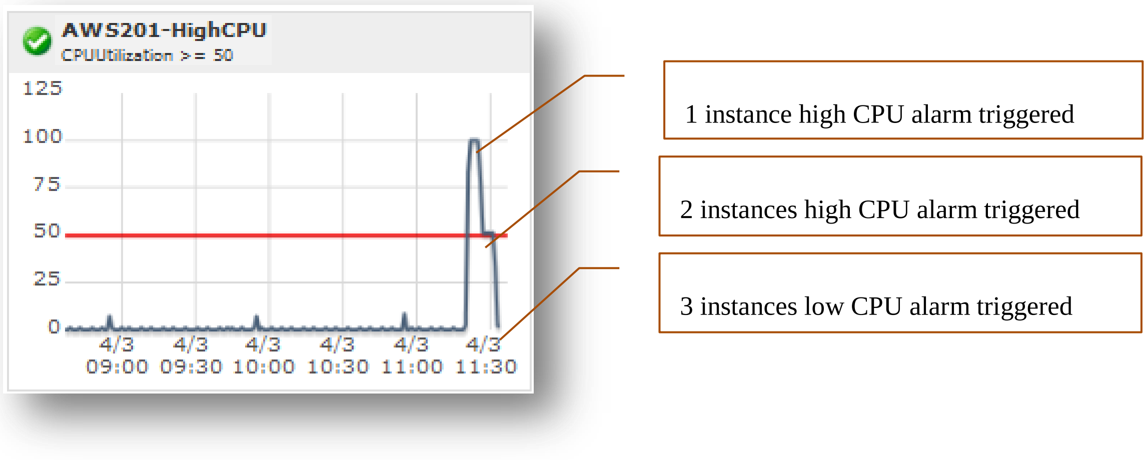
EC2 Instance ID: i-1ffd0b2a

Zone: us-west-2b

CPU Load: 100%

Performing CPU intensive operations
Instance under high CPU Load!

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.



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. This allows you to roll in new changes to your application by updating your launch config to a newer AMI, then triggering Auto Scaling events (e.g. increase min size).

Note: You have probably noticed that this is not an overly realistic test because it essentially simulates a user slamming a single server and does not take advantage of your load balancer. In this case, Auto Scaling will help additional customers, but does not load balance this “work” across multiple servers. We use this example for simplicity and to make more effective use of your lab time. *Appendix* A: Multi-User Auto Scaling Benchmark Example.

WRAPPING UP

VIEWING AUTO SCALING ACTIVITIES

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
```

SUSPENDING/RESUMING AUTO SCALING PROCESSES

Auto Scaling also allows you to intentionally tell an Auto Scaling group to suspend or resume auto scaling processes using the **as-suspend-processes** and **as-resume processes** commands. This can be helpful if you know ahead of time that certain activities (e.g. maintenance events) will trigger auto scaling alerts but you do not want instances to be automatically added or removed during the course of the event.

HOW LARGE AN AUTO-SCALING FARM CAN I HAVE?

By default each account is assigned limits on a per-region basis. Initially accounts are set to a maximum of 20 EC2 instances, 100 spot instances, and 5000 EBS volumes or an aggregate size of 20 TB (whichever is smaller).

Does the max-instance limit only count towards running instances? In other words, do Stopped Instances (not terminated) also count towards this limit?

The max-instances limit only applies for instances that are in the pending, running, shutting-down and stopping states. There is another limit, which is set to 4x the max-instances limit that governs the total number of instances in any state. So, for a default account with no limit overrides, you can have 20 "running" instances and up to 60 stopped instances; bringing this to a total of 80 instances.

All of these defaults can be changed by making a request online at http://aws.amazon.com/contact-us/.

CLI REFERENCES

Please visit the Auto Scaling documentation page for additional information about Auto Scaling.

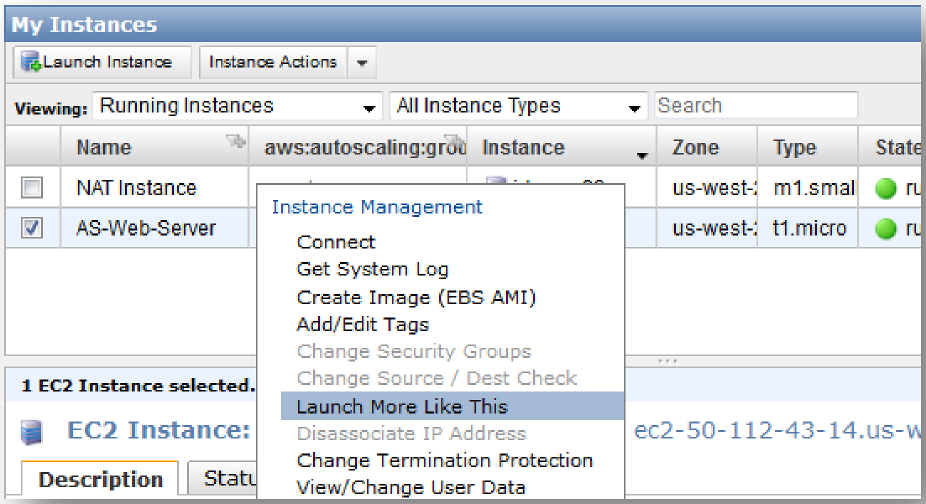
http://aws.amazon.com/documentation/autoscaling/

Also, check out the command line quick reference card for additional command line commands and options.

http://awsdocs.s3.amazonaws.com/AutoScaling/latest/as-qrc.pdf

APPENDIX A: 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**.



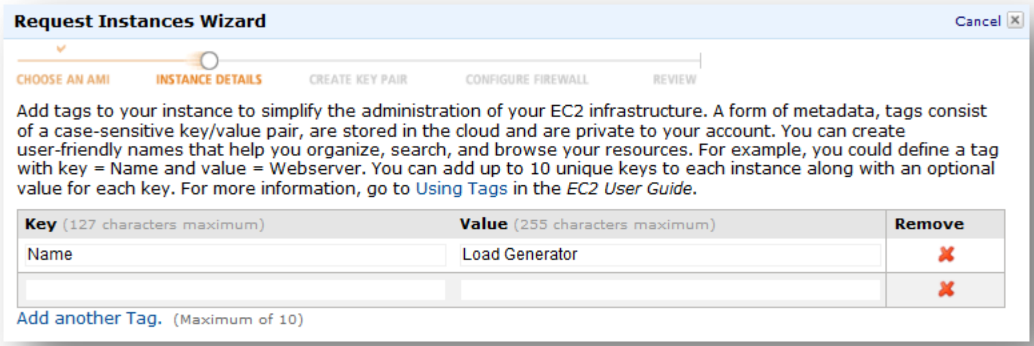
Viewing: Running Instances | All Instance Types | Search

	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

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



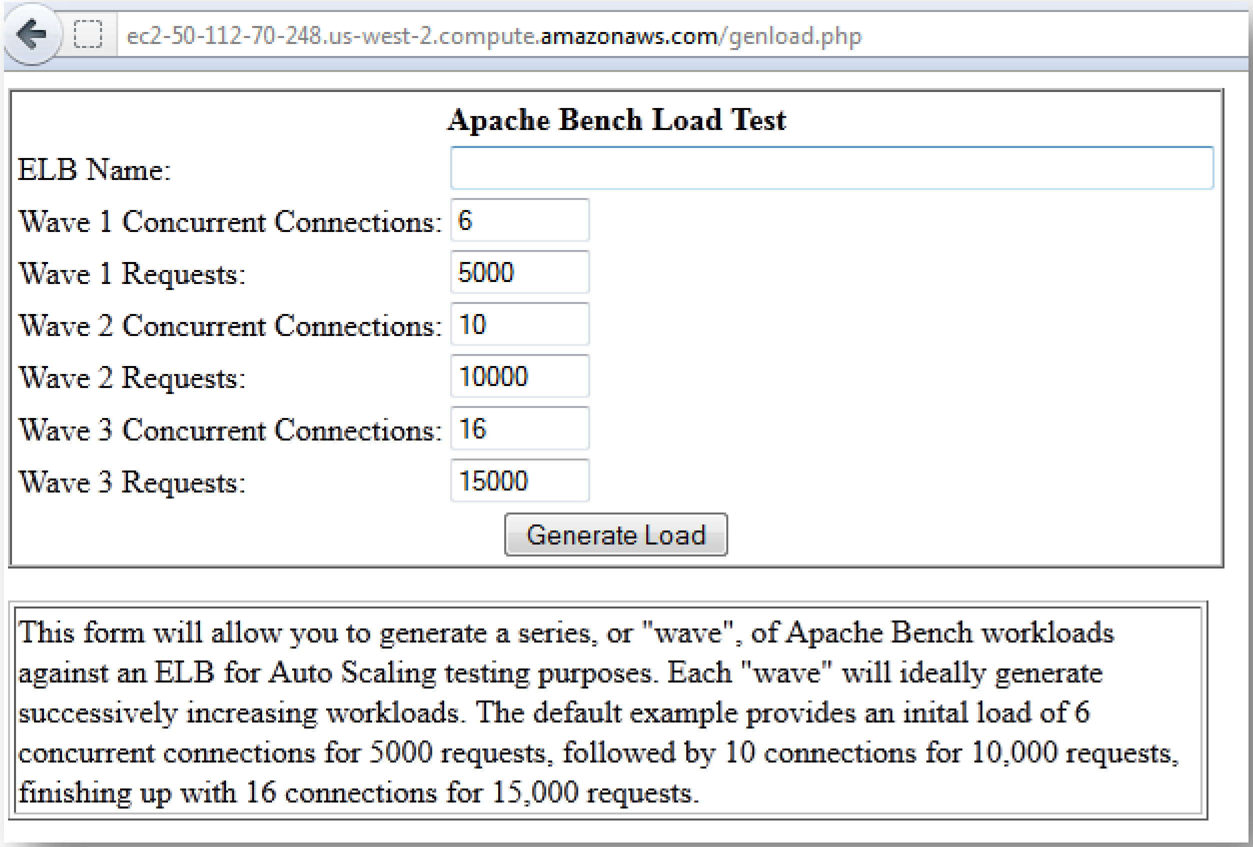
Request Instances Wizard

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

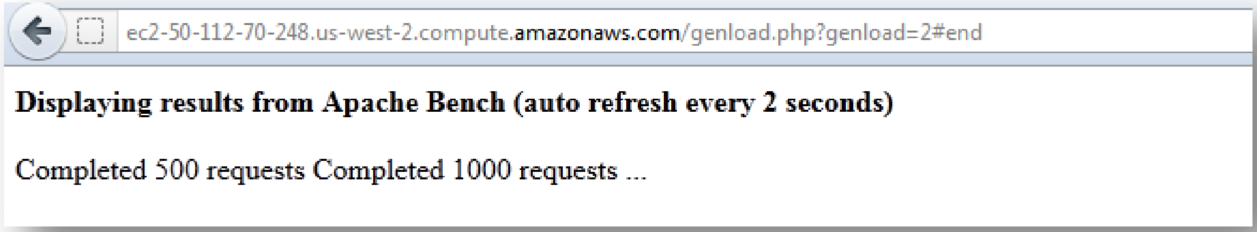
Apache Bench Load Test

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:

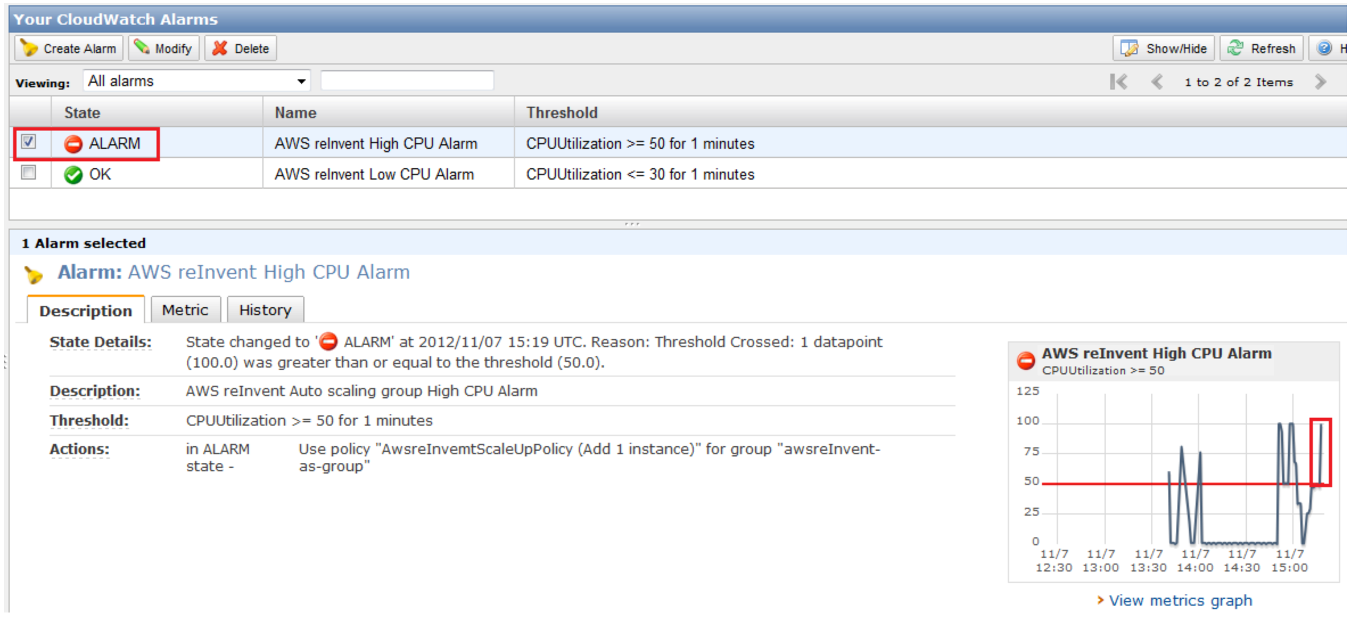


ec2-50-112-70-248.us-west-2.compute.amazonaws.com/genload.php?genload=2#end

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:



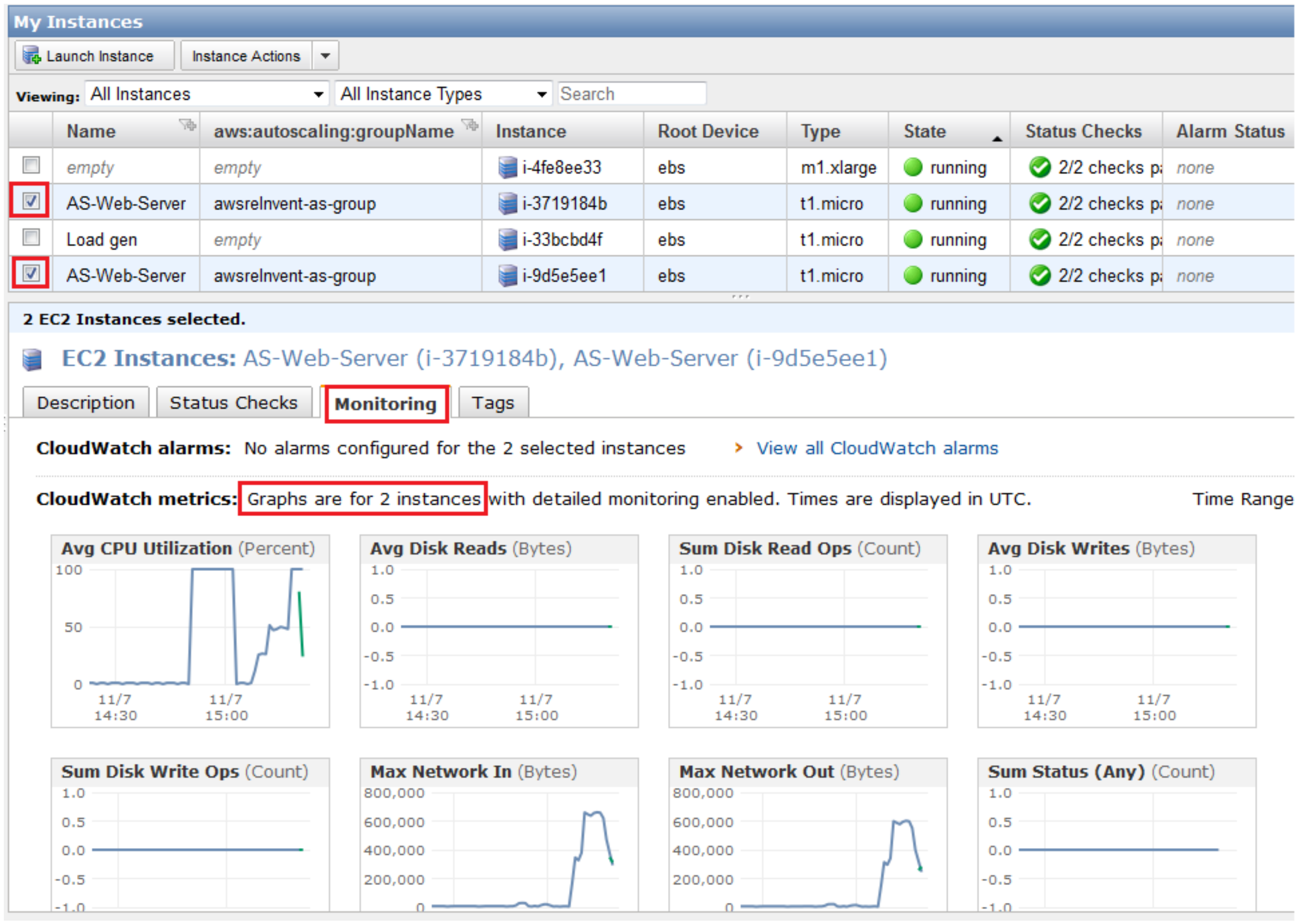
State	Name	Threshold
ALARM	AWS reInvent High CPU Alarm	CPUUtilization >= 50 for 1 minutes
OK	AWS reInvent Low CPU Alarm	CPUUtilization <= 30 for 1 minutes

1 Alarm selected

Alarm: AWS reInvent High CPU Alarm

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.



	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	awsrelnvent-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	awsrelnvent-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

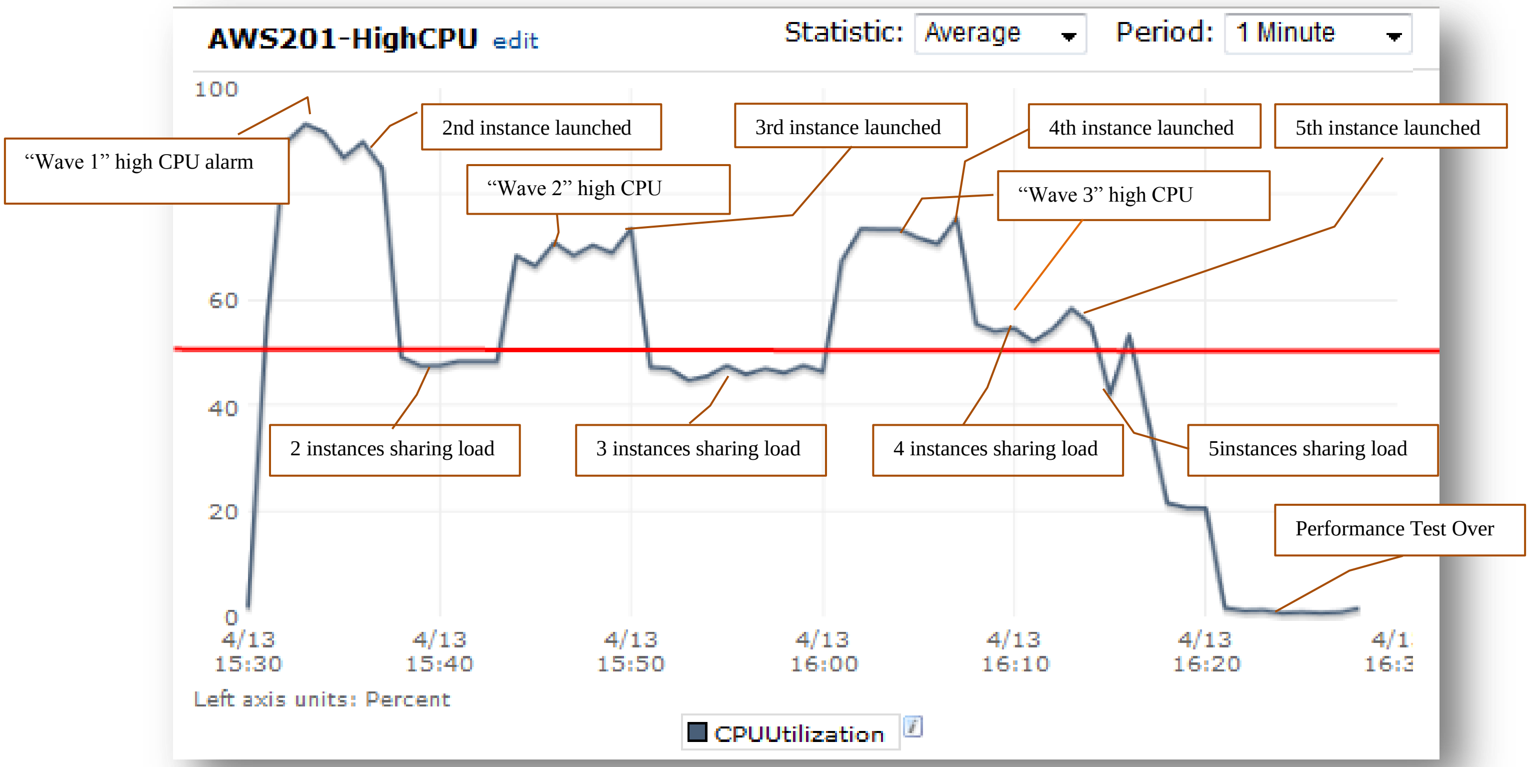
CloudWatch metrics: Graphs are for 2 instances with detailed monitoring enabled. Times are displayed in UTC.

EXPLANATION OF GRAPHS

CLOUDWATCH HIGH CPU ALARM GRAPH

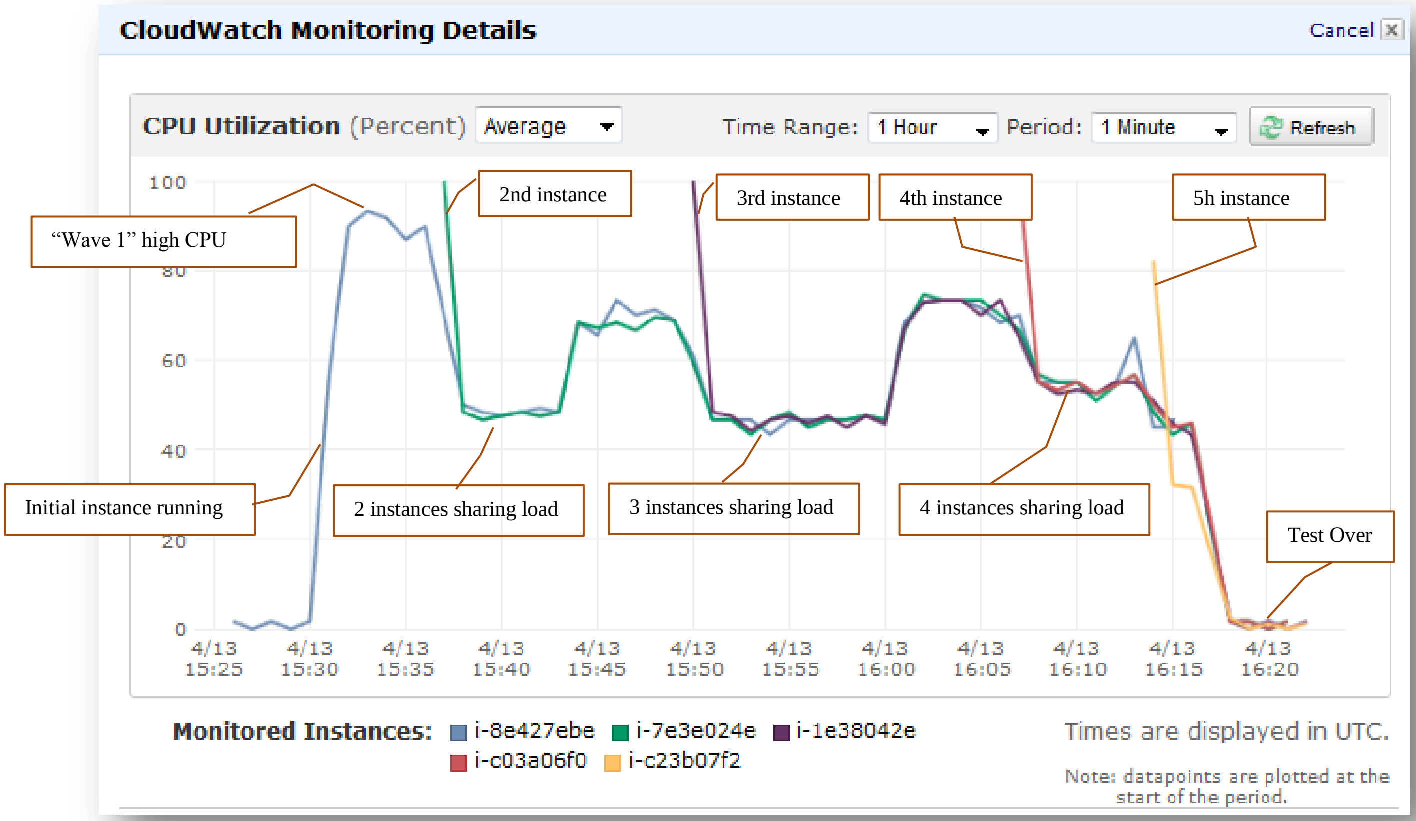
This graph covers the aggregate CPU utilization for the Auto Scaling group as a whole.

- "**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.

APPENDIX B: USING PROXY SERVERS

If you are behind a proxy server, you can use the EC2_JVM_ARGS environment variable to configure proxy settings.

Windows Parameter:

```
Set SERVICE_JVM_ARGS=-DproxySet=true -Dhttps.proxyHost=<HttpsProxyhostname> -
Dhttps.proxyPort=<HttpsProxyPort> -Dhttp.proxyHost=<proxyhostname> -
Dhttp.proxyPort=<HttpProxyPort>
```

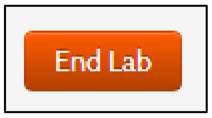
Mac Parameter:

```
export SERVICE_JVM_ARGS=”-DproxySet=true -Dhttps.proxyHost=<HttpsProxyhostname> -
Dhttps.proxyPort=<HttpsProxyPort> -Dhttp.proxyHost=<proxyhostname> -
Dhttp.proxyPort=<HttpProxyPort>”
```

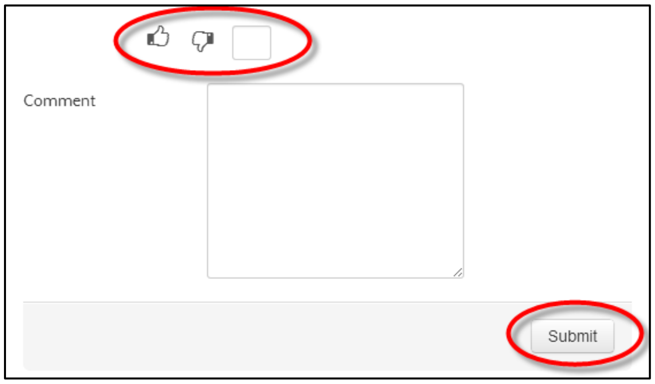
END LAB

Sign-out of the AWS Management Console.

Click the End Lab button in *qwikLAB*™.



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


Comment
Submit