Student Exercizes

Prepared by:

Amir Vahid

Mohsen Amini Salehi

Exercize description

The exercizes in this course built upon a common case study:

> RUBiS is an online web store prototype modeled after eBay.com and is used to evaluate application design patterns and application server's performance scalability. It is designed a client that emulates users behavior for various workload patterns and provides statistics.
>
> The auction site defines 26 interactions that can be performed from the client's Web browser. Among the most important ones are browsing items by category or region, bidding, buying or selling items, leaving comments on other users. Browsing items also includes consulting the bid history and the seller's information. In addition, RUBiS uses a MySQL database that contains 7 tables: users, items, categories, regions, bids, buy_now, and comments.

Over the exercizes we use different entities which their specs

This course includes the following exercizes:

1. Command Line tools
2. JAVA and Python Programming Libraries
3. Creating Amazon EC2 instance
4. Exercize setup installs the RUBiS on the created Amazon instance.
5. Creating and listing S3 buckets to prepare S3 to store huge number of pictures used by the e-commerce application.
6. Uploading files to S3 to be incorporated into the e-commerce web site.
7. Minimize the response time of loading pictures in the web site via utilizing CloudFront.
8. Applying best practices for hosting your application
9. Configuring your Amazon Machine Image (modifying existing images, creating your own Image)
10. IP address reservations
11. Choosing a proper instance type and region to minimize the cost(data transfer, instance usage, and monitoring cost) and maximize the performance
12. Creating a DMZ using Security Groups and Configuring the Firewall
13. Creating Scalability Groups to add auto scalability to your application
14. Amazon RDS LAB
15. Design your Java web application for the for Beanstalk
16. Use Beanstalk to set up your development, testing, production, and staging environments.
17. SQS Fundamental concepts and operation
18. SQS Case study
19. Virtual Private Cloud (VPC)

Lab Session 1. S3:

Exercize 1. Buckets and Objects

What is this exercise about

This exercise shows students how to create a bucket in S3 and also how to work with that bucket.

Background

- S3 is a paid storage service run by Amazon.
- Buckets are similar to directories (folders) in S3.
- Files stored in S3 are called "Objects" and the file names called "Keys".
- All objects (files and directories) are stored within buckets.

What should be able to do

At the end of this exercise, you should be able to:

- Create buckets on S3 through console, command line, and Java API.
- Upload files (objects) in to the S3 bucket and organizing them.
- Being able to carry out basic operations on S3 through different access methods.

Create a bucket on S3

Using AWS Management Console:

1. Go to Amazon web console and click on S3 panel.
2. Click on the "Create Bucket button".

3. Key in the name ("*s3Training*") and the region for new bucket and press *create*.
4. Keep the region as its default (US Standard)
5. To set up server access logging for the bucket you can click on*Set Up Logging>*

Exercize 2. Uploading files to S3

What should be able to do

Uploading files care called Creating Objects in a bucket.

In this exercise we upload pictures that are needed in our web site to S3.

Uploading files to S3

To carry out this exercize, first download needed files from the following address:

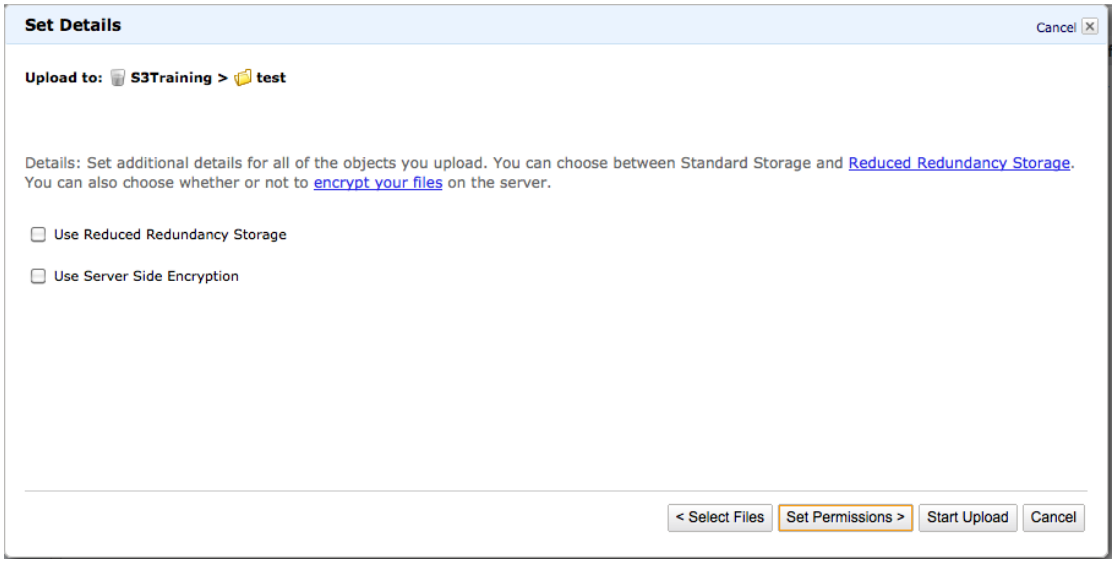
https://s3.amazonaws.com/S3Training/Archive.zip

Unzip the downloaded file, unzip it and store it in a folder (preferably name it "pictures"). Then, follow below steps:

1. Click on "Create Folder"
2. Key in the folder name as "pics".
3. Enter the pics folder by double-clicking on that.
4. Within the pics folder, click on "Upload".

5. Click on "Add file"
6. Select all the files from the picture directory onyour computer.
7. Press "Start Upload".

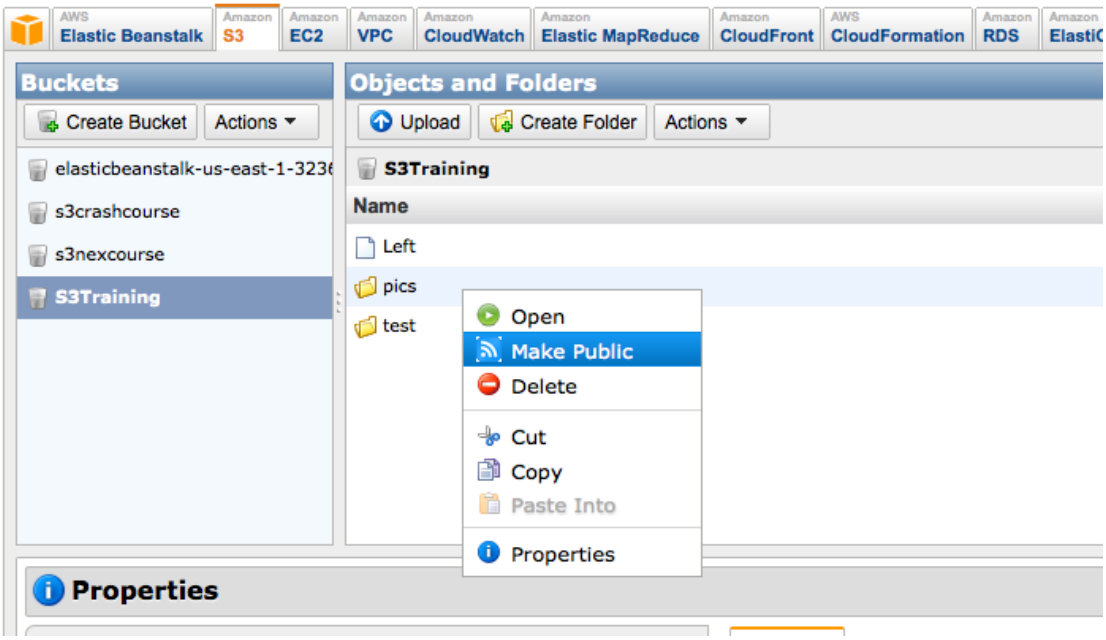
Note: When you want to upload a file, you can click on "Set Details" to set further details about that file.



- "Use Reduced Redundancy Storage" is suitable for less important objects and helps in reducing costs.
- "Use Server Side Encryption" is suitable for securing objects on the S3.

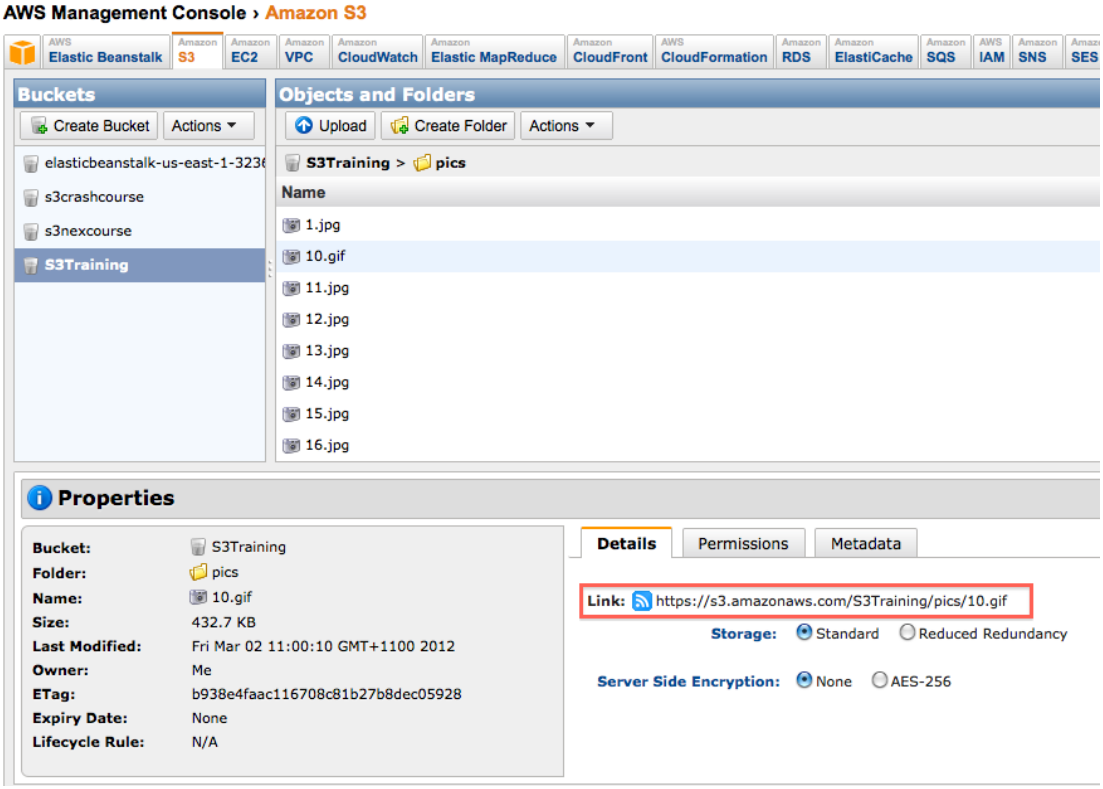
After uploading the pictures to the S3, you have to make them visible globally to be able to use in our web site. For that purpose, follow these steps:

1. Go back to the upper layer (pics folder).
2. Right-click and press "Make Public".



3. Press "OK" in the appeared message box.
4. Go to the folder and click on one file.
5. URL of that particular picture appear in the underneath pane (properties)
6. Copy the URL address of the index.html file and paste it in a browser!

Note: If you declare an html file to be "public" then, you can view that file as a web page! In fact, in this way, S3 acts as a web-server. Consider that this just makes sense for static web pages.



S3 objects can be used in the created web page (in the previous step). For that purpose, follow below steps:

1. Open the HTML source code of index.html on your local computer
2. In the last lines you can see some lines commented, uncomment them.
3. Copy pictures' URL from S3 (as explained above) and replace them with "FILE NAME" in the html code.
4. Upload the html file.
5. Make it public.
6. You should be able to see the web page with the pictures added.

Exercise 3. Minimize the response time

Background

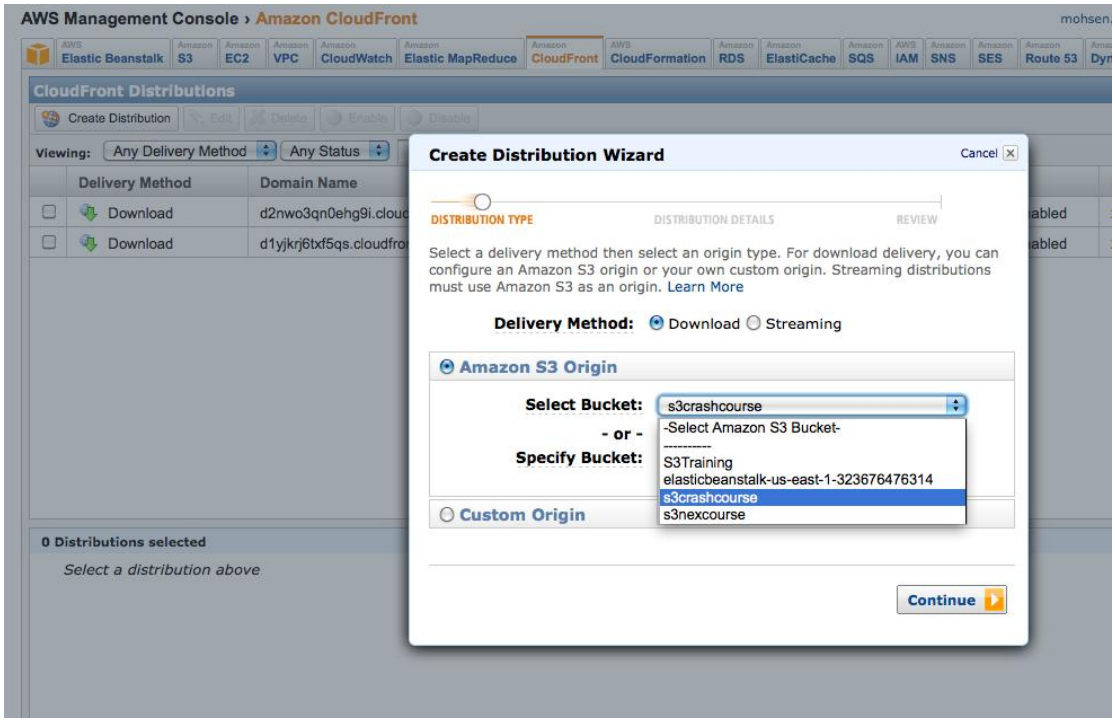
- Based on the S3 region, users in different geographic areas experience different latencies.
- Amazon CloudFront is a web service for content delivery.
- It offers businesses an easy way to distribute contents close to end users. Therefore, the latency will be low and data transfer speeds is high.

What should be able to do

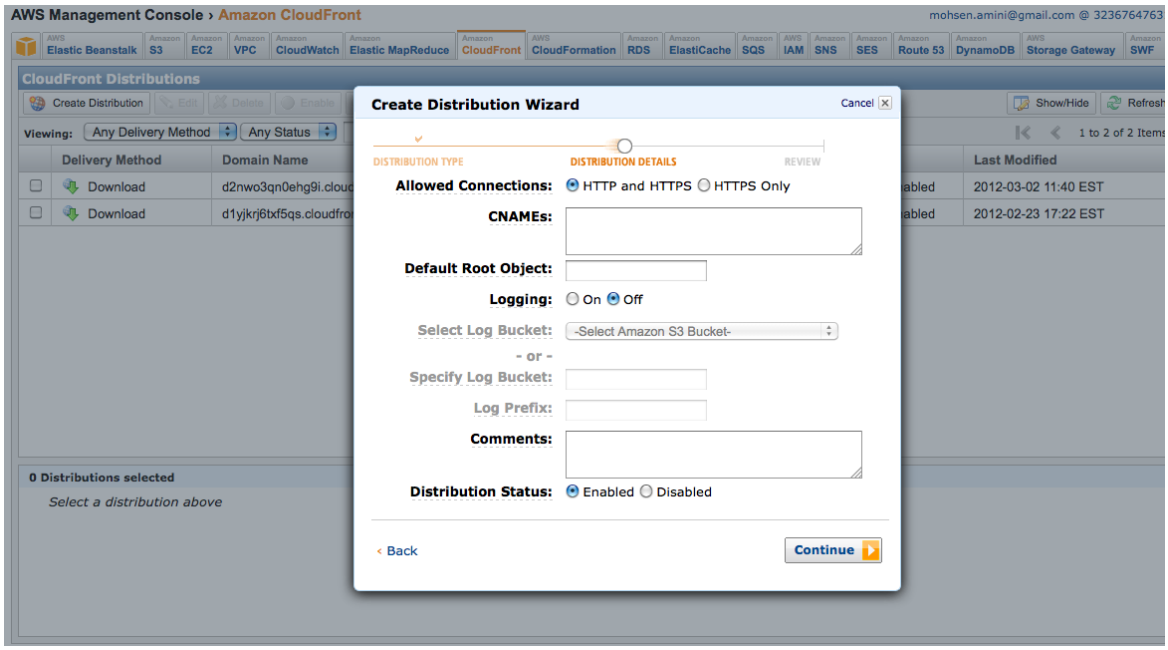
Connect the data in S3 to Amazon CloudFront and add the URL to the website. As a result the loading time of the web page is decreased.

Connecting Bucket to CloudFront

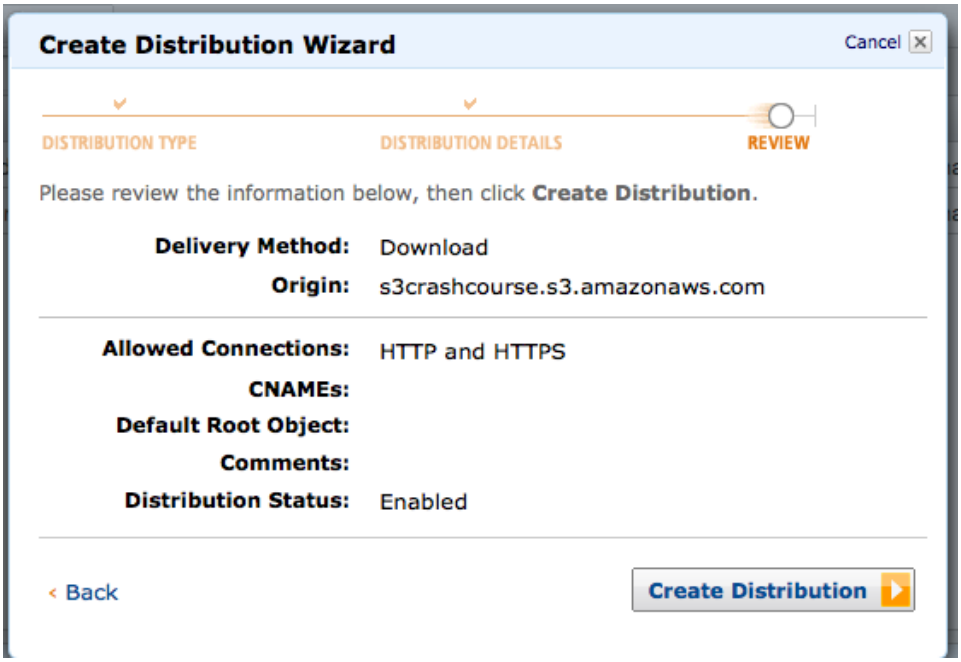
1. Create a new bucket (in this example: s3crashcourse).
2. Copy the content of the current bucket into the new bucket.
3. Click on the CloudFront in the Amazon management console.



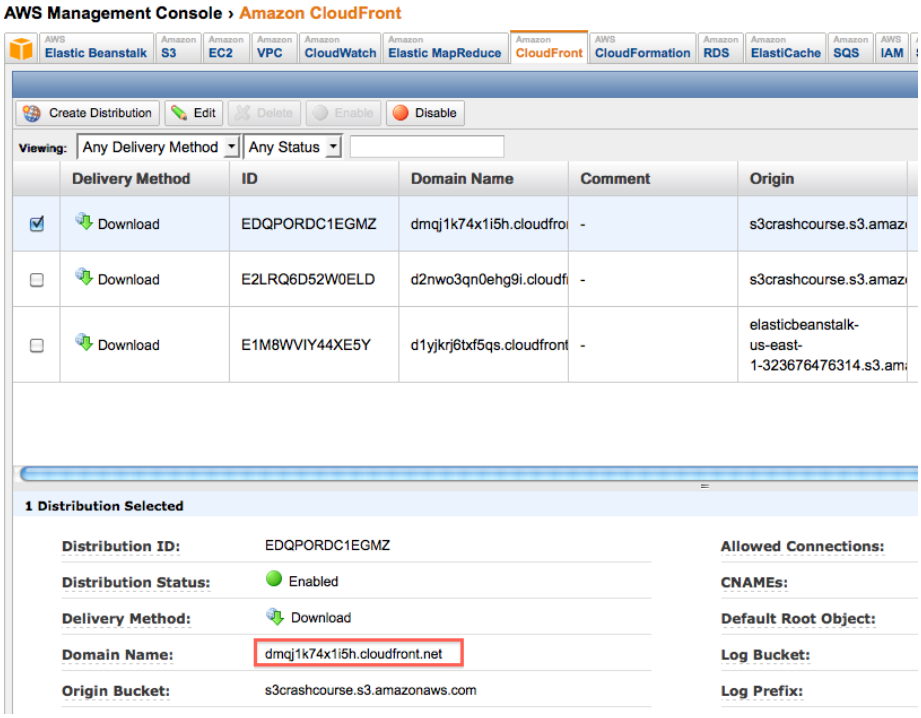
4. Select “s3crashcourse” from the dropdown list and press, “continue”.
5. In the next page, make sure that “HTTP and HTTPS” is selected and “Distribution Status” is enabled, then press “continue”.



6. In the next page, press “create distribution”



7. Wait until the created distribution changes to the "enabled" state.
8. Now the CDN-enabled URL of the pictures is constructed as follows:
   Bucket_domain_name/path_to_file_inside_bucket
9. Go to CloudFront page
10. Click on the created distribution. Its attributes appear underneath.
11. Copy the "domain name" from the CloudFront page.



12. Copy the object path from S3. For example, for 12.jpg we have:
    Pics/12.jpg
13. In the html source code of index.html, replace the CDN-based URL of the image for the FILE NAME.

14. Upload the modified index.html file.
15. Make the file public and copy the address it in the browser.
16. Now the pictures are downloaded both through the S3 and CloudFront.

Lab Session 2. EC2 and Beanstalk Labs

Exercize 1. Elastic Compute Cloud (EC2)

What should be able to do

1. EC2 components including availability zones, instances, images, security groups, ELB, auto-scaling, elastic IP.
2. Launch a new instance for your application server and database server
3. Giving your instance public IP address reservations
4. Creating a DMZ using Security Groups and Configuring the Firewall
5. Importing Your Own Virtual Machines
6. Configuring your Amazon Machine Image (modifying existing images, creating your own Image)
   Creating Scalability Groups to add auto scalability to your application

EC2 components

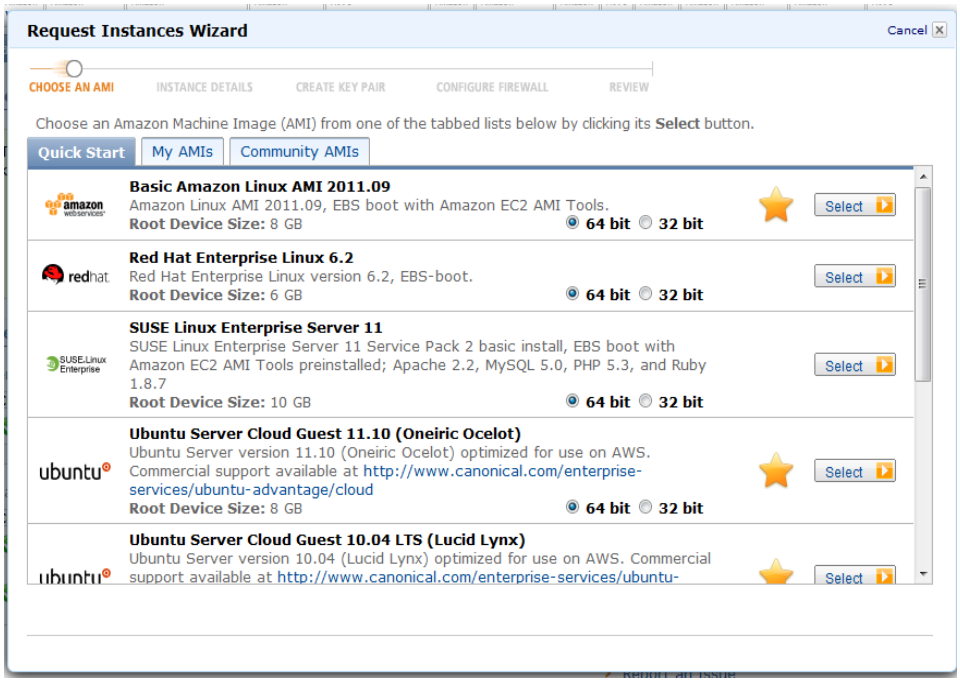
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that enables you to launch and manage Linux/UNIX and Windows server instances in Amazon's data centers.

Launching a Linux Instance

To launch a new instance you have to

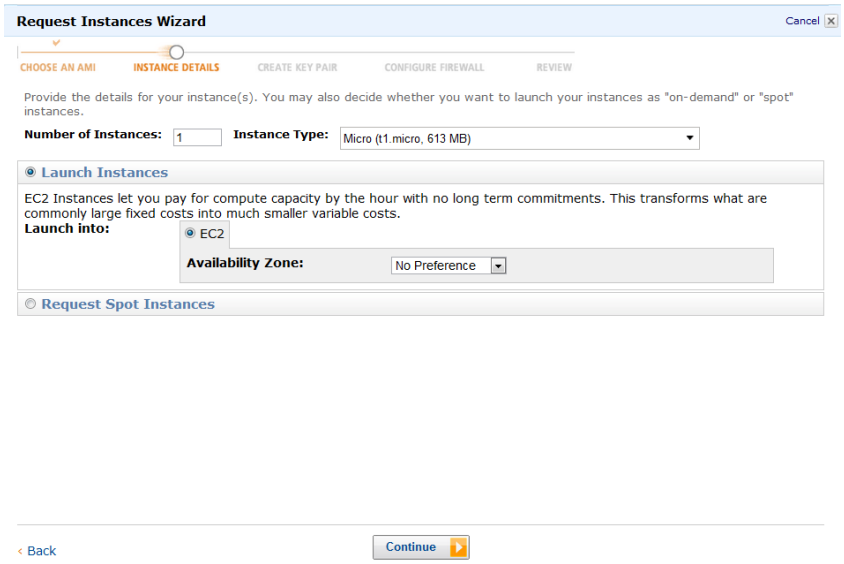
1 Choose the Region of your instance. For this lab exercise choose Singapore.

2 From the Amazon EC2 console dashboard, click Launch Instance. And choose Classic Wizard

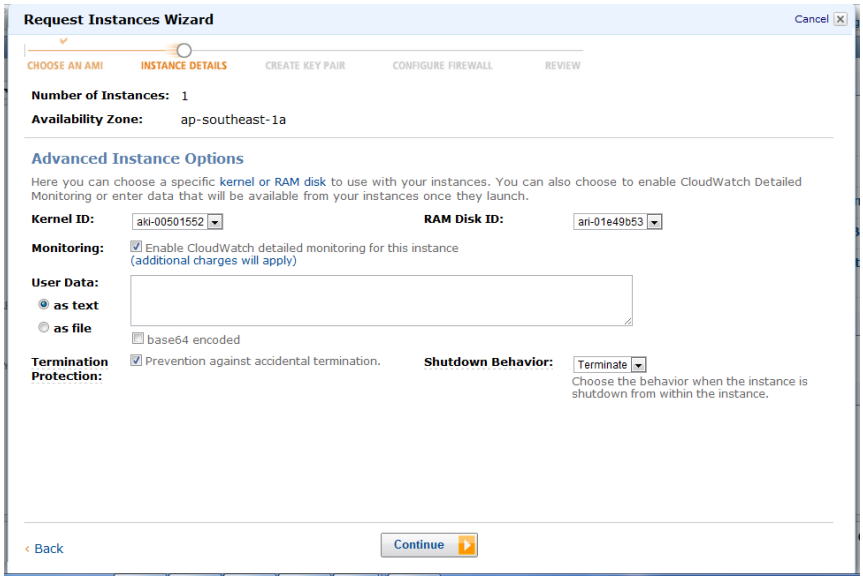


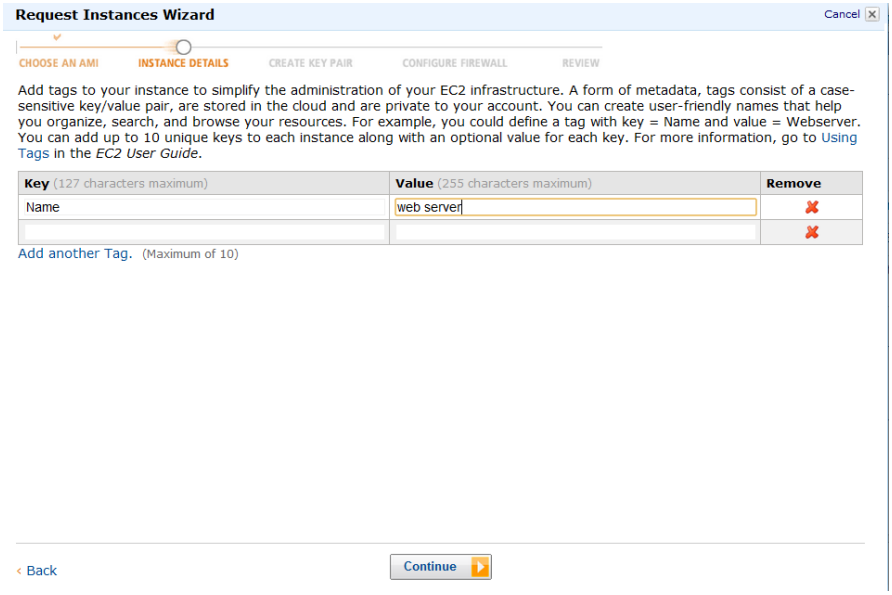
3 From Quick Start TAB, choose an AMI, for the Web instance. To make it 3-tier architecture, you will need another instance -from the same or another AMI- that will serve as the database server.

4 Next, Choose instance type as micro and leave the availability zone as default.

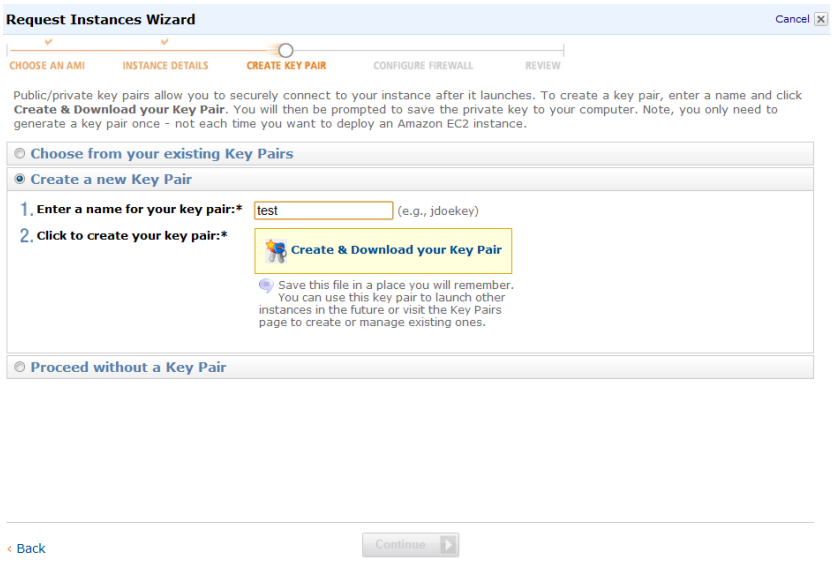


5 Set instance details as default. Name the web instance as “Web Server” and the database as “Database Server”





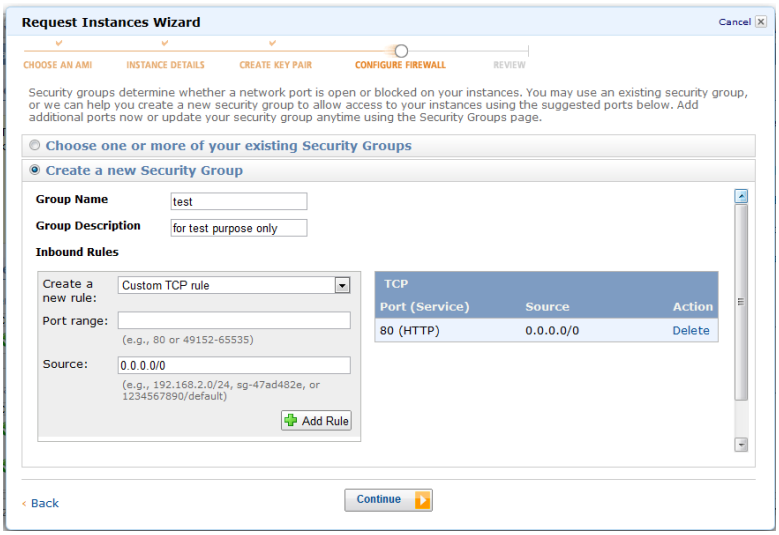
6 In this step you can create new key pair or choose form exiting ones. Public/private key pairs allow you to securely connect to your instance after it launches. Name it *test* and use it for both instances.



7 In this step you create *security groups*, which act as a firewall and determine whether a network port is open or blocked.

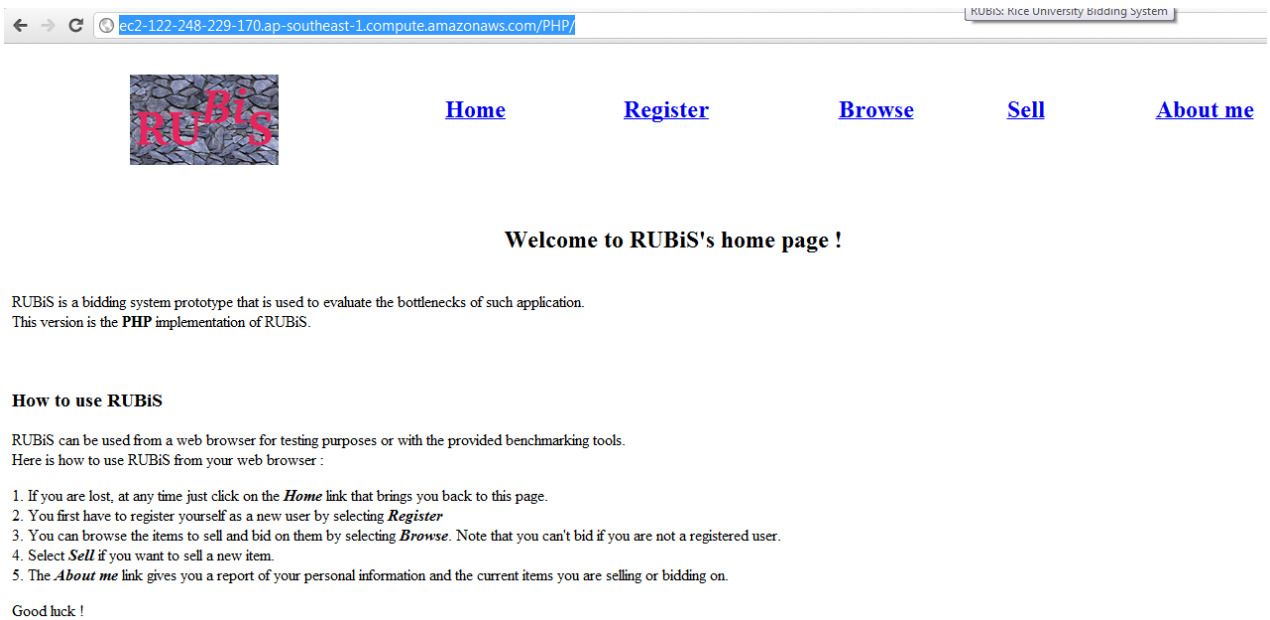
a. For Web Server instance open port 80 and 22. The source (-s) can be modified to allow connection only from particular network or IP for port 22.

b. For Database Server open 6618 for your Web server (set the source to your Web server IP address) and then port 22 for administrator access only.



8 Just review your setting and launch your instance.

9 Enter the "public DNS address/PHP" to your browser address bar and check whether your E-business site is up or not.



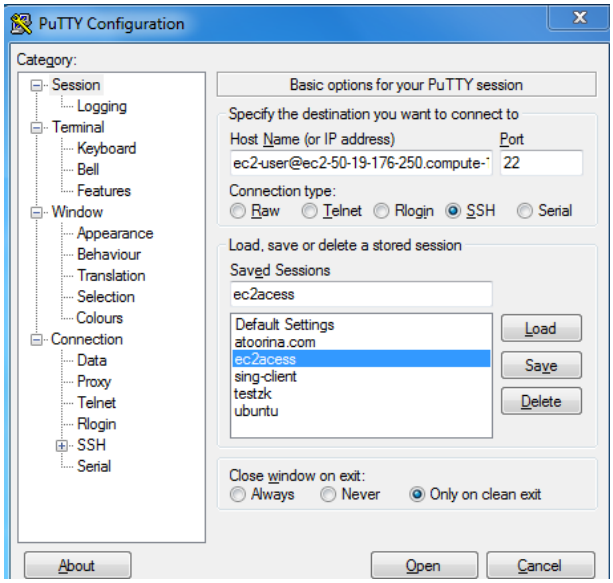
10 Click on Elastic IP in the navigation bar, and then on "Allocate New Address". Once created, associate them with your instances by right clicking on the IP addresses and chosseing the Accossiate option. Enter the "http://allocate_IP_address/PHP/" to your browser address bar to check whether it is asscoiated to your instance or not

Connecting to your instances from a Windows Machine Using PuTTY

1. You need to have *PuTTY*, *PuTTYgen* that is a key generation program, and *winscp* a graphical tool to securely transfer your files to your Linux instance.
2. Convert your private key to putty format. *PuTTY* does not natively support the private key format generated by Amazon EC2. *PuTTY* has a tool called *PuTTYgen*, which can convert keys to the required *PuTTY* format.
3. Load your *.pem* key to *PuTTYgen*
4. Click Save private key to save the key in *PuTTY's* format.

Configuring PuTTY to SSH to our instance

i. **Set Host Name field**: enter the public DNS name or IP address of your instance. If you launched an Amazon Linux instance, you can optionally prefix the DNS name with ec2-user@ to automatically log in as ec2-user when the session opens. If you used a different Amazon Machine Image (AMI) for your Linux/UNIX instance, you need to log in as the default user for the AMI. In this LAB exceise we use ubutnu image so the user name be "ubuntu"



ii. **Set Authentication Paramaters:** In the Category menu, under Connection, click SSH, and then Auth. Browse your private key file you generated in the preceding section.

iii. Save the session and open the SSH connection

Configuring Connection String in Web-Server instance

Once connected to the web Server instance, change directory to the PHP Directory by following command:

cd /var/www/PHP/

Here you can find all the PHP files of the E-business site, set the "link" variable in PHPprinter.php to point your database server by following commands:

nano PHPprinter.php

$link = mysql_pconnect("database server IP address", "root", "mysql")

Then save the file by pressing "Ctrl+o" and then press "Ctrl+x" to exit. Now you have to be able to browse categories (which are fetched from database) in web site by trying to following URL:

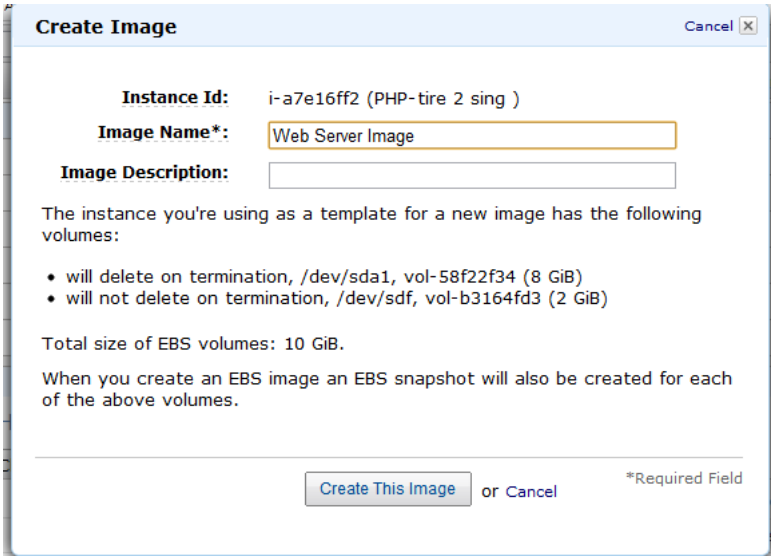
http://"Web Server IP address"/PHP/BrowseCategories.php

Creating your own Amazon Machine Image

To create an EBS image from existing instance:

1. Right click on the instance and choose "*Create Image*".

2. Set the name of image to "*Web Server Image*", and click on "*Create the Image*".
3. If you go to *Navigation bar>Images>AMIs* you can find your AMI listed there.



Exercize 2. Creating Scalability Groups to add auto scalability to your application

Auto Scaling is a web service designed to launch or terminate EC2 instances automatically based on user-defined policies, schedules, and health checks. It is currently support only horizontal scaling. Auto Scaling is useful for maintaining a fleet of Amazon EC2 instances that can handle the presented load. You can also use Auto Scaling to ensure that the instances in your fleet are performing optimally, so that your applications continue to run efficiently. Auto scaling groups can even work across multiple Availability Zones—distinct physical locations for the hosted EC2 instances—so that if an Availability Zone becomes unavailable, Auto Scaling will automatically redistribute applications to a different Availability Zone.

Type of Scaling

Amazon provides three types of auto scaling as manual scaling, scaling by schedule, and scaling by policy. Manual auto scaling is simply initiating a new instance when needed, and it is the most basic one therefore we are not discussing it here.

Scaling by Schedule to deal with seasonal load

It is applicable when we know exactly when load (this load calls seasonal load) is coming to the system. For example in the case of Amazon, Christmas is time when seasonal load hit the web site. In this type scaling actions are performed automatically as a function of time and date.

Scaling by Policy for flash crowd

We can define how you want to scale in response to changing conditions, but you don't know when those conditions will change. You can set up Auto Scaling to

respond for you. Note that you should have two policies, one for scaling up and one for scaling down, for each event that you want to monitor. For example, if you want to scale up when the network bandwidth reaches a certain level, you'll create a policy telling Auto Scaling to fire up a certain number of instances to help with your traffic. But you also want an accompanying policy to scale down by a certain number when the network bandwidth level goes back down.

An Auto Scaling scenario

In our scenario, our previously created Web server runs to cover the volume of customer traffic. And therefore we are creating **Auto Scaling groups** to cope with varying amount of our website traffic. **Auto Scaling groups** are defined with a minimum and maximum number of EC2 instances. The Auto Scaling service launches more instances (up to the defined maximum) for the Auto Scaling group to handle an increase in traffic and, as demand decreases, takes instances out of service to more efficiently use computing resources. In the following illustration, Internet traffic is routed from the public URL into an Auto Scaling group named *AS-web-server-group*.

The Auto Scaling group has **triggers** that increase or decrease the size of the Auto Scaling group based on the average CPU utilization for the whole group. Triggers are rules that tell the system when to add or subtract servers. When a trigger fires, Auto Scaling uses a **launch configuration** to create a new instance. Every Auto Scaling group you create has a launch configuration. Launch configurations enable you to describe the instance that Auto Scaling will create for you when a trigger fires. A **trigger** is a concept that combines two AWS features: a **CloudWatch** alarm (configured to watch a specified CloudWatch metric) and an **Auto Scaling policy** that describes what should happen when the alarm threshold is crossed. You can set a trigger to activate on any metric published to Amazon CloudWatch, such as CPUUtilization. When activated, the trigger launches a long-running process called a **Scaling Activity**. **Scaling Activity** is a process that changes the size of an Auto Scaling group. Capacity or size of the group can be changed by a percentage or by number.

1. Install the Auto-scaling API Command-Line Tools

The AWS Console does not support Autoscaling (yet). We'll install these tools on our local laptop machine. Download these from
http://aws.amazon.com/developertools/2535. (The link is case sensitive.)
These tools come in a zip file, just like the core EC2 API tools. Install them so that the files are in c:\ec2\bin and c:\ec2\lib. You may need to move subdirectories around in order to accomplish this.
Edit EC2.bat and add this line to the end of the file:

```
set AWS_AUTO_SCALING_HOME=c:\ec2
```

Then re-run the EC2.bat file, in order to update your command line environment with this variable.

2. Terminate Existing Instances

Autoscaling will launch instances as soon as you create an Autoscaling Group. Therefore, terminate the current instance in order to avoid confusion once Autoscaling kicks in.

3. Building the Autoscaling Script

The following script is a sample of building auto-scaling capability for our web server tire in the scenario.

1. It start with creating load balancer and then it create launch configuration which has the image id, instance type and the option for enabling the detailed monitoring the.
2. Then we create the auto scaling group we define which launch configuration to be used, what is the name of the load balancer.
3. the next step is making scaling policies which includes setting the scaling activity and cooldown parameter. The scaling activity is to change the capacity by adding 1 more instance and then cooldown for 300 second which is a period of time after a trigger is fired during which no other trigger activity can take place.
4. Finally, we have to define the alarm, which uses CPU Utilization metric from the cloud watch. The threshold of 60 % is defined for the alarm, therefore if in one period of 600 seconds CPU utilization is greater than 60%, an alarm would be triggered.

Setting the parameters

```
#!/bin/sh

#define the region

Region="ap-southeast-1"

#availability zone

ZONE="ap-southeast-1a"

#for the key name

KEY_NAME="test"

#define the used SECURITY_GROUP

SECURITY_GROUP="test"

#define the instance type for the scaling.

INSTANCE_SIZE="m1.small"

#define the name of the new load balancer here.

LB_NAME="my-load-balancer"
```

#define the name of the new launch configuration here.

LC_NAME="scalailityfor4000smallinstance"

#define the name of the image id is going to be used for scaling, which is the web server image created in the previous section.

LC_IMAGE_ID="ami-b22b50e0"

#define the name of the new scaling group here

SG_NAME="testAutoScaling"

Set up load balancer

elb-create-lb $LB_NAME --headers --listener "lb-port=80,instance-port=80,protocol=http" --availability-zones $ZONE --region $Region

elb-configure-healthcheck $LB_NAME --headers --target "HTTP:80/alive.php" --interval 6 --timeout 2 --unhealthy-threshold 2 --healthy-threshold 7

Create a launch config

as-create-launch-config $LC_NAME --image-id $LC_IMAGE_ID --instance-type $INSTANCE_SIZE --monitoring-disabled --key $KEY_NAME --group $SECURITY_GROUP --region $Region

Create an auto scaling group

as-create-auto-scaling-group $SG_NAME --availability-zones $ZONE --launch-configuration $LC_NAME --min-size 1 --max-size 2 --load-balancers $LB_NAME --region $Region

Set up scaling policies (scale up)

SCALE_UP_POLICY=`as-put-scaling-policy MyScaleUpPolicy1 --auto-scaling-group $SG_NAME --region $Region --adjustment=1 --type ChangeInCapacity --cooldown 300`

Setting up an alarm (scale up)

mon-put-metric-alarm MyHighCPUAlarm1 --comparison-operator GreaterThanThreshold --evaluation-periods 1 --metric-name CPUUtilization --namespace "AWS/EC2" --period 600 --statistic Average -threshold 60 --alarm-actions $SCALE_UP_POLICY --dimensions "AutoScalingGroupName=$SG_NAME" --

region $Region

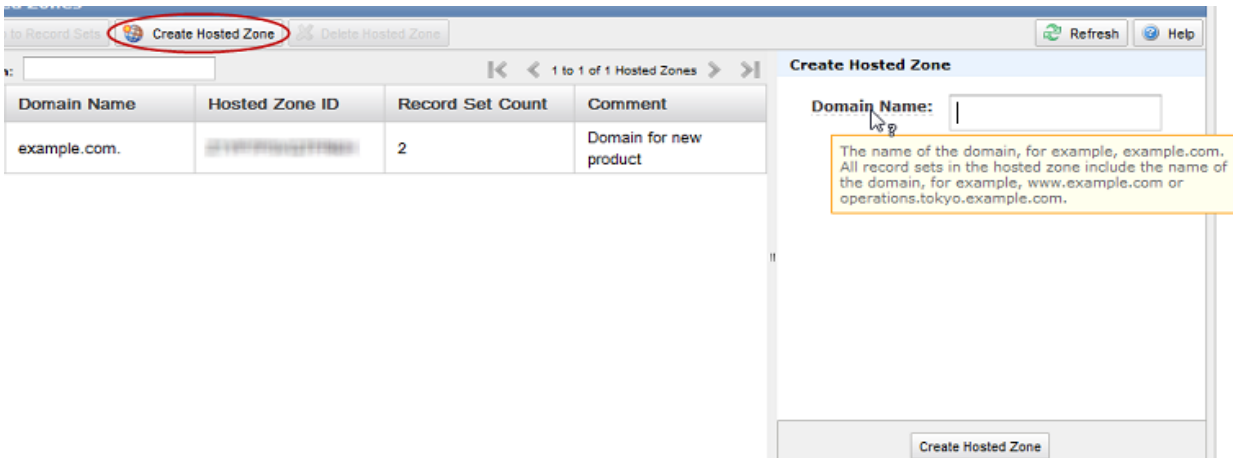
Set up scaling policies (scale down)

SCALE_DOWN_POLICY=`as-put-scaling-policy MyScaleDownPolicy1 --auto-scaling-group $SG_NAME --region $Region --adjustment=-1 --type ChangeInCapacity --cooldown 300`

mon-put-metric-alarm MyLowCPUAlarm1 --comparison-operator LessThanThreshold --evaluation-periods 1 --metric-name CPUUtilization --namespace "AWS/EC2" --period 600 --statistic Average --threshold 10 --alarm-actions $SCALE_DOWN_POLICY --dimensions "AutoScalingGroupName=$SG_NAME" --region $Region

Exercize 3. Link your Domain to your Web application (Utilizing AWS Route 53)

1. **Register your domain first**, for example visit domain.com, choose your domain name, and then purchase it for a desired period of time.
2. **Create a Hosted Zone:**
   i. In the Route 53 console, above the left pane, click Create Hosted Zone. In the right pane, enter a domain name and, optionally, a comment (for more information about a field, see the tool tip for the field).
   ii. Below the right pane, click Create Hosted Zone.

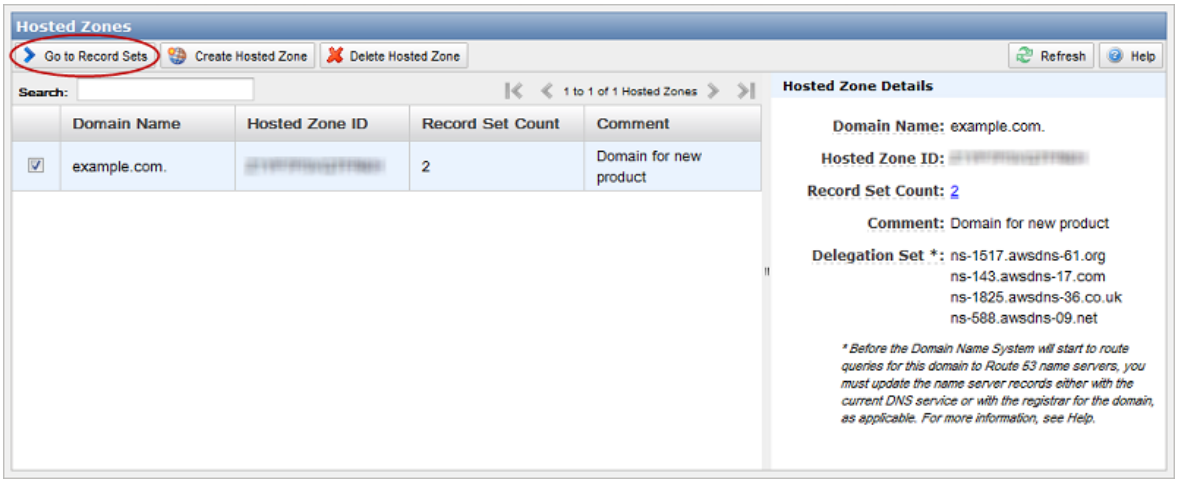


iii. Create Resource Record Sets in Your Route 53 Hosted Zone:

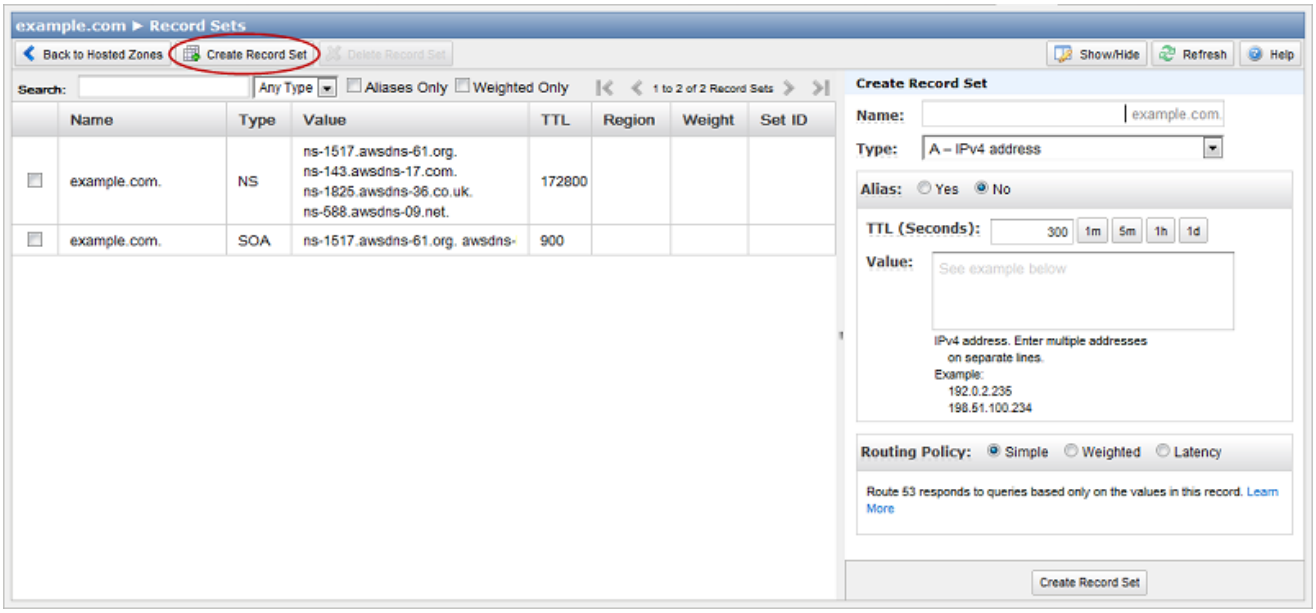
Now that you have a hosted zone, you can create resource record sets. For example, if you want a user who enters example.com in a web browser to be routed to a host (application on your web server) you would create a resource record set for example.com

with a Type of A and a Value of the Elastic IP you have associated earlier. To do that, follow the steps below:

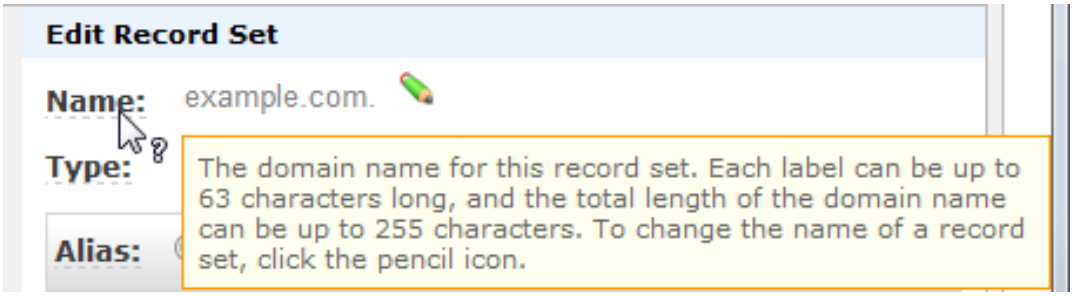
i. Click the row for the hosted zone in which you want to create record sets, and click Go to Record Sets. Double-click the row for the hosted zone.



ii. On the Record Sets page, above the left pane, click Create Record Set.



iii. In the right pane, enter the applicable values. For information about a field, see the tool tip for the field. Below the right pane, click Create Record Set.



iv. Simply, update the Registrar's Name Server Records to following name servers provided by AWS:

ns-1495.awsdns-58.org.
ns-1670.awsdns-16.co.uk.
ns-534.awsdns-02.net.
ns-275.awsdns-34.com.

Exercize 4. Beanstalk backup document

What should be able to do

1. Why AWS Elastic Beanstalk
2. How to start using AWS Elastic Beanstalk: creating, viewing, and updating your AWS Elastic Beanstalk application, as well as editing and terminating your AWS Elastic Beanstalk environment.
3. Describing different ways you can access AWS Elastic Beanstalk

"I have moved to Cloud to get rid of infrastructure management hassles, well here I am managing my AWS Infrastructure!"

Why AWS Elastic Beanstalk

AWS Elastic Beanstalk provides developers and systems administrators an easy, fast way to deploy and manage your application without having to worry about AWS infrastructure. If you already know the AWS resources you want to use and how they work, you might prefer AWS CloudFormation to create your AWS resources by creating a template. You can then use this template to launch new AWS resources in the exact same way without having to re-customize your AWS resources. Once your resources are deployed, you can modify and update the AWS resources in a controlled and predictable way, providing the same sort of version control over your AWS infrastructure that you exercise over your software.

No additional cost

There is no additional charge for AWS Elastic Beanstalk; you pay only for the underlying AWS resources that your application consumes. For details about pricing, see the AWS Elastic Beanstalk service detail page.

Using AWS Elastic Beanstalk

Create an Application

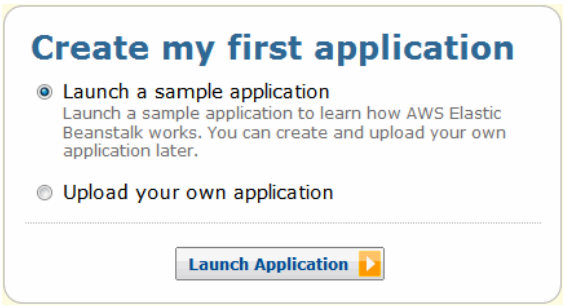
1. Open the AWS Elastic Beanstalk console at https://console.aws.amazon.com/elasticbeanstalk/.

2. Click the Launch Application button to start the application creation process.

To begin the process of creating the necessary components to run the sample application on AWS resources, AWS Elastic Beanstalk does the following:

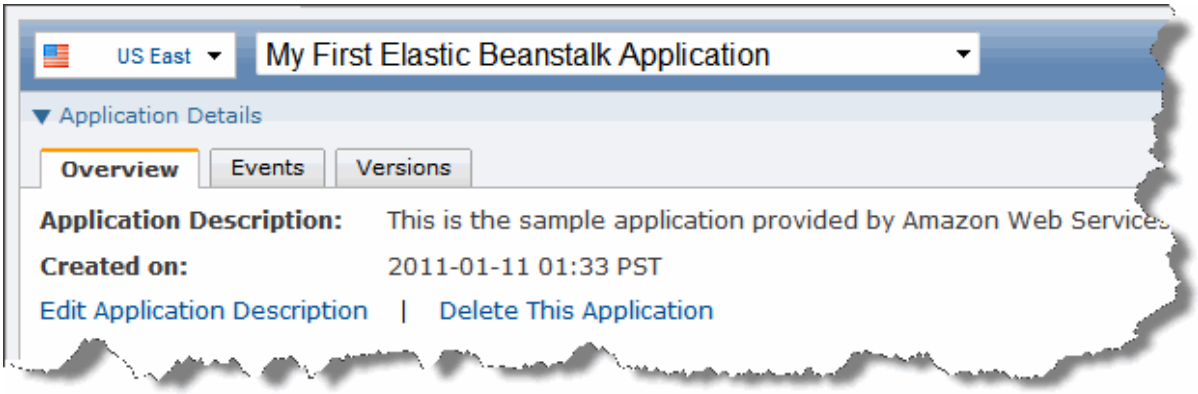
- Creates an AWS Elastic Beanstalk application named "My First Elastic Beanstalk Application."
- Creates a new application version labeled "Initial Version" that refers to a default sample application file.
- Launches an environment named "Default-Environment" that provisions the AWS resources to host the application.
- Deploys the "Initial Version" application into the newly created "Default-Environment."

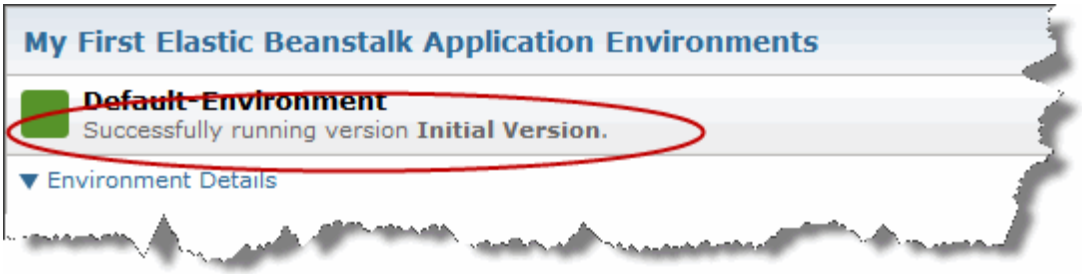
This process may take several minutes to complete.



View Application

After you create your application, the details and environment for the application appear in the AWS Management Console. The Application Details pane on the top of the console provides basic overview information about your application, including events associated with the application and all versions of the application. The Environment pane below the Application Details pane displays information about the Amazon EC2 instances that host your application, along with the AWS resources that AWS Elastic Beanstalk provisions when it launches your environment. While AWS Elastic Beanstalk creates your AWS resources and launches your application, the environment will be in a Launching state. Status messages about launch events are displayed on the environment's information bar.





To see the published version of your application

1. Click the **Environment Details** link in the **Environments** pane for your application.

The details appear for your application's environment.

2. Click the link the **URL** field in the **Overview** tab.

The application page opens in a new tab.

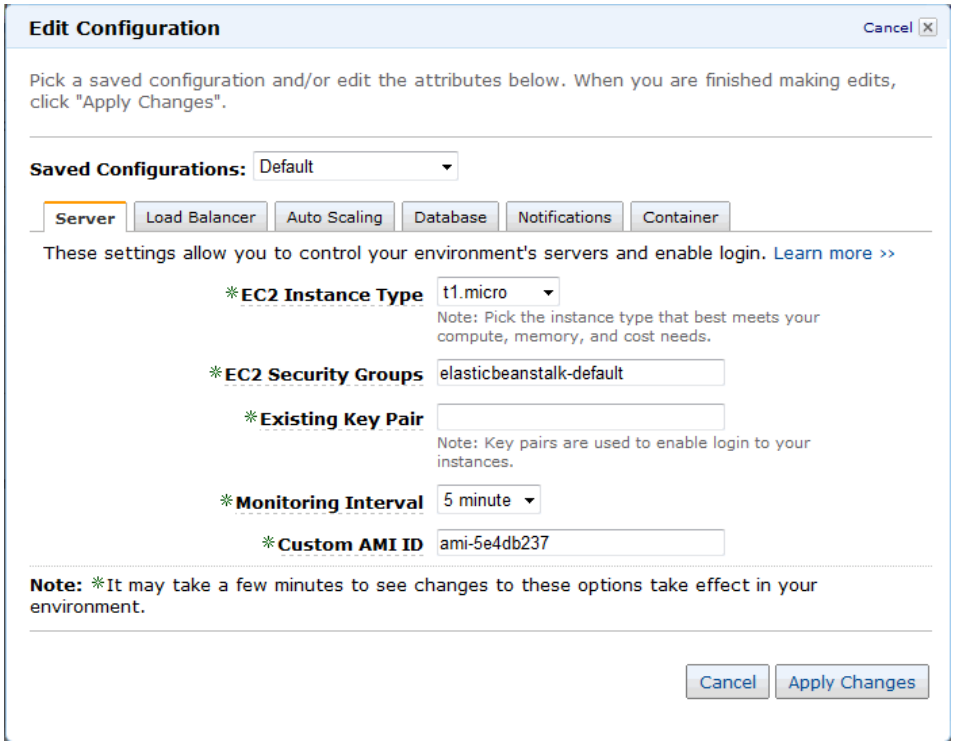
Change Configuration

You can customize your environment to better suit your application. For example, if you have a compute-intensive application, you can change the type of Amazon EC2 instance that is running your application. Some configuration changes are simple and happen quickly. Some changes require AWS Elastic Beanstalk to delete and recreate AWS resources, which can take several minutes. AWS Elastic Beanstalk will warn you about possible application downtime when changing configuration settings. In this task, you change the minimum instance settings for your Auto Scaling group from one to two and then verify that the change occurred. After the new instance gets created, it will become associated with your load balancer.

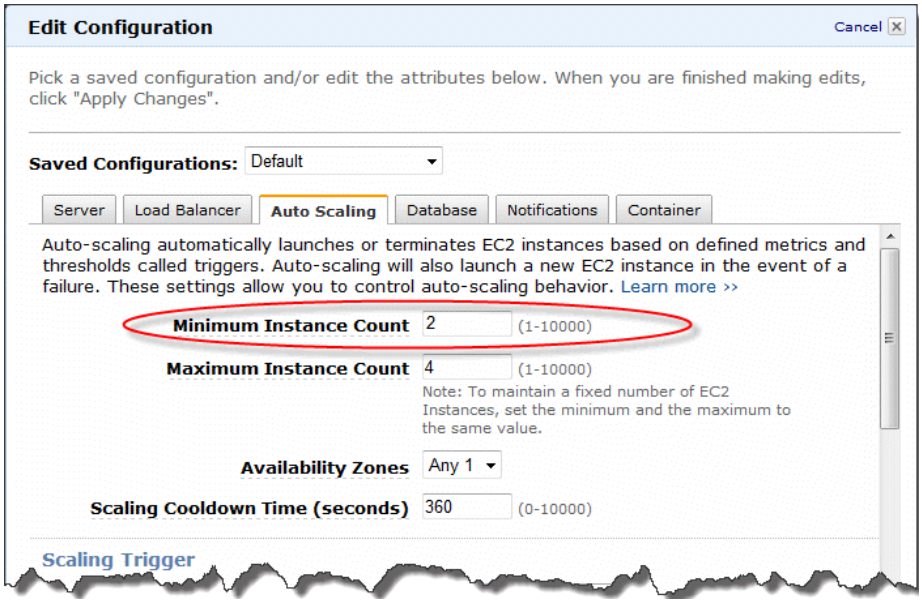
To change your environment configuration

1. Click the Actions drop-down menu on the right of the Environment pane, and select Edit/Load

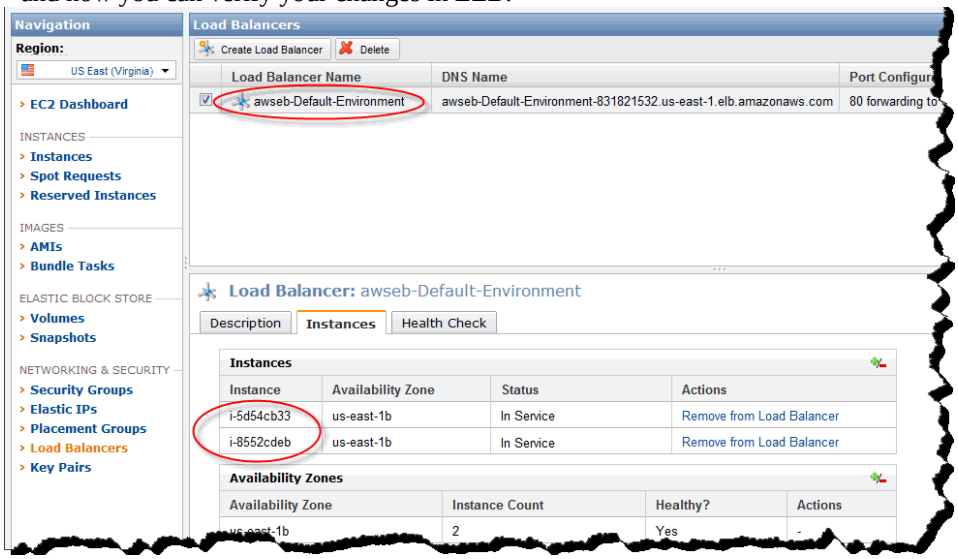
Configuration.



2. Click the **Auto Scaling** tab in the **Edit Configuration** dialog box.



3. Change Minimum Instance Count from 1 to 2. This change increases the minimum number of AutoScaling instances deployed in Amazon EC2.
4. Click Apply Changes. Wait for the environment's status to change from Updating to Ready, and now you can verify your changes in ELB.



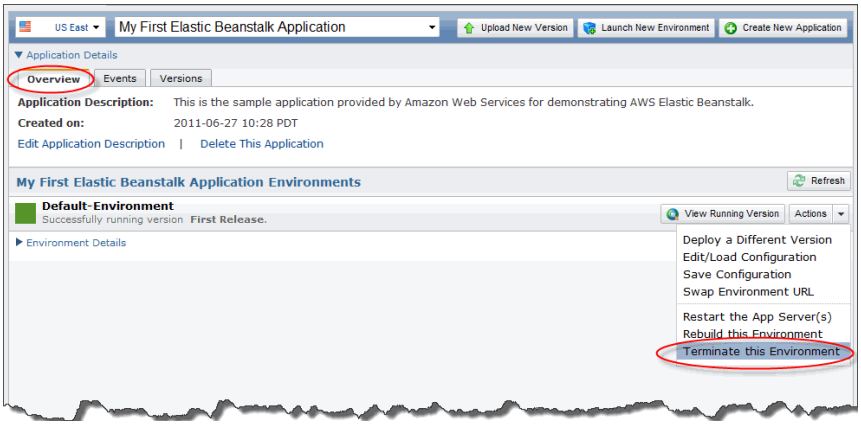
The information shows that two instances are associated with this load balancer, corresponding to the increase in Auto Scaling instances.

Clean Up

To make sure you are not charged for any services you don't need, you can clean up by deleting any unwanted applications and environments from AWS Elastic Beanstalk and AWS services. Verify that you are not using any AWS Elastic Beanstalk resources by reviewing your applications and deleting those you no longer need.

To completely delete the application

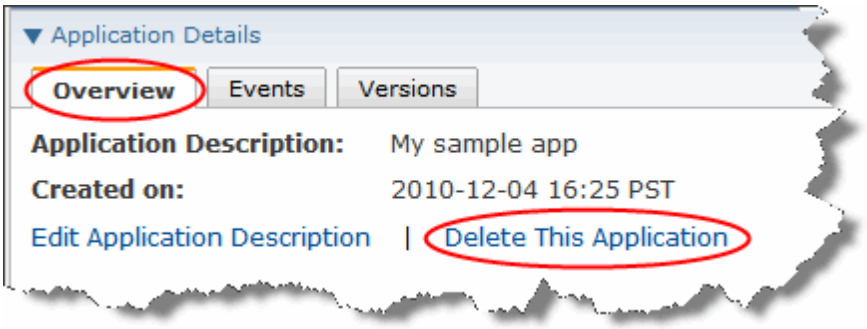
1. Terminate the environment:
    a. In the Application Details view, click the Overview tab.
    b. Click the Actions button next to the environment you want to delete and click Terminate this Environment.



The Terminate Environment dialog box appears.

c. Click the Terminate Environment button.

2. Delete all application versions:
   a. In the Application Details view, click the Versions tab.
   b. Select the check box next to your application versions and click the Delete Version button.
   c. In the Delete Application Version dialog box, select the Delete Version from Amazon S3 as well as check box.
   d. Click Yes, Delete.
3. In the Application Details view, click the Overview tab.



4. Click the Delete This Application link.
5. Click Yes, Delete.

Lab Session 3. Amazon RDS

This Lab walks you through creating and configuring your Amazon RDS DB Instance. We would cover the following subjects:

1) Create ›DB Parameter Groups
2) Create DB security group
3) Launch a DB instance
4) Create read Replicas for your database—RDS does not support auto scaling
5) Take snap shots and restoring your database to particular time
6) Connect to DB instance and migrate your database

Create DB Group

Go to http://aws.amazon.com and sign in using your credentials

Choose RDS tab, then from the navigator column at left select ›DB Parameter Groups

From the Amazon RDS Console Dashboard, Create ›DB Parameter Groups and select DB family (*mysql 5.1, or mysql 5.5*), then give it a name and description and press yes, create

Now it is a time to customize the DM parameters of the created group. This task sounds to be quite easy, but because the AWS Management Console does not allow the manipulation of parameter group values, you have to go the way using the API from your command line. Let's do it using the API tools from Amazon.

Install sun-java6-bin (if you dont have java installed yet!)

Download the API Client tools for RDS

Unpack them to /rds-tools

Move the file "credential-file-path.template" to credentials.txt. Open the file and copy the two keys from your AWS management console (click on 'Account' and then 'Security Credentials'!)

Set the AWS_RDS_HOME and AWS_CREDENTIAL_FILE variables

```
set  AWS_RDS_HOME=F:\amir\nextright\AWS tarining\RDSCli\RDSCli-1.6.001
set AWS_CREDENTIAL_FILE=G:\credential.txt
```

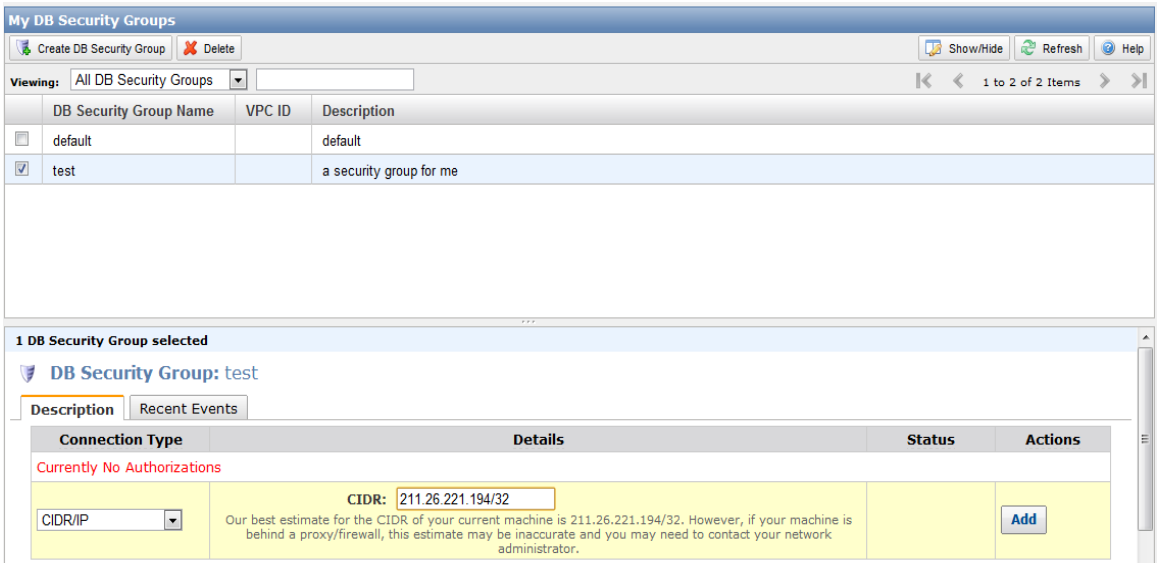
Now modify whatever parameter you want

```
CMD>rds-modify-db-parameter-group test – parameters "name=max_allowed_packet,  value=16777216, method=immediate"
```

Set up the security group

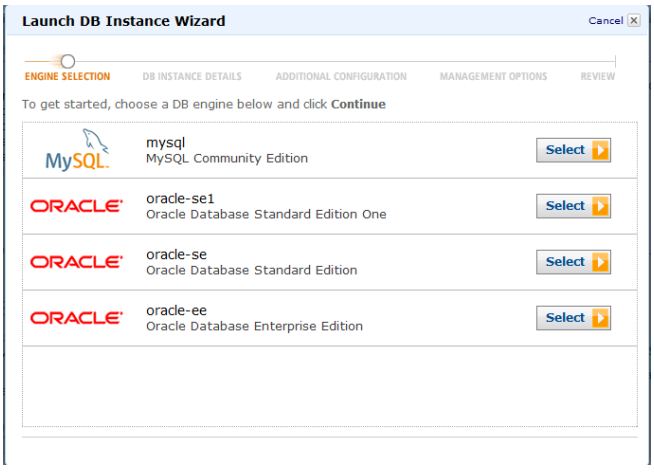
Choose the DB security group from navigator and click on the create security group. Create the security group by give it a proper name and description.

On the Description tab at the bottom of the window, select CIDR/IP from the Connection Type drop-down list, type the CIDR range for the ingress rule you would like to add to this DB Security Group into the CIDR text box, and click the Add button. The AWS Management Console displays the CIDR IP to use just below the CIDR text field if you want to authorize only the machine you are currently on.



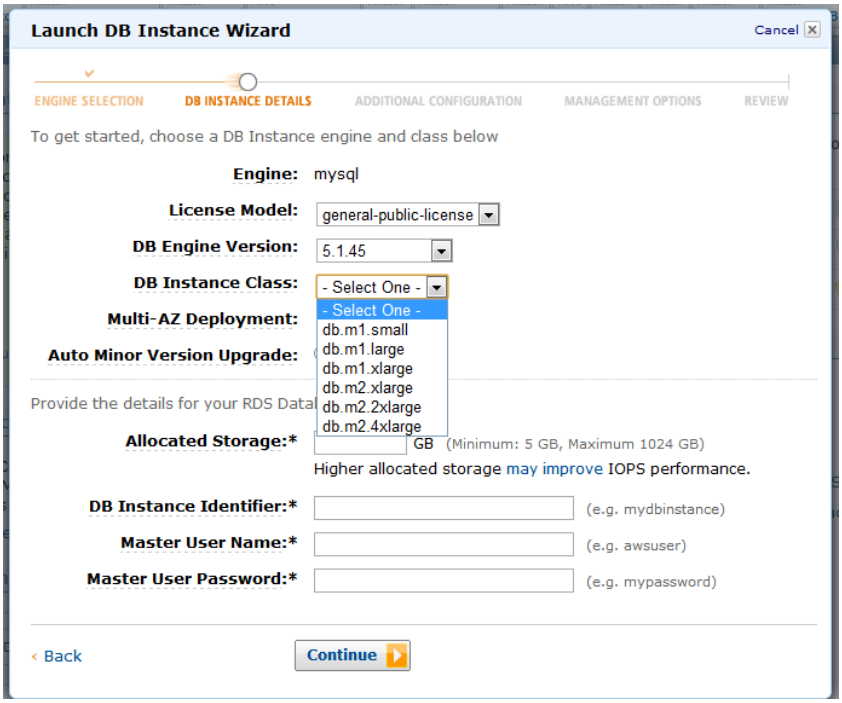
Launch a DB instance

- From the Amazon RDS Console Dashboard, click Launch DB Instance to start the Launch DB Instance Wizard. The wizard opens on the Engine Selection page.

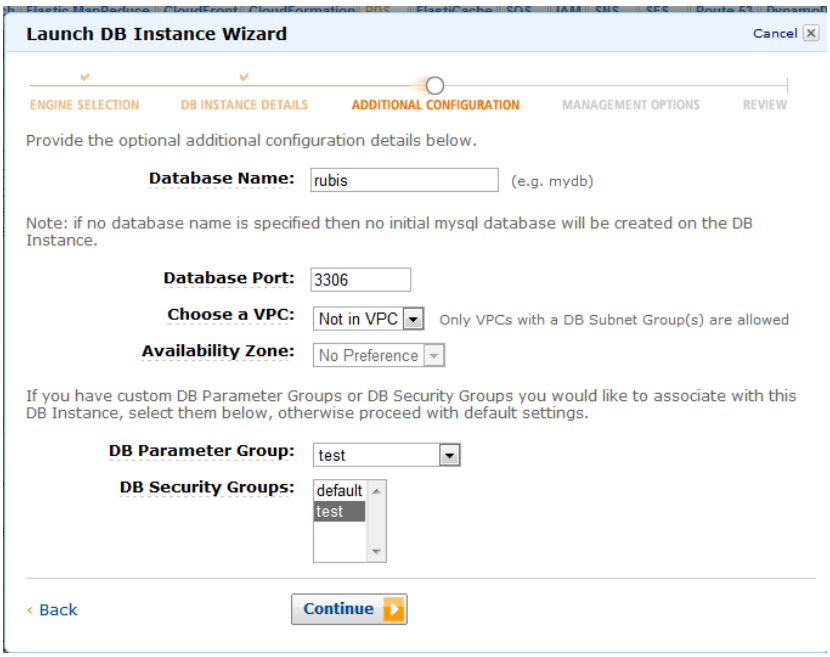


- Click the Select button next to the MySQL database engine. The wizard continues to the DB Instance Details page. The first page of the wizard displays a list of DB Instance Classes in the DB Instance Class drop-down list. The DB Instance class defines the CPU and memory capacity of your DB Instance. On the **DB Instance Details** page, specify your DB Instance details as shown in the following table, and then click **Continue**.

For this parameter...	...Do this:
License Model	Keep the default: `general-public-license`.
DB Engine Version	Select `5.1.45`
DB Instance Class	Select `db.m1.small`.
Multi-AZ Deployment	Select `No`.
Auto Minor Version Upgrade	Select `Yes`. The Auto Minor Version Upgrade option enables your DB Instance to receive minor engine version upgrades automatically when they become available.
Allocated Storage	You can specify how much storage in gigabytes you want initially allocated for your DB Instance. For this example, type `10`.
DB Instance Identifier	The DB Instance is a name for your DB Instance that is unique for your account in a Region. Type `mydbinstance` in the **DB Instance Identifier** text box.
Master Username	Type a name for your master user in the **Master Username** text box. You use the master user name to log on to your DB Instance with all database privileges.
Master Password	Type a password for your master user in the **Master User Password** text box.



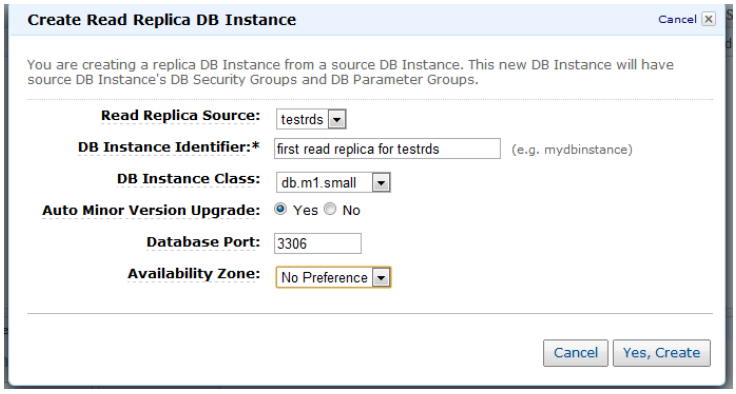
- After you click the Continue button, the Additional Configuration page opens where you can set the database name, port, DB parameters and security groups. Set it according to what is shown in the picture below. As you can see here we set to the DB parameters and security group to the groups we have created and set up in previous phases.



- After you click the Continue button, the Management Options page appears. The Management Options panel is where you can specify backup and maintenance options for your DB Instance. As you can see in the table if you set the back retention period to a positive number the automated backup will be enabled. And if you set it to 0, the automated backup will be enabled. Setting the other options allows you to set the daily range during which automated back and maintenance can happen. Once done you can continue to REVIEW panel to check your settings and if all the options are set correctly the DB can be launched by pressing the "launch DB instance" button.

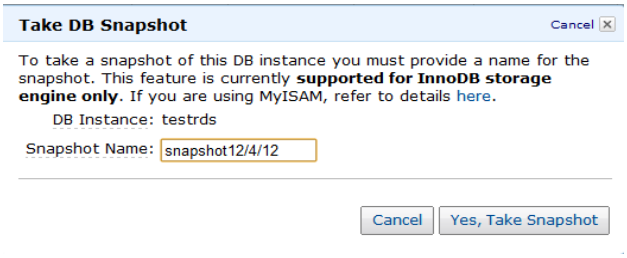
Creating Read Replica

- The Amazon RDS Read Replica feature lets you create a DB Instance as a replica of another DB Instance. In AWS console right click on the DB instance and select "create read replica", then choose the options (name, instance size, port number) of the read replica DB instance as shown in the picture below. Alternatively or use rds-create-db-instance-read-replica command line tool and specify the source DB Instance that you want to replicate. You can create up to five Read Replicas per DB Instance. When you initiate the creation of a Read Replica, Amazon RDS takes a snapshot of the source DB Instance and begins replication. As a result, you will experience a brief I/O suspension on your source DB Instance as the snapshot occurs (this I/O suspension is mitigated if the source DB Instance is a Multi-AZ deployment, because snapshots are taken from the Multi-AZ standby).



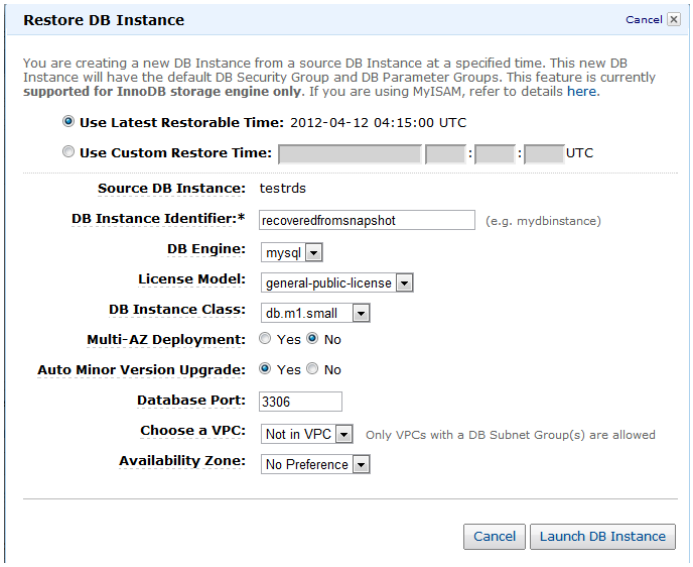
Take snap shots and restoring your database to particular time

- In AWS dashboard right-click on the DB instance and select "take snapshot". As following figure shows, you only need to just give it a proper name.



Restoring from a DB snapshot

- Click **DB Snapshots** in the **Navigation** list on the left side of the window.
- Click on the DB Snapshot that you want to restore from in the **My DB Snapshots** list.
- Click the **Restore from DB Snapshot** button.
- The **Restore DB Instance** window appears.
- Type the name of the restored DB Instance in the **DB Instance Identifier** text box, select the instance type and set the port number as appears in the figure below
- Click the **Launch DB Instance** button.



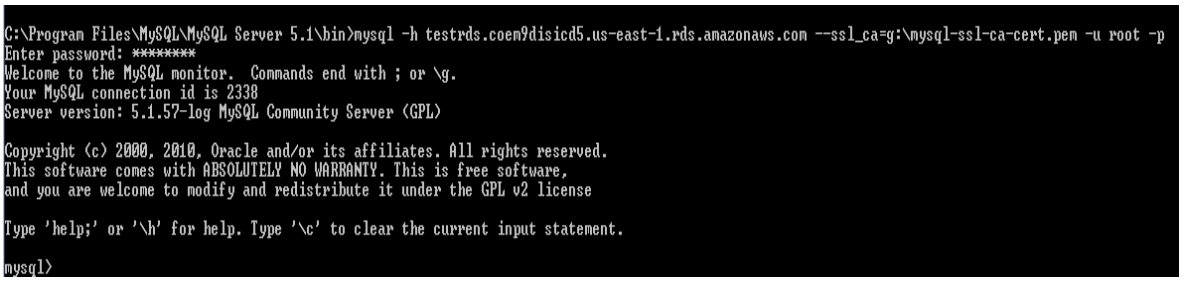
Connect to DB instance and migrate your database

- To connect to the DB instance you need a standard SQL client application. In this example, you connect to a DB Instance running the MySQL database engine using the MySQL command line tools.

To connect to a DB Instance with SSL using the MySQL monitor

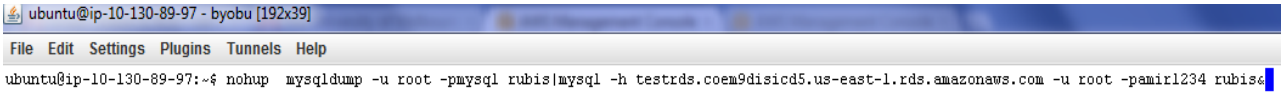
Download the public key for the Amazon RDS signing certificate from https://rds.amazonaws.com/doc/mysql-ssl-ca-cert.pem.

Type the following command at a command prompt to connect to a DB Instance with SSL using the MySQL monitor; substitute the DNS name for your DB Instance and the SSL certificate file name as appropriate. Enter the master user password when prompted.



In order to migrate your in-house databases to RDS, use mysqldump and run it in background. It is particularly effective for small amounts of data already stored in MySQL, the simplest way to transfer it to Amazon RDS is to extract

the data with mysqldump and pipe it directly into Amazon RDS. Here is an example:



- If the data stored in your MYSQL database is bigger than a 1 GB, then its best to migrate using mysqlimport. We'll start by listing the steps in order and discuss the details of each separately.
    - Create flat files containing the data to be loaded
        - On Unix-like systems (including Linux), use the 'split' command. For example, the following command splits the sales.csv file into multiple files of less than 1GB, splitting only at line breaks (-C 1024m). The new files will be named sales.part_00, sales.part_01, etc.

      split -C 1024m -d sales.csv sales.part_

    - Stop any applications accessing the target DB Instance
    - Create a DB Snapshot
    - Disable Amazon RDS automated backups
    - Load the data using mysqlimport
        - Use the mysqlimport utility to load the flat files into Amazon RDS. In the example we tell mysqlimport to load all of the files named "sales" with an extension starting with "part_". This is a convenient way to load all of the files created in the "split" example. Use the --compress option to minimize network traffic. The --fields-terminated-by=',' option is used for CSV files and the --local option specifies that the incoming data is located on the client. Without the --local option, MySQL will look for the data on the database host, so always specify the --local option.

```
mysqlimport --local --compress --user=username --password --
host=hostname \

--fields-terminated-by=',' Acme sales.part_*
```

    - Enable automated backups again

Lab Session 4. SQS and VPC

Exercize 1. Simple Queue Service (SQS)

What should be able to do

Develop a loosely coupled application that interact over a common queue (SQS)

Problem

•How can we enable the user to upload the pictures in any name, size, or format?
•The objective is building a solution that abstracts users from detailed limitations of uploading pictures to the web site.

Steps

1. Pov-ray is already installed on your instance. You can do a test by entering the following command:

```
povray/home/ubuntu/povray/povray-
3.6/scenes/advanced/benchmark.ini
+I/home/ubuntu/povray/povray-
3.6/scenes/advanced/landscape.pov +FT +W100 +H50
```

2. There are two jar files on your EC2 instance. You can find them in *SQSconsumer* directory.

3. There is a file in confing/prop.properties that you can set different attributes. Including bucket name that used by producer and consumer.

4. *prod.jar* is the producer:
    a. Run the jar file:

        Java –jar prod.jar

    b. Objects are read from the source bucket
    c. Objects are downloaded to the specified address of EC2 instance.
    d. It puts a message includes the file name into the queue.
5. *Consumer.jar* is the consumer program:
    a. Create another ssh connection to the EC2 instance and run the jar file:

        Java –jar consumer.jar

    b. Reads messages from the queue
    c. Renders the file mentioned in the queue.
    d. Uploads the result into destination bucket in S3.

You can follow documentation in java source codes from *consumer* and *producer* directories. Here we describe the main parts of *producer* and *consumer* code.

In *Producer* code:

//This class is in charge of downloading objects from S3 bucket to the processing nodes (EC2 instances) and send a message to SQS

```
public class Producer {

public static void main(String[] args) throws Exception {
```

// AWS access credentials

```
AmazonSQS sqs = new AmazonSQSClient(new
PropertiesCredentials(Producer.class.getResourceAsStream("AwsCredenti
als.properties")));

AmazonS3 s3 = new AmazonS3Client(new
PropertiesCredentials(Producer.class.getResourceAsStream("AwsCredenti
als.properties")));

try {
```

// Create a queue

```
System.out.println("Creating a new SQS queue.\n");

CreateQueueRequest createQueueRequest = new CreateQueueRequest(

                                "QUEUENAME");

String myQueueUrl = sqs.createQueue(createQueueRequest)

                                .getQueueUrl();
```

//we should maintain a sequence number to keep the order of items in SQS or identify the last element in the queue.

```
int seqNo = 0;
```

//##############################

//CODE BELOW IS FOR DOWNLOADING FROM S3 BUCKET

//##############################

```
System.out.println("Reading objects from S3");

ObjectListing objectListing = s3.listObjects(new
ListObjectsRequest().withBucketName(BUCKETNAME));

for (S3ObjectSummary objectSummary :
objectListing.getObjectSummaries()) {

      System.out.println(" -- " + objectSummary.getKey() + " --");

      System.out.println("Downloading an object bucket:" + BUCKETNAME

                        + " and Key:" + objectSummary.getKey());

      S3Object object = s3.getObject(new GetObjectRequest(BUCKETNAME,

                                    objectSummary.getKey()));
```

//copyfile is a function developed by ourselves.

```
      copyFile(object.getObjectContent(), objectSummary.getKey());
```

```
        seqNo++;

        System.out.println("Copying of the picture done.\n");
```

```
//##############################
```

//CODE ABOVE IS FOR DOWNLOADING FROM S3 BUCKET

```
//##############################
```

// After downloading file successfully, send a message to queue.

```
        System.out.println("Sending a message to Queue.\n");
```

//Constructing a message for queue. Objectname:sequenceNo

```
        String txtMsg=objectSummary.getKey()+":"+seqNo;

        SendMessageRequest msg = new
SendMessageRequest(myQueueUrl,txtMsg);
```

//Sending message to the queue

```
        sqs.sendMessage(msg);

}
```

//Generating a new sequence number for the last message.

```
        seqNo++;
```

//constructing the last message

```
        String endMsg="endq"+":"+seqNo;
```

//sending the last message into the queue.

```
        sqs.sendMessage(new SendMessageRequest(myQueueUrl,endMsg));

}
```

//Exception handling for any possible error

```
catch (AmazonServiceException ase) {
                    //Exception handling
              } catch (AmazonClientException ace) {
                    //Exception handling
              }

}
```

Consumer Code:

//This class is in charge of reading messages from queue and does the processing.

```
public class Consumer {

public static void main(String[] args) throws Exception {
```

// AWS access credentials

```
AmazonSQS sqs = new AmazonSQSClient(new
PropertiesCredentials(Consumer.class

      .getResourceAsStream("AwsCredentials.properties")));

AmazonS3 s3 = new AmazonS3Client(new
PropertiesCredentials(Consumer.class.getResourceAsStream("AwsCredenti
als.properties")));

int rcvCounter = 0;

boolean sw = false;

try {
```

//Getting the queue id (URL)

```
GetQueueUrlRequest getQueueUrlRequest = new
GetQueueUrlRequest("mohsenTest");

String myQueueUrl =
sqs.getQueueUrl(getQueueUrlRequest).getQueueUrl();

do {
```

// Retrieve messages (which is the file name):

```
System.out.println("Receiving messages from Queue.\n");

ReceiveMessageRequest receiveMessageRequest = new
ReceiveMessageRequest(myQueueUrl);

String msgBody = null;

String[] msgExtract = msgBody.split(":");

int seqNo = Integer.parseInt(msgExtract[1]);

if (msgExtract[0].equalsIgnoreCase("endq") && seqNo == rcvCounter +
1) {

      sw = true;

      break;

      } else {
```

//If the read message is “endq” but the sequence number is not consistent, it means that there are still other items in the queue.

```
      if (msgExtract[0].equalsIgnoreCase("endq") && seqNo !=
      rcvCounter + 1) {

            continue;

            }

}
```

//DO the main processing (here we simply consider renaming filenames).

```
String key = PATH + msgExtract[0];

String newName = key + ".test";
```

```
String cmdString = "mv " + key + " " + newName;

Runtime.getRuntime().exec(cmdString);

System.out.println("Processing of the picture done.\n");
```

// After processing done, write back the file to S3

```
System.out.println("Uploading the result file to S3.\n");

File outFile = new File(newName);

s3.putObject(new PutObjectRequest(BUCKETNAME, msgExtract[0]+ ".test",
outFile));

rcvCounter++;
```

// Delete the read message from the queue.

```
System.out.println("Deleting a message.");

String messageRecieptHandle = messages.get(0).getReceiptHandle();

sqs.deleteMessage(new
DeleteMessageRequest(myQueueUrl,messageRecieptHandle));

}
```

//Checks if the queue end is received (therefore sw=true), then exits the loop.

```
if (sw == true) {

            break;

            }

} while (true);
```

// We can delete the queue if it is not needed anymore.

```
// sqs.deleteQueue(new DeleteQueueRequest(myQueueUrl));

      } catch (AmazonServiceException ase) {

                  //Exception handling code.

      } catch (AmazonClientException ace) {

                  //Exception handling code.

      }

}
```

Exercize 2. VPC

What should be able to do

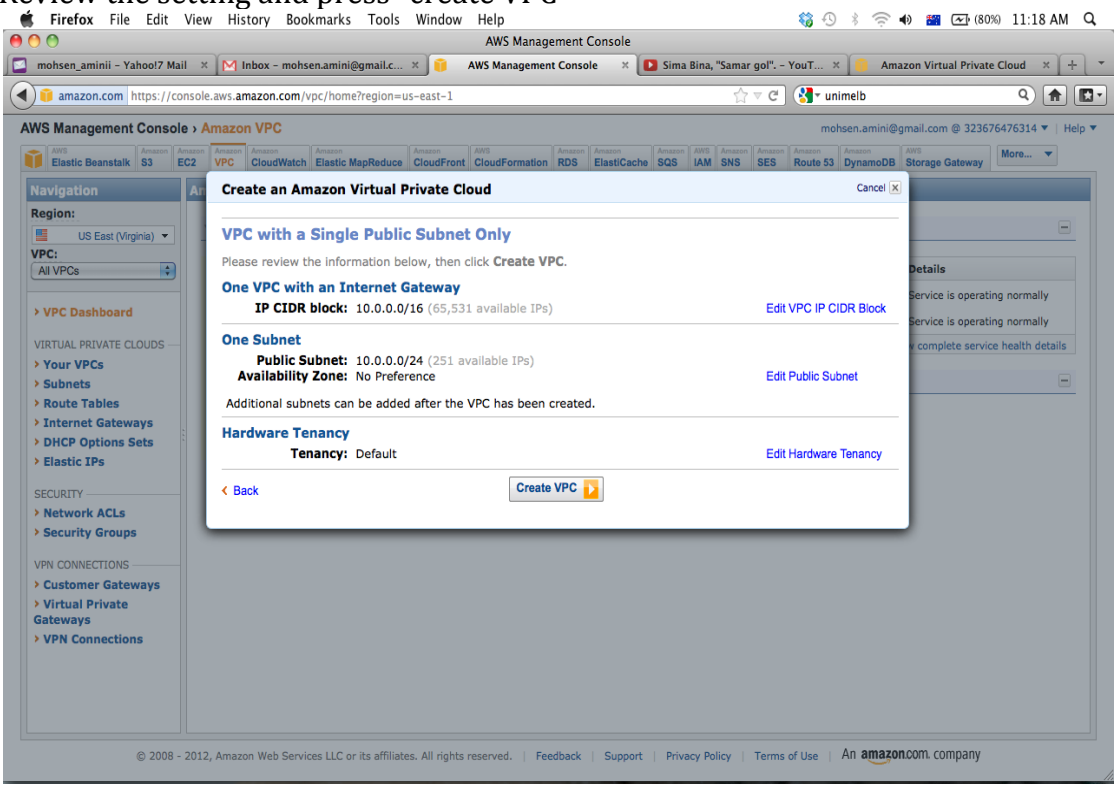
Problem

Create a VPN connection to an Amazon EC2 instance.

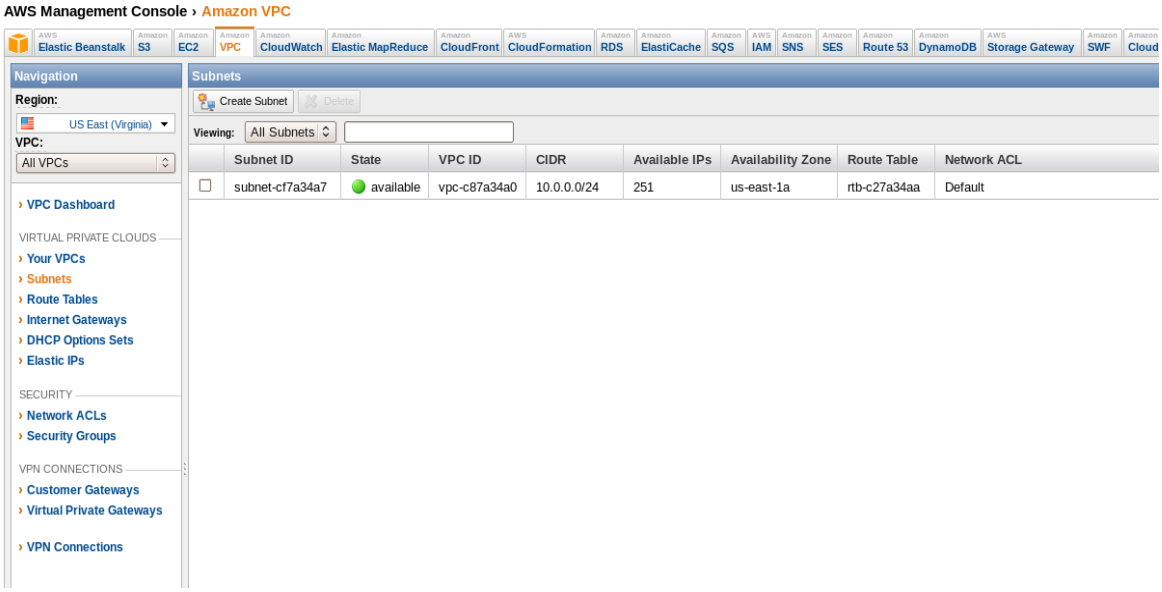
Steps

1. In the management Console, click on VPC tab.
2. Select “Get started create VPC”.

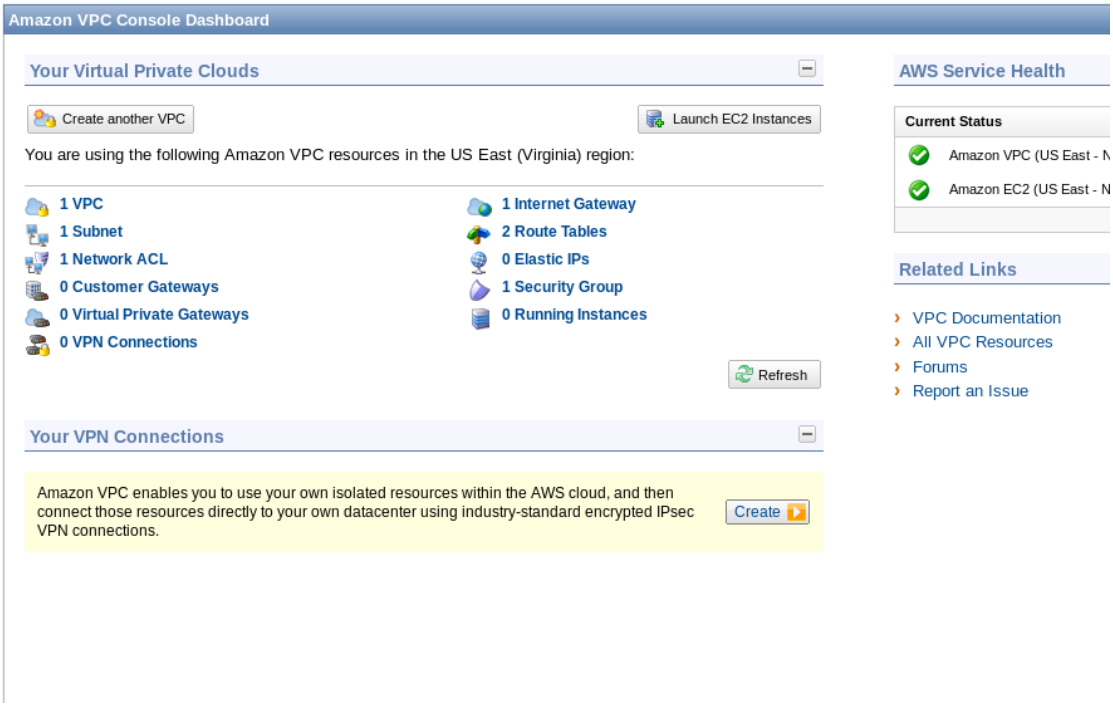
3. Review the setting and press “create VPC”



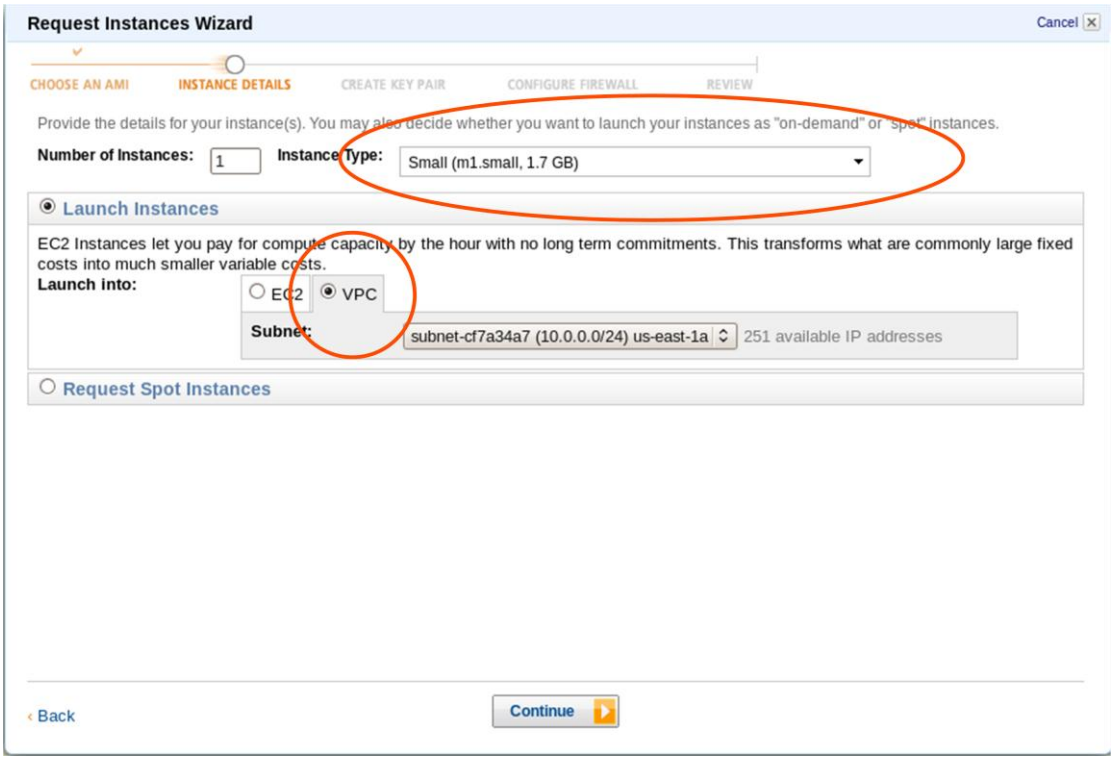
4. In this stage, a subnet is created and you can see it:



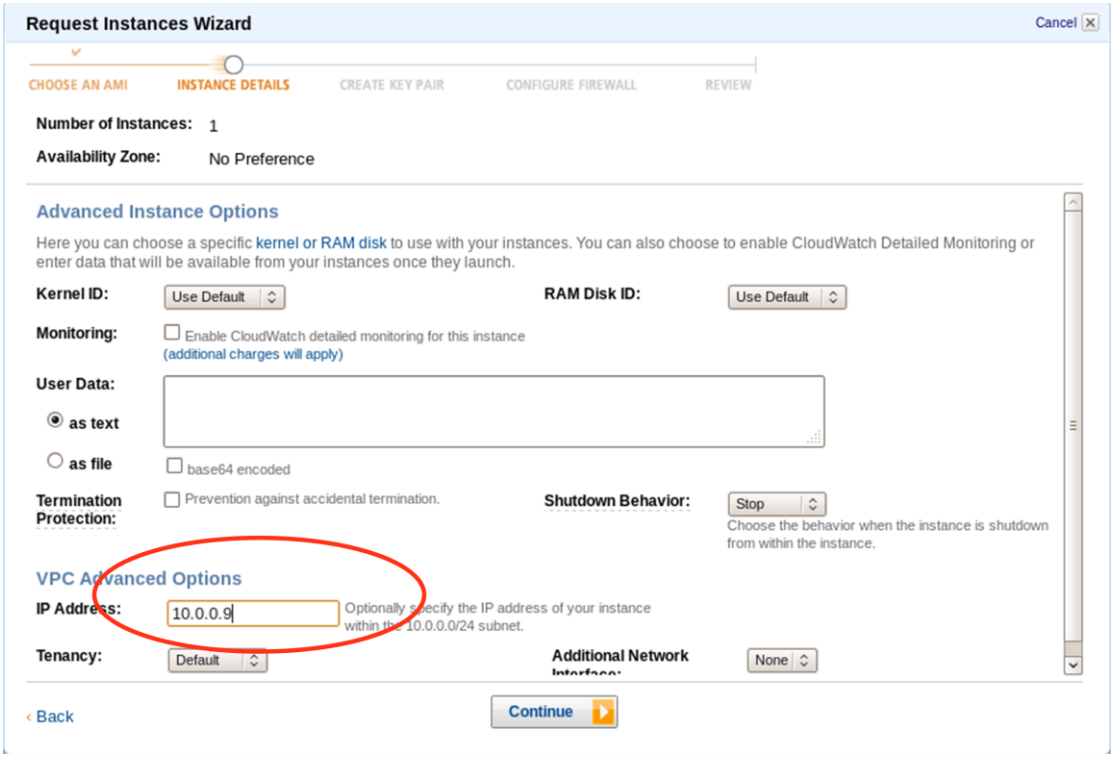
5. To create an instance within the subnet click on the “Launch EC2 instance”



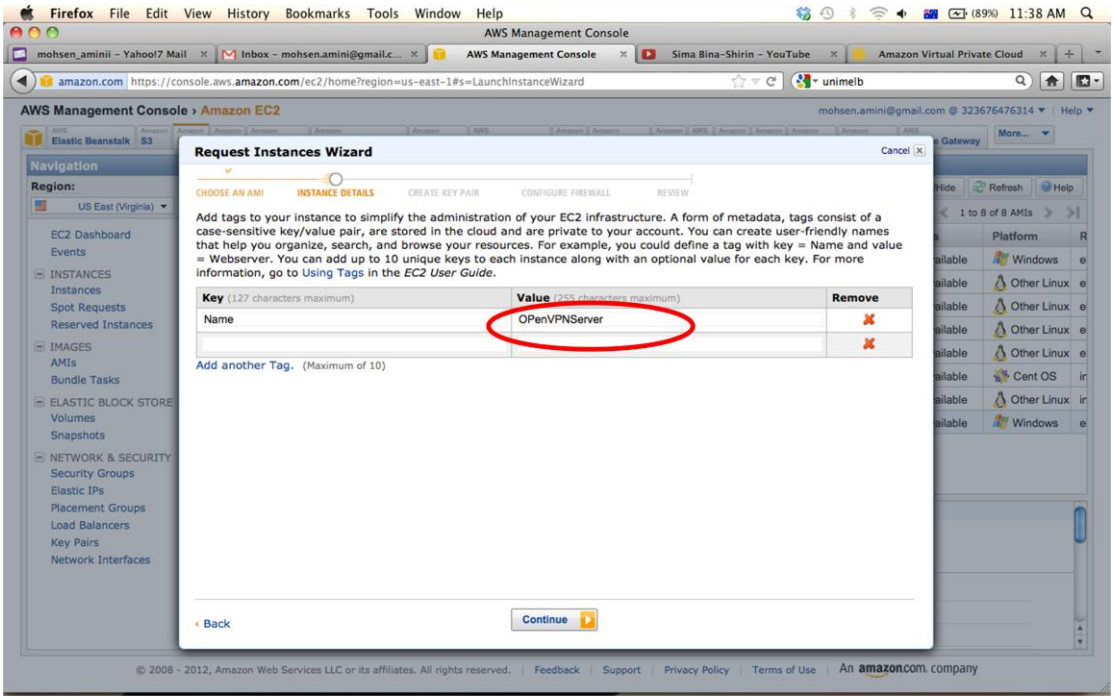
6. Launch the instance into VPC:



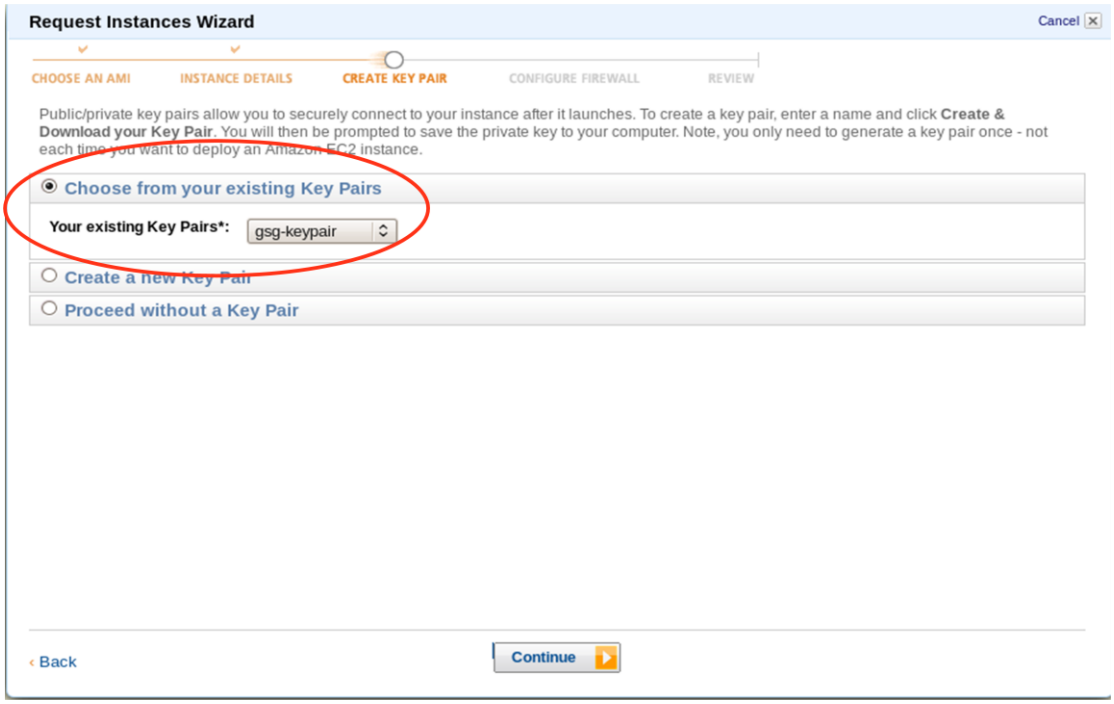
7. Assign IP address to the instance



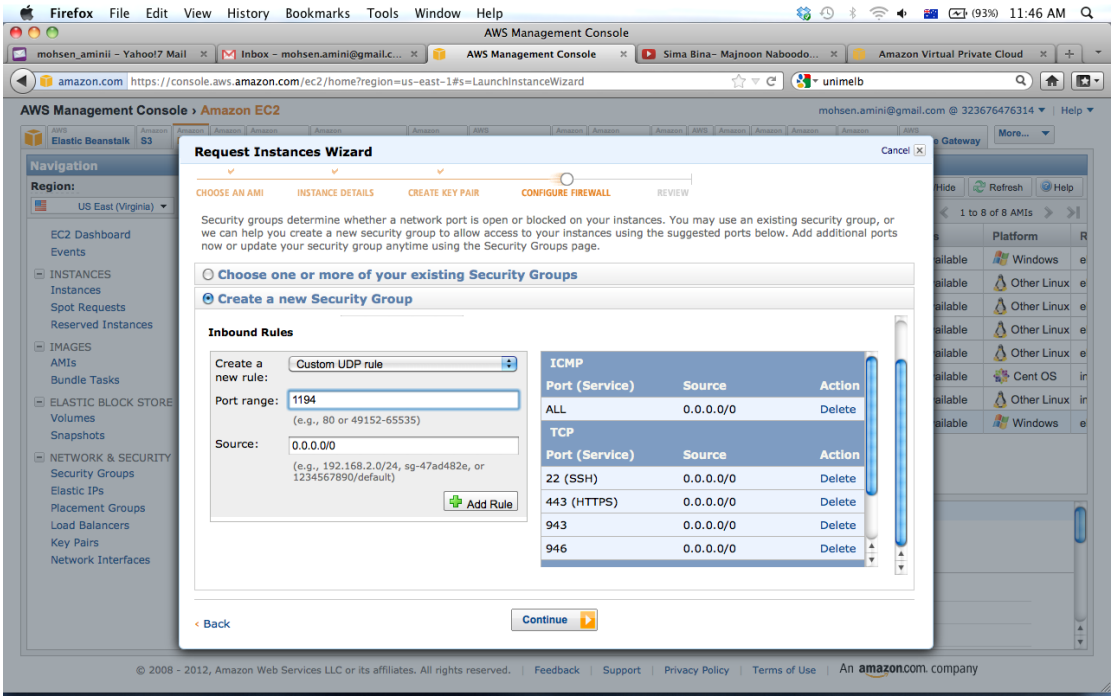
8. Give name to the instance



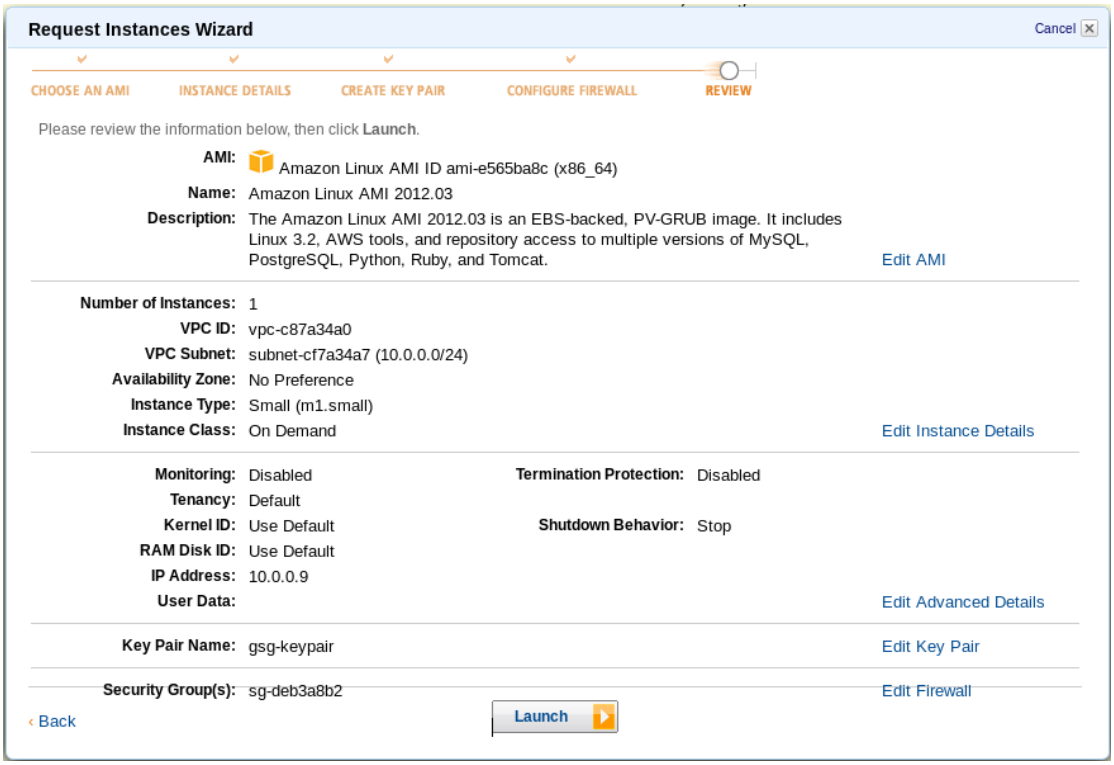
9. Assign Key pair



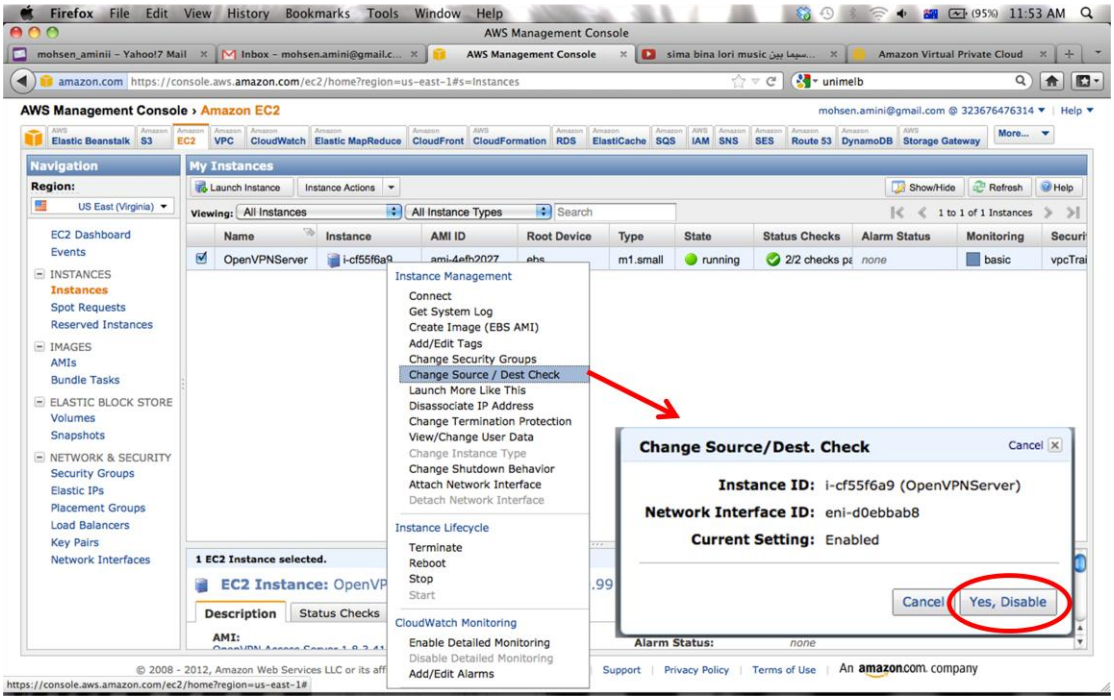
10. Create a new security group with the rules mentioned in the following picture:



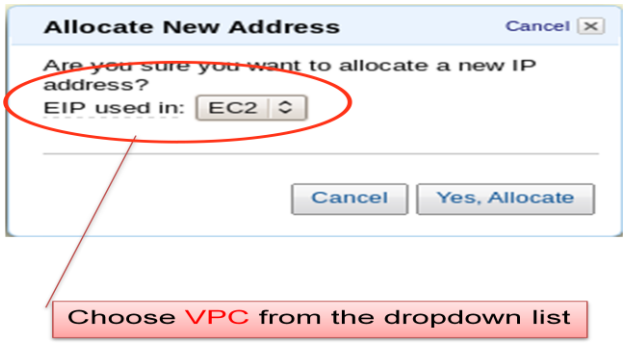
11. Review the setting and lunch the instance



12. Disable checking the source for instance

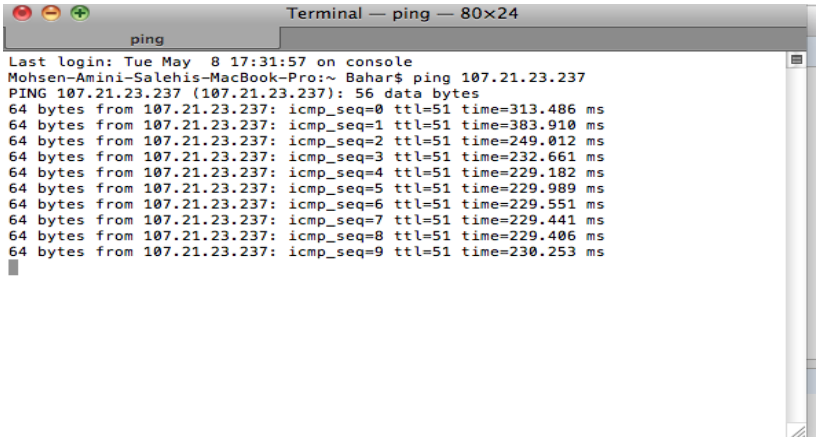


13. Create a new elastic IP address in VPC



14. Associate the IP address to the instance

15. Check if the IP is allocated and is available (Use ping)



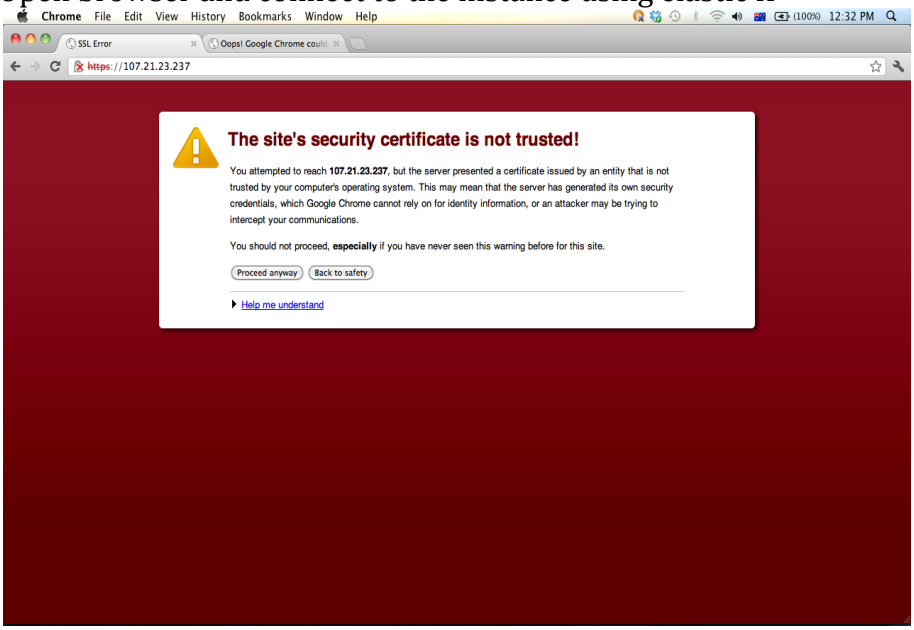
16. SSH to the created instance as root

    ssh –I YOURKEYFILE root@ELASTICIP

17. Set the instance to configure by default (press enter to save default).
18. There is a user called openvpn. Define password for openvpn user.

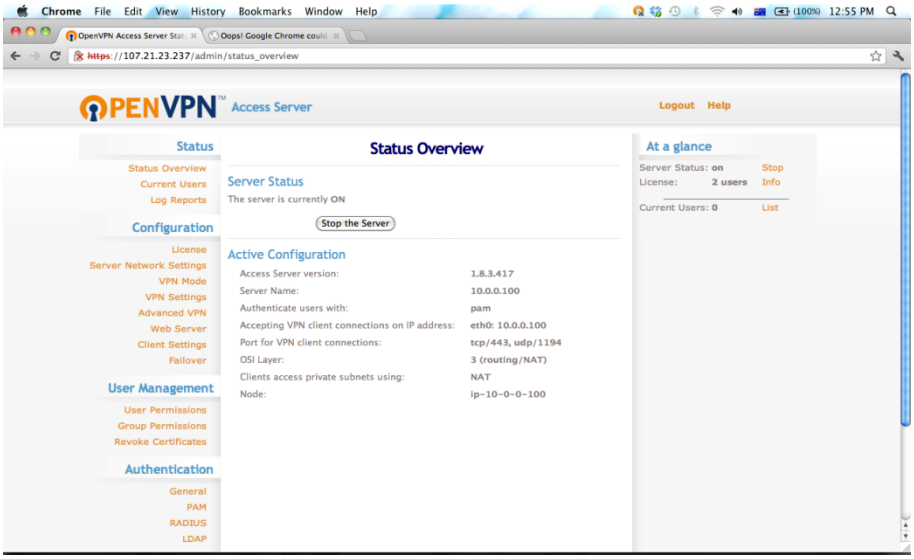
    Passwd openvpn

19. Key in username and password in the OpenVPN web client.
20. Download and Install OpenVPN
21. Open browser and connect to the instance using elastic IP

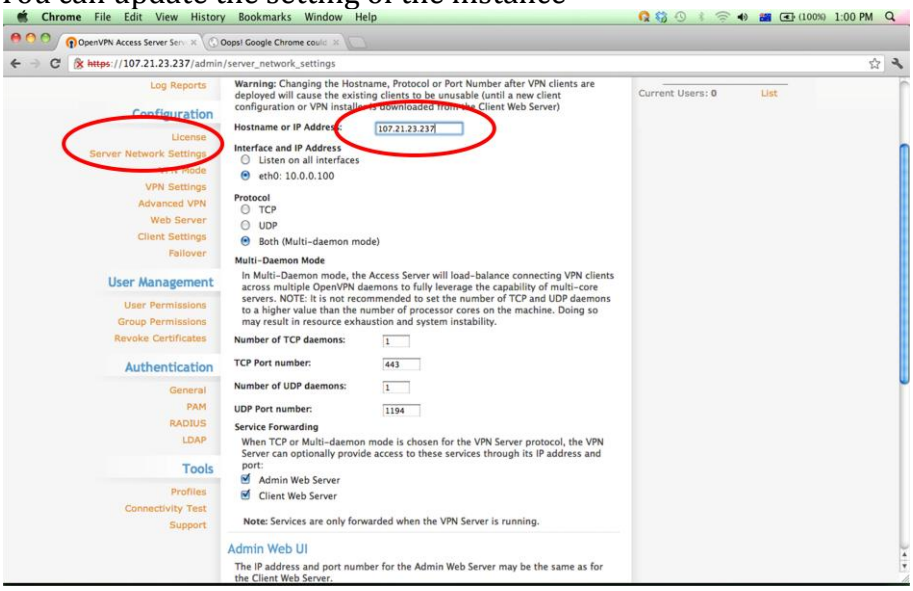


22. Connection based on HTTPS is created and you are asked for the credential of the instance (VPN server).

23. VPN server setting



24. You can update the setting of the instance



25. Save the changes to commit the setting


Chrome File Edit View History Bookmarks Window Help
https://107.21.23.237/admin/server_network_settings
OPENVPN Access Server
Logout Help
Status
Status Overview
Current Users
Log Reports
Configuration
License
Server Network Settings
VPN Mode
VPN Settings
Advanced VPN
Web Server
Client Settings
Failover
User Management
User Permissions
Group Permissions
Revoke Certificates
Authentication
General
PAM
RADIUS
LDAP
Settings Changed
The active profile 'Default' has been modified and saved.
Press the button below to propagate the changes to the running server.
Update Running Server
Server Network Settings
VPN Server
Warning: Changing the Hostname, Protocol or Port Number after VPN clients are deployed will cause the existing clients to be unusable (until a new client configuration or VPN installer is downloaded from the Client Web Server)
Hostname or IP Address: 107.21.23.237
Interface and IP Address
Listen on all interfaces
eth0: 10.0.0.100
Protocol
TCP
UDP
Both (Multi-daemon mode)
Multi-Daemon Mode
In Multi-Daemon mode, the Access Server will load-balance connecting VPN clients across multiple OpenVPN daemons to fully leverage the capability of multi-core servers. NOTE: It is not recommended to set the number of TCP and UDP daemons to a higher value than the number of processor cores on the machine. Doing so may result in resource exhaustion and system instability.
Number of TCP daemons: 1
At a glance
Server Status: on Stop
License: 2 users Info
Current Users: 0 List