

Handed out: 10/31/2015

Due by 11:59 PM on Friday, 11/06/2015

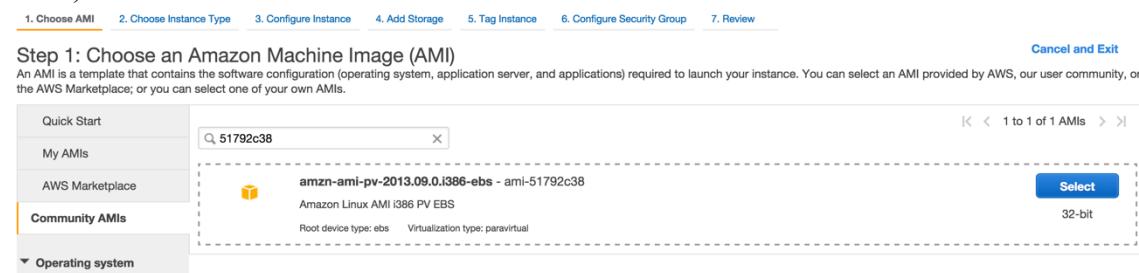
Place all of your narratives and illustrations in a single Word or PDF document named E90_LastNameFirstNameHW09.docx [.pdf]. Use this assignment as the initial template. Please implement code solutions as a separate class in one (Java, C#, Ruby, Python,...) project. Add your steps and your code below problem statements used for that problem. Upload your homework file and your working code (e.g., filename.java) into your Assignment 9 folder. Do not include executables. Please also do not zip your files.

Problem 1:

Start with **Amazon Linux AMI 2013.09 - ami-51792c38** (32-bit). You can actually start with any Linux or Windows AMI, if you feel comfortable with that AMI. If there is a need, install Apache Web server and PHP framework. Again, you are welcome to work with any other technology. Add either `index.php` I used in class or some other active page that would demonstrate that a request from your browser ended up on a particular server (AWS instance). If you are using `index.php`, remove the greeting. Leave only the IP address as the response produced by that file. Create a new AMI based on this instance. Create 3 (three) instances based on your new AMI. Out of those 3, place one instance in the same availability zone as the original instance and the remaining two in another availability zone. When creating new instances, please make sure that you check “Enable Cloud Watch detailed monitoring”. Create a load balancer and associate the original and 3 new instances with that load balancer. When creating the load balancer as the URL used for health check does specify `/index.php` or some other page you have added to your server. Demonstrate that the load balancer distributes incoming requests to different instances with approximately equal frequency. You could for example use just enabled Cloud Watch monitoring to show that after a moderately large number of requests sent to the `index.php` or a similar page, the usage pattern on all servers looks similar. You can do it all using AWS Console.

Points: [35]

- 1) Select AMI-51792c38 and create a micro instance.



Remember to configure the security group. Enable port 22 (SSH) and 80 (HTTP).

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

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description	Actions
sg-817073e5	default	default VPC security group	Copy to new
sg-85aaa9e1	ElasticMapReduce-master	Master group for Elastic MapReduce created on 2015-05-31T20:59:13.398Z	Copy to new
sg-80aaa9e4	ElasticMapReduce-slave	Slave group for Elastic MapReduce created on 2015-05-31T20:59:13.628Z	Copy to new
sg-adff3ca	launch-hqiu	security group for development environment in EC2	Copy to new
sg-71795716	launch-wizard-1	launch-wizard-1 created 2015-09-08T16:22:34.034-04:00	Copy to new

Inbound rules for sg-adff3ca (Selected security groups: sg-adff3ca)

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
Custom TCP Rule	TCP	8080	0.0.0.0/0
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

[Cancel](#) [Previous](#) [Review and Launch](#)

[Launch Instance](#) [Connect](#) [Actions](#)

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
i-539f11e4	i-539f11e4	t1.micro	us-east-1b	running	2/2 checks ...	None	ec2-54-172-44-217.co...	54.172.44.217

Instance: i-539f11e4 Public DNS: ec2-54-172-44-217.compute-1.amazonaws.com

Description	Status Checks	Monitoring	Tags
<p>Instance ID: i-539f11e4 Instance state: running Instance type: t1.micro Private DNS: ip-172-31-8-93.ec2.internal Private IPs: 172.31.8.93 Secondary private IPs: VPC ID: vpc-dfb48aba Subnet ID: subnet-66024a11 Network interfaces: eth0 Source/dest. check: True</p>	<p>Public DNS: ec2-54-172-44-217.compute-1.amazonaws.com Public IP: 54.172.44.217</p> <p>Elastic IP: -</p> <p>Availability zone: us-east-1b Security groups: launch-hqiu. view rules</p> <p>Scheduled events: No scheduled events</p> <p>AMI ID: amzn-ami-pv-2013.09.0.i386-ebs (ami-51792c38)</p> <p>Platform: - IAM role: -</p> <p>Key pair name: ec2hqiu</p>		

2) Connect to the instance, install Apache server software and PHP framework.

```
hqiu@bos-mpdei:> ssh -i "ec2hqiu.pem" ec2-user@54.172.44.217
The authenticity of host '54.172.44.217 (54.172.44.217)' can't be established.
RSA key fingerprint is ce:7e:a8:24:6b:e7:fd:89:00:43:44:0d:41:af:fb:f2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.172.44.217' (RSA) to the list of known hosts.

      _\ _ ) 
     _\ (   /  Amazon Linux AMI
     ___\_\_|_|
```

<https://aws.amazon.com/amazon-linux-ami/2013.09-release-notes/>
 Amazon Linux version 2015.09 is available.
[ec2-user@ip-172-31-8-93 ~]\$
[ec2-user@ip-172-31-8-93 ~]\$ [sudo yum update](#)
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main/latest
amzn-updates/latest
Resolving Dependencies
--> Running transaction check
---> Package PyYAML.i686 0:3.10-3.6.amzn1 will be obsoleted
---> Package acl.i686 0:2.2.49-6.9.amzn1 will be updated
---> Package acl.i686 0:2.2.49-6.11.amzn1 will be an update

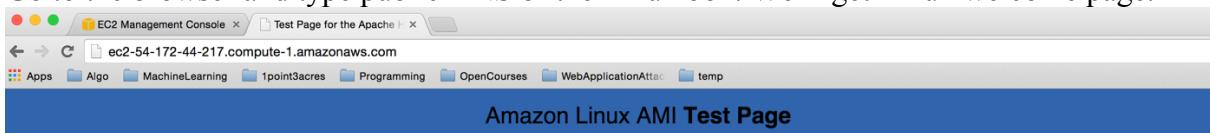
```
[ec2-user@ip-172-31-8-93 ~]$ sudo yum install httpd
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
--> Package httpd.i686 0:2.2.31-1.6.amzn1 will be installed
--> Processing Dependency: httpd-tools = 2.2.31-1.6.amzn1 for package: httpd-2.2.31-1.6.amzn1.i686
--> Processing Dependency: libaprutil-1.so.0 for package: httpd-2.2.31-1.6.amzn1.i686
--> Processing Dependency: libapr-1.so.0 for package: httpd-2.2.31-1.6.amzn1.i686
--> Processing Dependency: apr-util-ldap for package: httpd-2.2.31-1.6.amzn1.i686
--> Processing Dependency: system-logos for package: httpd-2.2.31-1.6.amzn1.i686
--> Running transaction check
--> Package apr.i686 0:1.5.0-2.11.amzn1 will be installed
--> Package apr-util.i686 0:1.4.1-4.17.amzn1 will be installed
--> Package apr-util-ldap.i686 0:1.4.1-4.17.amzn1 will be installed
--> Package generic-logos.noarch 0:17.0.0-2.5.amzn1 will be installed
--> Package httpd-tools.i686 0:2.2.31-1.6.amzn1 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
[ec2-user@ip-172-31-8-93 ~]$ sudo /etc/init.d/httpd start
Starting httpd: [ OK ]
[ec2-user@ip-172-31-8-93 ~]$ ps -ef | grep httpd
root      6550      1  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6552  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6553  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6554  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6555  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6556  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6557  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6558  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
apache    6559  6550  0 23:16 ?        00:00:00 /usr/sbin/httpd
ec2-user   6561 1536  0 23:16 pts/0    00:00:00 grep httpd
```

```
[ec2-user@ip-172-31-8-93 ~]$ sudo yum install php
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main/latest
amzn-updates/latest
Resolving Dependencies
--> Running transaction check
--> Package php.i686 0:5.3.29-1.8.amzn1 will be installed
--> Processing Dependency: php-common(x86-32) = 5.3.29-1.8.amzn1 for package: php-5.3.29-1.8.amzn1.i686
--> Processing Dependency: php-cli(x86-32) = 5.3.29-1.8.amzn1 for package: php-5.3.29-1.8.amzn1.i686
--> Processing Dependency: libgmp.so.3 for package: php-5.3.29-1.8.amzn1.i686
--> Running transaction check
--> Package compat-gmp4.i686 0:4.3.2-1.14.amzn1 will be installed
--> Package php-cli.i686 0:5.3.29-1.8.amzn1 will be installed
--> Package php-common.i686 0:5.3.29-1.8.amzn1 will be installed
--> Finished Dependency Resolution
```

Go to the browser and type public DNS of the Linux box. We'll get Linux welcome page.



If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

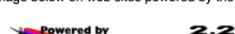
For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

For information on Amazon Linux AMI , please visit the [Amazon AWS website](#).

If you are the website administrator:

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the image below on web sites powered by the Apache HTTP Server:



3) Create “index.php” file. Record the IP address as the response in the file.

```
[ec2-user@ip-172-31-8-93 ~]$ cd /var/www/html/  
[ec2-user@ip-172-31-8-93 html]$ sudo vi index.php  
[ec2-user@ip-172-31-8-93 html]$  
[ec2-user@ip-172-31-8-93 html]$ sudo vi /etc/httpd/conf.d/welcome.conf
```

A screenshot of a terminal window titled "AWS - ec2-user@ip-172-31-8-93:/var/www/html - ssh - 113x29". The command "echo \\$_SERVER['SERVER_ADDR']; ?>" has been typed into the terminal. The output shows the IP address 172.31.8.93.

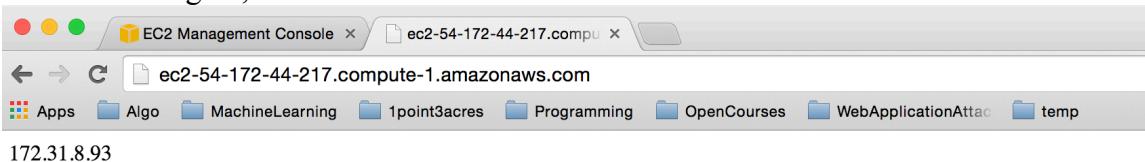
Comment out the content in file “welcome.conf”.

```
[ec2-user@ip-172-31-8-93 ~]$ cd /etc/httpd/conf.d/  
[ec2-user@ip-172-31-8-93 conf.d]$ sudo vi welcome.conf  
#  
# This configuration file enables the default "Welcome"  
# page if there is no default index page present for  
# the root URL. To disable the Welcome page, comment  
# out all the lines below.  
#  
#<LocationMatch "^/+$">  
#   Options -Indexes  
#   ErrorDocument 403 /error/noindex.html  
#</LocationMatch>
```

Restart the Apache service.

```
[ec2-user@ip-172-31-8-93 html]$ sudo /etc/init.d/httpd stop  
Stopping httpd: [ OK ]  
[ec2-user@ip-172-31-8-93 html]$ sudo /etc/init.d/httpd start  
Starting httpd: [ OK ]
```

Visit the site again, the server shows the internal IP address of the box.

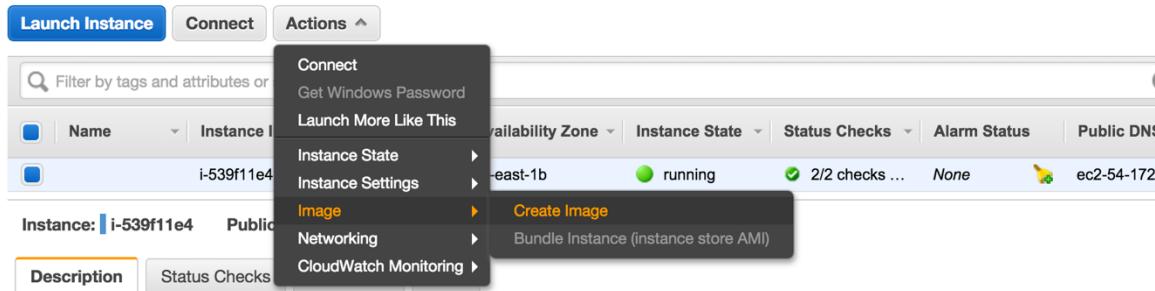


4) Make sure the Apache will start at the boot time.

```
[ec2-user@ip-172-31-8-93 html]$ sudo vi /etc/rc.local
```

```
#!/bin/sh  
#  
# This script will be executed *after* all the other init scripts.  
# You can put your own initialization stuff in here if you don't  
# want to do the full Sys V style init stuff.  
  
touch /var/lock/subsys/local  
/etc/init.d/httpd start
```

5) Go to EC2 dashboard, create an image/a new AMI based on this instance.



Create Image

Instance ID: i-539f11e4

Image name: LB Instance

Image description: Amazon Linux with Apache

No reboot:

Instance Volumes

Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Delete on Termination	Encrypted
Root	/dev/sda1	snap-7962c379	8	General Purpose (SSD)	24 / 3000	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Total size of EBS Volumes: 8 GiB
When you create an EBS image, an EBS snapshot will also be created for each of the above volumes.

Buttons: Cancel, Create Image

- 6) Once the new AMI is available, create 3 instances based on this AMI. Place one of the instance in the same availability zone as the original instance and the remaining two in another availability zone. Make sure that check “Enable Cloud Watch detailed monitoring”.

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

Step 1: Choose an Amazon Machine Image (AMI)

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

Cancel and Exit

Quick Start	Search my AMIs	1 to 1 of 1 AMIs
My AMIs	LB Instance - ami-03aed269	Select
AWS Marketplace	Amazon Linux with Apache	32-bit
Community AMIs	Root device type: ebs Virtualization type: paravirtual Owner: 217134905396	

Step 3: Configure Instance Details

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

Number of instances	<input type="text" value="1"/>	Launch into Auto Scaling Group i
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-dfb48aba (172.31.0.0/16) (default)	Create new VPC
Subnet	subnet-66024a11(172.31.0.0/20) Default in us-east-1b 4090 IP Addresses available	Create new subnet
Auto-assign Public IP	<input type="checkbox"/> Use subnet setting (Enable)	
IAM role	None	Create new IAM role
Shutdown behavior	Stop	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input checked="" type="checkbox"/> Enable CloudWatch detailed monitoring <small>Additional charges apply.</small>	
Tenancy	Shared tenancy (multi-tenant hardware)	<small>Additional charges will apply for dedicated tenancy.</small>

Step 3: Configure Instance Details

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

Number of instances	<input type="text" value="2"/>	Launch into Auto Scaling Group i
<p> You may want to consider launching these instances into an Auto Scaling Group to help you maintain availability in the future. Learn how Auto Scaling can help your application stay healthy and cost effective.</p>		
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-dfb48aba (172.31.0.0/16) (default)	Create new VPC
Subnet	subnet-85a0cddc(172.31.16.0/20) Default in us-east-1c 4091 IP Addresses available	Create new subnet
Auto-assign Public IP	<input type="checkbox"/> Use subnet setting (Enable)	
IAM role	None	Create new IAM role
Shutdown behavior	Stop	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input checked="" type="checkbox"/> Enable CloudWatch detailed monitoring <small>Additional charges apply.</small>	
Tenancy	Shared tenancy (multi-tenant hardware)	<small>Additional charges will apply for dedicated tenancy.</small>

Step 5: Tag Instance

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon resources.

Key <small>(127 characters maximum)</small>	Value <small>(255 characters maximum)</small>
Name	LB Instance
Create Tag <small>(Up to 10 tags maximum)</small>	

For the original instance, enable its “detailed monitoring” too.

CloudWatch metrics: Basic monitoring **Enable Detailed Monitoring**

CPU Utilization (Percent)

Disk Reads (Bytes)

Enable Detailed Monitoring

Enable detailed monitoring for your instance to get these metrics at 1-minute frequency. [Learn more](#)

Are you sure you want to enable detailed monitoring for the following instances?
(Additional charges apply.)

i-539f11e4

Cancel Yes, Enable

Enable Detailed Monitoring

Detailed monitoring has been enabled.

Close

List all the available instances:

<input type="checkbox"/>	LB Instance	i-694bc5de	t1.micro	us-east-1b	● running	🕒 Initializing	None
<input type="checkbox"/>		i-539f11e4	t1.micro	us-east-1b	● running	🕒 Initializing	None
<input type="checkbox"/>	LB Instance	i-e2629852	t1.micro	us-east-1c	● running	🕒 Initializing	None
<input type="checkbox"/>	LB Instance	i-e3629853	t1.micro	us-east-1c	● running	🕒 Initializing	None

Instances: i-e2629852 (LB Instance), i-694bc5de (LB Instance), i-e3629853 (LB Instance), i-539f11e4

- 7) Create a load balancer and associate the original and 3 new instances with that load balancer.

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you have created. You can also configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured a web server on port 80.

Load Balancer name:	ApacheServerLoadBalancer		
Create LB Inside:	My Default VPC (172.31.0.0/16)		
Create an internal load balancer:	<input type="checkbox"/> (what's this?)		
Enable advanced VPC configuration:	<input type="checkbox"/>		
Listener Configuration:			
Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80
Add			

Step 2: Assign Security Groups

You have selected the option of having your Elastic Load Balancer inside of a VPC, which allows you to assign security groups to your load balancer. Please select the security group that you want to use. This selection can be changed at any time.

Assign a security group: Create a new security group
 Select an existing security group

Security Group ID	Name	Description
sg-817073e5	default	default VPC security group
sg-85aaa9e1	ElasticMapReduce-master	Master group for Elastic MapReduce created on 2015-05-31T20:59:13.398Z
sg-80aaa9e4	ElasticMapReduce-slave	Slave group for Elastic MapReduce created on 2015-05-31T20:59:13.628Z
<input checked="" type="checkbox"/> sg-adfff3ca	launch-hqiu	security group for development environment in EC2
sg-71795716	launch-wizard-1	launch-wizard-1 created 2015-09-08T16:22:34.034-04:00

When creating the load balancer, set the Ping Path to “/index.php”. Set the advanced details carefully.

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, the load balancer will remove it from service. Customize the health check to meet your specific needs.

Ping Protocol	HTTP
Ping Port	80
Ping Path	/index.php

Advanced Details

Response Timeout	<input type="text" value="5"/> seconds
Health Check Interval	<input type="text" value="30"/> seconds
Unhealthy Threshold	<input type="text" value="2"/>
Healthy Threshold	<input type="text" value="5"/>

Add the EC2 instances into the load balancer. We can see the availability zone is equally distributed.

Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC: vpc-dfb48aba (172.31.0.0/16)

Select	Instance	Name	State	Security Groups	Zone	Subnet ID	Subnet CIDR
<input checked="" type="checkbox"/>	i-694bc5de	LB Instance	running	launch-hqi	us-east-1b	subnet-66024a11	172.31.0.0/20
<input checked="" type="checkbox"/>	i-e2629852	LB Instance	running	launch-hqi	us-east-1c	subnet-85a0cddc	172.31.16.0/20
<input checked="" type="checkbox"/>	i-e3629853	LB Instance	running	launch-hqi	us-east-1c	subnet-85a0cddc	172.31.16.0/20
<input checked="" type="checkbox"/>	i-539f11e4		running	launch-hqi	us-east-1b	subnet-66024a11	172.31.0.0/20

Availability Zone Distribution

2 instances in us-east-1b

2 instances in us-east-1c

- Enable Cross-Zone Load Balancing (i)
- Enable Connection Draining (i) 300 seconds

Step 7: Review

Please review the load balancer details before continuing

Define Load Balancer

Load Balancer name: ApacheServerLoadBalancer
Scheme: internet-facing
Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)

Configure Health Check

Ping Target: HTTP:80/index.php
Timeout: 5 seconds
Interval: 30 seconds
Unhealthy Threshold: 2
Healthy Threshold: 5

Add EC2 Instances

Cross-Zone Load Balancing: Enabled
Connection Draining: Enabled, 300 seconds
Instances: i-694bc5de (LB Instance), i-e2629852 (LB Instance), i-e3629853 (LB Instance), i-539f11e4

VPC Information

VPC: vpc-dfb48aba
Subnets: subnet-abe07780, subnet-85a0cddc, subnet-e3417bd9, subnet-66024a11

Load Balancer Creation Status

Successfully created load balancer

Load balancer [ApacheServerLoadBalancer](#) was successfully created.

Note: It may take a few minutes for your instances to become active in the new load balancer.

- 8) When the load balancer is ready, check its DNS Name. The instances status is: 4 of 4 instances in service.

Filter: X

Load Balancer Name	DNS Name	Port Configuration	Availability Zones	Instance Count	Health Check
ApacheServerLoadBalancer	ApacheServerLoadBalancer-1308527199.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1a, us-east-1b, us-east-1c	4 Instances	HTTP:80/index.php

Load balancer: ApacheServerLoadBalancer

Description Instances Health Check Monitoring Security Listeners Tags

DNS Name: ApacheServerLoadBalancer-1308527199.us-east-1.elb.amazonaws.com (A Record)

Note: Because the set of IP addresses associated with a LoadBalancer can change over time, you should never create an "A" record with any specific IP address. If you want to use a friendly DNS name for your load balancer instead of the name generated by the Elastic Load Balancing service, you should create a CNAME record for the LoadBalancer DNS name, or use Amazon Route 53 to create a hosted zone. For more information, see [Using Domain Names With Elastic Load Balancing](#).

Scheme: internet-facing

Status: 4 of 4 instances in service

Port Configuration: 80 (HTTP) forwarding to 80 (HTTP)
Stickiness: Disabled ([Edit](#))

Availability Zones: subnet-66024a11 - us-east-1b, subnet-85a0ccdc - us-east-1c, subnet-abe07780 - us-east-1a, subnet-e3417bd9 - us-east-1e

Cross-Zone Load Balancing: Enabled ([Edit](#))

We can also edit the parameters of the Health Check here.

Filter: X

Load Balancer Name	DNS Name	Port Configuration	Availability Zones
ApacheServerLoadBalancer	ApacheServerLoadBalancer-1308527199.us-east-1.elb.amazonaws.com	80 (HTTP) forwarding to 80 (HTTP)	us-east-1a, us-east-1b, us-east-1c

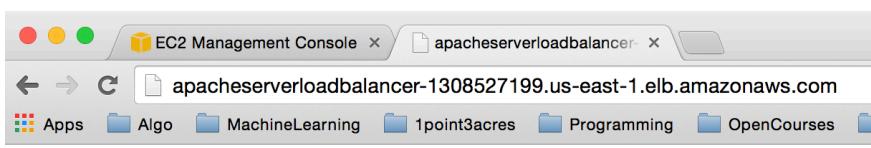
Load balancer: ApacheServerLoadBalancer

Description Instances Health Check Monitoring Security Listeners Tags

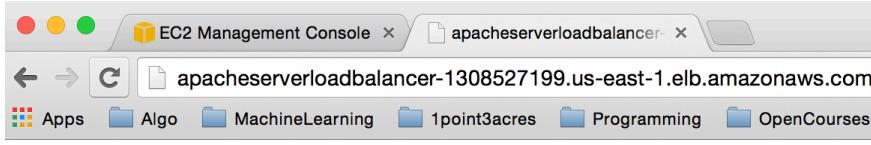
Ping Target: HTTP:80/index.php
Timeout: 5 seconds
Interval: 30 seconds
Unhealthy Threshold: 2
Healthy Threshold: 5

[Edit Health Check](#)

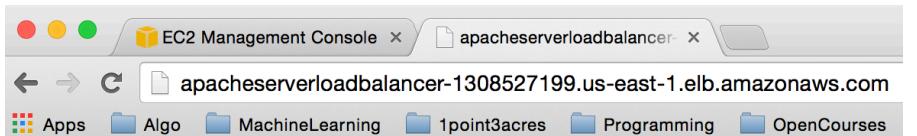
- 9) Test the load balancer to see if it distributes the incoming requests to different instances with approximately equal frequency. Open the browser and type in the DNS name of the load balancer. Refresh the page several times.



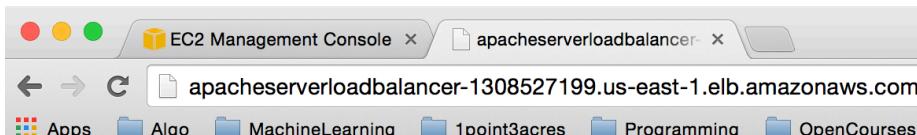
172.31.24.53



172.31.24.52

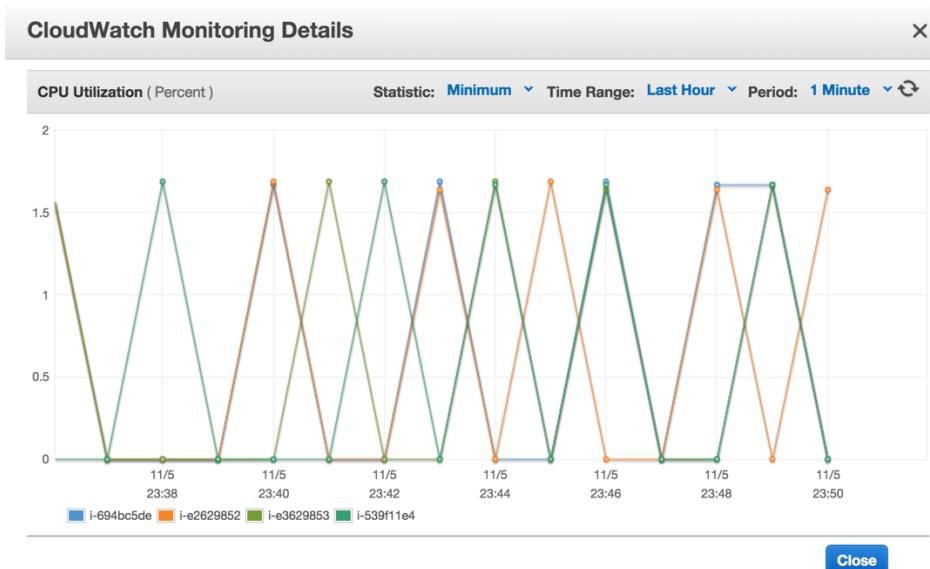


172.31.2.238



172.31.8.93

The load balance sends requests to all of my four instances, and it sends to a different server every time. The figure below shows the CloudWatch Monitoring Details. We can see that the 4 instances have similar/identical CPU utilization, which means they server requests equally.



Problem 2:

Use techniques we relied in the RESTful Web Service client class

`CustomerResourceClient.java` to build an automated testing tool for verifying performance of the Load Balancer. That testing tool should send Http GET requests to the Load Balancer and keep track of the responses. Hardcode the number of pings you are sending (100-1000). In the first iteration print responses coming from the Load Balancer (your application producing IP addresses). In the second iteration count how many times you received each IP address. Work with 3 instances behind the load balancer.

Points: [30]

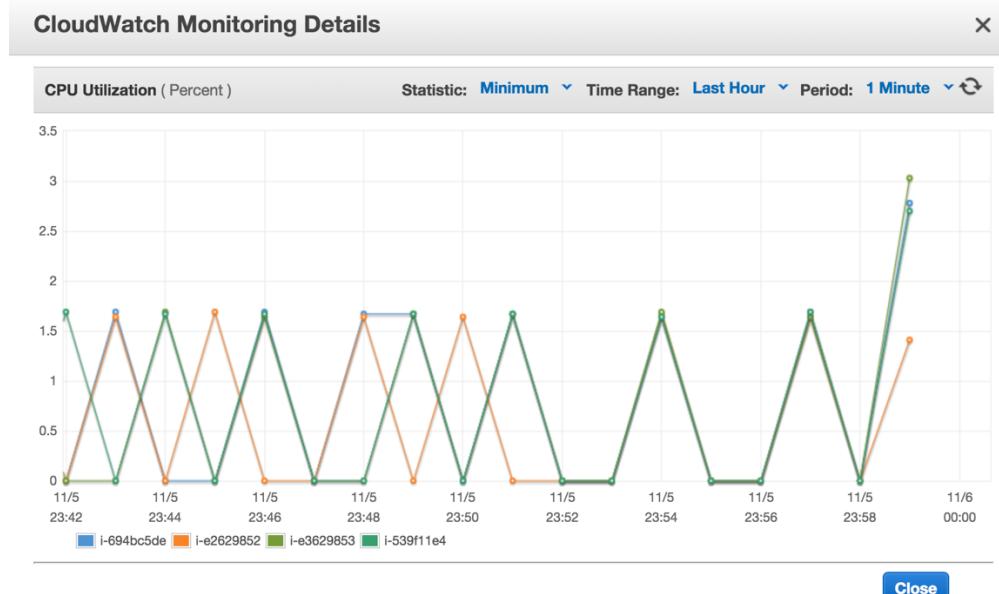
- 1) Create a new project called “LoadBalanceClient”, re-use the RESTful web service client class to sent HTTP GET requests to the load balancer automatically and keep track of the responses. First work with 4 instances.

The file is called “LoadBalancerClient.java”.

```
9 public class LoadBalancerClient {  
10    public static void main(String[] args) {  
11        HashMap<String, Integer> counter = new HashMap<String, Integer>();  
12  
13        try {  
14            for (int i = 0; i < 1000; i++) {  
15                System.out.println("\n*** Send HTTP GET Request ***");  
16                URL getUrl = new URL("http://apacheserverloadbalancer-1308527199.us-east-1.elb.amazonaws.com/");  
17                HttpURLConnection connection = (HttpURLConnection) getUrl.openConnection();  
18                connection.setRequestMethod("GET");  
19                System.out.println("Content-Type: " + connection.getContentType());  
20  
21                BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
22  
23                String line = reader.readLine();  
24                while (line != null) {  
25                    System.out.println(line);  
26                    // Record the times received for each IP address  
27                    if (counter.containsKey(line)) {  
28                        counter.put(line, counter.get(line) + 1);  
29                    } else {  
30                        counter.put(line, 1);  
31                    }  
32                    line = reader.readLine();  
33                }  
34  
35                connection.disconnect();  
36            }  
37  
38            System.out.println("\nThe times to print each IP address are:\n");  
39            for (String ip : counter.keySet()) {  
40                System.out.println("IP address: " + ip + ", times: " + counter.get(ip));  
41            }  
42  
43        } catch (Exception e) {  
44            e.printStackTrace();  
45        }  
46    }  
47}  
48 }
```

After sending the HTTP GET requests, analyze the responses. Use a Hashmap to count the number of requests received from each IP address.

The results after sending about 5000 requests:



There's a spike for each box.

Results from Eclipse console:

```

Problems @ Javadoc Declaration Console 
<terminated> LoadBalancerClient [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Hor

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.24.52

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.24.53

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.8.93

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.2.238

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.24.52

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.24.53

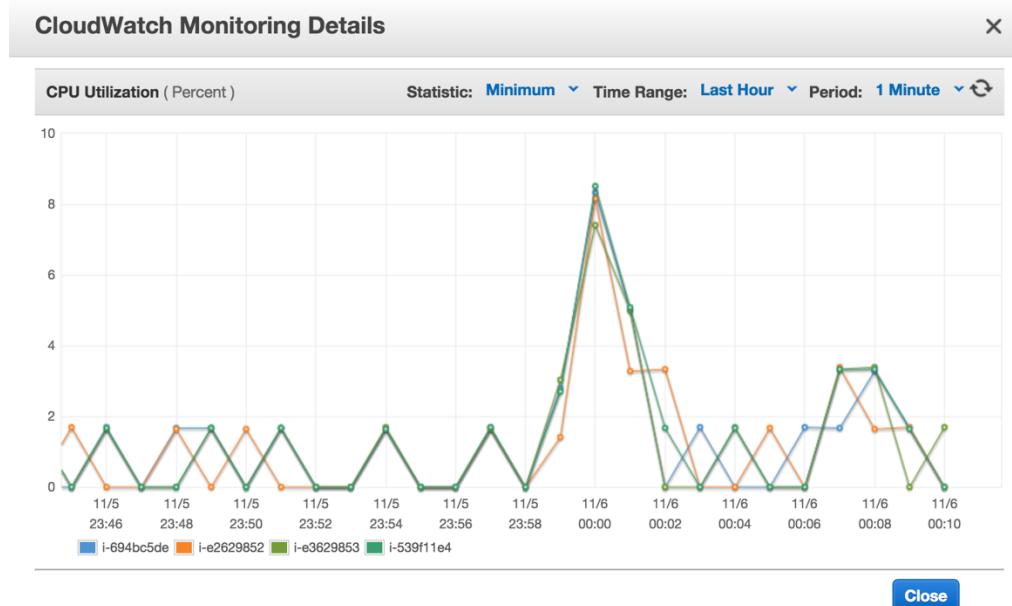
*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.8.93

The times to print each IP address are:
IP address: 172.31.24.53, times: 250
IP address: 172.31.8.93, times: 250
IP address: 172.31.2.238, times: 250
IP address: 172.31.24.52, times: 250

```

We can see that for 1000 requests, we will receive 250 requests from each IP address. The load balancer distributes the incoming requests approximately equal frequency.

The second spike shows performance after sending 1000 requests.



- 2) Repeat the test using 3 instances. Since I terminated the instances after the previous tests. I created 3 new instances here using the same AMI. The code is still the same.

The 3 new instances here:

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
<input type="checkbox"/>	LB Instance	i-9d93052a	t1.micro	us-east-1b	green running	green 2/2 checks ...	None
<input type="checkbox"/>	LB Instance	i-2c45849c	t1.micro	us-east-1c	green running	green 2/2 checks ...	None
<input type="checkbox"/>	LB Instance	i-544584e4	t1.micro	us-east-1c	green running	green 2/2 checks ...	None

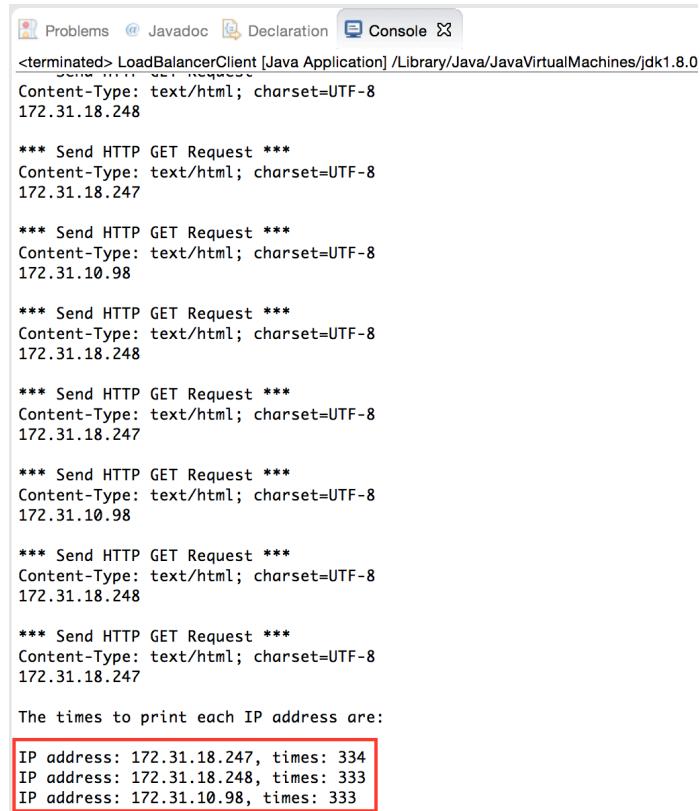
Edit the instances in the load balancer:

The screenshot shows the AWS EC2 Dashboard with the 'Load Balancer' section selected. Under 'Load balancer: ApacheServerLoadBalancer', the 'Instances' tab is active. A red box highlights the 'Edit Instances' button. Below it, the 'Add and Remove Instances' section lists three instances: i-9d93052a, i-2c45849c, and i-544584e4, all marked as 'running'. A red box highlights the 'Availability Zone Distribution' section, which states '1 instance in us-east-1b' and '2 instances in us-east-1c'.

After the edit:

The screenshot shows the AWS EC2 Dashboard with the 'Load Balancer' section selected. Under 'Load balancer: ApacheServerLoadBalancer', the 'Instances' tab is active. A red box highlights the 'Edit Instances' button. Below it, the 'Instances' table shows the three instances now listed under 'Status' as 'InService'. The 'Actions' column for each instance has a 'Remove from Load Balancer' link.

The results for 3 instances. The requests are still equally distributed although they are not in the same availability zone.



```
Problems @ Javadoc Declaration Console 
<terminated> LoadBalancerClient [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0
Content-Type: text/html; charset=UTF-8
172.31.18.248

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.18.247

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.10.98

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.18.248

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.18.247

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.10.98

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.18.248

*** Send HTTP GET Request ***
Content-Type: text/html; charset=UTF-8
172.31.18.247

The times to print each IP address are:
IP address: 172.31.18.247, times: 333
IP address: 172.31.18.248, times: 333
IP address: 172.31.10.98, times: 333
```

Problem 3:

Code provided for the following example is written in Java. You are most welcome to replace all three classes, two are really essential, with similar applications written in any language of your choice. If you are working with Java as the Cloud machine use the machine on which you installed Java 8.

In the attached: hwk9_edu.zip archive you will find classes

ServerSideConsumer.java, ClientSideProducer.java and ClientSideQMonitor.java used in class. Create an AWS Java project based on SQSSample application. Remove the class that comes with that project. Copy provided classes into the new project. Make sure that you configure your project with your AWS credentials. Make sure that you change compliance level of your Eclipse project to Java 1.5.

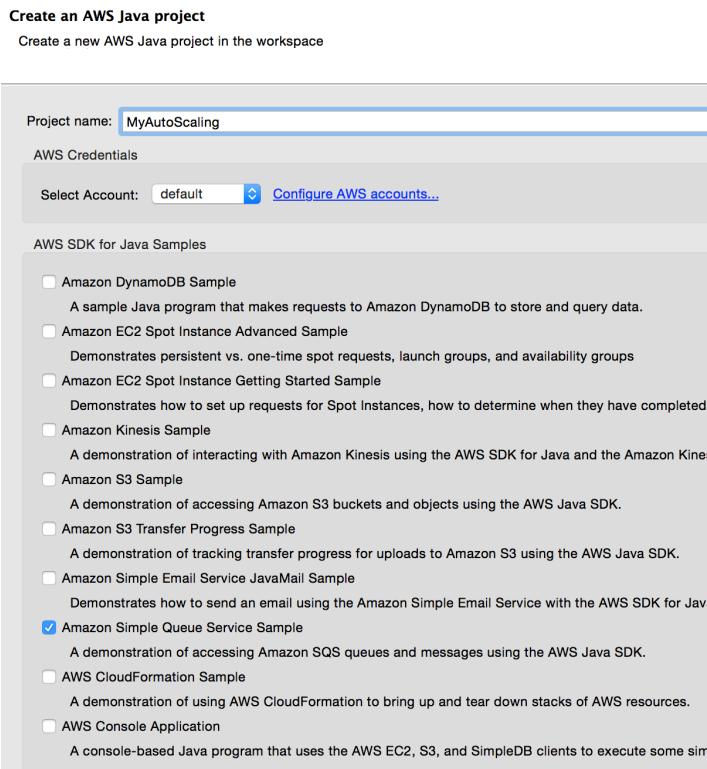
Create an SQS Queue and enter its URL in the appropriate places in those classes. Test your classes in Eclipse and make sure they can communicate with your SQS Queue. You will next create three executable jars, each with one of the above classes as the Main class. Follow the procedure used in class. Test those jars again on the operating system prompt. You noticed that my classes have hard coded wait times which determine the frequency with which producer sends messages to the queue and the frequency with which the consumer retrieves messages from the queue.

If you know how, modify those classes so that you can pass those wait times to the main classes at the run time as command line parameters. If you do not know how to do that, do not bother. Secure copy (`scp`) the executable jar with `ServerSideConsumer` class to the running instance in the Cloud. Start client side producer and make sure that the server side consumer truly consumes messages from the queue. Open AWS Console for SQS service and verify that messages are placed in queue and subsequently consumed. Modify your Cloud instance so that that the server side jar is started on every reboot or startup of that instance. Test your arrangement by stopping or rebooting the instance. Once the instance is restarted, right click on the instance in EC2 Console (My Instances page) and select `Get System Log`. If your jar works, at the bottom of the log, you will see its output confirming that messages are indeed read.

Points: [35]

ONCE YOU ARE DONE, PLEASE REMOVE ALL INSTANCES AND AMI-s YOU DO NOT NEED ANY MORE and other objects you created.

- 1) Create an AWS Java project based on the SQSSample application.



Remove the class that comes with the project. Copy the classes from HW 9 zip file.

- 2) Create an SQS queue. Copy the URL of the queue into our classes.

Create New Queue

Please enter a name for your new queue. Queue names must be 1-80 characters in length and be composed of alphanumeric characters, hyphens (-), and underscores (_). Your queue will be created in the US East (N. Virginia) region.

Region: US East (N. Virginia)

Queue Name: MyQueue

Configure your new queue by setting queue attributes (optional).

Queue Settings

Default Visibility Timeout: 20 seconds Value must be between 0 seconds and 12 hours.

Message Retention Period: 4 days Value must be between 1 minute and 14 days.

Maximum Message Size: 256 KB Value must be between 1 and 256 KB.

Delivery Delay: 0 seconds Value must be between 0 seconds and 15 minutes.

Receive Message Wait Time: 0 seconds Value must be between 0 and 20 seconds.

Dead Letter Queue Settings

Use Redrive Policy:

Dead Letter Queue: Value must be an existing queue name.

Maximum Receives: Value must be between 1 and 1000.

Cancel **Create Queue**

Queues			
Create New Queue Queue Actions ▼			
Filter by Prefix: <input type="text"/>			
Name		Messages Available	Messages in Flight
<input checked="" type="checkbox"/> MyQueue		0	0

1 SQS Queue selected.

Details **Permissions** **Redrive Policy**

Name: MyQueue	Default Visibility Timeout: 20 seconds
URL: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue	Message Retention Period: 4 days
ARN: arn:aws:sqs:us-east-1:217134905396:MyQueue	Maximum Message Size: 256 KB

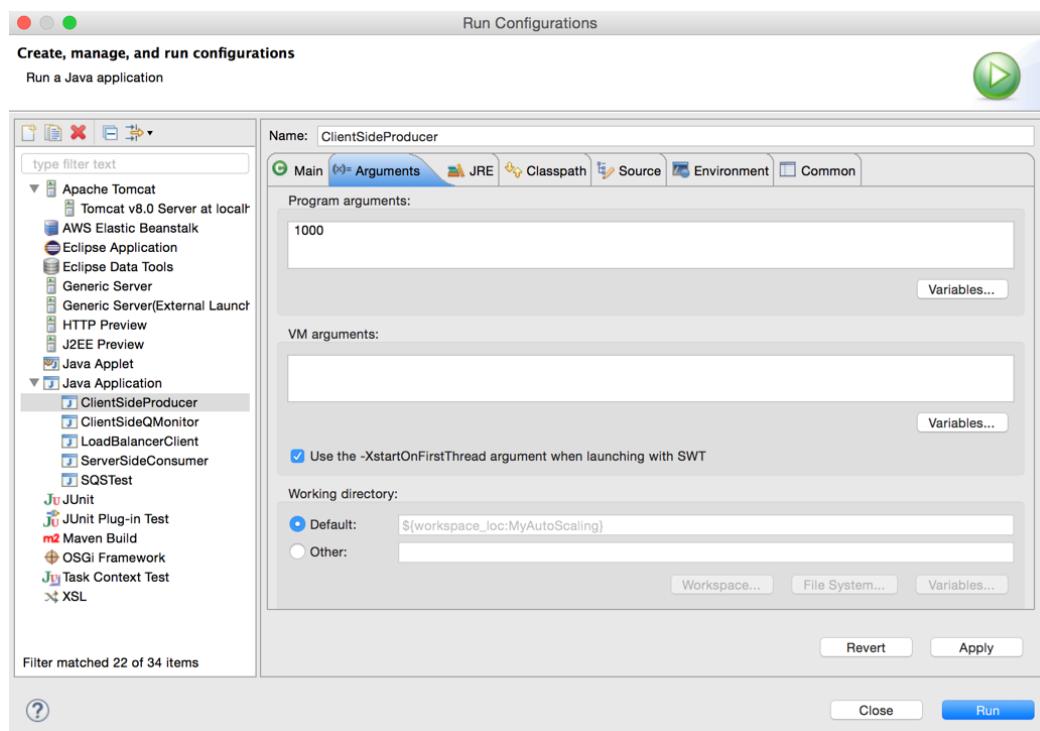
- 3) ClientSideProducer: (I hide my AWS credentials here). Send messages to the SQS queue continuously. The wait time can be passed in or use the hard code one.

```

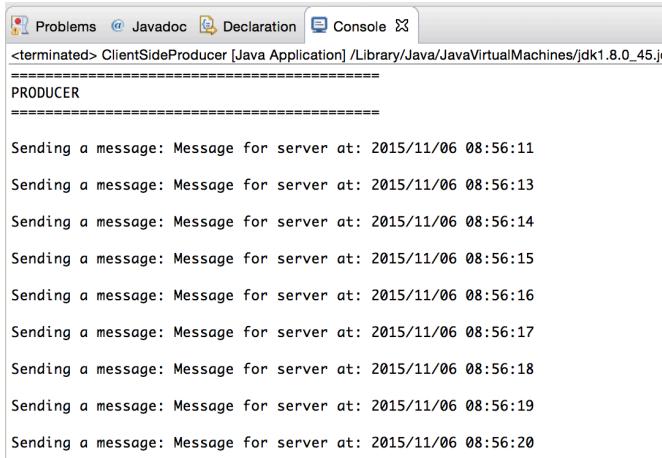
15  /**
16   * sends a message to user queue (see below in code!) every WAIT_MSG_ms millisecond
17   */
18  public class ClientSideProducer {
19      static final int WAIT_MSG_ms = 2000;
20
21  public static void main(String[] args) throws Exception {
22
23     /* replace with your access key and secret key */
24     AmazonSQS sqs = new AmazonSQSClient(new BasicAWSCredentials("Access key", "Secret access key"));
25
26     Region usEast1 = Region.getRegion(Regions.US_EAST_1);
27     sqs.setRegion(usEast1);
28
29     int wait_ms = args.length > 0 ? Integer.valueOf(args[0]).intValue() : WAIT_MSG_ms;
30
31     System.out.println("=====");
32     System.out.println("PRODUCER");
33     System.out.println("=====\n");
34
35     try {
36         while (true) {
37             String msg = "Message for server at: "
38                     + (new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(new Date()));
39             System.out.println("Sending a message: " + msg + "\n");
40             /* replace with your sqs queue URL */
41             sqs.sendMessage(new SendMessageRequest("https://sns.us-east-1.amazonaws.com/217134905396/MyQueue", msg));
42             Thread.sleep(wait_ms);
43         }
44     } catch (AmazonServiceException ase) {
45         System.out.println("Caught an AmazonServiceException, which means your request made it "
46             "to Amazon SQS, but was rejected with an error response for some reason.");
47     }

```

Set up the configuration. Give 1000 as the input argument.



Results from Eclipse console:



```

Problems @ Javadoc Declaration Console
<terminated> ClientSideProducer [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jt
=====
PRODUCER
=====

Sending a message: Message for server at: 2015/11/06 08:56:11
Sending a message: Message for server at: 2015/11/06 08:56:13
Sending a message: Message for server at: 2015/11/06 08:56:14
Sending a message: Message for server at: 2015/11/06 08:56:15
Sending a message: Message for server at: 2015/11/06 08:56:16
Sending a message: Message for server at: 2015/11/06 08:56:17
Sending a message: Message for server at: 2015/11/06 08:56:18
Sending a message: Message for server at: 2015/11/06 08:56:19
Sending a message: Message for server at: 2015/11/06 08:56:20

```

We can check the SQS Queue from the AWS console to see if the messages are sent into the queue.

Queues			
Filter by Prefix:		Messages Available	Messages in Flight
Name			
<input checked="" type="checkbox"/> MyQueue		65	0

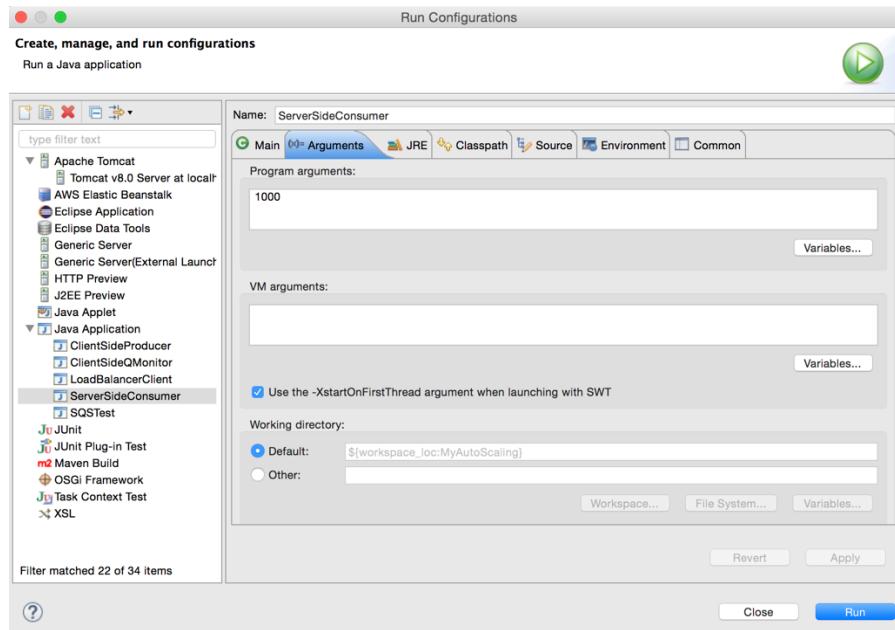
ServerSideConsumer:

```

24 public class ServerSideConsumer {
25
26     static final int CONSUMER_INTERVAL_ms = 5000;
27
28     public static void main(String[] args) throws Exception {
29
30         /* replace with your access key and secret key */
31         AmazonSQS sqs = new AmazonSQSClient(new BasicAWSCredentials("Access key", "Secret access key"));
32
33         Region usEast1 = Region.getRegion(Regions.US_EAST_1);
34         sqs.setRegion(usEast1);
35
36         int consumer_interval_ms = args.length > 0 ? Integer.valueOf(args[0]).intValue() : CONSUMER_INTERVAL_ms;
37
38         System.out.println("=====");
39         System.out.println("CONSUMER");
40         System.out.println("=====\\n");
41
42         try {
43             while (true) {
44                 // List queues
45                 for (String queueUrl : sqs.listQueues().getQueueUrls()) {
46                     System.out.println("\nQueueUrl: " + queueUrl);
47
48                     // Receive messages
49                     System.out.print("Checking msg: ");
50                     Message msg = consumeMsg(sqs, queueUrl);
51                     if (msg != null) {
52                         System.out.println(msg.getBody());
53                         String messageReceiptHandle = msg.getReceiptHandle();
54                         sqs.deleteMessage(new DeleteMessageRequest(queueUrl, messageReceiptHandle));
55                     } else {
56                         System.out.println("No message consumed from queue.");
57                     }
58                 }
59                 Thread.sleep(consumer_interval_ms);
60                 System.out.println();
61             }
62         } catch (AmazonServiceException ase) {
63             System.out.println("Caught an AmazonServiceException, which means your request made it " +

```

It will delete the message after reading it.



The screenshot shows the Eclipse Console view with the following output:

```
<terminated> ServerSideConsumer [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Co
=====
CONSUMER
=====

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 08:52:59

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 08:53:53

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 08:55:42

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 08:53:54

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 08:53:50
```

ClientSideQMonitor:

```

19 /**
20  * prints current time and estimated length of queue (visible messages)
21  * every WAIT_ms milliseconds, ad infinitum.
22 */
23 public class ClientSideQMonitor {
24     static final int WAIT_ms = 5000;
25     public static void main(String[] args) throws Exception {
26         AWS Credentials credentials = null;
27         try {
28             credentials = new ProfileCredentialsProvider("default").getCredentials();
29         } catch (Exception e) {
30             throw new AmazonClientException(
31                 "Cannot load the credentials from the credential profiles file. " +
32                 "location (/Users/hqiu/.aws/credentials), and is in valid format." , e);
33         }
34         AmazonSQS sqs = new AmazonSQSClient(credentials);
35         Region usEast1 = Region.getRegion(Regions.US_EAST_1);
36         sqs.setRegion(usEast1);
37         int wait_ms = args.length > 0 ? Integer.valueOf(args[0]).intValue() : WAIT_ms;
38         System.out.println("Monitoring SQS");
39         try {
40             // get queue length
41             GetQueueAttributesRequest attrReq = new GetQueueAttributesRequest(
42                 "https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue" );
43             attrReq.setAttributeNames(Arrays.asList(
44                 "ApproximateNumberOfMessages", "ApproximateNumberOfMessagesNotVisible"
45             ));
46             while (true) {
47                 GetQueueAttributesResult attrResult = sqs.getQueueAttributes(attrReq);
48                 Map<String, String> attrs = attrResult.getAttributes();
49                 System.out.println(
50                     (new SimpleDateFormat("yyyy/MM/dd HH:mm:ss")).format(new Date())
51                     + ": " + attrs.get("ApproximateNumberOfMessages"));
52                 Thread.sleep(wait_ms);
53             }
54         } catch (AmazonServiceException ase) {
55             System.out.println("Caught an AmazonServiceException, which means your request made it " +
56                 "to Amazon SQS, but was rejected with an error response for some reason.");
57         }
58     }
59 }

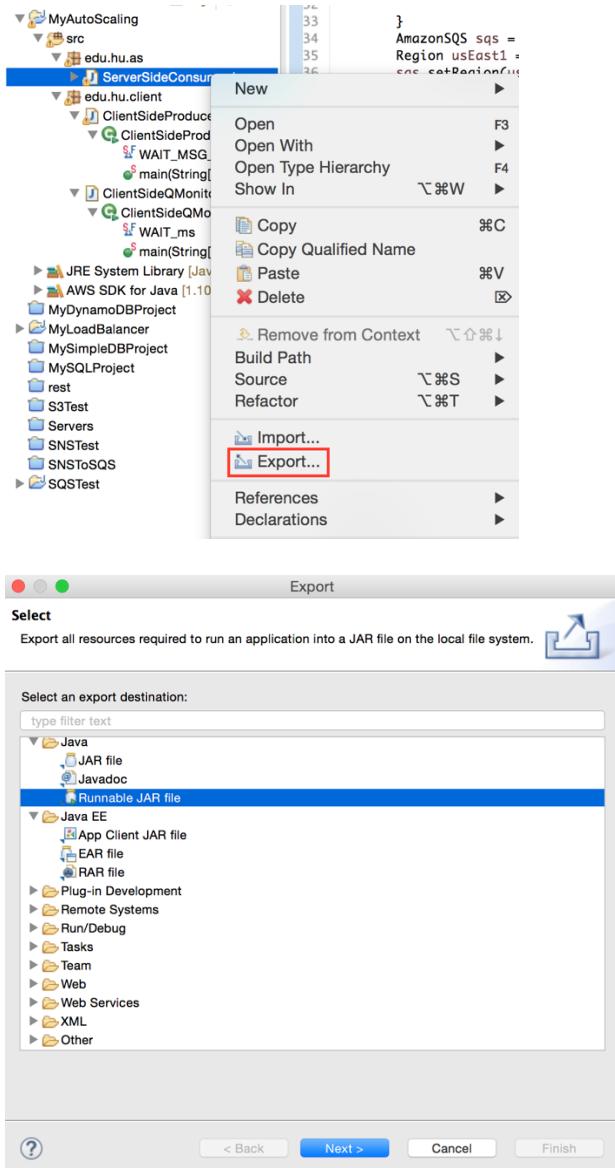
```

```

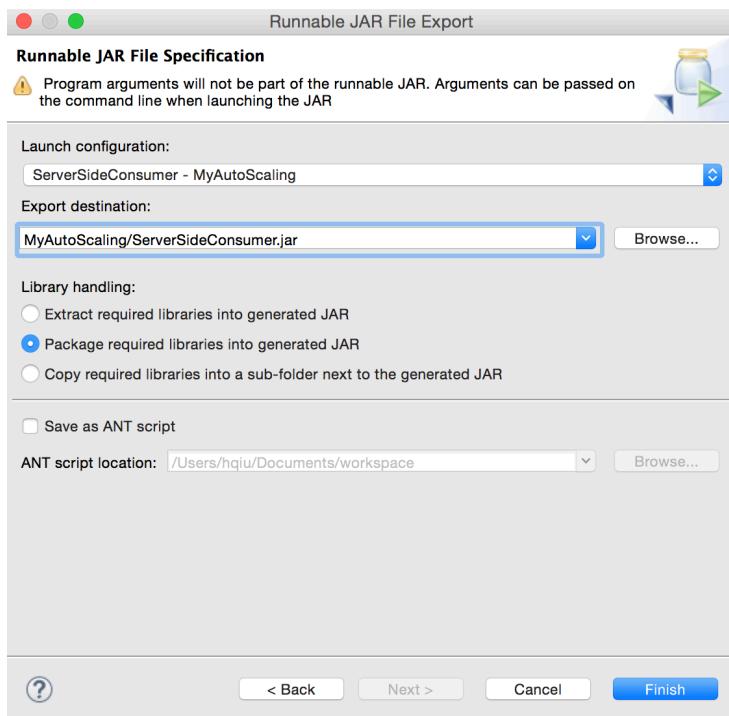
Problems @ Javadoc Declaration Console ✎
<terminated> ClientSideQMonitor [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Con
Monitoring SQS
2015/11/06 09:07:26: 5
2015/11/06 09:07:28: 7
2015/11/06 09:07:30: 5
2015/11/06 09:07:32: 11
2015/11/06 09:07:34: 13
2015/11/06 09:07:36: 15
2015/11/06 09:07:38: 11
2015/11/06 09:07:40: 13
2015/11/06 09:07:42: 17
2015/11/06 09:07:44: 19
2015/11/06 09:07:46: 22
2015/11/06 09:07:48: 27
2015/11/06 09:07:50: 29
2015/11/06 09:07:52: 29
2015/11/06 09:07:54: 27
2015/11/06 09:07:56: 35
2015/11/06 09:07:58: 29
2015/11/06 09:08:01: 35

```

- 4) Create three executable jars, each with one of the above classes as the Main class.



For each jar, the launch configuration should correspond to the Main class.



Verify the jars locally.

```
hqiu@bos-mpdei>> java -jar ClientSideProducer.jar 1000
=====
PRODUCER
=====

Sending a message: Message for server at: 2015/11/06 09:24:25
Sending a message: Message for server at: 2015/11/06 09:24:27
Sending a message: Message for server at: 2015/11/06 09:24:28
Sending a message: Message for server at: 2015/11/06 09:24:29
Sending a message: Message for server at: 2015/11/06 09:24:30
Sending a message: Message for server at: 2015/11/06 09:24:31
Sending a message: Message for server at: 2015/11/06 09:24:32
Sending a message: Message for server at: 2015/11/06 09:24:33
Sending a message: Message for server at: 2015/11/06 09:24:34
```

```
hqiu@bos-mpdei>> java -jar ClientSideQMonitor.jar 1000
Monitoring SQS
2015/11/06 09:24:06: 33
2015/11/06 09:24:07: 33
2015/11/06 09:24:08: 33
2015/11/06 09:24:09: 33
```

```
hqiu@bos-mpdei>> java -jar ServerSideConsumer.jar 2000
=====
CONSUMER
=====

QueueUrl: https://sns.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 09:07:36

QueueUrl: https://sns.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 09:24:28

QueueUrl: https://sns.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 09:15:57

QueueUrl: https://sns.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 09:07:25
```

- 5) Launch a EC2 instance and copy the executable jar with ServerSideConsumer to it. The AMI I chose already has Java 1.8 installed.

Filter by tags and attributes or search by keyword									
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
<input checked="" type="checkbox"/>	i-6cc236d2	t2.micro	us-east-1a	running	2/2 checks ...	None	None	ec2-54-210-8-174.com...	54.210.8.174
Instance: i-6cc236d2 Public DNS: ec2-54-210-8-174.compute-1.amazonaws.com									
Description	Status Checks	Monitoring	Tags						
Instance ID	i-6cc236d2				Public DNS	ec2-54-210-8-174.compute-1.amazonaws.com			
Instance state	running				Public IP	54.210.8.174			
Instance type	t2.micro				Elastic IP	-			
Private DNS	ip-172-31-60-29.ec2.internal				Availability zone	us-east-1a			
Private IPs	172.31.60.29				Security groups	launch-hqiu.. view rules			
Secondary private IPs					Scheduled events	No scheduled events			
VPC ID	vpc-dfb48aba				AMI ID	aws-elasticbeanstalk-amzn-2015.09.0.x86_64-java8-hvm-201510302253 (ami-fe057694)			

```
hqiu@bos-mpdei>> scp -i ec2hqiup.pem ~/Documents/workspace/MyAutoScaling/ServerSideConsumer.jar ec2-user@54.210.8.174:/home/ec2-user/AutoScaling
ServerSideConsumer.jar                                         100%   29MB   9.8MB/s   00:03
```

```
hqiu@bos-mpdei>> ssh -i "ec2hqiup.pem" ec2-user@54.210.8.174
   _\ _\_)_
   \_ ( /  Amazon Linux AMI
     \_\_|_|

https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
[ec2-user@ip-172-31-60-29 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-60-29 ~]$ mkdir AutoScaling
[ec2-user@ip-172-31-60-29 ~]$ cd AutoScaling/
[ec2-user@ip-172-31-60-29 AutoScaling]$ ls
ServerSideConsumer.jar
[ec2-user@ip-172-31-60-29 AutoScaling]$ sudo chmod 755 ServerSideConsumer.jar
```

- 6) Launch the ClientSideProducer to push messages into the SQS queue.

```

Problems Javadoc Declaration Console 
<terminated> ClientSideProducer [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45
=====
PRODUCER
=====

Sending a message: Message for server at: 2015/11/06 10:01:30
Sending a message: Message for server at: 2015/11/06 10:01:32
Sending a message: Message for server at: 2015/11/06 10:01:33
Sending a message: Message for server at: 2015/11/06 10:01:34
Sending a message: Message for server at: 2015/11/06 10:01:35
Sending a message: Message for server at: 2015/11/06 10:01:36
Sending a message: Message for server at: 2015/11/06 10:01:37
Sending a message: Message for server at: 2015/11/06 10:01:38

View/Delete Messages in MyQueue
Cancel 

View up to: 30 messages Poll queue for: 30 seconds Polling for new messages once every 2 seconds.
Stop Now

Delete Body Size Sent Receive Count
 Message for server at: 2015/11/06 10:01:33 42 bytes 2015-11-06 10:01:31 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:35 42 bytes 2015-11-06 10:01:34 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:40 42 bytes 2015-11-06 10:01:39 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:30 42 bytes 2015-11-06 10:01:29 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:34 42 bytes 2015-11-06 10:01:33 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:32 42 bytes 2015-11-06 10:01:30 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:39 42 bytes 2015-11-06 10:01:38 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:38 42 bytes 2015-11-06 10:01:37 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:36 42 bytes 2015-11-06 10:01:35 GMT-05:00 1
 Message for server at: 2015/11/06 10:01:37 42 bytes 2015-11-06 10:01:36 GMT-05:00 1

[ec2-user@ip-172-31-60-29 AutoScaling]$ java -jar ServerSideConsumer.jar 1000
=====
CONSUMER
=====

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:38

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:37

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:36

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:33

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:32

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:30

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:39

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:34

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:40

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:01:35

```

We can see that, the messages sent from the ClientSideProducer, the messages placed in the queue and the messages get from ServerSideConsumer are exactly the same!

So the server side consumer truly consumes messages from the queue!

The ServerSideConsumer drains the SQS Queue.

Queues		Messages Available	Messages in Flight
Filter by Prefix:			
Name		Messages Available	Messages in Flight
MyQueue		0	0

7) Set the jar to run it automatically on every bootup.

```
[ec2-user@ip-172-31-60-29 ~]$ sudo vi /etc/rc.local
```

```
ec2-user@i.../www/html      ec2-user@i...31-60-29:~      bash
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
java -jar /home/ec2-user/AutoScaling/ServerSideConsumer.jar 1000
~
```

Start the ClientSideProducer to place messages into the SQS queue.

Stop or reboot the EC2 instance.

Filter by tags and attributes or search by keyword						
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
<input checked="" type="checkbox"/>	i-6cc236d2	t2.micro	us-east-1a	stopped	None	

We will see the “Message Available” of “MyQueue” is decreasing!

Check the System Log file to confirm that messages are indeed read!

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there's a navigation bar with 'Launch Instance', 'Connect', and 'Actions'. Below that is a search bar and a table listing EC2 instances. One instance, 'i-6cc236d2', is selected. The 'Actions' dropdown menu is open over this instance, showing options like 'Connect', 'Get Windows Password', 'Launch More Like This', 'Instance State', 'Instance Settings' (which is currently selected), 'Image', 'Networking', 'CloudWatch Monitoring', 'Add/Edit Tags', 'Attach to Auto Scaling Group', 'Change Instance Type', 'Change Termination Protection', 'View/Change User Data', 'Change Shutdown Behavior', and 'Get System Log'. The 'Get System Log' option is highlighted with a yellow background. The main table below shows the instance details: 'i-6cc236d2', 'Public IP: 54.165.137.161', 'Status: Initializing', 'Alarm Status: None', and 'Public DNS: ec2-54-165-137-161.compute-1.amazonaws.com'. There are also tabs for 'CloudWatch metrics', 'Basic monitoring', and 'CloudWatch Detailed Monitoring'.

System Log: i-6cc236d2

C X

```
Starting sm-client: [ OK ]
Starting crond: [ OK ]
Starting atd: [ OK ]
Starting cloud-init: Cloud-init v. 0.7.6 running 'modules:final' at Fri, 06 Nov 2015 15:11:21 +0000
Cloud-init v. 0.7.6 finished at Fri, 06 Nov 2015 15:11:21 +0000. Datasource DataSourceEc2. Up 17.0
=====
CONSUMER
=====

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:10:03

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:09:51

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:10:00

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:10:18

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:09:53

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:09:55

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:10:20

QueueUrl: https://sqs.us-east-1.amazonaws.com/217134905396/MyQueue
Checking msg: Message for server at: 2015/11/06 10:10:46
```

Close

Terminate all the instances in the end.