Elastic Load Balancing
Auto Scaling

CSCI E90 Cloud Computing

Harvard University Extension School

Zoran B. Djordjević

ELB & AS

- Amazon AWS offers several services that let you applications handle large volumes of users or large computational loads.
- Two most basic services that help you applications scale are AWS Elastic Load Balancing and AWS Auto Scaling.

Elastic Load Balancing

Elastic Load Balancing

- Elastic Load Balancing automatically distributes incoming TCP application traffic across multiple Amazon EC2 instances.
- We can use load balancing to achieve fault tolerance in our applications and seamlessly provide the amount capacity we need in response to incoming application traffic.
- As traffic goes up, we might need to run our web application on multiple instances to make sure it says highly available.
- A single URL, such as www.my-great-idea.com, could represent several instances in several zones.
- ELB makes it easy to scale in response to he growth of incoming traffic.
- ELB can easily distribute traffic to multiple zones.

Overview of ELB

- Our application cluster is exposed through the DNS name of the Elastic Load Balancer.
- All traffic goes to load balancer. Load balancer distributes requests to multiple EC2 instances.
- A load balancer can span multiple zones with the same region, but it cannot span multiple regions.
- For critical applications, it is a good idea to have instances in multiple regions. Incoming traffic should be load balanced equally across all Availability Zones.
- It is a good idea to have equal capacity in each availability zone.

Features

- Elastic Load Balancing supports Amazon EC2 instances with any operating system currently supported by the Amazon EC2 service.
- You can perform load balancing for the following TCP ports: 25, 80, 443, and 1024-65535.
- Each Elastic Load Balancer has an associated IPv4, IPv6, and dualstack (both IPv4 and IPv6) DNS name. IPv6 is not supported in VPC at this time.
- You can configure your Amazon EC2 instances to only accept traffic from the Elastic Load Balancer.
- If using Amazon Virtual Private Cloud, you can configure security groups for the front-end of your Elastic Load Balancer.
- You can map HTTP port 80 and HTTPS port 443 to a single Elastic Load Balancer.

Features

- Elastic Load Balancer does not cap the number of connections that it can attempt to establish with your load balanced Amazon EC2 instances. You can expect this number to scale with the number of concurrent HTTP, HTTPS, or SSL requests or the number of concurrent TCP connections that the Elastic Load Balancer receives.
- Manually registering a Paid AMI based Amazon EC2 instance with the Elastic Load Balancer or using a Paid AMI with an Auto Scaling Group that is associated with the Elastic Load Balancer is not supported.

Sticky Session

- By default a load balancer routes each request independently to the application instance with the smallest load.
- However, we can use the *sticky session* feature (also known as *session affinity*) which enables the load balancer to bind a user's session to a specific application instance.
- Session affinity ensures that all requests coming from the user during the session will be sent to the same application instance what simplifies your application design and in some cases results in faster response times.

HTTPS Support

- HTTPS Support is a feature that allows you to use the SSL/TLS protocol for encrypted connections (also known as SSL offload).
- HTTPS Support enables traffic encryption between your load balancer and clients that initiate HTTPS sessions with your load balancer.
- This frees resources on your application instances and could improve overall performance.

Identify Client's IP, **X-Forwarded-For**

- The X-Forwarded-For request header helps us identify the IP address of a client.
- Because load balancers intercept traffic between clients and servers, our server access logs contain only the IP address of the load balancer.
- The IP address of the client can be read from

  `X-Forwarded-For` request header.
- Elastic Load Balancing stores the IP address of the client in the `X-Forwarded-For` request header and passes the header along to the servers.

Demo of Load Balancing

- In the following application we will demonstrate how a load balancer distributes incoming requests to servers in the cluster we assigned to the balancer.
- We will install an Apache server on a standard AWS Linux instance and add a small PHP script, which will identify for us the server which received the request and sent back the response to the client.

Create a New Linux Instance

- We will select ami-51792c38 and create a small or micro instance

Launch Instance | Connect | Actions

Filter: All instances | All instance types | 1 to 2

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Key Name
	i-1737336d	m1.small	us-east-1d	stopped		None		hu0906
	i-e023fa9b	t1.micro	us-east-1b	running	Initializing	None	ec2-54-227-73-25.com...	hu0906

Instance: i-e023fa9b Public DNS: ec2-54-227-73-25.compute-1.amazonaws.com

Description | Status Checks | Monitoring | Tags

Instance ID	i-e023fa9b	Public DNS	ec2-54-227-73-25.compute-1.amazonaws.com
Instance state	running	Elastic IP	-
Instance type	t1.micro	Private DNS	ip-10-238-185-225.ec2.internal
Availability zone	us-east-1b	Private IPs	10.238.185.225
Security groups	launch-wizard-2. view rules	Secondary private IPs	-
Scheduled events	No scheduled events	VPC ID	-
AMI ID	amzn-ami-pv-2013.09.0.i386-ebs (ami-51792c38)	Subnet ID	-
Platform	-	Network interfaces	-
IAM role	-	Source/dest. check	False
Key pair name	hu0906		
Owner	951414139794	EBS-optimized	False
Launch time	2013-10-24T21:07:38.000Z (less than one hour)	Root device type	ebs
Termination protection	False	Root device	/dev/sda1
Lifecycle	normal	Block devices	/dev/sda1

Identify Linux User for the Connection, `ec2-user`

Connect To Your Instance ×

I would like to connect with
- (●) A standalone SSH client
- ( ) A Java SSH Client directly from my browser (Java required)

To access your instance:

1. Open an SSH client. (find out how to connect using PuTTY)
2. Locate your private key file (hu0906.pem). The wizard automatically detects the key you used to launch the instance.
3. Your key must not be publicly viewable for SSH to work. Use this command if needed:

```
chmod 400 hu0906.pem
```

4. Connect to your instance using its Public DNS:

```
ec2-54-227-73-25.compute-1.amazonaws.com
```

Example:

```
ssh -i hu0906.pem ec2-user@ec2-54-227-73-25.compute-1.amazonaws.com
```

Please note that in most cases the username above will be correct, however please ensure that you read your AMI usage instructions to ensure that the AMI owner has not changed the default AMI username.

If you need any assistance connecting to your instance, please see our connection documentation.

Close

- We want to connect and install/update Apache server software.
- Apache is the Web Server of choice. Apache process is called `httpd` (`d` for daemon??)
- We will also need to install/update PHP software package, as well.

Connect

```
$ ssh -i hu0906.pem ec2-user@ec2-54-227-73-25.compute-1.amazonaws.com
The authenticity of host 'ec2-54-227-73-25.compute-1.amazonaws.com
(54.227.73.25)' can't be established.
RSA key fingerprint is
7e:82:55:1f:a9:ef:5f:15:51:89:79:bf:44:c4:3f:68.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-227-73-25.compute-
1.amazonaws.com,54.227.73.25' (RSA) to the list of known hosts.
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
https://aws.amazon.com/amazon-linux-ami/2013.09-release-notes/
[ec2-user@ip-10-238-185-225 ~]$

Run "sudo yum update" to apply all updates.

[ec2-user@ip-10-238-185-225 ~]$ sudo yum update
```

As `sudo` Install Apache Server, `httpd`

```
Install  1 Package (+5 Dependent packages)
Total download size: 2.0 M
Installed size: 3.9 M
Is this ok [y/d/N]: y
Downloading packages:
(1/6): apr-1.4.6-1.10.amzn1.i686.rpm                                | 109 kB     00:00
(2/6): apr-util-1.4.1-4.14.amzn1.i686.rpm                           |  84 kB     00:00
(3/6): apr-util-ldap-1.4.1-4.14.amzn1.i686.rpm                      |  17 kB     00:00
(4/6): generic-logos-17.0.0-2.5.amzn1.noarch.rpm                    | 589 kB     00:00
(5/6): httpd-2.2.25-1.0.amzn1.i686.rpm                              | 1.1 MB     00:00
(6/6): httpd-tools-2.2.25-1.0.amzn1.i686.rpm                        |  78 kB     00:00
--------------------------------------------------------------------------------------------
Total                                              3.4 MB/s | 2.0 MB     00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : apr-1.4.6-1.10.amzn1.i686                                    1/6
  Installing : generic-logos-17.0.0-2.5.amzn1.noarch                        2/6
  Installing : apr-util-1.4.1-4.14.amzn1.i686                               3/6
Verifying  : httpd-2.2.25-1.0.amzn1.i686                                    1/6
  Verifying  : apr-1.4.6-1.10.amzn1.i686                                    2/6
  Verifying  : httpd-tools-2.2.25-1.0.amzn1.i686                            3/6
  Verifying  : generic-logos-17.0.0-2.5.amzn1.noarch                        4/6
  Verifying  : apr-util-1.4.1-4.14.amzn1.i686                               5/6
  Verifying  : apr-util-ldap-1.4.1-4.14.amzn1.i686                          6/6
Installed:
  httpd.i686 0:2.2.25-1.0.amzn1
Dependency Installed:
  apr.i686 0:1.4.6-1.10.amzn1                apr-util.i686 0:1.4.1-4.14.amzn1
  apr-util-ldap.i686 0:1.4.1-4.14.amzn1      generic-logos.noarch 0:17.0.0-2.5.amzn1
  httpd-tools.i686 0:2.2.25-1.0.amzn1

Complete!
```

As `sudo` Install Apache Server, `httpd`

```
[ec2-user@ip-10-238-185-225 ~]$ sudo yum install httpd
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package httpd.i686 0:2.2.25-1.0.amzn1 will be installed
--> Processing Dependency: httpd-tools = 2.2.25-1.0.amzn1 for package: httpd-2.2.25-1.0.amzn1.i686
--> Processing Dependency: libapr-1.so.0 for package: httpd-2.2.25-1.0.amzn1.i686
---> Package apr.i686 0:1.4.6-1.10.amzn1 will be installed
---> Package generic-logos.noarch 0:17.0.0-2.5.amzn1 will be installed
---> Package httpd-tools.i686 0:2.2.25-1.0.amzn1 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
================================================================================
Package        Arch        Version              Repository      Size
================================================================================
httpd          i686        2.2.25-1.0.amzn1     amzn-main       1.1 M
Installing for dependencies:
 apr           i686        1.4.6-1.10.amzn1     amzn-main       109 k
 apr-util      i686        1.4.1-4.14.amzn1     amzn-main       84 k
 apr-util-ldap i686        1.4.1-4.14.amzn1     amzn-main       17 k
 generic-logos noarch      17.0.0-2.5.amzn1     amzn-main       589 k
 httpd-tools   i686        2.2.25-1.0.amzn1     amzn-main       78 k
Transaction Summary
================================================================================
```

Start Apache

```
[ec2-user@ip-10-238-185-225 ~]$ sudo /etc/init.d/httpd start
Starting httpd:                                            [  OK  ]
[ec2-user@ip-10-238-185-225 ~]$ ps -ef | grep httpd
root      4272     1  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4274  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4275  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4276  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4277  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4278  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4279  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4280  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
apache    4281  4272  0 21:46 ?        00:00:00 /usr/sbin/httpd
ec2-user  4283  1245  0 21:47 pts/0    00:00:00 grep httpd
```

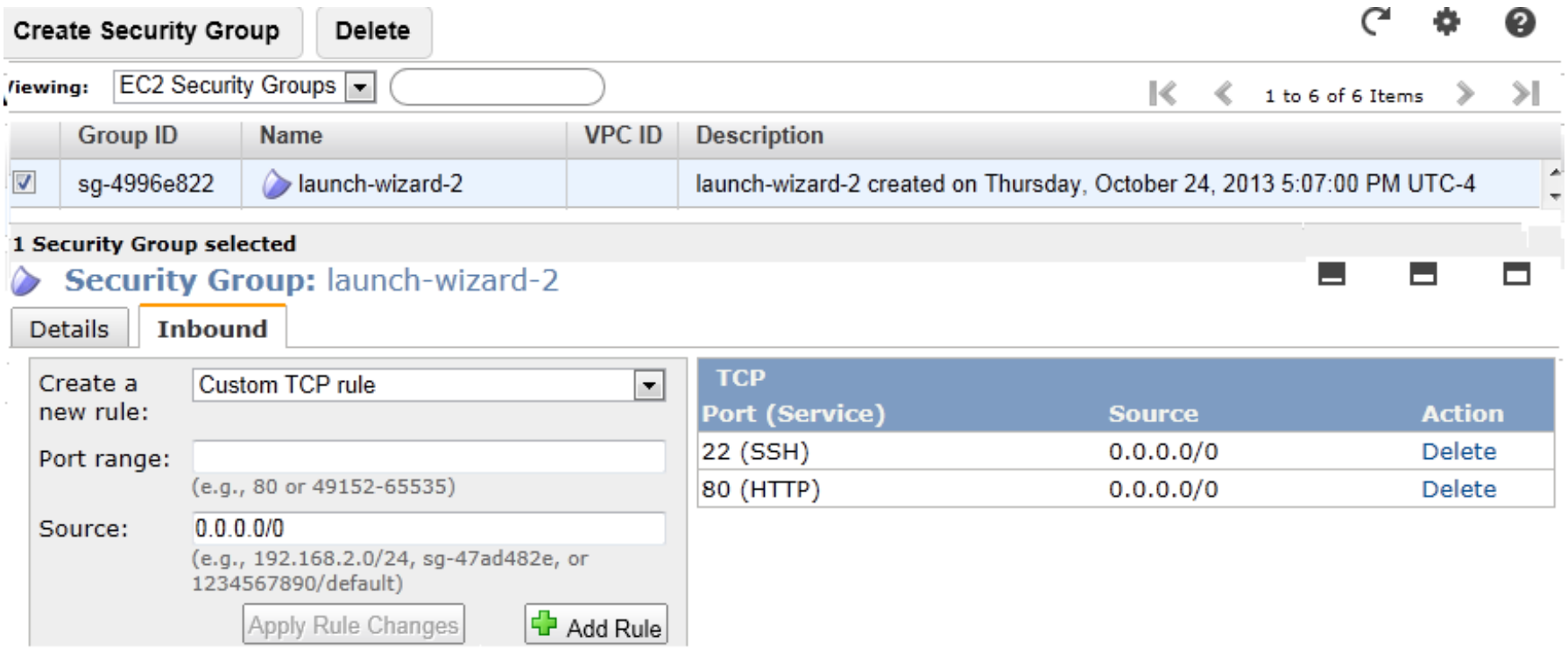
Install PHP

```
[ec2-user@ip-10-238-185-225 ~]$ sudo yum install php
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main/latest                                | 2.1 kB     00:00
amzn-updates/latest                             | 2.3 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package php.i686 0:5.3.27-1.0.amzn1 will be installed
--> Processing Dependency: php-cli(x86-32) = 5.3.27-1.0.amzn1 for package:
php-5.3.27-1.0.amzn1.i686
→  Processing Dependency: php-common(x86-32) = 5.3.27-1.0.amzn1 for
   package: php-5.3.27-1.0.amzn1.i686

→  . . . .
Installed:
  php.i686 0:5.3.27-1.0.amzn1

Dependency Installed:
  php-cli.i686 0:5.3.27-1.0.amzn1   php-common.i686 0:5.3.27-
1.0.amzn1

Complete!
```

Create `index.php` file

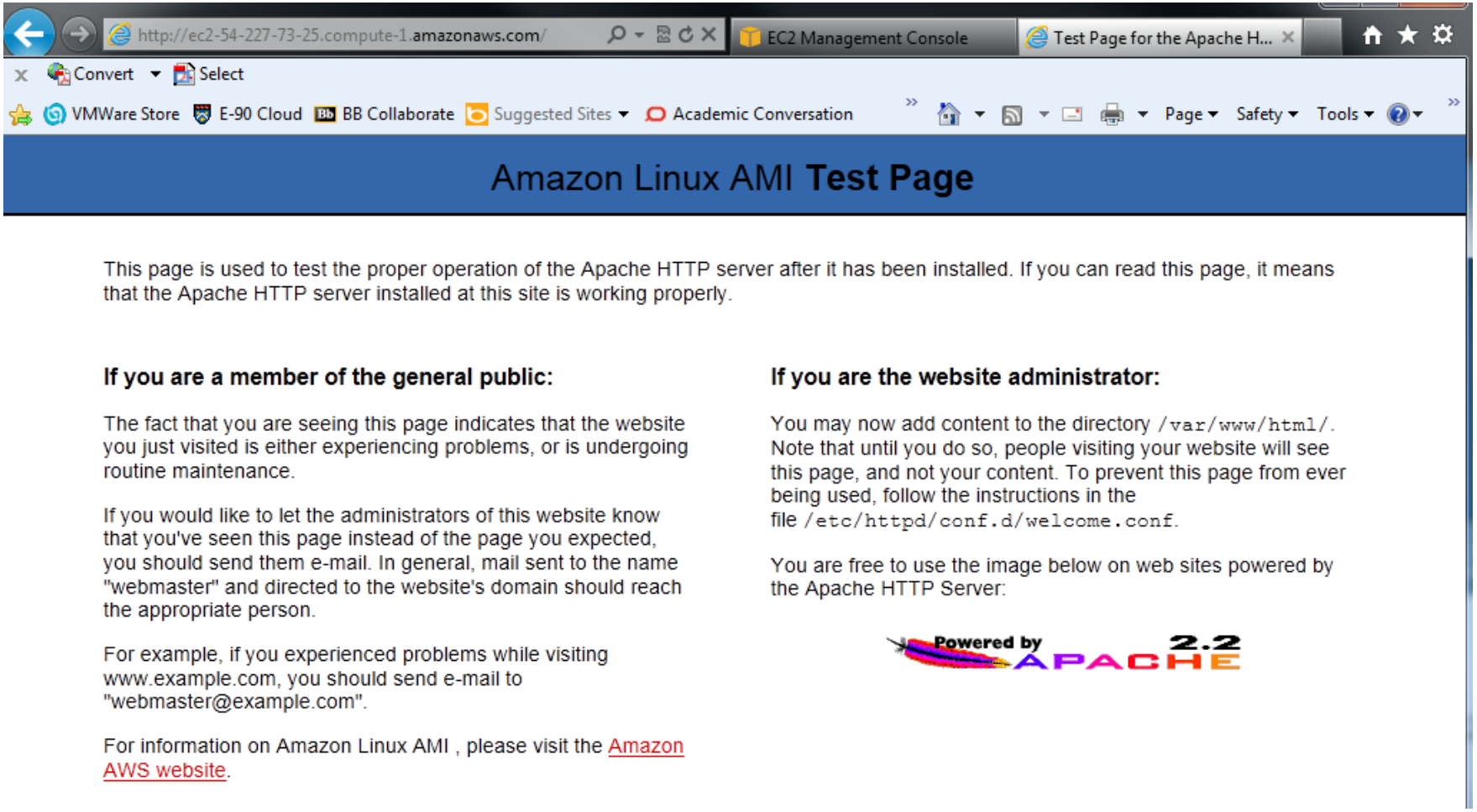
```
[ec2-user@ip-10-238-185-225 ~]$ cd /var/www/html
[ec2-user@ip-10-238-185-225 ~]$ sudo vi index.php
<?php
     echo "Helo! My IP addres is: ".$_SERVER['SERVER_ADDR'];  ?>
```

- In the EC2 Dashboard, select Security Groups and add port 80 to the security group `launch-wizard-2`



Examine your “Web site”

- Go to your browser and type Public DNS of your Linux box.
- You get Amazon Linux welcome page and not `index.php`



Examine your “Web site”

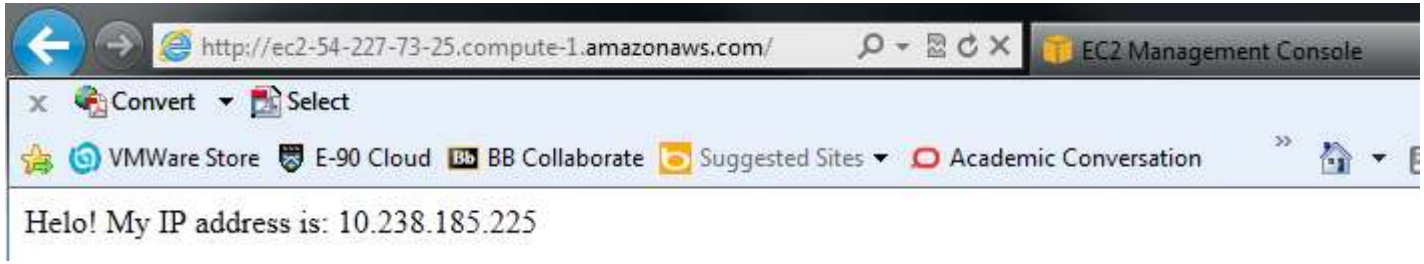
- Comment out the content of file `/etc/httpd/conf.d/welcome.conf`

```
$ sudo vi /etc/httpd/conf.d/welcome.conf
#<LocationMatch "^/+$">
#    Options -Indexes
#    ErrorDocument 403 /error/noindex.html
#</LocationMatch>
```

- Perhaps, stop and start Apache

```
$ sudo /etc/init.d/httpd stop
Stopping httpd:                                            [OK]
$ sudo /etc/init.d/httpd start
Starting httpd:                                            [OK]
```

- Visit the site, again:



- Server shows the internal IP address of the box.
- When on server, you only see the internal IP.

Make sure Apache starts on boot

- To make sure the Apache will start at the boot time, please add line:

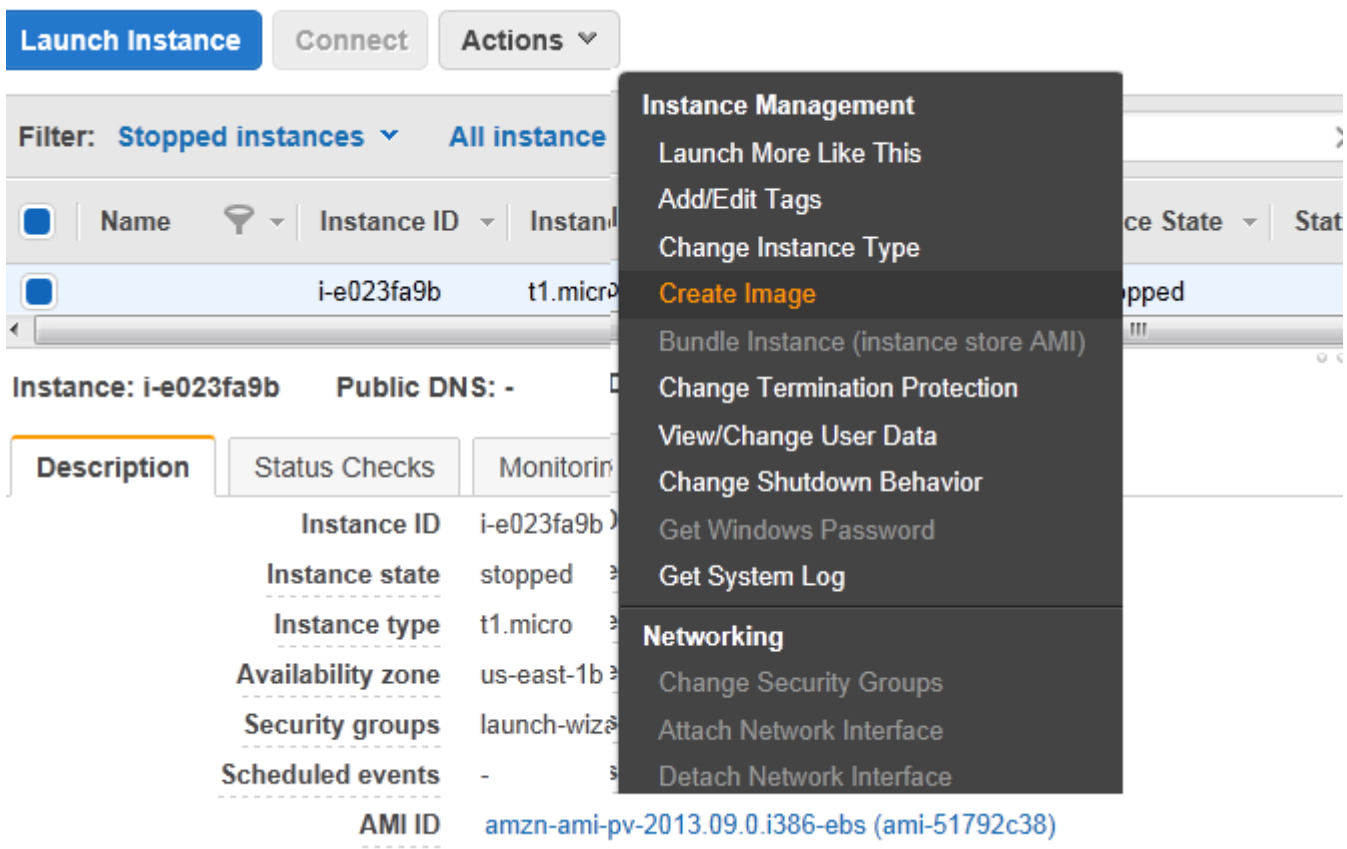
  `/etc/init.d/httpd start`

- to file `/etc/rc.local`
- On my system, after I "`sudo vi`"-ed, this file looked like this:

```
#!/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
```

Create AMI out of this Instance

- Go to `EC2 Dashoard`, right click on the running instance and select `Create Image (EBS AMI)`.



Create New Image Wizard

Create Image

Instance ID: i-e023fa9b

Image name: Copy of ami-51792c38

Image description: Amazon Linux with Apache

No reboot:

Instance Volumes

Type	Device	Snapshot	Size (GB)	Volume Type	IOPS	Delete on Termination
Root	/dev/sda1	snap-7962c379	8	Standard	N/A	

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.

Cancel Create Image

Configure Security Group

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 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-4996e822	launch-wizard-2	launch-wizard-2 created on Thursday, October 24...	Copy to new

Inbound rules for sg-4996e822

Protocol	Type	Port Range (Code)	Source
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0

Cancel | Previous | Review and Launch

Once new AMI is Available, Create a few (3) instances

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

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

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

Quick Start

My AMIs

AWS Marketplace

Community AMIs

1 to 1 of 1 AMIs

Copy of ami-51792c38 - ami-b70e51de

Amazon Linux with Apache

Root device type: ebs Virtualization type: paravirtual Owner: 951414139794

Select

32-bit

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Currently selected: t1.micro (up to 2 ECUs, 1 vCPUs, 0.613 GiB memory, EBS only)

All instance types

Micro instances
Free tier eligible

General purpose

Compute optimized

Micro instances

Micro instances are a low-cost instance option, providing a small amount of CPU resources. They are suited for lower throughput applications, and websites that require additional compute cycles periodically, but are not appropriate for applications that require sustained CPU performance. Popular uses for micro instances include low traffic websites or blogs, small administrative applications, bastion hosts, and free trials to explore EC2 functionality.

Size	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t1.micro	up to 2	1	0.613	EBS only		

Cancel | Previous | Review and Launch | Next: Configure

Configure Instance Details, Number of Instances

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

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 pricing, assign a management role to the instance, and more.

Number of instances	3
Purchasing option	☐ Request Spot Instances
Network	Launch into EC2-Classic — Create new VPC
Availability Zone	us-east-1b
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☑ Enable CloudWatch detailed monitoring Additional charges apply.

Cancel | Previous | Review and Launch | Next: Add Storage

Configure Security Group

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 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-4996e822	launch-wizard-2	launch-wizard-2 created on Thursday, October 24...	Copy to new

Inbound rules for sg-4996e822

Protocol	Type	Port Range (Code)	Source
SSH	TCP	22	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0

Cancel | Previous | Review and Launch

Review and Launch

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

Step 7: Review Instance Launch

AMI Details

Edit AMI

Copy of ami-51792c38 - ami-b70e51de
Amazon Linux with Apache
Root Device Type: ebs Virtualization type: paravirtual

Instance Type

Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance
t1.micro	up to 2	1	0.613	EBS only	-	Very Low

Security Groups

Edit security groups

Security Group ID	Name	Description
sg-4996e822	launch-wizard-2	launch-wizard-2 created on Thursday, October 24, 2013 5:07:00 PM UTC-4

All selected security groups inbound rules

Security Group ID	Protocol	Type	Port Range (Code)	Source
sg-4996e822	SSH	TCP	22	0.0.0.0/0
sg-4996e822	HTTP	TCP	80	0.0.0.0/0

Instance Details

Edit instance details

Storage

Edit storage

Cancel | Previous | Launch

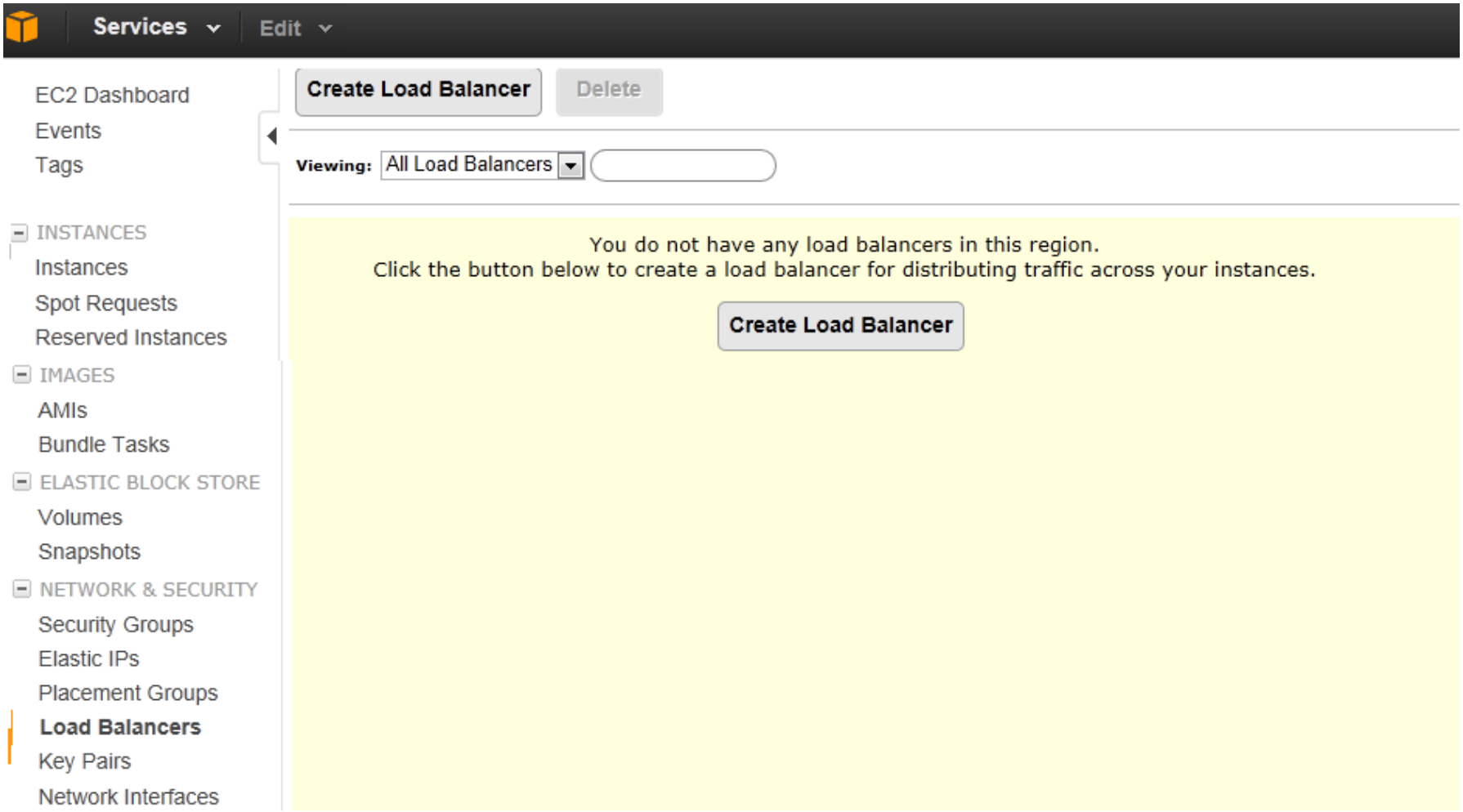
Filter: Running instances | All instance types

1 to 4 of 4 Instances

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Public DNS	Key Name	Launch Time
	i-d2e202a9	t1.micro	us-east-1b	running	2/2 checks...	ec2-54-226-127-104.co...	hu0906	2013-10-25T0...
	i-d0e202ab	t1.micro	us-east-1b	running	2/2 checks...	ec2-107-21-68-109.co...	hu0906	2013-10-25T0...
	i-afa10bd6	t1.micro	us-east-1d	running	Initializing	ec2-54-205-108-28.co...	hu0906	2013-10-25T0...
	i-a1a10bd8	t1.micro	us-east-1d	running	Initializing	ec2-54-224-209-18.co...	hu0906	2013-10-25T0...

Create Load Balancer

- In every region you need to create a separate load balancer
- Load balancers will not direct traffic to a region different from their own.



Give Load Balancer a Name and a Port

Create a New Load Balancer Cancel

DEFINE LOAD BALANCER | CONFIGURE HEALTH CHECK | ADD EC2 INSTANCES | REVIEW

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 might create. You will also need to 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 your load balancer with a standard web server on port 80.

Load Balancer Name: ApacheServersDemoLB

Create LB inside: EC2

Create an internal load balancer: (what's this?)

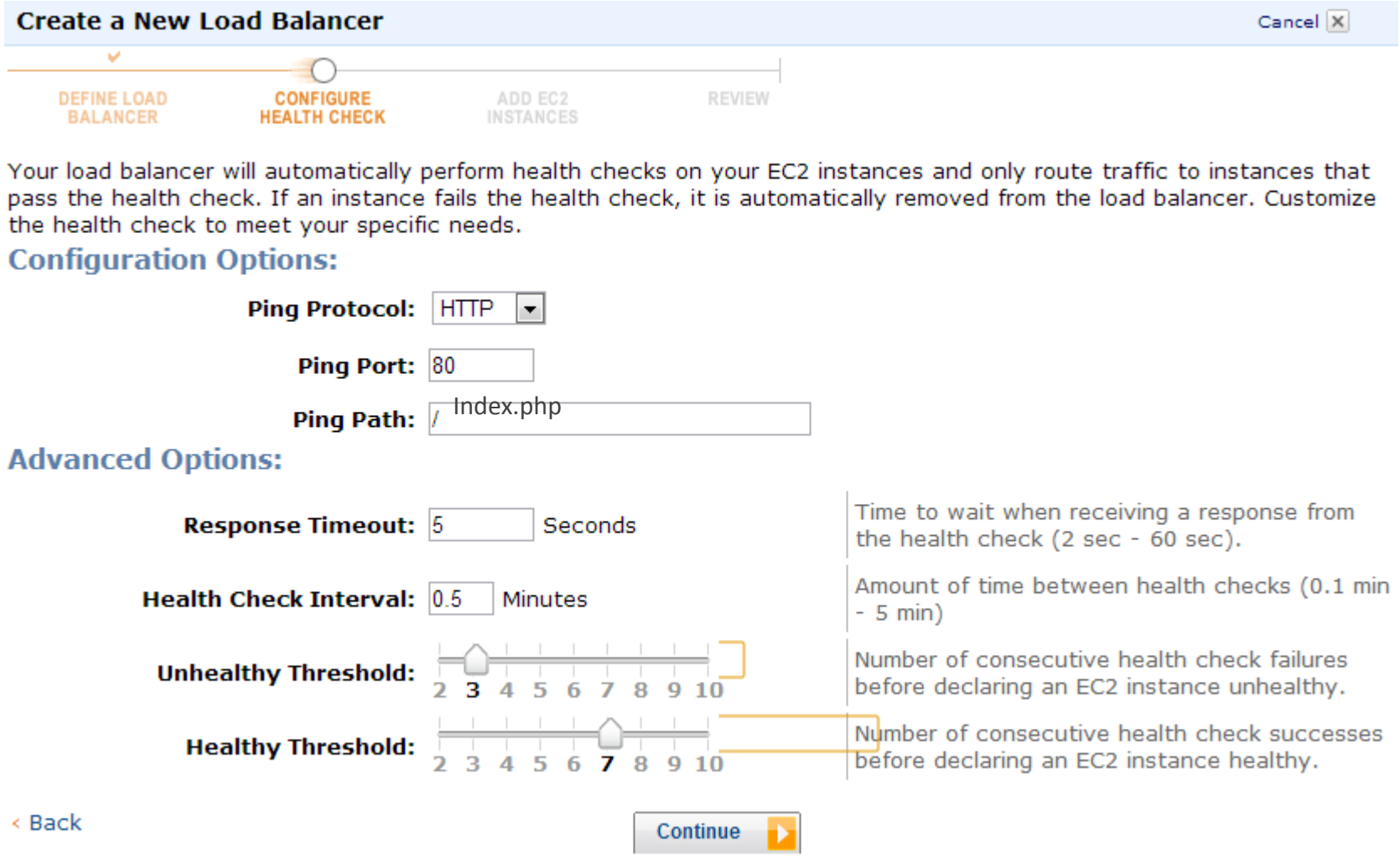
Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Actions
HTTP	80	HTTP	80	Remove
HTTP		HTTP		Save

Continue

Setup of Load Balancer

- This is a critical page and here we set the frequency with which balancer checks (pings) instances for their health



Select Instances, Warning on Availability Zones

- When selecting instances to be assigned to the load balancer, you should work with equally sized groups of instances in different availability zones

Create a New Load Balancer Cancel

DEFINE LOAD BALANCER | CONFIGURE HEALTH CHECK | ADD EC2 INSTANCES | REVIEW

The table below lists all your running EC2 Instances that are not already behind another load balancer or part of an auto-scaling capacity group. Check the boxes in the Select column to add those instances to this load balancer.

Manually Add Instances to Load Balancer:

Select	Instance	Name	State	Security Groups	Availability Zone
☑	i-d2e202a9		running	launch-wizard-2	us-east-1b
☑	i-d0e202ab		running	launch-wizard-2	us-east-1b
☑	i-afa10bd6		running	launch-wizard-2	us-east-1d
☑	i-a1a10bd8		running	launch-wizard-2	us-east-1d

select all | select none

Availability Zone Distribution:

2 instances in us-east-1b
2 instances in us-east-1d

‹ Back Continue

Final Check

Create a New Load Balancer Cancel

DEFINE LOAD BALANCER ✓ | CONFIGURE HEALTH CHECK ✓ | ADD EC2 INSTANCES ✓ | REVIEW

DEFINE LOAD BALANCER

Load Balancer Name: ApacheServerDemoLB
Scheme: internet-facing
Port Configuration:
80 (HTTP) forwarding to 80 (HTTP)

Edit Load Balancer Definition

CONFIGURE HEALTH CHECK

Ping Target: HTTP:80:/index.html
Timeout: 5
Interval: 0.5

Unhealthy Threshold: 2
Healthy Threshold: 5

Edit Health Check

ADD EC2 INSTANCES

EC2 Instances: i-d2e202a9, i-d0e202ab, i-afa10bd6, i-a1a10bd8

Edit EC2 Instance Selection

VPC INFORMATION

VPC:
Subnets:

‹ Back

Create

Please review your selections on this page. Clicking "Create" will launch your load balancer. Check the Amazon EC2 product page for load balancer pricing info

Create a New Load Balancer Cancel

Your load balancer has been created.

Note: It may take a few minutes for your instances to become active in the new load balancer.
› View my load balancers and check their status.

‹ Back

Close

Load Balancer is Created

- Description of Load Balancer contains its DNS Name

Create Load Balancer | Delete

Viewing: All Load Balancers

1 to

Load Balancer Name	DNS Name	Port Configuration	Availability Zones
ApacheServerDemoLB	ApacheServerDemoLB-1371290937.us-east-1.	80 (HTTP) forwarding to 80 (HTTP)	us-east-1b, us-east-1d

1 Load Balancer selected

Load Balancer: ApacheServerDemoLB

Description | Instances | Health Check | Monitoring | Security | Listeners

DNS Name:	ApacheServerDemoLB-1371290937.us-east-1.elb.amazonaws.com (A Record) ipv6.ApacheServerDemoLB-1371290937.us-east-1.elb.amazonaws.com (AAAA Record) dualstack.ApacheServerDemoLB-1371290937.us-east-1.elb.amazonaws.com (A or AAAA 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 LoadBalancer 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 the Using Domain Names With Elastic Load Balancing
Scheme:	internet-facing
Status:	0 of 4 instances in service
Port Configuration:	80 (HTTP) forwarding to 80 (HTTP) Stickiness: Disabled **(edit)**
Availability Zones:	us-east-1b us-east-1d
Source Security Group:	amazon-elb/amazon-elb-sg Owner Alias: amazon-elb Group Name: amazon-elb-sg
Hosted Zone ID:	Z3DZXE0Q79N41H
VPC ID:	-

Enable stickiness

- As mentioned previously, we might chooses to enable session affinity (session stickiness).
- Modern RESTful believers consider stickiness one of the great sins since it breaks the statelessness of your application.
- Stickiness is quite often very useful and is routinely used in financial and other commercial applications that do not have to scale to the Amazon's or Google's heights.
- If you select (edit) to the right of Port Configuration, on the previous page, you can choose the type of sticky session you will use.

Instances

- Load balancer's page tells us all instance `ids` and their location.
- +/- in the upper right corner let us add or remove instances/zones from the cluster.

Load Balancer Name	DNS Name	Port Configuration	Availability Zones
ApacheServerDemoLB	ApacheServerDemoLB-1371290937.us-east-1.	80 (HTTP) forwarding to 80 (HTTP)	us-east-1b, us-east-1d

1 Load Balancer selected

Load Balancer: ApacheServerDemoLB

Description | **Instances** | Health Check | Monitoring | Security | Listeners

Instances

Instance	Name	Availability Zone	Status	Actions
i-afa10bd6	*empty*	us-east-1d	In Service	Remove from Load Balancer
i-d0e202ab	*empty*	us-east-1b	In Service	Remove from Load Balancer
i-d2e202a9	*empty*	us-east-1b	In Service	Remove from Load Balancer
i-a1a10bd8	*empty*	us-east-1d	In Service	Remove from Load Balancer

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1b	2	Yes	Remove from Load Balancer
us-east-1d	2	Yes	Remove from Load Balancer

Listeners, Health Check Status

- You can edit parameters of the Health Check

Description | Instances | Health Check | Security | **Listeners**

The following listeners are currently configured for this load balancer:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port	Cipher	SSL Certificate	Actions
HTTP	80	HTTP	80	N/A	N/A	Remove
HTTP	80	HTTP	80	N/A	N/A	Save

Description | Instances | **Health Check** | Security | Listeners

Ping Target:	HTTP:80/
Timeout:	5 seconds
Interval:	30 seconds
Unhealthy Threshold:	3
Healthy Threshold:	7

Edit Health Check

Manually Add or Remove More Instances

- Hitting + or – sign on the main description page, brings us here:

Add and Remove Instances Cancel

The table below lists all your running EC2 Instances that are not already behind another load balancer or part of an auto-scaling capacity group. Check the boxes in the Select column to select the instances you would like behind this load balancer. If you select instances from an availability zone that isn't already part of the load balancer, we will add it for you.

Warning: Unchecking instances and clicking save will remove these instances from your load balancer.

Add or Remove Instances:

Select	Instance	Name	State	Security Groups	Availability Zone
☑	i-d2e202a9		running	launch-wizard-2	us-east-1b
☑	i-d0e202ab		running	launch-wizard-2	us-east-1b
☑	i-711ea616		running	launch-wizard-2	us-east-1b
☑	i-afa10bd6		running	launch-wizard-2	us-east-1d
☑	i-a1a10bd8		running	launch-wizard-2	us-east-1d

Availability Zone Distribution:

select all | select none

3 instances in us-east-1b
2 instances in us-east-1d

Note: You do not have an equal number of instances in each availability zone. We recommend having the same number of instances in each availability zone because the load balancer will distribute traffic equally across availability zones.

Save

Testing Load Balancer

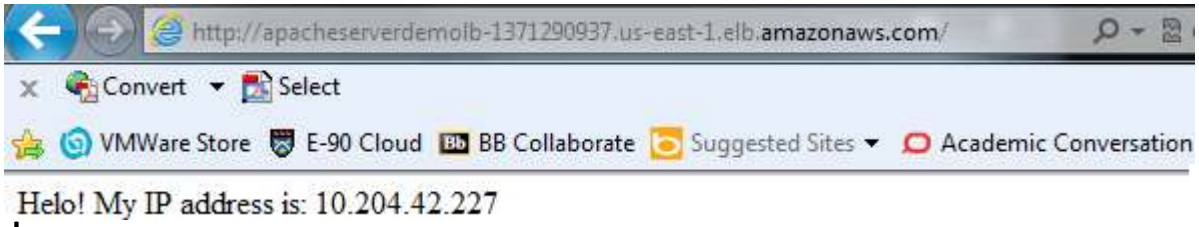
- All of our instances have the same simple PHP “application” which will tell us the IP address of the server that is responding to our request.
- We will open the browser pointing to the URL of the load balancer. In our case it reads:

`ApacheServerDemoLB-2007970259.us-east-1.elb.amazonaws.com`

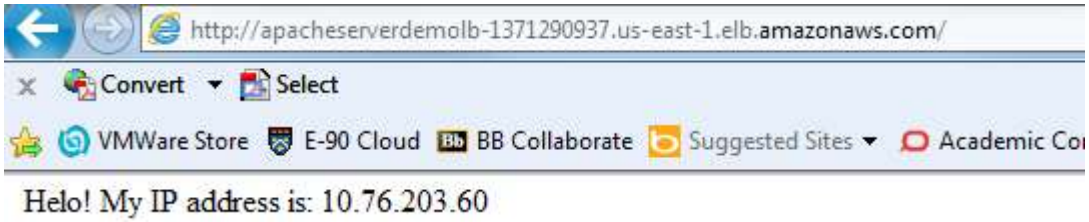
- Every subsequent refresh of our page sends a new request to the load balancer.
- Load balancer distributes those requests among healthy servers. Several subsequent requests should all answered by different servers, with different IP addresses.

Connect to the Load Balancer

- In your Browser, place Load Balancer's URL, http://apacheserverdemolb-1371290937.us-east-1.elb.amazonaws.com/



- Refresh



- Refresh again

- Load Balancer sends your request to a different server every time

Domain Name

- Our load balancer had this long name:
- http://apacheserversdemolb-1371290937.us-east-1.elb.amazonaws.com/
- We paste the DNS name generated by Elastic Load Balancing into the address field of a Web browser to connect to load balancer.
- We would prefer to use a user-friendly domain name, such as `www.example.com`, instead of the load balancer DNS name.
- We can create a custom domain name and then associate the custom domain name with the load balancer DNS name. When a request is placed to the load balancer using the custom domain name that we created, it resolves to the load balancer DNS name.
- To use a custom domain name for our load balancer instance, we have to first register our domain name with a Registrars service provider. List of registrars can be found at ICANN.org
- Amazon AWS is not registrars service provider.

What is Route 53

- Route 53 is a scalable Domain Name System (DNS) service.
- Route 53 provides secure and reliable routing to the infrastructure that uses Amazon Web Services (AWS) products, such as Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing, or Amazon Simple Storage Service (Amazon S3).
- We can also use Route 53 to route users to the infrastructure outside of AWS.
- Route 53 is an authoritative DNS service, meaning it translates friendly domains names like `www.example.com` into IP addresses like `192.0.2.1`.
- Route 53 responds to DNS queries using a global network of authoritative DNS servers, which reduces latency.

Amazon Route 53

- Create a domain using Amazon Route 53 as the DNS service Amazon Route 53 stores information about your domain in a hosted zone.
- A hosted zone is an Amazon Route 53 concept that is similar to a zone file on a DNS name server. Like a zone file, a hosted zone contains information about your domain name, including the subdomain names within the domain and mappings between names and IP addresses
- You'll use Amazon Route 53 to create a hosted zone for your domain (for example, example.com), and then create alias resource record sets.
- An alias resource record set contains a pointer to a resource record set that contains your DNS resource records. For example, an alias resource record set for your domain, *example.com*, can point to the DNS name of your Elastic Load Balancing load balancer instance `apacheserversdemolb-2007970259.us-east-1.elb.amazonaws.com`
- After creating a hosted zone, you can also create alias resource record sets to associate subdomain names with your Elastic Load Balancing instance.

AWS Auto Scaling

AWS Auto Scaling

- Auto Scaling is a web service designed to launch or terminate EC2 instances automatically based on user-defined policies, schedules, and health.
- We scale up, meaning increase the number of instances that respond to requests of users, when there is a spike in demand.
- Increased number of instances maintain performance, i.e. responsiveness of the system. Rather than waiting for one server to respond to many users, users' requests are spread over several or many servers.
- We scale down when the demand decreases.
- We scale down to keep costs down.
- Scaling is seamless and automated. Auto Scaling can be combined with other Amazon services like CloudWatch and/or Elastic Load Balancing to provide a flexible and configurable solution to keep up with the demand.

Benefits of AWS Auto Scaling

- **Elastic Capacity**
  - Automatically add compute capacity when application usage rises and remove it when usage drops.
- **Ease of Use**
  - Manage your instances spread across either one or several Availability Zones as a single collective entity, using simple command line tools or programmatically via an easy-to-use web service API.
- **Cost Savings**
  - Save compute costs by terminating underused instances automatically and launching new instances when you need them, without the need for manual intervention.
- **Geographic Redundancy and Scalability**
  - Distribute, scale, and balance applications automatically over multiple Availability Zones within a region.
- **Easier Maintenance**
  - Replace lost or unhealthy instances automatically based on predefined alarms and thresholds.
- **Scheduled Actions**
  - Schedule scaling actions for future times and dates when you expect to need more or less capacity.

Auto Scaling Group, Launch Configuration

- Auto scaling group is the fleet of instances that the auto scaling service manages.
- We define a Launch Configuration for auto scaling groups, telling the service, the AMI id of instances in a scaling group as well as: key pairs, security group, block device mapping, etc.
- Auto Scaling can be set:
  - By schedule, according to a defined time and date, when demand follows a predictable pattern
  - By policy, based on some metrics so dynamically scales up and down, according to triggers
  - To maintain current scaling level, using health checks to terminate and replace instances with degraded performance

Triggers

- 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.
- In most cases, we need two triggers—one trigger for scaling up and another for scaling down.
- For example, if we want to scale up when CPU usage increases to 80 percent, we need to configure a CloudWatch alarm and create an Auto Scaling policy. The alarm detects when the CPU usage has reached 80 percent and sends a message to Auto Scaling. Auto Scaling determines what to do by using the instructions in the scaling policy.
- If we want to scale down when CPU usage decreases to 40 percent, you need a second trigger. In other words, we need to configure a separate CloudWatch alarm to detect the 40 percent threshold and create a separate Auto Scaling policy that scales down.

Policies

- A *policy* is a set of instructions for Auto Scaling that tells the service how to respond to CloudWatch alarm messages. You can configure a CloudWatch alarm to send a message to Auto Scaling whenever a specific metric has reached a triggering value. When the alarm sends the message, Auto Scaling executes the associated policy on an Auto Scaling group to scale the group up or down.

CloudWatch

- AWS CloudWatch lets you monitor several different EC2 server performance metrics in real time, including…
  - CPU Utilization (%)
  - Memory Utilization (%)
  - Network Out Utilization (MB)
  - Memory Used (MB)
  - Memory Available (MB)
  - Swap Utilization (%)
  - Swap Used (MB)
  - Disk Space Utilization (%)
  - Disk Space Used (GB)
  - Disk Space Available (GB)
- …and many more. It’s up to you what to monitor, but the metrics most useful for knowing when you should scale up and add another server or scale down by terminating a server are probably CPU utilization, memory utilization or network utilization.

Auto Scaling by Schedule

- When auto scaling by schedule, we define scheduled actions with a time and date and how many instances should be up in the group.
- We can also schedule recurrent actions in the same fashion as the `cron` job in Unix.
- For example, we can specify that the group will scale down to 2 instances every Sunday, and then scale up to 7 instances every Monday.
- Scheduled scaling always specify an absolute number of desired instances (capacity).

Auto Scaling by Policy

- When auto scaling by policy, we configure auto scaling policies, or how the group must be scaled (or scaling actions).
- Policies tell whether the scaling should go up or down and can specify the new group size as an absolute number, an increment, or a percentage of the current group size.
- For example, it is possible to adjust the number of instances by increments of 10%, or by 3 instances every time, or set the capacity to 5 instances.
- Auto Scaling provides command to create policies.
- We have to wire trigger to implement policies.
- We could use CloudWatch, AWS Monitoring service, which provides many types of metrics for each of AWS services.
- For example, for EC2 instances, we can set alarms according to CPU utilization, bytes read or written to disk, etc.
- When CPU utilization reaches a configured threshold, CloudWatch sends a notification to a configured SNS topic. Based on this notification, you can execute an auto scaling policy.

Features

- Speeds of scaling up and scaling down do not have to be the same
  - For example, we can define a scale up condition to increase EC2 capacity by 10% and a scale down condition to decrease it by 5%.
- Selection of instances to terminate on scale down
  - When selecting an instance to terminate Auto Scaling attempts to preserve instances with the current launch configuration, and will terminate instances with older launch configuration.
  - Auto Scaling will terminate the instance running for the longest portion of a billable instance-hour (without running over). We can configure a policy to terminate the oldest or newest instance instead.
  - We target a specific instance for immediate termination with TerminateInstanceInAutoScalingGroup API.

Features

- Cannot scale up beyond your EC2 limit.
- Auto Scaling will not allow you to delete an Auto Scaling Group if it contains running Amazon EC2 instances.
- You have to empty the Auto Scaling Group by setting its size to 0 using the as-set-desired-capacity command from the command line.
- You can safely delete your Auto Scaling Group once it is empty

Use Cases for Auto Scaling

- Typically we use Auto Scaling when we can predict that the demand for our application will peak at a certain time.
- For example, if we have a Black Friday promotion, that day will have a greater volume of requests than the rest of the month.
- Similarly, when the stock market opens and closes, there are pronounced spikes in the number of transactions as compared with the rest of the day.
- Above examples can be solved with Auto Scaling by Schedule, where we can define the date and time when we want to scale up, and similarly the date and time when we want to scale down.

Demonstration Application

- We have created a simple application where the client side acts as a producer, sending messages to an SQS queue.
- The server side is the consumer of messages from this queue.
- The client is configured to send messages faster than the server side consumer can read them.
- We use `Thread.sleep()` in between message reads on the server side to simulate slow response to some “complicated processing” of messages in the server.
- We will start our auto scaling group with a small number of servers and use auto scaling to add new instances when server responses slow down.

`ServerSideConsumer.java` in `AutoScaling` Project

- We open a AWS Java project with a SQSSample application, which we then discard.

Project Explorer

- AutoScaling
  - src
    - edu.hu.as.server
      - ServerSideConsumer.jav
    - AwsCredentials.properties
  - JRE System Library [JavaSE-1.7]
  - AWS SDK for Java [1.3.22]

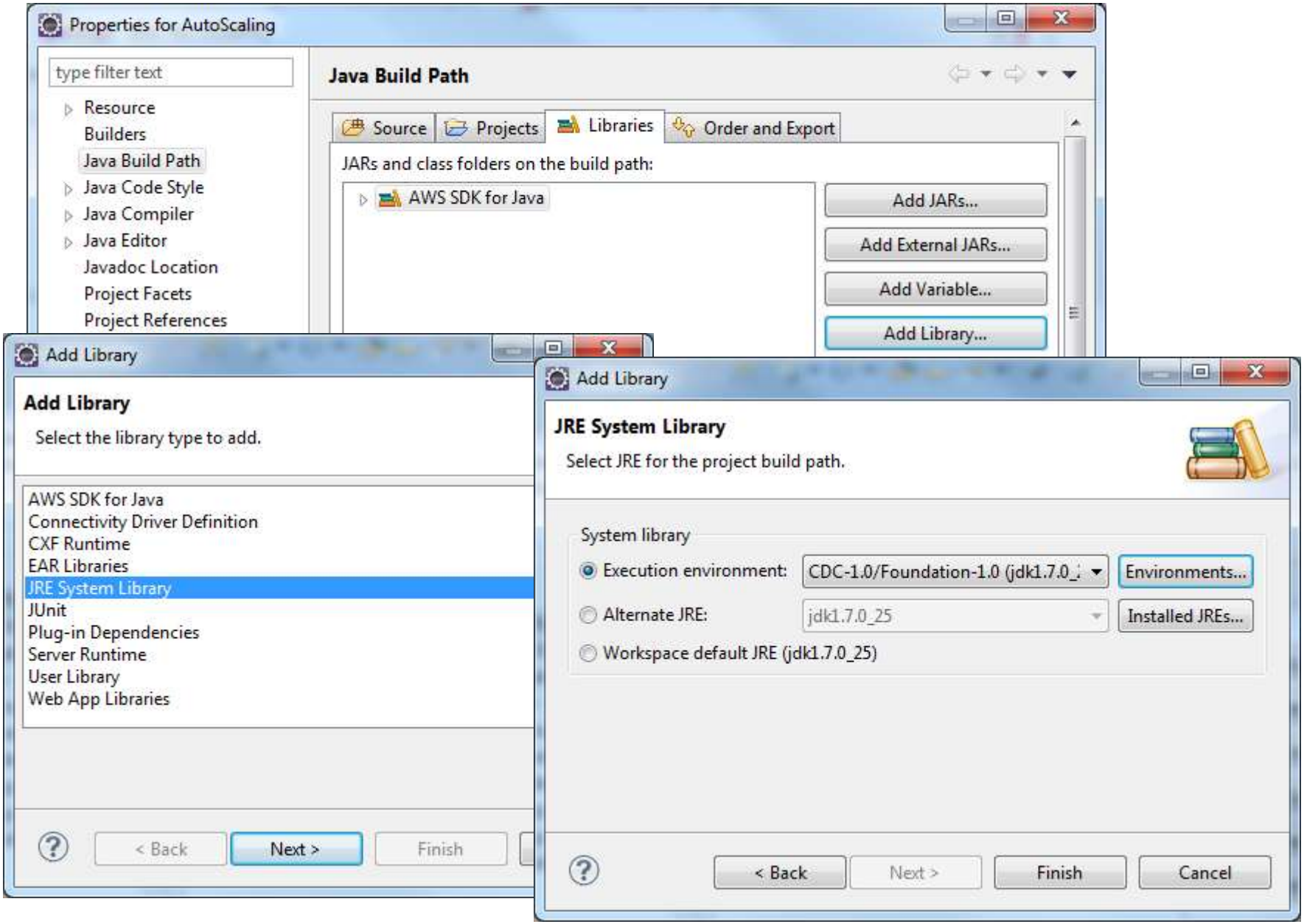
ServerSideConsumer.java | AwsCredentials.properties

```java
import java.util.List;

/**
 * reads a message from each queue defined for user, waiting CONSUMER_INTERVAL_ms miliseconds
 * between each read
 *
 * you can specify consumer_interval_ms from command line:
 * $> java consumer <interval in ms>
 */
public class ServerSideConsumer {

    static final int CONSUMER_INTERVAL_ms = 5000;

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

    /* replace with your access key and secret key
     *  BasicAWSCredentials(String accessKey, String secretKey)
     */
    AmazonSQS sqs = new AmazonSQSClient( new BasicAWSCredentials( "AKIAJXHJT7JYTRBNRQ4TQ", "gUlatSWI

        int consumer_interval_ms = args.length > 0 ? Integer.valueOf( args[0] ) : CONSUMER_INTERVAL_

        System.out.println("==========================================");
        System.out.println("CONSUMER");
        System.out.println("==========================================\n");

        try {
```

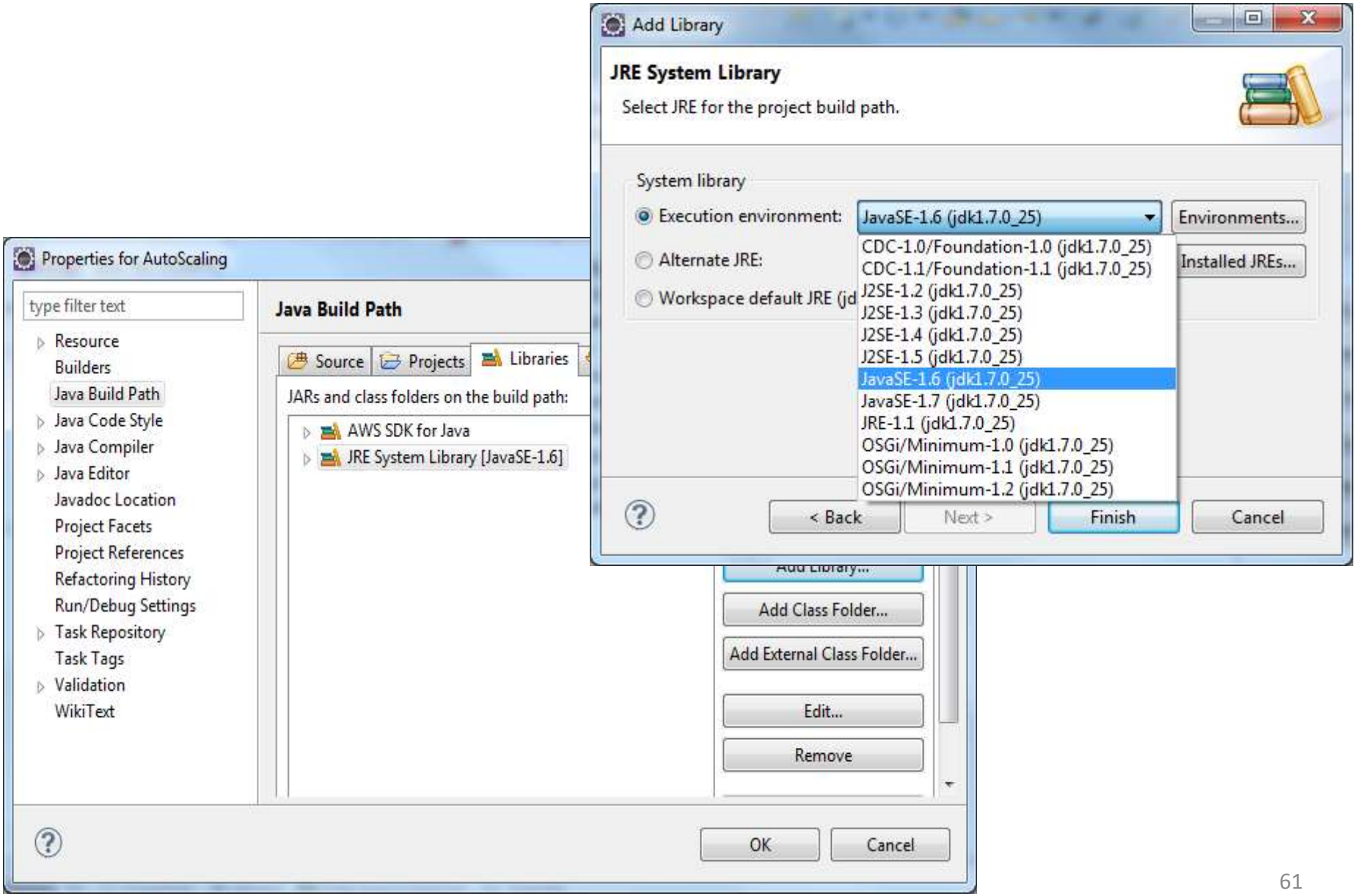
Change Java Runtime Environment

- Select Project > Properties > Libraries
- If you see JRE System Library [JavaSE-1.6] you are fine
- If you see JavaSE-1.7, highlight the library and remove it.
- Go to Add Library > JRE System Library
- Check Execution Environment and from the drop down select JavaSE-1.6(jre7), Click Finish, OK.
- The issue is that Amazon Linux runs on Java 6 and yum repositories are not letting you update to Java 7. If the executable jar we will make has Java 7 you will get version mismatch errors.
- You can brute force install Java 7 if you need it for some other reasons.

Remove JRE1.7 add JRE1.6



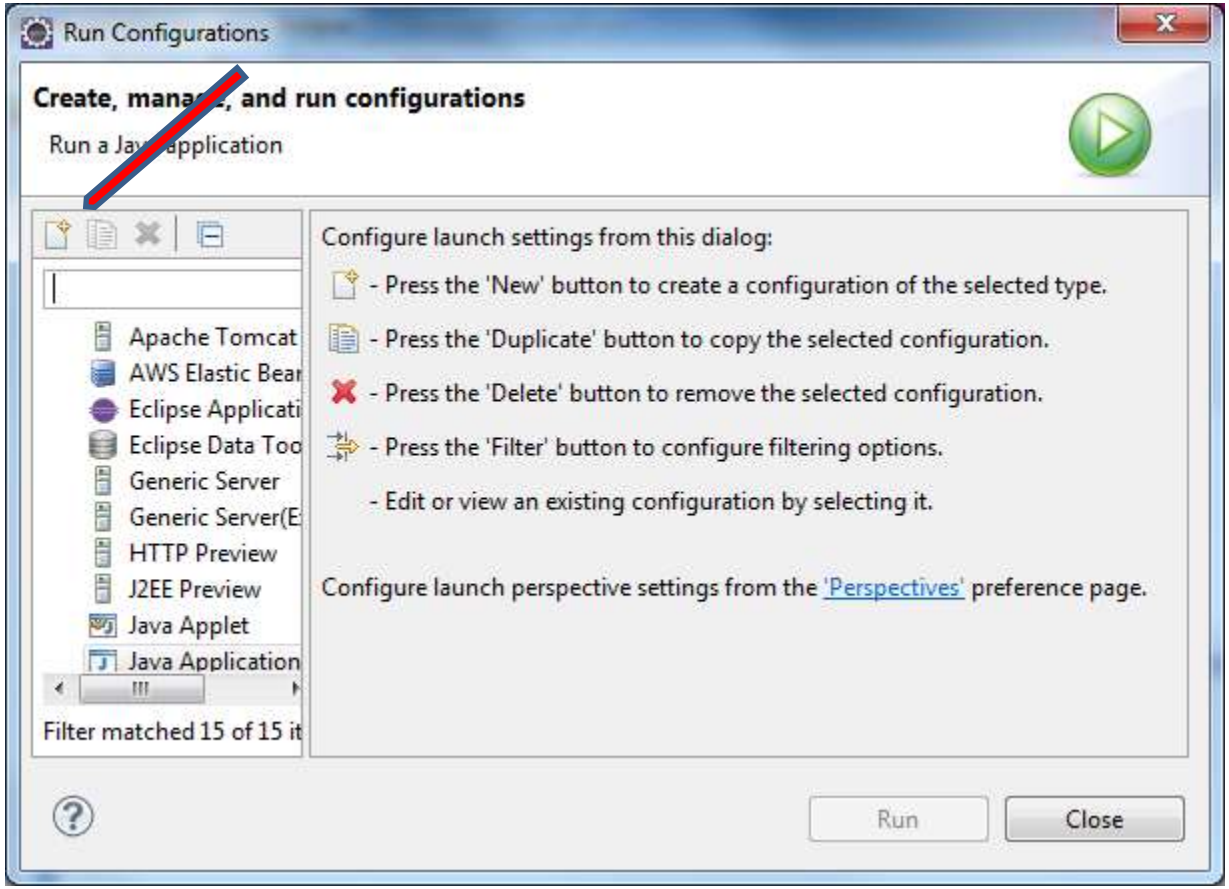
Adding JavaSE-1.6 Runtime Environment



Creating a Runnable Java Jar

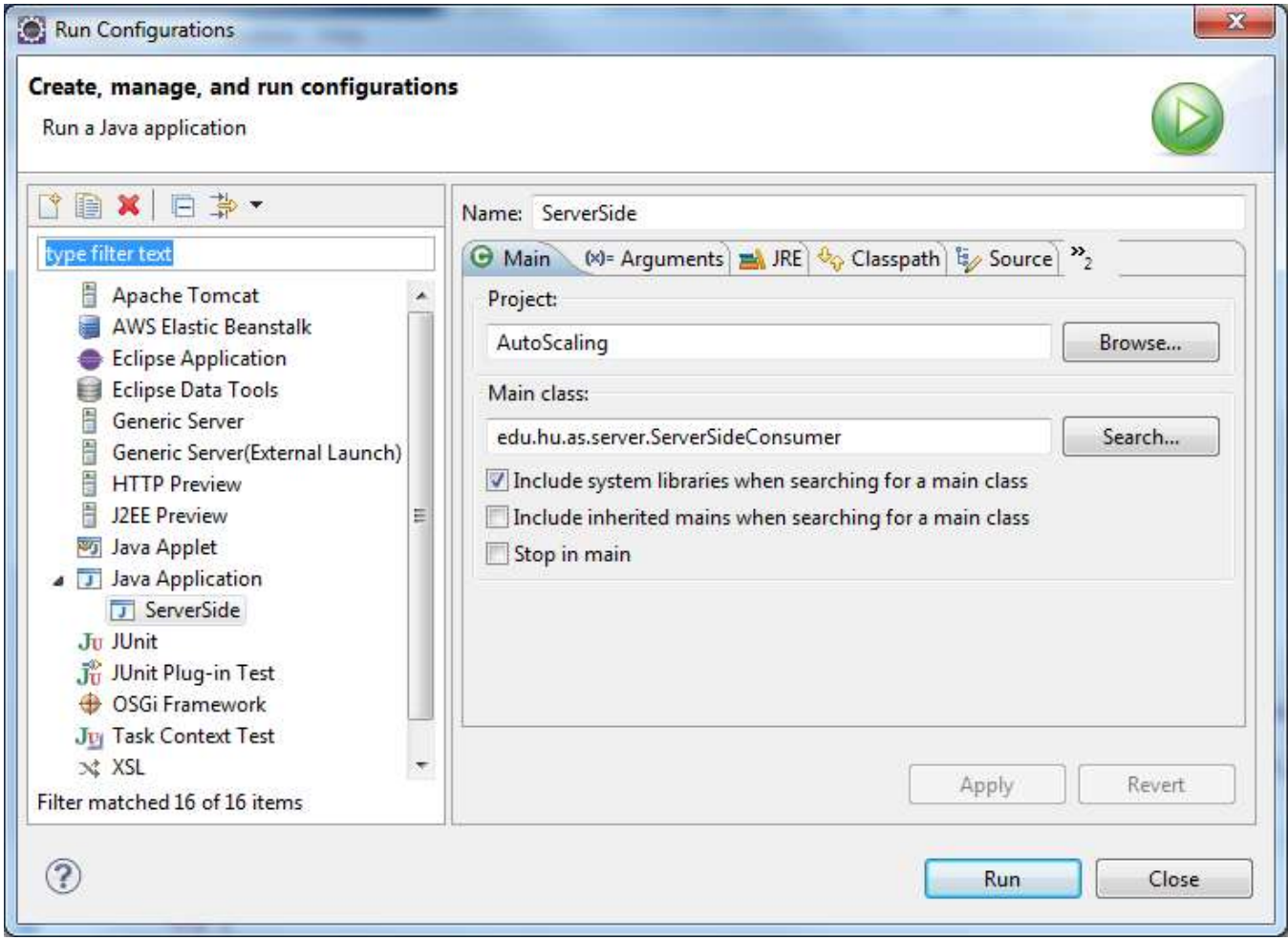
- Right click the class you want to package as an executable jar.
- Select `Run As` and then `Run Configurations`.
- The first time you run this option you will get a screen like this:

- Press the New button above the filter field on the left side.
- A modified widget will appear.



Setup Run Configuration

- Add project, if not there, add Class, i.e. `ServerSideConsumer`
- Change Name , e.g. `ServerSide`
- Check `Include system libraries.`
- When you hit `Run`, you will see output of your file. There were no messages consumed (yet).
- Next, export you project as a runnable jar.

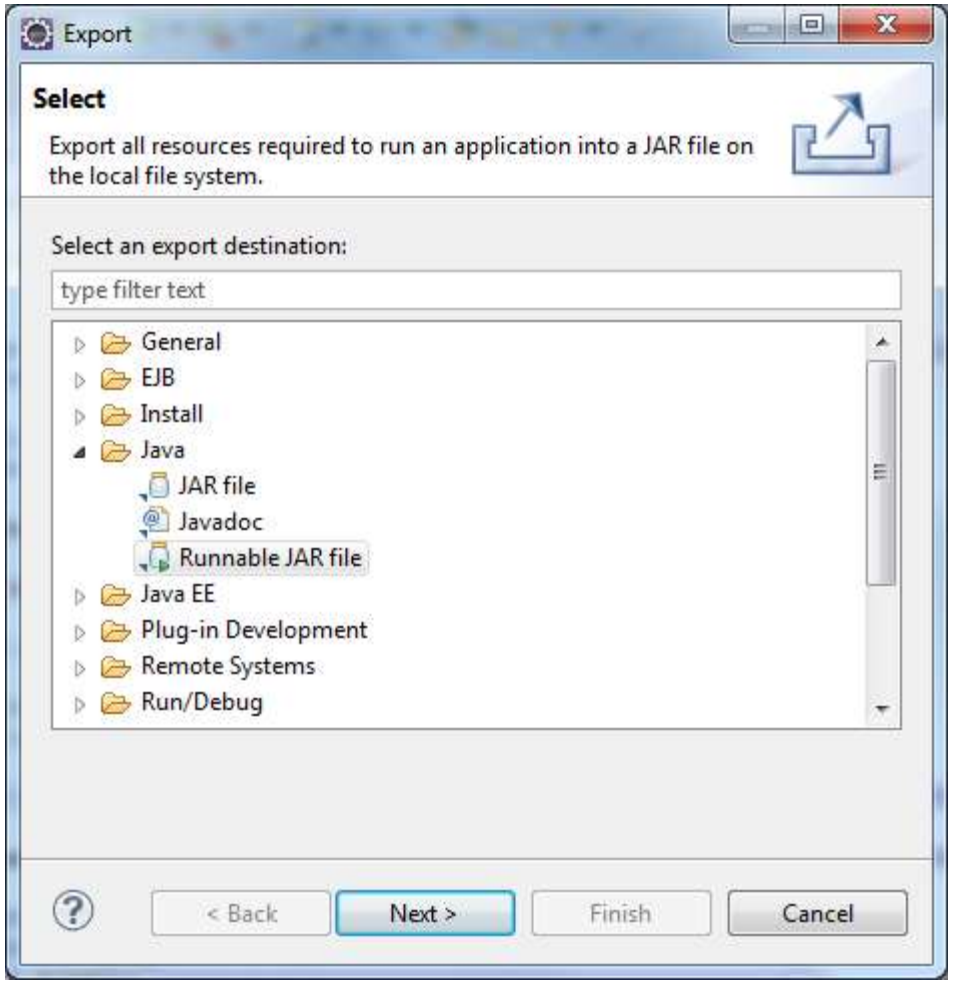


Export Project as Runnable Jar

- Right click on the project, Select

  `Export > Java > Runnable JAR file`

- Select `Next`



Launch Configuration & Export Destination

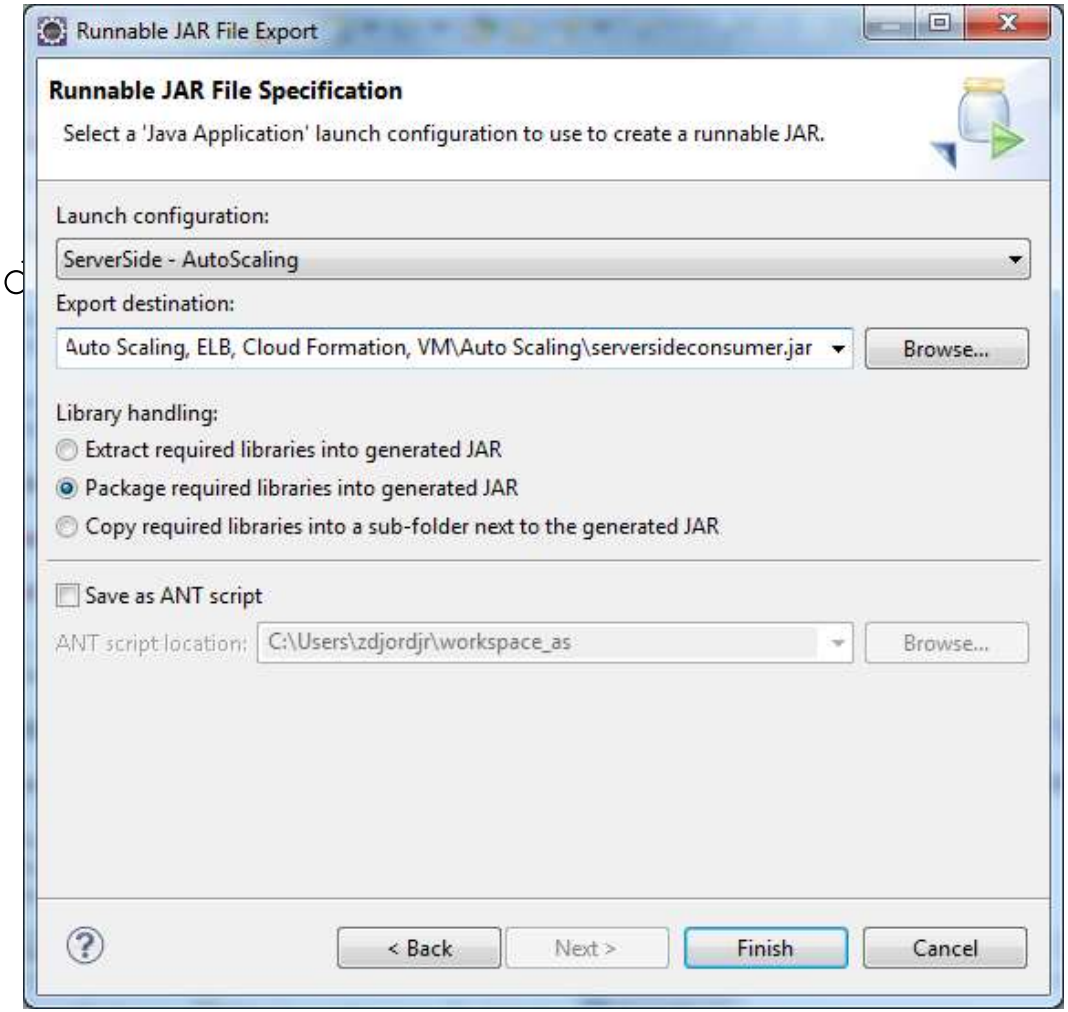
- Name exported file something, e.g.

`serversideconsumer.jar`

- Check

`Package required libraries into generated`

- Save the file to a known directory.
- You will move that file to the server side to the directory `/hom/ec2-user/as` using `scp` command



Modify `/etc/rc.local`, scp Executable jar

```
$ ssh -i ec2_hu.pem ec2-user@ec2-23-23-32-28.compute-1.amazonaws.com
[ec2-user@ip-10-64-27-21 ~] $ mkdir /home/ec2-user/as

[ec2-user@ip-10-64-27-21 ~]$ 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
java -jar /home/ec2-user/as/serversideconsumer.jar

:wq
```

- Move your executable jar to the server side:

```
$ scp -i hu0906.pem serversideconsumer.jar
ec2-user@ec2-23-23-32-28.compute-1.amazonaws.com:/home/ec2-user/as
serversideconsumer.jar                    100%   12MB   3.0MB/s   00:04
```

Add Client Classes:
ClientSideProducer, ClientSideQMonitor

Project Explorer

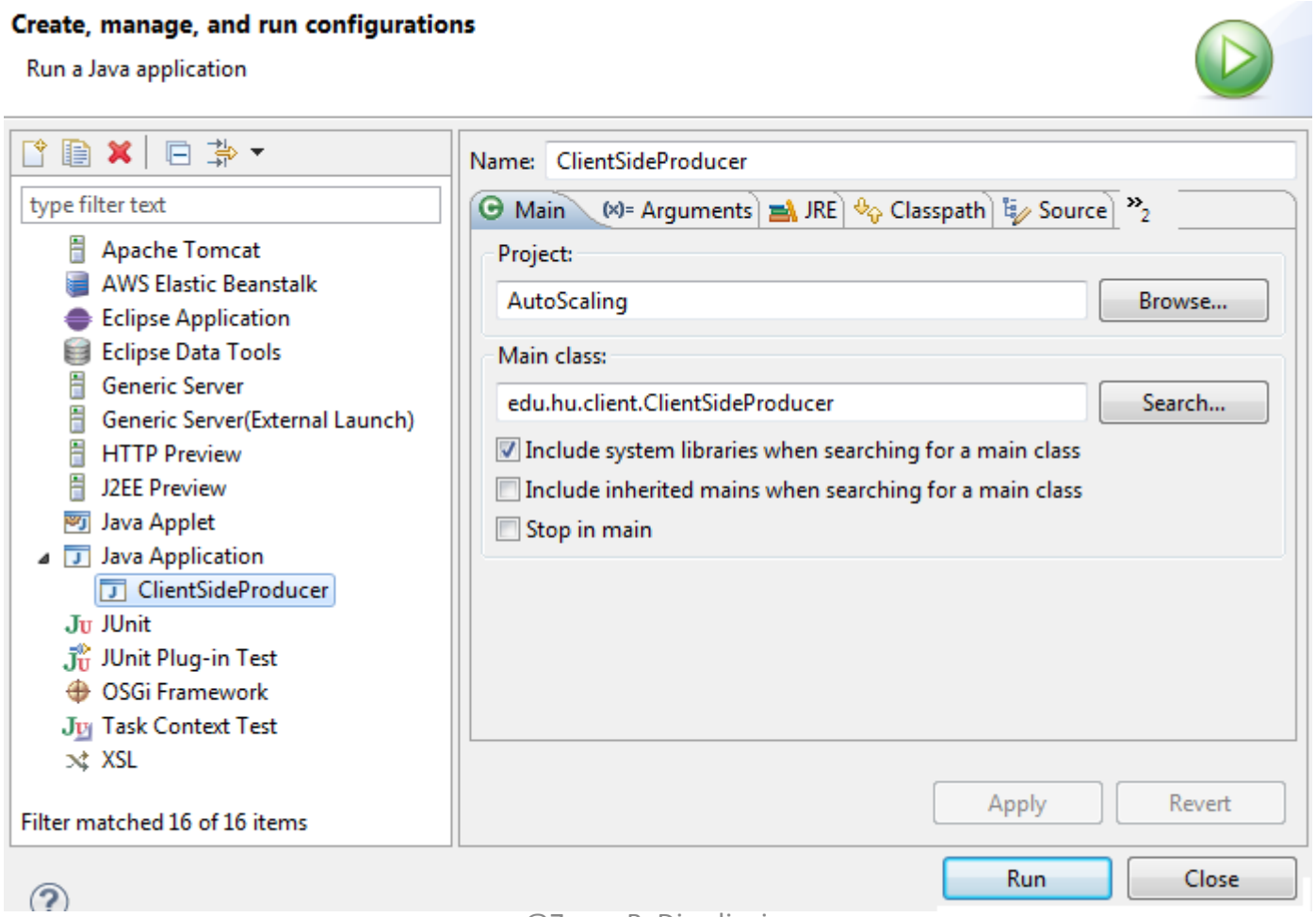
- AutoScaling
  - src
    - edu.hu.as.server
      - ServerSideConsumer.java
    - edu.hu.client
      - ClientSideProducer.java
      - ClientSideQMonitor.java
    - AwsCredentials.properties
  - JRE System Library [JavaSE-1.7]
  - AWS SDK for Java [1.3.22]

ServerSideConsu... | AwsCredentials.pr... | ClientSideProdu... | ClientSideQMonit...

```java
/**
 * prints current time and estimated length of queue (visible messages)
 * every WAIT_ms miliseconds, ad infinitum.
 */
public class ClientSideQMonitor {

    static final int WAIT_ms = 5000;

    public static void main(String[] args) throws Exception {
        AmazonSQS sqs = new AmazonSQSClient( new BasicAWSCredentials( "AKIAJXHJT7JYTRQ

        int wait_ms = args.length > 0 ? Integer.valueOf( args[0] ) : WAIT_ms;

        System.out.println("==========================================");
        System.out.println("Monitoring SQS");
        System.out.println("==========================================\n");

        try {
        // get queue length, provide know QUEUE URL of your Queue
            GetQueueAttributesRequest attrReq = new GetQueueAttributesRequest(
             "https://queue.amazonaws.com/951414139794/AutoScalingTestQueue"); //"Queu
            attrReq.setAttributeNames( Arrays.asList(
                    "ApproximateNumberOfMessages",
                    "ApproximateNumberOfMessagesNotVisible"
                    ));
```

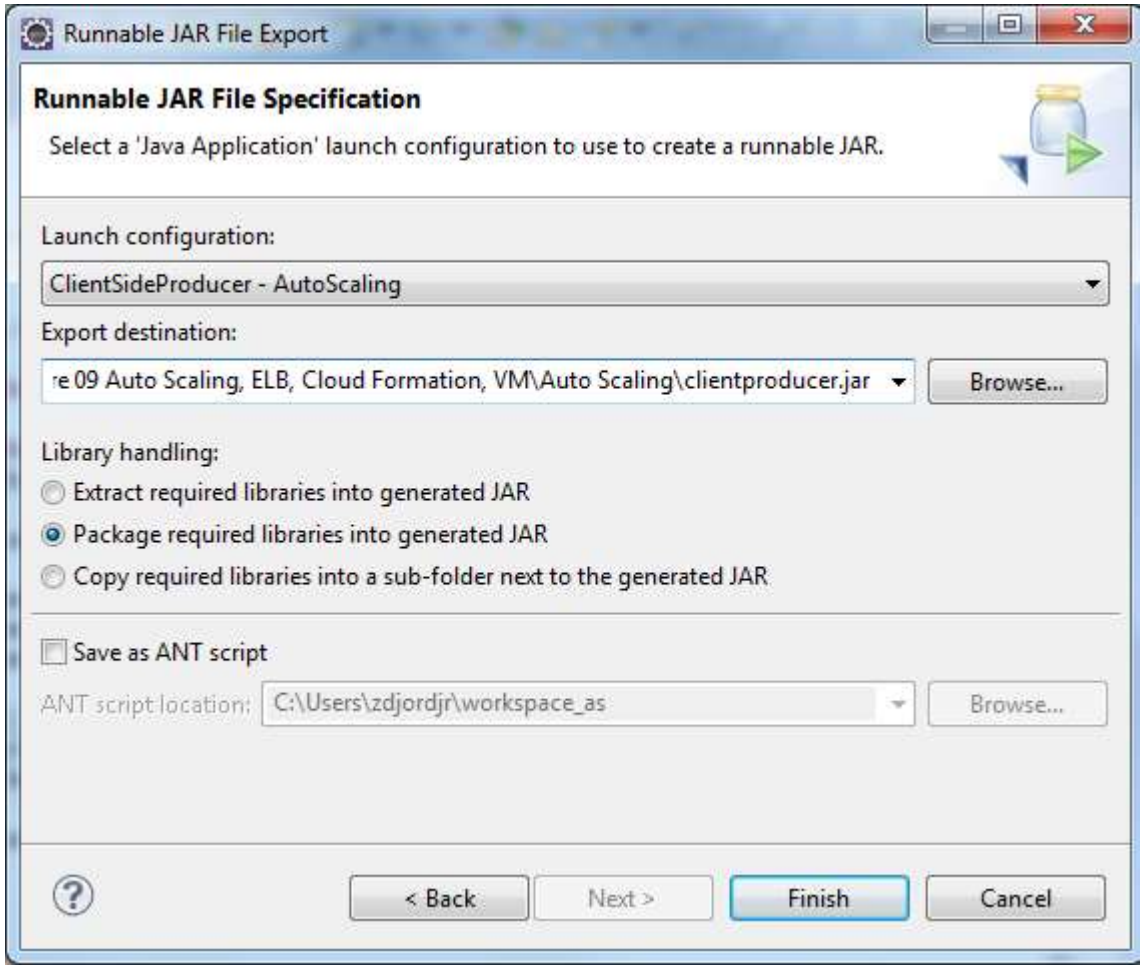
Must create new Run Configurations

- You must create new run configuration with `ClientSideProducer` as the main class and create new executable jar, e.g. `clientproducer.jar`
- Note that you must change the name of the configuration



Again Export > Java JAR file

- Name new export `clientproducer.jar`



Test Server Side Consumer

- You can create an SQS queue and enter its URL in the `ClientProducer.java.` If you run that `ClientProducer` class as a Java application it will send a bunch of messages to the queue.
- On the server side, you could go to `/home/ec2-user/as` directory and type:

```
$ chmod 777 serversideconsumer.jar
$ java –jar serversideconsumer.jar
```

- You will see the printout acknowledging that the class `ServerSideConsumer` did read messages accumulated in the queue.
- The server side instance is ready and can be cloned.

Set Up AutoScaling CLI Environment

- Download AutoScaling tools from http://aws.amazon.com/developertools/2535
- Expand the ZIP file, e.g. `C:\AWS\AutoScaling-1.0.61.3`
- Create the following environmental variables:
- **JAVA_HOME**=`C:\Program Files\Java\jdk1.7.0_07`
- **AWS_AUTO_SCALING_HOME=**`C:\AWS\AutoScaling-1.0.61.3`
- Add `AWS_AUTO_SCALING_HOME\bin` to your PATH
- set PATH=%PATH%;%AWS_AUTO_SCALING_HOME%\bin
- **Do not set** `AWS_CREDENTIAL_FILE` **variable.**
- AS CLI will use your `EC2_CERT` and `EC2_PRIVATE_KEY` variables
- Do not set your `REGION` to `us-east-1` if you are on the East Cost. Change to appropriate region, if you are somewhere else

`AWS_AUTO_SCALING_URL=https://`*`autoscaling.us-west-1.amazonaws.com`*

List Auto Scaling CLI commands, type: `as-cmd`

```
as-create-auto-scaling-group
    Create a new Auto Scaling group.
as-create-launch-config
    Creates a new launch configuration.
as-create-or-update-tags
    Create or update tags.
as-delete-auto-scaling-group
    Deletes the specified Auto Scaling group.
as-delete-launch-config
    Deletes the specified launch configuration.
as-delete-notification-configuration
    Deletes the specified notification configuration.
as-delete-policy
    Deletes the specified policy.
as-delete-scheduled-action
    Deletes the specified scheduled action.
as-delete-tags
    Delete the specified tags
```

List Auto Scaling CLI commands, type: `as-cmd`

```
as-describe-adjustment-types
    Describes all policy adjustment types.
as-describe-auto-scaling-groups
    Describes the specified Auto Scaling groups.
as-describe-auto-scaling-instances
    Describes the specified Auto Scaling instances.
as-describe-auto-scaling-notification-types
    Describes all Auto Scaling notification types.
as-describe-launch-configs
    Describes the specified launch configurations.
as-describe-metric-collection-types
    Describes all metric colle... metric granularity types.
as-describe-notification-configurations
    Describes all notification...given Auto Scaling groups.
as-describe-policies
    Describes the specified policies.
as-describe-process-types
    Describes all Auto Scaling process types.
```

List Auto Scaling CLI commands, type: `as-cmd`

```
as-describe-process-types
    Describes all Auto Scaling process types.
as-describe-scaling-activities
   Describes a set of activit...ties belonging to a group.
as-describe-scheduled-actions
   Describes the specified scheduled actions.
as-describe-tags
   Describes tags
as-describe-termination-policy-types
   Describes all Auto Scaling termination policy types.
as-disable-metrics-collection
    Disables collection of Auto Scaling group metrics.
as-enable-metrics-collection
    Enables collection of Auto Scaling group metrics.
as-execute-policy
    Executes the specified policy.
```

List Auto Scaling CLI commands, type: `as-cmd`

```
as-put-notification-configuration
    Creates or replaces notifi...or the Auto Scaling group.
as-put-scaling-policy
    Creates or updates an Auto Scaling policy.
as-put-scheduled-update-group-action
    Creates or updates a scheduled update group action.
as-resume-processes
    Resumes all suspended Auto... given Auto Scaling group.
as-set-desired-capacity
     Sets the desired capacity of the Auto Scaling group.
as-set-instance-health
     Sets the health of the instance.
as-suspend-processes
     Suspends all Auto Scaling ... given Auto Scaling group.
as-terminate-instance-in-auto-scaling-group
     Terminates a given instance.
as-update-auto-scaling-group
     Updates the specified Auto Scaling group.
```

Help for Individual commands `--help`

```
c:\AWS\hu>as-create-auto-scaling-group --help
SYNOPSIS
   as-create-auto-scaling-group
        AutoScalingGroupName  --availability-zones  value[,value...]
        --launch-configuration  value  --max-size  value  --min-size  value
       [--default-cooldown  value ] [--desired-capacity  value ]
       [--grace-period  value ] [--health-check-type  value ] [--load-balancers
       value[,value...] ] [--placement-group  value ] [--tag  "k=value,
       [id=value], [t=value], [v=value], [p=value]" [ --tag "k=value,
       [id=value], [t=value], [v=value], [p=value]" ...] ]
       [--termination-policies  value[,value...] ] [--vpc-zone-identifier  value
       ]  [General Options]
DESCRIPTION
       Creates a new Auto Scaling group with a specified name and other attributes.
ARGUMENTS
   AutoScalingGroupName
       User-supplied Auto Scaling group identifier which will uniquely identify
       the Auto Scaling group. You can also set this value using
       "--auto-scaling-group". Required.
EXAMPLES
 Create group 'test-group-3' with all parameters
    $PROMPT3> as-create-auto-scaling-group test-group-3 --launch-configuration test-
config-3   --availability-zones us-east-1a --min-size 0 --max-size 1 --default-cooldown
180               --load-balancers  test-lb-2 --health-check-type ELB --grace-period
240 --tag "k=stack,v=Production,p=true" --tag "k=Owner,v=TeamA,p=true"
```

Create new AMI using your server instance

- Right click on the running instance, select

`Create Image( EBS AMI)`, New AMI `ami-23643b4a` is created

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status
	ServerSideConsumerAMI	ami-037c236a	951414139794/...	951414139794	Private	available
	CopyAWSLinuxWithApache	ami-954817fc	951414139794/...	951414139794	Private	available

Image: ami-037c236a

Details | Permissions | Tags

Edit

AMI ID	ami-037c236a	AMI Name	ServerSideConsumerAMI
Owner	951414139794	Source	951414139794/ServerSideConsumerAMI
Status	available	State Reason	-
Platform	Other Linux	Architecture	i386
Image Type	machine	Description	AWS Linux AMI with a Queue Consumer application
Root Device Name	/dev/sda1	Root Device Type	ebs

Create a Launch Configuration

- The following command creates a launch configuration `as_lc` for my AMI `ami-23643b4a`.
- Launch configuration specifies the type of EC2 instance that you want Auto Scaling to create

```
C:\> as-create-launch-config lc3  --image-id ami-23643b4a
--instance-type m1.small
```

- Auto Scaling returns confirmation:

```
OK-Created launch config
```

- Based on this launch configuration and a load balancer we will create an auto scaling group.
- Load Balancer is important since it will examine the health of instances in the group and instruct the group to perhaps replace not-functioning members.

MyLoadBalancer before creation of auto scaling group

Load Balancer Name	DNS Name	Port Configuration	Availability Zones
MyLoadBalancer	MyLoadBalancer-346573099.us-east-1.elb.ama	80 (HTTP) forwarding to 80 (HTTP)	us-east-1b, us-east-1d

1 Load Balancer selected

Load Balancer: MyLoadBalancer

Description | **Instances** | Health Check | Monitoring | Security | Listeners

Instances

Instance	Name	Availability Zone	Status	Actions
No records found.				

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1b	0	No (why?)	Remove from Load Balancer
us-east-1d	0	No (why?)	Remove from Load Balancer

- Before creation of an auto scaling group we had one running instance

Filter: Running instances | All instance types | Search Instances

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-7b95fb03	m1.small	us-east-1d	running	2/2 check...	None	ec2-54-226-190-215.co...

Create Auto Scaling Group

- We anticipate that our group will vary in size from 2 to 10 instances and perhaps has an optimal size of 4:

```
c:\> as-create-auto-scaling-group as_gp3 --launch-configuration lc3 --
availability-zones us-east-1b, us-east-1d --min-size 2 --max-size 10 --
desired-capacity 4 –load-balancers MyLoadBalancer --health-check-type
ELB --grace-period 100

OK-Created AutoScalingGroup
```

- To examine which groups and with which parameters we have, we ask

```
c:\AWS\hu> as-describe-auto-scaling-groups --headers
AUTO-SCALING-GROUP  GROUP-NAME  LAUNCH-CONFIG  AVAILABILITY-ZONES
MIN-SIZE  MAX-SIZ
E  DESIRED-CAPACITY  TERMINATION-POLICIES
AUTO-SCALING-GROUP  as_gp3      lc3            us-east-1b,us-east-1d  2
10
   4                 Default
INSTANCE  INSTANCE-ID  AVAILABILITY-ZONE  STATE      STATUS   LAUNCH-
CONFIG
INSTANCE  i-34385150   us-east-1b         Pending    Healthy  lc3
INSTANCE  i-24569e43   us-east-1d         InService  Healthy  lc3
INSTANCE  i-26569e41   us-east-1d         InService  Healthy  lc3
INSTANCE  i-36385152   us-east-1b         InService  Healthy  lc3
```

Creation of `as_gp3` Group Resulted in 4 Instances

- If we examine EC2 Dashboard we will see that creation of the auto scaling group `as_gp3` with availability zones `us-east-1b` and `us-east-1d` resulted in creation of 4 (desired) new instances in 2 in each zone. Those instances had instance ids:

```
i-34385150   us-east-1b
i-24569e43   us-east-1d
i-26569e41   us-east-1d
i-36385152   us-east-1b
```

tances ▾ All instance types ▾ Search Instances

Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
i-24569e43	m1.small	us-east-1d	running	Initializing	None	ec2-54-205-76-49.comp...
i-26569e41	m1.small	us-east-1d	running	Initializing	None	ec2-54-242-251-116.co...
i-34385150	m1.small	us-east-1b	running	Initializing	None	ec2-23-22-190-121.com...
i-36385152	m1.small	us-east-1b	running	Initializing	None	ec2-54-227-130-255.co...
i-7b95fb03	m1.small	us-east-1d	running	2/2 check...	None	ec2-54-226-190-215.co...

- Load balancer was not forced to acquire any of these instance?

Change parameters of auto scaling group `as_gp3`

- We can change properties of an existing auto scaling group

```
C:\> as-update-auto-scaling-group as_gp3 --health-check-type ELB --grace-period 100
```

- Parameter `health-check ELB` specifies that the auto scaling group will rely on the load balancer for the assessment of the health of the group.
- Parameter `grace-period` tells the group that it does not have to recheck its compliance with the conditions more frequently than once in 100 seconds.

If you need to drop an Auto Scaling Group

- Sometimes you need to drop an auto scaling group and create new one.
- Normally you would first remove instances from the group

```
C:\> as-update-auto-scaling-group as_gp3
--min-size 0 --max-size 0
```

- By the way, you can use that same command to increase the size of a auto scaling group
- Now, you can delete the group

```
C:\> as-delete-auto-scaling-group as_gp3
 Are you sure you want to delete this AutoScalingGroup? [Ny]y
OK-Deleted AutoScalingGroup
```

- Finally, you can delete the launch configuration

```
c:\AWS\hu> as-delete-launch-config as_lc3
    Are you sure you want to delete this launch configuration?
[Ny]y
OK-Deleted launch configuration
```

Auto Scaling Policies

- In a typical situation, we will periodically check the CPU utilization of our servers or some other metrics and based on the observed value issue instruction to the group to increase or decrease its number of instances.
- Objects that hold the information about the action that should take place when we increase or decrease the number of instances in an auto scaling group are called auto scaling policies.
- We need two policies. One will specify how many instances we will add to the group when we want to increase the number of instances and the other will specify how many we will terminate when we want to decrease the number of instances. Numbers do not have to be the same.
- In our case, both the increment and decrement will be 1

Auto Scaling Policies

- Auto scaling policies are created with the CLI command:

```
c:\>as-put-scaling-policy --auto-scaling-group as_gp3      --name
scale-up --adjustment 1 --type ChangeInCapacity --cooldown 100
arn:aws:autoscaling:us-east-
1:951414139794:scalingPolicy:e8dcc907-9b78-4041-97e4-9ee9b7
e00c8a:autoScalingGroupName/as_gp3:policyName/scale-up
c:\>as-put-scaling-policy --auto-scaling-group as_gp3      --name
scale-dn "--adjustment=-1" --type ChangeInCapacity        --
cooldown 100
arn:aws:autoscaling:us-east-
1:951414139794:scalingPolicy:08fb202c-4683-4120-9da4-60c5ca
1238d4:autoScalingGroupName/as_gp3:policyName/scale-dn
```

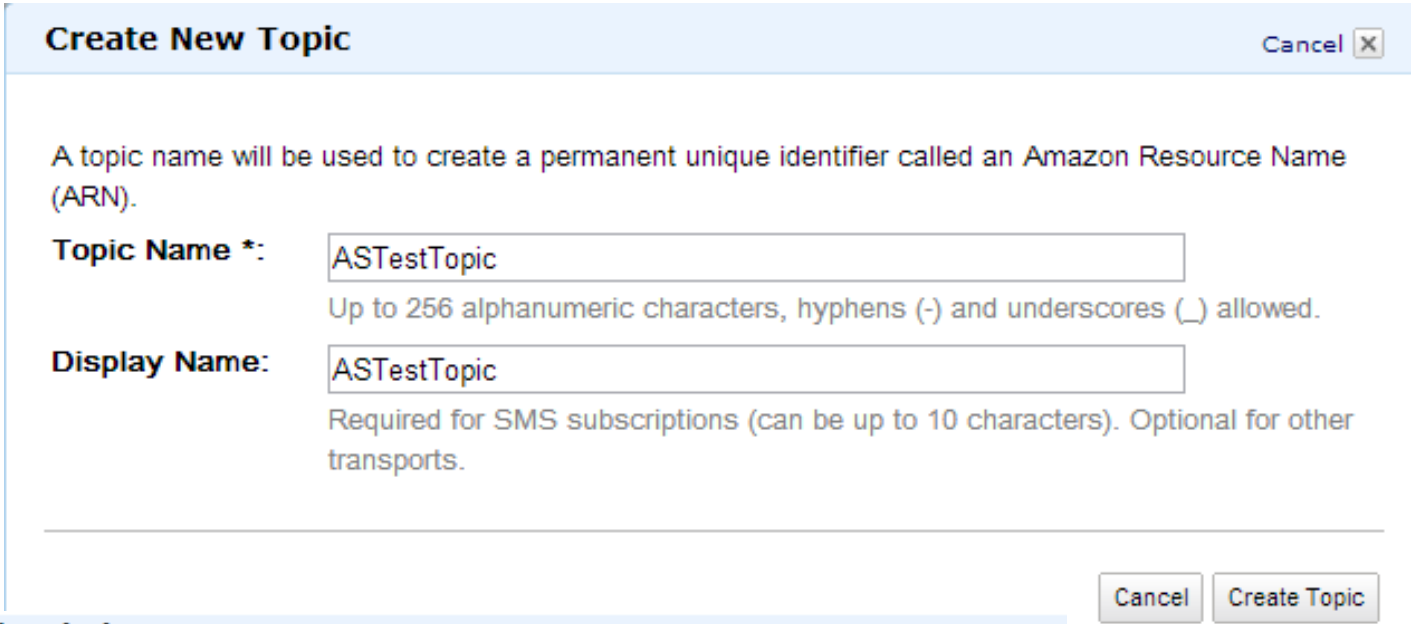
- We created two policies with respective names `scale-up` and `scale-dn`.
- The increment, the adjustment, for the scale-up policy is 1.
- The increment for scale-dn policy is -1. Quotes around “adjustment = -1” are significant.
- The `cooldown` parameter is similar to the `grace period` above and does not have to be the same on the way up and on the way down.

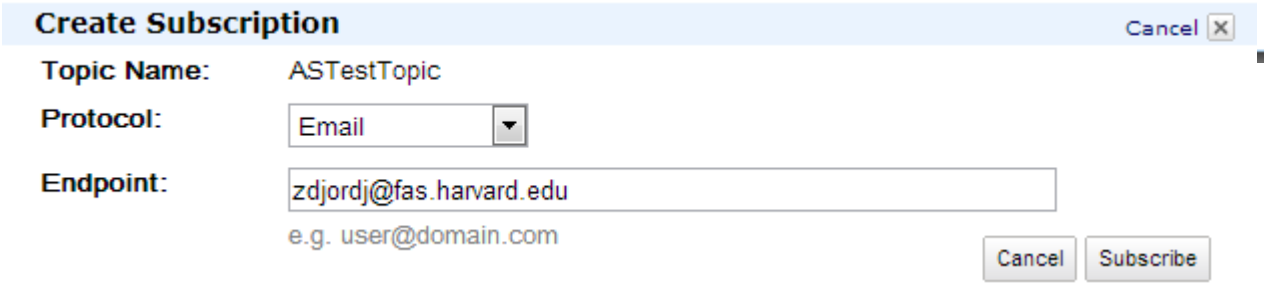
Auto Scaling Policy's `arn`

- The long strings that start with `arn:aws:` and end with `/scale-dn or /scale-up` are Amazon Resource Notations (`arn`-s) for two policies. Those arn-s identify those policies uniquely.
- We still do not know what will trigger changes.
- We need to define triggers and let them activate or execute policies. We will use Cloud Watch Alarms for the triggers.

Create a Topic

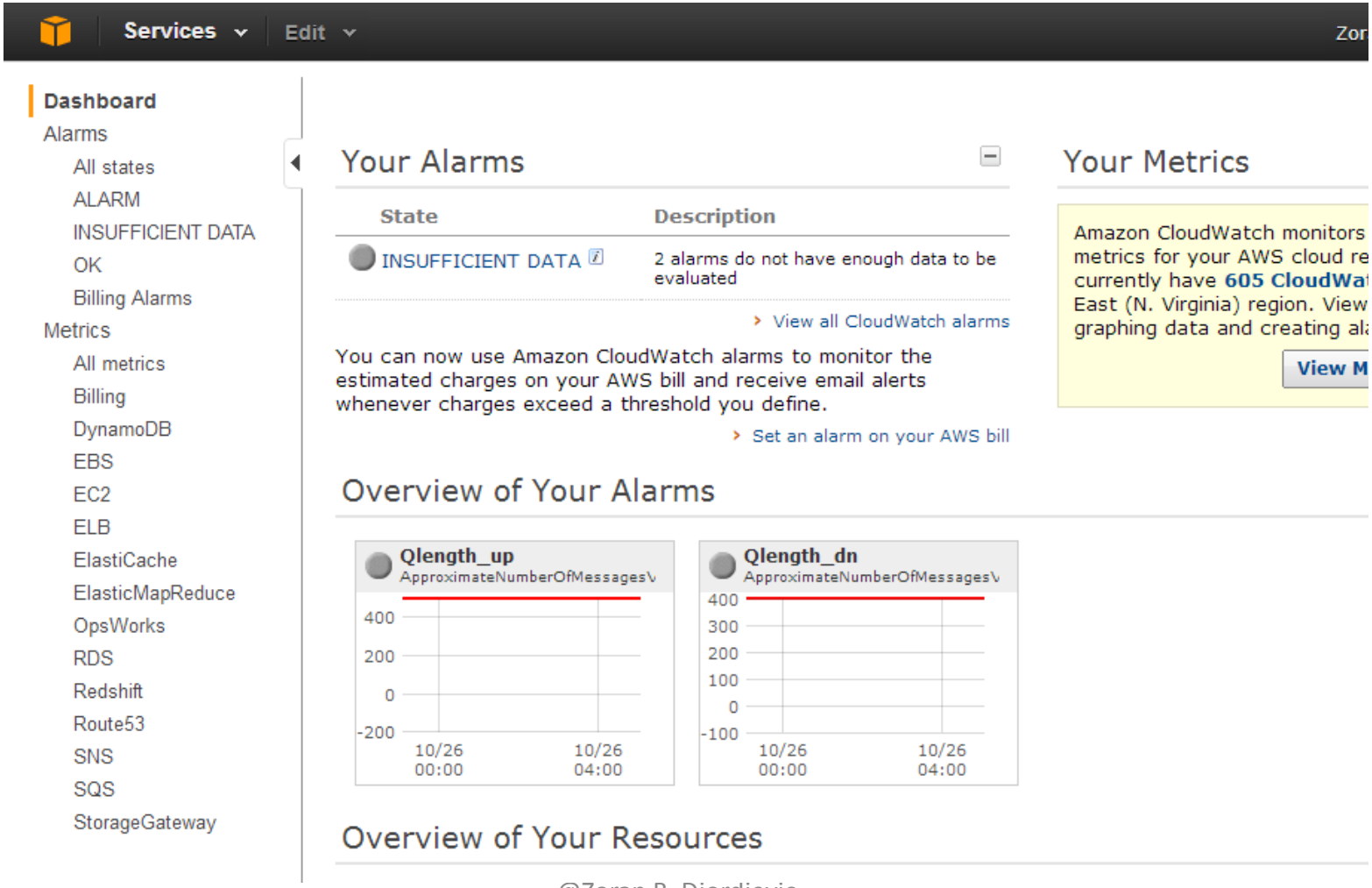
- We will need to send notifications about events taking place in our auto scaling group. Let us create an SNS Topic for that purpose.





Cloud Watch Alarms

- Go the all Services page and select CloudWatch. You will see Monitoring Dashboard. In the left navigation space select SQS.



SQS: Queue Metrics page

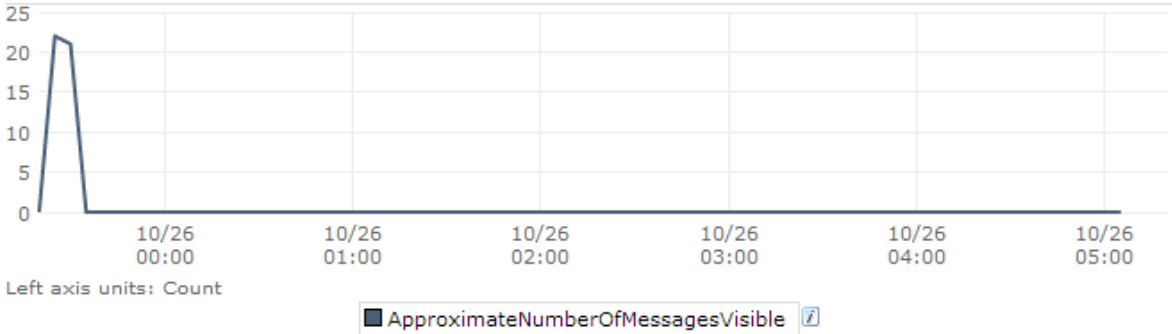
- `ASTestQueue` is the queue we set up for `ClientProducer` and `ServerSideConsumer` message exchange. Queue Metrics page lists metrics Cloud Watch monitors on an SQS queue:

Viewing: All Metrics | AWS/SQS | Search | 1 to 15 of 15 Metrics

SQS: Queue Metrics

	QueueName	MetricName
☐	ASTestQueue	ApproximateNumberOfMessagesDelayed
☐	ASTestQueue	ApproximateNumberOfMessagesNotVisible
☑	ASTestQueue	ApproximateNumberOfMessagesVisible
☐	ASTestQueue	NumberOfEmptyReceives
☐	ASTestQueue	NumberOfMessagesDeleted
☐	ASTestQueue	NumberOfMessagesReceived
☐	ASTestQueue	NumberOfMessagesSent
☐	ASTestQueue	SentMessageSize

ApproximateNumberOfMessagesVisible (Count)... edit | Average | 5 Minutes



Time Range

Zoom: 1h | 3h | 6h | 12h | 24h | 1W | 2W

Relative Absolute

From: 6 hours ago

To: 0 hours ago

Create Alarm | Copy URL | Refresh

Create Alarm: `ApproximateNumberOfMessagesVisible`

- On SQS Metrics page, check `ASTestQueue` `ApproxmateNumberOfMessagesVisible`
- Hit `Create Alarm` button. `Create Alarm Wizard` pops up

Create Alarm Wizard Cancel

SELECT METRIC | DEFINE ALARM | CONFIGURE ACTIONS | REVIEW

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

Identify Your Alarm

Assign your alarm a name and description.

Name: UpwardMovingSQSAlarm

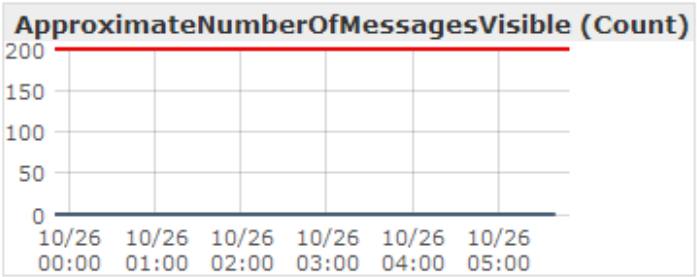
Description: UpwardMovingSQSAlarm

Define Alarm Threshold

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

This alarm will enter the ALARM state when ApproximateNumberOfMessagesVisible is >= 200 for 5 minutes.

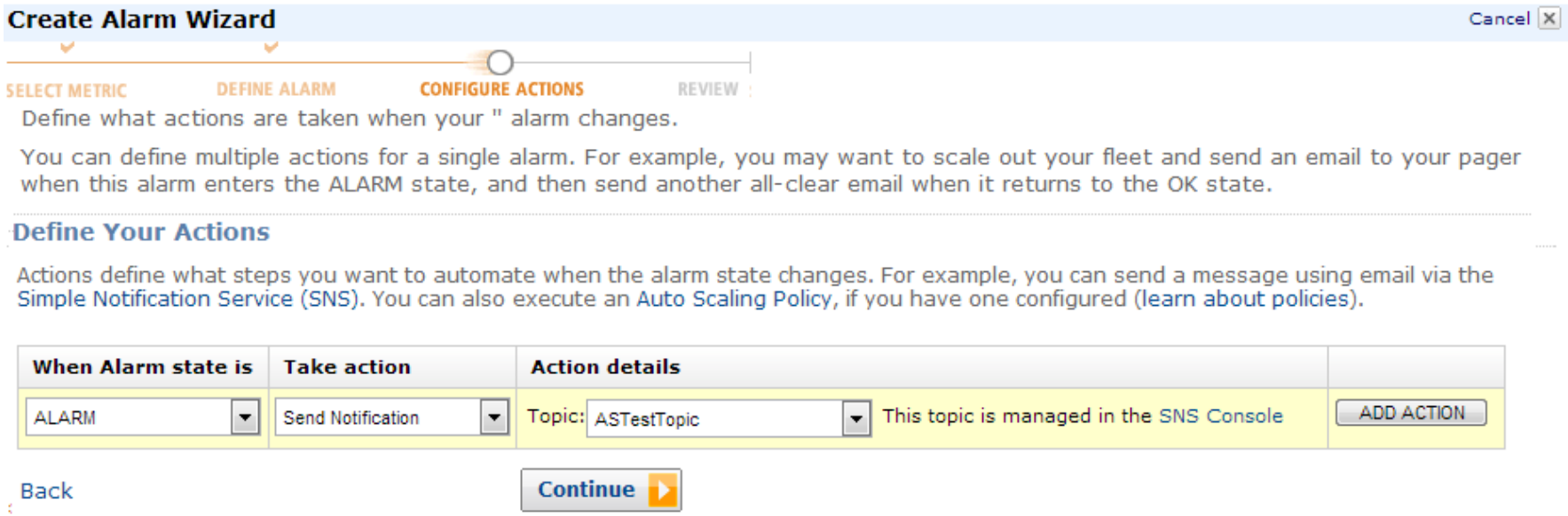
Metric: ApproximateNumberOfMessagesVisible
Period: 5 Minutes
Statistic: Average



‹ Back Continue

Specify Action following the Alarm

- On the next page, under Take action select Send notification and under Action details select the name of one of our `ASTestTopics`.
- Save the alarm and use `ClientProducer` to push the number of messages in the queue past the threshold we set for the alarm (200).

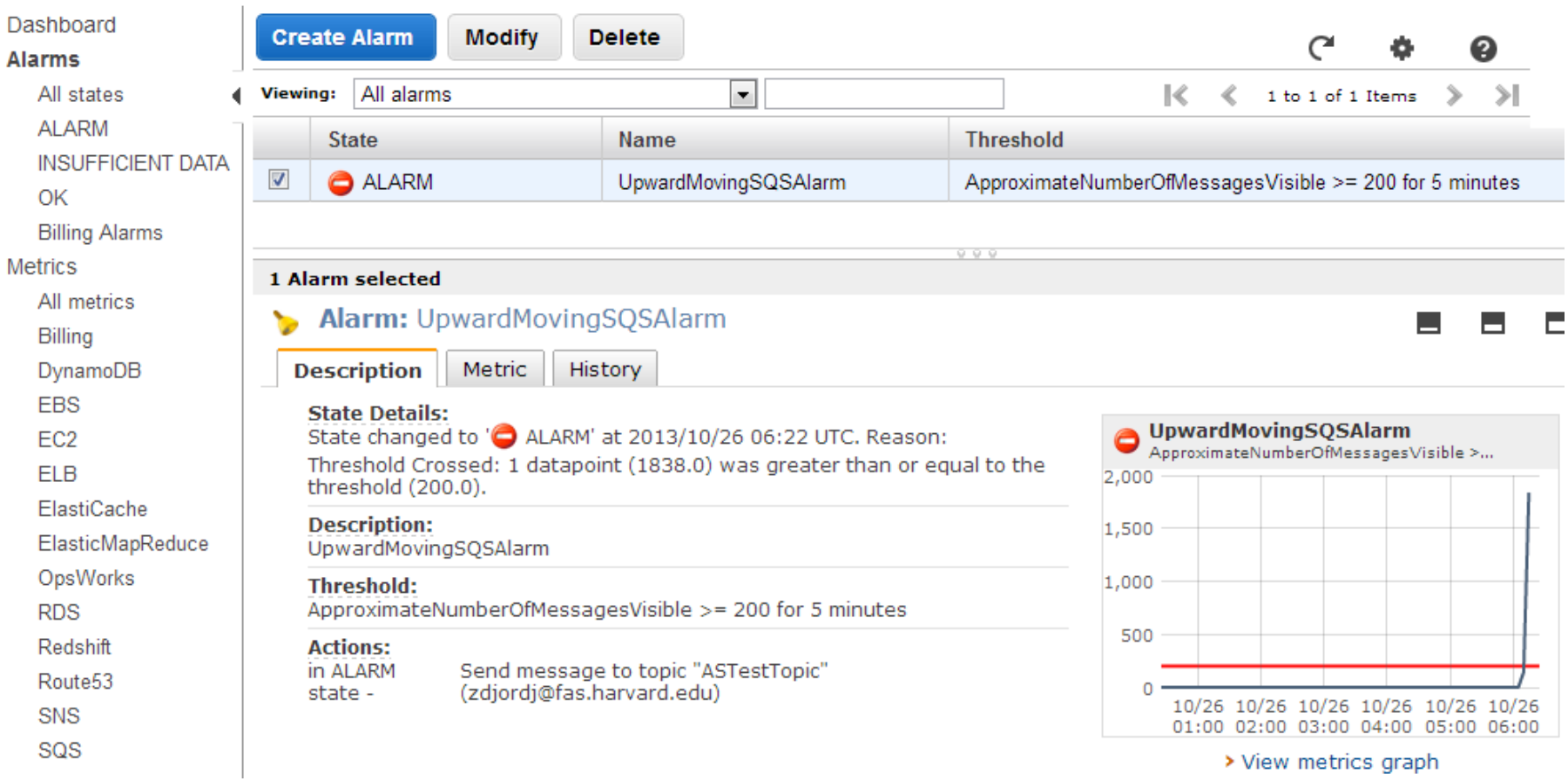


Alarm will send Notification to the Subscriber

- We can reduce the wait time of ClientProducer and it will fairly quickly pump a substantial number of messages into ASTestQueue.
- Cloud Watch examines its metrics with a 5 minute interval. This means that some 5 minutes after starting the ClientProducer, the alarm will take place and we will receive an email notification informing us that the queue has more messages than the threshold (200) we set in the alarm, etc.
- Notification email contains useful information.

State of the Alarm

- Cloud Watch dashboard displays the state of our Alarms.



Notification

ALARM: "UpwardMovingSQSAlarm" in US - N. Virginia

ASTestTopic [no-reply@sns.amazonaws.com] Actions

To: Djordjevic, Zoran Saturday, October 26, 2013 2:23 AM

You are receiving this email because your Amazon CloudWatch Alarm "UpwardMovingSQSAlarm" in the US - N. Virginia region has entered the ALARM state, because "Threshold Crossed: 1 datapoint (1838.0) was greater than or equal to the threshold (200.0)." at "Saturday 26 October, 2013 06:22:57 UTC".

View this alarm in the AWS Management Console:
https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#s=Alarms&alarm=UpwardMovingSQSAlarm

Alarm Details:
- Name: UpwardMovingSQSAlarm
- Description: UpwardMovingSQSAlarm
- State Change: OK -> ALARM
- Reason for State Change: Threshold Crossed: 1 datapoint (1838.0) was greater than or equal to the threshold (200.0).
- Timestamp: Saturday 26 October, 2013 06:22:57 UTC
- AWS Account: 951414139794

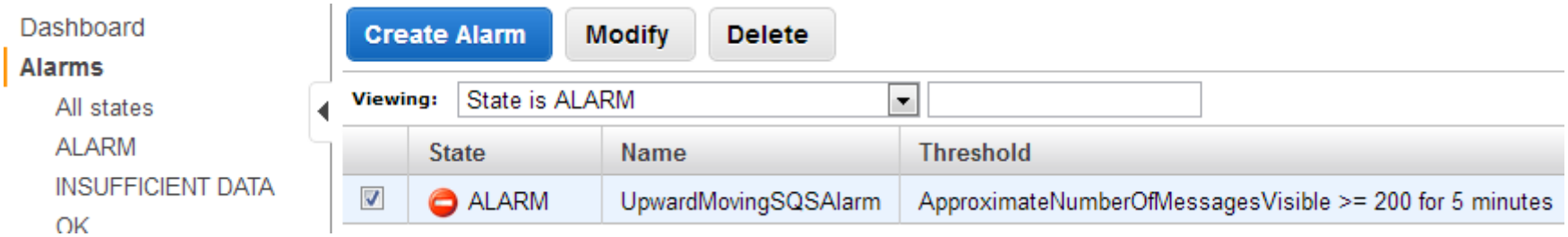
Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanOrEqualToThreshold 200.0 for 300 seconds.

Monitored Metric:
- MetricNamespace: AWS/SQS
- MetricName: ApproximateNumberOfMessagesVisible
- Dimensions: [QueueName = ASTestQueue]
- Period: 300 seconds
- Statistic: Average
- Unit: not specified

State Change Actions:
- OK:
- ALARM: [arn:aws:sns:us-east-1:951414139794:ASTestTopic]

Alarm triggers Auto Scaling Policy

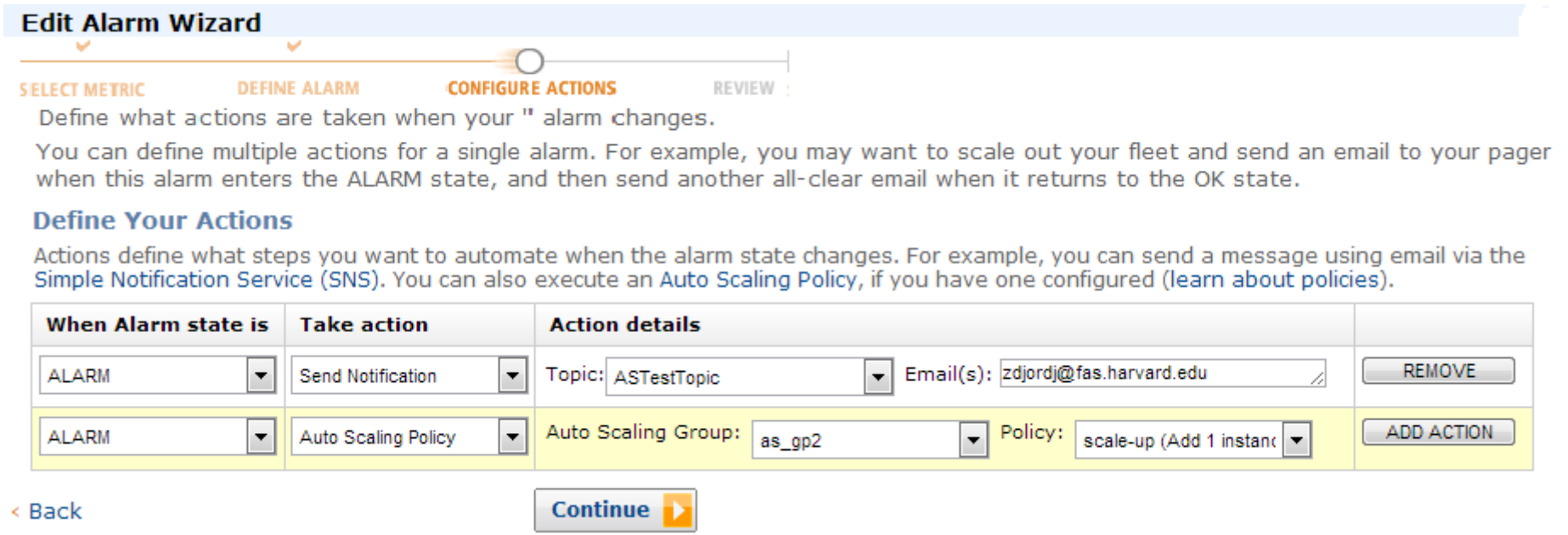
- We will use alarms like the one we just created to execute our auto scaling policies.
- On the top of the left navigation bar on the Cloud Watch dashboard select Alarms -> ALARM. We will see “Our CloudWatch Alarm” page.
- Check the existing alarm (UpwardMovingSQSAlarm) and then select Modify.



- Change the ApproximateNumberOfMessagesVisible to >= 500
- Click Continue
- On the next page we could Edit Alarm and add another action

Edit Alarm Wizard

- Any alarm can result in several actions. New action will be Auto Scaling Policy, under Take action. Under Auto Scaling Group select `as_gp3` group and under policy select `scale_up`, i.e. our "up" policy. Hit `ADD ACTION`



- Hit Continue
- On the following screen, review the alarm setup and `Save Alarm`

Review Alarm Setup and Save

Edit Alarm Wizard Cancel

SELECT METRIC ✓ DEFINE ALARM ✓ CONFIGURE ACTIONS ✓ REVIEW

Please review the alarm information below. If you would like to proceed with this configuration, click **Save Alarm**.
If you want to make any changes to this alarm, click **Back** or select a step on the right to edit.

Alarm Definition Edit Definition

Name:	UpwardMovingSQSAlarm
Description:	UpwardMovingSQSAlarm
In ALARM state when:	the value is >= 500 for 5 minutes

Metric Edit Metric

Namespace:	AWS/SQS
MetricName:	ApproximateNumberOfMessagesVisible
QueueName:	ASTestQueue
Period / Statistic:	5 Minutes / Average

Alarm Actions Edit Actions

Actions:

When alarm state is " ALARM "
Action Type: Send Notification
Action: Notify topic: ASTestTopic (zdjordj@fas.harvard.edu)

When alarm state is " ALARM "
Action Type: Auto Scaling Policy
Action: Use policy scale-up (Add 1 instance) for group as_gp2

‹ Back **Save Alarm**

Configure Downward Alarm

- We again select SQS: Queue Metrics page and hit Create Alarm button.
- This time we will create a downward moving Alarm that will react when the number of messages in `ASTestQueue` falls bellow a threshold. We will call that alarm `DownwardMovingSQSAlarm`.
- The alarm will send a notification to the topic `ASTestTopic` and invoke `scale-dn` auto scaling policy on group `as_gp3`.

Downward Alarm

Create Alarm Wizard Ca

SELECT METRIC ✓ DEFINE ALARM ✓ CONFIGURE ACTIONS ✓ REVIEW

Please review the alarm information below. If you would like to proceed with this configuration, click **Create Alarm**. If you want to make any changes to this alarm, click **Back** or select a step on the right to edit.

Alarm Definition Edit Definition

Name:	DownwardMovingSQSAlarm
Description:	DownwardMovingSQSAlarm
In ALARM state when:	the value is <= 100 for 5 minutes

Metric Edit Metric

Namespace:	AWS/SQS
MetricName:	ApproximateNumberOfMessagesVisible
QueueName:	ASTestQueue
Period / Statistic:	5 Minutes / Average

Alarm Actions Edit Actions

Actions:

When alarm state is " ALARM "
Action Type: Send Notification
Action: Notify topic: ASTestTopic (zdjordj@fas.harvard.edu)

When alarm state is " ALARM "
Action Type: Auto Scaling Policy
Action: Use policy scale-dn (Remove 1 instance) for group as_gp2

< Back **Create Alarm**

EC2 Dashboard, Instances and Load Balancer

- EC2 Dashboard will most probably show several instances.
- Only when the queue message count falls bellow the lower threshold will the number of instances in `as_gp3` group start to decrease.
- Load balancer should also show all of the instances under its monitoring.
- Cloud Watch reports the metrics every 5 minute. That makes these processes somewhat slow.
- One needs to learn the properties of all relevant processes in order to establish an efficient control over the number of EC2 instances.

Once done

- These experiments require a substantial number of instances.
- Empty your auto scaling group

```
c:\> as-update-auto-scaling-group as_gp3 --min-size 0 --max-size 0
```

- Delete your auto scaling group

```
c:\>as-delete-auto-scaling-group as_group2
```

- Drop launch_configuration
- Terminate all involved instances and AMI-s if any persist.
- Terminate your Queues and Topics
- Terminate Load Balancers