Introduction to Cloud Computing (CS 524)

(Lab Assignment 2)

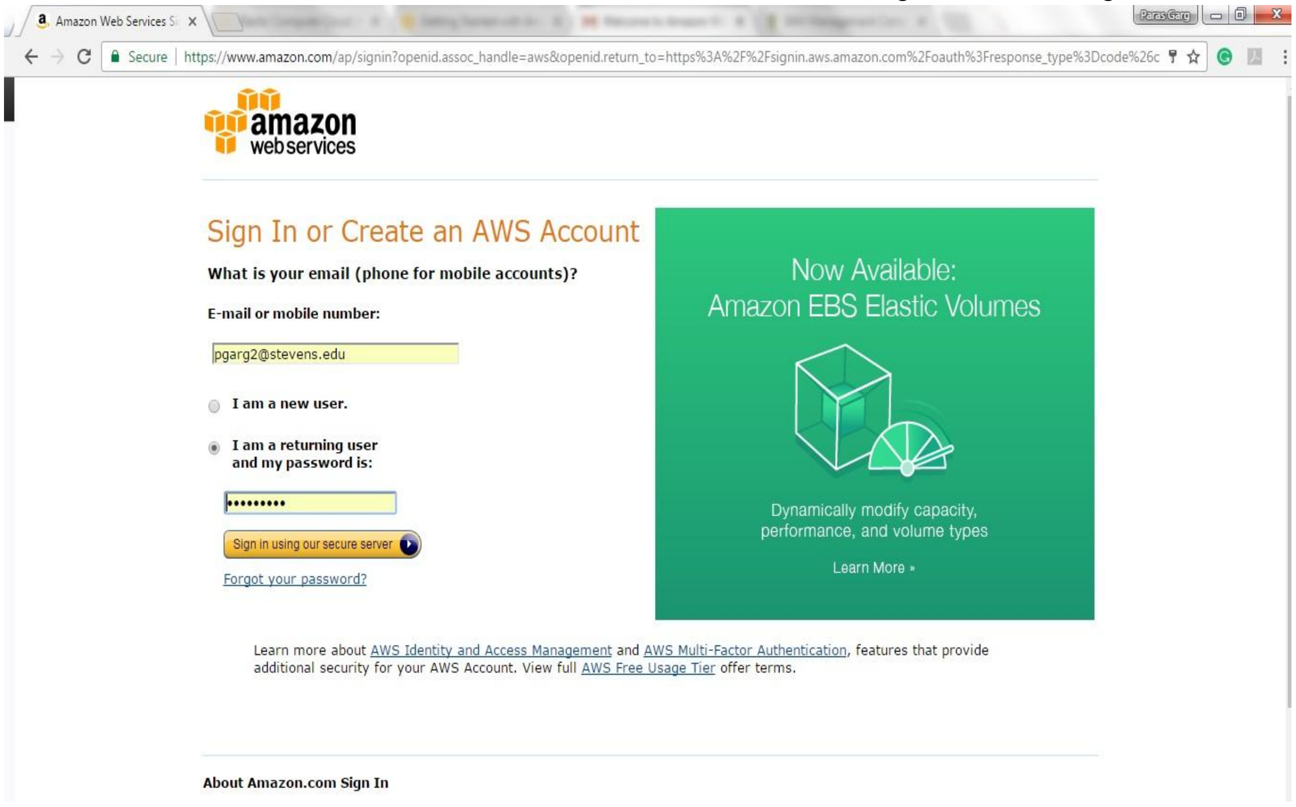
Prof. Igor Faynberg

Student Name: **Paras Garg**

Course Section: **CS 524-A**

Step for accessing Amazon EC2 Instances

- We must have a valid account with AWS. If not then create one and then sign in for accessing the resources.



https://www.amazon.com/ap/signin?openid.assoc_handle=aws&openid.return_to=https%3A%2F%2Fsignin.aws.amazon.com%2Foauth%3Fresponse_type%3Dcode%26c

Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

E-mail or mobile number: pgarg2@stevens.edu

I am a new user.

I am a returning user and my password is: •••••••••

Sign in using our secure server

Forgot your password?

Now Available: Amazon EBS Elastic Volumes

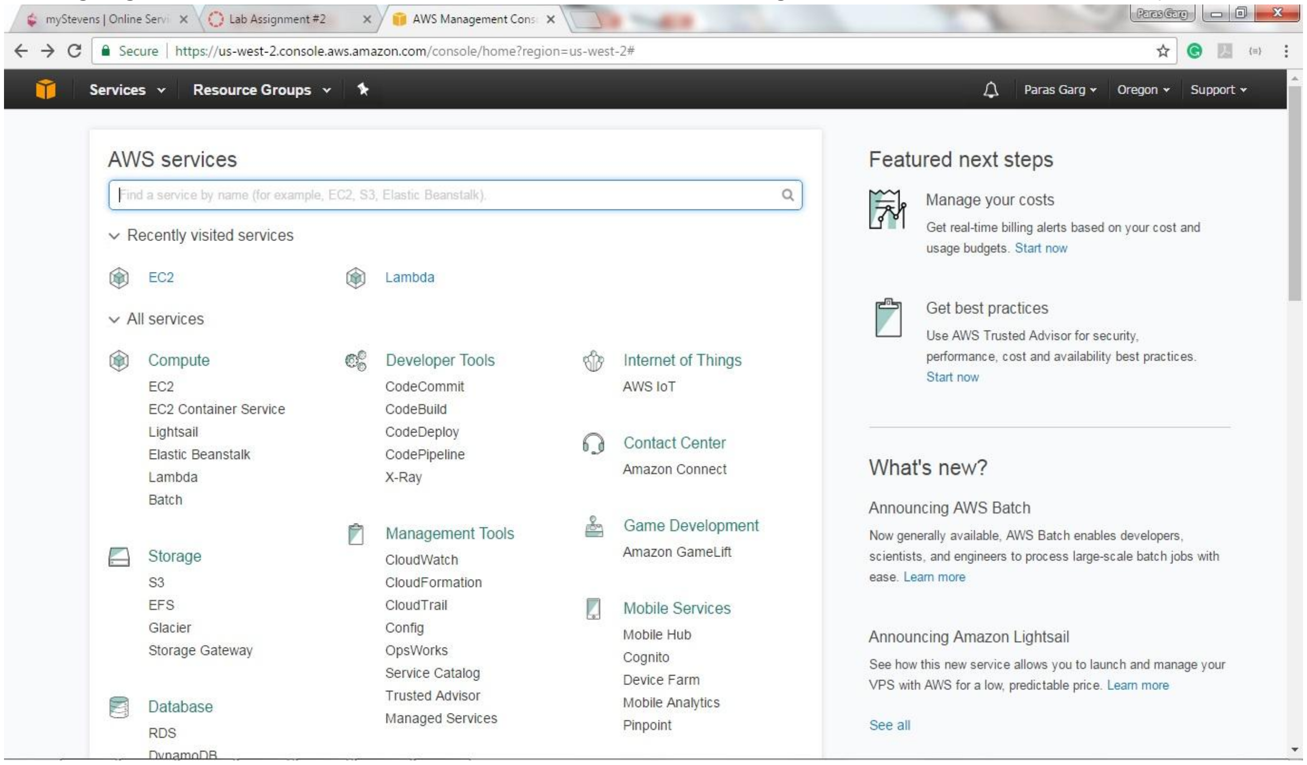
Dynamically modify capacity, performance, and volume types

Learn More »

Learn more about AWS Identity and Access Management and AWS Multi-Factor Authentication, features that provide additional security for your AWS Account. View full AWS Free Usage Tier offer terms.

About Amazon.com Sign In

- After signing in, landed on the dashboard of AWS services. Now clicking on EC2 under All services > Compute > EC2.



https://us-west-2.console.aws.amazon.com/console/home?region=us-west-2#

AWS services

Find a service by name (for example, EC2, S3, Elastic Beanstalk).

Recently visited services

EC2 Lambda

All services

Compute: EC2, EC2 Container Service, Lightsail, Elastic Beanstalk, Lambda, Batch

Storage: S3, EFS, Glacier, Storage Gateway

Database: RDS, DynamoDB

Developer Tools: CodeCommit, CodeBuild, CodeDeploy, CodePipeline, X-Ray

Management Tools: CloudWatch, CloudFormation, CloudTrail, Config, OpsWorks, Service Catalog, Trusted Advisor, Managed Services

Internet of Things: AWS IoT

Contact Center: Amazon Connect

Game Development: Amazon GameLift

Mobile Services: Mobile Hub, Cognito, Device Farm, Mobile Analytics, Pinpoint

Featured next steps

Manage your costs

Get real-time billing alerts based on your cost and usage budgets. Start now

Get best practices

Use AWS Trusted Advisor for security, performance, cost and availability best practices. Start now

What's new?

Announcing AWS Batch

Now generally available, AWS Batch enables developers, scientists, and engineers to process large-scale batch jobs with ease. Learn more

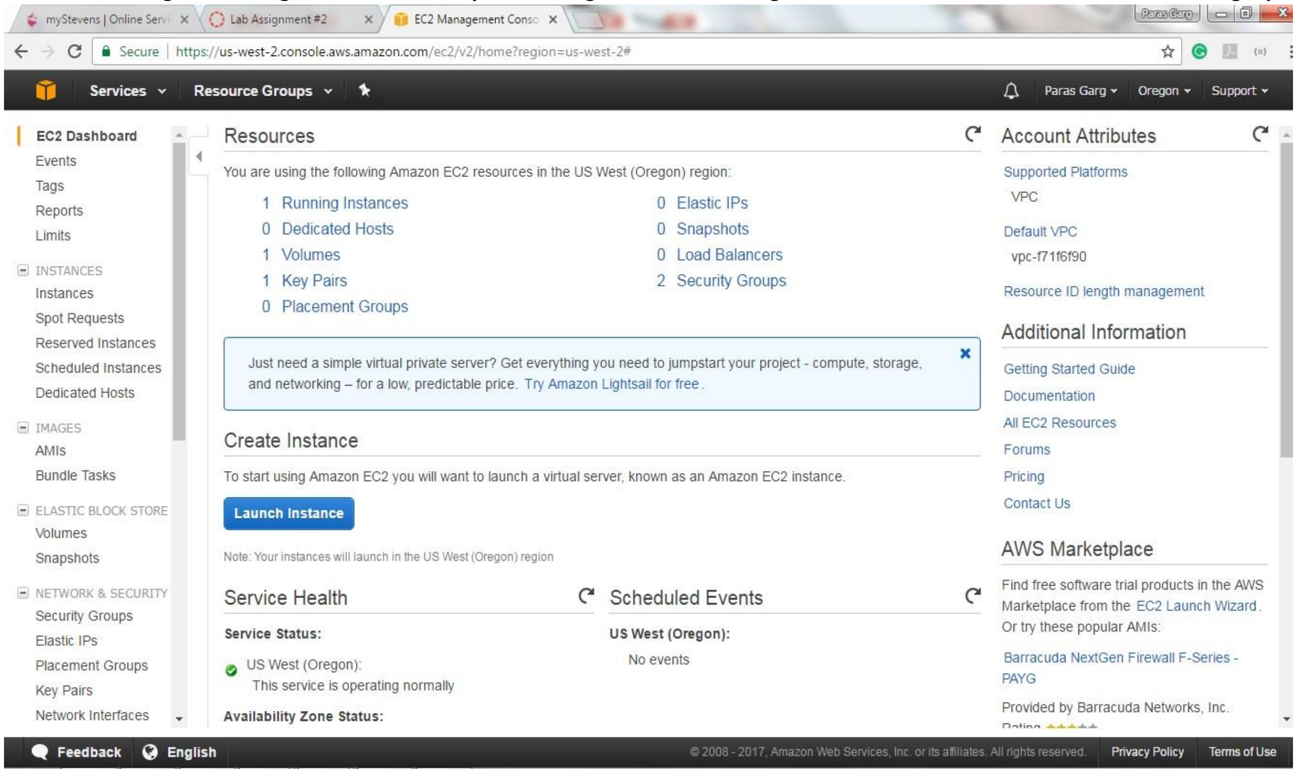
Announcing Amazon Lightsail

See how this new service allows you to launch and manage your VPS with AWS for a low, predictable price. Learn more

See all

Step for Creating an Amazon EC2 Instances (Using AWS Panel)

- Now in EC2 Dashboard, Launching Instance by clicking on "**Launch Instance**" button to create new instance and able to check existing running instances by clicking on Running Instances under Resources category.



Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

1 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
1 Volumes	0 Load Balancers
1 Key Pairs	2 Security Groups
0 Placement Groups	

Just need a simple virtual private server? Get everything you need to jumpstart your project - compute, storage, and networking – for a low, predictable price. Try Amazon Lightsail for free.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon):
This service is operating normally

Availability Zone Status:

Scheduled Events

US West (Oregon):

No events

Account Attributes

Supported Platforms
VPC

Default VPC
vpc-f71f6f90

Resource ID length management

Additional Information

Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace

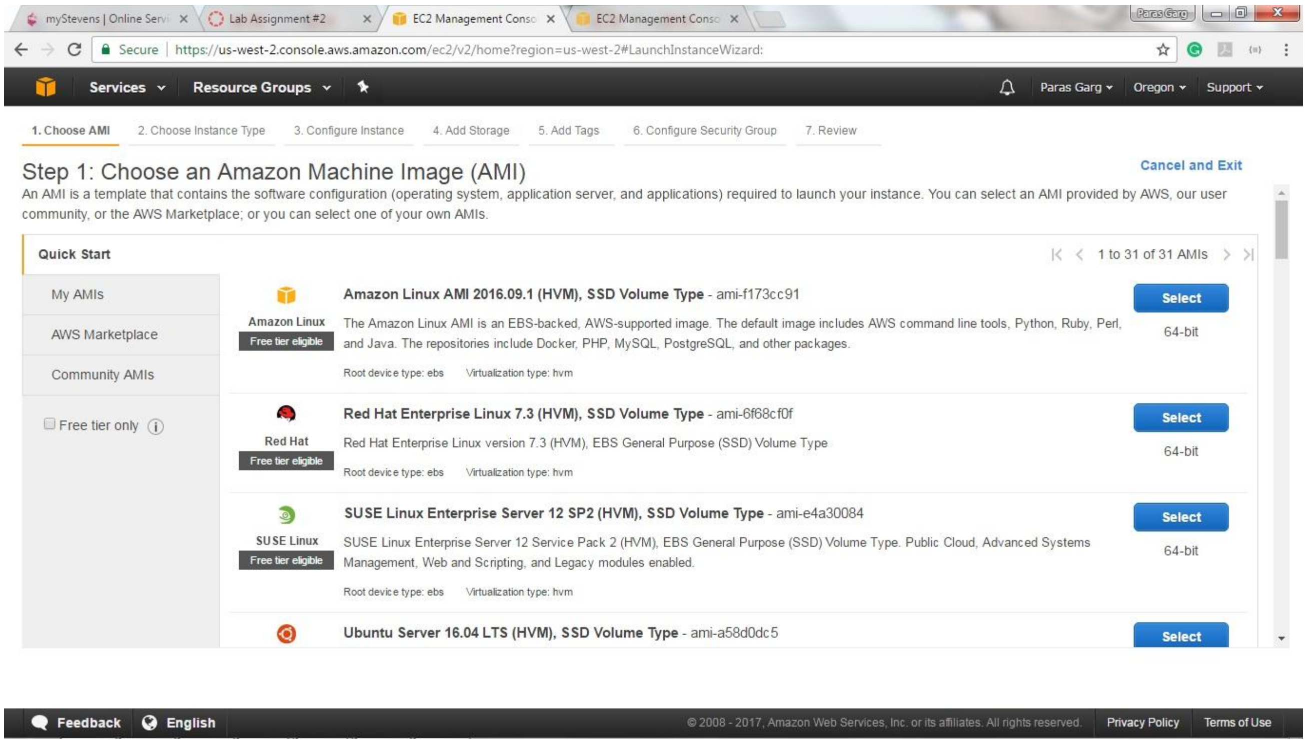
Find free software trial products in the AWS Marketplace from the EC2 Launch Wizard. Or try these popular AMIs:

Barracuda NextGen Firewall F-Series - PAYG

Provided by Barracuda Networks, Inc.

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- Now, this is a first step to create an AWS Instance, here we have to choose an Amazon Machine Image (AMI). Selecting select **Amazon Linux AMI (64-bit).**



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.

Amazon Linux — Free tier eligible
Amazon Linux AMI 2016.09.1 (HVM), SSD Volume Type - ami-f173cc91
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm — 64-bit

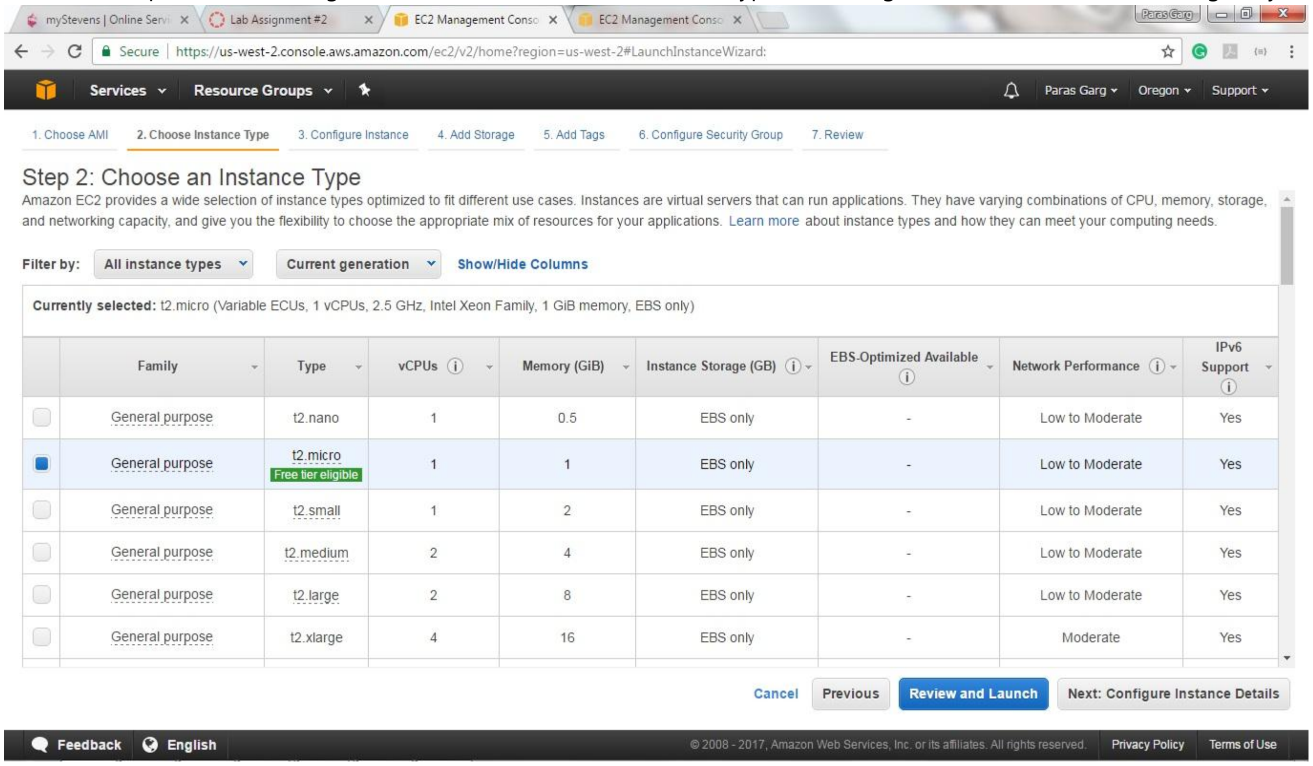
Red Hat — Free tier eligible
Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type - ami-6f68cf0f
Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm — 64-bit

SUSE Linux — Free tier eligible
SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-e4a30084
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm — 64-bit

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a58d0dc5

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- The next step after selecting AMI is to choose an Instance Type. Selecting **t2.micro** for free tier eligibility.



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

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.

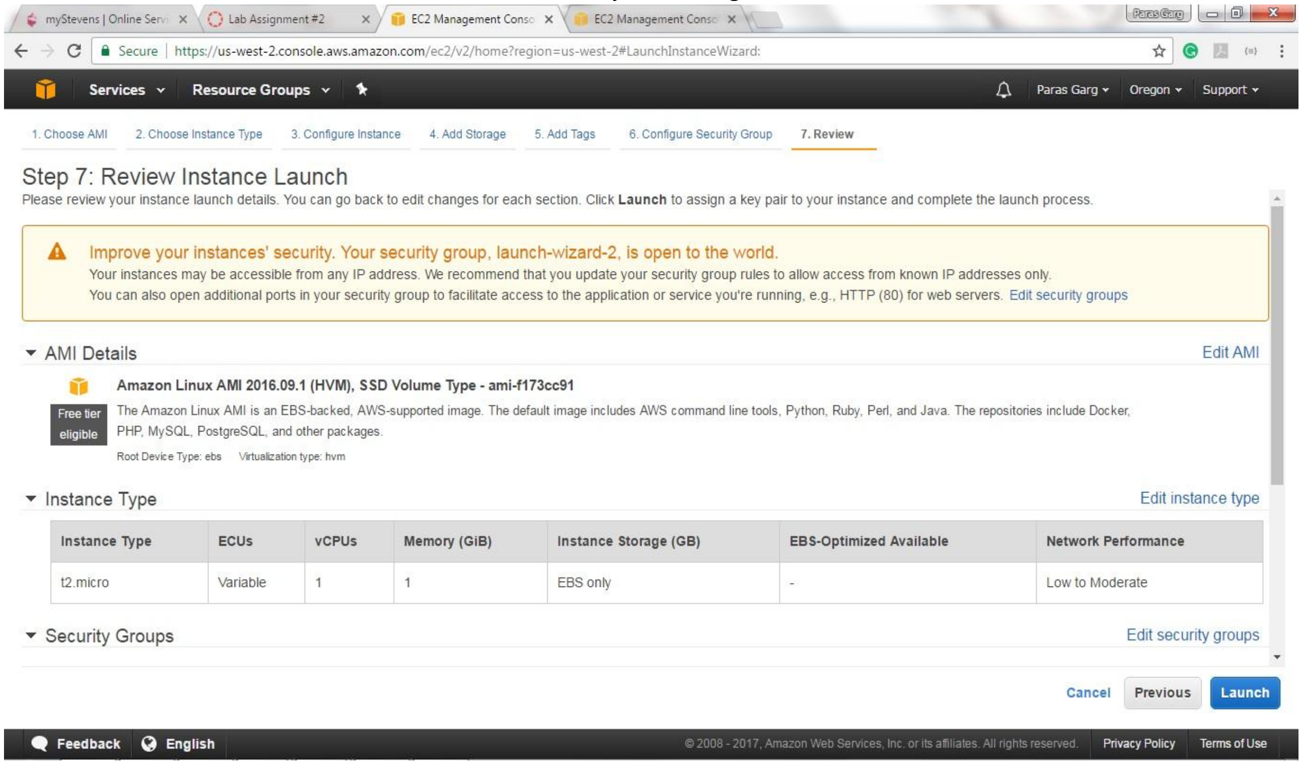
Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
■	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes

Cancel | Previous | Review and Launch | Next: Configure Instance Details

- As I have selected free tier instance type, I have jumped to the final step in instance creation. In this step I have to review the instance information and launch it by clicking on Launch button at the bottom.



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

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, launch-wizard-2, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

AMI Details — Edit AMI

Amazon Linux AMI 2016.09.1 (HVM), SSD Volume Type - ami-f173cc91
Free tier eligible
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root Device Type: ebs   Virtualization type: hvm

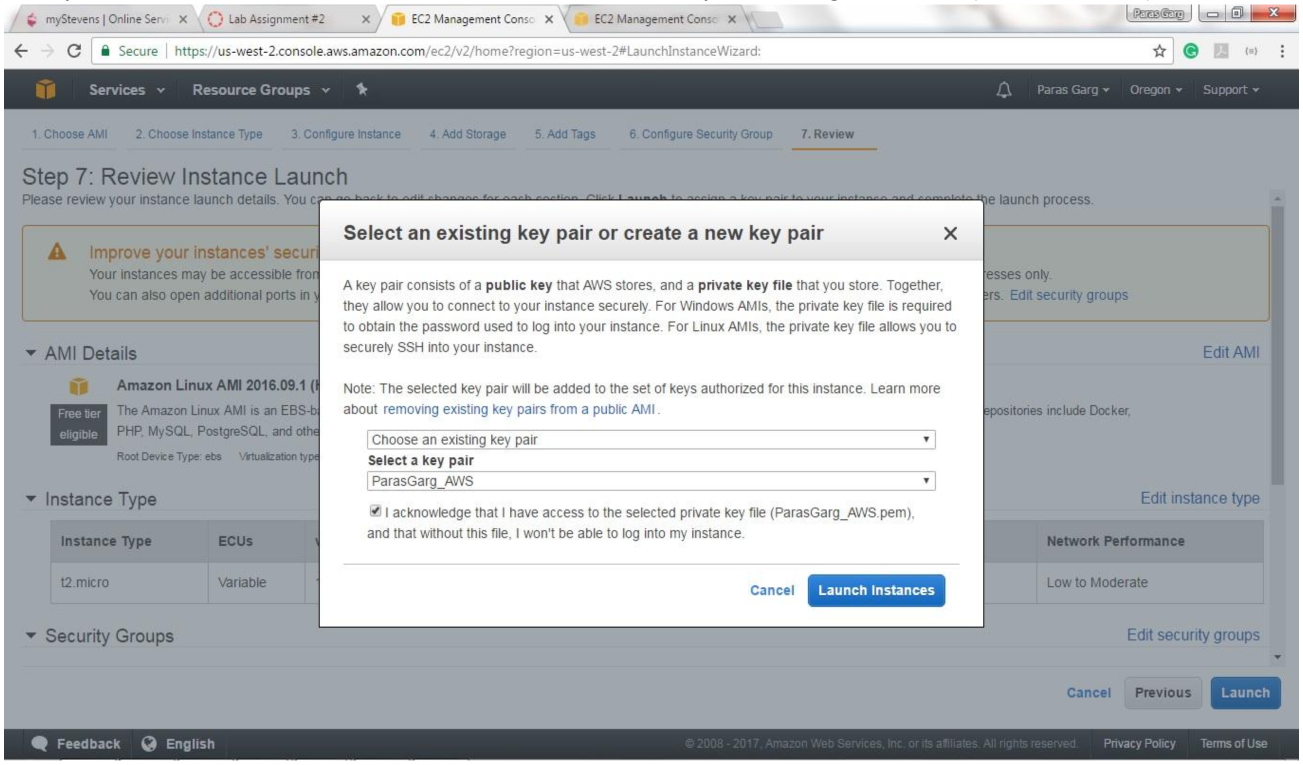
Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups — Edit security groups

Cancel | Previous | Launch

- After review, for instance security, I have to assign a key pair to the created instance. I can use existing key pair as I had already created one else I have to create one by choosing other option in dropdown list.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

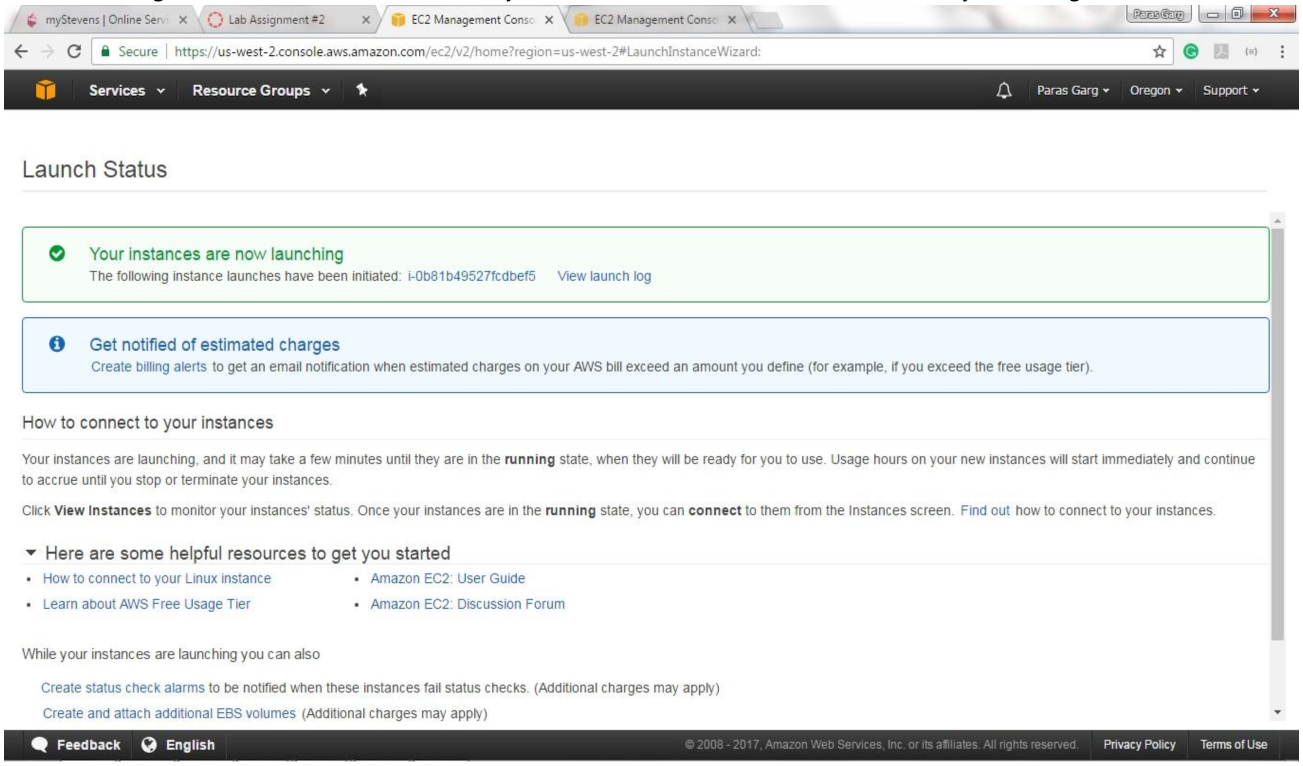
Choose an existing key pair

Select a key pair

ParasGarg_AWS

☑ I acknowledge that I have access to the selected private key file (ParasGarg_AWS.pem), and that without this file, I won't be able to log into my instance.

- After creating an instance, it is ready to launch and can view it by clicking View Instance.



Launch Status

Your instances are now launching

The following instance launches have been initiated: i-0b81b49527fcdbef5 View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

Here are some helpful resources to get you started

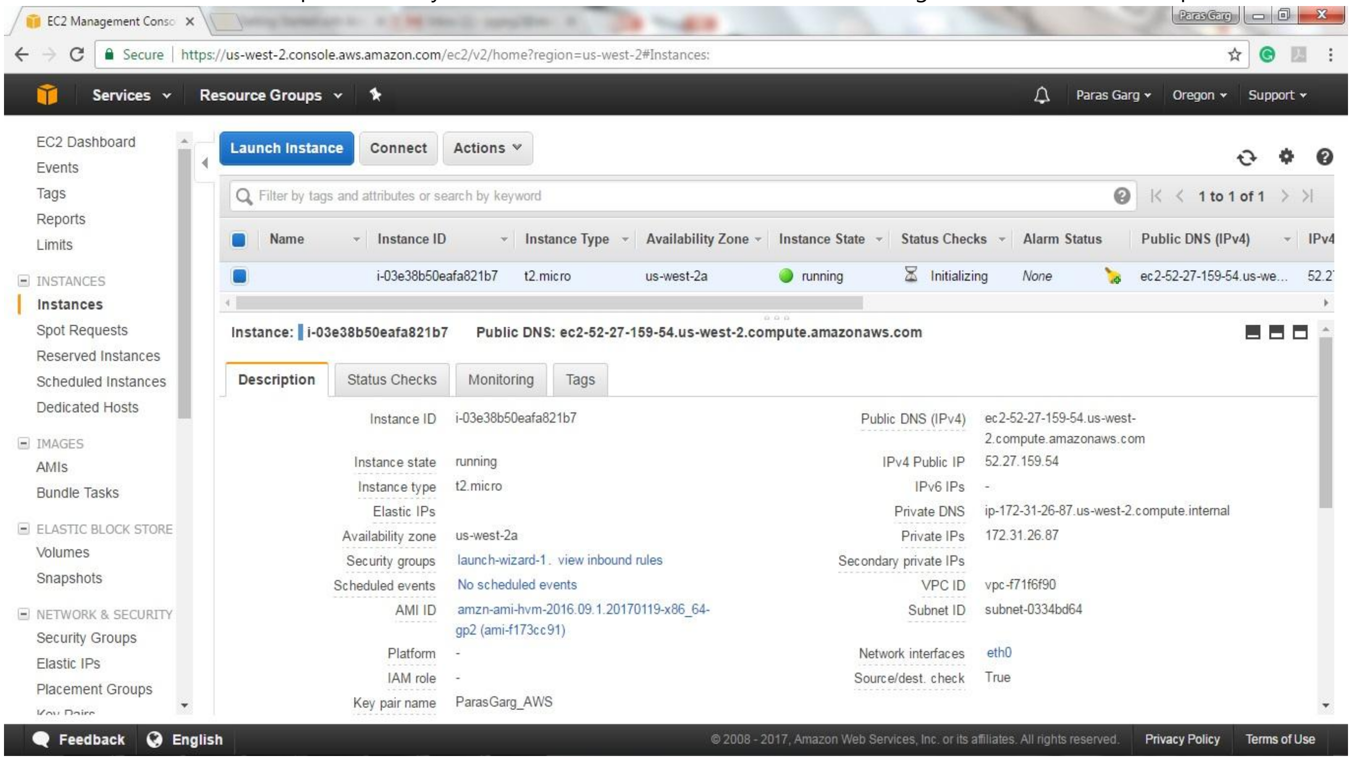
- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)

Create and attach additional EBS volumes (Additional charges may apply)

- Now, we can view description about your instance and other modules. Now viewing Instances and its description.

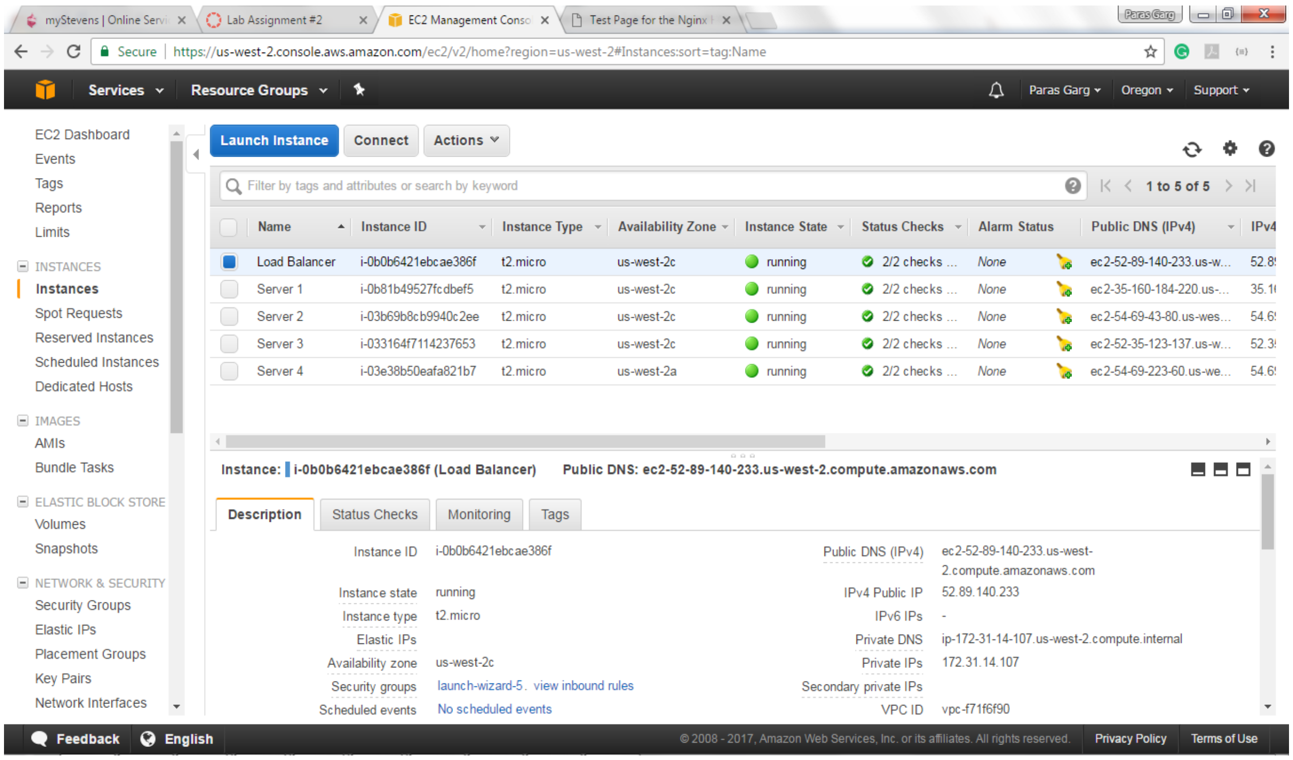


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4
	i-03e38b50eafa821b7	t2.micro	us-west-2a	running	Initializing	None	ec2-52-27-159-54.us-we...	52.2

Instance: i-03e38b50eafa821b7 Public DNS: ec2-52-27-159-54.us-west-2.compute.amazonaws.com

Instance ID	i-03e38b50eafa821b7	Public DNS (IPv4)	ec2-52-27-159-54.us-west-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	52.27.159.54
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-26-87.us-west-2.compute.internal
Availability zone	us-west-2a	Private IPs	172.31.26.87
Security groups	launch-wizard-1. view inbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-f71f6f90
AMI ID	amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2 (ami-f173cc91)	Subnet ID	subnet-0334bd64
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	ParasGarg_AWS		

- Now, repeat the same steps to create new EC2 instance every time. I have created five instances in total and named them as Load Balancer, Server 1, Server 2, Server 3, and Server 4.



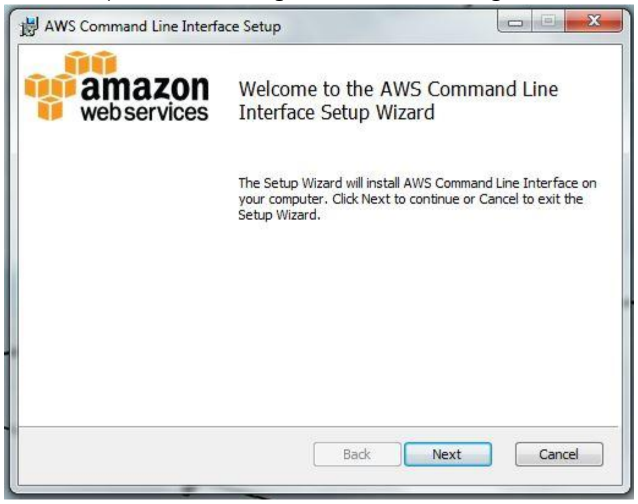
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4
Load Balancer	i-0b0b6421ebcae386f	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-52-89-140-233.us-w...	52.8
Server 1	i-0b81b49527fcdbef5	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-35-160-184-220.us-...	35.1
Server 2	i-03b69b8cb9940c2ee	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-54-69-43-80.us-wes...	54.6
Server 3	i-033164f7114237653	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-52-35-123-137.us-w...	52.3
Server 4	i-03e38b50eafa821b7	t2.micro	us-west-2a	running	2/2 checks ...	None	ec2-54-69-223-60.us-we...	54.6

Instance: i-0b0b6421ebcae386f (Load Balancer) Public DNS: ec2-52-89-140-233.us-west-2.compute.amazonaws.com

Instance ID	i-0b0b6421ebcae386f	Public DNS (IPv4)	ec2-52-89-140-233.us-west-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	52.89.140.233
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-14-107.us-west-2.compute.internal
Availability zone	us-west-2c	Private IPs	172.31.14.107
Security groups	launch-wizard-5. view inbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-f71f6f90

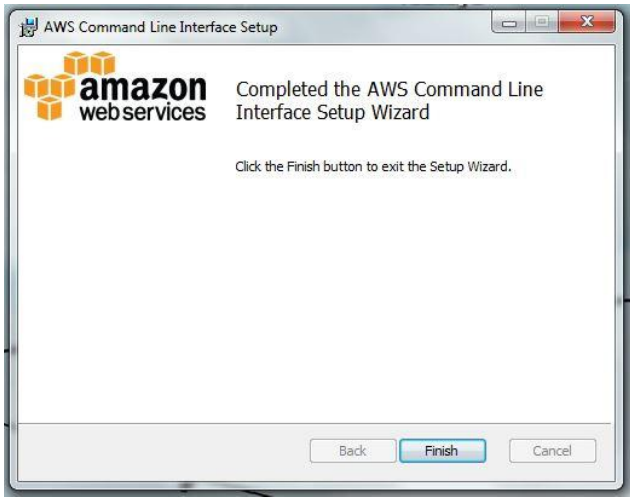
Step for Creating an Amazon EC2 Instances (Using AWS Command Line Interface)

- First step in creating instances using command line interface is to download and install it.


AWS Command Line Interface Setup

Welcome to the AWS Command Line Interface Setup Wizard

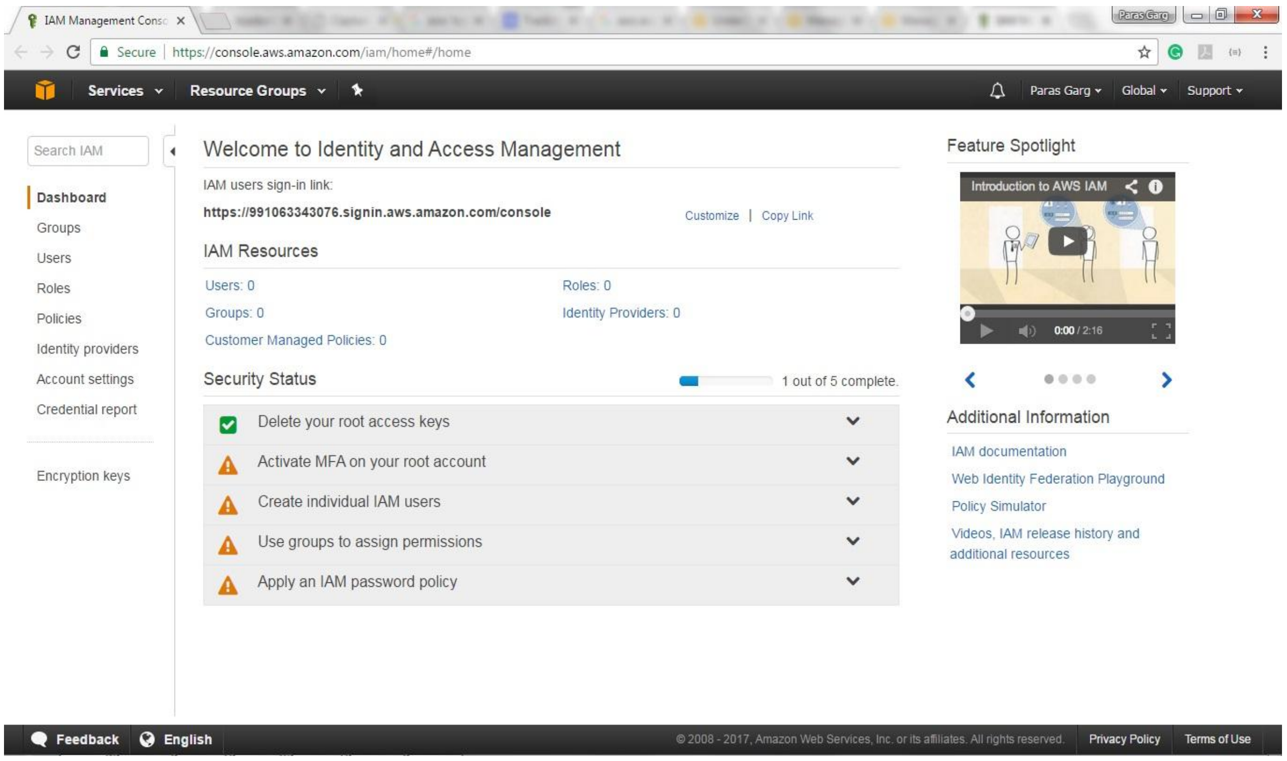
The Setup Wizard will install AWS Command Line Interface on your computer. Click Next to continue or Cancel to exit the Setup Wizard.


AWS Command Line Interface Setup

Completed the AWS Command Line Interface Setup Wizard

Click the Finish button to exit the Setup Wizard.

- Now to access the resources, we need **access key id** and **secret access key.** If we are using it first time, then we have to create one by login in to Amazon EC2 and then search for IAM in AWS service dashboard. The IAM dashboard would look like this, having everything 0 under IAM Resources tab.



Welcome to Identity and Access Management

IAM users sign-in link:

https://991063343076.signin.aws.amazon.com/console   Customize | Copy Link

IAM Resources

Users: 0   Roles: 0

Groups: 0   Identity Providers: 0

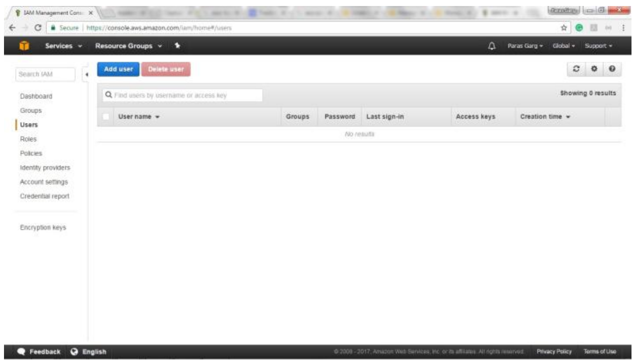
Customer Managed Policies: 0

Security Status   1 out of 5 complete.

- ✓ Delete your root access keys
- ⚠ Activate MFA on your root account
- ⚠ Create individual IAM users
- ⚠ Use groups to assign permissions
- ⚠ Apply an IAM password policy

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- Now in Users navigation bar and clicking Add user. Then entered a user name and, selected AWS Management Console access and we can either set custom password or auto generate it.

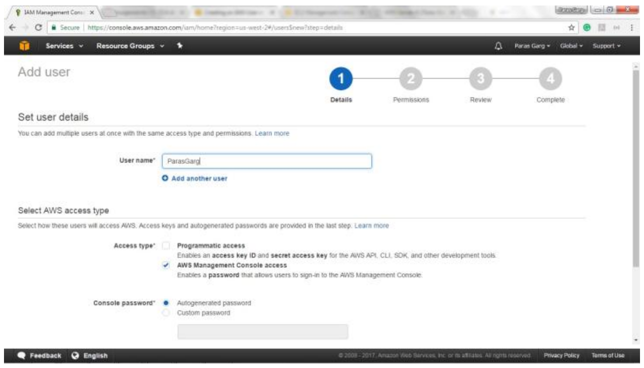


https://console.aws.amazon.com/iam/home#/users

Showing 0 results

User name	Groups	Password	Last sign-in	Access keys	Creation time

No results



https://console.aws.amazon.com/iam/home?region=us-west-2#/users$new?step=details

Add user

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* ParasGarg

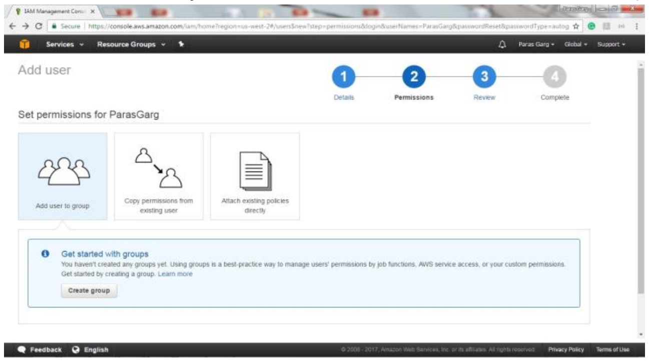
Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* ☐ Programmatic access
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
✓ AWS Management Console access
Enables a **password** that allows users to sign-in to the AWS Management Console.

Console password* ● Autogenerated password
○ Custom password

- Now inside permission step, if we wish to create a group then can create here else we will create it later. Next step is Review, to review your details and then, next is to finish the process by clicking complete button.

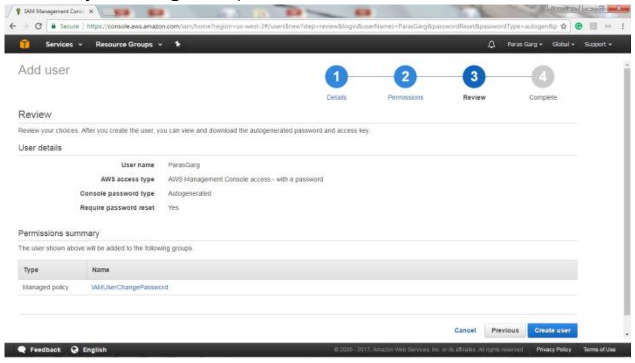


https://console.aws.amazon.com/iam/home?region=us-west-2#/users$new?step=permissions&login&userNames=ParasGarg&passwordReset&passwordType=autog

Add user

Set permissions for ParasGarg

Get started with groups
You haven't created any groups yet. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Get started by creating a group. Learn more



https://console.aws.amazon.com/iam/home?region=us-west-2#/users$new?step=review&login&userNames=ParasGarg&passwordReset&passwordType=autogen&p

Add user

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

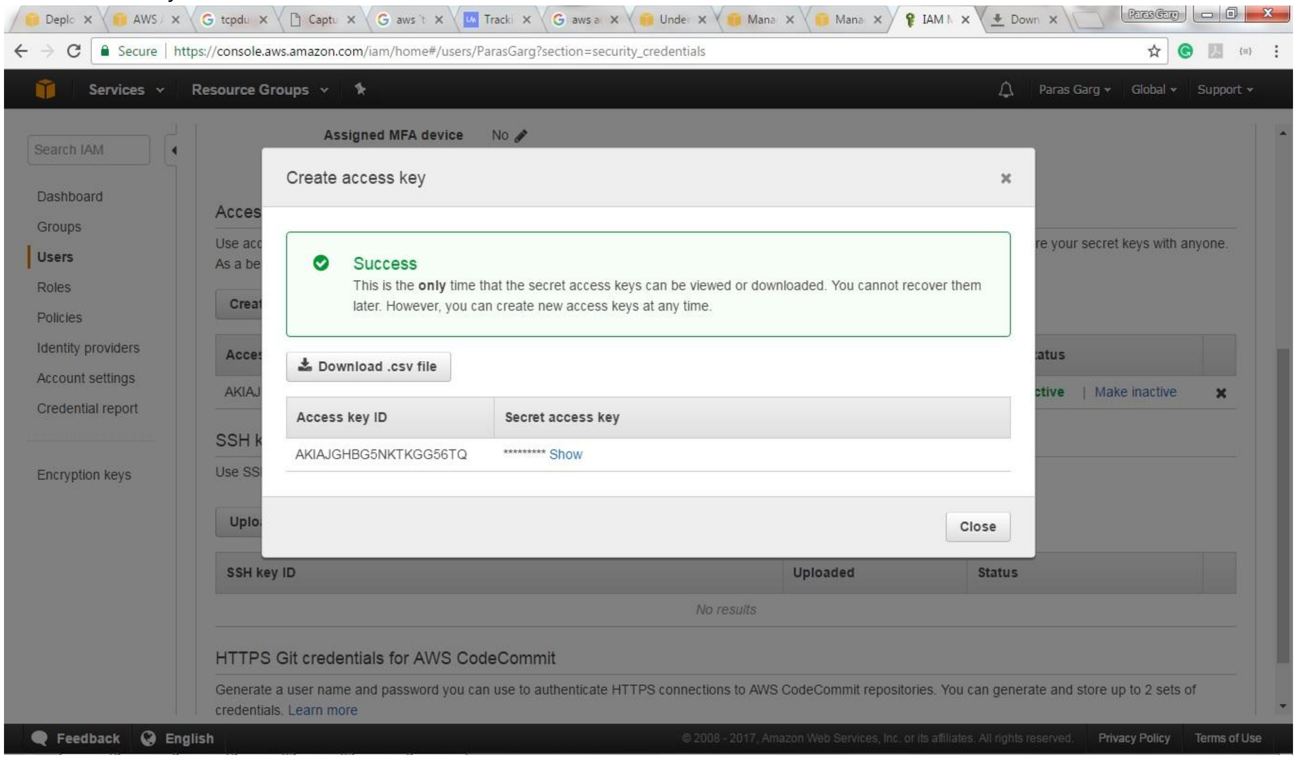
User name	ParasGarg
AWS access type	AWS Management Console access - with a password
Console password type	Autogenerated
Require password reset	Yes

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Managed policy	IAMUserChangePassword

- Now going back to Users tab in IAM dashboard. Selected the user we just created. We will see the details of the user. Clicked on **Security credentials,** under it clicked **Create access key.** From here we will create our Access key id and secret access key.



https://console.aws.amazon.com/iam/home#/users/ParasGarg?section=security_credentials

Assigned MFA device No

Create access key

Success
This is the **only** time that the secret access keys can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time.

Access key ID	Secret access key
AKIAJGHBG5NKTKGG56TQ	********* Show

SSH key ID	Uploaded	Status

No results

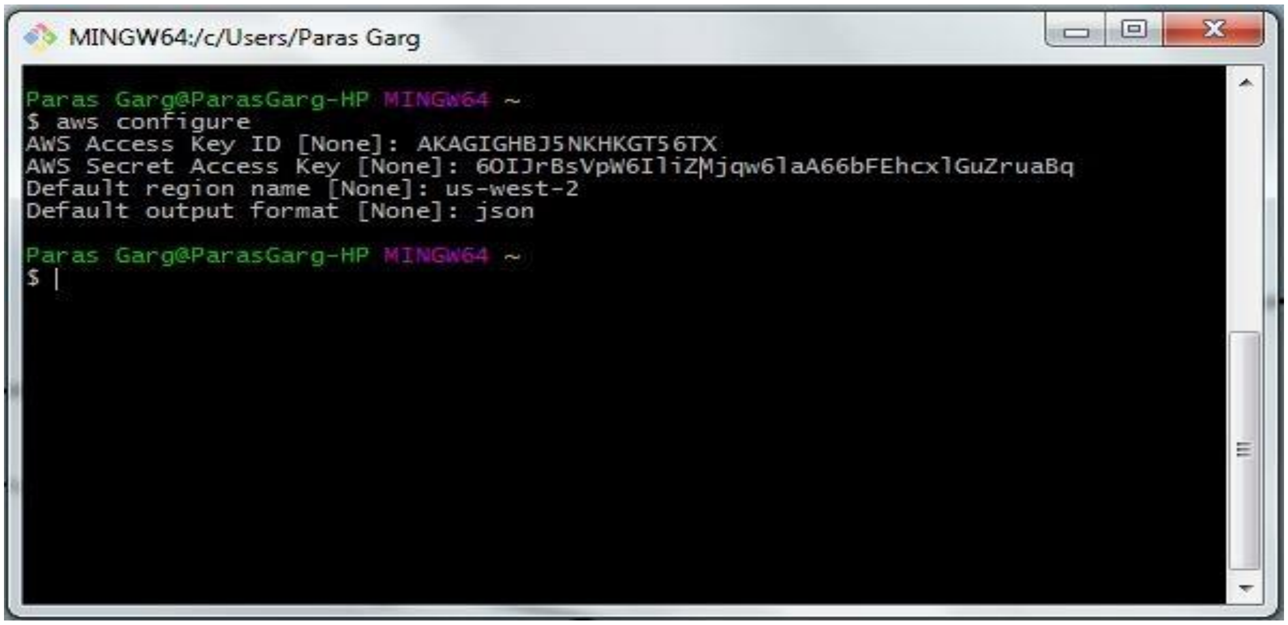
HTTPS Git credentials for AWS CodeCommit

Generate a user name and password you can use to authenticate HTTPS connections to AWS CodeCommit repositories. You can generate and store up to 2 sets of credentials. Learn more

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- Now, configuring AWS Access key id and Secret access key by executing following command and filling the prompted details.

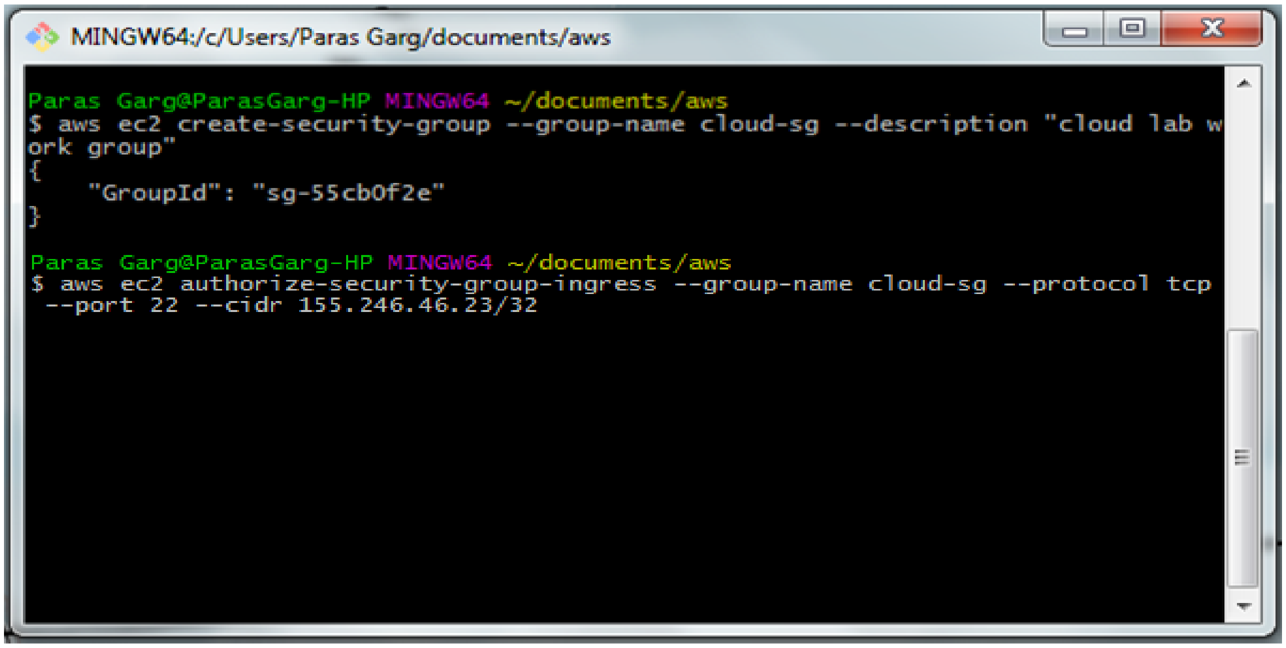
```
$ aws configure
```



```
Paras Garg@ParasGarg-HP MINGW64 ~
$ aws configure
AWS Access Key ID [None]: AKAGIGHBJ5NKHKGT56TX
AWS Secret Access Key [None]: 6OIJrBsVpW6IliZMjqw6laA66bFEhcxlGuZruaBq
Default region name [None]: us-west-2
Default output format [None]: json

Paras Garg@ParasGarg-HP MINGW64 ~
$ |
```

- Now, creating a security group by executing the following command. Make sure that the policy *AdministratorAccess* is attached with the user. (IAM Dashboard > Policies > Check *AdministratorAccess* > Click Policies actions > Attach and select Username).

```
$ aws ec2 create-security-group --group-name <ENTER NAME> --description "<ENTER DESCRIPTION>"

$ aws ec2 authorize-security-group-ingress --group-name <ENTER_GROUP_NAME> --protocol tcp --port 22 --cidr <YOUR_PUBLIC_IP>/32
```



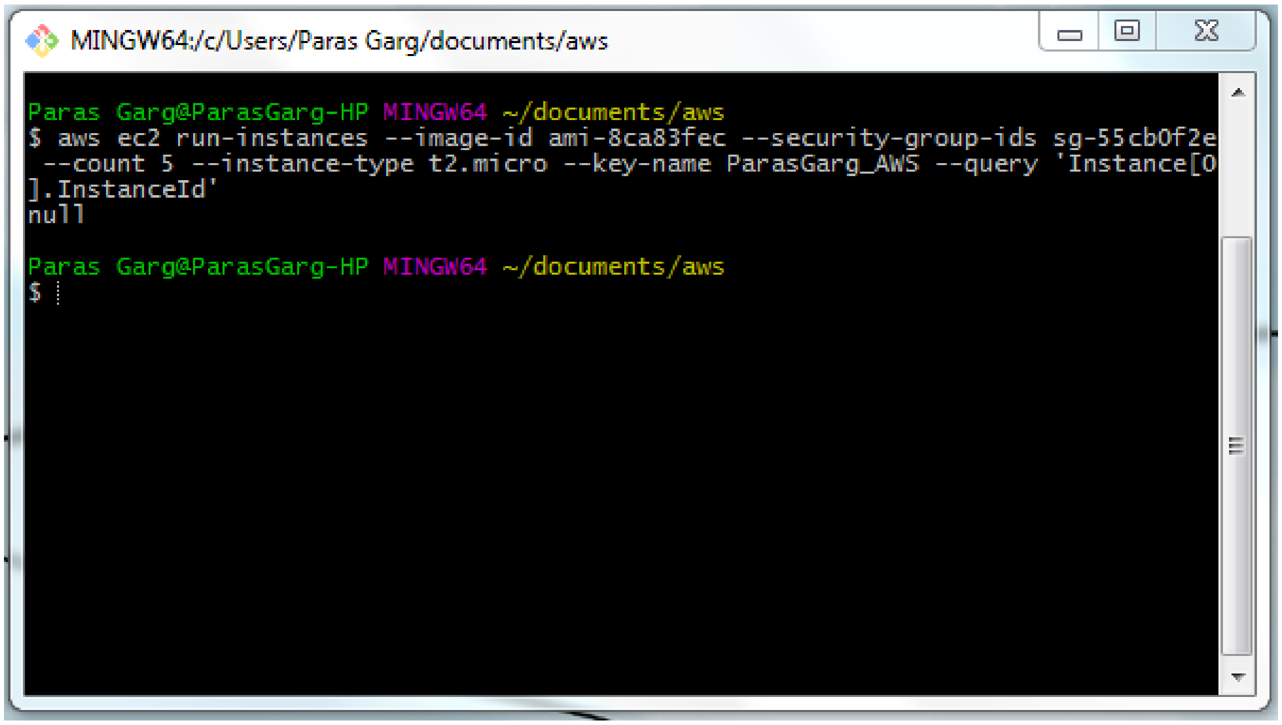
```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 create-security-group --group-name cloud-sg --description "cloud lab w
ork group"
{
    "GroupId": "sg-55cb0f2e"
}

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 authorize-security-group-ingress --group-name cloud-sg --protocol tcp
 --port 22 --cidr 155.246.46.23/32
```

- In this step, you can create a key pair to access aws resources by executing following command. If you already have one then no need to create new key pair. I already have a key-pair, so, I am using that.

```
$ aws ec2 create-key-pair --key-name ParasGarg_AWS --query “KeyMaterial” --ouput text
```

- Finally, to create instance using command line interface, we need **Amazon Linux AMI, Security Group Id, Instance Type** and **Key Pair Name**. And then execute the following command to create any number of instances.

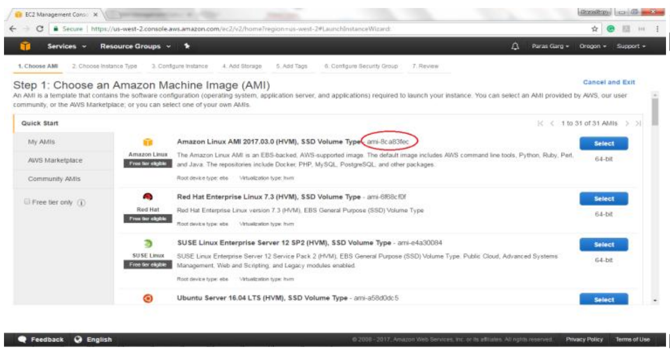
```
$ aws ec2 run-instances --image-id <ami_id> --security-group-ids <group-id-created-above> --count <number-of-instances> --instance-type <instance-type> --key-name <your-key-pair> --query ‘Instance[0].InstanceId’
```



MINGW64:/c/Users/Paras Garg/documents/aws

```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 run-instances --image-id ami-8ca83fec --security-group-ids sg-55cb0f2e
 --count 5 --instance-type t2.micro --key-name ParasGarg_AWS --query 'Instance[0
].InstanceId'
null

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$
```

  - Find AMI and Instance Type



Step 1: Choose an Amazon Machine Image (AMI)

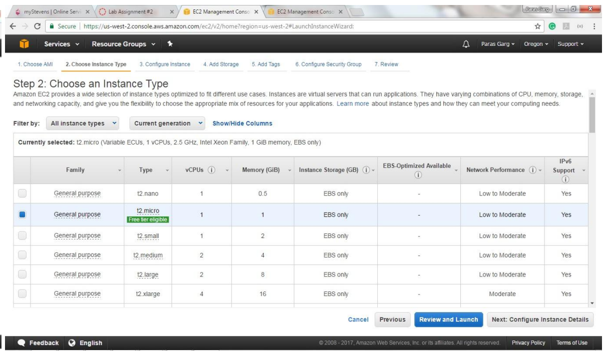
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux AMI 2017.03.0 (HVM), SSD Volume Type - ami-8ca83fec
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs   Virtualization type: hvm

Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type - ami-6f68cf0f
Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs   Virtualization type: hvm

SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-e4a30084
SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs   Virtualization type: hvm

Ubuntu Server 16.04 LTS (HVM), SSD Volume Type - ami-a58d0dc5



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.

Filter by: All instance types · Current generation · Show/Hide Columns

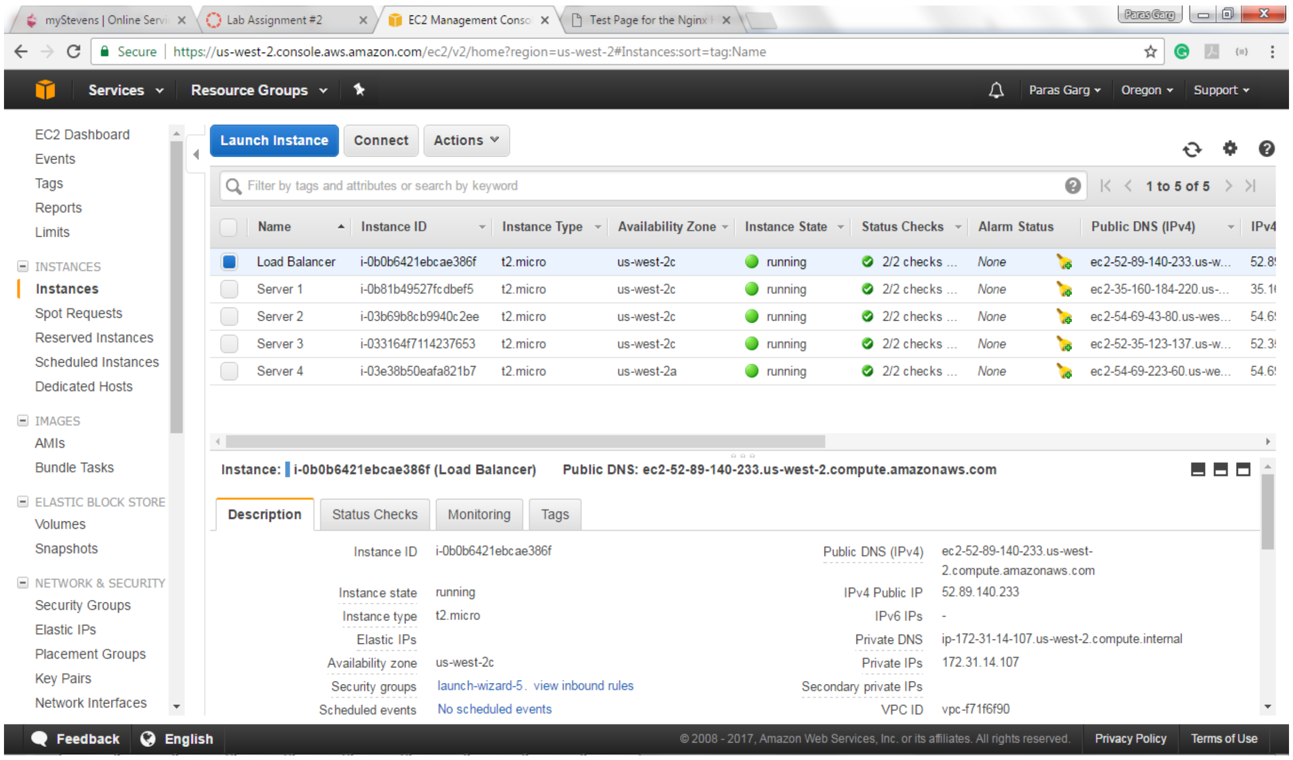
Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes

  - In my case

AMI	: **ami-8ca83fec**
Group Id	: **sg-55cb0f2e**
Instance Count	: **5**
Instance Type	: **t2.micro**
Key Name	: **ParasGarg_AWS**

- Now, our 5 instances have been created. I have created five instances in total and named them as Load Balancer, Server 1, Server 2, Server 3, and Server 4.



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#Instances:sort=tag:Name

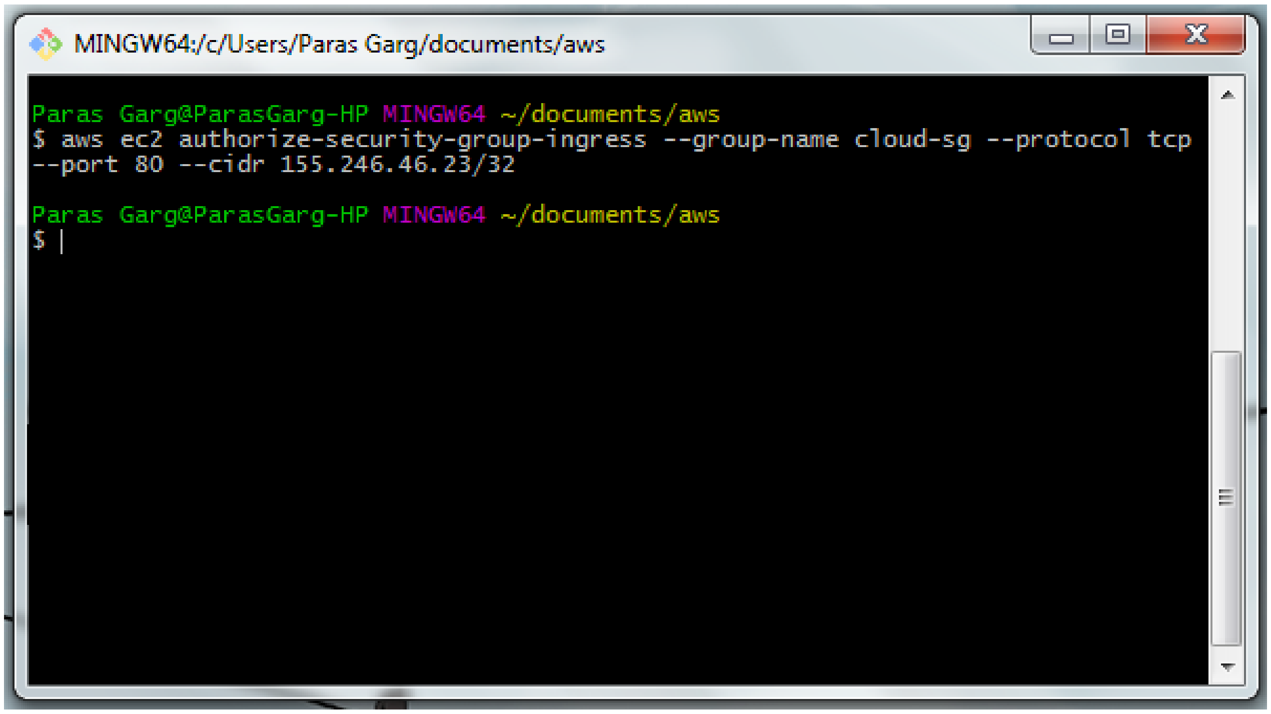
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4
Load Balancer	i-0b0b6421ebcae386f	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-52-89-140-233.us-w...	52.8
Server 1	i-0b81b49527fcdbef5	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-35-160-184-220.us-...	35.1
Server 2	i-03b69b8cb9940c2ee	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-54-69-43-80.us-wes...	54.6
Server 3	i-033164f7114237653	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-52-35-123-137.us-w...	52.3
Server 4	i-03e38b50eafa821b7	t2.micro	us-west-2a	running	2/2 checks ...	None	ec2-54-69-223-60.us-we...	54.6

Instance: i-0b0b6421ebcae386f (Load Balancer) Public DNS: ec2-52-89-140-233.us-west-2.compute.amazonaws.com

Instance ID	i-0b0b6421ebcae386f	Public DNS (IPv4)	ec2-52-89-140-233.us-west-2.compute.amazonaws.com
Instance state	running	IPv4 Public IP	52.89.140.233
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-14-107.us-west-2.compute.internal
Availability zone	us-west-2c	Private IPs	172.31.14.107
Security groups	launch-wizard-5. view inbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-f71f6f90

- The last step is to add HTTP port i.e. 80 to the security group for inbound access.

```
$ aws ec2 authorize-security-group-ingress –group-name cloud-sg –protocol tcp –port 80 –cidr 155.246.46.23/32
```

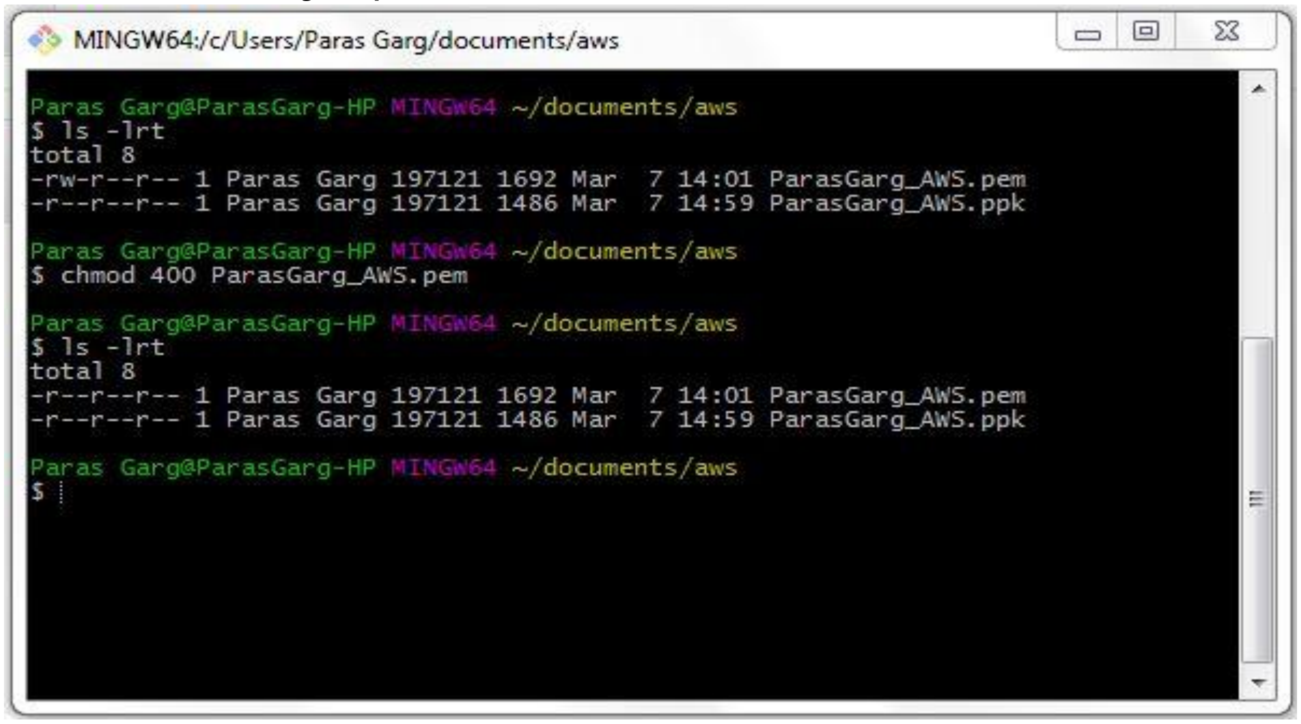


MINGW64:/c/Users/Paras Garg/documents/aws

```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 authorize-security-group-ingress --group-name cloud-sg --protocol tcp
--port 80 --cidr 155.246.46.23/32

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ |
```

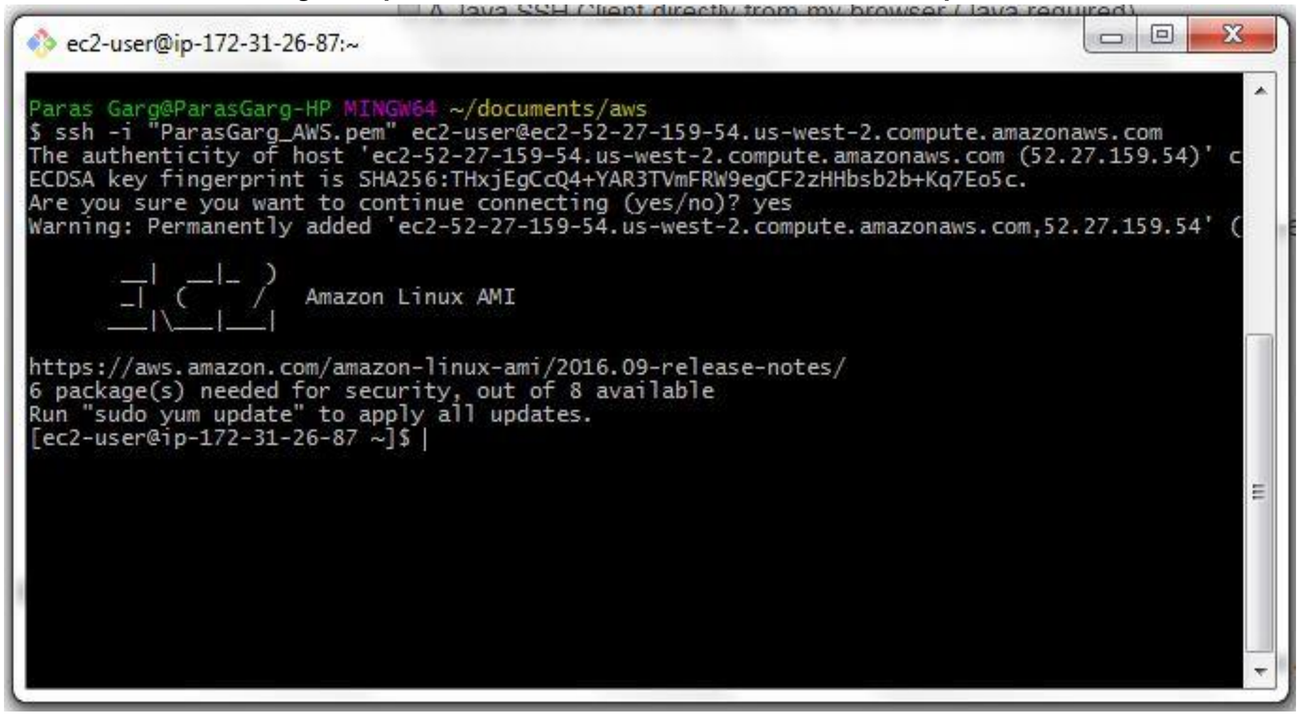
Step for Accessing AWS instance

- Ensuring read write permission on instance by executing below command
  $ chmod 400 ParasGarg_AW.pem



```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ ls -lrt
total 8
-rw-r--r-- 1 Paras Garg 197121 1692 Mar  7 14:01 ParasGarg_AWS.pem
-r--r--r-- 1 Paras Garg 197121 1486 Mar  7 14:59 ParasGarg_AWS.ppk

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ chmod 400 ParasGarg_AWS.pem

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ ls -lrt
total 8
-r--r--r-- 1 Paras Garg 197121 1692 Mar  7 14:01 ParasGarg_AWS.pem
-r--r--r-- 1 Paras Garg 197121 1486 Mar  7 14:59 ParasGarg_AWS.ppk

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$
```

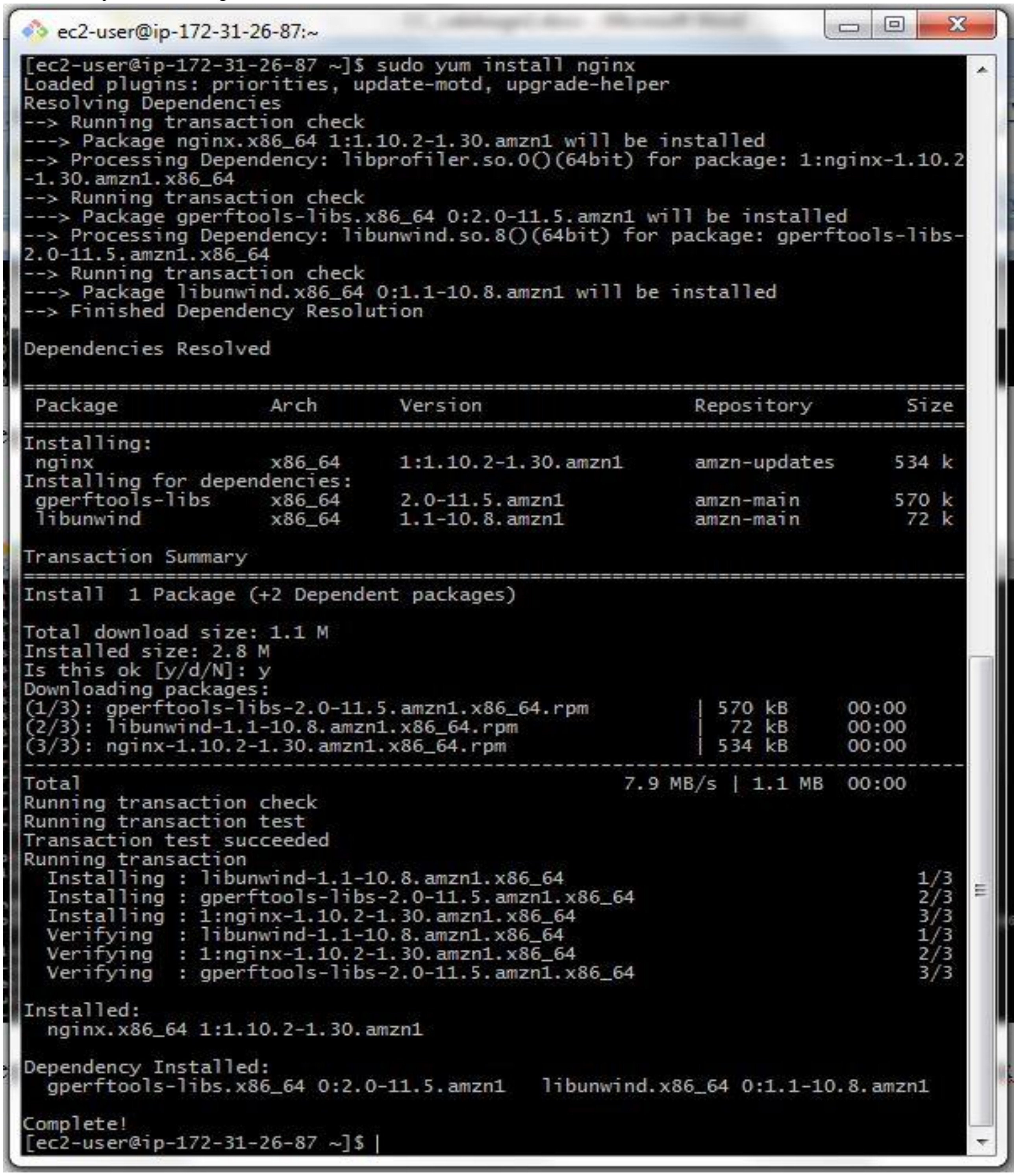
- Establishing connection with EC2 Instance by executing below command
  $ ssh –I "ParasGarg_AWS.pem" ec2-user@ec2-52-27-159-54.us-west-2.compute.amazonaws.com



```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ ssh -i "ParasGarg_AWS.pem" ec2-user@ec2-52-27-159-54.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-52-27-159-54.us-west-2.compute.amazonaws.com (52.27.159.54)' c
ECDSA key fingerprint is SHA256:THxjEgCcQ4+YAR3TVmFRW9egCF2zHHbsb2b+Kq7Eo5c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-52-27-159-54.us-west-2.compute.amazonaws.com,52.27.159.54' (

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

https://aws.amazon.com/amazon-linux-ami/2016.09-release-notes/
6 package(s) needed for security, out of 8 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-26-87 ~]$ |
```

Steps to install Nginx Server on Amazon EC2 instance

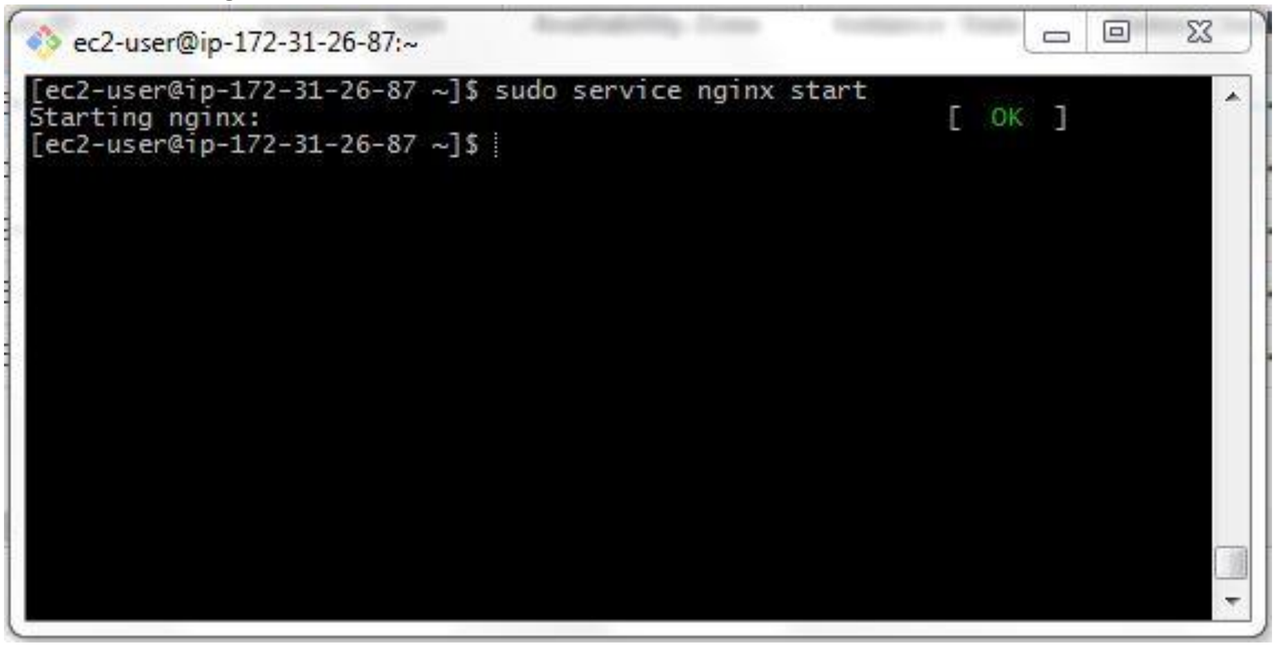
- After establish a connection with the EC2 instance, installing nginx server on it by executing the below command.
  $ sudo yum install nginx



```
[ec2-user@ip-172-31-26-87 ~]$ sudo yum install nginx
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
---> Package nginx.x86_64 1:1.10.2-1.30.amzn1 will be installed
--> Processing Dependency: libprofiler.so.0()(64bit) for package: 1:nginx-1.10.2
-1.30.amzn1.x86_64
--> Running transaction check
---> Package gperftools-libs.x86_64 0:2.0-11.5.amzn1 will be installed
--> Processing Dependency: libunwind.so.8()(64bit) for package: gperftools-libs-
2.0-11.5.amzn1.x86_64
--> Running transaction check
---> Package libunwind.x86_64 0:1.1-10.8.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package              Arch       Version                  Repository       Size
================================================================================
Installing:
 nginx                x86_64     1:1.10.2-1.30.amzn1      amzn-updates    534 k
Installing for dependencies:
 gperftools-libs      x86_64     2.0-11.5.amzn1           amzn-main       570 k
 libunwind            x86_64     1.1-10.8.amzn1           amzn-main        72 k

Transaction Summary
================================================================================
Install  1 Package (+2 Dependent packages)

Total download size: 1.1 M
Installed size: 2.8 M
Is this ok [y/d/N]: y
Downloading packages:
(1/3): gperftools-libs-2.0-11.5.amzn1.x86_64.rpm         | 570 kB     00:00
(2/3): libunwind-1.1-10.8.amzn1.x86_64.rpm               |  72 kB     00:00
(3/3): nginx-1.10.2-1.30.amzn1.x86_64.rpm                | 534 kB     00:00
--------------------------------------------------------------------------------
Total                                          7.9 MB/s | 1.1 MB  00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libunwind-1.1-10.8.amzn1.x86_64                              1/3
  Installing : gperftools-libs-2.0-11.5.amzn1.x86_64                        2/3
  Installing : 1:nginx-1.10.2-1.30.amzn1.x86_64                             3/3
  Verifying  : libunwind-1.1-10.8.amzn1.x86_64                              1/3
  Verifying  : 1:nginx-1.10.2-1.30.amzn1.x86_64                             2/3
  Verifying  : gperftools-libs-2.0-11.5.amzn1.x86_64                        3/3

Installed:
  nginx.x86_64 1:1.10.2-1.30.amzn1

Dependency Installed:
  gperftools-libs.x86_64 0:2.0-11.5.amzn1   libunwind.x86_64 0:1.1-10.8.amzn1

Complete!
[ec2-user@ip-172-31-26-87 ~]$ |
```

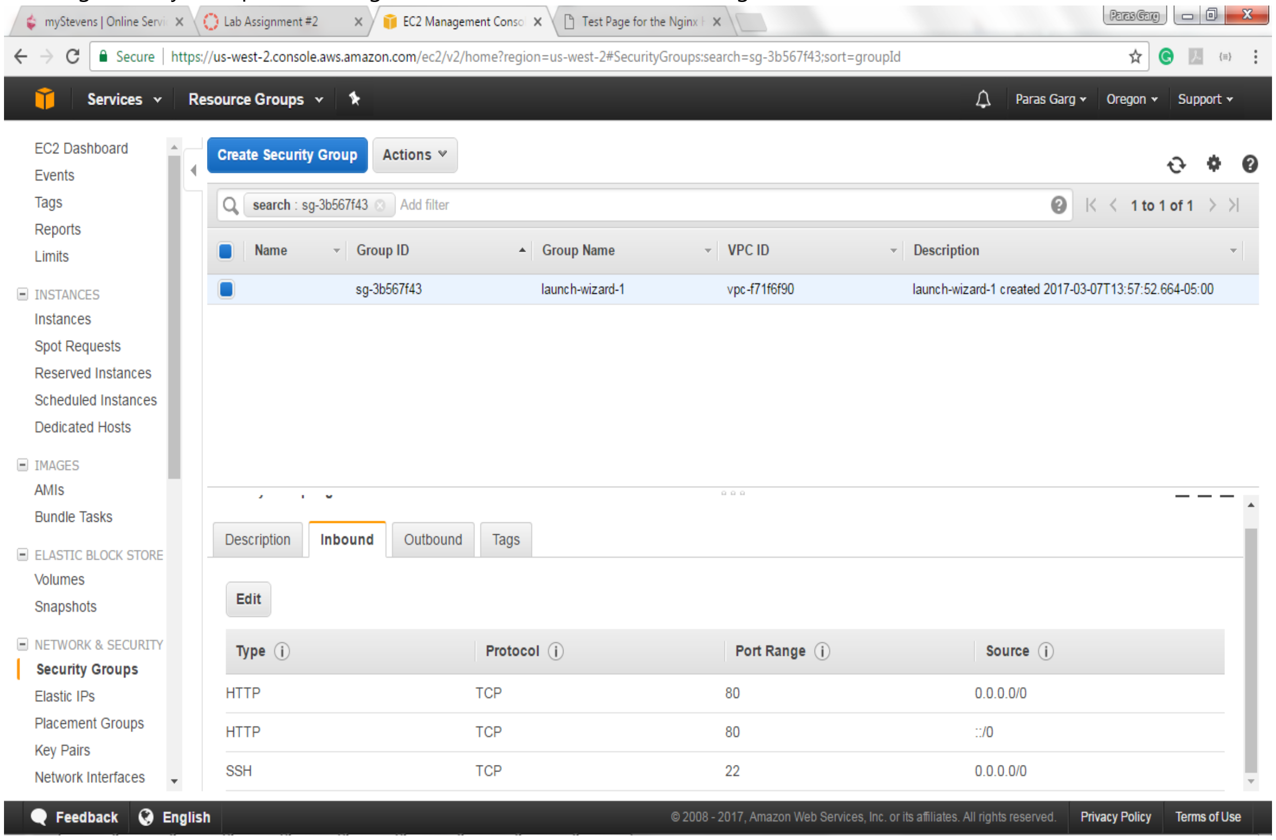
*(Click **y** for to download and install packages)*

- After installing nginx, starting the services of the nginx by executing
  $ sudo service nginx start



```
[ec2-user@ip-172-31-26-87 ~]$ sudo service nginx start
Starting nginx:                                            [  OK  ]
[ec2-user@ip-172-31-26-87 ~]$
```

- Checking Security Group and adding inbound rule for HTTP for running server on instance.



https://us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2#SecurityGroups:search=sg-3b567f43;sort=groupId

search : sg-3b567f43

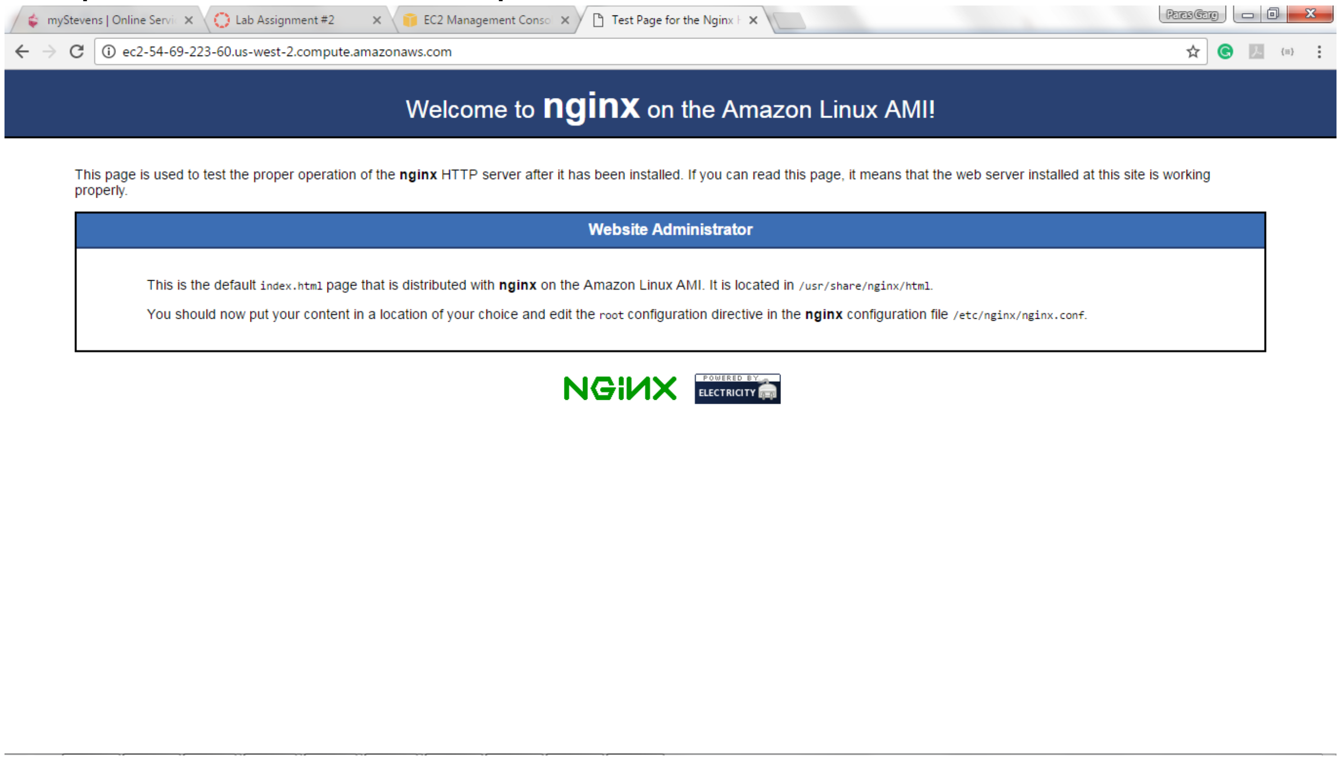
Name	Group ID	Group Name	VPC ID	Description
	sg-3b567f43	launch-wizard-1	vpc-f71f6f90	launch-wizard-1 created 2017-03-07T13:57:52.664-05:00

Inbound

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
SSH	TCP	22	0.0.0.0/0

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

- Testing server by running **Public DNS (IPv4)** on the browser.
  http://ec2-54-69-223-60.us-west-2.compute.amazonaws.com/



Welcome to **nginx** on the Amazon Linux AMI!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default `index.html` page that is distributed with **nginx** on the Amazon Linux AMI. It is located in `/usr/share/nginx/html`.

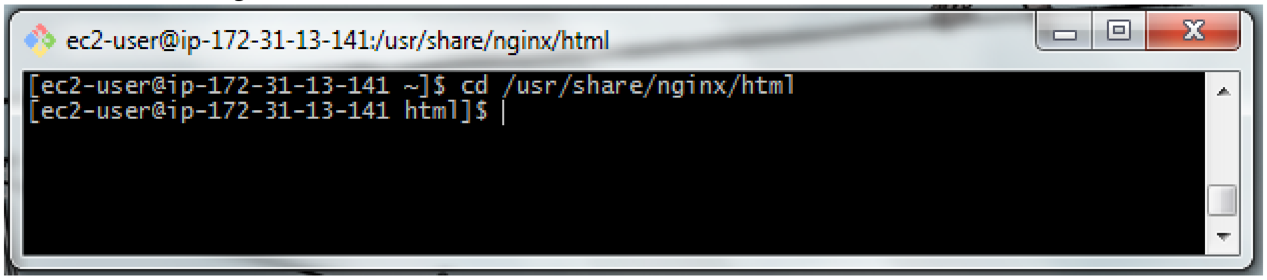
You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

- Now, repeat the same steps to install Nginx Server on each instance you want to install. I have installed the nginx server on each instance i.e. Server 1, Server 2, Server 3, Server 4, and Load Balancer.

Steps to change nginx server index.html file on Amazon EC2 instance

- After successfully installing nginx server, navigate to /usr/share/nginx/html directory. To navigate type following command in the terminal window

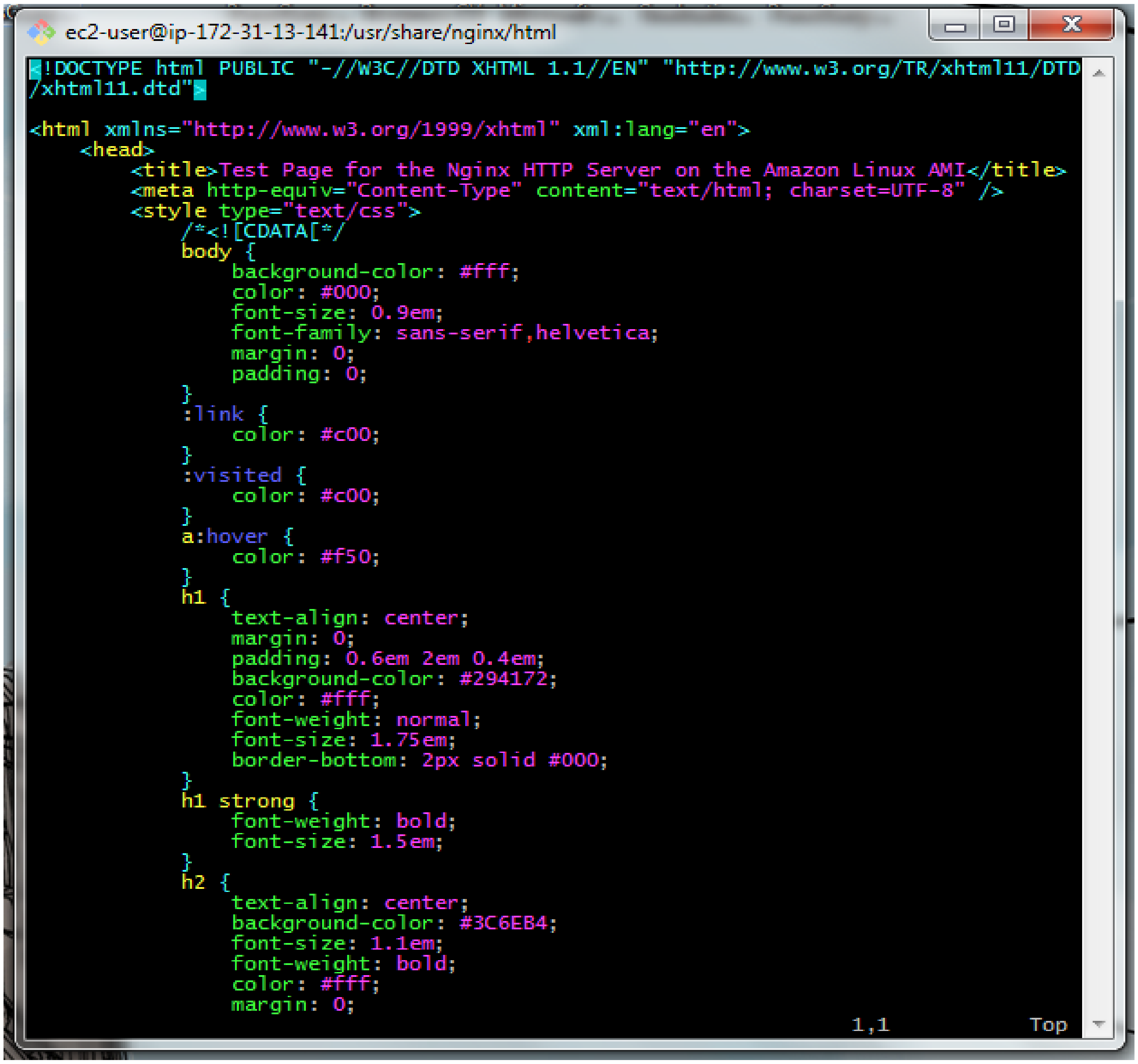
  $ cd /usr/share/nginx/html



```
ec2-user@ip-172-31-13-141:/usr/share/nginx/html
[ec2-user@ip-172-31-13-141 ~]$ cd /usr/share/nginx/html
[ec2-user@ip-172-31-13-141 html]$ |
```

- Then open the index.html file in **vim**. To open type following command in the terminal window

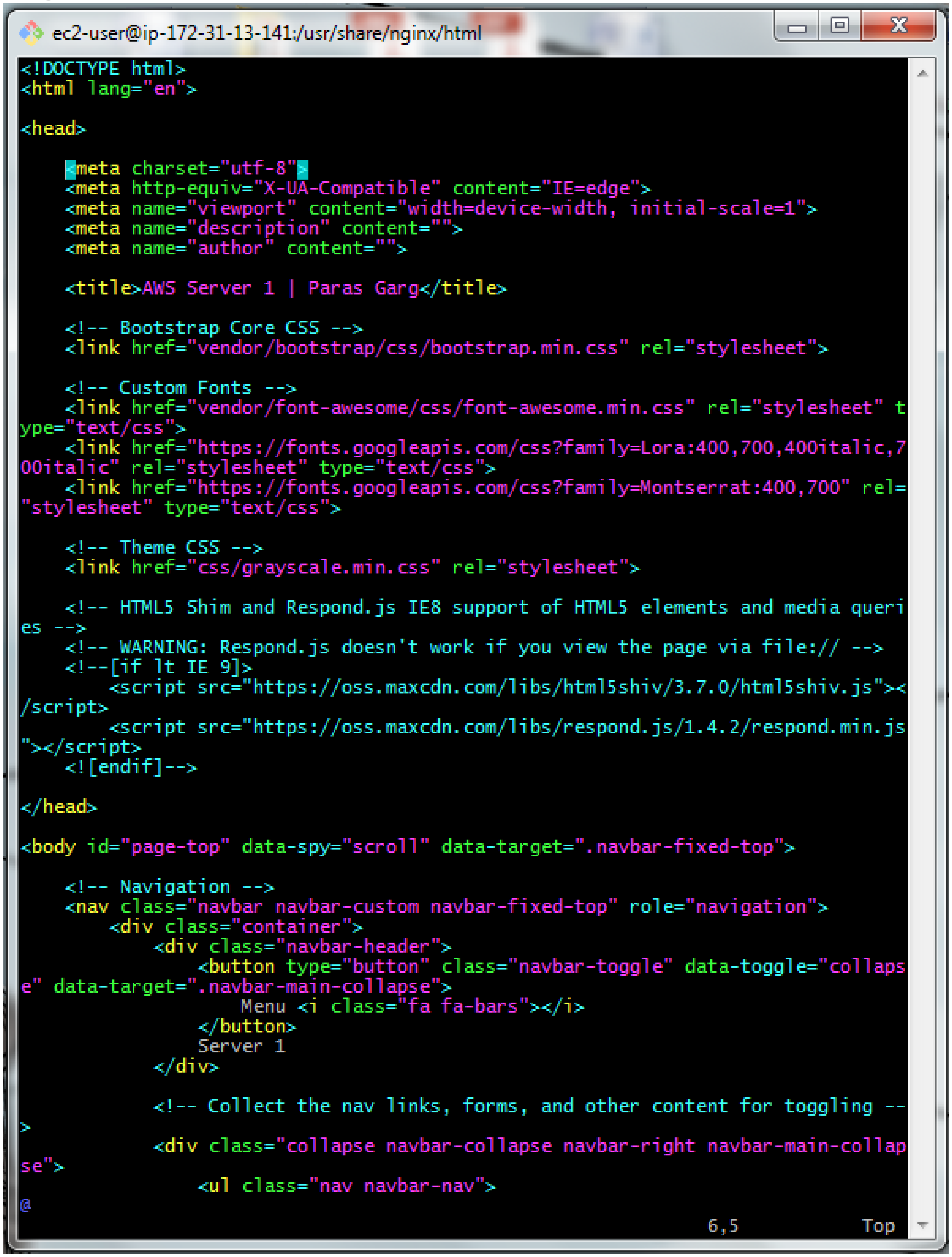
  $ sudo vim index.html



```
ec2-user@ip-172-31-13-141:/usr/share/nginx/html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD
/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on the Amazon Linux AMI</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
            body {
                background-color: #fff;
                color: #000;
                font-size: 0.9em;
                font-family: sans-serif,helvetica;
                margin: 0;
                padding: 0;
            }
            :link {
                color: #c00;
            }
            :visited {
                color: #c00;
            }
            a:hover {
                color: #f50;
            }
            h1 {
                text-align: center;
                margin: 0;
                padding: 0.6em 2em 0.4em;
                background-color: #294172;
                color: #fff;
                font-weight: normal;
                font-size: 1.75em;
                border-bottom: 2px solid #000;
            }
            h1 strong {
                font-weight: bold;
                font-size: 1.5em;
            }
            h2 {
                text-align: center;
                background-color: #3C6EB4;
                font-size: 1.1em;
                font-weight: bold;
                color: #fff;
                margin: 0;
                                                                  1,1           Top
```

(***sudo*** command allows you to write a read-only else you can write **:w !sudo tee % > /dev/null** after pressing escape and colon before qutting)

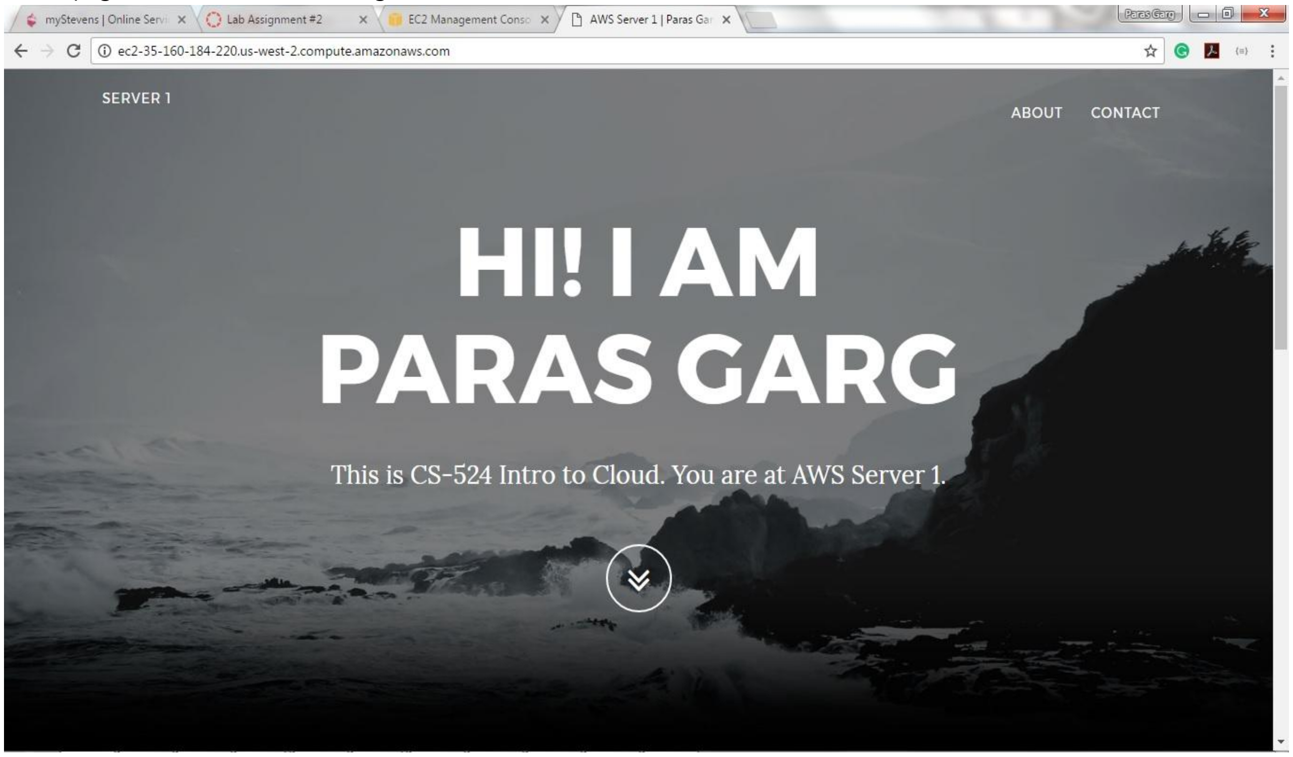
- Editing the index.html file for Server 1 or instance number 1.



ec2-user@ip-172-31-13-141:/usr/share/nginx/html

```
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>AWS Server 1 | Paras Garg</title>

    <!-- Bootstrap Core CSS -->
    <link href="vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <link href="vendor/font-awesome/css/font-awesome.min.css" rel="stylesheet" t
ype="text/css">
    <link href="https://fonts.googleapis.com/css?family=Lora:400,700,400italic,7
00italic" rel="stylesheet" type="text/css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:400,700" rel=
"stylesheet" type="text/css">

    <!-- Theme CSS -->
    <link href="css/grayscale.min.css" rel="stylesheet">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queri
es -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
        <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"><
/script>
        <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js
"></script>
    <![endif]-->

</head>

<body id="page-top" data-spy="scroll" data-target=".navbar-fixed-top">

    <!-- Navigation -->
    <nav class="navbar navbar-custom navbar-fixed-top" role="navigation">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collaps
e" data-target=".navbar-main-collapse">
                    Menu <i class="fa fa-bars"></i>
                </button>
                Server 1
            </div>

            <!-- Collect the nav links, forms, and other content for toggling --
>
            <div class="collapse navbar-collapse navbar-right navbar-main-collap
se">
                <ul class="nav navbar-nav">
@
                                                              6,5           Top
```

- Actual page on browser after editing index.html file over AWS EC2 for "Server 1" or Instance number 1.



HI! I AM PARAS GARG

This is CS-524 Intro to Cloud. You are at AWS Server 1.

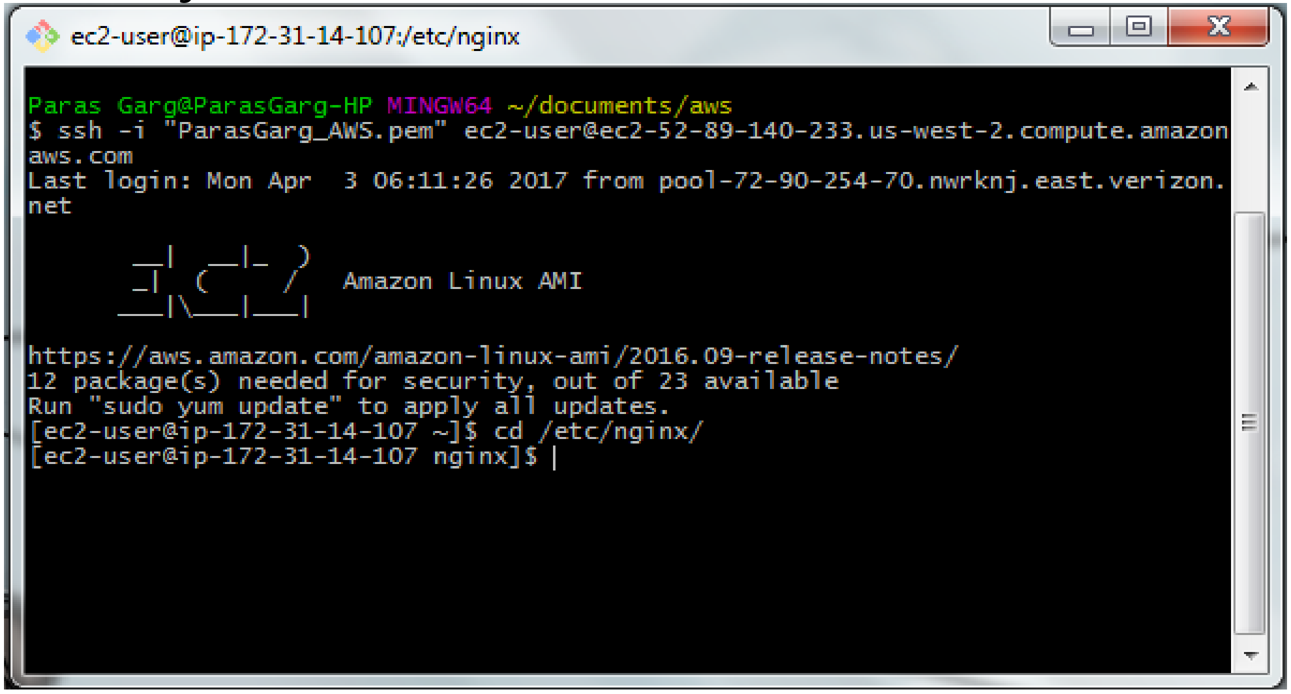
- Now, repeat the same steps to edit the index.html file on each instance you want. I have edited the index.html file for all my four instances i.e. Server 1, Server 2, Server 3, and Server 4. Please find the links below:
    - Server 1
      http://ec2-35-160-184-220.us-west-2.compute.amazonaws.com/
      http://35.160.184.220/
    - Server 2
      http://ec2-54-69-43-80.us-west-2.compute.amazonaws.com/
      http://54.69.43.80/
    - Server 3
      http://ec2-52-35-123-137.us-west-2.compute.amazonaws.com/
      http://52.35.123.137/
    - Server 4
      http://ec2-54-69-223-60.us-west-2.compute.amazonaws.com/
      http://54.69.223.60/

(Note, we can also edit index.html file of the Load Balancer instance. But there would be no use of modifying it, because whenever you try to hit/visit a public IP or DNS of Load Balancer, it would redirect you on one of the servers it is dealing with in my case it would redirect to one of the above servers. In next step, we will be handling this amazing feature of Load Balancer, and observer how it works to balance the load by redirecting to one of its servers.

Steps to configure Load Balancer on Amazon EC2 instance

- First to configure the Load Balance, connect to Load Balance instance. Then navigate to **/etc/nginx/** by executing the following commands in the terminal.

  $ ssh -i "ParasGarg_AWS.pem" ec2-user@ec2-52-89-140-233.us-west-.compute.amazonaws.com
  $ cd /etc/nginx/

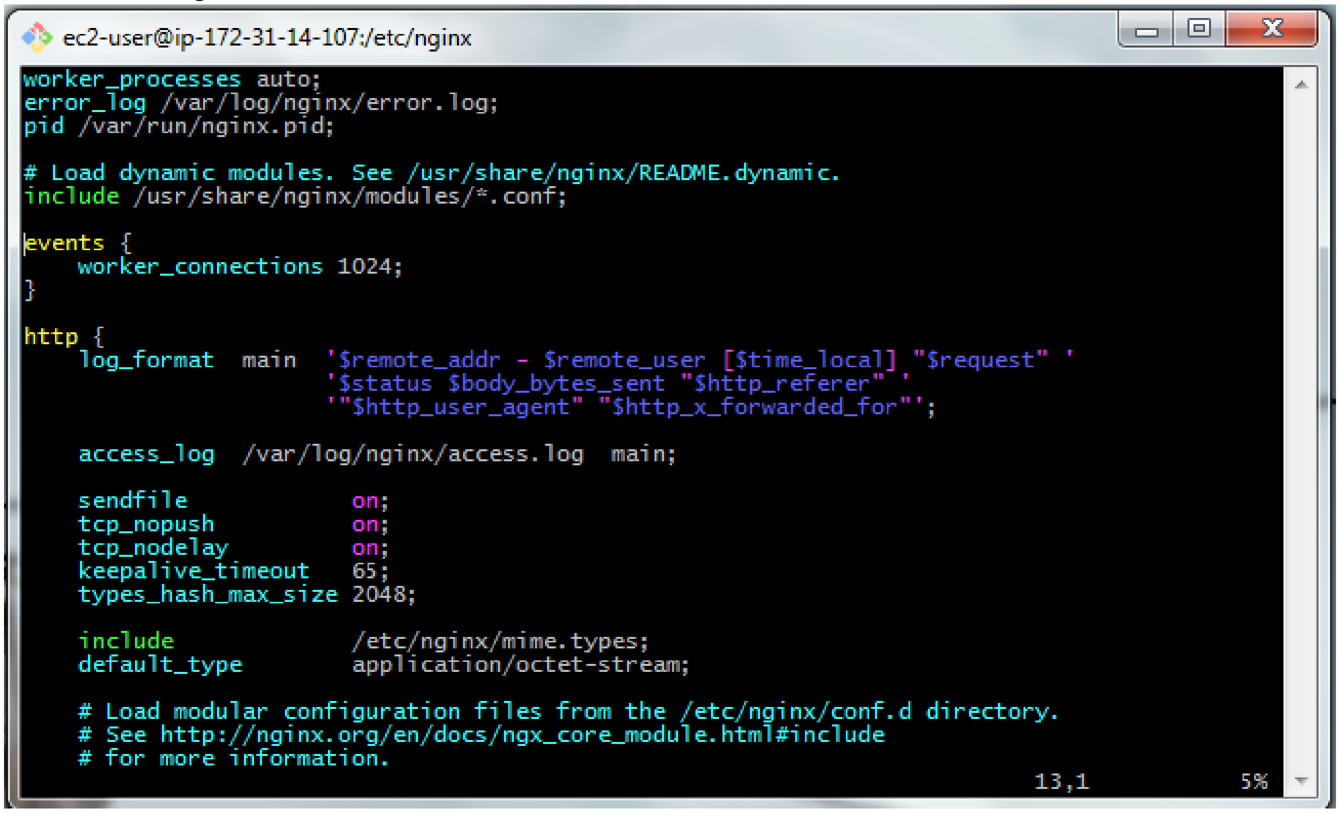


ec2-user@ip-172-31-14-107:/etc/nginx

```
Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ ssh -i "ParasGarg_AWS.pem" ec2-user@ec2-52-89-140-233.us-west-2.compute.amazon
aws.com
Last login: Mon Apr  3 06:11:26 2017 from pool-72-90-254-70.nwrknj.east.verizon.
net

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

https://aws.amazon.com/amazon-linux-ami/2016.09-release-notes/
12 package(s) needed for security, out of 23 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-14-107 ~]$ cd /etc/nginx/
[ec2-user@ip-172-31-14-107 nginx]$ |
```

- Now open **nginx.conf** using vim by executing following command

  $ sudo vim nginx.conf



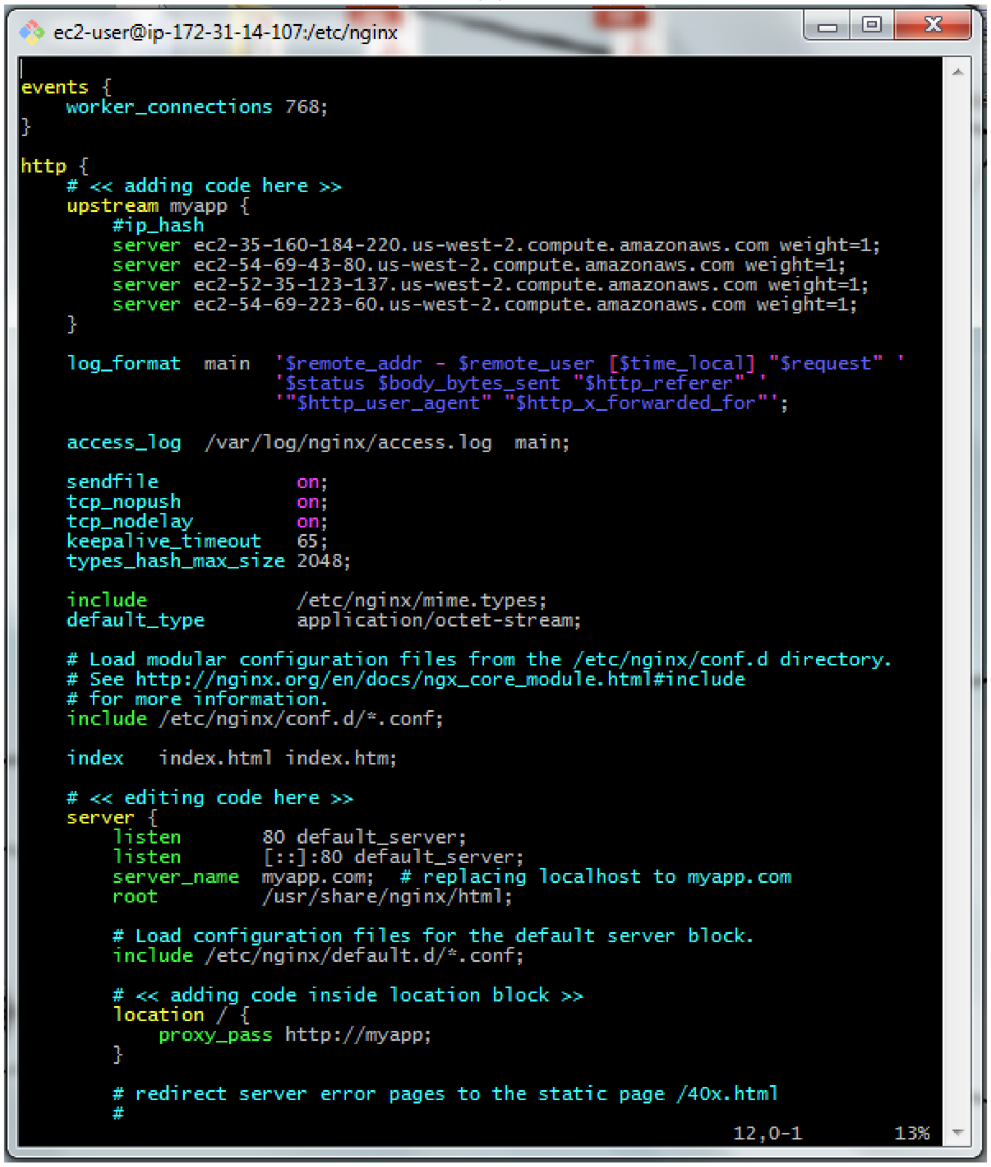
ec2-user@ip-172-31-14-107:/etc/nginx

```
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

# Load dynamic modules. See /usr/share/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
                                                              13,1          5%
```

- Now editing file by adding and replacing code by following codes

```
events {
worker_connections 768;
}
http {
upstream myapp {
#ip_hash;
            server [SERVER_PUBLIC_DNS_NAME] weight=1;
            server [SERVER_PUBLIC_DNS_NAME] weight=1;
            server [SERVER_PUBLIC_DNS_NAME] weight=1;
            server [SERVER_PUBLIC_DNS_NAME] weight=1;
      }
      server {
            listen 80;
            server_name myapp.com;
            location / {
                  proxy_pass http://myapp;
            }
      }
}
```

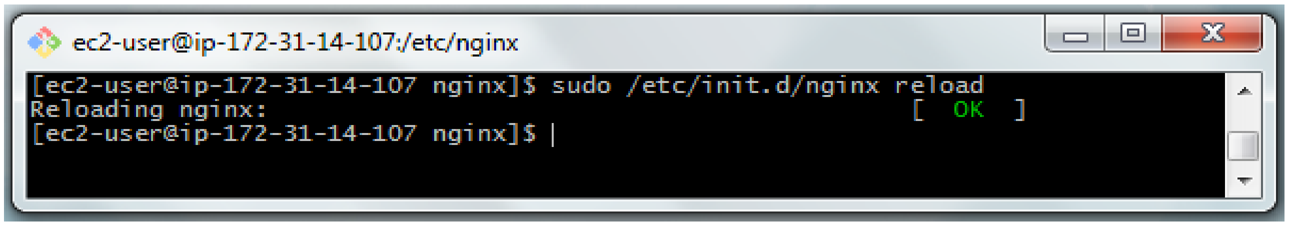
(**Note**: *SERVER_PUBLIC_DNS_NAME* would be replaced by your instance Public DNS)



ec2-user@ip-172-31-14-107:/etc/nginx

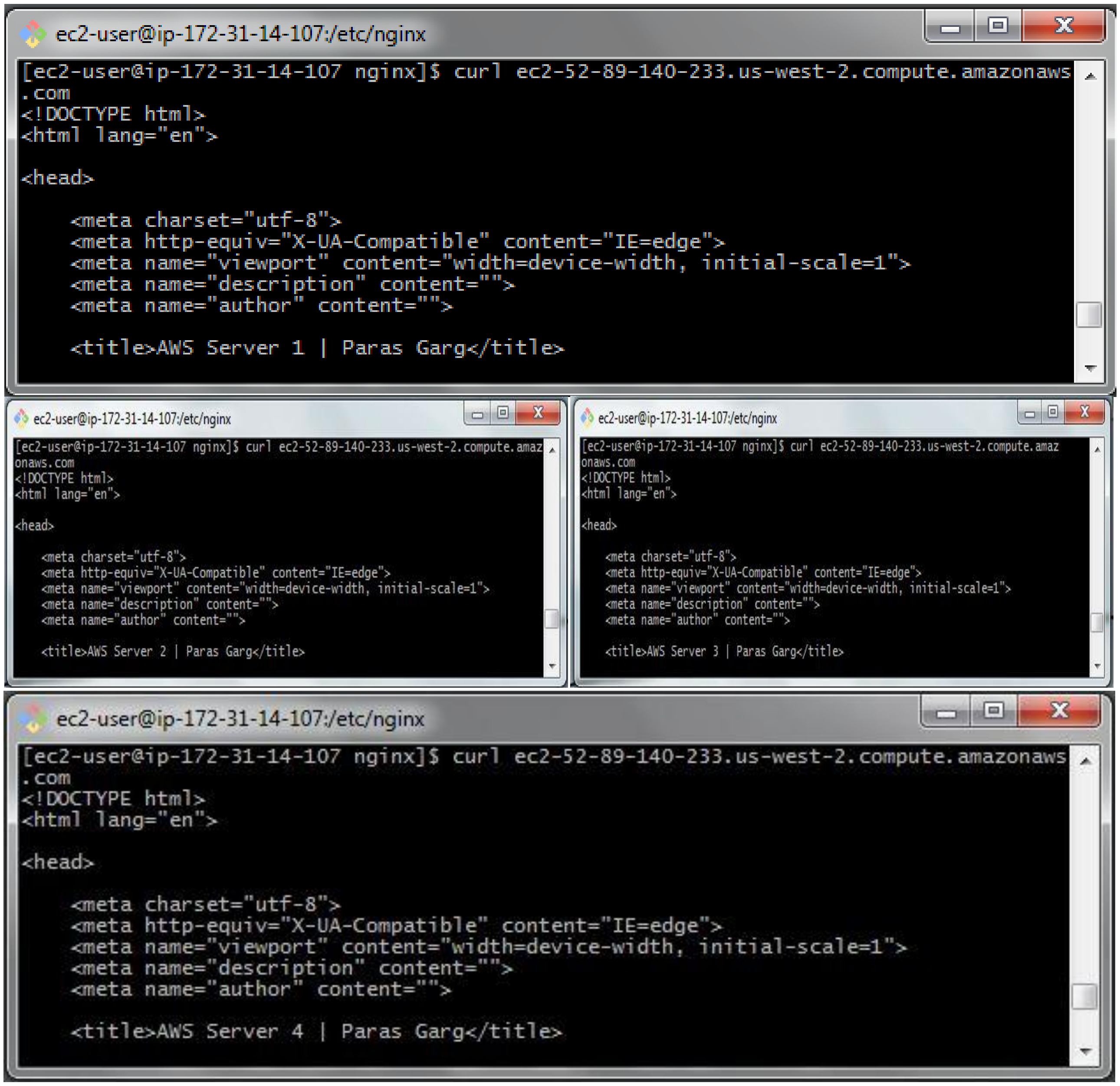
```
events {
    worker_connections 768;
}

http {
    # << adding code here >>
    upstream myapp {
        #ip_hash
        server ec2-35-160-184-220.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-43-80.us-west-2.compute.amazonaws.com weight=1;
        server ec2-52-35-123-137.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-223-60.us-west-2.compute.amazonaws.com weight=1;
    }

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;

    index   index.html index.htm;

    # << editing code here >>
    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  myapp.com;  # replacing localhost to myapp.com
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        # << adding code inside location block >>
        location / {
            proxy_pass http://myapp;
        }

        # redirect server error pages to the static page /40x.html
        #
                                                            12,0-1        13%
```

- Now, run the following command in the shell (this will cause the new configuration to take effect):
  $ /etc/init.d/nginx reload



```
[ec2-user@ip-172-31-14-107 nginx]$ sudo /etc/init.d/nginx reload
Reloading nginx:                                           [  OK  ]
[ec2-user@ip-172-31-14-107 nginx]$ |
```

- Now, we have to use the **curl** command in the shell to visit the load balancer, which will distribute traffic among ther servers. (Don't forget to change the)
  $ curl [LOAD_BALANCER_DNS_NAME]



```
[ec2-user@ip-172-31-14-107 nginx]$ curl ec2-52-89-140-233.us-west-2.compute.amazonaws
.com
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>AWS Server 1 | Paras Garg</title>
```

```
[ec2-user@ip-172-31-14-107 nginx]$ curl ec2-52-89-140-233.us-west-2.compute.amaz
onaws.com
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>AWS Server 2 | Paras Garg</title>
```

```
[ec2-user@ip-172-31-14-107 nginx]$ curl ec2-52-89-140-233.us-west-2.compute.amaz
onaws.com
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>AWS Server 3 | Paras Garg</title>
```

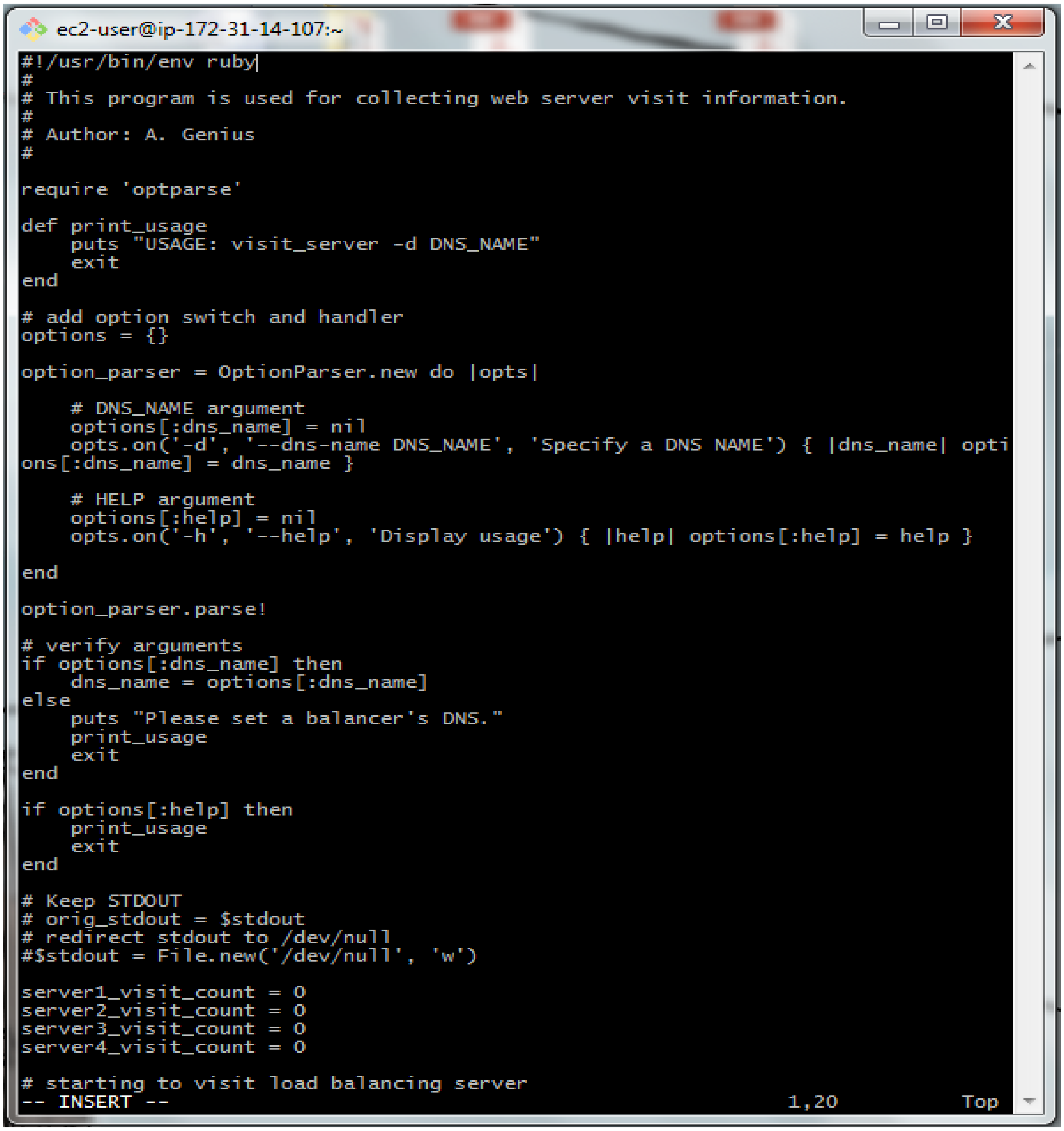
```
[ec2-user@ip-172-31-14-107 nginx]$ curl ec2-52-89-140-233.us-west-2.compute.amazonaws
.com
<!DOCTYPE html>
<html lang="en">

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">

    <title>AWS Server 4 | Paras Garg</title>
```

(**Notice** that on each curl command, the load balancer is distributting traffic to each server sequentially)

Steps to collect information on visits to your website using Amazon EC2 instance

- Setting up Visit Server tool to track the distribution of the load. This tool visits the cluster 2000 times and returns the visit count on each server

  $ vim visit_server

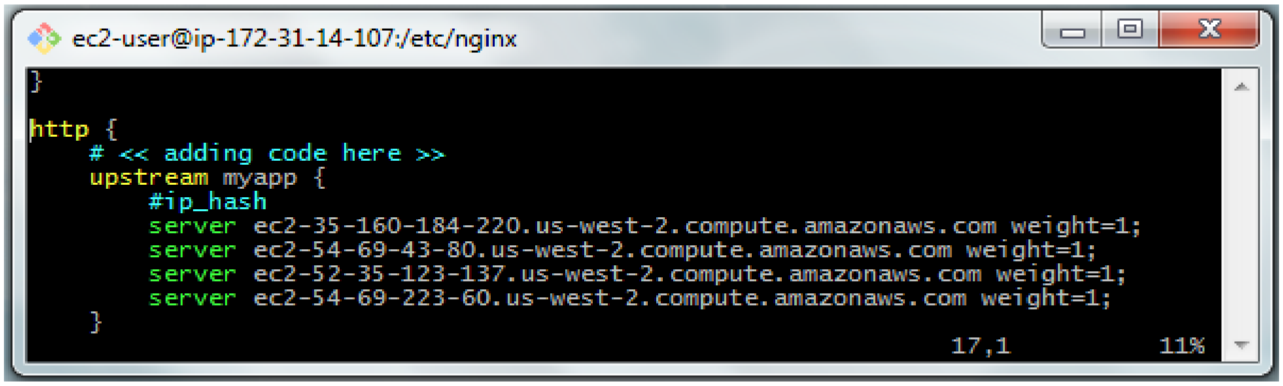


ec2-user@ip-172-31-14-107:~

```
#!/usr/bin/env ruby
#
# This program is used for collecting web server visit information.
#
# Author: A. Genius
#

require 'optparse'

def print_usage
    puts "USAGE: visit_server -d DNS_NAME"
    exit
end

# add option switch and handler
options = {}

option_parser = OptionParser.new do |opts|

    # DNS_NAME argument
    options[:dns_name] = nil
    opts.on('-d', '--dns-name DNS_NAME', 'Specify a DNS NAME') { |dns_name| opti
ons[:dns_name] = dns_name }

    # HELP argument
    options[:help] = nil
    opts.on('-h', '--help', 'Display usage') { |help| options[:help] = help }

end

option_parser.parse!

# verify arguments
if options[:dns_name] then
    dns_name = options[:dns_name]
else
    puts "Please set a balancer's DNS."
    print_usage
    exit
end

if options[:help] then
    print_usage
    exit
end

# Keep STDOUT
# orig_stdout = $stdout
# redirect stdout to /dev/null
#$stdout = File.new('/dev/null', 'w')

server1_visit_count = 0
server2_visit_count = 0
server3_visit_count = 0
server4_visit_count = 0

# starting to visit load balancing server
-- INSERT --                                                    1,20          Top
```

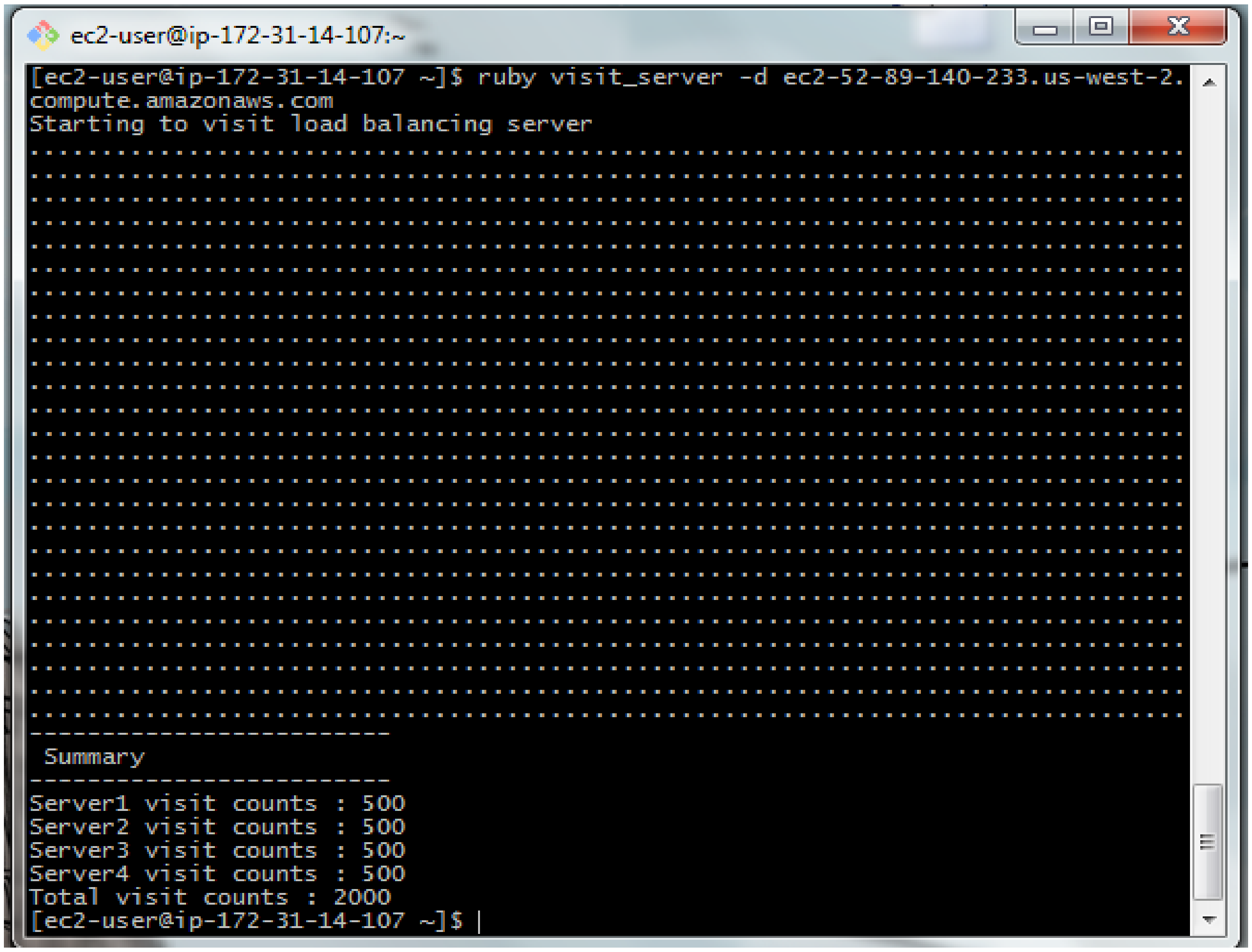
- Now, we will trace the load balancing server load distribution for 3 scenarios by changing the server weight in the **nginx.conf** file (which we edited in previous step).
  - Scenario #1 – Server1 weight=1, Server2 weight=1, Server3 weight=1, Server4 weight=1
  - Scenario #2 – Server1 weight=1, Server2 weight=2, Server3 weight=3, Server4 weight=1
  - Scenario #3 – Server1 weight=1, Server2 weight=2, Server3 weight=1, Server4 weight=2

- Tracing load balancer for Scenario #1 – Server1 weight=1, Server2 weight=1, Server3 weight=1, Server4 weight=1



```
}

http {
    # << adding code here >>
    upstream myapp {
        #ip_hash
        server ec2-35-160-184-220.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-43-80.us-west-2.compute.amazonaws.com weight=1;
        server ec2-52-35-123-137.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-223-60.us-west-2.compute.amazonaws.com weight=1;
    }
                                                              17,1          11%
```

(Preview of nginx.conf file for satisfying Scenario #1)

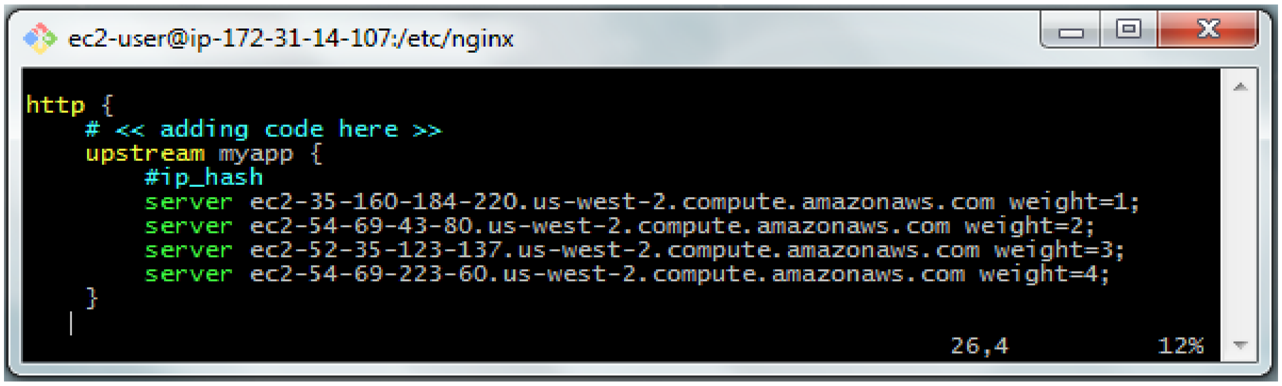


```
[ec2-user@ip-172-31-14-107 ~]$ ruby visit_server -d ec2-52-89-140-233.us-west-2.
compute.amazonaws.com
Starting to visit load balancing server
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
-------------------------
 Summary
-------------------------
Server1 visit counts : 500
Server2 visit counts : 500
Server3 visit counts : 500
Server4 visit counts : 500
Total visit counts : 2000
[ec2-user@ip-172-31-14-107 ~]$ |
```

(Preview of Load Balancer tracing summary)

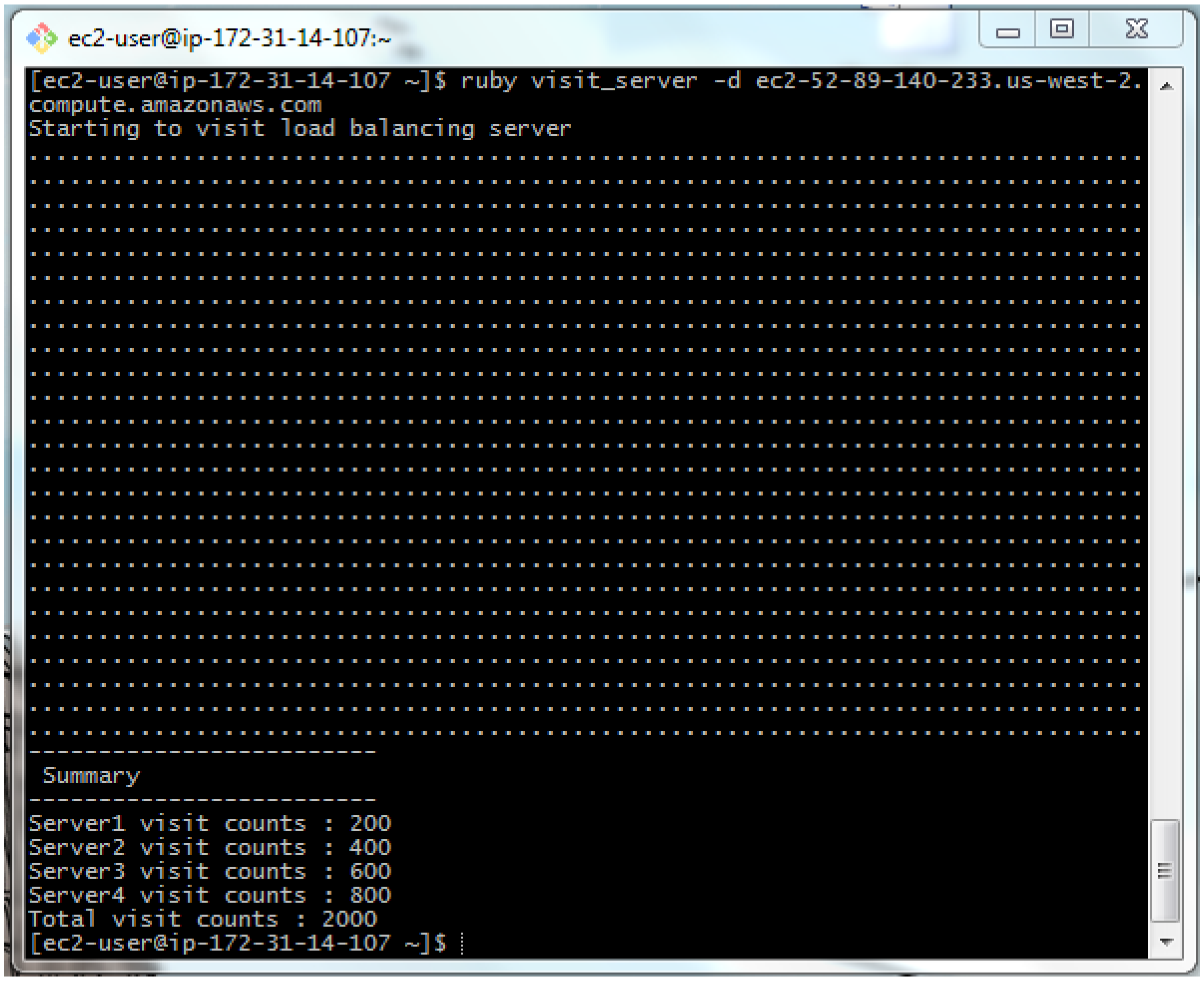
Notice, the visit counts for each server in the load balancer (like Server 1, Server 2, Server 3, and Server 4) are 500. As the weight for each server is 1. As the weight are 1, 1, 1, and 1.

- Tracing load balancer for Scenario #2 – Server1 weight=1, Server2 weight=2, Server3 weight=3, Server4 weight=4



```
ec2-user@ip-172-31-14-107:/etc/nginx

http {
    # << adding code here >>
    upstream myapp {
        #ip_hash
        server ec2-35-160-184-220.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-43-80.us-west-2.compute.amazonaws.com weight=2;
        server ec2-52-35-123-137.us-west-2.compute.amazonaws.com weight=3;
        server ec2-54-69-223-60.us-west-2.compute.amazonaws.com weight=4;
    }
   |
                                                           26,4          12%
```

(Preview of nginx.conf file for satisfying Scenario #2)



```
ec2-user@ip-172-31-14-107:~

[ec2-user@ip-172-31-14-107 ~]$ ruby visit_server -d ec2-52-89-140-233.us-west-2.
compute.amazonaws.com
Starting to visit load balancing server
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
-------------------------
 Summary
-------------------------
Server1 visit counts : 200
Server2 visit counts : 400
Server3 visit counts : 600
Server4 visit counts : 800
Total visit counts : 2000
[ec2-user@ip-172-31-14-107 ~]$
```

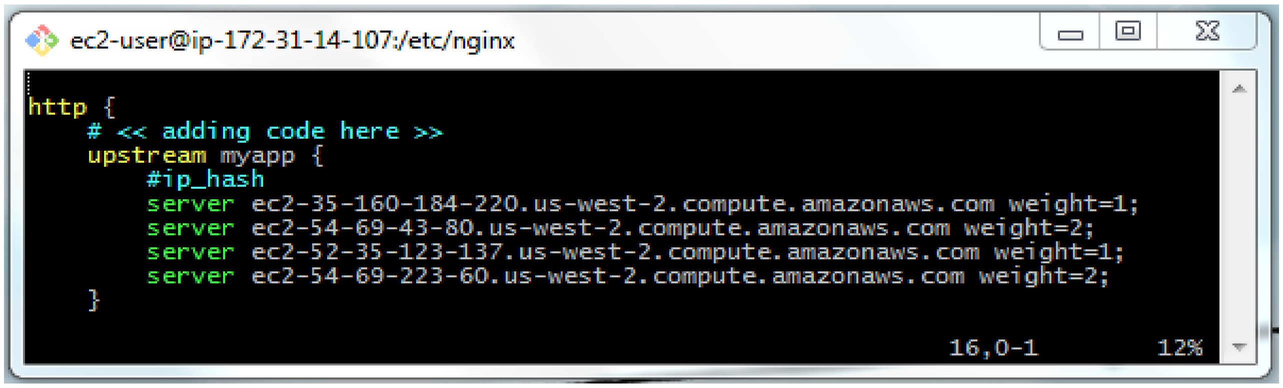
(Preview of Load Balancer tracing summary)

Notice, the visit counts for each server in the load balancer (like Server 1, Server 2, Server 3, and Server 4) is 200, 400, 600, and 800 respectively. As the weight are 1, 2, 3, and 4.

Note: nginx has been reloaded after the change by executing following command

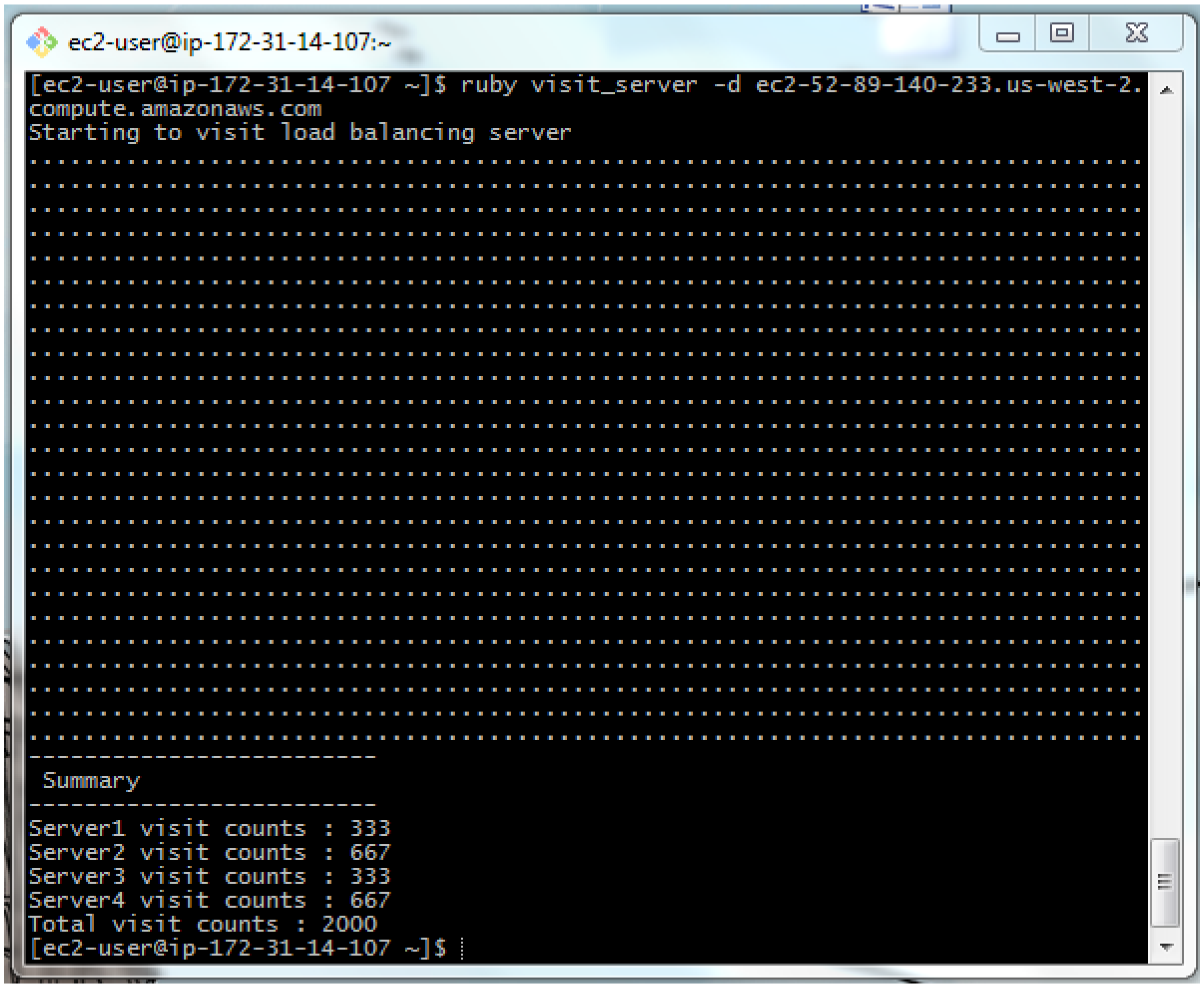
$ /etc/init.d/nginx reload

- Tracing load balancer for Scenario #3 – Server1 weight=1, Server2 weight=2, Server3 weight=1, Server4 weight=2



```
ec2-user@ip-172-31-14-107:/etc/nginx

http {
    # << adding code here >>
    upstream myapp {
        #ip_hash
        server ec2-35-160-184-220.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-43-80.us-west-2.compute.amazonaws.com weight=2;
        server ec2-52-35-123-137.us-west-2.compute.amazonaws.com weight=1;
        server ec2-54-69-223-60.us-west-2.compute.amazonaws.com weight=2;
    }
                                                              16,0-1          12%
```

(Preview of nginx.conf file for satisfying Scenario #3)



```
ec2-user@ip-172-31-14-107:~

[ec2-user@ip-172-31-14-107 ~]$ ruby visit_server -d ec2-52-89-140-233.us-west-2.
compute.amazonaws.com
Starting to visit load balancing server
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
................................................................................
-------------------------
 Summary
-------------------------
Server1 visit counts : 333
Server2 visit counts : 667
Server3 visit counts : 333
Server4 visit counts : 667
Total visit counts : 2000
[ec2-user@ip-172-31-14-107 ~]$
```

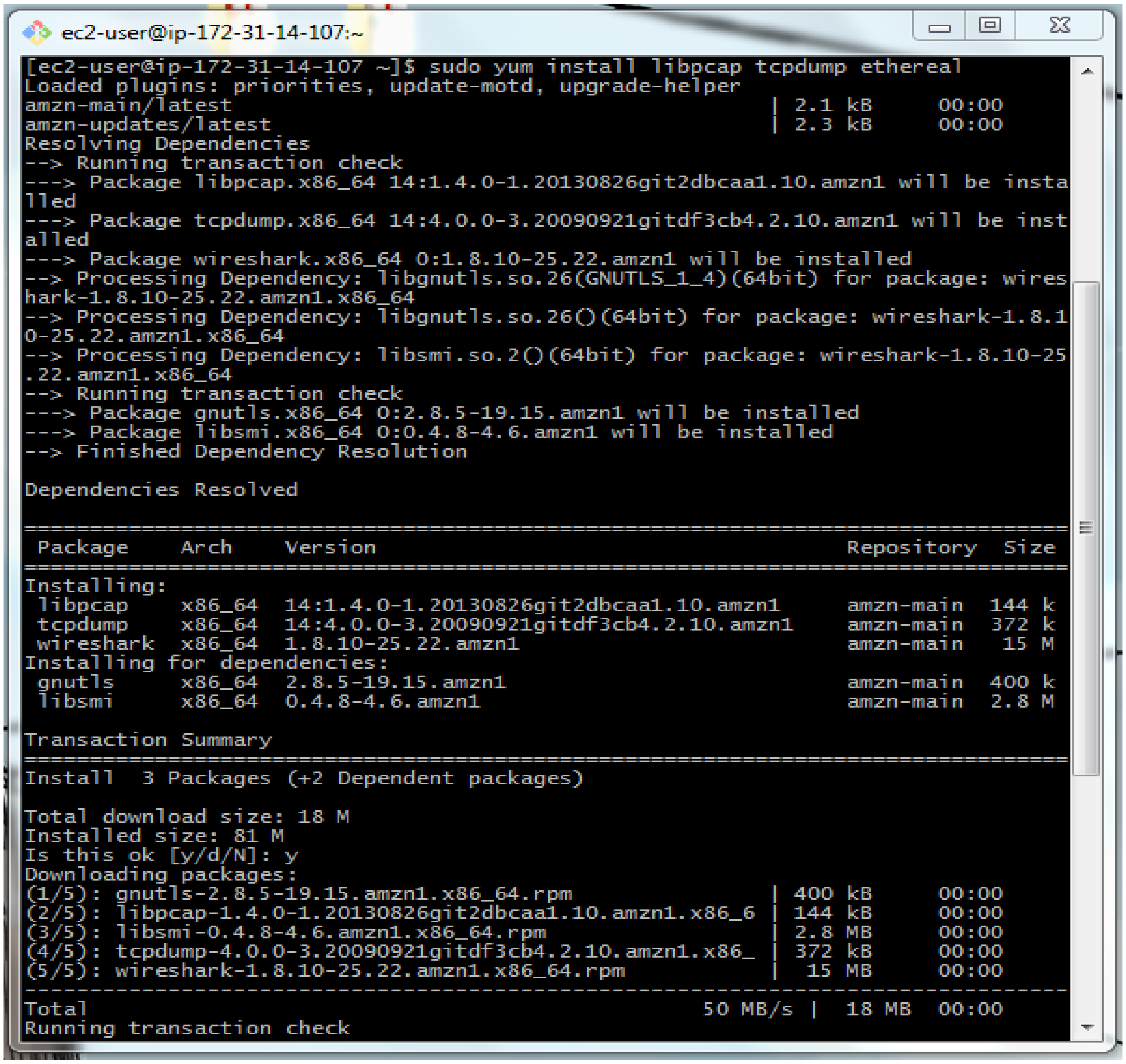
(Preview of Load Balancer tracing summary)

Notice, the visit counts for each server in the load balancer (like Server 1, Server 2, Server 3, and Server 4) is 333, 667, 333, and 667 respectively. As the weight are 1, 2, 1, and 2.

Note: nginx has been reloaded after the change by executing following command
$ /etc/init.d/nginx reload

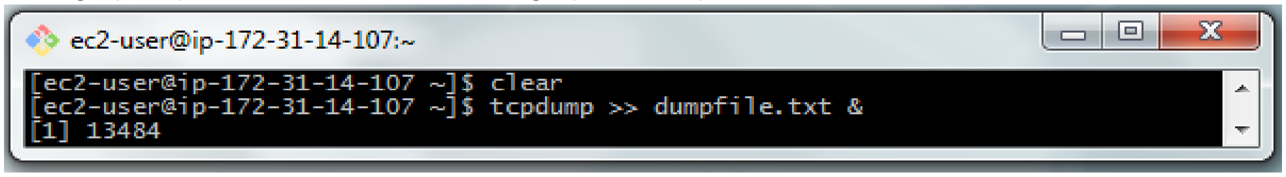
Steps to tcpdump Analysis using Amazon EC2 instance

- Installing tcpdump packages



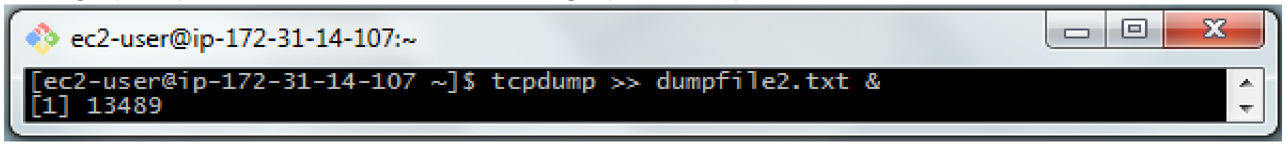
```
[ec2-user@ip-172-31-14-107 ~]$ sudo yum install libpcap tcpdump ethereal
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main/latest                                          | 2.1 kB     00:00
amzn-updates/latest                                       | 2.3 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package libpcap.x86_64 14:1.4.0-1.20130826git2dbcaa1.10.amzn1 will be insta
lled
---> Package tcpdump.x86_64 14:4.0.0-3.20090921gitdf3cb4.2.10.amzn1 will be inst
alled
---> Package wireshark.x86_64 0:1.8.10-25.22.amzn1 will be installed
--> Processing Dependency: libgnutls.so.26(GNUTLS_1_4)(64bit) for package: wires
hark-1.8.10-25.22.amzn1.x86_64
--> Processing Dependency: libgnutls.so.26()(64bit) for package: wireshark-1.8.1
0-25.22.amzn1.x86_64
--> Processing Dependency: libsmi.so.2()(64bit) for package: wireshark-1.8.10-25
.22.amzn1.x86_64
--> Running transaction check
---> Package gnutls.x86_64 0:2.8.5-19.15.amzn1 will be installed
---> Package libsmi.x86_64 0:0.4.8-4.6.amzn1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package     Arch     Version                                 Repository  Size
================================================================================
Installing:
 libpcap     x86_64   14:1.4.0-1.20130826git2dbcaa1.10.amzn1   amzn-main  144 k
 tcpdump     x86_64   14:4.0.0-3.20090921gitdf3cb4.2.10.amzn1  amzn-main  372 k
 wireshark   x86_64   1.8.10-25.22.amzn1                       amzn-main   15 M
Installing for dependencies:
 gnutls      x86_64   2.8.5-19.15.amzn1                        amzn-main  400 k
 libsmi      x86_64   0.4.8-4.6.amzn1                          amzn-main  2.8 M

Transaction Summary
================================================================================
Install  3 Packages (+2 Dependent packages)

Total download size: 18 M
Installed size: 81 M
Is this ok [y/d/N]: y
Downloading packages:
(1/5): gnutls-2.8.5-19.15.amzn1.x86_64.rpm               | 400 kB     00:00
(2/5): libpcap-1.4.0-1.20130826git2dbcaa1.10.amzn1.x86_6 | 144 kB     00:00
(3/5): libsmi-0.4.8-4.6.amzn1.x86_64.rpm                 | 2.8 MB     00:00
(4/5): tcpdump-4.0.0-3.20090921gitdf3cb4.2.10.amzn1.x86_ | 372 kB     00:00
(5/5): wireshark-1.8.10-25.22.amzn1.x86_64.rpm           |  15 MB     00:00
--------------------------------------------------------------------------------
Total                                           50 MB/s |  18 MB  00:00
Running transaction check
```

- Running tcpdump command first time and creating report in dumpfile.txt file



```
[ec2-user@ip-172-31-14-107 ~]$ clear
[ec2-user@ip-172-31-14-107 ~]$ tcpdump >> dumpfile.txt &
[1] 13484
```

- Running tcpdump command second time and creating report in dumpfile2.txt file



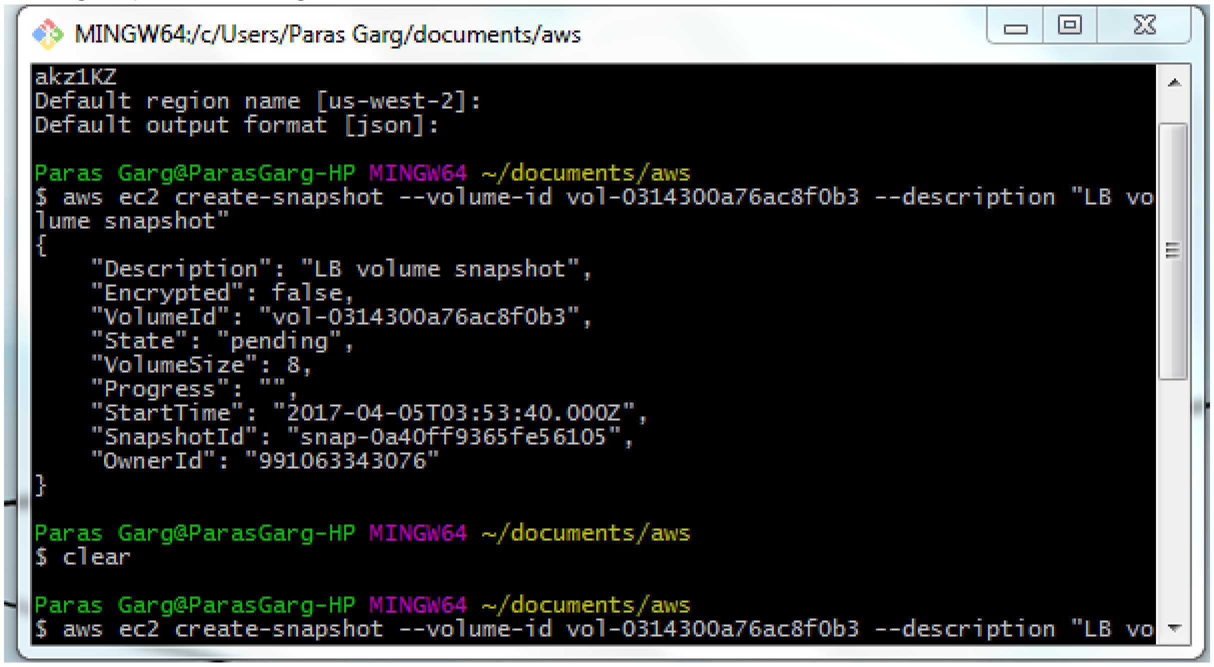
```
[ec2-user@ip-172-31-14-107 ~]$ tcpdump >> dumpfile2.txt &
[1] 13489
```

- Running tcpdump at the command prompt on the terminal would be of no help. Since tcpdump analyzes tcp/ip packets to and from the host, running the command on a remote terminal would go on indefinitely and would be full of the packet information mostly pertaining to packets exchanged while running the command itself. Hence the command is run with its output redirected to a remote file.

  When we analyze the file contents, we see the first few lines being sent by the remote host to my desktop. Then the remote host issues an ARP request to get its own mac address. Since I had made a http request to the load balancer while tcpdump was running (on the load balancer), there are packet information from my local desktop to the load balancer, then from the load balancer to one of the servers, and finally back all the way to my local desktop.

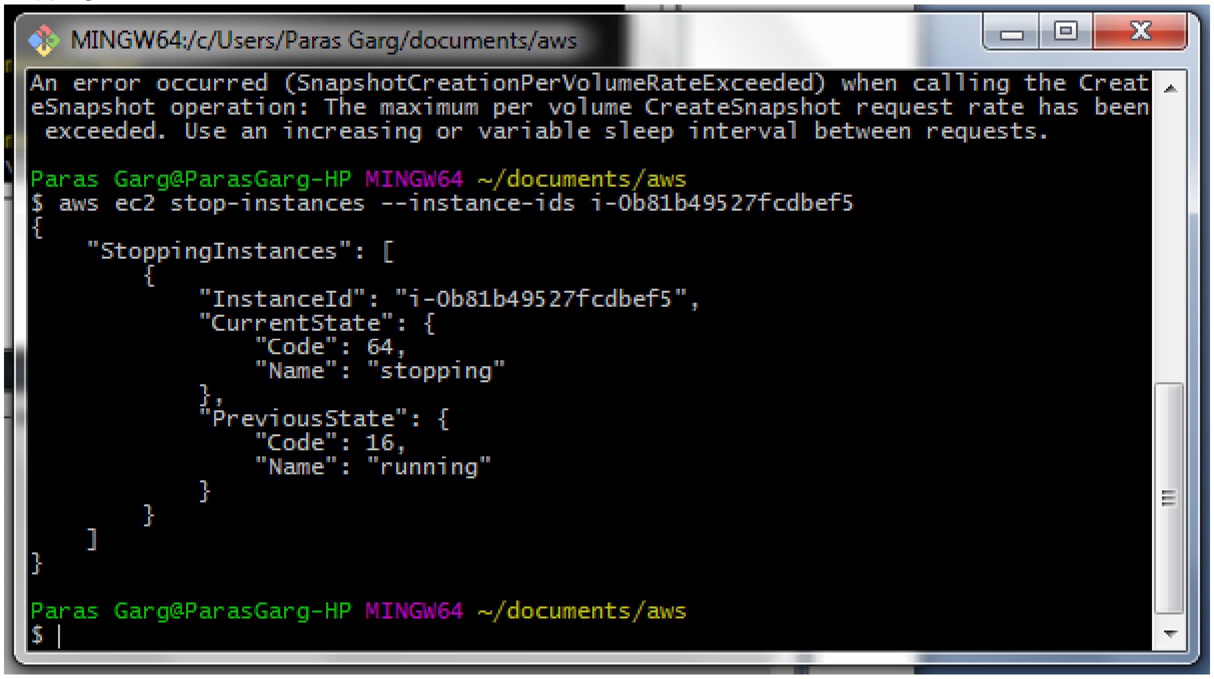
Steps to Backup and Restore on Amazon EC2 instance

- Creating snapshot of existing volume


MINGW64:/c/Users/Paras Garg/documents/aws

```
akz1KZ
Default region name [us-west-2]:
Default output format [json]:

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 create-snapshot --volume-id vol-0314300a76ac8f0b3 --description "LB vo
lume snapshot"
{
    "Description": "LB volume snapshot",
    "Encrypted": false,
    "VolumeId": "vol-0314300a76ac8f0b3",
    "State": "pending",
    "VolumeSize": 8,
    "Progress": "",
    "StartTime": "2017-04-05T03:53:40.000Z",
    "SnapshotId": "snap-0a40ff9365fe56105",
    "OwnerId": "991063343076"
}

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ clear

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 create-snapshot --volume-id vol-0314300a76ac8f0b3 --description "LB vo
```

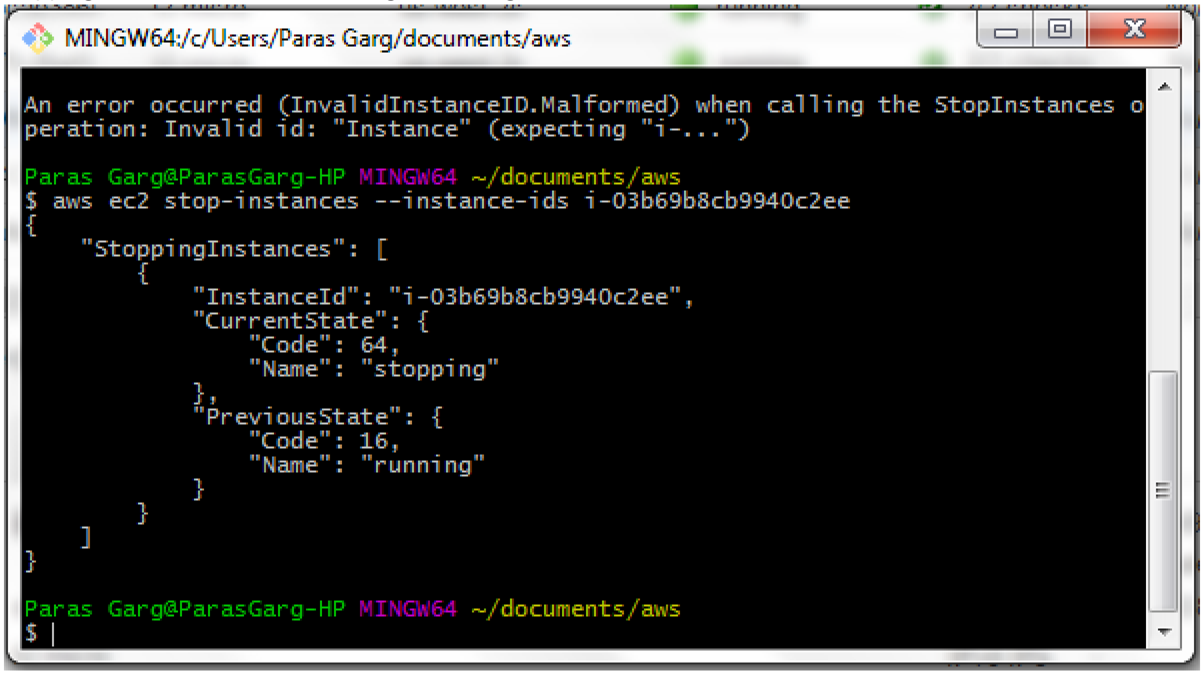
- Stopping the instance to detach the volume



MINGW64:/c/Users/Paras Garg/documents/aws

```
An error occurred (SnapshotCreationPerVolumeRateExceeded) when calling the Creat
eSnapshot operation: The maximum per volume CreateSnapshot request rate has been
 exceeded. Use an increasing or variable sleep interval between requests.

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 stop-instances --instance-ids i-0b81b49527fcdbef5
{
    "StoppingInstances": [
        {
            "InstanceId": "i-0b81b49527fcdbef5",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ |
```

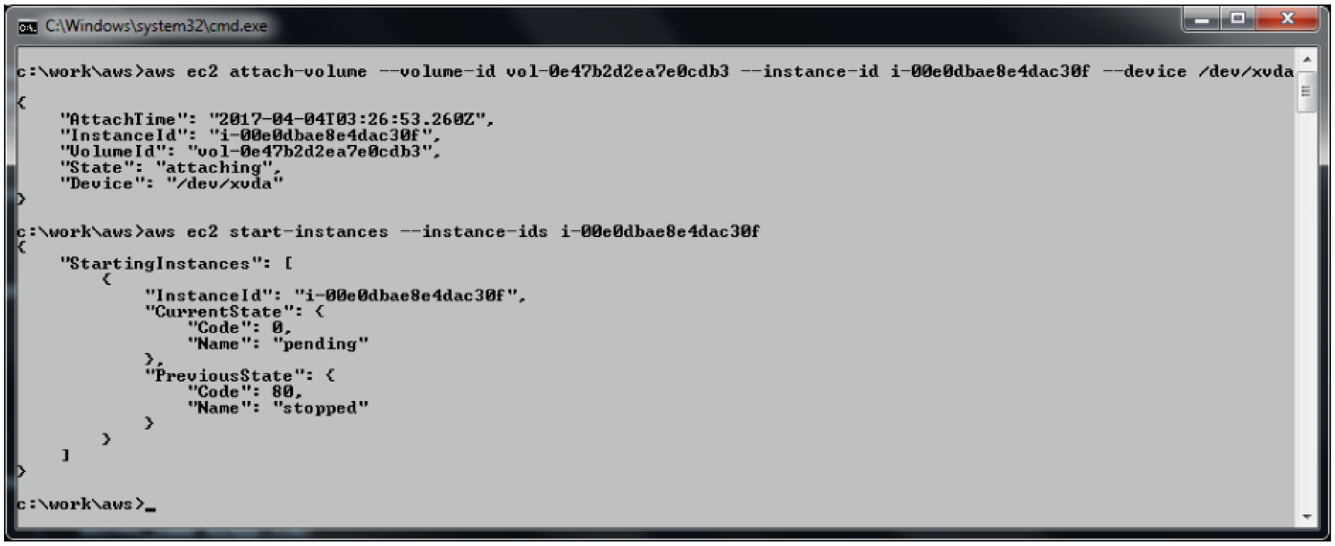
- Stopping new instance and detaching its existing volume



MINGW64:/c/Users/Paras Garg/documents/aws

```
An error occurred (InvalidInstanceID.Malformed) when calling the StopInstances o
peration: Invalid id: "Instance" (expecting "i-...")

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ aws ec2 stop-instances --instance-ids i-03b69b8cb9940c2ee
{
    "StoppingInstances": [
        {
            "InstanceId": "i-03b69b8cb9940c2ee",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

Paras Garg@ParasGarg-HP MINGW64 ~/documents/aws
$ |
```

- Attaching the previously detached volume to the new instance and starting the instance



```
c:\work\aws>aws ec2 attach-volume --volume-id vol-0e47b2d2ea7e0cdb3 --instance-id i-00e0dbae8e4dac30f --device /dev/xvda
{
    "AttachTime": "2017-04-04T03:26:53.260Z",
    "InstanceId": "i-00e0dbae8e4dac30f",
    "VolumeId": "vol-0e47b2d2ea7e0cdb3",
    "State": "attaching",
    "Device": "/dev/xvda"
}

c:\work\aws>aws ec2 start-instances --instance-ids i-00e0dbae8e4dac30f
{
    "StartingInstances": [
        {
            "InstanceId": "i-00e0dbae8e4dac30f",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}

c:\work\aws>_
```

- A volume which is the primary storage for an instance can’t be detached from the instance while it is running. We can compare this to C: drive of a Windows Computer, where the hard disk can’t be just taken out without shutting down the computer.