|  |  |
| --- | --- |
|  | **Course: Higher Diploma in Computer Science**  **Project Name: DevOps Assignment Two**  **Student Name: Bernadette Murphy**    **Student No.: W20077669**  **Date: 28/04/2019**  **Module: DevOps 2019** |

# Table of Contents

**Title Page Number**

Introduction 2

Diagram of Architecture as requested in Spec 2

Brief Explanation of Diagram

Explanation of how to create an elastic IP address 2

Screenshots of elastic IP address creation 2-3

Explanation of how to create a VPC 3-4

Screenshots of VPC creation 3-4

Explanation of how to create the security groups necessary for this project 5

Web security group 5

Screenshots of web-sg creation 5

Database security group 6

Screenshots of Db-sg creation 6

AWS -instance with nginx 7

Web-application instance 8

Load Balancing 11

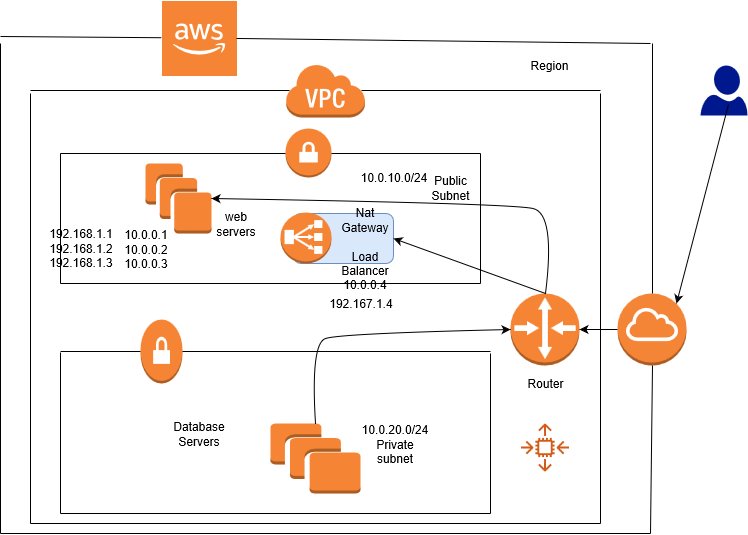
Auto Scaling 14

How to monitor CPU Utilization 15

# Introduction

This assignment will discuss the process of deployment and automated management of a load-balanced auto-scaling web application. There will be an explanation of launching an instance with Amazon Web Services (AWS). We will also explore how to enable nginx on this instance and such topics as elastic IPs, virtual private clouds, security groups, load balancing and changing the details of Instances and auto scaling. Each topic will be defined and the steps of how to carry out this task will be described. the deployment and automated management of a load-balanced auto-scaling web application. See the diagram below for the architecture

Diagram -Fig. 1



This diagram represents the architecture explained in detail below. A brief explanation for now .We created VPC with public and private subnets .And a loadbalancer that allows multiple instances .The instances should have security groups . First a web-security group for the webservers. This means that the inbound rules of the web security group are HTTP and SSH. The Databases instances have a database security group .This means that the inbound rules have MYSQLAuro. Outbound rules on both allow all traffic. Also each instance should have nginx enabled which is done by sudo commands see below in the paragraph about nginx instances. Now lets go in to this in more detail below in the following paragraphs.

**Elastic IP Address ID**

Elastic IP addresses are used by AWS to achieve its active cloud computing services. Within the AWS substructure, clienteles have virtual private clouds (VPCs). Within the VPCs, users have instances. To create the Elastic IP Address ID, go to ec2 within the AWS account Click elastic IP. Then allocate a new address. Select allocate and save the ID

Fig 2

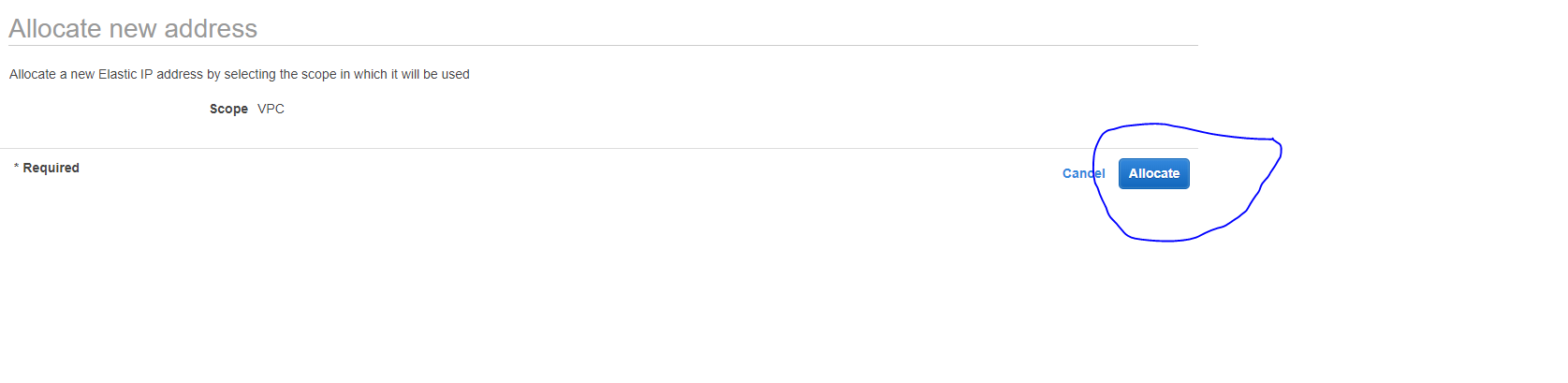
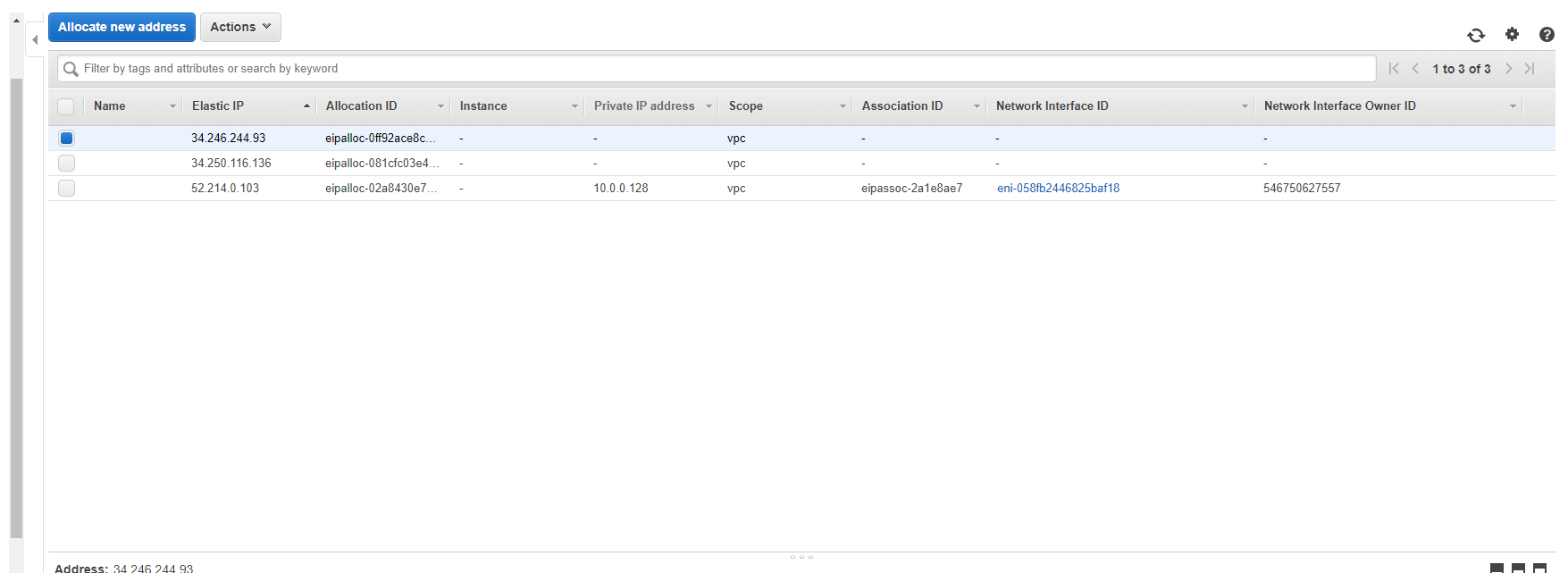
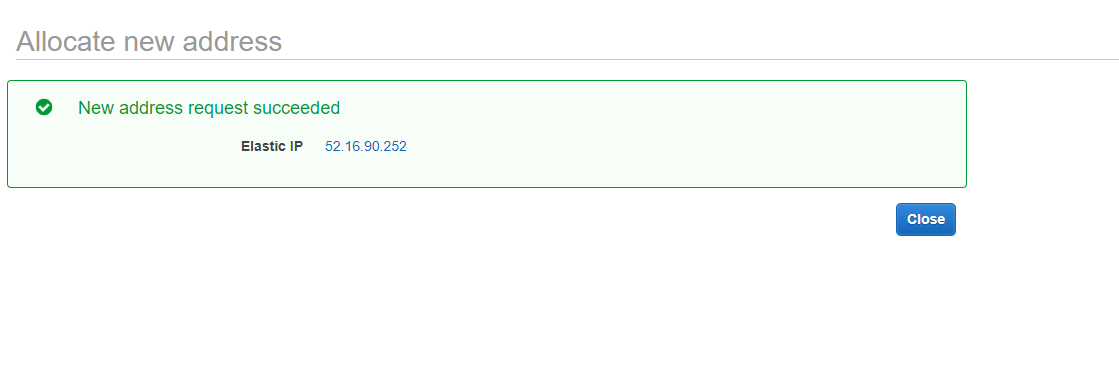


Fig 3





**Virtual Private Cloud (VPC)-** Creation of a VPC with public and private subnets into which your application will be deployed. Creation of suitable security groups.

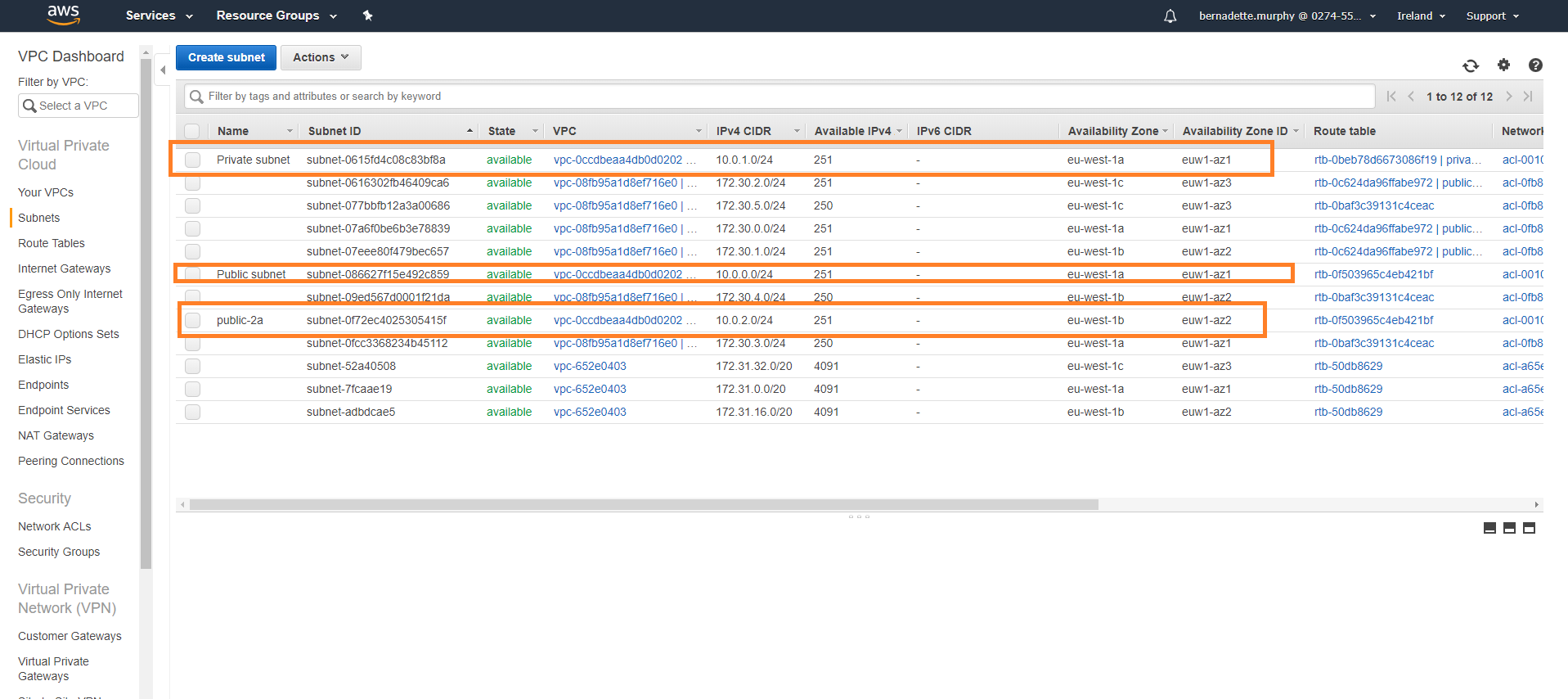
A VPC is a virtual private cloud which means that yourself and only people you/ the administrator have granted access can avail of storage on the internet and other criteria that the administrator has made available l i.e. public subnets/private subnets.

Below are the steps that are necessary to create a VPC using amazon webserver.

Go to VPC dashboard then start VPC wizard. Then select VPC public / private subnets

See Below:

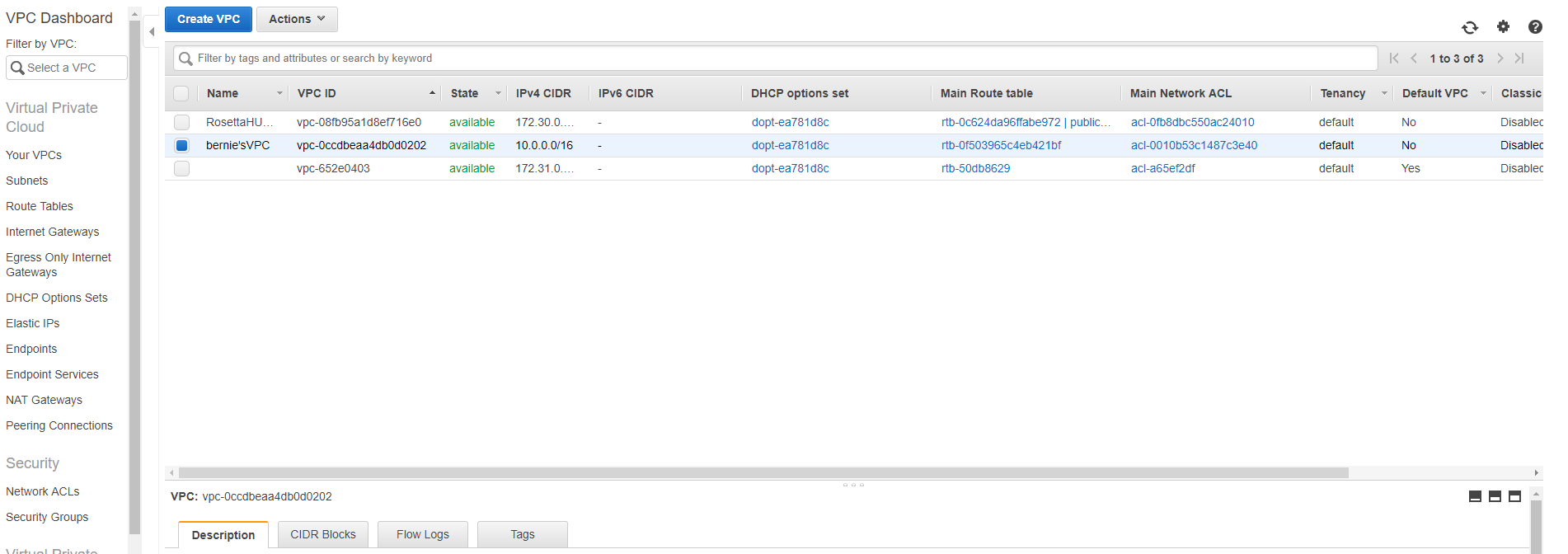
Fig 4



Name your VPC. Choose available zones (a). Choose the Elastic IP address ID created by the steps described previously and then select create VPC. Highlight new public subnet

*Click Subnet actions.* Choose Modify auto-assign IP settings then enable and save.

Fig 5



**Security Groups**

Below are the steps that are necessary to create two different security groups using amazon web server. Firstly, a web security group and second a database security group.

**WEB Security Group**

Go to EC2 dashboard andselect security groups**.** Then selectcreate security group Name your security group name for example (web-sg). You can give a description as well. The next step necessary is to add rules. To do this one must go to Inbound and then add rule. Choose HTTP 80 - 0.0.0.0/0 as this is the port that a web browser can be viewed on and SSH 22 - 0.0.0.0/0. Then outbound - add rule

Allow All traffic 0.0.0.0/0 so that all traffic is allowed. Select create. Assign a name tag to the security group so you can identify it later.

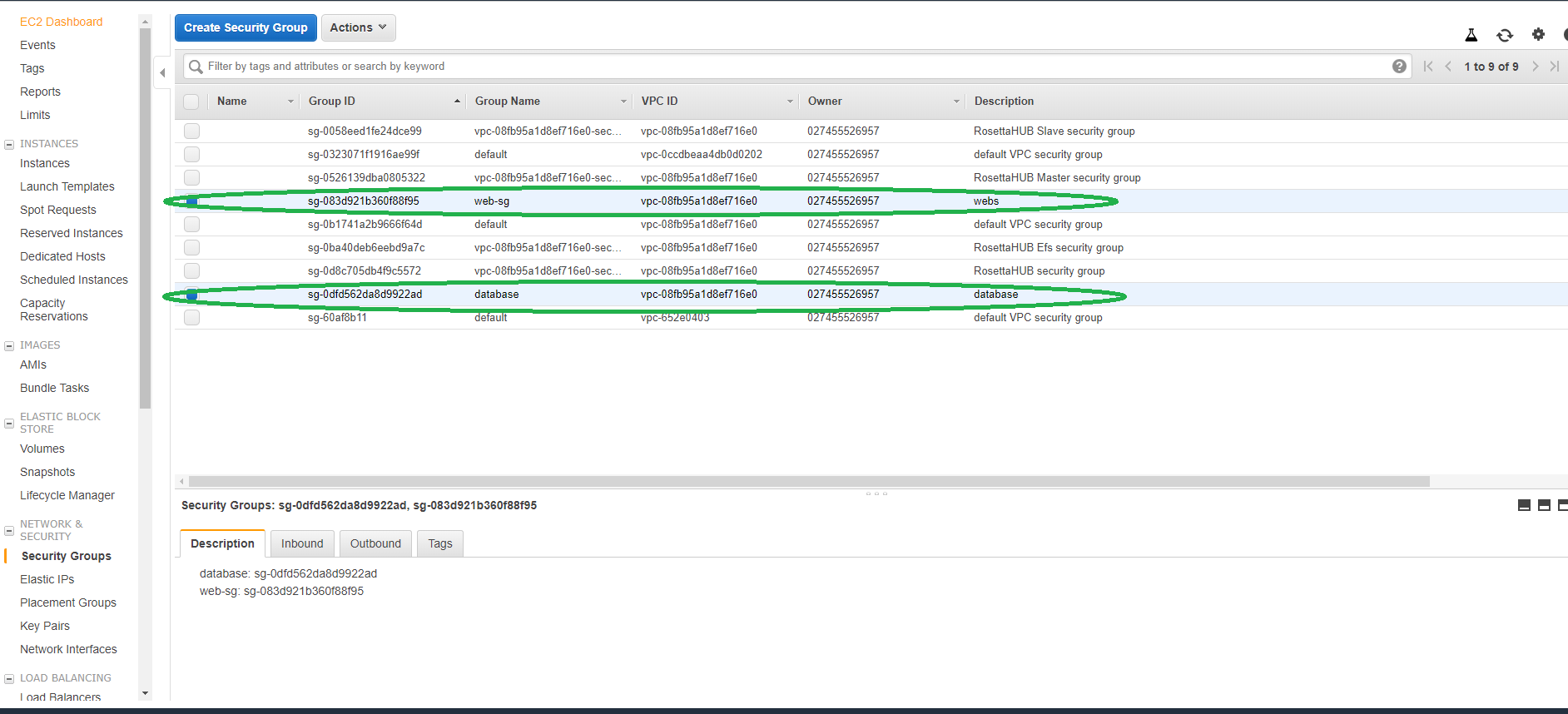


Fig 6

***A Database Security Group***

Select create security group. Name your security group for example (db.-sg). you have an option to give it a description. Choose your own VPC created by the steps described above. Then we need to add rules, so this allows for a database. The inbound Rule should be MYSQL/Aurora 3306 and the source is the web security group. Outbound rules are All traffic 0.0.0.0/. Select create. Assign a name tag to the security group so it can be easily identified later.

**AWS instance with NGINX**

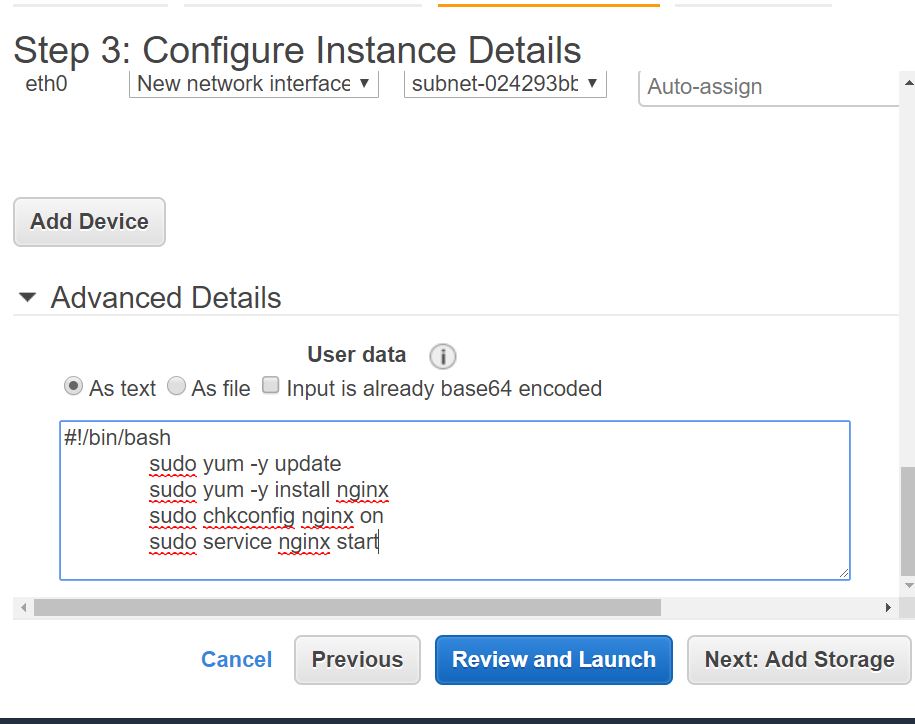
An instance is a computer-generated server in the AWS Cloud .

Note to self: NGINX Test8 is the instance I should use as it is set up already

For this assignment nginx should enabled on the instance.

To complete this task, one must login into AWS and go to the services and click on EC2 and then click on Launch Instance. Then click free tier only and then select amazon Linux AMI, select t2. micro. Then one must choose the VPC which you would have created with specific criteria outlined above in the section covering the VPC. It is necessary for the instance to have a public subnet and the Internet Protocol (IP) should be enabled. Then in advance details the following python code will be inserted this installs nginx on the instance and starts it up. There is no need to indent the code the code has been edited after this screenshot was taken.

Fig 7.



There is no change necessary to the storage allowed on the instance for this task. Then the web security group that was created by following the steps outlined in the section covering security groups should be selected. The click on review and launch and choose an existing key pair, launch the instance and then click view instances. It is necessary to wait for the instance to show a status of running and 2/2 to proceed to the next step this may take some time.

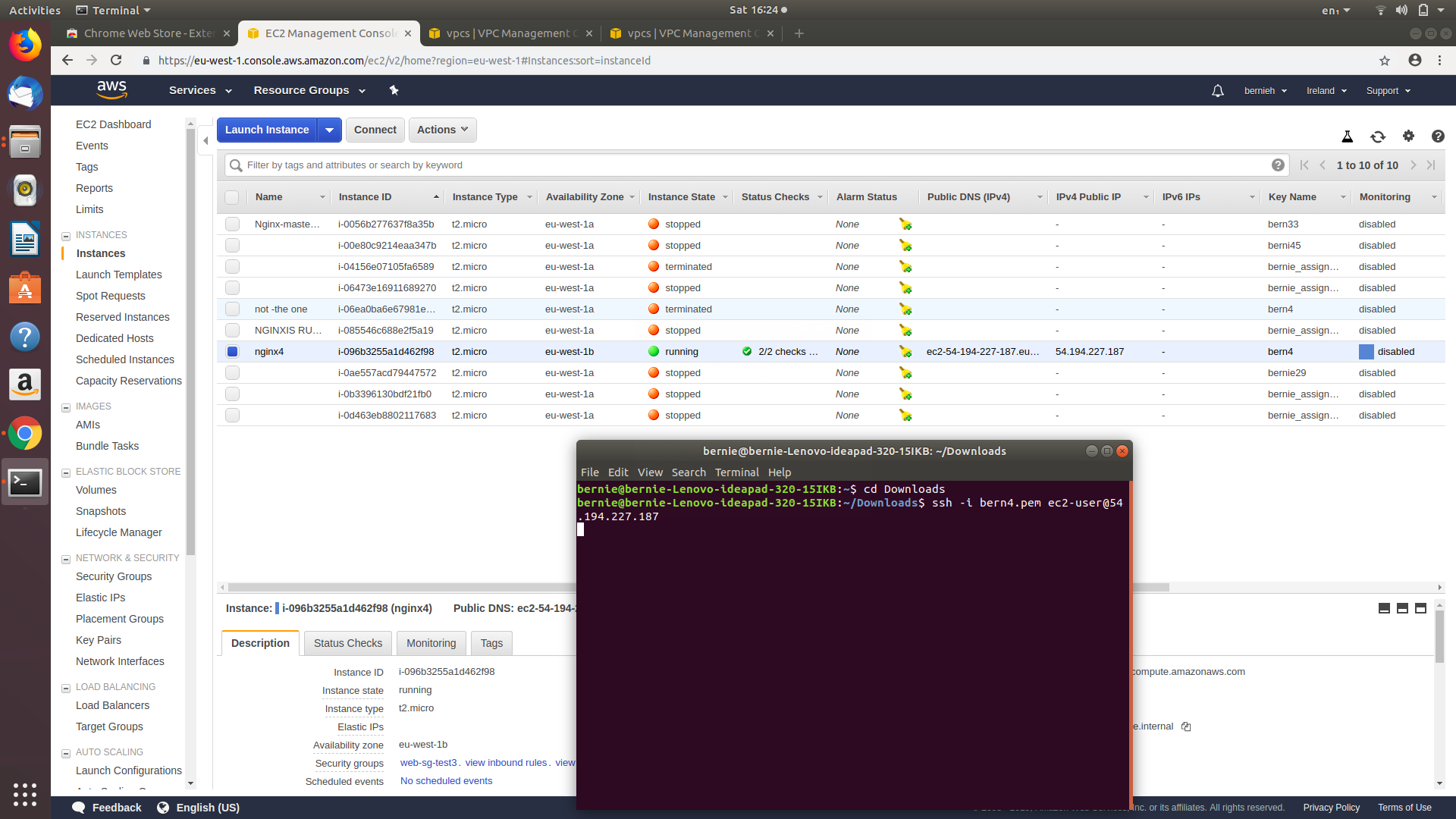
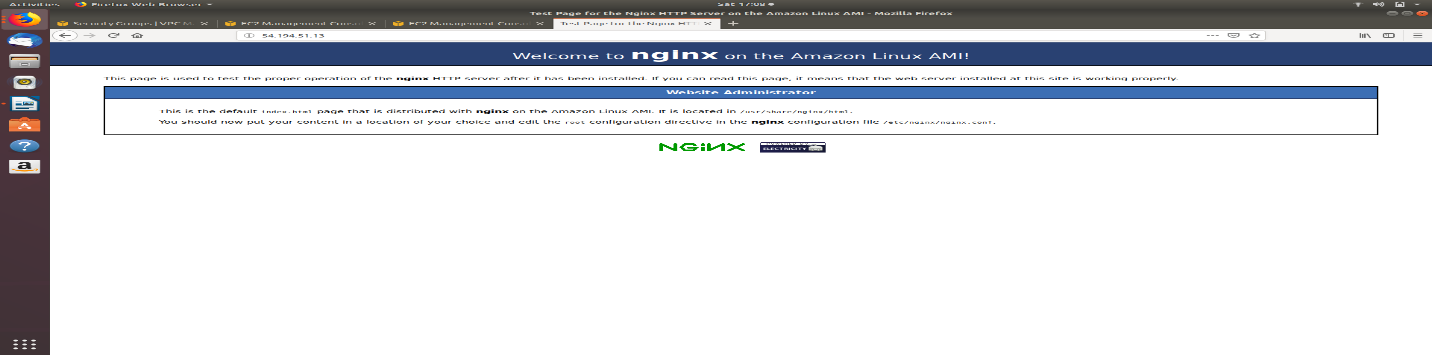
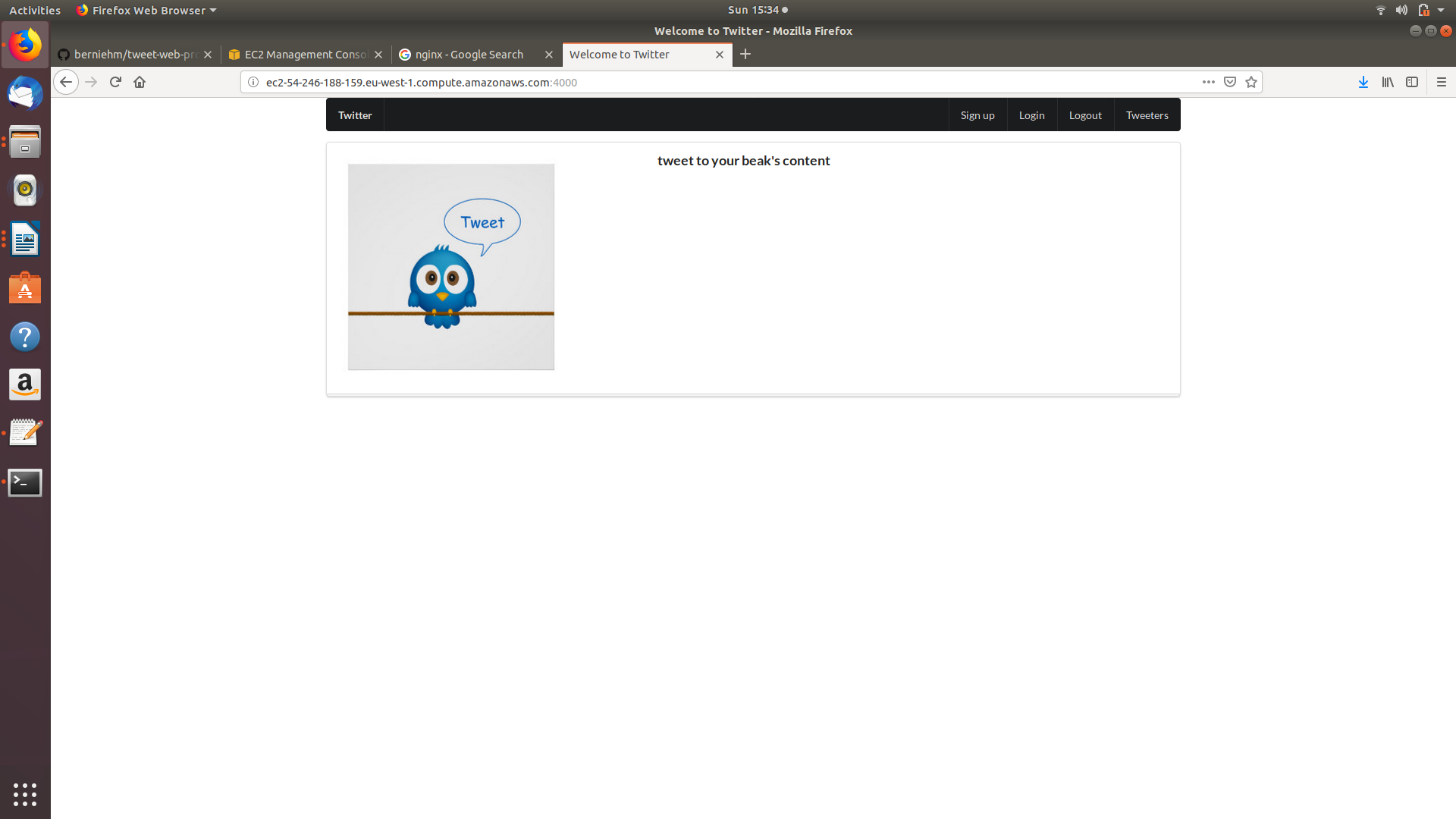


Fig 8.

Then we are going to copy and paste the IP address in to the web browser and hit enter. We should the n see a web page as seen below

Fig 9.



**Fig 10**

**Launching a web application-*Creation and configuration of a “master” instance of a web application. You may choose any web application, ideally one that relies on a third party or back-end service. Note: any back-end services/databases chosen should require minimal resources, e.g. use of micro instances and small amounts of storage.***

This is similar to launching an nginx instance however the security group has to have the following rules ,

ssh -- 22 – anywhere

HTTP -- 80 –anywhere

https -- 443 –anywhere

custom tcp -- 4000 -- anywhere

Than we are going to ssh into the instance just launched with the ssh command and then use the curl command

* **Curl command curl** -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.6/install.sh | bash

. ~/.nvm/nvm.sh

* + - nvm install 6.11.5
    - node -e "console.log('Running Node.js ' + process.version)"
    - sudo yum install git

Type in the following in to the terminal line by line

* check node -v
* check npm -v
* check git --version
* sudo yum -y update
* sudo yum -y upgrade
* git clone https://github.com/bernieh/twitter-web.git
* cd twitter-web/
* . ls
* cat index.js
* nano index.js (do this if your custom tcp 4000 does not match what's inside index.js)
* npm install
* node index.js
* copy from AWS ec2-54-246-188-159.us-west-1.compute.amazonaws.com
* copy into browser and add 4000
* ec2-54-246-188-159.us-west-1.compute.amazonaws.com:4000

Once all these steps are completed the app is now live on that instance and can be utilized as seen above in the screenshot

Creation of a custom AMI based on your master instance, to be used by EC2 auto-scaling.

**----**

**Load Balancing**

**Tasks**

-Creation of an elastic load balancer

-Generation of test traffic to the load balancer – e.g. using curl/wget or a web testing tool.

- Show that the load is distributed across more than one web server – e.g. by viewing web server or other logs. Screenshots and a brief explanation in your report will suffice for this.

Go to the load balancer’s section and select create load balancer. Choose application Load Balancer and then select create. Name the load balancer a name of your choice.

Select the VPC created earlier so we can have public and private subnets. Select your public subnet.

Add a tag of your choice. Then click next configure the security settings.

Click next configure security groups. Select existing security group. Choose the web security group that you have created so we can view our instances later with this load balancer.

Next, we will configure routing. Choose a name for the target group.

The protocol should be HTTP, so we can use a web browser and view a web page on the load balancer. The port that allows traffic on the web is port 80. Then target type – instance, Health checks protocol 80

And Path /

Next, we will register targets. Select instances you should have two instances running with nginx from the steps described above select these instances. Click add to registered. Next click on review

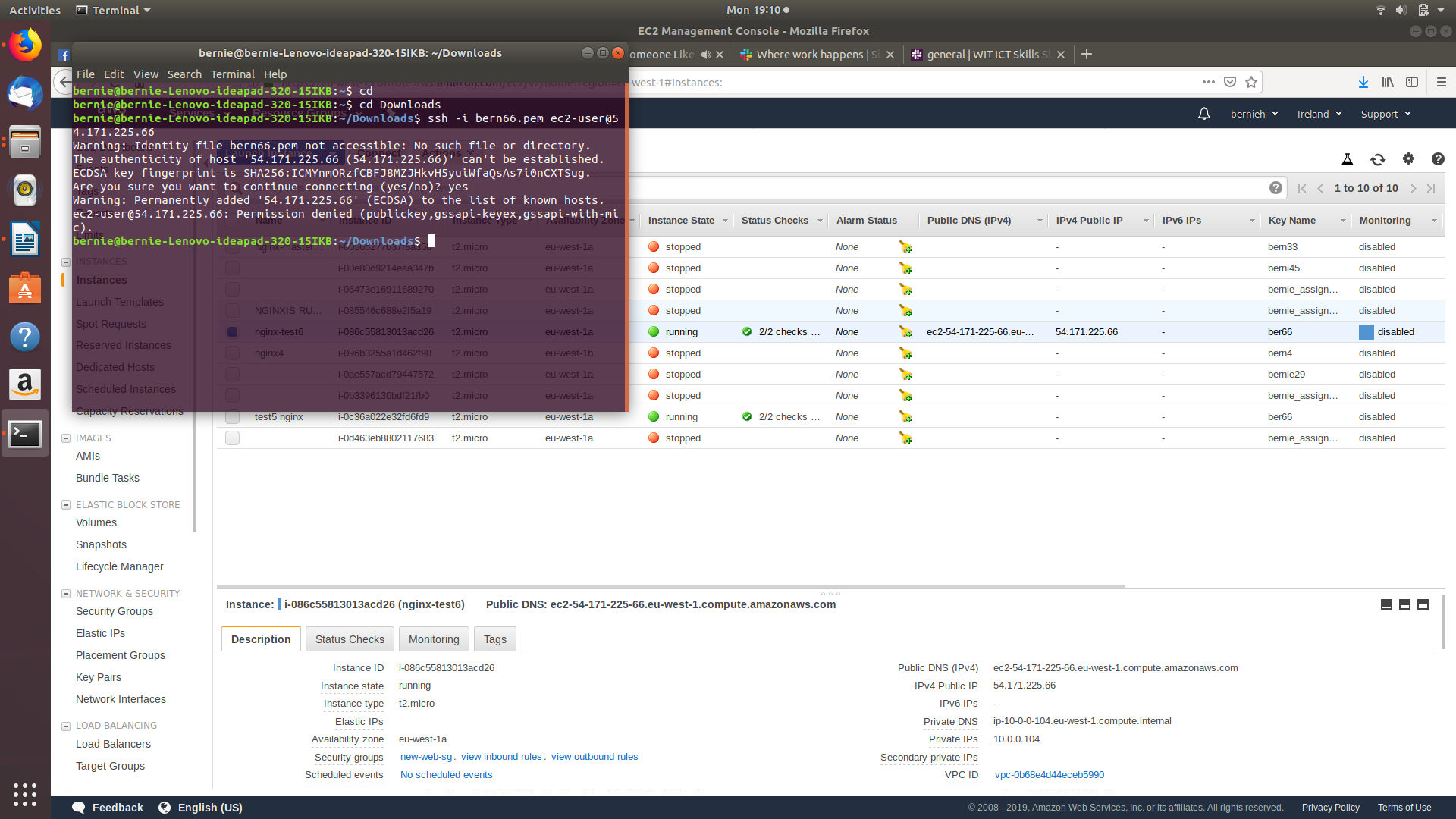
Click create and close. Pause till the state is active. Then go to next step below these steps allow will allow you to view your instances through the web browser and edit the web pages.

**Changing details in an instance**

Now We will change the details on one of the instances so we can recognize the difference between the two instance son the load balancer when generating traffic.

Go to your running instance. Go down to descriptions and highlight the IPv4 address. Right click on go to address. In the Web Admin box take note of /usr/share/nginx/html as We need this to change details on this page. Open your terminal and go to where pem key is within your directory. Input ssh -i <pem key> ec2-user@<ip address of instance>. Hit enter and then Type in 'yes'.

Fig 11.



Type in sudo nano /usr/share/nginx/html/index.html

You will see an editor page of your instance. Make a change so it will be noticed from original like changing the color of the page. When the changes are made press ctrl & x. To save changes press Y. To write changes press enter. Return to AWS site On EC2 page and go to where your Load Balancer is

Highlight DNS address (no A Record). Select go to address. Click refresh on webpage to go through instances. You will see the changes you have made from previous step see below.

Fig 12.



**Launch Configurations & Auto Scaling**

**Tasks**

* Creation of a launch configuration based on that custom AMI
* Creation of an auto-scaling group based on your load balancer and launch configuration; making changes to this auto-scaling group. 7. Creation of an auto-scaling policy; making changes to this policy. You need only scale the front-end server instances. 8. Using CloudWatch to trigger an increase in resources based on an alarm – based on a built-in metric such as CPU utilization network traffic, etc. support metric choice

How to

Go to EC2 Circumnavigate to auto scaling then Select launch configurations

We are going to create auto scaling group. We will then click create launch configuration

Then Select my AMIs and next configure details. Name the auto scaling group. Monitoring - enable

Click Advanced details and IP Address Type - Only assign Next - add storage (no change)

Next - Configure security group. Select existing security group

Choose web server security group. Review Create launch configuration. Select key pair. Create launch configuration. Name the group. Then we will start with 2 instances and select the VPC as described in the VPC section above.

Apply public subnets, then in advanced details tick box in load balancing, select the target group created.

Select the load balancer created in the project .Enable monitoring as we should monitor for this project.

Next we are going to configure scaling policies. Use scaling policies to adjust the capacity of this group Scale between 2 to 5 instances for this project.Scale the Auto Scaling group using step or simple scaling policies

Increase the group SizeAdd alarm

Uncheck send notification. Set >= to 70%. For at least 1 consecutive period(s) of 1 minute

Create alarm

Take the action - add 1 instance

Add alarm

Uncheck send notification

Set <= to 50%

For at least 1 consecutive period(s) of 1 minute

Create alarm

Take the action - remove 1 instance

**How to monitor CPU utilization**

**Tasks**

Use of your own script to monitor some activity on your server. For example, this could be web server or other server logs, or OS activity (CPU, memory, disk, number of processes, etc)

https://www.youtube.com/watch?v=NQx43bY4lNo

I watched this video and sourced the internet and devised my own script to monitor some activity on the webserver such as the cpu utilization and the memory disk and the number of processes. The process can be seen in script attached and screen shots.

Analysis of the architecture, using the following categories of the AWS Well Architected Framework (<https://aws.amazon.com/architecture/well-architected/>)

The five pillars of a well architurected frame work are security , reliability performance efficiency , performance efficiency , cost optimization and operational excellence.

▪ Security

This is done by the creation of security grouchiness secure the data of the architecture we have set up. We should have security at all layers. This is done by security groups for each instance. We could also consider encryption and isolation of infrastructure .

VPC route table give us the ability to control how the traffic is routed with VPC between the subnet and to and from the internet.

▪ Reliability

This means we can recover from architecture failures . We have the ability apply computing resources that meet demand for example the clients demand. And also fix disruptions such as misconfigurations. This can be done by testing recovery procedures. As well as autoscaling to scale to demand. We should also not estimate the capacity and know exactly what we need.

▪ Performance Efficiency

We need to adapt to meet technology demands as they change. We used auto scaling to increase or decrease instances as necessary.

We could have used databases but I only set up the security group for a database instance I did not attempt to launch the instance.

▪ Cost Optimization

Obviously the lower the cost the happier the customer. However we need value for money too. This can be done by using a student account if the user is a student. Turn off resources went they are not needed such as stopping instances and deleting ant gateways. We can also monitor using cloud watch alarms.

▪ Operational Excellence

This is centered upon monitoring services this is done in this project by creating a bash script. This bash script monitors CPU usage and memory disk and the number of processes. Also monitoring is done by using commands such as cpu100.

References

[*https://aws.amazon.com/architecture/well-architected/acessed*](https://aws.amazon.com/architecture/well-architected/acessed)*date accessed 26/04/19*

[*https://www.youtube.com/watch?v=NQx43bY4lNo*](https://www.youtube.com/watch?v=NQx43bY4lNo)*date accessed 26/04/19*

[*https://wit-hdip-comp-sci-2018.github.io/devops/*](https://wit-hdip-comp-sci-2018.github.io/devops/) *dates accessed 08/03/19, 19/04/19*