Lab 8: Load Balancing / Autoscaling

East Tennessee State University

CSCI 4417/5417: Introduction to System Administration

Spring 2016

Pramod Nepal

Pramod Nepal
19 April 2016

Purpose

The focus of this lab was to learn to setup a Load Balancer and Autoscaler. The machine image used by the Balancer and the Autoscaler simulated a web server and the request load was simulated using a tool called Siege.

Materials

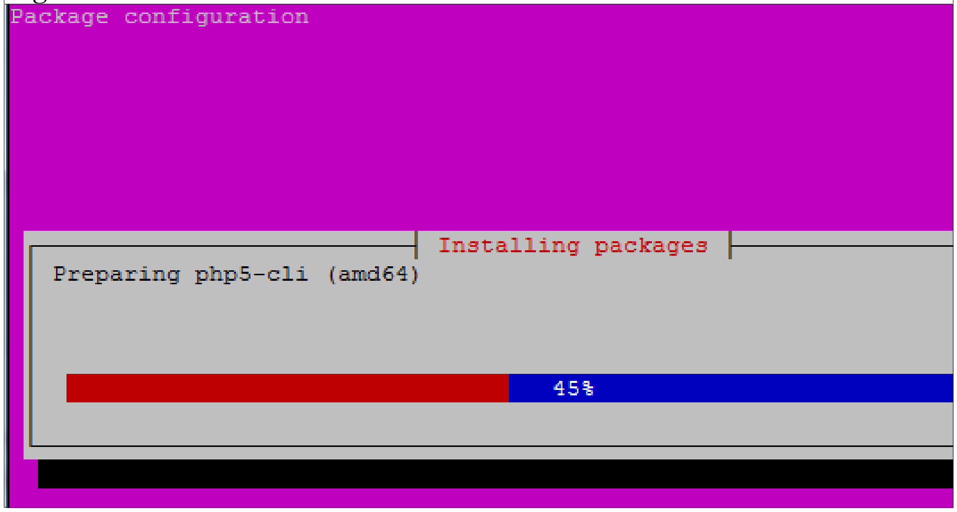
- Lab Instructions
- PuTTY
- Web browser

Procedure and Results

Setup

A new instance of Ubuntu was created using previously configured security group. No private IP was assigned as used in earlier labs. There were no instances that would want to privately connect to instances in the subnet. This image would be used by the Load Balancer and the Autoscaler. As per the lab instructions each instance would host a web server. Using 'tasksel', 'lamp-server' package was installed (see Fig. 1).

Figure 1: LAMP Server Installation



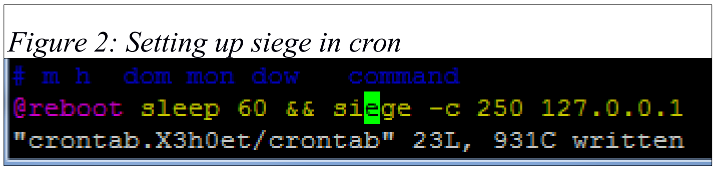
```
Package configuration

Installing packages
Preparing php5-cli (amd64)

45%
```

Then the server load simulator tool Siege was also installed. To automate the simulation of several requests 'siege' command would start after the instance's boot. This was specified in Linux Task Scheduler tool 'cron'. 250 simultaneous user requests were simulated using Siege (see Fig 2.).

Figure 2: Setting up siege in cron

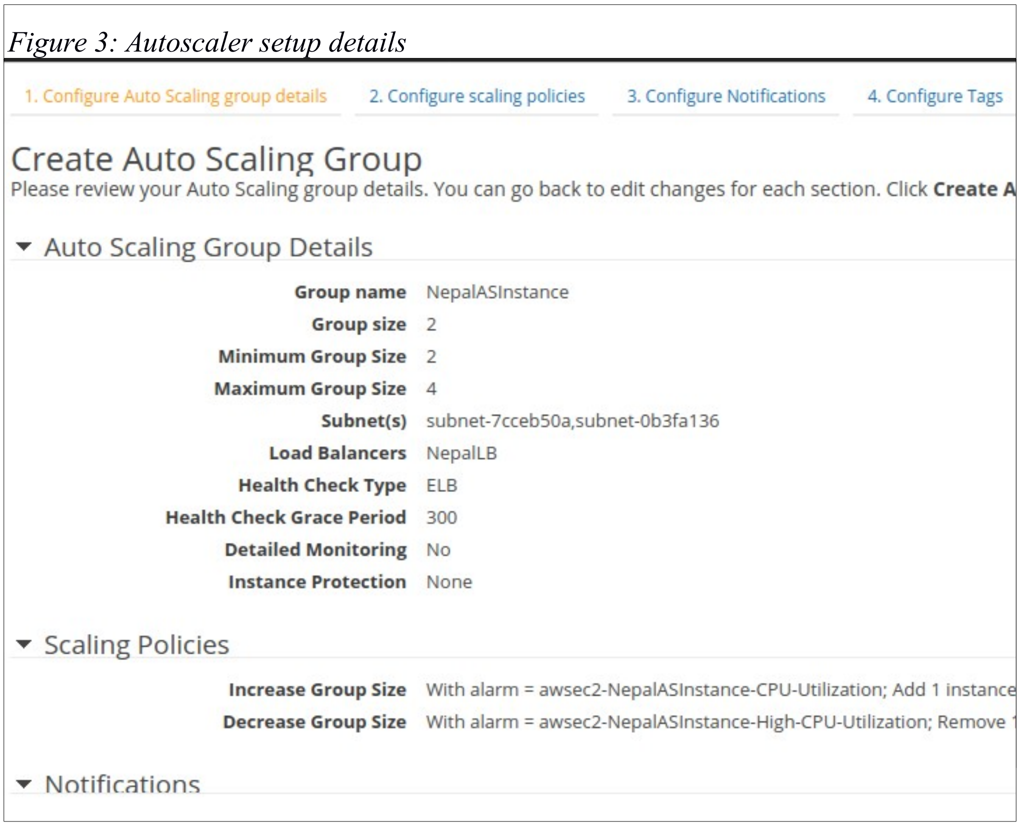


```
# m h  dom mon dow   command
@reboot sleep 60 && siege -c 250 127.0.0.1
"crontab.X3h0et/crontab" 23L, 931C written
```

Once the Ubuntu instance was configured to host a web server and simulate the requests a new public subnet was created. This subnet would use a different Availability Zone than used in the default public subnet that was used throughout earlier labs. The CIDR was set to 173.1.3.0/24 and it was configured to auto-assign public IP. This subnet also used the VPC's route table. Next a Load Balancer was created. This Load Balancer was configured to use both public subnets available in the server. The one discussed above and the one created in earlier labs. This would configure the it to balance the load in servers that exist in different availability zones. This would be the ideal scenario in a production system. 'Security Group' used in previous labs was selected.

After the Load Balancer was setup the next step was to configure an Auto Scaling group. A new 'Auto Scaling' group was created from 'Auto Scaling' tab in EC2 dashboard. To select the previously created Ubuntu instance 'My AMIs' was clicked and the image was selected. 'Assign a public IP address to every instance' option was selected so that each created instances would get their own public IP. The size of the group was set as 2. All the public subnets were added on the Subnet field. This would scale the image using both subnets used in this lab. An option called 'Receive traffic from Elastic Load Balancer(s)' was selected so that the Load Balancer and Autoscaler would work in coordination. The Load Balancer created above was selected. 'Health Check Type' was set to 'ELB' or Elastic Load Balancer. For the scaling policy section 'Scale' was selected between 2 and 4 instances. This setting would specify minimum instances as 2 and maximum instances as 4 for the Autoscaler. To trigger this effect, in the 'Increase Group Size' the 'Percentage' was set to 20. This would trigger the scaling rule if the CPU utilization exceeded 20 %. Similarly another rule was set to decrease the scaling when the CPU utilization was less than 19%. 'Take the action' field' was set to 1. This would set the removal of one instance when scaling down was triggered. Details of the Autoscaler can be seen in following figure (Fig. 3).

Figure 3: Autoscaler setup details



1. Configure Auto Scaling group details
2. Configure scaling policies
3. Configure Notifications
4. Configure Tags

Create Auto Scaling Group

Please review your Auto Scaling group details. You can go back to edit changes for each section. Click **Create A**

▾ Auto Scaling Group Details

Group name	NepalASInstance
Group size	2
Minimum Group Size	2
Maximum Group Size	4
Subnet(s)	subnet-7cceb50a,subnet-0b3fa136
Load Balancers	NepalLB
Health Check Type	ELB
Health Check Grace Period	300
Detailed Monitoring	No
Instance Protection	None

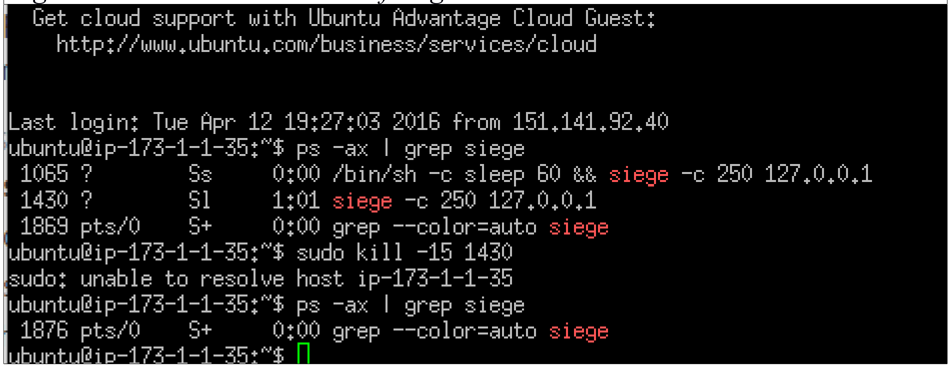
▾ Scaling Policies

Increase Group Size	With alarm = awsec2-NepalASInstance-CPU-Utilization; Add 1 instance
Decrease Group Size	With alarm = awsec2-NepalASInstance-High-CPU-Utilization; Remove 1

▾ Notifications

When the Autoscaler runs, the instances are launched. The siege application would start after 1 minute and simulate the requests. This would increase the load on the server and trigger the scaling up. Since the updates are seen after 5 minutes, the resources are added after this time has passed. To see the Load Balancer working properly the public addresses of the instances were browsed through the web browser. This would launch the Apache web site hosted on the instance. Once the instances were running and the scaling up was performed (if instances were less than 4), the siege application was manually killed to stop the requests simulation (see Fig. 4). This would trigger the scaled down and few instances would be terminated.

Figure 4: Manual termination of siege command

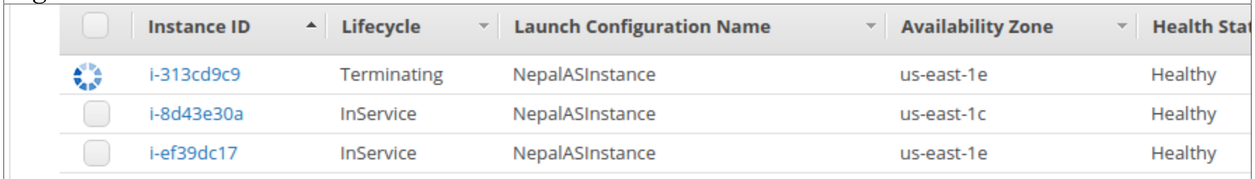


```
  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud


Last login: Tue Apr 12 19:27:03 2016 from 151.141.92.40
ubuntu@ip-173-1-1-35:~$ ps -ax | grep siege
 1065 ?        Ss     0:00 /bin/sh -c sleep 60 && siege -c 250 127.0.0.1
 1430 ?        Sl     1:01 siege -c 250 127.0.0.1
 1869 pts/0    S+     0:00 grep --color=auto siege
ubuntu@ip-173-1-1-35:~$ sudo kill -15 1430
sudo: unable to resolve host ip-173-1-1-35
ubuntu@ip-173-1-1-35:~$ ps -ax | grep siege
 1876 pts/0    S+     0:00 grep --color=auto siege
ubuntu@ip-173-1-1-35:~$ 
```

Several different use cases of scale up and scale down could be see in the EC2's Instances tab or the Load Balancer's tab (See Fig. 5).

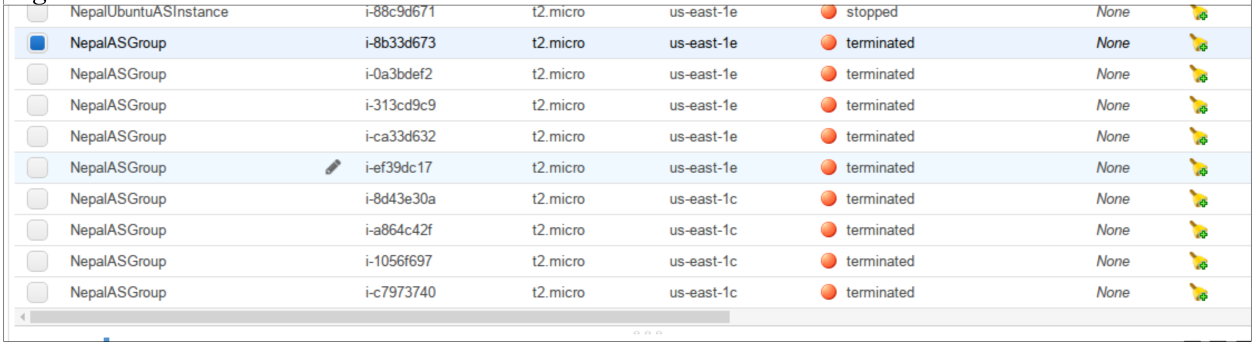
Figure 5: Instances in Load Balancer tab



□	Instance ID	Lifecycle	Launch Configuration Name	Availability Zone	Health Stat
	i-313cd9c9	Terminating	NepalASInstance	us-east-1e	Healthy
□	i-8d43e30a	InService	NepalASInstance	us-east-1c	Healthy
□	i-ef39dc17	InService	NepalASInstance	us-east-1e	Healthy

The Instances tab would show all the instances that were terminated by the Autoscaler during the Lab's execution (see Fig. 6).

Figure 6: Terminated Instances



□	NepalUbuntuASInstance	i-88c9d671	t2.micro	us-east-1e	●	stopped	None
■	NepalASGroup	i-8b33d673	t2.micro	us-east-1e	●	terminated	None
□	NepalASGroup	i-0a3bdef2	t2.micro	us-east-1e	●	terminated	None
□	NepalASGroup	i-313cd9c9	t2.micro	us-east-1e	●	terminated	None
□	NepalASGroup	i-ca33d632	t2.micro	us-east-1e	●	terminated	None
□	NepalASGroup	i-ef39dc17	t2.micro	us-east-1e	●	terminated	None
□	NepalASGroup	i-8d43e30a	t2.micro	us-east-1c	●	terminated	None
□	NepalASGroup	i-a864c42f	t2.micro	us-east-1c	●	terminated	None
□	NepalASGroup	i-1056f697	t2.micro	us-east-1c	●	terminated	None
□	NepalASGroup	i-c7973740	t2.micro	us-east-1c	●	terminated	None

At the end of the lab, to prevent Autoscaler and Load Balancer's trigger, both were deleted before any running instances were terminated.

Observations

This lab experimented with Load Balancer, Autoscaler and its effect on instances. Several use cases were performed to see how it would scale up and scale down. If the instances were terminated manually and the instance size went below minimum, the Autoscaler would scale up automatically. Similarly when the load was added using the request simulation, it would trigger creating more server instances. Similarly when request load was removed it would scaled down the instances. This use case is very important for organizations that would want to optimize the resources used for hosting their web applications. When the client requests increase the scalar would add more resources and when the requests are low it would save cost on resources. Similarly Load Balancer would evenly distribute the load so that all the instances would be able to fairly handle the requests.

For the lab itself the simulation of requests using 'siege' was a way to show what would happen when the website received many simultaneous requests. Working with Linux task scheduler cron and learning about siege was a big plus point for this Lab. Siege could be used in development environment to simulate the client requests.