Lab AmazonWeb Services – EC 2

V0 - Guillaume Urvoy-Keller
V0.1 – Fabien Hermenier

During this lab, you will deploy VMs (a.k.a instances) and install a Web service. Next, leveraging the functionnalities of EC2, you will first create a template and then set up an elastic Web service, that will adapt to the number of Web server instances to the actual load.

Part I: Creation of a first instance

- Log on AWS (**https://aws.amazon.com/**) with your credentials.
- From the EC2 console, click «*Launch Instance»*

The page «*Select an Amazon Machine Image (AMI)*» lists the available AMIs (template). Pick «Amazon Linux AMI».

- On the page «*Select an Instance Type*», keep (or enforce) the default choice, namely: *t1.micro instance*.
- Click on «*Review and Launch*» to continue
- Creatre a pair of keys with name *aws* and move it to your *~/.ssh/*.

To automate the login process, create a *login* file in your *~/.ssh/* directory and adapt the example below so that you can log to your instance by simply typing *ssh aws*

Host server1 server1.company.com

Hostname 12.34.56.78

User ubuntu

IdentityFile /media/11361B1123123634/server1.pem

Part II: Customize your instance.

- Connect to the VM via *ssh* using the previous key and the user name *ec2-user*. The public name of the VM is available in its description.
- Install the required packages:

```
$ sudo yum update -y
$ sudo yum groupinstall -y "Web Server" "PHP Support"
```

- Start the Web daemon and and enable an automatic start-up of the Web service at VM boot time:

```
$ sudo service httpd start
$ sudo chkconfig httpd on
```

Check that the server is up and running through your browser after adapting the security goup of the VM via the AWS console.

Data from your web site need to shared among several VMs. A solution consists in attaching an independent disk to all the VMs. Such a volume is called and *Elastic Block Store* and can be

configured via the EC2 console:

- Create a 10 GB disk in the same availability zone.
- Attach the disk to your instance.
- In a terminal of the VM, format this new partition, create a directory and modify */etc/fstab* to mount is at boot time:

```
$ sudo mkfs -t ext4 /dev/sdf
$ sudo mkdir /mnt/www
$ sudo nano /etc/fstab
$ cat /etc/fstab
#
LABEL=/     /           ext4    defaults,noatime  1   1
tmpfs       /dev/shm    tmpfs   defaults        0   0
devpts      /dev/pts    devpts  gid=5,mode=620  0   0
sysfs       /sys        sysfs   defaults        0   0
proc        /proc       proc    defaults        0   0
/dev/sdf /mnt/www       ext4          defaults 0 0
```

- Modify the configuration file of httpd so that */mnt/www* be your root directory for this serviceainvariable *DocumentRoot* in */etc/httpd/conf/httpd.conf)*.
- Install the following php script in */mnt/www/bench.php*:

```
<?php
function fibo($n) {
   return(($n < 2) ? 1 : fibo($n - 1) + fibo($n - 2));
}

echo $_SERVER["SERVER_NAME"] . ": " . fibo(23);
?>
```

- Restart the httpd daemon and test if the PHP script works.

Part III: Creating a template (AMI in AWS vocabulary)

- Stop your instance (do not terminate it!!!). In the EC2 console, go to «Instances», select your instance and choose «Create Image».
- Once this is done, go to the «EC2 Dashboard» and create a new instance out of your template, using the same keys. Update your ~/.ssh/login passwd.
- Check that the Web server is indeed accessible from outside by testing the *bench.php* script.

Part IV: Load Balancing

As its name indicates, the load balancer role is to distribute the load among several VMs.

- Create a load balancer from the «Network & Security» menu and associate your two instances (restart the initial one). Once associated (a few minutes), modify the *Health Check* parameters with 30 seconds in between 2 tests and /bench.php as a target.
- Find the name of the load balancer. Once it is up and running, check that the two VMs are accessible:

```
$ watch nslookup -debug load_balancer_name
```

- What is the lifetime of a response ?
- What do you observe by using the IP of the load balancer in your browser?
- You can now stop the 2 VMs.

PartV: Horizontal Scaling

The *auto-scaling* service enables to automate scaling operations.

- Click on « Auto-Scaling Group » et follow the creation process by picking your AMI.
- Continue the process to create an *auto-scaling* group. Between 1 and 5 instances can run. Declare a load balancer in the *Advanced details* section (you can also do it after by creating a load balancer and associating as instance the auto-scaling group). Use one of your subnets in your votre *availability-zone.*
- Next, you have to define the scaling rules. For instance:
  - create a new VM when the CPU load is over 60 %
  - remove a VM when the CPU load goes below 30 %

Your scaling group is now ready for testing. To test elasticity, code a script (or use a program like *ab*) which execute a variable amount of requests by calling the *bench.php* script.

Final part: Delete everything!!!

You have only 100$ of credits. Stop your Vms and all services used or you might receive an unpleasant invoice in a few months.