Lab AmazonWeb Services – EC2

Guillaume Urvoy-Keller, Fabien Hermenier, Quentin Jacquemart

During this lab, you will deploy Virtual Machines (VMs, a.k.a. instances) and install a Web service. Next, leveraging the functionalities 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.

1 Creation of a first instance

1. Log on AWS (`https://aws.amazon.com/`) with your credentials.
2. From the EC2 console, click "Launch Instance".
   The page "Select an Amazon Machine Image (AMI)" lists the available AMIs (template). Pick the free-tier available "Ubuntu Server".
3. On the page "Select an Instance Type", keep (or enforce) the default choice, namely: **t2.micro** instance.
4. Click on "Review and Launch" to continue
5. Do not modify the security settings
6. Create a pair of keys with the name "aws", and download the private key to your `~/.ssh/` folder.

Please note that the default username for your machine is is `ubuntu`. Connect to the machine with `ssh -i  /.ssh/aws.pem ubuntu@IP_of_machine`.

A possible alternative on Windows machines is to use the `ssh` client PuTTY (`http://www.chiark.greenend.org.uk/~sgtatham/putty/`). Unfortunately, the `.pem` file you just downloaded (with your private key) is not supported by PuTTY. In order to log in by using your key pair, you must first convert this private key to a format usable by PuTTY. For this, run PuTTYgen, click on "Load", and select the `.pem` file (you may have to force the file dialog to show you all file extensions). Now, click on "Save private key" and choose the location to save the key to. The key is now stored in a `.ppk` format that you can use with PuTTY. Open PuTTY and enter the hostname or IP address in the [Session] window. To specify a username, go to the [Connection → Data] window, and enter the login details. Finally, to login using a private key, go to [Connection → SSH → Auth] and select the `.ppk` file in the "Authentication parameters". Go back to [Session], enter a name (e.g. `server1`) in "Saved sessions" and click on "Save". PuTTY will now remember the configuration for `server1`. You will just need to double click on the session name to initiate a connection.

2 Customize your instance

1. Connect to the VM via `ssh` using the previous key and the user name `ubuntu`. The public name of the VM is available in its description.
2. Install the required packages:

```
$ sudo su
$ apt-get update
$ apt-get install apache2
$  apt-get install libapache2-mod-php
```

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

```
$ systemctl enable apache2
```

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

4. Create the following PHP script in `/var/www/html/bench.php`:

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

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

5. Modify the security group (last column on the right of the line defining the instance) of your instance so that it accepts HTTP traffic.

6. Restart the `apache2` daemon and test if the PHP script works.

3 Creating a template (AMI in AWS vocabulary)

1. Stop your instance (do not terminate it!!!). In the EC2 console, go to "Instances", select your instance and choose "Create Image".

2. Make sure the image is available in "Images → AMIs".

3. Once this is done, go to the "EC2 Dashboard" and create a new instance out of your template ("My AMIs" item on left side menu) , using the same keys.

4. Check that the Web server is indeed accessible from outside by testing the `bench.php` script.

4 Load Balancing

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

1. Create a load balancer from the "Load Balancing" (pick a classic one, not an application specific LB) menu and associate your two instances.

2. Check the status of the VMs and one or the two are declared OutofService, check the security group to see if there is no HTTP issue.

3. Find the name of the load balancer and check that it works. In case of problem, be sure that its security group is configured appropriately.

4. Once it is up and running, check that the two VMs are accessible by looking at the DNS resolutions of the Load balancers (nslookup, dig, host,etc...)

5. How is load balancing performed by the DNS server?

6. You can now terminate the two VMs.

5 Horizontal Scaling

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

1. Click on "Auto-Scaling Groups" and follow the instructions. In particular :
   (a) Pick your AMI as a template
   (b) Associate it to the load balancer you created previously
   (c) Define a scale up rule based on the average CPU load with a maximum of 5 instances
2. Your scaling group is now ready for testing.
   (a) 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. Increase the Fibonaci numbers computed in the php script (beware that it ramps up quickly) so as to increase the load.
   (b) Put in your report the tab with the history of your scaling group to demonstrate that you achieve the desired properties.
3. Delete the Scaling groups. After a few minutes, all the VMs should have been terminated automatically.

6 Virtual Private Clouds (VPC)

A VPC enables the creation of a set of private LANs by a tenant to deploy its VMs. Using the VPC creation wizard (it is a different service than EC2!) allows you to create a network prefix hosted over several availability zones within a data center.

In fact, there is always a default VPC associated to a AWS account. The default VPC allows you to create VMs with both private and public IP addresses. This is a good starting point to discover AWS but not suitable if you want to host part of your network in the cloud. You need to create a large network of private addresses, decide how to subdivide it and how to connect to the outside

6.1 VPC creation

Use the wizard for creating a new VPC with the default values. It will allow you to:

- Create a VPC with a /16 IPv4 block. Assign it a name you can easily find later, eg. VPC1 or MyVPC.
- Attach an Internet gateway to the VPC.
- Create a /24 IPv4 subnet (a range of 256 private IP addresses) in the VPC.

This corresponds to the diagram of Figure 1.

6.2 Creation of a first VM within the VPC

1. In the VPC panel, create a security group for the Web servers with SSH and HTTP allowed, plus all traffic from your VPC addresses (likely to be 10.0.0.0/8).
2. In the EC2 panel, create a VM (call it VM_pub), using your AMI template, and place it in the public subnet of your VPC. Assign to this VM your previously created security group. You will have to select "Configure Instance Details" to assign the VM to the appropriate VPC.
3. In the VPC panel, create an elastic IP address and assign it to your instance.
4. Check that you can connect to your web server.

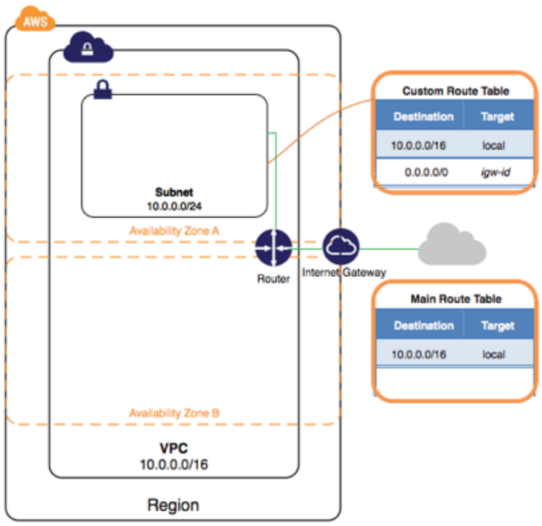

Figure 1: VPC initial set-up

6.3 Creation of a second subnetwork

Now that we have a machine running in our public network:

1. Create a new subnetwork in a different availability zones.
2. Create two machines, VM_priv_1 and VM_priv_2 in this subnet. Assign them to the correct security group.
3. Those machines can be contacted from VM_pub, but they cannot go to the Internet. Just like in a typical enterprise network, we might simply add a NAT. You can do this in the VPC control plane, associating the NAT gateway to your initial public subnetwork (not the newly created private one as the NAT gateway must access the Internet) and generating an elastic IP address at creation. Next, you have to edit the main route table (see the Main column in the route table of the Route Table of the VPC panel) adding a default route (to 0.0.0.0/0) via your NAT for the private subnet.

6.4 A bit of measurements

- As we expect availability zones to correspond to different physical areas, do some ping (average over at least 100) and iperf measurement (10 times) between your 3 VMs to .... see what you obtain and the stability of results.

7 Delete everything!!!

Stop your VMs and all services (e.g. horizontal scaling) to avoid receiving any unpleasant invoice in a few months.

For the VPC, first kill manually all the VMs (kill = terminate in AWS parlance) inside and then, when you kill the VPC, it will remove the internet gateway + the NAT that goes with it.

To remove the AMI, you have to disassociate it and then kill the snapshot associated to the image (you cannot kill the image directly).