Overview

This week we will enable autoscaling for a web application. Like the previous assignment, you will have to work through some of the detailed steps to complete the requirements. If you get stuck on a step, do a little research on the AWS documentation site or reach out to the class on Slack.

Requirements

You need to have a personal AWS account and GitHub account for this assignment.

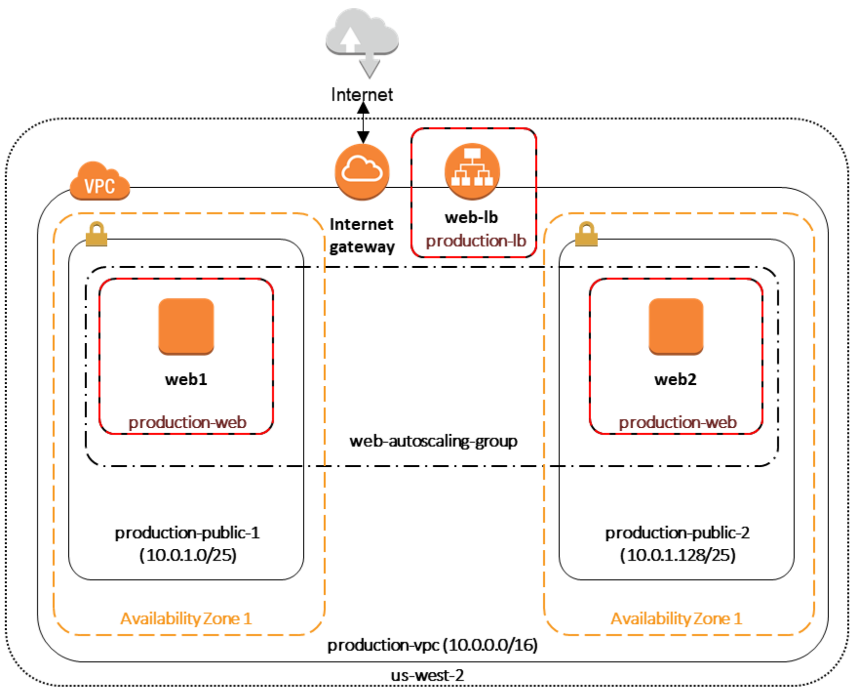
The assignment

Let's start scaling.

Create VPC and subnets

Each assignment in this course has been less prescriptive than the previous and this assignment is no different. By now, you should understand how to configure a VPC, subnets, routing tables, an internet gateway, an elastic load balancer, and EC2 instances. You will create all of these components and more in this week's assignment. Remember, you can always look back at previous assignments, class recordings, or online AWS documentation if you get stuck on a step.

You will build a VPC and set of AWS services which conform to the service diagram below.



Start the assignment by creating a VPC named `production-vpc` in the `us-west-2 (Oregon)` region with a CIDR block of `10.0.0.0/16`. Note, we're using a different region than previous assignments! The VPC needs to have two subnets located in **separate** availability zones (your choice):

- The `production-public-1` subnet will use the `10.0.1.0/25` network.
- The `production-public-2` subnet will use the `10.0.1.128/25` network.

All instances on the two production subnets will need to be able to communicate with the Internet. You will need to configure the subnets and routing tables in the VPC to allow this communication. How do you do this? You should be able to use the knowledge you gained from previous assignments to figure this out.

Create Security Groups

While you are in the VPC dashboard, go ahead and create a couple security groups. You will need to use these security groups when launching services later in the assignment. It's always a good idea to plan out the security requirements for your services in advance.

Create a security group called `production-lb`. This security group will accept traffic from the Internet and distribute it to web servers. The requirements for the security group include:

- production-lb
    - Name tag: production-lb
    - Description: production load balancer
    - VPC: production-vpc
    - Inbound:
        - HTTP allowed from the Internet

Create another security group called `production-web`. This security group will allow all external devices to communicate with the web server instances via SSH and HTTP. Here are the requirements for this security group:

- production-web
    - Name tag: production-web
    - Group name: production-web
    - Description: production web servers
    - VPC: production-vpc
    - Inbound:
        - SSH allowed from your workstation
        - HTTP allowed from the Internet

Create an application load balancer

The autoscaling group will deploy multiple web servers into your VPC. You will need an application load balancer to distribute requests to the web servers to ensure that the service is highly available.

Go to the EC2 dashboard and select the `Load Balancers` menu item. Create a new application load balancer called `web-lb` that will distribute HTTP (port 80) requests across web servers located in both us-west subnets. The load balancer should use `/index.php` as the health check ping path. You won't be able to associate any EC2 instances with the load balancer yet. That's not a problem since you can associate an instance with the load balancer later.

Note, make sure you select the correct VPC, subnets, and security group settings for the load balancer. Review previous assignments if you are unsure how to set these properties. Also, don't forget to tag the load balancer.

Create a load balancer target group called `production-webservers`. Remember to associate the proper subnets with this target group. The target group will remain empty for now.

Create web server instance and AMI

The next step is to create a custom AMI containing a basic webserver installation. You can use the new AMI to create an autoscaling launch configuration.

Launch an EC2 instance with the following properties:

- Amazon Linux AMI 64-bit
- t2.micro
- Network: production-vpc
- Subnet: production-public-1
- Auto-assign public IP: Enable
- IAM role: S3-full-access
- User data (in Advanced section):

```
#!/bin/bash
yum update -y
yum install -y git httpd php
service httpd start
chkconfig httpd on
aws s3 cp s3://seis665-public/index.php /var/www/html/
```

- Tag: Name = webserver1
- Security group: production-web

Note, if this is the first time you are launching instances in the us-west region, AWS will likely prompt you to create a new set of SSH access keys. You should give this keypair a different name than your other key pair. Also, if you are running Putty on Windows you will need to use the Puttygen tool to create a private key that you can import into Putty.

Once the instance is running, open a web browser on your desktop and browse to the public IP address of the new instance. You should see a test page appear in your browser. The user data script automatically updated the server and installed a number of basic components, including the Apache web server, PHP, and a MySQL library.

You can shutdown the EC2 instance now after you have confirmed that the server is running properly. Make sure you shutdown the instance, not terminate it! Stopping the instance will allow the data on the server to quiesce. After the instance stops, create an image based on the instance called `basicweb`. AWS will take a few minutes to build the new AMI. Make sure the AMI is configured to use hardware-assisted virtualization.

Go back to the EC2 dashboard and terminate the stopped instance.

Create an Auto Scaling group

The next step in the assignment is to create a launch configuration and auto scaling group. Select the `Launch Configuration` menu item and click on the button to create a new auto scaling group. Here are the properties for the new launch configuration:

- AMI: basicweb (located in your personal AMI listing)
- Type: t2.micro
- Launch configuration name: web-launch-config
- IP Address Type: Assign a public IP address to every instance. (located in advanced section)
- Security group: production-web

The auto-scaling group should have the following properties:

- Group name: web-autoscaling-group
- Group size: 2
- Network: production-vpc
- Subnets: production-public-1 & production-public-2
- Receive traffic from Elastic Load Balancers: Enabled
  - Select the `production-webservers` target group
  - Health check type: ELB
- Scaling policy: Keep this group at its initial size
- Tag:
  - Key = Group
  - Value = basicweb

Take a look at the Activity History of the auto scaling group you just launched. You should see a couple pending activities. If you see any failure messages, that means either the launch configuration settings or the auto scaling group settings are incorrect. You may need to delete the new launch configuration and scaling group to build the configuration again.

If the autoscaling process is working properly you should see two new instanced being launched in the EC2 dashboard. Once the instances are running, locate the DNS name of your load balancer and type it into your browser. You should see the index page from your website. Hit the refresh button on your browser a few times so that the load balancer redirects the request to a different server. Now you have a highly available website that is being managed by an auto scaling group.

Simulate an auto-scaling failure

Auto-scaling is designed to respond to events such as a failed instance or the CPU load on an instance increasing above a certain threshold. Let's simulate an instance failure by terminating one of the running instances. It doesn't matter which instance you choose.

After terminating one of the running instances, wait a few minutes and check out the list of EC2 instances on the dashboard. You should see a new instance starting up. Auto-scaling will automatically maintain two running instances at all times.

Collecting session data

Connect to your original web server instance using a terminal program. Let's pull some event information from the EC2 auto-scaling service.

First you need to configure the AWS CLI using your credentials. Type in the following command:

```
$ aws configure
```

The CLI will prompt you for your API credentials. Your AWS user account has an Access Key and a Secret Key configured to allow you to programmatically access the AWS API. The keys are located in your account properties in IAM. If you do not know your secret key, you can use IAM to generate a new one since previously generated keys cannot be recovered. Note, this secret API is not the same as your private server key.

The default region name for the AWS CLI configuration is: `us-west-2` and select the `json` output format.

Type in the following command to view all the autoscaling activities and copy them to a file called `activity.json`:

```
$ aws autoscaling describe-scaling-activities --auto-scaling-group-name web-
autoscaling-group > activity.json
```

Take a look at the contents of the `activity.json` file. You should see a long text output including a list of activities with each having a Description, ActivityId, StartTime, etc.

Create a small batch script called `getdata.sh` which performs the following:

- Copies the /var/log/httpd/access_log file to the current directory into a file with the same name

Check your work

Here is what the contents of your git repository should look like before final submission:

```
├ access_log
├ activity.json
└ getdata.sh
```

Submit your work

Check each of the files to make sure the files contain data. Add all of the files to the Git repository and commit your work.

Finally, create a new GitHub Classroom repository by clicking on this link: https://classroom.github.com/a/wYUzVn1U

Associate your local repository with this new GitHub repo and push the local master branch from your repository up to GitHub. Verify that your files are properly stored on Github.

Terminate server

The last step in the assignment is to delete the auto scaling group, launch configuration, EC2 instances, EC2 load balancer, AMI, EC2 snapshots, and VPC. I'll leave this as an exercise for you to figure out how to complete.

Remember, you will get billed for each hour these services are running (or at least lose free credits). You launched quite a few services this week, so if you don't terminate them you will have a nice bill waiting for you at the end of the month!

Submitting your assignment

I will review your published work on GitHub after the homework due date.