Final Project Instructions

Open note, open tech, but not open friend. Do the project on your own. Earn your grade.

For the Final Project, you will start to **build out a solution like the one you probably proposed in your Group Midterm Project and ones we've examined in the lab exercises**.

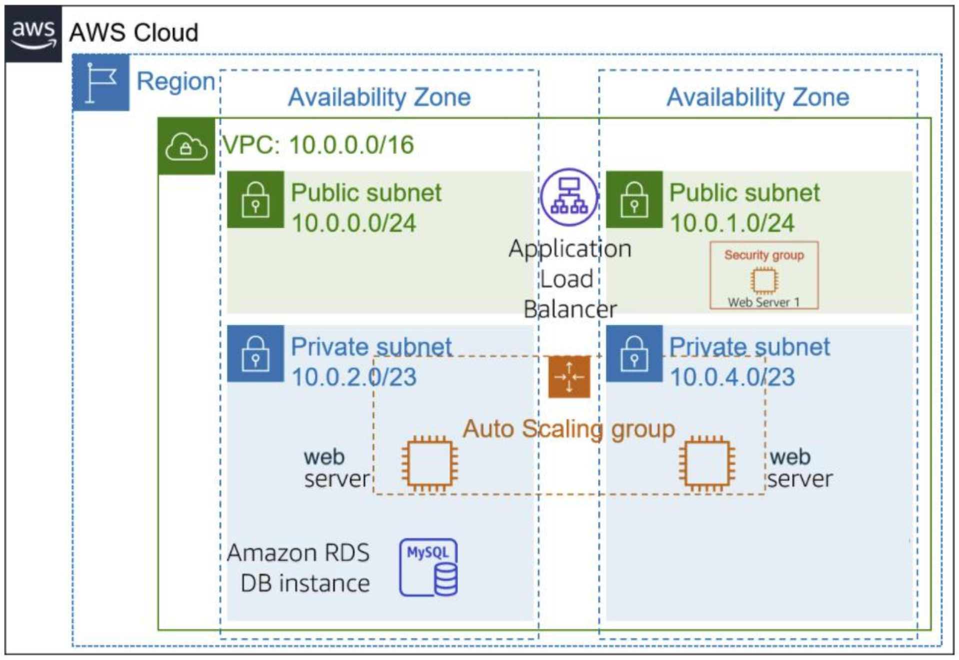
To accomplish this, you will work in the AWS Management Console from your AWS Academy Learner Lab account**.** See the announcement in our Canvas class titled "**You've been invited to enroll in the AWS Academy Learner Lab - Associate Services course**".

I want you in the **N. Virginia region**, so if your account puts you in a different region, select **N. Virginia** from the drop down to the right of your userid in the black banner across the top of the AWS Management Console.

You will upload a collection of screenshots through Canvas that documents what you were able to accomplish. You should upload a single file with all the screenshots in it**. Each screenshot must include the black banner across the top of the AWS Management Console which includes your unique login ID. NO CREDIT FOR SCREENSHOTS THAT DO NOT INCLUDE YOUR LOGIN ID.**

Alternatively, you can create a well-documented CloudFormation template to build the environment. Be sure to read the instructions to make sure you've considered everything, not just the diagram. If you choose to complete the assignment this way, you will submit your JSON or YAML template instead of the screenshots.

This is the environment you are mostly going to build. **If you can't do all of it, please do some of it so you earn partial credit. If you've been doing lab exercises, this will have the feel of a long lab exercise.**



To receive full credit for your project, you should accomplish and document the following:

- **VPC**: Allocate an Elastic IP address from Amazon's pool of IPv4 addresses and **screenshot the list of Elastic IP addresses**

- **VPC**: Create a VPC named **FinalVPC** that looks like the diagram. **Pay attention to the subnet CIDR blocks.** YOU WON'T BE ABLE TO USE THE VPC WIZARD SINCE IT ISN'T FULLY SUPPORTED YET IN AWS LEARNER LAB ACCOUNTS.
    - Create the VPC (default tenancy not dedicated)
    - Create the 4 subnets (public/private pair in one AZ and public/private pair in another). Name them sensibly (eg Public Subnet 1, Private Subnet 2, etc.). Create and attach an Internet Gateway named **Final-IGW**
    - Create a NAT Gateway named **Final-NAT** and use the Elastic IP you created. Place it in your 10.0.0.0/24 subnet.
    - Create a public route table (hint: include an Internet Gateway route for any address), add routes, and associate with public subnets
    - Create a private route table (hint: include a NAT Gateway route for any address), add routes, and associate with private subnets

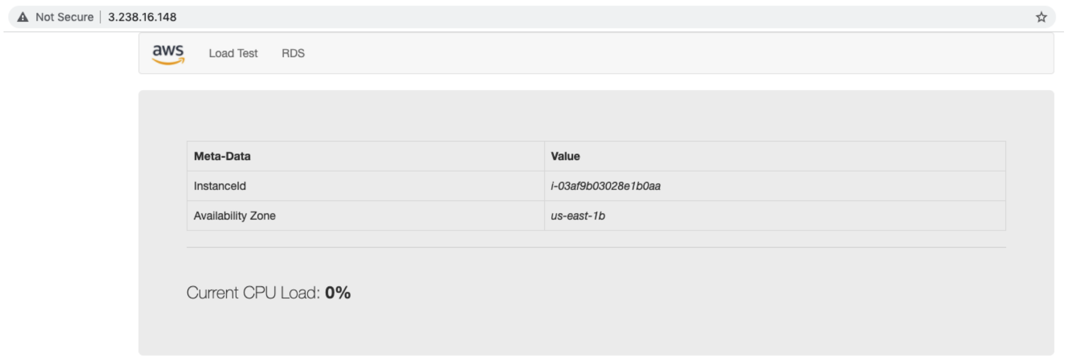
- **VPC**: **Screenshot the list of VPCs (with Details showing) and the list of subnets (with VPC, CIDR and Availability Zone columns showing).** You can decrease the width of the columns displayed by clicking on the column separator (|) and sliding it to the left. That will help you get all the columns I want to see on the screen.

- **EC2**: Launch an EC2 instance into your second PUBLIC subnet and tag it with the name **TigerEC2**
  - Instance type t2.micro, Amazon Linux 2 AMI (HVM), SSD Volume Type
  - **Enable** Auto-assign Public IP
  - Use this user data to bootstrap your instance:

```
#!/bin/bash

# Install Apache Web Server and PHP

yum install -y httpd mysql php

# Download Lab files

wget https://aws-tc-largeobjects.s3.amazonaws.com/AWS-TC-AcademyACF/acf-lab3-vpc/lab-app.zip

unzip lab-app.zip -d /var/www/html/

# Turn on web server

chkconfig httpd on

service httpd start
```

  - Create 17 GiB storage volume, type gp2, 100 IOPS, encrypted using default aws/ebs keys
  - Create a security group named **Web Security Group** which allows SSH and HTTP traffic
  - You won't need SSH access for this project, but you can "Choose an existing key pair" and check the acknowledgement box in the keypair dialog box when launching the instance
  - **Screenshot the list of running EC2 instances with the TigerEC2 instance selected**. I want to see the description information for that instance in your screenshot
  - **Screenshot the list of EBS volumes with the TigerEC2 volume selected**. I want to see the description information for that volume in your screenshot

- **EC2**: After your EC2 instance is running, visit the public IP of your EC2 instance using your browser. You should see the Amazon application screen. **Screenshot your browser window with the IP address showing**.



Meta-Data	Value
InstanceId	*i-03af9b03028e1b0aa*
Availability Zone	*us-east-1b*

Current CPU Load: **0%**

- Now that you've confirmed that the application works, you want to create PRIVATE EC2 instances. The typical way you would do this is to create an AMI from your running EC2 instance and then launch EC2 instances using your custom AMI. Unfortunately, due to a problem with the Learner Lab environment, we must do it another way by **using the same bootstrap information you used to create TigerEC2**.

- **EC2**: Now you are going to create TWO (2) new EC2 instances **using the same bootstrap userdata above that you used to create your public EC2 instance.** You will put one of your new EC2 instances in your first PRIVATE subnet and the other one in your second PRIVATE subnet.

- **This is only going to work if you have set up your NAT Gateway and private routes correctly because the private instances you are going to create will need to pull software from the public internet as shown in the bootstrap user data.**

- **EC2**: Launch an EC2 instance into each of your PRIVATE subnets and tag them with the name **TigerEC2-PV1 and TigerEC2-PV2**
    - Instance type t2.micro, Amazon Linux 2 AMI (HVM), SSD Volume Type
    - **Disable** Auto-assign Public IP since the instance is going into a PRIVATE subnet
    - Use this user data again to bootstrap your instances:

```
#!/bin/bash

# Install Apache Web Server and PHP

yum install -y httpd mysql php

# Download Lab files

wget https://aws-tc-largeobjects.s3.amazonaws.com/AWS-TC-AcademyACF/acf-lab3-vpc/lab-app.zip

unzip lab-app.zip -d /var/www/html/

# Turn on web server

chkconfig httpd on

service httpd start
```

-
    - Create 17 GiB storage volume, type gp2, 100 IOPS, encrypted using default aws/ebs keys
    - Use the existing security group named **Web Security Group** which allows SSH and HTTP traffic
    - You won't need SSH access for this project, but you can "Choose an existing key pair" and check the acknowledgement box in the keypair dialog box when launching the instance
    - After the instances come online, you can select an instance then click on Actions->Monitor and troubleshoot->Get system log. Inspect the log to see if the software was successfully downloaded to your private instance.

- **EC2**: Create an Application Load Balancer with both new EC2 instances in the target group
  - Create a Target Group named **TigerTargets** and register both your PRIVATE EC2 instances in the target group. Include them as pending and then register pending targets.
  - Name it **TigerELB**
  - Be sure to specify the **FinalVPC** VPC and forward traffic to **TigerTargets**
  - Create the load balancer and wait for it to come online (it will be in the Provisioning state for a short while before becoming Active)
  - <u>Screenshot **the list of load balancers with TigerELB selected**</u>
  - <u>Screenshot **the target group and show the Targets information**</u>
  - Using your browser, visit the public IP of your Load Balancer. <u>Screenshot **your browser window with the IP address showing.**</u>
  - If you've done everything correctly, you should see the same web page you did when you were testing your baseline web server TigerEC2.
  - If you keep refreshing the page, you should see that your load balancer is routing traffic to both your private EC2 instances

- **EC2 Autoscaling:** Configure EC2 autoscaling. Create a launch configuration named **TigerASLC** and use this information to do it.
  - For the AMI to use, search for **ami-02e136e904f3da870 .** Normally, we would create our own AMI and select it, but that is not supported in the Learner Lab environment.
  - Instance type t2.micro, use default storage values
  - Use the existing security group named **Web Security Group** which allows SSH and HTTP traffic
  - Under Advanced Details, use the usual bootstrap user data:

```
#!/bin/bash

# Install Apache Web Server and PHP

yum install -y httpd mysql php

# Download Lab files

wget https://aws-tc-largeobjects.s3.amazonaws.com/AWS-TC-AcademyACF/acf-lab3-vpc/lab-app.zip

unzip lab-app.zip -d /var/www/html/

# Turn on web server

chkconfig httpd on

service httpd start
```

- Don’t allocate public addresses for these autoscaled instances
- You won't need SSH access for this project, but you can "Choose an existing key pair", select vockey, and check the acknowledgement box in the keypair dialog box before creating the launch configuration
- Screenshot the **TigerASLC** launch configuration by selecting it from the launch configuration list. Include the details in the bottom pane.

* EC2 Autoscaling: Create Auto Scaling Group named **TigerAS.**
  - **Specify to launch from configuration TigerASLC.** You might have to click “Switch to launch configuration” before selecting **TigerASLC**.
  - Be sure to select the FinalVPC VPC and both PRIVATE subnets
  - Attach your autoscaling group to an existing load balancer and select the **TigerTargets** target group.
  - Group size should be 2 desired, 2 minimum, 6 maximum
  - Use a target tracking scaling policy and accept the default **Average CPU utilization** metric
  - No additional notifications will be configured
  - Tag your autoscaling instances with the name **TigerAS-Instance** before creating your autoscaling group
  - Screenshot **the TigerAS autoscaling group by selecting it from the autoscaling group list. Include the group details in the bottom pane.**
  - You should see **two new EC2 instances named TigerAS-Instance come online** as EC2 autoscaling increases capacity to meet your group size configuration.
  - Screenshot **the TigerTargets target group with the Targets tab clicked.** You should see 4 healthy instances – TigerEC2-PV1, TigerEC2-PV2, and 2 TigerAS-Instance instances

- **DB**: Create a new RDS instance (MySQL) named **ClemsonDB.** We're not going to connect it to anything for this project. We're just going to create it.
  - Create a DB Subnet Group named **TigerRDS-Subnets** which includes your two PRIVATE subnets
  - DB Instance Size: Burstable db.t2.micro or db.t3.micro; Storage Type: General Purpose SSD
  - **Disable storage autoscaling and do not create a standby instance**
  - Select the **FinalVPC** VPC and the **TigerRDS-Subnets** subnet group
  - No public access
  - Create a new security group named DB Security Group
  - Select your first AZ
  - Additional configuration information:
    - Set your initial database name to be **TigerData**
    - Disable Enhanced Monitoring
  - Create database
  - After your database is up, click its name and **screenshot the Connectivity & Security information and screenshot the Configuration information**

Good luck!