Assignment 1

Note:

This assignment focuses on bringing IaC concepts together in a business solution via a mini-network. Best practice is required throughout. A class period will be allocated to initial discussion on the assignment to enable team work in teasing out the problem(s) at hand. The assignment due date is as shown in Blackboard.

LO's covered by this assignment are:

1. Engage in practitioner-based research to compose solutions to deployment pipeline efficiency issues
2. Conceptualise, research and critique techniques and technologies to automate processes and job lifecycles
3. Undertake an analysis of incomplete, incorrect or contradictory solutions to a problem and communicate in a professional manner an alternative approach underpinned by justifications informed by clearly articulated research.
4. Critical awareness of the importance of coding best practice in disseminating information to enhance teamwork.
5. Analyse the impact of scripted pipelines on enhancing team work in delivering software across a pipeline.
6. Devise a scripted solution for a software delivery pipeline including design, implementation and test phases.
7. Review and refactor scripts via peer-review retrospective and refinement meetings.

Submission:

You are required to submit:

1. All cloud formation files.
2. A word/pdf document with:
   2.1. A link to your github repo with code to enable review of continuous updates to your files.
   2.2. A two page conclusion.
   2.3. An appendix of images of your final system running in AWS.

Create a Cloud Formation Stack using the designer.

The purpose is to replicate the mini network. Refer to the image for the example starting architecture.
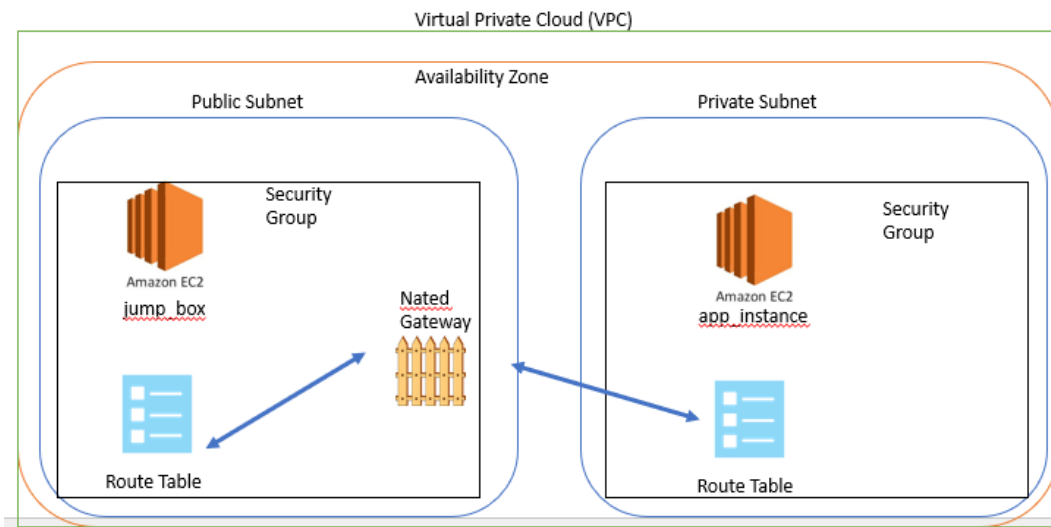
Figure 1 Initial Network Architecture

In this starting point you can see a Virtual private cloud, availability zone and two subnets. One subnet is public facing whilst the other is private. Within each are security groups. Your EC2 instances (servers) should be tied to the appropriate groups. Make sure you have a jump_box to connect via ssh to the instances in the private subnet. Set up appropriate routing tables and gateways.

Expand your network so it ends up like Figure 2. J is your jump_box. F is any other front facing server. A is the app_instance as per the above. D is any database you wish.
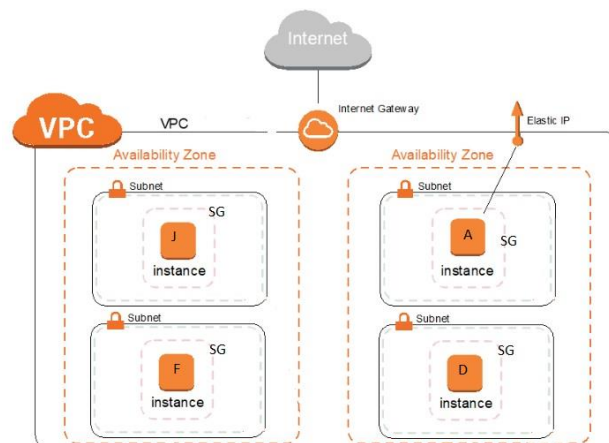

Figure 2 Final Network Architecture

To solve this problem you will need to carry out a number of steps. Some hints on relevant steps are given in the next page.

1. Create a suitable set of IP addresses for your network. Remember that your routing tables will rely on this as will your security group. Fill in the table for yourself for each part of the network. Remember that we want to deny by default and only allow whatever is essential to each subnet and security group.

Type	Protocol	Port	Source	Allow/Deny
SSH	TCP	22	(CIDR IP Address here)	Allow (or deny depending on where this rule is used!

2. Create routing tables to help with allowing communications from one subnet to another. The VPC address is a top level address so allow sufficient room for all your ip addresses. E.g. 10.1.2.0/16 or /24 depending on your needs. When working out the number of addresses you need don't forget to count the internet gateway.

3. An elastic ip address can be used to enable servers to be seen by the outside network in a production environment. However, as they cost money to run, you can simply use internal addresses here.

4. Link the components in your stack appropriately to make a mini-network.
   4.1. Test to ensure it works.
   4.2. Screen grab the results for the written report.

5. Tidy the file to make it reusable:
   5.1. break it up;
   5.2. Use parameters and mapping

6. In the conclusions discuss how you applied or considered:
   6.1. Best practice to tidy up the code
   6.2. Security
   6.3. Reliability and high availability (consider spreading across availability zones).

Plan carefully before you begin to code! What do you need? How do they connect? How do you keep it secure? Can you make the code clean and reusable?