**TriNimbus AWS Assignment**

Created by: Imran Alli

Date: 21st June, 2018

Table of Contents

[1 Design 3](#_Toc517364478)

[2 General Info 3](#_Toc517364479)

[3 Operating System 4](#_Toc517364480)

[4 Networking 4](#_Toc517364481)

[5 Website Content 4](#_Toc517364482)

[6 Instructions to Execute the Script 5](#_Toc517364483)

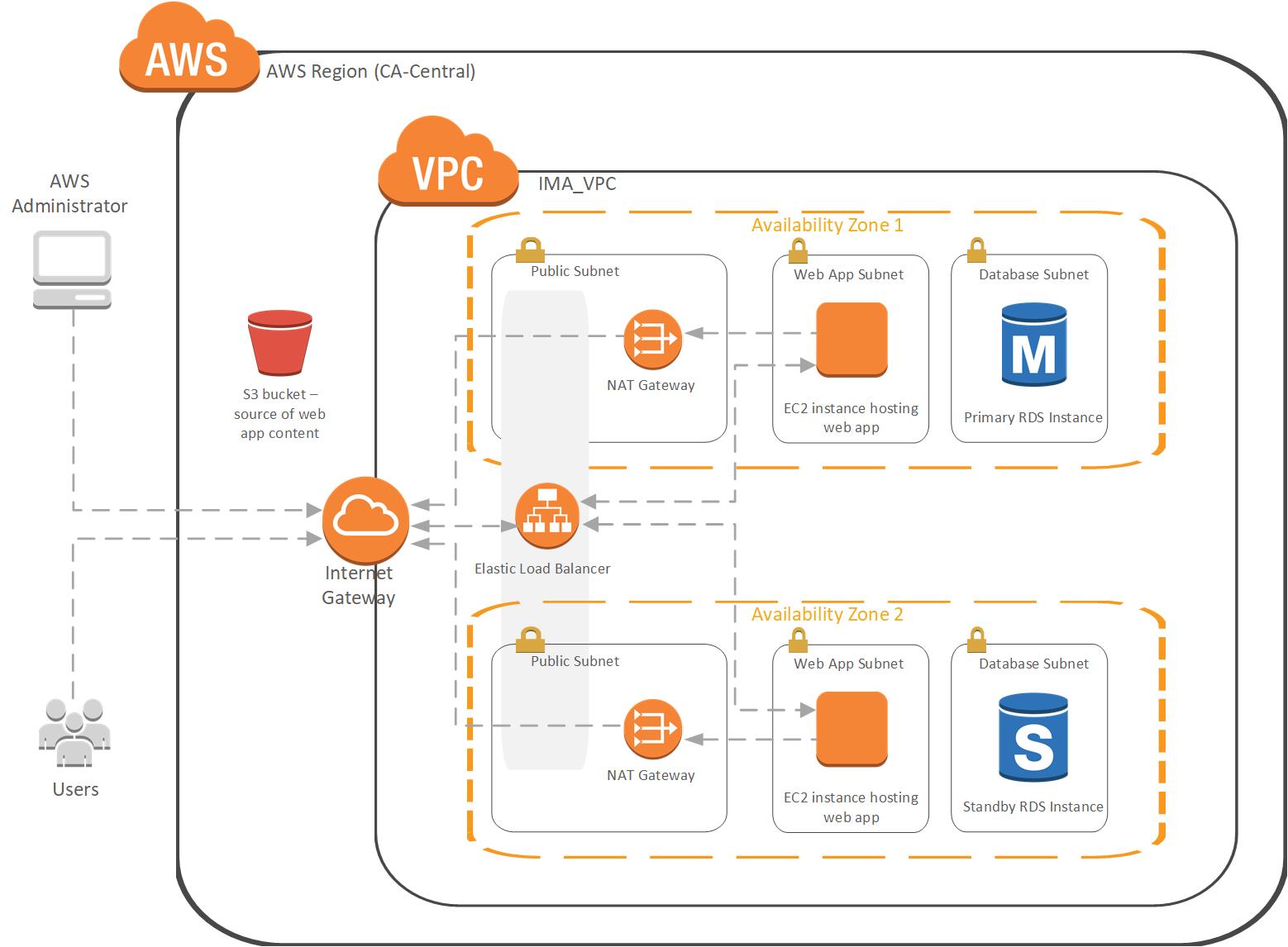
[6.1 Manual Steps Required 5](#_Toc517364484)

[6.2 Executing the script 6](#_Toc517364485)

[7 High Level Outline of the Deployment Script 7](#_Toc517364486)

# 

# Design



# General Info

For this assignment, a web app was built utilizing AWS infrastructure that is deployed via a BASH script. The design of the infrastructure is depicted in the image in the previous section. The app is accessible over HTTPS (using a self-signed certificate) and allows the user to input data to a MySQL database, and also retrieve data from it. The database is hosted by Amazon’s RDS service.

Both the web and database servers have been deployed in two availability zones in the Canada-Central region for high availability. All resources have been tagged with the prefix “IMA” in front of the tag.

Note that for this project, certain minor aspects and best practices were skipped in the interest of simplicity (Eg. Error checking, IAM roles, permissions, security group port blocking and other security related restrictions).

The entire script has been tested multiple times (from an Ubuntu machine) and takes just under 30 minutes to complete.

# Operating System

The web application was built on a LAMP stack and Ubuntu server. To make the process a bit simpler, a pre-built LAMP AMI from Bitnami was used (“LAMP with PHP 7.1 Certified by Bitnami”). This AMI is free and can be found on the AWS Marketplace.

Link to AMI : <https://aws.amazon.com/marketplace/pp/B072JNJZ5C?qid=1528984609535&sr=0-3&ref_=srh_res_product_title>  
  
Note: You must manually subscribe to this AMI by performing the steps in the “Manual Steps Required” section of this document prior to launching the AWS infrastructure build script for the first time. This manual step is required since there doesn’t seem to be a way to subscribe to an AWS Marketplace AMI from the CLI.

# Networking

The AWS deployment script provisions the resources within a VPC. The web and database servers reside in private subnets and access the Internet via NAT gateways. One EC2 instance is deployed in ca-central-1a and ca-central-1b. An application load balancer is provisioned to load balance user traffic across availability zones.

# Website Content

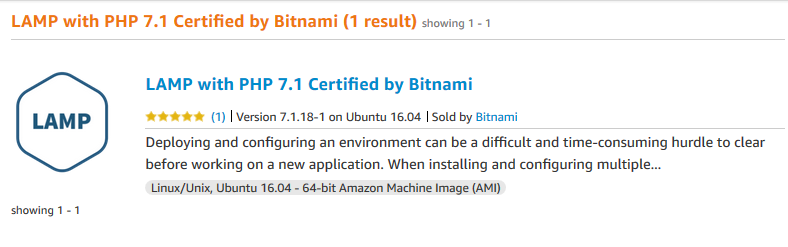
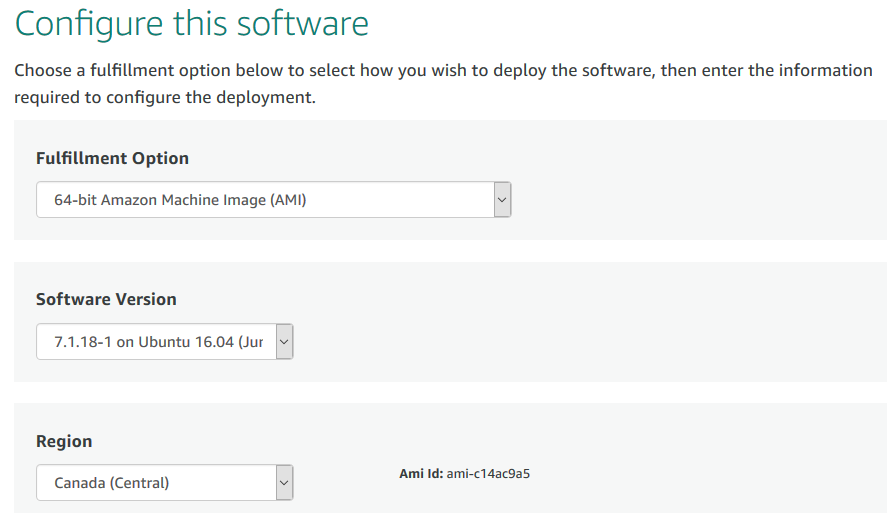
A free simple website template which uses HTML and CSS was downloaded from https://startbootstrap.com. Minor modifications were made to the template’s HTML and CSS, and some PHP code was added in order for the site to accept the user’s data from a web form, then write to or retrieve data from the database.

The EC2 instances downloads a zip file from S3 containing the website content, then the content is served via Apache.

# Instructions to Execute the Script

## Manual Steps Required

The steps below are only required prior to running the AWS infrastructure build script for the first time. These manual steps are needed since there doesn’t seem to be a way to subscribe to an AWS Marketplace AMI from the CLI.

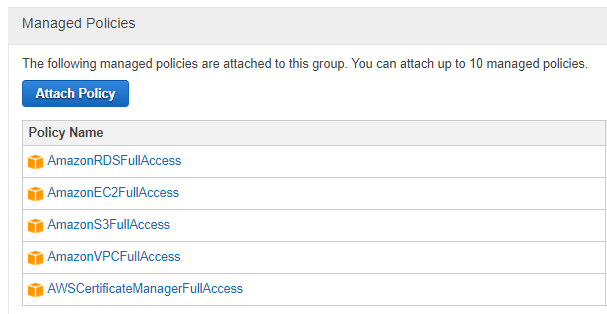
* + 1. Go to <https://aws.amazon.com/marketplace/> and login with the AWS credentials provided to TriNimbus.
    2. Search for “LAMP with PHP 7.1 Certified by Bitnami”.  
         
       
    3. Follow the steps to subscribe to the AMI. Choose the Canada region.   
         
         
       Note Amazon provides the AMI ID on the above page. This ID was used in the build script to launch instances from this AMI.

## Executing the script

The following files are required, and must all be in the same directory.

**“deploy\_script”** - This is the script that will deploy the entire environment.   
**“deployment\_inputs.txt”** – The script reads from this file in order to determine how to set up certain aspects of the environment (Eg. Subnet addresses, resource tags, database credentials, etc.). These parameters can be changed if the user desires, or if he or she would like to deploy multiple environments using the same script (Eg. Test, Development, Production, etc.)  
**“server.crt**” – SSL self-signed certificate.   
“**server.key**” – key for SSL certificate.  
**“userdata\_script.txt**” – The bootstrap script used when launching the EC2 instances.   
  
Credentials for the TriNimbus IAM user have also been provided. The access keys provided must be used when executing the script. The credentials can also be used to login to the GUI. The ‘**trinimbus’** IAM user has been given sufficient permissions for the script to deploy the resources successfully.

IAM permissions of ‘trinimbus’ user:



To execute the script, configure the AWS CLI with the ‘**aws configure’** command. Use the credentials provided and the ca-central-1 region. Then, you can use the “**sh** **deploy\_script”** command to run the script. It takes just under 30 minutes to provision all the resources. A URL will be displayed upon completion which can be used to access the web app. Note that since a self-signed certificate was used, your browser will display a warning when accessing the site.

# High Level Outline of the Deployment Script

Below is a high-level outline of the tasks accomplished by the deployment script.

1. Read config file to store user's parameters as variables.
2. Create VPC.
3. Create subnets for web servers, database and public subnets for load balancer.
4. Create Internet Gateway and attach to VPC.
5. Allocate elastic IP's for use with NAT gateways.
6. Create NAT gateways (1 in each AZ).
7. Add default route to main route table to route to Internet Gateway.
8. Create additional route tables for private subnets to route to NAT gateway.
9. Set default routes to route to NAT gateways.
10. Set private subnets to route to NAT gateways.
11. Create security groups and add rules.
12. Create RDS MySQL database instance (Multi AZ).
13. Pass RDS database login info to userdata\_script for use by web app.
14. Create EC2 instances, 1 in each AZ, and use user data script to set them up.
15. Create Elastic Load Balancer (Application Load Balancer).
16. Create ELB Target Group.
17. Register instances to the Target Group.
18. Import SSL certificate to AWS Certificate Manager (ACM).
19. Create ELB HTTPS Listener and associate with SSL certificate.
20. Output URL for user to access web app.