After we down load and configure terraform, make sure we run it under an account which has IAM privileges to provision the AWS resouces.

The credentials may be easily set as:

export AWS\_ACCESS\_KEY\_ID=your\_access\_key

export AWS\_SECRET\_ACCESS\_KEY=your\_secret\_access\_key

Using the method of your choice, write automation to create the following web application stack. You should submit your automation assets that we can run to provision the environment. If you have any questions then just make your best guess at our intent and note any assumptions you've made when submitting your assets.

We need to set a VPC environment for our apps to live in.

The default VPC info, which can be overridden in the command line are:

region = "us-west-2"

vpc\_name = "terradata"

vpc\_cdir = "10.190.0.0/16"

We want to build a secured, and High-Availability environment. We do a look up of the region to find all the Availability Zones. For each zone we build a public subnet, and a private subnet. We use certain convention to divide the CDIR for each subnet. For example, the us-west-2 region has 3 AZs: us-west-2a, us-west-2b, and us-west-2c then our subnets will have names, and CDIRs:

terradata-public-us-west-2a 10.190.0.0/24

terradata-public-us-west-2b 10.190.1.0/24

terradata-public-us-west-2c 10.190.2.0/24

terradata-private-us-west-2a 10.190.3.0/24

terradata-private-us-west-2b 10.190.4.0/24

terradata-private-us-west-2c 10.190.5.0/24

We provision an IGW, attach it to the VPC, and also 3 Nat Gateways, one in each AZ. The route tables will route traffic from public subnets to the IGW, and private subnets to the Nat in the same AZ. With security, and High Availability in mind, we build 3 Bastion hosts, each in a public AZ. The apps will stay in the private AZs, and they allow SSH connections from any of the bastion hosts.

1. Create an ELB

a. Listen on

i. HTTP Port 80

ii. HTTPS Port 443

1. Terminate SSL here

We create an ELB with its subnets are all the available public subnets. ELB will take traffic from anywhere on port 80, 443, and route to all the instances on port 8900. We treat connection to port 443 as normal http connection.

We pass these parameters to terraform to build our web servers.

app\_name = "terradata-app"

app\_primay = "apache2"

app\_secondary = "nginx"

app\_count = 3

2. Create 3 “t2.micro” EC2 instances

a. use ami-5e63d13e

i. uses "ubuntu" as user name for ssh b. in us-west-2 region

c. 2 of the instances must be configured as a web server using web server software of your choice:

i. apache

ii. nginx iii. node.js iv. tomcat v. tornado vi. etc

d. The 3rd instance must be configured as a web server using different software from the other 2 (e.g. if the first 2 use apache, this one uses nginx)

e. All 3 web servers must be configured as follows i. listen on port 8900 for HTTP

ii. Web access logs configured to write to /var/log/tdcustom/accesslogs/

iii. Web servers must not be running as root

iv. Web servers must return a "hello world!" type page

We currently use the 3 AMIs from each different region to create our web servers, and also the bastion hosts, for simplicity:

ami {

us-west-1 = "ami-3e21725e"

us-west-2 = "ami-5e63d13e"

us-east-1 = "ami-7dce6507"

}

We use the app\_count to create that number of web servers, and spread them evenly in all the private subnets. At this time only two of the web servers are available: apache2, nginx. If we set:

app\_primay = "apache2"

app\_secondary = "nginx"

Then only one server is bootstrapped with “nginx”, all other use “appache2”.

We can choose the instance type to provision the web servers, and also the bastions:

ec2\_instance\_type = "t2.micro"

ec2\_key\_name = "thuy-rsa"

**IMPORTANT**: the key pairs name must already exist in the region we want to provision.

3. Register all 3 EC2 web servers with ELB created in step 1

4. Networking:

a. Your laptop should be able to ssh into the instances

b. 141.206.246.10/32 should be able to ssh into the instances

c. The ELB should be forwarding web requests to hosts over port 8900

d. Port 8900 should only be accessible by the ELB

e. All other host ports not specified shouldn’t be accessible

Appropriate security groups are set such as:

Bastion hosts accept SSH from certain IPs.

ELB accept connections from anywhere on port 80, 443.

Web servers may accept SSH from bastions, HTTP on port 8900 from ELB.

No other in coming connections is allowed, except Outgoing.

Normally during a terraform run we can use local provisioner to obtain our public IP and use it as CDIR in a security group for the bastion hosts. Due to a bug in some versions of terraform, <https://github.com/hashicorp/terraform/issues/13034>, we use a wrapper to get our outgoing laptop public IP and pass it on to terraform. In future releases we may not need to do that.

If you have any issue running this code please do not hesitate to contact me:

[thuyqnguyen@yahoo.com](mailto:thuyqnguyen@yahoo.com)

(661)-406-4244

Additional instructions:

• Feel free to create VPC's/subnets/ssh keys/etc as required to complete your project

• Your submission should be runnable code either in chef / ansible/ bash/ python/

cloudformation/ etc.