**Technical Interview Summary Document**

**Ikenna Uzoije**

**Webserver for hosting static content**

To get this done, I first initialized the provider as AWS and passed in the region, access\_key and secret\_key variables to get terraform to have access to the aws account and be able to create and manage resources.

Next I provided a couple data sources using the “data” block which terraform uses to extract data like the latest instance ami, account Id, region, vpcs and availability zones. I made use of the local variables to hold the values of these data so I can reference them later on in the infrastructure.

I created an elastic IP next so that the webserver will have a constant IP address in the event that the instance hosting the server got unhealthy and had to be re-provisioned, users wouldn’t have to go to a new IP address to access the site.

Next was the creation of the webserver instance. This was done using the “aws\_instance” resource. The instance\_type, root block device, ebs block device, volume tags and instance tags were created using variables defined in a separate variables.tf file which makes it easier for an engineer to change these variables in one file instead of every point where it is being referenced in the code. I installed httpd and created an index.html file in the webserver using the user\_data function. I also made use of the remote-exec provisioner to run commands inside the instance during the creation time.

I also created an application and database server using the “for\_each” method to show how to provision more than one identical instances without having to copy and paste the same code more than once. Although the database and application EC2 instances do not have anything setup in it.

I created the security group the instances will use. This security group regulates traffic that gets into each instance and regulates what ports are allowed to be open. I opened ports 80, 443, 8080 and 22 to allow http, ssh traffic into the instance.

**What I learned**

I learned how to use dynamic blocks to open up multiple ports in a security group resource. I also learned how to use a lifecycle policy to reduce the downtime between provisioning new instances. The lifecycle policy I used is the “create\_before\_destroy” policy that creates a new instance before destroying an exisiting one.

**Blockers/Challenges**

At first I had to hardcode the instance AMI for terraform to provision the EC2 instance with but I had an issue when I changed the region because each region has a different AMI for the same instance type. I noticed that each instance type (e.g. amazon linux) has the same owner and similar version number. To get around that I created a data block that hardcodes the owner ID and the version number to get the latest version of the amazon linux AMI regardless of which region we are provisioning the instance in.