Contents

[Data Flow 1: User access to HTTP/HTTPS Website in private subnet 1](#_Toc130192396)

[Data Flow 2: Internal admin access to private subnet resources 2](#_Toc130192397)

# The Hello Mars Cloud Journey

1. Initial goal was to create a Hello World application in AWS using Terraform, add security stuff to it and ….
2. What I learned
   1. Simple 2 tier app with public subnet EC2 instances and Private subnet database
   2. 3 tier app with private subnet EC2 instances and private subnet database and public subnet bastion host for internal users to remote log (SSH) into private instances, whether EC2 or DB.

Week 1: Goal Setting

Week 2: Simple 2 tier app with public subnet EC2 instances and Private subnet database

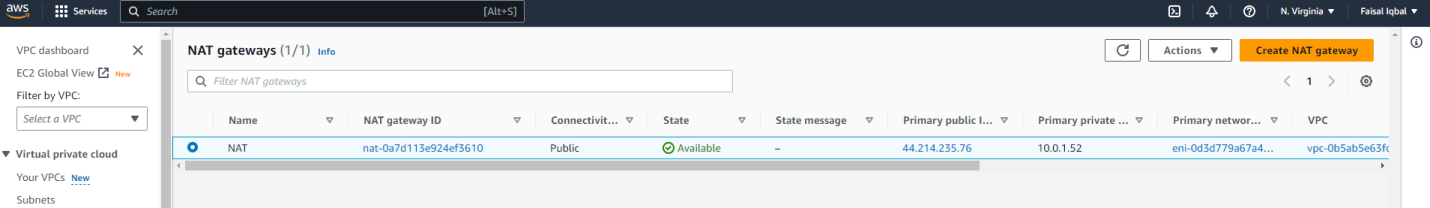
Week 3: 3 tier app with private subnet EC2 instances and private subnet database and public subnet bastion host for internal users to remote log (SSH) into private instances, whether EC2 or DB.

Week 4: Design review for site to site VPN and authentication via federation for corporate identities

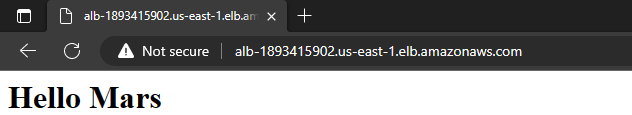
## Data Flow 1: User access to HTTP/HTTPS Website in private subnet

**Learning Points:** Public-Private Setup using NAT Gateway and ALB

1. Created a VPC
2. Created 6 subnets – 2 public and 4 private
3. Create IGW (internet access for public subnets)
   1. Attached to VPC
   2. Created routing for internet access for public subnets
4. Created a NAT Gateway hosted in public subnet so connectivity is public
   1. Created a



1. Added routes
   1. Add 0.0.0.0/24 to IGW
   2. Associate pub sub to IGW RT
   3. 0.0.0.0/24 to NATGW
   4. Associate pvt sub to NGWRT
2. Create EC2 Instances
   1. Application resides in replicated EC2 instances in private subnets
   2. Security Group for SSH access (1 sec group for both instances)
   3. Bastion host resides in public subnet for user to (See Later section for improvements)
   4. Edit security group rules for ALB and allow port 80 from ALB SG
3. Create the application load balancer
   1. Create a application lb in public subnets across 2 AZs
   2. Create a sec group for port 80
   3. Create target group / fix routing
   4. Register targets in private subnets

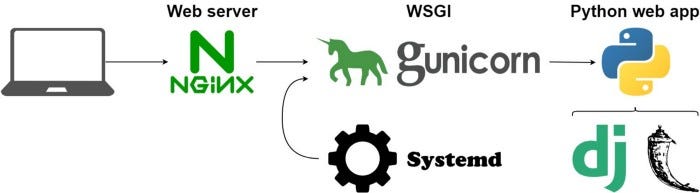


Related Files



## 

### Deploying the app



**Highlights and joys of deploying Python Flask application**

1. In the terminal, Flask is running in development mode, which is not production grade
2. Application is running in the foreground, and if we hit Ctrl + C, we will bring down the app
   1. Remediate it with
      1. Web server ( Nginx)
      2. A Web Server Gateway Interface ( Gunicorn),
      3. Demonize the execution so that the app will be running in the background
3. Database secrets management
   1. Currently, it is hardcoded into TF files
   2. TFVARS file is an options but it’s best to use any of the following with the
      1. KeyVault
      2. KMS
      3. Hashicorp Vault
      4. AWS Se

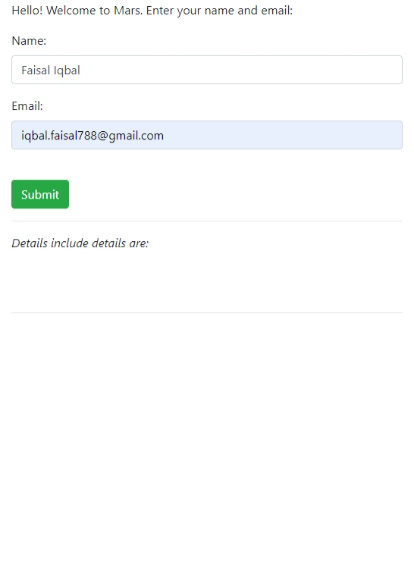
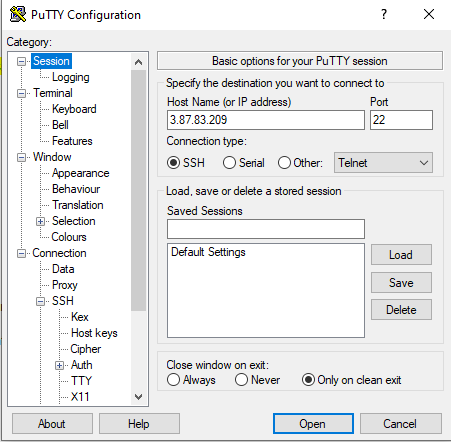
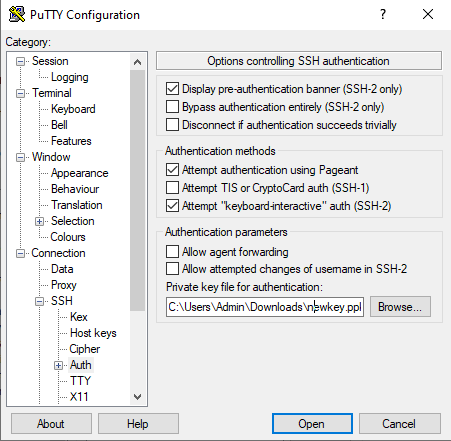


Image: Final Application

## Data Flow 2: Internal admin access to private subnet resources

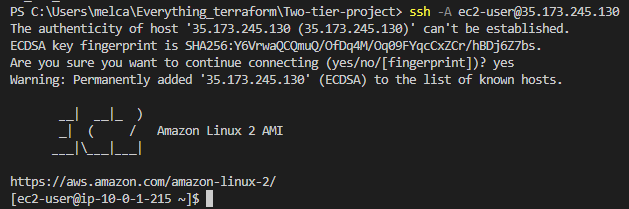
**Learning point:** How to set up a Bastion Host for access to private subnet resources





PPK file generated for the private key file to communicate securely with another party without having the corresponding public key.

Access the linux AMI

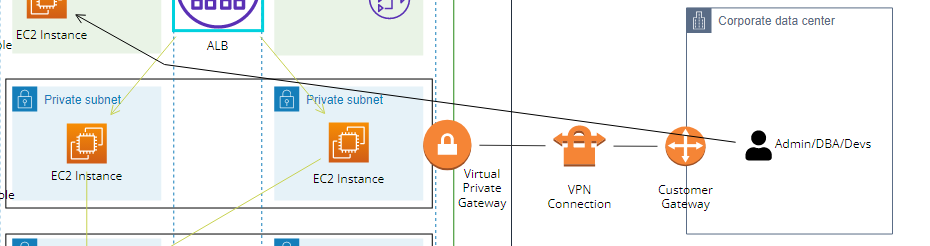


This is how to enter a database in the private subnet

mysql -h db-instance.cm4bp7mgz27s.us-east-1.rds.amazonaws.com -P 3306 -u admin -p

This design was later deprecated and replaced with VPN

Try using the VPN option:



1. Create a

Create a VPC peering