**SENECA**

**Final Project:**

**Two-Tier web application automation with Terraform**

Cloud Architecture and Administration (CAA)

Cloud Automation and Control Systems (ACS730)

**Professor: Leo Lu**

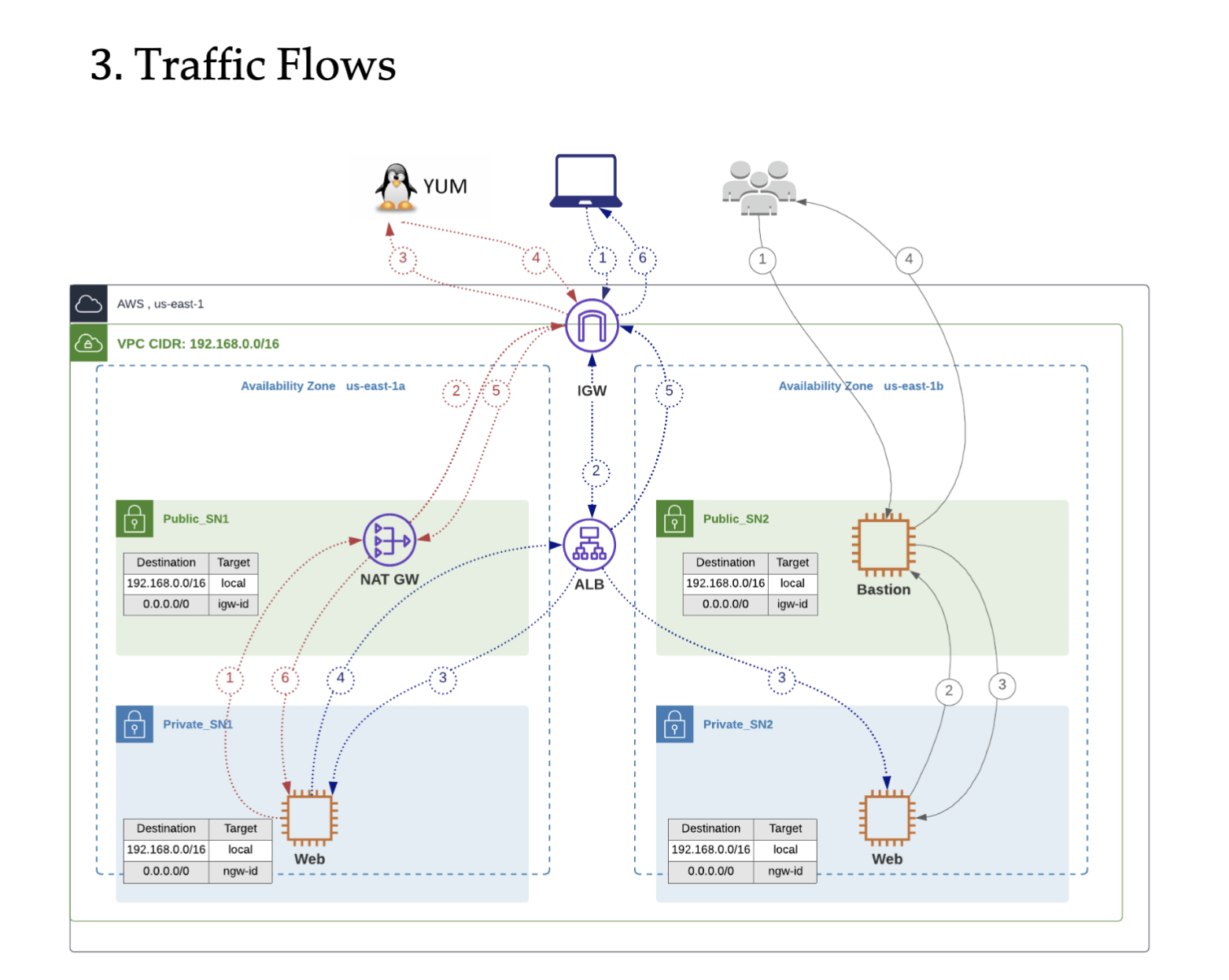
**Group 17**

**Section: NBB**

|  |  |
| --- | --- |
| **Team Members:** | **GitHub User** |
| Dirghayu Joshi | dirghayu101 |
| Mahesh Chalise | MaheshChalise |
| Malika Adhikari | madhikary |
| Parinaya Baniya | BaniyaParinya |
| Rujal Maharjan | rmaharjan11 |

Date: 2024-04-12

**Explanation of traffic flow**



The red network flow shows that the web Servers needs to install or update any packages or dependencies the request is sent to yum Linux package management application, first the request is sent through NAT Gateway which is transferred to Internet Gateway only then the YUM receives the request, then the Yum respond and sends the packages Through the Internet Gateway which is then forwarded to the Webserver through NAT Gateway, after which the webserver installs or updates the received packages. This workflow happens during installation or updating services or dependencies in the Web Server. This flow is triggered any time yum commands are run in the Web Servers.

The blue network flow shows that the user sends the request to webserver, the request goes through Internet Gateway that passes the request to Application Load Balance then Application Load Balancer sends the request to one of the Web Server depending upon the policies set, then the web Server responds which is transmitted to the User through Application Load Balance and Internet Gateway respectively. This workflow happens when a user accessed the webservers through web browsers typically http or https request are received.

The gray network flow shows that the cloud developer connects to the Web Server through bastion, where bastion acts as the middleman that helps developer connect to private webserver. The request that has been sent to webserver gets through bastion host after which the Web Server responds to the cloud developer about the state through Bastion Host.

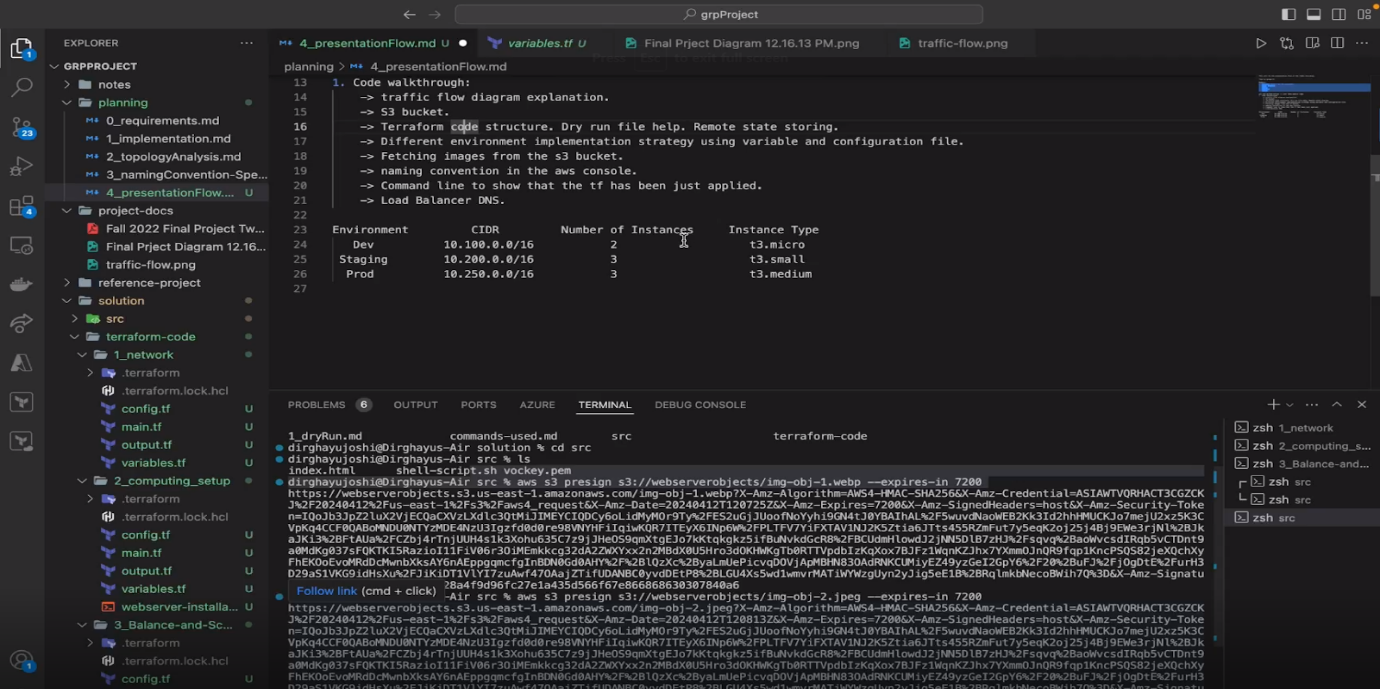
**Challenges faced and mitigating those challenges**

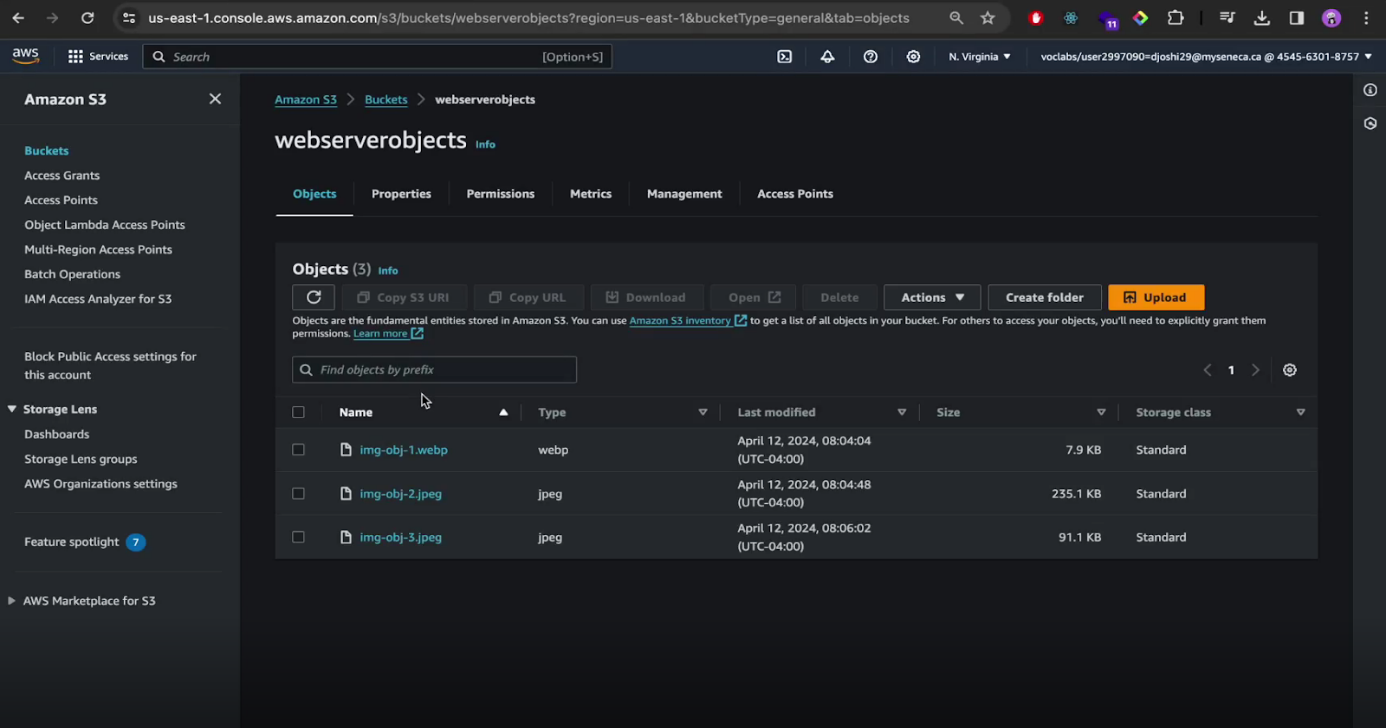
Initially, creating infrastructure for many settings sounded difficult. We streamlined the process by dividing requirements into manageable tasks and concentrating on one environment at a time. This step-by-step method enabled us to better comprehend the specs and design an effective scalable solution. Furthermore, writing Terraform code posed issues in guaranteeing consistency between environments. We tackled this by looking at best practices, developing reusable modules, and reducing code duplication.

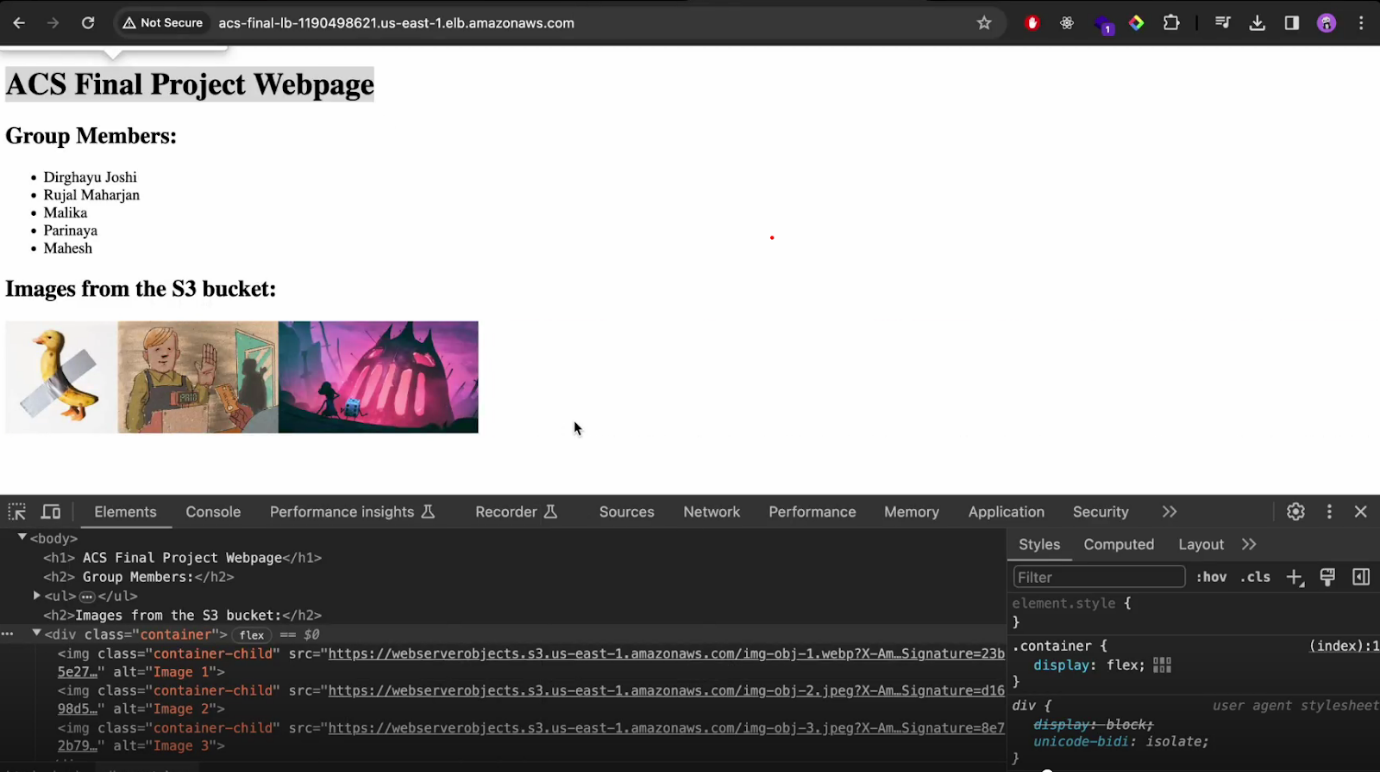
Another problem was coordinating collaboration through GitHub, especially in terms of assuring regular commits and integrating security scans. To address this, we established clear communication routes within the team and scheduled regular checkpoints to assess progress. Setting up GitHub Actions to run security scans on each push and pull request offered ongoing feedback on code quality and security compliance.

Throughout the deployment process, debugging numerous difficulties, such as resource conflicts and configuration mistakes, became critical. We took a systematic approach, beginning with determining the fundamental cause of the problem and then applying specific solutions. Using logging and monitoring tools allowed us to gather insights and track down issues more rapidly. By addressing these problems collectively and proactively, we successfully installed the infrastructure and met the assignment's objectives.

**Screenshot of the webservers serving traffic behind the Load Balancer**

****

****

****