**CLOUD COMPUTING**



**Assignment 2**

**Advanced Terraform & Nginx Multi-Tier Architecture**

**Submitted to:** Waqas Saleem

**Submitted by:** Aimen Hafeez

**Registration #:** 2023-BSE-002

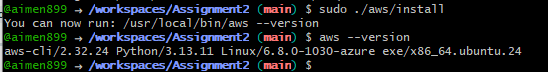
**Deadline:** 30th Dec 2025

**PART 1 – Infrastructure Setup:**

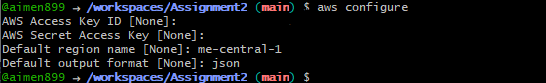
1. Install Terraform in Codespace



1. Install aws

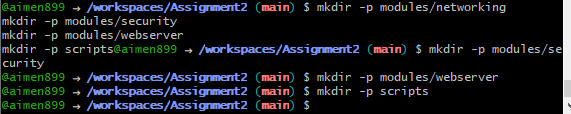


1. Configure AWS Credentials

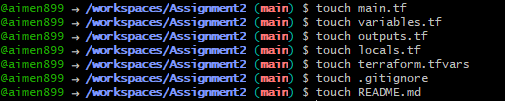


1. Create Complete Folder Structure

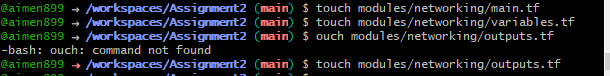
Create all directories



Create all root level files



Create networking module files



Create security module files



Create webserver module files



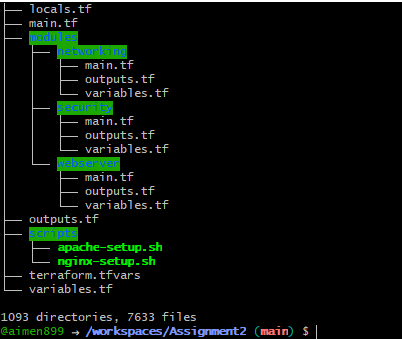
Create script files



Make scripts executable

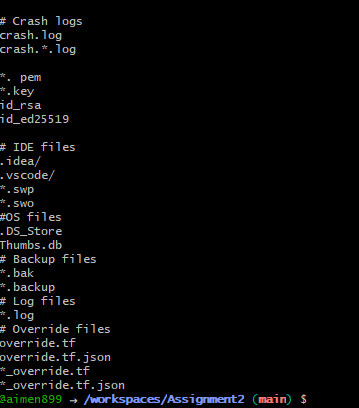


Verify structure with tree command

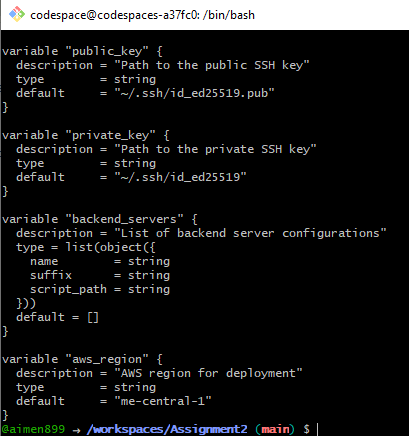


1. Create File Contents

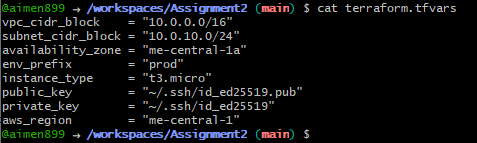
Create . gitignore File



1. Create variables.tf

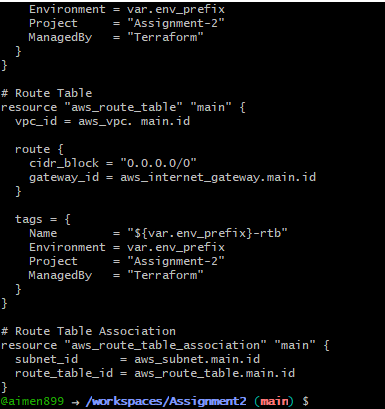


1. Create terraform.tfvars

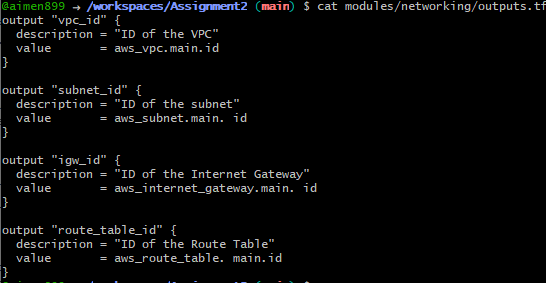


1. Create Networking Module

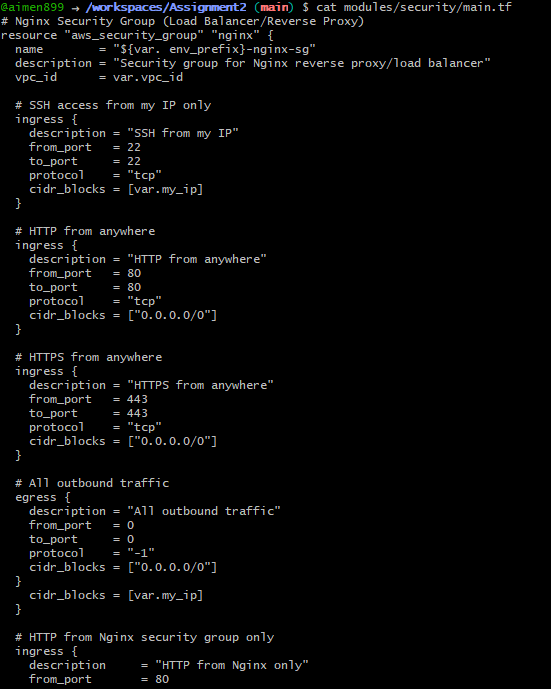
Main.tf



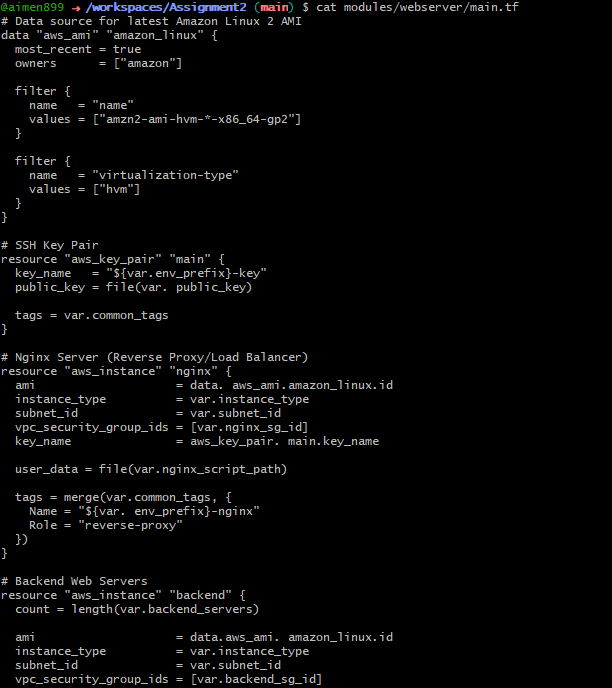
Outputs.tf



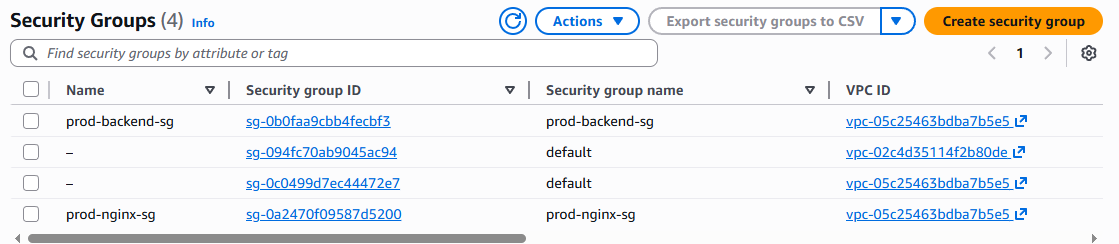
1. Create Security Module



1. Create Webserver Module



Security groups



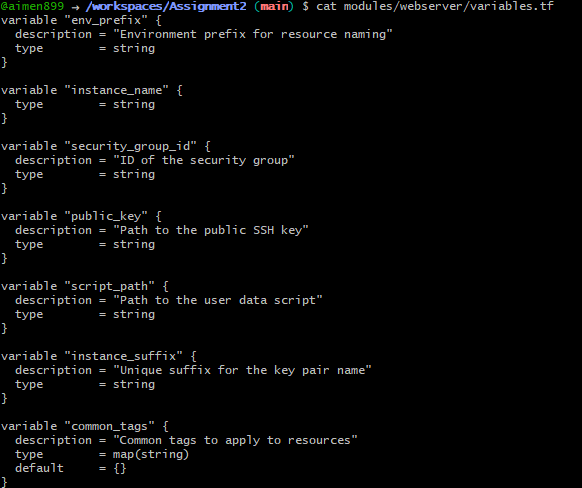
1. Create locals.tf



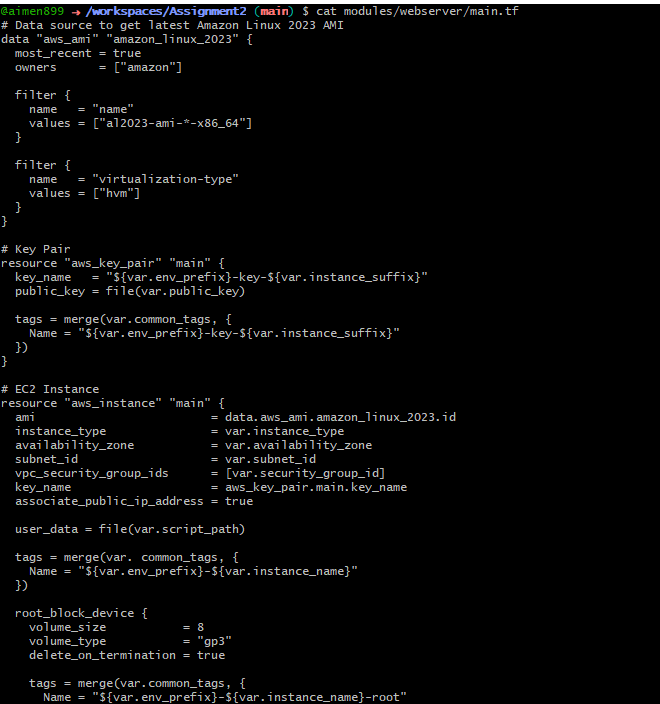
**Part 2: Webserver Module**

1. Create Webserver Module

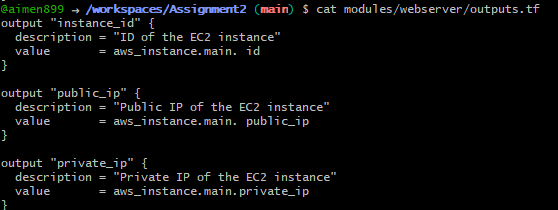
modules/webserver/variables.tf



modules/webserver/ main.tf



modules/webserver/ outputs.tf

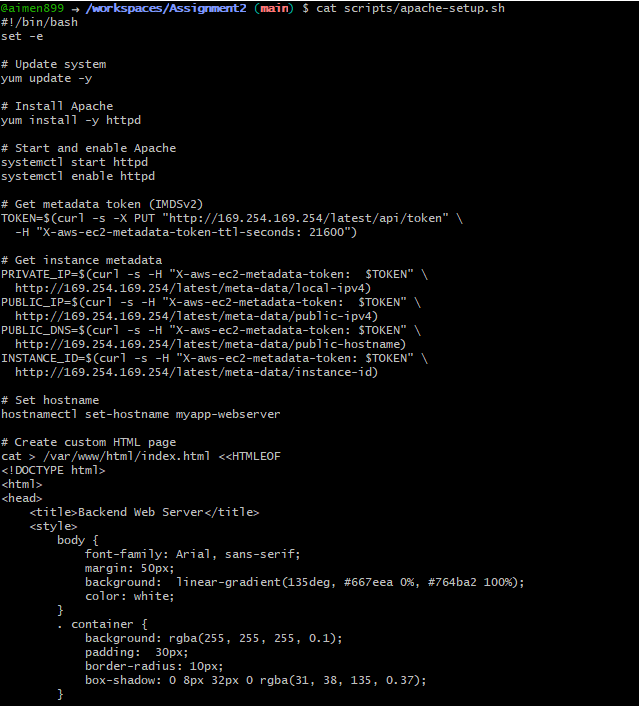


1. Create Root main.tf

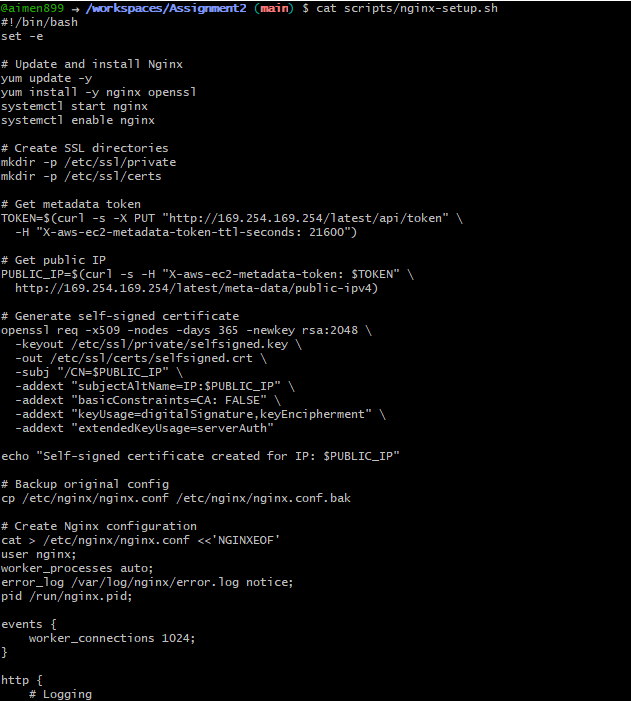


**Part 3: Server Configuration Scripts**

1. Create Apache Backend Script



1. Create Nginx Setup Script



**Part 4: Infrastructure Deployment**

1. Generate SSH Key Pair



1. Create outputs.tf



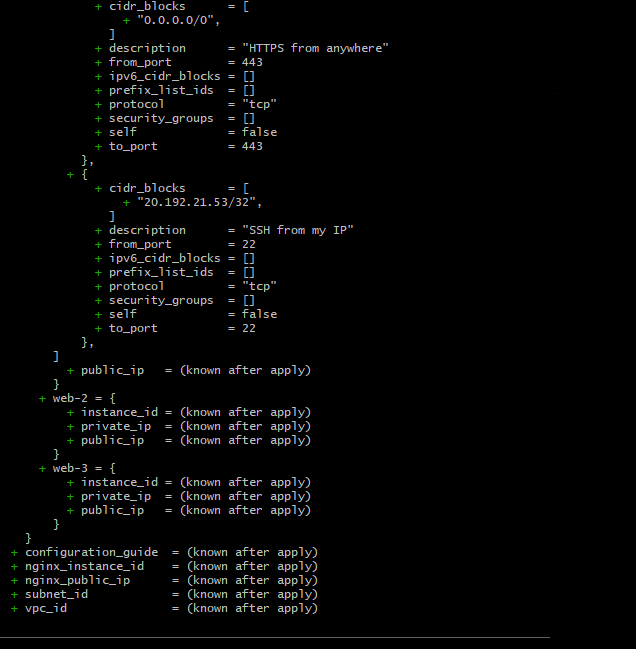
1. Initialize Terraform



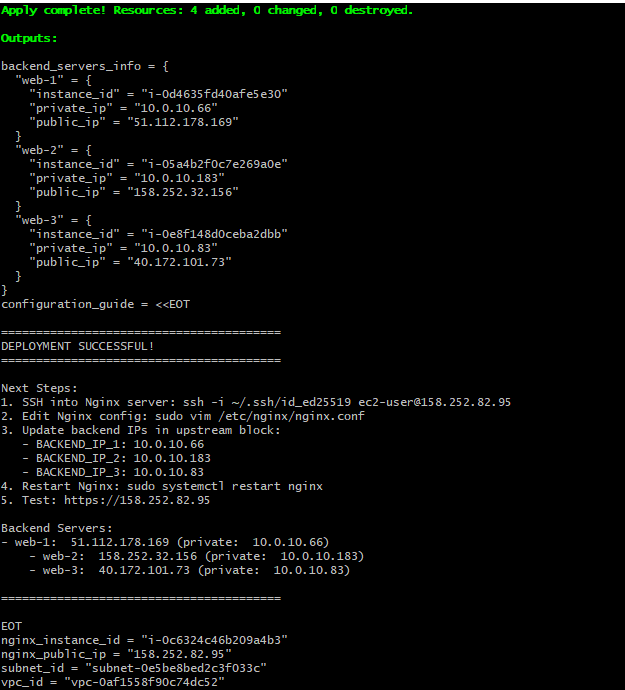
1. Validate Terraform Configuration



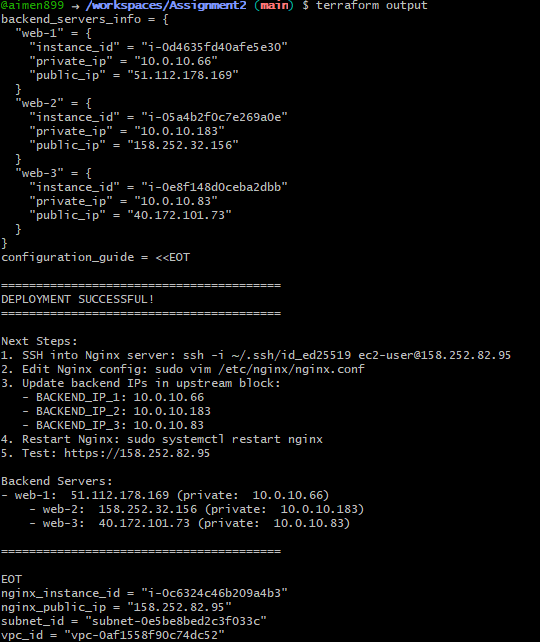
1. Plan Terraform Deployment



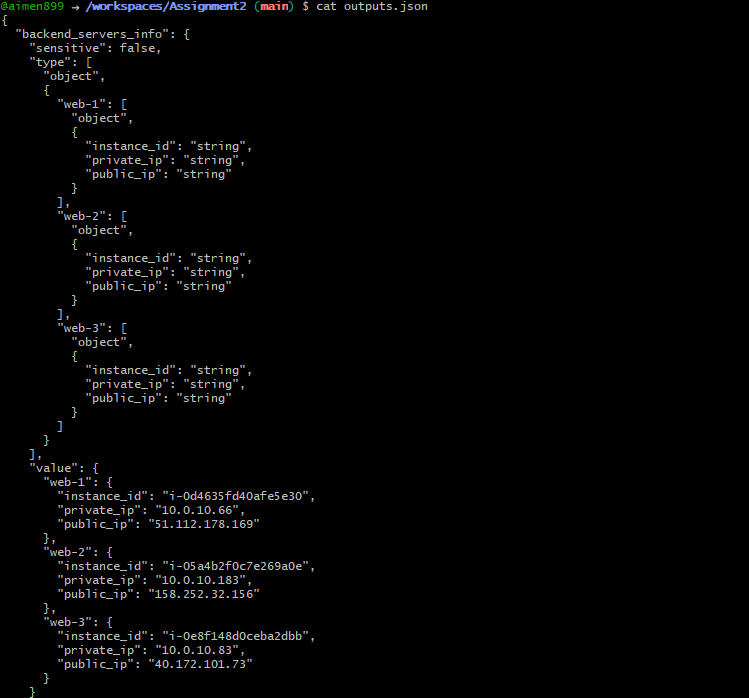
1. Apply Terraform Configuration



1. Terraform output:

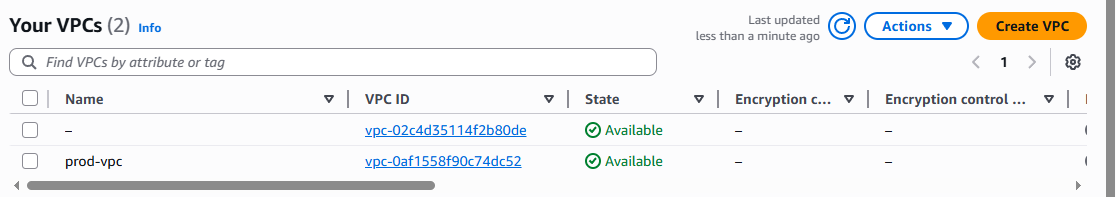


1. Output.json

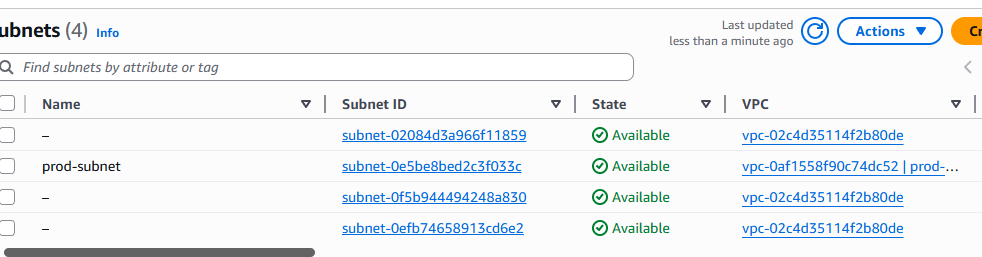


1. Verify Resources in AWS Console

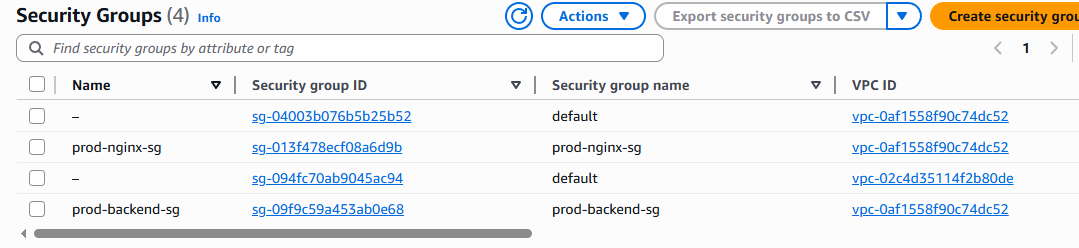
VPC:



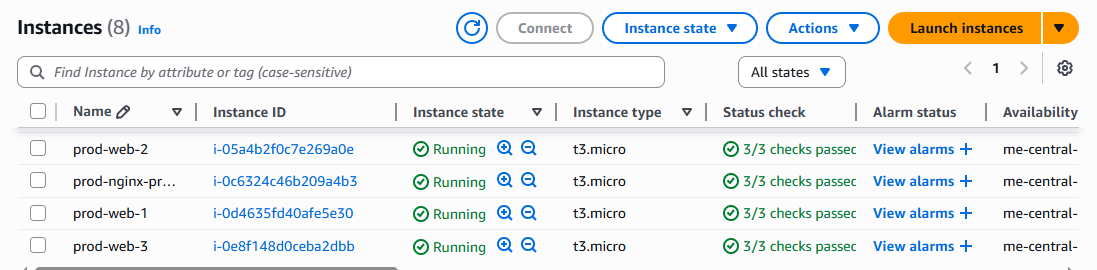
Subnet:



Security Groups:

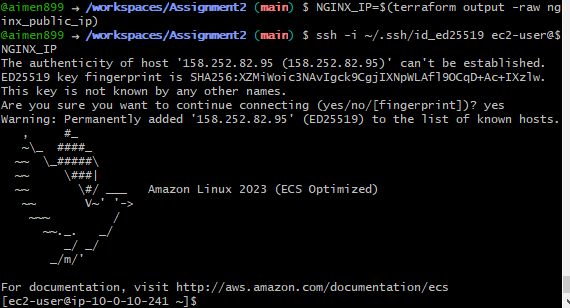


Instances:



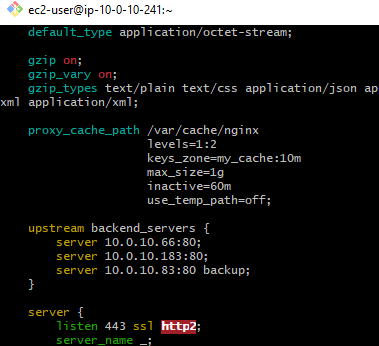
**Part 5: Nginx Configuration & Testing**

1. SSH into Nginx Server

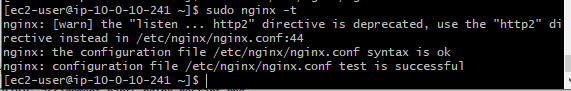


1. Update Nginx Configuration

On the Nginx server:



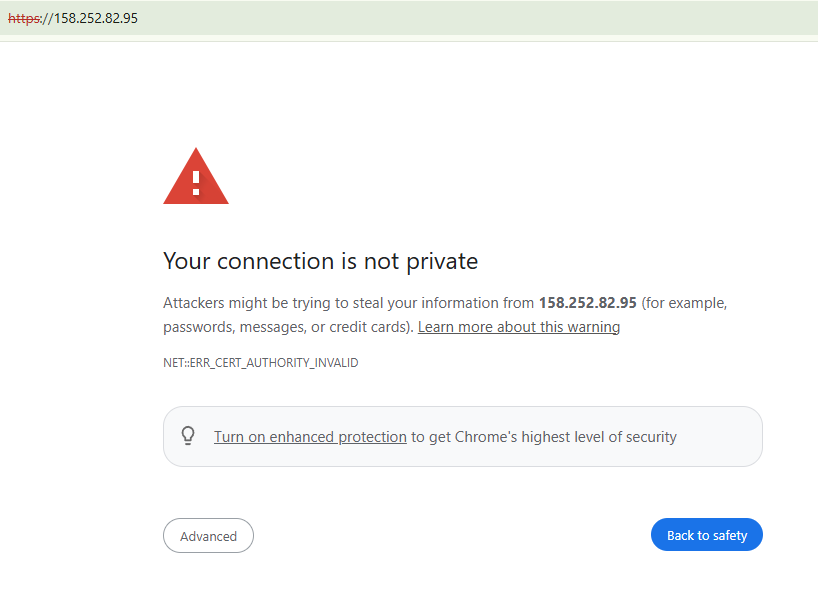
1. Test Nginx configuration



1. Nginx Restart:



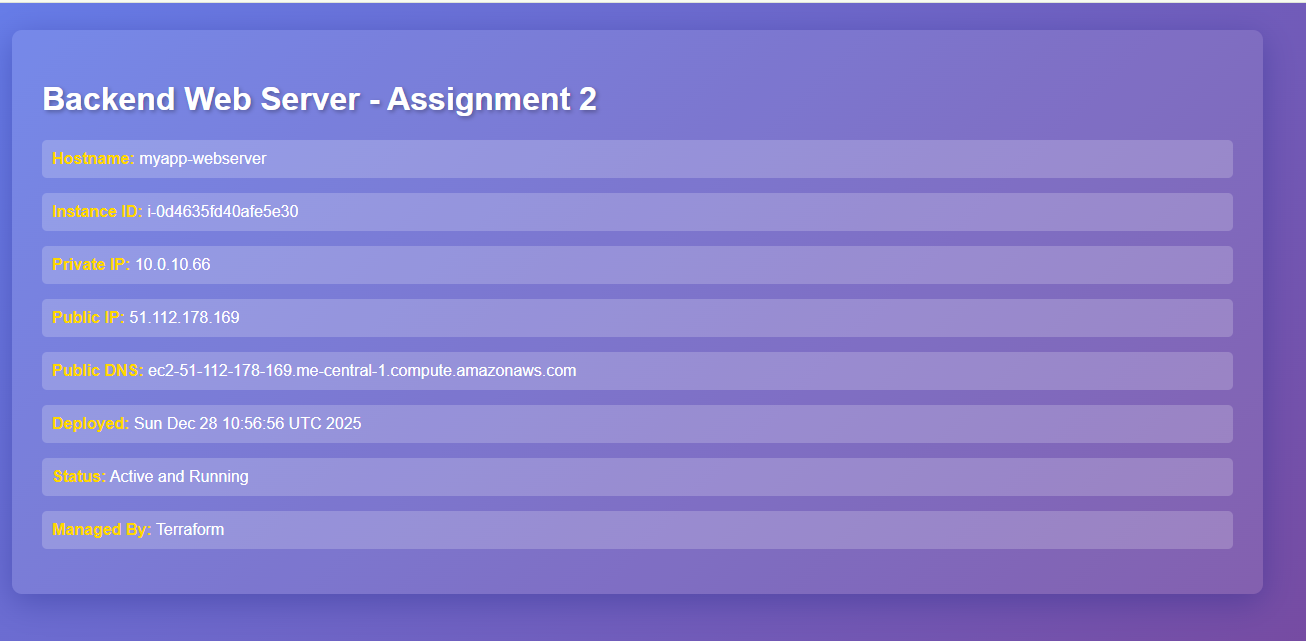
1. Test Load Balancing



Reload the page multiple times

Observe traffic alternating between web-1 and web-2

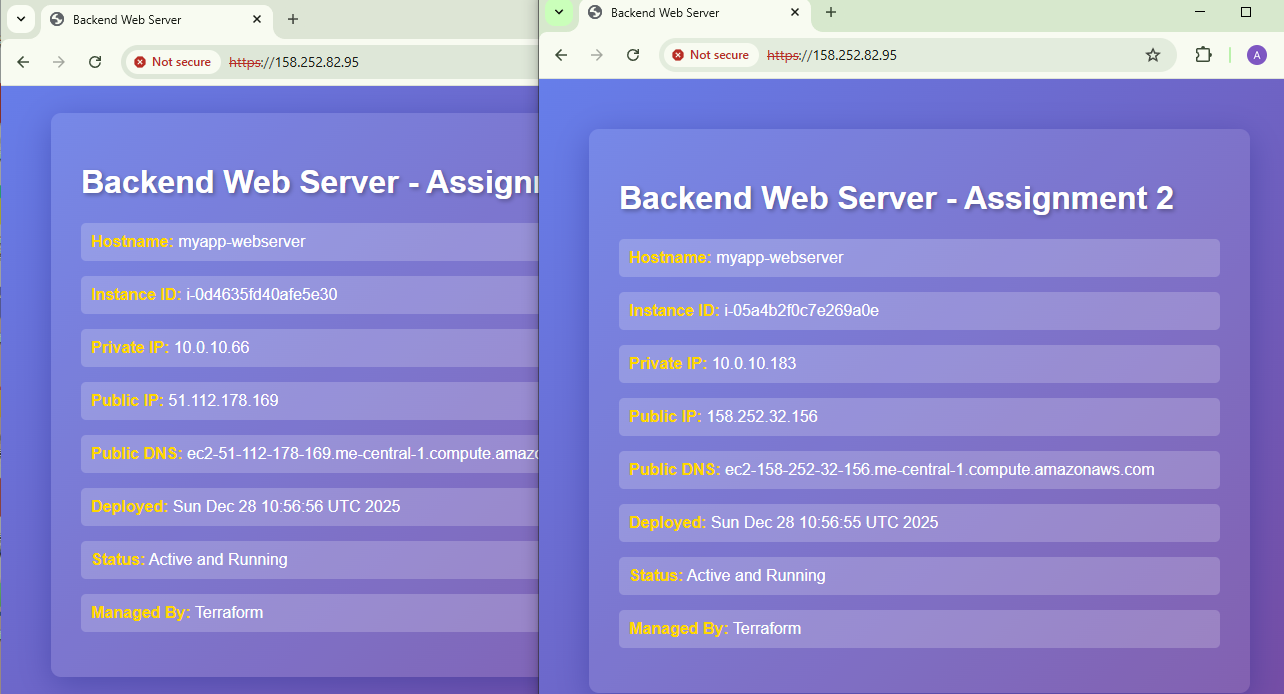
Web-1:



Web-2:

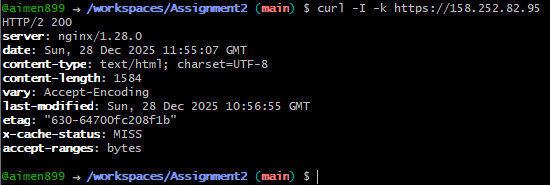


Load Balancing Demo:

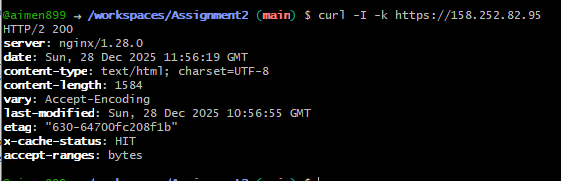


1. Test Cache Functionality

MISS on first request



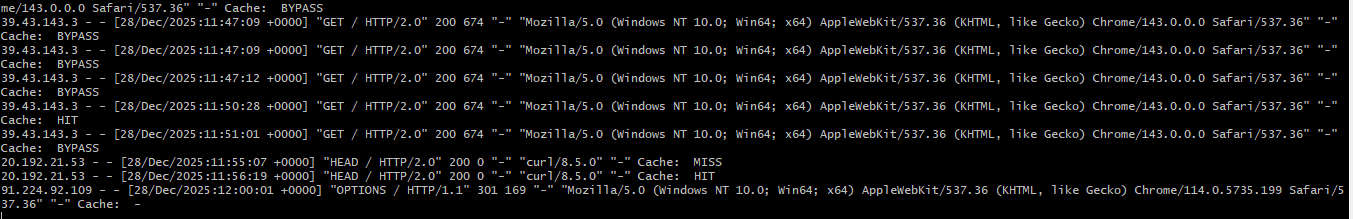
HIT on subsequent request



Cache Directory:

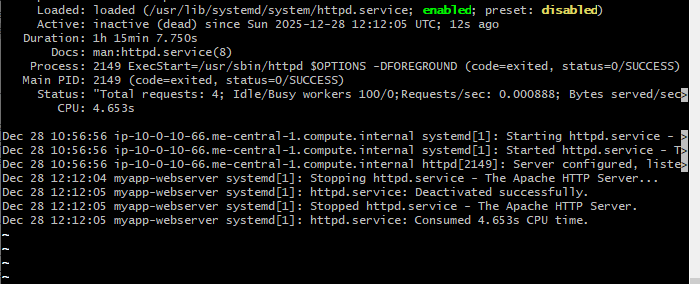
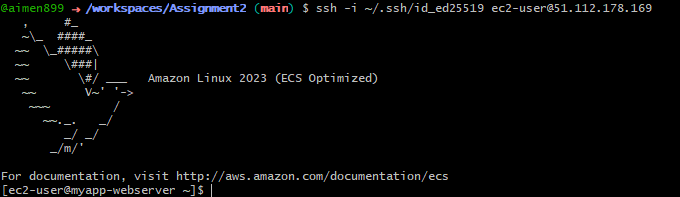


Access Log Cache:



1. Test High Availability

SSH into web-1 and stop Apache

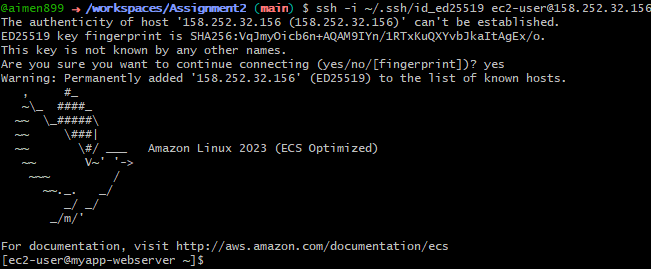


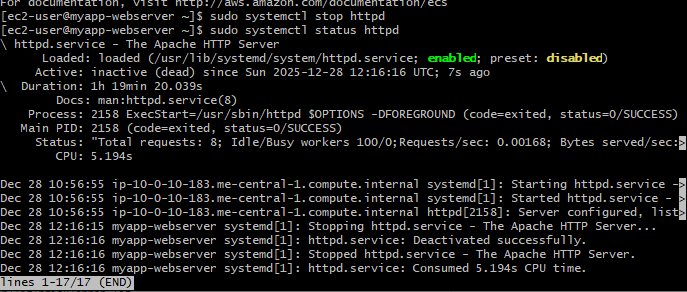
Test - should only show web-2 now

On subsequent reloads, only web-2 server is shown:



SSH into web-2 and stop Apache



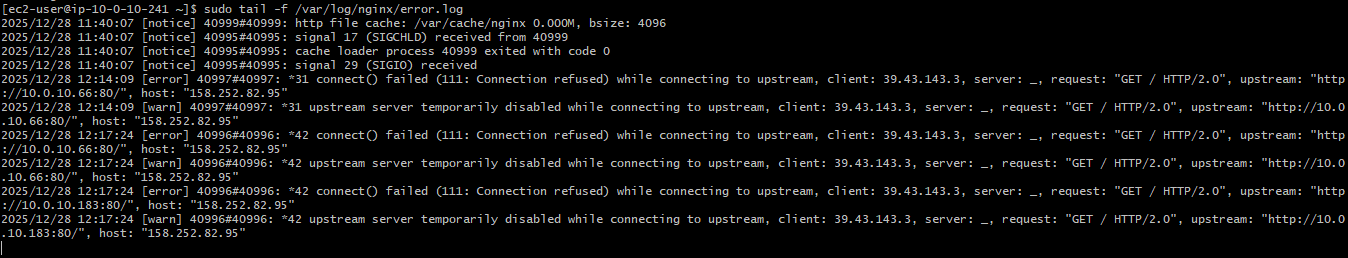


Test - should now show web-3 (backup activated)

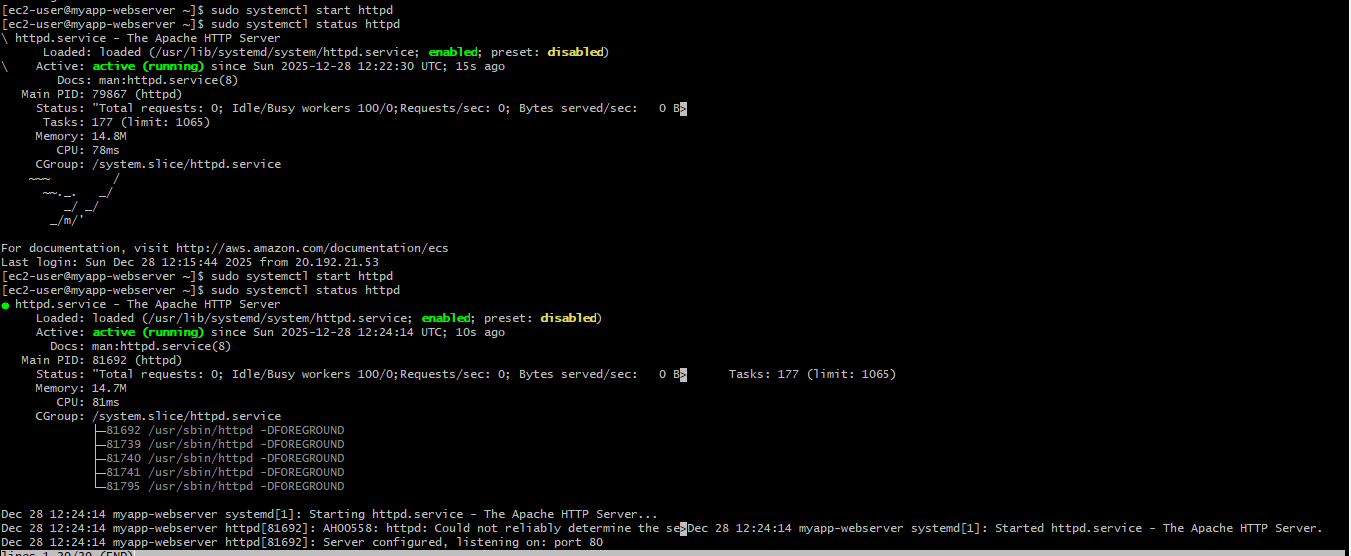
Reload browser



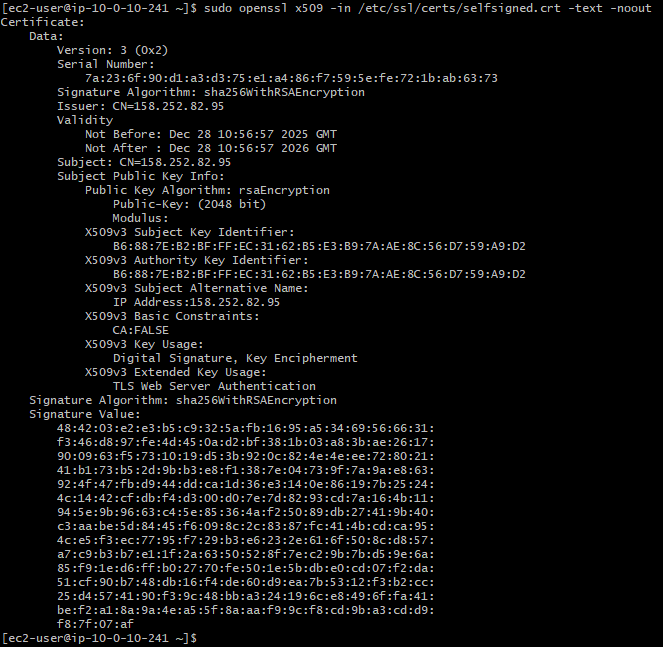
Check Nginx error logs



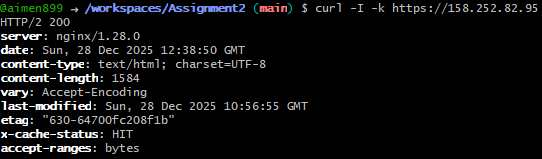
Restart services on web-1 & web-2:



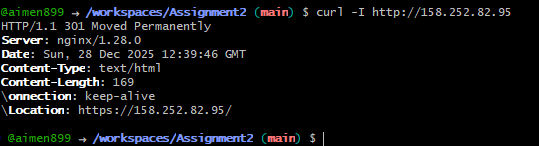
1. Security & Performance Analysis



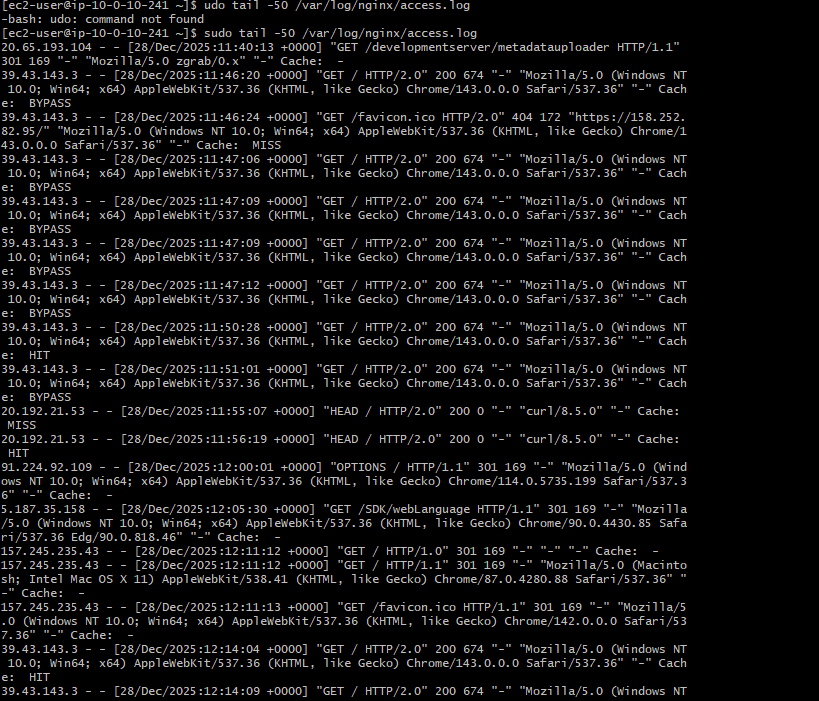
Check security headers



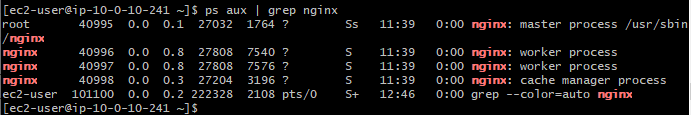
Test HTTP redirect



View logs



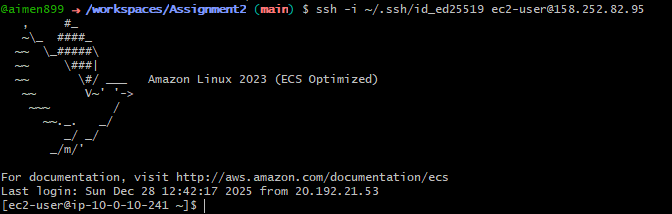
Check Nginx worker processes



**Bonus Tasks**

Bonus 1: Custom Error Pages

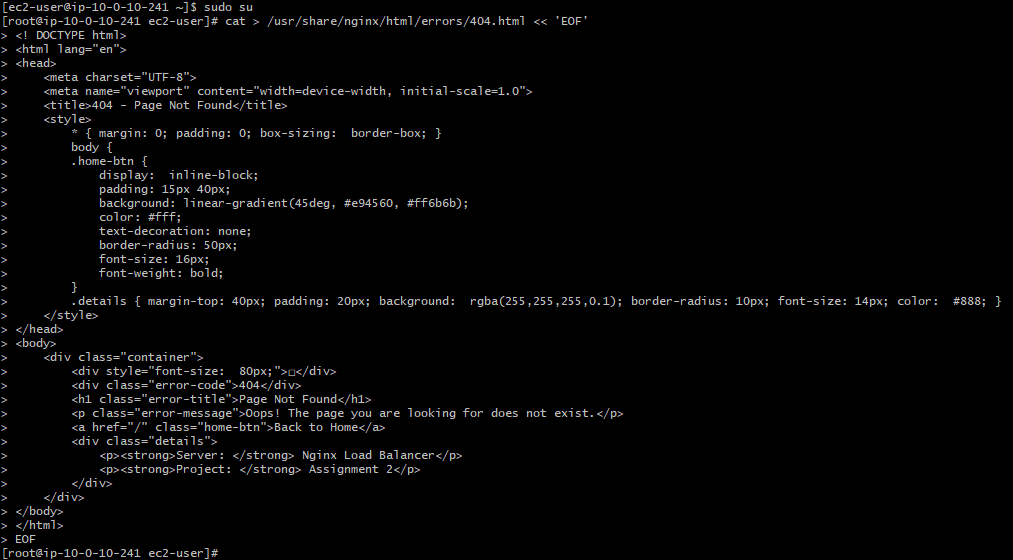
1. SSH into Nginx Server



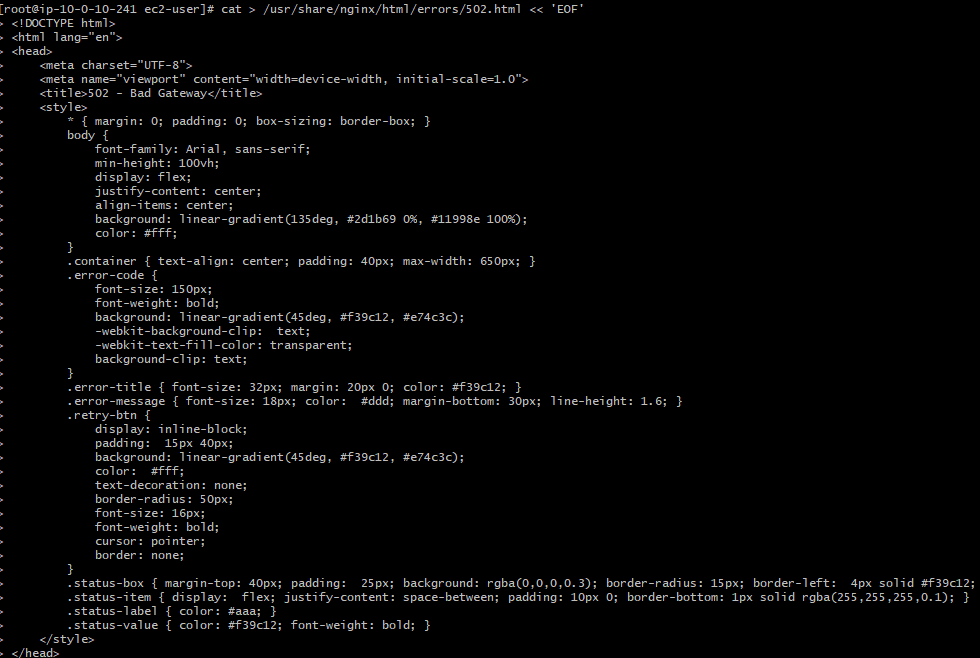
1. Create Error Pages Directory



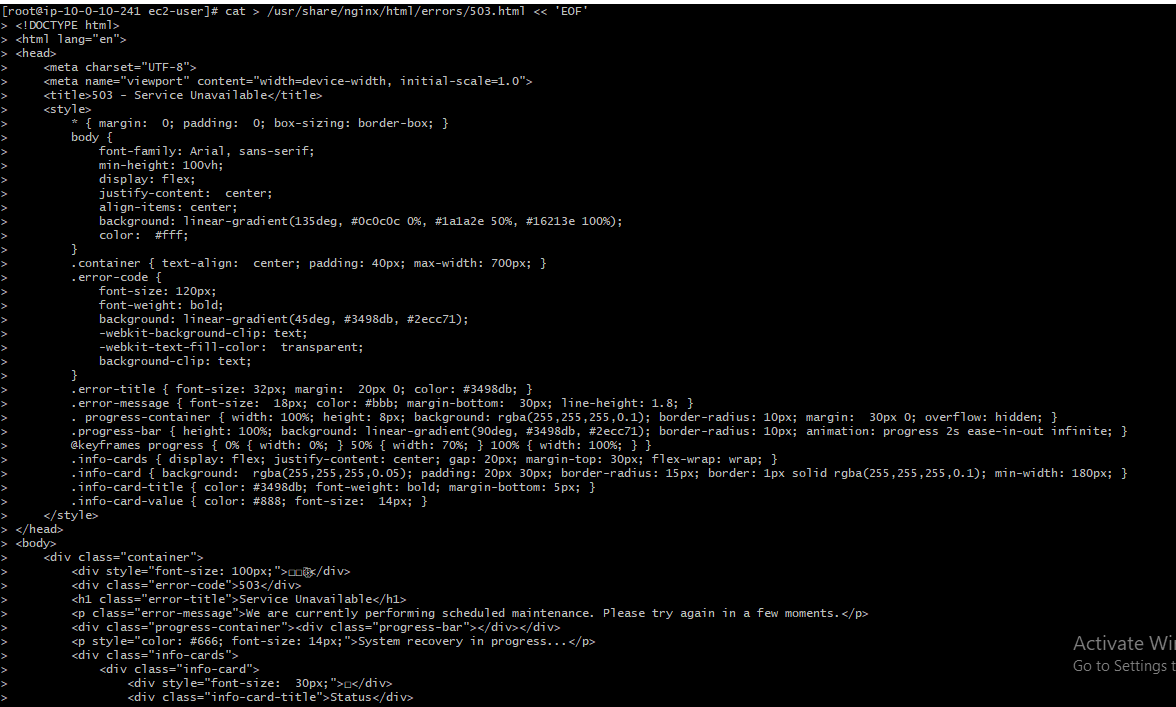
1. Create Custom 404 Error Page



1. Create Custom 502 Error Page - Bad Gateway Error Page



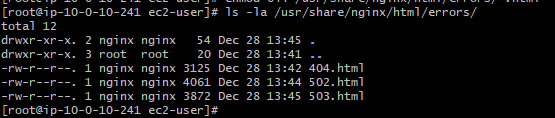
1. Create 503 error Page - Service Unavailable Error Page



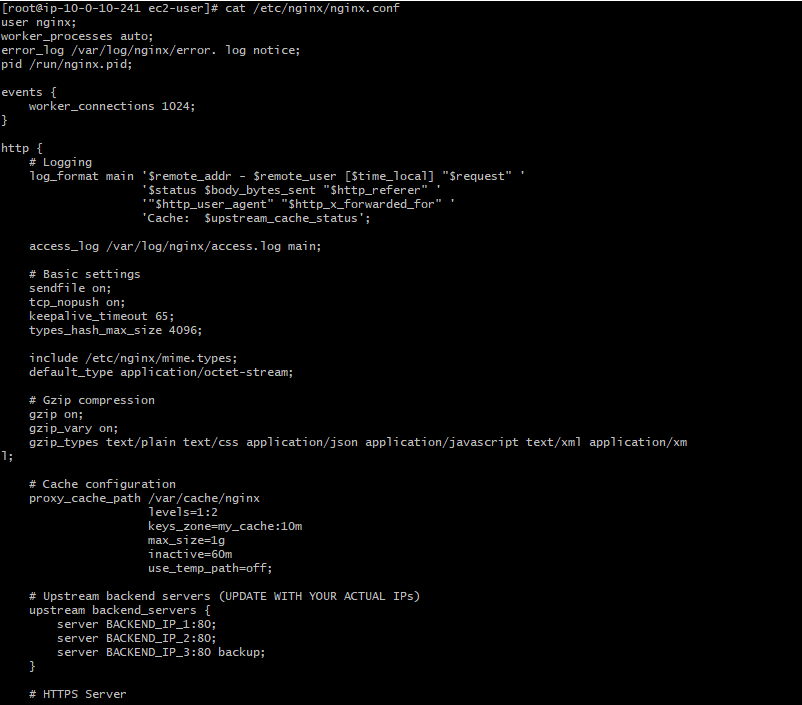
1. Set Proper Permissions



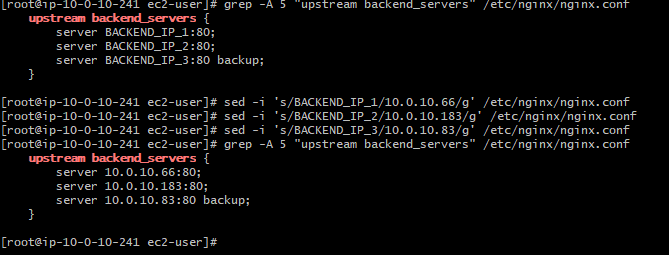
1. Verify files permissions



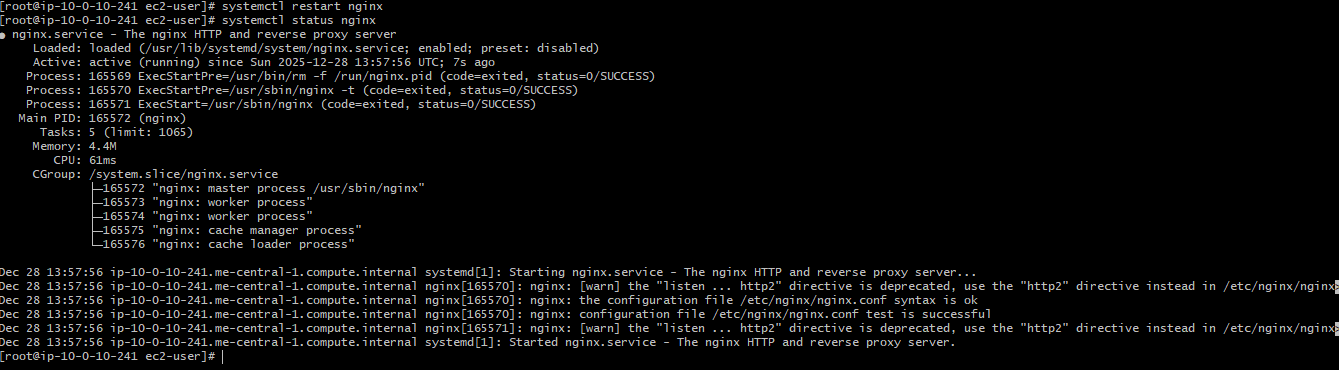
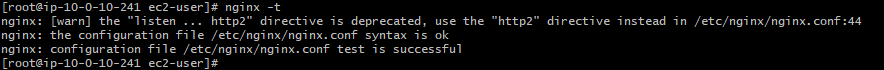
1. Update Nginx Configuration for Custom Error Pages



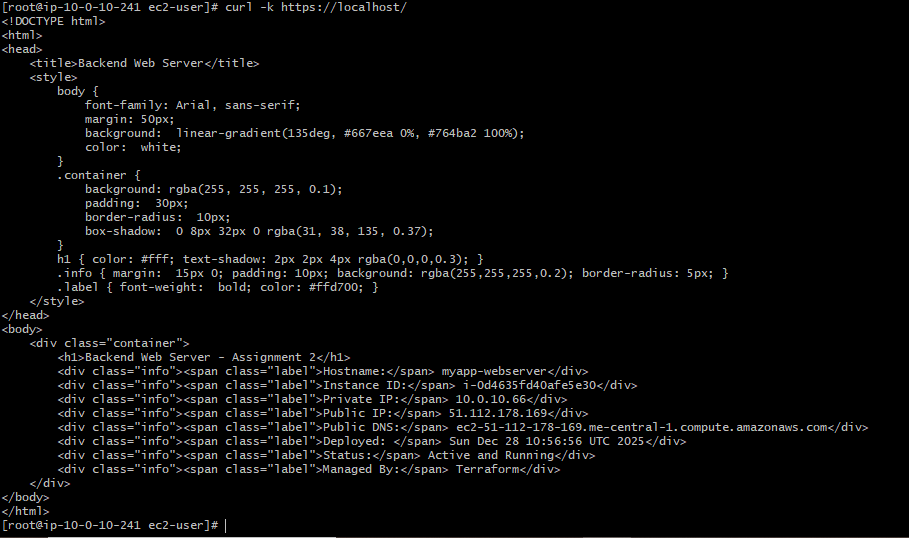
1. Update Backend IPs in Configuration



1. Test and Restart Nginx



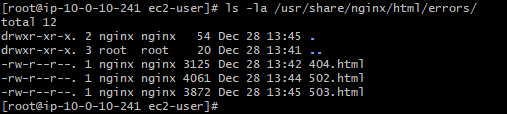
1. Verify Everything is Working



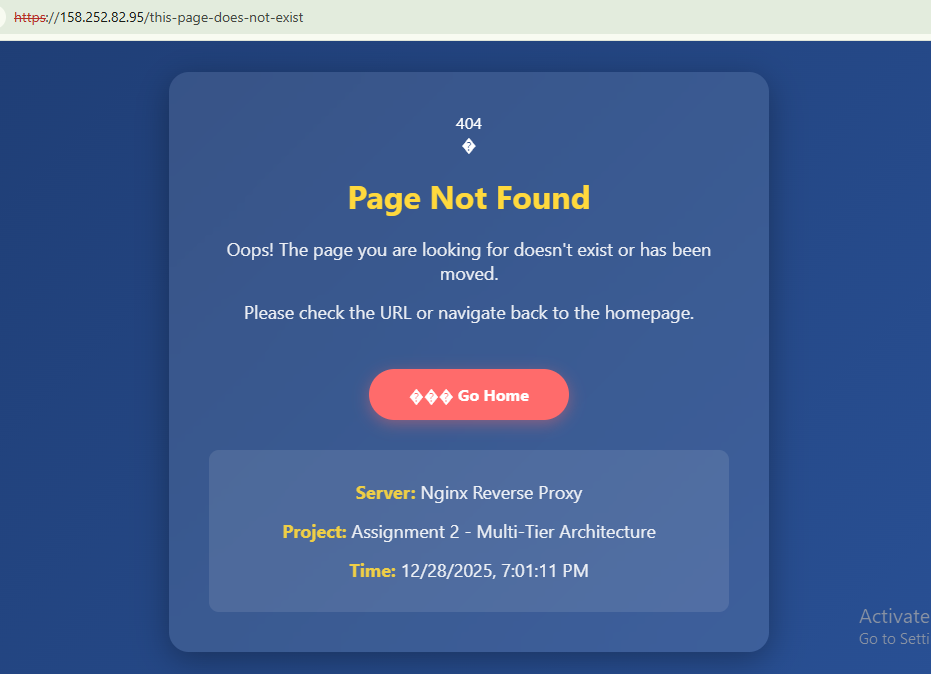
Test 404 error page



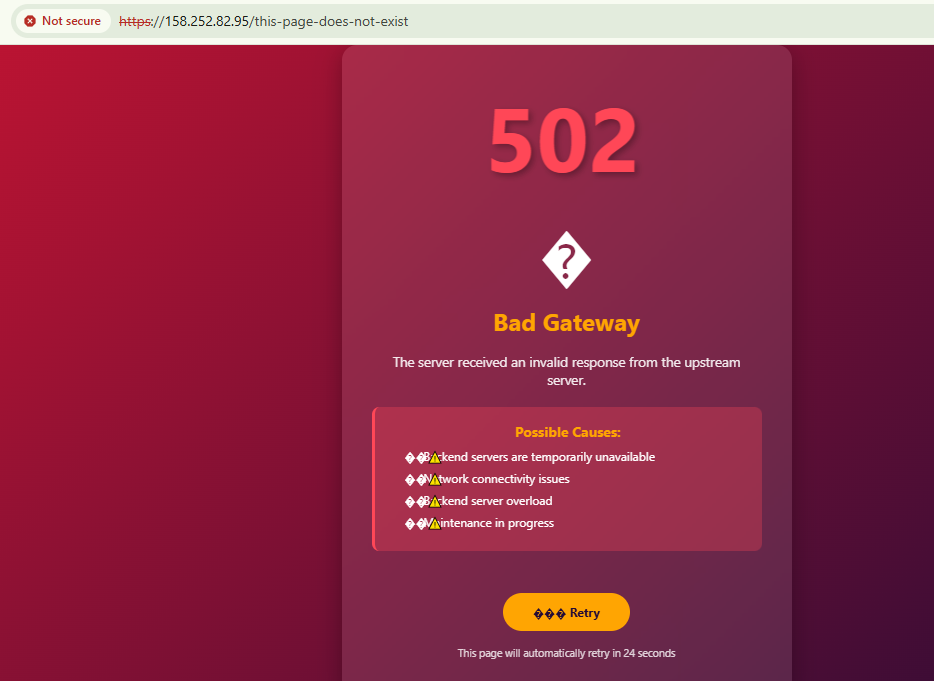
Check error pages directory



1. Test 404 Page



1. Test 502 Page



1. Restart Backend Servers



**Bonus 2: Implement Rate Limiting**

1. SSH into Nginx Server and Become Root

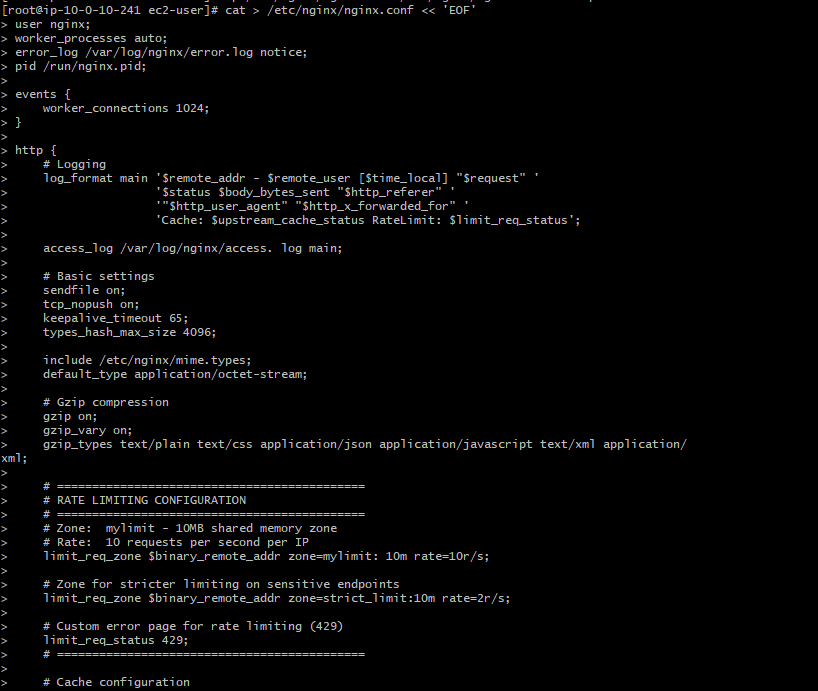


1. Update Nginx Configuration with Rate Limiting

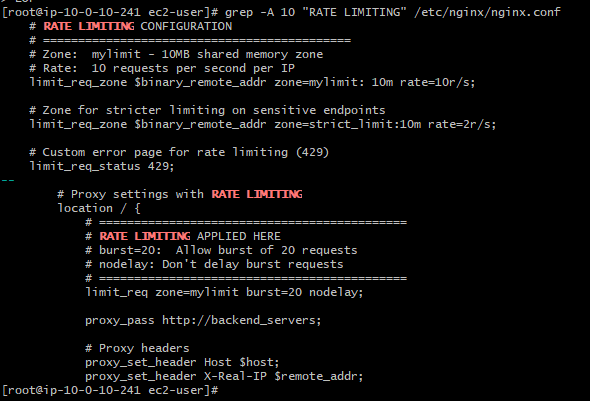
Backup current config



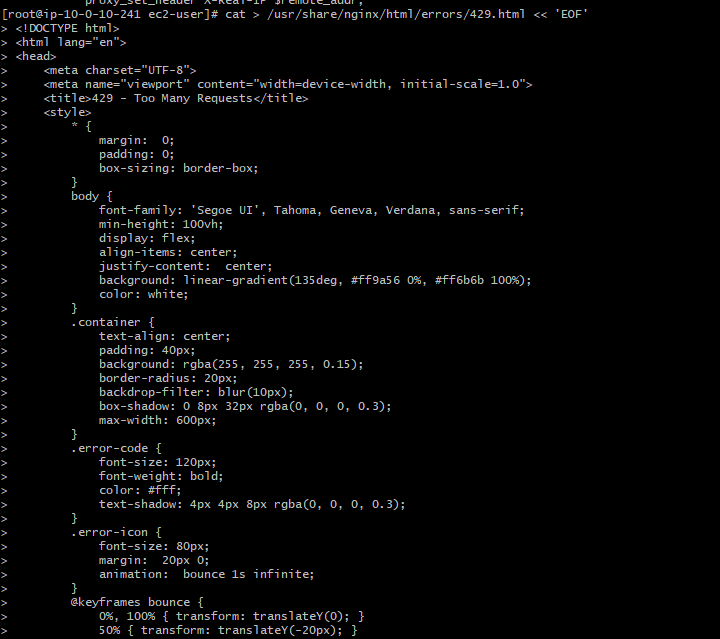
1. Create updated Nginx configuration with rate limiting



Display the rate limiting configuration



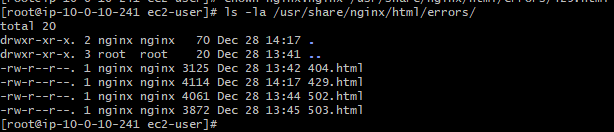
1. Create Custom 429 Error Page



Set permissions

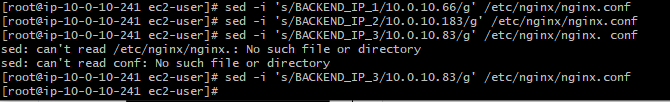


Verify file

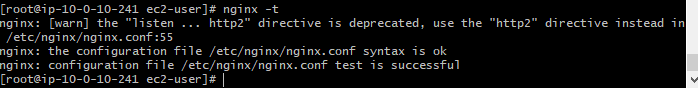


1. Update Backend IPs and Test Configuration

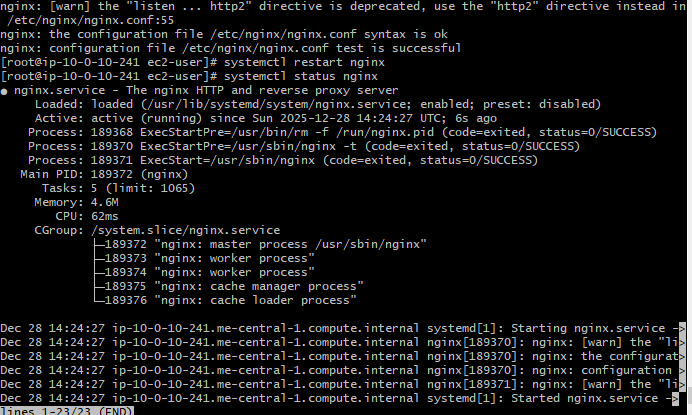
Replace with your actual backend private IPs



Test Nginx configuration

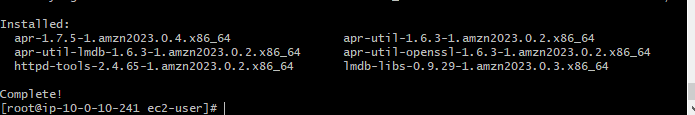


Restart Nginx

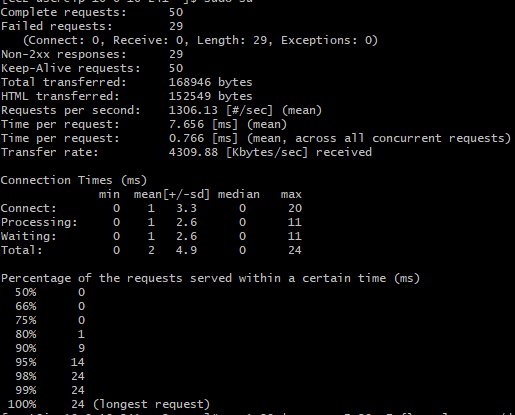


1. Test Rate Limiting with Rapid Requests

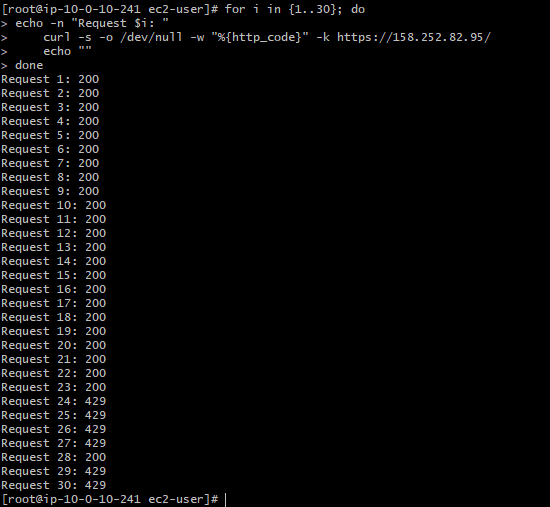
Install Apache Benchmark tool



Test rate limiting - Send 50 requests rapidly  
This should trigger rate limiting after the burst limit is exceeded



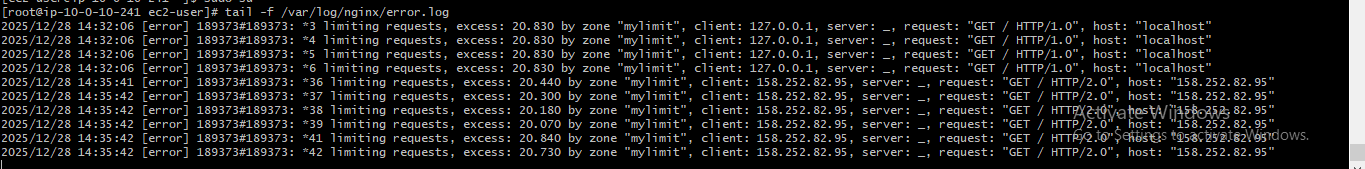
Alternative Testing with curl in rapid succession



1. Monitor Rate Limiting in Logs



Check error logs for rate limit messages

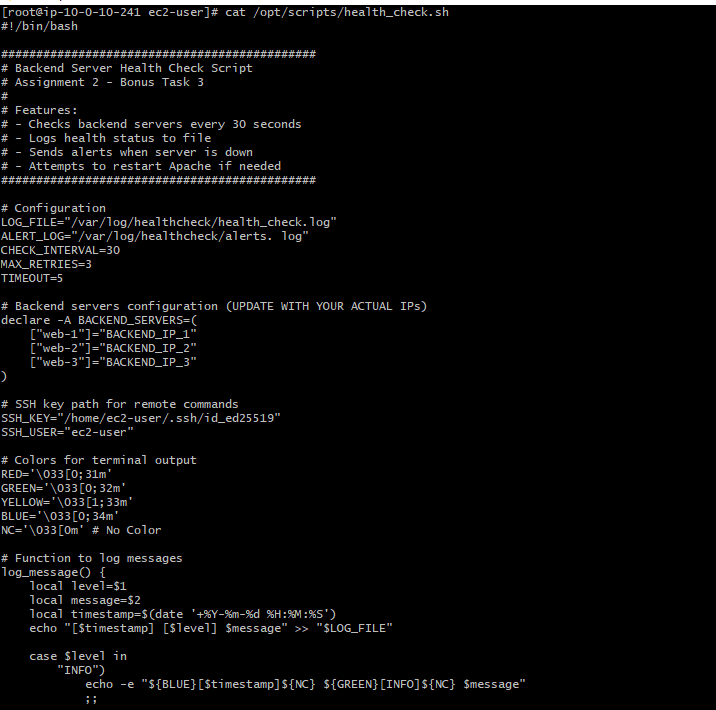


**Bonus 3: Health Check Automation**

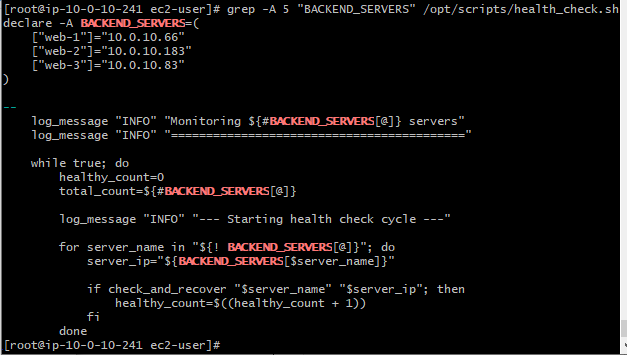
1. Create Scripts Directory



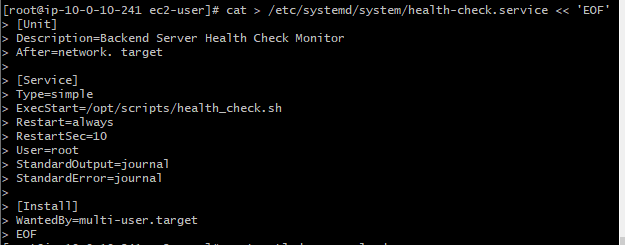
1. Create Health Check Script



1. Update Script with Actual Backend IPs



1. Create Systemd Service for Health Check



Reload systemd



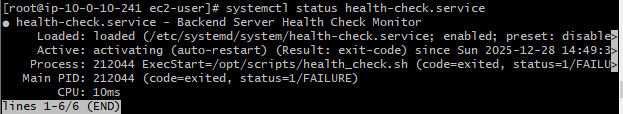
Enable service to start on boot



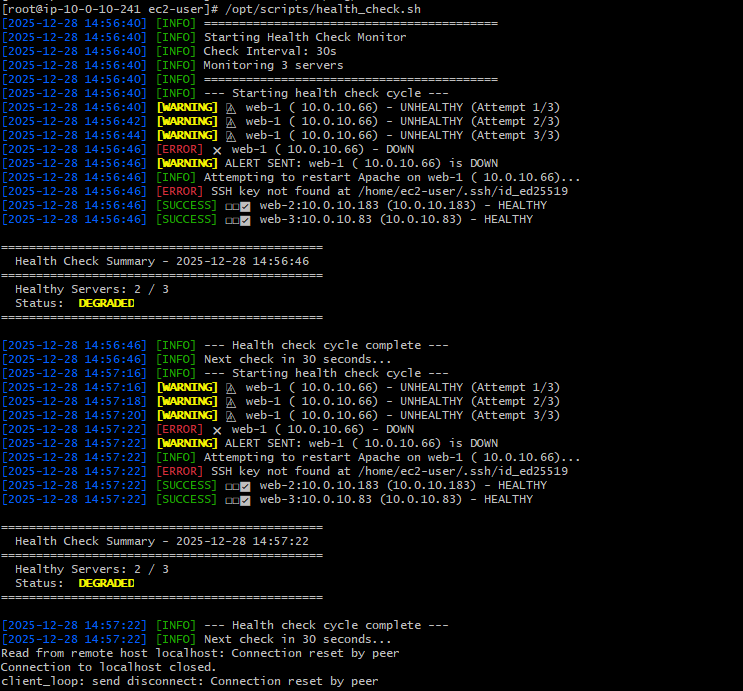
Start the service



Check status



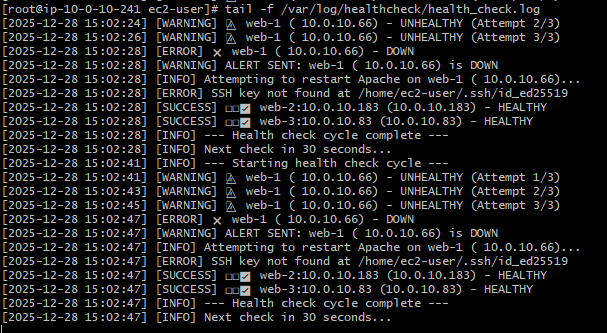
1. Run Health Check Manually for Testing



1. Start the Systemd Service

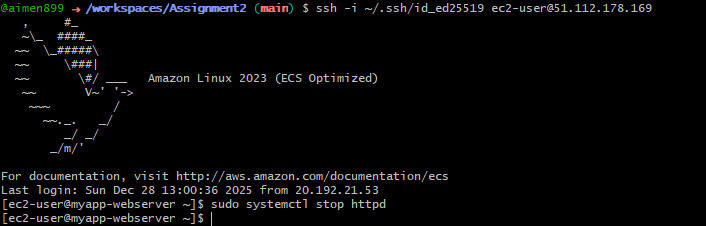


1. View the Logs



1. Test Alert by Stopping a Backend Server

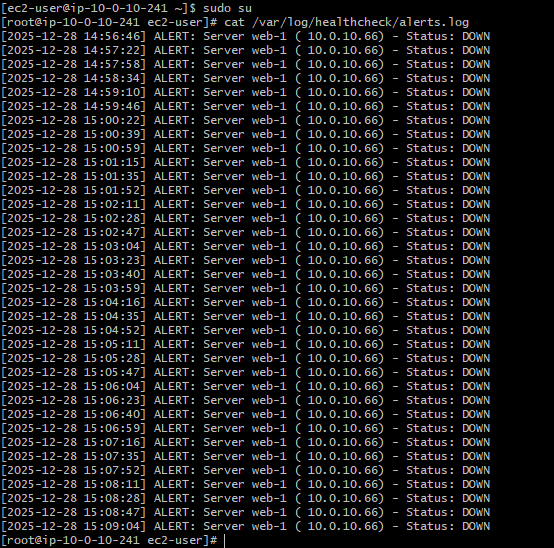
Open another terminal and stop Apache on one of the backend servers:



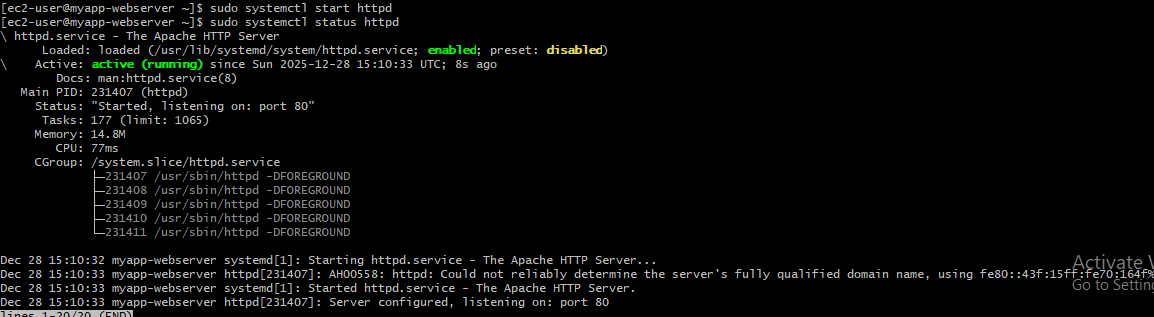
Then watch the health check logs on the Nginx server:

(ALERT: web-1 is down)

1. View Alert Log

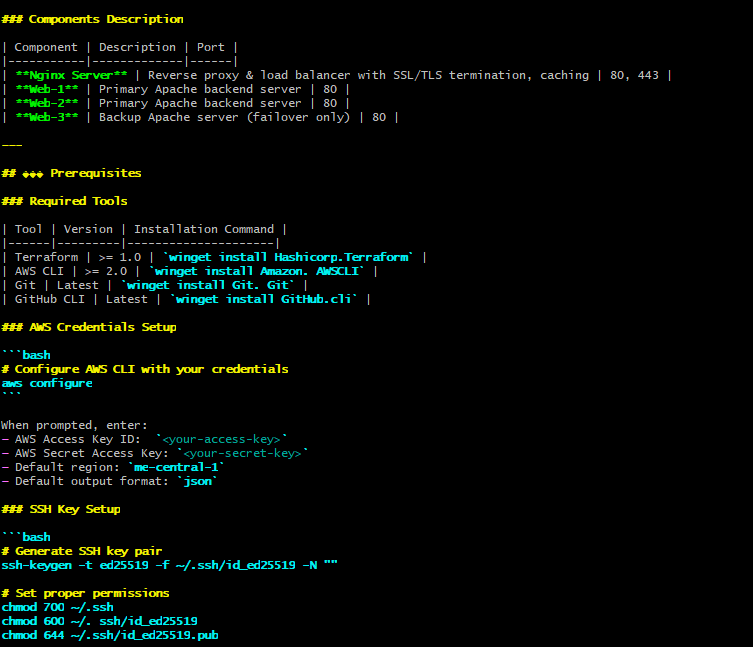
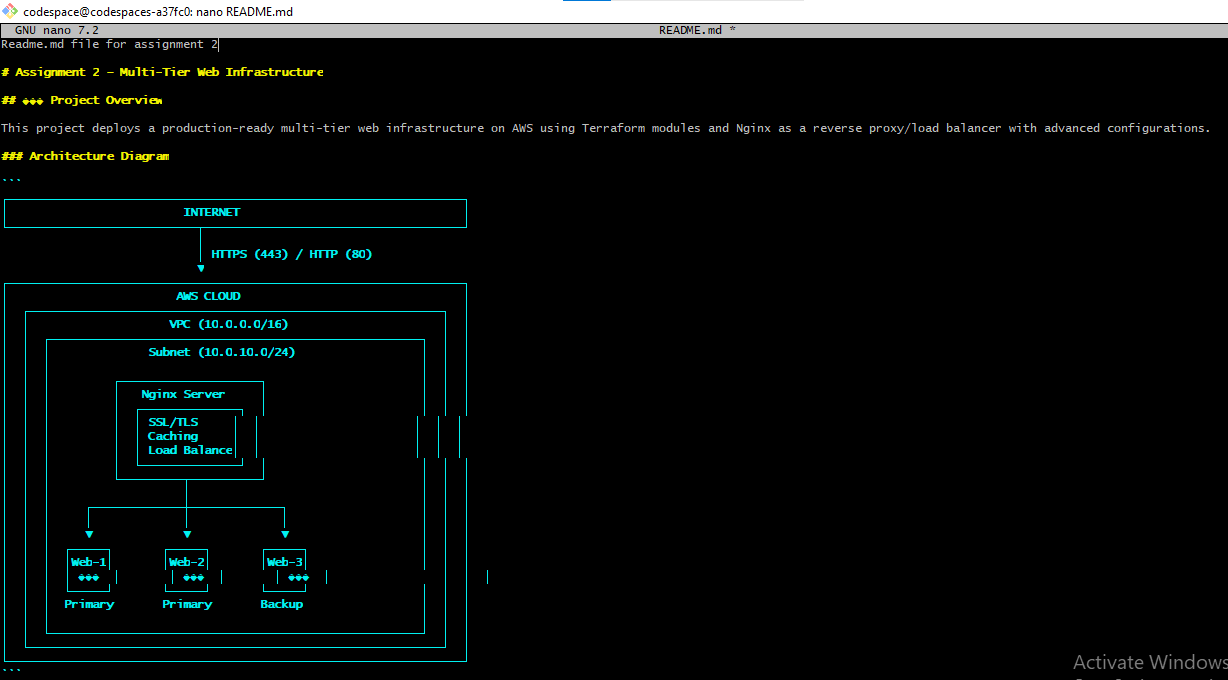


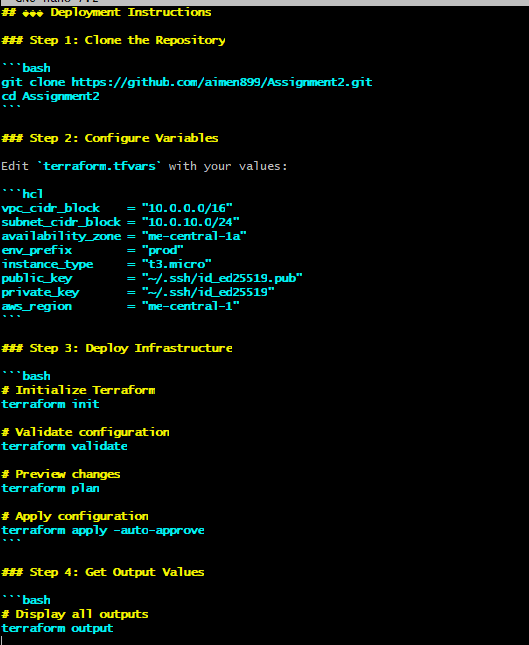
1. Restart the Backend Server

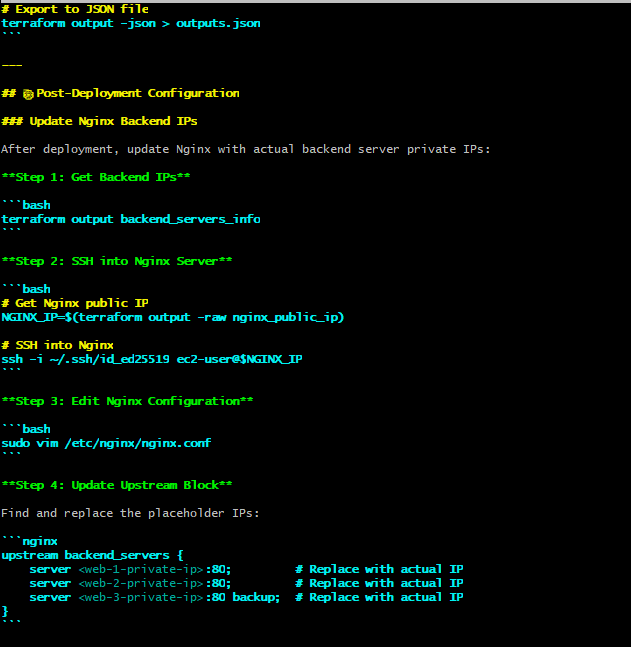


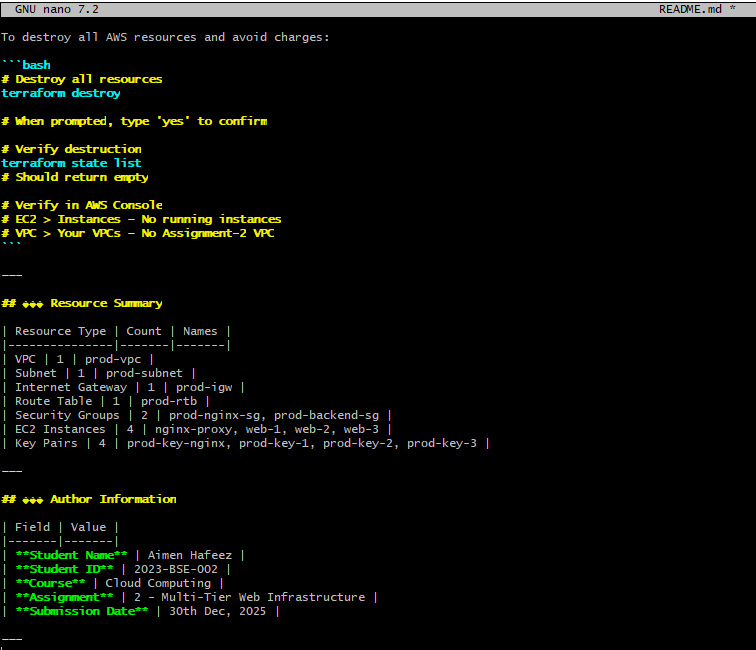
**Part 6: Documentation & Cleanup**

1. Create README. md

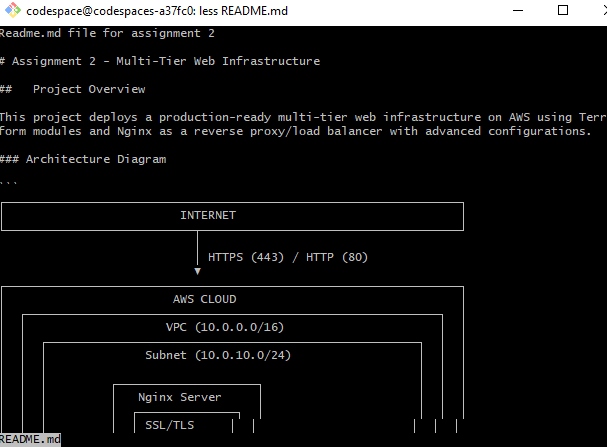




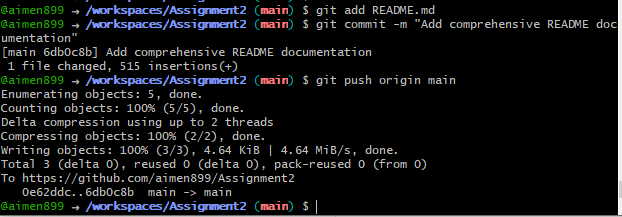




1. Alternatively, view README in a pager



1. Verify README renders correctly on GitHub



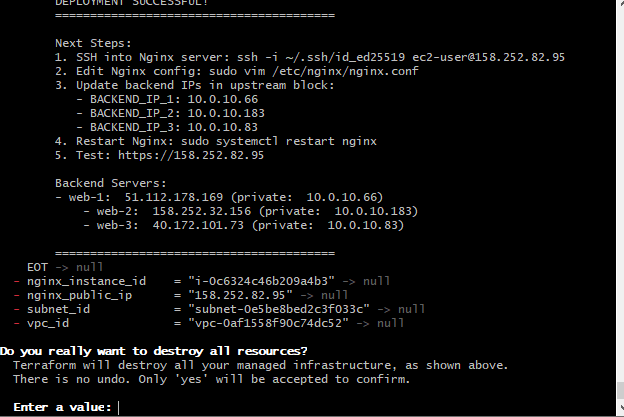
1. Infrastructure Cleanup

Prepare for Destruction

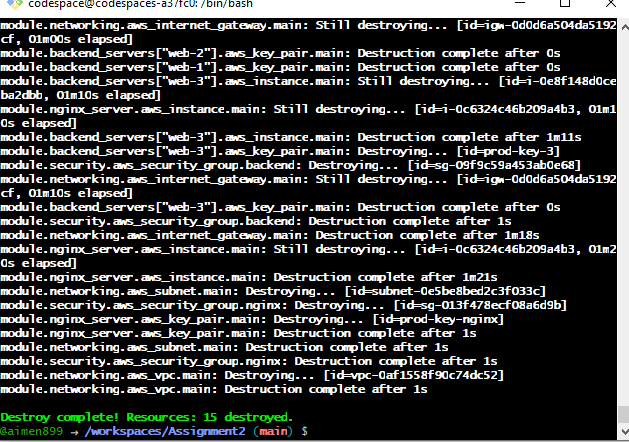
Before destroying, verify what resources exist:



1. Run Terraform Destroy



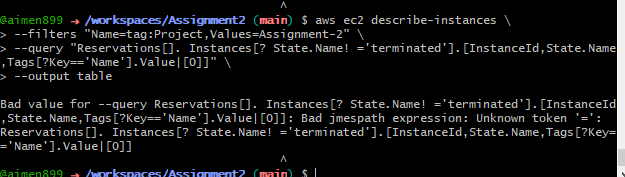
1. Confirm Destruction



1. Verify State File is Empty



1. Verify No Resources Remain Using AWS CLI



1. Verify in AWS Console

