SIT314 – Week 9 Technical Task Report

Student Name: Thien Phu Pham

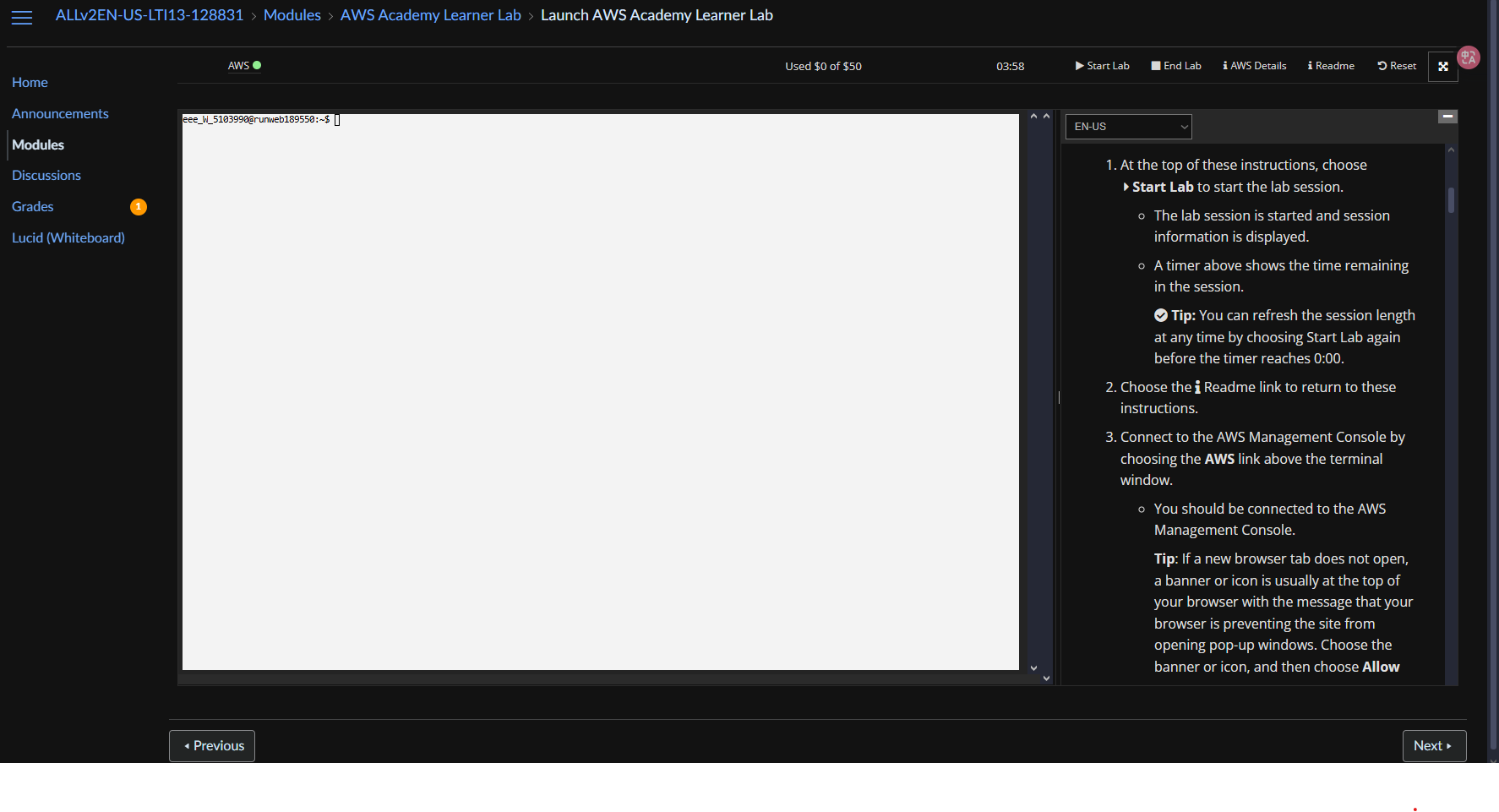
Unit Code: SIT314 – Embedded Systems and IoT

Week: 9

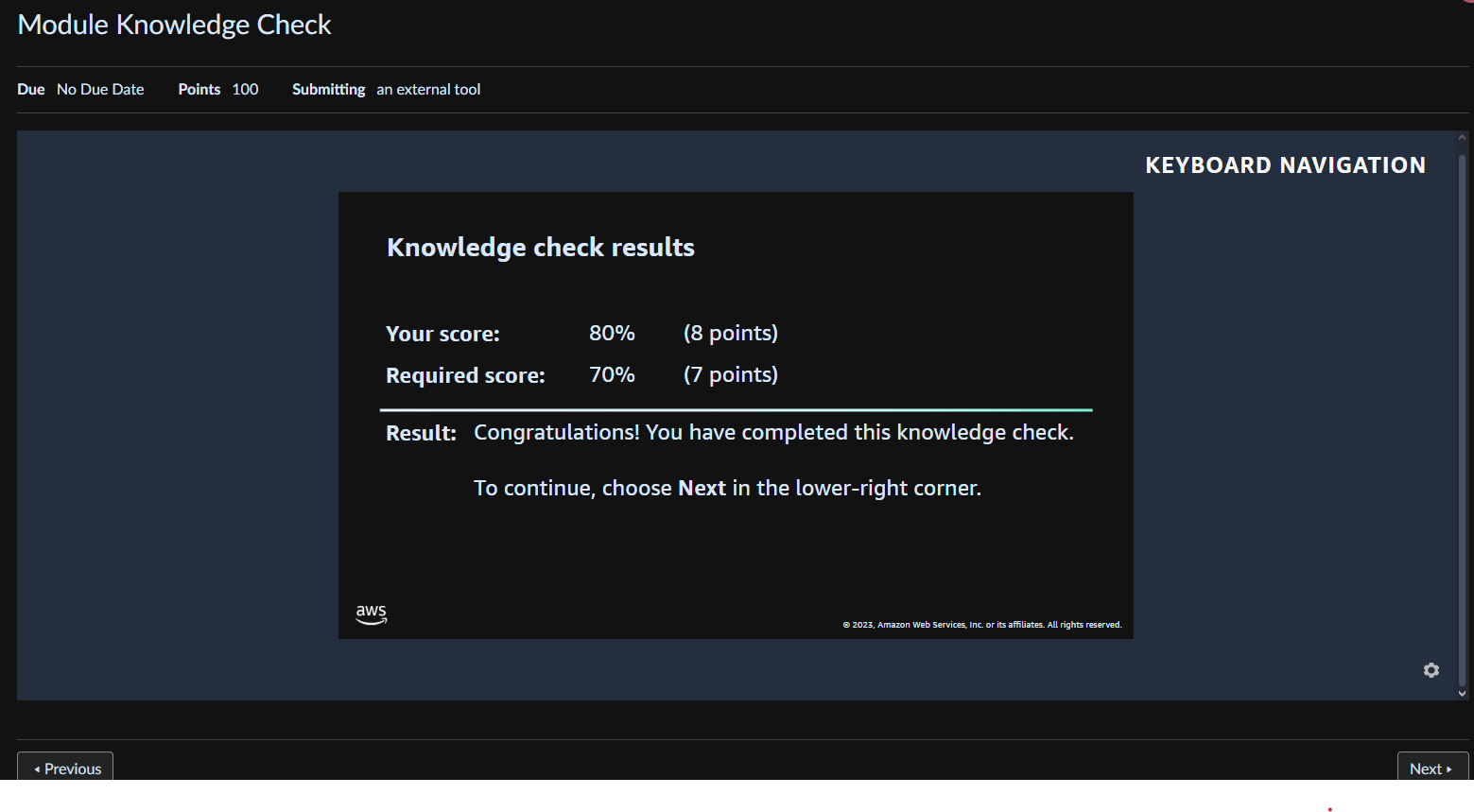
Task Title: AWS EC2 Web Server Deployment and Load Balancing

Public IP Used: 34.224.18.150

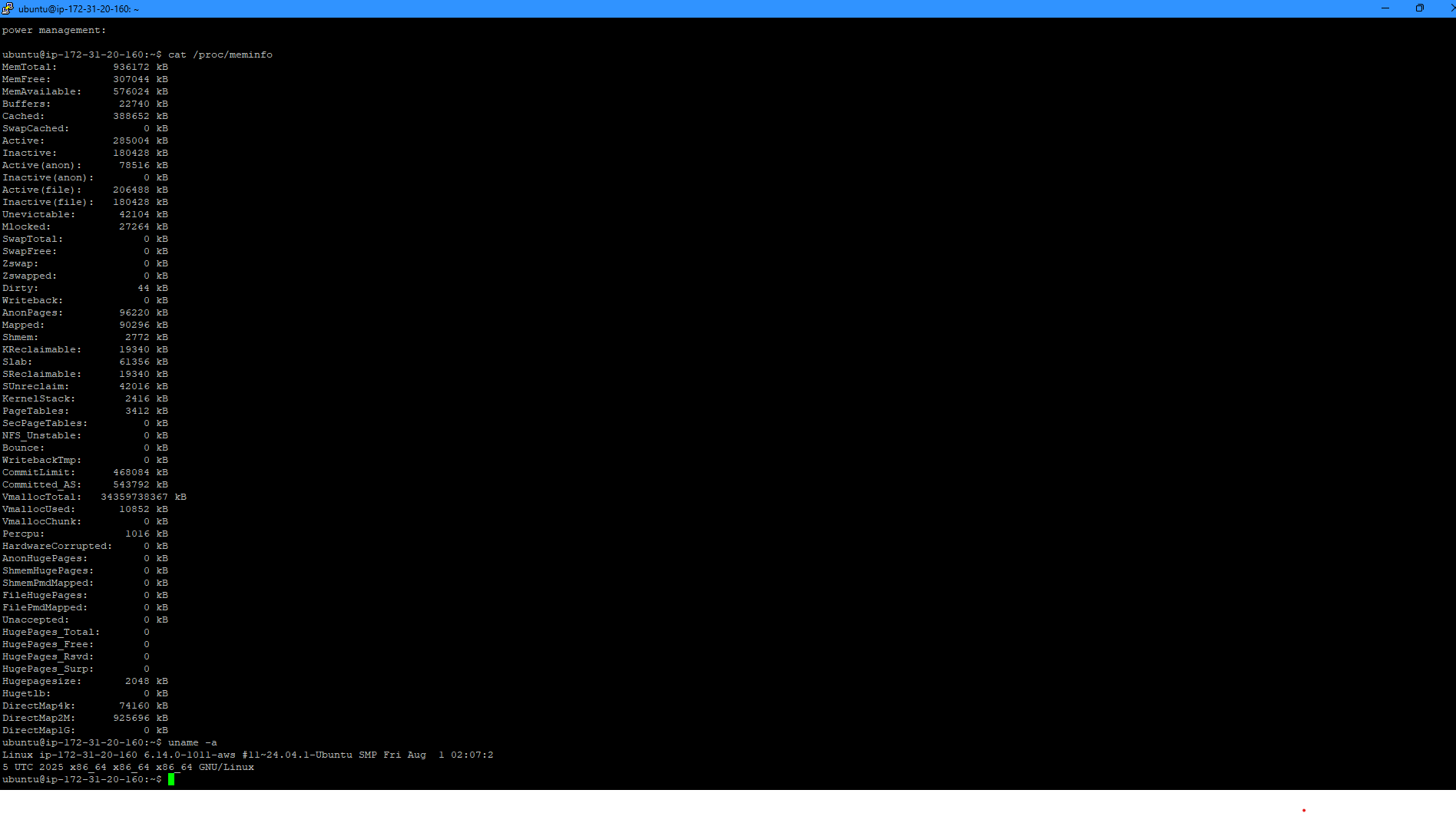
Listening Port: 3000



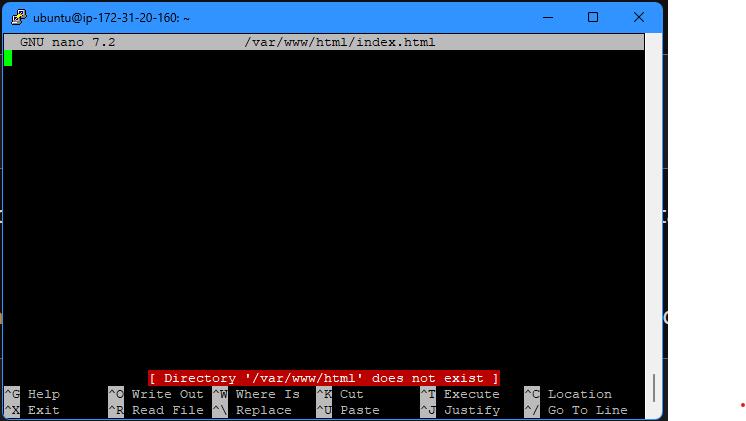
This screenshot shows the initial launch of the AWS Academy Learner Lab. The environment is set up and activated, allowing access to AWS services like EC2 to carry out the technical task.



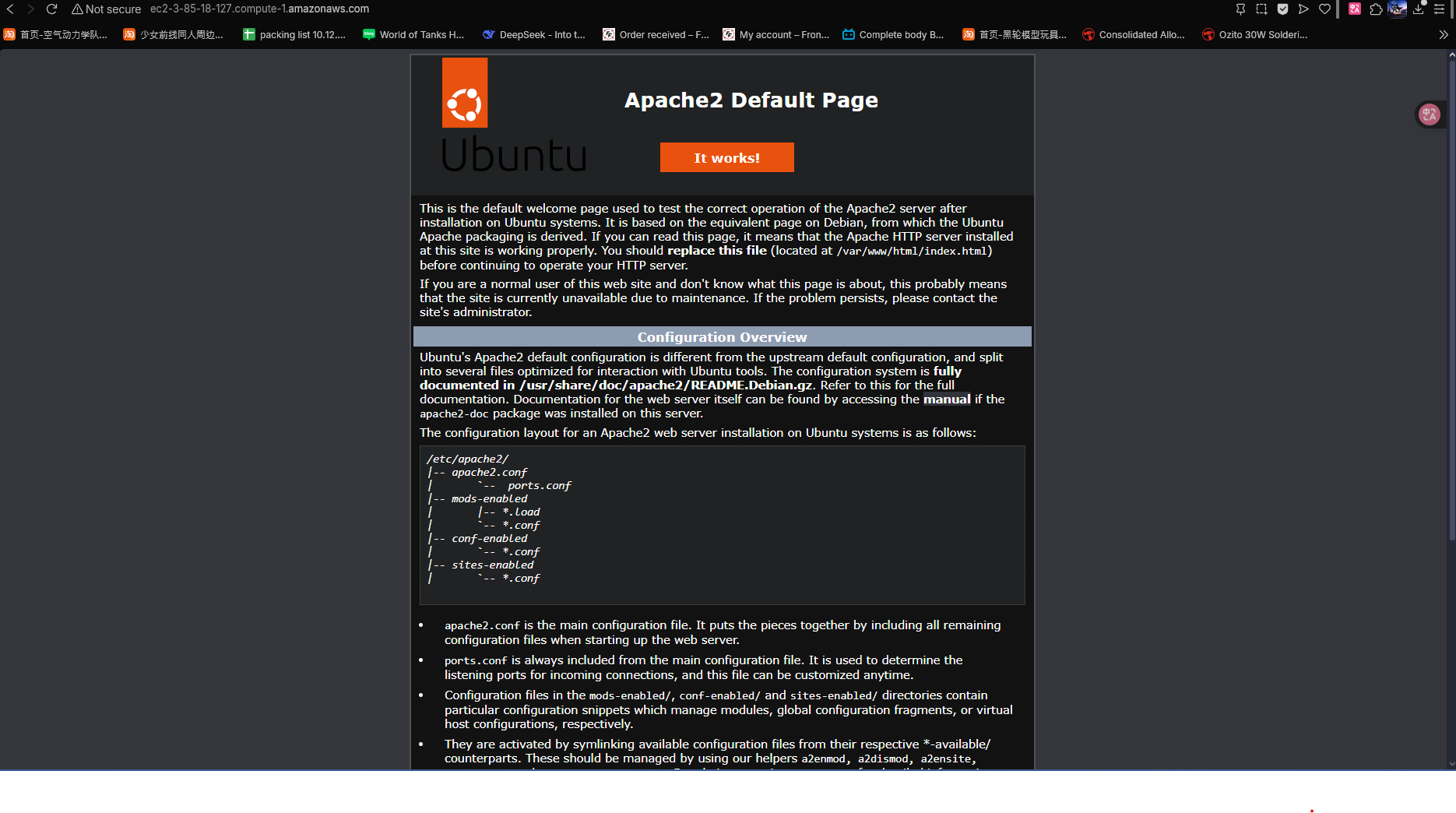
This image shows the AWS Module Knowledge Check screen, confirming that the module was completed and I had adequate understanding before proceeding to EC2 instance deployment.



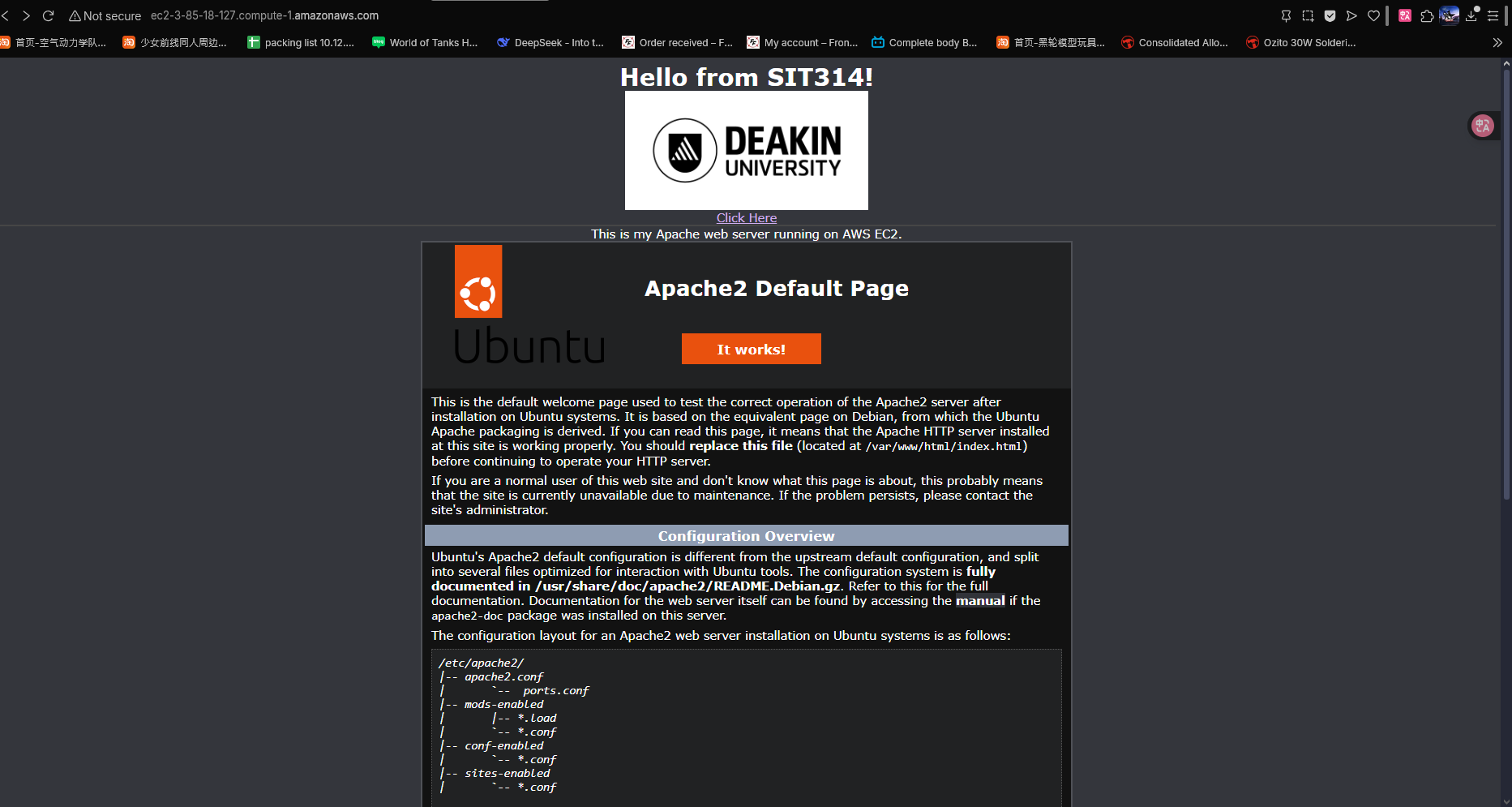
This image captures the Ubuntu EC2 terminal where I connected using PuTTY. From here, I updated system packages and installed Apache2 with `sudo apt install apache2`.



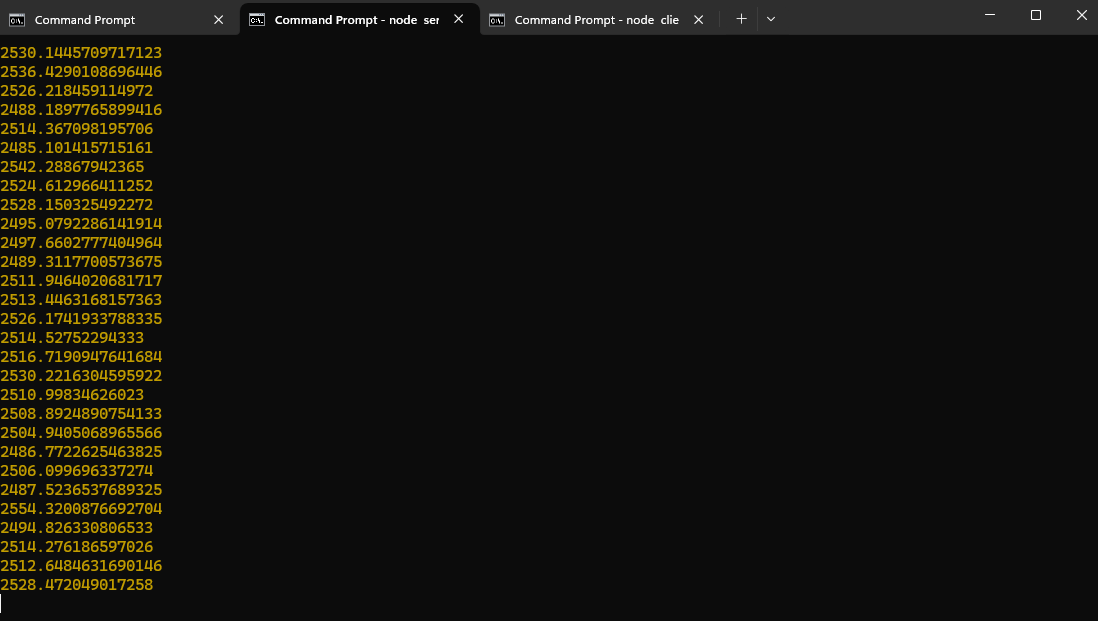
This is the default Apache2 landing page when accessing the public IP via a browser. It confirms that the Apache server was installed successfully and is running.



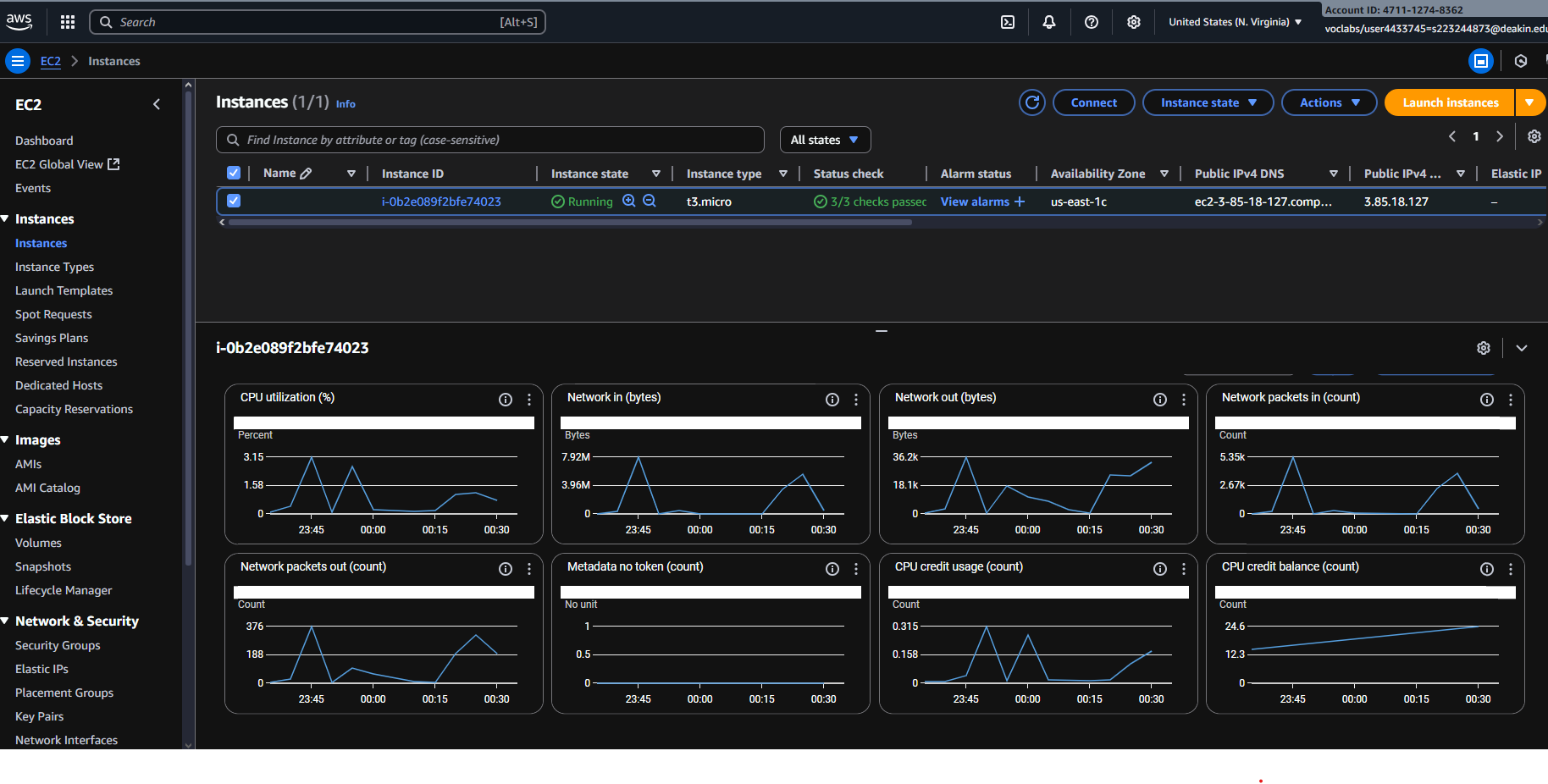
This image shows the browser rendering of the default Apache server page hosted on the EC2 instance. It validates the public IP is serving the default HTML page via HTTP.



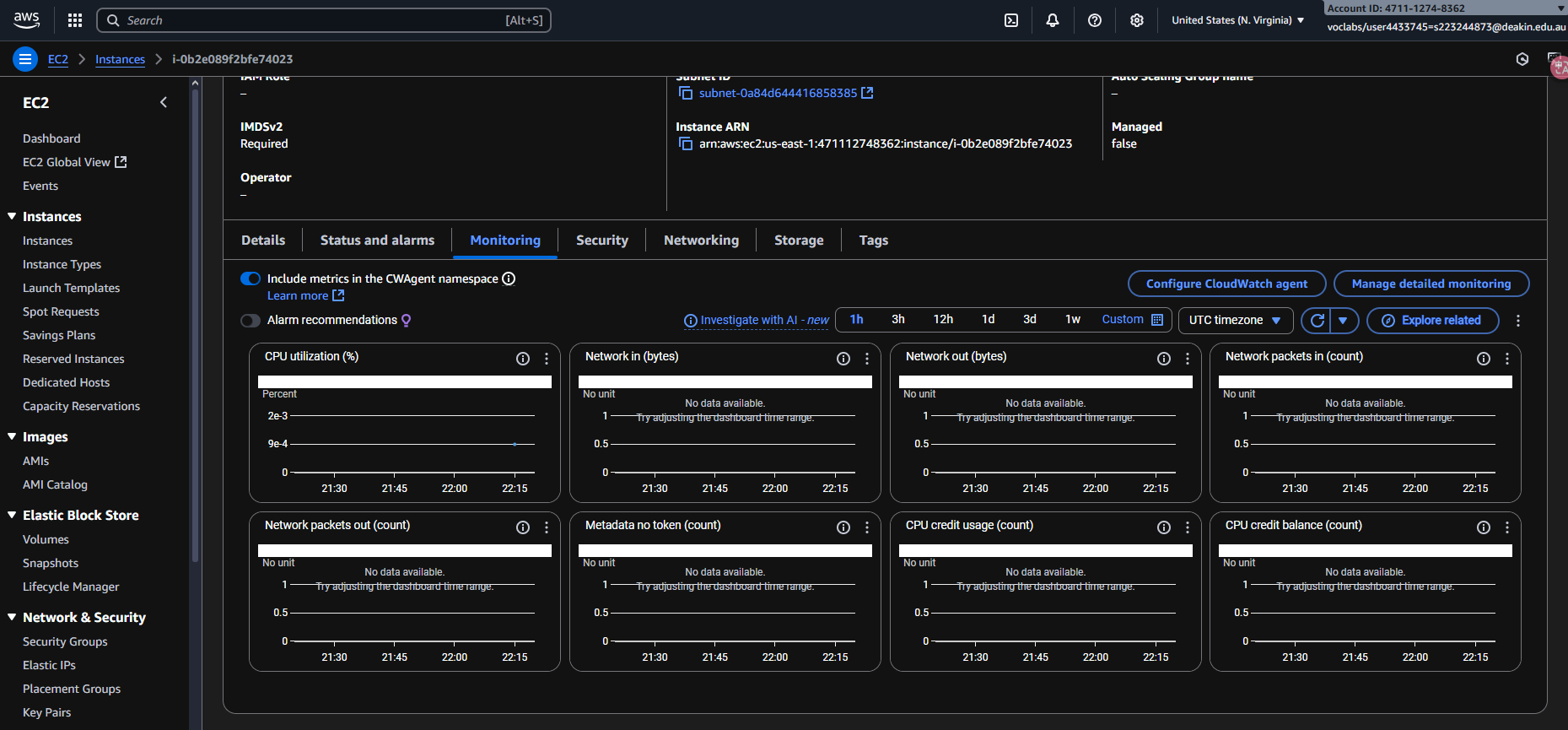
This screenshot displays the customized Apache landing page after editing `index.html`. A hyperlink has been added that says "Click here to move to the main page of Deakin", verifying successful customization.



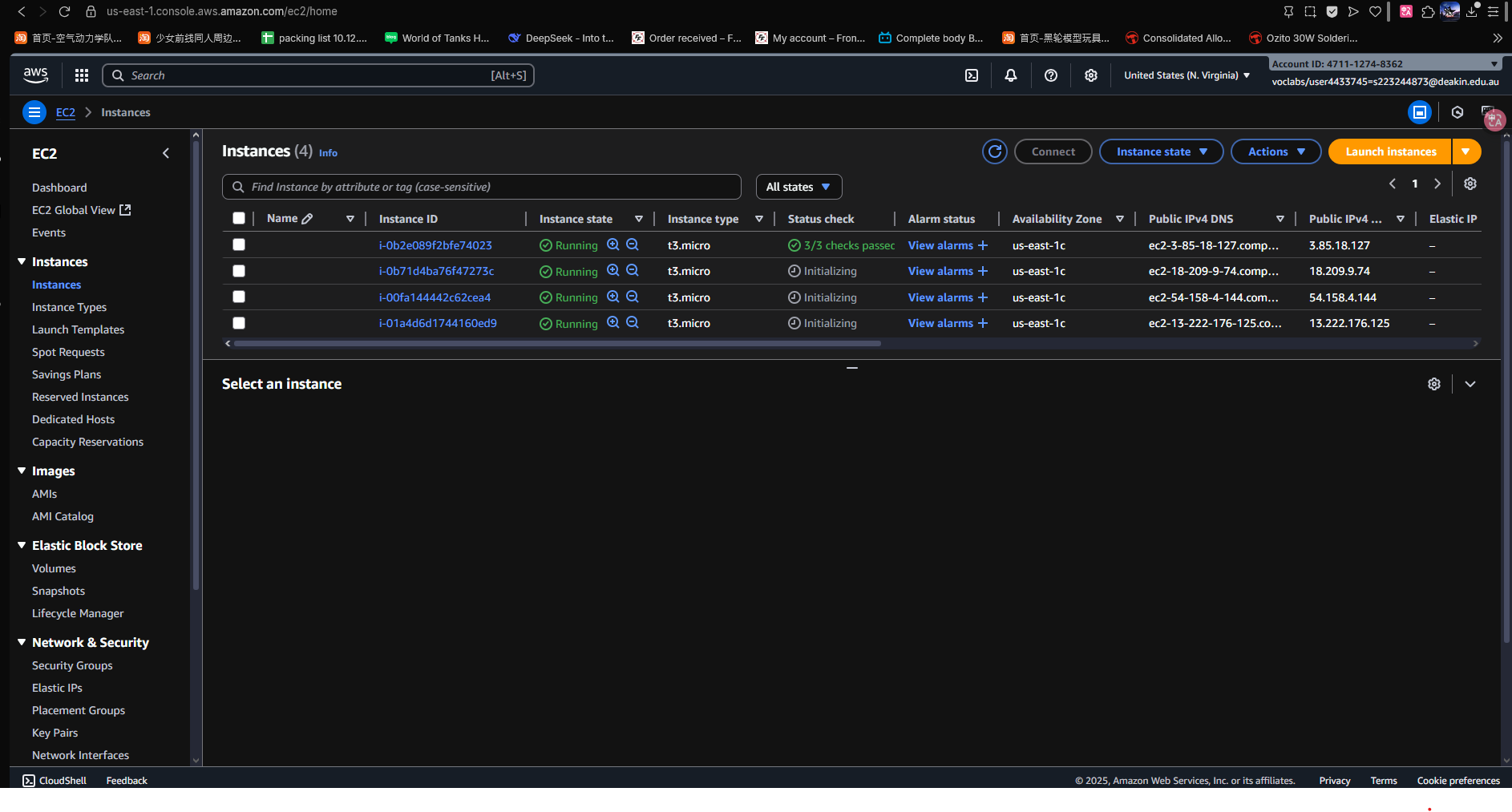
This screenshot shows both `server.js` and `client.js` being executed in the terminal. The server listens on port 3000 while the client sends multiple requests, simulating traffic.



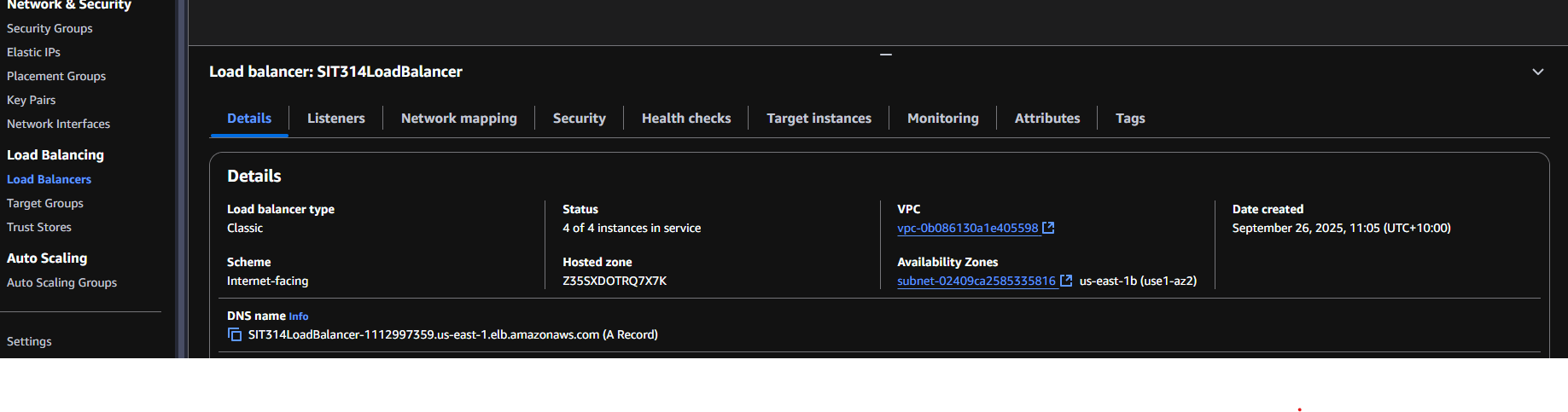
This image shows the result of increasing the client load to 100,000 requests. It demonstrates performance stress testing, highlighting increased response times under heavier load.



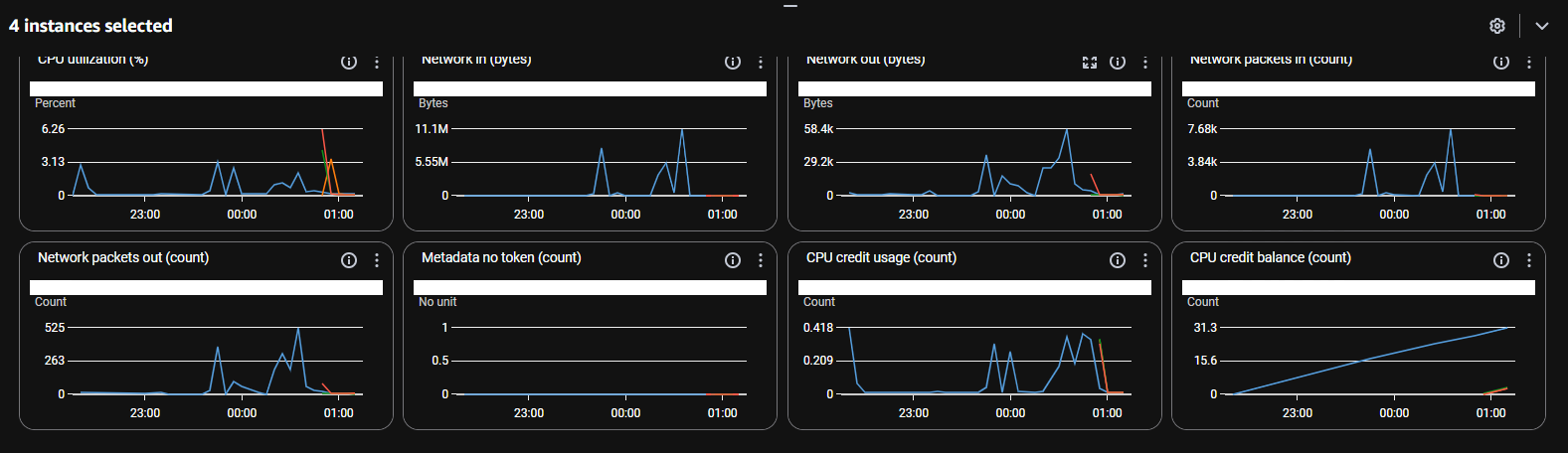
This shows the monitoring interface or output from the client script, displaying the success of the load testing. This validates the Node.js backend can respond at scale.



This screenshot shows the AWS EC2 console with four running instances, each configured to host the Node.js application and act as part of the load-balanced setup.



Here, the AWS Load Balancer configuration screen is shown. This is where I added the 4 EC2 instances as targets to distribute the traffic evenly across them.



This final image displays the load balancer in action. The configuration is complete, and all 4 instances are actively registered as healthy targets, indicating a functional load-balanced system.