**AWS CONFIGURATION - PROBLEM 2**

1. **INTRODUCTION**

This documentation outlines the step-by-step configuration and deployment process of a simple application (patient-emr) on the Amazon Web Services platform. The web application consists of a PostgreSQL database that stores data entered by users. The web application has been deployed using AWS Elastic Beanstalk and RDS for the database. The entire infrastructure has been designed to be highly available, scalable and secure.

1. **INFRASTRUCTURE OVERVIEW**

The designed architecture follows a multi-tiered design method which leverages the following AWS Services:

1. **Amazon VPC** - the infrastructure consists of a custom virtual private network with public and private subnets.
2. **Elastic Beanstalk** - the architecture includes a managed service for the deploying of the backend and frontend applications.
3. **Amazon RDS (PostgreSQL)** - the infrastructure also leverages a managed service for the customization of relational databases.
4. **AWS IAM** - Identity and Access Management for security roles.
5. **AWS CloudWatch** - for logging and monitoring services.
6. **AWS Auto Scaling** - to ensure high availability and scalability of services.
7. **AWS CloudFormation** - Infrastructure as code to automate deployments.

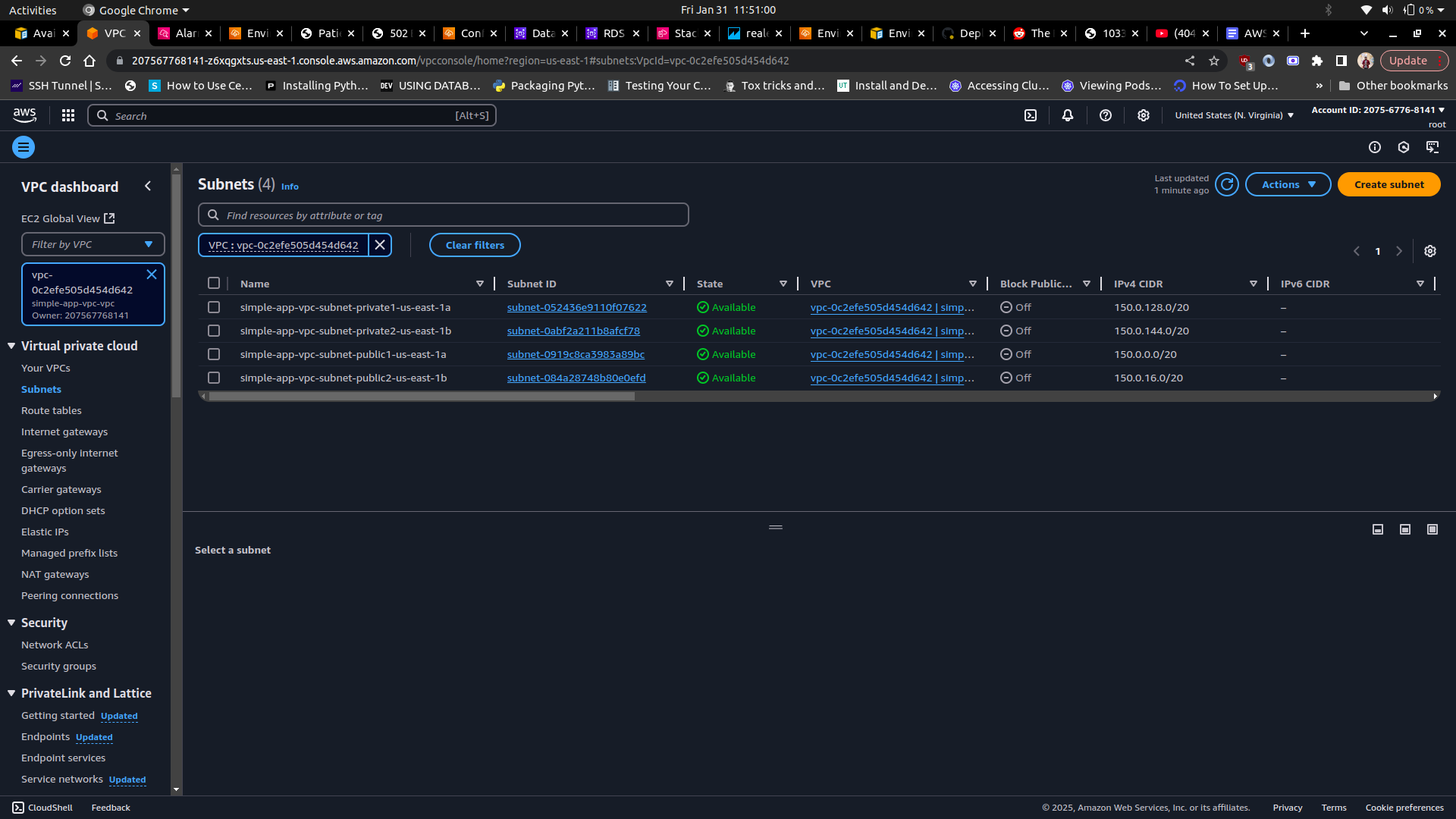
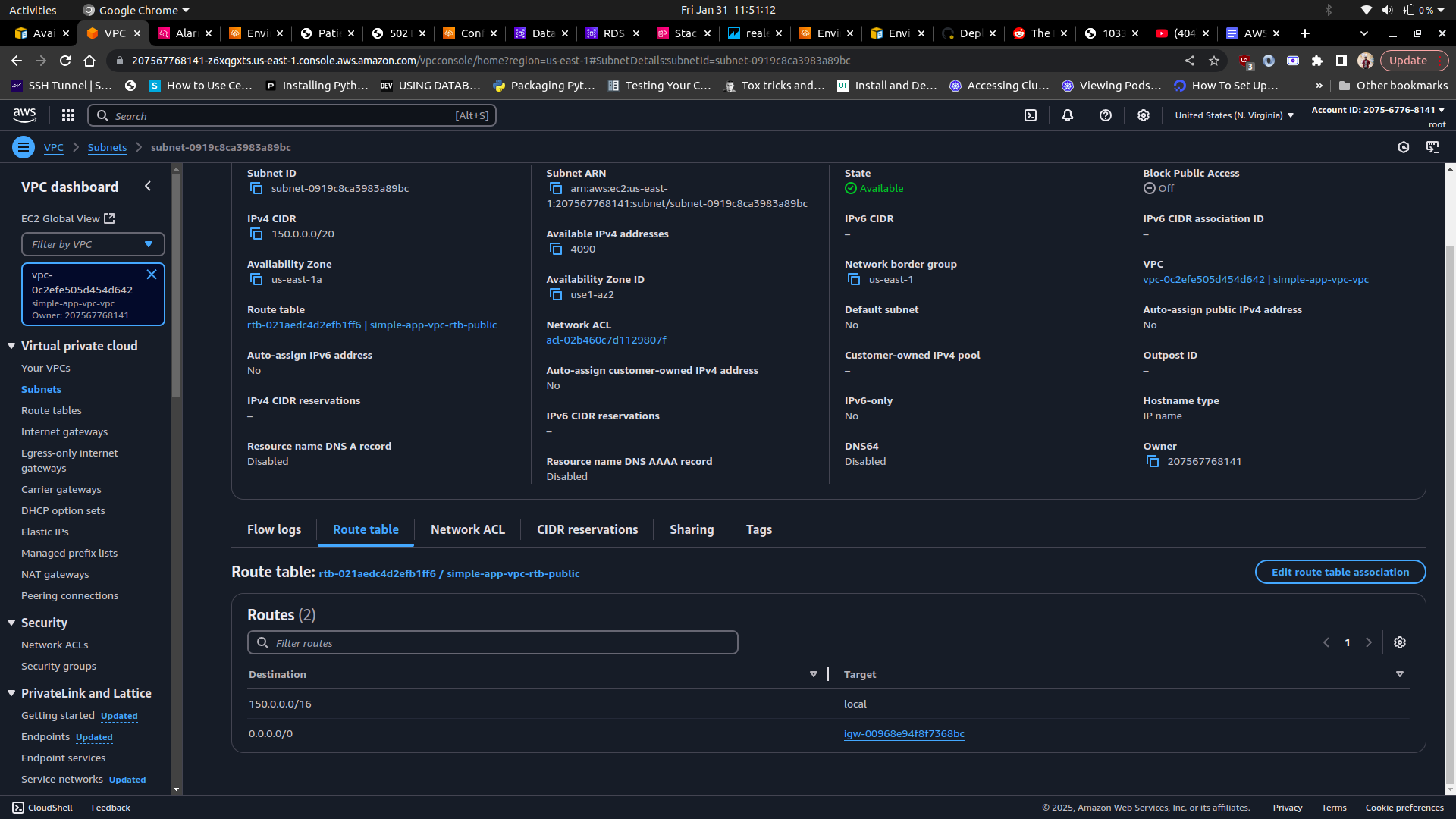
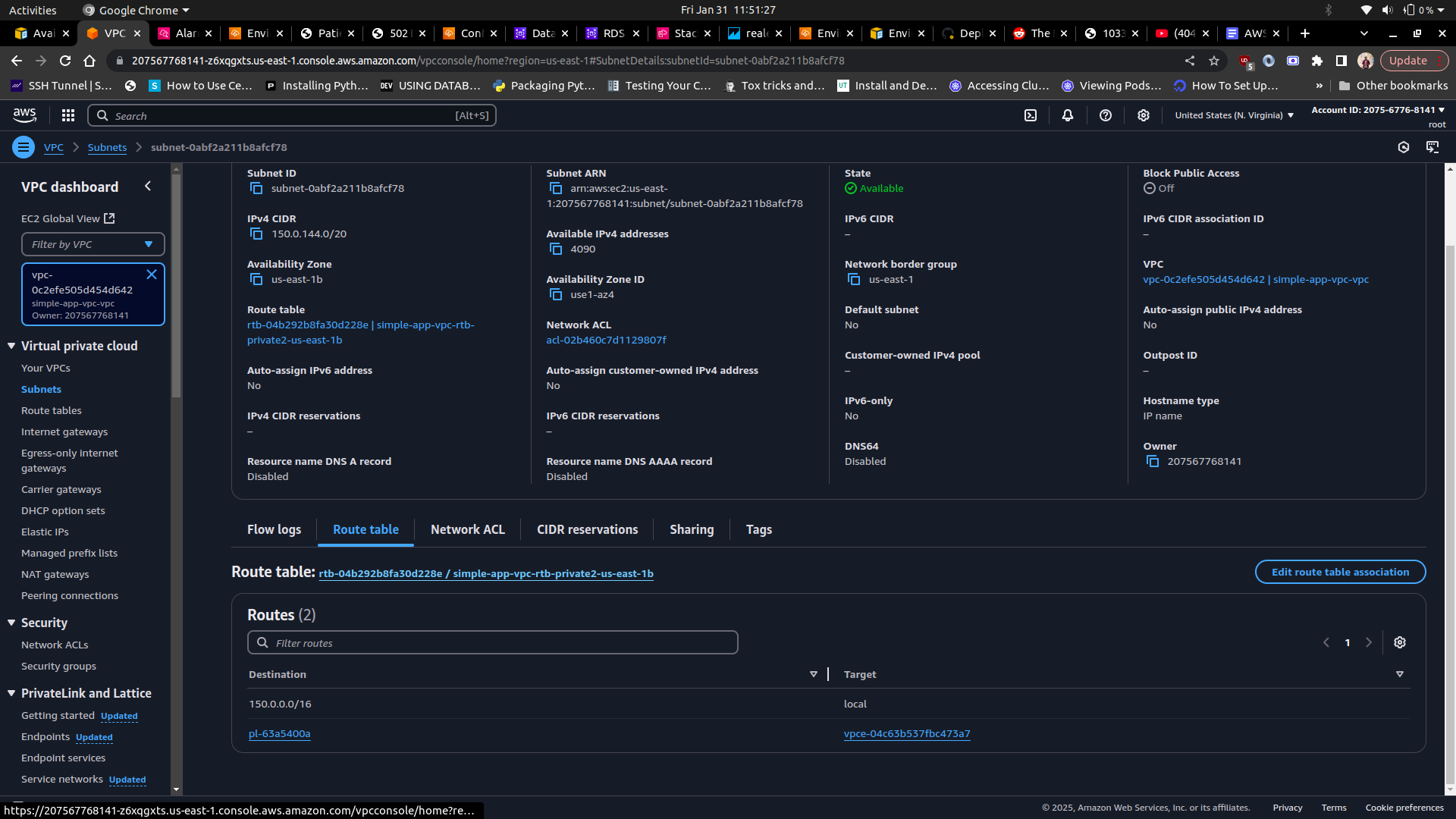
**3. AWS RESOURCE CONFIGURATION**

**3.1. Virtual Private Cloud (VPC)**

A custom VPC was designed in the architecture that includes:

* Two public subnets for the web application and an Elastic Load Balancer.
* Two private subnets for the database.
* An Internet Gateway for public access.
* A NAT Gateway to allow private subnets to access the internet securely.

The images below demonstrate the VPC and subnet configurations:

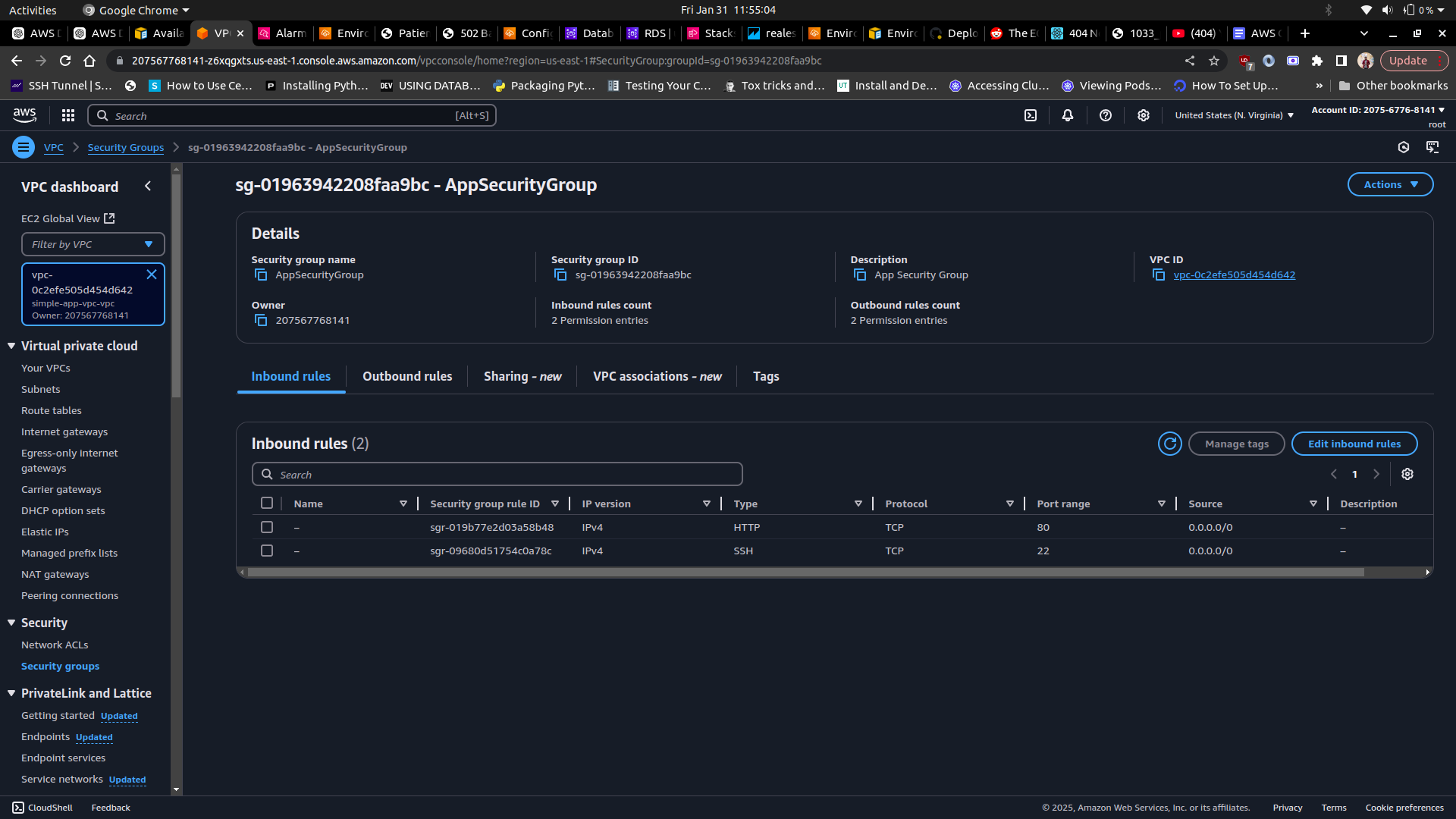
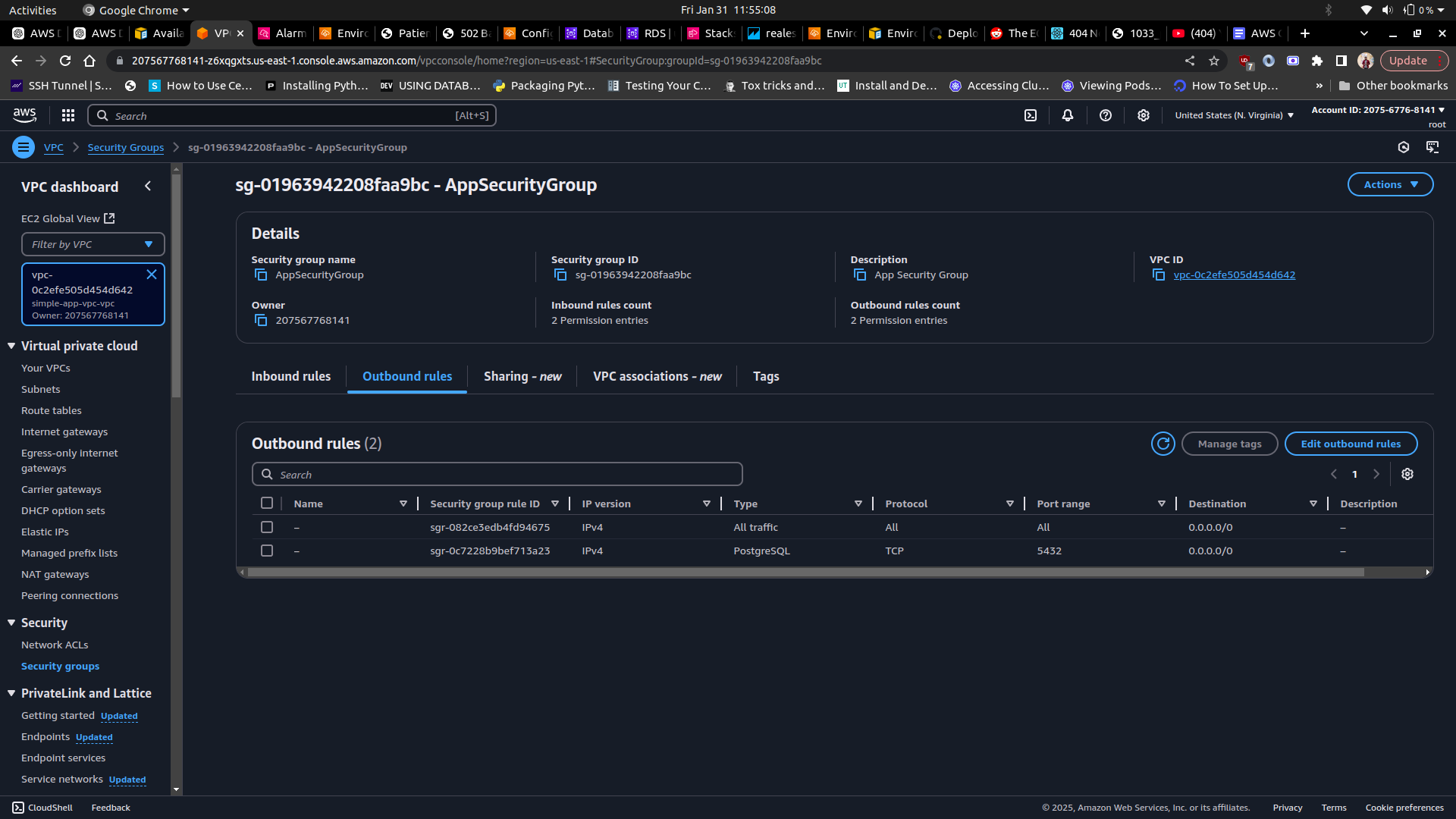
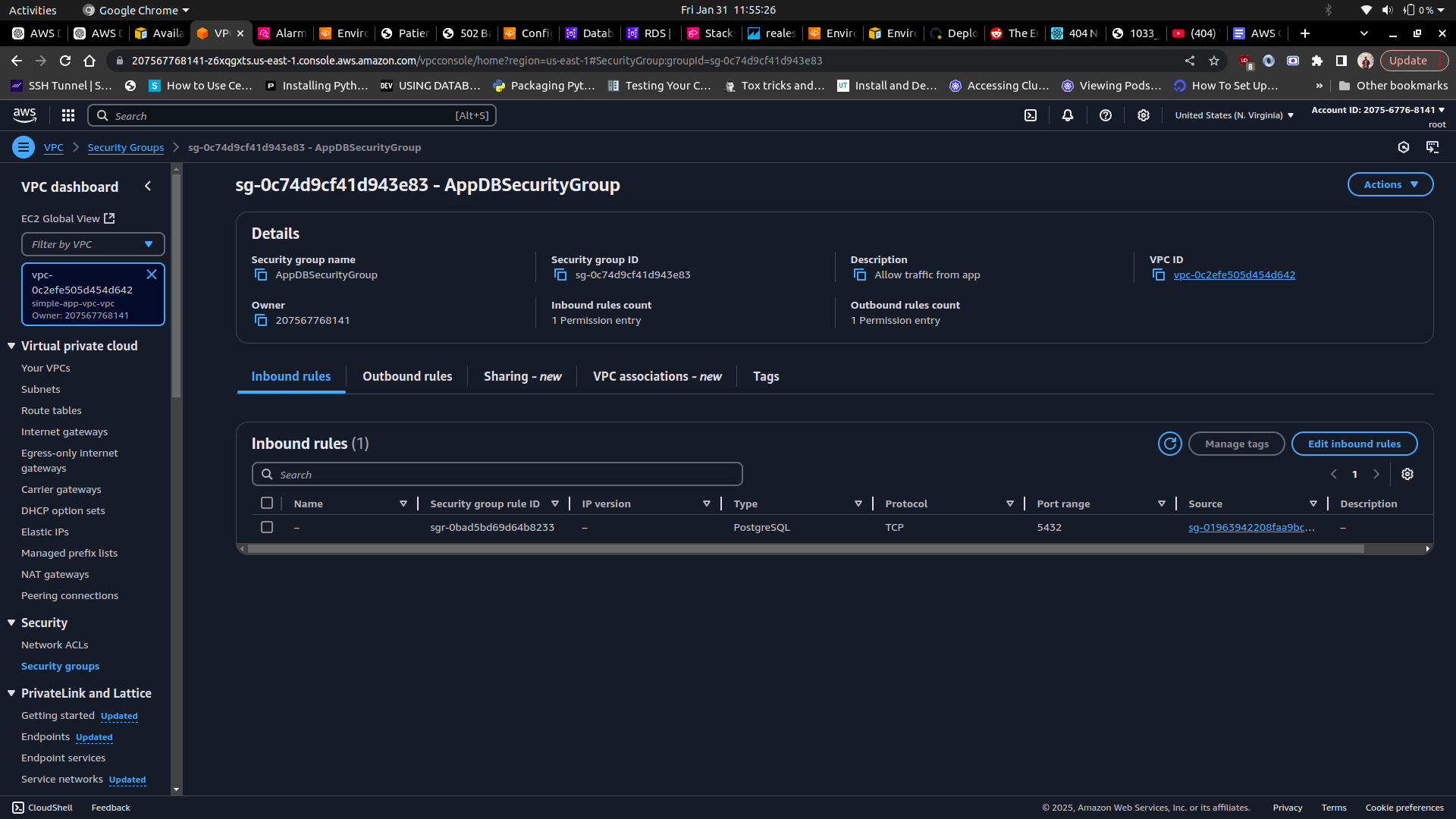


**3.2. Security Groups**

A couple of security groups have been configured to be used in the VPC, which include:

* Backend Security Group - this security group allows inbound traffic on port 8000 and 8080 from the application load balancer and an outbound traffic to the RDS database on port 5432.
* Database Security Group - this group allows PostgreSQL connections on port 5432 from the backend security group.

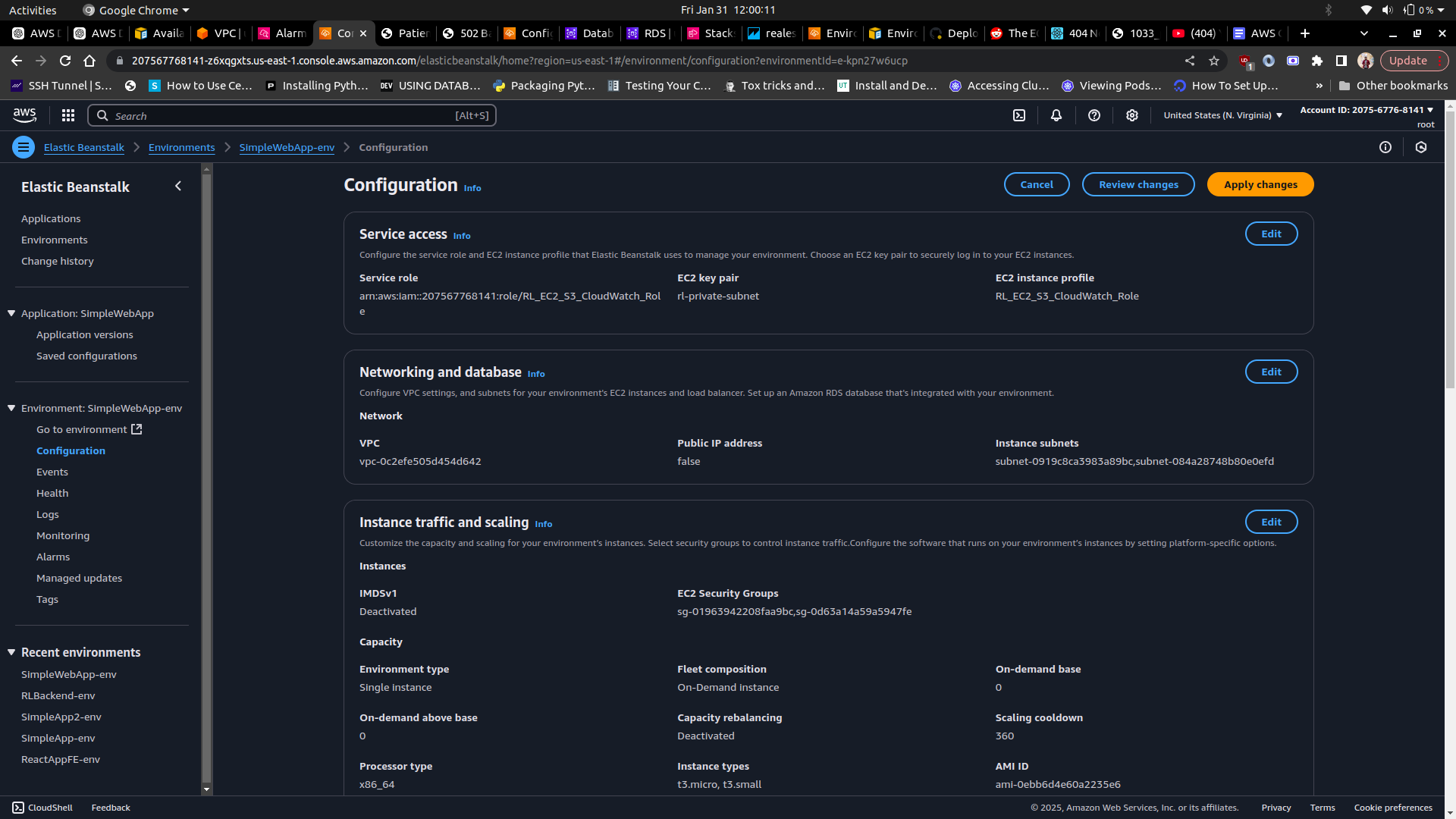
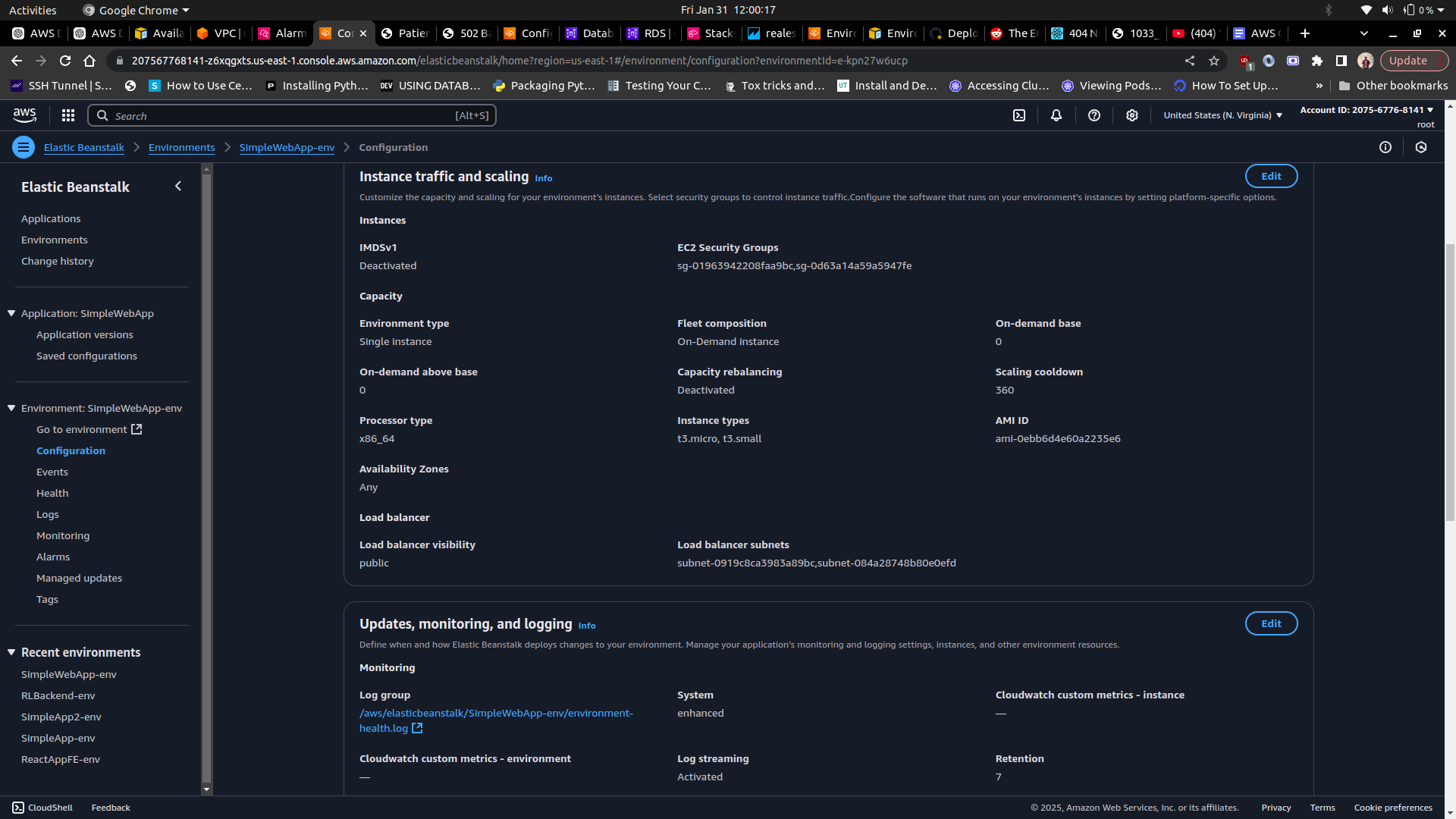
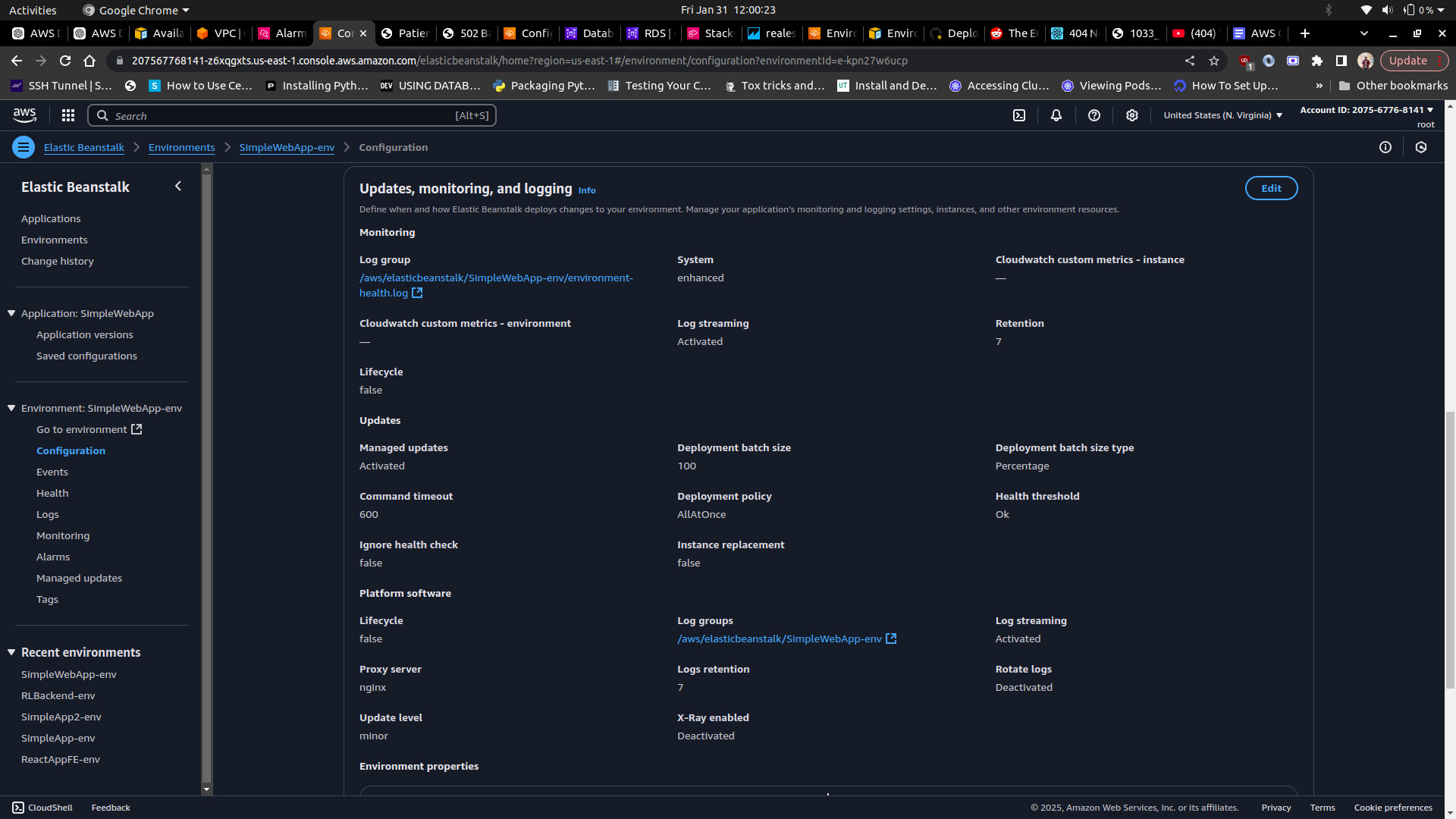
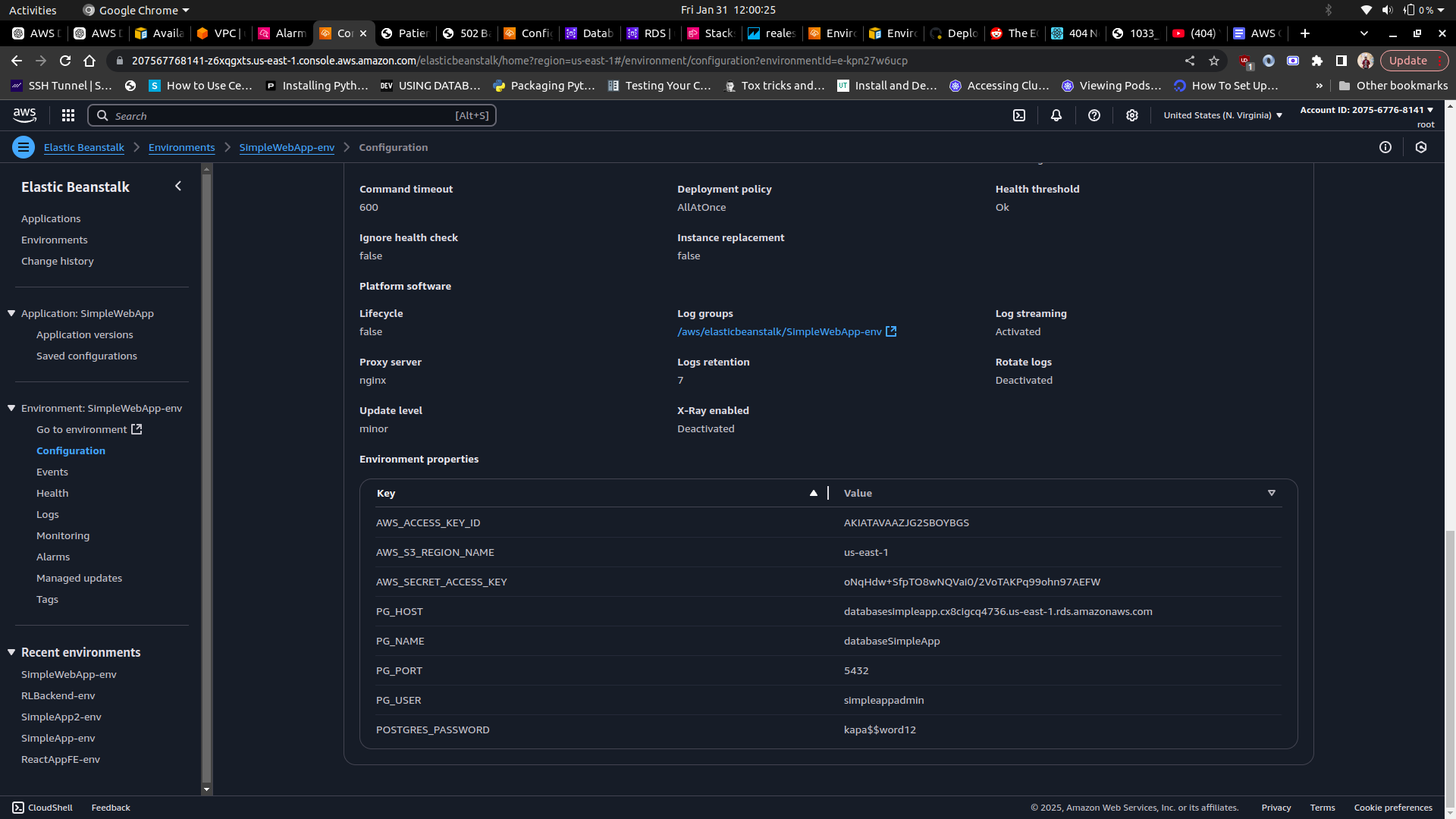
The images below demonstrate the configured security groups:



**3.3. Elastic Beanstalk Configuration**

**3.3.1 Web Application Deployment**

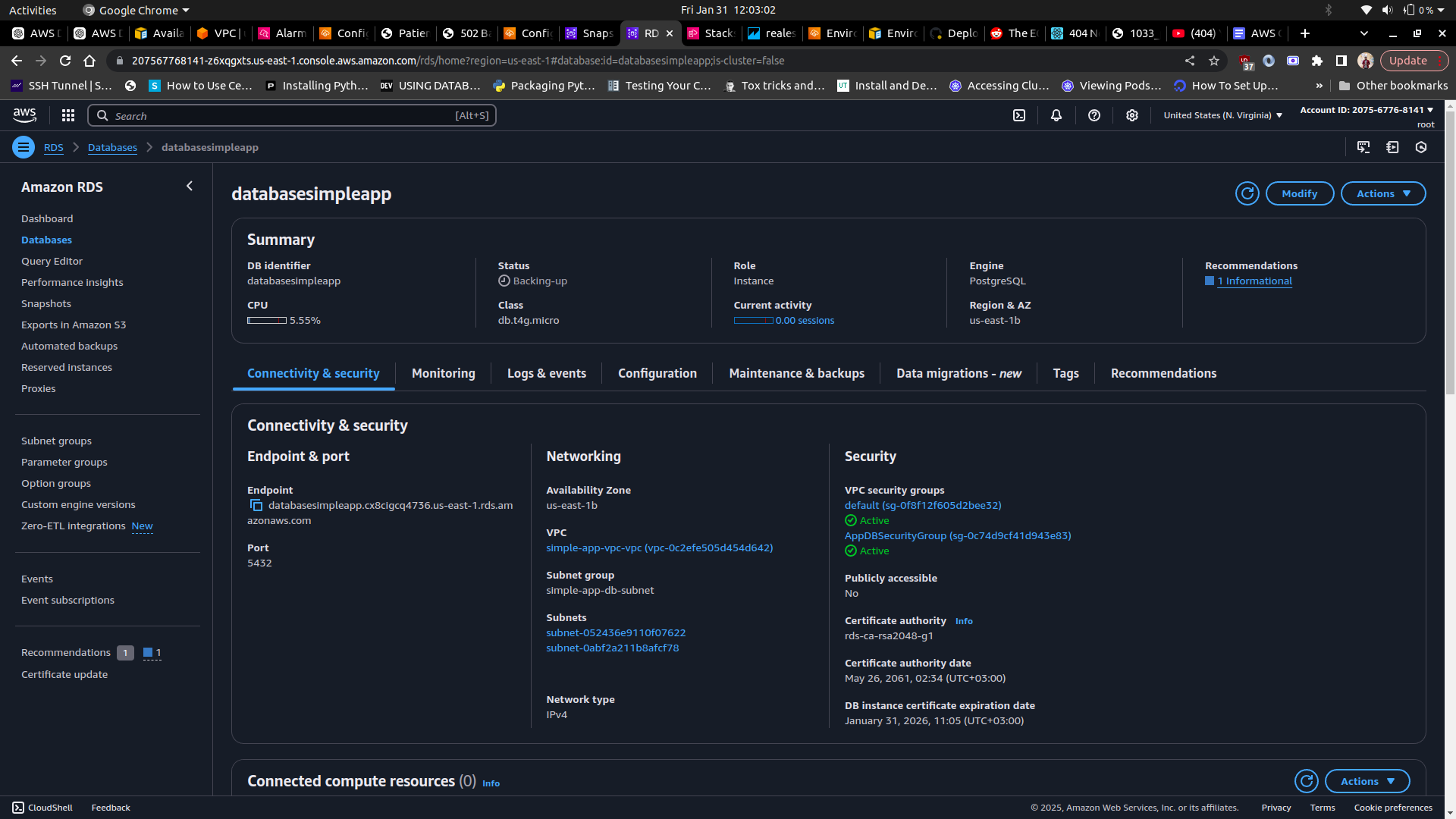
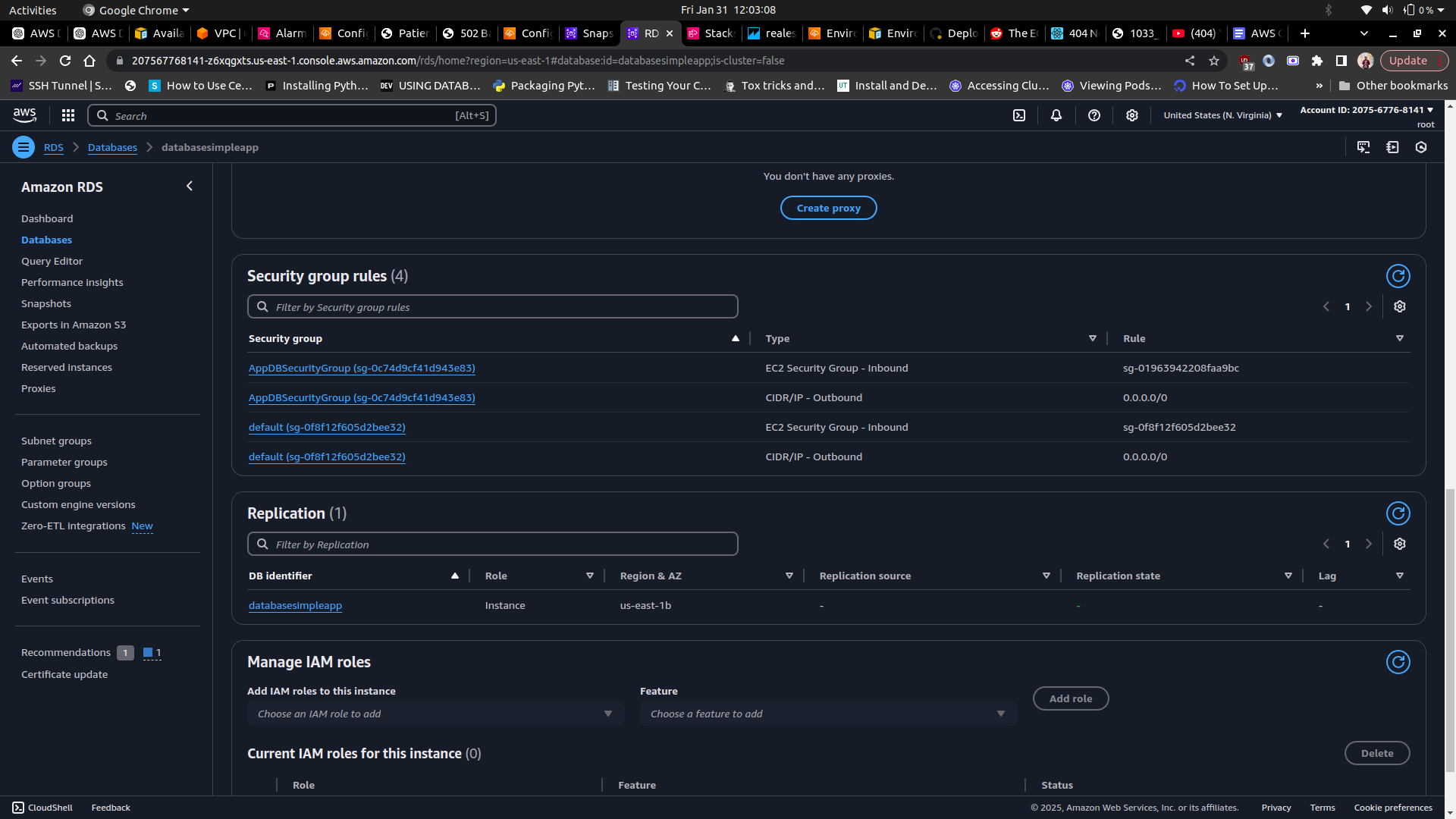
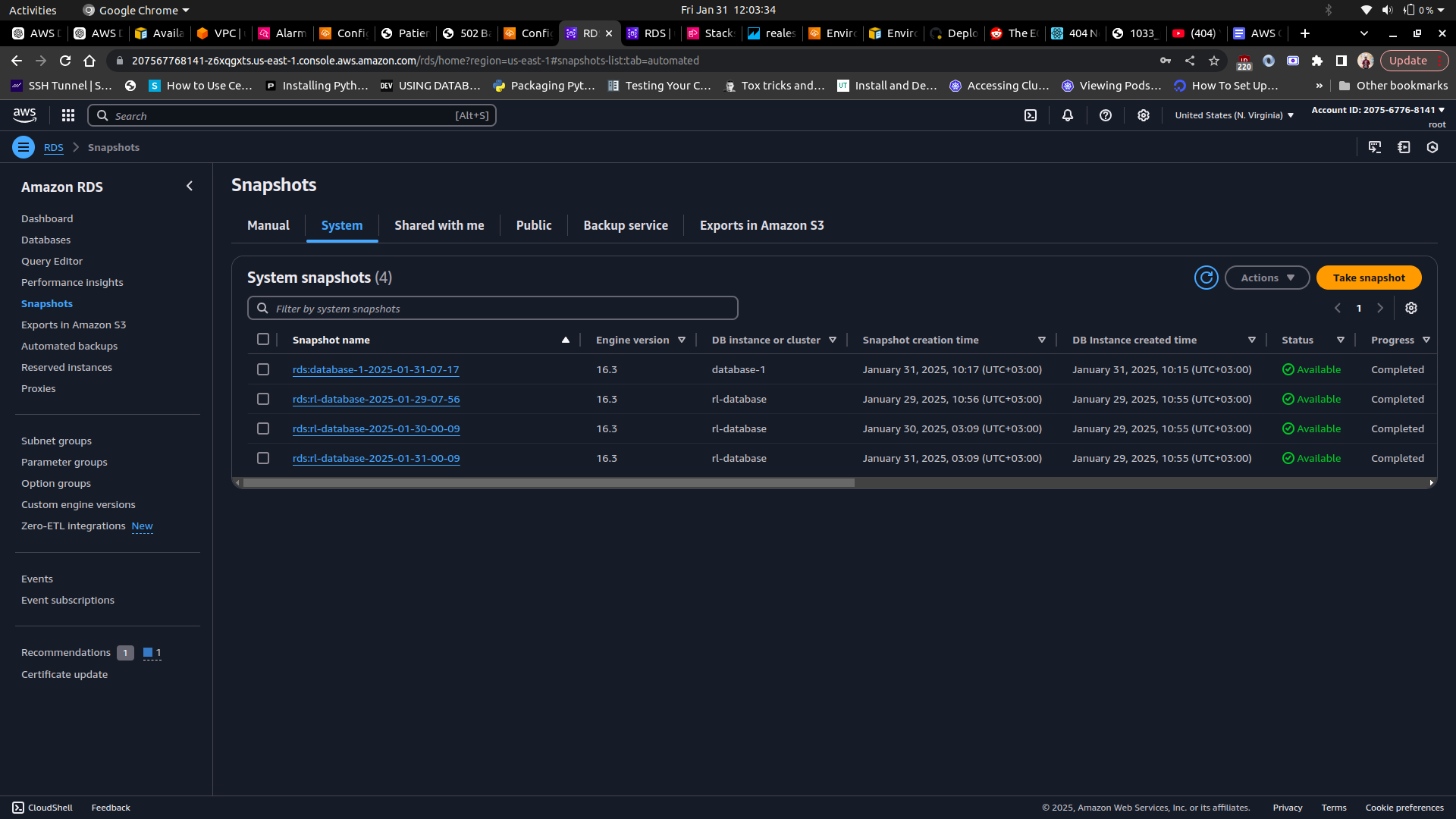
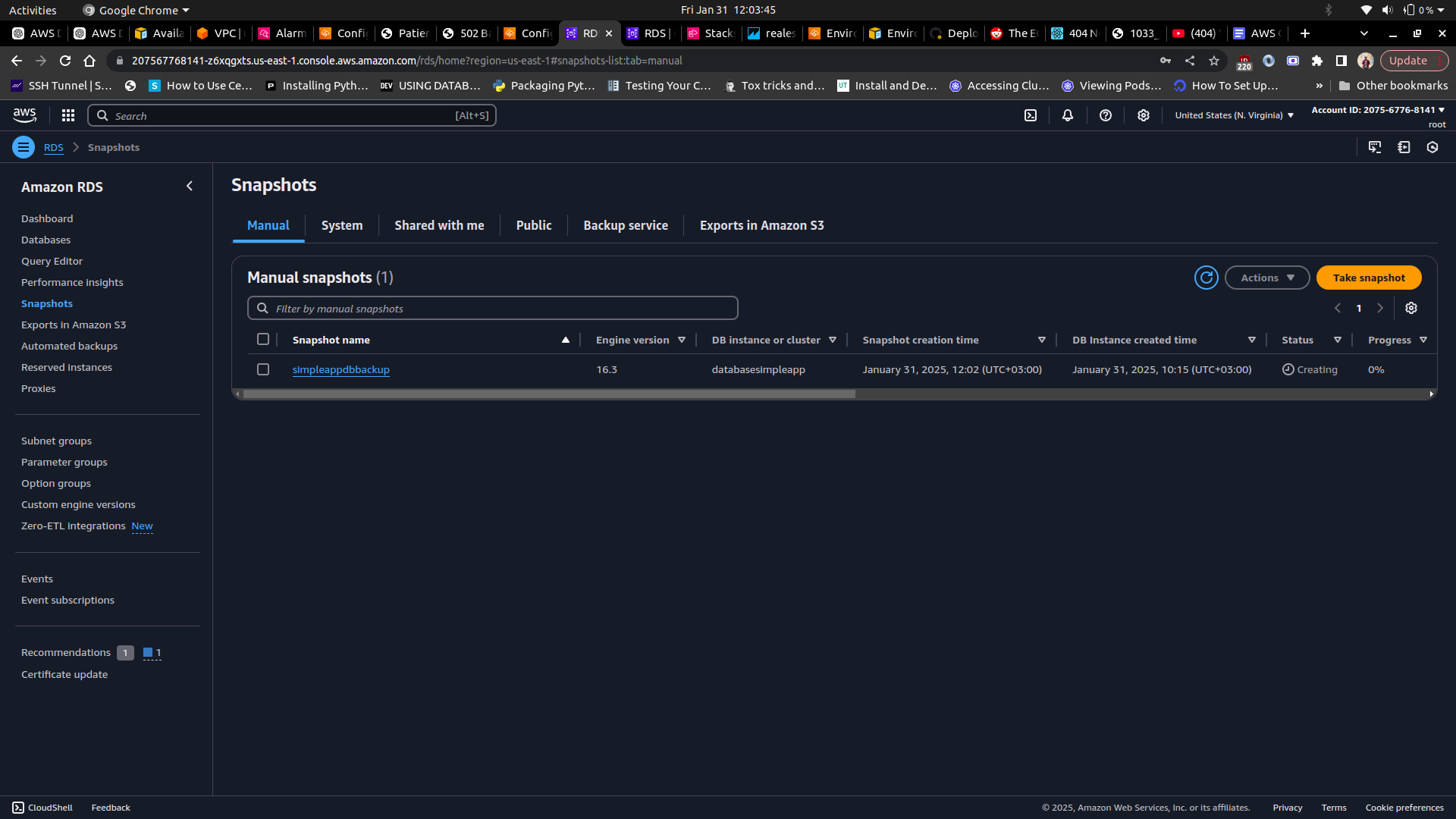
The web application has been dockerized and is hosted on an Elastic Beanstalk environment with a single container docker. The environment has been provisioned with environment variables required by the web application and for PostgreSQL connection. It has also been configured on two public subnets to ensure efficient scalability and high availability of the web application. The web application security group has been attached to this environment in order to restrict database access. The web application can be accessed [here](http://simple-web-app1.us-east-1.elasticbeanstalk.com/patients/). The images below demonstrate the environment configuration:



**3.4 Amazon RDS (PostgreSQL) Configuration**

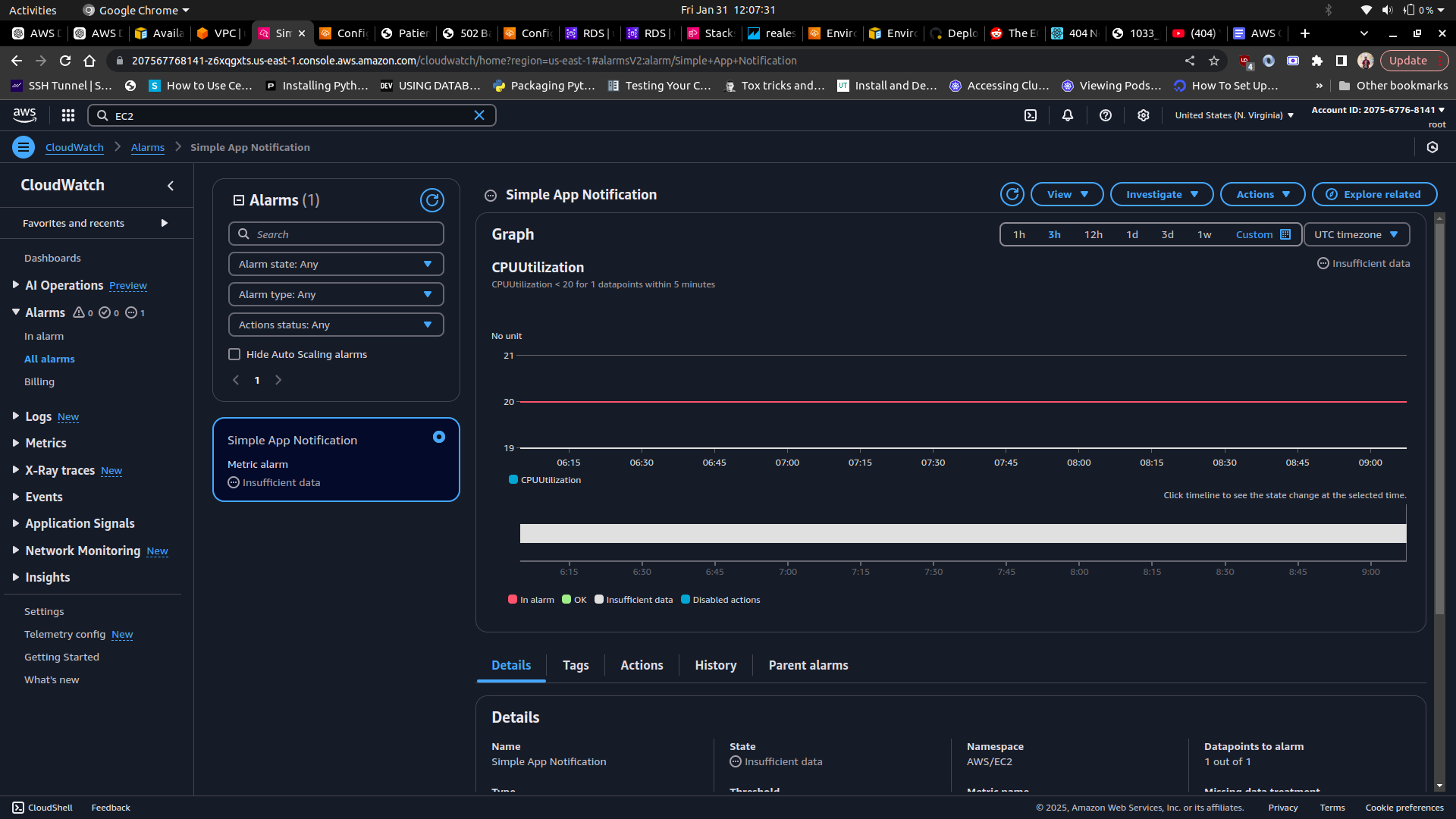
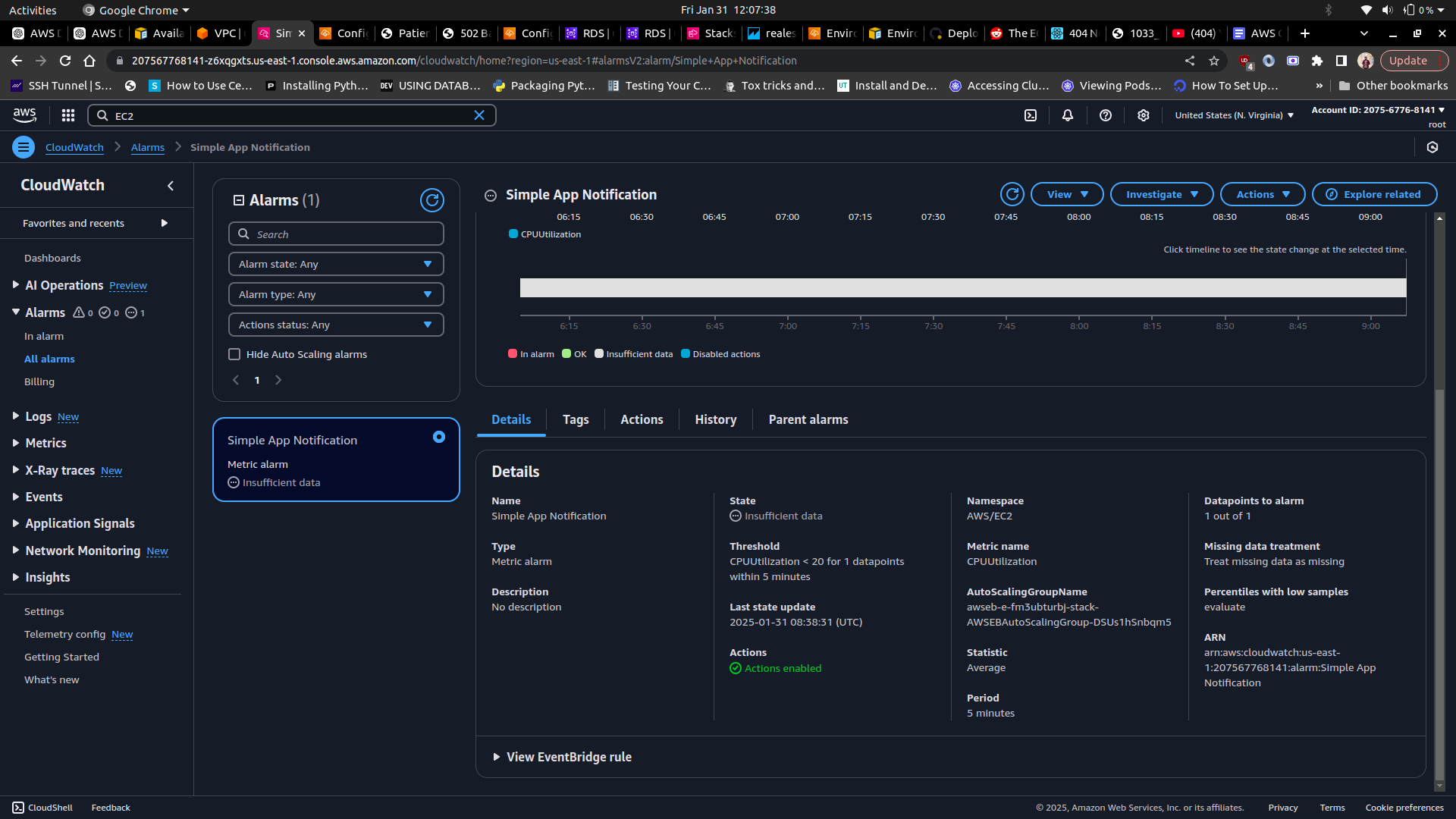
Amazon RDS has been used in the infrastructure to create a PostgreSQL database. The configured database features Multi-AZ deployment to ensure high availability. The database has been configured inside the private subnets in the VPC to prevent direct internet access. As part of the configuration, automatic backups and snapshots have been enabled for disaster recovery.

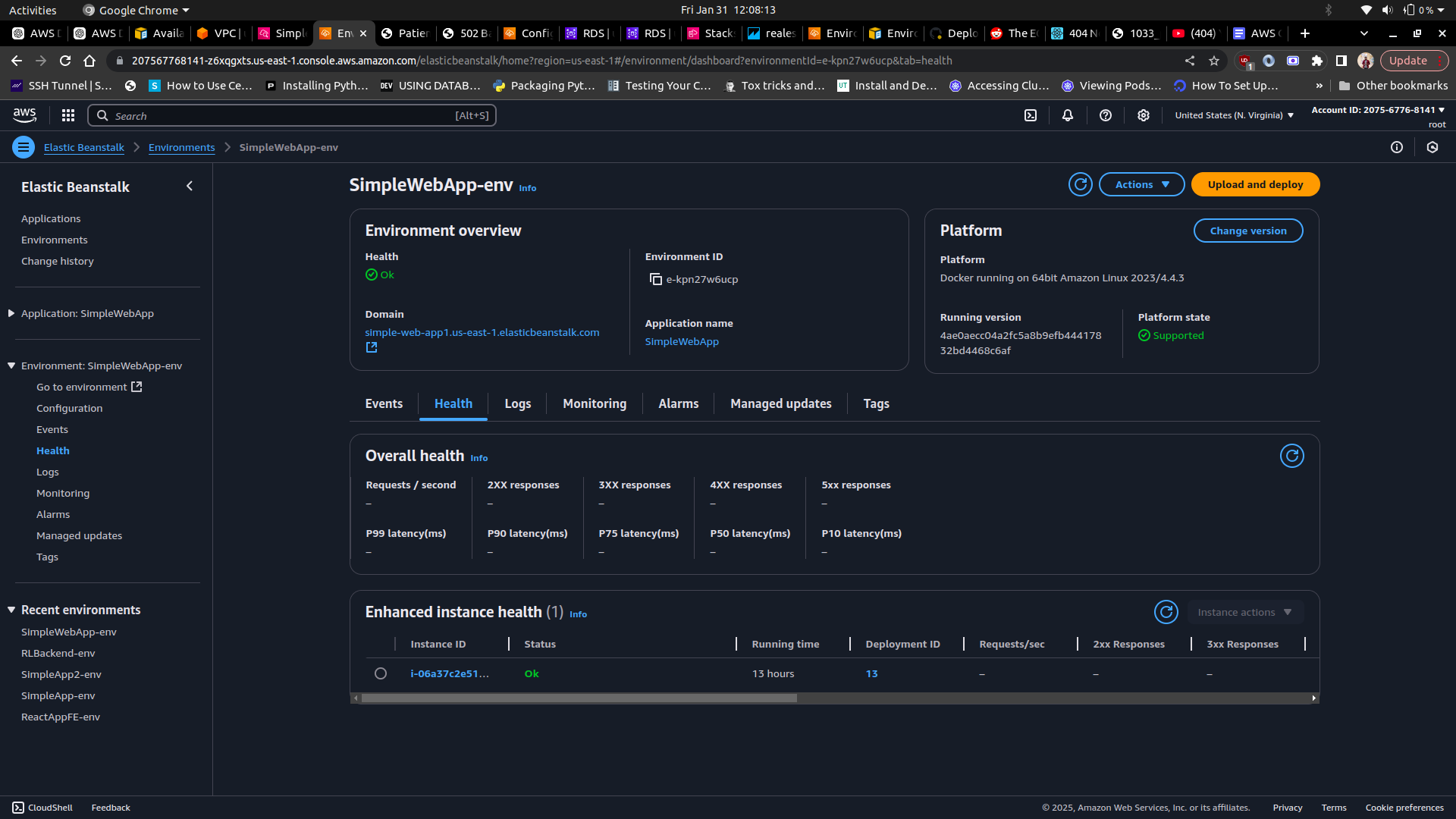
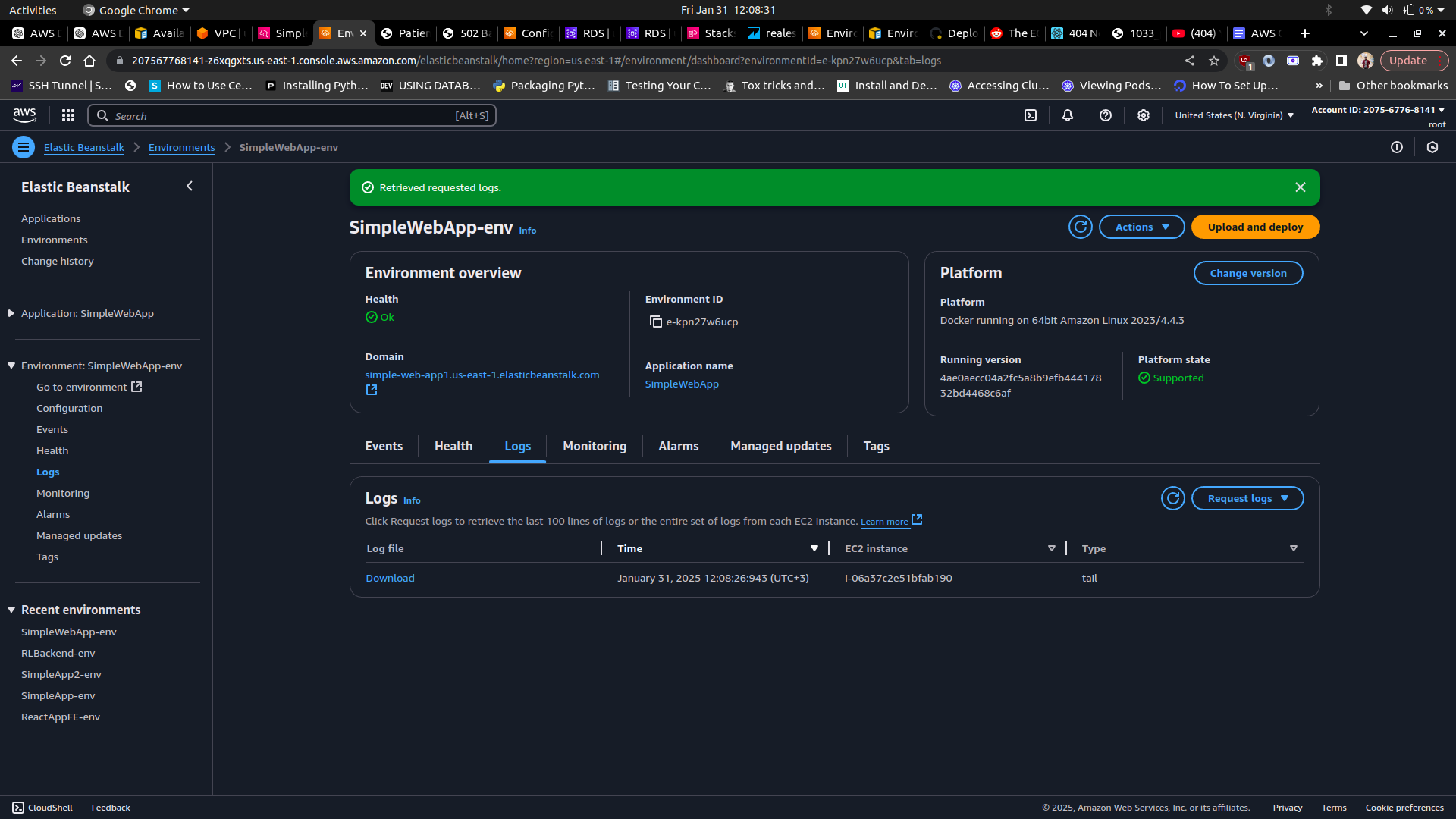
The images below demonstrate an overview of the configuration done:



**3.6. Logging and Monitoring**

The infrastructure leverages AWS CloudWatch logs for both the web application environments. Health monitoring has been enabled on the Elastic Beanstalk environments in order to keep track of the health state of the instances. Incident management and event notifications have been set up using Amazon Simple Notification Service (SNS), which allows for the automatic sending of email notifications when the EC2 instances scale up or down. The images below depict this representation:





**3.7. CloudFormation**

The entire infrastructure setup has been configured using AWS Cloudformation’s Infrastructure as Code (IaC). The image below highlights an overview of the entire infrastructure:



4**. Future Considerations**

The current infrastructure setup is able to provision the web application in a scalable manner and can guarantee high availability to its users. However, more can be done to enhance the scalability, security and maintainability of the application. Future improvements that can be implemented on the infrastructure include but are not limited to:

1. Implementation of AWS Web Application Firewall to protect against common web attacks.
2. Utilization of Amazon Aurora to further optimize the database in order to ensure better scalability and automated failover.
3. Set up tests on the application codes and configure CodePipeline for a more robust deployment automation.
4. Implement multi-region failover with Route 53 DNS failover in order to ensure a higher availability of the web application.

**5. Appendix**

Link to the web application: [simple-web-app](http://simple-web-app1.us-east-1.elasticbeanstalk.com/patients/)