**AWS CONFIGURATION - PYTHON AND REACT WEB APPLICATION**

1. **INTRODUCTION**

This documentation outlines the step-by-step configuration and deployment process of a minimal Real Estate Listing Web Application on the Amazon Web Services platform. The web application consists of a Django backend application with a PostgreSQL database and a React frontend for the User Interface. Both of the frontend and backend applications have been deployed using AWS Elastic Beanstalk, RDS for the database and S3 for storage. 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:

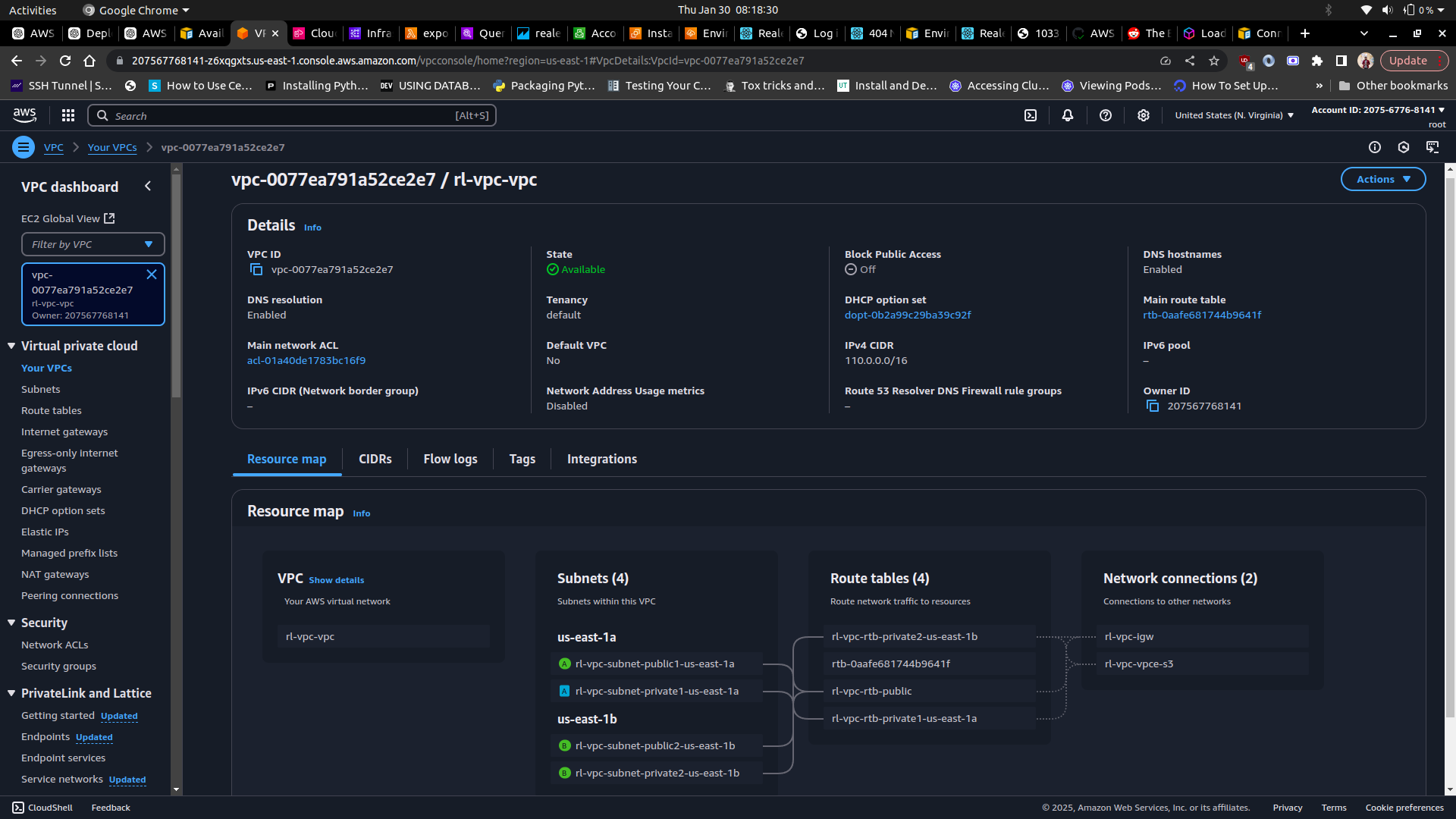
1. **Amazon VPC** - the infrastructure consists of a custom virtual private network with public and private subnets.
2. **Elastic Beanstalk** - the architecture includes a managed service for the deploying of the backend and frontend applications.
3. **Amazon RDS (PostgreSQL)** - the infrastructure also leverages a managed service for the customization of relational databases.
4. **Amazon S3** - a storage service for the frontend assets and user-uploaded images.
5. **AWS IAM** - Identity and Access Management for security roles.
6. **AWS CloudWatch** - for logging and monitoring services.
7. **AWS Auto Scaling** - to ensure high availability and scalability of services.
8. **AWS CloudFormation** - Infrastructure as code to automate deployments.
9. **AWS Athena and Glue** - services for querying and processing logs.

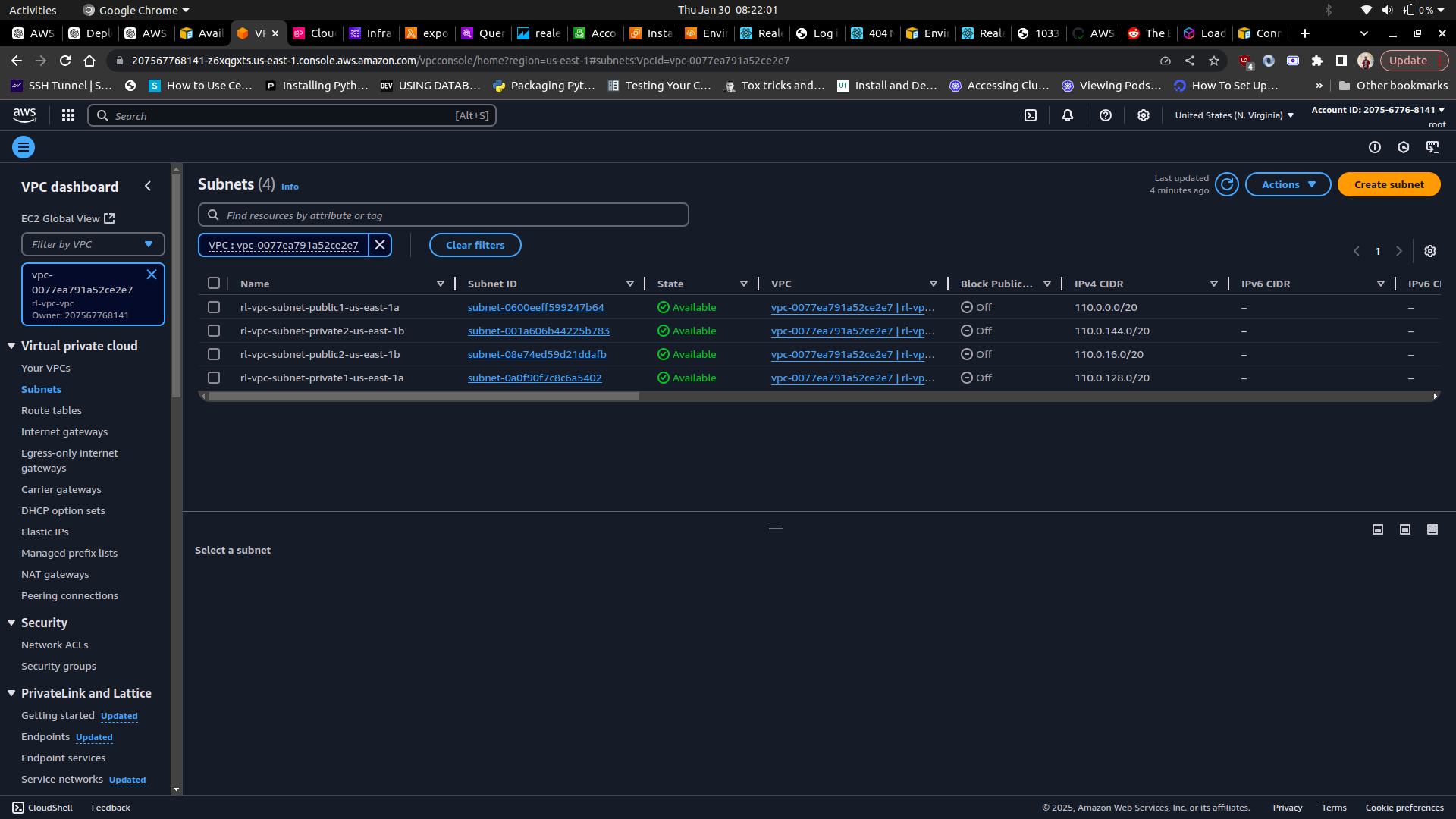
**3. AWS RESOURCE CONFIGURATION**

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

A custom VPC was designed in the architecture that includes:

* Two public subnets for the frontend REACT application and an Elastic Load Balancer.
* Two private subnets for the backend Python (Django) application and the database.
* An Internet Gateway for public access.
* A NAT Gateway to allow private subnets to access the internet securely.



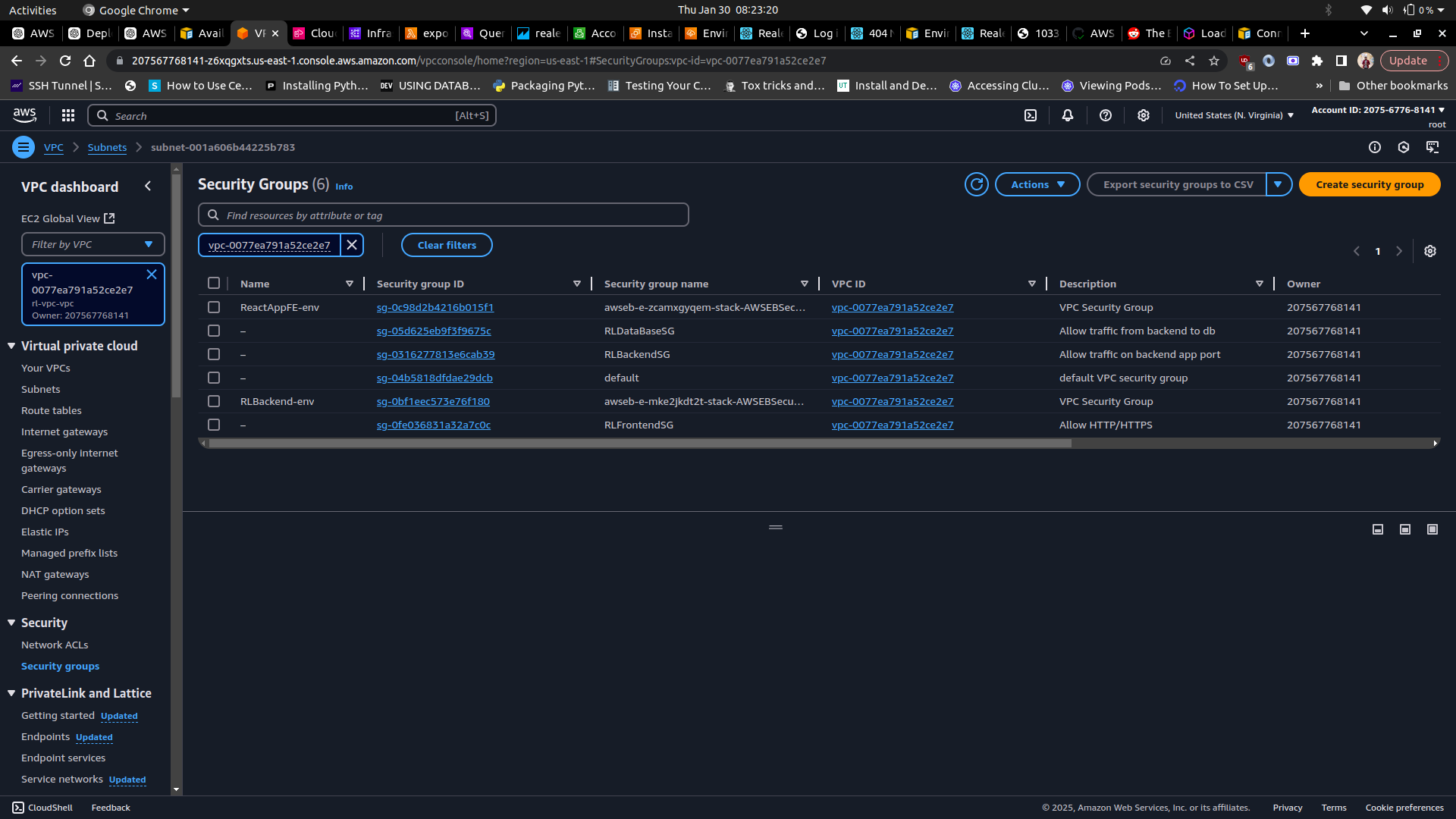


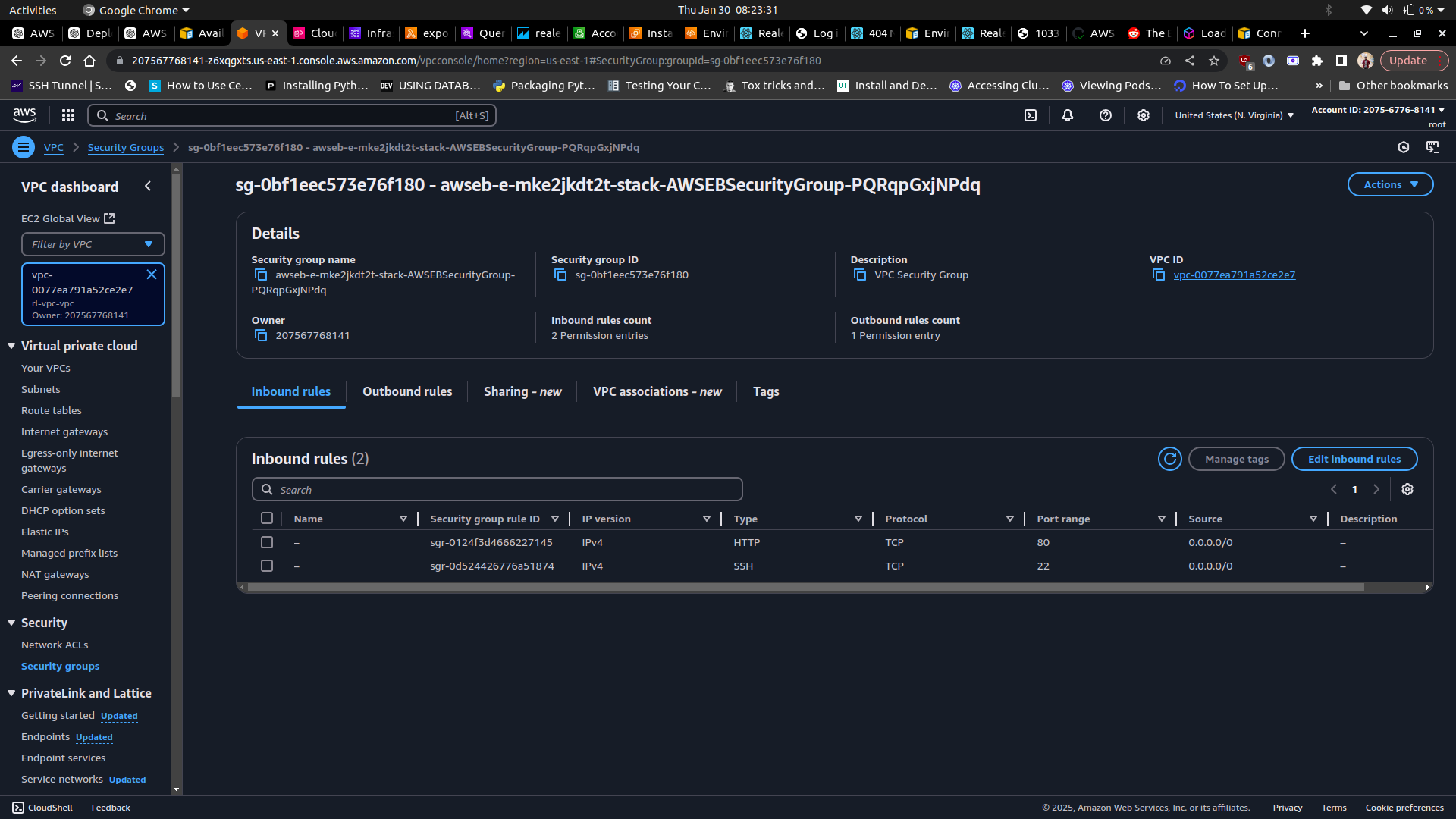


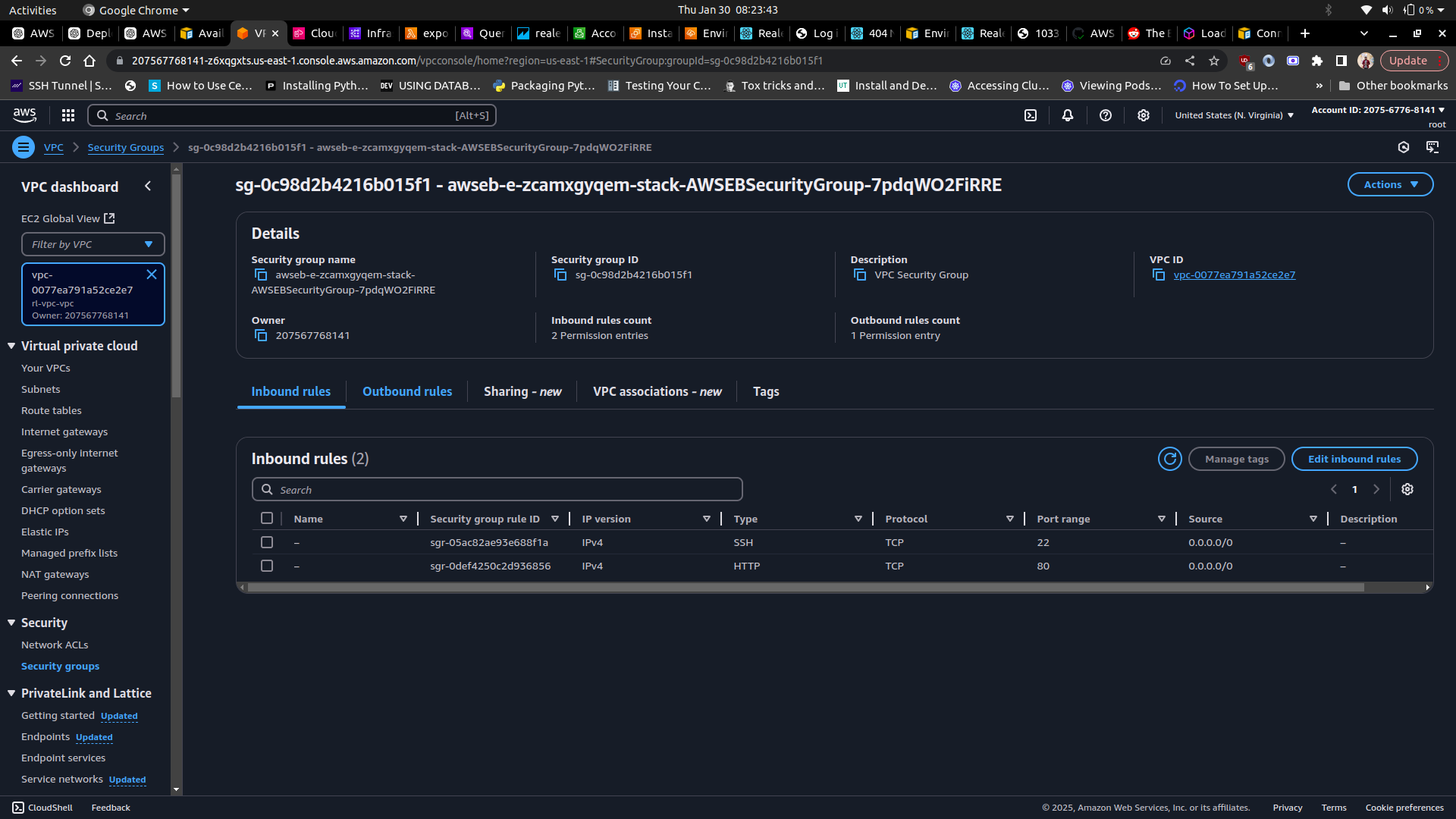
**3.2. Security Groups**

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

* Frontend Security Group - this group allows inbound traffic on port 80/443 from the internet.
* Backend Security Group - this security group allows inbound traffic on port 8000 and 8080 from the frontend security group.
* Database Security Group - this group allows PostgreSQL connections on port 5432 from the backend security group.



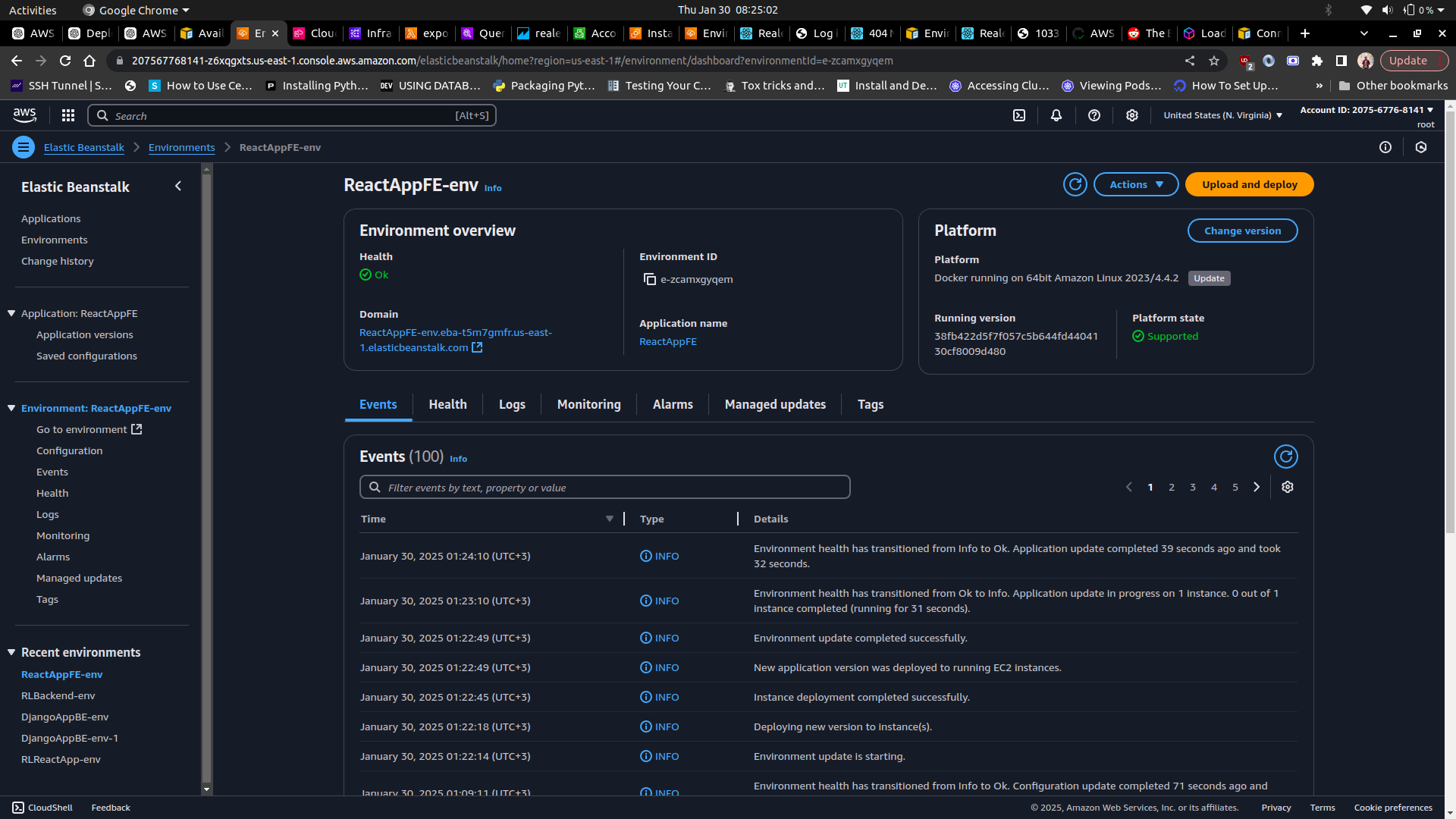


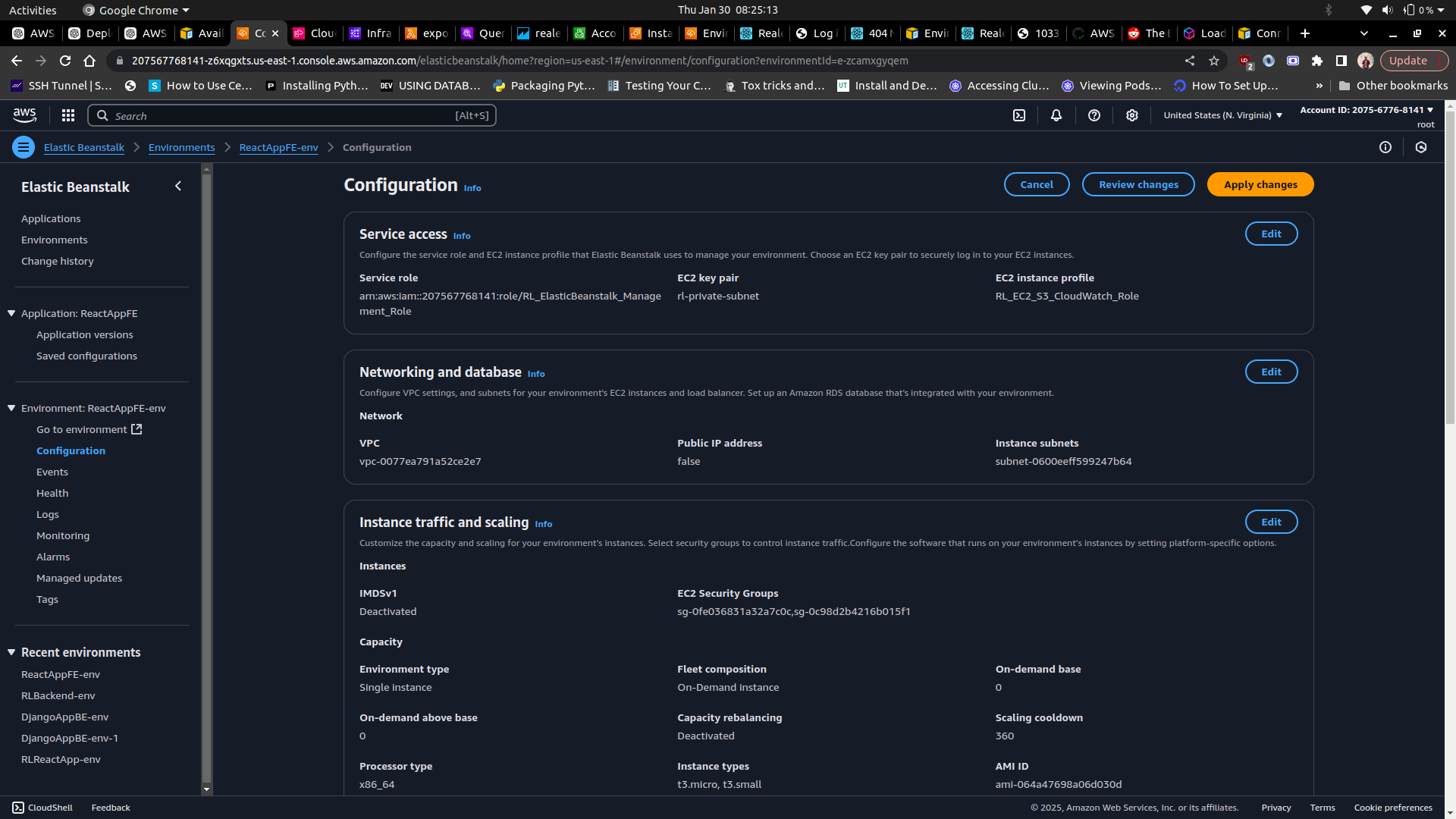


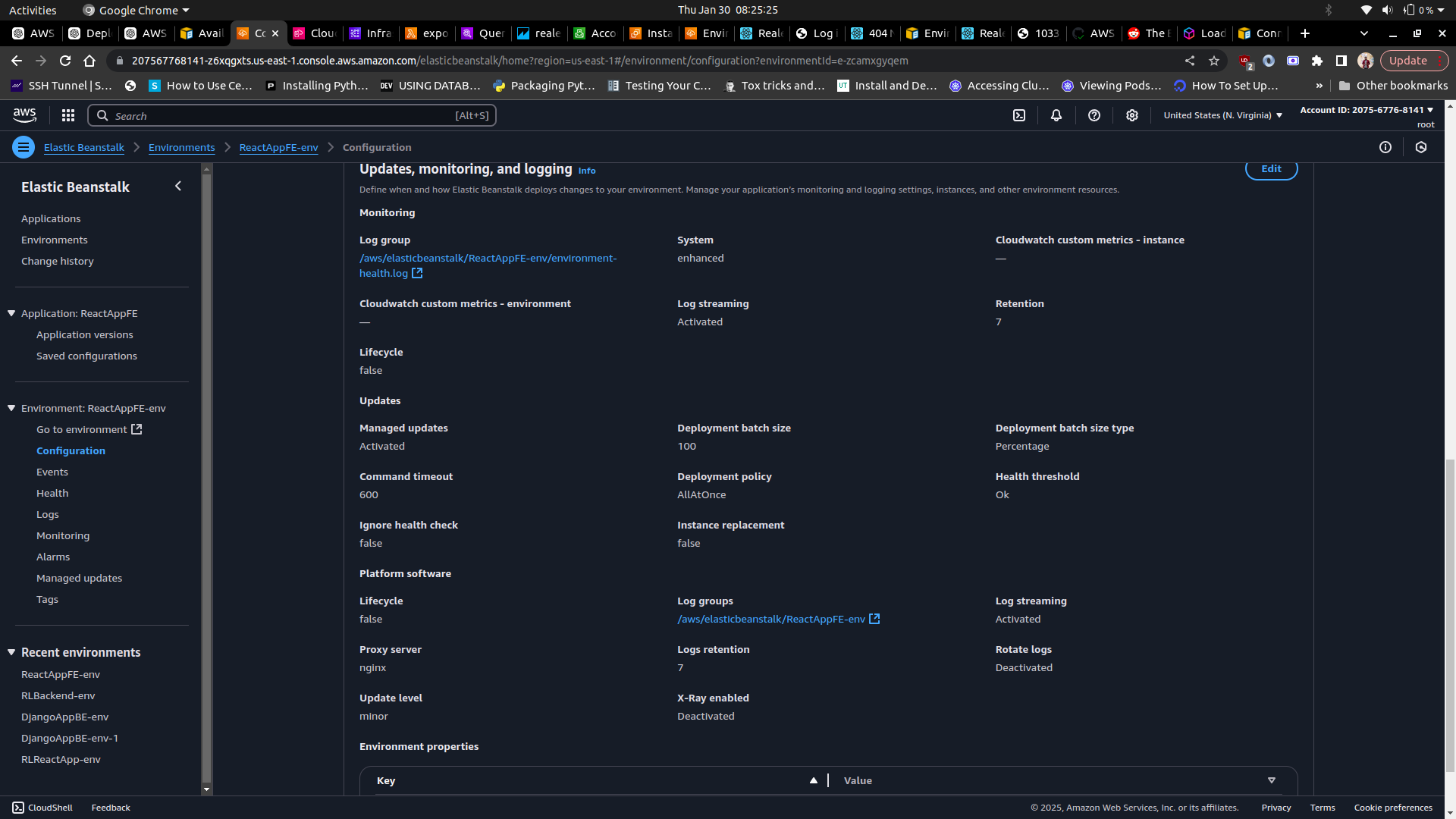
**3.3. Elastic Beanstalk Configuration**

**3.3.1 Frontend Deployment (React)**

The React frontend application has been dockerized first, then configured to automatically deploy to an Elastic Beanstalk environment in a single container docker once any code changes are pushed to the repository’s github master branch. Github actions CI/CD pipeline, through a Dockerrun.aws.json file, have been used in this case to automate the deployment. The Elastic Beanstalk environment has been configured with a load balancer and auto scaling group of 2 to 4 instances. The frontend application can be accessed [here](http://reactappfe-env.eba-t5m7gmfr.us-east-1.elasticbeanstalk.com/)

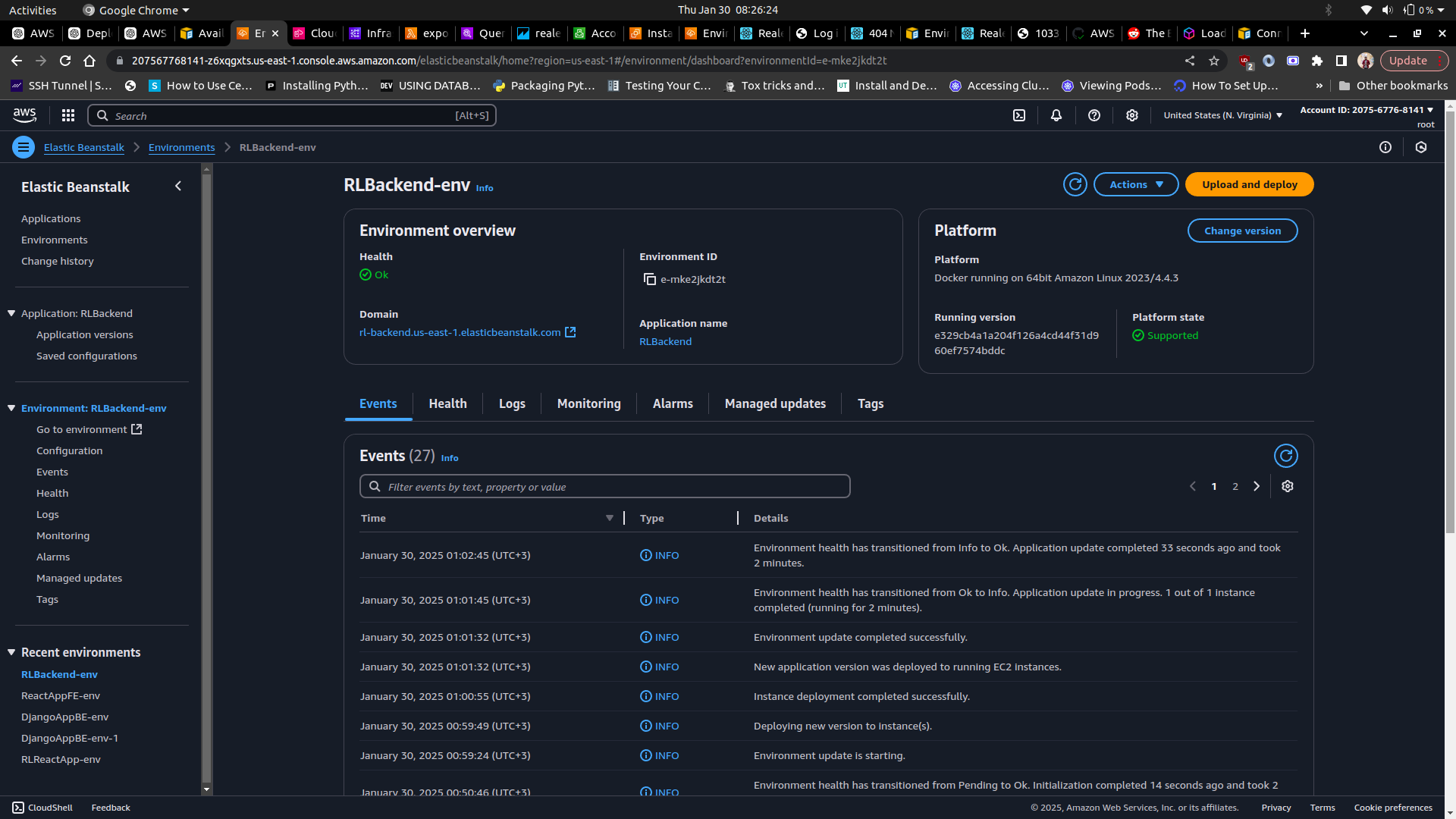




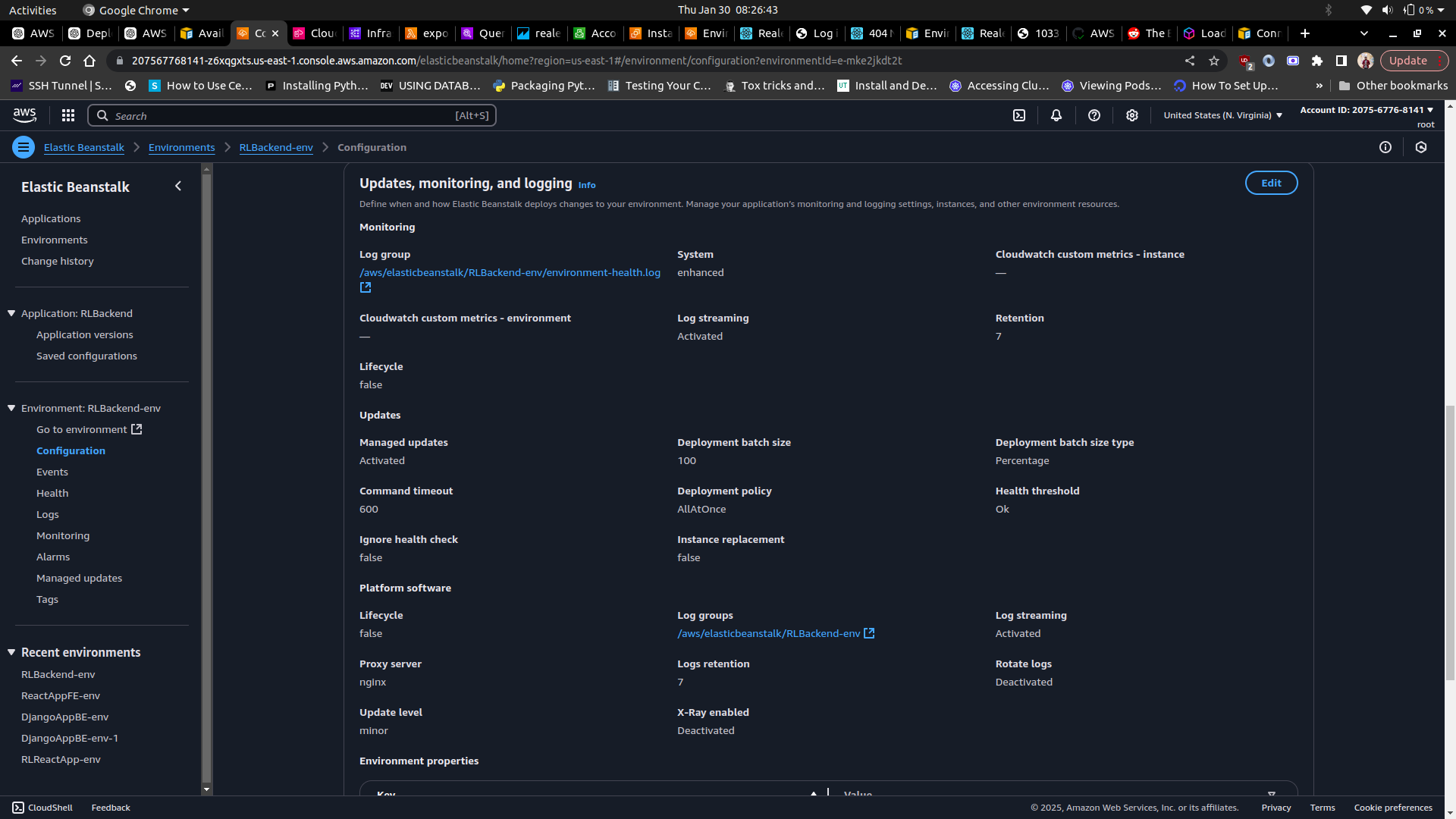


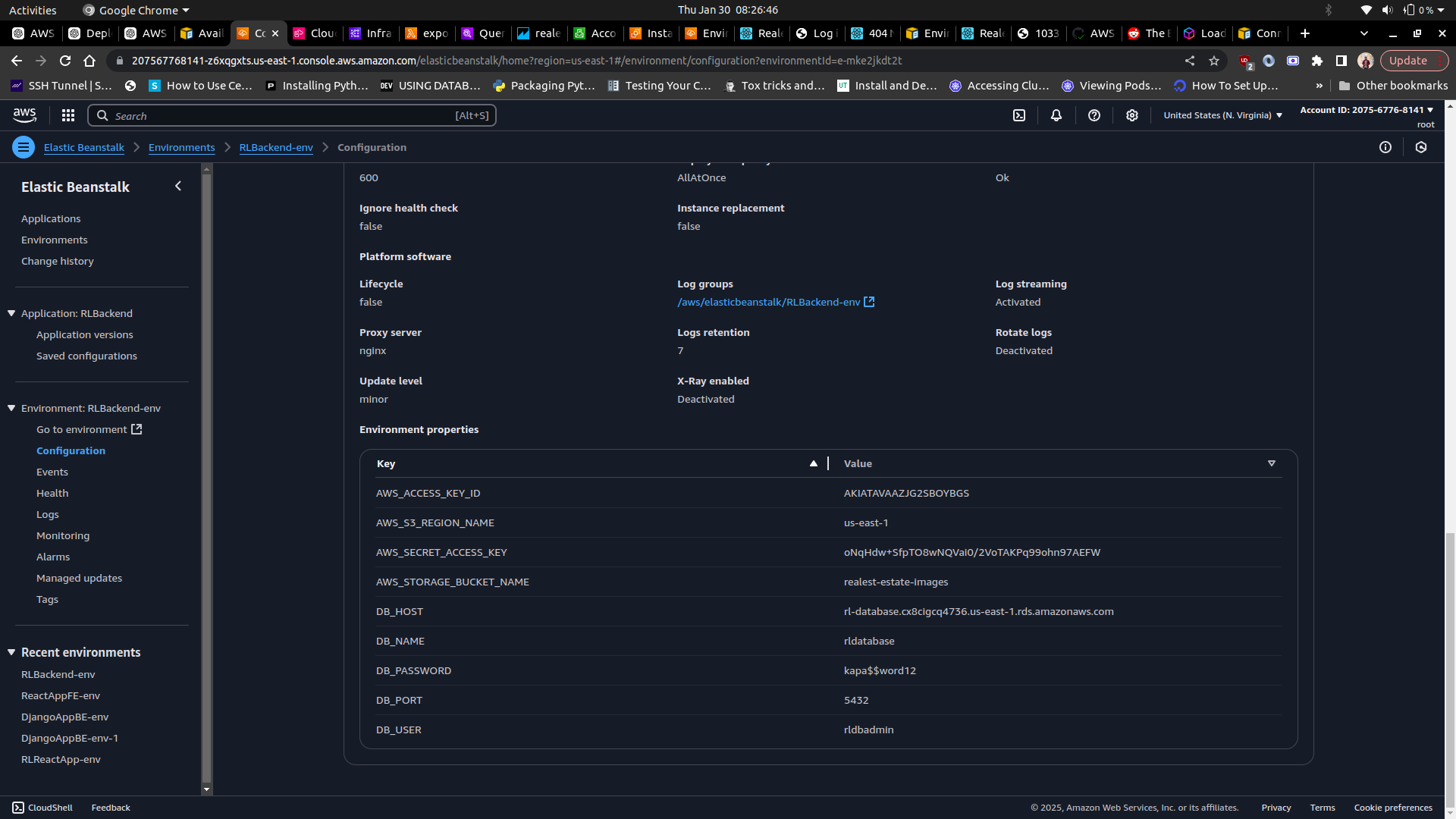
**3.3.2 Backend Deployment (Python/Django)**

The backend Django web application has also been dockerized and is hosted on another Elastic Beanstalk environment with a single container docker. The environment has been provisioned with environment variables required by the backend application and for PostgreSQL connection. The backend security group has been attached to this environment in order to restrict database access. The backend APIs can be accessed [here](http://rl-backend.us-east-1.elasticbeanstalk.com/api/listings/)



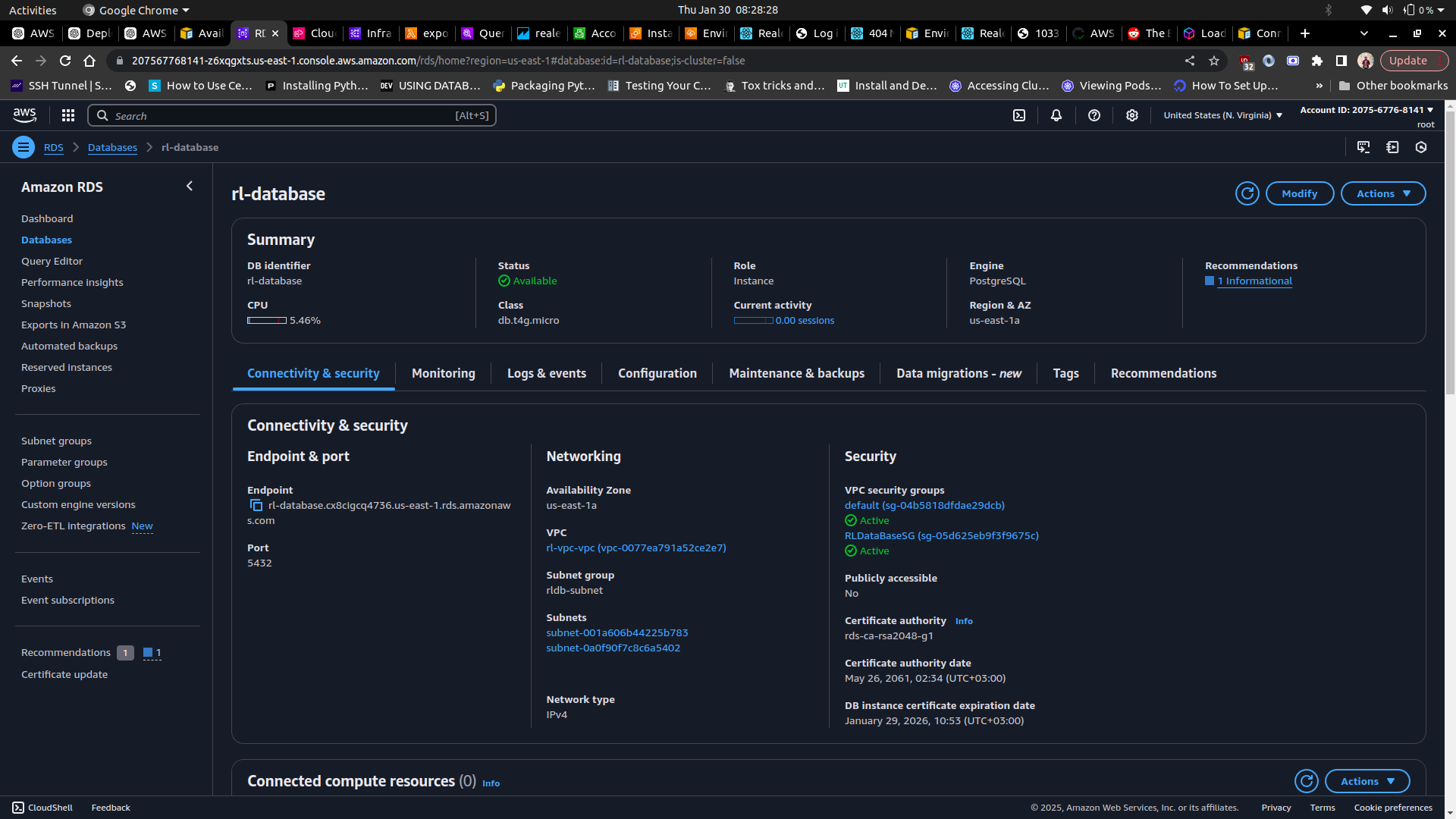


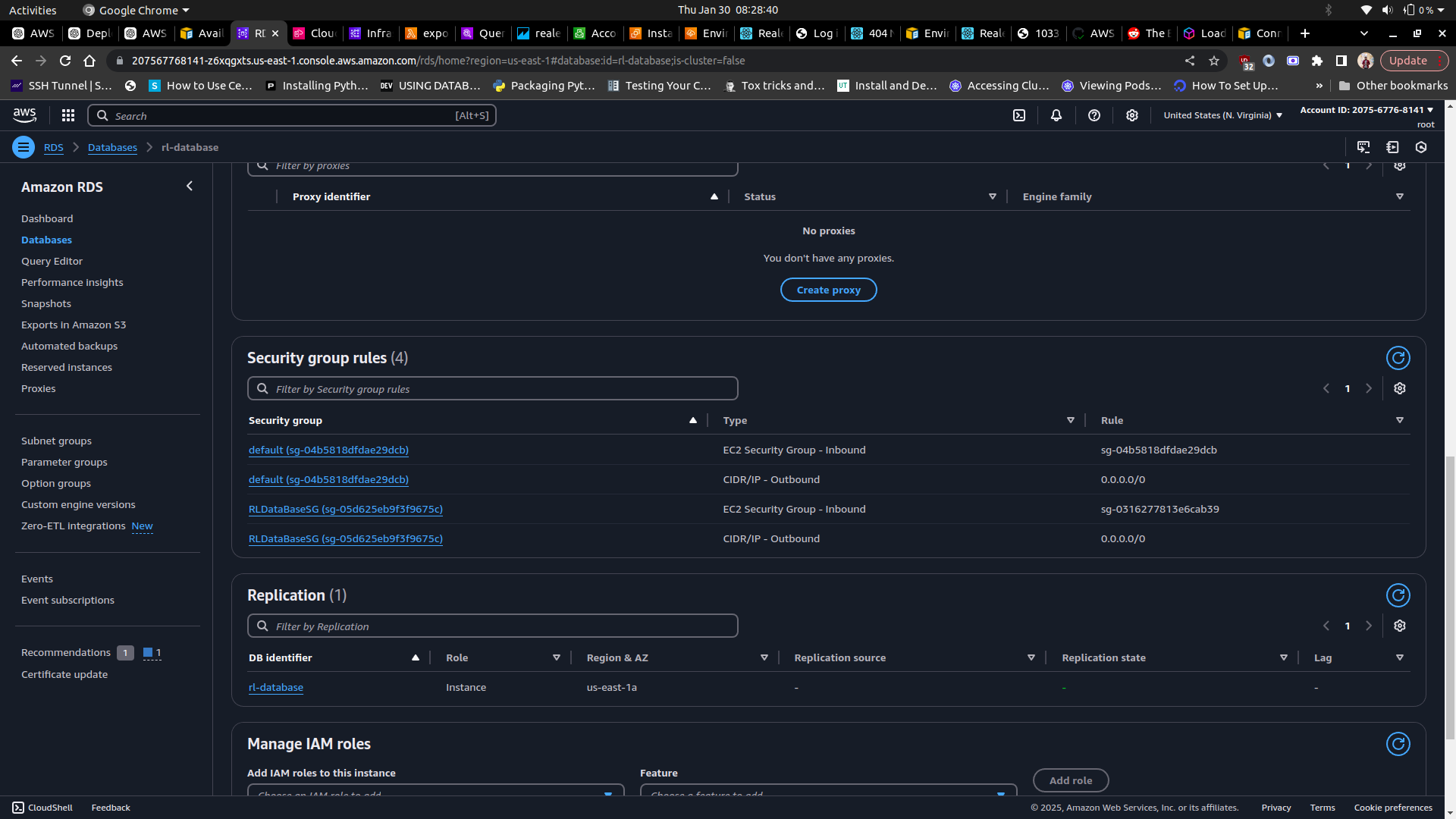




**3.4 Amazon RDS (PostgreSQL) Configuration**

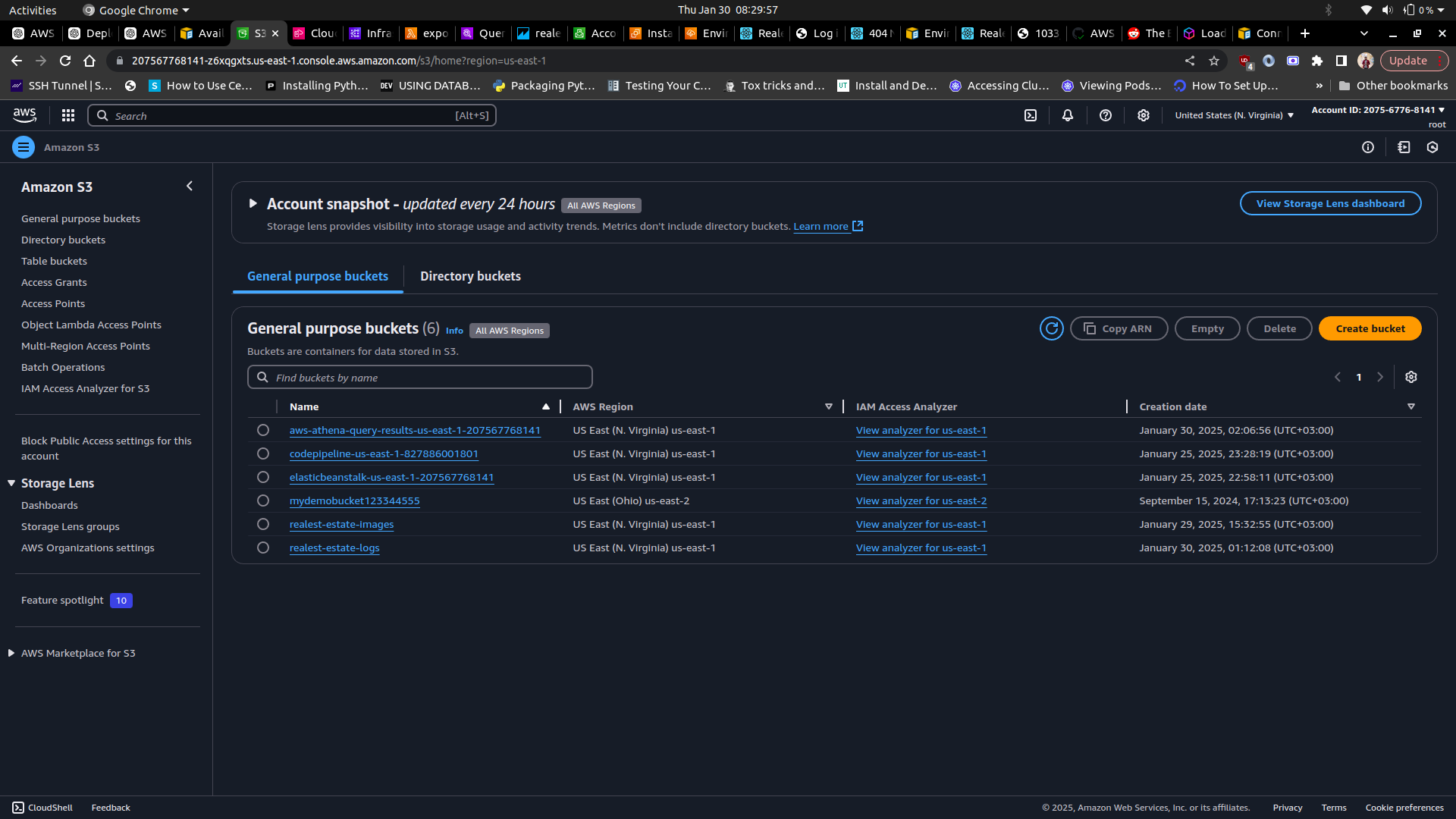
Amazon RDS has been used in the infrastructure to create a PostgreSQL database. The configured database features Multi-AZ deployment to ensure high availability. The database has been configured inside the private subnets in the VPC to prevent direct internet access. As part of the configuration, automatic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

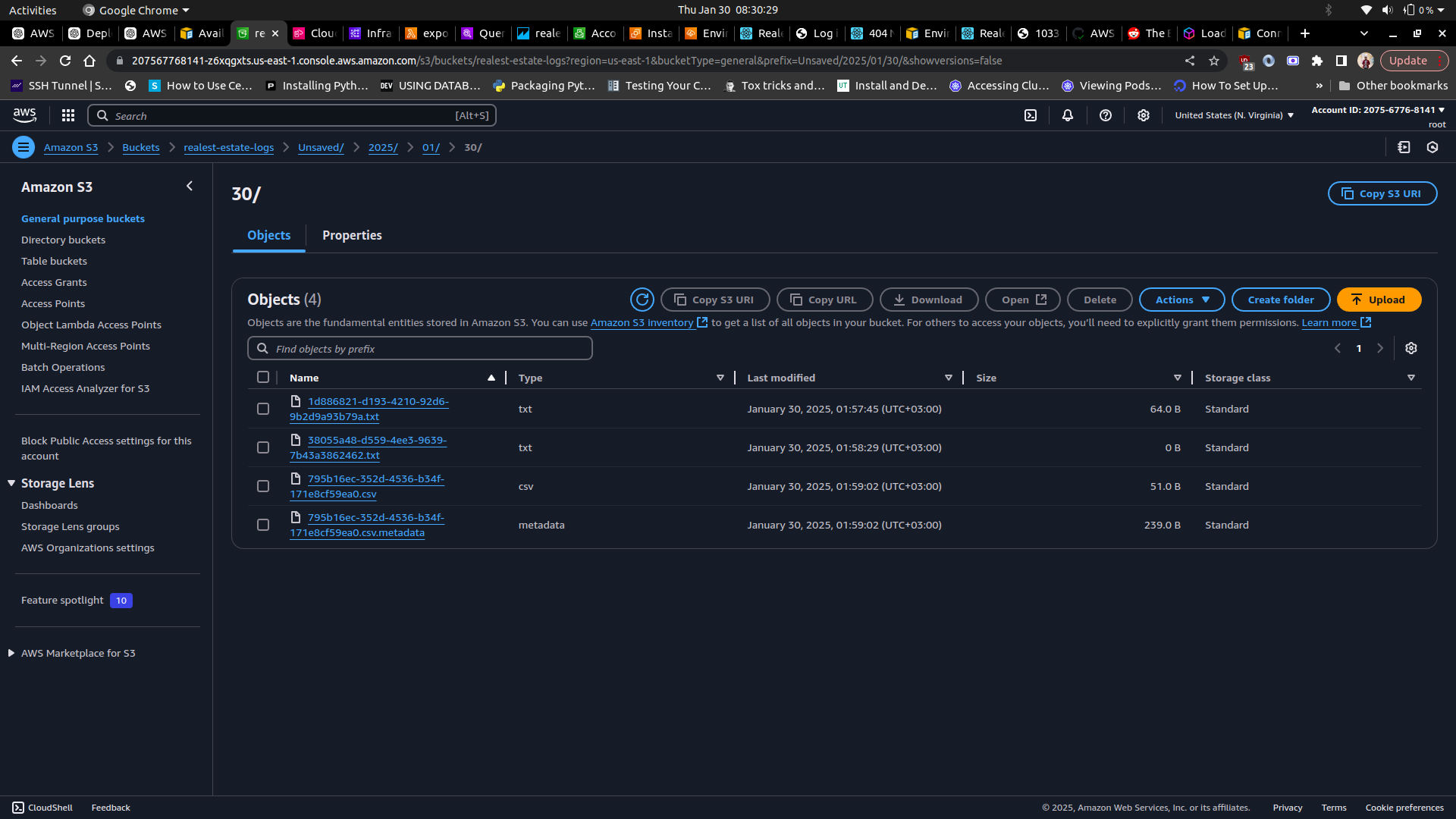


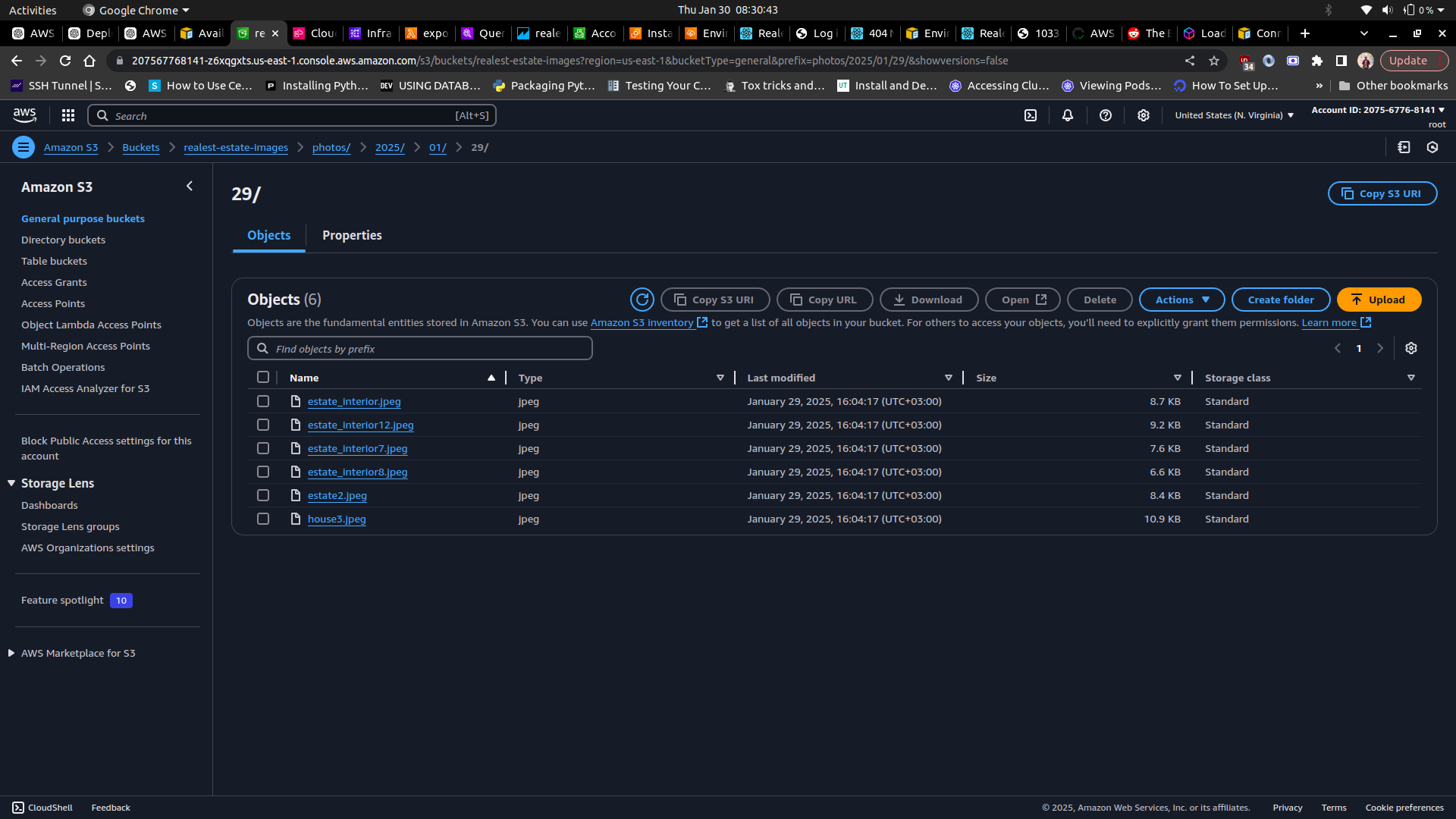


**3.5. Amazon S3 Configuration**

The infrastructure requires that S3 buckets are set up to store images uploaded on the web applications and also a separate bucket to store all the event logs. IAM Roles have also been configured to allow Django to upload images. The S3 configuration also includes a bucket policy that has been set to allow public read access for images.

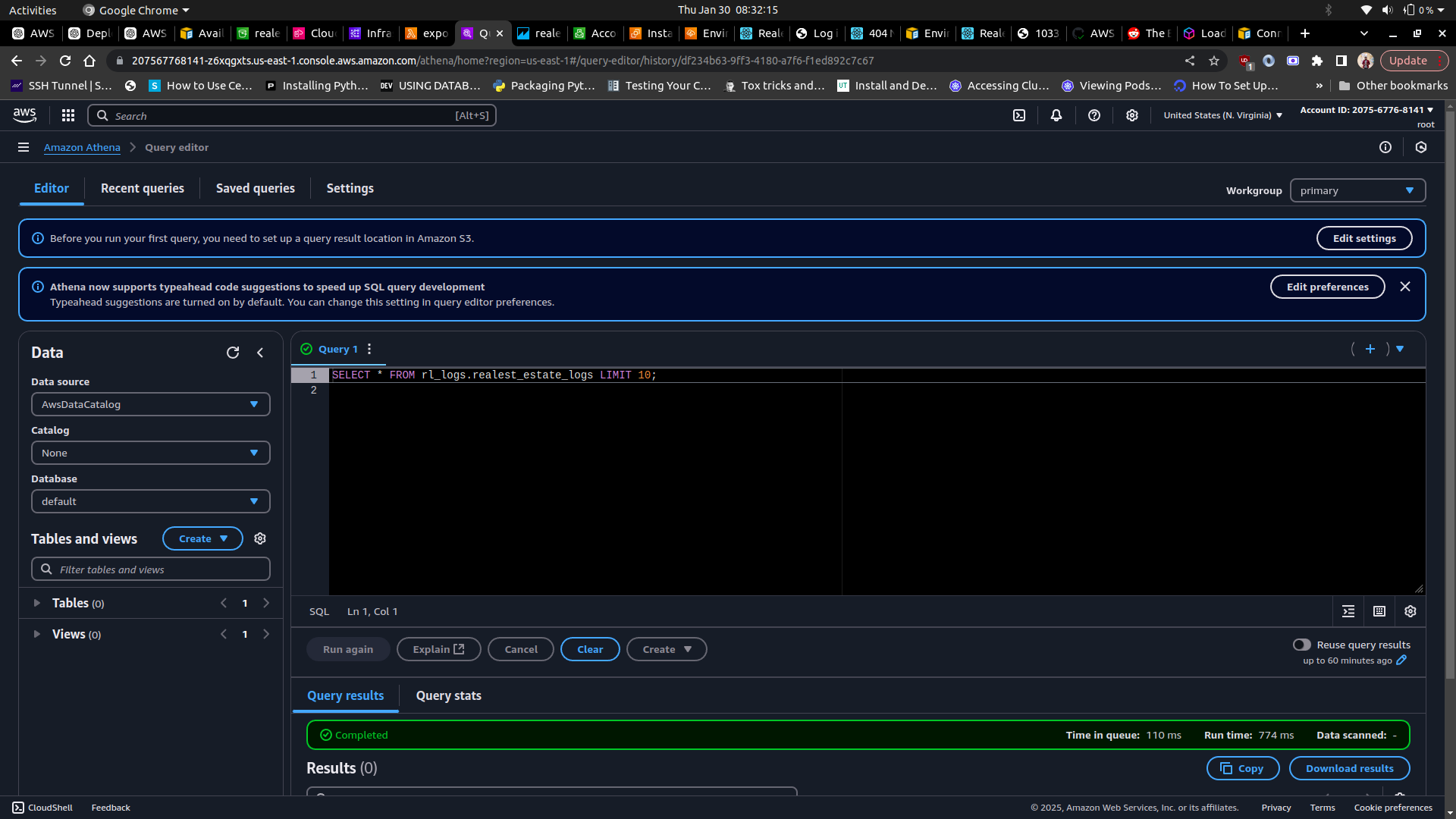


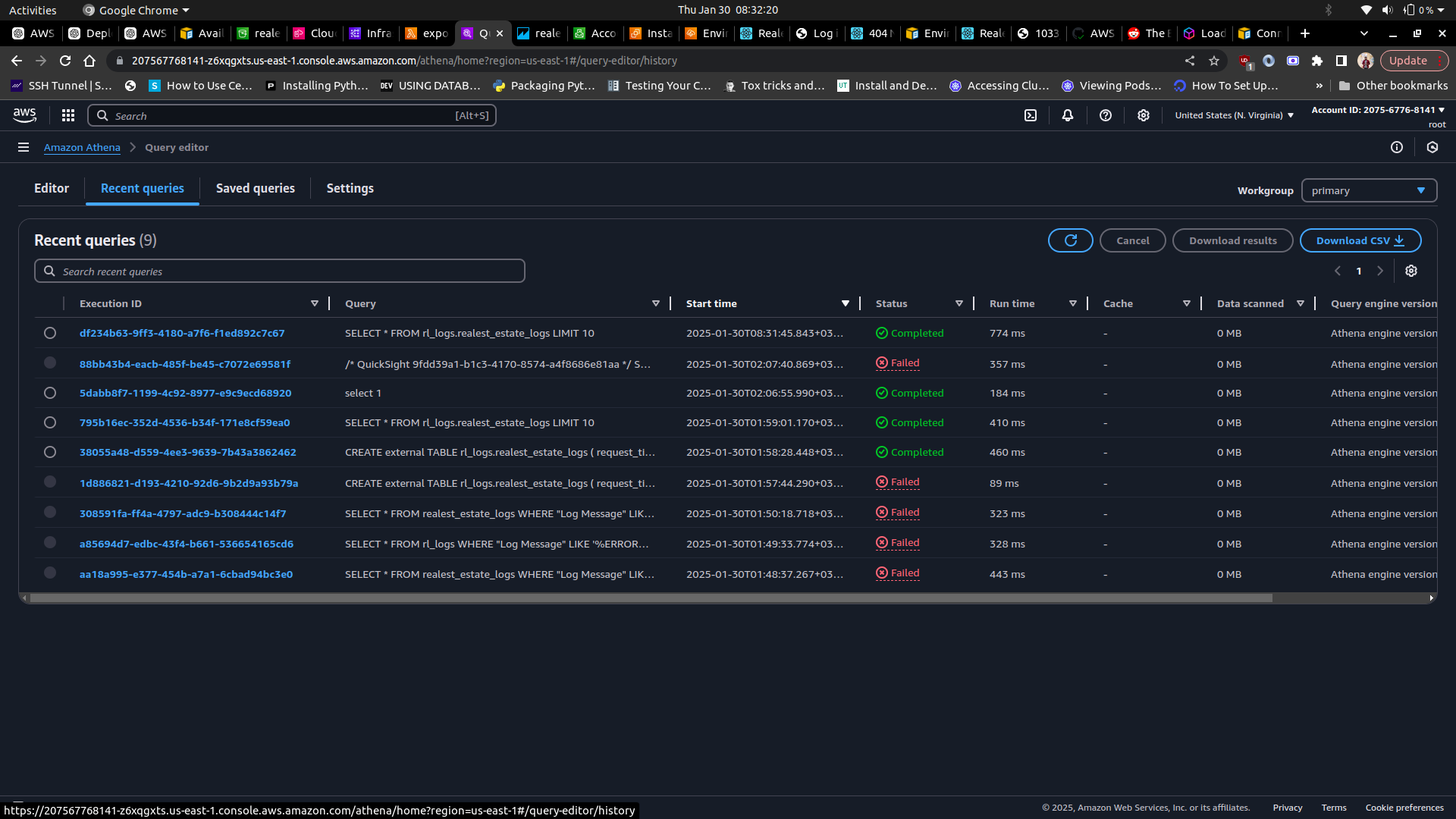


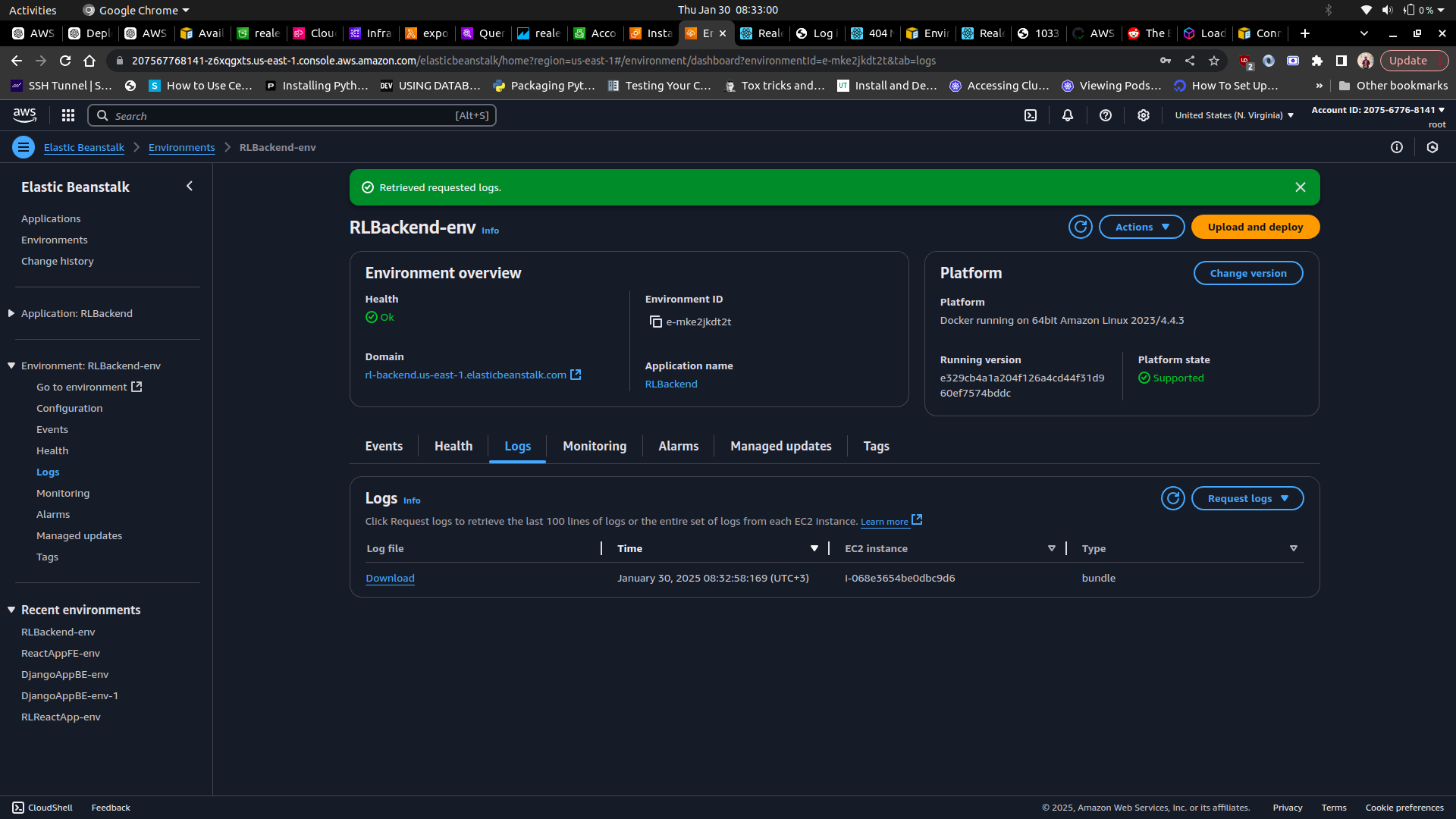


**3.6. Logging and Monitoring**

The infrastructure leverages AWS CloudWatch logs for both the backend and frontend application environments. Health monitoring has been enabled on the Elastic Beanstalk environments in order to keep track of the health state of the instances. The Athena service has also been utilized to analyze logs and automate queries.

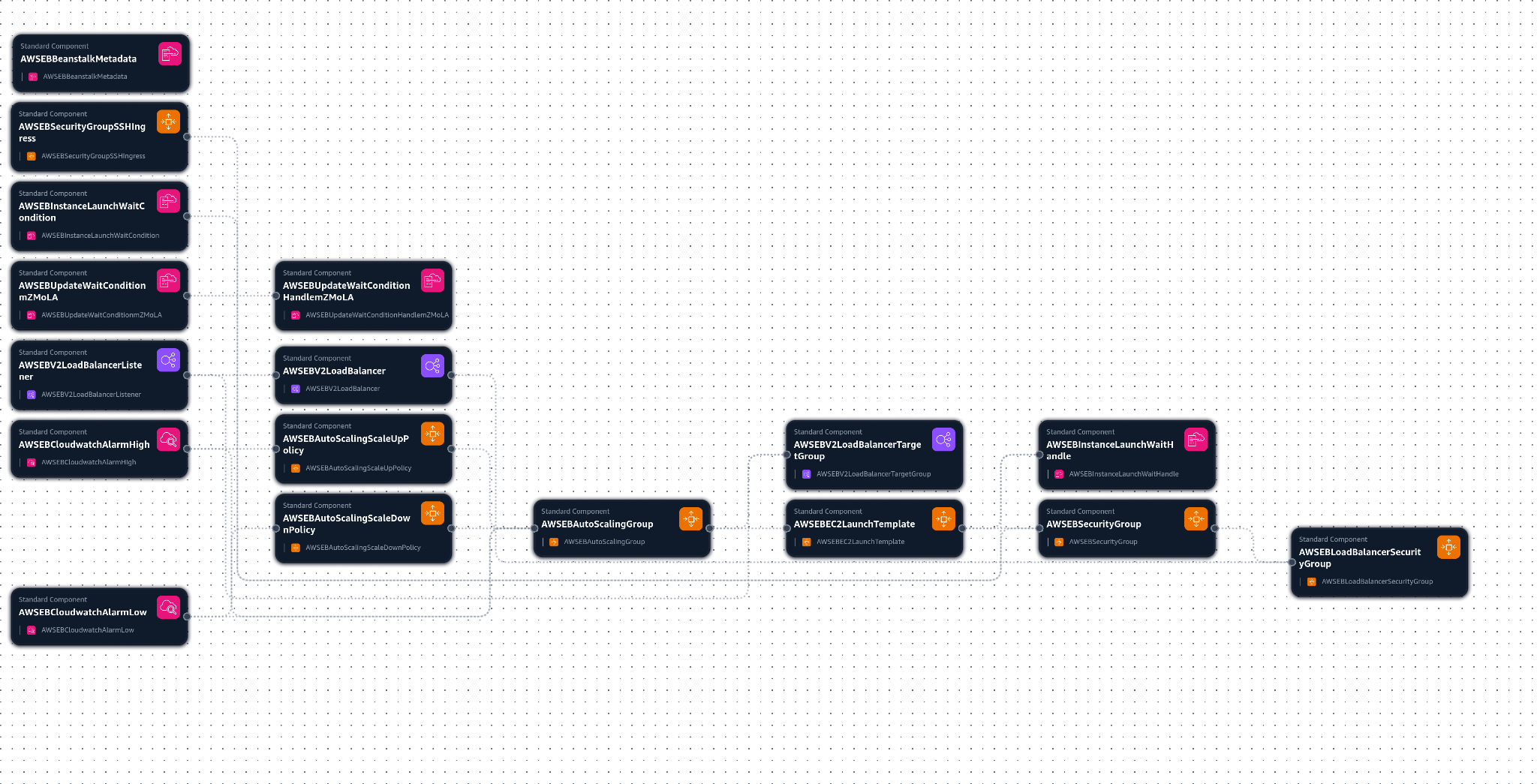






**3.7. CloudFormation**

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**

S3 link to images - [S3 Image Link](https://207567768141-z6xqgxts.us-east-1.console.aws.amazon.com/s3/buckets/realest-estate-images?region=us-east-1&bucketType=general&tab=objects)