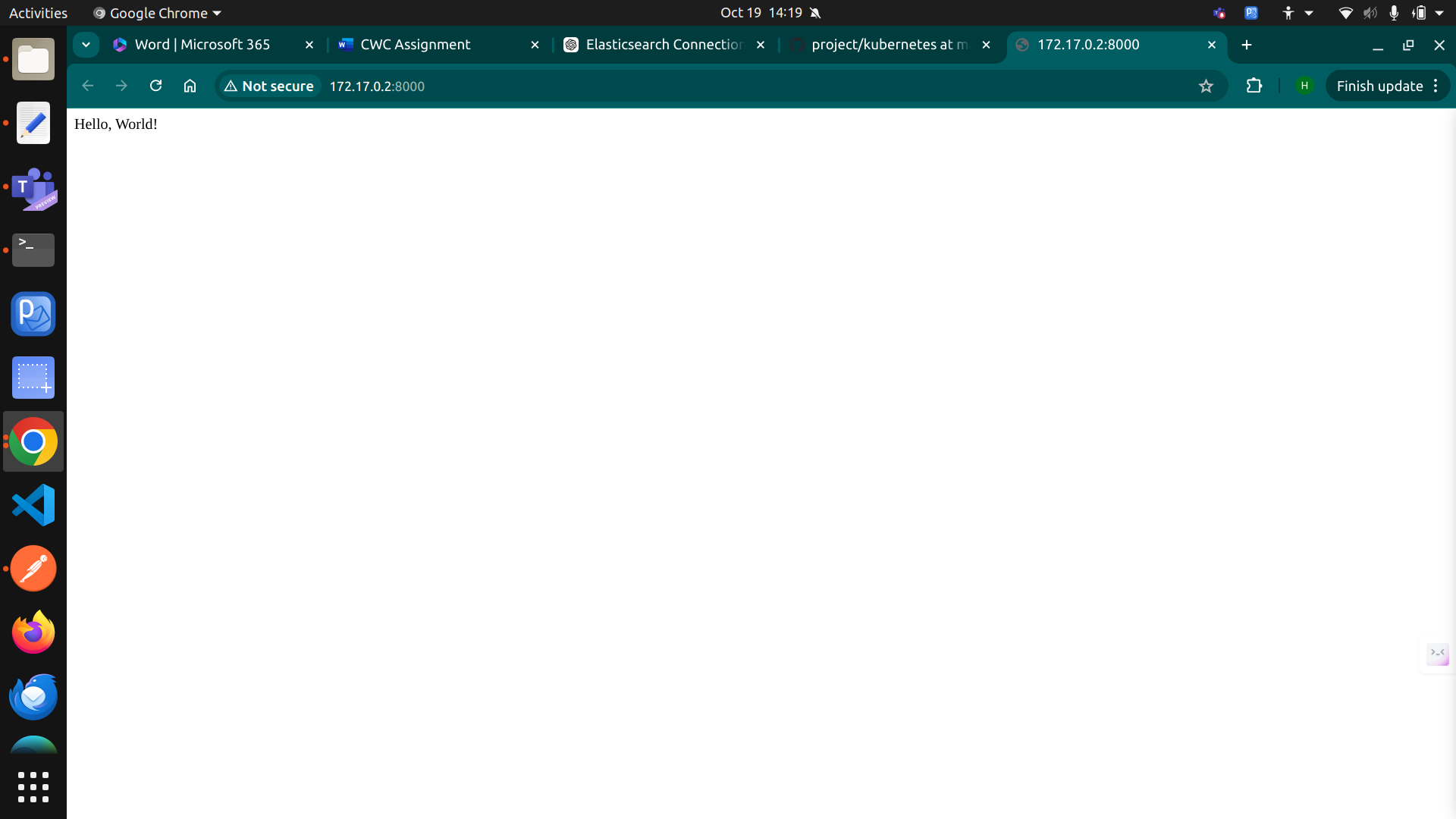
ASSIGNMENT

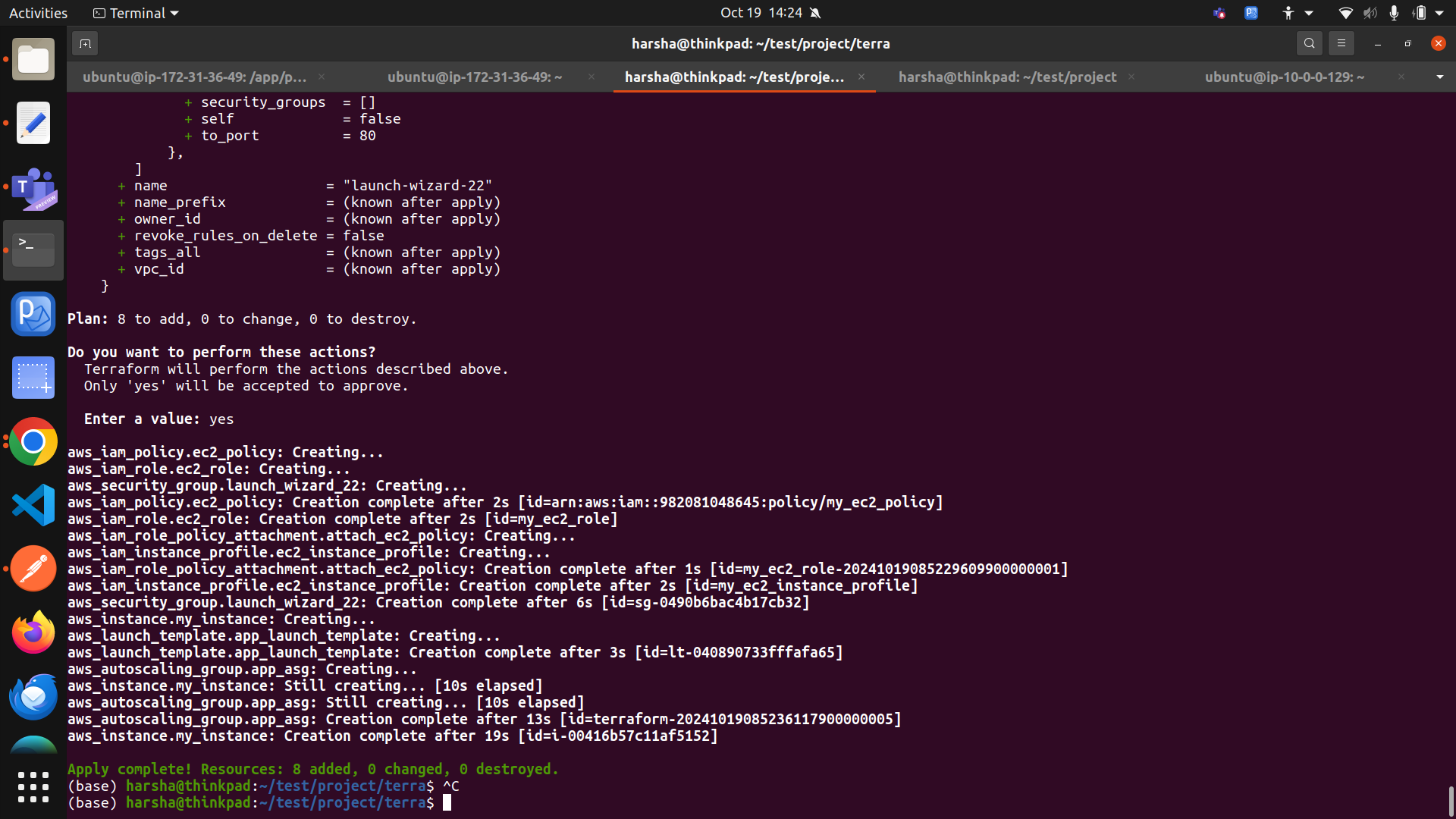
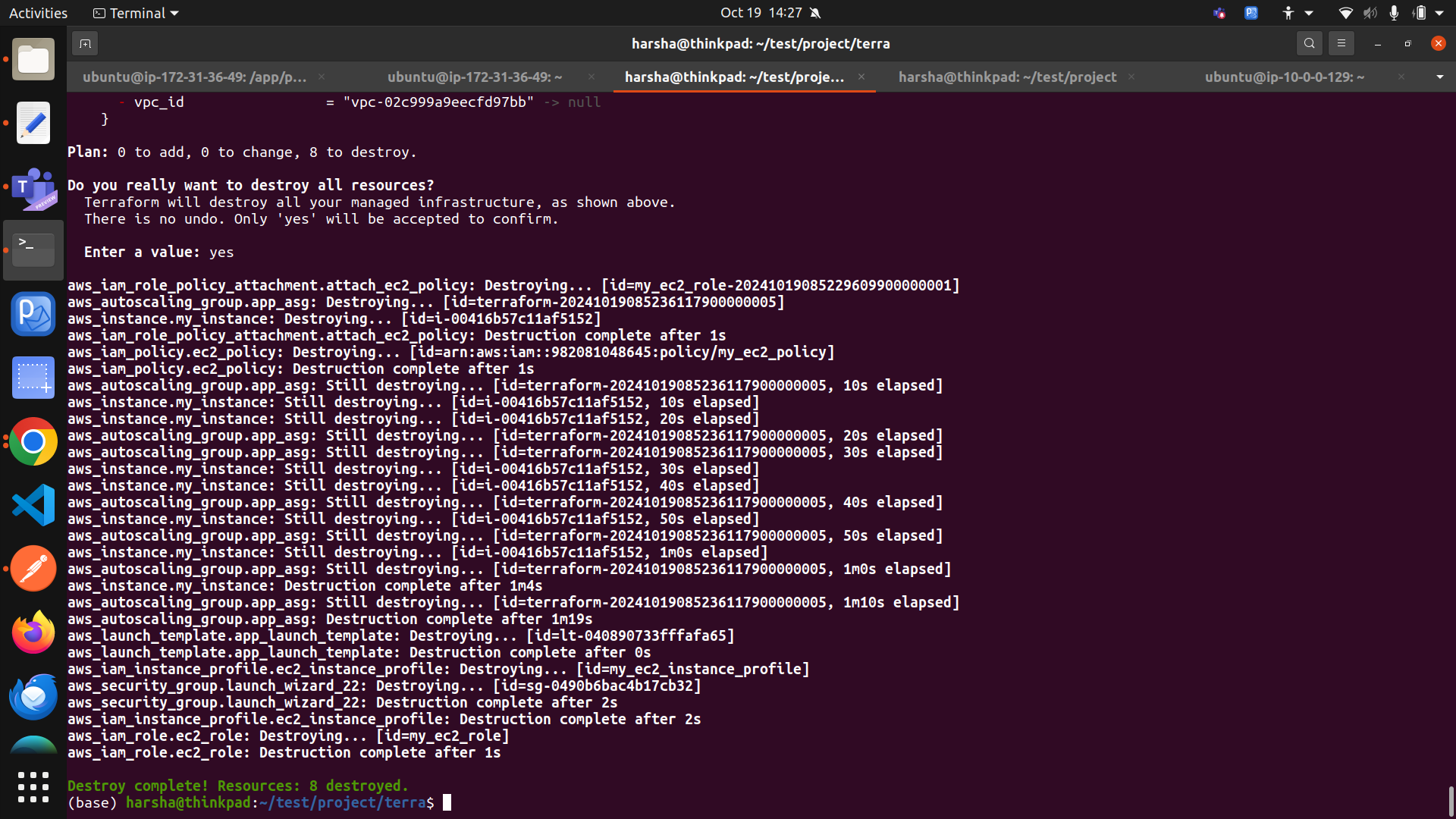
**Git Hub Url** ==> https://github.com/rhvreddy/project

**Infrastructure Setup** ==>

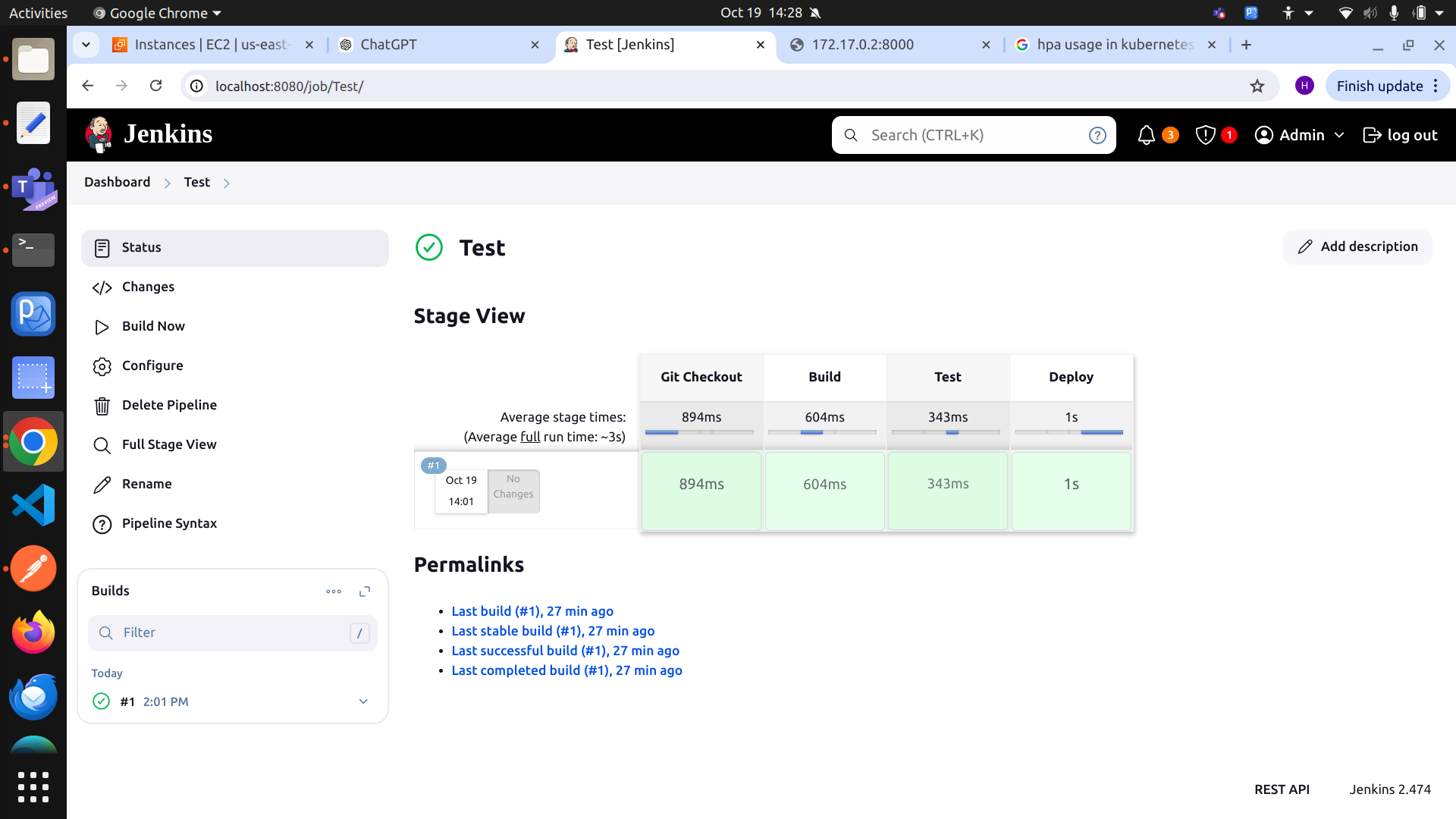
* I've set up a basic web application using Python Flask. The application code is in **app.py**, and the dependencies are listed in **requirements.txt**, both of which are stored in the **GitHub repository**."
* This clearly explains that you've created a Flask app, and the relevant files (**app.py and requirements.txt**) are in your **GitHub repository**
* I used **Infrastructure as Code** (IaC) with Terraform to automate the provisioning of resources in **AWS**. The code I developed handles **auto-scaling, creating new instances, IAM roles, and security groups**. The code is saved in the provided **GitHub URL**, within the **terraform folder**, specifically in the **main.tf** file.

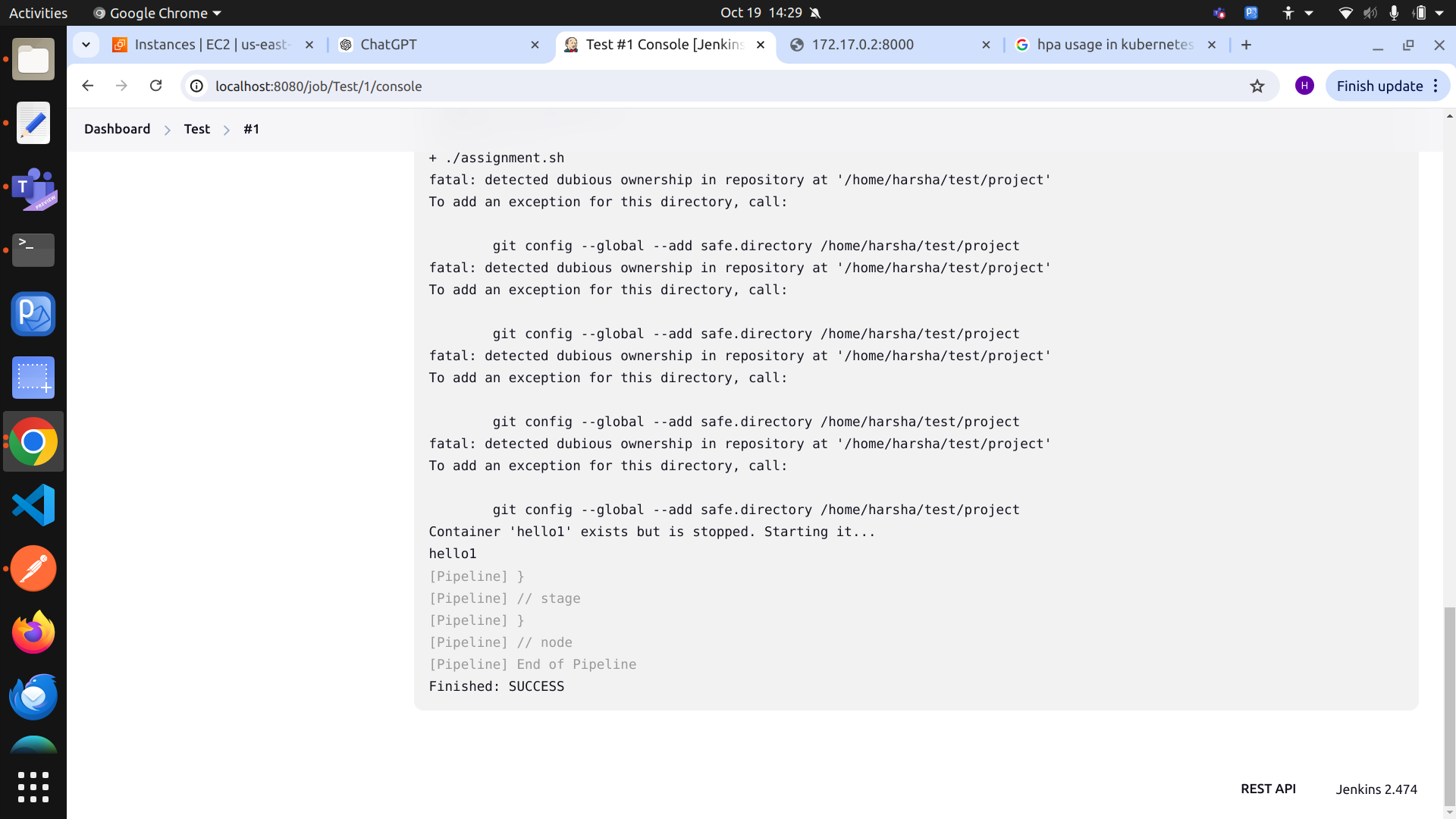
In the image below you can find my webpage hosting.  


In the below image you can find the performance related to terraform

terraform apply  
  
  
terraform destroy  
  
  
**CI/CD Pipeline Setup** ==>

* I created a pipeline using Jenkins that performs **build, test, and deployment** tasks. In the '**Poll SCM'** configuration, I set the schedule to **\* \* \* \* \***, which checks for changes in the respective repository every minute. Based on these changes, the pipeline automatically deploys updates. The code is stored in the **ci&cd** folder of the repository.

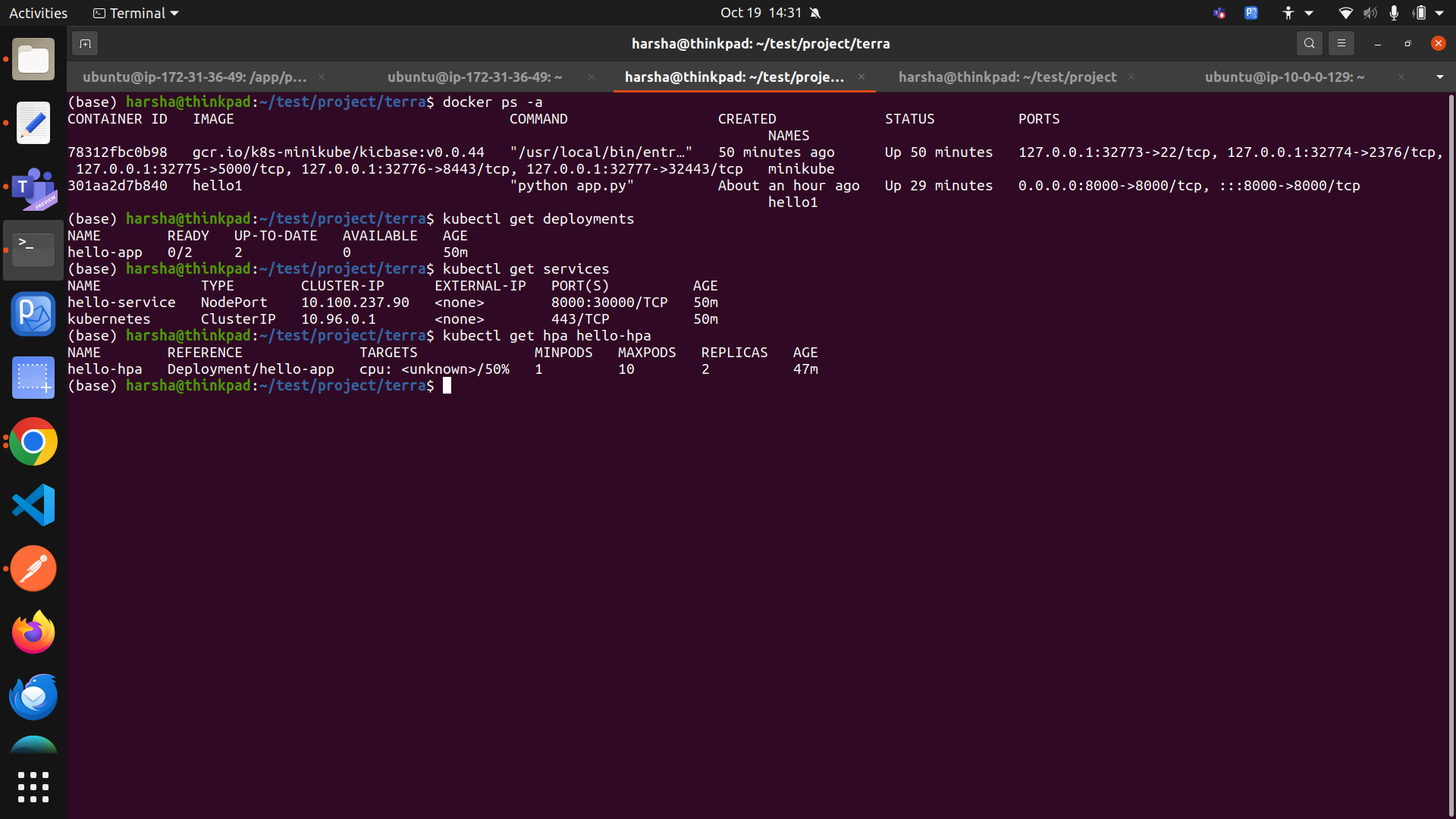
From the below image you can find the Jenkins pipeline image  
  




**Containerization and Orchestration** ==>

* I have containerized my web application using Docker, creating a **Dockerfile and docker\_run.sh** script to deploy the container. The application is currently running within that container. Additionally, I have set up an orchestration platform using **Kubernetes and implemented scaling functionality**. In the repository, I have provided the **deployment.yaml** file for creating the cluster and **hpa.yaml** for scaling, located in the kubernetes folder.

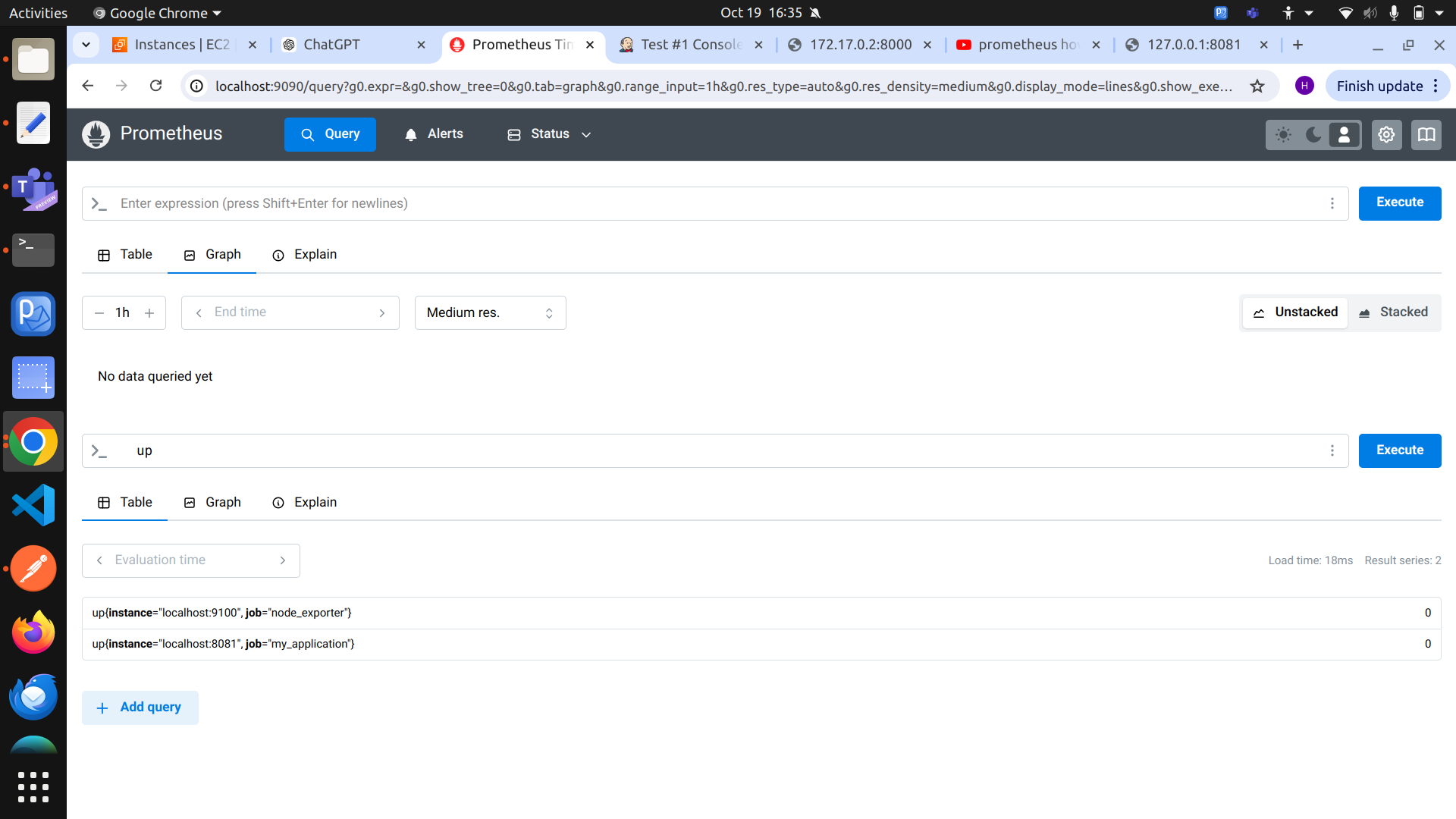
Below images help to find the running containers



**Monitoring and logging ==>**

* I have used Prometheus to collect and visualize metrics, monitoring CPU performance, memory, disk usage, and application health. I have also set up alerts for critical thresholds. In the GitHub repository, you can find the **prometheus** folder, which contains the **prometheus.yml** for configuration, **app\_metrics.py** for application metrics, and **alert.rules.yml** for setting up alerts.

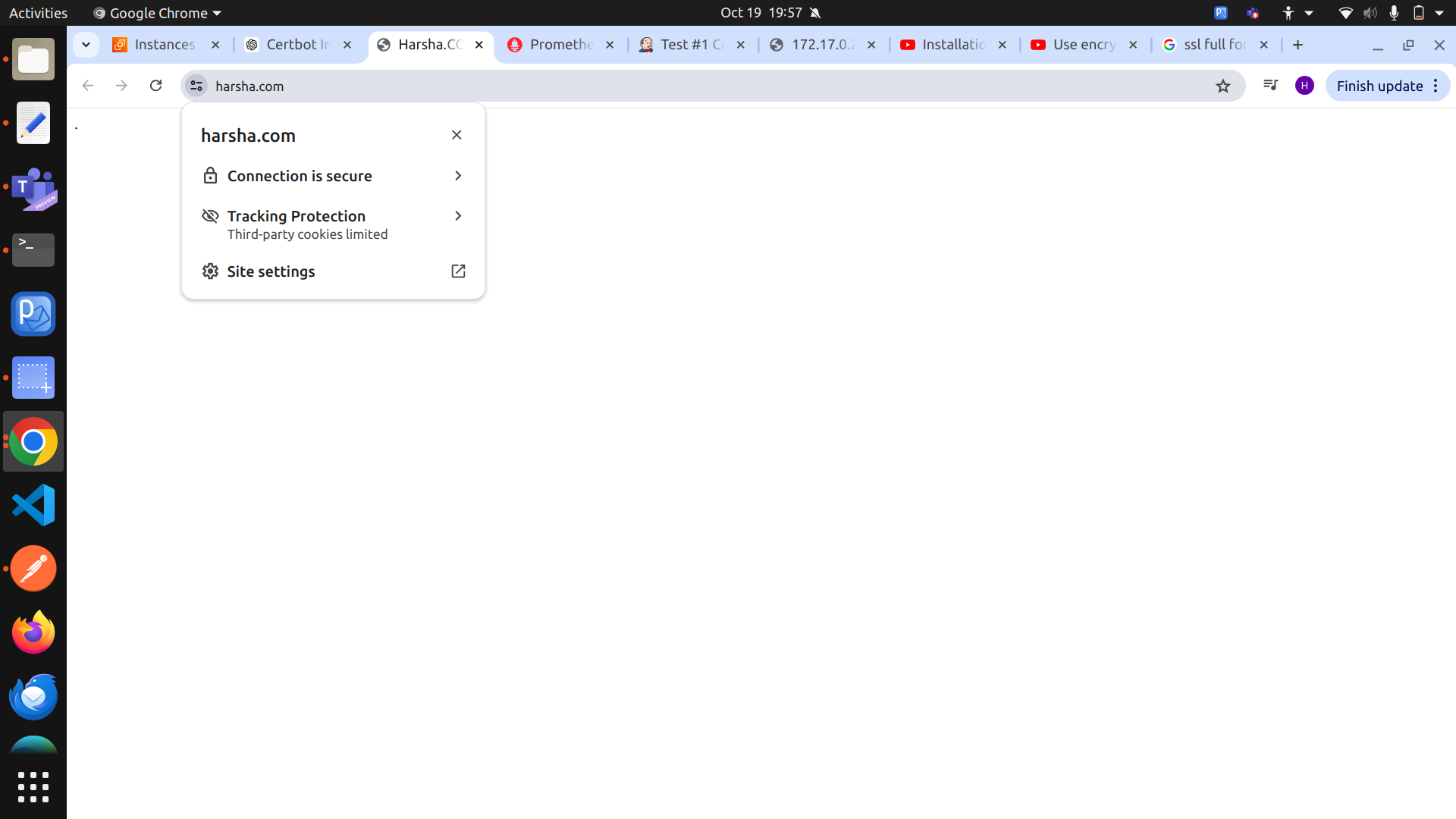
Below image is for reference



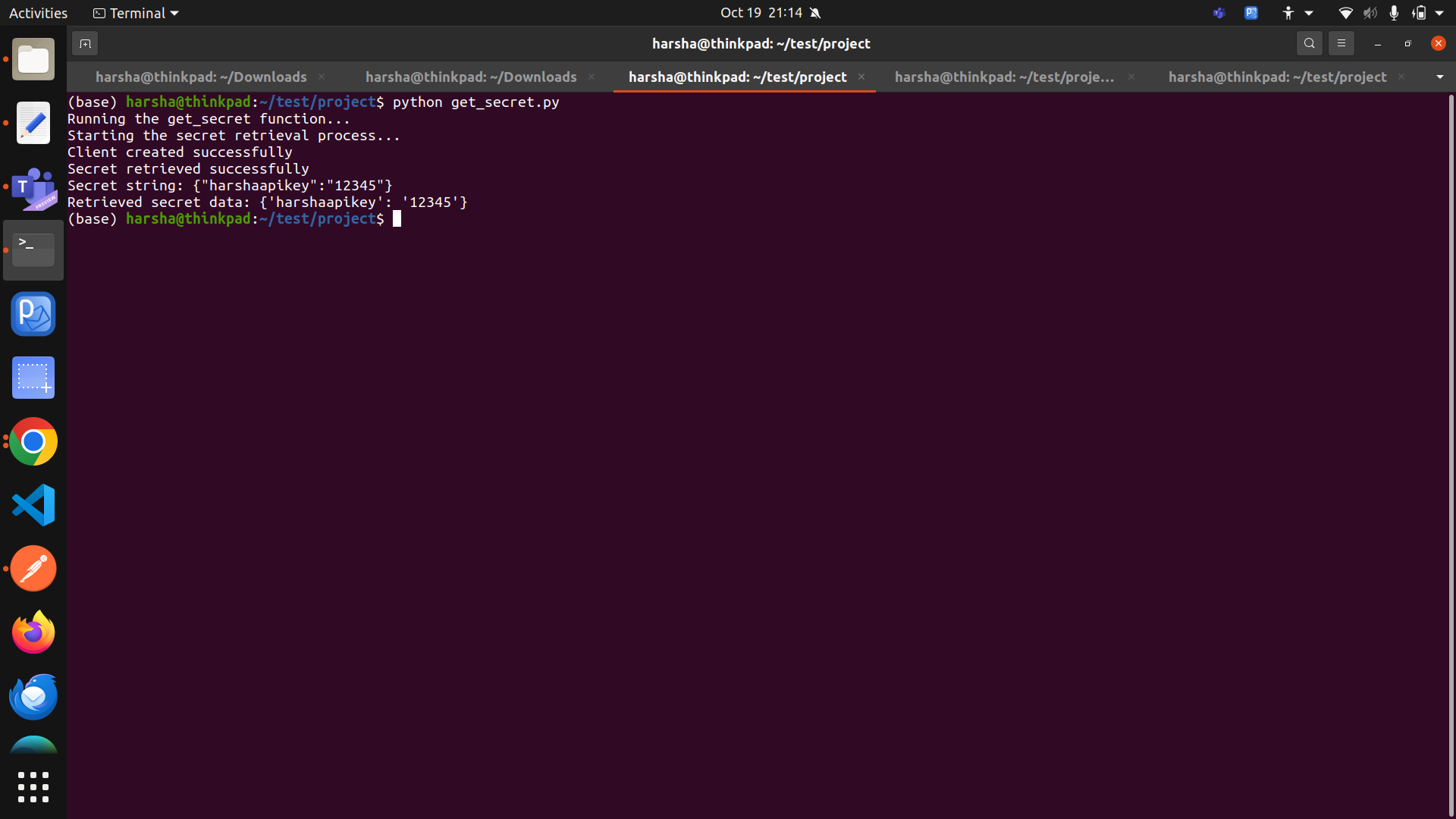
**Security Best Practices** ==>

* I have implemented encrypted connections using **SSL/TLS** and managed sensitive data with **AWS Secrets Manager**, where I created and stored a new secret. In the **GitHub repository**, you can find the Python code that handles the key details and IAM roles, which I configured with least privilege, granting read-only access to a specific S3 bucket. Additionally, I used **Trivy** for **scanning Docker images**.

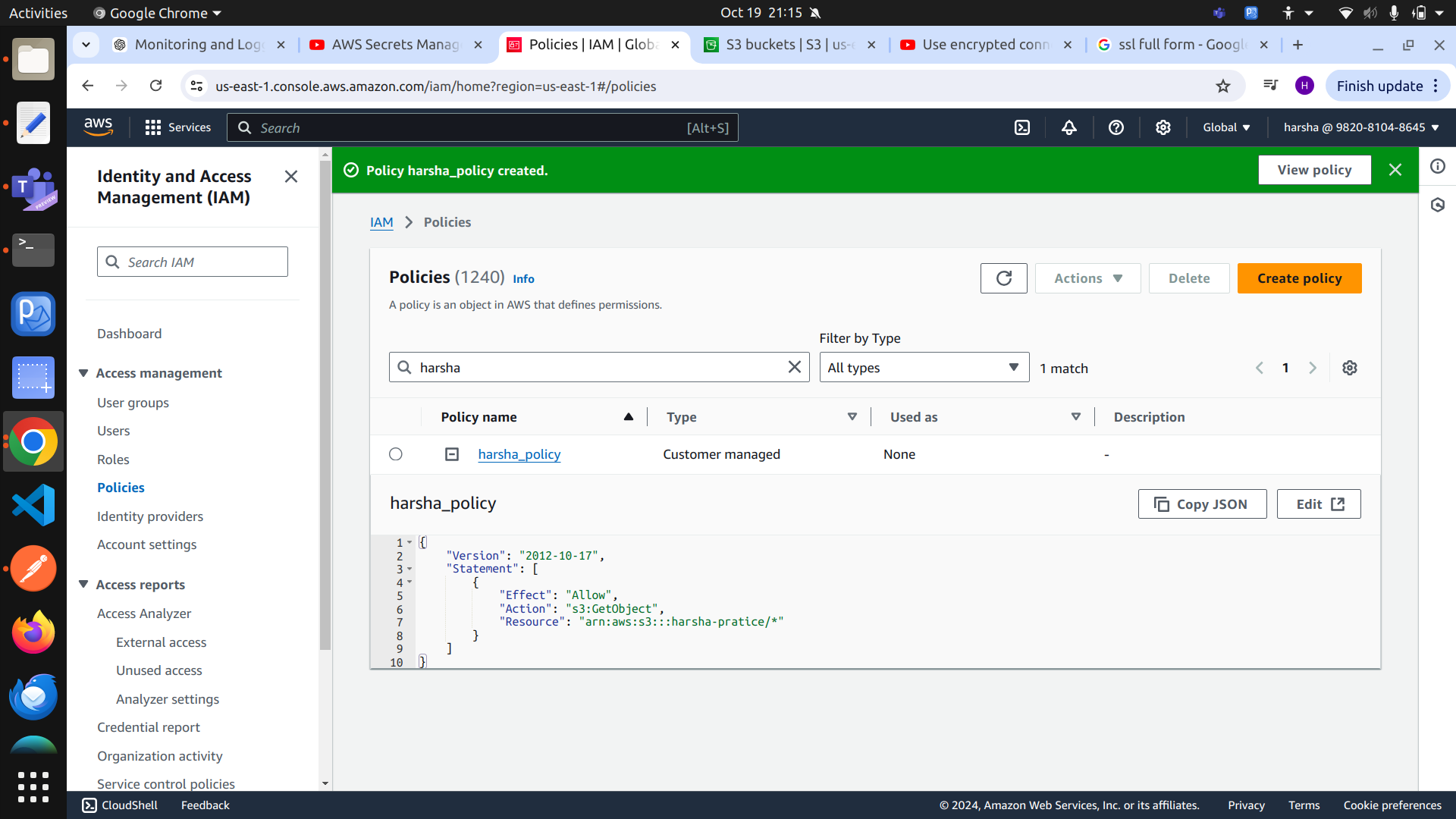
The below image tells us that you connection is secure



Below image give you infromation for AWS Secrets manager



Below image gives you deatils regarding IAM police with only read access



Below image reference for trivy scanning of docker image  
  
