**A Project Report On**

**APPLICATION AND DATABASE ORCHESTRATION USING KUBERNETES**

**TECHNICAL HUB**

**Submitted by**

**TEAM TECH THUNDERS**

***Under the esteemed guidance of***

**Mr. BOBBY & Mr.SURYA ASHOK**

**ABSTRACT**: We are building an application and that application is deploy by using Kubernetes tool, Application and Database Orchestration Using Kubernetes.

**TOOLS REQUIRED:** Docker, Minikube, Kubectl, Helm, Prometheus, Grafana .

**THEORY:**

Kubernetes is a popular open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. In addition to managing the containers themselves, Kubernetes can also be used to orchestrate the deployment and management of databases used by those applications.

Some of the popular terms used in Kubernetes are given below:

**Node:** A node is a physical or virtual machine that runs Kubernetes. Nodes are responsible for running containers and providing networking and storage resources to the containers.

**Pod:** A pod is the smallest deployable unit in Kubernetes. A pod contains one or more containers, and all the containers in a pod share the same network namespace, IP address, and storage volumes.

**ReplicaSet**: A ReplicaSet is used to manage the deployment and scaling of a set of pods in Kubernetes. ReplicaSets ensure that a specified number of replicas (copies) of a pod are always running.

**Deployment:** A Deployment is a higher-level abstraction that manages ReplicaSets and provides rolling updates and rollbacks for deployments.

**Service:** A Service is a Kubernetes resource that provides a stable IP address and DNS name for a set of pods. Services can be used for load balancing, service discovery, and exposing applications to the network.

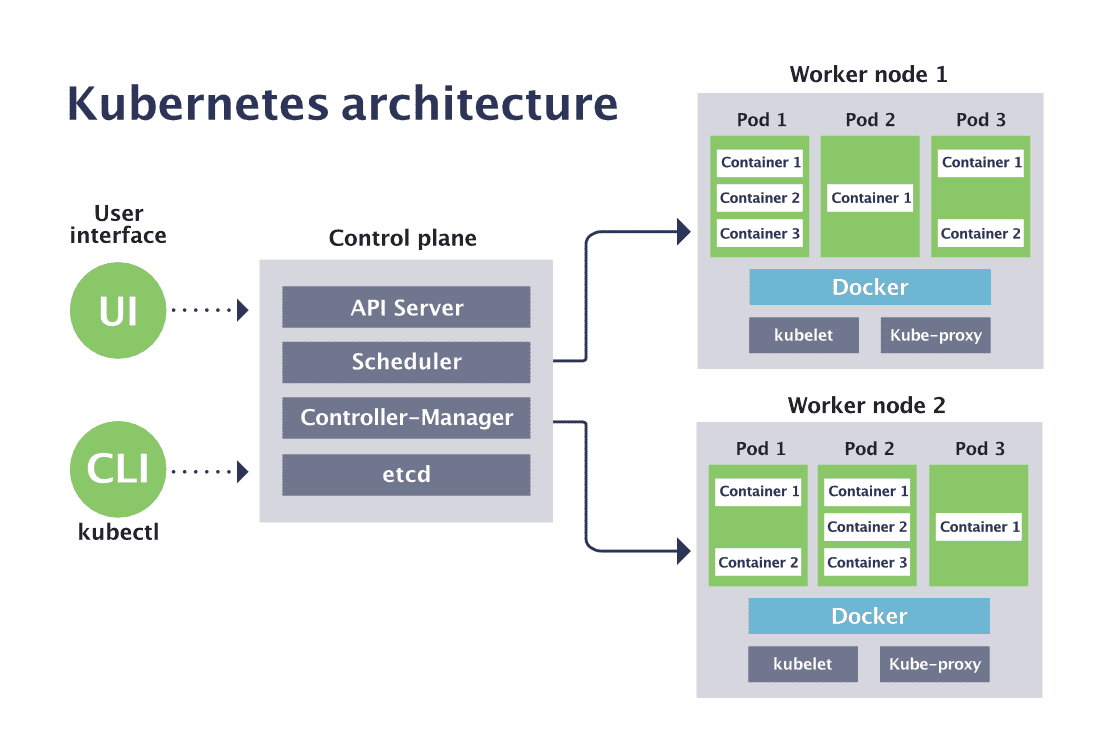
**Namespace:** A Namespace is a logical partition within a Kubernetes cluster that allows you to isolate resources and manage access control.

**ConfigMap:** A ConfigMap is a Kubernetes resource that provides a way to store configuration data, such as environment variables or configuration files, as key-value pairs.

**Secret:** A Secret is a Kubernetes resource that provides a way to store sensitive data, such as passwords or API keys, in an encrypted format.

**Ingress:** An Ingress is a Kubernetes resource that provides a way to expose HTTP and HTTPS routes from outside the cluster to services within the cluster.

**Container:** A container is a lightweight, standalone executable package of software that includes everything needed to run an application, including code, libraries, and dependencies.



🡪Kubernetes is an architecture that offers a loosely coupled mechanism for service discovery across a cluster.

🡪A Kubernetes cluster has one or more control planes, and one or more compute nodes.

**🡪Master Node:**The master node is a node that manages the overall state of the cluster and makes global decisions about the cluster. It includes API Server, Scheduler, Controller-manager, etc.

**🡪Worker Node**:In Kubernetes, Worker nodes are responsible for running the application workloads and handling the incoming traffic.

DESING PROCEDURE:

1.**Install Minikube**:

First, you need to install Minikube, which is a tool for running Kubernetes locally on your machine. You can follow the installation instructions provided by the official Minikube documentation for your operating system.

🡪Download the minikube and kubectl.

🡪Open Powershell and type the below command

New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force

Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' –UseBasicParsing

🡪Open PowerShell and click run as administrator and enter the below command

$oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)

if ($oldPath.Split(';') -inotcontains 'C:\minikube'){ `

[Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine) `

}

🡪And start your Minikube service in your local machine by entering the command below in command prompt or PowerShell.

Minikube start

2.**Create a docker images for that application**:

Create the separate folders for required resources frontend, version(v1),version (v2),details and create yaml file for the each folder using vscode.Then open gitbash from each folder and enter the below commands to create the docker images for that and push the images into your docker hub and run the containers atleast once.

Docker login

Docker build –t app\_name .

Docker tag app\_name:latest dockerhub\_reponame/app\_name

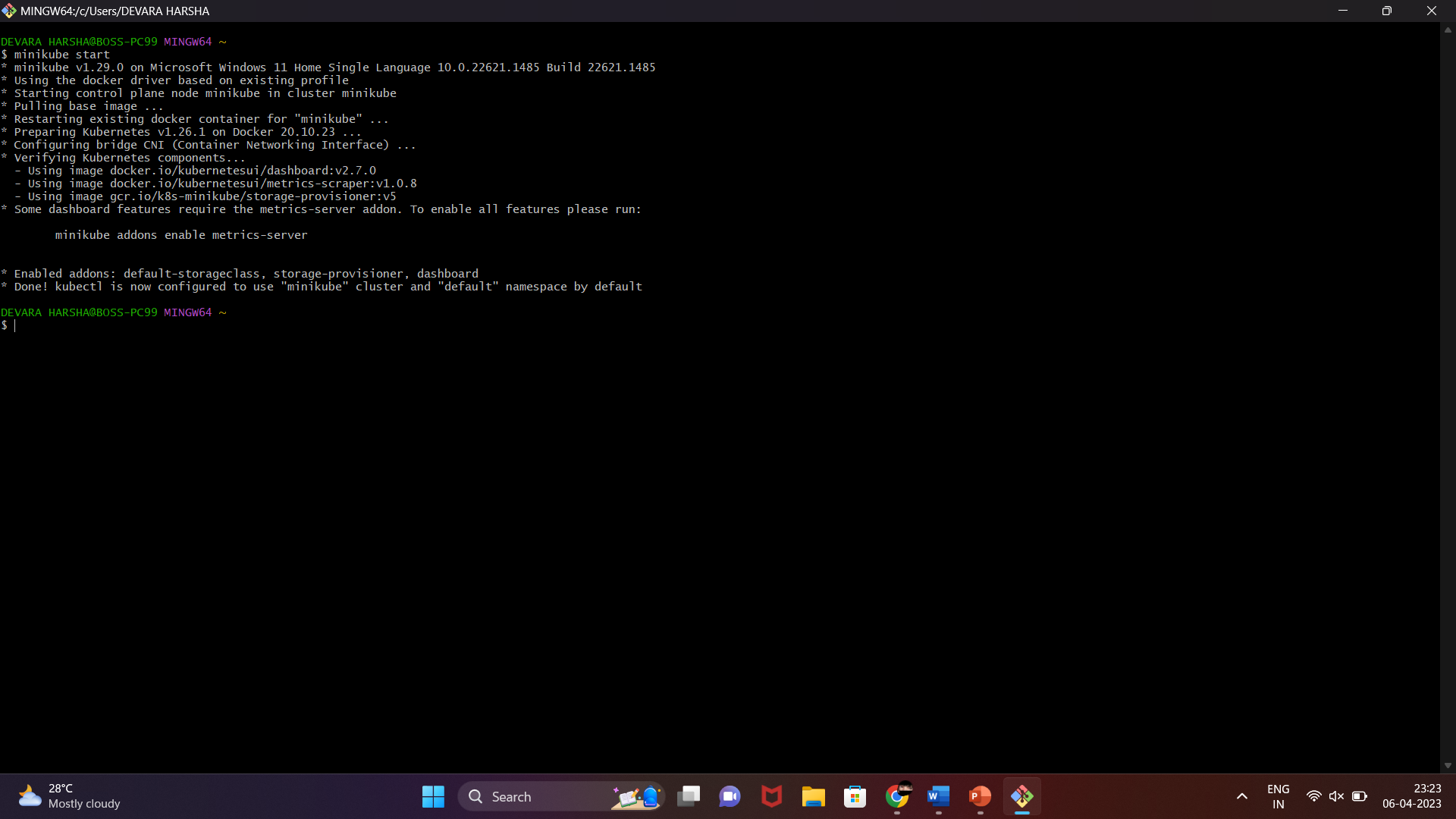
Docker push dockerhub\_reponame/file\_name.

Docker run –d –p port\_number:port\_number filename

3.**Creating cluster environment for the application:**

🡪Now open Git bash and click below command .Hence the cluster environment is created.

Minikube start



**4.Creating pods in the cluster.**

Now we need to create pods for each required folders by using the following commands.

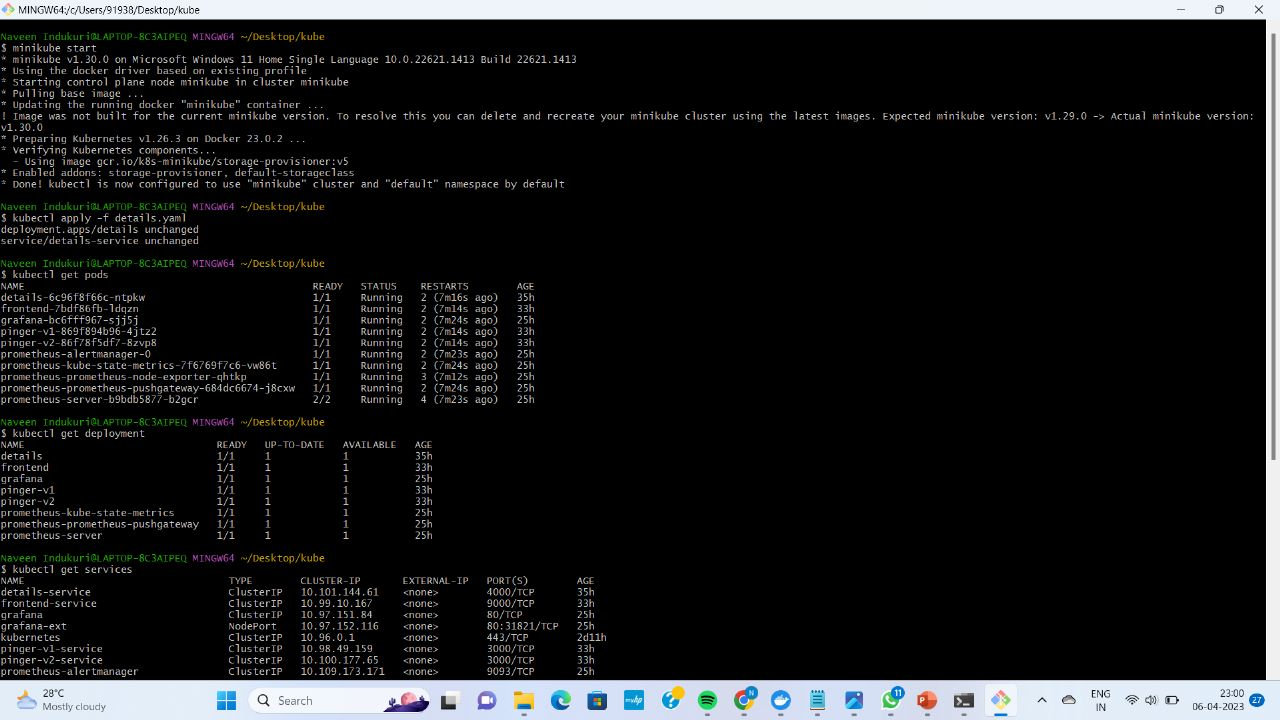
Kubectl apply –f file\_name.yaml //For creating pod for existing yaml file

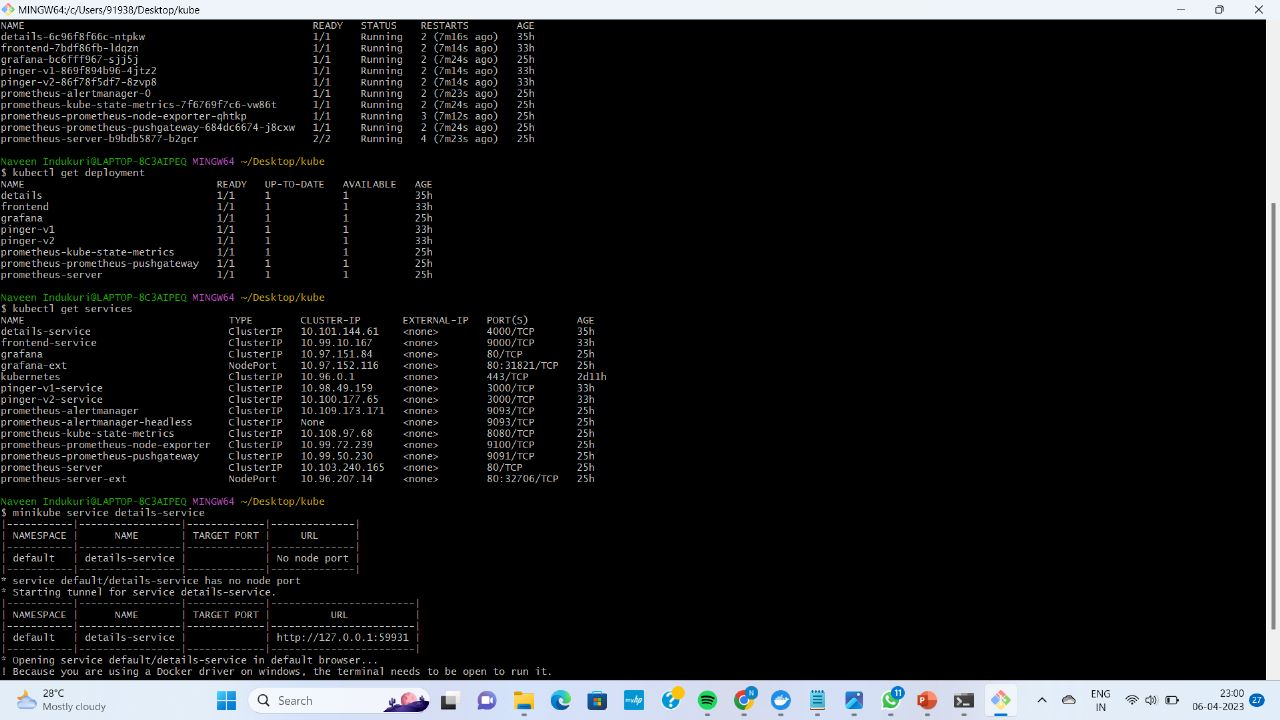
Kubectl get pods // It shows the all created pods

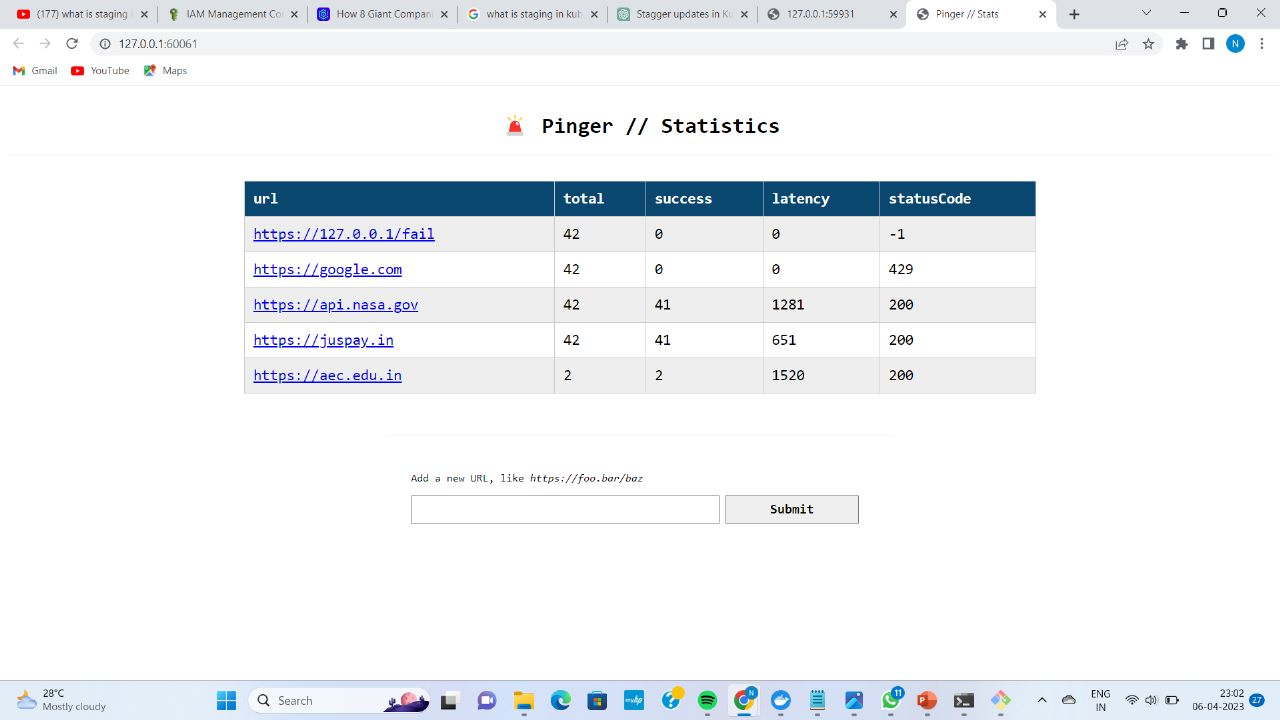
Kubectl get deployments

Kubectl get services //It shows all services outside the kubernetes cluster

Minikube service file\_name-service //For running the container within the pod.







**5.Check the state of containers:**

We have to check the state of the container whether they are running or not using the following command

Kubectl get all

MONITORINGN THE APPLICATION:

By using these tools for monitoring the application.

1.Helm

2.Prometheus

3.Grafana

1. Helm:

Helm is a package manager for Kubernetes, which simplifies the deployment and management of containerized applications and services in a Kubernetes cluster. It was created by the developers at Deis, and is now maintained by the Cloud Native Computing Foundation (CNCF).

Helm packages are called "charts," which are pre-configured templates for deploying and managing Kubernetes applications. Charts can be easily customized using configuration files called "values," which allow users to specify specific configurations for their deployments.

Helm also includes a command-line interface (CLI) tool, which provides a simple and consistent way to install, upgrade, and manage charts in a Kubernetes cluster. Helm CLI allows for easy installation of charts from remote repositories, as well as creation and management of local chart repositories.

In addition to simplifying the deployment and management of Kubernetes applications, Helm also provides a mechanism for versioning and rolling back deployments. This enables users to easily track changes to their deployments over time, and revert to previous versions if necessary.

Overall, Helm is a powerful tool for simplifying the deployment and management of containerized applications and services in Kubernetes, and is widely used in the Kubernetes community.

2. What is Prometheus and how it will work?

Prometheus is an open-source monitoring and alerting toolkit originally developed by SoundCloud. It was created to help monitor and manage systems and applications in a modern, cloud-based infrastructure.

Prometheus works by collecting metrics from various sources, such as applications, systems, and databases, and storing them in a time-series database. These metrics can be used to track performance, detect anomalies, and troubleshoot issues.

Prometheus uses a pull-based model, which means that it periodically scrapes targets to collect metrics. Targets can be configured using service discovery or static configuration. Once collected, the metrics are stored in a time-series database, which allows for queries and analysis over time.

Prometheus also includes a powerful query language called PromQL, which can be used to query and analyze metrics. Additionally, Prometheus includes a flexible alerting system that can be configured to trigger notifications based on predefined thresholds or conditions.

Overall, Prometheus provides a robust and flexible monitoring solution for modern, cloud-based systems and applications.

3. What is Grafana?

Grafana is an open-source data visualization and monitoring platform that allows users to create dashboards, alerts, and notifications for their systems and applications. It was created by Torkel Ödegaard in 2014 and is now maintained by Grafana Labs.

Grafana supports a wide variety of data sources, including popular databases like MySQL, PostgreSQL, and InfluxDB, as well as cloud-based services like AWS CloudWatch and Google Analytics. Users can connect to these data sources and use Grafana's query language to create custom queries and visualizations.

Grafana also includes a wide variety of built-in visualization options, including line charts, bar charts, heat maps, and more. Users can customize the appearance of their dashboards and visualizations using a drag-and-drop interface, and can share their work with others using Grafana's built-in sharing and collaboration features.

In addition to data visualization, Grafana also includes powerful monitoring and alerting features. Users can create custom alerts based on specific conditions, such as CPU usage or error rates, and receive notifications via email, Slack, or other channels.

Overall, Grafana is a flexible and powerful platform for data visualization and monitoring, and is widely used by both developers and operations teams in a variety of industries.

SETUPING THE PROMETHUS AND GRAFANA:

By using the below commands in command line or git bash.

Steps to set up Prometheus:

helm version

kubectl get pods

helm repo add Prometheus-community https://prometheus-community.github.io/helm-charts

help search repo Prometheus

helm repo update

helm install Prometheus Prometheus-community/Prometheus

kubectl get all

kubectl get svc

kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

minikube service prometheus-server-ext

kubectl get pods

Steps to set up Grafana:

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

helm install grafana grafana/grafana

kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext

minikube service grafana-ext

kube ctl get pods

To get user name and password in Grafana:

kubectl get secret --namespace default grafana -o yaml

echo "#admin user" | openssl base64 -d ; echo

echo "#admin passwd" | openssl base64 -d ; echo

or

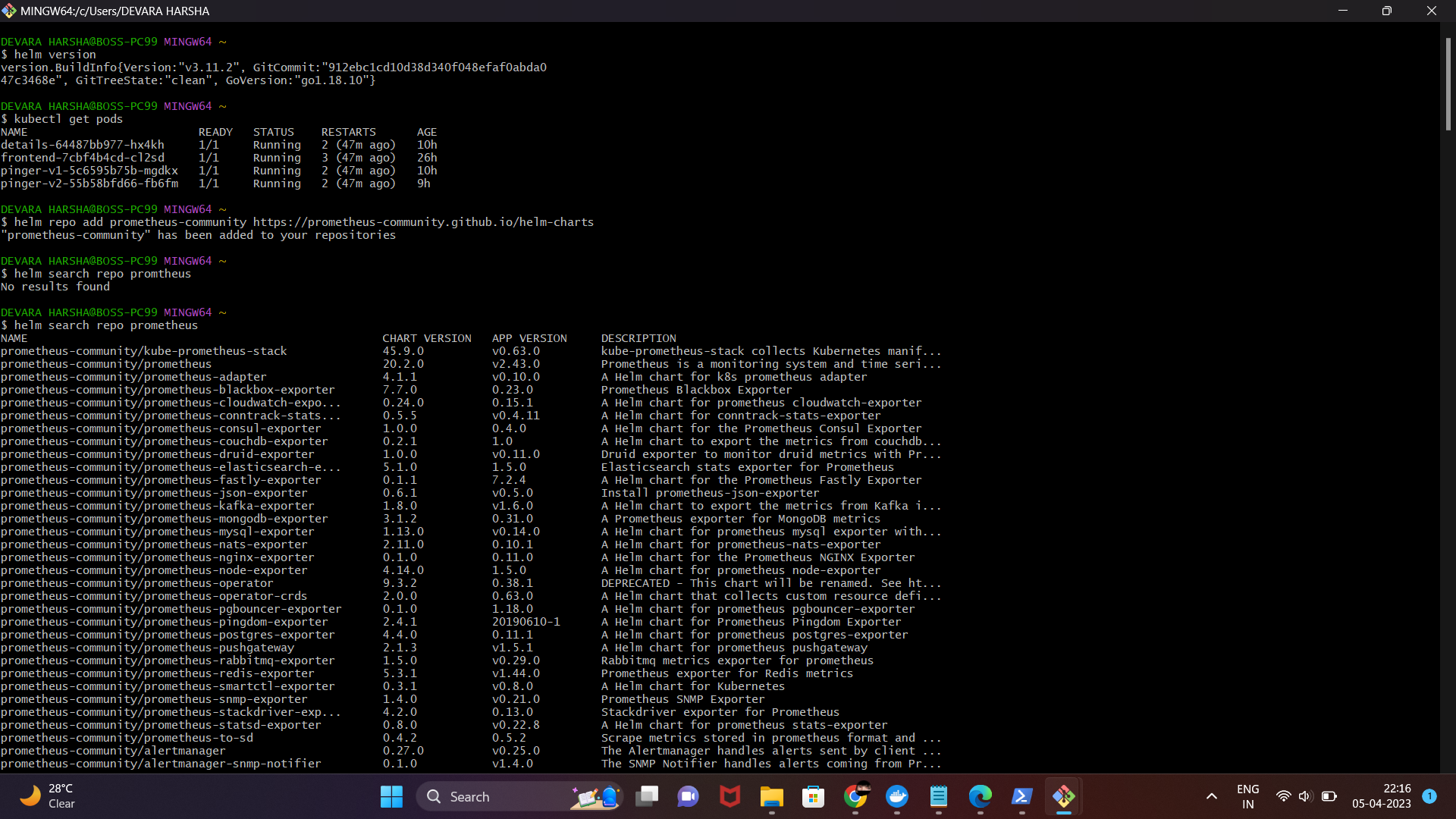
kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

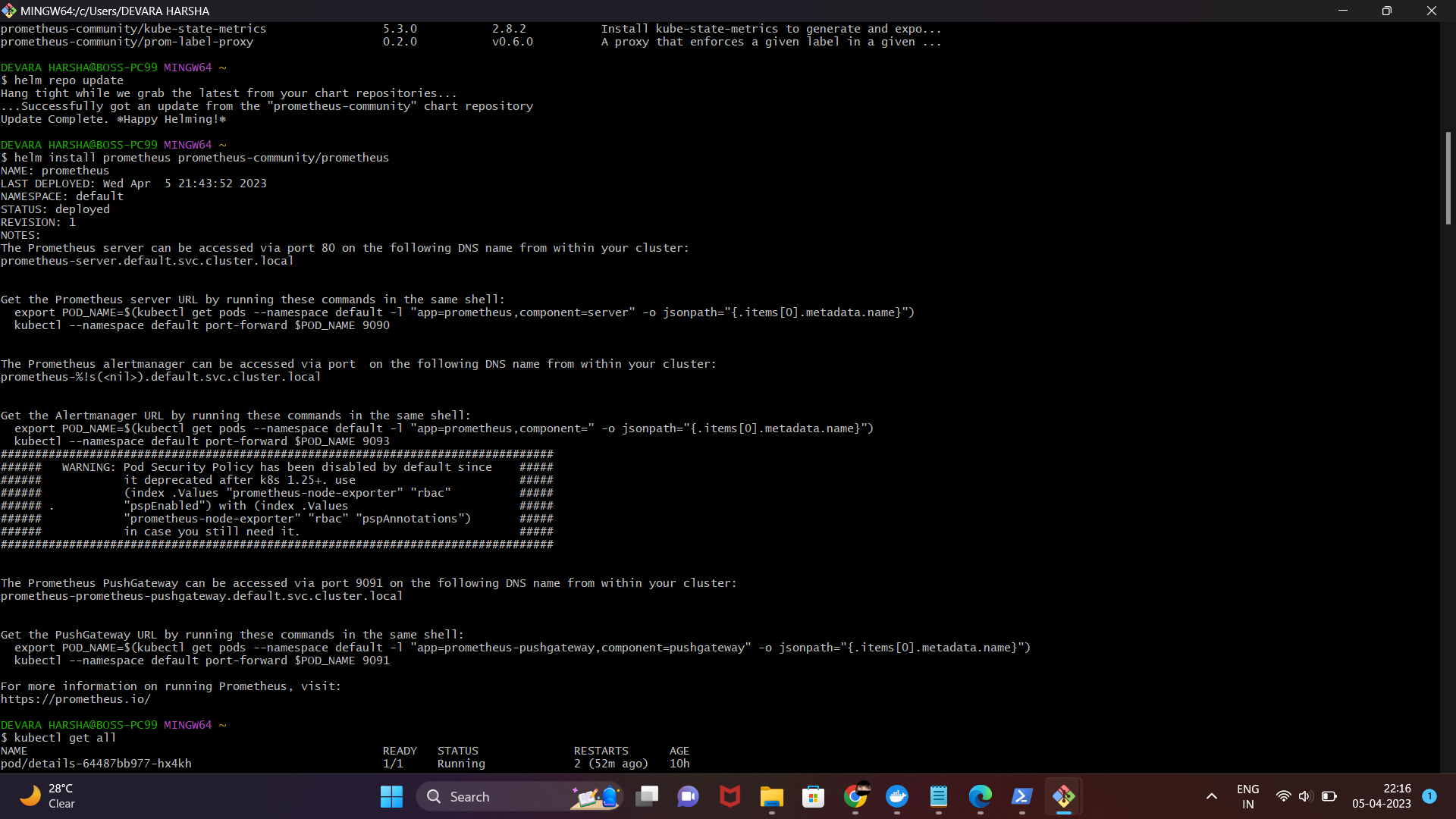
Dashboards: https://grafana.com/grafana/dashboards/6417

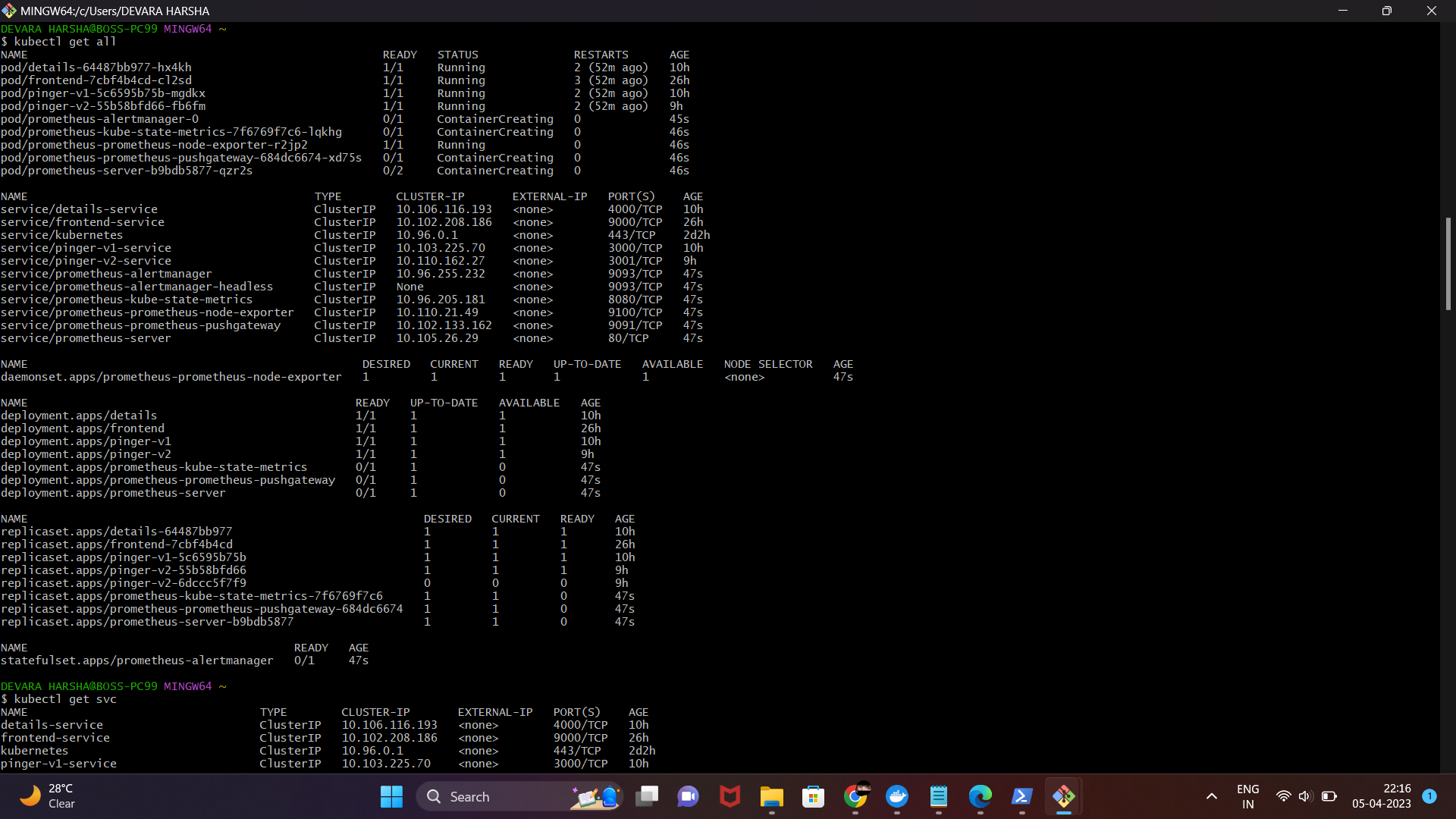
minikube service prometheus-server-ext

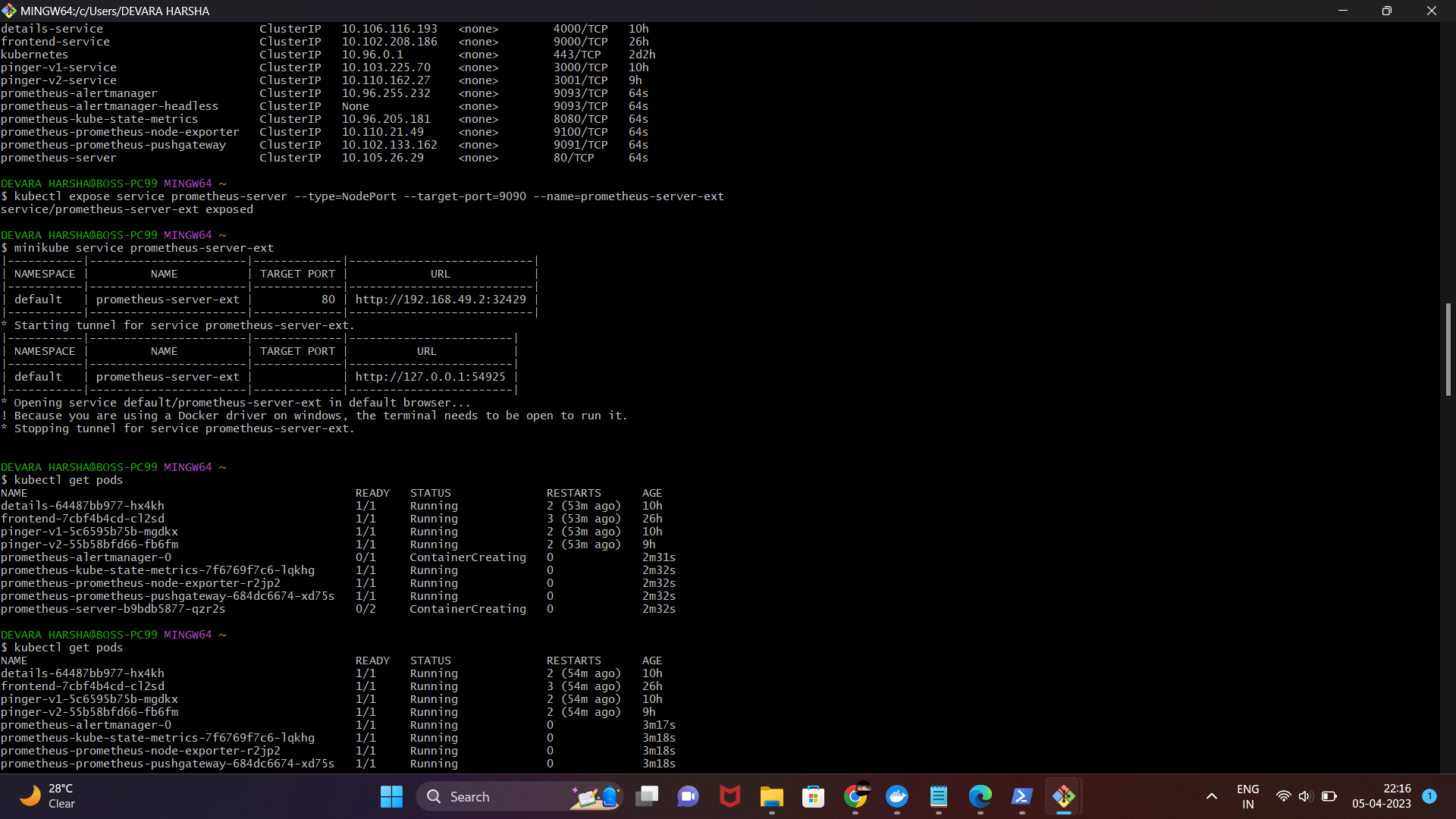
🡪And, now we can open a another git bash enter the below command for running the Grafana in your local system.

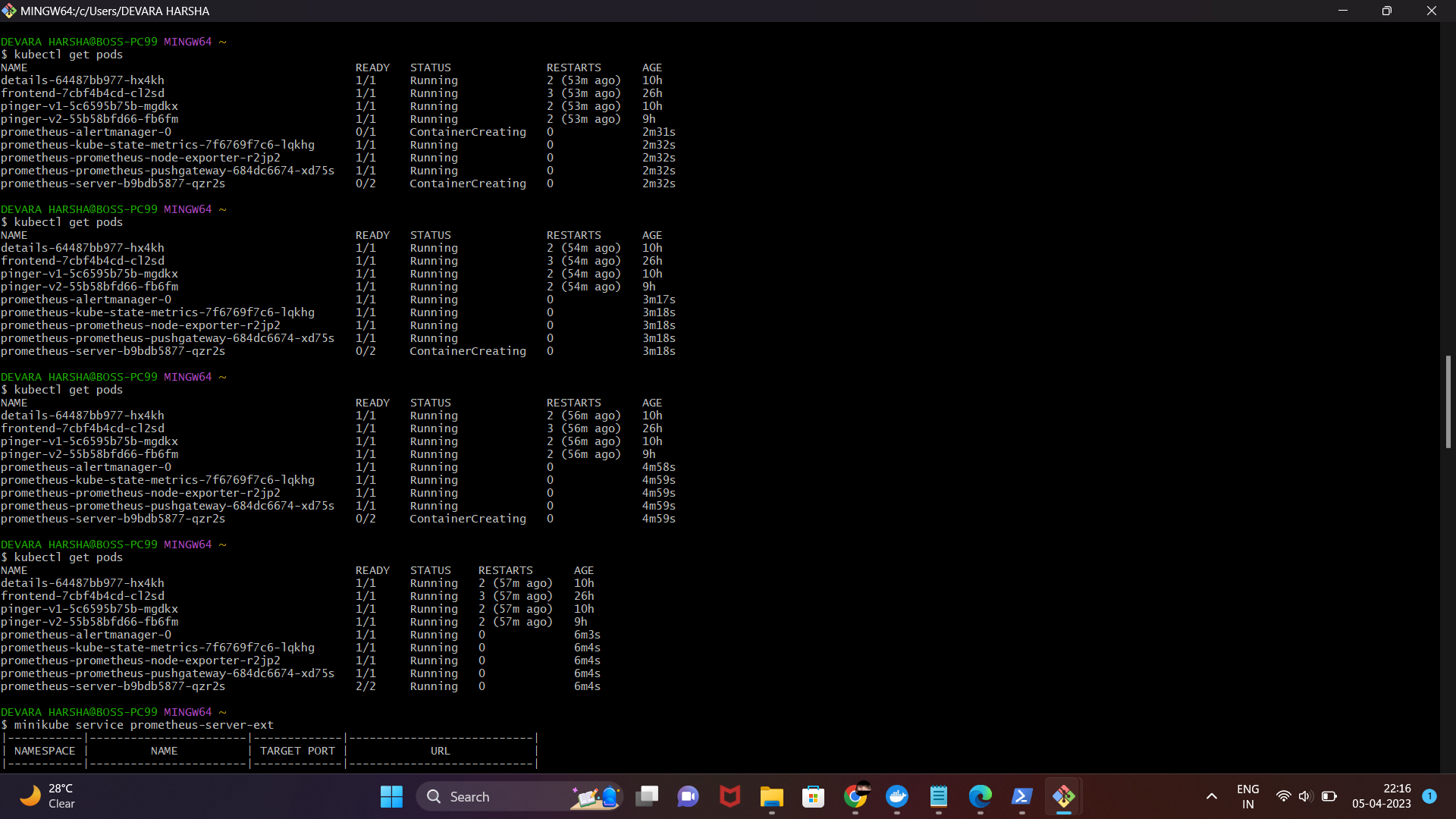
minikube service grafana-ext

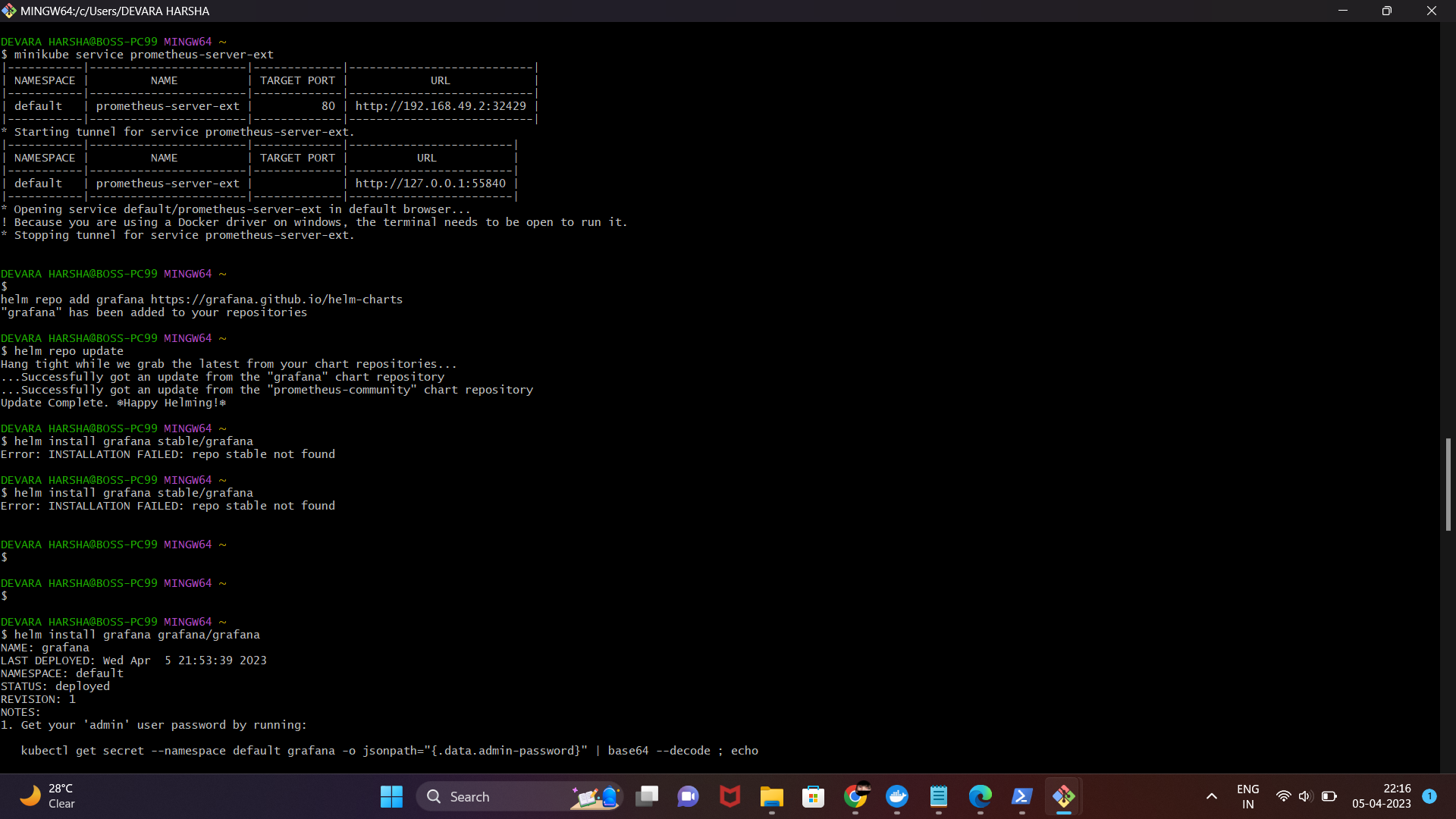


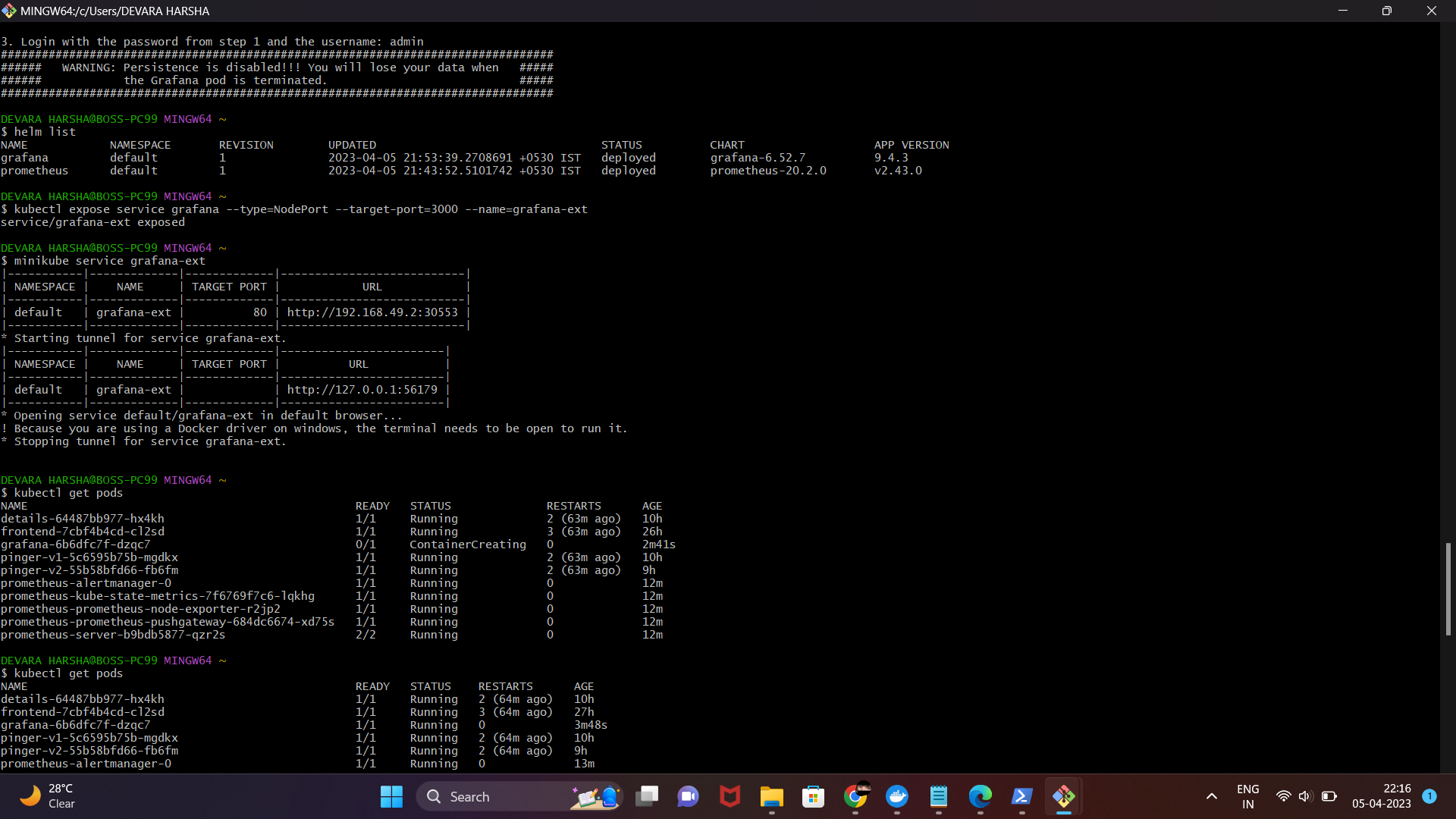


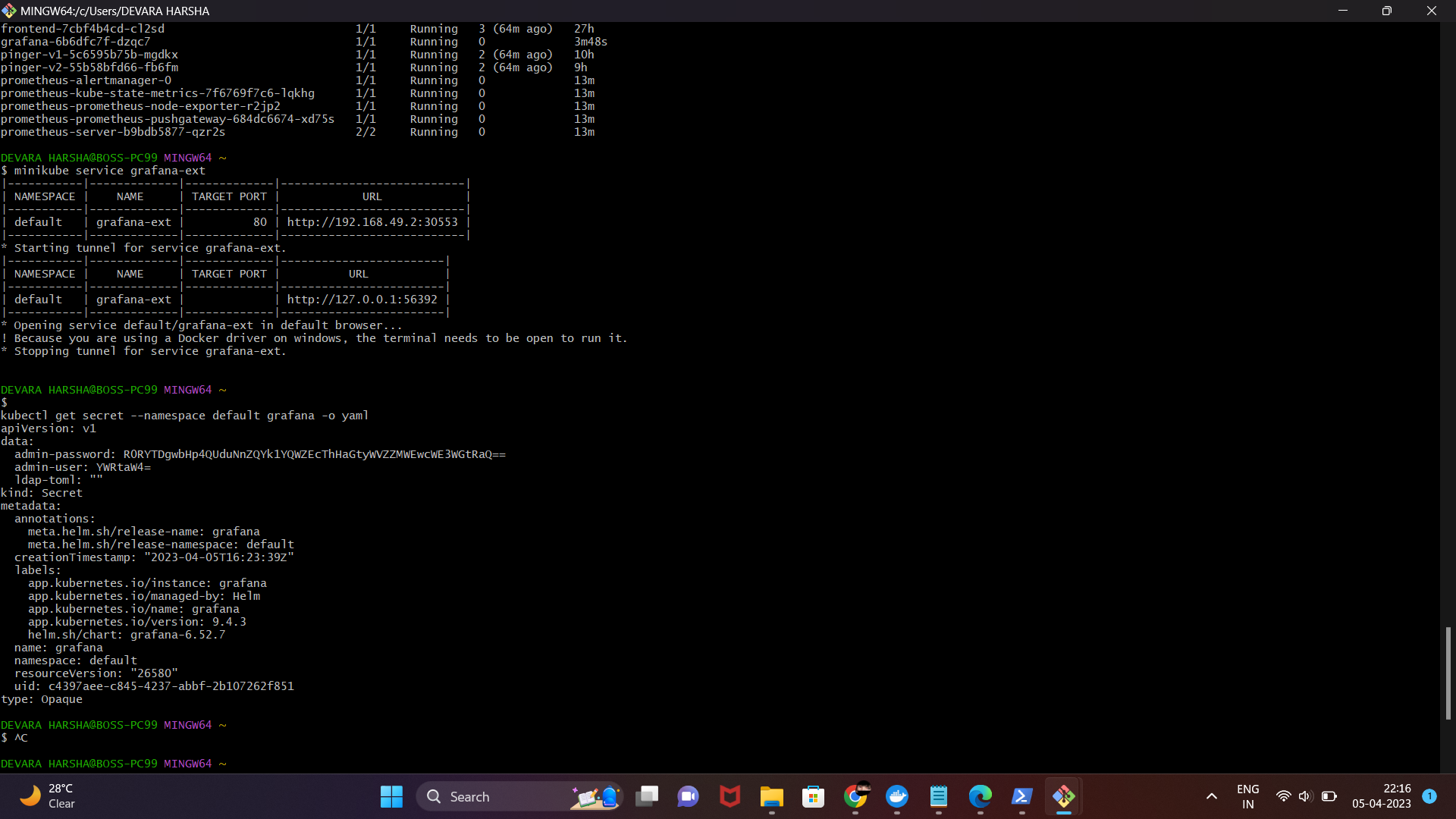


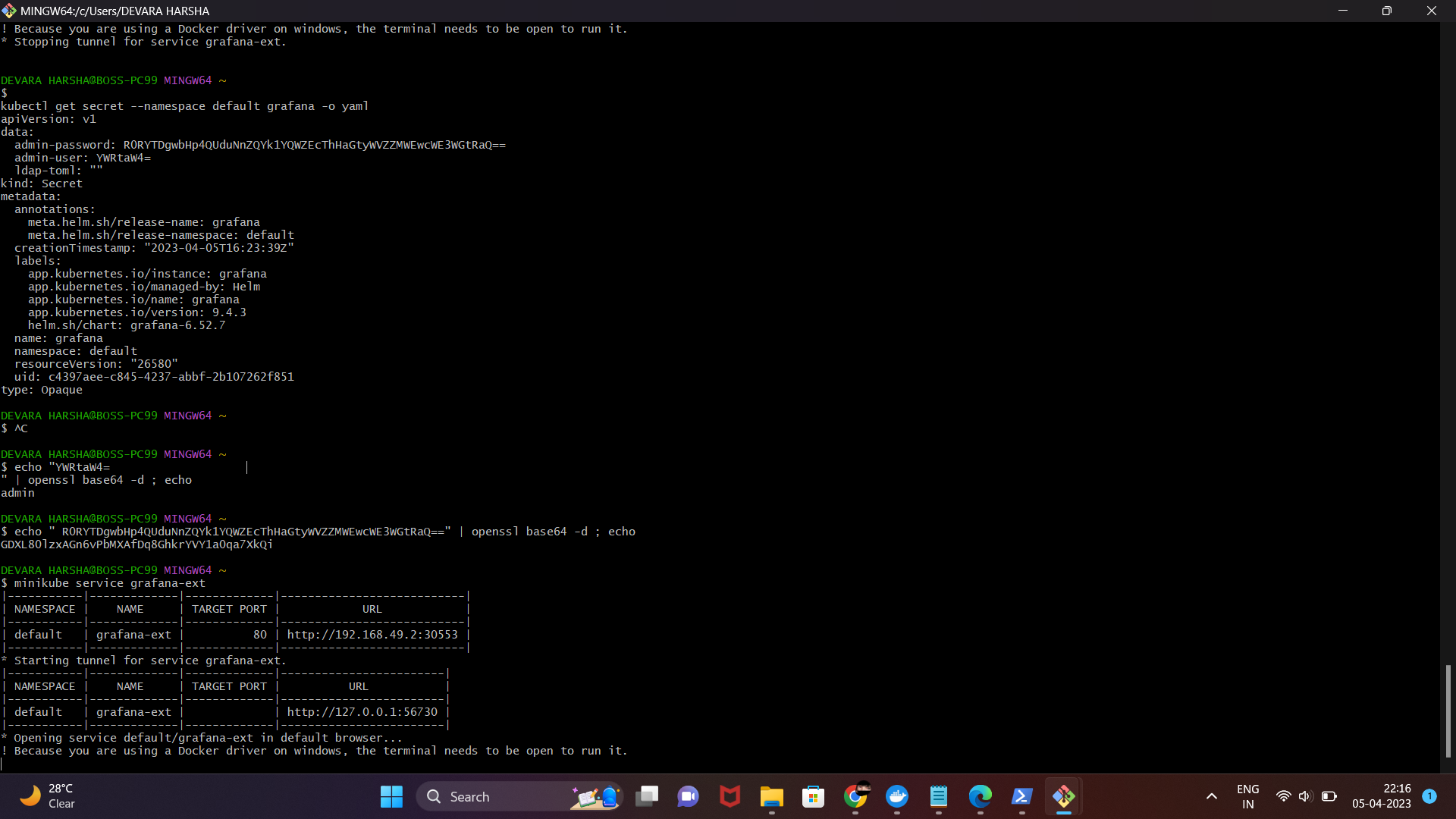


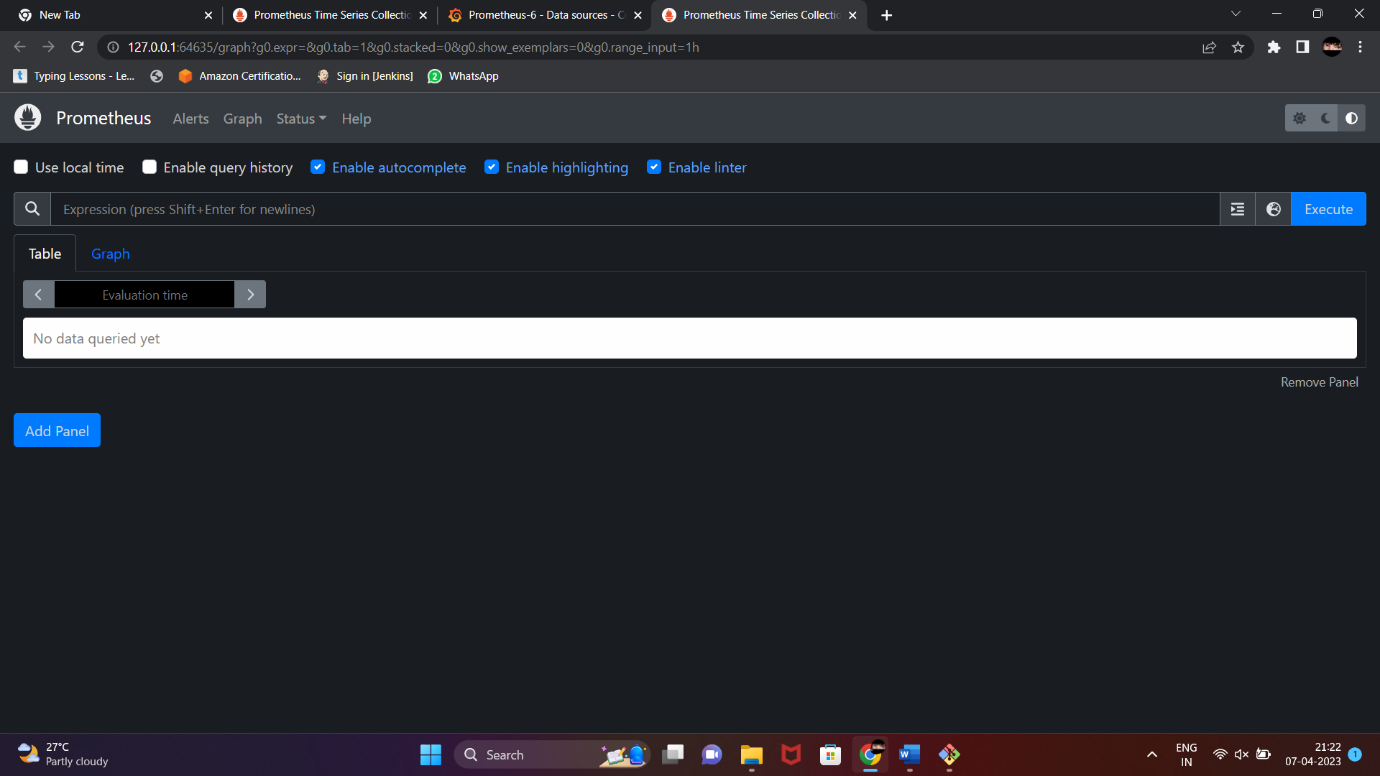




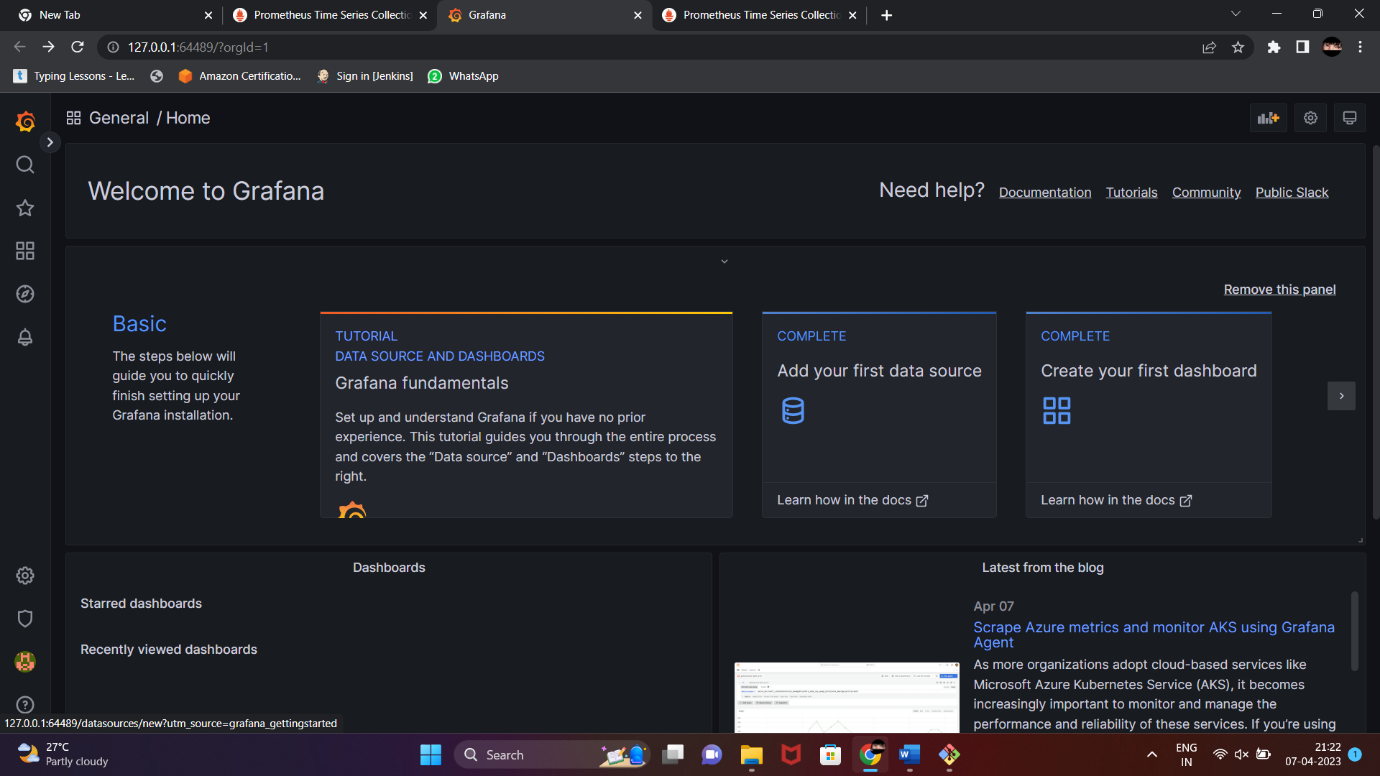


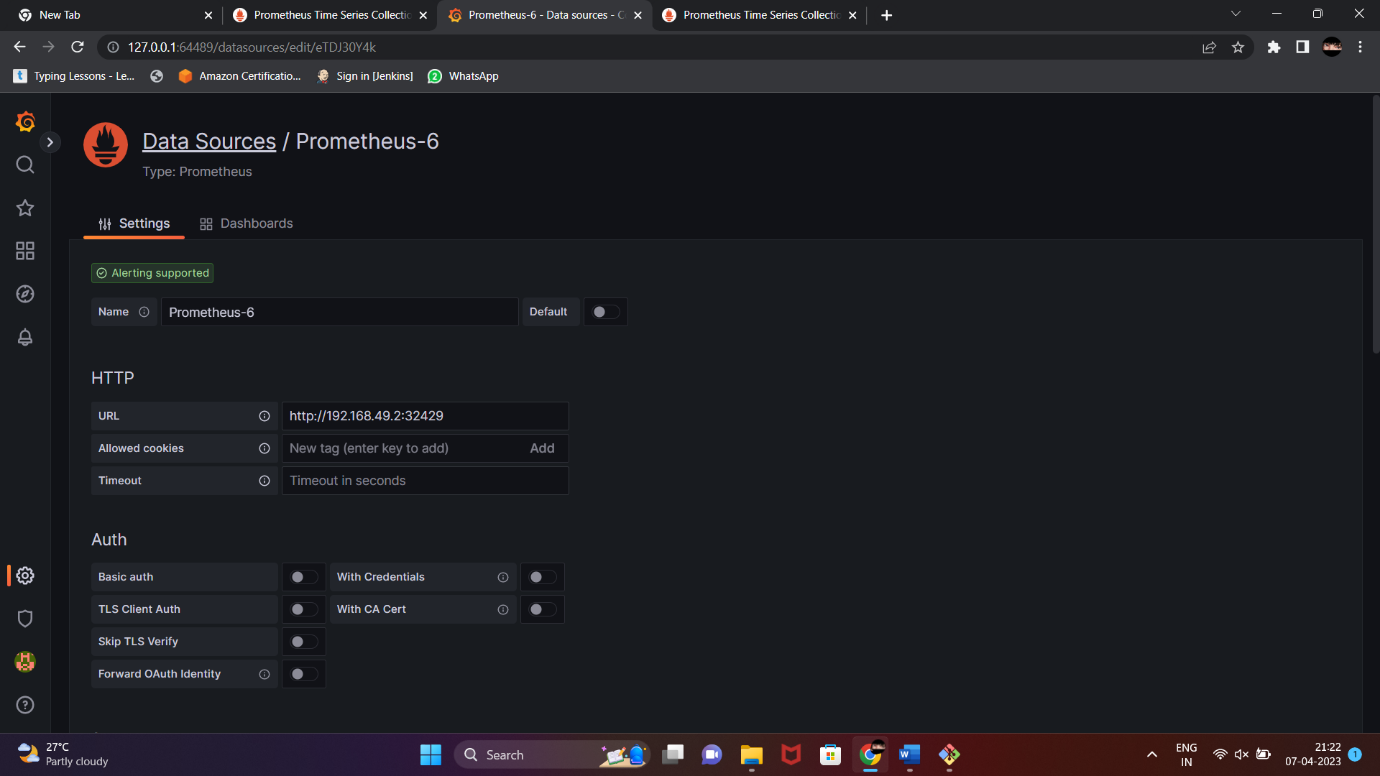


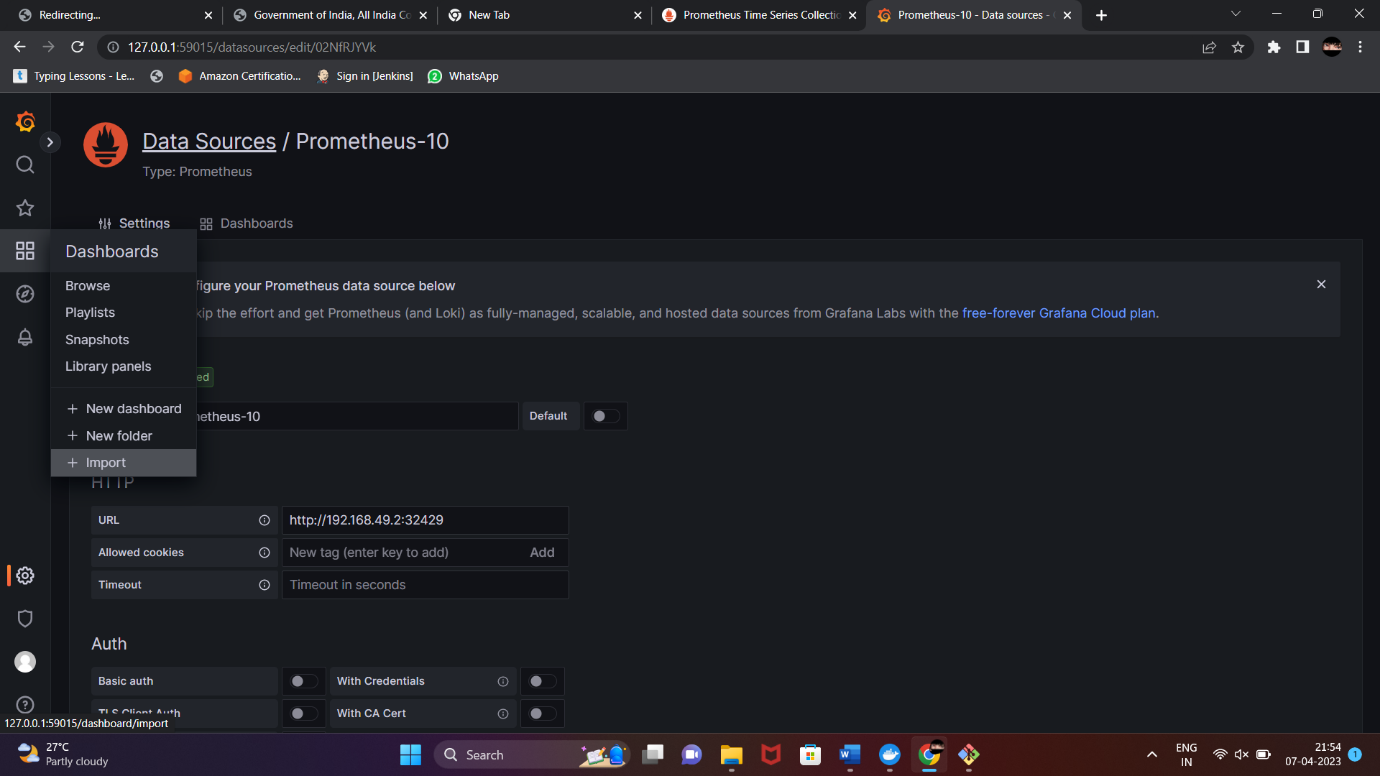


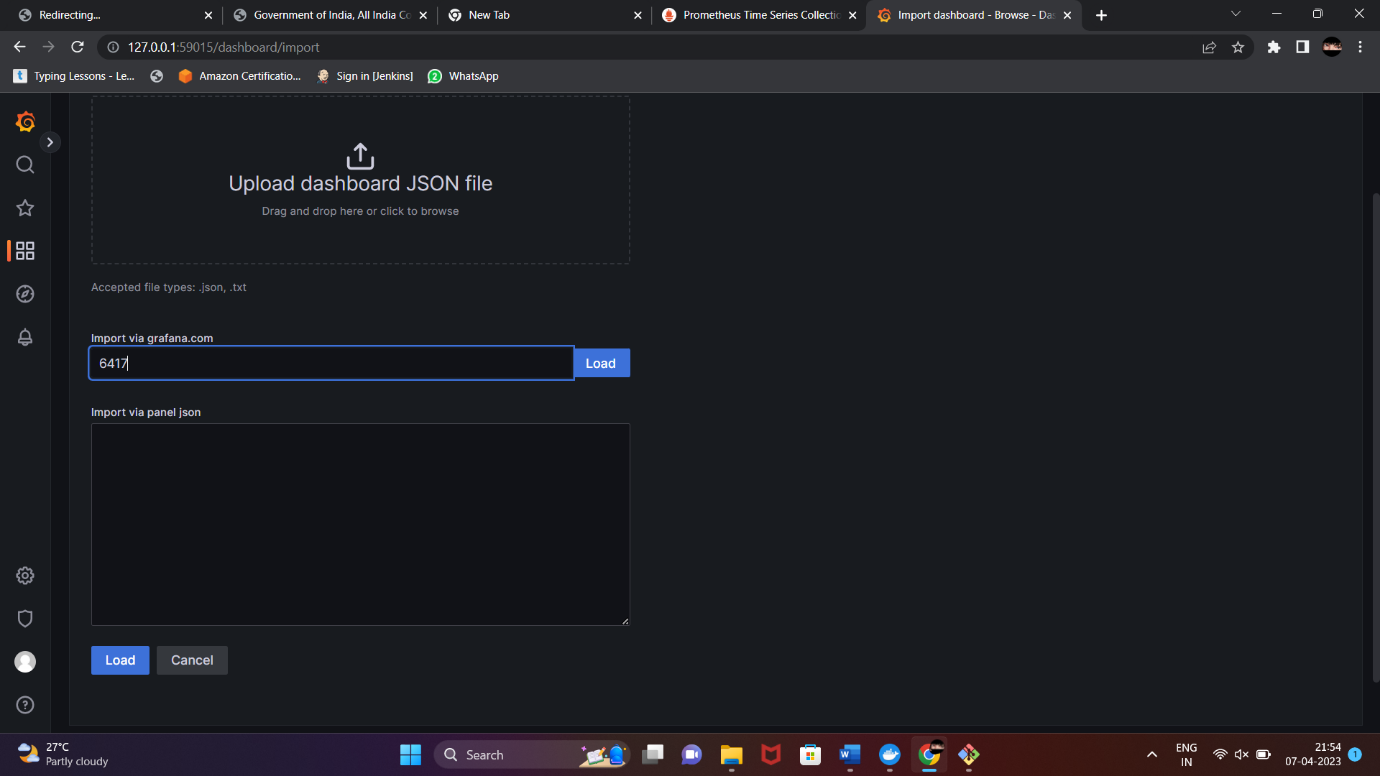


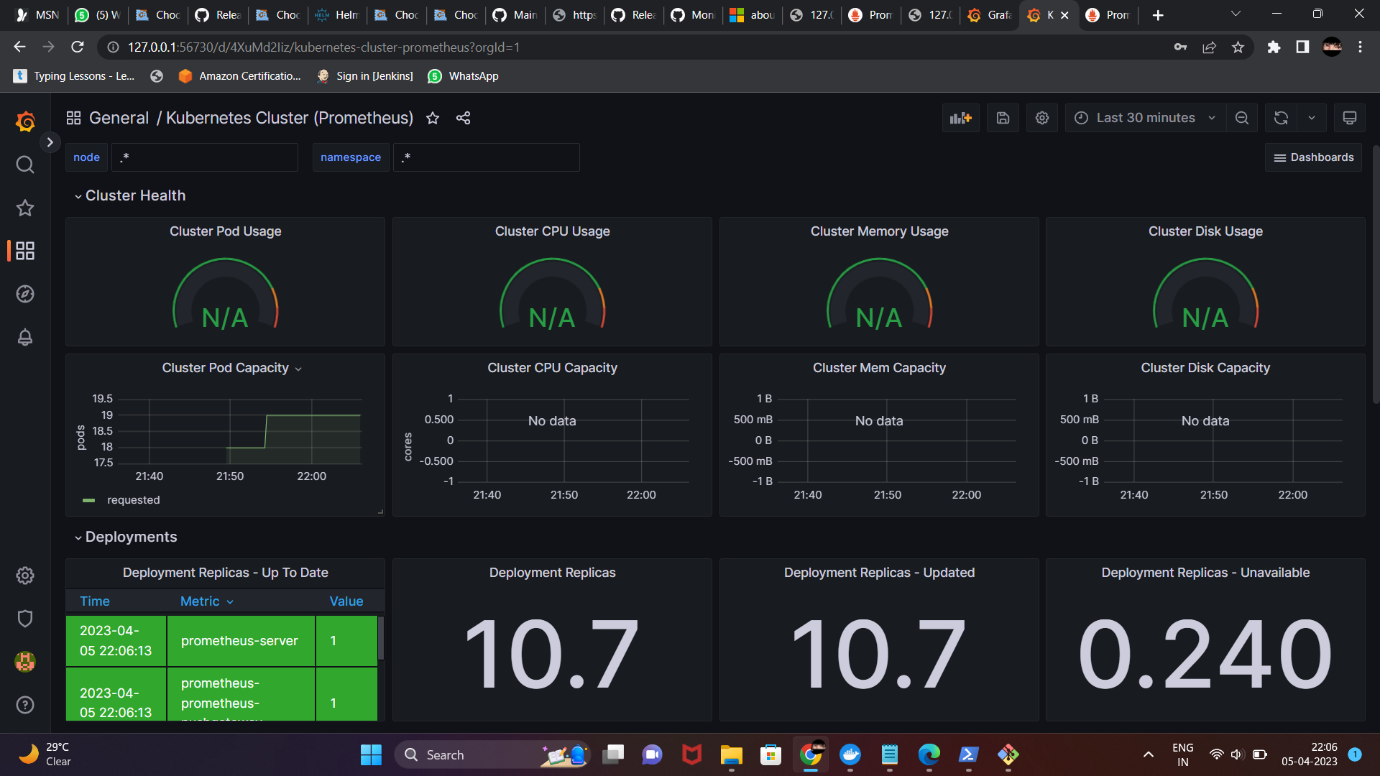












Conclusion: Overall, the application and database orchestration using Kubernetes project is a valuable tool that can help organizations to streamline their development and deployment processes and enable them to quickly respond to changing business requirements.