Cluster monitoring with Prometheus and Grafana

**Description**: created a minikube cluster using docker as a driver and by using helm installed Prometheus and Grafana for monitoring the cluster and visualizing the parameters.

Why monitoring tools?

In real world environment there would be hundreds of Kubernetes clusters and it is impossible for the human beings to monitor each cluster. So, we use monitoring tools.

Monitoring your Kubernetes cluster is essential for ensuring the health and performance of your applications and infrastructure. Here are some reasons why monitoring your Kubernetes cluster is important:

* Identify issues and troubleshoot: By monitoring your Kubernetes cluster, you can quickly identify issues such as application crashes, resource bottlenecks, and network problems. With real-time monitoring, you can troubleshoot issues before they escalate and impact your users.
* Optimize performance and capacity: Monitoring allows you to track the performance of your applications and infrastructure over time and identify opportunities to optimize performance and capacity. By understanding usage patterns and resource consumption, you can make informed decisions about scaling your infrastructure and improving the efficiency of your applications.
* Ensure high availability: Kubernetes is designed to provide high availability for your applications, but this requires careful monitoring and management. By monitoring your cluster and setting up alerts, you can ensure that your applications remain available even in the event of failures or unexpected events.
* Security and compliance: Monitoring your Kubernetes cluster can help you identify potential security risks and ensure compliance with regulations and policies. By tracking access logs and other security-related metrics, you can quickly detect and respond to potential security threats.

Why Prometheus?

Prometheus is an open-source monitoring and alerting system that helps you collect and store metrics about your software systems and infrastructure and analyze that data to gain insights into their health and performance. It provides a powerful query language, a flexible data model, and a range of integrations with other tools and systems. With Prometheus, you can easily monitor metrics such as CPU usage, memory usage, network traffic, and application-specific metrics, and use that data to troubleshoot issues, optimize performance, and create alerts to notify you when things go wrong.

Why Prometheus over other monitoring tools?

Prometheus is a popular choice for Kubernetes monitoring for several reasons:

* Open-source: Prometheus is an open-source project that is free to use and has a large community of contributors. This means that you can benefit from ongoing development, bug fixes, and feature enhancements without paying for a commercial monitoring solution.
* Native Kubernetes support: Prometheus is designed to work seamlessly with Kubernetes, making it easy to deploy and integrate with your Kubernetes environment. It provides pre-configured Kubernetes dashboards and supports auto-discovery of Kubernetes services and pods.
* Powerful query language: Prometheus provides a powerful query language that allows you to easily retrieve and analyze metrics data. This allows you to create custom dashboards and alerts, and to troubleshoot issues more easily.
* Scalability: Prometheus is designed to be highly scalable, allowing you to monitor large and complex Kubernetes environments with ease. It supports multi-node architectures and can handle large volumes of data without significant performance degradation.
* Integrations: Prometheus integrates with a wide range of other tools and systems, including Grafana for visualization, Alert manager for alerting, and Kubernetes API server for metadata discovery.

Why Grafana?

Grafana is a popular open-source data visualization and analytics platform that allows you to create custom dashboards and visualizations based on a variety of data sources. Grafana is often used for monitoring and analyzing metrics and logs in real-time, making it an ideal tool for monitoring systems and applications, including Kubernetes environments.

Grafana supports a wide range of data sources, including databases, time-series databases, and other data storage systems. It provides a powerful query language that allows you to retrieve and analyze data from these sources, and to create custom dashboards and alerts based on that data.

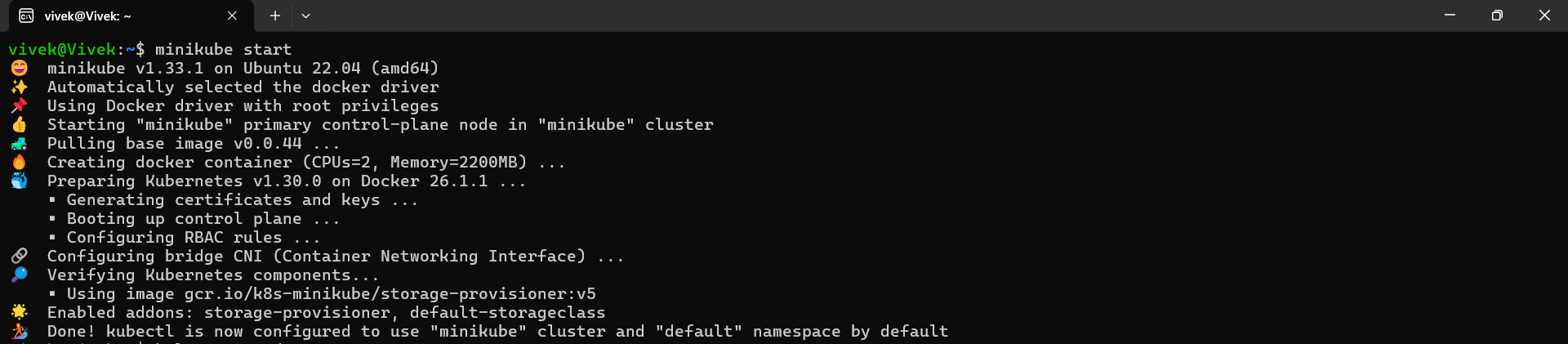
In addition to its powerful data visualization and analysis capabilities, Grafana is also highly extensible. It supports a wide range of plugins and integrations, including integrations with popular monitoring and logging tools like Prometheus, Elasticsearch, and InfluxDB.

**Procedure:**

1. Install the minikube, Docker and start the minikube by using the command

“ minikube start ” or “ minikube start –driver=docker ”.

If docker is installed, by default minikube uses docker as driver.



1. Prometheus installation using Helm:

#Add helm repo using command

“helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts> ”

#Update helm repo by executing below command

“helm repo update”

# Install helm

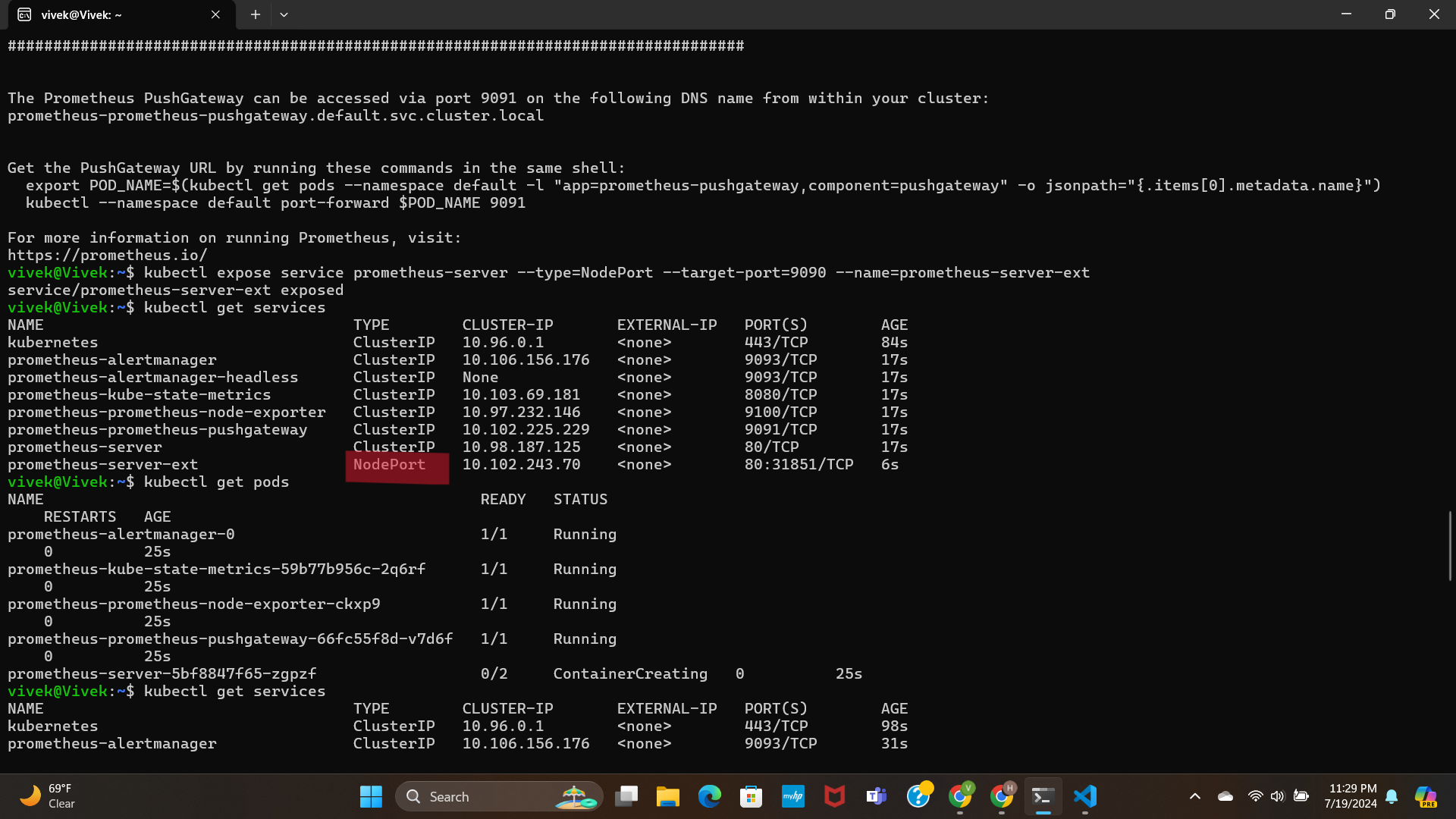
“helm install prometheus prometheus-community/prometheus”

A screenshot of a computer screen

Description automatically generated

#Expose Prometheus Service

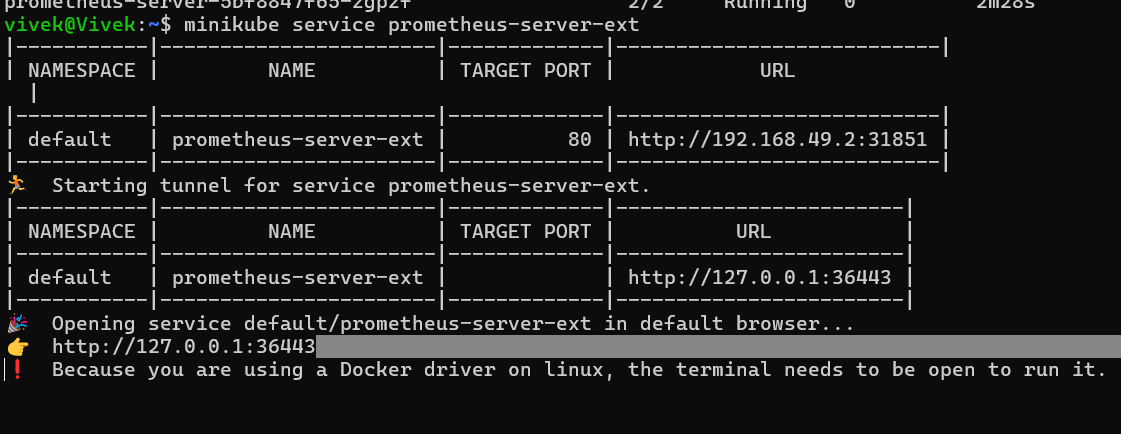
This is required to access prometheus-server using local host

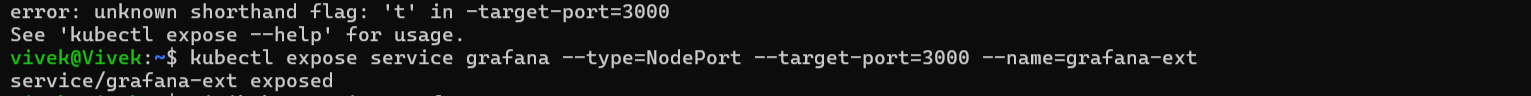
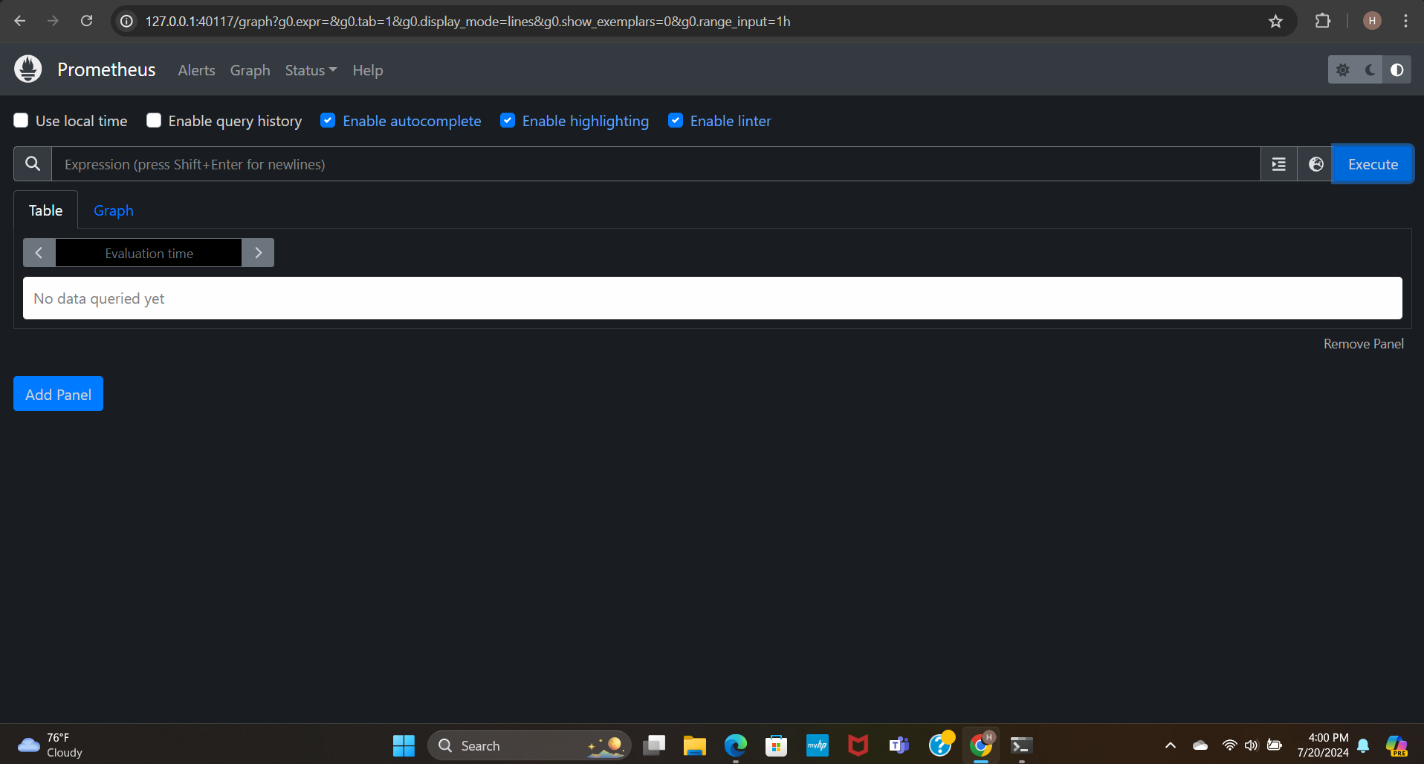
“kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext”

As we using docker as driver on ubuntu machine, we unable to access the prometheus even after exposing the pod(i.e changing the service type as NodePort). The terminal needs to be open to run it. So, we use command

“ minikube service prometheus-server-ext”

Which provides url as output to access the prometheus. Because when using docker driver on linux, the terminal needs to be open to run it.





But once we stop the tunnel we unable to access prometheus on local host. Kept it open and open a new ubuntu terminal for executing other things.

1. Grafana Installation using Helm

# Add helm repo

“ helm repo add grafana <https://grafana.github.io/helm-charts> ”

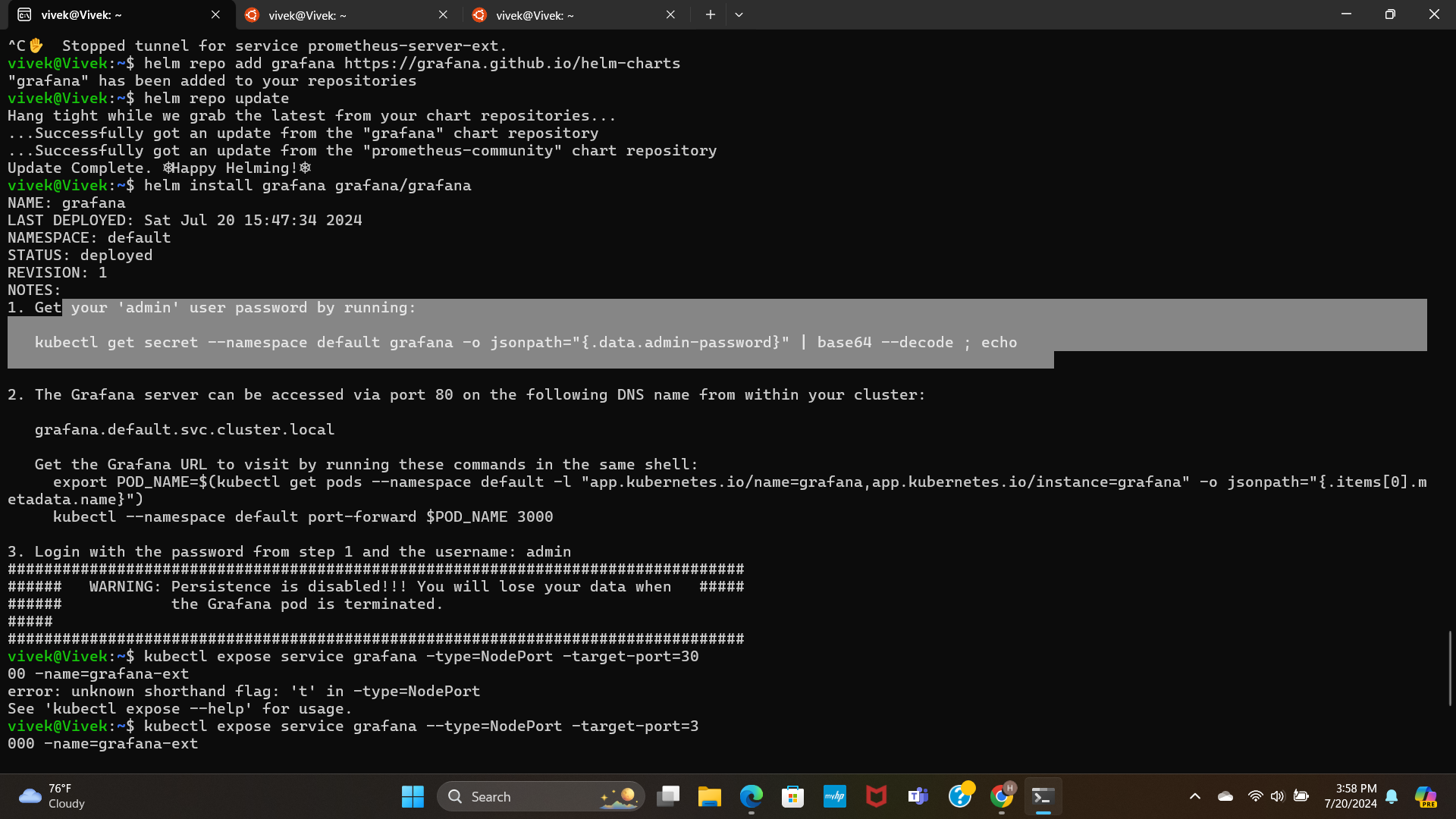
#Update helm repo

“helm repo update”

# Install helm

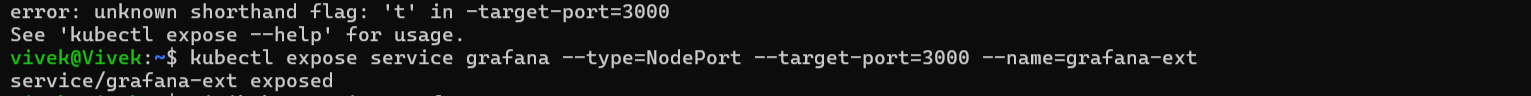
“helm install grafana grafana/grafana ”

After installation, it provides user name and command to get password for login.



#Expose Grafana Service

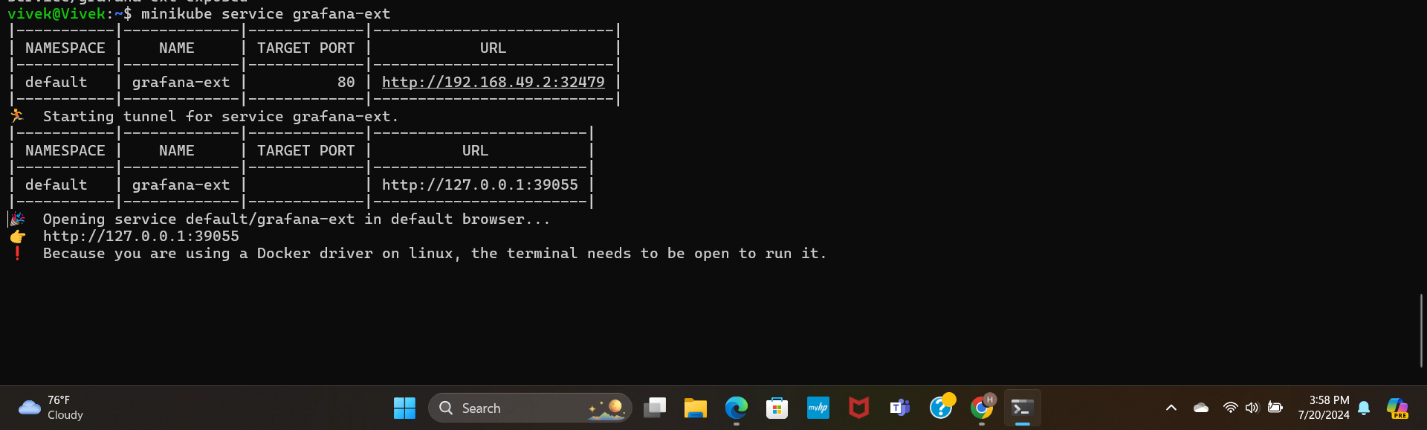
“ kubectl expose service grafana — type=NodePort — target-port=3000 — name=grafana-ext ”



As we using docker as driver on ubuntu machine, we unable to access the prometheus even after exposing the pod(i.e changing the service type as NodePort). The terminal needs to be open to run it. So, we use command

“ minikube service grafana-ext”

Which provides url as output to access the grafana. Because when using docker driver on linux, the terminal needs to be open to run it.



A screenshot of a computer

Description automatically generated

1. After login into grafana we need to add data source to get the data to visualize. In our case it is prometheus.

To make connection between grafan and promethues, we need to give prometheus server url. A screenshot of a computer

Description automatically generated

Next we need to create dashboard. I have imported default dashboard 3662 (standard grafana dasnboard templete) which is provided by grafana. A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

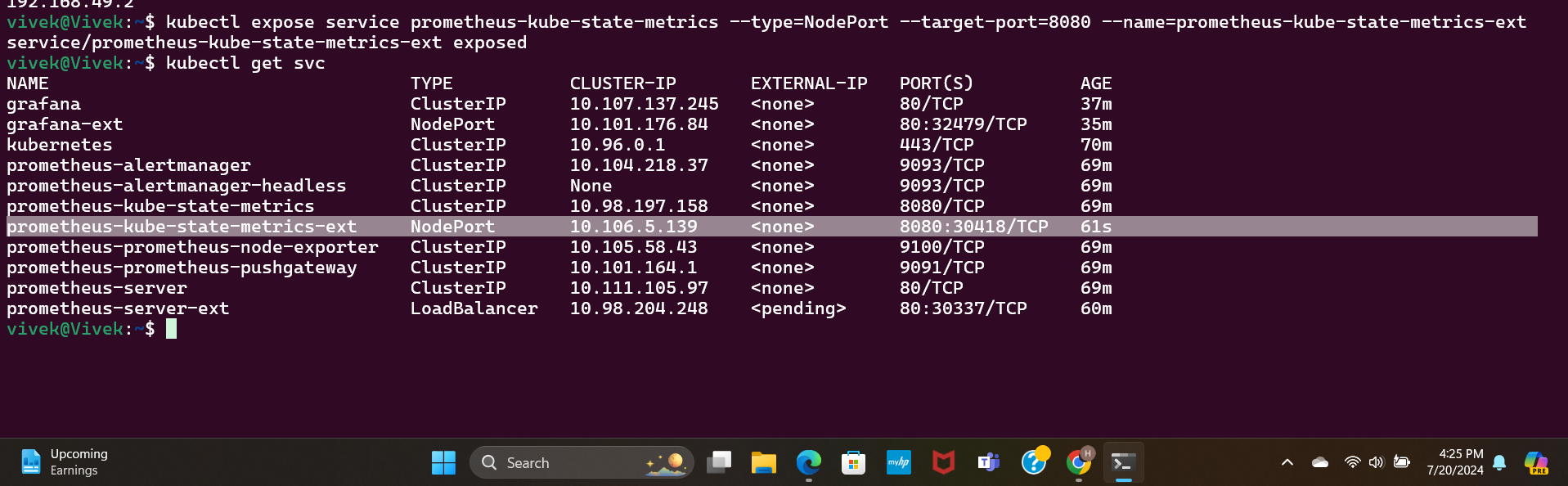
From the dashboard we can see that only few parameters are accessed using above prometheus server.

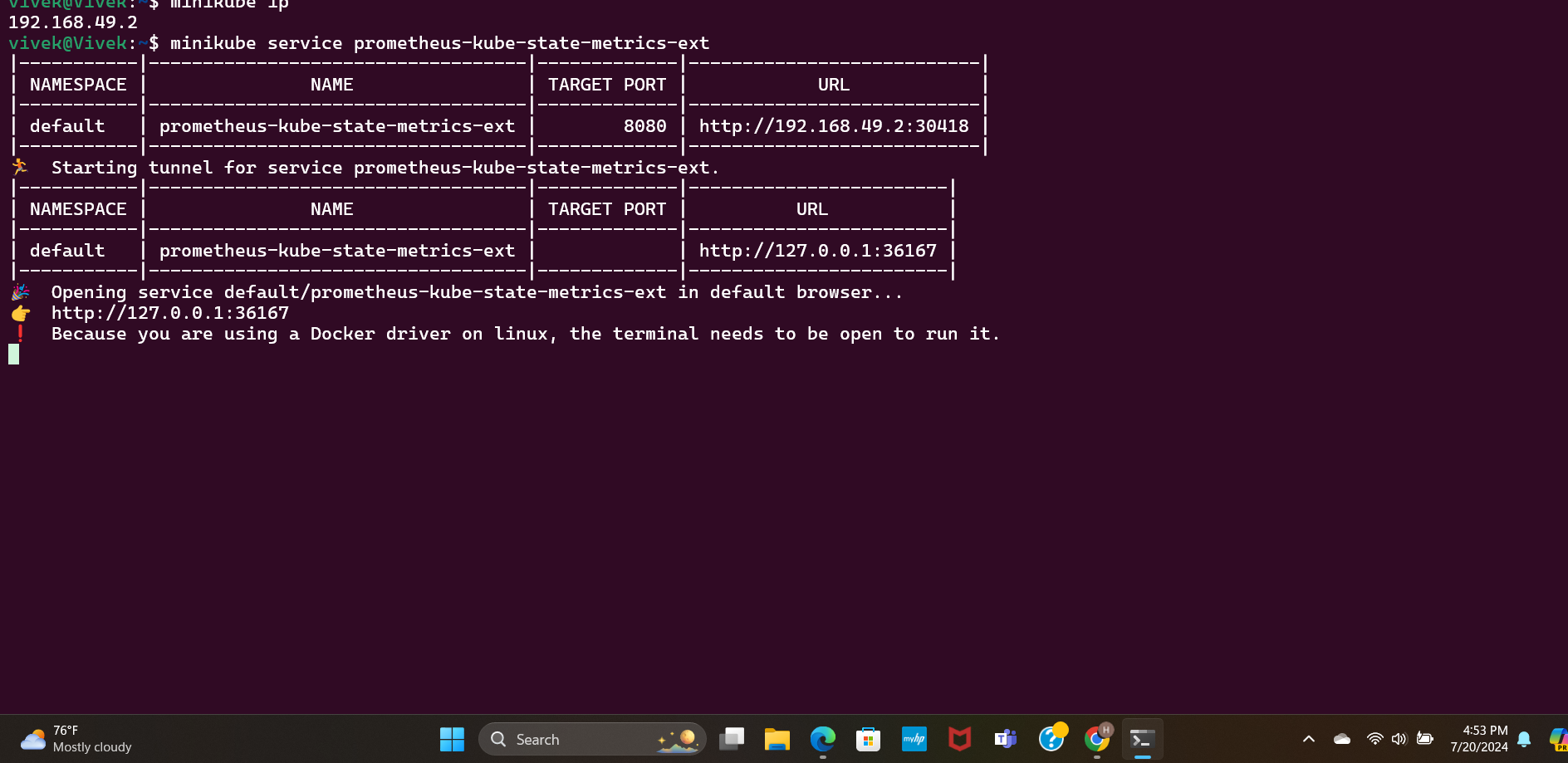
If organization requires more information we can expose the promethues-kube-state-metrics service and get lot more information about cluster.

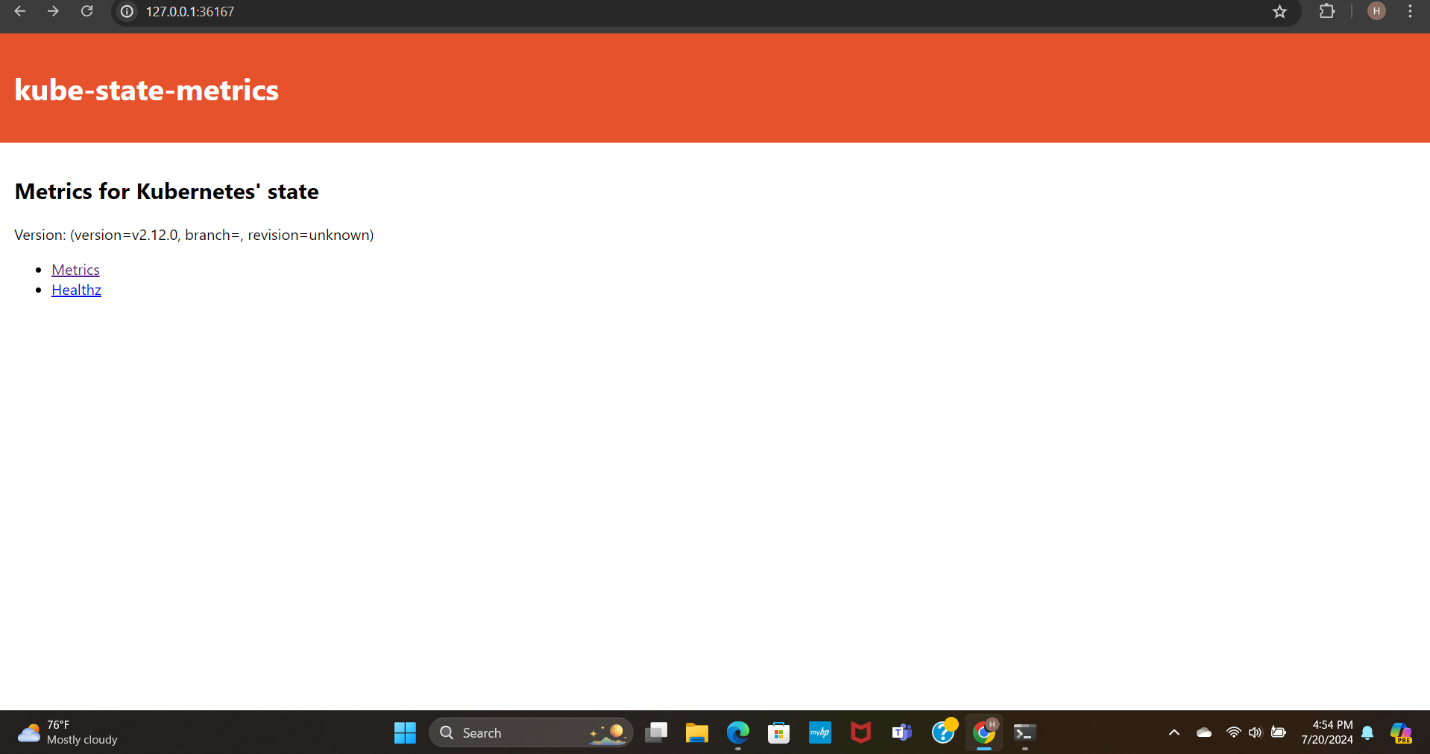
By using the command

“ kubectl expose service prometheus-kube-state-metrics –type=NodePort –target-port=8080 –name=prometheus- kube-state-metrics-ext ”

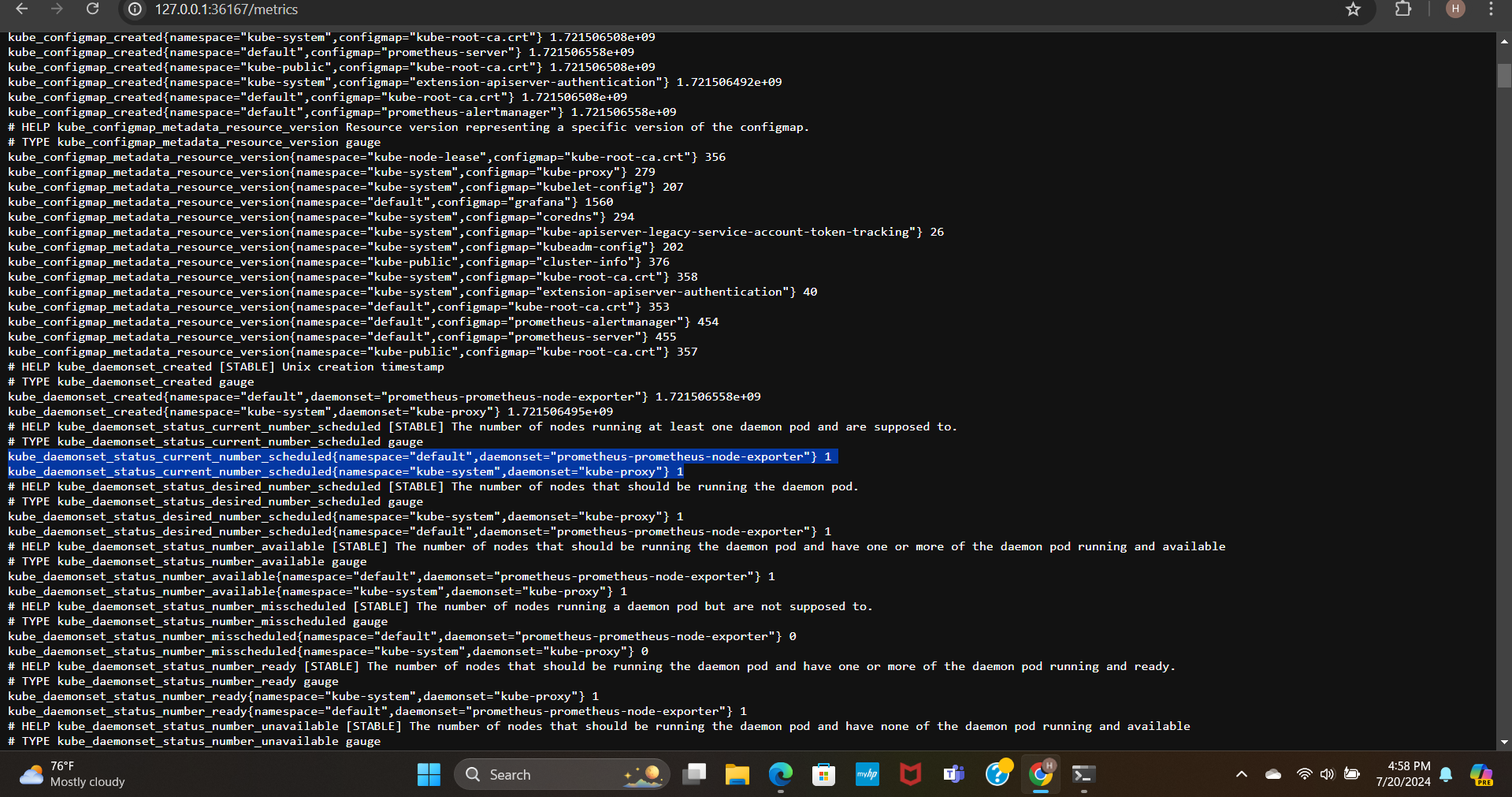
Will expose the kube-state-metrics.

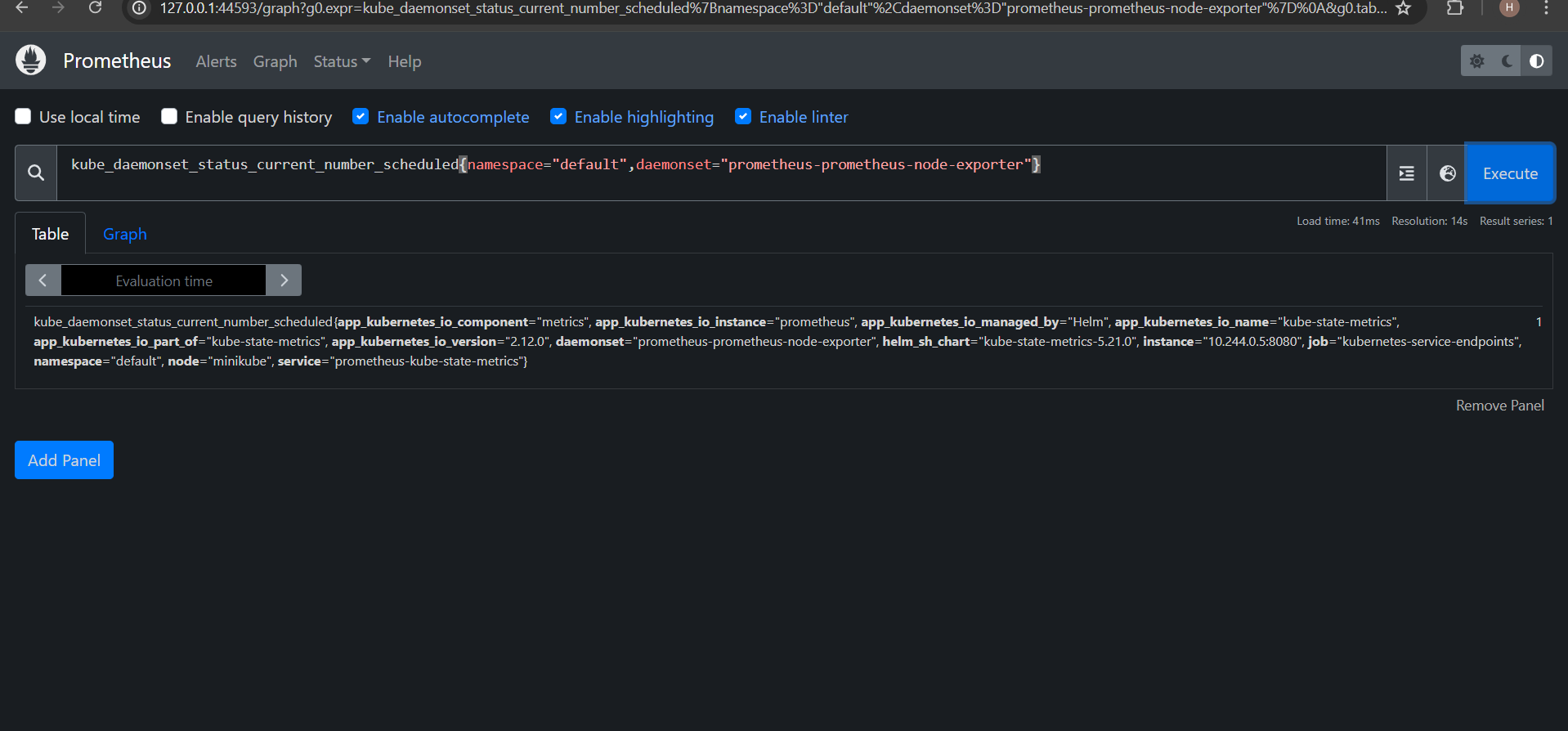


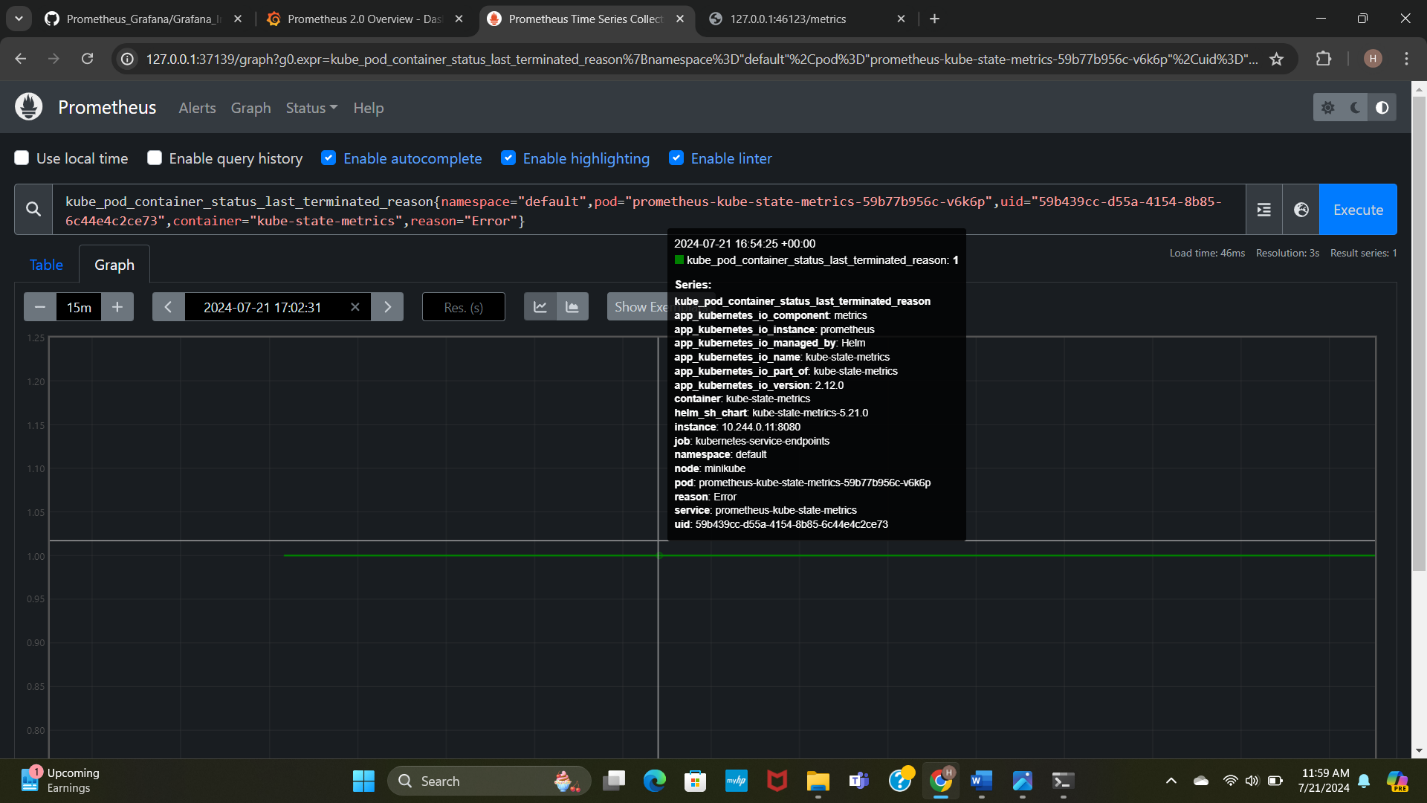




Kube-state-metrics will give the prometheus quiers i.e promQueries.



By copying and pasting the queries into prometheus, we can view the parameters as table or graph.



We can get this information inside prometheus by editing the configMap of prometheus service and adding new job on it.

**Conclusion:**

Monitoring tools prometheus along with grafana monitors the cluster every second and stores the data. Visualizes the data using grafana dashboards which will help the users to keep track health of every cluster. If we need to check the health of applicaton, we need to ask developers to write metrics server and then updating the metrics server host address in configMap of prometheus server.

**Future Scope:**

We can also add alerts to the monitoring tools which will send mails/ messages to user when there is error or cluster downtime.

**Error and Solution:**

Error: if we use docker as driver on linux, we unable to access the prometheus even after exposing the service type as nodeport.  
  
Solution: used command “ minikube service prometheus-server-ext “ which provided url as output to access the prometheus. Because when using docker driver on linux, the terminal needs to be open to run it.  
  
But once we stop the tunnel we unable to access prometheus on local host. So, it is better to use virtual box(windows)/ hyperkit(mac) as driver for minikube**.**