Contents

- Example Dashboard

1. Security Trends

1.1. Authentication

Authentication is the process of verifying the identity of a user, device, or system. Modern authentication methods go beyond simple passwords to include multifactor authentication (MFA), biometrics, and federated identity systems. These methods enhance security by making it harder for attackers to gain unauthorized access.

- **Multifactor Authentication (MFA)**: MFA requires users to provide two or more verification factors to gain access. These factors can include something the user knows (password), something the user has (security token), and something the user is (biometric verification).
- **Biometric Authentication**: Biometric systems use unique physical characteristics such as fingerprints, facial recognition, and iris scans. These methods are highly secure as they are difficult to replicate or steal.
- **Federated Identity**: Federated identity management allows users to use a single set of credentials across multiple systems. Services like SAML, OAuth, and OpenID Connect facilitate this by enabling trust relationships between different identity providers and service providers.

1.2. Authorization

Authorization determines what resources a user can access and what actions they can perform. Modern authorization strategies include role-based access control (RBAC), attribute-based access control (ABAC), and policy-based access control (PBAC).

- **Role-Based Access Control (RBAC)**: RBAC assigns permissions to users based on their roles within an organization. This simplifies management and ensures that users have access only to what they need to perform their job functions.
- **Attribute-Based Access Control (ABAC)**: ABAC considers multiple attributes (user role, department, time of day, etc.) to determine access rights. This provides a more granular and flexible approach to access control compared to RBAC.
- **Policy-Based Access Control (PBAC)**: PBAC uses predefined policies to control access. Policies are typically written in a language that specifies conditions under which access is granted or denied.

1.3. Encryption

Encryption is a critical component of modern security, protecting data both in transit and at rest. Encryption algorithms convert plaintext data into ciphertext, which can only be deciphered by someone with the correct decryption key.

- **Symmetric Encryption**: In symmetric encryption, the same key is used for both encryption and decryption. Common algorithms include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).
- **Asymmetric Encryption**: Asymmetric encryption uses a pair of keys – a public key for encryption and a private key for decryption. RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) are popular asymmetric encryption algorithms.
- **End-to-End Encryption (E2EE)**: E2EE ensures that data is encrypted on the sender's side and only decrypted on the recipient's side, preventing intermediaries from accessing the data. This is commonly used in messaging apps and secure communication channels.

1.4. Cybersecurity Current Trends

The field of cybersecurity is constantly evolving to address new threats and challenges. Key trends in 2024 include:

- **Zero Trust Security**: Zero Trust is a security model that assumes no implicit trust and verifies every request as if it originates from an open network. Core principles include continuous verification, least privilege, and micro segmentation.
- **Extended Detection and Response (XDR)**: XDR integrates multiple security products into a cohesive system that provides a holistic view of threats across an organization. It enhances threat detection and response through comprehensive visibility, advanced threat detection, and automated response.
- **Cloud Security**: With the increasing adoption of cloud services, securing cloud environments has become critical. Key aspects of cloud security include Cloud Security Posture Management (CSPM), Cloud Workload Protection Platforms (CWPP), and Secure Access to Cloud Services (SACS).
- **IoT Security**: The proliferation of IoT devices introduces new security challenges, as these devices can be entry points for attackers. Key IoT security measures include device authentication and authorization, firmware updates and patching, and network segmentation.

- **Artificial Intelligence in Security**: AI and machine learning are transforming security by enabling more effective threat detection, incident response, and predictive analytics. AI identifies patterns and anomalies, automates response actions, and predicts future threats based on historical data and trends.

1.5. Access Control and Accounting

Access control is the selective restriction of access to data, resources, or systems. Modern access control strategies include:

- **Identity and Access Management (IAM)**: IAM solutions manage digital identities and control access to resources. They include functionalities like single sign-on (SSO), multi-factor authentication (MFA), and role-based access control (RBAC).
- **Network Access Control (NAC)**: NAC solutions enforce security policies on devices accessing the network, ensuring that only compliant devices are allowed to connect.
- **Physical Access Control**: This includes mechanisms like biometric scanners, keycards, and security guards to control physical access to facilities and sensitive areas.
- **Accounting (Audit Logs)**: Accounting involves maintaining logs of user activities and access events. Audit logs are crucial for detecting and investigating security incidents, ensuring compliance, and providing forensic evidence in the event of a breach.
- **Logging and Monitoring**: Continuous logging and monitoring of user activities, system events, and network traffic help detect anomalies and potential security breaches in real time.

---

2. Kubernetes

2.1. Introduction to Kubernetes

Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. Developed by Google, Kubernetes has become the de facto standard for container orchestration due to its robust feature set, including:

- **Automated Rollouts and Rollbacks**: Kubernetes manages application updates and can roll back changes if something goes wrong.

- **Self-Healing**: Kubernetes automatically replaces failed containers, ensuring that applications remain available.

- **Horizontal Scaling**: Kubernetes scales applications automatically based on resource usage and demand.

- **Service Discovery and Load Balancing**: Kubernetes distributes traffic across containers and provides internal DNS for service discovery.

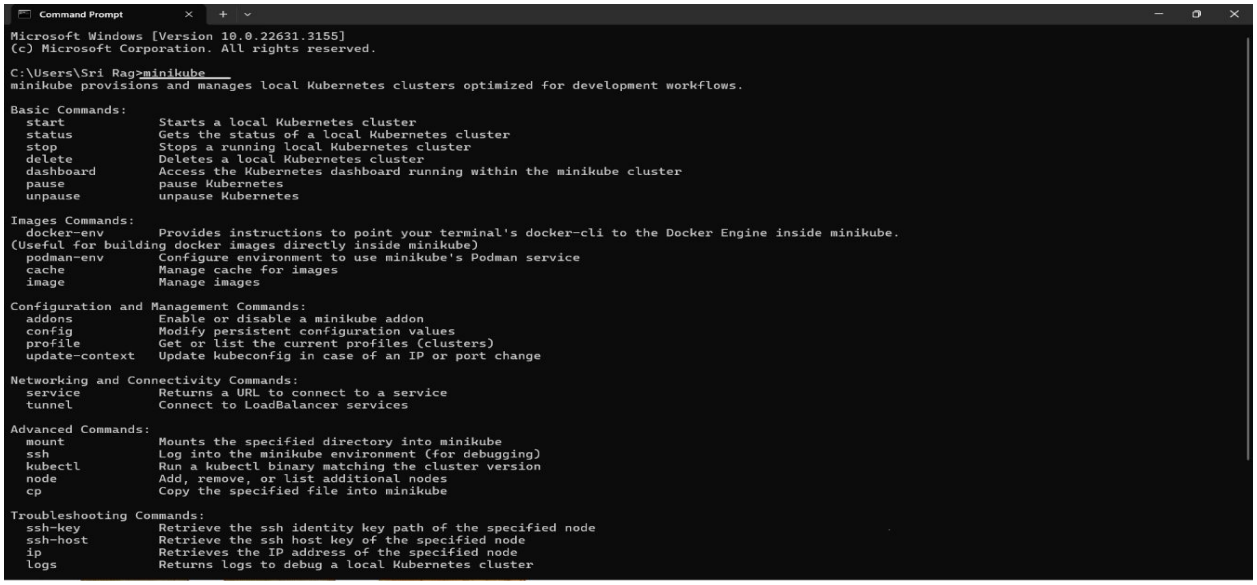
2.2. Creation of Minikube Cluster

Minikube is a tool that allows you to run Kubernetes locally. It provides a simple way to create a single-node Kubernetes cluster on your local machine, making it ideal for development and testing.

Steps to Create Minikube

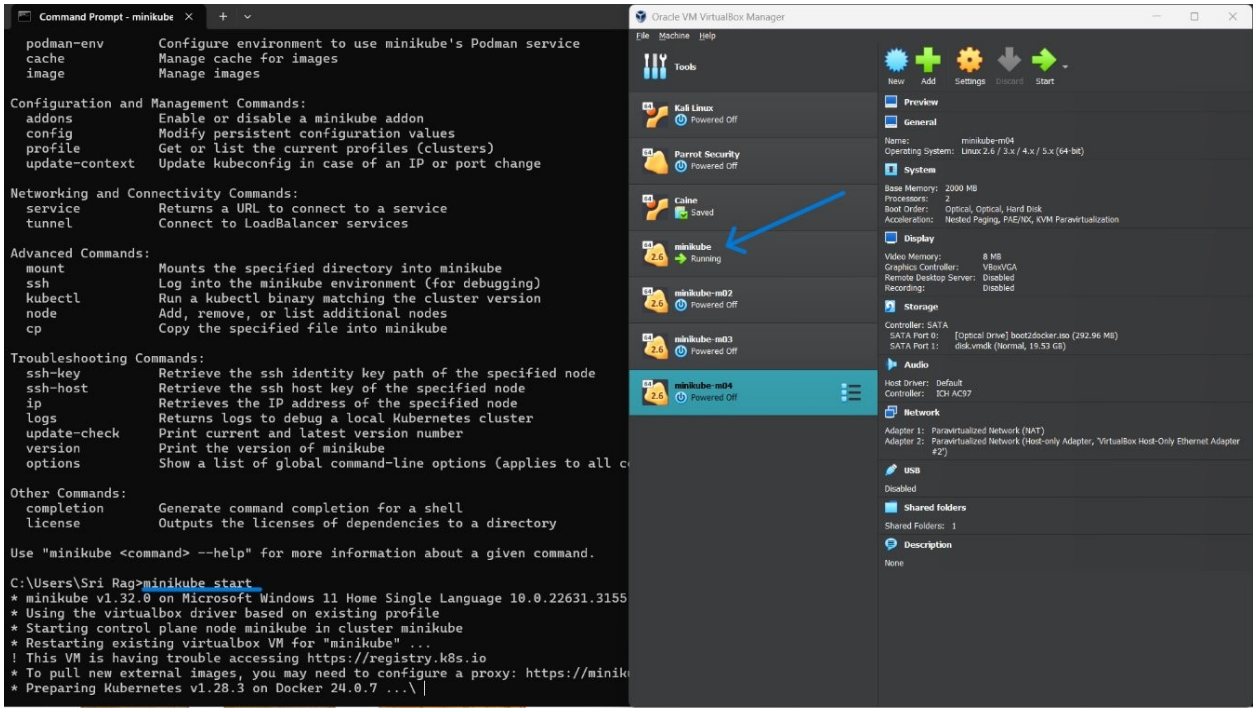
1. **Install Minikube**:

   ```
   curl -LO
   https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
   sudo install minikube-linux-amd64 /usr/local/bin/minikube
   ```



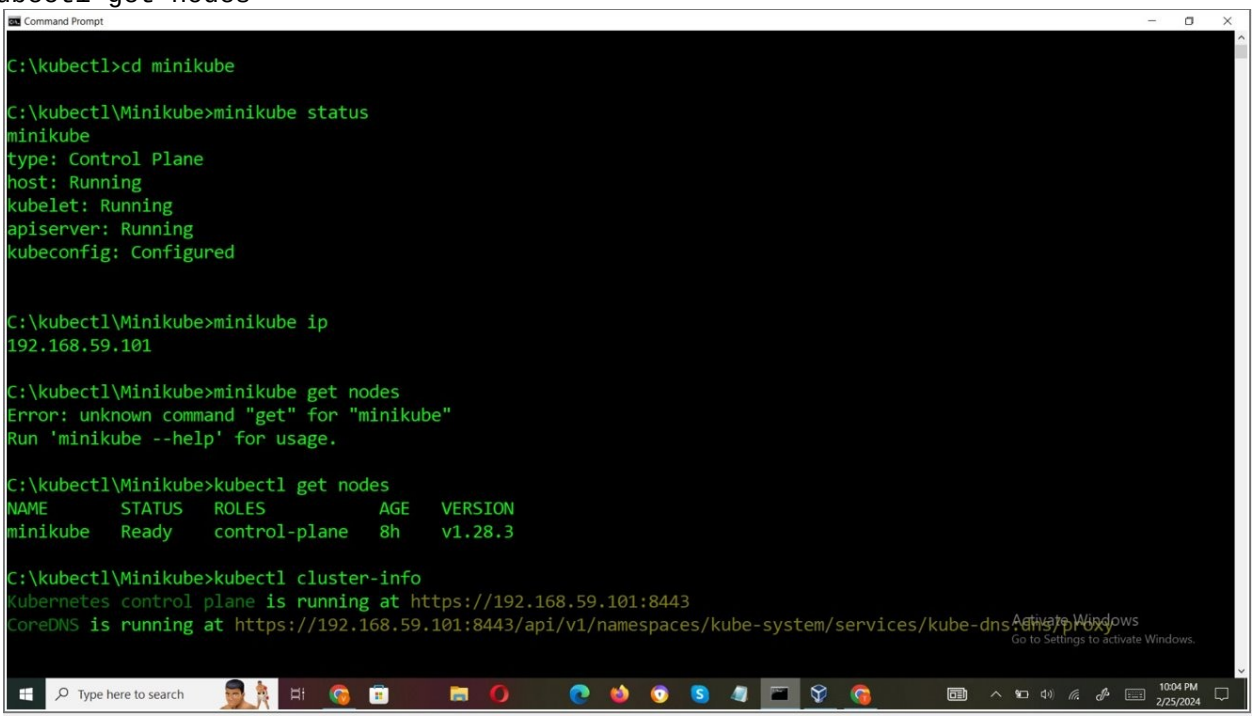
2. **Start Minikube**:

   ```
   minikube start
   ```



3. **Verify Minikube Installation**:

   `kubectl get nodes`

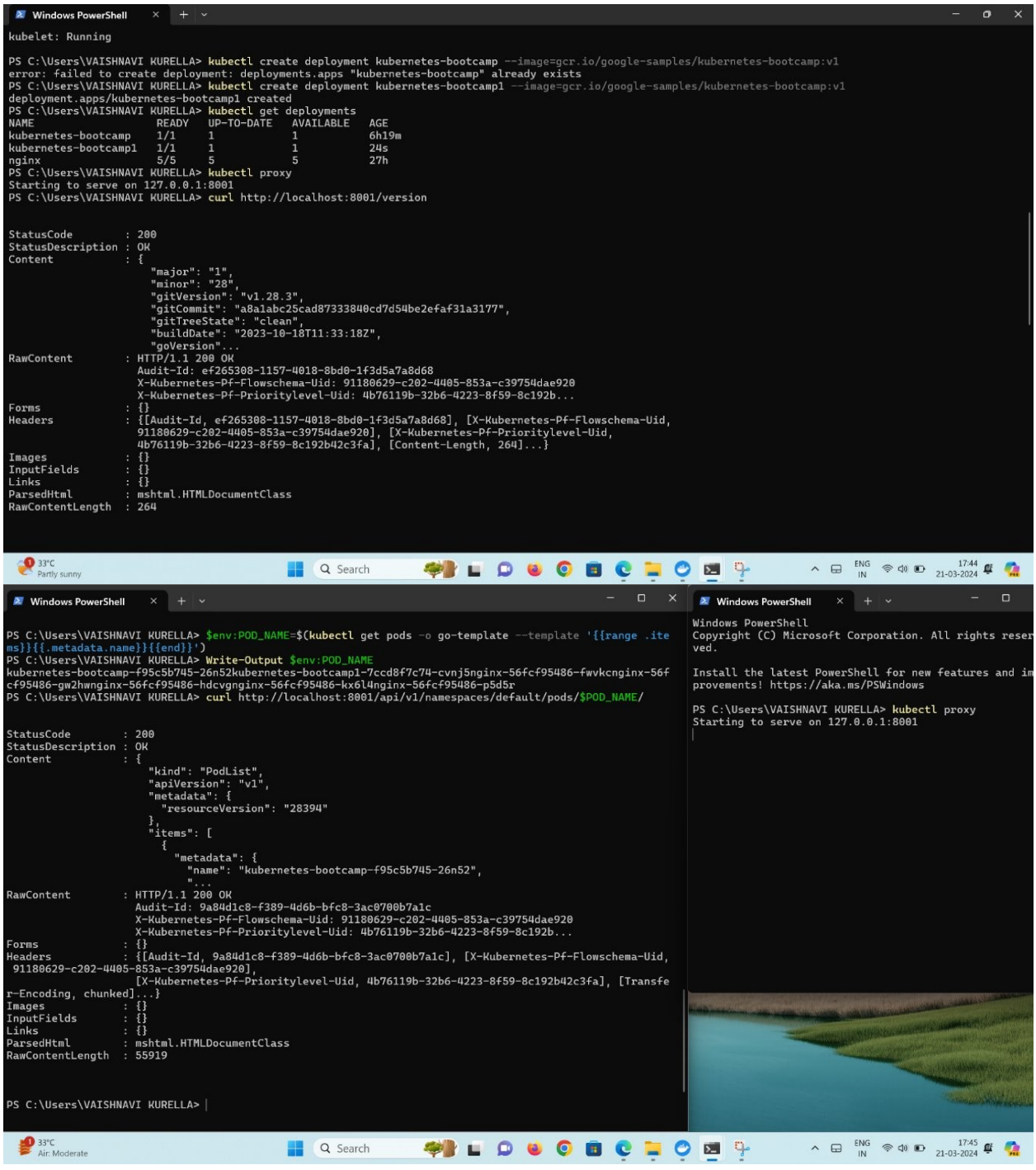


Minikube creates a local Kubernetes cluster with a single node, allowing you to experiment with Kubernetes features and run containerized applications.

2.3. Performing Basic Commands

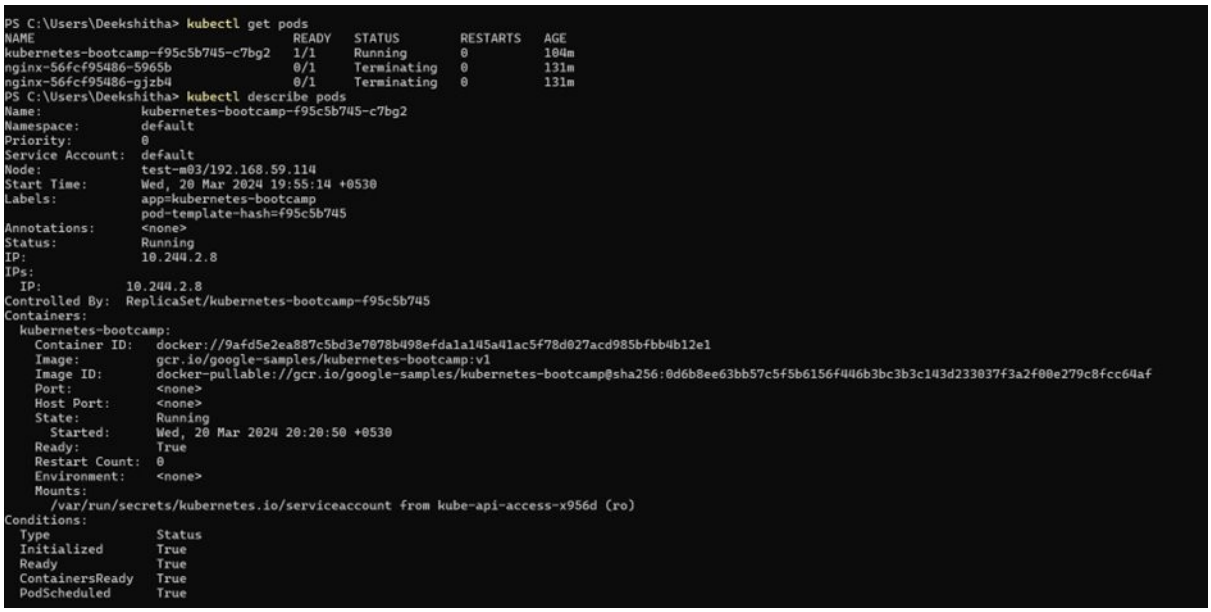
Once Minikube is set up, you can use Kubernetes commands to manage your cluster. Here are some basic commands to get you started:

Deploy an app and view the app



The command `kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1` deploys an application on Kubernetes. This creates a deployment which instructs Kubernetes to run your app encapsulated in a container. To view the application, you can use `kubectl proxy` to establish a connection with the cluster's internal network and then access the pod's API endpoint.

Explore Your App



```
PS C:\Users\Deekshitha> $env:POD_NAME="$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{end}}')"
PS C:\Users\Deekshitha> Write-Output $env:POD_NAME
kubernetes-bootcamp-f95c5b745-c7bg2nginx-56fcf95486-5965b
PS C:\Users\Deekshitha> curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME:8080/


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "kind": "PodList",
                      "apiVersion": "v1",
                      "metadata": {
                        "resourceVersion": "19506"
                      },
                      "items": [
                        {
                          "metadata": {
                            "name": "kubernetes-bootcamp-f95c5b745-c7bg2",
                            "...
RawContent        : HTTP/1.1 200 OK
                    Audit-Id: 6d5f9778-e625-405a-8471-d919a364e05d
                    X-Kubernetes-Pf-Flowschema-Uid: f5cfebcf-3ee5-4979-a285-cd050a145e49
                    X-Kubernetes-Pf-Prioritylevel-Uid: f9572119-ee13-48aa-a5b2-0e95cd...
Forms             : {}
Headers           : {[Audit-Id, 6d5f9778-e625-405a-8471-d919a364e05d], [X-Kubernetes-Pf-Flowschema-Uid, f5cfebcf-3ee5-4979-a285-cd050a145e49],
                    [X-Kubernetes-Pf-Prioritylevel-Uid, f9572119-ee13-48aa-a5b2-0e95cd68c599], [Transfer-Encoding, chunked]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 15967

PS C:\Users\Deekshitha> kubectl logs  $env:POD_NAME
Error from server (NotFound): pods "kubernetes-bootcamp-f95c5b745-c7bg2nginx-56fcf95486-5965b" not found
PS C:\Users\Deekshitha> kubectl logs kubernetes-bootcamp-f95c5b745-c7bg2
Kubernetes Bootcamp App Started At: 2024-03-20T14:50:50.891Z | Running On:  kubernetes-bootcamp-f95c5b745-c7bg2
```

```
PS C:\Users\Deekshitha> kubectl exec kubernetes-bootcamp-f95c5b745-cv4zk -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-f95c5b745-cv4zk
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
PS C:\Users\Deekshitha> kubectl exec -ti kubernetes-bootcamp-f95c5b745-cv4zk -- bash
root@kubernetes-bootcamp-f95c5b745-cv4zk:/# cat server.js
var http = require('http');
var requests=0;
var podname= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v=1\n");
  console.log("Running On:" ,host, "| Total Requests:", ++requests,"| App Uptime:", (new Date() - startTime)/1000 , "seconds", "| Log
Time:",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
root@kubernetes-bootcamp-f95c5b745-cv4zk:/# curl http://localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-f95c5b745-cv4zk | v=1
root@kubernetes-bootcamp-f95c5b745-cv4zk:/#
```

To view the application's output in the terminal, run `kubectl proxy` in a separate terminal. This creates a proxy for the cluster's internal network. Then, get the pod name using `kubectl get pods -o go-template ...` and use it to construct a URL for the pod's API endpoint. Finally, use `curl` with that URL to see the application's output.

Scaling a Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example
  template:
    metadata:
      labels:
        app: example
    spec:
      containers:
      - name: example-container
        image: nginx:latest
        ports:
        - containerPort: 80
```

```
PS C:\Users\THINKPAD> kubectl apply -f example-deployment.yaml

deployment.apps/example-deployment created

PS C:\Users\THINKPAD> kubectl get deployments

NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
example-deployment   0/3     3            0           9s

PS C:\Users\THINKPAD> kubectl scale deployment example-deployment --replicas=4

deployment.apps/example-deployment scaled

PS C:\Users\THINKPAD> kubectl get deployments

NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
example-deployment   4/4     4            4           2m26s

PS C:\Users\THINKPAD> kubectl get pods -o wide

NAME                                  READY   STATUS    RESTARTS   AGE    IP           NODE       NOMINATED NODE   READINESS GATES
example-deployment-54597f9d79-45hrl   1/1     Running   0          3m9s   10.244.0.5   minikube   <none>           <none>
example-deployment-54597f9d79-ftf6v   1/1     Running   0          98s    10.244.0.6   minikube   <none>           <none>
example-deployment-54597f9d79-qdsns   1/1     Running   0          3m9s   10.244.0.4   minikube   <none>           <none>
example-deployment-54597f9d79-zrkt2   1/1     Running   0          3m9s   10.244.0.3   minikube   <none>           <none>
```

Load Balancing

```
PS C:\Users\THINKPAD> kubectl get services

NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   11m
```

```
PS C:\Users\THINKPAD> kubectl apply -f my-service.yaml

service/my-service created

PS C:\Users\THINKPAD> kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        21m
my-service   NodePort    10.104.68.202   <none>        80:30985/TCP   23s
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: NodePort
```

```
PS C:\Users\THINKPAD> kubectl describe pods

Name:             example-deployment-54597f9d79-qdsns
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.103
Start Time:       Thu, 21 Mar 2024 16:31:03 +0530
Labels:           app=example
                  pod-template-hash=54597f9d79
Annotations:      <none>
Status:           Running
IP:               10.244.0.4
IPs:
  IP:           10.244.0.4
Controlled By:  ReplicaSet/example-deployment-54597f9d79
Containers:
  example-container:
    Container ID:   docker://525280fbfa933282530357b18fe8b8700e9c8024d7d2f393757ccc476c44ddab
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:6db391d1c0cfb30588ba0bf72ea999404f2764febf0f1f196acd5867ac7efa7e
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 21 Mar 2024 16:31:44 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qkr52 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-qkr52:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  30m   default-scheduler  Successfully assigned default/example-deployment-54597f9d79-qdsns to minikube
  Normal  Pulling    30m   kubelet            Pulling image "nginx:latest"
  Normal  Pulled     30m   kubelet            Successfully pulled image "nginx:latest" in 2.238s (37.408s including waiting)
  Normal  Created    30m   kubelet            Created container example-container
  Normal  Started    30m   kubelet            Started container example-container


Name:             example-deployment-54597f9d79-zrkt2
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.59.103
Start Time:       Thu, 21 Mar 2024 16:31:03 +0530
Labels:           app=example
                  pod-template-hash=54597f9d79
Annotations:      <none>
Status:           Running
IP:               10.244.0.3
IPs:
  IP:           10.244.0.3
Controlled By:  ReplicaSet/example-deployment-54597f9d79
Containers:
  example-container:
    Container ID:   docker://d1f4c46f4384a67948b2dba32f2d45fb8c8727060009a52470dc710034efa4cc
    Image:          nginx:latest
    Image ID:       docker-pullable://nginx@sha256:6db391d1c0cfb30588ba0bf72ea999404f2764febf0f1f196acd5867ac7efa7e
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 21 Mar 2024 16:31:42 +0530
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4r654 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
```

Scale Down

```
PS C:\Users\THINKPAD> kubectl scale deployments/example-deployment --replicas=2

deployment.apps/example-deployment scaled

PS C:\Users\THINKPAD> kubectl get deployments
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
example-deployment   2/2     2            2           13m

PS C:\Users\THINKPAD> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
example-deployment-54597f9d79-qdsns   1/1     Running   0          13m   10.244.0.4   minikube   <none>           <none>
example-deployment-54597f9d79-zrkt2   1/1     Running   0          13m   10.244.0.3   minikube   <none>           <none>
```

UPDATE YOUR APP-ROLLING UPDATES

```
      labels:
        app: example
    spec:
      containers:
      - name: example-container
```

```
PS C:\Users\THINKPAD> kubectl apply -f example-deployment.yaml

deployment.apps/example-deployment configured

PS C:\Users\THINKPAD> kubectl rollout status deployment/example-deployment

Waiting for deployment "example-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "example-deployment" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "example-deployment" rollout to finish: 1 old replicas are pending termination...
deployment "example-deployment" successfully rolled out
```

Old version

```
      labels:
        app: example
    spec:
      containers:
      - name: example-container
        image: nginx:latest
        ports:
        - containerPort: 80
```

```
PS C:\Users\THINKPAD> kubectl delete deployment example-deployment

deployment.apps "example-deployment" deleted

PS C:\Users\THINKPAD> kubectl delete service my-service

service "my-service" deleted
```

By performing these basic commands, you can gain hands-on experience with Kubernetes and understand how it manages containerized applications.

3. eBPF Technologies

3.1. Introduction to eBPF

eBPF (extended Berkeley Packet Filter) is a powerful technology that allows programs to run within the Linux kernel, providing capabilities for network monitoring, performance profiling, and security enforcement. eBPF programs are loaded into the kernel and attached to various events such as system calls, network packets, and tracepoints.

3.2. Falcon

Falcon is an eBPF-based security platform designed for real-time visibility and threat detection. It monitors system activities and network traffic, providing deep insights into system behavior. Key features include:

- **Real-Time Monitoring**: Falcon leverages eBPF to capture and analyze system events in real time, enabling immediate detection of suspicious activities.
- **Threat Detection**: Using predefined rules and machine learning models, Falcon identifies potential threats and generates alerts for security teams.
- **Forensic Analysis**: Falcon provides detailed logs and traces of system activities, aiding in forensic investigations and incident response.

3.3. Tracee

Tracee is an open-source runtime security and forensics tool that uses eBPF to trace system calls and other events. It is designed to detect anomalies and provide detailed insights into system behavior.

- **Anomaly Detection**: Tracee identifies anomalies by monitoring system calls and comparing them against known patterns of malicious behavior.
- **Detailed Tracing**: Tracee provides detailed traces of system events, including arguments and return values of system calls, aiding in root cause analysis.
- **Extensibility**: Tracee allows users to define custom detection rules and extend its capabilities to suit specific security needs.

3.4. Tetragon

Tetragon is an eBPF-based platform for runtime security and observability. It offers deep visibility into system activities and enforces security policies based on real-time observations.

- **Deep Observability**: Tetragon monitors system calls, network activities, and other kernel events, providing comprehensive insights into system behavior.
- **Security Enforcement**: Tetragon enforces security policies dynamically, based on real-time observations, preventing malicious activities before they can impact the system.
- **Performance Efficiency**: Tetragon's use of eBPF ensures minimal performance overhead, making it suitable for production environments.

3.5. StackRox

StackRox is a security platform for Kubernetes that leverages eBPF to provide deep visibility and runtime protection for containerized applications. It integrates with Kubernetes to enforce security policies and detect threats.

- **Kubernetes Integration**: StackRox integrates seamlessly with Kubernetes, providing visibility and security controls specific to containerized environments.

- **Runtime Protection**: Using eBPF, StackRox monitors system activities and enforces security policies to protect against runtime threats.

- **Compliance Monitoring**: StackRox helps organizations meet compliance requirements by monitoring and auditing Kubernetes configurations and activities.

3.6. Advantages of Tetragon

Tetragon offers several advantages that make it a valuable tool for security and observability:

- *Deep Observability*

  Tetragon provides unparalleled visibility into system behavior by leveraging eBPF to monitor system calls, network activity, and other kernel events. This deep observability allows for the detection of subtle indicators of compromise and provides detailed insights necessary for thorough forensic analysis.

- *Security Enforcement*

  Tetragon enables dynamic enforcement of security policies based on real-time system behavior. This capability ensures that security measures are adaptive and responsive to emerging threats, preventing malicious activity before it can impact the system.

- *Performance Efficiency*

  Tetragon's use of eBPF ensures that its monitoring and enforcement capabilities have minimal performance overhead. This efficiency is critical for maintaining the performance of production systems while ensuring robust security and observability.

- *Flexibility*

  Tetragon supports a wide range of use cases, from runtime protection to compliance monitoring. Its flexibility allows it to be integrated into various environments and workflows, making it a versatile tool for security teams.

3.7. Why We Use Tetragon

We use Tetragon for several reasons:

1. **Comprehensive Monitoring**: Tetragon's ability to monitor system calls, network activity, and other kernel events provides comprehensive coverage of potential attack vectors.
2. **Real-Time Response**: The platform's dynamic enforcement capabilities ensure that security policies can respond to threats in real time, enhancing our security posture.
3. **Minimal Overhead**: Tetragon's eBPF-based approach ensures that security monitoring and enforcement do not degrade system performance.
4. **Integration Capabilities**: Tetragon integrates well with other tools in our security stack, including observability platforms like Grafana, providing a cohesive and efficient security solution.

4. Execution of Policies Using Tetragon

4.1. Deploy on a Cluster

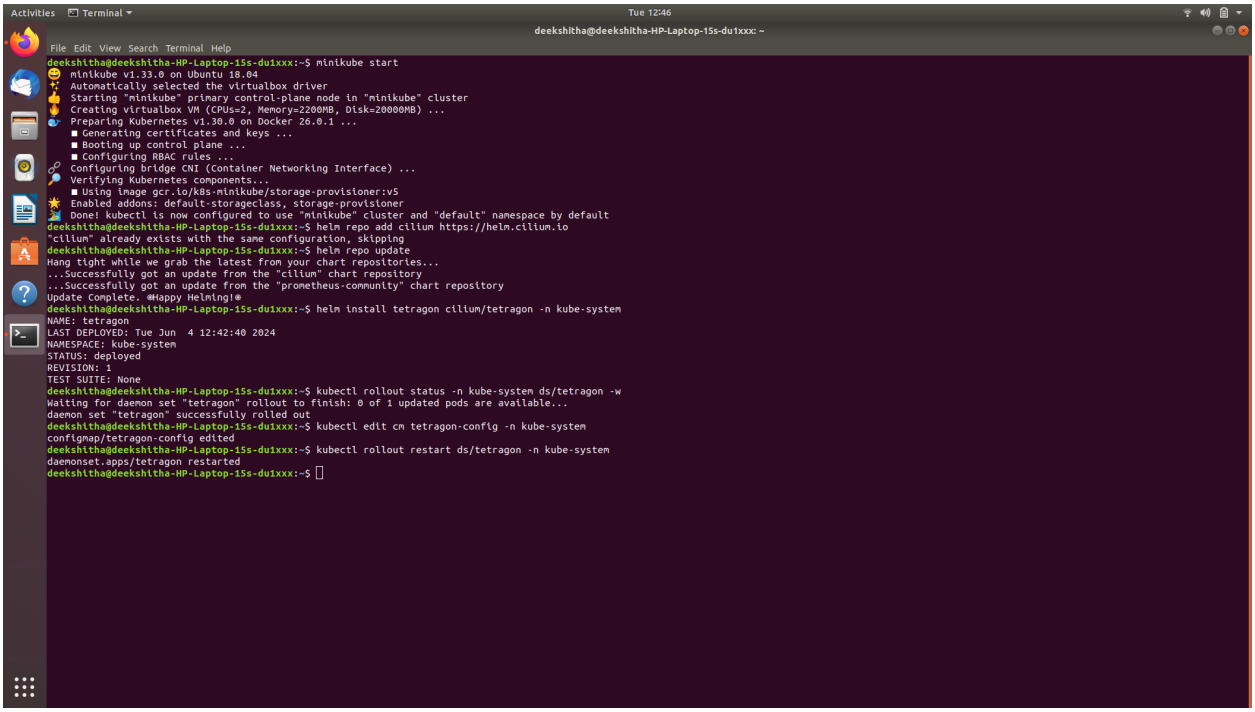
helm repo add cilium https://helm.cilium.io

helm repo update

helm install tetragon cilium/tetragon -n kube-system

To wait until Tetragon deployment is ready, use the following kubectl command:

kubectl rollout status -n kube-system ds/tetragon -w



4.2. Defining Security Policies

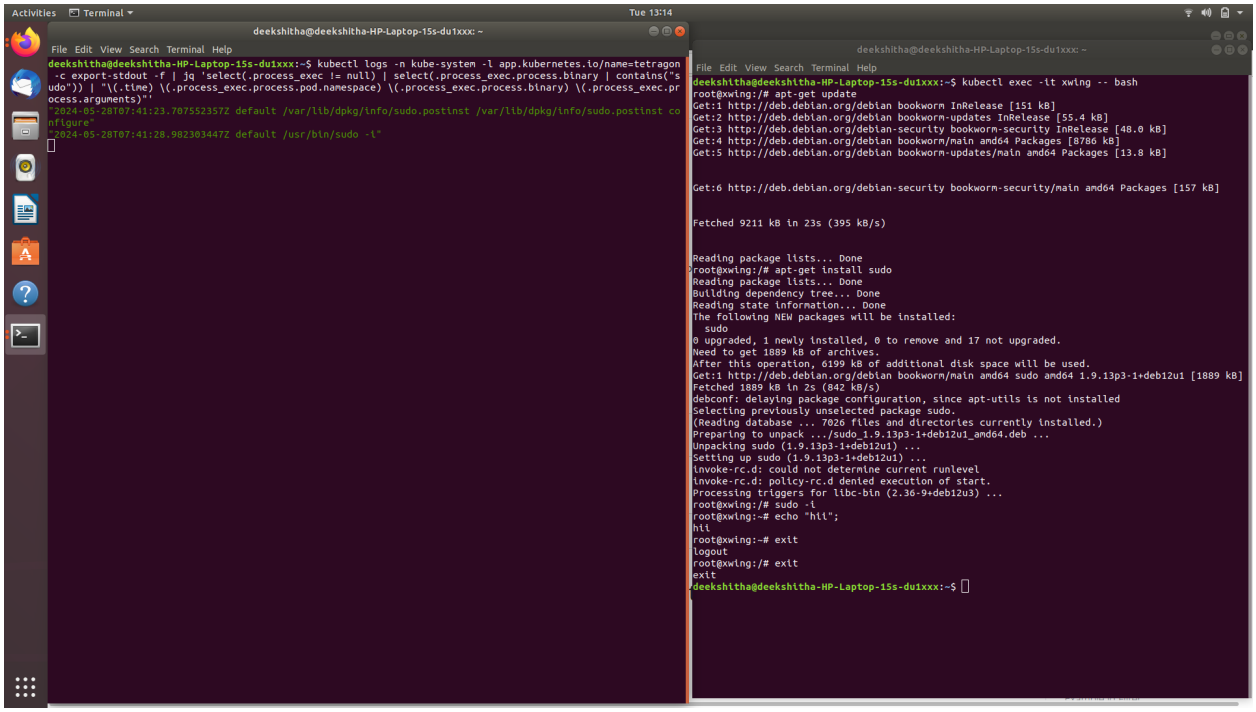
Security policies in Tetragon are defined using eBPF programs that are attached to specific system events. These policies can be tailored to enforce various security measures, such as preventing unauthorized executions, blocking network connections, or monitoring file access.

4.3. Example Policy: Sudo Invocation Monitoring

Use Case

sudo is used to run executables with particular privileges. Creating an audit log of sudo invocations is a common best-practice.

No policy needs to be loaded, standard process execution observability is sufficient.



4.4. Policy Management

Tetragon provides tools for managing policies, including loading, attaching, and detaching eBPF programs. Policies can be dynamically adjusted based on real-time observations, ensuring that security measures remain effective against evolving threats.

5. Integration of Grafana with Tetragon

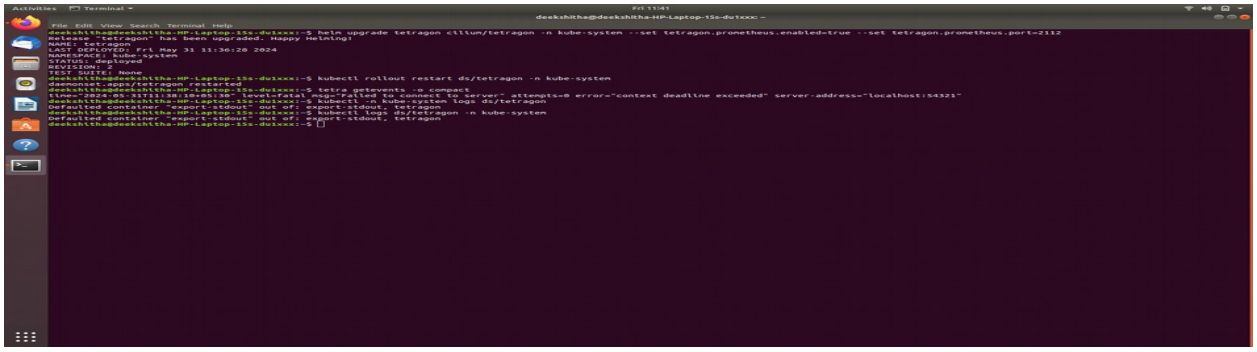
5.1. Overview of Grafana

Grafana is an open-source platform for monitoring and observability. It enables the visualization of metrics, logs, and traces from various data sources, providing a comprehensive view of system health and performance.

5.2. Integrating Grafana with Tetragon for metrics

To integrate Grafana with Tetragon, follow these steps:

Enable prometheus metrics as follows:



Typically, metrics are scraped by Prometheus or another compatible agent (for example OpenTelemetry Collector), stored in Prometheus or another compatible database, then queried and visualized for example using Grafana.

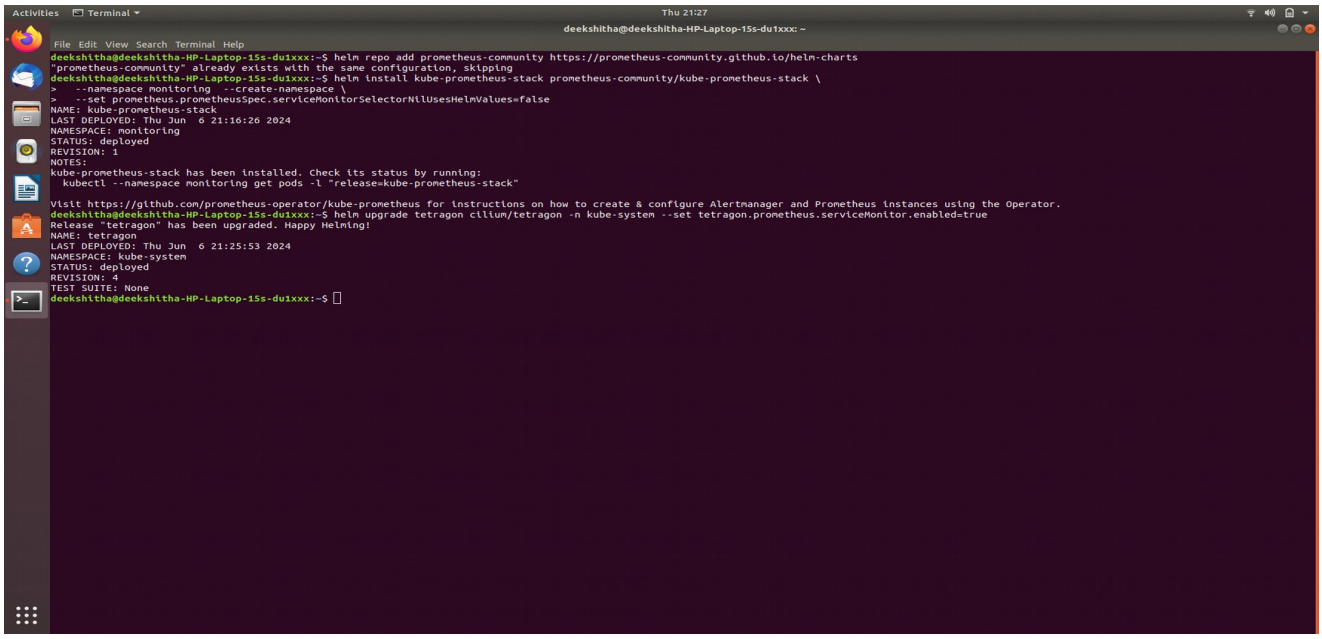
In Kubernetes, you can install Prometheus and Grafana using the kube-prometheus-stack Helm chart:

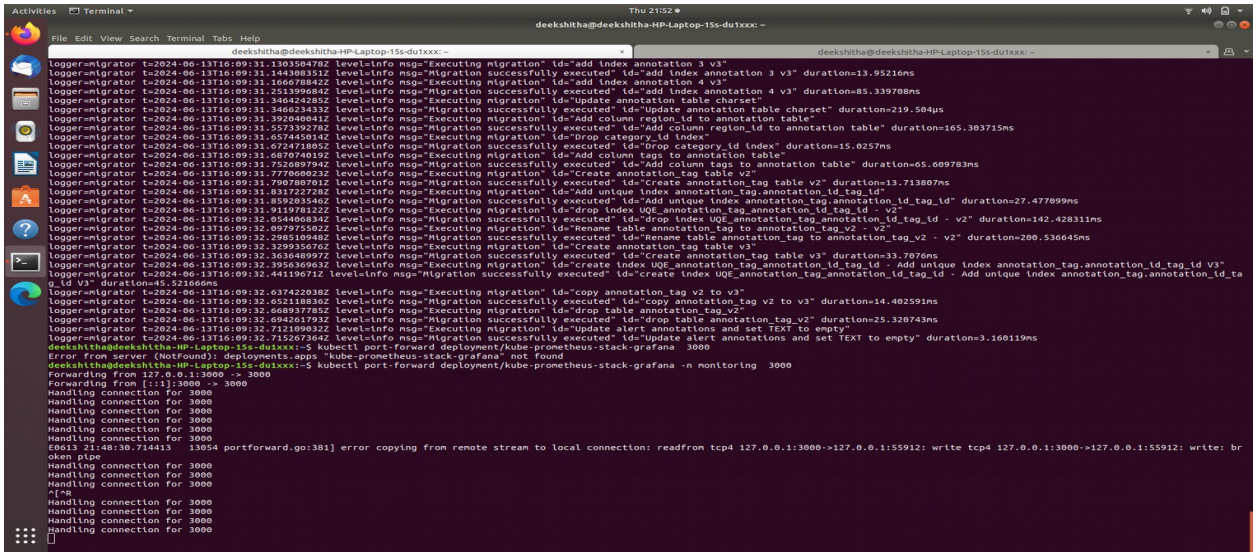
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

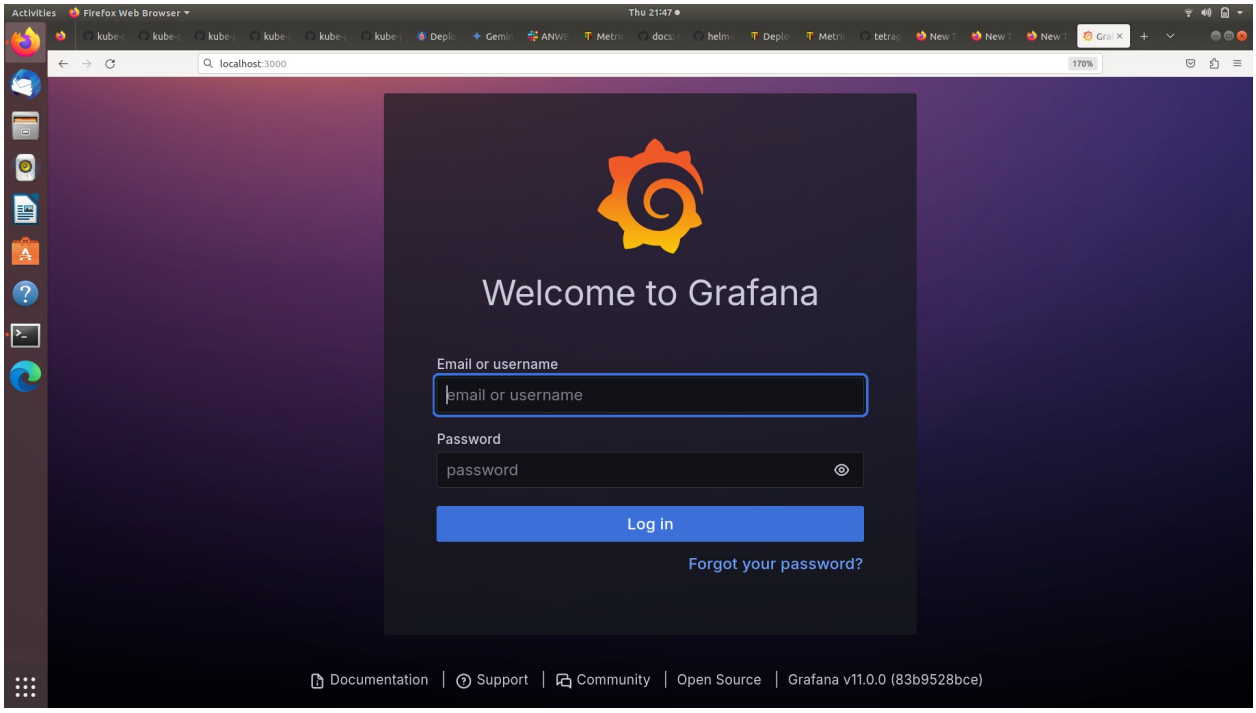
helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack \

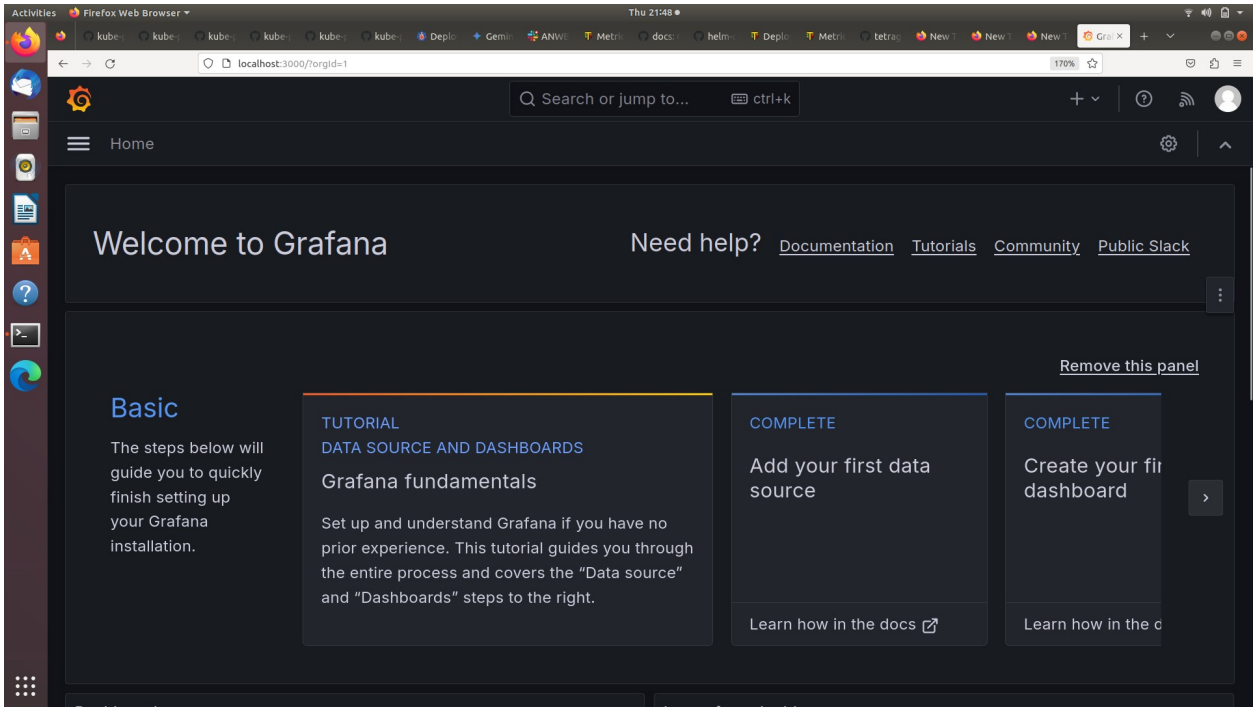
  --namespace monitoring --create-namespace \

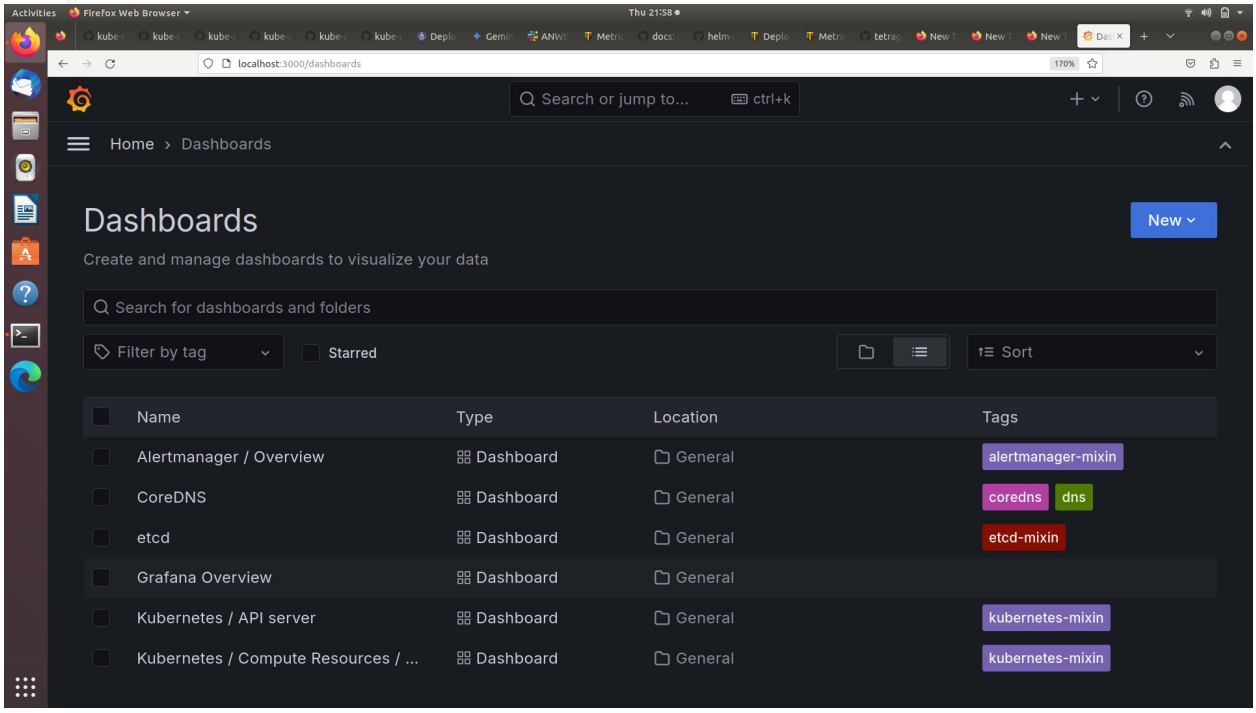
  --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false


Activities
Terminal
Thu 21:27
deekshitha@deekshitha-HP-Laptop-15s-du1xxx: ~
File Edit View Search Terminal Help
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" already exists with the same configuration, skipping
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack \
> --namespace monitoring --create-namespace \
> --set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false
NAME: kube-prometheus-stack
LAST DEPLOYED: Thu Jun 6 21:16:26 2024
NAMESPACE: monitoring
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
kubectl --namespace monitoring get pods -l "release=kube-prometheus-stack"
Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ helm upgrade tetragon cilium/tetragon -n kube-system --set tetragon.prometheus.serviceMonitor.enabled=true
Release "tetragon" has been upgraded. Happy Helming!
NAME: tetragon
LAST DEPLOYED: Thu Jun 6 21:25:53 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 4
TEST SUITE: None
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$

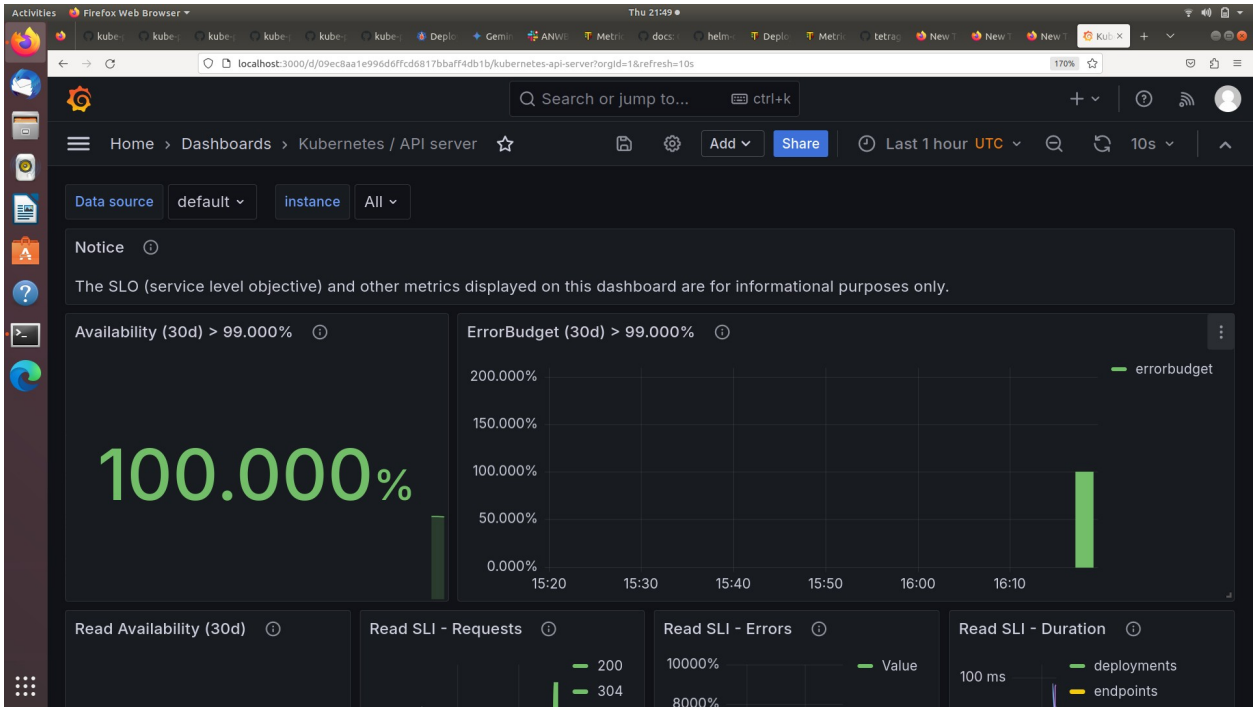

Activities
Terminal
Thu 21:52
deekshitha@deekshitha-HP-Laptop-15s-du1xxx: ~
File Edit View Search Terminal Tabs Help
deekshitha@deekshitha-HP-Laptop-15s-du1xxx: ~
deekshitha@deekshitha-HP-Laptop-15s-du1xxx: ~
logger=migrator t=2024-06-13T16:09:31.130350478Z level=info msg="Executing migration" id="add index annotation 3 v3"
logger=migrator t=2024-06-13T16:09:31.144308351Z level=info msg="Migration successfully executed" id="add index annotation 3 v3" duration=13.95216ms
logger=migrator t=2024-06-13T16:09:31.166678842Z level=info msg="Executing migration" id="add index annotation 4 v3"
logger=migrator t=2024-06-13T16:09:31.251399684Z level=info msg="Migration successfully executed" id="add index annotation 4 v3" duration=85.339708ms
logger=migrator t=2024-06-13T16:09:31.346424285Z level=info msg="Executing migration" id="Update annotation table charset"
logger=migrator t=2024-06-13T16:09:31.346623433Z level=info msg="Migration successfully executed" id="Update annotation table charset" duration=219.504µs
logger=migrator t=2024-06-13T16:09:31.392048041Z level=info msg="Executing migration" id="Add column region_id to annotation table"
logger=migrator t=2024-06-13T16:09:31.557339278Z level=info msg="Migration successfully executed" id="Add column region_id to annotation table" duration=165.303715ms
logger=migrator t=2024-06-13T16:09:31.657445014Z level=info msg="Executing migration" id="Drop category_id index"
logger=migrator t=2024-06-13T16:09:31.672471805Z level=info msg="Migration successfully executed" id="Drop category_id index" duration=15.0257ms
logger=migrator t=2024-06-13T16:09:31.687074019Z level=info msg="Executing migration" id="Add column tags to annotation table"
logger=migrator t=2024-06-13T16:09:31.752689794Z level=info msg="Migration successfully executed" id="Add column tags to annotation table" duration=65.609783ms
logger=migrator t=2024-06-13T16:09:31.777060823Z level=info msg="Executing migration" id="Create annotation_tag table v2"
logger=migrator t=2024-06-13T16:09:31.790780701Z level=info msg="Migration successfully executed" id="Create annotation_tag table v2" duration=13.713807ms
logger=migrator t=2024-06-13T16:09:31.831722728Z level=info msg="Executing migration" id="Add unique index annotation_tag.annotation_id_tag_id"
logger=migrator t=2024-06-13T16:09:31.859203546Z level=info msg="Migration successfully executed" id="Add unique index annotation_tag.annotation_id_tag_id" duration=27.477099ms
logger=migrator t=2024-06-13T16:09:31.911978122Z level=info msg="Executing migration" id="drop index UQE_annotation_tag_annotation_id_tag_id - v2"
logger=migrator t=2024-06-13T16:09:32.054406834Z level=info msg="Migration successfully executed" id="drop index UQE_annotation_tag_annotation_id_tag_id - v2" duration=142.428311ms
logger=migrator t=2024-06-13T16:09:32.097975502Z level=info msg="Executing migration" id="Rename table annotation_tag to annotation_tag_v2 - v2"
logger=migrator t=2024-06-13T16:09:32.298510948Z level=info msg="Migration successfully executed" id="Rename table annotation_tag to annotation_tag_v2 - v2" duration=200.536645ms
logger=migrator t=2024-06-13T16:09:32.329935676Z level=info msg="Executing migration" id="Create annotation_tag table v3"
logger=migrator t=2024-06-13T16:09:32.363648997Z level=info msg="Migration successfully executed" id="Create annotation_tag table v3" duration=33.7076ms
logger=migrator t=2024-06-13T16:09:32.395636963Z level=info msg="Executing migration" id="create index UQE_annotation_tag_annotation_id_tag_id - Add unique index annotation_tag.annotation_id_tag_id V3"
logger=migrator t=2024-06-13T16:09:32.44119671Z level=info msg="Migration successfully executed" id="create index UQE_annotation_tag_annotation_id_tag_id - Add unique index annotation_tag.annotation_id_tag_id V3" duration=45.521666ms
logger=migrator t=2024-06-13T16:09:32.637422038Z level=info msg="Executing migration" id="copy annotation_tag v2 to v3"
logger=migrator t=2024-06-13T16:09:32.652118836Z level=info msg="Migration successfully executed" id="copy annotation_tag v2 to v3" duration=14.402591ms
logger=migrator t=2024-06-13T16:09:32.668937785Z level=info msg="Executing migration" id="drop table annotation_tag_v2"
logger=migrator t=2024-06-13T16:09:32.694261793Z level=info msg="Migration successfully executed" id="drop table annotation_tag_v2" duration=25.320743ms
logger=migrator t=2024-06-13T16:09:32.712109032Z level=info msg="Executing migration" id="Update alert annotations and set TEXT to empty"
logger=migrator t=2024-06-13T16:09:32.715267364Z level=info msg="Migration successfully executed" id="Update alert annotations and set TEXT to empty" duration=3.160119ms
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ kubectl port-forward deployment/kube-prometheus-stack-grafana 3000
Error from server (NotFound): deployments.apps "kube-prometheus-stack-grafana" not found
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ kubectl port-forward deployment/kube-prometheus-stack-grafana -n monitoring 3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
E0613 21:48:30.714413 13054 portforward.go:381] error copying from remote stream to local connection: readfrom tcp4 127.0.0.1:3000->127.0.0.1:55912: write tcp4 127.0.0.1:3000->127.0.0.1:55912: write: broken pipe
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
^[^8
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000
Handling connection for 3000

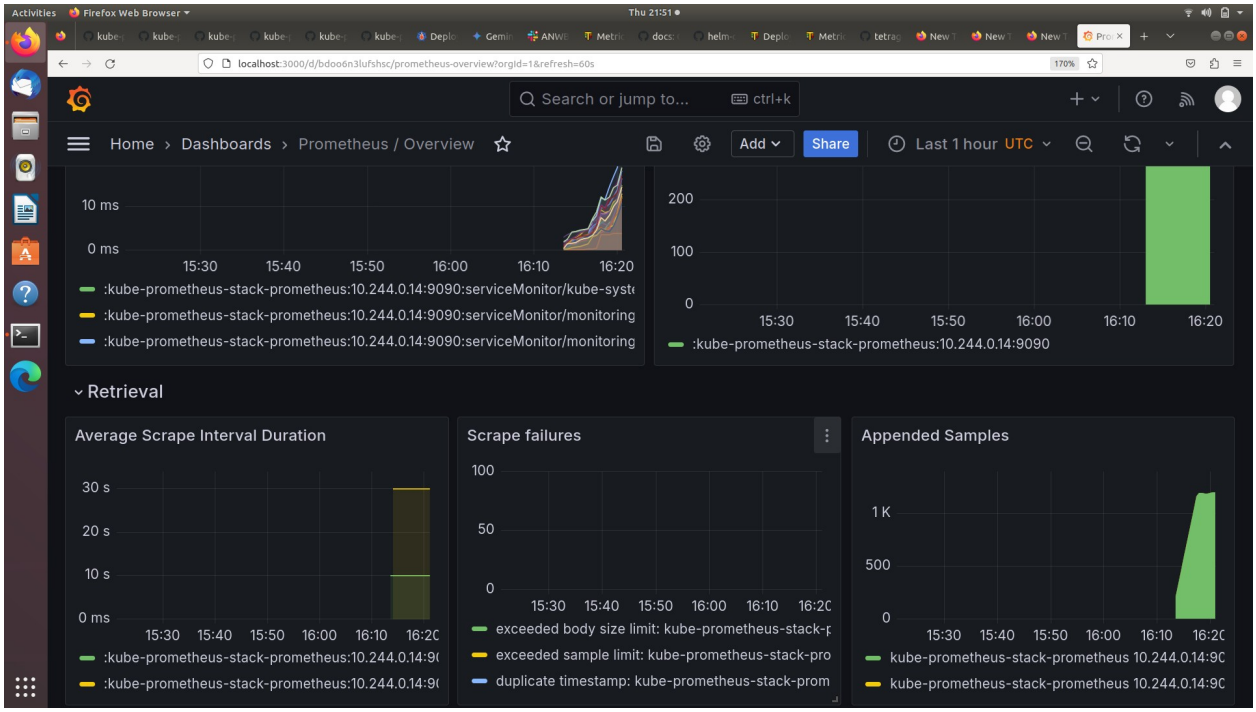

Activities
Firefox Web Browser
Thu 21:47
localhost:3000
Welcome to Grafana
Email or username
email or username
Password
password
Log in
Forgot your password?
Documentation
Support
Community
Open Source
Grafana v11.0.0 (83b9528bce)

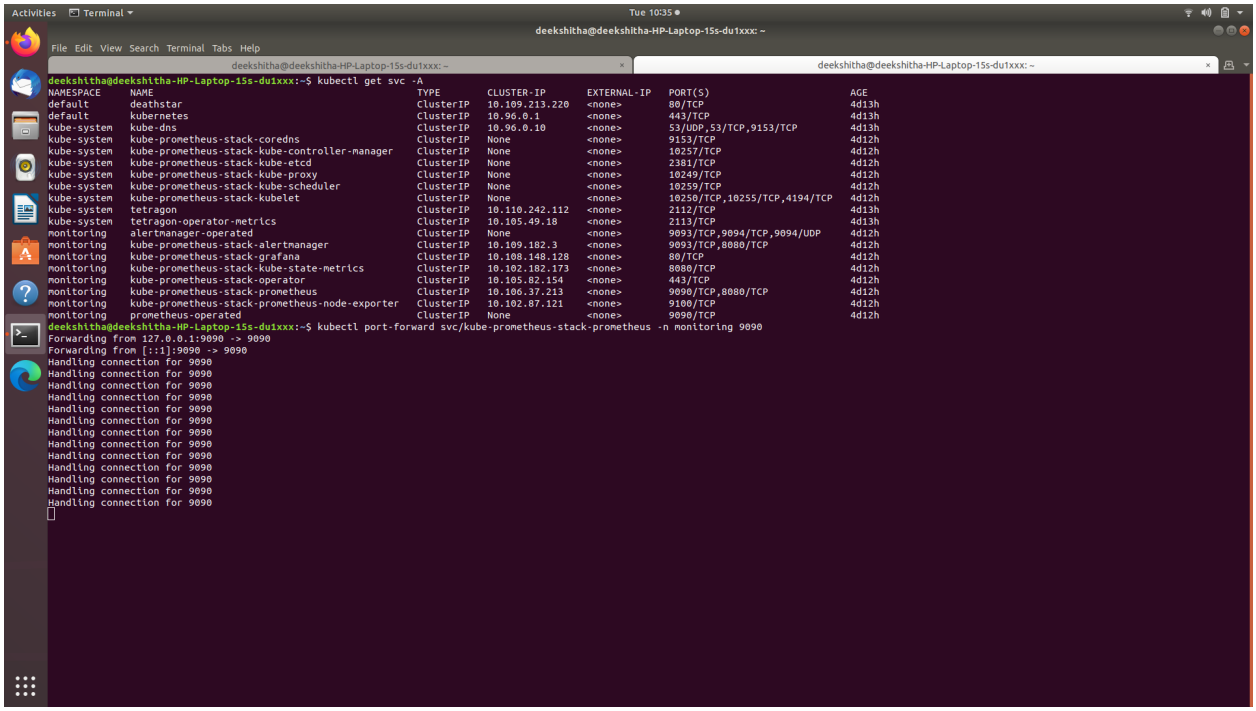

Activities
Firefox Web Browser
Thu 21:48
localhost:3000/?orgId=1
Search or jump to...
ctrl+k
Home
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your fi dashboard
Learn how in the d



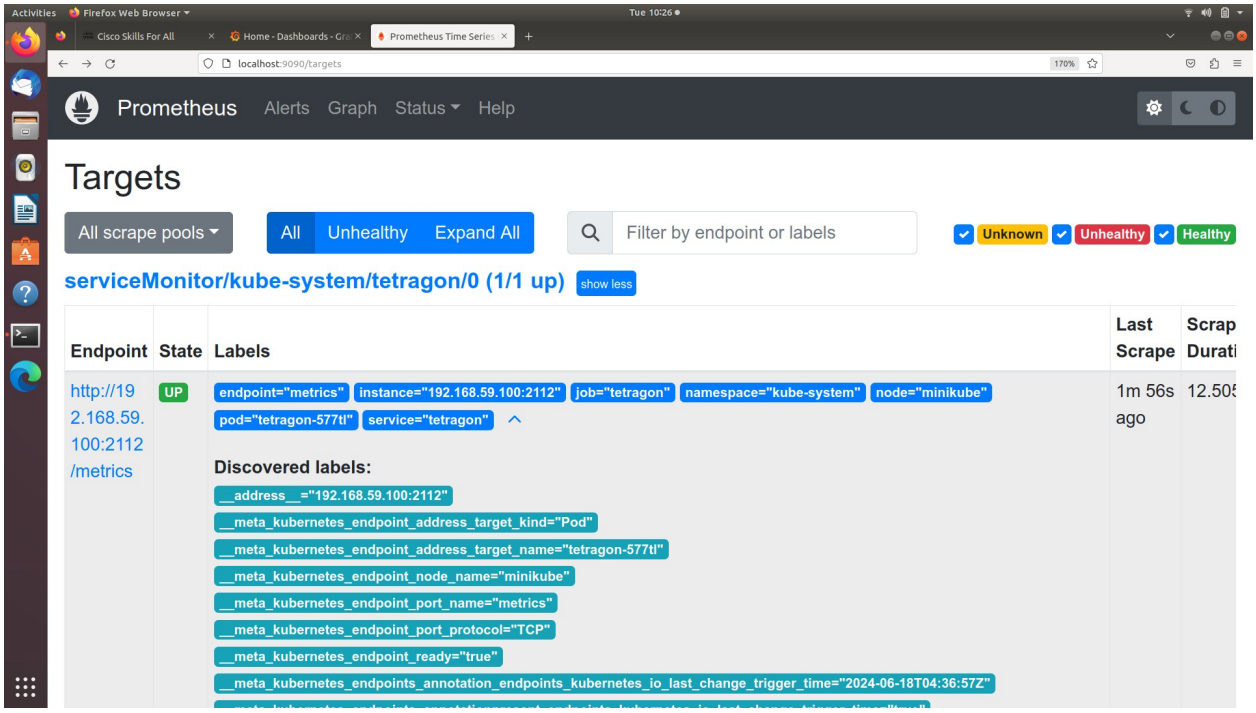
Example: Grafana Dashboard







Prometheus User Interface: check whether tetragon metrics are healthy or not


Activities
Firefox Web Browser
Tue 10:26
Cisco Skills For All
Home - Dashboards - Gra
Prometheus Time Series
localhost:9090/targets
170%
Prometheus
Alerts
Graph
Status
Help
Targets
All scrape pools
All
Unhealthy
Expand All
Filter by endpoint or labels
Unknown
Unhealthy
Healthy
serviceMonitor/kube-system/tetragon/0 (1/1 up)
show less
Endpoint
State
Labels
Last Scrape
Scrap Durati
http://192.168.59.100:2112/metrics
UP
endpoint="metrics"
instance="192.168.59.100:2112"
job="tetragon"
namespace="kube-system"
node="minikube"
pod="tetragon-577tl"
service="tetragon"
1m 56s ago
12.505
Discovered labels:
__address__="192.168.59.100:2112"
__meta_kubernetes_endpoint_address_target_kind="Pod"
__meta_kubernetes_endpoint_address_target_name="tetragon-577tl"
__meta_kubernetes_endpoint_node_name="minikube"
__meta_kubernetes_endpoint_port_name="metrics"
__meta_kubernetes_endpoint_port_protocol="TCP"
__meta_kubernetes_endpoint_ready="true"
__meta_kubernetes_endpoints_annotation_endpoints_kubernetes_io_last_change_trigger_time="2024-06-18T04:36:57Z"