

Contents

1. Security Trends
 - o Authentication
 - o Authorization
 - o Encryption
 - o Cybersecurity Current Trends
 - o Access Control and Accounting
2. Kubernetes
 - o Introduction to Kubernetes
 - o Creation of Minikube Cluster
 - o Performing Basic Commands
3. eBPF Technologies
 - o Introduction to eBPF
 - o Falcon
 - o Tracee
 - o Tetragon
 - o StackRox
 - o Advantages of Tetragon
 - o Why We Use Tetragon
4. Execution of Policies Using Tetragon
 - o Deploy on a Cluster
 - o Defining Security Policies
 - o Example Policy: Sudo Invocation Monitoring
 - o Policy Management
5. Integration of Grafana with Tetragon
 - o Overview of Grafana
 - o Integrating Grafana with Tetragon

o 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
```

```
Microsoft Windows [Version 10.0.22631.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sri Ragam\minikube
minikube provisions and manages local Kubernetes clusters optimized for development workflows.

Basic Commands:
  start      Starts a local Kubernetes cluster
  status     Gets the status of a local Kubernetes cluster
  stop       Stops a running local Kubernetes cluster
  delete    Deletes a local Kubernetes cluster
  dashboard  Access the Kubernetes dashboard running within the minikube cluster
  pause      pause Kubernetes
  unpause   unpause Kubernetes

Images Commands:
  docker-env  Provides instructions to point your terminal's docker-cli to the Docker Engine inside minikube.
  (Useful for building docker images directly inside minikube)
  podman-env  Configure environment to use minikube's Podman service
  cache       Manage cache for images
  image       Manage images

Configuration and Management Commands:
  addons      Enable or disable a minikube addon
  config      Modify persistent configuration values
  profile     Get or list the current profiles (clusters)
  update-context  Update kubeconfig in case of an IP or port change

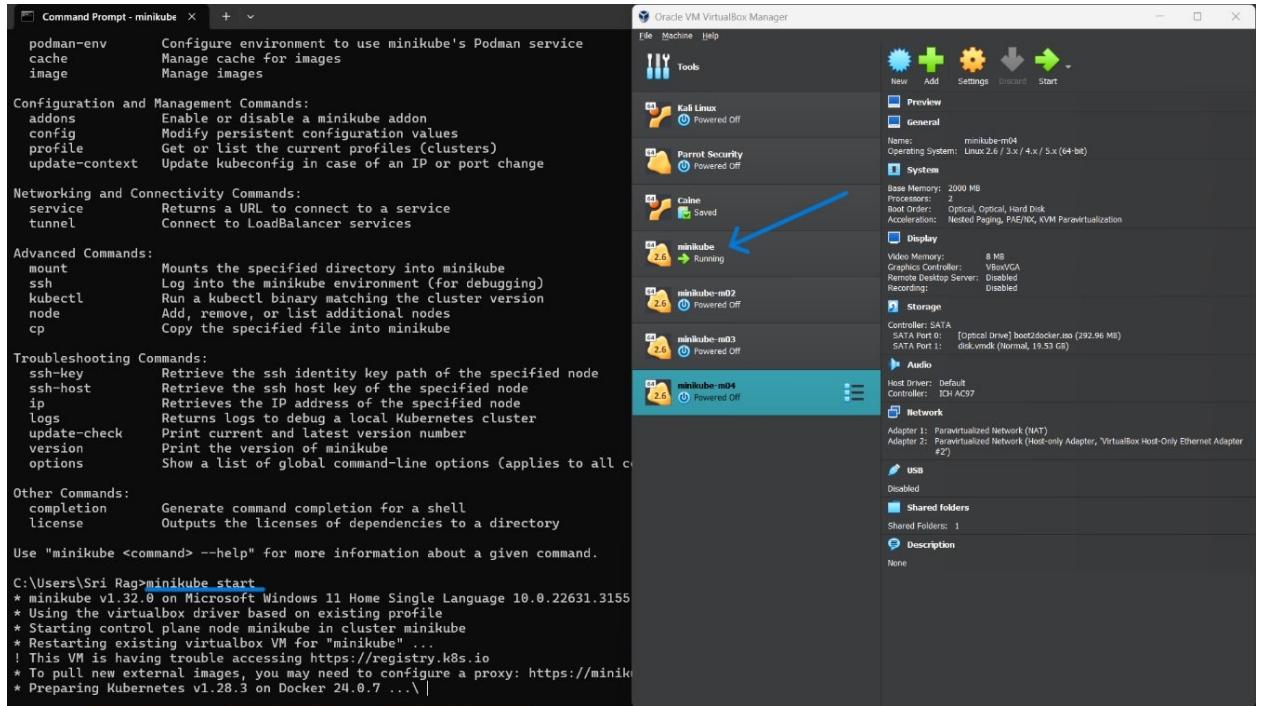
Networking and Connectivity Commands:
  service     Returns a URL to connect to a service
  tunnel      Connect to LoadBalancer services

Advanced Commands:
  mount       Mounts the specified directory into minikube
  ssh        Log into the minikube environment (for debugging)
  kubectl    Run a kubectl binary matching the cluster version
  node       Add, remove, or list additional nodes
  cp         Copy the specified file into minikube

Troubleshooting Commands:
  ssh-key    Retrieve the ssh identity key path of the specified node
  ssh-host   Retrieve the ssh host key of the specified node
  ip        Retrieves the IP address of the specified node
  logs      Returns logs to debug a local Kubernetes cluster
```

2. Start Minikube:

```
minikube start
```



3. Verify Minikube Installation:

kubectl get nodes

```
C:\>kubectl>minikube start
* minikube v1.32.0 on Microsoft Windows 11 Home Single Language 10.0.22631.3155
* Using the virtualbox driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Restarting existing virtualbox VM for "minikube" ...
! This VM is having trouble accessing https://registry.k8s.io
* To pull new external images, you may need to configure a proxy: https://minikube.svc.cluster.local:8443
* Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...\\|
```

```
C:\>kubectl>cd minikube
C:\>kubectl>minikube status
minikube
type: Control Plane
host: Running
kublet: Running
apiserver: Running
kubeconfig: Configured

C:\>kubectl>minikube ip
192.168.59.101

C:\>kubectl>minikube get nodes
Error: unknown command "get" for "minikube"
Run 'minikube --help' for usage.

C:\>kubectl>kubectl get nodes
NAME      STATUS    ROLES     AGE   VERSION
minikube  Ready     control-plane  8h    v1.28.3

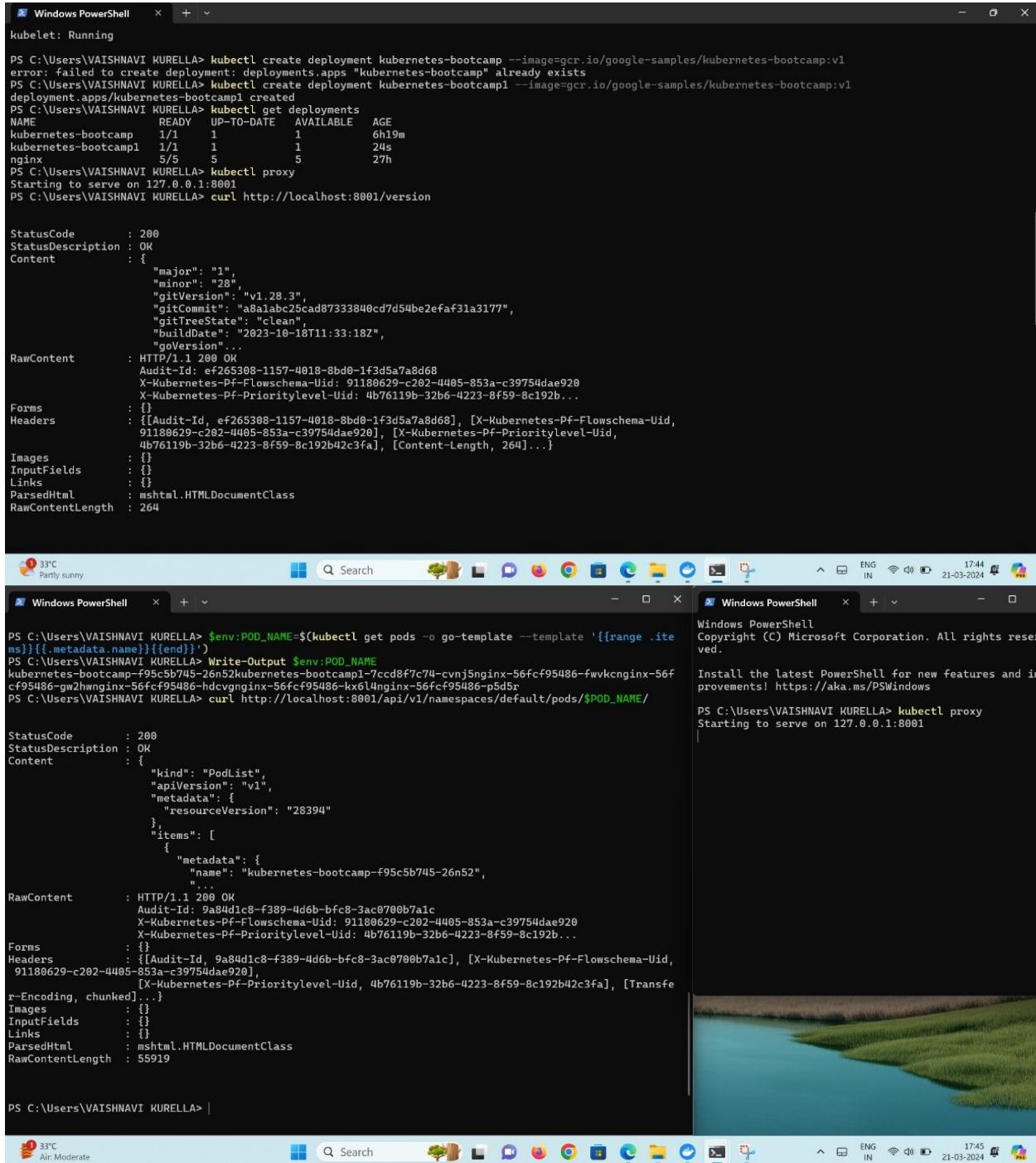
C:\>kubectl>kubectl cluster-info
Kubernetes control plane is running at https://192.168.59.101:8443
CoreDNS is running at https://192.168.59.101:8443/api/v1/namespaces/kube-system/services/kube-dns:80/proxy
```

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 screenshot shows a Windows desktop environment with two open PowerShell windows and a web browser window.

Top PowerShell Window (Windows PowerShell):

```
kubelet: Running
PS C:\Users\VAISHNAVI KURELLA> kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1
error: failed to create deployment: deployments.apps "kubernetes-bootcamp" already exists
PS C:\Users\VAISHNAVI KURELLA> kubectl create deployment kubernetes-bootcamp1 --image=gcr.io/google-samples/kubernetes-bootcamp:v1
deployment.apps/kubernetes-bootcamp1 created
PS C:\Users\VAISHNAVI KURELLA> kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           6h19m
kubernetes-bootcamp1  1/1     1            1           24s
nginx           5/5     5            5           27h
PS C:\Users\VAISHNAVI KURELLA> kubectl proxy
Starting to serve on 127.0.0.1:8001
PS C:\Users\VAISHNAVI KURELLA> curl http://localhost:8001/version

StatusCode : 200
StatusDescription : OK
Content : {
    "major": "1",
    "minor": "28",
    "gitVersion": "v1.28.3",
    "gitCommit": "a8a1abc25cad87333840cd7d54be2efaf31a3177",
    "gitTreeState": "clean",
    "buildDate": "2023-10-18T11:33:18Z",
    "goVersion": ...
}
RawContent : HTTP/1.1 200 OK
Audit-Id: ef265308-1157-4018-8bd0-1f3d5a7a8d68
X-Kubernetes-PF-Flowschema-Uid: 91180629-c202-4405-853a-c39754dae920
X-Kubernetes-PF-PriorityLevel-Uid: 4b76119b-32b6-4223-8f59-8c192b...
Forms : {}
Headers : {[{"Audit-Id": "ef265308-1157-4018-8bd0-1f3d5a7a8d68"}, {"X-Kubernetes-PF-Flowschema-Uid": "91180629-c202-4405-853a-c39754dae920"}, {"X-Kubernetes-PF-PriorityLevel-Uid": "4b76119b-32b6-4223-8f59-8c192b42c3fa"}, {"Content-Length": "264"}...]
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 264
```

Bottom PowerShell Window (Windows PowerShell):

```
PS C:\Users\VAISHNAVI KURELLA> $env:POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{end}}')
PS C:\Users\VAISHNAVI KURELLA> Write-Output $env:POD_NAME
kubernetes-bootcamp-f95c5b745-26n52kubernetes-bootcamp1-7cd8f7c74-cvnj5nginx-56fcf95486-fwvkcninx-56fc95486-gw2hwnginx-56fcf95486-hdcvgnginx-56fcf95486-kx6l4nginx-56fcf95486-p5d5r
PS C:\Users\VAISHNAVI KURELLA> curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME

StatusCode : 200
StatusDescription : OK
Content : {
    "kind": "PodList",
    "apiVersion": "v1",
    "metadata": {
        "resourceVersion": "28394"
    },
    "items": [
        {
            "metadata": {
                "name": "kubernetes-bootcamp-f95c5b745-26n52",
                ...
}
RawContent : HTTP/1.1 200 OK
Audit-Id: 9a84d1c8-f389-4d6b-bfc8-3ac0700b7a1c
X-Kubernetes-PF-Flowschema-Uid: 91180629-c202-4405-853a-c39754dae920
X-Kubernetes-PF-PriorityLevel-Uid: 4b76119b-32b6-4223-8f59-8c192b...
Forms : {}
Headers : {[{"Audit-Id": "9a84d1c8-f389-4d6b-bfc8-3ac0700b7a1c"}, {"X-Kubernetes-PF-Flowschema-Uid": "91180629-c202-4405-853a-c39754dae920"}, {"X-Kubernetes-PF-PriorityLevel-Uid": "4b76119b-32b6-4223-8f59-8c192b42c3fa"}, {"Transfer-Encoding": "chunked"}...]
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 55919
```

Browser Window:

The browser window shows a simple landing page with the text "Welcome to Kubernetes".

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> kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-f95c5b745-c7bg2  1/1     Running   0          104m
nginx-56fcf95486-5965b      0/1     Terminating   0          131m
nginx-56fcf95486-gjzb4      0/1     Terminating   0          131m
PS C:\Users\Deekshitha> kubectl describe pods
Name:           kubernetes-bootcamp-f95c5b745-c7bg2
Namespace:      default
Priority:       0
Service Account: default
Node:          test-m03/192.168.59.114
Start Time:    Wed, 20 Mar 2024 19:55:14 +0530
Labels:         app:kubernetes-bootcamp
                pod-template-hash=f95c5b745
Annotations:   <none>
Status:        Running
IP:            10.244.2.8
IPs:
  IP:          10.244.2.8
Controlled By: ReplicaSet/kubernetes-bootcamp-f95c5b745
Containers:
  kubernetes-bootcamp:
    Container ID:  docker://9afdf5e2ea887c5bd3e7078b498efdalal45a41ac5f78d027acd985bfb4b12e1
    Image:         gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:     docker-pullable://gcr.io/google-samples/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279c8fcc64af
    Port:          <none>
    Host Port:    <none>
    State:        Running
      Started:   Wed, 20 Mar 2024 20:20:50 +0530
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-x956d (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
```

```
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
```

```
error: pod, type/name or --filename must be specified
```



```
PS C:\Users\Deekshitha> kubectl proxy
Starting to serve on 127.0.0.1:8001
```

```

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 $env:POD_NAME
kubernetes-bootcamp-f95c5b745-c7bg2nginx-56fcf95486-5965b
PS C:\Users\Deekshitha> curl http://localhost:8080/api/v1/namespaces/default/pods/$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-d919a364e85d
X-Kubernetes-PF-Flowschema-Uid: f5cfebcf-3ee5-4979-a285-cd050a145e49
X-Kubernetes-PF-PriorityLevel-Uid: f9572119-ee13-48aa-a5b2-0e95cd68c599
Forms          : {}
Headers        : {[Audit-Id, 6d5f9778-e625-405a-8471-d919a364e85d], [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

```
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-54597f9d79-45hr1   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-qdnsn  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
```

```
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@sha256:6db391d1c0cfb30588ba0bf72ea999404f2764febfof1f196acd5867ac7efa7e
    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
  PodsScheduled  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@sha256:6db391d1c0cfb30588ba0bf72ea999404f2764febfof1f196acd5867ac7efa7e
    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
  PodsScheduled  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
```

```
deployment "Example" successfully rolled out
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

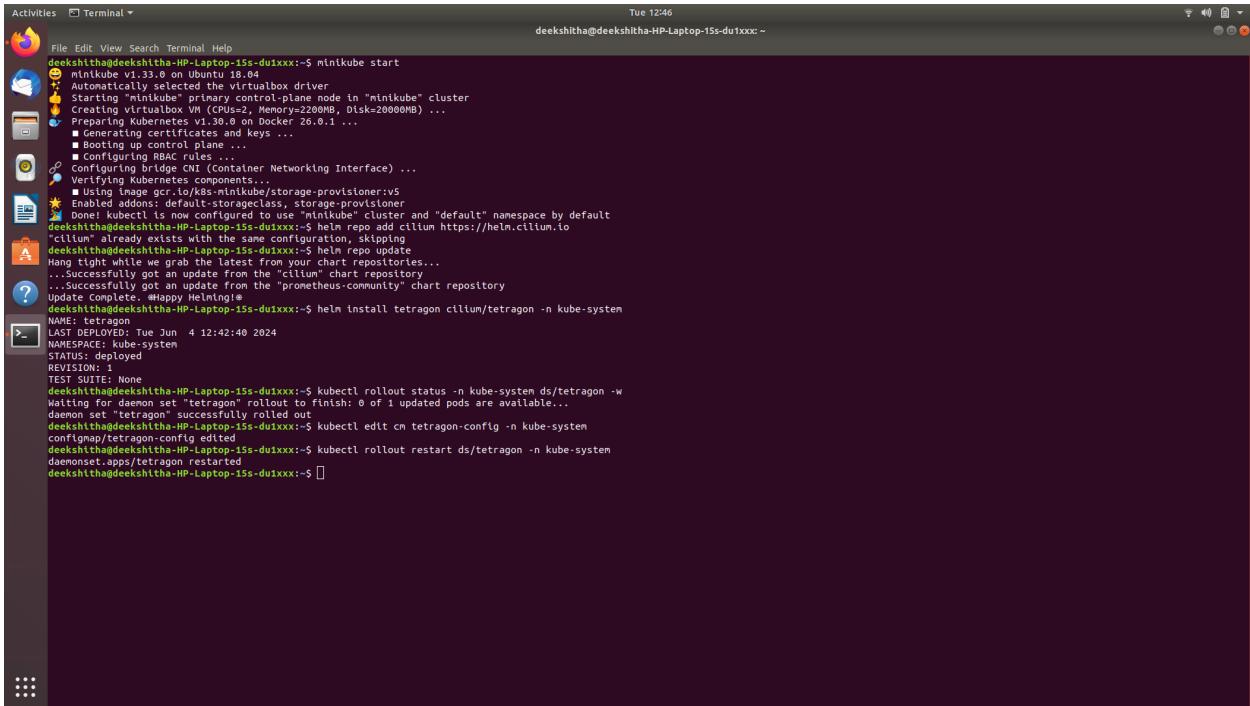
```
helm repo add cilium https://helm.cilium.io
```

```
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
```



```
Tue 12:46
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~
```

```
minikube v1.33.0 on Ubuntu 18.04
* Automatically selected the virtualbox driver
* Starting local proxy and control-plane node in "minikube" cluster
* Creating VirtualBox VM (Cpu=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.0.1 ...
  □ Generating certificates and keys ...
  □ Booting up control plane ...
  □ Configuring RBAC rules
  □ Configuring bridge CNI (Container Networking Interface) ...
  □ Configuring kubelet components...
  □ Using image gcr.io/k8s-minikube/storage-provisioner:v5
Enabled addons: default-storageclass, storage-provisioner
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ helm repo add cilium https://helm.cilium.io
cilium already exists with the same configuration, skipping
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ helm fetch --untar prometheus-community
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "cilium" chart repository
...Successfully got an update from the "prometheus-community" chart repository
Update Complete. @Happy HelmNg!
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ helm install tetragon cilium/tetragon -n kube-system
NAME: tetragon
LAST DEPLOYED: Tue Jun 4 12:42:40 2024
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ kubectl rollout status -n kube-system ds/tetragon -w
Waiting for daemon set "tetragon" rollout to finish: 0 of 1 updated pods are available...
daemon set "tetragon" successfully rolled out
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ kubectl edit cm tetragon-config -n kube-system
configmap/tetragon-config edited
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$ kubectl rollout restart ds/tetragon -n kube-system
daemonset.apps/tetragon restarted
deekshitha@deekshitha-HP-Laptop-15s-duxxxx:~$
```

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.

```

deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ kubectl logs -n kube-system -l app.kubernetes.io/name=tetragon
<-- export>| jq '.select(.process_exec != null) | select(.process_exec.process.binary | contains("sudo")) | "(.time) \".process_exec.process.pod.namespace\" \".process_exec.process.binary\" \".process_exec.process.arguments\""
2024-05-28T07:41:23.707552357Z default /var/lib/dpkg/info/sudo.postinst /var/lib/dpkg/info/sudo.postinst co
nfigng
"2024-05-28T07:41:28.982303447Z default /usr/bin/sudo -t"

```

```

deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~$ kubectl exec -it xw1ng -- bash
root@xw1ng:/# apt-get update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8786 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [13.8 kB]

Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [157 kB]

Fetched 9211 kB in 23s (395 kB/s)

Reading package lists... Done
root@xw1ng:/# apt-get install sudo
Reading package lists... Done
Building dependency tree... Done
Reading status information... Done
The following NEW packages will be installed:
  sudo
0 upgraded, 1 newly installed, 0 to remove and 17 not upgraded.
Need to get 1889 kB of archives.
After this operation, 6199 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 sudo amd64 1.9.13p3-1+deb12u1 [1889 kB]
Fetched 1889 kB in 2s (842 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package sudo.
(Reading database... 7026 files and directories currently installed.)
Preparing to unpack .../0-sudo_1.9.13p3-1+deb12u1_amd64.deb ...
Unpacking sudo (1.9.13p3-1+deb12u1) ...
Setting up sudo (1.9.13p3-1+deb12u1) ...
invoke-rc.d: could not determine current runlevel
Processing triggers for libc-bin (2.36-9+deb12u3) ...
Processing triggers for policy-rc.d (0.9.9-1+deb12u1) ...
root@xw1ng:/# sudo -t
root@xw1ng:/# echo "hi"
hi
root@xw1ng:/# exit
logout
root@xw1ng:/# exit
exit

```

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

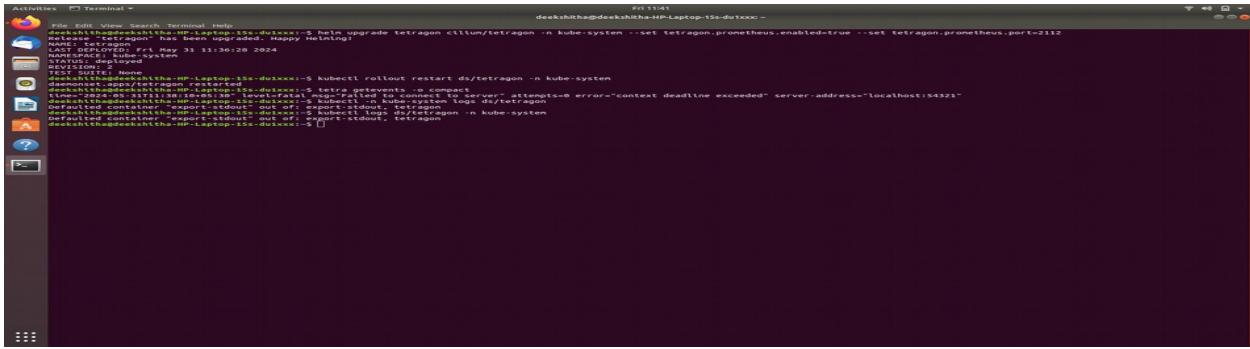
5.1. Overview of Grafana

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:



```
Activities Terminal - deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ helm upgrade tetragon citium/tetragon -n kube-system --set tetragon.prometheus.enabled=true --set tetragon.prometheus.port=2112
NAME: tetragon
LAST DEPLOYED: Fri May 31 11:36:19 2024
NAMESPACE: kube-system
STATUS: deployed
TEST SUITE: None
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ kubectl rollout restart ds/tetragon -n kube-system
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ tetra getevents -n compact
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ kubectl get events --sort-by='lastseen' --export=yaml > /tmp/tetra-export.yaml
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ kubectl -n kube-system logs ds/tetragon
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ kubectl logs ds/tetragon -n kube-system
deekshitha@deekshitha-HP-Laptop-15s-dx0xx: ~$ [
```

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
```

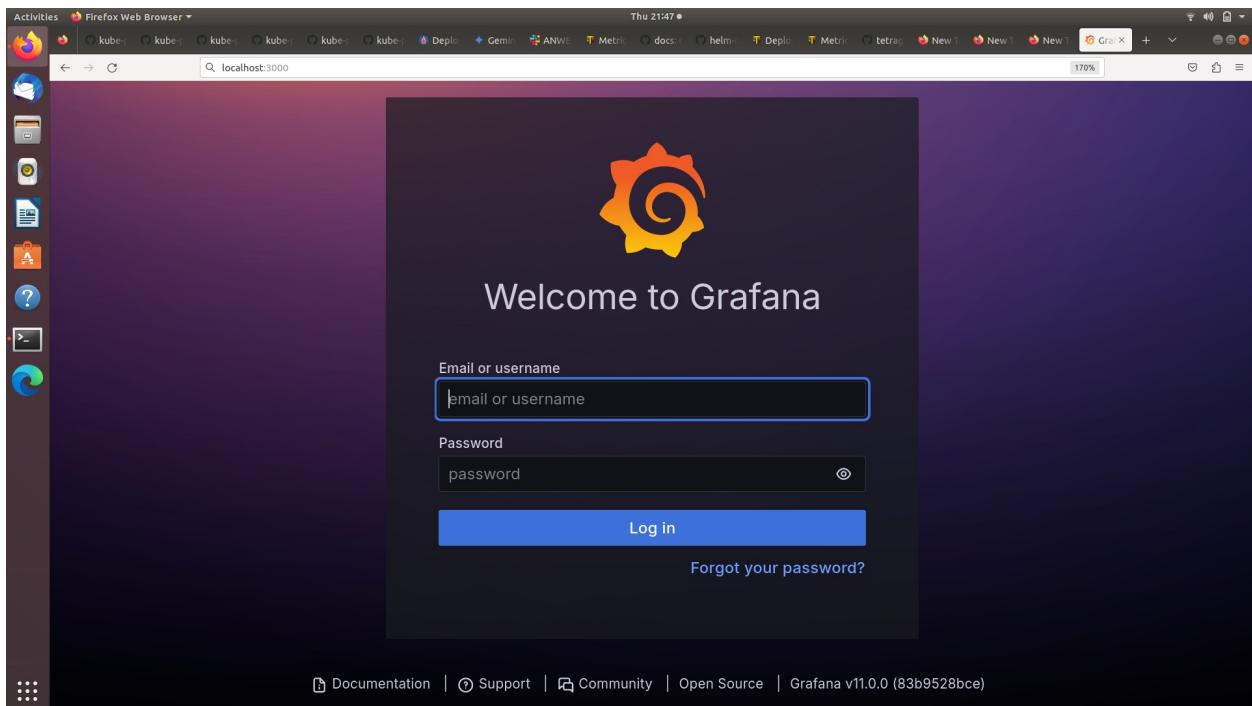
```
helm install kube-prometheus-stack prometheus-community/kube-prometheus-stack \
```

```
--namespace monitoring --create-namespace \
```

```
--set prometheus.prometheusSpec.serviceMonitorSelectorNilUsesHelmValues=false
```

```
Activities Terminal Thu 21:27
deekshitha@deekshitha-HP-Laptop-15s-du1xxx:~ 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 kube-system --create-namespace \
> --set 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 cillum/tetragon -n kube-system --set tetragon.prometheus.serviceMonitor.enabled=true
Release "tetragon" has been upgraded. Happy Helm-ing!
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:~
```

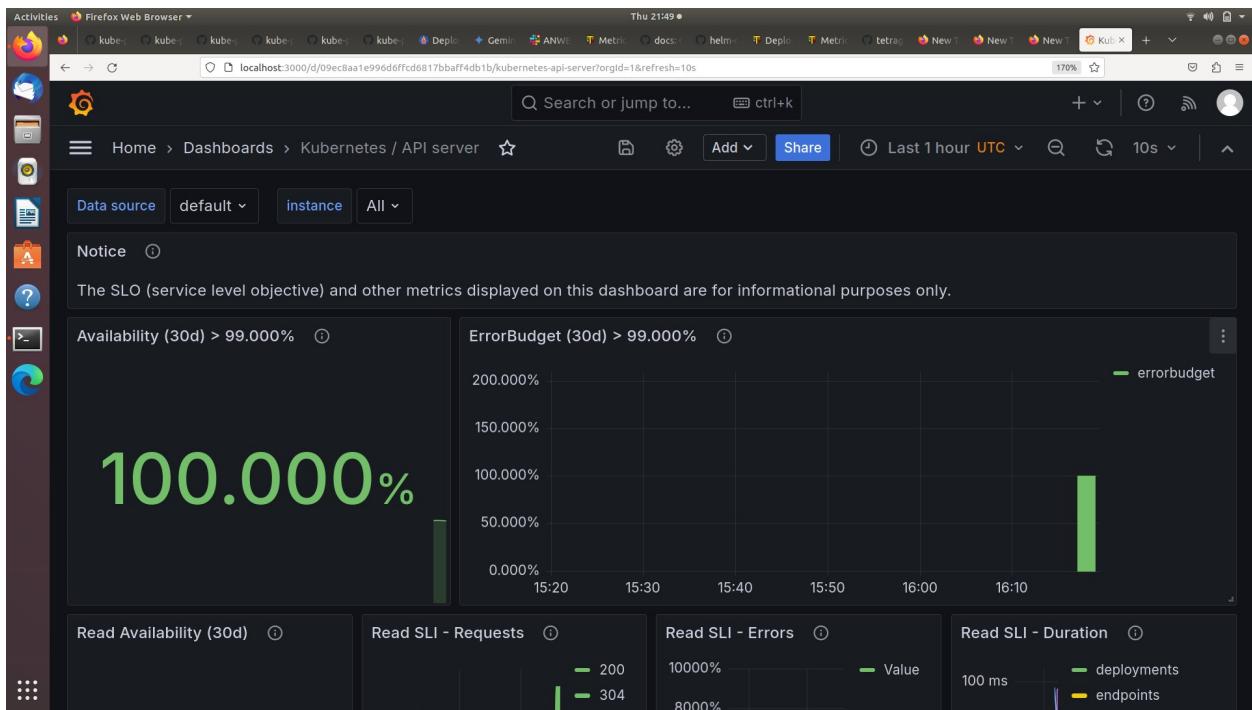


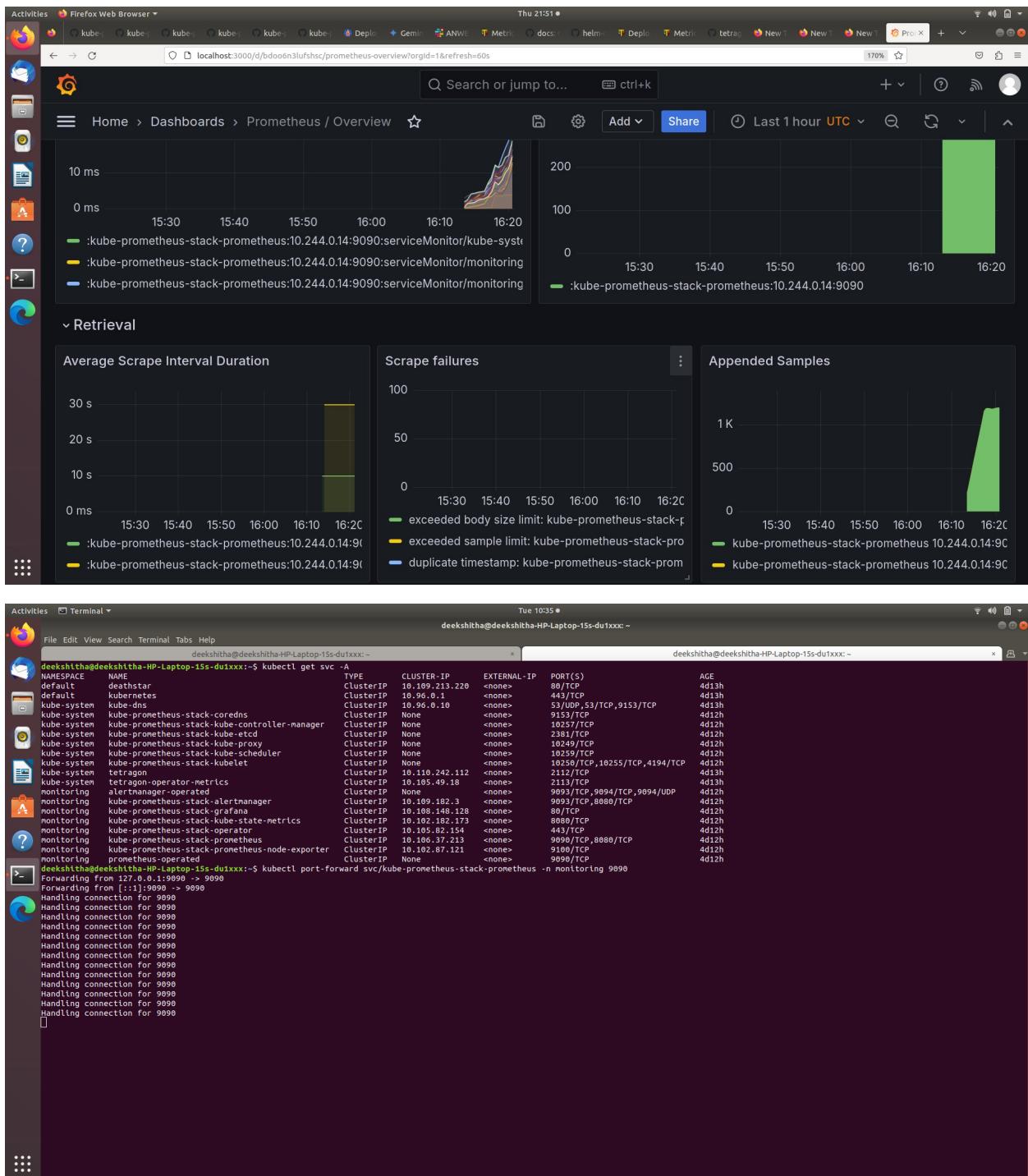
A screenshot of the Grafana dashboard setup screen in a Firefox browser window. The desktop environment is the same as the previous screenshot. The browser window title is "Activities Firefox Web Browser". The URL in the address bar is "localhost:3000/?orgId=1". The main content shows the "Welcome to Grafana" message and a "Need help?" section with links to documentation, tutorials, community, and public slack. Below this, there are three panels for setup: "Basic", "TUTORIAL DATA SOURCE AND DASHBOARDS", and "COMPLETE". The "Basic" panel contains text about setting up Grafana. The "TUTORIAL" panel is titled "Grafana fundamentals" and describes setting up and understanding Grafana. The "COMPLETE" panel is titled "Add your first data source" and "Create your first dashboard". Each panel has a "Learn how in the docs" link at the bottom. There is also a "Remove this panel" link in the top right of the "COMPLETE" panel.

The screenshot shows the Grafana Dashboards interface. At the top, there's a search bar and a 'New' button. Below it is a table listing six dashboards:

Name	Type	Location	Tags
Alertmanager / Overview	Dashboard	General	alertmanager-mixin
CoreDNS	Dashboard	General	coredns dns
etcd	Dashboard	General	etcd-mixin
Grafana Overview	Dashboard	General	
Kubernetes / API server	Dashboard	General	kubernetes-mixin
Kubernetes / Compute Resources / ...	Dashboard	General	kubernetes-mixin

Example: Grafana Dashboard





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

Activities Firefox Web Browser ▾ Cisco Skills For All Home - Dashboards Grafana Prometheus Time Series ▾ + Tue 10:26 ▾

localhost:9090/targets 170% ⌂ ⓘ ⌂ ⓘ ⌂ ⓘ

Prometheus

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 Duration
http://192.168.59.100:2112/metrics	UP	<code>endpoint="metrics" instance="192.168.59.100:2112" job="tetragon" namespace="kube-system" node="minikube" pod="tetragon-577t1" service="tetragon"</code>	1m 56s ago	12.50s

Discovered labels:

```
_address_ = "192.168.59.100:2112"  
_meta_kubernetes_endpoint_address_target_kind="Pod"  
_meta_kubernetes_endpoint_address_target_name="tetragon-577t1"  
_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"
```