NETWORK AUTOMATION FORUM

Your Logo Here

Dynamic Traffic Generation

Lab1

Mau - Deepinder - Emre

AUTOCON 3

THE NETWORK AUTOMATION CONFERENCE

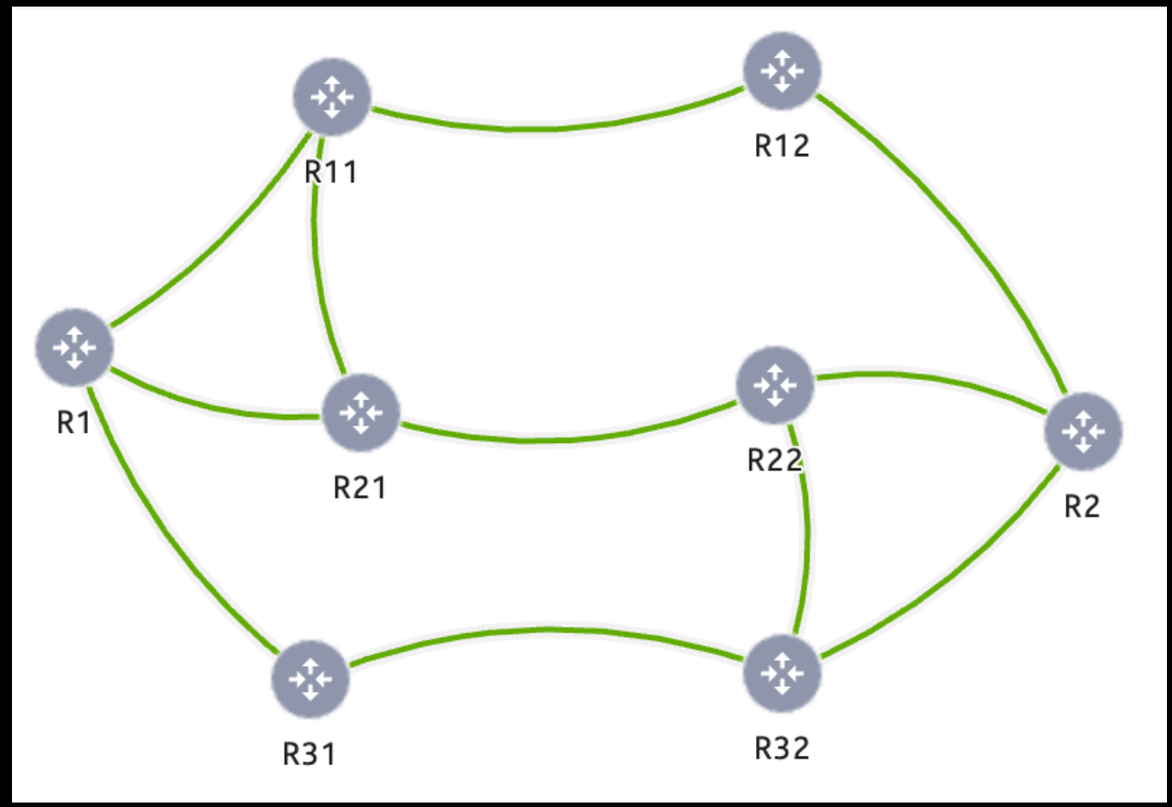
Dynamic Traffic Generation (Deep)

- Deploy a containerized iPerf app in Kind for traffic simulation
- Use custom reconcilers for dynamic test configurations
- Integrate Containerlab and Kind for advanced network automation
- Gain hands-on experience with iPerf, GitHub Codespaces, and VS Code

RSVP-MPLS Use Case

```
topology:
  kinds:
    vr-sros:
      image: vrnetlab/vr-sros:23.7.R1
```

```
 links:
    - endpoints: ["R1:eth1","R11:eth1"]
    - endpoints: ["R1:eth2","R21:eth1"]
    - endpoints: ["R1:eth3","R31:eth1"]
    - endpoints: ["R2:eth1","R12:eth1"]
    - endpoints: ["R2:eth2","R22:eth1"]
    - endpoints: ["R2:eth3","R32:eth1"]
    - endpoints: ["R11:eth2","R21:eth2"]
    - endpoints: ["R21:eth3","R22:eth2"]
    - endpoints: ["R22:eth3","R32:eth2"]
    - endpoints: ["R32:eth3","R31:eth2"]
```

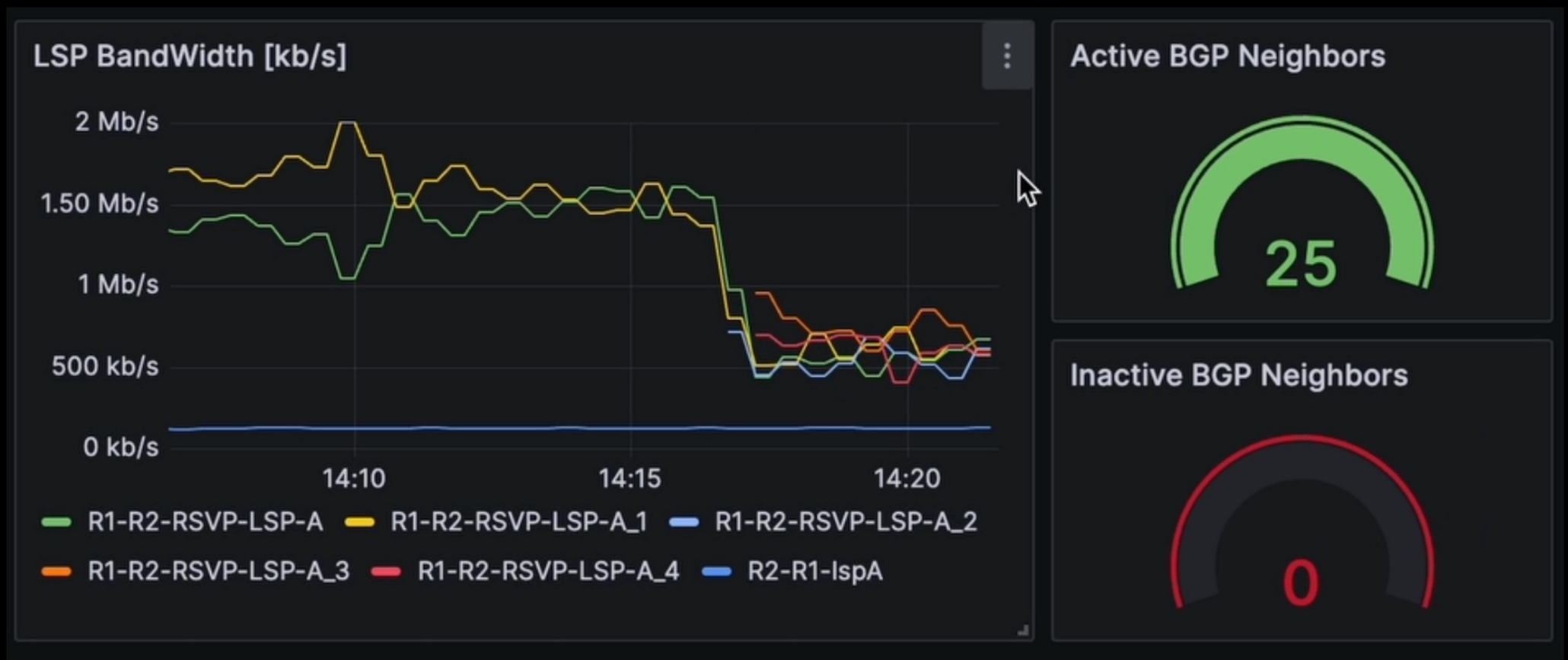


Nokia NSP: Intent LSP Deployments

- RSVP LSP/ ECMP

	Deployment Status	Configuration Status	NE Name	NE ID	Identifier
☐	Deployed Aligned	Modified	R2	1.1.1.2	R2-R1-lspA
☐	Deployed Aligned	Modified	R1	1.1.1.1	R1-R2-RSVP-LSP-A
☐	Deployed Aligned	Modified	R1	1.1.1.1	R1-R2-RSVP-LSP-A_1
☐	Deployed Aligned	Modified	R1	1.1.1.1	R1-R2-RSVP-LSP-A_2
☐	Deployed Aligned	Modified	R1	1.1.1.1	R1-R2-RSVP-LSP-A_3
☑	Deployed Aligned	Modified	R1	1.1.1.1	R1-R2-RSVP-LSP-A_4

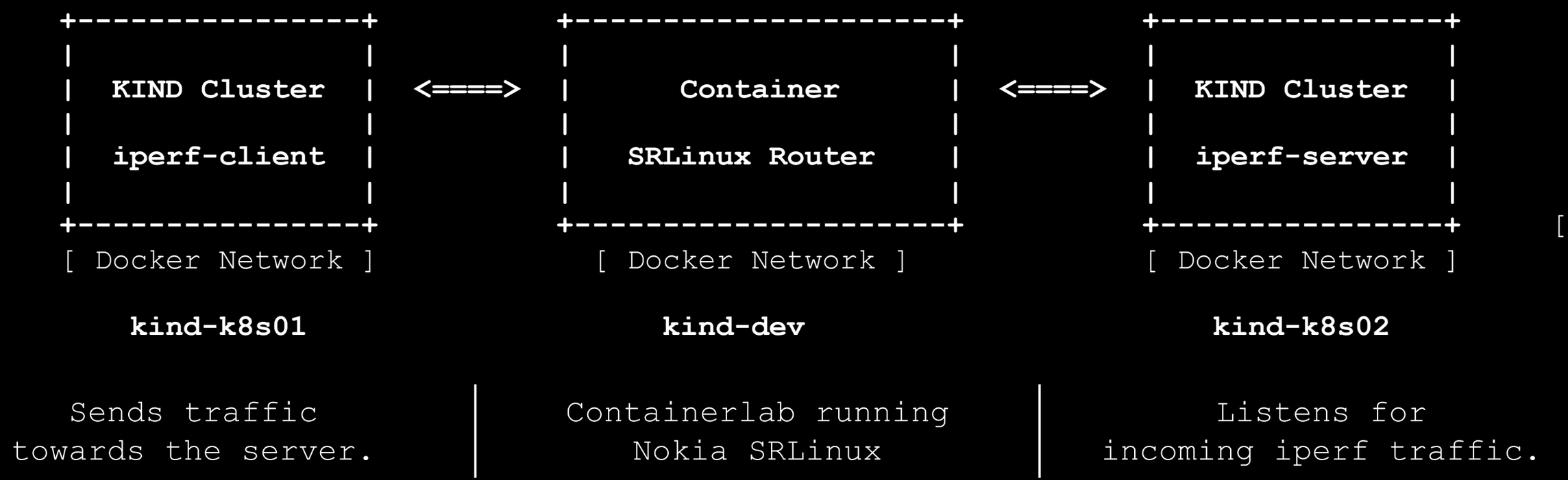
Nokia NSP: Dynamic Traffic Generator

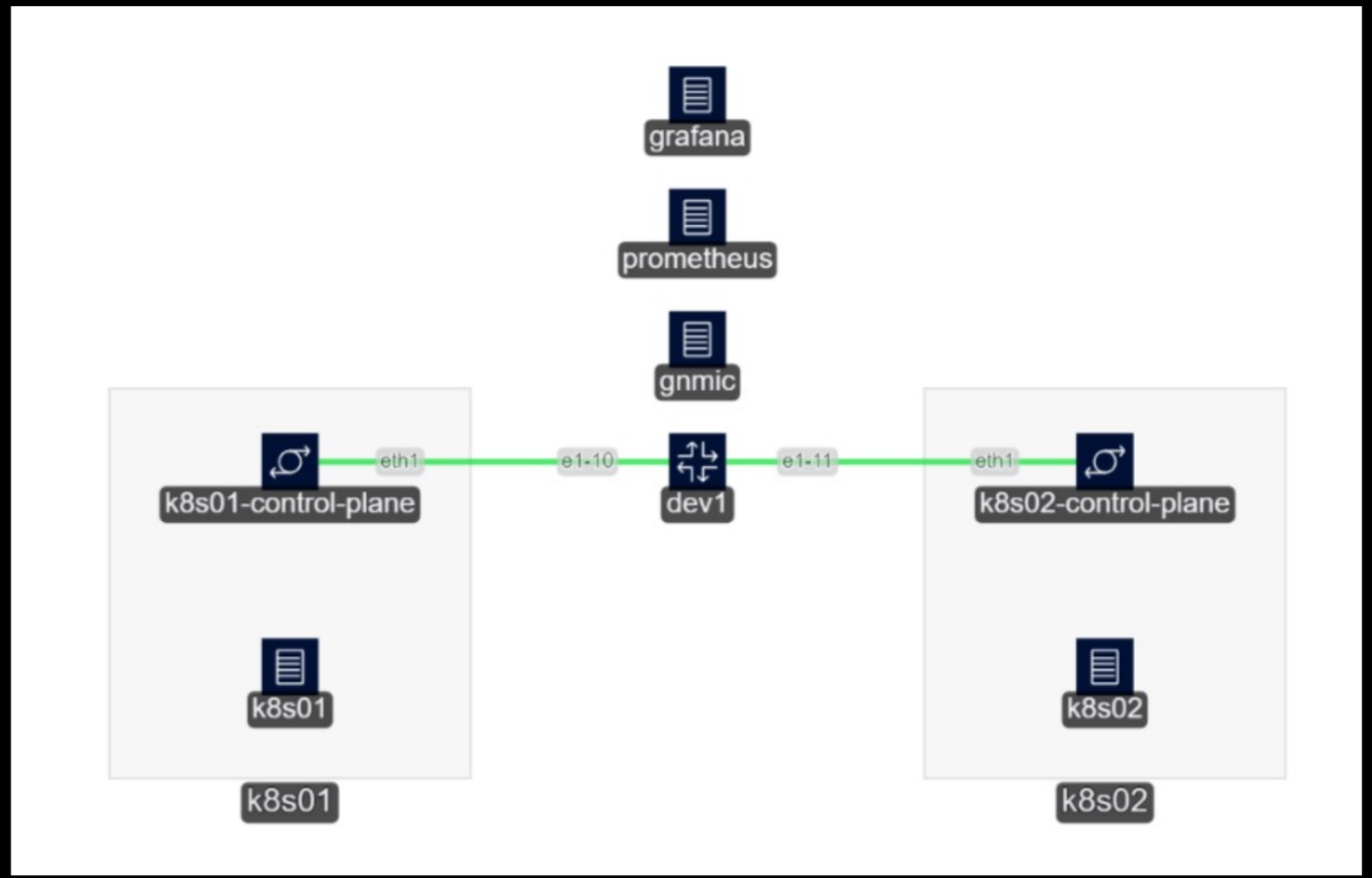



NAF
NETWORK AUTOMATION FORUM
Now The Lab!


NAF
NETWORK AUTOMATION FORUM
Pray Before We Start
"Dear Demo Gods, grant us stable networks, responsive systems, and merciful bugs. May every click lead where it should, and may technology be kind to us today."

Lab1 Topology Representation




grafana
prometheus
gnmic
eth1
e1-10
e1-11
eth1
k8s01-control-plane
dev1
k8s02-control-plane
k8s01
k8s02
k8s01
k8s02

Lab 1

Working Directory

```
cd /workspaces/autocon3-ws-a3/lab1
```

Building Docker Images

Docker build

```
docker build -t iperf3-client:0.1a -f iperf-images/Dockerfile.iperf3client .
```

Dockerfile

```
# Set up environment variables for customization
ENV REMOTE_SERVICE_IP=172.254.102.101
ENV CUSTOM_PORT=5201
ENV CUSTOM_NUMBER_OF_FLOWS=1
ENV CUSTOM_TIME_SECONDS=10

# Command to run the iperf3 client with custom parameters
CMD ["sh", "-c", "iperf3 -c $REMOTE_SERVICE_IP -p $CUSTOM_PORT -P
$CUSTOM_NUMBER_OF_FLOWS -t $CUSTOM_TIME_SECONDS"]
```

Building Docker Images

Docker build

```
docker build -t iperf3-server:0.1a -f iperf-images/Dockerfile.iperf3server .
```

Dockerfile

```
# Set up environment variables for customization
ENV SERVER_PORT=5201

# Expose the custom port
EXPOSE $SERVER_PORT

# Command to run the iperf3 server on the custom port
CMD ["sh", "-c", "iperf3 -s -p $SERVER_PORT"]
```

Load pre-cached Container Images

```
docker image load -i /var/cache/srlinux.tar
```

```
docker image load -i /var/cache/kindest-node.tar
```

Setting up the Container Networking

Pre-create the Kind Docker Bridge

```
docker network create -d=bridge \
    -o com.docker.network.bridge.enable_ip_masquerade=true \
    -o com.docker.network.driver.mtu=1500 \
    --subnet fc00:f853:ccd:e793::/64 kind
```

Add iptable Rules

```
sudo iptables -I DOCKER-USER -o br-$(docker network
inspect -f '{{ printf "%.12s" .ID }}' kind) -j ACCEPT
```

Clab topology

```
 nodes:
    k8s01:
      kind: k8s-kind
    k8s02:
      kind: k8s-kind
    dev1:
      kind: nokia_srlinux
links:
    - endpoints: ["dev1:e1-10", "k8s01:eth1"]
    - endpoints: ["dev1:e1-11", "k8s02:eth1"]
```

ContainerLab Topology

Start the Topology

```
cd clab-topology
sudo containerlab deploy
cd ..
```

Containerlab Contexts

```
sudo kubectl config get-contexts
CURRENT   NAME          CLUSTER       AUTHINFO      NAMESPACE
          kind-k8s01    kind-k8s01    kind-k8s01
*         kind-k8s02    kind-k8s02    kind-k8s02
```

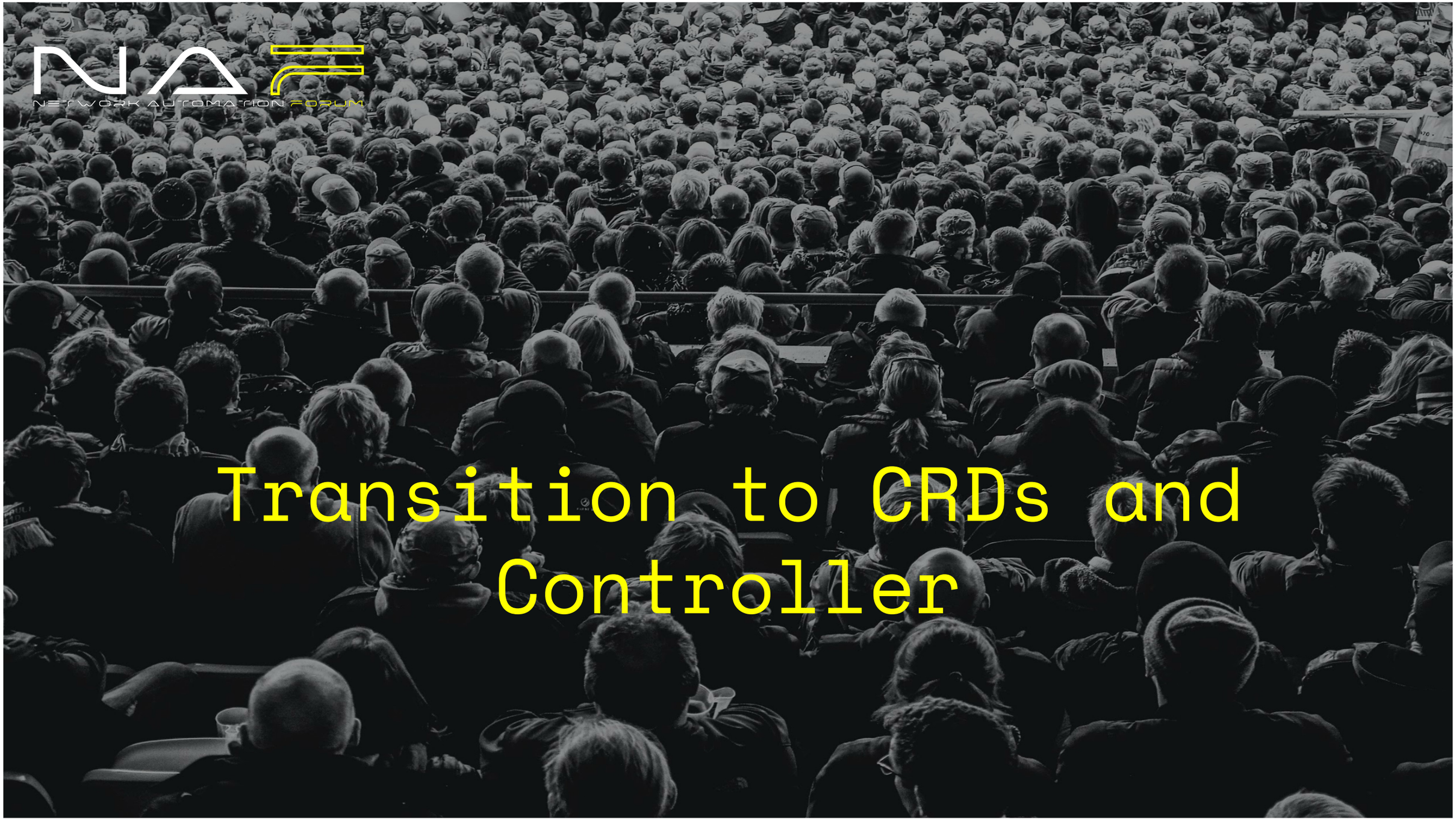
Load iperf images to KinD K8s

```
kind load docker-image iperf3-client:0.1a --name k8s01
kind load docker-image iperf3-server:0.1a --name k8s02
```

Setting up dev1

```
docker exec -ti dev1 sr_cli
```

```
enter candidate
    network-instance default {
      interface ethernet-1/10.0 {
      }
      interface ethernet-1/11.0 {
      }
...
```



Transition to CRDs and Controller

Manifests

```
./iperf-server-controller.yaml
./iperf-client-setup.yaml
./iperf-client-crd.yaml
./iperf-server-configmap.yaml
./iperf-client-controller.yaml
./iperf-server-crd.yaml
./iperf-client-configmap.yaml
./iperf-server-setup.yaml
```

Traffic generators CRDs

```
sudo kubectl apply -f ./CRDs/iperf-server-crd.yaml --context
kind-k8s02
sudo kubectl apply -f ./CRDs/iperf-client-crd.yaml --context
kind-k8s01
```

```
sudo kubectl get CustomResourceDefinition  --context kind-k8s02
NAME                              CREATED AT
iperfservers.example.com          2025-01-07T21:52:04Z
sudo kubectl get CustomResourceDefinition  --context kind-k8s01
NAME                              CREATED AT
iperfclients.example.com          2025-01-07T21:52:04Z
```

Traffic generators CRDs (Client)

```yaml
kind: CustomResourceDefinition
metadata:
  name: iperfclients.example.com
spec:
  group: example.com
  names:
    kind: IperfClient
    listKind: IperfClientList
    plural: iperfclients
    singular: iperfclient
  scope: Namespaced
```

```yaml
spec:
  type: object
  properties:
    targetIP:
      type: string
    initPort:
      type: integer
    endPort:
      type: integer
    image:
      type: string
```

Traffic generators CRDs (Server)

```yaml
kind: CustomResourceDefinition
metadata:
  name: iperfservers.example.com
spec:
  group: example.com
  names:
    kind: IperfServer
    listKind: IperfServerList
    plural: iperfservers
    singular: iperfserver
  scope: Namespaced
```

```yaml
spec:
  type: object
  properties:
    replicas:
      type: integer
    initPort:
      type: integer
    endPort:
      type: integer
    image:
      type: string
```

iperf server/client setup

```
apiVersion: example.com/v1
kind: IperfClient
metadata:
  name: iperf3-client
spec:
  targetIP: "172.254.102.101"
  initPort: 30001
  endPort: 30007
  image: iperf3-client:0.1a
```

```
apiVersion: example.com/v1
kind: IperfServer
metadata:
  name: iperf3-server
spec:
  replicas: 10
  initPort: 30001
  endPort: 30010
  image: iperf3-server:0.1a
```

ConfigMap / Controller Code

```
kind: ConfigMap
metadata:
  name: iperf-client-controller
data:
  controller.py: |
    from kubernetes import client, config
    ...
    # Load in-cluster configuration
    ...
    def reconcile_iperf_client():
    def create_or_update_pods(name, spec):
...
```

Reconciler in ConfigMap

```
sudo kubectl apply -f ./CRDs/iperf-server-configmap.yaml --context
kind-k8s02
sudo kubectl apply -f ./CRDs/iperf-server-controller.yaml --context
kind-k8s02
sudo kubectl apply -f ./CRDs/iperf-client-configmap.yaml --context
kind-k8s01
sudo kubectl apply -f ./CRDs/iperf-client-controller.yaml --context
kind-k8s01
```

Verify Controller

```
sudo kubectl get pods --context kind-k8s02

NAME                        READY   STATUS    RESTARTS   AGE
iperf-server-controller     1/1     Running   0          2m36s


sudo kubectl get pods --context kind-k8s01

NAME                        READY   STATUS    RESTARTS   AGE
iperf-client-controller     1/1     Running   0          2m36s
```

Server CRDs Values

./CRDs/iperf-server-setup.yaml

```
apiVersion: example.com/v1
kind: IperfServer
metadata:
  name: iperf3-server
spec:
  replicas: 10
  initPort: 30001
  endPort: 30010
  image: iperf3-server:0.1a
```

```
sudo kubectl apply -f ./CRDs/iperf-server-setup.yaml --
context kind-k8s02
```

Verify Server

```
sudo kubectl get pods --context kind-k8s02
```

```
NAME                                READY   STATUS    RESTARTS   AGE
iperf-server-controller             1/1     Running   0          4m17s
iperf3-server-0-cdfcf7f6b-98ccm     1/1     Running   0          10s
iperf3-server-1-66597c5d6f-lph4j    1/1     Running   0          10s
iperf3-server-2-78469bdcbd-7rzkl    1/1     Running   0          10s
iperf3-server-3-58779bc979-c2knl    1/1     Running   0          10s
iperf3-server-4-656545d446-6m54g    1/1     Running   0          10s
iperf3-server-5-5cd44f9cf7-87kwx    1/1     Running   0          10s
iperf3-server-6-6b5fd6cc9b-rbqmc    1/1     Running   0          10s
iperf3-server-7-5594487d47-p7pk9    1/1     Running   0          9s
iperf3-server-8-76b5fb8794-p9x77    1/1     Running   0          9s
iperf3-server-9-5b469c667c-xwhd7    1/1     Running   0          9s
```

Client CRDs Values

./CRDs/iperf-client-setup.yaml

```
apiVersion: example.com/v1
kind: IperfClient
metadata:
  name: iperf3-client
spec:
  targetIP: "172.254.102.101"  # Replace with server cluster IP
  initPort: 30001
  endPort: 30007
  image: iperf3-client:0.1a
```

```
sudo kubectl apply -f ./CRDs/iperf-client-setup.yaml --
context kind-k8s01
```

Verify Client

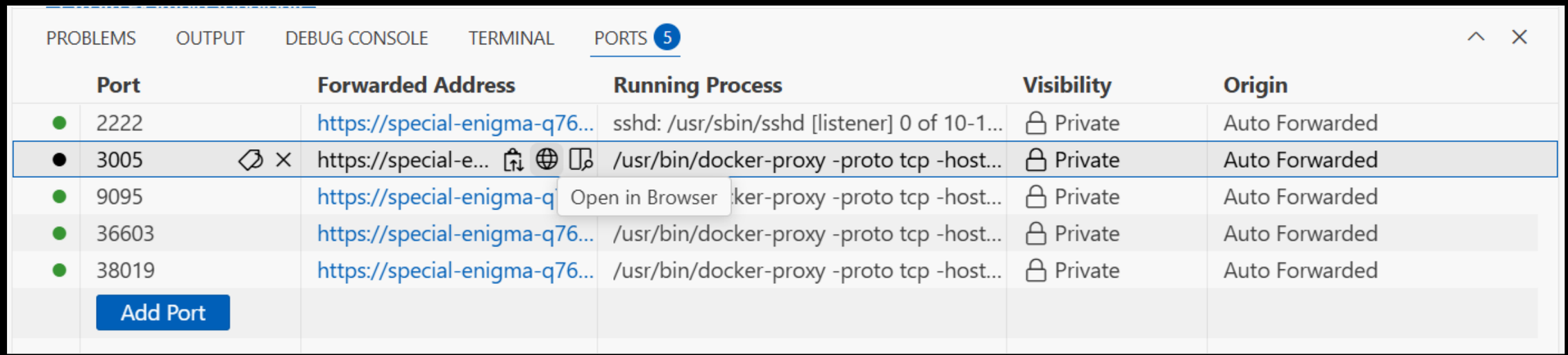
```
sudo kubectl get pods --context kind-k8s01

NAME                          READY   STATUS    RESTARTS   AGE
iperf-client-controller       1/1     Running   0          100m
iperf3-client-30001           1/1     Running   0          99m
iperf3-client-30002           1/1     Running   0          99m
iperf3-client-30003           1/1     Running   0          99m
iperf3-client-30004           1/1     Running   0          99m
iperf3-client-30005           1/1     Running   0          87m
```

Inspect Logs to Verify Traffic

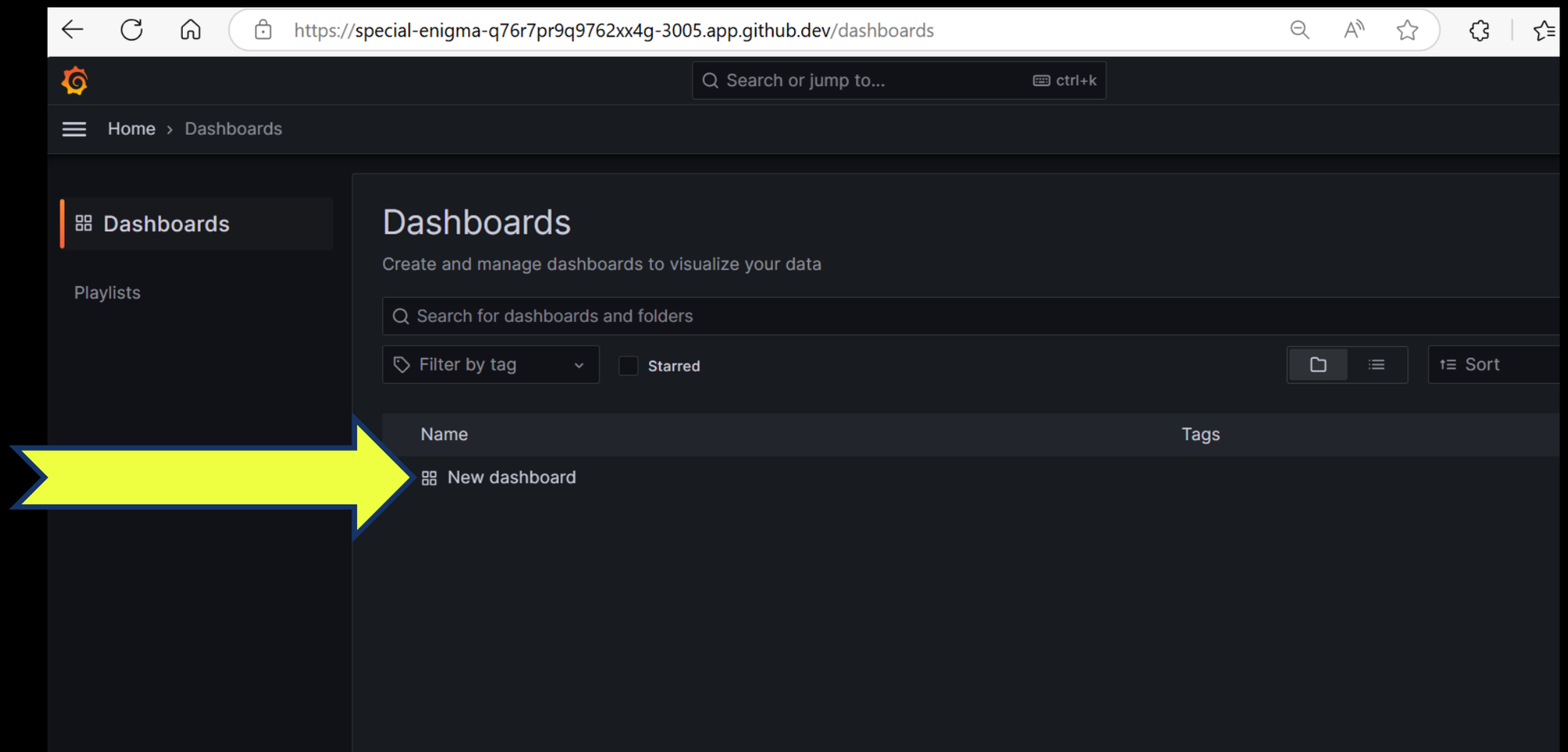
```
sudo kubectl logs iperf3-client-30001 --context kind-k8s01
```

Gnmi/grafana

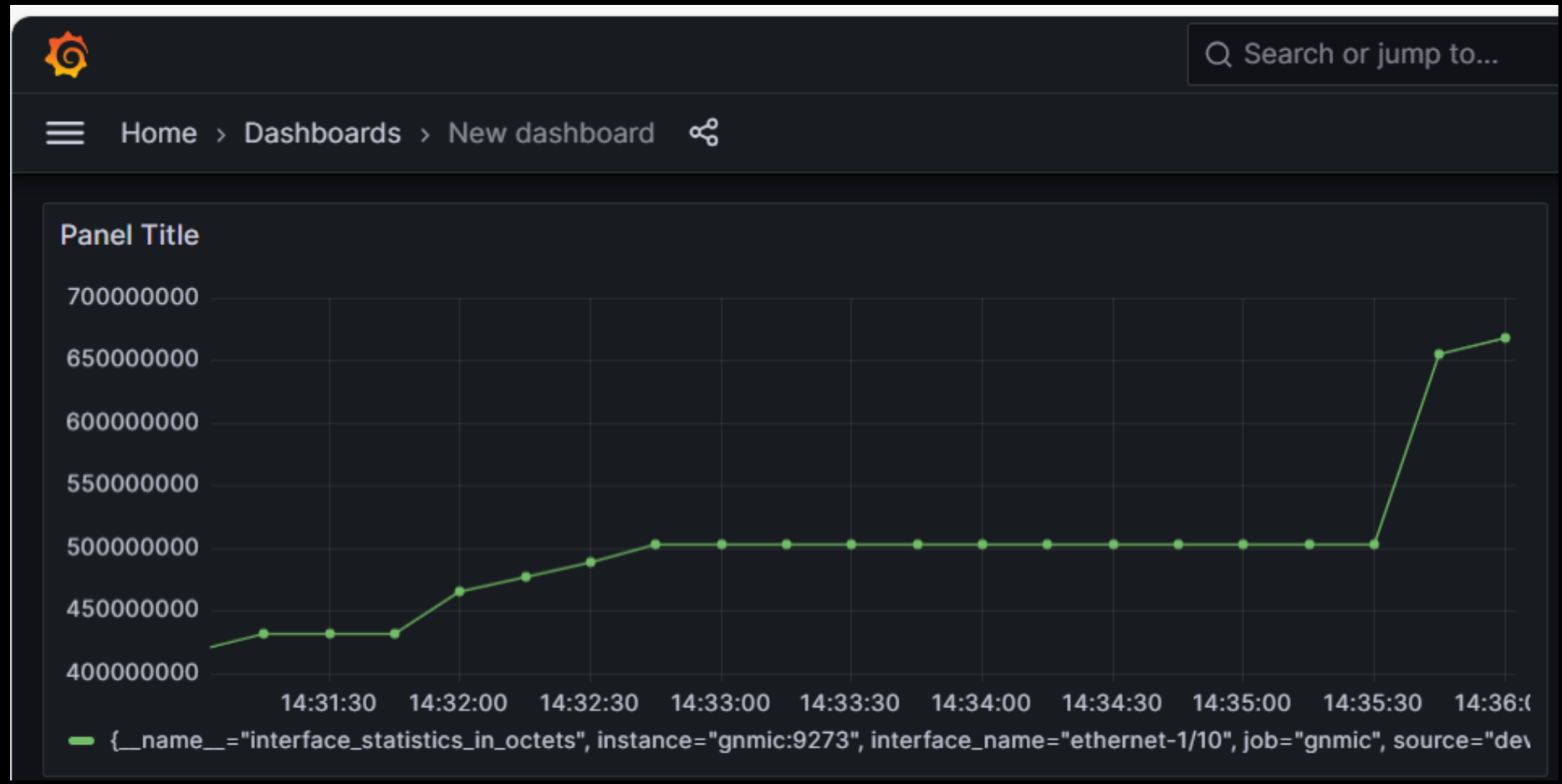
- Open **Grafana Dashboard**. In this case Grafana resides on port 3005



Grafana Dashboard



Visualize the Traffic



Cleanup

```
sudo kubectl delete -f ./CRDs/iperf-client-setup.yaml --context kind-k8s01
sudo kubectl delete -f ./CRDs/iperf-server-setup.yaml --context kind-k8s02
sudo kubectl delete -f ./CRDs/iperf-server-controller.yaml --context kind-k8s02
sudo kubectl delete -f ./CRDs/iperf-server-configmap.yaml --context kind-k8s02
sudo kubectl delete -f ./CRDs/iperf-client-controller.yaml --context kind-k8s01
sudo kubectl delete -f ./CRDs/iperf-client-configmap.yaml --context kind-k8s01
sudo kubectl delete -f ./CRDs/iperf-server-crd.yaml --context kind-k8s02
sudo kubectl delete -f ./CRDs/iperf-client-crd.yaml --context kind-k8s01
```


NAF
NETWORK AUTOMATION FORUM
Questions?