

# Deploying Monarch

Deployment involves 4 primary phases:

- 1. External Components:** Deploy the Service Orchestrator and NFV Orchestrator, ensuring they are prepared to manage and support Monarch's functions.
- 2. Core Monarch Components:** Set up Monarch's core modules that handle critical processing and system coordination.
- 3. Network Slice Segment (NSS) Components:** Install the NSS components across all segments for full slice coverage. For our workshop, we only have one segment.
- 4. Slice Monitoring and KPI Modules:**  
Submit a monitoring request to trigger the creation of Monitoring Data Exporter and KPI Computation Modules.

# Before Starting Deployment

## 1. Navigate to the home directory

Use the `cd` command to ensure you're starting from your home directory.

## 2. Clone the Monarch repository

Use `git clone` to fetch the source code from the Monarch GitHub repository.

```
cd ~  
git clone https://github.com/niloysh/5g-monarch.git  
cd 5g-monarch
```

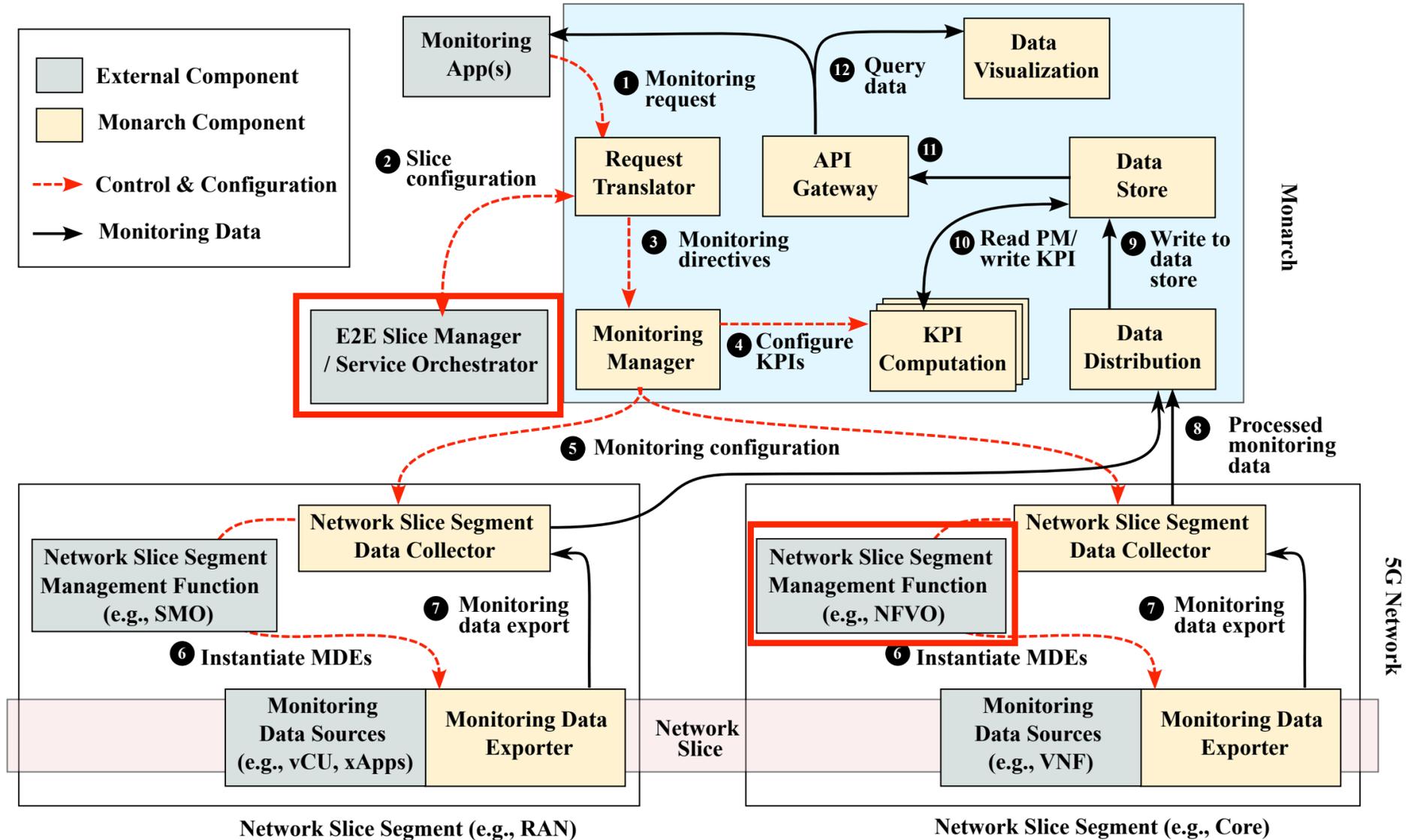
## 3. Set Up Environment Variables

Monarch involves several interconnected components, so it's crucial to configure them correctly to ensure they can communicate with each other.

- Open the `.env` file in VSCode and locate the `NODE_IP` field.
- Run the `./replace-node-ip.sh` script to automatically add the correct node IP.

# Phase 1 - Deploying External Components

# External Components



# Deploying External Components

## 1. Run the deployment script

Execute the following command to set up essential external dependencies for Monarch, including the **Service Orchestrator** and **NFV Orchestrator**.

```
./deploy-external.sh
```

### What this does:

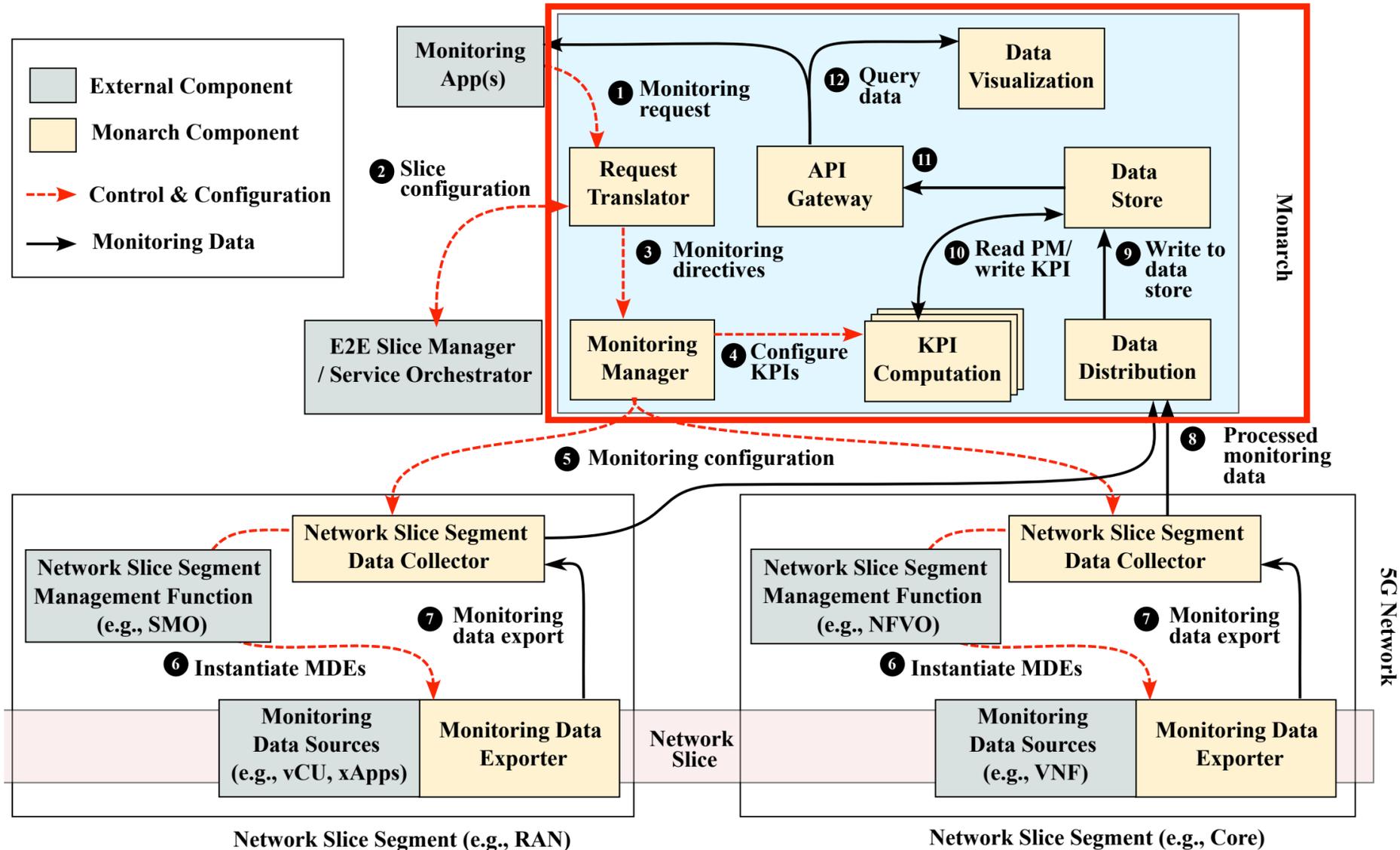
- **Service Orchestrator:** Allows Monarch to retrieve information about network functions (NFs) in each slice and the segments in which these NFs are deployed.
- **NFV Orchestrator:** Manages the deployment of network functions (NFs) and Monarch's data exporters within the network.

**Note:** In this workshop, we assume simple implementations of these components for demonstration purposes. Projects such as ONAP (Open Network Automation Platform) provide can realistic implementations for these components.



# **Phase 2 - Deploying Monarch Core Components**

# Monarch Core Components



# Deploying Monarch Core

## 1. Run the deployment script

```
./deploy-monarch-core.sh
```

This script will initialize key core modules, including:

- **Request Translator and Monitoring Manager** – Deployed using Python Flask, these components handle translation of monitoring requests and management of monitoring tasks.
- **Data Store** – Configured with MongoDB for database storage and MinIO S3 for scalable object storage.
- **Data Distribution** – Uses Thanos to efficiently collect monitoring data from different network segments.
- **Data Visualization** – Leverages Grafana for comprehensive visualization of key performance indicators (KPIs) and metrics.

# Verifying Monarch Core Deployment

Monarch components are deployed in the `monarch` namespace.

While Monarch components are being deployed, you can use `kubectl get pods -n monarch` with the `watch` command in a **new terminal** to see the progress.

```
watch kubectl get pods -n monarch
```

```
Every 2.0s: kubectl get pods -n monarch
```

| NAME  | READY | STATUS  | RESTARTS | AGE   |
|---|-------|---------|----------|-------|
| datadist-thanos-query-8f79545dd-5nmgh           | 1/1   | Running | 0        | 5m33s |
| datadist-thanos-query-frontend-65cb6b64df-rcn5v | 1/1   | Running | 0        | 5m33s |
| datadist-thanos-receive-0                       | 1/1   | Running | 0        | 5m33s |
| datadist-thanos-storegateway-0                  | 1/1   | Running | 0        | 5m33s |
| datastore-minio-695cb778d5-jbbf7                | 1/1   | Running | 0        | 6m7s  |
| datastore-mongodb-0                             | 1/1   | Running | 0        | 5m55s |
| dataviz-grafana-55b6ffc6bf-25nrx                | 1/1   | Running | 0        | 4m58s |
| monitoring-manager-bf4c87cc6-mqs76              | 1/1   | Running | 0        | 4m42s |
| request-translator-88d6f4469-5cblz              | 1/1   | Running | 0        | 4m32s |

# Testing Monarch Core Components

Once deployed, you can test the Monarch core components with the testing script.

```
./test-monarch-core.sh
```

```
##### Testing Request Translator (Monarch Core [5/5]) #####  
--- Performing health check for Request Translator ---  
INFO: Checking health status of the service...  
Service health check passed! The service is up and running.  
--- Checking supported KPIs ---  
INFO: Fetching supported KPIs...  
Supported KPIs retrieved successfully:  
{  
  "status": "success",  
  "supported_kpis": [  
    {  
      "kpi_description": "Throughput per slice",  
      "kpi_name": "slice_throughput",  
      "kpi_unit": "bits/sec"  
    }  
  ]  
}
```

# Accessing Graphical User Interfaces (GUIs)

Some Monarch components provide user-friendly GUIs, enabling you to monitor and visualize data effectively.

## Thanos (Data Distribution)

- Access Thanos at: <http://localhost:31004>
- Use Thanos to explore distributed data storage and retrieve historical monitoring data. We will see how to use Thanos later in [Lab3](#).

## Grafana (Data Visualization)

- Access Grafana at: <http://localhost:32005>. Login details are: **username:** `admin` and **password:** `monarch-operator`.
- Grafana enables interactive dashboards and visualization of key performance indicators (KPIs). After submitting a monitoring request, you'll view a slice KPI monitored live in the Grafana dashboard.

# Phase 3 - Deploying Monarch NSS Components

# Deploy Monarch NSS Components

## 1. Run the deployment script

```
./deploy-monarch-nss.sh
```

### What this does:

This script initializes the NSSDC (Network Slice Segment Data Collector), which is implemented using **Prometheus**.

**Prometheus** is an open-source monitoring toolkit. It collects and stores metrics as time series data, offering powerful querying capabilities and seamless integrations with tools like Grafana for visualization.

We will cover Prometheus in more detail in **Lab 1** and **Lab 2**, where you will learn how to instrument an application to expose metrics to Prometheus and work with the Prometheus query language.

# Verify and Test Monarch NSS Deployment

Monitor the status of the `prometheus-nssdc-prometheus-0` pod. Wait for it to become `READY (3/3)`.

Once it is ready, you can test the Monarch NSS components with the testing script.

```
./test-monarch-nss.sh
```

```
##### Testing NSSDC (Monarch NSS [1/1]) #####  
--- Performing health check for NSSDC ---  
Health check passed! NSSDC (Prometheus) is healthy.  
--- Fetching active targets from API ---  
INFO: Active scrape pools found:
```

You can access the Prometheus GUI at: <http://localhost:30095>

# Phase 4 - Network Slice Monitoring

# Submitting a Slice Monitoring Request (1/4)

To submit a slice monitoring request, we interact with the **Request Translator** module, which processes the request and forwards it for further actions.

## 1. Create the JSON Request

The request is submitted using a JSON file, where you define the KPI for monitoring.

Here's a snippet from the `request_translator/requests/request_slice.json` file:

```
{"kpi":  
  {"kpi_name": "slice_throughput",  
    "kpi_description": "Throughput of the network slice",  
    "sub_counter": {"sub_counter_type": "SNSSAI",  
                    "sub_counter_ids": ["1-000001", "2-000002"]}}
```

We can submit a request as follows:

```
cd request_translator  
python3 test_api.py --json_file requests/request_slice.json submit
```

# Submitting a Slice Monitoring Request (2/4)

After submitting the slice monitoring request, Monarch will initiate the creation of **Monitoring Data Exporters (MDEs)** and the **KPI computation module**.

## 1. Verify MDE Creation

Once the request is processed, three metrics services should be created, corresponding to the MDEs. You can verify their creation by running the following command:

```
kubectl get svc -n open5gs | grep metrics
```

## 2. Check for KPI Calculator Pod

Next, in your terminal where the `kubectl get pods -n monarch` command is running, you should observe a new pod for the KPI Calculator. This pod is responsible for calculating and processing KPIs based on the monitoring data.

# Submitting a Slice Monitoring Request (3/4)

## 3. View Logs of the KPI Calculator

To inspect the logs of the KPI calculator pod, use the following command:

```
kubectl logs deployments/kpi-calculator -n monarch
```

In the logs, you should see output indicating that the slice SNSSAI is being monitored, but with a rate of **0 Mbps** (since no traffic is flowing through the slice during this phase).

# Submitting a Slice Monitoring Request (4/4)

## 4. Send Traffic to Trigger KPI Calculations

To observe the monitoring in action, send traffic through the slice(s) by performing a ping test.

You can do this from the **UE1** and **UE2** pods, using the `uesimtun0` interface, just like we did earlier in the core deployment session.

```
kubectl exec -it deployments/ueransim-ue1 -n open5gs -- /bin/bash
```

```
ping -I uesimtun0 www.google.ca
```

Once you've started the ping test, keep it running and check the KPI calculator logs again.

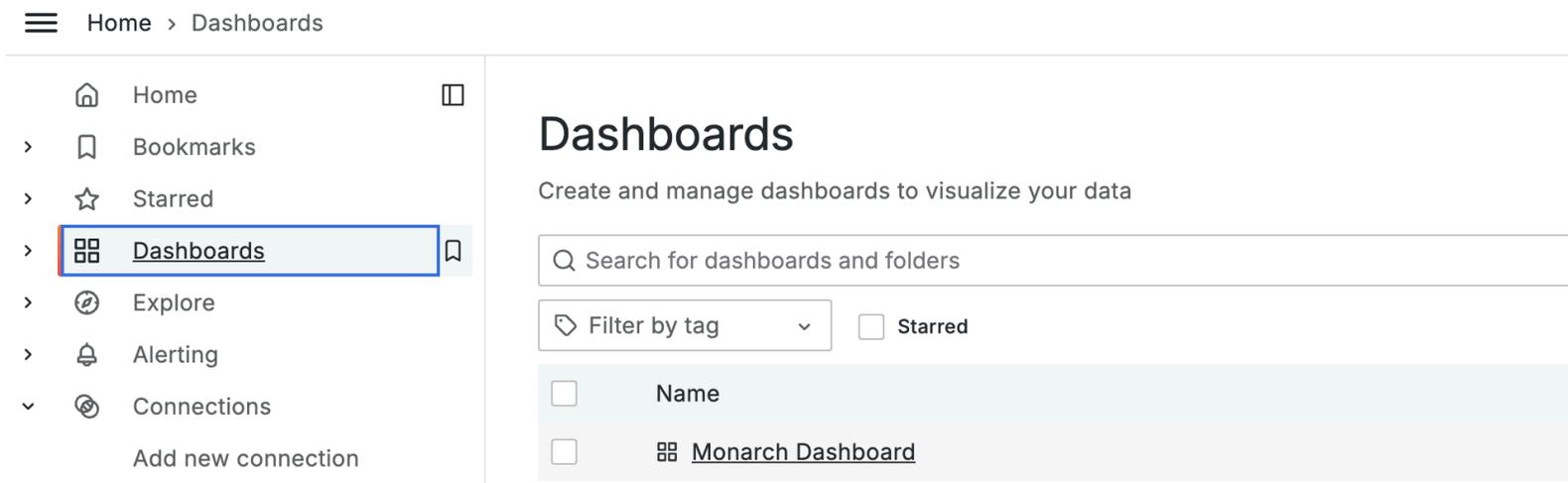
# Visualizing Slice KPI using Grafana (1/4)

## 1. Access Grafana

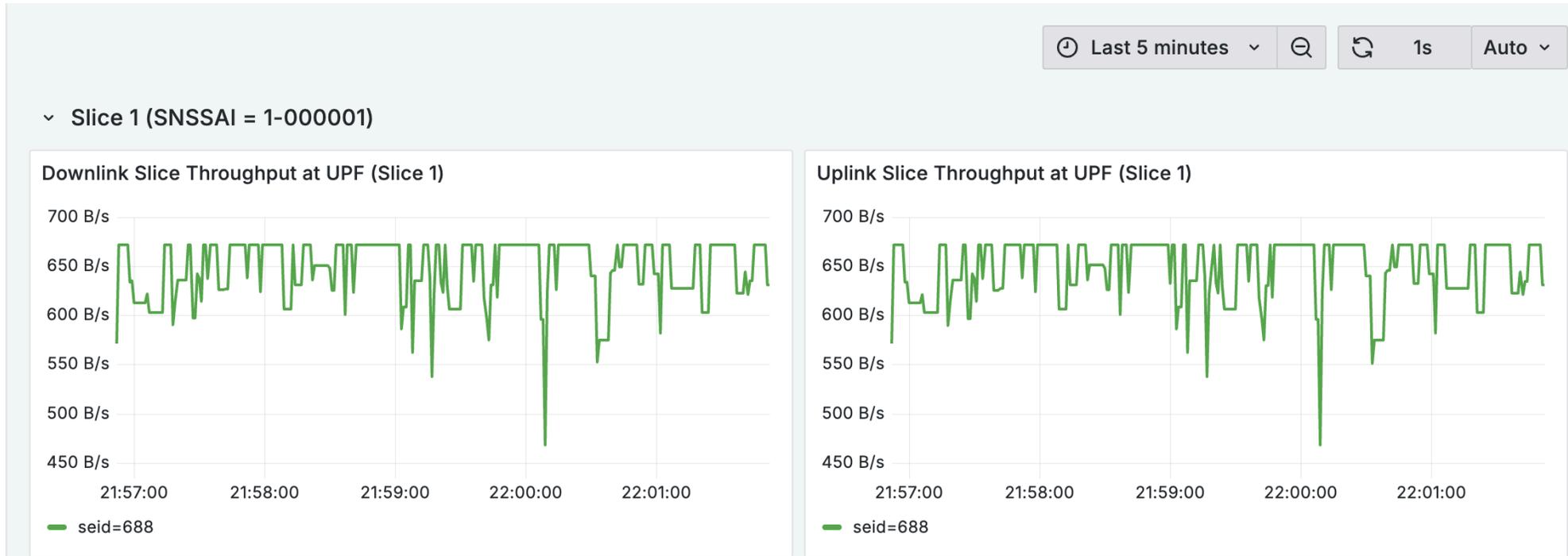
Open the Grafana interface (<http://localhost:32005>) and log in using the following credentials: **username:** `admin` and **password:** `monarch-operator`.

## 2. Navigate to the Monarch Dashboard

Once logged in, select **Dashboards** from the left sidebar. Locate and open the **Monarch Dashboard**, which has been pre-configured to display the `slice_throughput` KPI for our two slices.



# Visualizing Slice KPI using Grafana (2/4)

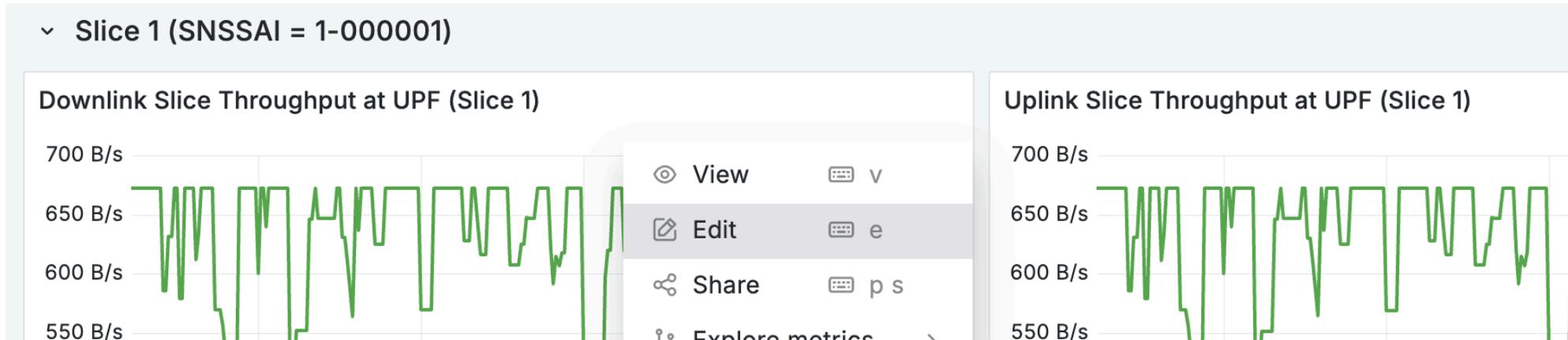


Each panel shows throughput per slice and direction (uplink/downlink). At the bottom, you'll find the **Session IDs (SEIDs)** associated with each slice.

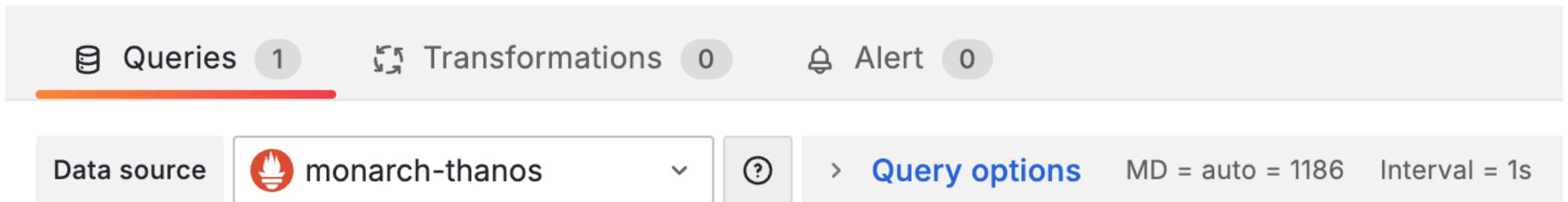
If you deployed UE3 in our previous core deployment session, you should see two SEIDs displayed here, corresponding to multiple active sessions.

# Visualizing Slice KPI using Grafana (3/4)

To customize a panel, click the 3 little dots in the corner, and select the Edit option:



In the edit view, you'll notice the **Data Source** – The data source for this panel is Thanos, which aggregates and retrieves monitoring data.



# Visualizing Slice KPI using Grafana (4/4)

Panel: A (monarch-thanos)

Kick start your query Explain

Metrics browser > `slice_throughput{direction="downlink",snssai="1-000001"}`

> Options Legend: seid={{seid}} Format: Time series Step: auto Type: Range Exemplars: false

In the **Metrics Browser** section, you'll see a query written in PromQL (Prometheus Query Language). This query defines how the data is fetched and displayed on the panel. We will look at PromQL in detail in our **Labs**.

# Looking Under the Hood

Now that the slice KPIs are being computed, let's run Monarch core test script again.

```
./test-monarch-core.sh
```

This time, in the **Monitoring Manager** test output, you should see the monitoring directives that were generated from the high-level monitoring request we submitted earlier.

```
{
  "metrics": [
    "fivegs_ep_n3_gtp_outdatavolumen3upf_seid_total",
    "fivegs_ep_n3_gtp_outdatavolumen3upf_seid_total"
  ],
  "nf": "upf",
  "nss": "edge",
  "pod_ip": "10.244.0.133",
  "pod_name": "open5gs-upf2-5c878f6688-m2thj",
  "type": "pod"
}
```

# Next Steps

## Congratulations!

You've successfully completed the following:

- Completed deployment of Monarch.
- Submitted a slice monitoring request.
- Visualized slice KPIs in Grafana.

## What's Next?

Continue to [Lab 1](#) to learn about using instrumenting applications with metrics support and querying metrics.