TeraFlowSDN WebUI and QKD Nodes Setup and Management Guide

Introduction

This document provides a structured guide on setting up, simulating, and managing Quantum Key Distribution (QKD) nodes within the TeraFlowSDN WebUI. By following this guide, users will gain insight into the core functionalities of the WebUI, including deploying the environment, configuring QKD nodes, establishing secure connections, and utilizing service and application management features. This demonstration assumes familiarity with software-defined networking (SDN) principles and the foundational aspects of TeraFlowSDN.

Step 1: Deployment and Initial Setup

Objective

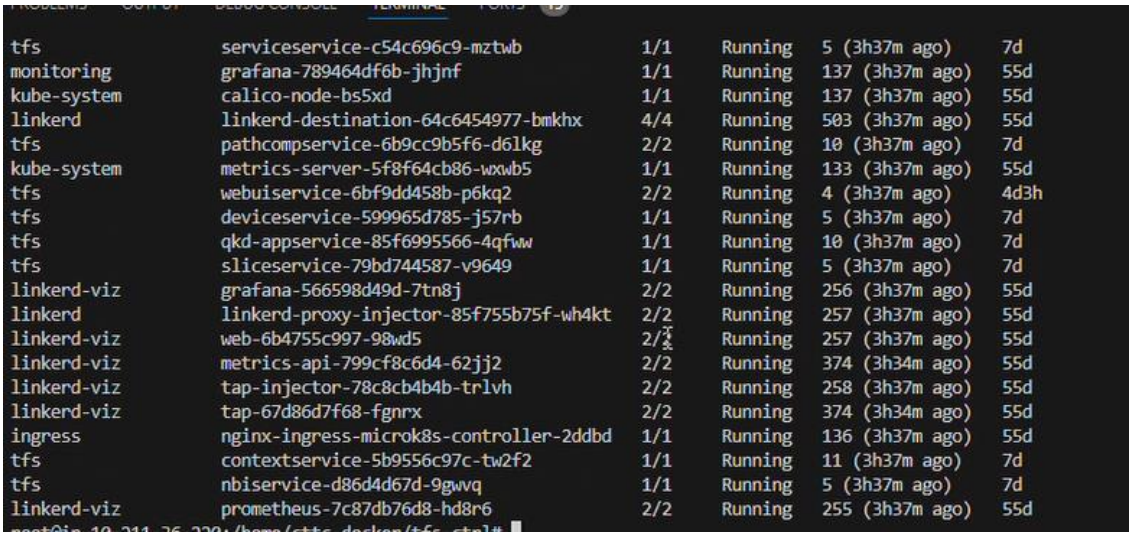
The first step is to deploy the TeraFlowSDN environment to ensure that the WebUI and all related components are correctly initialized and operational.

1. **Navigate to the Project Directory:** Access the TeraFlowSDN project directory where deployment scripts are located.
2. **Execute the Deployment Commands:**

```
source my_deploy.sh
./deploy/all.sh
```

This command sequence initiates the TeraFlowSDN controller, deploying core services required for the WebUI and SDN functionalities.

3. **Verify Deployment:** To confirm that all TeraFlowSDN components are running, verify the status of the controller pods. Once confirmed, access the WebUI via **http://{IP}/webui.**



```
tfs             serviceservice-c54c696c9-mztwb           1/1   Running   5 (3h37m ago)    7d
monitoring      grafana-789464df6b-jhjnf                 1/1   Running   137 (3h37m ago)  55d
kube-system     calico-node-bs5xd                        1/1   Running   137 (3h37m ago)  55d
linkerd         linkerd-destination-64c6454977-bmkhx     4/4   Running   503 (3h37m ago)  55d
tfs             pathcompservice-6b9cc9b5f6-d6lkg         2/2   Running   10 (3h37m ago)   7d
kube-system     metrics-server-5f8f64cb86-wxwb5          1/1   Running   133 (3h37m ago)  55d
tfs             webuiservice-6bf9dd458b-p6kq2            2/2   Running   4 (3h37m ago)    4d3h
tfs             deviceservice-599965d785-j57rb           1/1   Running   5 (3h37m ago)    7d
tfs             qkd-appservice-85f6995566-4qfww          1/1   Running   10 (3h37m ago)   7d
tfs             sliceservice-79bd744587-v9649            1/1   Running   5 (3h37m ago)    7d
linkerd-viz     grafana-566598d49d-7tn8j                 2/2   Running   256 (3h37m ago)  55d
linkerd         linkerd-proxy-injector-85f755b75f-wh4kt  2/2   Running   257 (3h37m ago)  55d
linkerd-viz     web-6b4755c997-98wd5                     2/2   Running   257 (3h37m ago)  55d
linkerd-viz     metrics-api-799cf8c6d4-62jj2             2/2   Running   374 (3h34m ago)  55d
linkerd-viz     tap-injector-78c8cb4b4b-trlvh            2/2   Running   258 (3h37m ago)  55d
linkerd-viz     tap-67d86d7f68-fgnrx                     2/2   Running   374 (3h34m ago)  55d
ingress         nginx-ingress-microk8s-controller-2ddbd  1/1   Running   136 (3h37m ago)  55d
tfs             contextservice-5b9556c97c-tw2f2          1/1   Running   11 (3h37m ago)   7d
tfs             nbiservice-d86d4d67d-9gwvq               1/1   Running   5 (3h37m ago)    7d
linkerd-viz     prometheus-7c87db76d8-hd8r6              2/2   Running   255 (3h37m ago)  55d
```

Explanation

This setup configures the WebUI backend, allowing interaction with QKD nodes and services. Ensuring all pods are active is critical, as these components handle various network and control functions required for QKD integration within the TeraFlowSDN framework.

Step 2: Simulating QKD Nodes

Objective

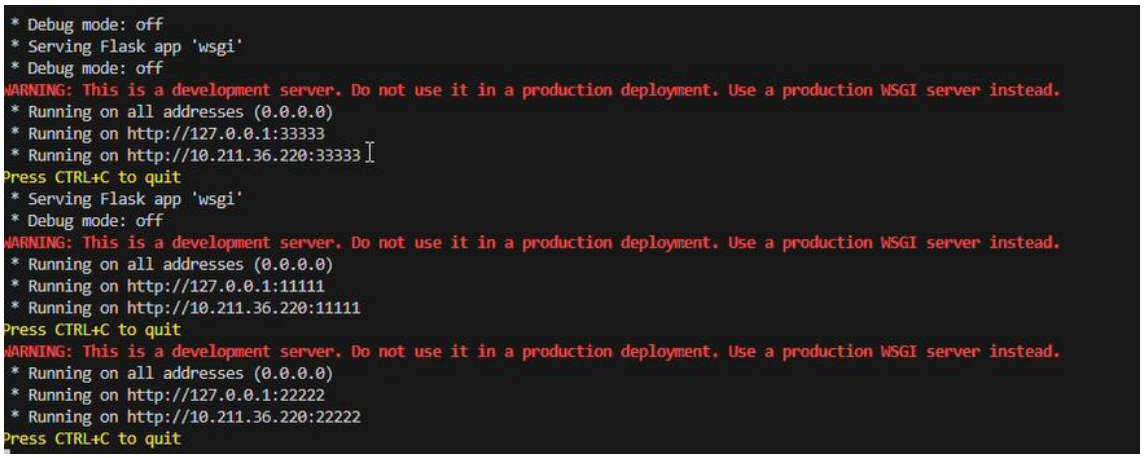
If actual QKD nodes are unavailable, a simulation of QKD nodes is necessary for testing and demonstration purposes.

1. **Run the QKD Node Simulation Script**:

```
src/tests/tools/mock_qkd_nodes/start.sh
```

This script creates three virtual nodes: QKD1, QKD2, and QKD3, each with unique IP addresses and designated ports. The simulation will allow these nodes to be

visualized and managed within the WebUI.



Explanation

Simulating QKD nodes enables a controlled environment where QKD behaviors and interactions can be tested. This setup reflects realistic node characteristics, providing insights into how actual QKD devices might interact within a secure SDN framework.

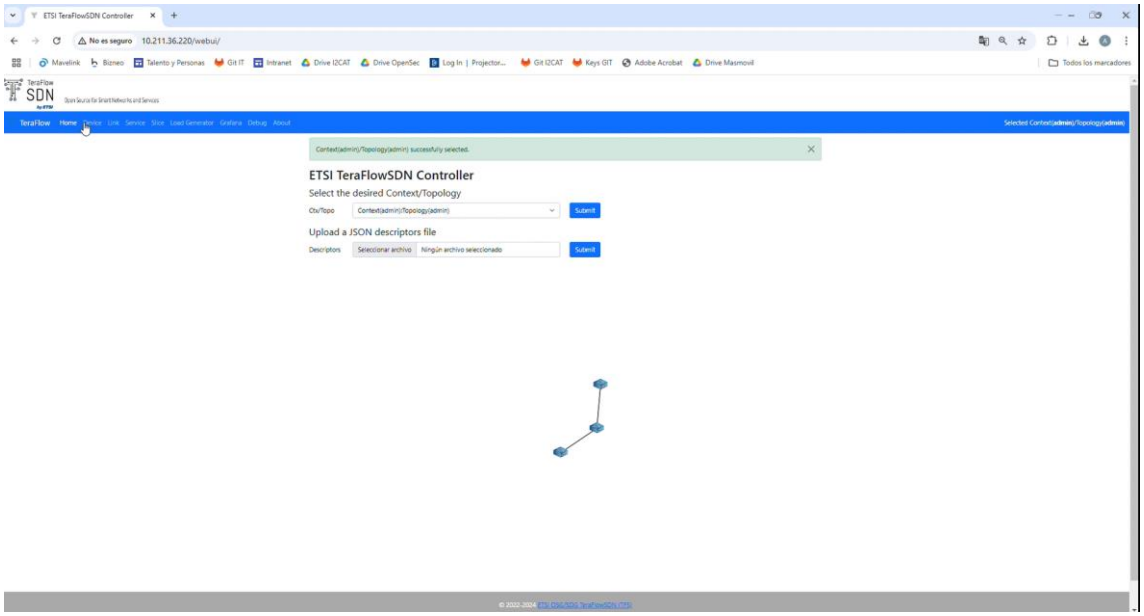
Step 3: Uploading the File Descriptor

Objective

The next step is to define the QKD network topology by uploading a descriptor file. This file provides essential information about node configurations and link definitions.

- **File Descriptor Details**:
    - **Device Configuration**: Each QKD node (QKD1, QKD2, QKD3) is defined with its IP, port, and configuration settings.
    - **Link Configuration**:
        - The descriptor specifies bidirectional links between QKD1 and QKD2, as well as QKD2 and QKD3.
        - Each link includes link_uuid identifiers and endpoint_uuid values to establish node connectivity. In this mock setup:
            - 1 refers to QKD1, 2 to QKD2, and 3 to QKD3.
            - Port positioning indicates the directionality of each connection.

Once uploaded, the WebUI will display this topology within the Admin Context, providing a visual overview of nodes and links.



Explanation

Uploading the descriptor enables the TeraFlowSDN controller to recognize and manage the network topology, defining the physical and logical relationships between QKD nodes. Accurate configurations here ensure the nodes and links behave as expected within the SDN environment.

---

4. Step 3: Uploading the Network Topology Descriptor

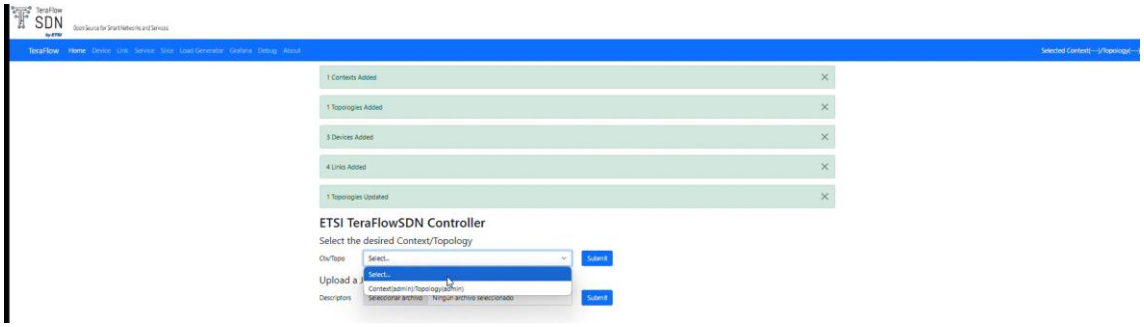
Objective

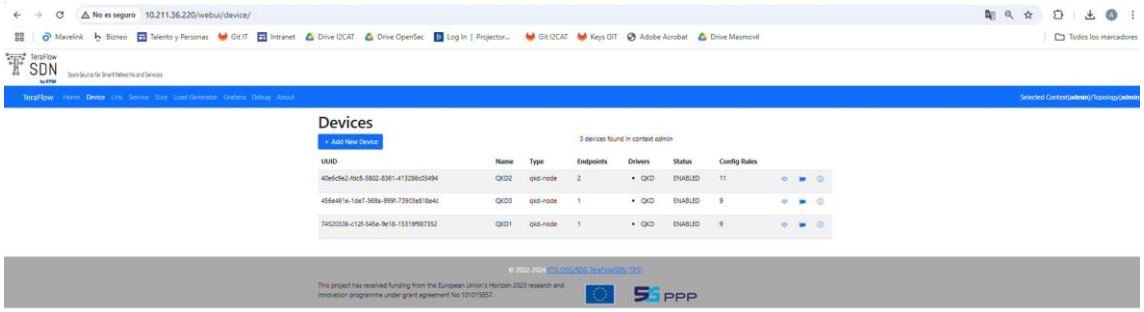
The next step is to define the QKD network topology by uploading a descriptor file. This file provides essential information about node configurations and link definitions.

- **File Descriptor Details**:
    - **Device Configuration**: Each QKD node (QKD1, QKD2, QKD3) is defined with its IP, port, and configuration settings.
    - **Link Configuration**:
        - The descriptor specifies bidirectional links between QKD1 and QKD2, as well as QKD2 and QKD3.
        - Each link includes link_uuid identifiers and endpoint_uuid values to establish node connectivity. In this mock setup:
            - 1 refers to QKD1, 2 to QKD2, and 3 to QKD3.

- Port positioning indicates the directionality of each connection.

Once uploaded, the WebUI will display this topology within the Admin Context, providing a visual overview of nodes and links.





Explanation

Uploading the descriptor enables the TeraFlowSDN controller to recognize and manage the network topology, defining the physical and logical relationships between QKD nodes. Accurate configurations here ensure the nodes and links behave as expected within the SDN environment.

---

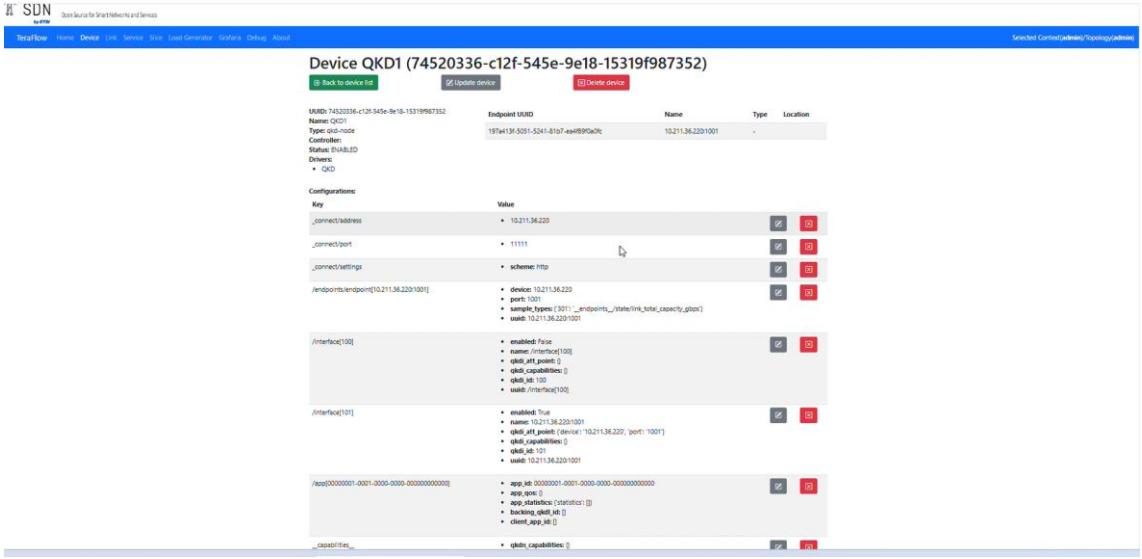
5. Step 4: Exploring the Devices Section

Objective

With the topology loaded, the Devices section in the WebUI allows for an in-depth look at each QKD node's configuration and operational status.

- **Device Information Displayed**:
    - **Status, IP, Port, and Driver Information**: Each node shows its operational state, network address, and driver type.
    - **Detailed Node Configuration**: QKD2, being central with connections to both QKD1 and QKD3, has additional configuration rules to manage traffic from multiple nodes.

- **View Device Details**:
    - Select QKD1 to inspect specific settings such as IP addresses and ports, crucial for QKD1's intermediary role in the network.



Explanation

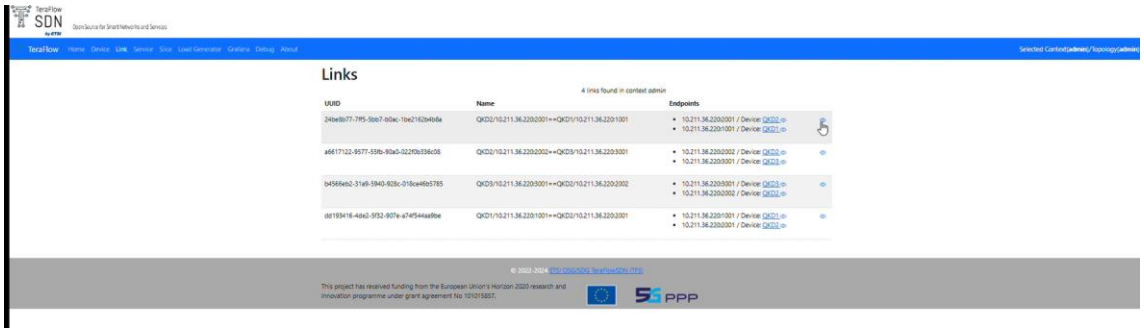
The Devices section provides real-time visibility into each QKD node's configuration, which is essential for managing and troubleshooting network behavior, particularly in a multi-node quantum communication setup.

Step 5: Exploring the Links Section

Objective

The Links section displays established connections, allowing you to differentiate between physical and virtual connections.

- **View Physical Links**:
    - Links between QKD1 ↔ QKD2 and QKD2 ↔ QKD3 are displayed, with endpoints and connection details.
- **Inspect Link Details**:
    - Select the link between QKD1 and QKD2 to view specific endpoints, associated devices, and connection ports.



Explanation

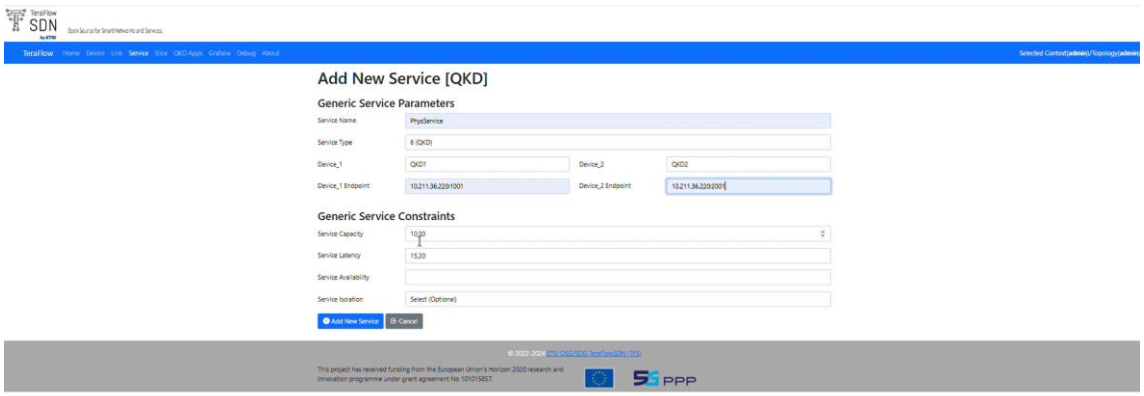
By providing detailed link information, the Links section allows for an assessment of physical and virtual connectivity, enabling administrators to validate network structure and troubleshoot potential issues.

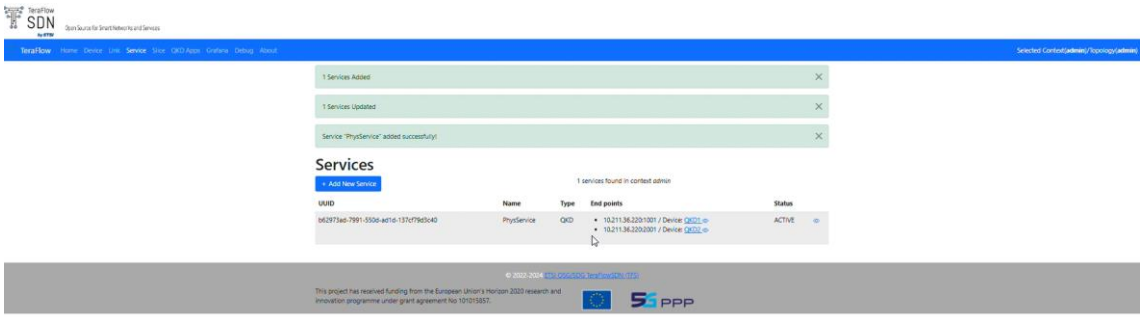
Step 6: Creating a Physical Service

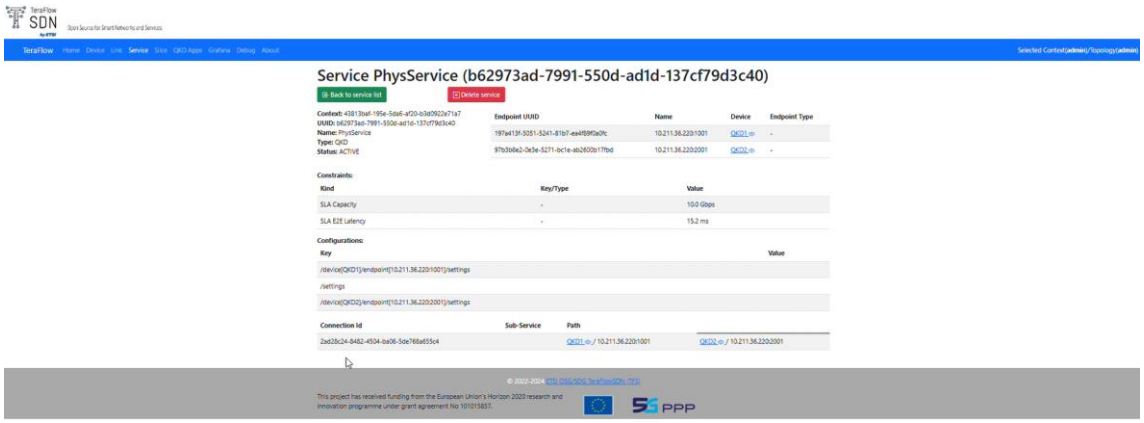
Objective

To enable QKD services, begin by creating a physical service between adjacent nodes, such as QKD1 and QKD2.

1. **Service Creation Process**:
    - Enter a service name, select QKD1 and QKD2, and specify endpoints.
    - After creation, the service will appear in the list, and associated rules will display under Devices.







Explanation

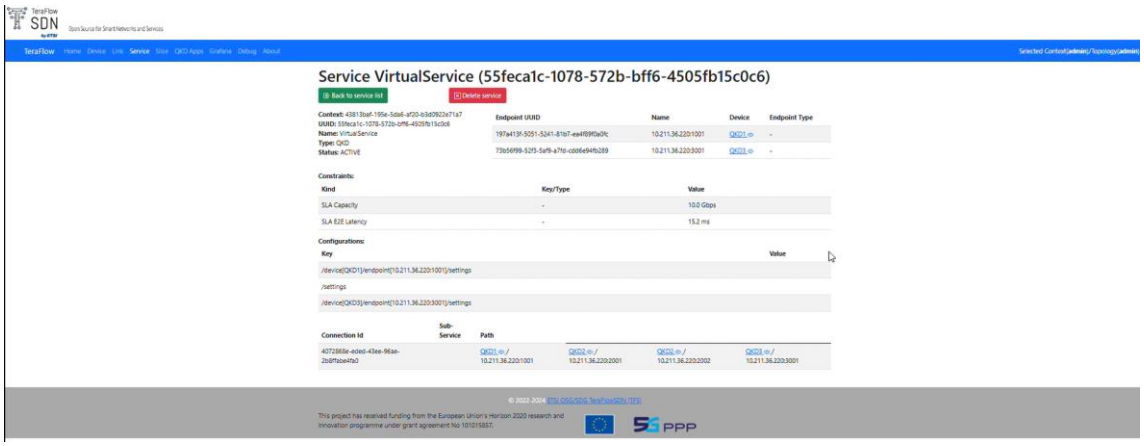
A physical service in TeraFlowSDN defines a direct, secured channel for key exchange between two nodes. This functionality is foundational for establishing secure SDN-enabled QKD networks.

Step 7: Creating a Virtual Service

Objective

For nodes without a direct link, create a virtual service, such as between QKD1 and QKD3, with QKD2 as the intermediary.

1. **Virtual Service Configuration**:
    - Define QKD1 and QKD3 with routing through QKD2.
    - After creation, QKD2 will have new configuration rules managing the virtual service.



Explanation

Virtual services extend the network, enabling secure communication across non-adjacent nodes. By routing through intermediary nodes, TeraFlowSDN ensures QKD1 and QKD3 maintain a secure key distribution path, even without a direct link.

Step 8: Exploring the Applications Section

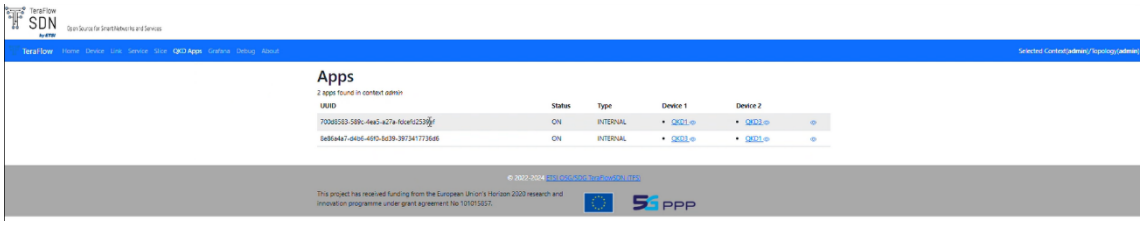
Objective

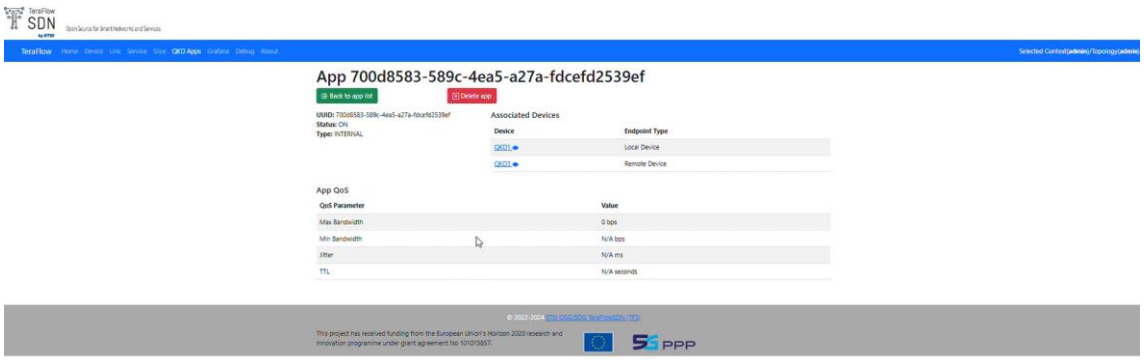
The Applications section of the TeraFlowSDN WebUI is where users manage both internal and external QKD applications. This step explains how internal applications are automatically generated when a virtual service is created, and how to manually configure external applications that enable secure connections between QKD nodes and external entities.

Internal Applications: Auto-Generated for Virtual Services

When a virtual service is set up to connect non-adjacent nodes (e.g., QKD1 and QKD3 through QKD2), TeraFlowSDN automatically creates internal applications to ensure secure communication along each part of the path. These applications:

- **Enable Multi-hop Communication**: Each internal application represents a part of the multi-hop path needed to connect distant nodes securely.
- **Maintain Encryption and Routing Rules**: Internal applications enforce the necessary policies for key distribution over virtual links.
- **Create Bidirectional Paths**: Two internal applications are created for each direction (e.g., QKD1 to QKD2, and QKD2 to QKD3), ensuring a reliable bidirectional communication channel.





Key Parameters for Internal Applications

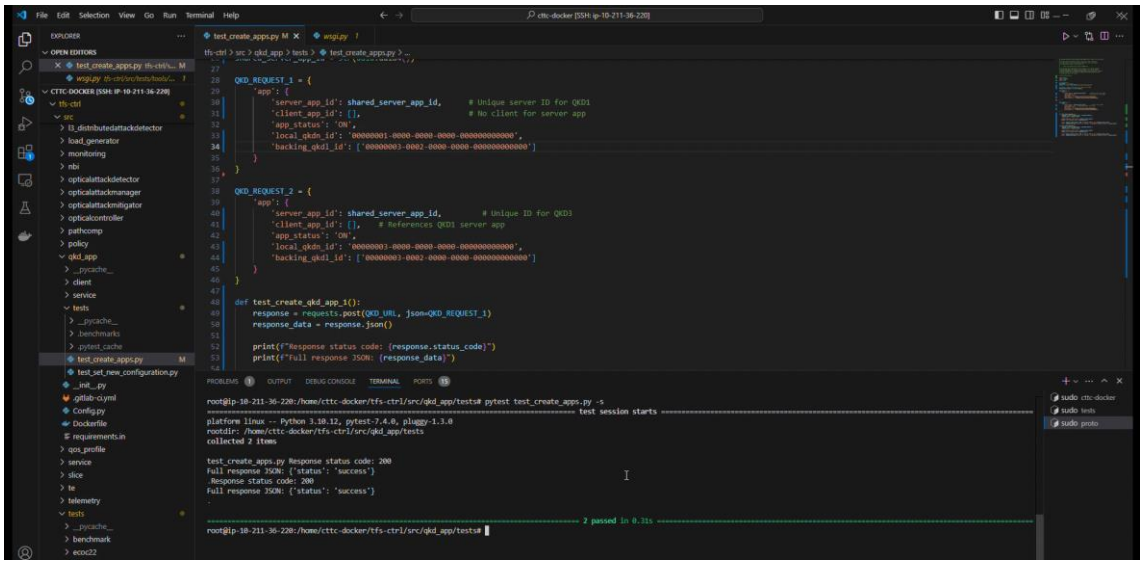
Internal applications generated by the system follow a consistent configuration, which can be reviewed in the WebUI:

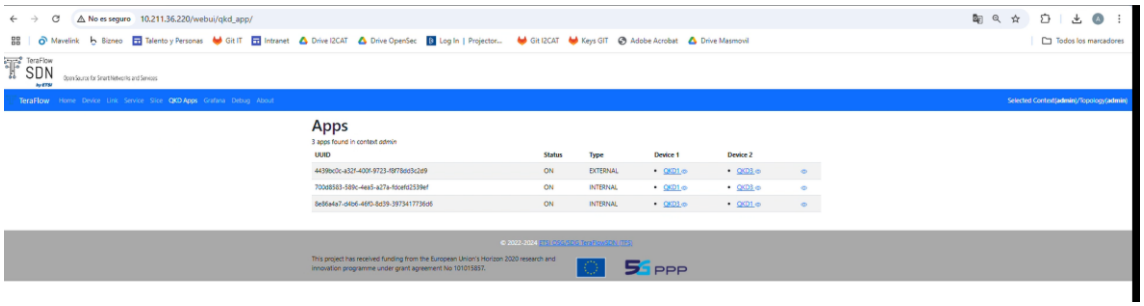
1. **app_status**: Each application is set to “ON,” indicating it is actively maintaining secure communication.
2. **local_qkdn_id**: This identifies the node where the application is hosted (e.g., QKD1, QKD2).
3. **backing_qkdl_id**: This specifies the QKD link that supports the application, ensuring secure communication between nodes.

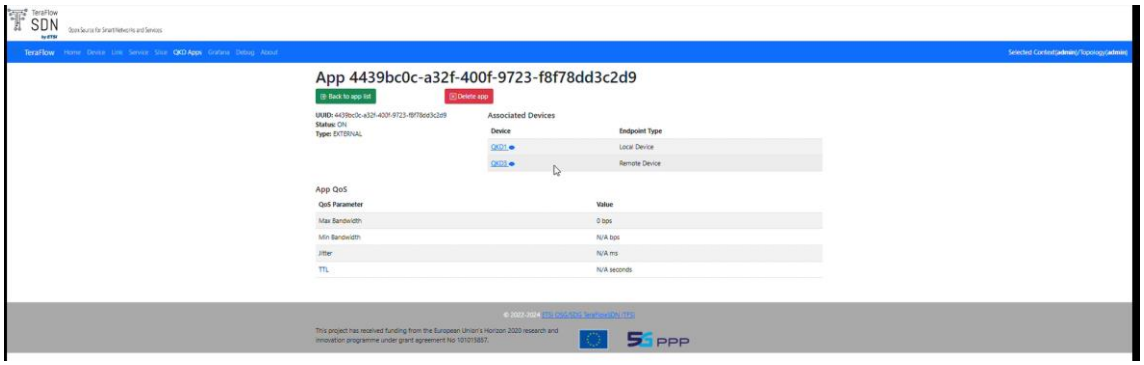
These internal applications can be seen in the Applications section, providing visibility and control over each part of the virtual service path.

External Applications: Configured for Secure Key Exchanges

External applications enable secure key exchanges between QKD nodes or external entities that require a secure quantum connection. These applications need to be configured manually and link two QKD nodes, designated as Local Device (server) and Remote Device (client).







Key Parameters in External Application Setup

To configure an external QKD application in TeraFlowSDN, specific parameters must be defined. Here is a breakdown of the most critical parameters used in the demo:

1. **server_app_id**:
    - A unique identifier shared by both nodes to link the applications on the Local and Remote Devices as part of one external QKD application.
    - In this setup, server_app_id links QKD1 (server) and QKD3 (client), creating a continuous secure channel.

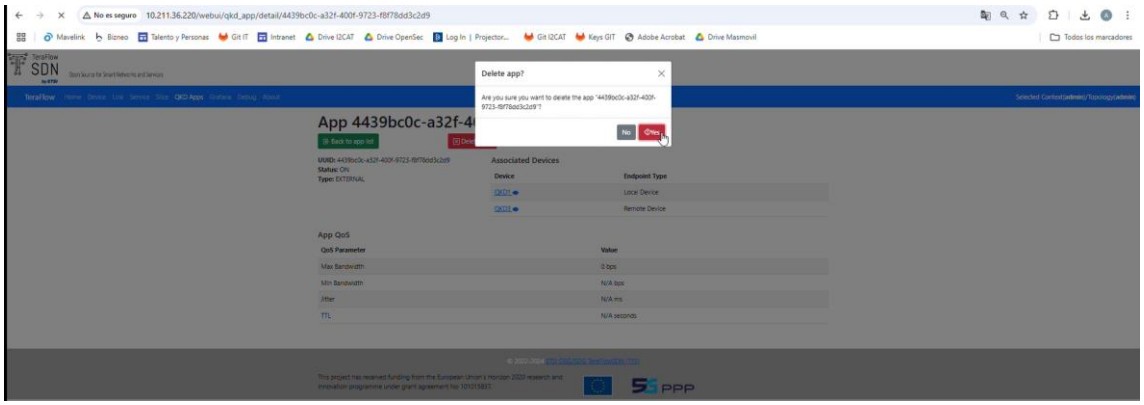
2. **client_app_id**:
    - Set to an empty list ([]) for standalone applications. The client node (e.g., QKD3) can reference the server_app_id of the main node if additional client applications are required.
3. **app_status**:
    - Set to “ON” for both QKD1 and QKD3, indicating that each application is active and prepared for secure data exchange.
4. **local_qkdn_id**:
    - Specifies the unique node ID where each application is hosted. For instance:
        - QKD1 uses local_qkdn_id: '00000001-0000-0000-0000-000000000000'.
        - QKD3 uses local_qkdn_id: '00000003-0000-0000-0000-000000000000'.
5. **backing_qkdl_id**:
    - Identifies the QKD link supporting the application, securing the communication channel. In this setup, both nodes use the same backing_qkdl_id: ['00000003-0002-0000-0000-000000000000'], representing the secure link between QKD1 and QKD3.

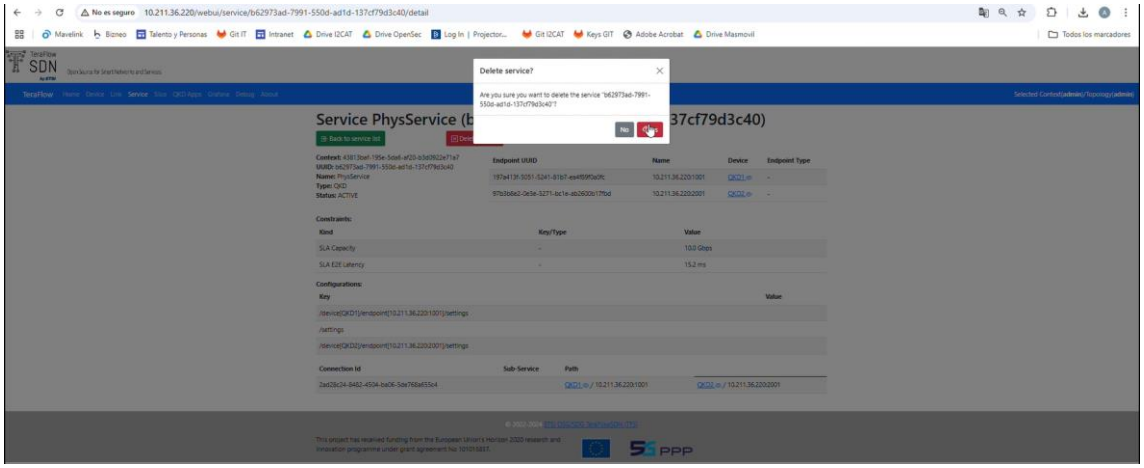
Step 9: Deleting Apps and Services

Objective

The WebUI allows deletion of services and applications, with future releases set to improve automatic status updates for dependent apps.

- **Internal App Management**: Internal apps can be deleted independently but should ideally update status in response to service changes.





Explanation

The ability to delete and manage services independently provides flexibility, though further automation will streamline application and service dependencies.

Conclusion

his guide, paired with demonstration videos, provides a comprehensive overview of setting up and managing QKD nodes and services within the TeraFlowSDN WebUI. By leveraging these tools, users can establish, monitor, and manage secure quantum communications in an SDN environment.

Thank you for following along!