OptiFlow Orchestrator – Foundational Implementation Manual

# 1. Developer Portal Implementation

\*\*Purpose:\*\*

The Developer Portal is a web-based interface allowing users to manage core OptiFlow resources such as:

- Nodes

- Workloads

- Events

- MPODs

- Monitoring data

\*\*Tech Stack:\*\*

- FastAPI (Backend)

- Jinja2 or React (Frontend – future enhancement)

- SQLite3 (Primary database)

- Redis (Caching layer)

- Uvicorn (ASGI Server)

\*\*Features:\*\*

- /nodes/ – Register & manage nodes

- /workloads/ – Add workloads with status tracking

- /events/ – Log events with timestamp and linkage to workload/MPOD

- /mpods/ – Manage multi-pod deployments

- /status/ – View cluster health

# 2. SQLite3 Database Implementation

\*\*Purpose:\*\*

SQLite3 is used as the primary lightweight embedded database to persist orchestrator resources.

\*\*Tables Defined (Refer to Excel sheet "") – Summary:\*\*

<<< Add table here >>>

\*\*Sample Initialization (Python):\*\*

```python

import sqlite3

conn = sqlite3.connect('optiflow.db')

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS nodes (

id INTEGER PRIMARY KEY,

name TEXT,

ip\_address TEXT,

status TEXT

);

""")

# Continue for other tables

conn.commit()

```

# 3. Redis Server for Caching

\*\*Purpose:\*\*

Redis enhances performance by caching frequently accessed data.

\*\*Setup:\*\*

```bash

sudo apt install redis-server

sudo systemctl enable redis

sudo systemctl start redis

```

\*\*Python Integration:\*\*

```python

import redis

cache = redis.Redis(host='localhost', port=6379, db=0)

cache.set("node\_1\_status", "Healthy")

```

# 4. etcd Distributed Key-Value Store

\*\*Purpose:\*\*

etcd is used for maintaining cluster state and key metadata.

\*\*Installation (Linux):\*\*

```bash

sudo snap install etcd

```

\*\*Configuration:\*\*

```bash

etcd --name node1 --initial-advertise-peer-urls http://localhost:2380 \

--listen-peer-urls http://localhost:2380 \

--listen-client-urls http://localhost:2379 \

--advertise-client-urls http://localhost:2379

```

\*\*Test:\*\*

```bash

etcdctl put /optiflow/nodes/node1/status Healthy

etcdctl get /optiflow/nodes/node1/status

```

# 5. OptiFlow Orchestrator Core Modules

\*\*A. Controller Manager\*\*

- Responsible for state reconciliation.

- Monitors node/workload status.

\*\*B. AI-Driven Intelligent Scheduler\*\*

- Assigns workloads based on node health and availability.

\*\*Sample:\*\*

```python

async def schedule\_workloads():

# Query DB for pending workloads

# Assign to healthy nodes using Redis + decision logic

```

\*\*C. Networking Layer\*\*

- Default: flannel for simple overlay

- Optional: calico for advanced routing

# 6. Optional Enhancements

- Monitoring (Prometheus/Grafana)

- Authentication (JWT/OAuth2)

- Testing (pytest, httpx)

- Containerization (Docker/Kubernetes)

# Summary

| Component | Status | Notes |

|---------------------|------------------|-----------------------------------------|

| Developer Portal | ✅ Implemented | FastAPI based, RESTful endpoints |

| SQLite3 DB | ✅ Implemented | Modular schema based on Excel spec |

| Redis | ✅ Implemented | Caching for performance |

| etcd | ⚙️ In progress | Planning for distributed HA |

| AI Scheduler | ✅ Completed | Integrated & tested |

| Controller Manager | ✅ Completed | Manages resource state |

| Networking | ⚙️ Configurable | Flannel default, Calico supported |