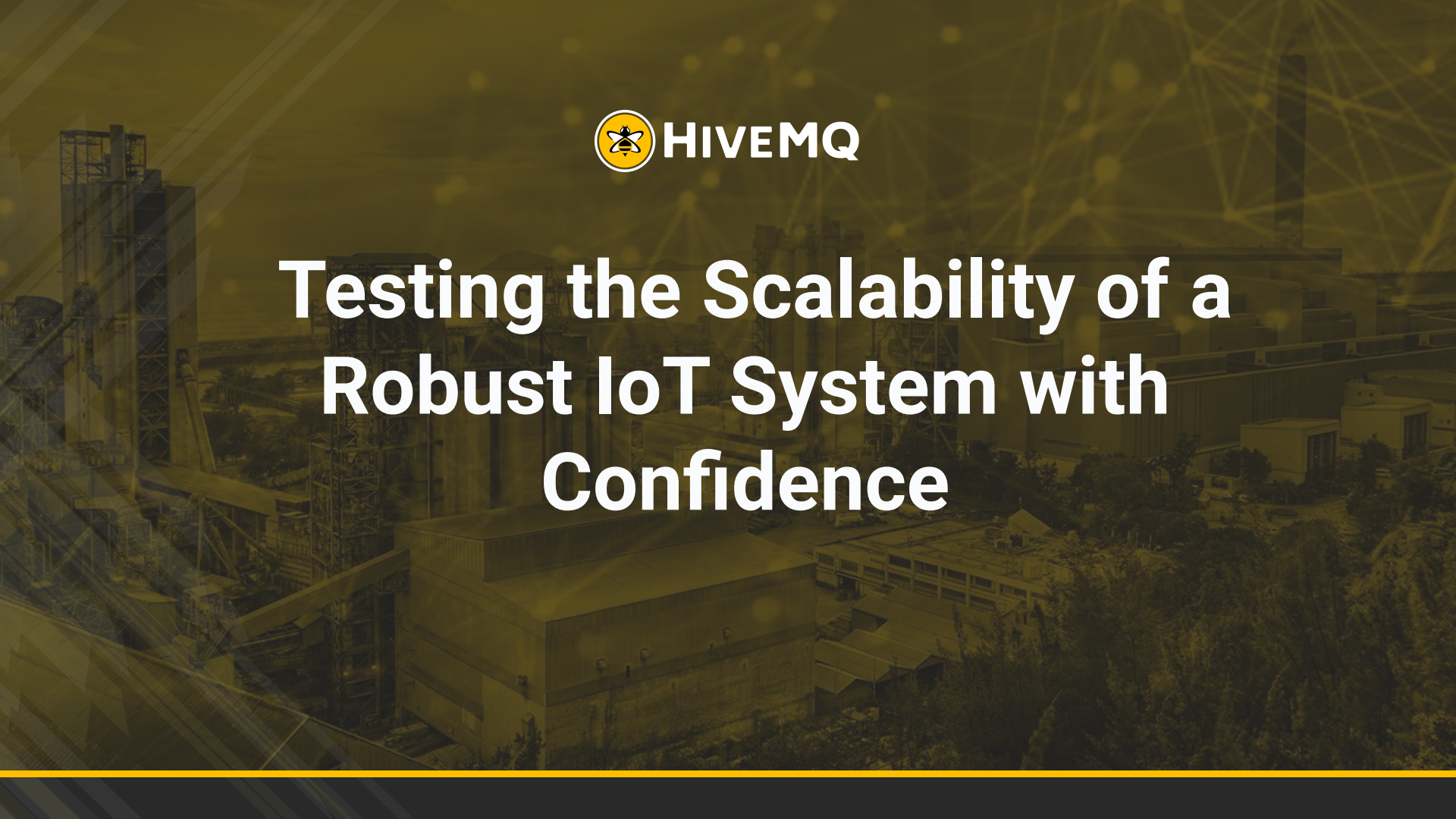
WELCOME TO

We will start this session shortly

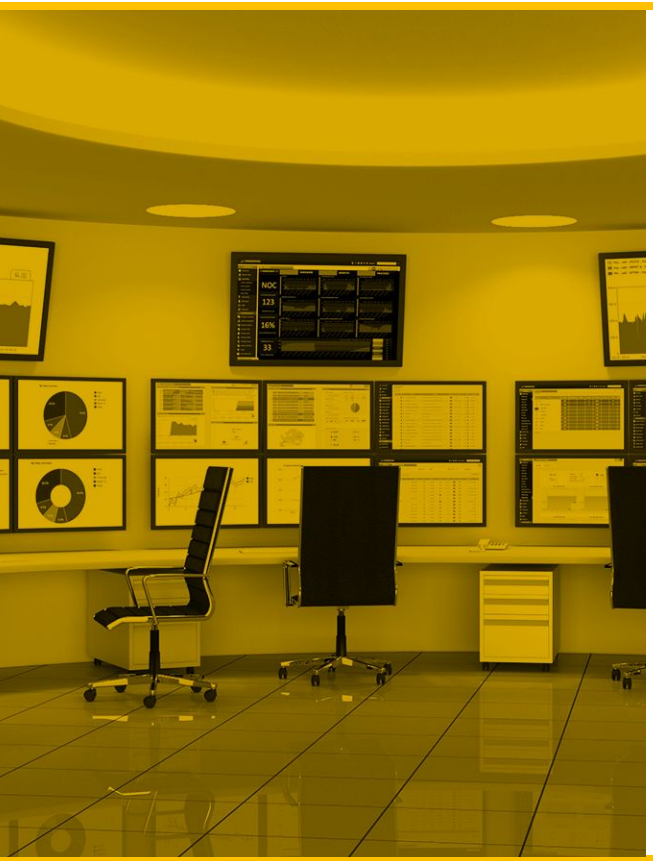

HIVEMQ
Testing the Scalability of a Robust IoT System with Confidence

Why IoT Testing is Important

Why IoT Testing is Important

Fixing IoT Production Errors are Costly to Fix in the Field

Why IoT Testing is Important



Load & Stress Testing of Complete End-to-end IoT System is Required to Determine System Resilience

Why IoT Testing is Important?

Capacity planning required to:

- Budget network and infrastructure costs
- Budget financial costing for cloud hosting

Challenges for IoT Testing

IoT systems are massive distributed systems that can be difficult to test

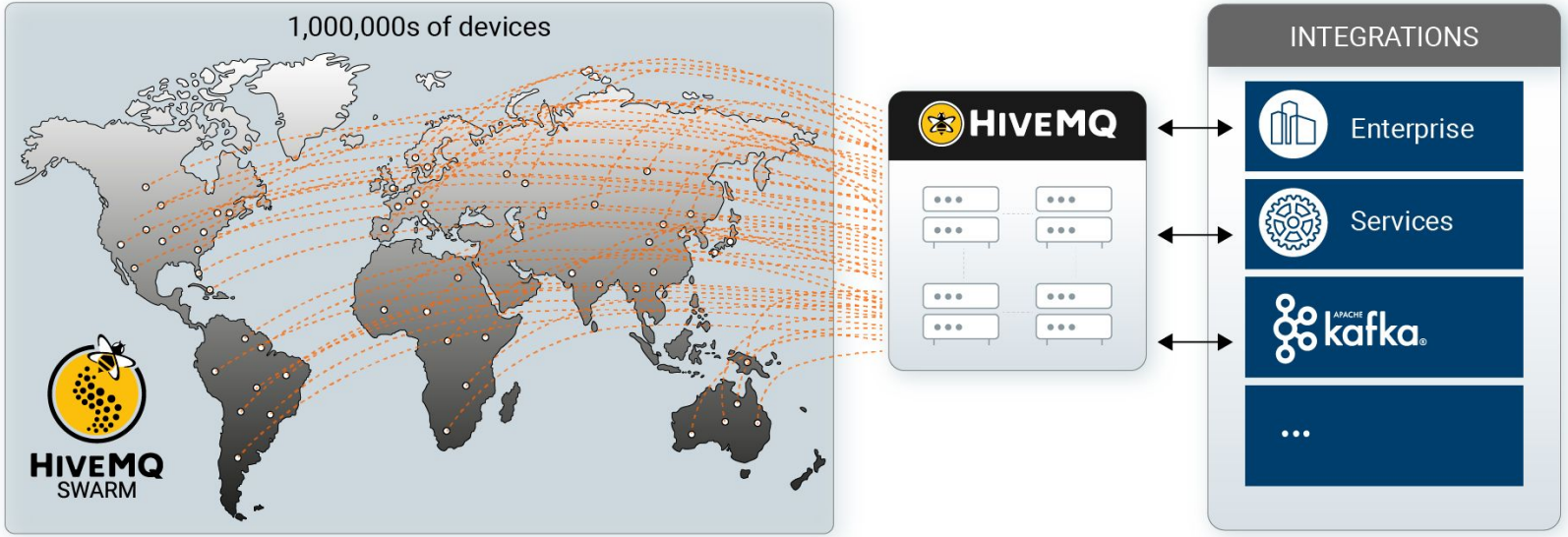
Test environment is often different from production behaviour

Individual IoT devices can have multiple complex behaviour patterns

IoT production data can have a high degree of variability

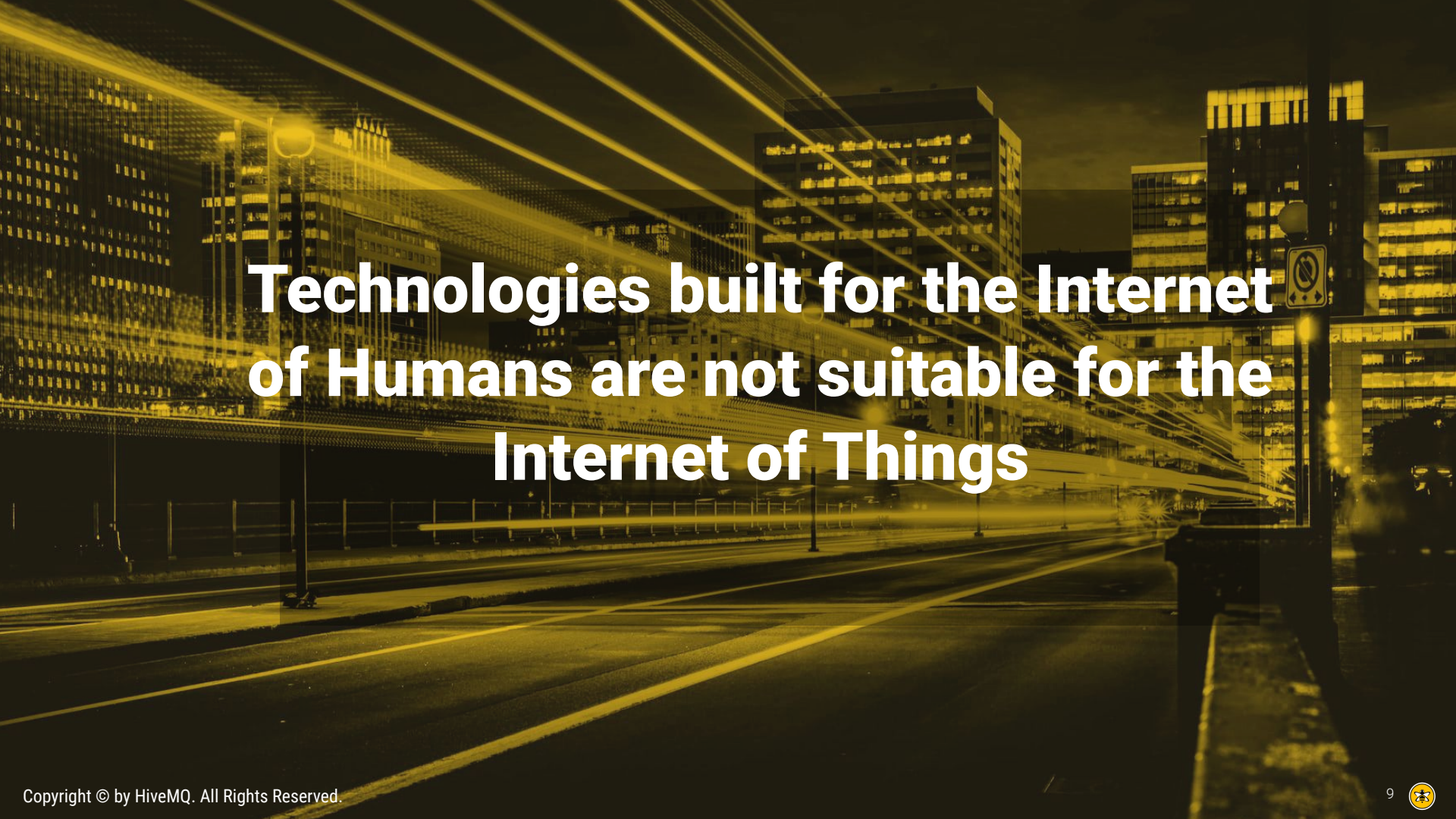
Testing at massive scale

Challenges for IoT Testing

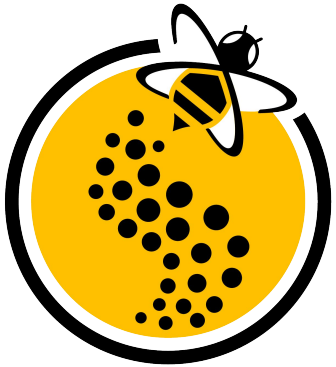


- Massive distribution
- Very hard to set up in testing / staging

- Single Systems (10s-100s)
- Easy to setup in testing /staging



Introducing HiveMQ Swarm



HIVEMQ
SWARM

- Distributed platform able to create millions of unique network connections
- Simulating millions of devices, messages and MQTT topics
- Develop reusable scenarios that simulate device behaviours
- Custom data generator that simulate complex use cases
- Resource friendly and easy deployment to public clouds (AWS, Azure, etc.) and Kubernetes

Use Cases

Load and stress testing

Capacity planning

Quality assurance testing

Test HiveMQ custom extensions

IoT Scenario Testing

Device rollout simulations

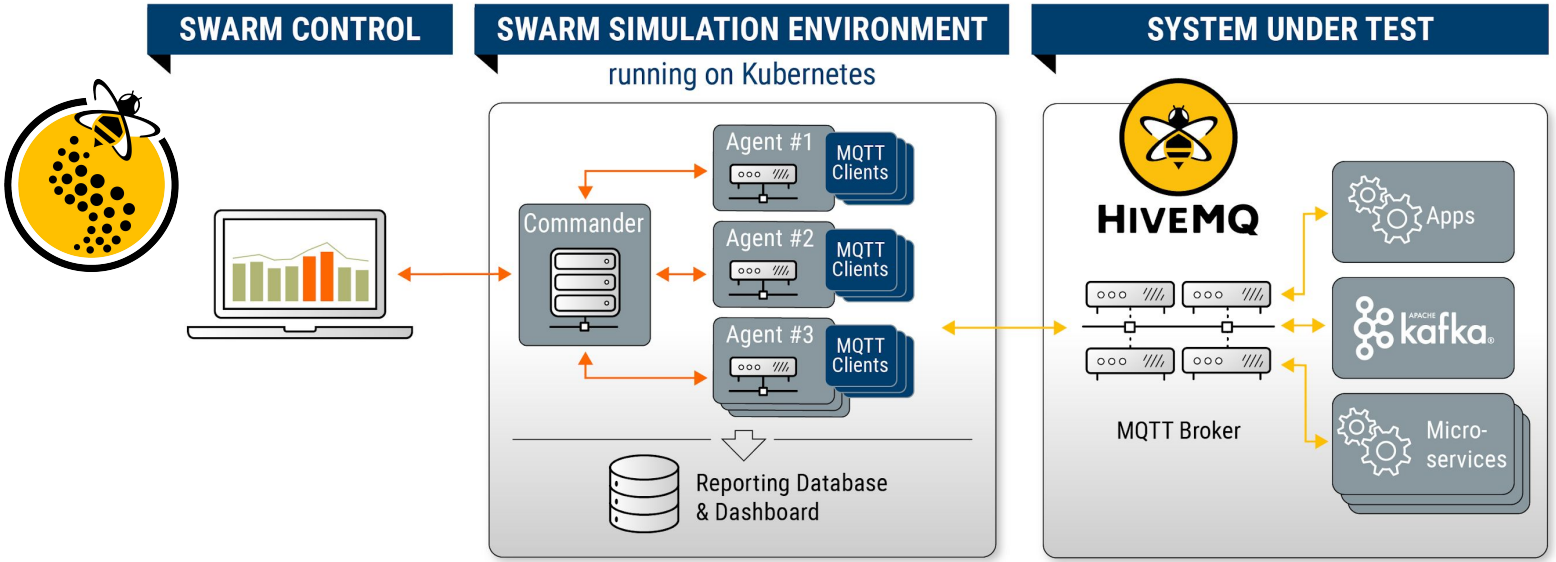
Troubleshooting

HIVEMQ
SWARM

Key Features

- Declarative and reusable scenarios
- Local and distributed setup
- Up to 10,000,000 real MQTT connections
- Built-in monitoring, logging, and reporting
- REST interface for metrics (Prometheus compatible)
- Custom data generator support (with SDK)
- Runs everywhere (Cloud, K8s, local DC, local machine)
- MQTT CLI integration

Distributed IoT Testing and Simulation

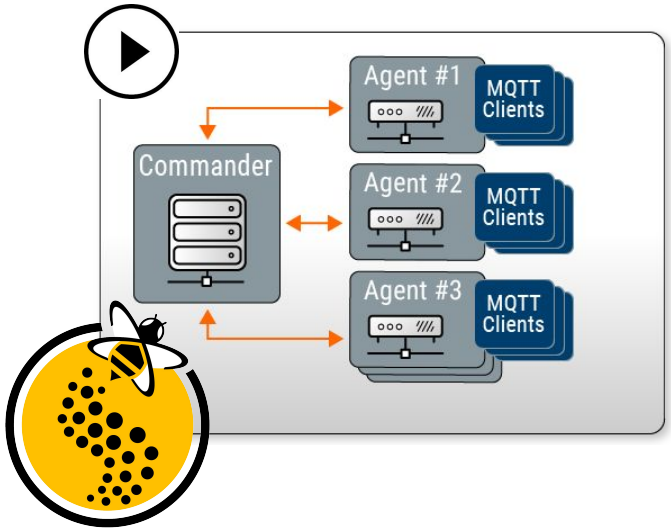


Swarm Lifecycle

1. CREATE SCENARIO

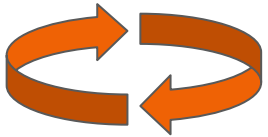
```
<xml>
<brokers>
<address></address>

....
...

</xml>
```

2. EXECUTE IN SIMULATION ENVIRONMENT



3. REPORT

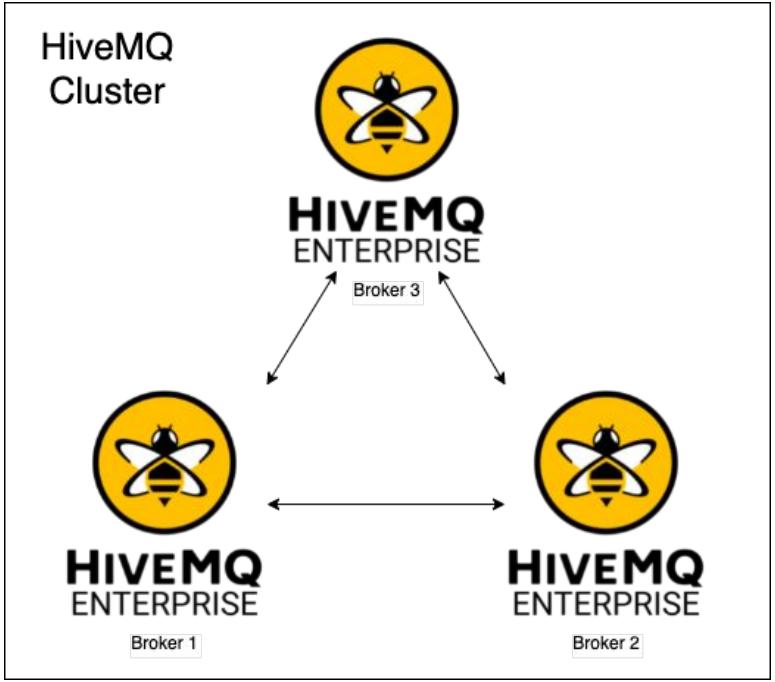
REPEAT



Demo

HiveMQ Setup

- 3 HiveMQ Nodes
- Running on Kubernetes



The Load

- 20k **Current Connections**
- 10k/min incoming **CONNECT**
- 10k/min incoming **PUBLISH**
- 10k/min outgoing **PUBLISH**

Scaling?

? **Is the Setup able to handle 2x / 4x the amount of connections, and messages without adding additional HiveMQ Nodes?**

1x

- 20k **Current Connections**
- 10k/min incoming **CONNECT**
- 10k/min incoming **PUBLISH**
- 10k/min outgoing **PUBLISH**

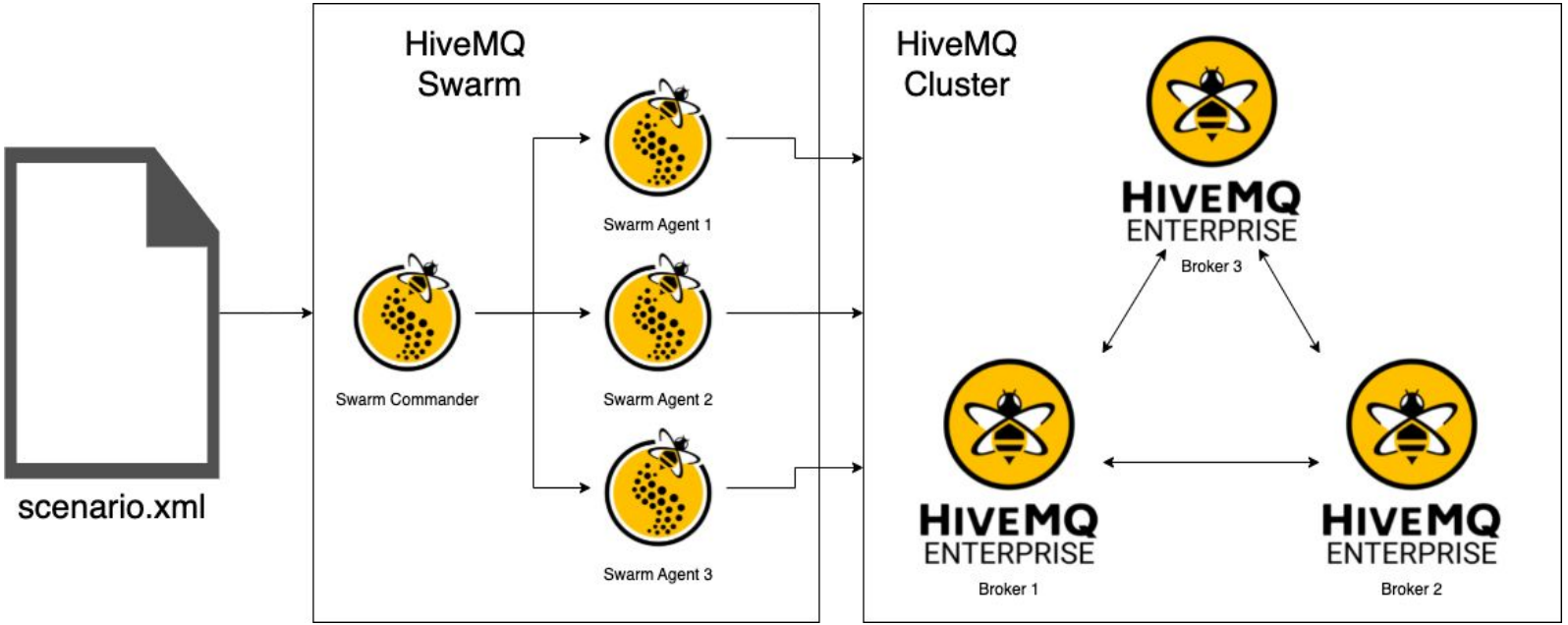
2x

- 40k **Current Connections**
- 20k/min incoming **CONNECT**
- 20k/min incoming **PUBLISH**
- 20k/min outgoing **PUBLISH**

4x

- 80k **Current Connections**
- 40k/min incoming **CONNECT**
- 40k/min incoming **PUBLISH**
- 40k/min outgoing **PUBLISH**

HiveMQ Swarm Setup



HiveMQ Swarm Clients

- 20k **Current Connections**
- 10k/min incoming **CONNECT**
- 10k/min incoming **PUBLISH**
- 10k/min outgoing **PUBLISH**

10k **Connectors**

- Connect / Disconnect every minute

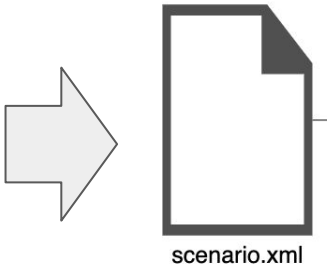
10k **Publishers**

- Connect
- Publish 1 Message every minute
- Disconnect

Conclusion

- Analyzed monitoring to create a HiveMQ Swarm Scenario

- 20k **Current Connections**
- 10k/min incoming **CONNECT**
- 10k/min incoming **PUBLISH**
- 10k/min outgoing **PUBLISH**



- Scaled up and executed the HiveMQ Swarm scenario to verify that the deployment is able to operate under higher load

ANY QUESTIONS?

THANK YOU