Lessons learnt from building Confluent Control Plane

Gwen Shapira, Principal Engineer II

Vivek Sharma, Senior Engineer

Welcome To Confluent Cloud

Create cluster

1. Select cluster type

Basic

For basic or development use cases. Upgrade to Standard at any time

Base cost: none

Uptime SLA: 99.5%

Standard

For production-ready use cases. Full feature set and standard limits

Base cost: $1.50 /hr

Uptime SLA: 99.95%

Dedicated

For use cases with high traffic or that require private networking

Price as sized

Uptime SLA: 99.95%

Size your cluster ⓘ

1 CKU / $2.66 /hr

Ingress	up to 50 MBps	Storage	up to 10,000 GB / Unlimited ⓘ
Egress	up to 150 MBps	Partitions	up to 4,500
Client connections	up to 3,000		

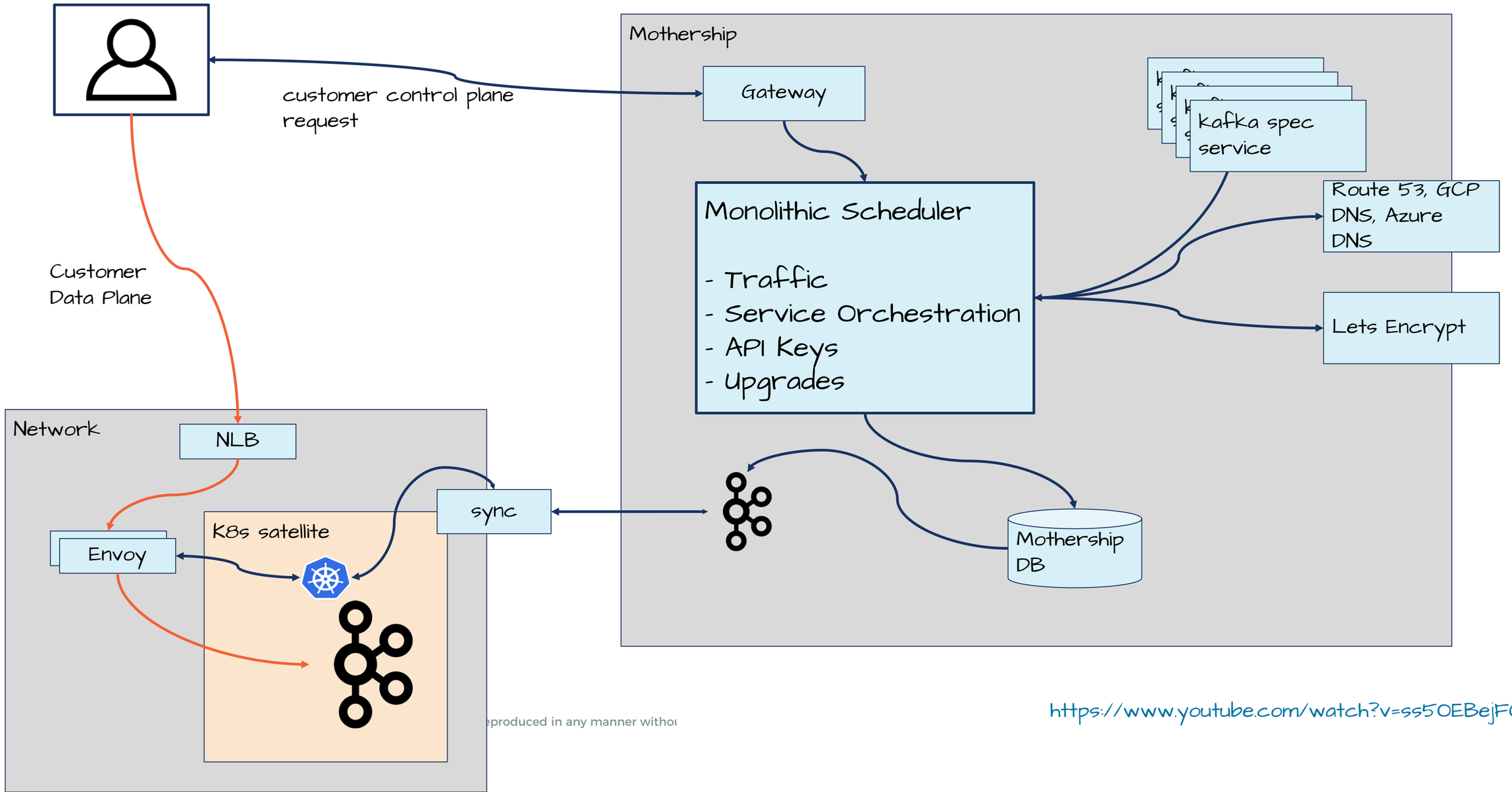
Our control plane is a challenging problem.

Because we have a lot of cloud.

- 1000s of clusters managed
- 20 teams
- 4 customer facing services
- 4 network options
- 3 constantly changing SKUs
- 3 cloud vendors
- Tight security with limited access


16 Azure regions
20 AWS regions
22 GCP regions

Confluent Cloud Control Plane

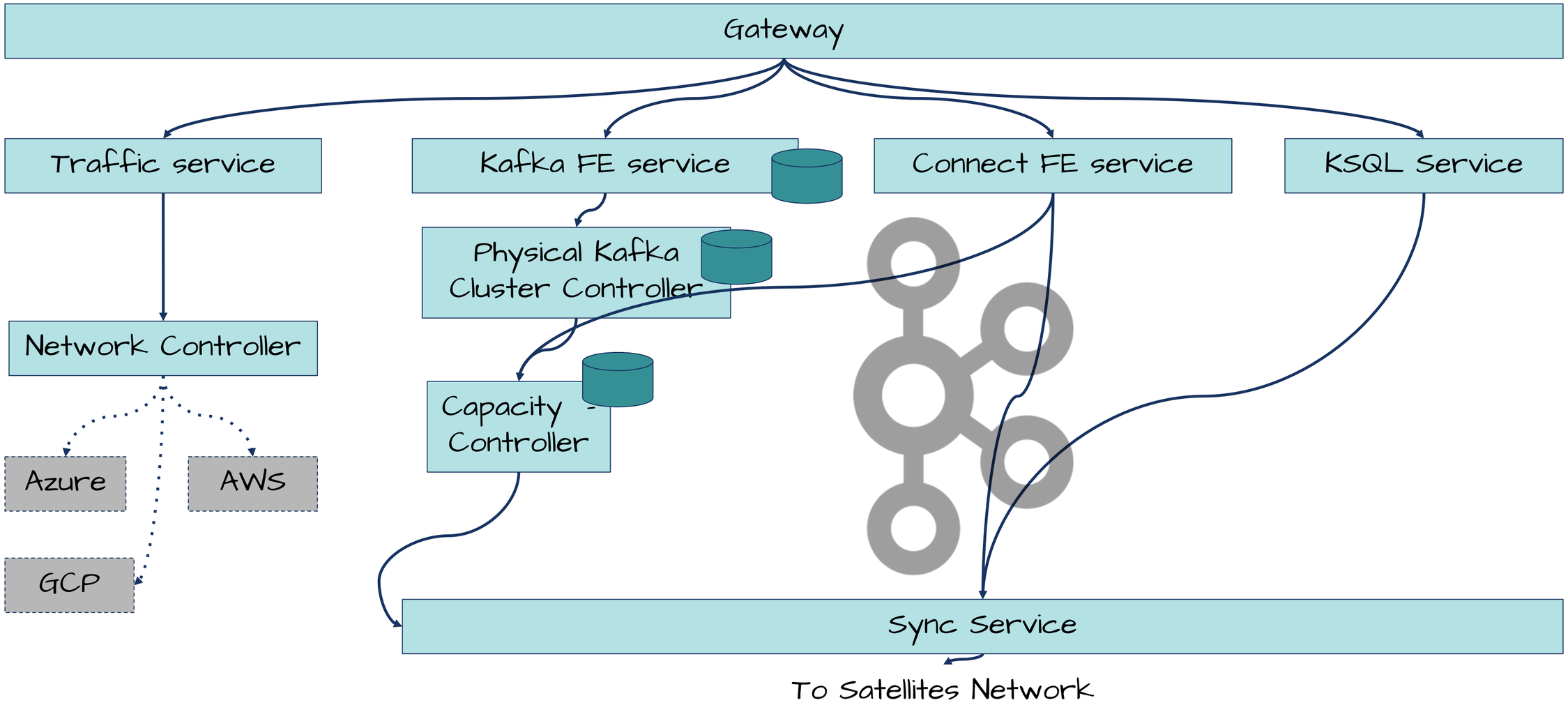


https://www.youtube.com/watch?v=ss5OEBejFCs

New requirements

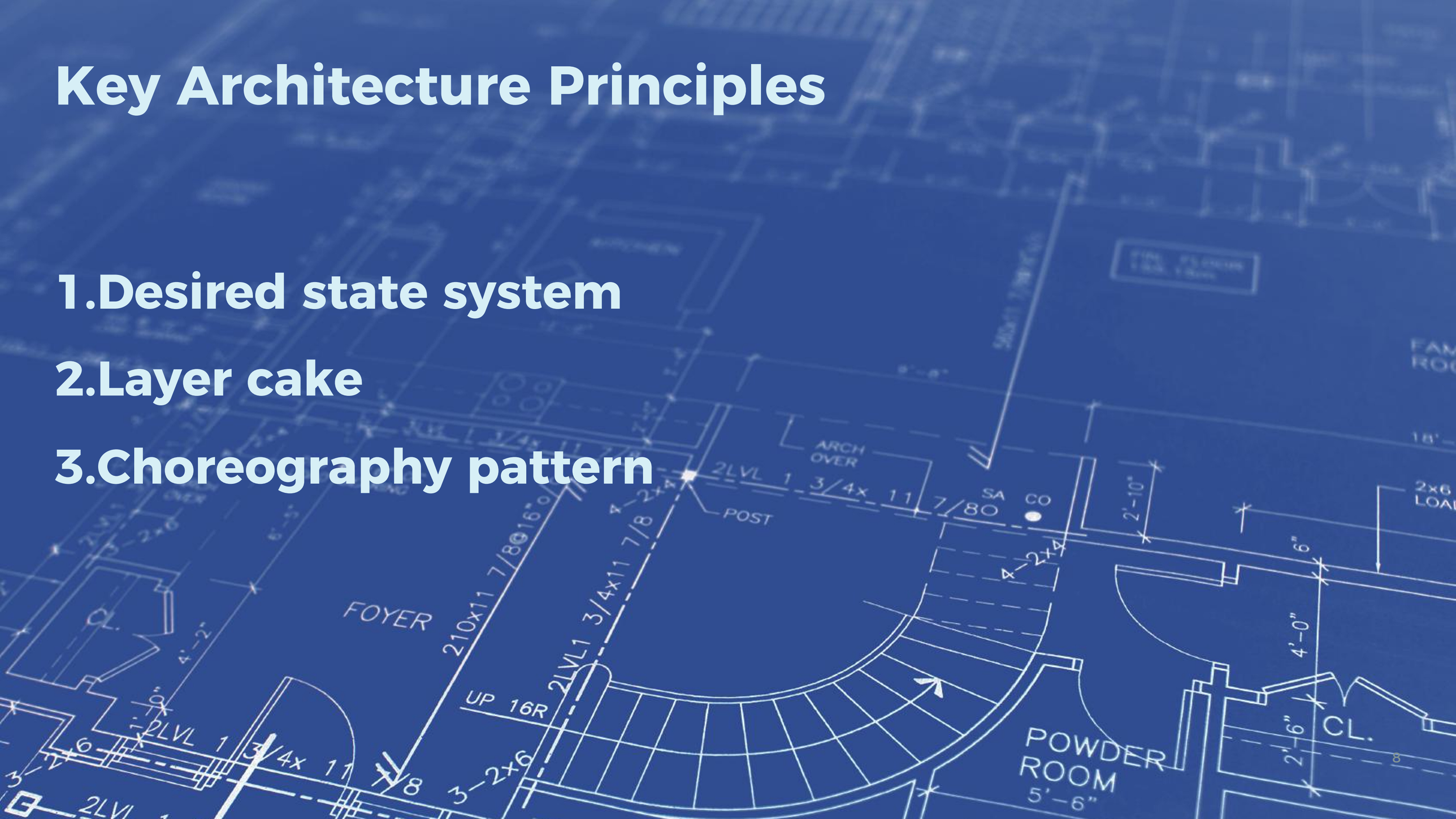
- Keep data plane independent
- Decouple resources
- Trusted source of truth
- Clean dependencies, clean interfaces
- Self heal

Architecture diagram of new state



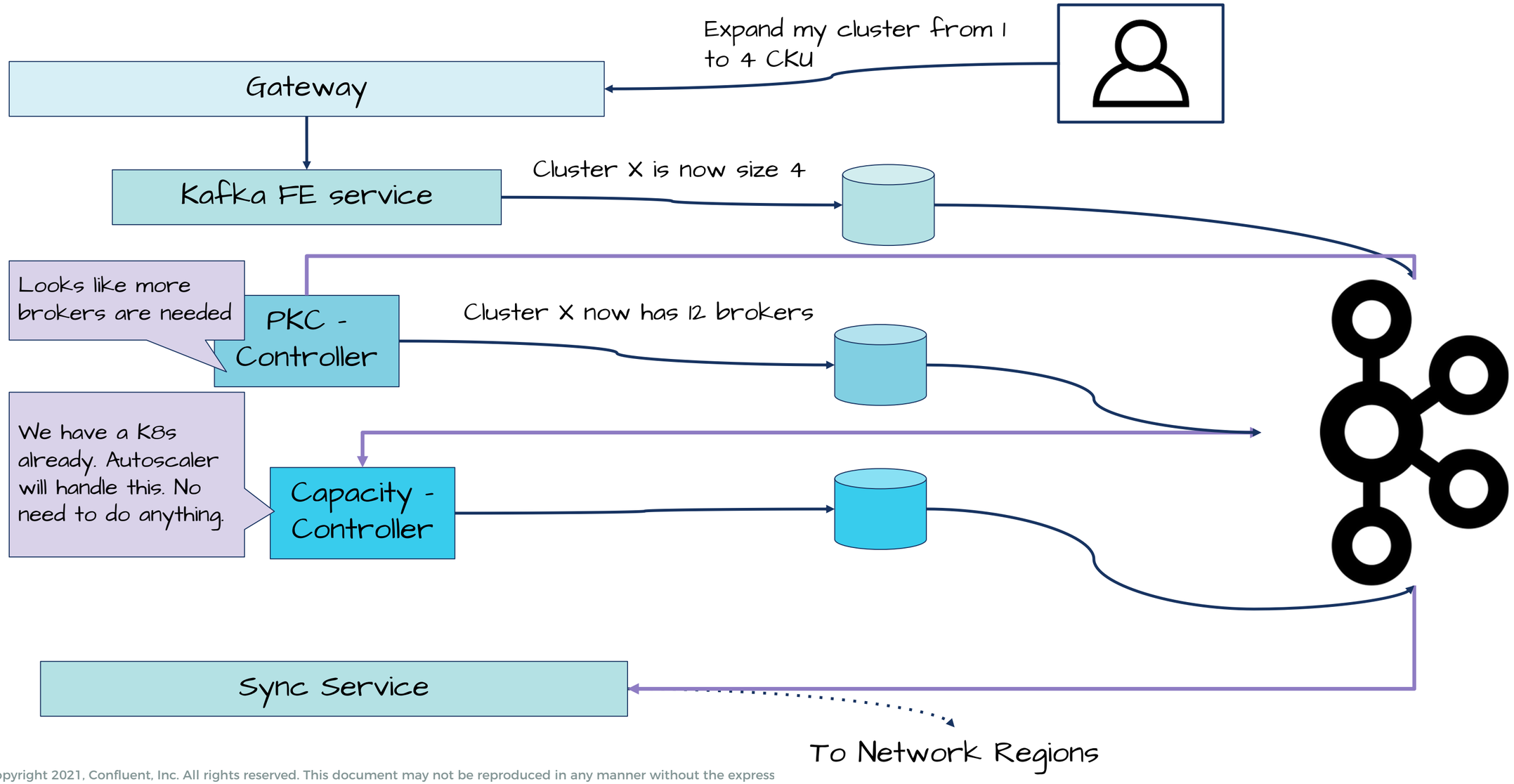
Key Architecture Principles

1. Desired state system
2. Layer cake
3. Choreography pattern



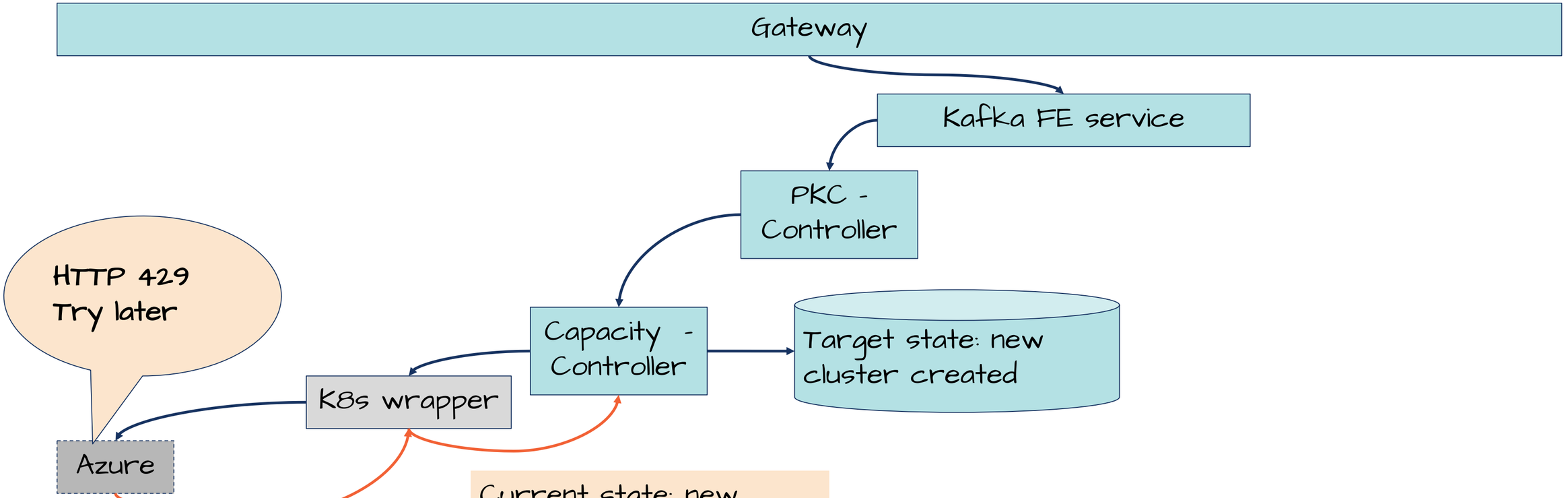
Desired state system

Persisted Desired State of Resources

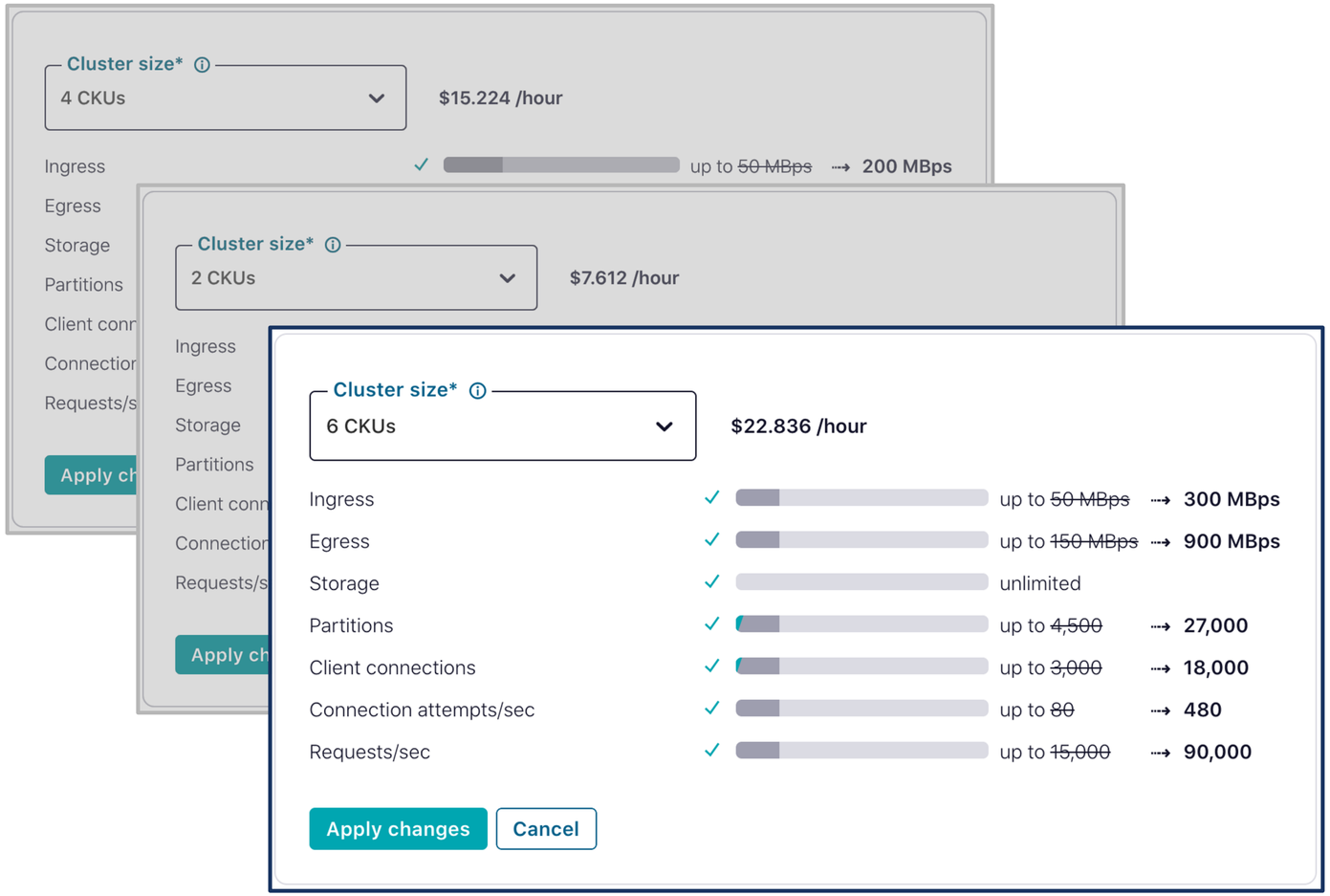


Failure scenario:

We tried to create a Kafka cluster and ran out of API calls in Azure



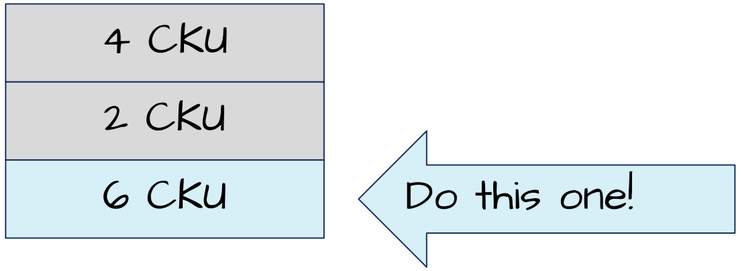
We now do a bit more...



State	Command
4 CKU	Expand by 2 CKU
2 CKU	Shrink by 2 CKU
6 CKU	Expand by 4 CKU

Idempotent operations are better fit for CDC and event streaming

We really just need to...

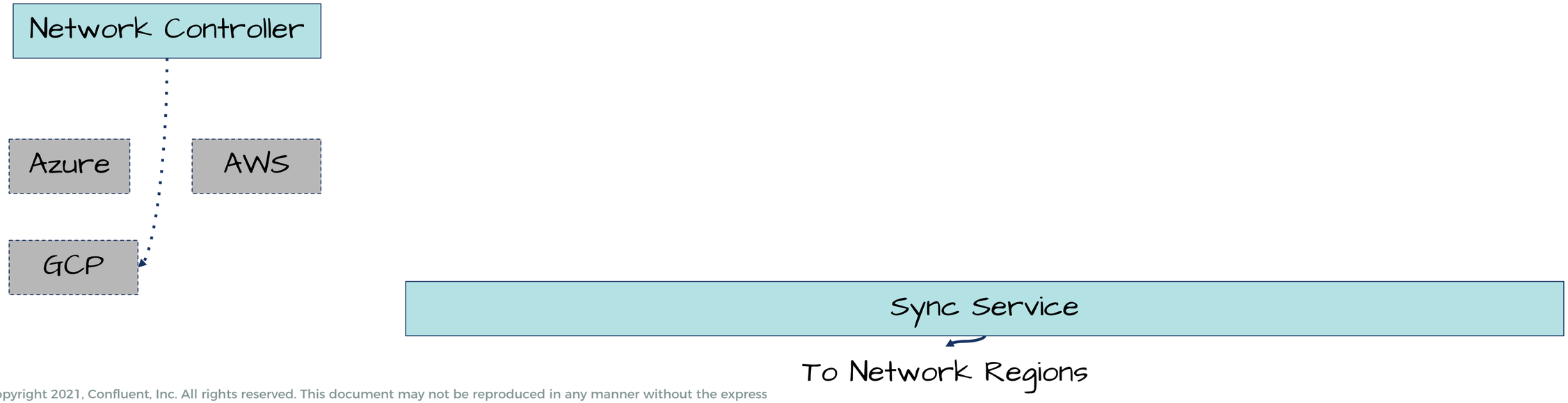


Persisted Desired State Rocks!

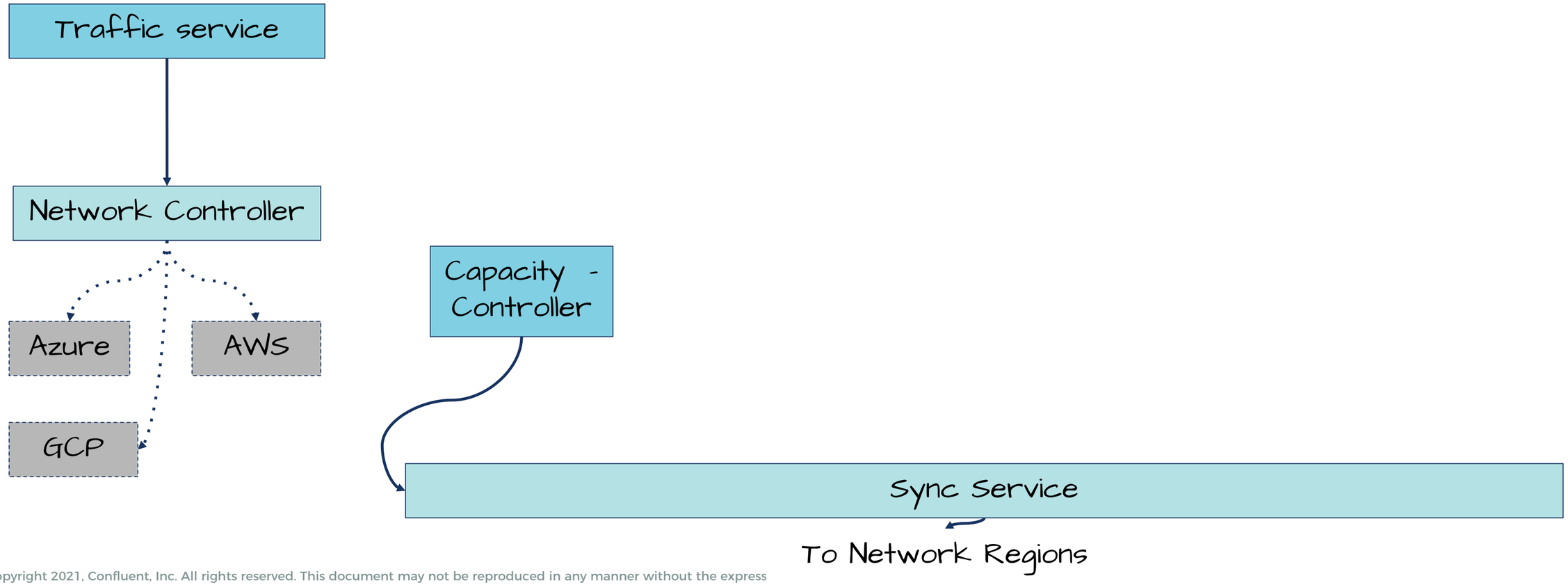
- Recoverable
- Single source of truth
- Auditable history
- Testable

Layer Cake

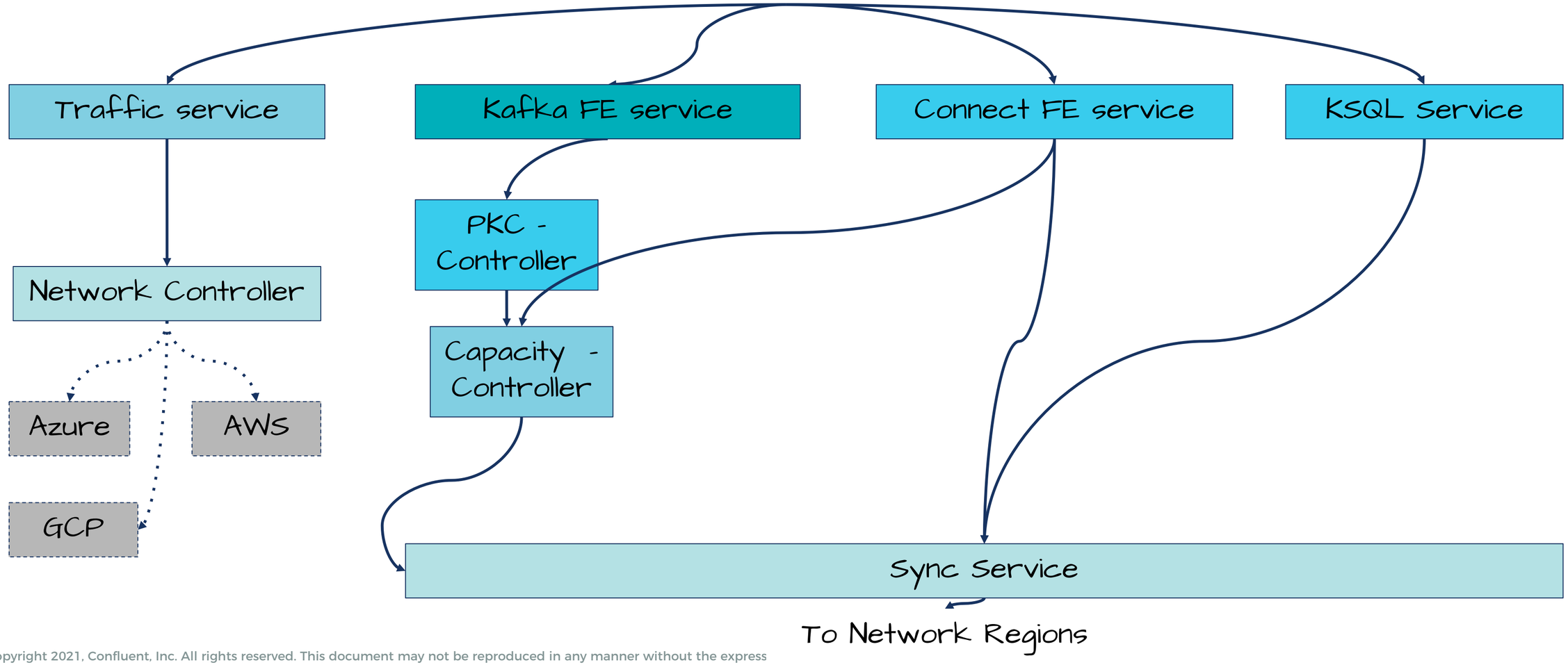
First layer - base services with no dependencies



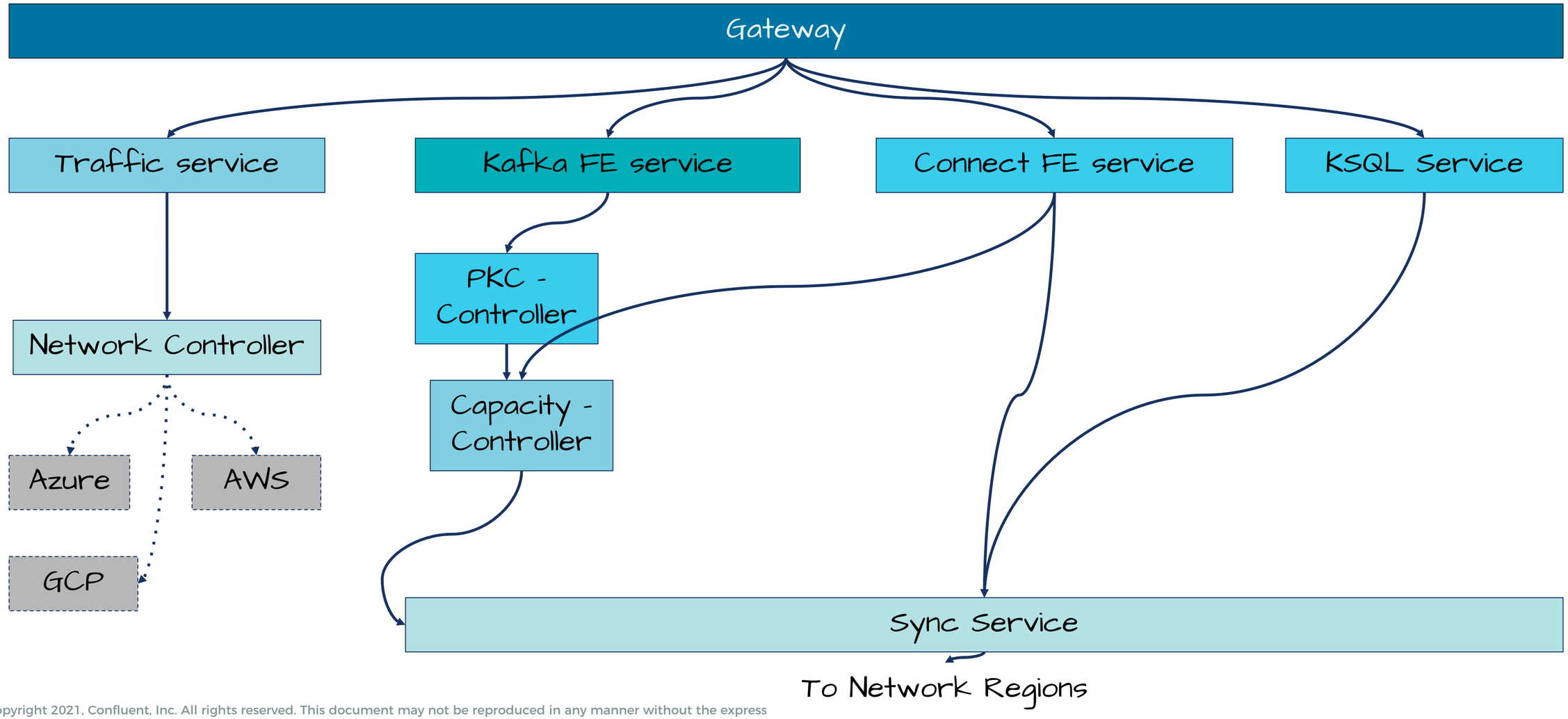
Starting second layer



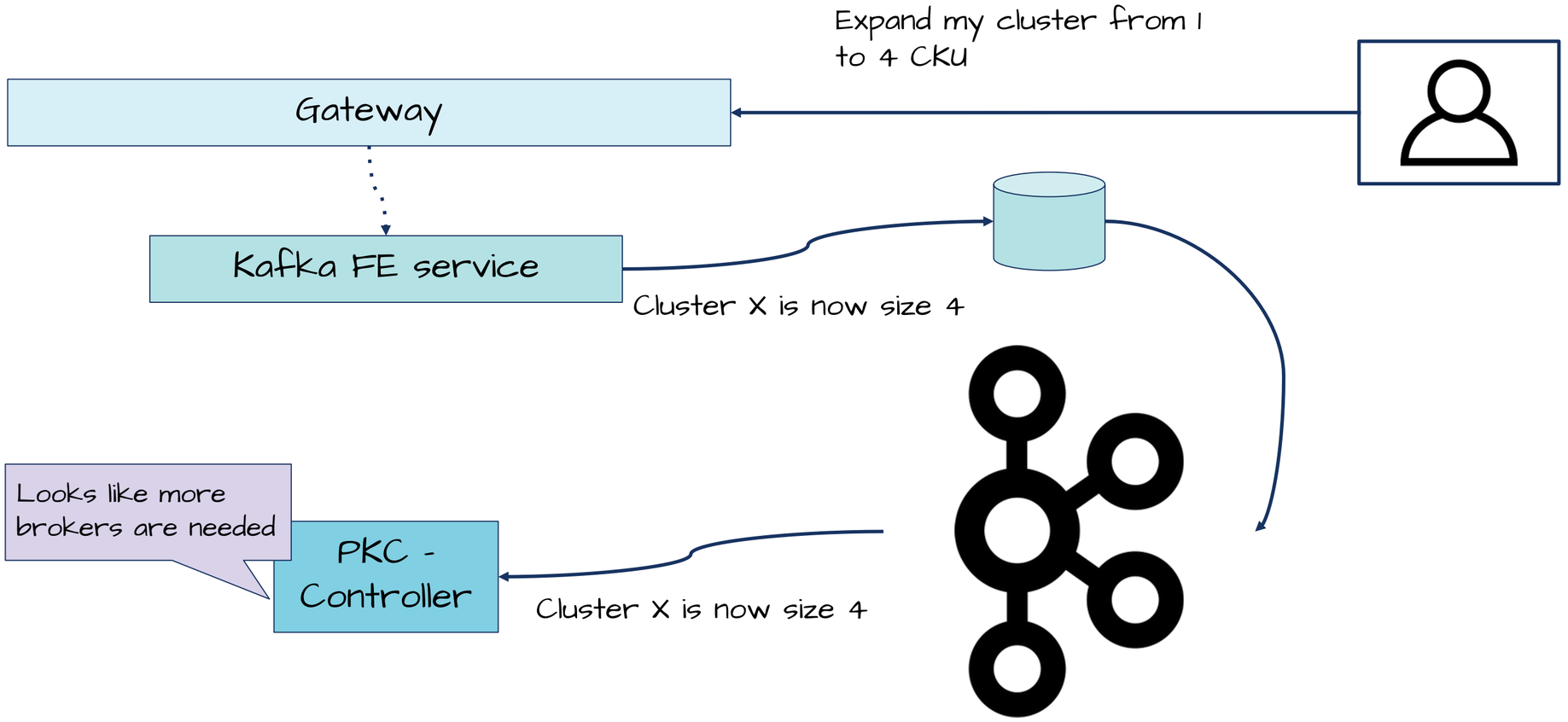
Most FE services depend on K8s
Kafka has an extra layer



Finally the gateway - depends on all FE services



How (most) services message each other



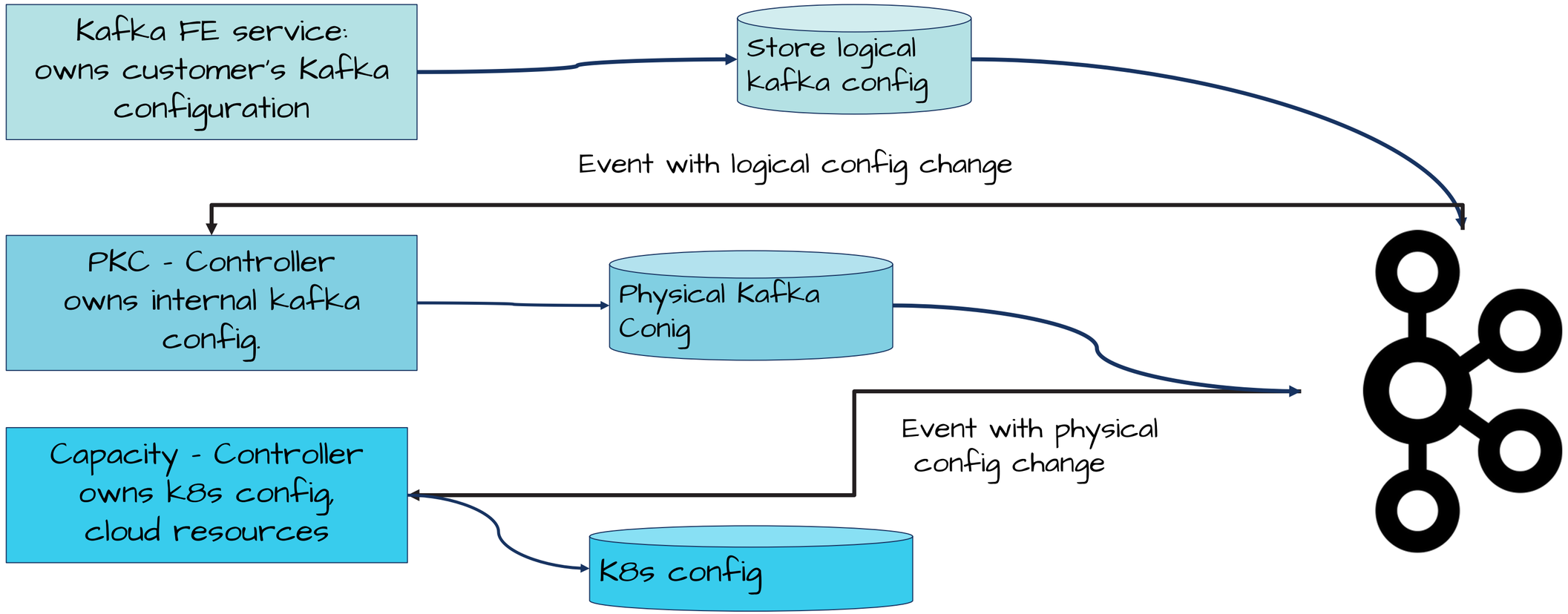
Choreography of Microservices

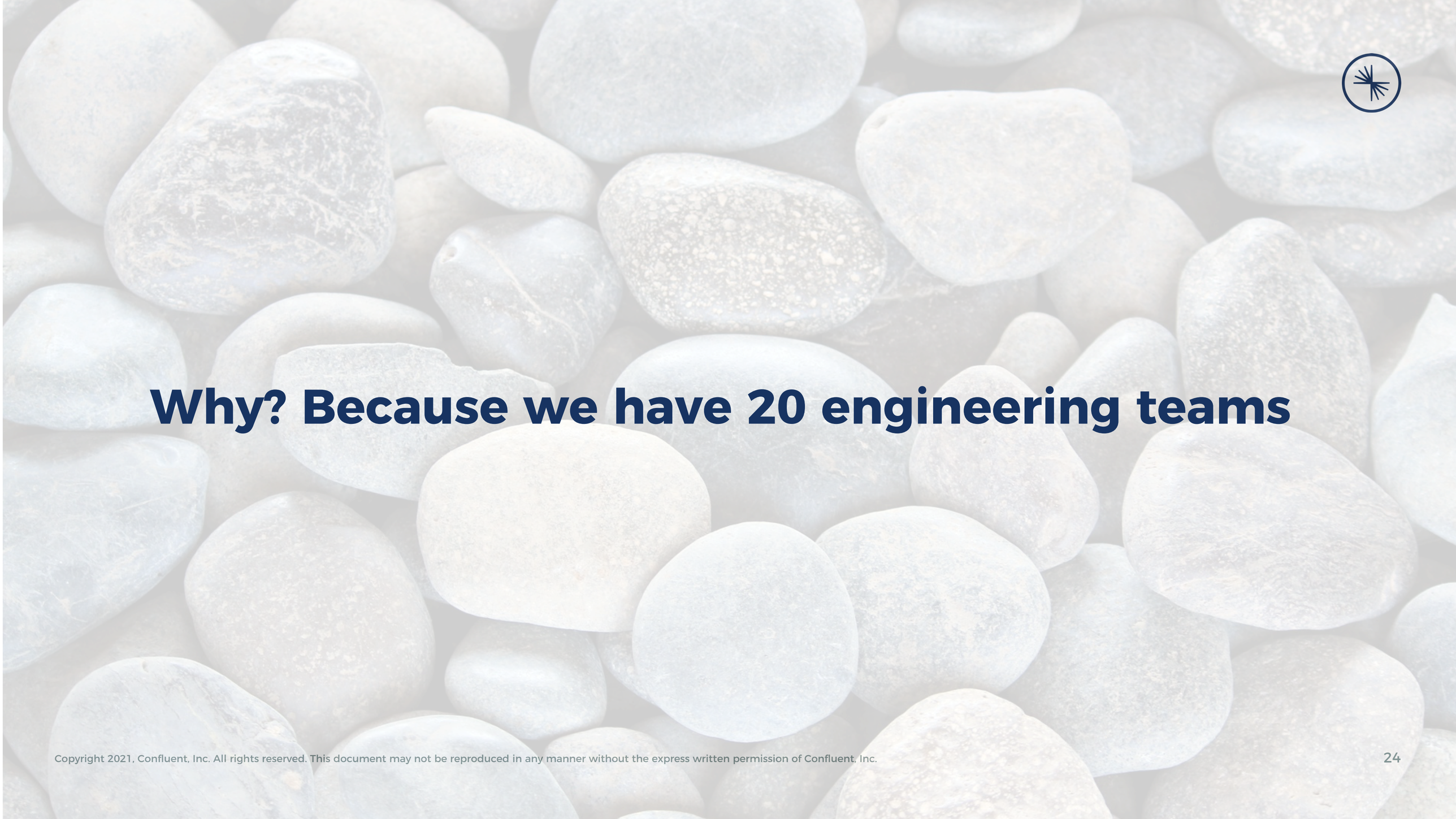
Each service owns state of a resource

Changes to state is shared as events

Each service owns reactions to events

Ownership, events and reactions





Why? Because we have 20 engineering teams

Challenges

Observability

Migrations

End to end tests

Operational Tools

Key Lessons

Persisted State with Kafka Events

The foundation of a recoverable, auditable, observable, testable system.

Layer Cake

Avoid cyclic dependency by consuming events instead of RPCs.

Breaking monolith

Always a challenge. Use appropriate tools to help with operations, testing.

Thank you!

@gwenshap @sharmavr
gwen@confluent.io
vsharma@confluent.io

 **cnfl.io/meetups**

 **cnfl.io/blog**

 **cnfl.io/slack**