FORM3
FINANCIAL CLOUD

Choosing cloud native technologies for the journey to multi-cloud

Adelina Simion

Join us to learn all about our perilous journey to the uncharted waters of multi-cloud! ⚓

@classic_addetz

Hello!

I’m excited to share the work our Engineering Teams have done!

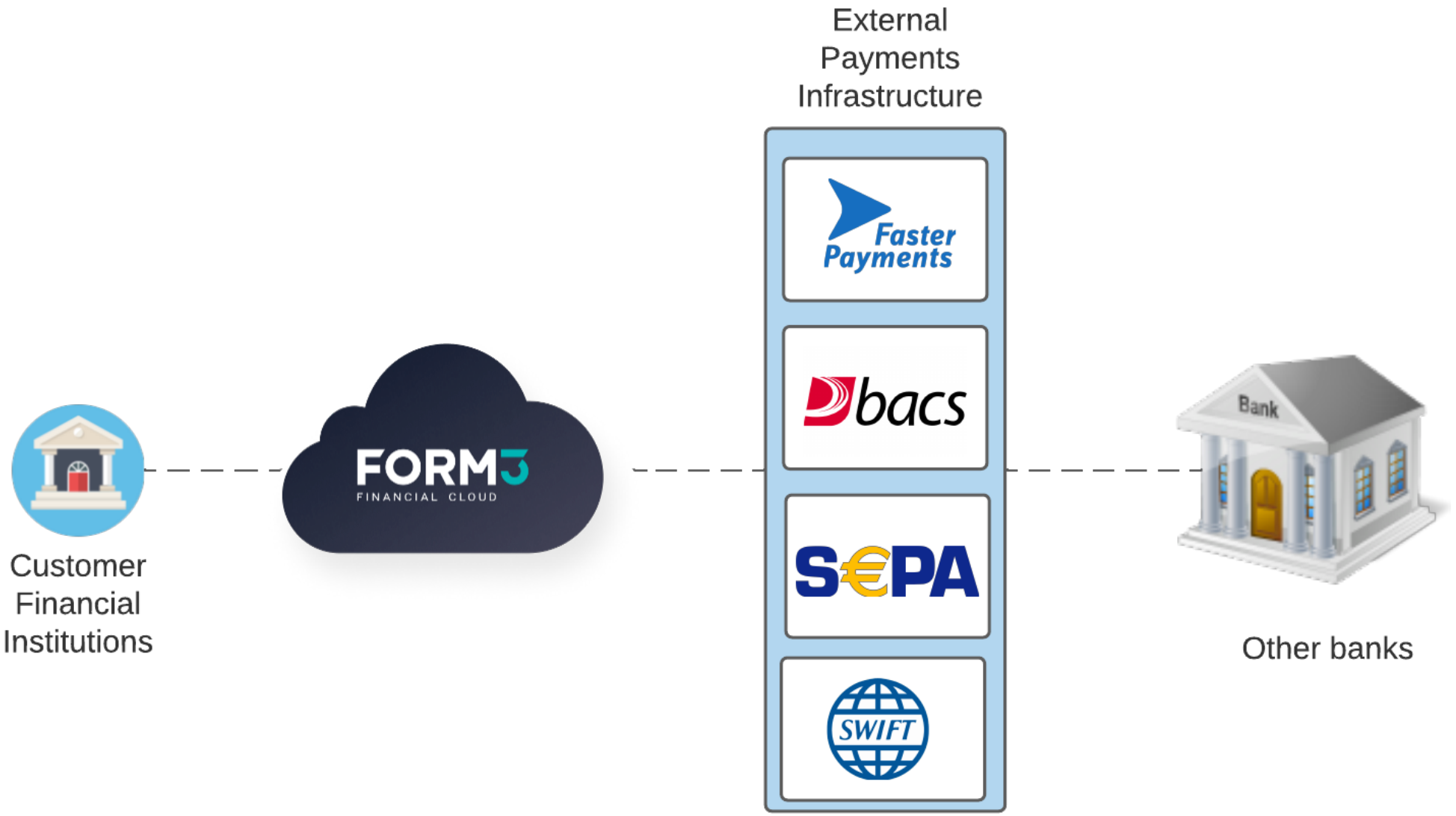
- Technology Evangelist @ Form3
- Huge honor to be speaking to you today
- I will share some of the amazing work our teams have been doing

classic_addetz

FORM3
FINANCIAL CLOUD

Introduction to multi-cloud

Payments system overview



Challenges of payments processing

Volume

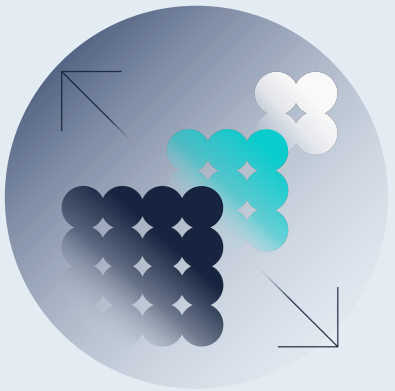
Our platform processes huge amounts of transactions with virtually an unlimited number of users.

Reliability & durability

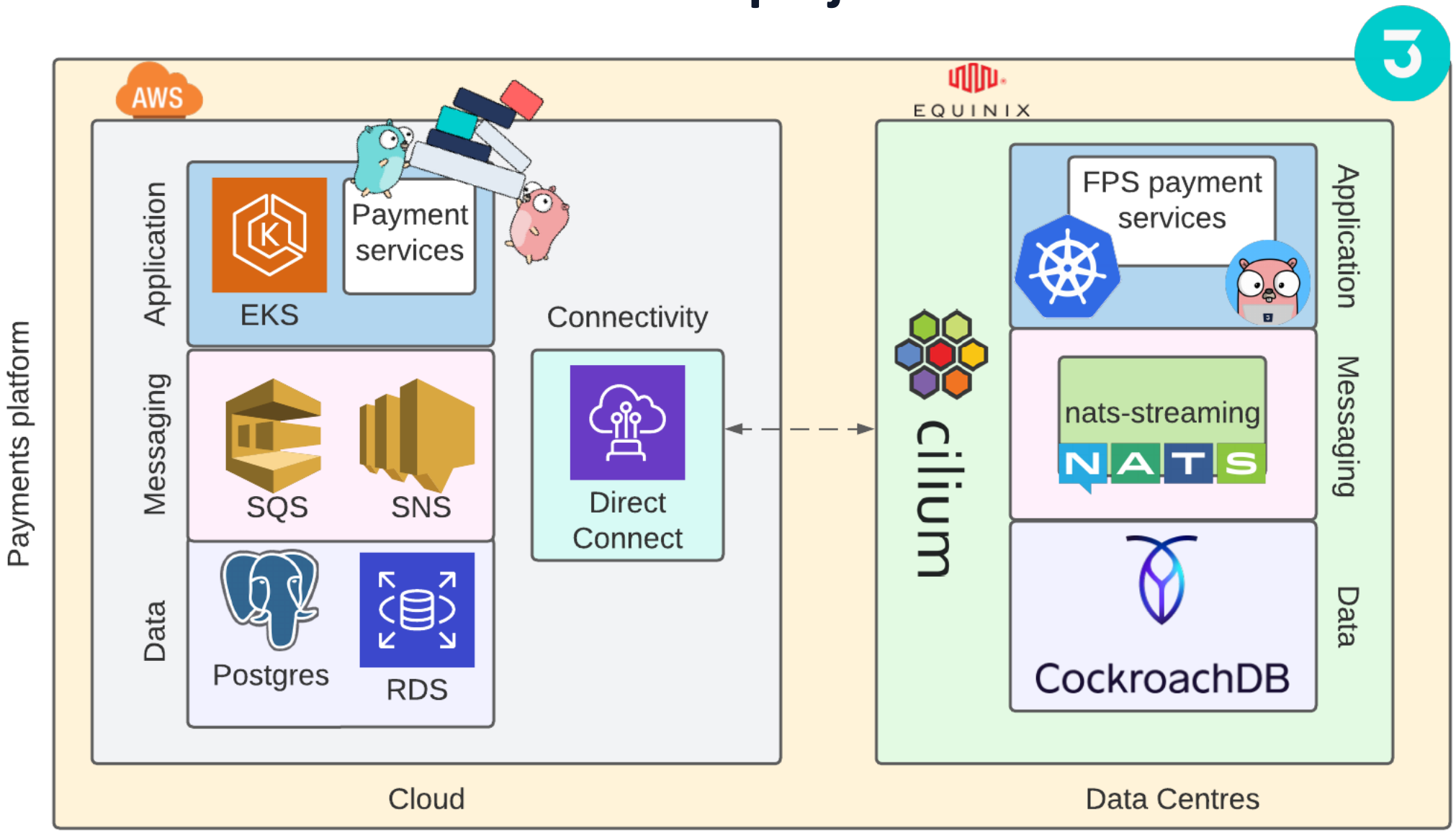
We must maintain reliability SLAs for our clients. Outages and errors must be handled without loss of data.

Maintenance

External payment schemes update their endpoints and requests.



Architecture before multi-cloud project

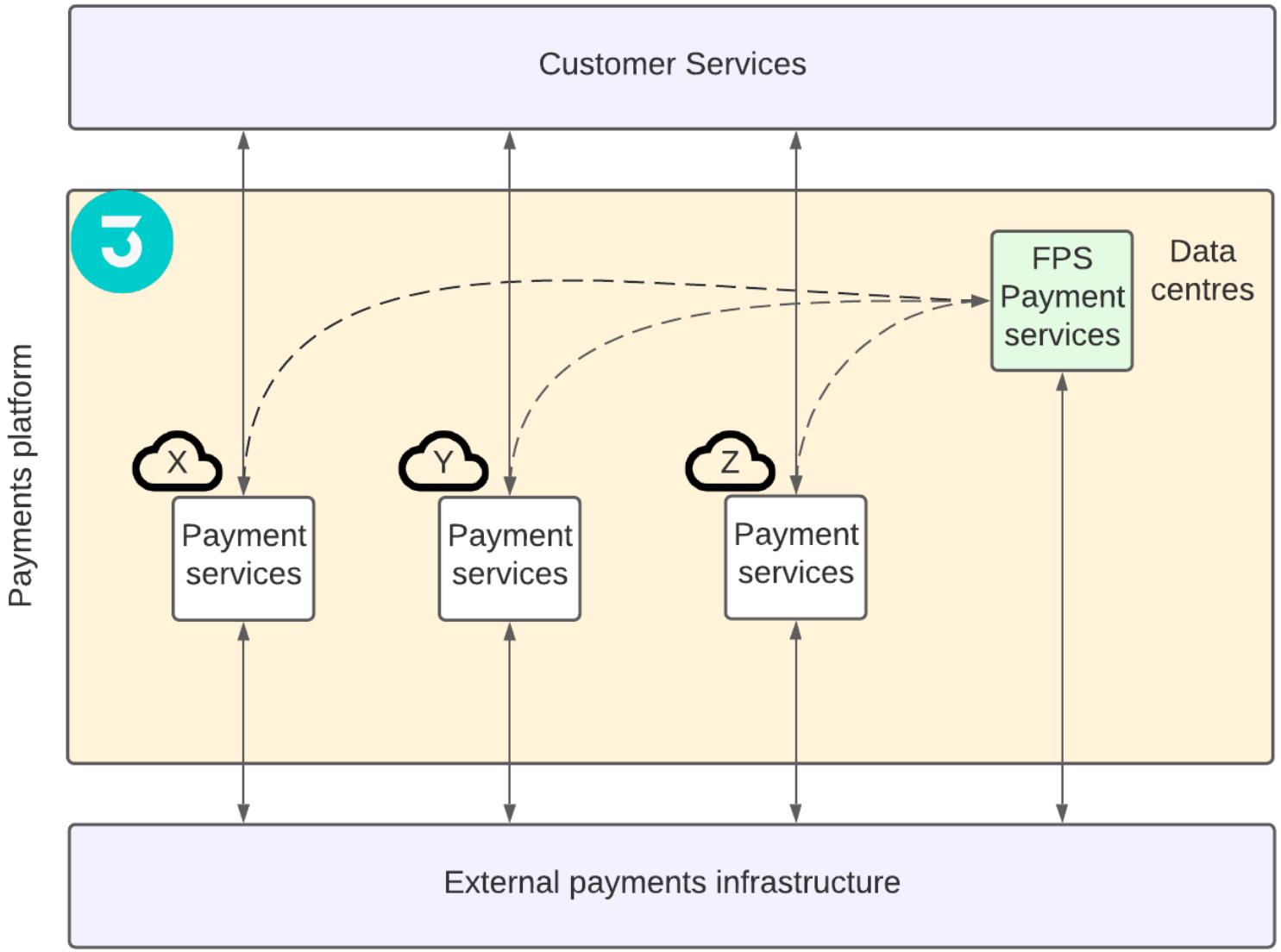


Introduction to multi-cloud

The banking sector is regulated and is one of the last to move to the cloud.

How do we prevent vendor lock in and reliance on one single provider?

A cloud by any other name...

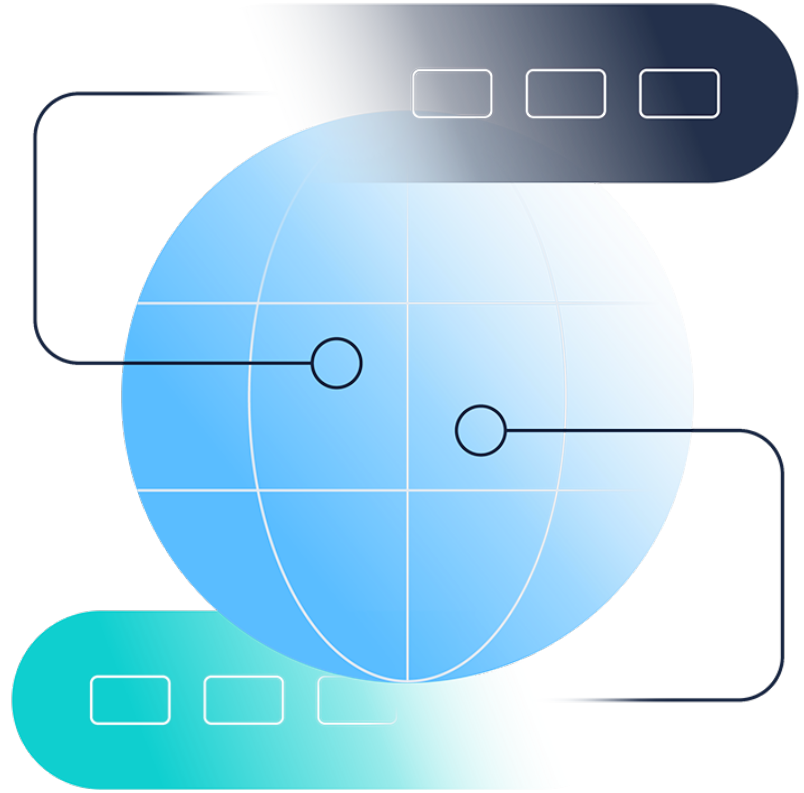


Technical challenges

As soon as we start to discuss moving away from a single cloud, our teams identified some technical challenges from the beginning.

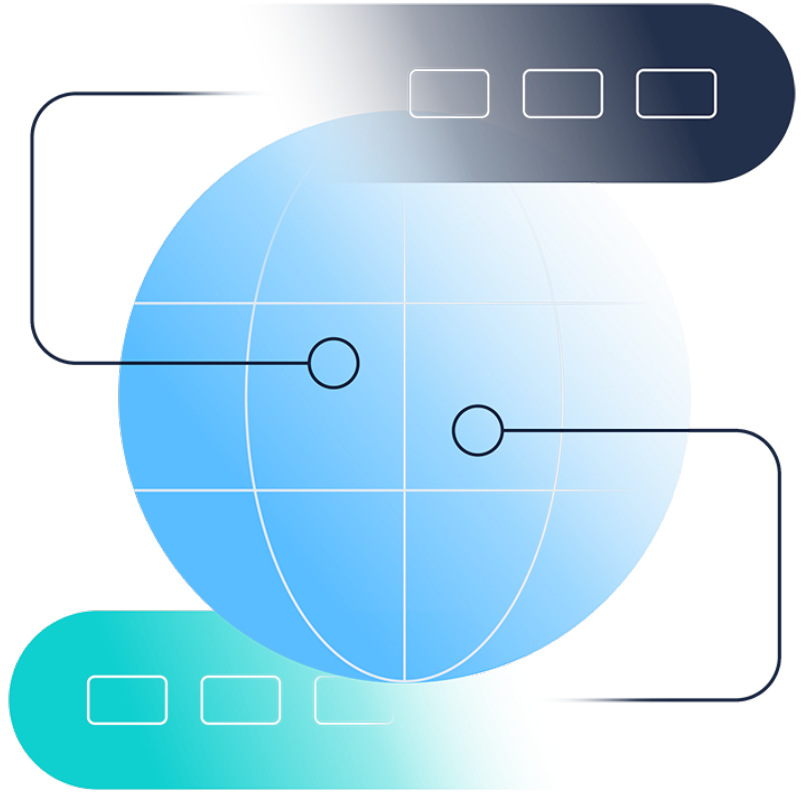
- Secure networking
- DNS routing
- Service discovery
- Synchronization data volume
- Latency between clouds
- Performance

Building our platform with cloud agnostic technologies allows us to abstract away from the cloud, as opposed to building specific versions of our platform.



Deploying with Kubernetes

Our teams need a developer experience that abstracts away which cloud vendor a specific service is being deployed on. All clouds should behave identically.



Why go with K8s?

Automation

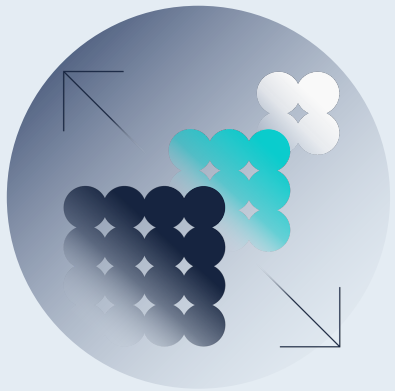
Allows us to automate the deployment, scaling and management of applications

Cloud agnostic

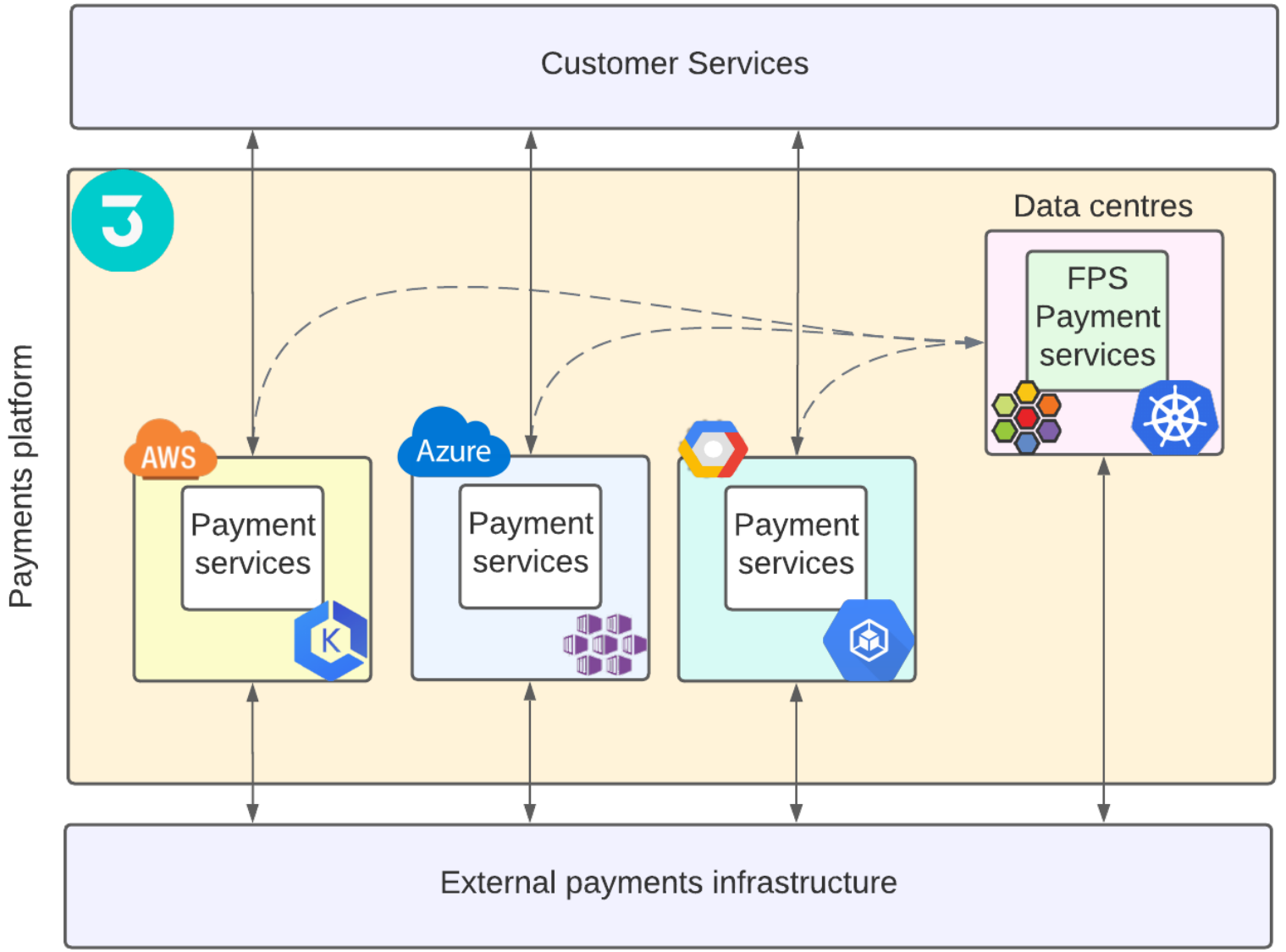
Deploys workloads in any public or private cloud. Easily moving services/workloads to wherever we need.

Extendable

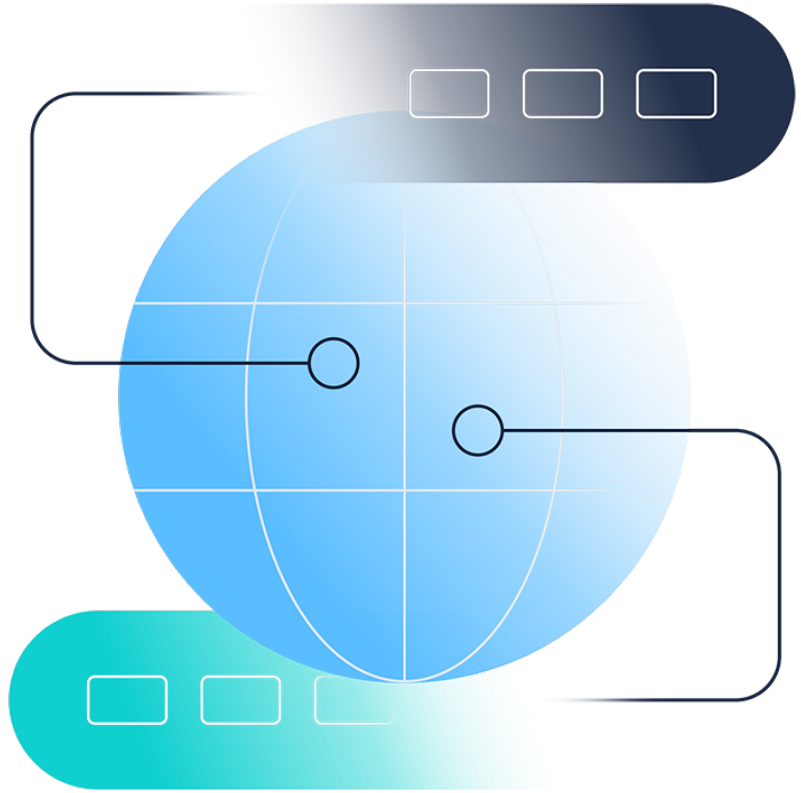
Allows us to create custom native behaviour with Operators and CRDs.



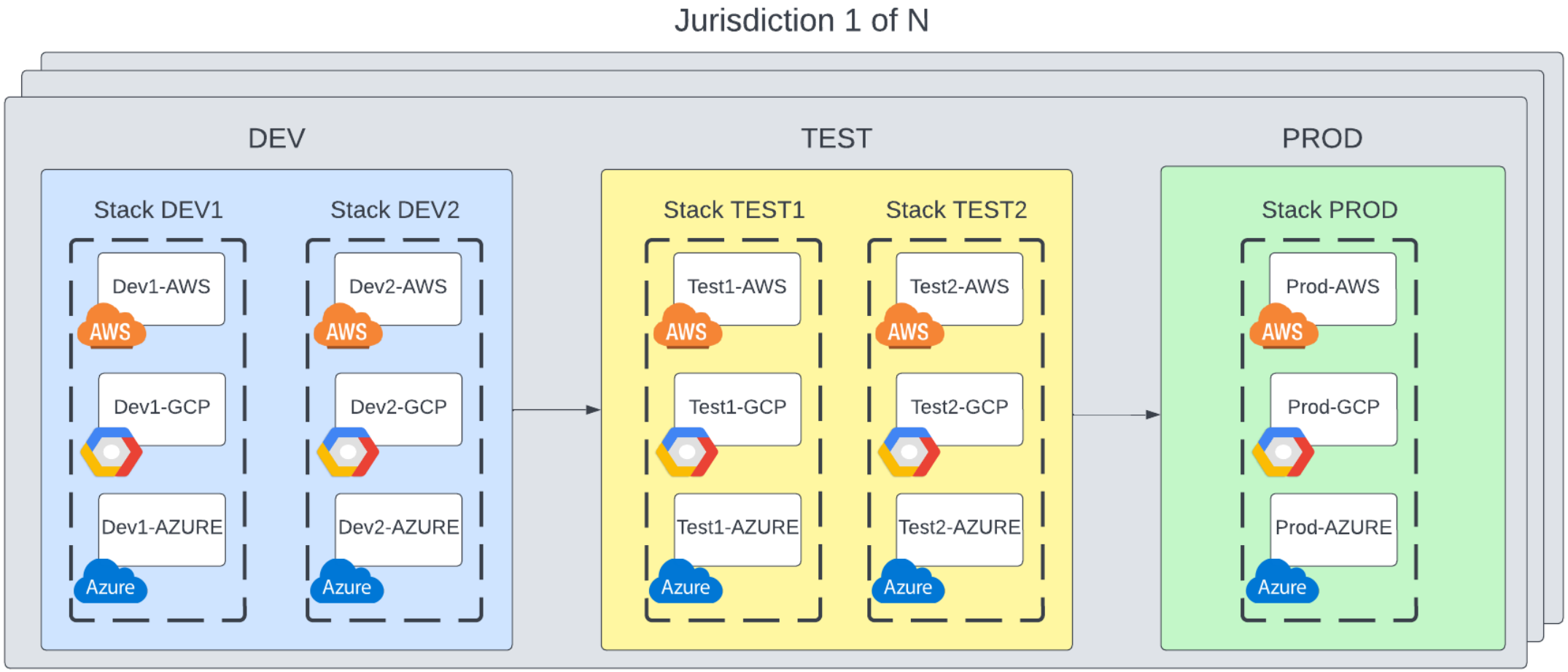
Going multi-cloud



Running Kubernetes independently can be complex. We push as much platform responsibility to the cloud vendors as possible and concentrate on the value add.



Our stacks and environments



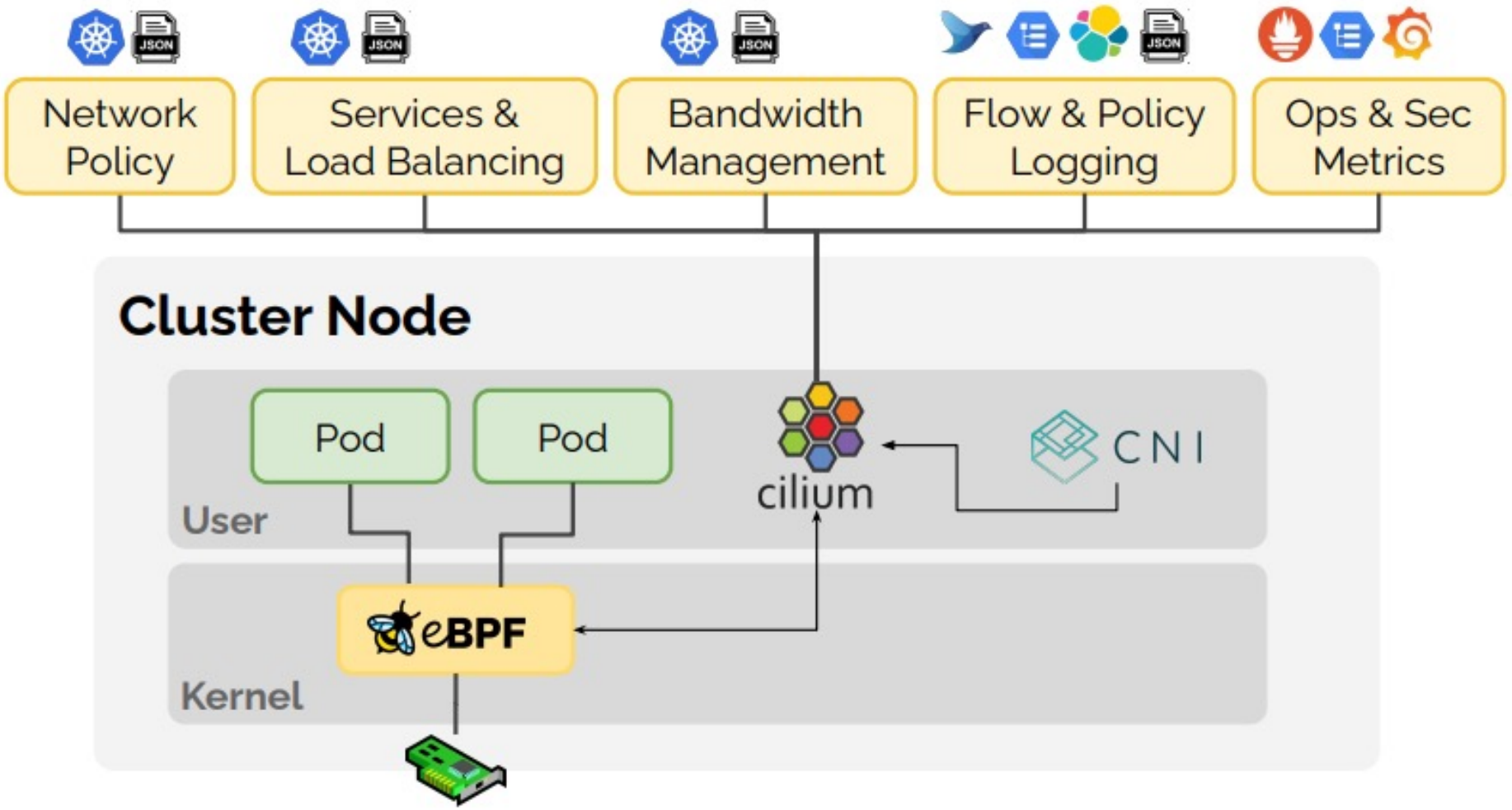
Networking with Cilium

Cloud connectivity requirements

Our solution must satisfy our internal and client requirements.

- Low latency
- Resilient and Fault tolerant
- Secure
- Highly available
- Automatic failover
- Audited third party vendors

What is Cilium?



source: https://github.com/cilium/cilium

Why go with Cilium?

Network policies

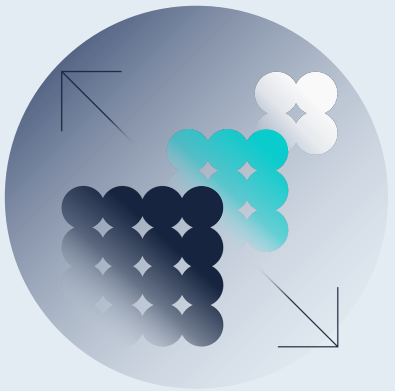
Enhance Kubernetes network policies to restrict ingress and egress.

Cloud agnostic

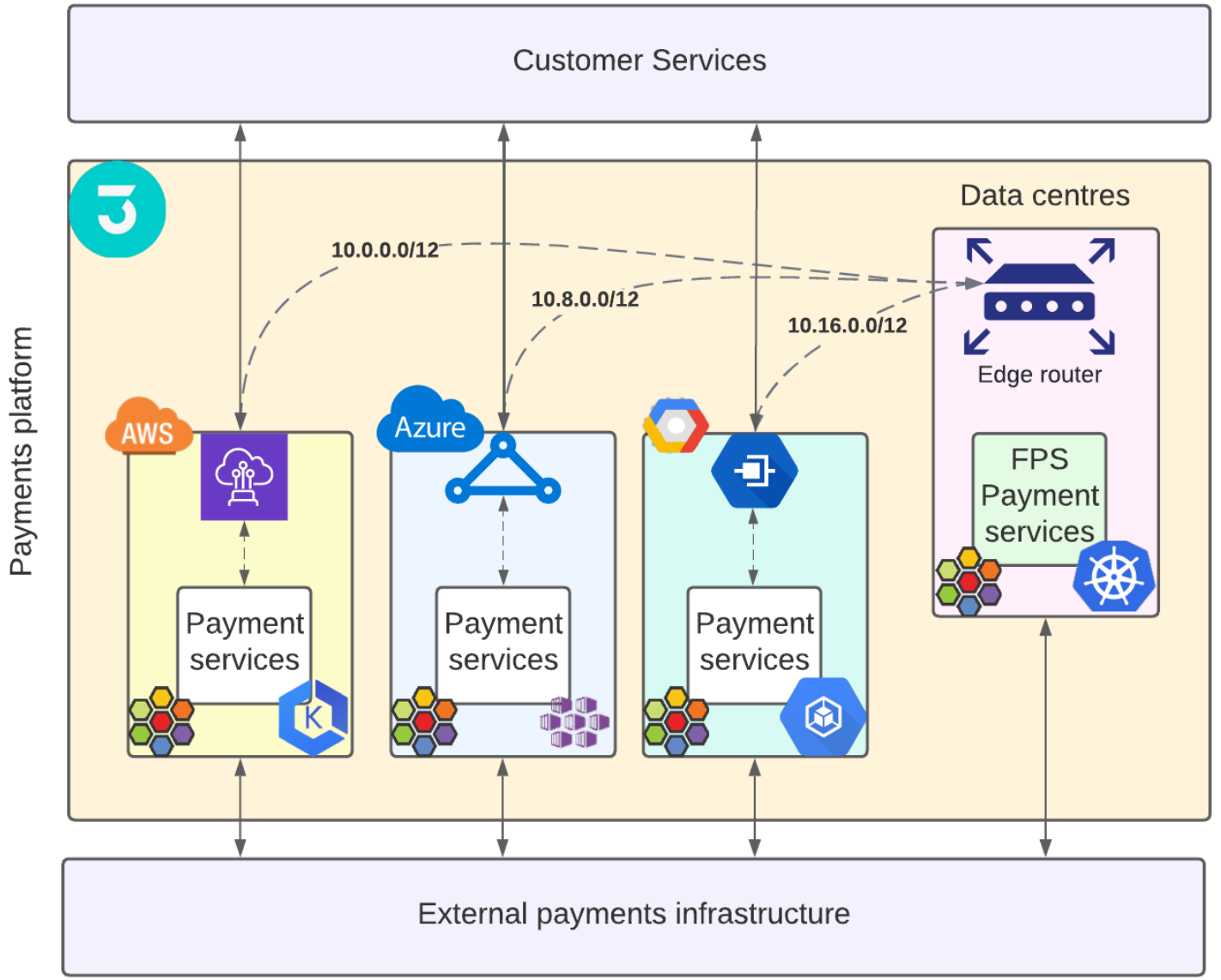
Provides connectivity, service discovery and load balancing across clouds.

Inbuilt observability

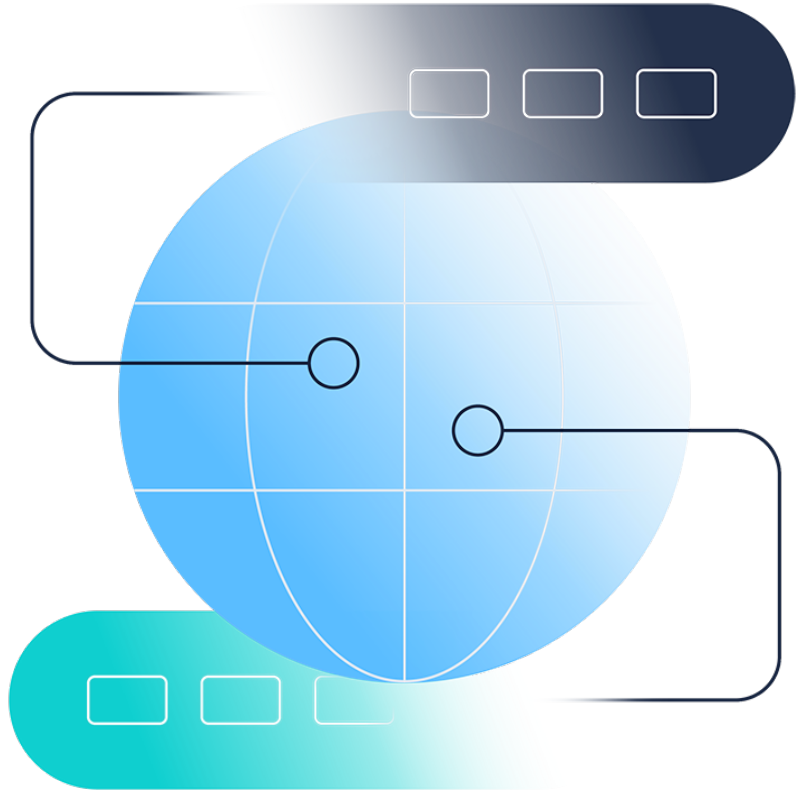
Provides observability with Hubble and Grafana.



Connecting multi-cloud



Cilium provides us with the ability to run our multi-cloud solution using cluster mesh in the future.



Event streaming with NATS

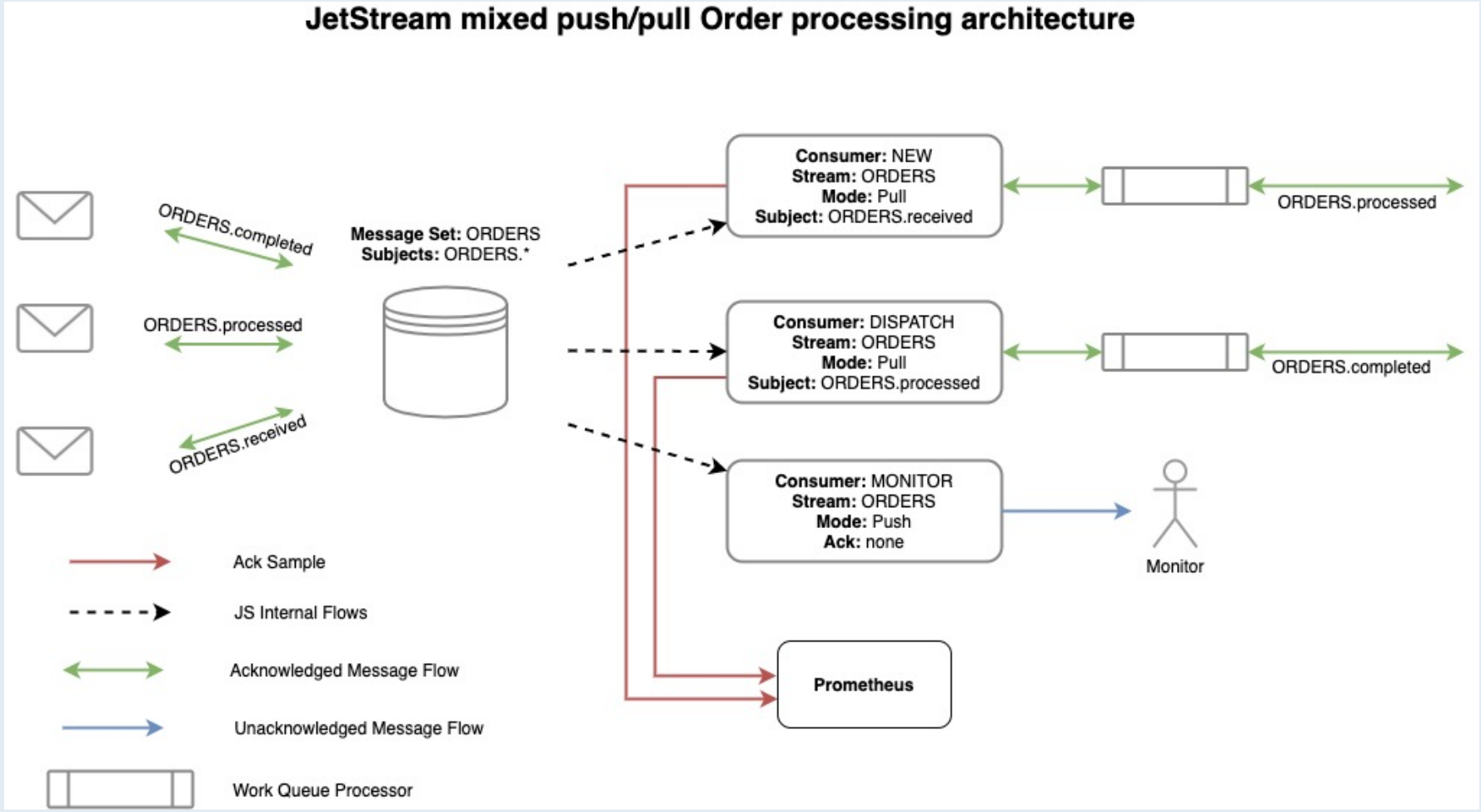
Event streaming with NATS

Messaging system requirements

We need to send data across our multi-cloud system to ensure data is correctly replicated.

- Secure
- Supports multi-cloud
- Scalable
- Persistent
- Go client support
- Monitoring and instrumentation

What is NATS JetStream?



source: https://docs.nats.io/nats-concepts/jetstream

FORM3
FINANCIAL CLOUD

Why go with NATS JetStream?

Push & Pull clients

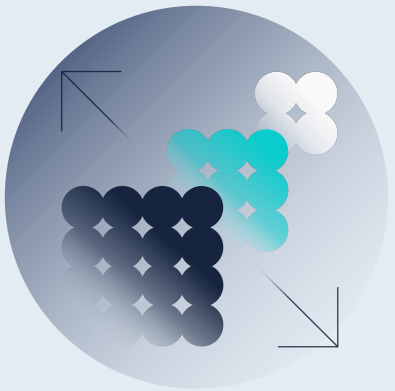
Streams support push and pull clients, as well as wildcard subscriptions.

Cloud agnostic

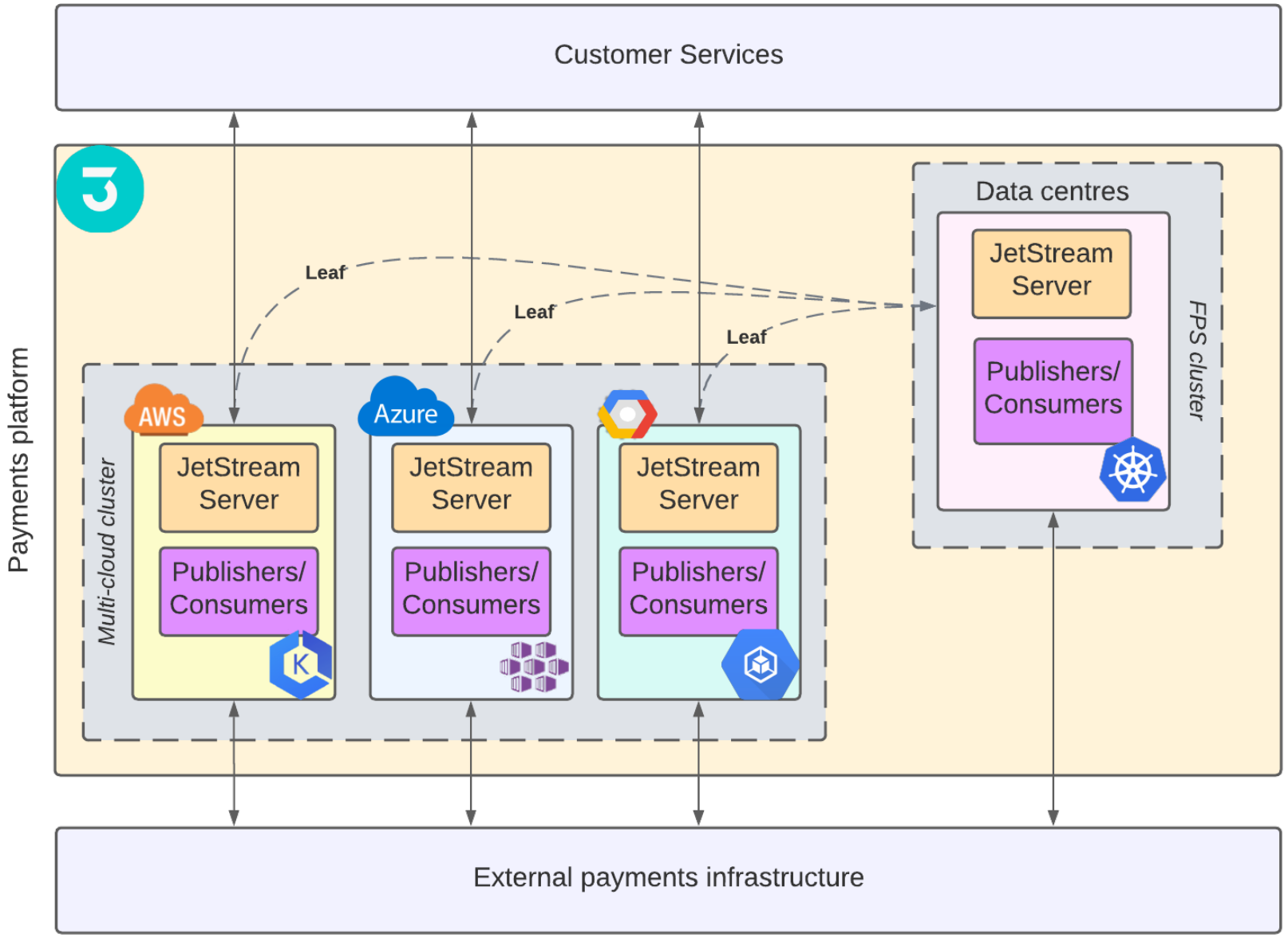
Open source and written in Go, it can work in any cloud.

Scalability & durability

Provides horizontal scalability as well as exactly once message delivery guarantees.



Messaging in multi-cloud



FORM3
FINANCIAL CLOUD

Database storage with CockroachDB

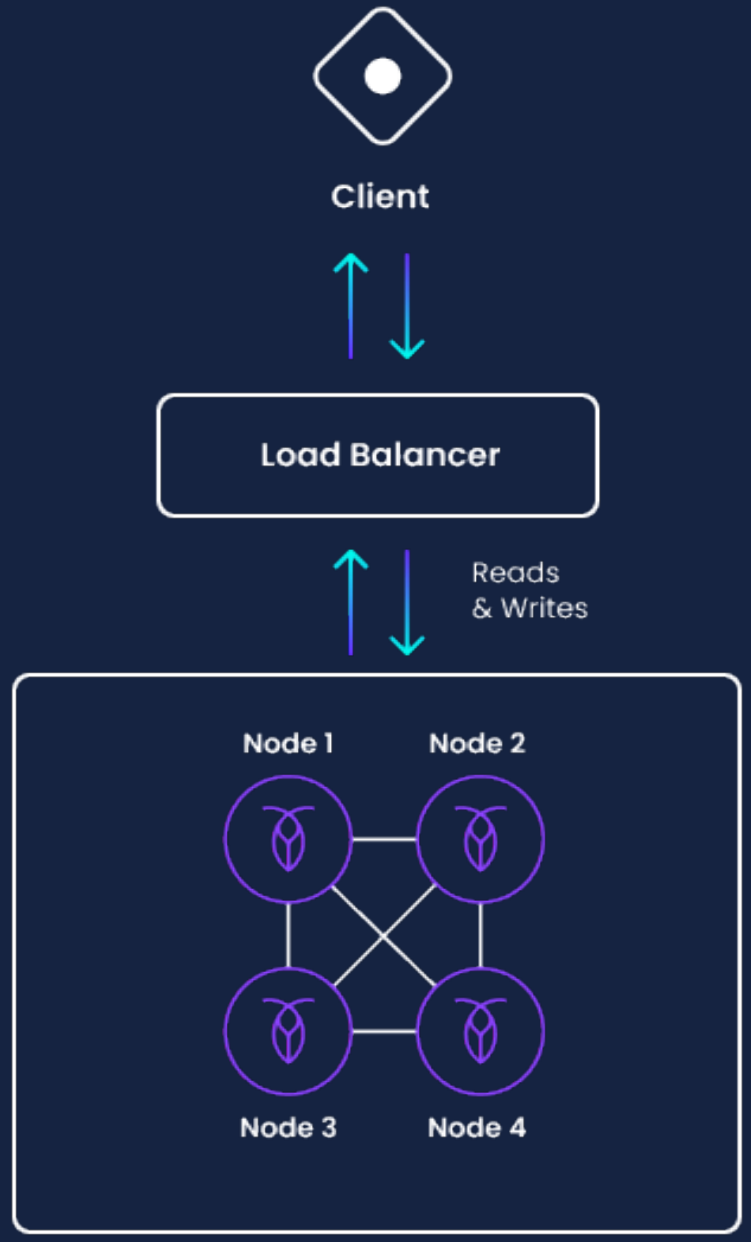
Database requirements

Just like with our other components, we had key requirements for our database.

- Supports multi-cloud
- SQL compatibility
- Strongly consistent
- Scalable
- Easy to run in Kubernetes
- Secure

What is CockroachDB?

- Distributed database
- Write to any node
- ACID compliant
- Balances ranges automatically
- Open source



source: https://www.cockroachlabs.com/product/

FORM3
FINANCIAL CLOUD

Why go with CockroachDB?

PostgreSQL compatibility

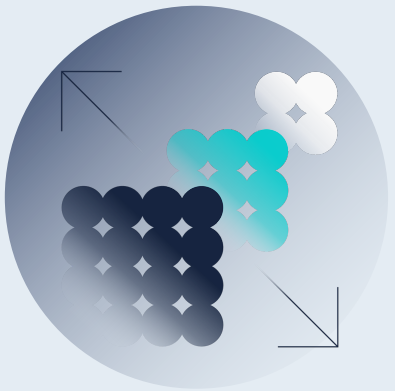
Switching our workloads from our current Postgres setup was easier.

Cloud agnostic

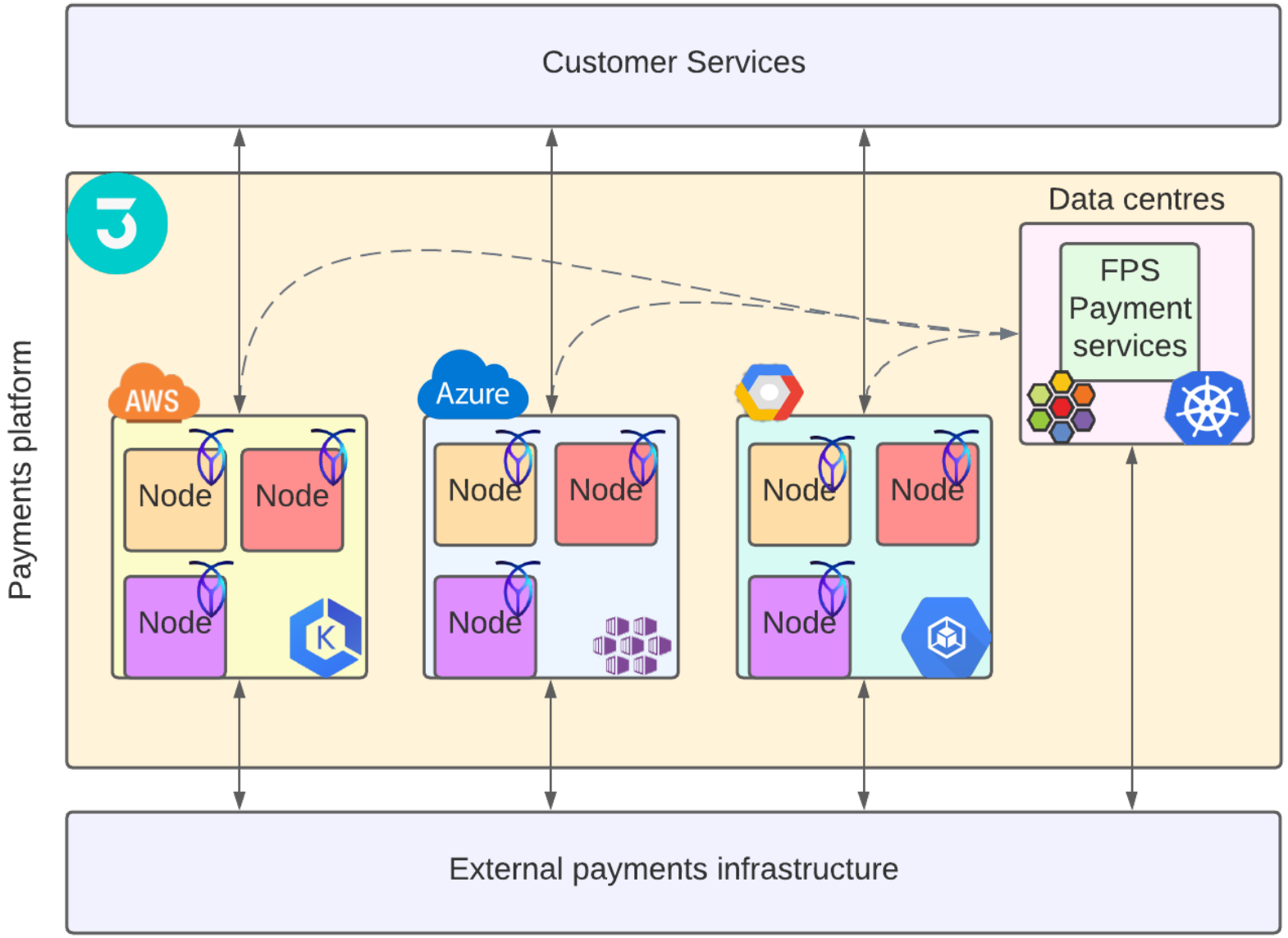
Runs in Kubernetes across cloud vendors.

Strongly consistent

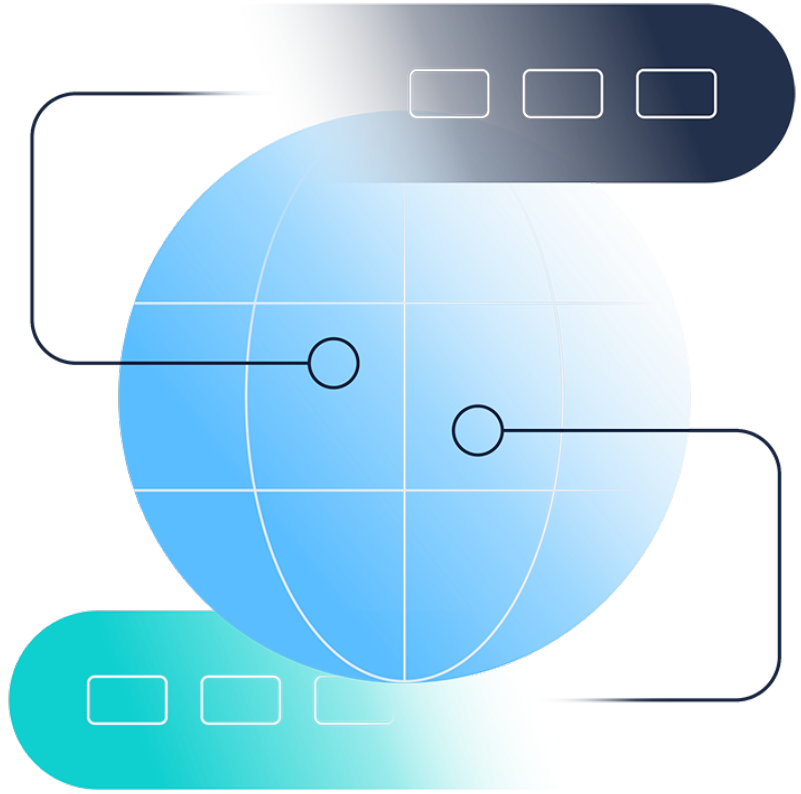
We should not drop or duplicate transactions.



Our database setup



CockroachDB is technically a CP system. We achieve availability with our multi-cloud setup and CockroachDB's inbuilt self healing which rebalances nodes.



Conclusions

Three rules of thumb

Test end to end

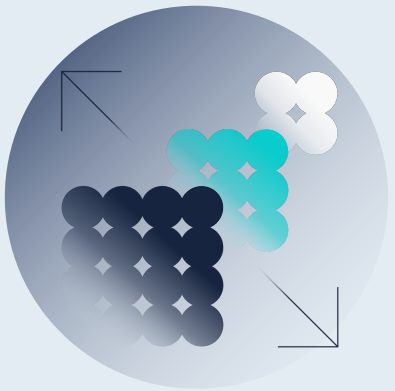
We test our services end to end with a variety of types of load.

Expect errors & retries

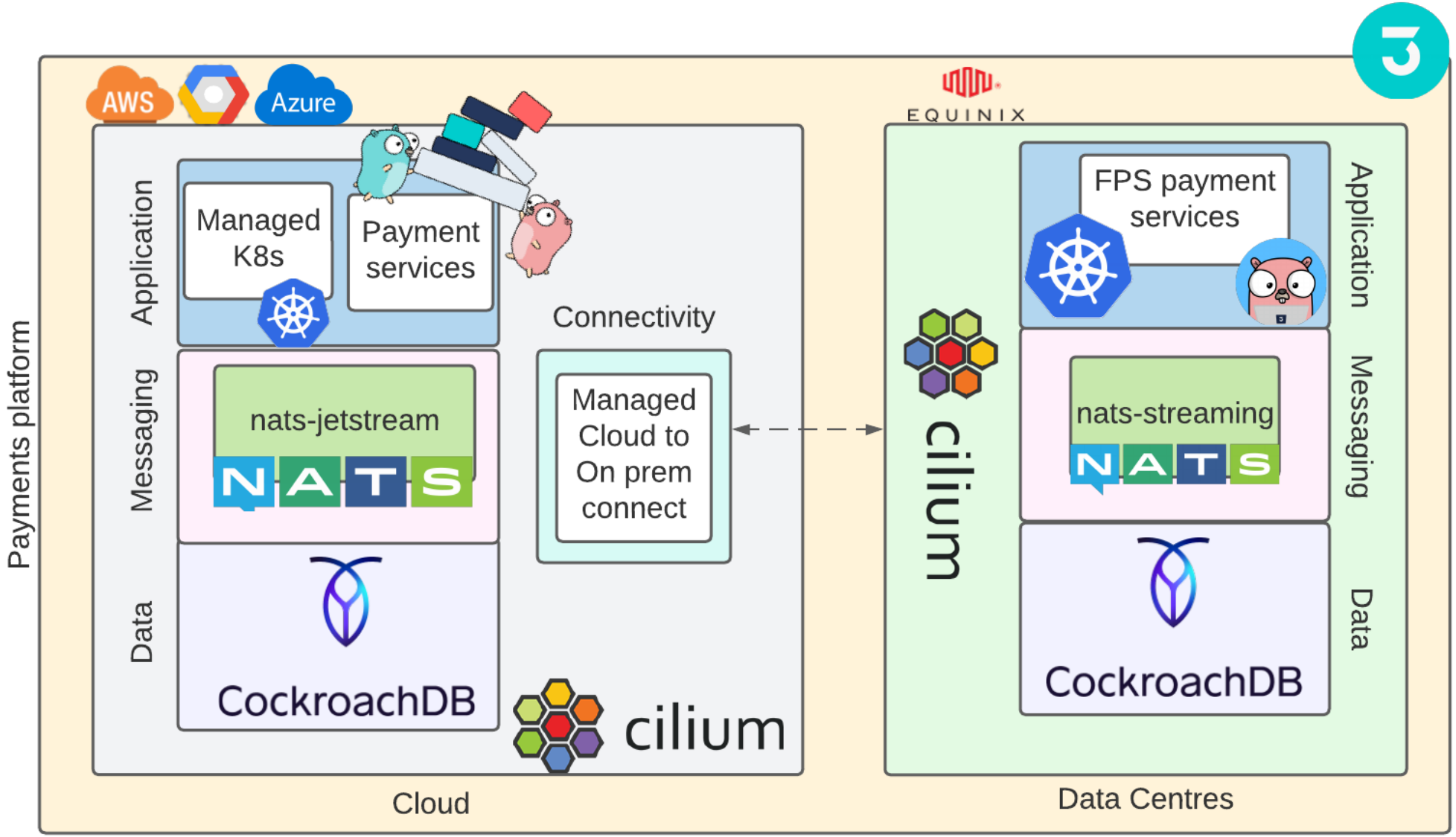
Errors and retries are even more prevalent in multi-cloud. Make sure you design your services with this in mind.

Go cloud agnostic

Push as much work to specialized services as possible. Rely on the public cloud vendor as much as possible, while avoiding lock-in.



Cloud native technologies



Thanks for listening! ❤️

Check out our podcast, our engineering site and our Twitter account.

techpodcast.form3.tech

form3.tech/engineering

@Form3Tech

We'll be at our stand in Pavilion 1!

FORM3
FINANCIAL CLOUD