Building multi-region & multi-cloud services with Kafka

Travis Jeffery

→ github.com/travisjeffery

→ twitter.com/travisjeffery

→ medium.com/@travisjeffery

What we're talking about today

→ Building services on multiple regions, multiple clouds in a manageable way

→ Example: running services in AWS with others in GCP or us-west-2 & us-east-1

→ 3 system architectures according to message ingress and egress

Why?

Why multiple regions?

→ **Latency**: Customers want your service to be fast.

→ **Compliance**: You want a global customer base but different countries have different laws saying where/how data is stored.

→ **Availability**: Increase your resilience above what a single region's availability zones.

Why multiple clouds?

→ Features and services available differ on each cloud.

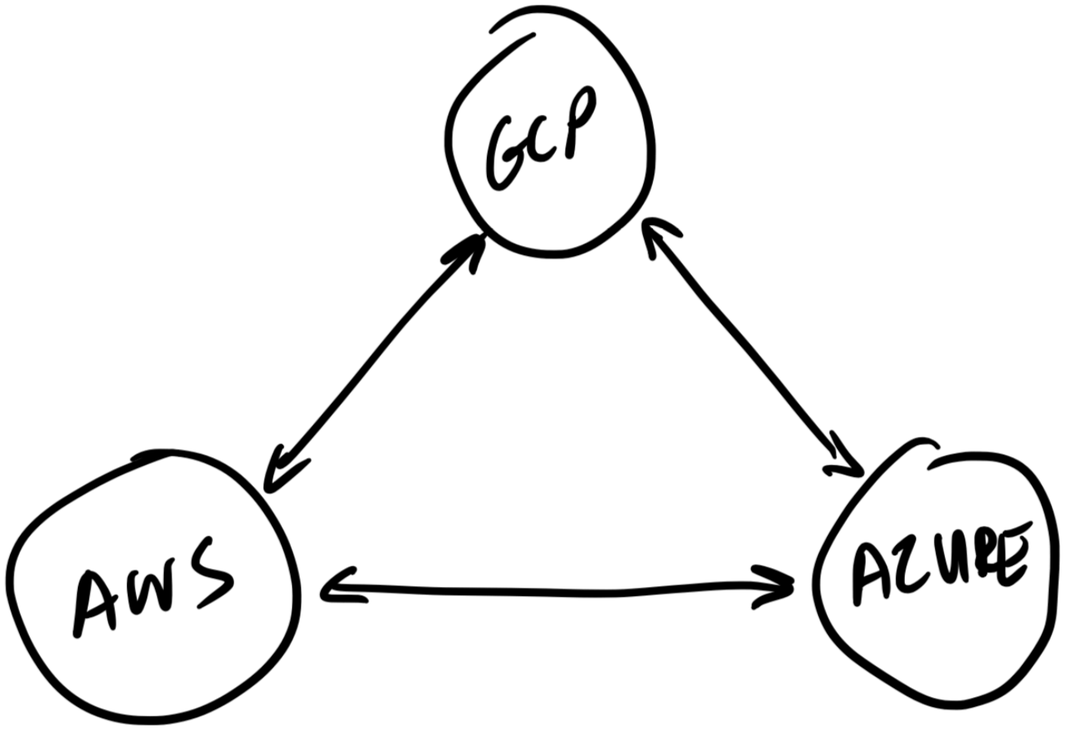
→ Save money with betters deals on different vendors.

→ Reduce dependency on your provider.

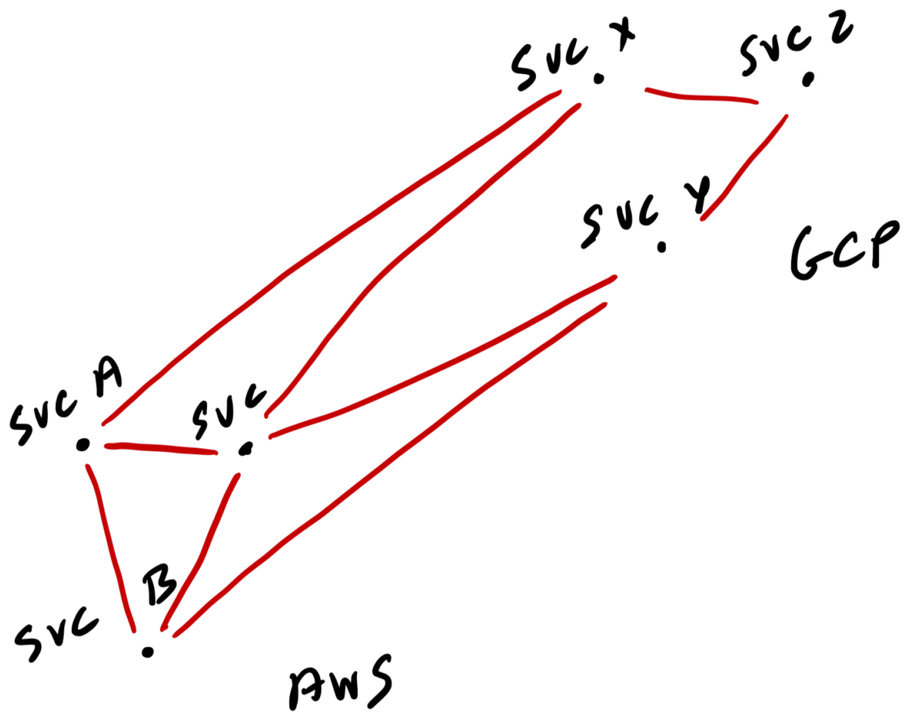
How?

A typical, naive system

Request-response to each region & cloud



Request-response to each region & cloud



Request-response to each region & cloud

Bad:

→ Security: Secure all endpoints

→ Routing: VPN tunnels, firewalls

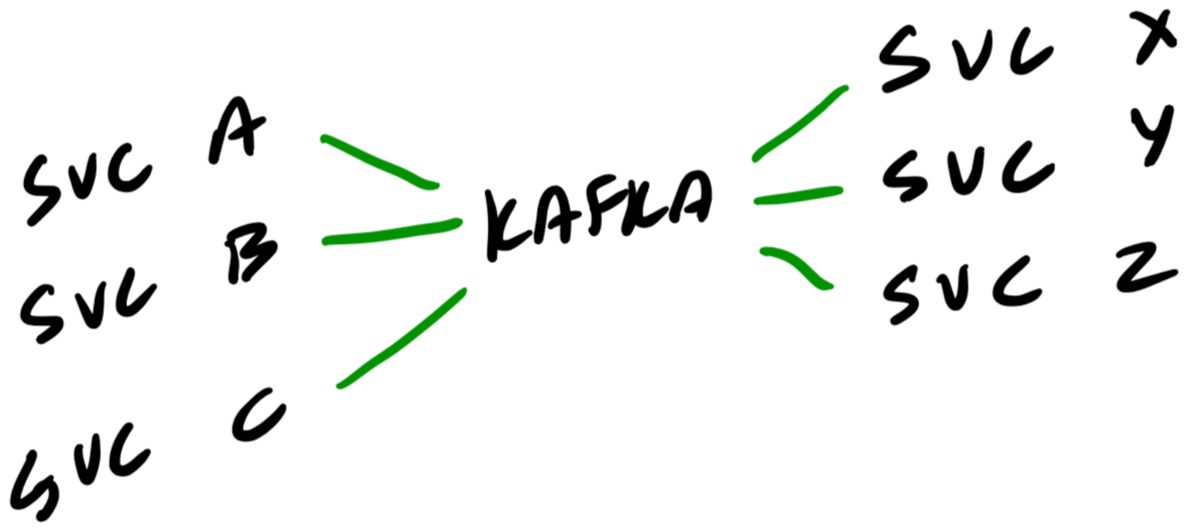
→ Service discovery: Must know where & how to msg

Good:

→ Latency: Direct connections.

A new, smarter system

Using Kafka as central message hub



Message header routing: low ingress, low egress

Message header routing: low ingress, low egress

```
MSG = {
  ROUTER = {
    CLOUD: 'AWS',
    REGION: 'US-WEST-2'
  }
  DATA: ...
}
```

→ CREATE_ACCOUNT_TOPIC

REGIONAL SERVICE

MSG ← CONSUMER

IF MSG IS FOR ME
  OPERATE
ELSE
  IGNORE

Topic per cloud and region: high ingress, low egress

```
MSG = {
  DATA: ...
}
```

→ CREATE_ACCOUNT_AWS_US_WEST_2

Topic per cloud and region into local kafka clusters: high ingress, high egress

Extending for when you need a request-response

→ Problem: End-to-end latency, for example a UI with a live search field

→ Solution: Give the UI a regional API to request directly

Extending for when you need a request-response

Questions?

Absolutely anything: Kafka, gRPC, Terraform...

Can ask me after too.

Where to go next

→ Don't want to manage Kafka? Confluent Cloud: https://www.confluent.io/confluent-cloud/

→ github.com/travisjeffery

→ twitter.com/travisjeffery

→ medium.com/@travisjeffery