Building Confluent Cloud

Travis Jeffery

- twitter.com/travisjeffery
- github.com/travisjeffery
- medium.com/@travisjeffery

Talk roadmap

How to build services on multiple regions, multiple clouds in a manageable way.

→ What is Confluent Cloud

→ How we built it

→ Lessons learned

→ Where we're heading

→ Ask Me Anything

Travis Jeffery

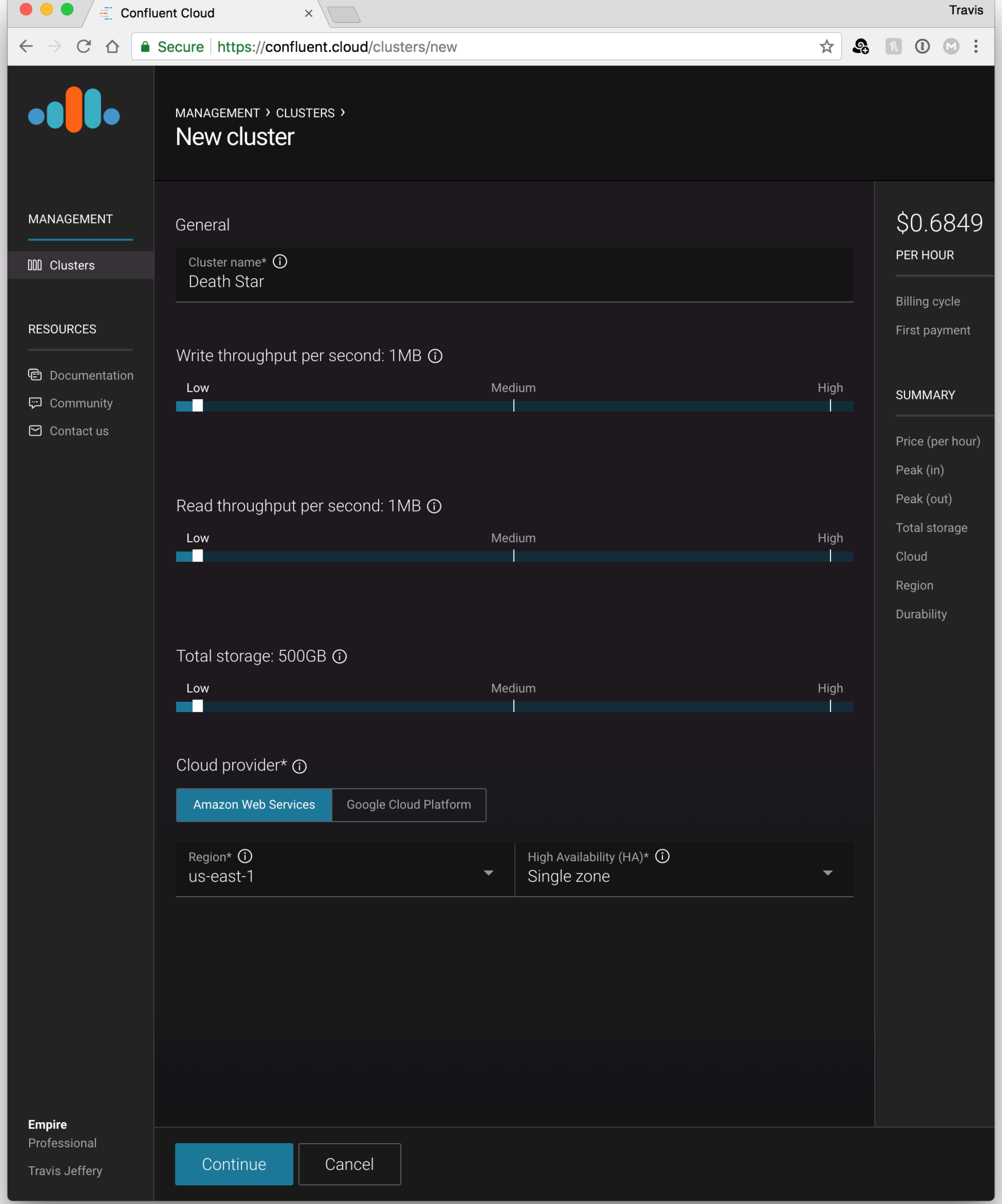
- → Cloud Eng at Confluent
- → Wrote my own Kafka in Go called Jocko
- → Other projects: Timecop, Mocha, Clang-Format on Xcode
- → Worked at Basecamp, Segment

My face on the internet ->

What is Confluent Cloud?

→ Today: Kafka as a Service

→ Available on AWS and GCP in many regions



Confluent Cloud

Secure https://confluent.cloud/clusters/new

MANAGEMENT › CLUSTERS ›

New cluster

General

Cluster name*: Death Star

Write throughput per second: 1MB

Read throughput per second: 1MB

Total storage: 500GB

Cloud provider*: [✓ Amazon Web Services] Google Cloud Platform

Region*: us-east-1

High Availability (HA)*: Single zone

$0.6849 PER HOUR

What is Confluent Cloud?

Tomorrow:

→ More services: Schema Registry, Connect, Streams, Confluent Control Center

→ More features: Topic management, dashboard metrics, SSO auth, etc.

→ More regions, more clouds

Building it

The goal: Try to build a PaaS on multiple regions, multiple clouds in a manageable and cost efficient way.

Starting point

→ CLI in Python

→ Orchestrated by Kubernetes

→ Generated YAML with Jinja

→ Shelled out to kubectl

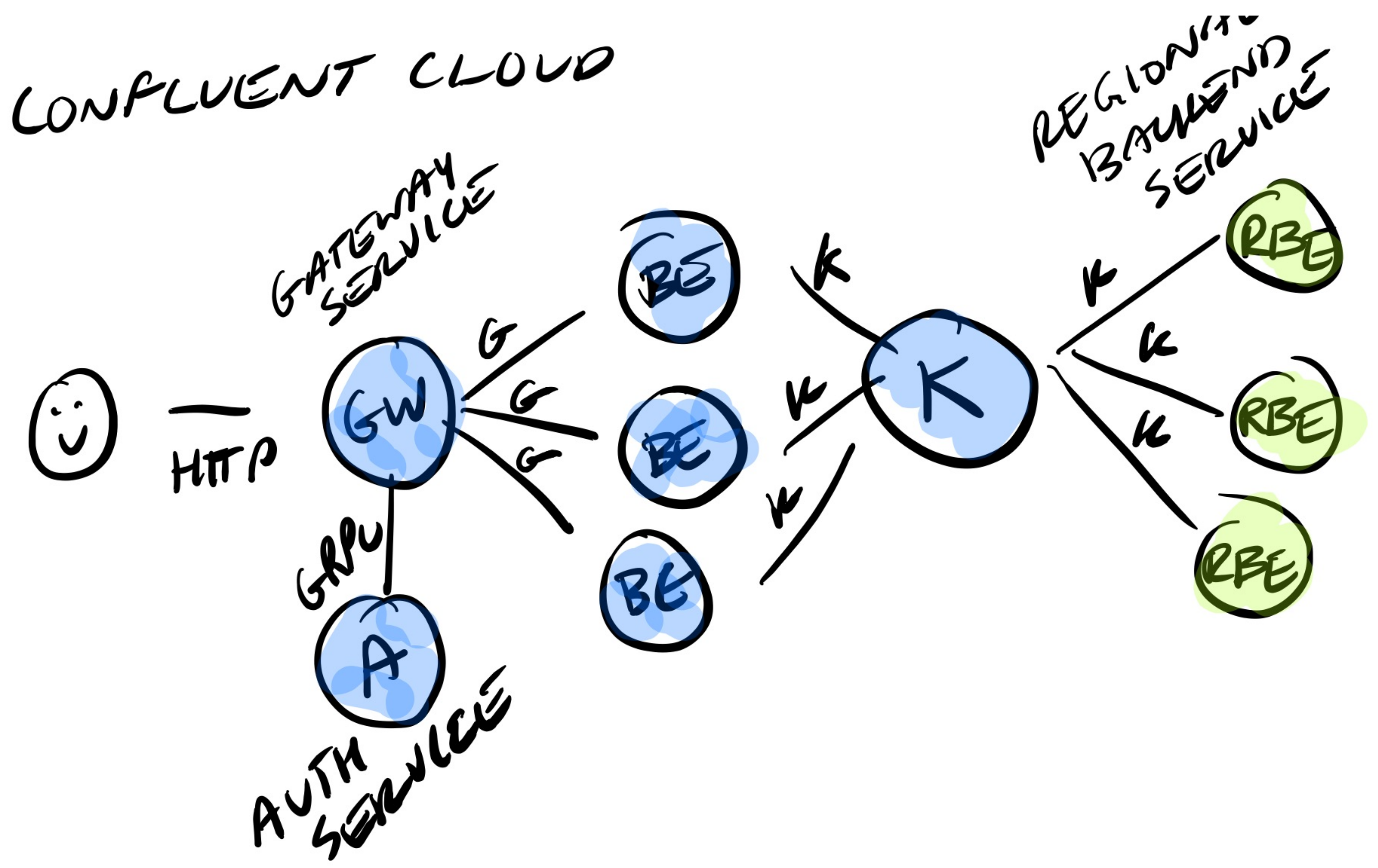
Problems with this setup

- → No API to build on
- → No UI - no user access, staff managed
- → No infrastructure management
- → No version consensus
- → Hard to test
- → Static configuration
- → Lots of work to onboard to dev or use it

Fresh start

- → Libraries/APIs first
- → Type-safe calls via Go and gRPC
- → Kafka for async, secure messaging to other regions
- → Terraform managed infrastructure/configuration
- → Still orchestrated by Kubernetes

Request flow



Libraries first, then APIs, then CLIs

- → Helps you focus on flexible, robust API
- → Can put aside requirements of end program
- → End program is a small layer tying together config and libs
- → Accessed via HTTP/RPC and CLI

Go and gRPC

- → Same lang as our infra, tighter integration and clients: Kubernetes, Terraform, Docker
- → Type-safe calls
- → Easy to run different API versions
- → Defined/managed in protocol buffers
- → Service clients for free

Kafka

→ Cross region, cross cloud, simple, and secure networking

→ Central cluster in mothership

→ SASL/PLAIN authentication

→ All services just need to know its endpoint on the internet

Routing messages per region

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

→ CREATE_ACCOUNT_TOPIC
```

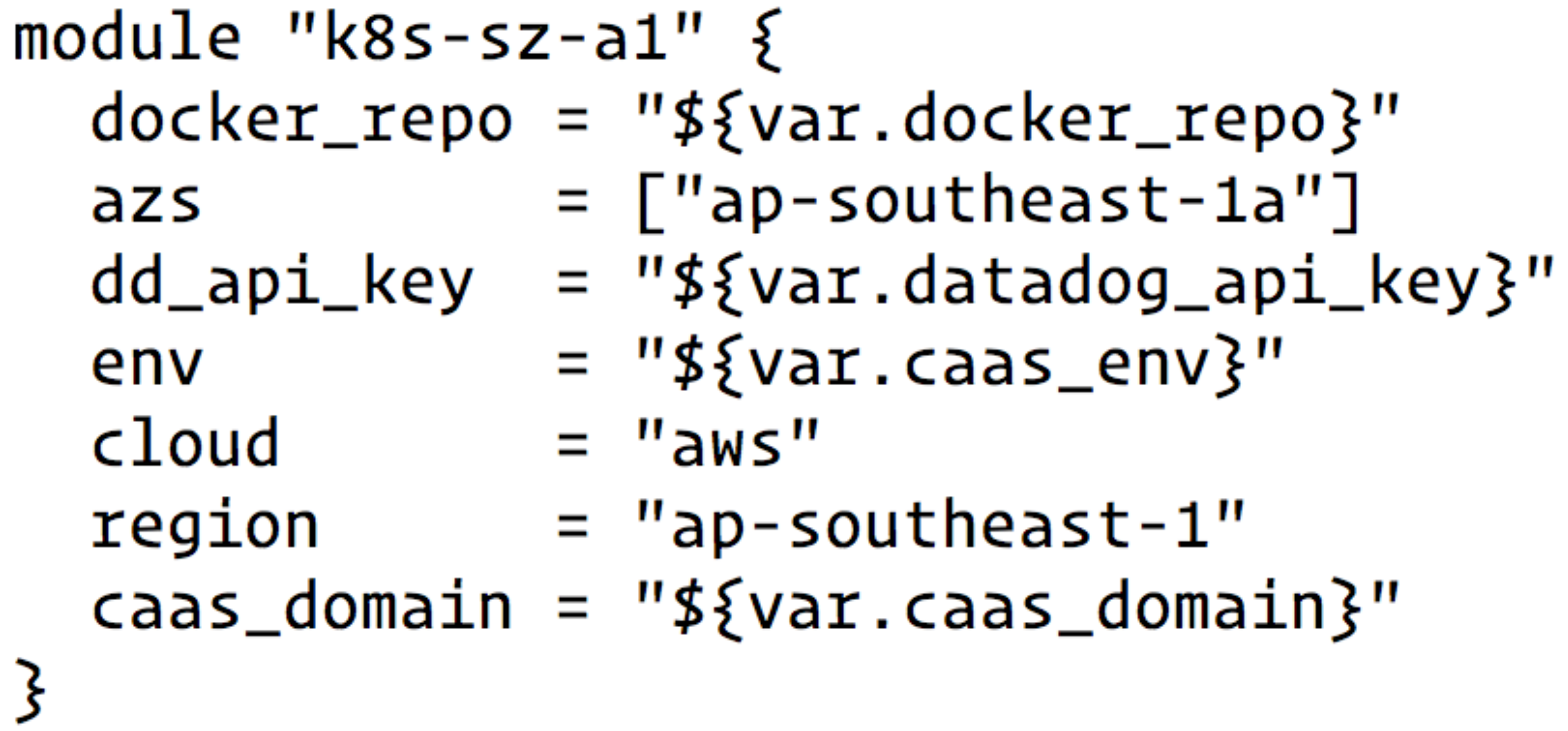
```
REGIONAL SERVICE

MSG ← CONSUMER
  IF MSG IS FOR ME
    OPERATE
  ELSE
    IGNORE
```

Terraform

- → Provisions infrastructure
- → Ties configuration
- → Secrets stored in/looked up from KMS

All it takes to add a new region ->



```
module "k8s-sz-a1" {
  docker_repo = "${var.docker_repo}"
  azs         = ["ap-southeast-1a"]
  dd_api_key  = "${var.datadog_api_key}"
  env         = "${var.caas_env}"
  cloud       = "aws"
  region      = "ap-southeast-1"
  caas_domain = "${var.caas_domain}"
}
```

Billing

- → Using Stripe for payments
- → Subscription API was too limited - no postpay billing
- → Wrote our own billing service
- → Event table stores each change user made on their clusters with associated price per second
- → Job runs next month, sums total from events, bills

Bill item example

→ Cluster created March 15th at a $0.10 price per second

→ Cluster updated March 16th to a $0.20 price per second

→ Cluster deleted March 17th

```
=> [0.10 * (24*60*60)] + [0.20 * (24*60*60].
```

What's next

- → Schema Registry, Connect, Streams, Confluent Control Center
  - → Metrics: Traffic metrics for users via Kafka and showno in our UI
- → Tooling: Internal services for support, on-call, etc.
- → Testing: Integration and UI

Ask Me Anything

Where to go

- → Blog posts and open source from what we're building
- → github.com/travisjeffery
- → medium.com/@travisjeffery
  - → medium.com/@travisjeffery
  - → github.com/travisjeffery
- → Get in touch if you're interested in working on this