Spotify®

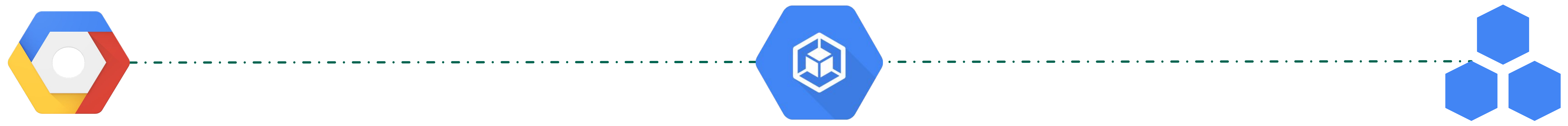
How Spotify Accidentally Deleted All Its Kube Clusters with No User Impact

David Xia @davidxia_

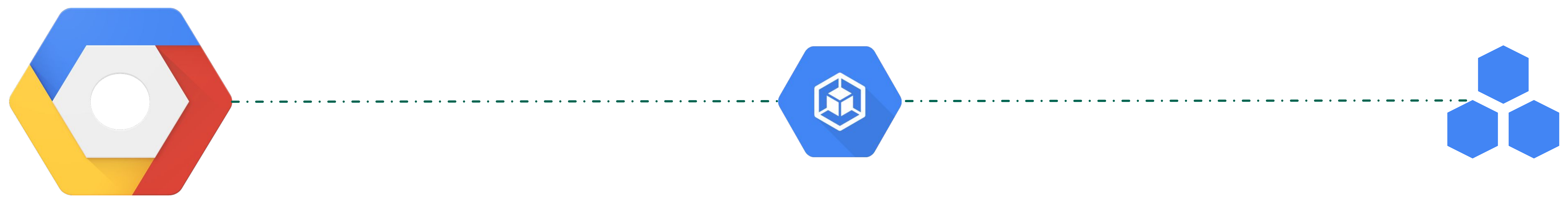
About Myself and Spotify

- infrastructure engineer
- music streaming company with 100M+ subscribers and 200M+ MAU
- 1K+ developers continuously deploying code to 10K+ VMs

Context on Spotify's Compute Environment

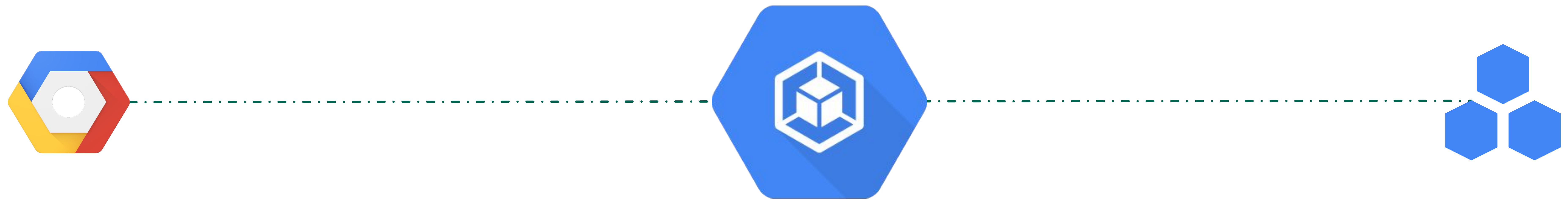


Context on Spotify's Compute Environment

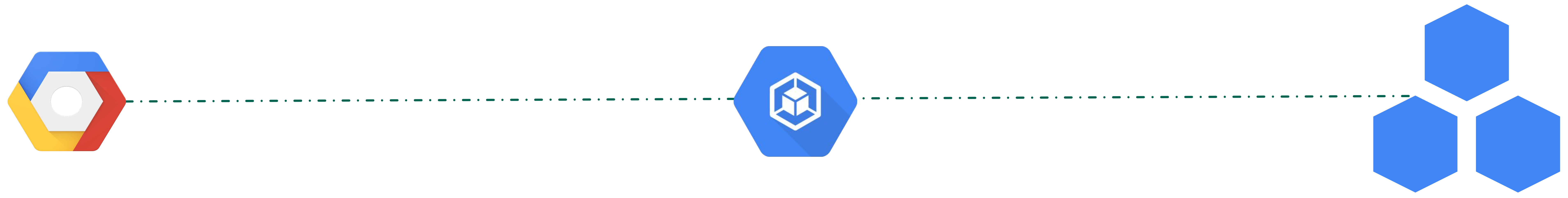


Google Cloud Platform
GCP

Context on Spotify's Compute Environment



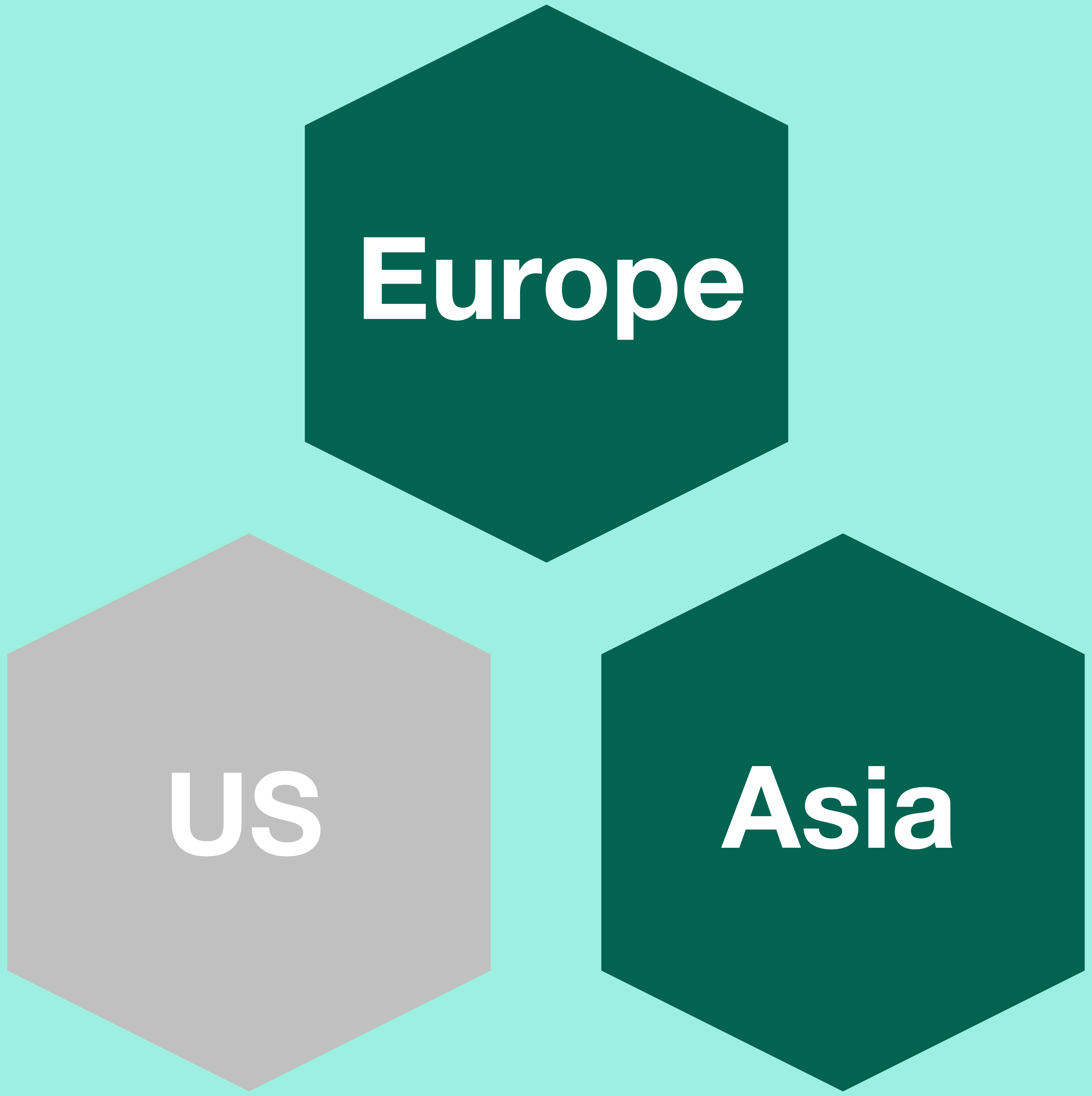
Context on Spotify's Compute Environment



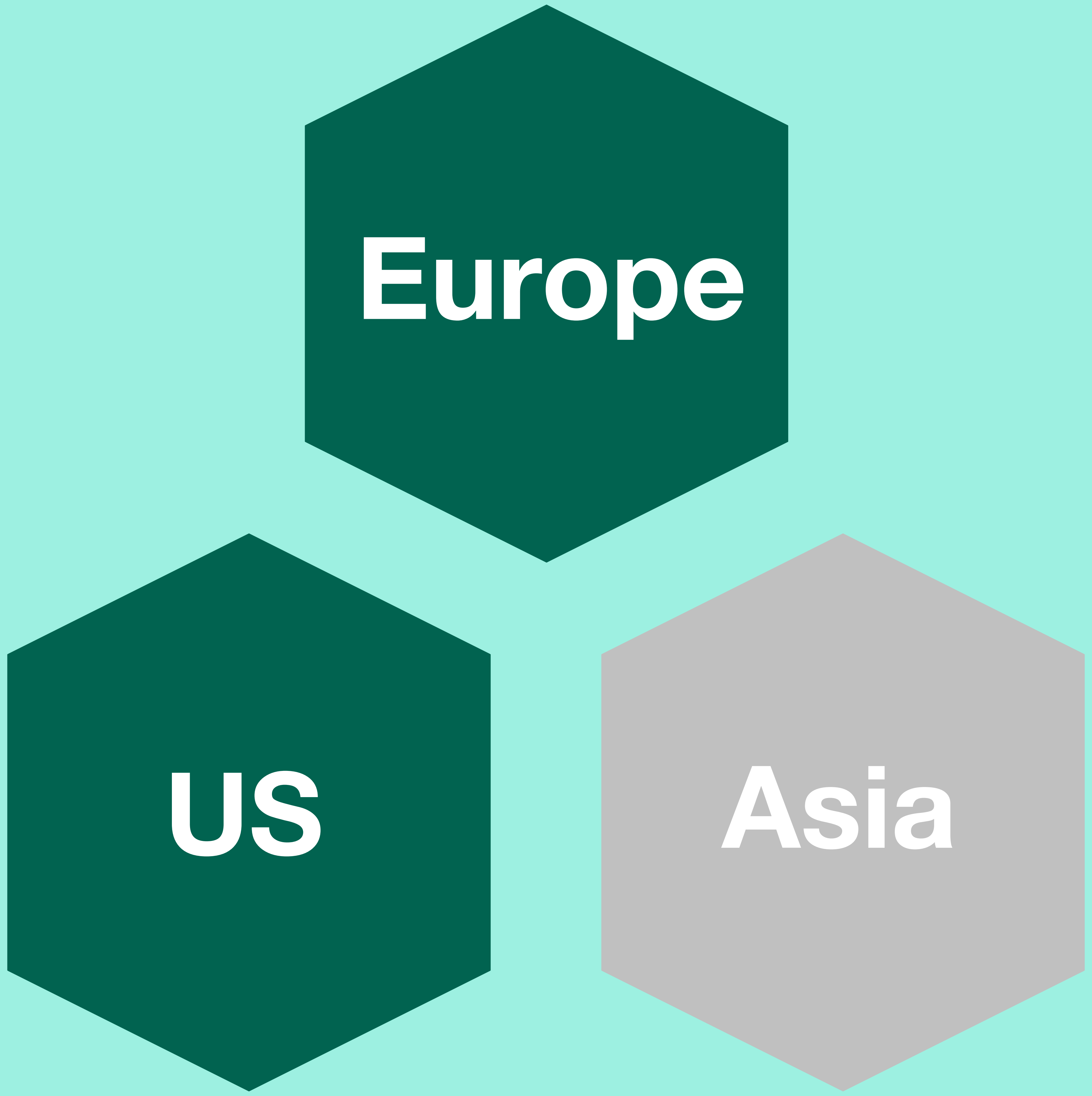
3 production clusters

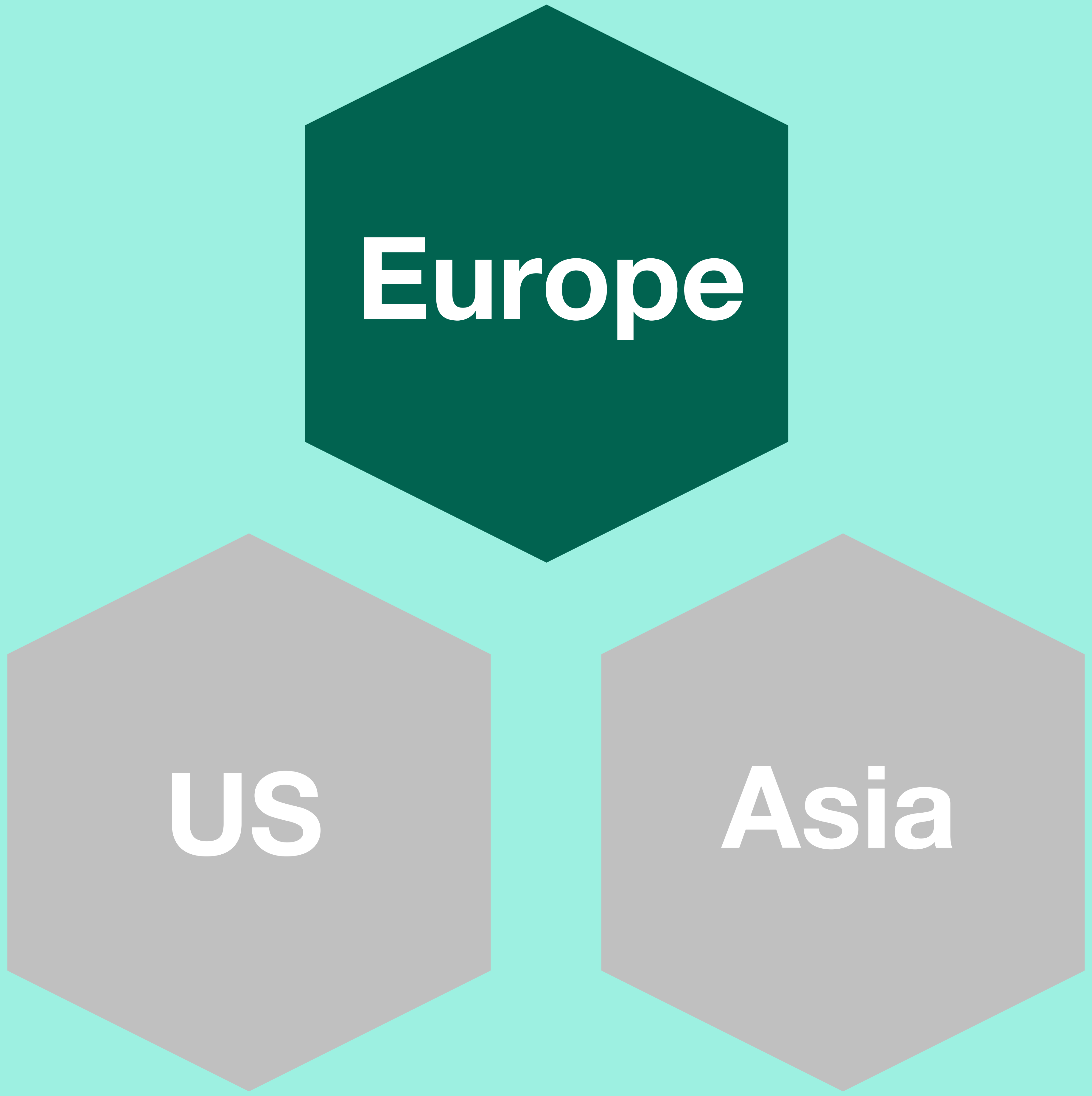
Story Time

Europe
US
Asia

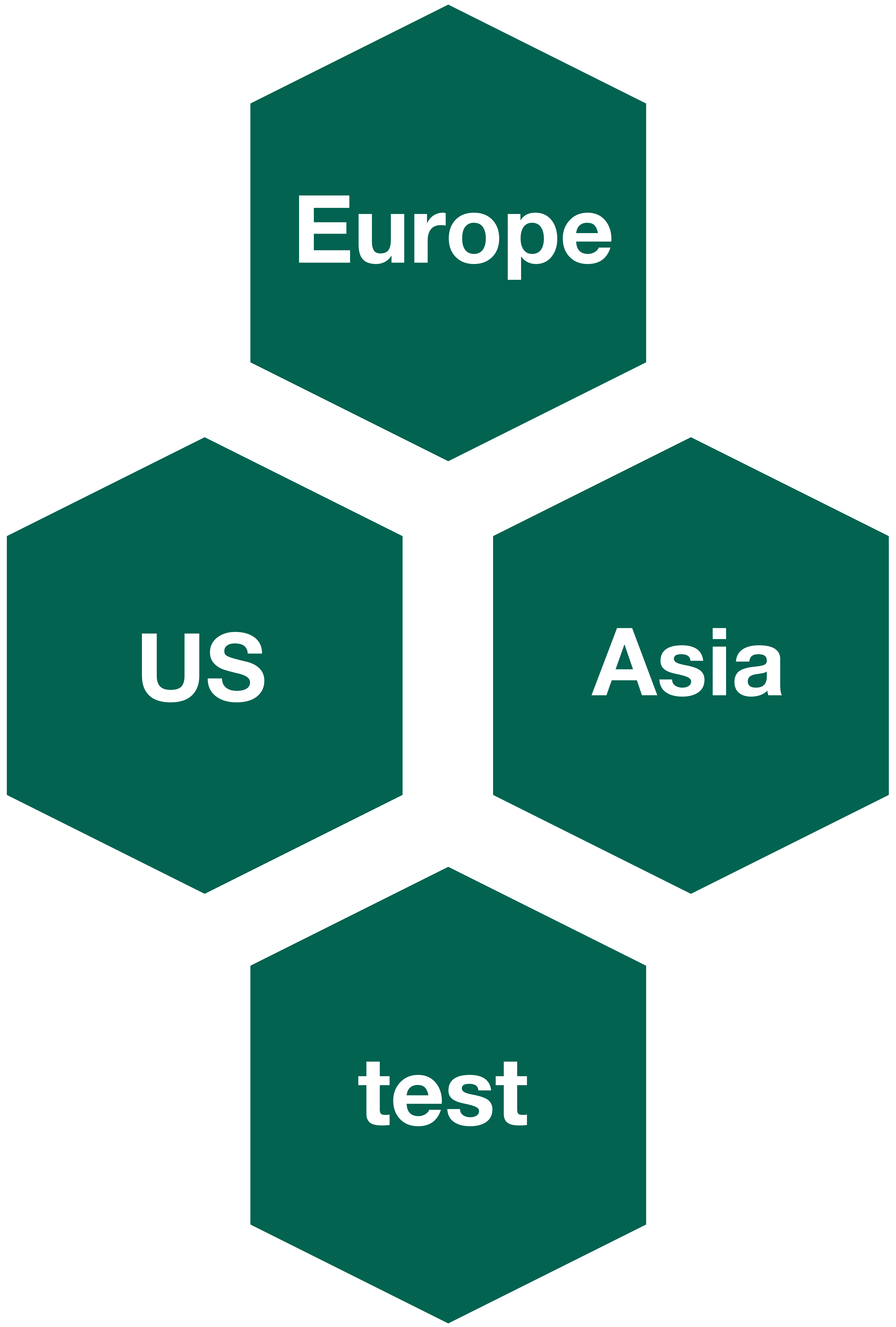

Europe
US
Asia

Europe
US
Asia


Europe
US
Asia

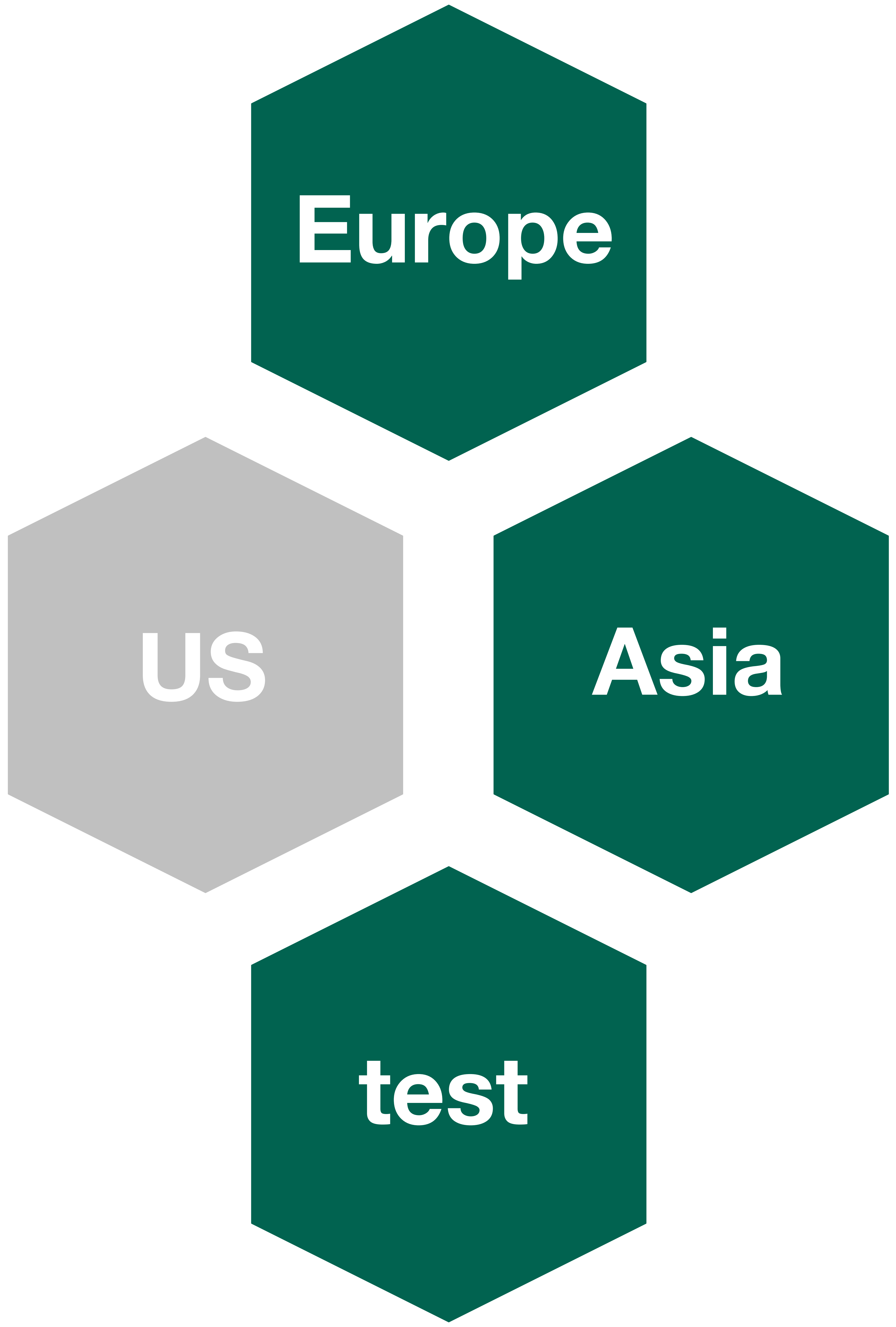

Europe
US
Asia

That Moment When You Realize



That Moment When You Realize

I deleted a 50-node production cluster running dozens of workloads.



How Do I Make It Stop?

You don't.

Cluster Restoration

- took 3.25 hours
- bugs in cluster creation scripts
- incomplete and incorrect documentation
- cluster creation process wasn't resumable, all or nothing

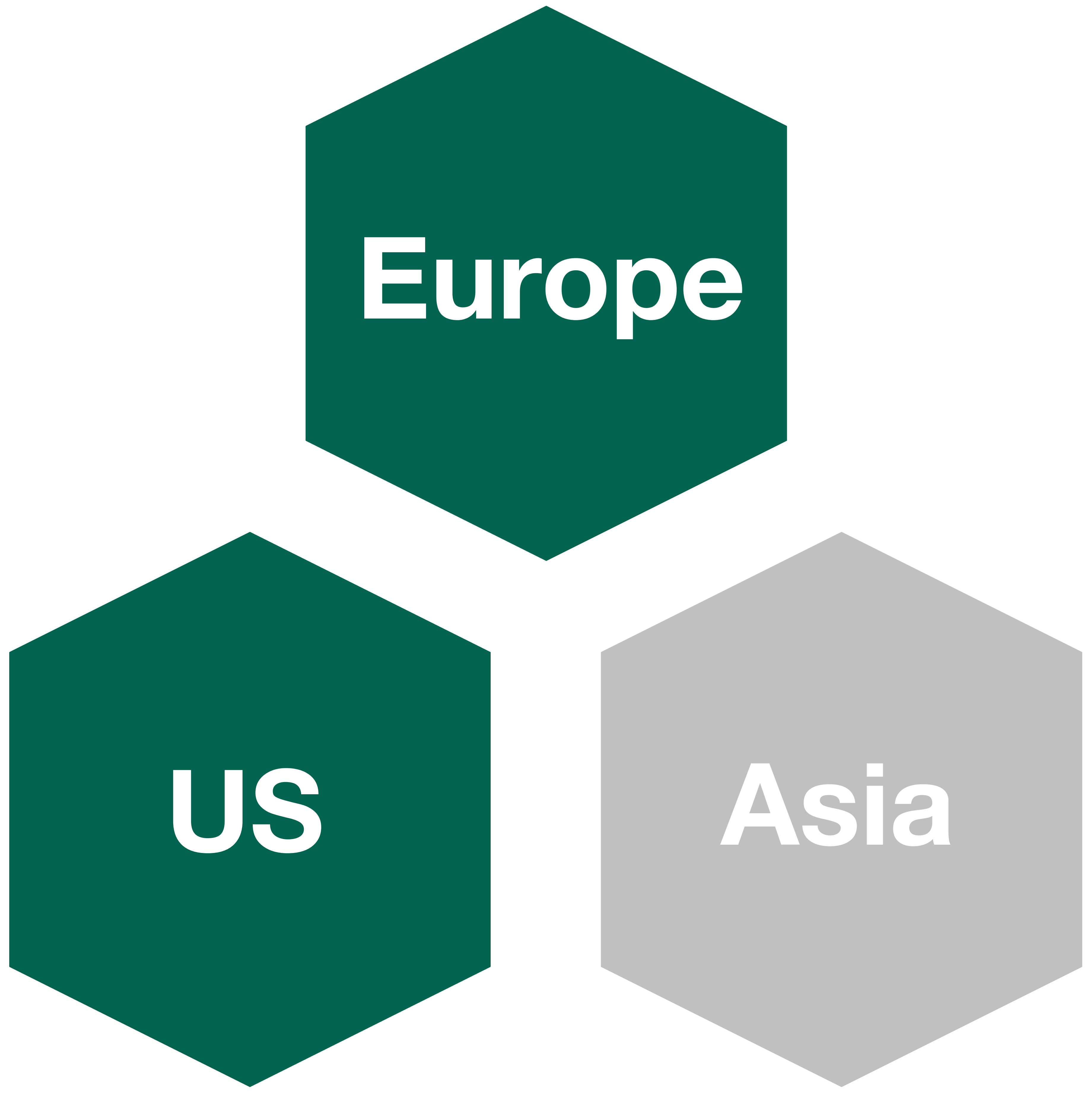
A Month Later

- trying to prevent accidental cluster deletions by codifying them
- unknowingly modified global state during review builds
- two PRs merged out of order

Europe
US
Asia

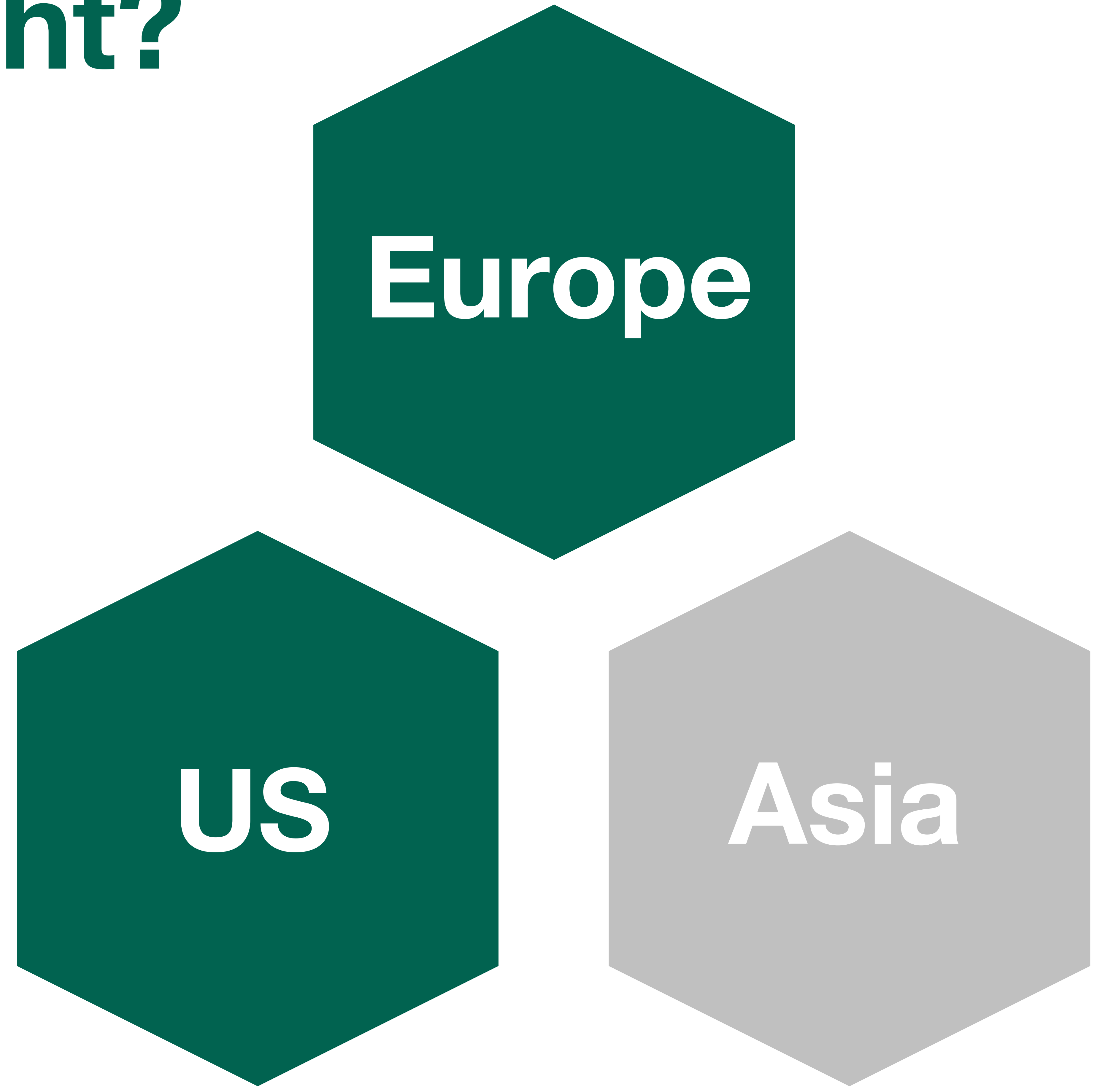
A Month Later

- trying to prevent accidental cluster deletions by codifying them
- unknowingly modified global state during review builds
- two PRs merged out of order



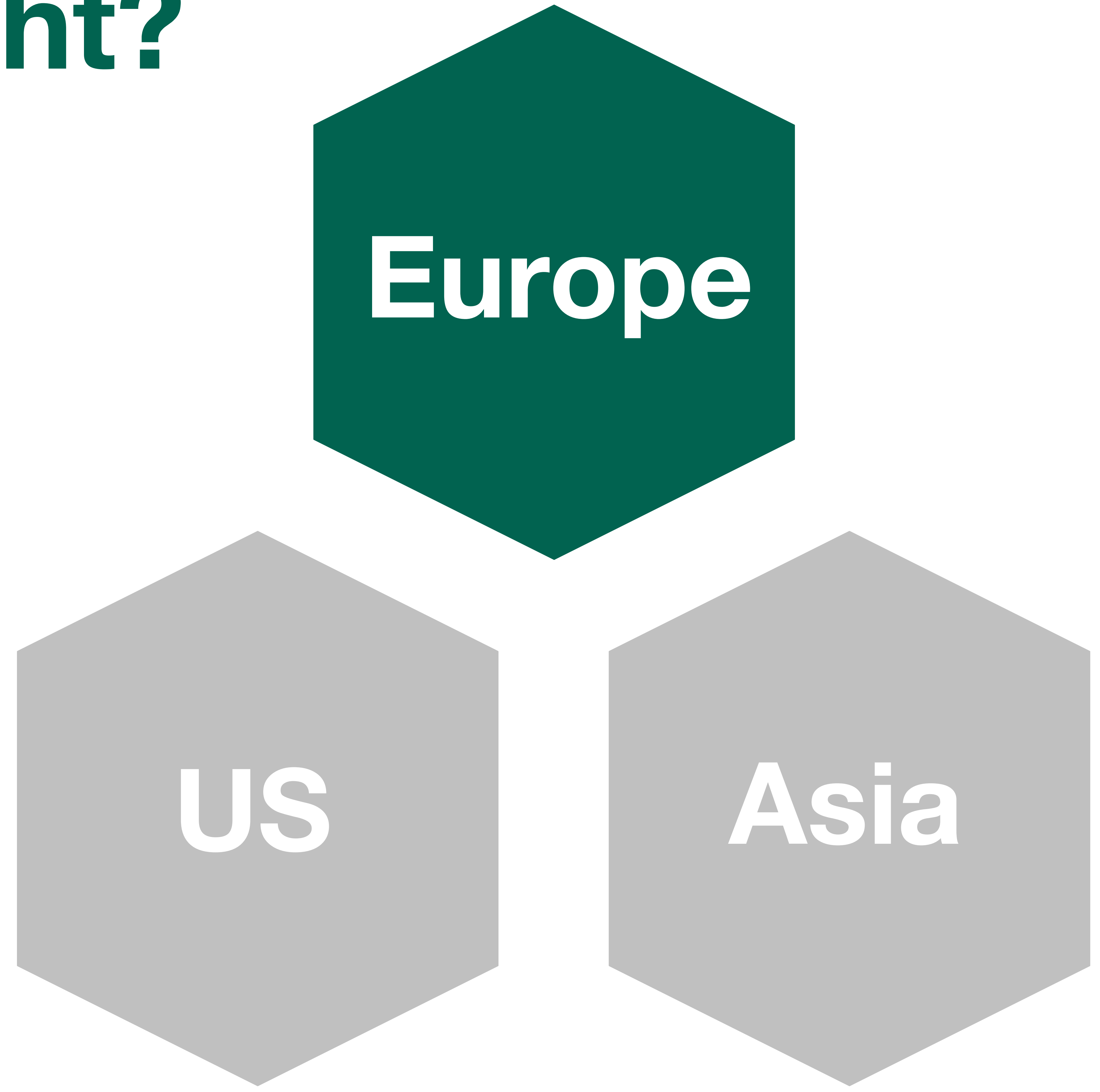
Can't Get Any Worse, Right?

- we try to recreate the cluster by merging the remaining PR
- cluster creation fails from lack of permissions
- we grant enough **but different** permissions to make it work
- caused Terraform's view of the clusters to change



Can't Get Any Worse, Right?

- we try to recreate the cluster by merging the remaining PR
- cluster creation fails from lack of permissions
- we grant enough **but different** permissions to make it work
- caused Terraform's view of the clusters to change



Developer Impact

- one team had to create more non-K8s VMs
- my team had to update all the places we had hardcorded the old master IP
- everyone had to refresh cluster credentials

End User Impact

What We Did Right

- we planned for failure
- we migrated large scale, complex infrastructure **gradually**
- we have a culture of learning

How Did We Plan for Failure?

1. we recommended teams only migrate services partially to K8s
2. the way we registered services running on K8s
3. resulting failover to non-K8s instances

Partial K8s Migration on Per-Service Level

- K8s usage at Spotify was marked as beta at the time
- we recommended teams only migrate some but not all of each service's instances to K8s
- we continue work on integrations, reliability, managing multiple clusters

The Saving Grace of Registering Services the non-K8s Way

- our internal service discovery system uses Pod IPs
- we don't use the K8s Service IP
- we polls Services' Endpoints and update service discovery
- our team was paged to make service discovery no longer poll deleted cluster

Failover to Non-K8s Instances

- service discovery system was restarted
- K8s Pods removed from service discovery
- clients only got a list of non-K8s instances

Best Practices

- backed up our clusters
- codified our infrastructure
- performed disaster recovery tests
- made team members practice disaster scenarios

Backed Up Our Clusters

- our cluster backups were essential
- we had already tested restoring from these
- if you have never restored from backups, you don't have backups

Codified Our Infrastructure

- introduced new tools gradually
- standardized the workflow and change management of infra code
- added linters and validators
- added the output of the dry run as a comment to the pull request
- required status checks to pass before merging
- required feature branch to be up to date
- required approving reviews
- failed review builds if certain keywords in the dry run like "destroy"

Performed Disaster Recovery Tests

- disasters will happen whether you plan for them or not, so plan for them
- scheduled them in advance
- announced widely to operators and users
- tested different failure conditions
- recorded and fixed issues quickly

Practice Makes Perfect

- it took me 3.25 hrs to restore cluster I deleted along with all its integrations
- second cluster deletion incident lasted from 8PM to 5AM
- now we can restore larger clusters in 1 hour

Culture of Learning, Not Blame

What We Did Right

- we planned for failure
- we are migrating large scale, complex infrastructure **gradually**
- we have a culture of learning

Next Steps for K8s at Spotify

- told service owners their services can now be entirely on K8s
- manage configuration and workload distribution across many clusters
- create redundancy by deploying services to multiple clusters in a region

Thank you!

Spotify®

spotifyjobs.com
David Xia @davidxia_