AWS re:Invent

ARC213-R

Architecture patterns for multi-region active-active

Girish Dilip Patil
Senior Solutions Architect
Amazon Web Services

Jonathan Dion
Senior Technical Evangelist
Amazon Web Services

Thomas Jackson
Head of Core & Data Infrastructure
Wish

Agenda

Why do you need it?

Design principles

Foundational pillars

A customer’s journey: Wish’s path to multi-region

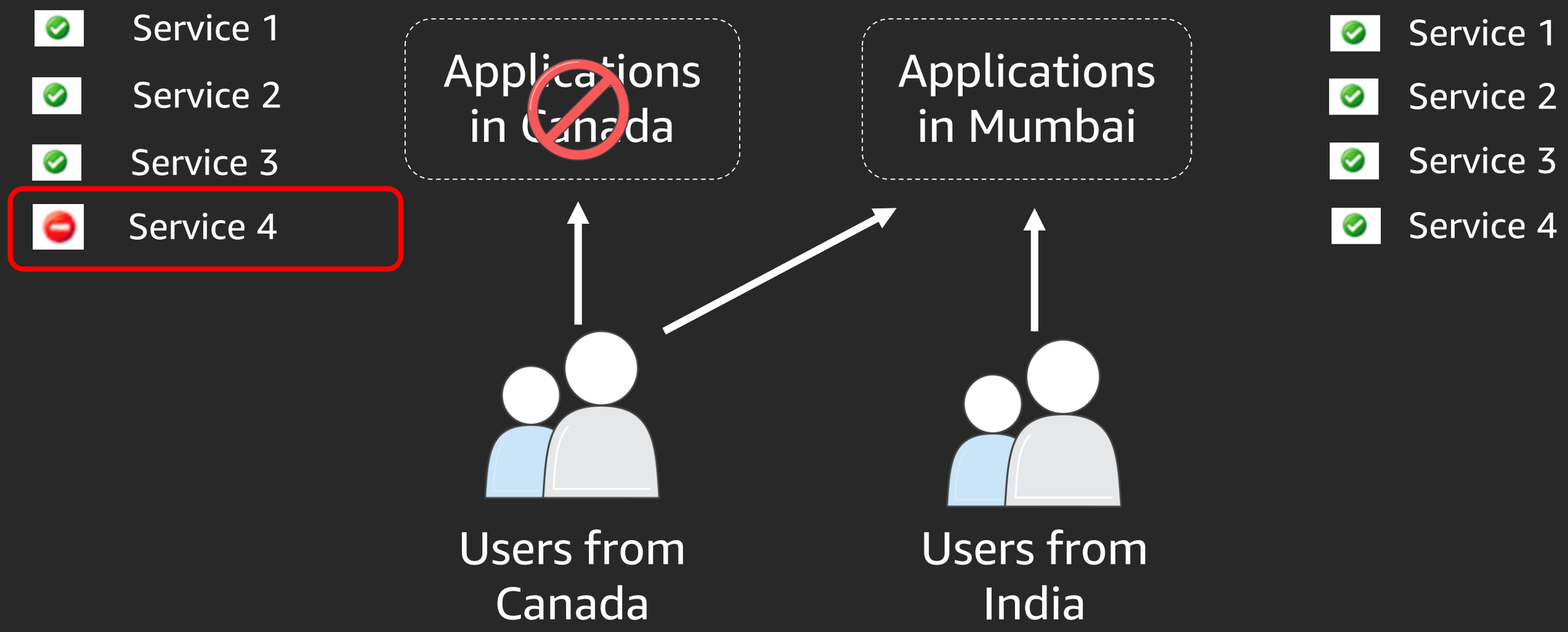
Why do you need a multi-region active-active architecture?

Everything fails, all the time

Werner Vogels

CTO Amazon.com

Guarding against failure of your applications in one region



We need to reduce blast radius of adverse events

What is the problem with conventional DR solutions?

DR environments that aren’t used

1. Fall out of sync, eventually

2. Waste money

Advantage of multi-region active-active architecture

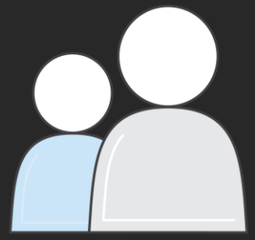
Serving geographically distributed customer base

Canada

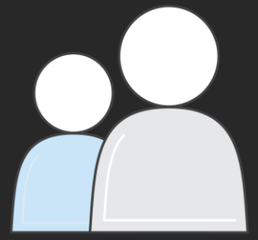
Mumbai

Ohio

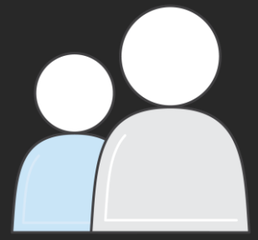
Sydney



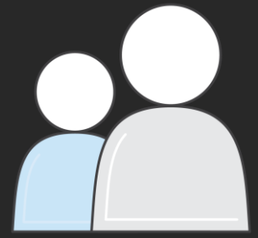
Users from Canada



Users from India



Users from USA



Users from Australia

What are the design principles for a multi-region active-active architecture?

Tolerance for network partitioning

1. Any problem in one region should not lead to failure of applications in another
2. You should aim for regional independence for request serving
   - Minimal blocking API or database calls from one region to another
   - Graceful degradation of service in case network connectivity is lost

Minimal data replication requirements

- Does all data need to be replicated?
- If yes, does it need to be replicated synchronously?
- Does all data need to be replicated continuously?

Data classification

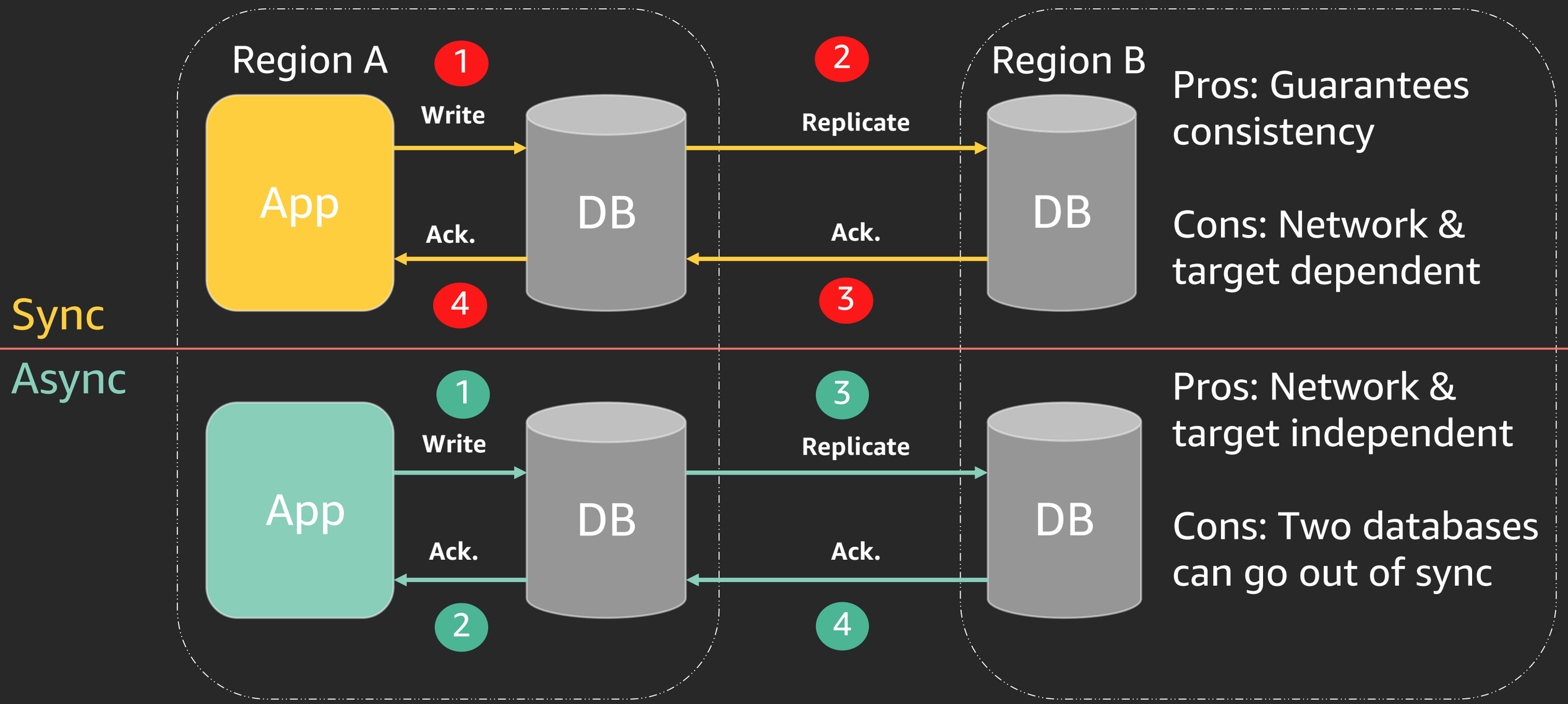
Example: E-commerce workload

Transactions	Catalogue information	Events, objects	Server logs
Purchase record	Product details	Click stream	HTTPS logs

Low volume,
but highly critical

High volume,
but less critical

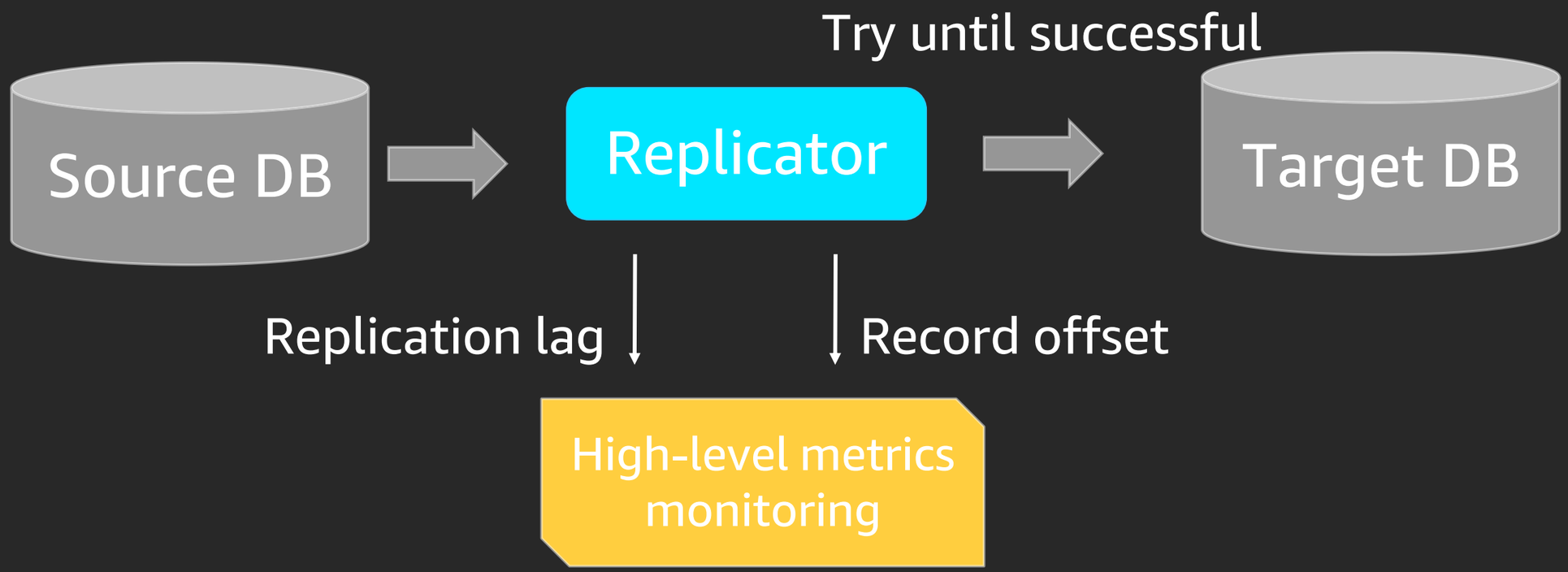
Synchronous vs. asynchronous replication modes



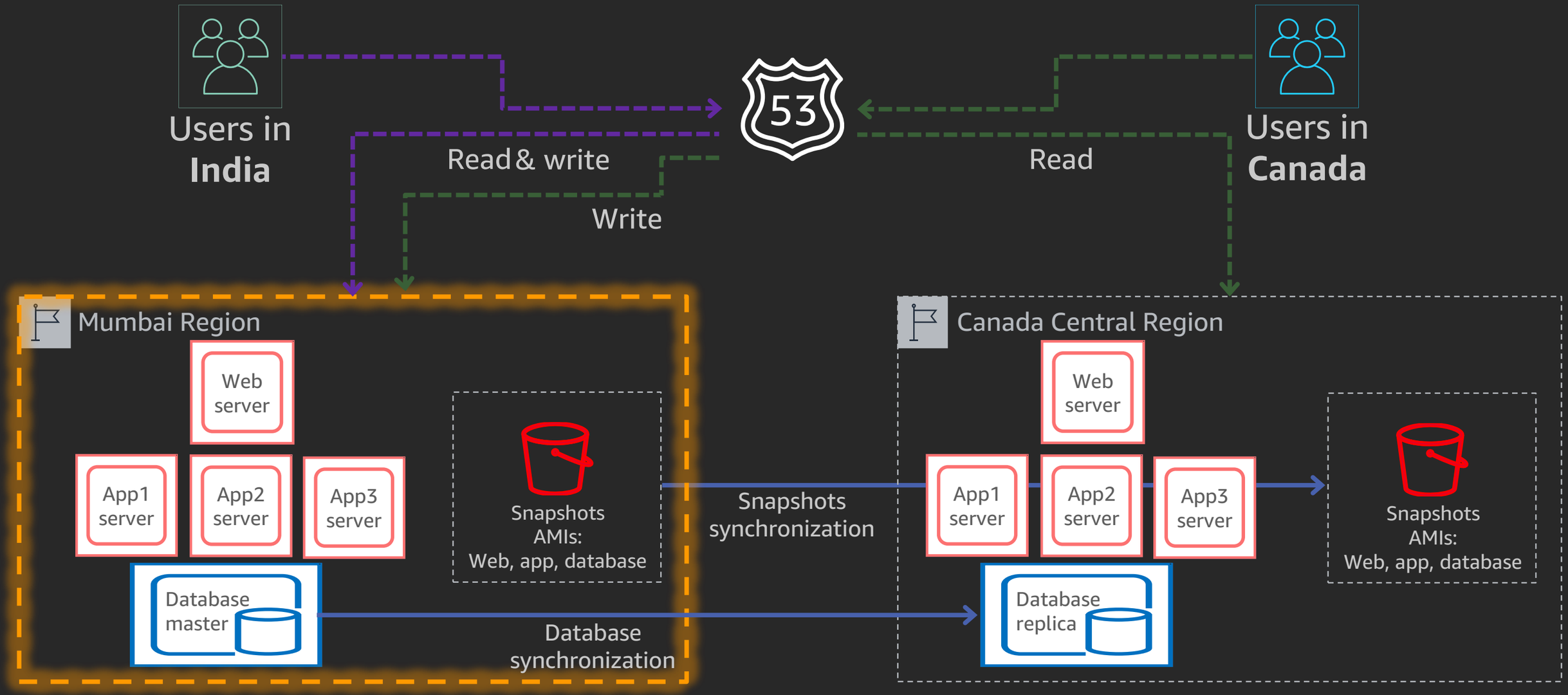
Ideal replication system

Should

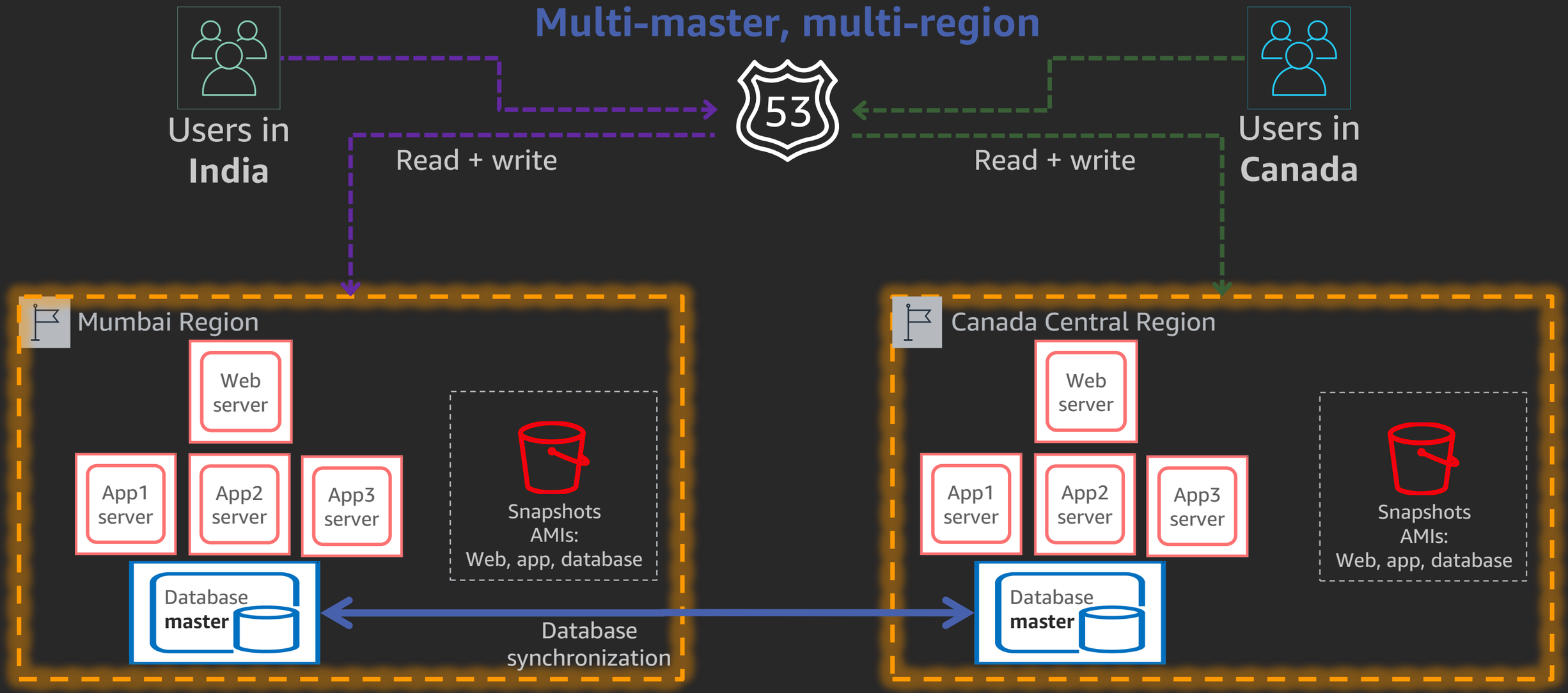
- Report replication lag
- Report record offset
- Be able to retry replication of failed records



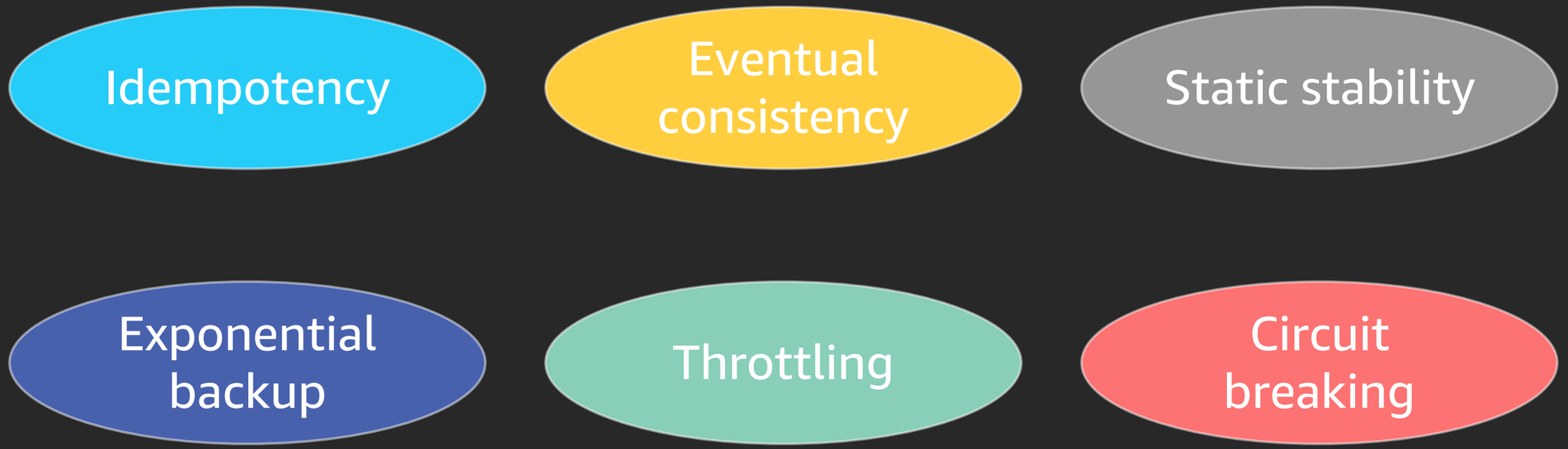
Pattern 1: Read local, write global



Pattern 2: Read local, write local



Distributed system design best practices

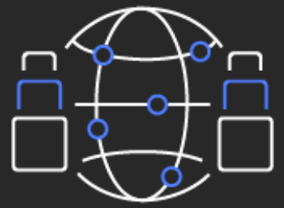


AWS Well-Architected Framework
Reliability Pillar whitepaper
http://bit.ly/31u0UbP

What are the foundational pillars for a multi-region active-active design?

Foundational pillars of a multi-region active-active architecture

High availability



Data replication

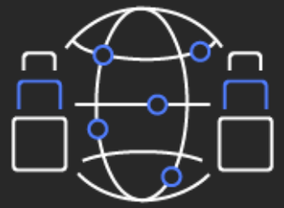
Networking

Traffic routing

Management

Foundational pillars of a multi-region active-active architecture

High availability



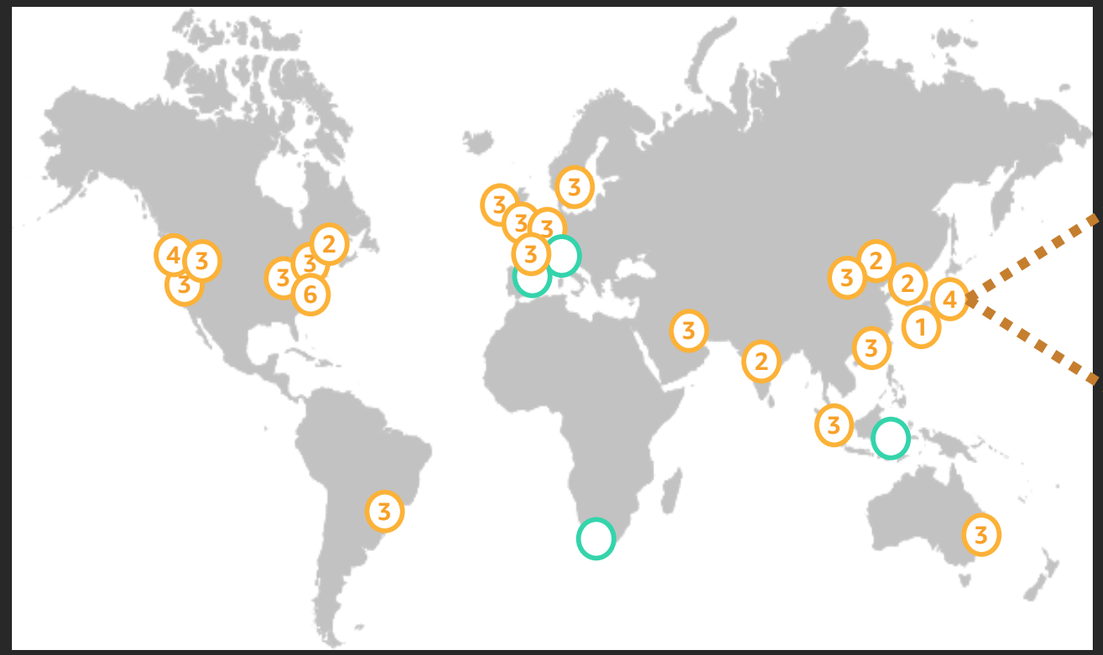
Data replication

Networking

Traffic routing

Management

Region topology

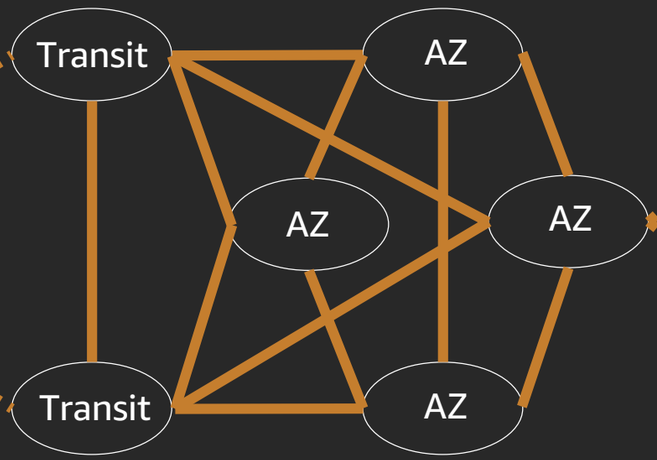


Region & number of Availability Zones

Announced Regions

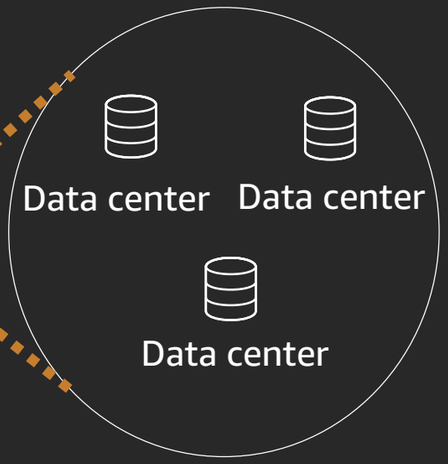
Cape Town, Jakarta, Milan, and Spain

AWS Region



A Region is a physical location in the world where we have multiple **Availability Zones**

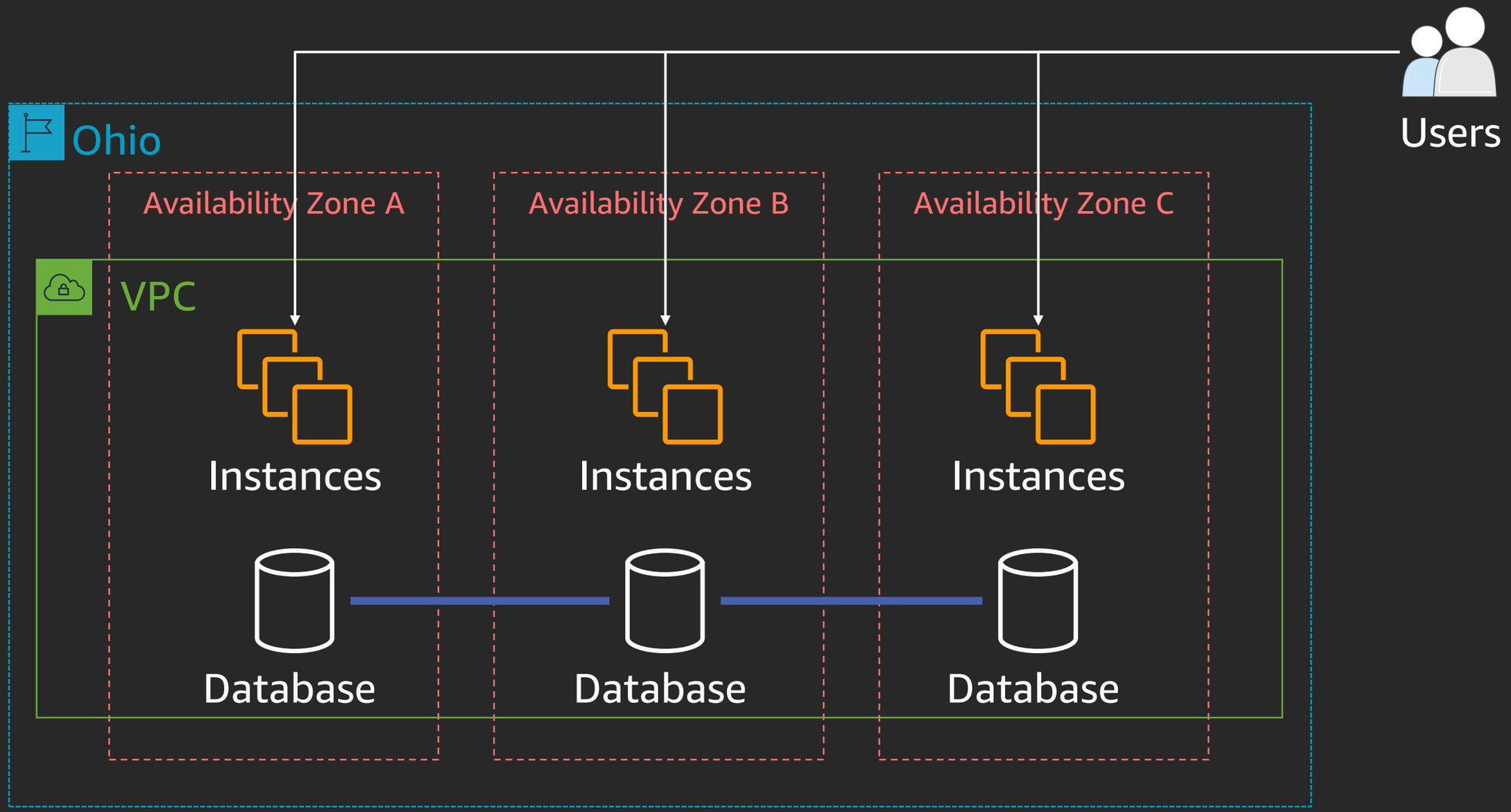
AWS Availability Zone (AZ)



Availability Zones consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities

Single-region high-availability approach

Leverage multiple Availability Zones (AZs)



Examples of services Multi-AZ

By default Multi-AZ

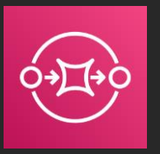
Amazon S3

Amazon EFS

Amazon DynamoDB

Amazon QLDB

Amazon Kinesis



Amazon SQS

Amazon ElasticSearch Service

Amazon RDS

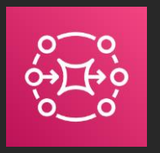
Amazon Aurora

Amazon Neptune

Amazon DocumentDB

Amazon ElastiCache

Amazon Managed Streaming for Kafka



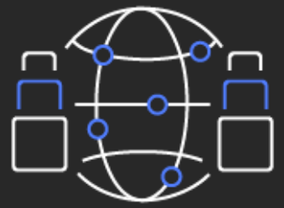
Amazon MQ

File stores | **Search** | **RDBMS** | **No-SQL & in-memory caching** | **Streams & queues**

Configurable for Multi-AZ deployment

Foundational pillars of a multi-region active-active architecture

High availability



Data replication

Networking

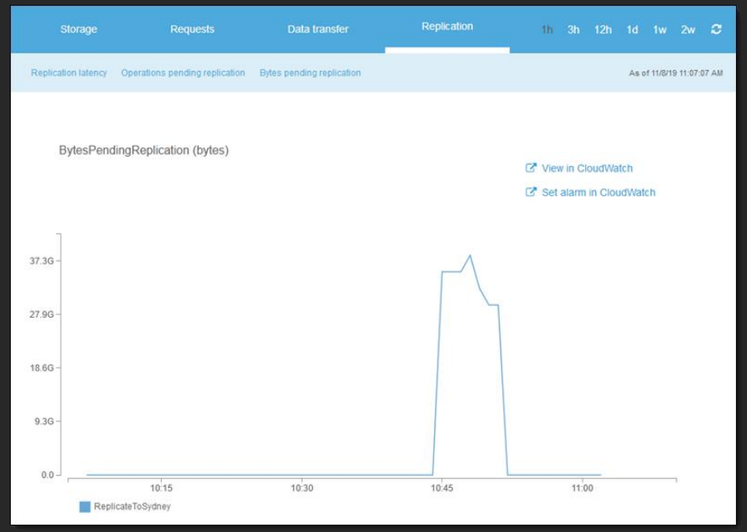
Traffic routing

Management

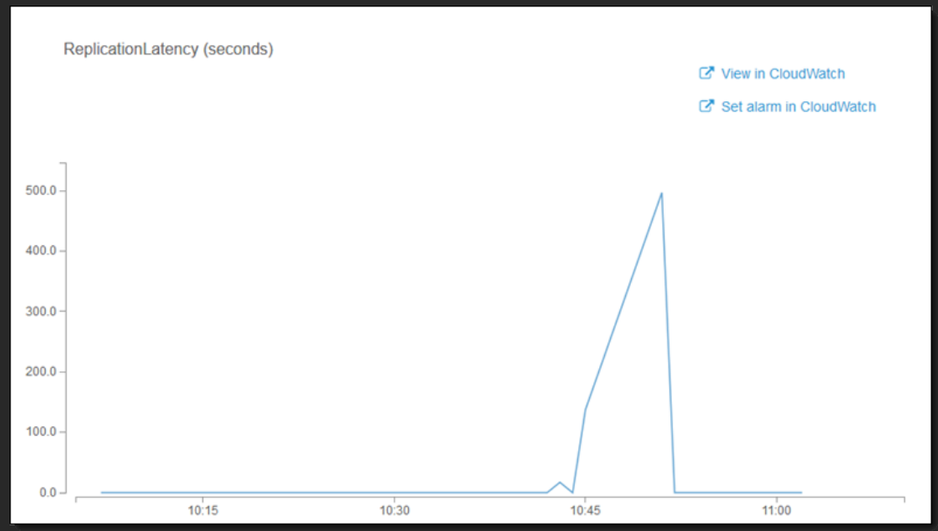
S3 cross-region replication

- Automatically replicate data to any other AWS Region
- Replicate by object, bucket, or prefix
- Replication time control

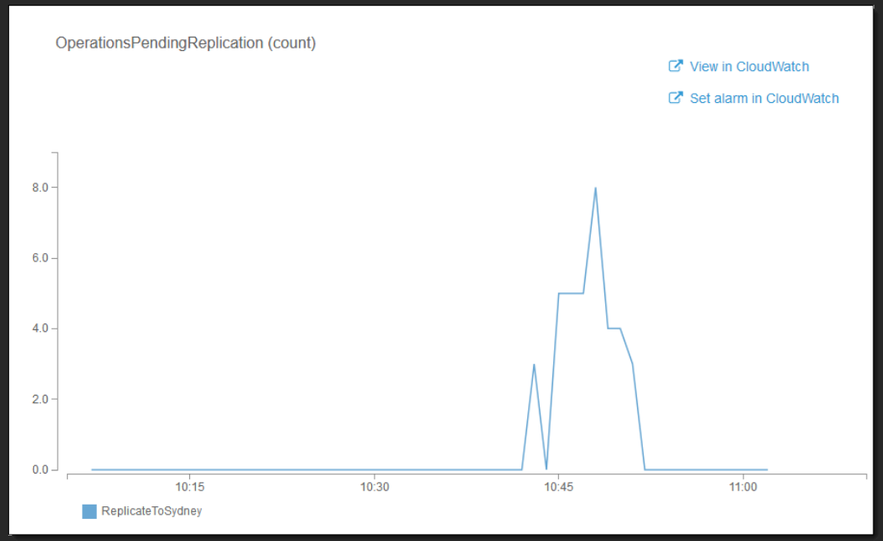
S3 replication metrics



BytesPendingReplication

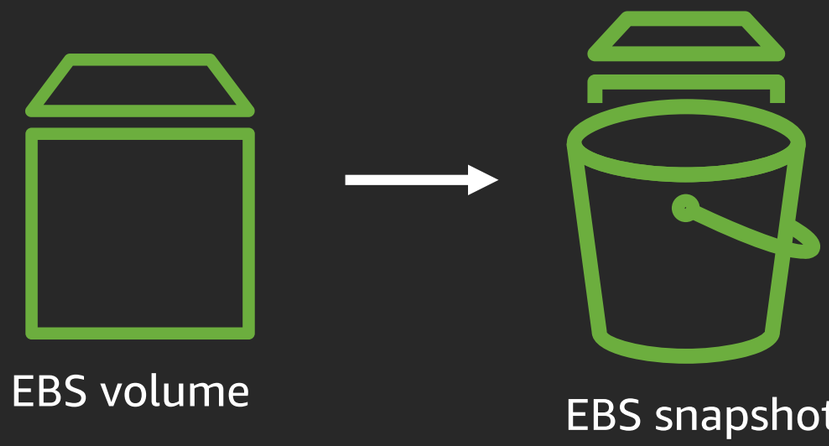


ReplicationLatency



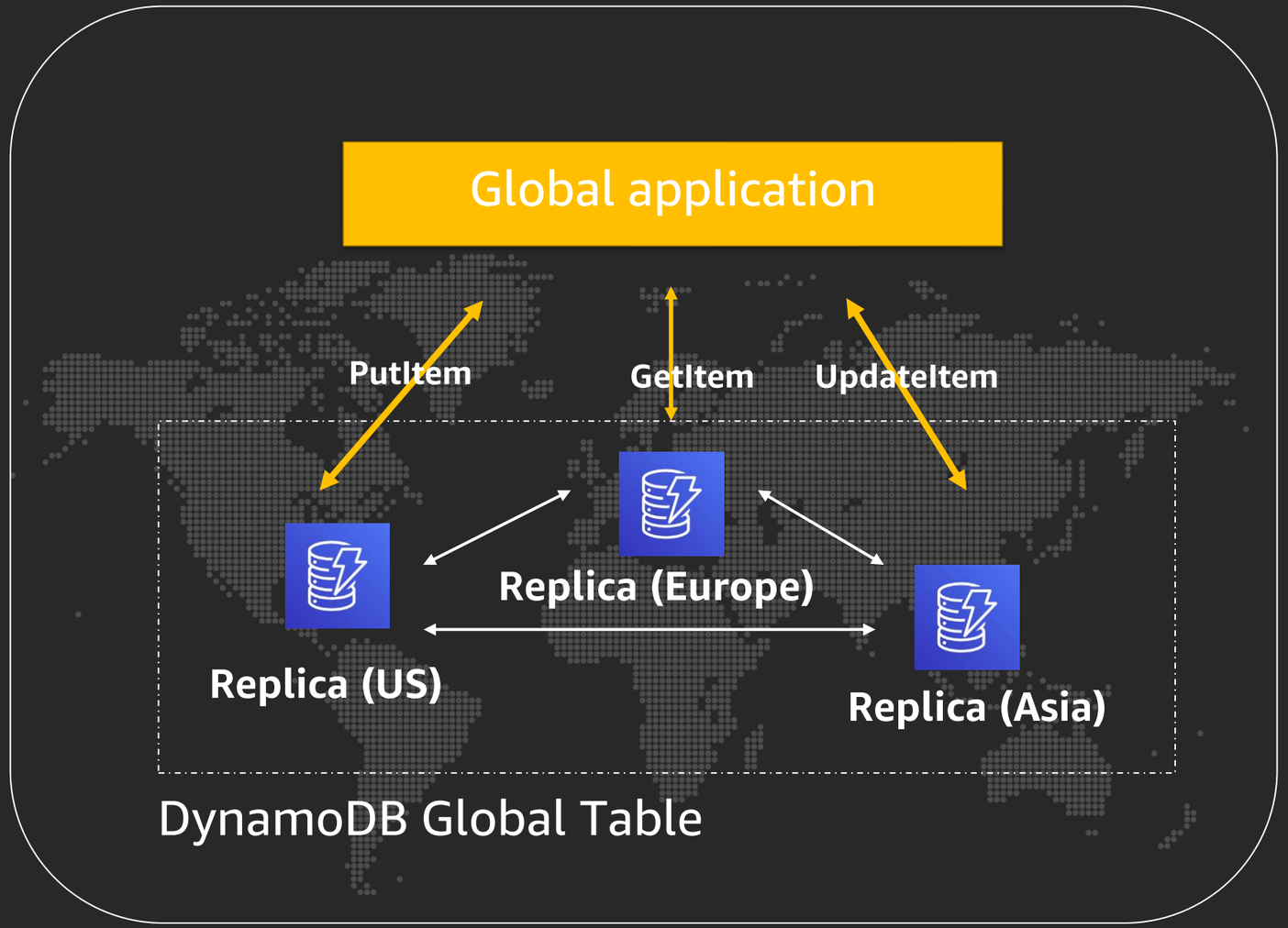
OperationsPendingCount

Amazon Elastic Block Store snapshots



- Point-in-time backup
- Stored in S3
- Incremental
- Cross-region copy

DynamoDB Global Tables

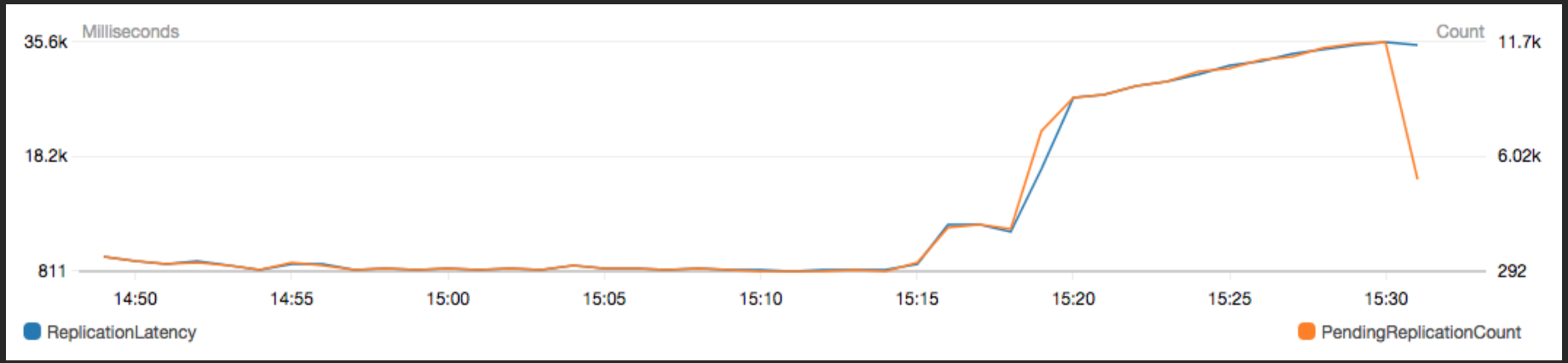


- Replicates table across multiple regions
- Read & write any items in any region
- Eventual convergence
- Conflicting updates resolved with “last write wins”

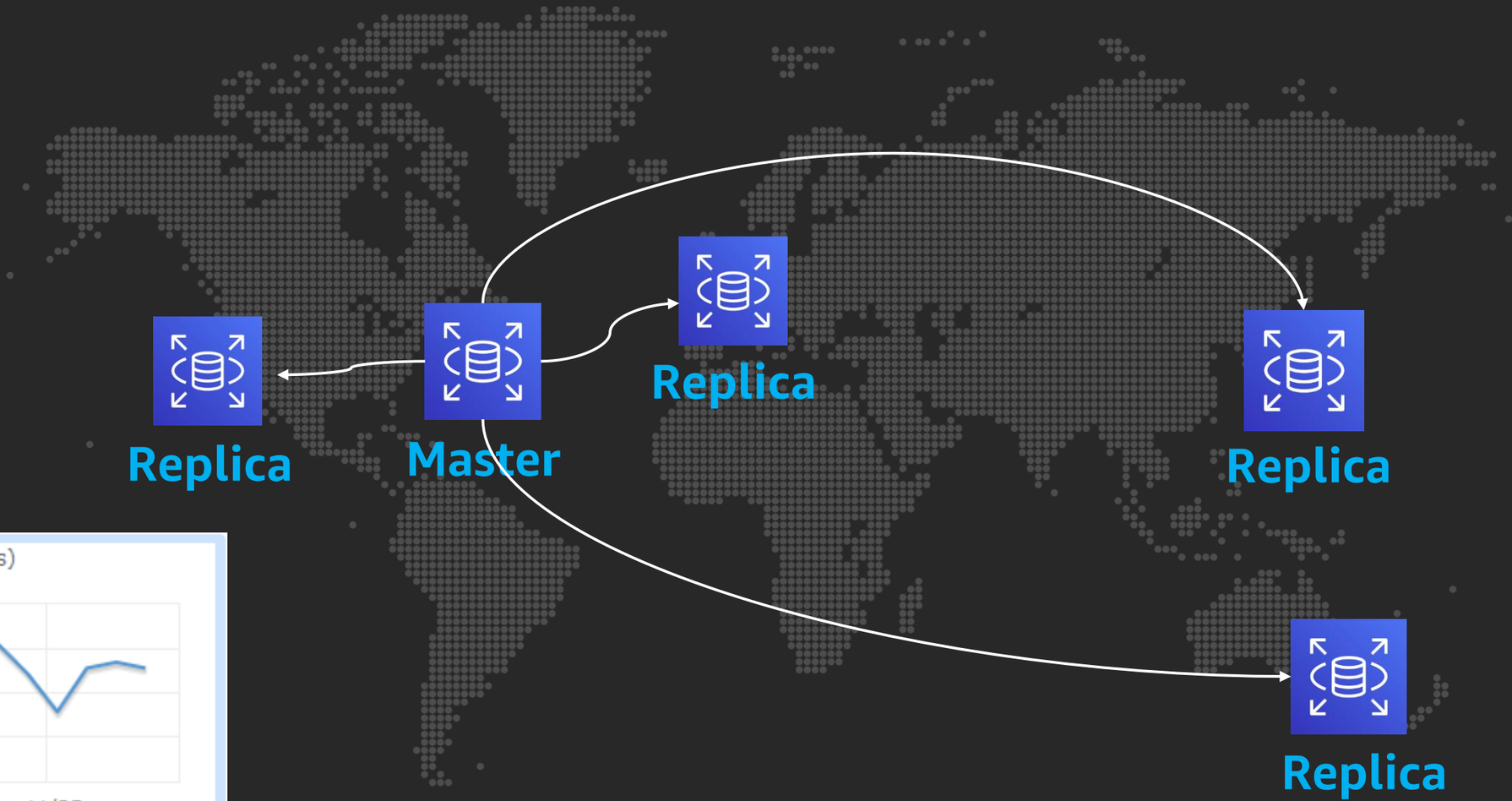
Global Tables management

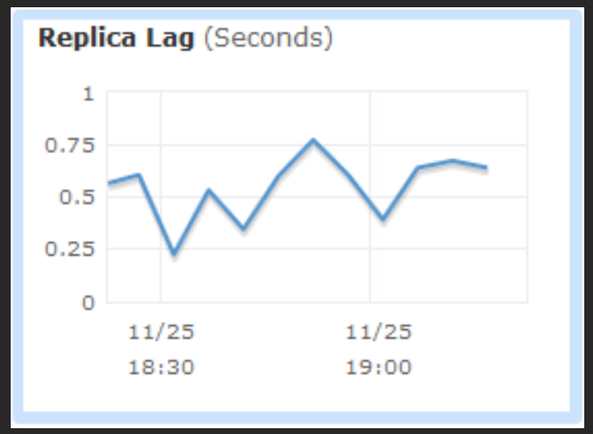
Amazon CloudWatch metrics

- **ReplicationLatency**: Elapsed time of propagating
- **PendingReplicationCount**: Number of items written to one replica but not propagated to other regions

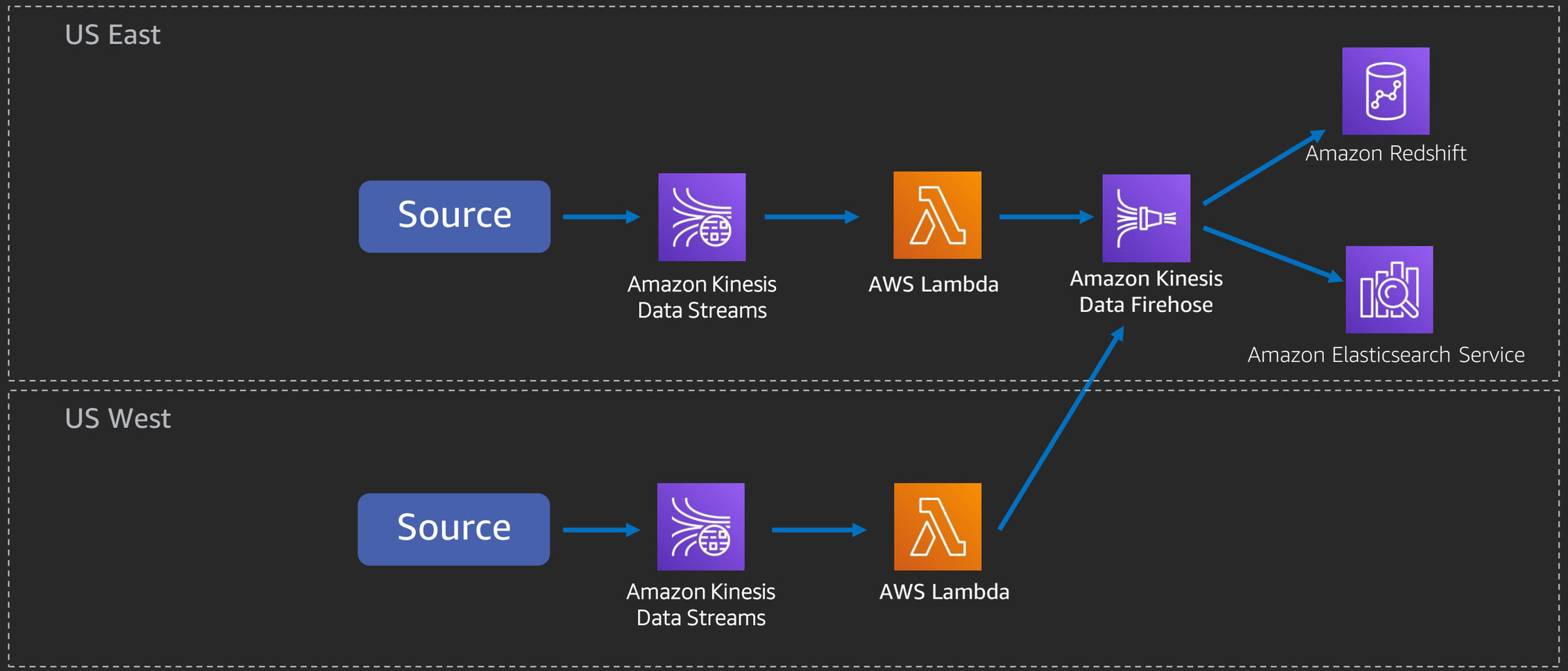


Amazon RDS cross-region replication



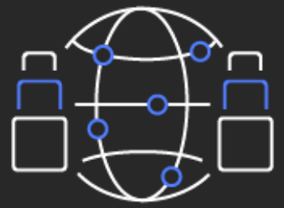


Multi-region consolidation of analytics data



Foundational pillars of a multi-region active-active architecture

High availability



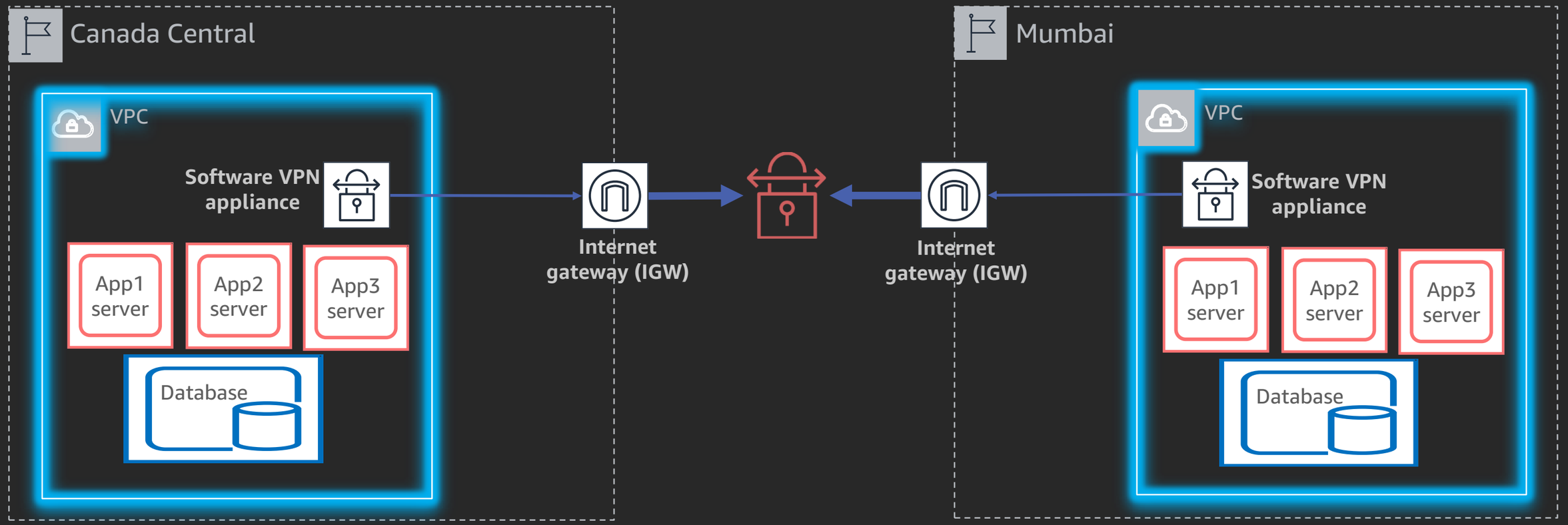
Data replication

Networking

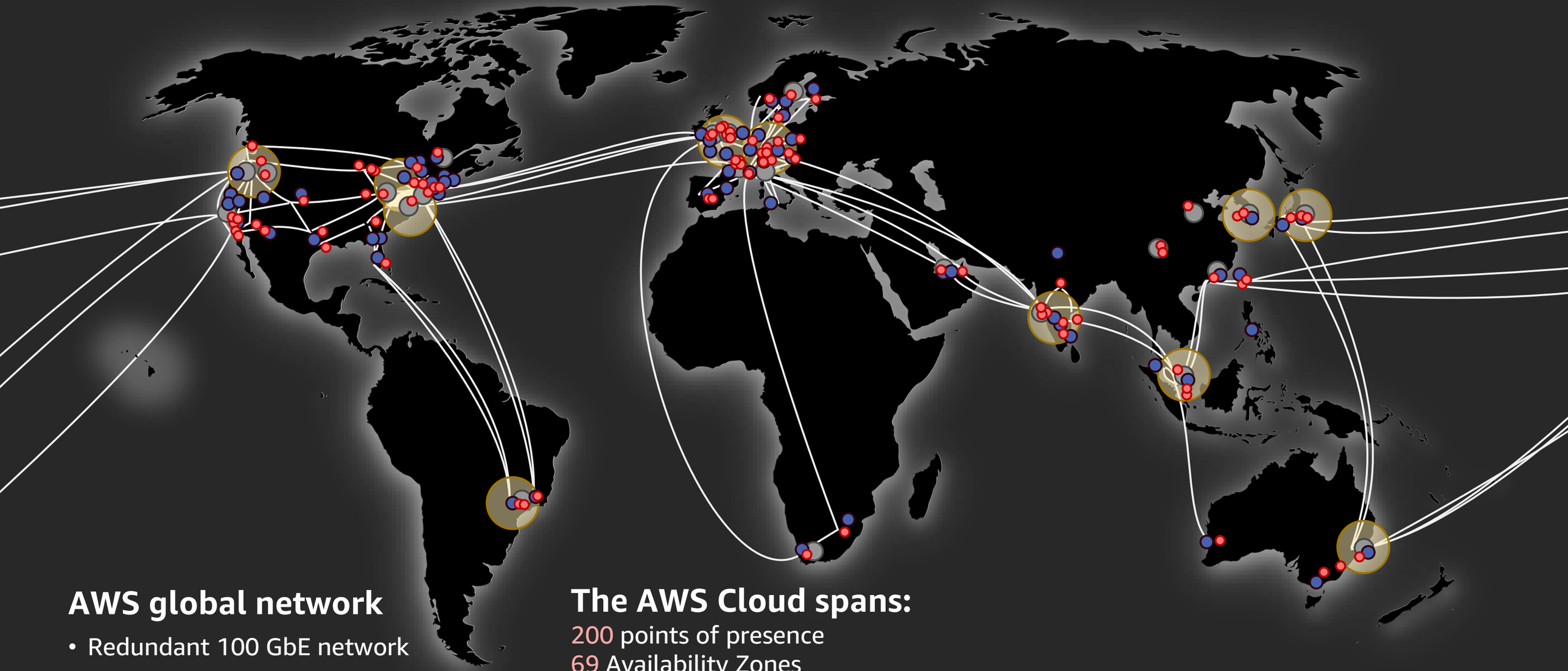
Traffic routing

Management

Amazon VPC and software VPN



AWS Global Infrastructure



AWS global network

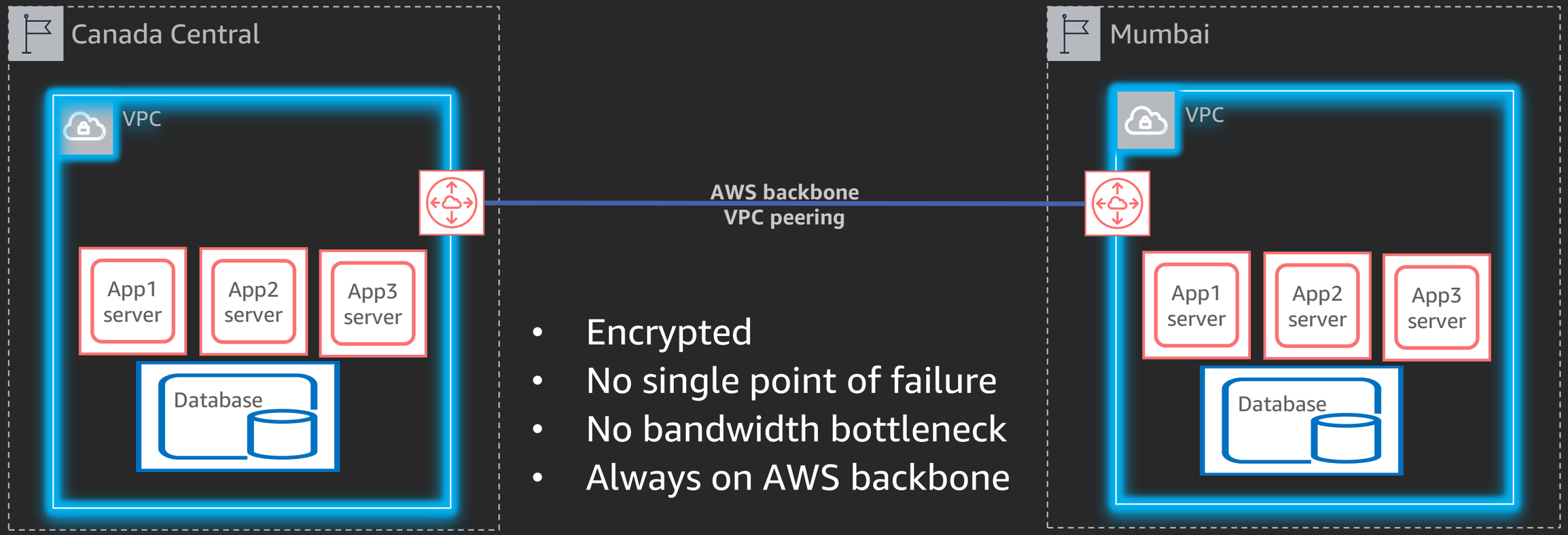
- Redundant 100 GbE network
- Private network capacity between all AWS Regions, except China

The AWS Cloud spans:

200 points of presence
69 Availability Zones
22 geographic Regions around the world

*With announced plans for 13 more Availability Zones and three more Regions in Cape Town, Jakarta, Milan, and Spain

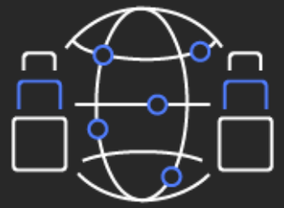
Inter-Region VPC peering



- Encrypted
- No single point of failure
- No bandwidth bottleneck
- Always on AWS backbone

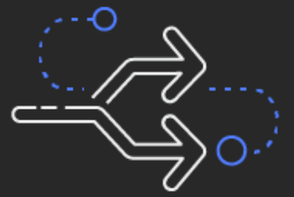
Foundational pillars of a multi-region active-active architecture

High availability



Data replication

Networking

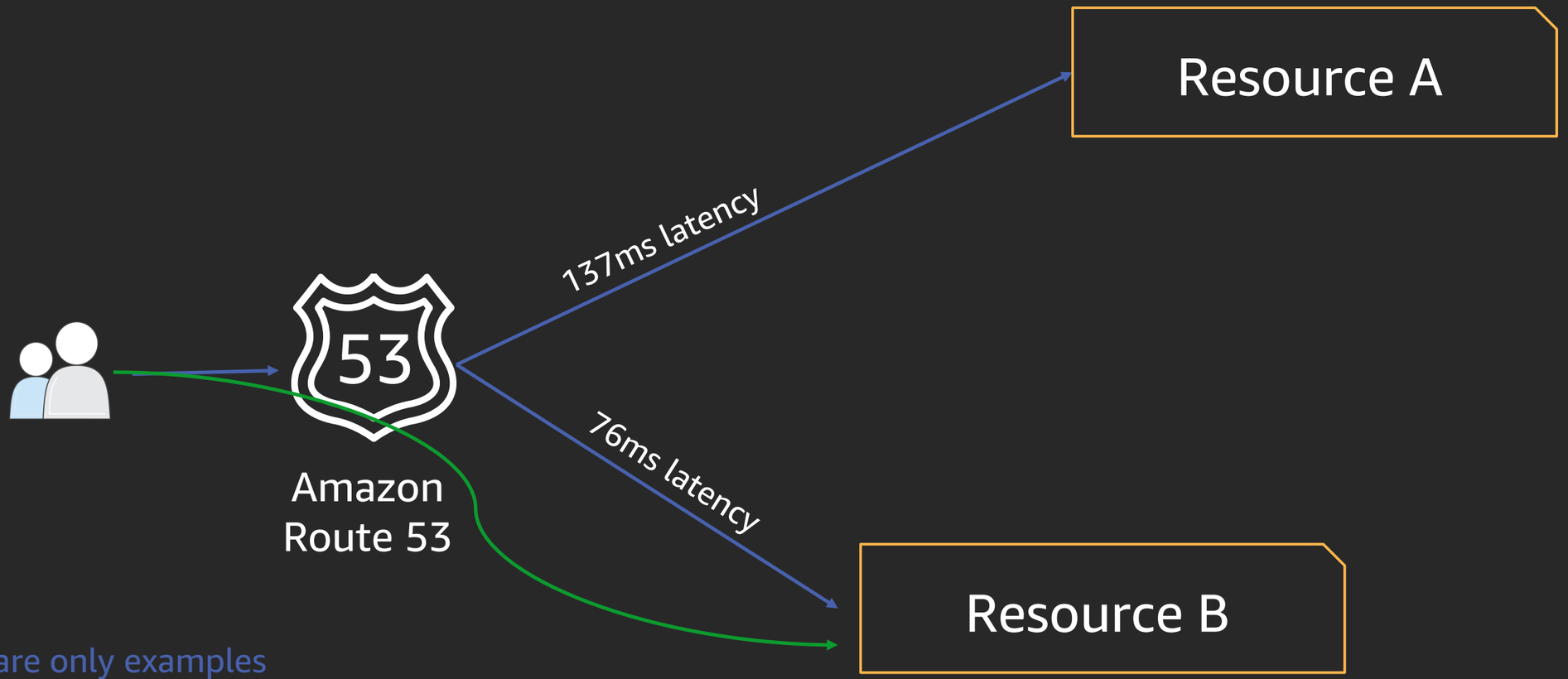


Traffic routing

Management

Traffic routing with Amazon Route 53

Latency-based routing

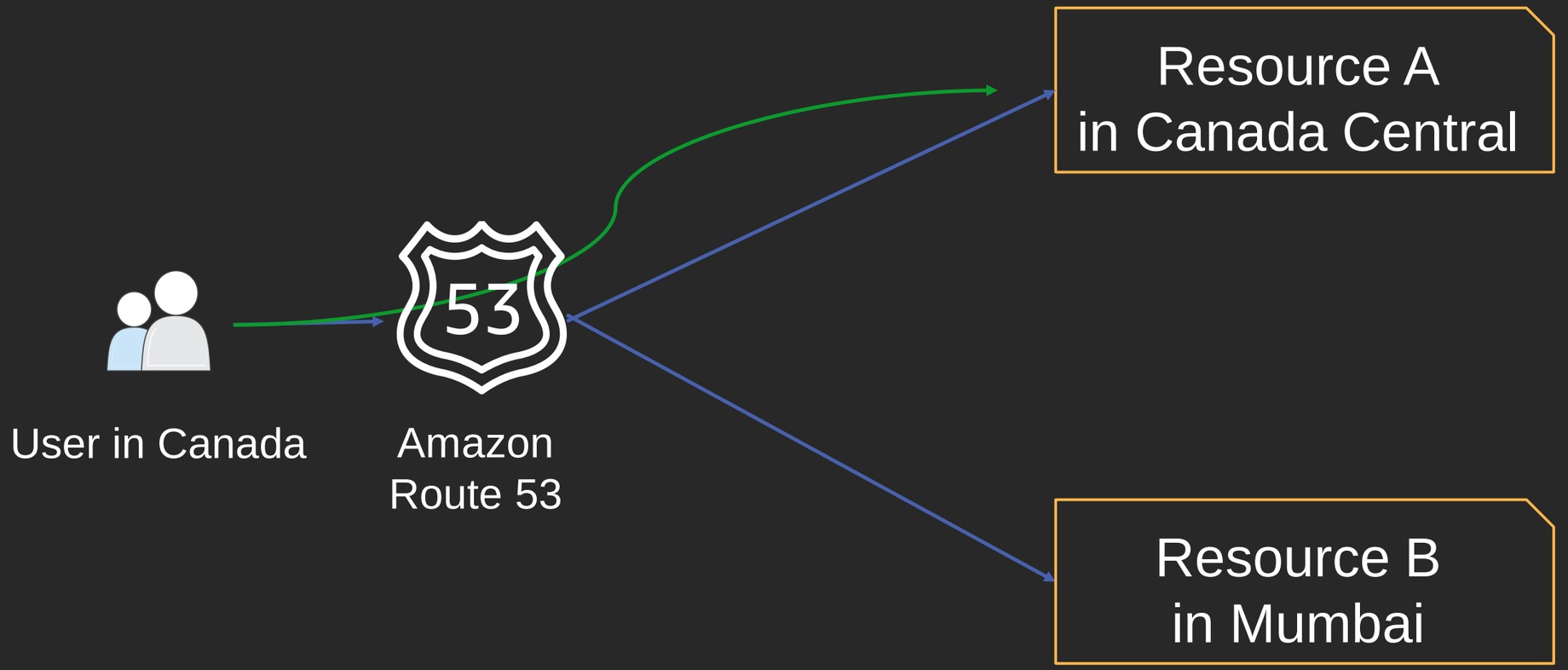


*Latency numbers are only examples

Traffic routing with Amazon Route 53

Latency-based routing

Geolocation routing

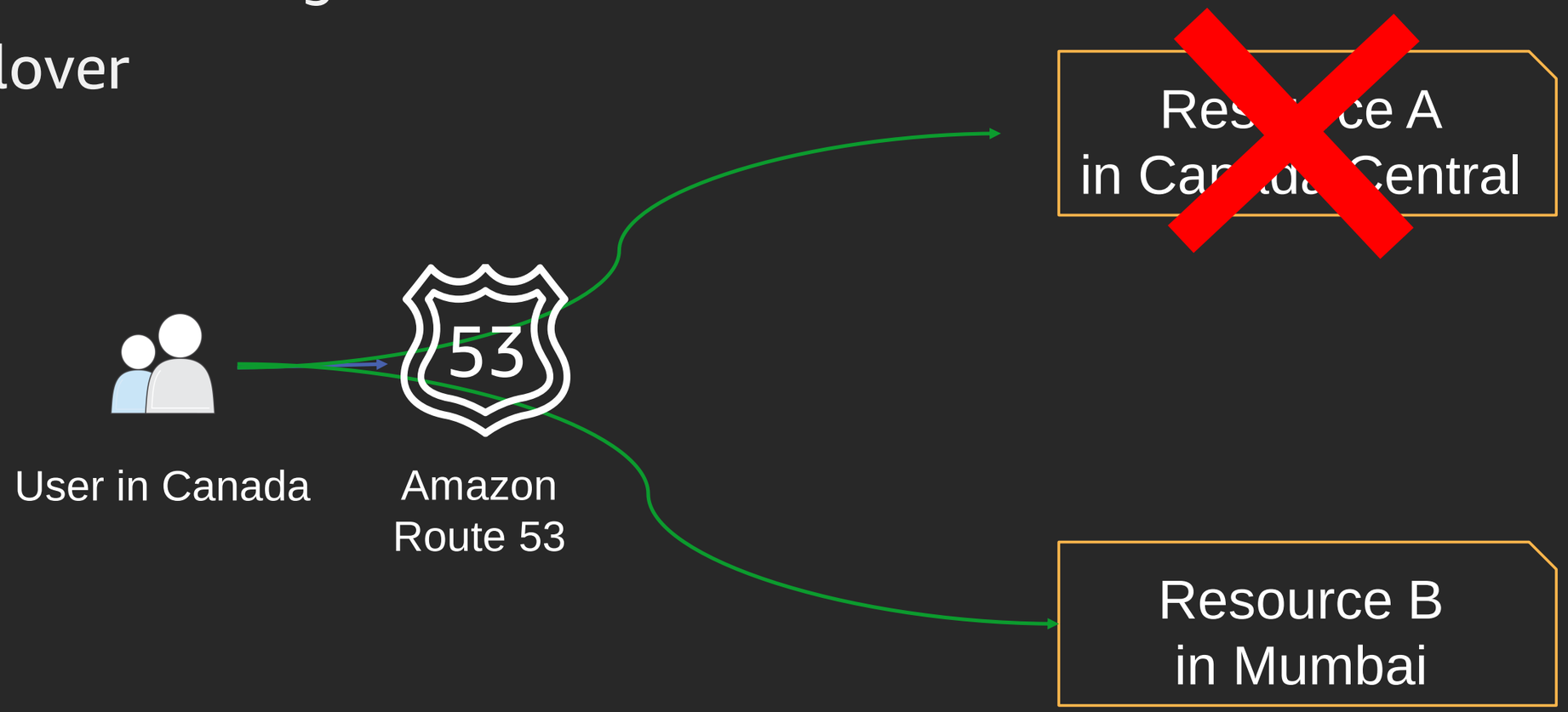


Traffic routing with Amazon Route 53

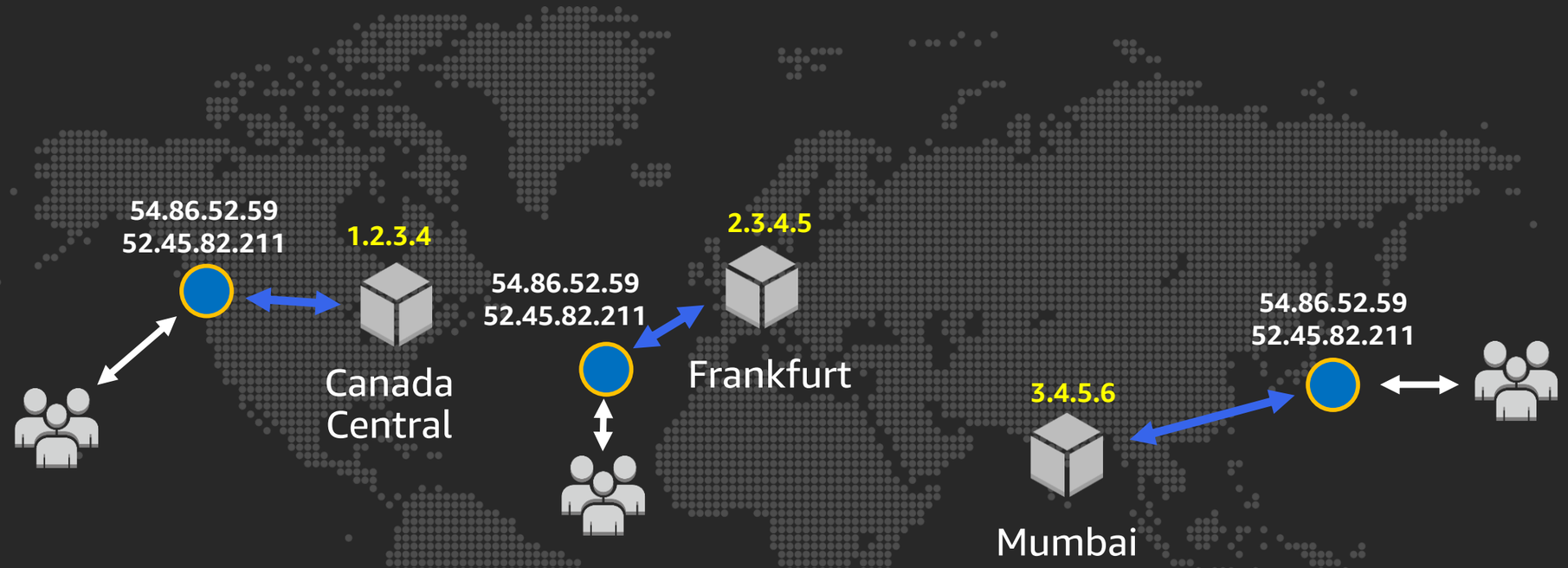
Latency-based routing

Geolocation routing

DNS failover



Traffic routing with AWS Global Accelerator

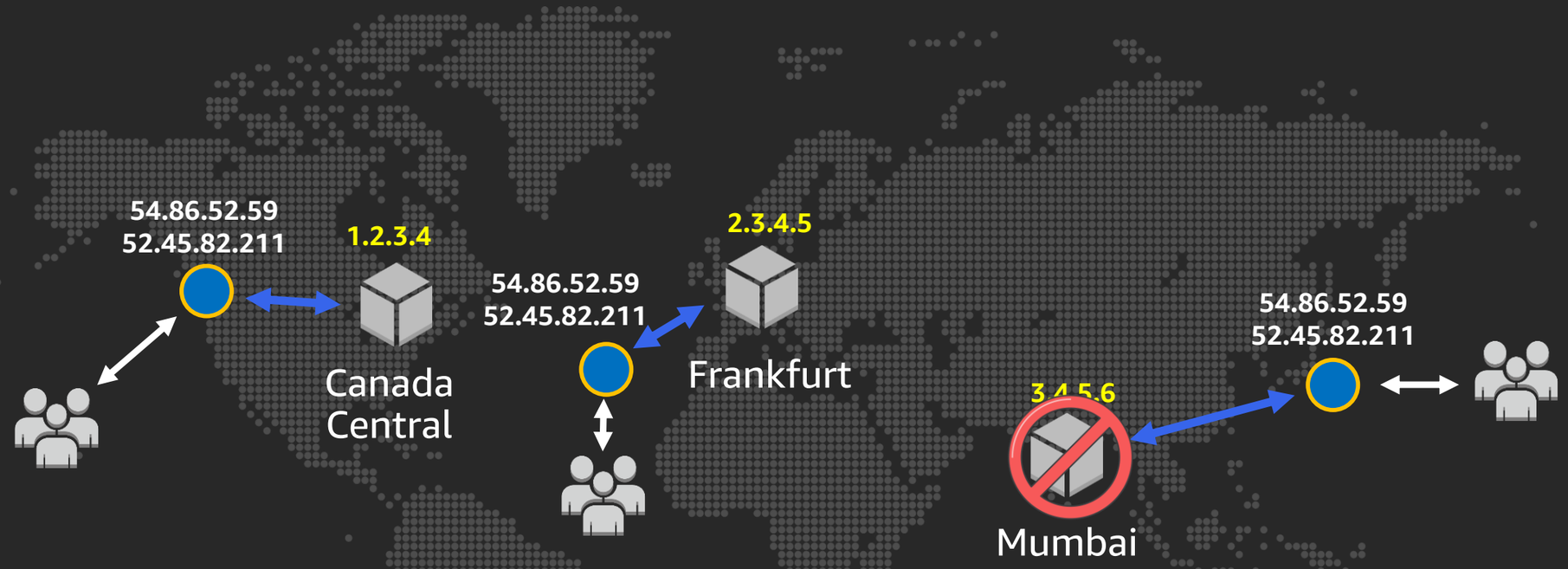


All clients point to the same static IPs and are directed to the closest PoP

Global Accelerator chooses the optimal AWS Region based on the geography of the client

Global Accelerator endpoint with anycast IP
e.g., **54.86.52.59, 52.45.82.211**

Traffic routing with AWS Global Accelerator

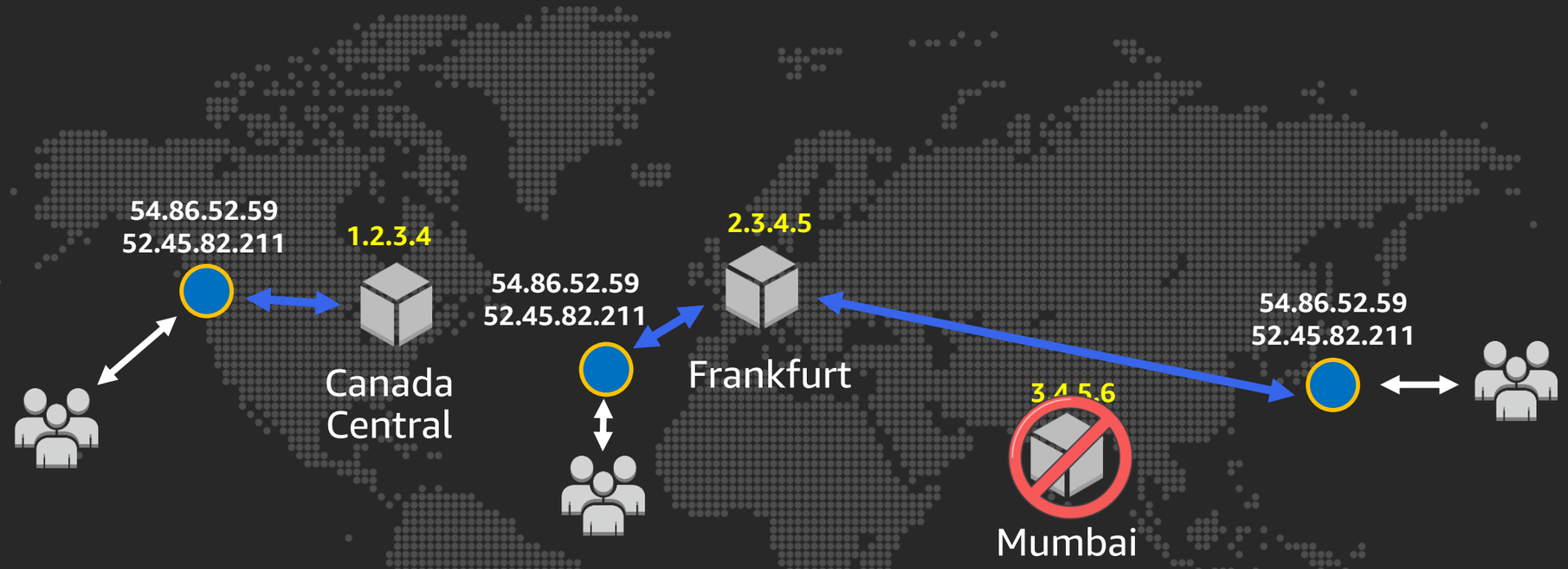


All clients point to the same static IPs and are directed to the closest PoP

Global Accelerator chooses the optimal AWS Region based on the geography of the client

Global Accelerator endpoint with anycast IP
e.g., **54.86.52.59, 52.45.82.211**

Traffic routing with AWS Global Accelerator



All clients point to the same static IPs and are directed to the closest PoP

Global Accelerator chooses the optimal AWS Region based on the geography of the client

Global Accelerator endpoint with anycast IP
e.g., **54.86.52.59, 52.45.82.211**

DNS-based solutions vs. Global Accelerator

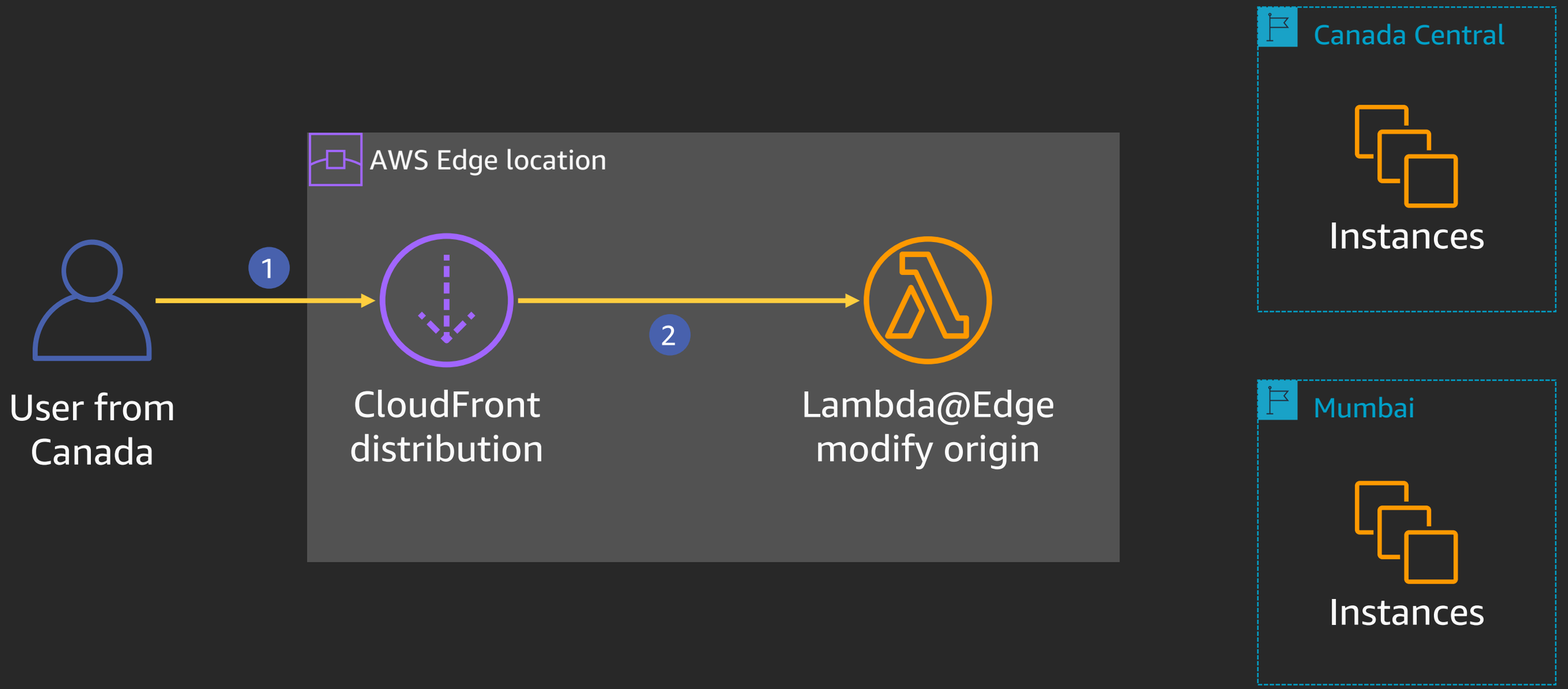
DNS-based traffic management solutions

- Client devices can cache DNS answers for a long time
- Hard to know when users will have the updated IP addresses
- This matters for blue/green deployments, backend failure, or change in routing preferences

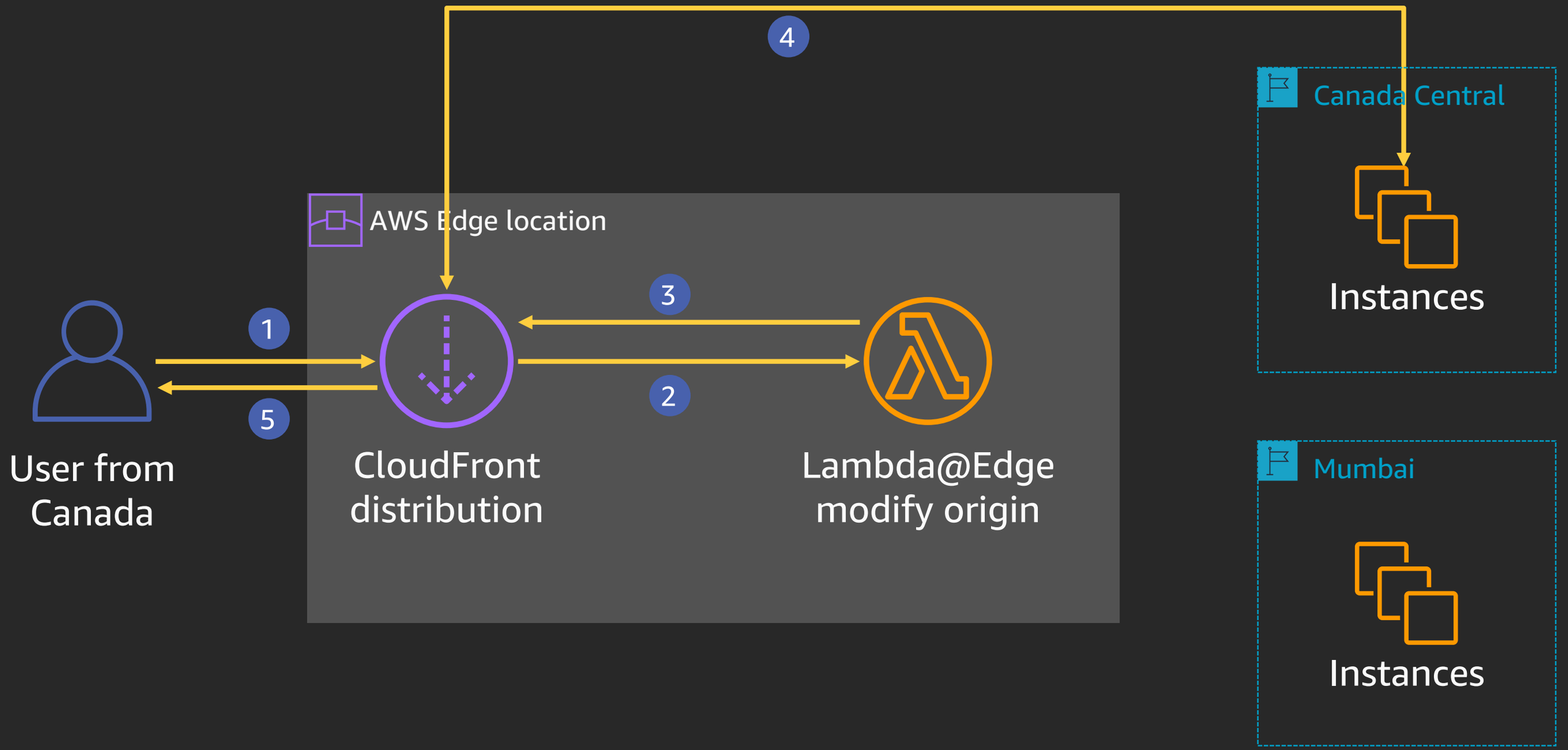
Global Accelerator

- No reliance on IP address caching of client devices
- Reduced downtime (change propagation in seconds)
- Static IP addresses — No need to update DNS or clients when moving endpoints across Regions/AZs

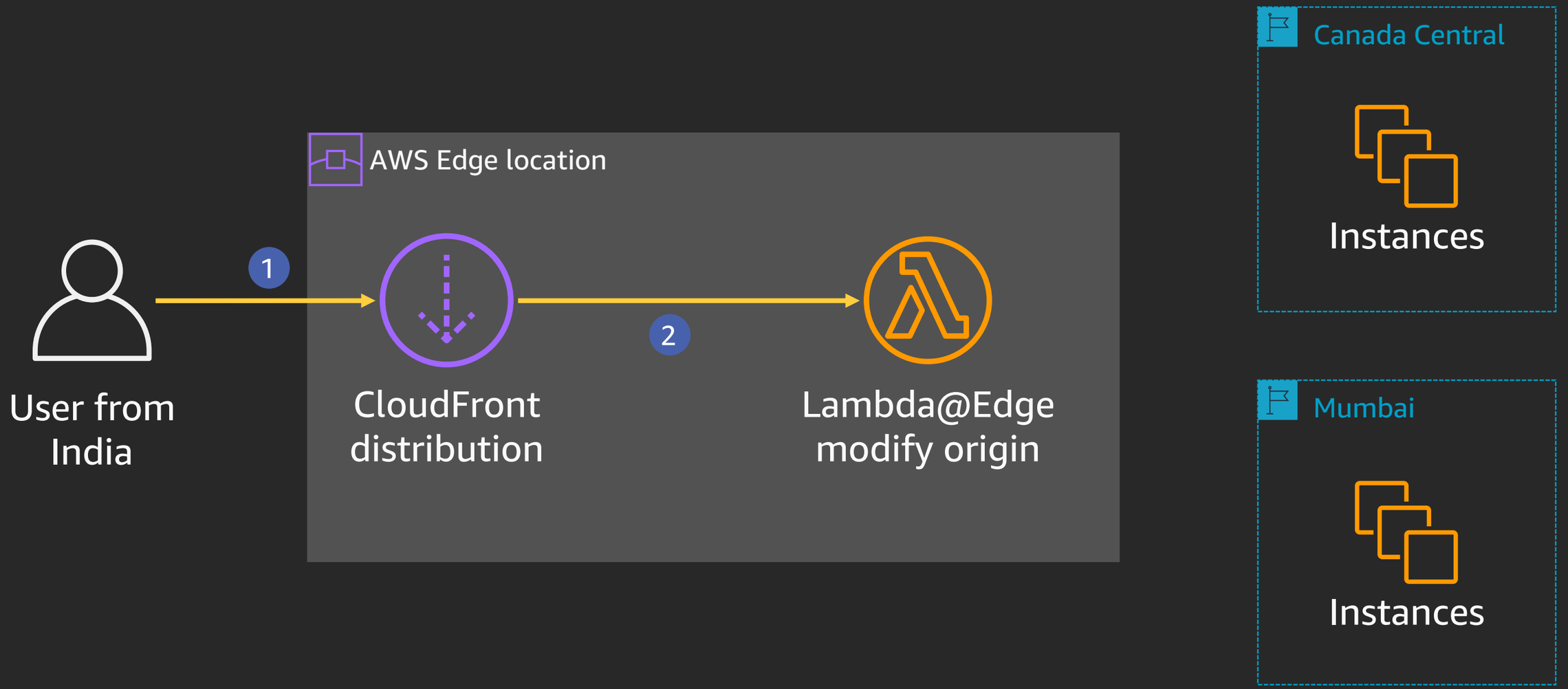
Traffic routing with Amazon CloudFront + Lambda@Edge



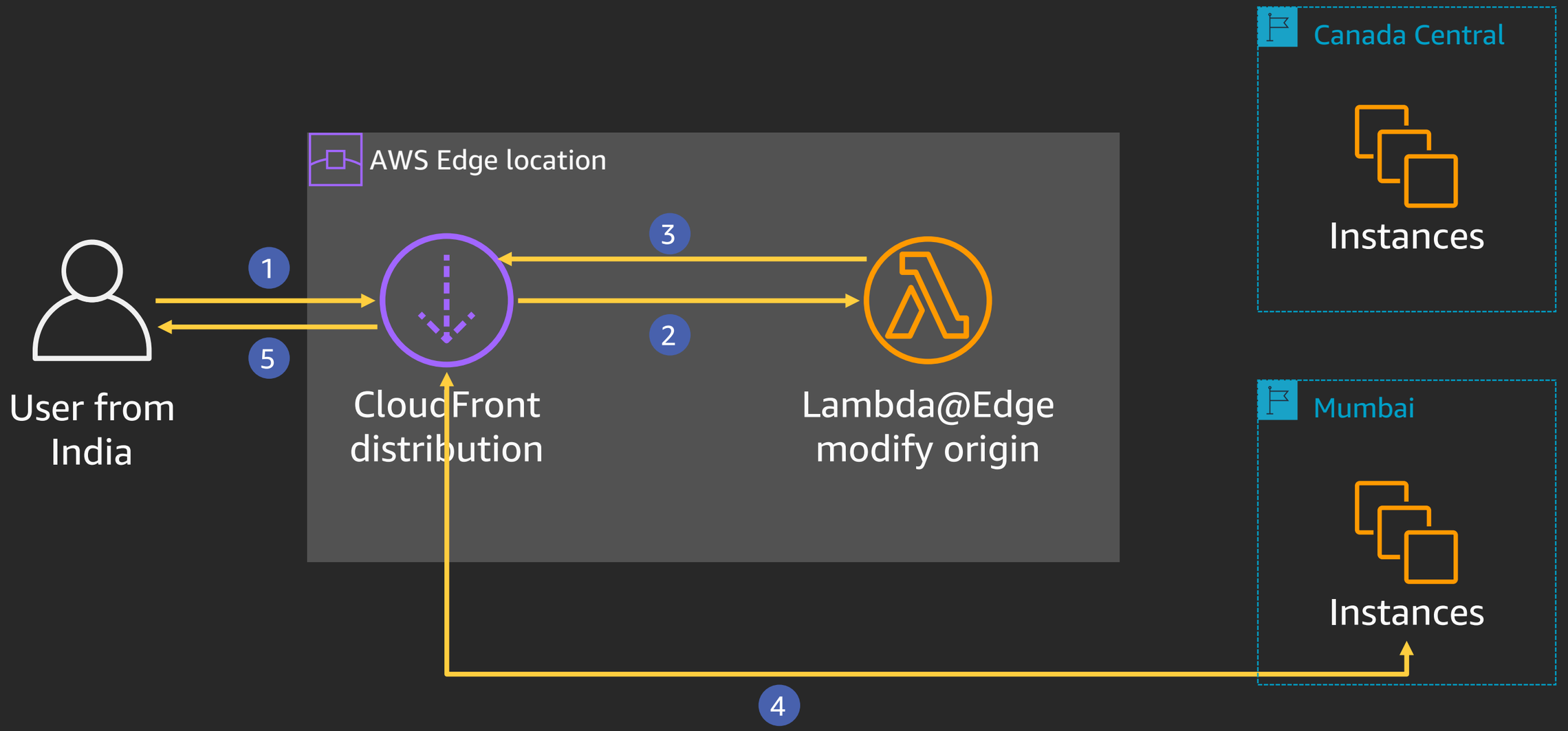
Traffic routing with Amazon CloudFront + Lambda@Edge



Traffic routing with Amazon CloudFront + Lambda@Edge

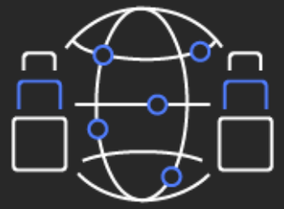


Traffic routing with Amazon CloudFront + Lambda@Edge



Foundational pillars of a multi-region active-active architecture

High availability



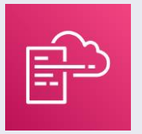
Data replication

Networking

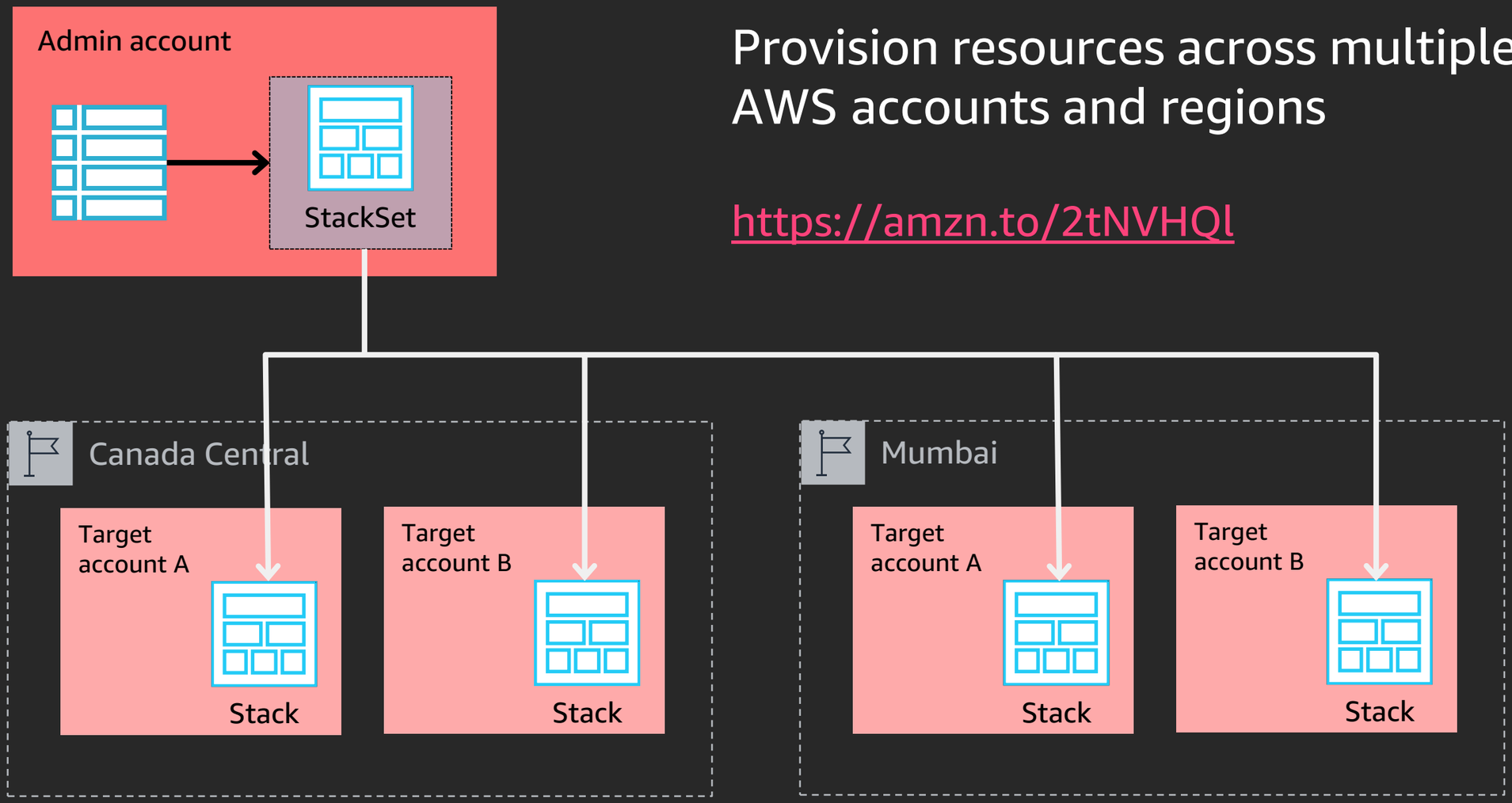
Traffic routing

Management

Management of multi-region deployment

Management area	AWS service
Security and compliance	AWS Config Rules 
Automation and inventory	AWS Systems Manager 
Monitoring and logs	Amazon CloudWatch 
Resources provisioning	AWS CloudFormation StackSets 

AWS CloudFormation StackSets



Provision resources across multiple AWS accounts and regions

https://amzn.to/2tNVHQl

Customer story

Multi-Region Architectures

Why, what, and how

Me: Thomas Jackson

- Head of Core & Data Infrastructure @ Wish

- Work experience:
  - Network Engineer
  - Corporate IT
  - Small Startups
  - Freelance Work
  - LinkedIn (professional social network)
  - Wish (mobile-first ecommerce platform)

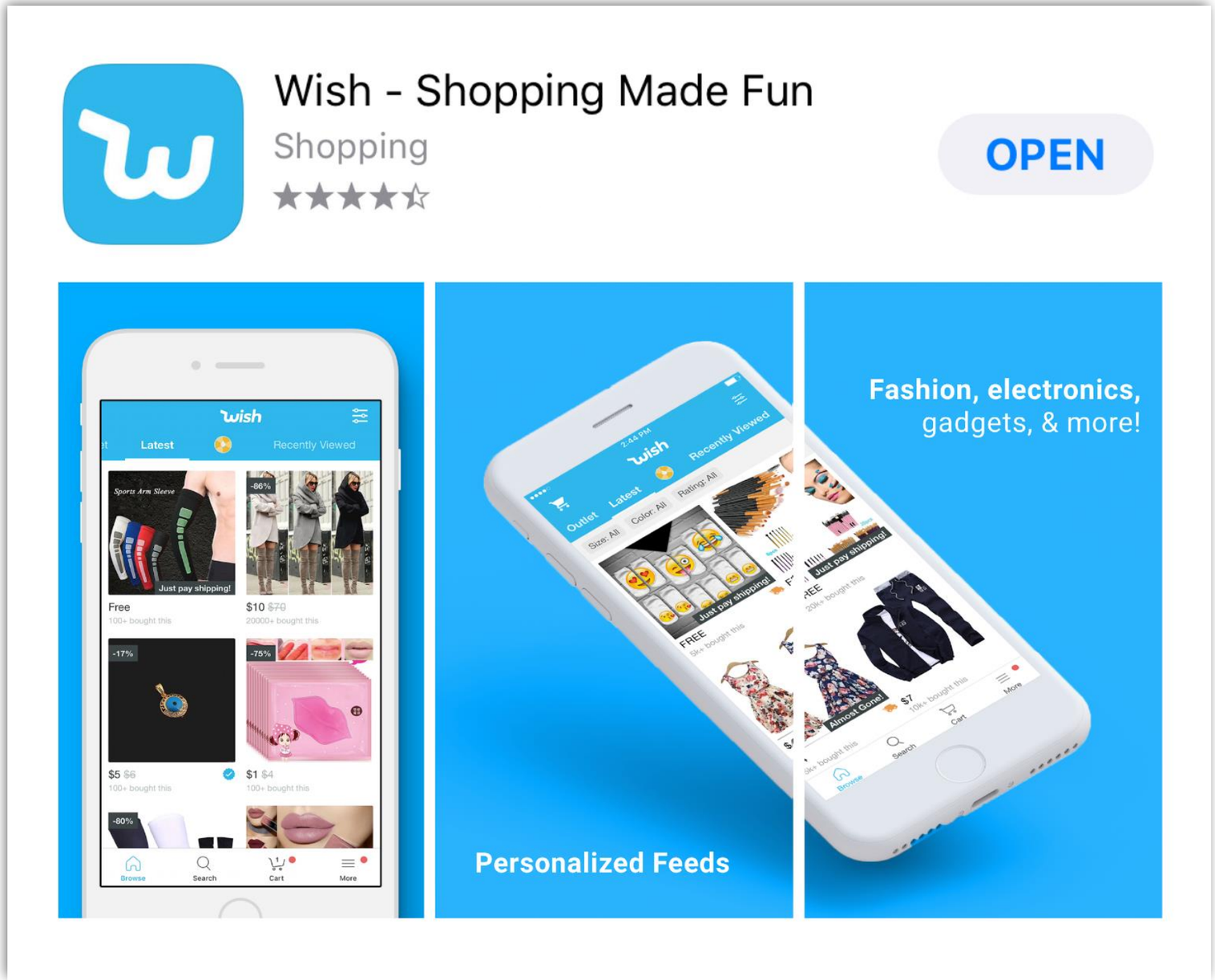
About Us

Who We Are

Leading mobile commerce platform in US and EU.

Our Mission

To offer the most affordable, convenient, and effective mobile shopping mall in the world.



Global Reach

500M+
Users

1M+
Merchants

200M+
Items

Wish (the company)

600+
Employees

7
Offices

$11.2B
Valuation

San Francisco (HQ)
United States

San Jose
United States

Toronto
Canada

Seattle
United States

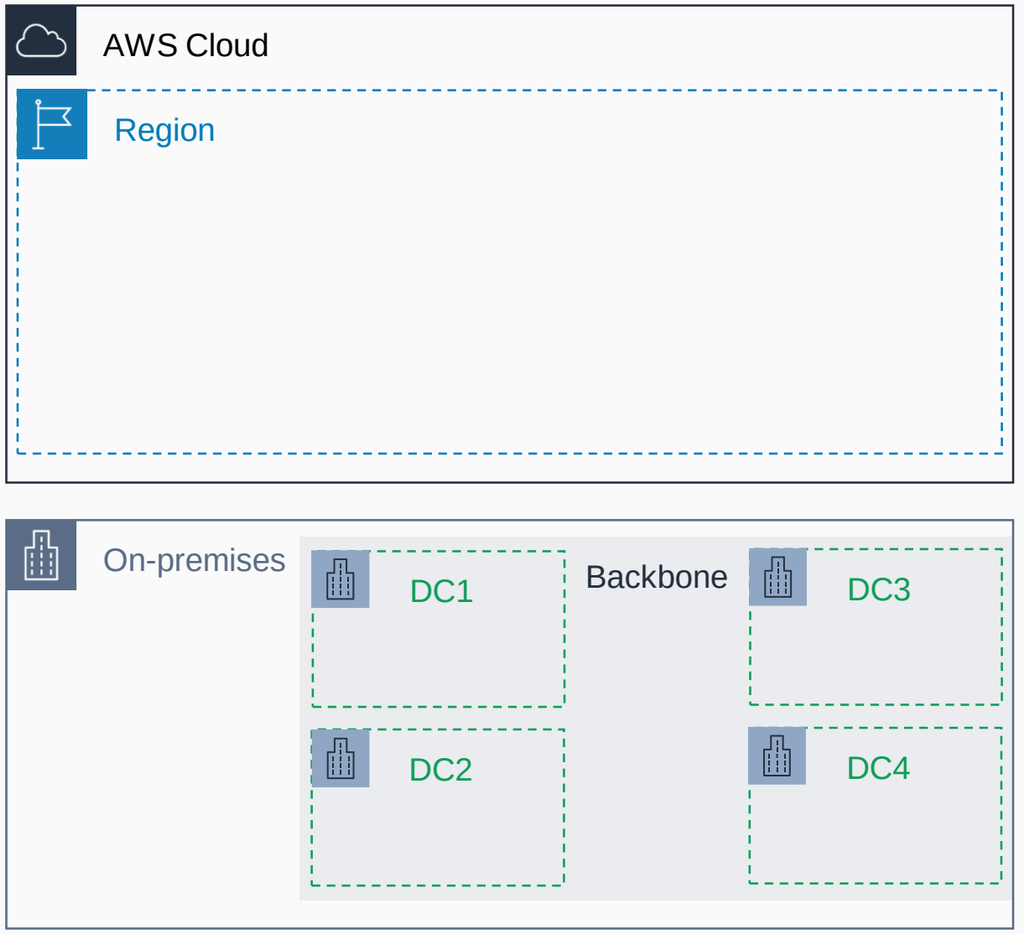
Amsterdam
The Netherlands

Shanghai
China

Hangzhou
China

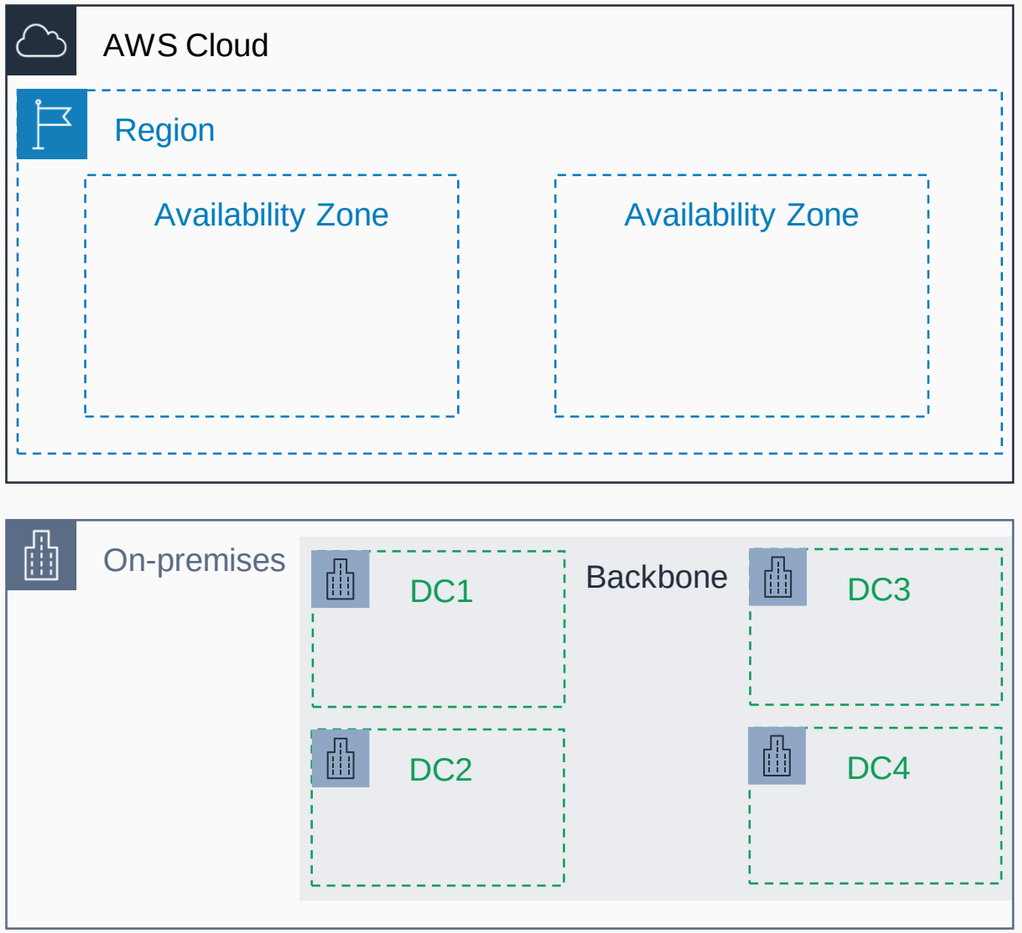
High-Level Architecture (before)

- Hybrid cloud
  - Single AWS Region
  - Multiple data centers with backbone/DX
- Read-heavy application
  - ~90%+ reads
  - Globally sharded+replicated database



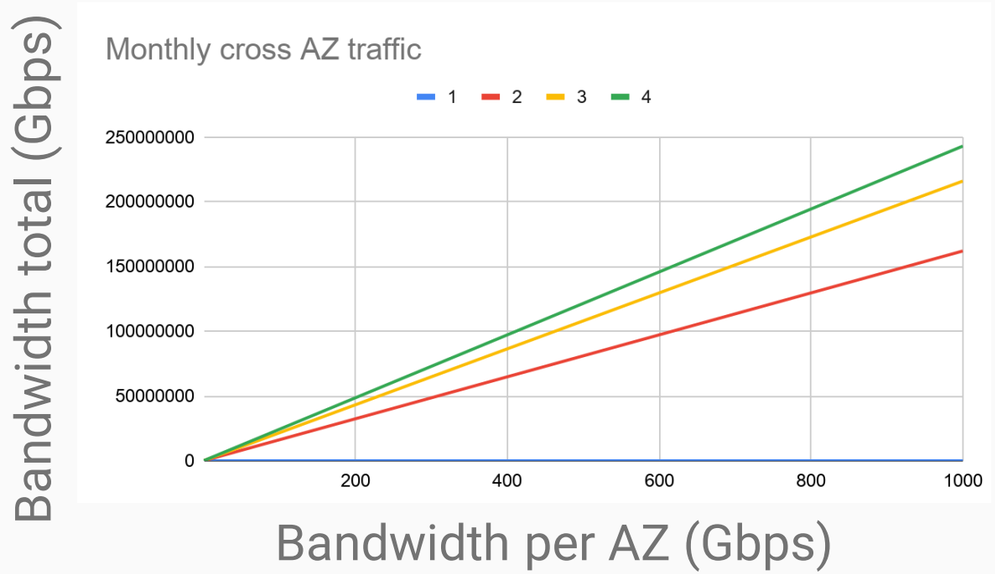
High-Level Architecture (before)

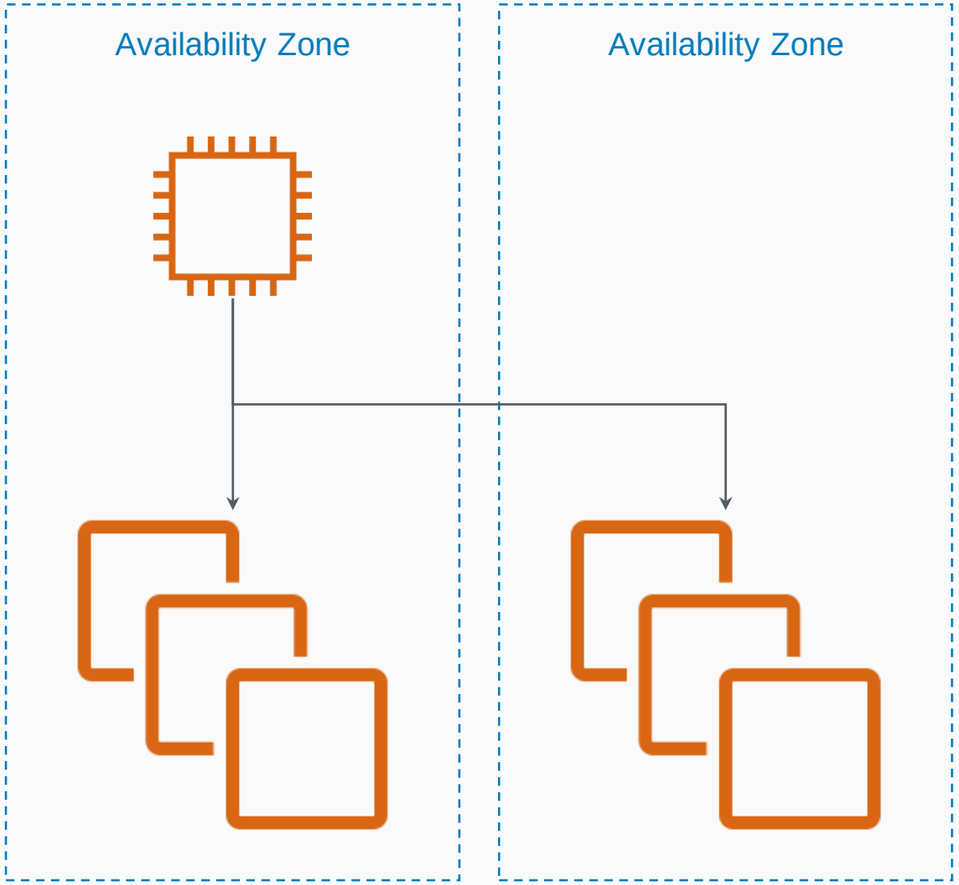
- Hard AZ split
  - AZ is already the AWS failure unit
  - “Simple”: self-contained AZ (mostly)
  - Avoid cross-AZ network transfer costs
    - Some (such as DB replication) are required



Cross-AZ transfer costs

- What?
  - Service fan-out means at each hop we have a chance for cross-AZ traffic

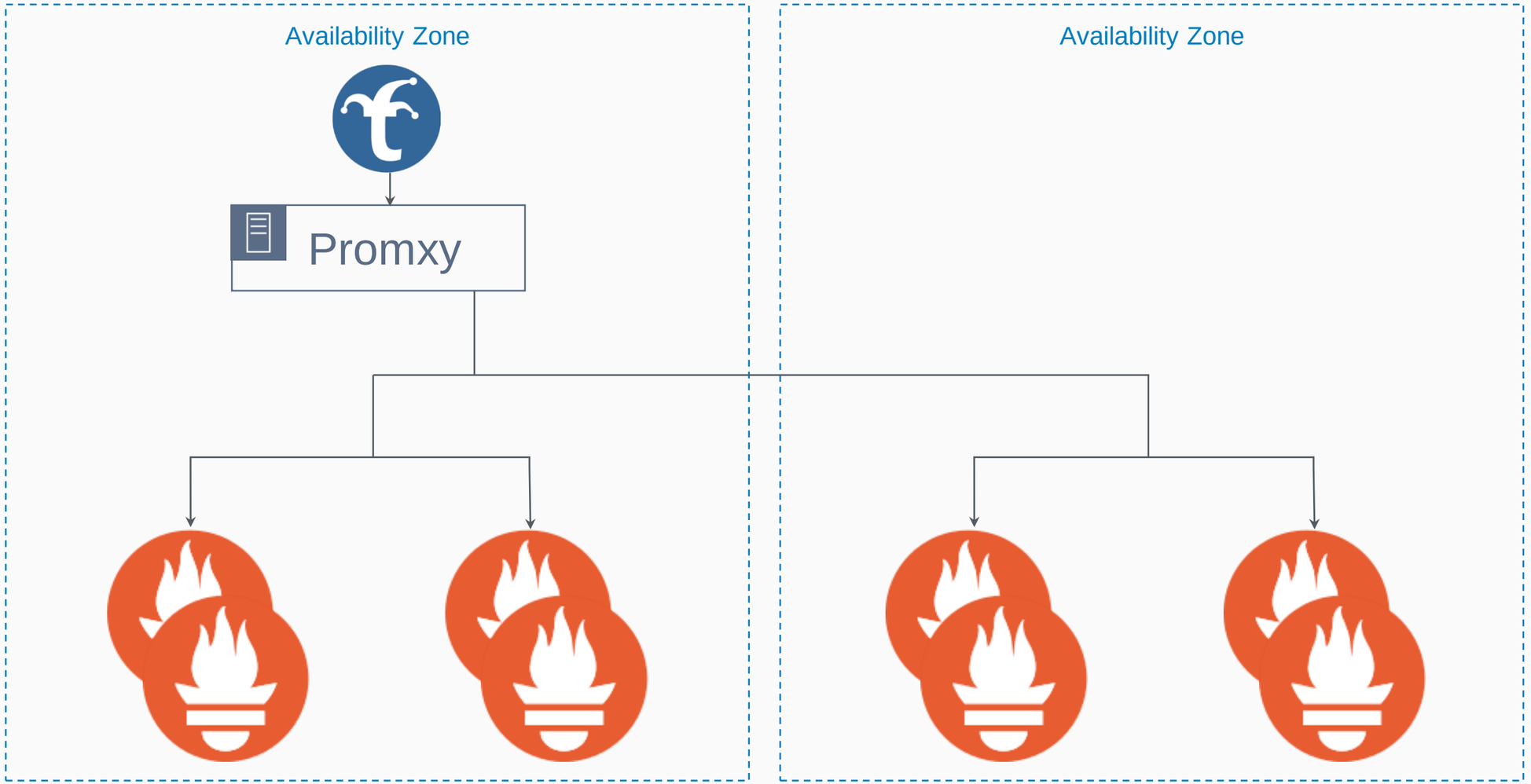




High-Level Architecture (before)

- Monitoring
  - Prometheus: scraping, storage, alerting
  - Promxy: aggregation, alerting
  - Trickster: caching

High-Level Monitoring Architecture (before)



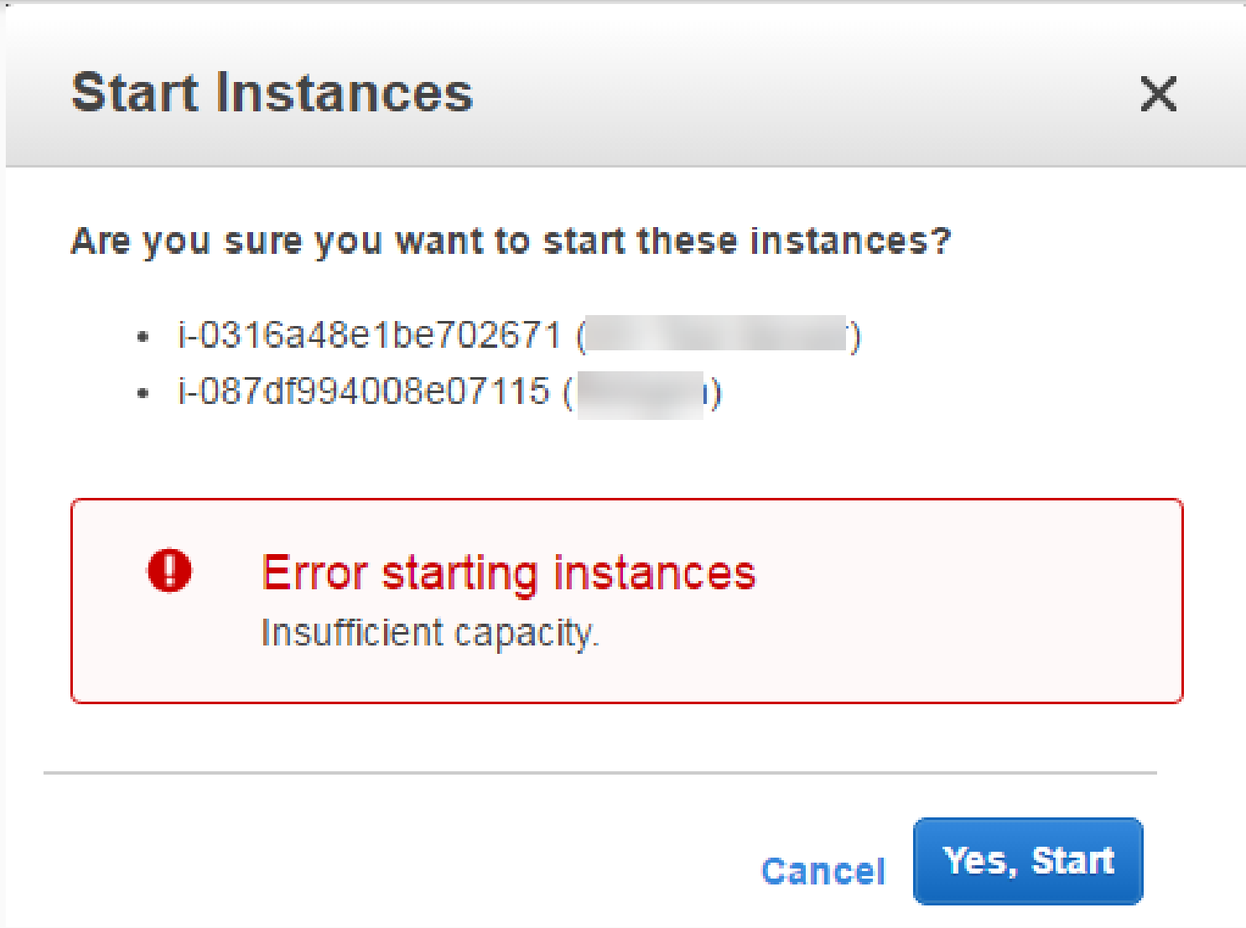
More details: bit.ly/2Lilbf0

Why?

Why change anything?

Why?

- Scalability
  - ICE: InsufficientInstanceCapacity
- The cloud is elastic, until it isn’t
  - An issue in “crunch” times
  - An instance type might become “hot” in one or more AZs
  - Mitigate by using many instance types



Why?

- Availability
- At our scale, an outage is too costly
  - Any DR adds complexity, not *always* worth it

The plan (v1): Active/Active

- Passive is expensive and doesn’t solve our scale issues

The plan (v1): Scope

- Focus on Mean Time To Recovery (MTTR)
  - Uptime isn’t free (it's not even cheap!)
  - Not ***every*** system requires 99.999% uptime
- Focus on where we can make the most impact
  - Include user-facing services
  - Excluded other services (e.g., data analytics) that can handle a brief service interruption

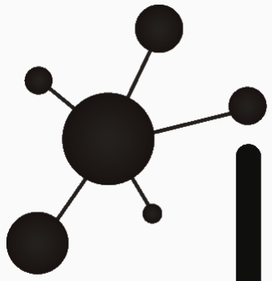
User-facing system

Data analytics

The plan (v1): Tech-Debt Cleanup

- Deprecations
  - Chef
  - Icinga
  - Graphite
  - Outdated OS versions
  - Etc.

CHEF



icinga

The plan (v1): Newer systems

- Saltstack
- Kubernetes
- Prometheus

The plan (v1): Change plan

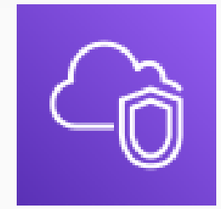
- Supporting multi-region requires changes
- Changes should be applied in *current* region
  - Minimizes “change” when bringing up new site
  - Avoids “drift” of system during planning and execution

The plan (v1): Deadline

- End of October (before November holiday shopping season)

Hurdle 1: Internal network

- Context/background
  - Very little cross-AZ traffic, but some
  - Need to support cross-region as well (e.g., notifications cluster)
- Solutions
  - Inter-region VPC peering for AWS <-> AWS
  - Backbone for Colo <-> Colo

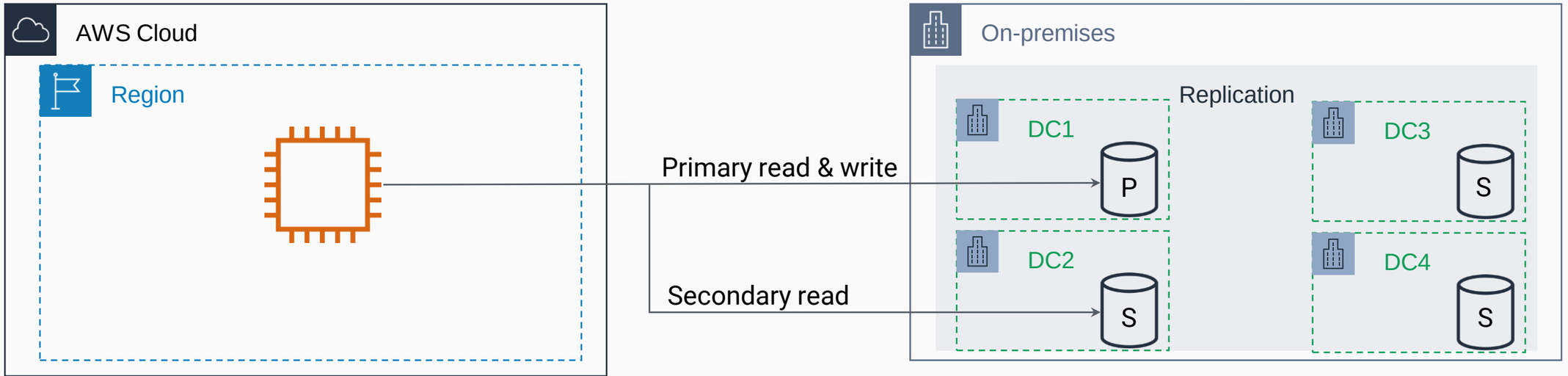


Amazon VPC

Peering connection

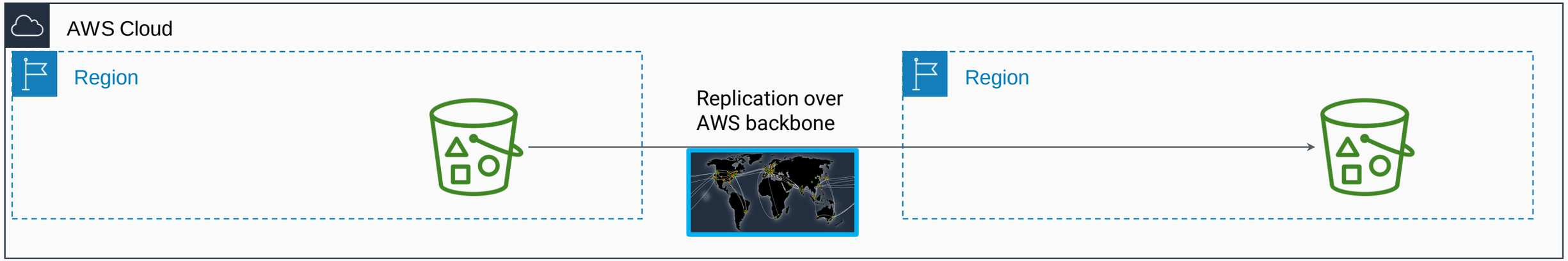
Hurdle 2: Data consistency

- Main app DB -- Globally sharded DB
  - Single primary per shard — auto failover through election
    - As a read-heavy application, we can take a ~70ms hit on writes
  - Most reads can be “stale”
    - split reads between primary/secondary reads for performance/capacity



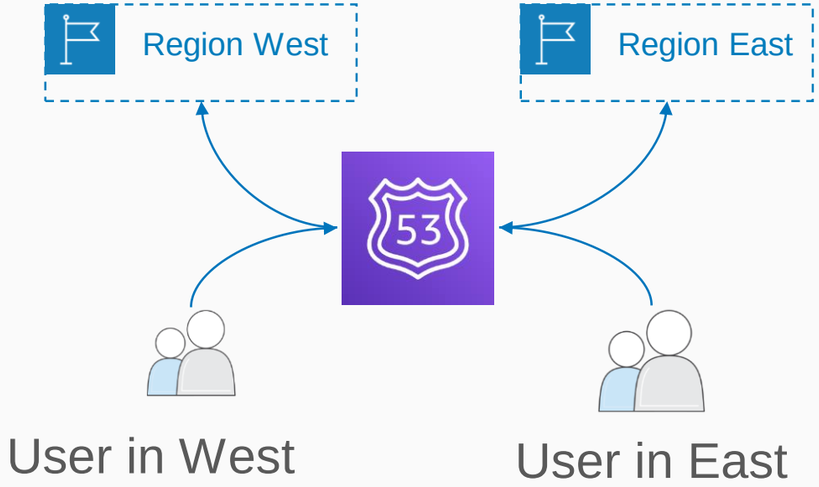
Hurdle 2: Data consistency

- S3
  - Images, static content, etc.
  - Unidirectional cross-region replication



Hurdle 3: Traffic routing

- External
  - DNS: Don’t want different domain names for users
  - Route 53 geo-based balancing
- Internal
  - Migrated “the last few things” to service discovery (tech debt cleanup)



HashiCorp Consul

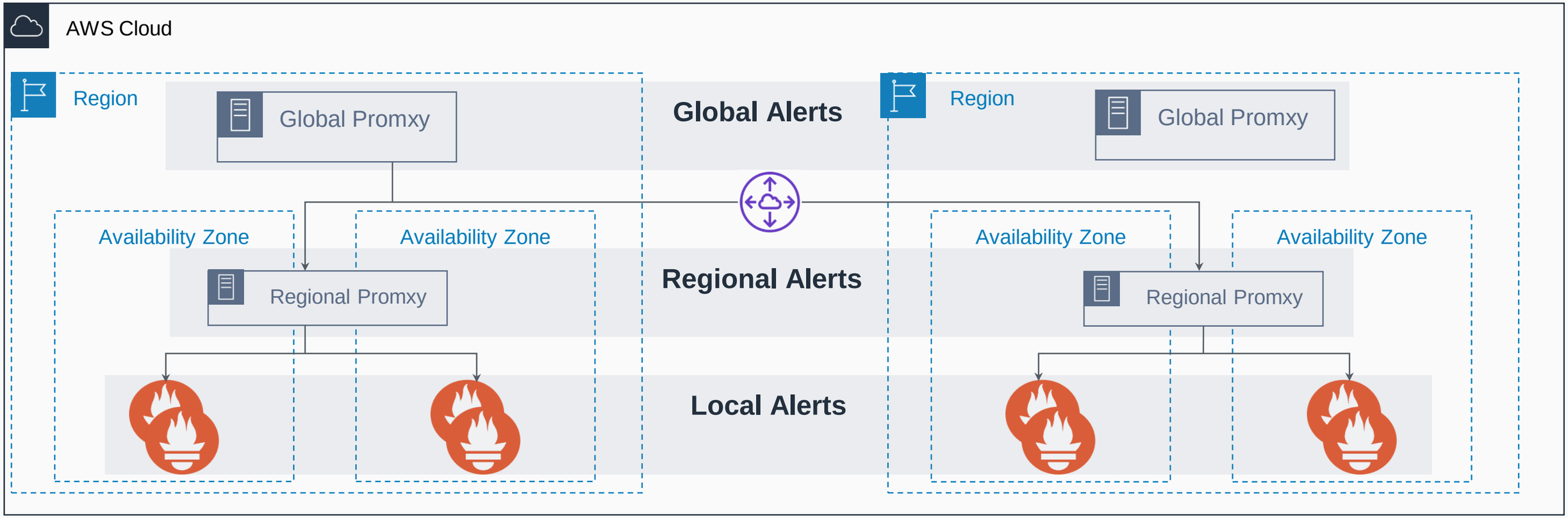
An aside on system behavior

- Systems tend to do what they ***can***, not what they ***should***
  - “no hard coded IPs” — but if it works, someone might
  - “Handle 100% traffic increase due to region failover”
- If you don’t want X to be done, don’t allow X or audit for X regularly
  - In k8s, hard-coded IPs don’t work
  - Regularly scheduled failovers force discipline

Hurdle 4: Monitoring

- Still using Prometheus and Cloudwatch
  - Internal metrics: 15s granularity
  - Cloudwatch metrics: API rate limits trying to ingest at that rate
- Tiered alerting (not all alerts require access to all the data)
  - Global
  - Regional
  - Local

Monitoring/Alerting Architecture



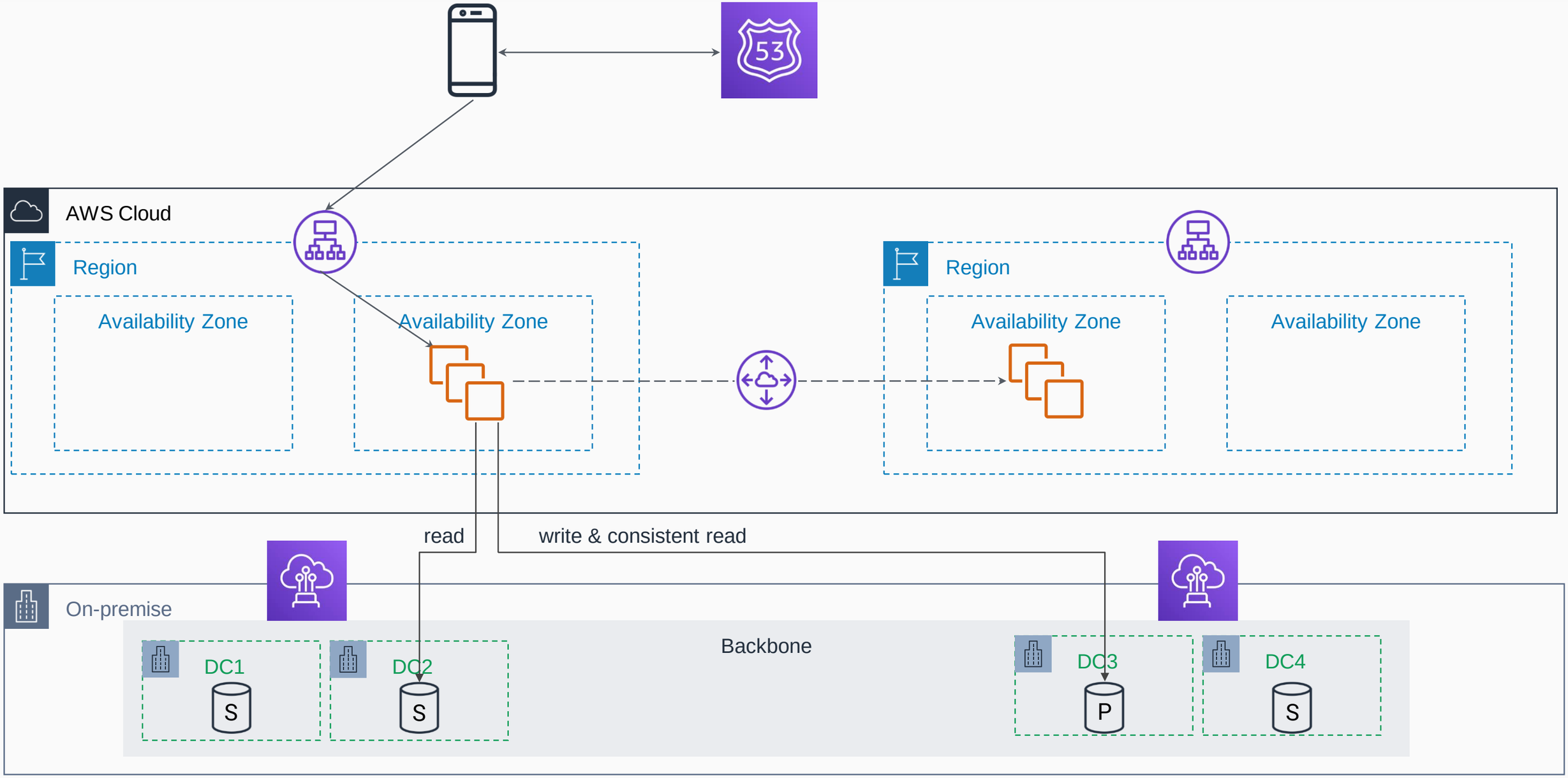
The best laid plans of mice and men often go awry

- Robert Burns

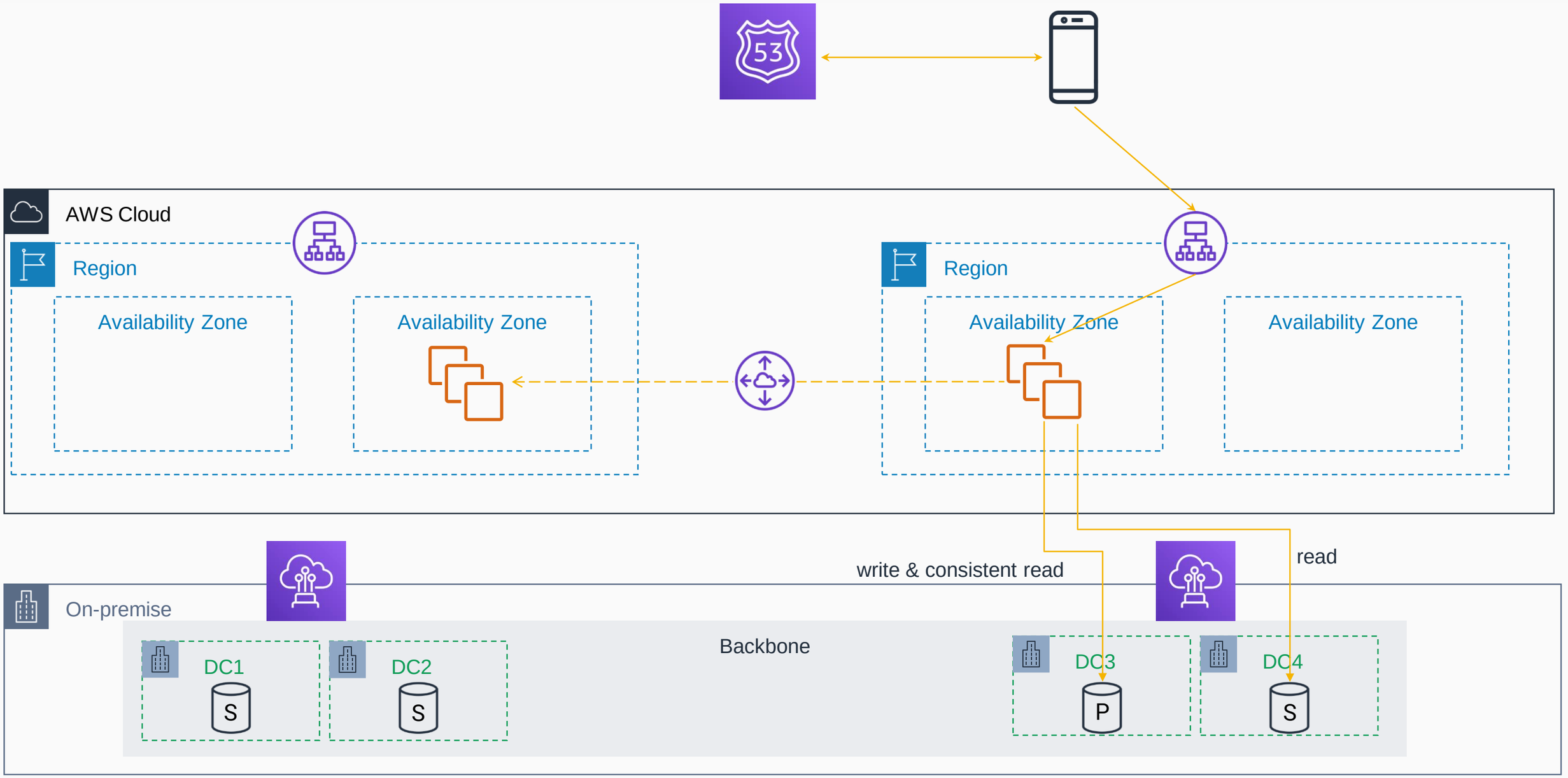
Adjustments

- Laxed requirements for some deprecations (Chef)
- Some changes were canaried in the new region first

High-Level Architecture (after)



High-Level Architecture (after)



Takeaways

- Understand your current architecture and why it is so (tech and business)
- Have a reliable mechanism to measure everything
- Cleanup tech debt along the way (where practical)
- Set timeline and scope, but be ready to adapt

Takeaways

Takeaways

- For reliability and high availability, use a Multi-AZ architecture as a first step
- Multi-region active-active architectures help limit the blast radius in cases of major adverse events
- There is higher reliability than conventional disaster recovery architectures, at the expense of additional complexity, due to constant usage of all regions

Takeaways

- Simplify
  - Minimize blocking dependencies between Regions
  - Minimize synchronous replications between Regions
  - Graceful degradation in case of connectivity issues
- AWS provides replication solutions for different data stores, such as DynamoDB Global Tables, RDS Cross-Region Read Replicas, S3 Cross-Region Replication, etc.
- Analytics stacks are typically deployed in one Region only.

Takeaways

- AWS Managed Services uses the AWS Backbone to replicate
- Use Inter-Region VPC peering to manage your own replication
- Think about traffic routing and manage it using Global Accelerator, Route 53 or CloudFront + Lambda@Edge
- Plan to manage the environment using tools like CloudFormation StackSets

Related breakouts

ARC309: Hands-on: Building a multi-region active-active solution

SVS337: Best practices for building multi-region, active-active serverless applications

ARC406: Building multi-region microservices

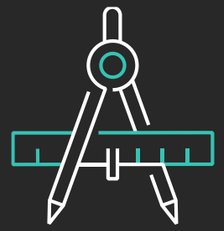
DAT308: Real case on boosting performance with Amazon ElastiCache for Redis

ARC304: From one to many: Diving deeper into evolving VPC design

NET202: Using AWS Global Accelerator for multi-region applications

Learn to architect with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Free foundational to advanced digital courses cover AWS services and teach architecting best practices

Classroom offerings, including Architecting on AWS, feature AWS expert instructors and hands-on labs

Validate expertise with the **AWS Certified Solutions Architect - Associate** or **AWS Certification Solutions Architect - Professional** exams

Visit aws.amazon.com/training/path-architecting/

aws training and certification

Thank you!

Girish Dilip Patil
linkedin.com/in/girish-cloud

Jonathan Dion
linkedin.com/in/jotdion
@jotdion

Thomas Jackson
tjackson@wish.com
linkedin.com/in/jacksontj
github.com/jacksontj

Please complete the session survey in the mobile app.