ARC319

AWS re:Invent

Multi-Region Active-Active Architecture

Glenn Gore, Director, Solutions Architecture
Girish Patil, Senior Solutions Architect
Darin Briskman, Developer Evangelist

AWS re:Invent

aws

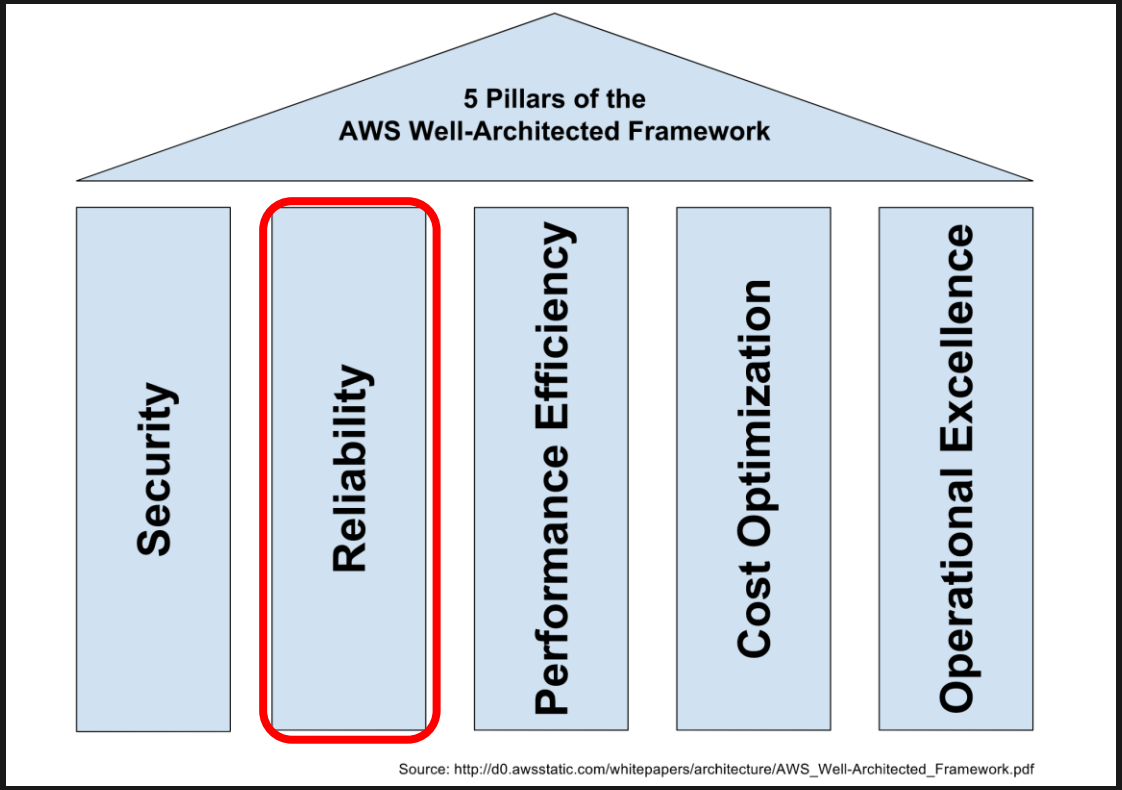
Session objectives

Session objectives

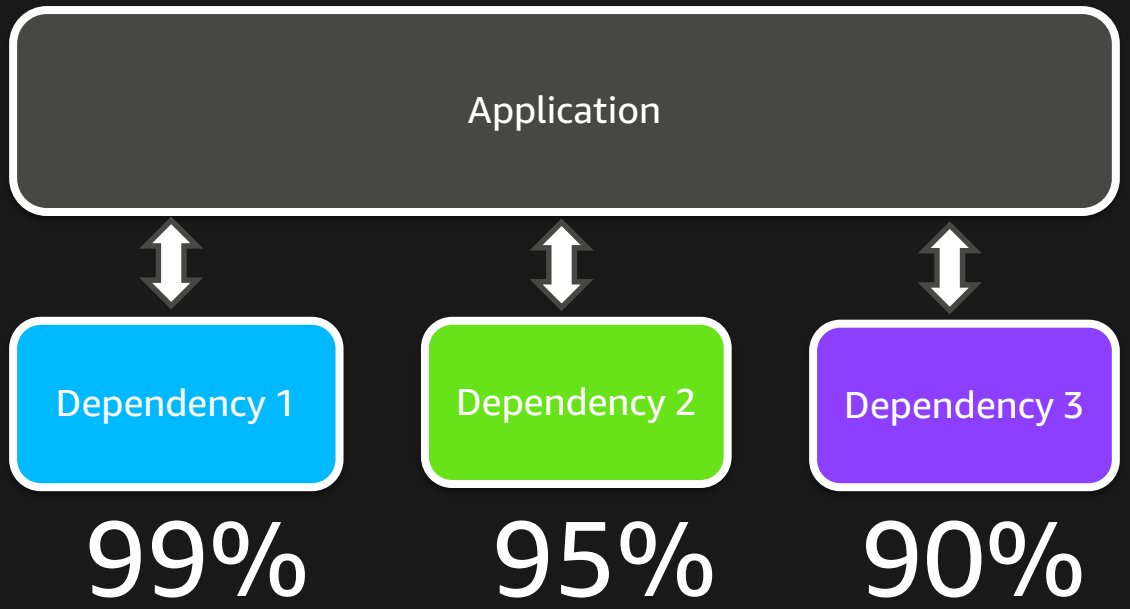
1. Understand how to think about application availability
2. Understand how AWS makes services highly-available
3. Understand why you might need Multi-Region Active-Active architecture
4. Review AWS services that are useful for Multi-Region Active-Active design
5. Trade-offs involved in the design

Understand how to think about application availability

Where are we?



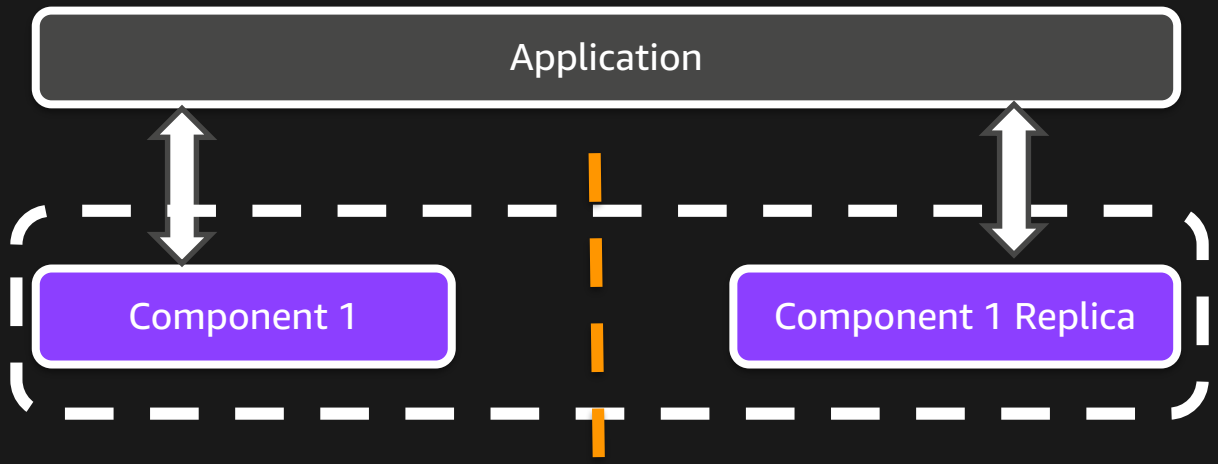
Calculating availability with hard dependencies



<= 90%

"A chain is only as strong as its weakest link."

Calculating availability with redundant components



90%

90%

Let us understand intuitively

~99%

Assuming instantaneous failover!

“Component redundancy increases availability significantly!”

“Nines” of availability

Availability	Max Disruption (per year)	Application Categories
99%	3 days 15 hours	Batch processing, data extraction, transfer, and load jobs
99.90%	8 hours 45 minutes	Internal tools like knowledge management, project tracking
99.95%	4 hours 22 minutes	Online commerce, point of sale
99.99%	52 minutes	Video delivery, ridesharing services, broadcast systems
99.999%	5 minutes	ATM transactions, telecommunications systems

Reliability Pillar White-paper (Nov 2017)

Your application's availability depends …

… not only on availability of AWS services, but also on the quality of your software and your adherence to operational best practices!

"Nines" of availability

Availability	Max Disruption (per year)	Application Categories
99%	3 days 15 hours	Batch processing, data extraction, transfer, and load jobs
99.90%	8 hours 45 minutes	Internal tools like knowledge management, project tracking
99.95%	4 hours 22 minutes	Online commerce, point of sale
99.99%	52 minutes	Video delivery, ridesharing services, broadcast systems
99.999%	5 minutes	ATM transactions, telecommunications systems

Consider Multi-Region
Active-Active for this class

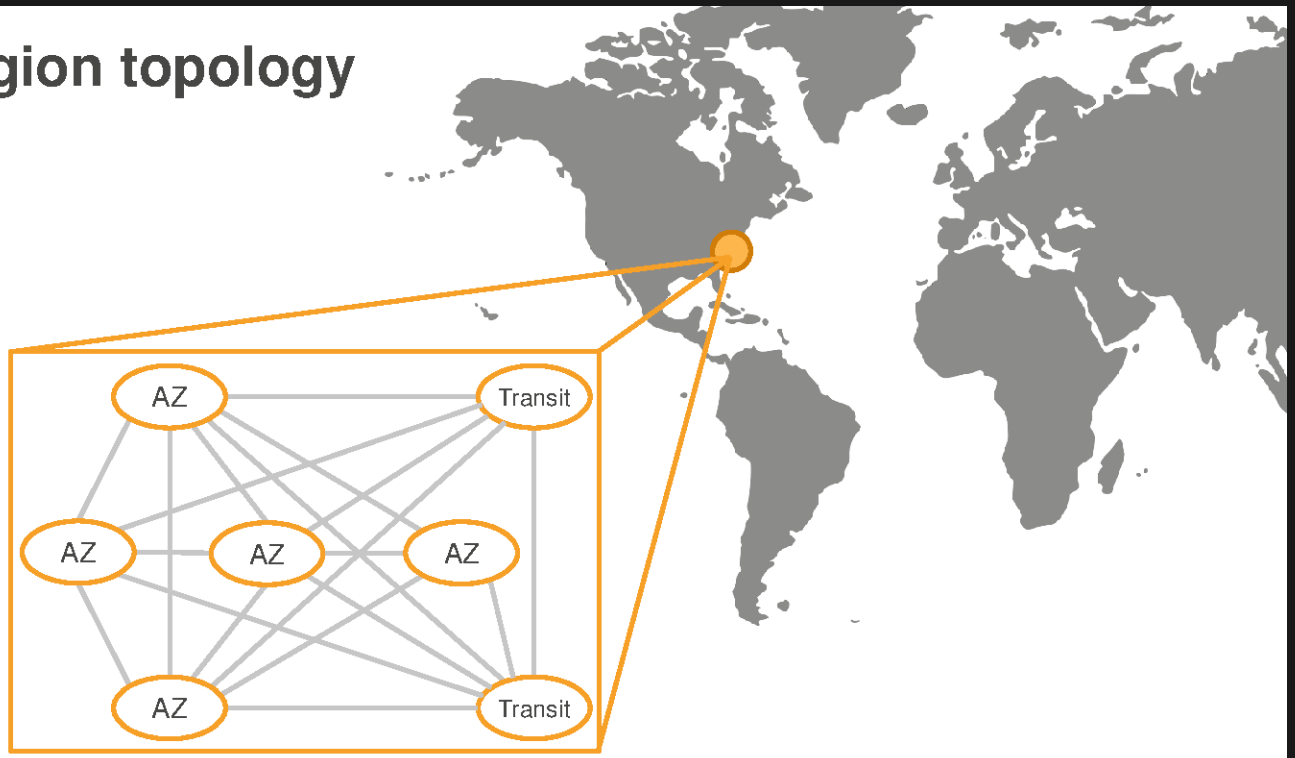
Reliability Pillar whitepaper (Nov 2017)

Understand how AWS makes services highly available

AWS re:Invent

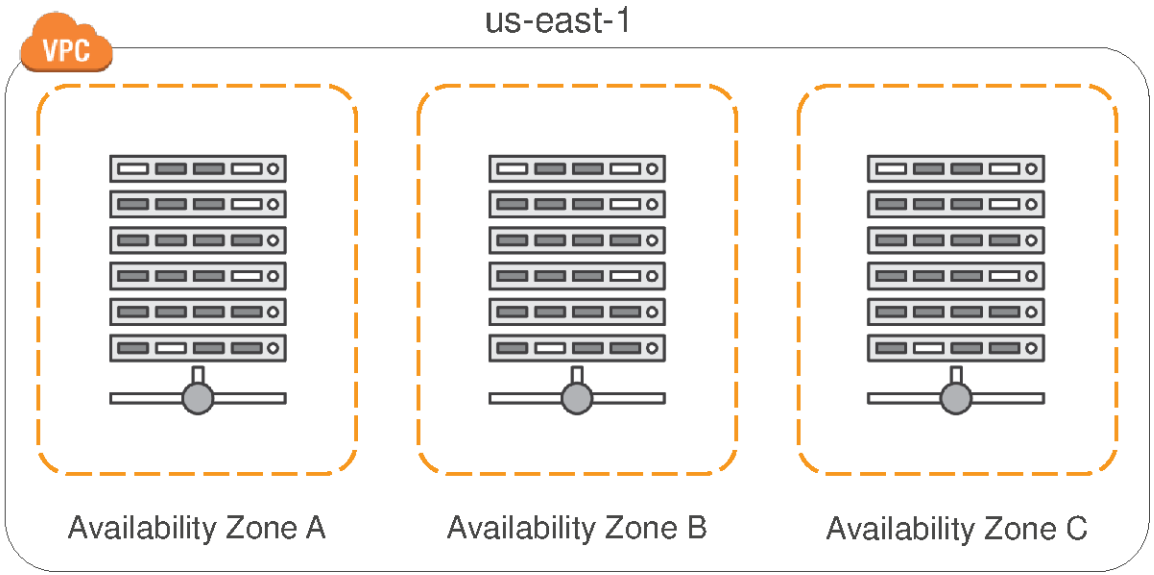
aws

Region topology



Single region high-availability approach

- Leverage multiple Availability Zones (AZs)



Region-wide AWS services

Default


Amazon S3


Amazon DynamoDB


Amazon EFS


Amazon SQS


Amazon Kinesis

Configurable for multi-AZ deployment


Amazon RDS


Amazon ElastiCache


Amazon ES

High availability for EC2 – instance recovery

New instance, same in all manners as old instance

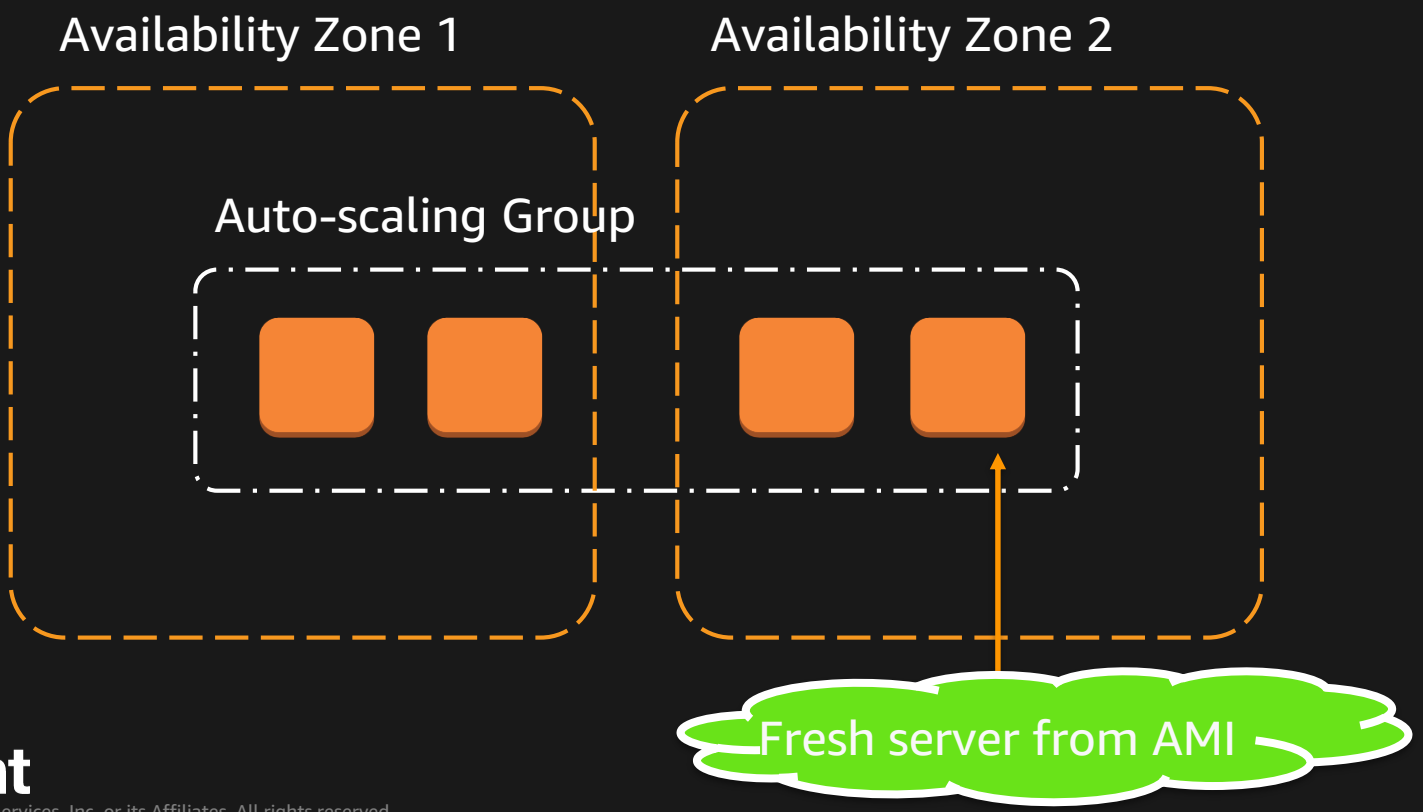
Instance ID,
private IP addresses,
Elastic IP addresses, and
all instance metadata

Instance ID,
private IP addresses,
Elastic IP addresses, and
all instance metadata

Loss of network connectivity
Loss of system power
Software issues on the physical host
Hardware issues on the physical host that impact network reachability

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html

High availability for EC2 – Auto Scaling



Understand why you might need Multi-Region Active-Active architecture

AWS re:Invent

aws

Serving geographically distributed customer base

USA West

Users from San Francisco

USA East

Users from New York

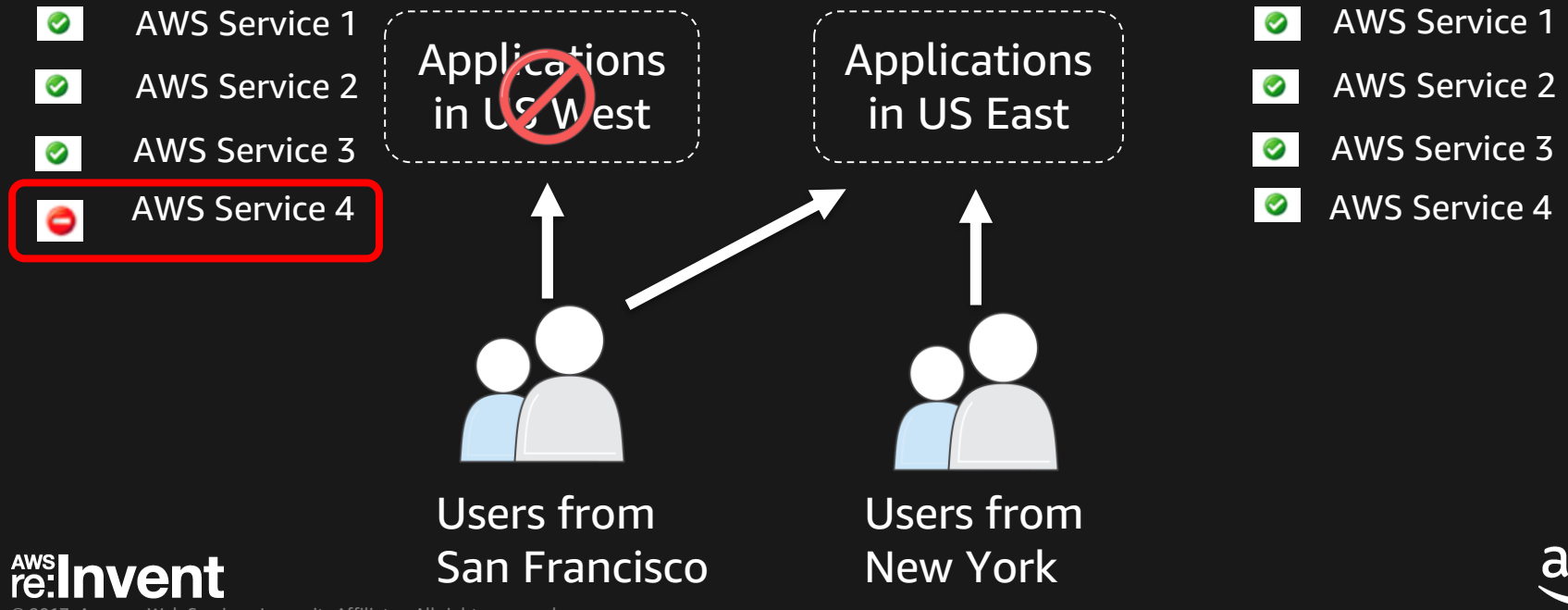
EU-East

Users from London

China

Users from Shanghai

Guarding against failure of your applications in one region



Cost effective DR: Why not use DR all the time?

DR environments that don't get used

1. Fall out of sync, eventually

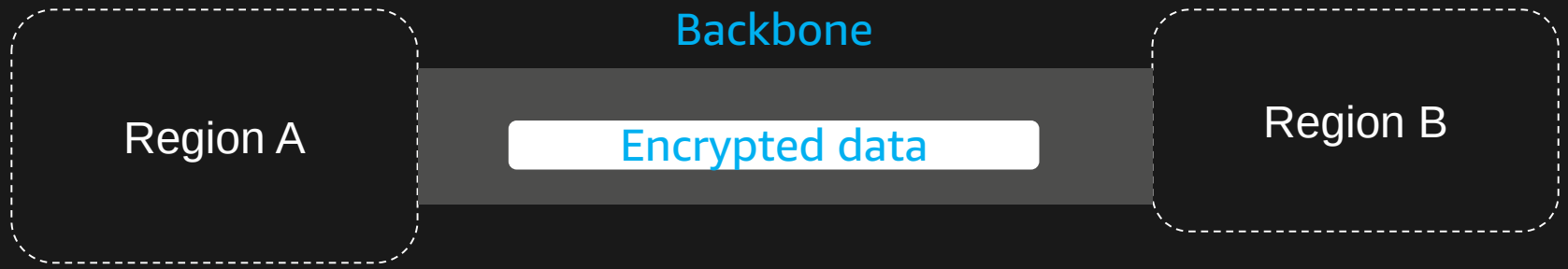
2. Waste money

Key Technology Requirements

Reliable & secure network

Network backbone

VPN or encrypted (SSL) communication over public IP



Data replication

1. Synchronous
2. Asynchronous
    - Nearly continuous (lag of few seconds or minutes)
    - Batch (hourly/daily/weekly copy)

Code synchronization

1. Staged deployment across regions
    - Rolling
    - Blue-Green
    - Rollback facility

2. Parameterized localization

Traffic segregation & management

Segregation options

Explicit – different URLs

- e.g. east.abc-corp.com and west.abc-corp.com

Implicit (DNS level) – the same URL

- e.g. www.abc-corp.com

Traffic management infrastructure

- Throttling
- Internal redirecting
- External redirecting

When users reach wrong regions

Monitoring

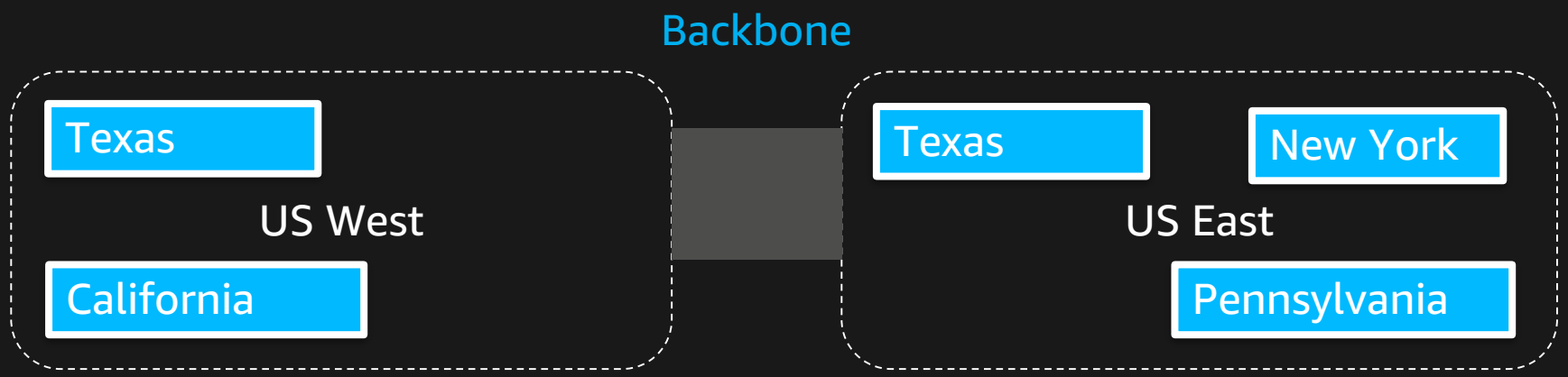
Application & infrastructure health

Replication lag & code sync monitoring

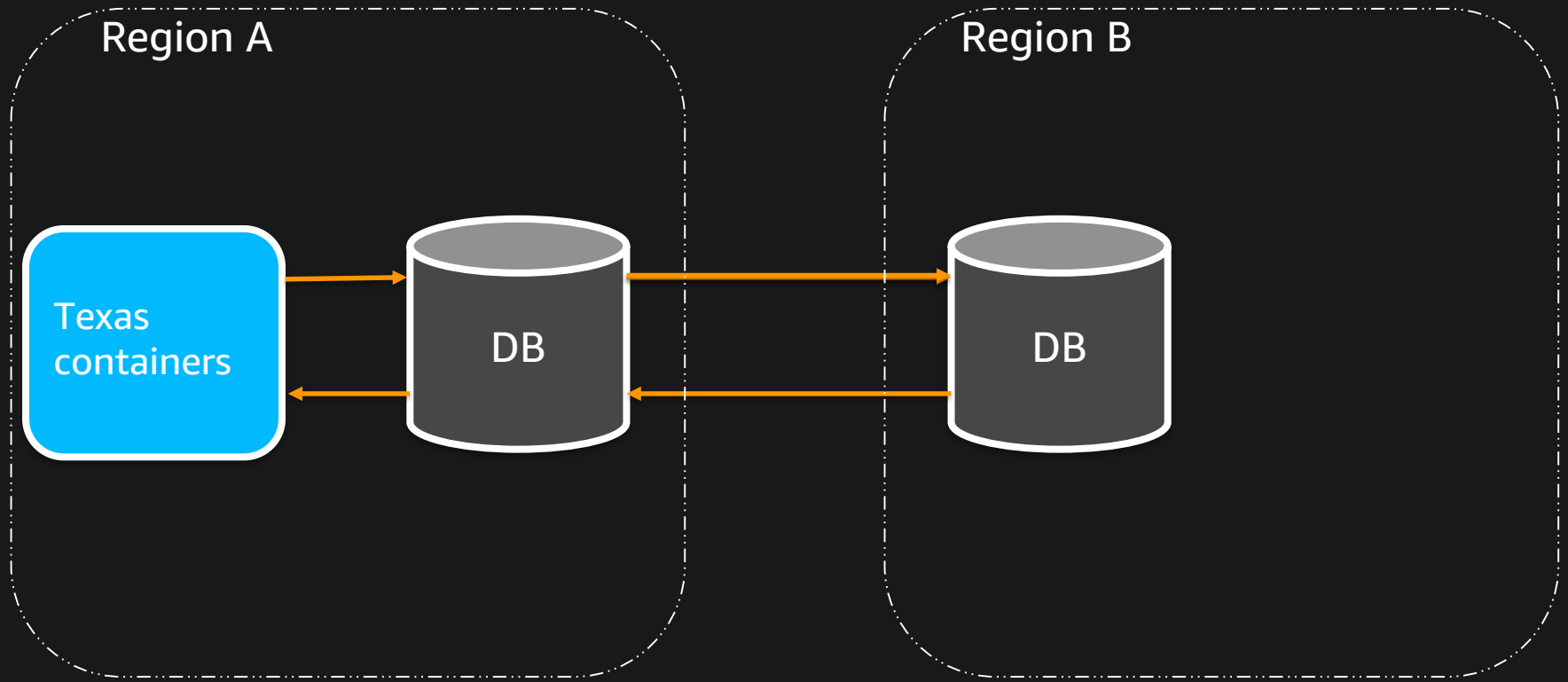
Multi-tenancy

What is a tenant?

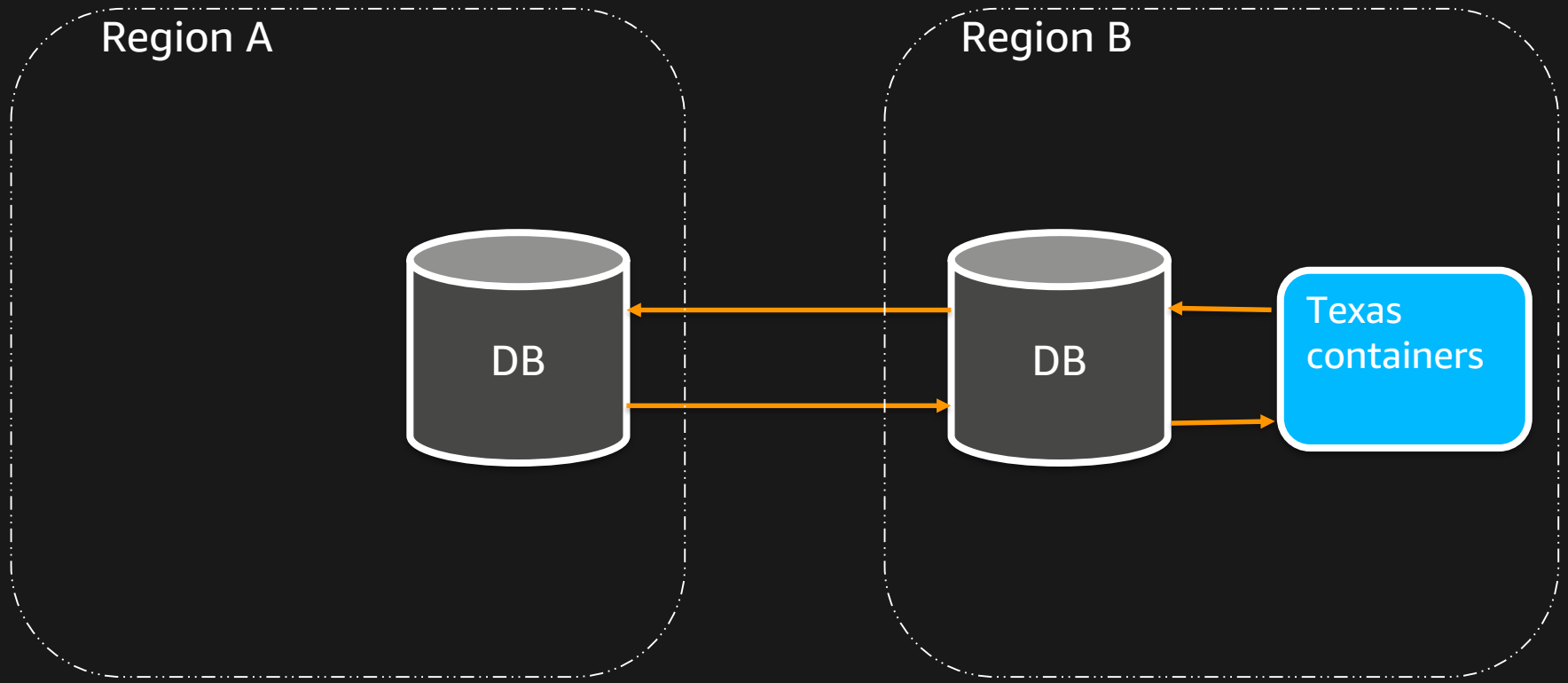
-A unit of movement/failover



Direction of replication – before



Direction of replication – after

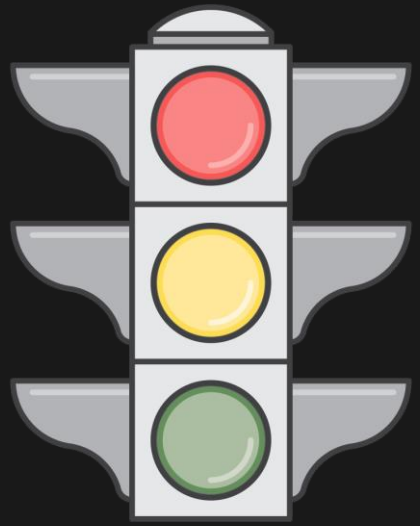


Failover scripts

Failover scripts should be able to:

1. Traffic rerouting for a tenant

2. Change direction of replication



Key architectural considerations

Tolerance for network partitioning

Failure of one region should not lead to failure of applications in another

Regional independence for request serving – no API calls from one region to another

Minimal data replication requirements

Does all data need to be replicated?

If yes, does it need to replicated synchronously?

Does all data need to be replicated continuously?

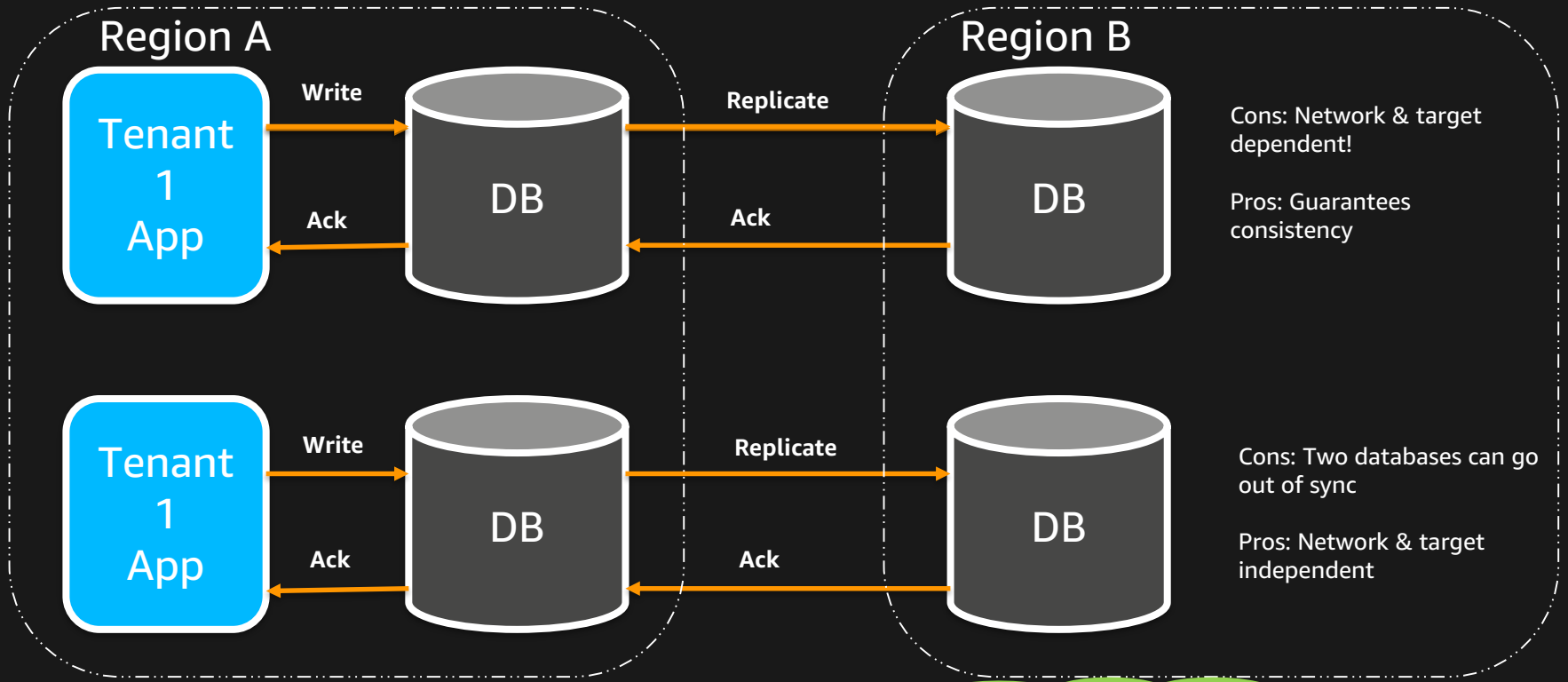
Classification of data

Transactions	Catalog information	Events, objects	Server logs
Purchase record	Product details	Click stream	Https logs

Low volume, but highly critical

High volume, but less critical

Sync vs. async replication modes



Async preferred, when possible!

Concept of data replication lanes

Synchronous replication — Transactions

Most difficult to manage

Asynchronous nearly continuous replication — Catalog information

Asynchronous batch replication — Events

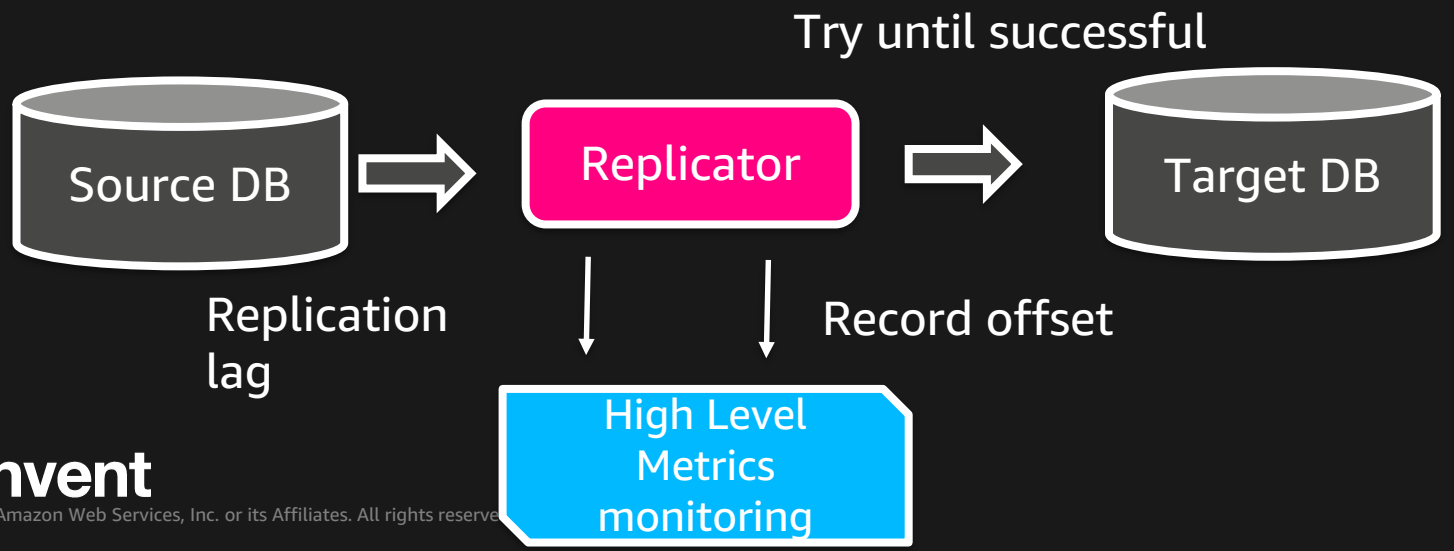
Easiest to manage

Ideal replication system

Each data store type will need a different technology.

But at the the minimum:

1. It should report replication lag
2. It should report record offset
3. Should be able to retry replication of failed records



Distributed system design best practices

Idempotency

Eventual consistency

Static stability

Exponential backup

Throttling

Circuit breaking

Reliability Pillar whitepaper (Nov 2017)

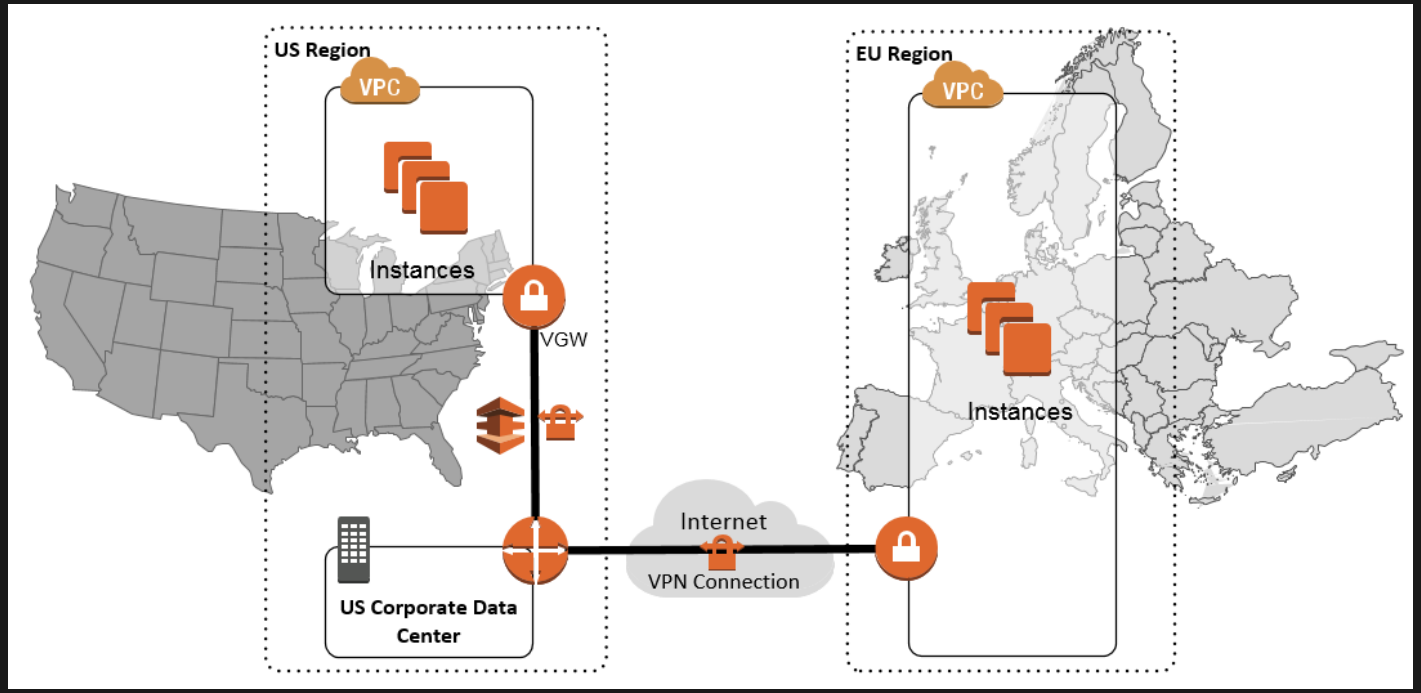
How AWS enables customers to deploy Multi-Region Active-Active Architecture

AWS re:Invent

aws

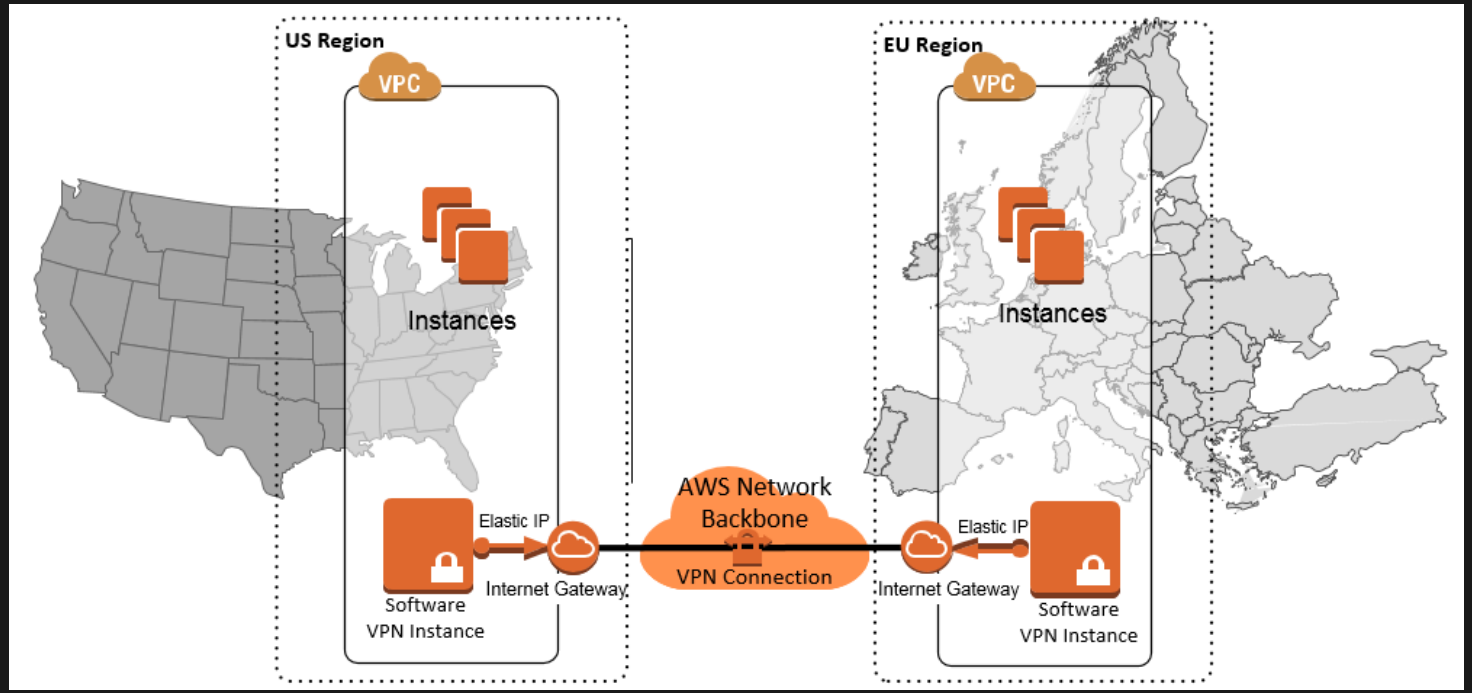
AWS worldwide network backbone

Multi-Region VPN – over non-AWS network



https://aws.amazon.com/answers/networking/aws-multiple-region-multi-vpc-connectivity/

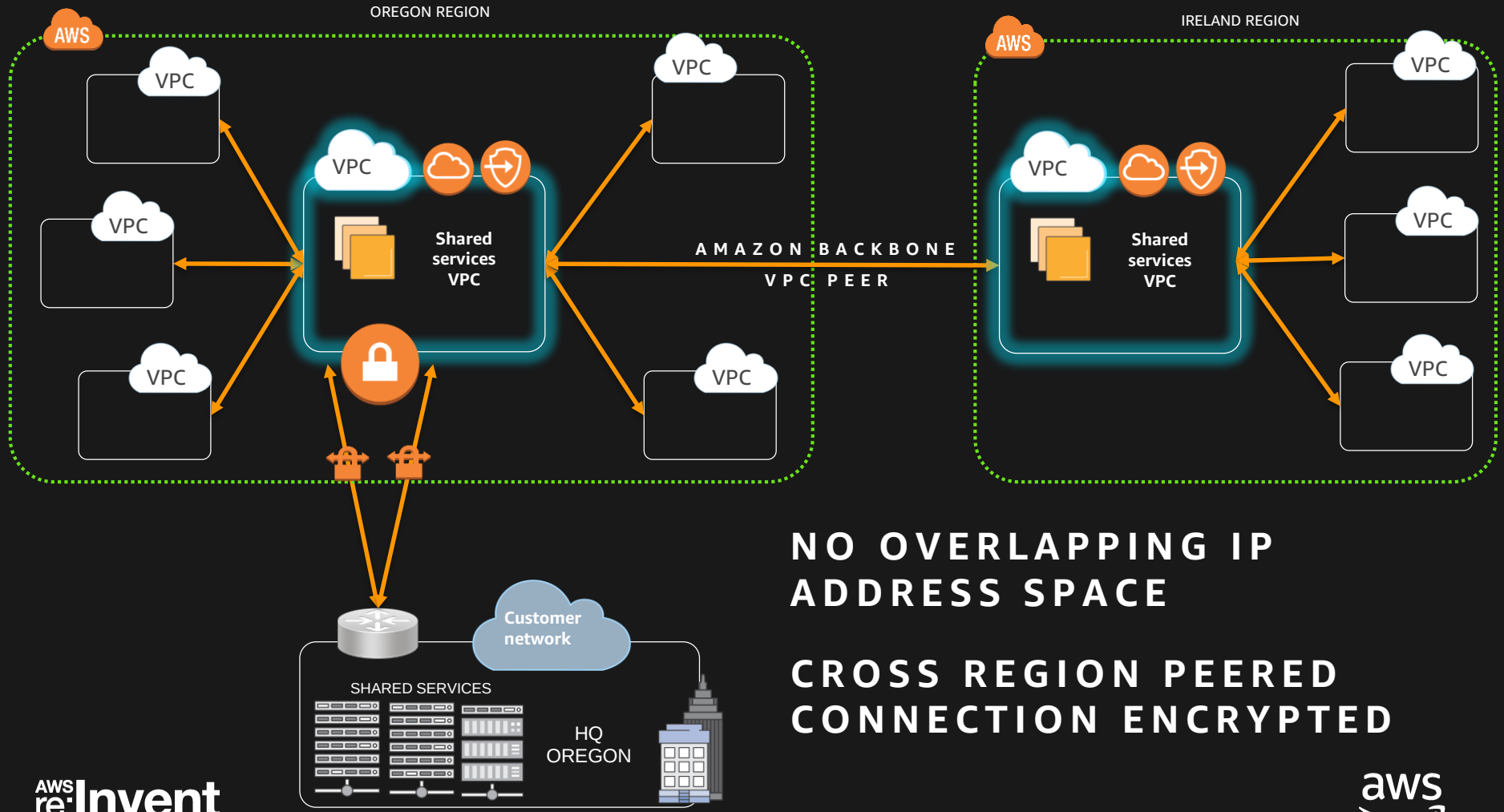
Multi-Region VPN – over AWS network



https://aws.amazon.com/answers/networking/aws-multiple-region-multi-vpc-connectivity/

NEW

INTER REGION VPC PEERING



NO OVERLAPPING IP ADDRESS SPACE

CROSS REGION PEERED CONNECTION ENCRYPTED

AWS re:Invent

aws

Key benefits

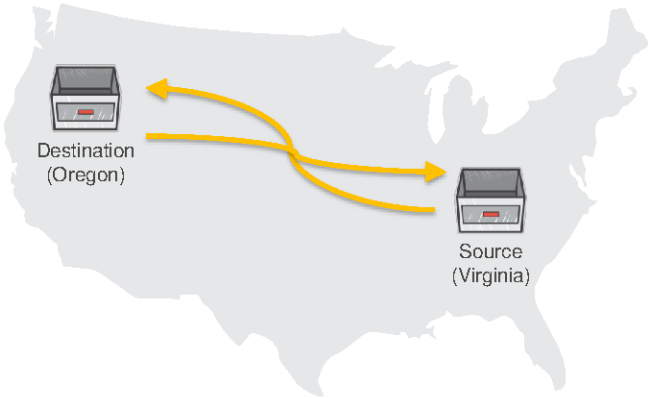
1. Works similar to existing Intra Region VPC Peering
2. Data always stays on the AWS backbone
3. Data always encrypted by default
4. No need to use Gateways, third-party VPN solutions to connect across regions.
5. No additional charges for using interregion VPC peering. Customers pay standard data transfer rates

S3 – cross-region replication

Automated, fast, and reliable asynchronous replication of data across AWS regions

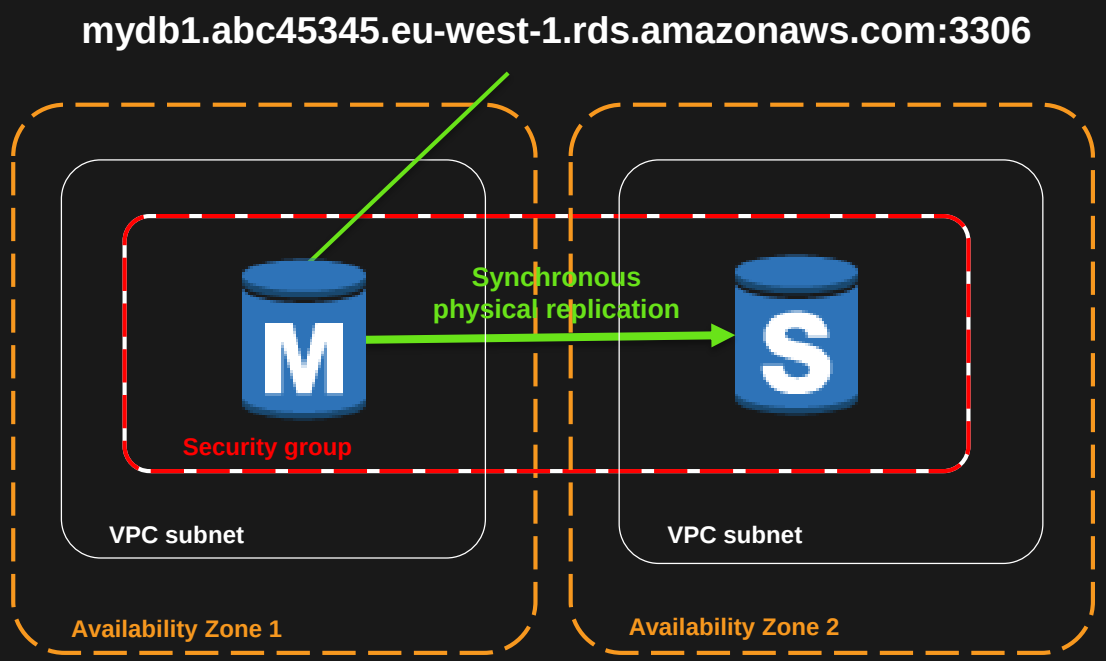
Use cases:

- Compliance—store data hundreds of miles apart
- Lower latency—distribute data to regional customers
- Security—create remote replicas managed by separate AWS accounts



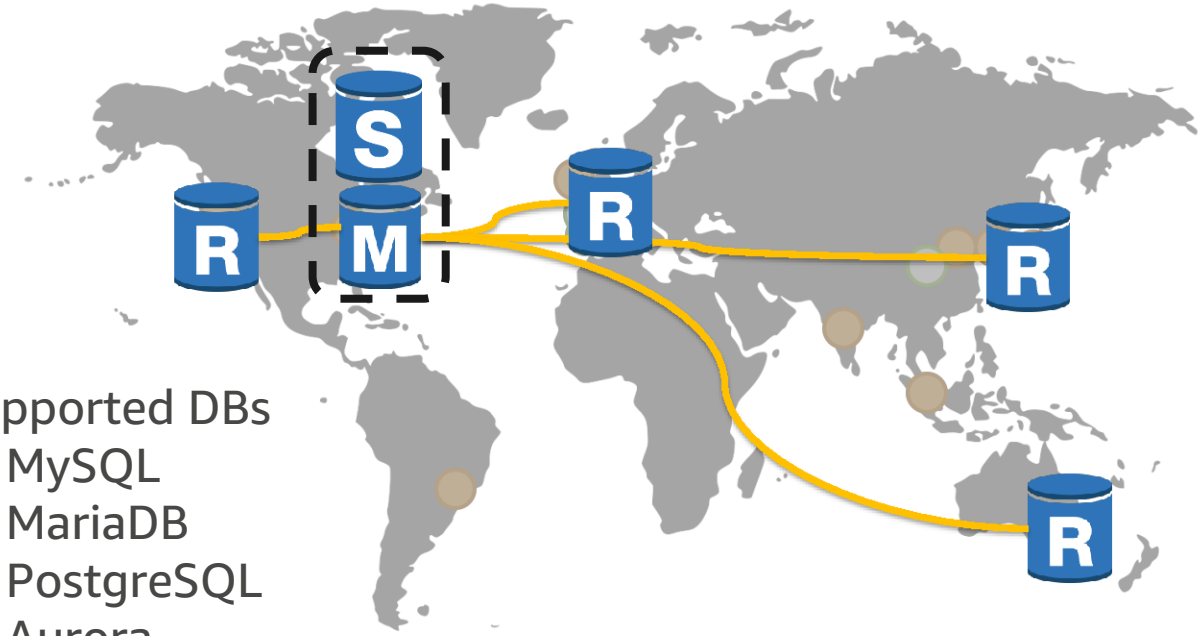
- Only replicates new PUTs. Once S3 is configured, all new uploads into a source bucket will be replicated
- Entire bucket or prefix based
- 1:1 replication between any 2 regions / storage classes
- Transition S3 ownership from primary account to sub-account

Amazon RDS Multi-AZ deployment



- Standbys ensure zero data loss in event of the master's failure.
- Always have a stand-by. Always!!!
- Also note, cross AZ failovers are automatic & fast; whereas cross-region failovers take time and need nontrivial planning.

RDS cross-region replication

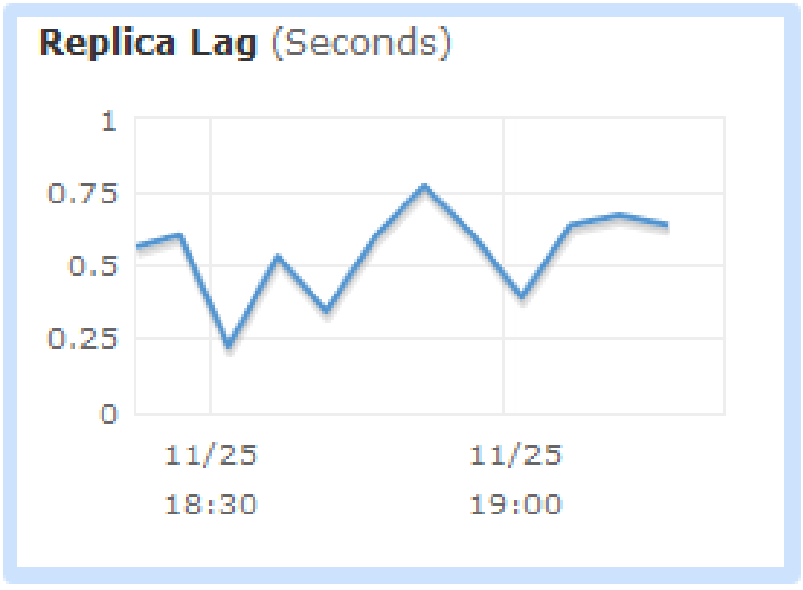


Supported DBs

- MySQL
- MariaDB
- PostgreSQL
- Aurora

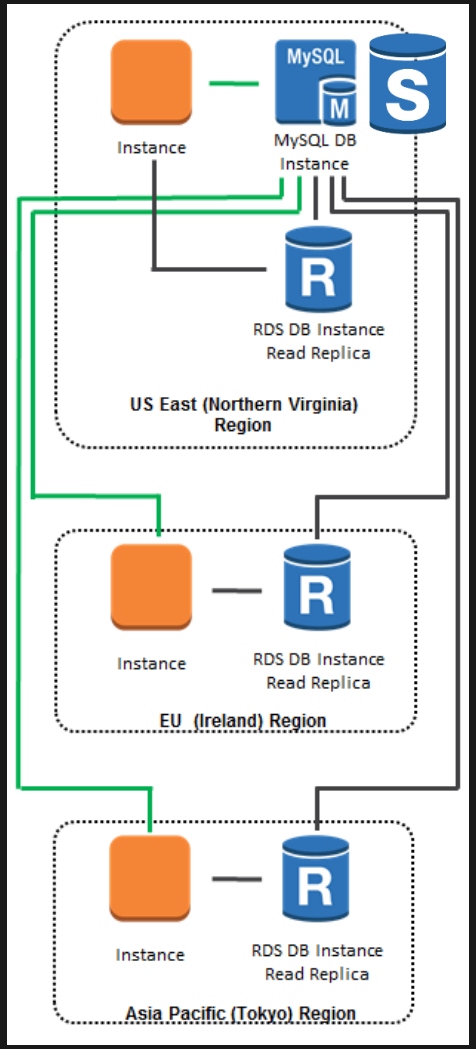
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html#USER_ReadRepl.XRgn
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/AuroraMySQL.Replication.CrossRegion.html

Monitoring RDS replication lag



Very simple plan

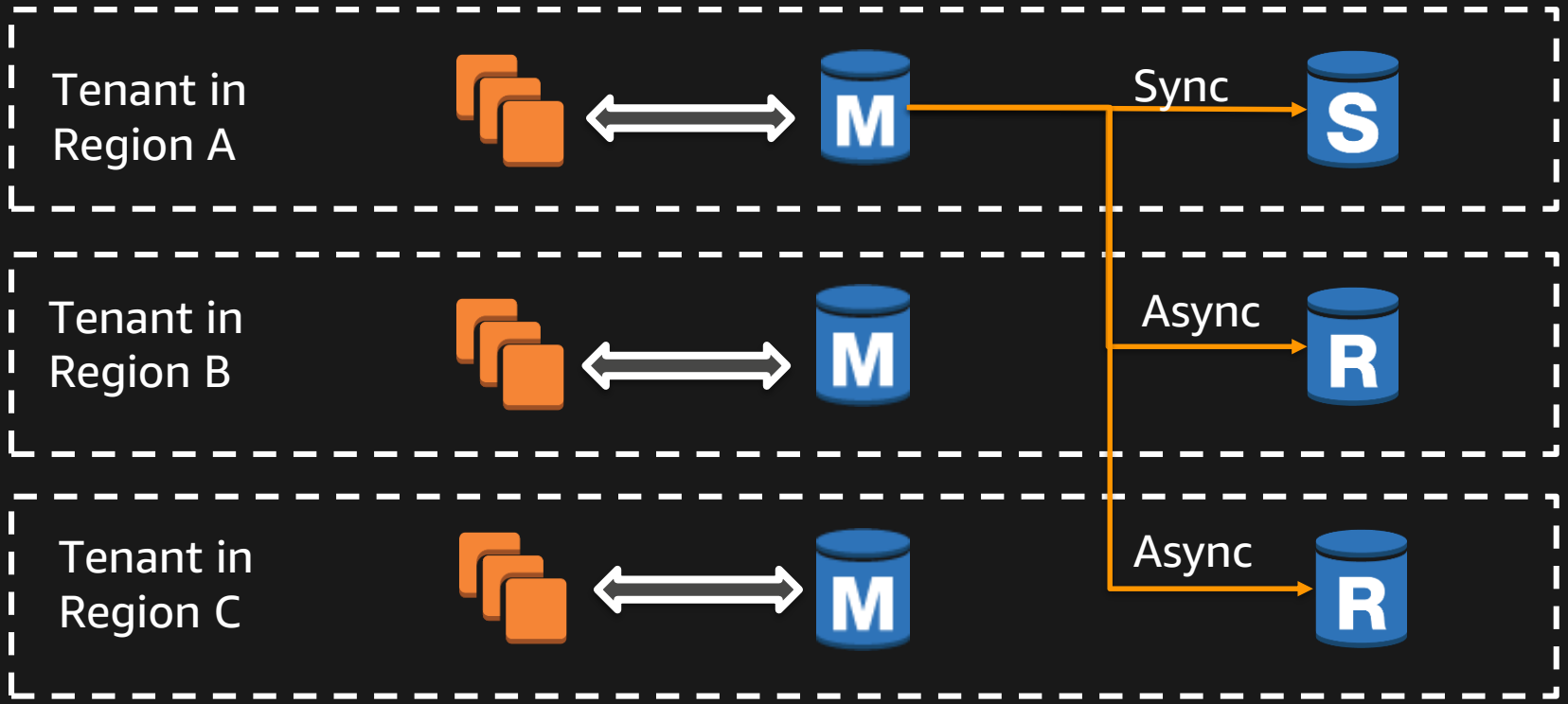
All regions send critical write traffic to a single master



Replication traffic

“Simple, but higher latency and network dependency”

Better plan (for most cases)



Applications get faster response

Some committed transactions may not make it to the failover region, before the switch.

However

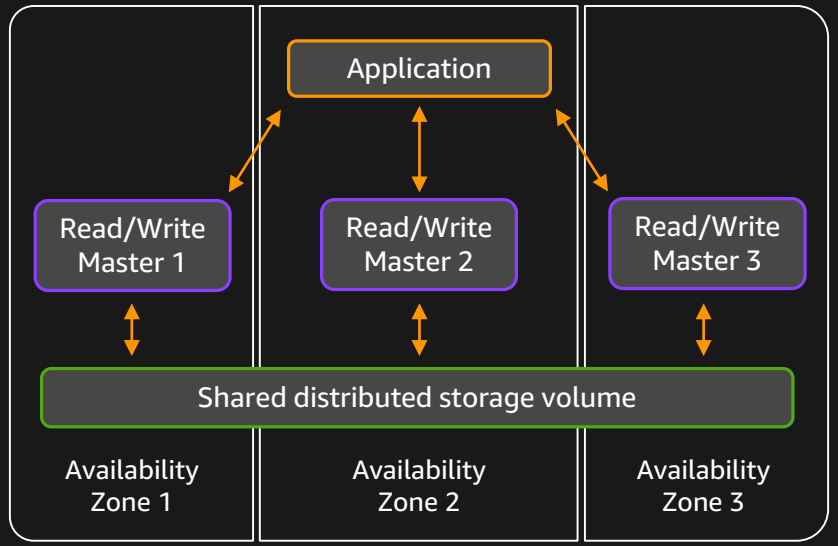
Committed transactions are safe due to standby.
They can be recovered with help of reconciliation techniques.

Aurora multi-master—scale out reads & writes

NEW!

First MySQL compatible DB service with scaleout across multiple data centers

Scale out both reads **and writes**



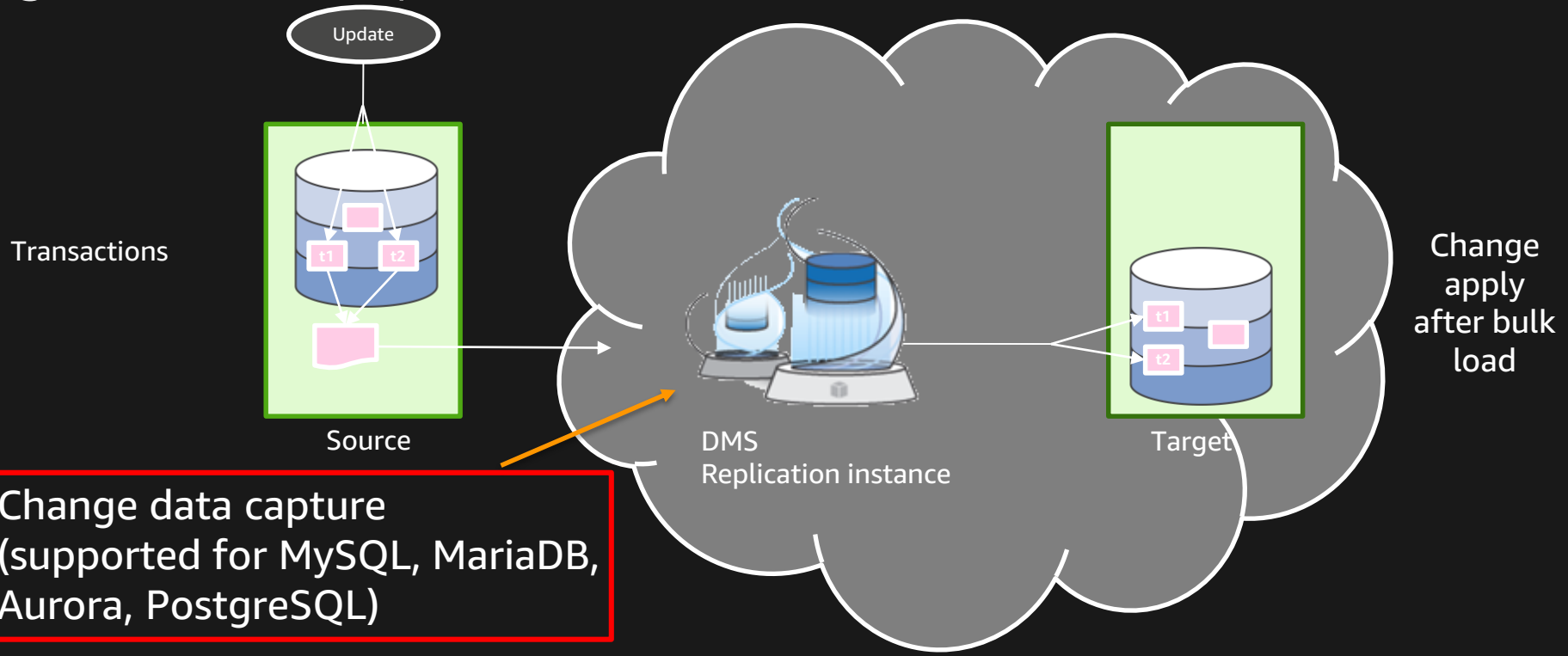
Zero application downtime from ANY instance failure

Zero application downtime from ANY AZ failure

Faster write performance and higher scale

Sign up for single-region multi-master preview today;
Multi-Region Multi-Master coming in 2018

Database Migration Service (DMS) for granular replication



Change data capture
(supported for MySQL, MariaDB, Aurora, PostgreSQL)

Details: http://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.html

Amazon DynamoDB

Fast and flexible NoSQL database service for any scale

Highly scalable

Auto-scaling to hundreds of terabytes of data that serve millions of requests per second

Fast, consistent performance

Consistent single-digit millisecond latency; DAX in-memory performance reduces response times to microseconds

Fully managed

Automatic provisioning, infrastructure management, scaling, and configuration with zero downtime

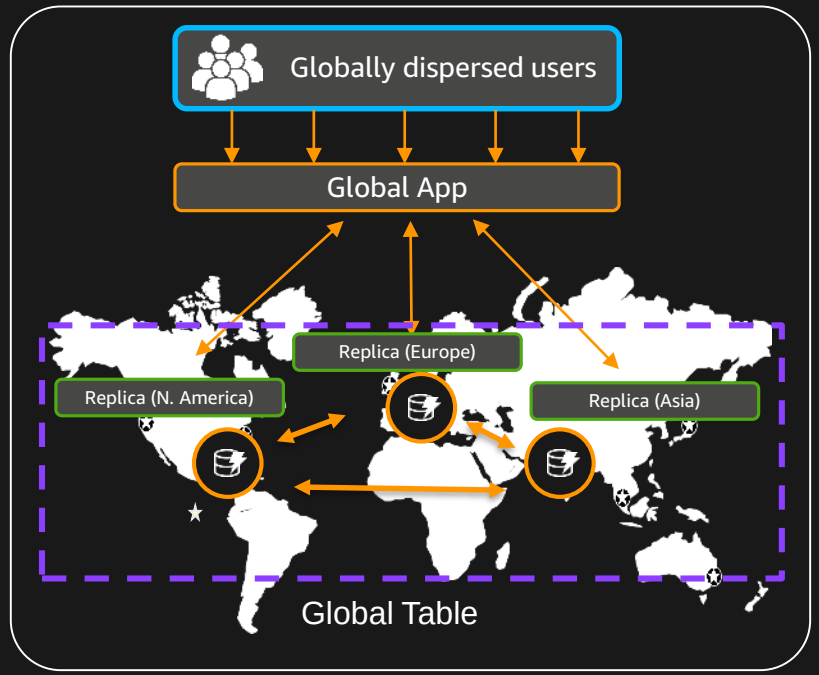
Business critical reliability

Data is replicated across fault tolerant Availability Zones, with fine-grained access control

NEW!

Amazon DynamoDB Global Tables (GA)

First fully managed, multi-master, multi-region database

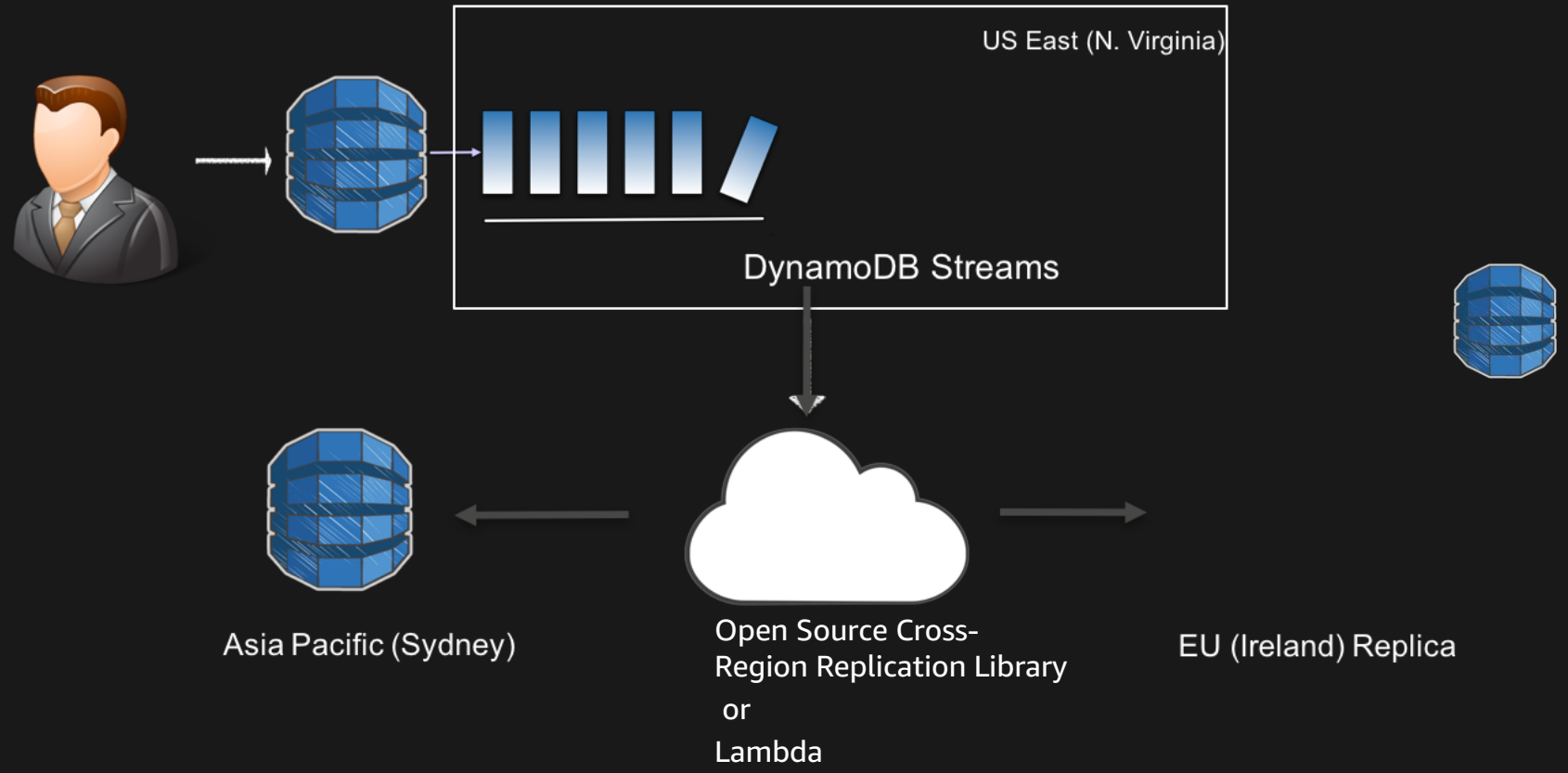


- Build high performance, globally distributed applications
- Low latency reads & writes to locally available tables
- Disaster proof with multi-region redundancy
- Easy to set up and no application rewrites required

AWS re:Invent

aws

Amazon DynamoDB cross-region replication with streams

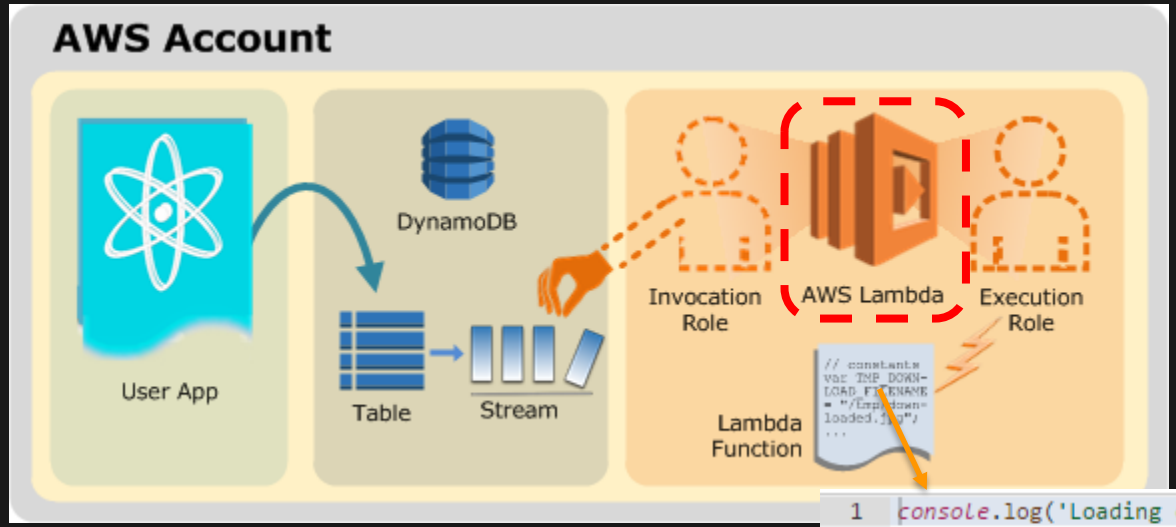


What is AWS Lambda

- It is a serverless, event driven compute service
- Define functions, and the service executes them as many times as input arrives
- Many supported event sources including: Kinesis and DynamoDB streams
- Scales automatically with event rate
- Supported languages: Java, .Net, Node.js, Python

Ensures streamed records in shards are processed (replicated in this case) in order!

DynamoDB Streams and AWS Lambda




```
1  console.log('Loading event');
2  exports.handler = function(event, context) {
3    console.log("Event: %j", event);
4    for(i = 0; i < event.Records.length; ++i) {
5      record = event.Records[i];
6      console.log(record.EventID);
7      console.log(record.EventName);
8      console.log("DynamoDB Record: %j", record.Dynamodb);
9    }
10   context.done(null, "Hello World");  // SUCCESS with message
11 }
```

```
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff INSERT
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff DynamoDB Record:{ "NewImage": { "name": { "S": "sivar" }, "hk": { "S": "3" } }, "SizeBytes": 15, "StreamViewType": "NEW_AND_OLD_IMAG
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff Message: "Hello World"
```

Error handling in Lambda

1. For stream-based event sources (Amazon Kinesis Streams and DynamoDB streams), AWS Lambda polls your stream and invokes your Lambda function.

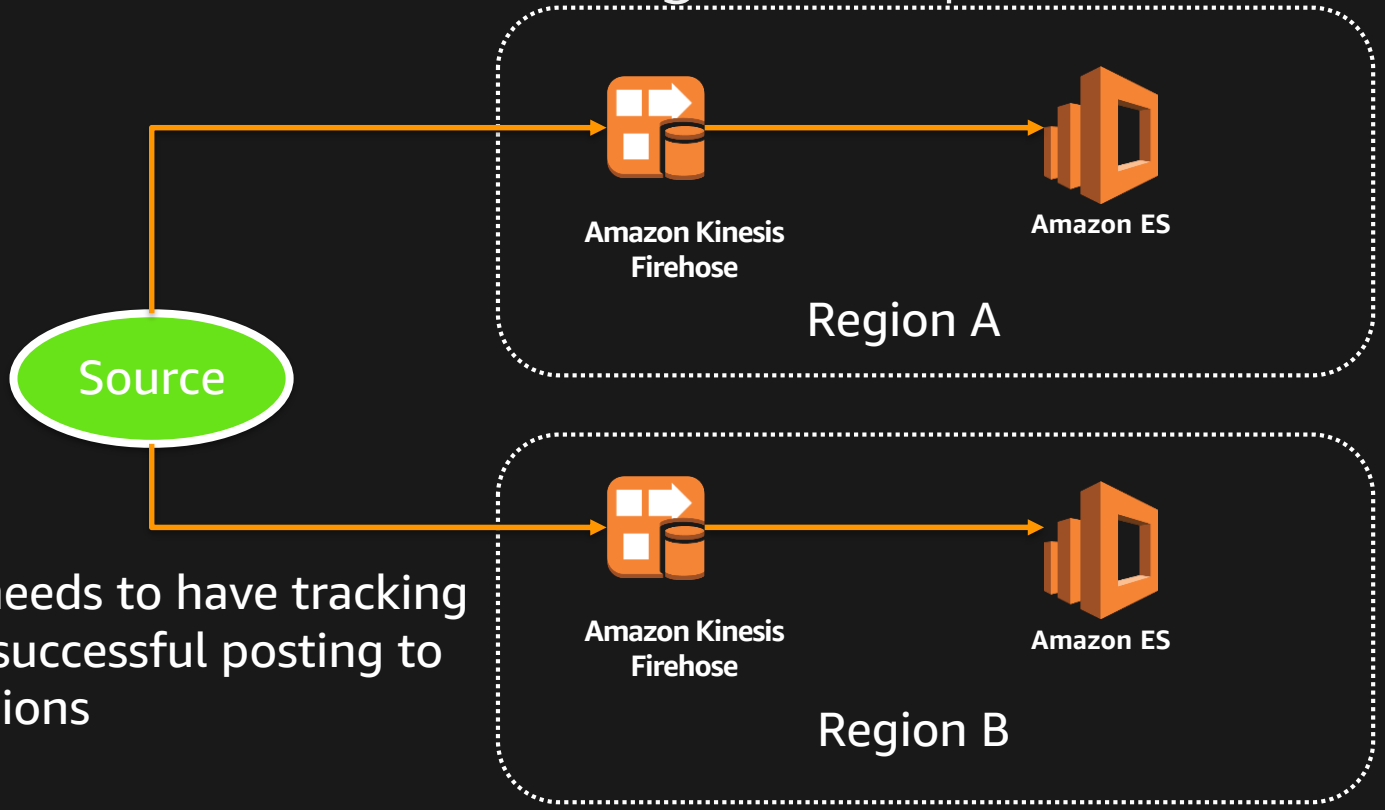
2. Therefore, if a Lambda function fails, AWS Lambda attempts to process the erring batch of records until the time the data expires from the stream.

3. The exception is treated as blocking, and AWS Lambda will not read any new records from the stream until the failed batch of records either expires or processed successfully.

This ensures that AWS Lambda processes the stream events in order.

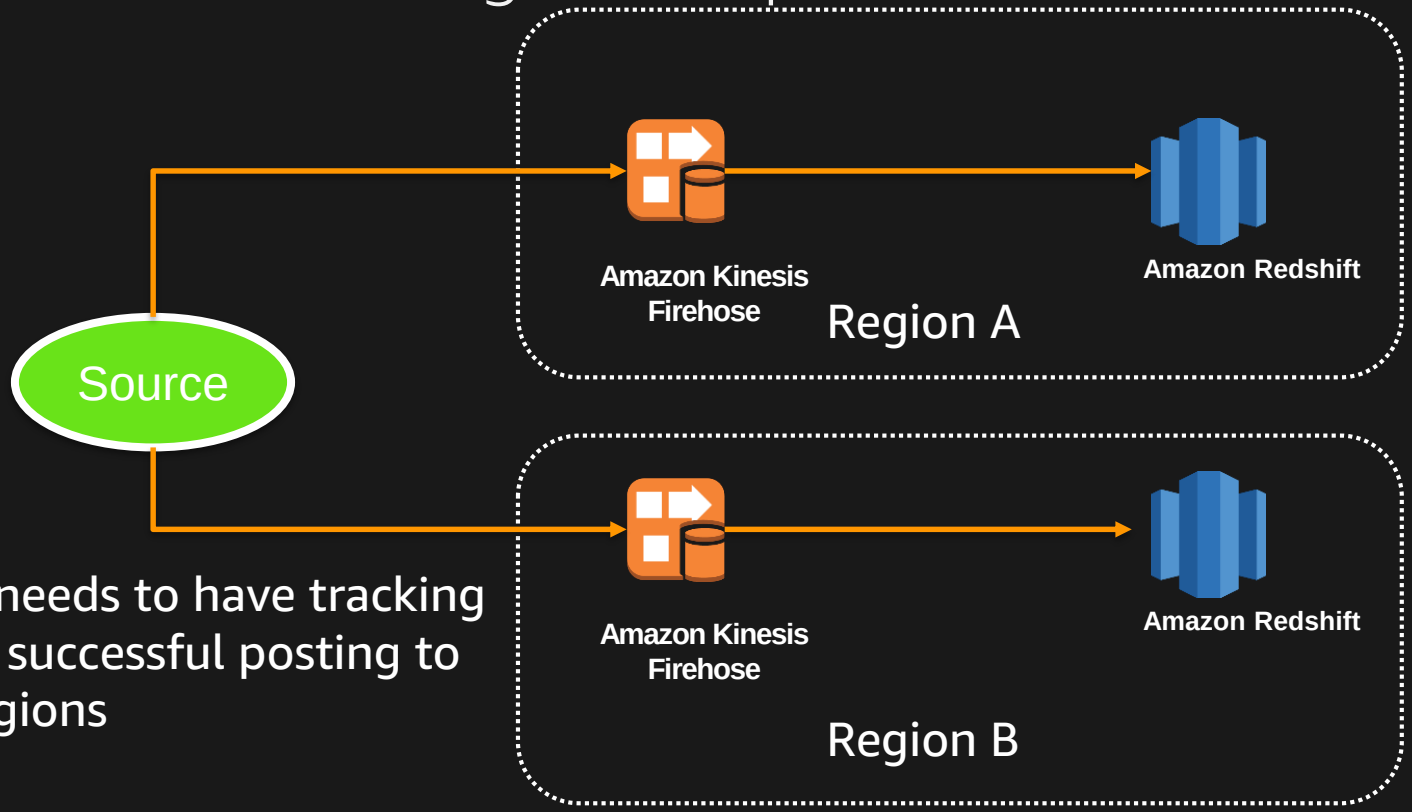
http://docs.aws.amazon.com/lambda/latest/dg/retries-on-errors.html

ElasticSearch cross region replication



Source needs to have tracking to have successful posting to both regions

Redshift cross region replication



Source needs to have tracking to have successful posting to both regions

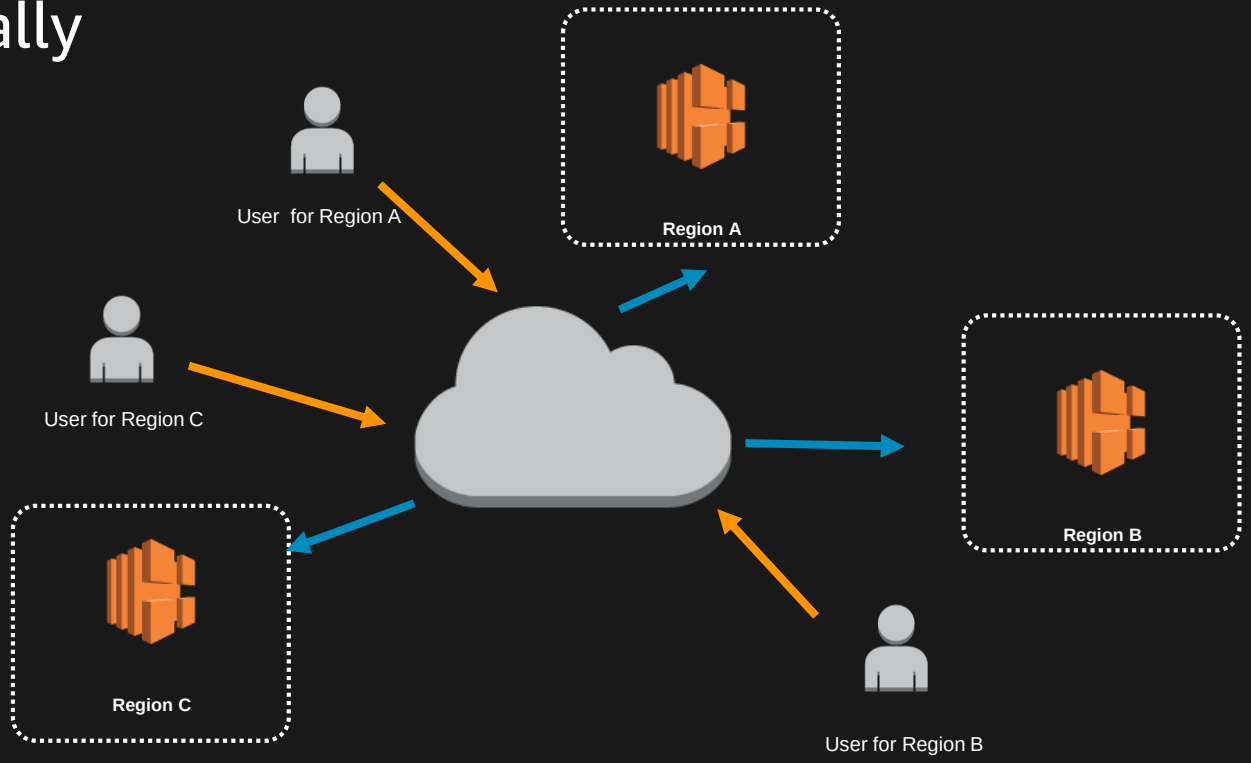
Global Traffic Management with Route 53

AWS re:Invent

aws

Global service

Load balancers and/or servers across multiple regions globally



Traffic flow: endpoints

- Hybrid/low level infrastructure: IP address or CNAME
- ELB Classic Load Balancer / Application Load Balancer
- Amazon S3 website
- Amazon CloudFront distribution
- AWS Elastic Beanstalk environment

Traffic flow: Quick overview of rules

Simple routing policy – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.

Failover routing policy – Use when you want to configure active-passive failover.

Geolocation routing policy – Use when you want to route traffic based on the location of your users.

Traffic flow: Quick overview of rules

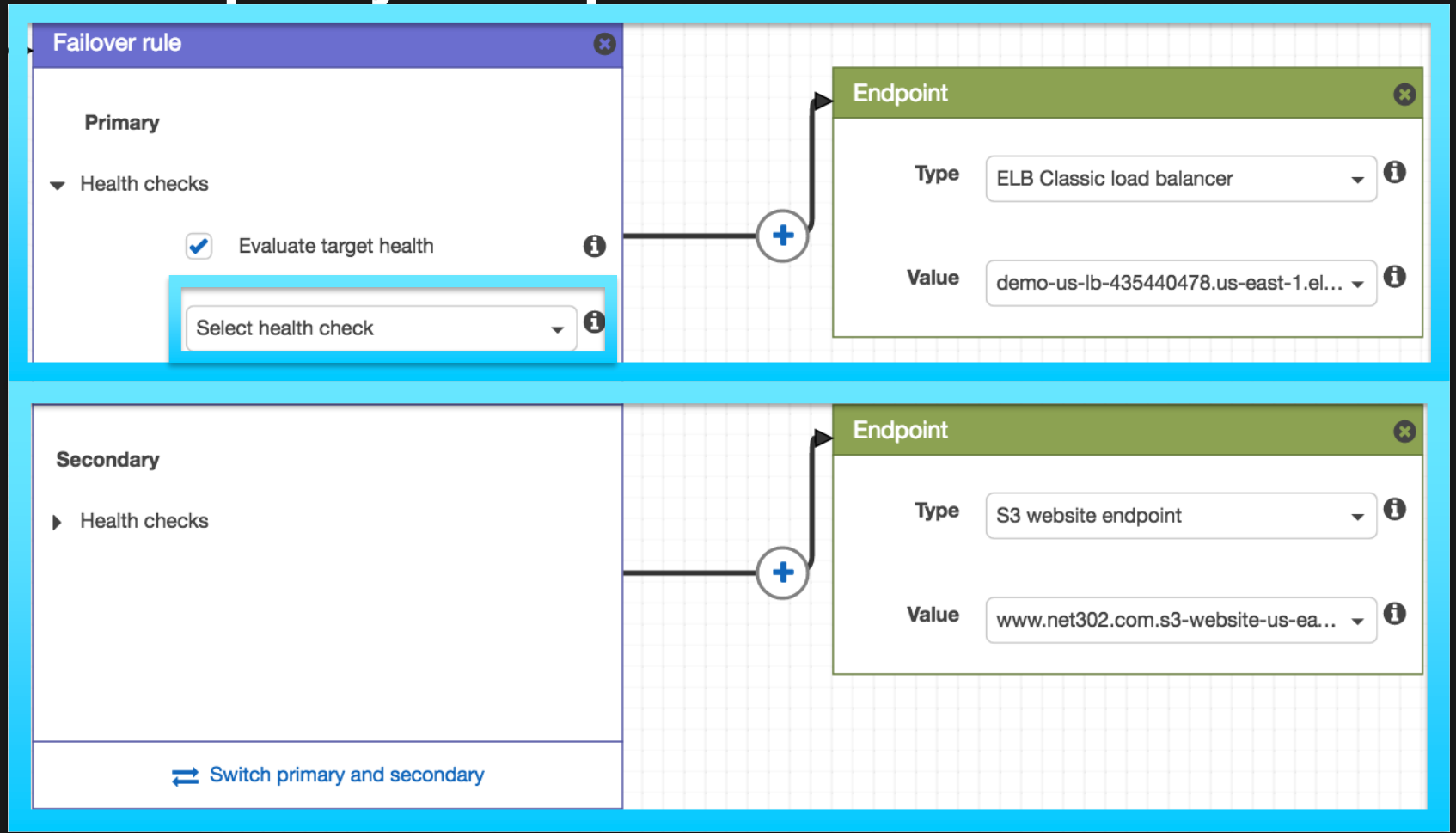
Geoproximity routing policy – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from one resource in one location to resources in another.

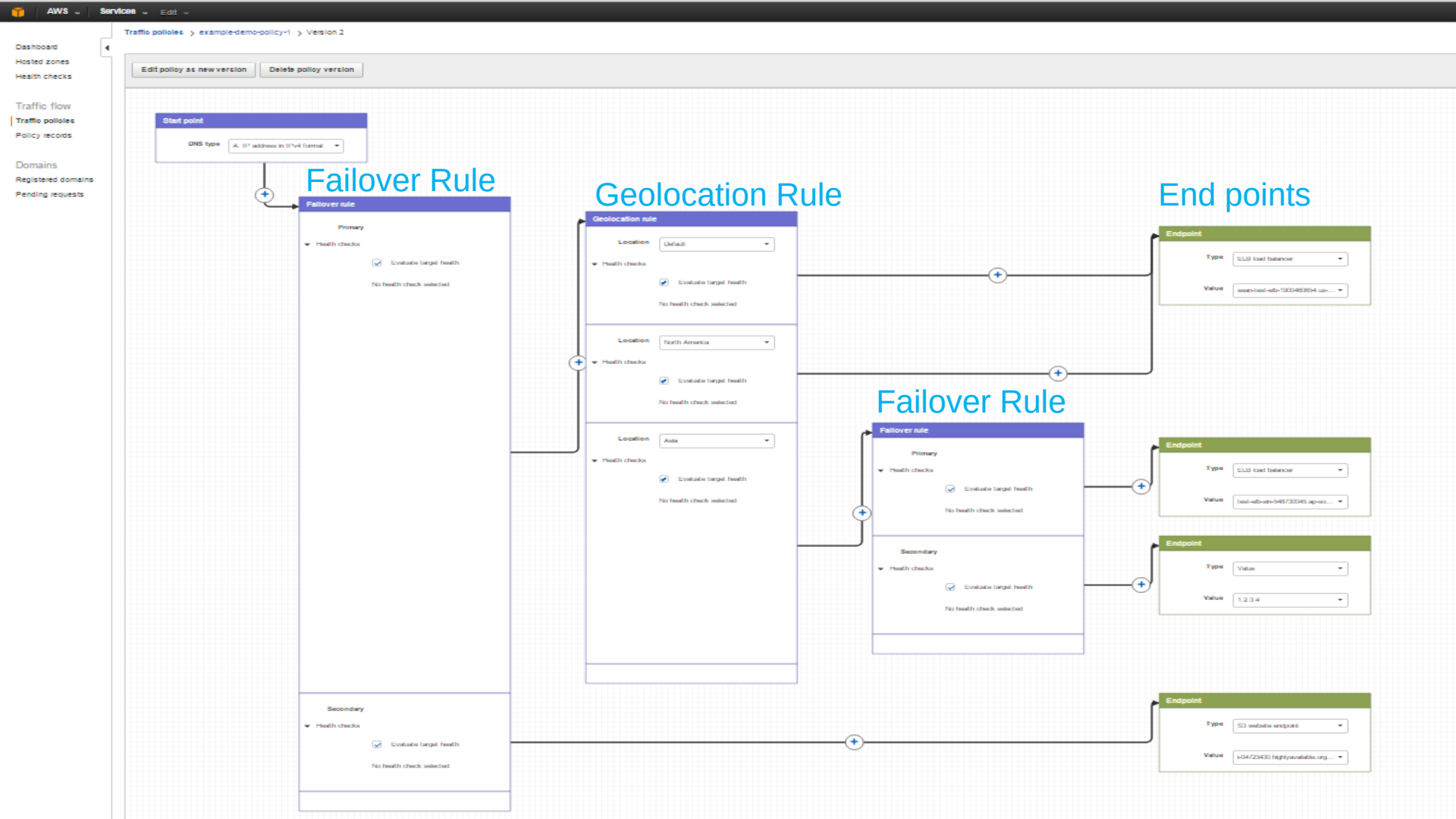
Latency routing policy – Use when you have resources in multiple locations and you want to route traffic to the resource that provides the best latency.

Multivalue answer routing policy – Use when you want Amazon Route 53 to respond to DNS queries with up to eight healthy records selected at random.

Weighted routing policy – Use to route traffic to multiple resources in proportions that you specify.

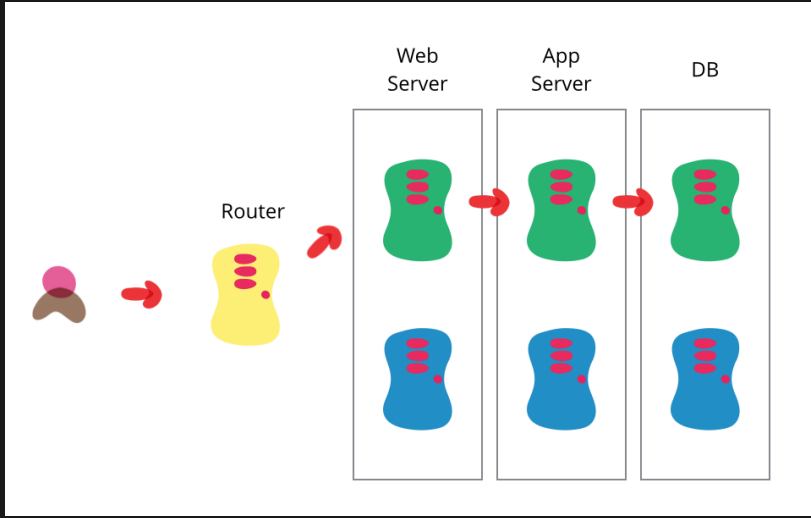
Traffic policy example




AWS
Services
Edit
Traffic policies > example-demo-policy-1 > Version 2
Dashboard
Hosted zones
Health checks
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Pending requests
Edit policy as new version
Delete policy version
Start point
DNS type
Failover Rule
Failover rule
Primary
Health checks
Evaluate target health
No health check selected
Secondary
Geolocation Rule
Geolocation rule
Location
Default
North America
Asia
End points
Endpoint
Type
ELB load balancer
Value
Failover Rule
Failover rule
Primary
Secondary
Endpoint
Type
Value
1.2.3.4
Endpoint
Type
S3 website endpoint
Value

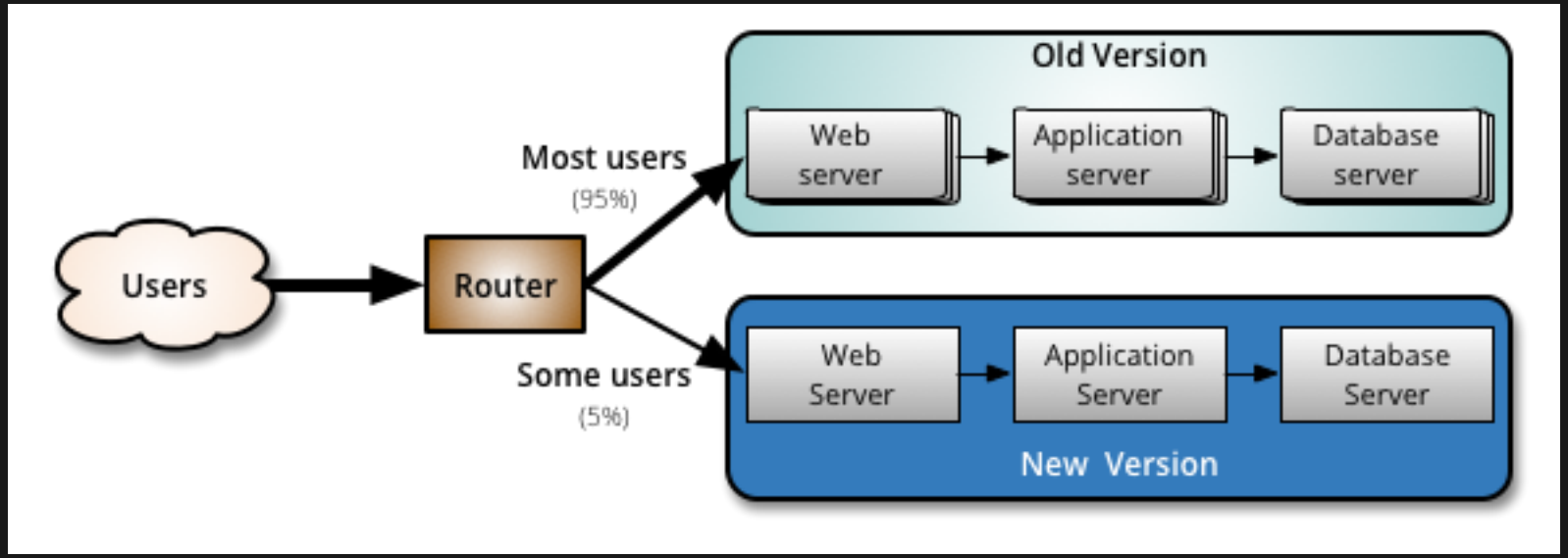
Cross-region code deployment

BLUE-GREEN



Link

CANARY



Link

Is code deployment any different?

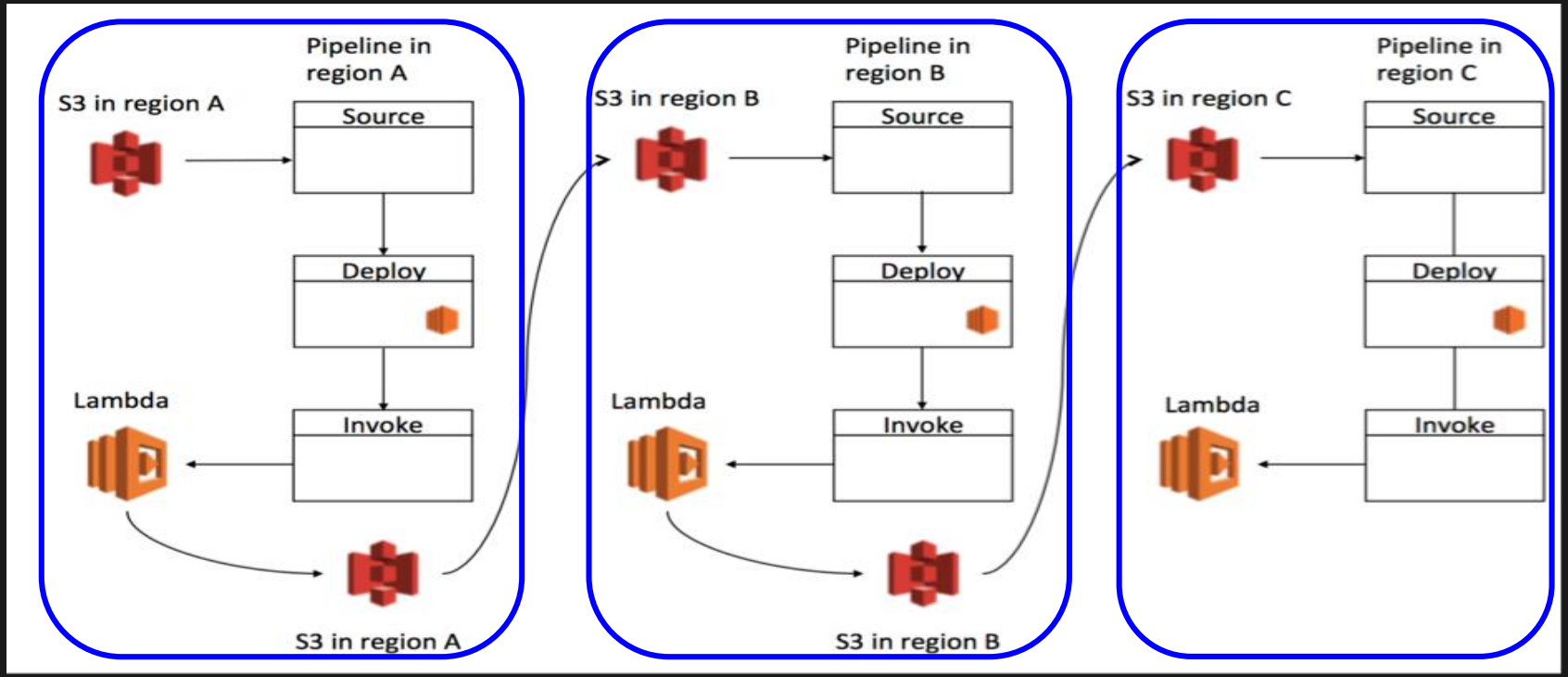
No. DevOps pipelines work the same way.

Important trade-off you should make:

1. Simultaneous deployments 

2. “One region at a time” deployments 

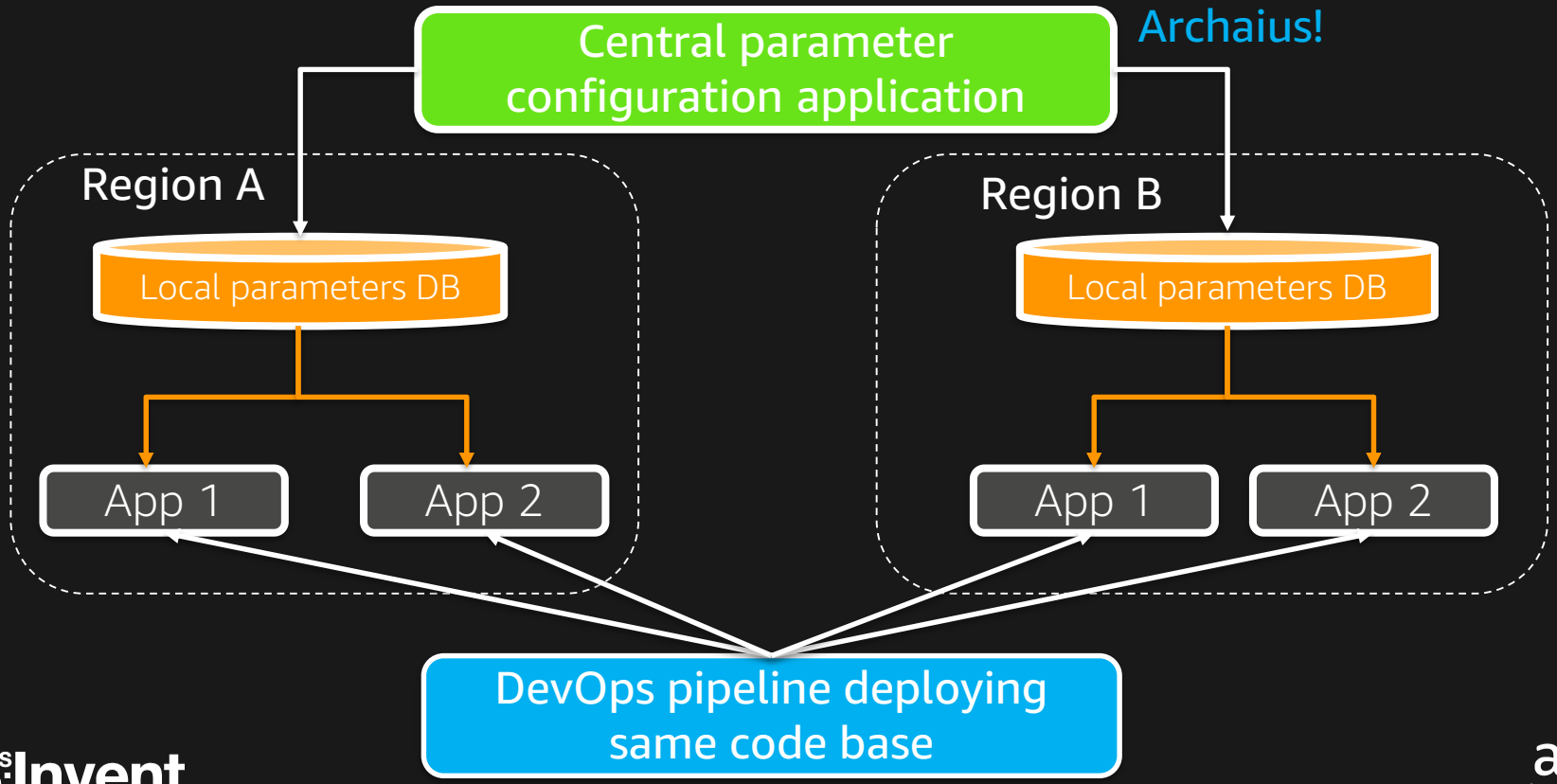
Using AWS services



https://aws.amazon.com/blogs/devops/building-a-cross-regioncross-account-code-deployment-solution-on-aws/

Parameter localization

Take a look at Netflix Archaius!



Cross Region Monitoring

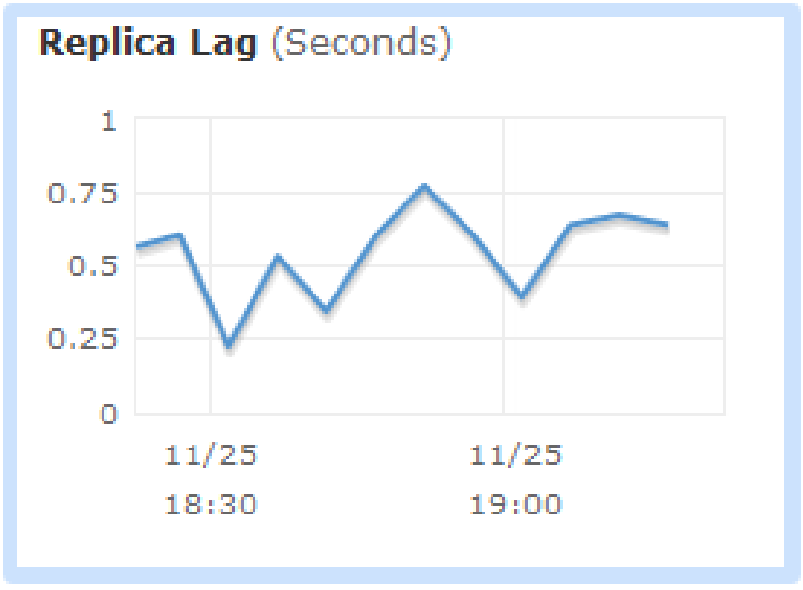
Is monitoring any different?

Yes. You need to do additional monitoring.

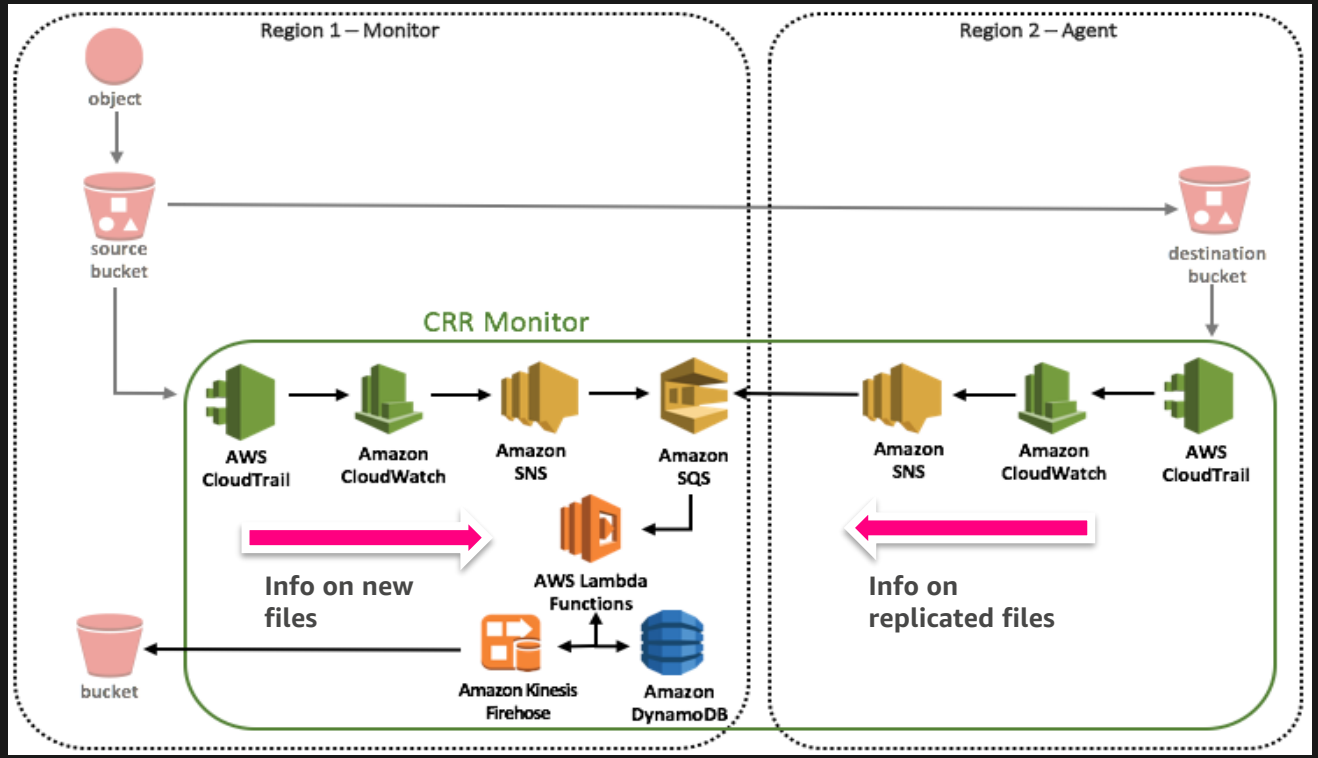
1. Replication lags (very critical)
2. Record offset tracking

Need to put important stats for two/more regions on a single canvas for proper monitoring.

Monitoring replication lag – CloudWatch

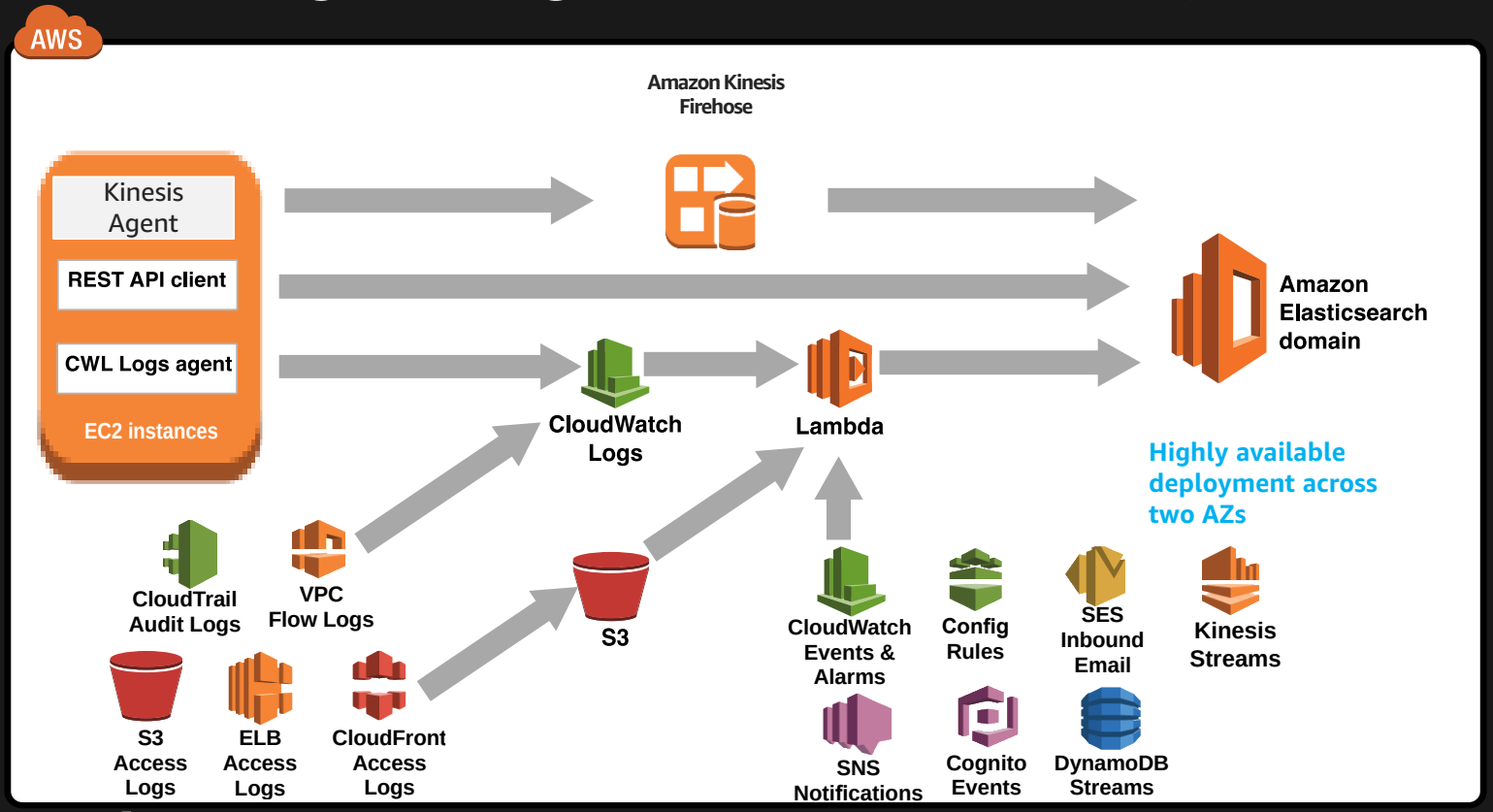


Monitoring Amazon S3 file replication progress



https://aws.amazon.com/answers/infrastructure-management/crr-monitor/

Monitoring using serverless components



Arrow direction indicates general direction of data flow

https://d0.awsstatic.com/whitepapers/whitepaper-use-amazon-elasticsearch-to-log-and-monitor-almost-everything.pdf

Not all metrics are created equal!

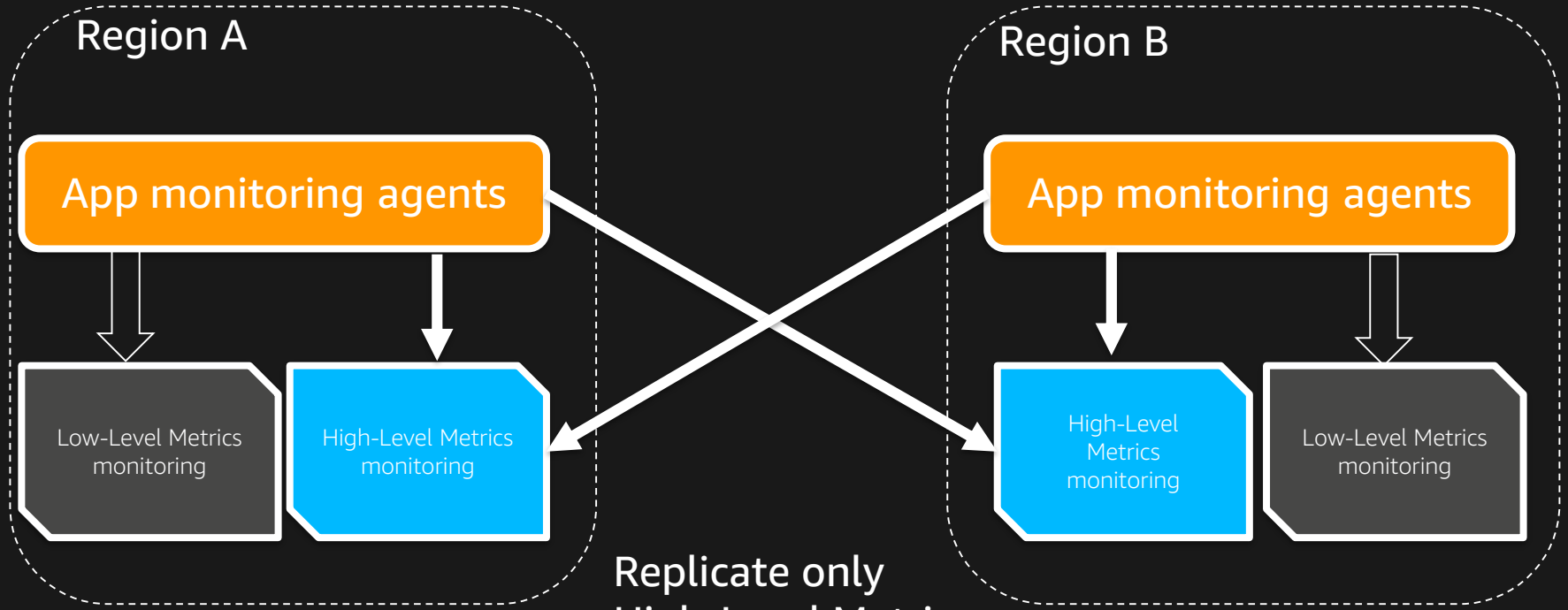
High-Level Metric (high importance, low volume):

- User experience
- User count
- Transaction count
- Replication status

Low-Level Metric (relatively low importance, high volume):

- HTTP request count
- Read vs. write throughout
- Cache hit vs. miss

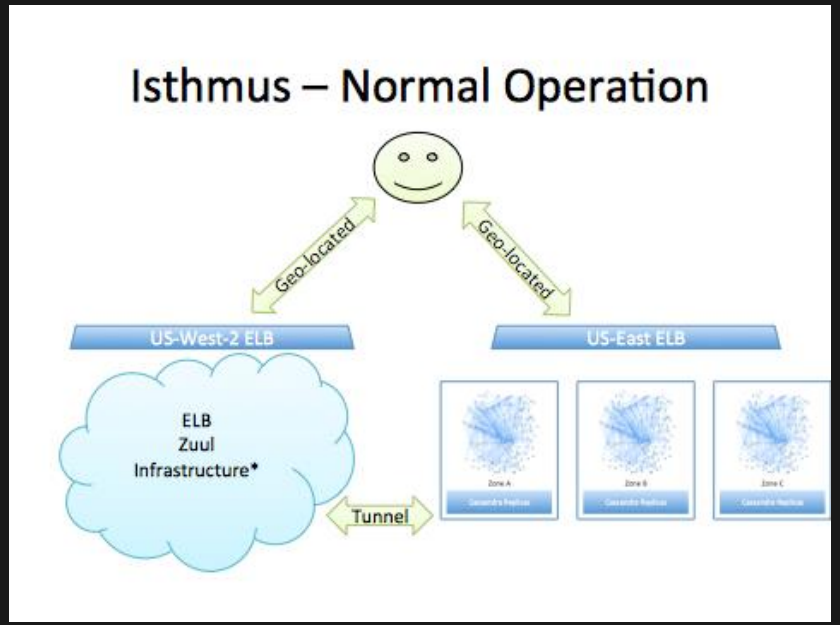
High-Level Metrics monitoring

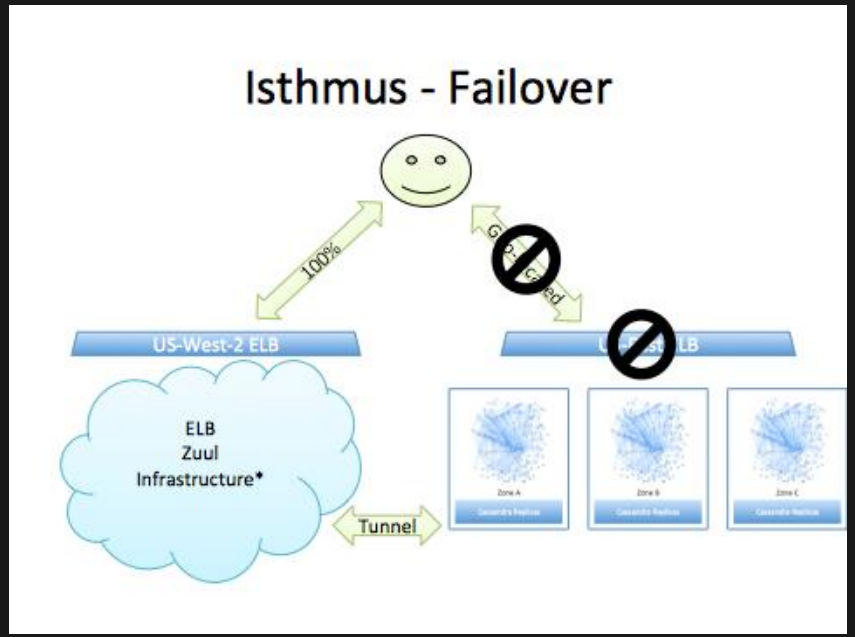


Replicate only
High-Level Metrics.
Use region tags.

Some useful open source projects

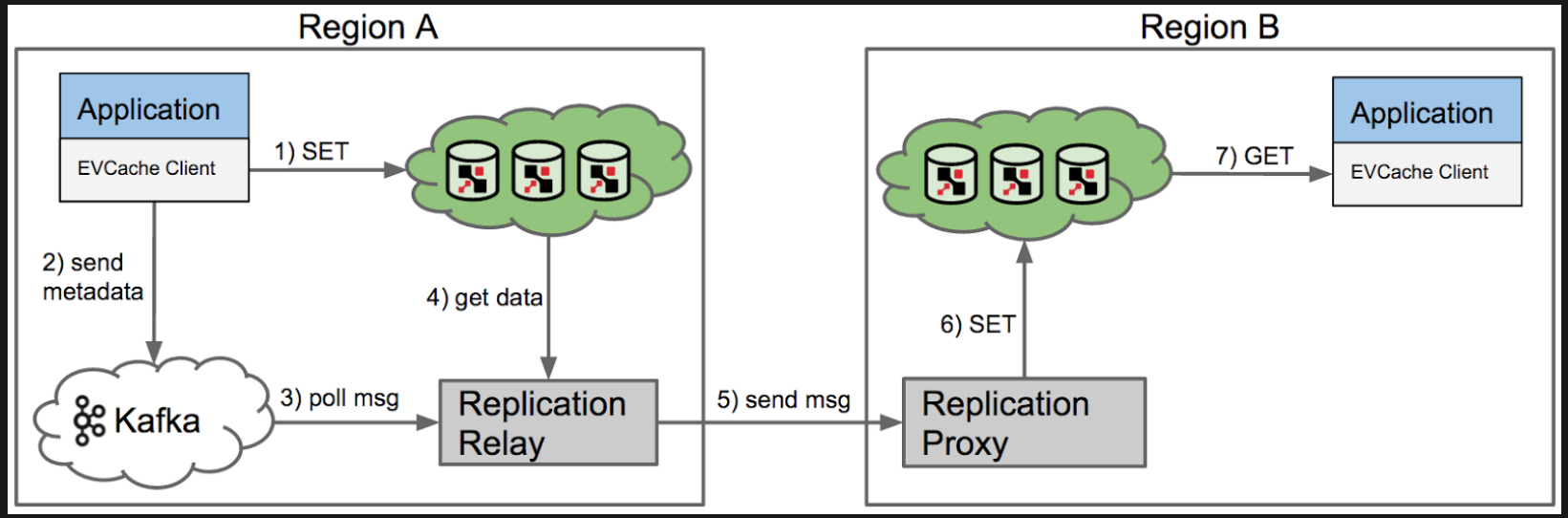
Zuul & Isthmus from Netflix





Request rerouting for users coming from unwanted region, handling load balancer failures

EVCache from Netflix



For remote cache invalidation and synchronization

Conclusions

Conclusions

- Avoid synchronous replication & simultaneous deployments as much as possible
- Design applications for idempotency & eventual consistency as much as possible
- Closely monitor replication & code sync delays
- Have push buttons ready to switch traffic for tenants
- Make High-Level Metrics monitoring systems also Multi-Region

Conclusions

- It is an involved exercise. It requires careful planning and design.
- However, various AWS services make implementation much easier by doing undifferentiated heavy lifting for our customers.

Conclusions

- For companies with extremely high availability requirements or/and geographically distributed user base, benefits of Multi-Region Active-Active architecture can be profound.

- In such cases, consider designing applications for Multi-Region Active-Active implementation from DAY ONE.

But, again as we say in Amazon, today is DAY ONE!

AWS re:Invent

THANK YOU!