
aws
SUMMIT
ATLANTA | MAY 18–19, 2022

ARC301

Best practices for creating multi-Region architectures on AWS

Joe Chapman (he/him)
Sr. Solutions Architect
AWS

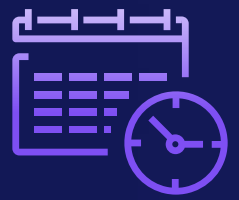
Agenda

- Learn about requirements
- Reasons for multi-Region
- Design considerations
- Design principles
- Takeaways

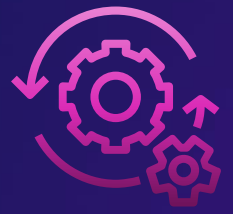
Organizations around the world depend on architectural best practices to help them build workloads and applications that are:

Innovative

Resilient



Rapidly deployable



Optimized for performance, cost, and sustainability

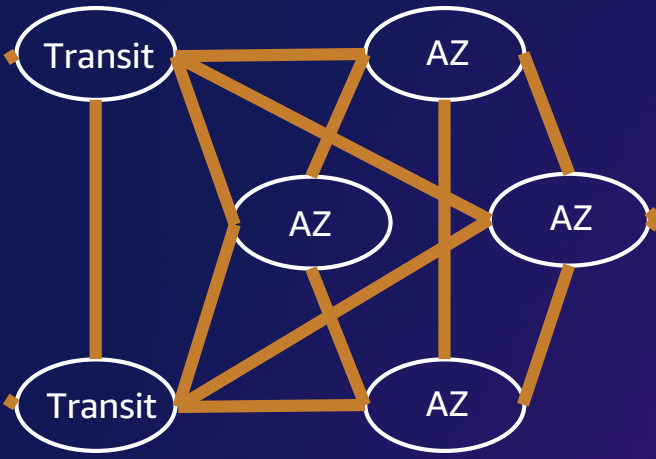
“We need to create a multi-Region architecture.”

Maybe, but first . . .

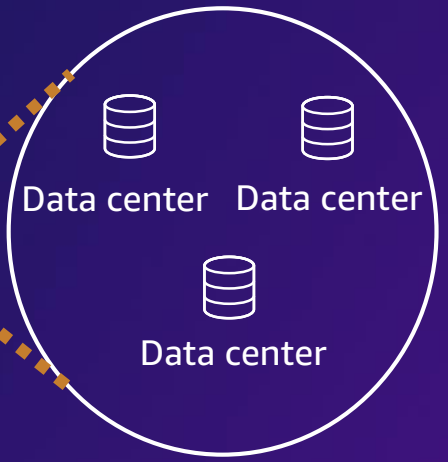
- Take a step back
- Need to understand
  - What initiated this (motivation)
  - Current architecture
  - (Actual) requirements
- Level set on expectations

Remember: AWS Regions are built for resiliency

- Are you Well-Architected?
- Are you taking advantage of Multi-AZ?
- AWS Regional Services



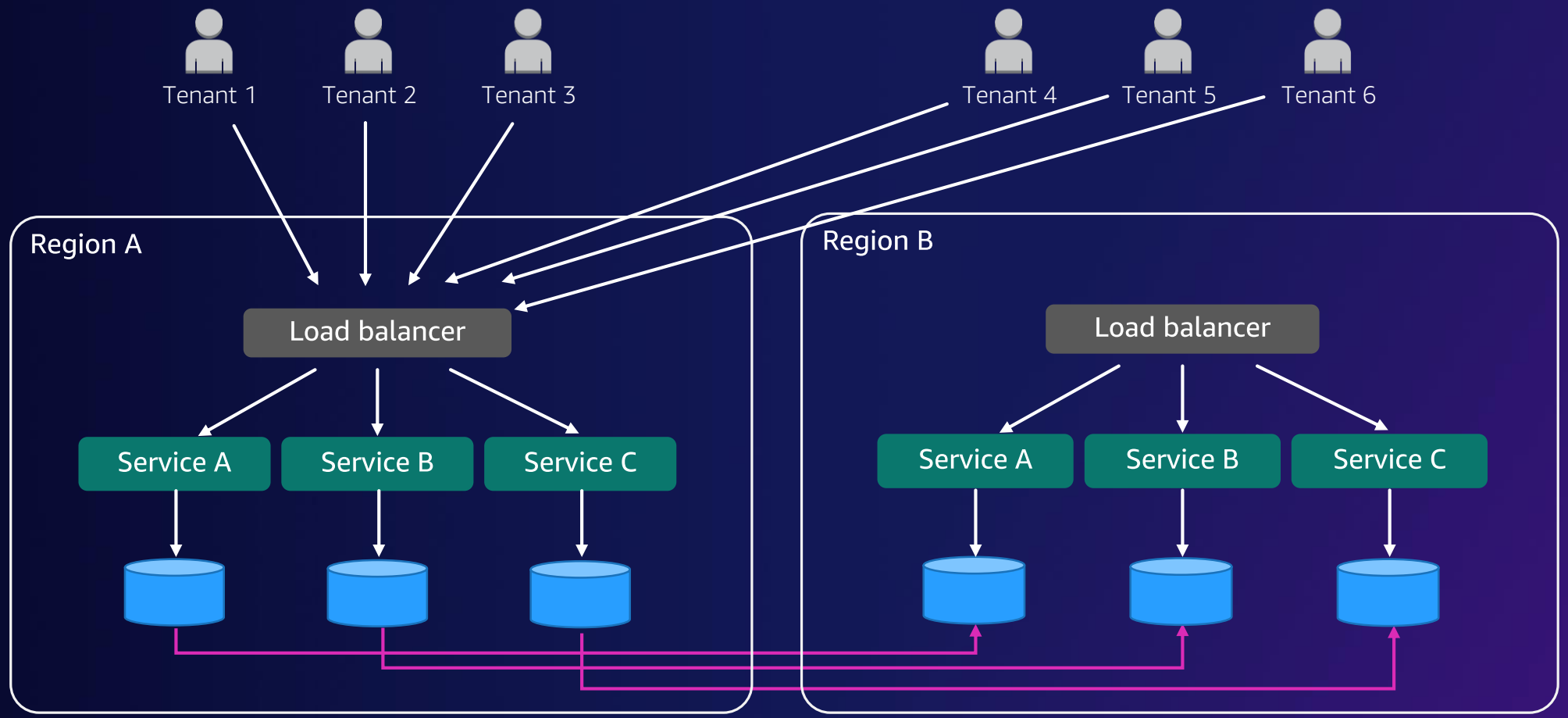
AWS Region



AWS Availability Zone (AZ)

Reasons for multi-Region

1. Disaster recovery



→ Unidirectional data replication between Regions

Strategies for disaster recovery

1.2 Extreme resiliency

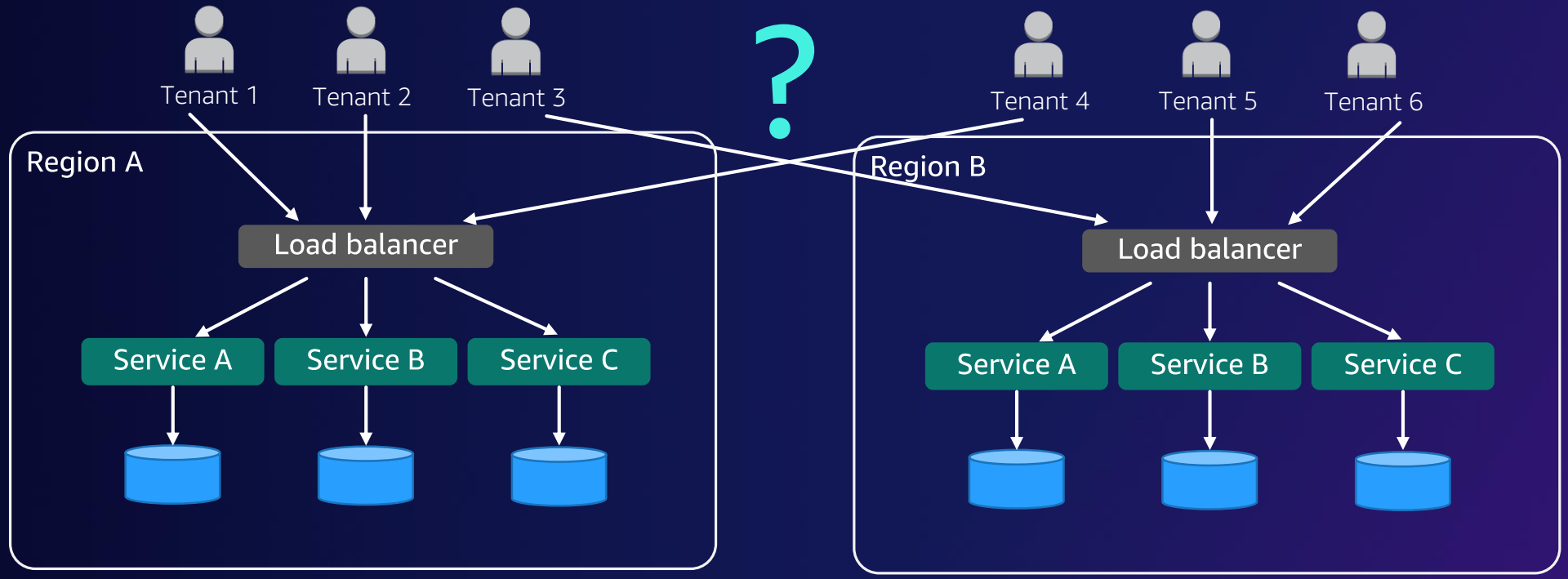
Tier 0 applications

- Financial and stock trading
- Mission-critical systems
- Ecommerce order management

RPO/RTO:
Real time

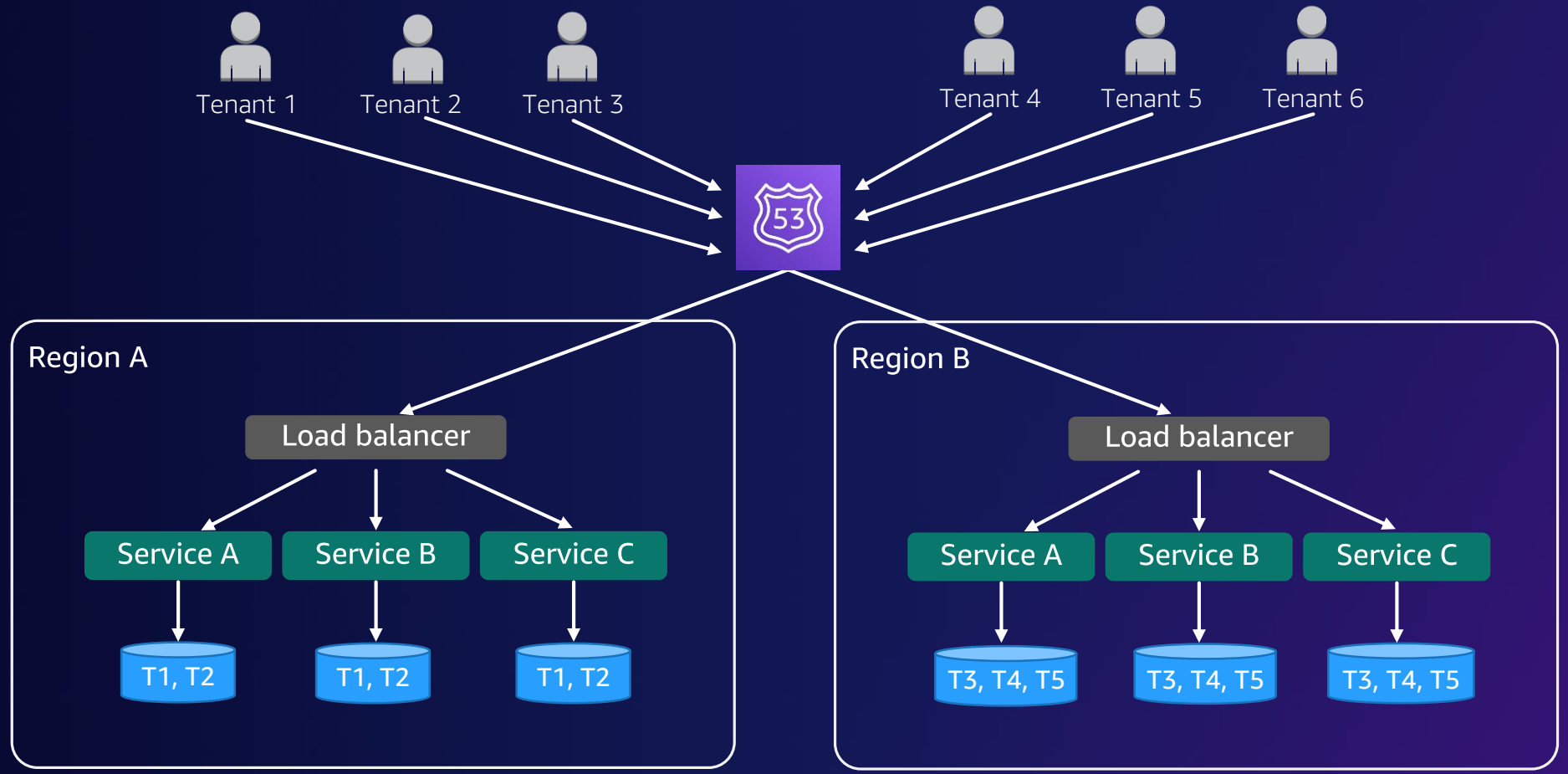
- Zero downtime
- Near-zero data loss
- Mission-critical services
- Cost $$$$

2. Data residency requirements



No data replication between Regions

Data residency requirements, isolated mirror deployment



Data residency – Options and helpers

AWS Regions

26 Regions,
+8 more announced,
310+ PoPs

AWS Local Zones

26 new countries announced

AWS Outposts

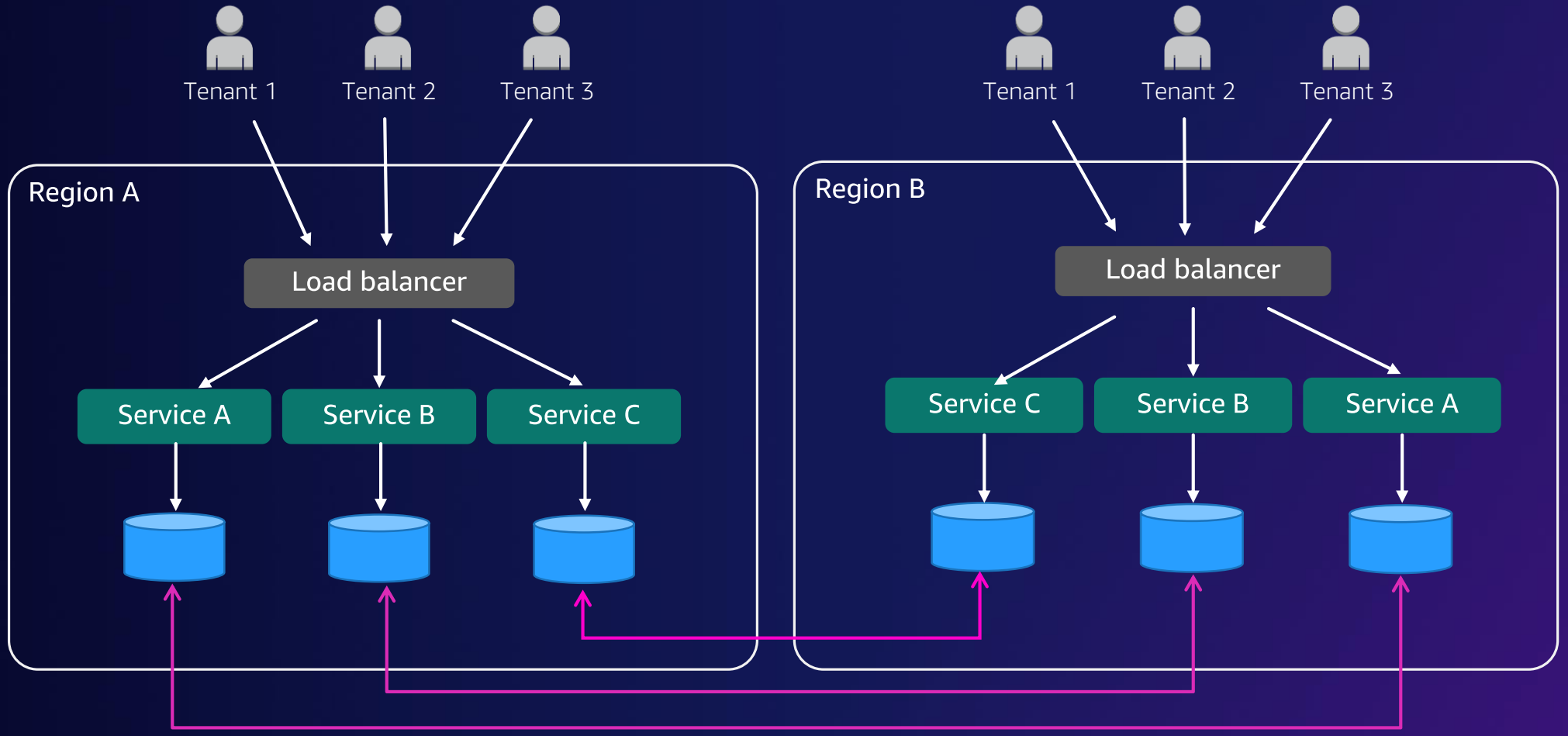
Available in 50+ countries (and growing)

AWS Control Tower

Guardrails to enhance data residency protection

AWS Data Residency whitepaper

3. Low-latency user experience



←→ Bidirectional data replication between Regions

Primary drivers for multi-Region

Increased availability

Regulatory requirements

Improved performance

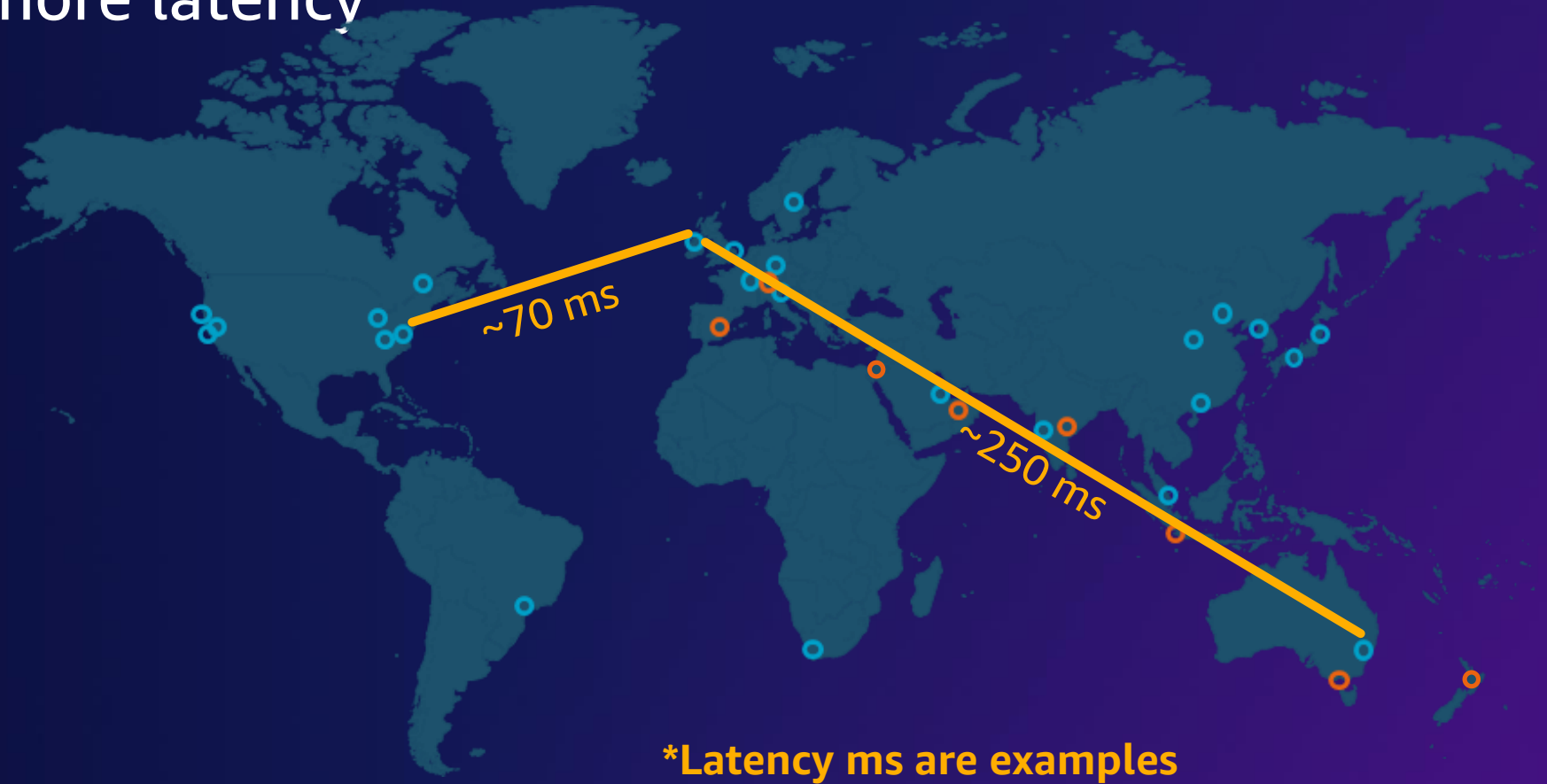
Multi-Region considerations

Speed of light

- 186,000 miles/sec

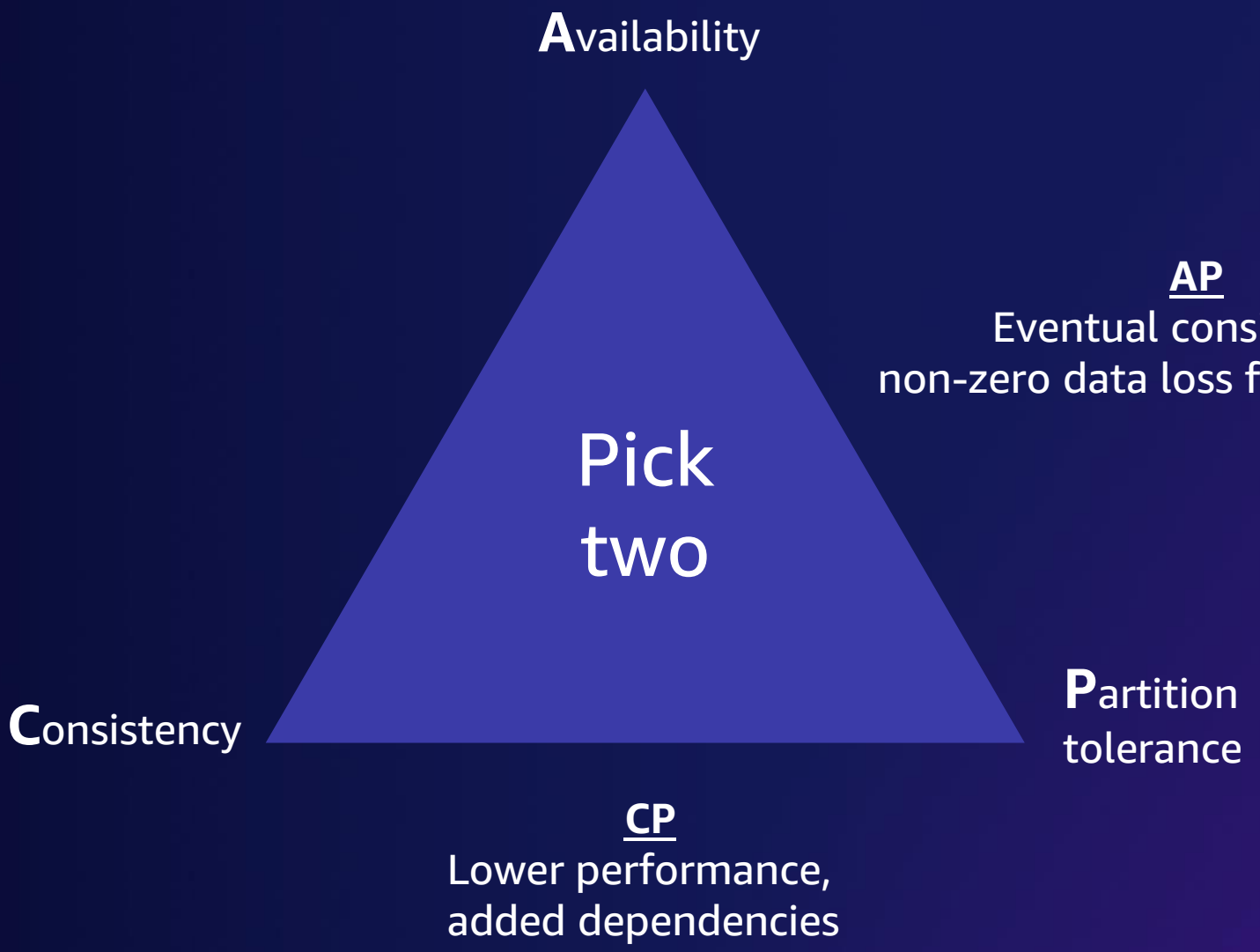
Speed of light

- 186,000 miles/sec
- Longer distance = more latency



*Latency ms are examples

CAP theorem

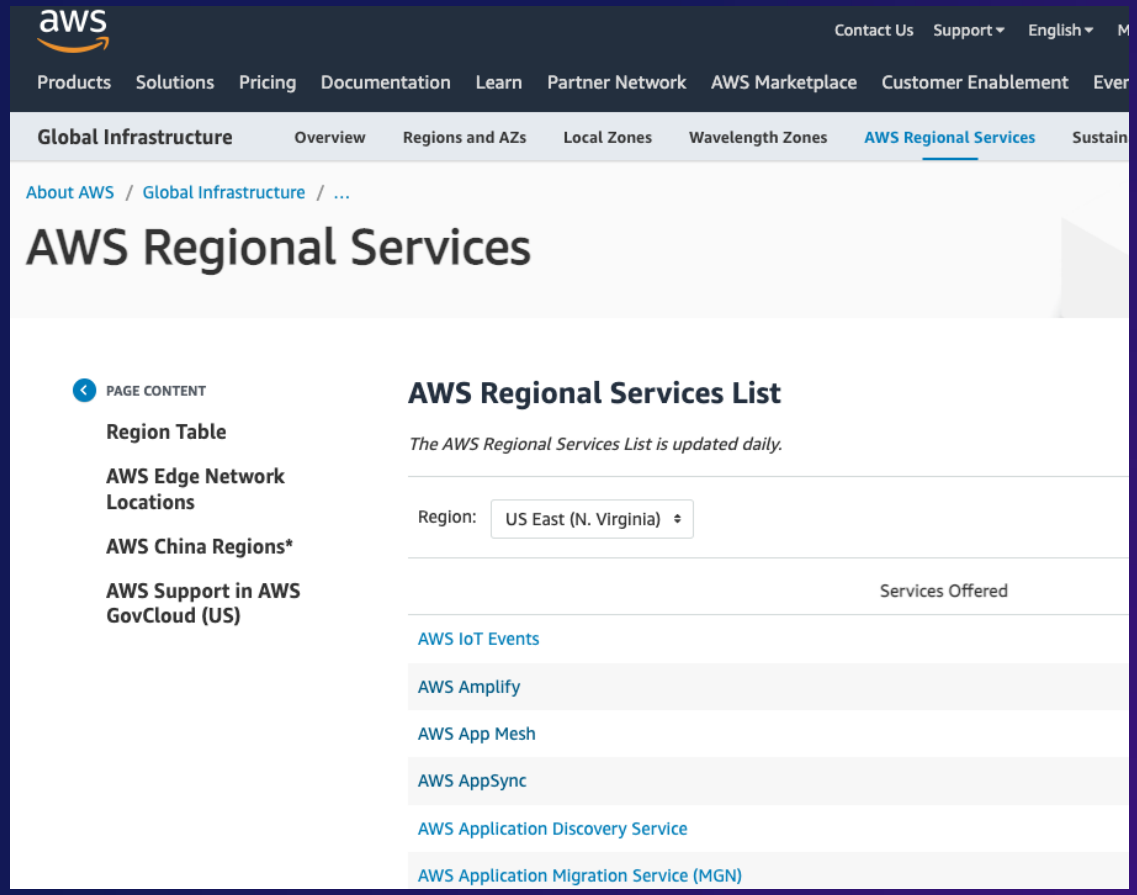


AP
Eventual consistency,
non-zero data loss for in-flight tx

CP
Lower performance,
added dependencies

AWS services in Region

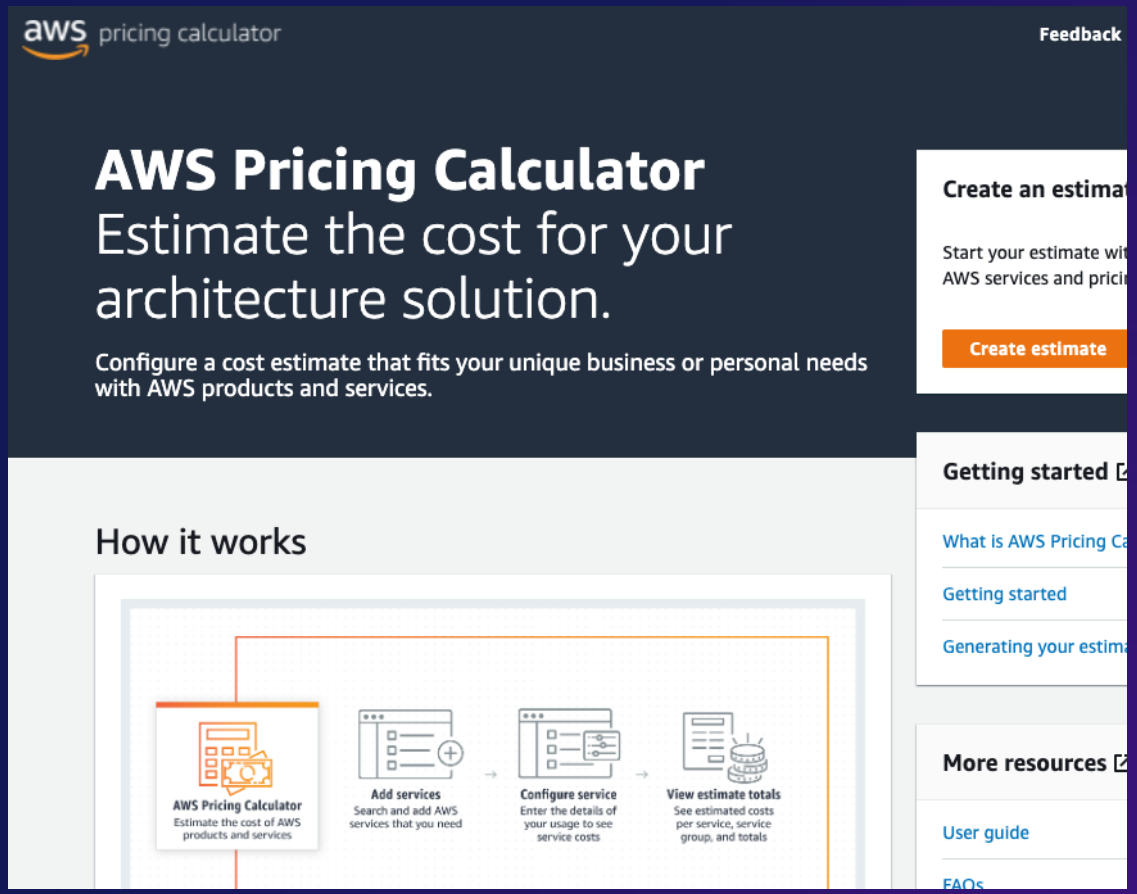
- Check if a service is in desired Regions



AWS Regional Services

Cost

- Regional price differences



AWS Pricing Calculator

Cost

- Regional price differences
- Data transfer costs

AWS Architecture Blog

Overview of Data Transfer Costs for Common Architectures

by Birender Pal, Sebastian Gorczynski, and Dennis Schmidt | on 30 JUN 2021 | in Amazon EC2, Amazon RDS, Architecture, AWS Direct Connect, AWS Site-To-Site VPN, AWS Transit Gateway | Permalink | Share

Data transfer charges are often overlooked while architecting a solution in AWS. Considering data transfer charges while making architectural decisions can help save costs. This blog post will help identify potential data transfer charges you may encounter while operating your workload on AWS. Service charges are out of scope for this blog, but should be carefully considered when designing any architecture.

Data transfer between AWS and internet

There is no charge for inbound data transfer across all services in all Regions. Data transfer from AWS to the internet is charged per service, with rates specific to the originating Region. Refer to the pricing pages for each service—for example, the pricing page for Amazon Elastic Compute Cloud (Amazon EC2)—for more details.

Data transfer within AWS

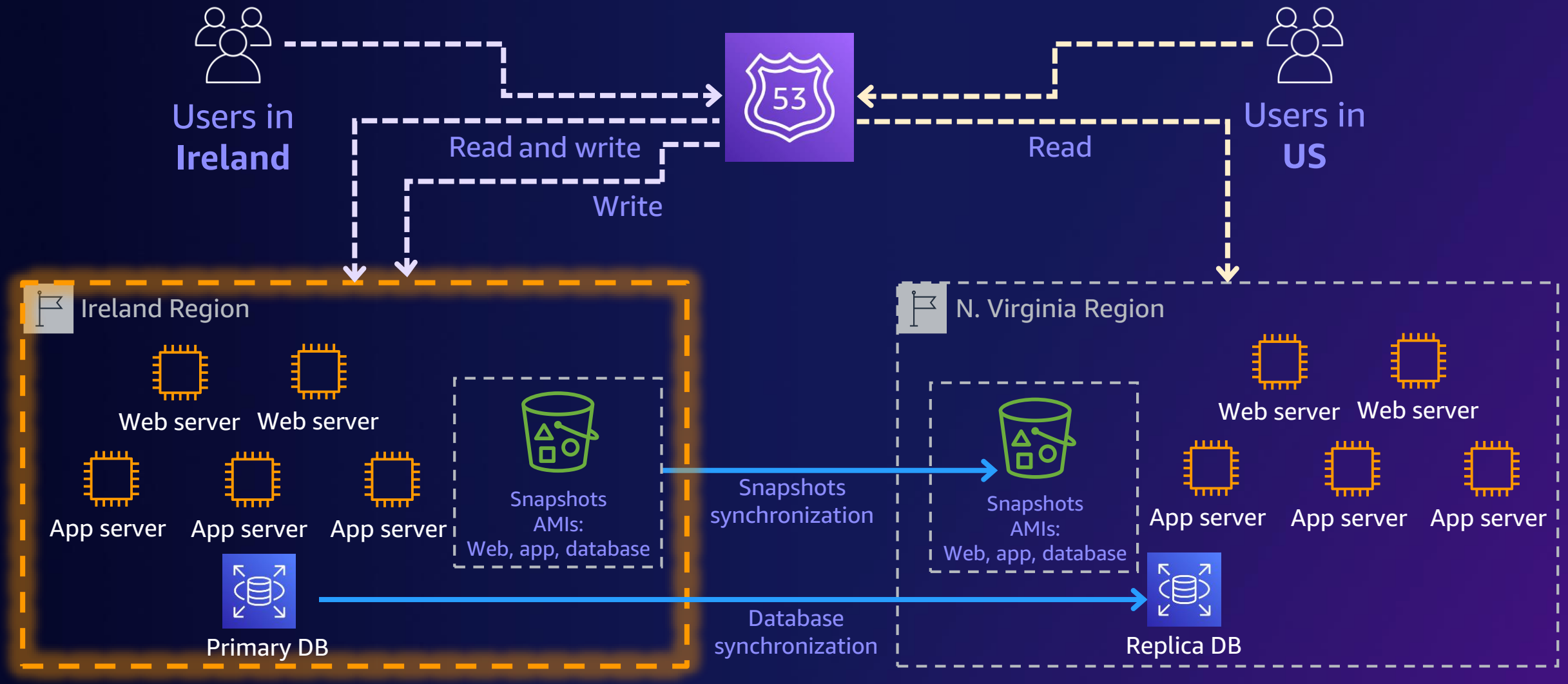
Overview of Data Transfer Costs for Common Architectures

Application tiers

All applications are not created equal

Tiers	Site failover time	Data loss
Tier 1	<10 minutes	None
Tier 2	<2 hours	<5 minutes
Tier 3	<24 hours	<1 hour

Active/active pattern 1: Read local, write global



Active/active pattern 2: Read local, write local

Multi-writer, multi-Region

Multi-Region active/active

99.95%

99.95%

99.999% (5 9s)

Multi-Region design principles

Regional independence

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

Understand and simplify

Understand dependencies

- Between applications
- Third parties
- Recovery mechanisms

Simplify recovery steps

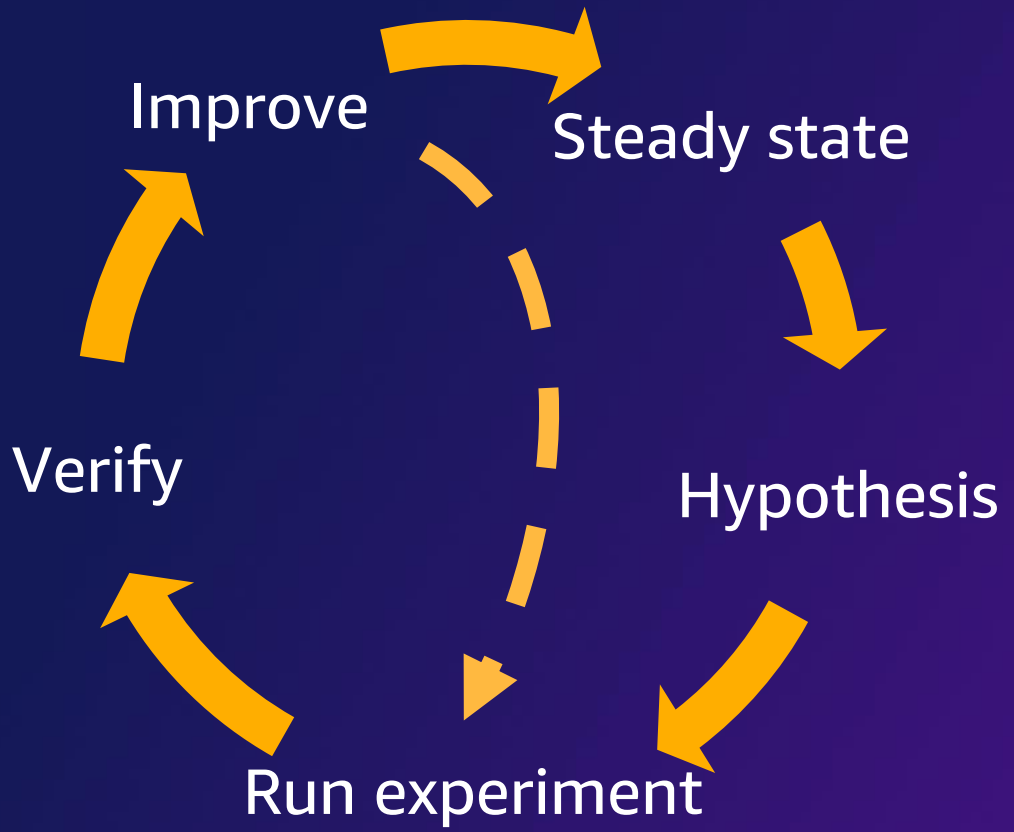
- Over-engineering creates risk
- Recovery steps should be straightforward
- Test regularly

Creating Disaster Recovery Mechanisms Using Amazon Route 53

Test regularly

- For DR, establish decision criteria and testing strategy for recovery
- What happens if Regions can’t talk to each other?
- Create hypothesis and inject failure

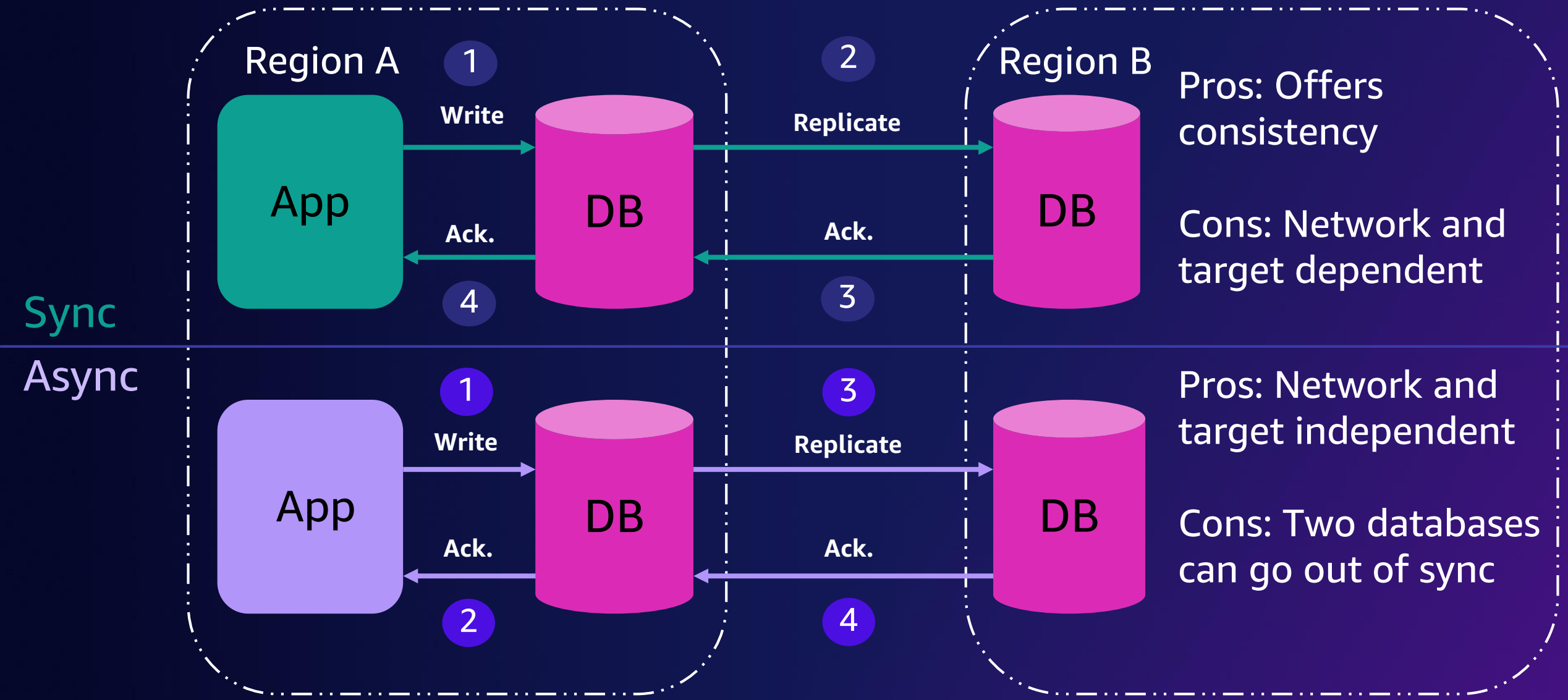
AWS Fault Injection Simulator



Minimize data replication

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

Synchronous vs. asynchronous replication modes



Infrastructure as code

- Reliably and consistently provision and configure infrastructure
- Foundational for DevOps and fast software delivery
- Version controlled
- Rollback changes
- Share and enforce best practices

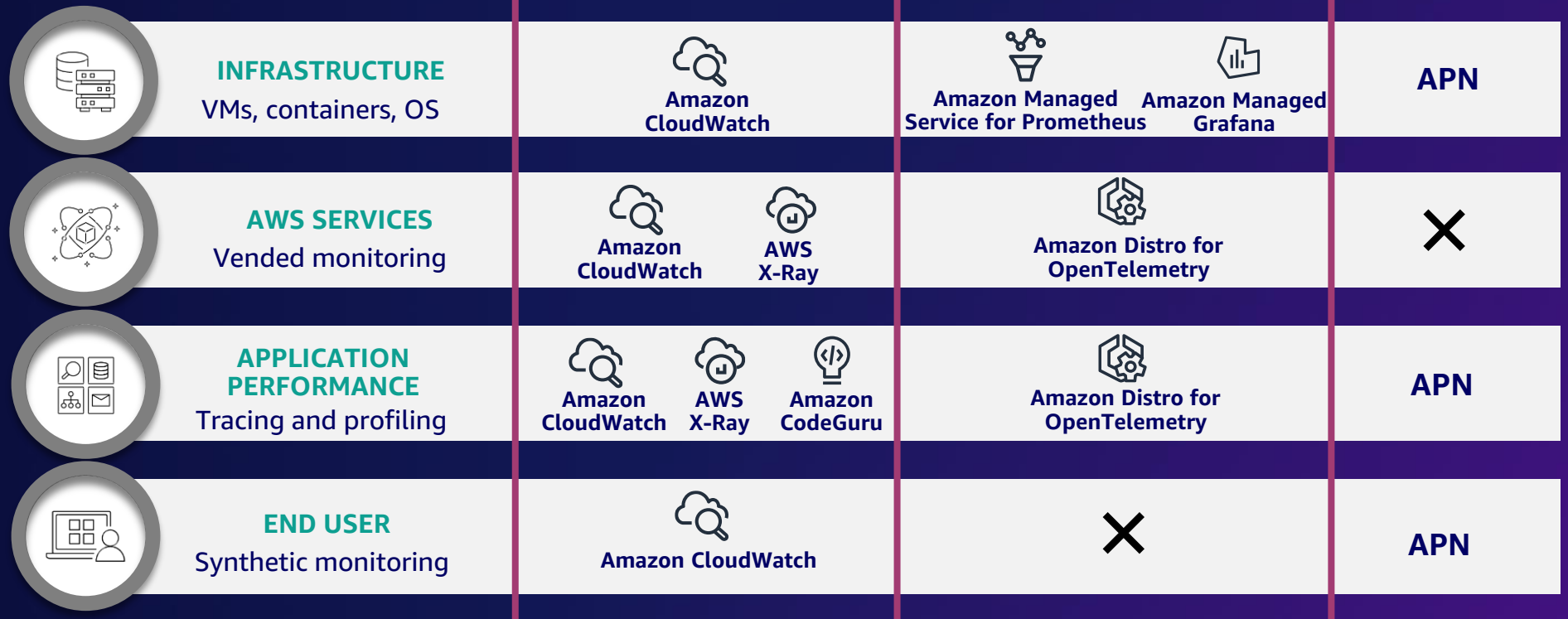
AWS SAM

AWS Cloud Development Kit (AWS CDK)

AWS CloudFormation

Monitoring and observability

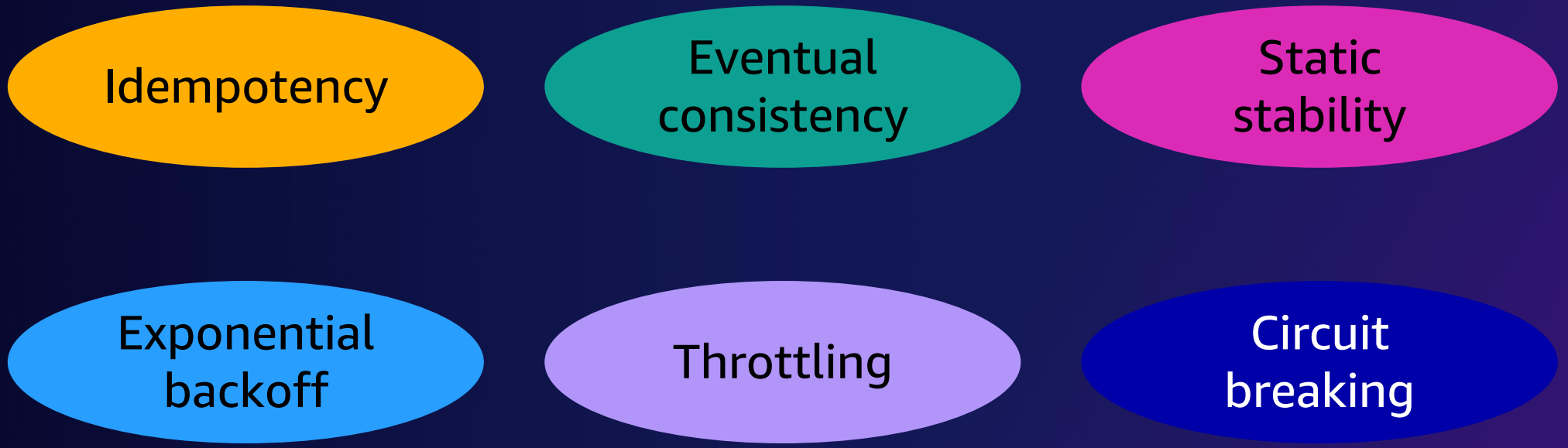
This will get more complicated and become more critical!



Stagger deployments

- Healthy CI/CD practices
- Introduce delays for rolling out changes
- Multi-Region in at least one pre-prod environment
- Observe, track, and even stop deployment if necessary

Distributed system design best practices



Build with AWS services

Creating a Multi-Region Application with AWS Services blog series

Learn in-demand AWS Cloud skills

AWS Skill Builder

Access **500+ free** digital courses and Learning Plans

Explore resources with a variety of skill levels and **16+** languages to meet your learning needs

Deepen your skills with digital learning on demand

Train now

AWS Certifications

Earn an industry-recognized credential

Receive Foundational, Associate, Professional, and Specialty certifications

Join the **AWS Certified community** and get exclusive benefits

Access **new** exam guides

Key takeaways

- Applications built on AWS can handle extreme resilience requirements, but most applications do not require it
- Carefully consider the CAP theorem, as well as overall application design and cost
- Test continuously for known scenarios; experiment for the chaos of unknowns
- Ensure you have healthy IaC, observability, and code deployment practices
- Tightly control your bounded context; understand your dependencies

Thank you!

Joe Chapman

https://www.linkedin.com/in/chapmanjoe/