AWS re:Invent

NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

FSI310

Goldman Sachs : The journey to zero downtime

Robert Cossin
(he/him)
Vice President &
Tech Fellow
Goldman Sachs

Naga Sai Aakarshit Batchu
(he/him)
Vice President
Goldman Sachs

Manjula Nagineni
(she/her)
Senior Solutions Architect
AWS

Today’s discussion

Problem statement

Who we are

Availability requirements and strategies

Key wins and lessons learned

Future enhancements

Problem statement

What factors fuel the desire for zero downtime in banking?

Resiliency

Prepare for unexpected events, such as natural disasters or system failures

Client trust

Clients expect seamless and convenient banking services. Providing uninterrupted services is essential to meeting these expectations

High availability

Provide continuous service availability

Payment SLAs

Ensure we meet our commitments

Who we are

Goldman Sachs (GS)

A leading global financial institution

Transaction Banking (TxB)

Goldman Sachs TxB helps clients build a treasury of the future, while powering tech-forward financial platforms to deliver enhanced offerings. Our mission is simple: provide a global transaction banking platform that is nimble, secure, and easy for clients to use and for partners to connect to.

https://www.goldmansachs.com/what-we-do/transaction-banking/index.html

TxB operational goals and SLA

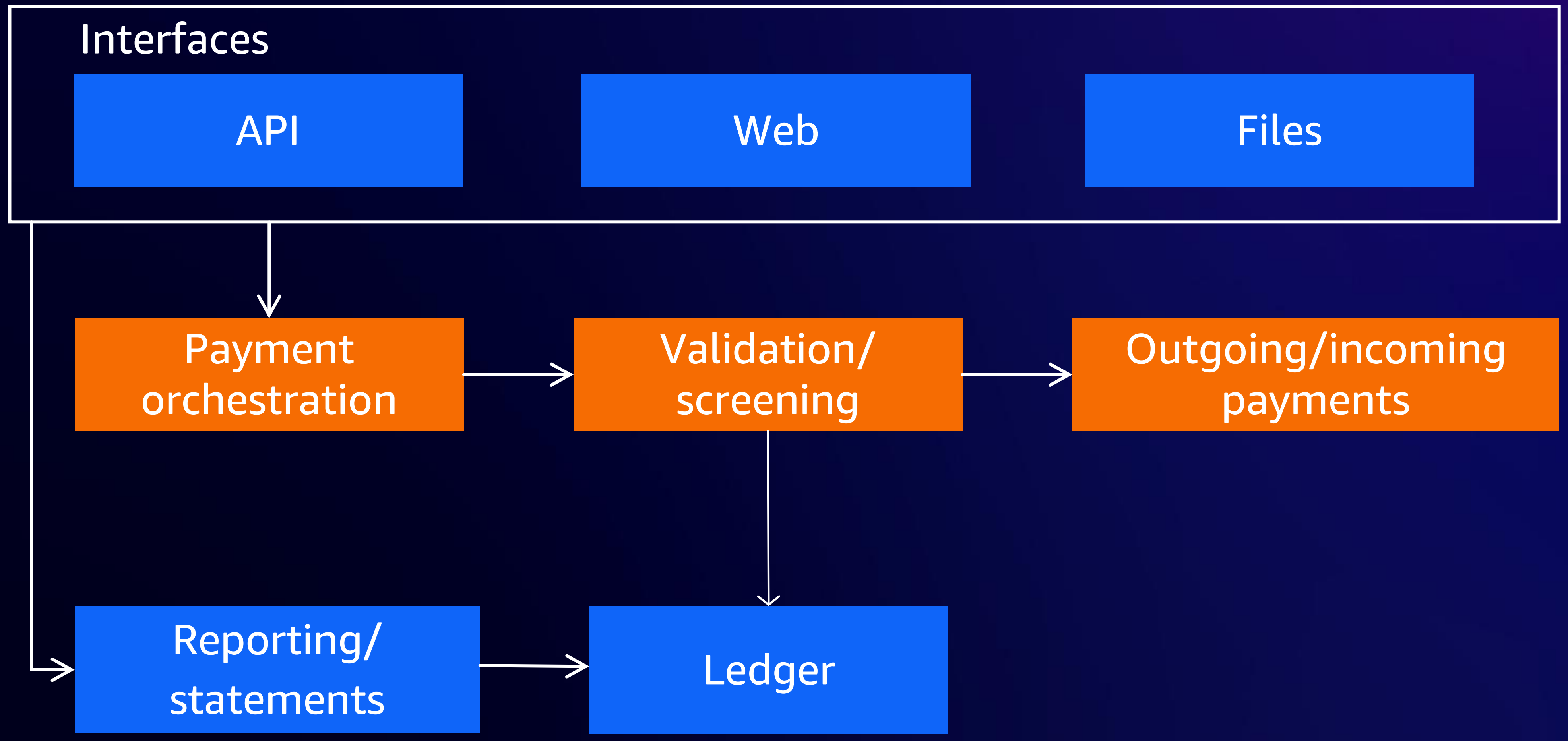
Key application metrics

- Natively built on AWS
- Hundreds of microservices
- ~0 (near zero) RPO
- < 2 mins RTO in Region
- Thousands of deployments annually

Key usage metrics / SLA

- Billions of dollars processed per day
- Strict requirements for payment response times
- Must be highly available
  - RTP requires 99.5% uptime (at most 3.6 hours of unplanned downtime); in the future, may increase to 99.9%
  - Planned downtime is between 2-6 AM ET Sundays, limited to 8 hours total per month

High-level functional bank components



Our approach

→ High-availability architecture

→ Deployment strategies

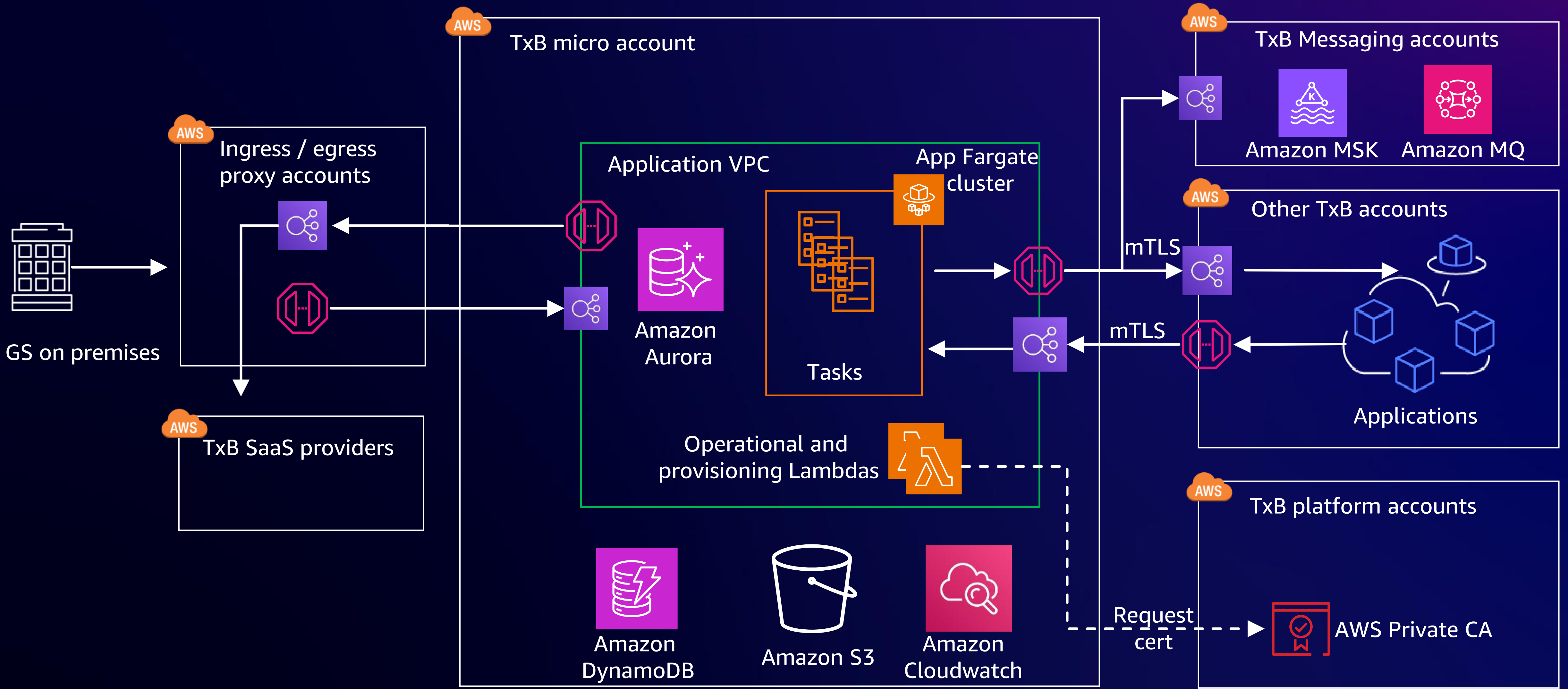
High-availability architecture

Tenets

Services are deployed independently and isolated in their own micro account.
To the extent possible, accounts and services look alike. Uniformity = efficiency.

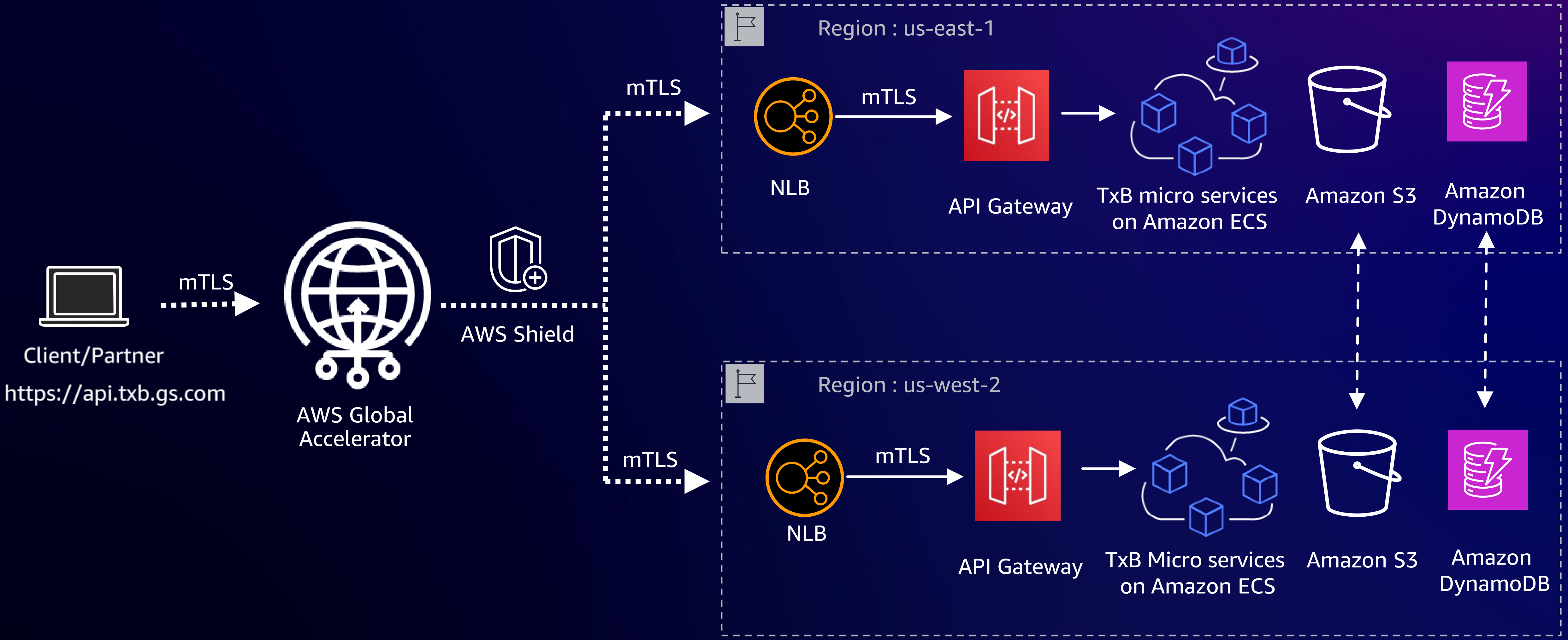
- Amazon ECS / AWS Fargate preferred
- Common IaC modules
- Micro accounts and VPC endpoint services
- Independent zero-downtime deployments
- DevOps cultural philosophy

Nimble microservice blueprint



https://www.youtube.com/watch?v=5cnob8HIswY

Cross-Region blueprint



Deployment strategies

Why zero-downtime deployment?

Developer's dream

Major releases often occur during weekends

Developers typically prefer not to work on weekends

Continuous deployment

Minimizes impact on production caused by faulty deployments

Prevents SLA breaches

Validate

Ability to verify application functions as intended prior to production rollout

Components of zero-downtime deployment

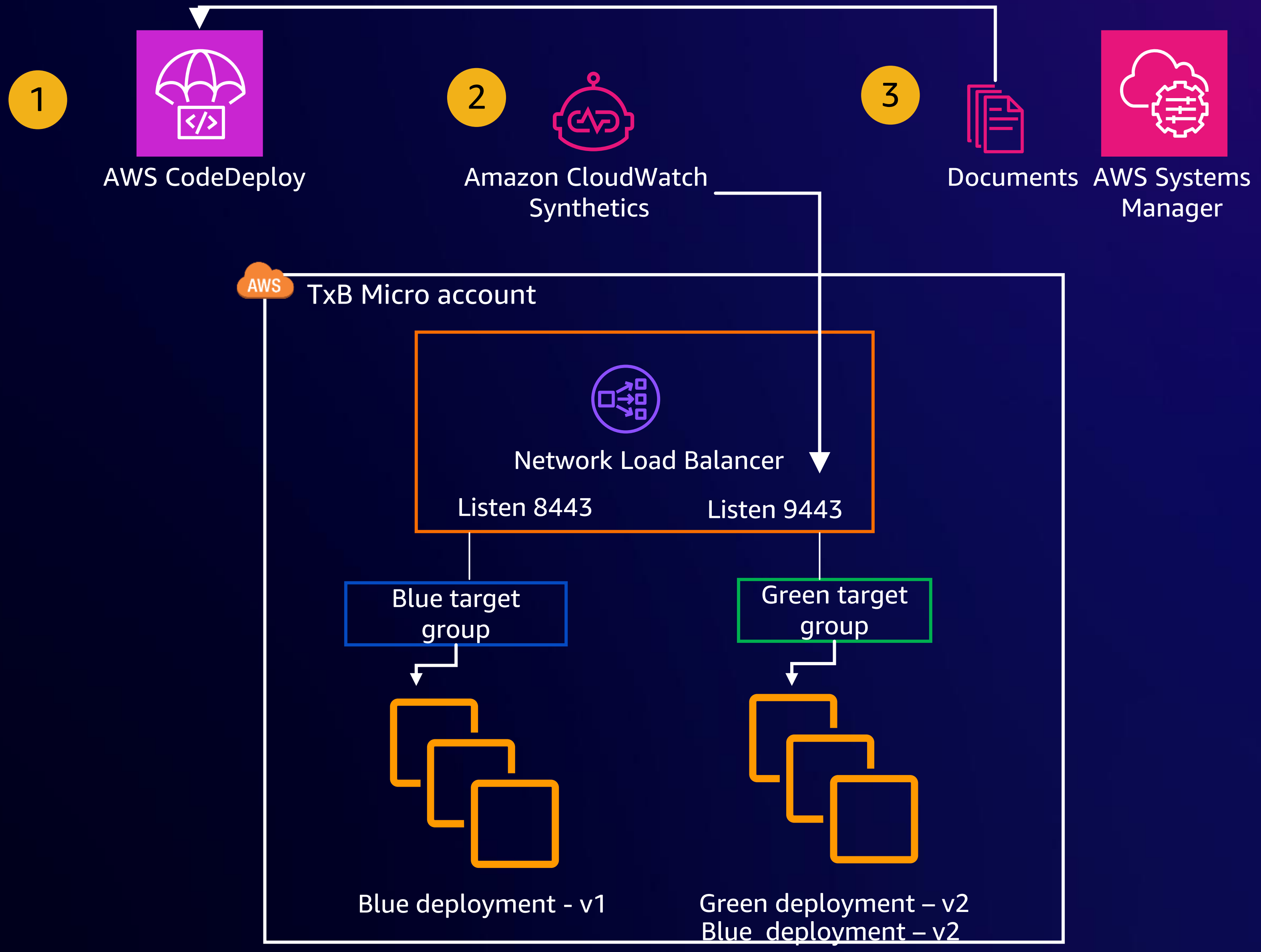
- Stateless services
- Stateful resources
- Release procedure

Deployment strategies

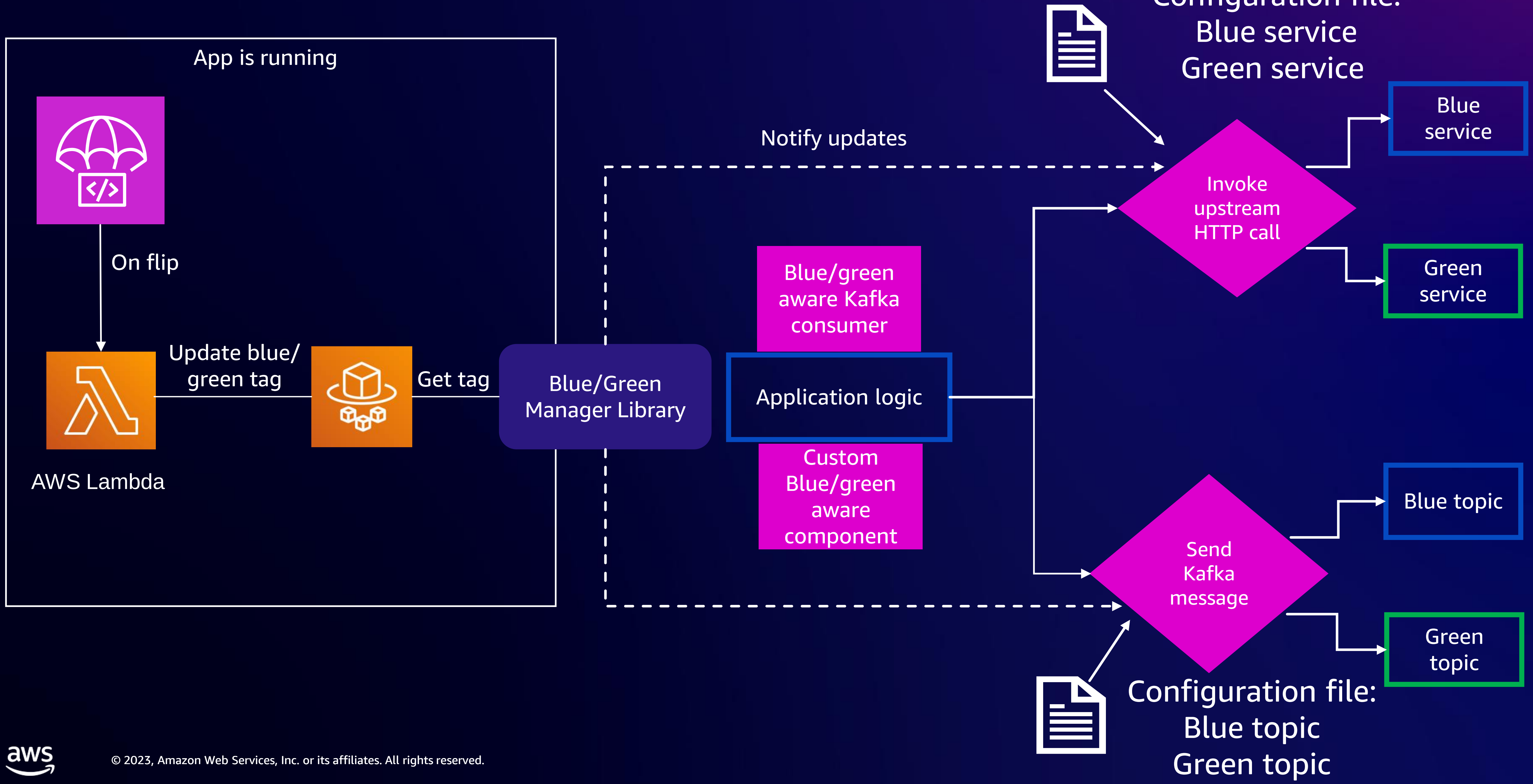
Stateless services on AWS Fargate

Blue/Green	Validate	Safe Flip
AWS CodeDeploy	Automated synthetic validations	Traffic routing and automated flip to serve the live traffic

RESTful Services



Blue/Green Manager Library



Deep health check

- Use ECS/Load Balancer health check to monitor availability of app services
- Implementation based on Spring Boot Actuator
- Easily add in arbitrary functionality using custom health indicators
- Can test business flows and AWS connectivity
- Recommended for rolling restart and blue/green deployments

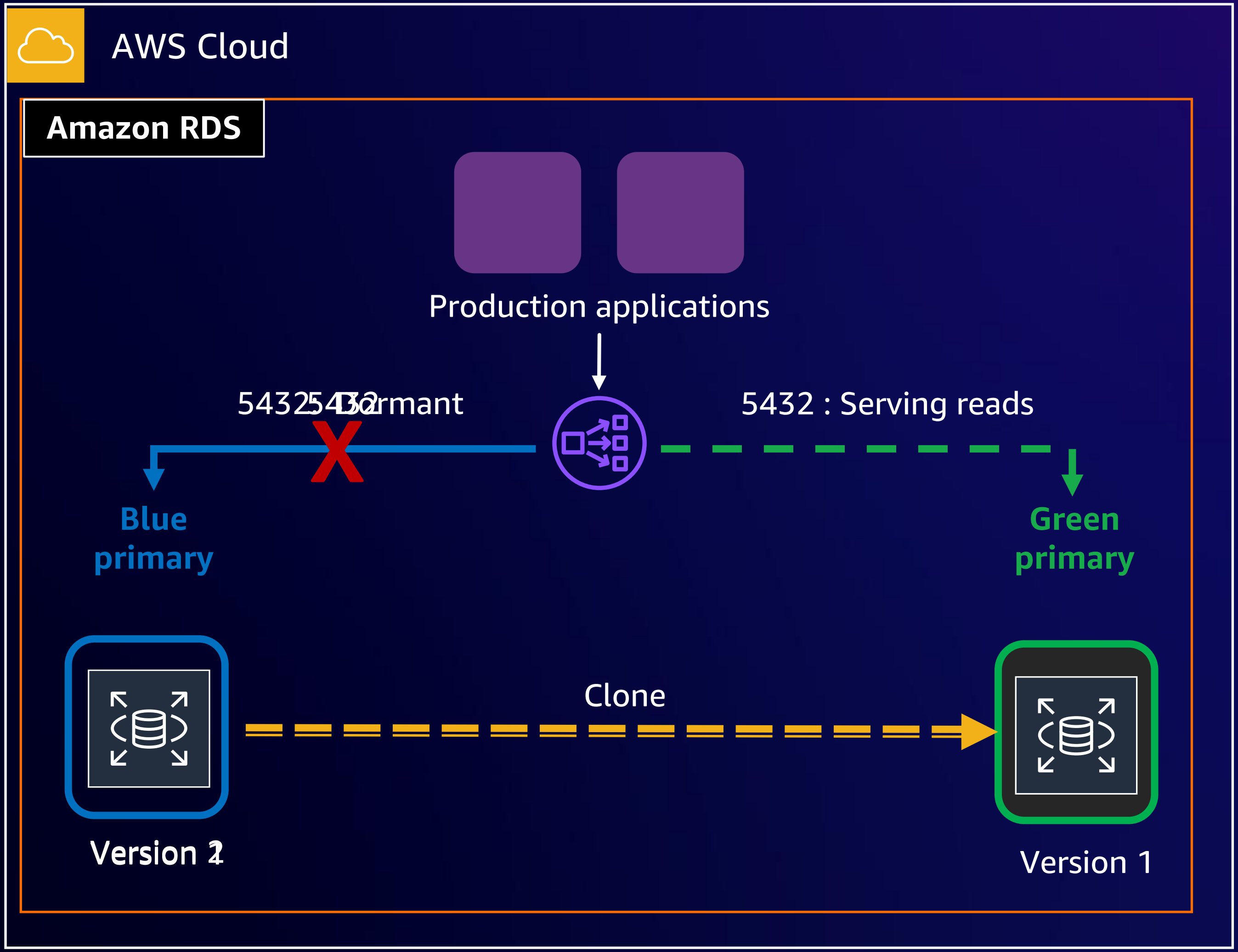
```
health-check:
  controller:
    cache-interval-
seconds: ${CONTROLLER_HEALTHCHECK_SECONDS:120}
    paths-to-
check: /health/mfa,/health/transactions
  dynamodb:
    cache-interval-
seconds: ${DYNAMODB_HEALTHCHECK_SECONDS:120}
    table-name: transactions
```

Deployment strategies

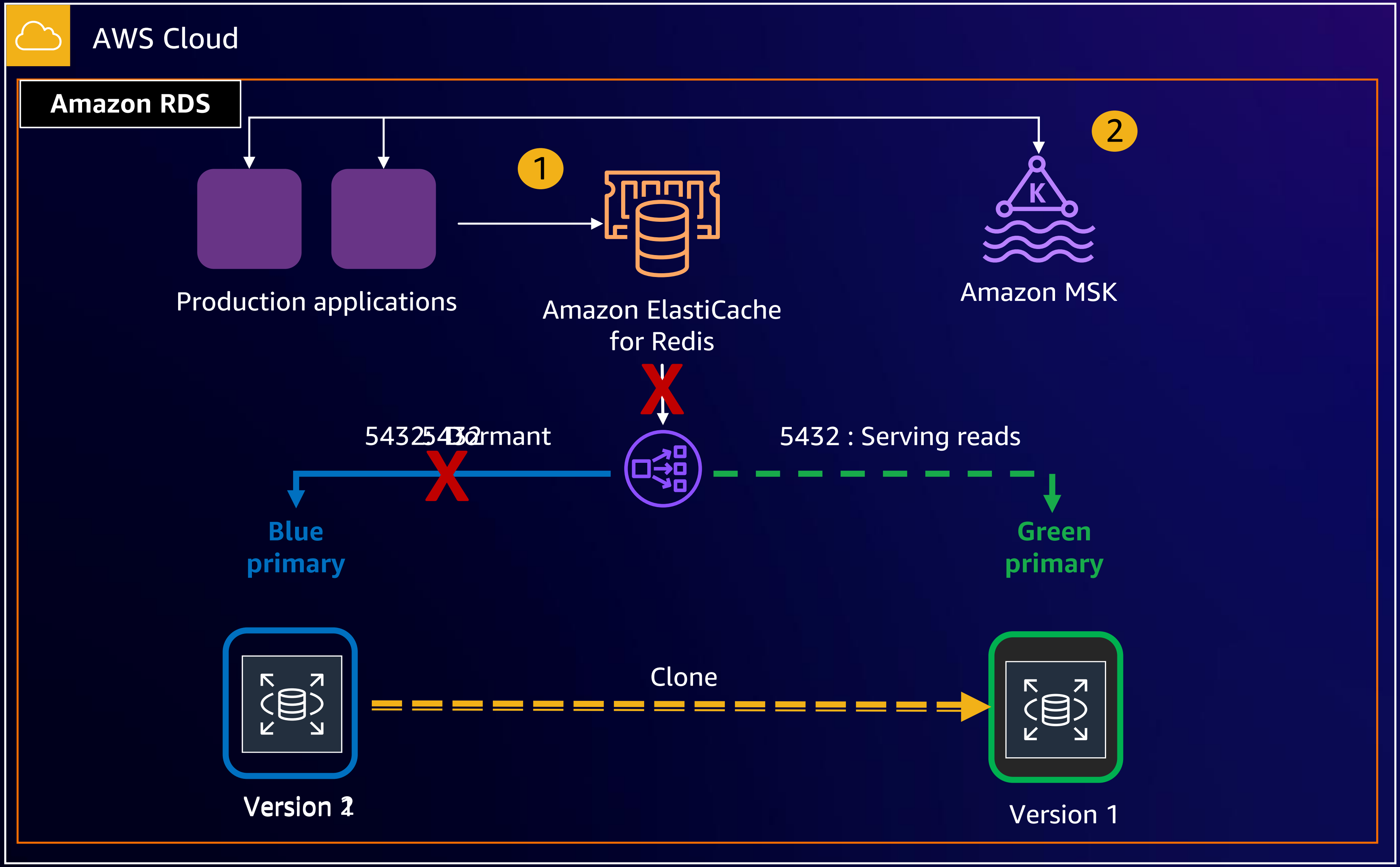
Stateful resources

Amazon RDS	Amazon MSK	DynamoDB
Use Amazon Aurora cloning to create a clone database Temporarily disable the write API	Schema registry to ensure backward and forward compatibility	Avoid schema-incompatible changes

Stateful resources - reads



Stateful resources - writes



Release procedure – staging and testing

Prepare → SaC → Stage → Test

Prepare

- Code reviewed and tagged
- Change approved

SaC

- IaC with defined preventative rules
- Prevent/avoid accidental changes

Stage

- Proceed to staging
- New green environment is up
- Runs up to 48 hours

Test

- Health check, canaries, and manual testing
- Verify health of green environment

Release procedure - flip and verify

Canaries on green

- Run on a schedule
- Health checks to validate

Monitor

- AWS console displays the results
- Green? Ready for flip
- Red? Engineers to address service errors

SSM flip

- Green tasks to blue
- Shutdown blue tasks

Canaries on blue

- Health checks and canaries on blue task

TxB Game Days

Build a well-orchestrated process to failover

Multi-Region deployments
Pre-provision the components in the secondary region

Verify resiliency strategy for Region or service outages

Conduct twice a year

Goldman Sachs Blog: https://bit.ly/gs-rds-resilience

Key wins and lessons learned

Key wins

Projects that use zero-downtime deployment strategies have seen many benefits.

Prod release validation time	0 minutes ⟶	48 hours
Saturday dev team release hours	reduced by	40 hours per month
Aurora downtime per deployment	1 hour ⟶	0 minutes
Release-related incidents	few ⟶	significant reduction

Lessons learned

Do	Don’t
Use deep health checks	Use TCP Ping for NLB health check/PS for ECS
Use ALB for instant update after flip	Assume flip is instant with NLB
Use graceful shutdown and appropriate deregistration delay/stop timeout	Assume all clients are disconnected after flip
Enable code deploy for daemons using stub NLB	Assume processes are down once ECS status changes

Future enhancements

Aurora blue/green deployment

Chaos testing with AWS Fault Injection Simulator

Zero-touch releases and auto-flip the traffic

Thank you!

Please complete the session survey in the mobile app

Robert Cossin

Naga Sai Aakarshit Batchu

Manjula Nagineni