

The background features a dark blue gradient with abstract geometric shapes in lighter blue and orange. A thin orange line forms a triangle on the left side, and a thin blue line forms a rectangle on the right side. The text is centered in the upper right quadrant.

AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

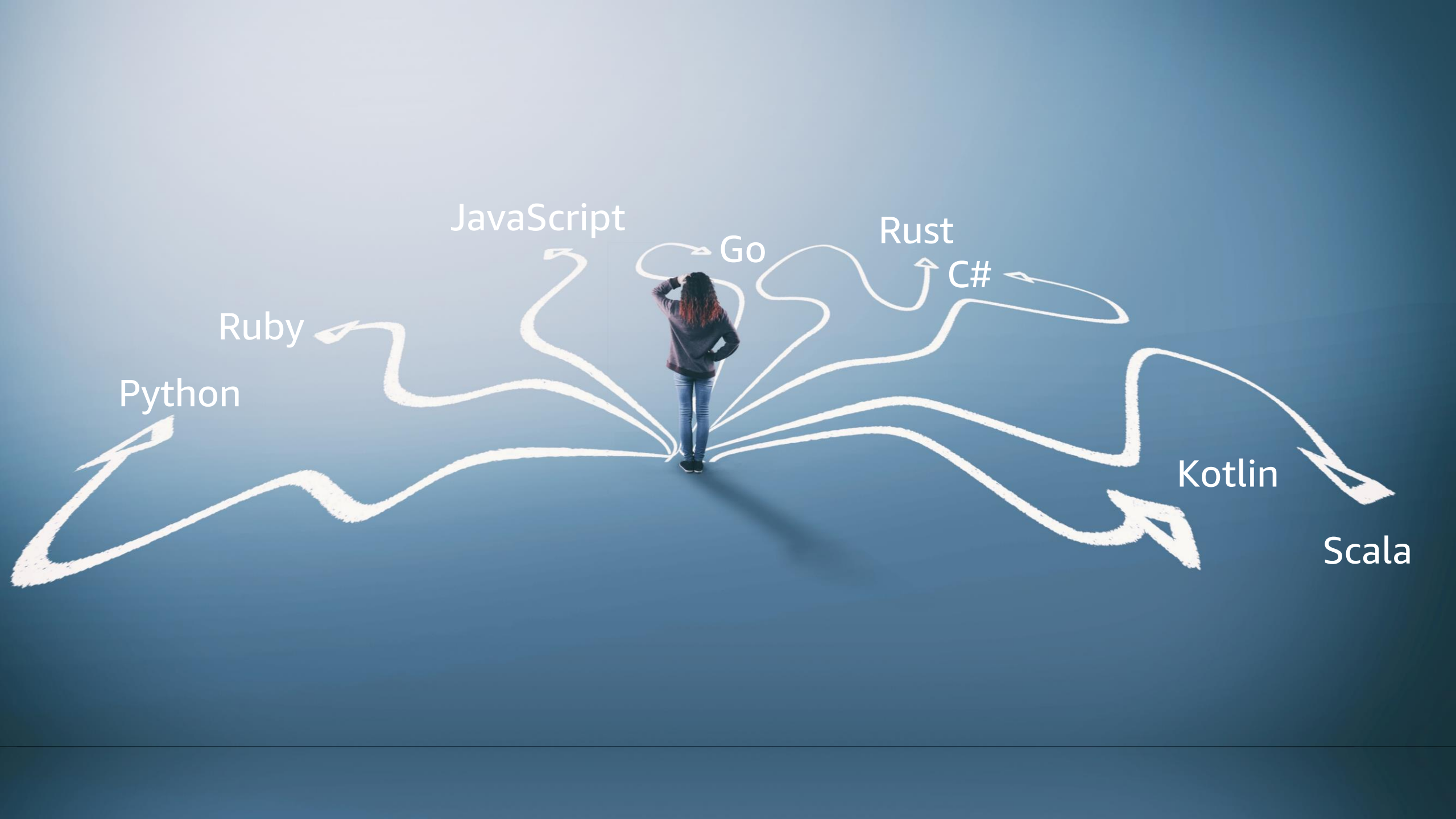
ENT304-R1

Modernize Java enterprise applications with AWS services

Dennis Kieselhorst (he/him)

Sr. Solutions Architect





JavaScript

Go

Rust

C#

Ruby

Kotlin

Python

Scala

“Java is the most popular programming language.”

IDC PaaSView 2021 survey

June 2021



Why migrate and modernize?



Java end of life/ license issues

End of life for certain JRE/JDK and applications server versions

- End of public updates, additional costs for extended support
- Security challenges with outdated versions

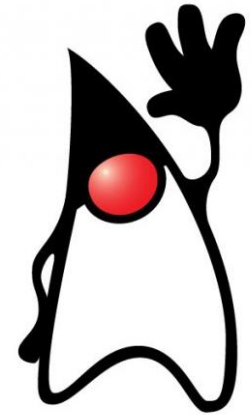
License issues

- Commercial vs. open-source licenses
- License audits/restrictions
(no Cloud usage allowed)



Amazon Corretto

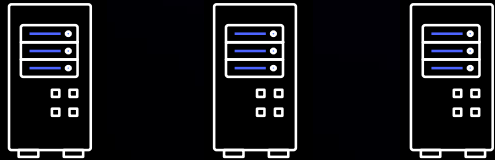
- Amazon supported distribution of OpenJDK
- Run internally on thousands of production services
- Multiplatform: Amazon Linux 2, Windows, macOS, Docker, Ubuntu, etc.
- No cost: distributed under open-source license—no charge for use or distribution
- No cost LTS (patches, performance improvements)
- Corretto 8, 11, 15, 16, and 17
- Open source – all patches and enhancements will be up streamed to OpenJDK: <https://github.com/corretto>



J2EE → JEE → Jakarta EE



Traditional Java enterprise application architecture



Web servers

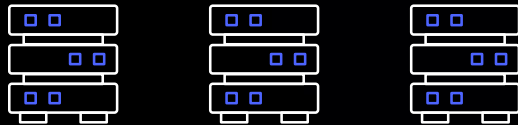
Presentation layers

JavaServer Faces (JSF)

Servlets

JavaServer Pages (JSP)

Applets



Application servers

Business logic

Enterprise JavaBeans (EJB)

Webservices (JAX-RPC, JAX-WS, JAX-RS)

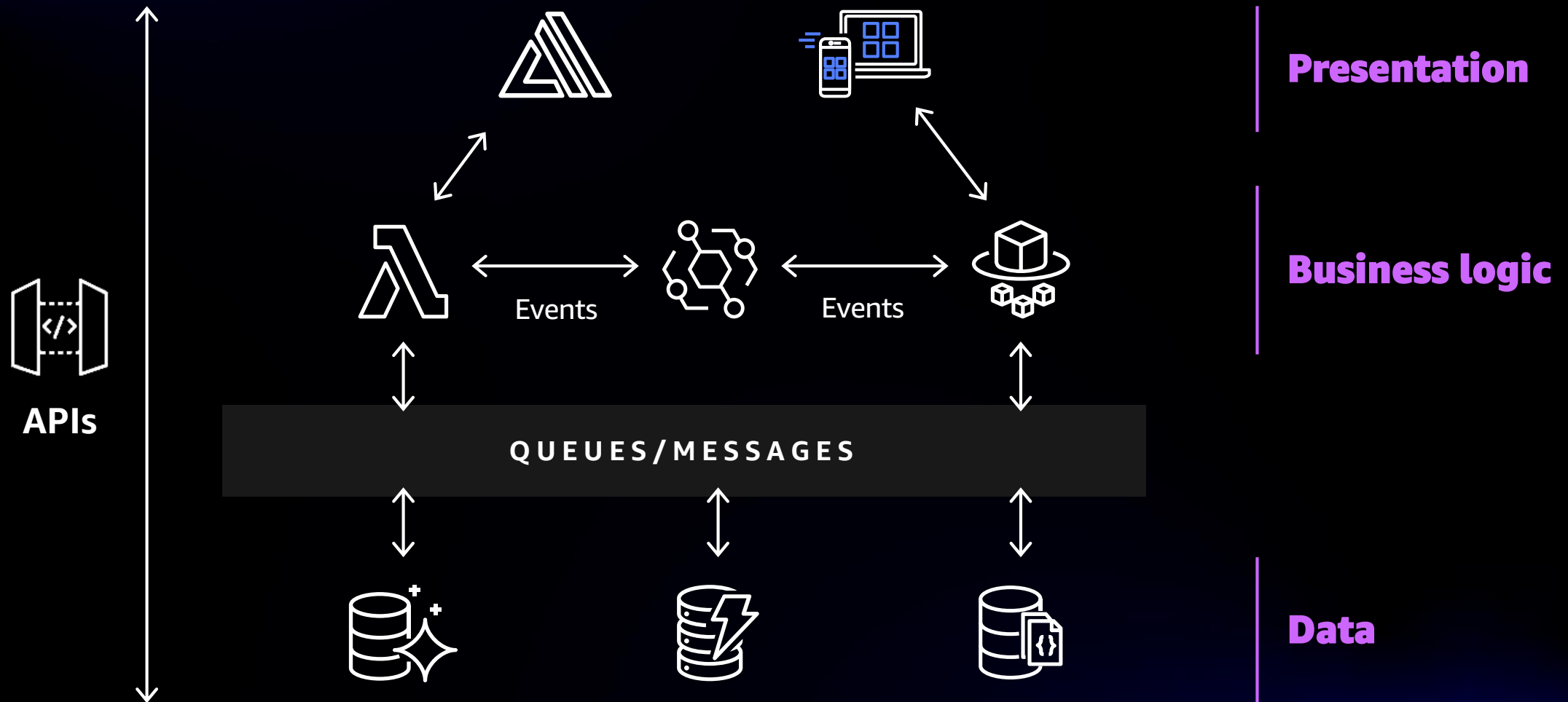
Java Message Service (JMS)



Database servers

Data layer

A modern 3-tier application architecture



Modernization pathways – Overview

Relocate/rehost

MIGRATE TO THE CLOUD

Apps/DBs run on VMs
No code changes



VMware Cloud
on AWS



Amazon EC2

Applications

Re-platform

CONTAINERIZE APPLICATIONS

Develop and deploy faster
Application portability
No code changes



Amazon ECS



AWS Fargate



Amazon EKS

Refactor/rewrite

MOVE TO OPEN-SOURCE

License freedom/savings
Performance improvement
Cross-platform support

MOVE TO SERVERLESS FUNCTIONS

Move from idea to market, faster
Lower costs



AWS Lambda

Databases



Database on Amazon EC2

Customer operates everything
above the infrastructure

MOVE TO MANAGED

Managed provisioning, backups,
patching, monitoring, and scaling
No code changes



Amazon RDS

PURPOSE-BUILT DATABASES

High performance and scalability
Licensing savings



Amazon
Aurora



Amazon
DynamoDB



Amazon
Neptune



Amazon
Redshift

...

Modernization pathways – Application focus

Java enterprise applications

Relocate/rehost

MIGRATE TO THE CLOUD

Apps/DBs run on VMs
No code changes



VMware Cloud on AWS



Amazon EC2

On-prem → Cloud

Re-platform

CONTAINERIZE APPLICATIONS

Develop and deploy faster
Application portability
No code changes



Amazon ECS



AWS Fargate



Amazon EKS

VMs → Containers

Refactor/rewrite

MOVE TO OPEN-SOURCE

License freedom/savings
Performance improvement
Cross-platform support

Examples:

Oracle WebLogic
IBM WebSphere
Pivotal Cloud Foundry



Apache Tomcat
Apache TomEE
WildFly

MOVE TO SERVERLESS FUNCTIONS

Move from idea to market, faster
Lower costs



AWS Lambda

Monolith → Microservices

AWS App2Container



Discover and analyze

Create application inventory and analyze runtime dependencies

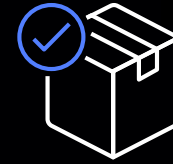
1



Extract and containerize

Extract application with dependencies and create container image

2



Create deployment artifacts

Generate the ECS tasks or Kubernetes pod definitions and create CI/CD pipelines

3

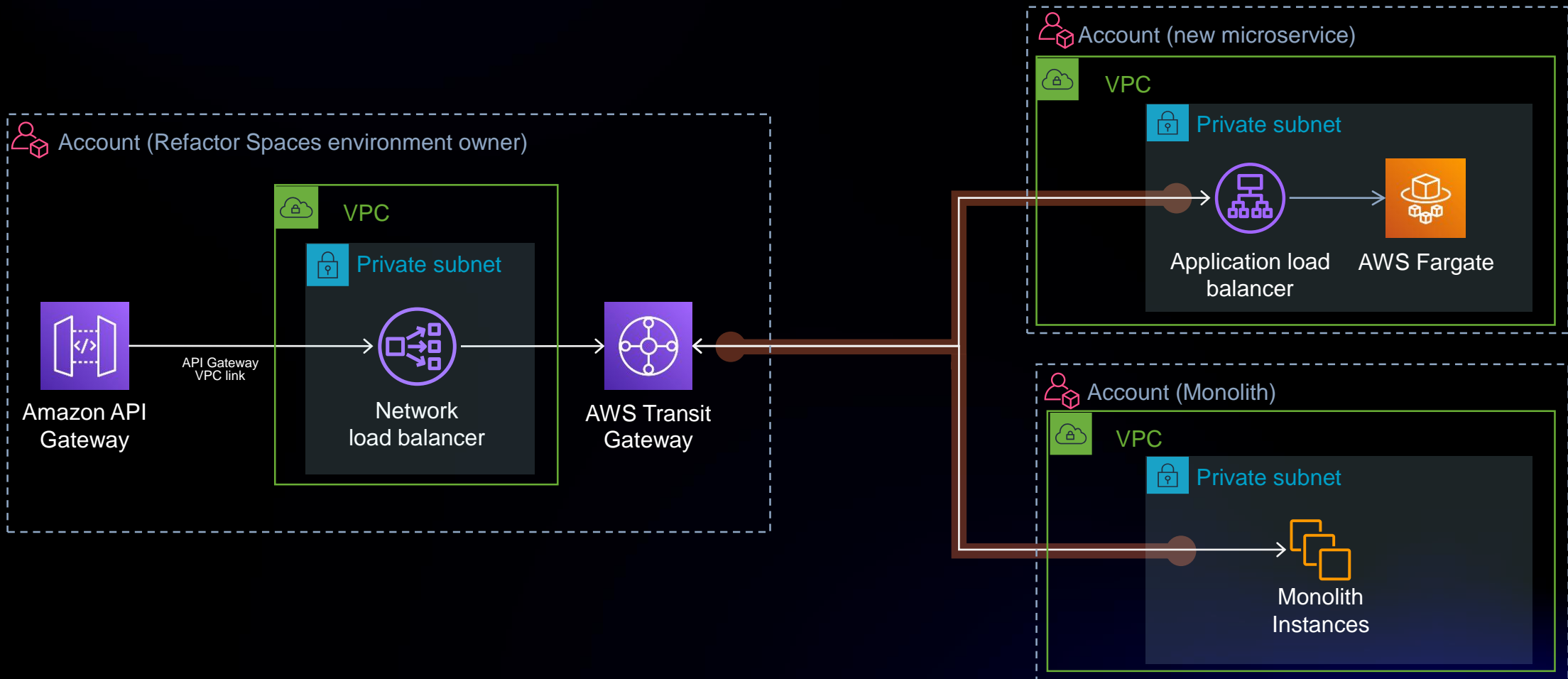
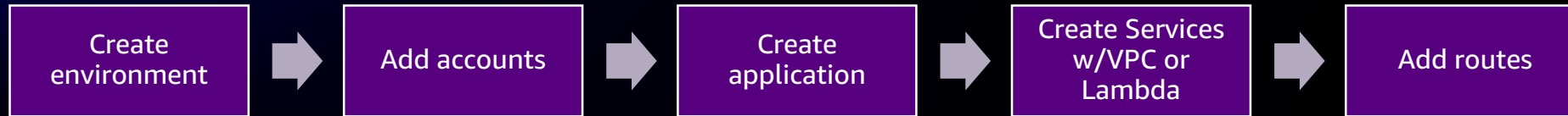


Deploy to AWS and launch

Store image in Amazon ECR and deploy to Amazon ECS or Amazon EKS

4

AWS Migration Hub Refactor Spaces



Modern Java frameworks run well on AWS

Popular ones with specific AWS docs/submodules:

- Quarkus
- Spring Boot/Spring Native
- Micronaut

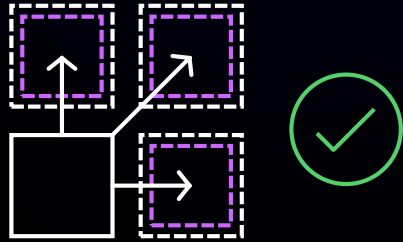
Performance tuning

- Native compilation using GraalVM
- Tiered compilation stop at level 1
- Analyze garbage collection



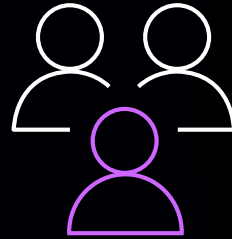
Appendix

Key pillars of modernization



Technology & architecture

Independent business functions



People, process, & culture

Organized for value



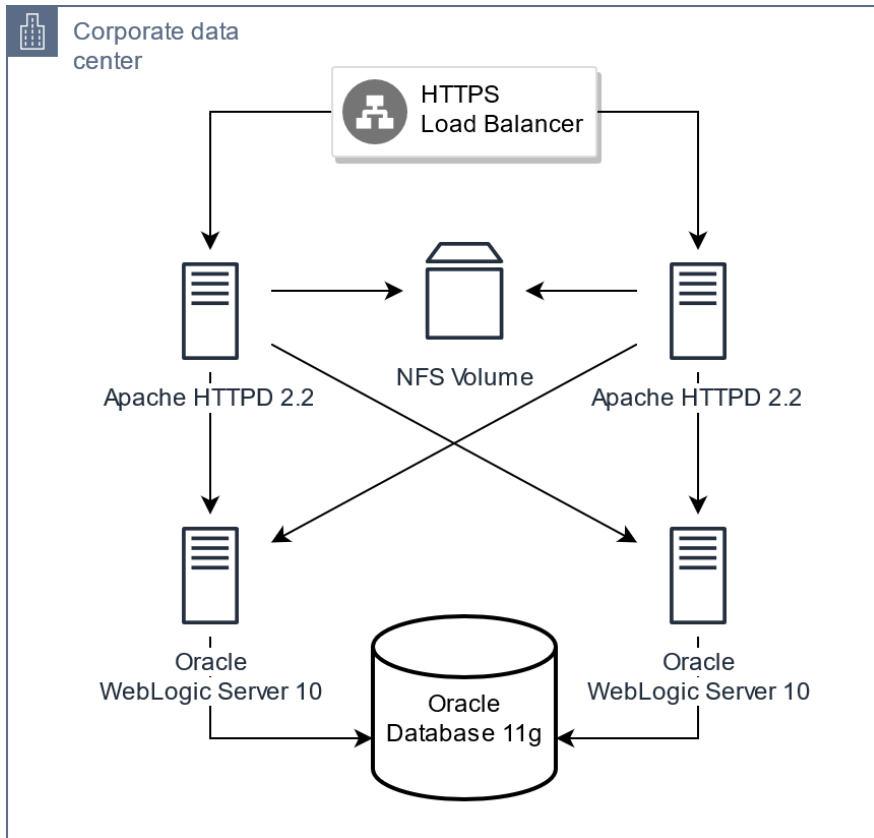
Ops & governance at scale

Automate, enable, & self-service

Modernization is the refactoring of legacy technology by combining modern infrastructure, architecture, and organization patterns together to maximize resiliency, engineering efficiency, and business agility

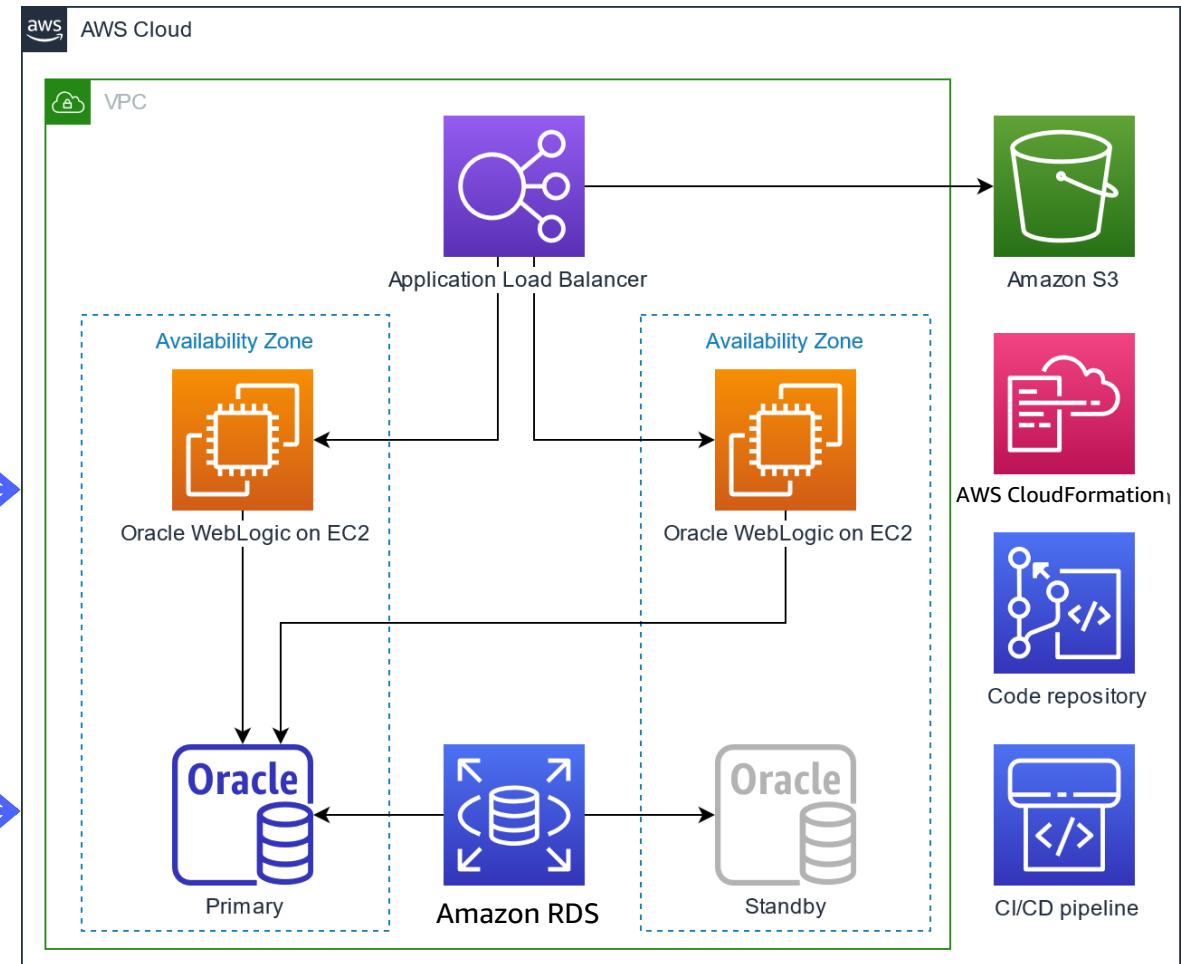
Example for an iterative migration and modernization

Migration with first modernization steps

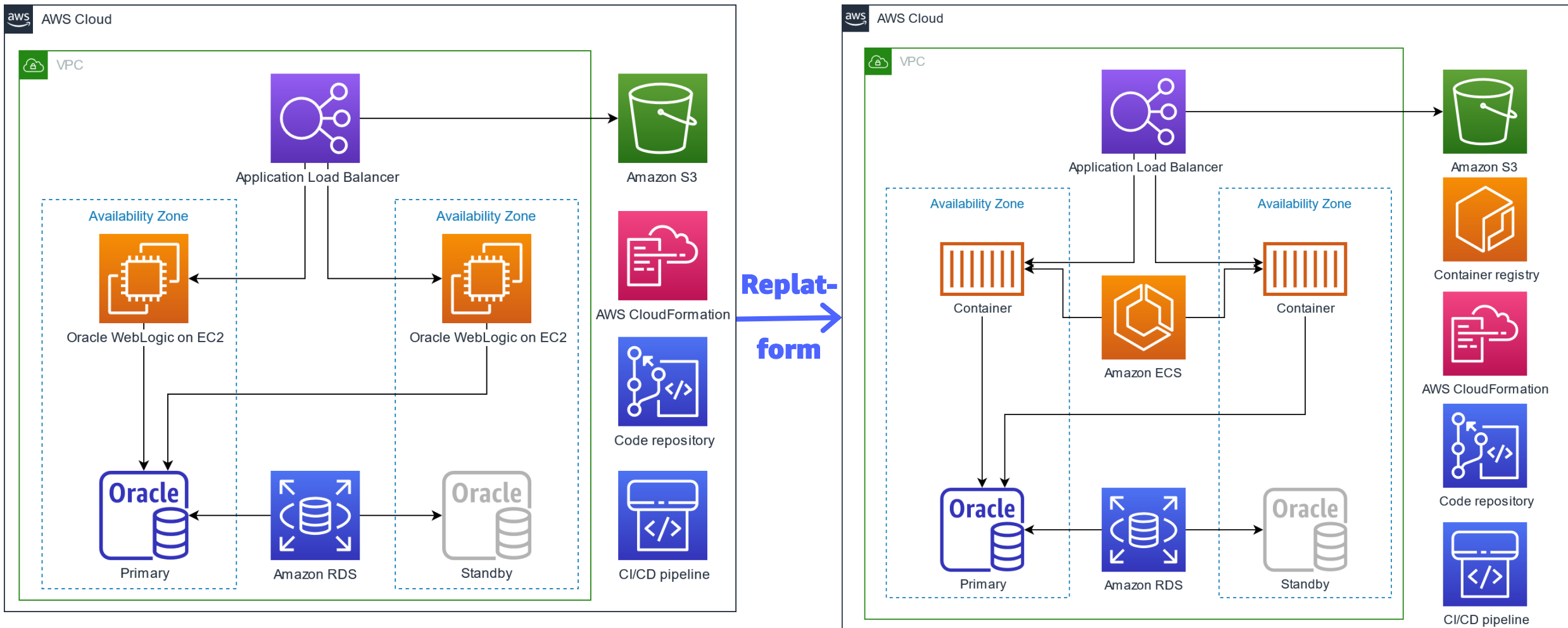


Rehost+

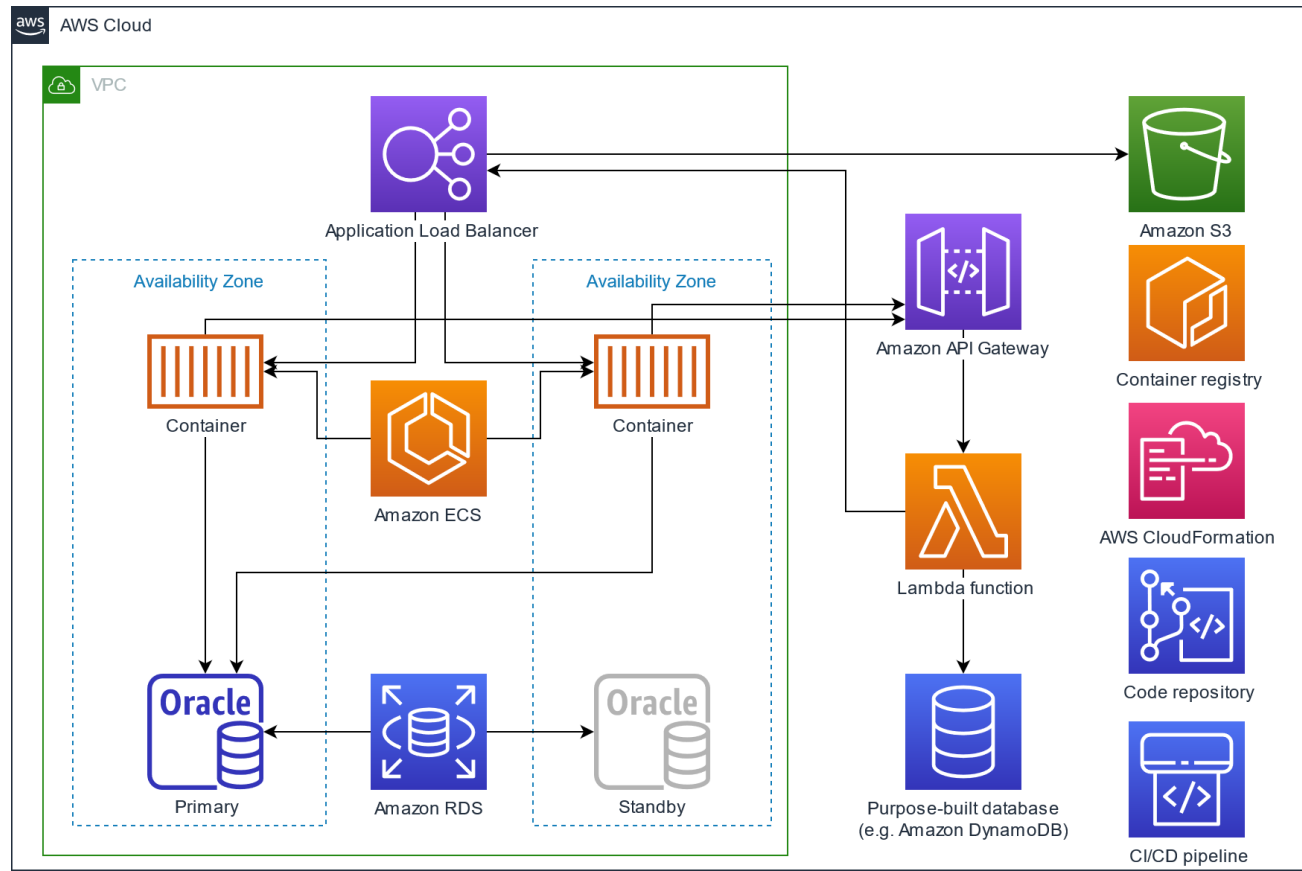
Replatform



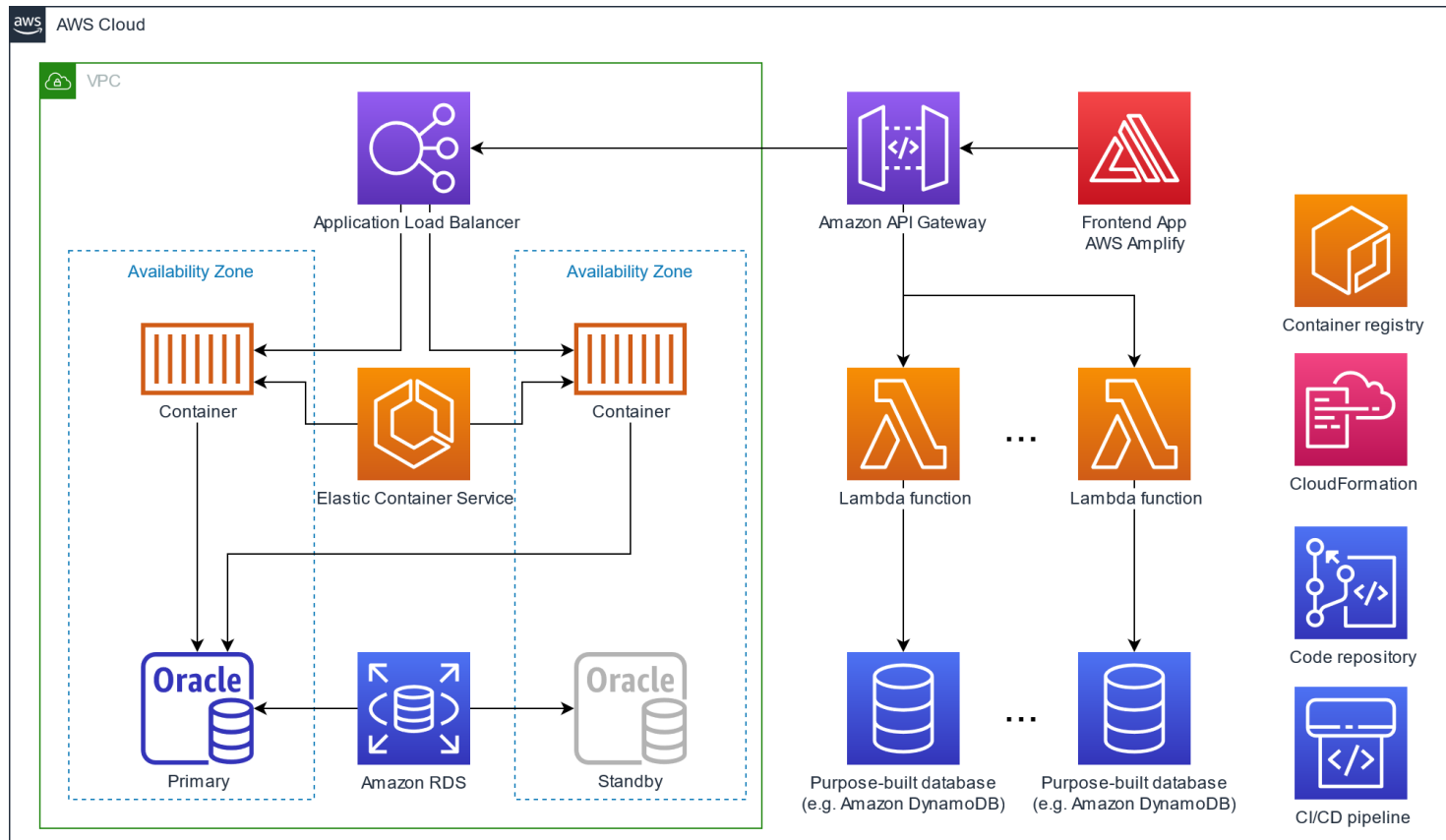
Modernization with containers



Rearchitecting/refactoring to use serverless functions



Rearchitecting/refactoring, including front-end



Other helpful AWS services



Storage

Object



Amazon
S3

Block



Amazon
EBS

File



Amazon
EFS



Amazon FSx for
Windows File Server













Amazon FSx
for Lustre

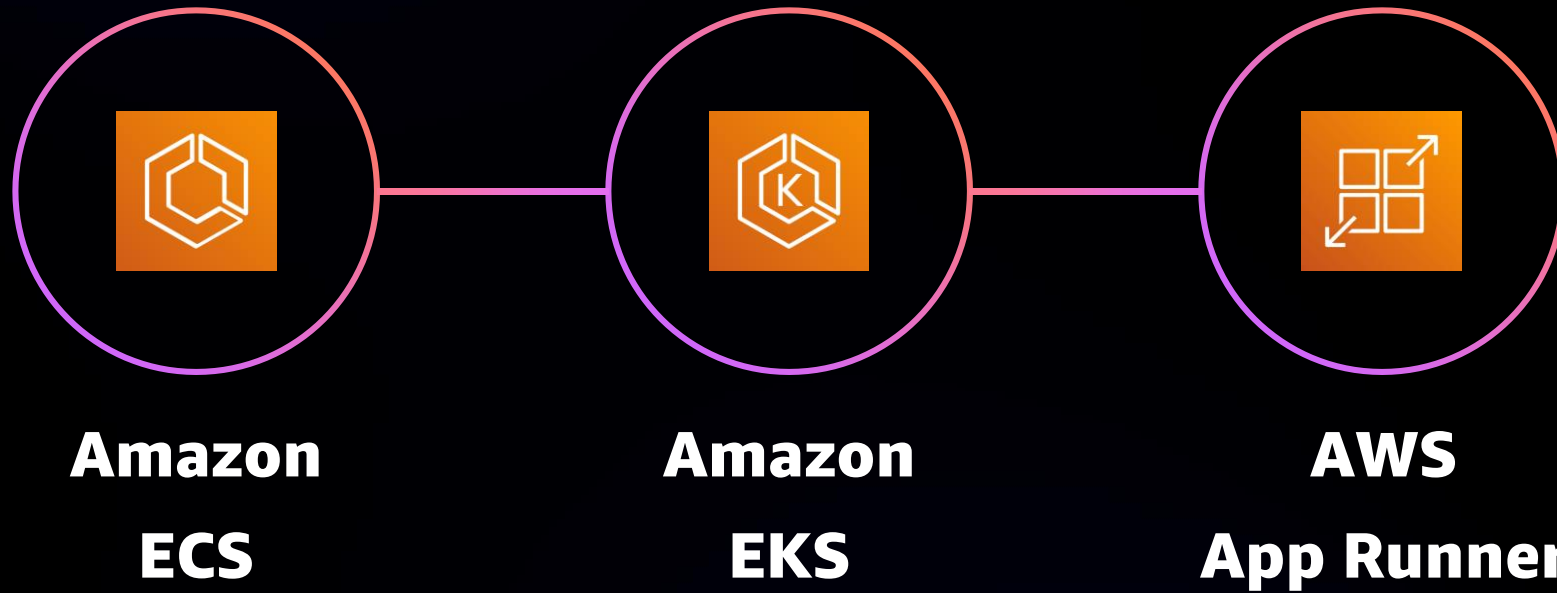


Amazon FSx
for NetApp
ONTAP

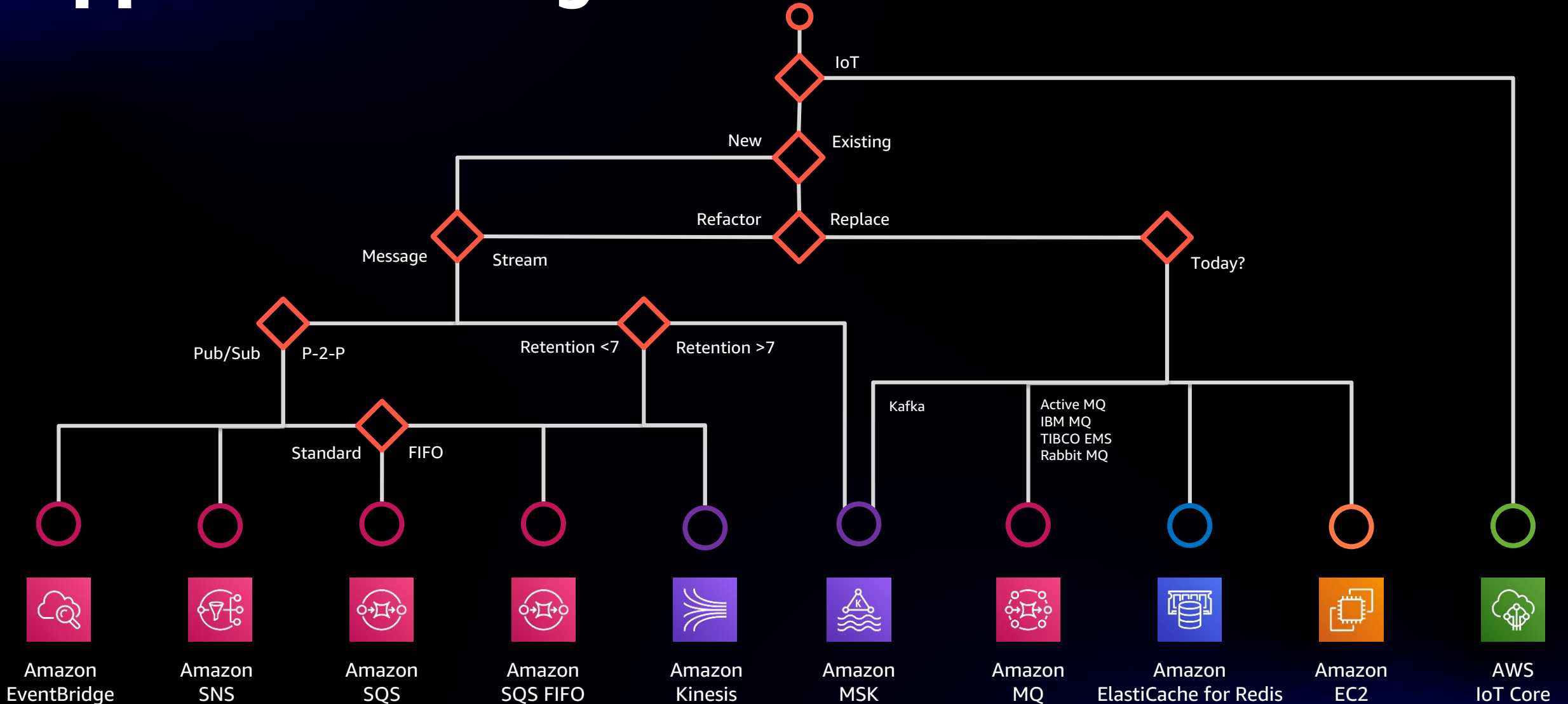
Databases

	Relational	Key value	Document	In memory	Graph	Time series	Ledger	Wide column
	Referential integrity, ACID transactions, schema-on-write	High-throughput, low-latency reads and writes; endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Collect, store, and process data sequenced by time	Complete, immutable, and verifiable history of all changes to application data	Scalable, highly available, and managed Apache Cassandra-compatible service
AWS SERVICE(S)	 Amazon RDS   Amazon Aurora Amazon Redshift	 Amazon DynamoDB	 Amazon DocumentDB	 Amazon ElastiCache	 Amazon Neptune	 Amazon Timestream	 Amazon QLDB	 Amazon Keyspaces for Apache Cassandra
COMMON USE CASES	Lift-and-shift, ERP, CRM, finance	Real-time bidding, shopping cart, social, product catalog, customer preferences	Content management, personalization, mobile	Leaderboards, real-time analytics, caching	Fraud detection, social networking, recommendation engine	IoT applications, event tracking	Systems of record, supply chain, health care, registrations, financial	Build low-latency applications, leverage open source, migrate Cassandra to the cloud

Containers



Application integration services



AWS observability portfolio

Observability

AWS native monitoring service

Amazon CloudWatch ServiceLens

Container insights

Lambda insights

Synthetics

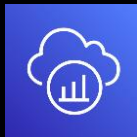
Contributor insights



Amazon CloudWatch Logs



CloudWatch metrics



AWS X-Ray

Open-source managed services

Amazon Managed Grafana  Grafana

Do it yourself (DIY)



Amazon Managed Service for Prometheus



Amazon OpenSearch Service



Jaeger & Zipkin - Tracing

Insights & ML

Instrumentation



CloudWatch agent



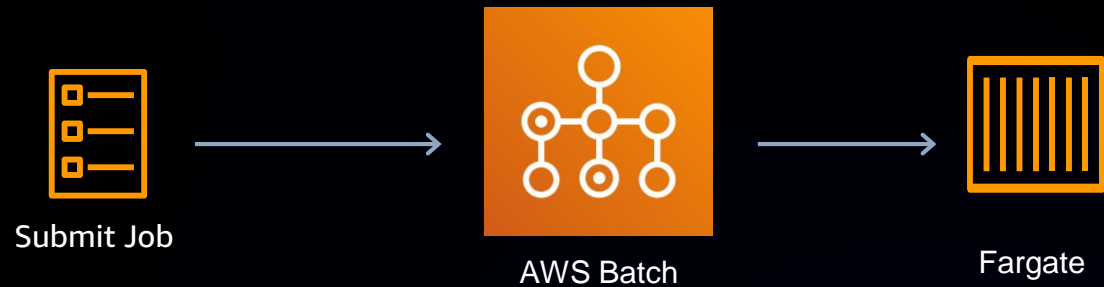
X-Ray agent



AWS Distro for OpenTelemetry (ADOT)

AWS Batch jobs for Fargate

- Fully serverless batch computing with AWS-owned compute resources; no need to specify instance type or manage machine images
- AWS Batch provides you with a managed batch queue, complete with the ability to specify priority, dependencies, and retries



- Use Fargate Spot for savings of up to 70%
- Batch handles queueing, submission, and lifecycle management

Session state

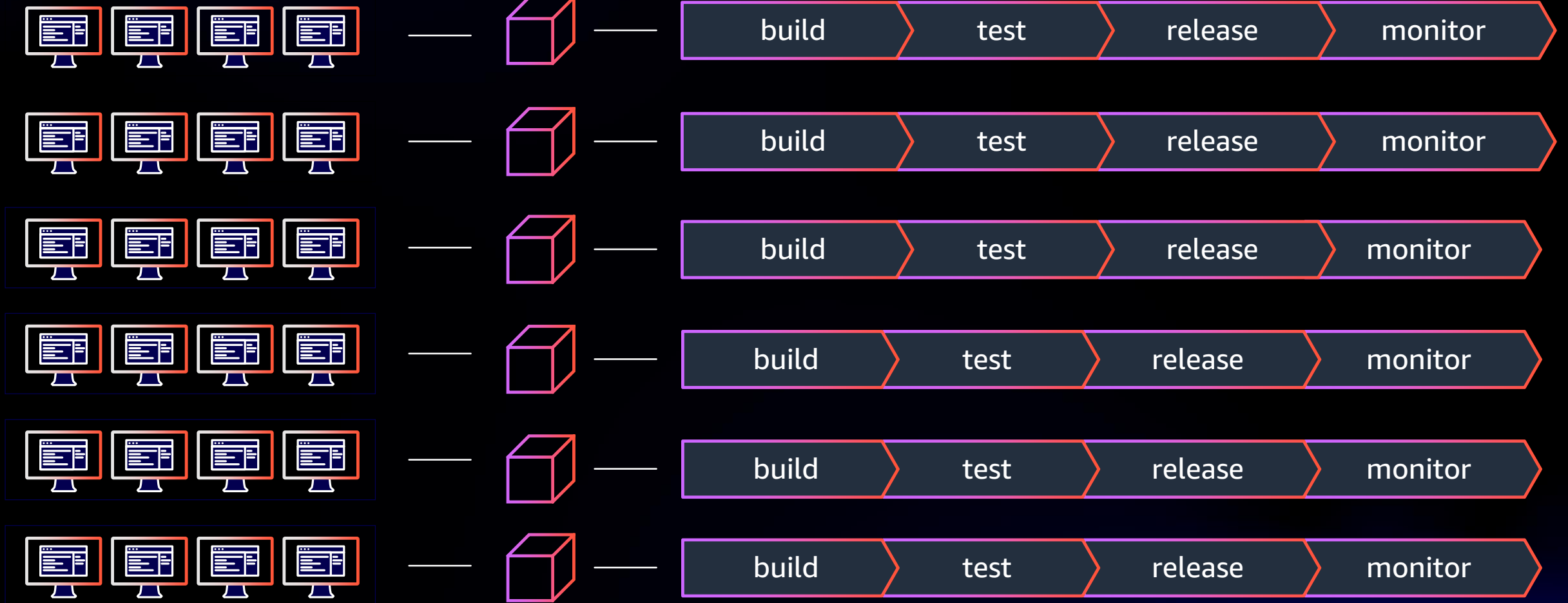
- Enable session stickiness in Elastic Load Balancing
- The JGroups communication protocol is common for synchronizing session state between application server nodes
- Multicast is not supported inside Amazon VPC
- Examples how to implement it:
 - Amazon ElastiCache for Redis in combination with Tomcat and the Redisson library: <https://github.com/redisson/redisson/tree/master/redisson-tomcat>
 - Amazon Route 53 in combination with Infinispan configuration: <https://aws.amazon.com/blogs/compute/migrate-wildfly-cluster-to-ecs-using-service-discovery/>

Development lifecycle

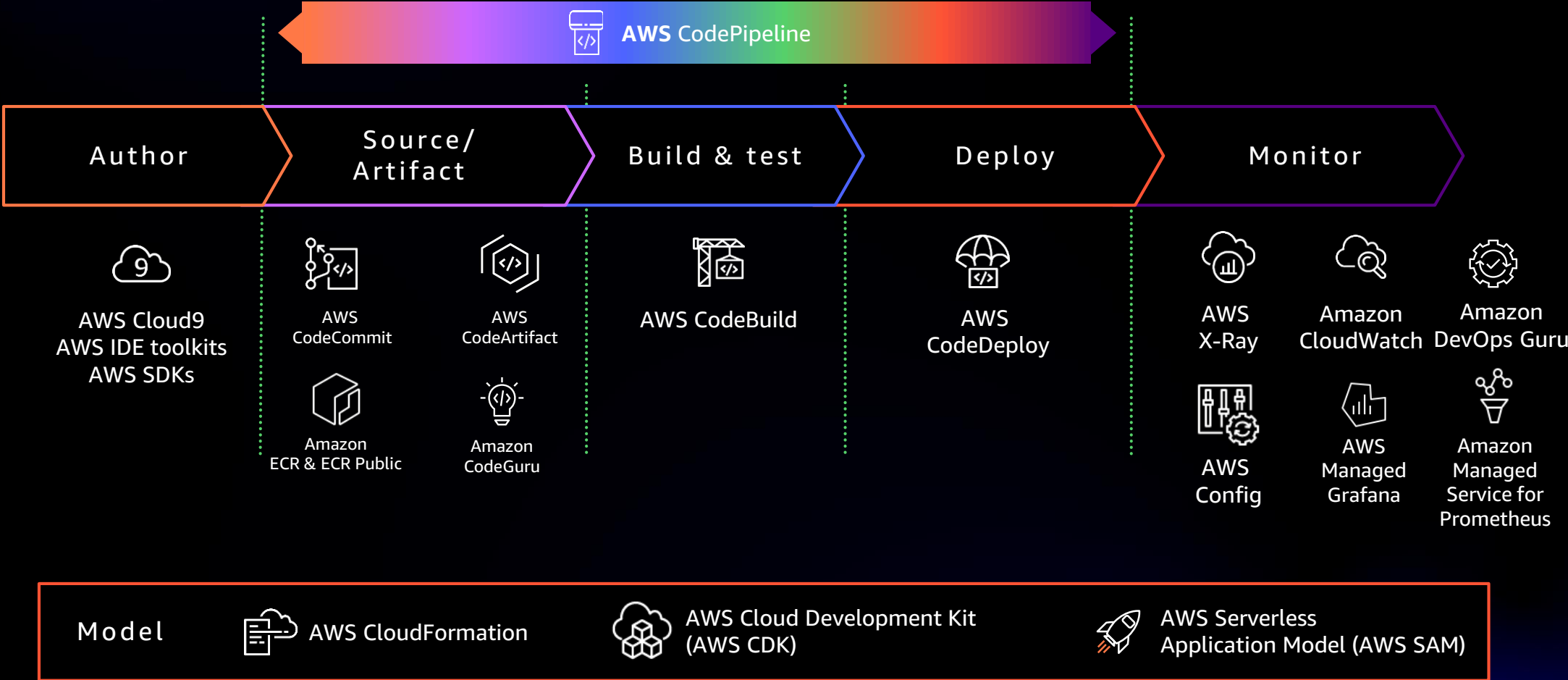
developers

applications

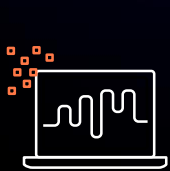
delivery pipelines



AWS developer tools



Guardrails instead of central control



Monitoring

CPU Utilization

Database
throughput

Business processes



Provisioning

Access permissions

Resource
availability

Configuration



Deployment

Time window

Toolsets available

Size or timing
of test releases



Cost management

Resource costs

Resource
utilization

Spend run rates



Security & compliance

Account
setup/access

Standards
compliance

Certificate
maintenance

AWS Proton

Increase control over your cloud infrastructure, accelerating the **pace of innovation** for your development teams



Infrastructure operators

Create application infrastructure templates



AWS Proton

Monitor and update deployments



Development teams

Find and deploy application infrastructure

Thank you!

Dennis Kieselhorst

dkieselh@amazon.de

