AWS re:Invent

ARC201-R

Comparing serverless and containers

Josh Kahn
Senior Solutions Architect
Amazon Web Services

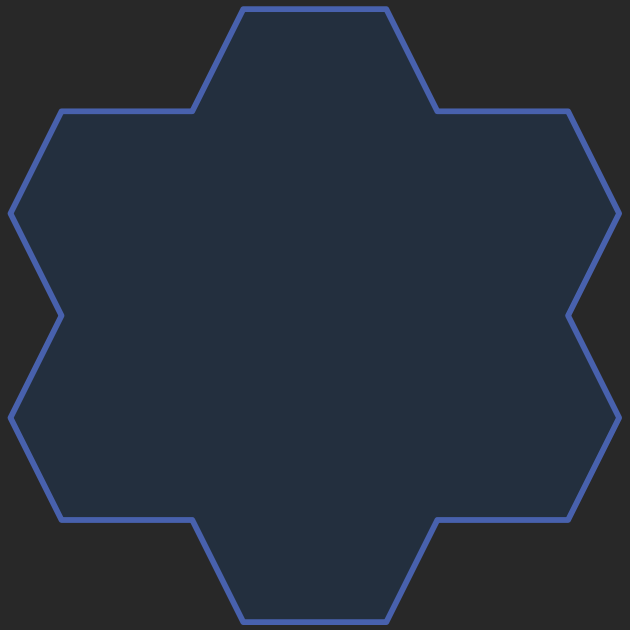
Mayank Thakkar
Global Life Sciences Sr. Solutions Architect
Amazon Web Services

Our goal

Explore the benefits and trade-offs of container and serverless architectures

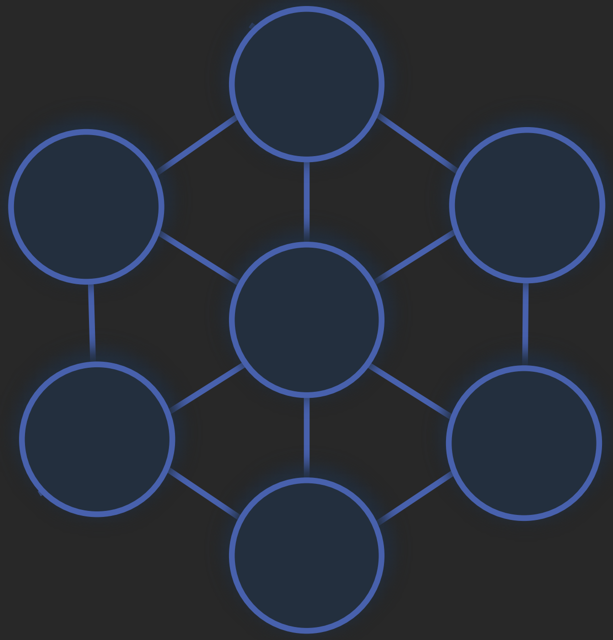
Architecture evolution

When the impact of change is small, release velocity can increase



Monolithic application

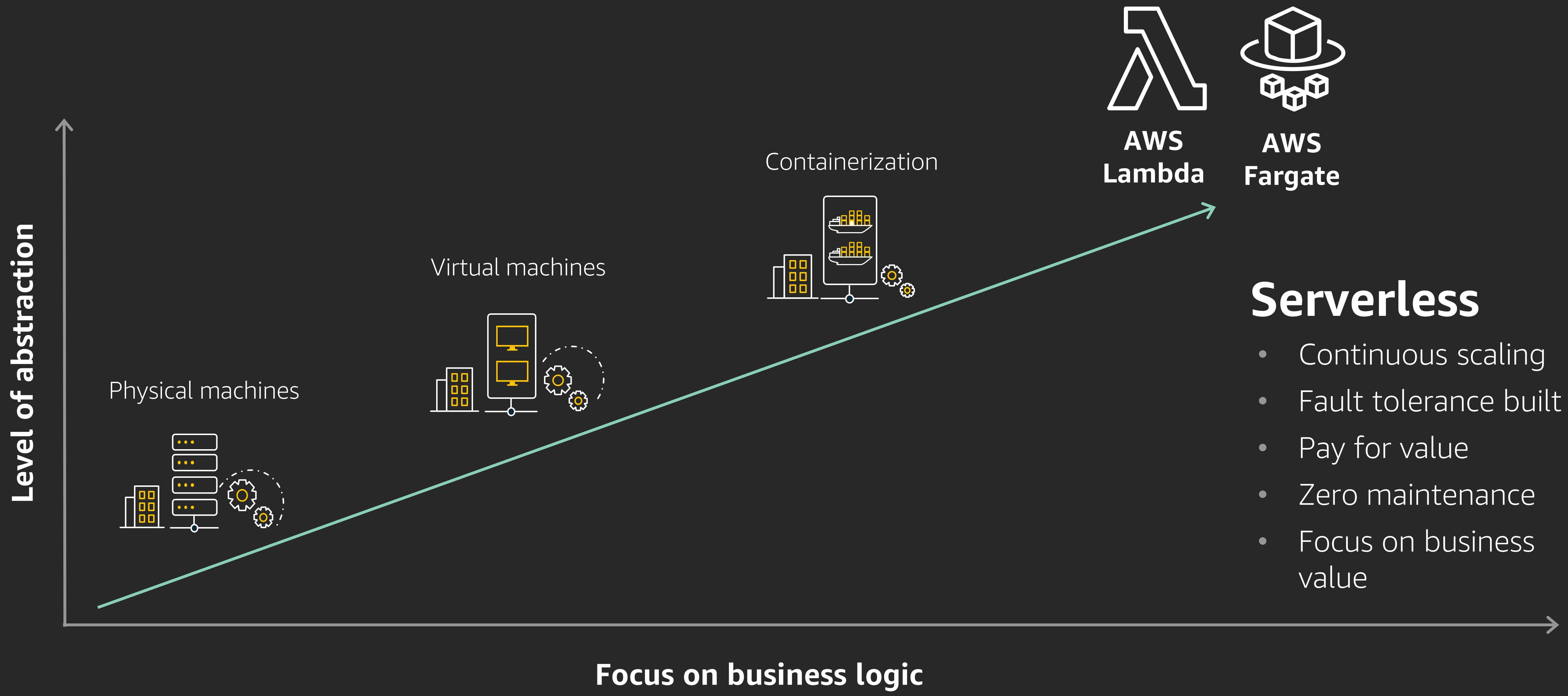
Does everything
Shared release pipeline
Rigid scaling
High impact of change
Hard to adopt new technologies



Microservices

Does one thing
Independent deployments
Independent scaling
Small impact of change
Choice of technology

We are witnessing a paradigm shift



AWS Lambda

AWS Fargate

Containerization

Virtual machines

Physical machines

Level of abstraction

Focus on business logic

Serverless

- Continuous scaling
- Fault tolerance built in
- Pay for value
- Zero maintenance
- Focus on business value

Containers & serverless on Amazon Web Services (AWS)

Options for architecting your microservices

Containers

Amazon Elastic Container Service (Amazon ECS)

Amazon Elastic Kubernetes Service (Amazon EKS)

Serverless

Lambda

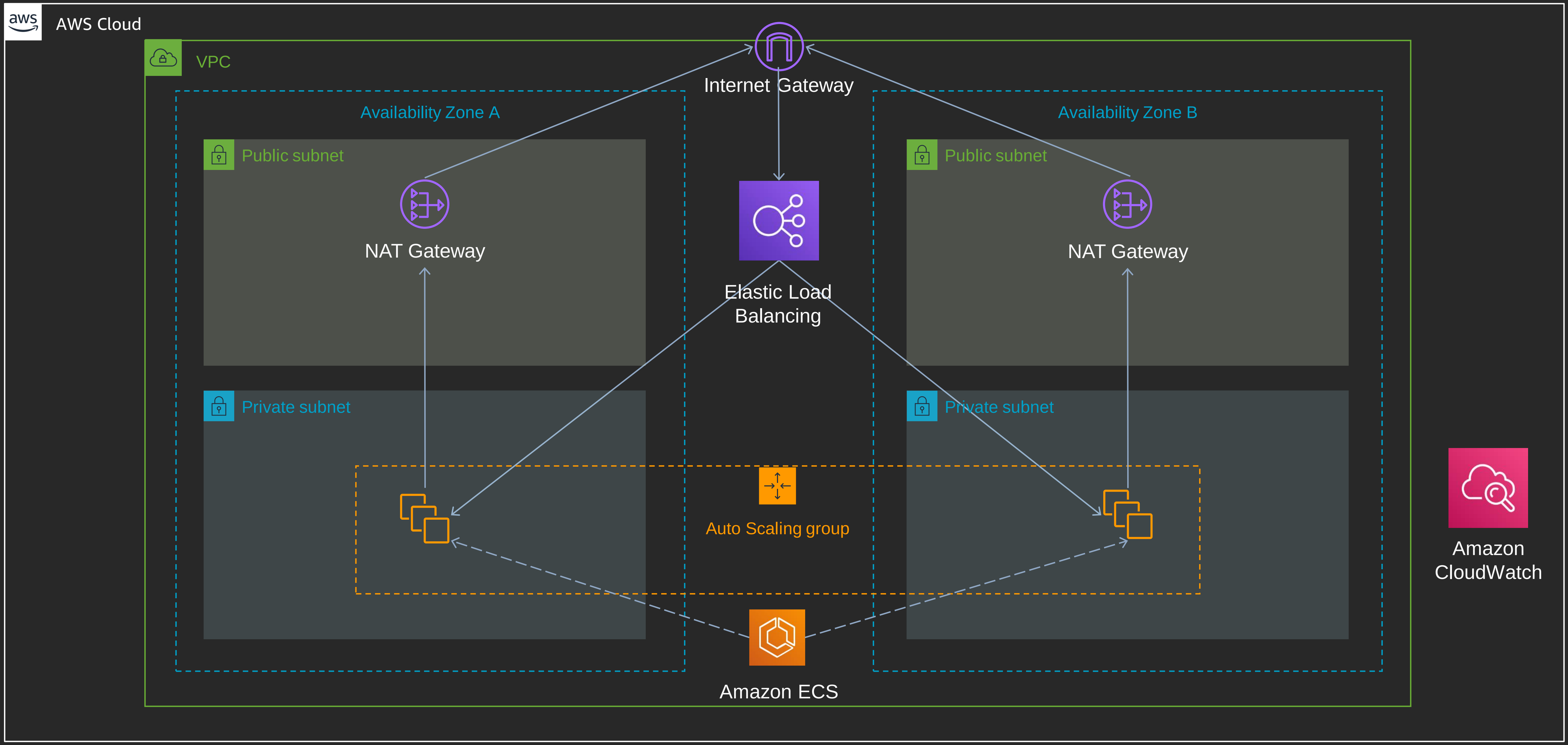
Fargate

Containers: ECS, EKS, Fargate

- Platforms to run containerized applications on AWS
- Easily run and scale applications to meet your needs
- Native integration with other AWS services
- Pairs well with CI/CD

- Benefits
  - Portability
  - Control
  - Rich ecosystem

AWS container web architecture



AWS container services landscape

Management

Deployment, scheduling, scaling & management of containerized applications

Amazon ECS

Amazon EKS

Hosting

Where the containers run

Amazon EC2

Fargate

Image Registry

Container image repository

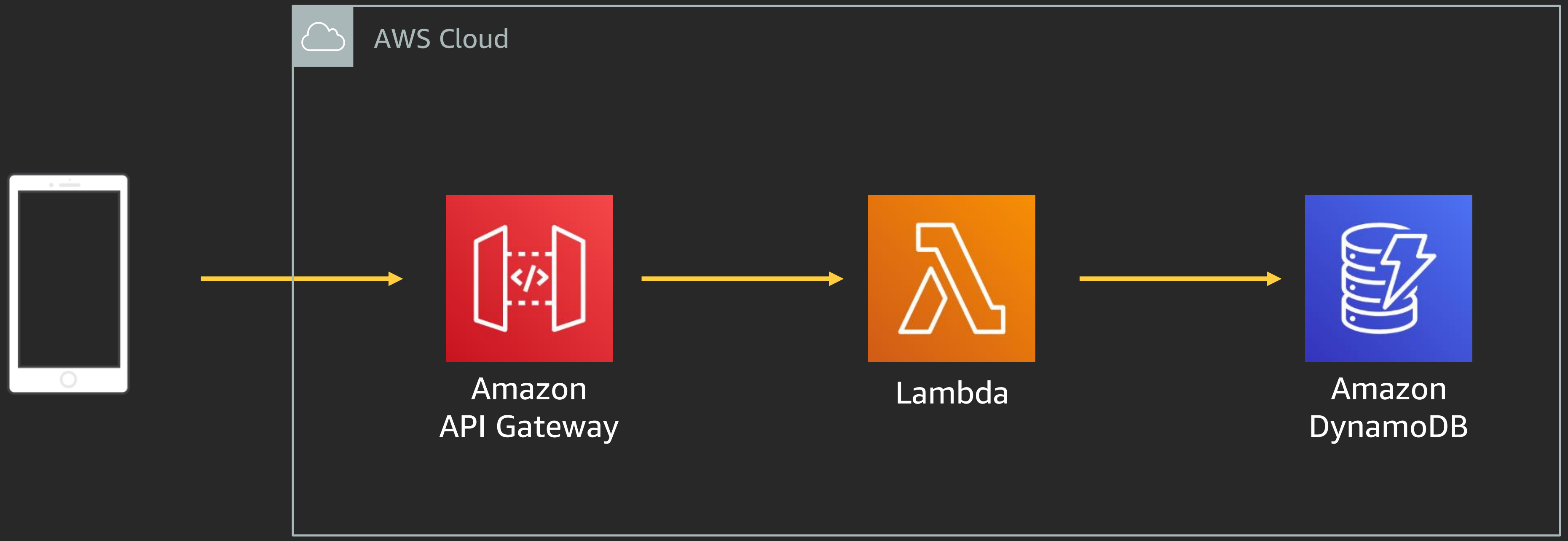
Amazon Elastic Container Registry (Amazon ECR)

Serverless: Lambda

- Run code without thinking about servers
  - Fully managed—no servers to provision or manage
- Scales with usage
- Pay for value
- Native integration with other AWS services

- Benefits
  - Event-driven framework
  - Multiple invocation models
  - Opinionated

AWS serverless web architecture



Architectural considerations

EKS/ECS/Fargate

- Custom code & services: Build your own environment
- Numerous choices
- Requires management and orchestration*

Lambda

- Standardized choices
- Opinionated approach drives scalable platform
- Security and scaling managed by AWS
- Serverless and “containerless”

* Fargate streamlines this model

Other considerations

EKS/ECS/Fargate

- Wide array of power options
  - Power level managed by container, scoped by host
  - Fargate offers streamlined model
- Multiple networking modes
- Mature tooling

Lambda

- Easy power level selection
  - 128MB to 3GB
  - Network and CPU assigned proportionally
- Can overwhelm backends
- Developing tooling
- Organizational readiness?

How do I decide?

When to leverage ECS/EKS (EC2-backed)

When you need . . .

- Support for long-running compute jobs (> 15 minutes)
- Predictable, high traffic usage
- Lower runtime startup latency (sub-second)
- Application with a non-HTTP/S listener
- Stateful applications (including EFS integration)
- Agent/daemon/side car to run alongside your service
- Specialized hardware (GPUs, etc.) or kernel tuning support
- Support for Windows containers or legacy .NET

When you want . . .

- Complete control of compute environment (but not just for the sake of control)
- Hybrid scenarios or on-premises portability
- Container image portability with Docker runtime
- Ability to purchase through different billing models (On-Demand, RI, Spot)

When to leverage AWS Fargate

When you need . . .

- Support for long-running compute jobs (> 15 minutes)
- More compute than Lambda offers (>3 GB memory)
- Application with a non-HTTP/S listener
- Run side cars with your service (agents only supported as side cars)
- Predictable scaling OR longer start times are acceptable

When you want . . .

- Managed serverless container environment
- Container image portability with Docker runtime

When to leverage Lambda

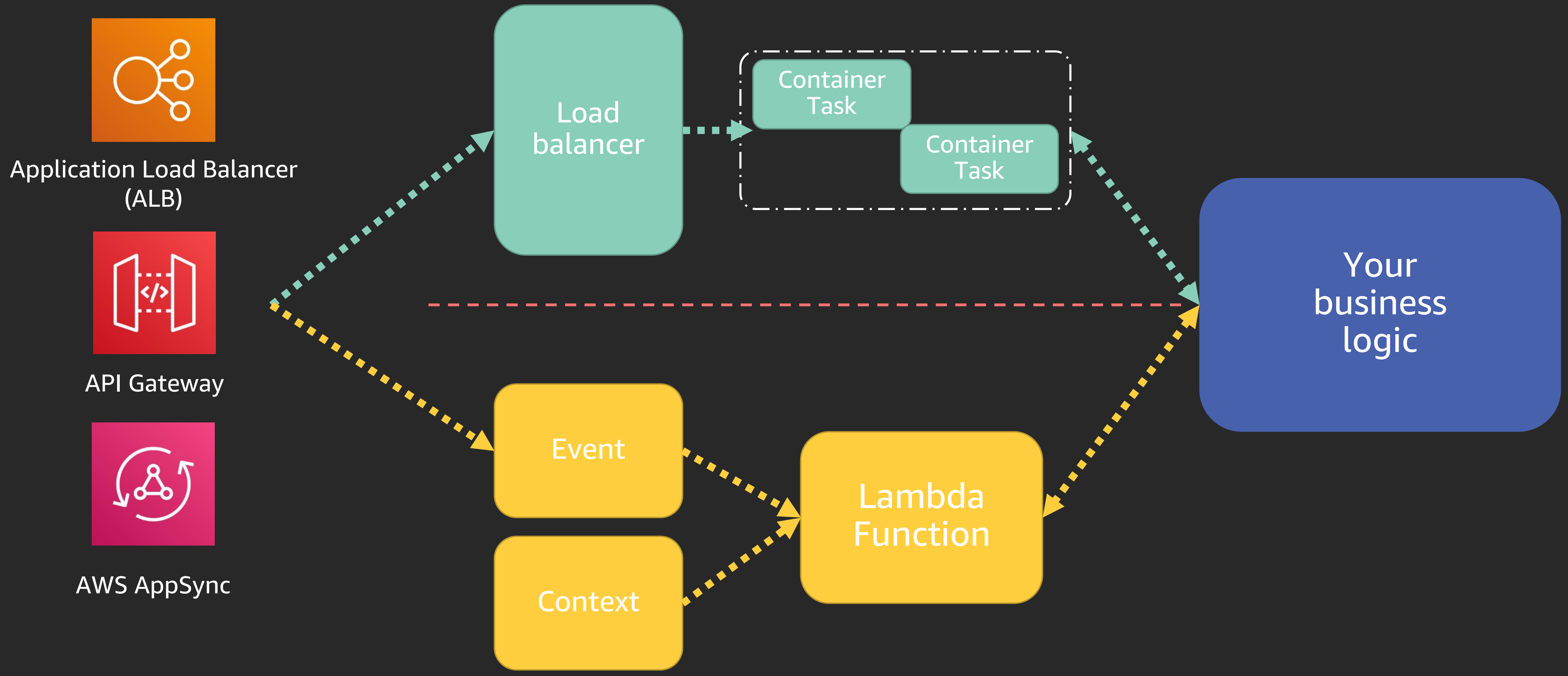
When you need . . .

- To trigger action on an event
- Support for varying utilization
- Ability to handle unknown demand
- Lighter-weight, application-focused stateless computing

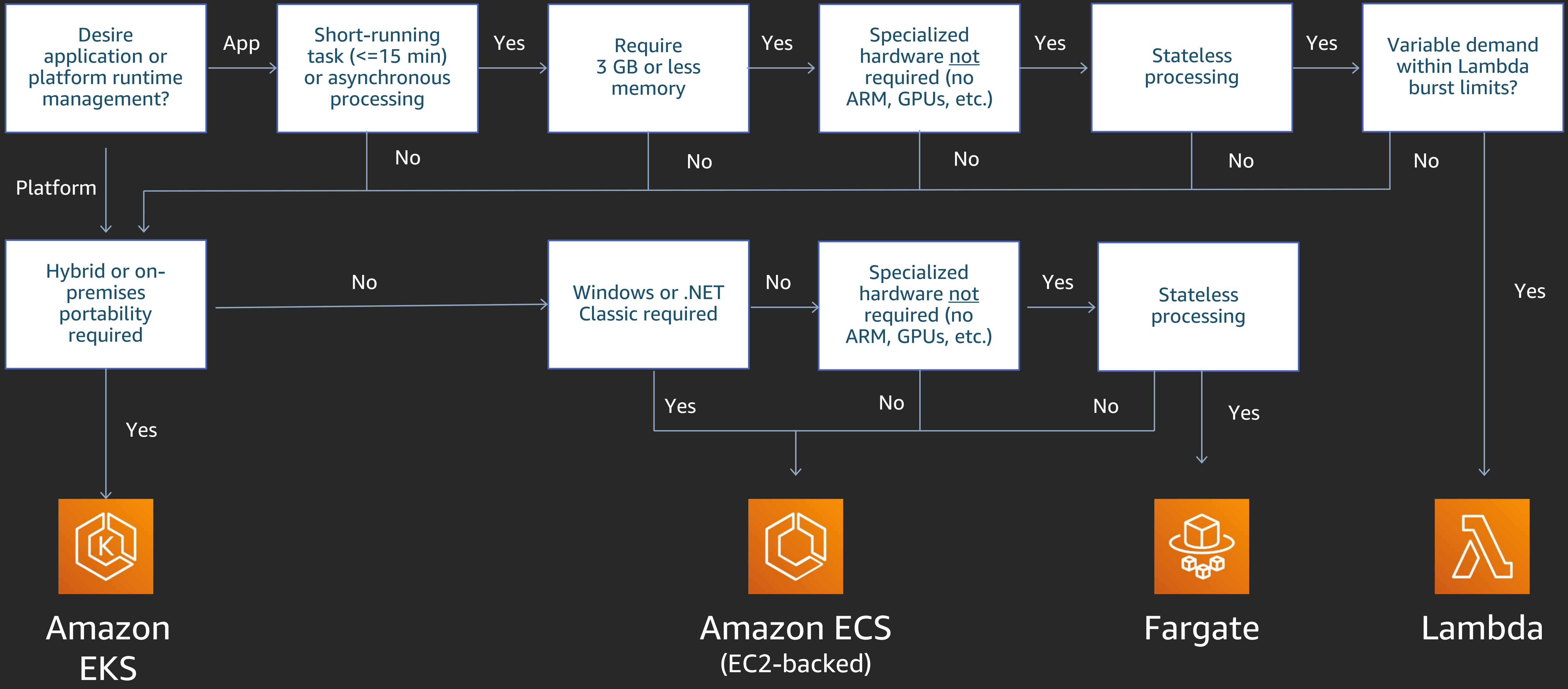
When you want to . . .

- Focus on business logic and not IT operations
- Enable simplified IT automation
- Enable real-time data processing or serverless backend
- Hand operational complexity (patching, scaling, fault tolerance) to AWS
- Reduce complexity for development and operations

What if I can't decide?



Decision tree



Desire application or platform runtime management?

App

Short-running task (<=15 min) or asynchronous processing

Yes

Require 3 GB or less memory

Yes

Specialized hardware not required (no ARM, GPUs, etc.)

Yes

Stateless processing

Yes

Variable demand within Lambda burst limits?

No

No

No

No

No

Platform

Hybrid or on-premises portability required

No

Windows or .NET Classic required

No

Specialized hardware not required (no ARM, GPUs, etc.)

Yes

Stateless processing

Yes

Yes

No

No

Yes

Yes

Amazon EKS

Amazon ECS (EC2-backed)

Fargate

Lambda

To learn more . . .

Containers

https://aws.amazon.com/containers/

Serverless

https://aws.amazon.com/serverless/

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/

Thank you!

Josh Kahn

jkahn@amazon.com

Mayank Thakkar

mbt@amazon.com

❗

Please complete the session survey in the mobile app.