PUBLIC SECTOR SYMPOSIUM

BRUSSELS | MARCH 28, 2023

BAT305

Dive deep on when to choose serverless or containers on AWS

Yohan Wadia

AWS Solutions Architect

Disclaimer: *May contain material some viewers may find objectionable; cloud architecture guidance is advised.*

Image source: https://knowyourmeme.com/memes/why-would-you-say-something-so-controversial-yet-so-brave

Agenda

Asking common challenges/questions

Understanding how to run code using AWS

Selecting the right service for the right job

Defining a strategy

Conclusions and further reading

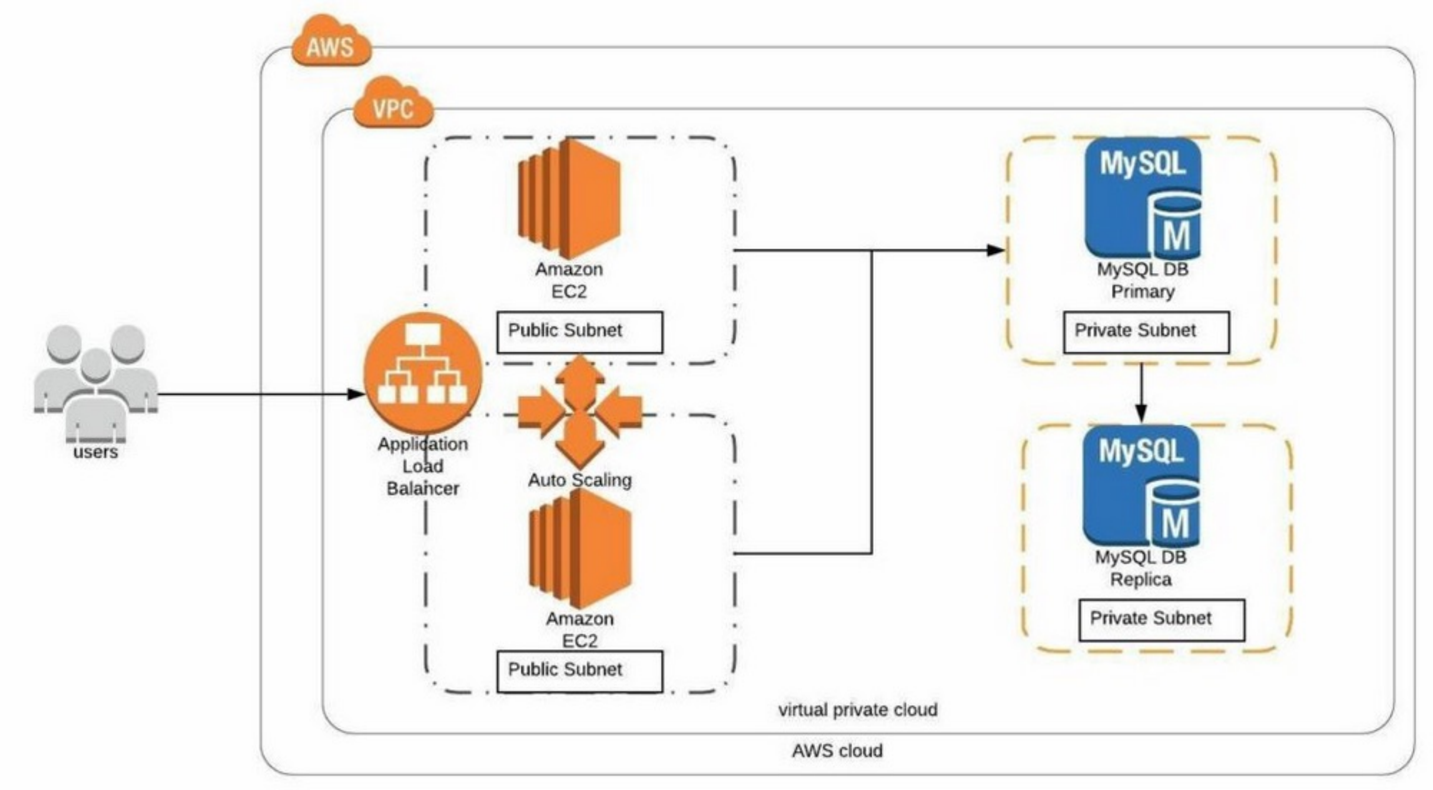
Challenge (*and goal*)

“How do I pick the right path to modernize my development and/or my deployment on AWS?”

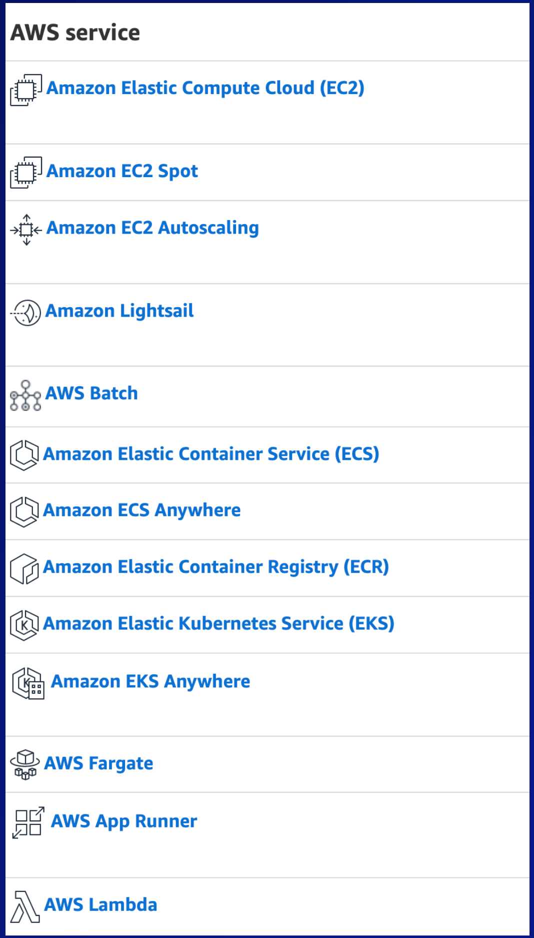
“What is the right strategy when it comes to deciding between serverless and containers?”

“Which is the right service for my modern application?”

In the good old days there was one way (EC2)



Today there are more choices!



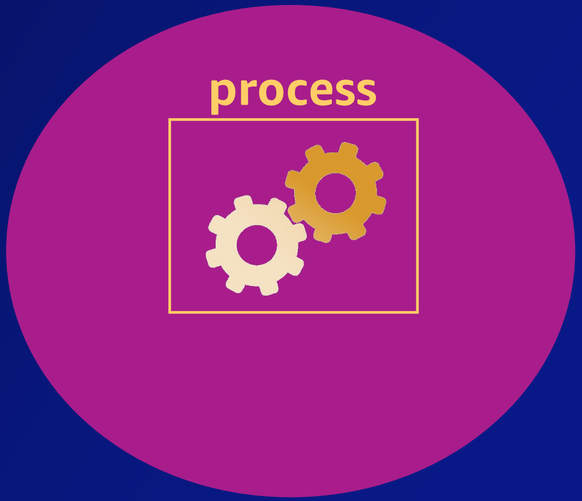
AWS service
Amazon Elastic Compute Cloud (EC2)
Amazon EC2 Spot
Amazon EC2 Autoscaling
Amazon Lightsail
AWS Batch
Amazon Elastic Container Service (ECS)
Amazon ECS Anywhere
Amazon Elastic Container Registry (ECR)
Amazon Elastic Kubernetes Service (EKS)
Amazon EKS Anywhere
AWS Fargate
AWS App Runner
AWS Lambda

Customers love that they can pick the right tool for the job but that comes with some decision fatigue

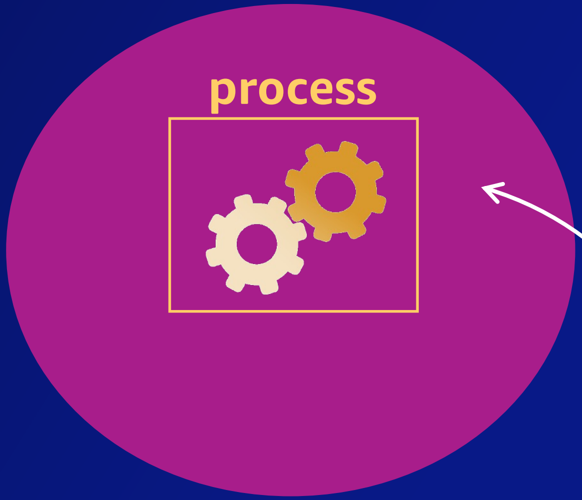
Image source: https://imgflip.com/memetemplate/326050501/Kid-in-a-candy-shop

Understanding the Container Model

Containers: Programming model or package?

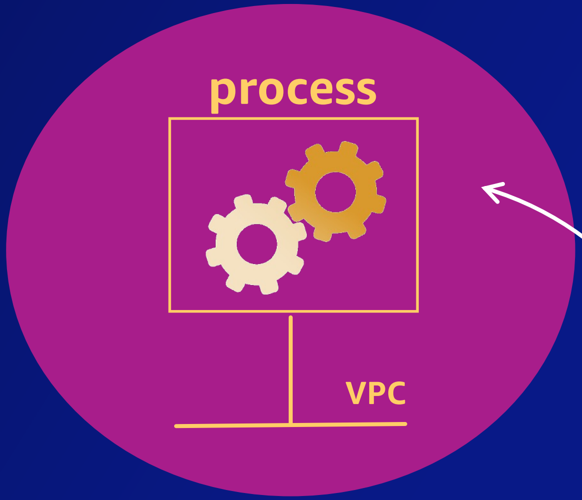


Containers: Programming model or package?



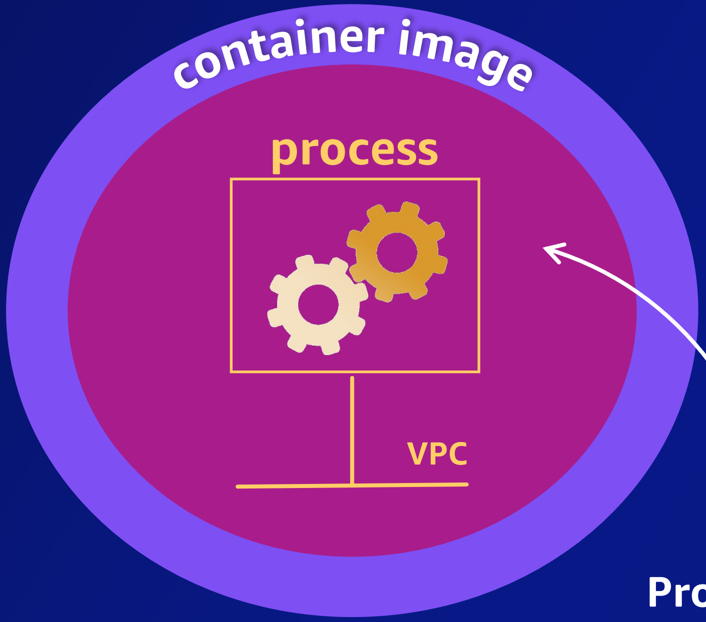
Process-based workload (aka there is likely a binary)

Containers: Programming model or package?



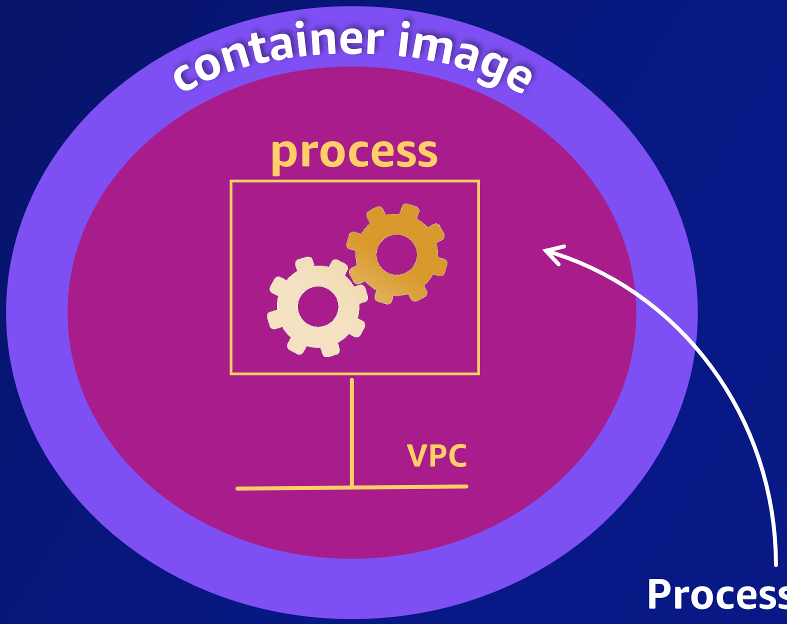
Process-based
workload (aka there
is likely a binary)

Containers: Programming model or package?

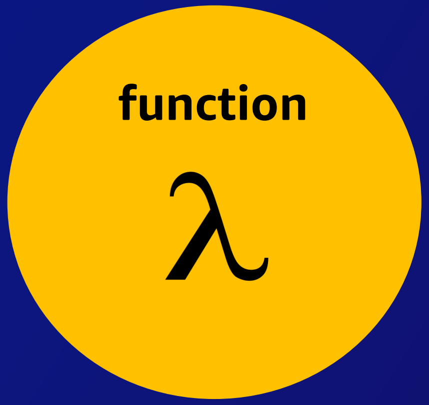


Process-based workload (aka there is likely a binary)

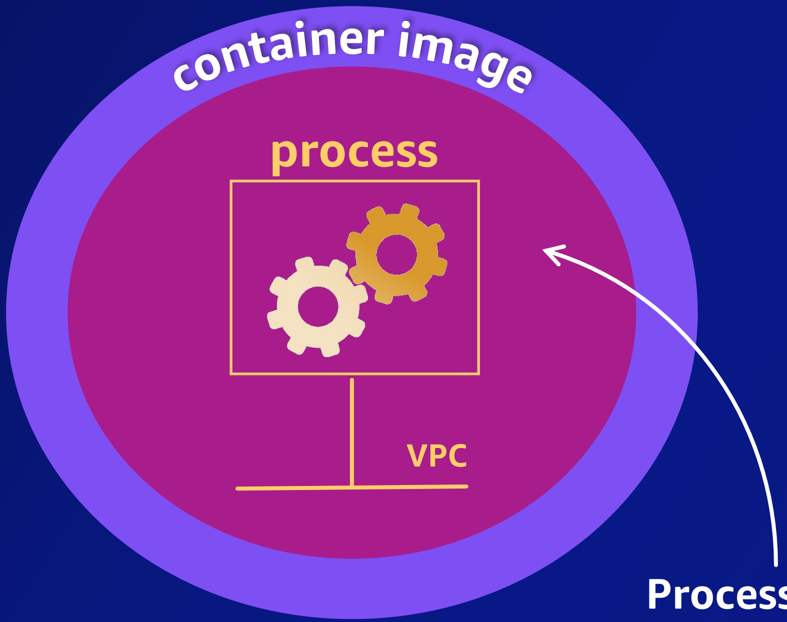
Containers: Programming model or package?



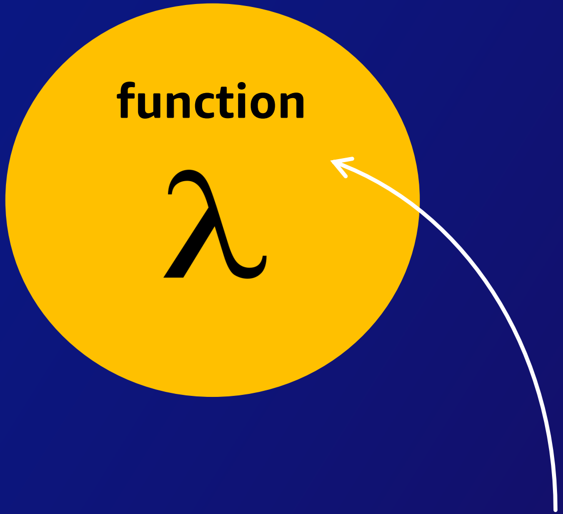
Process-based
workload (aka there
is likely a binary)



Containers: Programming model or package?

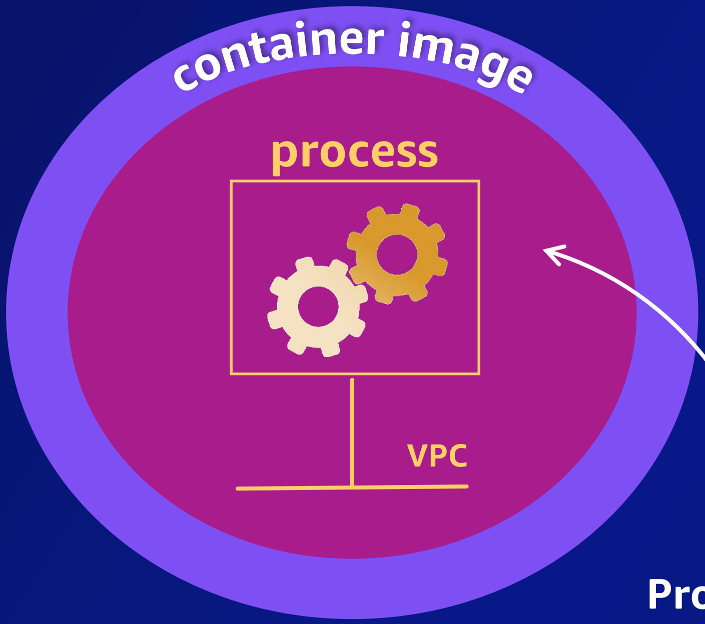


Process-based workload (aka there is likely a binary)

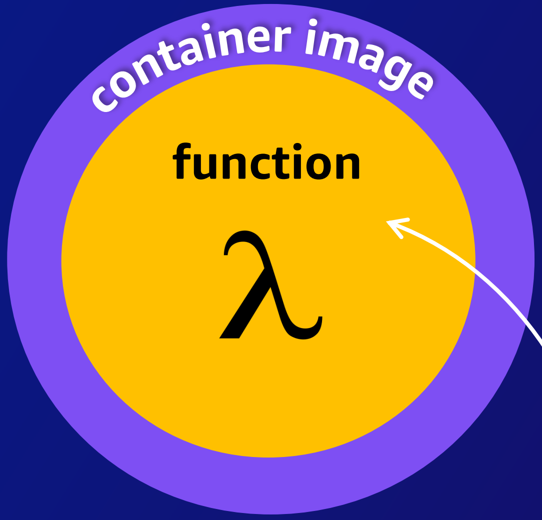


Event-driven applications

Containers: Programming model or package?

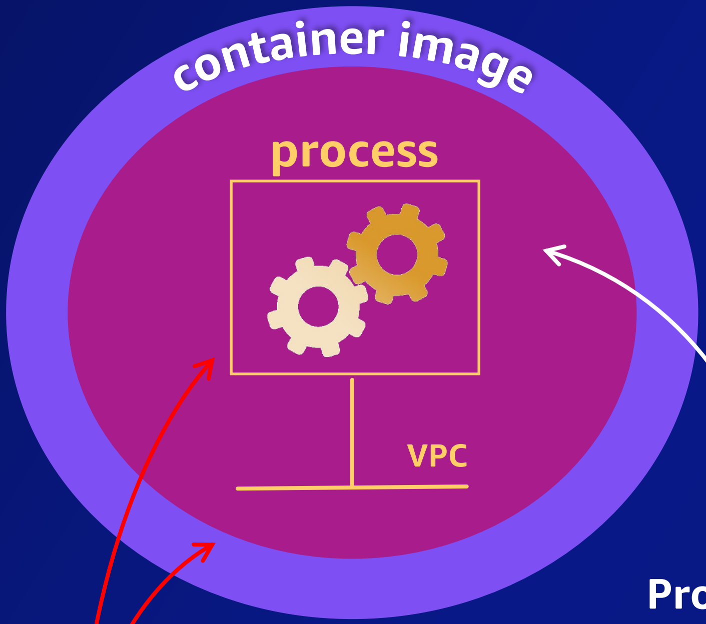


Process-based workload (aka there is likely a binary)

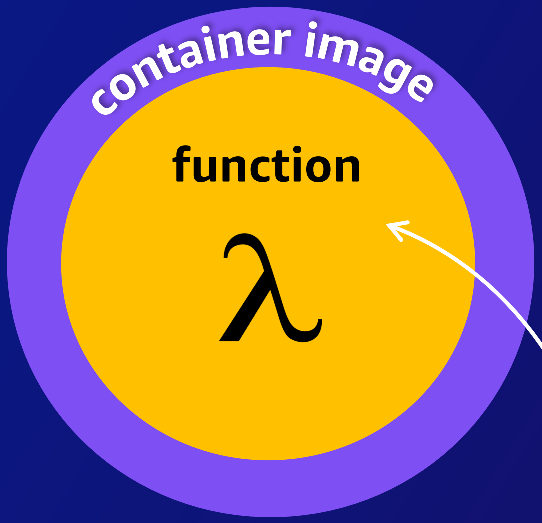


Event-driven applications

Containers: Programming model or package?



Process-based workload (aka there is likely a binary)

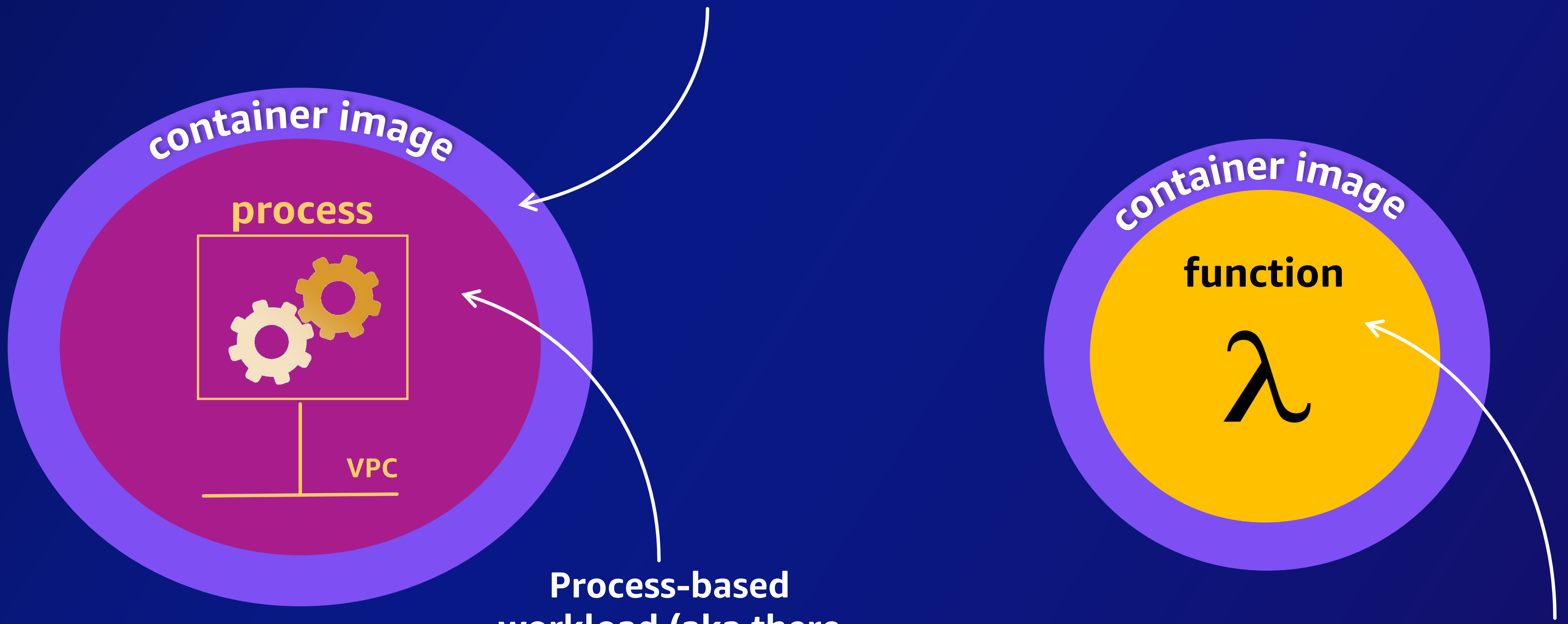


Event-driven applications

We historically thought of containers as the combination of these two entities

Containers: Programming model or package?

The container image format (OCI) as a universal packaging mechanism



Process-based workload (aka there is likely a binary)

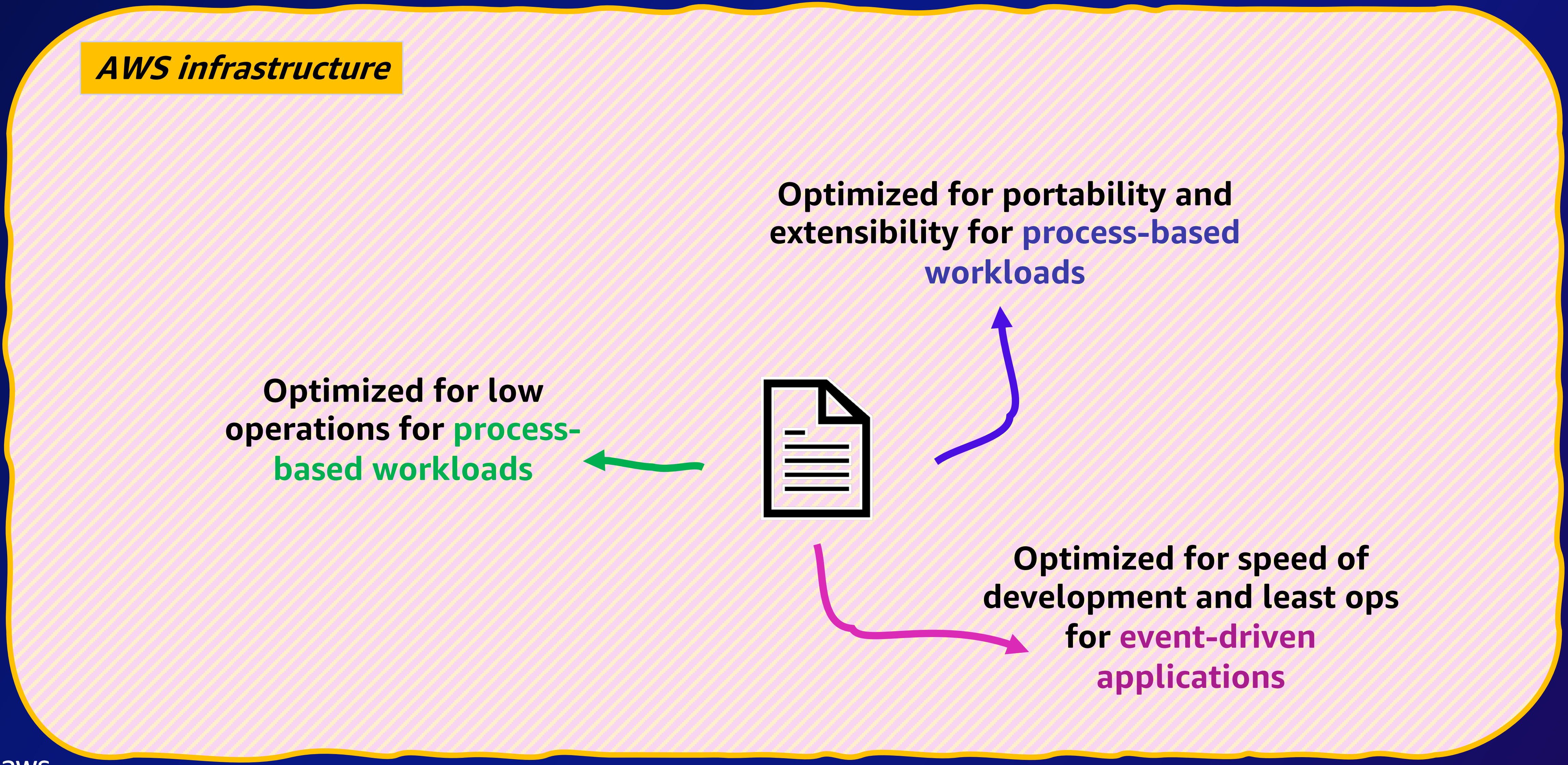
Event-driven applications

"How do I pick the right path to modernize my development and/or my deployment on AWS?"

3 major models for ”running code” with AWS

AWS infrastructure

3 major models for "running code" with AWS



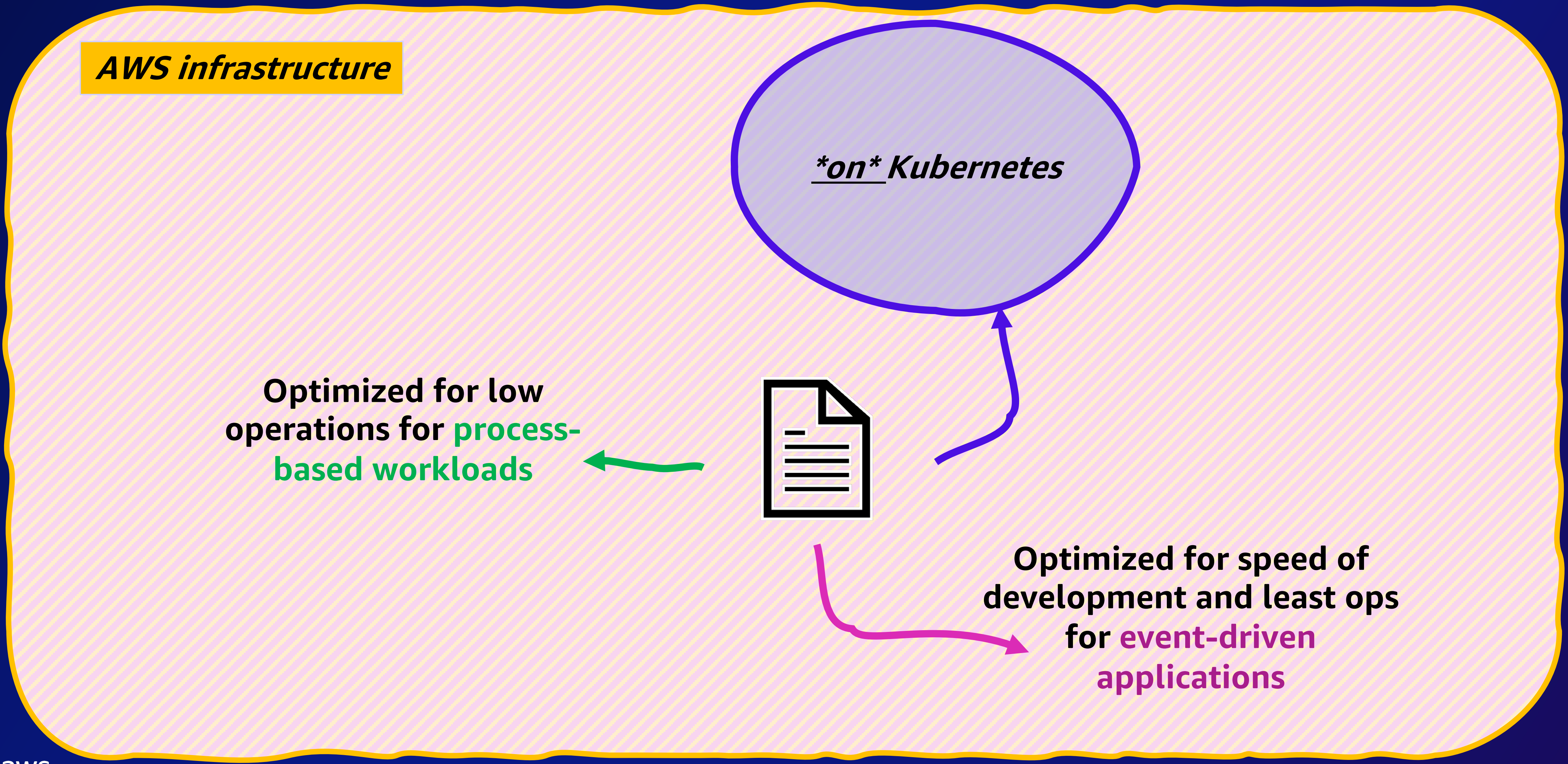
AWS infrastructure

Optimized for portability and extensibility for process-based workloads

Optimized for low operations for process-based workloads

Optimized for speed of development and least ops for event-driven applications

3 major models for "running code" with AWS



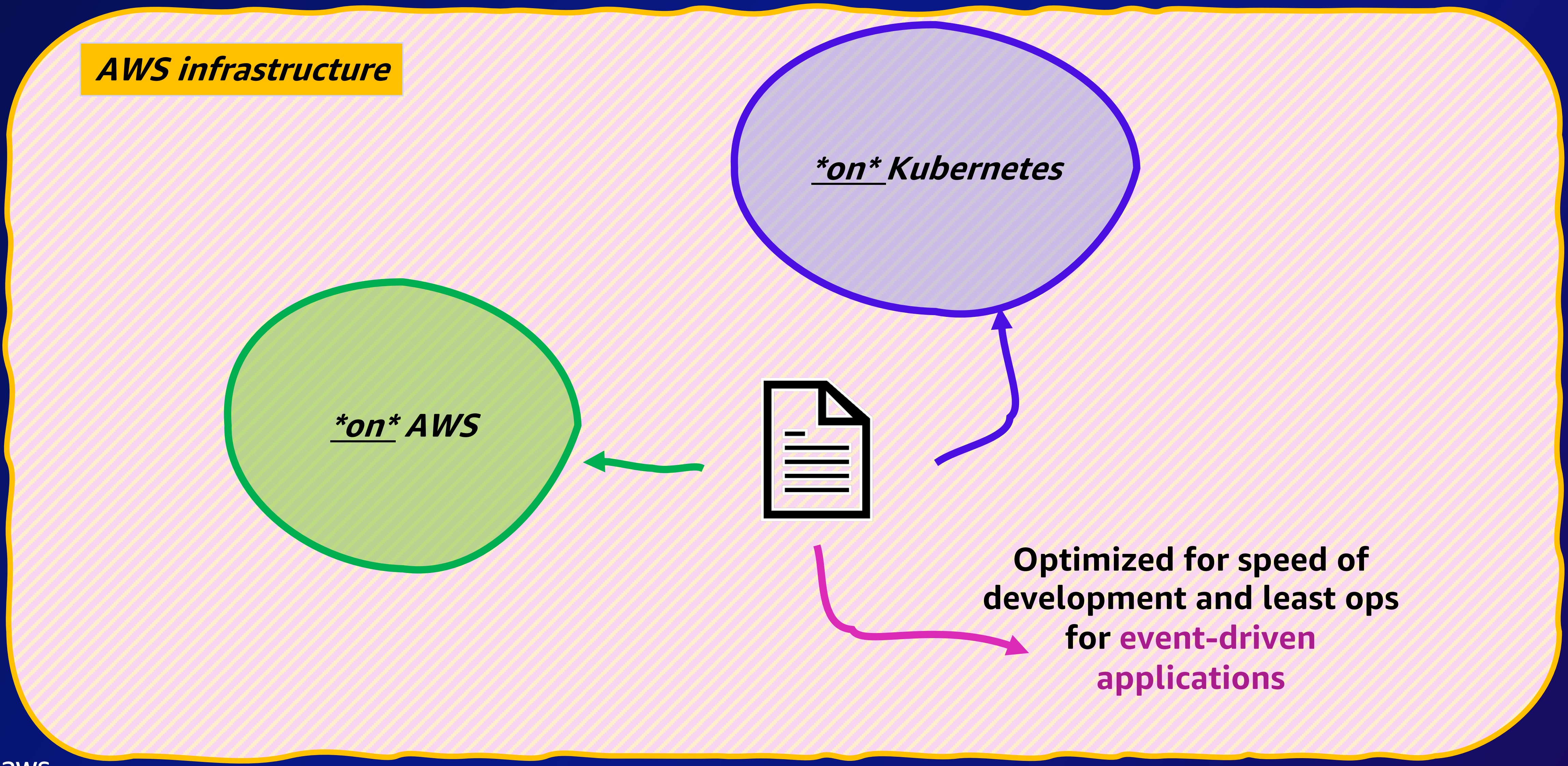
AWS infrastructure

**on* Kubernetes*

Optimized for low operations for process-based workloads

Optimized for speed of development and least ops for event-driven applications

3 major models for ”running code” with AWS



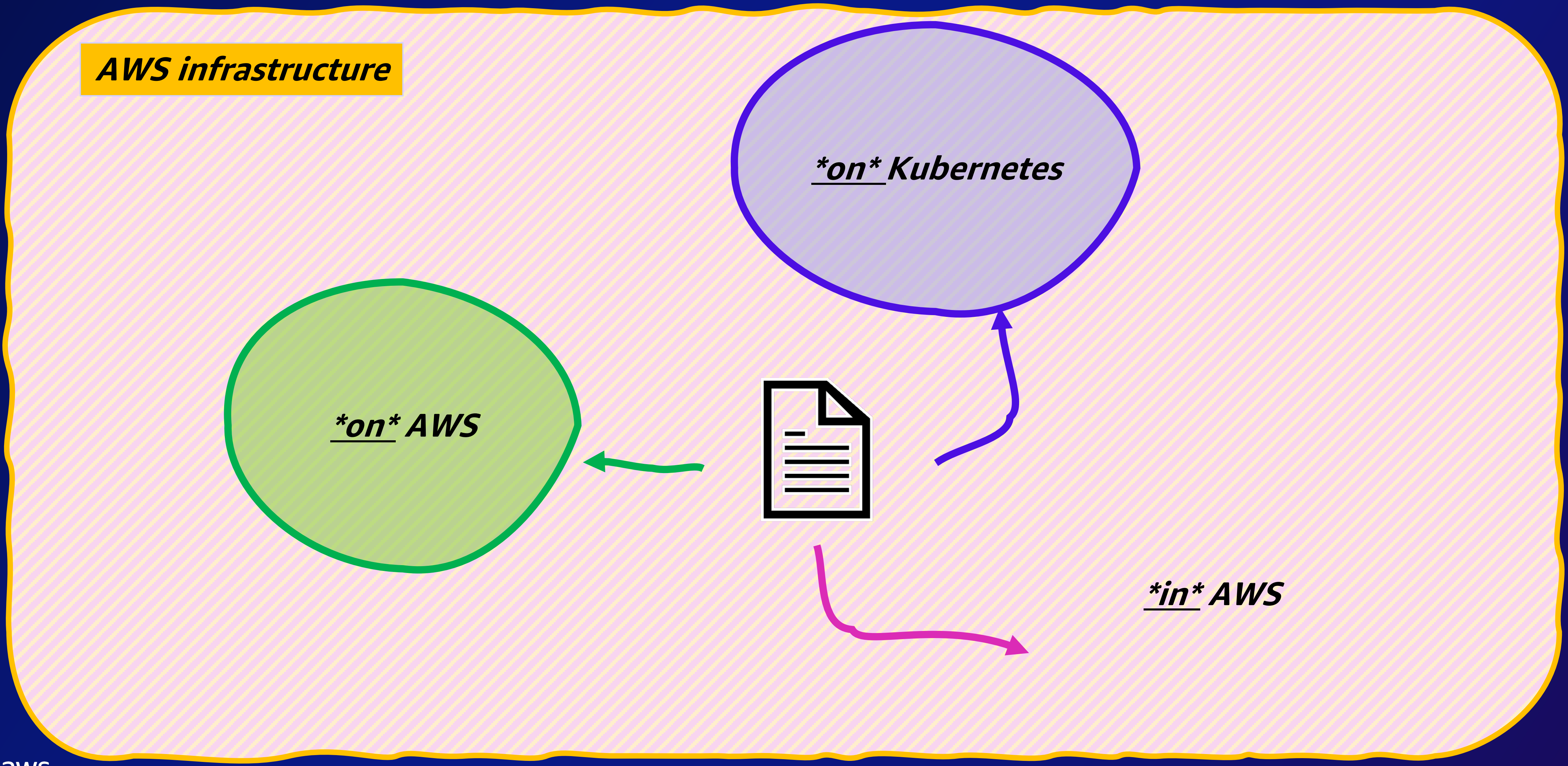
AWS infrastructure

on Kubernetes

on AWS

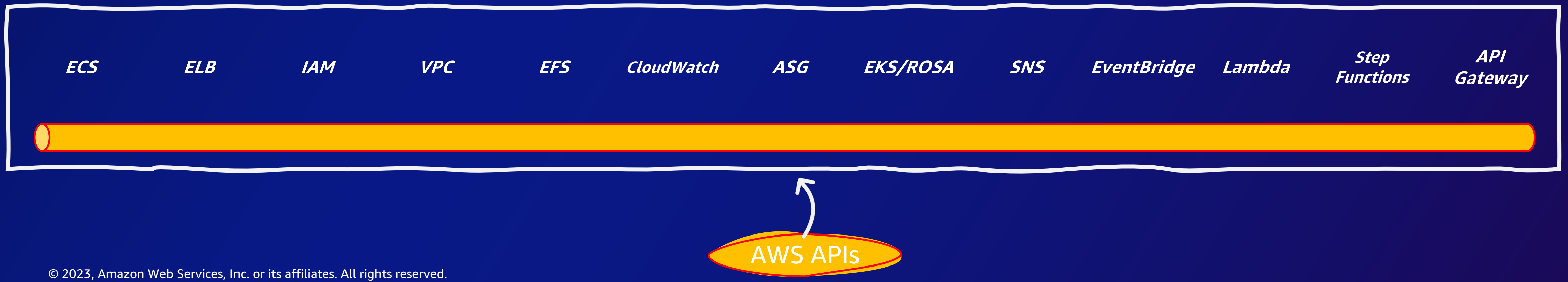
Optimized for speed of development and least ops for event-driven applications

3 major models for "running code" with AWS



Let’s take it apart!

Let’s take it apart

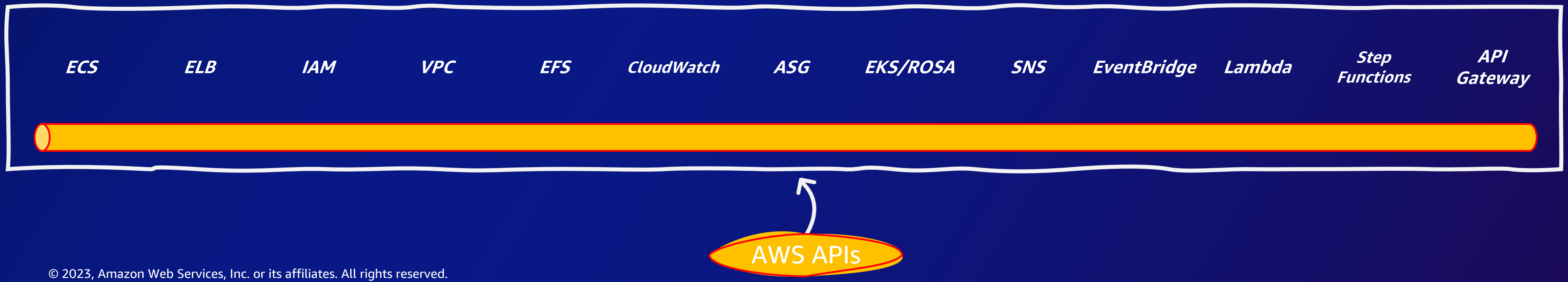


ECS *ELB* *IAM* *VPC* *EFS* *CloudWatch* *ASG* *EKS/ROSA* *SNS* *EventBridge* *Lambda* *Step Functions* *API Gateway*

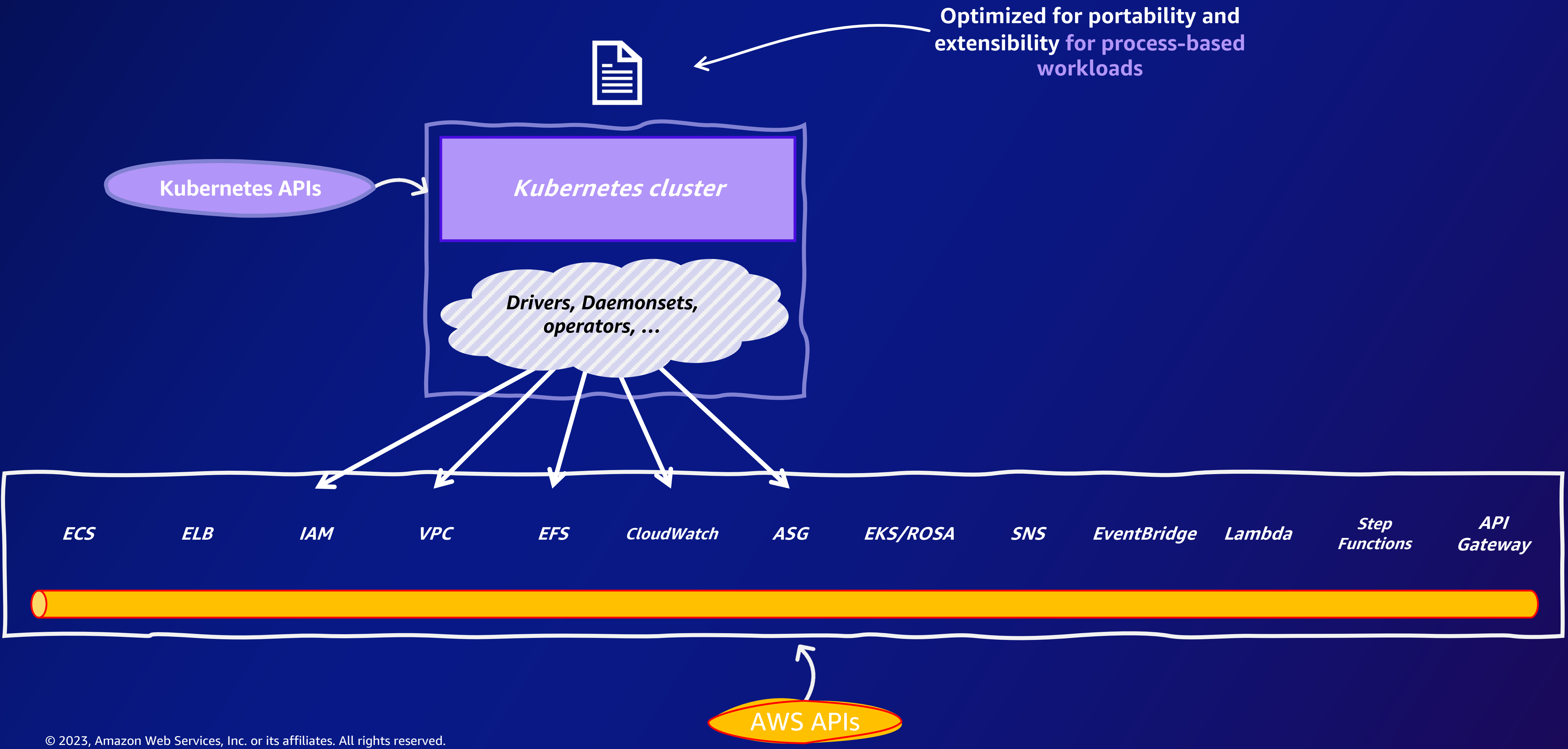
AWS APIs

Let’s take it apart

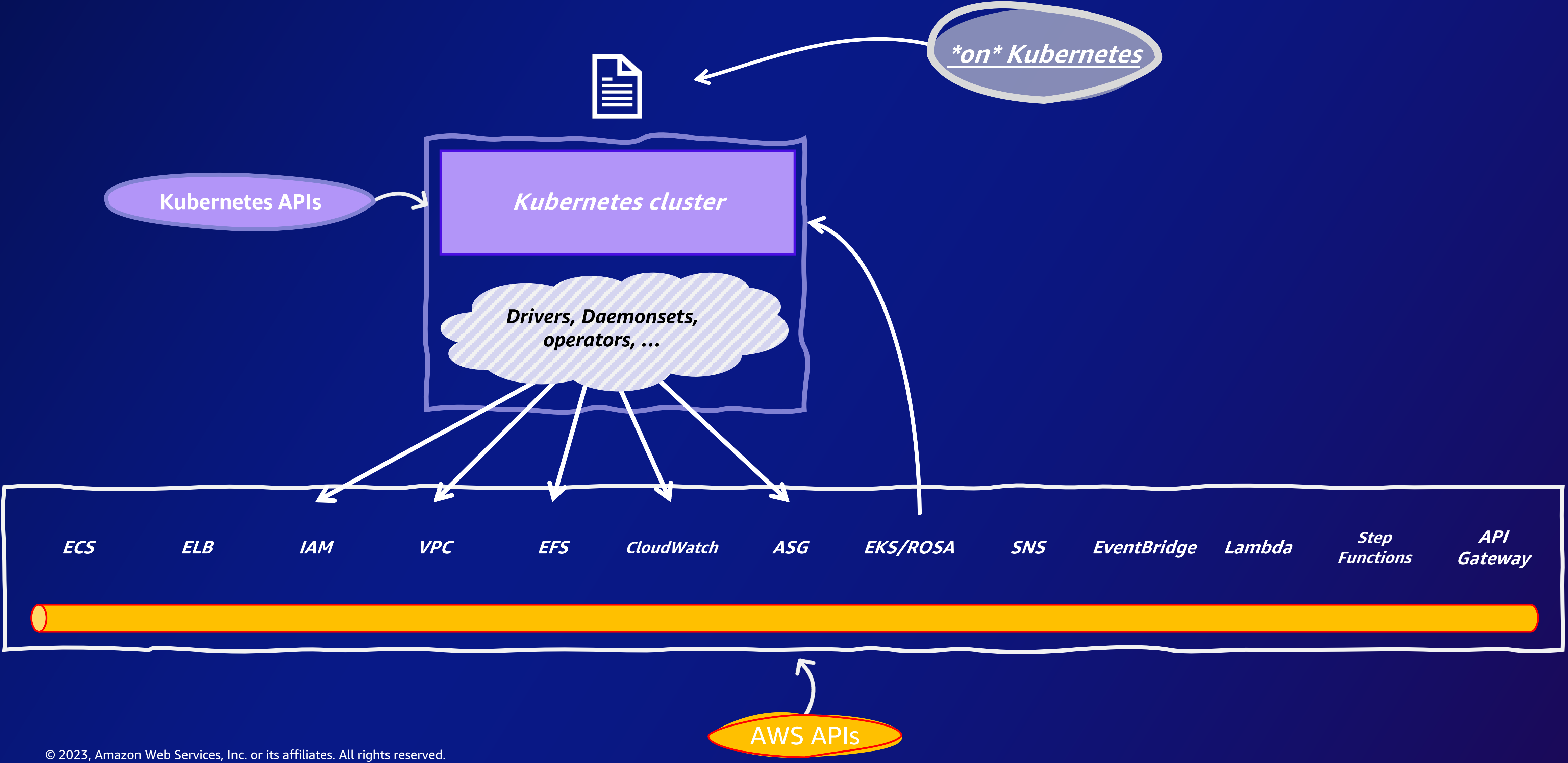
Optimized for portability and extensibility for process-based workloads



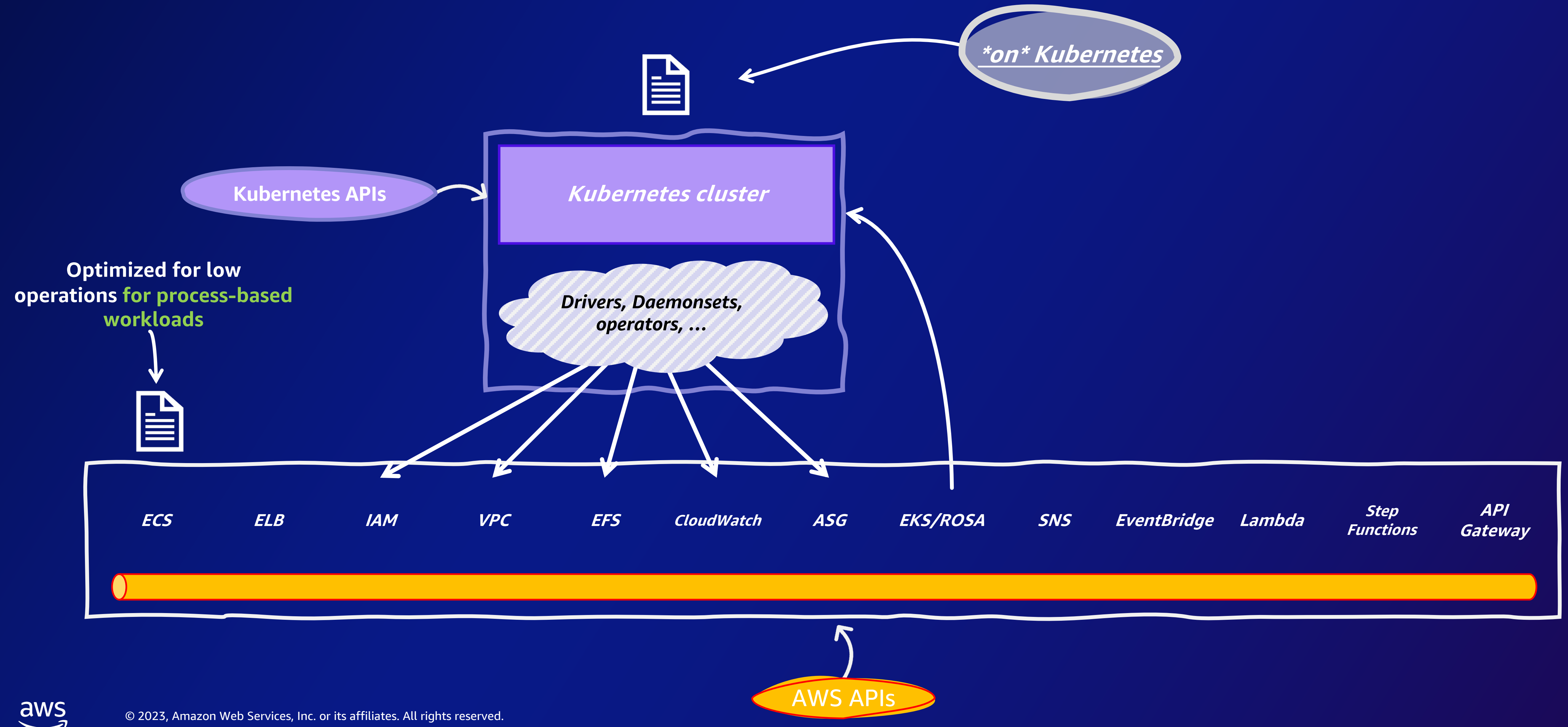
Let’s take it apart



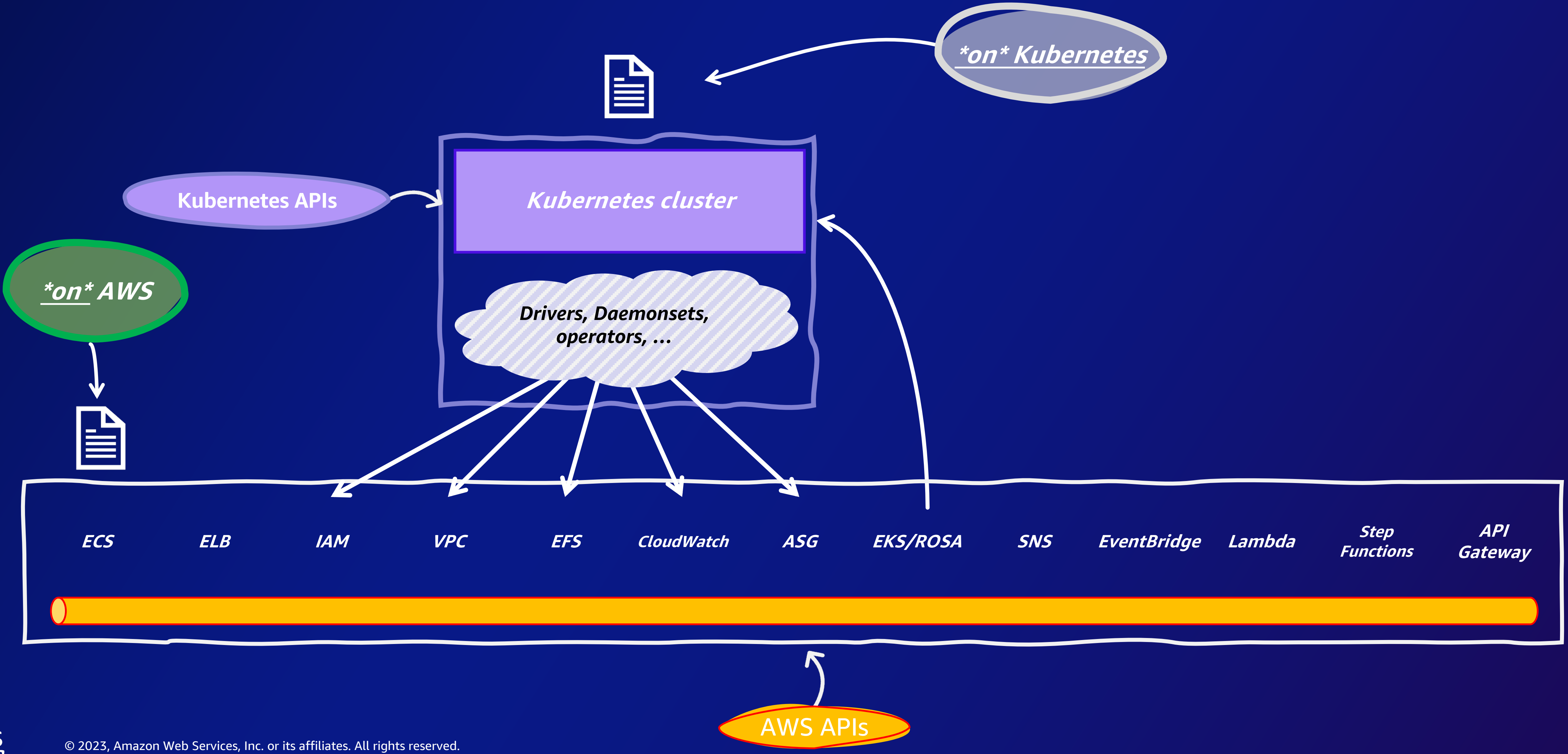
Let’s take it apart



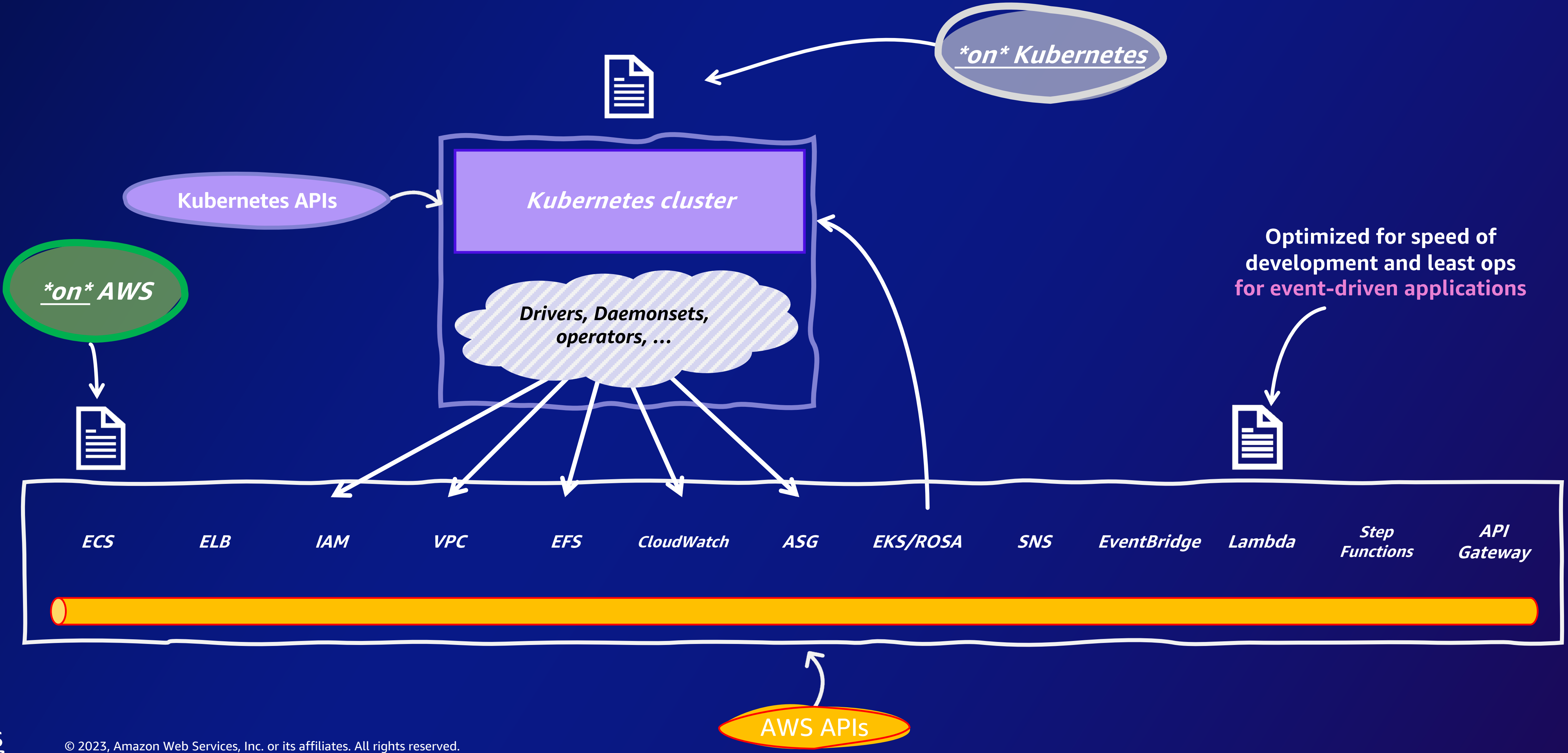
Let’s take it apart



Let’s take it apart

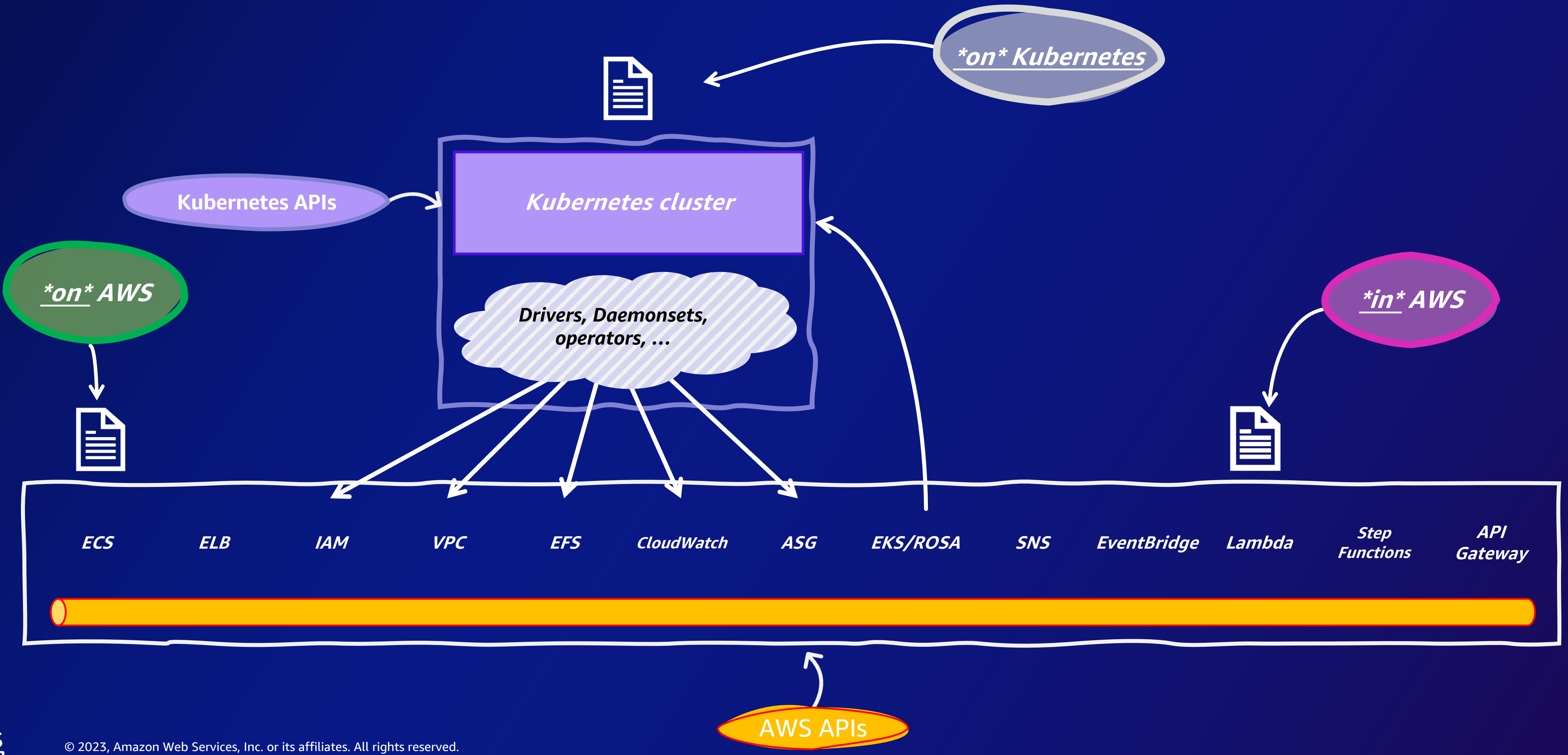


Let’s take it apart



Optimized for speed of development and least ops for event-driven applications

Let’s take it apart



Characteristics summary

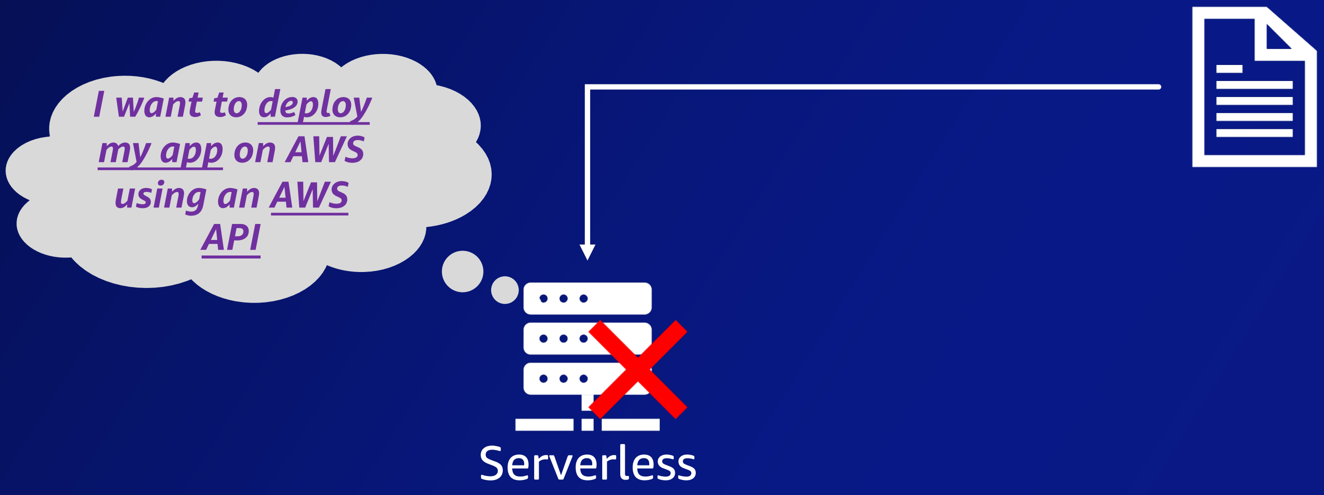
	On Kubernetes	On AWS	In AWS
Portability	*High (code & IaC)*	*Yes (code)*	*Low*
Extensibility	*High*	*Medium*	*Medium*
Operational efficiency	*Low*	*Medium*	*High*
Time to results/production	*Longer*	*Medium*	*Shorter*

"Which is the <u>right service</u> for my modern application?"

A better mental model

CODE!

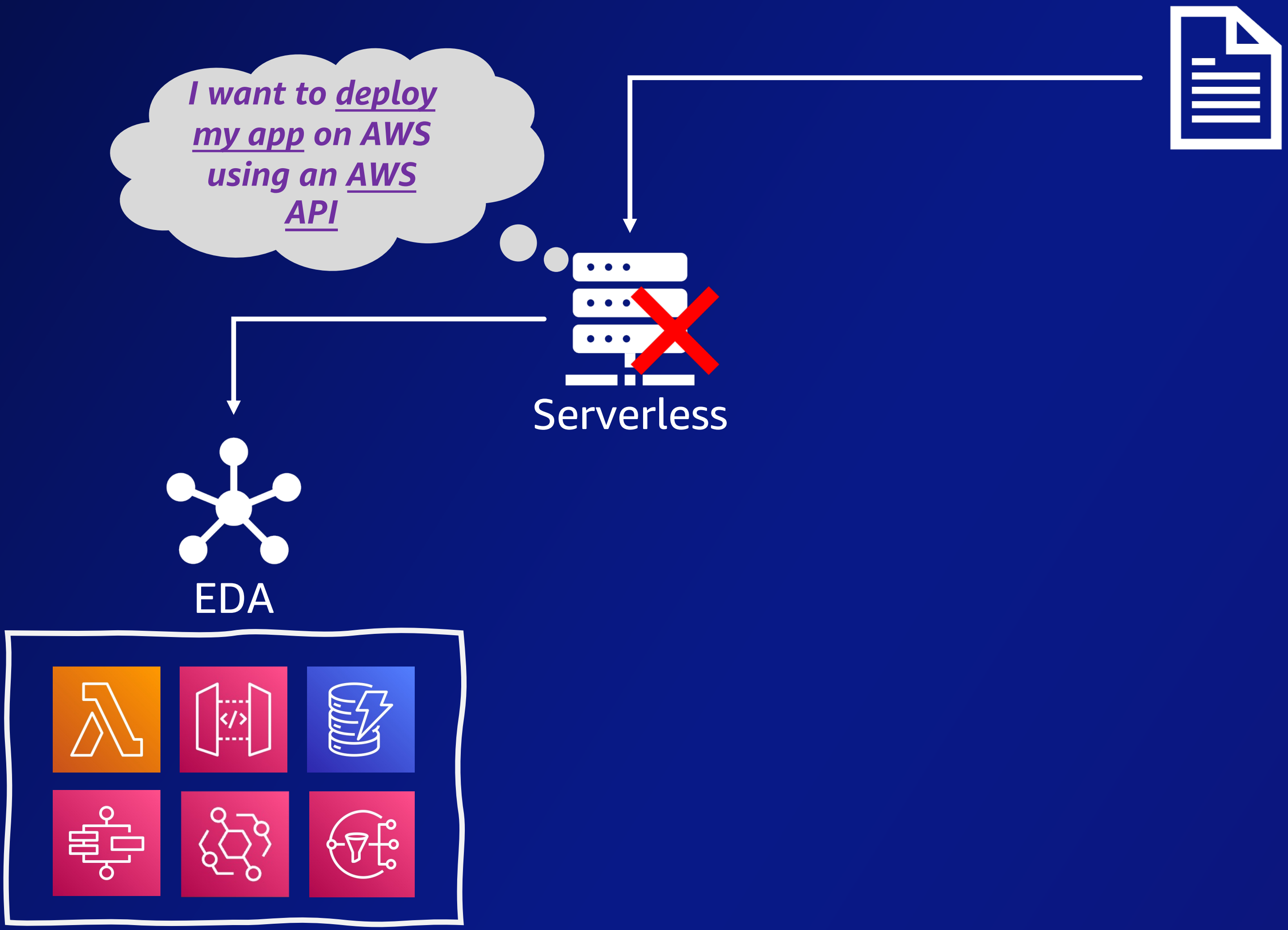
A better mental model



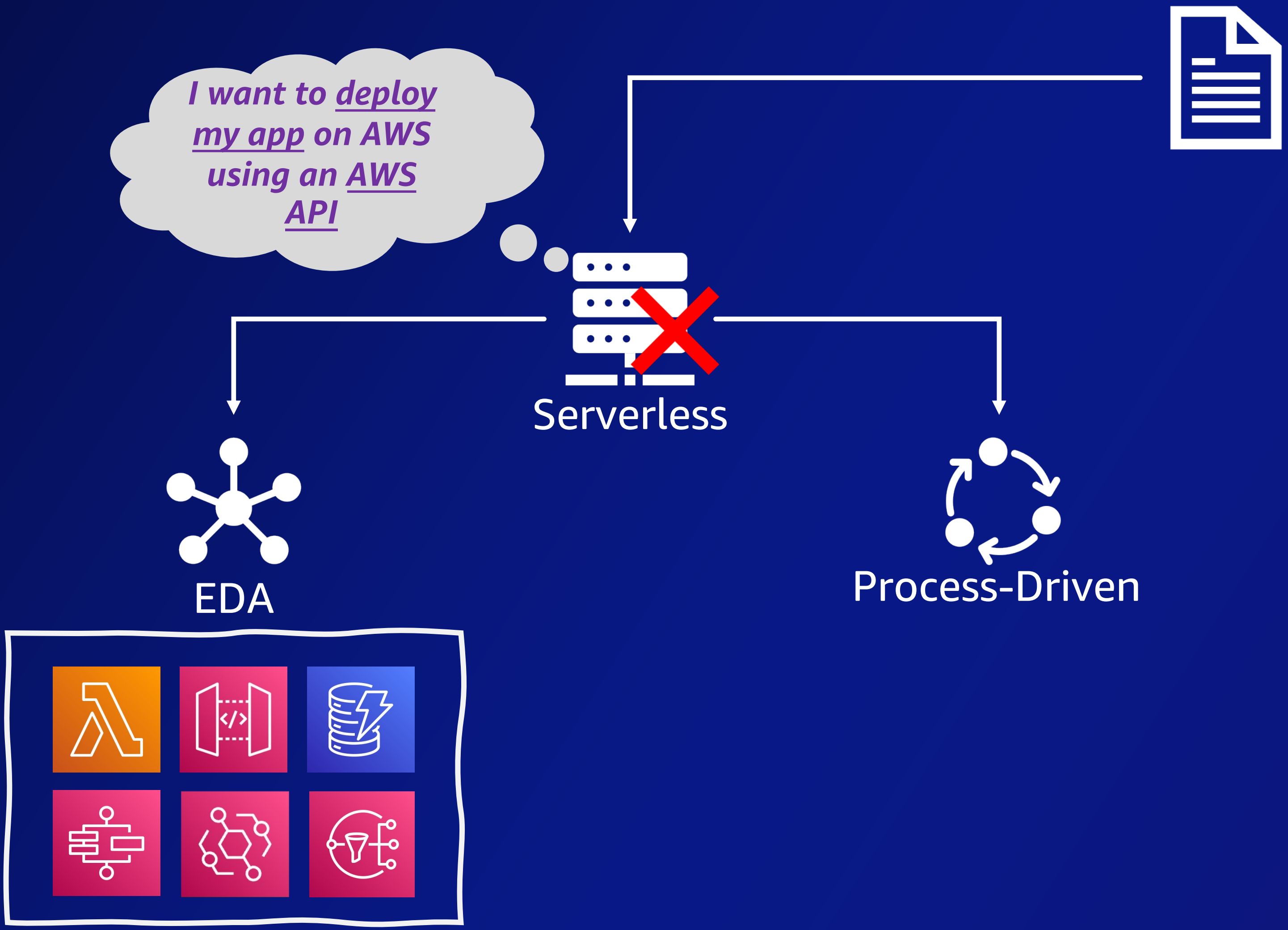
I want to deploy my app on AWS using an AWS API

Serverless

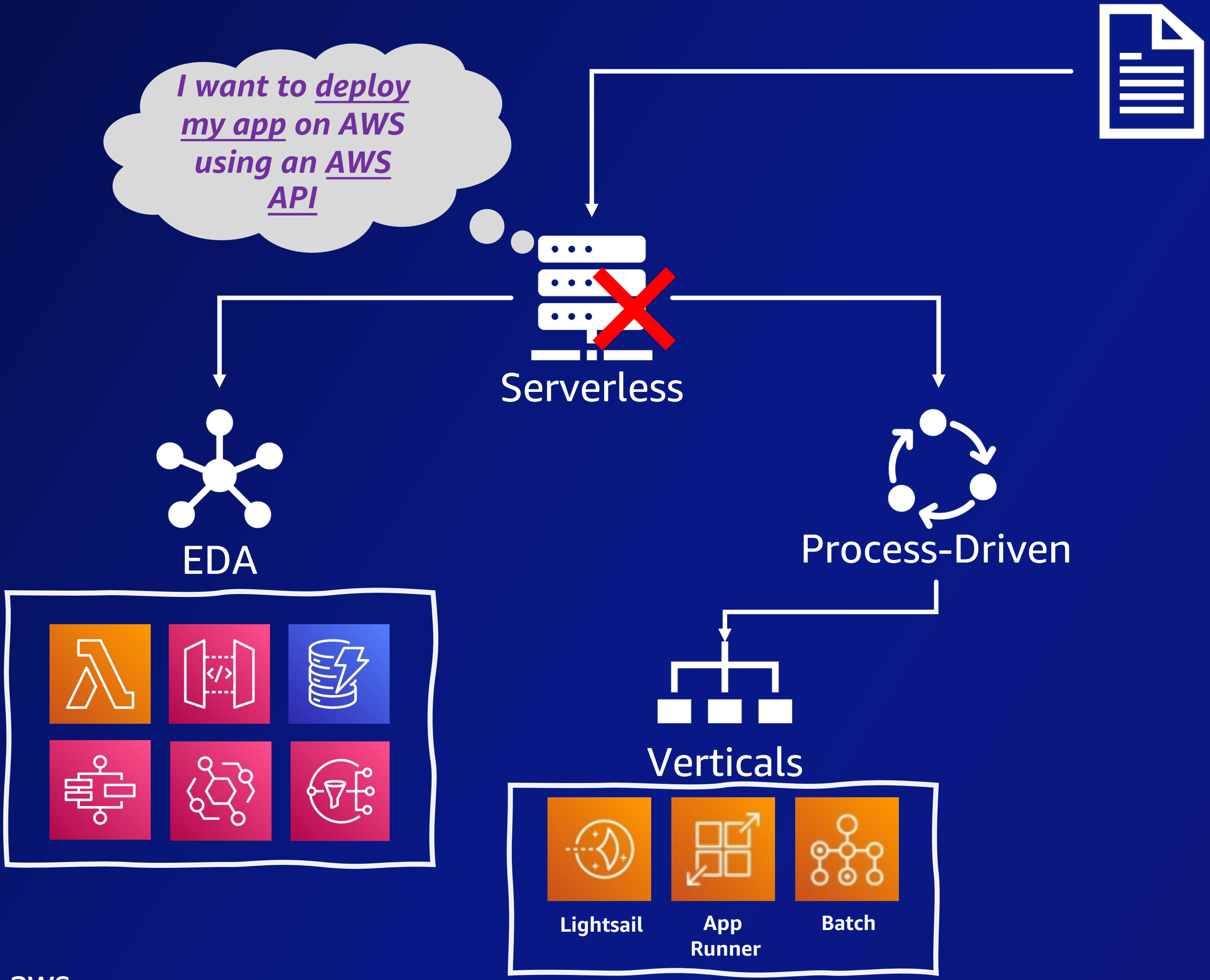
A better mental model



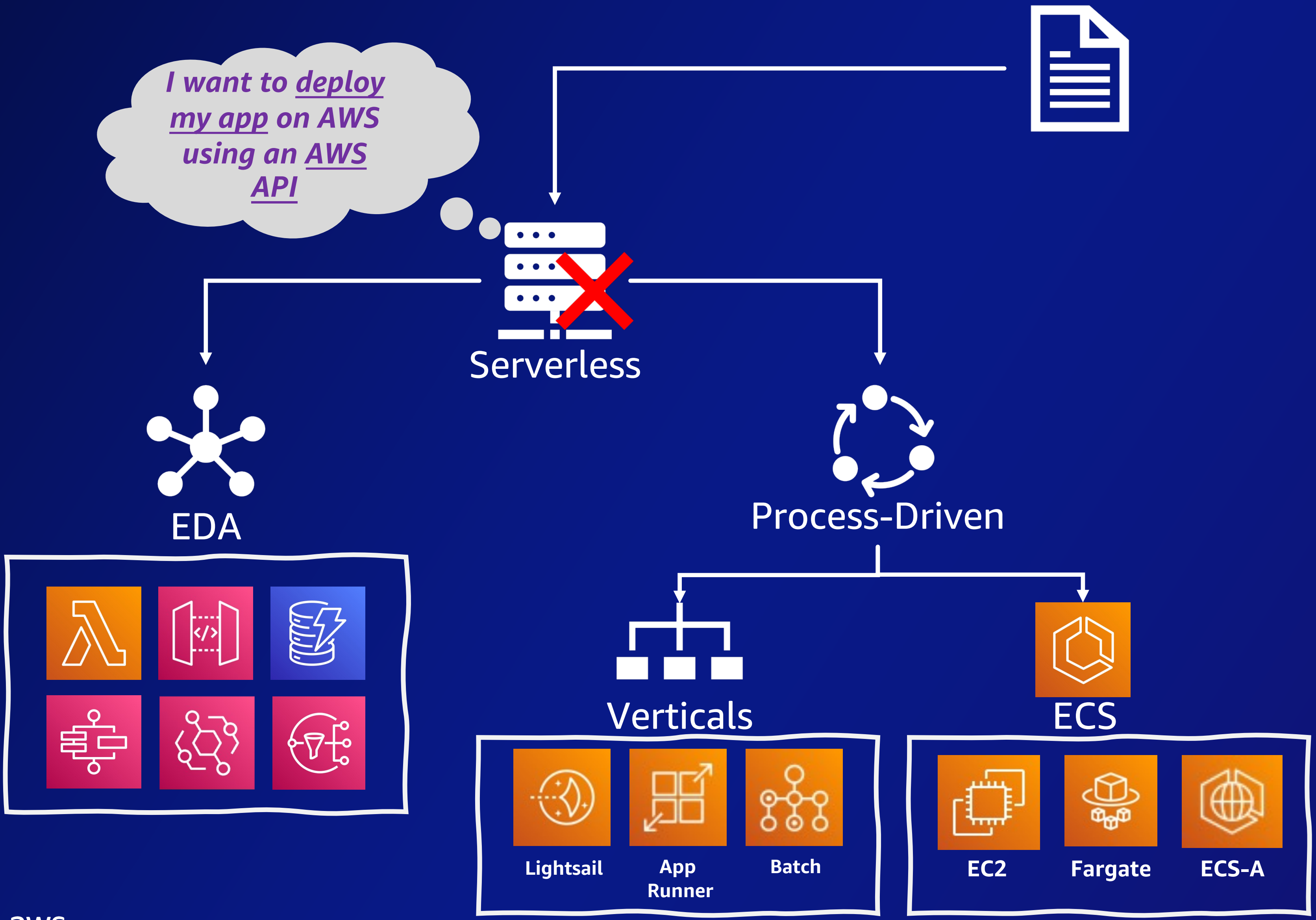
A better mental model



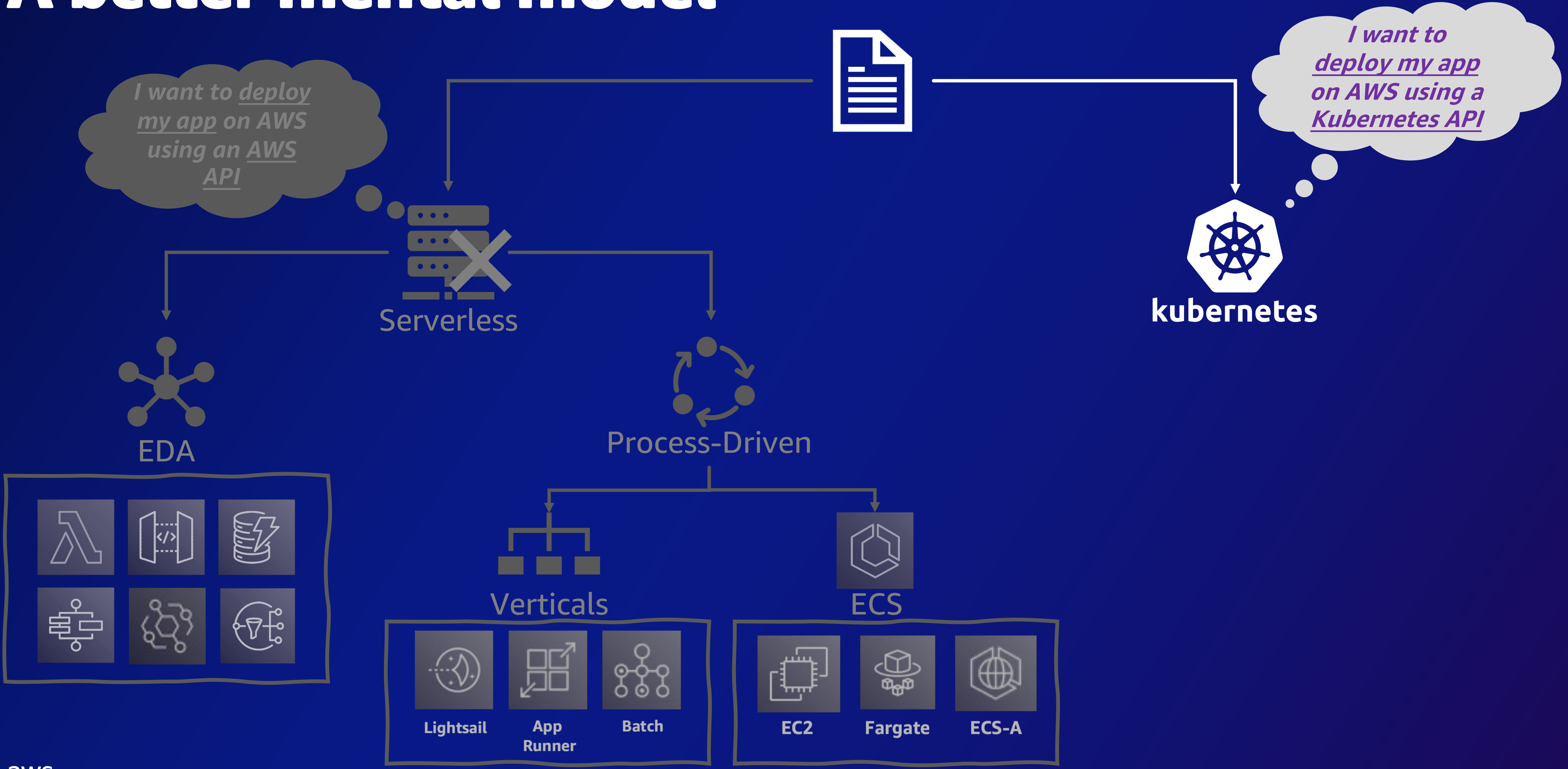
A better mental model



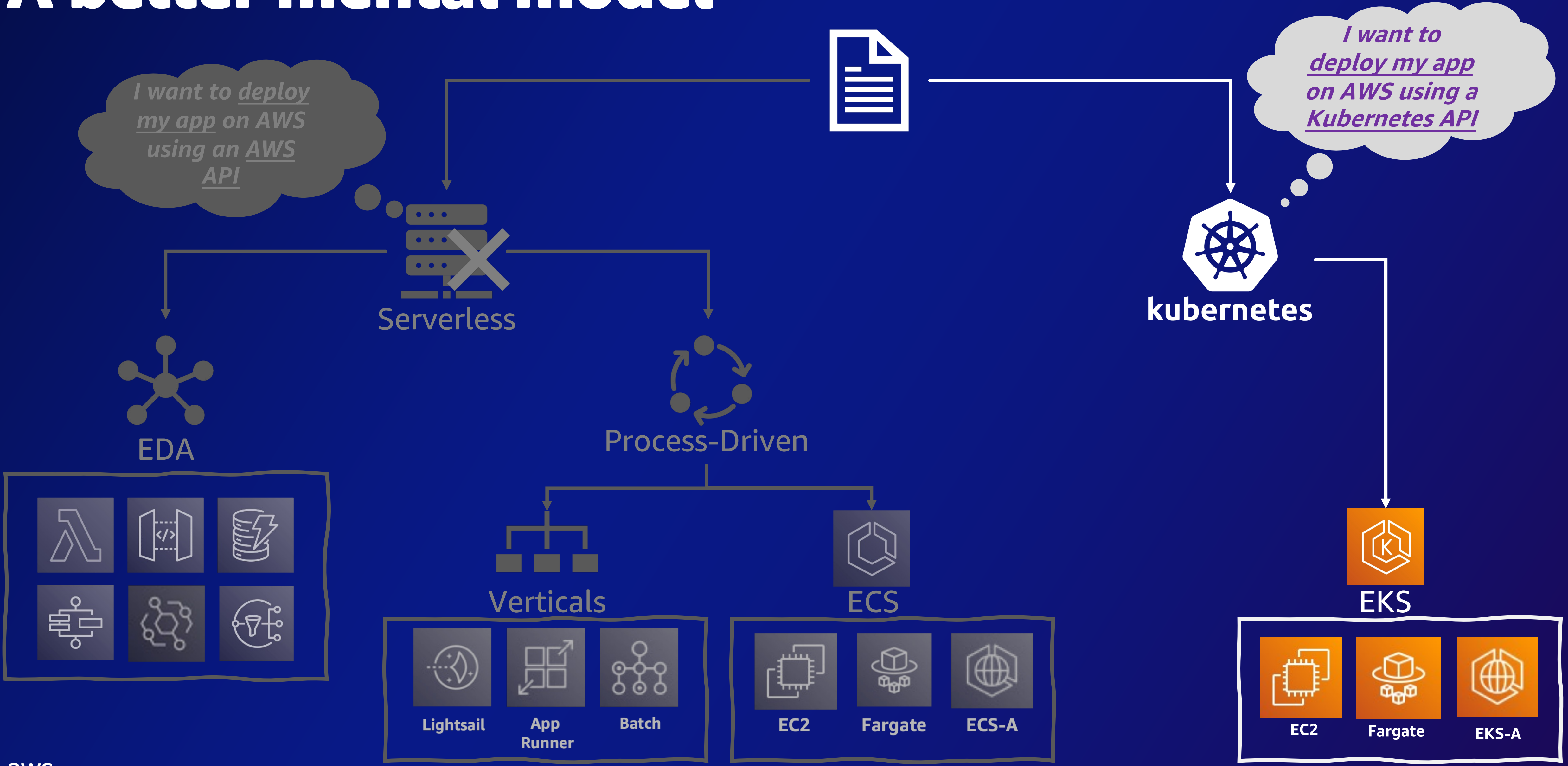
A better mental model



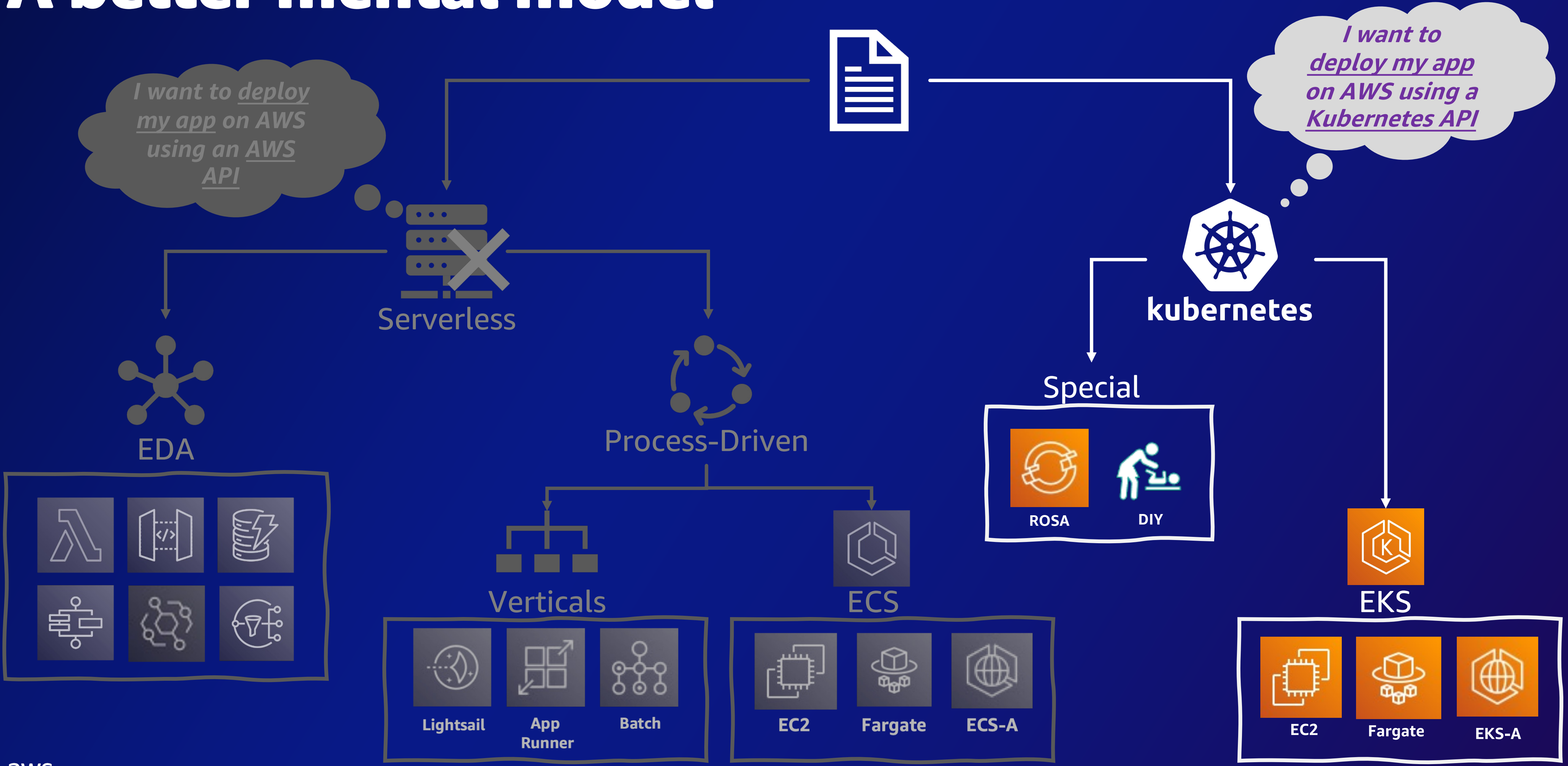
A better mental model


I want to deploy my app on AWS using an AWS API
Serverless
EDA
Process-Driven
Verticals
ECS
Lightsail
App Runner
Batch
EC2
Fargate
ECS-A
I want to deploy my app on AWS using a Kubernetes API
kubernetes

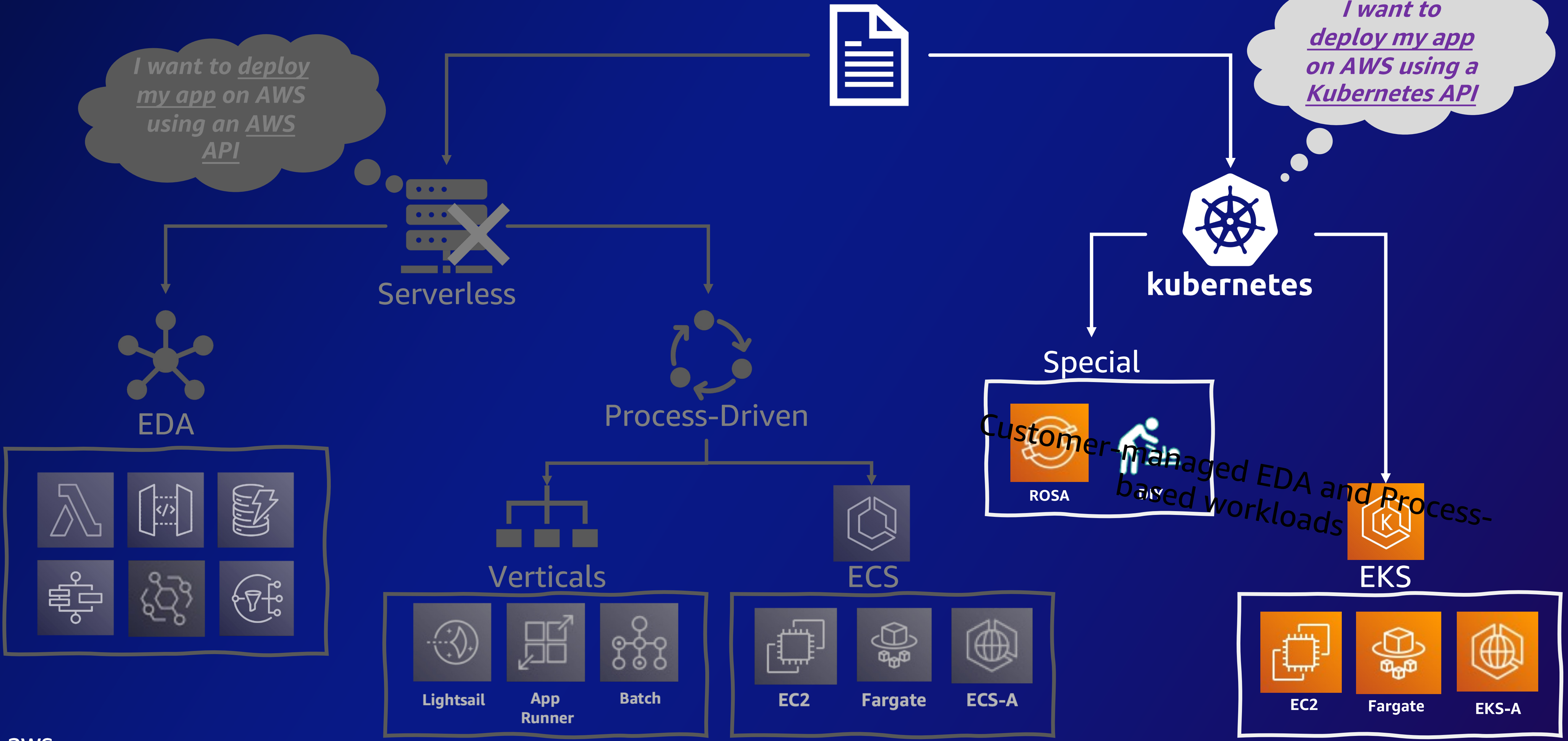
A better mental model



A better mental model



A better mental model



I want to deploy my app on AWS using an AWS API

Serverless

EDA

Process-Driven

Verticals

Lightsail

App Runner

Batch

ECS

EC2

Fargate

ECS-A

I want to deploy my app on AWS using a Kubernetes API

kubernetes

Special

ROSA

DIY

EKS

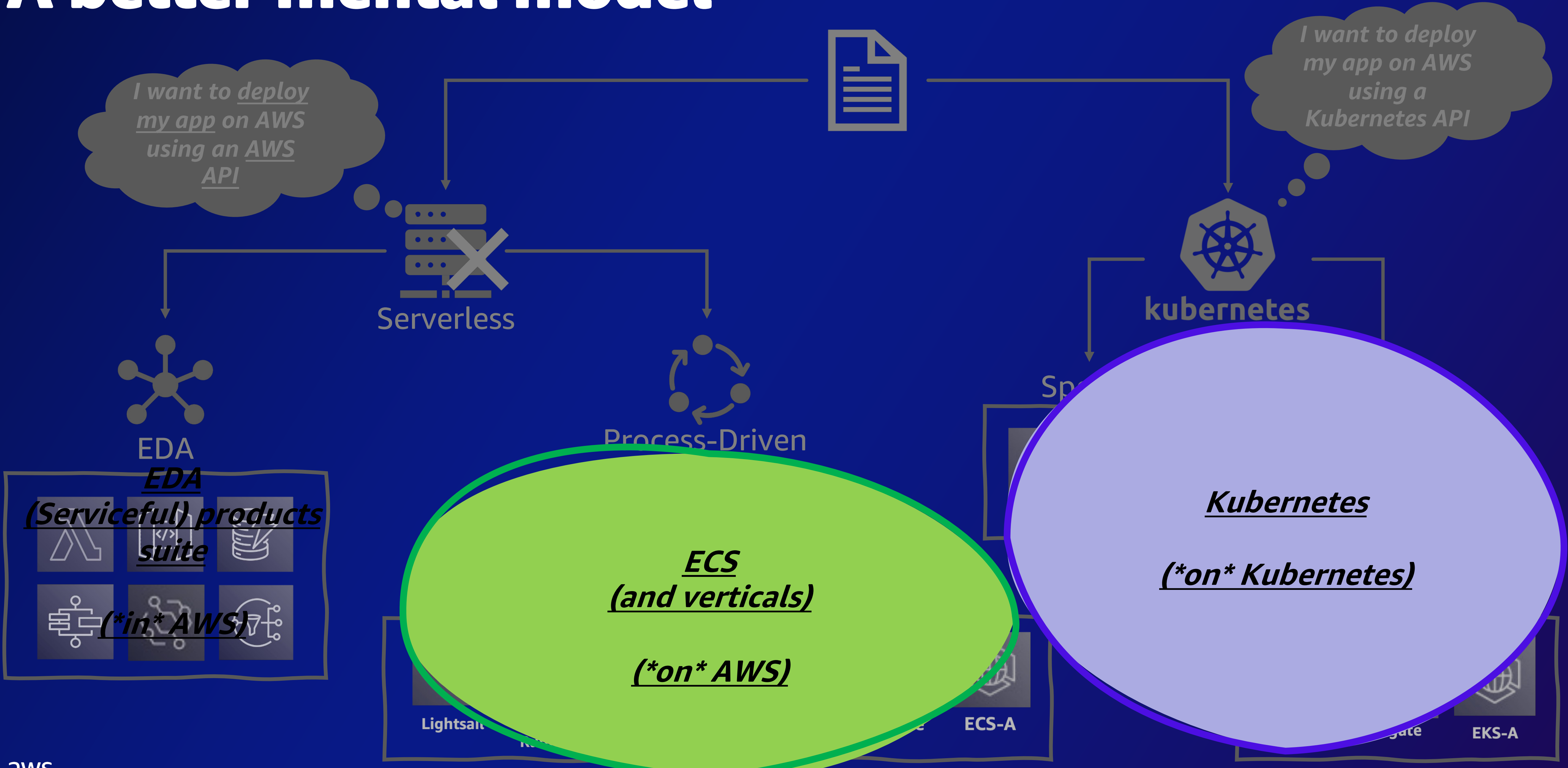
EC2

Fargate

EKS-A

Customer-managed EDA and Process-based workloads

A better mental model



I want to deploy my app on AWS using an AWS API

I want to deploy my app on AWS using a Kubernetes API

EDA (Serviceful) products suite

(*in* AWS)

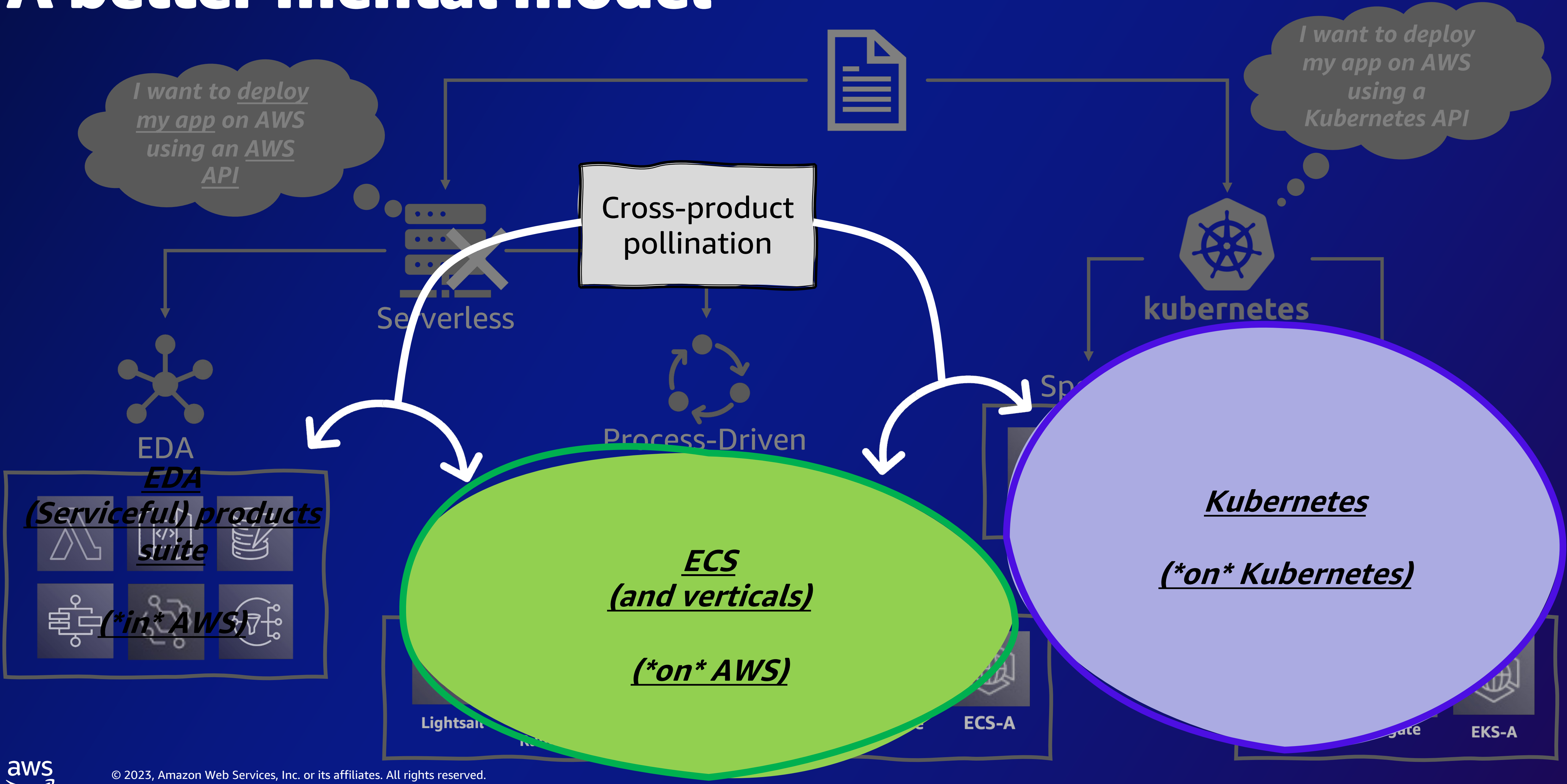
ECS (and verticals)

(*on* AWS)

Kubernetes

(*on* Kubernetes)

A better mental model



I want to deploy my app on AWS using an AWS API

I want to deploy my app on AWS using a Kubernetes API

Cross-product pollination

EDA
(Serviceful) products suite

(*in* AWS)

ECS
(and verticals)

(*on* AWS)

Kubernetes

(*on* Kubernetes)

"What is the right strategy when it comes to deciding between serverless and containers?"

7 step approach

Step 1: Understand and determine your criteria

Abstraction vs **Standardisation**

Architecture characteristics	*AWS Serverless* supports most architectures and patterns with specific services that provide optimizations for performance, scalability, reliability, and cost. *Kubernetes* supports most architectures where consistency across technology stack is preferred. Some levels of optimizations available, but will require more integration and management effort.

Step 1: Understand and determine your criteria

Flexibility vs **Consistency**

Workloads

AWS Serverless supports a range of workload patterns with specific services optimized for specific workloads.

Kubernetes supports a wide range of workload patterns where consistent deployment models across clouds or on-premises data centers is preferred.

Step 1: Understand and determine your criteria

Ease vs **Effort**

Prototyping

AWS Serverless is optimized for allowing customers to write code quickly, deploy it, and change it, making it a useful option for doing fast prototyping work

Kubernetes often requires a setup of special clusters dedicated for prototyping, which needs maintenance of their own.

Step 1: Understand and determine your criteria

Building Blocks vs **Ecosystem**

Integrations

AWS Serverless offers integrations with more than 200+ managed services.

Kubernetes provides a rich partner ecosystem (open source as well as enterprise) as well as out-of-the-box support for AWS services.

Step 1: Understand and determine your criteria

Code vs Infrastructure

Application Portability

AWS Serverless can easily port business logic from Lambda, App Mesh, or ECS to other compute environments..

Kubernetes patterns can at times rely on service mesh in code (like Istio, or programming models like KNative) means that your application requires Kubernetes, and it is not portable to something other than Kubernetes.

Step 2: Determine how much you want to manage

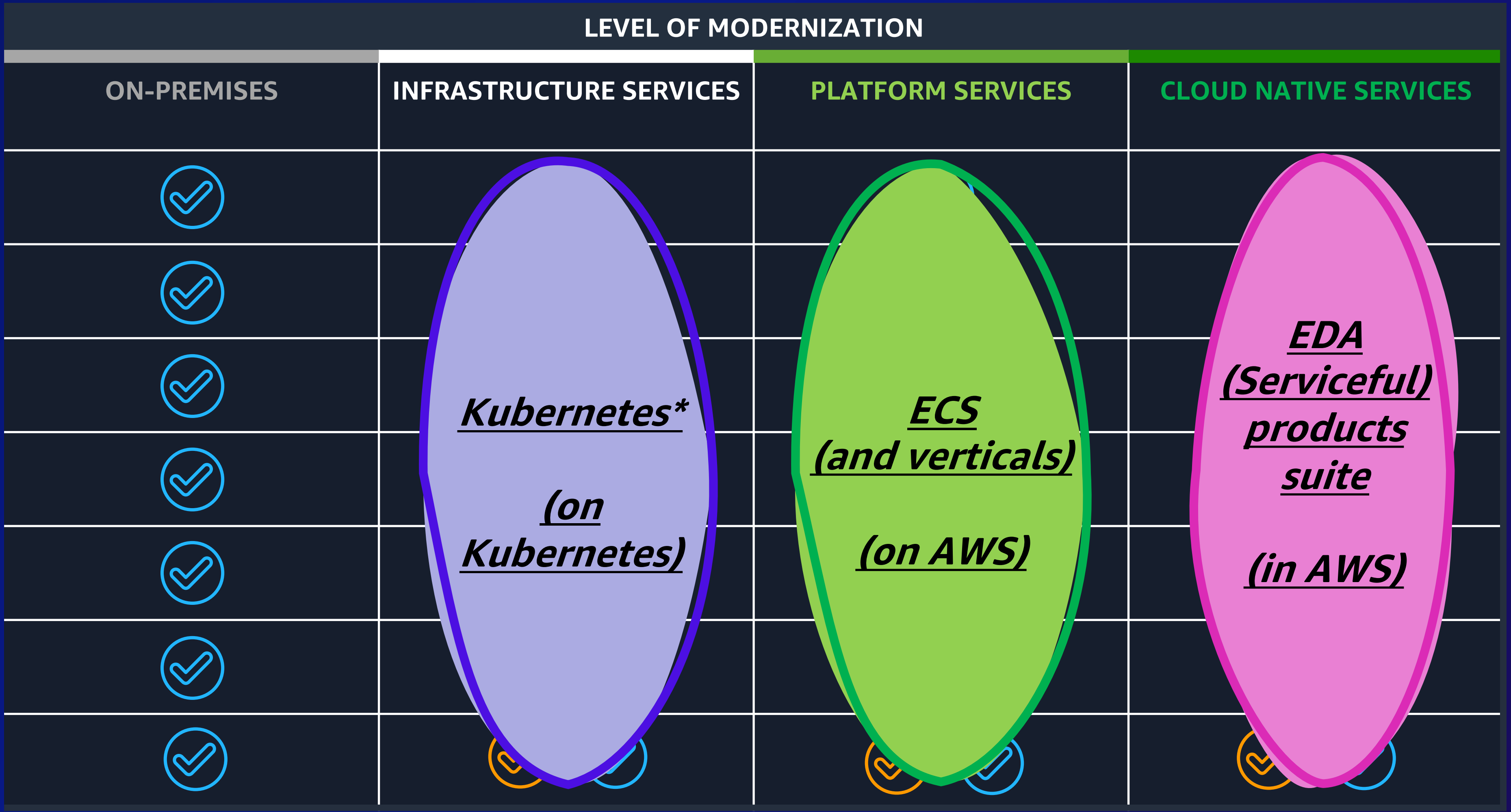
	LEVEL OF MODERNIZATION			
	ON-PREMISES	INFRASTRUCTURE SERVICES	PLATFORM SERVICES	CLOUD NATIVE SERVICES
Application code	🔵	🔵	🔵	🔵
Data source integrations	🔵	🔵	🔵	🟠
Capacity planning and scaling	🔵	🔵	🔵	🟠
Software install and maintenance	🔵	🔵	🟠	🟠
Infrastructure provisioning	🔵	🔵	🟠	🟠
Physical server, storage, networking, and facilities	🔵	🟠	🟠	🟠
Security and network configuration	🔵	🟠 🔵	🟠 🔵	🟠 🔵

MANAGED BY

🔵 CUSTOMER

🟠 AWS

Step 2: Determine how much you want to manage



	LEVEL OF MODERNIZATION			
	ON-PREMISES	INFRASTRUCTURE SERVICES	PLATFORM SERVICES	CLOUD NATIVE SERVICES
Application code	✓			
Data source integrations	✓			
Capacity planning and scaling	✓			
Software install and maintenance	✓			
Infrastructure provisioning	✓			
Physical server, storage, networking, and facilities	✓			
Security and network configuration	✓	✓ ✓	✓ ✓	✓ ✓

Kubernetes*

(on Kubernetes)

ECS (and verticals)

(on AWS)

EDA (Serviceful) products suite

(in AWS)

*T&C apply

MANAGED BY ✓ CUSTOMER ✓ AWS

Step 3: Determine your use case

Legacy App Modernization

- Java and .NET Apps
- Legacy homegrown Linux Apps
- Monoliths

AI/ML

- Autonomous Vehicles (Object Tracking, Sensor Fusion)
- Robotics (Vision, Grasping, Motion Control)
- Modeling, Training, and Inference

Data Processing

- Real time
- MapReduce
- Batch

Backends

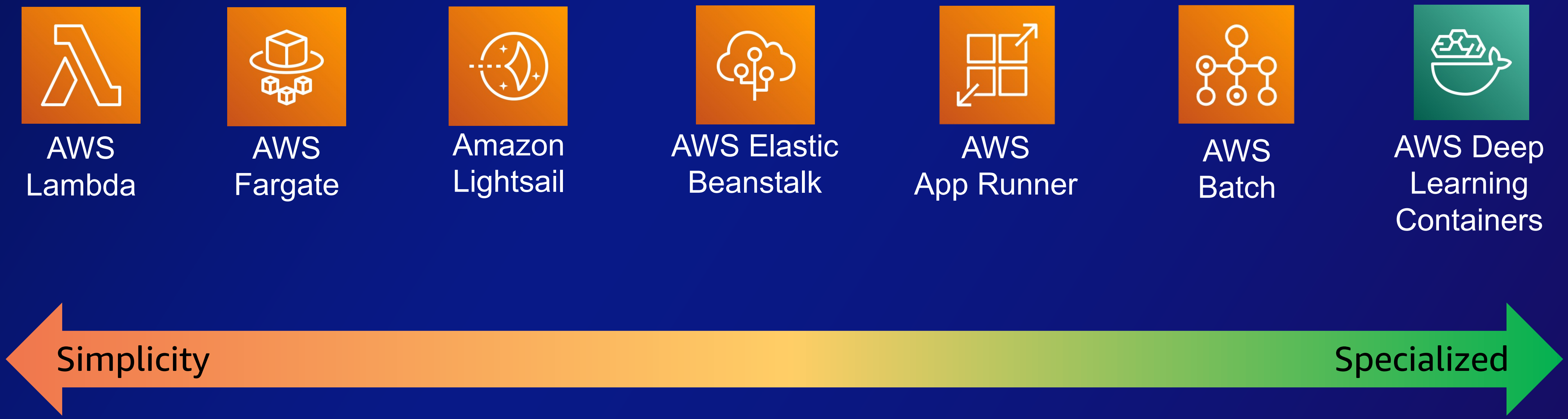
- Apps & services
- Mobile
- IoT
- Operations

Web Applications

- Static websites
- Complex web apps
- EDA

Most organizations will support multiple options or workload patterns to allow for workload or developer choice.

Step 4: Compare and make the right choices for your workloads



AWS Lambda

AWS Fargate

Amazon Lightsail

AWS Elastic Beanstalk

AWS App Runner

AWS Batch

AWS Deep Learning Containers

Simplicity

Specialized

Step 5: Avoid common pitfalls

Understand standardization in your environment

Understand your architecture characteristics

Understand people, and processes

Step 6: Decide your approach

AWS Serverless with AWS

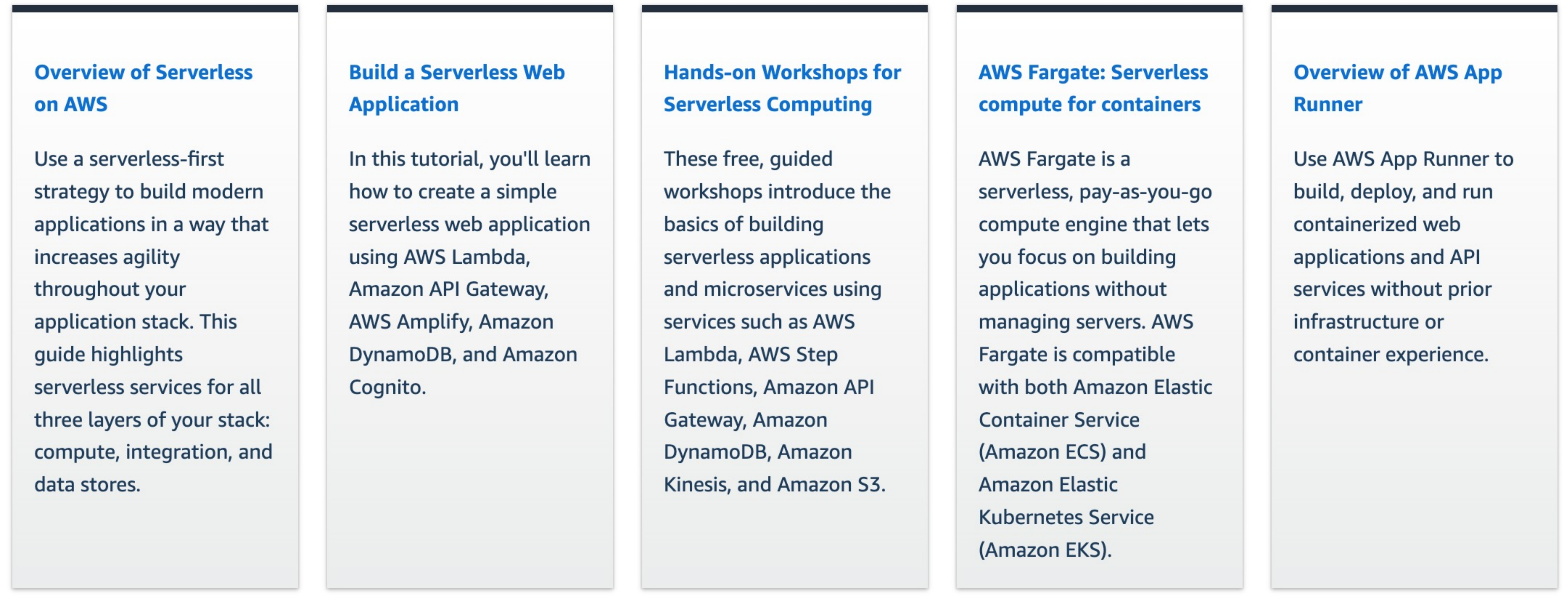
- Use AWS managed services and tools as your first choice (Amazon ECS, AWS Lambda, AWS Fargate, etc.).
- Invest in developing discipline around AWS including provisioning, DevOps, infrastructure automation, security, networking, and observability/operations.
- Increase productivity and minimize operational burden.

Kubernetes on AWS

- Use Kubernetes as your primary compute platform interface (Amazon EKS, ROSA).
- Adopt discipline around running and managing several Kubernetes clusters and the workload and tools on them, advanced patterns like GitOps.
- Integrate with different ecosystems and partner tools.

Step 7: Implement your approach

AWS Serverless with AWS



Overview of Serverless on AWS

Use a serverless-first strategy to build modern applications in a way that increases agility throughout your application stack. This guide highlights serverless services for all three layers of your stack: compute, integration, and data stores.

Build a Serverless Web Application

In this tutorial, you'll learn how to create a simple serverless web application using AWS Lambda, Amazon API Gateway, AWS Amplify, Amazon DynamoDB, and Amazon Cognito.

Hands-on Workshops for Serverless Computing

These free, guided workshops introduce the basics of building serverless applications and microservices using services such as AWS Lambda, AWS Step Functions, Amazon API Gateway, Amazon DynamoDB, Amazon Kinesis, and Amazon S3.

AWS Fargate: Serverless compute for containers

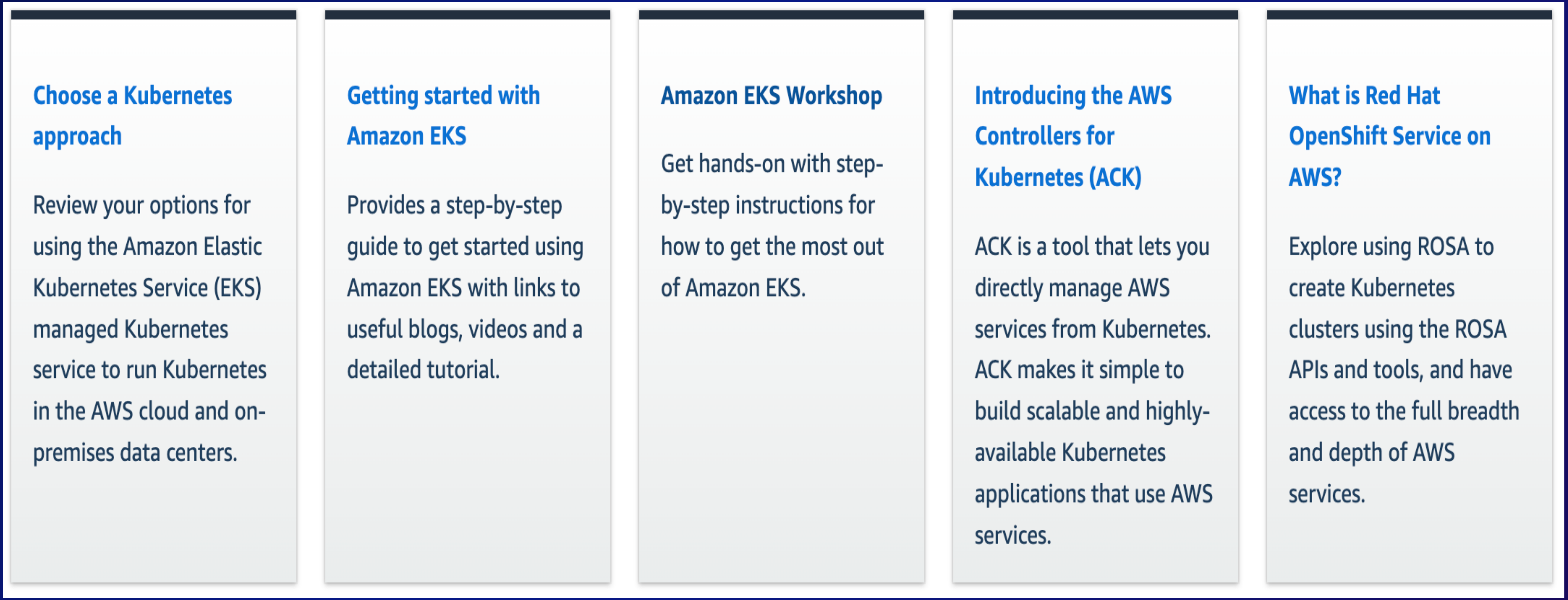
AWS Fargate is a serverless, pay-as-you-go compute engine that lets you focus on building applications without managing servers. AWS Fargate is compatible with both Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS).

Overview of AWS App Runner

Use AWS App Runner to build, deploy, and run containerized web applications and API services without prior infrastructure or container experience.

Step 7: Implement your approach

Kubernetes on AWS



Choose a Kubernetes approach

Review your options for using the Amazon Elastic Kubernetes Service (EKS) managed Kubernetes service to run Kubernetes in the AWS cloud and on-premises data centers.

Getting started with Amazon EKS

Provides a step-by-step guide to get started using Amazon EKS with links to useful blogs, videos and a detailed tutorial.

Amazon EKS Workshop

Get hands-on with step-by-step instructions for how to get the most out of Amazon EKS.

Introducing the AWS Controllers for Kubernetes (ACK)

ACK is a tool that lets you directly manage AWS services from Kubernetes. ACK makes it simple to build scalable and highly-available Kubernetes applications that use AWS services.

What is Red Hat OpenShift Service on AWS?

Explore using ROSA to create Kubernetes clusters using the ROSA APIs and tools, and have access to the full breadth and depth of AWS services.

Conclusion

- Two “families” and three “models” to build/run apps on AWS
- It’s no longer Containers or Serverless
- Containers are becoming a ubiquitous code packaging mechanism
- It’s Serverless (ECS++ and EDA suite) or Kubernetes
- Yes, on Kubernetes **you** can run/manage all types of workloads

Interested in knowing more?

Serverless or Kubernetes on AWS

AWS re:Invent 2022 - Competition of the modern workloads: Serverless vs Kubernetes on AWS (COM207-R)

Thank you!

Please complete the session survey in the mobile app

Yohan Wadia

@yohanwadia88

linkedin.com/in/yohanwadia88