CS443 Cloud Computing Week 3

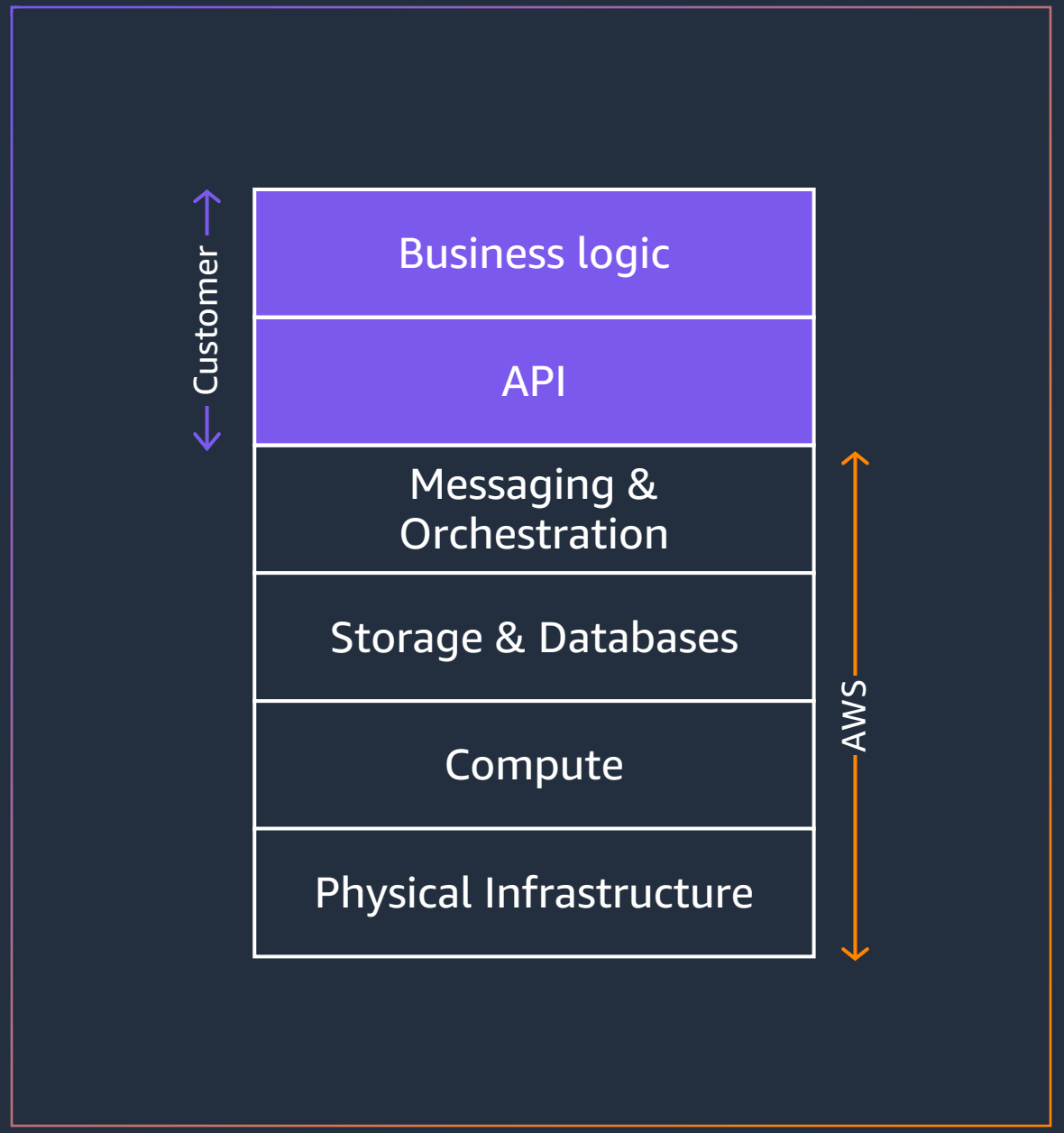
Serverless on AWS

Eren Akbaba

Solutions Architect, AWS

What is Serverless?

Serverless services simplify the management and scaling of cloud applications by shifting undifferentiated operational tasks to the cloud provider so **development teams can focus on writing code** that solve business problems

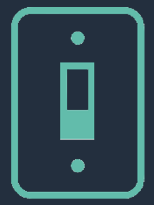


Why serverless

TAKE FULL ADVANTAGE OF THE CLOUD TO MODERNIZE APPLICATIONS AND ACCELERATE INNOVATION

No infrastructure provisioning, no management

Automatic scaling

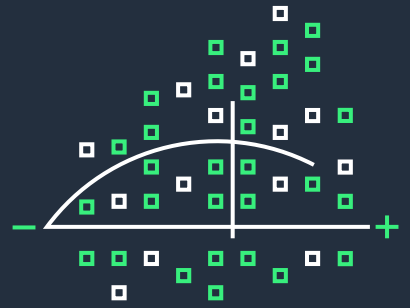


Pay for value

Highly available and secure

What are customers building with Serverless?

IT Automation



Data processing

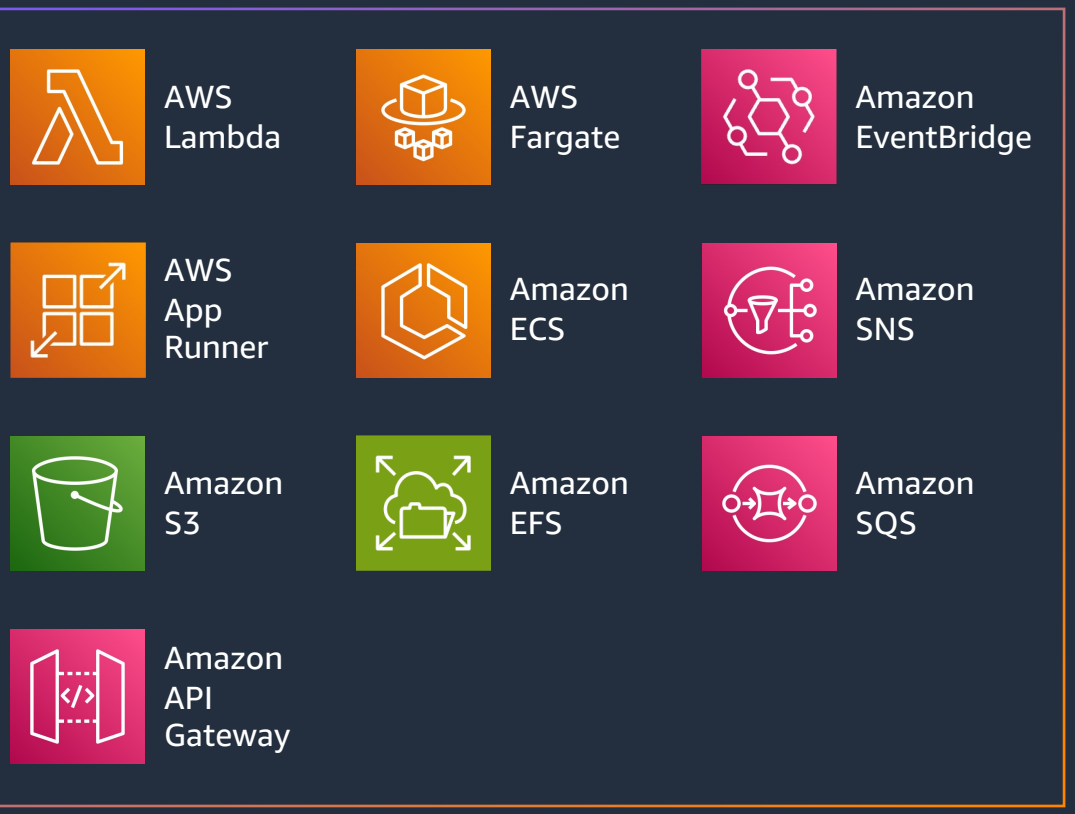
Web applications

Machine Learning

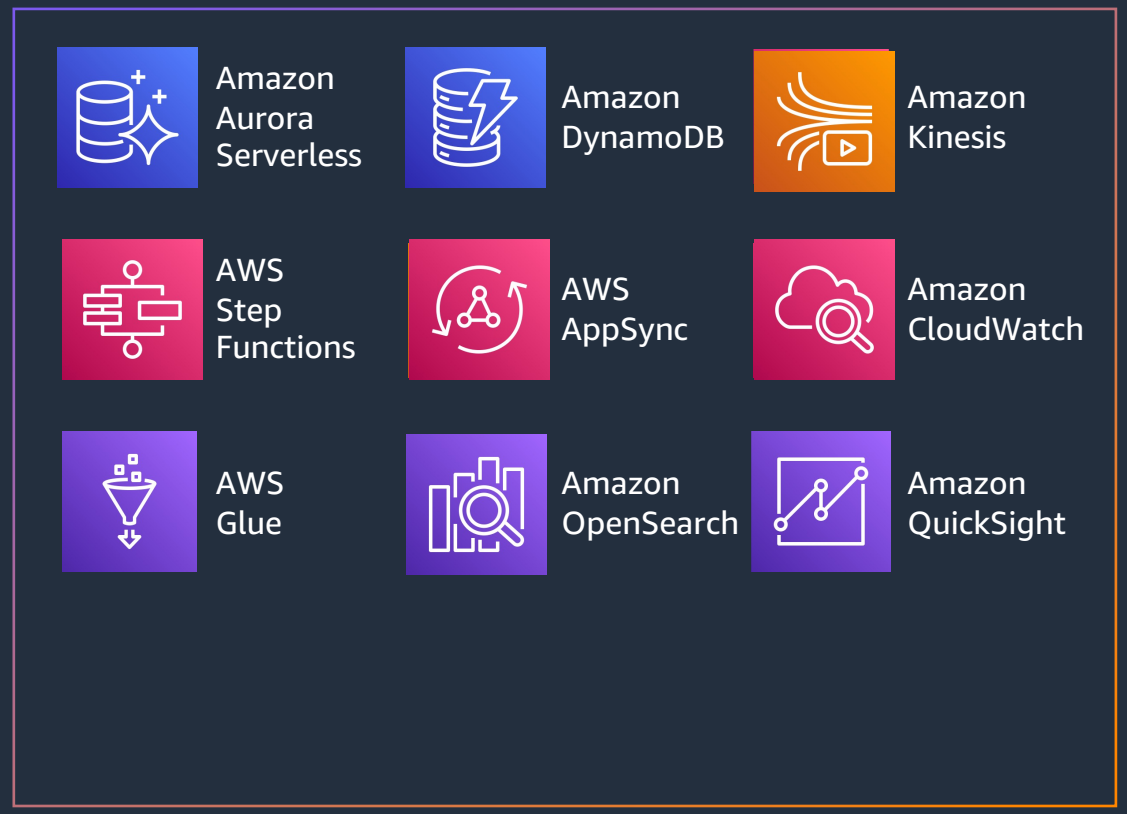
AWS serverless spectrum

AWS OFFERS THE WIDEST PORTFOLIO OF SERVERLESS SERVICES FOR RUNNING AND BUILDING MODERN APPS

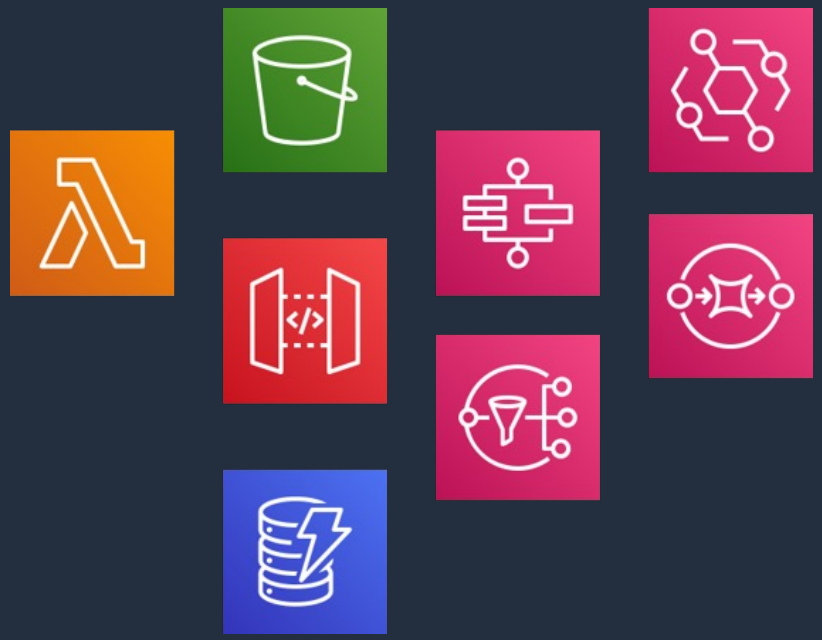
From Primitives to...
(Examples: compute, containers, buses)

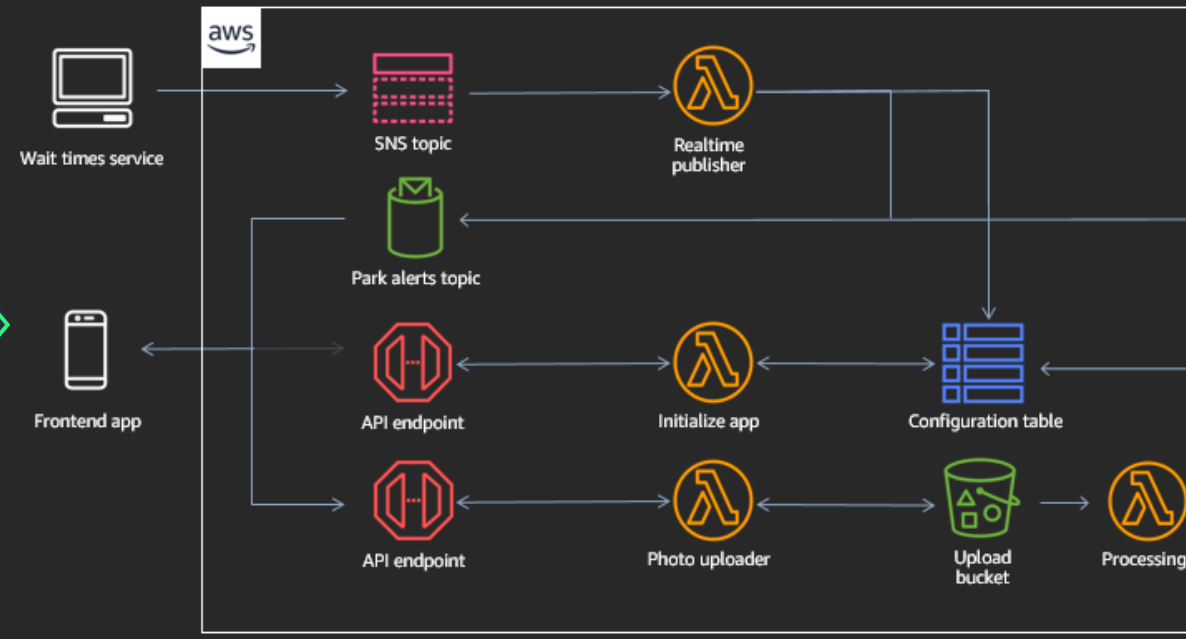


...Peripherals
(Examples: databases, analytics, workflows)



Small pieces, loosely joined





AWS Lambda

Event-driven function-as-a-service

Serverless Architecture

Event Source

Changes in data state

Requests to endpoints

Changes in resource state

Function

Node.js
Python
Java
C#
Go
Ruby
Bring Your Own

Services / Other

Anatomy of a Lambda Function

Handler function

- Function executed on invocation
- Processes incoming event

Event

- Invocation data sent to function
- Shape differs by event source

Context

- Additional information from Lambda service
- Examples: request ID, time remaining

app.py

```
def handler(event, context):
  msg = 'Hello {}'.format(
                    event['name']
                   )
  return { 'message': msg }
```

Lambda Function Configuration

Power rating

- Select between 128MB and 10GB
- CPU and network allocated proportionally
- Power tune to balance cost and speed

Permissions model

- Execution role grants function access to resources via IAM
- Resource-based policy controls invocation from other AWS accounts or organizations and AWS services

Lambda Function Configuration

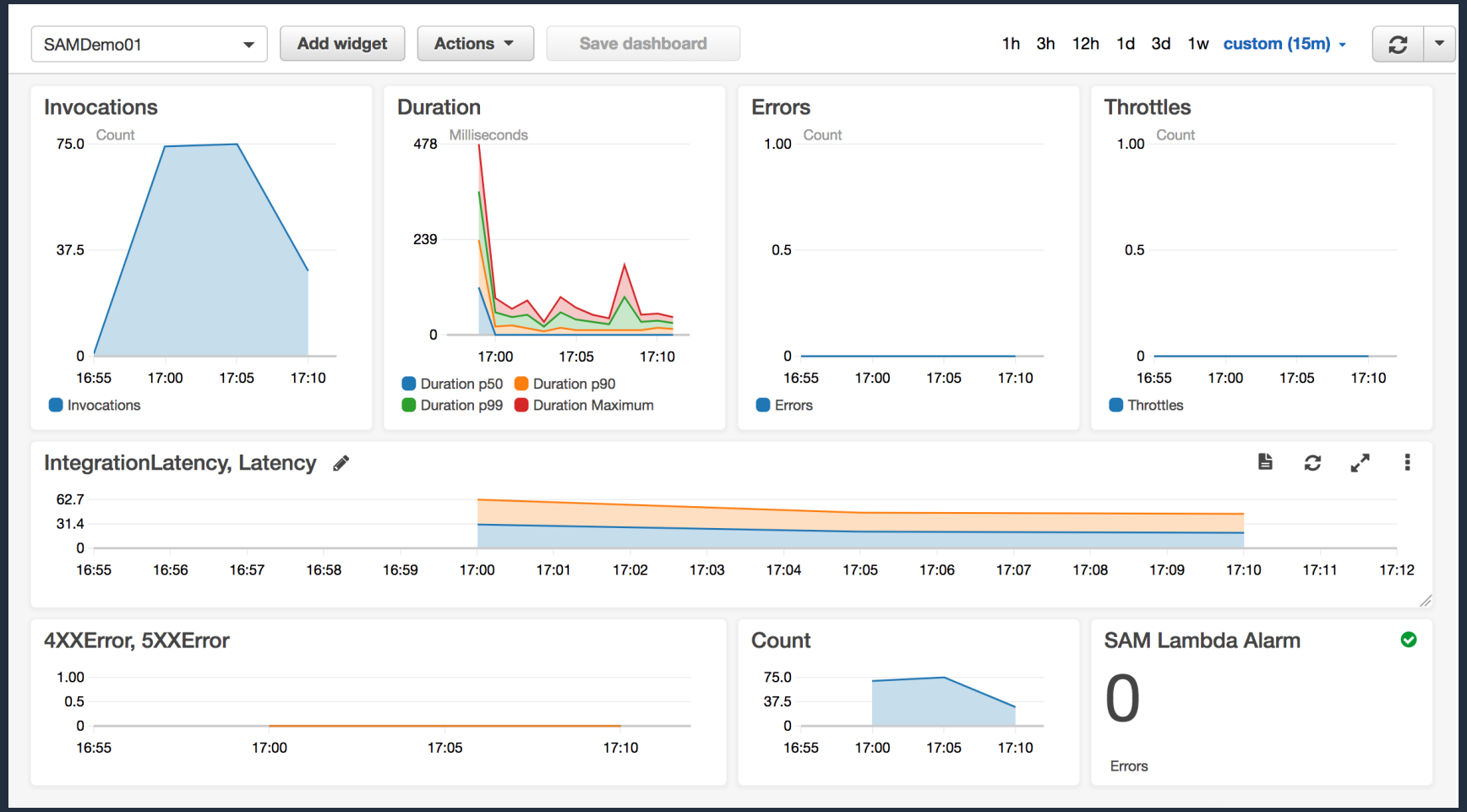
Timeout

- Up to 15 minutes
- Synchronous vs Asynchronous
  - API Gateway timeout = 30 sec

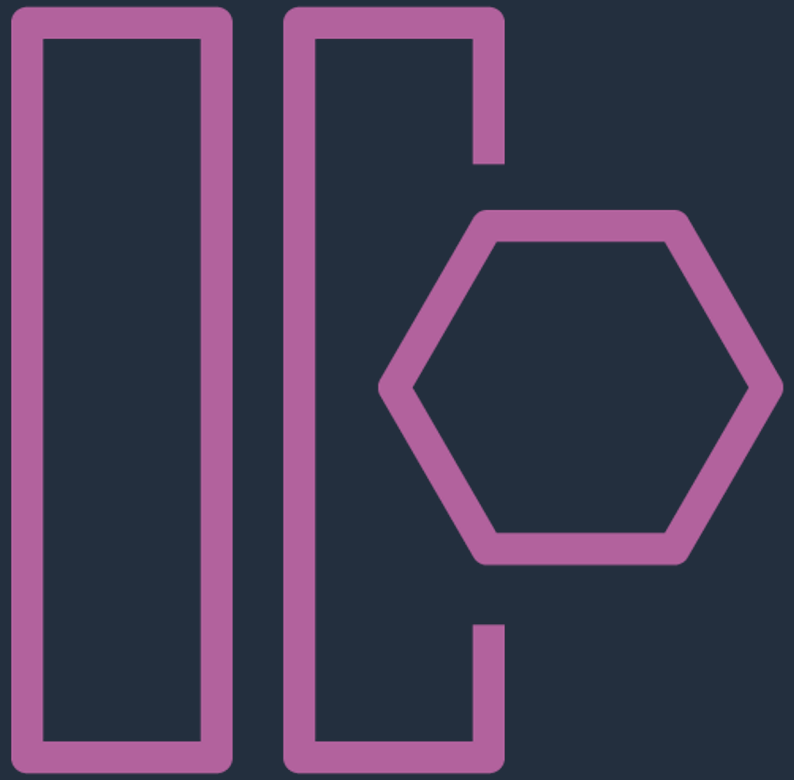
Network access

- Configure access to VPC
- Security Group rules apply
- VPC does not enhance security of function

Built in monitoring



Serverless architecture patterns



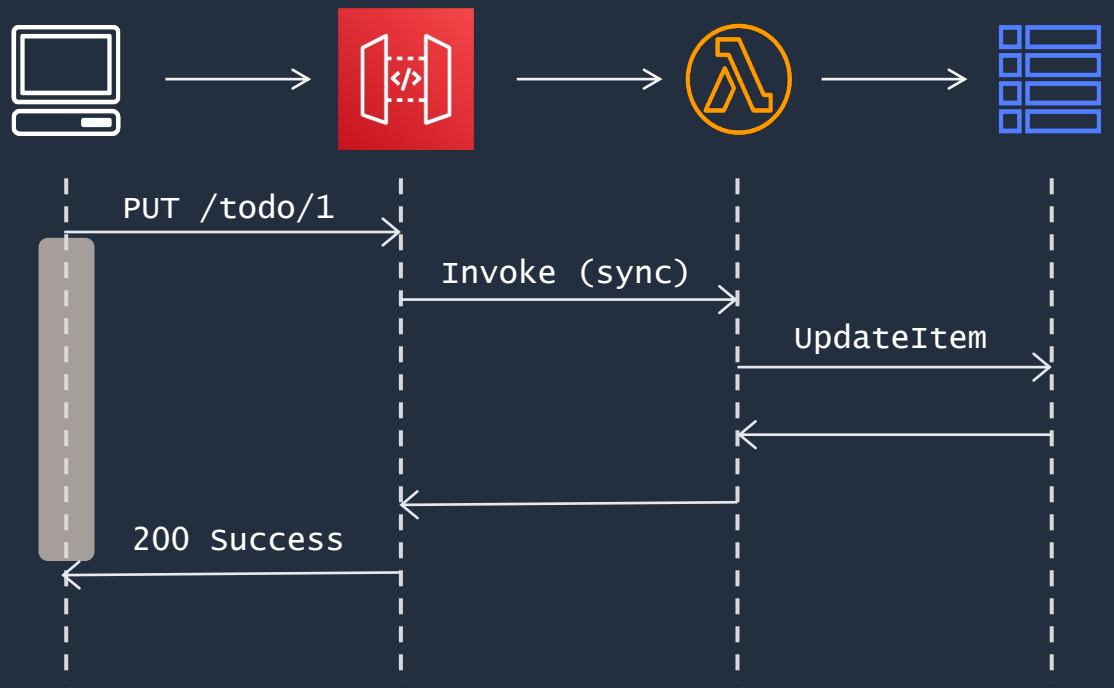
event

[i-’vent] **noun**

A thing that happens, particularly one of importance.

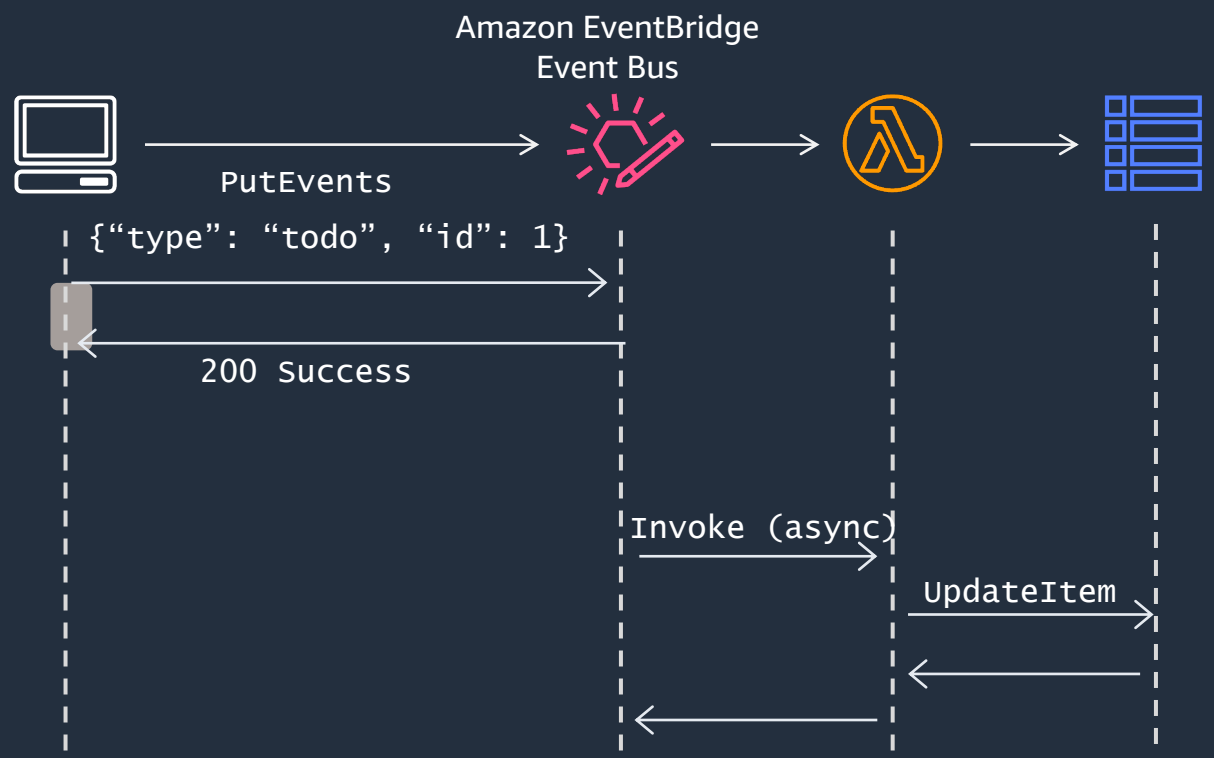
API-Driven Architectures

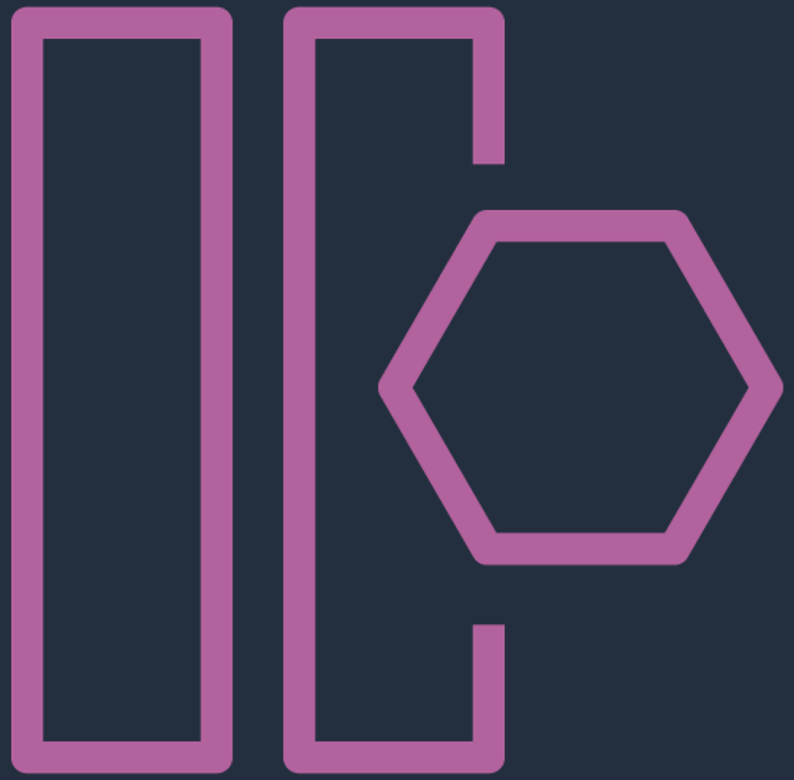
- API defines the interface
  - e.g. REST, GraphQL
- Caller expects an immediate response
  - Response contains the result of the work
  - Generally, under 30 seconds
  - Synchronous processing
- Client must implement error handling, retry logic



Event-Driven Architectures (EDA)

- Event payload defines the interface
  - e.g. JSON
- Response is “message received” (or not)
  - Asynchronous processing
  - Process can be long running (beyond 30s)
  - Updating client (e.g. UI) can be more challenging
- Higher resiliency, durability
  - Driven by messaging service
- Built-in retries, configurable to use case





events

in serverless architectures

Connective tissue of loosely coupled systems.

Represents change in state. Improving resilience.

Trends of event-driven architectures

WHY CUSTOMERS ARE MOVING TO EVENT-DRIVEN APPLICATIONS

1

Speed & agility

Move faster. Build and deploy services independently.

2

Resiliency

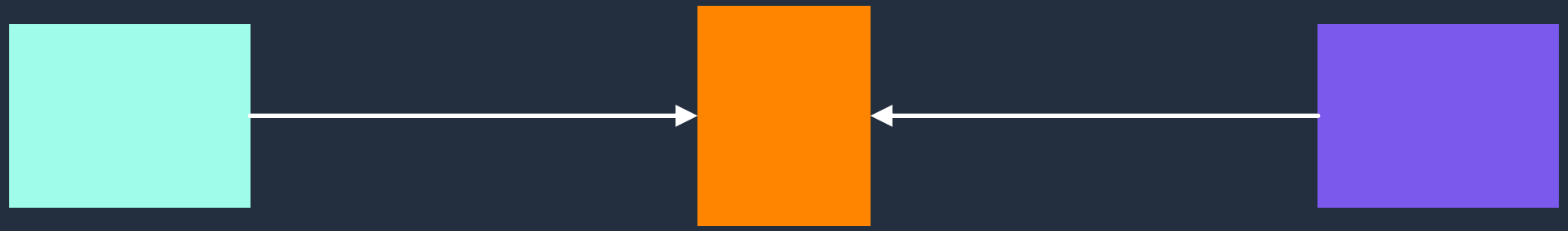
Loosely coupled systems can run and fail independently.

3

Scalability

Minimize waiting time through async and parallel processing.

EDA components



Producer

Event producers are systems that detect a change in state or notice updates and publish those facts. Application code, a database or a time-based trigger can serve as event producers, among other things.

Event broker

An event broker is a meeting place between the producers and consumers. The broker is how events are exchanged.

Consumer

Event consumers are systems that listen for events and use them for their purposes. It's possible, and common, for a consumer to receive an event, perform some work and publish its event in response.

Event brokers

Queues

- Amazon SQS
- Amazon MQ

Streams

- Kinesis
- Kafka/MSK

Routers

- EventBridge
- SNS

EDA on AWS

Amazon SQS

Messaging

Durable and scalable
Fully managed
Comprehensive security

Amazon SNS

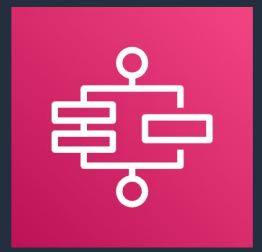
Notifications

Performance at scale
Fully managed
Enterprise-ready

Amazon EventBridge

Choreography

Event filtering
Managed & scalable
SaaS integration



AWS Step Functions

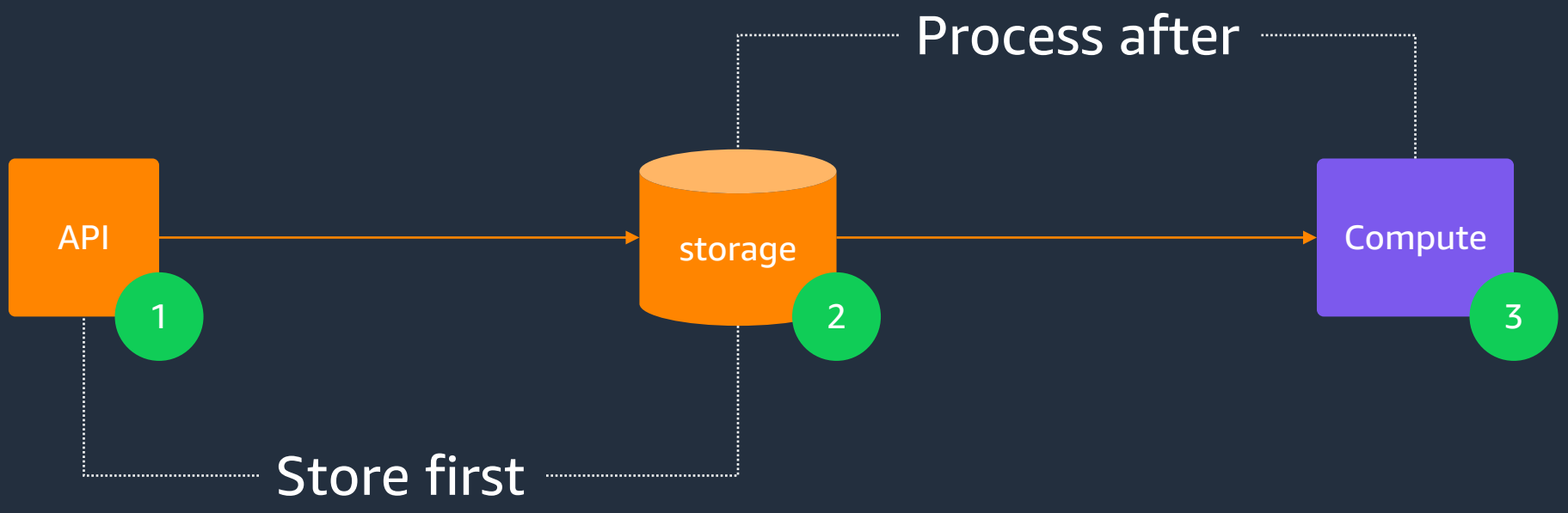
Orchestration

Sequencing
Parallel execution
State management

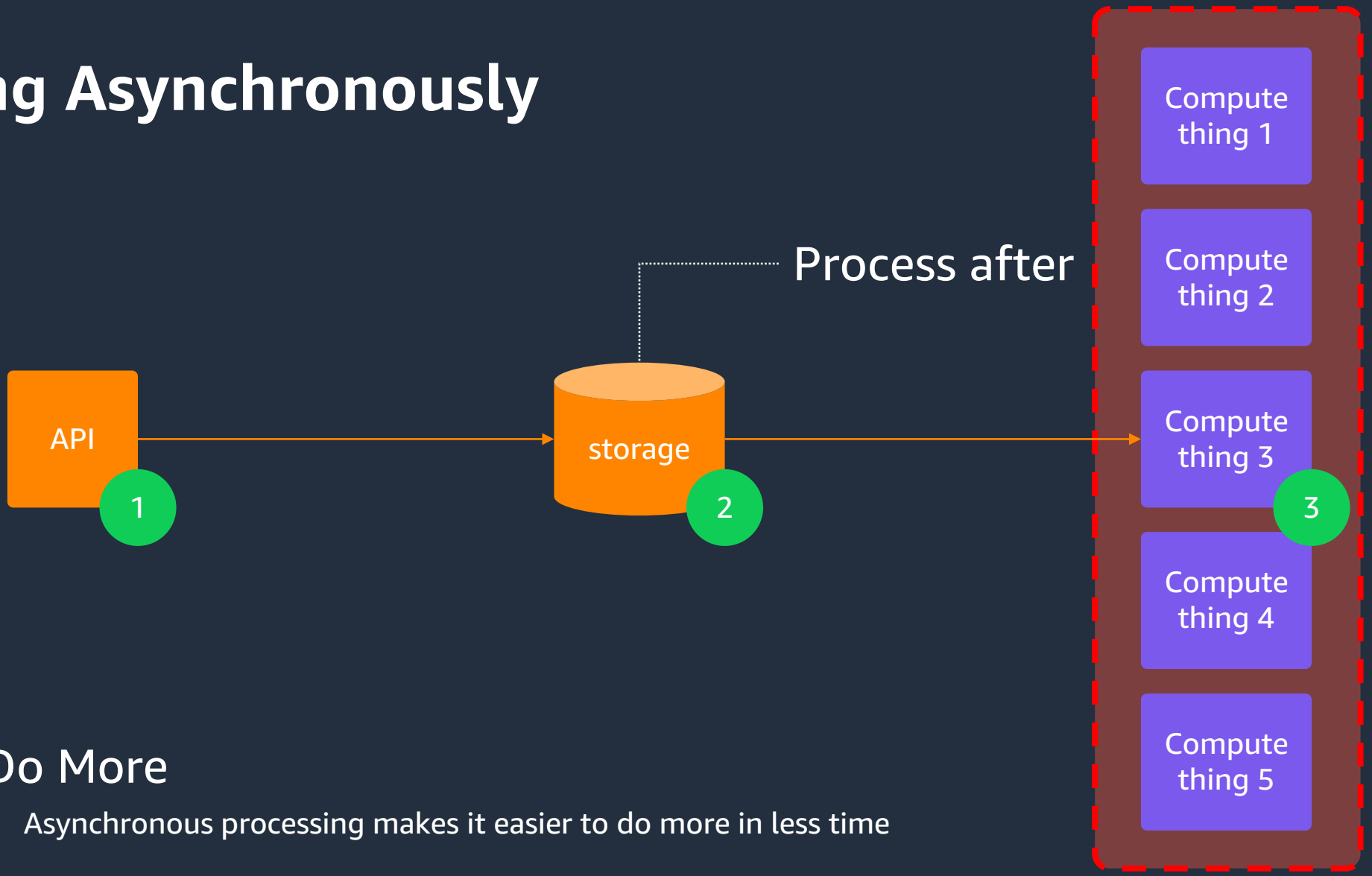
Thinking Asynchronously

What does it mean?

Thinking Asynchronously



Thinking Asynchronously



Do More

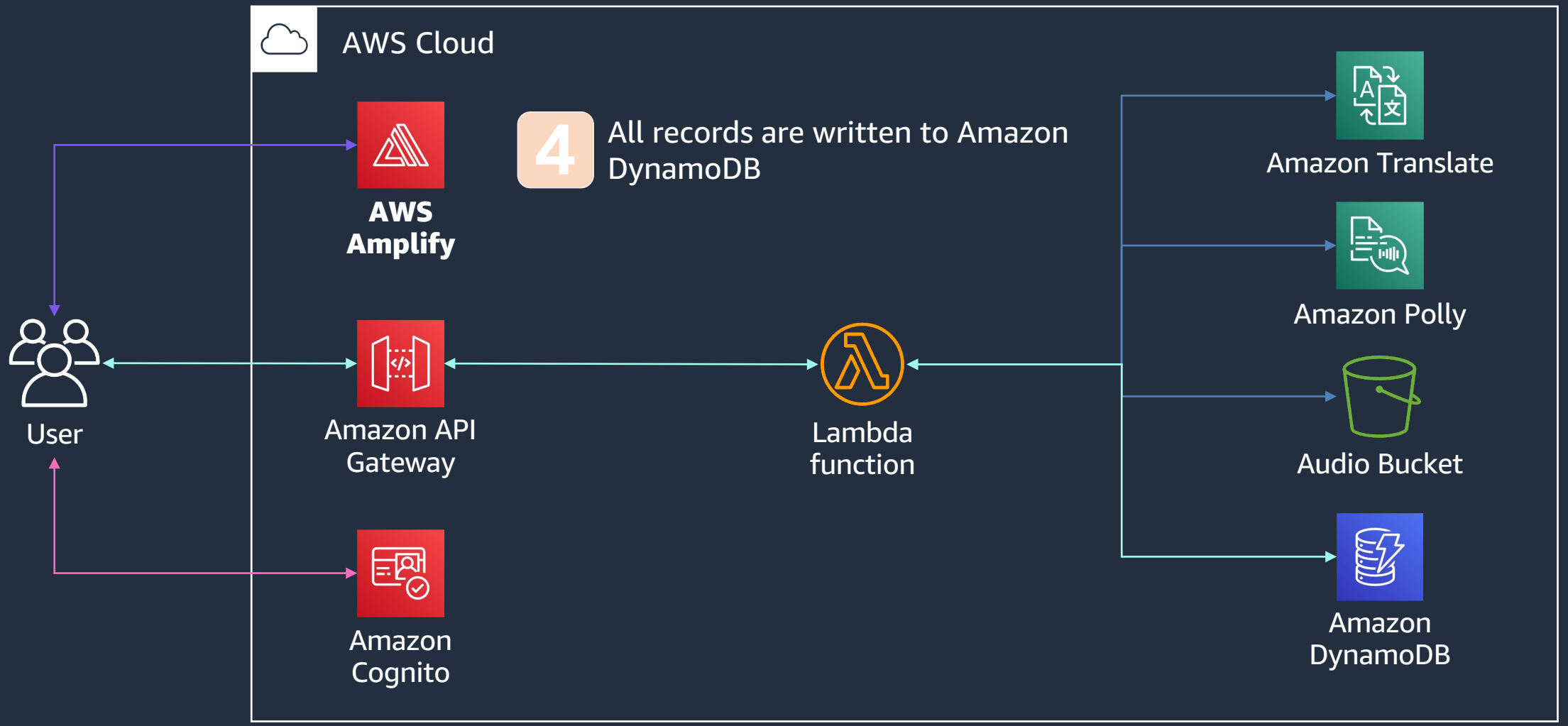
Asynchronous processing makes it easier to do more in less time

Synchronous application

Event data

```
{
     "data":”Hello, welcome to my translation application.",
     "culture":[”fr-FR","tr-TR”,“de-DE”,”fr-CA”]
}
```

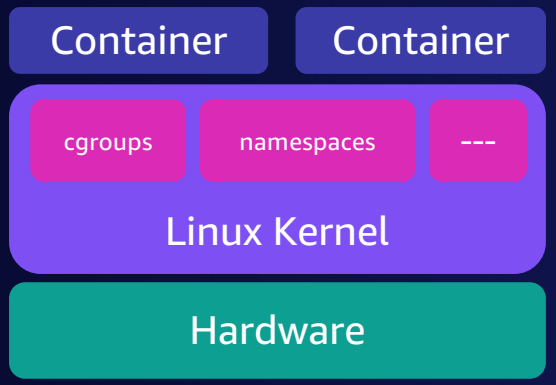
Example synchronous application



Containers vs VMs

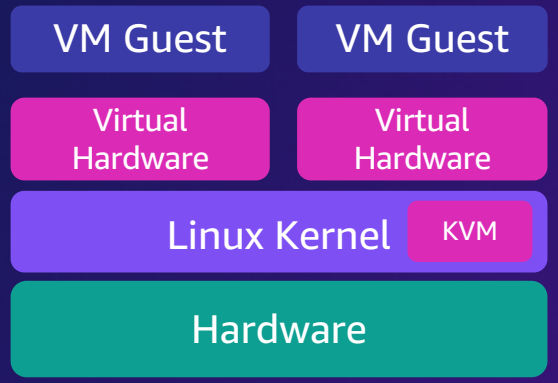
Containers

- Using Linux primitives for isolation
- Share Linux Kernel
- Fast starts, minimal overhead
- Flexible isolation



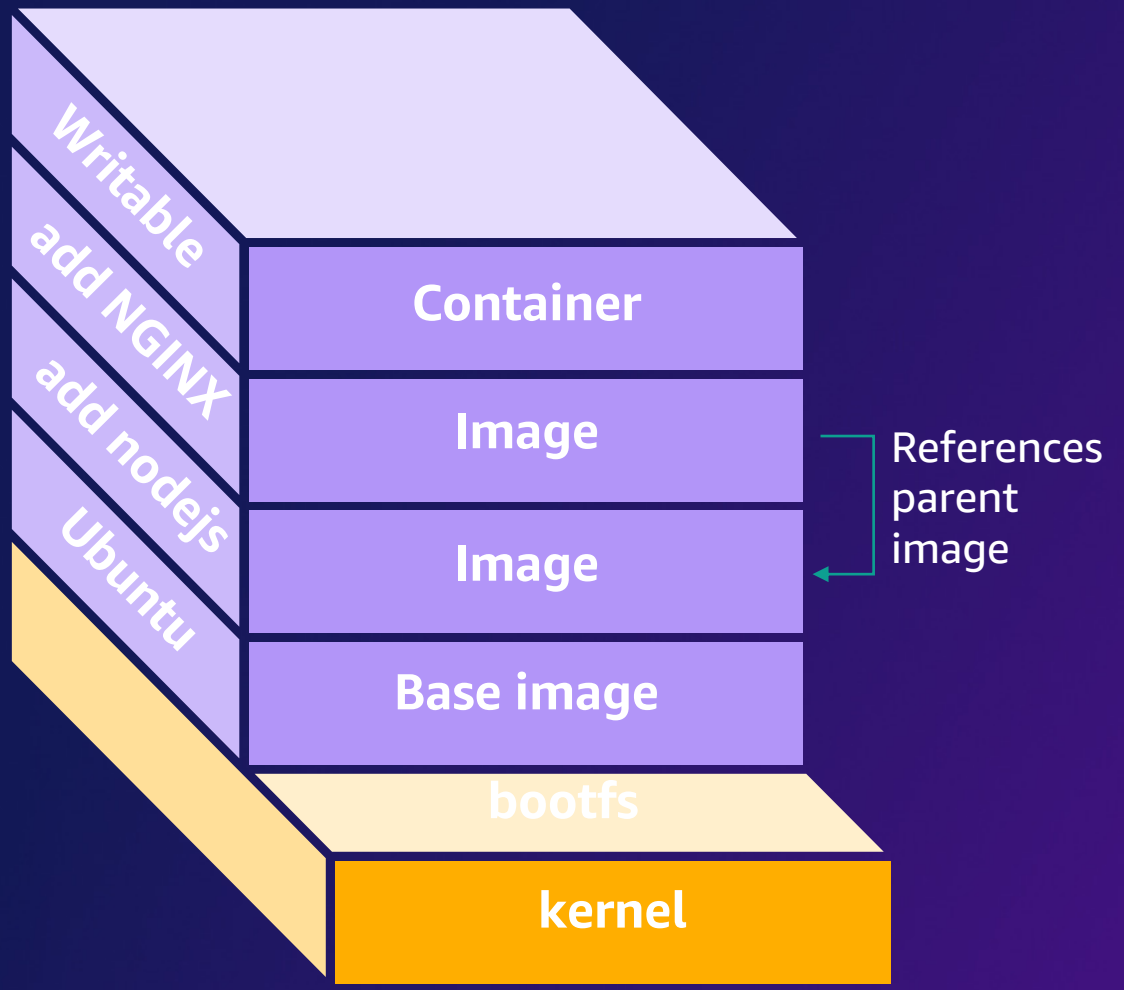
Virtual Machines

- Virtualisation or emulate hardware components
- Completely separate kernels (maybe not Linux)
- Slower starts, must boot kernel and set-up hardware.

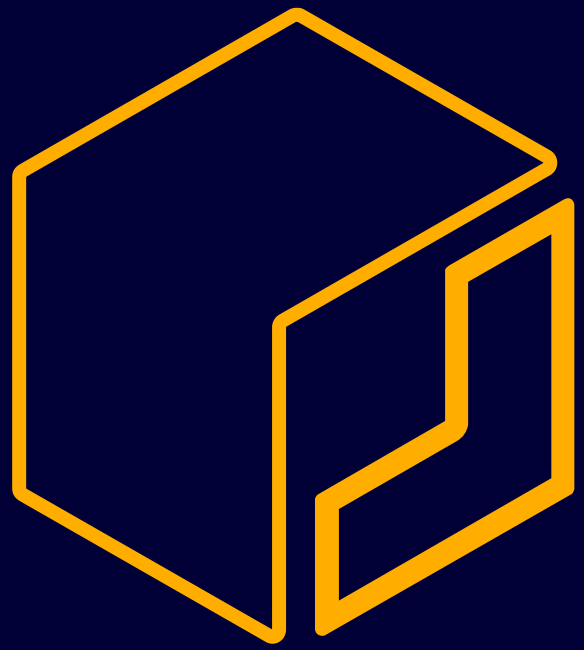


Container images

- Read only image that is used as a template to launch a container.
- Start from base images that have your dependencies, add your custom code.
- Dockerfile for easy, reproducible builds.



Managing many containers is hard

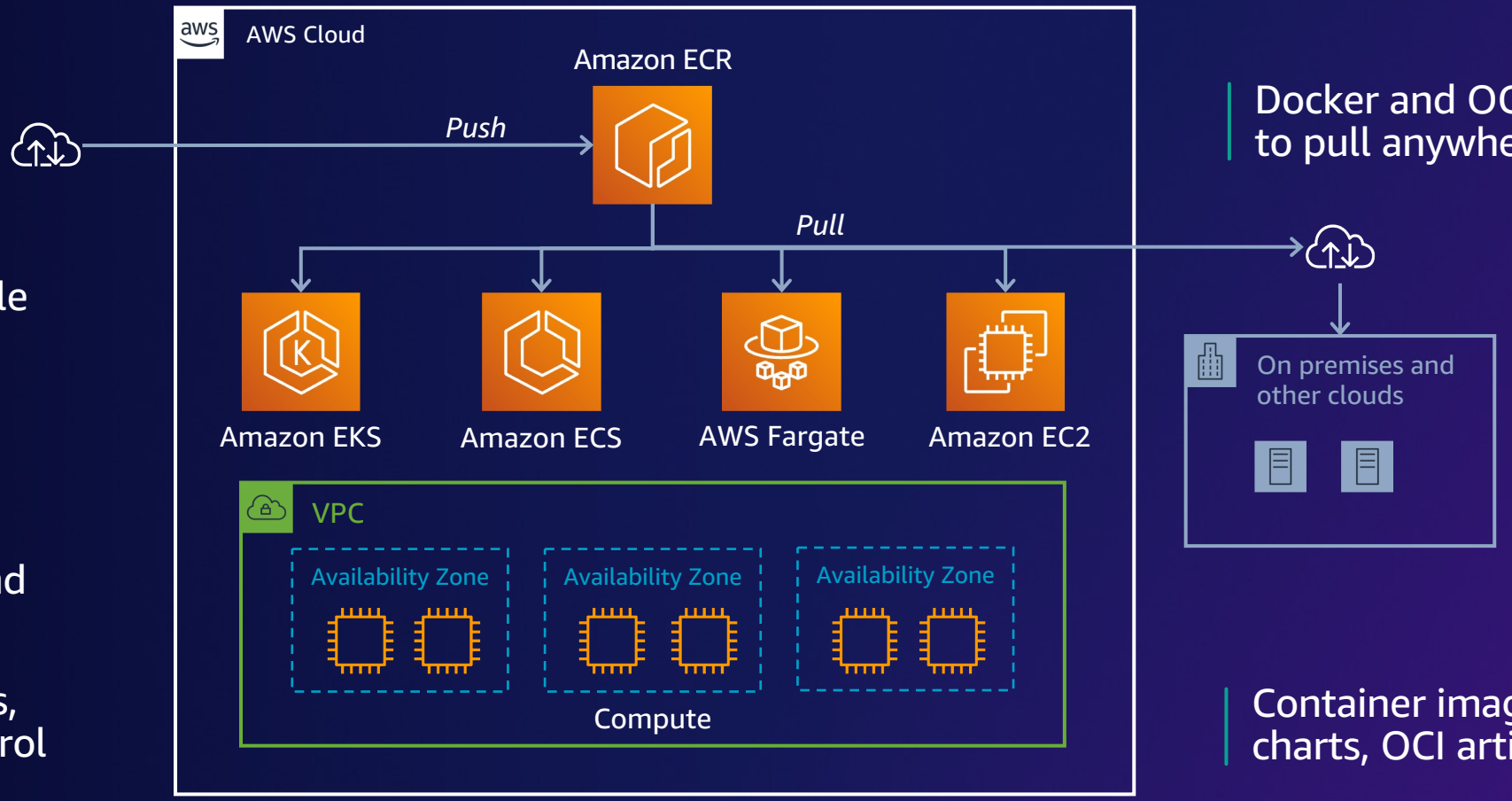


Amazon Elastic Container Registry

aws

What is Amazon ECR

FULLY-MANAGED CONTAINER ARTIFACT REGISTRY



- Managed and scalable infrastructure
- Highly available, high performance
- Security with encrypted images and vulnerability scans
- Authenticated access, centralized IAM control

- Docker and OCI compliant to pull anywhere
- Container images, Helm charts, OCI artifacts

- Native integration to AWS orchestrators and compute

Amazon ECR Registries

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

Amazon ECR Repositories

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

team-a/web-app

team-b/web-app

Container Images

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

team-a/web-app

team-b/web-app

My app image

OCI image
OCI Artifacts

Container Images: Lifecycle policies

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

team-a/web-app

My app image

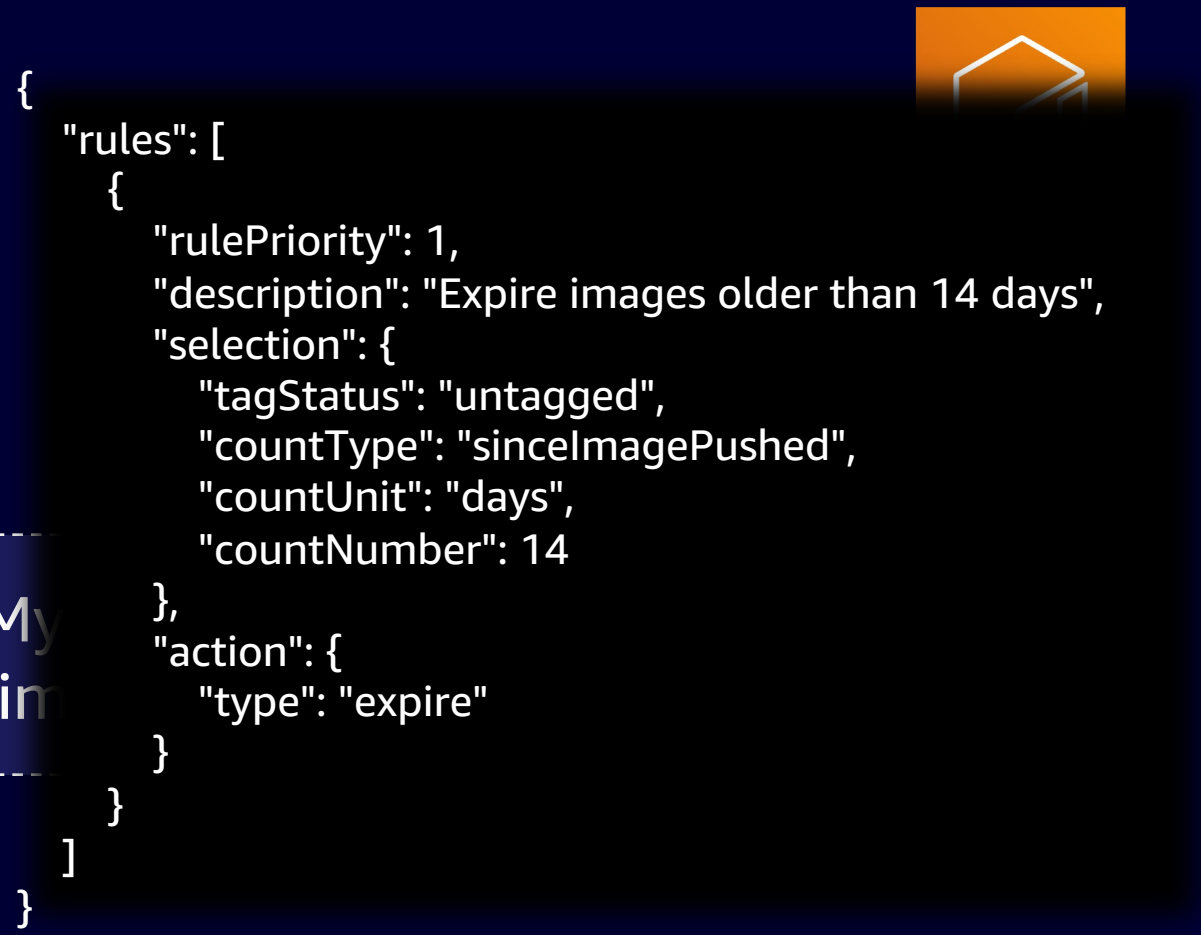
My app image **:1**

My app image **:<no tag>**

My app image **:3**

Container Images: Lifecycle policies

https://205094881157.dkr.ecr.us-west-2.amazonaws.com



```
{
   "rules": [
      {
         "rulePriority": 1,
         "description": "Expire images older than 14 days",
         "selection": {
            "tagStatus": "untagged",
            "countType": "sinceImagePushed",
            "countUnit": "days",
            "countNumber": 14
         },
         "action": {
            "type": "expire"
         }
      }
   ]
}
```

My app image

My app image :1

My app image :<no tag>

My app image :3

Container Images: Image scanning

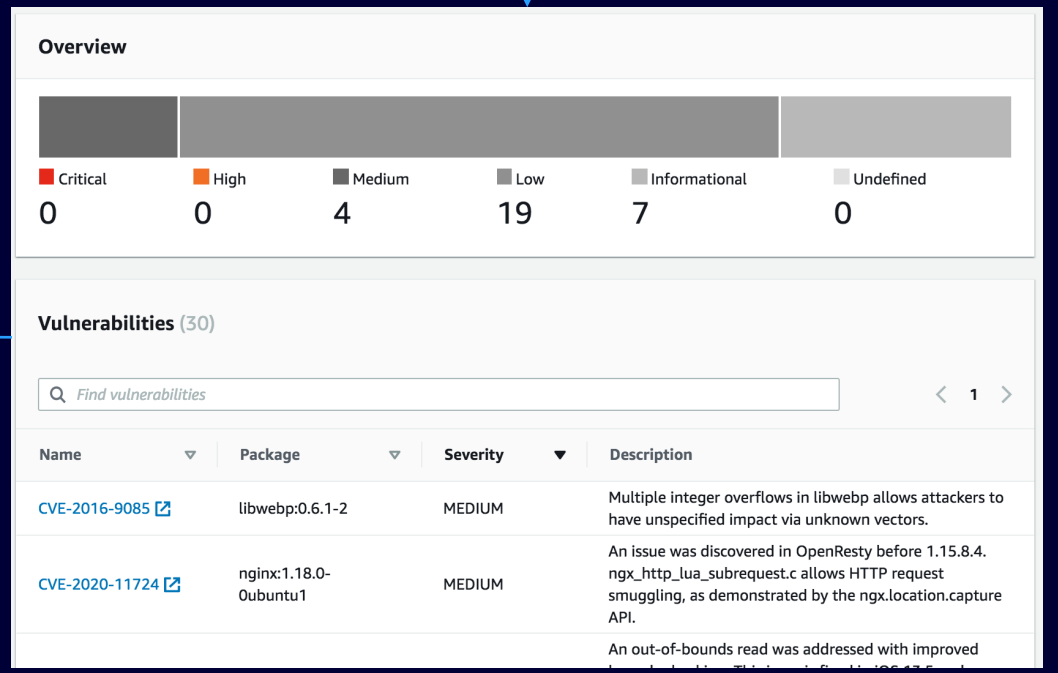
https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon
ECR

team-a/web-app

My app
image

Amazon
EventBridge



Container Images

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

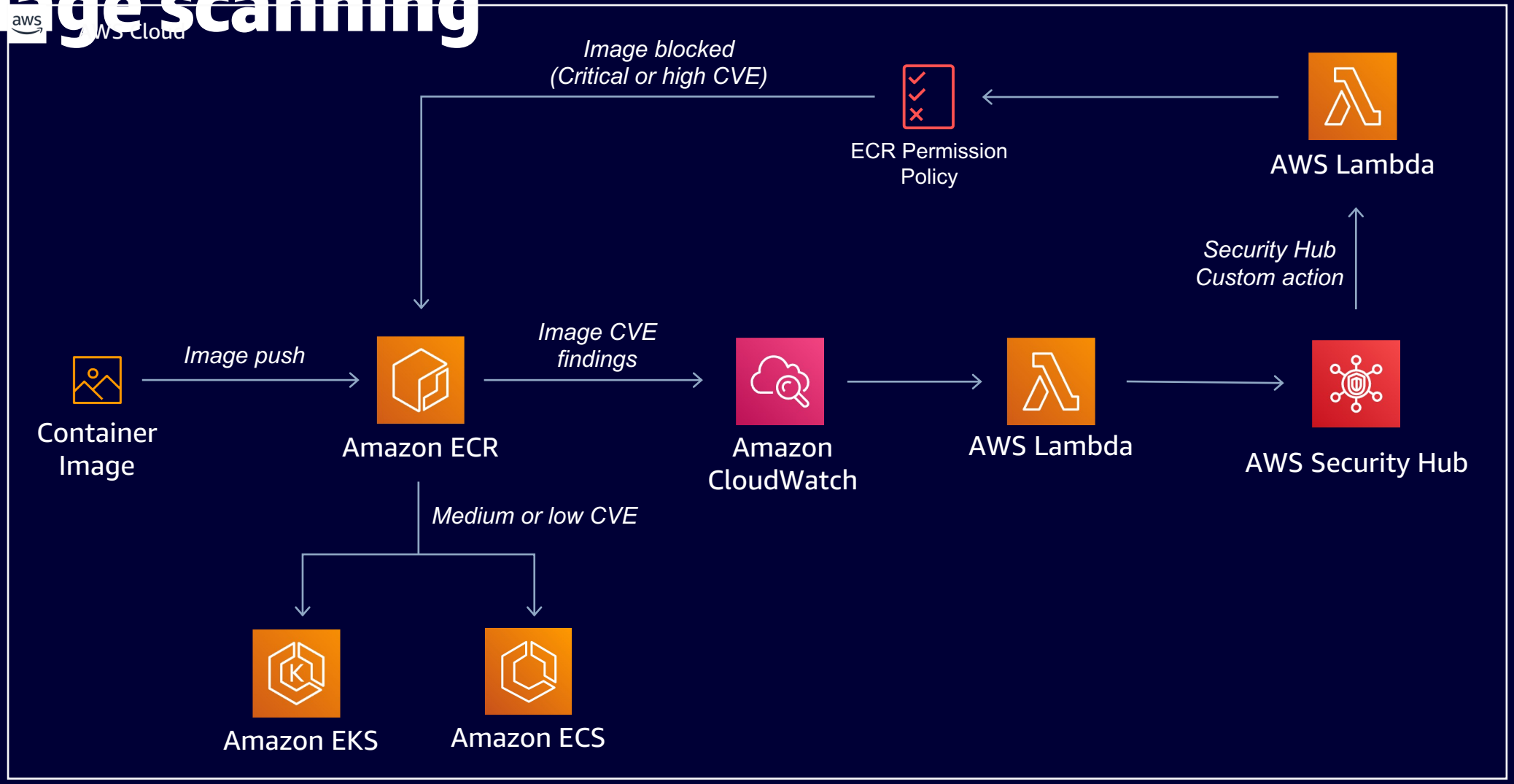
team-a/web-app

team-b/web-app

My app image

OCI image
OCI Artifacts

Container Images: Automate compliance with image scanning



Security in Amazon Elastic Container Registry

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

Amazon ECR

team-a/web-app

Team B

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ecr:GetAuthorizationToken",
            "ecr:BatchCheckLayerAvailability",
            "ecr:GetDownloadUrlForLayer",
            "ecr:GetRepositoryPolicy",
            "ecr:DescribeRepositories",
            "ecr:ListImages",
            "ecr:DescribeImages",
            "ecr:BatchGetImage",
            "ecr:GetLifecyclePolicy",
            "ecr:GetLifecyclePolicyPreview",
            "ecr:ListTagsForResource",
            "ecr:DescribeImageScanFindings"
         ],
         "Resource": "*"
      }
   ]
}
```

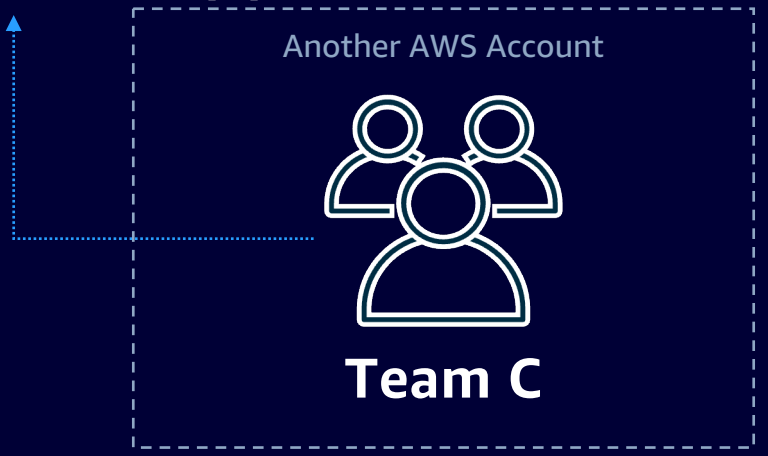
Security in Amazon Elastic Container Registry

https://205094881157.dkr.ecr.us-west-2.amazonaws.com

```
{
   "Version": "2008-10-17",
   "Statement": [
      {
         "Sid": "AllowPushPull",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::account-id:user/push-pull-user-1",
               "arn:aws:iam::account-id:user/push-pull-user-2"
            ]
         },
         "Action": [
            "ecr:GetDownloadUrlForLayer",
            "ecr:BatchGetImage",
            "ecr:BatchCheckLayerAvailability",
            "ecr:PutImage",
            "ecr:InitiateLayerUpload",
            "ecr:UploadLayerPart",
            "ecr:CompleteLayerUpload"
         ]
      }
   ]
}
```

Amazon ECR

team-b/web-app

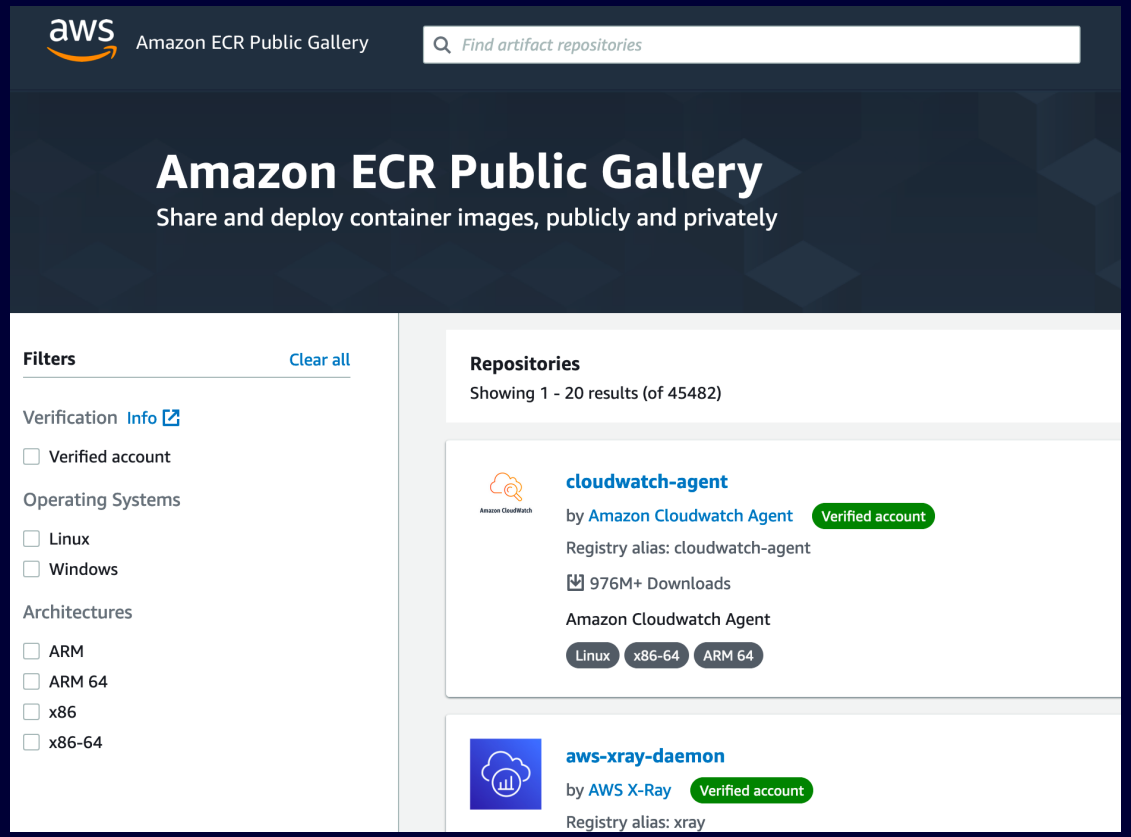


Amazon ECR Public

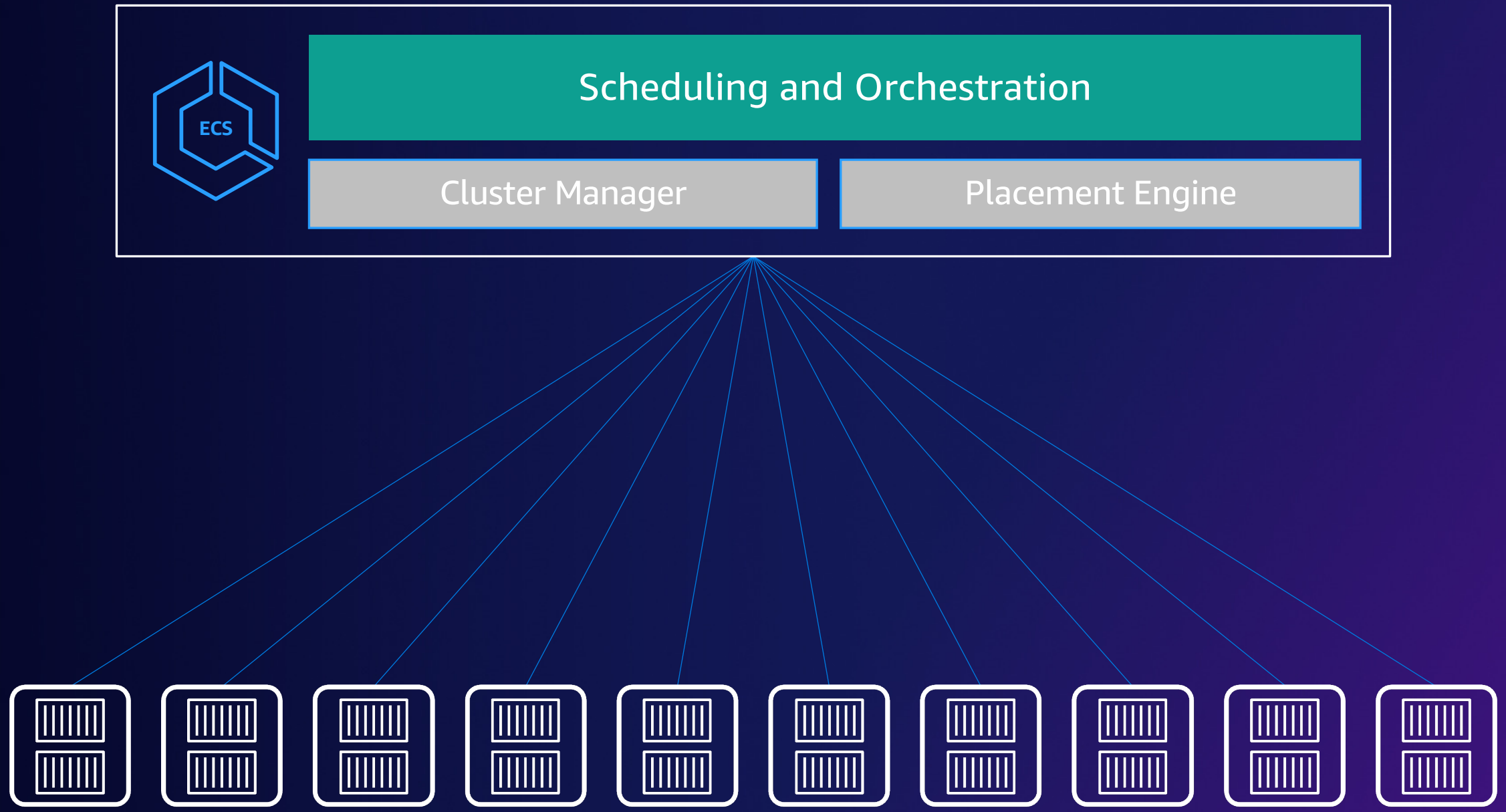
registry-alias/web-app

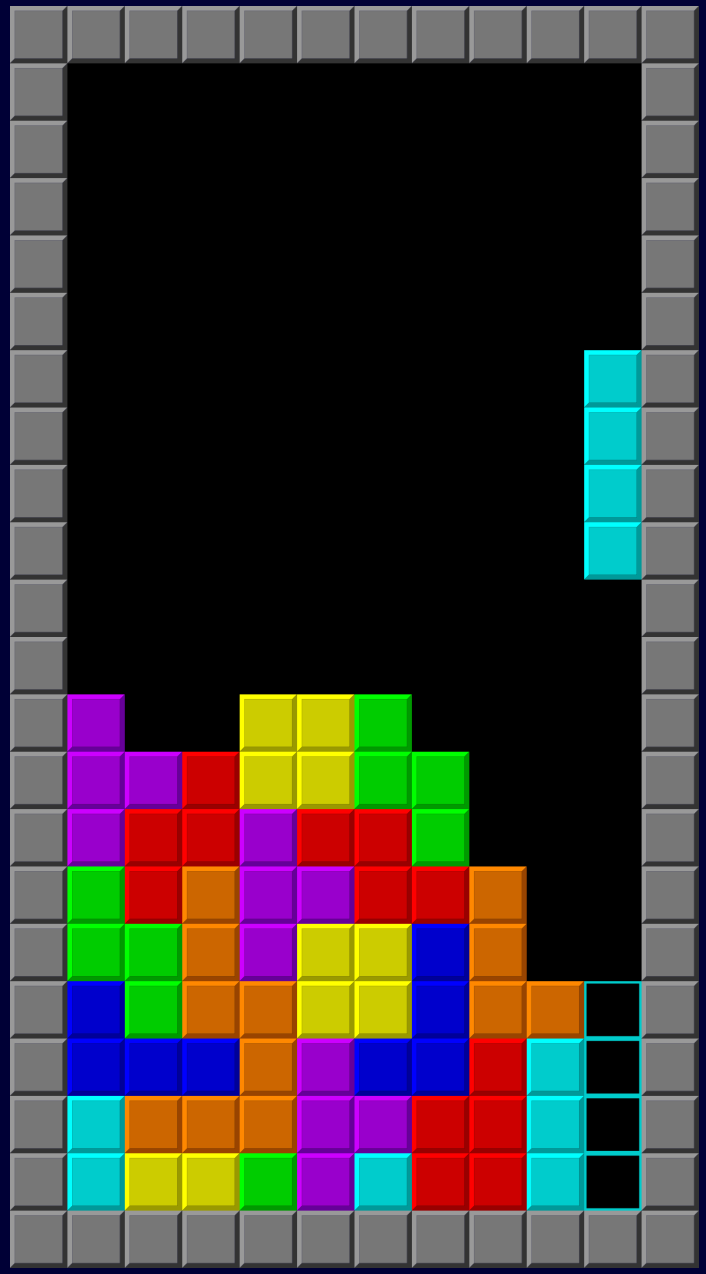
```
{
   "Version": "2008-10-17",
   "Statement": [
      {
         ”Sid": ”ECR Public Repository Policy"
            "Effect": "Allow",
            ”Principal": {
               ”AWS”: "arn:aws:iam::account-id:user/username”
            }
          "Action": [
             "ecr-public:DescribeImages ",
             "ecr-public:DescribeRepositories "
            ]
      }
   ]
}
```

https://gallery.ecr.aws



Amazon Elastic Container Service





aws

Amazon ECS constructs

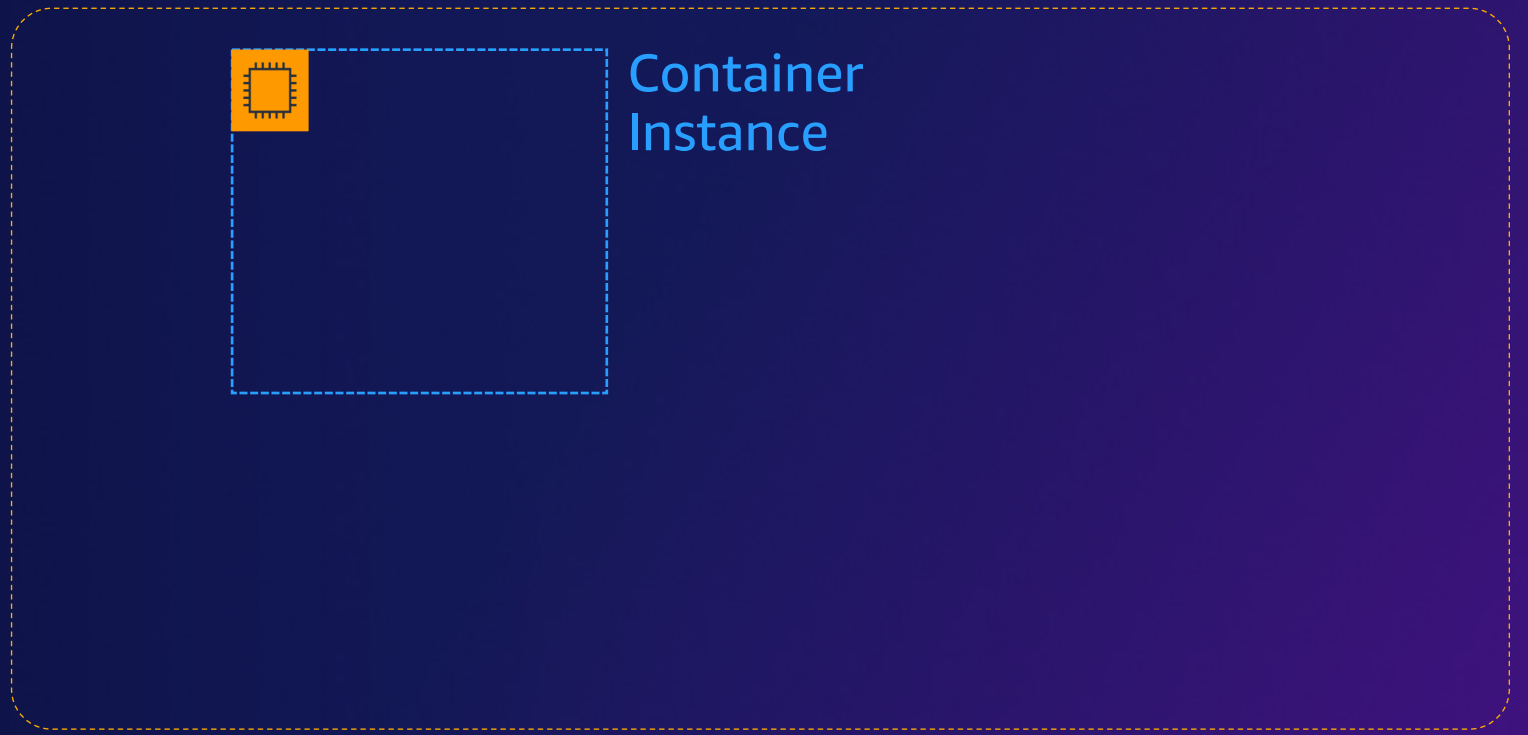
Cluster

- Resource grouping and isolation
- IAM permissions boundary

Amazon ECS constructs

Cluster

- Resource grouping and isolation
- IAM permissions boundary



Amazon ECS constructs

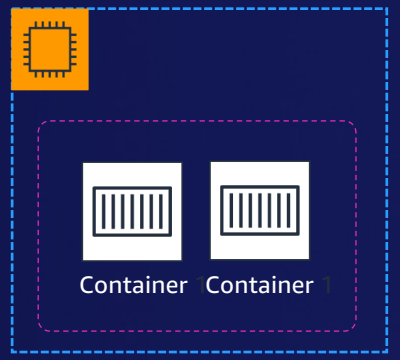
{ ; }
JSON

Task definition

- Template used by Amazon ECS to launch tasks
- Parallels to docker run parameters
- Defines requirements:
  - CPU/Memory
  - Container image(s)
  - Logging
  - IAM role
  - Etc.

Cluster

- Resource grouping and isolation
- IAM permissions boundary



Container Instance

Task

- Running instance of a task definition
- One or more containers

Amazon ECS constructs

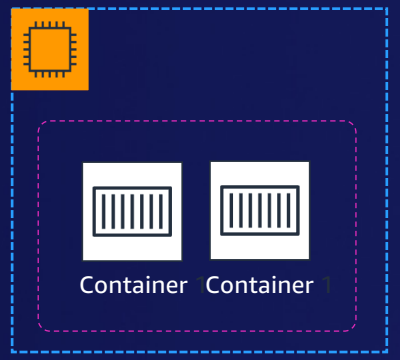
{ ; }
JSON

Task definition

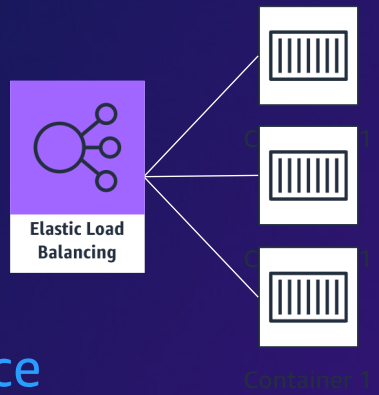
- Template used by Amazon ECS to launch tasks
- Parallels to docker run parameters
- Defines requirements:
  - CPU/Memory
  - Container image(s)
  - Logging
  - IAM role
  - Etc.

Cluster

- Resource grouping and isolation
- IAM permissions boundary



Container Instance



Task

- Running instance of a task definition
- One or more containers

Service

- Maintains desired # of running tasks
- Replaces unhealthy tasks
- ELB integration

Task definition

```
{
  "containerDefinitions": [
    {
      "memory": 128,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true,
      "name": "nginx-container",
      "image": "nginx",
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "ecs-log-streaming",
          "awslogs-region": "us-west-2",
          "awslogs-stream-prefix": "fargate-task-1"
        }
      },
```

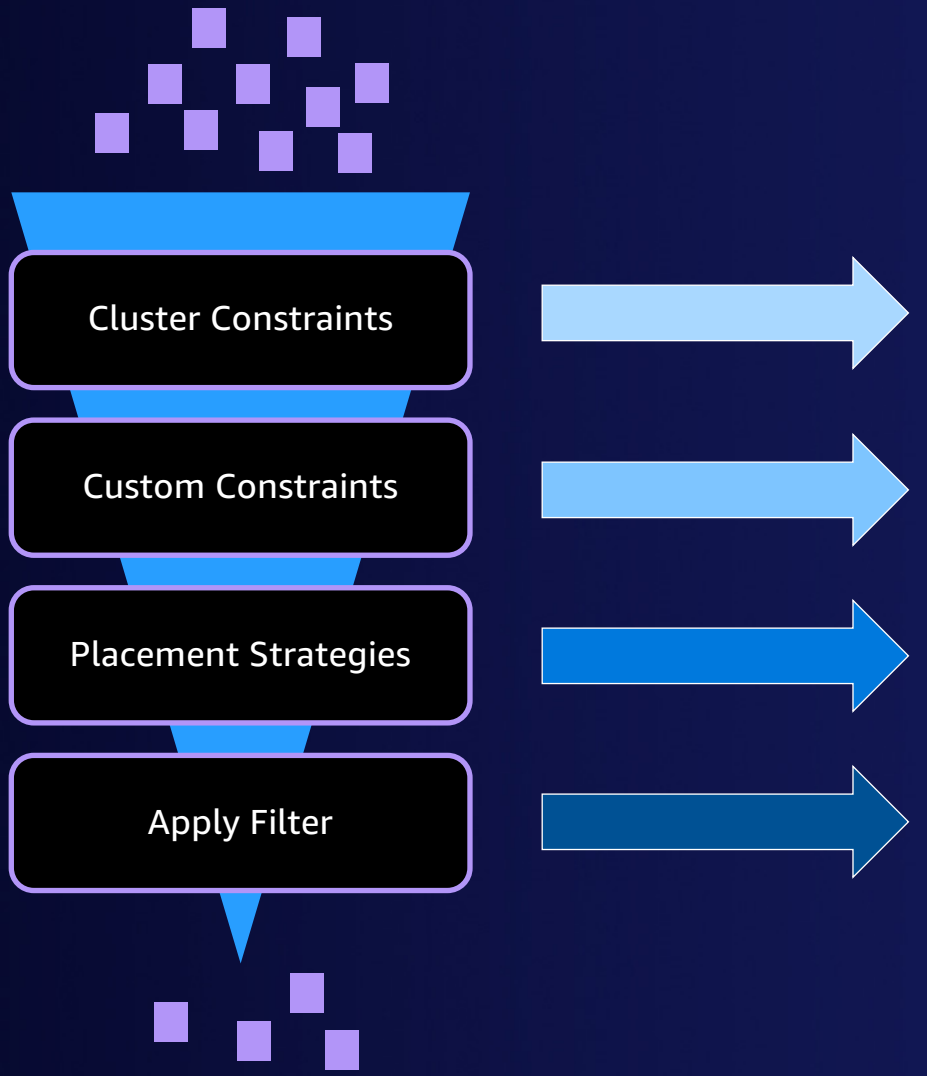
continued…

```
      "cpu": 0
    }
  ],
  "networkMode": "awsvpc",
  "executionRoleArn":
"arn:aws:iam::123456789012:role/ecsTask
ExecutionRole",
  "memory": "2048",
  "cpu": "1024",
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "family": "example_task_1"
}
```

Deploying on ECS: Tasks vs Services

On-Demand Workloads	Long-Running Apps
ECS task scheduler	ECS service scheduler
Run once or at intervals	Health management
Batch jobs	Scale-up and scale-down
RunTask API	AZ aware
StartTask (custom)	Grouped containers

Task placement



Satisfy CPU, memory, and networking requirements

Filter for location, instance-type, AMI, or other custom attribute constraints

Identify instances that meet spread or binpack placement strategy

Select final container instances for placement

Amazon ECS powers Amazon

Amazon
SageMaker

Amazon
Lex

Amazon
Polly

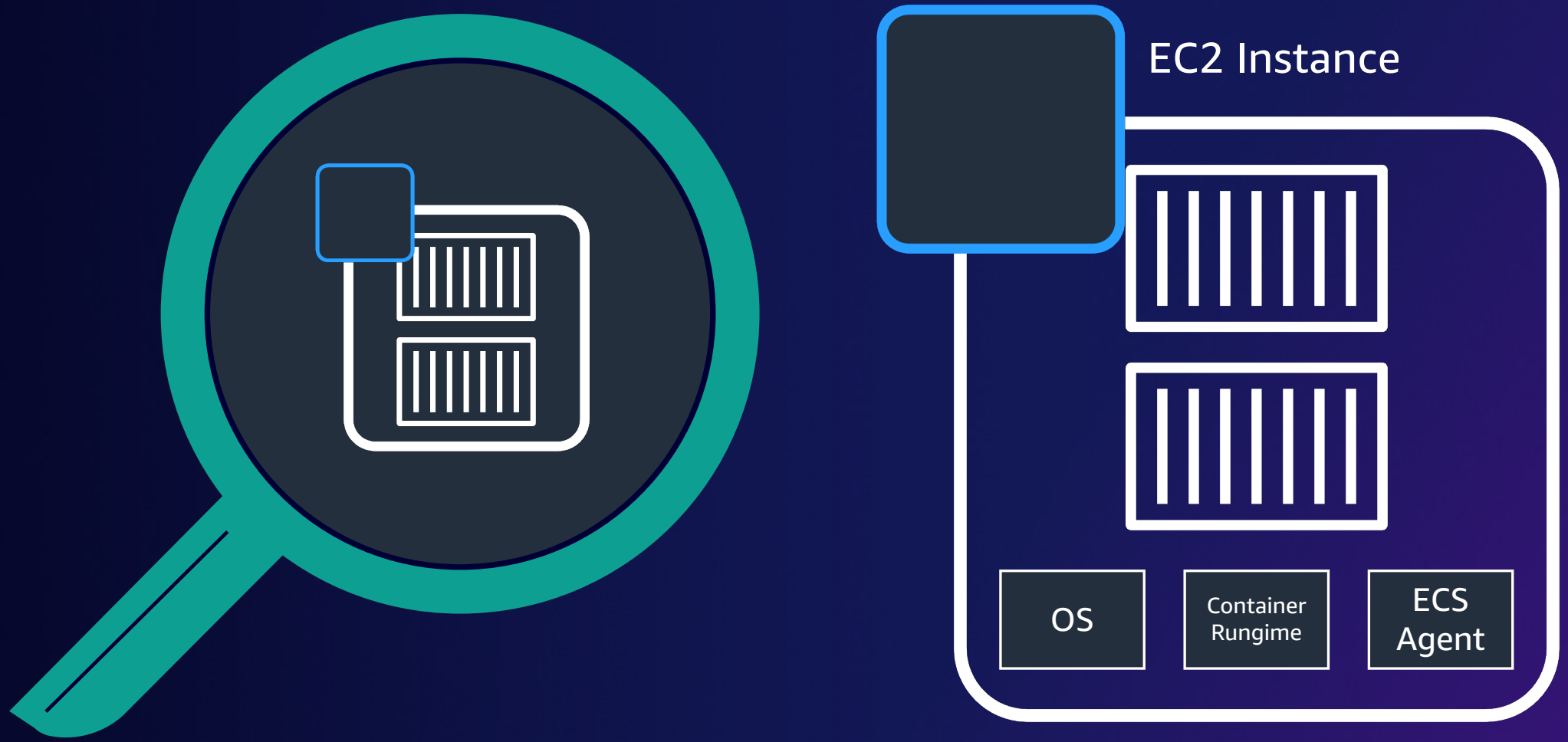
AWS
Batch

Amazon ECS forms the building blocks
for various services at Amazon

Built for security, reliability, availability, and scale

AWS Fargate

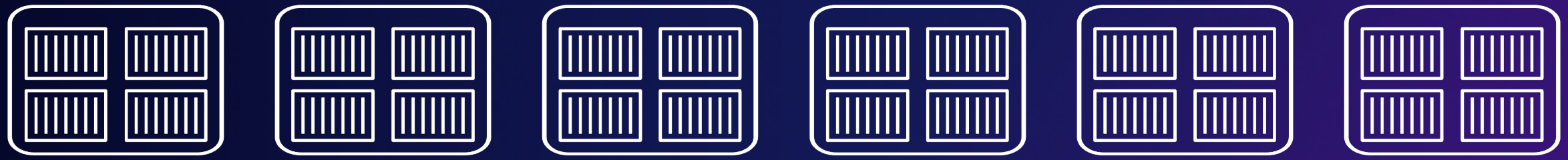
Without Fargate, you end up managing more than just containers



- - Patching and Upgrading OS, agents, etc.
- - Scaling the instance fleet for optimal utilization

Amazon Elastic Container Service

Amazon Elastic Container Service

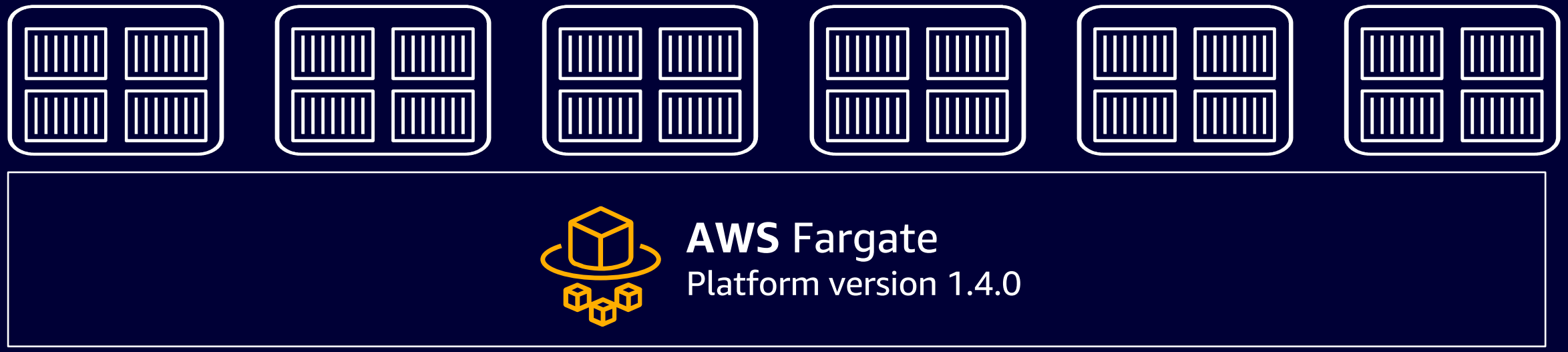


AWS Fargate

run serverless containers

AWS Fargate platform versions

Amazon Elastic Container Service



Amazon ECS on AWS Fargate

Operating systems supported

- Amazon Linux 2 (ARM64 and X86_64)

- Microsoft Windows Server:
  - 2019 Full (X86_64)
  - 2019 Core (X86_64)
  - 2022 Full (X86_64)
  - 2022 Core (X86_64)

Security: Benefits of Fargate

We do more, you do less.

- Patching (OS, Docker, ECS Agent, etc.)
- Task isolation (via Clusters)
- No `--privileged` mode for containers
- Requires awsvpc network mode so there is an ENI and SG per Task
- Simple secure and auditable way to run commands in container – ECS Exec

Your containerized applications

Managed by AWS

No EC2 Instances to provision, scale or manage

Elastic

Scale up & down seamlessly. Pay only for what you use

Integrated

With the AWS ecosystem: VPC Networking, Elastic Load Balancing, IAM Permissions, CloudWatch and more

Enterprise Grade

Payment Card Industry (PCI) Security Standard

DOD Cloud Security Req's Guide (SRG)

FedRAMP Moderate and High (GovCloud)

Criminal Justice Information Service Security Policy (CJIS)

U.S. Health Insurance Portability and Accountability Act (HIPAA)

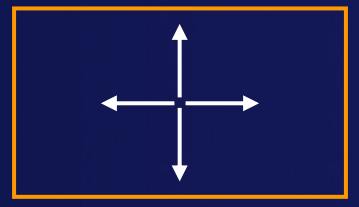
SP 800-53 (rev 4)
SP 800-171

Federal Information Processing Standard Pub (FIPS) 140-2

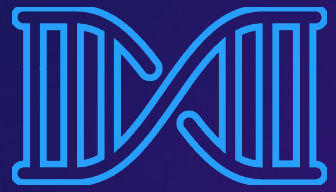
Health Information Trust Alliance Common Security Framework

What is Kubernetes?

Open source container management platform



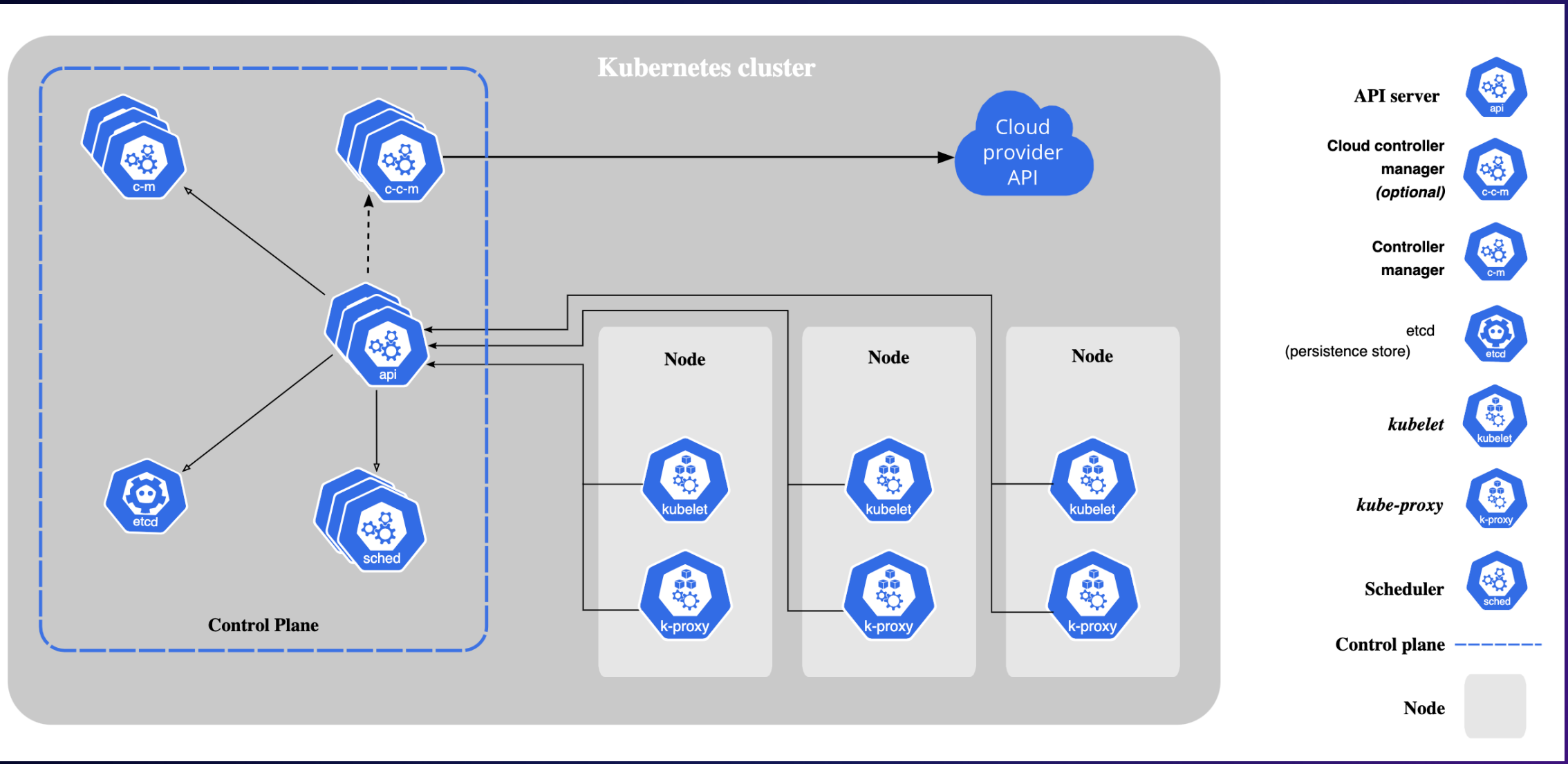
Helps you run containers at scale



Gives you primitives for building modern applications

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Kubernetes basic architecture



What is Kubernetes?

Objects

Pod

A thin wrapper around one or more containers

DaemonSet

Implements a single instance of a pod on a worker node

Deployment

Details how to roll out (or roll back) across versions of your application

ReplicaSet

Ensures a defined number of pods are always running

Objects

Job

Ensures a pod properly runs to completion

Service

Maps a fixed IP address to a logical group of pods

Label

Key/Value pairs used for association and filtering

kubectl

Command line interface

Namespace

logically named group

Amazon EKS

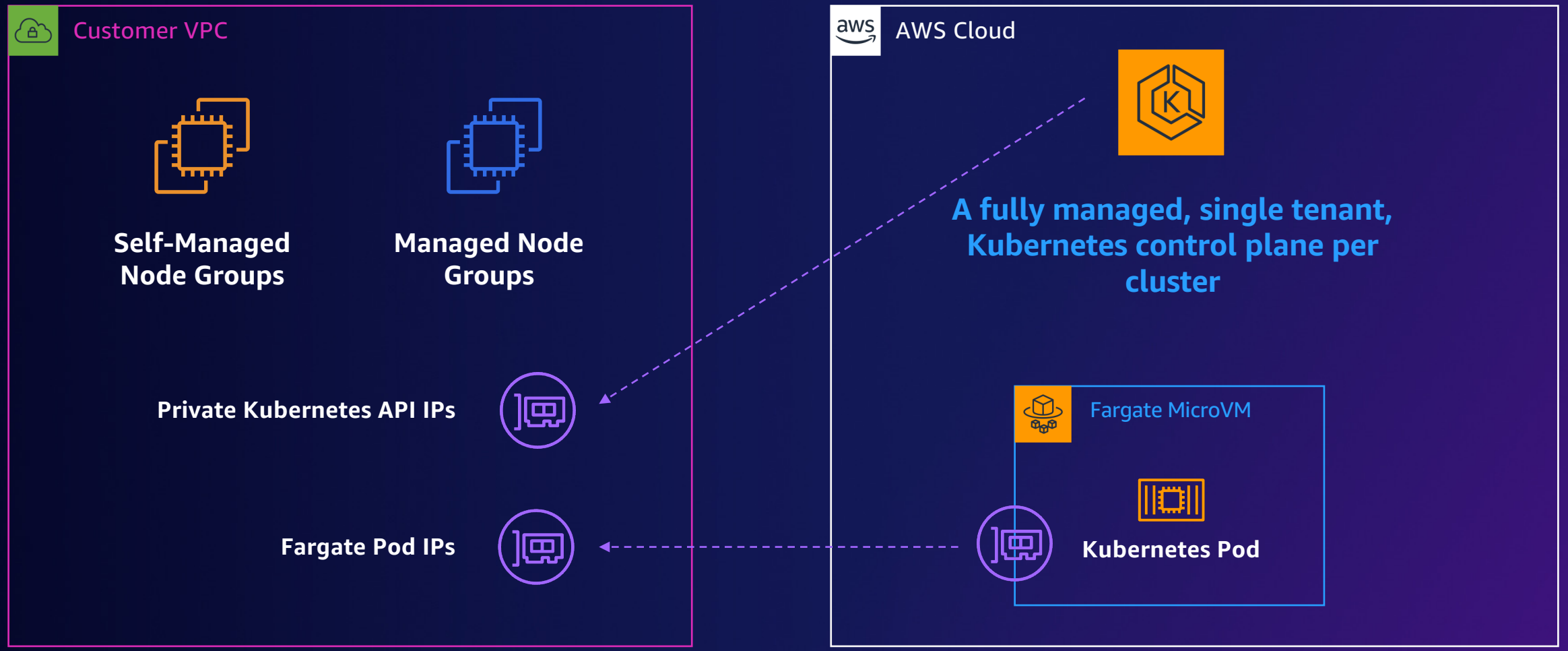
What is Amazon EKS?

Amazon **EKS**

- Amazon EKS runs vanilla Kubernetes; EKS is upstream and a certified conformant version of Kubernetes (with backported security fixes)
- Amazon EKS supports 4 versions of Kubernetes, giving you time to test and roll out upgrades
- Amazon EKS provides a managed Kubernetes experience for performant, reliable, and secure Kubernetes
- Amazon EKS makes Kubernetes operations, administration, and management simple

Amazon EKS helps you build reliable, stable, and secure applications in virtually any environment

Amazon EKS High Level Architecture



Amazon EKS Control Plane Architecture

Survive single-AZ events

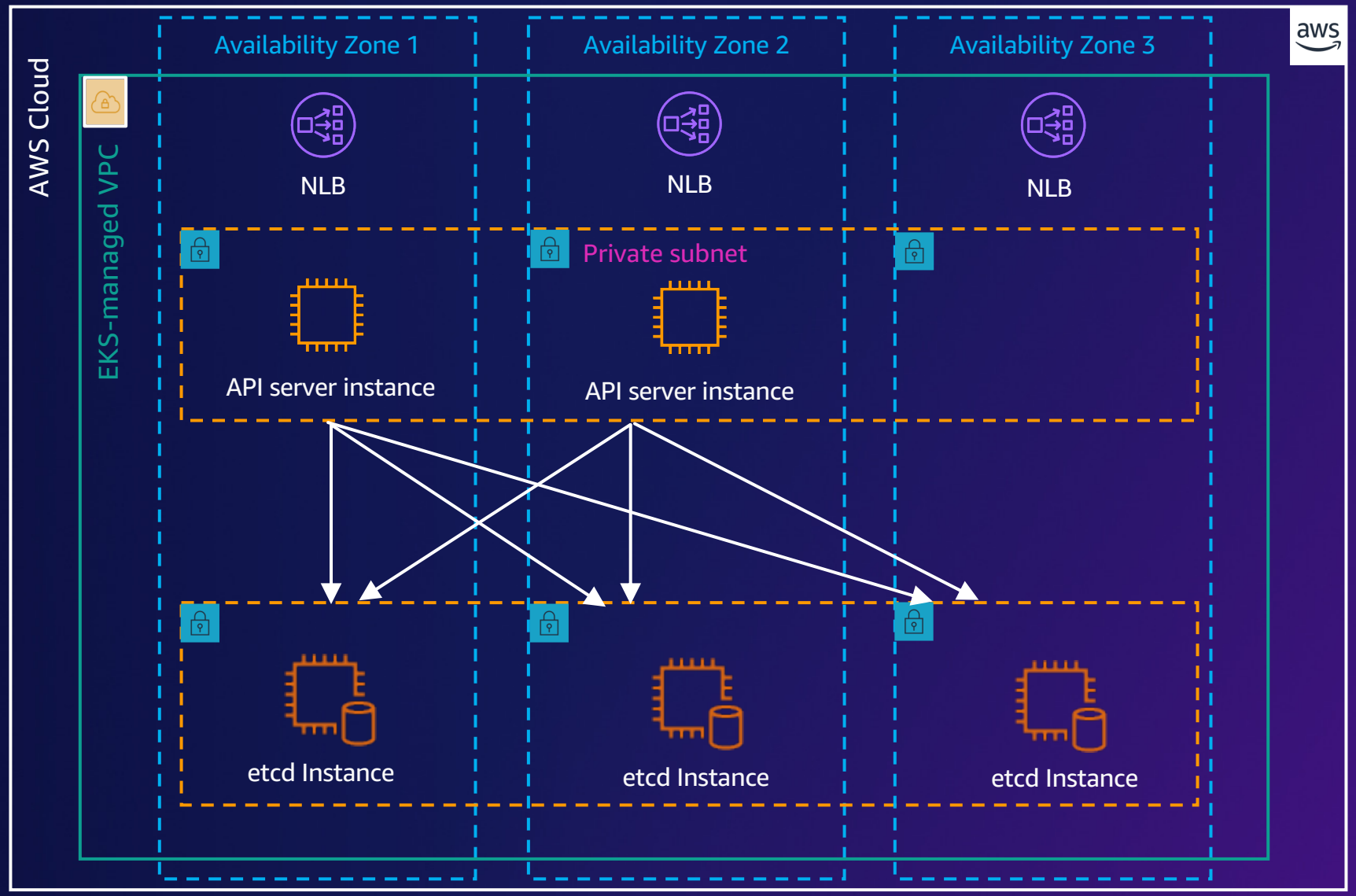
Highly available cluster endpoint

99.95% SLA

24x7x365 support

Automatic Resizing

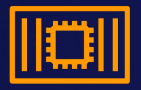
Increased the volume throughput 6x



Amazon EKS Runtimes Overview

Administrator deploys Pod

Kubernetes schedules the pod



Amazon EKS supports running containers on EC2 instances or on AWS Fargate. Clusters can run containers on a single runtime or multiple runtimes at the same time.

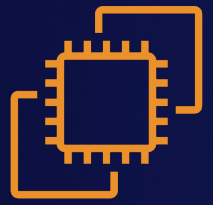
Amazon **EC2**

Run containers on EC2 instances within your account that you manage and configure.

AWS **Fargate**

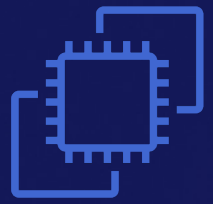
A fully managed container environment with no infrastructure management.

Amazon EC2 Runtime Options



Self-Managed Node Groups

Bring your own Autoscaling Groups running your own custom AMI. You are responsible for patching and the underlying OS.



Managed Node Groups

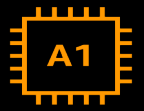
Provisioned by EKS in your VPC. They run the latest EKS optimized AMI. Handles automatic draining and rolling out new AMIs.

Runtime Options

GPU Based Instances

Nitro Based Instances

AMD Backed Instances



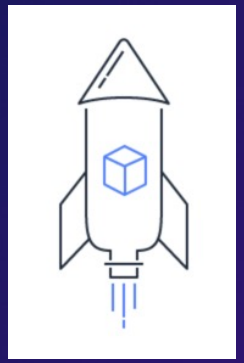
Graviton 2 Instances

Spot Instances and Compute Savings Plans

AWS Outpost

AWS Local Zones

Bring Your Own OS or use ours

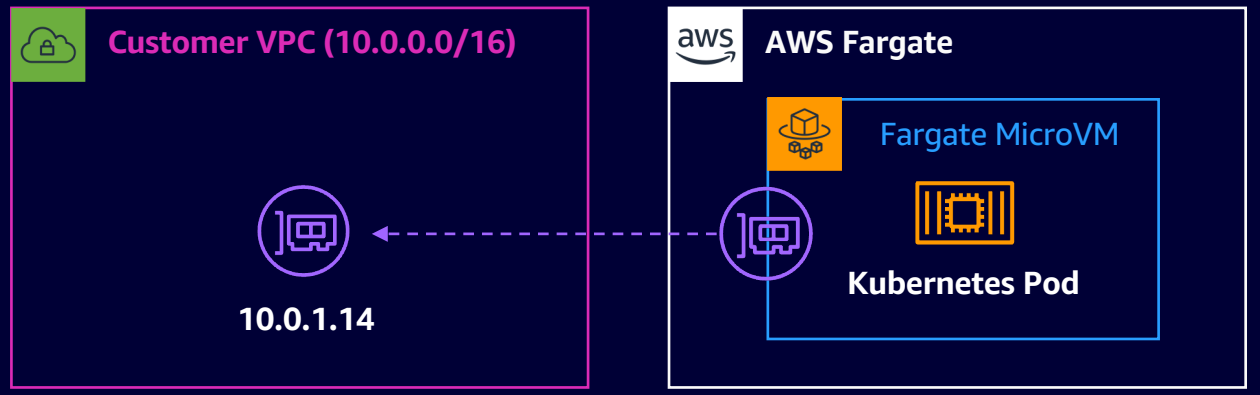


- **EKS AMI Build Scripts**
- https://github.com/awslabs/amazon-eks-ami
- Source of truth for Amazon EKS Optimized AMI
- Easily build your own Amazon EKS AMI
- Build assets for Amazon EKS AMI for each supported Kubernetes version

- **Bottlerocket**
- Purpose-built by Amazon Web Services for running containers on virtual machines or bare metal hosts
- Includes only the essential software to run containers, which improves resource usage and reduces the attack surface
- https://aws.amazon.com/bottlerocket/

AWS Fargate Runtime: Overview

AWS Fargate



- Secure isolation of Tasks by running a dedicated kernel without shared CPU, memory, or storage.
- Right—sized instances based CPU and Memory requests within the pod specification.
- AWS manages the patching and upgrading of the underlying infrastructure, you manage the container

Considerations

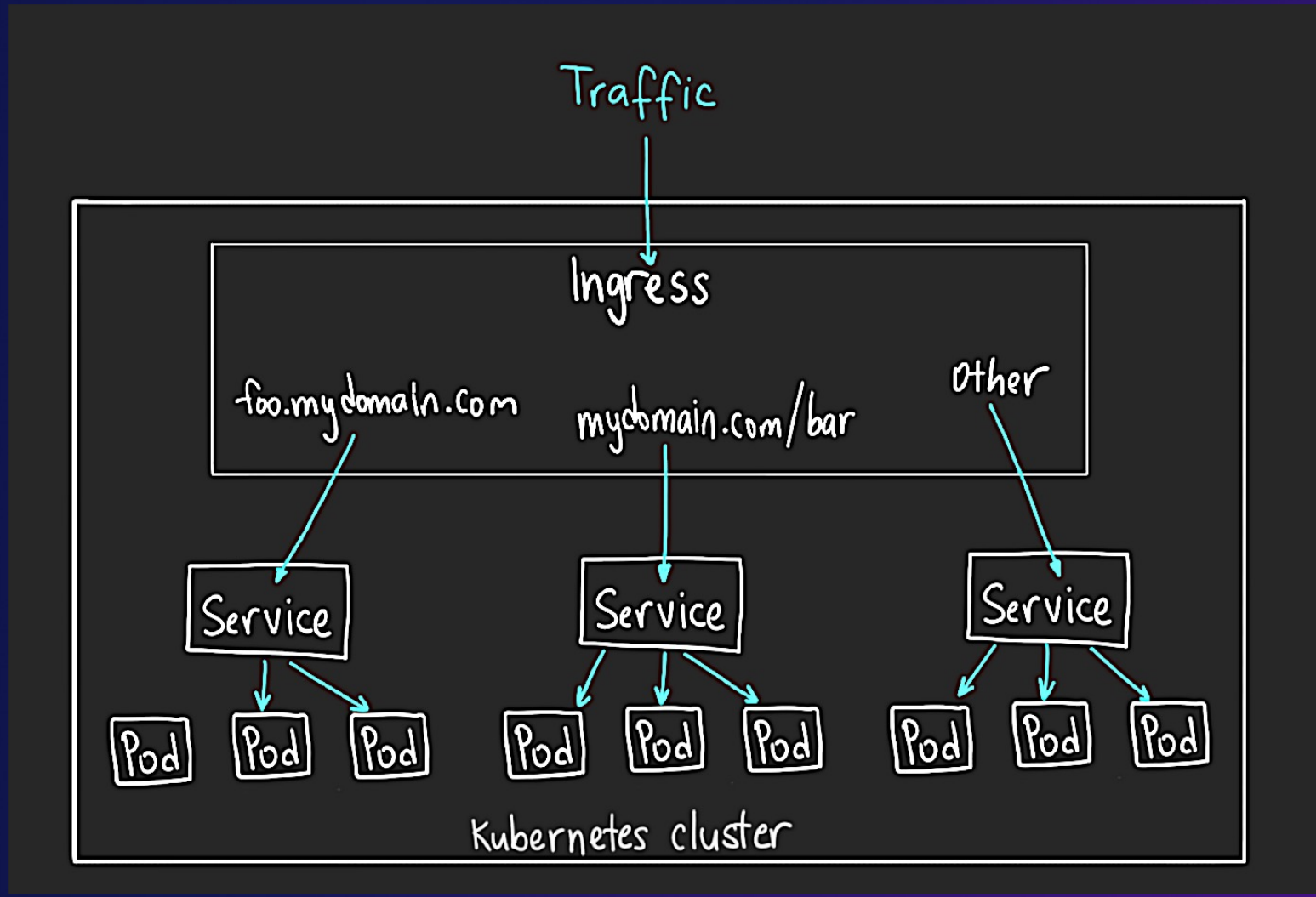
- DaemonSets are not supported
- Only supports Application Load Balancers
- Privileged Containers are Not Supported
- Do not support host networking
- Can only run within private subnets

aws

Kubernetes Ingress and The AWS Load Balancer Controller

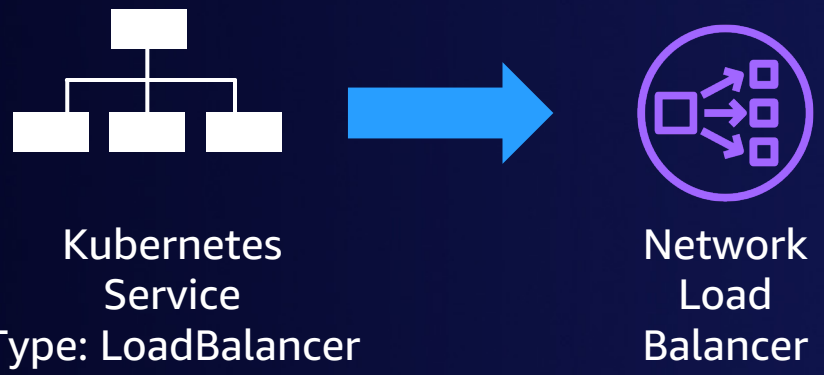
Kubernetes Ingress object

- Exposes HTTP/HTTPS routes to services within the cluster
- Many implementations: ALB, NGINX, F5, HAProxy etc.
- Default service type: ClusterIP



AWS Load Balancer Controller

A KUBERNETES CONTROLLER FOR ELASTIC LOAD BALANCERS

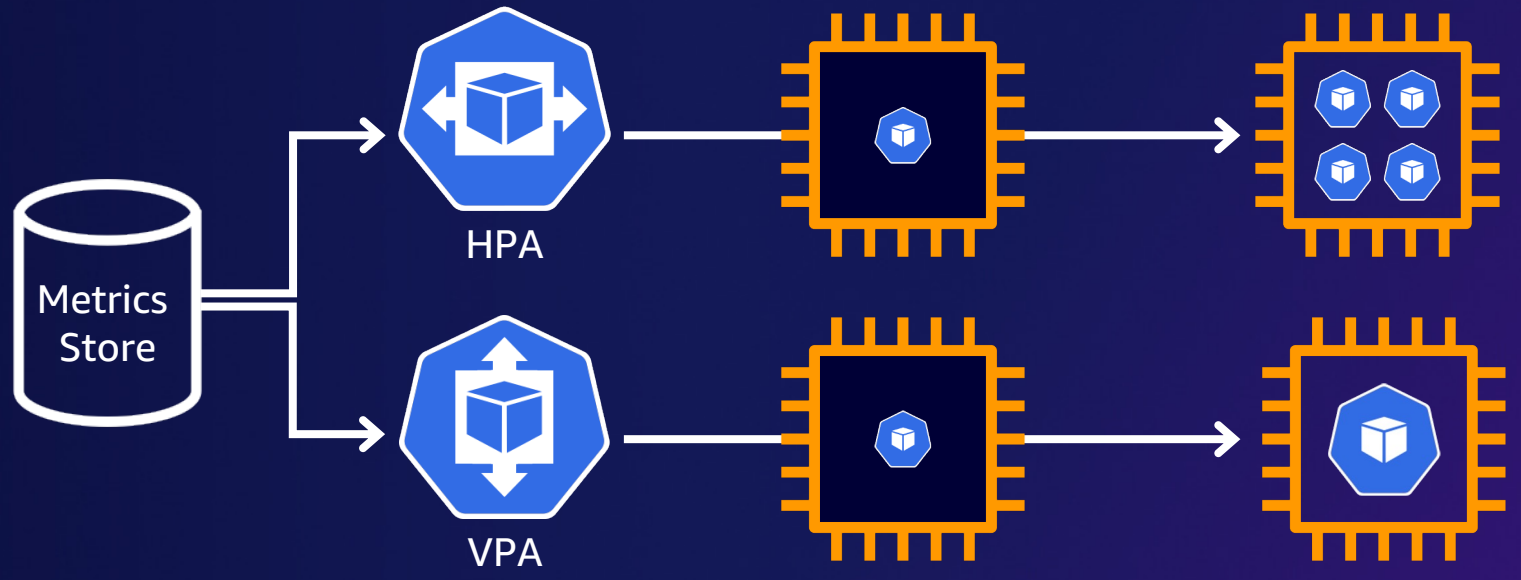


- Manages AWS Elastic Load Balancing for an Amazon EKS or a Kubernetes cluster
- Uses standard Kubernetes resources
  - v1: Service (Type: LoadBalancer)
  - networking.k8s.io/v1: Ingress
  - networking.k8s.io/v1: IngressClass
- Custom resources
  - elbv2.k8s.aws/v1beta1: TargetGroupBinding
- Supports latest ELB features

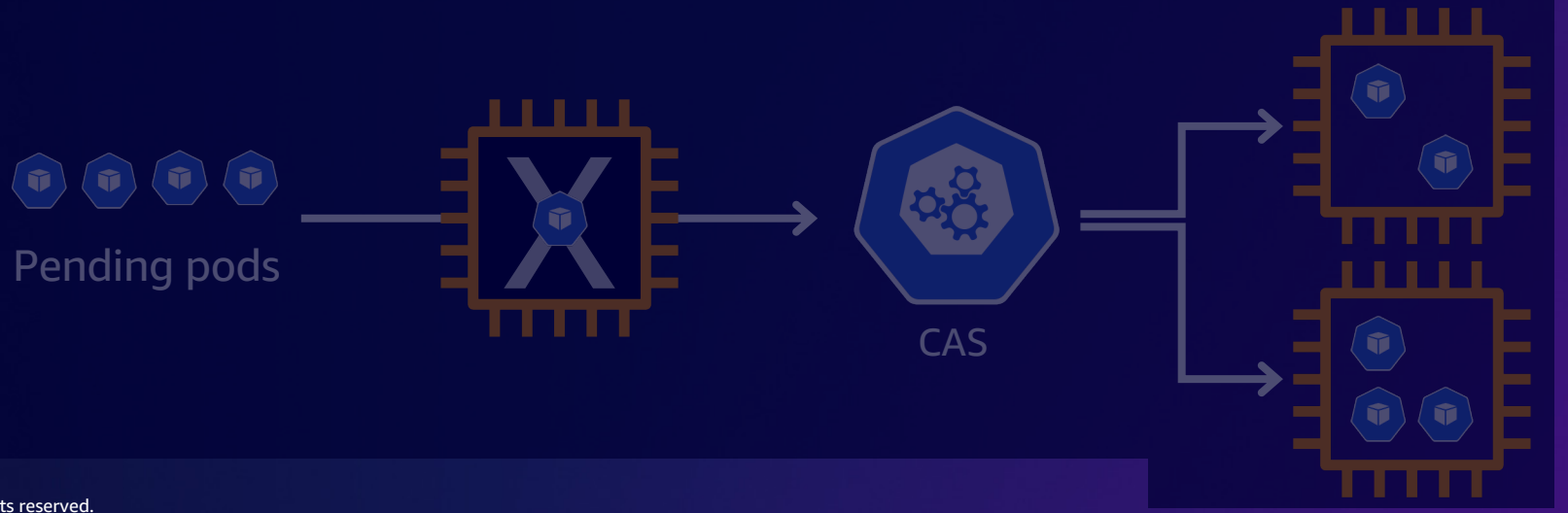
Scaling applications with Kubernetes

Kubernetes Autoscaling

1. Horizontal Pod Autoscaling (HPA)
2. Vertical Pod Autoscaling (VPA)

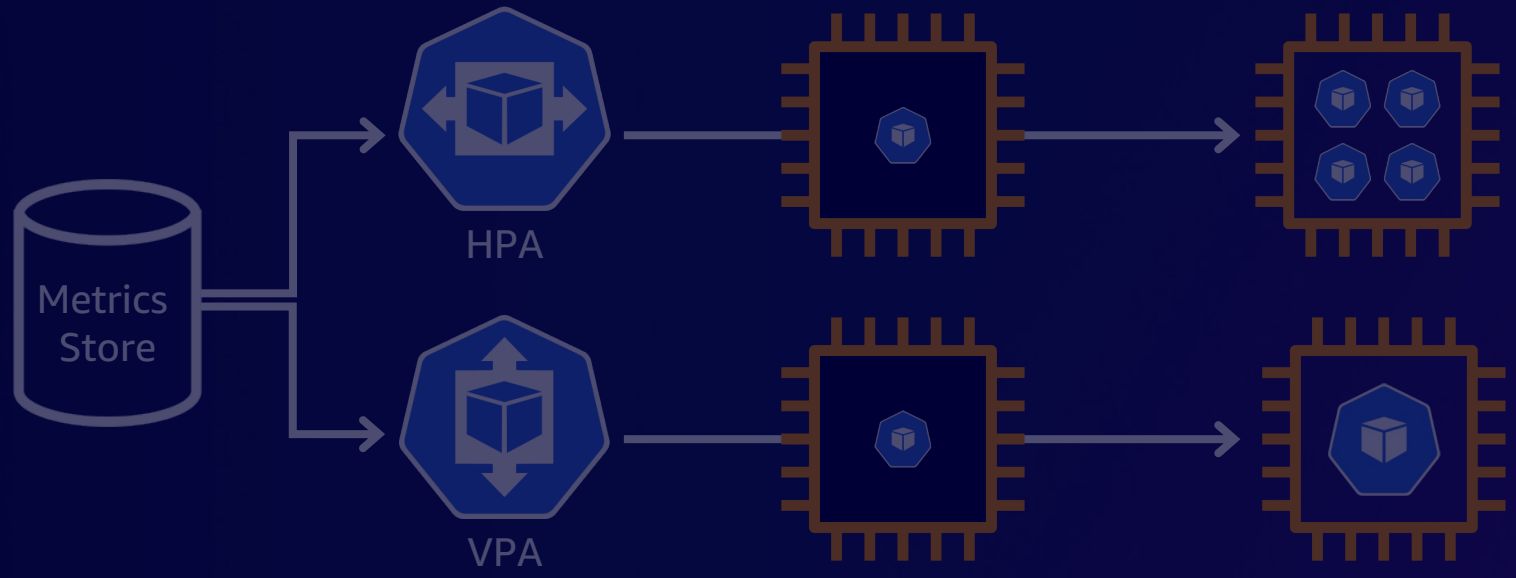


3. Cluster Autoscaler (CAS)

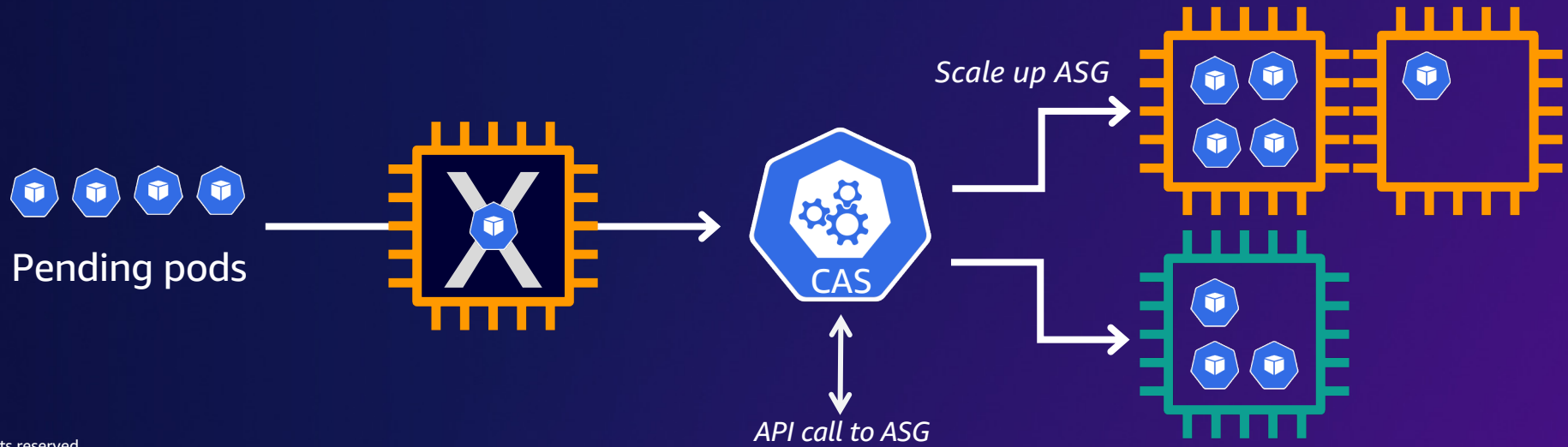


Kubernetes Autoscaling

1. Horizontal Pod Autoscaling (HPA)
2. Vertical Pod Autoscaling (VPA)



3. Cluster Autoscaler (CAS)



What is Karpenter?

Karpenter is an ***open-source***, ***flexible***, and ***high-performance*** Kubernetes cluster autoscaler.

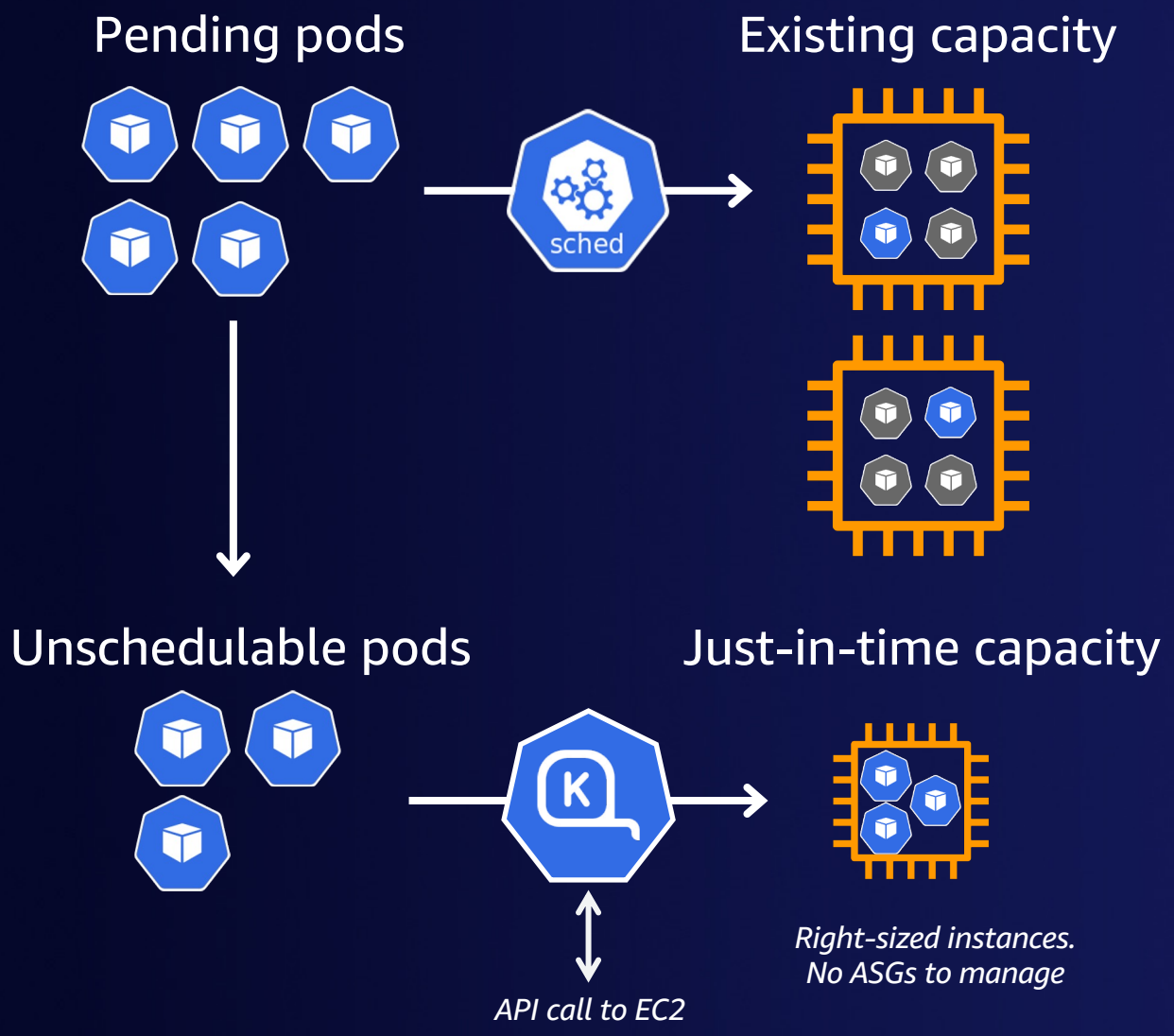
Open source and Kubernetes-native

Dynamic, group-less node provisioning

Automatic node sizing

Rapid scaling

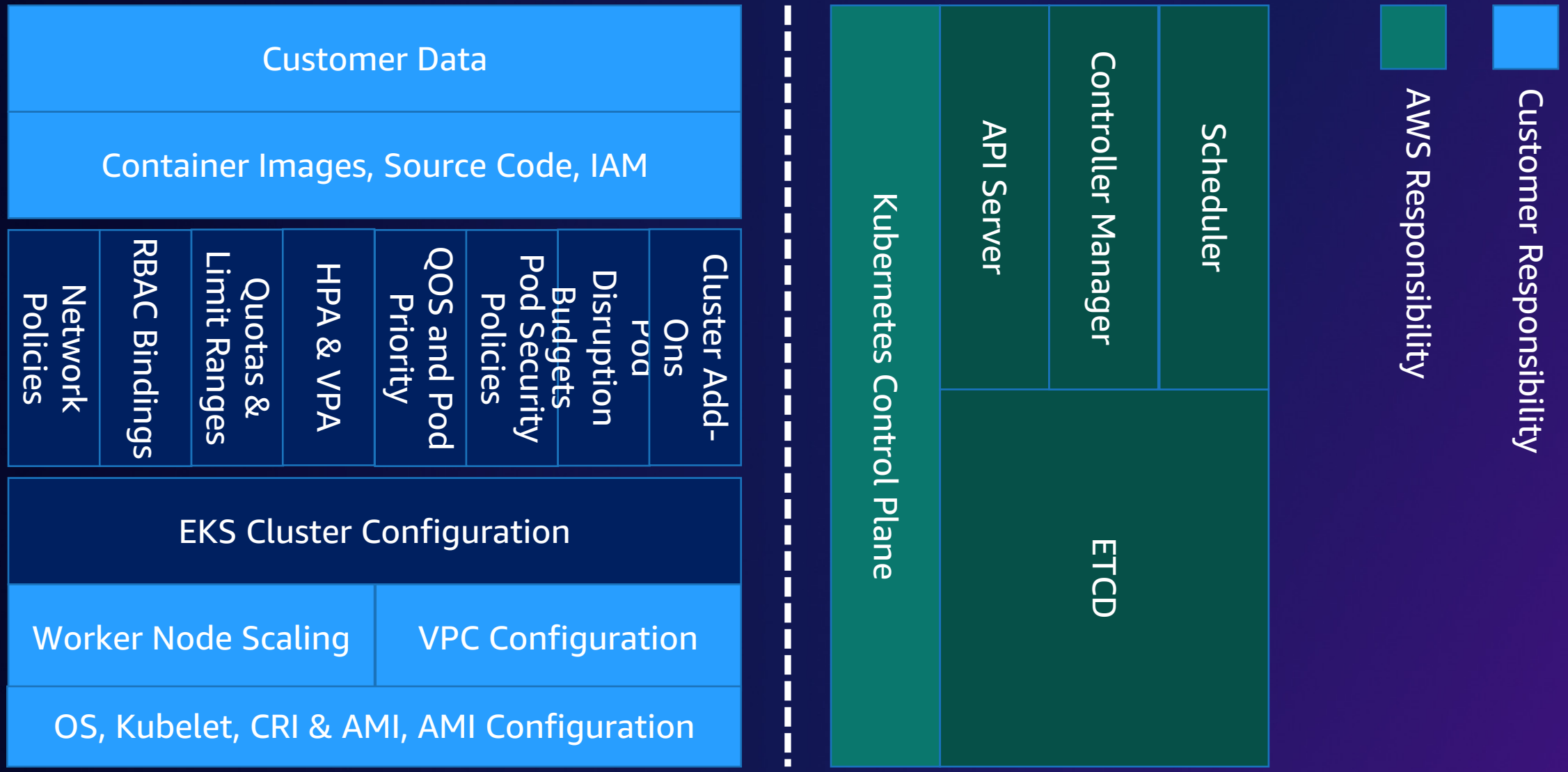
How Karpenter Works



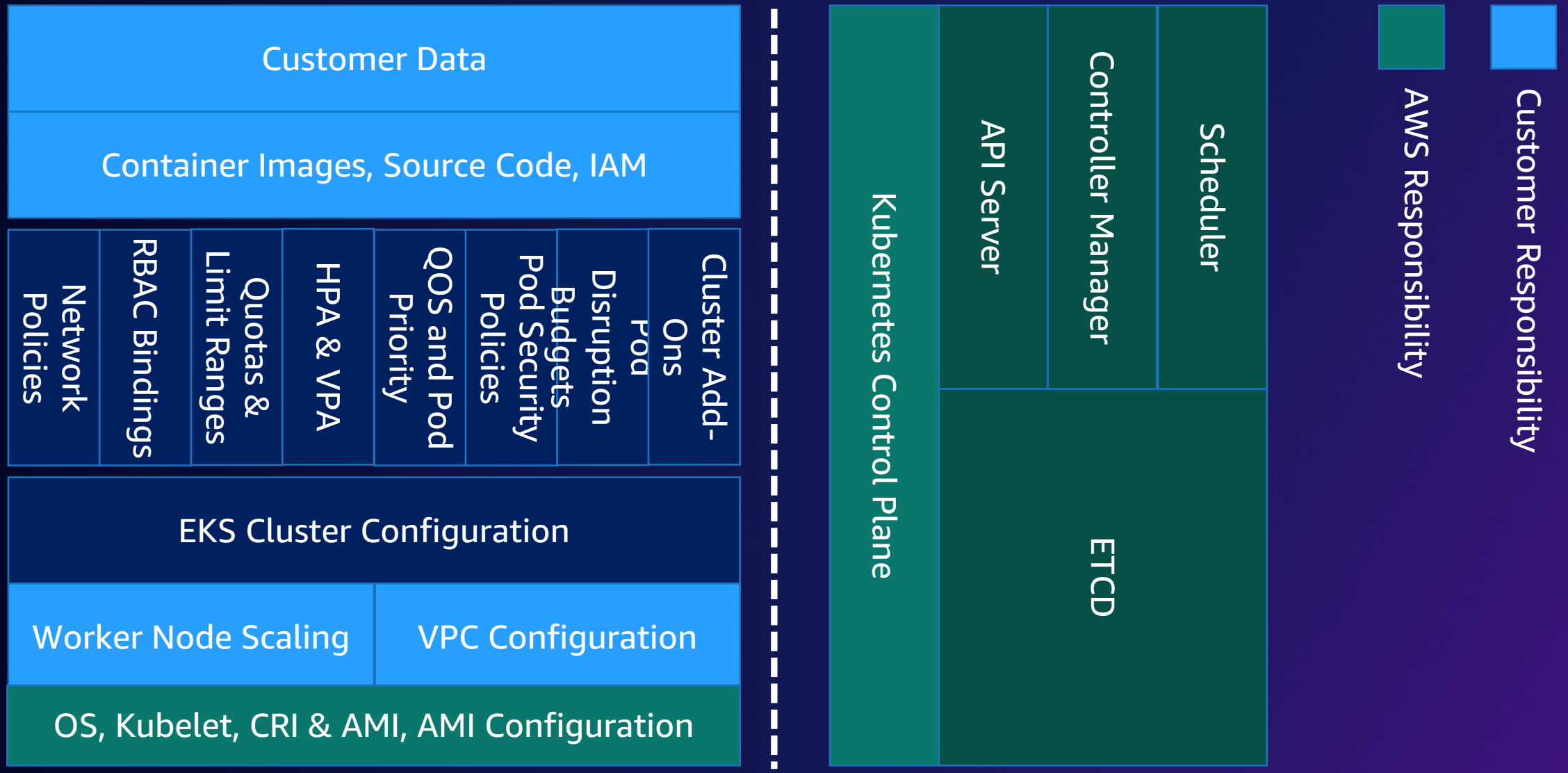
- Deeply integrated with EC2
  - EC2 Fleet API, no ASGs
- Deeply Kubernetes native
  - Watch API, Labels, Finalizers
- Automated instance selection
  - Matches workload needs to instance type based on the Provisoner profile
- Karpenter terminates underutilized nodes

Kubernetes Security

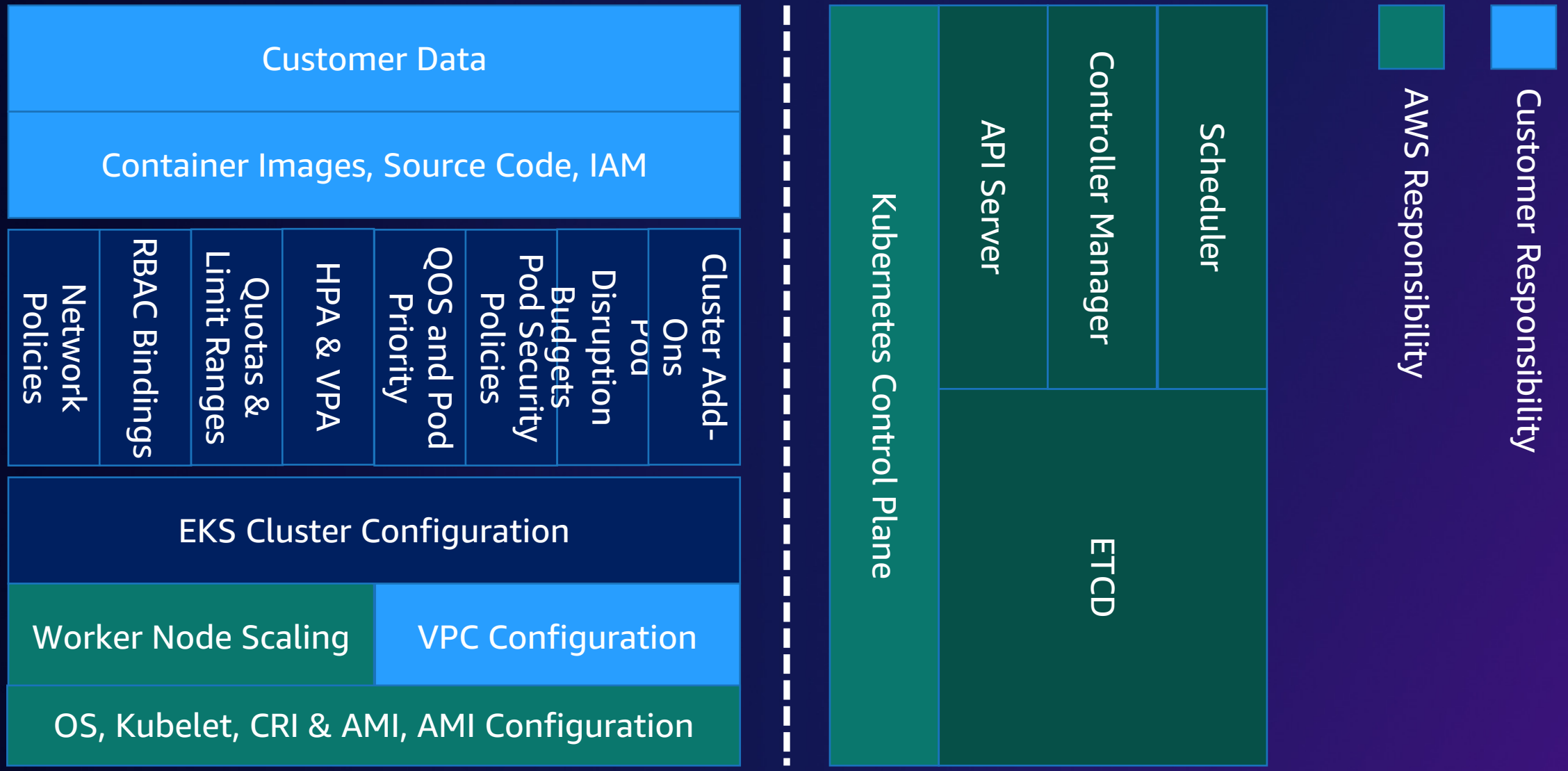
EKS – Self Managed Workers



EKS – Managed Node Groups



EKS – Fargate



Event-driven use cases

Asynchronous processing

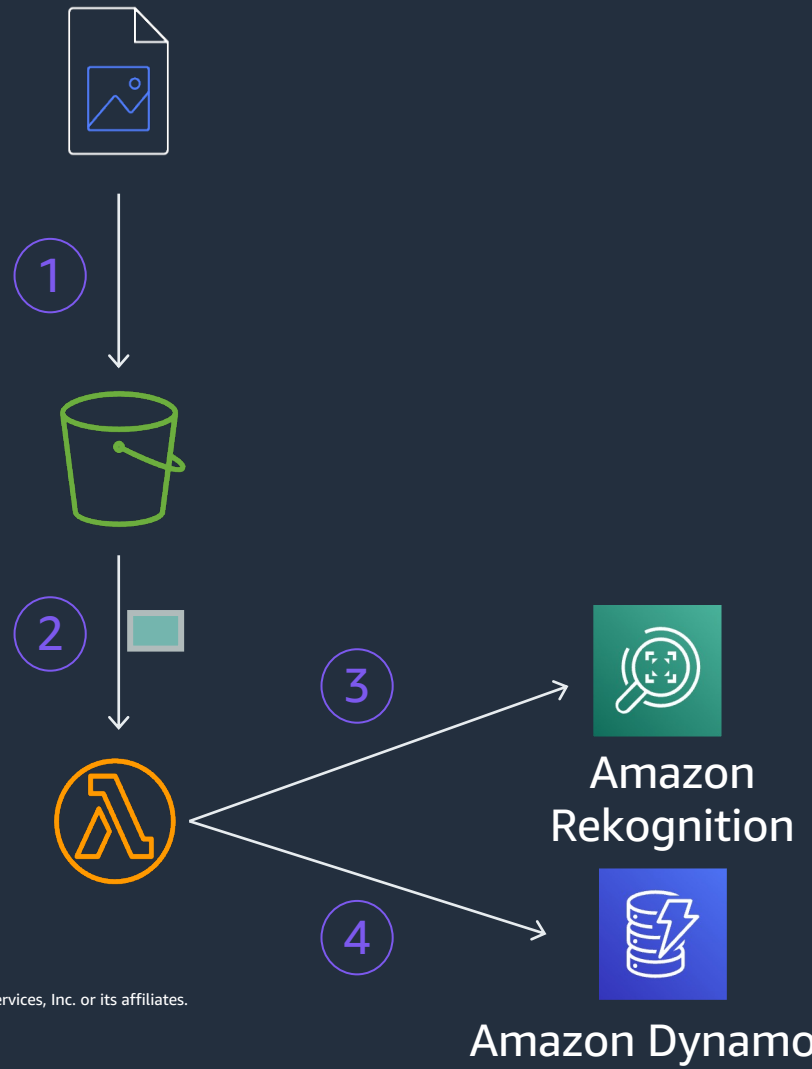
Processing file uploads

RESIZE PHOTO, EXTRACT TEXT, TRANSLATE, ETC.

1. Object uploaded to Amazon S3 Bucket
2. Asynchronous invoke of Lambda function, event payload includes:
   - Bucket name
   - Object key

Processing file uploads, quickly add new functionality

ADD IMAGE ANALYSIS AND METADATA STORAGE

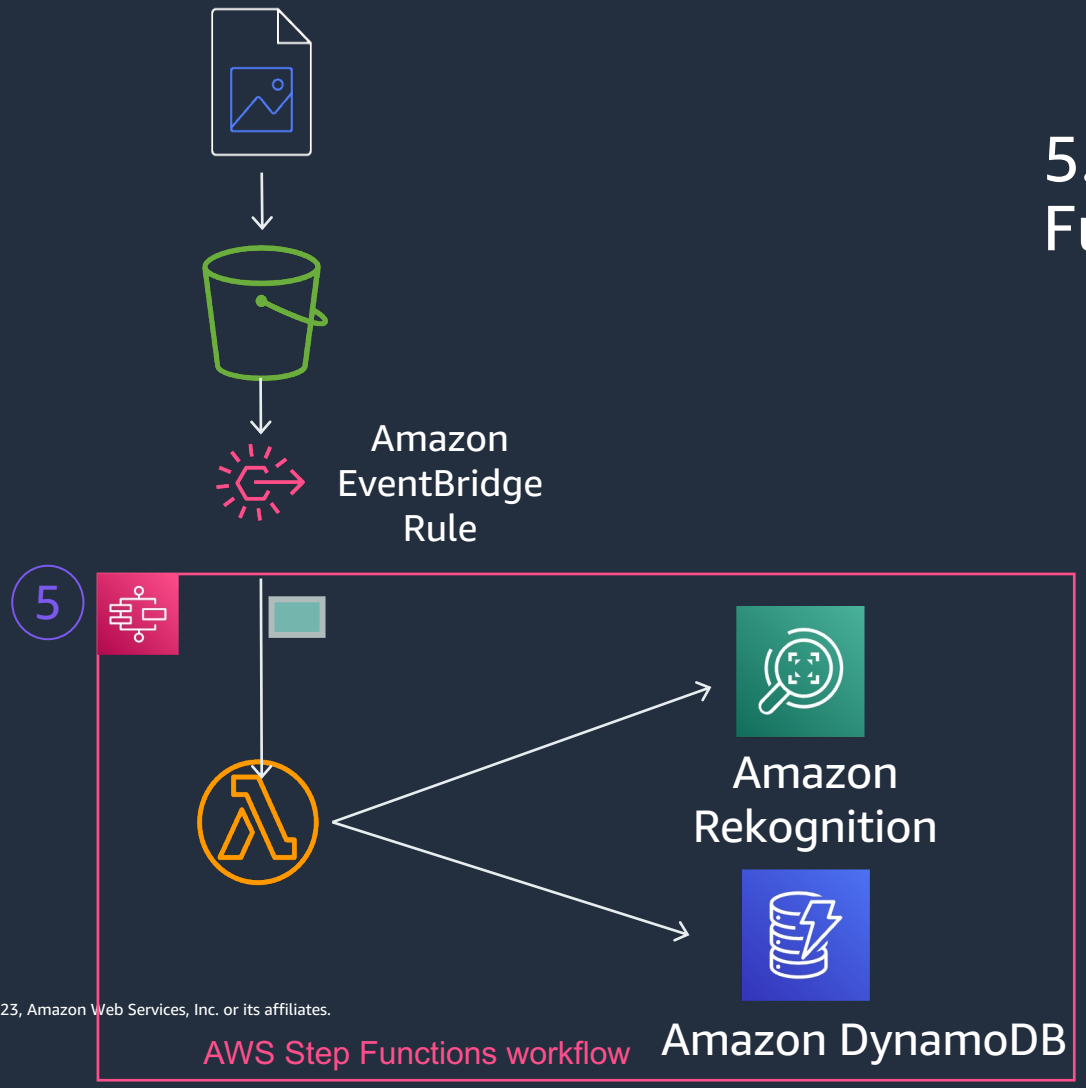


3. Analyze photo with Amazon Rekognition

4. Store image details and results of analysis

Processing file uploads, quickly add new functionality

ADD IMAGE ANALYSIS AND METADATA STORAGE



5. Simplify business logic with Step Functions

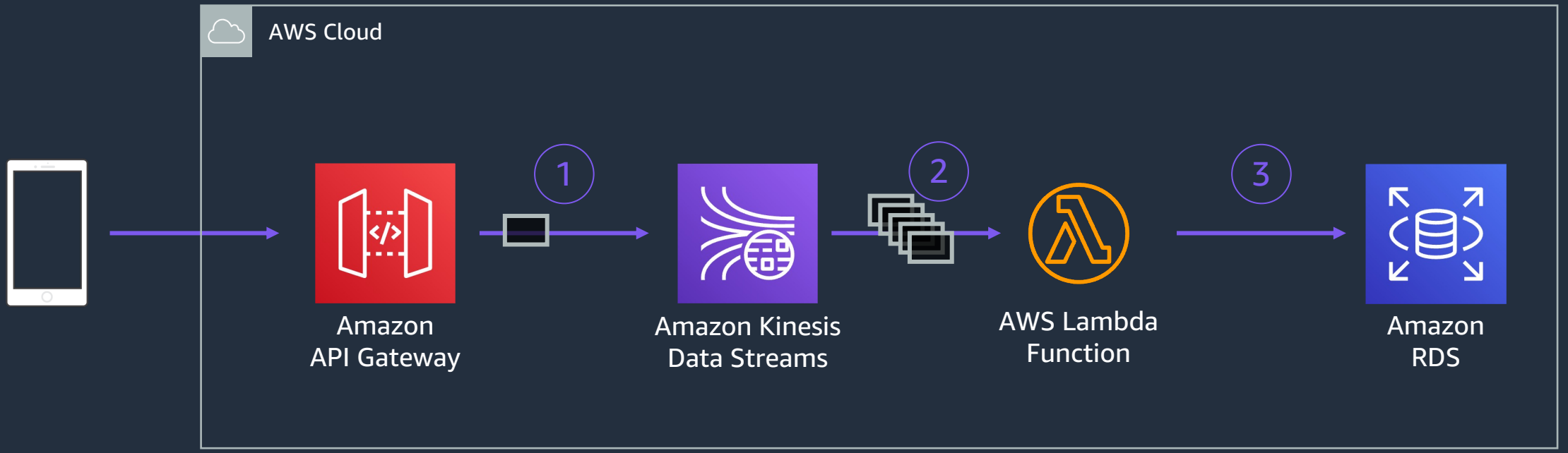
Streaming data ingestion and storage

CONSUME, PROCESS, AND STORE DATA

1. Lambda service polls Kinesis Data Stream for messages
2. Function is synchronously invoked with batches of messages
3. Function processes and/or pushes data to downstream data stores

“Semi-Serverless” Webhook

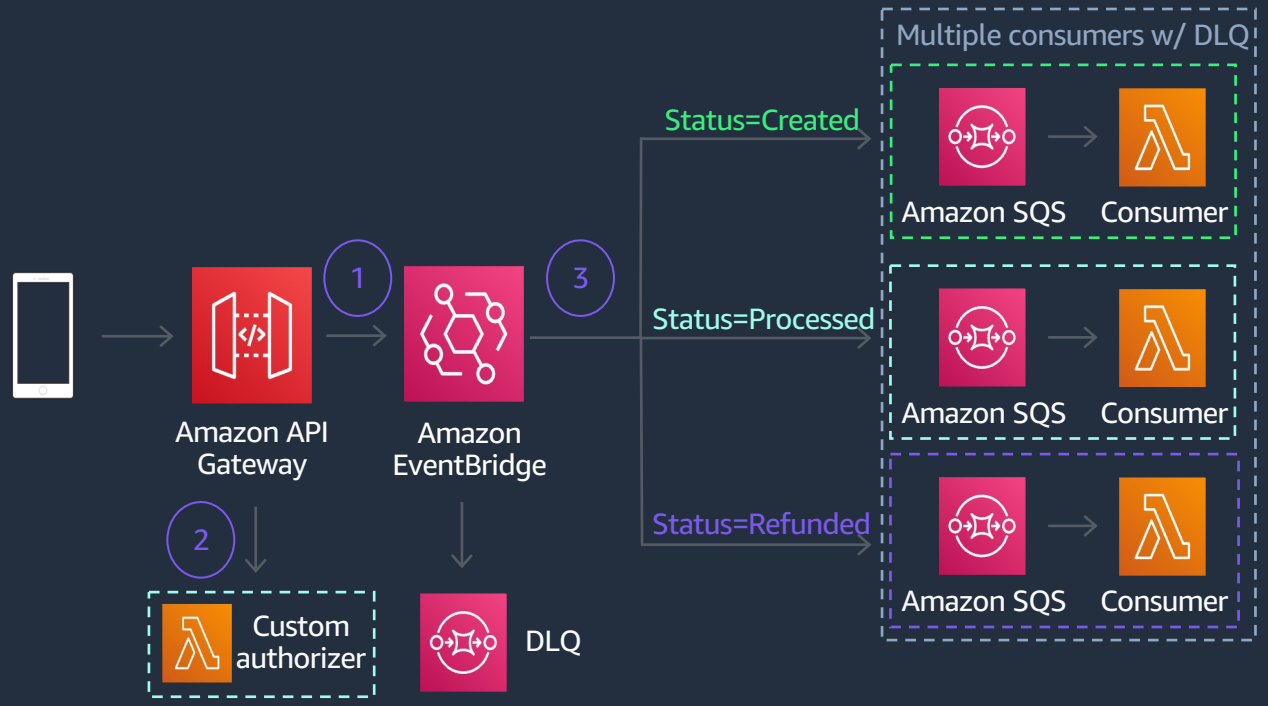
SCALABLE, RESILIENT. BUFFER DOWNSTREAM CONCURRENCY.



1. “Storage-first” pattern: API Gateway writes directly to Kinesis Data Stream
2. Kinesis as a buffer to limit concurrency downstream
3. Perform transaction(s) on batch

Fan out

PUSH UPDATES TO MULTIPLE SUBSCRIBERS

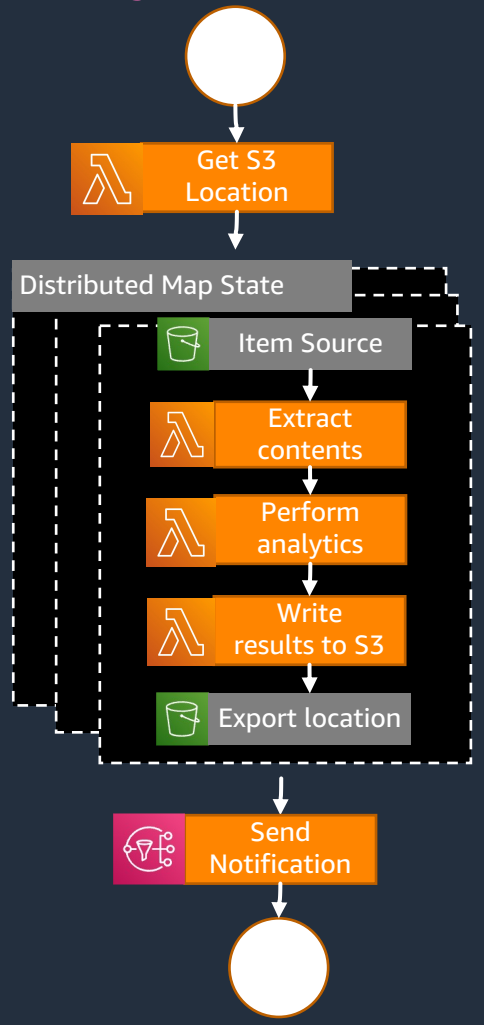


1. “Storage first”: integrate API Gateway directly to EventBridge
2. Enforce authorization
3. Use routing for efficient processing

Large scale data processing with AWS Step Functions

DYNAMICALLY PROCESS LARGE ARRAYS OF DATA WITH DISTRIBUTED MAP STATE

- Coordinate large-scale parallel workloads in Step Functions
- Iterate over millions of S3 objects such as JSON or CSV files
- Up to 10,000 parallel executions
- Invoke Lambda or ECS/Fargate for large scale on demand serverless compute



Integrate third party applications with AWS

EXAMPLE: INTEGRATE SALESFORCE WITH AWS

Designed for event driven architecture and real-time applications.