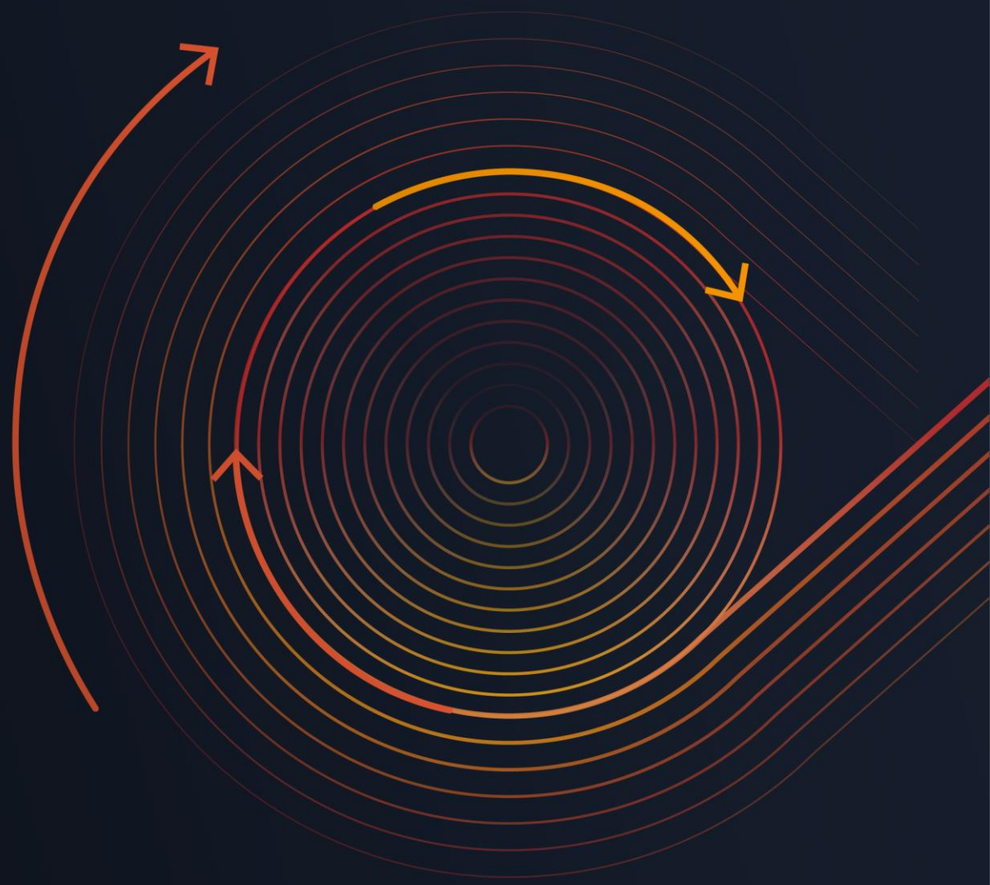
aws modern apps

Breaking Down the Monolith

Using Containers and Serverless

Nathan Peck, Senior Developer Advocate
Heeki Park, Principal Solutions Architect

Why containers?



aws modern apps

Applications aren’t just code, they have dependencies

Code

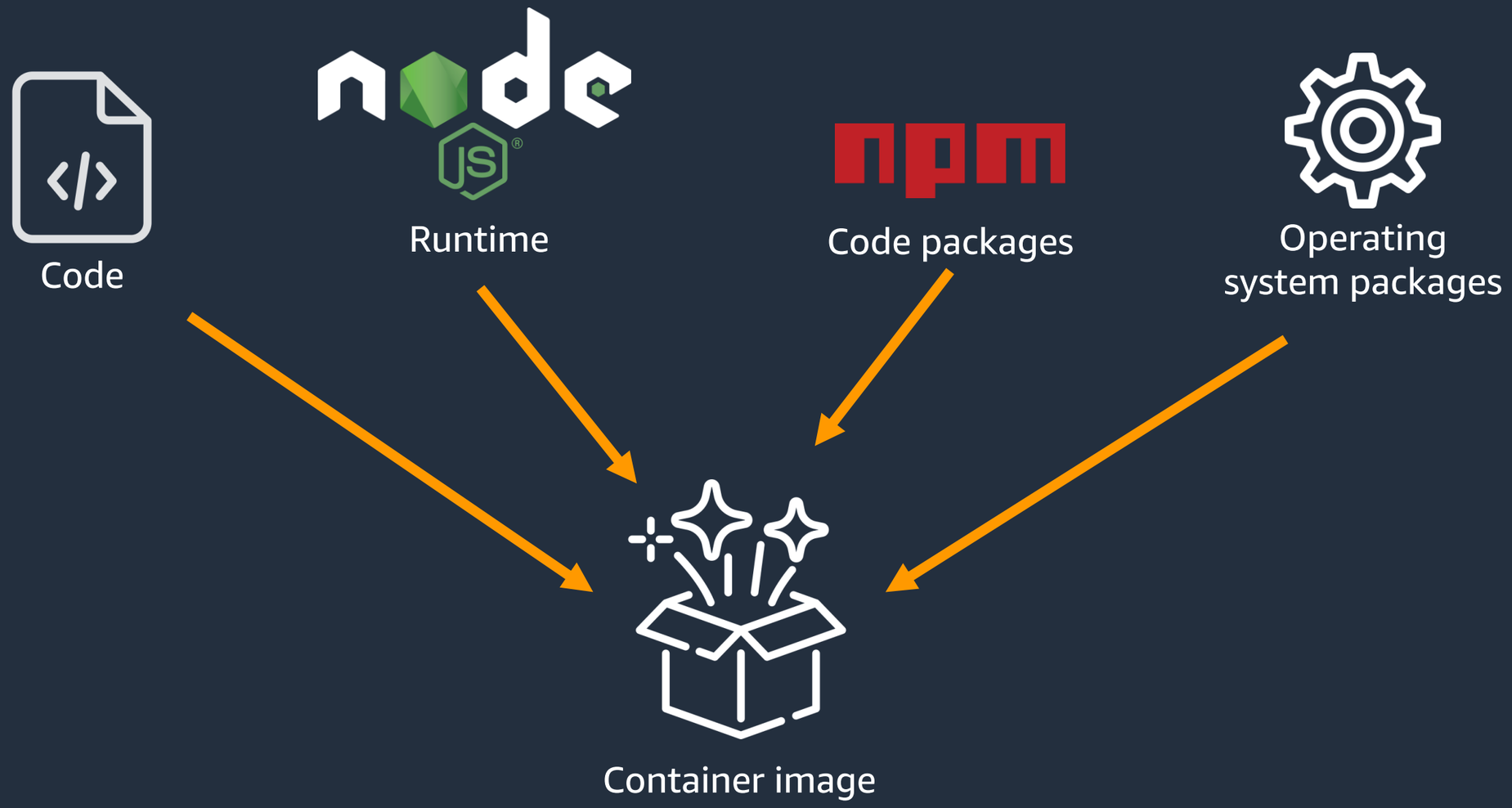
Runtime

Code packages

Operating system packages

aws modern apps

Containers turn applications into one deployable artifact



aws modern apps

Build

Gather the app and its dependencies. Create an immutable container image.

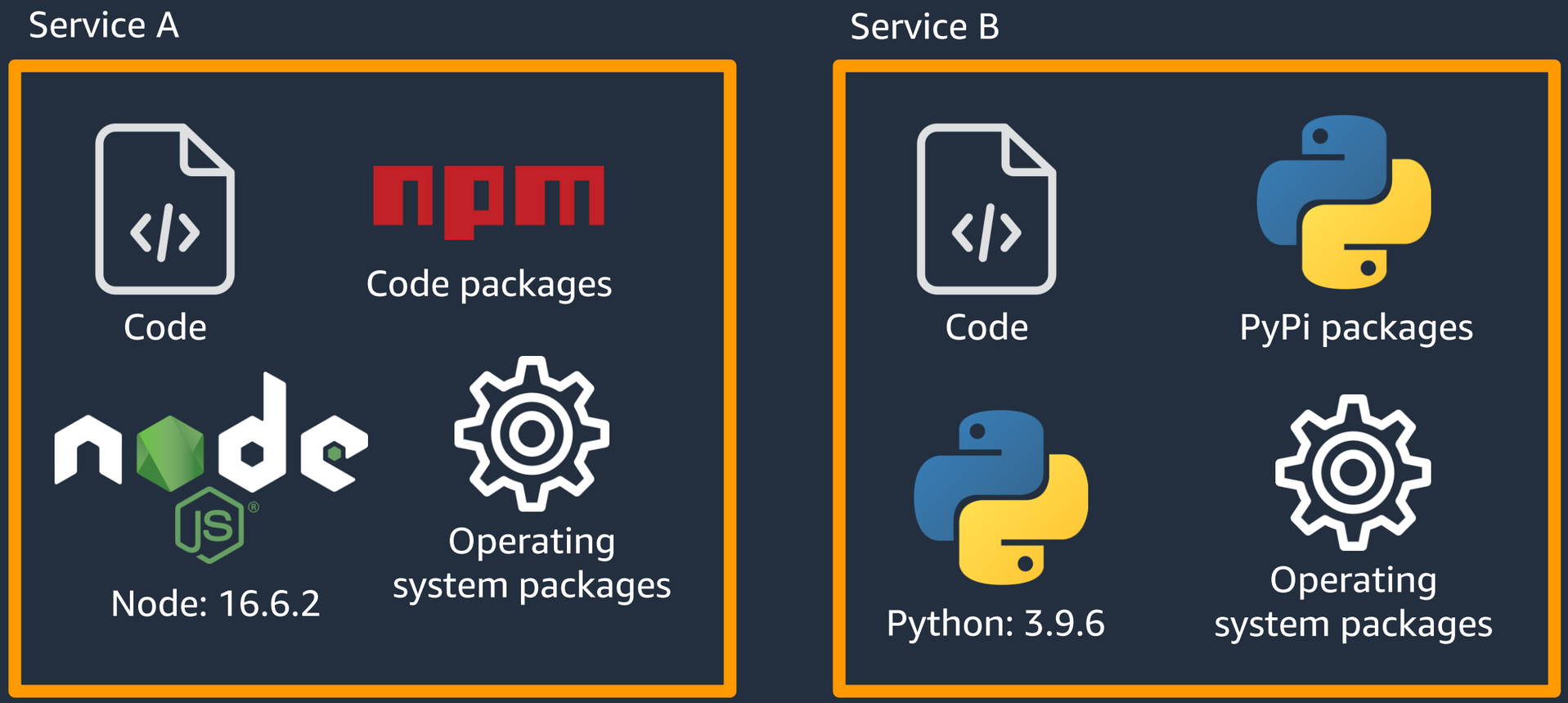
Push

Store the container image in a registry so it can be downloaded to compute

Run

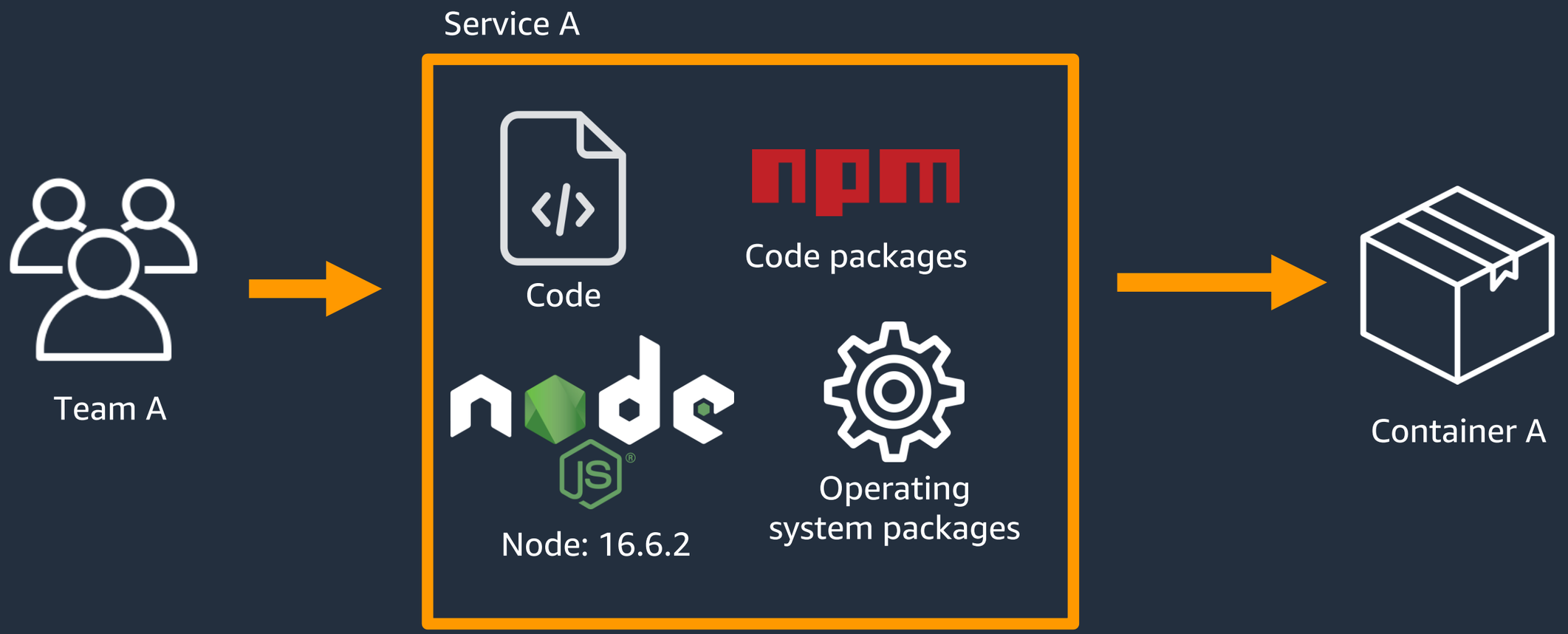
Download image to compute, unpack it, and run it in an isolated environment

Breaking a monolith is scary because more services mean more dependencies



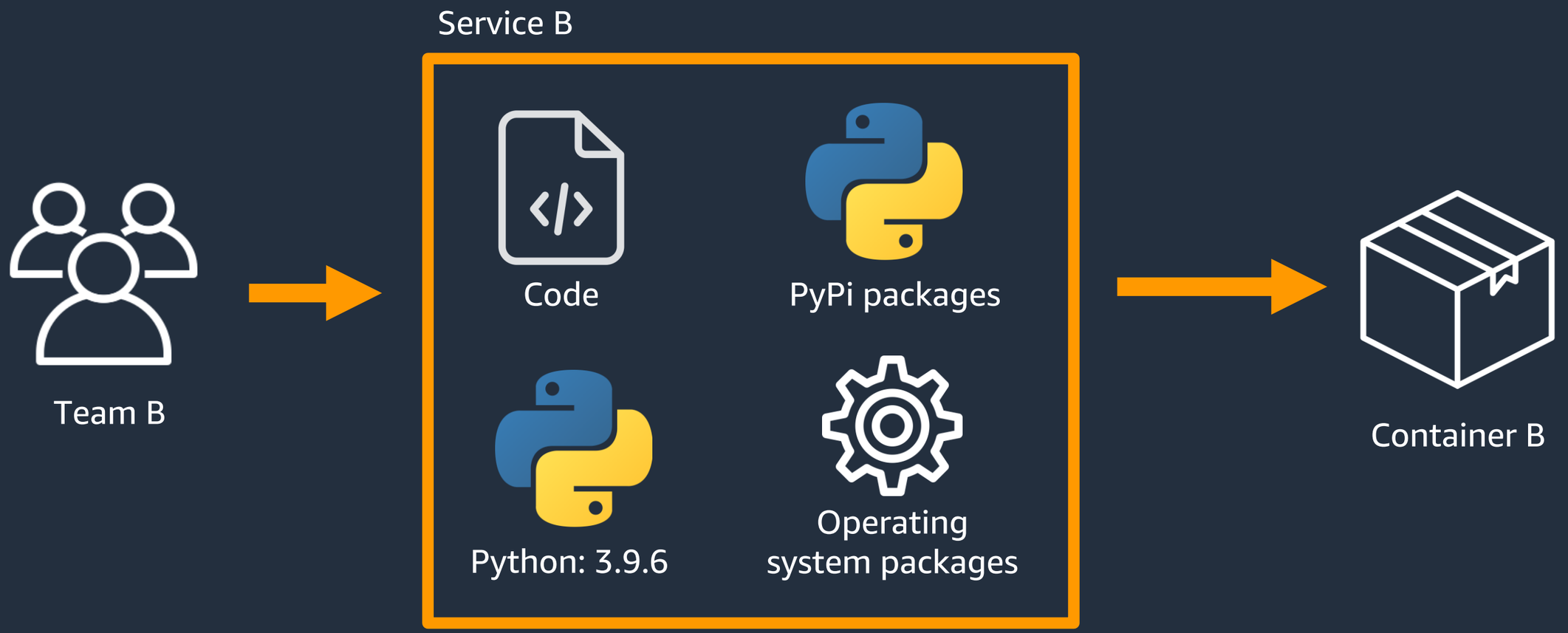
aws modern apps

Containers make dependencies a decentralized job



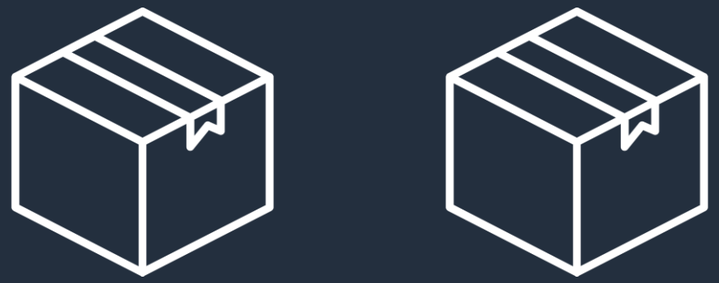
aws modern apps

Containers make dependencies a decentralized job



aws modern apps

Infrastructure is now agnostic to container contents

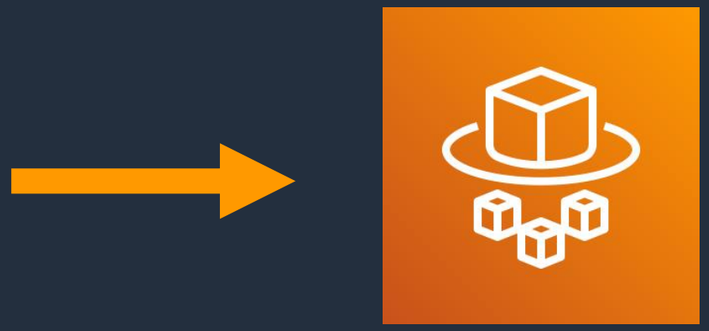


Container A

Container B

Container C

Container D

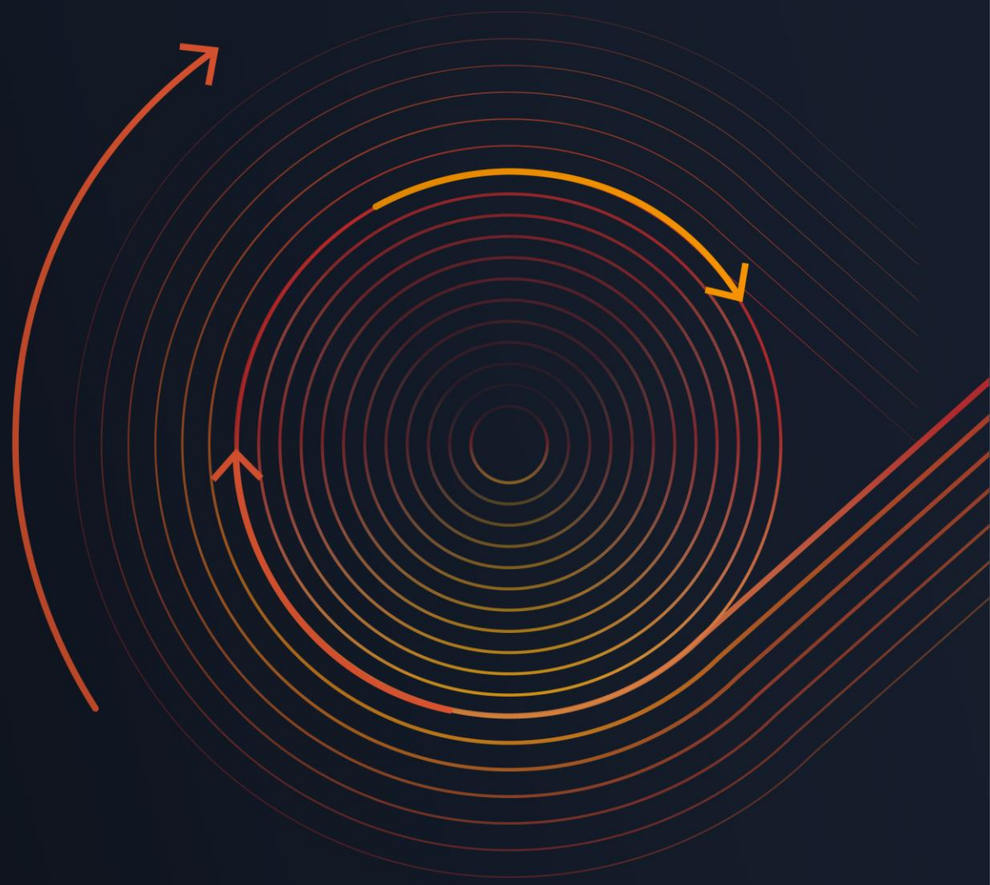


AWS Fargate

Different containers can be handled the same way using the same tooling

aws modern apps

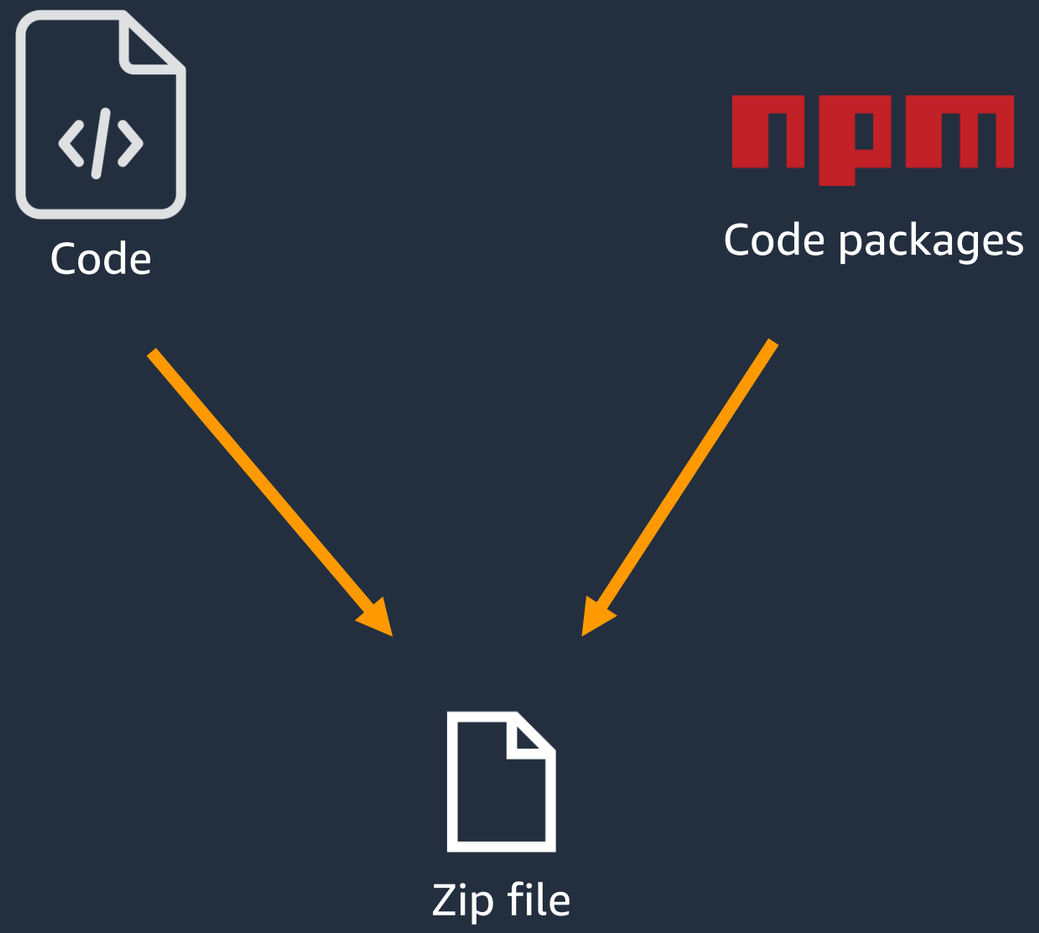
Why Serverless?



aws modern apps

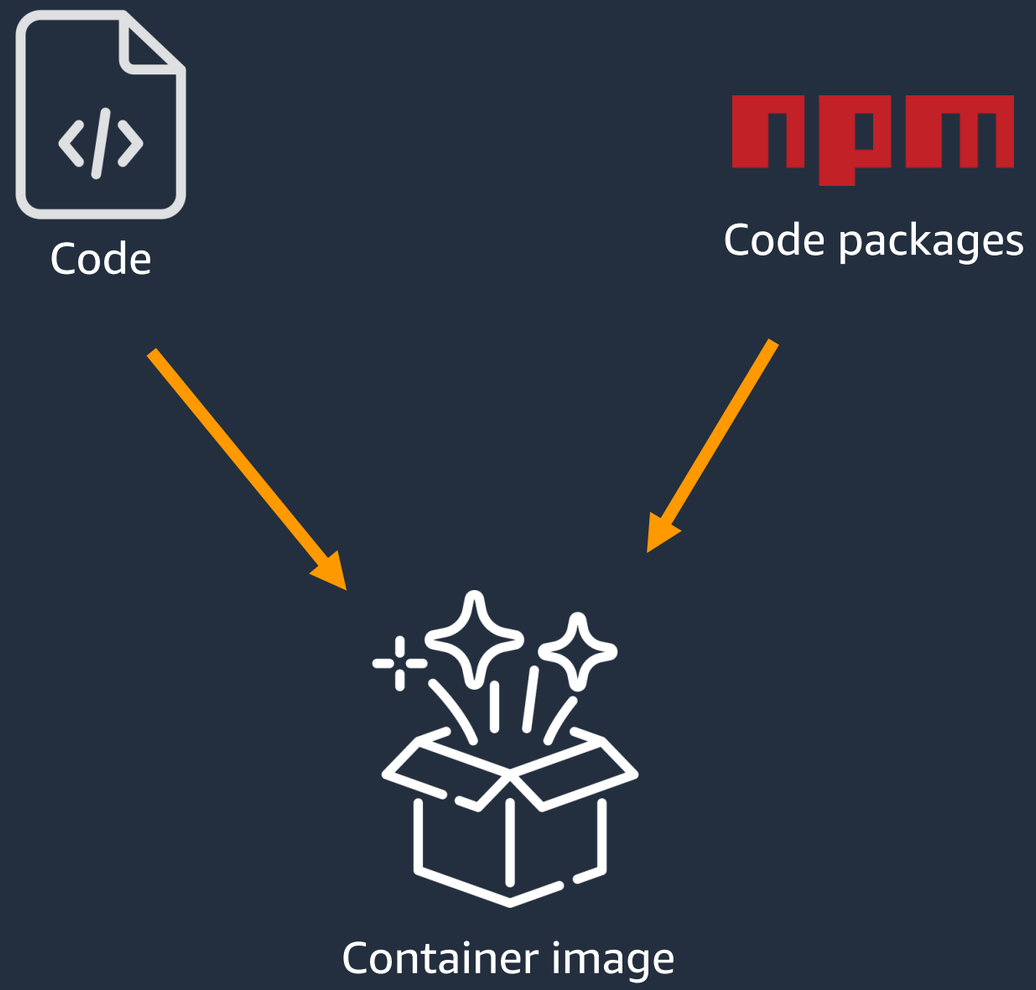
Serverless can help simplify the execution module

Applications can be packaged as a zip artifact



aws modern apps

Applications can also be packaged as a container artifact



aws modern apps

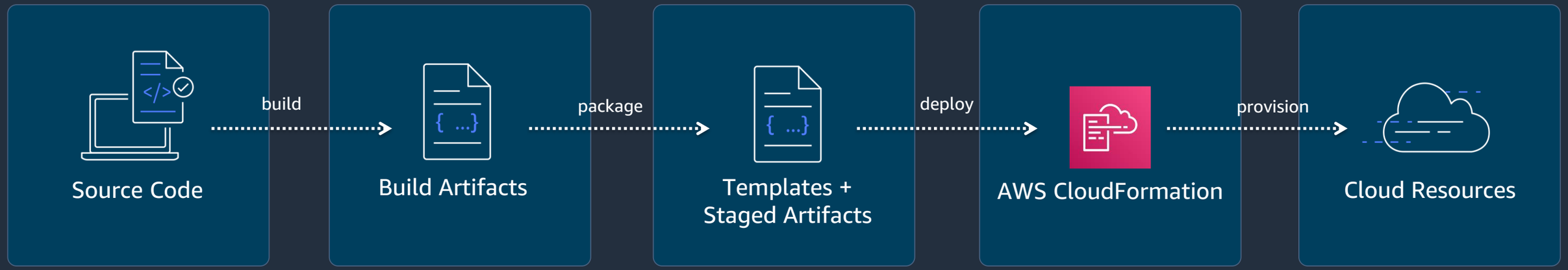
Package dependencies separately as Lambda Layers

Lets functions easily share code: Upload layer once, reference within any function

Promote separation of responsibilities, lets developers iterate faster on writing business logic

Built in support for secure sharing by ecosystem

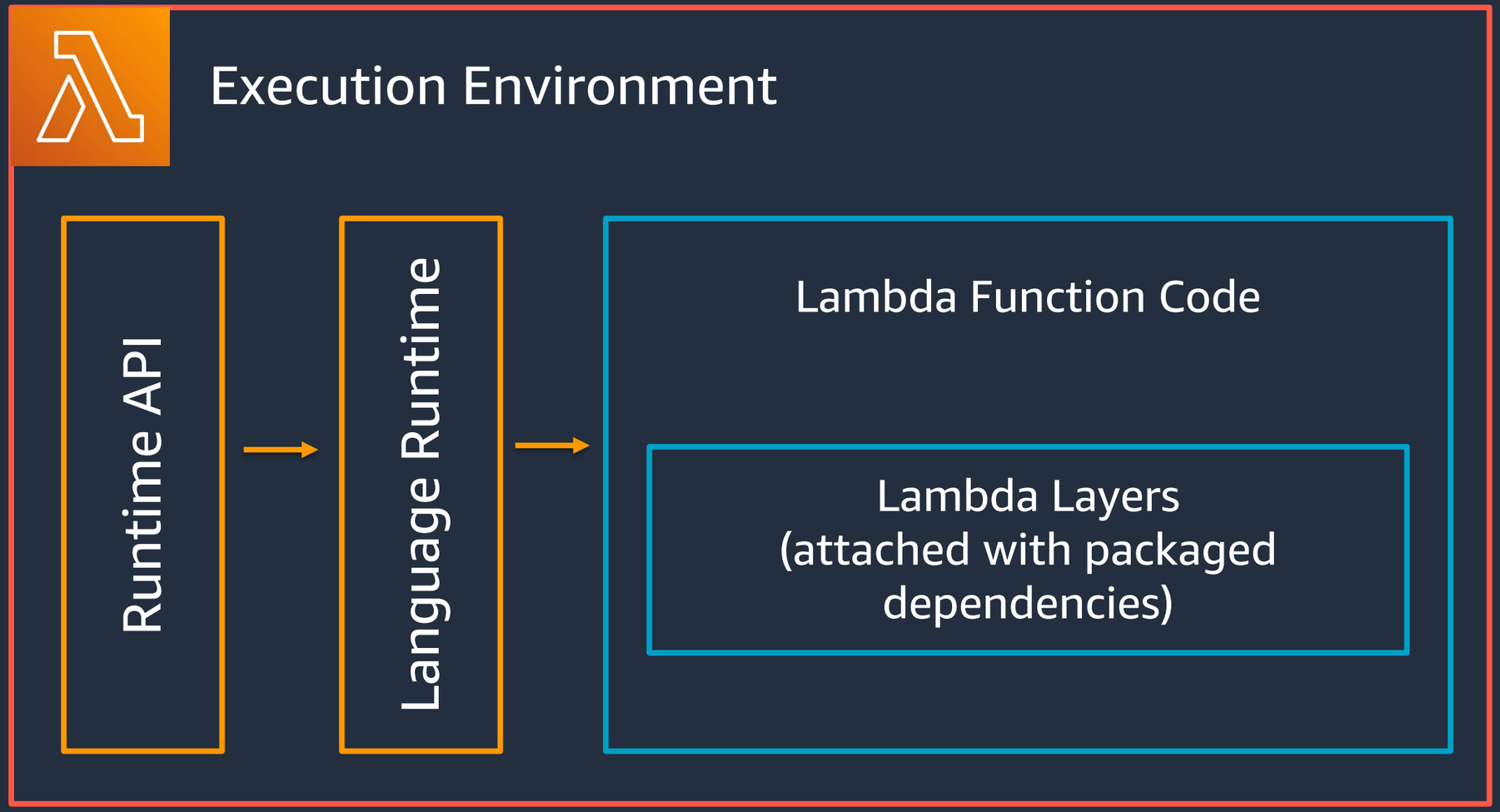
Deploy the code as resources




```
sam init              // create new project
sam build             // build artifacts
sam package     // generate templates and stage artifacts
sam deploy            // deploy resources
```

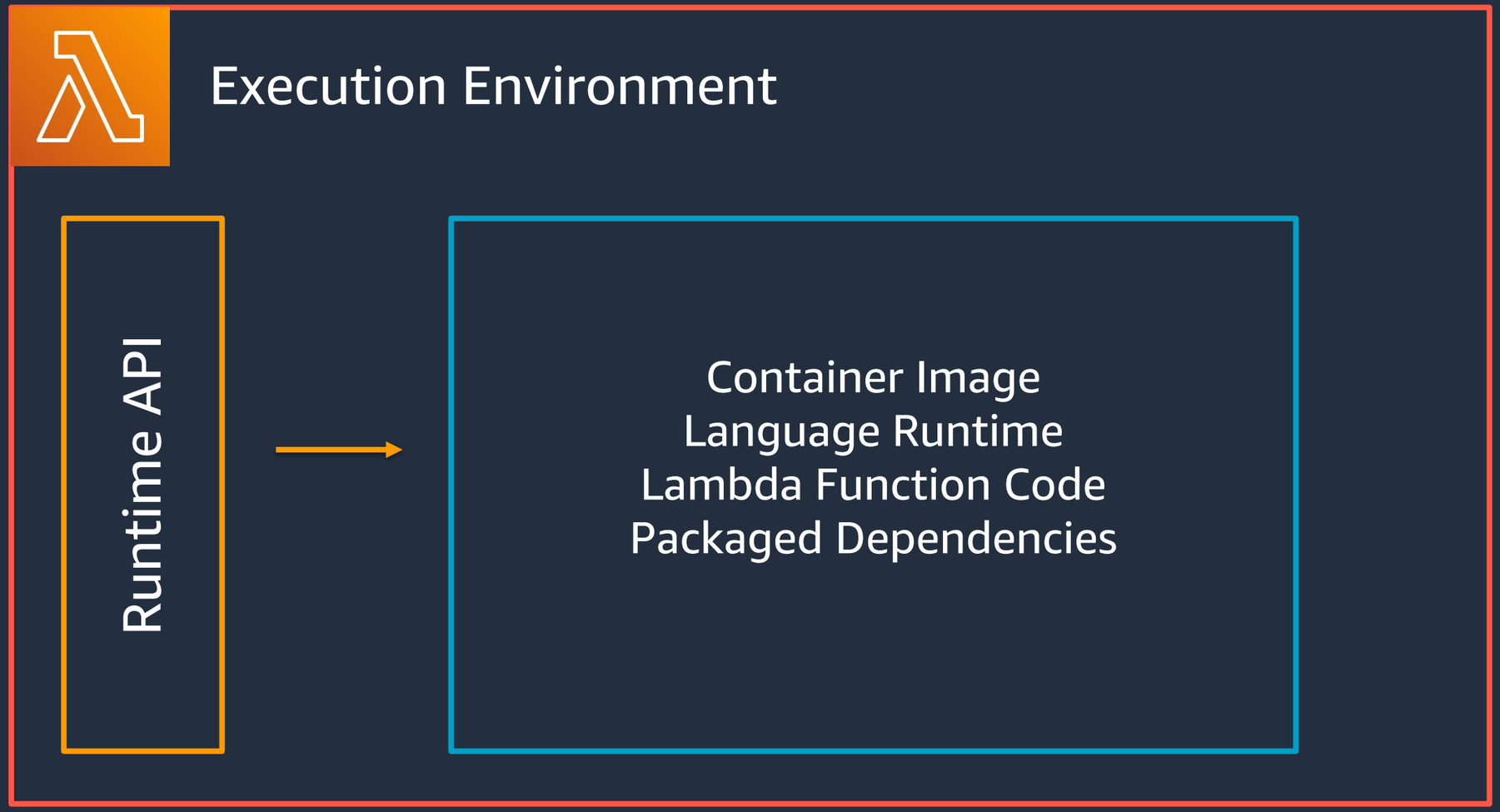

aws modern apps

Run the code as a zip artifact



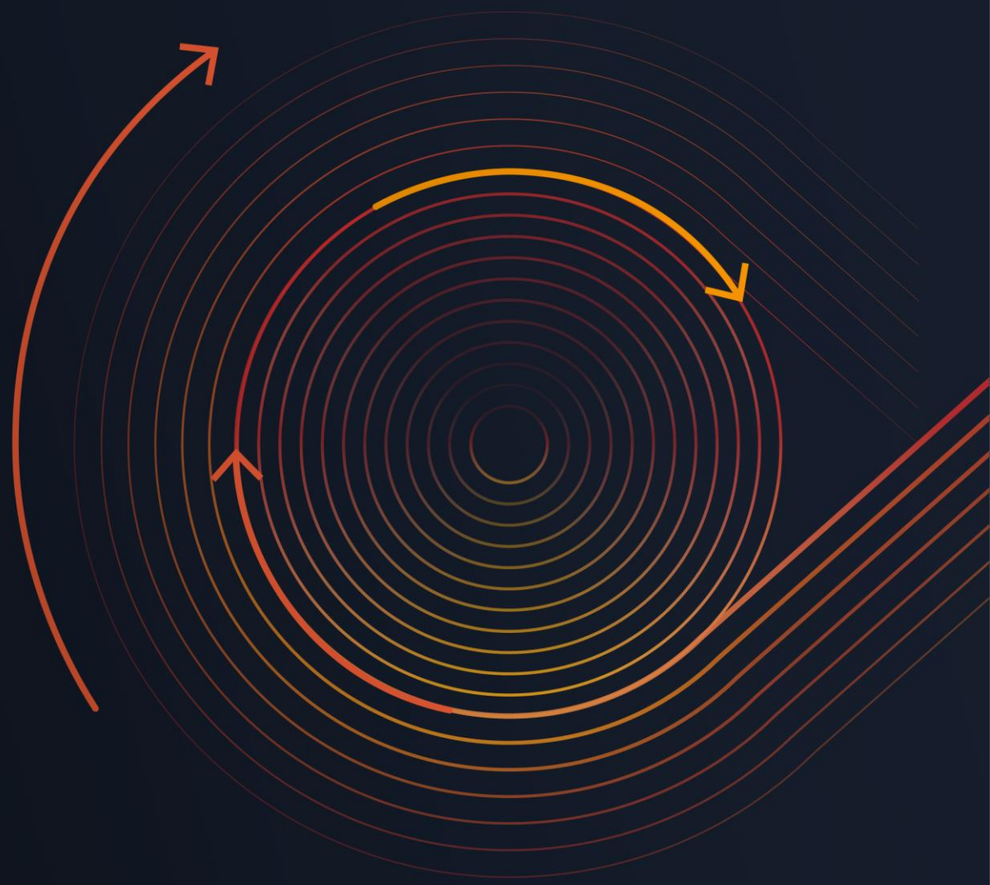
aws modern apps

Run the code as a container image



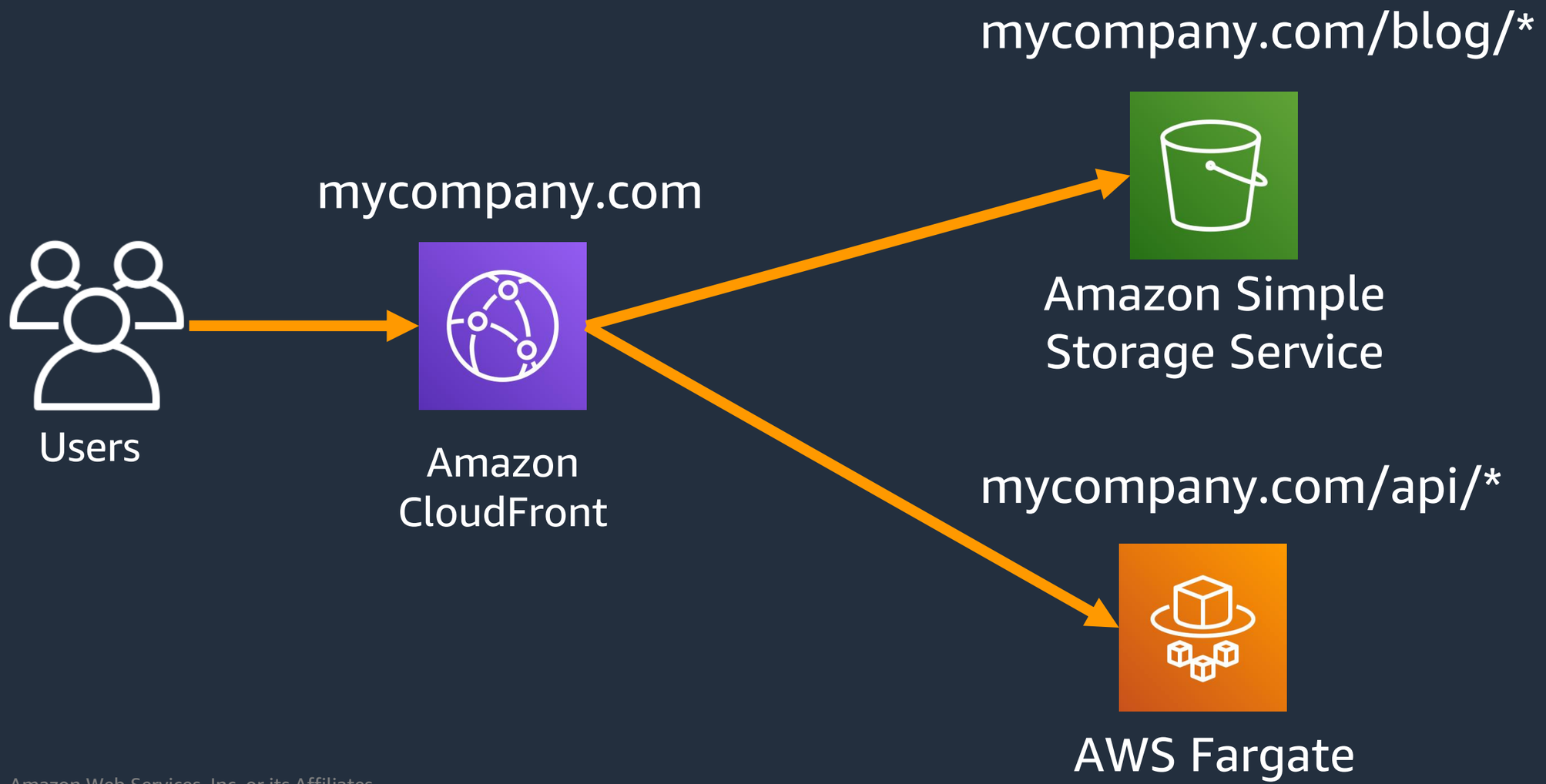
aws modern apps

Popular Decoupling Patterns with Containers



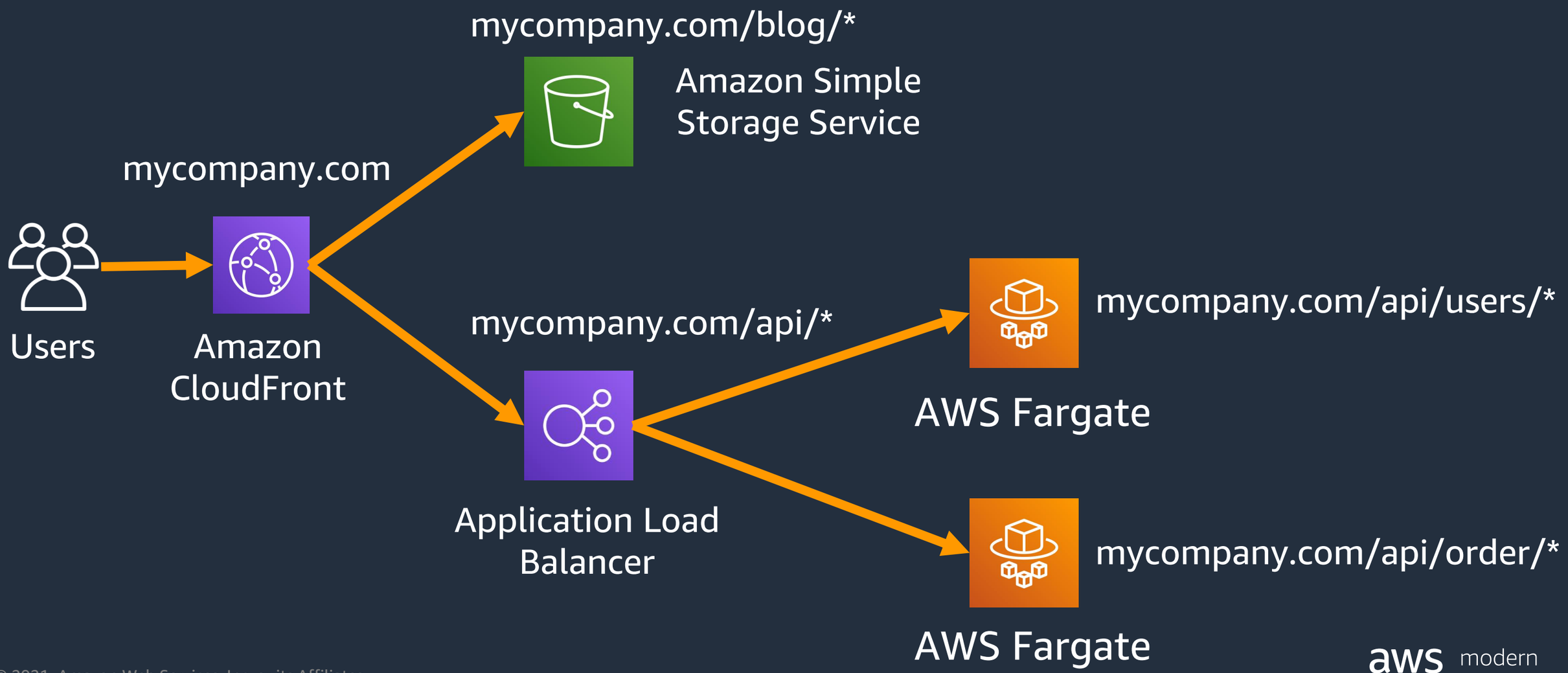
aws modern apps

Decouple traffic: One domain, multiple services



aws modern apps

Decouple API into microservices



aws modern apps

Application Load Balancer

The list of targets is managed by Elastic Container Service

Amazon Elastic Container Service

Listener

Listener Rule

Target Group

Listen for traffic on port 80 and 443

Check to see if traffic matches a rule

If it does, then send it to a target from this group

aws modern apps

Listener Rule

Up to 100 rules

Match on host

Hostname == mycompany.com
Hostname == api.mycompany.com

Match on path

Path == /api/users
Path == /api/orders

Match on query string

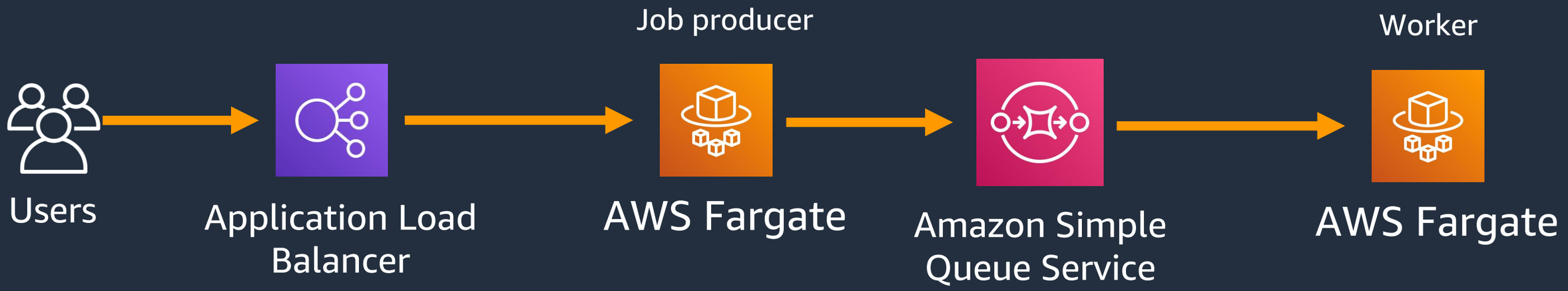
?utm_source==bot

Match on header

Version == 1.0.0
User-Agent == mobile

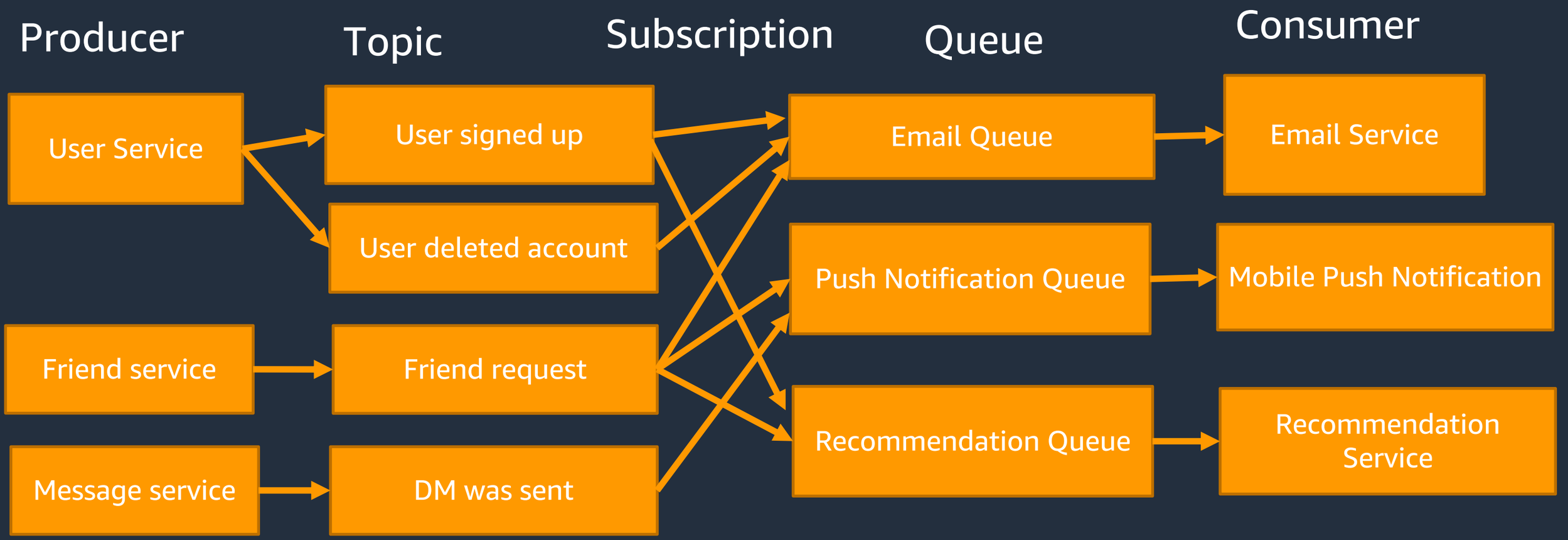
aws modern apps

Decouple background workers



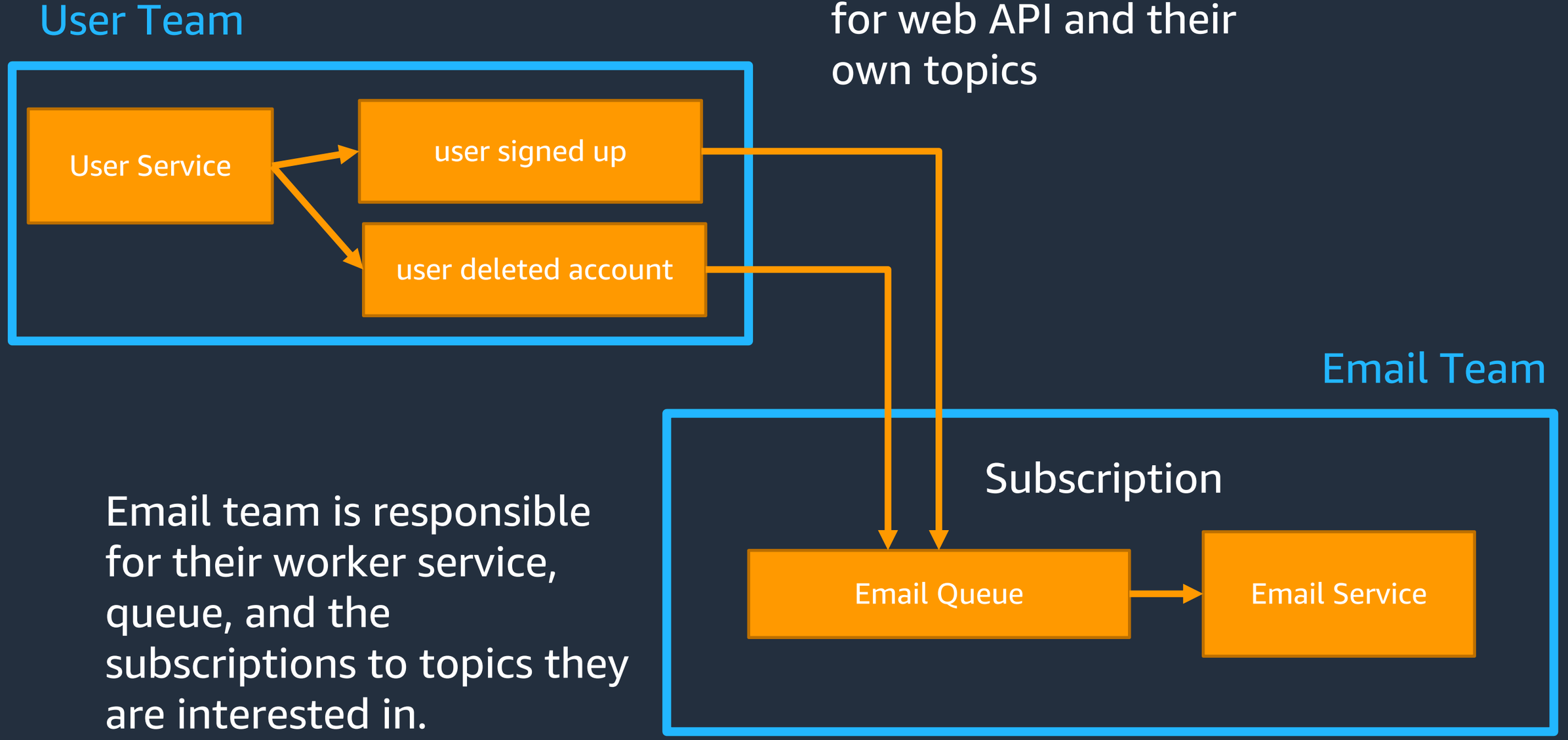
aws modern apps

Use Topics and Queues for more complicated business logic



aws modern apps

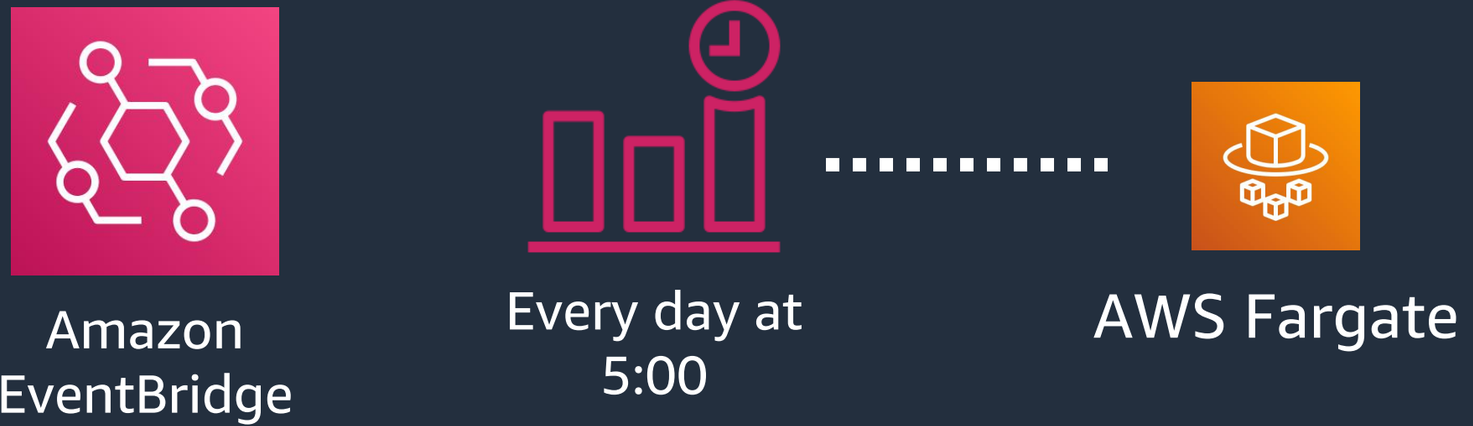
User team is responsible for web API and their own topics



Email team is responsible for their worker service, queue, and the subscriptions to topics they are interested in.

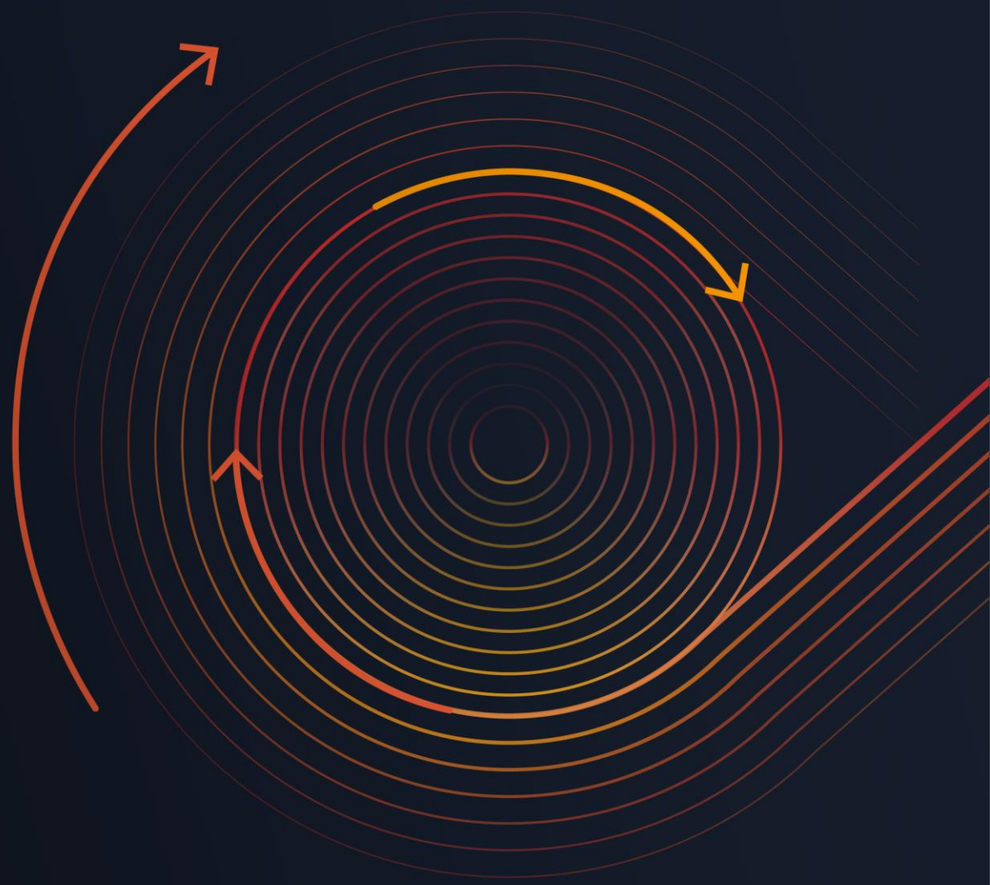
aws modern apps

Decouple scheduled tasks from the monolith



aws modern apps

Popular Decoupling Patterns with Serverless

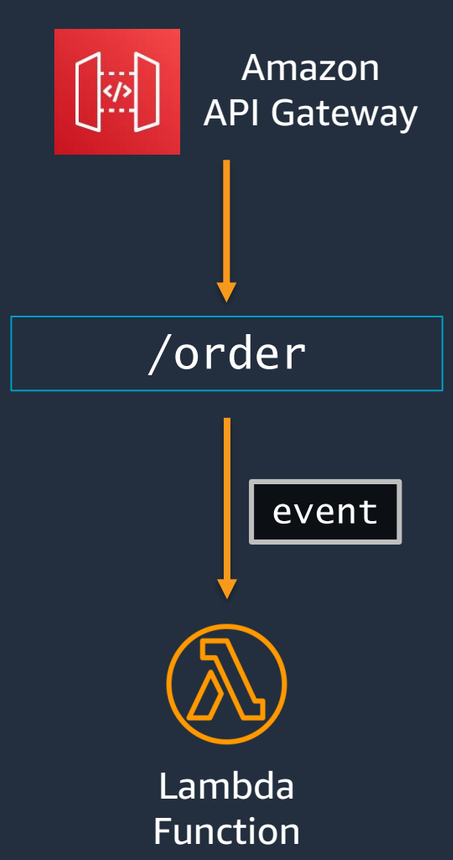


aws modern apps

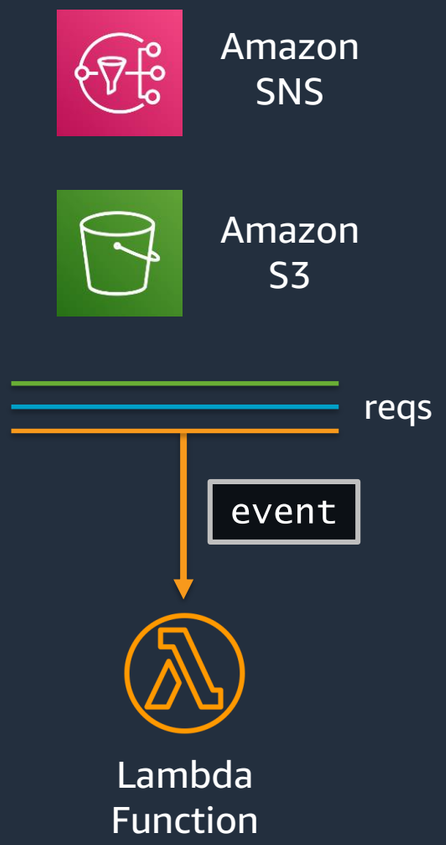
Lambda can be invoked via three different methods

All methods deliver an event payload

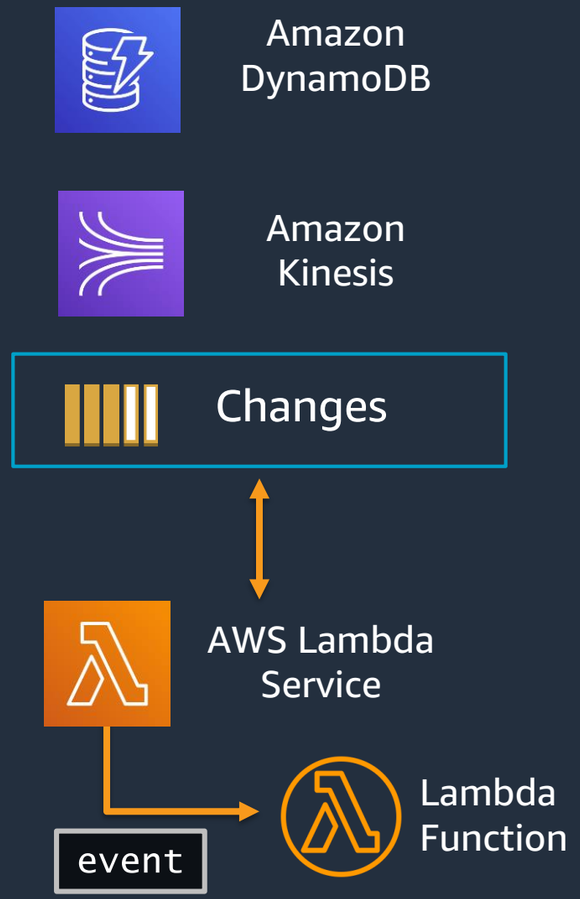
Synchronous



Asynchronous



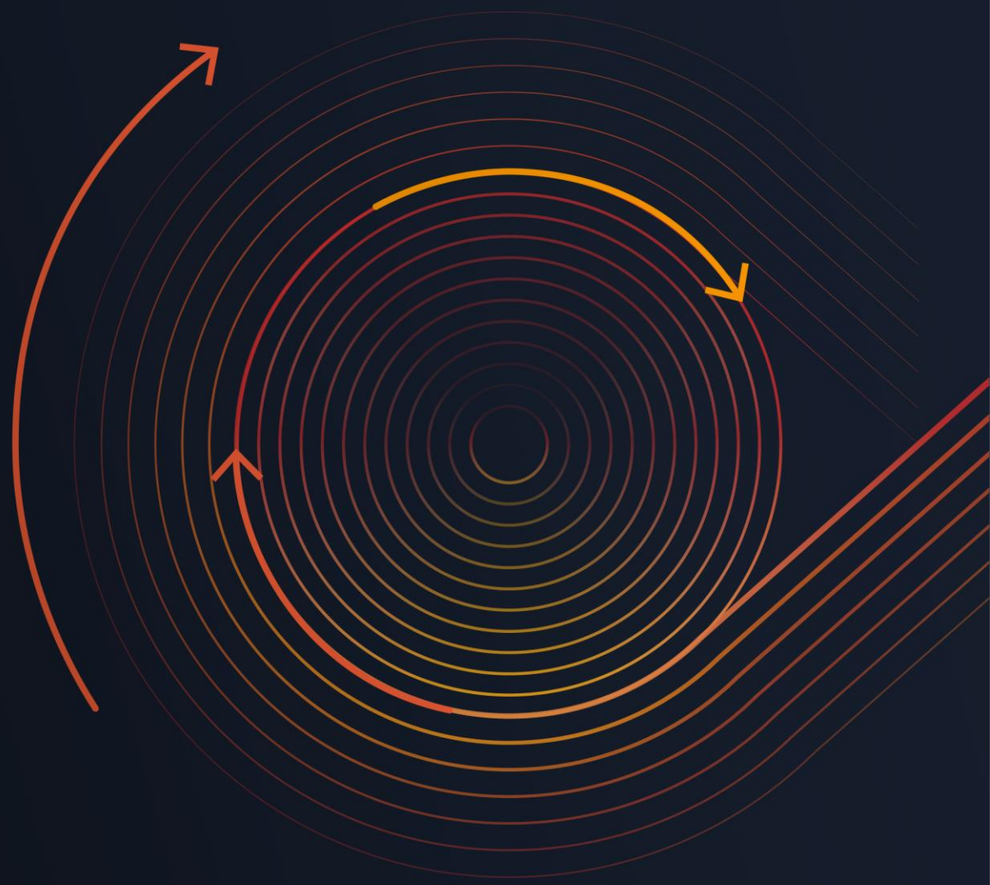
Poll-Based



aws modern apps

API-Driven Use Cases

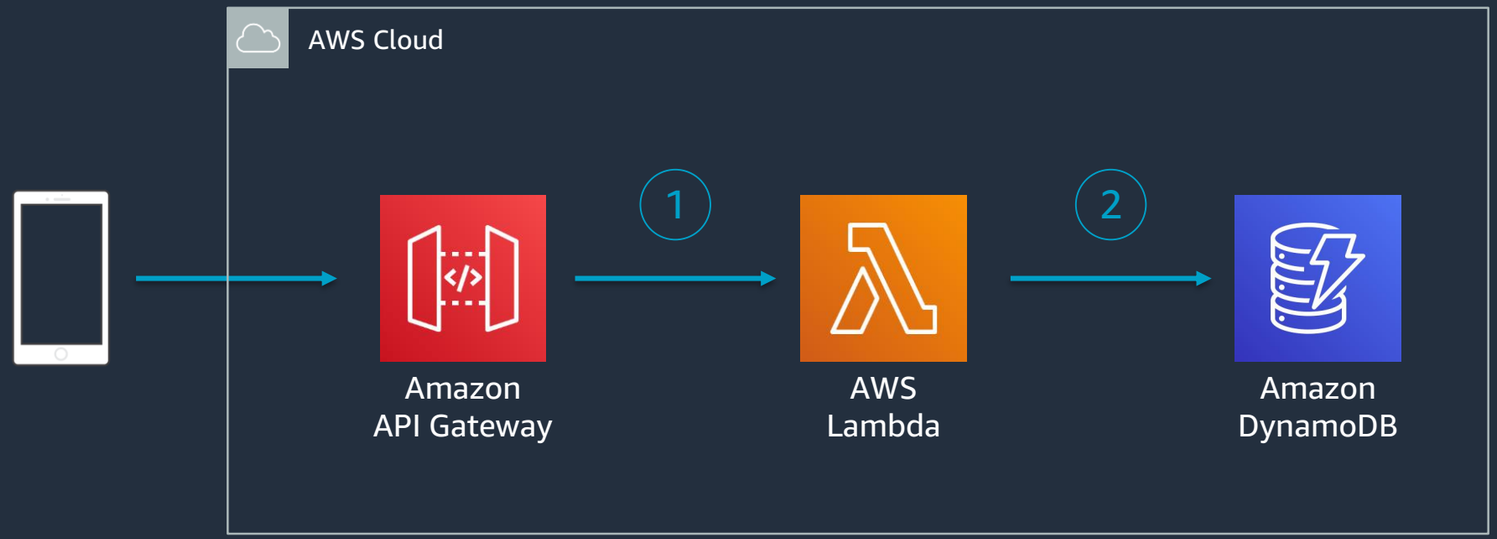
Also event driven, synchronously processed



aws modern apps

RESTful Microservices

Highly-scalable microservices

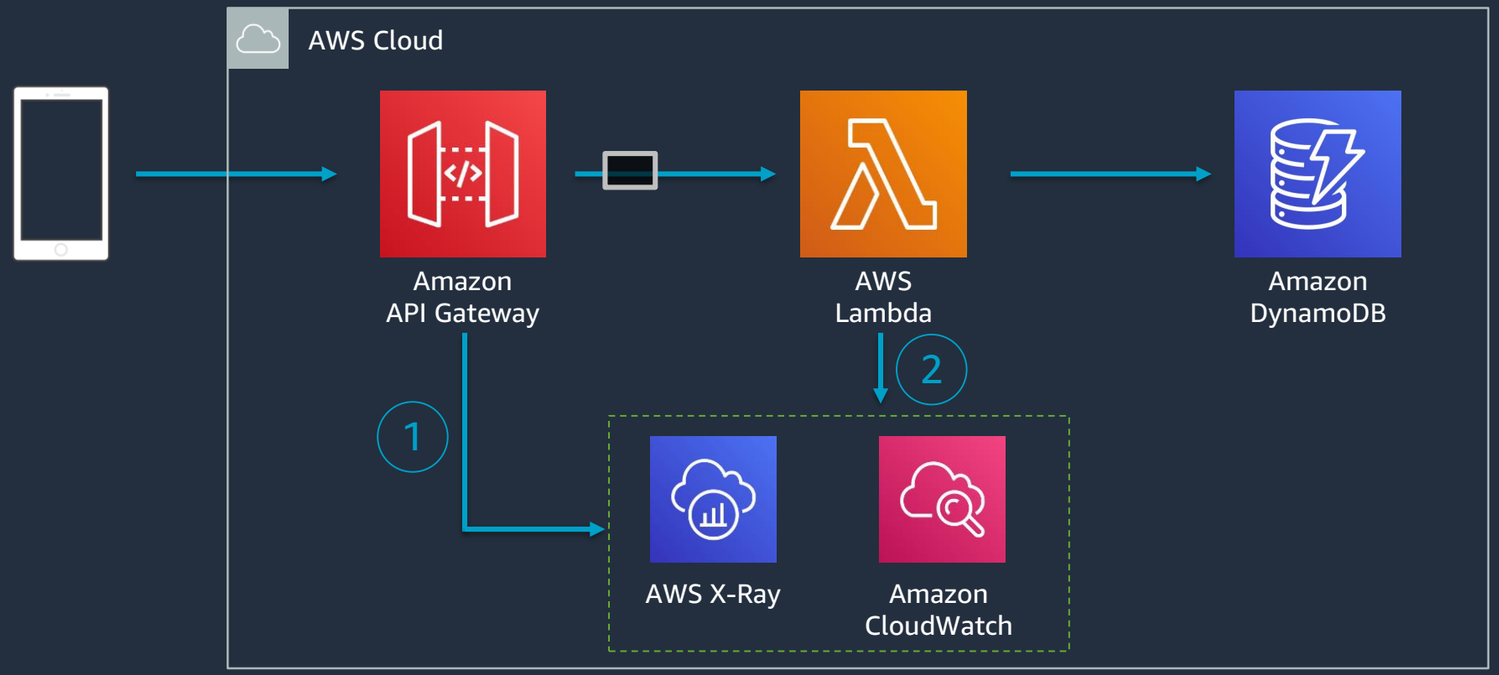


1. API Gateway “translates” incoming HTTP request to event payload
2. Lambda reads / writes data from data store

aws modern apps

RESTful Microservices with enhanced observability

Enable access logs, structured logging, and instrument code



1. Enable access logs and tracing
2. Instrument code and create metrics asynchronously with CloudWatch Embedded Metric Format

aws modern apps

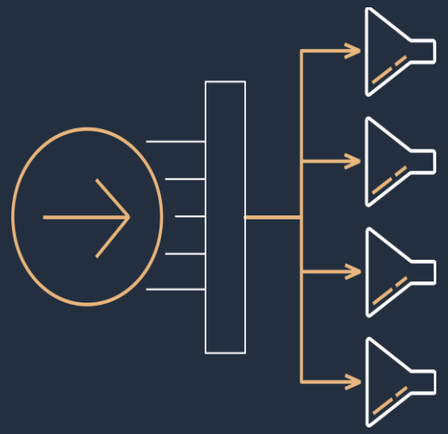
Event-Driven Use Cases

Streams, Topics, and Queues

aws modern apps

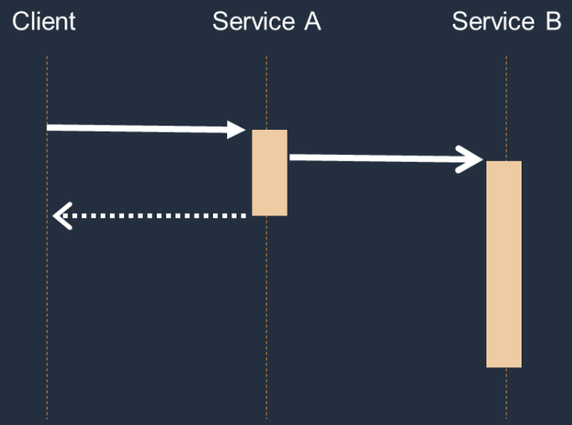
Event-driven architectures drive reliability and scalability

Event Routers



Abstract producers and consumers from each other

Asynchronous Events



Improve responsiveness and reduce dependencies

Event Stores

Buffer messages until services are available to process

aws modern apps

Processing file uploads

Resize photo, extract text, translate, etc.

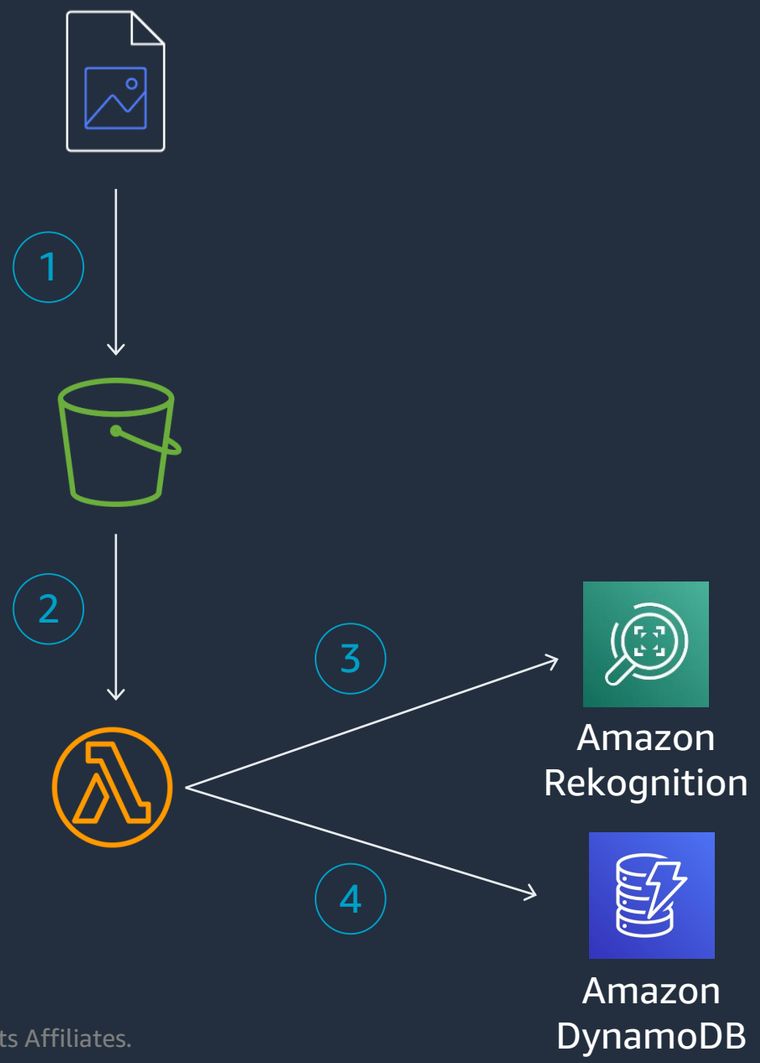
1. Object uploaded to Amazon S3 Bucket

2. Asynchronous invoke of Lambda function, event payload includes:
   - Bucket name
   - Object key

aws modern apps

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

aws modern apps

Streaming data ingestion and storage

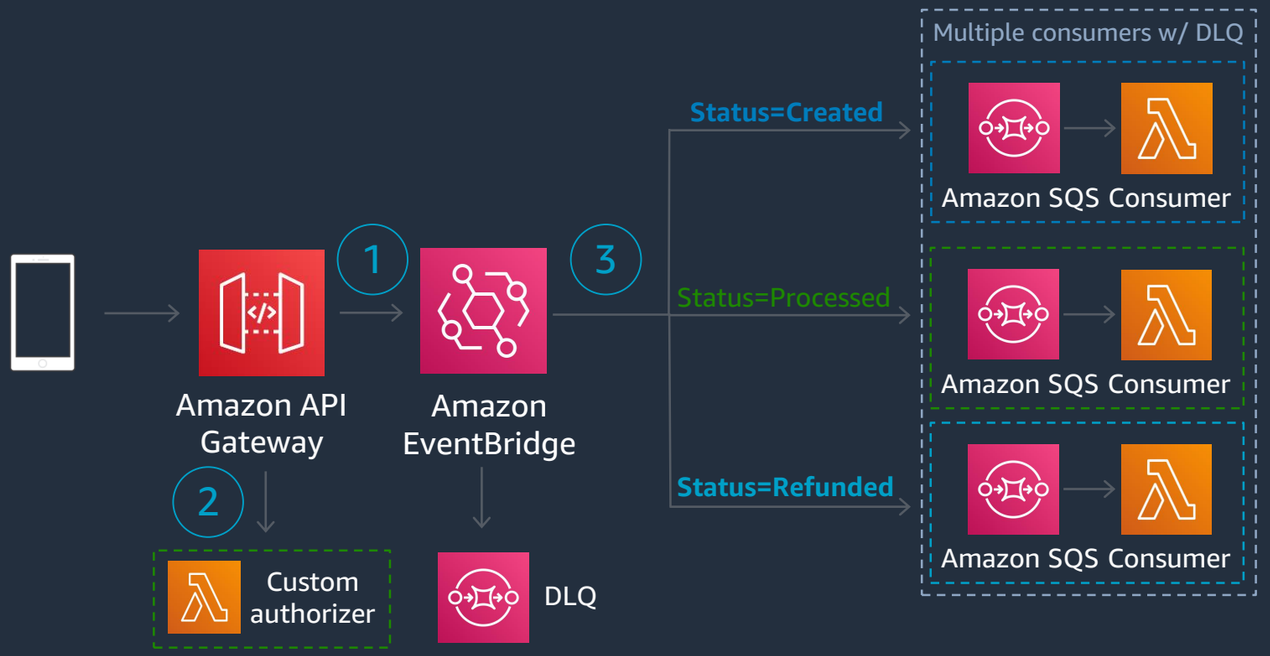
Consume, process, and store data

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

aws modern apps

Fan out

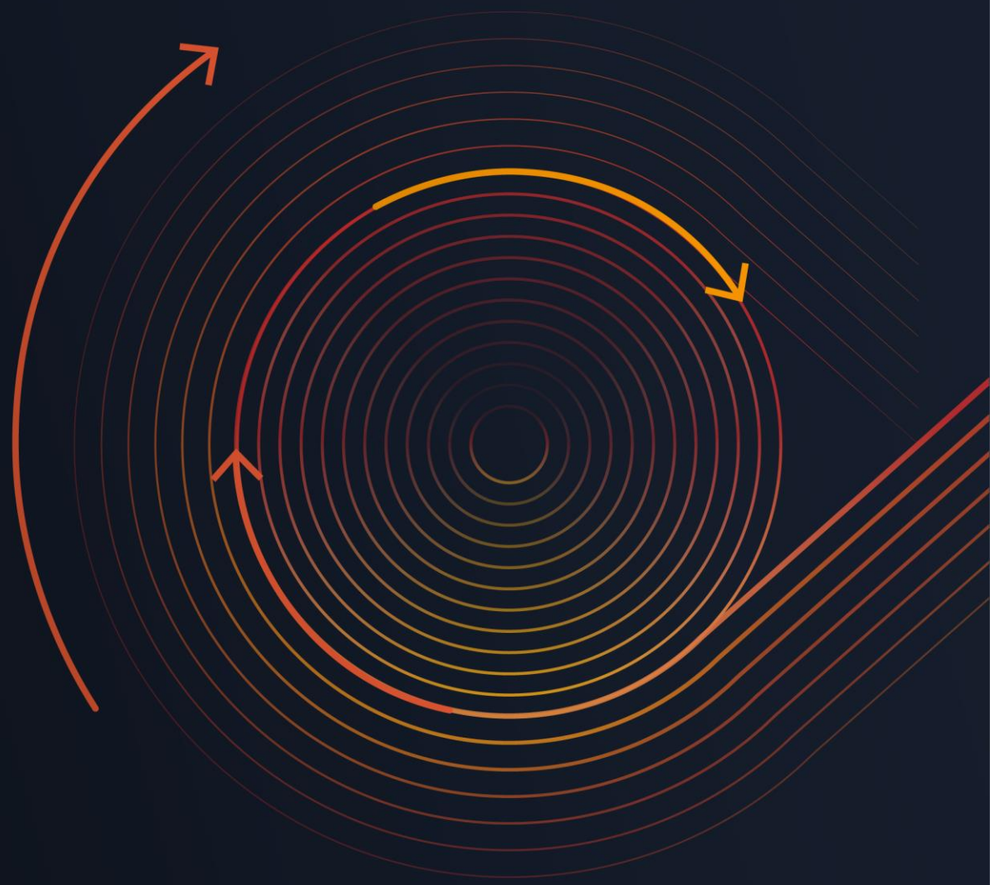
Push updates to multiple subscribers



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

aws modern apps

Practical Decoupling Tips



aws modern apps

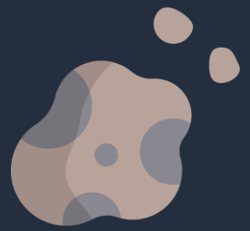
Practical decoupling: from monolith to micro

Functioning monolith

Trying to break things up too fast is a recipe for disaster

Kind of works?

Buggy service



Semi-functioning microservice

Boom!

Broken implementation

This part works great!

aws modern apps

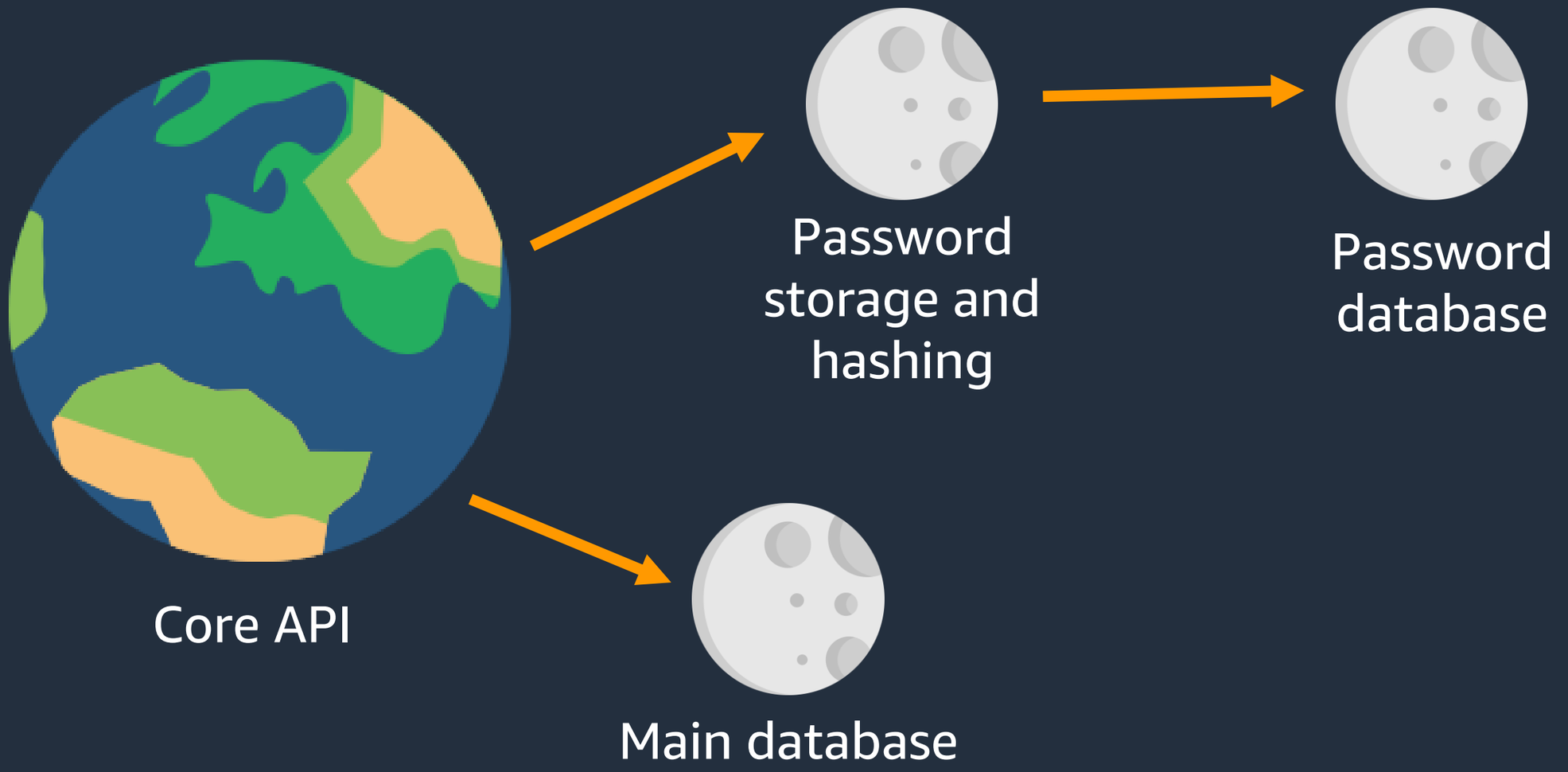
Decouple gradually, leave the central monolith for a while

Efficient microservice

Functioning monolith

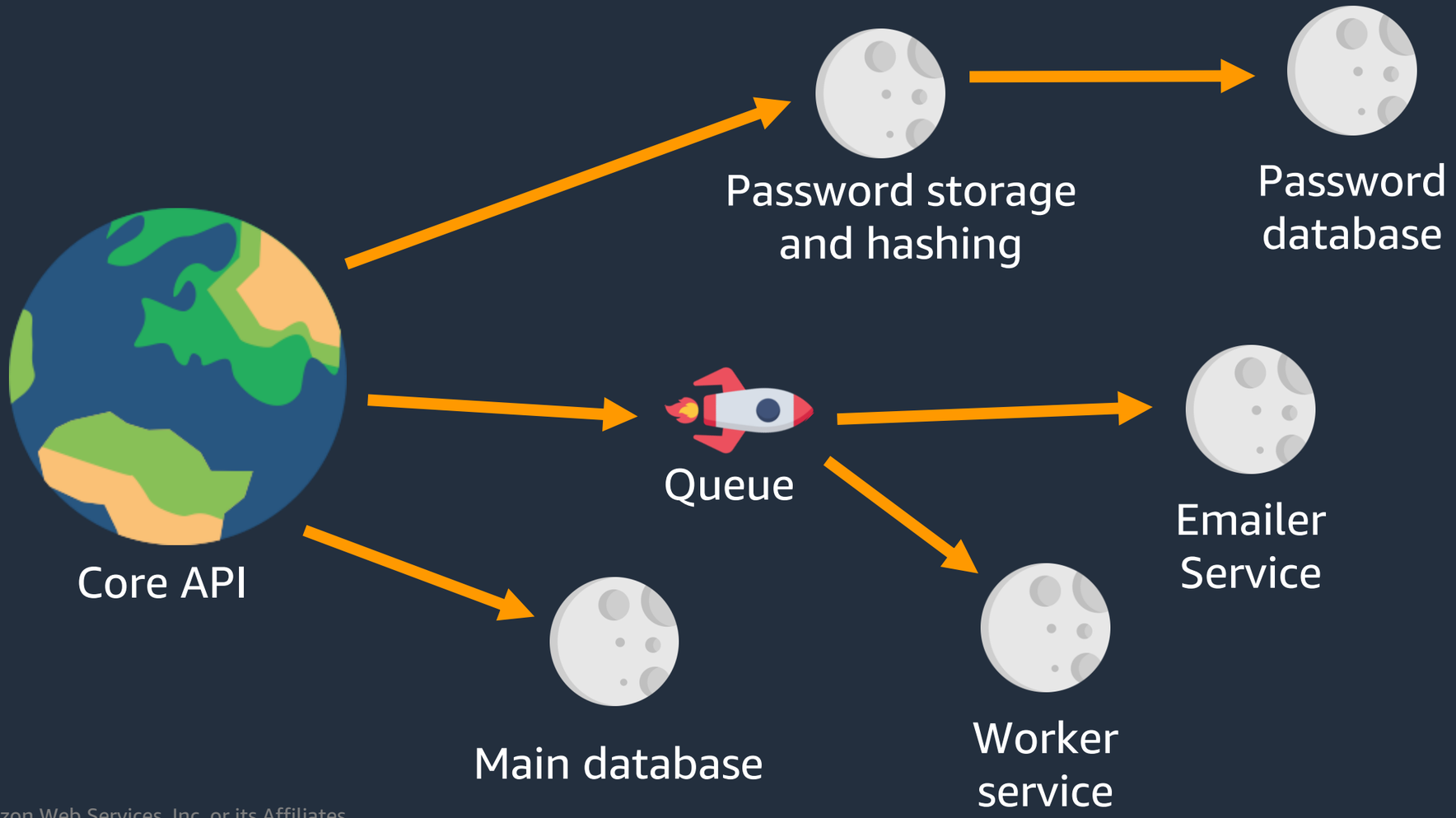
Well functioning service

Some practical places to start: User signup



aws modern apps

Some practical places to start: User signup



aws modern apps

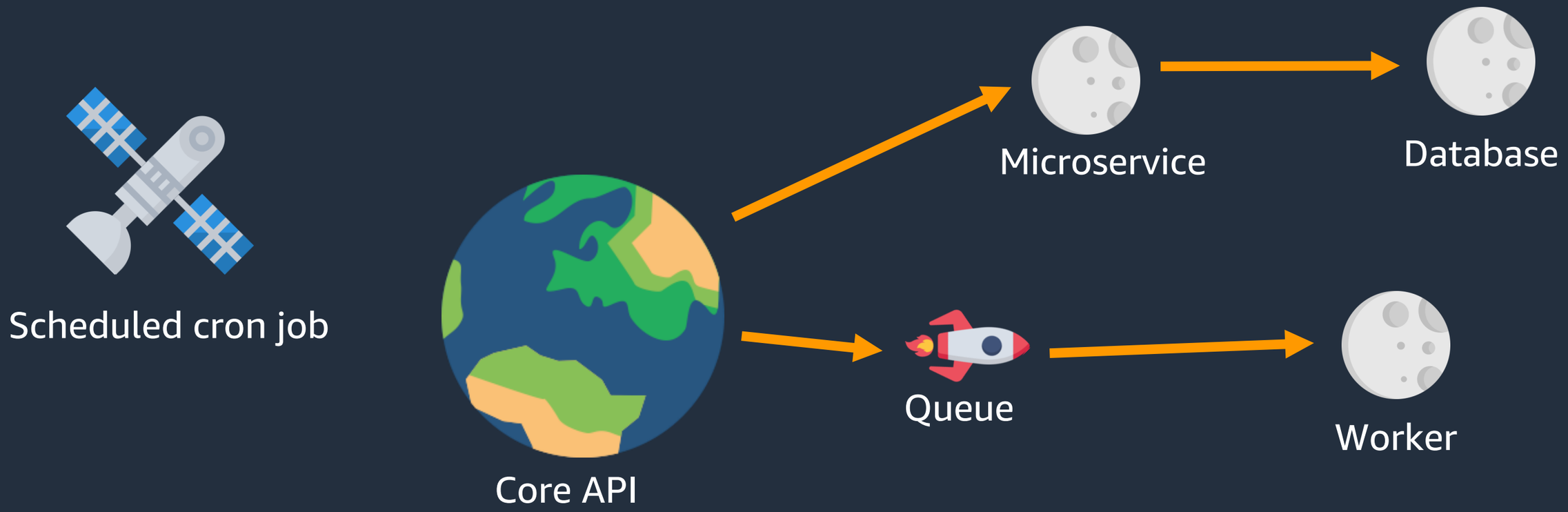
Look for transactions that have longer than average response time, or different resource needs:

- Social media application
  - Contact list upload, and server side contact matching (Big payloads)
  - Friend recommendations (Heavy queries, lots of data)

- Store
  - Product recommendation (Training models)
  - Payment processing (Need to keep payment details safe)

- Media
  - Upload processing, media transcoding (Heavy CPU, bandwidth)
  - Server side image resizing (Heavy CPU, bandwidth)

Decouple workloads responsibly - it is okay to have a central monolith



aws modern apps