AWS Fargate & ECS Masterclass

Kalyan Reddy Daida

AWS Fargate & ECS Masterclass

Course Contents

Course Outline

- Fargate & ECS - First Steps
- Docker Fundamentals
- Fargate & ECS Fundamentals
- ECR – Elastic Container Registry
- Load Balancing & Service Autoscaling
- Continuous Integration & Continuous Delivery
- Microservices Deployment without Service Discovery
- Microservices Deployment with Service Discovery
- Microservices Deployment with AWS App Mesh and X-Ray
- Microservices Canary Deployment with AWS App Mesh
- CloudFormation for Fargate Deployments

AWS Fargate & ECS

Introduction

ECS & Fargate - Introduction

- ECS – Elastic Container Service
- Fargate – Serverless Container Service
- ECS is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster.
- We can host our cluster on a serverless infrastructure that is managed by Amazon ECS by launching our services or tasks using the Fargate launch type.
- We can use Amazon ECS to schedule the placement of containers across our cluster based on our resource needs, isolation policies, and availability requirements.
- Amazon ECS eliminates the need for us to operate our own cluster management and configuration management systems or worry about scaling our management infrastructure.

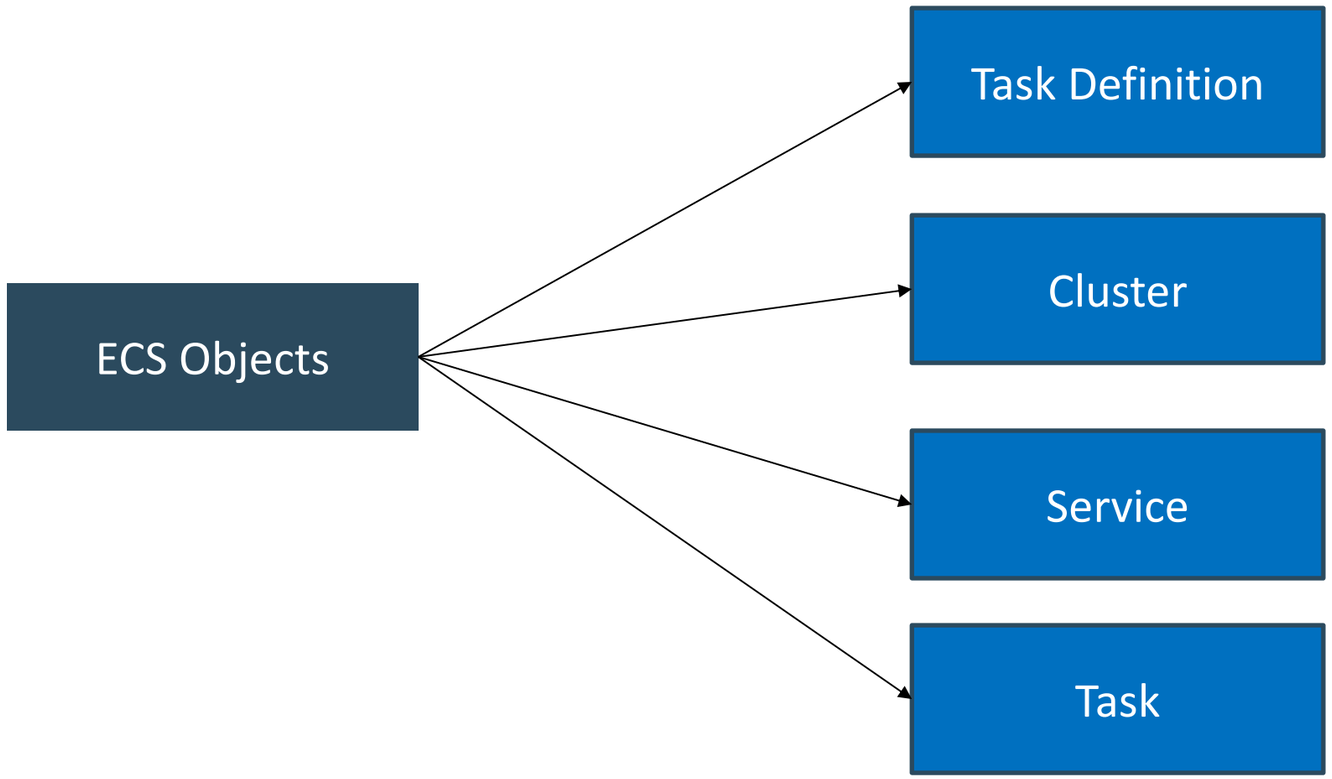
ECS & Fargate - Introduction

- Amazon ECS can be used to create a consistent deployment and build experience, manage, and scale batch and Extract-Transform-Load (ETL) workloads, and build sophisticated application architectures on a microservices model.

AWS Fargate & ECS

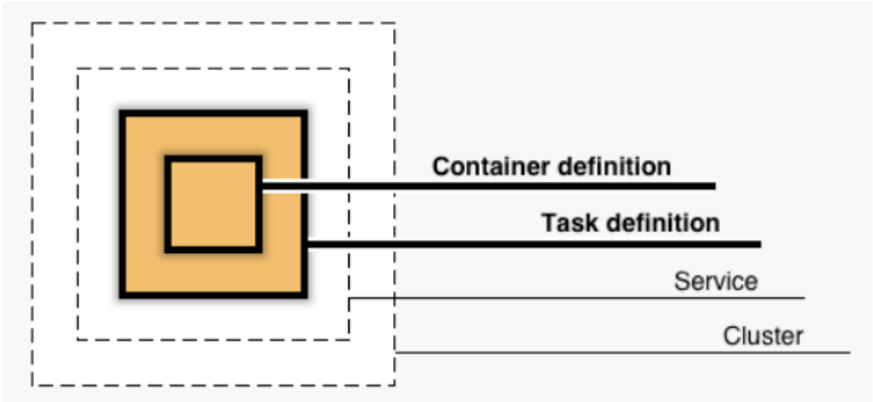
First Steps

Fargate or ECS Objects



Fargate & ECS – First Steps

- Container Definition
  - Nothing but container image and container level settings (Example: Container Image, Port, registry, Environment Variables to pass to container etc)
- Task Definition
  - A task definition is a blueprint for our application and describes one or more containers through attributes.
  - Very few attributes are configured at the task level, but majority of attributes are configured per container.
  - It is a combination of multiple container definitions if we are using more than one container image in a Task.
- Service
  - A service allows you to run and maintain a specified number (the "desired count") of simultaneous instances of a task definition in an ECS cluster.
- Fargate Cluster
  - The infrastructure in a Fargate cluster is fully managed by AWS. Our containers run without we managing and configuring individual Amazon EC2 instances.
- Task
  - A *task* is the instantiation of a task definition within a cluster.
  - After we have created a task definition for our application within Amazon ECS, we can specify the number of tasks that will run on our cluster (run task directly or configure to run from a service).
  - Each task that uses the Fargate launch type has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.

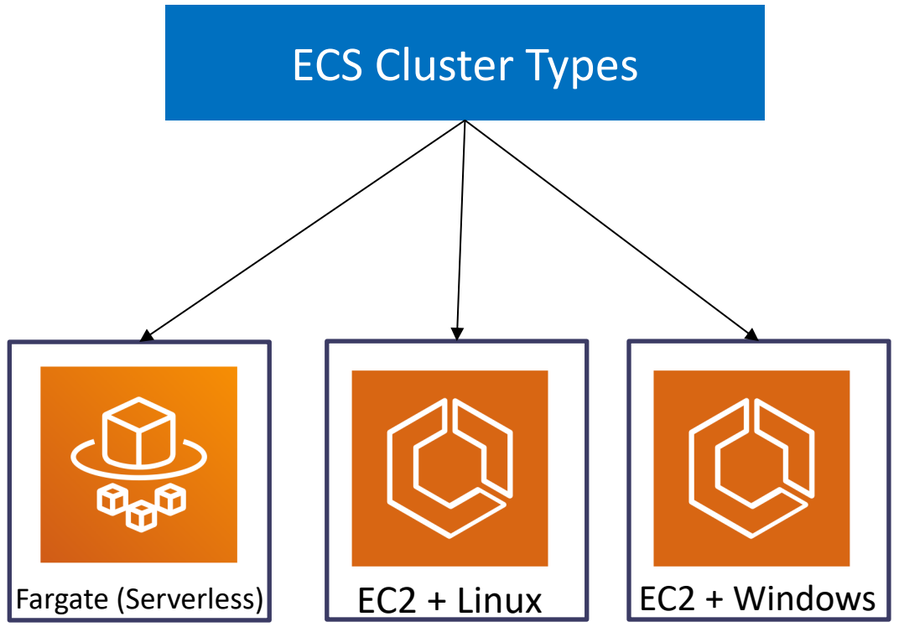


AWS Fargate & ECS

Clusters

Fargate & ECS Fundamentals – Clusters Introduction

- We have 3 types of cluster templates available in ECS.
  - Fargate - Serverless
  - EC2 – Linux
  - EC2 - Windows
- An ECS cluster is a logical grouping of tasks or services.
- Clusters are Region-specific.
- Clusters can contain tasks using both the Fargate and EC2 launch types.



AWS Fargate & ECS

Cluster Features

Fargate & ECS Fundamentals – Cluster Features

Services	A service allows you to run and maintain a specified number of simultaneous instances of a task definition in an ECS cluster.
Tasks	A *task* is the instantiation of a task definition within a cluster
ECS Instances	Will be created when we create the cluster launch type of EC2+Linux or EC2+Windows
Metrics	CloudWatch Container Insights collects, aggregates, and summarizes metrics and logs from your containerized applications and microservices.
Scheduled Tasks	Used primarily for long running stateless services and applications.
Tags	ECS resources can be tagged with values that we define, to help us organize and identify them.
Capacity Providers	A capacity provider is used in association with a cluster to determine the infrastructure that a task runs on.
Update Cluster	Update the cluster settings primarily leading to Cluster Capacity Providers
Delete Cluster	Used to delete ECS EC2 or Fargate Clusters

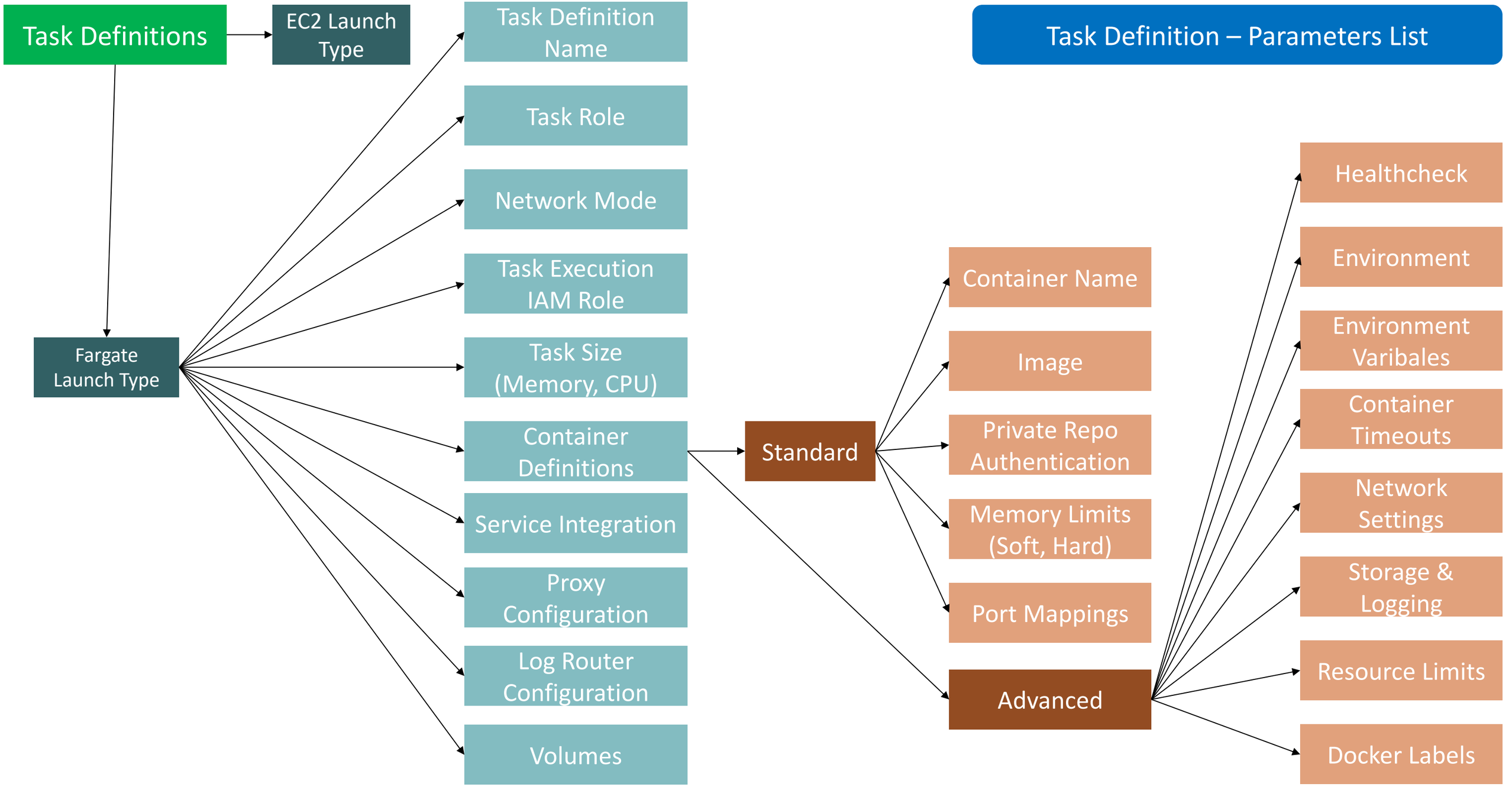
AWS Fargate & ECS

Task Definition

Fargate & ECS Fundamentals – Task Definition

- Task Definition
  - A task definition is required to run Docker containers in Amazon ECS
  - A task definition is a blueprint for our application and describes one or more containers through attributes.
  - Some attributes are configured at the task level, but majority of attributes are configured per container.
- Task Definition Parameters - Core
  - The Docker image to use with each container in your task
  - How much CPU and memory to use with each task
  - The launch type to use, which determines the infrastructure on which our tasks are hosted (EC2 or Fargate)
  - The Docker networking mode to use for the containers in our task (Fargate defaults to awsvpc, where as EC2 supports docker networking models like Birdged, Host, None and awsvpc too).
  - The logging configuration to use for our tasks
  - Whether the task should continue to run if the container finishes or fails
  - Any data volumes that should be used with the containers in the task
  - And many more……..

Task Definition – Parameters List

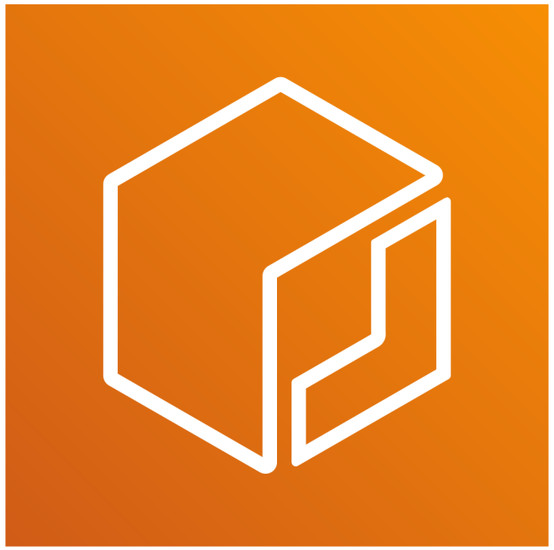


Fargate & ECS Fundamentals – Task Definition

- Step-1: Create Task Definition
  - Task Role
    - IAM role that tasks can use to make API requests to authorized AWS services
  - Network Mode
    - For Fargate we have only option available is awsvpc in addition we will have Docker Bridge, Docker Host Only and None network modes. We will see them during ECS EC2 Cluster.
  - Task Execution Role
    - This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on our behalf.

Fargate & ECS Fundamentals – Task Definition

- Create Task Definition
  - Task Size
    - The task size allows us to specify a fixed size for our task.
    - Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type.
    - Container level memory settings are optional when task size is set.
    - Task size is not supported for Windows containers.
  - Container Definition
    - Standard Settings
      - Container Name
      - Image: stacksimplify/dockerintro-springboot-helloworld-rest-api:1.0.0-RELEASE
      - Private Repo
      - Memory Limits
      - Port Mappings
    - Advanced Container Configurations
      - Storage & Logging: Log Configuration


AWS Fargate & ECS

Elastic Container Registry - ECR

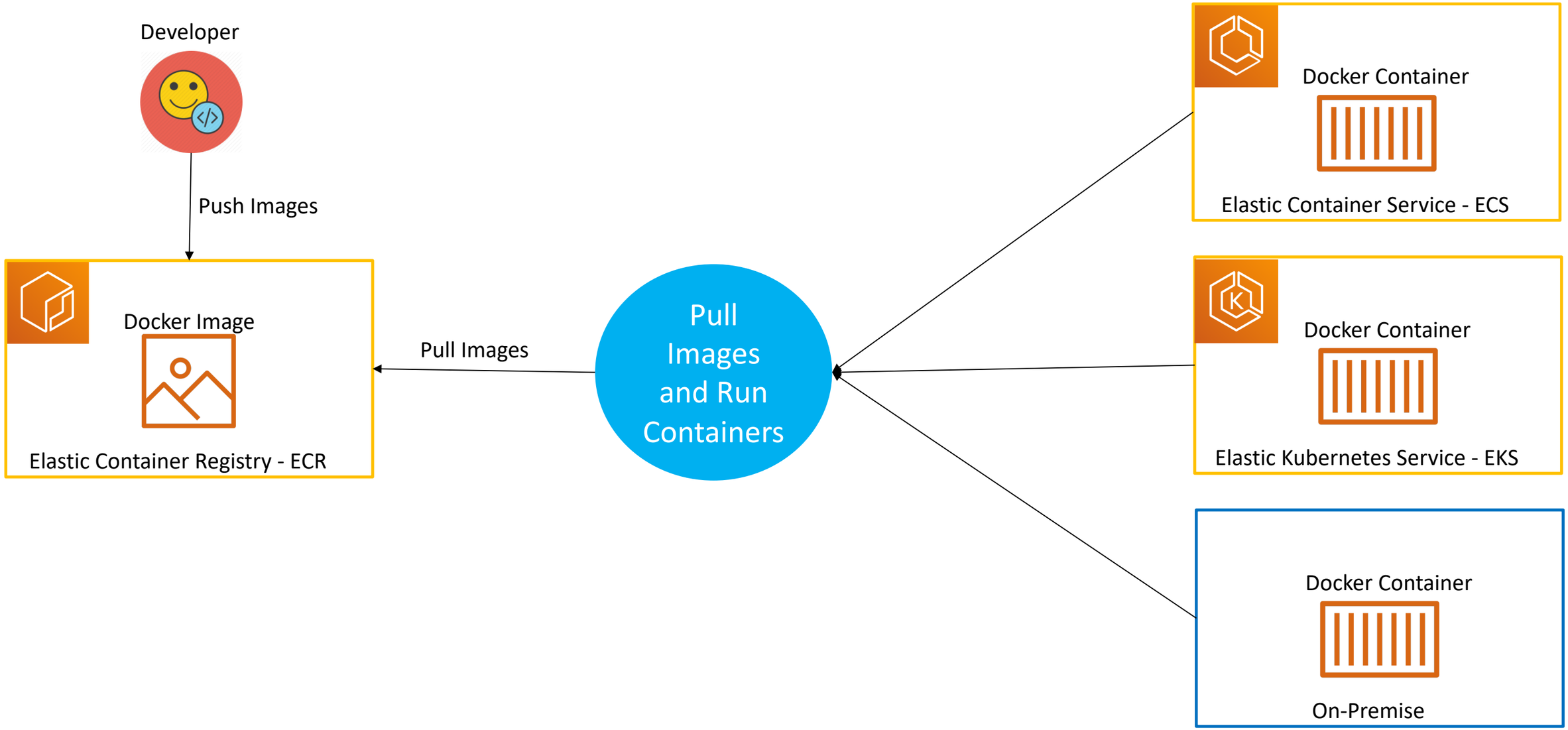
Elastic Container Registry - ECR

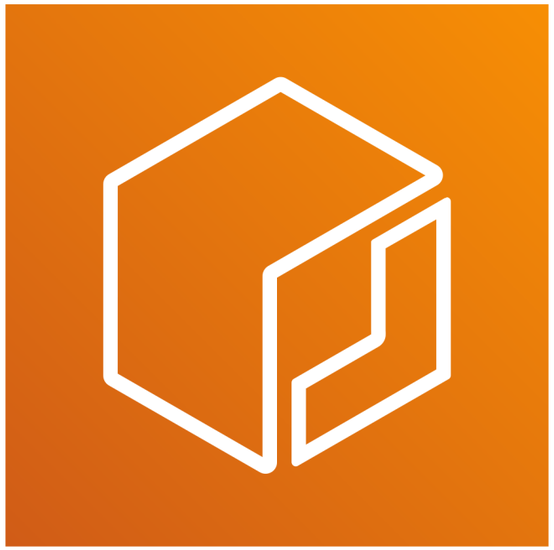
- Elastic Container Registry (ECR) is a fully-managed Docker container registry that makes it easy for developers to store, manage, and deploy Docker container images.
- ECR is integrated with Elastic Container Service (ECS), simplifying our development to production workflow.
- ECR eliminates the need to operate our own container repositories or worry about scaling the underlying infrastructure.
- ECR hosts our images in a highly available and scalable architecture, allowing us to reliably deploy containers for our applications.
- Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.
- With Amazon ECR, there are no upfront fees or commitments. We pay only for the amount of data you store in your repositories and data transferred to the Internet.

Elastic Container Registry - ECR

- Benefits
  - Full managed
  - Secure
  - Highly Available
  - Simplified Workflow

How ECR Works?




AWS Fargate & ECS

Continuous Integration & Continuous Delivery


CodeCommit


CodeBuild


CodeDeploy


CodePipeline


CloudWatch


Simple Notification Service

Stages in Release Process

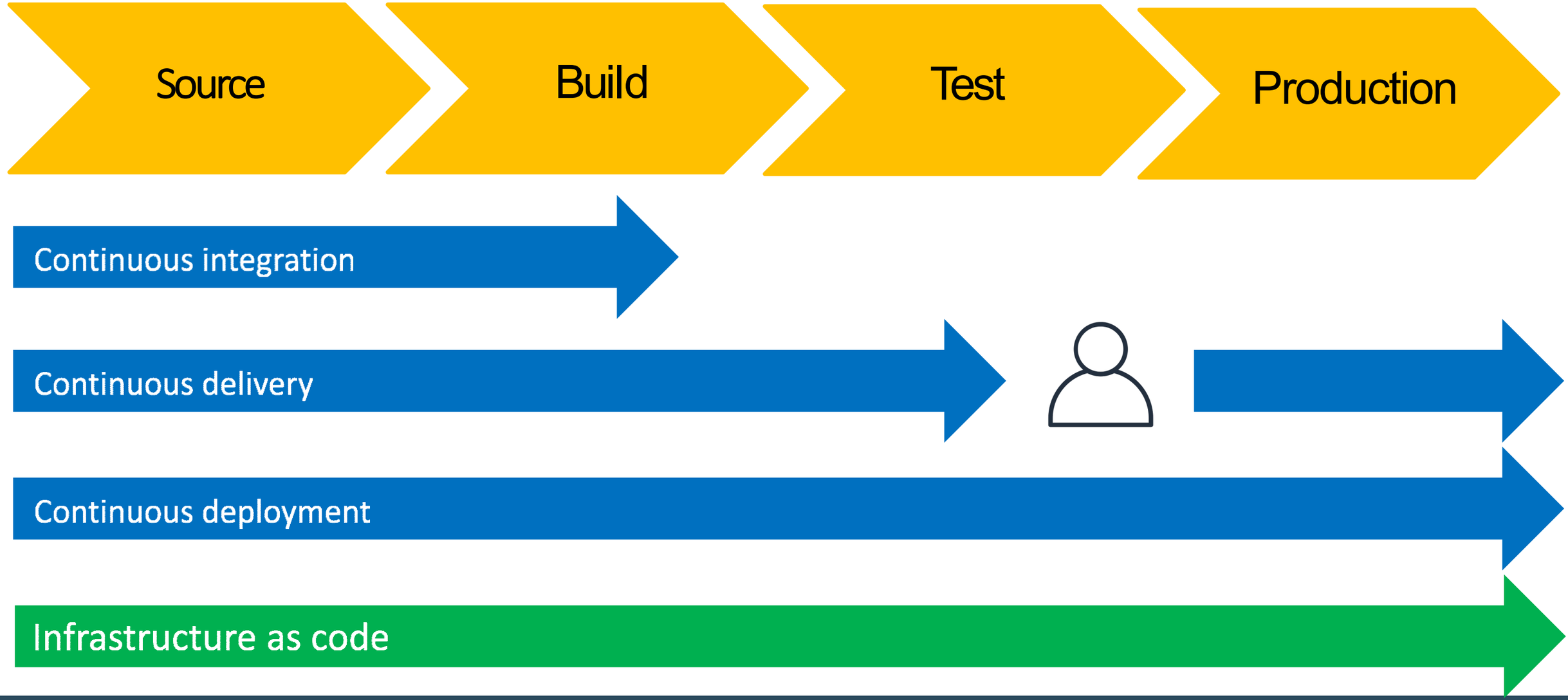
- Check-in source code
- Peer review new code
- Pull Request process

- Compile Code & build artifacts (war ,jar, container images, Kubernetes manifest files)
- Unit Tests

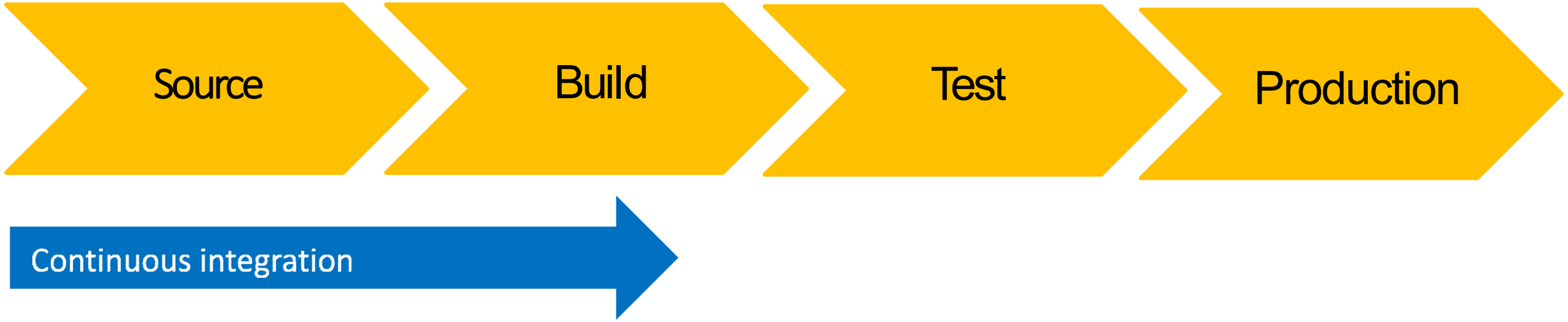
- Integration tests with other systems.
- Load Testing
- UI Tests
- Security Tests
- Test Environments (Dev, QA and Staging)

- Deployment to production environments
- Monitor code in production to quickly detect errors

Stages in Release Process



Continuous Integration

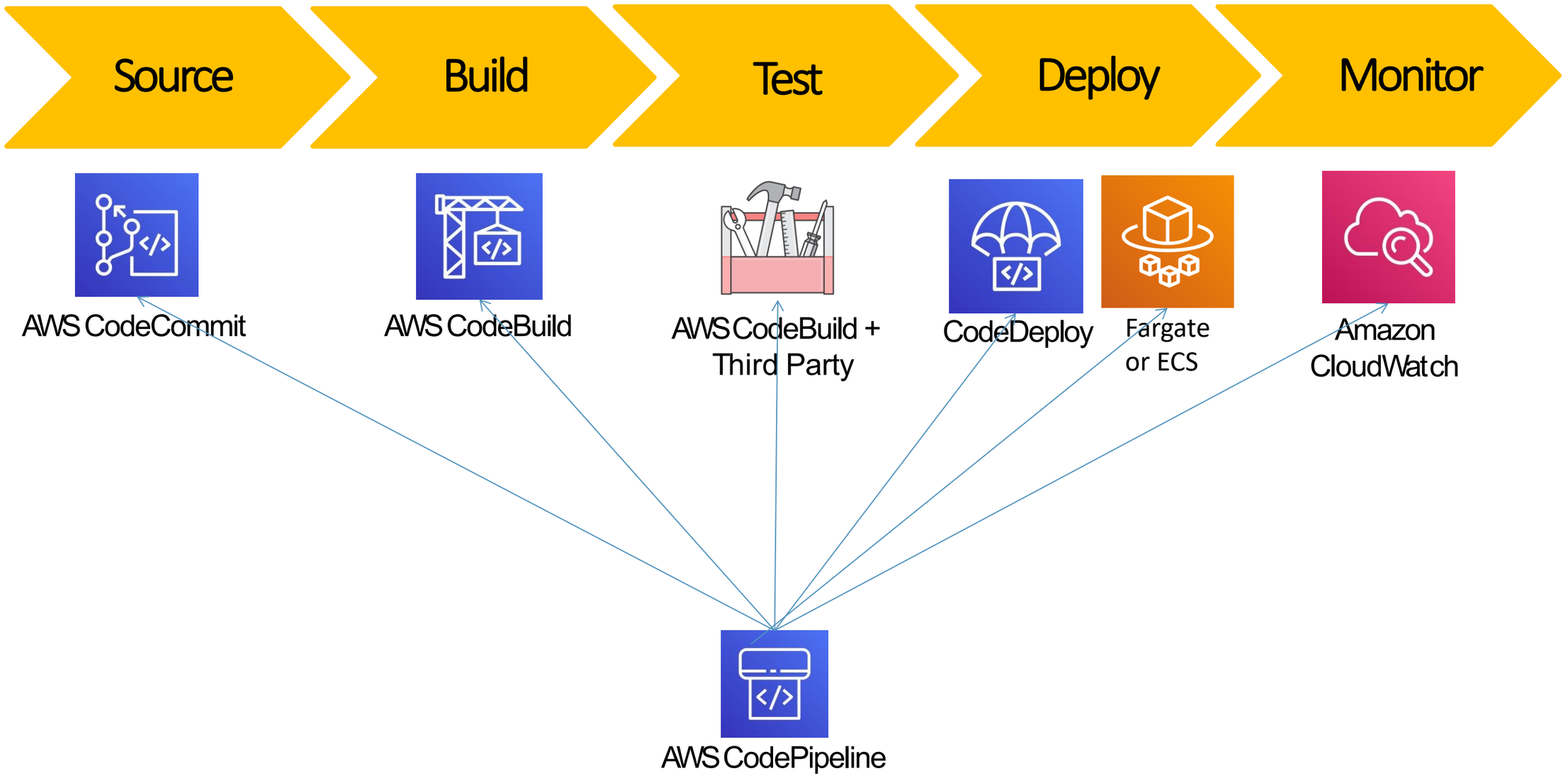


- Automatically kick off a new release when new code is checked-in
- Build and test code in a consistent, repeatable environment
- Continually have an artifact ready for deployment

Continuous Delivery

- Automatically deploy new changes to staging environments for testing
- Deploy to production safely without affecting customers
- Deliver to customers faster
- Increase deployment frequency, and reduce change lead time and change failure rate

AWS Developer Tools or Code Services



AWS Developer Tools or Code Services

AWS CodeCommit

AWS CodeBuild

AWS CodeBuild + Third Party

Fargate or ECS

Amazon CloudWatch

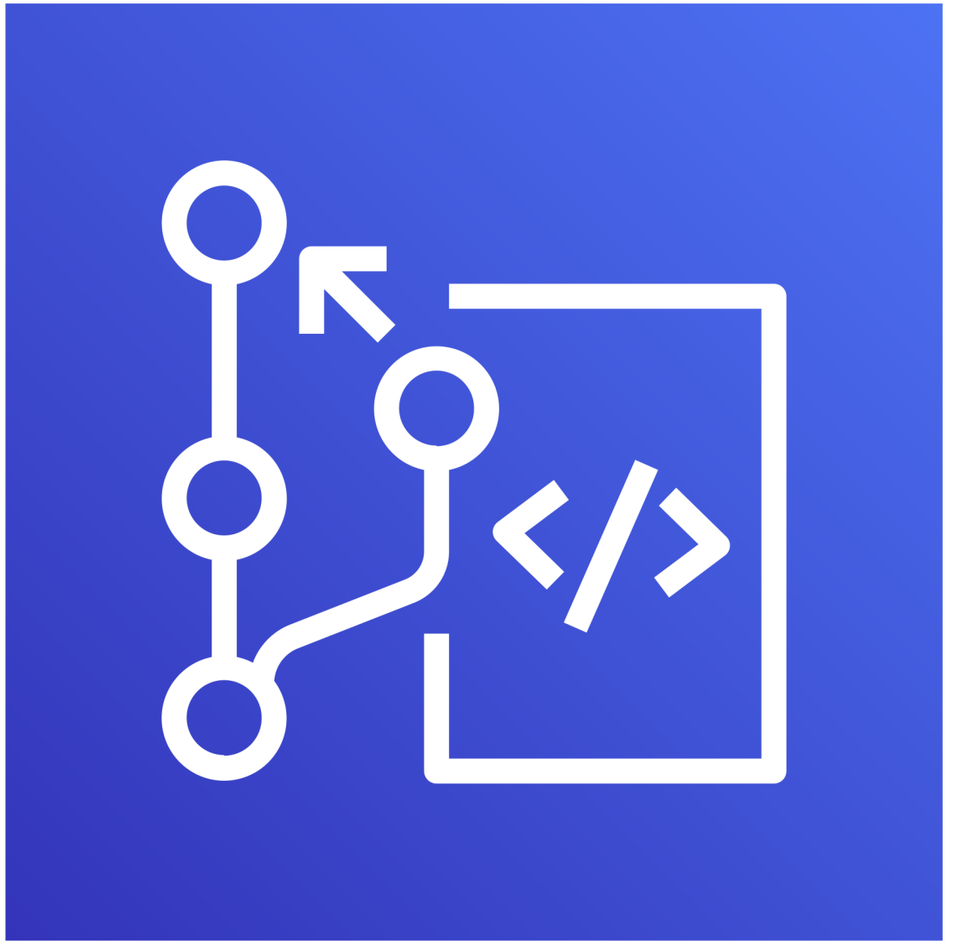
Continuous integration

Continuous delivery

Continuous deployment

AWS CodePipeline

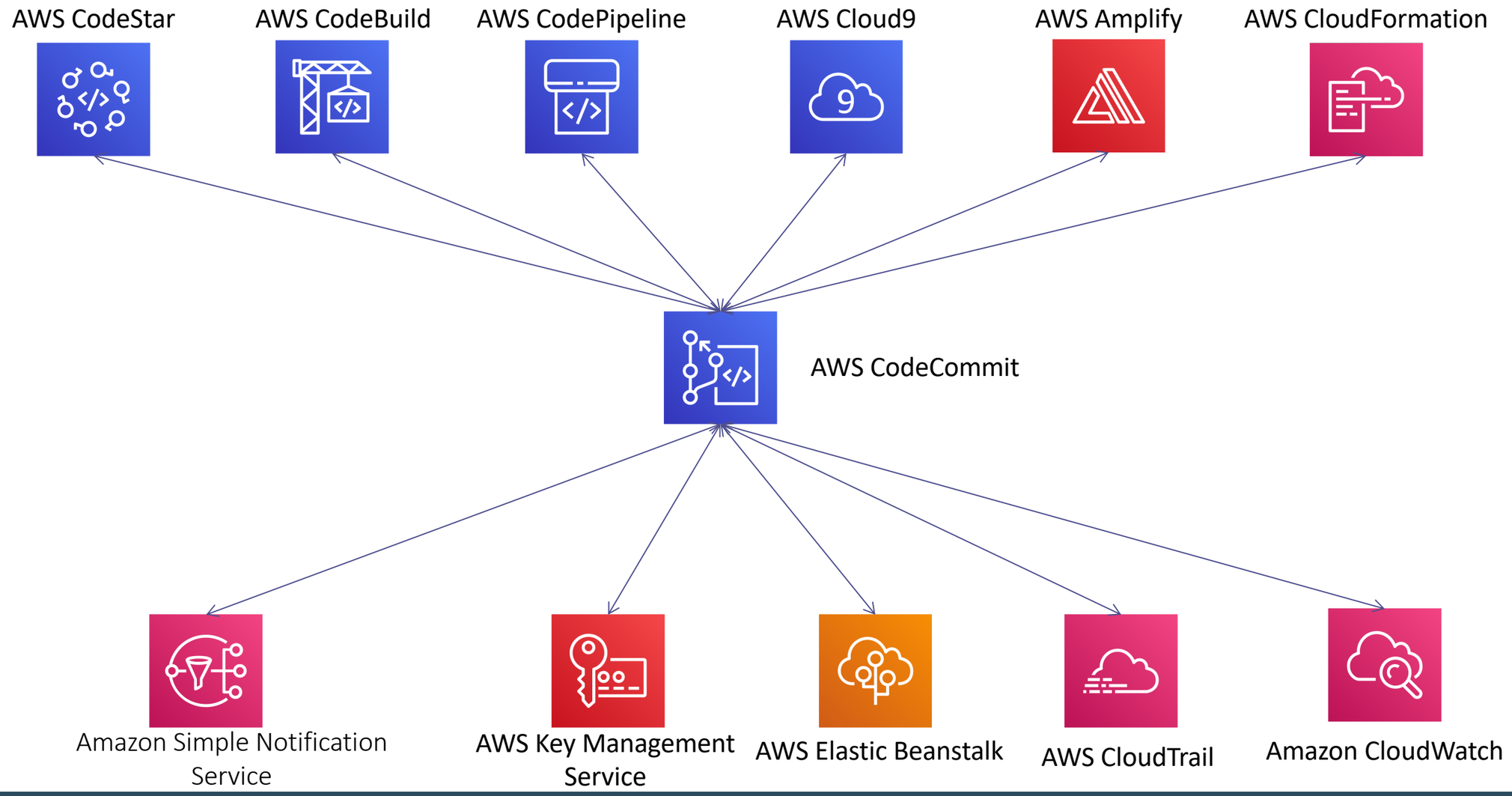
AWS CodeCommit



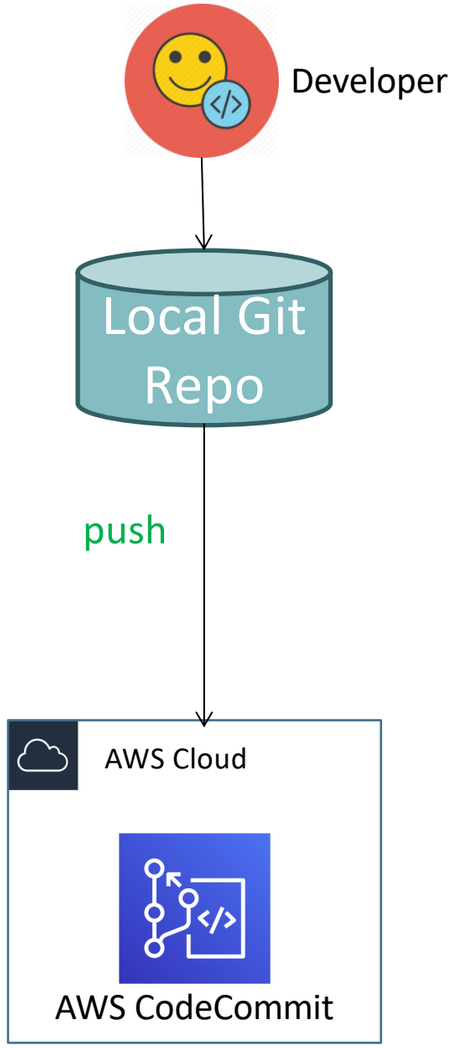
AWS CodeCommit - Introduction

- Version Control Service hosted by AWS
- We can privately store and manage documents, source code, and binary files
- Secure & highly scalable
- Supports standard functionality of Git (CodeCommit supports Git versions 1.7.9 and later.)
- Uses a static user name and password in addition to standard SSH..

CodeCommit – Integration with AWS Services



CodeCommit - Steps

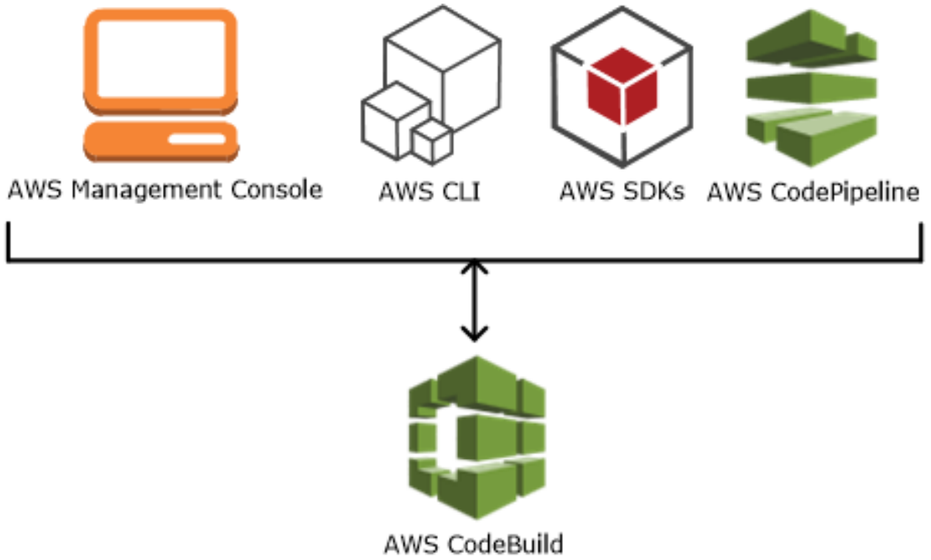


AWS CodeBuild

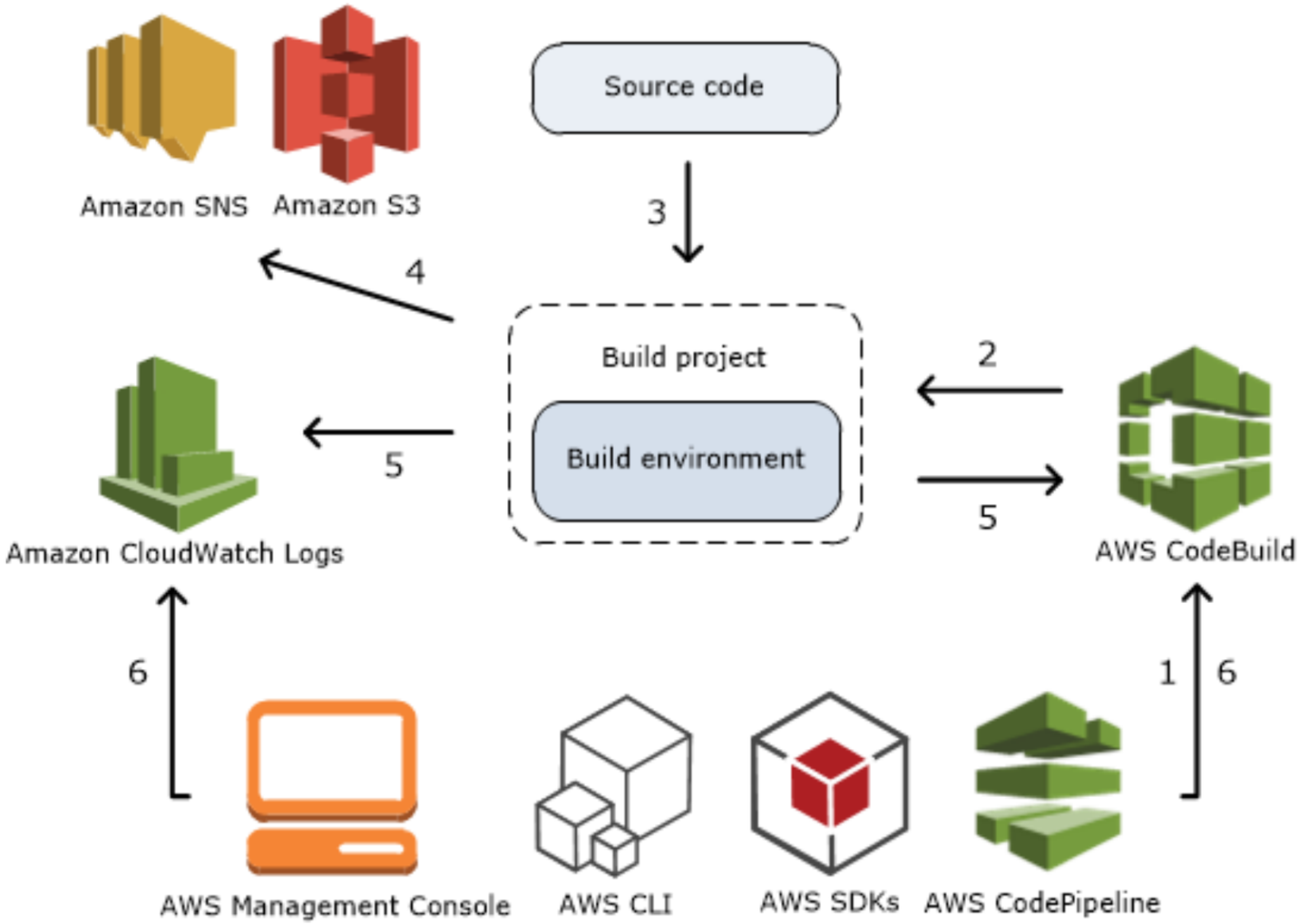
CodeBuild - Introduction

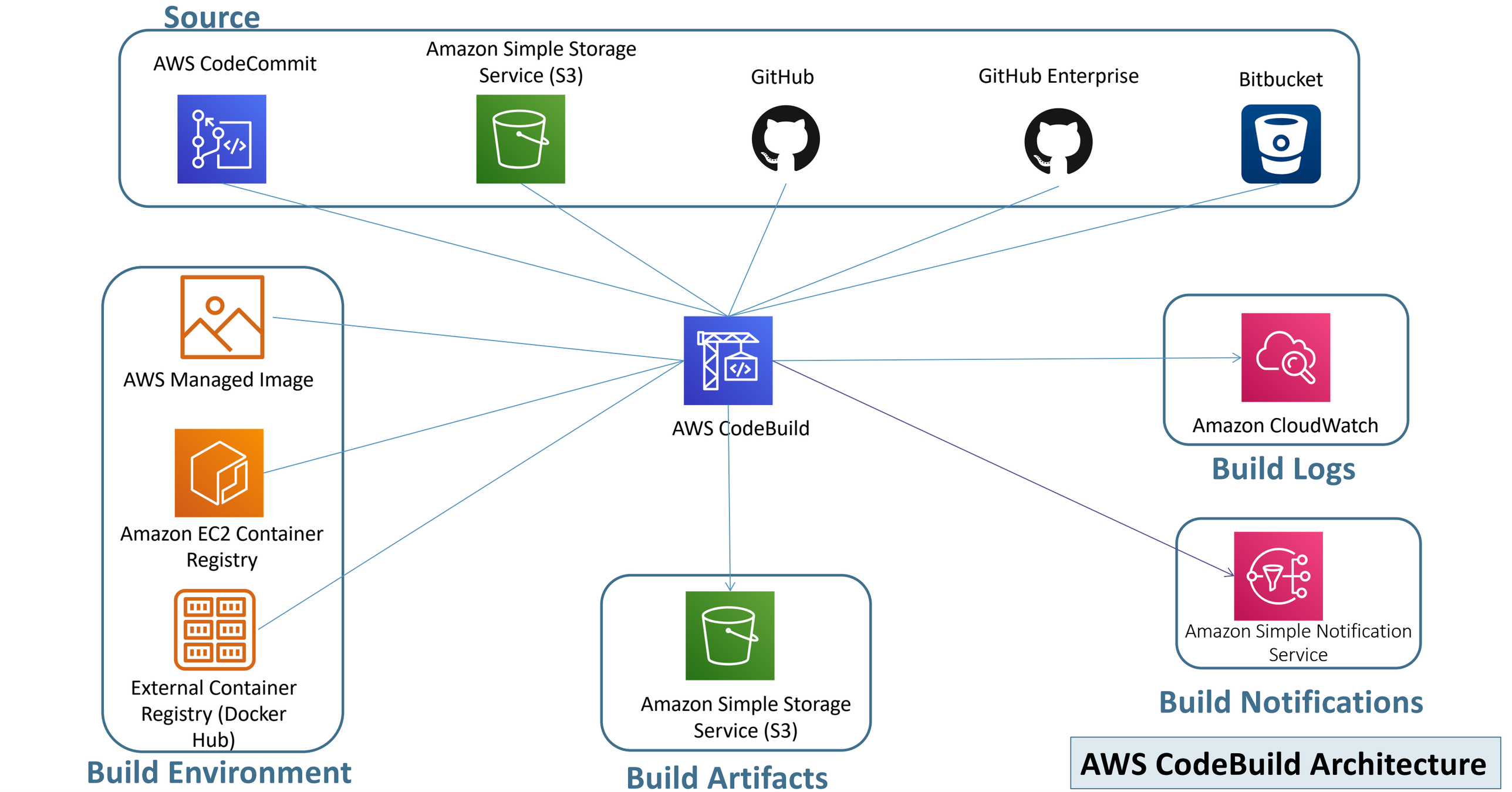
- CodeBuild is a fully managed build service in the cloud.
- Compiles our source code, runs unit tests, and produces artifacts that are ready to deploy.
- Eliminates the need to provision, manage, and scale our own build servers.
- It provides prepackaged build environments for the most popular programming languages and build tools such as Apache Maven, Gradle, and many more.
- We can also customize build environments in CodeBuild to use our own build tools.
- Scales automatically to meet peak build requests.

How to run CodeBuild?

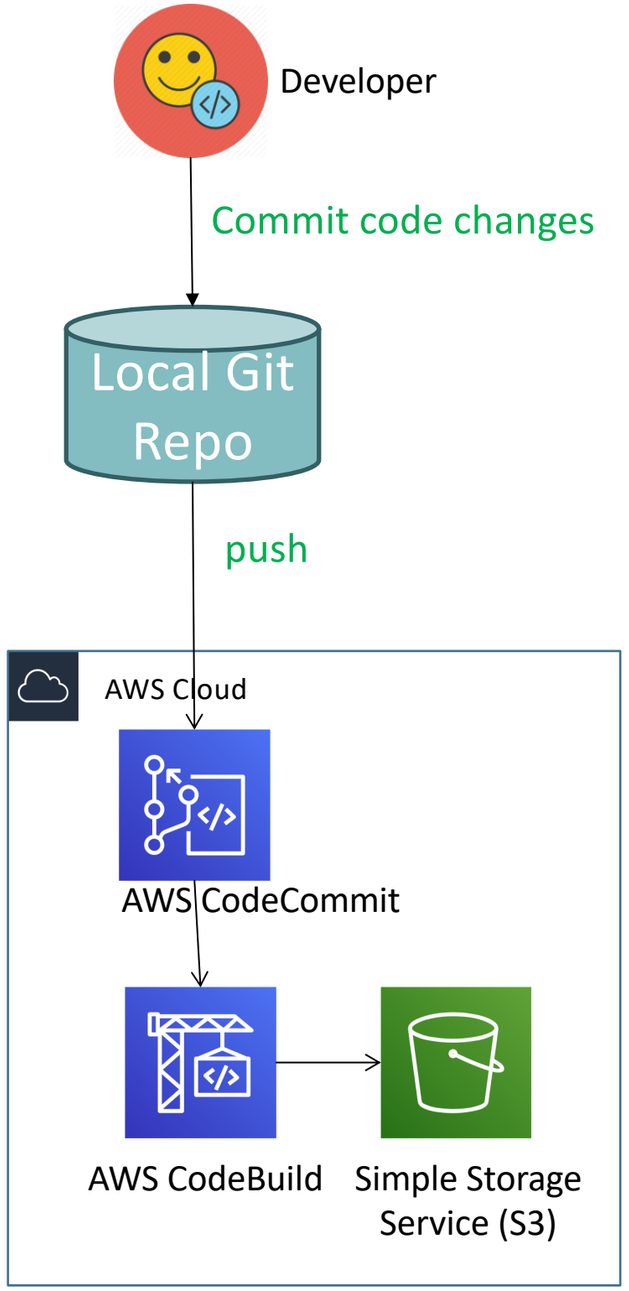


How CodeBuild works?




Source
AWS CodeCommit
Amazon Simple Storage Service (S3)
GitHub
GitHub Enterprise
Bitbucket
AWS Managed Image
Amazon EC2 Container Registry
External Container Registry (Docker Hub)
Build Environment
AWS CodeBuild
Amazon CloudWatch
Build Logs
Amazon Simple Storage Service (S3)
Build Artifacts
Amazon Simple Notification Service
Build Notifications
AWS CodeBuild Architecture

CodeBuild - Steps



AWS CodePipeline

CodePipeline - Introduction

- AWS CodePipeline is a continuous delivery service to model, visualize, and automate the steps required to release your software.
- Benefits
  - We can automate our release processes.
  - We can establish a consistent release process.
  - We can speed up delivery while improving quality.
  - Supports external tools integration for source, build and deploy.
  - View progress at a glance
  - View pipeline history details.

AWS CodePipeline Architecture

AWS CodePipeline

AWS CodeCommit

Amazon EC2 Container Registry

Simple Storage Service (S3)

GitHub

Source

AWS CodeBuild

Jenkins

Build

AWS CloudFormation

AWS CodeDeploy

AWS Elastic Beanstalk

AWS Service Catalog

Amazon Elastic Container Service

Amazon Elastic Container Service (Blue/Green)

Simple Storage Service (S3)

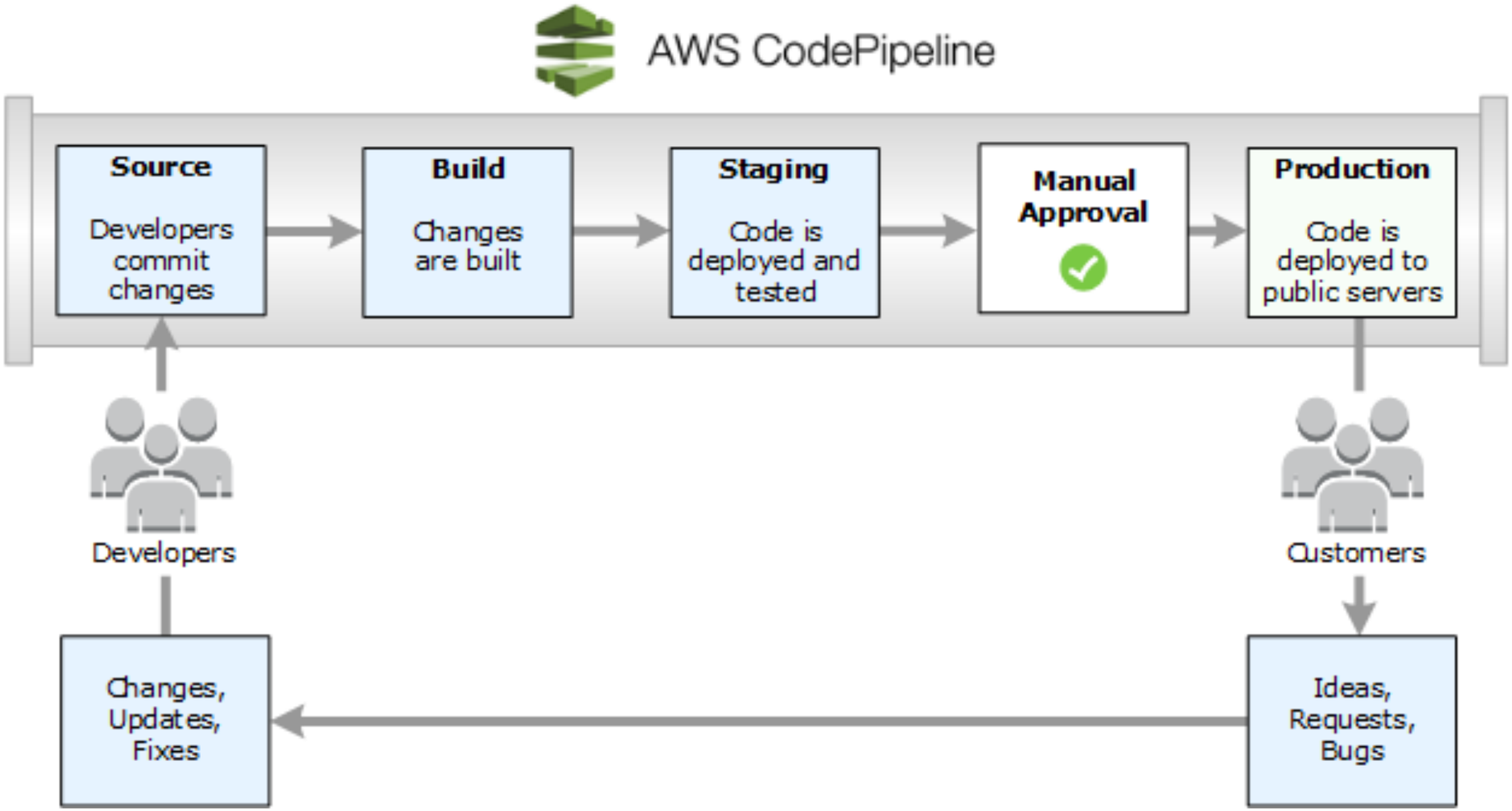
Deploy

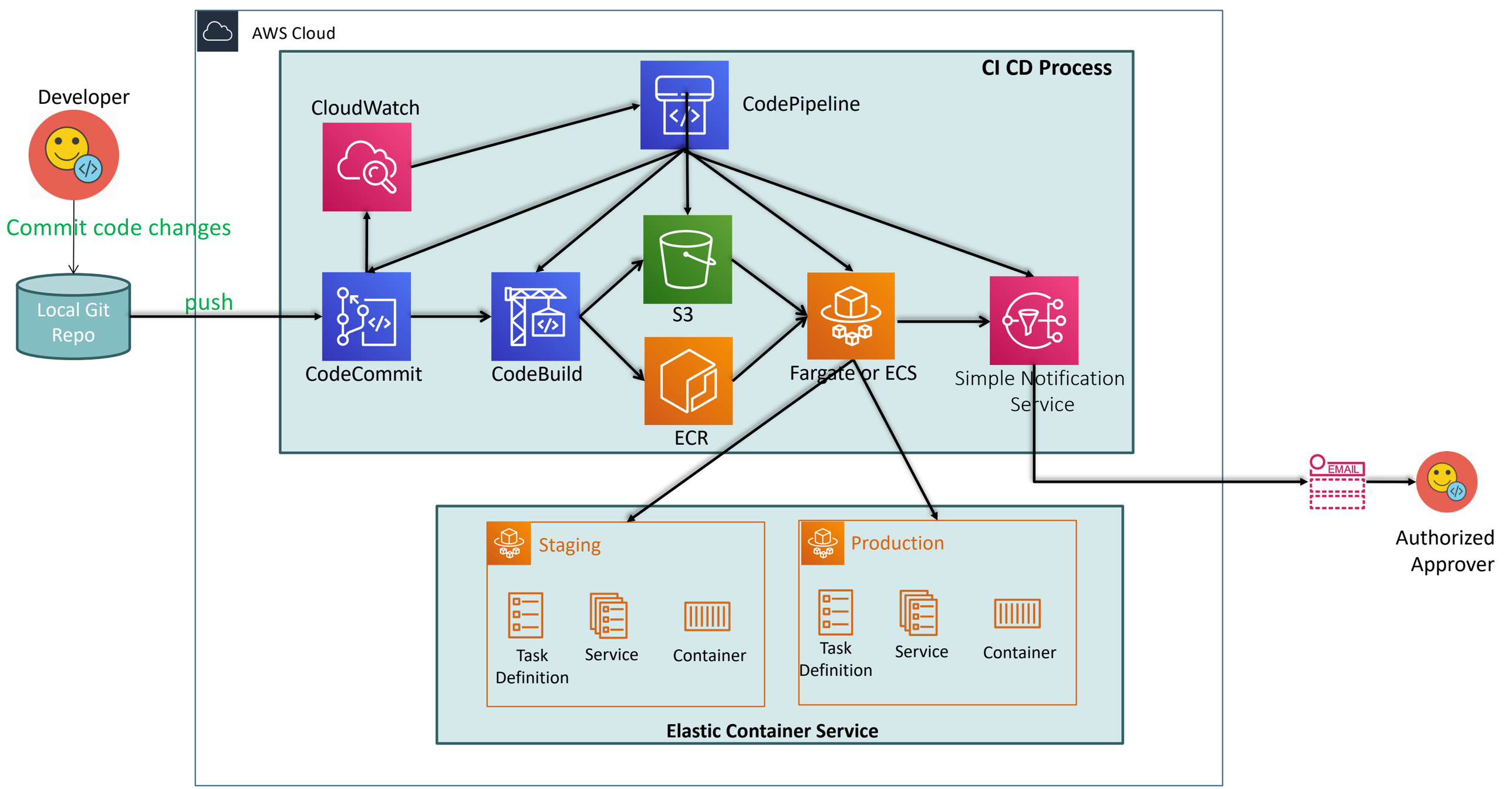
Amazon CloudWatch

GitHub Webhooks

Monitor Source Changes

Continuous Delivery




AWS Cloud
CI CD Process
Developer
Commit code changes
Local Git Repo
push
CloudWatch
CodePipeline
S3
CodeCommit
CodeBuild
ECR
Fargate or ECS
Simple Notification Service
EMAIL
Authorized Approver
Staging
Task Definition
Service
Container
Production
Task Definition
Service
Container
Elastic Container Service

AWS Fargate & ECS

What are Microservices?

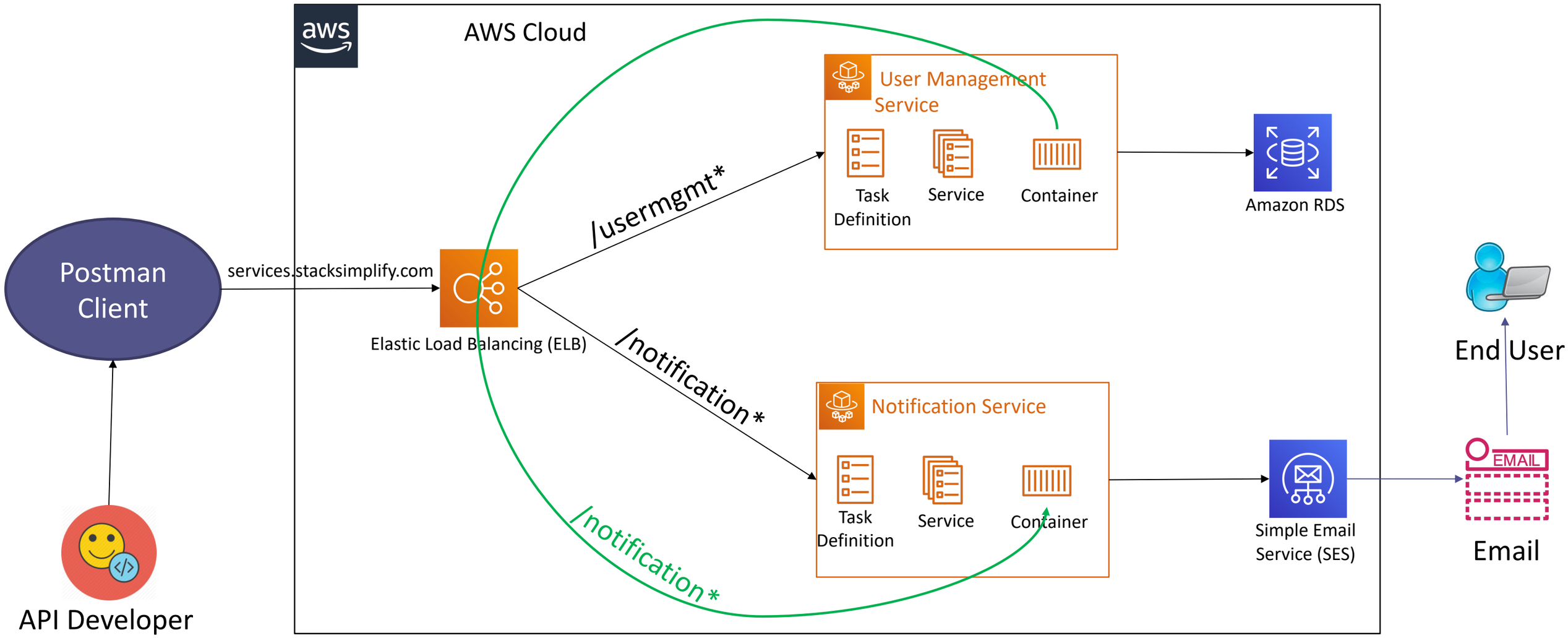
What are Microservices?

- Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are
  - Highly maintainable and testable
  - Loosely coupled
  - Independently deployable
  - Organized around business capabilities
  - Owned by a small team

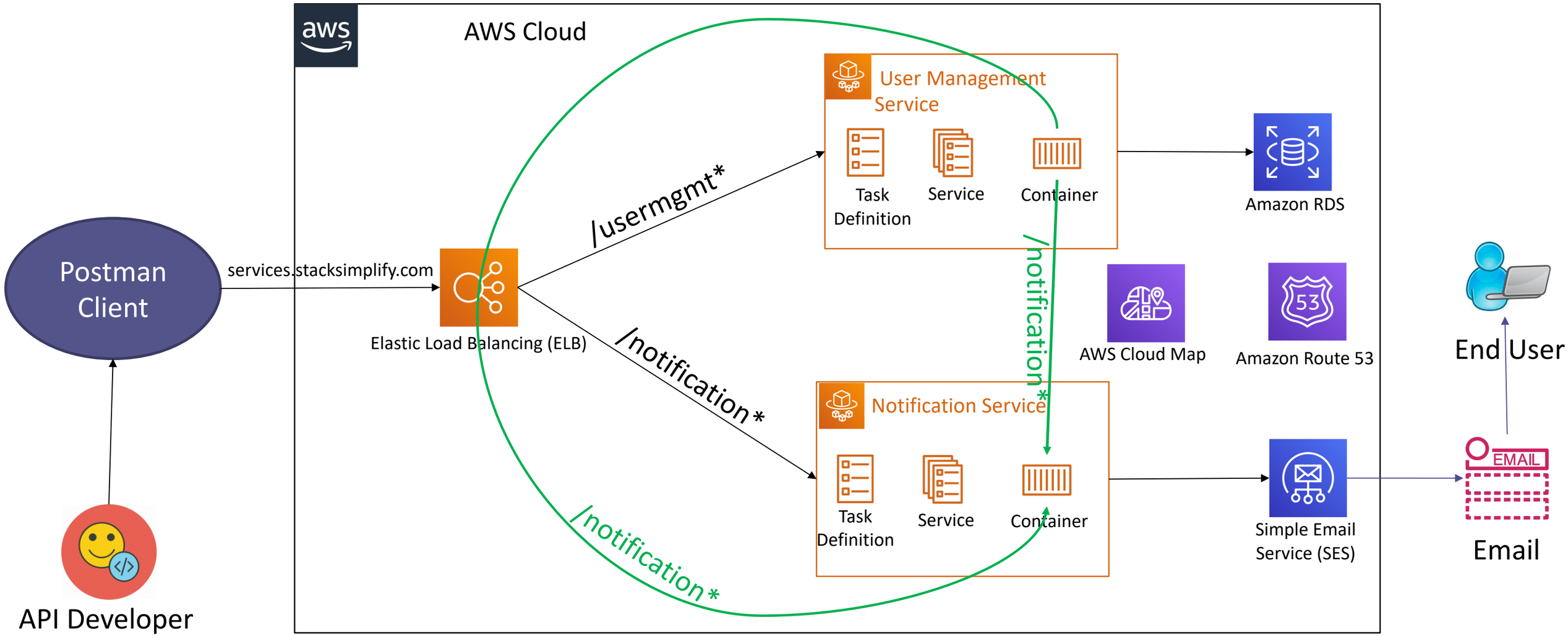
Microservices - Benefits

- **Developer independence**: Small teams work in parallel and can iterate faster than large teams.
- **Isolation and resilience**: If a component dies, you spin up another while and the rest of the application continues to function.
- **Scalability**: Smaller components take up fewer resources and can be scaled to meet increasing demand of that component only.
- **Lifecycle automation**: Individual components are easier to fit into continuous delivery pipelines and complex deployment scenarios not possible with monoliths.
- **Relationship to the business**: Microservice architectures are split along business domain boundaries, increasing independence and understanding across the organization.

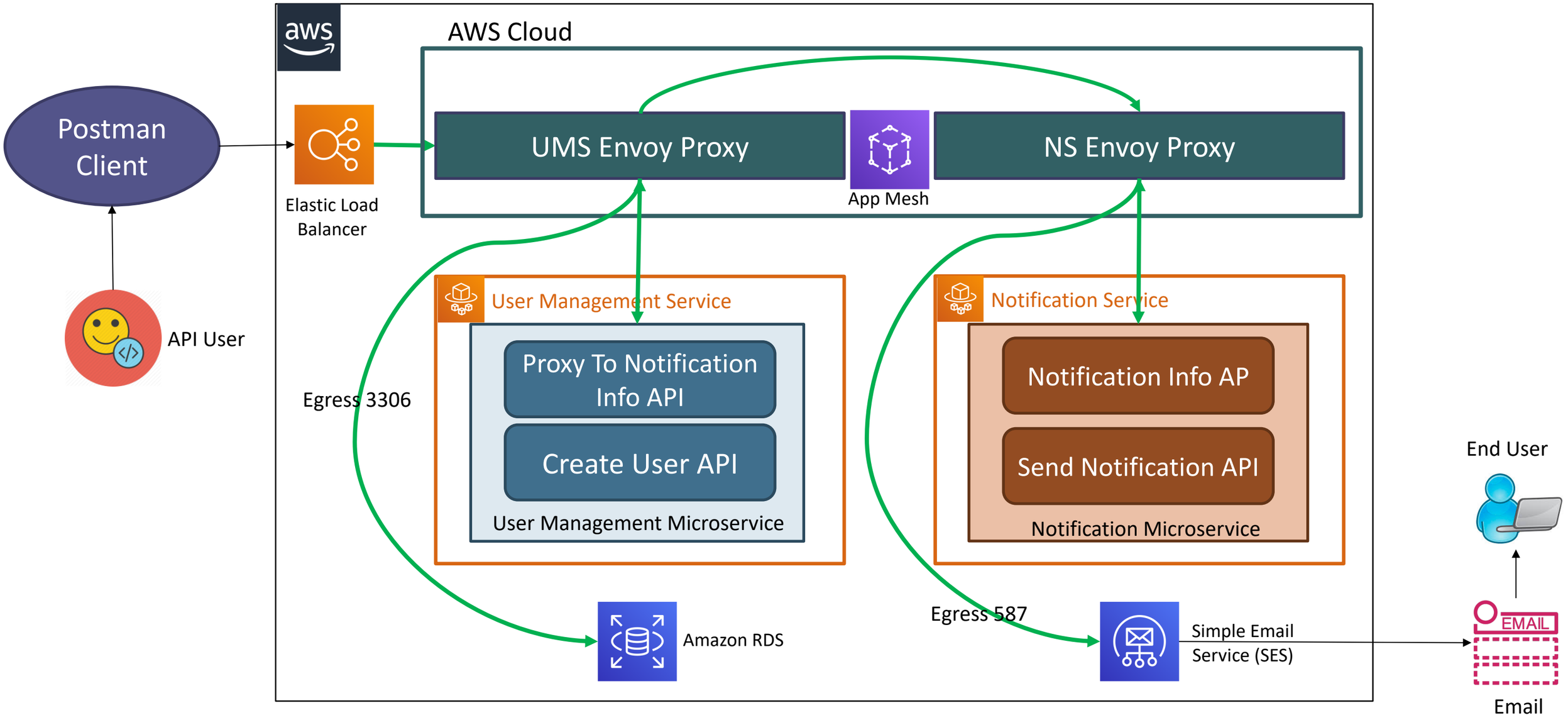
Microservices Deployment on AWS ECS – No Service Discovery



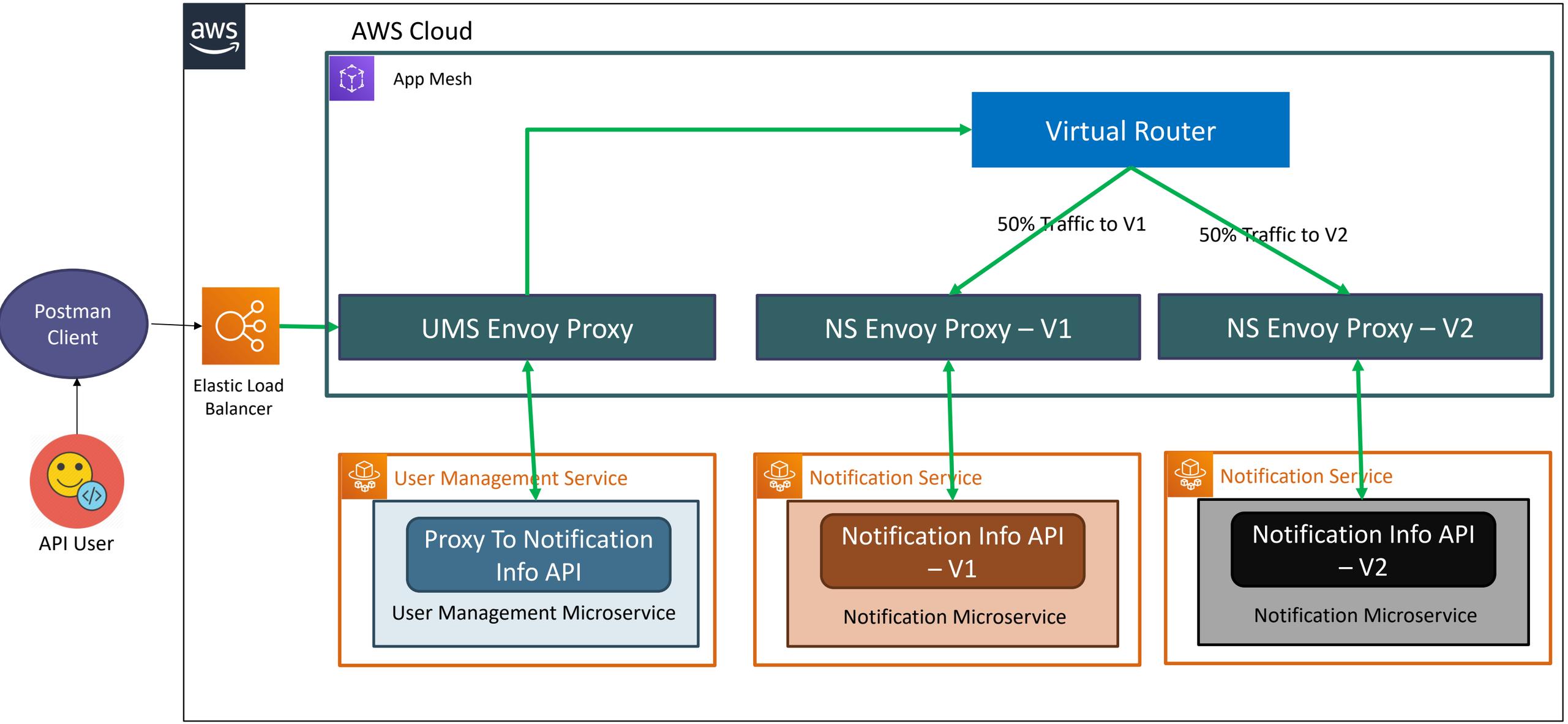
Microservices Deployment on ECS - with Service Discovery



Microservices – with AWS AppMesh on ECS



Microservices – Canary Deployments with AppMesh on ECS



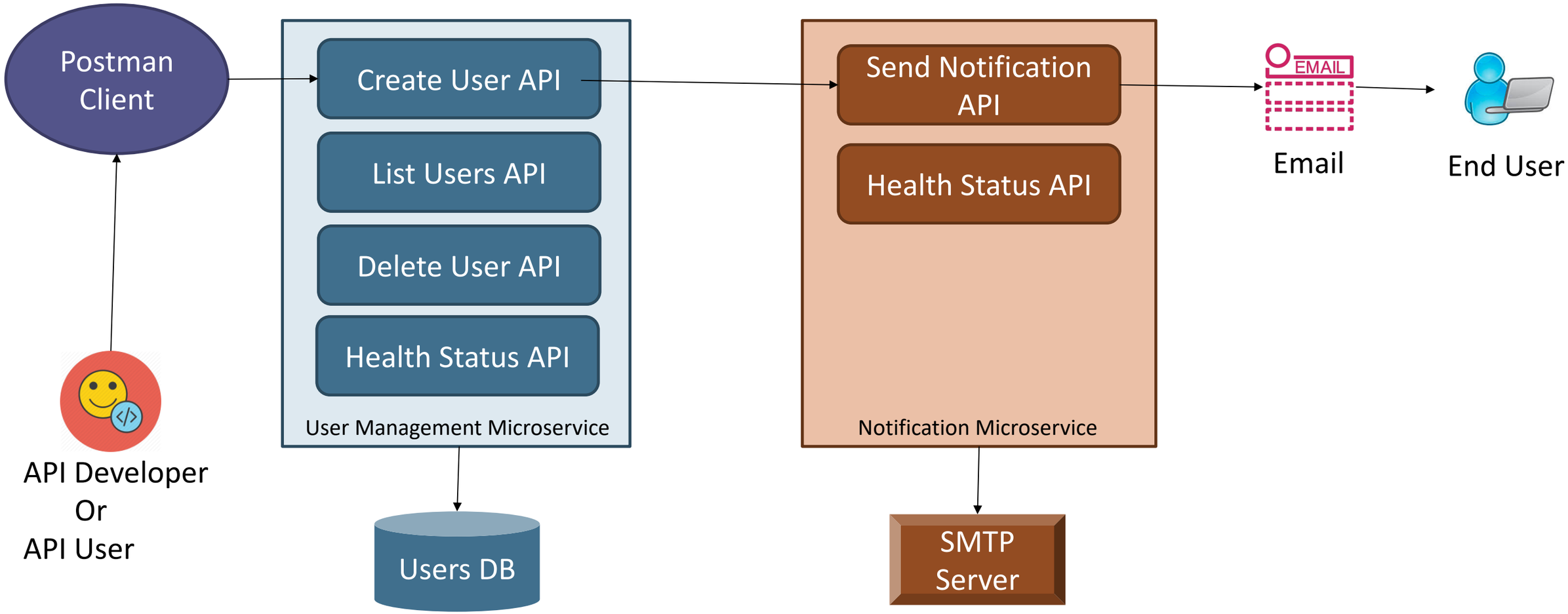
AWS Fargate & ECS

Microservices Deployment

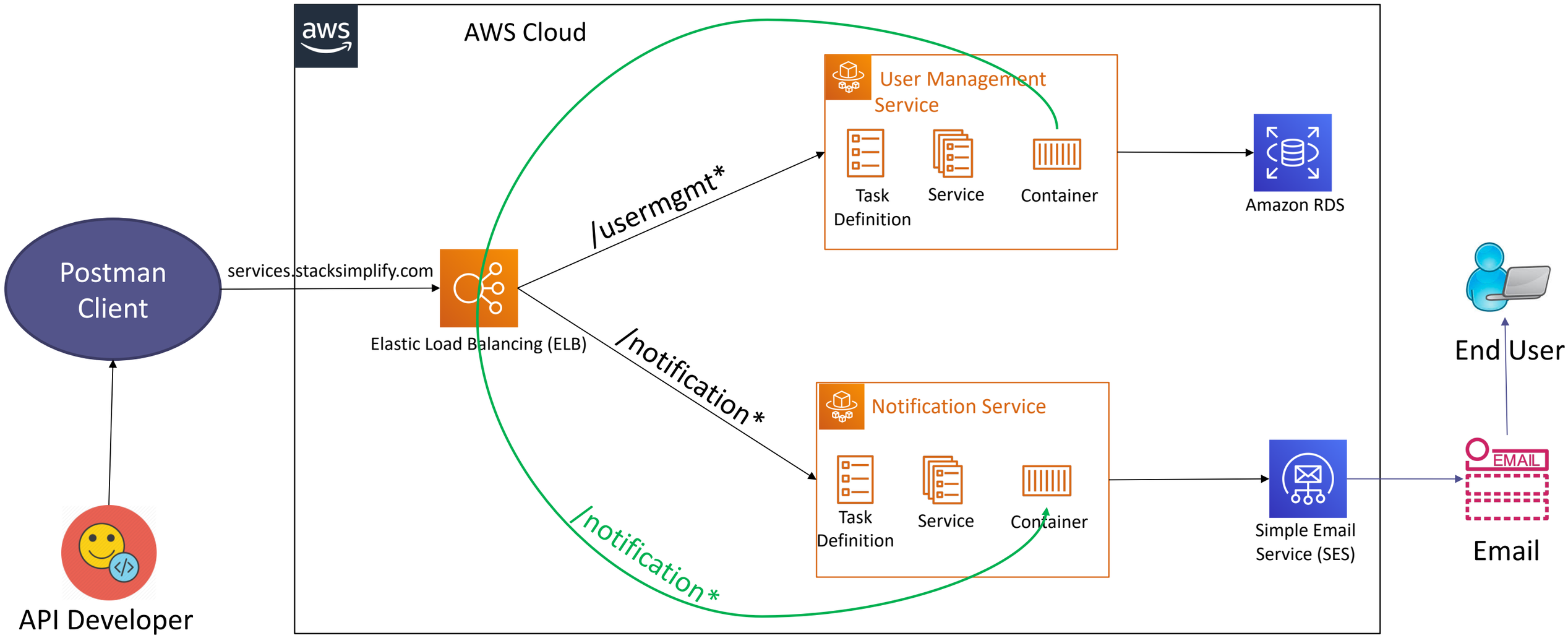
Microservices

- User Management Microservice
- Notification Microservice

Microservices

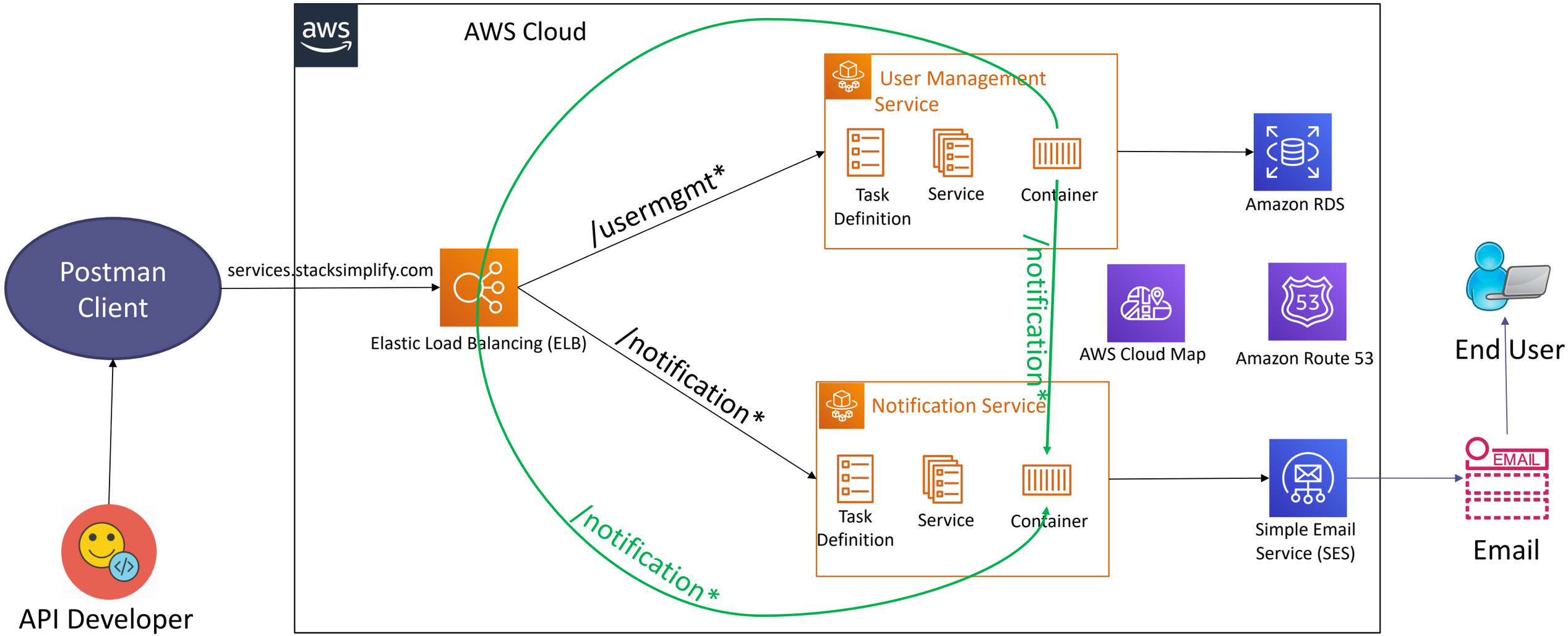


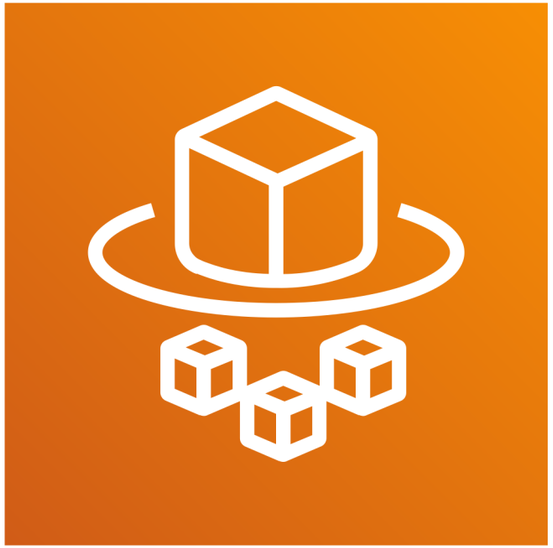
Microservices Deployment on AWS ECS



AWS Fargate & ECS
Microservices
Service Discovery

Microservices Deployment on ECS with Service Discovery



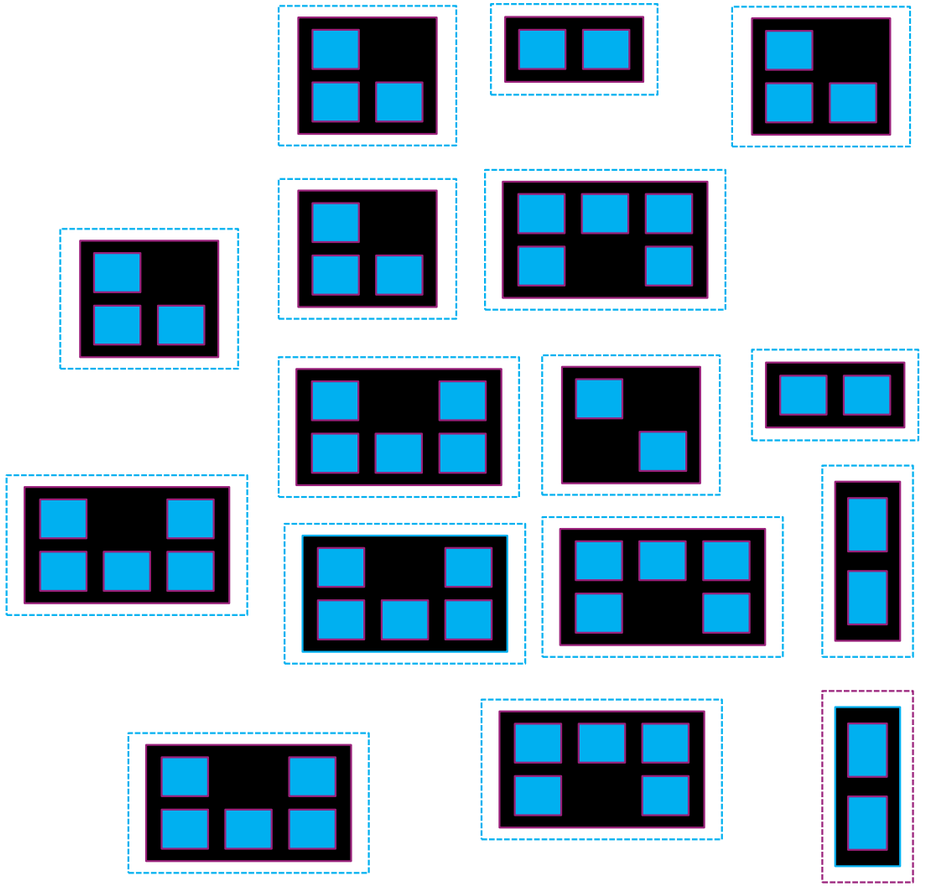


AWS Fargate & ECS

Cloud Map

Complexity of modern architectures

- Wide variety of resources
- Complexity grows exponentially
- Multiple versions and stages coexist
- Infrastructure scales dynamically
- Unhealthy resources are replaced



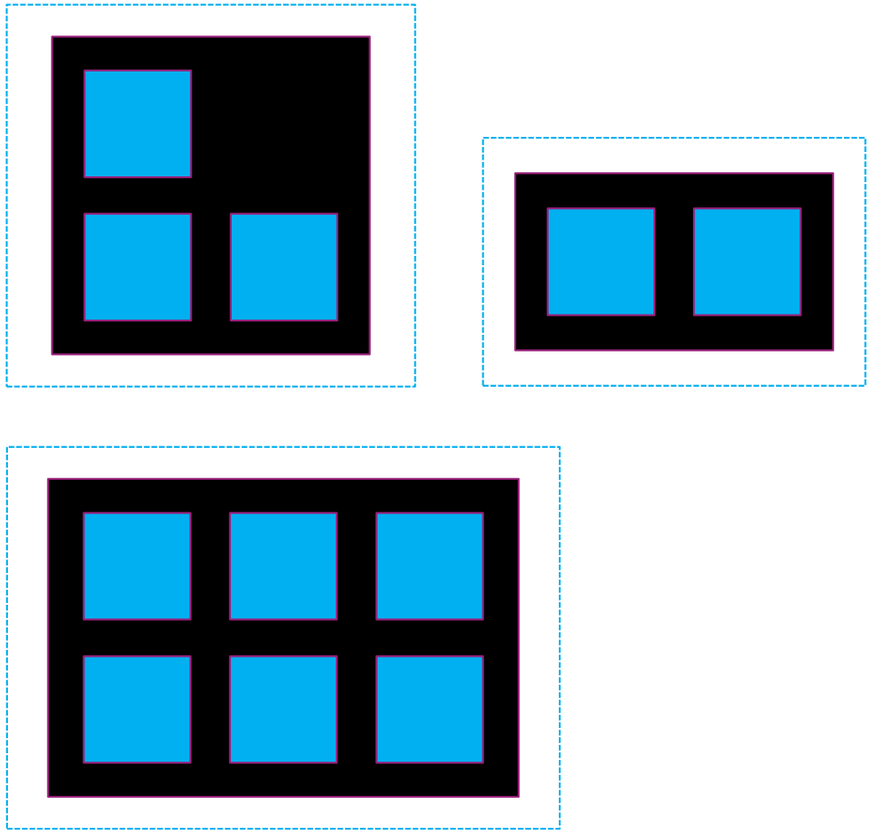
How to find resources to connect to?

Service Discovery

Finding the location of a service provider

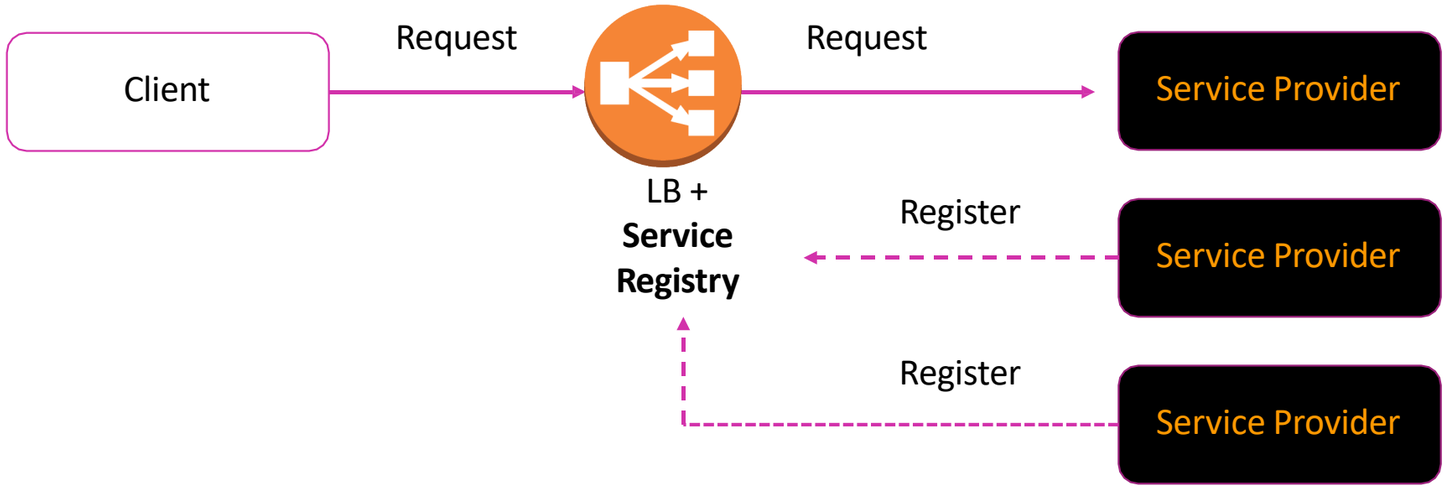
myapp: {10.20.30.4:8080, 10.20.30.6:8080}

mylogs: {S3bucket1, S3bucket2}



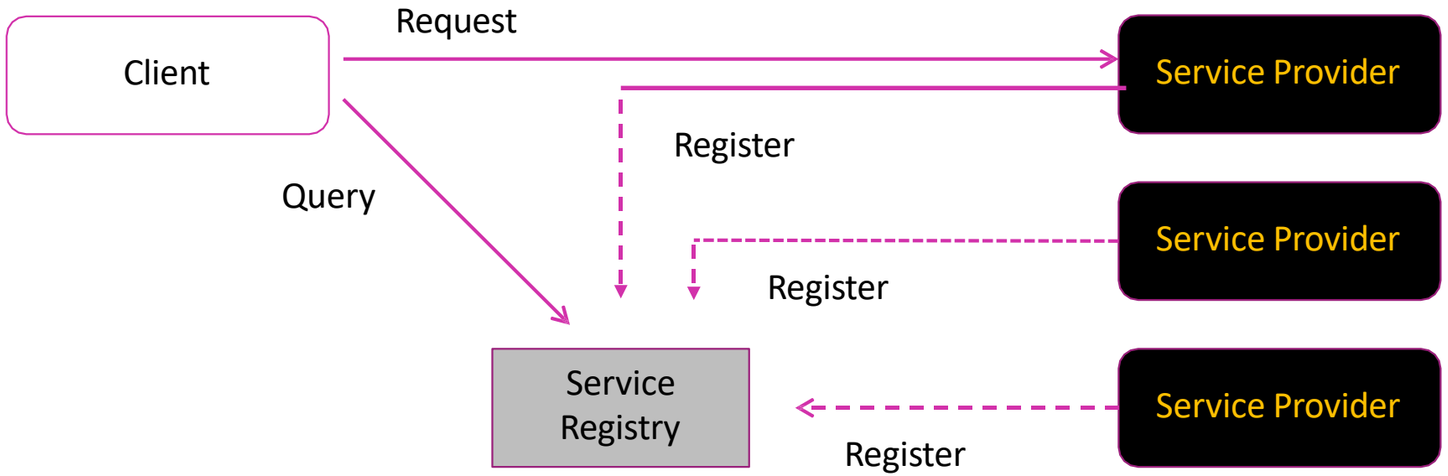
Server-side service discovery pattern

- Connections are proxied
- Discovery is abstracted away
- Availability and capacity impact
- Additional latency

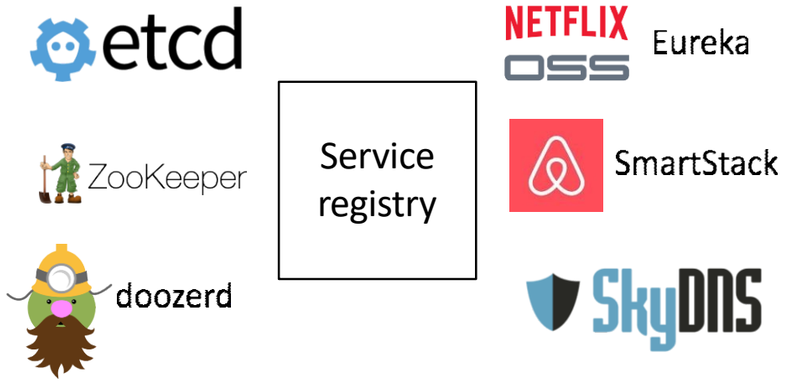


Client-side service discovery pattern

- Clients connect directly to providers
- Fewer components in the system
- Clients must be registry-aware
- Client-side load balancing



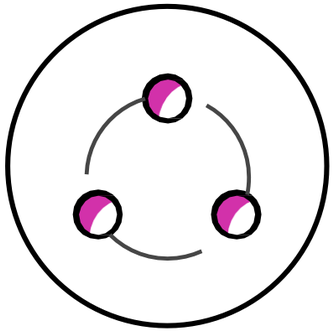
Existing solutions require setup and management



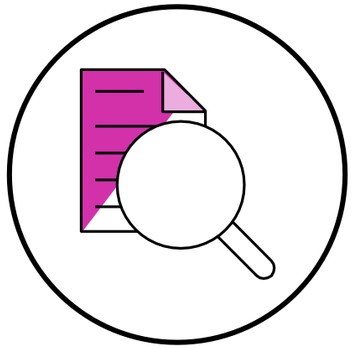
Service Registries

AWS Cloud Map

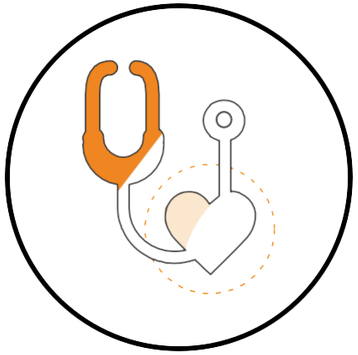
Build the dynamic map of your cloud



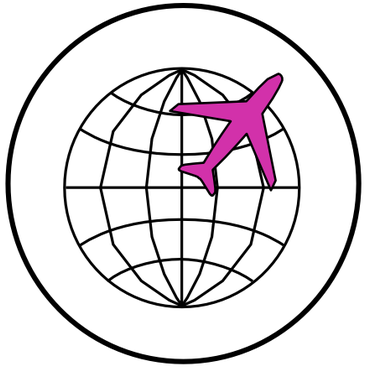
Define convenient names for all cloud resources



Discover resources with specific attributes



Ensure only healthy resources are discovered



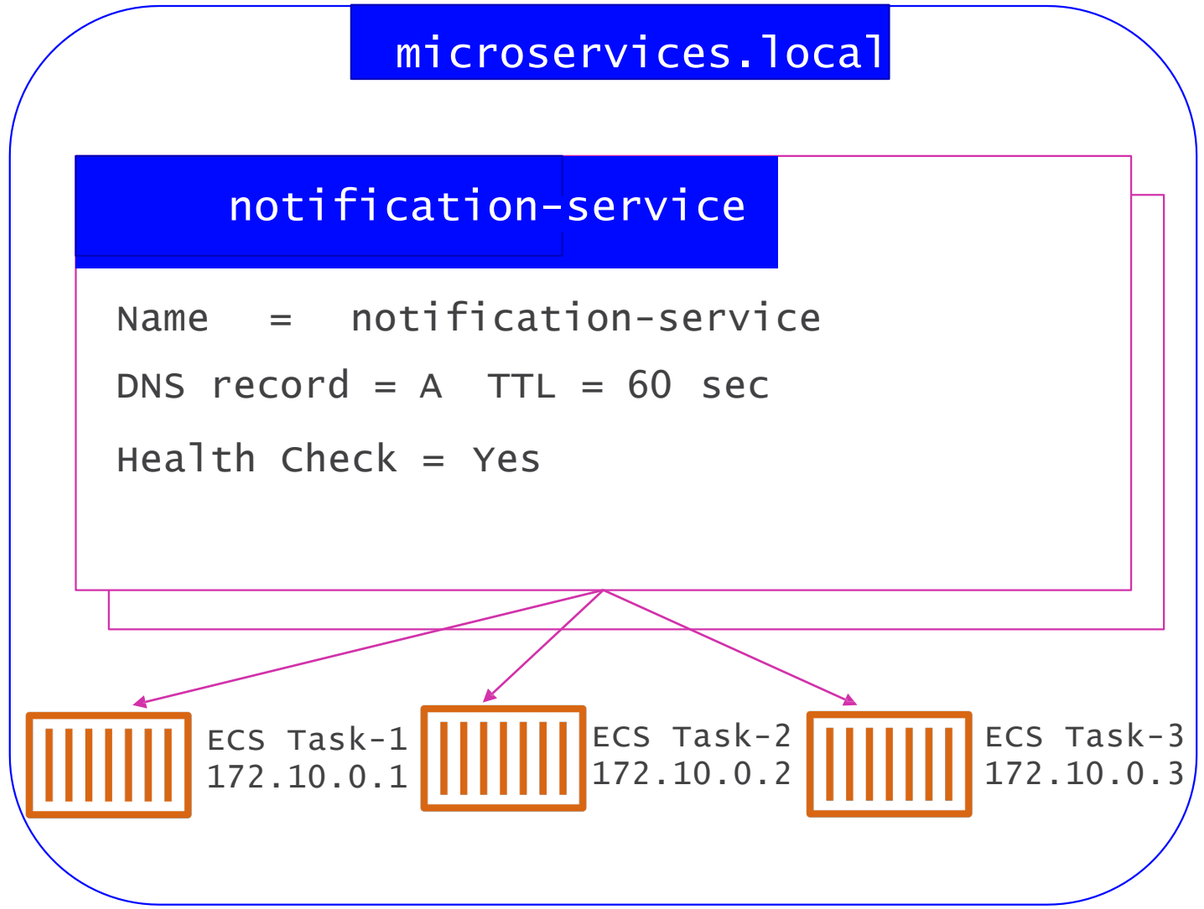
Use highly available DNS and regional API

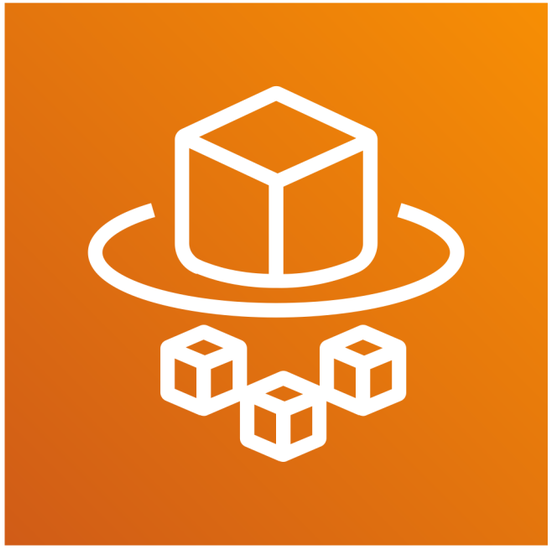
AWS Cloud Map - Introduction

- AWS Cloud Map is a cloud resource discovery service.
- With Cloud Map, you can define custom names for your application resources, and it maintains the updated location of these dynamically changing resources.
- This increases your application availability because your web service always discovers the most up-to-date locations of its resources.
- Cloud Map allows you to register any application resources, such as databases, queues, microservices, and other cloud resources, with custom names.
- Cloud Map then constantly checks the health of resources to make sure the location is up-to-date.
- The application can then query the registry for the location of the resources needed based on the application version and deployment environment.

AWS Cloud Map registry

- Namespace
- Service
- Service Instance






AWS Fargate & ECS

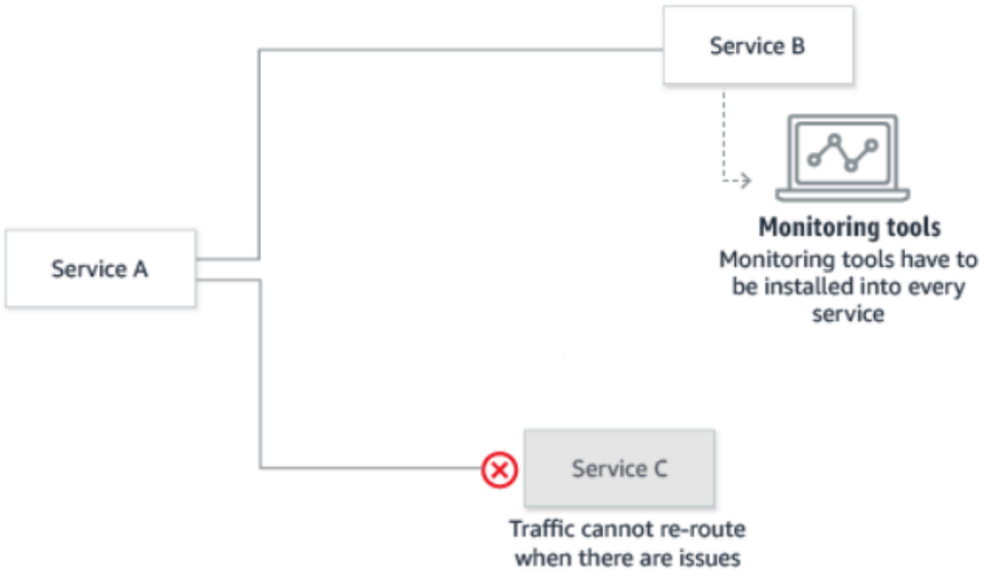
Microservices & App Mesh

AWS App Mesh

How it works

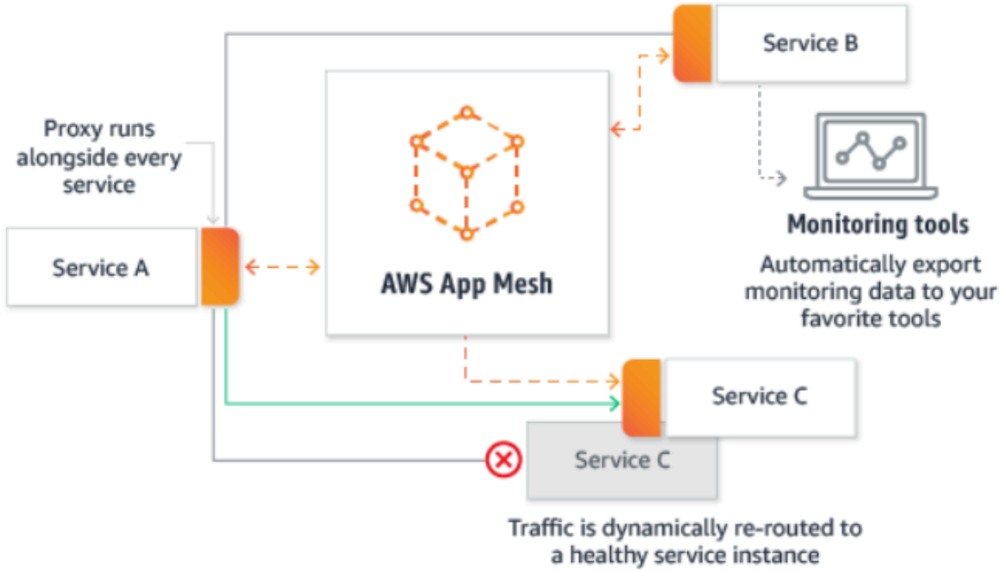
Before App Mesh

Communications and monitoring are manually configured for every service.



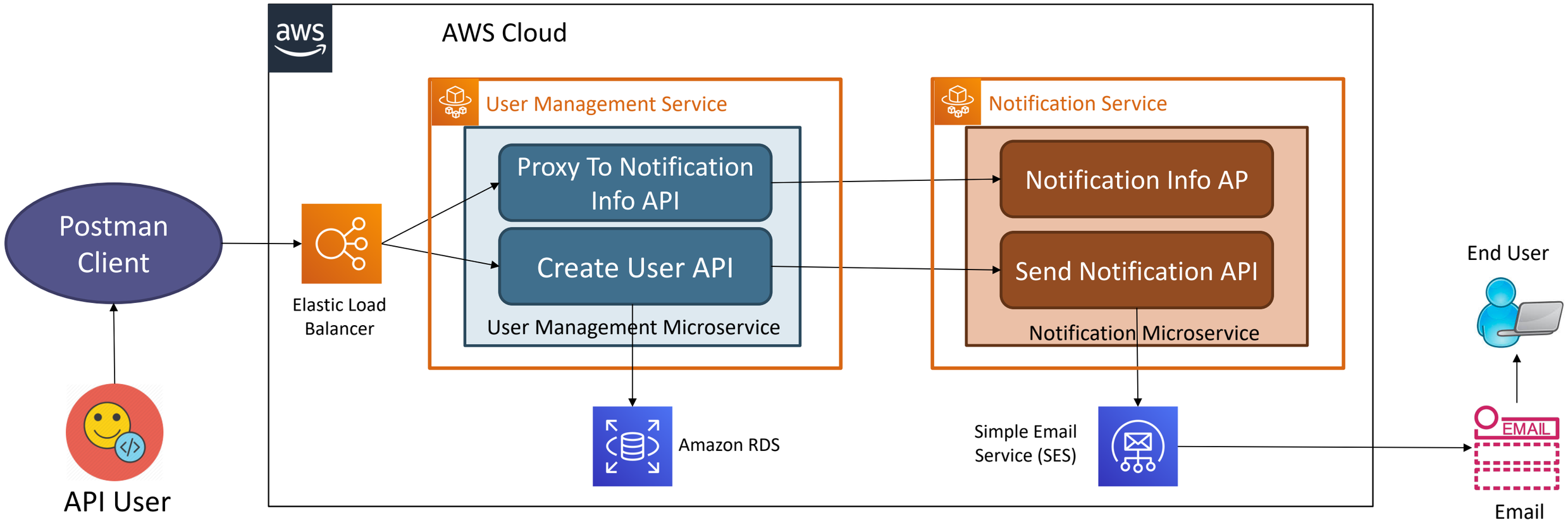
After App Mesh

App Mesh configures communications and monitoring for all services.

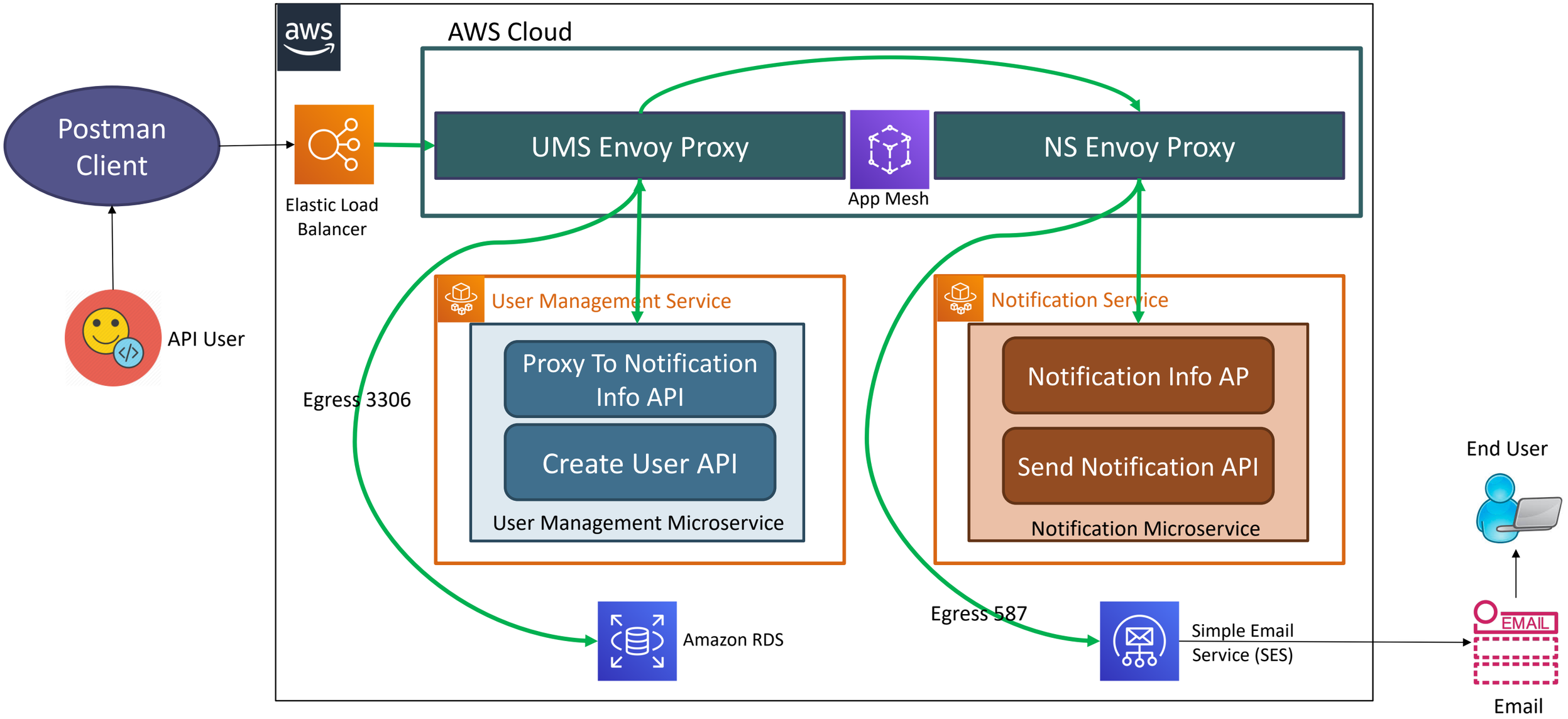


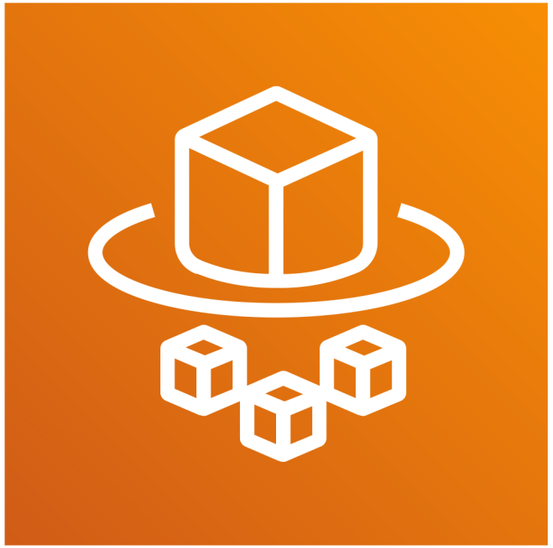
Refernce: https://aws.amazon.com/app-mesh/

Microservices – without AWS AppMesh on ECS



Microservices – with AWS AppMesh on ECS

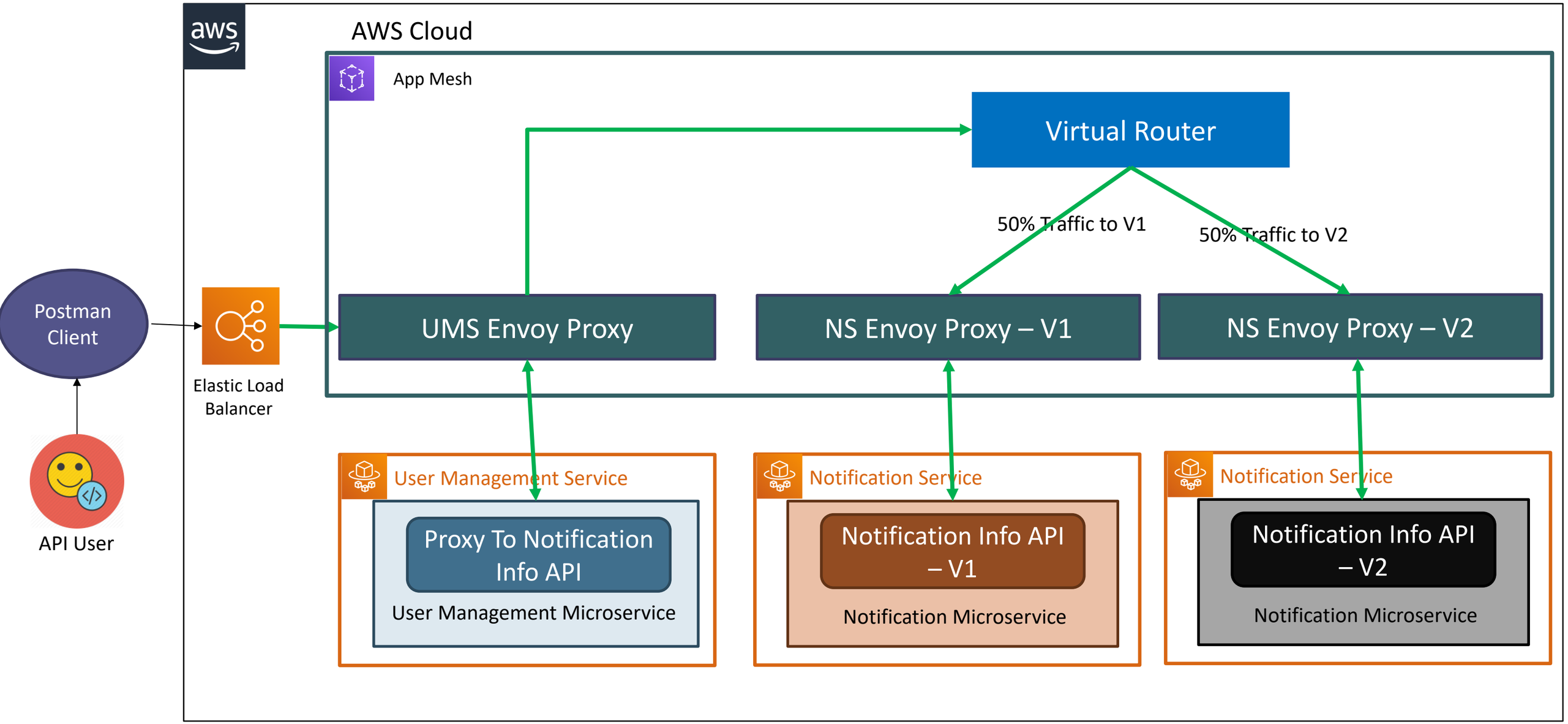


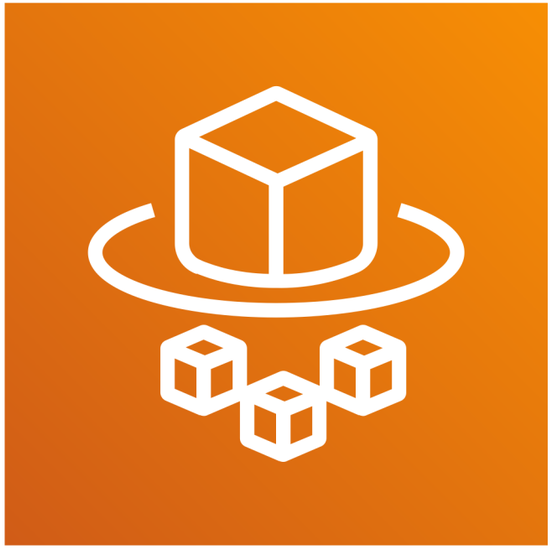



AWS Fargate & ECS

Microservices Canary Deployments with App Mesh

Microservices – Canary Deployments with AppMesh on ECS

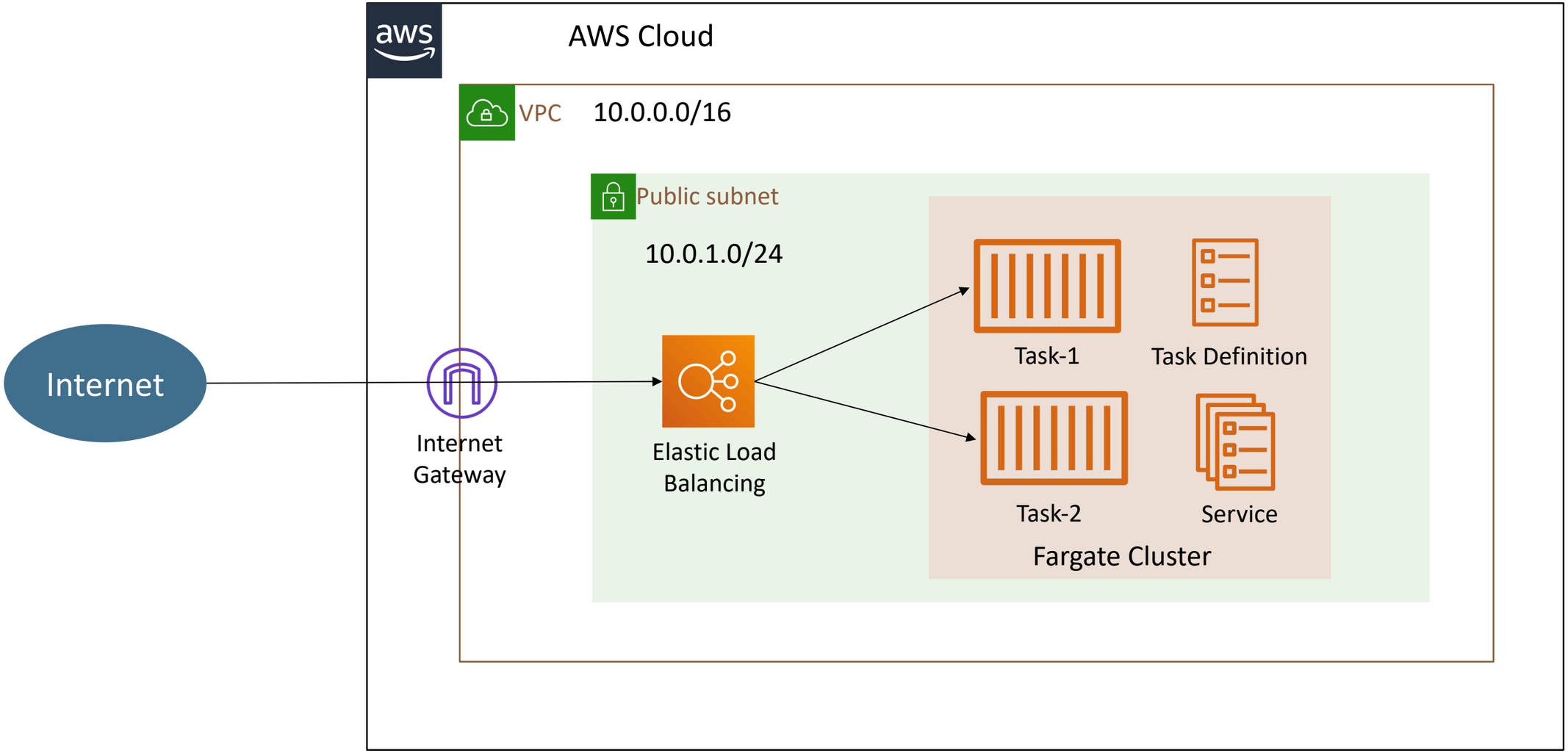




AWS Fargate & ECS

CloudFormation

Fargate Tasks – Public Subnet in a VPC



Thank You