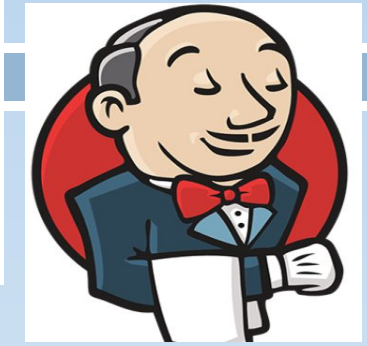
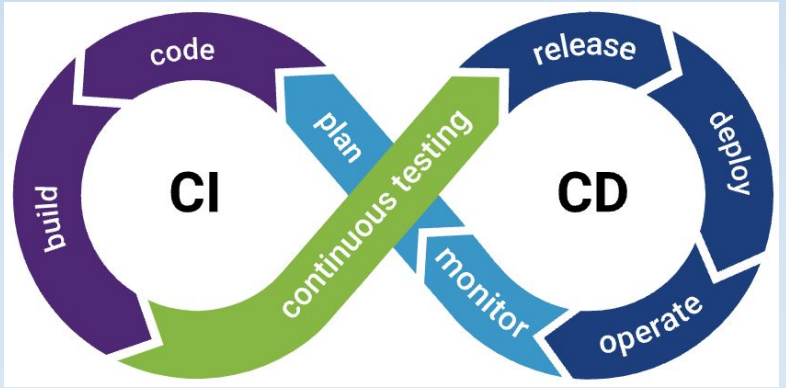


DevOps Atölyesi



Hakan BAYRAKTAR

Eğitim İçeriği

21 Mayıs | GitHub Actions ile CI/CD'ye Giriş (Python Flask App - AWS ECS)

- DevOps ve CI/CD Kavramlarına Giriş
- Github Actions Nedir? Github Actions Temel Kavramları
- Docker, AWS ECS Hakkında Kısa Bilgi

Uygulamalı Proje: Github Actions Kullanarak Python Flask Uygulamasının AWS ECS Ortamına Otomatik Olarak Deploy Edilmesi

28 Mayıs | GitHub Actions ve ArgoCD ile GitOps Workflow (Node.js - GKE)

- Gitops Nedir? ArgoCD Temel Kavramları
- GKE(Google Kubernetes Engine) Hakkında Temel Bilgiler

Uygulamalı Proje: GitHub Actions ve ArgoCD ile Node.js Uygulamasının GKE Cluster'a Otomatik Olarak Deploy Edilmesi

Eğitim İçeriği

4 Haziran | Jenkins ile 3-Tier Application Deploy (Django App – Docker on EC2)

- Jenkins Nedir? Jenkins Kavramları
- 3-Tier Application Mimarisi
- EC2, Autoscaling, Load Balancer ,RDS Servisleri Hakkında Temel Bilgiler

Uygulamalı Proje: Jenkins ile Django + React Uygulamasının AWS EC2 Sunucularına 3-Tier CI/CD Pipeline ile Dağıtılması

11 Haziran | Jenkins ile Blue/Green Deployment (Java App - AWS EKS)

- Blue/Green Deployment Mimarisi Nedir?
- Kubernetes ve AWS EKS Temel Bilgiler
- Maven, Sonarqube, Trivy Hakkında Kısa Bilgiler
- Bitirme Ödevi Tanıtımı

Uygulamalı Proje: Java Uygulamasının Jenkins'de Blue/Green Metodoloji Kullanılarak AWS EKS Kubernetes Cluster Ortamına Deploy Edilmesi

GitHub Actions ile CI/CD'ye Giriş (Python Flask App - AWS ECS)

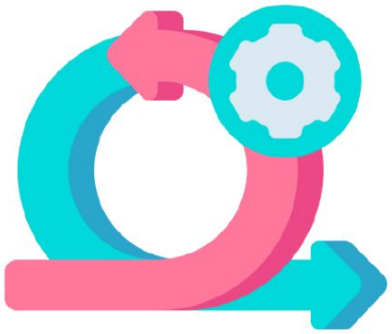
- DevOps
- CI/CD
- Github Actions
- Docker
- AWS ECS

Uygulamalı Proje: Github Actions Kullanarak Python Flask Uygulamasının AWS ECS Ortamına Otomatik Olarak Deploy Edilmesi

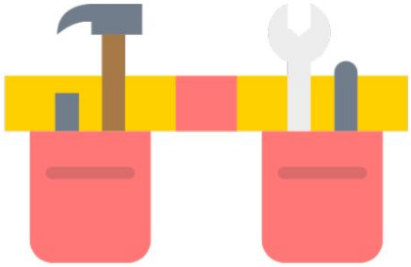
What is Devops?

Way of software development

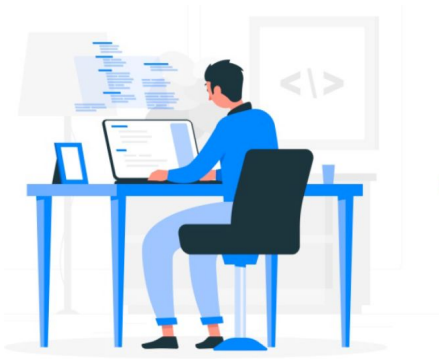
Values & Principle



Methodologies

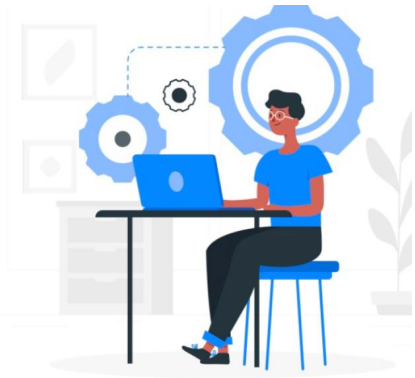


Tools



Developers

DevOps

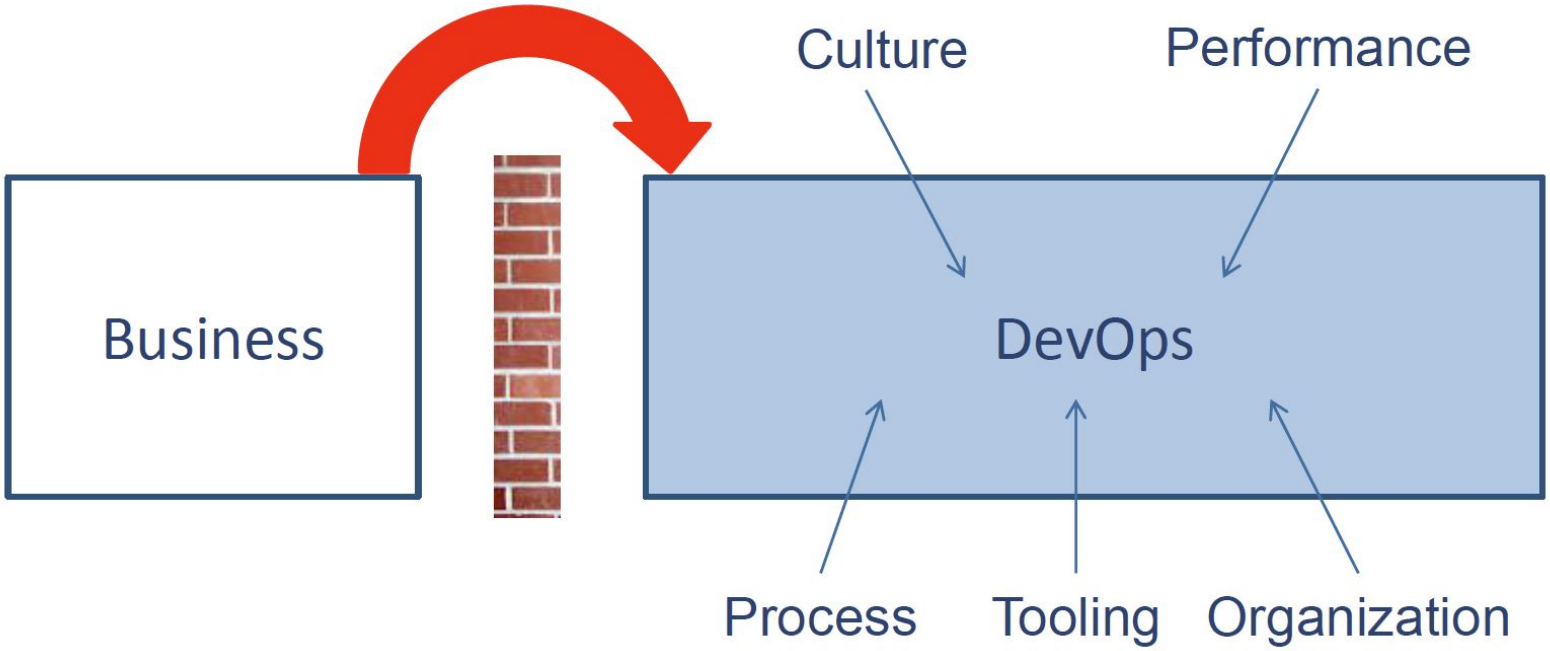


Operations

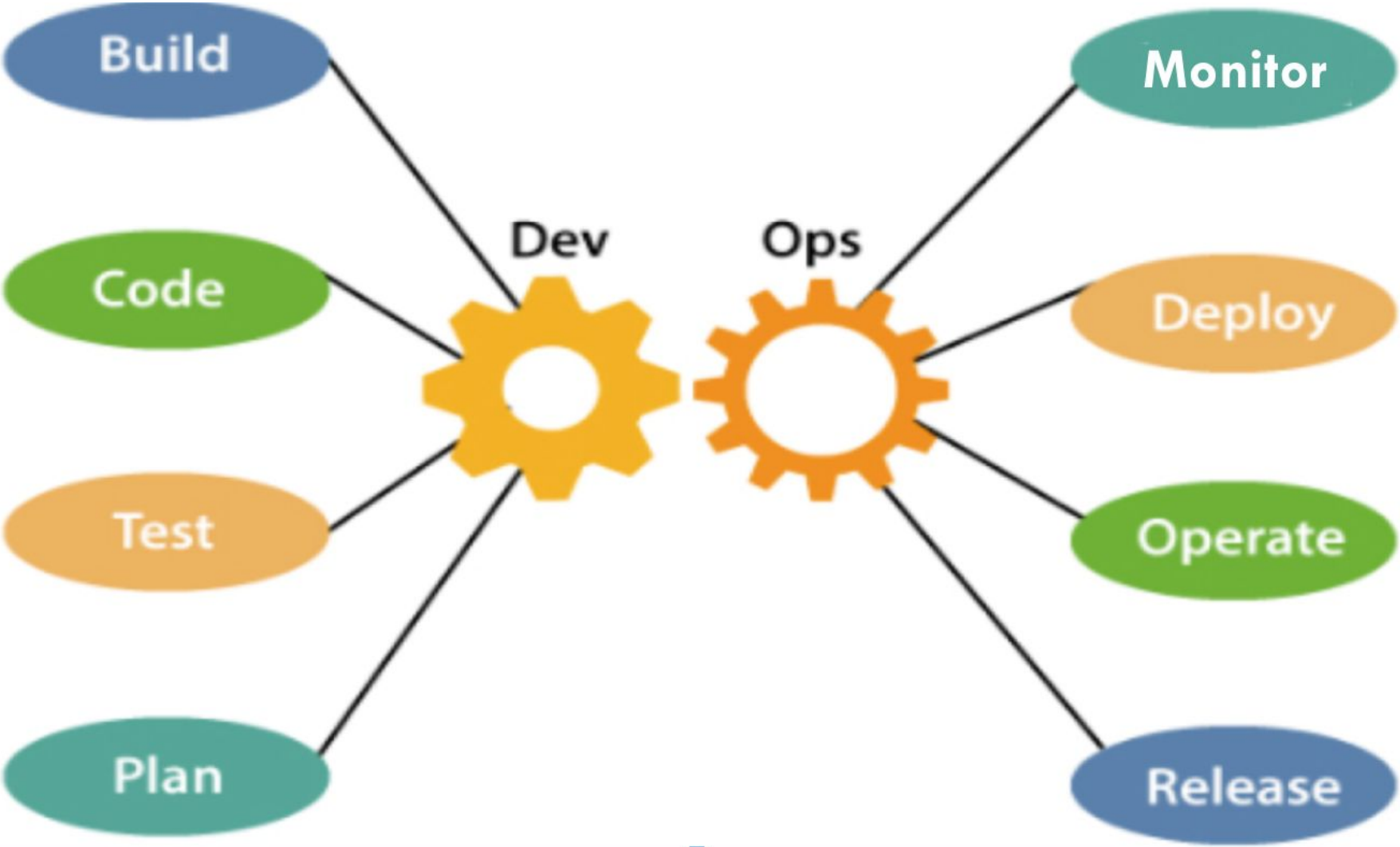
Devops'un Ana Hedefleri

DevOps, yazılım geliştirme ve operasyon ekiplerini bir araya getirerek dört temel hedefe ulaşmayı amaçlayan bir hareket veya yaklaşımdır:

- Improve **Quality** of delivery (First time right)
- Deliver higher **Business Value**(Customer)
- Increase **Speed** of delivery (Performance)
- Improve **Efficiency** (Reduce Cost)

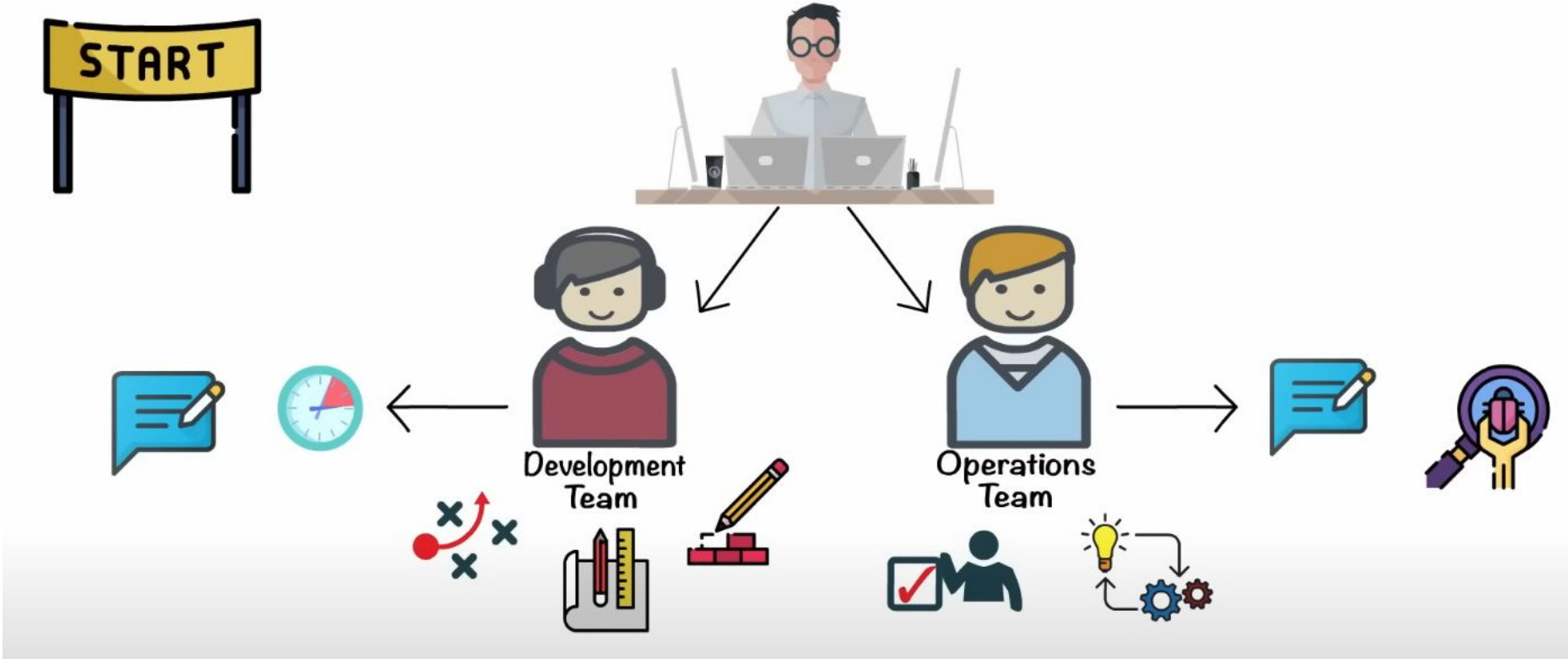


DevOps Components



Who is a DevOps Engineer?

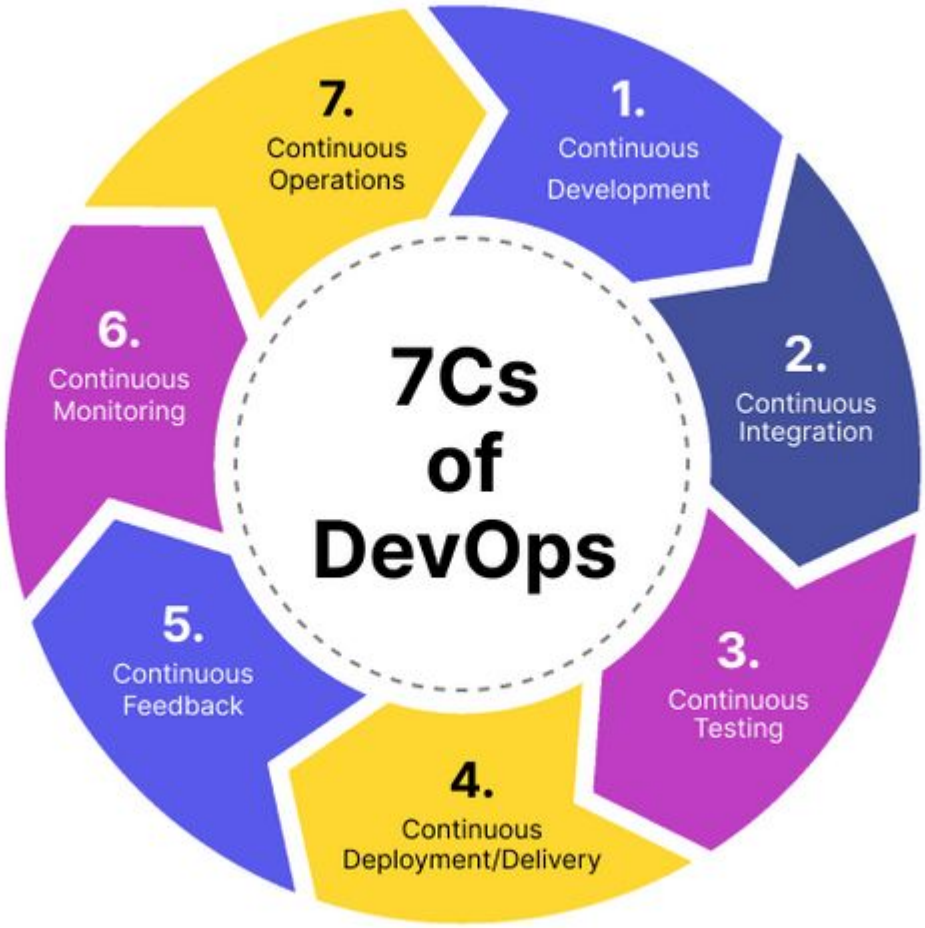
DevOps mühendisleri, **hem IT operasyonları hem de yazılım geliştirme alanlarında bilgi sahibi** olan profesyonellerdir.



Devops Mühendisinin Görevleri

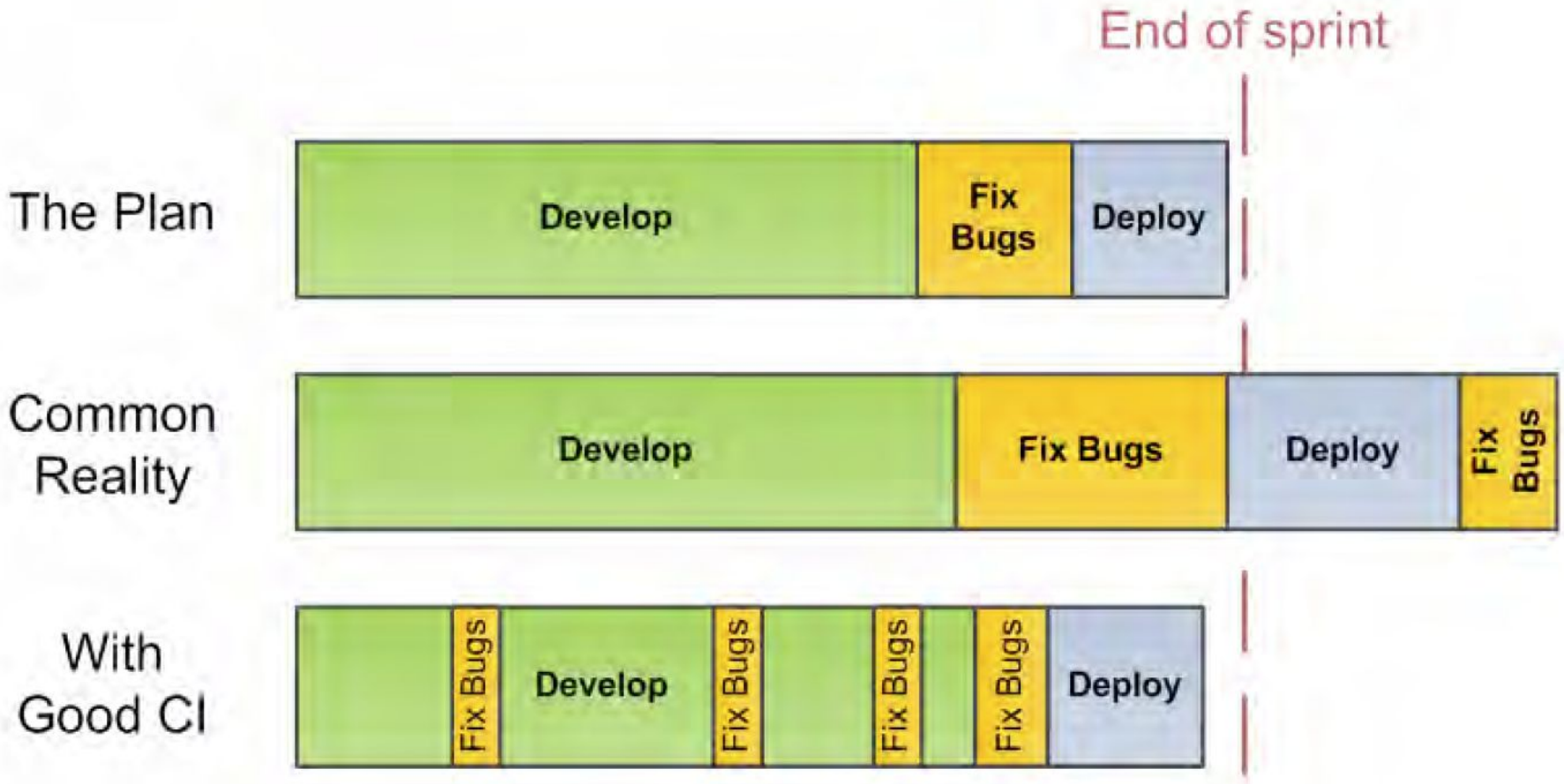
- Otomasyon
- CI/CD Pipeline Kurma
- Konteyner & Orkestrasyon
- Infrastructure as Code
- Monitoring ve Log Yönetimi
- Cloud Yönetimi
- Güvenlik
- Süreç & İletişim

Devops Lifecycle



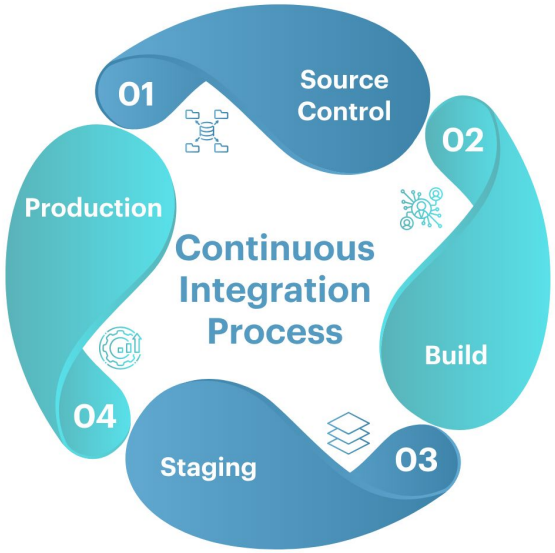
Continuous Integration

Sürekli entegrasyon, yazılım geliştirme sürecini hızlandırırken, kaliteyi artırmak ve hata oranını azaltmak için kritik bir yöntemdir.



Continuous integration process

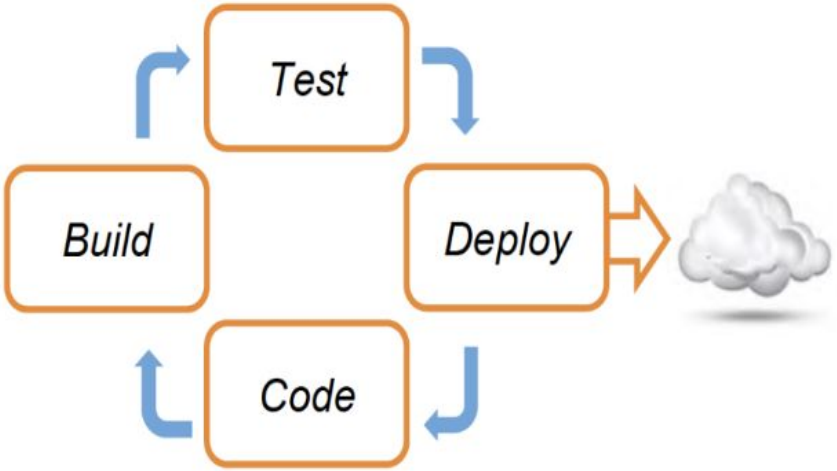
- Source Control (Commit Change)
- Build (Run Build And Unit Tests)
- Staging (Deploy To Test Environment And Run Tests)
- Production (Deploy To Production Environment)



Continuous Delivery & Continuous Deployment

Continuous Delivery (CD)

Sürekli teslimat, yazılımın her zaman üretim ortamına geçmeye hazır hale getirilmesini ifade eder.



Continuous Deployment

Sürekli dağıtım, her yeni kod güncellemesinin otomatik olarak üretim ortamına aktarılması sürecidir.

Continuous (Integration Delivery Deployment)

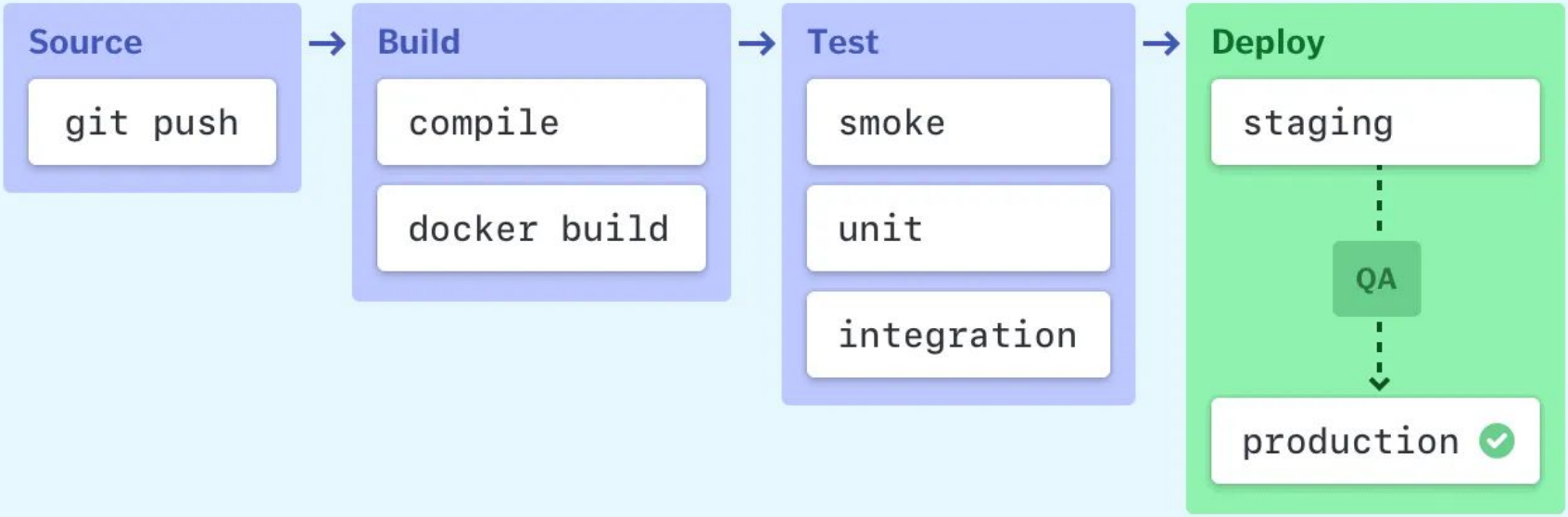
Continuous Integration

Continuous Delivery

Continuous Deployment

Continuous Delivery'de deploy işlemi manuel olurken
Continuous Deployment'da deploy işlemi otomatik yürütülmektedir.

CI/CD Pipeline



CI/CD Tools

Open Source

Jenkins | GoCD | GitLab CI | Drone CI | Spinnaker | Buildbot

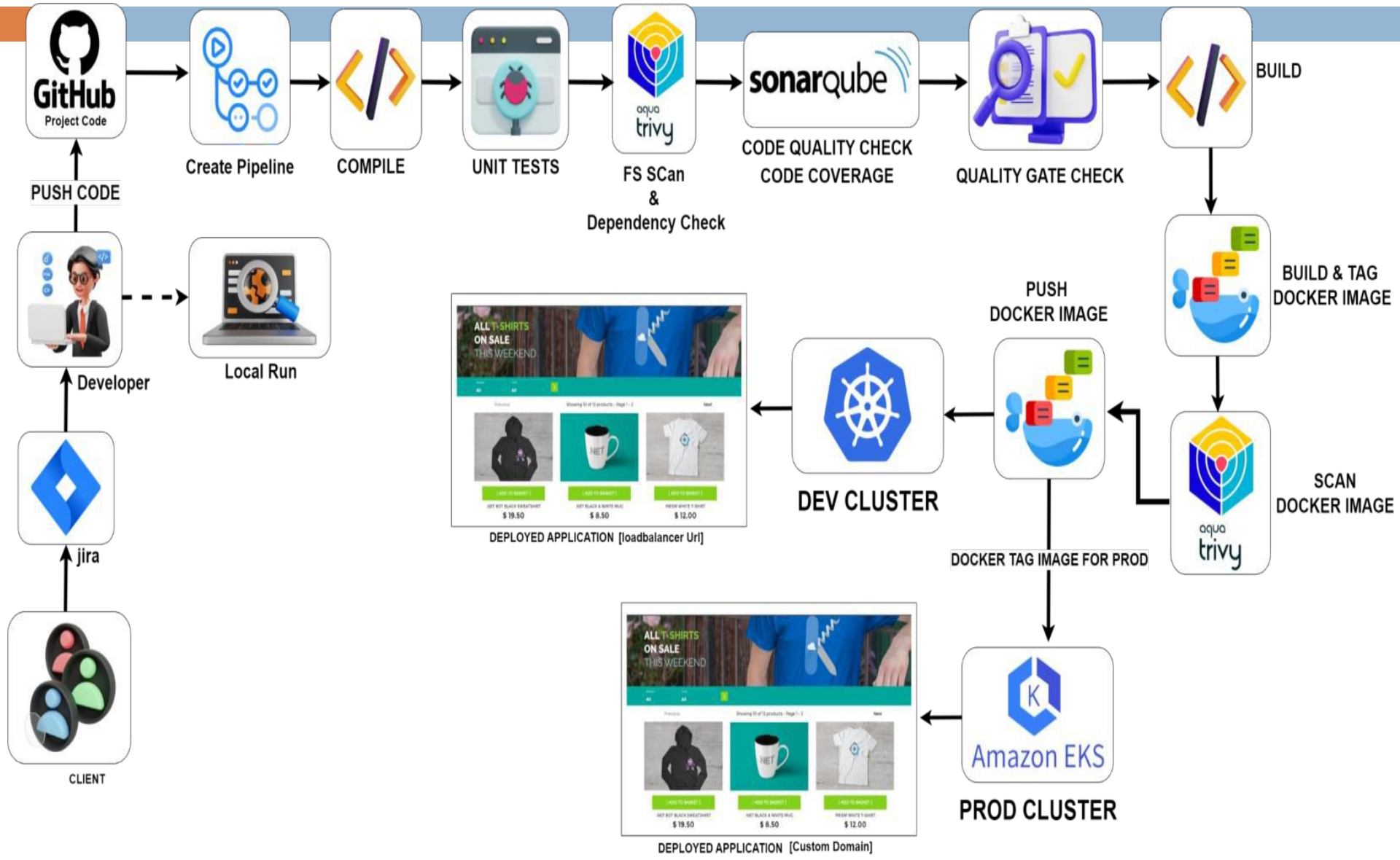
SaaS

Codeship | Travis CI | TeamCity | CircleCI | GitHub Actions | Semaphore

Cloud Services

Azure DevOps | AWS CodePipeline | Google Cloud Build

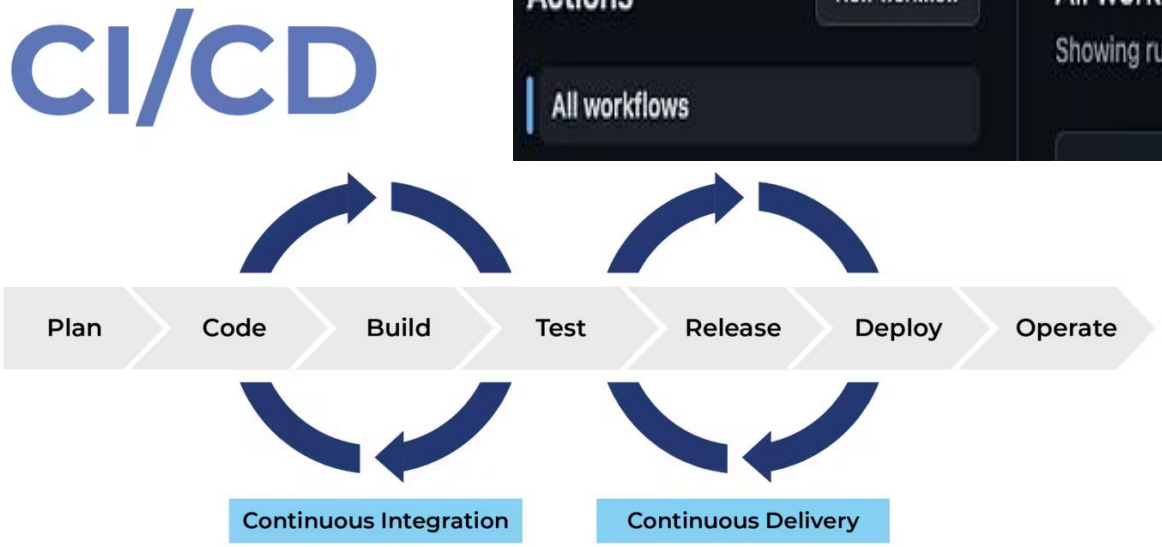
CI/CD DevOps Project

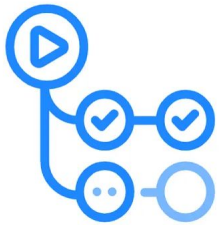


Github Actions

GitHub Actions Nedir?

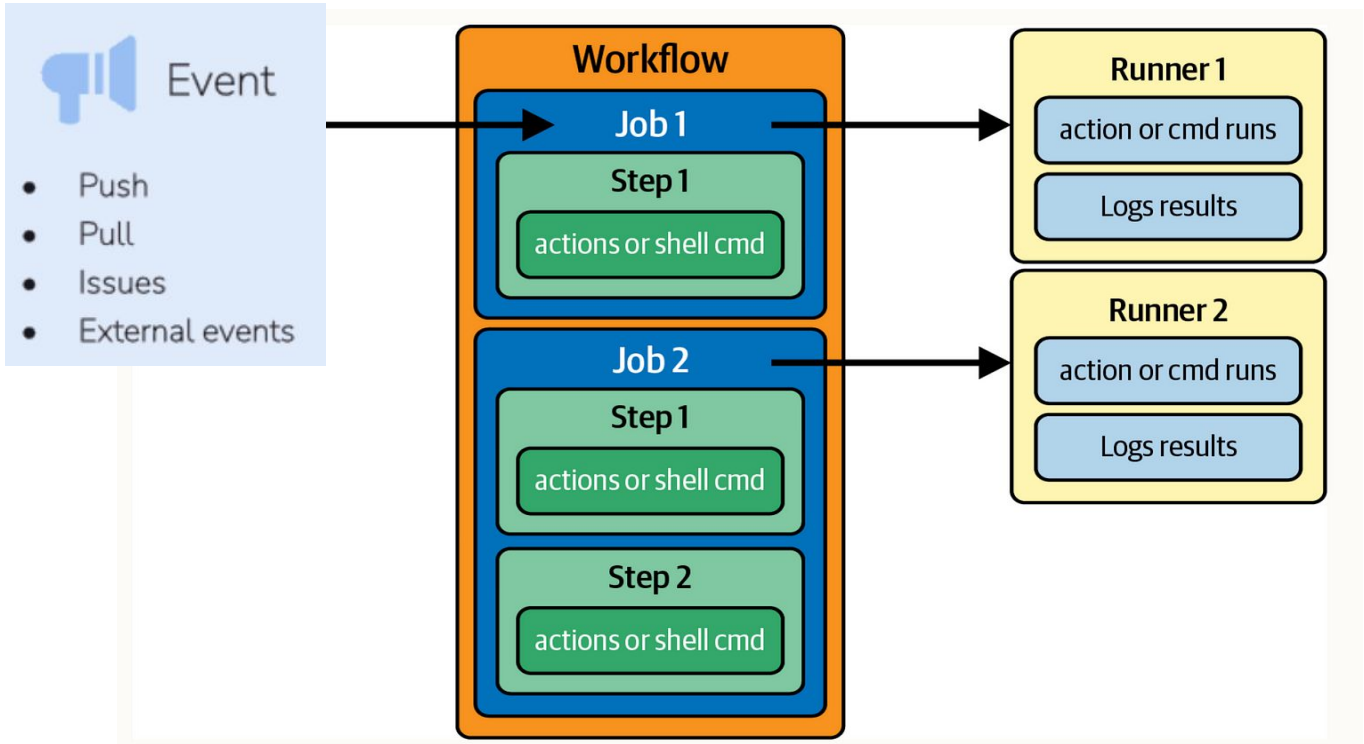
GitHub Actions, GitHub üzerinde çalışan CI/CD servisidir.Kodunuzda yapılan her değişikliği otomatik olarak test edebilir, build edebilir ve deploy edebilirsiniz. YAML tabanlı betiklerle çalışır, GitHub deposu içine **.github/workflows/** klasöründe tanımlanır.



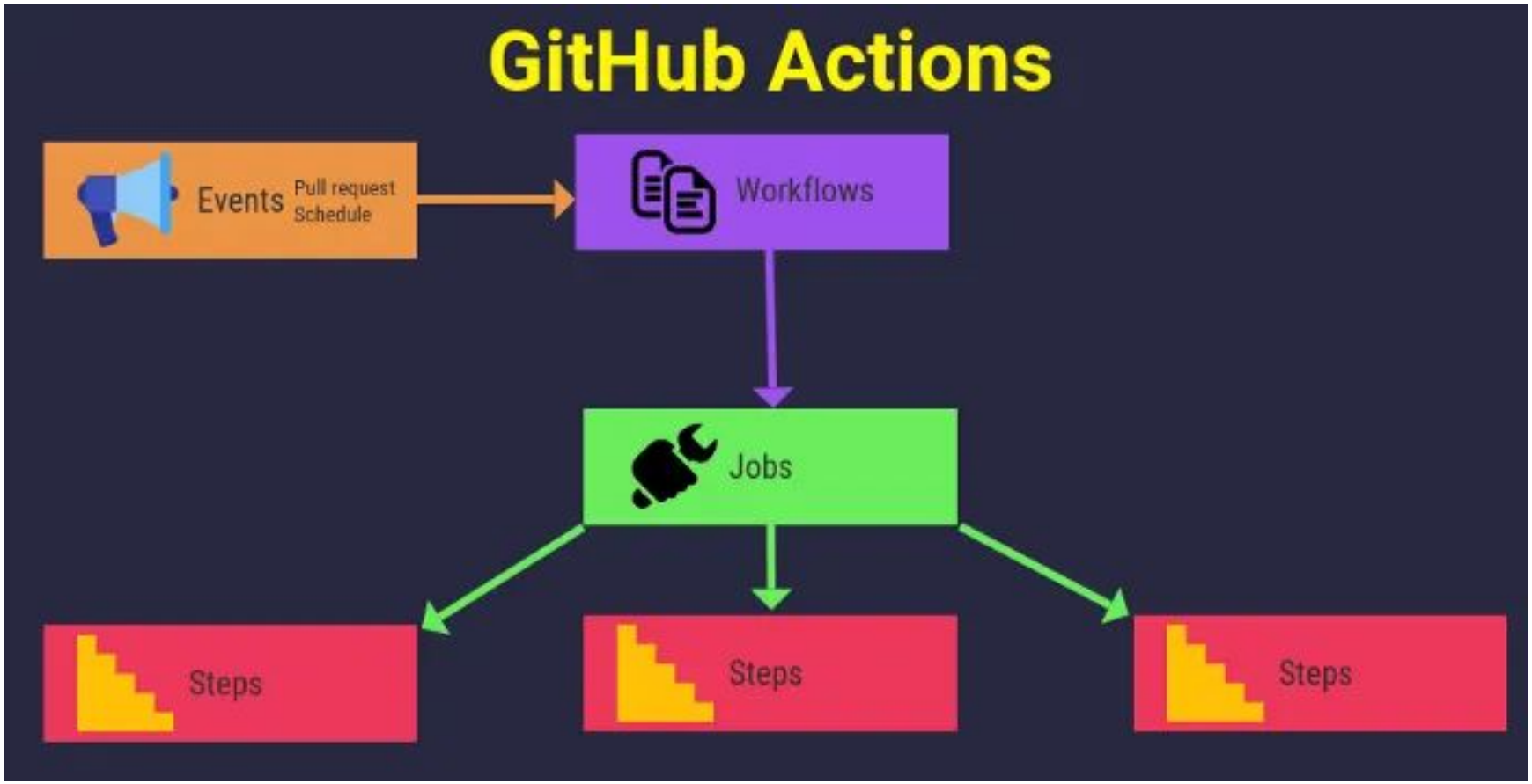


GitHub Actions Temel kavramlar

Workflow: Otomasyon süreci (bir veya birden fazla job içerir)
Job: Belirli bir ortamda (runner) çalışan adımlar grubu
Step: Komut ya da action'dan oluşan tek bir adım
Runner: Workflow'ların çalıştığı sanal makine
Action: Yeniden kullanılabilir script bloğu



GitHub Actions Nasıl Çalışıyor



GitHub Actions Workflow Yapısı

```yaml
name: Build and Test

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14'

      - name: Install dependencies
        run: npm install

      - name: Run tests
        run: npm test
```

Açıklama:

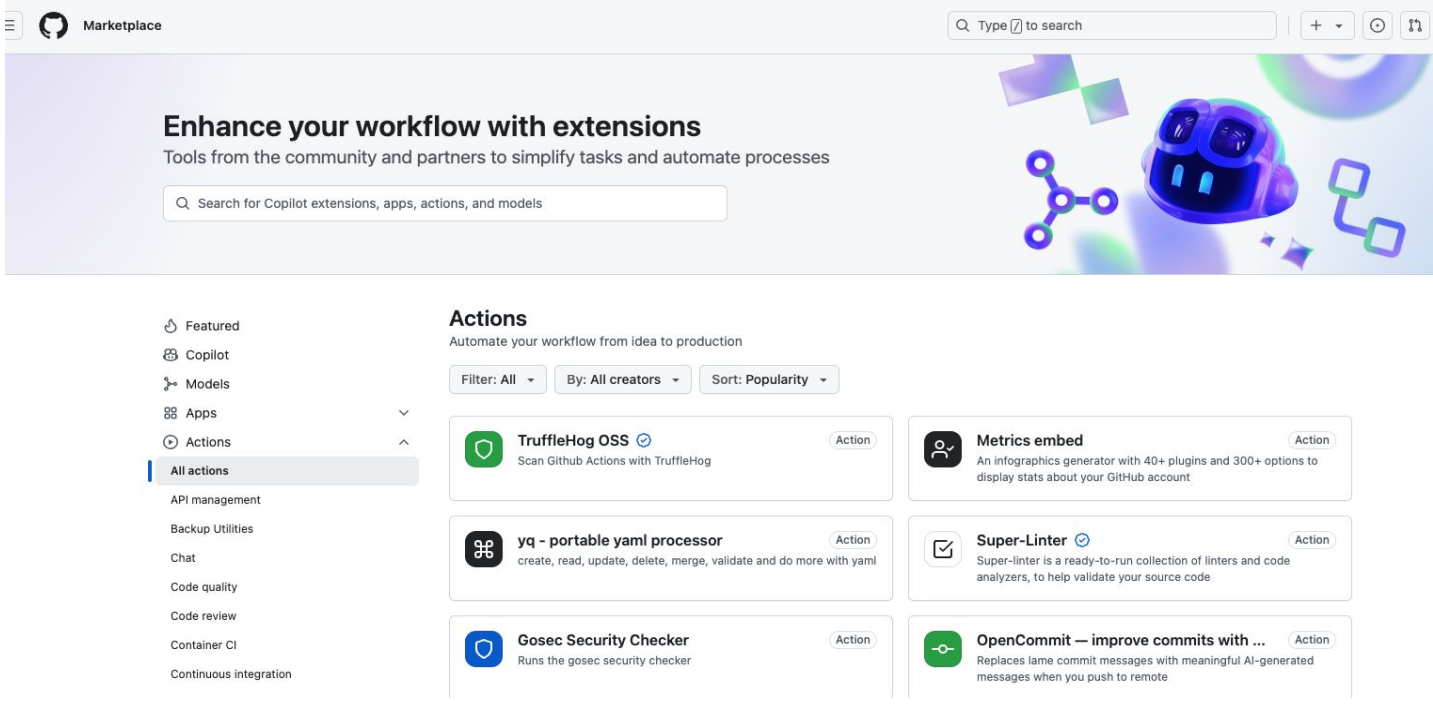
- `on`: İş akışını tetikleyen olay. Bu örnekte `main` dalına yapılan her push işlemi.
- `jobs`: Bir veya daha fazla iş tanımlar.
- `runs-on`: Hangi ortamda çalıştırılacağını belirler (`ubuntu-latest`).
- `steps`: Her adım, bir komut veya bir GitHub Action kullanır.

GitHub Actions Market Place

GitHub Action topluluğunda birçok hazır **aktions** bulunur.

https://github.com/marketplace?type=actions

- **`actions/checkout`** → repo klonlama
- **`actions/setup-node`** → Node.js kurulumu
- **`docker/build-push-action`** → Docker imajı oluşturma ve push etme



GitHub Actions Avantajları

- **Otomasyon:** Kod kalitesini artırır, süreçleri hızlandırır.
- **Entegrasyon:** GitHub ekosistemine tam uyumludur.
- **Esneklik:** Farklı platform ve ortamlarda çalışabilir.
- **Modülerlik:** Action'lar sayesinde tekrar kullanılabilir yapılar sunar.
- **Güvenlik:** Secrets ve role-based erişimle güvenli otomasyon sağlar.

DOCKER

Docker Nedir?

- Docker, uygulamaları **konteyner (container)** adı verilen izole ortamlar içinde çalıştıran açık kaynaklı bir platformdur.
- "Bir kez yaz, her yerde çalıştır" prensibine dayanır.
- Sanal makinelere göre çok daha hafif, hızlı ve taşınabilirdir.

- Docker as a "Company"
- Docker as a "Product"
- Docker as a "Platform"
- Docker as a "CLI Tool"
- Docker as a "Computer Program"

docker

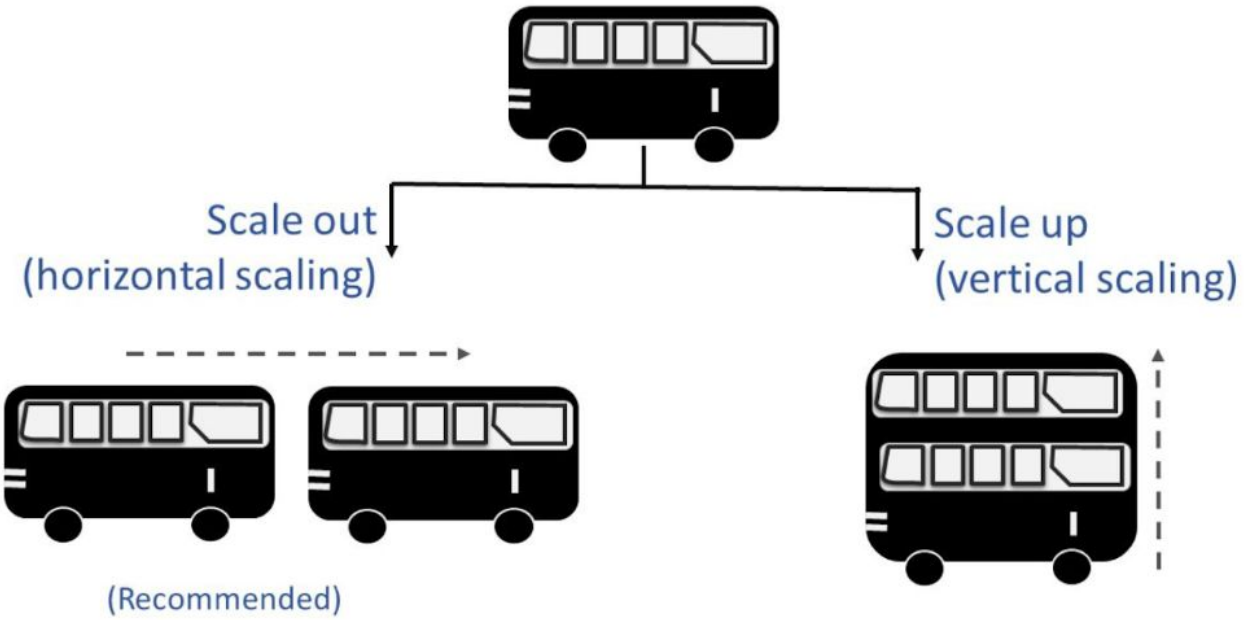
Why Docker?

- Increased Portability
  - Don't have to worry about environment

-

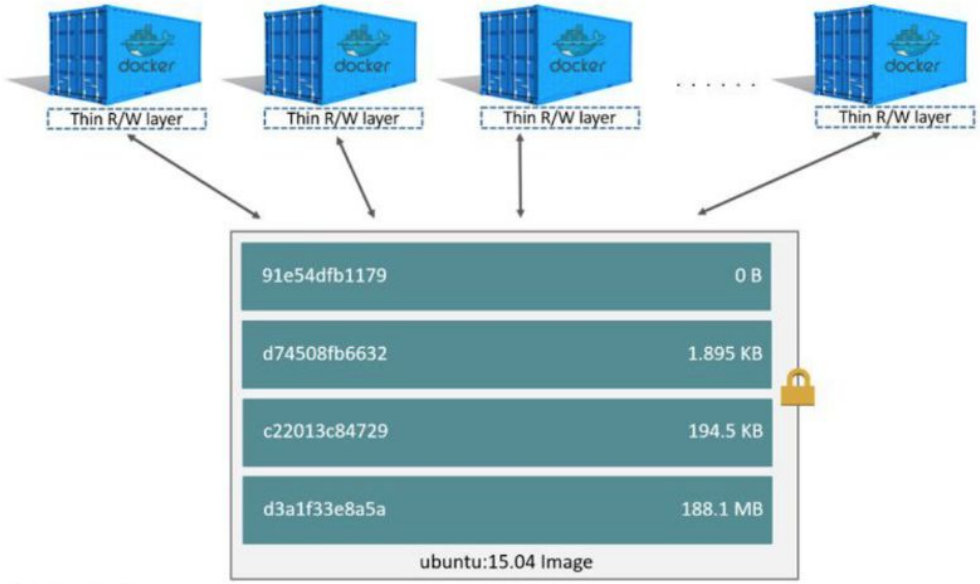
Why Docker?

- Improve Scalability
  - Vertical
  - Horizontal



Why Docker?

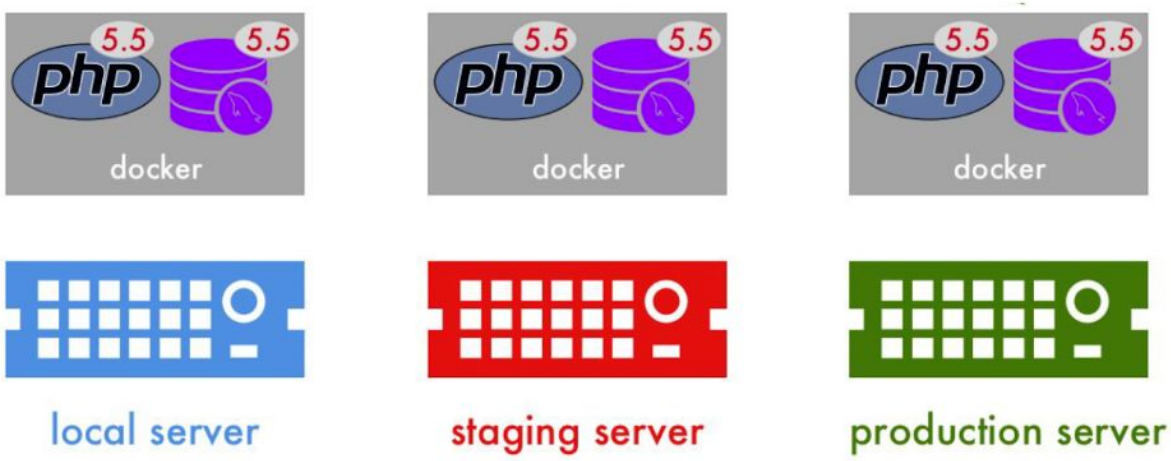
- Simple and fast deployment
  - quickly create new containerized instances or rapidly destroy multiple containers



docker

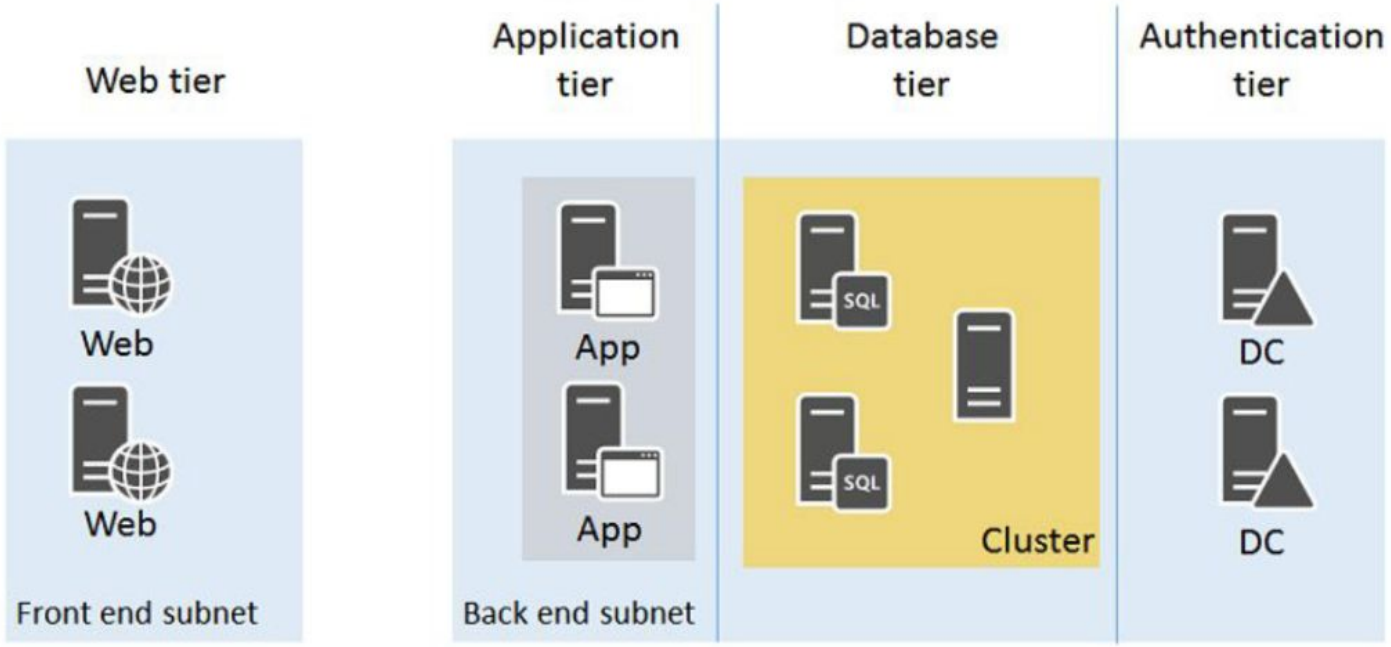
Why Docker?

- Enhance Productivity
  - promotes a rapid development environment
  - simplify the installation process and decrease dependency errors



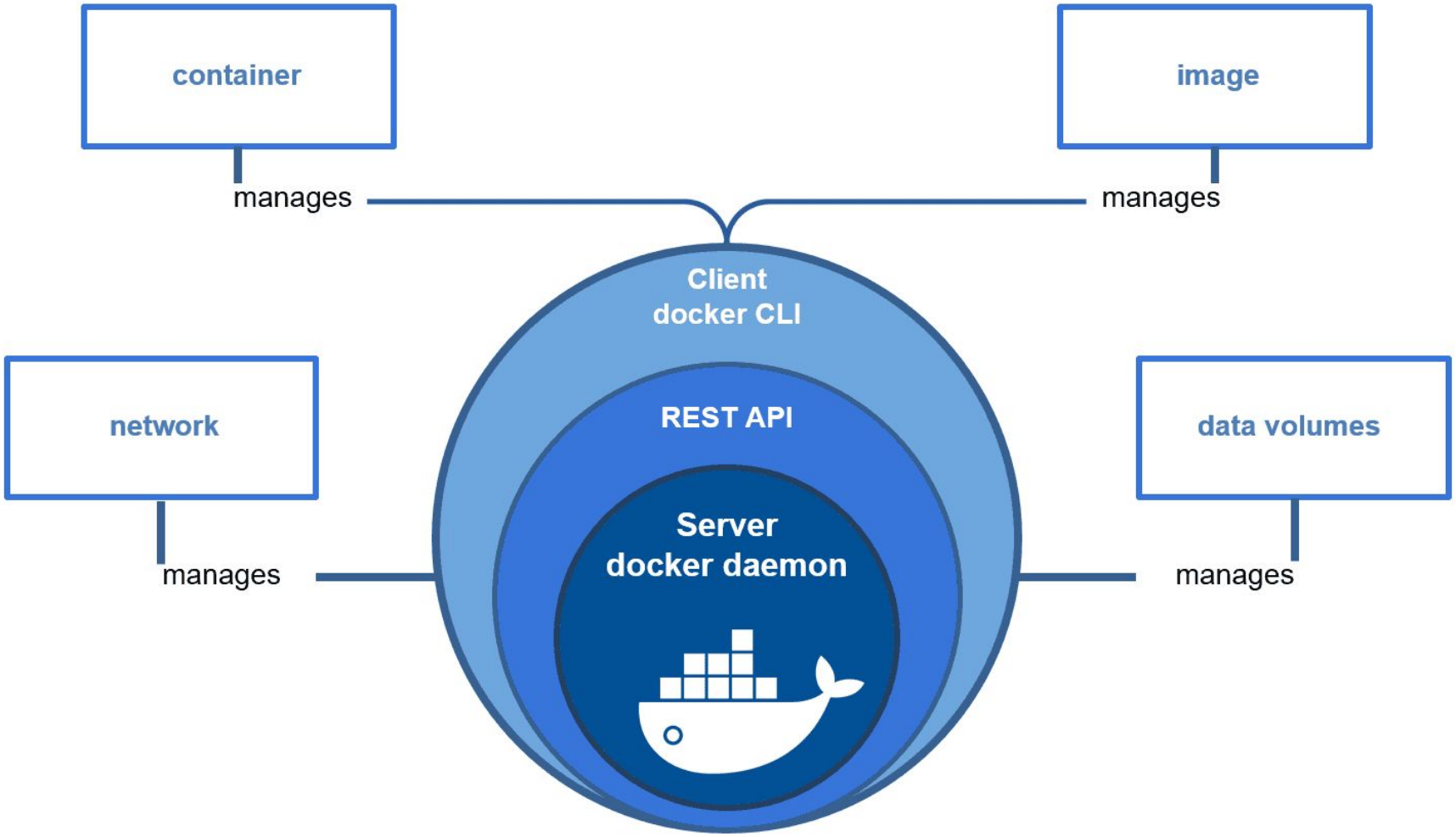
Why Docker?

- Improve Security
  - each application's major process apart from one another in separate containers

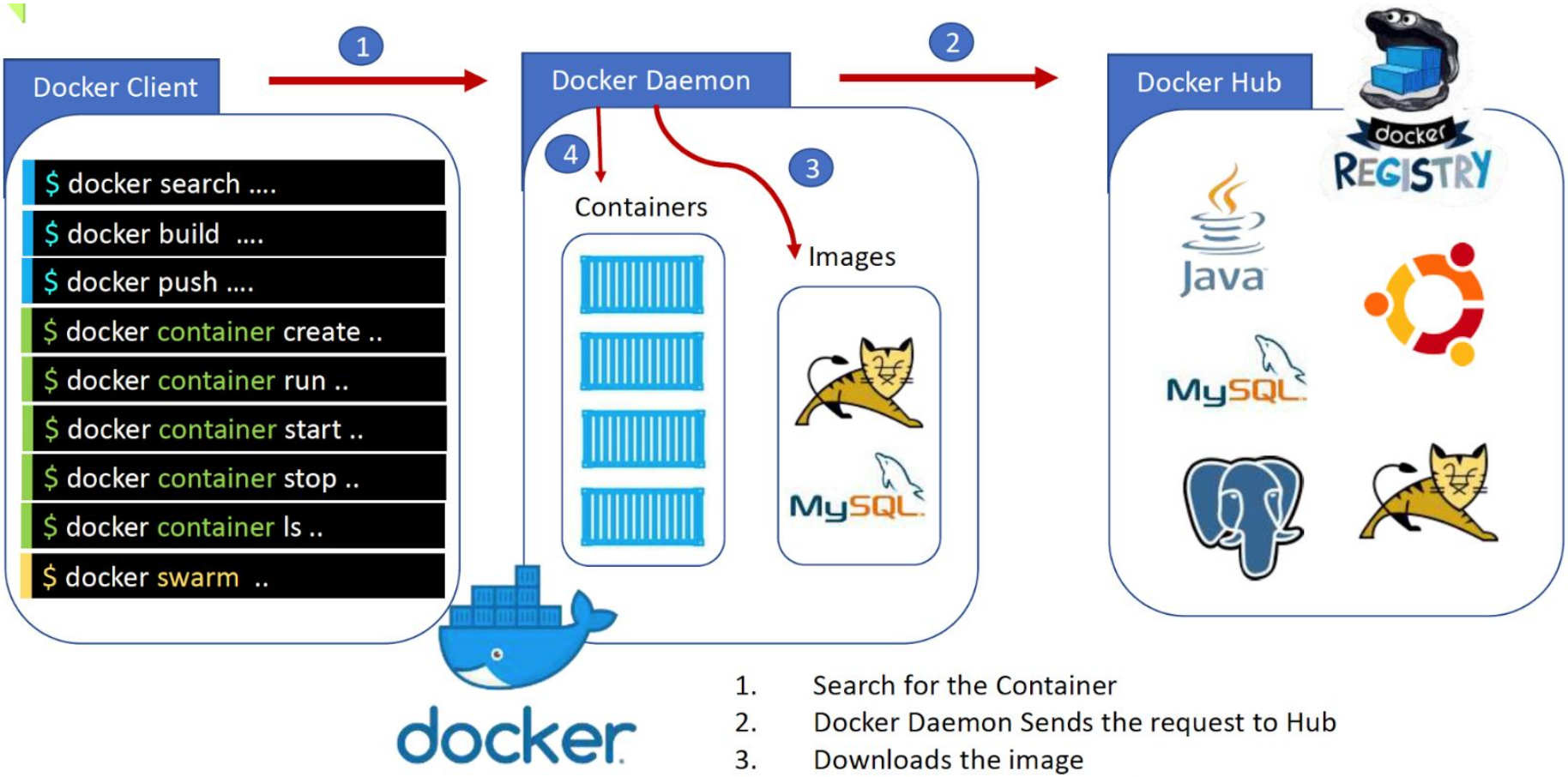


docker

Docker Engine Components

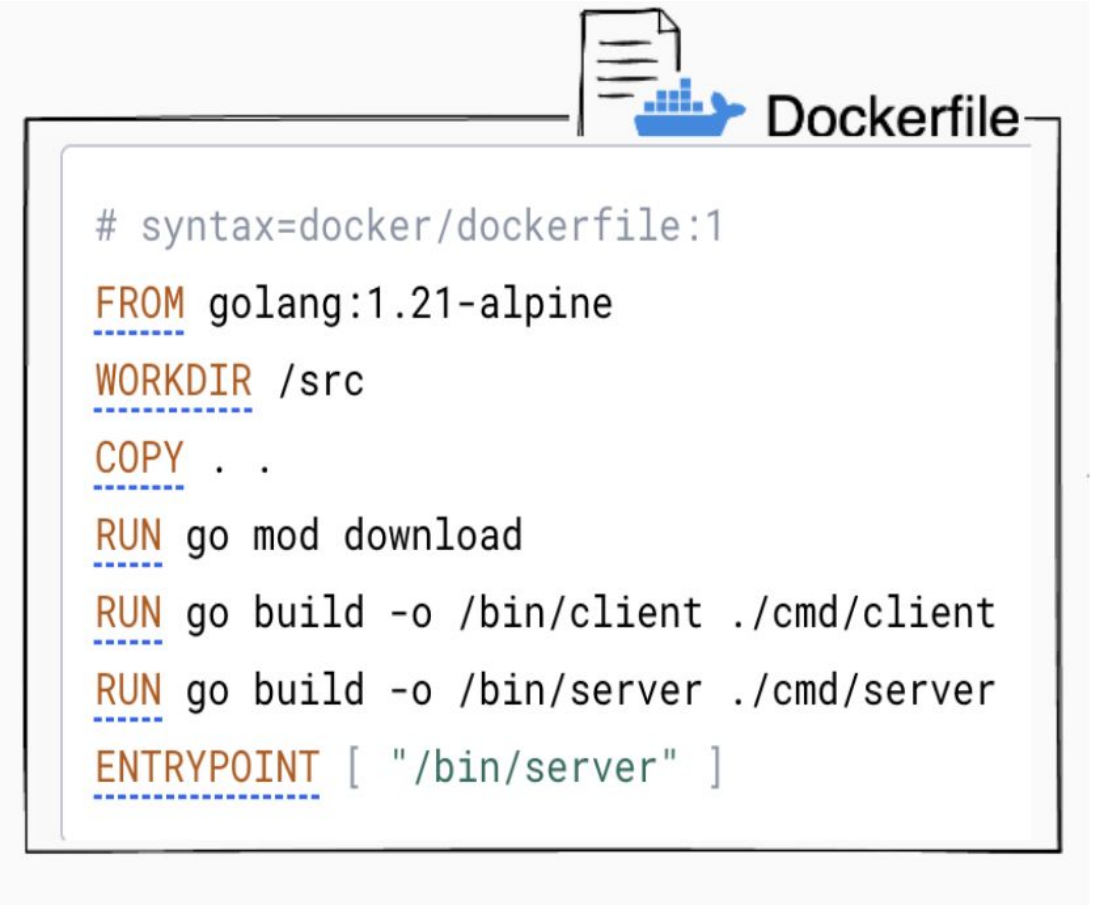


How Docker Works..



1. Search for the Container
2. Docker Daemon Sends the request to Hub
3. Downloads the image
4. Run the Container from the image

Dockerfile

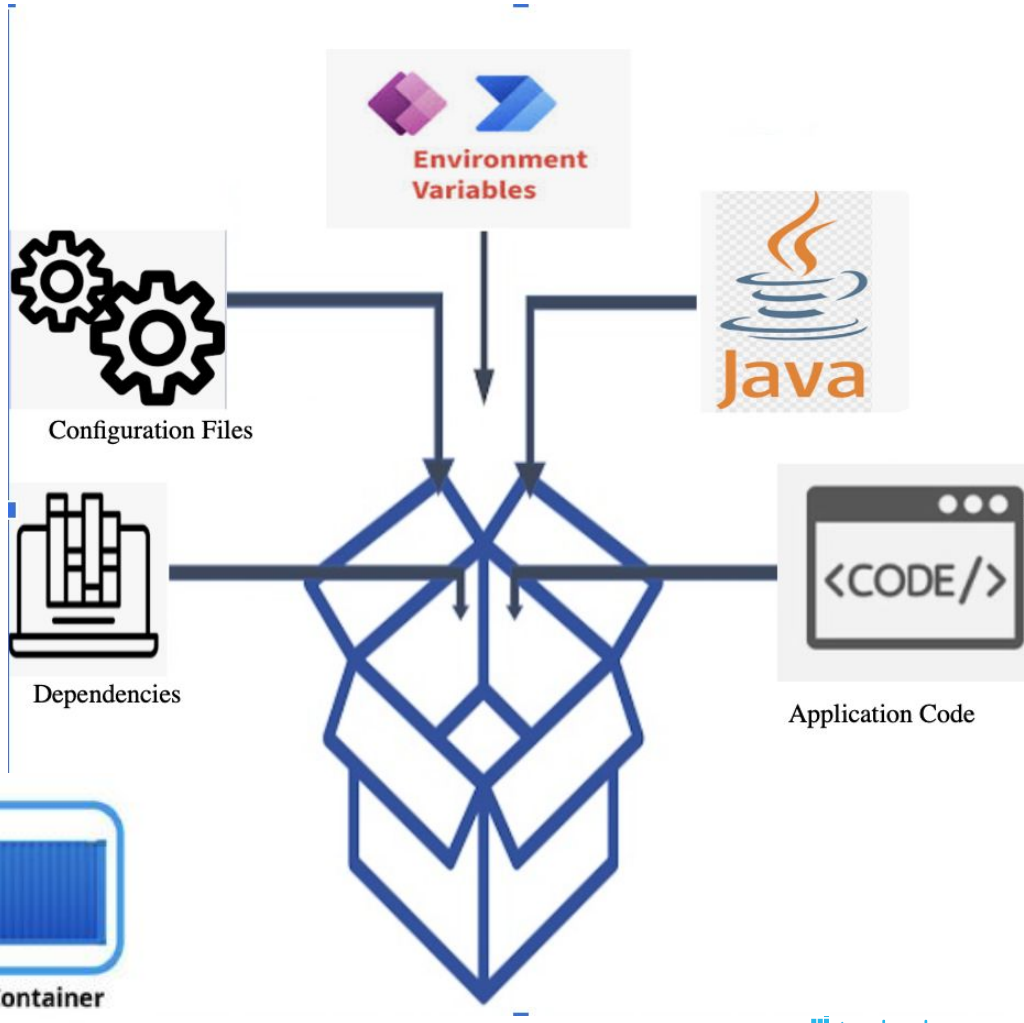


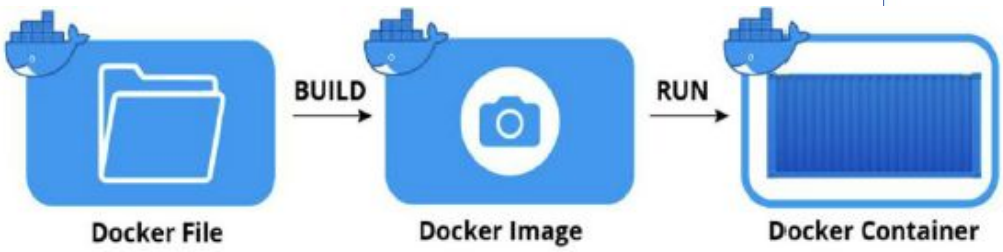
The application

```
.
├── Dockerfile
├── cmd
│   ├── client
│   │   ├── main.go
│   │   ├── request.go
│   │   └── ui.go
│   └── server
│       ├── main.go
│       └── translate.go
├── go.mod
└── go.sum
```

Docker Image Nedir?

Docker image: Bir uygulamanın çalışması için gereken her şeyi (kütüphaneler, bağımlılıklar, yapılandırma dosyaları vb.) tek bir paket içinde bir araya getirir.

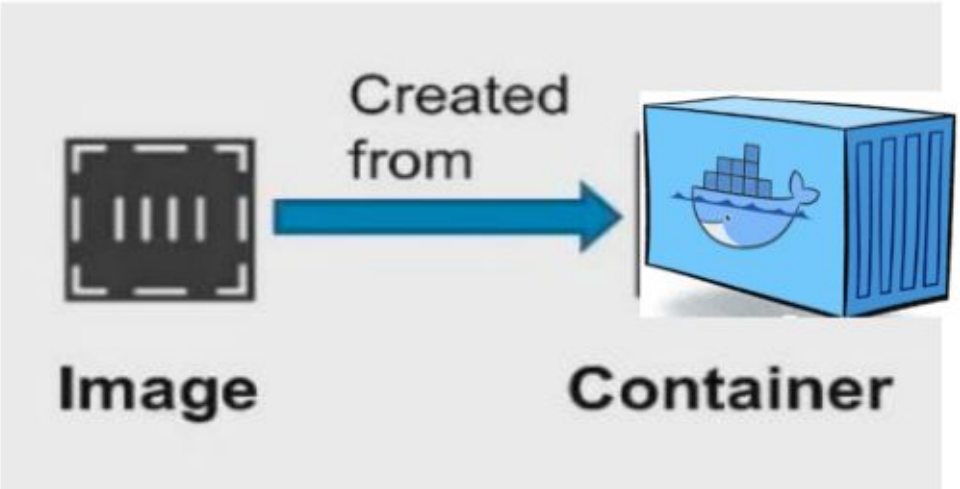


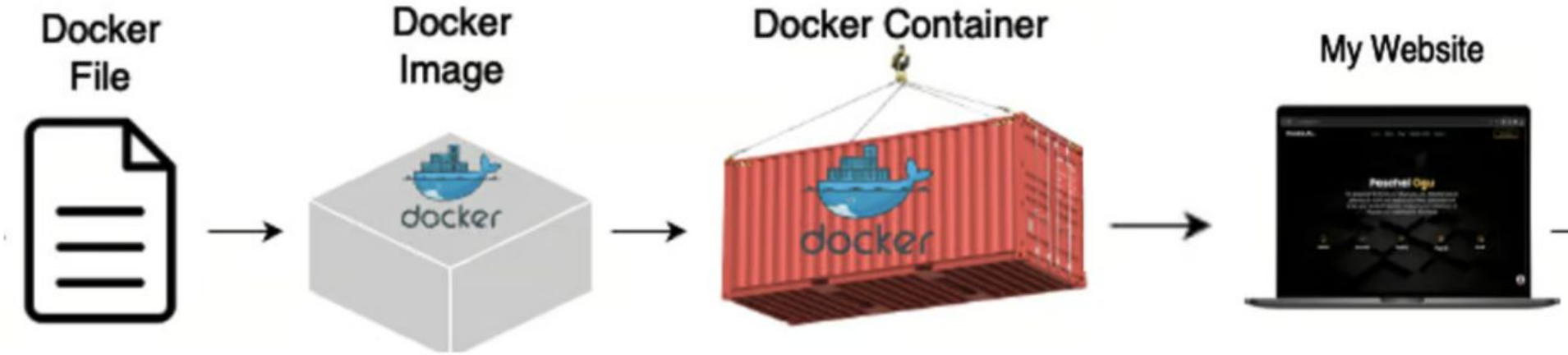


docker

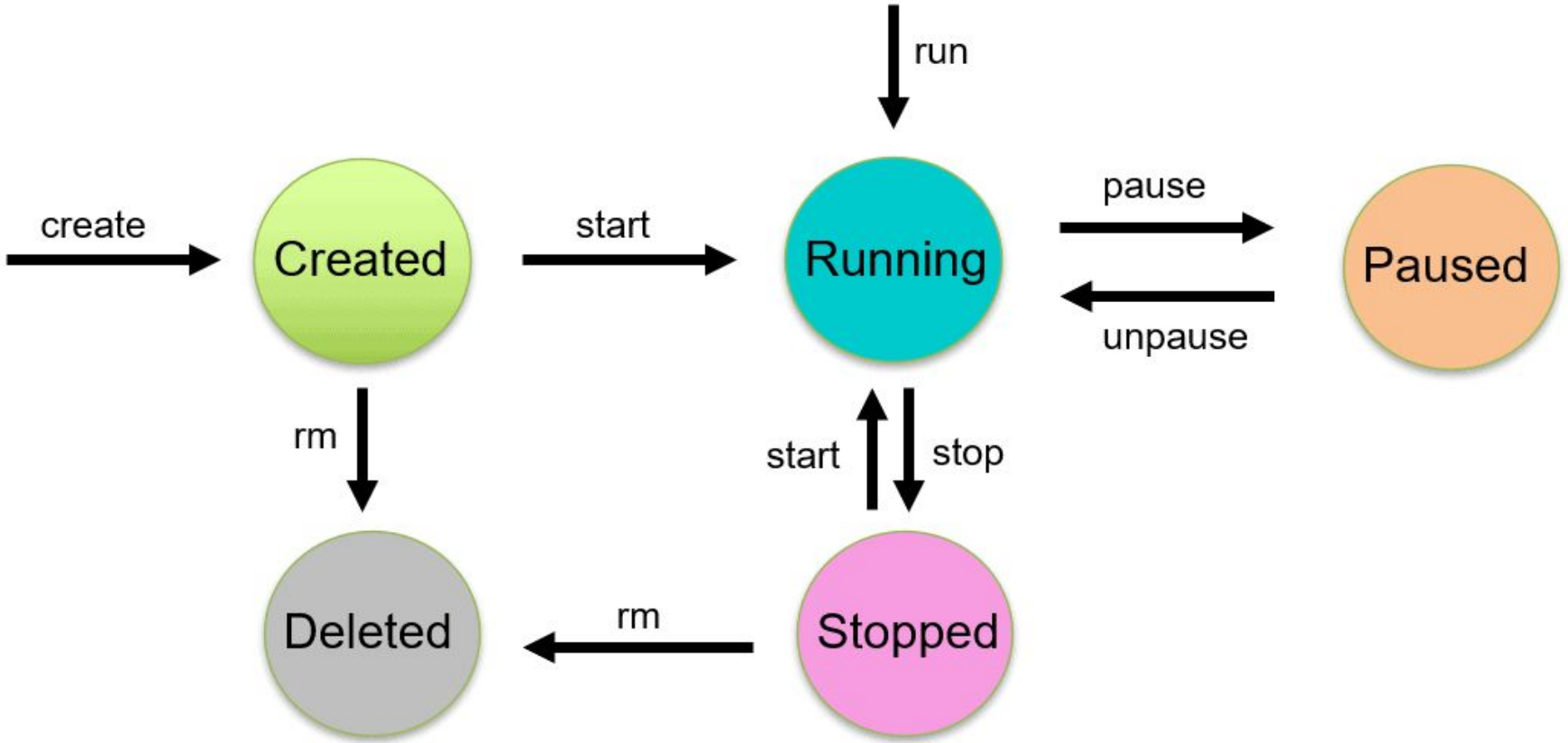
Docker Container Nedir?

Container: Docker image'ın çalıştırılmış halidir.





Docker Container Lifecycle



Temel Docker Komutları

Komut	Görev
`docker build -t app .`	Dockerfile'dan image oluşturur
`docker run app`	Image'tan container başlatır
`docker ps`	Çalışan container'ları listeler
`docker stop container-id`	Container'ı durdurur
`docker rm container-id`	Container'ı siler
`docker images`	Tüm image'leri listeler

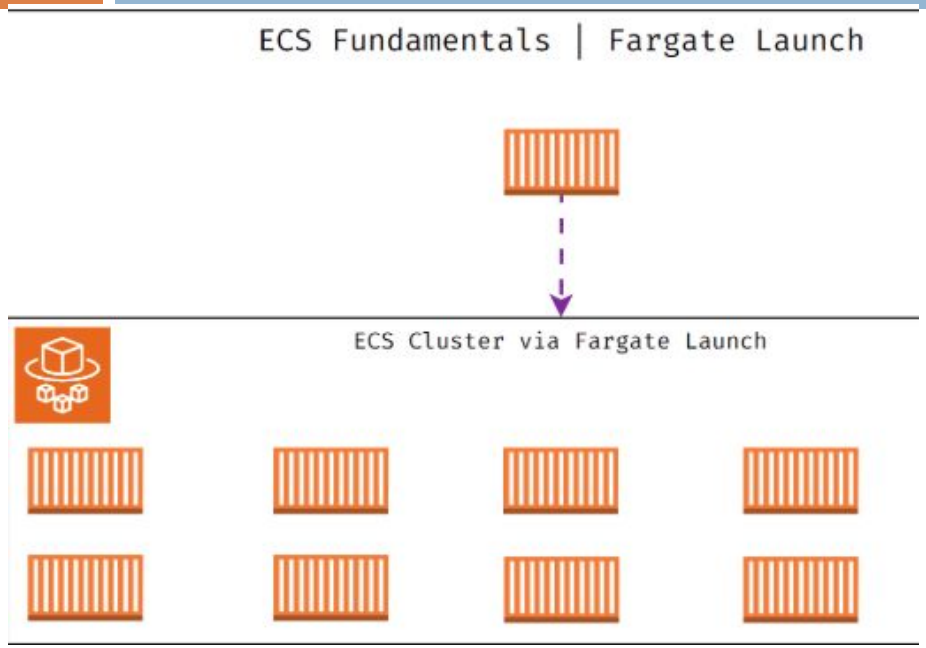
ECS

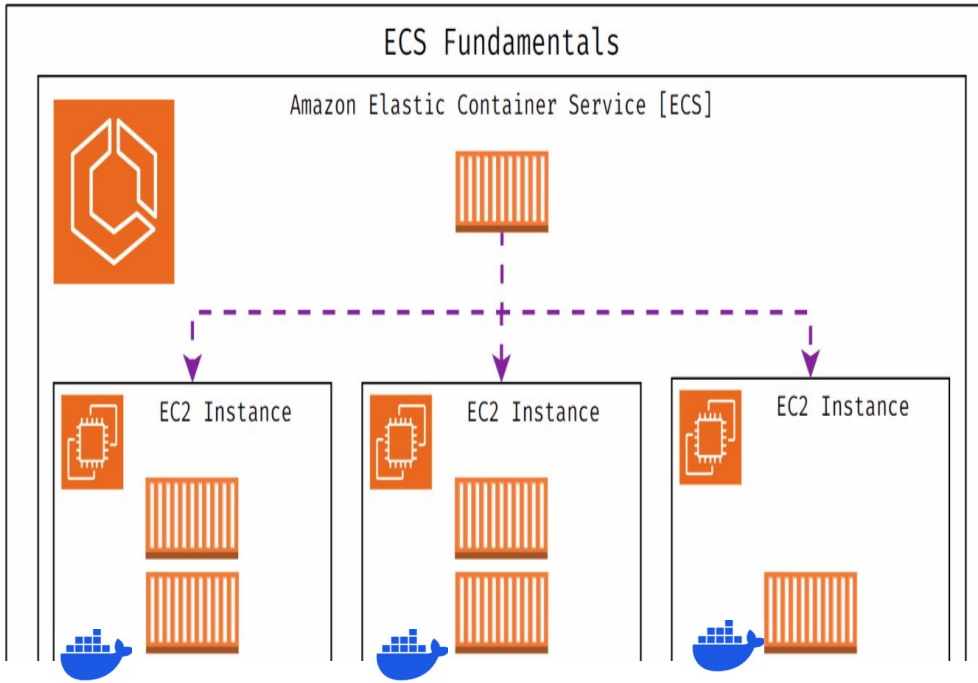
Amazon Elastic Container Service (ECS)

ECS (Elastic Container Service), AWS tarafından sağlanan, tam yönetilen bir **container orkestrasyon hizmetidir**.

- Docker container'ları kolayca çalıştırmak, yönetmek, ölçeklendirmek ve dağıtmak için kullanılır.
- EC2 veya Fargate ile çalışabilir (sunuculu veya sunucusuz)
- Otomatik ölçeklendirme, yük dengeleme ve izleme desteği sunar
- Kullanılan kadar öde modeli ile maliyet etkilidir

Amazon ECS'nin iki Dağıtım Seçeneği



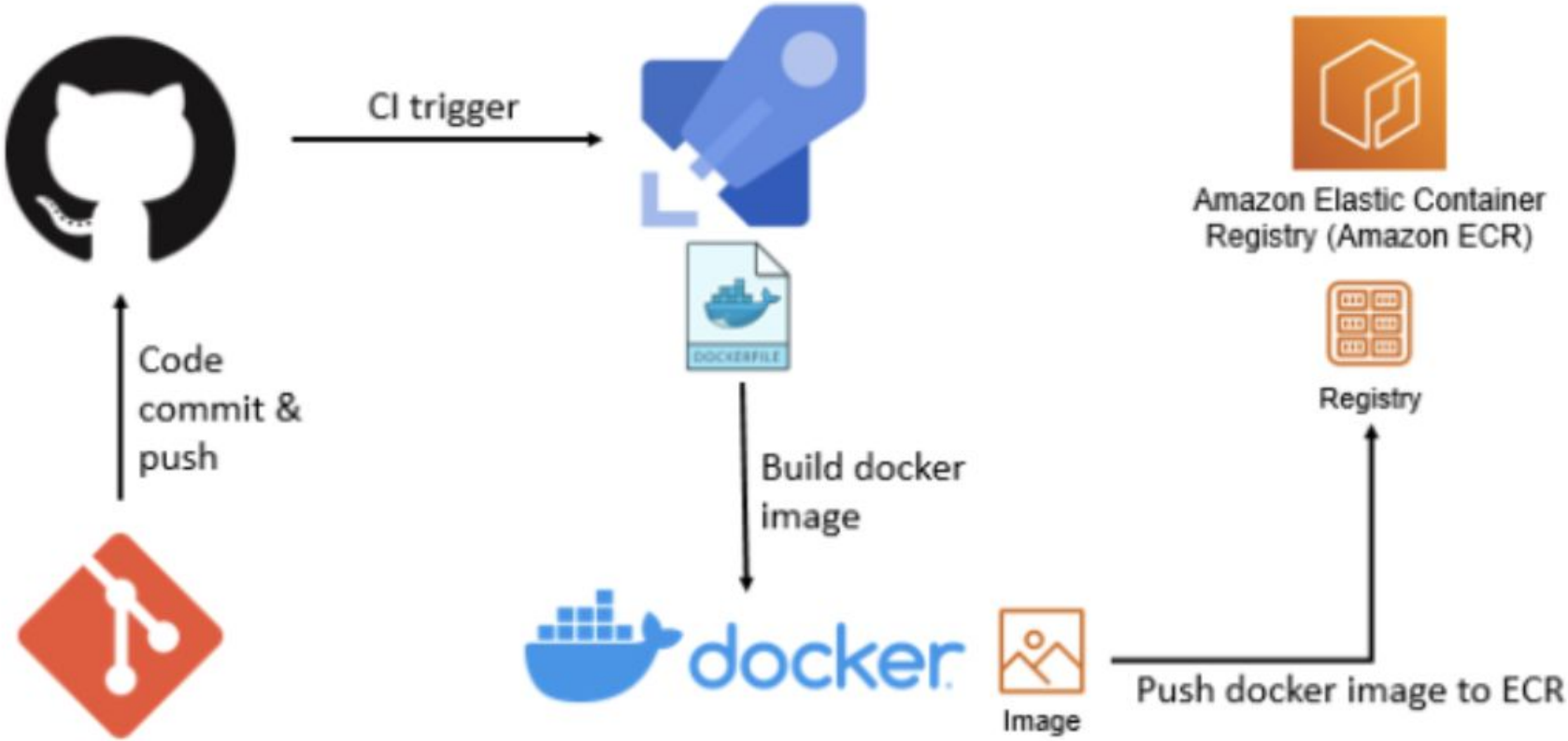


ECS Mimarisi (Temel Bileşenler)

Bileşen	Görevi
Cluster	EC2 veya Fargate görevlerini barındıran yapı
Task Definition	Container'ların nasıl çalışacağını tanımlar (Docker image, CPU, memory vs.)
Task	Bir veya birden fazla container'dan oluşan çalışan birim
Service	Task'ları yönetir, sayısını korur, yeniden başlatır
Launch Type	Fargate (sunucusuz) veya EC2 (sunuculu) seçimi

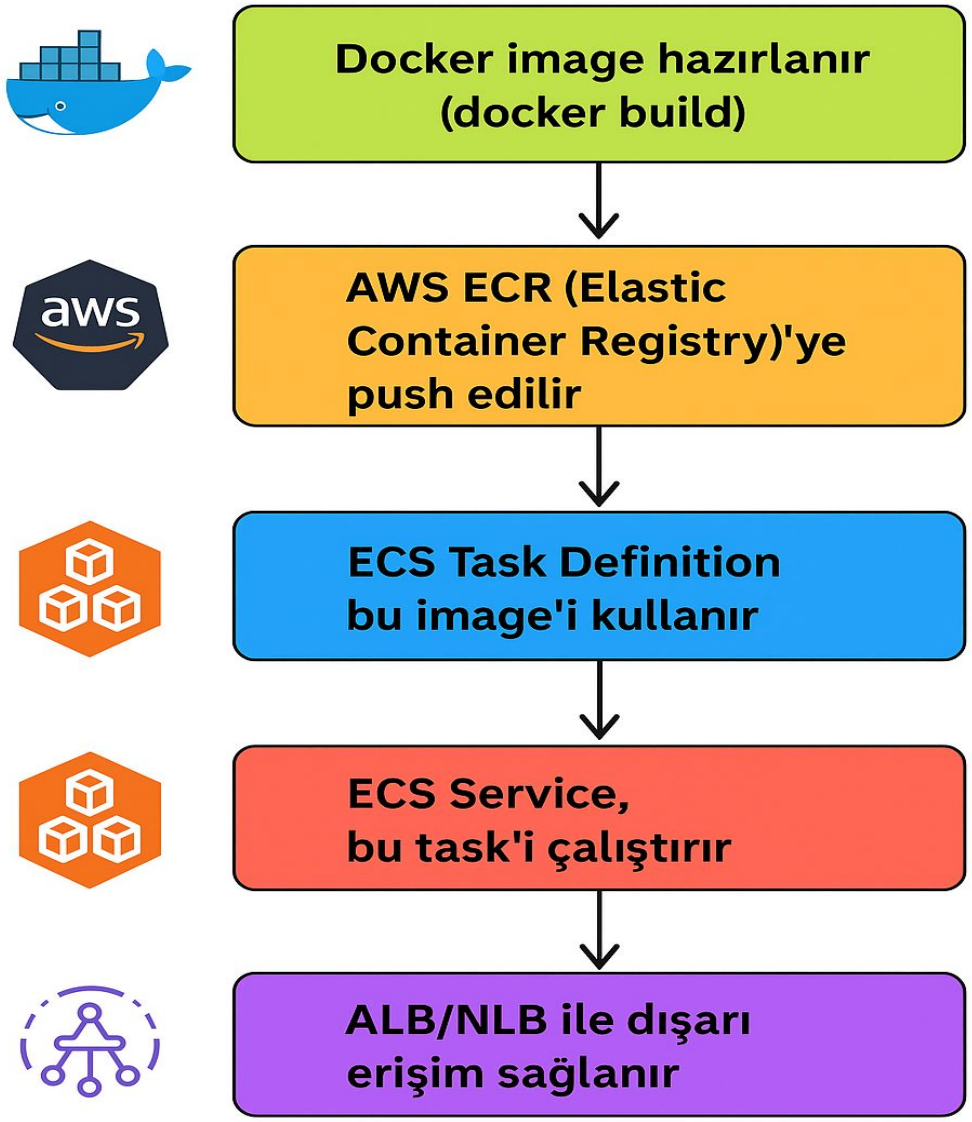
Amazon Elastic Container Registry (ECR)

AWS ECR Nedir?: Amazon Elastic Container Registry (**ECR**), **container imajlarını depolamak, yönetmek, paylaşmak ve dağıtmak** için kullanılan yönetilen bir AWS hizmetidir.

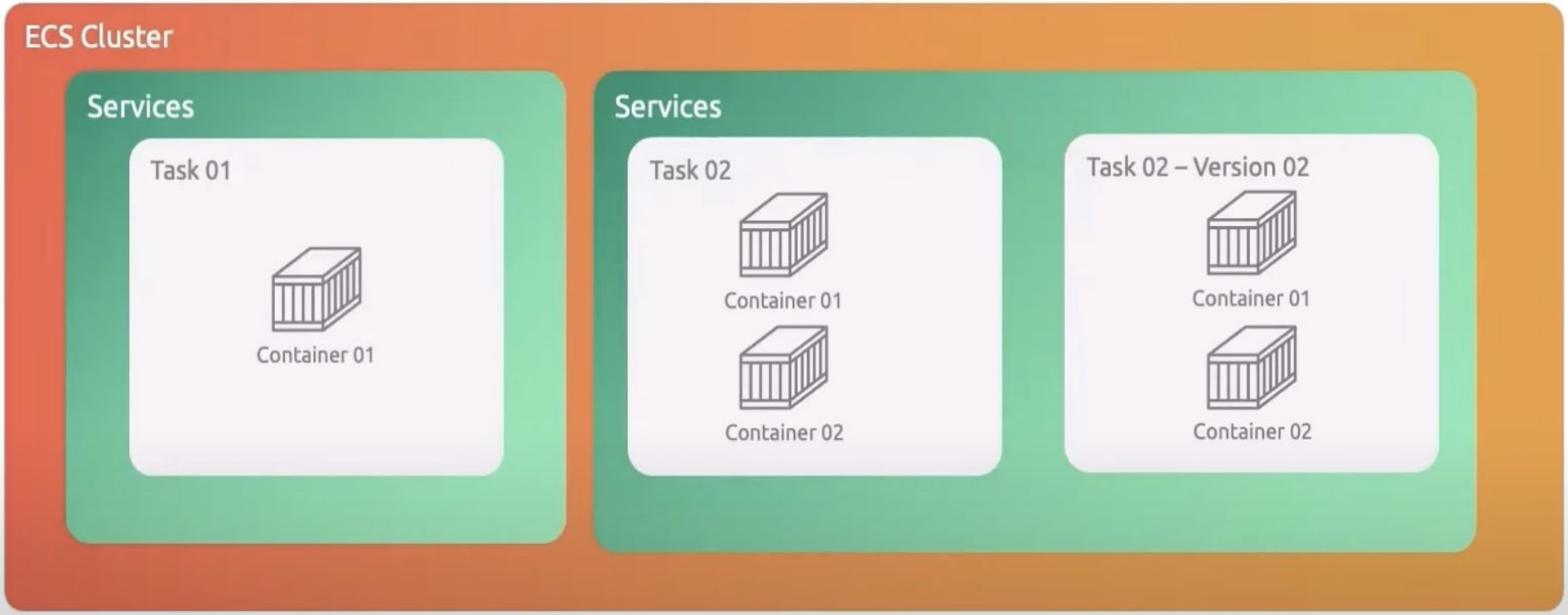


Skundunotes: Push Docker images to Amazon ECR using Azure Pipelines

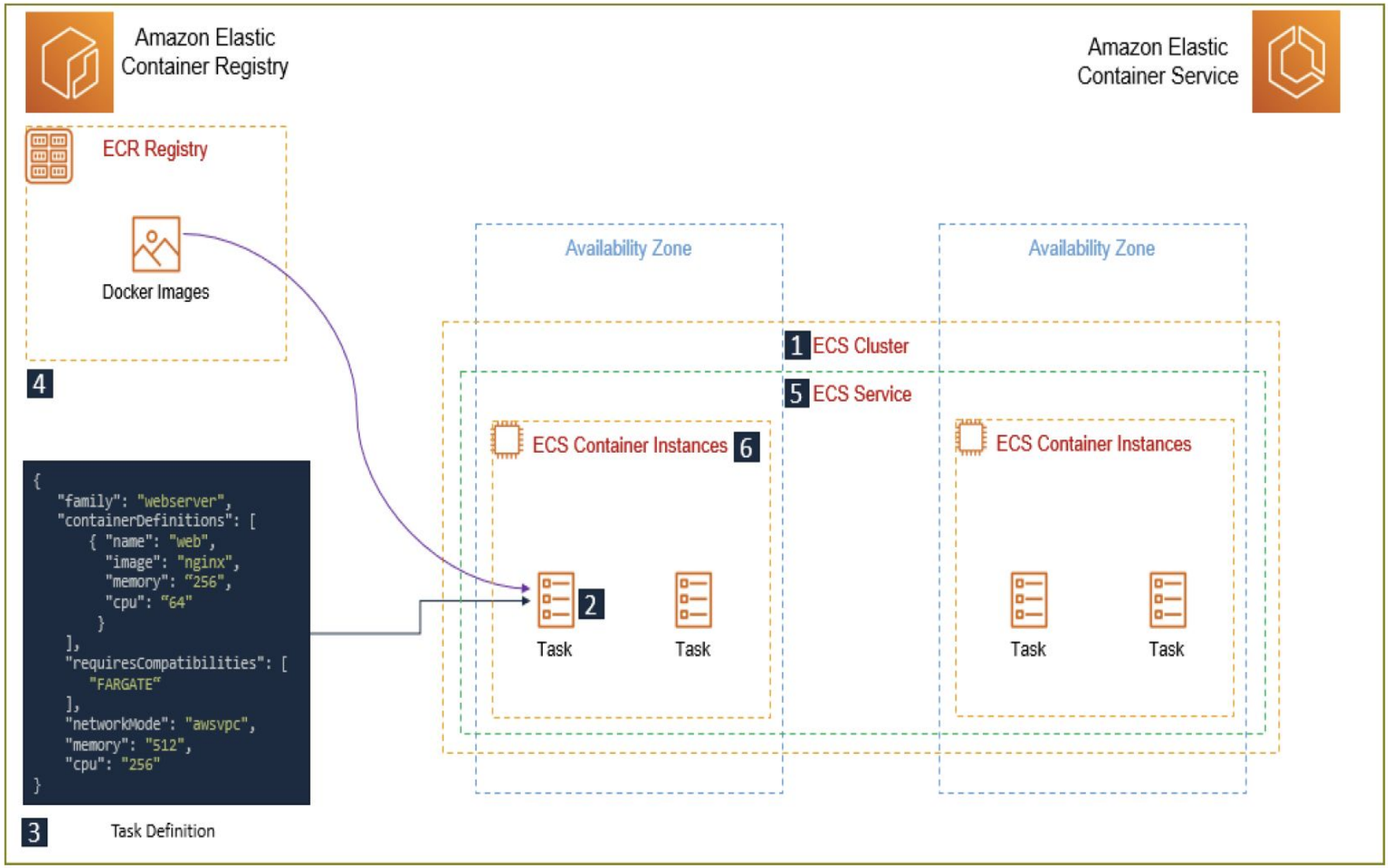
ECS ile Docker Kullanımı



Cluster Services Task



Amazon ECS Mimarisi



ECS'nin Sağladığı Özellikler

🔁 Otomatik yeniden başlatma, sağlıklı container takibi

🔄 Yük dengeleme (ALB/NLB) entegrasyonu

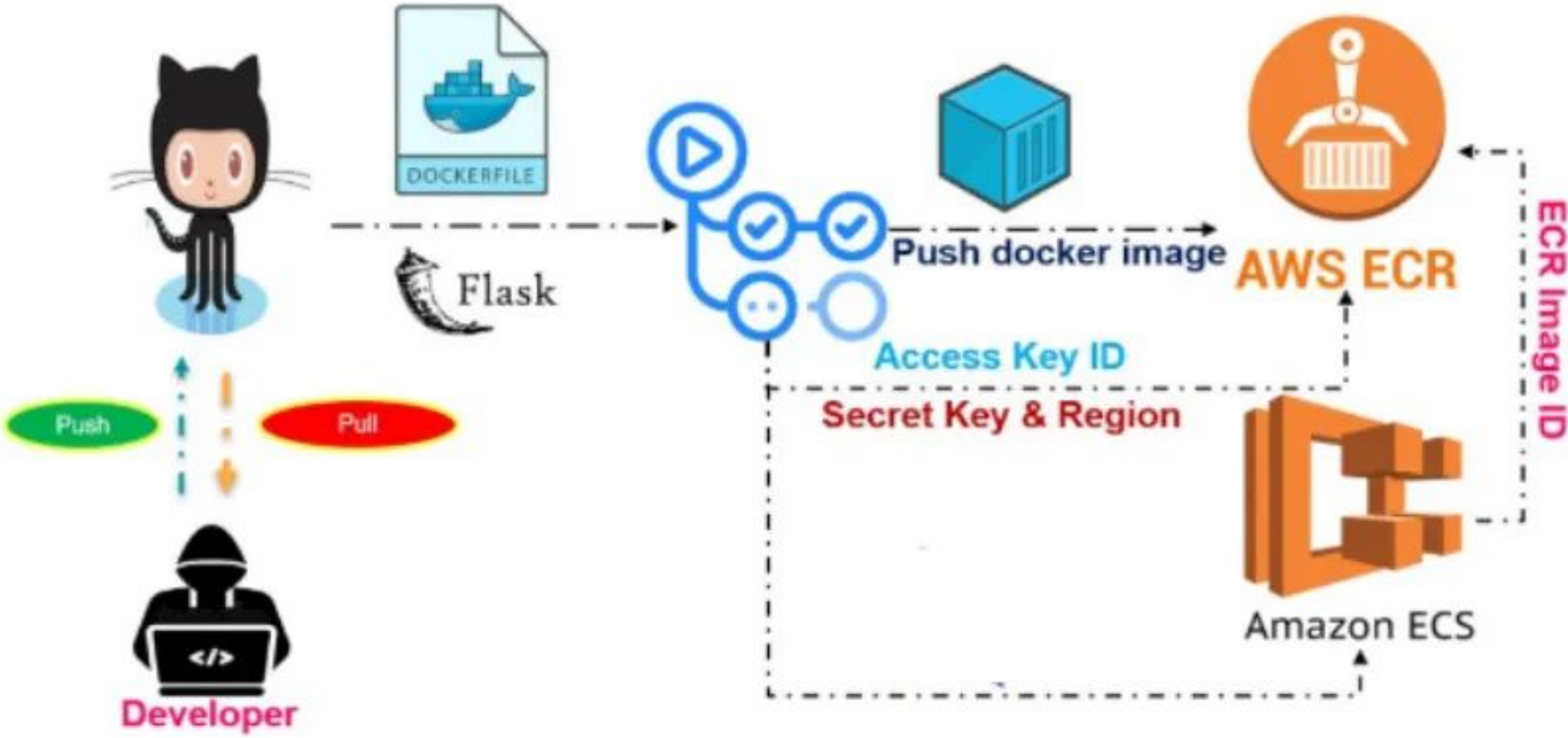
📈 CloudWatch ile detaylı log ve metrik izleme

🔐 IAM rolleriyle detaylı erişim kontrolü

☁️ VPC, Security Group, EFS gibi AWS kaynaklarıyla entegrasyon

GitHub Actions ile CI/CD’ye Giriş (Python Flask App - AWS ECS)

Proje Şeması



ProJenin Ana Adımları

1. **Proje Klasörünü Hazırlama**
2. **Flask Uygulamasını Yazma**
3. **Dockerfile oluşturma**
4. **Lokal Test**
5. **GitHub Repository Oluşturma ve Push**
6. **AWS Altyapısını Kurma**
   - **ECR** Repository oluşturma
   - **IAM** Yetkileri
   - **ECS** Cluster oluşturma
     1. Fargate cluster
     2. Task definition (container tanımı)
     3. Service
7. **Manuel Image Push**
8. **GitHub Actions Workflow**
9. **Canlı Test ve Doğrulama**