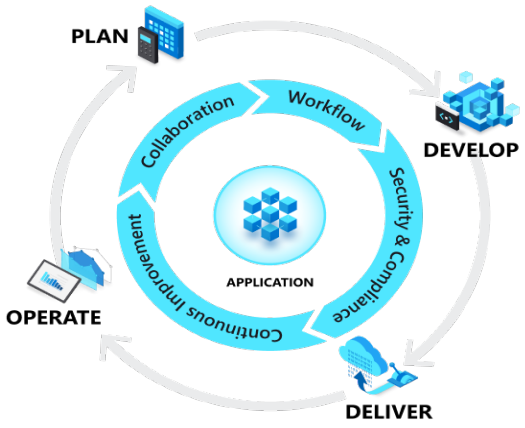
**DevOps Nedir**

DevOps kavramını tam olarak anlayabilmemiz için neden bu yaklaşım ortaya çıkmıştır onu anlamamız gerekebilir. Bu kavramın ortaya çıkma sebeplerinden biri olan şirketlerdeki iki farklı takım üzerinden konuya başlayabiliriz. Bunlar geliştirme ve operasyon takımlarıdır. Öncellikle geliştirici takımları kimlerden oluşur ve ne yaparlar onlara bakabiliriz. Geliştirme takımının içinde yazılım geliştiriciler ve testçiler gibi ekipler yer alır.
Geliştiriciler genel olarak yeni ürünler ortaya çıkarma ya da var olan ürünler üzerinden bu ürünlere güncelleme ve test yaparak sorunları çözmekle ve ortaya çıkan güvenlik açıklarını kapatmakla yükümlüdürler. Bunları yaparlarken karşılaştıkları bazı sorunlar vardır. Bu sorunlar arasında deployment süreçlerindeki problemler ve geliştirme ortamları ile canlı ortamların uyuşmaması gibi birtakım sorunlarla karşılaşabiliyorlar.

Diğer ekibimiz olan operasyon ekibinin içinde ise sistem yöneticileri, veritabanı ekipleri, güvenlik ve ağ ekipleri olabilir. Operasyon ekiplerinin başlıca görevi sistemlerin kesintisiz çalışmasını sağlayıp, erişilebilirliğin sürdürülmesinden sorumludurlar. Aynı zamanda geliştirme ekibinden gelen uygulamaların kullanacağı ortamları hazırlanması gibi istekleri gerçekleştirip ve bunların takibini yapmaktan da sorumludurlar. Operasyon ekiplerinin öncelikli amacı sistemlerin kesintisiz çalışmasını sağlamak olduğu için her zaman geliştirme ekibinden gelen istekleri tam zamanında karşılayamazlar.
Bu tarz sorunların çözümünü sağlamak, müşteri gereksinimlerini daha iyi karşılamak ve ekiplerin birbirleriyle daha uyumlu çalışmasına sağlamak amacıyla DevOps kavramı ortaya çıkmıştır. Bununla birlikte Devops sürekli entegrasyon (CI) ve sürekli dağıtım (CD) yaparak uygulama yaşam döngü yaklaşımını sağlar.




Böylece bu iki ekipte DevOps uygulama yaşam döngüsü yaklaşımları olan planlama, geliştirme, dağıtım ve yayınlama aşamalarına uyarlar. Aşamalar birbirleriyle bağlantılı olan bir döngü içerisindedir. Bu aşamalar hakkında kısaca bahsedersek;

Planlama aşamasında sistem gereksinimleri ve özellikleri tanımlanmaya çalışılır. Burada Scrum ve Kanban gibi uygulama geliştirme çerçevelerini kullanırlar.

Geliştirme aşamasında kod yazma, test etme ve ekip arkadaşları ile kodlar üzerinden uyumlu çalışabilme gibi yazılım geliştirme aşamaları gerçekleştirilir. Burada sürüm kontrol sistemleri için çoğunlukla Git kullanırlar.

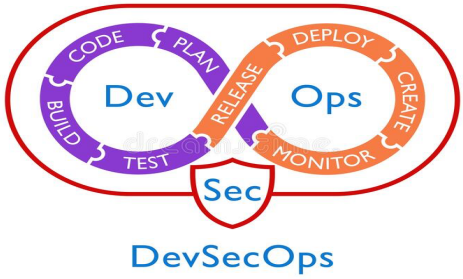
Dağıtım aşamasında uygulamanın üretim ortamlarına dağıtma sürecindeki aşamalarını otomatikleştirerek test etme sürecini gerçekleştirirler.

Yayınlama aşamasında ise uygulamaların bakımı, izlenmesi ve güvenli bir şekilde sorunları giderme süreçlerini kapsar.

Sonuç olarak bu aşamalarla birlikte DevOps sayesinde iş geliştirme süreçlerinde hız, güvenlik, performans artımı ve ölçeklendirme gibi avantajlara sahip olup, geliştirme ekibi ve operasyon ekibi arasında iş birliği sağlanarak yaşanan sorunlara bir çözüm getirebilir.

KAYNAK:
https://docs.microsoft.com/tr-tr/devops/what-is-devops
https://azure.microsoft.com/tr-tr/overview/what-is-devops/#devops-overview
https://aws.amazon.com/tr/devops/what-is-devops/

**DevSecOps**

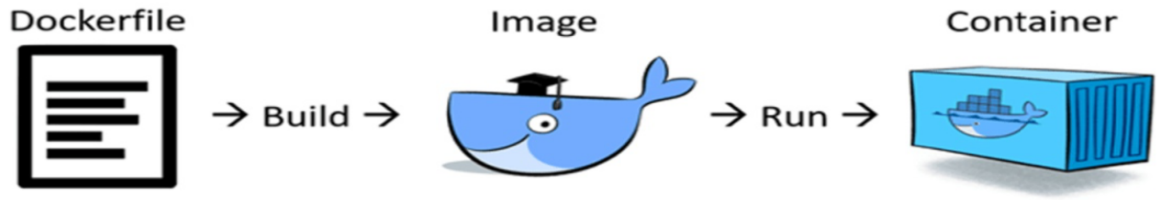



DevOps yaklaşımda sürekli entegrasyon ve sürekli dağıtımla otomasyon ve hız önemli bir yere sahiptir. Bu da beraberinde bazı güvenlik sorunlarını yanında getirebiliyordu. Projede yanlış bir yapılandırma yapılması ya da uygulamanın güvenlik açıkları olan konteyner içerisinde tutulması güvenlik açıklarına neden olabiliyor. DevOps uygulama yaşam döngüsünün her aşamasına güvenlik eklenmesi ile DevSecOps yaklaşımı ortaya çıkmıştır. DevSecOps geliştirme, güvenlik ve operasyonlar kelimelerinden oluşur. Ekipler arasında iş birliği sağlanıp, güvenlik sorunları erkenden fark edilip çok hızlı bir şekilde müdahale edilebilir. Bu sayede uygulama geliştirme süresi önemli ölçüde azalır ve daha güvenli kod yazımı sağlanır.

KAYNAK:
https://www.mshowto.org/devsecops-ve-bilesenleri.html
https://endpoint-labs.com/blog/2020/06/24/blog-post-5-eck/
https://www.ibm.com/tr-tr/cloud/learn/devsecops

**Dockerfile**

Dockerfile, image görüntüsü oluşturmak için oluşturan dosyadır. Dockerfile dosyasının bir uzantısı yoktur. Dockerfile içerisine hangi dosyaların kullanılacağı, uygulama parametleri ve kullanılacak image yazılır. Oluşturulan dosyanın ismi "Dockerfile" şeklinde yazılmalıdır. Docker, Dockerfile dosyasında tüm komutlar çalıştırılır, docker build komutu ile Docker Image oluşur ve run komutuyla oluşan konteyner çalıştırılır.

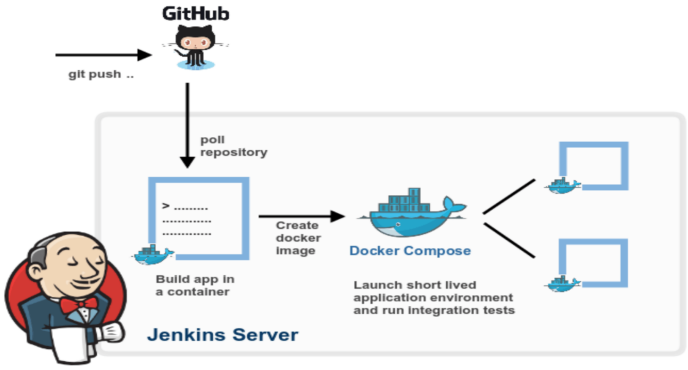



Dockerfile ile bir komutla projeyi defalarca aynı şekilde ayağa kaldırabiliriz. Böylece sunucu yapılandırması ile uğraşmak zorunda olmayız ve uygulamaya olan bağımlılıklarımızdan kurtulabiliriz.

Kaynak
https://medium.com/devopsturkiye/dockerfile-c0e5f5b751c0
https://keytorc.com/blog/docker-file-ve-image-nasil-olusturulur/

**Jenkins**

Jenkins, Java ile yazılmış açık kaynaklı DevOps otomasyon aracıdır. Sürekli entegrasyon (CI) ve sürekli Dağıtım (CD) amacıyla ortaya çıkmıştır.
Jenkins pipeline kurarak süreçleri otomatize eder. Her bir pipeline adımları yazılımcı tarafından kodlanır ve Jenkins bir ortam sağlar. Proje ekibi ile bir versiyon kontrol sistemi üzerinden geliştirme yapılırken, Jenkins sürekli proje yapılarını, değişikleri test eder ve hata durumda bildirim ile hataları bize gösterir. Bu adımlar build başarılı olana kadar devam eder build çalıştığında sunucuya dağıtılır. Bu sayede kodlardaki hatalar hemen fark edilip sorunlar çözülür ve yazılım geliştirme süreçlerinin hızlanmasına yardımcı olur.




Kaynak
https://keytorc.com/blog/jenkins-nedir-ve-nasil-kurulur/
https://kerteriz.net/jenkins-nedir-kurulumu-ve-ci-cd-surec-ornegi