**DevOps Nedir?**

DevOps kesin bir tanıma sahip olmasa da neyi kapsayıp, neyi kapsamadığı konusunda birçok fikir vardır.

**DevOps Tanımlarından Bazıları Şöyledir:**

* DevOps, IT bölümleri içindeki farklı disiplinlerin (developers & operations) sağlam bir güven ilişkisi ve etkili iletişim üzerinden beraber çalışmasıdır.
* DevOps, bir felsefe, bir kültürel yaklaşımdır. Yazılımcılar (dev) ve sistem yöneticileri (ops) arası etkili iletişimi teşvik eder.
* DevOps, yazılımcılar (dev) ve sistem yöneticilerinin (ops); tasarım, geliştirme, test ve yüklemeden oluşan yazılım yaşam döngüsünde ve daha sonrasındaki destek aşamasında da ortaklaşa çalışmaları için gerekli faaliyetlerin bütününü temsil eder.
* DevOps, yazılımcıların (dev) kendi sistemlerinin çalışması için kullandığı tekniklerin, sistem yöneticileri (ops) tarafından da kullanılmasıdır. Bu araçlar, kaynak kodu kontrolünden (source code control) teste, oradan [Agile](http://www.acm-software.com/agile/) geliştirme süreçlerine ([Agile](http://www.acm-software.com/agile/)development processes) kadar uzanır.

**Dev**

“Dev”, genelde yazılım geliştiricileri için kullanılır, ancak gerçekte daha geniş bir topluluğu temsil eder ve “yazılım geliştirme için çalışan herkes”i kapsar.

**Ops**

“Ops”, “Dev”e göre daha geniş bir kapsamı olup sistem mühendisleri, sistem yöneticileri, sürüm mühendisleri, veritabanı yöneticileri (DBAs), network mühendisleri, güvenlik uzmanları ve daha birçok değişik alt displinler için kullanılır.

## DevOps Nasıl, Hangi İhtiyaçtan Ortaya Çıktı?

Günümüzde yazılım ürünlerinin (uygulamaların) sürüm (version/release) değiştirme hızı, ürünler arası rekabetin artmasıyla birlikte artmıştır ve artmaya devam etmektedir. Şirketler ve ürünleri, bu yarışa benzer ortamda müşterilerini memnun etmek için kaliteli sonuçlar da elde etmek zorundadırlar. Bazı şirketler kullanıcı deneyimini iyileştirmek, hataları düzeltmek veya yeni özellikler eklemek için bazen gün içinde onlarca, belki yüzlerce yazılım değişikliği yapmakta ve kullanıcılara gerçek zamanlı sunmaktadır. İşte bu yüzden yazılım üretiminin, çeviklik (agility) ile yerine getirilmesi kaçınılmaz hale gelmiştir.

Eskiden “Dev” ve “Ops” birbirinden kopuk olarak çalışırken istenilen çeviklik elbette sağlanamıyordu, yazılım geliştiren şirketler kendi içlerinde verimsizlik yaşıyor ve nihayetinde müşteriler de durumdan memnun olmuyordu. Geciken projeler, hatalı ürünler, başarısız sürüm denemeleri, boşa harcanan para ve zaman, itibar kaybı… DevOps, işte bu gidişata dur diyecek bir hareket olarak ortaya çıkmış ve yazılım geliştiriciler (dev) ve sistem yöneticilerl (ops) arasındaki iletişimi, işbirliğini ve bütünleşmeyi (entegrasyonu) vurgulamayı esas edinmiştir. Sonuç olarak, şirketlerin IT hizmetleriyle beraber yazılım ürünlerinin de daha hızlı ve daha sık teslim edilebilir (continuous delivery) olmasına yardımcı olmaya başlamıştır.

## DevOps ve Agile

DevOps ile Agile yaklaşımın güçlü bağları vardır. [Agile](http://www.acm-software.com/agile/) yazılım geliştirme müşteri ile ürün yöneticileri, geliştiriciler, bazen de kalite kontrolcüler arasındaki işbirliğini zenginleştirmeyi hedefler ve daha yüksek kaliteli, daha sık ve sorunsuz ürün çıktısı sağlar. Müşteri için yaratılan değer de aynı şekilde yüksek olur. DevOps, yazılım geliştiricilerin çıktısı olan ürünün diğer sistemlerle etkileşiminin de müşteriye sunulan değerin temel bir parçası olduğunu söyler ve böylece ürün geliştiriciler, sistemler etkileşimini de hesaba katmalıdır der. Yani, DevOps basit anlatımla Agile prensipleri daha da genişleterek sadece üretilen kodu değil hizmetin tamamını göz önünde bulundurur diyebiliriz.

# **Agile**

Yazılım projeleri yaşam döngüleri boyunca kaynakların doğru kullanılmaması, yanlış önceliklendirmeler ve müşteri taleplerinin doğru anlaşılamaması nedeni ile başarısızlık tehlikesi ile karşı karşıya kalmaktadır. The Standish [how to give an informative speech](https://sigmaessays.com/informative-speech-topics/) Group’un 2009 yılı Chaos Report’una göre yazılım projeleri dünya genelinde %68 oranında başarısızlıkla sonuçlanmaktadır. Bu durum, Avrupa ve Amerika’daki büyük şirketleri, üretkenliği daha yüksek projeler üretmek üzere çeşitli yöntemleri denemeye yöneltmiş ve çoğu şirket yönetimde ve uygulamada en başarılı buldukları [**Agile**](http://www.acm-software.com/agile/) (çevik) yazılım geliştirme yaklaşımlarını benimsemişlerdir.

Agile yaklaşımlar uygulandığında artan;

* proje üretkenliği,
* projenin değişikliklere hızlı adapte olabilme yeteneği,
* proje kalitesi ve
* projenin pazara çıkış hızı

doğrultusunda projelerin **başarı oranları %80’lere çıkartılabilmiş** ve elde edilen önemli rekabet avantajı ile birlikte proje karlılıkları artırılmıştır.

Agile (çevik), dünya üzerinde kabul edilen yöntemler arasında [dissertation writing review](https://thedissertationhouse.com/) en hızlı ve güvenli proje geliştirme yaklaşımıdır. Şu anda dünyada irili ufaklı birçok şirket tarafından yoğun bir şekilde kullanılmaktadırlar.

**DevOps Yaşam Döngüsü**

Geleneksel yazılım geliştirme döngüsü (software development lifecycle) geçmişte genel olarak [Waterfall](https://en.wikipedia.org/wiki/Waterfall_model) yöntembilimini (metodoloji) izlerken günümüzde Agile SCRUM yöntembilimi yaygın hale gelmiştir. Ancak çoğu kurumda bu yaşam döngüsünü [Agile](http://www.acm-software.com/agile/), SCRUM ve Waterfall yöntembilimlerinin bir karışımı olarak görmek de mümkündür.

**DevOps Yaşam Döngüsü Aşağıdaki Gibidir:**

1. Kod, kaynak kontrol sistemine aktarılır (check in)
2. Kod, derlenmek için kaynak kontrol sisteminden çekilir (pull)
3. Testler çalıştırılır. Sürekli entegrasyon (continuous integration) sunucusu, yapıları (build) ve sürümleri (release) üretir. Entegrasyon ve kullancı testleri yapılır
4. Testleri geçen çıktılar (artifacts) ve yapılar (build) depolanarak saklanır
5. Otomatik bir sürümleme (release) aracı kullanılarak dağıtım yapılır
6. Ortam (environment) konfigüre edilir
7. Veritabanları güncellenir
8. Uygulamalar güncellenir
9. Test edilmiş uygulamalar kullanıcılara bir kesinti yaşamalarına gerek kalmadan aktarılır
10. Uygulama ve ağ performansı izlenir (monitoring) ve sorunlar oluşmadan engellenmeye çalışılır
11. Her adım biraz daha iyileştirilerek baştan alınır

**DevOps kullanılarak, bir geri dönüş (feedback) döngüsüyle ürünlerin sürekli olarak dağıtımı (deployment) şu aşamaların yürütülmesi ile gerçekleşir:**

* Altyapı Otomasyonu
* Konfigürasyon Yönetimi
* Dağıtım Otomasyonu
* Altyapı İzleme
* Log Yönetimi
* Uygulama ve Performans Yönetimi

## DevOps’un Faydaları

DevOps uygulamalarını yürüten şirketler çok daha verimli çalışırlar. DevOps yaklaşımı ile çalışmayanlara göre 30 kata kadar daha fazla dağıtım (deployment) yapabilirler. [Puppet Labs 2013 State of DevOps araştırma sonuçlarına](https://puppetlabs.com/wp-content/uploads/2013/03/2013-state-of-devops-report.pdf) göre dağıtımları %50 daha az başarısız olmuştur.

DevOps ile gelen en büyük değişim, eskiden ayrık çalışan ve farklı becerilere sahip takım üyelerinin; geliştiriciler, veritabanı yöneticileri, sistem yöneticileri, sistem analistleri vb.; bir araya gelip tek bir takım oluşturması olmuştur. Farklı rollerdeki bu işbirliğinin birçok avantajı vardır.

### Teknik Faydaları

* Sürekli teslim (continuous delivery)
* Düzeltilecek sorunların daha az karmaşık olması
* Sorunlara daha hızlı çözümler

### İşletmeye Olan Faydaları

* Ürün özelliklerinin daha hızlı teslimi
* Daha istikrarlı, stabil çalışan sistemler
* Sorun gidermekten daha çok, iyileştirme için zaman ayırma

## DevOps Araçları

### **Veritabanı**

* **MySql:** <https://www.mysql.com/>
* **MongoDB:** <https://www.mongodb.org/>
* **PostgreSQL:** <http://www.postgresql.org/>

### **Konfigürasyon Yönetimi (Configuration Management)**

* **Chef:** <https://www.chef.io/chef/>

### **Uygulama Dağıtımı (Application Deployment)**

* **Capistrano:** <http://capistranorb.com/>
* **Go:** <https://www.go.cd/>
* **Juju:** <https://jujucharms.com/>

### **İzleme (Monitoring)**

* **New Relic:** <http://newrelic.com/server-monitoring>
* **Sensu:** <https://sensuapp.org/>

### **Kaynak Kodu Versiyon Kontrolü (Version Control)**

* **git:** <http://git-scm.com/>
* **Apache Subversion:** <https://subversion.apache.org/>
* **Github:** <https://github.com/>
* **Bitbucket:** <https://bitbucket.org/>

### Yapılandırma ve Test (Build & Test)

* **Gradle:** <http://gradle.org/>
* **Jenkins:** <http://jenkins-ci.org/>
* **Maven:** <https://maven.apache.org/>
* **Grunt:** <http://gruntjs.com/>
* **sbt:** <http://www.scala-sbt.org/>

### Zengin Kaynak Yönetimi (Repository Management)

* **Nexus:** <http://www.sonatype.com/nexus/solution-overview>
* **Archiva:**<https://archiva.apache.org/index.cgi>
* **Nuget:** <https://www.nuget.org/>

### Sürekli Entegrasyon (Continuous Integration)

* **Jenkins:** <http://jenkins-ci.org/>
* **Bamboo:** <https://www.atlassian.com/software/bamboo/>
* **Travis:** <https://travis-ci.org/>
* **TeamCity:** <https://www.jetbrains.com/teamcity/>

### Middleware

* **Tomcat:**<http://tomcat.apache.org/>
* **Jetty:** <http://www.eclipse.org/jetty/>
* **Geronimo:** <http://geronimo.apache.org/>

### Bulut

* **Amazon Web Services:**<https://aws.amazon.com/>
* **Azure:** [https://azure.microsoft.com](https://azure.microsoft.com/)
* **Jelastic:**<https://jelastic.com/>

