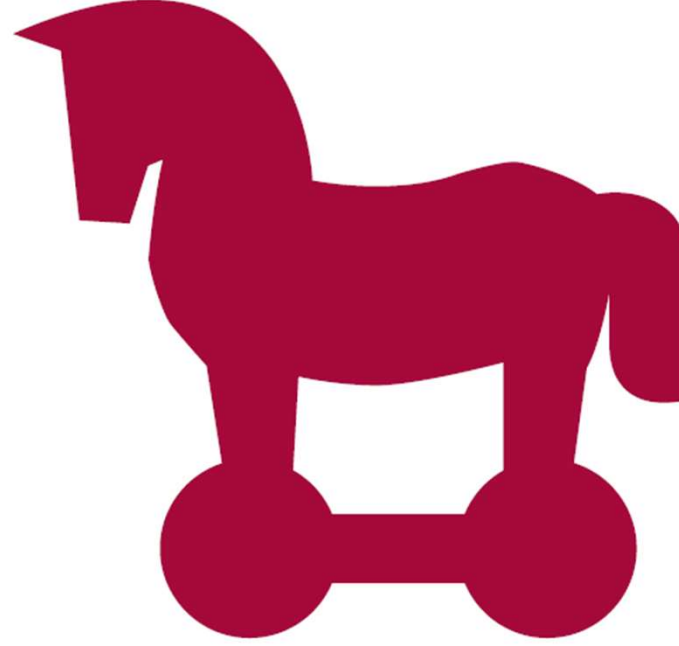


GANTEK ACADEMY

40+ YILLIK TECRÜBE İLE

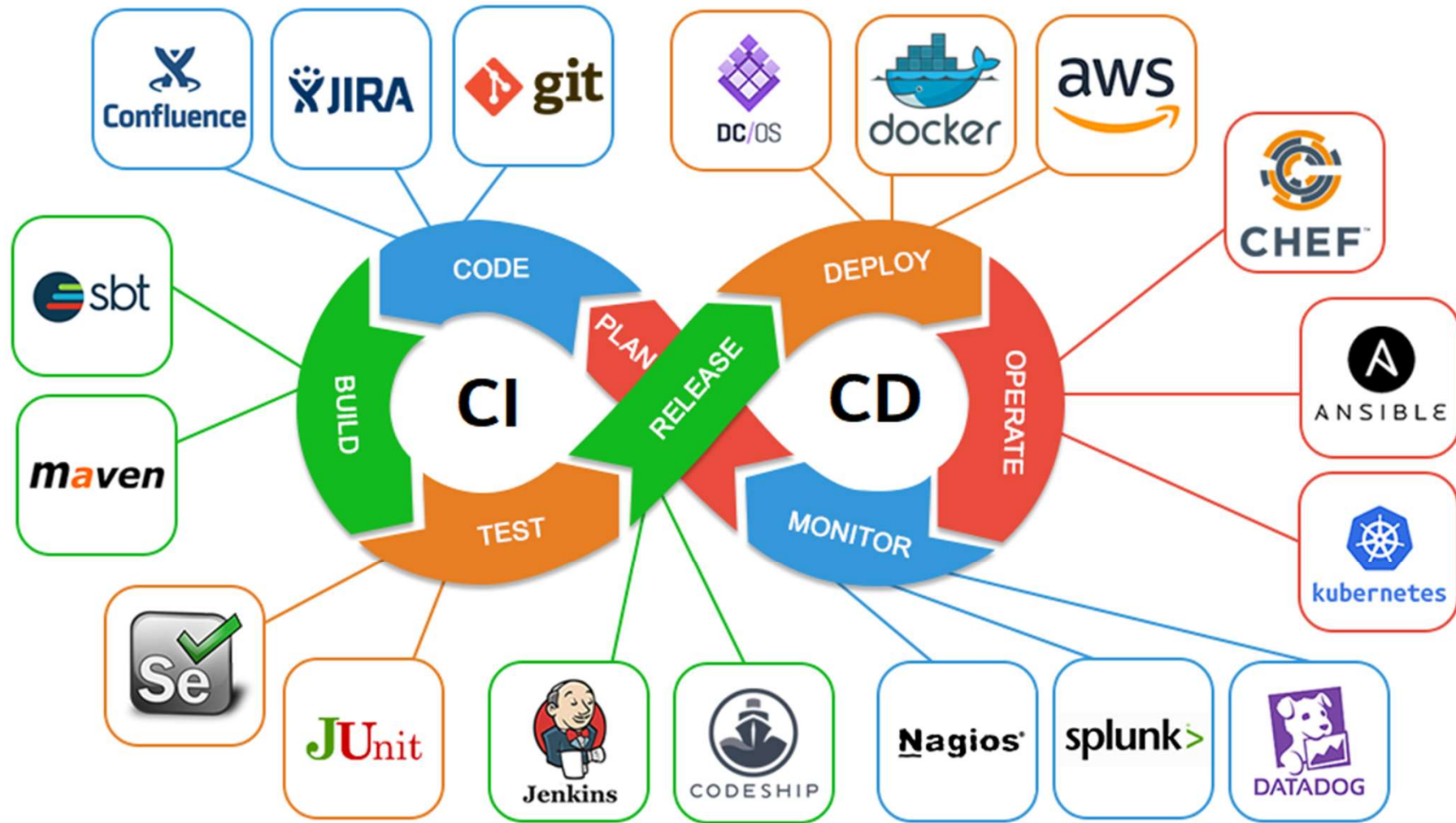


GANTEK

Ben kimim?

- Adım Hakan Güray Şenel.
- 1990 ODTÜ Elektrik Elektronik mühendisliği mezunuyum. 1993'te Vanderbilt Üniversitesi Elektrik ve Bilgisayar ve bilgisayar bilimleri bölümünden Yüksek Lisans derecesi aldım. 1997'de Vanderbilt Üniversitesi Elektrik ve Bilgisayar Mühendisliğinden doktor unvanıyla mezun oldum.
- İyi bir programcı olduğumu düşünüyorum. Sistem yazılımları, çok prosesli yazılımlar, işletim sistemleri, gerçek zamanlı ve gömülü sistemlerde programlama yaptım. Yapay zeka konularında çalışmalarım var.
- Uzman olduğum konular arasında Linux, gerçek zamanlı programlama, DevOps teknolojileri, PostgreSQL, açık kaynak kodlu sistemler, Bulut mimarileri, Kubernetes, OpenShift, vd yer alıyor.
- Programcılıktan daha çok Üniversitelerde, belediyelerde, kamu kurumlarında ve finans kuruluşlarında , IT sektöründe yöneticilik ve danışmanlık yaptım. Yazılım projelerinde yönetici/direktör olarak çalıştım.
- Son iki yıldır Gantek'te eğitim direktörü olarak görev yaptım. Ardında da 2023 itibariyle CIO oldum.

DevOps Mühendisinin Yol Haritası



NE ÖĞRENECEĞİZ?

Bu eğitimin sağladığı yeterlilikler

- CI/CD süreçlerinin neden gerekli olduğunu anlatabileceksiniz.
- DevOps pratiklerinin firmalara sağlayacağı faydaları listeleyebileceksiniz.
- Jenkins'i kurarak, herhangi bir proje için CI/CD döngüsü(pipeline) kurabileceksiniz.
- Lokalde git deposu oluşturacak ve Github, Gitlab ve Bitbucket'a SSH üzerinde güvenli şekilde gönderebileceksiniz.
- Lokaldeki git kod deposunda, «**commit**», «**merge request**» yapabilecek ve uzak kod deposuyla eşleştirebileceksiniz.
- Bitbucket ve Gitlab'da CD süreçlerini çalıştırmak için gerekli yapılandırmayı yapabilecek ve YAML dosyaları oluşturabileceksiniz.
- Statik kod analizi ve birim setlerini CI/CD döngülerine ekleyebileceksiniz.

ÖNEMLİ NOT

- İlk hafta anlatacaklarımıza bakarak, bu eğitimin AKADEMİK bir eğitim olduğuna hemen karar vermeyin.
- BU EĞİTİM AKADEMİK DEĞİL PROFESYONEL İNSANLARIN İHTİYAÇLARINA YÖNELİK HAZIRLANMIŞTIR.
- İlk hafta, temel kavramları öğreneceğiz. Çünkü teknoloji eğitiminde, aradaki bilgi boşluklarını doldurmak çok önemlidir.
- Bir teknolojiyi öğrenirken, o teknolojinin ne amaçla geliştirildiği, hangi aşamalardan geçtiği, ürünün hangi gereksinimler doğrultusunda değişim/dönüşüm geçirdiğini öğrenmek çok önemlidir. Çünkü gelecekte neyle karşılaşacağınızı ancak bunları öğrenerek bilebilirsiniz.
- Bol uygulama yapacağız. Ama sizin de benimle birlikte uygulama yapmanız çok önemli. Bu amaçla
 - Ubuntu 22.04 işletim sistemini sanal olarak makinenizde kurunuz.
 - İş yeriniz sanal makine kurup bu tür bir faaliyete katılmanızı istemiyorsa, evinizdeki bilgisayarını getirerek eğitime katılın.

Konteyner temelli teknolojiler

- Konteyner, son derece basit kelimelerle işletim sisteminden izole edilerek çalışan paketlenmiş bir sistemi ifade eden bir kelime olduğu söylenebilir.
- Konteynerler bir yazılım paketleme (image) yaklaşımıdır. İşletim sisteminde belirli kriterlere göre ve altta çalışan işletim sisteminden izole şekilde bu paketin çalıştırılması sağlanır.
- İzolasyon yaklaşımı bir çeşit sanallaştırma sağlar ve bu yaklaşıma işletim sistemi seviyesinde sanallaştırma adı verilir.
- Konteynerler arasında sanal ağ sistemi kurulabilir ve belirli kriterlere göre konteynerlerin birbirleri arasında haberleşmeleri sağlanabilir.

Konteyner temelli teknolojiler

- Konteyner temelli teknolojiler, yazılımların paketlenmesini ve hızla devreye alınmasını sağlayan önemli bir teknoloji haline gelmiştir.
 - Geliştirilen yazılım, gerekli kütüphaneler, yardımcı programlarla birlikte paketlenerek bir imaj haline getirildiğinde, bu imaj herhangi bir sistemde çalışabilir hale gelmektedir.
 - Bu imaj, işletim sisteminde kaynak kullanımı sınırlandırılacak çalıştırılabilir.
 - Sanal ağ sistemi içinde diğer konteynerlerle beraber çalışabilir.
 - Ortak disk alanlarına erişebilir.
- Bulut temelli sistemler için temel teknolojilerden biridir.
- DevOps konularında ve CI/CD süreçlerinde konteyner temelli sistemler, de facto bir standart haline gelmiştir.

git

- Yazılım geliştirme süreçlerinde kullanılan, kaynak kodu yönetim ve sürüm kontrol sistemidir.
- Git, Linux kaynak kodunun yönetilmesi amacıyla 2005 yılında Linux Torvalds tarafından yazılmıştır.
- Temel olarak bir **DevOps** aracıdır ve küçük büyük fark etmeksizin her türlü çapta projede kullanılmaktadır.
- Git, kaynak kodunda olan değişiklikleri takip etmekte ve bir çok geliştiricinin aynı kaynak kodu üzerinde çalışmasını sağlamaktadır.
- Merkezi git sunucusu sayesinde her geliştirici, projenin kaynak kodunun kopyasını kendi lokal disk sisteminde barındırmakta ve kaynak kodu dosyaları üzerinde yaptığı değişiklikler git tarafından algılanarak ve depoya aktarılarak diğerleri tarafından görülebilmesi sağlanmaktadır.

Git'te Dal (branch) kavramı

- Merkezi sürüm ve kaynak kodu yönetim sistemlerinin en büyük sorunu, aynı dosyada aynı alanlarda farklı geliştiricilerin benzer değişiklikler yapmasıdır. Bunların yönetilmesi zordur.
- Git'in en önemli avantajlarından biri, ana geliştirme dalından farklı bir dal üzerinde geliştirme yapması imkanındır.
- Dal, proje dosyasında geliştiricilerin yapacağı değişikliklerin ayrı bir kaynak kodu kopyasında yapılmasını sağlayan bir olanaktır.
- Dal üzerinde yapılan iyileştirmeler, ana dalı etkilememekte ve ancak stabil bir sürüm elde edildiğinde ana dala birleştirilmektedir.
- Bu şekilde proje kodunun yapılan değişikliklerle bozulması engellenmekte ve değişikliklerin özellik dallarında yapılması sağlanmaktadır.

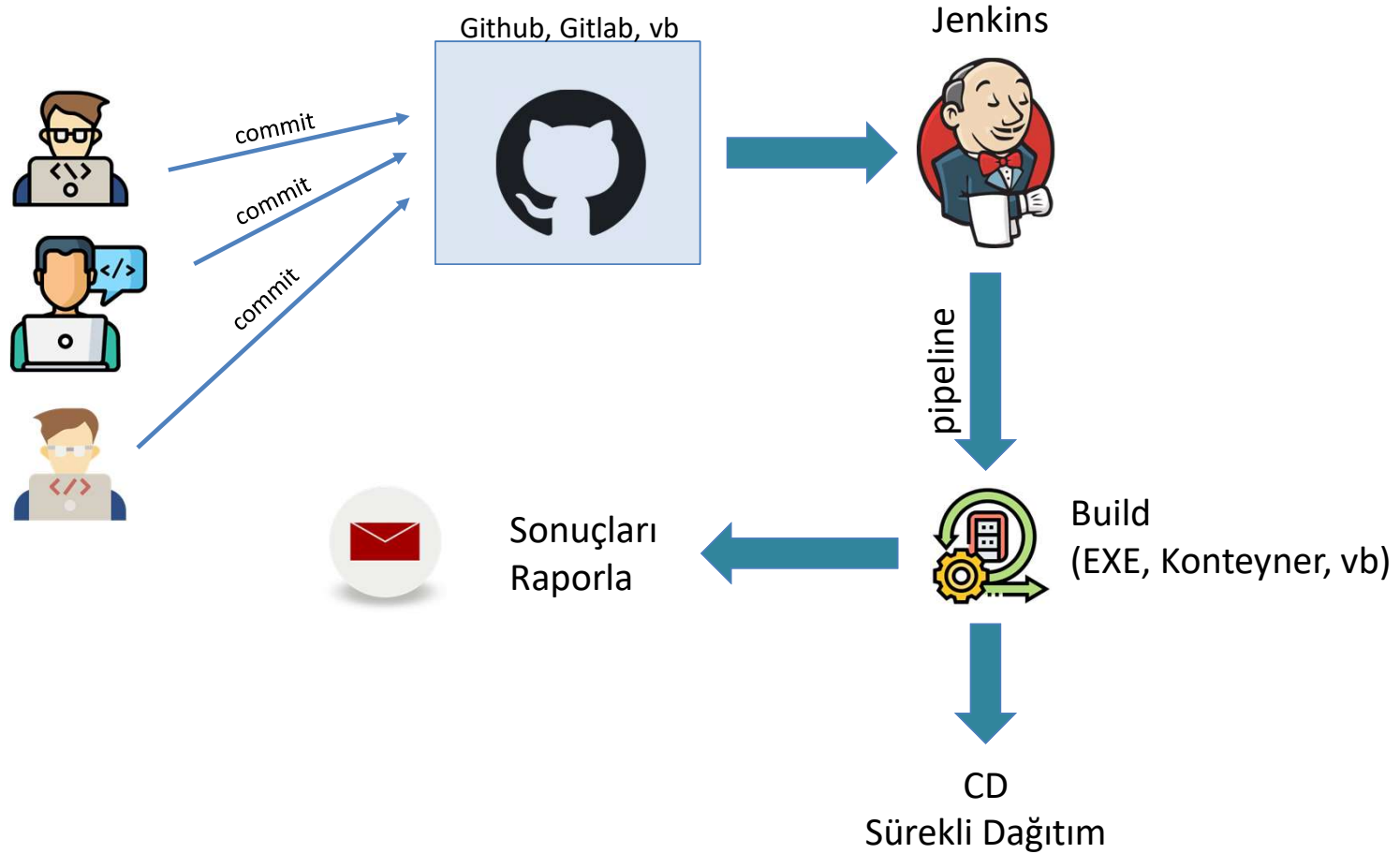
<https://www.simplilearn.com/tutorials/git-tutorial/what-is-git>



Jenkins nedir?

- Jenkins açık kaynak kodlu bir otomasyon sunucusudur.
- İlk sürümü 2011'de çıkmıştır.
- Sisteme eklenebilen çok sayıda eklenti (plugin) sayesinde yazılım geliştirme süreçlerinin, kodlama, sınama, dağıtım ve diğer aşamalarında, otomasyon amacıyla kullanılabilir.
- Plugin mekanizması sayesinde yeni özellikler eklenebilir ve önceki özellikler geliştirilebilir.
- Web arayüzü üzerinden yapılandırılabilir.
- CI/CD süreçlerinde yer alabilir. İstenirse CI tarafında çalışırken, istenirse CD tarafı için yapılandırılabilir.
- Linux, Windows, MacOS gibi ortamlarda çalışabilir.
- Docker konteyner olarak Docker Hub'da imajları mevcuttur.

Jenkins nasıl çalışır?



Github nedir?

- Sürüm kontrol sistemi olarak **git**'i kullanan ve yazılım geliştirme projelerinin depolanması, değişikliklerin izlenmesi ve üzerlerinde ortak çalışılmasını sağlamak üzere tasarlanmış web temelli kod depolama sistemidir.
- 2008'de «Tom Preston-Werner», «Chris Wanstrath», «P. J. Hyett» ve «Scott Chacon» tarafından geliştirilmiştir.
- IT sektöründe, açık kaynak projelerin dağıtımının yapıldığı önemli bir sistem haline gelmiştir.
 - Örneğin, güvenlik sistemlerindeki açıkların kullanıldığı kodlar ve programlar, buradan dağıtılmaktadır.

Gitlab nedir?

- Github'a benzer bir sistemdir.
- Gitlab sunucunu, kendi sisteminizde lokalde veya konteyner olarak kurabilirsiniz.
- Açık kökenli projedir. Ücretsiz olarak «toplum» sürümü kullanılabilir. Daha çok hizmetin ve özelliğin sunulduğu «kurumsal» sürümü ücretlidir ve destek hizmeti sağlanmaktadır.
- Gitlab içinde şu sistemler bulunur:
 - Git
 - NGINX
 - PostgreSQL
 - Chef
 - Redis
- İçinde CI/CD araçları bulunur. Github için kendiniz kurmanız gerekir.

<https://about.gitlab.com/install/?version=ce>

Bitbucket nedir?

- Yazılım geliştirme ekipleri için kod depolama, yönetme ve takip etme gibi özellikleri olan bulut temelli bir çözümdür.
- Git sistemi dışında, bazı önemli özellikleri bulunmaktadır:
 - «**pull request**»lerinde kodun gözden geçirilmesi için bir çözüm sunmaktadır.
 - JIRA entegrasyonu bulunmaktadır. Kod içindeki hataların tespiti ve takibi için Bitbucket'a entegre edilebilmektedir.
 - Her kod parçası için yorum eklenebilmektedir.
 - Github üzerinde kimlik doğrulama yapılabilmektedir.
 - 5 kişilik takımlara kadar ücretsizdir.
 - Farklı tür repoları (SVN, CVS, vb) sisteme yükleyebilmektedir.
 - Trello ile entegre edilebilmektedir.

Ansible nedir?

- Açık kaynak kodlu ve ücretsiz konfigürasyon otomasyon yazılımıdır. Ücretli versiyonunda bazı ek özellikler bulunur.
- Michael DeHaan tarafından yazılmıştır ve 2013'te kurulan Ansible Inc. firması 2015'te Red Hat tarafından satın alınmıştır.
- Ansible'ın kelime kökeni 1966'da yazılmış bir fantastik romanda kullanılan ve kurgu ürünü anlık iletişim sistemine dayanmaktadır.
- Linux/Unix'te SSH ve Windows sistemlerde Windows Remote Management arayüzüyle PowerShell ile hedef bilgisayarlara bağlanarak, orada istenen işleri gerçekleştirmek üzere kullanılır.
- Genellikle Python'da yazılmıştır ve çok sayıda işlevsel modülü bulunur. Her modül, birden çok kez çalıştırılsa da aynı şekilde sonuç üretir (idempotent).

DevOps, CI/CD, Docker, git, Jenkins, Ansible,
Github, Gitlab, Bitbucket

Yazılım Projelerinde Yaşanan Sorunlar

- Yazılım projelerinin zamanla gitgide daha da karmaşıklaşması, birçok sorunu beraberinde getirmiştir:
 - Zaman çizelgesine uyulamaması veya takip edilen zaman çizelgesinin gerçekçi olmaması
 - Sistem gereksinimlerinin netleştirilememesi
 - Müşterilerden gelen taleplerin bitmemesi
 - Yazılıma çok fazla özelliğin eklenmeye çalışılması
 - Yaşanan entegrasyon/derleme/sürüm çıkarma sorunları
 - Takımlar arası iletişimin zayıflığı
 - Takım içindeki iletişim kopukluğu
 - Yazılımın kullanılacağı kuruluşlarda kullanıcı talepleri ve üst yönetim talepleri arasındaki uçurum
 - Kalite güvence çalışmalarına zaman kalmaması
 - Kullanıcıların, kullandıkları eski yazılımların değiştirilmesini istememesi

Yazılım proje kuralları

- **KURAL 1:** Müşteri ve yazılım geliştiren ekip çok iyi geçinememez ve aralarında sürtüşmeler olması doğaldır.
- **KURAL 2:** Geliştirilen yazılım, müşterinin beklentilerini karşılamaz. Müşterinin beklediği yazılımın ihtiyaçlarını karşılamasıyken, yazılım geliştirme ekibi analiz ve tasarım süreçlerinde kendilerine verilen gereksinimleri karşılamaya gayret eder.
- **KURAL 3:** Müşteri, gereksinimlerini tam anlamıyla iş analistlerine aktaramaz ve aklına geldikçe «şu da vardı» diye belirtir.
- **KURAL 4:** Hiçbir yazılım geliştirme projesi sorunsuz geçmez. Sorunsuz geçiyorsa bilin ki sonu kötü bitecektir.
- **KURAL 5:** Hatasız kul olmadığı gibi hatasız kod da olmaz.
- **KURAL 6:** Hatalar genellikle demo yapılırken ortaya çıkar ve «şirkette çalışıyordu bu ya» diye savunma yapılır.
- **KURAL 7:** Zamanında bitirilen proje nadirdir hatta yoktur denilebilir. Genellikle beklenen sürenin iki katı kadar ve hatta daha fazla süre tutar.
- **KURAL 8:** Proje başında hesaplanan adam/ay ve bitiş süresi tahminlerinin tutmama olasılığı %100'dür. Tutarsa yasal süreç başlar ve iptal edilen sözleşmeler havada uçuşur.

Sürenin Aşılması

- **90-90 kuralı** (ninety-ninety rule) olarak bilinen aforizma şunu der:
 - Bir projedeki ilk %90 kod, geliştirme süresi için ayrılan sürenin %90'ında tamamlanır ama geri kalan %10 kod geliştirme süresinin %90'ında tamamlanır.
 - Nasıl yani? $\%90 + \%90 = \%180$
 - Her proje planlanan süreyi **kesinlikle** aşar.
- **Pareto prensipi:** Sonuçların %80'i, nedenlerin %20'sinden kaynaklanır:
 - %20 kodun yazılması %80 zaman alır, %20 zamanda kodun %80'i yazılır.
- **Hofstadter yasası:** «*Hofstadter yasasını dikkate alsan da her şey beklenenden daha uzun sürer*»

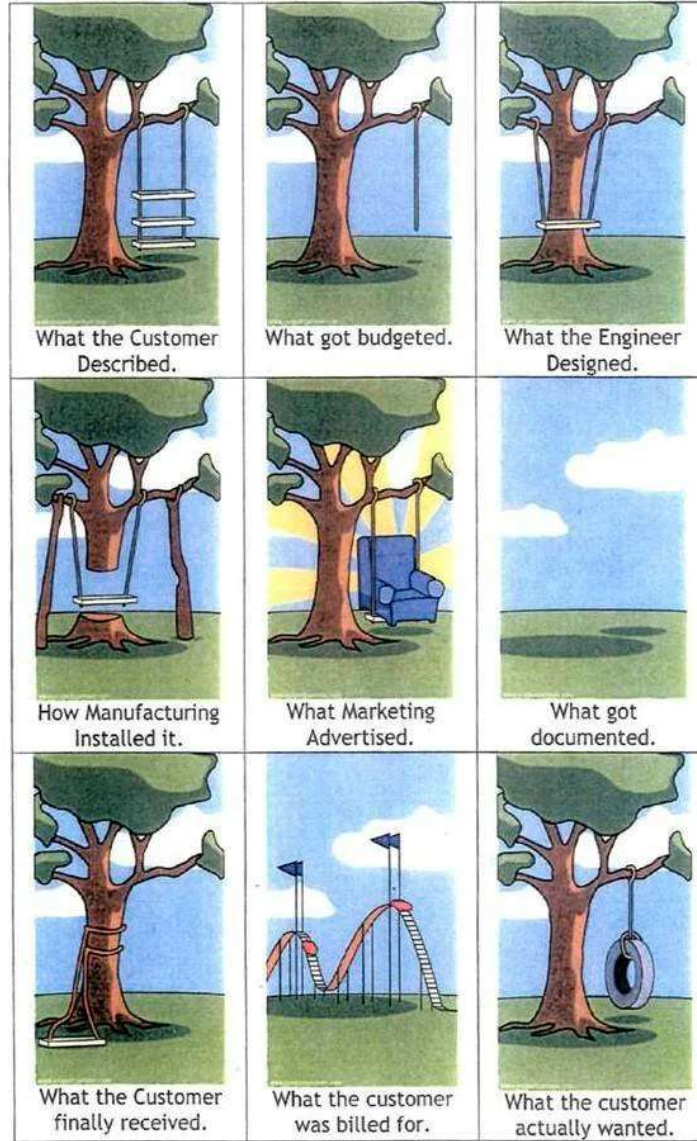
Gereksinim Analizi



Müşterinin gerek duyduğu ve gereksinim analizinde tarif ettiği



Proje sonucunda elde edilen ürün



Tespitler

- Bilgisayar sektöründe geçen yarım yüzyılı aşkın çalışmalar, yazılım geliştirme işinin sadece kod yazmak olmadığını göstermiştir.
- **HIZLI ÜRÜN ÇIKARMAK:** Projenin (süreçlerin, kaynakların, müşteri taleplerinin/beklentilerinin, vb) uygun şekilde yönetimi ve yazılım hazırlama sürecinden itibaren en hızlı şekilde ürünü müşteriye sunmak en önemli görevlerdir.
- **DEVOPS EKİBİ:** Farklı takımların yazdığı kodun bir araya getirilmesi kadar, bu kodun derlenmesi ve müşteriye sunulmak üzere hazırlanması süreçlerini yürütecek ve bunların çalışacağı sistemleri hazırlayacak nitelikli elemanların varlığı da önemlidir.
- **ÇEVİK ÜRETİM:** Müşteriye, mümkün olduğunca sık aralarla ve özellikler açısından artırımlı şekilde bir ürün çıkarılması, yazılım geliştirme süresini azalttığı tespit edilmiştir.
- **DERLEME/ENTEGRASYON SÜREÇLERİNİN SIKLAŞTIRILMASI:** Her kod değişikliğinde derlemenin yeni baştan yapılması, kodlama hatalarının daha erken bulunabileceğini göstermiştir.

Proje Yönetimi

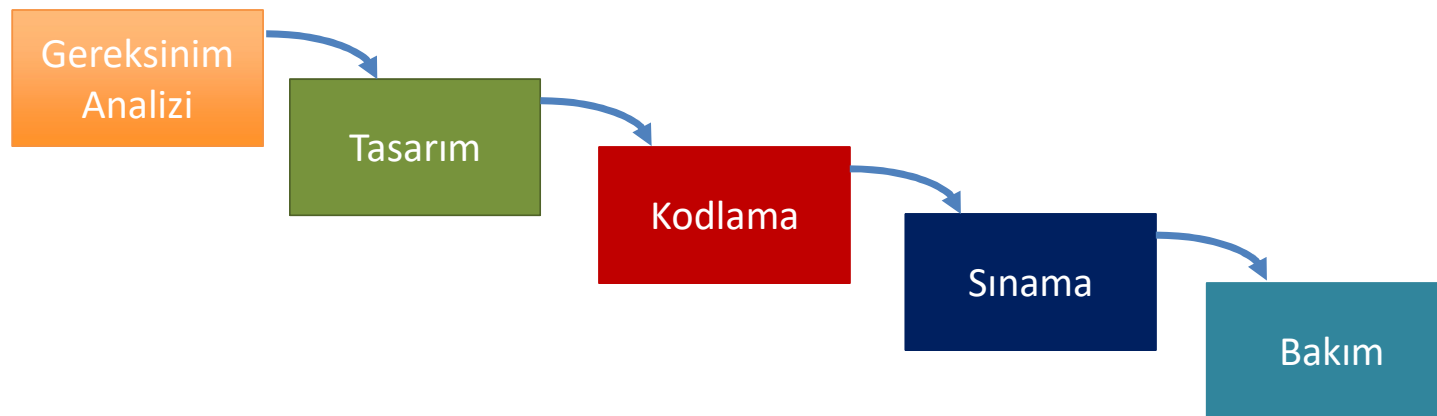
- Yazılım projelerinin yaygınlaşmasıyla birlikte çok sayıda proje yönetim metodolojisi geliştirildi.
- Yönetim metodolojisi aynı olsa da her proje aynı şekilde yönetilebilir mi?

HAYIR?

- Hatta, önceki projenizin yönetimi şu andaki projenizde aynı olmayabilir.
- Hangi yönetim metodolojisini tercih edeceğiniz konusu aşağıdakilerle ilgilidir:
 - Maliyet ve bütçe
 - Takım büyüklüğü
 - Risk alabilme durumu
 - Esneklik
 - Zaman çizelgesi
 - Müşteriyle olan ilişkilerin düzeyi

Proje Yönetimi: Şelale

- Projelerin teknolojik olarak geliştirilmesi yanında ürünü kullanıcıya istenen şekilde ve sürede ulaştırabilmek için proje yönetiminin ne kadar önemli olduğu 1960'larda itibaren fark edilmiştir.
- 1970'de W. Royce «Waterfall» proje yönetim metodolojisini tanıtan bir makale yayınlamıştır. Bu metodolojide, proje yönetimi, gereksinimlerin belirlenmesi, tasarım, kodlama, sınama, bakım gibi ardışık görevlerden oluşur
- Bir görev bitmeden sonrakine geçilmez (şelaleden suyun akışı gibi)



Çevik Proje Yönetimi

- Şelale yöntemindeki sorunlardan biri, gerçekte yazılım süreçlerinin uzun sürmesi ve projenin başında müşteriden toplanan gereksinimlerin zaman içinde değişmesidir.
- Bu nedenle, yazılım süreçlerinin hızlandırılması için, 2000'li yıllarında başında çevik proje yönetim metodolojileri önerildi.
- Buradaki amaç, yazılımın geliştirilmesi ve sunulması arasında geçen süreyi azaltırken, müşteriye tam olarak gereksinimleri karşılamasa da bir sürüm sunabilmektedir.
- Yazılım geliştirme süreçlerinde hedef, döngüsel bir yaklaşımla, örneğin ayda bir çalışan bir sürüm sunabilmektir.
- Çalışan sürümü kullanan müşteriden geri dönüt alınarak, talepleri ve işaret ettiği sorunların çözümleri bir sonraki iterasyonda planlanmaktadır.
- Farklı çevik proje yönetim metodolojileri bulunmaktadır.
 - Scrum (İngilizce itişme kakışma anlamına gelir)
 - FDD: Feature Driven Development
 - ASD: Adaptive Software Development
 - LSD: Lean Software Development
 - XP: Extreme Programming
 - vd

Proje Yönetim Metodolojileri

- **Günümüzde, yazılım firmaları için önemli olan, hazırlanan yazılımı mümkün olan en kısa sürede müşterinin önüne koyabilmektir.**
- Ancak, günümüzde, eskide olduğu gibi kaynak kodları derlenip, çalıştırılacak kod bir kurulum paketi haline getirilip müşteriye sunulmuyor.
- Web temelli sistemlerin çoğunluğunda, kod derleme dışında, sistemin sunucular/konteynerler üzerine kurulması, test edilmesi ve izlenmesi gibi işler de olur.
- Çevik tarzdaki proje yönetiminde, «sprint» adı verilen 1-2 haftalık geliştirme döngülerinin sonunda yazılım derlenir ve müşteriye sunulur.
- Kısa süreler içinde yapılan kodlama, derleme, test, dağıtım ve kurulum gibi faaliyetlerin, sprintlerin sonunda başarıyla yürütülmesinin insan gücüyle yapılması neredeyse imkansızdır.
- Otomasyon araçları kullanarak, kodlamanın başladığı andan dağıtıma (deployment) kadar giden sürecin otomatik hale getirilmesi gerekir.

Entegrasyon Kavramı

- Yazılım mühendisliğinde, entegrasyon kavramı çok sık kullanılır ve «derleme» ve «çalıştırılabilir kod oluşturma» ve «kurulum dosyası oluşturma» kavramlarıyla karıştırılır.
- **UNUTMAYIN:** Yazılım geliştirme faaliyetlerinin sonucunda sadece yazılımın derlenmesi, paketlenmesi gibi işler yapılmıyor.
- Entegrasyon (integration) çok sayıda farklı bileşenden (farklı sistemler, kodlar, betik programlar, DB, API'ler, farklı kütüphaneler/sistemler – Kafka, Redis, vb) oluşan bir projenin, parçalarının uygun şekilde birleştirilerek bir (veya daha fazla) sistem haline getirilmesine denir.
- Entegrasyon faaliyetlerinden sonra, bu sistemin test edilmesi ve ardından teslimatının (deployment) yapılması gerekir.

Takım içinde gerçekleşen çalışmalar

- Çok sayıda takım ve yazılımcı olan bir ortam düşünün.
- Her takımın genel bir görevi, takım elemanlarının da bireysel görevleri bulunur.
- Projenin bütününe bir depoda tutan ve programcılarının checkout işlemiyle kodun son sürümünü (çalışabilir sürümünü) kendi makinelerine alabildikleri sürüm kontrol (version control) sistemleri kullanılır.
- Takım elemanı, sorumlu olduğu kısmı değiştirir ve sadece değişiklik yaptığı kısmı test edersiniz.
- Mükemmel çalıştığına emin olarak kodu depoya geri yüklersiniz. Ancak birim olarak çalışan kod tüm proje içinde istendiği gibi çalışmayabilir (entegrasyon sorunu).

Sürüm Çıkarma Süreci

- Yeni bir sürüm çıkarılacağı zaman, bütün takımlara kodları depoya yüklemeleri için tarih verilir.
- Bütün takım elemanlarının görevlerini bitirdikleri ve kodu depoya/havuza yükledikleri emin olduğu zaman, derlemeyle ilgili bir kişi projenin tamamını sürüm kontrol sisteminden kendi makinesine «checkout» ederek indirir ve derlemeye başlar.
- Programın derlenmesi sırasında çıkan derleme hataları entegrasyon sorunlarıdır. Takım liderleri çıkan sorunlara karşı uyarılır ve düzeltmeleri istenir.
- Derlenebilirse, çalışan kod test makinesine yüklenir ve takımların çalışan kod üzerinde testleri yapmaları istenirdi.
- Genel test prosedürleri uygulanır ve ardından otomatik testlere başlanır.
- Bu süreç en iyi ihtimalle 1-2 hafta hatta bir ay sürebilirdi.

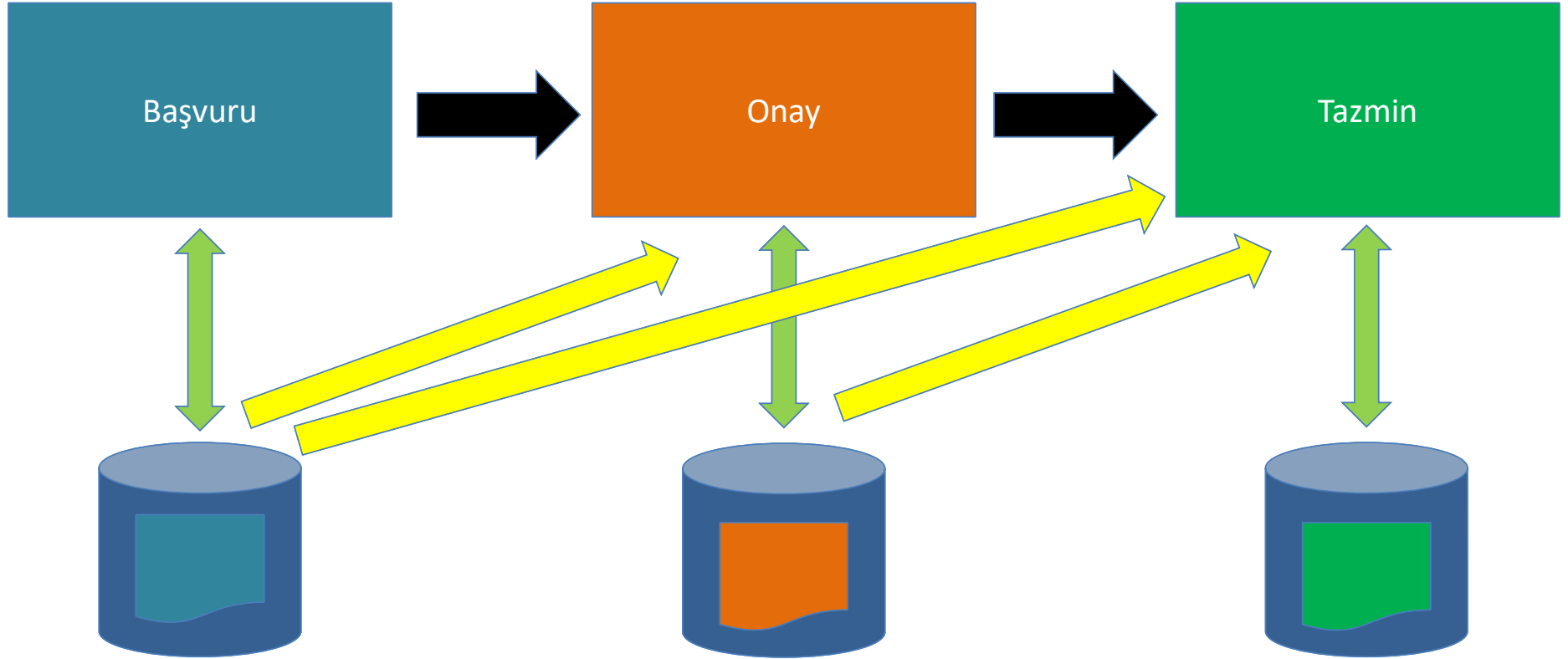
Entegrasyon Sorunları

- Yazılım geliştirme faaliyetleri içinde sorun yaratan ve yazılım geliştirme sürecini uzatan en önemli unsurlardan biri entegrasyon faaliyetleridir.
- Çok sayıda yazılım geliştirme takımından oluşan ve farklı bileşenler ve tanımlı süreçler bulunan bir projedeki belki de en sorunlu anlardan biri, sürüm çıkarmak için entegrasyon aşamasına geçilmesidir.
- Yazılım ekipleri hazırladıkları kodları kod havuzuna yüklemekten önce test etmelerine rağmen, tüm sistemin entegrasyonunda sorun yaşanabilir:
 - Bir bileşenin API'sindeki yöntemlerdeki değişiklik, bunu çağıran diğer bileşenlerin derlenmesini engelleyebilir.
 - Bir veritabanı tablosuna eklenen yeni bir alan, diğer takımlara duyurulmadığı zaman tüm bileşenler olumsuz etkilenir.
 - Bir önceki versiyonda yapılan işlemler için kaydedilen durum bilgileri, yeni sürümde hataya yol açabilir.
 - Bir takımın hazırladığı koddaki sorun, diğer takımın hazırladığı koddaki hataları tetikleyebilir ve çalışma zamanı hatası alınmasına yol açabilir.

Örnek

- Bir bankada, bir müşterinin kredi başvurusuyla başlayan süreç üç adımda tamamlanmaktadır: Başvuru → Onay → Tazmin
- **KREDİ BAŞVURU:** Başvuru müşteri tarafından web/app üzerinden yapıldığında, veritabanındaki bir tabloya başvuru sırasında veriler eklenmektedir.
- **KREDİ ONAY:** Başvuru tamamlandıktan sonra, banka kullanıcısı, bu başvuruyu görmekte ve işleme alarak açmaktadır. Bu sırada başvurudaki bilgileri görmekte (veritabanındaki tabloya erişmekte), aldığı notlar, girdiği bilgiler, farklı kaynaklardan (Findeks, Mersis, NVİ, VD) aldığı veriler veritabanındaki farklı bir tabloya yazılmaktadır.
- **TAZMİN:** Onayı alınan bir kredide geri ödeme konusunda sorun yaşanırsa zararın tazmini için adım atılmaktadır. Tazmin sürecinde, hem başvuru hem de tazmin tablolarına erişilmekte ve tamamlandığında, girilen veriler, alınan kararlar veritabanındaki tazmin tablosuna yazılmaktadır.

Örnek - Entegrasyon



Entegrasyon Sorunları

- Yazılım bileşenleri, farklı takımlar halinde geliştirildiğinde, kodların entegrasyonunda sorun yaşanması beklenir.
- Çevik proje yönetimi uygulansa da «sprint» periyodları azaltılsa da, entegrasyon sorunları, yazılımın derlenmesi ve devreye alınması konusunda ciddi sorunlar yaşanabilir ve sürüm teslimatında gecikmeler beklenebilir.
- Çözüm olarak yapılabilir?
 - *Entegrasyon her birkaç haftada bir değil günlük yapılmalı*
 - Her entegrasyon, otomatik araçlarla işlenerek, hatalar mümkün olduğunca en kısa sürede bulunmalı ve takımlara geri besleme sağlanmalı



SÜREKLİ ENTEGRASYON – CONTINUOUS INTEGRATION (CI)

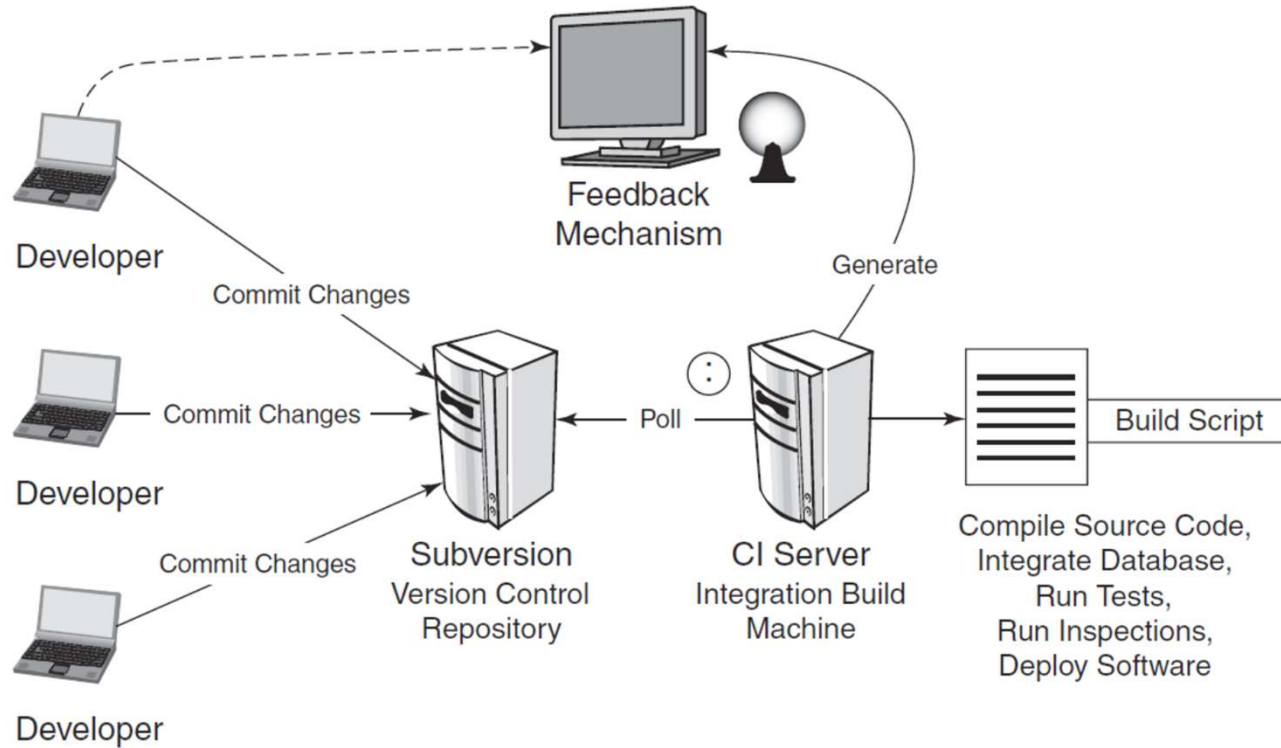
Paul Duval, Continuous Integration: Improving Software Quality and Reducing Risk, AW,2007

★ Sürekli Entegrasyon ne sağlar? ★

- Kod deposuna gönderilen *her kod değişikliğinde, 1 satırda değişiklik yapılsa dahi*, otomatik entegrasyon (derleme, vb) yordamları(pipeline) çalıştırılır ve derlenen kod üzerinde çeşitli testler uygulanmalıdır
 - Aslında kolay bir iş değil
 - Bu kadar sık yapılmasa da olur
- Bu şekilde, sorun ortaya çıktığı anda tespit edilir ve entegrasyonda/derlemede karşılaşılan hata mesajları ilgili takımlara gönderilerek, düzeltme yapmaları istenir.
- Sürekli entegrasyon (CI), yazılım ekiplerine, hatanın çıktığı anda müdahale imkanı sağladığından, ürünün daha hızlı çıkartılmasını kolaylaştırır.

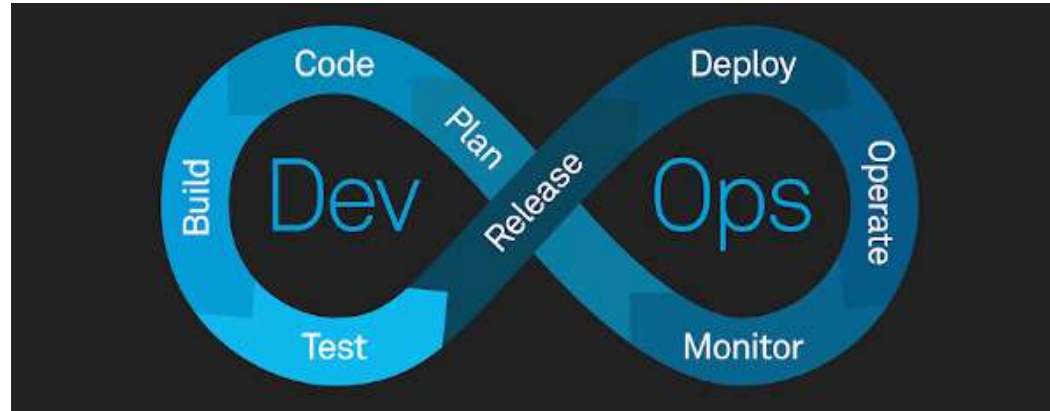
Sürekli Entegrasyon – Senaryo

- *Paul Duval*, 2007'deki kitabında **CI** kavramının uygulaması için bir senaryo sunmaktadır.



CI/CD (Sürekli Entegrasyon ve Sürekli Dağıtım)

- CI/CD, uygulama geliştirme sürecine eklenen otomasyon araçlarıyla uygulamaların geliştirildikçe müşterilere sunulması anlamına gelir.
- Sürekli entegrasyon ve sürekli dağıtım/sunum kavramlarıyla ifade edilir ve bir boru-hattı kavramıyla metafor haline getirilir.



<https://martinfowler.com/articles/continuousIntegration.html>

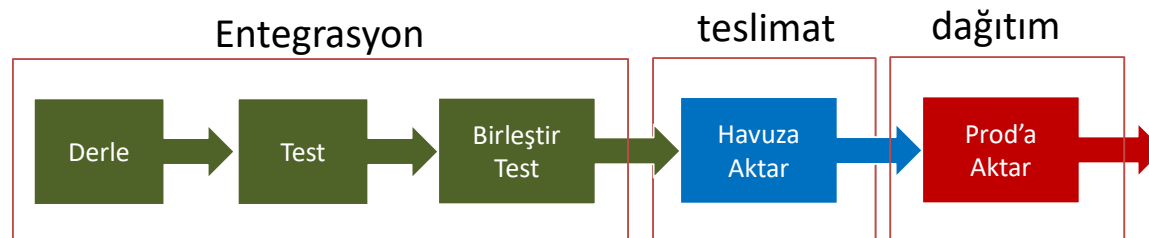
<https://cd.foundation/blog/2020/09/17/ci-cd-patterns-and-practices/>

CI/CD

- CI (Sürekli Entegrasyon) yazılım geliştirme takımlarını ve iş süreçlerini daha etkin hale getirmek için kullanılabilecek otomasyon işlerini kapsar.
- Amaç, çalıştırılabilir programın oluşturulmasını otomatik hale getirmektir. Başarılı bir CI, bir çok takımın parçalarını geliştirdiği uygulamanın düzenli aralıklarla oluşturulduğu (derlendiği, kodların üretildiği), test edildiği ve bir depoda birleştirildiği bir süreçtir.
- CD (Sürekli Dağıtım - Continuous Delivery/Deployment) kavramı, entegrasyonu yapılmış bir yazılımın üretim ortamına taşınmasıyla alakalı işleri tarif eder. Amaç, yazılımı «prod» ortamına taşımak, izlemek ve kullanılabilir hale getirilene kadar sinayarak müşteriye hatasız şekilde sunmaktır.
- CI/CD kavramları bir araya getirildiğinde, yazılımın yaşam döngüsü boyunca entegrasyon ve sinama fazlarına kadar geçen süreçlerde yapılan sürekli otomasyon ve izleme faaliyetlerini kapsar ve süreklilik arz eden döngüsel süreçler CI/CD iletim-hattı (pipeline) kavramıyla ifade edilir.

CI/CD

- Birleşik olarak CI/CD üç fazdan oluşur
 - Sürekli entegrasyon
 - Derle → Test et → parçaları bütünleştir → test et
 - Sürekli teslimat (delivery)
 - Kod havuzunu güncelle
 - Sürekli dağıtım (deployment)
 - Otomatik olarak «prod»a aktar



Yazılım testleri

- Yazılım test süreci yazılım geliştirme yaşam döngüsünün önemli bir parçasıdır:
 - Yazılımın, gereksinimleri karşılayıp karşılamadığını gösterir.
 - Sistemde olabilecek boşlukları, hataları ve eksik gereksinimlerin, yazılım müşteriye sunulmadan önce tespit edilmesini kolaylaştırır.
 - Uygulama kodu içinde hataların bulunmamasını ve kodun mantıksal hata içermemesini sağlar.
- Yazılım test süreci, CI/CD kanalları içinde önemli bir aşamadır ve çıkabilecek mantıksal hataların daha oluştuğu anda tespit edilmesi ve ardından hızla ortadan kaldırılması için elzemdir.
- Belirli bir dizi test mekanizmasından geçirilmeyen yazılımın, müşteri isteklerine uygun şekilde çalışması neredeyse imkansızdır. Test sonuçları müşterinin taleplerinin karşılandığını gösteren kanıt niteliği taşır.

Test tipleri, yöntemleri ve yaklaşımları

- Test tipleri ikiye ayrılır:
 - **Otomatik** : yazılım üzerinde çeşitli yazılımlar veya betik programlar sayesinde testlerin otomatik olarak gerçekleştirilmesi.
 - **Elle**: testlerin, belirli senaryoları takip ederek yazılım geliştiriciler veya test elemanları tarafından yapılması
- Test yöntemleri
 - **Statik**: kaynak kodu üzerinde yapılan testler
 - **Dinamik**: programı çalıştırarak, girdilerini ve çıktılarını kontrol edilmesine dayanan testler.
- Test yaklaşımları:
 - **Kutu yaklaşımı**
 - **Kara kutu**: testi gerçekleştiren kişi, programın iç yapısını bilmez
 - **Beyaz kutu**: testi gerçekleştiren kişi, programın iç yapısını bilir ve buna göre testi yapar.
 - **Gri kutu**: kara kutu ve beyaz kutu yaklaşımlarının birleşimidir. Testi yapan kişi programın iç yapısını çok fazla bilmez ve testi buna göre yapar.
 - Model temelli yaklaşım (istatistiksel bilgilerle ve hata yüzdeleriyle)
 - Uluslararası standartlara dayalı yazılım geliştirme
 - Buluşsal (heuristic) yöntemler

Test standartları

- Yazılım testleri içinde çeşitli standartları geliştirilmiştir.
 - ISO/IEC 9126: yazılımların kalitesini artırmaya yönelik metodolojiyi tanımlar.
 - ISO/IEC 9241-11: Etkin ve yeterli bir ürünün nasıl olması gerektiğini tanımlar.
 - ISO/IEC 25000:2005: Software Quality Requirements and Evaluation (SQuaRE) için gerekli yöntemleri tanımlar
 - ISO/IEC 12119: müşteriye teslim edilecek yazılım paketleriyle ilgilidir. Gereksinimler ve test süreçlerini tanımlar.
 - Diğerleri (IEEE829, IEEE1061, IEEE1059, IEEE1008, IEEE1012, vb)
- Farklı sistemler için (aviyonik, askeri, medikal, vb) farklı kalite standartları ve metodolojileri de bulunmaktadır.

https://www.tutorialspoint.com/software_testing/software_testing_iso_standards.htm

DEVOPS NEDİR?

DevOps nedir?

- DevOps, kelime olarak, yazılım geliştirme (**Dev**elopment) ve IT operasyonları (**Ops**) kelimelerinden oluşturulmuştur.
- DevOps altında, yazılım geliştirme (**de**velopment) ve operasyon (**op**erations) ekipleri ortak bir hedef doğrultusunda çalışır.
- DevOps çevik (agile) yazılım geliştirme süreçlerini tamamlar niteliktedir.
- DevOps'un amacı, otomasyon araçlarını kullanarak yazılım geliştirme yaşam döngüsünü kısaltmak ve yazılımın müşteriye dağıtımında süreklilik sağlayabilmektir.
- DevOps, günümüzde, bir yazılım firmasının geliştirdiği uygulamaları en hızla şekilde kullanıcılara sunabildiği yöntemlerin tamamı anlamına gelir.

<https://www.atlassian.com/devops/what-is-devops>

<http://radar.oreilly.com/2012/06/what-is-devops.html> ← Mutlaka okuyun

DevOps - Tanım

- Devops tanımını «Gartner Teknoloji» sözlüğünde şu şekilde yer almaktadır:
 - DevOps, *IT kültüründe*, tüm sisteme yönelik bir yaklaşımla yürütülen ve IT servislerinin hızlı teslimine odaklanan çevik ve yalın pratikleri ifade eder.
 - DevOps, insanlara ve dolayısıyla kültüre odaklanır ve operasyon ve yazılım geliştirme ekipleri arasında işbirliklerini geliştirme yolları arar.
 - DevOps, yazılım geliştirme döngüsünde, programlanabilir ve dinamik altyapı sağlayan otomasyon araçları gibi teknolojik araçlar kullanır.
- Kısacası, DevOps, yazılım geliştirme süreçlerini hızlandırmak, iyileştirmek ve güncellemeleri daha hızlı ve sağlıklı yapabilmek için kullanılan yazılım geliştirme proseslerini, kurumdaki kültürel değişikliği ve farklılaşan zihinsel süreçleri içeren bir terimdir.

<https://www.gartner.com/it-glossary/devops/>

<https://newrelic.com/devops/what-is-devops>

DevOps nedir?

- DevOps IT sektöründe *kültürel bir değişiklik* anlamına gelir ve yazılımın kodlanmasından sunulmasına (izlenmesine) kadar olan sürecin otomasyon araçlarıyla gerçekleştirilmesi ve bu şekilde hızlandırılması ilkesine dayanır.
- DevOps takımları müşterilerden elde ettikleri geri beslemeleri ve çalışan yazılımlardan elde edilen gerçek zamanlı izleme verisiyle, gelecekte yazılım üzerinde yapılacak geliştirmeleri planlar.
- DevOps için önemli olan son üründür ve son ürün dikkate alınarak bütün süreçler planlanır ve entegrasyonu ve teslimi otomatik hale getirilir.

<https://www.atlassian.com/devops/what-is-devops>

DevOps - Tarihçe

- DevOps kavramı bir anda ortaya çıkmadı.
- Yazılım sektöründe geleneksel yazılım geliştirme süreçlerindeki sorunların iyiden iyiye hissedilmeye başlanmasıyla birlikte 2007 yılında dile getirildi.
- O zamana kadar IT sektöründe, yazılım geliştiricileri ve IT operasyonlarını yürüten ekipler farklı şekilde organize olmaktadır ve yöneticileri de farklıydı.
- Hatta, bazı durumlarda aynı firma bünyesinde yer alan iki ekibin birbirine çok dostça bakmadıkları da olmuştur.
- Her iki ekibin beraber çalışmamasının, bir çok projenin başarısızlıkla sonuçlanmasına neden olduğu fark edilmişti. Örneğin, aşağıdaki durumların projenin başarısız olmasını sağlayacağı açıktır:
 - Veritabanı yöneticisi ile veritabanı üzerinde kod yazan kişiler birlikte çalışmamakta ve birbirilerine doğrudan yardımcı olmamaktadır.
 - Yazılımı hazırlayan ekibin, yazılımın çalıştırılması için gerekli altyapıyı sağlayan takımla görüşmeden yazılımı hazırlamaktadır.

<https://newrelic.com/devops/what-is-devops>

DevOps Tarihçe

- Yazılım tesliminde kronikleşen sorunlarla ilgili yazılım geliştirme ekipleri ve operasyon ekiplerinin çeşitli konferanslarda ve online forumlarda bir araya gelmiş ve yaşanan sıkıntıların nedenleri tartışılmıştır.
- Yazılım geliştirmeye uğraşan ekiplerle ve sistem yöneticilerinin ortak görüşü, müşteri taleplerinin her iki grubu farklı yönere çekmesidir:
 - Yazılıma yeni özelliklerin eklenmesi talep edilmekte ve yazılım geliştirme ekipleri de kod üzerinde sürekli güncelleme yapmaktadır.
 - Yeni özelliklerin bulunduğu yeni sürümlerin kesinti yaşanmadan hızla devreye alınması ve stabil şekilde çalışması istenmektedir.
- Ancak güncellemelerin hızla devreye alınmak istenmesi ve birbiri ardına gelen güncellemeler, otomasyon kullanılmadığı için operasyon ekiplerine büyük yük oluşturmaktadır.
- Ayrıca, yazılım geliştirme ekiplerinin koddaki yaptıkları hızlı değişikliklerin, yazılım kalitesine olumsuz etkide bulunduğu ve yazılımın kararlılığını düşürdüğü düşüncesiyle, operasyon ekipleriyle geliştirme ekipleri karşı karşıya gelmektedir.
- Diğer bir deyişle, geleneksel yazılım geliştirme süreçlerinde, yazılım geliştirme ve operasyon ekiplerinin karşı karşıya geleceği bir çok durum ortaya çıkmaktadır.

DevOps Tarihçe

- Zamanla, odaklandıkları işler nedeniyle birbirinden ayrışan yazılım geliştirme ve operasyon ekipleri nasıl bir araya getirilebilir konusunda yapılan çalışmalar, her iki ekibin şu ilkelere göre hareket etmesi gerektiğini ortaya çıkarmıştır:
 - Her iki grubun beklentilerinin ve önceliklerinin ortaya çıkarılması gerekir. Bu bilgiler birbirlerine iletilmelidir.
 - Yazılım geliştirme ve operasyon süreçlerinde tekrar eden işlerin otomasyon araçlarıyla gerçekleştirilmesi, süreci hızlandıracaktır.
 - Her yapılan işin sonrasında, ekiplere geri besleme yapılması gereklidir. Her alt süreç en küçük ayrıntısına kadar izlenebilmelidir.
 - Beraber çalışma kültürünün oluşması için her türlü verinin ekipler tarafından birbirileriyle paylaşılması iyi olacaktır.
- Bu prensipleri göz önüne alarak, yazılım geliştirme ve operasyon ekiplerinin bir araya gelmesini temsil eden DevOps kelimesini, ilk olarak, 2009'da Patrick Debois kullanmıştır.

DevOps ve Çevik Proje Yönetimi

- DevOps kelimesi genellikle çevik (agile) ve yalın (lean) proje yönetim kavramıyla karıştırılmaktadır. Her iki yaklaşımın da hedefleri yazılımın kalitesini artırmak ve daha kısa sürede yazılımı hazırlamaktır.
- Çevik ve yalın proje yönetimi, iterasyonla geliştirme sürecinin kısaltılmasıyla alakalıdır ve kültürel değişimi işaret eder. Ayrıca, çevik yönetim tarzı için hangi aracın kullanıldığının önemi yoktur.
- Çevik yönetim yazılım için gereksinim analizi yapılmasından kodun tamamlanmasına kadar olan süreci ele alırken, DevOps bu süreci devreye alma ve bakıma kadar ilerletir.
- Çevik yönetim, yazılım geliştiricilerin ve ürün yöneticilerinin (sahiplerinin) etkileşime girmesine odaklanırken, DevOps operasyon takımlarını da sürece dahil eder.
- DevOps ise birbiriyle kesişen takımların birbirleriyle nasıl iş birliği yapabileceklerine yöneliktir.

DevOps ve Çevik Proje Yönetimi

- Çevik Yönetim küçük artırımlarla iteratif geliştirmeye ağırlık verirken, DevOps test ve devreye alma otomasyonuna odaklanır. DevOps kültürel değişimden başlayarak, otomasyon araçlarının kullanılmasını gerektirir.
- DevOps'un farklı çeşitleri (örneğin DevSecOps gibi) bulunur ve yazılım geliştirme süreçlerinin en erken safhalarında uygulanan yöntemlerin değiştirilmesiyle ilgidir.
- Çevik Yönetim, yazılım geliştiriciler için planlanan işlerin yapılması için bir yapı ortaya koyarken, DevOps, operasyon süreçlerinde genellikle rastlanan planlanmamış işleri ele alır.

DevOps

- DevOps yazılım geliştirme ve operasyon süreçlerinin her bir aşamasına sirayet etmekte ve çevik proje yönetim metodolojisinin de bir adım ötesine geçmektedir.
- **Diğer deyişle, DevOps, örgütsel değişime çevik bir yaklaşımdır.**
- Zaman içinde, yazılım geliştiren ekiple operasyon ekiplerinin beraber çalışmaya başlaması, yazılım ekiplerinin kodlarına daha fazla güvenmesine sebep olmuştur.
- Çünkü, geliştirilen yazılımların başarısı sadece yazılımı geliştirenlerin kodlarını nasıl yazdıklarına bağlı değildir. Başarı, kurulan sistemin performansına bağlıdır.
- Örneğin, çalışan yazılımın mikroservis mimarisiyle yazıldığını ve Kubernetes ortamında çalıştığını düşünün. Sistemin performansı, güvenliği, tepki süresi, sadece yazılımın performansına değil, üzerinde çalıştığı konteynerlere, Kubernetes düğümlerine, konteynerleri birbirine bağlayan sanal ağlara bağlıdır.

DevOps yaklaşımı nasıl çalışmalı?

- Yazılım geliştirme ve operasyon ekiplerini bir araya getirmek ve birbirleriyle işbirliği içinde çalışmalarını sağlamak, DevOps sürecinin belki de ilk aşamasıdır. Gerekli olan adımlar şunlardır:
 - Takımlar arasında işbirliği
 - Otomasyon araçları kullanımı
 - Sürekli entegrasyon
 - Sürekli test
 - Sürekli yazılım teslimi
 - Sürekli izleme

DevOps

- Özetle, DevOps, yazılım geliştirme ve operasyon işlerini, yüksek performanslı bir sistem geliştirme hedefi doğrultusunda bir araya getirir.
- DevOps'un kuruluşlara faydaları şunlardır:
 - Yazılım ve operasyon ekiplerinin, tek bir hedef doğrultusunda çalışması sağlanır ve ekipler arasında işbirliği ve güven ortamı yaratılır.
 - Daha hızlı ürün ve daha kaliteli ürünler çıkarılır.
 - Çalışan yazılımlarla ilgili yazılım ve operasyon ekiplerinden sorunun her yönüyle ilgili geri besleme alınır ve hızla değerlendirilerek kod üzerinde değişiklik veya sistemde güncelleme yapılır.
 - Önceden planlanmamış işler veya çözülmesi gereken sorunlar daha hızlı giderilir.

DevOps Kültürü

- DevOps, yazılım firmalarında sadece iş yapış tarzlarında değişiklik demek değildir ve kültürel bir değişiklik anlamına gelir.
- Yazılım geliştirenler ve operasyondan sorumlu personel artık tek bir hedef doğrultusunda, **yani en kısa sürede yeni bir sürüm çıkarmak amacıyla** beraber çalışmaktadır.
- Her adımın otomasyonla yürütülmesi DevOps kültüründe önemlidir. **DevOps mühendisi, tekrar eden her süreci otomasyonla çözebilmelidir.**
- Önceki dönemlerde, sistem yöneticileri veya veritabanı yöneticileri, yazılım geliştirme süreçlerine talep edildiği zaman destek verirken, artık ürünlerin geliştirilmesi için fiili olarak çalışmaktadır.
- DevOps kültürünün temelinde saydamlık, iletişim (özellikle takımlar arası) ve ortak çalışma bulunmaktadır. Paylaşımçı bir sorumluluk anlayışı bulunur.
- DevOps kültürünün yerleştirilmesi, genellikle örgütsel ve kurumsal anlayışta büyük değişikliklerin yapılmasını gerektirir. Kurumlarda bu tür değişimler her zaman zorluklarla başılır.

<https://www.atlassian.com/devops/what-is-devops/devops-culture>

Şirketler DevOps için ne yapmalı?

- Yapılan bir ankete göre, önemli yazılım geliştirme firmalarının üçte ikisi DevOps pratiklerini yoğun bir şekilde süreçlerine entegre etmiştir.
- Yazılım firmaları ne yapmalı?
 - **Çevik proje yönetimine geçilmelidir.** Müşterinin önüne her zaman çalışan bir program konulacak şekilde yazılım geliştirme ve devreye alma süreçlerinin hızlandırılması gerekmektedir. Scrum tarzı bir yönetim felsefesi uygulanabilir.
 - CI/CD yaklaşımının benimsenmesi önemlidir.
 - Kod yazımından devreye almaya kadar bütün süreçlerin, otomasyona geçilmesi için irade gösterilmelidir.
 - **Otomasyon** için doğru araçların tercih edilmesi önemlidir. Öğrenilmesi zor araçların kullanılması büyük risk yaratır.
 - Tüm süreçlerin gözlenebilir olması sağlanmalıdır. Otomatik araçlardan elde edilen loglar, veriler, grafikler, hata kodlarının incelenebileceği bir altyapı kurulmalıdır.
 - Değişimin zor olacağı bilinmeli ve top yekûn kurumsal anlayış değişikliğine odaklanmalıdır.

<https://www.atlassian.com/devops/what-is-devops/devops-culture>

<https://cloud.google.com/blog/products/devops-sre/take-the-2022-state-of-devops-survey>





DevOps uygulayan firmalar

- DevOps pratiklerini uygulayan firmalarda, en önemli değişim kurumsal kültürde olmaktadır.
- DevOps'u benimseyen firmalarda, iç dokümantasyonun DevOps kültürünün yerleşmesinde önemli bir unsur olduğu görülmüştür.
- DevOps kullanan firmalarda, bıkkınlık ve işten soğuma anlamına gelen «**burnout**» sürecinin daha az yaşandığını göstermiştir.
- Amerikan Patent Ofisi, DevOps süreçlerini uygulayan kurumlardan biridir. Her hafta yazılımlarında yapılan değişiklikler sonucu, sistemlerinde her hafta 1000 tane otomatik derleme yapıldığı bildirilmiştir.
- Bulut ve konteyner teknolojilerinin DevOps pratiklerinin benimsenmesi için önemli bir itici güç oluşturduğu bilinmektedir.

DevOps nasıl uygulanmalı

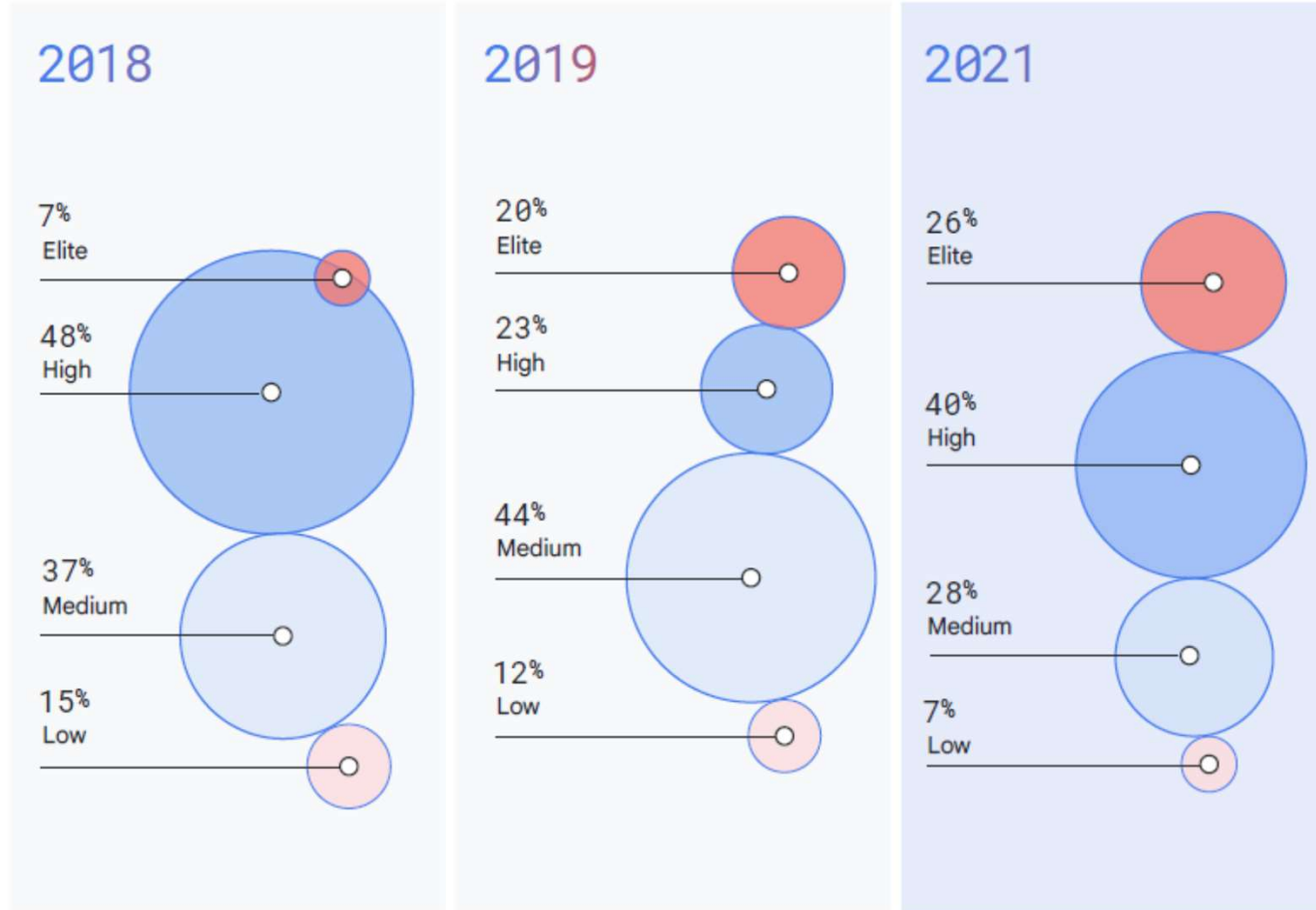
- DevOps'un tam anlamıyla uygulanması için ne tür sonuçlar elde edilmelidir:
 - Devreye alma (**deployment**) ekiplerden talep geldiğinde yapılabilir olmalıdır. Günde birden fazla devreye alma olabilir.
 - Kodda yapılan değişikliğin ardından, derleme, entegrasyon, test, vb bir saat içinde «**prod**» ortamı oluşturulabilmelidir. Bu hedefi gerçekleştirebilmek için her süreç otomasyon araçlarıyla tamamlanabilmelidir.
 - Yeni teslim edilen ve devreye alınan bir yazılım sürümünde arıza çıkması durumunda, en fazla bir saat içinde eski sürümün devreye alınmalıdır.
 - Yeni sürümde hata çıkma olasılığının %15'ten az olması gereklidir.

DevOps uygulama başarısı

Software delivery performance metric	Elite	High	Medium	Low
 Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
 Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
 Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
 Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

<https://cloud.google.com/devops/state-of-devops>

DevOps uygulayan firmalar



DevOps firmalara ne sağlar?

- Atlassian'ın 2020'de yayınladığı raporda, firmaların %85'inin DevOps pratiklerinin tam anlamıyla uygulanması konusunda sorunlar yaşadıklarını bildirmektedir.
- Ancak, firmaların %99'u, DevOps'un kendilerine olumlu etkide bulunduğunu belirtmiştir:
 - %48 firma çalışanı, maaşlarında iyileştirme yapıldığını belirtmiştir.
 - %61 oranında daha kaliteli ürün üretildiği bildirilmektedir.
 - %49'u pazara daha hızlı ürün sürebildiklerini ve dolayısıyla devreye alma sürelerini azalttığını belirtmiştir.
- Google Cloud'un yaptığı araştırmada, DevOps uygulayan firmalarda, eski çalışma tarzına göre şu sonuçlarla karşılaşılmıştır:
 - 973 kat daha fazla devreye alma
 - Devreye almadan önce gerekli olan zamanın 6570 kat düşürülmesi
 - 6570 kat daha hızlı eski sürüme dönüş hızı
 - Üçte bir oranında daha az yazılım kusuruyla karşılaşılmaması

DevOps'un Önündeki Engeller

- Atlassian'ın yaptığı araştırmada, firmaların %85'i DevOps uygulamalarında bariyer niteliğinde bazı sorunlarla karşılaştıklarını bildirmiştir.
- Bu sorunların
 - %37'sinin çalışan elemanların becerilerinin olmaması ve bilgilerinin azlığı,
 - %36'sının kullanılagelen altyapının eksiklikleri,
 - %35'inin kurumsal kültürün yeniliğe uyum sağlamadaki zorluğu olduğu belirtilmiştir.
- Firmaların %74'ü DevOps'un sağladığı faydalarını (özellikle teslimat sıklığının artışı) bir şekilde ölçebildiğini belirtmiştir.
 - Genellikle ölçülen kriter teslimat sıklığıdır. Bu tür bir bakış açısı önemli bir sorun oluşturmaktadır. Yeni sürümden eskisine dönüş hızı, teslimat öncesi harcanan zaman, başarısızlık oranı gibi kriterlerin de kullanılması gerekmektedir.
- %97'si ise DevOps'un bir ölçüde faydalı olduğunu hissettiklerini vurgulamıştır. Ancak, Firma yetkililerinin %58'si DevOps'un faydalarını ölçmenin doğrudan bir yöntemi bulunmadığını anlatmıştır.

DevOps Uygulamaları

- Atlassian'ın yaptığı araştırmada, firmaların %69'unda DevOps takımlarının veya unvanında DevOps mühendisi/sorumlusu geçen çalışanların bulunduğu belirtilmektedir.
- Firmaların %20'sinde DevOps işi yapan elemanların bulunduğu ama DevOps unvanının bulunmadığı bildirilmiştir.
- 500 kişiden fazla kişinin çalıştığı firmaların %57'sinde unvanında DevOps geçen çalışanlar bulunmaktadır.
- 2020 itibariyle firmaların sadece %20'si 5 seneden fazla süredir DevOps pratiklerini uygulanmaktadır.
- DevOps'u en fazla 3 sene önce başlayan firmalarının oranı %39 olarak ölçülmüştür.
- Müşterilerin %36'sı, DevOps uygulamalarından sonra gelirlerinde artış olduğunu belirtmektedir.

DevOps Uygulamaları

TOOLCHAIN USAGE



Atlassian DevOps Trends Survey, 2020

DevOps'un başarısı nasıl ölçülür?

- Firmaların da sık bir şekilde kullandığı kriter «**deployment**» sıklığıdır. Ne kadar hızla müşteriye yazılımın sunulması anlamına gelir.
- Diğer kriter, eski sürüme ortalama geri dönüş zamanıdır. «**Deploy**» edilen yeni sürümde ortaya çıkan bir hata sonrası, eski sürüme ne kadar hızlı dönüleceği de önemli bir kriterdir ve yeni sürümde görülen hatalar çok sık karşılaşılan bir durumdur.
- «**Deployment**» için geçen hazırlık sürecinde, derleme, entegrasyon, test (statik kod, birim testleri, fonksiyonel testler, entegrasyon testleri, vb) ve devreye alma uzun zaman alabilmektedir. Bu sürenin azlığı, DevOps süreçlerinin etkin bir şekilde kullanıldığını doğrudan göstermektedir.
- «**Deployment**» sonrası yaşanan sorunların yüzdesi, yazılım geliştirme veya DevOps süreçlerindeki hataları göstermesi açısından önemlidir. Eğer bu oran %25'ten fazlaysa, DevOps süreçlerinde sorun olduğu kabul edilir.

DevOps Mühendisi ne yapar?

- Yazılım geliştirme ve operasyon süreçlerini bilen ve yazılım geliştirme süreçlerinin tamamına hakim mühendistir.
- Şu konulara vakıf olmalıdır:
 - Kodlama pratikleri (Java, Python, Go, vb)
 - Sistem yöneticiliği (Linux, Docker, Kubernetes, vb)
 - Kod depo (repository) yönetimi, entegrasyon, vb
 - Altyapının yönetimi (Ağ, güvenlik duvarı, WAN, LAN)
 - Sistem güvenliği
 - DevOps otomasyon araçlarının kullanımı, yönetimi
 - CI/CD yönetimi
 - Çevik proje yönetimi
 - Kod kalite güvencesi sistemleri
 - Otomatik test sistemleri

<https://www.atlassian.com/devops/what-is-devops/devops-engineer>

DevOps Mühendisinin Yol Haritası

- Temel DevOps kavramları (DevOps, DevSecOps, CI/CD, vb)
- Bilgisayar ağları, ağ cihazları, protokoller ve ağ güvenliği
- Linux
- Kabuk programlama (Linux'te Bash, Windows'ta powershell)
- Python (hızlı kod yazmaya yönelik herhangi bir programlama dili olabilir)
- Web sunucusu (NGINX, Apache, vb)
- Web framework (Node.js, Flask, vb)
- Sanallaştırma (temel kavramlar, sanallaştırma türleri, konteyner temelli teknolojiler)
- Docker
- AWS/Azure/GCP
- Kubernetes
- Git
- Jenkins
- Ansible
- Prometheus
- Grafana
- Terraform

<https://roadmap.sh/pdfs/roadmaps/devops.pdf>