Önceki dersin özetini yap

Akıllı robot-----biz tasarladık , biz yaptık, yaptık ettik, kanepenin altına girmezse, parkede oluyor, plaka da kayıyorsa (sürekli patinaj çekiyorsa) ya da halıflex de olmuyorsa yine bir anlamı kalmadı.

Dolayısıyla benim develop ettiğim şey, ürün, opereşinsa veya müşteriye geçtiğinde bir anda çöp hükmünde olabilir. Bu aynısı yazılım için de geçerli.

Ne demeştik hepinizden rikuayrımınslar almıştık, her birinizin rikuayrımıntsları farklıydı. Ve başkalarınının rikuayrımıntslarını duyduğunuzda, zaman geçtiğinde bir anda aklınıza yeni bir rikuayrımınts gelecek ve diyeceksiniz ki ben bunu şöyle istiyordum da aslında şunu da eklemeliyim demeliyim. Kütüphanede çocuklara ödül kısmı yoktu mesela bunu duyunca yaa evet bunu da yapmalıyım diyebiliyoruz, araştırıp bakıp daha güzel bir özellik eklemek isteyebiliriz.

Tabi bunu ileriki aşamada verirsek ne olur bize her şeye baştan başlamalıyım(çöp) maliyet %50 artacak gibi bir şey söyledi bize. Şimdi bu tür durumların olmaması lazım, bizim bunları yaşamamamız lazım, Niye yaşamamamız lazım, bir inceleyelim arkadaşlar.

Amazon, facebook, bunlar hep devops teknolojilerini kullanan firmalar. Bunlarda hiç böyle sıkıntılar yaşıyor muyuz, hepimiz hep kullanıyoruz. Bakıyoruz, sürekli bir gelişme var, güncelleme var.

Ama waterfoll da nasıl olacaktık, başlayacaktık, bitecekti yenisine geçecektik. Süreç böyle devam edecekti sonunda ürüne ulaşacaktım ben.

Normalde böyle mi oluyor her gün kullandığımız programlarda , sürekli bir güncelleme geliyor, sürekli yenileniyor. Kullanıcıdan yeniden mi rikuayrımınts alınıyor, yeniden mi projeye çıkılıyor. Böyle olsa facebook un haftada bir yeniden bir proje başlatması lazım ki haftada bir o projeler bitsin ki bana güncelleme gelsin. Ama mevcut çalışan da bir sistem var. Yenilikler nasıl sisteme geliyor. Bugünkü dersimizde bunu göreceğiz ve büyük firmaların niye bu kadar büyük olduğunu ve fark yarattığını da görmüş olacağız.

Bu firmaların haftada bir güncellemeleri geliyorsa bana, bunların her bir güncellemesi için çalışan 10, 20 tane takım mı olmalı. Bir takım gelecek güncellemeyi yapacak bitirecek, diğer takım gelip diğer haftanın güncellemesini mi yapacak. Çünkü bir takım bittiği yerde diğer takım başlamalı , bu beraber ilerliyor. 10, 20, 30 tane takım mı olmalı bunun için çünkü böyle olsa çok maliyetli olması gerekir. Aynı iş için sen 10 katı adam çalıştırıyorsun belki de. Bu firmalar bunu yapmaz, yapmıyorlar ki büyük ve zenginler.

Bir web tasarımı yapalım desem veya kütüphane yazılımımızı herkese ayrı ayrı hazırlayın desem herkes farklı bir şey ortaya çıkaracak. Dolayısıyla ben nihaiyi ürüne giderken bir de bunların koordinasyonunu yapmam lazım. Bu işler daha da karmaşıklaştı, Waterfol tamam eksikleri var ama kendimi geliştirmek istediğimde problemlerle karşılaşıyorum. Şimdi insan sayısını mı arttıracağım, insan sayısını arttırdığımda bu işleri koordine etmem gerekiyor. Birinin yaptığı diğerinden farklı olmamalı.

Veya şöyle bir örnek verelim ; iki kişiye web sitesi tasarımı görevi verdim sizden, web sitesini tasarladınız---index.html---son, yani ertesi gün yada bir saat sonra yeni bir şeyler ekleyeceksiniz bu arada diğer kişi bu dosyaya bakacak son yazıyor, değişiklik yapılmış son hali mi ya da daha değişiklik yapılmadan önceki halimi, veya version vereceksiniz, en son hangisini baz alacak. İki kişinin bile aynı projede yer alması karmaşıklığa sebep oluyor.

Bir kere yaptığımız işte birbirimizden haberdar olmamız lazım, ve bunu yaparken de ürünlerimizin, yaptığımız şeylerin aynı doğrultuda gidiyor olması lazım.

Web sitesi geliştiriyoruz birimiz butonları yaptık, birimiz tasarım ve renklendirmeyi yaptık hiç biri birbiriyle uyumlu değil, dolayısıyla ne oldu boşa gitmiş oldu.

Bunu da geçiyorum biz bunları yaptık bir başka arkadaşta Amazon dan sorumlu, biz daha bu hatalarımızı farketmeden ( birbirimizle uyumsuz olduğumuzu ) o arkadaşta bunu aldı baktı elindeki de çalışıyor yükledi amazon a. Web sayfası canlıya verildi. Hem ortada yanlış giden iki iş var, biri zaten uyumsuz siteydi diğeri de bunun canlıya gitmesi.

Dolayısıyla bu kadar yazılım işleri karmaşıklaştığında işlerin çığırından çıkması çok normal. O zaman biz ne yapmalıyız.

Şimdi sizin burada aktif katılmanızı istiyorum, biz az önce web sitesi yaparken çıkan sıkıntıların üstesinden gelebilmek için nasıl bir sistem , çözüm üretmeliyiz.

Bir kültür oluşturmak, insanların birbirinden izole çalışmak yerine, birbirini etkileyen konularda ortak çalışma kültürüne sahip olması.

Ben bir veri tabanı yapacaksam bu web sitesi yapacak arkadaşı etkiliyor. Burada bir değişiklik yapmadan önce gidim bir konuşayım, koordine edeyim usullere göre, çalışma yöntemlerimize göre bir süreç çalıştırayım ki işler boşa gitmesin. Kendi içinizde sürekli bir iletişim var, sürekli bir etkileşim var, sürekli kelimesi, burada önemli olan bu. Wodırfol da neydi, süreklilik yoktu. Dizayn ekibi yaptı etti kodlamaya geçti arada iletişim bitti. Bir daha bir daha dizayn a gitmiyoruz, o insanları tanımıyoruz bile.

Veya bir yazılım yaparım geliştiririm kim ne zaman ne yapıyor takip ederim. Bu da arkadaşlar nedir? otomeşınıl. Otomeşın yine bizim yoktu wodırfullumuzda. Ama şimdi konuşuldukça, daha iyi bir sistem nasıl olmalı diye sorguladıkça yeni şeyler ortaya çıkıyor.

Peki yazılmış bir program var mıdır, yoksa biz mi yazalım bu yazılımı?

Google bakalım mı, hadi bakalım

Software version control system

Projemizde bir çözüm üretmiş olduk.

Yazılım kontrol ile ilgili daha prosesin en başındayken biraz beyin jimlastiği yaparak neleri bulduk ;

Sürekli bir etkileşim olması gerektiğini, bir kültür olması gerektiğini ve otomatik olması gerektiğini bulduk.

Peki yazılımları ortak halledebilmeyi çözdük , bunun diğer ki aşamalarında test ekibinde bir kişi çalıştığında, ne kadar uzman biri de olsanız bunu ne kadar yapabilirsiniz veya nerelerde sorun çıkabilir.

Örneğin, biriniz bir form yaptınız (web) isim , soyisim, mail adresi şifre kaydet. Kullanıcı tanımlama formu yapmış 4-5 tane alan var orada, siz de bunu test ediyorsunuz.

Ne yapacaksınız şimdi nasıl test edeceğiz ???

(Binlerce kayıt gönderip kitleyebilir)

Giriş yaparken ne gibi hatalar verebilir. (en az 8 karakter, küçük büyük harf, @ işareti, sağ sol yazıları)

(Operasyonel açıdan bakmak, bu test kimin için ??)

Yaş analizi yokmuş, siz test ekibinde yoktunuz ve bu atlanmış oldu.

Yaşı küçük olan birisi geldi kayıt yaptı, üye oldu, daha sonra almaması gereken bir şeyi aldı ne oldu bu bir sorun olacak.

Demek ki testlerin de daha modern daha uygun bir şekilde yapılması lazım. Bir kişiye bağlı olmamalı, testleri de yapanın hata yapabileceğini düşünmeliyiz.

Buna nasıl çözüm bulalım o zaman ????

* Test programları düşünmeliyiz
* Sürekli geri bildirim almalı

Bunları düşününce de biz de bulabiliyoruz değil mi , hiçbir fikir uzaydan gelmemiş, bilgisayarcı düşünce tarzıyla düşününce hepsi ortaya çıkacaktır.

Evet her aşamada geri bildirim alırsam bu iyi olur. Peki her aşamada geri bildirim alırsam bunun altından nasıl kalkarım database, web sitesi, cod ile ilgili geri bildirim, bunları ne yapacak raporlayacak, ilgili kişiye ulaştıracak….

Bu sefer diğer maddeye geldik, diğer çözümle birleştirmemiz gerekiyor. Bu da nedir, burada ben yazılım kullanayım yine. Nasıl yazılımlar kullanayım, bu yazılımı alsın, bu işlemleri otomatikleştirsin.

Bunu da hallettik, daha sonra bunu canlıya verme durumum var.

Bunu tester dan aldık developerlardan aldık kopyasını gidip servera kuracağız, scriptler yazdık, konfigürasyon ayarları yaptık kurduk gittik.

Ne oldu başka biri bir hata oldu geldi baktı, ayarları ilk yapan kişi ile aynısını yapamadı. Kullanıcı izinleri tanımlarken rw veya yürütme yetkisinde hata oldu,

Peki burada ne yaparım ben?????

Devamlılığı sağlamak adına doküman mı hazırlamalı mı ???

otomeşin

Mümkün olduğu kadar insanları entegrasyondan, süreçlerin birbirine geçişlerinden, hatta bazı durumlarda test vs gibi konularda işin kendisinden çıkartabilirsem ( yanı testi yapan kişinin değil, testi yapan kişinin bu toollarla testi yapması demek ) .

Evet o program çalıştığı zaman ne yapsın, insandan bağımsız olarak yapabildiği kadar testini yapsın, sonuçlarını otomatik olarak üretip bunu raporlasın, hatta yeni bir test edilmesi gereken şey geldiğinde otomatik olarak alsın onu da teste soksun. Tester öyle bir test programı yazsın.

Veya diğer arkadaşlarım bana öyle bir diploymınt sistemi kursunlar ki yeni bir kod geldiği zaman sistem otomatik bunu alsın teste göndersin, testen sonucu da alıp kullanıcının sunucusuna yüklesin.

Ben ne yapayım aradaki o configürasyon ayarlarını yazayım bir daha da hiçbir şeyle uğraşmayayım., hatadan izole olsun.

Ve bu bunun bir getirisi de şu olacak arkadaşlar, wodırfol modelinde bir web sitesinin rengini değiştirmek istesem, ya da bir butonun rengini değiştireceğim desem taa en başa dönüp süreci çalıştırıp, tasarım, dizayn, koding tekrar işin içinden geçip belli bir zaman sonra ancak o butonun rengini değiştirebilirim.

Fakat şöyle bir yöntem olmalı, kullanıcı bu butonun rengini gerçekten kırmızı değil mavi istiyorsa ve bende bunu yazılımcı olarak, firma olarak geçerli olarak görüyorsam tasarımı yapan kişi butonu değiştirip save butonuna bastığı zaman versiyon kontrol sistemi onu otomatik alıp işlemeli, daha sonra o buton rengi değişikliği test süreçlerine girmeli, test süreçlerinden geçerse başka bir şeyleri bozmadıysa yaptığımız bu değişiklik otomatik olarak da sunucuya yüklenmeli.

Eskiden web sitelerini açtığımızda şöyle şeyler görüyorduk. Şu an meintinins var, update yapılıyor, 3 gün sonra görüşelim, 3 saat sonra görüşelim diyordu. Bunları görüyor muyuz artık, varsa bile çok az artık. Baktığımızda çoğu şey siz sayfaya geldiğinizde değişmiş oluyor. Bu artık bunların otomatikleşmesinden dolayı ortadan kalmasıyla alakalı. Artık yazılım dünyasında işlerin bu şekilde döndüğünü , artık bu şekilde yapıldığını bize gösteriyor.

Yansılarımıza geçelim

Pre-clas çalıştınız mı

DevOps hakkında ne biliyorsunuz?

İş hayatına atıldığınız zaman hepiniz, muhtemelen bir devops ortamı göreceksiniz. Devops ekibi olacak bir organizasyonun içerisinde. Bu ekip ne iş yapari ilk gittiğinizde devops varmış diyeceğiniz, önceden biliyor olarak gitmeniz size avantaj sağlayacak. Devops u şu aşamada bilmek, bir şeyi ezberlemek değil, şu an bir mantalite bir kültür öğreneceğiz. Devops ekibi ne yapar, devops kültürü nedir, neden bu ihtiyaç ortaya çıkmıştır, sohbet ortamında ezberlemeden, devops ta hangi toolları kullanıyoruz, o kültürü nasıl işletiyoruz, bunlara bakmaya çalışacağız.

Devops un İçeriği

* What is DevOps?
* DevOps Model
* DevOps Phases
* DevOps Processes ----devops un içerisindeki proseste hangi toolları kullanıyoruz
* Advantages
* Disadvantages

DevOps neye odaklanır?

DevOps un ilgilendiği alan ana odak noktası nedir

developerlar ve diğer roller arasında işbirliği sağlamaktır. Ekibin diğer üyeleri arasındaki iş birliğini en üst seviyeye çıkarmayı amaçlıyor. (Birazdan detaylarına gireceğiz.)

DevOps Nedir

SDLC dersinde modelleri gördük,

Devops ta aslında yazılım geliştirme şirketi içindeki bir alana yönelik bir kültür diyebiliriz. Belli rollere yönelik bir kültür bu.

Watırfol a baktık, bu dersimizde devops nedir ona bakacağız, gelecek derslerimizde de agile nedir ona bakacağız.

Devops nedir derken aslında devops ne değildir i de cevaplasak iyi olur. Sosyal medyada çokça duymuşsunuzdur. Yanlış kullananlarda oluyor, tek bir role indirgemememiz lazım, devops encinir diye. Bir tool a da indirgeyemeyiz, ya da herkes bir otomasyon aracı gibi görebilir, bazen bir takıma , ekibe devops ekibi dendiğini de duyabilirsiniz, bunlar yanlış demeyelim ama kullanımı yanlış diyebiliriz.

\*\*

2009 yılında Patrick Debois isimli geliştirici tarafından ortaya atılan bu terim, tasarımdan geliştirme aşamalarına kadar, tüm program geliştirme sürecinde iş birliği yapan bir ekibin çalışmasını tanımlayan bir şemsiye terimdir.

2009 yılında bu Devops işi yazılım alanına konuluyor fakat bir kültür olarak kuruluyor. Altına kişilerle, rollerle, takımlarla, değişik toollarla ya da manteliteyle doldurabiliyorsunuz ama bir kültür olarak ortaya konuluyor. Bunu bilmeniz yeterli. Duyacaksınız yani devops toolları, devops takımları…. Devopsun basite indirgenmesi diyebiliriz bunlara, biz bunu duyunca bunun geneli kapsayan bir şey olduğunu bilmemiz bize artı sağlayacaktır.

DevOps, Yazılım geliştime ekibinin sonuç üretme yeteneğini ve hızını artıran araç ve felsefelerin tamamını kapsar.

Bilgeliğin başlangıcı terimlerin tanımıdır. (Socrates)

Burada gördüğünüz bir sonsuzluk işareti var, planlamadan koda, testten diploymınt ve monitöre kadar bir çok safha devopsun alt maddeleri olarak. Bunları bu dersimizde ayrıntılı bir şekilde göreceğiz.

Devops :

(SDLC) yani yazılım geliştirme yaşam döngüsünü kısaltmayı ve yüksek yazılım kalitesi sağlamayı amaçlar.

DevOps ----- en genel hali ile development ve operations kelimelerinin bir araya getirilmesinden oluşmuştur. (yazılım dünyasında developer ile bunları sisteme yükleyen bu kodları müşteri ile buluşturan ve bu kodların yedi 24 çalışmasını sağlayacak olan operasyon ekipleriyle daha iyi çalışabilmesini hedefleyen bir kültür.

Bunların bir araya getirilmesiyle bir ürün de ortaya çıkıyor, bir amaç var neden bunların bir araya getirilmesi lazım, bunların koordineli bir şekilde çalışıyor olması lazım bunlardan bahsetmiştik, neydi, bu yazılım geliştirme sürecini, daha kısa tutabilmek ve sürekli bir şekilde müşteriye , devamlı updete edilmiş, devamlı güncellenen ürün sağlayabilmeliyiz. Her an güncelleme geliyor şu an ki programlar biliyoruz. Bir program diğer programdaki güncellemeyi görüp bu bende de olsun deyip yeni bir güncelleme istiyor. Bunlar müşteriyi cezbetmek ve elinde tutmak için hızlı bir şekilde yazılımı her daim geliştirmek amaçlanıyor burada. Bir güncelleme olmadığında müşteri diğerine kayabilir diye bunu yapıyor. Kaliteyi arttırmak, devamlı sistemi güncel tutmak, müşteriyi kendinde tutmayı sağlar o nedenle bu continyus delivery, Continuous Integration kelimelerini IT dünyasında çok duyacaksınız.

Devops ta aslında bu takımları, dev ve ops dediğimiz ekibin arasındaki iş birliğini arttırmasının amacı da bu.

Agile yi de göreceğiz daha sonra, agile ın bir tamamlayıcısı diyebiliriz. Devops daha geniş bir alanı kapsıyor diyebiliriz ama agile yi de kapsıyor. Agile daha eski yıllara dayandığı için onun geliştirilmiş hali de diyebiliriz.

Yazılım firmalarında departmanlar vardır. Developer yazılım geliştiriciler bir departmanda ve operations operasyon ekibi farklı departmanda. Bu iki departman arasında devops köprü görevi görür.

Devops kültürü, developer ve opereşins, bu ekibin birbirleriyle, iş birliği kültürüyle çalışması üzerine dayalı bir yaklaşım.

Aslında devoloper ve opereşinil arasında bir iletişimsizlik ve bir duvar var, bir boşluk var, o boşluğu biz devops anlayışıyla doldurmuş oluyoruz.

Şimdi bir örnek vereceğiz bu konuyla ilgili, biz developer ekibi ile opereşin ekibinin gerçek hayatta ne yaptığını anlamamız lazım. Aralarındaki boşluk nedir bu örnekle daha iyi oturacak kafamızda şimdi.

Örnek :

Şimdi bir araba firması düşünün mühendisleri var developer diyebileceğimiz mühendisleri, teknik resimlerle makinaların bu motorlarının yapılarını, bloklarını çiziyorlar, bu motorun altındaki tekerlerle ilgili, arabanın stabilizesiyle ilgili , hızı ayrobiği ve bir çok şeyle ilgili mühendislik hesaplamalarıyla ilgili bu kişiler kağıt üzerinde bir çok hesap yapıyorlar, devamlı geliştirmeye, ürüne devamlı yeni bir özellik ekleme gibi ve en sonda paketlerin de hayata geçirilmemiş, canlıya inmemiş bir ürün çıkarıyorlar.

Test artamında , sanal ortamda muazzam , çok iyi çalışıyordur, değil mi. Yazılım dünyasında bakarsak bu kişiler geliştiriciler arkadaşlar. Bu kişiler müşterilerle buluşurken çekilen çileyi bilmiyorlar. Sadece Teoriler üzerinden ürünlerini üretip bir şekilde ürünü ortaya çıkarıyorlar. Yazılım alanında da bu böyle.

Fakat gerçek dünyaya geçtiğimizde hayatın gerçekleriyle karşılaşıyoruz. Mühendis in ürettiği parçaları birleştirmeye çalışıyor, veya üretim bantlarını düşünün, belki o üretim bandında mühendisin düşünmediği bir çok zorlukla karşılaşılıyor. Ya da ürün ortaya çıktıktan sonra ürünü geliştiren kişi şu kadar saniyede şu kadar km ye çıkacak diyor fakat operasyon ekibi bu ürünün montajını sağlıyor, gerçek hayata bunu uyguluyor, araba ortaya çıkıyor, tam garajdan çıkacak araba beklenen performansı sergilemiyor. Testlere bakılıyor bir problem çıkmamış sonra üretimi yapan kişiye bakılıyor sonra bu kişiler arasında kavgalar başlıyor.

Ne oldu boşa gitti her şey.

Otomeşin önemli dedik, otomeşin da yine bu sistemin içerisinde ayrılmaz bir parçası, ve yaptığınız tüm bu değişiklikler otomatik olarak yapılan bir yaklaşımdan bahsediyoruz. Yani birbirini suçlama başlıyor.

Biz mühendisler olarak böyle düşünmüştük operasyon ekibi yanlış düşünmüş diyor mesela.

Yazılımla ilgili de bu böyle olabiliyor. Ürünün geliştiricisi ile operasyon ekibi ile bir kopukluk olabilir, ürün geliştirilirken bile tasarım ekibi ve kodlayanlar bile görüşmediği zamanlar olabilmiştir. Birbirinden bağımsız bir ekip ama aynı iş üzerinde çalışıyorlar, aynı ürünün başarıya koşmasını bekliyorlar ama aralarında bir koordinasyon yok.

Developerlar da kodu yazar mükemmel derecede çalışıyor vaziyette, testleri yapılır bakarlar problem yok….

Ancak operations takımına gelirsek, bir yazılımın ihtiyaç duyduğu ortamı, alt yapıyı hazırlar ona uygun olarak gerekli serverları açar, takibini yaparlar, farklı sistemlerin bir birleri ile uyum içinde çalışmasını, uygun bir alt yapı kurarak sağlarlar.

Her yenilik yapılan her değişiklik sistemde de farklılığa gidilmesine neden olur. Çünkü bambaşka ihtiyaçlar doğurabilir.

Mesela bi python framework’u olan flask kullanılarak hazırlanmış bir yazılımın alt yapısı oluşturulması ile yine farklı bir python framework’u olan django ile hazırlanmış bir yazılımın alt yapısı farklıdır. Alt yapıdan kastedilen hangi işletim sistemi kullanılacak, ne kadar kaynak tüketiyorsa ona göre cloud’da açılacak makinenin özellikleri ne olacak, hangi database’e ihtiyaç duyacak ona göre data-base için ilave bir makineye daha ihtiyaç olacak mı?

Kendim yaşadım. Django ile hazırlanmış bir yazılımı aws’in elastic beanstalk denen bir tool’u var onu kullanarak ayağa kaldıracağım. Ben flaskta yaparken sorun çıkartmıyordu ama django’da bir türlü yüklenmedi. sonra araştırdım django sqlite kullanılmış, elastic beanstalk mysql veya postgre db’i arıyor bulamadığı için yükleyemiyor. Daha sonra yazılımcı arkadaş ile görüştük postgre kullanılacak şekilde ayarladık.

Bu örnekle böyle düşündüğümüzde arkadaşlar operasyon ekipleriyle geliştiricilerin, developer ekibinin arasında büyük oranda bir koordinasyon olması gerektiğini görüyor bu kişiler, yazılım geliştiriciler. O yüzden de devops kültürü ortaya çıkıyor.

Bu kişiler birbirinden bağımsız kişiler değil, aynı amaca hizmet ediyorlar, aynı ürün üzerinde çalışıyorlar, birbiriyle devamlı iletişim halinde olmalılar. Birbirlerinin dilinden anlamalılar. En azından haftada bir iki gün ya da ayda birkaç defa bir araya gelip iletişim halinde olması gerektiğini düşünüyorlar.

Ve tabi günümüz yazılım şirketlerinin yapısına döndüğümüzde, az önce geliştiricilerden ve operasyonculardan bahsetmiştim, şu an yazılım dünyasına bunu döndürdüğümüz zaman arkadaşlar bakın bu devolopmınt timinde kimler var, kimlerden oluşuyor.

Aranızda duyan var mı bunları ????

Sol tarafta geliştirme aşamasındaki,

* Project Manager
* Software Architects
* Developers
* Testers/QA, etc.
* Database analistleri
* Halkla ilişkiler de aklenebilir

Birde bu developmınt ekibinin çıkardığı ürünü, bunların ürettiği ürünü sahaya sürecek asıl işle ilgilenen kişiler bunlar;

* System Administrators----linux den dolayı alt yapıların yönetimi
* Database Administrators
* Release Engineers---dediğimiz ürünlerin hem test ortamına hem developer ortamına çıkarılması katkı sağlayan kişiler.
* Network Engineers ---bizim sistemlerin birbirleriyle uyuşması için bir alt yapı gerekiyor.
* Security Professionals, etc--- vazgeçilmezimiz diyoruz ama en çok da engeli bunlar çıkarabiliyor, bu olur bu olmaz diye.

Peki biz size bunu niye anlatıyoruz, neden bunları size gösteriyoruz, bunları şu nedenden gösteriyoruz;

Sol taraftaki developmınt ekibinin sorumluluğu devamlı yenilik, devamlı geliştirme ve devamlı değişiklik istemeleri, şirkette bulunuyor olmasının temel amacı bu.

Fakat sağ taraf stabil çalışan program, sistem ister.

Bu ikisi arasında bir duvar var, bir iletişimsizlik var sağ tarafta da bu işin opereşin ekibi duruyor.

Biz ne yapıyoruz, bu duvarı bir daha inşa edilmemek üzere yıkıyoruz arkadaşlar.

Sol taraf değişim odaklı olması, sağ tarafın da bu değişime karşı olması aralarında ir probleme neden oluyordu. Bu duvarı devopsla yıkmamız gerekiyor.

Kayık ve bomba.

DevOps olmayan bir ortamda, developerlar operation ekibiyle çok az veya hiç koordinasyon olmadan uygulamaları geliştirir ve pimi çekilmiş bomba gibi karşı tarafa bırakır.

Operasyon ekibi daha sonra deployment için gerekli ortamı hazırlayıp, ürünü hizmete sunulmaya hazır hale getirecektir. Bu arada ürünün teslim süresinin ya da güncelleme süresinin yaklaştığını düşünün,

Uygulama o ortamda çalışmazsa, her iki ekip de aynı şekilde tekrar kontrollerini testlerini yapar, yeniden dener. Bu esnada ne oldu zaman kaybı, emek israfı. Üzerine bir de stres arttı.

2009 ve 2010 da bu devopsun anlayışının artaya çıkması şundan dolayı, bindikleri geminin aynı gemi olduğunun farkına varmasıyla oluyor.

İşte bu haberleşmenin olmaması, bir koordinasyonun olmamasından dolayı birçok yazılım şirketi batıyor arkadaşlar. İstatistikleri agile dersinde sizde görünce şaşıracaksınız.

Halbuki bu iki tim arasında işbirliği olsa, program geliştirilmeye devam ederken karşılıklı koordine olsa bu tarz sorunlar en aza indirildiği gibi çok daha hızlı ve daha az maliyetle yazılım geliştirilebilir.

İşte devops akımı bu düşünceden hareketle ortaya çıkmıştır.

Bu batmanın önüne geçmek için, ekiplerin birbiriyle barışık halde çalışabilmesi için, güzel bir ortam sağlıyorlar, devops ortamı sağlıyorlar ve bu ortamla da bu iletişimsizliği ortadan kaldırıyoruz.

\*\*

Tabi bu sistem hemen olsun böyle denince olmuyor, bu işin bir manteliteye oturtmamız gerekiyor, bir şekilde mantık çerçevesinde bu kişilerin çalışması lazım.

Bu da nasıl oluyor, iki tarafında birbirine empati yapması sağlanıyor, nasıl devolopır tarafı evet yenilik istiyor fakat bunu sağ tarafta bunu nasıl stabil hale getirebiliriz diye bakmaya başlıyor, opereşin tarafı da biz ürünümüze her daim müşteriye yeniliği nasıl kazandırabiliriz, bunu müşteriyle daha hızlı nasıl buluşturabiliriz diye bir görüş benimsetiliyor. (müşteriye nasıl sunabiliriz, deniliyor)

Her iki departmandaki ekipleri bir araya getirip projenin başından itibaren o proje için tek bir ekip gibi çalışan “cross functional” diye tabir edilen fonksiyonel timler oluşturalım denmiş. Bu sayede karşılıklı işbirliği ve dayanışmayı artırmak hedeflenmiştir.

Birbirlerinin halinden anlar hale geliyorlar, birbirlerini tanıyarak, toplantılar yaparak, aynı ekip, aynı firma için çalışıyoruz, entegre olmak, iletişimin artması, sorunlara birlikte çözüme gitme, firmanın yararına olacak şekilde çalışılıyor. (Her iş ekip işi, yaptığımız ve üstlendiğimiz görev tek olmuş oluyor. )

Aralarındaki iletişimi geliştirmeye çalışılıyor (!!!!—Türkiyedeki firmallar bunu çok önemsemese bile çok yararlı oluyor) ( tabi bu modeli hakkıyla yürüten de var )

İletişim ---- İşbirliği ------ Entegrasyon

* Dev, sistem kararlılığına nasıl yardımcı olabilir? (istikrara, uyuma)
* Operasyonlar, özellik sunumunu hızlandırmaya nasıl yardımcı olabilir? (güncellemeye)

Halbuki bu iki tim arasında işbirliği olsa, program geliştirilmeye devam ederken karşılıklı koordine olsa bu tarz sorunlar en aza indirildiği gibi

çok daha hızlı ve daha az maliyetle yazılım geliştirilebilir.

İşte devops akımı bu düşünceden hareketle ortaya çıkmıştır.

Ops, geliştiricinin iyi kod sağlayacağına güveniyor

Dev, operasyonların kodu hızla ürüne koyacağına güveniyor

Görünürlük “güven ama doğrula”yı mümkün kılar

İki tarafta birbirine güvenecek, yeteneklerine ve yaptıklarına.

Devops enginirs bu işin neresinde derseniz, bir köşeye çekilip ihtiyaç duyulduğunda çağırılacak bir kişi değil devops encinir, ( her departmanda zaman zaman bulunabilir) ürünün planlama aşamasından müşteriyle buluşma aşamasına kadar her safhada küçük de olsa büyük de olsa her zaman bir rolü var arkadaşlar.

Ortaya çıkan sıkıntıları gideren (karşılaşacağı sıkıntıları biliyorsunuz---kod yazanların) ve süreci hızlandıran bir rol oynuyor diyebiliriz. (alt yapının yürütülmesini sağlayan kişi veya takım, gurup diyebiliriz.)

Slayt—19—

* Mucitler ve teknisyenler arasında geri bildirim döngüleri oluşturun
* Dev'in gerçek dünya koşullarında çalışan sistemden öğrenmesini sağlayan operasyonlardan gerçek zamanlı ölçümleri ortaya çıkarın
* Üretim ihtiyaçlarını öngörmek ve erken girdi sağlamak için geliştirmeyi etkinleştiren operasyonlardan gerçek zamanlı metrikler ortaya çıkarın
* İşlevler arası ekipler, tüm altyapı, yazılım kodu ve yapılandırmalar dahil olmak üzere tüm çalışma sistemlerini sunmak için işbirliği yapar---------

Şöyle düşünelim, siz bir programı yapın server a koyun yükleyin, işte geri bildirimler gelmesin olmuyor arkadaşlar. Yani bu şekilde ilerlemiyor işler ( yani artık işlemiyor, eskiden böyle olabilir).

Mesela e-ticaret şirketiniz var, büyük bir şirketsiniz, size şöyle şikayetler geliyor, buton çalışmıyor, satın al da gözüküyor basınca çıkmıyor, değişik istekler geldiğini düşünün, bir anda piyasada alt üst olur. Bu da beceriksiz bir dev-ops ekibinin olduğunu gösterir, biz bu tarz şeyleri önlemek için iletişimi çok güçlü tutmali, her ekibin otomatikleşen programlarla doğru ve düzgün bir şekilde yapılmalı, ve hızlı bir şekilde yapılmalı.

Slayt-20

Sol taraftaki agile kısmının konusu, bir sonraki derste göreceğiz, aslında agile nin amacı nedir şu duvarları yıkmak. Ama biz bu tarafın duvarını yıkacağız, devamlı gelişme isteyen bir ekiple devamlı stabillik ( devamlı çalışan bir sistem isteyen ) bir ekibin duvarını yıkmayı amaçlıyor.

Devops kültürü bunu yıktıktan sonra gördüğünüz gibi devamlı toplantıya dahil olan bir ekip var. Bazen developer ekibi çok büyük bir sorun olarak düşündüğü bir şeyi çok kolay bir şekilde çözüleceğini ifade edebiliyorsunuz ya da tam tersi olabiliyor. Sizin için çok zor olan bir şey deoloper ın çok kısa kodlarla yapabileceği bir iş haline gelebiliyor.

Yani aynı dili konuşmak, aynı gemide olabilmek, aynı masada oturmak, aynı amaca hizmet ediyor olduğunu bilmek size çok büyük avantajlar sağlayacaktır.

Yine devops kültüründe değişim iyidir diyoruz, değişik toolları kullanıyoruz. Şeffafız, bir çok yöntemi var insanları ödüllendirme yöntemi , onları motive etme yöntemi, suçlama yok arkadaşlar, iki ekipte tek bir amaç için tek bir başarı için savaşıyor. birinin sistemi yıkmak gibi yada yavaşlatmak gibi bir amacı söz konusu bile olamaz, iletişimin yüksek olması lazım ( yardımlaşmanın en üst seviyede olması lazım, birinin yapamadığına diğeri yol gösteriyor. )

Bu kültürü bu başlıklar halinde oturtmak, ekibe bunu benimsetmek çok önemli.

Agile yi göreceğiz arkadaşlar, devops un da bunun üstüne eklenmesiyle her şeyin ne kadar hızlı işlediğini göreceksiniz.

Siz bir ürünü çıkaracaksınız ve onu saniyeler içerisinde sörvır a yükleyeceksiniz ve kullanıcıya bunu çok kısa sürede sunmuş olacaksınız. Bu da işte kullanacağınız toollar sayesinde olacak. Doğru tulları seçmek, o tulları etkili bir şekilde kullanmak ve ürünü müşteriye buluşturmak, insan bunu görünce mutlu oluyor gerçekten. Çok kısa sürede müşterinin istediği yeniliği belki 3 gün 5 gün içerisinde ulaştırmış oluyorsunuz. Ve şeyden hiçbir zaman kaçınmayın, bazen zor görünen, sıfırdan yapılan şeyler yapamayacak gibi gözükebilir size, devops ta göreceksiniz.

Siz hiçbir şeyi sıfırdan kurmuyorsunuz, şirketinizde muhtemelen kurulu bir ortama gideceksiniz, siz bu tolların mantığını bilmeniz, nasıl yapılacağını bilmeniz, nasıl işlediğini bilmeniz, sizi zaten 1-0 öne çıkarıyor.

(size kod yaz denmez, size bir iş verilir, googlayın, araştırın, doküman okuyacaksınız, tecrübelerinizi uygulayacaksınız

DEVSECOPS nedir ????

Slayt bilgisi

Bir DevOps ekibindeki herkesin odak noktası güvenlik olduğunda, buna DevSecOps denir. DevSecOps, IT güvenliğine “güvenlikten herkes sorumludur” yaklaşımının bir sonucudur. Bir kuruluşun DevOps düzenine, ilave güvenlik uygulamaları eklemesini içerir. Amaç, güvenliği yazılım geliştirme iş akışının tüm aşamalarına dahil etmektir.

Önceden sanki bu güvenlik başkalarının işiymiş gibi bakılırdı. ( Yani herkes bir başkasının işiymiş gibi düşünülürmüş.) ve bunun için birkaç kişi seciruty gurup gibi bu şirketlerde çalışırmış. Ve bu kişiler bundan sorumlu olurmuş.

Ama bu şimdi Devsecops la beraber aslında güvenlik açığı, bu güvenliğin her birimizin sorumluluğu olduğunu hatırlatıyor bize. Yani biz her alanda şu sonsuzluk işareti var ya bunun her safhasında güvenliği de düşünmemiz lazım. Güvenlik bunların her yerinde diyoruz. (database deki kişisel verilerin korunması)

(Bunu sağlamak için Bulut sistemlerinde aws xray, Guarduty, inspector gibi araçlar vardır. Güvenlik açıklarını kontrol etmek için araçlar bunlar.

İkinci olarak kod yazarken şifrelerinizi doğrudan koda yazılmaz farklı bir dosya içinde tutulur. Buna ilave olarak oluşturduğunuz bu dosyadaki şifre ve kimlik bilgilerinizi de şifreli olarak yazılması gibi hususları örnek gösterebiliriz.)----

Gerçekten securty kısmı olayı bizim için hayati bir önem arz ediyor.

Slayt 25----

* DevOps, agile yazılım geliştirme modelinden bir evrimdir
* DevOps, Geliştiriciler ve Operasyonlar arasındaki boşluğu ele aldı
* Geliştirme ekibi, uygulamayı uygulama için operasyon ekibine sunacaktır.
* Operasyon ekibi uygulamayı izleyecek ve geliştiricilere ilgili geri bildirimi sağlayacaktır.

Devops u özetleyecek olursak bu iki ekibin aynı birbirleriyle aynı takım içerisinde olduklarını hatırlatmak. Bu kısımdan gelen fidbeklerin müşteriden gelmiş gibi developmınt ekibi tarafından iyi bir şekilde appilication lara yansıtılması ve ekibinde birbirine güven içerisinde çalışmasının sağlanması diyebiliriz.

Devops un safhaları nedir arkadaşlar ???

Devops un Safhaları

Devops olayı sonsuz devam eden bir şey, yani continyus deploymınt dediğimiz şeydir. Bu belli bir mantalite olmasının yanında belli tollara hakimiyet de gerektiriyor. Hangi tolları bilirseniz bu aşamayı geçmiş olursunuz ya da buna hâkim olursunuz.

Devopsun safhalar

* Plan diyoruz
* codun yazılması
* Ardından bu kodların build edilmesi
* Yazdığımız ve built ettiğimiz kodların test aşamasından geçmesi
* Testten sonra release edilmesi ---yani onun sürümünün piyasaya çıkması
* Bu sürümü kim kullanacaksa, daha doğrusu bunu kim talep ettiyse o kişiye ya da o kuruma diploy edilmesi, hizmetine sunulması.
* Daha sonra o ürünün opereşın kısmının yapılması, yani idame edilmesi, devamlı çalışır vaziyette tutulması
* Ve yeni bildirimlerin yapılabilmesi için, ya da sağlıklı bir şekilde bunun operasyonunun sağlanabilmesi için devamlı bunun monite edilmesi. Bunun da ilgili tolları var arkadaşlar.

Plan;

Hatırlarsanız, CT dersinde algoritma yazma aşamaları görmüştük. Psedocod yazdık. Bunlar aslında bu plan safhasının işlemleri.

Burada iş sahipleri diyeceğimiz, piyasada ürün arayan kişiler, kendi işlerini kolaylaştıracak, bilgilerini Cloud üzerinden müşteriye aktaracak bir arayüz programlara ihtiyaç duyuyorlar. Tabi bunu bir kişi alıp ben planlayayım şeklinde olmuyor. Nasıl oluyor, hem müşteri ekibinden biri geliyor hem takım içerisinden (prodact ovner ) yazılım geliştirme işinde de çalışmış bu tarz kişiler toplanıyor, projenin amaçları, ürün nasıl olacak, nasıl bir ürün istiyoruz, fonksiyonları neler olacak, bu ürün bize ne katacak, bunlarla ilgili bir konuşma yapılacak ve bir planlama yapılacak. Daha sonra bunlar ayrılıp kendi ekipleriyle de beraber toplantı yapacak , işte isterler bunlar, bizden böyle bir şey talep ediyorlar, bizden bu ürün çıkacak, bizden şu şu şu fonksiyonları icra edecek, şu bilgileri verecek, şuraya yükleyeceğiz gibi üst seviye bir planla başlamış oluyor.

Prodact owner dediğimiz, ürünün sahibi, Project meneger dediğimiz, proje yöneticisi tarafından bu olayların developer lara verilmesiyle, önlerine bu kişilerin serilmesiyle devam ediyor.

Tabi bunu nasıl yapılıyor arkadaşlar, psidocod ve akış şemasıyla oluyor. Developer başlıyor buna bakarak kodları yazmaya (her şey sıfırdan olmuyor tabiki, patern rokeşin). Bunun için version kontrol sistemlerini de kullanması gerekiyor. (git veya başka bir tool git lab, beanstolk, louncpad)

Ardından build dediğimiz aşamaya geçiliyor.

Kodlar yazıldıktan sonra bunları bir paketleme işlemine tabi tutmamız gerekiyor. Önceden disketlerde ya da CD lerde Office programları gelirdi, yada bir küçük bir mario oyunu gelirdi. Bunlar yazılmış bir kodun paketlenerek size ulaştığı exe dosyaları, zip dosyası ya da hangi programda yükleniyorsa, bir çok farklı uzantılı dosya build edilmiş, değişik program kullanarak kodların entegre haline getirilmiş ve paketlenmiş halidir. Ama artık bizim server larımız var, otomasyon serverlarımız, ne yapıyoruz, diyoruz ki bu kod yazıldıktan sonra sen bu kodu al (belli programlar var) bu programları kullanarak build et diyoruz. Programın uzunluğuna göre bir süre sonra paket bir dosyayla yazılımın anahtar teslimini yapıyor. (bunu yapan otomatik tullar var, jenkis gibi)

Şimdi devops un safhalarında göreceksiniz bu kadar değişik tollar var. Her safhanın farklı tulları var ama bu gözünüzü korkutmasın bunlardan birkaç tanesini bilmeniz yetecek. (birini öğrenince diğerini 3-5 saatte öğrenebiliyorsunuz.) Bu tamamen sizin çalışacağınız şirket ile alakalı şeyler. Ya da hiç build işlemini yapmadan devam eden işlemler var.

Burada da en sık kullanılan tullar var.

Testing aşamasına geliyor build edilen codları test ediyoruz. Bu bir otomeşin program sayesinde oluyor. Selenyum denen programda yapılabiliyor. Bunun için bir kod yazılıyor.

Sizin ürününüzü en ince ayrıntısına kadar inceleniyor, testten geçen ve geçmeyenleri ayıklıyor ve geçmeyenleri tekrar coding kısmına maille gönderiliyor. Coding bölümünde de tüm kodu değil hata veya çalışmayan yer neresiyse onu düzeltip tekrar gönderiyor.

Bazen kodu yazan kendi test programını da yazabiliyor.

İntegred kısmına geldik. Neden entegre ediyoruz, eskiden kodu tek seferde yazıyorlarmış. Tek bir dosyada ya da klasörde yazılıyormuş.

Günümüzde ise bu nasıl yapılıyor, microservis yapısıyla çalışılıyor. Nedir bu microservis yapısı, her bir siteyle alakalı hizmet verecek fonksiyon bir kod altında yazılıyor. Yani bir butona tıklandığında kullanıcı bilgilerinin alındığı sayfaya beni yönlendiriyorsa bu sayfanın yapacağı işlemler bunun gibi her biri ayrı başlık altında yazılıyor. Bu programlar ayrı kişiler tarafından yazılıyor, bunlar ayrı ayrı teste tabi tutuluyor ve daha sonra bunlar birbirine entegre ediliyor., yani bir araya getiriliyor bu microservis yapıları. Bunları conteynırlar sayesinde çalışıyor, bunlar birlikte çalışırken entegre olabiliyorlar mı bunu da jenkis sayesinde yapıyoruz ve test ediliyor bir problem çıkıyor mu diye.

Entegrasyondan sonra steycına çekiyoruz, steycın dediğimizde prodakşın serverlarına benzer serverlar var, server üzerine kuruluyor ve belli bir süre o server üzerine yük veriliyor, sanki müşteri siteye girip çıkıyor gibi her günı8,, onun gibi sanal trafik gönderiliyor server üzerine veya testerlar bunu çalıştırıyor gerekli programları kullanarak.

Gerekli yükü aldı, programlar düzgün bir şekilde çalıştı ondan sonra bu fonkşın testlerinden de geçti (mikroservisler dedik ya, Bir sorun çıkmadı entegre olduysa artık bu program, ürün releyz ediliyor,

Bir sonraki aşamaya geçiyoruz arkadaşlar, deployment safhası, en önemli safhalardan birisi bu, artık entegrasyon olarak oldu, relaize olarak bu yayınlandı,

bu relez kısmı çıktıktan sonra yani ürünümüz artık müşteriyle buluşabilir diyoruz.

Sonrasında ne yapıyoruz, biz devops ekibi olarak deploy safhasına geldiğimizde serverların hazır olması gerekiyor ( donanımı plan aşamasında ayarlanması gerekiyor, bu şartlara uyacak bir alt yapı istiyor developer lar), bu aşamada biz ürün müşteriyle buluşmadan önce bu belirlenen server alt yapısını hazırlıyoruz. Aslında büyük oranda işimiz bu bizim. Network alt yapısı, onun DNS ayarları, normal İP adreslerini insanların anlayacağı DNS isimlerine çevrilmesi, bu tarz işlemleri yapıyoruz ve sonrasında jenkis serverımız veya başka otomasyon serverımız tolu ne varsa onu kullanarak bizim bu alt yapısını hazırladığımız server üzerine buradaki paketlenmiş, kullanıma hazır kodumuzu alıp servera yüklüyoruz. (müşteri hangi server a isterse oraya deploy ediyoruz). Şu an artık kimse kendi server ını işletmiyor, herkes Cloud üzerindeki ya da şirketin hizmet verdiği serverdan hizmet alıyor.

Yani Office programı gibi düşünün önce cd den alıp programı yüklüyorduk şimdi üyelik alıp kullanıcı hesabi bilgileriyle Microsoft 365 e bağlanıp hizmet alabiliyorsunuz, kurulmuş hazır programları kullanabiliyoruz.

Burada da aynı buna benzer bir yöntemle kodumuzu diploy ediyoruz, artık müşteri onu hazır kullanacak şekilde diploy etmiş oluyoruz.

Bu tabi deployment sağlandı işimiz bitti anlamına gelmiyor, daha sonra bu programın çalıştığı serverlar linux serverlar, bu linux serverlar güncelleme isteyen, yeni bir güvenlik açığı çıktığında bir paket yayınlanıyor o paketlerin güncellenmesi lazım ve bu serverlarda birden fazla linux serverlar çalışıyor, bu serverların her birinin aynı zamanda update edilmesi lazım, onunda otomasyon tulları var ansible gibi, gönderdiğiniz komutlar 20 tane server ı aynı anda update edebiliyor. Veya iş yerine devops ekibinden biri katıldı diyelim, ne yapacağız peki bütün serverlara tek tek açıp kullanıcı yetkisi mi vereceğim ? ne yapıyorum ansible ile bu işlemi yapabiliyorum.

Bu kısım da opereşin kısmıydı.

Bunun yanında biz yeni bir server da açabiliriz, sitelere yoğun taleplerin olduğu günler, (benim tahmin ettiğim yükü aştığı zaman ne yapıyoruz, bunu da otomasyona bağlıyoruz. ( yoksa site çöküyor---eba gibi)

Monitöring, aslında bu opereşinla bağlı bir safha, çünkü serverlarınızı 7-24 monitör etmek zorundasınız, yük durumunu veya güvenlik açığını ( ya da dünyanın farklı yerinden sizi hekleme ile karşılaşabilirsiniz ). Aslında siz Cloud ortamındaki serverlarda çalıştığınız zaman onun koruması altında oluyorsunuz, Microsoft yada amazon un o tecrübesini ufak ücretler karşılığında siz kullanmış oluyorsunuz. Monitöring kısmında server a kaç kişi girdi, kaç tane müşteri vardı, hangi saatler arası yoğunluk oluyor, bir sene sonra bu veriler sizin için değerli hale gelecek. Daha önceden ne olacağını tahmin edebilmiş oluyorsunuz ve ona göre serverlarınızı daha uygun alma imkanınız olabiliyor. ( daha fazla ise daha az almalıyım deme şansınız oluyor. )

Bunları da işte monitörüng safhasında, bazı özel tullar var o tullarla yapıyorsunuz. Oradan gelen bilgileri kullanıyorsunuz, daha sonra sisteminizi daha stabil bir şekilde çalışmasını sağlıyorsunuz.

Özetlemek gerekirse, plan, kod, build, test, entegre, deploy, serverda işletilmesi ve o serverın monitör edilmesi işlemi kalıyordu. Çünkü ne için para kazanıyoruz biz bu serverların sağlıklı bir şekilde çalışabilmesini idame ettirebilmek için kazanıyoruz ve müşteriye daha hızlı daha sağlıklı hizmet verebilmek için yapıyoruz. Özellikle şu son 3 aşamayı yapamazsak sıkıntılar başlayacak demektir.

Sol taraf biraz gecikse bir sorun teşkil etmez belki ama bu tarafın bir dakikalık bir gecikmesi bile bilindik firmaların ismine leke getirebiliyor arkadaşlar.

Devops processes

Burada bir pipelines , bir boru hattı terimi var,

Devops'un amacı, yazılım geliştirirken, birbiri ile uyum içinde sık aralıklarla, versiyonlama, düzeltme ve güncellemeler yaparak SDLC sürecini kısaltmaktır.

Bunu bir boru hattı gibi düşünelim, kod git’te push edildikten sonra otomatik olarak tüm işlemler sırasıyla yapılmaya başlanır. Boru hattından ileri doğru akmaya başlar.

Bu aslında birden fazla görevi, task i bir araya getirip bunları otomasyon haline sokma işlemine diyoruz. Burada bir serverımız var arkadaşlar, jenkisi duyacaksınız, ücretsiz bir yazılım, genelde her firma bu yazılımı kullanır, bu firmalar bu yazılımı alıp kendi serverlarında deployment pipeline nını kendileri işletirler. Tüm bu aşamaları ( az önce size açıkladığım aşamaları) bu server üzerinde job lar halinde, prosesler olarak tanımlarız daha sonra (trigir dediğimiz ) otomatik olarak bu pipeline çalışır, en son şu developmınt tan alınan şu kodu, buradan giren bu kodu şuradan prodakşın a , yani bizim müşteriyle konuşan serverlarımıza ürün olarak yüklenir, otomatik olarak bunu yapar hepsi. Ama bütün safhaları başarılı bir şekilde geçmesi lazım. Bunu bir defa değil, her tetiklendiği zaman bu işlemleri tekrarlar. ( git de branch işlemi gerçekleşince)

Devops özeti

Devop’un her aşamasında süreklilik vardır. Sistem otomatik hale getirildiği için her aşamanın kendi içinde de bu pipeline aslında döngüsel olarak çalışır.

Mesela planlama ile beraber kodlama başladığında yani develpoment süreci başladığında Contiunous development başlıyor.

Devops süreci ne sağlar?:

Tüm geliştiriciler için Standartlaştırılmış bir alt yapı

(dünyanın öbür ucundan remote çalışan bir yazılımcı için bile)

Her yeni güncellemenin başarısızlık oranını azaltır

Yazılım teslim süresini kısalttığı gibi güncellemeler arasındaki süreyi de kısaltır

(eskiden bir yazılıma yılda bir güncelleme gelirken şimdi her ay güncelleme gelebiliyor)

44- Bu slayttaki terimleri iyi bilmeniz lazım, hangi alanda olursanız olun, yazılım geliştirmenin ana terimleri oluyor bunlar.

* Continious integration--- developer in kodunu github a yükledikten sonra bunun unic testleri ve entegrasyon testleri yapıldıktan sonra çıkan ürün safhasına kadar olan kısıma biz
* Continious Delivery--- ürünler, prodakşına benzer serverlar üzerine yüklenip müşterinin kabul testlerine (biz testi yaptık ama bir de müşteri görecek ) bu testlerden geçip yeni bir ürün çıktığı kısma kadar olan yere ……diyoruz. Müşteriye devamlı bir ürün teslimi.
* Continious deployment---- en son müşterinin kabul ettiği her şey çok güzel çalışıyor, biz buna hazırız, hemen kendi sistemimize yükleyebiliriz dediği yer oluyor.

Bu aşamalardan geçerken hangi tulları kullanıyoruz , hangilerinden yardım alıyoruz.

Jacoco code coverage

Graphana dashboard

\*\*

Devops sürecinin amacı, yazılım geliştirirken, birbiri ile uyum içinde sık aralıklarla, versiyonlama, düzeltme ve güncellemeler yaparak SDLC sürecini kısaltmaktır.

Avantajları

Süreç kısalıyor

Zor işlemi kolaylaştırmış oluyoruz

Müşteri memnunuyeti- şirket başarısı

Güncellemeleri anında halledebiliyor

en büyük avantajı zaman konusunda , süreci çok daha hızlı hale getirir

karmaşıklığı ve çatışmaları azaltır

karşılıklı işbirliği sağlar

pazarda daha hızlı olmanız nedeniyle öne çıkmanızı sağlar

Dezavantajları

Maddi anlamda biraz zorlayabilir

Gerekli insan gücü gerekiyor, yetenek eksikliği

Bu bir kültür demiştik, karşılıklı saygı sevgi işbirliği yoksa devops tool’larını kullanmanız birşey ifade etmez. Bu kültürü oturtmak ise zor ve zaman alıcıdır.

Yeterli donanım bilgi ve bec eriye sahip eleman sayısı sektörde az sizleri bekliyoruz arkadaşlar. Kendinizi iyi yetiştirin

Bütçeden kasıt, devopscu maaşları ABD’ de 1000-1500bin dolar, gayet güzel bi mebla. Firma için biraz sıkıntı tabi.

KAHOOT

1. DevOps için aşağıdaki ifadelerden hangisi doğru DEĞİLDİR?
2. DevOps süreçlerinin hangi aşaması yazılımın planlanmasını ve kodlanmasını içerir?
3. DevOps, geliştirme, kalite güvencesi ve operasyonları tek bir sürekli süreç setinde birleştirmeyi amaçlar.
4. Hangisi DevOps'un bir özelliği DEĞİLDİR?
5. DevOps şu anlama gelir (bir veya daha fazla seçin)…
6. Devops bir nedir
7. Devops un amacı nedir
8. Hangisi bir DevOps Aracı değildir?

Sonuç;

Devops ta ne olduğunu, wadırfol da ne olmadığını az önce görmüş olduk.

Devops un bir kültür olduğunu, aynı zamanda bunun bir anlayış olduğunu, aynı zamanda bunun uygulamalar olduğunu ve bu uygulamaları hayata geçireceğiniz araçlar, tool lar olduğunu gördük.

Ne işe yarıyordu bunların hepsi, bir organizasyonun istenen yazılımı daha hızlı, daha kaliteli ve gelişime açık bir şekilde hatalardan daha elimine edilmiş bir şekilde bize üretme imkanı sağlayan bir yöntem olduğunu gördük.

Peki bu wodırful da olmayan continius ( sürekli ) kelimesinin olmadığını gördük. Aynı zamanda entegreşin da yoktu, yani her şey olup bitiyordu tek seferde, fakat biz nasıl bir entegrasyon tespit ettik, test sonuçlarında veya kullanıcının bulduğu sonuçlarda bir sıkıntı varsa otomatik olarak bunlar geri beslemeli dedik değil mi, aradaki entegrasyonu ürün ve sonuç, test ve geliştirmeyle entegrasyon içerisinde çalışması gerektiğini gördük.

Yine wodırfol da ne eksikti arkadaşlar, otomasyon yoktu. İşlemler sırayla ve manuel olarak yapılıyordu. Her şey bir insan üzerine yüklenmiş ve o şekilde çalışıyordu. Fakat biz neyi gördük otomatik olması gerektiğini, entegrasyon kontrolünün otomatik olması gerektiğini, testlerin otomatik yapılması gerektiğini, diploymıntın otomatik yapılması gerektiğini gördük ve tüm bunların sürekli yapılması gerektiğini gördük.

Aynı zamanda bizim klasik yaklaşımımızda yani wodırfol modelde, müşteri ürünü görmek için bekleyecek , projenin bitiş süresini bekleyecek. O tarih nasıl belirlendiyse o tarihe kadar ürünü görmeyecek. Fakat Devops ta ürünü sürekli olarak görebiliyoruz.

Yani şunu demek istiyoruz, kullanıcı her aşamada her şeye müdahale edebilir.