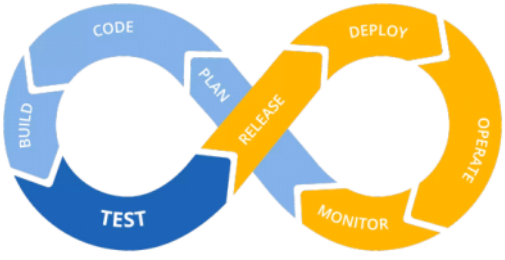
CodePipeline/CodeCommit/CodeBuild

Friday, September 6, 2024 1:18 PM

AWS CI/CD Pipeline

Table of Contents

What is DevOps?



DevOps is a software development approach which involves:

- continuous development,
- continuous testing,
- continuous integration,
- continuous deployment
- continuous monitoring

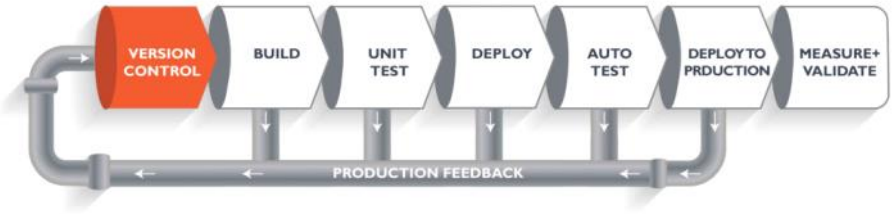
of the software throughout its development lifecycle

Software development cyclein cesitli tolarla ve dongu icerisinde tamamlanma surecine devops diyoruz.
Burada her safhanin kendine ait toollari ve yontemleri var.
Aws bu toolarin hepsini bir araya getiriyor. Buna da CI/CD diyoruz tum toolari tek bir merkezde toplayip yonetme isi.
Kucuk olcekli sirketler bunu daha cok tercih ediyor daha basit, daha hizli daha uygulanabilir daha uyum prblemi yasamayacagi birsey oldugu icin.
Temel basamaklar:

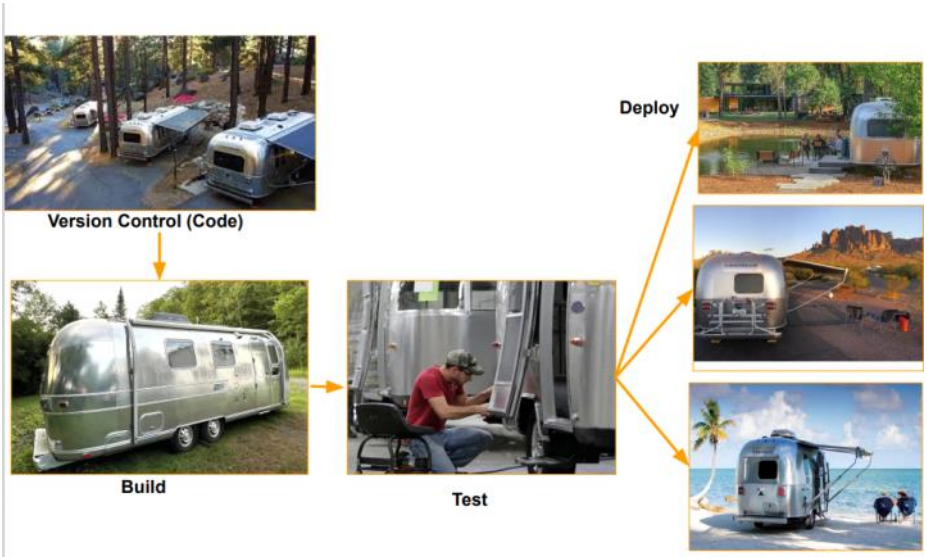
- Kodu aliyoruz
- Kodu build ediyoruz.
- Daha sonra test ediyoru.
- Test ettikten sonra denemeye cikiyor.
- Testten sonra deploy ediyoruz yani bir yerde calistiriyoruz.
- Operate kisminda guncellemelerini yapip varsa configuration yapip
- Monitor ediyoruz yani bir sikinti var mi vs

Bu aslinda iki haftalik bir surec.
Bu surecin her safhasinda biz yokuz ama sureci biz kuruyoruz.

CI/CD



Bu aslinda bir boru hatti gibi.
Version kontrolle basliyor. Yani birden fazla developerin bir programin belirli parcalarini bagimsiz olarak dagitmasi ve pushlamasi. Mesela githubta 10 sayfalik bir proje var. ilk 2 sayfasini biri digeri iki sayfasini biri vs aliyor. Bir branch aciliyor. Herkes kendi branchinda calismalarini yapiyor. Isini tamamlayanlar test kismina o calismalarini gonderiyor. Ama diger calisanlar ilk pushayana gore onun gonderdigini pull edip tekrar duzenlemeye aliyorlar boylece devamli olarak bir pullama sureci basliyor. Versionlama ile entegrasyon saglanmis oluyor. Cunku birlikte calisiyoruz ve uyum ve entegrasyonun saglanmasi gere.
Sonra bunlar build ediliyor.
Daha sonra unit testte kod parcaciklarinin isini yapip yapmadigini kontrol ediyor.
Daha sonra deploy ediyoruz.
Daha sonra otomatik testler yapiliyor.
Daha sonra deploy ediyoruz. Burada functional testler de devreye giriyor.
Son olarakta measure yapip olcup bicip bi sikinti varmi diye kontrol edip isletmeyi en basa donduruyoruz.

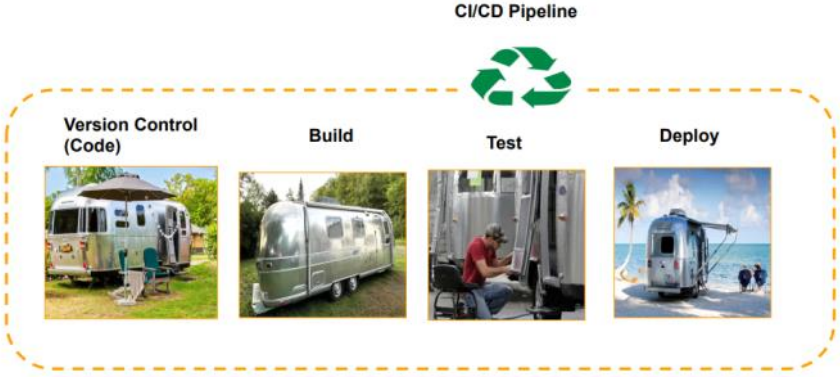


Build dedigimiz sey butun karavan hazirliginin tamamlanmis olmasi, calisabilecek halde olmasi. Yani version kisminda kurdugun mangal hamak vs lerin birlestiriliyor olmasi.

Brunchlar ise su sekilde: biz calistiklarimizi main brauchina push yapamiyoruz. Bizim calistigimiz brunch dev brunchina bagli bir feature brunch.
MAIN BRUNCH
DEV BRUNCH
FEATURE BRUNCH
Seklinde. Yani featurdakiler oldu dedikleri calismalarini dev bruncha gonderiyor. Dev de hepsi olunca merge edip maine gonderiyoruz.

Daha sonra test asamasi oluyor. Karavan calisiyor mu parcalari saglam mi vs. Daha sonra deploy ediyoruz. Yani bu calistirmak demek. Ister dagda ister bayirda karavanini calistir demek. Yani ister lambda uzerinde deploy et, ister on premiesete, ister kubernette vs vs
S3 te bir deploy sekli. Run etmek kosmak tabiri bunun icin gecerli.

CI/CD

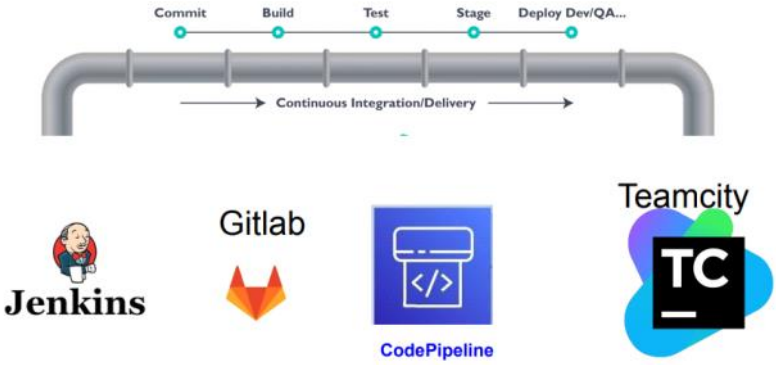


Bu seruvenin tamina da CI/CD PIPELINE diyoruz. Bu cyclenin surdurulebilirligine pipeline diyoruz ve bu pipelinei biz kuruyoruz.
Burada sitemi ben kuruyorum ama full stackciler isi isletiyor.
Build kismi bize ait, test kismi testerlara ait ama test enviromentinin bu pipelinein icine sokulmasini devopscu yapiyor.
Deploy sureci de yine bana ait. Surecin tamami bana ait. Sahne bana it ama sahnedeki sorumluluklar kime aitse o yapiyor.

3 CI/CD Pipeline Stages

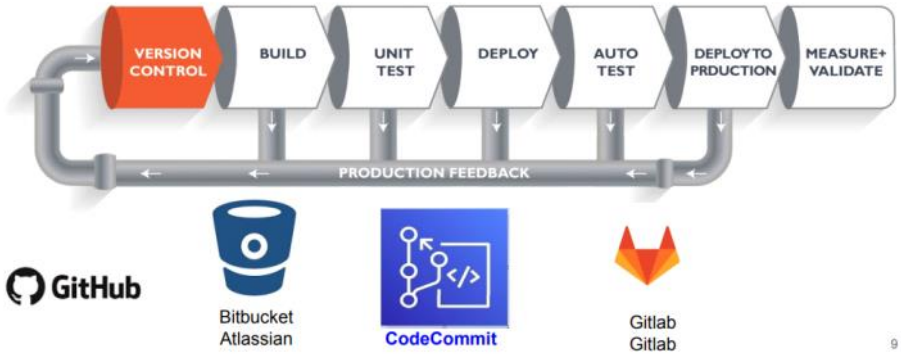
CI/CD Pipeline

Piyasada bu pipelinea karsilik gelen toollar var. bunlarin arasinda en populer tool jenkins.
Biz codepipeline ile yapiyor olacagiz hands-on kisminda.



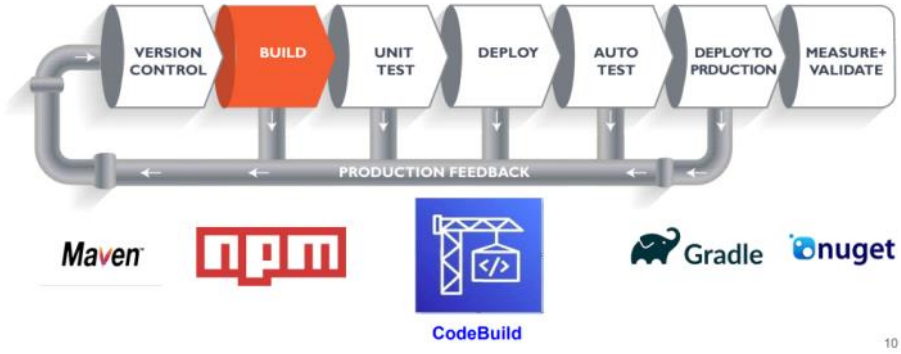
Version Control Stage

Version kontrol github, bitbucket, codecommit, gitlabla entegre edilerek calisir. Biz githubi kullanacagi z kendimiz hands-on yaparken.
Btbucketta ayni github gibi once git add ., sonra git commit -m sonra git push seklinde kullaniyoruz. Bunun icin ayri bir komut girmiyoruz ayni seyler burada da var.
Burdan hanigisini kullanmak istersek isteyelim sistem ayni calisiyor ve dahi komutlar.
Yani bi sirkete girsek ve oradakiler bitbucket kullaniyor olsalar bunu bilmiyorum diye dusunup korkmaya gerek yok bunu da biliyorsun aslinda.



Build Stage

Build stage tum dependencyleri hazir hale getirme yeri.
Eski zamanlarad masaustundeki bir fotografi baska yerde calsitirmak istediginde calismamasi gib mesela. Bunun icin once resmi build etmen yani baska yerde de calisabilir hale getirmen gerekiyor ki bir powerpointte kullanmak istediginde orada da calissin. Maven cok meshur ve biz bunu kullanacagiz. Burada yok ama jenkinste cok meshur ve jenkins her yerde var.



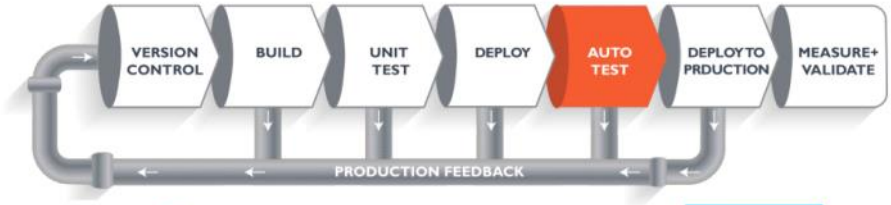
Deploy Stage

Deploy stage nerde nasil calisacagiyla alakali.
Bazi toollar iki stagete de yer alabiliyor. Mesela ansiblei hem operatee hem deploya sokabiliyorsunuz. Biz codedeploy ile bunu yapiyor olacagiz. Ama ilerde bunun asil baba sekli olan kubernetesi goreceksiniz.

Deploy etmek demek calistirmak demek.
Elimizde bir kod var gidiyorum onu conteynir uzerinde, ec2 uzerinde, fargate uzerinde calistiriyorum gibi.

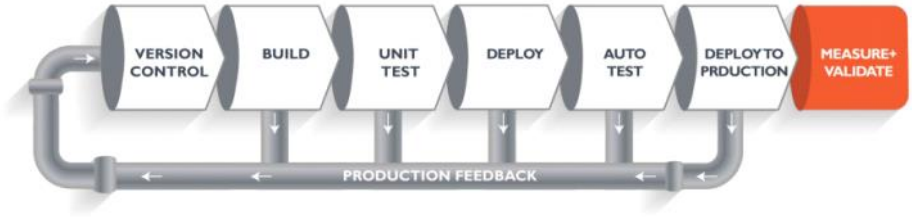
Auto Test Stage

User Acceptance Tests
Functional Tests
Load Tests
Security Tests



Functional testleri yapan en yaygin selenium. Functional test demek mesela bi site yaptiginda ona tikladiginda aciliyormu, suraya + koydugunda o sayfa aciliyormu vs gibi test etmek.
Biz buradan seleniumu gorecegiz.
Bu bizim isimiz degil ama surecin icerisinde kontrol etmemiz gereken bir durum.
Bizim yapacagimiz is mesela jenkins uzerine selenium elentilerini yukleyip testi yapan adamlardan alip o kisiyle birlikte calisiyor mu diye test etmek.
Devops gordugunuz gibi sahaya surulmesine kadar bir sekilde bizim tezgahiizda geciyor, bir sekilde dahil oluyoruz.

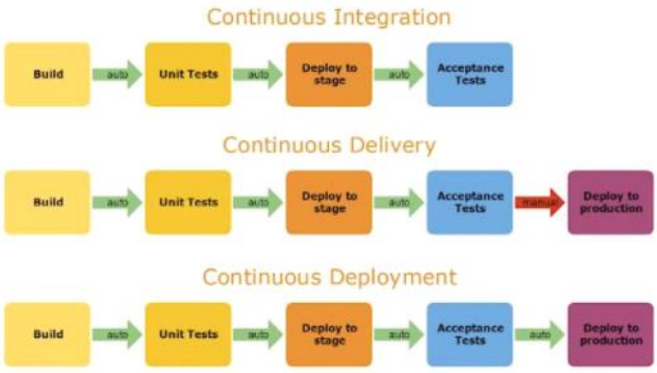
Measure + Validate



13

Bunlari yaptik evet ama bir de feedback almamiz gerek.
Grafana ve prometheus en meshurlari. Bunlari cloudwatch gibi dusunebiliriz.
Loglari alip metricleri inceliyor. Bunlari gostermek icinde promethus uzerinden beslenen grafana kullaniyor. Gorsel bir dashboard gibi dusunebiliriz.
Aslinda grafana olmadan da prometheus calisabiliyor. Ama grafanayla daha beste bes bir sekilde oluyor.
Buradan cpulari, ramleri kac tane instance var kac tane conteynir var vs burdan gozlemyebiliyoruz.

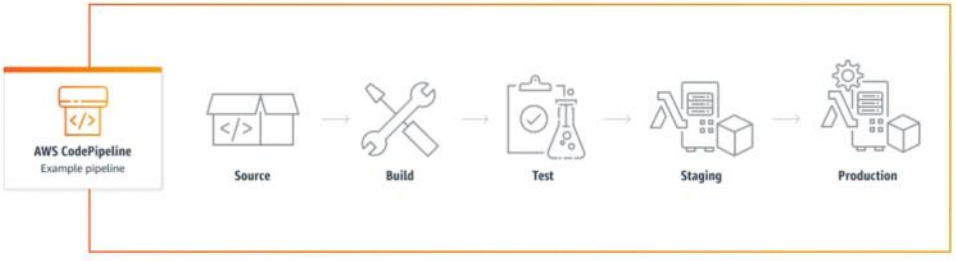
Continuous Delivery vs Continuous Deployment



CI Continuous Integration demek.
CD ise Continuous Delivery ya da Deployment demek.
Integration asamasinda yazilan kodun mevcut programda sorunsuz calistigini gorduk. Burada dev brunchindayiz.
Contiuonus delivery ise bundan sonra yeni kodu manuel olarak devreye sokmak. Yani eski kod ec2 larda calisiyor.
Yani artik sahaya cikmaya hazir oldugunun kanaatine varildigi step. Ve buradan sonra artik dugmeye basip sahaya cikartiyoruz.
Yni eski deploymenti cekiyorum yenisini sahaya suruyoruz.
Continuous deployment ise tamamiyle otomatik. Ben sadece sistemi kurdum. Yeni kodun sahaya cikip musteriyle bulusmasina dahi entegre olmuyorum. Ben sadece sistemi kurdum. Sistem o kadar otomatik calisiyor ki bir developer yazilim kodunu degistirip pushladiginda otomatik olarak kendisi kubernetese, dockera veya nerede calisiyorsa yeni enviromentinde hayat buluyor, calisiyor ve eskisiyle yer degistiriyor. Ve ben bir anda yeni versiyonunu karsimda buluyorum ve bu asamada hic bir dahilim yok.
Continuous delivery ya da deployment bir gelismislik degil tercih meselesi. Otomatik yapmak isteyen deployment tercih eder, konsol sevdalilari konsoldan yapar, bunlar sirketlerin tercihi olarak secilir.
BU GENELDE INTERVIEWLERDE CIKAR!!!! TERCIH TAMAMEN SIRKET POLITIKASINA BAGLI. BUNUN NE ARTISI NE EKSISI VARDIR DEMEK LAZIM.

CI/CD Pipeline with AWS

AWS CodePipeline



AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.

AWS CodeCommit

AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

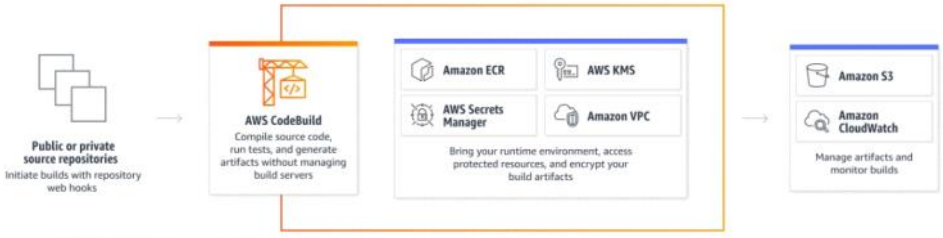
Codecommit aslinda aws in version control opsiyonu git/github gibi. Ama artik cok fazla kullanilmiyor. Daha dogrusu spesifik seyler kullaniliyor.
Git ile entegrasyonu var ve biz git uzerinden yapacagiz.
Codecommitte tuttugunuz dosyalar mutlak surette private oluyor.

AWS CodeCommit

- It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.
- It hosts private repos
- CodeCommit repositories are encrypted at rest as well as in transit.
- Support pull requests where users can review and comment on each other's code changes before merging them to branches
- CodeCommit has no limit on the size of your repositories
- You can migrate to CodeCommit from any Git-based repository.

Pull requestleri karsilar. Yani developer bir kodu push yapti. Onun bir ustu bunu inceleyip request yapiyor. Yani nceleyip onaylayip bruncha dhil olmasini sagliyor. Bir developer bir kodu yazip gonderince hemen entegre olmuyor. Hemen feature brunchinden dev brunchina gecmiyor. Bastaki kisi kontrol edip dogrulugunu onaylayinca entegre oluyor.
Git-based oldugu icin gitte kullandigimiz tum komutlari burada da kullanabiliyoruz.

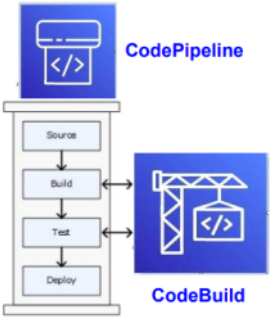
AWS CodeBuild



- compiles your source code,
- runs unit tests,
- produces artifacts that are ready to deploy.

Build safhasind butun herseyi bir aray getiriyor, compile ediyor.
Ayni zamanda burada unit testleri de yapabiliyoruz. Sonucta burada artifact dedigimiz ara mamuller cikabiliyor.
Ayni zamanda secret managerla, vpc ile, encrypte servisiyle, konteynirla ortak calisabiliyor.
En cok kullanilan jenkins veya gitlab.
Illa build degil bazen deploylala birlestirilebiliyor.
Codebuild aslinda bir servis ama arkada manage iki tane ec2 calistiriyor. Ve bu ec2lara biz build specfile dedigimiz bir dosya yukluyoruz ve bunlar bizim yerimize islemleri yapiyor.
Codebuild aslinda bir devopscu. Biz nasil userdatayi giriyoruz o da bizim yerimize bizim belirledigimiz seyleri yapiyor, kopyaliyor, dosyalari aliyor vs. Mesela jenkinste arka planda bir virtual machine calisir. Ben userdatya nasil komut giriyorsam onun da bir komut girme sistemi var.
Userdatadan daha spesisifik manifesto belgeleri icerir.
Codebuild buildspec file diye gecer. Ozet olarak aws tarafindan arka planda ypnetilen bir makine. Biz gormuyoruz o makineyi. Userdatayi girer gibi komutlari giriyorum. Ve o benim yerime calistiriyor.

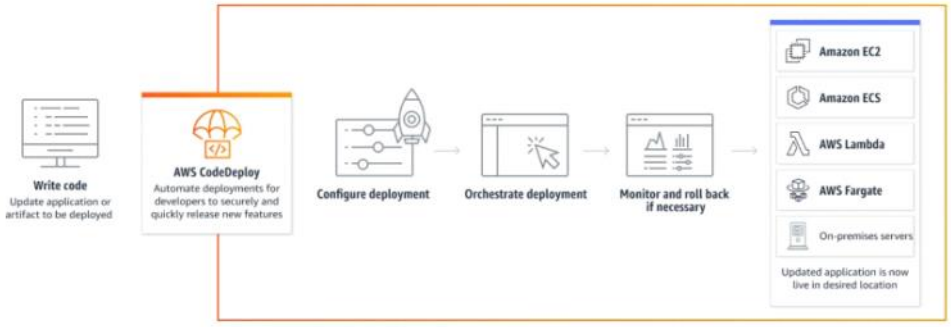
AWS CodeBuild



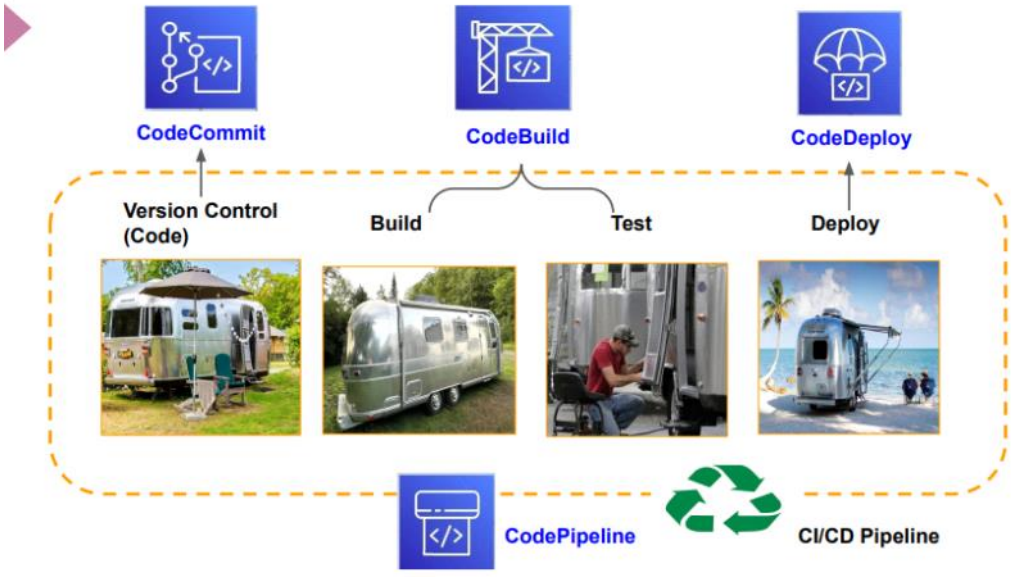
- Fully managed – CodeBuild eliminates the nee to set up, patch, update, and manage your own build servers.
- On demand – CodeBuild scales on demand to meet your build needs. You pay only for the number of build minutes you consume.
- Out of the box – CodeBuild provides preconfigured build environments for the most popular programming languages. All you need to do is point to your build script to start your first build.
- Easily reachable from Codepipeline Console

AWS tarafindan yonetiliyor, fully managed.
Build ettigniz kadar oduyorsunuz.

AWS Deploy



AWS CodeDeploy is a fully managed deployment service that automates software deployments to various compute services, such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Container Service (ECS), AWS Lambda, and your on-premises servers.



Codedeploy diger servisler icerisinde bunu calistirabilmeyi sagliyor. Ec2 kosabiliyorsun, lambda fargate uzerinde, on-premises konteynir servisler uzerinde.
Bir nihai urunum var hazir, bunu ec2 uzerinde kosturuyorum mesela. Sadece bunlar degil s3 uzerinde de yapabiliyoruz.