DevOps ve DevSecOps

Murat Ertik
221307065
Bilişim Sistemleri Mühendisliği
dmuratertik1@gmail.com

***Abstract*—In this document, we are talking about a concept that has been popular with companies since the 2010s: Devops. What is DevOps? What is DevSecOps? What is the major of these concept and what is task in companies? We are talking about DevOps Life Circle and What is the software and technologies we have to know as very well for working be a DevOps? We are talking about what is that softwares and we give some example projects. We are studying like that questions and we use IEEE format in that document.**

***Index Terms*—DevOps,DevSecops,Software**

I. Giriş

Devops kavramı ilk kez 2007 senesinde isimsel bir şekilde olmasada amacı IT sektöründe duyuldu. 2008 yılında Toronto'da yapılan etkinliklerde ise kelime olarak kullanılmaya başlandı fakat yapılan bu etkinliğin ana amacı DevOps kavramı değil Devops kavramının alt bir dalı olan deployment ile alakalıydı. Bir sene sonra Belçikada yapılan bir etkinlikte ise ilk kez DevOps kavramı hakkında bir etkinlik düzenlendi ve Devops kavramı hayatımıza girmiş oldu. 2010 ve sonraki yıllar için dünyaca ünlü şirketler DevOps kavramı ile ilgili dökümantasyonlar yayınlamaya başladı. Devops kelimesi Yazılım geliştirici ve Operayoncu kelimelerinin ingilizcelerinin birleşiminden gelmektedir. 2010'lu yıllara yaklaşırken DevOps kavramının IT sektörüne girmesinin sebebi yazılım geliştiriciler kodlarını yazıp localde test ettikten sonra kodu canlıya alırlardı Operasyoncu olan ekip ise canlıda kodu test ederken hatalar meydana gelince ya da yazılımcıların dedikleri gibi kod patlayınca yazılım geliştirici ekibinin eksik veya hatalarının olduğunu söylerlerdi fakat developer ekibide kodun localde çalıştığını canlı ortamında operasyon ekibinin hatasının olduğunu söyleyince ortaya bir anlaşmazlık çıkardı. Bu anlaşmazlık şirketlere hem prestij hemde para kaybı olarak geri dönüyordu. Bu anlaşmazlığı DevOps kavramı çözüm getirdi ve 2 ekibin arasındaki ilişkiyi çok iyi sağlamış oldu. Bu şekilde DevOPs kavramı yazılım dünyasında yerini almış oldu.[4]

DevOps kavramı hakkında bilgi vermeden önce IT sektöründe yazılım geliştirici ekibinin ve operasyon ekibinin görevlerinden bahsedelmesi gerekir. IT sektöründe operasyon takımının görevi kullanıcı ve teknik ekipler arasında bir köprü kurmaktır. Sunucu yedekleme ve kurtarma işlemlerinden ve sunucu kurar de sorumludur. Sorunları tanımlar ve bu sorunları çözmek için gereken eylemleri gerçekleştirir. Bir başka önemli görevi ise sistem analizi ve çalışan uygulama neyse onun performansını arttırmak için gerekli adımları atar.Sunucuları,eposta sistemlerini, databaseleri ve güvenlik sistemlerini yönetir. Bu ekipte çalışmak isteyen kişilerde ise olması gereken teknik özellikler Linux işletim sistemi,SQL bilgisi,işletim sistemleri vb. Teknik özelliklere sahip olmaları beklenmektedir. Sosyal özelliklerine değinmek gerekirse iletişim kurma konusunda iyi bir düzeyde olmalı,Yönetim vb özellikler beklenmektedir. Yazılım ekibine bakacak olursak kullanıcının istediği fonksiyon,tasarım vb. istekleri bir programlama dili kullanarak geliştiriler. Geliştirdikleri ürünle ilgili kullanıcıdan başka bir istek gelirse ya da hata verirse çözüm yeteneklerinin gelişmiş olmalıdır. Sahip olmaları gereken teknik özellikler ise gelişmiş bir algoritma bilgisi ve sorun çözme yeteneği olarak söylenebilir.[1][2]

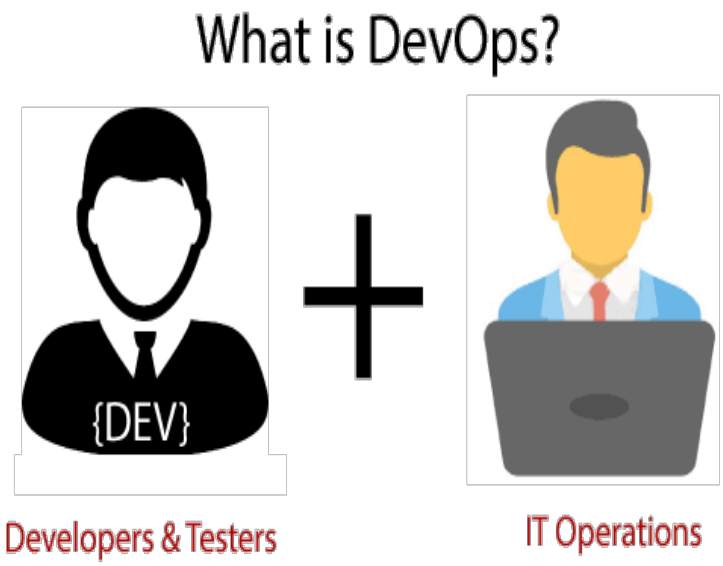

Fig. 1.

DevOps kavramı sektörde yerini aldıktan sonra sektörde de birçok yeni teknoloji girmiş oldu ve bu teknolojiler operasyon sürecindeki işlemleri manuel yapılırken otomatikleştirmiş oldu. Yazılan kodların daha efektif yazılmasını ve maliyet açısından daha karlı olduğu için ileride IT sektöründe adından bahsettirmeye devam edecektir. Sıradaki bölümlerde sektöre giren bu teknolojilerin ne olduğu ve görevleri anlatılacaktır. DevOps kavramının daha iyi anlaşılabilmesi için bir DevOps ekibinde olan ve bir Yazılım geliştirme ekibinde olan 2 kişinin

yaşam döngülerini aşağıda gösterilmiştir.[3]

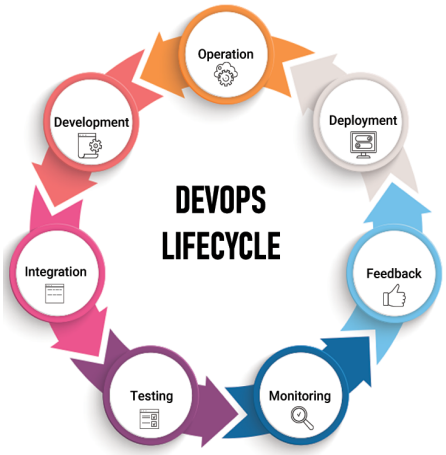

Fig. 2. DevOps yaşam döngüsü

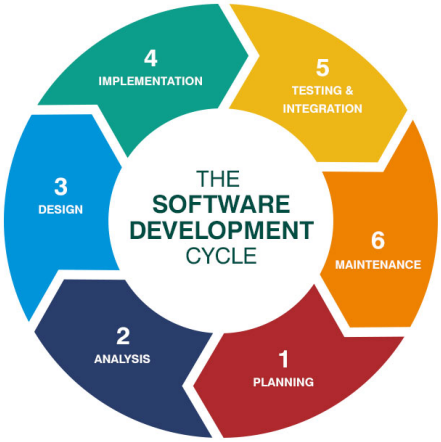

Fig. 3. Developer yaşam döngüsü

II. DevSecOps

A. *DevSecOps Nedir*

DevOps yazılım geliştirme sürecinde kod yazan geliştirici tarafıyla sistem yönetici arasında daha verimli ve etkin bir iş birliği kurmayı amaçlayan bir metodolojidir.

DevSecOps ise DevOps sürecine güvenlik uygulamalarını entegre etme sürecidir. DevSecOps yazılım geliştiriciler ile güvenlik kısmında çalışan kişiler arasında sürekli ve esnek bir işbirliği ile bir güvenlik kültürü 'Kod güvenliği' oluşturup bunu projenin başından sonuna kadar etkin bir şekilde yürütmeyi amaçlar. DevSecOps yöntemi tıpkı DevOps gibi yazılım geliştirme sürecindeki karmaşıklıkları bir bütün haline getirip geliştirme sürecine daha etkili bir pencereden bakıp etkili çözümler getirmeyi amaçlar. Amaç kodun daha sağlam ve güvenli kullanıcıya teslim edilmesini sağlarken Bilgi Teknolojileri ve güvenlik birimleri arasındaki oluşması muhtemel boşlukları doldurmaktır. Kodu kullanıcıya gitmeden önceki tüm süreçlerinde olağan tüm iletişimsel ve güvenlikle alakalı konuların sorumluluğu, muhtemel çıkabilecek tüm anlaşmazlıklarda taraflar arasında verimli ve çözüm odaklı müzakerelerde bulunması güvenli uygulamalar oluşturmak için çok önemlidir. DevSecOps yazılım geliştirme sürecini özellikle güvenlik denetimleri de dahil olmak bütün bir süreci otomatikleştirmeyi amaçlayan bir yöntemdir.

DevSecOps metodolojisi yazılım geliştirmeyi bileşenlerine ayırır ve o şekilde inceler . DevSecOps yöntemini 5 ana başlığa ayrılır:

1)Geliştirme (Development) Yazılım geliştirme aşaması, yazılım uygulamalarının oluştuğu aşamadır. Bu aşamada kod üretilir, mevcut kod güncellemeleri yapılabilir.Bu aşamadada güvenlik konuları dikkat edilmeli kod üzerinde herhangi bir açık olmamasına dikkat edilmelidir.

2)Sürekli Entegrasyon ve Sürekli Dağıtım (CI/CD) Bu iki önemli yaklaşım yazılım geliştirme sürecinin otomasyonunu ve hızlanmasını hedefler. CI (sürekli entegrasyon) yazılım geliştirme sürecinin bir basamağıdır ve her bir yazılım geliştiricinin yazdığı, geliştirdiği yazılımları düzenli aralıklarla belirli bir kod deposunda (git vb.) birleştirilmesi ve bu birleştirilen yazılımın belirli otomatik testlerle sınanmasıdır. CI ile bu süreç boyunca geliştirilen yazılımda yapılan kod değişikliklerinin mevcut ana kod a uygunluğu ve işlevselliği gözden geçirilmiş olur . Eğer bir sorunla karşılaşılırsa ilgili geliştirici hemen geri bildirim alır.

CD(sürekli entegrasyon) ise CI'ın bir uzantısıdır. CD genel olarak yazılımın hızlı ve güvenli bir şekilde kullanıcıya,müşteriye ulaştırılmasını amaçlar. Mesela CI sürecinden başarıyla geçen yazılım otomatik olarak canlı olarak o an dağıtılır. CD süreci canlı bir süreç olduğundan performans testleri ve güvenlik kontrolleri gibi otomatik testler içerebilir. CI/CD işlemlerini otomatize bir şekilde yürütebilmek için Jenkins, Travis CI, Circle CI, GitlabCI/CD ve Azure DevOps gibi araçlar kullanılır. Bunlardan biraz bahsetmek gerekirse ;

Jenkins açık kaynak bir otomasyon sunucusudur. Otomatik test etme özelliği sayesinde zaman tasarrufu sağlar. Plugin sistemi sayesinde farklı teknolojilere entegre edilebilir. Travis CI otomatik bir şekilde yazılımın dilini algılar , kod değişikliklerini otomatik olarak algılayıp kodu build eder, test adımlarını çalıştırır, kodu deploy eder. Sonrasında ise geri bildirim verir. GitlabCI/CD Git tabanlı bir kod yönetim aracıdır . GitLabın sağladığı diğer özelliklerlede uyumlu çalışır. Azure DevOps Microsoft tarafından sunulan bulut tabanlı bir hizmettir.

3)Güvenlik Testleri Bu kısım, yazılımın güvenlik açıklarını ve tehdit olabilecek şeyleri bulmak için uygulanır. Bu kısımda kendi içinde statik analiz(kod inceleme), dinamik analiz(uygulama testleri), açık kaynak yazılım analizleri gibi yöntemlere ayrılır. Açık kaynak yazılım bileşenlerinin güvenlik ve lisans denetimleri için OWASP Dependency-Check ve WhiteSource gibi araçlar kullanılır.

4)Otomasyon DevSecOps tan bahsederken genellikle kullandığım bi kelimede otomasyondu. Bunun nedeni bütün bu süreç boyunca amaç bu güvenlik politikalarını ve kontrolleri otomatik olarak uygulamayı sağlamak. Bu yüzden

otomasyonda DevSecOps un güvenlik gibi ana hedeflerinden biri.

5) İşbirliği DevSecOps bütün bu süreç boyunca geliştirme, operatör ve güvenlik ekipleri arasında tam zamanlı bir işbirliğini teşvik eder. Bu işbirliğinin önemi güvenlikle ilgili çıkabilecek herhangi bir sorunda en erken aşamada hızlı çözümler üretebilmek. DevSecOps projeleri için kullanılan bir takım teknoloji ve araçlar da şunlardır; Güvenlik Bilgi ve Olay Yönetimi (SIEM) Araçları Günlük verilerini toplamak, analizini yapmak ve güvenlik sürecini takip etmek için SIEM araçları kullanılır. Örnek olarak Splunk, LogRhythm ve ELK Stack bulunur. Saldırı Yüzeyi Azaltma Araçları (Attack Surface Reduction) Kubernetes için kube-hunter veya Docker içinse Dockscan gibi araçlar güvenlik açıklarını tespit etmek için kullanılabilir. Güvenlik İzleme Araçları Saldırıları ve tehditleri izlemek, uygulama ve alt yapı düzeyinde güvenliği takip etmek için kullanılır. Intrusion Detection Systems (IDS) ve Intrusion Prevention Systems (IPS) gibi örnekler bulunur.[5][6]

III. DevOps için Kullanılan Teknolojiler

A. Linux

Bu bölümde bir DevOps mühendisinin neden Linux bilgisinin olması gerektiğini anlatılacaktır. Bir DevOps mühendisinin kesinlikle linux tabanlı işletim sistemlerine hakim olması beklenmektedir çünkü bilişim sektöründe özellikle sunucu ve bulut bilişim alanında en yaygın kullanılan işletim sistemlerinden biri olduğundan dolayı. Amazon Web Services (AWS), Microsoft Azure ve Google Cloud Platform (GCP) dahil olmak üzere çoğu bulut bilişim platformu Linux tabanlıdır. Bir DevOps mühendisinin bu platformlarla etkili bir şekilde çalışabilmesi gerekmektedir.Linux açık kaynaklı bir işletim sistemidir; bu nedenle son derece özelleştirilebilir olduğu ve çeşitli farklı ihtiyaçlara uyacak şekilde kolayca uyarlanabileceği anlamına gelir. Bu özellik onu DevOps ortamında uygulama geliştirmek ve dağıtmak için ideal bir platform haline getirir.Linux, görevleri otomatikleştirmesi, sunucuları yönetmesi ve sorunları gidermesi gereken DevOps mühendisleri için gerekli olan güçlü bir komut satırı arayüzü (CLI) sağlar. Linux CLI'nin iyi anlaşılması herhangi bir DevOps mühendisi için çok önemlidir.Linux, Nagios, Zabbix ve Ganglia gibi sistemlerin izlenmesi ve yönetilmesi için zengin araçlar ve yardımcı programlar sağlar. Bu araçlar, DevOps mühendislerinin sorunları hızlı ve verimli bir şekilde tanımlamasına ve gidermesine yardımcı olabilir. Linux son derece güvenlidir ve SELinux, AppArmor ve iptables gibi bir dizi güvenlik özelliği sunar. Bir DevOps mühendisinin, sistemlerinin güvenli olduğundan ve siber tehditlere karşı korunduğundan emin olmak için bu özelliklere aşina olması gerekir.

Docker gibi konteynerleştirme teknolojileri DevOps'ta uygulamaları paketlemek ve dağıtmak için yaygın olarak kullanılmaktadır. Docker'a, Kubernetes gibi konteyner düzenleme platformlarına ve konteyner ağı, depolama ve görüntü yönetimi gibi ilgili kavramlara aşinalık, konteynerli uygulamaları yönetmek için oldukça faydalıdır. DevOps genellikle Terraform, Ansible veya Puppet gibi araçları kullanarak altyapıyı yönetmeyi içerir. Bu araçlar, altyapı kaynaklarını tanımlamak ve sağlamak için bildirime dayalı veya zorunlu komut dosyaları kullanır. Altyapının kod komut dosyaları olarak nasıl yazılacağını ve yönetileceğini anlamak, altyapı tedariğinin ve yapılandırmasının otomatikleştirilmesi açısından değerlidir. Görevleri otomatikleştirmek ve verimli iş akışları oluşturmak için kabuk komut dosyası oluşturma uzmanlığı çok önemlidir. DevOps mühendisleri, tekrarlanan görevleri otomatikleştirmek, yapılandırmaları yönetmek ve dağıtımları düzenlemek için kabuk komut dosyaları yazma ve sürdürme konusunda rahat olmalıdır. Awk, sed, grep ve normal ifadeler gibi araçlara ilişkin bilgi, metin işleme ve otomasyon görevleri için değerli olabilir.[7][8]

B. Bash Scripting

Bash Script kabuk programlama dilini kullanan betik dosyaları ifade eder. Syntaxı Java ,C# ve Javascript gibi dillere göre daha zor ve kuralcıdır. Bir DevOps mühendisinin bash script bilmesi sistemin otomasyonunu sağlamak,Sistem analizi ve altyapı kaynaklarını yönetmek, dosyaları takip etmek ve kayıt almak gibi görevler için çok önemli bir konumda olmaktadır.

C. Vagrant

Bir DevOps mühendisinin Vagrant, altyapı yönetimi komut dosyalarının geliştirilmesi ve test edilmesi için size tek kullanımlık bir ortam ve tutarlı bir iş akışı sunar. Kabuk komut dosyaları gibi şeyleri hızlı bir şekilde test edebilirir. Vagrant yazılım geliştirme süreçlerinde yazılımcıların işlerini kolaylaştırmak ve bu işi bir döngü haline getirmek için kullanılan bir yazılım aracıdır.Proje geliştirme esnasında geliştiricilere sanal makineleri kolayca oluşturup yönetme,geliştirme ve projeleri farklı sistemlerde sorunsuzca çalıştırma imkanı sunar.

Vagrant aracılığı ile sanal makineyi birden fazla makineye bölüp rahat bir şekilde yönetebilir. Seçilen sanal makinelerimizi herkese açık yayınlayabilir. Herkes tarafından görülmesi istenmiyorsa eğer erişim engeli koyabilir.

Vagrant sayesinde sanal makinelerin yapılandırılması kod yazılarak mümkündür bu olay ile birlikte kodlar depolarda tutulur ve sürümlerin kontrolü daha basit bir hale gelir.[9]

Vagrant, bir "Vagrantfile" adını verdiği yapılandırma dosyasını kullanarak sanal makine ortamlarını tanımlar. Bu dosyayı oluşturarak, geliştiriciler, projeleri için hızlıca geliştirme ortamı oluşturabilirler.

Ubuntu işletim sistemine nasıl vagrant kurulumu nasıl olur? İlk olarak;

- sudo apt update
- sudo apt install vagrant

komutları girilerek ubuntu işetim sistemine yüklemesi yapılır

- vagrant init

komutu kullanılarak bulunduğumuz dizindeki dosyada vagrantı kurmuş oluruz. Aşağıdaki örnekte komutlar veirlmiştir.

```
vboxuser@ubuntu101:~/Desktop$ mkdir DevOps_Proje
vboxuser@ubuntu101:~/Desktop$ cd DevOps_Proje/
vboxuser@ubuntu101:~/Desktop/DevOps_Proje$ mkdir Vagrant_Proje
vboxuser@ubuntu101:~/Desktop/DevOps_Proje$ cd Vagrant_Proje/
vboxuser@ubuntu101:~/Desktop/DevOps_Proje/Vagrant_Proje$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
vboxuser@ubuntu101:~/Desktop/DevOps_Proje/Vagrant_Proje$ ls Vagrantfile
Vagrantfile
```

Fig. 4. Bulunduğumuz dizinde vagrantı başlatmak

Vagrantfile dosyasının içindeki kodlara gedit editörünü kullanarak bakabiliriz bunu içinde kullanılacak komut

- gedit Vagrantfile

Bu komutla beraber projemizin ihtiyacına göre içerisindeki yapılandırma değiştirebilir.

- vagrant up

bu komut ile beraber sanal makineyi başlatmaya yarayan konuttur.

- vagrant ssh

bu komut ile birlikte sanal makineyi başlatabilir.

temel yapılarıyla vagrant kullanımı linux işletim sisteminde bu şekildedir.

D. Networking

Bilgisayar bağlantısı iki yada daha fazla bilgisayarın bağlantısını sağlamak demektir.Bilgisayar bağlantısındaki önemli bileşenler;En az iki adet bilgisayar yada internete bağlanabilen bağlanabilen cihaz veya bilgisayarları birbirine bağlayacak kablo yada kablosuz network ve NOS(network operating system) yapıları örnek verilebilir.

Peki bilişim sektöründe neden networking kavramını kullanılır? DevOps mühendisleri, internet ağına olan bağımlılığı azaltmayı hedefleyip, otomasyon ve hızlı dağıtımlarla yazılım geliştirme süreçlerini hızlandırmaya çalışırlar. Bu nedenle, ağ yönetimi ve networking konuları, DevOps uygulamalarında önemli bir rol oynar.

1.)Ağ altyapısı yönetimi: Uygulamaların kullandığı sunucular,konteynerlar yada sanal makineler arasındaki ağların altyapılarının oluşturulması ve yapılandırılması gibi işlemlerde bu yönetim kullanılır.Bu alt yapı yönetimi ağ bileşenlerinin kurulumunu ve yönetimini içerir.

2.)Otomasyon: Ağ altyapısının otomasyonu, ağ kurulumlarının ve değişikliklerinin yazılım kodu veya temsilciler aracılığıyla otomatikleştirilmesini içerir. Böylece, ağ değişiklikleri hızlı bir şekilde yapılarak yazılım dağıtım süreçlerini destekler.

3.)Konteyner Ağları: Konteyner teknolojileri kullanıldığında bu konteynerlerin ağa nasıl bağlanacağı, ağ ile olan iletişimi ve getirilcek kısıtlamalar çok büyük bir önem taşımaktadır.Bu bahsedilen kavramlar ne kadar efektif bir şekilde yapılırsa konteynerlardan alınacak verim artacaktır.

4.)Hata Ayıklama: Ağ üzerindeki performansı ve durumu izlemek, hata ayıklama işlemleri sırasında büyük bir önem taşımaktadır. Ağ trafiğini izlemek ve ağ sorunlarını tespit etmek, sorun giderme süreçlerine yardımcı olur.

ISO (International Organization for Standardization) tarafından 1984 yılında geliştirilmiştir. Dünyada milyonlarca belki daha fazla olan ağların birbiri ile iletişim kurulmasına olanak tanımaktadır başka bir deyişle evrensel bir dil sağlar. OSI modeli ağ iletişimi işlevlerini yedi katmana bölen bir çerçevedir.

Bu model, ağ protokollerinin ve işlevlerinin yedi farklı katmana bölünmüş bir yapısıdır. Bu katmanlar arasındaki iletişimi açıklar. Her katman, belirli görevleri yerine getirir ve yukarıdan aşağıya doğru iletişim sağlar.

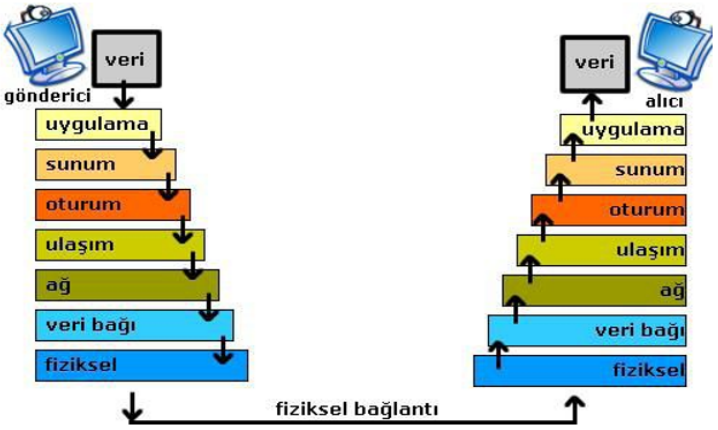

Fig. 5. OSI modelinin katmanları

1.)Fiziksel Katman (Physical Layer): Fiziksel katman verilerin bit veri türü şeklinde gönderildiği katmandır. Bu katmanın amacı verileri iletmektir.Bu katman, verinin elektrik sinyalleri, optik sinyaller, radyo dalgaları veya diğer fiziksel ortamlar aracılığıyla iletilmesini sağlar. Kablolama, veri iletim hızı ve sinyal gücü gibi konular bu katmanın içinde yer alır.

2.)Veri Bağlantısı Katmanı(Data Link Layer):Bu katmanın amacı fiziksel katmana nasıl erişileceğini belirler.Verileri bu katmanda, ağ katmanından fiziksel katmana gönderilir.Bu katman, veri paketleri halinde kapsüllenmiş dijital sinyaller olan veri çerçevelerini yönetir. Verilerin akış ve hata kontrolleri genellikle veri bağlantısı katmanının temel odak noktalarıdır. Veri bağlantı katmanı genellikle iki alt katmana ayrılır: Ortam Erişim Kontrolü (MAC) katmanı ve Mantıksal Bağlantı Kontrolü (LLC) katmanı. Veri bağlantısı katmanı, ağ kartı üzerinde çalışır.

3.)Ağ Katmanı (Network Layer): Bu katman, veri paketlerinin farklı ağlar arasında yönlendirilmesi ve hedefe ulaştırılmasıyla ilgilenir. IP adreslemesi ve yönlendirme işlemleri bu katmanın sorumluluk alanına girer.

4.)Taşıma Katmanı (Transport Layer): Veri akışının doğru ve güvenilir bir şekilde hedefe ulaşmasını sağlar. Veri bütünlüğünü korumak, hata düzeltme ve akış kontrolü gibi işlevleri bu katmanda yer alır. Üst katmanlardan gelen her türlü bilgi(veri) taşıma katmanı tarafından diğer katmanlara ve hedeflere ulaştırılır. Taşıma katmanı, uygulama katmanından aldığı veriyi paket haline getirir. Paketleme, büyük verileri küçük verilere bölme işlemidir ve bu bölünen parçalara segment ismi verilmektedir.

5.)Oturum Katmanı (Session Layer): İletişim oturumlarını oluşturur, sürdürür ve sonlandırır. Güvenlik, oturum yönetimi ve senkronizasyon gibi işlevleri içerir.Oturum katmanı, cihazlar arasındaki bağlantıları kontrol eder. Sunu katmanında gönderilicek veriler, farklı oturumlarda birbirinden ayrılır.

6.)Sunum Katmanı (Presentation Layer):Bu katman, veri formatını çözümler, sıkıştırır ve şifreler. Farklı veri formatları arasında dönüşüm sağlar ve veri şifreleme işlemleri bu katmanda gerçekleştirilir.

7.)Uygulama Katmanı (Application Layer): Kullanıcı uygulamaları ve ağ arasındaki iletişimi sağlar. Bu katman, e-posta, web tarayıcıları, dosya paylaşımı ve diğer uygulamalar için iletişim sağlar.

Uygulama katmanı son kullanıcıya en yakın olan OSI katmanıdır. Yani hem OSI uygulama katmanı hem de kullanıcı doğrudan yazılımla, uygulamayla etkileşimde bulunur. Bu katman iletişim bileşenini yürüten uygulamayla etkileşime girer. Bazı uygulamalar OSI modelin kapsamı dışına çıkabilir. Uygulama katmanı sayesinde iletişim kuran kişiler tanımlanır, kaynak kullanılabilirliğine karar verilir ve senkronize iletişim gerçekleştirilir. İletişim kuran kişiler tanımlanırken kişilerin bir uygulama üzerinden veri göndermek için gereken kimliğine ve kullanılabilirliğin yeterli olup olmadığına karar verir.

Açık Sistemler Arası Bağlantı (OSI) modelindeki katmanlar, uygulama ve temel sistemler ne kadar karmaşık olursa olsun bir uygulamanın farklı bir cihazdaki başka bir uygulamayla ağ üzerinden iletişim kurabilmesi için tasarlanmıştır. Bunu yapmak için üstteki veya alttaki katmanla iletişim kurmak üzere çeşitli standartlar ve protokoller kullanılır. Katmanların her biri bağımsız olup yalnızca üstündeki ve altındaki katmanla iletişim kurmak için kullanılan arabirimlerden haberdardır.

Tüm bu katman ve protokoller birbirine zincirlenmesiyle karmaşık veri iletişimleri bir üst düzey uygulamadan diğerine gönderilebilir. Süreç şu şekilde işler:

1.)Göndericinin uygulama katmanı, veri iletişimini bir sonraki alt katmana aktarır.

2.)Her katman, veriyi aktarmadan önce kendi başlık ve adreslemelerini ekler.

3.)Veri iletişimi, sonunda fiziksel ortam aracılığıyla iletilene kadar katmanlar arasında aşağı doğru ilerler.

4.)Ortamın diğer ucunda her katman, veriyi o düzeydeki ilgili başlıklara göre işler.

5.)Alıcı uçta veri, katmanda yukarı doğru hareket eder ve diğer uçtaki uygulamaya teslim edilene kadar kademeli olarak paketten çıkarılır.[10][11][12]

E. Container ve Docker Nedir?

Container teknolojisi, uygulamaların bir sunucu üzerinde bağımsız ve izole bir şekilde çalışmasını sağlayan teknolojidir. Bu teknoloji, uygulamaların, bağımlılıklarının ve gereksinimlerinin bir arada paketlendiği taşınabilir konteynerler oluşturur.

Konteynerler, her biri kendi sanal bilgisayarmış gibi davranırlar bir konteynerı silsek bir diğer konteyner etkilenmez ya da biri istenmeyen ekipler tarafından ele geçirilse diğeri yine etkilenmez yani birbirlerinden tamamen habersizlerdir. ancak fiziksel bir bilgisayar üzerinde çalışmaktadırlar. Konteynerler, yazılım uygulamalarının hızlı ve güvenilir bir şekilde dağıtılmasını, yönetilmesini ve ölçeklendirilmesini sağlayan güçlü bir teknolojidir. Bu sayede e-ticaret uygulamaları gibi karmaşık uygulamaların daha kolay ve etkili bir şekilde işletilmesi mümkün olur.

Konteynerler, bir bilgisayardan diğerine kolayca taşınabilirler, birbirlerinden ve ana bilgisayar sisteminden izole bir şekilde çalışmaktadırlar.

Konteyner orkestrasyon araçları, birden fazla konteyneri otomatik olarak yönetmek ve ölçeklendirmek için kullanılır. Bu, büyük ölçekli uygulamaların yönetimini kolaylaştırmaktadır.Konteynerler, daha az sistem kaynağı kullanır ve aynı anda daha fazla konteynerin çalışmasına izin verir. Bu, donanımın daha etkili bir şekilde kullanılmasını sağlar.

Konteyner teknolojisi, yazılım geliştirme ve işletme süreçlerini daha verimli ve esnek hale getirir. Bu avantajlar, konteynerlerin yaygın bir şekilde kullanılmasının sebeplerindendir bilişim sektöründe.

Docker, 2013 yılında çıktığında büyük bir etki yaratmış ve yazılım geliştirme süreçlerini temelden değiştirmiştir. Docker'ın yaratıcısı Solomon Hykes, 2010 yılında Docker'ı geliştirmeye başladı.Konteyner teknolojisinin gelişmesini ve uygulanmasını kolaylaştırmayı hedefliyordu.Docker, 2013 yılında açık kodlu kaynak olarak ilk kez piyasaya sürüldü ve hızla büyük bir ilgi gördü. Docker, konteyner teknolojisini kullanıcı dostu bir arayüzle sunarak yazılım geliştiricilerin uygulamalarını paketleme ve dağıtma süreçlerini büyük ölçüde kolaylaştırdı. 1.0 versiyonu yayınlandığında bazı bağımlılıklarını ortadan kaldırdı ve bütün altyapısını go programlama dili ile yazdı. Docker'ın başarısı, yazılım geliştirme süreçlerini büyük ölçüde değiştirdi ve konteyner teknolojisinin popülerliğini artırdı. Docker, hala birçok organizasyonun ve geliştiricinin tercih ettiği bir yazılım dağıtım ve yönetim aracıdır ve konteyner teknolojisinin önde gelen temsilcilerinden biridir.

Docker Engine: Docker engine için docker platformunun kalbi denilebilir. Server client mimarisinde bir uygulamadır. 3 temel parçadan oluşur.

1.)Docker Daemon: Docker Enginenin kalbini oluşturur. Docker objelerinin oluşturulmasını ve yönetilmesini sağlamaktadır. Bunu linux bir servera kurup üstüne konteynerlar kurup kullanabiliriz

2.)Docker RestAPI: Dış dünya ile Docker Daemon arasındaki bağlantıyı oluşturur.

3.)Docker CLI: Başka bir bilgisayardan docker daemonın cloud teknolojisi ile yönetilmesini sağlar.

2 farklı docker engine ürünü şu an piyasada bulunmaktadır.Bunlardan biri Docker Communitydir. Bu uyugulama kullanıcılardan herhangi bir ücret talep etmeden özelliklerini kullanıcıya sunar. Bir diğer uygulama ise Docker Engine Enterprise uygulamasıdır. Bu uygulama ücretlidir ve sıklıkla şirketler kullanılmaktadır. Uygulamanın en iyi yanı ortaya çıkabilecek herhangi bir problemde docker mühendisleri ile iletişim kurarak sorunun çözümüne destek verirler.[13][14][15]

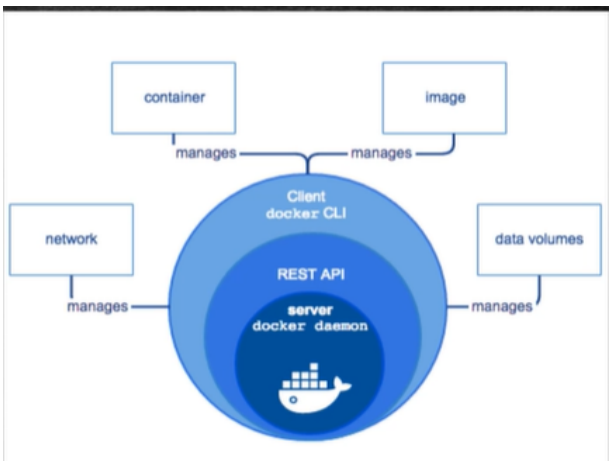

Fig. 6. Docker Engine katmanları

Docker İmage: Bir uygulamanın ve onun içinde o uygulamanın çalışması için ekstra olan şeylerin paketlenmiş halinde Docker Image denir. İçinde herhangi bir kernel bulunmamaktadır çünkü konteynerlar üstünde koştukları işletim sisteminin kernelini kullanırlar. Image ile konteyner arasındaki farka değinmek gerekirse konteyner imagein çalışır vaziyette olan halidir.

Docker sanal makine arasındaki fark nedir? Sanal makine ile konteynerların en önemli farkı konteyner içersinde bir işletim sistemi çekirdeğine ihtiyaç duymadığımız için ortamda yönetmemiz gereken ve kaynak tüketimi yüksek olan işletim sistemi bulunmaz

Docker Hub: Docker imajlarını depolayıp dağıtabildiğimiz bir altyapı sağlayan servise verilen addır. Bu depolar public yada private olabilir.

Docker engine'i linuxa nasıl kurulacağı komutları ve alınabilmesi muhtemel hatalar "https://docs.docker.com/engine/install/ubuntu/" sitesinde yer almaktadır.Kurulum sonunda eğer kurulumda bir hata olup olmadığını teyit edilmek isteniyorsa ;

- sudo docker run hello-world

komutu kullanılabilir.

Docker CLI: (Commond Line Interface) Sisteme docker yüklediğimiz zaman otomatik olarak yüklenen ve o docker engine'i yönetmemizi sağlayan komut satırı arayüzüdür. Daemon ile karıştırılmamalıdır.Docker kullanıclara bir rehber sunmaz bilgileri bazı komutlar ile terminalde yazdırır.

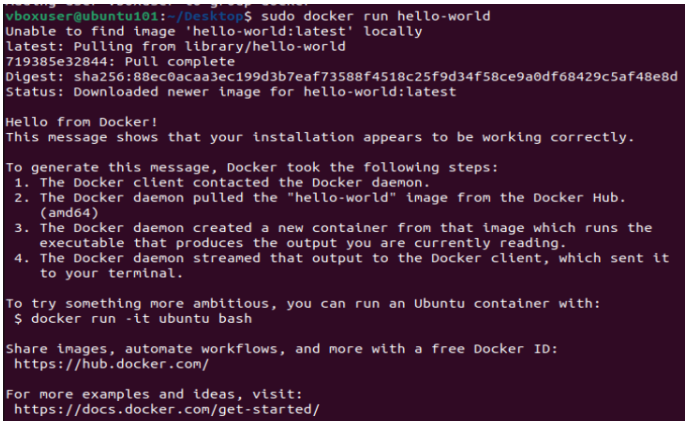

Fig. 7. Docker engine uygulamasını sorunsuz indiğini gösteren komut

Her konteyner imageinde o imageden bir conteiner yarattığımız zaman varsayılan olarak çalışması için bir uygulama vardır ve bu uygulama çalıştığı sürece konteyner ayakta kalır. Uygulama çalışmayı bıraktığında konteyner da kapatılır.Konteynerlar tek bir uygulama çalıştırmak için kullanılırlar.Kullanılan bu uygulama durduğunda konteynerlarda durdurulur.Dockerın bir özelliği ile bir konteyner kapatıldığında aynı image kullanılarak başka bir konteyner oluşturulur. Eğer konteyner içindeki uygulamada bir sıkıntı çıkarsa konteyner içine bağlanılıp bu problem çözülmez konteyner kapatılır ve yeni bir konteyner açılır eğer sorun image tarafında ise oradaki sorun çözülür ve yeni bir konteyner üretilir özetle konteynerlar çok sık değiştirilir bu yüzden konteyner içerisinde bilgi tutmak çok tercih edilen bir yöntem değildir. Bu yüzden docker volume kavramını kullanıcılara sunmuştur. Konteynerla bir bağları yoktur konteynerlar silinse bile volumeler sistemde saklanılmaya devam eder.Bir konteynaer içinde yapacağımız değişiklik volume içinde kaydedildiye o kayıtı ibre başka konteyner içindede açabiliriz.

- docker volume create (volumeismi): Yeni bir volume oluşturmak için kullanılan komuttur.
- docker volume inspect (volumeismi): Oluşturulan volumelerin detaylarını ekrana bastırır.

konteynerlara volume bağlamak için -v (volumeismi) kullanılabilir run komutunun sonuna eklenerek.

Union File System: Docker, uygulama ve bağımlılıklarını bir konteyner içine paketleyerek bu konteynerin farklı ortamlarda aynı şekilde çalışmasını sağlar. Docker'ın bu yeteneği kullanıcıların "union file system" veya "birleşik dosya sistemleri" olarak bilinen bir özelliği sayesindedir.

Union file system, farklı dosya sistemlerini birleştiren bir yapıdır. Docker, bu özelliği kullanarak, bir konteynerin dosya sistemini oluşturur. Konteyner, bir ana dosya sistemine (base image) sahiptir ve bu ana dosya sistemine, daha sonra eklenen katmanlar (layers) aracılığıyla değişiklikler yapılabilir.Bu birleşik dosya sistemleri, konteynerlerin hızlı başlatılmasını, daha az depolama alanı kullanılmasını ve kaynakların daha etkin kullanılmasını sağlar.

Docker konteynerleri bu birleşik dosya sistemleri sayesinde, her konteynerin kendi dosya sistemini izole edilmiş bir şekilde yönetmesini ve paylaşılan bileşenleri daha etkin bir şekilde kullanmasını sağlar.

Bazı Docker Komutları:

- docker version: Docker uygulamasının son sürümünü ve API versiyonu gibi birkaç özellik daha gösterir. Eğer bu bilgiler ekrana yansımmıyorsa docker daemon çalışmıyor yada client daemona bağlanamamış demektir.
- docker info: Sistemde çalışan docker ile ilgili temel bilgileri elde edilir.
- docker: Docker CLI'da kullanılacak komutları listeler.
- docker container run: Bu komut sistemde olmayan bir konteynerı oluşturur ve çalıştırmaya başlar ya da sistemde olan konteynerı çalıştırmaya başlatır bir bakıma run komutu create ve start komutlarının birleşimidir.
- docker container ls: Sistemdeki konteynerları listeler. -a ve -la gibi parametreleride alabilir.
- docker container run –name (verilecekisim): Bu komut sayesinde konteynerlara erişimi kolaylaştırmak için isim verebilir. Eğer kullanıcı bir isim girmezse docker kendisi otomatik bir isim atamaktadır. Dockerın isim atama mantığı ise ilk ingilizce bir sıfat sonra bir bilim adamının soyadını vermektedir.
- docker login: Bu komut Docker Hub'a bağlanmayı ve imageleri atmayı veya çekmeyi sağlar.
- docker image build -t imageadı . : Bu komutla beraber adını kullanıcının verdiği bir image oluşturulmuş olur.
- docker push imageadıDockerHubusername/DockerHubReposiyoryname: Bu komutla beraber var olan image'i Docker Hubda bulunan dosyaya atılır.
- docker pull DockerHubusername/DockerHubReposiyoryname:tagname: Bu komutla beraber belirtilen yerdeki image çekilir.
- docker run DockerHubusername/DockerHubReposiyoryname:tagname : Bu komutla beraber çekilen image çalıştırılır.
- docker –help: Sistemdeki tüm komutların ne olduğunu bir cümle ile özetler.

F. GIT

Git dünyada en çok kullanılan bir versiyon kontrol sistemidir. Git sayesinde yapılacak projelerin adım adım versiyonlarının kopyalarını alarak daha sonra ihtiyaç duyduğunuzda aldığınız kopyalara yani versiyonlara kolayca dönülebiliyor. İlk sürümü Linux çekirdeğinin geliştirilmesinde kullanılmak üzere 2005 yılında Linus Torvalds tarafından tasarlanıp geliştirilmiştir. Açık kaynaklı özgür bir yazılım ürünü olmaktadır. farklı geliştiricilerin aynı projede aynı anda çalışmalarını sağlar. Bu durum ekip üyelerinin kodlarını ayrı kollar üzerinde geliştirmelerine ve daha sonra bu değişiklikleri birleştirmelerine olanak tanır.

Git farklı görevler veya özellikler üzerinde çalışmak için bağımsız dal oluşturmayı (branch) ve daha sonra bu değişiklikleri ana geliştirme dalına birleştirmeyi (merge) kolaylaştırır. Bu kodunuzun düzenli ve kontrol edilebilir bir şekilde geliştirilmesine olanak tanır. DevOps mühendisleri, Git kullanarak kodun test ve üretim ortamlarına geçişini yönetebilirler. Git kodunuzu bir önceki işlevsel durumuna geri almanızı veya hataları düzeltmenizi sağlar. Bu özellikler, geliştirme sürecini daha güvenli ve kontrol edilebilir hale getirir. Bu, sürekli entegrasyon (CI) ve sürekli dağıtım (CD) süreçlerini etkili bir şekilde uygulamalarını sağlar.[16]

Bazı git komutları;

- git init: Klasörün içine giti aktif eder. 2 kere kullanılınca sistem hata verebilir o yüzden statu ile kontrol edilmelidir.
- git add: bu komut ile birlikte dosyalar sahnelenir(staging area) ve commit edilmeye hazırdır.
- git commit -m "commit mesajı": Bu komutla beraber hazırlanan kod commit edilmiş olur.
- git log : Bu komutla beraber atılan commitleri gösterir.
- git ignore: Bu komutu kullanmak ilk olarak dosya oluşturulmalıdır sonra dosyanın içine görmezden geleceğimiz dosyaların adı yazılır.dosya hala da kullanılabilir fakat git komutları kullanılmaz. touch .gitignore komutu kullanılarak açılan dosyanın içine yazılır gözükmesini istenmeyen kod.
- git branch :Bu komut Bizim için tüm güncel branchleri gösterecektir.head ne demek: bizim tam olarak nerede olduğunuzu gösterir main branchdir.Yeri değiştirilebilir. Bu komut -r parametresi alırsa githubdaki güncel branchleri alırız.
- git branch feature: Bu komut feature adında bir branch oluşturur.
- git switch feature : bu komut ile birlikte masterdan feature adlı breache geçeriz.
- git merge feature : iki tane branchi birleştirir.dallar bağlanır ve yazılanların hepsi görülür.
- git checkout komutu: git checkout <commit_kod>: Bu komut atılan son commiti geri alır.(Detached HEAD) geri alınan commite dönmek için git switch master komutu yazılabilir.(Reattched HEAD)
- git fetch : git fetch origin master şeklinde kullanılır proje ile repository durumu öğrenilir.

Bu komutlar gitin temel komutlarıdır diyebiliriz. Bu komutlar kullanılırken dikkatli olmak çok önemlidir çünkü yaratılacak bir çatışmada(conflict) yazılan kodlar yanabilir ya da başka bir developerın yazdığı kodlar yanabilir .

G. Kubernetes

konteynera dönüştürülmüş dağıtık mimarideki uygulamaları kesintisiz biçimde çalıştırmamıza imkan veren bir uygulamadır.Sıkça "buluta yönelik işletim sistemi" olarak tanımlanan Kubernetes, konteynere alınmış uygulama ve hizmet kümelerini yönetmek amacıyla tasarlanmış, açık kaynaklı bir platformdur.Kubernetes ayrıca, uygulamalarınızın

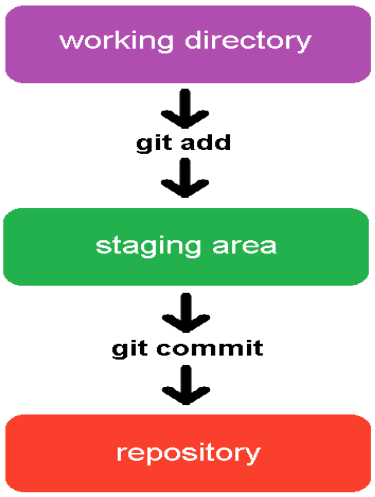

Fig. 8. Kod commit edilirkenki aşamalar

sağlıklı bir şekilde çalıştığını kontrol etme (ve hatta dağıtım sırasında herhangi bir şeyi olumsuz etkiliyorsa değişikliği tersine çevirme), tercih ettiğiniz depolama sistemini kurma, uygulamalarınızı ölçeklendirme, kendi kendini iyileştirme (container'ları otomatik olarak değiştirme) gibi bir dizi başka otomatik işlevi yerine getirir. Gerektiğinde yanıt vermeyenleri etkisiz hale getir, otomatik ölçeklendirme gerektiğinde, başarısız kapsayıcıları yeniden başlatabilir veya yeniden zamanlayabilir.

Docker ve Kubernetes tartışması, ikisi arasında seçim yapmaktan çok, bunları birlikte kullanmanın yollarını bulmakla ilgilidir. Çünkü bu iki platform farklı işlevlere hizmet eder. Docker, kapsayıcıları oluşturan ve dağıtan açık kaynaklı bir kapsayıcı platformudur. Kubernetes ise, bir kapsayıcı düzenleme platformudur. 2013'te piyasaya sürülen Docker, varsayılan kapsayıcı dosya biçimidir ve esasen "kapsayıcıların" ile eş anlamlı hale gelmiştir. Hem Linux hem de Windows ile uyumludur, şirket içinde ve bulutta çalışabilir.

Docker kapsayıcıları oluşturur ve çalıştırır; Kubernetes ise bunları planlayan, ölçekleyen ve taşıyan denetleyici-yöneticidir. İşbirliği içinde, kapsayıcılarınızı oluşturmak, çalıştırmak ve kapsayıcı görüntülerini depolamak için Docker'ı kolayca kullanabilir, ardından bu kapsayıcıları (ve kaynaklarını) bir Kubernetes kontrol düzleminden düzenlemek için Kubernetes'i kullanabilirsiniz. Docker ve Kubernetes'i birlikte kullanmak, deneyimleri ve geliştiricilerin ölçeklenebilir uygulamalar oluşturmasını kolaylaştırırken; ekiplerin bulutta yerel mimarileri veya mikro hizmetleri daha verimli bir şekilde oluşturmasına olanak tanır.[17][18]

Monolith Mimari: Bir uygulamanın büyük ve bağımsız bir yapıda tasarlandığı ve çalıştığı bir mimari tarzı ifade etmektedir. Monolith mimari, tüm uygulama bileşenlerinin çoğunlukla aynı kod tabanı içinde bir araya getirildiği bir felsefeyi temsil eder. Bu tip mimaride, genellikle kullanıcı arayüzü, iş mantığı ve veritabanı yönetimi gibi farklı işlevler tek bir uygulama içinde entegre edilir. Monolith mimari projelerin geliştirilmesini daha kolay bir hale getirmektedir lakin Proje büyüdükçe hakimiyetin azalması bir dezavantajdır.

Mikroservis Mimari: Mikroservis mimarisi, bir yazılım uygulamasını küçük ve birbirinden bağımsız hizmetlere bölen bir yazılım geliştirme yaklaşımını ifade eder. Bu hizmetler genellikle belirli işlevleri yerine getirir ve genellikle kendi veritabanlarına sahiptir. Mikroservis mimarisi, monolitik mimariden farklı olarak, bir uygulamanın farklı bileşenlerini ayrı servisler olarak tasarlamayı ve geliştirmeyi amaçlar.Mikroservis mimarisi, büyük ve karmaşık uygulamaların geliştirilmesi ve yönetilmesi konusunda bazı avantajlar sağlayabilir. Ancak, bu mimari tarzının getirdiği yönetim karmaşıklıkları ve altyapı ihtiyaçları göz önüne alındığında, doğru kullanım durumları seçilmelidir.[19]

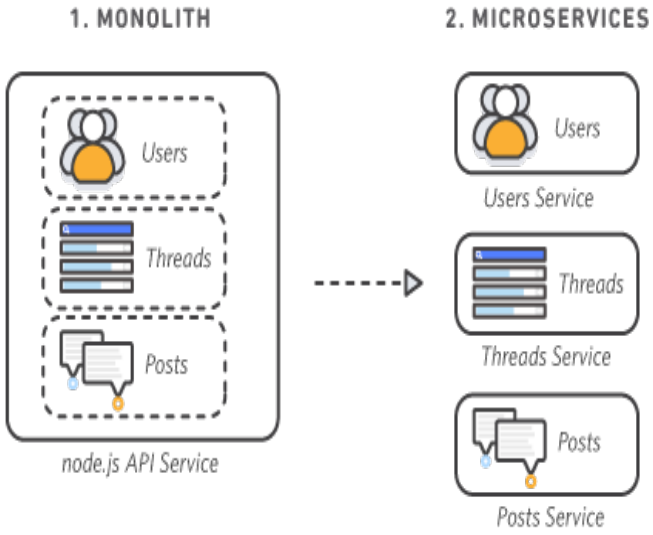

Fig. 9. Monolith Mimari ve Mikroservis Mimari

Kubernetes Mimarisi: Tek bir Kubernetes adında bir uygulama yoktur. Kubernetes platfromunu oluştuan bir çok servis mevcuttu ve bu servisler doğru kurgulanarak kubernetes platformu oluşur.

Node: Kubernetesde çalışan fiziksel veya sanal bir makineyi ifade eder.Nodelar iş yüklerini çalıştıran sunucu olarak hizmet verir.

1.)Kube-apiserver:Kubernetes kontrol panelinin en önemli bileşini ve giriş noktasıdır. Diğer tüm bileşenlerle direk iletişim kuran tek bileşendir.

2.)etcd: cluster verisi, metadata bilgileri ve Kuberneteste oluşturulan objelerin bilgilerinin tutulduğu key-value deposudur.

3.)kube-scheduler:yeni oluşturulan podları izler ve üzerine çalışılan bir node seçer.

4.)Kube-contoller-manager:Karmaşıklığı azaltır. Bu 4 bileşen Kubernetesin yönetim kısmını oluşturmaktadır.

Container Runtime: Konteynerları çalıştırmaktan sorumlu olan yazılımdır.Docker Kubernetesin desteklediği bir Container Runtime olmaktadır.

kubelet:Çeşitli mekanizmalar aracılığı ile pod tanımı alır ve konteynerların çalışmasını ve ayakta kalmasını sağlar.

kube-proxy:nodelar üstünde ağ kurallarını yönetir.

Kubernetes'te en küçük dağıtılabilir ve yönetilebilir birim "Pod" olarak adlandırılır. Bir Pod bir veya birden fazla konteyneri içeren en küçük dağıtım birimidir. Bu konteynerler birlikte çalışır ve aynı ağ ve depolama alanını paylaşırlar.Podlar genellikle bir uygulamanın belirli bir bileşeni veya mikro hizmetini temsil eder. Konteynerlerin bir arada çalışmasını sağlar ve birlikte aynı ağ ve depolama kaynaklarını paylaşırlar.Kuberneteste bir pod eklemek için, bir YAML dosyası oluşturulması ve bu dosyayı kullanarak bir kaynak tanımı oluşturulması gerekir.[20]

H. CICD

CI/CD (Continuous Integration/Continuous Delivery)(Sürekli ekleme/sürekli dağıtım)CICD felsefesi bilişim sektöründe çok önemli bir noktada bulunmaktadır. Yazılım geliştirme sürecinde yer alan bir dizi otomatikleştirilmiş işlemdir. Bu süreçler, yazılımın kalitesini artırmak ve yazılımın hızlı bir şekilde kullanıma sunulmasını sağlamak için tasarlanmıştır. Sürekli entegrasyon farklı ekiplerin yazdıkları kodların birleştirilip kodların test edilmesidir bu işlemle beraber hataların önceden görülmesi ve ortaya çıkacak ürünün daha hızlı bir şekilde konulması hedeflenir. Sürekli dağıtım ise yazılan kodların güvenli bir ortamda birleşmesine olanak sağlar ve müşteri tarafından verilecek feedback mekanizmasına karşı önlemler alınıp ürün değiştirilebilir. [21][22]

I. Jetkins

Jetkins açık kaynaklı bir CICD uygulamasıdır.Bulut tabanlı bir mimariyi destekler ve Java ile oluşturulmuştur. Jetkins çok geniş bir paket eklenti sınıfına sahiptir bu yüzden farklı teknolojiler ve platformların uyumlu çalışmasına olanak sağlamaktadır bu da tercih edilme nedenlerinin başında gelmektedir. Somut bir şekilde anlatmak gerekirse bir proje Git gibi bir versiyon kontrol sistemi üzerinde bulunsun.Jetkins proje sistemde iken gerekli testlerini yapıp işlemler hakkında bilgi veriyor ve CICD felsefesi ile birlikte uygulamayı otomatize ediyor. Herhangi bir sıkıntıda ise Jetkins ilgili ekibe sıkıntıyı iletiyor.Yani hem yazılım geliştirme kısmında bir değişiklik varsa repositoryden bu değişikliği algılıyor ve uygulamaya üzerinde uyguluyor, uygulamada bir sıkıntı varsa mesaj yolluyor ve uygulamayı güvenli bir şekilde müşteri ile buluşturuyor.[23][24][25]

J. Terraform

Terraform, HashiCorp tarafından oluşturulan ve Go programlama dilinde yazılmış açık kaynaklı bir araçtır ve Declarative bir dildir. IaC(altyapıyı kod ile tanımlama) altyapı kaynaklarını yazılım geliştirme prensipleri ve pratikleriyle yönetme yaklaşımını ifade eder. Bu, altyapı kaynaklarını kod

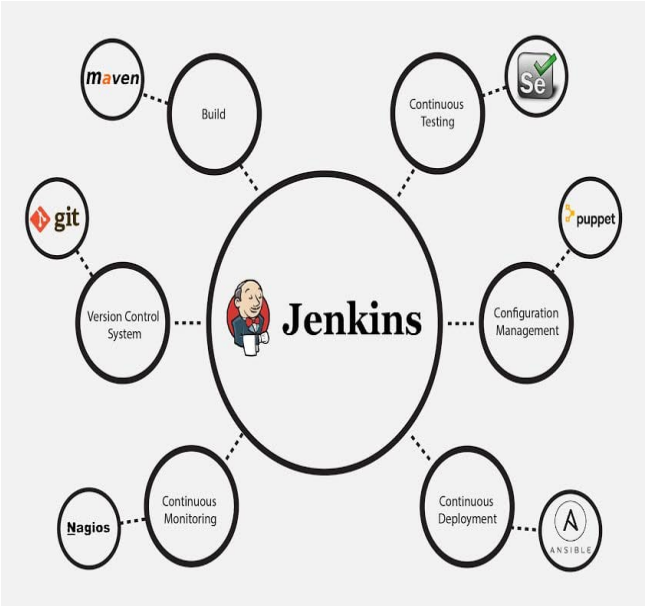

Fig. 10. Jetkins

olarak tanımlayarak, yöneterek ve dağıtarak manuel işlemleri en aza indirgeme amacını taşır Terraform da bir IaC aracıdır. Go kodu, terraform adı verilen tek bir binary dosyada derlenir.

Terraform altyapı kaynaklarını verimli ve temiz bir şekilde oluşturulmasına,yönetilmesine ve güncellenmesini sağlayan açık kaynaklı bir uygulamadır.Terraformun bilişim sektöründe tercih edilmesinin nedenlerinden biri Terraform, altyapıyı basit ve anlaşılır bir şekilde tanımlamanızı sağlayan bildirimsel bir dile sahip olmasıdır.Terraformu kullanarak sanal makineler, ağlar, veritabanları ve daha fazlası gibi çeşitli kaynaklar oluşturabilir ve yönetilebilir. Terraform ayrıca kaynaklar arasındaki bağımlılıkları tanımlamasına ve bu bağımlılıkları tutarlı ve öngörülebilir bir şekilde yönetilmesine olanak tanımaktadır.Tüm altyapı Terraform yapılandırma dosyalarında tanımlabilir ve bu dosyaları sürüm kontrolüne tabi tutabilir. Ardından bu altyapıyı dağıtmak için terraform apply gibi belirli Terraform komutlarını çalıştırılır. Terraform binary dosyası, kodu ayrıştırır ve kodda belirtilen bulut sağlayıcılarına bir dizi API çağrısına dönüştürür.Yazılım ekibinden birinin altyapıda değişiklik yapması gerektiğinde altyapıyı manuel olarak ve doğrudan sunucularda güncellemek yerine değişikliklerini Terraform yapılandırma dosyalarında yapar bu değişiklikleri otomatik testler ve kod reviewler yoluyla doğrular, güncellenen kodu garanti eder. Sürüm kontrolü ve ardından Terraform'un değişiklikleri dağıtmak için gerekli API çağrılarını yapmasını sağlamak için terraform apply komutunu çalıştırır.[26][27][28]

Linux işletim sisteminde Terraform kurmak için gerekli komutlar:

- sudo apt-get update
- sudo apt-get install terraform

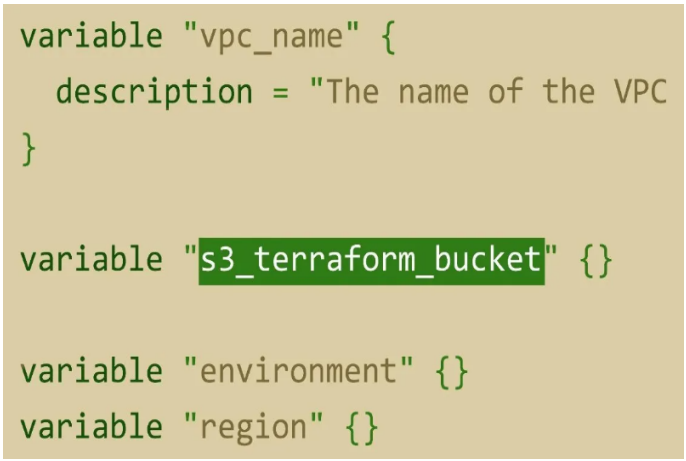

Fig. 11. Terrafrom kod örneği

K. Ansible

Ansible Red Hat şirketinin kurduğu açık kaynak bir şekilde ya da ticari versiyonunda bulunduğu bir IaC aracıdır. Ansible yazılım ekibindeki bir personelin sunucuları tek bir çatı altında toplayıp istenilen işlerin(Otomasyon, konfigürasyon ve işletim sistemi ayarları, kullanıcı işlemleri, servis ayarları, dosya işleri, güvenlik ayarları, continuous deployment (CD), update etme gibi işlemleri otomatikleştiren ve SSH bağlantısı ile gerçekleştirmemizi sağlayan araçtır.) yapılmasına olanak tutan bir uygulamadır. Python ve Ruby dilleri ile birlikte geliştirilmiştir.Ansible, hedef makinelere SSH protokolü ile bağlantı sağlayarak işlemleri gerçekleştirir.

Ansible, konfigürasyon dosyalarını kullanarak birden çok sunucu üzerinde yapılandırma değişiklikleri yapabilir. Bu dosyalar, sunucularınızın ne yapmasını istediğinizi belirtir ve hangi komutların çalıştırılması gerektiğini belirtir eğer istenirse bu değişiklikler geri alınabilir. Bu özelliği ile Terraformdan ayrılmaktadır Terraformda bu özellikler bulunmamaktadır.İki uygulama arasındaki bir diğer fark ise Terraform Declarative bir uygulama iken Ansible Prodecural bir uygulama olarak kullanıcıların karşısına çıkıyor.

Linux işletim sisteminde Ansible kurmak için gerekli komutlar:

- sudo apt-get update
- sudo apt-get install ansible

Ansible'ı kullanmadan önce, yapılandırma dosyası oluşturulması gerekir. Bu dosya Ansible'ın nasıl çalışacağını belirleyen bilgileri içerir. Yapılandırma dosyası ansible.cfg olarak adlandırılır ve ansible kurulu olduğu dizinin içinde bulunur.Ansible'ı kullanmak için, bir adım adım görev tanımlamanız gerekir. Bu görevler, YAML dilinde tanımlanır ve adımları ile birlikte açıkça belirtilir.[29][30][31]

L. Maven

Maven java programında yer alan komutarın derlenmesi sırasında kullanılan bir inşa aracıdır.Maven, Java geliştiricilerinin günlük işlerini kolaylaştırır ve genellikle herhangi bir Java tabanlı projenin anlaşılmasına yardımcı olur. Maven sayesinde projeler hızlıca build edilir, proje istenen formatta kolaylıkla derlenebilir, projenin bağımlılıkları kolayca yönetilir ve proje versinon kontrol sistemlerine entegre etmesi kolaylaşır.

Project Object Model(POM) dosyası hem proje hakkında bilgileri hemde projenin konfigürasyonu hakkında, bağımlılıkları, kaynağı, kullanılan pluginler, projeyi derlemek için gerekli komutlar vb. Bilgi içeren bir dosyadır. Eğer bir proje üzerinde Maven komutlarını kullanılmak isteniyorsa o projenin POM.xml dosyasını uzantısına sahip olması gerekmektedir.[32][33][34]

M. Monitoring

Kompleks yapılı sistemlerde, cihazların donanımsal durumunu veya içine kurulan yazılımları izleme işine denir. Monitoring 3 Ana bölümden oluşur bunlar: Alarm, Performans ve Eventler olmaktadir.

1.)Performans: Sistemlerin performans değerini izlemek ve bunu grafiksel ortama aktarmaktır. Yorum kabiliyetini artırmak adına sayısal verilerin görsel verilere dönüşmesi gerekir. Mevcut sistemde geçmişe dönük raporlamada çok önemli değer taşıyan performans grafikleri, anlık kaynak ihtiyacını ve gelecekteki kaynak ihtiyacını da kullanıcılara yol gösteren önemli bir etkendir. İzlenen performans değerlerinin izleme sıklığı vardır. Genel olarak 5 dakika bir değer alınır.Bu süre kullanıcının ihtiyacına göre alınır fakat sistemden çok sık bilgi alınması sistemin yoğunluğunu arttırıp performans sorunlarına yol açabilmektedir.

2.)Event: Her programın her cihazın bir loglama mekanizması vardır yaptığımız her işte her adımda arkada iz bırakılır. Bu iz bırakılan yerler bir sorun olduğunda ilk bakılacak yerlerdir. Hem güvenlik olarak hem de sistemi izleme açısında önemlidir. Bazı şirketler herhangi bir sıkıntıya yol açmaması için Eventi yasal bir zorunluluk haline getirmiştir. O yüzden merkezi bir yerde tüm logları toplayıp yönetilmesi gerekir.

3.)Alarm: Sistemlerde istenmeyen durumların oluşmasından önce haberdar olmak hem de istenmeyen durumların oluştuğunda hemen farkına varılması için Alarm bazlı Monitoring kullanılır.[35][36]

N. Puppet

Puppet, sunucuların dağıtımı, yapılandırılması ve yönetilmesi için kullanılan bir Konfigurasyon Yönetimi aracıdır.Puppet hem ticari hem de açık kaynaklı bir yazılım olarak iki farklı versiyondan oluşmaktadır. Ruby programlama dili ile geliştirilmiştir.Her bir host için ayrı konfigürasyonların tanımlanması ve gerekli konfigürasyonun yerinde olup olmadığının ve değiştirilmediğinin sürekli olarak kontrolünü sağlar. Yapılan değişikliğin simüle edilmesi ve uygulama sonrasında rapor verme özelliği sayesinde bilişim alanında tercih edilmektedir.[37][38]

O. Cloud Technoloyies

Bilgisayar ve depolama kaynaklarını internet aracılığıyla paylaşma ve erişme maksadıyla kullanılan teknolojiye bulut teknolojisi denir. Geleneksel bilgi işleme modellerinde,

kullanıcılar çoğunlukla kişisel bilgisayarlarındaki veya yerel ağlarındaki donanım ve yazılım kaynaklarını kullanırlar. Fakat bulut teknolojisi, bu kaynakları internet aracılığıyla sağlayan ve kullanıcılara geniş bir hizmet seçeneği sunan bir modeldir.

Bu teknolojinin ana özellikleri arasında erişim kolaylığı bulunur; bulut hizmetleri, kullanıcılara herhangi bir yerden, herhangi bir aygıt üzerinden internet aracılığıyla erişim imkanı sağlar. Bununla birlikte paylaşılan kaynaklar içerir; bulut, birçok kullanıcı tarafından paylaşılan sanal kaynakları içerir. Bu kaynaklar arasında sunucular, depolama, ağlar ve hizmetler bulunabilirler.

Ölçeklenebilirlik, bulut teknolojisinin diğer bir önemli özelliğidir. Kullanıcılar, isteklerine göre hızlı ve kolayca kaynaklarını artırabilir veya azaltabilirler, bu da ölçeklenebilirlik olarak ifade edilir. Ayrıca, bulut teknolojisi farklı hizmet modellerini içerir: Altyapı (Infrastructure as a Service - IaaS), platform (Platform as a Service - PaaS), ve yazılım (Software as a Service - SaaS).

Altyapı Hizmetleri (IaaS - Infrastructure as a Service)

Altyapı Hizmetleri, bulut bilişim hizmet modelinin bir parçasıdır ve kullanıcılara temel bilişim kaynaklarını sanal olarak temin eder. Bu, fiziksel donanım, depolama, ağ ve diğer bilişim kaynaklarını içerir. IaaS modeli, kullanıcılara gereksinim duydukları altyapıyı yönetme özgürlüğü sağlar, bununla birlikte uygulamalarını ve iş yüklerini bu altyapı üzerine inşa edebilirler.

IaaS hizmetleri genellikle şu temel özelliklere sahiptir:

1.)Sanallaştırma : Fiziksel donanım kaynakları sanallaştırılır, bu da kullanıcılara sanal makineler (VM'ler) gibi sanal kaynaklara erişim sağlar.

2.)Ölçeklenebilirlik : Kullanıcılar, ihtiyaçlarına bağlı olarak kaynaklarını hızla artırabilir veya azaltabilirler. Bu, iş yüklerine dinamik olarak uyum sağlamak için önemlidir.

3.)Self-Service : Kullanıcılar, genellikle bir kontrol paneli veya API aracılığıyla kendi altyapılarını yönetebilirler. Bu, hızlı dağıtım ve konfigürasyon sağlar.

4.)Ücretlendirme : Kullanıcılar, kullandıkları kaynaklar için ödeme yaparlar. Bu, maliyetleri optimize etmeye ve ihtiyaçlara göre kaynakları kullanmaya olanak tanır.

5.)Ağ Yönetimi : Kullanıcılar, genellikle sanal ağları yapılandırabilir ve yönetebilirler. Bu, uygulama ve hizmetlere özgü ağ yapılarını destekler.

6.)Güvenlik : IaaS sağlayıcıları, güvenlik önlemleri sağlar ve kullanıcılara çeşitli kimlik doğrulama ve erişim kontrolü araçları sunar.

Amazon EC2 (Elastic Compute Cloud)

Amazon EC2, Amazon Web Services (AWS) tarafından sunulan bir IaaS örneğidir. EC2, kullanıcılara sanal sunucular (EC2 örnekleri) sağlar. Bu örnekler, iş yüklerini barındırmak ve çalıştırmak için kullanılabilir. EC2, bir dizi işletim sistemi, bellek, depolama ve ağ kapasitesi seçeneği sunar.

EC2 kullanıcıları, AWS Management Console veya API'ler aracılığıyla EC2 örneklerini başlatma, durdurma, ölçeklendirme ve yapılandırma gibi işlemleri gerçekleştirebilirler. EC2, özellikle ölçeklenebilir ve esnek bilişim kaynaklarına ihtiyaç duyan uygulamalar için uygun bir seçenek sunar.

Microsoft Azure Virtual Machines

Microsoft Azure Virtual Machines, Microsoft'un bulut hizmet platformu Azure üzerinde çalışan bir IaaS örneğidir. Kullanıcılar, Windows veya Linux tabanlı sanal makineleri seçebilir ve özelleştirebilirler. Azure Portal veya Azure'un komut satırı araçları aracılığıyla sanal makineleri yönetme yeteneği, kullanıcılara esneklik sağlar.

Azure Virtual Machines, yüksek performanslı hesaplama, depolama ve ağ özelliklerini destekler. Ayrıca, iş yüklerini düzenlemek ve ölçeklendirmek için özel sanal makineler oluşturabilir.

Google Compute Engine

Google Compute Engine, Google Cloud Platform'un bir parçası olarak sunulan bir IaaS örneğidir. Kullanıcılar, Google'ın altyapısını kullanarak sanal makineler oluşturabilir ve yönetebilirler. Compute Engine, ölçeklenebilir ve hızlı hesaplama kaynakları sağlar.

Kullanıcılar, iş yüklerini ölçeklendirmek, yüksek performans elde etmek ve özel konfigürasyonlar oluşturmak için Compute Engine'i kullanabilirler. Ayrıca, Compute Engine üzerinde çalışan sanal makineler, diğer Google Cloud hizmetleriyle entegre edilebilir, bu da geniş bir bulut ekosistemine erişim sağlar.

Platform Hizmetleri (PaaS - Platform as a Service)

Platform Hizmetleri (PaaS), bulut bilişim hizmet modelinin bir parçasıdır ve uygulama geliştirme sürecini destekler. PaaS, kullanıcılara uygulama oluşturma, dağıtma ve yönetme süreçlerini kolaylaştıran bir platform sağlar. Bu hizmet modeli, altyapıyı düşünmek zorunda kalmadan geliştiricilere daha fazla odaklanma özgürlüğü sunar.

PaaS hizmetleri genellikle şu temel özelliklere sahiptir:

Uygulama Geliştirme Araçları : PaaS, genellikle entegre geliştirme ortamları (IDE'ler) ve uygulama geliştirme araçları sağlar. Bu araçlar, yazılım geliştirme sürecini kolaylaştırır.

Veritabanı Yönetimi : PaaS, veritabanı yönetimini içerebilir, bu da geliştiricilere veritabanları ile ilgili görevleri kolayca yönetme imkanı tanır.

Otomatik Ölçekleme : Uygulama ihtiyaçlarına bağlı olarak otomatik olarak ölçeklenebilir. Bu, kullanıcılara iş yüklerini etkili bir şekilde yönetme yeteneği sağlar.

Web Sunucuları ve Çerçeveler : PaaS, genellikle önceden yapılandırılmış web sunucuları ve çerçeveler içerir, bu da uygulama geliştiricilerinin web uygulamalarını hızlı bir şekilde oluşturmasına yardımcı olur.

Self-Service Yönetimi : Kullanıcılar genellikle bir kontrol paneli veya API aracılığıyla kendi uygulama ortamlarını yönetebilirler.

Heroku

Heroku, bulut tabanlı bir PaaS hizmetidir ve uygulama geliştirme süreçlerini hızlandırmak için tasarlanmıştır. Heroku'nun temel özellikleri şunlardır:

- Dil Çeşitliliği : Heroku, farklı programlama dillerini destekler, bu da geliştiricilere dil özgü uygulamalar oluşturma esnekliği tanır. - Otomatik Ölçekleme : Heroku, uygulama

taleplerine otomatik olarak yanıt verir ve kaynakları dinamik olarak ölçekler. - Entegrasyonlar : Heroku, popüler veritabanları, araçlar ve hizmetlerle entegre olabilir.

Microsoft Azure App Service

Azure App Service, Microsoft Azure üzerinde çalışan bir PaaS hizmetidir ve uygulama geliştirme süreçlerini kolaylaştırmak için tasarlanmıştır. Temel özellikleri şunlardır:

- Çeşitli Dil Desteği : Azure App Service, .NET, Java, Node.js, Python ve PHP gibi farklı dilleri destekler. - Otomatik Yedekleme ve Güvenlik : Uygulama servisi, otomatik yedekleme ve güvenlik özellikleri içerir, bu da kullanıcılara veri bütünlüğünü ve güvenliği sağlama imkanı tanır. - DevOps Entegrasyonu : App Service, sürekli entegrasyon ve sürekli dağıtım (CI/CD) araçlarıyla entegre edilebilir.

Google App Engine

Google App Engine, Google Cloud Platform'un bir parçası olarak sunulan bir PaaS hizmetidir ve uygulama geliştirme süreçlerini basitleştirmeyi amaçlar. Başlıca özellikleri şunlardır:

- Otomatik Ölçekleme : App Engine, uygulama trafiğine göre otomatik olarak ölçeklenir, bu da performansı optimize etmeye yardımcı olur. - Çeşitli Dil Desteği : Python, Java, Go, Node.js, Ruby ve .NET gibi farklı dilleri destekler. - Veri Depolama : App Engine, Google Cloud Datastore'u kullanarak uygulama verilerini depolar ve yönetir.

Yazılım Hizmetleri (SaaS - Software as a Service)

Yazılım Hizmetleri (SaaS), bulut bilişim hizmet modelinin bir parçasıdır ve kullanıcılara internet üzerinden erişilebilen hazır yazılım uygulamalarını sunar. Bu hizmet modeli, kullanıcıların yazılım satın almak yerine abonelik tabanlı olarak kullanmalarını sağlar. SaaS, genellikle işbirliği, ofis üretkenliği, müşteri ilişkileri yönetimi (CRM), içerik yönetimi ve daha birçok alanda çeşitli uygulamalar içerir.

SaaS hizmetlerinin temel özellikleri şunlardır:

Erişim Kolaylığı : SaaS uygulamalarına internet tarayıcıları aracılığıyla herhangi bir cihazdan kolayca erişilebilir.

Güncelleme ve Bakım : Yazılım sağlayıcıları, SaaS uygulamalarını düzenli olarak günceller ve bakımını yapar, bu da kullanıcıların güncel ve güvenli bir yazılım sürümüne erişmelerini sağlar.

Pay-as-You-Go Modeli : Kullanıcılar genellikle kullandıkları özelliklere veya lisanslara göre ödeme yaparlar, bu da maliyetleri daha yönetilebilir hale getirir.

Çoklu Kullanıcı Desteği : SaaS, genellikle çoklu kullanıcı desteği sunar, bu da ekiplerin aynı uygulamayı birlikte kullanmalarını sağlar.

Özelleştirilebilirlik : SaaS uygulamaları, kullanıcılara belirli ayarları ve özellikleri özelleştirme imkanı tanır.

Google Workspace (G Suite)

Google Workspace, işbirliği ve ofis üretkenliği araçlarını bir araya getiren bir SaaS paketidir. Temel özellikleri şunlardır:

- E-posta ve Takvim : Gmail ve Google Takvim, kullanıcılara güçlü e-posta iletişimi ve zaman yönetimi imkanı sağlar. - Belge Oluşturma ve Paylaşma : Google Dokümanlar, Çalışma Sayfaları ve Sunumlar, çevrimiçi belge oluşturma ve paylaşma yetenekleri sunar. - İşbirliği Araçları : Google Meet, Drive ve Chat, ekiplerin çevrimiçi işbirliği yapmalarını kolaylaştırır.

Microsoft 365

Microsoft 365, ofis uygulamalarını ve bulut hizmetlerini birleştiren bir SaaS paketidir. Başlıca özellikleri şunlardır:

- Ofis Uygulamaları : Word, Excel, PowerPoint ve Outlook gibi ofis uygulamalarını içerir. - OneDrive : Dosyaları depolama, paylaşma ve işbirliği yapma olanağı sunar. - Teams : Ekiplerin çevrimiçi toplantılar yapmalarını, sohbet etmelerini ve dosya paylaşmalarını sağlar.

Salesforce

Salesforce, müşteri ilişkileri yönetimi (CRM) alanında öncü bir SaaS sağlayıcısıdır. Temel özellikleri şunlardır:

- Müşteri İlişkileri Yönetimi : Satış, pazarlama ve müşteri hizmetleri için kapsamlı bir platform sağlar. - Özelleştirilebilir Raporlar ve Analizler : İşletmelerin performanslarını izlemeleri ve değerlendirmeleri için raporlama araçları sunar. - Bulut Tabanlı Uygulama Geliştirme : Salesforce, özel uygulamalar oluşturmak için güçlü bir bulut tabanlı geliştirme platformu sunar.

Depolama ve Veritabanı Hizmetleri

Depolama ve veritabanı hizmetleri, bulut teknolojisinin önemli bir bileşenidir ve kullanıcılara veri depolama, paylaşma ve yönetme imkanı sunar. Bu hizmetler, genellikle büyük veri setlerini güvenli bir şekilde saklamak ve işlemek için kullanılır. İşte bu kategorideki bazı önemli hizmetler:

Amazon S3 (Simple Storage Service)

Amazon S3, Amazon Web Services (AWS) tarafından sunulan bir nesne depolama hizmetidir. Temel özellikleri şunlardır:

- Ölçeklenebilir Depolama : Kullanıcılar, büyük miktarda veriyi güvenli bir şekilde depolayabilir ve gerektiğinde ölçekleyebilir. - Veri Yedekleme ve Kurtarma : Veriler, birden çok bölgeye otomatik olarak yedeklenir ve kurtarma işlemleri kolayca gerçekleştirilebilir.

Microsoft Azure Blob Storage

Azure Blob Storage, Microsoft Azure bulut platformu üzerinde bulunan bir nesne depolama hizmetidir. Başlıca özellikleri şunlardır:

- Çoklu Depolama Sınıfı : Verilere farklı performans ve maliyet seviyelerinde erişim sağlayan çeşitli depolama sınıfları sunar. - Geniş Entegrasyon : Azure Blob Storage, birçok Microsoft ve üçüncü taraf uygulamasıyla entegre olabilir.

Google Cloud Storage

Google Cloud Storage, Google Cloud Platform tarafından sağlanan ölçeklenebilir ve güvenli bir nesne depolama hizmetidir. Temel özellikleri şunlardır:

- İnteraktif ve Düşük Gecikme Süresi : Verilere hızlı ve etkileşimli erişim sağlar, düşük gecikme süresi sunar. - Veri Transferi ve Paylaşım : Veriler, kullanıcılar tarafından belirlenen izinlere göre güvenli bir şekilde paylaşılabilir ve transfer edilebilir.

Amazon RDS (Relational Database Service)

Amazon RDS, AWS tarafından sağlanan yönetilen bir ilişkisel veritabanı hizmetidir. Başlıca özellikleri şunlardır:

Çeşitli Veritabanı Motorları : MySQL, PostgreSQL, Oracle, Microsoft SQL Server gibi farklı veritabanı motorlarını destekler. Otomatik Yedekleme ve Güvenlik : Veritabanı yedekleme işlemleri otomatiktir ve güvenlik duvarları ile erişim kontrolü sağlanır.

Microsoft Azure SQL Database

Azure SQL Database, Microsoft Azure üzerinde çalışan ve yönetilen bir SQL veritabanı hizmetidir. Temel özellikleri şunlardır:

- Elastik Ölçeklendirme : İhtiyaca göre kaynakları otomatik olarak ölçeklendirme imkanı sunar. - Yüksek Kullanılabilirlik : Otomatik yedekleme ve felaket kurtarma özellikleri ile yüksek kullanılabilirlik sağlar.

Google Cloud SQL

Google Cloud SQL, Google Cloud Platform üzerinde çalışan, ölçeklenebilir ve yönetilen bir SQL veritabanı hizmetidir. Başlıca özellikleri şunlardır:

- İleri Düzey Güvenlik : Veritabanları, kullanıcı erişim kontrolleri ve şifreleme ile güvence altına alınır. - Otomatik Yedekleme ve Veri Repikasyonu : Otomatik yedekleme işlemleri ve veri repikasyonu özellikleri sunarak veri kaybını en aza indirir.

Bu depolama ve veritabanı hizmetleri, kuruluşların verilerini güvenli, ölçeklenebilir ve erişilebilir bir şekilde yönetmelerine olanak tanır.

Ağ Hizmetleri

Bulut teknolojisinin ağ hizmetleri, kullanıcılara sanal ağlar oluşturma, yönetme ve güvenli bir şekilde bağlantı kurma imkanı sağlar. İşte bu kategorideki önemli hizmetler:

Amazon VPC (Virtual Private Cloud)

Amazon VPC, AWS'nin sanal özel bulut hizmetidir ve kullanıcılara özelleştirilebilir sanal ağlar oluşturma imkanı tanır. Temel özellikleri şunlardır:

- İzole Ağ Ortamı : Kullanıcılar, kendi sanal özel ağlarını oluşturarak kaynaklarını izole edebilir. - Alt Ağlar ve Yapılandırma : Ağlarını alt ağlara bölebilir ve özelleştirilebilir IP adresi aralıkları belirleyebilirler.

Microsoft Azure Virtual Network

Azure Virtual Network, Azure üzerinde kaynakları birbirine bağlamak ve güvenli bir ağ ortamı sağlamak için kullanılır. Başlıca özellikleri şunlardır:

- Site-to-Site VPN ve Uzak Erişim : Şirket içi ağlar ile güvenli bağlantılar kurabilir ve uzaktan erişim sağlayabilir. - Ağ Güvenliği Duvarları (NSG) : Trafik akışını kontrol etmek için ağ güvenliği duvarları konfigüre edebilir.

Google Cloud Virtual Private Cloud

Google Cloud Virtual Private Cloud (VPC), Google Cloud Platform'un bir parçası olarak sunulan sanal özel bulut hizmetidir. Temel özellikleri şunlardır:

- Özelleştirilebilir IP Adres Aralıkları : Kullanıcılar, kendi IP adres aralıklarını belirleyerek özelleştirilebilir sanal ağlar oluşturabilir. - İleri Düzey Ağ Yönetimi : Ağ politikalarını ve yönlendirmeleri özelleştirebilir, yüksek performanslı bağlantılar kurabilir.

Bu ağ hizmetleri, kuruluşların esnek, güvenli ve yönetilebilir bir ağ altyapısı oluşturarak farklı kaynakları birbirine bağlamalarına ve kullanmalarına olanak tanır.

Büyük Veri ve Analitik Hizmetleri

Bulut teknolojisinin büyük veri ve analitik hizmetleri, büyük miktardaki veriyi depolama, işleme ve analiz etme imkanı sağlar. İşte bu kategorideki önemli hizmetler:

Amazon EMR (Elastic MapReduce)

Amazon EMR, AWS üzerinde büyük veri işleme ve analizi için kullanılan bir hizmettir. Temel özellikleri şunlardır:

- Ölçeklenebilir Hadoop ve Spark : Apache Hadoop ve Apache Spark gibi büyük veri işleme araçlarına ölçeklenebilir bir şekilde erişim sağlar. - Yönetilen Küme : Otomatik ölçeklendirme, hata tolere edebilirlik ve yönetilen Hadoop kümesi sağlar.

Microsoft Azure HDInsight

Azure HDInsight, Apache Spark, Hadoop, HBase gibi açık kaynaklı büyük veri teknolojilerini destekleyen bir hizmettir. Başlıca özellikleri şunlardır:

- Çeşitli Büyük Veri Araçları : Apache Spark, Apache Hive, Apache HBase gibi farklı büyük veri araçlarına destek sağlar. - Yüksek Güvenlik Standartları : Güvenlik ve kimlik yönetimi için entegre araçlar içerir.

Google Cloud Dataproc

Google Cloud Dataproc, Google Cloud Platform üzerinde çalışan büyük veri işleme hizmetidir. Temel özellikleri şunlardır:

- Yönetilen Spark ve Hadoop : Spark ve Hadoop üzerine kurulu büyük veri çözümlerini yönetir ve ölçeklendirir. - Özel Küme Yapılandırması : Kullanıcılar, ihtiyaçlarına uygun özel büyük veri küme yapılandırmalarını oluşturabilir.

BigQuery

Google Cloud Platform'un bir parçası olan BigQuery, tamamen yönetilen bir büyük veri analitik veritabanıdır. Başlıca özellikleri şunlardır:

- Serverless Veritabanı Hizmeti : Kullanıcılar, altyapı yönetimi olmadan büyük veri analizi yapabilirler. - Hızlı SQL Sorguları : Yüksek performanslı SQL sorguları ile büyük veri kümelerini hızlı bir şekilde analiz eder.

Bu hizmetler, büyük veri setlerini etkili bir şekilde yönetme ve analiz etme ihtiyacı olan kuruluşlar için güçlü ve ölçeklenebilir çözümler sunar.

Kimlik Yönetimi ve Güvenlik

Bulut teknolojisinin kimlik yönetimi ve güvenlik hizmetleri, kullanıcıların bulut hizmetlerine güvenli bir şekilde erişimini sağlamak ve bu hizmetleri güvence altına almak için tasarlanmıştır. İşte bu kategorideki önemli hizmetler:

AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM), AWS üzerinde kimlik ve erişim yönetimi için kullanılan bir hizmettir. Başlıca özellikleri şunlardır:

- Rol Bazlı Erişim Kontrolü : Kullanıcılara ve kaynaklara erişimi, roller ve politikalar üzerinden yönetir. - Gelişmiş Güvenlik Ayarları : Kimlik doğrulama, çok faktörlü kimlik doğrulama (MFA) gibi güvenlik ayarlarını içerir.[39][40]

Fig. 12. AWS

Microsoft Azure Active Directory

Azure Active Directory (Azure AD), kimlik yönetimi ve tek oturum açma (SSO) sağlayan bir hizmettir. Temel özellikleri şunlardır:

- Kimlik Federasyonu : Farklı hizmetler arasında tek bir oturum açma (SSO) deneyimi sağlar. - Gelişmiş Güvenlik Özellikleri : Tehdit koruması, kimlik hırsızlığı algılama gibi güvenlik özelliklerini içerir.

Google Cloud Identity and Access Management

Google Cloud Identity and Access Management, Google Cloud Platform üzerinde kimlik ve erişim yönetimi için kullanılan bir hizmettir. Temel özellikleri şunlardır:

- İlkesel Erişim Kontrolü : Kullanıcıların ve kaynakların erişimini ilkesel olarak yönetir. - Audit ve İzleme : Erişim olaylarını izleme ve denetleme imkanı sağlar.

Bu hizmetler, bulut tabanlı altyapıların güvenliğini artırmak ve yetkilendirme süreçlerini etkili bir şekilde yönetmek isteyen organizasyonlar için önemli araçlar sunar.

Geliştirme Araçları

Bulut teknolojisinin geliştirme araçları, yazılım geliştirme süreçlerini desteklemek ve bulut ortamında uygulamaların kolayca oluşturulmasını sağlamak amacıyla kullanılır. İşte bu kategorideki önemli araçlar:

AWS Cloud9

AWS Cloud9, bulut üzerinde entegre bir geliştirme ortamı (IDE) sağlayan bir hizmettir. Temel özellikleri şunlardır:

- Çevrimiçi İşbirliği : Bir ekip halinde aynı projede çalışmayı kolaylaştırır. - Hızlı Prototip Oluşturma : Hızlı bir şekilde kod yazma ve prototip oluşturma imkanı sunar.

Microsoft Azure DevOps

Azure DevOps, yazılım geliştirme süreçlerini destekleyen bir dizi araç ve hizmet içeren bir platformdur. Başlıca özellikleri şunlardır:

- Sürüm Kontrolü : Kodun sürüm geçmişini yönetme imkanı sağlar. - Sürekli Entegrasyon ve Dağıtım (CI/CD) : Otomatik test ve dağıtım süreçlerini kolaylaştırır.

Google Cloud SDK

Google Cloud SDK, bulut kaynaklarıyla etkileşimde bulunmak ve bulut tabanlı uygulamaları yönetmek için kullanılan bir dizi komut satırı aracıdır. Temel özellikleri şunlardır:

- Komut Satırı Arayüzü : Bulut kaynaklarına komut satırından erişim sağlar. - API Entegrasyonu : Google Cloud Platform'un sunduğu API'ları kullanmayı kolaylaştırır.

Bu araçlar, geliştiricilere bulut ortamında daha verimli çalışma ve yazılım projelerini daha etkili bir şekilde yönetme imkanı sunar.[41][42]

References

[1] https://aws.amazon.com/tr/devops/what-is-devops/
[2] https://www.oracle.com/tr/devops/what-is-devops/
[3] https://medium.com/devopsturkiye/devops-nedir-9f7e39b63b9c
[4] https://www.youtube.com/watch?v=XV-RR0hDsI8
[5] https://medium.com/@nagihankivanc/devsecops-kavramı-e78bb011c964
[6] https://www.argenova.com.tr/devsecops-nedir
[7] https://www.linkedin.com/posts/saiqa-jutt-9b9547237_how-much-linux-knowledge-is-required-to-become-activity-7077479491568119808-6J4T/?trk
[8] https://www.quora.com/How-much-Linux-knowledge-is-required-to-become-a-DevOps-engineer
[9] https://developer.hashicorp.com/vagrant/intro
[10] https://tr.wikipedia.org/wiki/OSI_modeli
[11] https://yakupseker.medium.com/osi-modeli-nedir-3f917dd8e65f
[12] https://aws.amazon.com/tr/what-is/osi-model/
[13] https://github.com/ozgurozturknet/AdanZyeDocker
[14] https://medium.com/@ismailsozen/container-konteyner-teknolojisi-nedir-e5ade9b9478c
[15] https://aws.amazon.com/tr/docker/
[16] https://coderspace.io/blog/git-nedir-nasil-kullanilir/
[17] https://www.oracle.com/tr/cloud/cloud-native/container-engine-kubernetes/what-is-kubernetes/
[18] https://bulutistan.com/blog/kubernetes-nedir/
[19] https://aws.amazon.com/tr/compare/the-difference-between-monolithic-and-microservices-architecture/
[20] https://github.com/ozgurozturknet/AdanZyeDocker
[21] https://medium.com/devopsturkiye/continuous-integration-ve-continuous-delivery-nedir-neden-kullanılmalı-9f5ce57f200e
[22] https://www.datamarket.com.tr/sozluk/ci-cd/
[23] https://medium.com/bilişim-hareketi/jenkins-nedir-neden-kullanılır-17e5a207d009
[24] https://coderspace.io/sozluk/jenkins-nedir
[25] https://kerteriz.net/jenkins-nedir-kurulumu-ve-ci-cd-surec-ornegi/
[26] https://medium.com/devopsturkiye/terraform-nedir-infrastructure-as-code-nedir-2-bff310cd5782
[27] https://kerteriz.net/terraform-nedir/
[28] https://www.moradam.com/20180211209065/terraform-nedir/
[29] https://medium.com/devopsturkiye/ansible-nedir-dosya-yapısı-nasıldır-nasıl-kullanılır-4d8c90cdb266
[30] https://medium.com/türk-telekom-bulut-teknolojileri/ansiblea-giriş-fbfb49690142
[31] https://sezer.in/ansible-ile-sistemlerin-otomatik-yonetimi/
[32] https://www.yusufsezer.com.tr/maven/
[33] https://medium.com/yazilim-vip/bu-yazının-amacı-maven-ile-siz-okurları-tanıştırmaktır-aab0f6ff91f4
[34] https://emrecanayar.com/2021/05/04/maven-nedir/
[35] https://www.serdarbayram.net/systemnetwork-izlememonitoring-nasil-yapilir.html
[36] https://www.mediaclick.com.tr/blog/monitoring-nedir/
[37] https://sudo.ubuntu-tr.net/puppet
[38] https://medium.com/devopsturkiye/puppet-nedir/
[39] https://turk.net/blog/bulut-bilisim-cloud-computing-nedir/
[40] https://www.endustri40.com/bulut-bilisim-cloud-computing-nedir/
[41] https://www.oracle.com/tr/cloud/what-is-cloud-computing/
[42] https://www.milleni.com.tr/blog/teknoloji/bulut-teknolojisi