**DOCKER BASICS**

# What is Docker? | How Docker Works? | Why Docker? | Microservices | DevOps Tools | KnowledgeHut

[**https://www.youtube.com/watch?v=zfclTRtKOSM&t=25s**](https://www.youtube.com/watch?v=zfclTRtKOSM&t=25s)

In this lesson we will cover;

* + - * What is Docker?
      * Docker Architecture
      * Docker vs. VMs
      * Terminology

**INTRODUCTION**

**What is Docker? [Docker nedir?]**

💡  *Wikipedia defines Docker as*  *an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux. [💡  Wikipedia, Docker'ı, Linux'ta işletim sistemi düzeyinde sanallaştırmanın ek bir soyutlama katmanı ve otomasyonu sağlayarak, yazılım uygulamalarının kaplar içinde dağıtımını otomatikleştiren açık kaynaklı bir proje olarak tanımlar.]*

"DOCKER" refers to several things. ['DOCKER' birkaç şeyi ifade eder.] This includes an open-source community project which started in 2013; tools from the open-source project; Docker Inc., the company that is the primary supporter of that project; and the tools that the company formally supports. [Bu, 2013'te başlayan bir açık kaynaklı topluluk projesini; açık kaynak projesinden araçlar; Bu projenin ana destekçisi olan Docker Inc.; ve şirketin resmi olarak desteklediği araçlar.]

**💡Tips: From Wikipedia [💡İpuçları: Wikipedia'dan]**

Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources. [Bilgi işlemde sanallaştırma, sanal bilgisayar donanım platformu, işletim sistemi (OS), depolama aygıtı veya bilgisayar ağ kaynakları dahil ancak bunlarla sınırlı olmamak üzere bir şeyin (gerçek yerine) sanal bir sürümünü oluşturma eylemini ifade eder.]



* Docker is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called containers) to run on the host operating system i.e. [Docker, geliştiricilerin, sistem yöneticilerinin vb. uygulamalarını ana bilgisayar işletim sisteminde, yani ana bilgisayar işletim sisteminde çalıştırmak için bir sanal alanda (konteyner adı verilen) kolayca dağıtmalarına olanak tanıyan bir araçtır.] Linux. [Linux.]
* The key benefit of Docker is that it allows users to package an application with all of its dependencies into a standardized unit for software development. [Docker'ın en önemli yararı, kullanıcıların bir uygulamayı tüm bağımlılıklarıyla birlikte yazılım geliştirme için standart bir birime paketlemesine olanak sağlamasıdır.] Unlike virtual machines, containers do not have high overhead and hence enable more efficient usage of the underlying system and resources. [Sanal makinelerin aksine, kapsayıcılar yüksek ek yüke sahip değildir ve bu nedenle temel alınan sistem ve kaynakların daha verimli kullanılmasını sağlar.]

***Here's a quick explanation: [İşte hızlı bir açıklama:]***

* The IT software "Docker” is containerization technology that enables the creation and use of Linux® containers. [BT yazılımı 'Docker', Linux® kapsayıcılarının oluşturulmasını ve kullanılmasını sağlayan kapsayıcı teknolojisidir.]
* The open source Docker community works to improve these technologies to benefit all users [Açık kaynaklı Docker topluluğu, bu teknolojileri tüm kullanıcılara fayda sağlayacak şekilde geliştirmek için çalışır.]
* The company, Docker Inc., builds on the work of the Docker community, makes it more secure, and shares those advancements back to the greater community. [Docker Inc. şirketi, Docker topluluğunun çalışmalarını temel alır, daha güvenli hale getirir ve bu gelişmeleri daha büyük toplulukla paylaşır.]

**💡Tips: [💡İpuçları:]**

With DOCKER, you can treat containers like extremely lightweight, modular virtual machines. [DOCKER ile kapsayıcılara son derece hafif, modüler sanal makineler gibi davranabilirsiniz.] And you get flexibility with those containers—you can create, deploy, copy, and move them from environment to environment, which helps optimize your apps for the cloud. [Ve bu kapsayıcılarla esneklik elde edersiniz; bunları oluşturabilir, dağıtabilir, kopyalayabilir ve ortamdan ortama taşıyabilirsiniz, bu da uygulamalarınızı bulut için optimize etmenize yardımcı olur.]



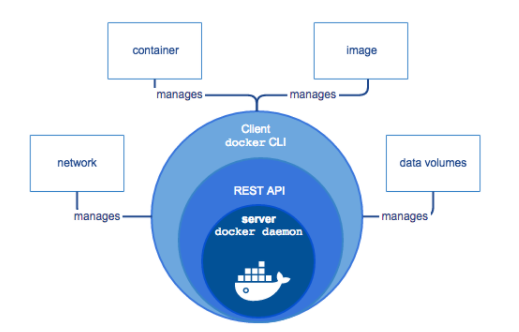
Q: What is a Docker? [S: Docker nedir?]  
A: **Docker** is defined as the platform for containerizing the applications to isolate it from each other in order to ensure high availability and more efficiency irrespective of the environments such as **Development, Testing or Production**. [A: Docker, Geliştirme, Test veya Üretim gibi ortamlardan bağımsız olarak yüksek kullanılabilirlik ve daha fazla verimlilik sağlamak için uygulamaları birbirinden izole etmek için kapsayıcı hale getirme platformu olarak tanımlanır.] All the application related dependencies such as libraries, jar files, server related configurations, infrastructure-related elements will be packaged and formed as a container called containerized applications which does not need any dependency and works independently. [Kütüphaneler, jar dosyaları, sunucu ile ilgili konfigürasyonlar, altyapı ile ilgili öğeler gibi uygulama ile ilgili tüm bağımlılıklar, herhangi bir bağımlılığa ihtiyaç duymayan ve bağımsız çalışan containerized uygulamalar adı verilen bir kapsayıcı olarak paketlenecek ve oluşturulacaktır.] It ensures the application to be run irrespective of the external factors. [Uygulamanın dış etkenlerden bağımsız olarak çalışmasını sağlar.] Containers in Docker have support from Docker Engine and Host Operating System to support all the operational or infrastructural related dependencies. [Docker'daki kapsayıcılar, operasyonel veya altyapıyla ilgili tüm bağımlılıkları desteklemek için Docker Engine ve Host İşletim Sisteminden destek alır.]

 - Interview Q&A

### Docker Engine [liman işçisi motoru]

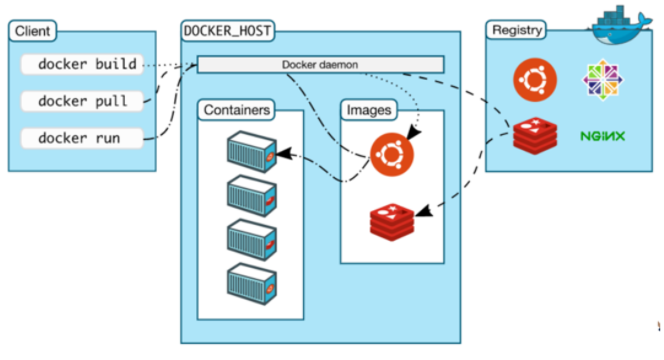
Docker Engine is a client-server application with these major components: [Docker Engine, şu ana bileşenlere sahip bir istemci-sunucu uygulamasıdır:]

* A server which is a type of long-running program called a daemon process (the dockerd command). [Daemon process (dockerd komutu) adı verilen uzun süre çalışan bir program türü olan bir sunucu.]
* A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do. [Programların arka plan programıyla konuşmak ve ona ne yapması gerektiğini bildirmek için kullanabileceği arabirimleri belirten bir REST API.]
* A command-line interface (CLI) client (the docker command). [Bir komut satırı arabirimi (CLI) istemcisi (docker komutu).]
* The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. [CLI, komut dosyası oluşturma veya doğrudan CLI komutları aracılığıyla Docker arka plan programını kontrol etmek veya onunla etkileşim kurmak için Docker REST API'sini kullanır.]



### Docker Architecture [Docker Mimarisi]

Docker uses a client-server architecture. [Docker, bir istemci-sunucu mimarisi kullanır.] The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. [Docker istemcisi, Docker kapsayıcılarınızı oluşturmanın, çalıştırmanın ve dağıtmanın ağır işlerini yapan Docker arka plan programıyla konuşur.] The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. [Docker istemcisi ve arka plan programı aynı sistemde çalışabilir veya bir Docker istemcisini uzak bir Docker arka plan programına bağlayabilirsiniz.] The Docker client and daemon communicate using a REST API, over UNIX sockets, or a network interface. [Docker istemcisi ve arka plan programı, UNIX yuvaları veya bir ağ arabirimi üzerinden bir REST API kullanarak iletişim kurar.]



💡  *The core components that compose Docker: [💡  Docker'ı oluşturan temel bileşenler:]*

* + - *The Docker client and server also called the Docker Engine. [Docker istemcisi ve sunucusu, Docker Engine olarak da adlandırılır.]*
    - *Docker Images [Docker Görselleri]*
    - *Registries [kayıtlar]*
    - *Docker Containers [Docker Konteynerleri]*

**Docker Client and Server [Docker İstemcisi ve Sunucusu]**

Docker is a client-server application. [Docker bir istemci-sunucu uygulamasıdır.] The Docker client talks to the Docker server or daemon, which, in turn, does all the work. [Docker istemcisi, sırayla tüm işi yapan Docker sunucusu veya arka plan programı ile konuşur.] You’ll also sometimes see the Docker daemon called the Docker Engine. [Ayrıca bazen Docker Engine adlı Docker arka plan programını da görürsünüz.]

**Docker images [liman işçisi görüntüleri]**

Images are the building blocks of the Docker world. [Görüntüler, Docker dünyasının yapı taşlarıdır.] Containers are launched from images. [Konteynerler görüntülerden başlatılır.] Docker images are light-weight, portable, reproduce-able, and declarative. [Docker görüntüleri hafif, taşınabilir, yeniden üretilebilir ve bildirimseldir.]

**Registries [kayıtlar]**

Docker stores the images you build in registries. [Docker, oluşturduğunuz görüntüleri kayıtlarda saklar.] There are two types of registries: public and private. [İki tür kayıt vardır: genel ve özel.] Docker, Inc., operates the public registry for images, called the Docker Hub. [Docker, Inc., görüntüler için Docker Hub adı verilen genel kayıt defterini işletir.] You can create an account on the Docker Hub and use it to share and store your images. [Docker Hub'da bir hesap oluşturabilir ve bunu görüntülerinizi paylaşmak ve depolamak için kullanabilirsiniz.] You can also re-use the registry repository capability for free at your premises to store your images. [Ayrıca, görüntülerinizi depolamak için kayıt defteri havuzu özelliğini tesislerinizde ücretsiz olarak yeniden kullanabilirsiniz.]

**Containers [konteynerler]**

Docker helps you build and deploy containers that encapsulate your packages, applications, infrastructures, and services. [Docker, paketlerinizi, uygulamalarınızı, altyapılarınızı ve hizmetlerinizi kapsayan kapsayıcılar oluşturmanıza ve dağıtmanıza yardımcı olur.] Docker containers try to maximize resource sharing between containers and allow us to isolate those parts of the applications that are different and need their own space. [Docker kapsayıcıları, kapsayıcılar arasındaki kaynak paylaşımını en üst düzeye çıkarmaya çalışır ve uygulamaların farklı olan ve kendi alanlarına ihtiyaç duyan kısımlarını izole etmemize izin verir.]

Because of **high resource sharing**, we can build Docker containers with a **small footprint** that makes application **distribution easier** and it makes container **startup times faster**. [Yüksek kaynak paylaşımı nedeniyle, uygulama dağıtımını kolaylaştıran ve kapsayıcı başlatma sürelerini hızlandıran küçük bir ayak izine sahip Docker kapsayıcıları oluşturabiliriz.]

Q: What are the components of Docker Architecture and explain? [S: Docker Mimarisinin bileşenleri nelerdir ve açıklayınız?]  
A: The Docker works on a client-server architecture. [A: Docker, bir istemci-sunucu mimarisi üzerinde çalışır.] The Docker client establishes communication with the Docker Daemon. [Docker istemcisi, Docker Daemon ile iletişim kurar.] The Docker client and Daemon can run on the same system. [Docker istemcisi ve Daemon aynı sistem üzerinde çalışabilir.] A Docker client can also be connected to a remote Docker Daemon. [Bir Docker istemcisi, uzak bir Docker Daemon'a da bağlanabilir.]

* **Docker Client:** This performs Docker build pull and run operations to establish communication with the Docker Host. [Docker İstemcisi: Bu, Docker Host ile iletişim kurmak için Docker derleme çekme ve çalıştırma işlemlerini gerçekleştirir.] The Docker command uses Docker API to call the queries to be run. [Docker komutu, çalıştırılacak sorguları çağırmak için Docker API'sini kullanır.]
* **Docker Host:** This component contains Docker Daemon, Containers and its images. [Docker Host: Bu bileşen Docker Daemon, Containers ve görüntülerini içerir.] The images will be the kind of metadata for the applications which are containerized in the containers. [Görüntüler, kapsayıcılarda kapsanan uygulamalar için bir tür meta veri olacaktır.] The Docker Daemon establishes a connection with Registry. [Docker Daemon, Registry ile bağlantı kurar.]
* **Registry:**This component will be storing the Docker images. [Kayıt Defteri: Bu bileşen, Docker görüntülerini depolayacaktır.] The public registries are Docker Hub and Docker Cloud which can be used by anyone. [Herkese açık kayıtlar, herkes tarafından kullanılabilen Docker Hub ve Docker Cloud'dur.]

 Interview Q&A

### Docker vs. [Docker vs.] VM's [vm'ler]

#### **What are VMs? [VM'ler nedir?]**

A virtual machine (VM) is an emulation of a computer system. [Sanal makine (VM), bir bilgisayar sisteminin öykünmesidir.] Put simply, it makes it possible to run what appears to be many separate computers on hardware that is actually one computer. [Basitçe söylemek gerekirse, aslında bir bilgisayar olan donanım üzerinde birçok ayrı bilgisayar gibi görünenleri çalıştırmayı mümkün kılar.]

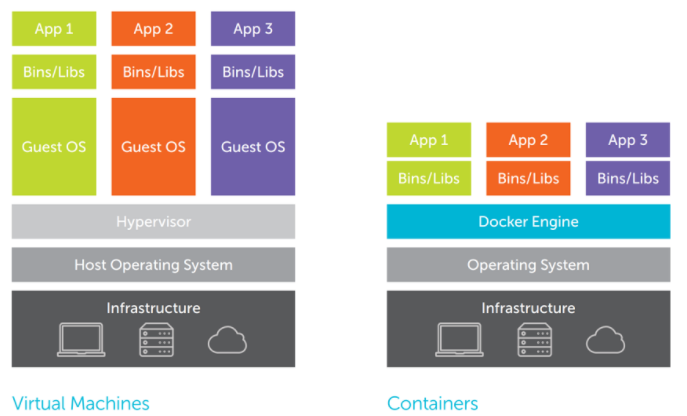
#### **Difference Between VM and Docker [VM ve Docker Arasındaki Fark]**

Virtual machines have a full OS with its own memory management installed with the associated overhead of virtual device drivers. [Sanal makineler, ilgili sanal aygıt sürücülerinin ek yüküyle birlikte kurulu kendi bellek yönetimine sahip tam bir işletim sistemine sahiptir.] In a virtual machine, valuable resources are emulated for the guest OS and hypervisor, which makes it possible to run many instances of one or more operating systems in parallel on a single machine (or host). [Bir sanal makinede, konuk işletim sistemi ve hiper yönetici için değerli kaynaklar öykünür, bu da bir veya daha fazla işletim sisteminin birçok örneğini tek bir makinede (veya ana bilgisayarda) paralel olarak çalıştırmayı mümkün kılar.] Every guest OS runs as an individual entity from the host system. [Her konuk işletim sistemi, ana bilgisayar sisteminden bağımsız bir varlık olarak çalışır.] Hence, we can look at it an independent full-fledged house where we don't share any resources as shown below: [Dolayısıyla, aşağıda gösterildiği gibi, hiçbir kaynağı paylaşmadığımız bağımsız, tam teşekküllü bir ev olarak bakabiliriz:]



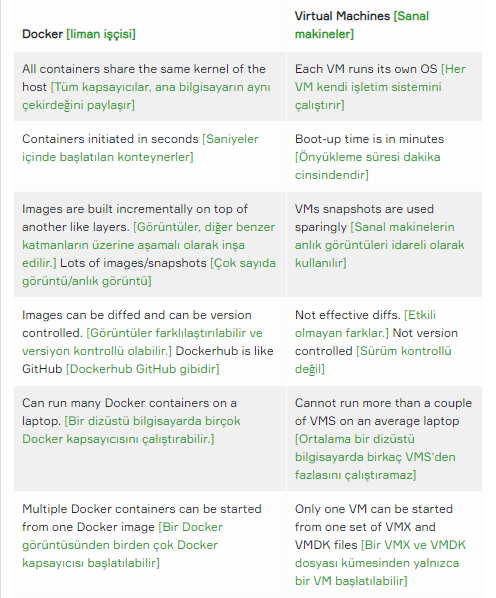
Docker containers are executed with the Docker engine rather than the hypervisor. [Docker kapsayıcıları, hiper yönetici yerine Docker motoruyla yürütülür.] Containers are therefore smaller than Virtual Machines and enable **faster startup** with **better performance**, **less isolation** and **greater compatibility** possible due to **sharing** of the host’s kernel. [Konteynerler bu nedenle Sanal Makinelerden daha küçüktür ve daha iyi performans, daha az izolasyon ve ana bilgisayarın çekirdeğinin paylaşılması nedeniyle daha fazla uyumluluk ile daha hızlı başlatma sağlar.] Hence, it looks very similar to the residential flats system where we share resources of the building. [Dolayısıyla binanın kaynaklarını paylaştığımız konut daire sistemine çok benziyor.]

**💡Tips: [💡İpuçları:]**  
  *From Wikipedia:* A hypervisor (or virtual machine monitor, VMM) is computer software, firmware or hardware that creates and runs virtual machines. [Wikipedia'dan: Bir hiper yönetici (veya sanal makine monitörü, VMM), sanal makineleri oluşturan ve çalıştıran bilgisayar yazılımı, bellenim veya donanımdır.] A computer on which a hypervisor runs one or more virtual machines is called a host machine, and each virtual machine is called a guest machine. [Bir hipervizörün üzerinde bir veya daha fazla sanal makine çalıştırdığı bir bilgisayara ana makine, her sanal makineye de konuk makine denir.] The hypervisor presents the guest operating systems with a virtual operating platform and manages the execution of the guest operating systems. [Hiper yönetici, konuk işletim sistemlerine sanal bir işletim platformu sunar ve konuk işletim sistemlerinin yürütülmesini yönetir.] Multiple instances of a variety of operating systems may share the virtualized hardware resources. [Çeşitli işletim sistemlerinin birden çok örneği, sanallaştırılmış donanım kaynaklarını paylaşabilir.]



Virtual Machines are built over the physical hardware, there is a layer of Hypervisor which sits between physical hardware and operating systems. [Sanal Makineler fiziksel donanım üzerine kuruludur, fiziksel donanım ve işletim sistemleri arasında oturan bir Hiper Yönetici katmanı vardır.] In a broader view, Hypervisor is used to virtualize the hardware which is then configured with the way a user wants it to. [Daha geniş bir görünümde, Hiper Yönetici, daha sonra kullanıcının istediği şekilde yapılandırılan donanımı sanallaştırmak için kullanılır.] Unlike virtual machines where hypervisor divides physical hardware into parts, Containers are like normal operating system processes. [Hipervizörün fiziksel donanımı parçalara böldüğü sanal makinelerin aksine, Konteynerler normal işletim sistemi süreçleri gibidir.]

Docker Containers versus Virtual Machines (VM) [Docker Konteynerleri ve Sanal Makineler (VM)]

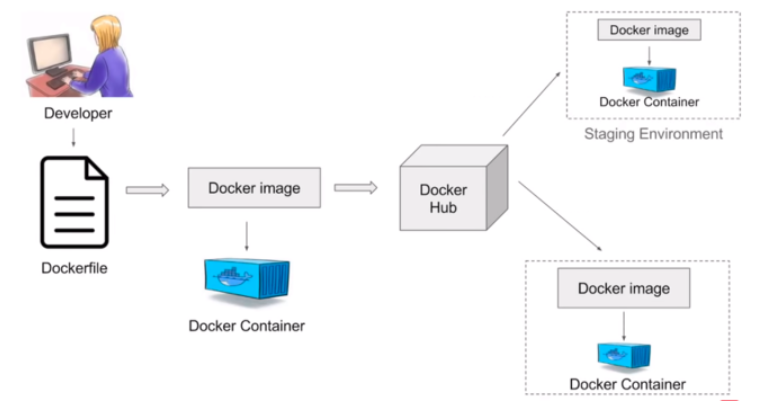


Q: What is the advantage of Docker over hypervisors? [S: Docker'ın hipervizörlere göre avantajı nedir?]  
A: Docker is light weight and more efficient in terms of resource uses because it uses the host underlying kernel rather than creating its own hypervisor. [C: Docker hafiftir ve kaynak kullanımı açısından daha verimlidir, çünkü kendi hipervizörünü oluşturmak yerine ana makinenin altında yatan çekirdeği kullanır.]

 Interview Q&A

**Terminology [terminoloji]**

In this course, we will use a lot of Docker-specific jargon which might be confusing to some. [Bu kursta, bazıları için kafa karıştırıcı olabilecek birçok Docker'a özgü jargon kullanacağız.] So before we go further, we need to clarify some terminology that is used frequently in the Docker ecosystem. [Daha ileri gitmeden önce, Docker ekosisteminde sıkça kullanılan bazı terminolojiyi netleştirmemiz gerekiyor.]



**Docker Editions [Docker Sürümleri]**

* **Docker Community Edition (CE)** is ideal for Developers who are looking for experimenting docker and creating container-based applications. [Docker Community Edition (CE), docker'ı denemek ve kapsayıcı tabanlı uygulamalar oluşturmak isteyen Geliştiriciler için idealdir.] It’s free. [Bedava.]
* **Docker Enterprise Edition (EE)** is a Containers-as-a-Service (CaaS) platform. [Docker Enterprise Edition (EE), bir Hizmet Olarak Kapsayıcılar (CaaS) platformudur.] Enterprise Edition Subscription packages include an integrated Docker platform and tooling for container management and security. [Enterprise Edition Abonelik paketleri, kapsayıcı yönetimi ve güvenliği için entegre bir Docker platformu ve araçları içerir.]

**Docker ID [liman işçisi kimliği]**

* Your free Docker ID grants you access to Docker services such as the Docker Store, Docker Cloud, Docker Hub repositories, and some beta programs. [Ücretsiz Docker Kimliğiniz, Docker Store, Docker Cloud, Docker Hub depoları ve bazı beta programları gibi Docker hizmetlerine erişmenizi sağlar.] Your Docker ID becomes repository namespace used by hosted services such as Docker Hub and Docker Cloud. [Docker Kimliğiniz, Docker Hub ve Docker Cloud gibi barındırılan hizmetler tarafından kullanılan depo ad alanı haline gelir.] All you need is an email address. [Tek ihtiyacınız olan bir e-posta adresi.]

**Registry [kayıt]**

* A Docker registry stores Docker images. [Docker kayıt defteri, Docker görüntülerini depolar.]
* **Docker Hub (Like GitHub)** is a cloud-based registry service that allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts. [Docker Hub (GitHub gibi), kod havuzlarına bağlanmanıza, görüntülerinizi oluşturmanıza ve test etmenize, manuel olarak aktarılan görüntüleri depolamanıza ve görüntüleri ana bilgisayarlarınıza dağıtabilmeniz için Docker Cloud'a bağlantı kurmanıza olanak tanıyan bulut tabanlı bir kayıt hizmetidir.]
* **Docker Cloud** uses the hosted Docker Cloud Registry, which allows you to publish Dockerized images on the internet either publicly or privately. [Docker Cloud, Dockerized görüntüleri internette genel veya özel olarak yayınlamanıza olanak tanıyan barındırılan Docker Cloud Registry'yi kullanır.] Docker Cloud can also store pre-built images, or link to your source code so it can build the code into Docker images, and optionally test the resulting images before pushing them to a repository. [Docker Cloud ayrıca önceden oluşturulmuş görüntüleri depolayabilir veya kaynak kodunuza bağlanabilir, böylece kodu Docker görüntülerinde oluşturabilir ve isteğe bağlı olarak elde edilen görüntüleri bir havuza göndermeden önce test edebilir.]

**Docker Client [liman işçisi istemcisi]**

* The command-line tool that allows the user to interact with the daemon. [Kullanıcının arka plan programı ile etkileşim kurmasını sağlayan komut satırı aracı.] It is the primary user interface to Docker. [Docker'ın birincil kullanıcı arabirimidir.] Accepts commands from the user and communicates back and forth with a Docker daemon. [Kullanıcıdan gelen komutları kabul eder ve bir Docker arka plan programı ile ileri geri iletişim kurar.]

**Docker Daemon [liman işçisi arka plan programı]**

* The background service running on the host that manages the building, running and distributing Docker containers. [Docker kapsayıcılarını oluşturmayı, çalıştırmayı ve dağıtmayı yöneten ana bilgisayarda çalışan arka plan hizmeti.] Runs on a host machine. [Bir ana makinede çalışır.]

**Dockerfile [liman dosyası]**

* Dockerfile is a text document that contains all commands a user could call on the command line to create an image. [Dockerfile, bir kullanıcının bir görüntü oluşturmak için komut satırında arayabileceği tüm komutları içeren bir metin belgesidir.]

**Docker Image [Docker Görüntüsü]**

* Docker image is a read-only template with instructions for creating a Docker container. [Docker görüntüsü, Docker kapsayıcısı oluşturmaya yönelik talimatları içeren salt okunur bir şablondur.]

**Docker Container [liman işçisi konteyner]**

* Created from Docker images and run the actual application. [Docker görüntülerinden oluşturuldu ve gerçek uygulamayı çalıştırın.] It is a runnable instance of an image. [Bir görüntünün çalıştırılabilir bir örneğidir.]

**Docker Compose [Docker Oluştur]**

* Compose is a tool for defining and running multi-container Docker applications. [Compose, çok kapsayıcılı Docker uygulamalarını tanımlamak ve çalıştırmak için bir araçtır.]

Q: What are Docker Image and Docker Hub? [S: Docker Image ve Docker Hub nedir?]  
A: The Docker Image is a set of files and a combination of parameters that will allow creating the instances to run in separate containers as an isolated process. [C: Docker Görüntüsü, örneklerin izole bir süreç olarak ayrı kaplarda çalışacak şekilde oluşturulmasına izin verecek bir dosya kümesi ve bir parametre kombinasyonudur.] The Docker hub is a kind of repository to the images where these images can be stored and this access is public. [Docker hub, bu görüntülerin saklanabileceği ve bu erişimin herkese açık olduğu bir tür görüntü deposudur.] The Docker run command can be used to create the instance called container which can be run using the Docker image. [Docker run komutu, Docker görüntüsü kullanılarak çalıştırılabilen kapsayıcı adlı örneği oluşturmak için kullanılabilir.] Docker hub is the largest public repository of the image containers which is being maintained by the community of developers and individual contributors. [Docker hub, geliştiriciler ve bireysel katkıda bulunanlar topluluğu tarafından sürdürülen görüntü kapsayıcılarının en büyük genel deposudur.]

 - Interview Q&A

### Installing Docker

In this lesson we will cover;

* + - * Docker Desktop for Windows
      * Docker Desktop for Mac
      * Docker for Linux

### 

## **Docker Desktop for Windows**

 Docker on Windows

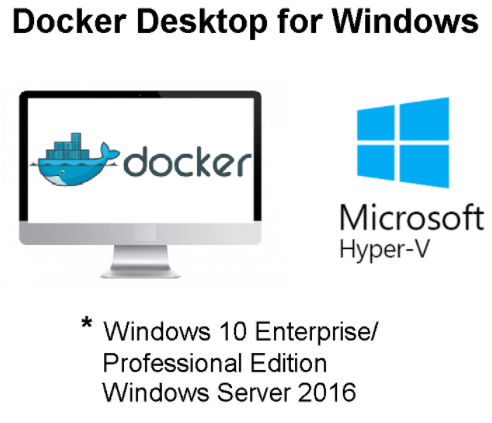
✔ Install Docker Desktop on Windows

### Docker on Windows

* Docker Desktop for Windows is the Community version of Docker for Microsoft Windows. You can download Docker Desktop for Windows from Docker Hub.

[Download from Docker Hub](https://hub.docker.com/editions/community/docker-ce-desktop-windows/)

The installation provides Docker Engine, Docker CLI client, Docker Compose, Docker Machine, and Kitematic. Containers and images created with Docker Desktop for Windows are shared between all user accounts on machines where it is installed. This is because all Windows accounts use the same VM to build and run containers.

****

**⚠️Avoid ! :What to know before you install**  
**System Requirements:**

* Windows 10 64-bit: Pro, Enterprise, or Education (Build 15063 or later).
* Hyper-V and Containers Windows features must be enabled.
* The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
  + 64 bit processor with Second Level Address Translation (SLAT)
  + 4GB system RAM
  + BIOS-level hardware virtualization support must be enabled in the BIOS settings.

### Docker Containers

In this lesson we will cover;

* + - * + What are the containers?
        + Run your first container
        + Container naming
        + Starting a stopped container
        + Attaching to a container
        + Finding out more about our container
        + Deleting a container

### What are containers? [Konteynerler nelerdir?]

The industry standard today is to use Virtual Machines (VMs) to run software applications. [Bugün endüstri standardı, yazılım uygulamalarını çalıştırmak için Sanal Makineleri (VM'ler) kullanmaktır.] VMs run applications inside a guest Operating System, which runs on virtual hardware powered by the server’s host OS. [VM'ler, uygulamaları sunucunun ana işletim sistemi tarafından desteklenen sanal donanım üzerinde çalışan bir konuk İşletim Sistemi içinde çalıştırır.]

VMs are great at providing full process isolation for applications: there are very few ways a problem in the host operating system can affect the software running in the guest operating system, and vice-versa. [VM'ler, uygulamalar için tam süreç izolasyonu sağlamada harikadır: ana bilgisayar işletim sistemindeki bir sorunun konuk işletim sisteminde çalışan yazılımı etkilemesinin çok az yolu vardır ve bunun tersi de geçerlidir.] But this isolation comes at great cost — the computational overhead spent virtualizing hardware for a guest OS to use is substantial. [Ancak bu izolasyonun maliyeti yüksektir - konuk işletim sisteminin kullanması için donanımı sanallaştırmak için harcanan hesaplama yükü çok fazladır.]

Containers take a different approach: by leveraging the low-level mechanics of the host operating system, containers provide most of the **isolation** of virtual machines at a **fraction of the computing power**. [Kapsayıcılar farklı bir yaklaşım benimsiyor: ana bilgisayar işletim sisteminin düşük seviyeli mekaniğinden yararlanan kapsayıcılar, sanal makinelerin izolasyonunun çoğunu bilgi işlem gücünün çok küçük bir bölümünde sağlıyor.]

Containers offer a **logical packaging** mechanism in which applications can be **abstracted** from the environment in which they actually run. [Kapsayıcılar, uygulamaların gerçekte çalıştıkları ortamdan soyutlanabilecekleri mantıksal bir paketleme mekanizması sunar.] This **decoupling** allows container-based applications to be deployed easily and consistently, regardless of whether the target environment is a private data center, the public cloud, or even a developer’s personal laptop. [Bu ayrıştırma, hedef ortamın özel bir veri merkezi, genel bulut veya hatta bir geliştiricinin kişisel dizüstü bilgisayarı olup olmadığına bakılmaksızın, kapsayıcı tabanlı uygulamaların kolay ve tutarlı bir şekilde dağıtılmasına olanak tanır.] This gives developers the ability to create predictable environments that are isolated from the rest of the applications and can be run anywhere. [Bu, geliştiricilere diğer uygulamalardan izole edilmiş ve her yerde çalıştırılabilen öngörülebilir ortamlar oluşturma yeteneği verir.]

Q: What is Docker Container [S: Docker Konteyneri Nedir?]  
A: A Docker Container is a form of encapsulation to the application which holds all the dependencies which share the kernel with other containers in the duration of running the isolated processes on the host operating system. [A: Docker Konteyneri, ana bilgisayar işletim sisteminde izole edilmiş işlemlerin çalıştırıldığı süre boyunca çekirdeği diğer kapsayıcılarla paylaşan tüm bağımlılıkları tutan uygulamanın bir kapsülleme biçimidir.] A Docker container can be created by creating a Docker image. [Bir Docker görüntüsü oluşturularak bir Docker kapsayıcı oluşturulabilir.] These Docker images can be run after that using Docker commands. [Bu Docker görüntüleri, bundan sonra Docker komutları kullanılarak çalıştırılabilir.] Docker containers are the instances of the Docker images at the runtime. [Docker kapsayıcıları, çalışma zamanında Docker görüntülerinin örnekleridir.] Docker images can be stored in any public hosts or private hosts like Docker hub. [Docker görüntüleri, herhangi bir genel ana bilgisayarda veya Docker hub gibi özel ana bilgisayarlarda saklanabilir.] Docker Image is a set of files which can be run in an isolated process. [Docker Image, yalıtılmış bir işlemde çalıştırılabilen bir dizi dosyadır.]

 Interview Q&A

### Run your first container

We’re going to start with checking that Docker is working correctly, and then we’re going to take a look at the basic Docker workflow: creating and managing containers. [Docker'ın doğru çalıştığını kontrol ederek başlayacağız ve ardından temel Docker iş akışına bir göz atacağız: kapsayıcılar oluşturma ve yönetme.] We’ll take a container through its typical lifecycle from creation to a managed state and then stop and remove it. [Bir kapsayıcıyı, oluşturmadan yönetilen duruma kadar tipik yaşam döngüsü boyunca ele alacağız ve ardından durdurup kaldıracağız.]

Firstly, let’s check that the docker **binary**exists and is functional: [İlk olarak, docker ikili dosyasının var olduğunu ve çalışır durumda olduğunu kontrol edelim:]

Input:



Output:



Here, we’ve passed the info command to the docker binary, which returns a list of any containers, any images (the building blocks Docker uses to build containers), the execution and storage drivers. [Burada, herhangi bir konteynerin, herhangi bir görüntünün (Docker'ın konteynerler oluşturmak için kullandığı yapı taşları), yürütme ve depolama sürücülerinin bir listesini döndüren docker ikili dosyasına info komutunu ilettik.] Docker is using, and its basic configuration. [Docker kullanıyor ve temel yapılandırması.]

**Your Container [Konteyneriniz]**

Now let’s try and launch our first container with Docker. [Şimdi ilk kapsayıcımızı Docker ile deneyelim ve başlatalım.] We’re going to use the docker run command to create a container. [Bir konteyner oluşturmak için docker run komutunu kullanacağız.] The docker run command provides all of the ”launch” capabilities for Docker. [Docker run komutu, Docker için tüm 'başlatma' yeteneklerini sağlar.] We’ll be using it a lot to create new containers. [Yeni kapsayıcılar oluşturmak için onu çok kullanacağız.]

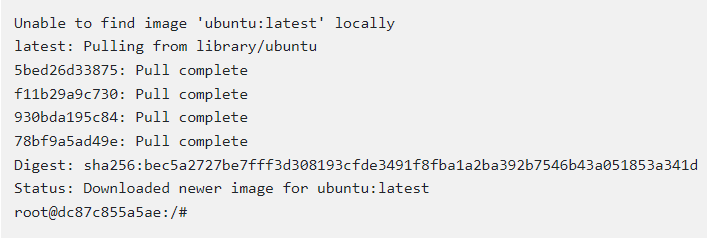
**💡Tips: [💡İpuçları:]**

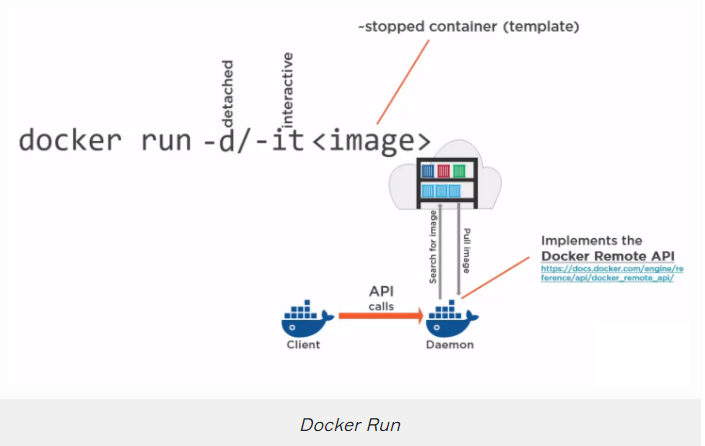
* You can find a full list of the available Docker commands [**here**](https://docs.docker.com/engine/reference/commandline/cli/)or by typing **docker help**. [Kullanılabilir Docker komutlarının tam listesini burada veya docker help yazarak bulabilirsiniz.] You can also use the Docker man pages (e.g., man docker-run). [Docker man sayfalarını da kullanabilirsiniz (örneğin, man docker-run).]

input :



output





First, we told Docker to run a command using docker run. [İlk olarak, Docker'a docker run kullanarak bir komut çalıştırmasını söyledik.] We passed it two command line flags: -i and -t. [İki komut satırı bayrağını geçtik: -i ve -t.]

The -i flag keeps STDIN open from the container, even if we’re not attached to it. [-i bayrağı, biz ona bağlı olmasak bile STDIN'i kapsayıcıdan açık tutar.] This persistent standard input is one half of what we need for an interactive shell. [Bu kalıcı standart girdi, etkileşimli bir kabuk için ihtiyacımız olanın yarısıdır.]

The -t flag is the other half and tells Docker to assign a pseudo -tty to the container we’re about to create. [-t bayrağı diğer yarısıdır ve Docker'a oluşturmak üzere olduğumuz kapsayıcıya sözde -tty atamasını söyler.] This provides us with an interactive shell in the new container. [Bu bize yeni kapsayıcıda etkileşimli bir kabuk sağlar.] This line is the base configuration needed to create a container with which we plan to interact on the command line rather than run as a daemonized service. [Bu satır, bir arka plan programı olarak çalıştırmak yerine komut satırında etkileşim kurmayı planladığımız bir kapsayıcı oluşturmak için gereken temel yapılandırmadır.]

**💡Tips: [💡İpuçları:]**

* You can find a full list of the available Docker run flags by typing **docker run help**. [Docker run help yazarak mevcut Docker run bayraklarının tam listesini bulabilirsiniz.] You can also use the Docker man pages (e.g., man docker-run). [Docker man sayfalarını da kullanabilirsiniz (örneğin, man docker-run).]

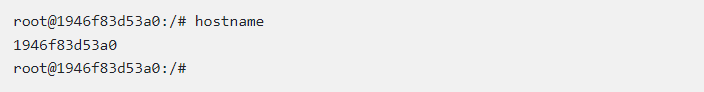
We told Docker which image to use to create a container, in this case, the ubuntu image. [Docker'a bir kapsayıcı oluşturmak için hangi görüntüyü kullanacağını söyledik, bu durumda ubuntu görüntüsü.] The ubuntu image is a stock image, also known as a ”base” image, provided by Docker, Inc., on the Docker Hub registry. [Ubuntu görüntüsü, Docker Hub kayıt defterinde Docker, Inc. tarafından sağlanan 'temel' görüntü olarak da bilinen bir stok görüntüdür.]

So what was happening in the background here? [Peki burada arka planda neler oluyordu?] Firstly, Docker checked locally for the ubuntu image. [İlk olarak, Docker ubuntu görüntüsünü yerel olarak kontrol etti.] If it can’t find the image on our local Docker host, it will reach out to the Docker Hub registry run by Docker, Inc., and look for it there. [Resmi yerel Docker ana makinemizde bulamazsa, Docker, Inc. tarafından işletilen Docker Hub kayıt defterine ulaşacak ve orada arayacaktır.] Once Docker had found the image, it downloaded the image and stored it on the local host. [Docker görüntüyü bulduktan sonra görüntüyü indirdi ve yerel ana bilgisayarda depoladı.]

Docker then used this image to create a new container inside a filesystem. [Docker daha sonra bu görüntüyü bir dosya sistemi içinde yeni bir kapsayıcı oluşturmak için kullandı.] The container has a network, IP address, and a bridge interface to talk to the local host. [Kapsayıcı, yerel ana bilgisayarla konuşmak için bir ağ, IP adresi ve bir köprü arabirimine sahiptir.] Finally, we told Docker which command to run in our new container, in this case launching a Bash shell with the /bin/bash command. [Son olarak, Docker'a yeni kapsayıcımızda hangi komutu çalıştıracağını söyledik, bu durumda /bin/bash komutuyla bir Bash kabuğu başlatıyoruz.]

## **Working with Container [Container ile Çalışmak]**

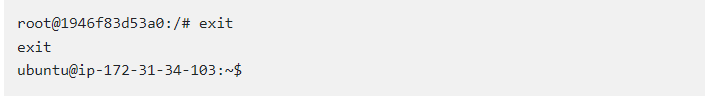
We are now logged into a new container, with the catchy ID of 1946f83d53a0, as the root user. [Artık kök kullanıcı olarak 1946f83d53a0'ın akılda kalıcı kimliğiyle yeni bir kapsayıcıda oturum açmış durumdayız.] This is a fully-fledged Ubuntu host, and we can do anything we would like in it. [Bu tam teşekküllü bir Ubuntu sunucusu ve içinde istediğimiz her şeyi yapabiliriz.] Let’s start by asking for its hostname. [Ana bilgisayar adını sorarak başlayalım.]



### Installing a package in container

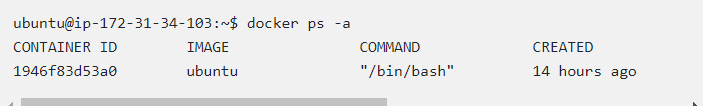


You can keep playing with the container for as long as you like. [Konteynerle istediğiniz kadar oynamaya devam edebilirsiniz.] When you’re done, type exit, and you’ll return to the command prompt of your Ubuntu host. [İşiniz bittiğinde, çıkış yazın ve Ubuntu ana makinenizin komut istemine dönersiniz.]



## It has now stopped running. [Şimdi çalışmayı durdurdu.] The container only runs for as long as the command we specified, /bin/bash, is running. [Kap, yalnızca belirttiğimiz komut olan /bin/bash çalıştığı sürece çalışır.] Once we exited the container, that command ended, and the container was stopped. [Konteynerden çıktığımızda bu komut sona erdi ve konteyner durduruldu.]**docker ps -a command**

The container still exists; we can show a list of current containers using the docker ps -a command. [Kapsayıcı hala var; docker ps -a komutunu kullanarak mevcut kapların bir listesini gösterebiliriz.]



By default, when we run just docker ps, we will only see the running containers. [Varsayılan olarak, sadece docker ps çalıştırdığımızda, sadece çalışan konteynerleri göreceğiz.] When we specify the -a flag, the docker ps -a command will show us all containers, both stopped and running. [-a bayrağını belirttiğimizde, docker ps -a komutu bize hem durmuş hem de çalışan tüm kapsayıcıları gösterecektir.]

We learn quite a bit of information about our container: its ID, the image used to create it, the command it last ran, when it was created, and its exit status (in our case, 0, because it was exited normally using the exit command). [Kapsayıcımız hakkında oldukça fazla bilgi öğreniyoruz: kimliği, onu oluşturmak için kullanılan görüntü, en son çalıştırdığı komut, oluşturulduğu zaman ve çıkış durumu (bizim durumumuzda 0, çünkü normalde çıkış komutu).] We can also see that each container has a name. [Ayrıca her kapsayıcının bir adı olduğunu görebiliriz.]

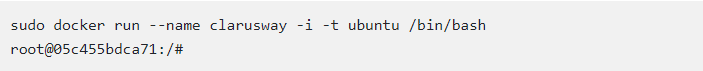
**💡Tips:**

There are three ways containers can be identified:

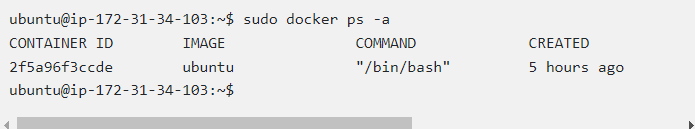
* a short UUID (1946f83d53a0),
* a longer UUID (like 1946f83d53a021548190053c4053f93fda09c3e5fa6bd670acbc7697a96b9961), and
* a name (tender\_visvesvaraya).

### Container naming

Docker will automatically generate a name at random for each container we create. [Docker, oluşturduğumuz her kapsayıcı için otomatik olarak rastgele bir ad oluşturacaktır.] We see that the container we’ve created is called tender\_visvesvaraya. [Oluşturduğumuz kapsayıcının ihale\_visvesvaraya olarak adlandırıldığını görüyoruz.] If we want to specify a particular container name in place of the automatically generated name, we can do so using the --name flag. [Otomatik olarak oluşturulan ad yerine belirli bir kapsayıcı adı belirtmek istiyorsak, bunu --name bayrağını kullanarak yapabiliriz.]



You can check it in another terminal and verify the following output. [Başka bir terminalde kontrol edebilir ve aşağıdaki çıktıyı doğrulayabilirsiniz.]

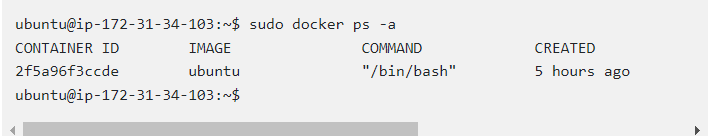
****

This would create a new container called clarusway. [Bu, clarusway adında yeni bir kapsayıcı yaratacaktır.] A valid container name can contain the following characters: a to z, A to Z, the digits 0 to 9, the underscore, period, and dash (or, expressed as a regular expression: [a-zA-Z0-9\_.-]). [Geçerli bir kapsayıcı adı şu karakterleri içerebilir: a'dan z'ye, A'dan Z'ye, 0'dan 9'a rakamlar, alt çizgi, nokta ve tire (veya normal ifade olarak ifade edilir: [a-zA-Z0-9\_.-] ]).]

We can use the container name in place of the container ID in most Docker commands. [Çoğu Docker komutunda kapsayıcı kimliği yerine kapsayıcı adını kullanabiliriz.] It is also much easier to remember a specific container name than a container ID or even a random name. [Ayrıca belirli bir kapsayıcı adını hatırlamak, bir kapsayıcı kimliğinden veya hatta rastgele bir addan çok daha kolaydır.]

Names are unique. [İsimler benzersizdir.] If we try to create two containers with the same name, the command will fail. [Aynı ada sahip iki kapsayıcı oluşturmaya çalışırsak, komut başarısız olur.] We need to delete the previous container with the same name before we can create a new one. [Yeni bir tane oluşturabilmemiz için önceki kapsayıcıyı aynı isimle silmemiz gerekiyor.] We can do so with the docker rm command. [Bunu docker rm komutu ile yapabiliriz.]

### Starting a stopped container

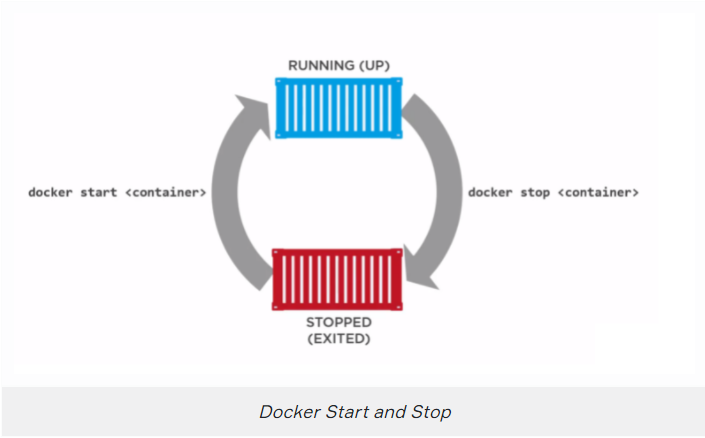
****

As we see above, we stopped clarusway container. [Yukarıda gördüğümüz gibi, clarusway container'ı durdurduk.] If we want, we can restart a stopped container like so: [İstersek, durdurulan bir kapsayıcıyı şu şekilde yeniden başlatabiliriz:]

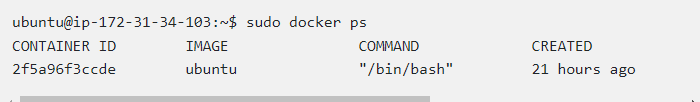


We could also refer to the container by its container ID instead. [Bunun yerine kapsayıcıya kapsayıcı kimliğiyle de başvurabiliriz.]





If we run the docker ps command without the -a flag, we’ll see our running container. [-a bayrağı olmadan docker ps komutunu çalıştırırsak, çalışan konteynerimizi görürüz.]

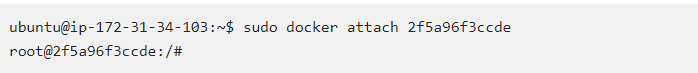


### Attaching to a container

Our container will restart with the same options we’d specified when we launched it with the docker run command. [Konteynerimiz docker run komutu ile başlattığımızda belirttiğimiz seçeneklerle yeniden başlayacaktır.] So there is an interactive session waiting on our running container. [Yani çalışan kapsayıcımızda bekleyen etkileşimli bir oturum var.] We can reattach to that session using the docker attach command. [Docker ekleme komutunu kullanarak bu oturuma yeniden bağlanabiliriz.] So, we’ll be brought back to our container’s Bash prompt. [Böylece, kapsayıcımızın Bash istemine geri döneceğiz.]



We could also do it via ID.



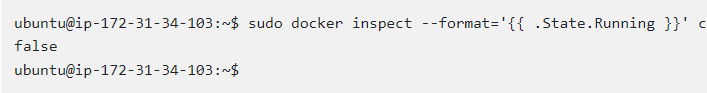
**Note:** Instead of typing a lengthy characters of container id, a few character that is enough to differentiate it from the other containers can be used. For example sudo docker attach 2f5 is the same with sudo docker attach 2f5a96f3ccde.

### Finding out more about our container

In addition to the information we retrieved about our container using the docker ps command, we can get a whole lot more information using the docker inspect command. [docker ps komutunu kullanarak kapsayıcımız hakkında aldığımız bilgilere ek olarak, docker inspect komutunu kullanarak çok daha fazla bilgi alabiliriz.]



The docker inspect command will interrogate our container and return its configuration information, including names, commands, networking configuration, and a wide variety of other useful data. [Docker inspect komutu, kapsayıcımızı sorgulayacak ve adlar, komutlar, ağ yapılandırması ve çok çeşitli diğer yararlı veriler dahil olmak üzere yapılandırma bilgilerini döndürecektir.] We can also selectively query the inspect results hash using the -f or --format flag. [Ayrıca -f veya --format bayrağını kullanarak inceleme sonuçları karmasını seçici olarak sorgulayabiliriz.]



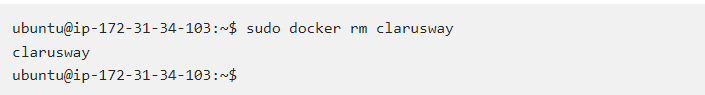
This will return the running state of the container, which in our case is false. [Bu, bizim durumumuzda yanlış olan konteynerin çalışma durumunu döndürür.]

**💡Tips: [💡İpuçları:]**

* In addition to inspecting containers, you can see a bit more about how Docker works by exploring the /var/lib/docker directory. [Kapsayıcıları incelemeye ek olarak, /var/lib/docker dizinini keşfederek Docker'ın nasıl çalıştığı hakkında biraz daha bilgi edinebilirsiniz.] This directory holds your images, containers, and container configuration. [Bu dizin, resimlerinizi, kapsayıcılarınızı ve kapsayıcı yapılandırmanızı tutar.] You’ll find all your containers in the /var/lib/docker/containers directory. [Tüm kapsayıcılarınızı /var/lib/docker/containers dizininde bulacaksınız.]

### Deleting a container

If we are finished with a container, we can delete it using the docker rm command. [Bir container ile işimiz bittiyse docker rm komutu ile onu silebiliriz.]



# Docker Beginner Tutorial 10 - What are Docker Containers | How to create Docker Containers

<https://www.youtube.com/watch?v=Rv3DAJbDrS0>

### Docker Volumes

In this lesson we will cover;

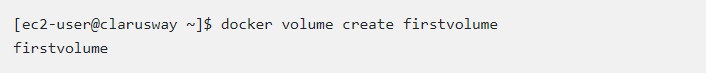
* + - * + Manage data in Docker
        + Declaration of volumes
        + Usage volume with different containers
        + Use a read-only volume

### Manage data in Docker

By default, all files created inside a container are stored on a writable container layer. [Varsayılan olarak, bir kapsayıcı içinde oluşturulan tüm dosyalar, yazılabilir bir kapsayıcı katmanında depolanır.] This means that the data doesn’t persist when that container no longer exists. [Bu, o kapsayıcı artık mevcut olmadığında verilerin kalıcı olmadığı anlamına gelir.]

Docker volumes, which are special directories in a container, store files in the host machine so that the files are persisted even after the container stops. [Bir kapsayıcıdaki özel dizinler olan Docker birimleri, dosyaların kapsayıcı durduktan sonra bile kalıcı olması için ana makinede depolar.]

Volumes are created and managed by Docker. [Birimler Docker tarafından oluşturulur ve yönetilir.] You can create a volume explicitly using the docker volume create command. [Docker birim oluşturma komutunu kullanarak açıkça bir birim oluşturabilirsiniz.]

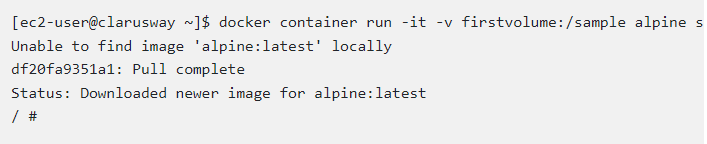


When you create a volume, it is stored within a directory on the Docker host. [Bir birim oluşturduğunuzda, Docker ana bilgisayarındaki bir dizinde depolanır.] When you mount the volume into a container, this directory is what is mounted into the container. [Birimi bir kapsayıcıya bağladığınızda, kapsayıcıya monte edilen dizin bu dizindir.] Look at the Mountpoint. [Mountpoint'e bakın.]



### Declaration of volumes

Volumes can be declared on the command-line, with the --volume or -v flag for docker run. [Birimler, docker çalıştırması için --volume veya -v bayrağıyla komut satırında bildirilebilir.] Let's create an alpine container. [Bir alpin kapsayıcı oluşturalım.]



**💡Tip:**

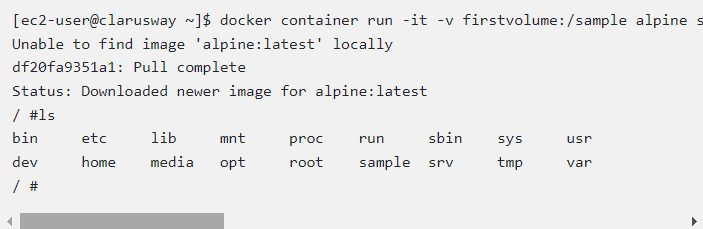
-v or --volume: Consists of three fields, separated by colon characters (:). [-v veya --volume: İki nokta üst üste karakterleriyle (:) ayrılmış üç alandan oluşur.] The fields must be in the correct order. [Alanlar doğru sırada olmalıdır.]

* the first field is the name of the volume, and is unique on a given host machine. [ilk alan birimin adıdır ve belirli bir ana makinede benzersizdir.] In this example volume name is firstvolume. [Bu örnekte birim adı birinci cilttir.]
* The second field is the path where the file or directory are mounted in the container. [İkinci alan, dosyanın veya dizinin kapsayıcıya bağlandığı yoldur.] In this example folder in container is /sample. [Bu örnekte kapsayıcıdaki klasör /sample'dır.]
* The third field is optional, and is a comma-separated list of options, such as ro (read only). [Üçüncü alan isteğe bağlıdır ve ro (salt okunur) gibi virgülle ayrılmış bir seçenekler listesidir.]

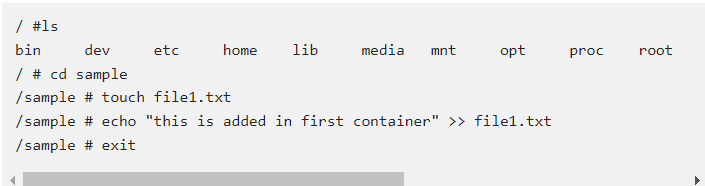
**Alpine:**

* Alpine Linux is an independent, non-commercial, general purpose Linux distribution designed for power users who appreciate security, simplicity and resource efficiency. [Alpine Linux, güvenlik, basitlik ve kaynak verimliliğini takdir eden ileri düzey kullanıcılar için tasarlanmış bağımsız, ticari olmayan, genel amaçlı bir Linux dağıtımıdır.]
* Because of its small size, it is commonly used in containers providing quick boot-up times. [Küçük boyutu nedeniyle, hızlı başlatma süreleri sağlayan kaplarda yaygın olarak kullanılır.]

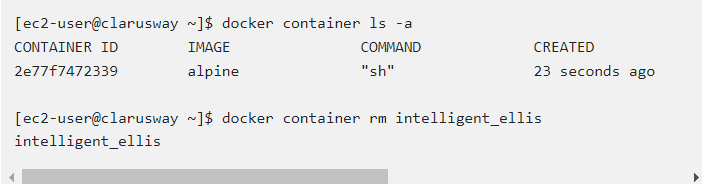
When we type ls command in alpine terminal, we can see the sample folder.



We create a file in the sample folder and exit.



We remove the alpine container.



Let's check the file1.txt.



As we see above, file1.txt is still there even if we remove the container.

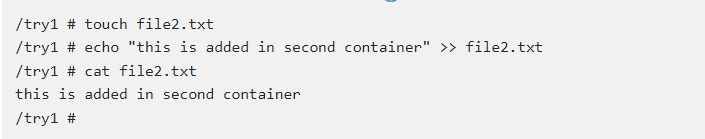
### Usage volume with different containers

Let's run an alpine image and this time we will create try1 folder instead of sample folder.

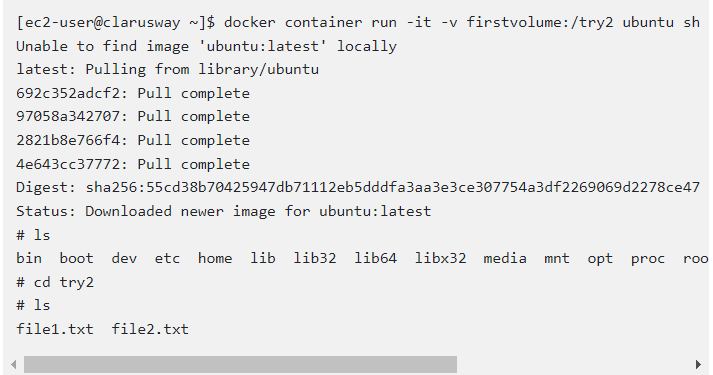


As we see, we can reach file1.txt via a new container.

We can add a new file to the try1 folder.



We create an ubuntu image.



We can use the same volumes with different containers.

sudo systemctl start docker

sudo systemctl start docker

sudo systemctl status docker

sudo usermod -aG docker ec2-user

newgrp docker

1 ls -al

2 sudo yum update -y

3 docker ps -a

4 docker pull ubuntu:latest

5 docker image ls

6 docker info

7 docker run -it --name new\_ubuntu ubuntu

8 sudo docker ps -a

9 docker stop webserver

10 sudo docker ps -a

11 docker stop webserver2

12 sudo docker ps -a

13 sudo docker rm 870

14 sudo docker ps -a

15 sudo docker rm 62a

16 sudo docker ps -a

17 sudo docker rm b12

18 sudo docker ps -a

19 history

1 ls -al

2 sudo yum update -y

3 docker ps

4 sudo docker ps

5 docker network ls

6 sudo docker network ls

7 sudo docker network inspect

8 docker network inspect

9 docker network inspect b1270e91a3ac

10 sudo docker network inspect b1270e91a3ac

11 docker network inspect 172.17.0.2

12 docker network inspect 2bbfc809ae1f

13 sudo systemctl start docker

14 sudo systemctl enable docker

15 docker network inspect 2bbfc809ae1f

16 ping -c 4 172.17.0.2

17 ping -c 4 172.17.0.1

18 sudo docker network inspect 2bbfc809ae1f

19 docker run -it -d --name webserver -p 80:80 nginx

20 sudo docker run -it -d --name webserver -p 80:80 nginx

21 sudo docker ps

22 sudo docker ls

23 docker ls

24 docker ps

25 sudo docker ps

26 sudo docker ps -a

27 sudo docker run -it -d --name webserver2 -p 81:80 nginx

28 docker exec -it 870 bash

29 docker sudo exec -it 870 bash

30 sudo docker exec -it 870 bash

31 history