So what is Docker?

So **Docker is a way to package all our application and all the related libraries** and whatever the things that we can use within our application **into a single bundle**.  
  
So all these things, okay, all these different different applications, all the different properties

are the different libraries.

All will be bundled together in a container and this container will be running in your Docker.  
A blackboard with white text and drawings

AI-generated content may be incorrect.  
So you see your application has Java , Maven , Application so instead of handling them differently you package them into a single bundle this is known as Docker Image   
  
So whenever you create this image, you can see that this image is become portable, which means you can pick this image and you can go to any machine and you can go ahead and deploy this application that much portable.  
  
you do not have to worry about all the different configurations and all the different libraries that will need to be installed in that particular machine.

Everything will be contained in that particular container, in that particular image, and you just run that particular image.  
  
  
  
  
Now, where this image will live,?  
  
This image will live in a container registry.

Now, I will give you the simple example.

We have different maven plug in.So all this MAVEN plug ins and MAVEN libraries lives where it lives in the MAVEN repository, right?

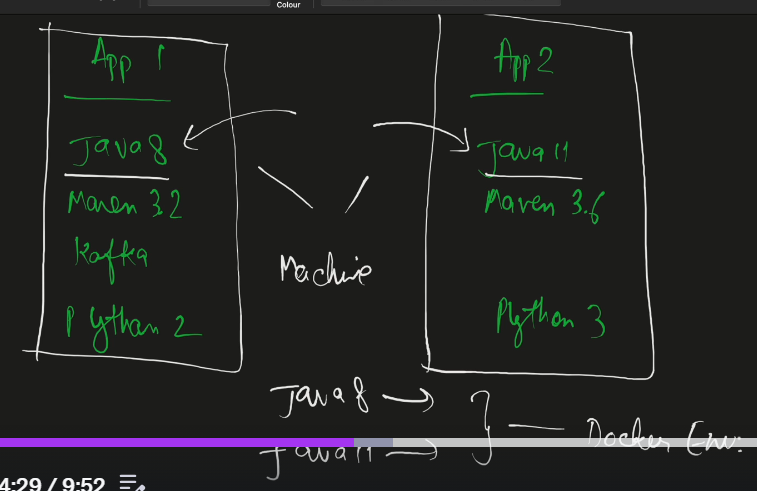
So everything will be there in the MAVEN repository and whoever wants, they can take those data or take those libraries from the MAVEN repository.

Similar is with the Docker images.

**So all those Docker images, all these Docker images will be there in the Docker registry.**

**And this Docker registry can be public as well and can be private as well.  
-----------------------------------------------------------------------------------------------------**So by default, Docker gives the Docker Hub registry that's a public registry where all our images will live.

But if you want, you can create the private registry as well and within your organization you can create private and all your Docker images can live there.

And whenever you want to use those images, you can get those images from the Docker registry and you can build your applications on that.  
------------------------------------------------------------------------------------------------------  
  
Need of Docker Enviroment ?  
So earlier like when you were have 2 applications you have to manage configuration of both applications in our host machine but with Docker   
But when you are using Docker, what you can do **is you can easily create the images of it.You can create and download the images for Java eight that can be easily configured**.

**You can have the images for Java 11 that also can be easily configured and these all are within the Docker environment itself and you can easily access all those data, all those things from your host machine, that is your system machine and everything, whatever the application that you need will be there in the Docker environment.  
No need of manual configuration.  
----------------------------------------------------------------------------------------------  
Docker Container:  
A blackboard with white text

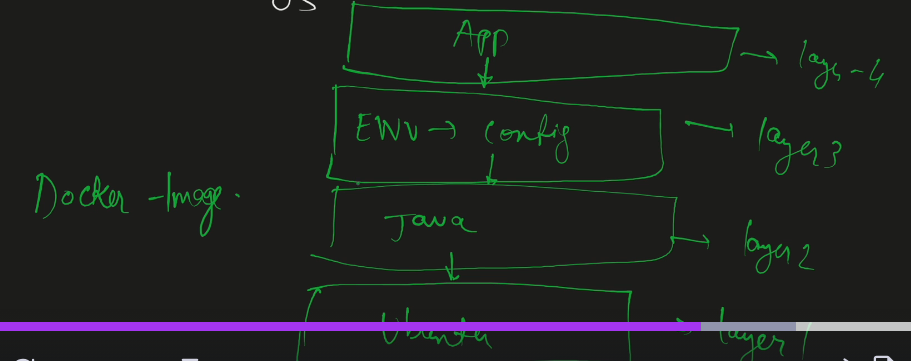
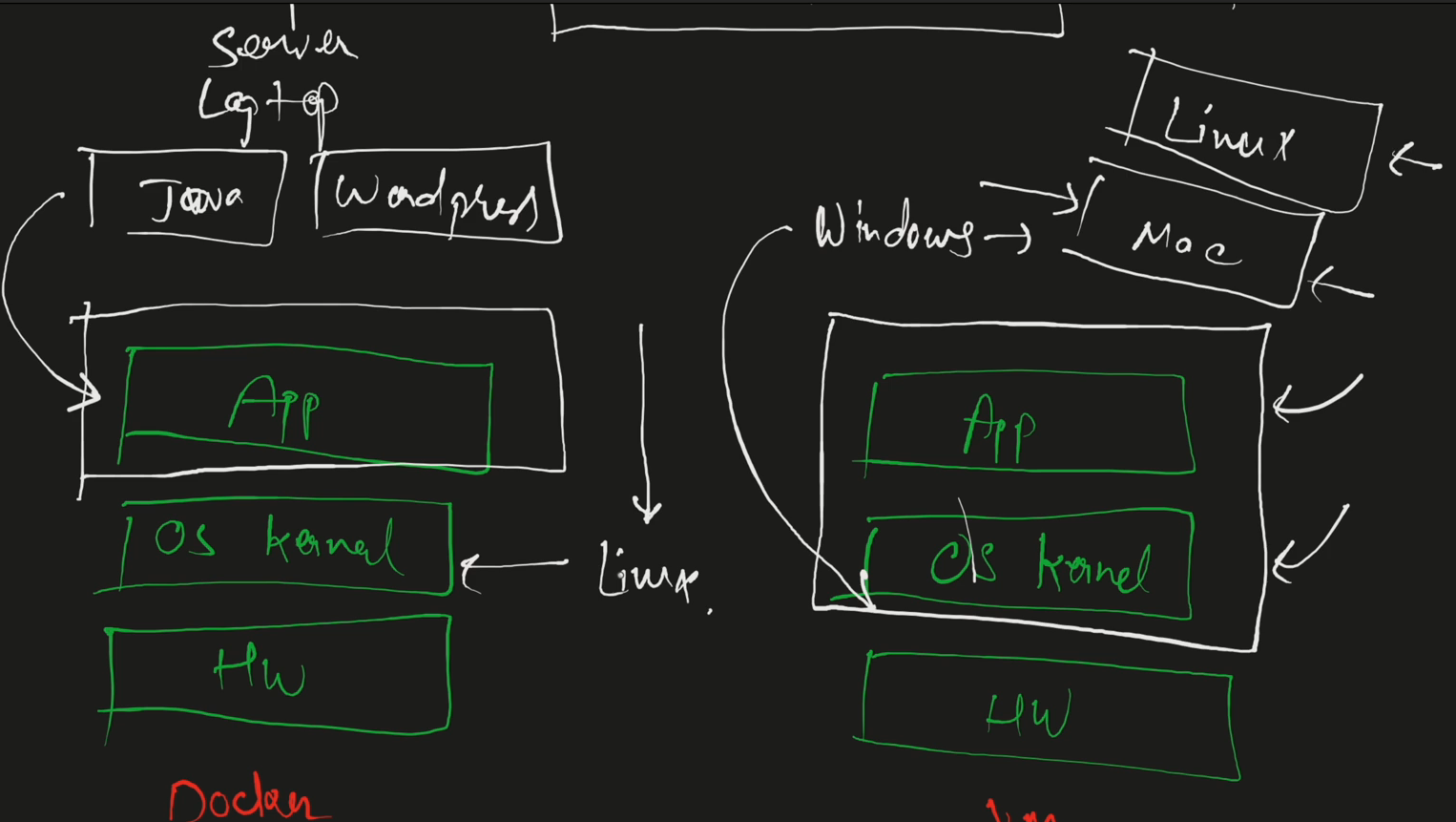
AI-generated content may be incorrect.**Docker container is a package itself, which includes each and everything with all the libraries, with configuration, with the application.

Everything combined together forms a Docker container and this Docker container will be running in your Docker environment**.**

**A blackboard with white text

AI-generated content may be incorrect.  
[1]**So **Docker container is** created on **base of a Docker image.**

So first of all, there will be always a Docker image.So we'll be creating a Docker image. **Docker image is something that will consist of each and every configuration in each and everything.  
[3] So this Docker image will live in the Docker registry.**And whenever we will run this Docker image that is called container, that will be the container which will have each and everything, which is a running entity in the Docker environment.  
--------------------------------------------------------------------------------------------  
So every docker image is consisted of different layers.  
  
**So suppose let's take the example that you want the Java application to be packaged in your Docker image.**So there will be Linux (ubuntu ) layer , then Java Layer , then ENV Variables and config as layer 3 .

At the end you will get your you will get Application layer.  
  
  
**So at the end, Docker image will consist of all the different layers, and those layers combined together will make a single Docker image and that Docker image we can run and that is going to be called as my Docker container.  
  
------------------------------------------------------------------------------------------------  
Install Docker :**You can go to install desktop on computer 🡪 there you can see all 3 options mac os and linux   
Also create a account in Docker hub used in Docker Image 🡪   
**Docker hub is a default registry provided by the Docker itself and here you can see a lot of different images.  
------------------------------------------------------------------------------------------------------  
** Docker **will use just the application layer**, but the vm will use the application layer + operating system and kernel layer.

**So in result, Docker will be having a much lower file sizes and it will be very portable to move around to deploy on any of the environments.  
  
[2]** **A blackboard with green lines and numbers

AI-generated content may be incorrect.**  
So here in the above application two containers are there in Docker but with host we can’t connect with two ports 8080 at same time so we configure host port with two ports 8080 and 8081 and they can connect with port 8080 docker port .  
So there will be 2 Combinations : host port 8080 and Docker port 8080  
another host port 8081 and Docker port 8080  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Main Docker Commands :   
A black rectangle with white text and a hand pointing at it

AI-generated content may be incorrect.  
  
that is Docker pull 🡪will download and install or start the container in your local machine.  
Like this   
A screen shot of a computer

AI-generated content may be incorrect.  
  
A black screen with white text

AI-generated content may be incorrect.  
  
docker images -> will show you the list of images installed on system  
A black screen with white text

AI-generated content may be incorrect.  
So Docker run command is used to run docker image 🡪 the command is made up of  
docker run –name container-name img:tag (instead of tag you can use image id as well)  
  
dockers ps 🡪 will show you the list of containers available   
from here you can know which port is exposed for that this is docker port not host port.  
A black background with white text

AI-generated content may be incorrect.  
For host port we will add extra configuration as well:later   
A black background with white text

AI-generated content may be incorrect.  
docker stop 🡪 will stop container docker stop container   
A black background with white text

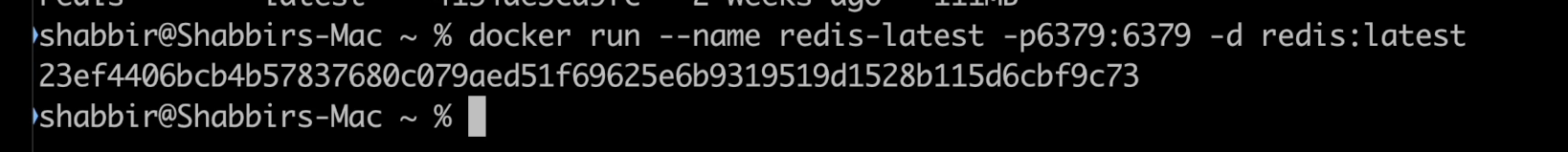
AI-generated content may be incorrect.  
A black background with white text

AI-generated content may be incorrect.  
Now if you run docker ps you you can see no information.  
  
A black background with white text

AI-generated content may be incorrect.  
docker ps -a 🡪 will show you all containers i.e. A black screen with white text

AI-generated content may be incorrect.  
docker start 🡪 will start container again with containerId.  
**A screenshot of a computer

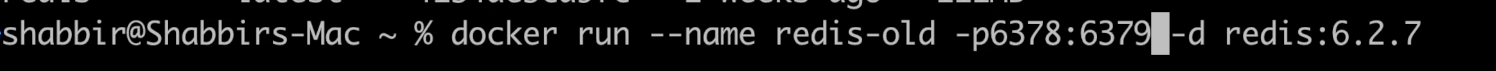
AI-generated content may be incorrect.**  
**so to remove container we will stop images by using docker stop containerId   
and then use 🡪 docker rm containerId  
A screenshot of a computer

AI-generated content may be incorrect.  
see your container is deleted but your image is still there .  
  
docker run –-name -phostPort:dockerPort -d nameContainer:Type 🡪 adding host port   
  
-d 🡪 will run application in detach mode means you can run your application in background and we can run other applications as well .  
A screenshot of a computer

AI-generated content may be incorrect.  
now the docker is exposed to hostPort .**

**Docker pull redis 🡪 will take latest   
  
docker pull redis:version number   
--------------------------------------------------------------------------------------------------  
Now under Docker images 🡪 will have two sections  
A screenshot of a computer

AI-generated content may be incorrect.  
Now suppose you want to use old version you name it as redis-old and redis:tagName as 6.2.7  
A black screen with white text

AI-generated content may be incorrect.  
  
  
Earlier we have used the 6379 as well so this time we change host port to 6378 so that both our containers get run   
A screenshot of a computer program

AI-generated content may be incorrect.  
Now from docker-ps command we can get container running information.  
  
docker –-help will tell you the more information about commands .  
-----------------------------------------------------------------------------------------------------  
Debugging in Docker :  
A screenshot of a computer

AI-generated content may be incorrect.  
[1]docker inspect ImageID : will tell you all the information about this image .**That also that information will also get you will get the information about what **environment variables** are set for that particular image.

**You will get to see what are the commands added, like which commands will run when the container should**

**stats for this particular image?**

**Is there any volumes attached?**

**What is a working directory?**

**What are the entry point commands?  
All these informations you will get from docker inspect command .  
[2]** **A screenshot of a computer

AI-generated content may be incorrect.**  
Same thing you can follow for container as well   
**You can see that this is the container ID when it was created, the part where the entry point command**

**has been added.**

**What are the arguments?**

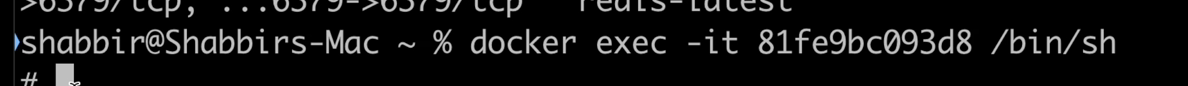
**What is the state of the container?**

**Currently, it's running right when it was started.**

**What is the image used?**

**And you will also get the information regarding the IP address.  
[3] docker logs containerId 🡪 will show you the logs available for the containerId  
A screenshot of a computer

AI-generated content may be incorrect.  
[4] docker exec containerId 🡪 will enable you to go to inside container and execute commands inside container   
-it 🡪 means interactive command .  
A screen shot of a computer

AI-generated content may be incorrect.** ****  
we have used /bin/sh 🡪 now we are inside container.  
**A black screen with white text

AI-generated content may be incorrect.**  
we are under /data folder   
**A black background with white text

AI-generated content may be incorrect.**  
These are all the folders container itself.  
**A screenshot of a computer

AI-generated content may be incorrect.**  
now here redis cli 🡪 will allow you to login inside container   
here keys command 🡪 will give you all the available keys   
**A screenshot of a computer code

AI-generated content may be incorrect.**  
From exit command to get out.  
**A screen shot of a computer

AI-generated content may be incorrect.**  
this home directory 🡪 ls will show you all the host machines