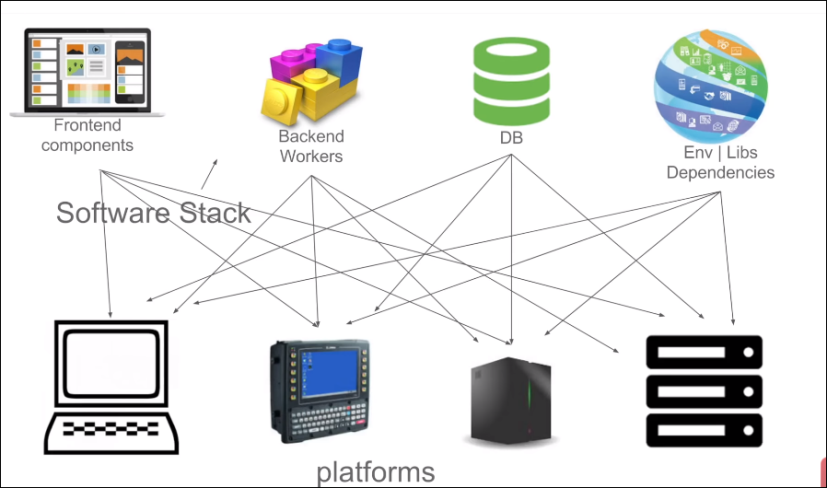
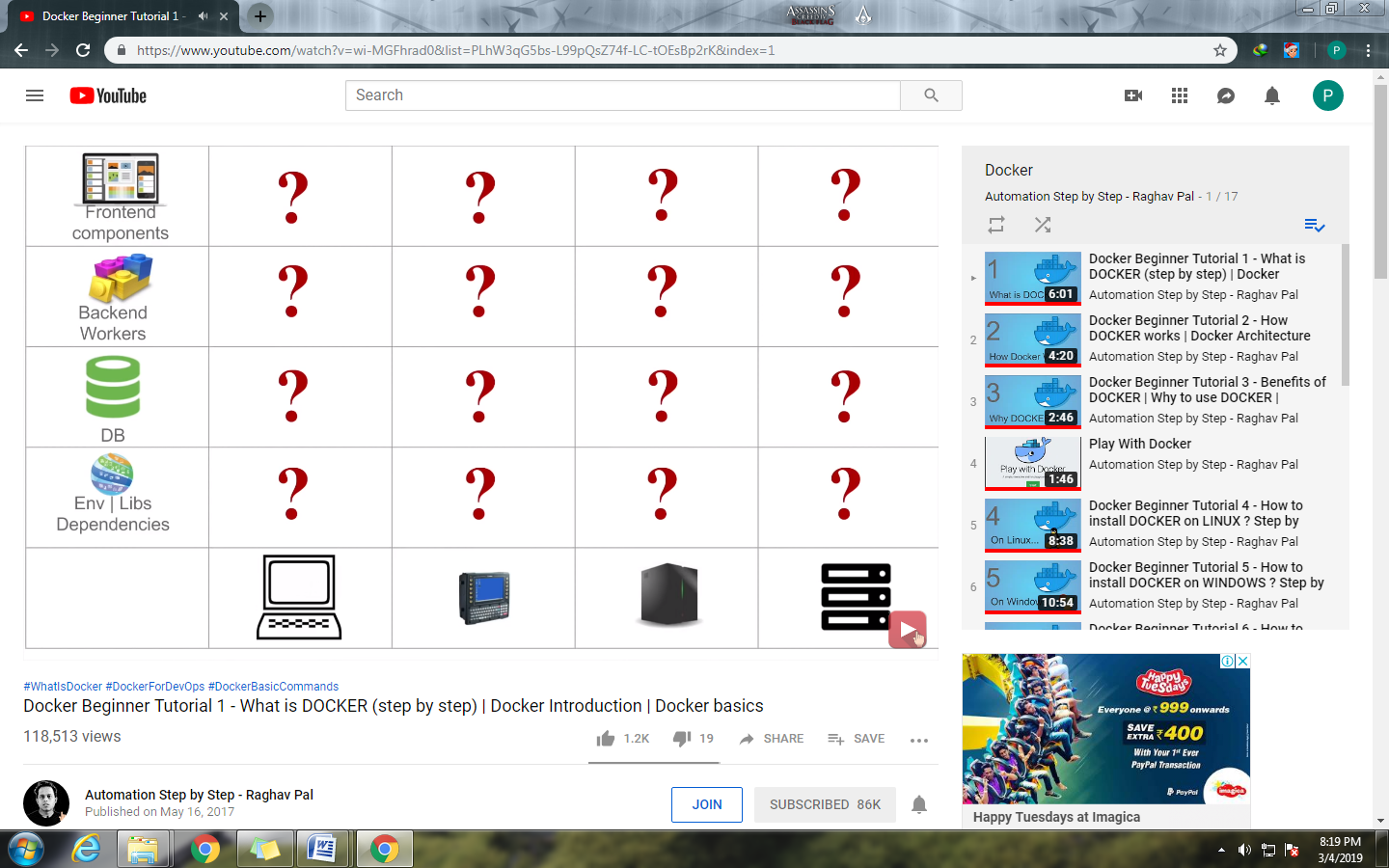
**Docker**

**Introduction:**

Docker Comes into picture at deployment stage, Docker makes the process of application deployment very easy and efficient and resolves lot of issues related to deploying applications

**Problem statement for docker:**

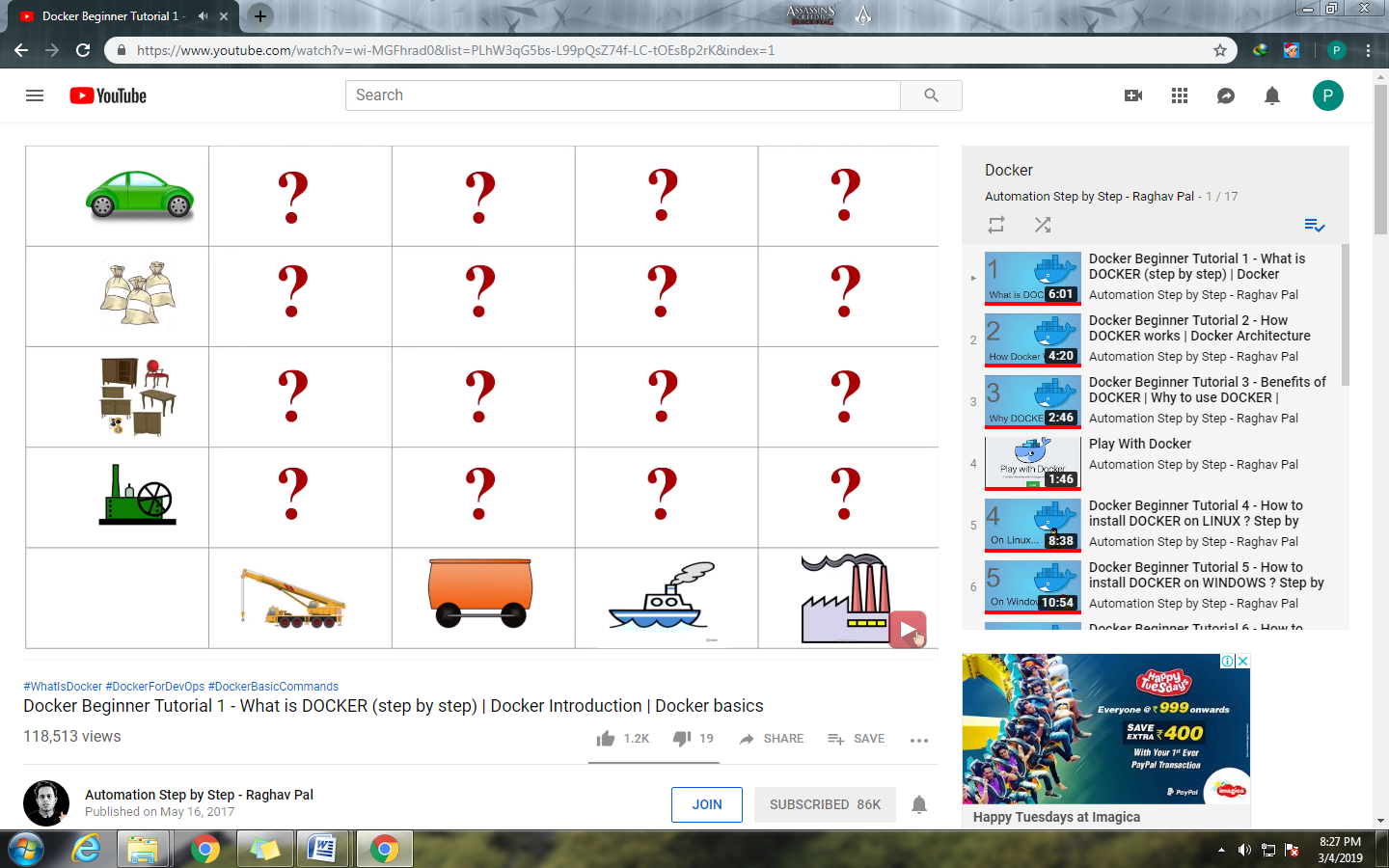
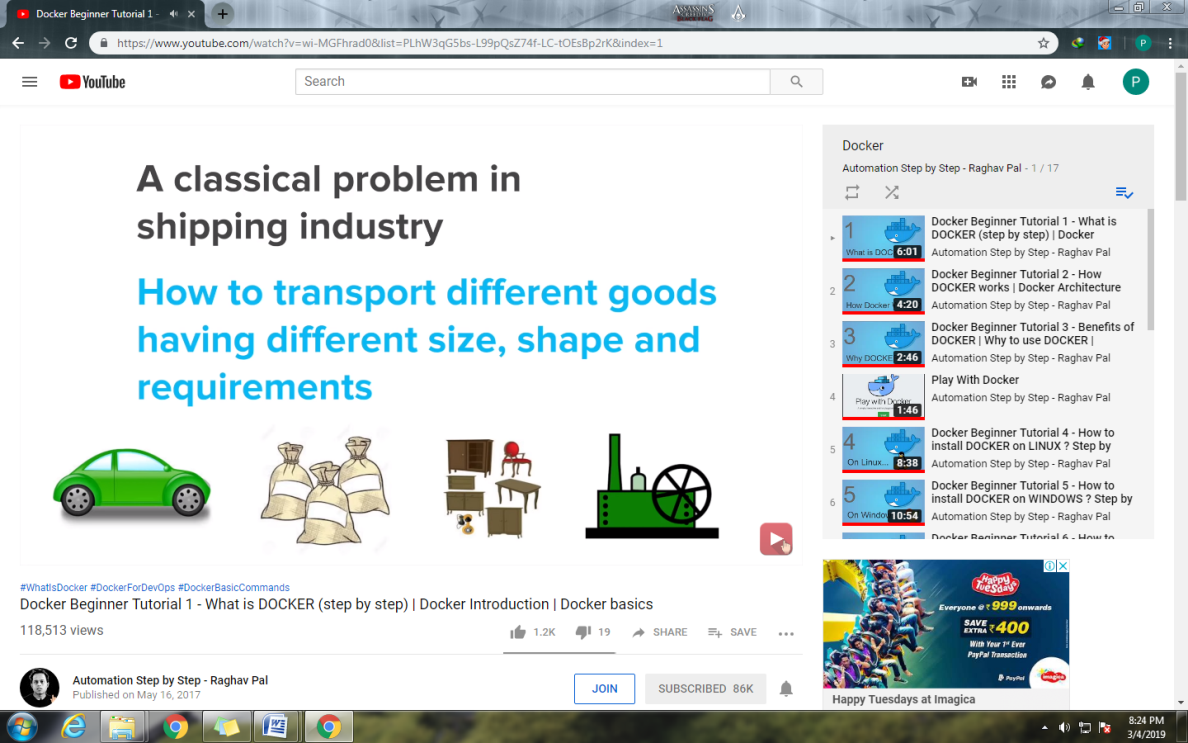
A typical software application consist of frontend component, back end, databases env, libs/dependencies. we have to make sure that these component should run on different types of platforms like desktop, laptop etc. It looks like very complicated structure like below, and it is very difficult to ensure software application run on each platform.

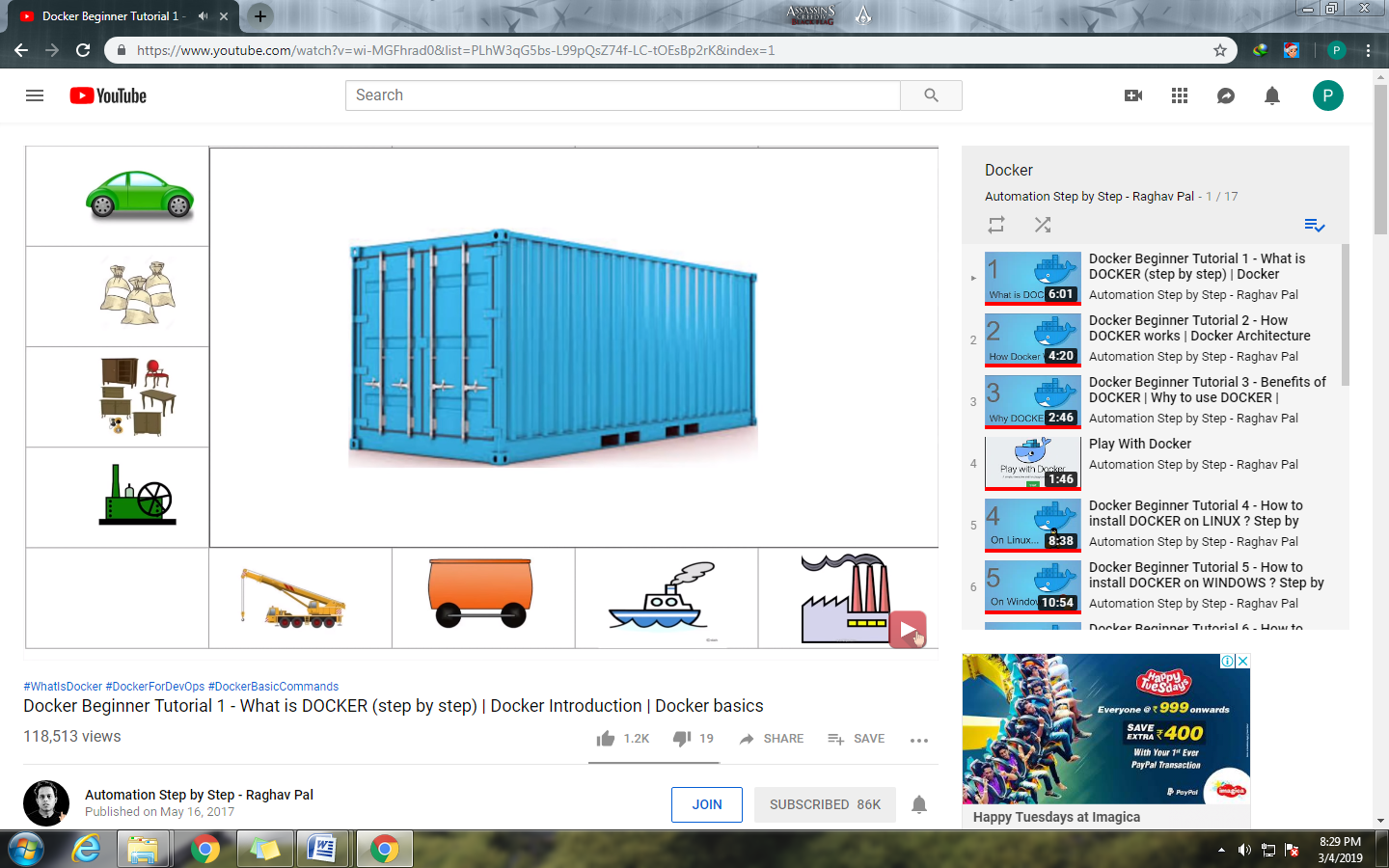


Now a days in many organizations developers and operations teams spend lots of time on ensuring the application run on each platform

Solution to these problem:

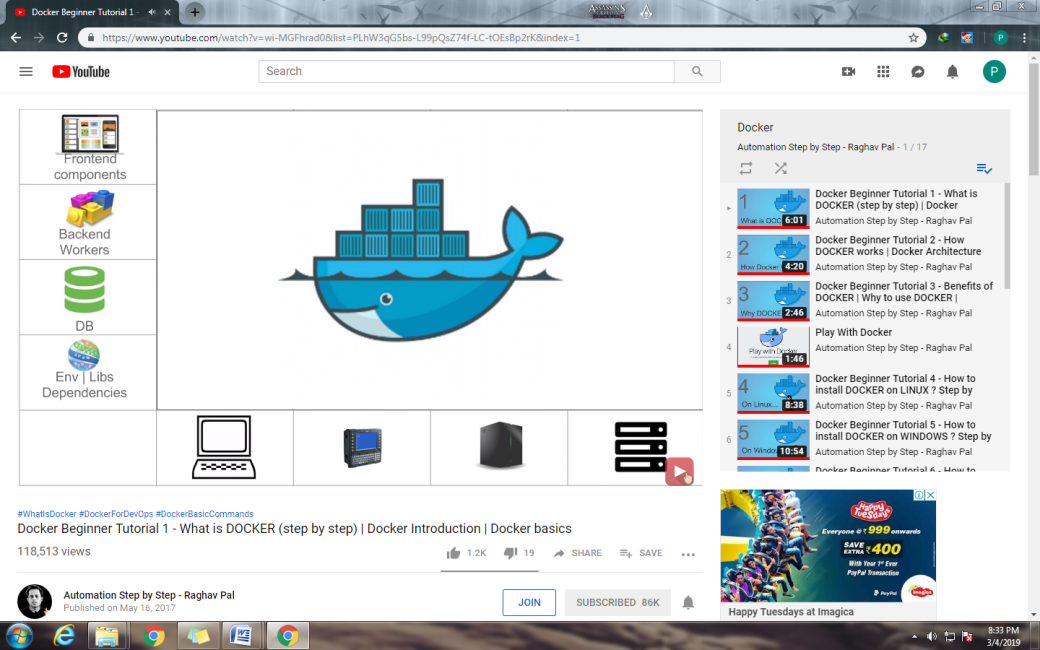
Ex: shipping problem

To solve above shipping problem containers are used as shown below:



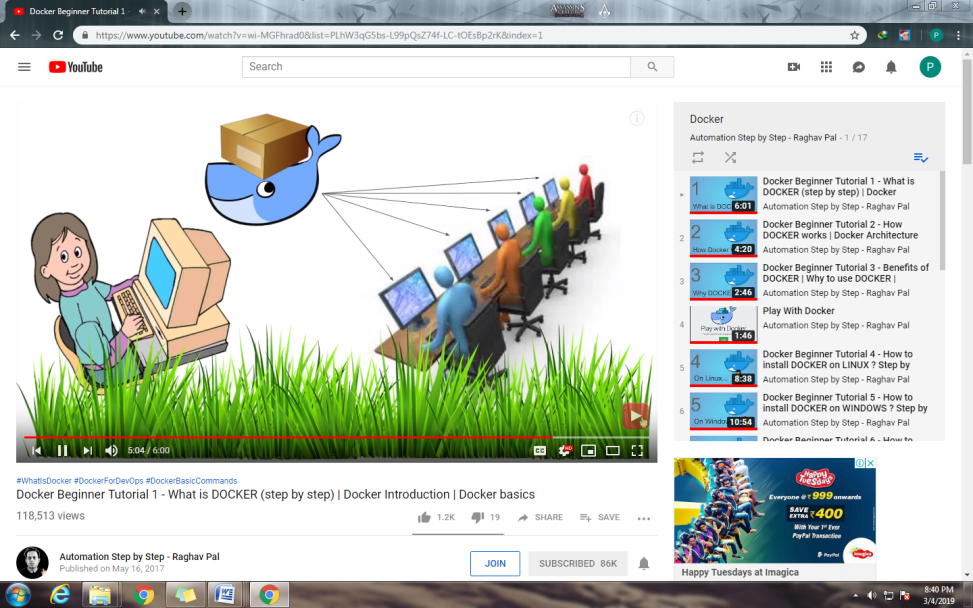
If you want to transfer the furniture, you only have worry about how to packaging the furniture in container. You have no need to worry about the how its transfer by road, water, & no need to worry about the expertise of workers who handle this containers.

So same thing can be applied to above software matrix similar like shipping industry containers so concept of dockers comes into picture

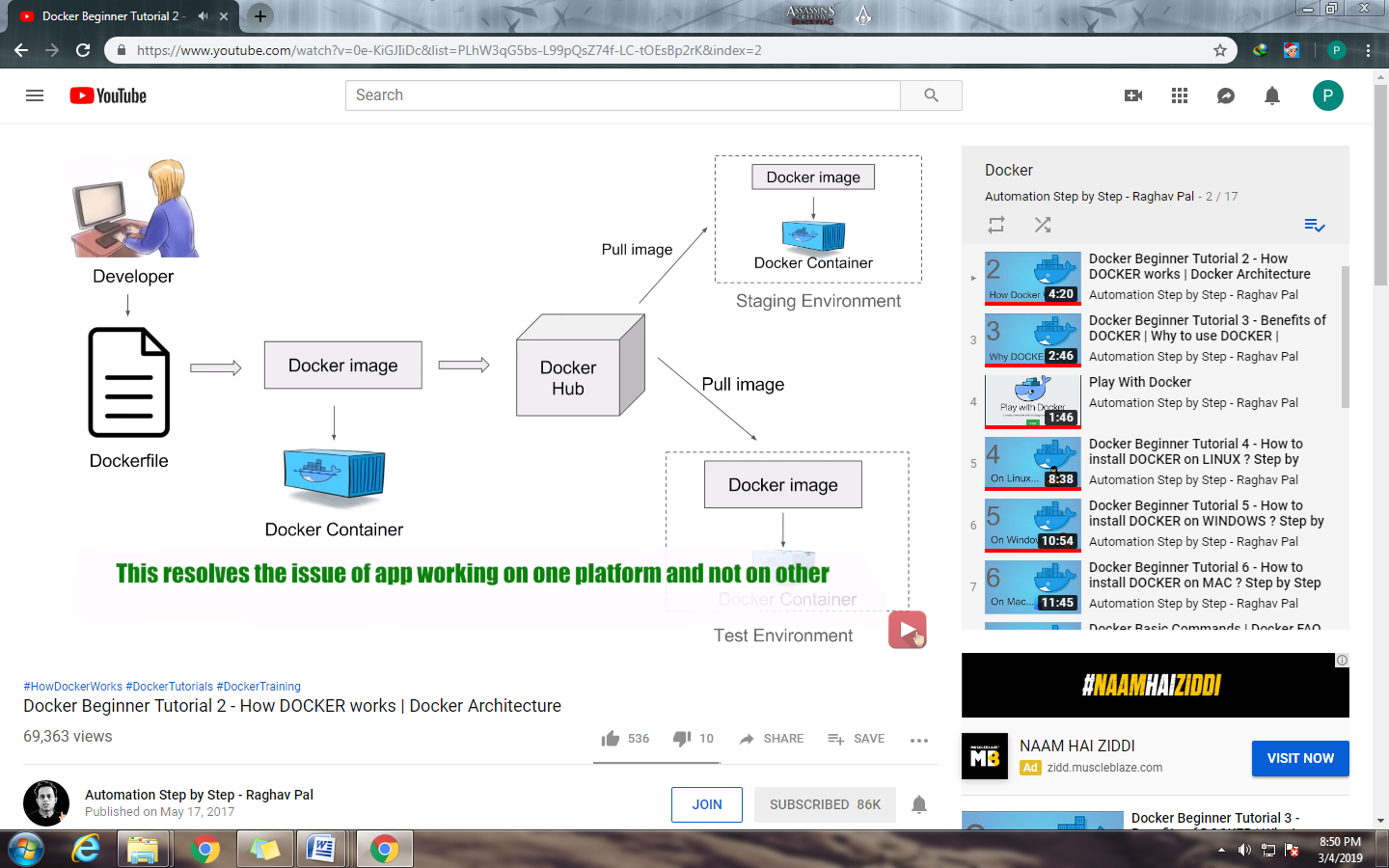


**What is Docker?**

* Docker is a containerization platform which packages application and all it's dependencies together in form of container.
* Docker containers wrap a piece of software in complete file system that contains everything needed to run: code, runtime, system tools, system liberies etc.
* This guarantees that the software will always run the same, regardless of environment.

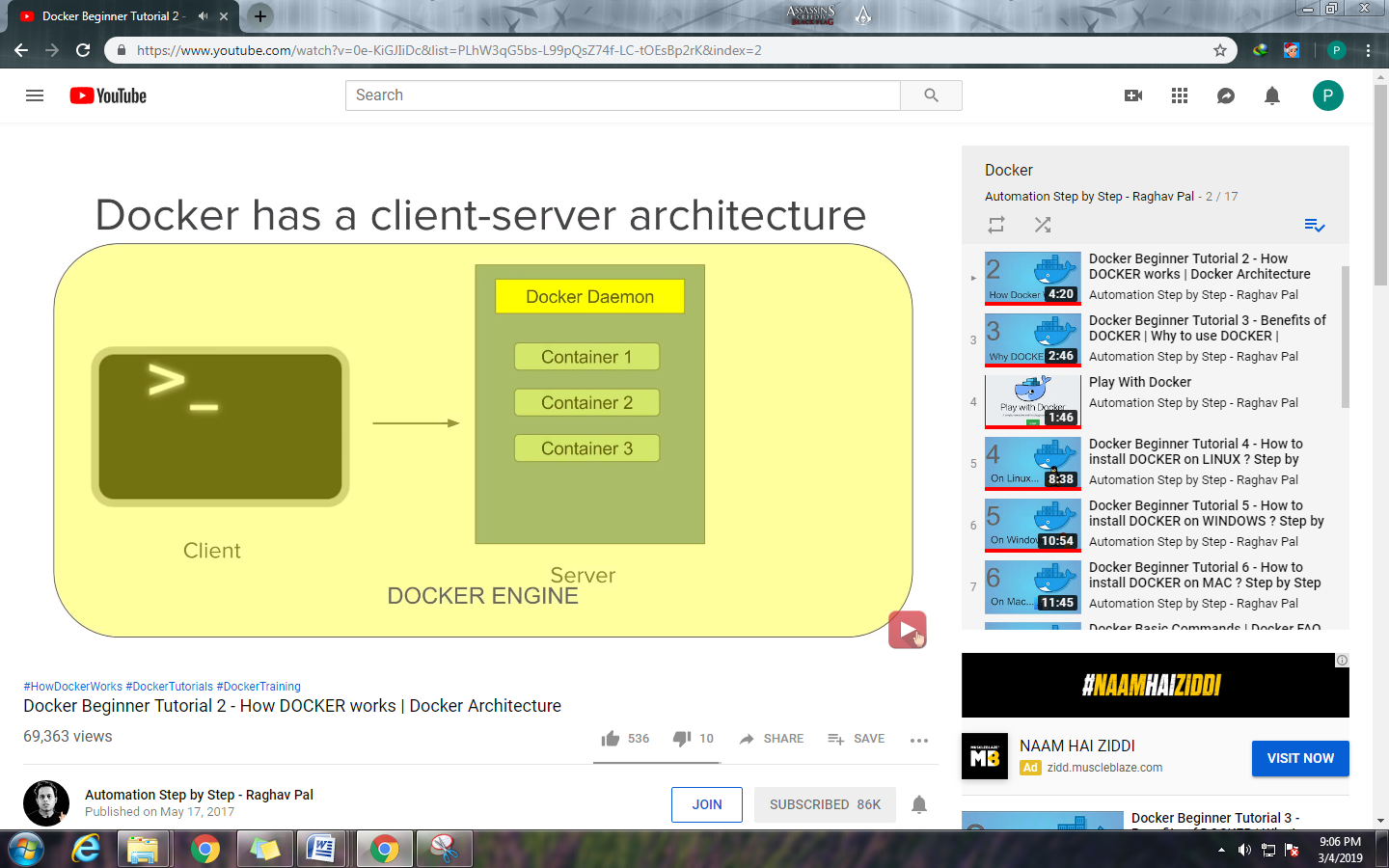


**General workflow of Docker:**



|  |  |
| --- | --- |
| Virtualization | Containerization |
|  |  |
| Higher utilization of resources | Low utilization of resources |
| Consumes higher memory as on each VM has its own OS. | Consumes less memory |
| Each OS has its own kernel | Uses shared kernel |
| Boot up very slow | Boot up faster |
| Different types of OS on hypervisior | Different types of containers on docker engine. |
| Different Kernels per guest OS are used using shared hypervisor | It uses special features of Unix file system to create each isolated environment. |

|  |  |  |
| --- | --- | --- |
| **Docker vs Hypervisors** | | |
| **Features** | **Hypervisors** | **Docker** |
| *Default Security Support* | To a great degree | To a slightly less degree |
| *Memory on disk required* | Complete OS plus apps | App requirement only |
| *Time Taken to start up* | Substantially longer as it requires boot of OS plus app loading | Substantially shorter as apps only need to start as the kernel is already running |
| *Portability* | Portable with proper preparation | Portable within image format; typically smaller |
| *Operating System* | Supports multiple OS | It uses the host OS |



Docker Client:

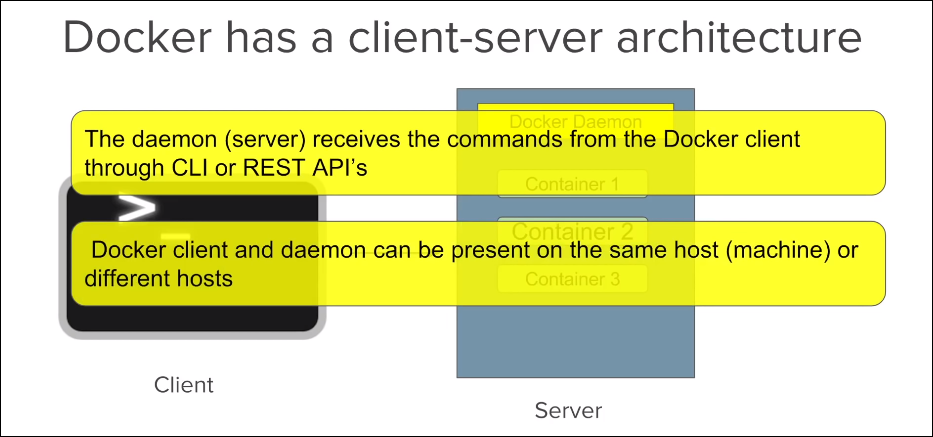
* Command line interface is the client

Docker Server or Docker Daemon:

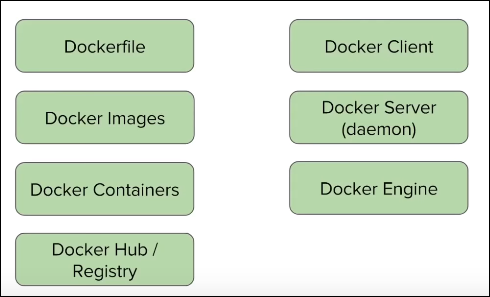
* Which contains all the containers.
* Docker server receives command from docker client in the form of commands or rest api request.

Docker Engine:

* Docker client and Doker Daemon/Server together called as Docker Engine.



Basics of Docker



Advantages of Docker:

**Build app only once**: An application inside a container can run on any system that has docker installed. So there is no need to build and configure app multiple times on different platforms.

with docker you test your application inside a container and ship it inside a contianer. This means the environment you test is identical to one on the which the app will run in production.

**Portability**: Docker container can run on any platform. It can run on local machine Amazon ec2, virtual box, rackspace, google cloud platform etc.

ex: A container running on AWS can easily be ported to virtual box.

**Isolation:**

* With docker every application works in isolation in its own container and does not interferes with other application running on same system.
* For removal also you can simply delete the container and it will not leave behind any files or traces on system.

**Productivity:**

Docker allows faster and more efficient deployments without worrying about the running application on each environment.

What are images?

Images are the template used to create Docker Containers. Container is running instance of image.

Images are stored on local or registry (hub.docker.com)

To pull docker image from registry use below command with options:

docker pull <image\_name> ex: docker pull Ubuntu

docker pull <imagename:tag> ex: docker pull Ubuntu:18.04

|  |  |
| --- | --- |
| Command | Description |
| docker images --help | Gives info about all the available options of docker image command |
| docker images | List all the available images. |
| docker images -q | Only shows numeric ids |
| docker images –f “dangling=false” | Filter output on given condition.  Dangling image is one that is not tagged and is not referenced by any container. |
| docker inspect<image\_name|ID> | List the details of images |
| docker rmi <image\_name|ID>:<tag(optional)> | To remove docker image use –f option to remove image forcefully |
| docker history <image\_name> | Shows the history of image |

What are container?

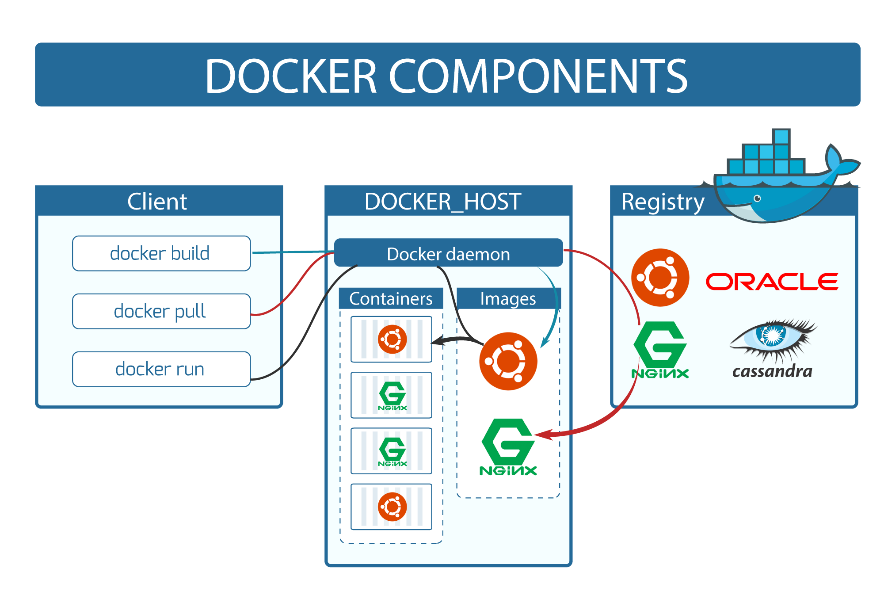
Running instances of docker images.

A container image is lightweight, stand-alone, executable package of piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.

Features:

1. Lightweight
2. Uses less resources
3. Booting of container is very fast
4. Can start stop kill remove containers easily and quickly
5. OS resources shared within docker
6. Container run on same machine sharing same OS kernel. This makes it faster

Docker architecture:



Docker Client: console used for running docker commands is called as docker client. Incase of windows command prompt, for mac it is terminal, and for linux it is bash shell

Docker Host: It consist of-

1. Docker demon
2. Container
3. Images

Registry: it can be on local or on remote (eg. hub.docker.com)

Commands:

|  |  |
| --- | --- |
| Command | Description |
| docker run –name myContainer -it <image\_name> | It will run the image in interactive mode, it will first check image locally. If image not found locally then it searches in registry (hub.docker.com) and downloads that image and then run that image to create container |
| docker ps | List all the containers |
| docker pull image\_name | Used to pull image from registry  (eg. hub.docker.com) |
| docker start <container\_name|ID> | Starts the container |
| docker stop<container\_name|ID> | Stops the container |
| docker pause <container\_name|ID> | Pause the running container, so it will not perform any operations. |
| docker unpause <container\_name|ID> | Unpauses the running container, so it will starts performing operations again. |
| docker top<container\_name|ID> | Shows top processes of container |
| docker stats<container\_name|ID> | Gives stats of running container |
| docker attach<container\_name|ID> | To attach running container to current command prompt |
| docker kill<container\_name|ID> | Kills the running container |
| docker rm<container\_name|ID> | Remove the container |

Create Jenkins on Docker:

Step 1: pull jenkin image from docker hub (hub.docker.com) **docker pull jenkins**

Step 2: run the jenkin container

docker run -p 8080:8080 -p 50000:50000 -v /your/home:/var/jenkins\_home jenkins

Step 3: how to set jenkins home on Docker volume and Host machine

docker run --name myjenkins -p 8080:8080 -p 50000:50000 -v /var/jenkins\_home jenkins

docker volume create myjenkins

docker run -p 8080:8080 -p 50000:50000 -v myjenkins:/var/jenkins\_home jenkins

Volume is helpful when you want to share between containers keep your data unchanged.

Docker file:

A text file with instrunctions to build image, automation of docker image creation.

Step 1: Create a the file named Dockerfile

Step 2: Write instructions

FROM: getting base image you can use existing image or empty image

ex: FROM Ubuntu or FROM scratch (scratch is used for empty image from hub.docker.com)

MAINTAINER: mail id of dockerfile maintainer

RUN: it will executed during building of image.

ex : RUN apt-get update (gets the update from docker hub for base image)

CMD: command that only gets executed when you create container out of that image.

Ex: CMD [“echo”,”Hello World…! From my first docker image”]

Step 3: build dockerfile to create image

docker build –t imageName:tag directoryofDockerfile

Step 4: Run image to create container.

docker images

docker run <imageID>

reference: [https://github.com/wsargent/docker-ch...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Fwsargent%2Fdocker-cheat-sheet%23dockerfile&redir_token=lfu-hdErx5XacvLdVHJ_QqLEfvZ8MTU1MTc2MjU4NUAxNTUxNjc2MTg1&v=LQjaJINkQXY&event=video_description)

[https://docs.docker.com/engine/refere...](https://www.youtube.com/redirect?q=https%3A%2F%2Fdocs.docker.com%2Fengine%2Freference%2Fbuilder%2F%23environment-replacement&redir_token=lfu-hdErx5XacvLdVHJ_QqLEfvZ8MTU1MTc2MjU4NUAxNTUxNjc2MTg1&v=LQjaJINkQXY&event=video_description)

What is Docker Compose?

* Tool for defining & running multi container docker applications
* Use yml files to configure application services (docker-compose.yml)
* Can start all services with single command: docker compose up
* Can stop all services with single command: docker compose down
* Can scale up selected services when ever required.

Step 1: install docker compose (already installed on windows and mac with docker) docker-compose -v

2 Ways to install docker on linux

1. [https://github.com/docker/compose/rel...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Fdocker%2Fcompose%2Freleases&event=video_description&v=HUpIoF_conA&redir_token=FdCB7d66IuJ5eaeeEavzJv5hFXB8MTU1MTc2NDA3MUAxNTUxNjc3Njcx)

2. Using PIP pip install -U docker-compose

Step 2: Create docker compose file at any location on your system docker-compose.yml

Step 3: Check the validity of file by command docker-compose config

Step 4: Run docker-compose.yml file by command docker-compose up -d

Step 5: Bring down application by command docker-compose down

TIPS How to scale services —

scale docker-compose up -d --scale database=4

References: [https://hub.docker.com](https://www.youtube.com/redirect?q=https%3A%2F%2Fhub.docker.com&event=video_description&v=HUpIoF_conA&redir_token=FdCB7d66IuJ5eaeeEavzJv5hFXB8MTU1MTc2NDA3MUAxNTUxNjc3Njcx)[https://github.com/docker/compose/rel...](https://www.youtube.com/redirect?q=https%3A%2F%2Fgithub.com%2Fdocker%2Fcompose%2Freleases&event=video_description&v=HUpIoF_conA&redir_token=FdCB7d66IuJ5eaeeEavzJv5hFXB8MTU1MTc2NDA3MUAxNTUxNjc3Njcx)[https://docs.docker.com/compose/compo...](https://www.youtube.com/redirect?q=https%3A%2F%2Fdocs.docker.com%2Fcompose%2Fcompose-file%2F&event=video_description&v=HUpIoF_conA&redir_token=FdCB7d66IuJ5eaeeEavzJv5hFXB8MTU1MTc2NDA3MUAxNTUxNjc3Njcx)

Docker Commands: more info refer https://docs.docker.com/engine/reference/commandline/ps/

Docker Basic Commands:

|  |  |
| --- | --- |
| Command | Description |
| Basic | |
| docker version | Give info about docker client and docker server which is the engine. |
| docker -v / docker --version | Gives the version of docker |
| docker info | Gives detailed info about docker installed on your sysytem. |
| docker --help | Gives info about all docker commands. use docker [command\_name] --help to fetch details about particular docker command. |
| docker login | Used to login in https://hub.docker.com |
| Images | |
| docker images | Gives the list of images. |
| docker pull | Used to get the images from repository (code repository or https://hub.docker.com) |
| docker rmi | Used to remove one or more images. |
| Containers | |
| docker ps | List all the running containers. |
| docker run <images name> | To run the container. If image not present locally then it downloads the image from docker hub and then it starts the container |
| docker start <container\_id> | Starts container by container id |
| docker stop <container\_id> | Stops container by container id |
| System | |
| docker stats | Shows memory usage by container |
| docker system df | Shows disk usage of docker |
| Docker system prune --a | Removes all the unused images, container from docker. Use this command very carefully. |