**Docker:** Docker makes the application deployment very easy and efficient and resolves lot of issues related to deploying applications.

**Docker is present everywhere in the workflow. But its main use is in deployment stage.**

* Docker is a tool designed to make it easier to deploy & run applications using containers.
* Containers allow developers to package their application with all the necessary libraries, dependencies together as one and is deployed across various platforms

An application inside a container can run on any system that has Docker installed. We need not build and configure applications multiple times on different platforms.

**Benefits of Docker:**

* Portability is a great feature with Docker. For instance, if we created a container using Amazon EC2 machine, we can easily port this container to virtual box, and it will run as is. **A container running on EC2 can be easily ported to virtual box.**
* Docker has built-in version control system. Docker containers work just like GIT repositories allowing to commit changes to the images and version control them.
* With Docker, every application works in isolation in its own container and doesn’t interfere with other applications running on the same system.
* Docker simplifies the deployment of applications.

**Docker Commands:**

**Basic:**

* docker version : Gives the information about the version of Docker client & server
* docker info : Gives detailed information about number of containers, images etc
* docker -- help : Can be used to know the usage and also list of options with each keyword. For instance, docker images -- help
* docker login : can be used to login to docker hub

**Images:**

* docker images : Can be used to list the images
* docker pull : Can be used to pull any image available. (docker pull mongodb)
* docker rmi : Can be used to remove one or more images (docker rmi image-id)
* docker images -f “dangling=false” : To filter the images
* docker inspect [image-id or name] : Get info about the image (tags, layers)

**Containers:**

* docker ps : List the containers.
* docker run : To run the image. ( docker run ubuntu ) Image has to be already installed prior to execute this command.
* docker run -it ubuntu : In interactive mode we enter ubuntu
* docker start [container-id] : To start the container.
* docker stop [container-id] : To stop the container.

**System:**

* docker stats : Shows the memory usage of containers.
* docker system df : To check the disk usage of container.
* docker system prune : \*\* **Be careful with this command as it is used to remove unused images which are not associated with any running containers and dangling images (untagged images)\*\***

**Docker Images:**

* Docker images are the templates used to create Docker containers. Container is a running instance of an image.
* Docker can build images automatically by reading the instructions from a Docker file.
* A single image can be used to create multiple containers.

**Docker Containers:**

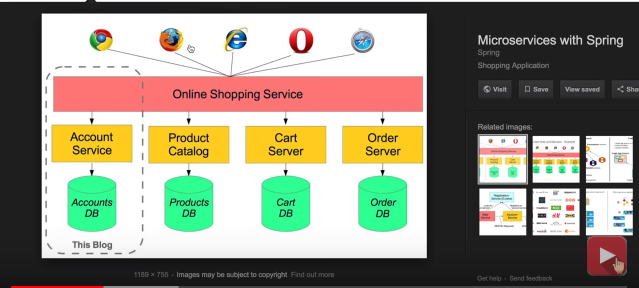
* Containers are the running instances of images.
* docker run [image-name | id] : It will check if the image exists, if not it will pull from the docker hub and store it in the local host memory.

**Docker File:**

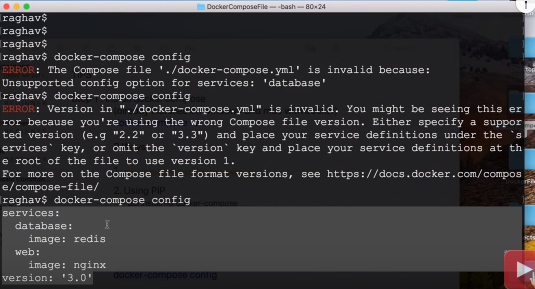
* Docker File is a simple text file with instructions to build a docker image.
* RUN: Run gets executed during the building of the image.
* CMD: Command given inside CMD gets executed only when the container is created out of the image.
* We can build the docker image with the command docker build [docker file location]
  + docker build -t myImage1:1.0 and then run
* **Steps for building an image are as follows:**
  + Create a Docker file.
  + Add Instructions in Docker file.
  + Build docker file to create an image
  + Run image to create container.

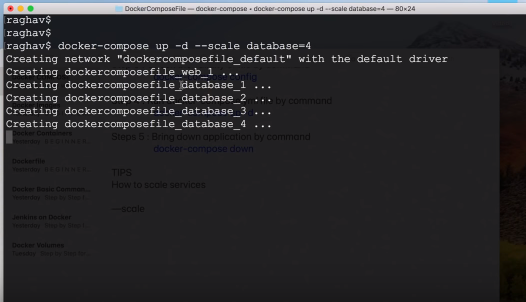
**Docker Compose:**

* Tool for defining and running multi-container docker applications.
* We use yaml files to configure application services. (docker-compose.yml)



* Create docker compose file at any location with the file name: (docker-compose.yml)
* Docker compose
  + : tool for defining & running multi-container docker applications
  + : use yaml files to configure application services (docker-compose.yml)
  + : can start all services with a single command : docker compose up
  + : can stop all services with a single command : docker compose down : can scale up selected services when required.





**Docker Volumes:**

* Volumes are the preferred mechanism for persisting data generated by and used by Docker containers
* Decoupling container from storage
* Share volume (storage/data) among different containers
* Attach volume to container
* On deleting container volume does not delete