Docker:

Docker container is simply another process on your machine that have been isolated from all other processes on the host machine. That isolation leverages kernel namespaces and cgroups of Linux feature. Docker made these capabilities approachable and easy to use.

Container Image: an isolated filesystem provided by container image, it contains everything it needs to run an application – all dependencies, configurations, scripts, binaries, etc. the image also contains other configurations for the container, such as environment variables, a default command to run and other metadata.

1. Get started. Open Docker Desktop, this will start docker deamon process
2. Some common docker commands:
3. Docker run -dp 80:80 myimage:label
4. Create a docker file, then run # “docker build -t getting-start .” to build the image in the current directory.
5. Docker ps; docker stop <containerId>; docker rm -f <the-container-id>
6. Create repository in Docker Hub. To publish this image: “docker push your-username/getting-started. Before push, run docker tag getting-started your-username/getting-started;
7. Container Volumes:
8. Fds
9. Container Volumes: (named volume)

Provide the ability to connect specific filesystem paths of the container back to host machine. If a directory in the container is mounted, changes in that directory are also seen on the host machine. And if we mount that same directory across container restarts, we’d see the same files.

1. Named volumes: by creating a volume and attaching (often called “mounting”) it to the directory the data is stored in, we can persist the data.
2. Create a volume by using: “docker volume create todo-db”
3. Docker run -dp 3000:3000 -v todo-db:/etc/todos getting-started
4. To check the information on the created volume: “docker volume inspect todo-db”; you should be able to see “Mountpoint” on the host machine.
5. Container Volume: ( bind mounts )
6. With bind mount, we control exact mountpoint on the host.
7. With bind mount, we can provide additional data into containers, like source code changes. So it’s good to set up the local dev env.
8. Multi-Container Apps: each container should do one thing and one thing well.
9. Container Networking: are the two containers in the same network? There are two ways to put a container on a network: 1. Assign it from the start. 2. Connect an existing container. Better to use nicolaka/netshoot.
10. Docker compose: need dockerfile; docker-compose.yml

docker-compose version;

docker-compose up -d

docker-compose logs -f app

docker-compose down ( remove volume with –volumes)

1. Image Building Best Practices:
2. Security Scanning: docker scan imageName
3. Image Layering: docker image history imageName;

Use “.dockerignore files to selectively copy only image relevant files.

1. Multi-Stage Builds:
2. What’s next: http://localhost:8080/tutorial/what-next/
3. fds

Even though the vast majority of enterprise businesses use Kubernetes

Docker (or specifically, the docker command) is used to manage individual containers, docker-compose is used to manage multi-container applications and Kubernetes is a container orchestration tool.

* Docker – when you want to deploy a single (network accessible) container
* Docker Compose – when you want to deploy multiple containers to a single host from within a single YAML file
* Docker swarm – when you want to deploy a cluster of docker nodes (multiple hosts) for a simple, scalable application
* Kubernetes – when you need to manage a large deployment of scalable, automated containers

Graphics Processing Unit (GPU)

Docker Compose Tutorial:

<https://youtu.be/HG6yIjZapSA>

useful command:

remove all images:

docker image rm $(docker image ls -q)

same for the containers:

docker container rm $(docker container ls -q)

data.json and data.yml or yaml

json file for exchange data: Object {}

yml file for configuration: start with ----; use indentation, - and no need for “”

docker exec -it -u root 8c6 sh ( to get into a container )

<https://docs.docker.com/compose/gettingstarted/>

docker-compose.yml with dockerfile example.

Maven vs Jenkins:

Maven is a build tool that manages dependencies and the application life cycle. It also had a plug in design that allows you to add other tasks to the standard compile/test/package/install/deploy tasks.

Jenkins is a continuous integration suite that checks your code out of a repository, builds and packages it, and dumps it out to a server so you can test it - all hands-off. It can use Maven or Ant as its build tool.

In summary, Jenkins can use Maven as its build tool for continuous integration. You can use Maven without Jenkins if you choose not to do CI.

Docker comes with an embedded DNS server that contains the name and ip of those containers now inside. ( DNS resolve talks to DNS server to find target container )

Docker-Compose:

Using Compose is basically a three-step process:

1. Define your app’s environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker compose up and the [Docker compose command](https://docs.docker.com/compose/#compose-v2-and-the-new-docker-compose-command) starts and runs your entire app. You can alternatively run docker-compose up using the docker-compose binary.

Compose has commands for managing the whole lifecycle of your application:

* Start, stop, and rebuild services
* View the status of running services
* Stream the log output of running services
* Run a one-off command on a service

<https://docs.docker.com/compose/cli-command-compatibility/>

Dockerfile:

<https://www.baeldung.com/spring-boot-docker-images>

can use eclipse or maven command line: mvnw spring-boot:build-image.

**4.2. Viewing and Extracting Layers**

Next, we need to extract the layers from the jar so that the Docker image will have the proper layers.

To examine the layers of any layered jar, we can run the command:

java -Djarmode=layertools -jar demo-0.0.1.jar list

Then to extract them, we would run:

java -Djarmode=layertools -jar demo-0.0.1.jar extract

Related Software: https://maven.apache.org/run-maven/index.html