Docker

Docker is a containerization platform that encapsulates an application and its dependencies into a container, ensuring consistent operation across different computing environments. It leverages OS-level virtualization to deliver software in packages called containers, providing isolation and resource efficiency, and facilitating CI/CD practices by streamlining deployment and scaling.

Installation

Docker can be installed on different operating systems. For local workstations, Docker Desktop is the recommended installation. For servers, Docker Engine is the recommended installation.

Docker Desktop

Docker Desktop is a software application that enables developers to build, package, and run applications using Docker containers on their local machines. It provides an easy-to-use graphical interface and includes the necessary tools and components for managing Docker containers, such as the Docker engine, images, and networking capabilities.

For more information, see Docker Desktop

Install Docker Engine

One click installation script:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

Run docker as non root user:

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

For more information, see [Install Docker Engine]

Using Docker

Running Containers

COMMAND	DESCRIPTION
`docker run <image>`	Start a new container from an image
`docker run -it <image>`	Start a new container in interactive mode
`docker create <image>`	Create a new container
`docker start <container>`	Start a container
`docker stop <container>`	Graceful stop a container
`docker kill <container>`	Kill (SIGKILL) a container
`docker restart <container>`	Graceful stop and restart a container
`docker pause <container>`	Suspend a container
`docker unpause <container>`	Resume a container
`docker rm <container>`	Destroy a container

Container Bulk Management

COMMAND	DESCRIPTION
`docker stop $(docker ps -q)`	To stop all the running containers
`docker stop $(docker ps -a -q)`	To stop all the stopped and running containers

COMMAND	DESCRIPTION
`docker kill $(docker ps -q)`	To kill all the running containers
`docker kill $(docker ps -a -q)`	To kill all the stopped and running containers
`docker restart $(docker ps -q)`	To restart all running containers
`docker restart $(docker ps -a -q)`	To restart all the stopped and running containers
`docker rm $(docker ps  -q)`	To destroy all running containers
`docker rm $(docker ps -a -q)`	To destroy all the stopped and running containers
`docker pause $(docker ps  -q)`	To pause all running containers
`docker pause $(docker ps -a -q)`	To pause all the stopped and running containers
`docker start $(docker ps  -q)`	To start all running containers
`docker start $(docker ps -a -q)`	To start all the stopped and running containers
`docker rm -vf $(docker ps -a -q)`	To delete all containers including its volumes use
`docker rmi -f $(docker images -a -q)`	To delete all the images
`docker system prune`	To delete all dangling and unused images, containers, cache and volumes
`docker system prune -a`	To delete all used and unused images
`docker system prune --volumes`	To delete all docker volumes

Inspect Containers

COMMAND	DESCRIPTION
`docker ps`	List running containers
`docker ps --all`	List all containers, including stopped
`docker logs <container>`	Show a container output
`docker logs -f <container>`	Follow a container output
`docker top <container>`	List the processes running in a container
`docker diff`	Show the differences with the image (modified files)
`docker inspect`	Show information of a container (json formatted)

Executing Commands

COMMAND	DESCRIPTION
`docker attach <container>`	Attach to a container
`docker cp <container>:<container-path> <host-path>`	Copy files from the container
`docker cp <host-path> <container>:<container-path>`	Copy files into the container
`docker export <container>`	Export the content of the container (tar archive)
`docker exec <container>`	Run a command inside a container
`docker exec -it <container> /bin/bash`	Open an interactive shell inside a container (there is no bash in some
`docker wait <container>`	Wait until the container terminates and return

COMMAND	DESCRIPTION
`docker image ls`	List all local images
`docker history <image>`	Show the image history

COMMAND	DESCRIPTION
`docker inspect <image>`	Show information (json formatted)
`docker tag <image> <tag>`	Tag an image
`docker commit <container> <image>`	Create an image (from a container)
`docker import <url>`	Create an image (from a tarball)
`docker rmi <image>`	Delete images
`docker pull <user>/<repository>:<tag>`	Pull an image from a registry
`docker push <user>/<repository>:<tag>`	Push and image to a registry
`docker search <test>`	Search an image on the official registry
`docker login`	Login to a registry
`docker logout`	Logout from a registry
`docker save <user>/<repository>:<tag>`	Export an image/repo as a tarball
`docker load`	Load images from a tarball

Volumes

COMMAND	DESCRIPTION
`docker volume ls`	List all vol1umes
`docker volume create <volume>`	Create a volume
`docker volume inspect <volume>`	Show information (json formatted)
`docker volume rm <volume>`	Destroy a volume
`docker volume ls --filter="dangling=true"`	List all dangling volumes (not referenced by any container)
`docker volume prune`	Delete all volumes (not referenced by any container)

Backup a container

Backup docker data from inside container volumes and package it in a tarball archive. `docker run --rm --volumes-from <container> -v $(pwd):/backup busybox tar cvfz /backup/backup.tar <container-path>`

An automated backup can be done also by this Ansible playbook. The output is also a (compressed) tar. The playbook can also manage the backup retention. So older backups will get deleted automatically.

To also create and backup the container configuration itself, you can use `docker-replay` for that. If you lose the entire container, you can recreate it with the export from `docker-replay` . A more detailed tutorial on how to use docker-replay can be found here.

Restore container from backup

Restore the volume with a tarball archive. `docker run --rm --volumes-from <container> -v $(pwd):/backup busybox sh -c "cd <container-path> && tar xvf /backup/backup.tar --strip 1"`

Troubleshooting

Networking

```
docker run --name netshoot --rm -it nicolaka/netshoot /bin/bash
```