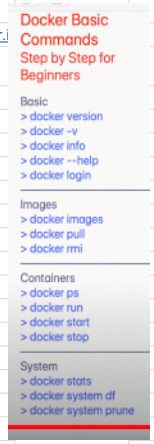
Sr No.	Basic Command
1	sudo docker version
	sudo apt-get install -y docker.i
	sudo docker-machine -v
2	sudo docker -version
3	sudo docker -v
4	sudo docker info
5	docker --help
6	docker images
7	docker images --help
8	docker run
9	docker run --help
10	sudo docker login
	sudo docker logout
11	sudo docker pull ubuntu
12	docker rmi --help
13	docker ps --help
14	docker start container_id
15	docker stop container_id
16	sudo docker stats
17	sudo docker system df
18	sudo docker system prune --he



Sr No.	Docker Images
1	docker images --help
2	docker pull image
3	docker images
4	docker images -q
5	docker images -f “dangling=false”
6	docker images -f “dangling=false” -q
7	docker run image
8	docker rmi image
9	docker rmi -f image
10	docker inspect
11	docker history imageName

Sr No.	Docker Containers Beginners
1	docker ps
2	docker run ImageName
3	docker start ContainerName/ID
4	docker stop ContainerName/ID
5	docker pause ContainerName/ID
6	docker unease ContainerName/ID
7	docker top ContainerName/ID
8	docker stats ContainerName/ID
9	docker attach ContainerName/ID
10	docker kill ContainerName/ID
11	docker rm ContainerName/ID
12	sudo docker exec -it ContainerName/ID bash
13	sudo docker exec -it ContainerName/ID bash
14	exit
15	docker history ContainerName/ID

Sr No.	Jenkins on Docker

1	docker pull jenkins	
2	docker run -p 8080:8080 -p 50000:50000 jenkins	
3	`docker run -p 8080:8080 -p 50000:50000 -v /your/home:/var/jenkins_home jenkins`	
4		
5		
Sr. No	**Dockerfile Beginners**	
1	what is docker file	A text file with instruction to build image. Automation of docker image creation
2	Steps	1) Create a file Named Dockerfile 2) Add instruction in Dockerfile 3) Build Dockerfile to create container 4)Run image to create container
3	FROM	
4	RUN	
5	CMD	
6		
7		
8		
9		

Sr. No	**Docker Compose**	1. What | Why - Docker Compose 2. How to install 3. How to create docker compose file 4. How to use docker compose file to create services 5. Basic Commands TIPS 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 Step 1 : install docker compose (already installed on windows and mac with docker) docker-compose -v 2 Ways 1. https://github.com/docker/compose/rel...
1	docker-compose -v	
2	docker-compose --version	
3	docker-compose version	
4	docker-compose --help	
5	sudo docker-compose up -d	
6	sudo docker-compose up -d	
7		
8		
9		

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

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

TIPS
How to scale services

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

Sr No.	Docker Volume
	Docker Volume
	docker Volume ls
	docker volume --help
	docker volume create
	docker volume inspect VolumeName

Today we will learn:
1. What are Volumes
2. How to create / list / delete volumes
3. How to attach volume to a container
4. How to share volume among containers
5. What are bind mounts
Volumes are the preferred mechanism for persisting data generated by and used by Docker containers

: docker volume //get information
: docker volume create
: docker volume ls
: docker volume inspect
: docker volume rm
: docker volume prune

Instead of deleting containers one by one of docker ps -a , we can use docker container prune. and for docker ps (running containers) we can use docker rm $(ps -aq)

Use of Volumes
==========
Decoupling container from storage
Share volume (storage/data) among different containers
Attach volume to container
On deleting container volume does not delete

Commands
docker run --name MyJenkins1 -v myvol1:/var/jenkins_home -p 8080:8080 -p 50000:50000 jenkins
docker run --name MyJenkins2 -v myvol1:/var/jenkins_home -p 9090:8080 -p 60000:50000 jenkins
docker run --name MyJenkins3 -v /Users/raghav/Desktop/Jenkins_Home:/var/jenkins_home -p 9191:8080 -p 40000:50000 jenkins

References
https://hub.docker.com/_/jenkins/
https://docs.docker.com/storage/volumes/

By default all files created inside a container are stored on a writable container layer

The data doesn't persist when that container is no longer running

A container's writable layer is tightly coupled to the host machine where the container is running. You can't easily move the data somewhere else.

Docker has two options for containers to store files in the host machine
so that the files are persisted even after the container stops

VOLUMES and BIND MOUNTS

Volumes are stored in a part of the host filesystem which is managed by Docker

Non-Docker processes should not modify this part of the filesystem

Bind mounts may be stored anywhere on the host system

Non-Docker processes on the Docker host or a Docker container can modify them at any time

In Bind Mounts, the file or directory is referenced by its full path on the host machine.

Volumes are the best way to persist data in Docker

volumes are managed by Docker and are isolated from the core functionality of the host machine

A given volume can be mounted into multiple containers simultaneously.

When no running container is using a volume, the volume is still available to Docker and is not removed automatically. You can remove unused volumes using docker volume prune.

When you mount a volume, it may be named or anonymous.

Anonymous volumes are not given an explicit name when they are first mounted into a container

Volumes also support the use of volume drivers, which allow you to store your data on remote hosts or cloud providers, among other possibilities.

Sr No	Docker Network
1	docker network ls
2	docker network ls --help
3	docker network --help
4	docker inspect ContainerName/Id

5	docker network inspect bridge
6	docker run --name alpine-2 --network=none alpine
7	docker network create --driver bridge --subnet 182.18.0.1/24 --gateway 182.18.0.1 wp-mysql-network
8	docker run -d -e MYSQL_ROOT_PASSWORD=db_pass123 --name mysql-db --network wp-mysql-network mysql:5.6
9	docker run --network=wp-mysql-network -e DB_Host=mysql-db -e DB_Password=db_pass123 -p 38080:8080 --name webapp --link mysql-db:mysql-db -d kodekloud/simple-webapp-mysql
10	
Sr No	**Docker Swarm**
1	docker-machine
2	docker-machine --help
	docker-machine ls
	docker-machine lp ManagerName