Docker Primer / Command Overview

After 2017, most commands take the form of docker [command] [subcommand] [options], being more specific to the intended item, (i.e., 'docker container _____', or 'docker image ____'. Prior to this, commands such as 'docker [command] [identifier]' were used, often pertaining to containers: docker attach, build, commit, cp, create, diff, events, exec, export, history, images, import, info, inspect, kill, load, login, logout, logs, pause, port, ps, pull, push, rename, restart, rm, rmi, run, save, search, start, stats, stop, tag, top, unpause, update, version, wait, etc.

Below, subcommands with names similar to Unix/ Linux commands with that name will not be explained (unless needed), their actions are predictable in context. Prune always means remove unused.

System Information
docker system events | info | df | prune | inspect | version Info displays system-wide info, events show in realtime.
docker volume create | inspect | ls | rm | prune | update Manage volumes containers can use for data (persistent storage)
docker trust inspect | revoke | sign | key [generate | load] | signer [add | remove]
docker scout cves | version Checks container layers for CVEs and gives countermeasures (or known secure images)
docker search Search Docker Hub for images

Managing networks.
docker network create | ls | rm | prune | inspect | connect | disconnect
docker container port

Container-specific commands
docker container start | stop| restart | create | rm | rename | run | kill | prune
docker container top | ps | stats | logs | port | ls
docker container pause | unpause Refers to all processes within one or more containers
docker container exec Execute a command in a running container
docker container cp Copy files/folders between a container and the local filesystem
docker container diff Inspect changes to files or directories on a container's filesystem
docker container attach Attach local standard input/output/error streams to a running container
docker container export Export a container's filesystem as a tar archive
docker container commit Create a new image from a container's changes
docker container update Update configuration of one or more containers
docker container inspect
docker container wait Blocks container(s) until they stop, then print their codes

Image-specific commands
docker image build Builds image from a Dockerfile
docker image inspect | history
docker image ls | rm | prune | tag Tag adds a label
docker image pull | push Actions with local registry/ repo
docker image load | save | import Load/save from/to tarball or stdin/out. See docs for diff of import and load
docker image tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
Note: for push and pull, use 'docker login | logout' to log in/out from a registry

Docker Compose commands (define and run multi-container applications)
docker compose build Build or rebuild services
docker compose config Parse, resolve and render compose file
docker compose start | stop| restart | create | rm | rename | run | kill | prune
docker compose cp | exec | port | ps | top | version | events | log | push | pull Same as the container equivalents
docker compose images | ls With Compose, ls lists compose projects and ls lists images containers use
docker compose up | down Pertains to starting and to stop/remove containers, networks
docker compose pause | unpause Pertains to the running containers of a service

Docker Build and Builder-X (BuildKit)
docker builder build | prune Build an image from a Dockerfile, prune removes the build cache
docker buildx bake | build Bake orders to build from a file, build directs to start a build
docker buildx imagetools create | inspect Create a new image based on source images; show details of an image
docker buildx use | create | ls | rm | stop | inspect |du | prune Apply to a builder instances; prune removes cache

---
docker plugin create | enable | disable | install | ls | rm | inspect | upgrade | push
docker plugin set -change settings (not a typo)
Plugins extend Docker's functionality and help users connect with other popular services
Create makes a plugin from a rootfs and configuration. Plugin data directory must contain config.json and rootfs directory.

docker context create | import | export | ls | rm | use | show | inspect | update Import/export works with zip or tar data
Context info is metadata specifying a name, endpoint configuration(s), TLS info, orchestrator(s), usually json files

docker checkpoint create | ls | rm
Checkpoint and Restore allows you to freeze a running container by checkpointing it, which turns its state into a collection of files on disk. Later, the container can be restored from the point it was frozen.

docker manifest create | annotate | rm | inspect | push
A manifest holds info on one image- layers, size, and digest. The command returns os and arch it was built for. A manifest list holds several image names, is intended for images identical in function for different os/arch combinations, so are often referred to as “multi-arch images”. Docker calls the command ‘experimental’ but has since 2017.

Docker Swarm/ Cluster Management Commands
Should be executed on a swarm manager node.

docker config create | inspect | ls | rm	Manage Swarm configs
docker secret create | inspect | ls | rm	Passcode for Swarm management
docker stack config | deploy | ps | ls | rm | services	List stacks with ls, services lists services, config outputs config file
docker node promote | demote | ps | ls | rm | inspect | update	Manage Swarm nodes
docker service create | rollback | scale | ps | ls | rm | inspect | logs | update	
docker swarm ca | init | update | join-token | unlock-key | join | leave | unlock	

Docker Run Options

docker run --name mycontainer3 -it [IMGNAME] [CMD]	Names the container, -i keeps STDIN open, -t gives a pseudo-tty
docker run -a stdin -a stdout -it ubuntu /bin/bash	-a specifies terminal access; ubuntu is the image, cmd is bash
docker run -p 80:8080/tcp -it ubuntu /bin/bash	-p is port-map container's 8080 to host's 80 (can add host IP too)

Detached mode:
- Running -d for detached runs container in background. Allows closing the terminal session without stopping the container
- Containers exit when the root process starting it exits; using -d with --rm, it’s removed on exit or when the daemon exits.
- Don’t send a command like ‘service nginx start’ to a detached container. Use this syntax: nginx -g 'daemon off;'
- To do input/output with a detached container use network connections or shared volumes. These are required because
- The container stops listening to the terminal where ‘run’ was executed. Net connections or shared volumes are needed for I/O and this is why the -it option is needed (provides TTY)

Foreground mode:
- Default mode when -d isn’t specified. The streams stdout and stderr are attached if you dont use the -a option (no sdtin)
- Using the options "-it" is still common to provide TTY access, but you can also say -a stdin -a stdout -a stderr

Names are more user-friendly than UUID long and short identifiers assigned by the Docker daemon. When networking, containers on the default bridge network must be linked to communicate by name. There may also be some caveats with custom names, but there is a containerID/PID file option to remedy automation, etc., designated using --cidfile="___"
[Namespace designation options get out of scope of this document. See https://docs.docker.com/engine/reference/run/]

Container images can be accessed more specifically. Often a specific image version will be the tag added, such as in "ubuntu:22.04". Getting even more specific by supplying a hash (digest) value as in "nginx@sha256:9cacb71397...."

Docker Run Network Options

--network=" "	'bridge'	Network stack on default Docker bridge
'container'		<name | id>': reuse another container's network stack
'host'		Use the host network stack
'<network-name> | <network-id>'		Connect to a user-defined network
'none'		None
--network-alias=[ ]		Add network-scoped alias for the container
--add-host=" "		Add a line to /etc/hosts (host:IP)
--mac-address=" ", --ip=" ", --ip6=" "		Specifically as needed, set container's ethernet MAC, IPv4, or IPv6 addresses
--link-local-ip=[ ]		Set 1+ container's Eth link local IPv4/IPv6 addresses
--dns=[ ]		DNS servers

By default, networking is enabled on all containers unless disabled (with "none"); can make any outgoing connections- yet mapping ports as previously seen here and linking to other containers only works with the same default bridge. These are legacy Docker methods which have evolved. It's obvious from the command list above things are now more granular and natural to what we expect from actual hardware.

This is part I of a Docker overview currently being re-written from scratch. Part II will finish networking, cover security topics, build files, and wrap up the general topics