Managing Docker Containers

Quick Review of Container Image Creation and Usage

- 1: Build Image: `docker build -t my-app:0.1 .`
- 2: Push to Container Registry: `docker push my-app:0.1`
- 3: Pull from Container Registry: `docker pull my-app:0.1` [OPTIONAL]
- 4: Run a Container: `docker run my-app:0.1`
- NOTE: If image does not exist on the host, daemon would pull the image first and then run it
- NOTE: A set of images are cached on the host. Daemon would check the cache before pull the image again.

Explain important container management commands

Pulling an Image:

- Download an public image from a Container registry

```
docker pull <IMAGE_NAME>
```

Pulling an Image with specific version:

- Download a specific version of a public image from a Container registry

```
# Version or Tag
docker pull <IMAGE_NAME>:<SPECIFIC_VERSION>
```

On this page

Quick Review of Container Image Creation and Usage

Explain important container management commands

How do you set memory & CPU usage for containers & get Memory, and CPU usage info about specific containers?

Explain the Docker Container Lifecycle

How do you get all the system-level information about docker? Like events, disk space, caching, etc..

```
# Example:
docker pull node:24.01
```

Pulling an Private Image from Registry:

- To pull a private Docker image from a registry, you need to follow these steps:
  - **Docker Login**: Authenticate with the registry using the docker login command (Default Registry: Docker Hub)

```
# <registry-name>.azurecr.io
docker login <registry-url>
```

  - **Pull the Image**: Use the `docker pull` command with the full image name (including the registry URL)

```
docker pull <registry-url>/<image-name>:<tag>
```

  - **Example**:

```
docker login myreg.example.com
docker pull myreg.example.com/my-private-image:latest
```

Running an Image:

- Runs a container in detached mode (container runs in the background) and maps a host port to a container port

```
# HOST_PORT: The port number on your host machine
#             where you want to receive traffic
# CONTAINER_PORT: The port number within the container
#             that's listening for connections
docker run -d -p <HOST_PORT>:<CONTAINER_PORT> <IMAGE_NAME>
```

- Runs a container interactively (you can execute commands) with a terminal session

```
docker run -it <IMAGE_NAME> /bin/bash
```

Listing:

- List Currently Running Containers

```
docker container ls
```

```
CONTAINER ID   IMAGE          STATUS         NAMES
d1f4e5c8a123   nginx:latest   Up 5 minutes   sharp_haibt
f3c9e7a4b456   redis:alpine   Up 2 hours     eloquent_mo
abc1234def56   mysql:8.0      Up 24 hours    brave_banac
```

- List All Containers (Running + Stopped)

```
docker container ls -a
```

```
CONTAINER ID   IMAGE          STATUS                     NAMES
d1f4e5c8a123   nginx:latest   Up 5 minutes               sharp_haibt
f3c9e7a4b456   redis:alpine   Exited 10 minutes ago      eloquent_mo
abc1234def56   mysql:8.0      Up 24 hours                brave_banac
c4d7e2f9e789   alpine         Exited 2 hours ago         clever_poin
```

- Legacy Equivalent Commands (Still Commonly Used)

```
docker ps
docker ps -a
```

Start:

- Starts a stopped container

```
docker container start <CONTAINER_ID or NAME>
```

Stop:

- Stops a running container by sending a SIGTERM signal, followed by a SIGKILL signal after a grace period, terminating all processes in the container
  - A SIGTERM signal is a request for a process to terminate gracefully
  - While a SIGKILL signal is a forceful and immediate termination

```
docker container stop <CONTAINER_ID or NAME>
```

Kill:

- Immediately stops a container by sending a SIGKILL signal

```
docker container kill <CONTAINER_ID or NAME>
```

Restart:

- Restarts a running container

```
docker container restart <CONTAINER_ID or NAME>
```

Pause:

- Pauses all processes within the container, freezing them in their current state

```
docker container pause <CONTAINER_ID or NAME>
```

Unpause:

- Unpauses all processes, allowing the container to continue from where it was paused

```
docker container unpause <CONTAINER_ID or NAME>
```

Copy Files from Host to Running Container:

- Copies a file or directory from your host machine into a running container

```
docker cp /path/on/host <CONTAINER_ID_or_NAME>:/path/in/container

# Example:
docker cp ./app/config.json my_container:/usr/src/app/config.json
```

Remove:

- Removes one or more stopped containers

```
docker container rm <CONTAINER_ID or NAME>
```

- Removes all stopped containers

```
docker container prune
```

View logs

- Fetches the logs of a container

```
docker logs <CONTAINER_ID or NAME>
```

```
2045-06-30T12:00:00Z Starting Nginx server...
2045-06-30T12:00:01Z Listening on port 80
2045-06-30T12:02:15Z GET /index.html 200 OK
2045-06-30T12:03:44Z GET /favicon.ico 404 Not Found
2045-06-30T12:05:10Z Received SIGTERM, shutting down...
```

- Fetches the logs of a container and follows the log output

```
docker logs -f <CONTAINER_ID or NAME>
```

Managing Images:

- Lists all images on the local machine

```
docker images
```

```
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
nginx           latest    a1b2c3d4e5f6   2 days ago      142MB
redis           alpine    f6e5d4c3b2a1   1 week ago      32MB
mysql           8.0       1234abcd5678   3 weeks ago     495MB
```

- Displays the history of an image, showing each layer that was created during the build process

```
docker history <IMAGE_ID>
```

```
##simplified
IMAGE          CREATED        CREATED BY       SIZE      COMMENT
a1b2c3d4e5f6   2 days ago     CMD ["nginx"..    0B        -
7f8e9a0b1c2d   2 days ago     EXPOSE 80         0B        -
3c4d5e6f7g8h   2 days ago     RUN apt-get ..   53MB       -
1a2b3c4d5e6f   2 days ago     ADD files ..     22MB       -
```

Remove:

- Delete one or more Docker images from the local system

```
docker rmi <IMAGE_ID or NAME>
```

Cleanup:

- Removes unused data such as containers, images, networks, and build cache

```
docker system prune
```

How do you set memory & CPU usage for containers & get Memory, and CPU usage info about specific containers?

- **To limit the memory usage**:
  - use the `--memory` (or -m) flag
- **To limit the CPU usage**:
  - use the `--cpus` flag

```
docker run -d --memory="512m" --cpus="0.5" nginx:latest

# cpus - This restricts the container to use
#        at most 50% of a single CPU core.
# It doesn't mean the container gets a dedicated half-core
```

```
# It means the container will not exceed 0.5 CPU
# in terms of total CPU time across available cores.
```

- Get Usage for a Specific Container:

```
docker stats CONTAINER_ID_or_NAME
```

```
##simplified
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT
d1f4e5c8a123   sharp_haibt     0.45%     25.3MiB / 512MiB
```

Explain the Docker Container Lifecycle

Create:

- Define a container from an image without starting it

```
docker create --name mycontainer myimage
```

Start:

- Run a container that has been created or restart a stopped container

```
docker start mycontainer
```

Run:

- Create and start a container in one step

```
docker run --name mycontainer myimage
```

Stop:

- Gracefully halt the processes in a running container

```
docker stop mycontainer
```

Kill:

- Forcibly stop a running container immediately

```
docker kill mycontainer
```

Pause:

- Suspend all processes in a running container

```
docker pause mycontainer
```

Unpause:

- Resume all processes in a paused container

```
docker unpause mycontainer
```

Restart:

- Stop and then start a container again

```
docker restart mycontainer
```

Remove:

- Delete a stopped container from the Docker host

```
docker rm mycontainer
```

How do you get all the system-level information about docker? Like events, disk space, caching, etc..

- Display system-wide information about Docker installation, including number of containers, images, and storage driver details

```
docker info
```

```
Server Version:          34.0.2
Storage Driver:          overlay2
Number of Containers:    5 (3 running, 2 stopped)
Number of Images:        12
Default Runtime:         runc
```

- Show Docker disk usage, including total and used space for images, containers, and volumes

```
docker system df
```

```
TYPE            TOTAL     ACTIVE  SIZE    RECLAIMABLE
Images          12        4       2.4GB   1.5GB (62%)
Containers      5         2       450MB   300MB (67%)
Local Volumes   3         2       600MB   150MB (25%)
Build Cache     15        -       1.2GB   900MB (75%)
```

- Monitor Docker events in real-time, such as container starts/stops, image pulls, and network changes

```
docker events
```

```
2045-06-30T12:34:56Z container create
  d1f4e5c8a123 (image=nginx:latest, name=sharp_haibt)

2045-06-30T12:34:57Z container start
  d1f4e5c8a123 (image=nginx:latest, name=sharp_haibt)

2045-06-30T12:36:20Z image pull
  sha256:def456 (nginx:latest)

2045-06-30T12:38:42Z network connect
  container=abc123 network=bridge
```

- Display real-time CPU, memory, and network usage statistics for running containers

```
docker stats [container_id]
```

```
##simplified
CONTAINER ID   NAME            CPU %     MEM USAGE / LIMIT
d1f4e5c8a123   sharp_haibt     0.45%     25.3MiB / 512MiB
```

- Show the history of an image, including commands and metadata used to create each layer

```
docker history [image_name]
```

```
##simplified
IMAGE          CREATED        CREATED BY       SIZE
a1b2c3d4e5f6   2 days ago     CMD ["nginx"..    0B
7f8e9a0b1c2d   2 days ago     EXPOSE 80         0B
3c4d5e6f7g8h   2 days ago     RUN apt-get ..   53MB
1a2b3c4d5e6f   2 days ago     ADD files ..     22MB
```

- Return low-level information about Docker objects, such as containers, images, networks, and volumes

```
docker inspect [object_name]
```

```
[
  {
    "Id": "abc1234def5678gh9012ijkl3456mnop7890qrstuvwx",
    "Name": "/web_server",
    "Image": "nginx:latest",
    "State": {
      "Status": "running",
      "Running": true,
      "StartedAt": "2045-06-30T12:34:56.789012345Z"
    },
    "NetworkSettings": {
      "IPAddress": "172.17.0.2",
      "Ports": {
        "80/tcp": [
          {
            "HostIp": "0.0.0.0",
            "HostPort": "8080"
          }
        ]
      }
```

```
    },
    "Mounts": [
      {
        "Type": "volume",
        "Name": "myvolume",
        "Destination": "/usr/share/nginx/html"
      }
    ]
  }
]
```

- Remove unused data, such as stopped containers, dangling images, and unused networks and volumes

```
docker system prune
```

- Remove unused images

```
docker image prune
```

- Remove unused containers

```
docker container prune
```

- Remove unused networks

```
docker network prune
```

Previous
Building Docker Images

Next
Persisting Data in Docker

Home ↗

Springboot ↗

Cloud ↗

Roadmaps ↗

Flashcards ↗

Bookshelf ↗