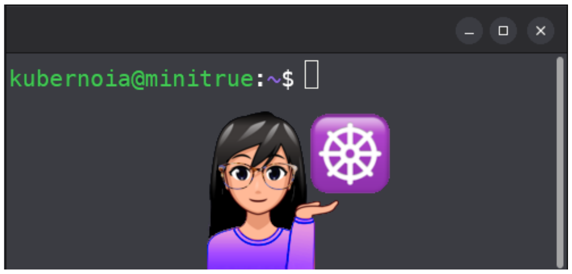
DOCKER CHEATSHEET

BY PAULOBA



It's usual to deploy applications to k8s in the form of application containers created from Dockerfiles.

Docker commands

- Build a Docker image from a Dockerfile and a context
  - `docker build .`
- Run a Docker container derived from a Docker image
  - `docker run -d -p 80:80 my_image service nginx start`
- Run a Docker container in detached mode with -d
  - `docker run -d -p 80:80 my_image service nginx start`
- Stop 2 containers
  - `docker stop confident_agnesi great_lamarr`
- Build an image and tag it. The repository name will be `vieux/apache` and the tag will be `2.0`
  - `docker build -t vieux/apache:2.0 .`
- Build a Docker image from a given Dockerfile
  - `docker build -f Dockerfile.debug .`

- Execute a command within an existing container
  - `docker exec -it beautiful_curie sh`

- Run a container with a volume binded to a volume mounted within the container. This is useful when developing and testing applications locally without the need to restart the container to get data persistence (example here)
  - `docker run -d -v /home/pau/docker-test/app/etc:/etc/data-base -p 3000:3000 -v /home/pau/docker-test/app/src:/app/src getting-started`

- Delete all Docker images
  - `docker rmi -f $(docker images -q)`

- List full container IDs (not truncated)
  - `docker container ls --quiet --no-trunc`

- Force delete containers
  - `docker container rm -f $(docker container ls -aq)`

- Check container size
  - `docker container ls --latest -s`

- Show all running containers
  - `docker ps`

- Show all containers that were running recently
  - `docker ps -a`

- Run a Docker image detached from the user (runs the container on the background of the terminal)
  - `sudo docker run -d gcr.io/my-gcp-project/my-custom-jenkins-slave-image:1`

- Delete images without a tag
  - `docker image prune`

- The `docker logs` command will show you the output a container is generating when you don't run it interactively. This is likely to include the error message.
  - `docker logs --tail 50 --follow --timestamps mediawiki_web_1`

- Push an image built locally to a GCR
    - `gcloud auth print-access-token | docker login -u oauth2accesstoken --password-stdin https://gcr.io`
    - `docker build -t my-image-name .`
        - You can use the `--network=host` docker flag
          `$ docker build --network=host -t my-image-name .`
        - If you see CDN errors while building images in Ubuntu, similar to these ones:

```
 ---> Running in 44279151beee
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/main/x86_64/APKINDEX.tar.gz
fetch
https://dl-cdn.alpinelinux.org/alpine/v3.17/community/x86_64/APKINDEX.tar.gz
WARNING: Ignoring https://dl-cdn.alpinelinux.org/alpine/v3.17/main: temporary
error (try again later)
WARNING: Ignoring https://dl-cdn.alpinelinux.org/alpine/v3.17/community:
temporary error (try again later)
ERROR: unable to select packages:
  bash (no such package):
      required by: world[bash]
```

    - `docker image ls`
    - `docker tag 50e90e3b3cc6 gcr.io/my-gcp-project/my-custom-jenkins-image:dev`
    - `docker push gcr.io/my-gcp-project/my-custom-jenkins-image:dev`

- Creating a Dockerfile can be done using a **heredoc**.
  A heredoc, short for here document, is a way to define a multi-line string literal in programming languages. It allows you to create a block of text that spans multiple lines without having to use escape characters to concatenate multiple strings.
  To create a Dockerfile in the current directory run the following cat command to use a heredoc:

```
cat << EOF > Dockerfile
FROM nginx:latest
COPY . /usr/share/nginx/html

EXPOSE 80
EOF
```

Troubleshooting Docker issues

- Docker: “build” Requires 1 Argument Error → https://www.baeldung.com/ops/docker-build-argument-error
- Docker logs https://www.baeldung.com/ops/docker-logs#using-default-log-file