Working with Docker

- Cheatsheet -

MVP: Run an interactive container

```
docker run -it -rm -d \
  -p <port_ext>:<port_int> \
  --name my_app \
  <app_name>
```

MVP: Build an image

```
docker build \
    -t <img_name>:<tag> \
    -f </path/to/dockerfile/> \
    <app_name>
```

Remove sudo constraint for docker commands

```
$ sudo groupadd docker
$ sudo gpasswd -a $USER docker
```

Build an app image locally

```
$ docker build -t <name:tag> .
$ docker build -f <path/to/.dockerfile>
```

Start up a container | Print a list of containers

```
$ docker run --name <name> <image-name>
$ docker ps -a
```

Start a terminal session in a container

```
$ docker attach <container-name>
$ docker start -a -i <container-name>
```

Stop and delete a container

```
$ docker stop <container-name>
$ docker rm <container-name>
```

Get rid of old images and containers

```
$ docker image prune -a
$ docker system prune
```

.dockerfile

Blueprint for a utility container, e.g. for a vue cli runtime

```
FROM node:14
WORKDIR /vue-setup

# Replace below command with any CLI to be available in the container
RUN npm install -g @vue/cli

# The following commands ensure access to our files
# If we left them out, changing files on our local setup
# would fail due to insufficient permissions.
RUN userdel -r node

ARG USER_ID
ARG GROUP_ID

RUN addgroup --gid $GROUP_ID user
RUN adduser --disabled-password --gecos '' --uid $USER_ID --gid $GROUP_ID user

# Set the active user and open the interactive terminal
USER user

ENTRYPOINT [ "bash" ]
```

Blueprint for a 2-stage building workflow

```
# build stage
FROM node:alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# production stage
FROM nginx:apline as production-stage
COPY --from=build-stage /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```