# dockerfiles

What is it?

- YML file, scriptable replacement for running manual image creation commands
- use with “docker build” to automate creation of images
- NOTE: each command creates a new intermediate image, so generally commands are concatenated with \

```
FROM ubuntu
MAINTAINER Bob the Builder

RUN apt-get install -y python-software-properties python python-setuptools ruby rubygems
RUN echo "deb http://us.archive.ubuntu.com/ubuntu/ precise universe" >> /etc/apt/sources.list
ENV APACHE_RUN_USER www-data

WORKDIR /var/www/files
COPY /tmp/hosts /etc/hosts
RUN printf "[include]\nfiles = /var/www/Supervisorfile\n" >> /etc/supervisord.conf
VOLUME /var/www;/var/www/files

ADD . /var/www
ONBUILD /var/scripts/createApp /var/www/app.js

CMD ["/usr/local/bin/supervisord", "-n", "-c", "/etc/supervisord.conf"]
```

```
docker build -t <user>/<image>:<tag> <image>
```

# “abstract” dockerfiles

Popular approaches:

- Enforce that the child image provides a set of files (imported via volume options), which are then automatically picked up by the parent when running
  - eg. A node.JS parent image which launches an app.js file in a known folder (also containing node_modules)

- Use ONBUILD hooks in non-instantiable parent images, and then have child images provide the implementations called by these hooks.
  - eg. Configuration for a market-specific implementation of a Global Platform

In both of these approaches, the “shape” and control of the image is still provided by the parent image.s