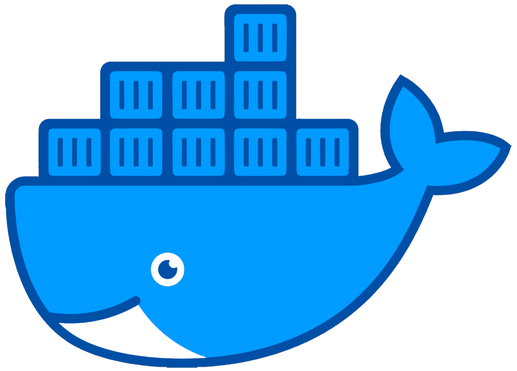
Docker for scientific research



Paxton Fitzpatrick

December 1, 2021

What is Docker?

- Self-contained, isolated software environments called *“containers”*
- Create, share, and deploy applications & services
- Popular tool in production environments

But how is this useful for research?

Reproducible code!

- Experiments
- Data analyses

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
- Data analyses

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
  - Run multiple experiments on the same computer
- Data analyses

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
  - Run multiple experiments on the same computer
  - Launch complex experiments with a single command
- Data analyses

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
  - Run multiple experiments on the same computer
  - Launch complex experiments with a single command
- Data analyses
  - Ensure identical packages & versions across project team

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
  - Run multiple experiments on the same computer
  - Launch complex experiments with a single command
- Data analyses
  - Ensure identical packages & versions across project team
  - Isolate environments for each project from each other

But how is this useful for research?

Reproducible code!

- Experiments
  - Run the same on any computer
  - Run multiple experiments on the same computer
  - Launch complex experiments with a single command
- Data analyses
  - Ensure identical packages & versions across project team
  - Isolate environments for each project from each other
  - Run analyses on Discovery in the same environment you use locally

Quick Docker jargon…

Image

"Template" environment with everything needed to perform a certain task

Container

A running instance of an image

Dockerfile

File containing instructions to build an image

docker-compose

Tool for defining and running multiple, coordinated containers

Docker Hub

GitHub-like website with repositories of pre-built images

Data analyses in Docker

Goal: create an environment that:

- has Python installed
- contains all packages needed for analyses
- can run Jupyter notebooks
- is isolated from the host machine, but has access to it
- is easy to create, use, tweak, and share
- will always be exactly the same, no matter when or where it’s used

Data analyses in Docker

Steps to create and use our environment:

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"

ENTRYPOINT ["tini", "-g", "--"]

CMD ["jupyter", "notebook"]
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

FROM

- Specifies the base image to build on top of
- Can be an image you built or any image on Docker Hub
- Format:

```
FROM <user>/<repository>:<tag>
```

- Debian 10 (“Buster”) image with Miniconda 4.9.2 installed

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

FROM

- Spec
- Can
  Doc
- Forr
- Deb
  insta

```
FROM debian:buster-slim

LABEL maintainer="Anaconda, Inc"

ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

RUN apt-get update -q && \
    apt-get install -q -y --no-install-recommends \
        bzip2 \
        ca-certificates \
        git \
        libglib2.0-0 \
        libsm6 \
        libxext6 \
        libxrender1 \
        mercurial \
        openssh-client \
        subversion \
        wget \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

ENV PATH /opt/conda/bin:$PATH

CMD [ "/bin/bash" ]

ARG CONDA_VERSION=py38_4.9.2

RUN set -x && \
    UNAME_M="$(uname -m)" && \
    if [ "${UNAME_M}" = "x86_64" ]; then \
        MINICONDA_URL="https://repo.anaconda.com/miniconda/Miniconda3-${CONDA_VERSION}-Linux-x86_64.sh"; \
        SHA256SUM="1314b90489f154602fd794accfc90446111514a5a72fe1f71ab83e07de9504a7"; \
    elif [ "${UNAME_M}" = "s390x" ]; then \
        MINICONDA_URL="https://repo.anaconda.com/miniconda/Miniconda3-${CONDA_VERSION}-Linux-s390x.sh"; \
        SHA256SUM="4e6ace66b732170689fd2a7d86559f674f2de0a0a0fbaefd86ef597d52b89d16"; \
    elif [ "${UNAME_M}" = "aarch64" ]; then \
        MINICONDA_URL="https://repo.anaconda.com/miniconda/Miniconda3-${CONDA_VERSION}-Linux-aarch64.sh"; \
        SHA256SUM="b6fbba97d7cef35ebee8739536752cd8b8b414f88e237146b11ebf081c44618f"; \
    elif [ "${UNAME_M}" = "ppc64le" ]; then \
        MINICONDA_URL="https://repo.anaconda.com/miniconda/Miniconda3-${CONDA_VERSION}-Linux-ppc64le.sh"; \
        SHA256SUM="2b111dab4b72a34c969188aa7a91eca927a034b14a87f725fa8d295955364e71"; \
    fi && \
    wget "${MINICONDA_URL}" -O miniconda.sh -q && \
    echo "${SHA256SUM} miniconda.sh" > shasum && \
    if [ "${CONDA_VERSION}" != "latest" ]; then sha256sum --check --status shasum; fi && \
    mkdir -p /opt && \
    sh miniconda.sh -b -p /opt/conda && \
    rm miniconda.sh shasum && \
    ln -s /opt/conda/etc/profile.d/conda.sh /etc/profile.d/conda.sh && \
    echo ". /opt/conda/etc/profile.d/conda.sh" >> ~/.bashrc && \
    echo "conda activate base" >> ~/.bashrc && \
    find /opt/conda/ -follow -type f -name '*.a' -delete && \
    find /opt/conda/ -follow -type f -name '*.js.map' -delete && \
    /opt/conda/bin/conda clean -afy
```

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

ARG

- Defines an argument that can be passed via the command line when building the image
- The port the Jupyter notebook server will listen on
- Variable persists for the remainder of the build

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

ARG

- Defines an argument that can be passed via the command line when building the image
- The port the Jupyter notebook server will listen on
- Variable persists for the remainder of the build

ENV

- Sets an environment variable for the remainder of the build *and* in containers run from the resulting image

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

RUN

- Executes commands in a new "*layer*" on top of the current image and "*commits*" the result
- **Docker images work a lot like git repositories**
  - Each instruction modifies the image by adding a new layer on top of it
  - Docker stores images as a series of "diffs" between layers
- RUN creates a container from the current layer

```
ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

RUN

- Configure conda
  - Disable auto-updates and update notification
  - Use `conda-forge` channel to install packages
- Install requirements
  - Python (check expected version)
  - `pip` package manager
  - Jupyter notebooks, interactive widgets, some handy notebook extensions
  - `tini` init manager
  - Packages for analyses
- Clear caches to reduce image size

```
ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

RUN

- Why all the **&&**'s?

```
ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

RUN

- Why all the **&&**'s?
- Each RUN instruction creates an intermediate layer

2.19GB

```
RUN conda config --set auto_update_conda false
RUN conda config --set notify_outdated_conda false
RUN conda config --prepend channels conda-forge
RUN conda config --set channel_priority strict
RUN conda install -Sy \
        python==3.8.5 \
        pip==20.2.4
RUN conda install -Sy \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0
RUN conda install -Sy \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0
RUN conda clean -afy
```

```
ARG port=8888

ENV NOTEBOOK_PORT $port
```

918MB

```
RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy
```

```
COPY jupyter_notebook_config.py /root/.jupyter/
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

RUN

- Why all the **&&**'s?
- Each RUN instruction creates an intermediate layer

```
ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

COPY

- Copies files or directories from the host into the image
- Config for Jupyter notebook server

```
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"

ENTRYPOINT ["tini", "-g", "--"]
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

WORKDIR

- Sets the working directory for subsequent instructions and containers run from the image
- `/mnt` will be the “mount point” for our container

```
    && conda install -sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"

ENTRYPOINT ["tini", "-g", "--"]

CMD ["jupyter", "notebook"]
```

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

Dockerfile Instructions

ENTRYPOINT

- Executable *always* run when a container starts. *Cannot* be overridden from the command line.

CMD

- *Default* arguments passed to ENTRYPOINT (or default command to run, if no ENTRYPOINT). *Can* be overridden from the command line.

```
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"

ENTRYPOINT ["tini", "-g", "--"]

CMD ["jupyter", "notebook"]
```

Data analyses in Docker

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

```
FROM continuumio/miniconda3:4.9.2

ARG port=8888

ENV NOTEBOOK_PORT $port

RUN conda config --set auto_update_conda false \
    && conda config --set notify_outdated_conda false \
    && conda config --prepend channels conda-forge \
    && conda config --set channel_priority strict \
    && conda install -Sy \
        python==3.8.5 \
        pip==20.2.4 \
        notebook=6.1.4 \
        ipywidgets=7.5.1 \
        jupyter_contrib_nbextensions=0.5.1 \
        tini=0.18.0 \
        numpy=1.19.1 \
        pandas=1.1.2 \
        matplotlib=3.2.2 \
        seaborn=0.11.0 \
    && conda clean -afy

COPY jupyter_notebook_config.py /root/.jupyter/

WORKDIR "/mnt"

ENTRYPOINT ["tini", "-g", "--"]

CMD ["jupyter", "notebook"]
```

Data analyses in Docker

The `docker build` command

- Builds an image from a Dockerfile and a "*build context*"
- "Build context": files that can be used during the build process
- Can be a local directory path, GitHub repo URL, or tarball (`.tar.gz` file)
- Default: pass context to `docker build`; Docker looks for "Dockerfile" in its root directory

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

The `docker build` command

- To build the image from our Dockerfile:

```
$ docker build -t tutorial-image .
```

Name and optionally tag the image (default tag: "`latest`")

Build context: current directory

(If you're following along, don't run this)

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

The `docker build` command

- To build the image from our Dockerfile:

Set the `$port` variable from the ARG instruction in the Dockerfile

```
$ docker build -t tutorial-image --build-arg port=8889 .
```

Name and optionally tag the image (default tag: "`latest`")

Build context: current directory

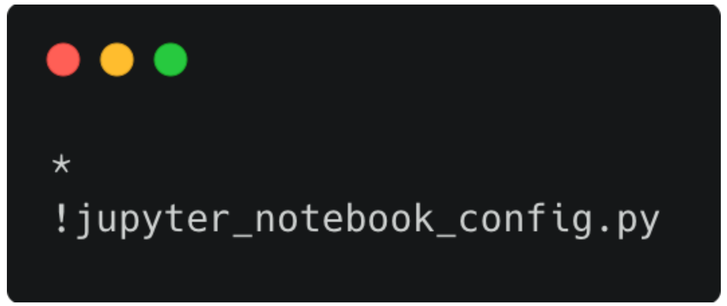
(If you're following along, don't run this)

1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

The `docker build` command

- Large build context can lead to *very* slow build
- You can exclude files and directories from context with a `.dockerignore` file

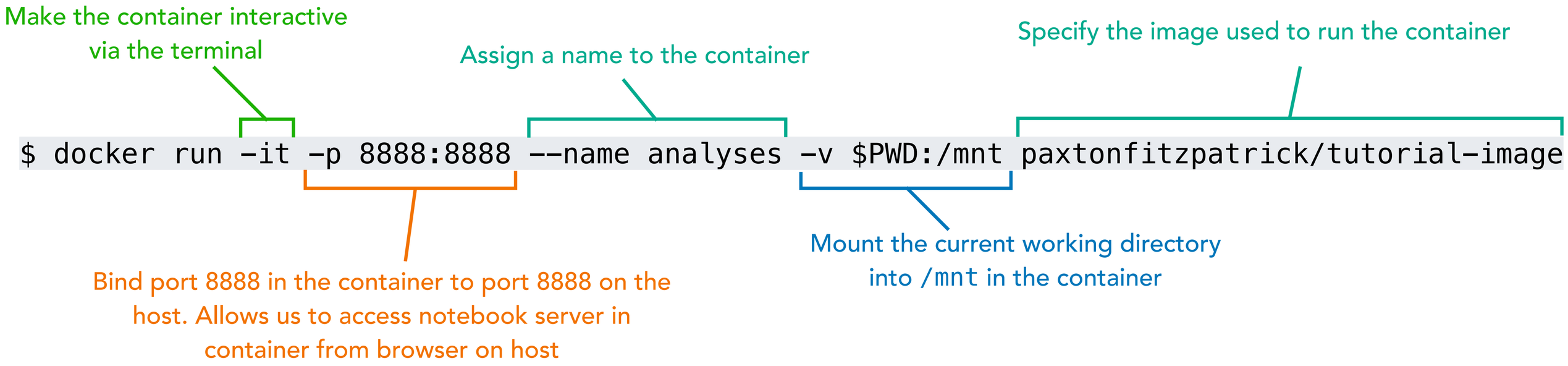


1. Write a *Dockerfile*
2. Build an *image*
3. Run a *container*

Data analyses in Docker

The `docker run` command

- `docker run` creates and runs a container from an image
- Can be an image you've built locally or any image on Docker Hub
- To run a container from the pre-built version of our image on Docker Hub:



Data analyses in Docker

Demo… 🤞

Data analyses in Docker

Using our container on Discovery

Demo… 🤞

Behavioral experiments in Docker

Running psiTurk through Docker

```
FROM python:3.6-stretch

ENV PSITURK_GLOBAL_CONFIG_LOCATION "/exp"

WORKDIR "/exp"

RUN pip install \
        psiturk==2.3.8 \
        pymysql==0.10.0 \
        python-Levenshtein==0.12.0 \
    && rm -rf ~/.cache/pip

CMD ["bash"]
```

Behavioral experiments in Docker

Running psiTurk through Docker

- Specify the base image to build from
- Set environment variable (tells psiTurk where to look for `.psiturkconfig`)
- Set the working director
- Install psiTurk and some extra required packages
- Clear cache to keep image small
- Set the command executed when running a container (launch a bash shell)

Behavioral experiments in Docker

Running psiTurk through Dock

- We need 3 support services to run on MTurk
- Normally complex to configure, tedious to manage
- But Docker makes this *much* easier!
  - All 3 available on Docker Hub as ready-to-run images
  - `docker-compose`: build, start and stop entire setup with a single command
- Configure full application in `docker-compose.yml`

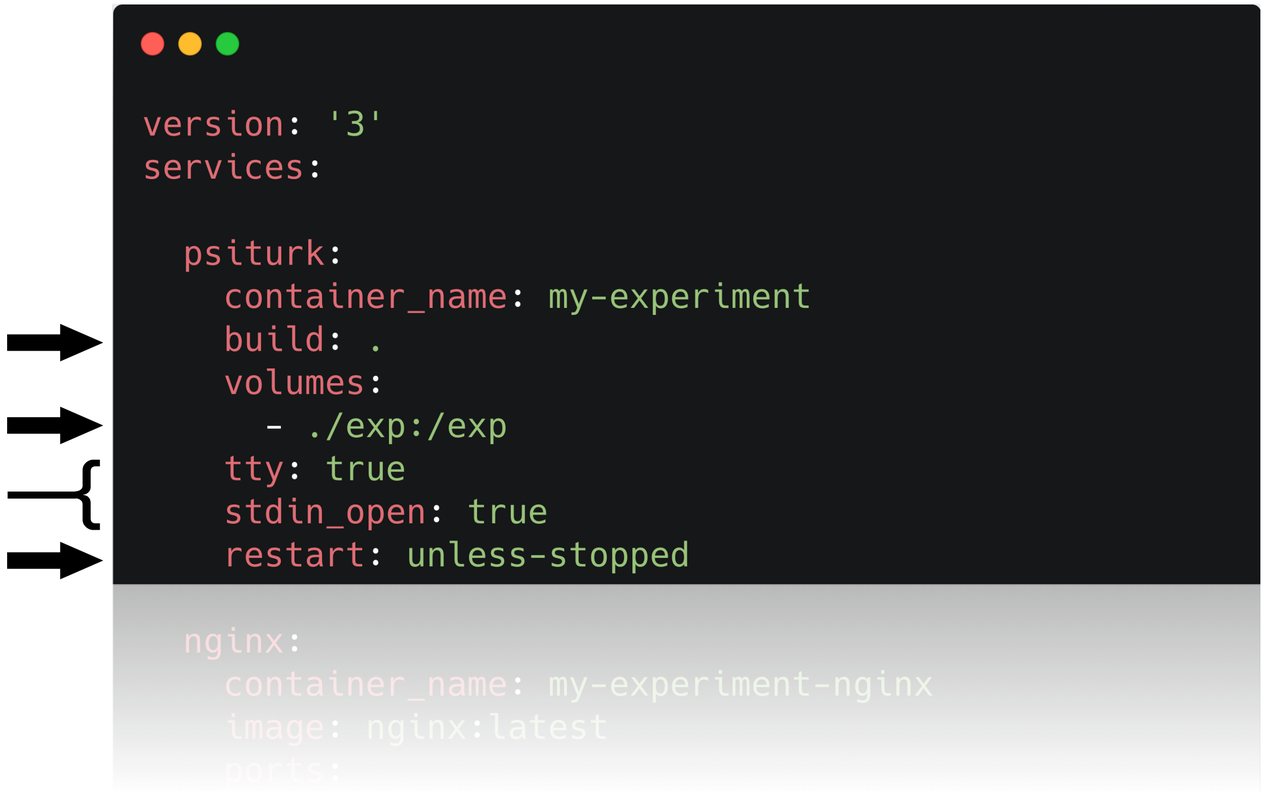
```yaml
version: '3'
services:

  psiturk:
    container_name: my-experiment
    build: .
    volumes:
      - ./exp:/exp
    tty: true
    stdin_open: true
    restart: unless-stopped

  nginx:
    container_name: my-experiment-nginx
    image: nginx:latest
    ports:
      - 80:80
    volumes:
      - ./exp:/var/www/exp:ro
      - ./default.conf:/etc/nginx/conf.d/default.conf
    restart: unless-stopped

  db:
    container_name: my-experiment-db
    image: mysql:5.7
    volumes:
      - ./data/db:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: participants
      MYSQL_USER: paxton
      MYSQL_PASSWORD: psiturk
    restart: unless-stopped

  adminer:
    container_name: my-experiment-adminer
    image: adminer:latest
    ports:
      - 127.0.0.1:8080:8080
```

Behavioral experiments in Docker

`docker-compose.yml`

Container 1: **psiTurk server**

- Build context
- Mount the host directory `exp` to `/exp` in the container
- Allocate a pseudo-TTY and send stdin to the container (so we can run an interactive shell)
- If the container fails for some reason, restart it unless we explicitly stopped it.



```yaml
version: '3'
services:

  psiturk:
    container_name: my-experiment
    build: .
    volumes:
      - ./exp:/exp
    tty: true
    stdin_open: true
    restart: unless-stopped

  nginx:
    container_name: my-experiment-nginx
    image: nginx:latest
    ports:
```

Behavioral experiments in Docker

`docker-compose.yml`

Container 2: **NGINX**

(reverse proxy server for load balancing)

- Pull the `nginx:latest` image from Docker Hub
- Map host port 80 to container port 80
- Create two mount points:
  - host exp directory ➡ NGINX web root
  - NGINX config file ➡ expected location

```
    volumes:
      - ./exp:/exp
    tty: true
    stdin_open: true
    restart: unless-stopped

  nginx:
    container_name: my-experiment-nginx
    image: nginx:latest
    ports:
      - 80:80
    volumes:
      - ./exp:/var/www/exp:ro
      - ./default.conf:/etc/nginx/conf.d/default.conf
    restart: unless-stopped

  db:
    container_name: my-experiment-db
    image: mysql:5.7
    volumes:
      - ./data/db:/var/lib/mysql
    environment:
```

Behavioral experiments in Docker

`docker-compose.yml`

Container 3: **MySQL database**
(greater concurrency than default SQLite DB)

- Pull the `mysql:5.7` image from Docker Hub
- Mount `data/db` directory on host to `/var/lib/mysql` in container
- Set container environment variables:
  - Password for MySQL `root` user (required)
  - Name of DB created in the container
  - Username/password for psiTurk to write to DB

Behavioral experiments in Docker

`docker-compose.yml`

Container 4: **Adminer**
(PHP app for viewing/downloading data)

- Pull the `adminer:latest` image from Docker Hub

- Map host port 8080 to container port 8080 (allows us to access Adminer via a web browser)

```
    volumes:
      - ./data/db:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: participants
      MYSQL_USER: paxton
      MYSQL_PASSWORD: psiturk
    restart: unless-stopped

  adminer:
    container_name: my-experiment-adminer
    image: adminer:latest
    ports:
      - 127.0.0.1:8080:8080
```

Behavioral experiments in Docker

```yaml
version: '3'
services:

  psiturk:
    container_name: my-experiment
    build: .
    volumes:
      - ./exp:/exp
    tty: true
    stdin_open: true
    restart: unless-stopped

  nginx:
    container_name: my-experiment-nginx
    image: nginx:latest
    ports:
      - 80:80
    volumes:
      - ./exp:/var/www/exp:ro
      - ./default.conf:/etc/nginx/conf.d/default.conf
    restart: unless-stopped

  db:
    container_name: my-experiment-db
    image: mysql:5.7
    volumes:
      - ./data/db:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: mypassword
      MYSQL_DATABASE: participants
      MYSQL_USER: paxton
      MYSQL_PASSWORD: psiturk
    restart: unless-stopped

  adminer:
    container_name: my-experiment-adminer
    image: adminer:latest
    ports:
      - 127.0.0.1:8080:8080
```

Behavioral experiments in Docker

Demo… 🤞