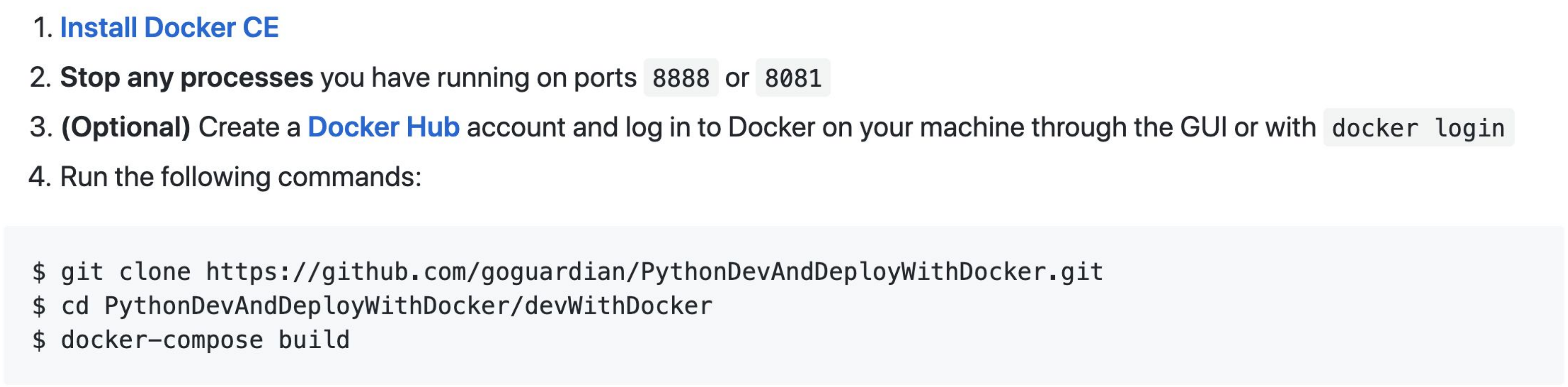
Welcome to GoGuardian

PLEASE FOLLOW THESE INSTRUCTIONS BEFORE WE START:

Instructions at: github.com/goguardian/PythonDevAndDeployWithDocker



1. Install Docker CE
2. **Stop any processes** you have running on ports `8888` or `8081`
3. **(Optional)** Create a Docker Hub account and log in to Docker on your machine through the GUI or with `docker login`
4. Run the following commands:

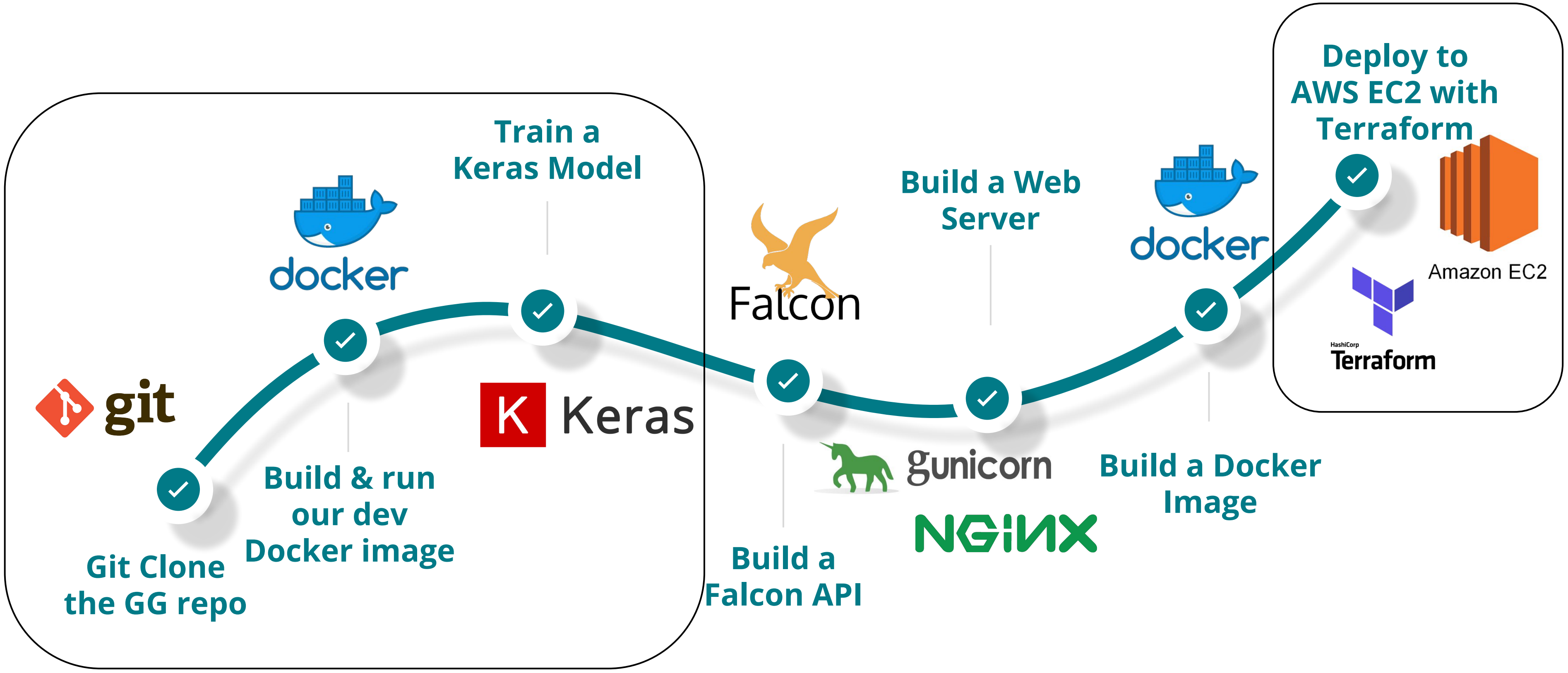
```
$ git clone https://github.com/goguardian/PythonDevAndDeployWithDocker.git
$ cd PythonDevAndDeployWithDocker/devWithDocker
$ docker-compose build
```

Wifi: GG_Guest **Password: GG_Guest_Welcome!**

Python Development with Docker

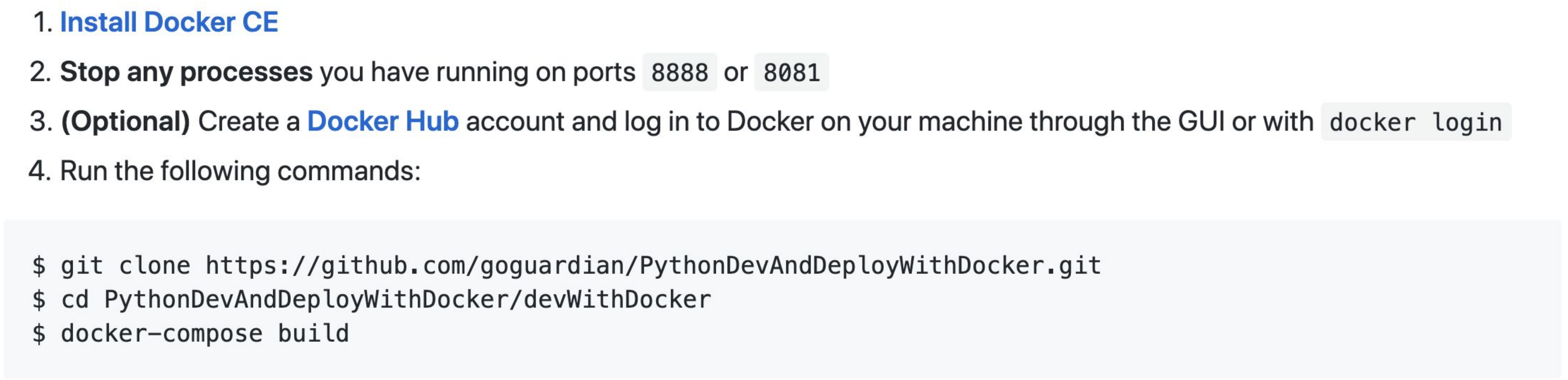
Michael G. Frantz, PhD (GoGuardian)

What will we do today?



Before we continue...

Instructions at: github.com/goguardian/PythonDevAndDeployWithDocker



1. Install Docker CE
2. **Stop any processes** you have running on ports `8888` or `8081`
3. **(Optional)** Create a Docker Hub account and log in to Docker on your machine through the GUI or with `docker login`
4. Run the following commands:

```
$ git clone https://github.com/goguardian/PythonDevAndDeployWithDocker.git
$ cd PythonDevAndDeployWithDocker/devWithDocker
$ docker-compose build
```

Different Stacks Require Different Environments

1 **Team Member 1**

Scikit-learn, NumPy, Pandas, Scipy…

2 **Team Member 2**

Django, Flask, Falcon…

3 **Team Member 3**

OpenCV, Pillow, Tensorflow…

Classical Solutions

Virtual Environments

python™
virtualenv

Virtual Machines

vmware®

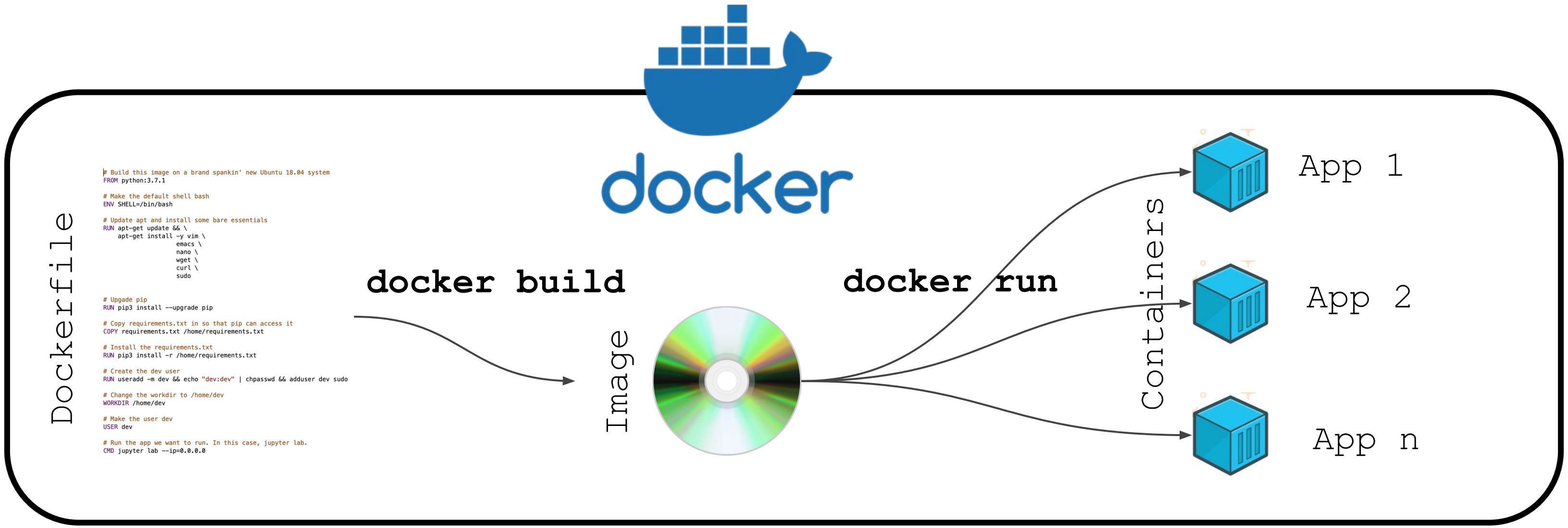
Amazon Machine
Image (AMI)

ORACLE
VirtualBox

VAGRANT

What is docker?

- Docker is software that can build and run containers
- Containers are one instance of a docker image
- A docker image is a unit of software that allows one to package up code and all its dependencies



```dockerfile
# Build this image on a brand spankin' new Ubuntu 18.04 system
FROM python:3.7.1

# Make the default shell bash
ENV SHELL=/bin/bash

# Update apt and install some bare essentials
RUN apt-get update && \
    apt-get install -y vim \
                       emacs \
                       nano \
                       wget \
                       curl \
                       sudo


# Upgade pip
RUN pip3 install --upgrade pip

# Copy requirements.txt in so that pip can access it
COPY requirements.txt /home/requirements.txt

# Install the requirements.txt
RUN pip3 install -r /home/requirements.txt

# Create the dev user
RUN useradd -m dev && echo "dev:dev" | chpasswd && adduser dev sudo

# Change the workdir to /home/dev
WORKDIR /home/dev

# Make the user dev
USER dev

# Run the app we want to run. In this case, jupyter lab.
CMD jupyter lab --ip=0.0.0.0
```

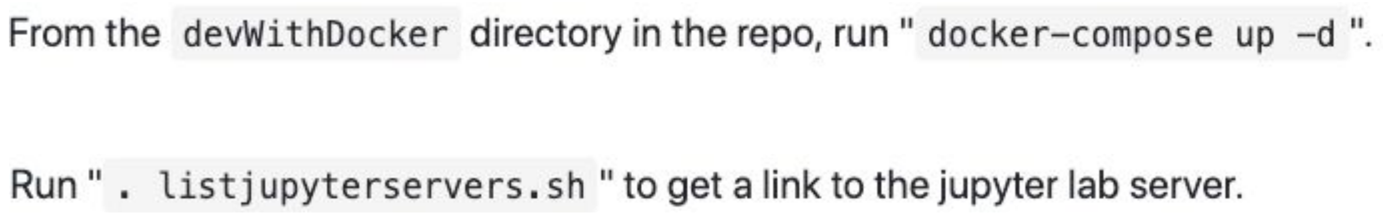
Why develop with Docker?

- Docker is **platform-agnostic**. If your machine has Docker, you can develop in the same environment.
- Docker Hub can be used to **share version-controlled environments** among team members. Think GitHub for docker images!
- Docker containers are **ephemeral**, so if you mess it up you can start over easily.
- You can **build on top** of what other people or organizations have built.
- When development is done, it’s **easy to deploy** with services like ECS, Kubernetes.

What will we do today?

- **Build**
  - The `Dockerfile` defines the environment and app in code
  - Our Dockerfile builds upon `Ubuntu 18.04`, installs some things with `apt` and `pip`, and last defines startup behavior to run `jupyter lab`.
- **Run**
  - The `docker-compose.yml` configures how you want to build and run your container. This can be thought of as a `docker run` command in a `.yml` file.
- **Train**
  - MNIST is the “hello world” for computer vision

Getting Started...



From the `devWithDocker` directory in the repo, run "`docker-compose up -d`".

Run "`. listjupyterservers.sh`" to get a link to the jupyter lab server.

We’re using docker compose instead of the following commands:

```
docker build -t dockerjupyter <path_to_context>

docker run -d --rm -p 8888:8888 -v ..:/home/dev/ \
    --name jupyter dockerjupyter
```

Let’s dive in!