Docker: For Noobs, By Noobs

Stay Late And Code

Wen Chang, Anna Gapuz

Agenda

* Verify Installation
* Brief Overview
* Labs, labs, labs!

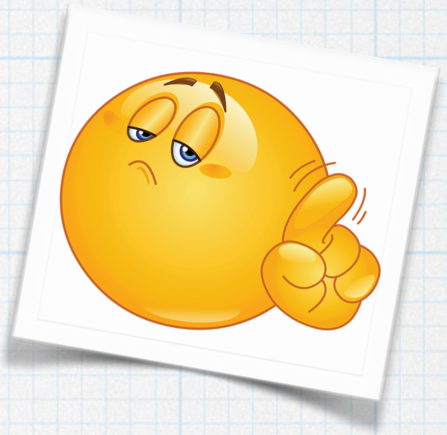
Fannie Mae Laptop?

Sorry folks, Docker is not available for installation through MyServices

You must use a personal laptop

The Installs

For the folks who didn't follow the directions on Confluence



- Sign up for a free Docker ID
  - https://hub.docker.com/signup
- Install Docker Desktop or Docker Engine
  - Mac: https://hub.docker.com/editions/community/docker-ce-desktop-mac
  - Windows: https://hub.docker.com/editions/community/docker-ce-desktop-windows
  - Linux: https://hub.docker.com/search/?type=edition&offering=community&operating_system=linux&platform=server

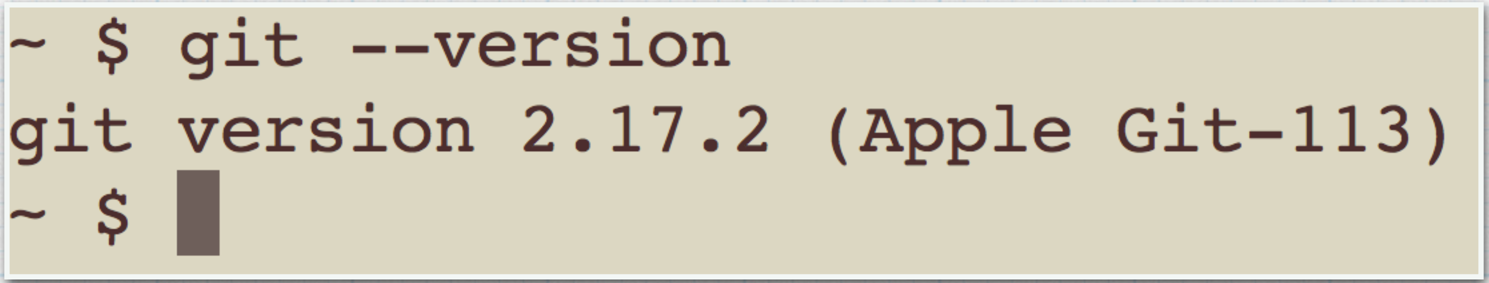
The Installs

For the folks who didn't follow the directions on Confluence

* Install Git
    * https://git-scm.com/downloads
* Sign up for a free Github account
    * https://www.github.com/join
    * Set up SSH access: https://help.github.com/articles/connecting-to-github-with-ssh
* Have a lightweight text editor handy
    * Notepad, TextEdit, Atom, Notepad++, TextWrangler, VS Code, SublimeText (if you're fancy)
* Have a terminal handy
    * Terminal, PowerShell, ITerm2, Cygwin, ConEmu, Cmder

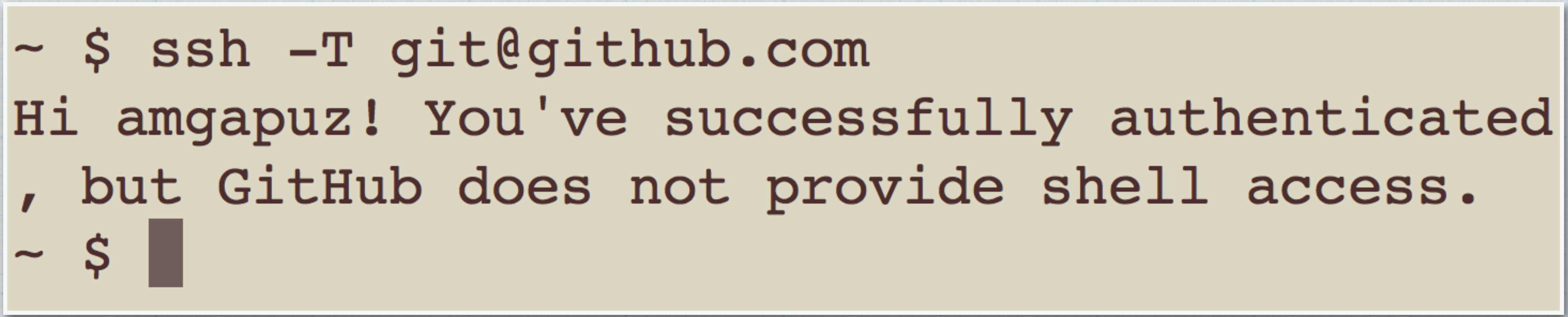
Verify Git and Github

* Open a terminal
* `git --version`



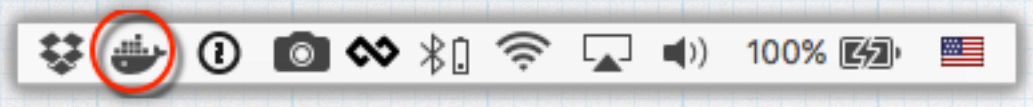
```
~ $ git --version
git version 2.17.2 (Apple Git-113)
~ $ █
```

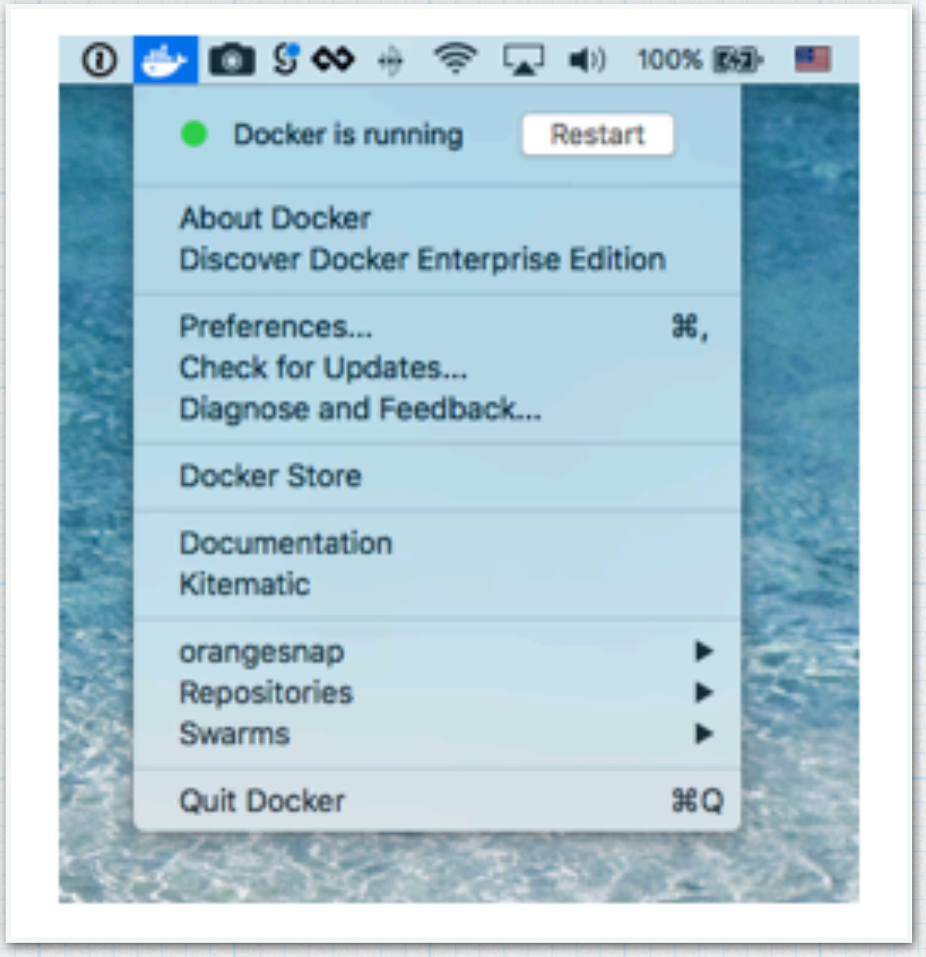
* `ssh -T git@github.com`

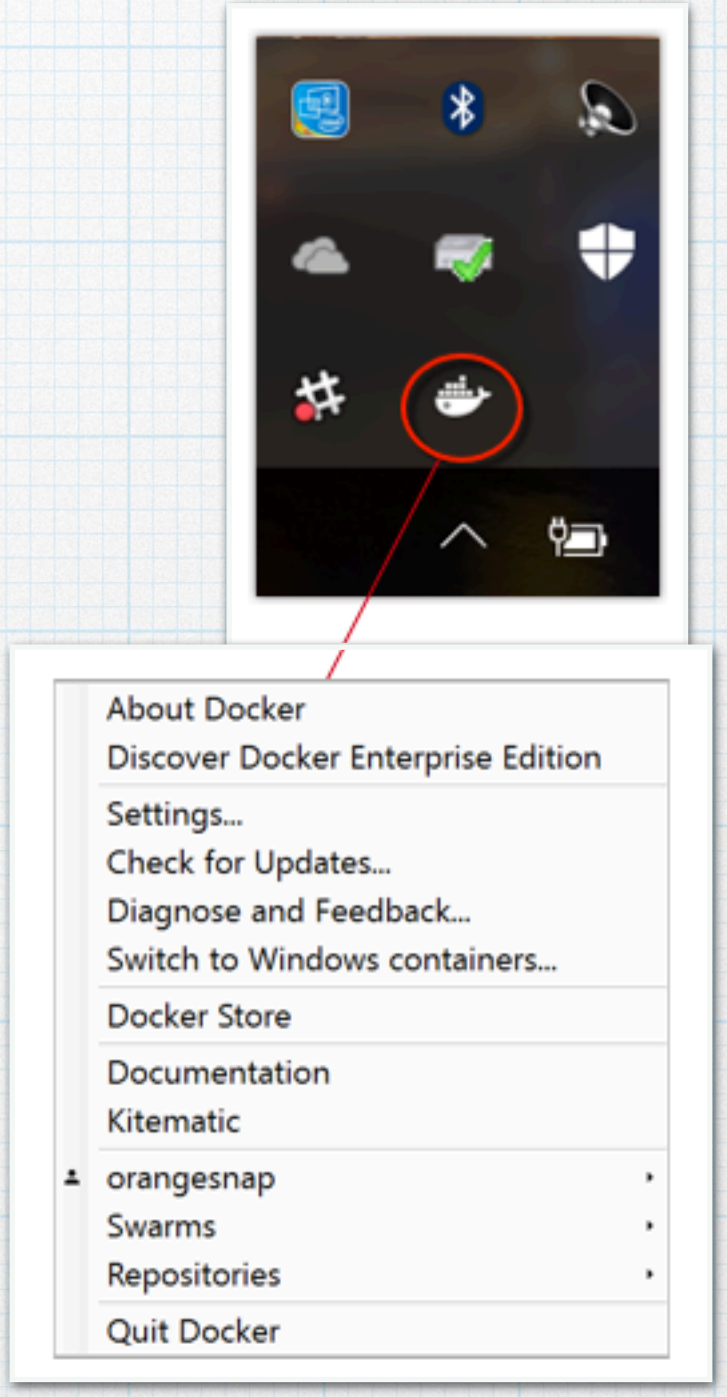


```
~ $ ssh -T git@github.com
Hi amgapuz! You've successfully authenticated
, but GitHub does not provide shell access.
~ $ █
```

Start Docker

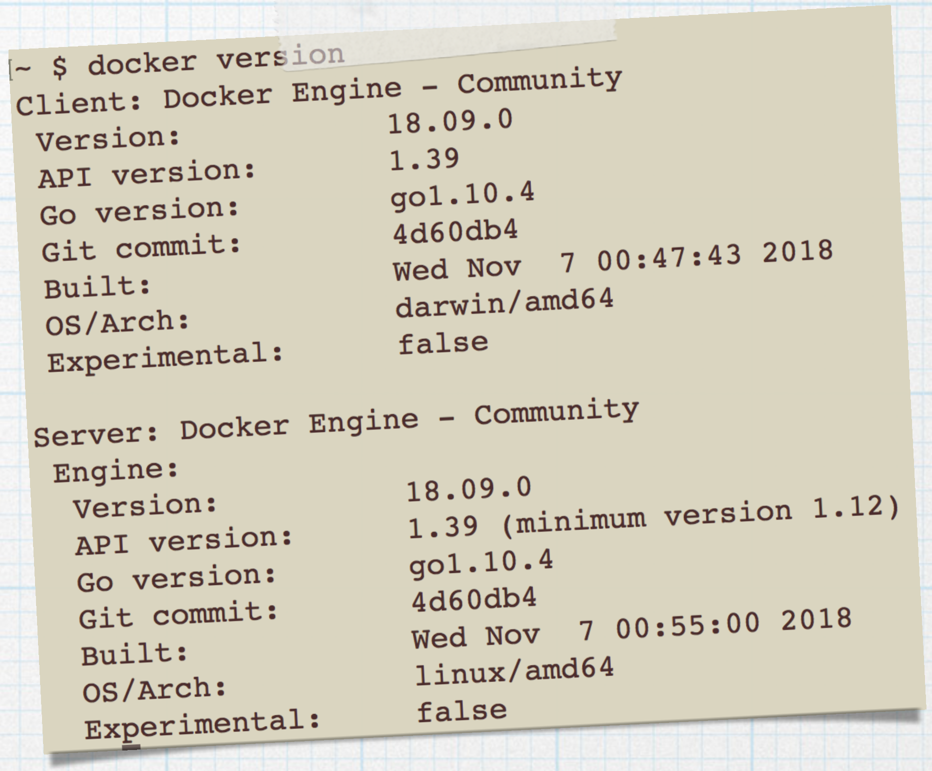






Verify Docker

* Run the following:
  * `docker version`
  * `docker info`
  * `docker run hello-world`



```
~ $ docker version
Client: Docker Engine - Community
 Version:           18.09.0
 API version:       1.39
 Go version:        go1.10.4
 Git commit:        4d60db4
 Built:             Wed Nov  7 00:47:43 2018
 OS/Arch:           darwin/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.0
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.4
  Git commit:       4d60db4
  Built:            Wed Nov  7 00:55:00 2018
  OS/Arch:          linux/amd64
  Experimental:     false
```

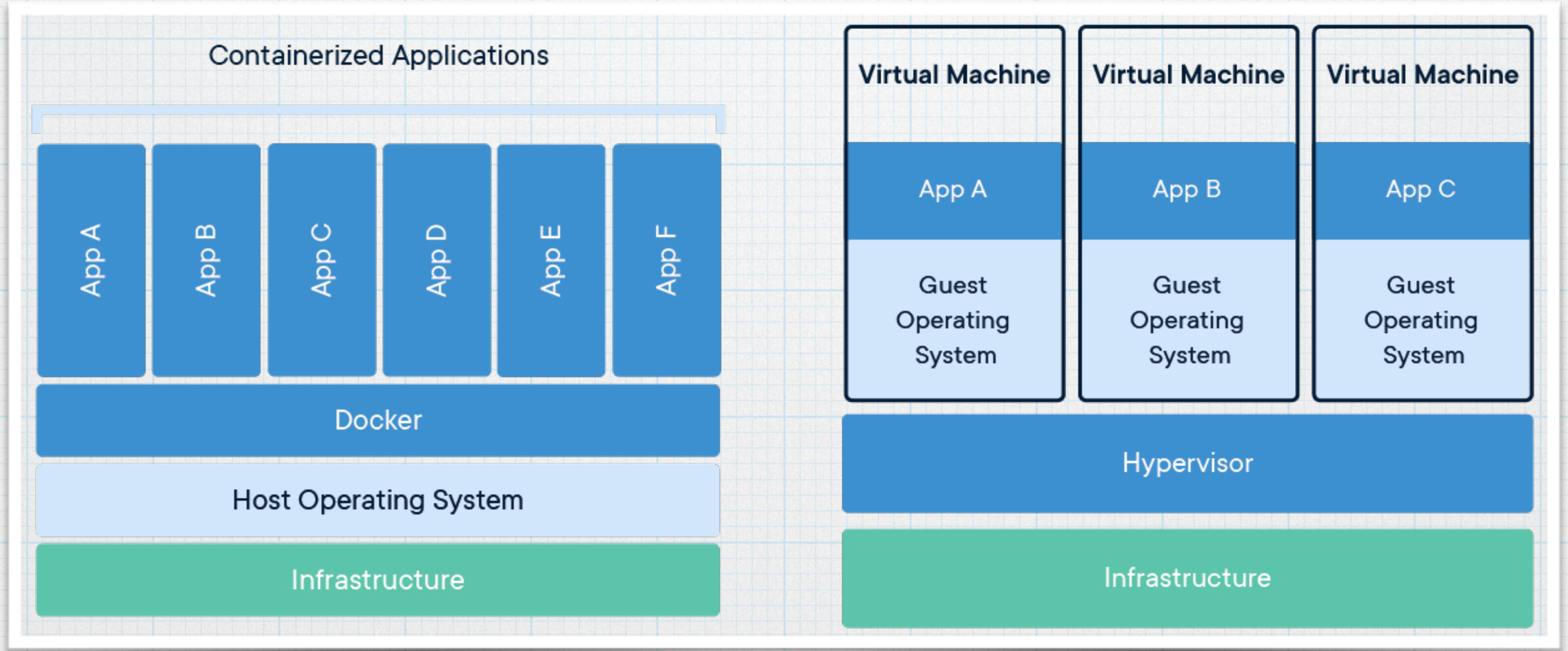
A Noob's Overview

The boring words part

* Benefit #1: Save Time And Money
* Benefit #2: Portability
* Benefit #3: Innovation and Empowerment

Save Time and Money

* Docker maximizes utilization of resources on a machine via the Docker daemon, thereby minimizing disk and memory usage

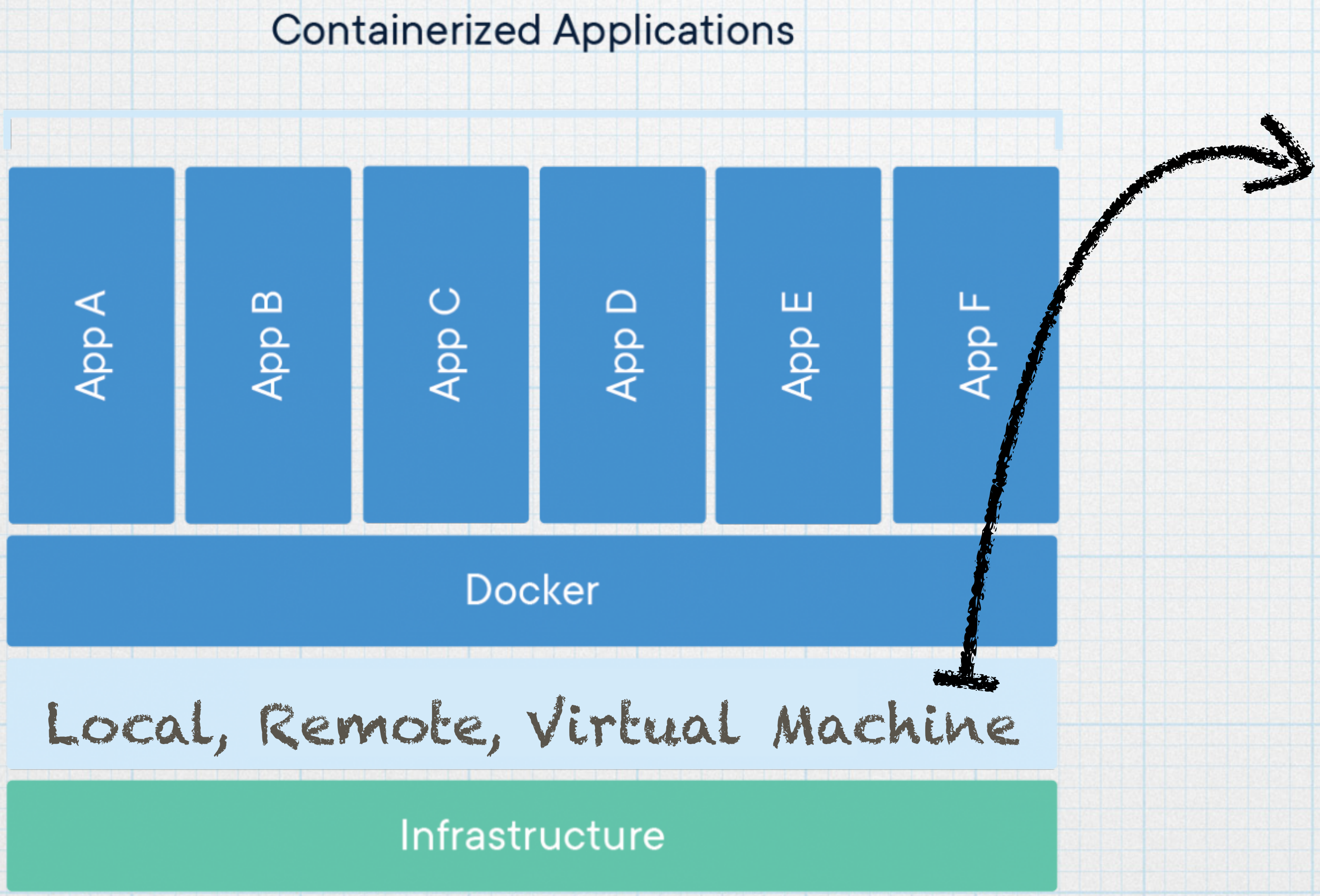


Save Time and Money

* What you don't **NEED**:
  * VM software/hypervisor, e.g. VirtualBox
  * `n` local guest OS's to mimic run environments, e.g. Windows Server, Solaris, CentOS
  * `n` virtual servers in your favorite cloud provider
* What you don't **GET**:
  * Loss of local CPU and memory for each running guest OS
  * Laborious uninstallation
  * Bored, because Docker starts in milliseconds

Portability

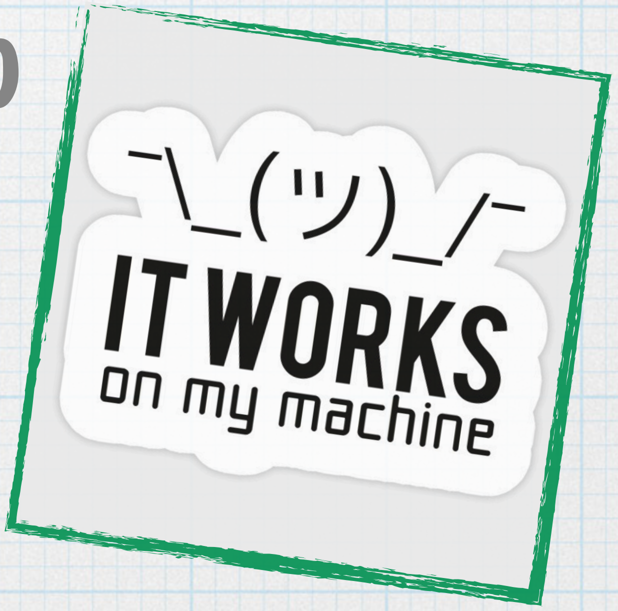
* As long as the Docker daemon is available you can run the same Docker images on any OS, in any environment



* Windows
* MacOS
* Linux
* AWS
* Azure

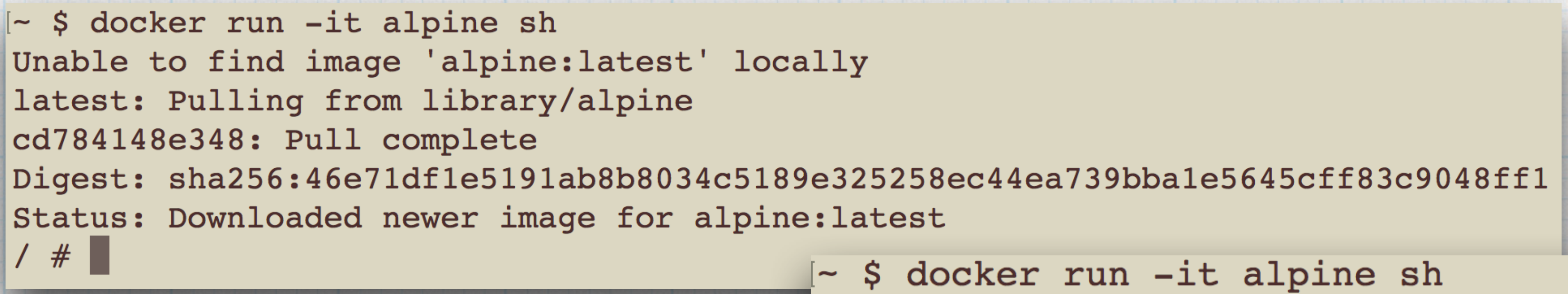
Innovation and Empowerment

* Try and assess new technologies quickly
* Choose the best tools for the job
* Collaborate over code, not setup
* Expose functionality via APIs
* Use the languages you want

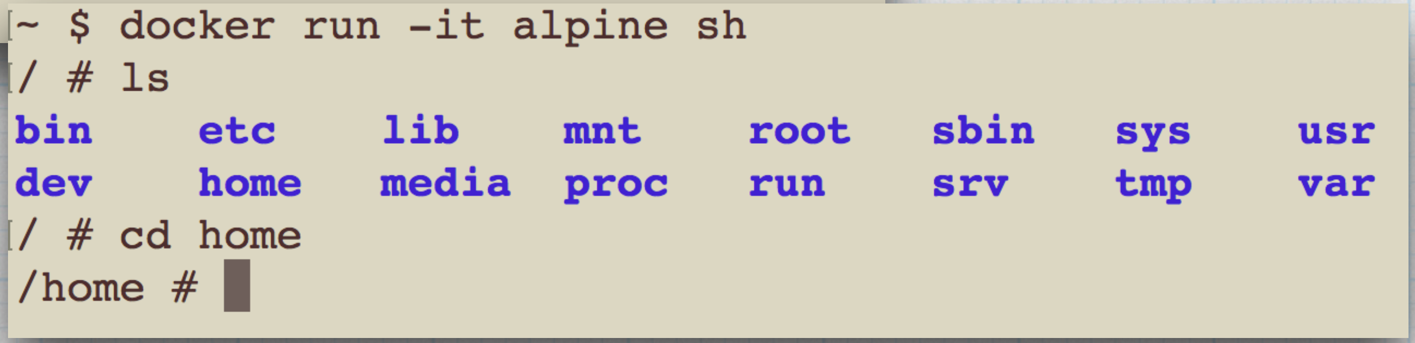


Yes, it's that easy

* `docker run -it alpine sh`
* Run some commands!



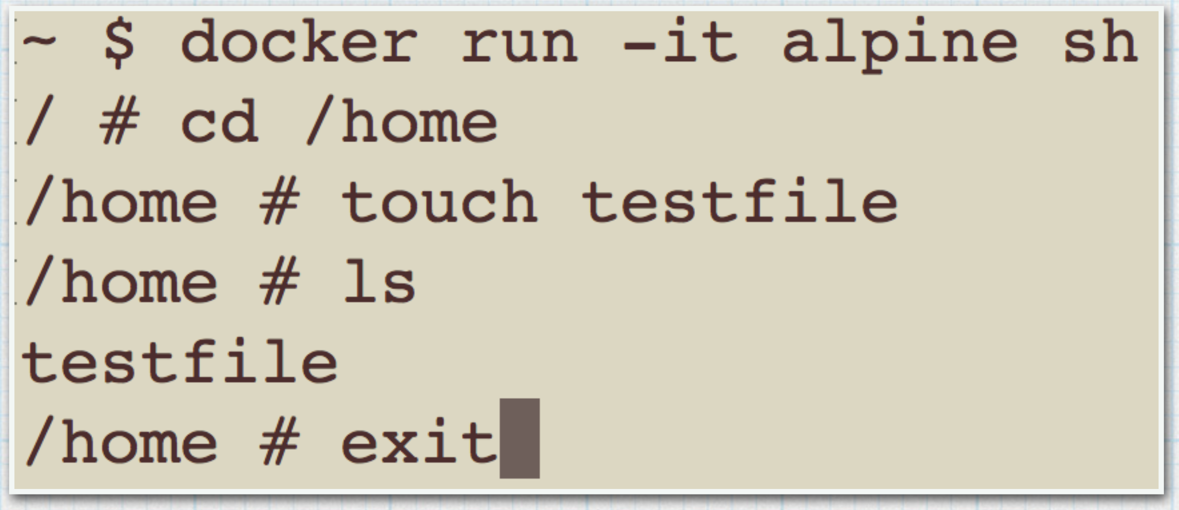
```
~ $ docker run -it alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
cd784148e348: Pull complete
Digest: sha256:46e71df1e5191ab8b8034c5189e325258ec44ea739bba1e5645cff83c9048ff1
Status: Downloaded newer image for alpine:latest
/ # 
```



```
~ $ docker run -it alpine sh
/ # ls
bin     etc     lib     mnt     root    sbin    sys     usr
dev     home    media   proc    run     srv     tmp     var
/ # cd home
/home # 
```

Containers are Immutable

- Run your Alpine interactive shell
- Create a file and then exit



```
~ $ docker run -it alpine sh
/ # cd /home
/home # touch testfile
/home # ls
testfile
/home # exit
```

- Run your Alpine interactive shell again

Lab #1

https://github.com/dspl283/DockerSLAC/tree/master/lab1

Base Images

* Docker base, or parent, images are minimal-dependency runtime environments for things like:
    * A Linux box
    * A specific programming language
    * A database installation
    * A web or application server
* Go to https://hub.docker.com

Dockerfile: The Magic

* If an image is a stack of layers, then the Dockerfile defines those layers

* Dockerfile format:

```
# Comment
INSTRUCTION arguments
```

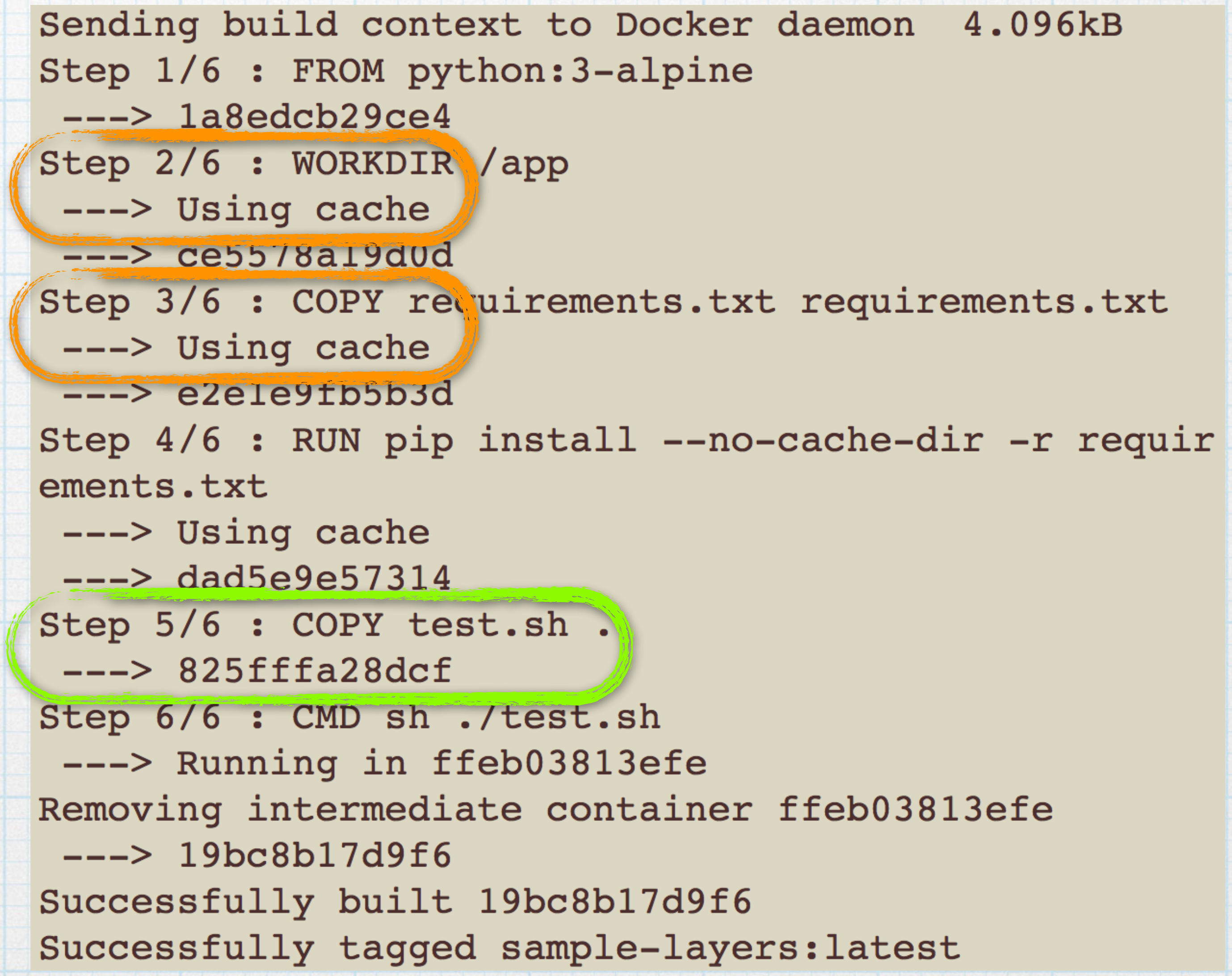
* https://docs.docker.com/engine/reference/builder

Dockerfile: An Example

```
# Base image
FROM python:3-alpine
# Command that will run in a shell
RUN mkdir /app
# Copies from build context to the container
COPY testfile.py /app
# Sets base directory for any
RUN, CMD, ENTRYPOINT, COPY or ADD instruction
that comes afterward
WORKDIR /app
# Defines an executing container and can only
exist once
CMD python ./testfile.py
```

Optimizing Layers

* Dockerfile instructions are read from top to bottom
* A build will utilize cached layers if no change is detected
* Make sure the most volatile instructions are closer to the bottom of the file, e.g. application code



```
Sending build context to Docker daemon  4.096kB
Step 1/6 : FROM python:3-alpine
 ---> 1a8edcb29ce4
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> ce5578a19d0d
Step 3/6 : COPY requirements.txt requirements.txt
 ---> Using cache
 ---> e2e1e9fb5b3d
Step 4/6 : RUN pip install --no-cache-dir -r requir
ements.txt
 ---> Using cache
 ---> dad5e9e57314
Step 5/6 : COPY test.sh .
 ---> 825fffa28dcf
Step 6/6 : CMD sh ./test.sh
 ---> Running in ffeb03813efe
Removing intermediate container ffeb03813efe
 ---> 19bc8b17d9f6
Successfully built 19bc8b17d9f6
Successfully tagged sample-layers:latest
```

Lab #2

https://github.com/dsp1283/DockerSLAC/tree/master/lab2

Lab #3

https://github.com/dsp1283/DockerSLAC/tree/master/lab3