THE O'REILLY COURSE - DAY ONE

DOCKER: BEYOND THE BASICS (CI/CD)

INSTRUCTOR: SEAN P. KANE

@SPKANE

FOLLOW ALONG

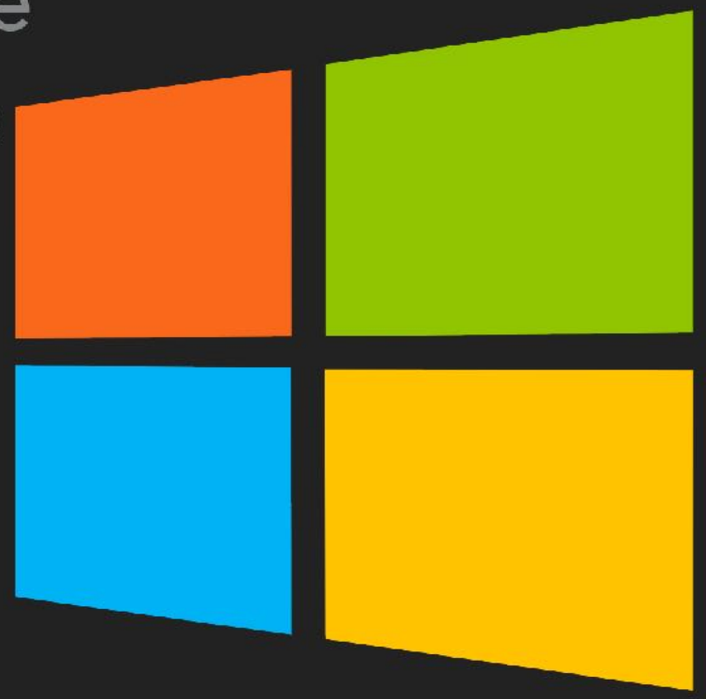
https://gist.github.com/spkane/df208d1c8fb644822f8d48e5ccf21fab

PREREQUISITES

- Recent computer and operating system
  - Recent Windows, Linux, or OS X
    - Tested on Windows 10 and OS X 10.12
  - root/admin rights
  - Sufficient resources to run 1 virtual machine
- Reliable and fast internet connectivity
- Git client
- KVM/xhyve/HyperV (or other VM technology)
- Docker Community Edition for Mac/Windows/Linux

A NOTE FOR WINDOWS USERS

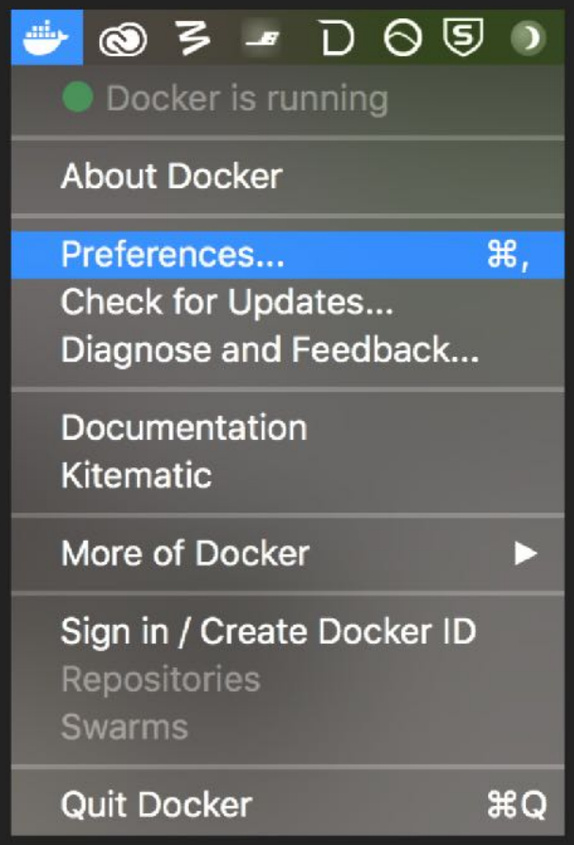
- This class was written from a largely Unix based perspective, but everything can be made to work in Windows with very little effort.

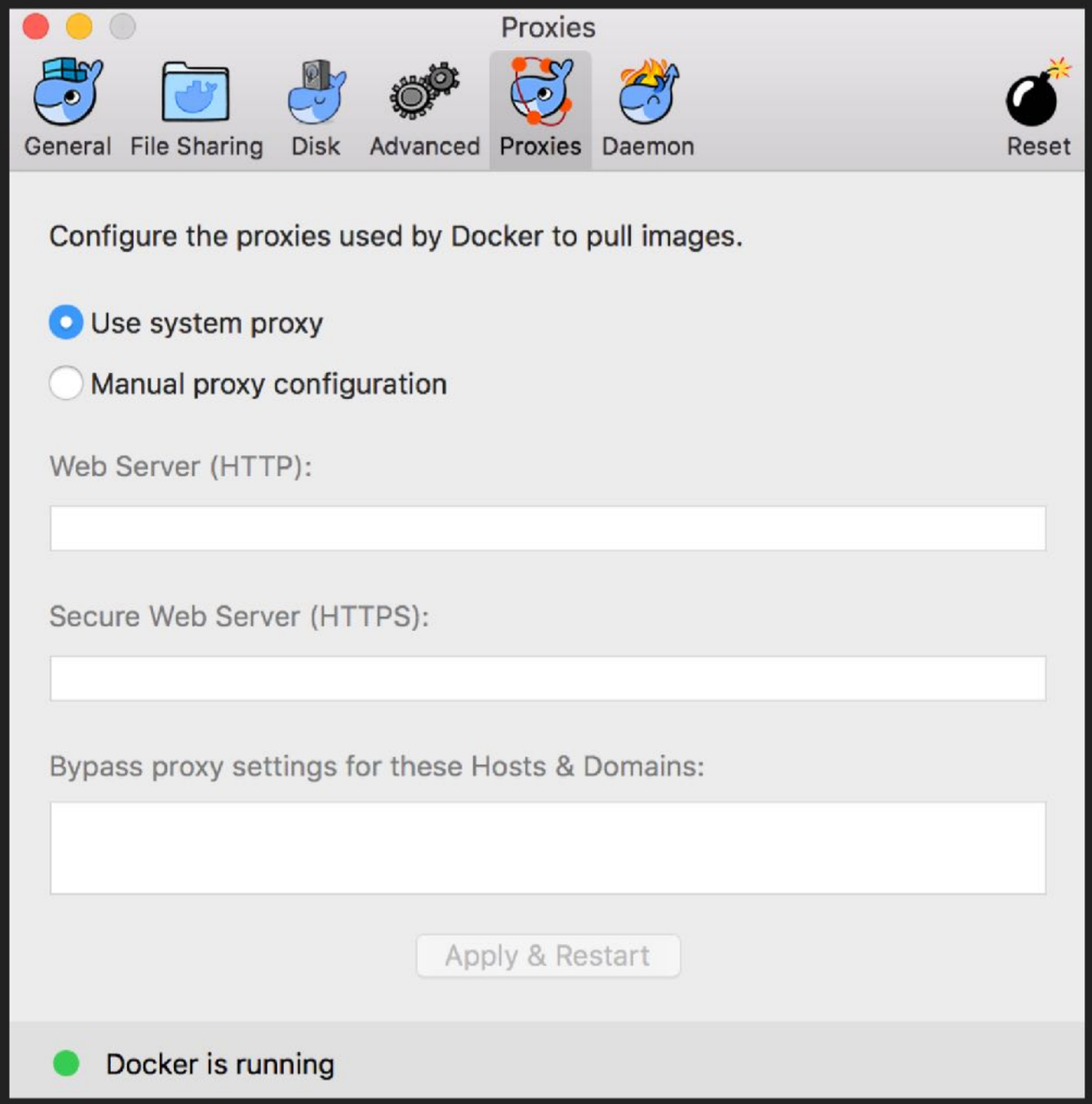


- Unix Variables
  - `export MY_VAR=test; echo ${MY_VAR}`
- Windows Variables (powershell)
  - `$env:my_var = "test"`
  - `Get-ChildItem Env:my_var`

A NOTE ABOUT PROXIES

- If required, you can configure a proxy in Docker: Community Edition via the preferences.



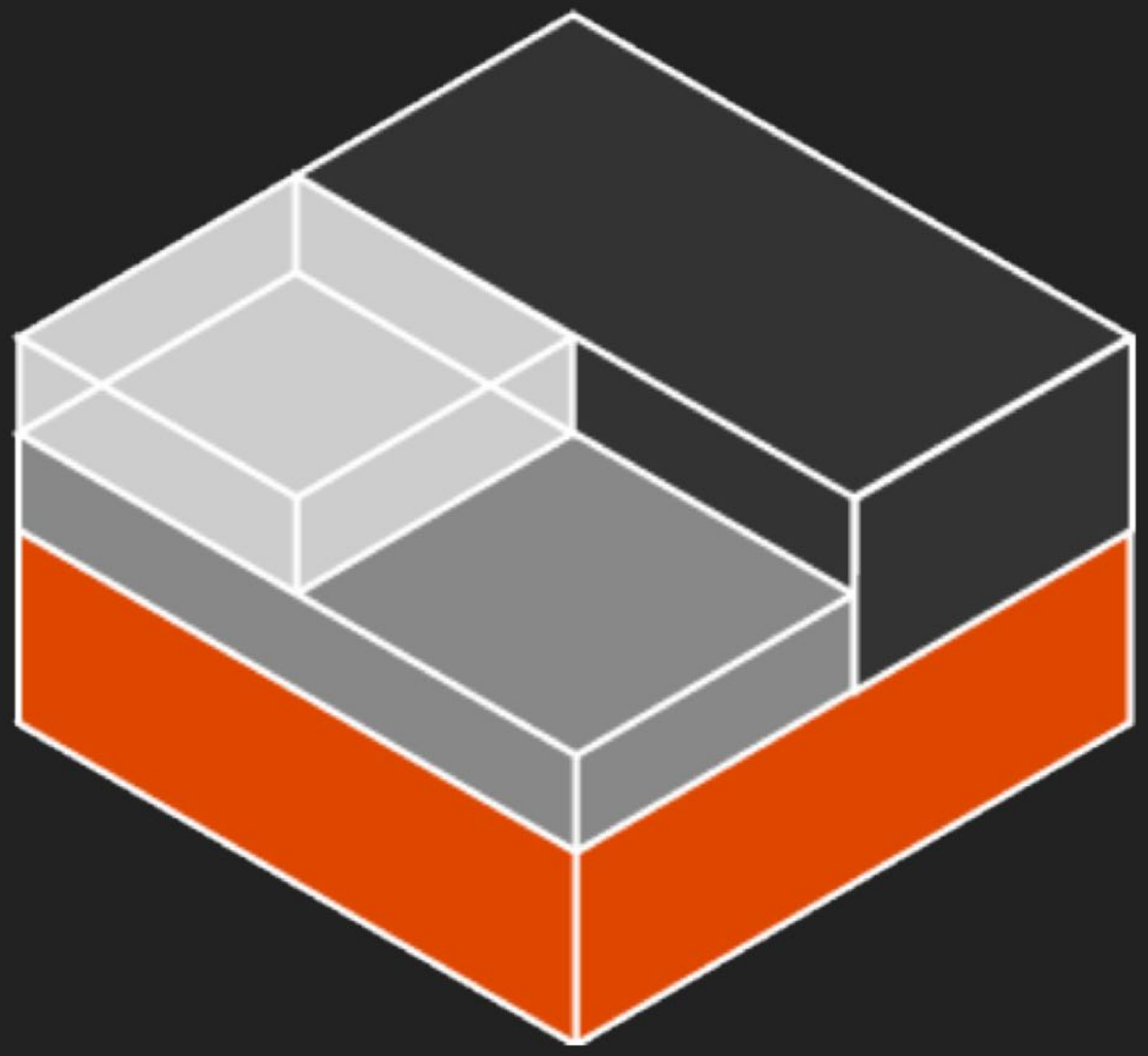


INSTRUCTOR ENVIRONMENT

- Operating System: Mac OS X (v10.12.X+)
- Terminal: iTerm2 (Build 3.X.X+)
  - https://www.iterm2.com/
- Shell Customization: Bash-it
  - https://github.com/Bash-it/bash-it
- Shell Prompt Theme: Zork
  - `export BASH_IT_THEME="zork"`
- Shell Prompt Font: Adobe Source Code Pro
  - https://github.com/adobe-fonts/source-code-pro
- Text Editor: Visual Studio Code (v1.X.X+)
  - https://code.visualstudio.com/

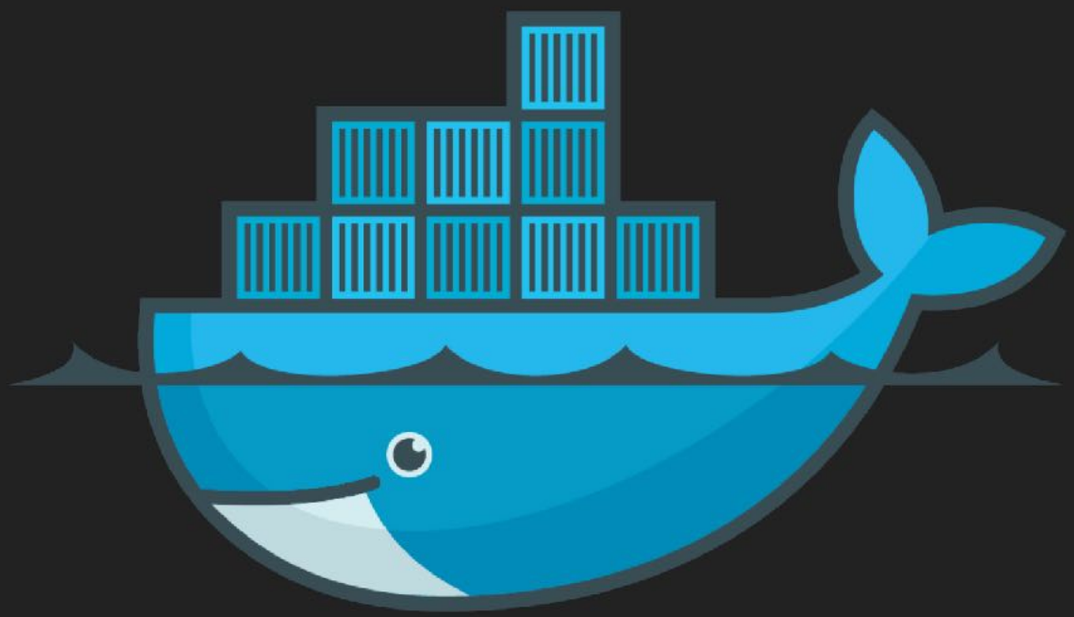
THE SLOW RISE OF LINUX CONTAINERS

- chroot system call (1979)
  - Version 7 Unix
- Virtuozzo (2000)
- jail (2000)
  - FreeBSD 4.0
- Solaris Zones (2004)
  - Solaris 10
- OpenVZ (2005)
- Linux Containers - LXC (2008)
  - version 2.6.24 of the Linux kernel



THE QUICK RISE OF DOCKER

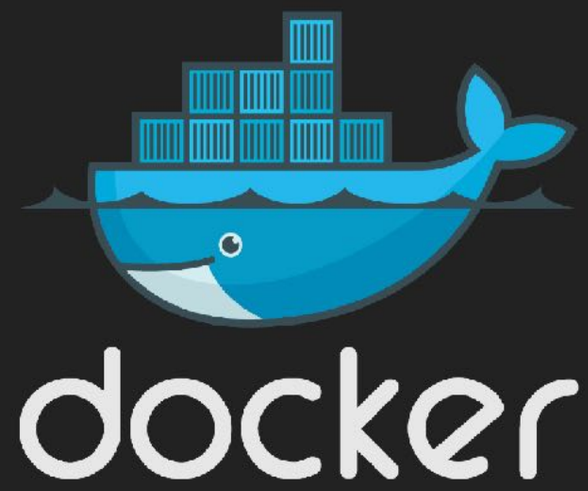
- Docker Engine
  - Announced March 15, 2013
- Provided:
  - Application packaging
  - App & dependencies packed together
  - Single deployment artifact
  - Abstract software from hardware without sacrificing resources



docker

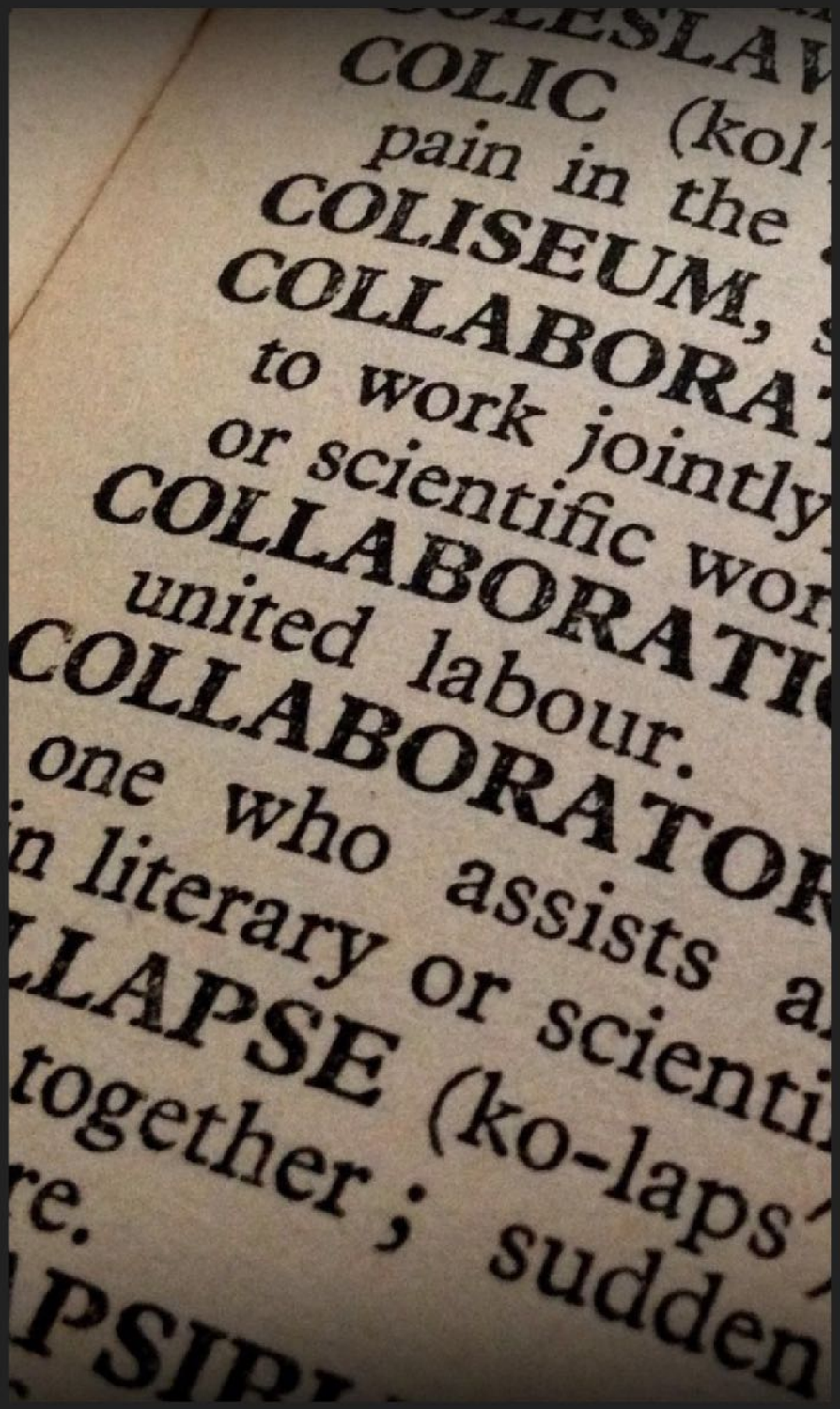
DOCKER ENGINE ISN'T A. . .

- virtualization platform (VMware, KVM, etc.)
- cloud platform (AWS, Azure, etc.)
- configuration management tool (Chef, Puppet, etc.)
- deployment framework (Capistrano, etc.)
- workload management tool (Mesos, Kubernetes, etc.)
- development environment (Vagrant, etc.)



DOCKER IN TRANSLATION

- **Docker client (Docker Engine)** - The docker command used to control most of the Docker workflow and talk to remote Docker servers.
- **Docker server (Docker Engine)** - The dockerd command used to launch the Docker daemon. This turns a Linux system into a Docker server that can have containers deployed, launched, and torn down via a remote client.
- **Virtual Machine (Docker CE)** - In general, the docker server can be only directly run on Linux. Because of this, it is common to utilize a Linux virtual machine to run Docker on other development platforms. Docker Community Edition makes this very easy.



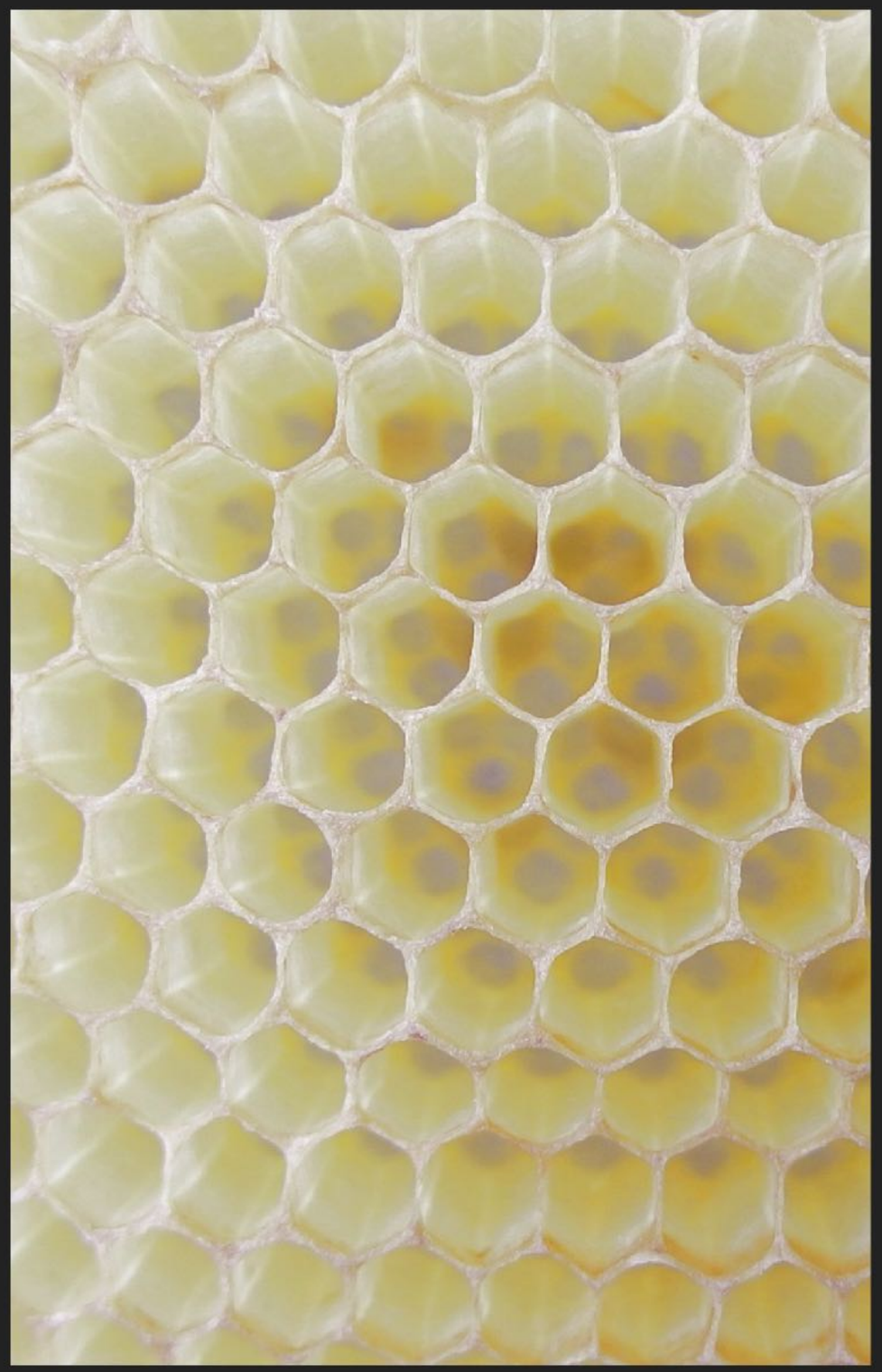
DOCKER IN TRANSLATION

- **Docker images** - Docker images consist of one or more filesystem layers and some important metadata that represent all the files required to run a Dockerized application. A single Docker image can be copied to numerous hosts. A container will typically have both a name and a tag. The tag is generally used to identify a particular release of an image.

- **Docker containers** - A Docker container is a Linux container that has been instantiated from a Docker image. A specific container can only exist once; however, you can easily create multiple containers from the same image.

LINUX NAMESPACES

- Mount (filesystem resources)
- UTS (host & domain name)
- IPC (shared memory, semaphores)
- PID (process tree)
- Network (network layer)
- User (user and group IDs)



CONTROL GROUPS (CGROUPS)

- Resource limiting
- Prioritization
- Accounting
- Control

SETTING THE STAGE

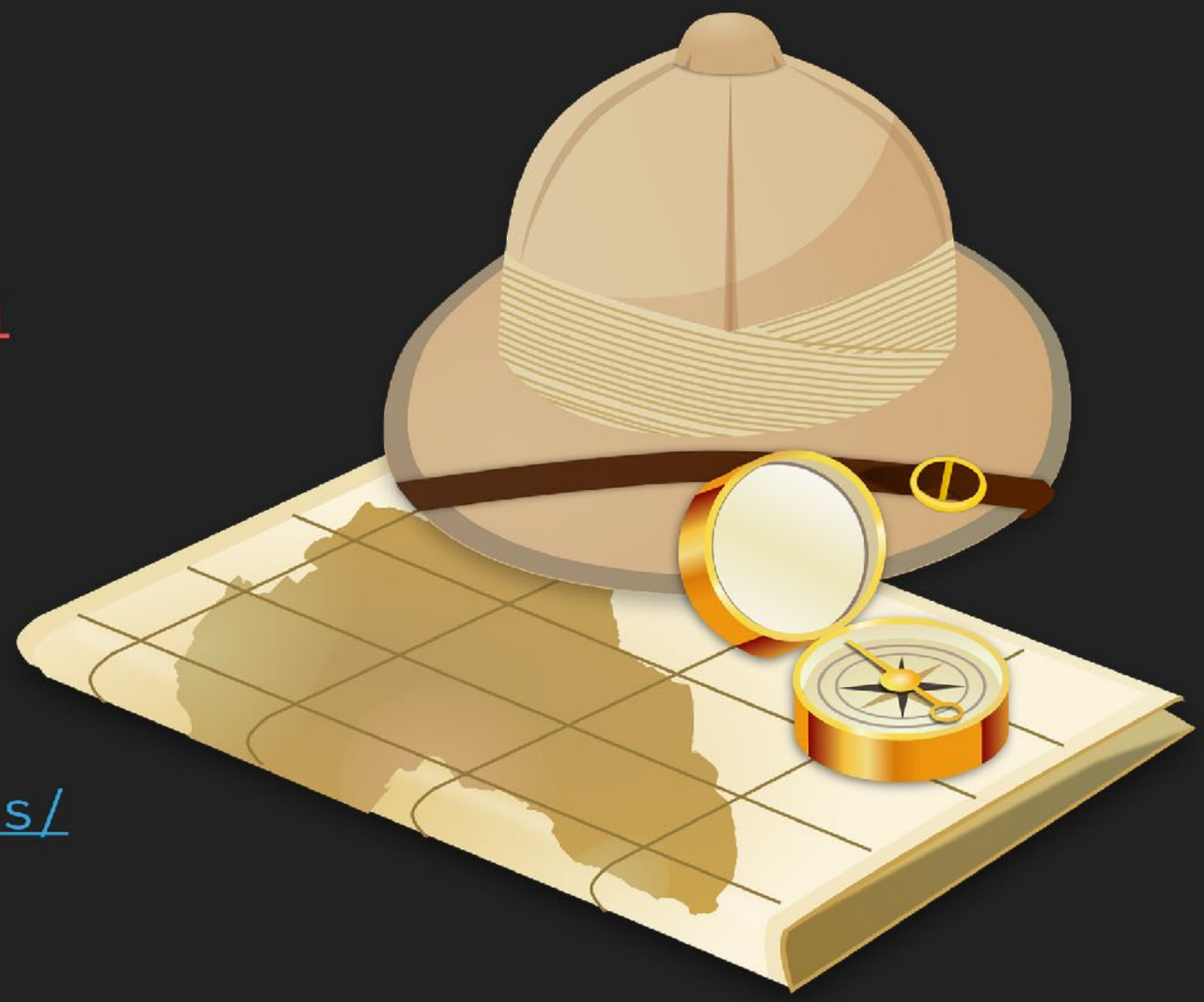
- `cd ${HOME}`
- `mkdir docker-class-201`
- `cd docker-class-201`
- `mkdir code`
- `git clone https://github.com/spkane/docker201.git layout --config core.autocrlf=input`
- `cd layout`
- `ls`

LAUNCHING A CONTAINER

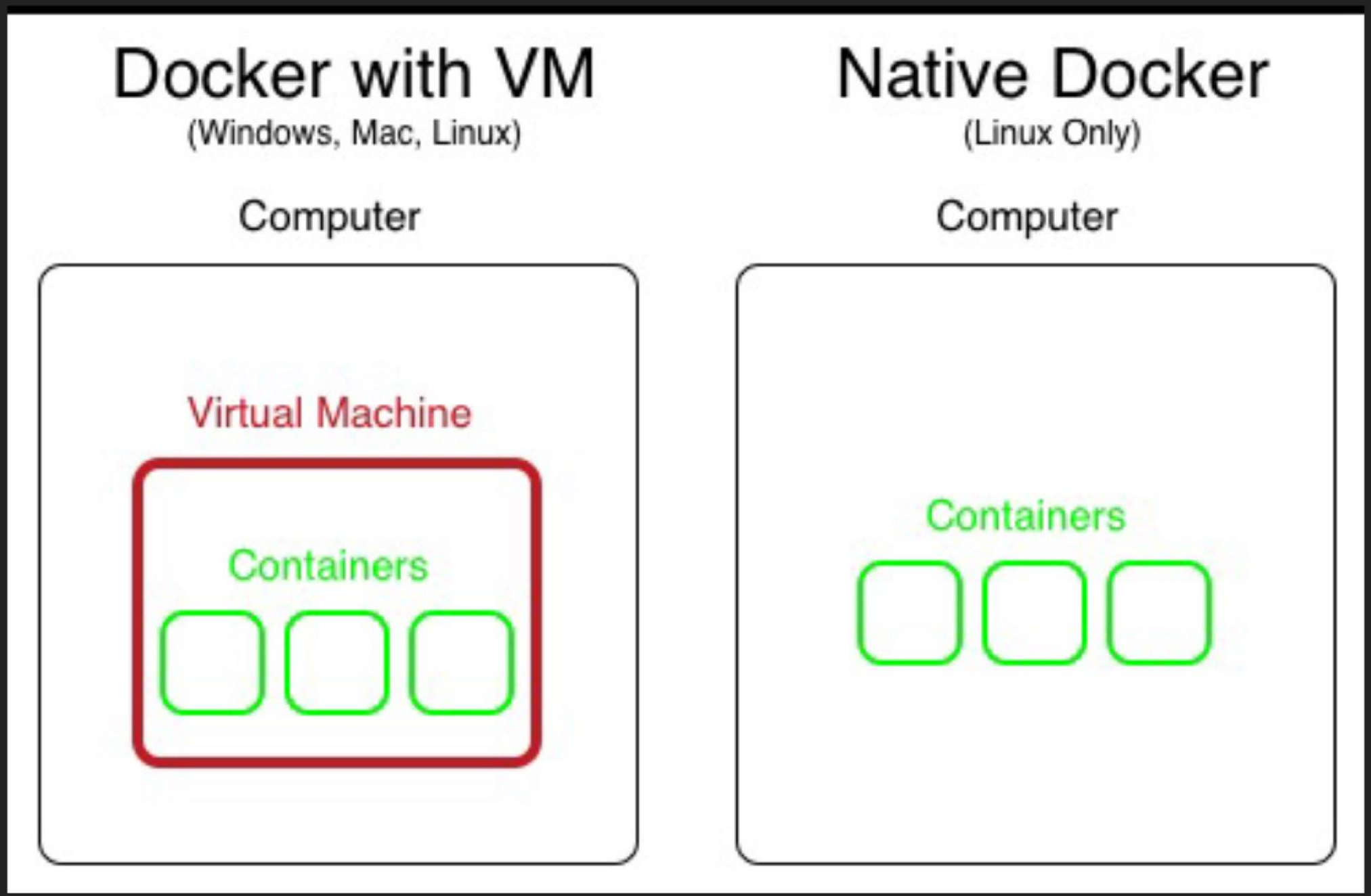
- `docker images`
- `docker run --rm -ti spkane/starwars`
  - Exit: CTRL-] followed by quit<return>
- `docker images`
- `docker ps -a`

EXPLORING THE DOCKER VM

- Based on Alpine Linux (apk)
- `docker run -it --privileged --pid=host debian nsenter -t 1 -m -u -n -i sh`
  - `cat /etc/os-release`
  - `exit`
- http://man7.org/linux/man-pages/man1/nsenter.1.html

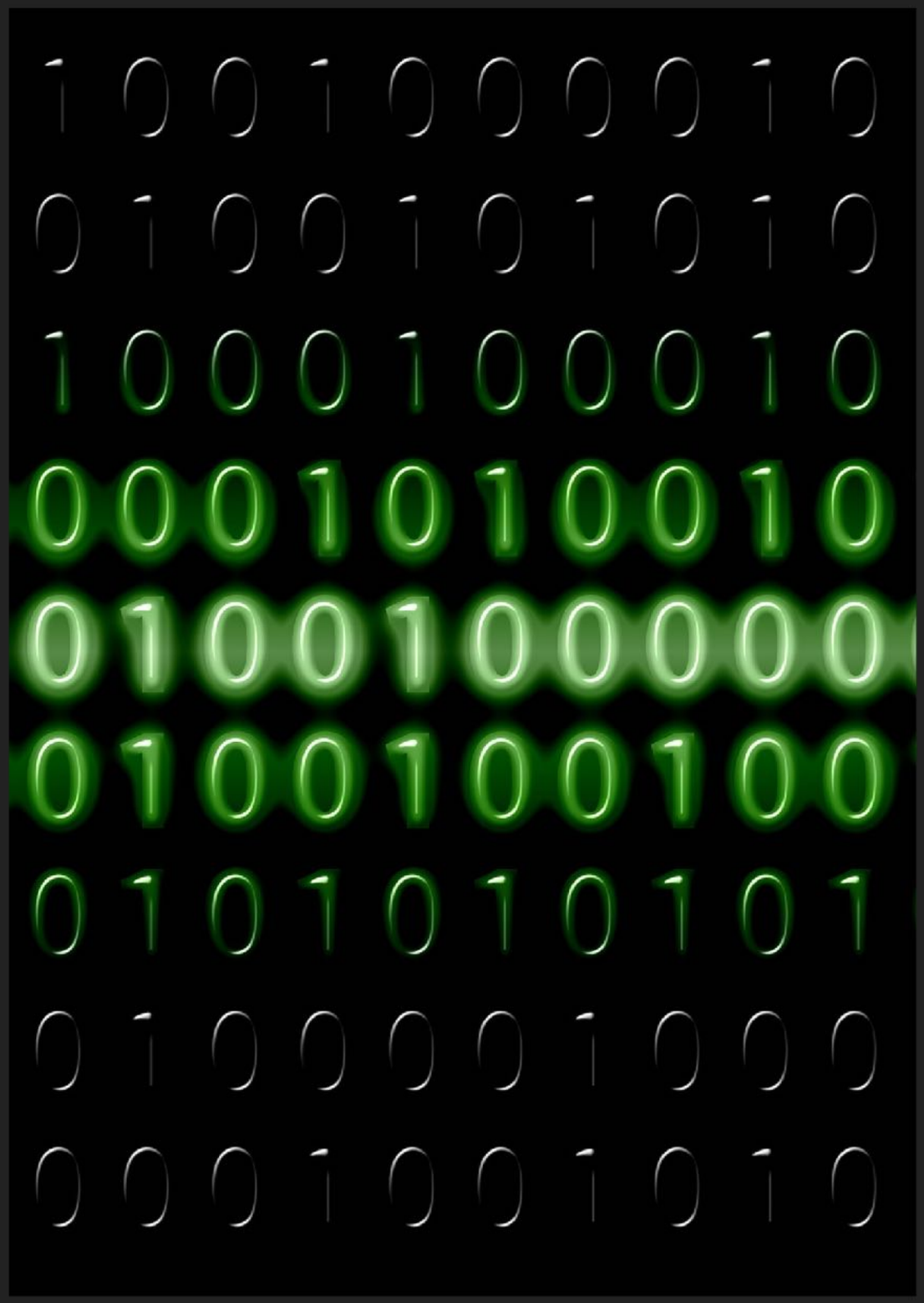


WHERE CONTAINERS RUN

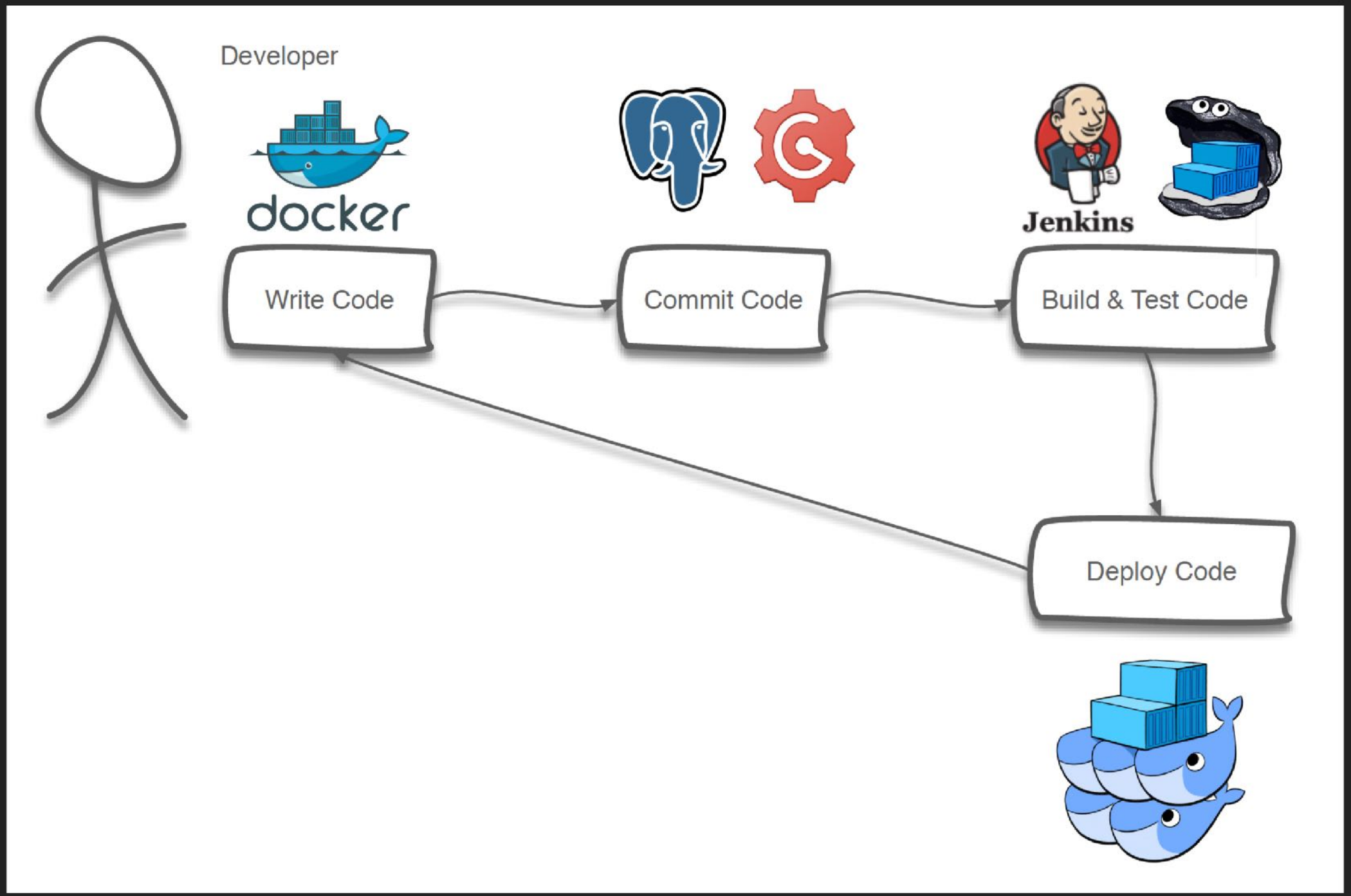


AUTOMATING WORKFLOW

- Datastore
  - Postgres
- Collaborative Source Code Repository
  - Gogs
- Docker Image Repository
  - Docker Distribution
- Build, Test, and Deploy
  - Jenkins
- Production Workload
  - Docker Swarm



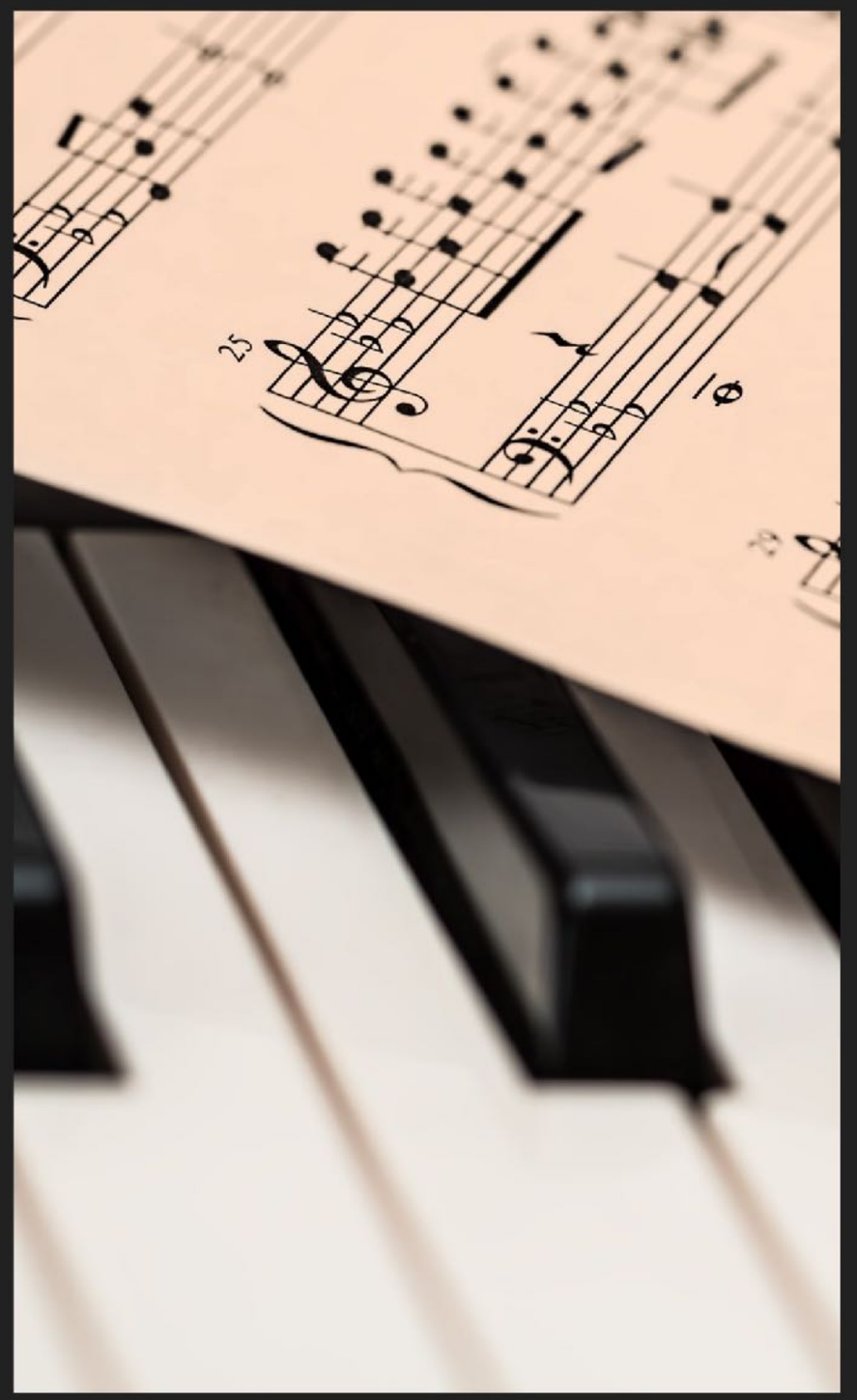
VISUALIZING WORKFLOW



COMPOSING A DOCKER SERVICE

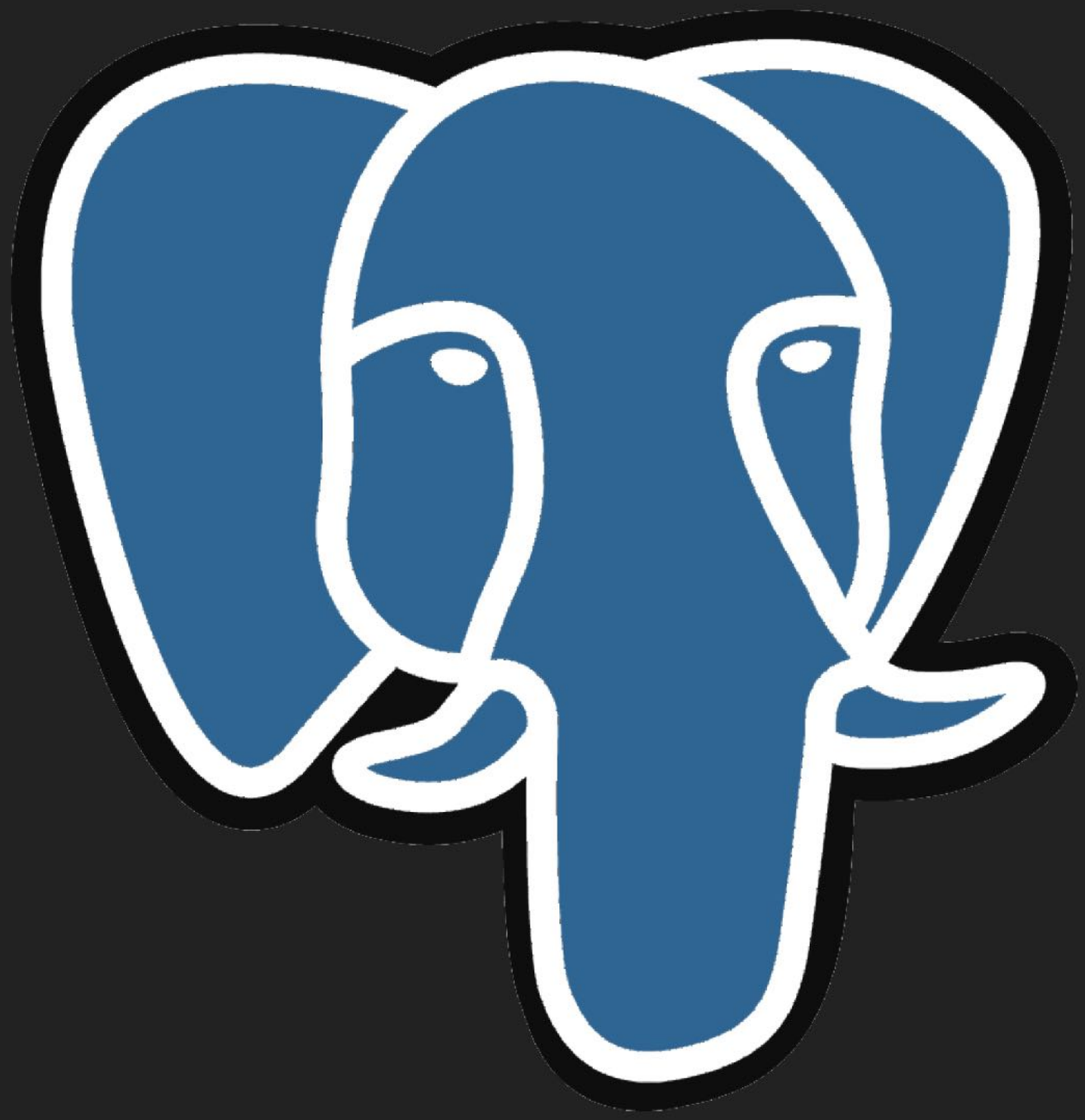
- (unix) `alias dc='docker-compose'`
- (win-ps) `New-Alias dc docker-compose.exe`

- Full Documentation:
  - https://docs.docker.com/compose/compose-file/



CREATING A DATASTORE

- `cd compose/review/1st`
- `vi docker-compose.yml`
  - Note DB user & password



CREATING A SOURCE REPO

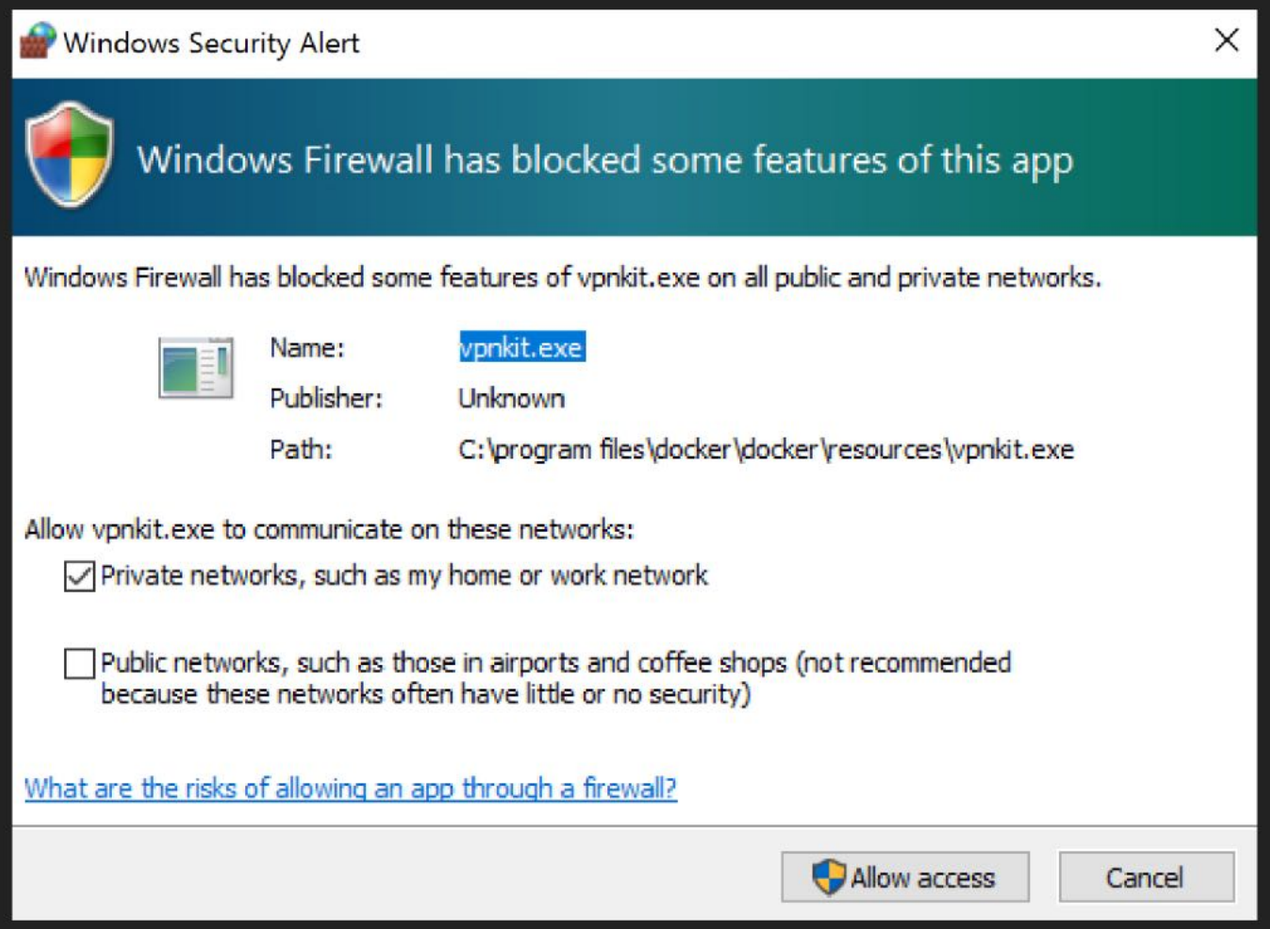
- `cd ../2nd`
- `vi docker-compose.yml`

DOCKER DISTRIBUTION

- cd ../3rd
- vi docker-compose.yml

A NOTE FOR WINDOWS USERS

- In the next section you might see these prompts more than once.
- Be sure to select "Allow access" and "Share it" each time that you see them.



JENKINS

- `cd ../../final/{unix,windows}`
- `vi docker-compose.yml`
- `docker-compose config`
- `docker-compose build`
- `docker-compose up -d`
- `docker-compose ps`
- `docker-compose logs -f`
  - 2017/07/01 20:06:31 [INFO] Listen: http://0.0.0.0:3000
  - LOG: database system is ready to accept connections
  - msg="debug server listening localhost:5001"
  - Please use the following password to proceed to installation

EXPLORING COMPOSE

- `docker-compose top`
- `docker-compose exec jenkins bash`
  - `exit`
- `docker-compose start`
- `docker-compose stop`
- `docker-compose pause`
  - (win) only Hyper-V containers can be paused
- `docker-compose unpause`
- # docker-compose down
  - **Don't run this command over the next 2 days.**

CONFIGURE GOGS

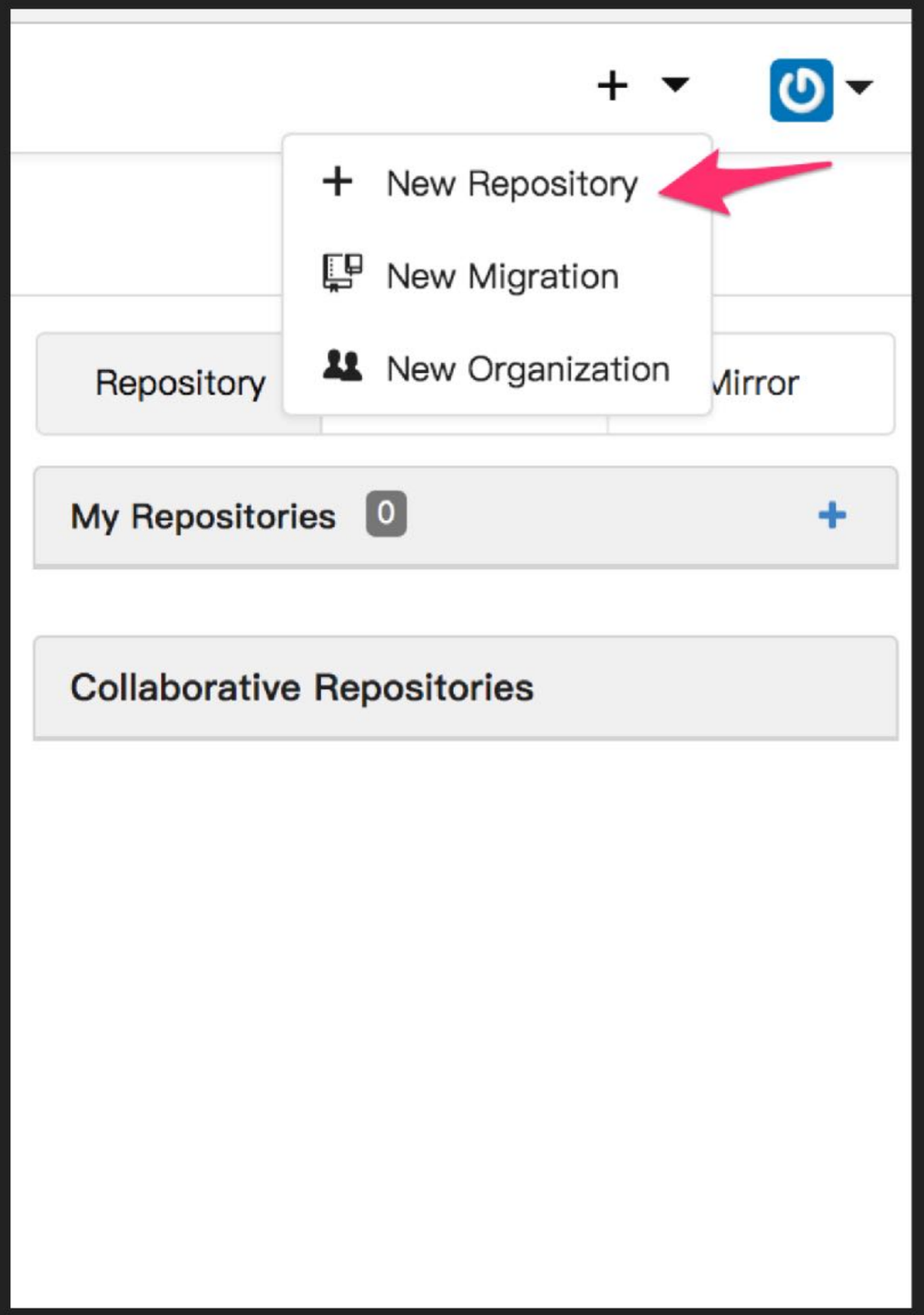
- Navigate web browser to:
  - http://127.0.0.1:10080/install
    - Database Type: Postgres
    - Host: postgres:5432
    - User: postgres
    - Password: myuser-pw!
    - SSH Port: 10022
    - Application URL: http://127.0.0.1:10080/
    - Create cogs user
      - Username: myuser
      - Password: myuser-pw!
      - Email Address: myuser@example.com

LOG IN TO GOGS

- Navigate web browser to:
  - http://127.0.0.1:10080/
    - Click: Sign In
      - Enter: Email Address
      - Enter: Password
      - Check: Remember Me
      - Click: Sign In

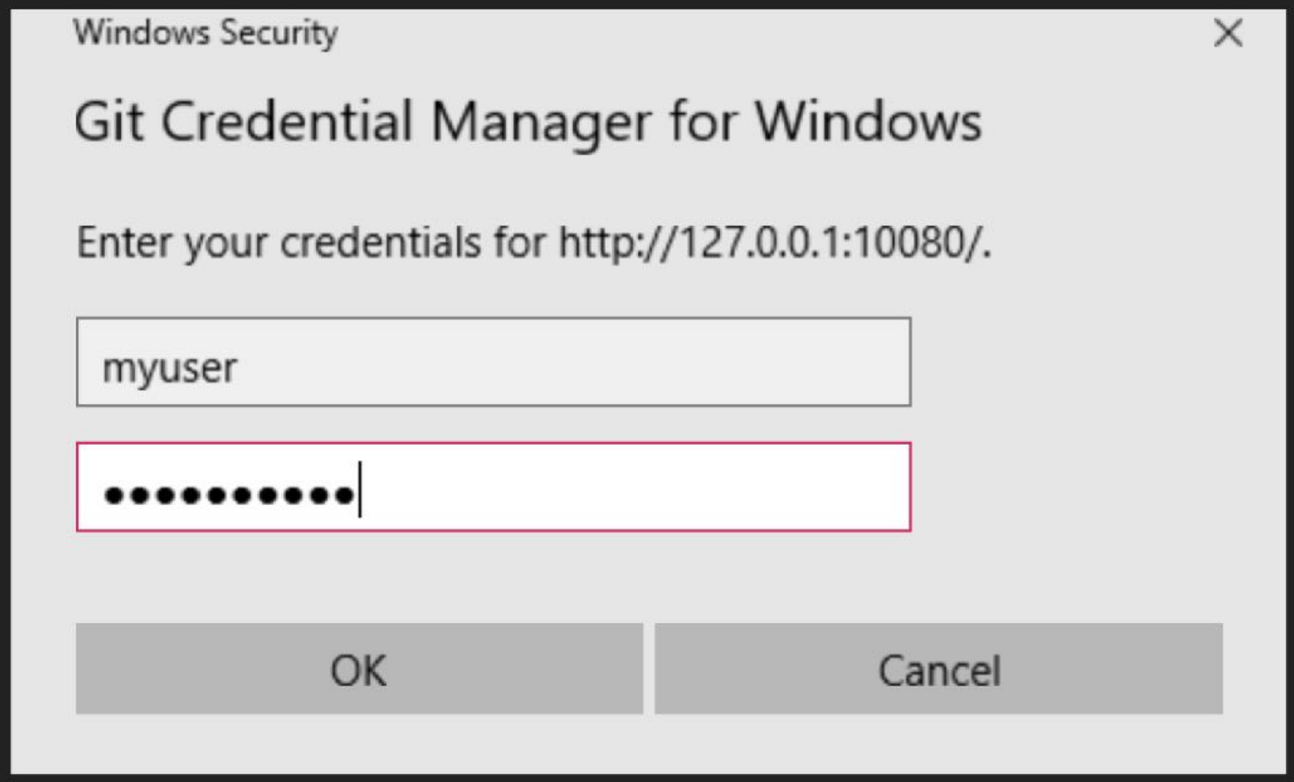
CREATE GIT REPO

- Click: +
  - Click: + New Repository
    - Repository Name: outyet
    - Click: Create Repository



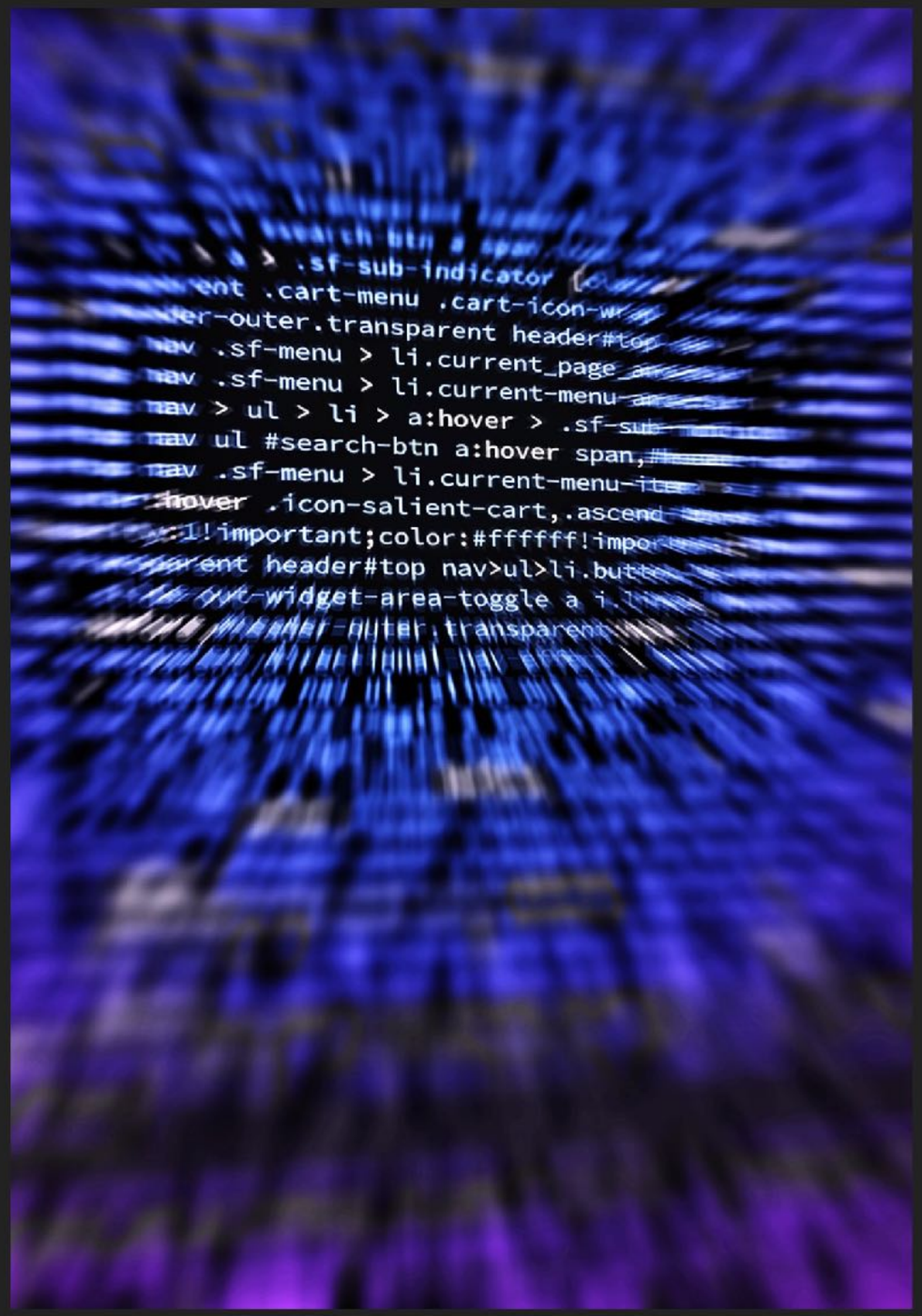
A NOTE FOR WINDOWS USERS

- In the next section you might see a GUI based password prompt from git.
- Be sure to provide your gogs username and password here.



FIRST CODE COMMIT

- `cd ../../..`
- `cp -a outyet ../code/`
  - (win) `mkdir ..\code\outyet`
  - (win) `xcopy outyet ..\code\outyet`
- `cd ../code/outyet/`
- `git init`
- `git config core.autocrlf input`
- `git add .`
- `git commit -m "first commit"`
- `git remote add origin http://127.0.0.1:10080/myuser/outyet.git`
- `git push -u origin master`
  - username: myuser
  - password: myuser-pw!



TEST DOCKER DISTRIBUTION

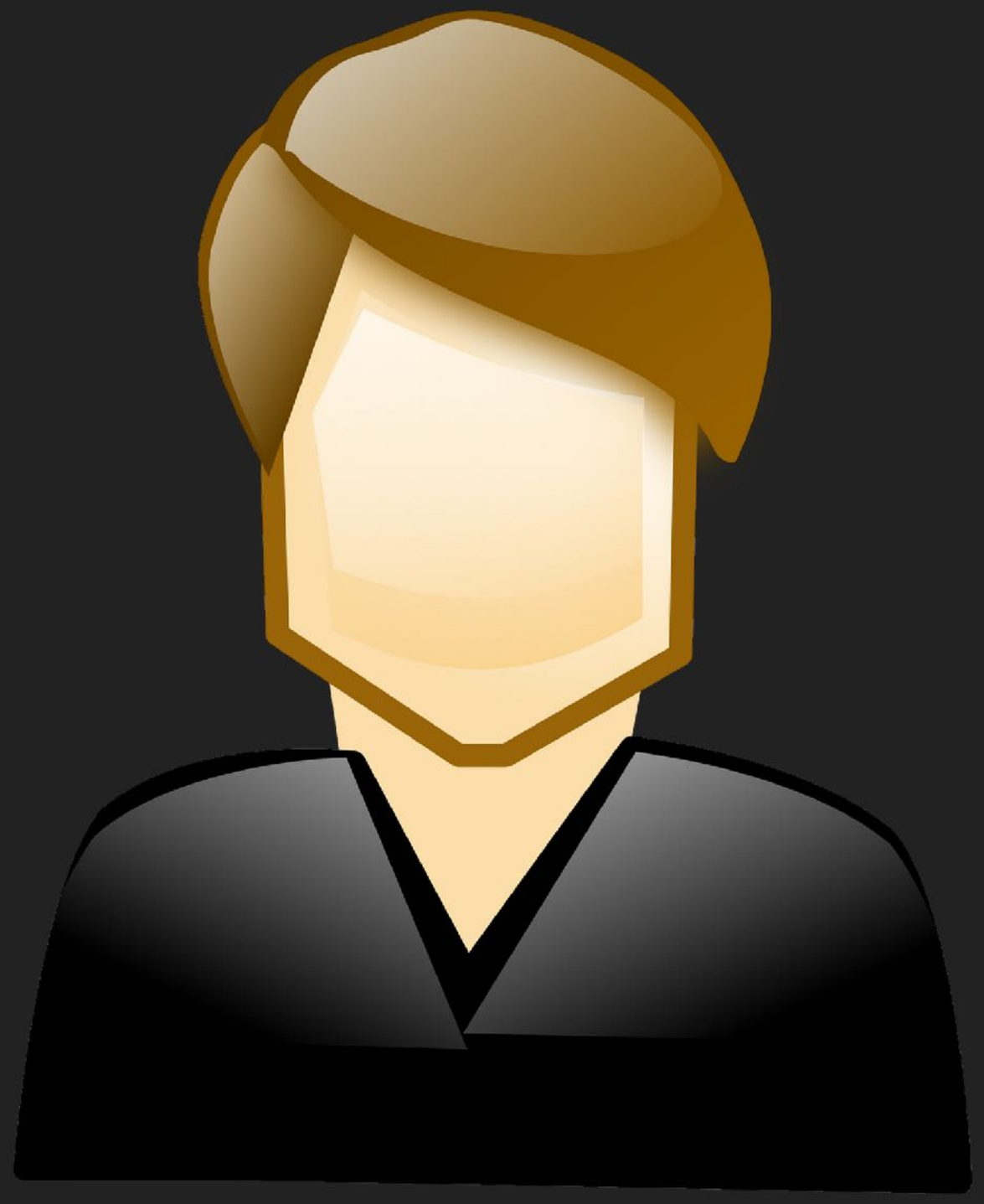
- `docker login 127.0.0.1:5000`
  - `myuser`
  - `myuser-pw!`
- `docker pull spkane/starwars:latest`
- `docker image ls spkane/starwars:latest`
- `docker tag ${IMAGE_ID} 127.0.0.1:5000/myuser/starwars:latest`
- `docker push 127.0.0.1:5000/myuser/starwars:latest`

CONFIGURE JENKINS

- `cat ../../layout/jenkins/data/secrets/initialAdminPassword`
- Navigate web browser to:
  - http://127.0.0.1:10081/
    - Paste Administrator Password
      - Click: Continue
    - Click: Select plugins to install
    - Click: None
      - Click: Install

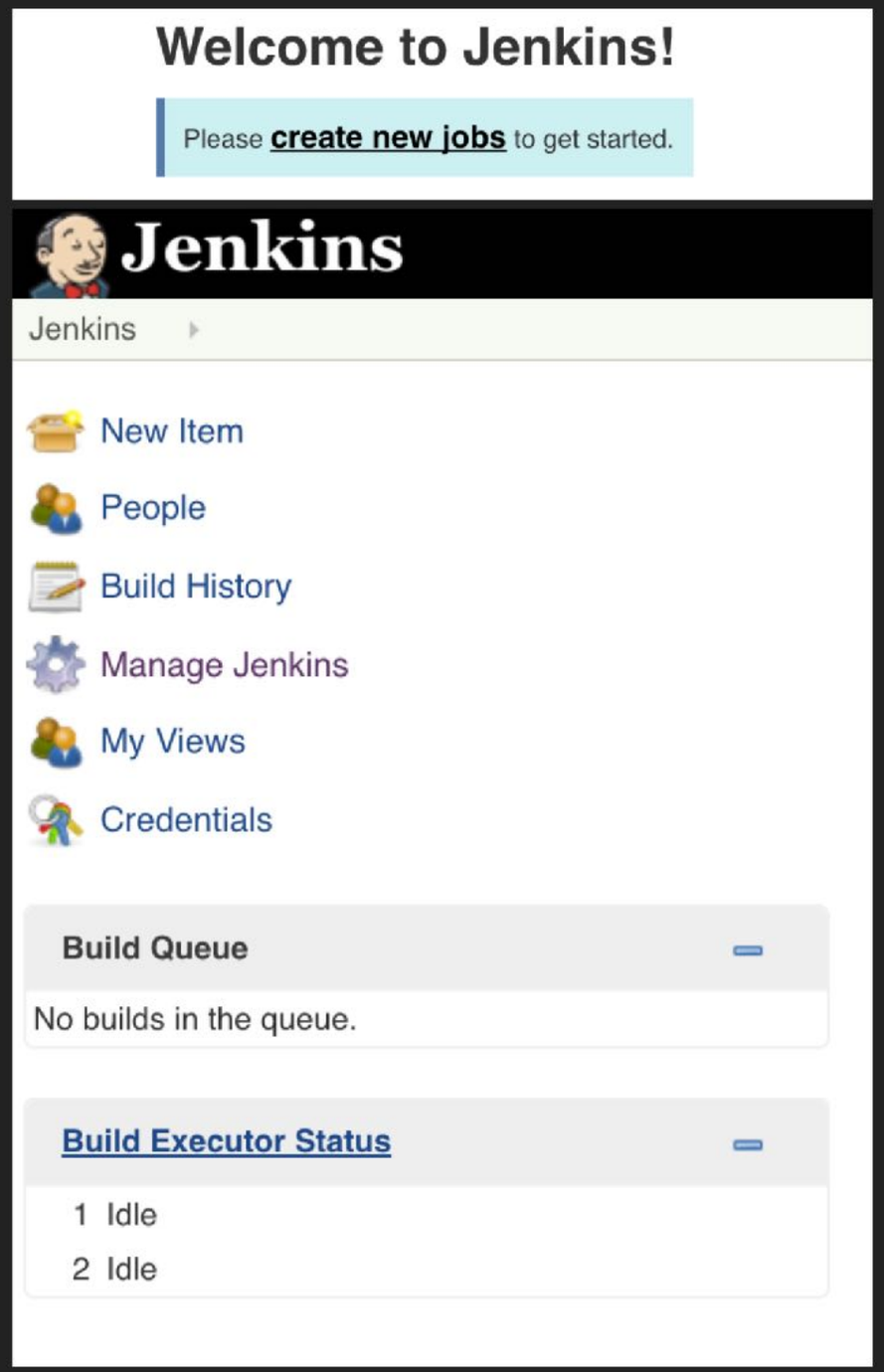
CONFIGURING JENKINS

- Create Admin User
  - Username: myuser
  - Password: myuser-pw!
  - Full Name: My User
  - E-Mail Address: myuser@example.com
    - Click: Save and Finish
- Click: Start Using Jenkins



COMPONENTS ASSEMBLED

- Postgres Database
  - https://www.postgresql.org/
- Gogs - Source Code Manager
  - https://gogs.io/
- Docker Distribution
  - https://github.com/docker/distribution
- Jenkins CI
  - https://jenkins.io/
- `cd ~/docker-class-201/layout/compose/final/{unix,windows}`
- `docker-compose stop`



WHAT HAVE WE LEARNED?

- General Docker Review
- Explored the Docker VM
- Docker Compose
  - Building / Running
  - Ports
  - Volumes
  - Networks
- Launched and configured:
  - Postgres / Gogs
  - Docker Distribution
  - Jenkins

IMAGE ATTRIBUTION

https://pixabay.com/

https://linuxcontainers.org/

https://www.docker.com/

https://www.microsoft.com/

https://gogs.io/

https://www.postgresql.org/

https://jenkins.io/

https://traefik.io/

INSTRUCTOR: SEAN P. KANE

@SPKANE