

Docker: Beyond the Basics

CI/CD (Day One)

Instructor
Sean P. Kane
@spkane

Follow Along Guide

Textual Slides

Prerequisites

- A recent computer and OS
 - Recent Linux, OS X, or Windows 10
 - root/admin rights
 - Sufficient resources to run one 2 CPU virtual machine (VM)
 - CPU Virtualization extensions **MUST** be enabled in your BIOS/EFI
 - Reliable and fast internet connectivity
- Docker Community Edition

Prerequisites

- A graphical web browser
- A text editor
- A software package manager
- Git client
- General comfort with the command line will be helpful.
- [optional] tar, wget, curl, jq, SSH client

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 10 Variables (powershell)

- `$env:my_var = "test"`
- `Get-ChildItem Env:my_var`

A Note About Proxies

Proxies can interfere with some Docker activities if they are not configured correctly.

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/>

Docker in Translation

- **Docker client**

- The docker command used to control most of the Docker workflow and talk to remote Docker servers.

- **Docker server**

- 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.

Docker in Translation

- **Virtual Machine**

- 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 in Translation

- **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.

The Slow Rise Of Linux Containers

- **chroot system call** (*1979*)
 - Version 7 Unix
- **jail** (*2000*)
 - FreeBSD 4.0
- **Solaris Zones** (*2004*)
 - Solaris 10
- **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 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.)

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

Testing the Docker Setup

```
$ docker images  
$ docker run -d --rm --name quantum -p 8080:8080 spkane/quantum-game:latest  
$ docker ps
```

- In a web browser, navigate to port 8080 on your Docker server.
 - (e.g.) <http://127.0.0.1:8080/>

```
$ docker kill quantum  
$ docker ps
```


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>

```
$ docker run -ti --rm spkane/quantum-game:latest cat /etc/os-release
```


Setting the Stage

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


Automating Workflow

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

Iterative Workflow

- Core Technology - Docker

User develops code locally (Docker)

User commits code (Gogs backed by Postgres)

Pipeline builds & tests code (Jenkins & Docker Distribution)

Pipeline deploys code to production.

and then iterate...

Composing a Docker Service

For unix: `$ alias dc='docker-compose'`

For Windows Powershell: `PS C:\> New-Alias dc docker-compose.exe`

- Open & explore docker-compose.yaml in your text editor
- 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
```


Important Note For Windows Users

- In the next section you might see:
 - a Windows Security Alert for `vpnkit.exe`, be sure and select `Allow access`.
 - Multiple `Docker for Windows - Share drive` alerts. Be sure and select `Share it` for each prompt.
- If you have problems with file mounts you may need to set:
`$Env:COMPOSE_CONVERT_WINDOWS_PATHS=1`

Jenkins

```
$ cd ../../final/{unix,windows}
$ vi docker-compose.yml
$ docker-compose config
$ 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
```


Important

Note: Don't run `docker-compose down` until class is over.

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 Gogs User

Username: myuser

Password: myuser-pw!

Confirm Password: myuser-pw!

Email Address: myuser@example.com

Click: Install Gogs

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 for the prompt.

Explore the Code

```
$ cd ~/class-docker-cicd/code/outyet
```

- Explore with your favorite code editor
 - Dockerfile
 - main.go
 - main_test.go

```
docker-compose up -d
```


Examine Application

- Navigate web browser to:
 - <http://127.0.0.1:10088/>

```
$ docker-compose down
```


First Code Commit

```
$ cd ../../..  
$ cp -a outyet ../code/  
$ 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
```

- username: myuser
- password: myuser-pw!

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

Confirm password: myuser-pw!

Full Name: My User

E-Mail Address: myuser@example.com

Click: Save and Continue

Configuring Jenkins

- Final Details

Jenkins URL: `http://127.0.0.1:10081/`

Click: `Save and Finish`

Click: `Start Using Jenkins`

Shutdown Services

```
$ cd ~/class-docker-cicd/layout/compose/final/{unix,windows}  
$ docker-compose stop
```


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/>

What We Have Learned

- Docker Compose
- Building / Running
- Ports, Volumes, and Networks
- Launched and configured:
 - Postgres / Gogs
 - Docker Distribution
 - Jenkins

Additional Learning Resources

<https://www.safaribooksonline.com/>

Any Questions?