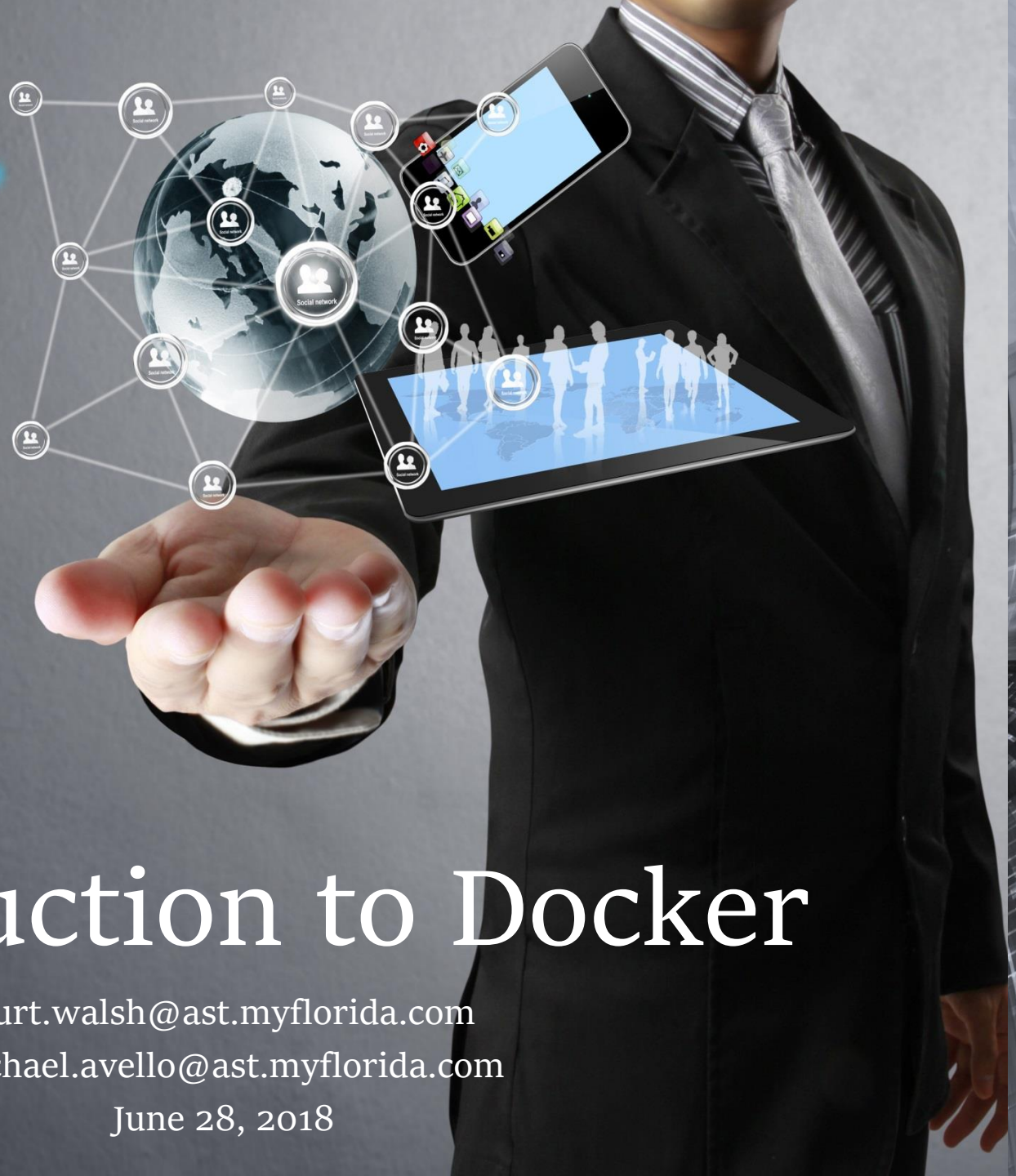


AGENCY FOR STATE TECHNOLOGY

Introduction to Docker

burt.walsh@ast.myflorida.com
michael.avello@ast.myflorida.com

June 28, 2018

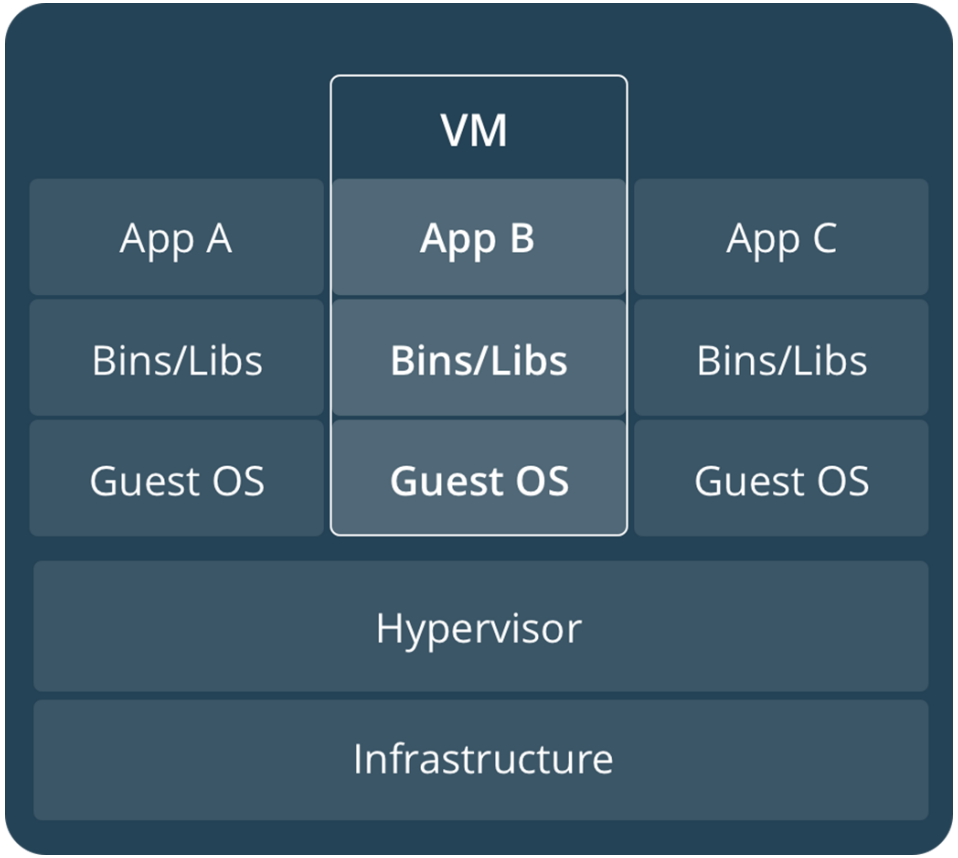
Parts of Presentation

Part One Docker Concepts

Part Two Building a Multi-tier System

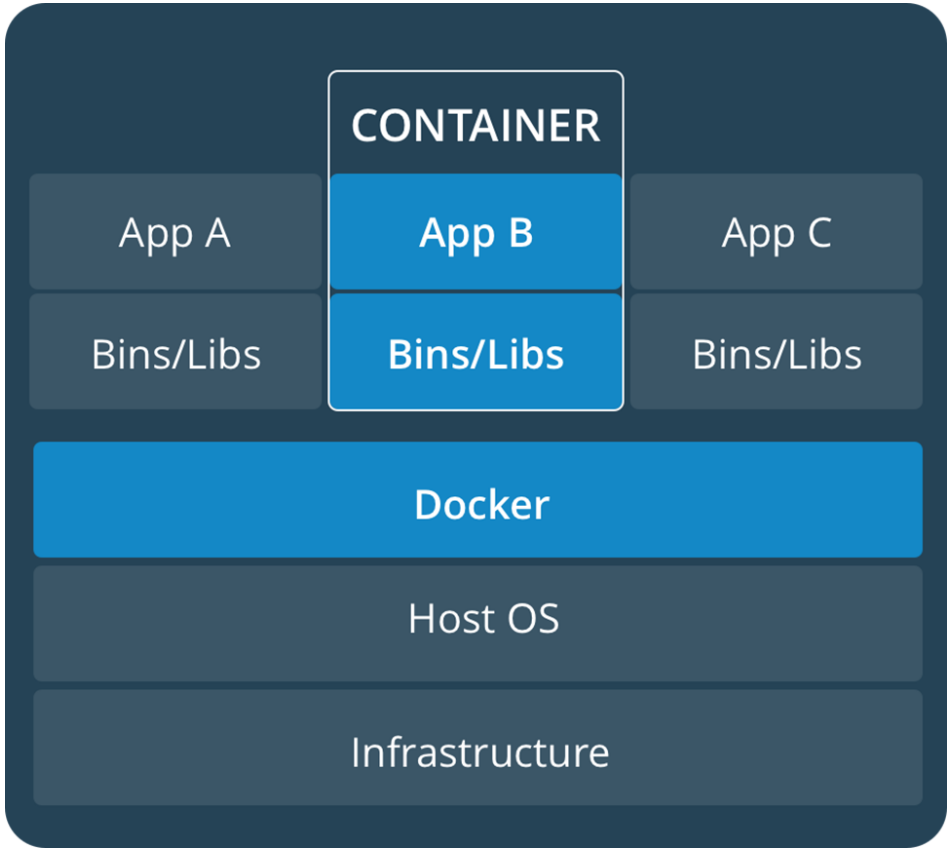
Part One Docker Concepts

Virtual Machine
(long boot time/machine/uses many resources)

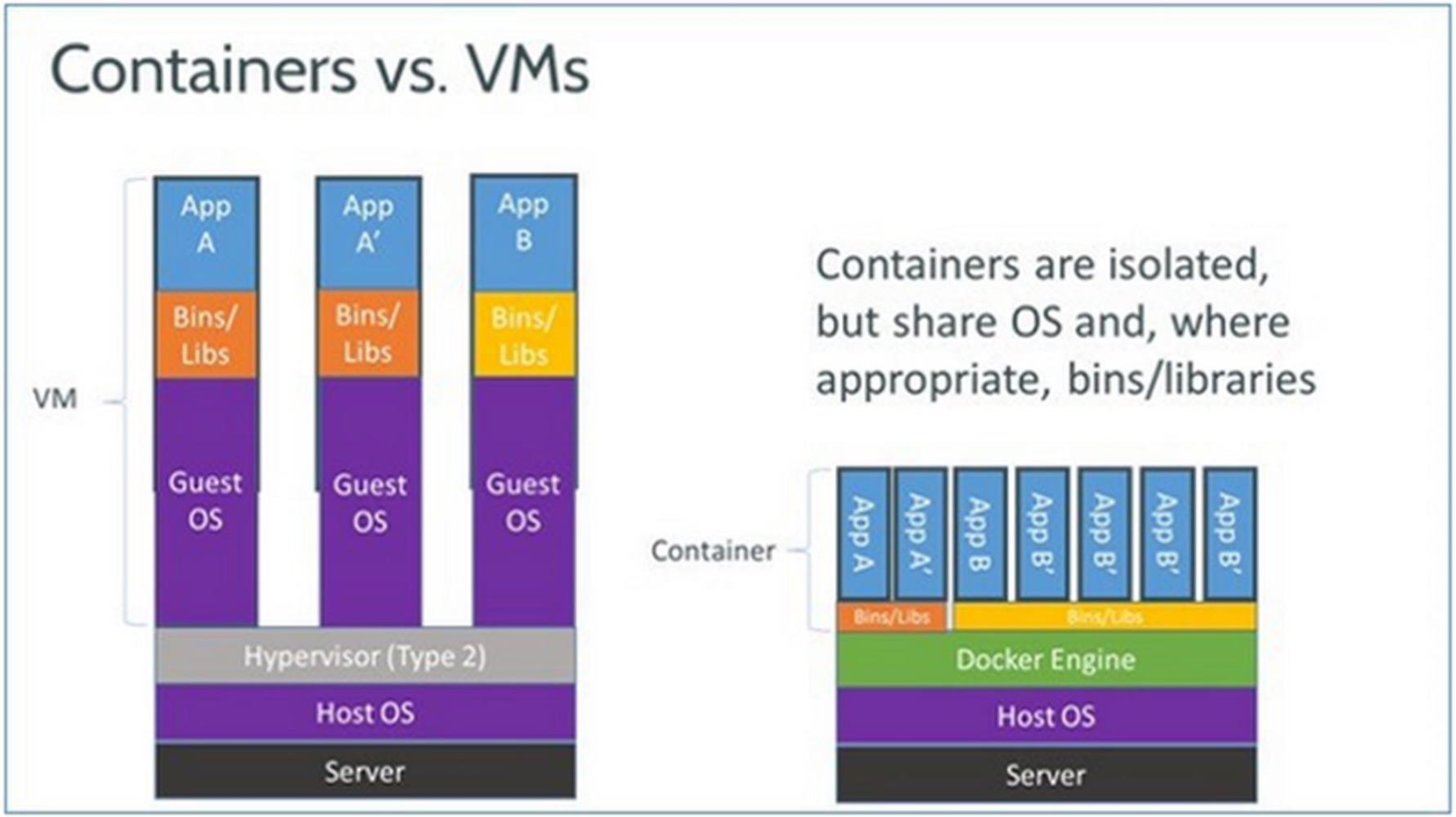


Introduction to Docker

Docker Container
(fast start/process/lightweight)



Introduction to Docker (Layering Concept)



Install Docker for Windows

- https://docs.docker.com/docker-for-windows/install/#install-docker-for-windows-desktop-app

- (setup and build a sample application) https://docs.docker.com/get-started/

Pull Image and Start Container

```
Burts-MacBook-Pro:~ burtwalsh$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:f5233545e43561214ca4891fd1157e1c3c563316ed8e237750d59bde73361e77
Status: Image is up to date for hello-world:latest
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

Burts-MacBook-Pro:~ burtwalsh$
```

Review of Container Start

- Pulled Image from dockerhub.com to local machine repository
- Docker engine started a container in which the image was run
- Docker container is not an OS but rather a process so it is lightweight (in resources) and quick to start
- You can run many containers on a machine

Running Another Image

- There are a number of prebuilt images on dockerhub.com
- Many of these images run open source software
- Next we will run an instance of the open source database MySQL (https://docs.docker.com/samples/library/MySQL/)

Starting and Working in a Container

- Docker run will load the image to local repository if necessary
- We indicate the version of MySQL to use (5.6)
- The MySQL process is running in the container and not visible outside the container

Running Another Image
Look Ma… “No MySQL” !

```
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=notforyou -d mysql:5.6
b06b4a1eb7e97fdaade76a85aed035ad577cf3c8448b71000fa20f83ee086a98
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED                  STATUS              PORTS               NAMES
b06b4a1eb7e9        mysql:5.6           "docker-entrypoint.s…"   Less than a second ago   Up 2 seconds        3306/tcp            some-mysql
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ ps -ef | grep mysql
  501  9413  9388   0  2:59PM ttys001    0:00.00 grep mysql
Burts-MacBook-Pro:~ burtwalsh$
```

Starting and Working in a Container (continued)

- We can log onto the container (start a bash shell) with the docker exec command
- Running ps (to see running processes) shows that the processes for MySQL are not visible outside the container
- We can log onto MySQL with the password passed in when starting the container (notforyou)

Running Another Image

```
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
b06b4a1eb7e9        mysql:5.6           "docker-entrypoint.s…"   About a minute ago  Up About a minute   3306/tcp            some-mysql
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker exec -it b06b4a1eb7e9 bash
root@b06b4a1eb7e9:/#
root@b06b4a1eb7e9:/# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.40 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Local Image Store and Layers

```
Burts-MacBook-Pro:~ burtwalsh$ docker images | grep mysql
linkedin/wherehows-mysql       1                   02311a88c836        3 weeks ago         256MB
linkedin/wherehows-mysql       latest              02311a88c836        3 weeks ago         256MB
mysql                          5.6                 5f5ccdc8aedc        4 weeks ago         256MB
mysql                          latest              a8a59477268d        4 weeks ago         445MB
mysql                          5.7.21              5195076672a7        2 months ago        371MB
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$
Burts-MacBook-Pro:~ burtwalsh$ docker history 5f5ccdc8aedc
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
5f5ccdc8aedc        4 weeks ago         /bin/sh -c #(nop)  CMD ["mysqld"]               0B
<missing>           4 weeks ago         /bin/sh -c #(nop)  EXPOSE 3306/tcp              0B
<missing>           4 weeks ago         /bin/sh -c #(nop)  ENTRYPOINT ["docker-entry…   0B
<missing>           4 weeks ago         /bin/sh -c ln -s usr/local/bin/docker-entryp…   34B
<missing>           4 weeks ago         /bin/sh -c #(nop) COPY file:36275029ba18b014…   6.26kB
<missing>           5 weeks ago         /bin/sh -c #(nop)  VOLUME [/var/lib/mysql]      0B
<missing>           5 weeks ago         /bin/sh -c {   echo mysql-community-server m…   146MB
<missing>           5 weeks ago         /bin/sh -c echo "deb http://repo.mysql.com/a…   56B
<missing>           5 weeks ago         /bin/sh -c #(nop)  ENV MYSQL_VERSION=5.6.40-…   0B
<missing>           5 weeks ago         /bin/sh -c #(nop)  ENV MYSQL_MAJOR=5.6          0B
<missing>           5 weeks ago         /bin/sh -c set -ex;  key='A4A9406876FCBD3C45…   22.1kB
<missing>           5 weeks ago         /bin/sh -c apt-get update && apt-get install…   39.9MB
<missing>           5 weeks ago         /bin/sh -c mkdir /docker-entrypoint-initdb.d    0B
<missing>           5 weeks ago         /bin/sh -c set -x  && apt-get update && apt-…   4.44MB
<missing>           5 weeks ago         /bin/sh -c #(nop)  ENV GOSU_VERSION=1.7         0B
<missing>           5 weeks ago         /bin/sh -c apt-get update && apt-get install…   10.2MB
<missing>           5 weeks ago         /bin/sh -c groupadd -r mysql && useradd -r -…   329kB
<missing>           5 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>           5 weeks ago         /bin/sh -c #(nop) ADD file:ec5be7eec56a74975…   55.3MB
Burts-MacBook-Pro:~ burtwalsh$
```

Looking at How the Image is Built

GitHub, Inc. [US] | https://github.com/docker-library/mysql

Microsoft is acquiring GitHub! **Read our blog** and **Satya Nadella's post** to learn more.

Search or jump to... / Pull requests Issues Marketplace Explore

docker-library / mysql

Watch 82 | Star 865 | Fork 962

Code | Issues 18 | Pull requests 8 | Projects 0 | Insights

Docker Official Image packaging for MySQL Community Server https://dev.mysql.com/

220 commits | 1 branch | 0 releases | 19 contributors | GPL-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

tianon Merge pull request #402 from wpalmer/process_init_file-function ... Latest commit fc3e856 on Apr 30

5.5	Merge pull request #402 from wpalmer/process_init_file-function	a month ago
5.6	Merge pull request #402 from wpalmer/process_init_file-function	a month ago
5.7	Merge pull request #402 from wpalmer/process_init_file-function	a month ago
8.0	Merge pull request #402 from wpalmer/process_init_file-function	a month ago
.travis.yml	Add "travis_retry" to our build to reduce transitory failures	a year ago
LICENSE	Add GPL-2+ license	3 years ago
README.md	Update generated README	2 months ago
generate-stackbrew-library.sh	Move "latest" to 8.0 now that it's GA	2 months ago
update.sh	Update 5.7, 5.6, and 5.5 to "debian:stretch-slim"	3 months ago

README.md

Images are Built from a Dockerfile

- A Docker file contain a FROM clause indicating the type of OS to build upon
- This could include scratch which means just use the underlying OS
- They have RUN commands for commands to execute on this OS “Layer”

Images are Built from a Dockerfile (continued)

- You can mount Volumes, COPY files, EXPOSE ports and execute commands
- Each Instruction is a “Layer”
- The ENTRYPOINT list the script with commands for container startup (container main process)

Getting Source Dockerfile

```
Burts-MBP:tmp burtwalsh$
Burts-MBP:tmp burtwalsh$
Burts-MBP:tmp burtwalsh$ cd /tmp
Burts-MBP:tmp burtwalsh$
Burts-MBP:tmp burtwalsh$ git clone https://github.com/docker-library/mysql
Cloning into 'mysql'...
remote: Counting objects: 949, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 949 (delta 1), reused 1 (delta 0), pack-reused 946
Receiving objects: 100% (949/949), 176.25 KiB | 784.00 KiB/s, done.
Resolving deltas: 100% (410/410), done.
Burts-MBP:tmp burtwalsh$
Burts-MBP:tmp burtwalsh$
Burts-MBP:tmp burtwalsh$ cd mysql/5.6
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
```

Inspecting the Dockerfile

```
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$ head -20 Dockerfile
FROM debian:stretch-slim

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependencies get added
RUN groupadd -r mysql && useradd -r -g mysql mysql

RUN apt-get update && apt-get install -y --no-install-recommends gnupg dirmngr && rm -rf /var/lib/apt/lists/*

# add gosu for easy step-down from root
ENV GOSU_VERSION 1.7
RUN set -x \
        && apt-get update && apt-get install -y --no-install-recommends ca-certificates wget && rm -rf /var/lib/apt/lists/* \
        && wget -O /usr/local/bin/gosu "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-architecture)" \
        && wget -O /usr/local/bin/gosu.asc "https://github.com/tianon/gosu/releases/download/$GOSU_VERSION/gosu-$(dpkg --print-architecture).asc" \
        && export GNUPGHOME="$(mktemp -d)" \
        && gpg --keyserver ha.pool.sks-keyservers.net --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4 \
        && gpg --batch --verify /usr/local/bin/gosu.asc /usr/local/bin/gosu \
        && rm -rf "$GNUPGHOME" /usr/local/bin/gosu.asc \
        && chmod +x /usr/local/bin/gosu \
        && gosu nobody true \
        && apt-get purge -y --auto-remove ca-certificates wget
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$ tail Dockerfile
        && echo '[mysqld]\nskip-host-cache\nskip-name-resolve' > /etc/mysql/conf.d/docker.cnf

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /usr/local/bin/
RUN ln -s usr/local/bin/docker-entrypoint.sh /entrypoint.sh # backwards compat
ENTRYPOINT ["docker-entrypoint.sh"]

EXPOSE 3306
CMD ["mysqld"]
Burts-MBP:5.6 burtwalsh$
```

Building the Image from the Dockerfile

```
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ pwd
/tmp/mysql/5.6
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ ls
Dockerfile              docker-entrypoint.sh
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ docker build . -t burtmysql
Sending build context to Docker daemon   12.8kB
Step 1/18 : FROM debian:stretch-slim
stretch-slim: Pulling from library/debian
f2aa67a397c4: Already exists
Digest: sha256:aa33563fd6dc2b14e8cae20cc74f4581d4cb89a3952d87b4cab323707040f035
Status: Downloaded newer image for debian:stretch-slim
 ---> 26f0bb790e25
Step 2/18 : RUN groupadd -r mysql && useradd -r -g mysql mysql
 ---> Running in db598fdf4710
Removing intermediate container db598fdf4710
 ---> 342dc030b978
Step 3/18 : RUN apt-get update && apt-get install -y --no-install-recommends gnupg dirmngr && rm -rf /var/lib/apt/lists/*
 ---> Running in 45da874bd3ba
Get:2 http://security.debian.org/debian-security stretch/updates InRelease [94.3 kB]
Ign:1 http://cdn-fastly.deb.debian.org/debian stretch InRelease
Get:3 http://cdn-fastly.deb.debian.org/debian stretch-updates InRelease [91.0 kB]
Get:5 http://security.debian.org/debian-security stretch/updates/main amd64 Packages [442 kB]
Get:4 http://cdn-fastly.deb.debian.org/debian stretch Release [118 kB]
Get:6 http://cdn-fastly.deb.debian.org/debian stretch-updates/main amd64 Packages [12.1 kB]
Get:7 http://cdn-fastly.deb.debian.org/debian stretch Release.gpg [2434 B]
Get:8 http://cdn-fastly.deb.debian.org/debian stretch/main amd64 Packages [9530 kB]
```

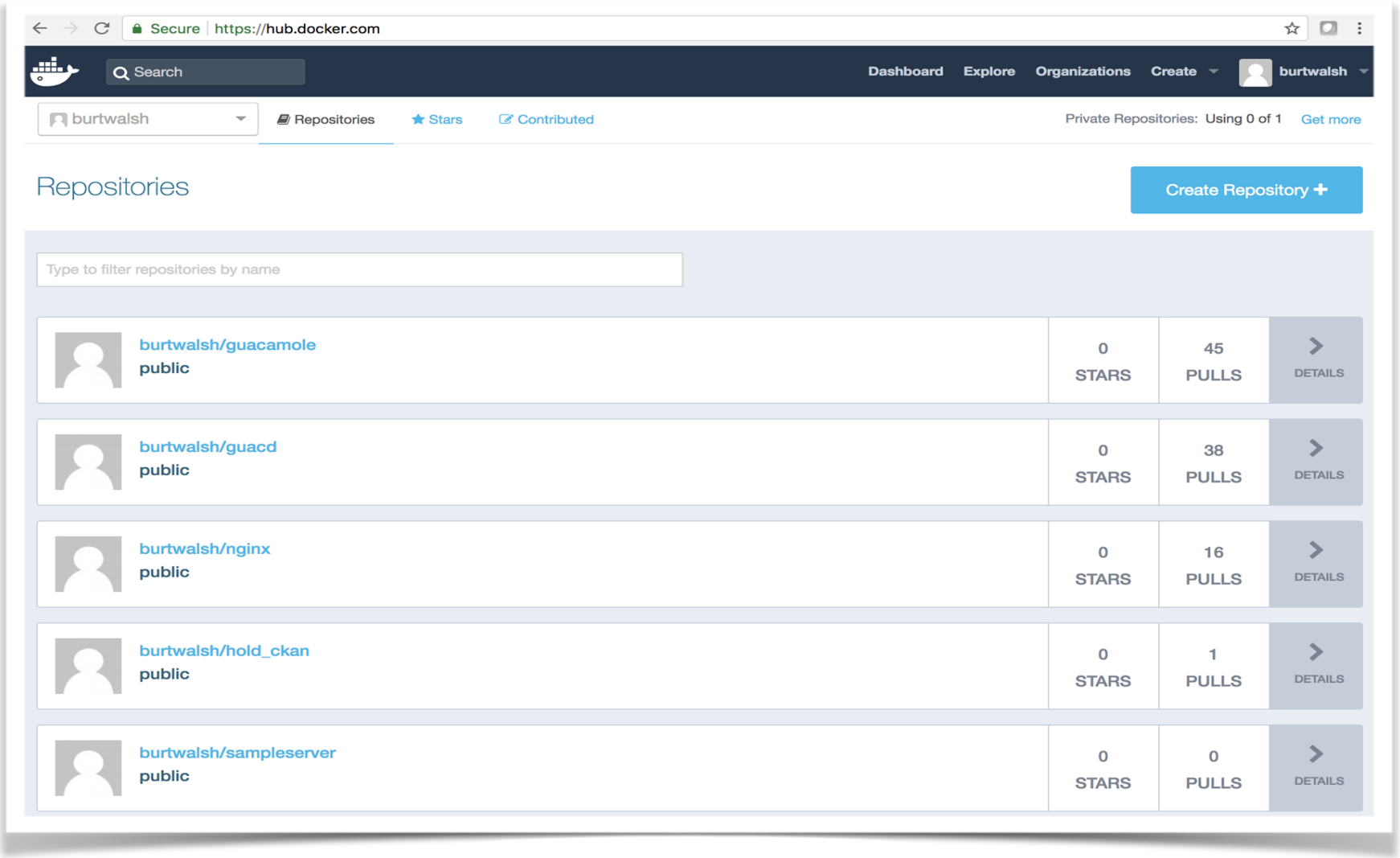
Say We Needed to Update the Image in Development and Then Move it to Test

```
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ docker images | grep burt
ckan_burt                      latest              901ecaa2e0af        2 months ago        1.41GB
burtone                        latest              bf6d29a150d8        2 months ago        1.38GB
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ docker history bf6d29a150d8
IMAGE               CREATED             CREATED BY                                      SIZE                COMMENT
bf6d29a150d8        2 months ago        /bin/bash                                       1.27GB
f975c5035748        3 months ago        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           3 months ago        /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>           3 months ago        /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$…   2.76kB
<missing>           3 months ago        /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>           3 months ago        /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   745B
<missing>           3 months ago        /bin/sh -c #(nop) ADD file:c753df38640ab6e24…   112MB
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ docker save bf6d29a150d8 -o burt_fixed_mysql.tar
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ ls burt_fixed_mysql.tar
burt_fixed_mysql.tar
Burts-MacBook-Pro:5.6 burtwalsh$ ls -al burt_fixed_mysql.tar
-rw-------  1 burtwalsh  wheel  1413384704 Jun  5 00:03 burt_fixed_mysql.tar
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ MOVE THE FILE  then "docker load..."
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$
Burts-MacBook-Pro:5.6 burtwalsh$ STOP THE RUNNING CONTAINERS AND START THE NEW LOADED ONE
```

Pushing the Image to Dockerhub

```
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username (burtwalsh):
Password:
Login Succeeded
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$ docker tag 1aefc93ffa3 burtwalsh/sampleserver:latest
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$ docker push burtwalsh/sampleserver:latest
The push refers to repository [docker.io/burtwalsh/sampleserver]
102c01a40de6: Layer already exists
629b40dfa782: Layer already exists
e4c5f48eee96: Layer already exists
90c102120d07: Layer already exists
11881e13963c: Layer already exists
8fe4f8cdea7f: Layer already exists
059ad60bcacf: Layer already exists
8db5f072feec: Layer already exists
67885e448177: Layer already exists
ec75999a0cb1: Layer already exists
65bdd50ee76a: Layer already exists
latest: digest: sha256:4bda61cdb11ace7bc008bca37dad4d3ff0bc1a067f375140e1592969651d16f6 size: 2611
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
Burts-MBP:5.6 burtwalsh$
```

Image in Dockerhub



Review

- Docker container is a (has a) main running process not a full Operating System (OS)
- Docker container is not an OS so it is lightweight (in resources) and quick to start
- It is easy to create an image from a container and save it to a file and move it from one system to another

Review

- You can have many containers running on a machine each with different library (shared object/DLL) configurations
- The instructions to build an image are put in a Dockerfile

Building Multi-Tier System

Goals

- Create an image with a webserver
- Run the image in a container on the local machine
- Docker-compose a system of containers
- Create a pool of webservers behind a load balancer
- Create and leverage a microservice in a container

Webserver Code (Run in Nodejs)

```
'use strict';

const express = require('express');

const PORT = 3000;
let HOST = "not known";

//require means load or create an object in this case
//get the dns object and uses its lookup method to lookup the ip address
//of our host.  We get the hostname using the os object and calling the
//hostname() method on this object
//
//nodejs uses only one thread so we need to use callbacks which get executed
//by the principal and only thread when the results -- usually from async i/o
//get returned

//In ENGLISH this line says look up the ip address of our hostname and when you
//get it invoke our CALLBACK
require('dns').lookup(require('os').hostname(),

  function mycallback(err, host_ip_address, fam) {

     HOST = host_ip_address;
     const app = express();
     app.get('/', (req, res) => {
       res.send('Hello world from ' + `Running on http://${HOST}:${PORT}`);
     });

     app.listen(PORT, HOST);
})
```

Dockerfile

- We need to create an image with a base OS
- We need to add libraries for Nodejs, express
- We need to copy our web server code to the image
- We need to run the server and expose the server port (3000)

Example Dockerfile

```
FROM ubuntu:latest

#install needed software on the Docker container
RUN apt-get update && apt-get install nodejs vim net-tools npm -y

#create the web application directory on the Docker container
RUN mkdir -p /mywebapp

WORKDIR /mywebapp

#copy the application files to the Docker container
#into the WORKDIR
# our webserver code server.js needs to have this name
# so that npm start (below) works
COPY server.js /mywebapp

#create the project directory support structure
RUN npm init -f

#install express which is the library that we use to run our webserver
RUN npm install express --save

#make sure our code is executable
RUN chmod -R 755 /mywebapp

#(not needed with new Nodejs version)
#create a symbolic link so nodejs can be executed as node
#RUN ln -s /usr/bin/nodejs /usr/bin/node
#RUN chmod 755 /usr/bin/node

#expose the port that our webserver is running on
EXPOSE 3000

#start the server
CMD ["npm", "start"]
~
```

Github Repository

GitHub, Inc. [US] | https://github.com/ASTApplicationWorkGroup/dockertutorial_part1

Microsoft is acquiring GitHub! **Read our blog** and **Satya Nadella's post** to learn more.

Search or jump to... | Pull requests | Issues | Marketplace | Explore

ASTApplicationWorkGroup / **dockertutorial_part1**

Watch 0 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

No description, website, or topics provided.

Edit

Add topics

13 commits | 2 branches | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

burtwalsh more read me — Latest commit 754256c on Feb 26

nginx	add directory structure	3 months ago
webserver	my change	3 months ago
README.md	more read me	3 months ago
docker-compose.yaml	presentation work	4 months ago
docker.pptx	full pres	3 months ago

README.md

dockertutorial

Sample code in support of the presentation by the AST on docker. This overview demonstrates what Docker is and why it is important. We look at various docker commands such as the following:

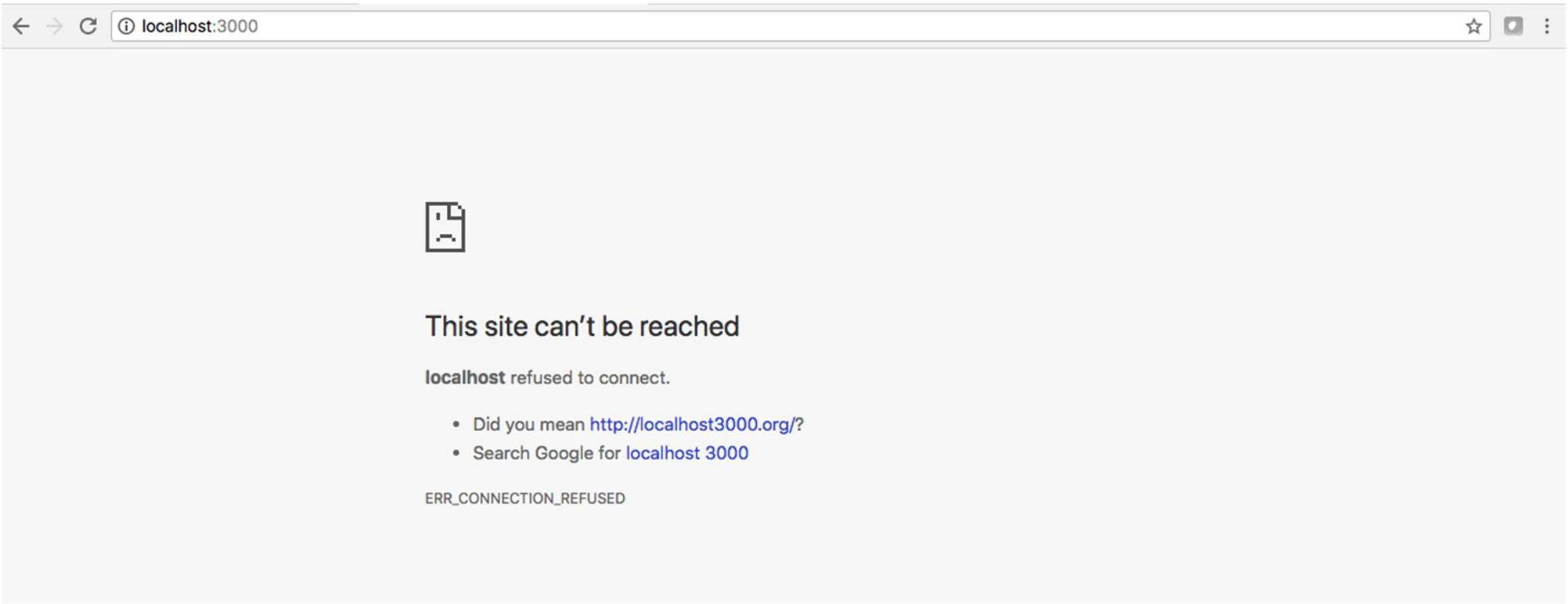
Building the Image

```
Burts-MacBook-Pro:tmp burtwalsh$
Burts-MacBook-Pro:tmp burtwalsh$ git clone https://github.com/ASTApplicationWorkGroup/dockertutorial_part1
Cloning into 'dockertutorial_part1'...
remote: Counting objects: 79, done.
remote: Total 79 (delta 0), reused 0 (delta 0), pack-reused 79
Unpacking objects: 100% (79/79), done.
Burts-MacBook-Pro:tmp burtwalsh$ cd dockertutorial_part1/
Burts-MacBook-Pro:dockertutorial_part1 burtwalsh$
Burts-MacBook-Pro:dockertutorial_part1 burtwalsh$ git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.
Burts-MacBook-Pro:dockertutorial_part1 burtwalsh$
Burts-MacBook-Pro:dockertutorial_part1 burtwalsh$
Burts-MacBook-Pro:dockertutorial_part1 burtwalsh$ cd webserver/
Burts-MacBook-Pro:webserver burtwalsh$
Burts-MacBook-Pro:webserver burtwalsh$ ls
Dockerfile      package.json    server.js
Burts-MacBook-Pro:webserver burtwalsh$
Burts-MacBook-Pro:webserver burtwalsh$
Burts-MacBook-Pro:webserver burtwalsh$ docker build . -t samplewebserver
Sending build context to Docker daemon   5.12kB
Step 1/12 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
a48c500ed24e: Downloading [======>                                          ]  3.751MB/30.96MB
1e1de00ff7e1: Download complete
0330ca45a200: Download complete
471db38bcfbf: Download complete
0b4aba487617: Download complete
```

Starting the Image

```
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker images | grep web
samplewebserver                 latest              1aefc93ffa35        28 minutes ago      482MB
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker run -t -d 1aefc93ffa35

3c1bbde5ef90309fbf5ef5fec04719e54da0896a15153d49946f46a2b5103fe2
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED                  STATUS              PORTS
3c1bbde5ef90        1aefc93ffa35        "npm start"         Less than a second ago   Up 1 second         3000/tcp
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker exec -i -t 3c1bbde5ef90  bash
root@3c1bbde5ef90:/mywebapp#
root@3c1bbde5ef90:/mywebapp#
root@3c1bbde5ef90:/mywebapp#
root@3c1bbde5ef90:/mywebapp#
root@3c1bbde5ef90:/mywebapp# netstat -a | grep 3000
tcp        0      0 3c1bbde5ef90:3000       0.0.0.0:*               LISTEN
root@3c1bbde5ef90:/mywebapp#
```

The Site is not Accessible



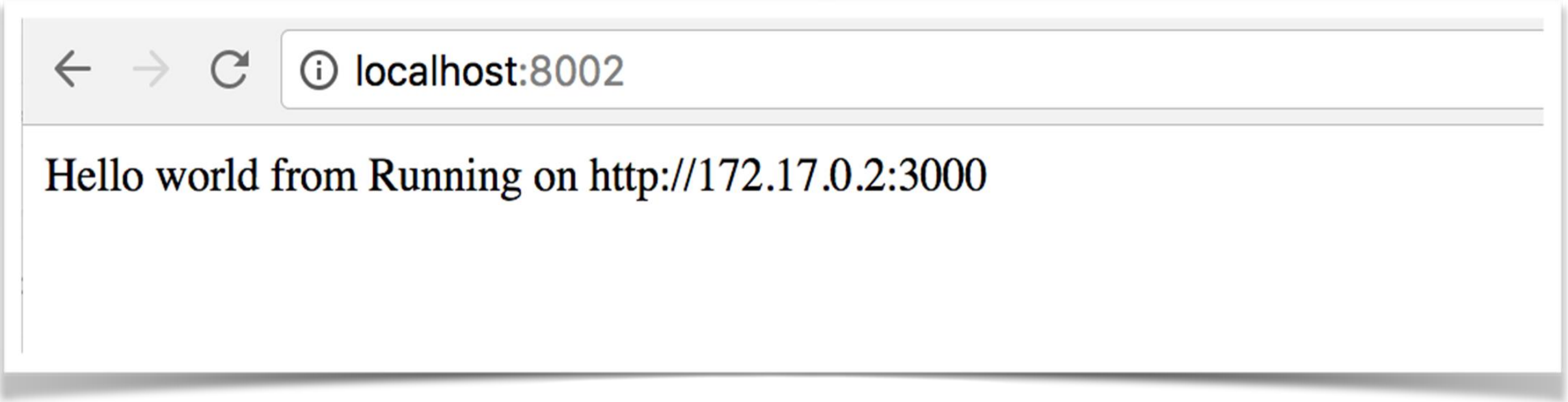
Inspecting the Container

```
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
4b417f6de9a0        1aefc93ffa35        "npm start"         8 minutes ago       Up 8 minutes        3000/tcp            sleepy_curran
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker inspect --format='{{json .NetworkSettings.Networks}}' 4b417f6de9a0 | jq
{
  "bridge": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "NetworkID": "e7212f8656bb3df4557d6cf822caaadb105cf470bc61ef4bdefda4368aa40d35",
    "EndpointID": "09c79443721f15277c9e06e2136e6ac83cab3f4cd60f091288b83709bea44718",
    "Gateway": "172.17.0.1",
    "IPAddress": "172.17.0.2",
    "IPPrefixLen": 16,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "MacAddress": "02:42:ac:11:00:02",
    "DriverOpts": null
  }
}
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker network ls
NETWORK ID          NAME                      DRIVER              SCOPE
e7212f8656bb        bridge                    bridge              local
8a1cf2782557        docker_default            bridge              local
e40701549344        docker_gwbridge           bridge              local
a7280490b66c        host                      host                local
f52dfce65739        none                      null                local
6e35b51dadc0        wherehows-docker_default  bridge              local
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
```

Running with Network Address Translation (NAT) “Bridge Network”

```
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker rm -f 4b417f6de9a0
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker run -t -d -p 8002:3000 samplewebserver
7aa96fb9e397b2b48bb231373b6635f612f05ddc4bbbc69356dccd7bf72ef9f6
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED                  STATUS              PORTS                     NAMES
7aa96fb9e397        samplewebserver     "npm start"         Less than a second ago   Up 3 seconds        0.0.0.0:8002->3000/tcp    admiring_borg
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
Burts-MBP:~ burtwalsh$
```

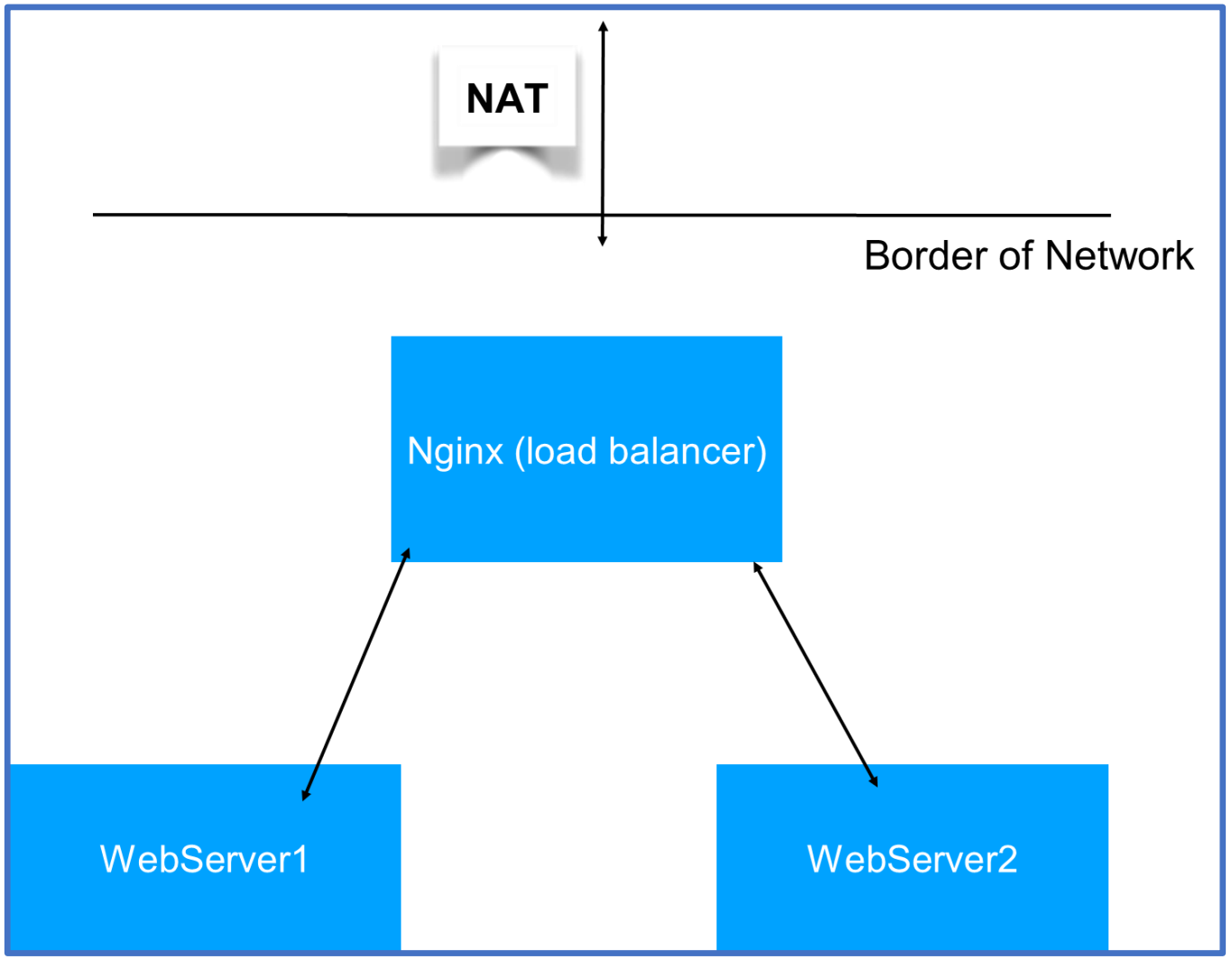
NAT in Action



Docker Compose

- Docker compose lets you define a group of containers and relationships between them
- You can bring up the group, restart the group, stop the group as a unit
- This is defined in a simple YAML file

Architecture



Nginx

Will be a container like our webservers

- Acts as a load balancer (reverse proxy)
- Will shift requests between our two web servers

Directory for Nginx Resources

```
ASTMACs-MacBook-Air:docker_file astmac$ ls
Dockerfile              docker-compose.yaml     nginx                   package.json            server.js               service.sh
ASTMACs-MacBook-Air:docker_file astmac$ vi nginx/nginx.conf
ASTMACs-MacBook-Air:docker_file astmac$ pwd
/Users/astmac/ok/docker_file
ASTMACs-MacBook-Air:docker_file astmac$ find nginx/
nginx/
nginx//Dockerfile
nginx//nginx.conf
```

Nginx Configuration

```
worker_processes auto;

events { worker_connections 1024; }

http {

        upstream node-app {
              least_conn;
              server webserver1:3000 weight=10 max_fails=3 fail_timeout=30s;
              server webserver2:3000 weight=10 max_fails=3 fail_timeout=30s;
        }

        server {
              listen 80;

              location / {
                proxy_pass http://node-app;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
              }
        }
}
```

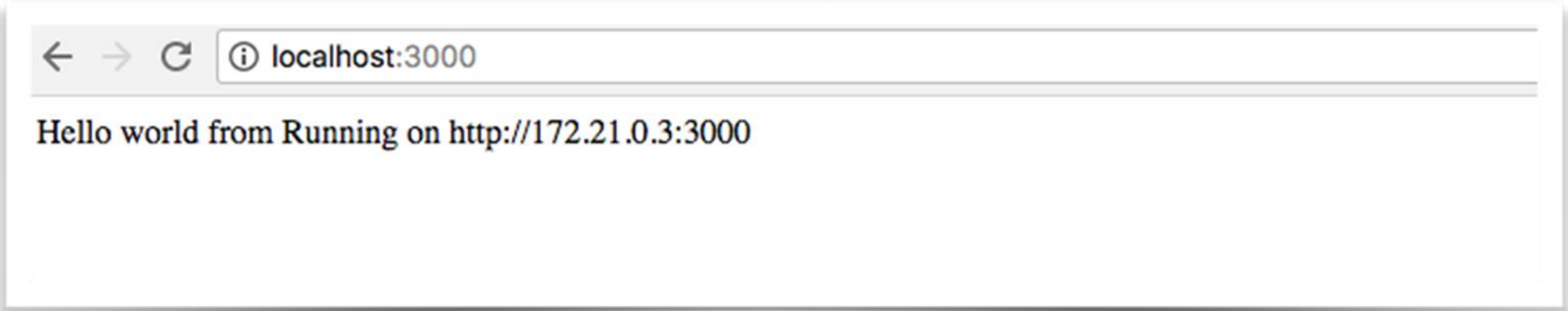
Docker Compose File

```
ASTMACs-MacBook-Air:docker_file astmac$ cat docker-compose.yaml
version: '2'
services:
  nginx:
     build: ./nginx
     links:
        - webserver1
        - webserver2
     ports:
        - "3000:80"
     networks:
        - mynetwork
  webserver1:
     image: burtwalsh/my_node_web_server:latest
     ports:
        - "3000"
     volumes:
        - holdwebdata:/saveddata
     networks:
        - mynetwork
  webserver2:
     image: burtwalsh/my_node_web_server:latest
     ports:
        - "3000"
     volumes:
        - holdwebdata:/saveddata
     networks:
        - mynetwork
networks:
  mynetwork:
    driver: "bridge"
volumes:
  holdwebdata:
    driver: "local"
ASTMACs-MacBook-Air:docker_file astmac$ docker-compose up -d
dockerfile_webserver2_1 is up-to-date
dockerfile_webserver1_1 is up-to-date
dockerfile_nginx_1 is up-to-date
```

Created Network

```
ASTMACs-MacBook-Air:docker_file astmac$ docker network ls
NETWORK ID          NAME                    DRIVER              SCOPE
d5dde7091e89        bridge                  bridge              local
b98773262161        dockerfile_default      bridge              local
9bb54f8b131b        dockerfile_mynetwork    bridge              local
21e09b1b7c18        host                    host                local
19723a90d6ef        none                    null                local
dcc231e51d40        rubix.local             bridge              local
cad7d6b71bea        tykquickstart_gateway   bridge              local
```

Alternation between Webservers



Containers

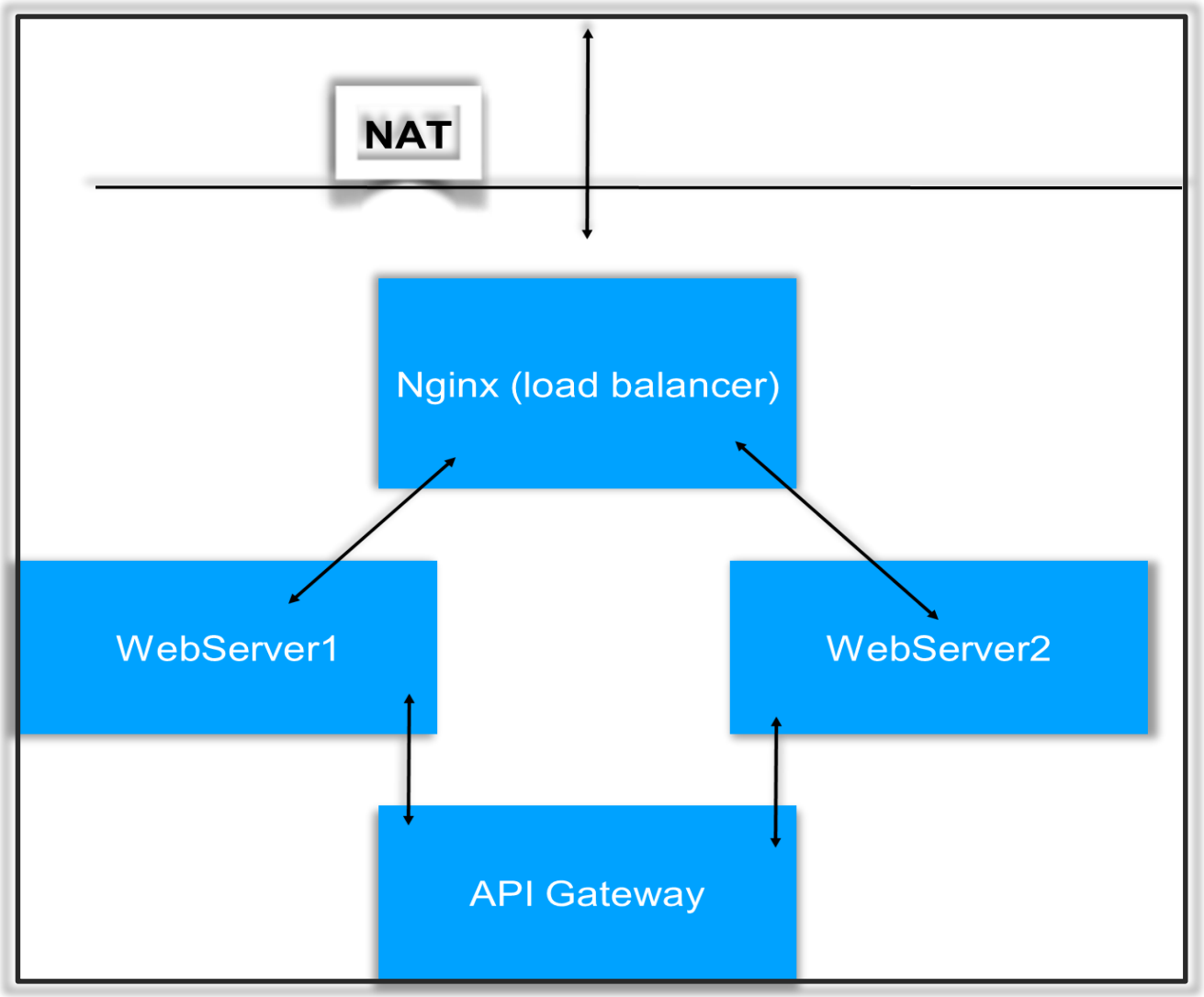
```
ASTMACs-MacBook-Air:docker_file astmac$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS
af7402a8e665        dockerfile_nginx                     "nginx -g 'daemon of…"   3 minutes ago       Up 3 minutes        0.0.0.0:3000->80/tcp
d63099162436        burtwalsh/my_node_web_server:latest  "npm start"              6 minutes ago       Up 6 minutes        0.0.0.0:32771->3000/tcp
081372bd210b        burtwalsh/my_node_web_server:latest  "npm start"              6 minutes ago       Up 6 minutes        0.0.0.0:32770->3000/tcp
```

Microservices

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

https://martinfowler.com/articles/microservices.html

New Architecture



Changes to docker-compose.yaml

```
  webserver2:
     build: ./webserver
     environment:
        - BUSINESS_LAYER_IP=businesslayer
        - BUSINESS_LAYER_PORT=8012
     links:
        - businesslayer
     ports:
        - "3000"
     volumes:
        - holdwebdata:/saveddata
     networks:
        - mynetwork
  businesslayer:
     build: ./microservices
     stdin_open: true
     ports:
        - "8012"
     networks:
        - mynetwork
networks:
  mynetwork:
    driver: "bridge"
volumes:
  holdwebdata:
    driver: "local"
```

Service Definition

```
let { ServiceBroker } = require('moleculer');
let ApiService = require('moleculer-web');

var translate = require('yandex-translate')('trnsl.1.1.20180224T101627Z.60834ac718491fb6.1d4eeed31e7a59f39e0bda72271a8026b29dcd8a');

let broker = new ServiceBroker({logger:console});

function process(myString, callback) {
   translate.translate(myString, { to: 'ru' }, callback);
}

//TEST CODE process('burt', function(e,r) { console.log(r.text); });

broker.createService({
   name: "stringFun",

   //this exposes our service on port 8012 of the ApiGateway
   //the ApiGateway is a gatekeeper to our api
   mixins: [ApiService],
   settings: { port: 8012 },
   actions: {

      translate(ctx) {
        return new Promise(function(resolve,reject) {
             process(ctx.params.p, function(err, result) {
                 resolve('Your string (' + ctx.params.p + ') in Russian is (' + result.text + ')');
             });
        });
      }
   }
});

//start the server
broker.start();
```

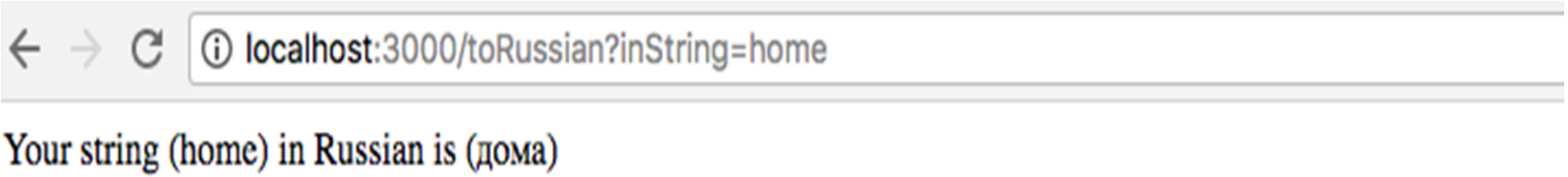
Changes to docker-compose.yaml

```
'use strict';

const express = require('express');
const request = require('request');

const PORT = 3000;
let HOST = "not known";
const BUSINESS_LAYER_IP = process.env.BUSINESS_LAYER_IP;
const BUSINESS_LAYER_PORT = process.env.BUSINESS_LAYER_PORT;

require('dns').lookup(require('os').hostname(),

  function mycallback(err, host_ip_address, fam) {

     HOST = host_ip_address;
     const app = express();
     app.get('/', (req, res) => {
       res.send('Hello world from ' + `Running on http://${HOST}:${PORT}`);
     });

     //Translate passed in string
     app.get('/toRussian', (req, res) => {

       let value = req.query.inString;
       let response = res;

       //call our microservice through the API Gateway
       request.get(`http://${BUSINESS_LAYER_IP}:${BUSINESS_LAYER_PORT}/stringFun/translate?p=` + value, function (err, res, body) {
           if (!err) {
             var resultsObj = JSON.parse(body);
             response.send(resultsObj);
           }
           else {
             response.send('and error occurred: ' + JSON.stringify(err));
           }
      });
     });

     app.listen(PORT, HOST);
})
```

Running the New Solution

```
ASTMACs-MacBook-Air:dockertutorial_part1 astmac$ docker-compose -f docker-compose.yaml -p kathy up
ASTMACs-MacBook-Air:dockertutorial_part1 astmac$
ASTMACs-MacBook-Air:dockertutorial_part1 astmac$ docker container ls
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                     NAMES
cfbfd6d2a393        kathylincliam_nginx            "nginx -g 'daemon of…"   13 seconds ago      Up 11 seconds       0.0.0.0:3000->80/tcp      kathylincliam_nginx_1
5d55edb3e878        kathylincliam_webserver2       "npm start"              14 seconds ago      Up 12 seconds       0.0.0.0:32857->3000/tcp   kathylincliam_webserver2_1
76d78adb597c        kathylincliam_webserver1       "npm start"              14 seconds ago      Up 12 seconds       0.0.0.0:32856->3000/tcp   kathylincliam_webserver1_1
6b6cacdc2b0e        kathylincliam_businesslayer    "/apigateway/ok.sh"      15 seconds ago      Up 14 seconds       0.0.0.0:32855->8012/tcp   kathylincliam_businesslayer_
1
ASTMACs-MacBook-Air:dockertutorial_part1 astmac$
```

Calling the Business Layer through the Webserver



References

- Three tier web application with docker compose (https://serversforhackers.com/dockerized-app/docker-compose)
- Great tutorial for around $10 (https://www.udemy.com/docker-mastery/)
- Main docker site (https://www.docker.com/)
- Docker hub for publishing and pulling (https://hub.docker.com/)

Other Things to Look At

- Docker Swarm (https://docs.docker.com/engine/swarm/)
- Nodejs (https://nodejs.org/en/)
- Source code and slides (https://github.com/burtwalsh/dockertutorial)
- https://moleculer.services/
- https://apiumbrella.io/ (API Gateway)

Questions?

Burt Walsh
burt.walsh@ast.myflorida.com

Michael Avello
michael.avello@ast.myflorida.com

Agency For State Technology
4050 Esplanade Way, Suite 115
Tallahassee, Florida 32399