



RED HAT® JBOSS®
MIDDLEWARE

Package your Java EE applications using Docker and Kubernetes

Arun Gupta, @arungupta
Red Hat



Arun Gupta

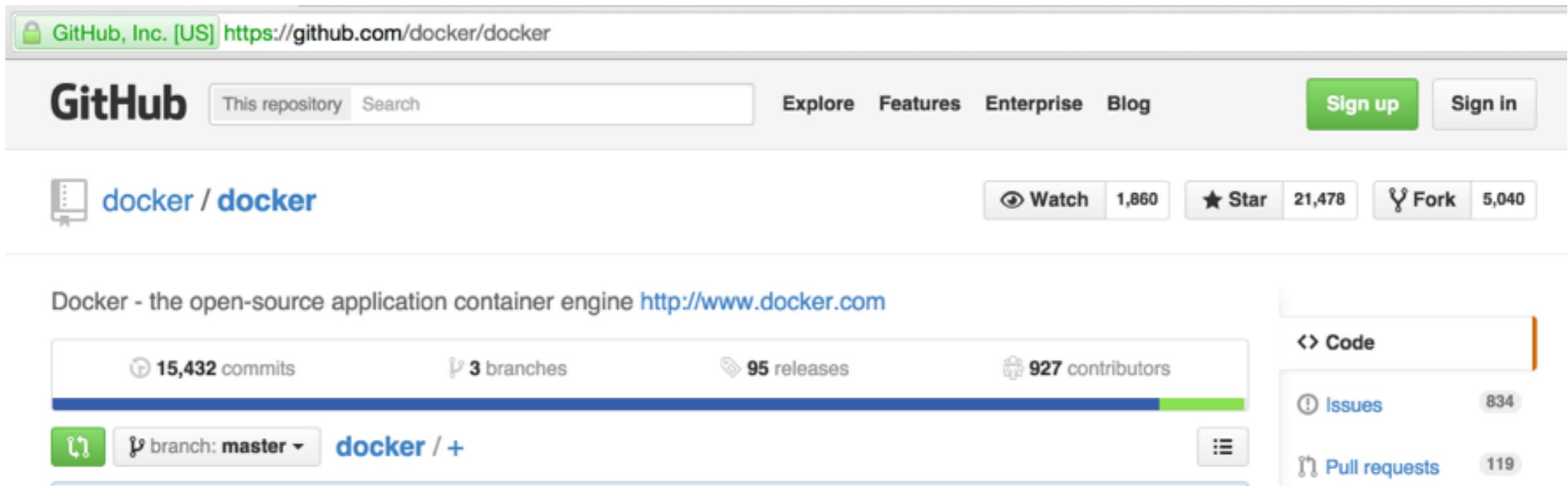
Director, Technical Marketing &
Developer Advocacy

[@arungupta](https://twitter.com/arungupta)
blog.arungupta.me
arungupta@redhat.com



What is Docker?

- Open source project and company



A screenshot of the GitHub repository page for "docker / docker". The page shows the repository's details: 15,432 commits, 3 branches, 95 releases, and 927 contributors. It also displays the master branch, issues (834), and pull requests (119). The Docker logo is visible in the top right corner of the page header.

GitHub, Inc. [US] <https://github.com/docker/docker>

GitHub This repository Search Explore Features Enterprise Blog Sign up Sign in

docker / docker Watch 1,860 Star 21,478 Fork 5,040

Docker - the open-source application container engine <http://www.docker.com>

15,432 commits 3 branches 95 releases 927 contributors

branch: master docker / +

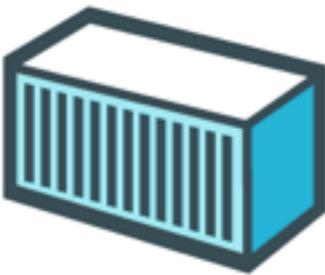
Issues 834 Pull requests 119

- Used to create containers for software applications
- Package Once Deploy Anywhere (PODA)



Build

Develop an app using Docker containers with
any language and any toolchain.



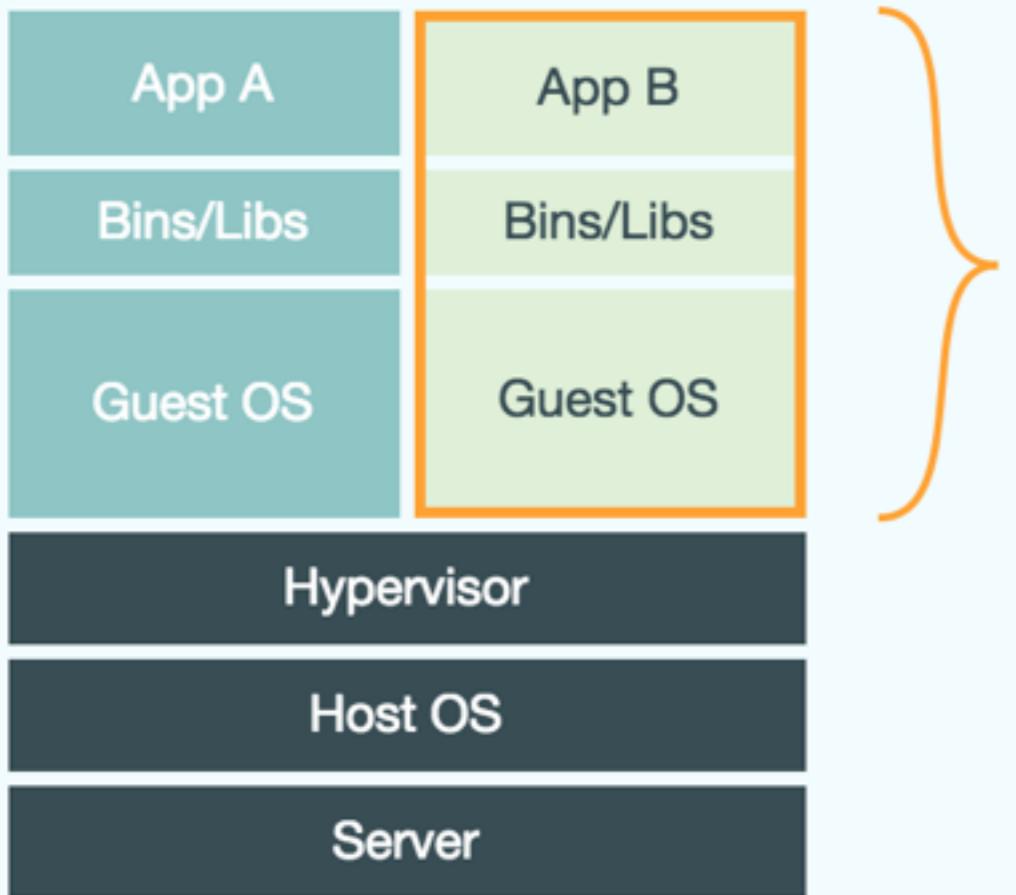
Ship

Ship the “Dockerized” app and dependencies
anywhere - to QA, teammates, or the cloud -
without breaking anything.



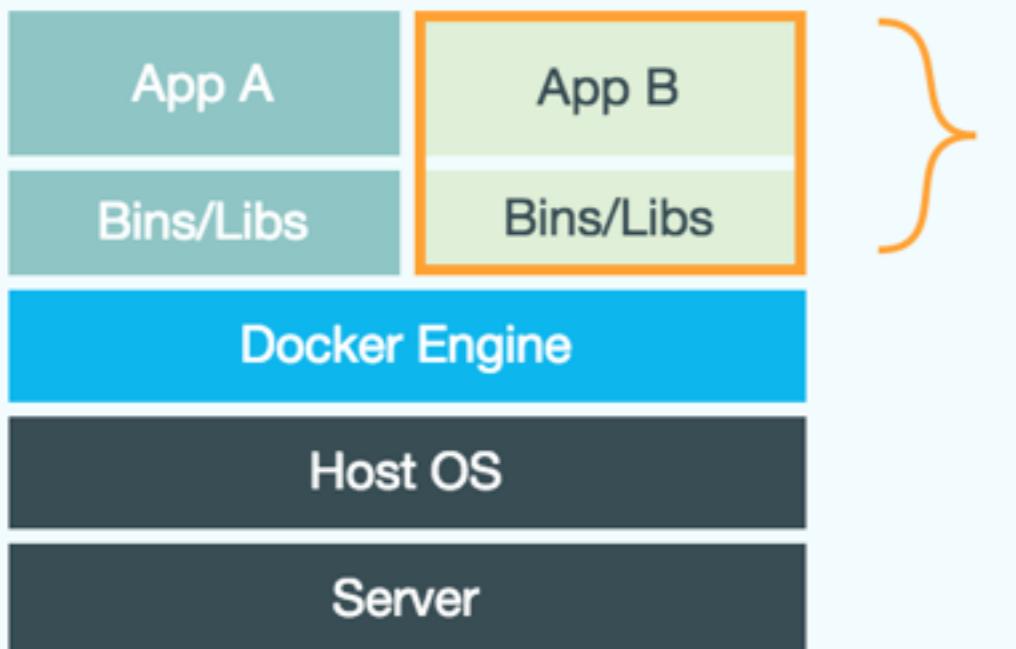
Run

Scale to 1000s of nodes, move between data
centers and clouds, update with zero
downtime and more.



Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.

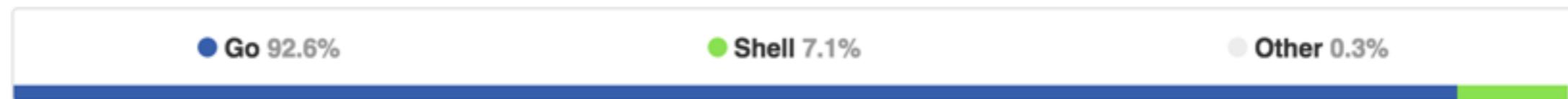


Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

Underlying Technology

- Written in Go



- Uses several Linux features
 - **Namespaces** to provide isolation
 - **Control groups** to share/limit hardware resources
 - **Union File System** makes it light and fast
 - **libcontainer** defines container format

Is it only Linux?

- Natively supported in Linux
- Can be installed on Mac or Windows using `boot2docker`
 - Tiny Core Linux VM



Software Containers

- Decoupled apps and OS
- Repeatable
- Security sandbox
- Portability - “it works on my machine”
- Isolation
- Snapshotting
- Limit resource usage
- No external dependencies
- Sharing



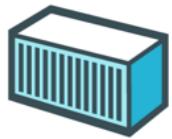
Build

Develop an app using Docker containers with
any language and any toolchain.

- Image defined in text-based **Dockerfile**
- List of commands to build the image

```
FROM fedora:latest
CMD echo "Hello world"
```

- `docker build` or `pull`



- Images shared using registry
- Docker Hub is public SaaS

Ship
Ship the “Dockerized” app and dependencies anywhere - to QA, teammates, or the cloud - without breaking anything.

The screenshot shows the Docker Hub homepage at <https://registry.hub.docker.com>. At the top, there's a navigation bar with links for "What is Docker?", "Use Cases", "Try It!", "Browse", "Install & Docs", "Log In", and "Sign Up". Below the navigation is a search bar with the placeholder "Search the Registry" and a magnifying glass icon. Under the search bar, there's a section titled "Official Repositories" featuring three cards: "redis" with its logo, "ubuntu" with the text "The Official Ubuntu base image", and "WORDPRESS" with its logo and the text "WordPress is a free and open source blogging tool".

- Private registries can be setup inside firewall
- `docker push` or `pull <IMAGE_ID>`



Run

Scale to 1000s of nodes, move between data centers and clouds, update with zero downtime and more.

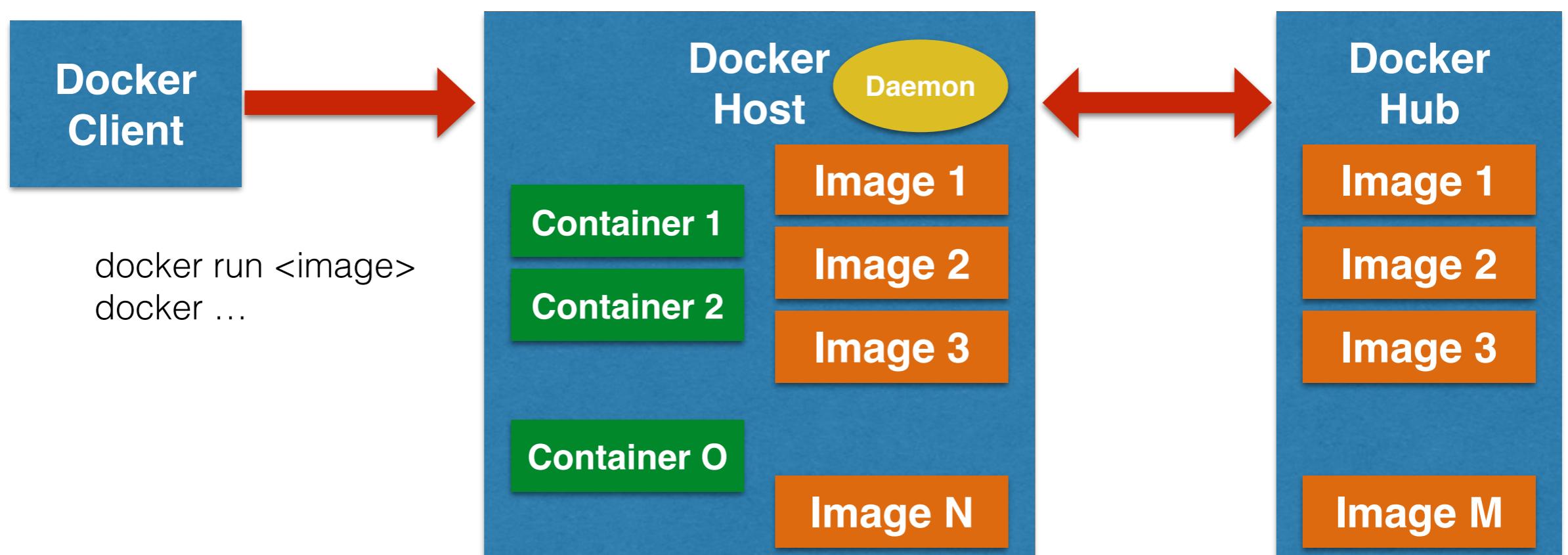


- Container built from the image
- Runtime representation of the image
- Self contained execution environment
- `docker run <IMAGE_ID>`

Docker commands

- **docker ps**: List running containers
- **docker stop**: Stop a running container
- **docker rm**: Remove a running container
- **docker rmi**: Remove an image
- ...

Docker Workflow



```

1 FROM centos
2 MAINTAINER Arun Gupta <arungupta@redhat.com>
3
4 # Execute system update
5 RUN yum -y update && yum clean all
6
7 # Install packages necessary to run EAP
8 RUN yum -y install xmlstarlet saxon augeas bsdtar unzip && yum clean all
9
10 # Create a user and group used to launch processes
11 # The user ID 1000 is the default for the first "regular" user on Fedora/RHEL,
12 # so there is a high chance that this ID will be equal to the current user
13 # making it easier to use volumes (no permission issues)
14 RUN groupadd -r jboss -g 1000 && useradd -u 1000 -r -g jboss -m -d /opt/jboss -s /sbin/nologin -c "JBoss user" jboss
15
16 # Set the working directory to jboss' user home directory
17 WORKDIR /opt/jboss
18
19 # User root user to install software
20 USER root
21
22 # Install necessary packages
23 RUN yum -y install java-1.7.0-openjdk-devel && yum clean all
24 #RUN yum -y install java-1.8.0-openjdk-devel && yum clean all
25
26 # Switch back to jboss user
27 USER jboss
28
29 # Set the JAVA_HOME variable to make it clear where Java is located
30 ENV JAVA_HOME /usr/lib/jvm/java
31
32 # Set the WILDFLY_VERSION env variable
33 ENV WILDFLY_VERSION 8.2.0.Final
34
35 # Add the WildFly distribution to /opt, and make wildfly the owner of the extracted tar content
36 # Make sure the distribution is available from a well-known place
37 RUN cd $HOME && curl -O http://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-$WILDFLY_VERSION.zip && unzip wildfly-$WILDFLY_
38
39 # Set the JBOSS_HOME env variable
40 ENV JBOSS_HOME /opt/jboss/wildfly
41
42 # Expose the ports we're interested in
43 EXPOSE 8080 9990
44
45 # Set the default command to run on boot
46 # This will boot WildFly in the standalone mode and bind to all interface
47 CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-c", "standalone-full.xml", "-b", "0.0.0.0"]

```



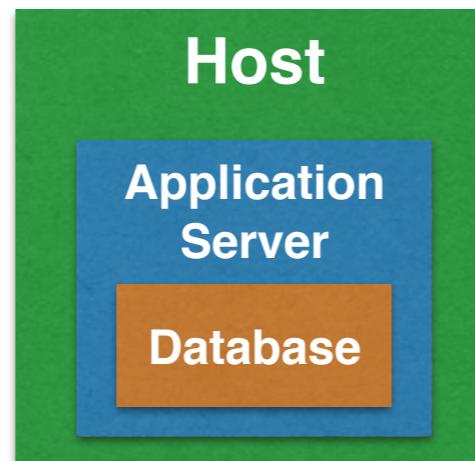
Recipe 1.1

- Docker Machine
- Zero to Docker
 - Downloads boot2docker
 - Setup ssh keys and certificates





Recipe #1.2

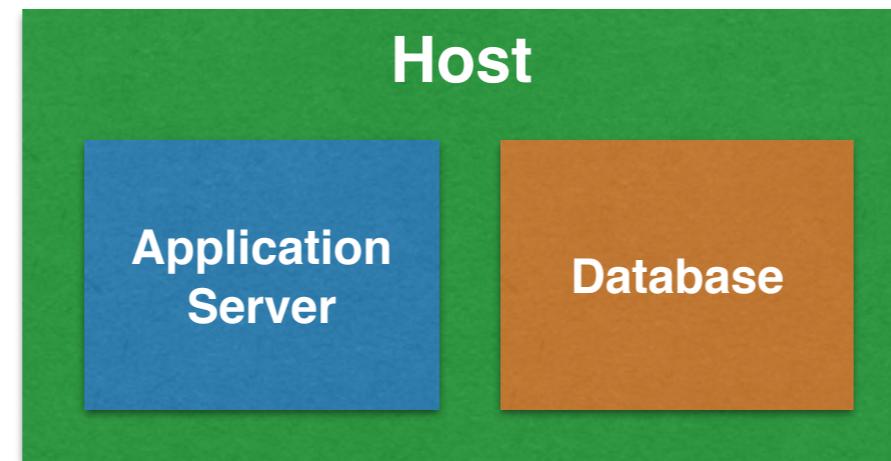


```
FROM jboss/wildfly

RUN curl -L https://github.com/javaee-samples/javaee7-hol/raw/master/solution/
movieplex7-1.0-SNAPSHOT.war -o /opt/jboss/wildfly/standalone/deployments/
movieplex7-1.0-SNAPSHOT.war
```

```
docker run -it -p 8080:8080 arungupta/javaee7-hol
```

Recipe #1.3

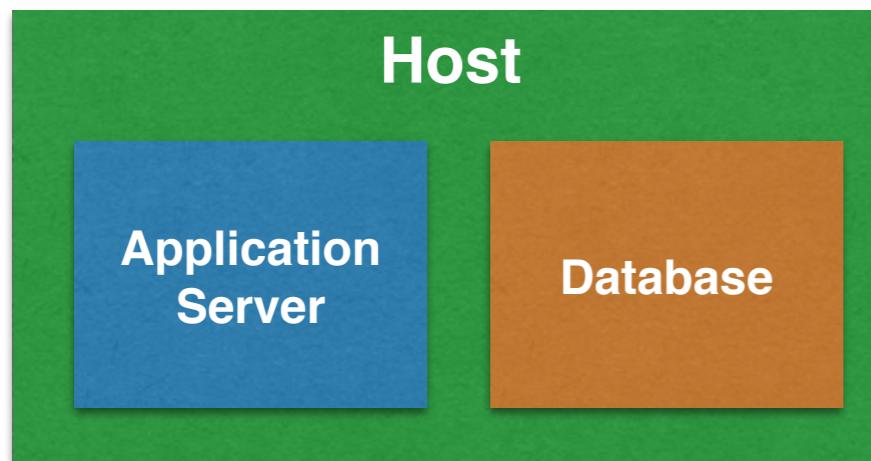


```
docker run --name mysqlDb -e MYSQL_USER=mysql -e MYSQL_PASSWORD=mysql -e MYSQL_DATABASE=sample -e MYSQL_ROOT_PASSWORD=supersecret -d mysql
```

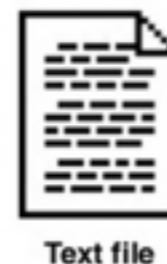
```
docker run --name mywildfly --link mysqlDb:db -p 8080:8080 -d arungupta/wildfly-mysql-javaee7
```

```
data-source add --name=mysqlDS --driver-name=mysql --jndi-name=java:jboss/datasources/ExampleMySQLDS --connection-url=jdbc:mysql://$DB_PORT_3306_TCP_ADDR:$DB_PORT_3306_TCP_PORT/sample?useUnicode=true&characterEncoding=UTF-8 --user-name=mysql --password=mysql --use-ccm=false --max-pool-size=25 --blocking-timeout-wait-millis=5000 --enabled=true
```

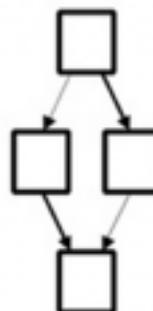
Recipe #1.4



Docker Compose:
Get an app running in one command.



→ \$ docker up →

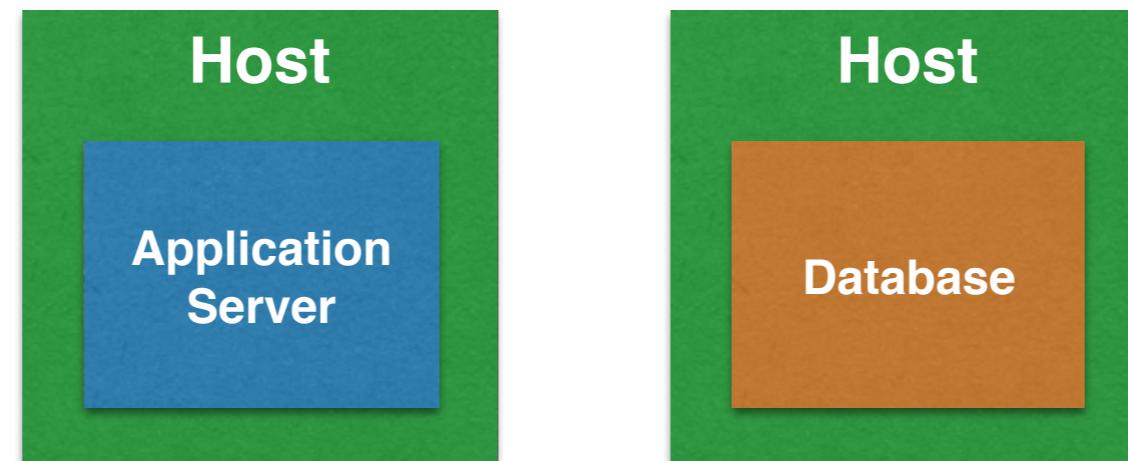


```
mysqlDb:  
  image: mysql:latest  
  environment:  
    MYSQL_DATABASE: sample  
    MYSQL_USER: mysql  
    MYSQL_PASSWORD: mysql  
    MYSQL_ROOT_PASSWORD: supersecret  
  
mywildfly:  
  image: arungupta/wildfly-mysql-javaee7  
  links:  
    - mysqlDb:db  
  ports:  
    - 8080:8080
```

docker-compose up -d



Recipe #1.5



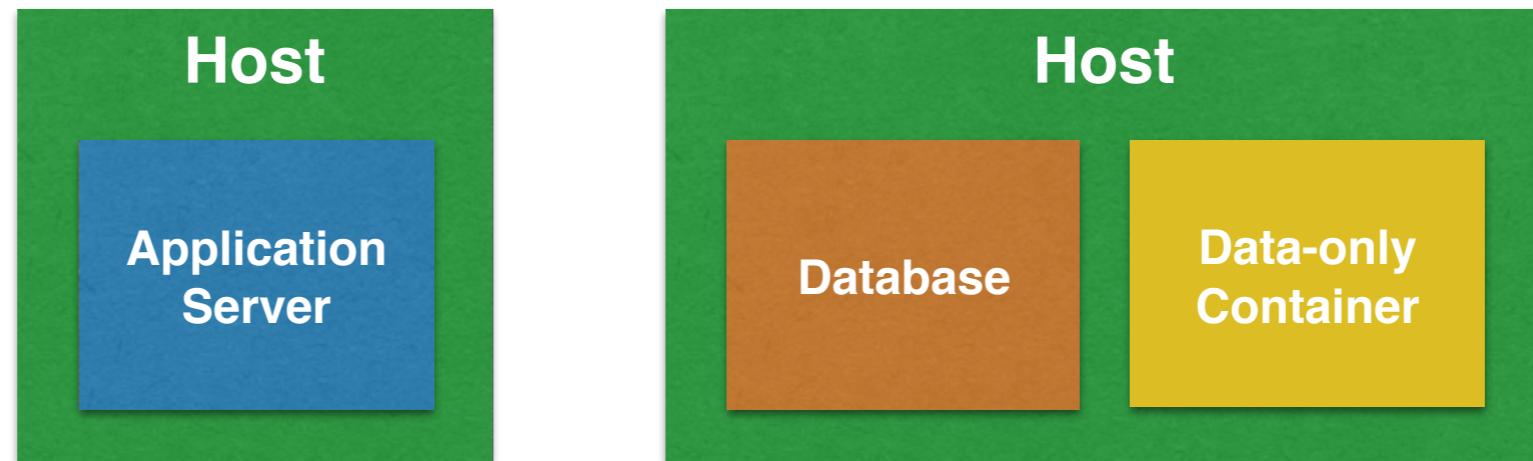
```
docker run --name mysqlDb -e MYSQL_USER=mysql -e MYSQL_PASSWORD=mysql -e MYSQL_DATABASE=sample -e MYSQL_ROOT_PASSWORD=supersecret -d mysql
```

```
data-source add --name=mysqlDS --driver-name=mysql --jndi-name=java:jboss/datasources/ExampleMySQLDS --connection-url=jdbc:mysql://$MYSQL_HOST:$MYSQL_PORT/smpl
```

```
docker build -t arungupta/wildfly-mysql-javaeer .
docker run --name mywildfly -e MYSQL_HOST=<IP_ADDRESS> -e MYSQL_PORT=5306 -p 8080:8080 -d arungupta/wildfly-mysql-javaeer
```

Recipe #1.6

- Data-only containers



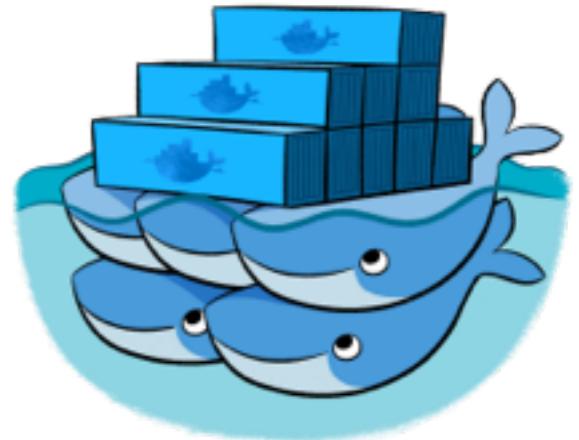
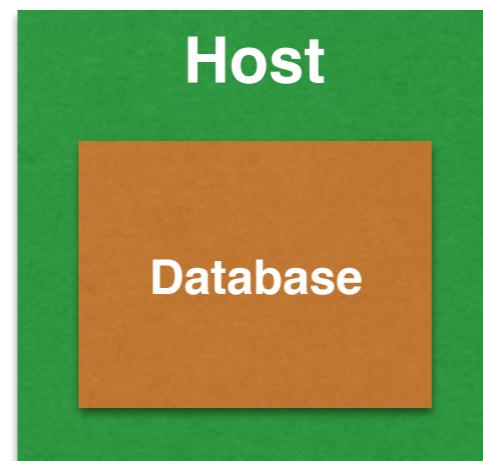
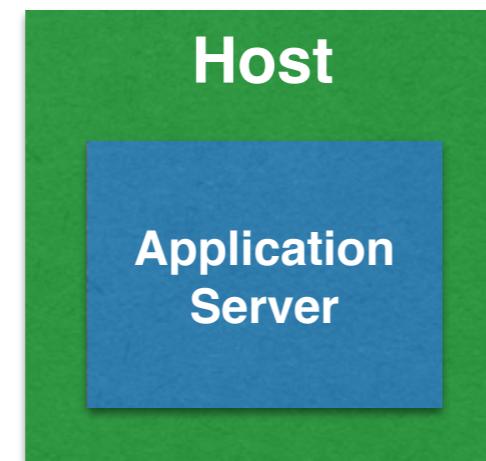
```
docker create --name mysql_data -v /var/lib/mysql mysql
```



```
docker run --name mysql_db --volumes-from mysql_data -v /var/lib/mysql:/var/lib/mysql -e MYSQL_USER
```

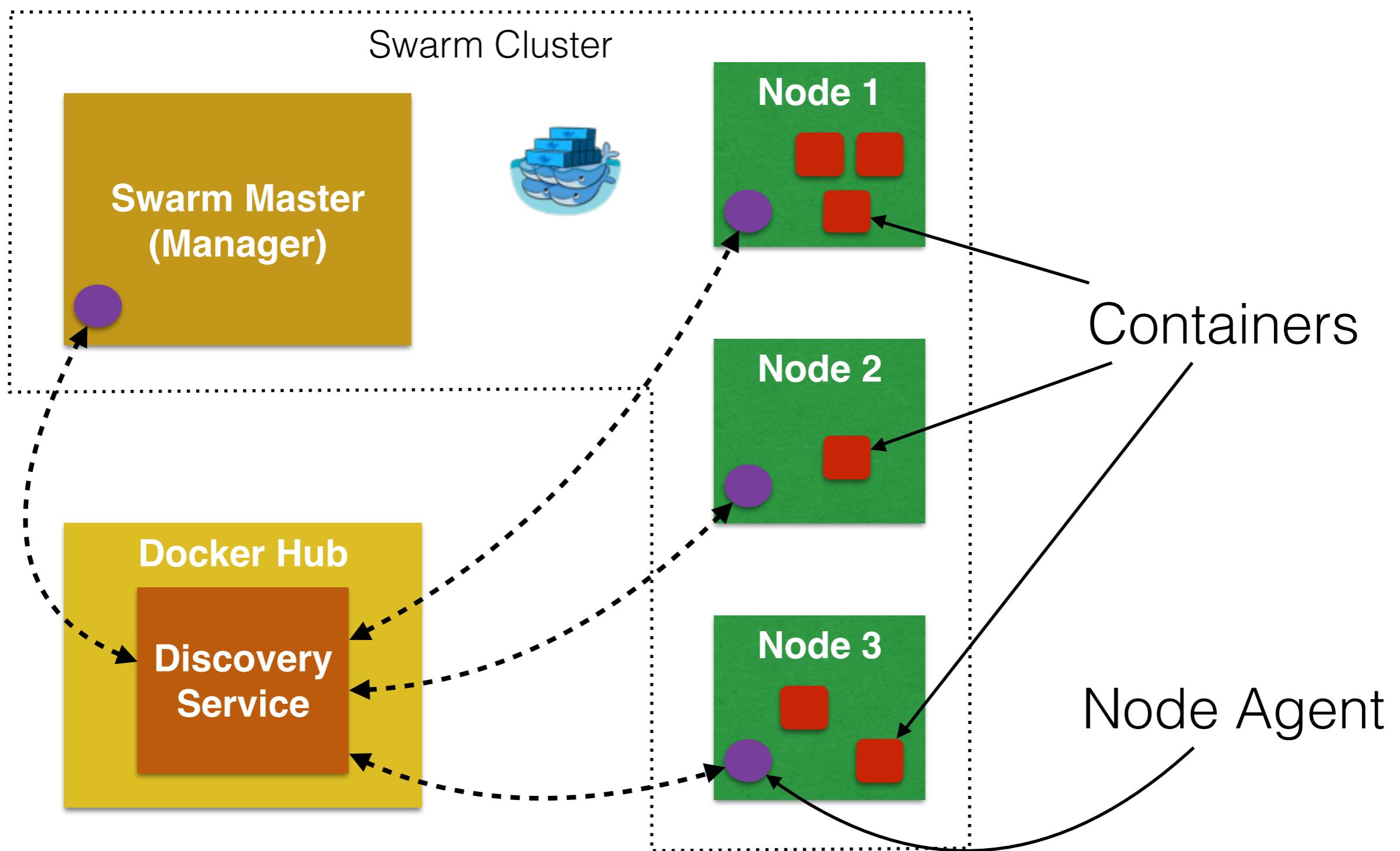


Recipe #1.7



- Docker Swarm: Native clustering for Docker
- Turns a pool of Docker hosts into a single, virtual host

Recipe #1.7



Recipe #1.8

- Running Java EE application on Docker Swarm

- Connect to Docker Swarm

```
$ (docker-machine env --swarm swarm-master)
```

- Start MySQL

```
docker run --name mysqladb . . . -p 3306:3306 -d mysql
```

- Get IP

```
docker inspect --format '{{ .Node.Ip }}' $(docker ps -q --filter 'name=mysqladb')
```

- Run WildFly

```
docker run --name mywildfly -e MYSQL_HOST=<IP> -e MYSQL_PORT=<PORT> -p 8080:8080 -d arungupta/wildfly-mysql-javaee7:host
```

Recipe 1.9



```
docker run -it -p 8080:8080 -v /Users/arungupta/tmp/deployments:/opt/jboss/wildfly/standalone/deployments/:rw jboss/wildfly
```

Docker volumes + Local deployment

```
docker run -it -p 8080:8080 -p 9990:9990 arungupta/wildfly-management
```

Management API + Remote deployment

Recipe #1.10

- Arquillian Cube: Controls the lifecycle of Docker images as part of test cycle - automatically or manually
- Uses Docker REST API to talk to container
- Talk using WildFly remote adapter (in container)

```
git clone https://github.com/javaee-samples/javaee7-samples.git  
git checkout docker  
mvn test -f servlet/simple-servlet/pom.xml -Pwildfly-docker-arquillian
```

Recipe 1.11

- **docker-maven-plugin**: Manage Docker images and containers from Maven
- Goals
 - Image: `docker:build/push/remove`
 - Container: `docker:start/stop`
 - Logs: `docker:logs`

```
<plugin>
  <groupId>org.jolokia</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.11.5-M1</version>
  <configuration>
    <images>
      <image>
        <alias>user</alias>
        <name>arungupta/shoppingcart-user</name>
        <build>
          <from>arungupta/wildfly:8.2</from>
          <assembly>
            <descriptor>assembly.xml</descriptor>
            <basedir>/</basedir>
          </assembly>
          <ports>
            <port>8080</port>
          </ports>
        </build>
        <run>
          <ports>
            <port>user.port:8080</port>
          </ports>
        </run>
      </image>
    </images>
  </configuration>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

assembly.xml

```
<assembly . . .>
  <id>user</id>
  <dependencySets>
    <dependencySet>
      <includes>
        <include>org.javaee7.wildfly.samples:user</include>
      </includes>
      <outputDirectory>/opt/jboss/wildfly/standalone/deployments/</outputDirectory>
      <outputFileNameMapping>user.war</outputFileNameMapping>
    </dependencySet>
  </dependencySets>
</assembly>
```

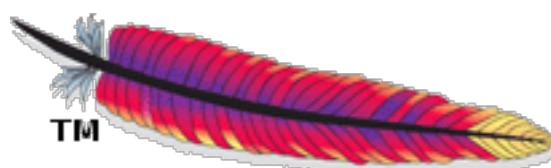
Recipe 1.xx

- Registry
- Save running container state
- List of exact version of the layers used
 - What if latest tag is updated?
 - How to have a predictable set of environment
- Load balancing containers

Docker: Pros and Cons

- PROS
 - Extreme application portability
 - Very easy to create and work with derivative
 - Fast boot on containers
- CONS
 - Host-centric solution
 - No higher-level provisioning
 - No usage tracking/reporting

Application Operating Environment



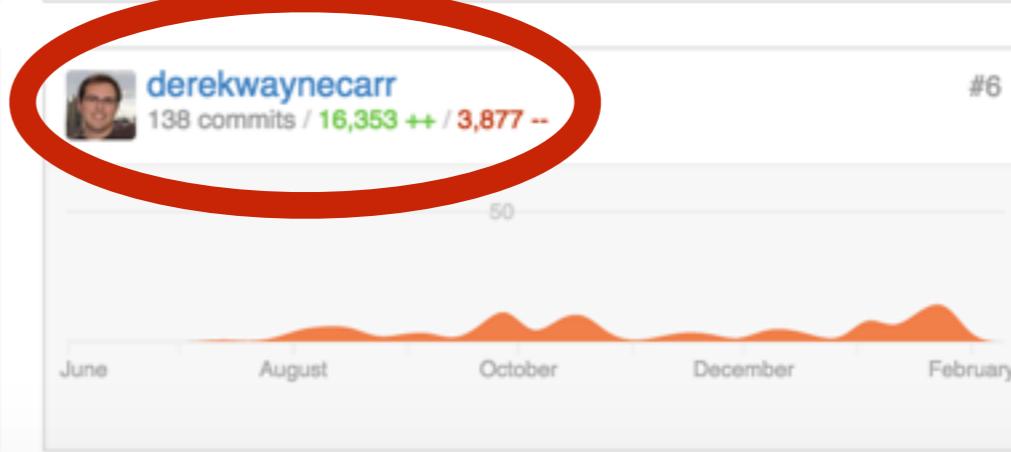
Kubernetes

- Open source orchestration system for Docker containers
- Provide declarative primitives for the “desired state”
 - Self-healing
 - Auto-restarting
 - Schedule across hosts
 - Replicating

Jun 1, 2014 – Feb 10, 2015

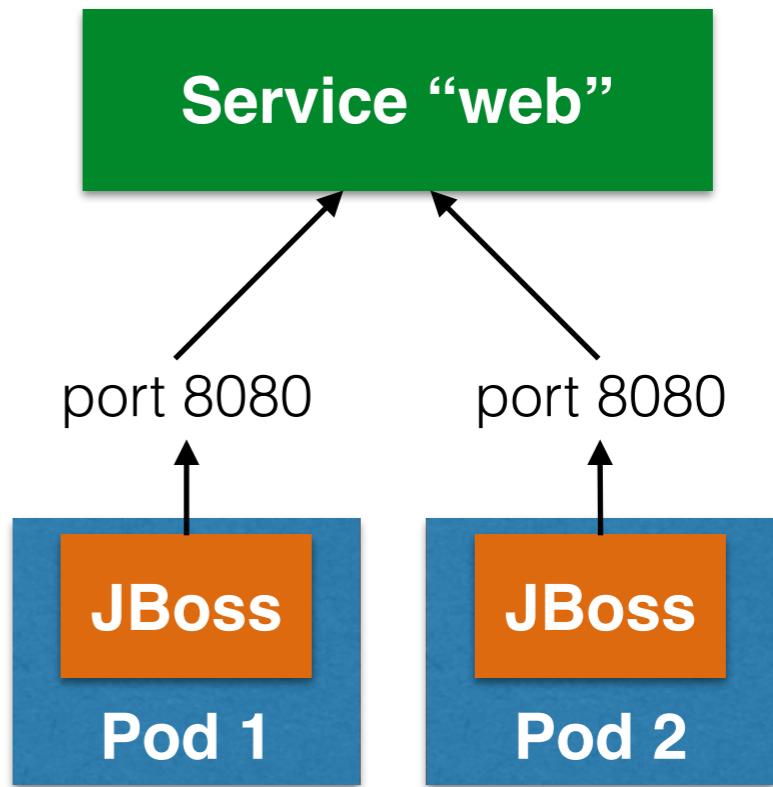
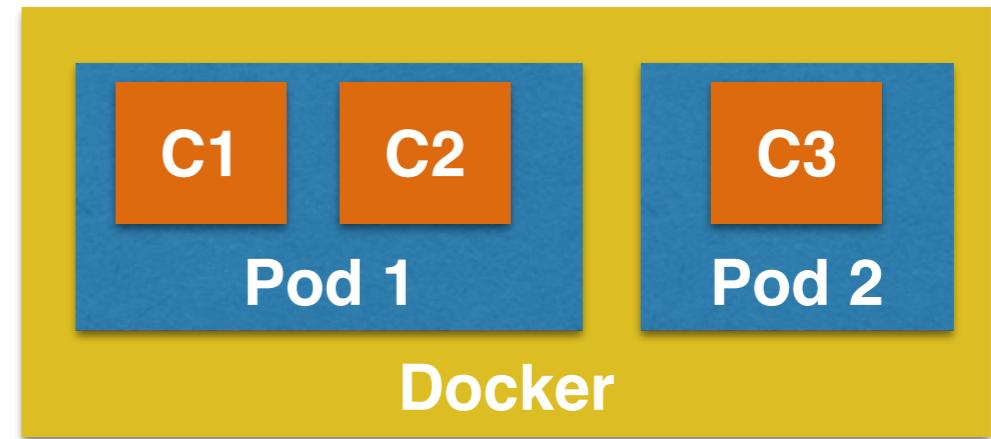
Contributions to master, excluding merge commits

Contributions: **Commits** ▾



Concepts

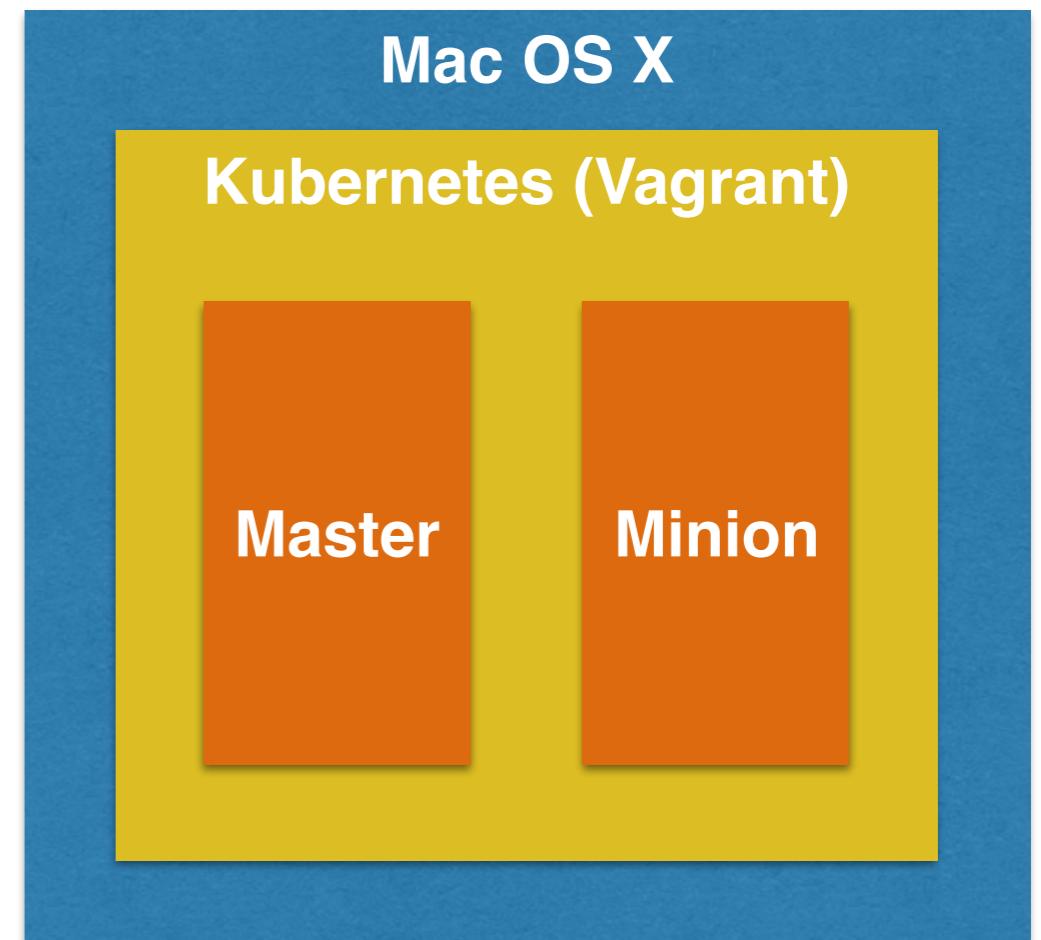
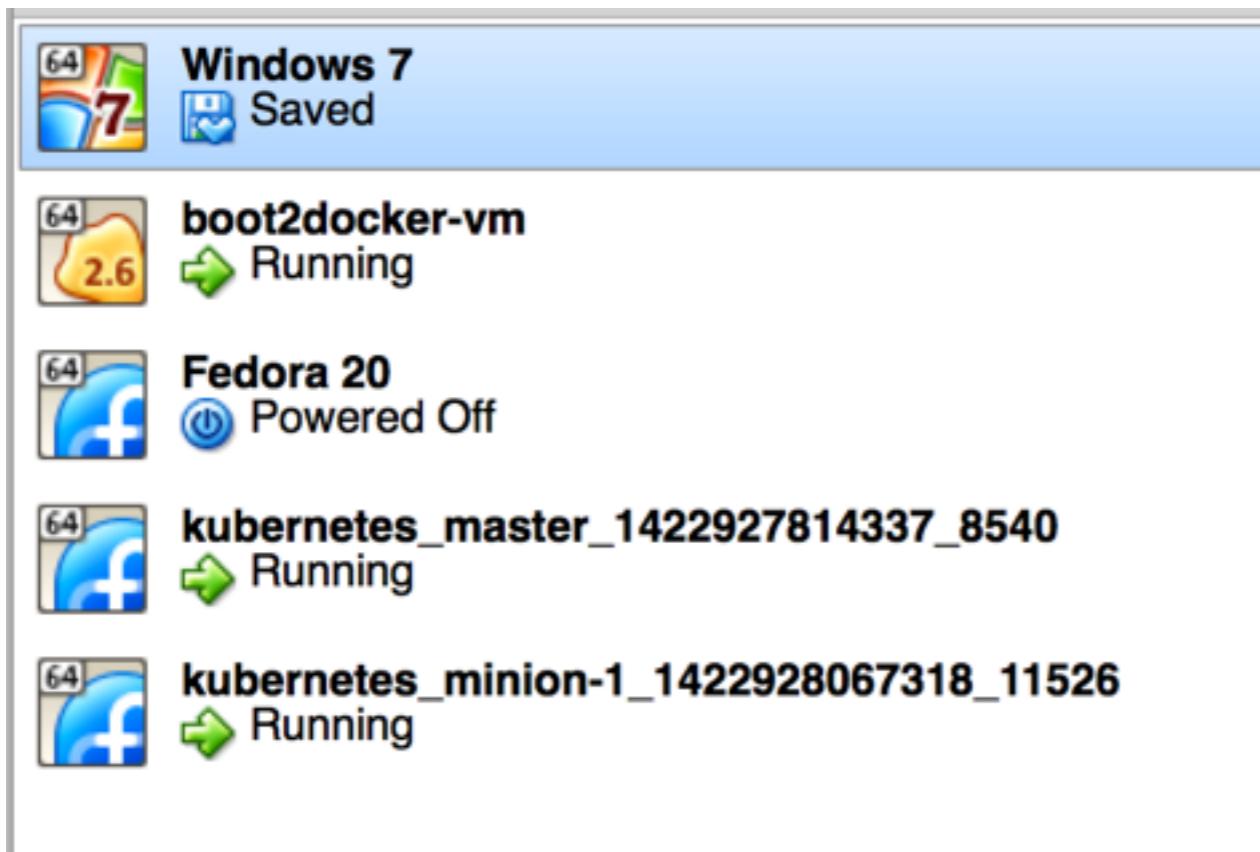
- **Pods**: collocated group of Docker containers that share an IP and storage volume
- **Service**: Single, stable name for a set of pods, also acts as LB
- **Replication Controller**: manages the lifecycle of pods and ensures specified number are running
- **Label**: used to organize and select group of objects



kubectl

- Controls the Kubernetes cluster manager
- `kubectl get pods or minions`
- `kubectl create -f <filename>`
- `kubectl update or delete`
- `kubectl resize --replicas=3 replicationcontrollers <name>`

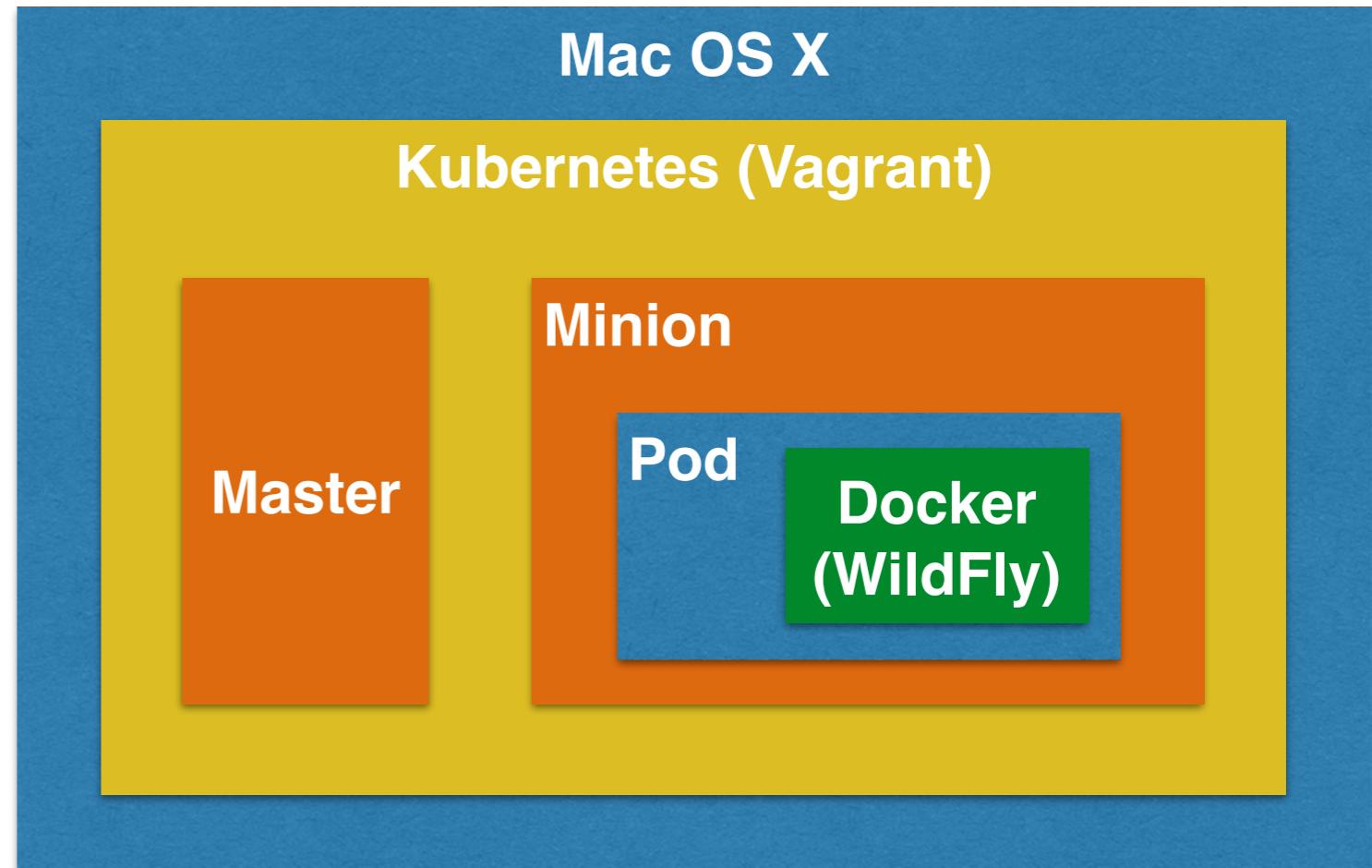
```
export KUBERNETES_PROVIDER=vagrant  
./cluster/kube-up.sh
```



Recipe #2.1

```

1  {
2      "id": "wildfly",
3      "kind": "Pod",
4      "apiVersion": "v1beta1",
5      "desiredState": {
6          "manifest": {
7              "version": "v1beta1",
8              "id": "wildfly",
9              "containers": [
10                 {
11                     "name": "wildfly",
12                     "image": "arungupta/javaee7-hol",
13                     "cpu": 100,
14                     "ports": [
15                         {
16                             "containerPort": 8080,
17                             "hostPort": 8080
18                         },
19                         {
20                             "containerPort": 9090,
21                             "hostPort": 9090
22                         }
23                     ]
24                 }
25             },
26             "labels": {
27                 "name": "wildfly"
28             }
29         }
30     }
31 }
```



Services

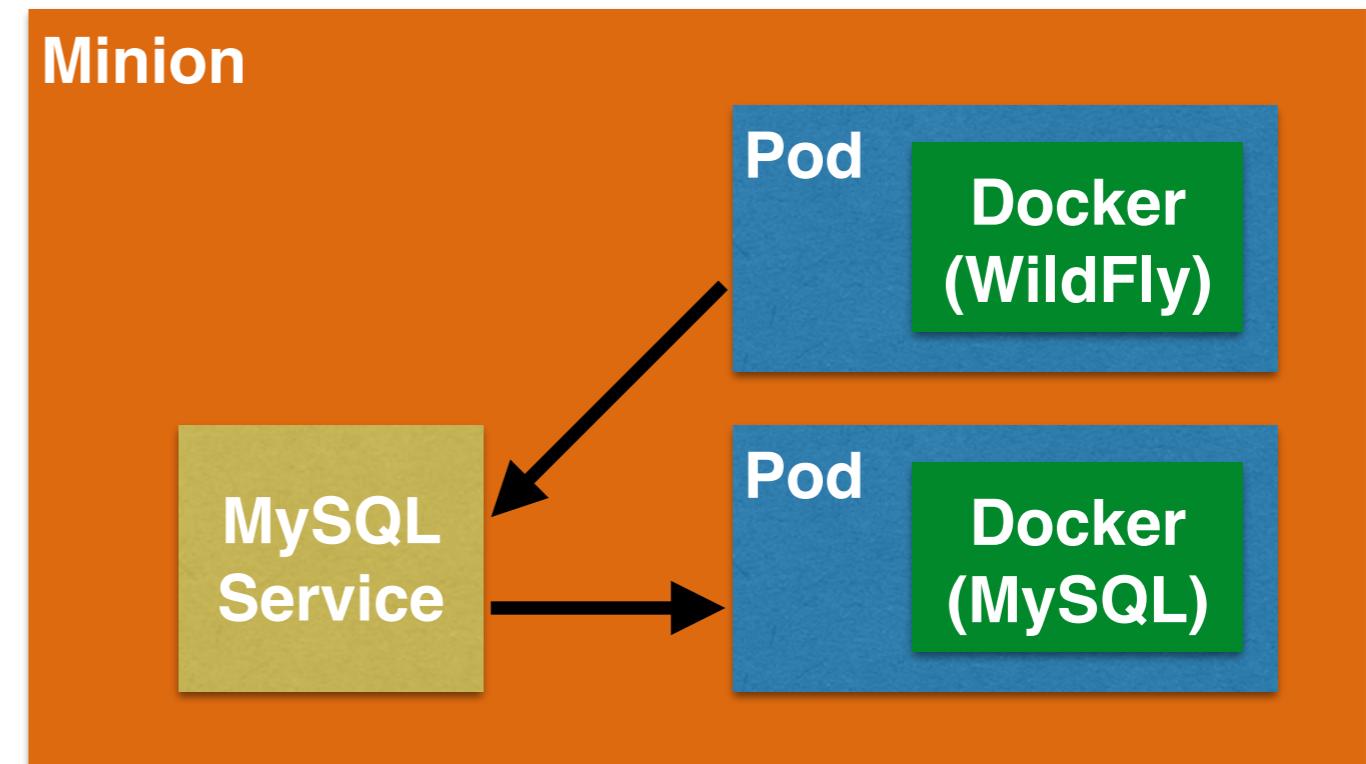
- Abstract a set of pods as a single IP and port
 - Simple TCP/UDP load balancing
- Creates environment variables in other pods
 - Like “Docker links” but across hosts
- Stable endpoint for pods to reference
 - Allows list of pods to change dynamically

Recipe #2.2

```

1  {
2    "id": "mysql",
3    "kind": "Pod",
4    "apiVersion": "v1beta1",
5    "desiredState": {
6      "manifest": {
7        "version": "v1beta1",
8        "id": "mysql",
9        "containers": [
10          {
11            "name": "mysql",
12            "image": "mysql:latest",
13            "cpu": 100,
14            "env": [
15              {
16                "name": "MYSQL_USER",
17                "value": "mysql"
18              },
19              {
20                "name": "MYSQL_PASSWORD",
21                "value": "mysql"
22              },
23              {
24                "name": "MYSQL_DATABASE",
25                "value": "sample"
26              },
27              {
28                "name": "MYSQL_ROOT_PASSWORD",
29                "value": "supersecret"
30              }
31            ],
32            "ports": [
33              {
34                "containerPort": 3306,
35                "hostPort": 3306
36              }
37            ]
38          },
39          "labels": {
40            "name": "mysql"
41          }
42        }
43      }
44    }
45  }

```

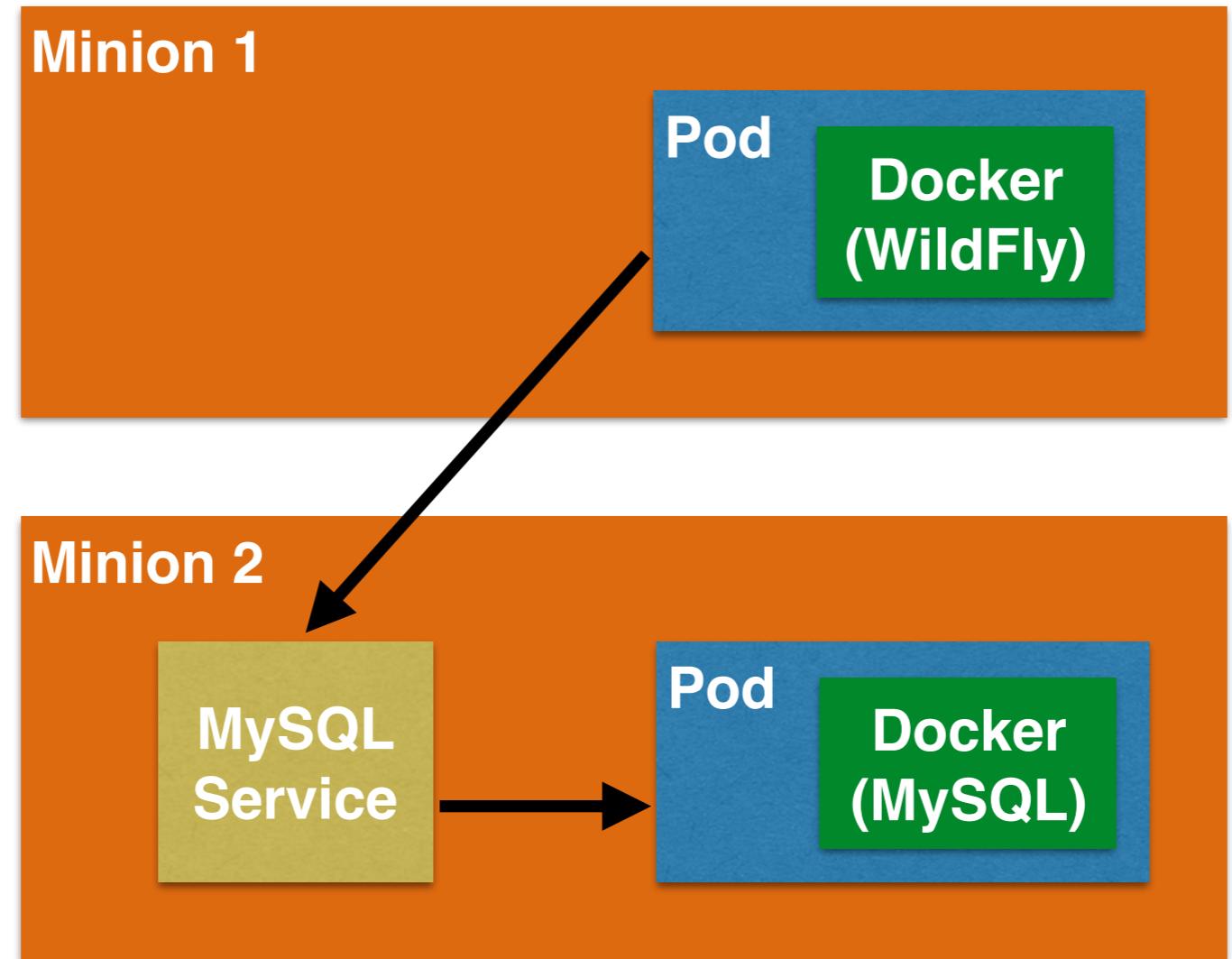


```

1  {
2    "id": "mysql",
3    "kind": "Service",
4    "apiVersion": "v1beta1",
5    "port": 3306,
6    "containerPort": 3306,
7    "selector": {
8      "name": "mysql"
9    },
10   "labels": {
11     "name": "mysql"
12   }
13 }

```

Recipe #2.3



Replication Controller

- Ensures specified number of pod “replicas” are running
- Pod templates are cookie cutters
- Rescheduling
- Manual or auto-scale replicas
- Rolling updates

Recipe #2.4

```

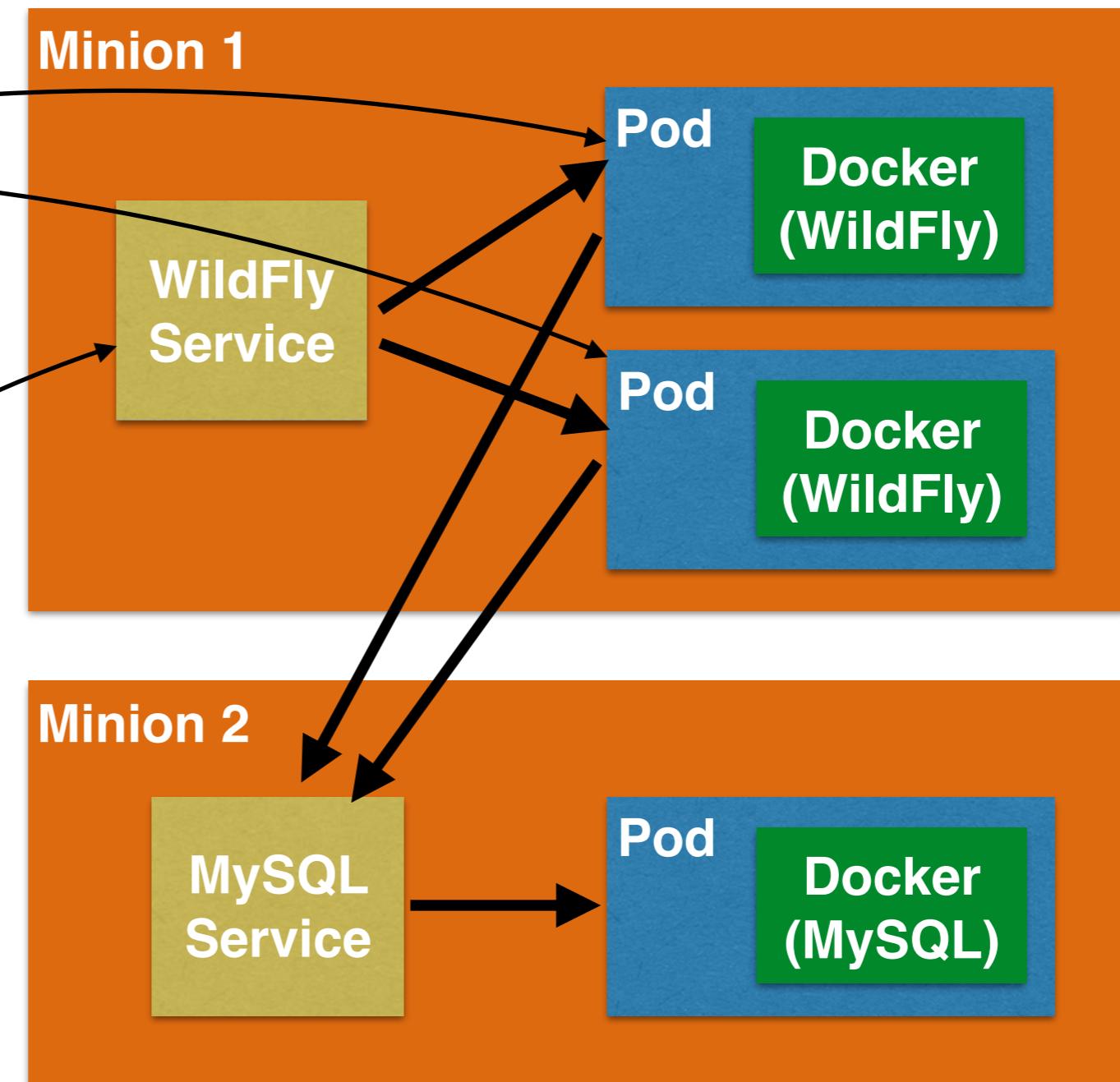
1  id: wildfly-controller
2  kind: ReplicationController
3  apiVersion: v1beta1
4  desiredState:
5    replicas: 2
6    selector:
7      name: wildfly
8    podTemplate:
9      desiredState:
10        manifest:
11          version: v1beta1
12          id: wildfly
13          containers:
14            - name: javaee7-hol
15              image: arungupta/javaee7-hol
16          labels:
17            name: wildfly

```

```

1  id: wildfly
2  kind: Service
3  apiVersion: v1beta1
4  port: 8080
5  containerPort: 8080
6  selector:
7    name: wildfly
8  labels:
9    name: wildfly

```



Kubernetes: Pros and Cons

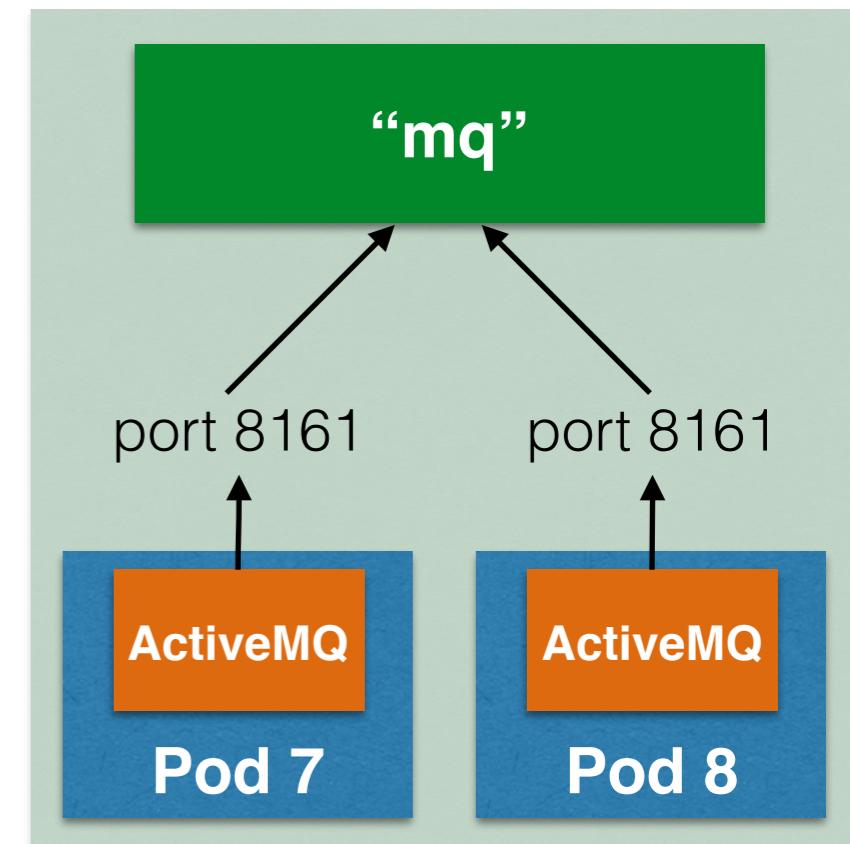
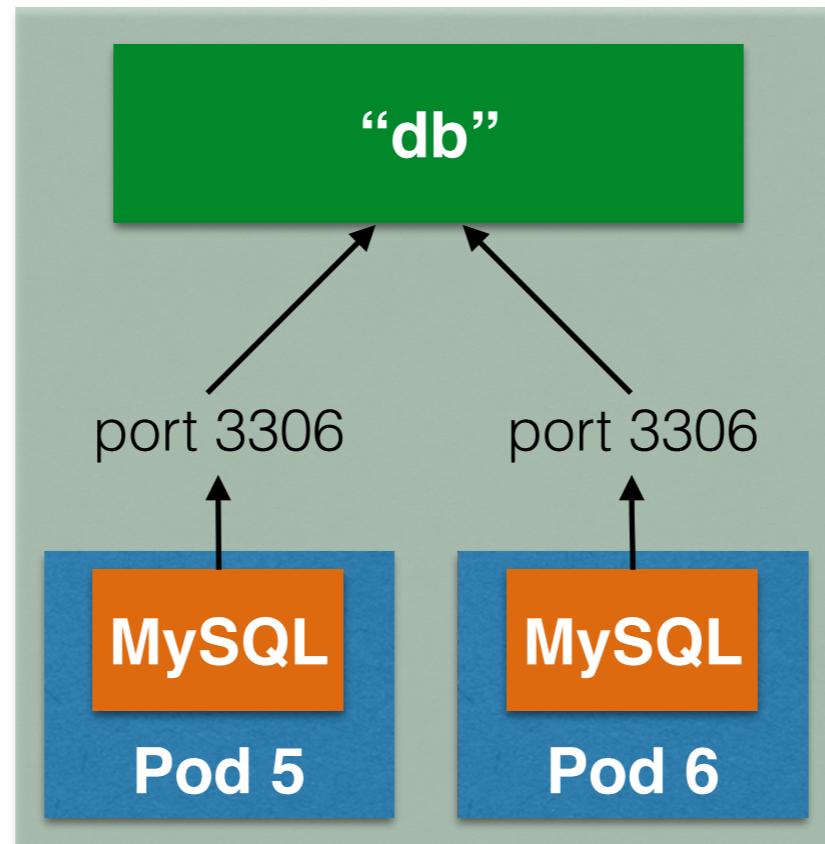
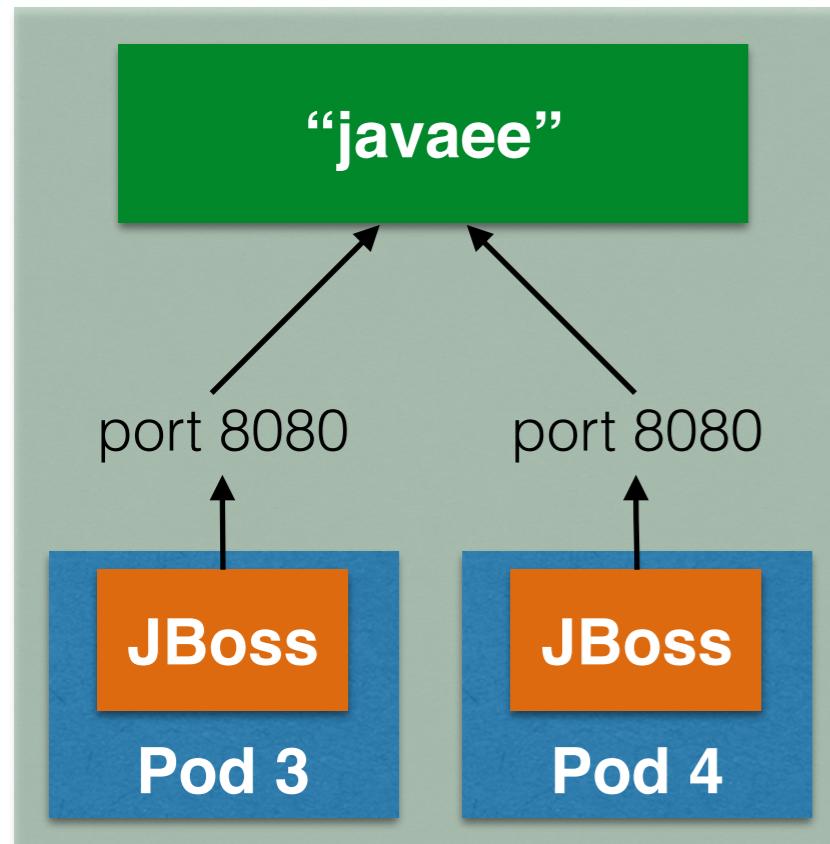
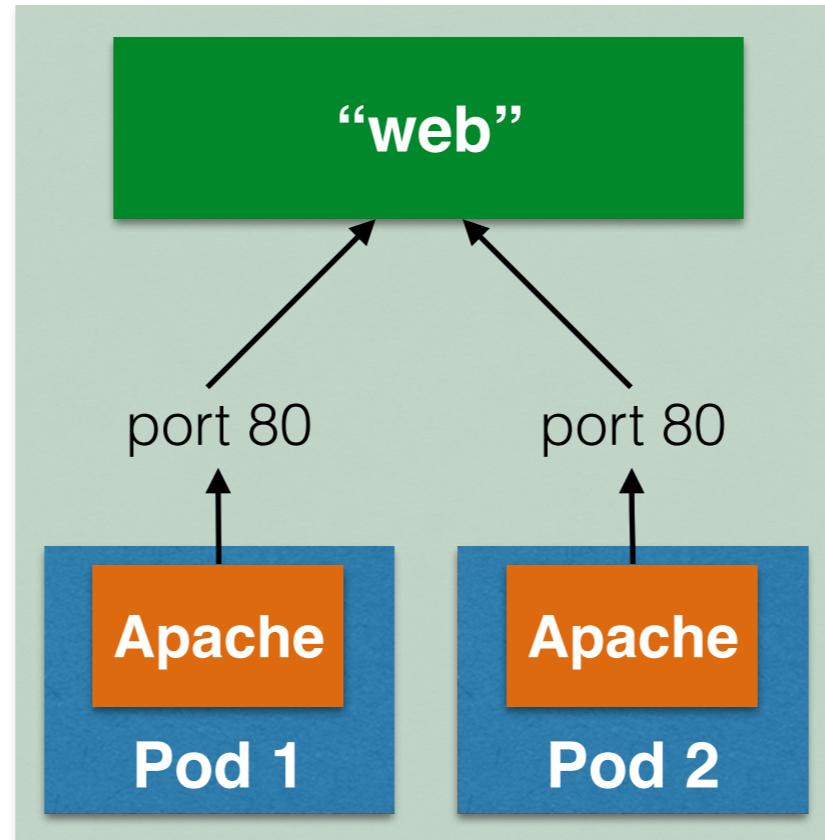
- PROS
 - Manage related Docker containers as a unit
 - Container communication across hosts
 - Availability and scalability through automated deployment and monitoring of pods and their replicas, across hosts

Kubernetes: Pros and Cons

- CONS
 - Lifecycle of applications - build, deploy, manage, promote
 - Port existing source code to run in Kubernetes
 - DevOps: Dev -> Test -> Production
 - No multi-tenancy
 - On-premise (available on GCE)
 - Assumes inter-pod networking as part of infrastructure
 - Requires explicit load balancer



OPENSHIFT



User Experience



Containerized Services



Orchestration



Container



Container Host



OpenShift 3 Features

- Push to production - full DevOps
- Developer and operation tools for building web applications
- Centralized administration and management of application component libraries
- Team and user isolation of containers, builds, and network communication in an easy multi-tenancy system

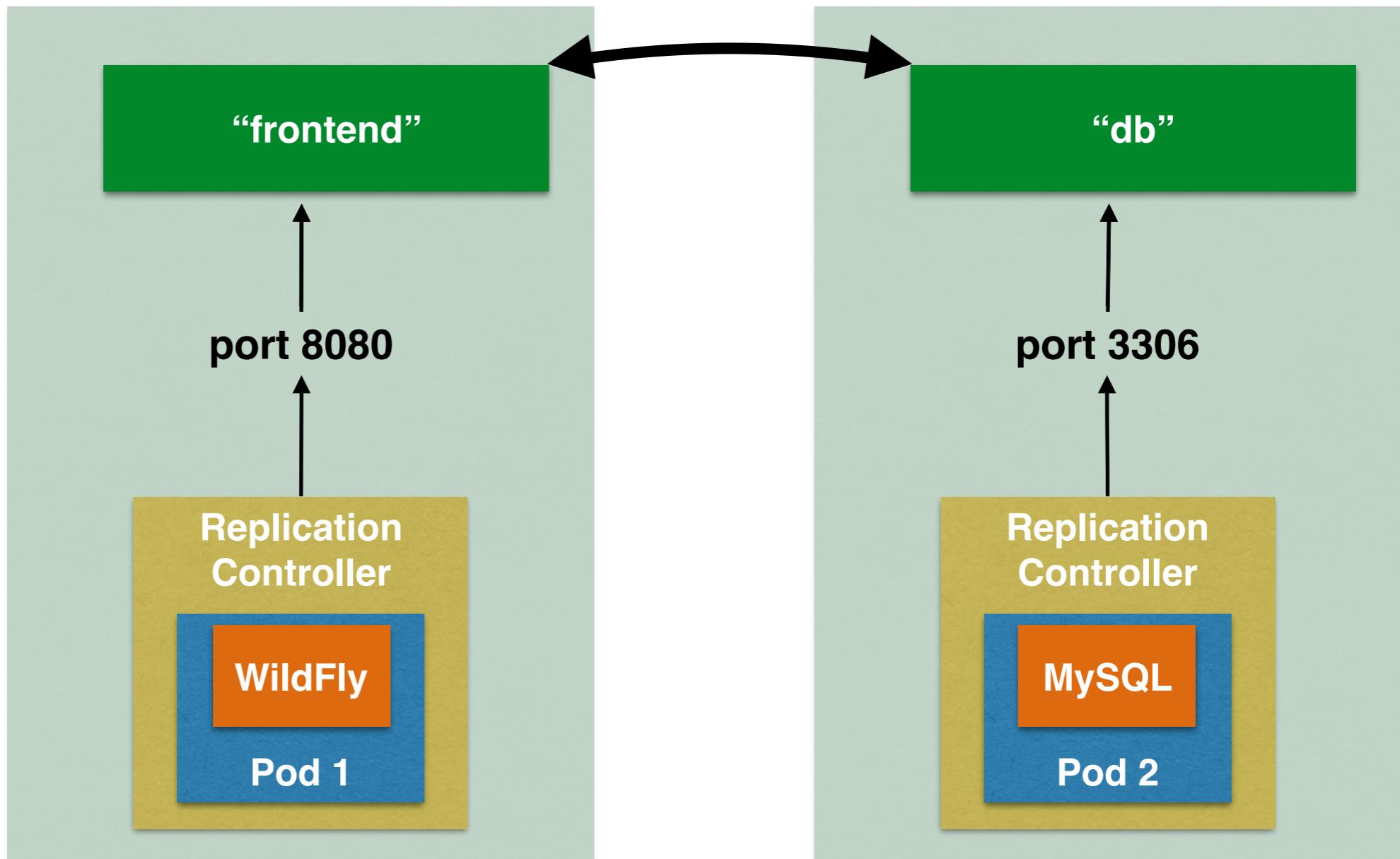
Recipe #3.1

- Start OpenShift as Docker container

```
$ docker run -d --name "openshift-origin" --net=host --privileged \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /tmp/openshift:/tmp/openshift \
openshift/origin start
```

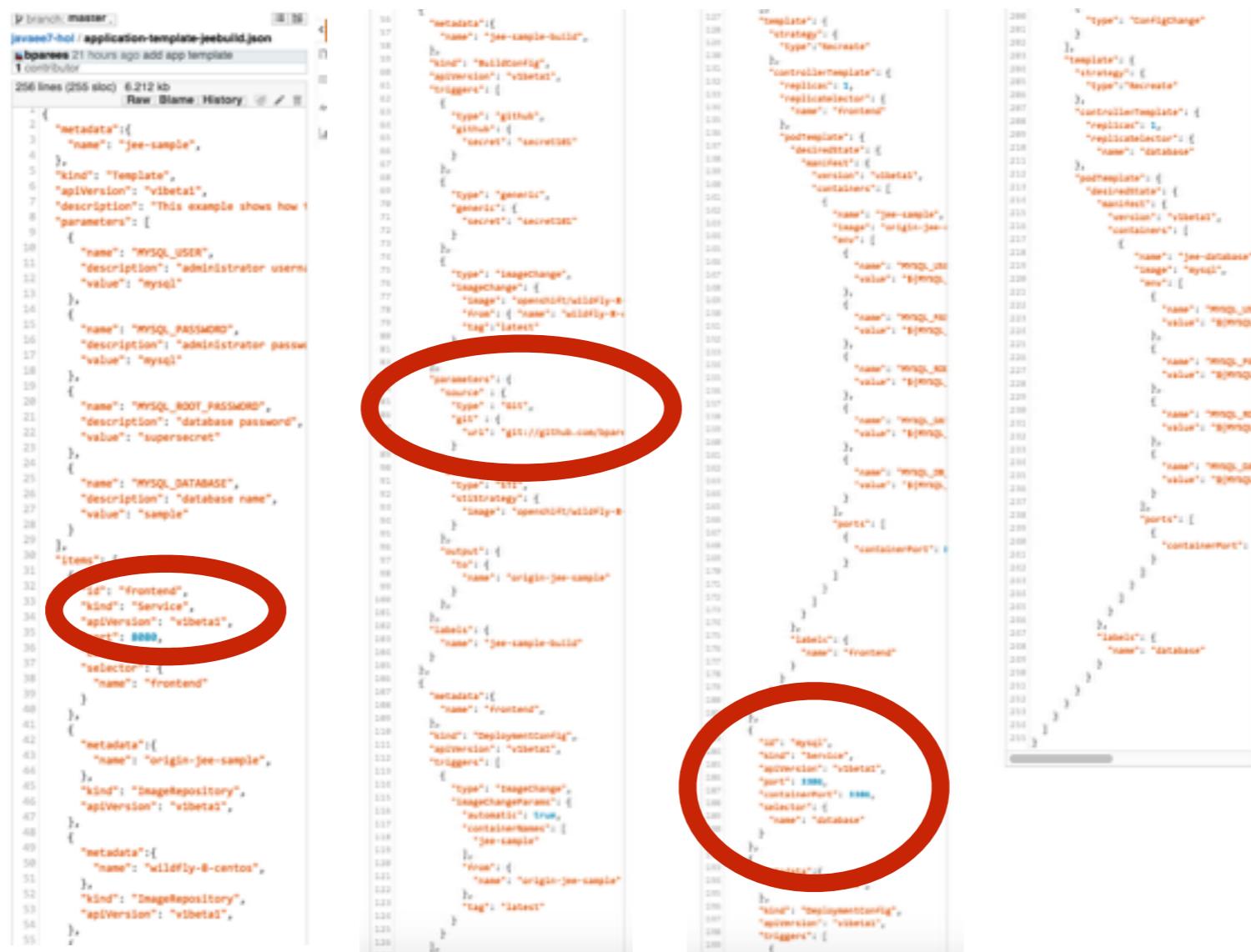
- Or run natively
- Use **osc** (OpenShift Client) instead of **kubectl** with Kubernetes configuration file

Recipe #3.2



Recipe #3.2

- (Alpha) tools generate project JSON configuration file that provide build/deployment



```

{
  "kind": "Service",
  "apiVersion": "v1beta1",
  "metadata": {
    "name": "frontend"
  },
  "spec": {
    "selector": {
      "matchLabels": {
        "app": "origin-jee-sample"
      }
    },
    "ports": [
      {
        "port": 8080,
        "targetPort": 8080
      }
    ],
    "type": "LoadBalancer"
  }
}

{
  "kind": "DeploymentConfig",
  "apiVersion": "v1beta1",
  "metadata": {
    "name": "jee-sample-build"
  },
  "spec": {
    "replicas": 1,
    "strategy": {
      "type": "Rolling"
    },
    "triggers": [
      {
        "type": "ImageChange",
        "imageChangeParams": {
          "from": {
            "kind": "ImageStreamTag",
            "name": "wildfly-8-centos",
            "tag": "latest"
          }
        }
      }
    ],
    "template": {
      "spec": {
        "containers": [
          {
            "name": "jee-sample",
            "image": "origin-jee-sample:latest",
            "env": [
              {
                "name": "MYSQL_USER",
                "value": "mysql"
              },
              {
                "name": "MYSQL_PASSWORD",
                "value": "mysql"
              },
              {
                "name": "MYSQL_ROOT_PASSWORD",
                "value": "supersecret"
              },
              {
                "name": "MYSQL_DATABASE",
                "value": "sample"
              }
            ],
            "ports": [
              {
                "containerPort": 8080
              }
            ]
          }
        ],
        "imagePullSecrets": [
          {
            "name": "registry-secret"
          }
        ]
      }
    }
  }
}

```

Projects

OpenShift 3 Sample

Filter by labels

Label key

+ Filter

Overview

Project OpenShift 3 Sample

frontend - routing TCP traffic on 172.30.17.158:8080 to port 8080

There are currently no pods for this service.

mysql - routing TCP traffic on 172.30.17.254:3306 to port 3306

a few seconds ago , triggered by deployment configuration change

jee-database

Ports:

Image: mysql

3306 (TCP)

Running

172.17.0.5

This screenshot shows the 'Overview' section of the OpenShift 3 Sample project. It lists two services: 'frontend' and 'mysql'. The 'frontend' service is currently inactive with no pods. The 'mysql' service is active, with a single pod named 'jee-database' running on IP 172.17.0.5. The pod is configured to expose port 3306 (TCP). A note indicates the service was updated 'a few seconds ago' due to a deployment configuration change.

✓ A build of jee-sample-build is complete.

a few seconds ago , triggered by new image for test/origin-jee-sample:latest

jee-sample

Ports:

Image: test/origin-jee-sample (5e9dac86eb)

Build: jee-sample-build (71fd8dca-b)

</> Source:

Running

172.17.0.9

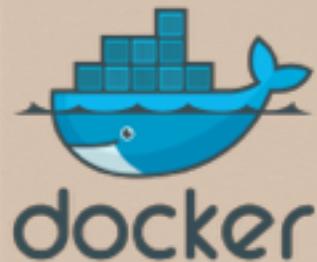
This screenshot displays a success message indicating that a build of 'jee-sample-build' has completed successfully. It provides details about the build trigger ('new image for test/origin-jee-sample:latest'), the resulting pod ('jee-sample'), its image ('test/origin-jee-sample'), the build ID ('71fd8dca-b'), and its current state ('Running' on IP 172.17.0.9). The pod also exposes port 8080 (TCP).

Recipe #3.3

- Integration with JBoss Developer Studio (cooking)



Summary



- Container runtime and image distribution



- Roll your own solutions for everything
- Runtime and operational management of containers



- Lifecycle of applications - build, deploy, manage, promote
- Manage tens of thousands of applications with teams

References

- blog.arungupta.me/topics/containers/
- github.com/openshift/origin