DevOps with Java EE

Thomas Qvarnström, @tvarnst
Arun Gupta, @arungupta

Thomas Qvarnström

Technical Marketing Manager

@tqvarnst
blog.thomasqvarnstrom.com
tqvarnst@redhat.com

redhat

Arun Gupta

Director, Technical Marketing & Developer Advocacy

@arungupta

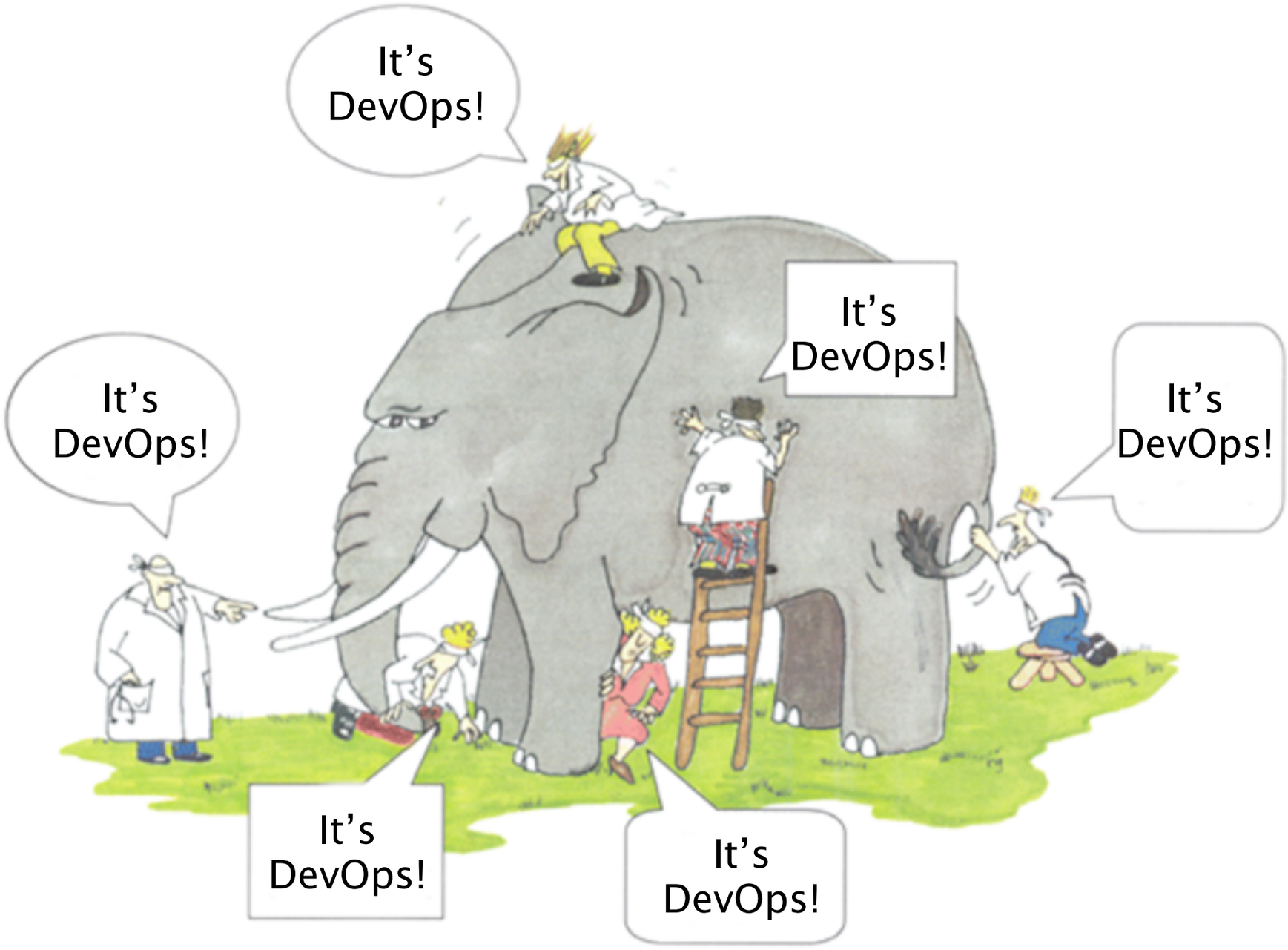
blog.arungupta.me

arungupta@redhat.com

Agenda

- What is DevOps?
- Docker Containers
- Testing Java EE with Docker: Arquillian Cube
- DevOps with Java EE using Docker

What is DevOps?



redhat

DevOps is...

* A philosophy that starts with passion
* A cultural, professional movement with attitude and values
* A reaction to poor communication
* About creating visibility between dev and ops
* About the symbiotic relationship between dev and ops
* Cross-functional teams over organizational silos
* Products not projects
* Automation over documentation (and more automation... and more...)
* About creating self-service infrastructure for teams
* Knowing that good software doesn't end with development / release
* Software that doesn't require support
* Ensuring a continual feedback loop between development and operations
* Cross-functional teams over organizational silos
* Creating products that are owned by the delivery team
* Knowing that a project is only finished when it is retired from production
* Something you can do without doing agile

https://sites.google.com/a/jezhumble.net/devops-manifesto/


STRIKE
GRAD YEAR
TRAVEL
MARRIAGE
KIDS
DIVORCE
I'm BACK!
DIRTY LAUNDRY

Five “C”s of DevOps

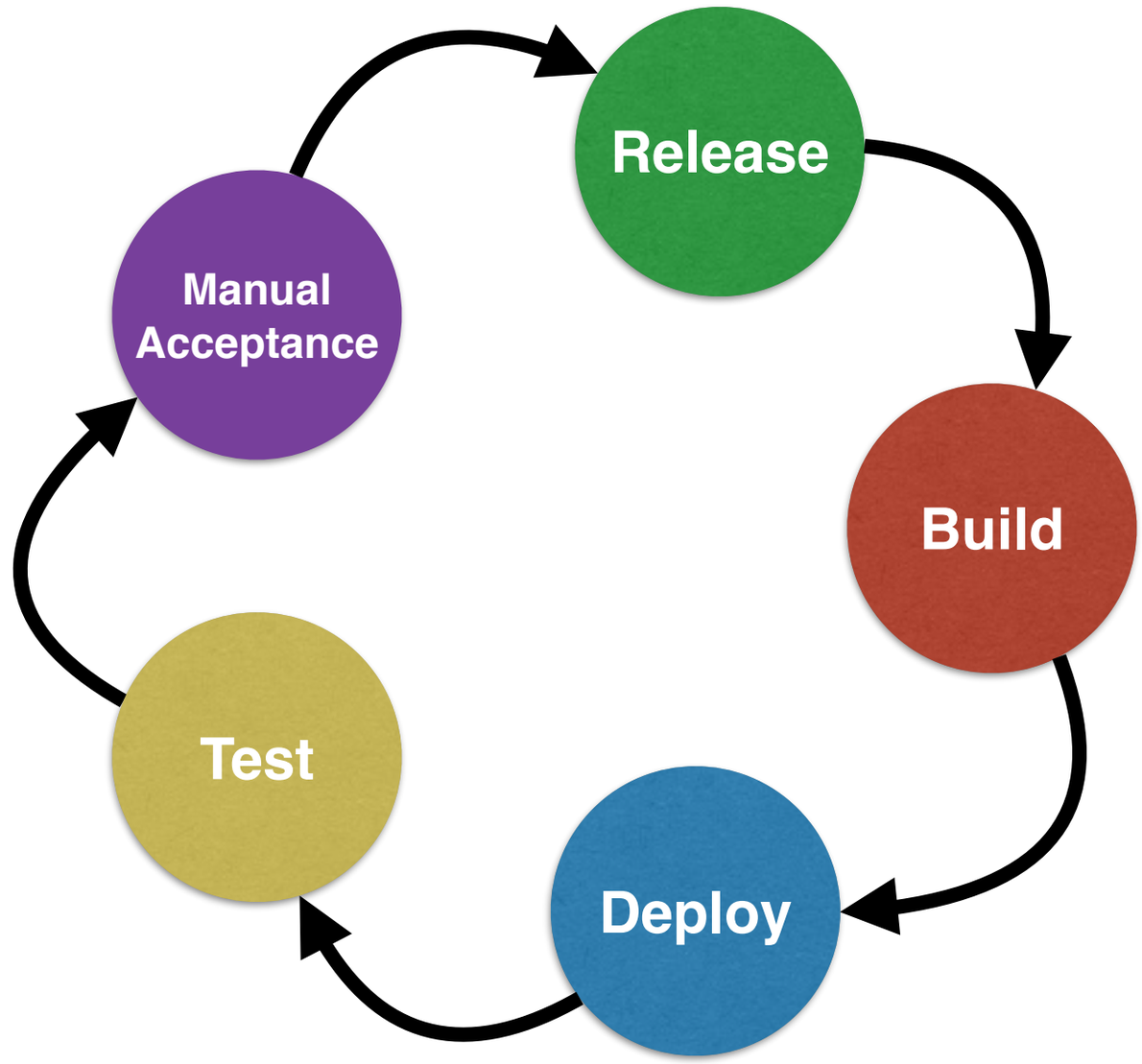
- **C**ollaboration between “dev” and “ops”
- **C**ulture
- **C**ode everything - application and configuration
- **C**onsistency - automation over documentation
- **C**ontinuous delivery


“Dev” “Ops”

“Never send a human to do a machine’s job”

Continuous Delivery

- Continuous Integration
- Fail fast and recover
- Self service
- 100% Automation
- Push to Prod
- Proactive monitoring and metrics



	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundary dev/ ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• Team priorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and reliable releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• Changes to datastores automatically performed as part of the deployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functional tests • Manual performance/ security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceability • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

http://blog.arungupta.me/continuous-integration-delivery-deployment-maturity-model/

What is Docker?

- Open source project and company

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

Docker Contributors by employers/hackers

gistfile1.txt

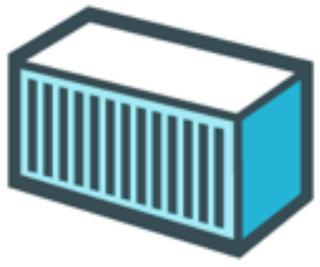
```
Top changeset contributors by employer
(Unknown)                    4520 (47.9%)
"Docker"                     3821 (40.5%)
"Red Hat"                     685 (7.3%)
"IBM"                         232 (2.5%)
"Google"                      119 (1.3%)
"Cisco"                        49 (0.5%)
"Amadeus"                       4 (0.0%)
"VMWare"                        2 (0.0%)
"CoreOS"                        1 (0.0%)
```

https://gist.github.com/arun-gupta/7d5a373099ff831d7213

http://www.infoworld.com/article/2925484/application-virtualization/look-whos-helping-build-docker-besides-docker-itself.html

Build

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

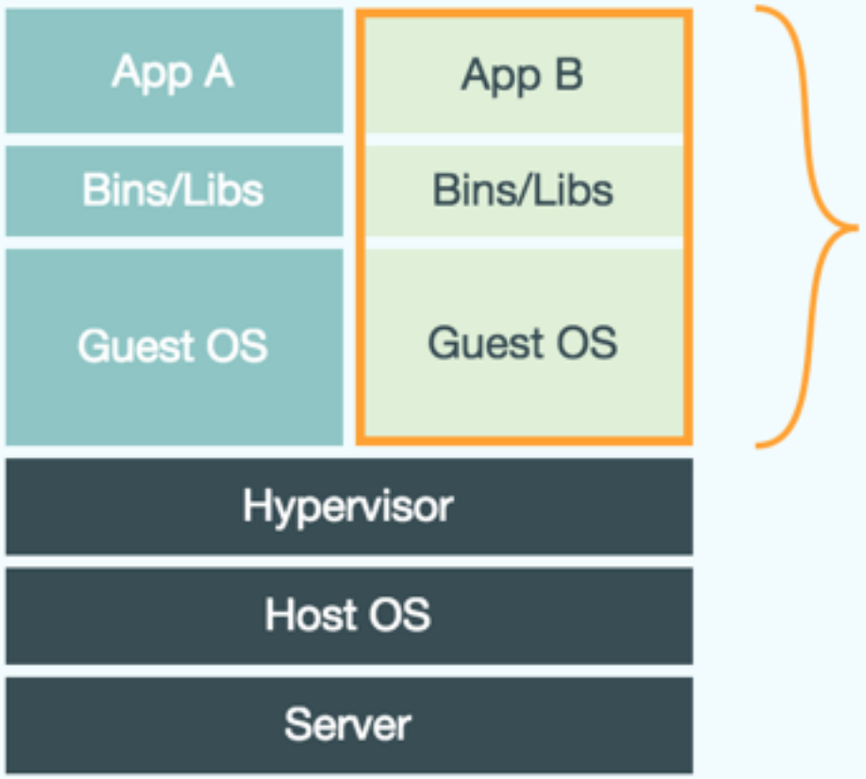


Ship

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

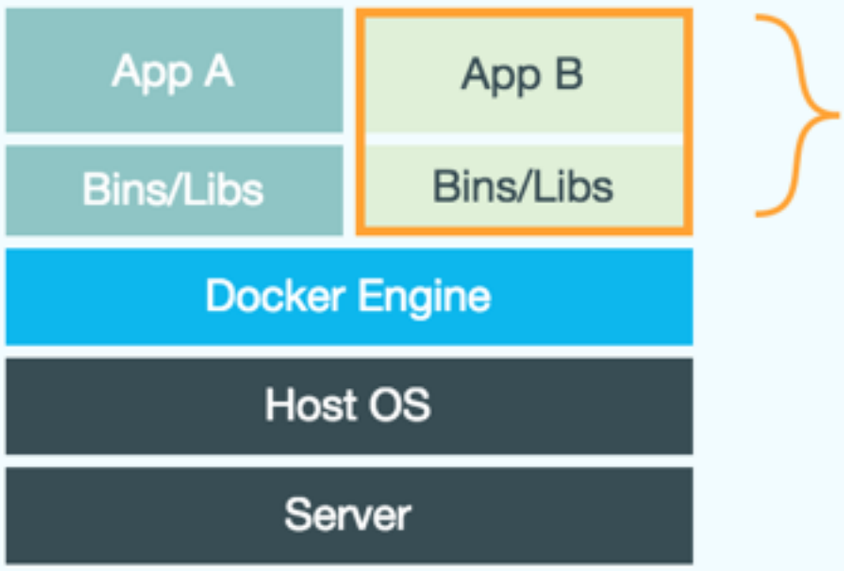
Run

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



Virtual Machines

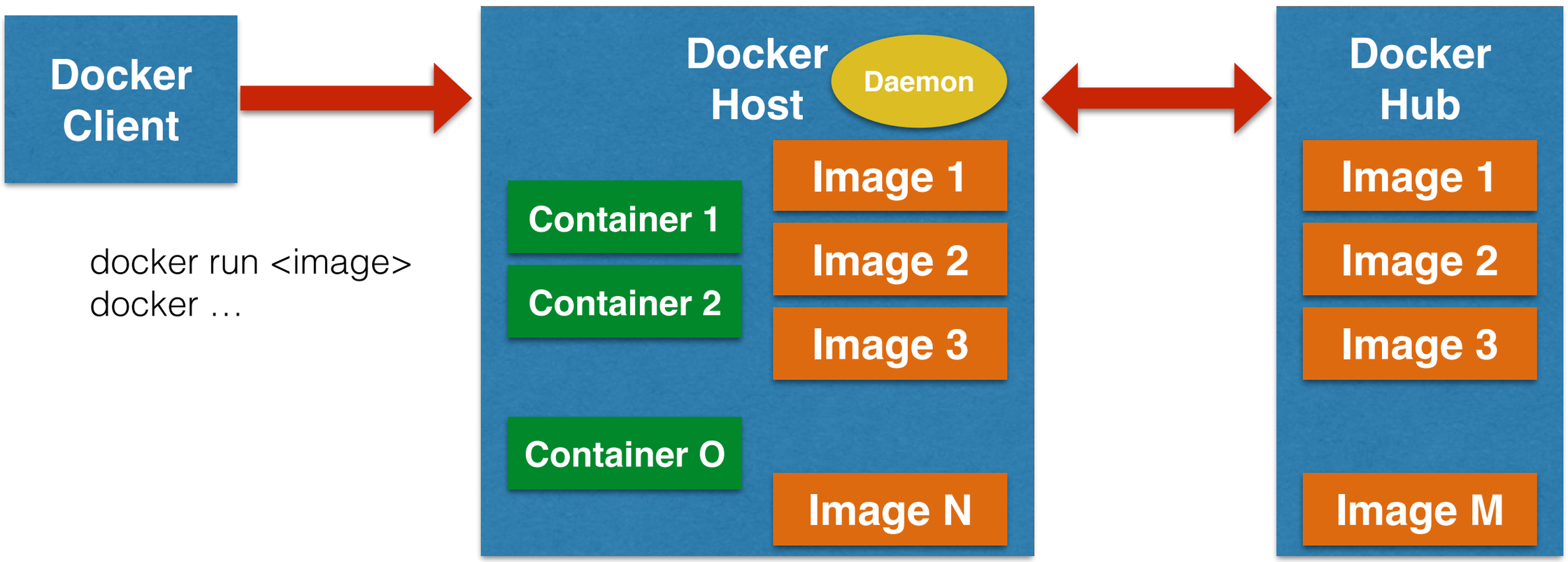
Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.



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.

Docker Workflow



docker run <image>
docker …

Docker Machine

- Create Docker Host on computer or cloud provider

```
docker-machine create --driver=virtualbox
myhost
```

  - Configure Docker client to talk to host
  - Create and pull images
  - Start, stop, restart containers
  - Upgrade Docker

- Not recommended for production yet

Docker Compose

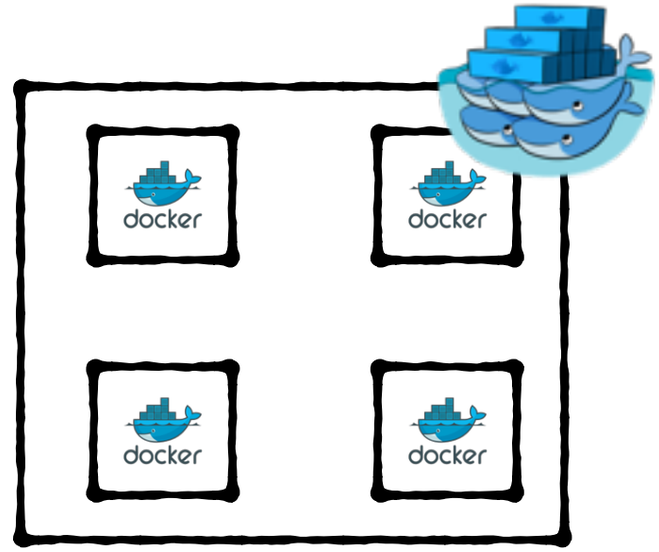
- Defining and running multi-container applications
- Configuration defined in a single file
- Great for dev, staging, and CI
- Not recommended for production yet

docker-compose.yml

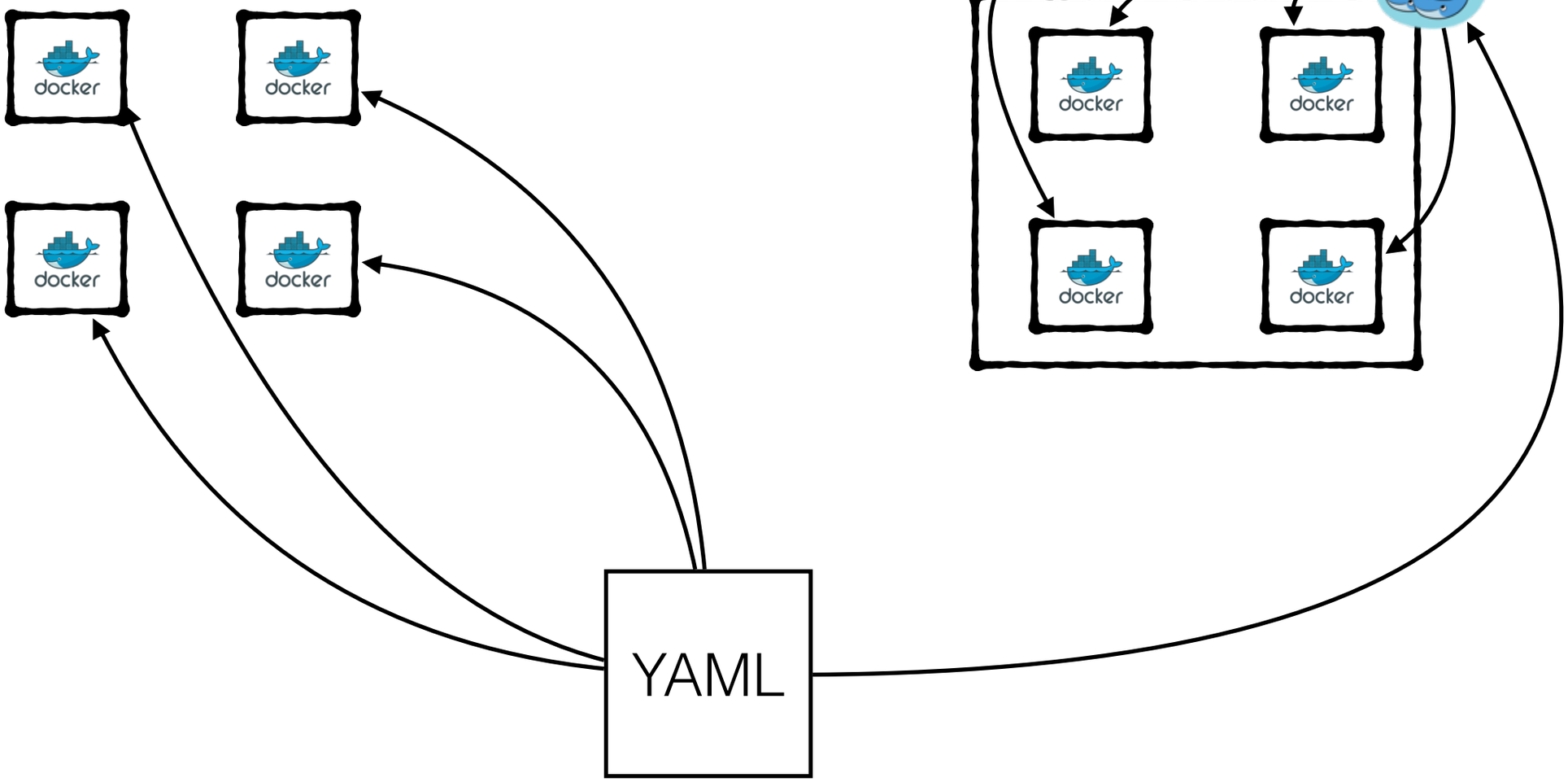
```
mysqldb:
  image: mysql
  environment:
    MYSQL_DATABASE: sample
    MYSQL_USER: mysql
    MYSQL_PASSWORD: mysql
    MYSQL_ROOT_PASSWORD: supersecret
mywildfly:
  image: arungupta/wildfly-mysql-javaee7
  links:
    - mysqldb:db
```

Docker Swarm

- Native clustering for Docker
- Provides a unified interface to a pool of Docker hosts
- Fully integrated with Machine
- Serves the standard Docker API
- Partially integrated with Compose
- Not recommended for production yet



Machine



Swarm

Compose

Advantages of Containers

- Immutability
- Reproducibility
- Isolation
- Faster deployments
- Portability - “it works on my machine”
- Snapshotting
- Security sandbox
- Limit resource usage
- Simplified dependency
- Sharing

Kubernetes

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

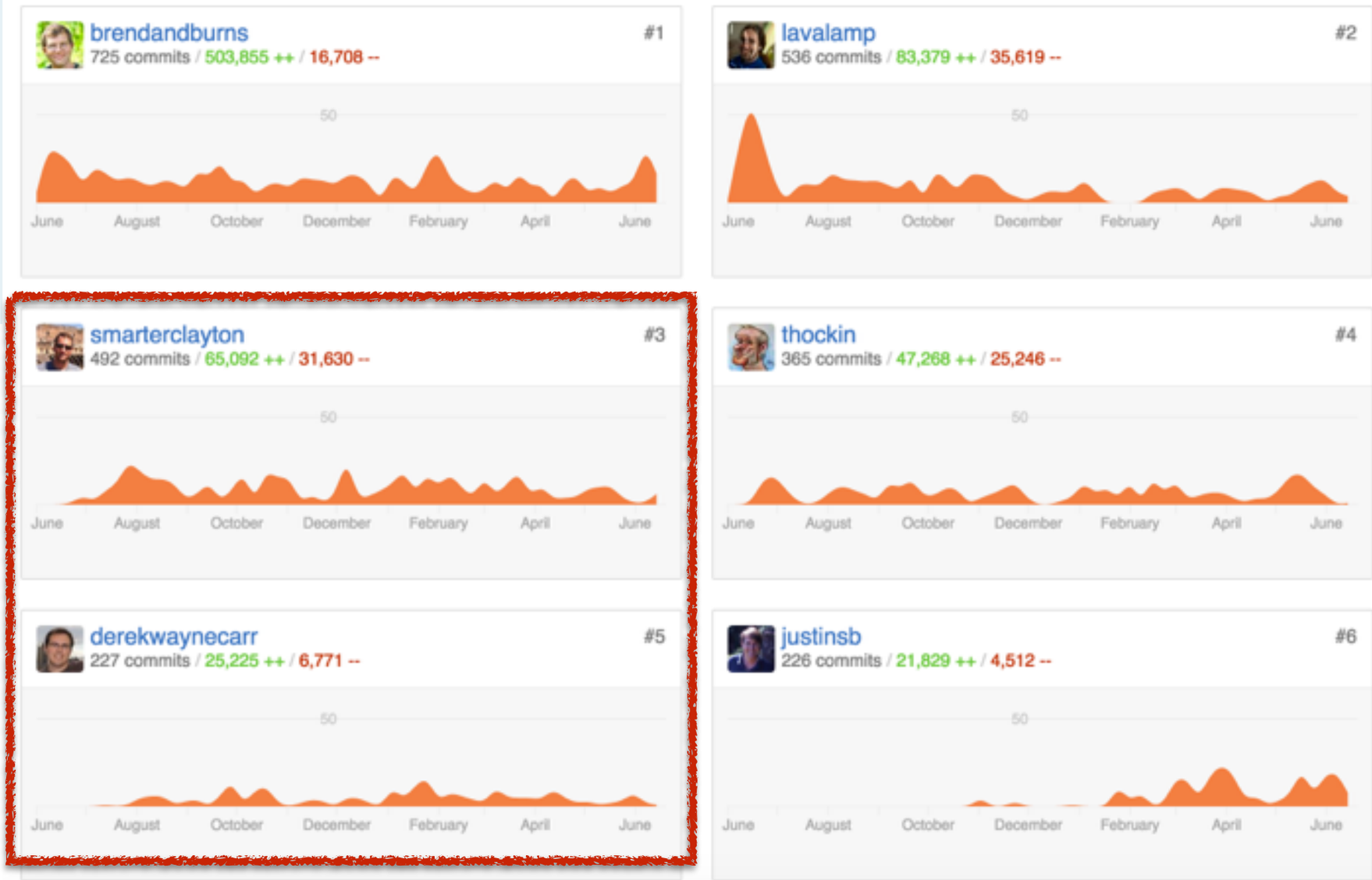
Collect statistics about who wrote kubernetes

gitdm (git data mine) is a tool written by Jonathan Corbet at LWN which he uses to do his 'who wrote the kernel' articles. I spent a couple of minutes to run it against kube. Some intersting results...

```
Developers with the most changesets
Brendan Burns                 643 (10.3%)
Daniel Smith                  485 (7.8%)
Clayton Coleman               453 (7.3%)
Tim Hockin                    312 (5.0%)
derekwaynecarr                209 (3.4%)

Developers with the most changed lines
Brendan Burns                455888 (18.0%)
Nan Monnand Deng             379610 (15.0%)
Yifan Gu                     219191 (8.7%)
Patrick                      169265 (6.7%)
nikhiljindal                 124915 (4.9%)

Top changeset contributors by employer
"Google"                     2562 (58.3%)
"Red Hat"                    1252 (20.5%)
(Unknown)                    899 (14.7%)
"CoreOS"                     144 (2.4%)
"FathomDB"                   142 (2.3%)
```



Arquillian Cube

- Controls the lifecycle of Docker images as part of test cycle
- Uses Docker REST API to talk to container
- Use JBoss EAP/WildFly remote adapter (inside container)

1 Select a container

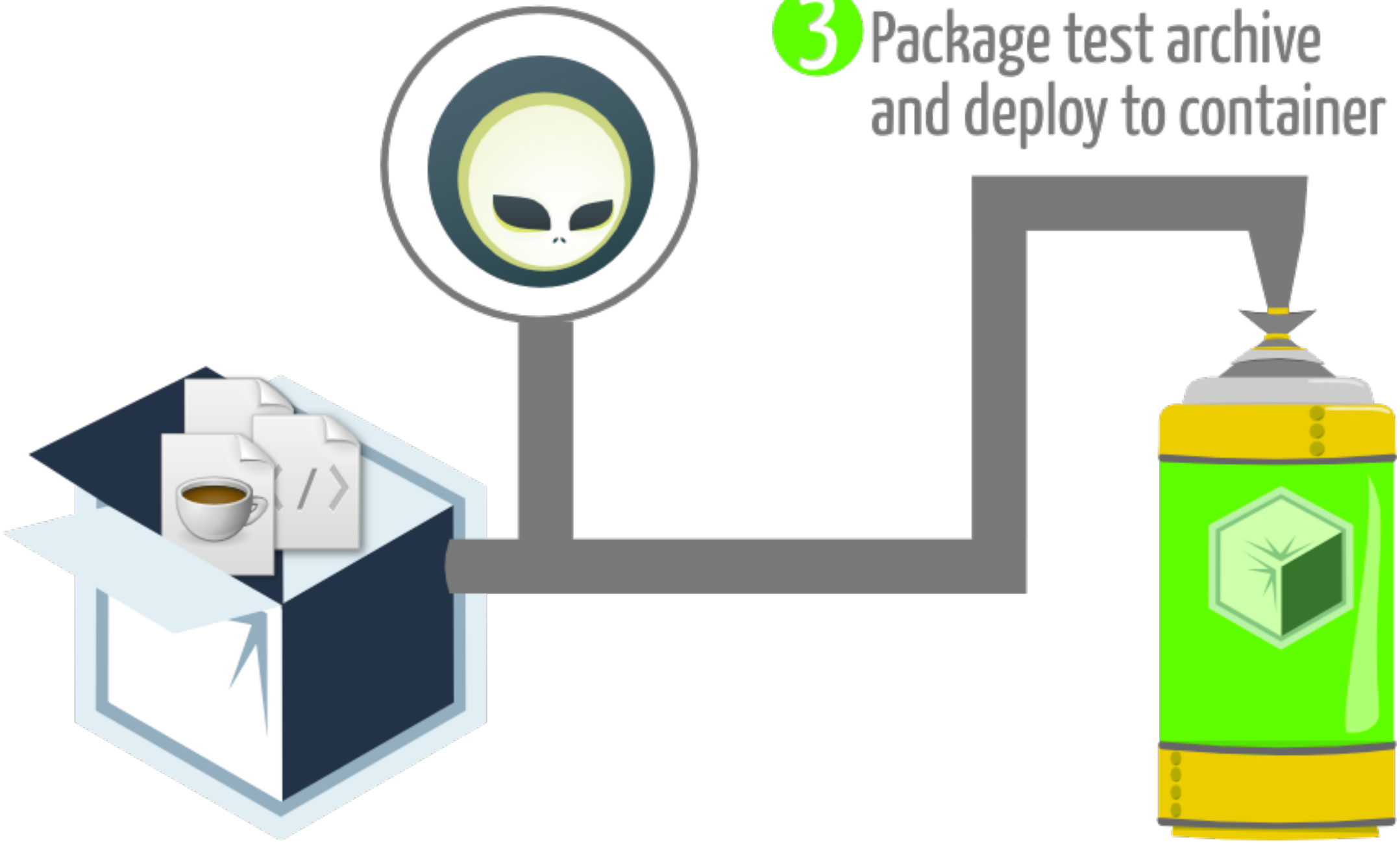
Arquillian

2 Start or connect to a container

redhat

Arquillian

3 Package test archive and deploy to container

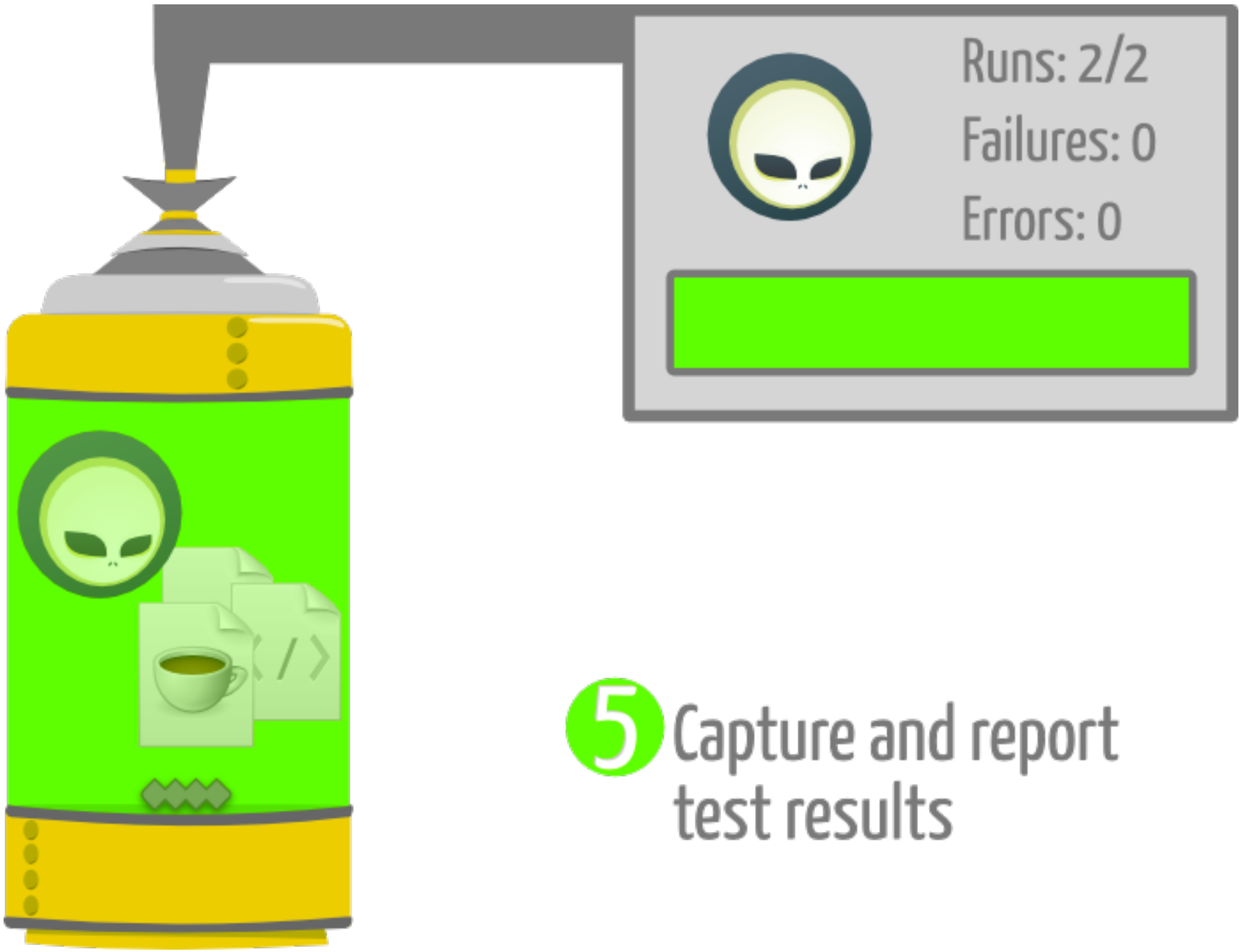


redhat

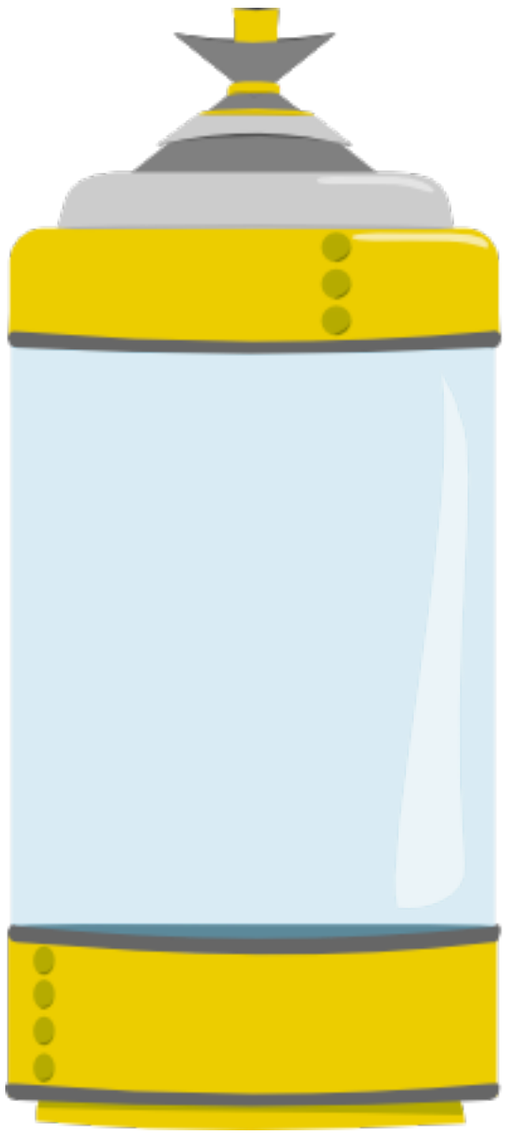
Arquillian

4 Run test in-container

redhat



5 Capture and report test results

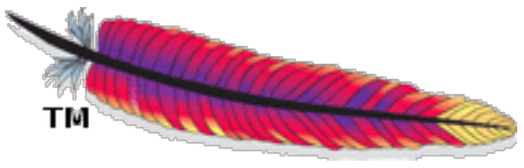


6 Undeploy test archive and stop or disconnect from container

pom.xml

```
<dependency>
    <groupId>org.arquillian.cube</groupId>
    <artifactId>arquillian-cube-docker</artifactId>
    <version>${org.arquillian.cube.version}</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.wildfly</groupId>
    <artifactId>wildfly-arquillian-container-remote</artifactId>
    <version>${org.wildfly}</version>
    <scope>test</scope>
</dependency>
```

Application Operating Environment



RED HAT® JBOSS®
ENTERPRISE
APPLICATION PLATFORM

RED HAT® JBOSS®
ENTERPRISE
APPLICATION PLATFORM

Dev/Test/Prod Environments

- Simple to setup
- Repeatable - no impedance mismatch
- Isolated
- Disposable
- Lightweight

More Details

- Hands-on Lab: Docker for Java Developers
  - Thursday, Jun 25, 10:30am
- Using OpenShift and PaaS to accelerate DevOps & Continuous Delivery
  - Wednesday, Jun 24, 10:30am