CONTINUOUS DELIVERY WITH DOCKER CONTAINERS AND JAVA EE

Markus Eisele
markus@jboss.org

@myfear
blog.eisele.net

THE WORLD WE LIVE IN TODAY

Customers and consumers

- Ubiquitous access to data and services
- Impatient, want everything NOW
- Increased QoS expectations

Businesses

- New opportunities and markets
- Threat of being disrupted, intense competition
- Small time frames to get products and services out

TRADITIONAL SOFTWARE DELIVERY ENVIRONMENT

TYPICAL ASSUMPTIONS AND EXPECTATIONS

Software should never break.

Ops teams are not required in application design discussions.

Production environments are provisioned/ through a mostly manual process.

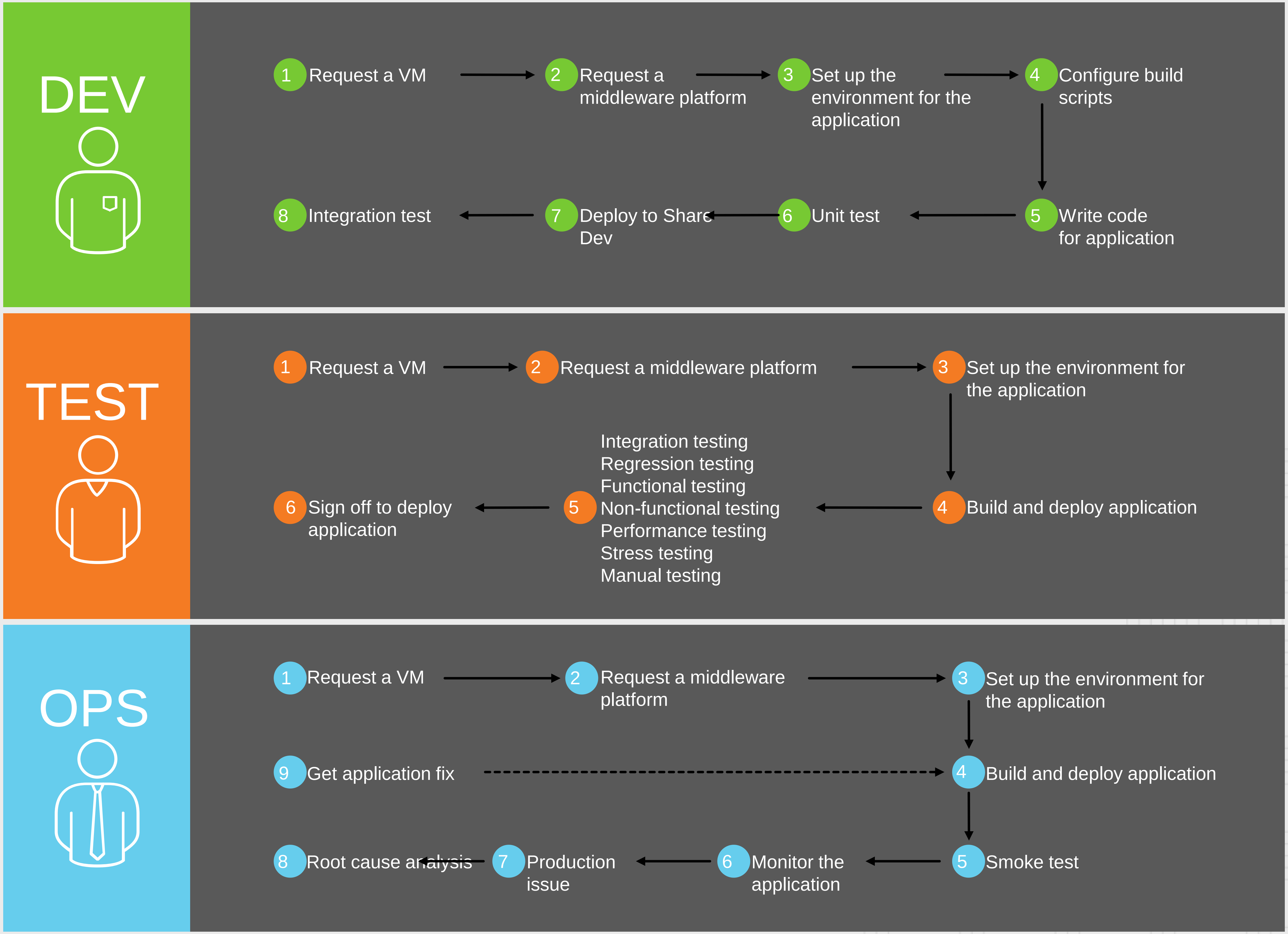
Developers should not have any access to the production environment.

You have to give a lot of lead time for getting an application environment.

An application is deployed to production after all development is complete.

Deployments are a headache—software is deployed using a mostly manual process.

We cannot keep deploying code to production on a regular basis.



DEV

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Configure build scripts

5 Write code for application

6 Unit test

7 Deploy to Share Dev

8 Integration test

TEST

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Integration testing
Regression testing
Functional testing
Non-functional testing
Performance testing
Stress testing
Manual testing

6 Sign off to deploy application

OPS

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Smoke test

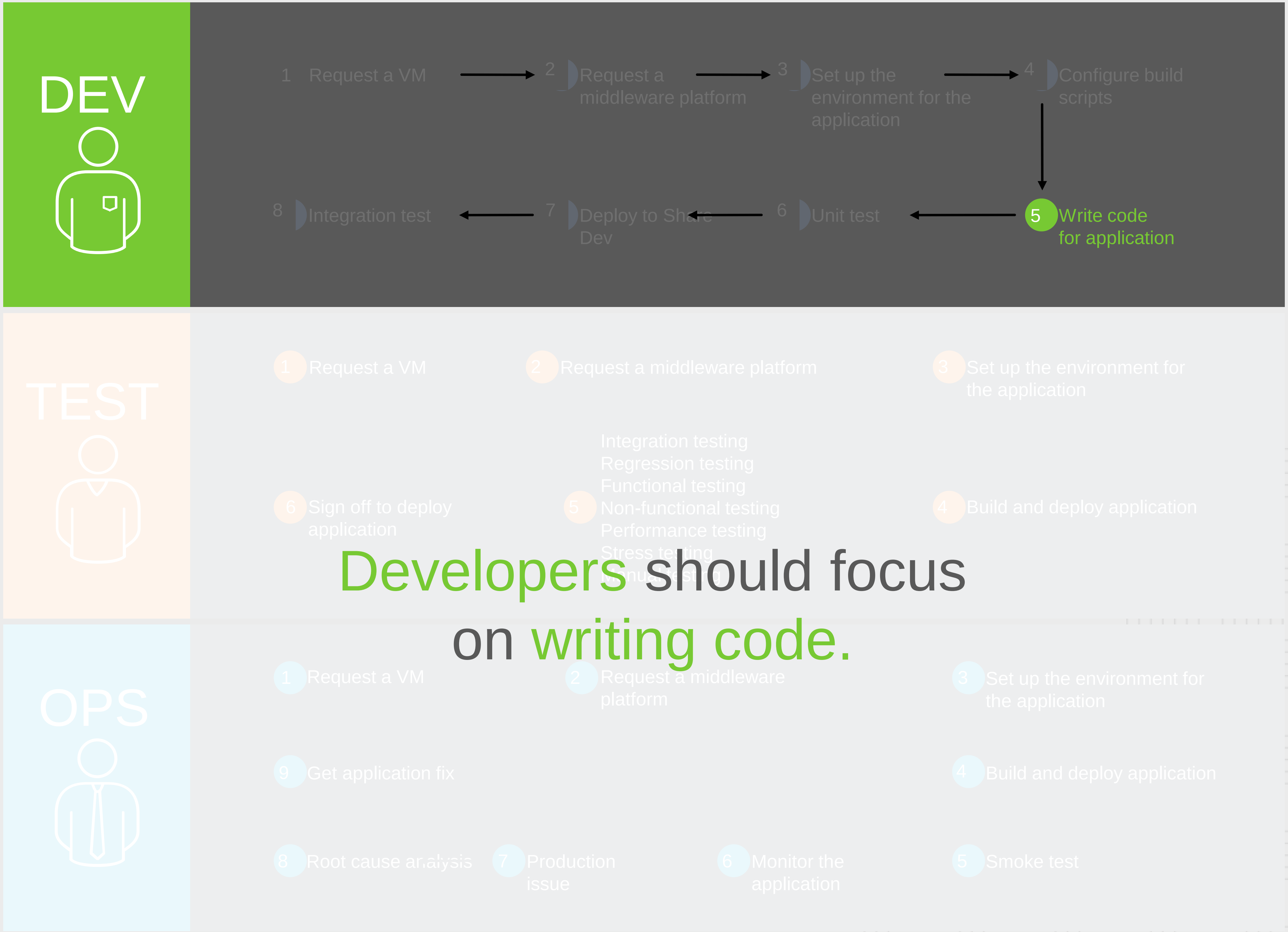
6 Monitor the application

7 Production issue

8 Root cause analysis

9 Get application fix

Developers should focus on writing code.



DEV

1 Request a VM → 2 Request a middleware platform → 3 Set up the environment for the application → 4 Configure build scripts → 5 Write code for application → 6 Unit test → 7 Deploy to Share Dev → 8 Integration test

TEST

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Integration testing
Regression testing
Functional testing
Non-functional testing
Performance testing
Stress testing
Manual testing

6 Sign off to deploy application

OPS

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Smoke test

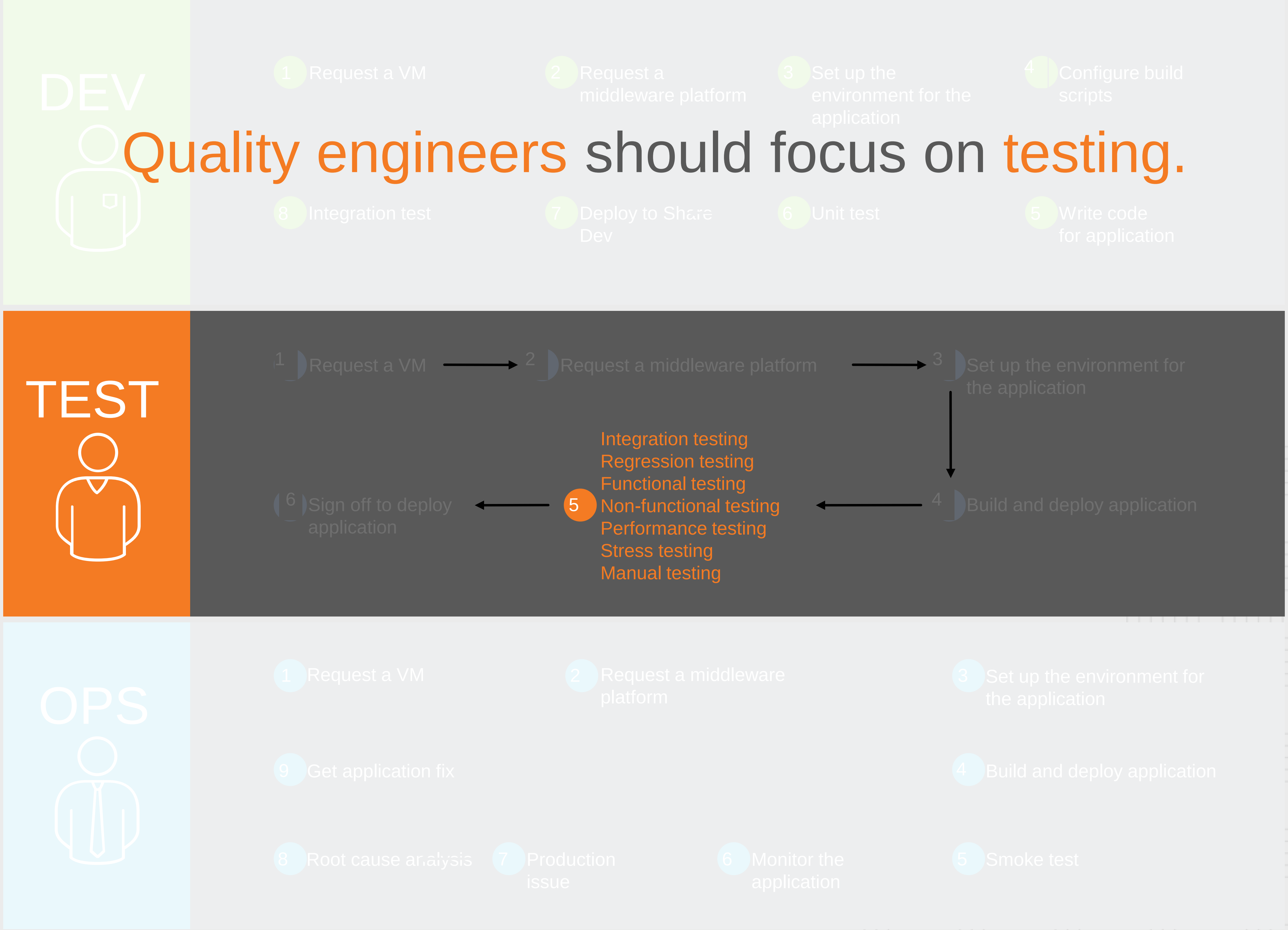
6 Monitor the application

7 Production issue

8 Root cause analysis

9 Get application fix

Quality engineers should focus on testing.



DEV

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Configure build scripts

5 Write code for application

6 Unit test

7 Deploy to Share Dev

8 Integration test

TEST

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Integration testing
Regression testing
Functional testing
Non-functional testing
Performance testing
Stress testing
Manual testing

6 Sign off to deploy application

OPS

1 Request a VM

2 Request a middleware platform

3 Set up the environment for the application

4 Build and deploy application

5 Smoke test

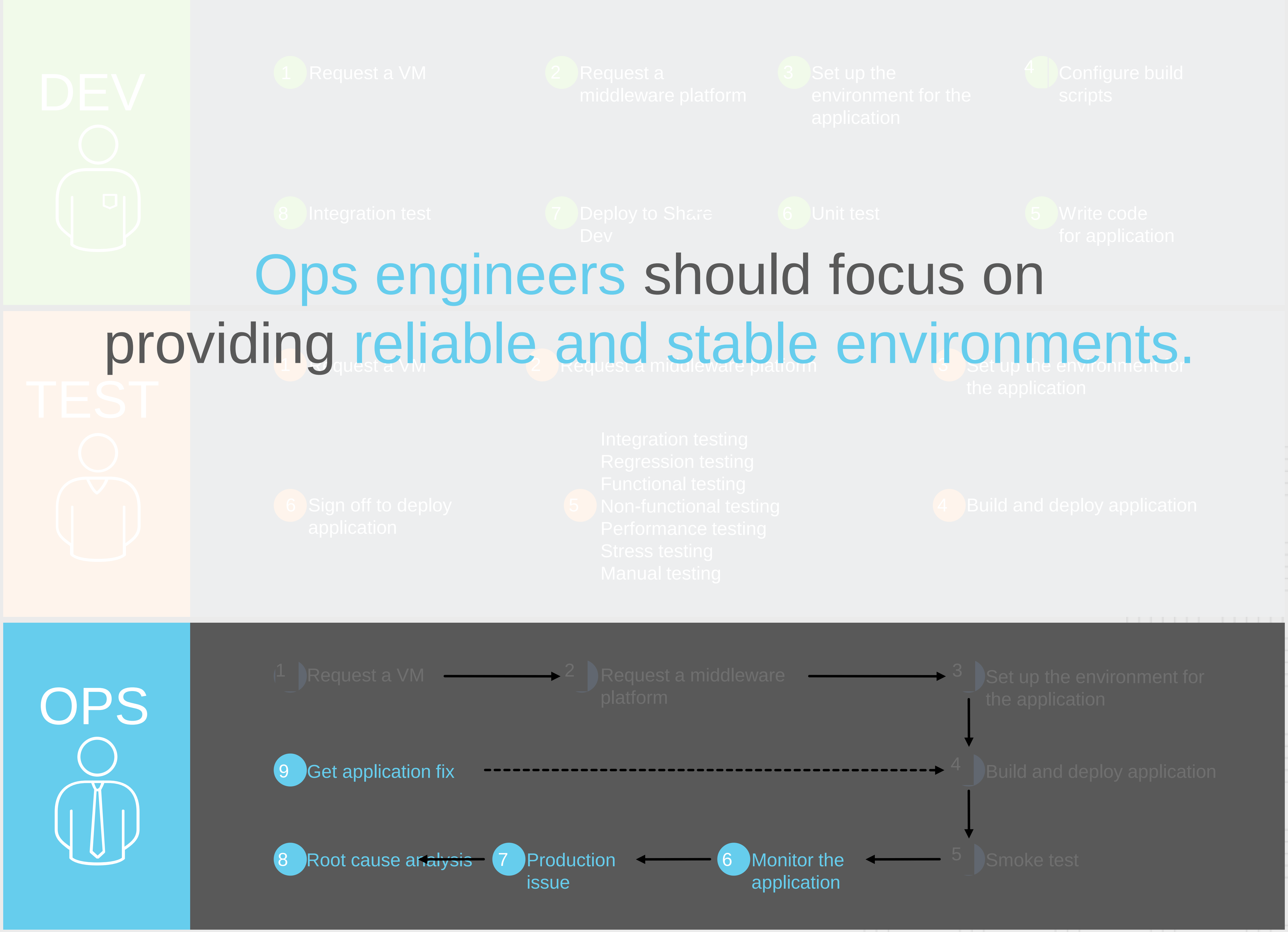
6 Monitor the application

7 Production issue

8 Root cause analysis

9 Get application fix

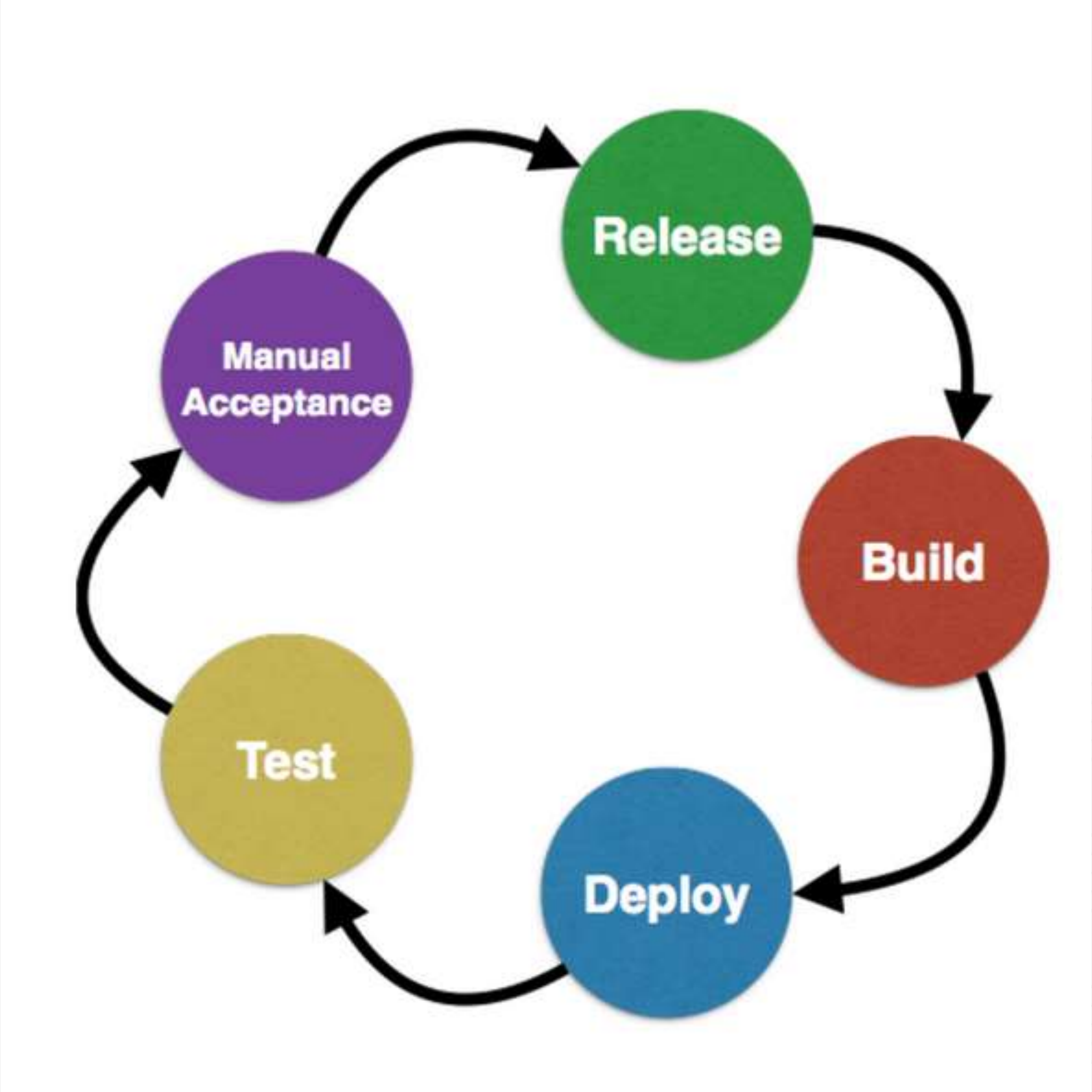
Ops engineers should focus on providing reliable and stable environments.


DEV
1 Request a VM
2 Request a middleware platform
3 Set up the environment for the application
4 Configure build scripts
5 Write code for application
6 Unit test
7 Deploy to Share Dev
8 Integration test
TEST
1 Request a VM
2 Request a middleware platform
3 Set up the environment for the application
4 Build and deploy application
5 Integration testing
Regression testing
Functional testing
Non-functional testing
Performance testing
Stress testing
Manual testing
6 Sign off to deploy application
OPS
1 Request a VM
2 Request a middleware platform
3 Set up the environment for the application
4 Build and deploy application
5 Smoke test
6 Monitor the application
7 Production issue
8 Root cause analysis
9 Get application fix

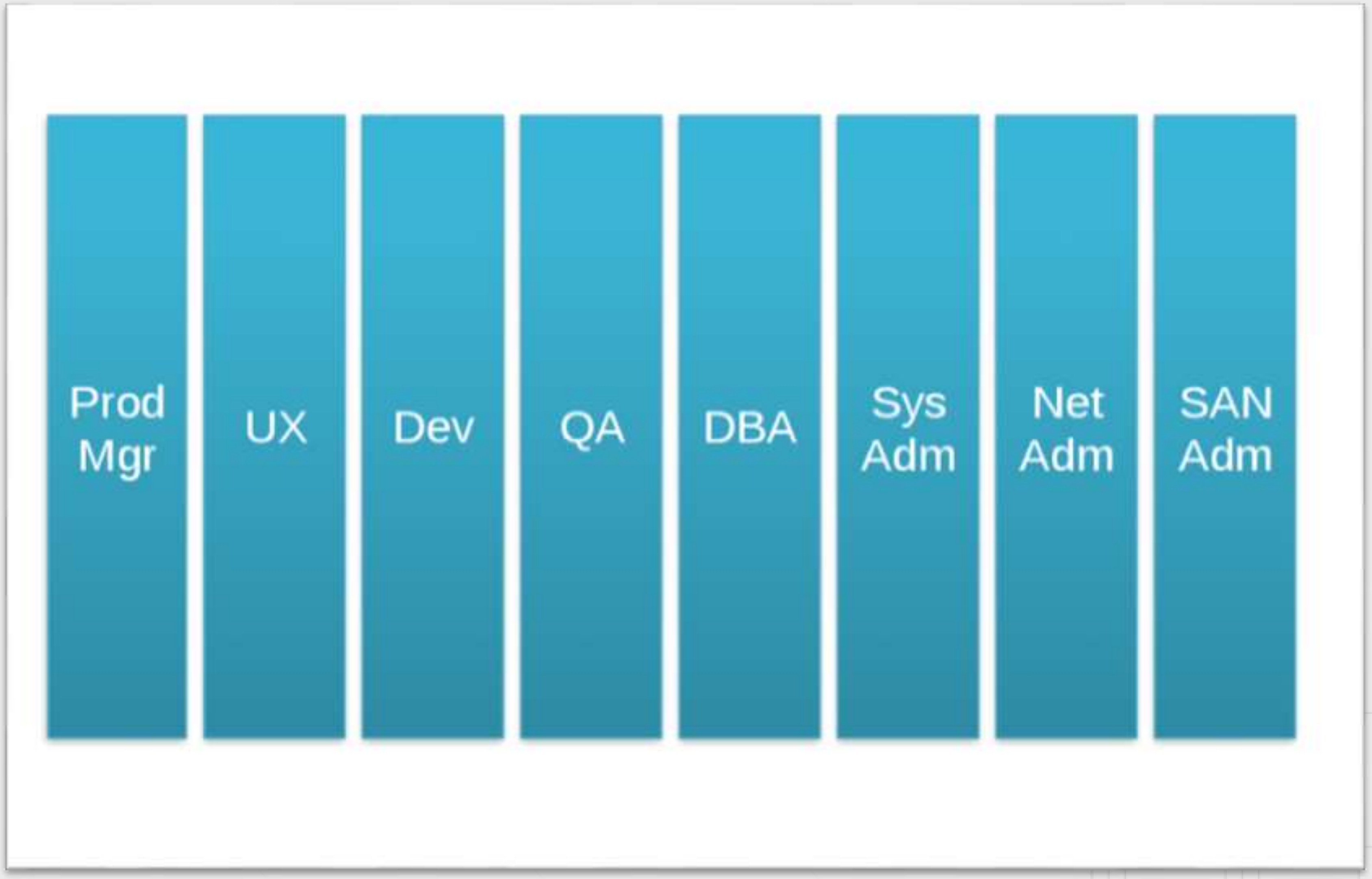
BUT THIS IS SOOOOOOOOOOOOOOO **YESTERDAY.**

CONTINUOUS DELIVERY

- Continuous Integration
- Fail fast and recover
- Self service
- 100% Automation
- Push to Prod
- Proactive Monitoring and Metrics
- Requires fast and Consistent Build and Deploy



TRADITIONAL SILOS



Prod Mgr

UX

Dev

QA

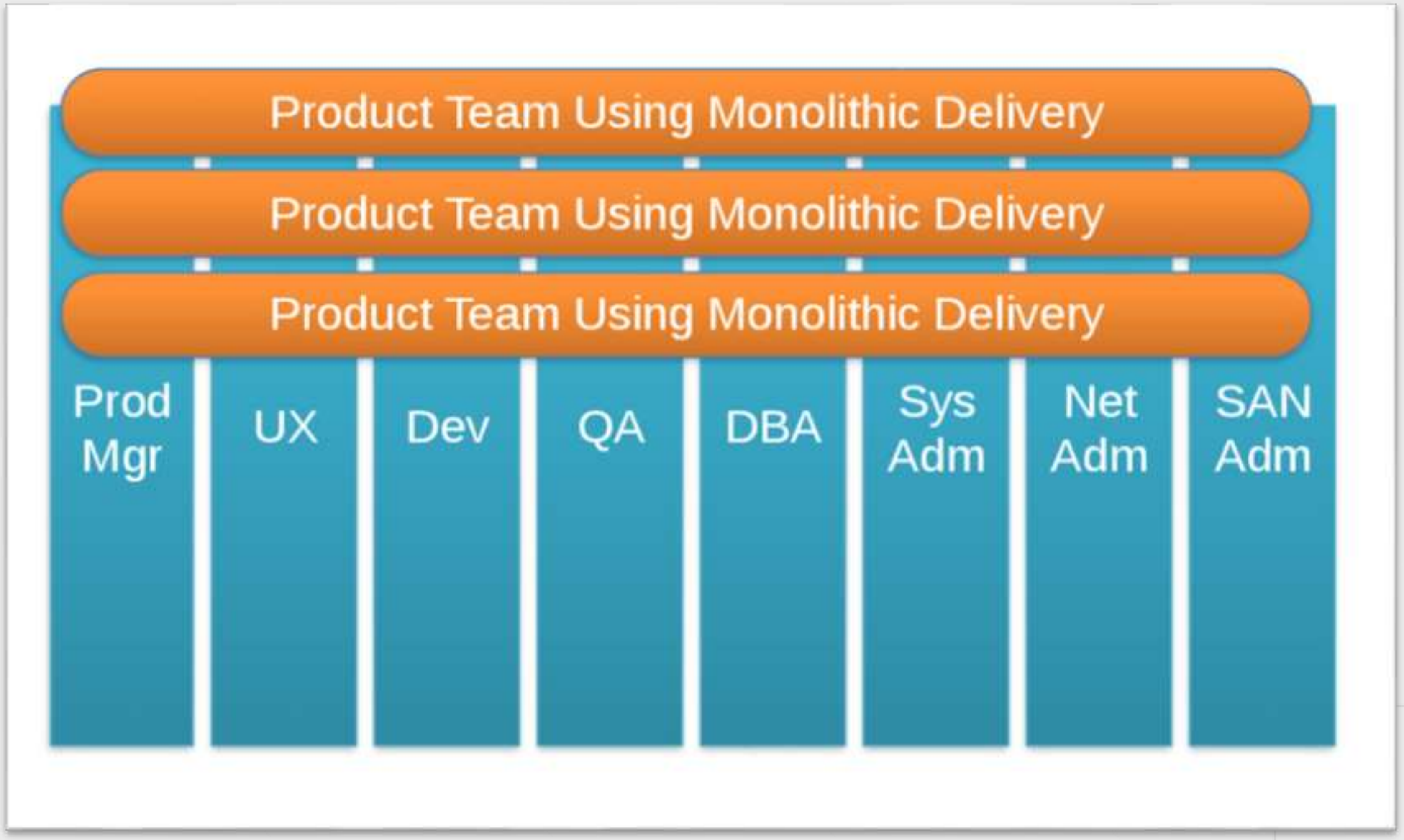
DBA

Sys Adm

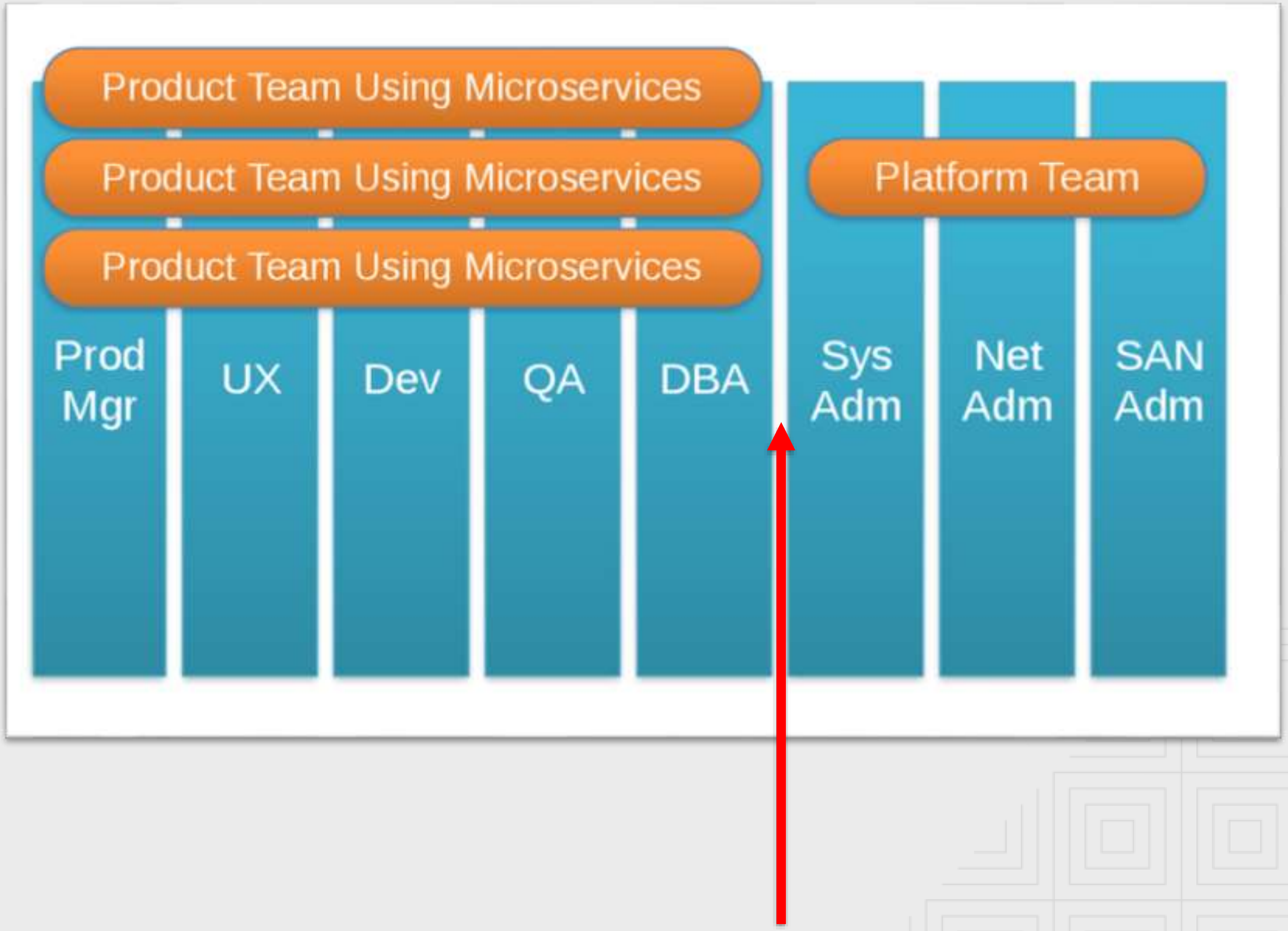
Net Adm

SAN Adm

BREAKING THEM DOWN (THE MONOLITHIC WAY)

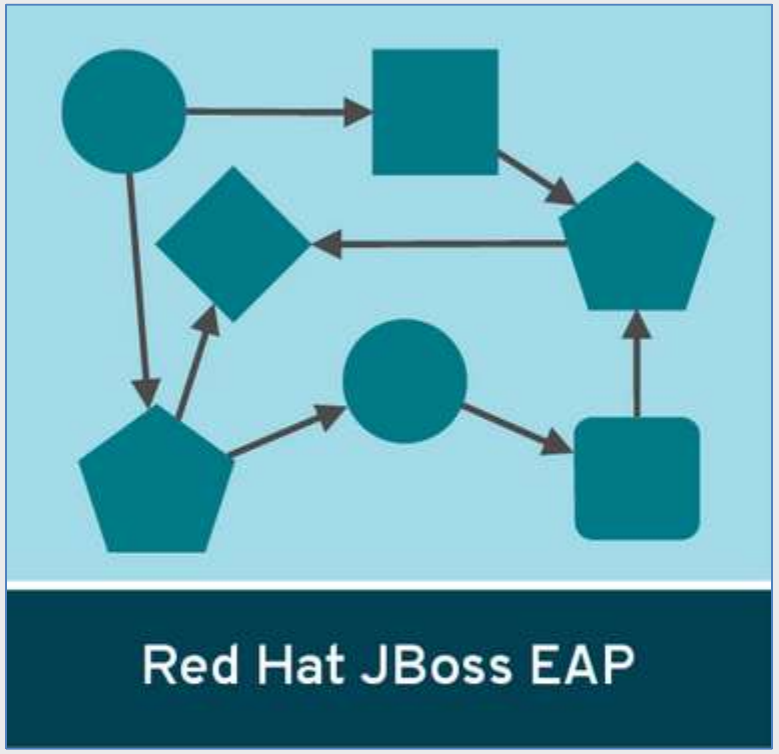

Product Team Using Monolithic Delivery
Product Team Using Monolithic Delivery
Product Team Using Monolithic Delivery
Prod Mgr
UX
Dev
QA
DBA
Sys Adm
Net Adm
SAN Adm

BREAKING THEM DOWN (THE MICROSERVICE WAY)

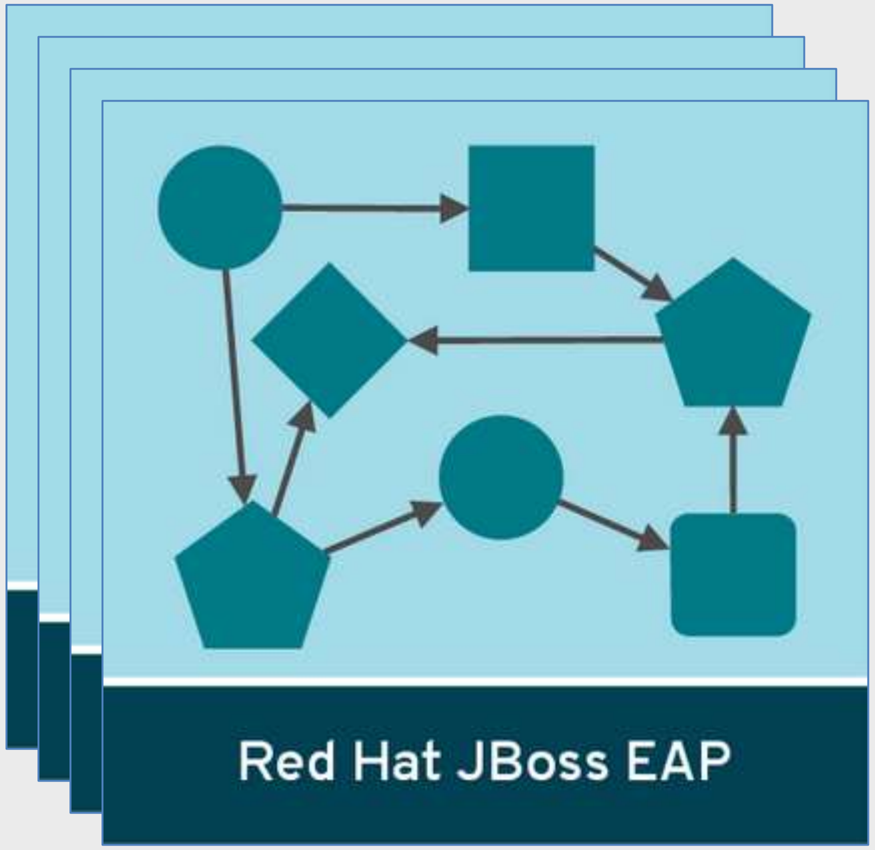

Product Team Using Microservices
Product Team Using Microservices
Product Team Using Microservices
Platform Team
Prod Mgr
UX
Dev
QA
DBA
Sys Adm
Net Adm
SAN Adm

API

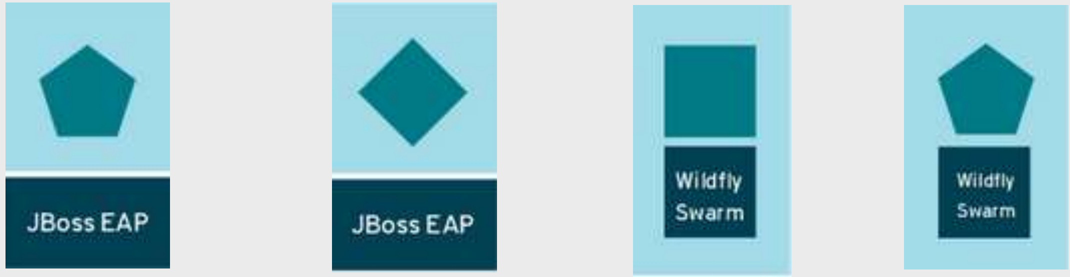
TRADITIONAL ARCHITECTURE



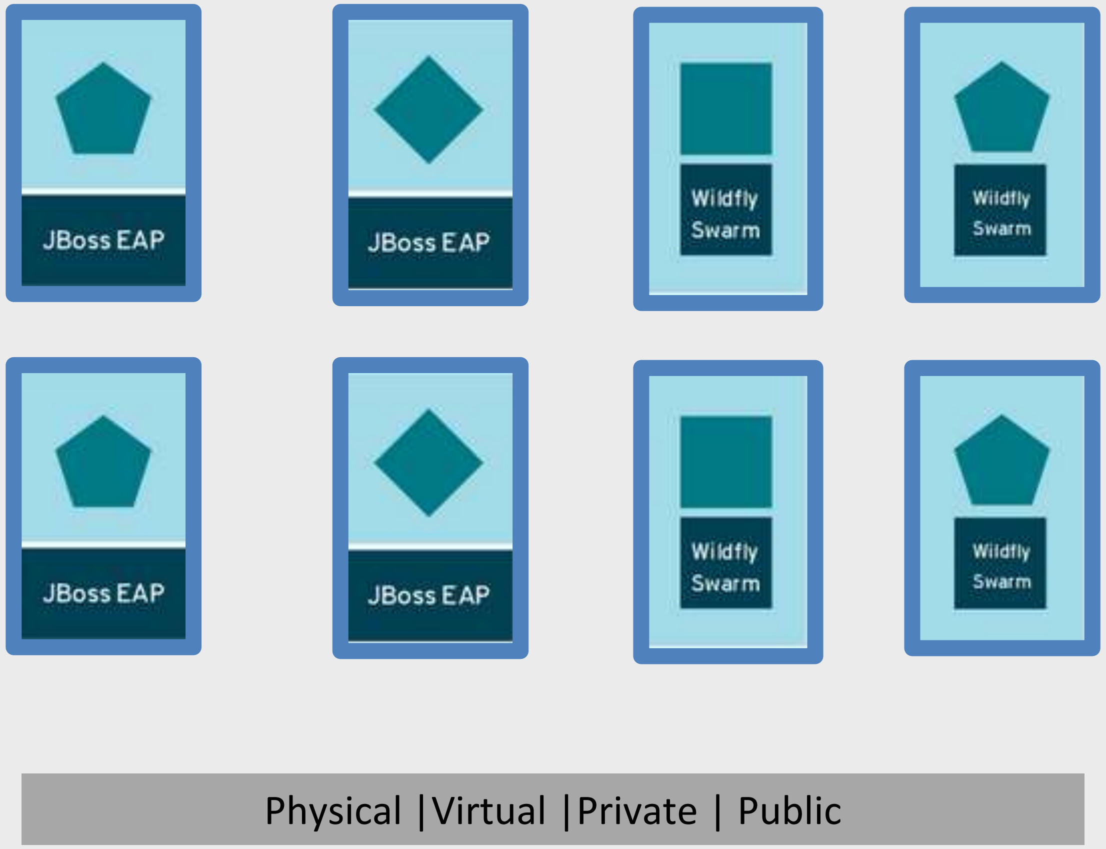
SCALING == SCALING THE COMPLETE STACK



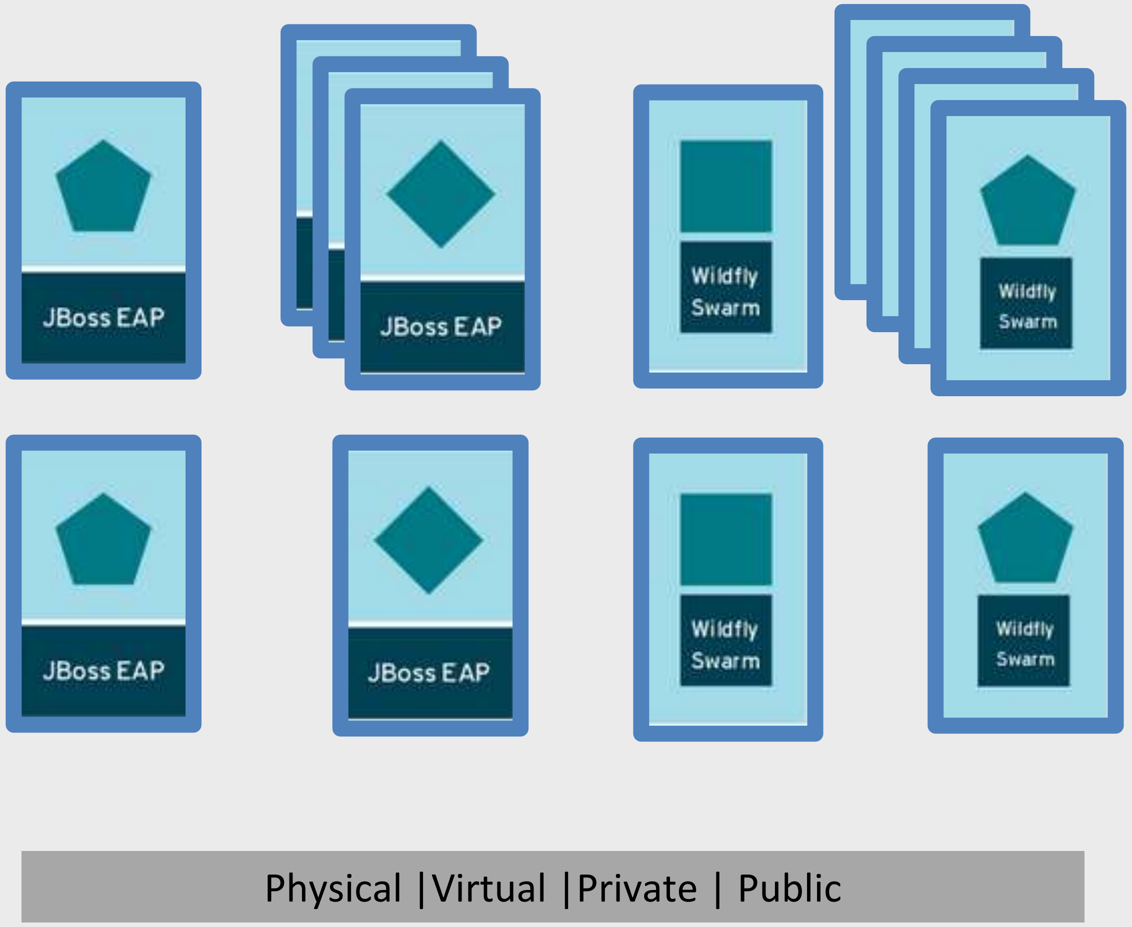
TOMORROWS APPROACH (MICROSERVICES)



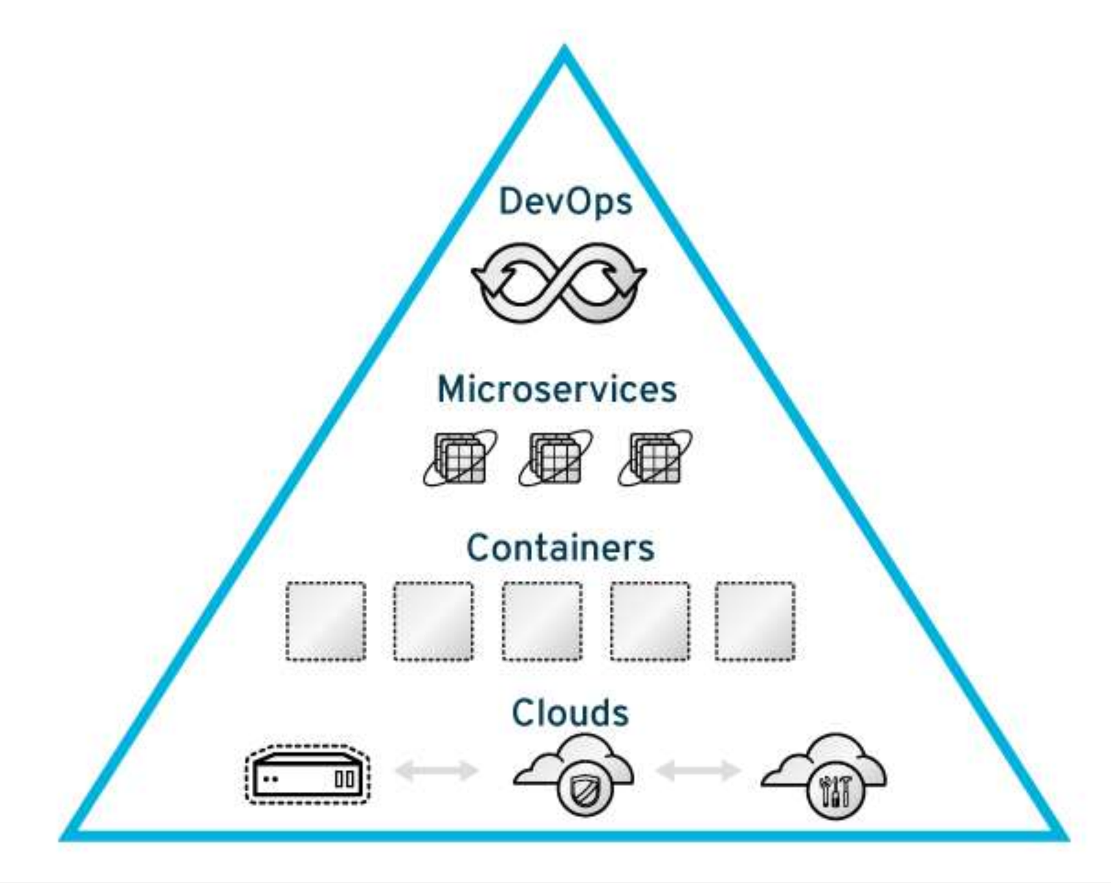
PACKAGE ONCE RUN EVERYWHERE (CONTAINER)



SCALING SERVICES ON DEMAND



PYRAMID OF MODERN APPLICATION DEVELOPMENT



WHAT‘S IN IT FOR JAVA EE DEVELOPERS?

- Developers can focus on the app
- Standardized Development Environment
- Spin-up and Tear-Down of different instances in seconds.
- Easy, centralized configuration
- “Build, Ship and Run: Any App, Anywhere”
- Faster and more predictable delivery

CONTAINERS, WHAT'S SO GREAT?

- Put everything in a box
- It's cheap (compared to special shipments)
- I don't have to care if they are delivered on a ship, a truck or a train
- I can put a look on it and make sure that my integrity is kept
- If a container breaks replace it

CONTAINERS, WHAT ARE THE DRAWBACKS?

- One size doesn’t fit all.
- I hope no one drops my container.
- No one is going to care if I put a sign on it saying fragile.
- I cannot change the routing while in process.
- I’m not in control.

CONTAINER DEPLOYMENT

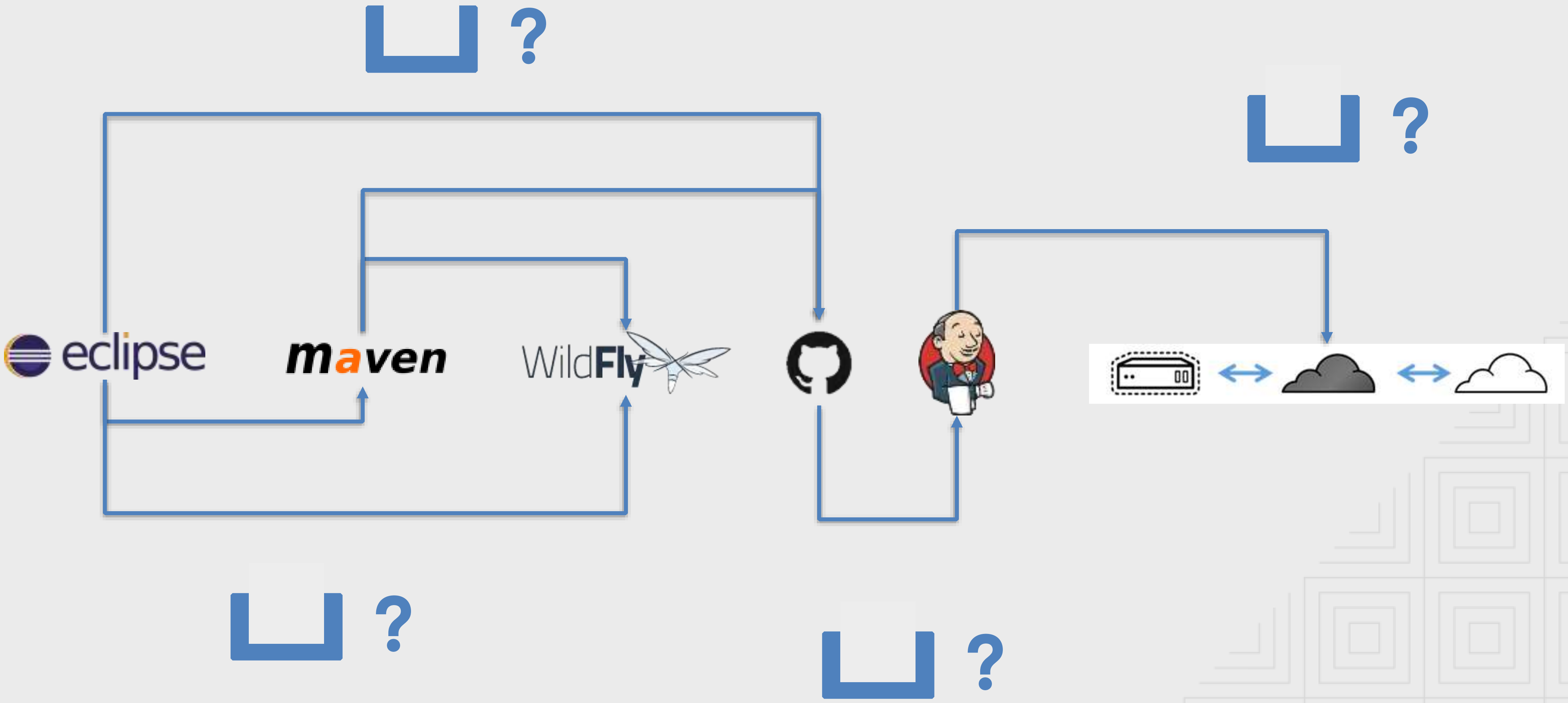
- Many organizations today are not ready to directly deploy the same containers in all environments
- “Build, Ship and Run Any App, Anywhere” (Docker Slogan)
- Requires adopting DevOps principles and Microservices architecture
- Today typical Java EE application still requires small variations between environments.
- E.g. Memory usage, clustering, endpoints, security patches etc
- The key is to minimize the variations.

CAN CONTAINERS BE ABUSED?

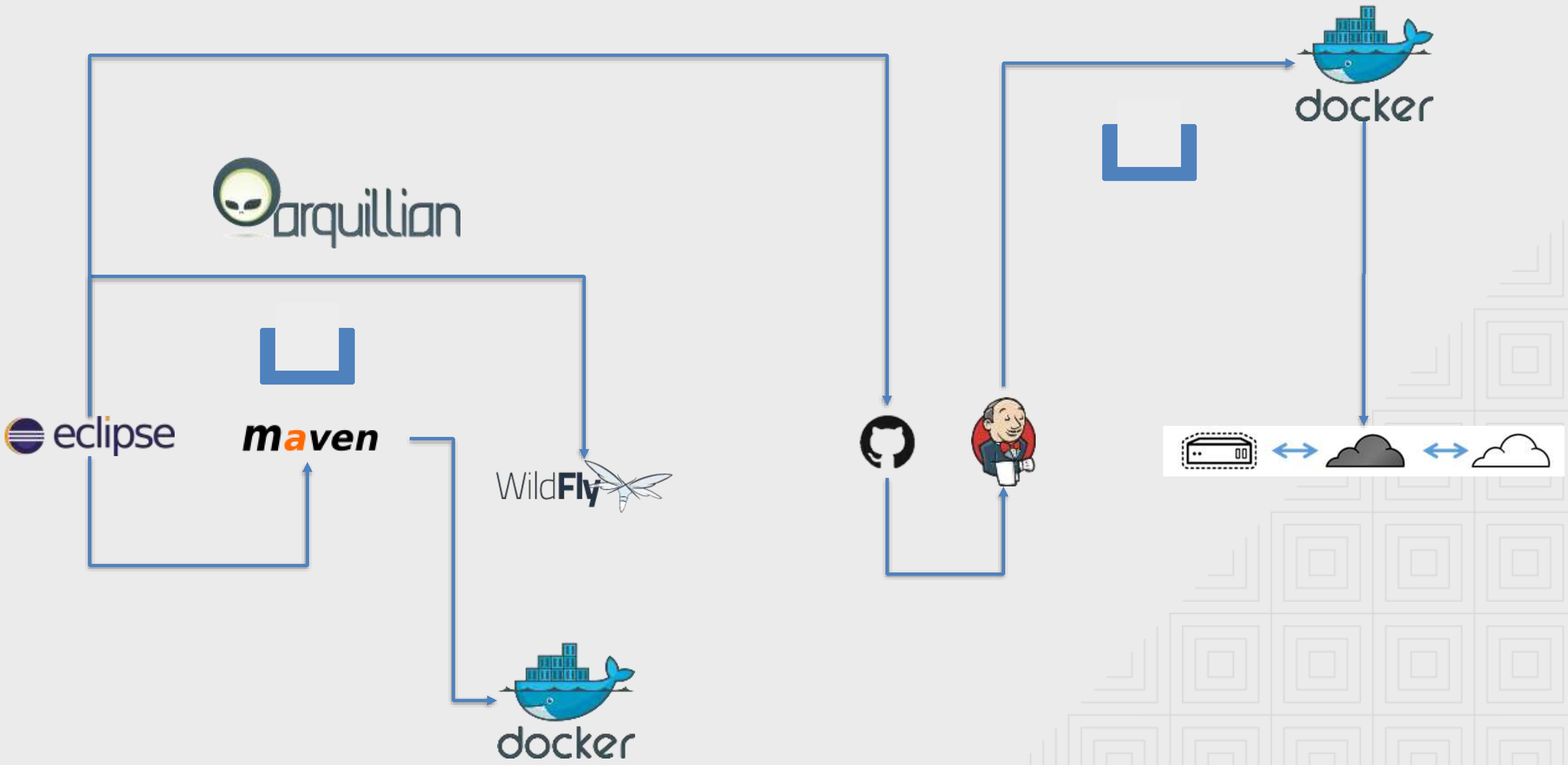
- Containers are immutable
- Release vs Patch
- If it break, replace it. Do not try to fix them.
- Complex environment gets complex to maintain even with containers
- Use a orchestration tool like OpenShift

HOW TO DO IT?

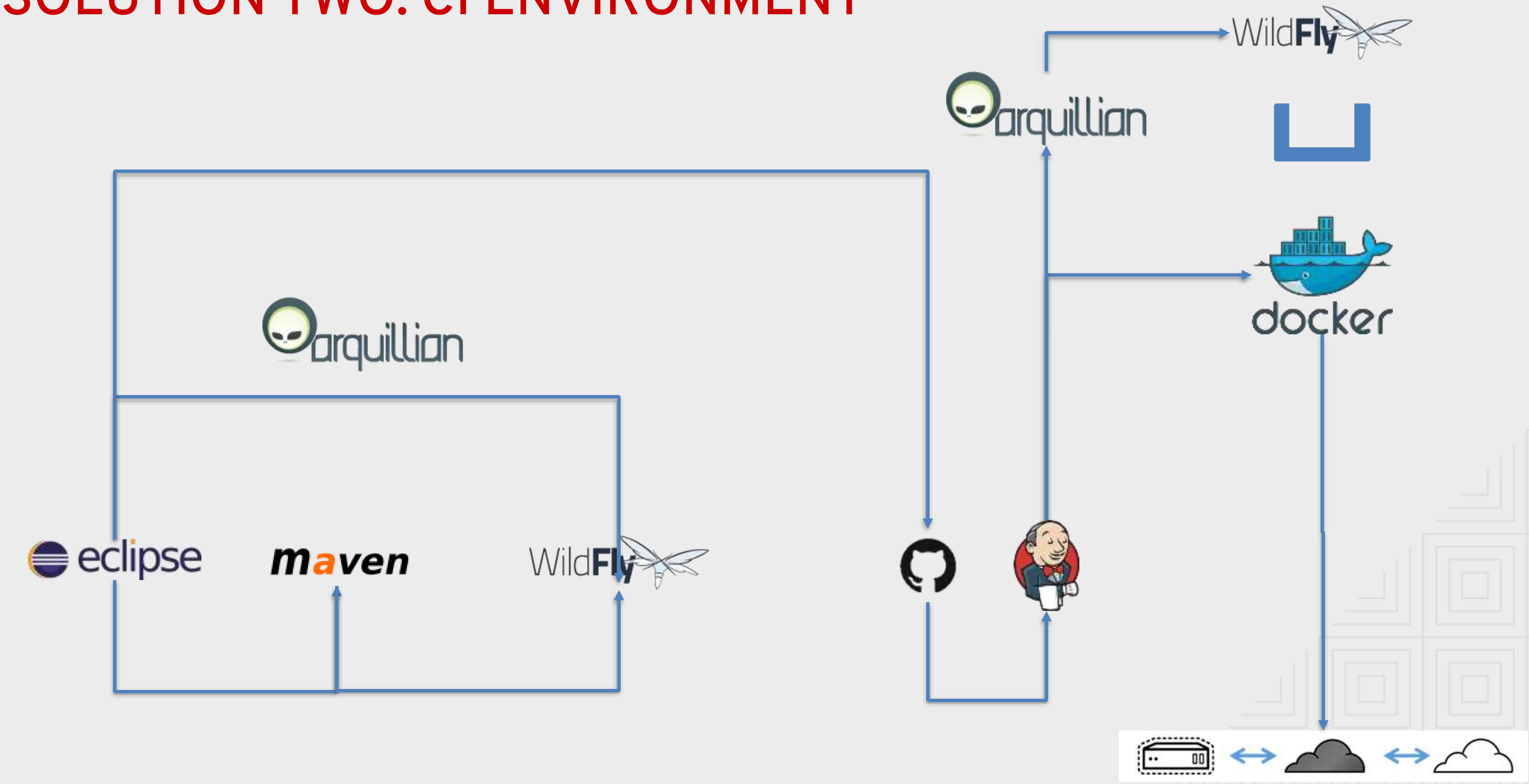
JAVA EE DEVELOPMENT WORKFLOW



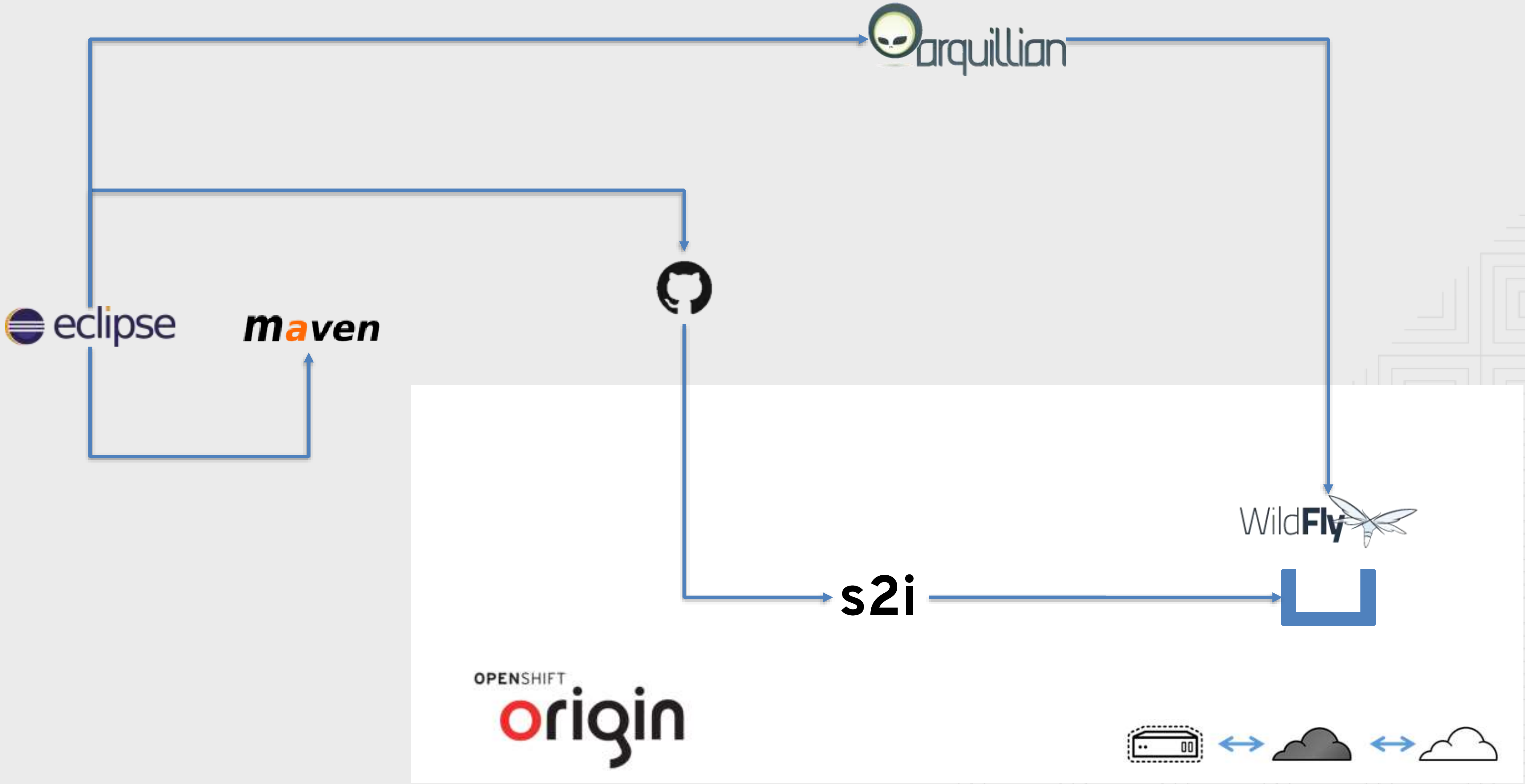
SOLUTION ONE: DEVELOPMENT ENVIRONMENT



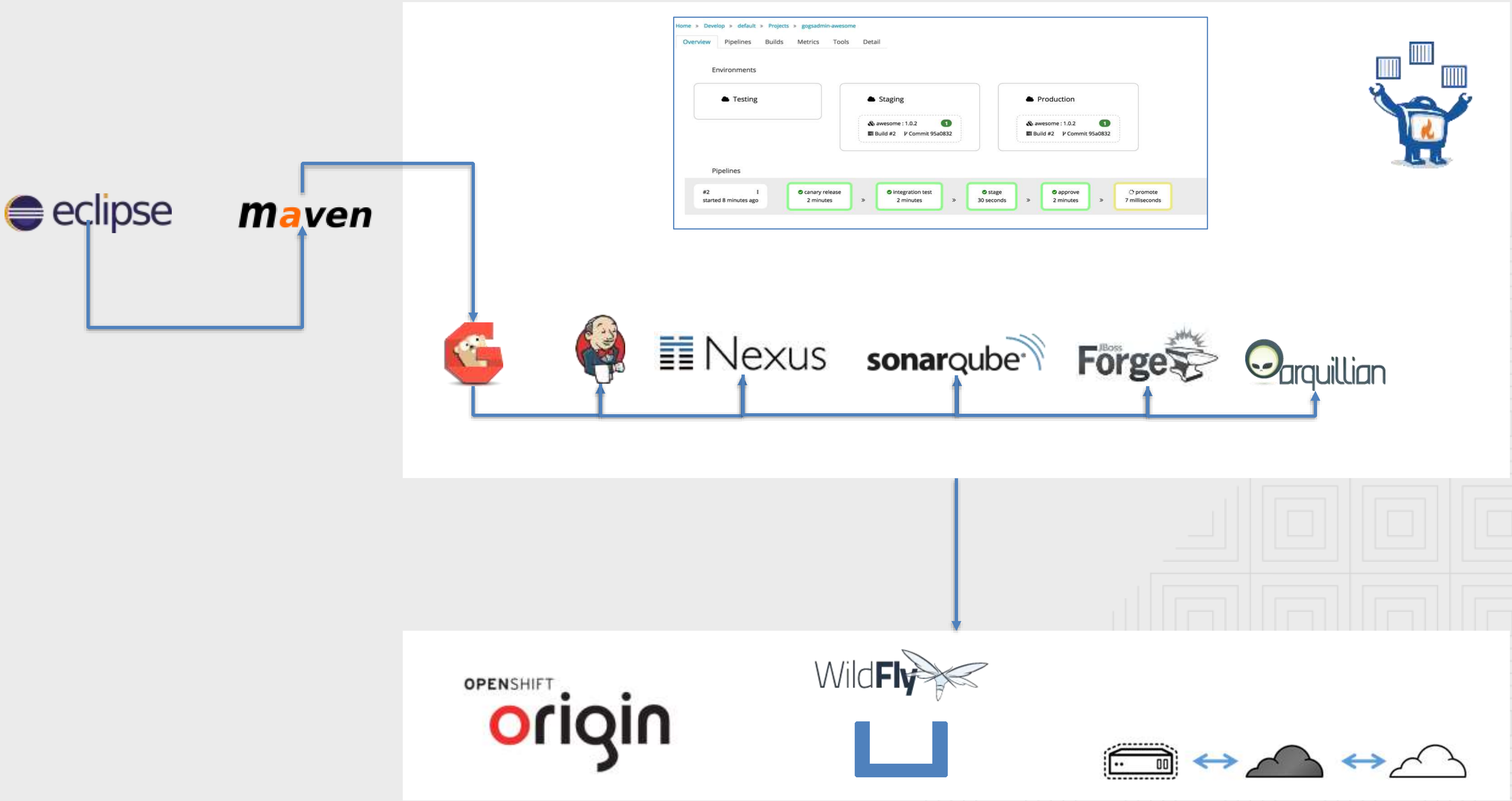
SOLUTION TWO: CI ENVIRONMENT



SOLUTION THREE: CLOUD DEV ENVIRONMENT



SOLUTION FOUR: CLOUD DEVOPS ENVIRONMENT



Home » Develop » default » Projects » gogsadmin-awesome

Overview | Pipelines | Builds | Metrics | Tools | Detail

Environments

Testing

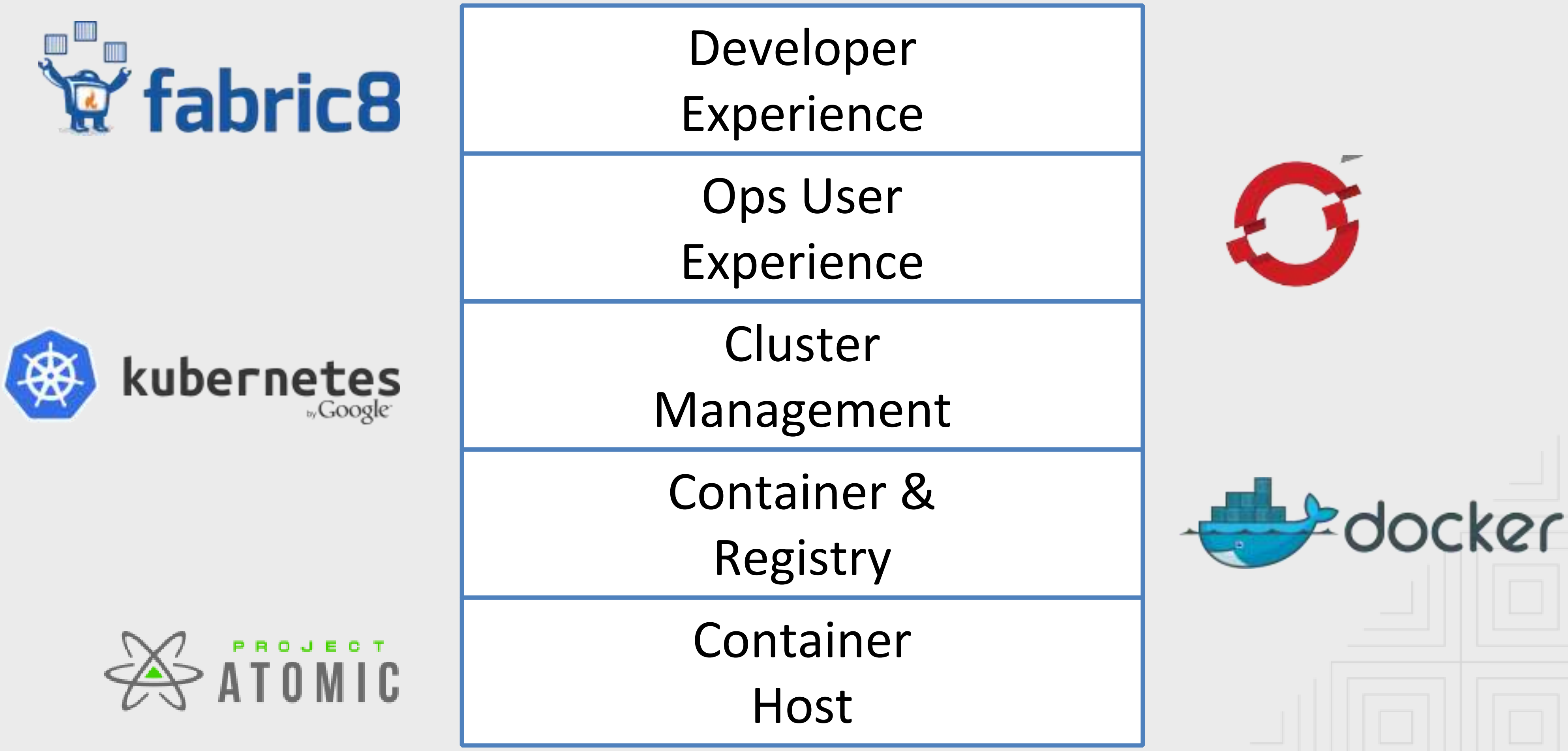
Staging
awesome : 1.0.2 1
Build #2 Commit 95a0832

Production
awesome : 1.0.2 1
Build #2 Commit 95a0832

Pipelines

#2
started 8 minutes ago

✓ canary release
2 minutes
»
✓ integration test
2 minutes
»
✓ stage
30 seconds
»
✓ approve
2 minutes
»
promote
7 milliseconds



Developer Experience

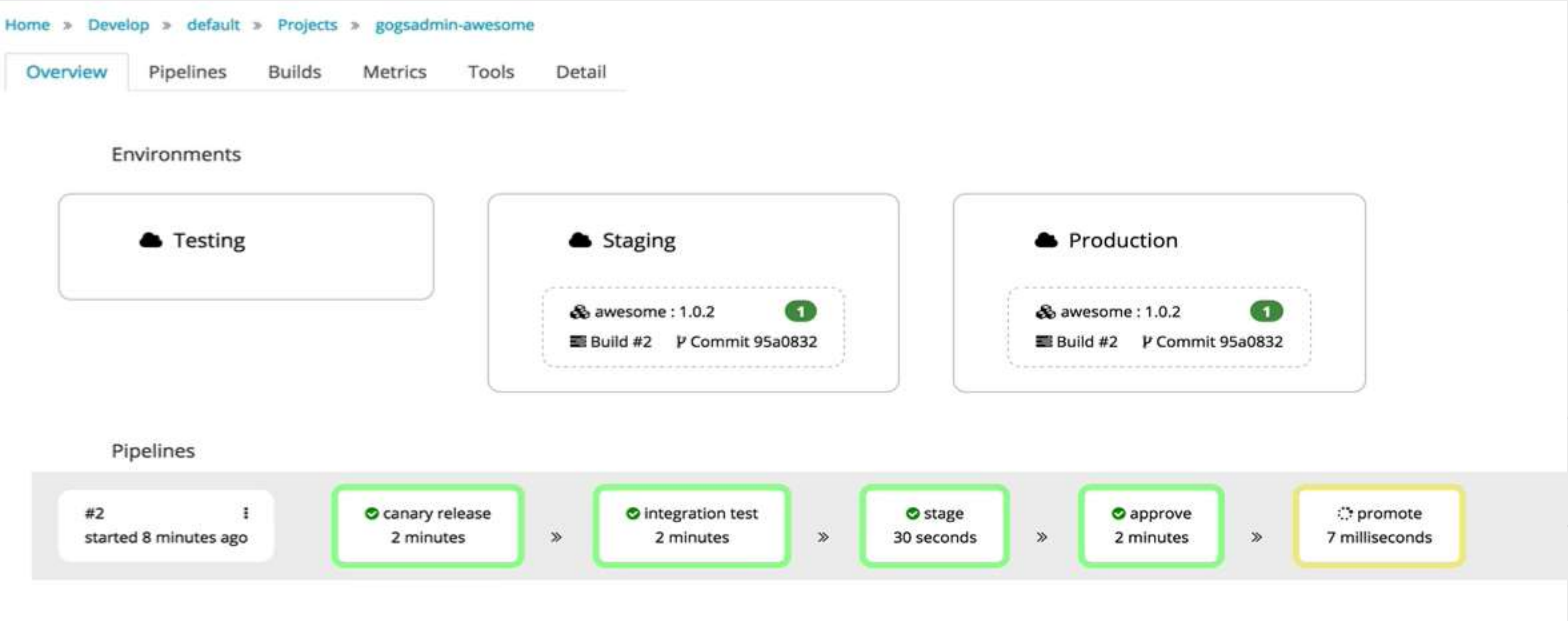
Ops User Experience

Cluster Management

Container & Registry

Container Host

DEVELOPER UI



Home » Develop » default » Projects » gogsadmin-awesome

Overview | Pipelines | Builds | Metrics | Tools | Detail

Environments

Testing

Staging

awesome : 1.0.2 1
Build #2 Commit 95a0832

Production

awesome : 1.0.2 1
Build #2 Commit 95a0832

Pipelines

#2
started 8 minutes ago

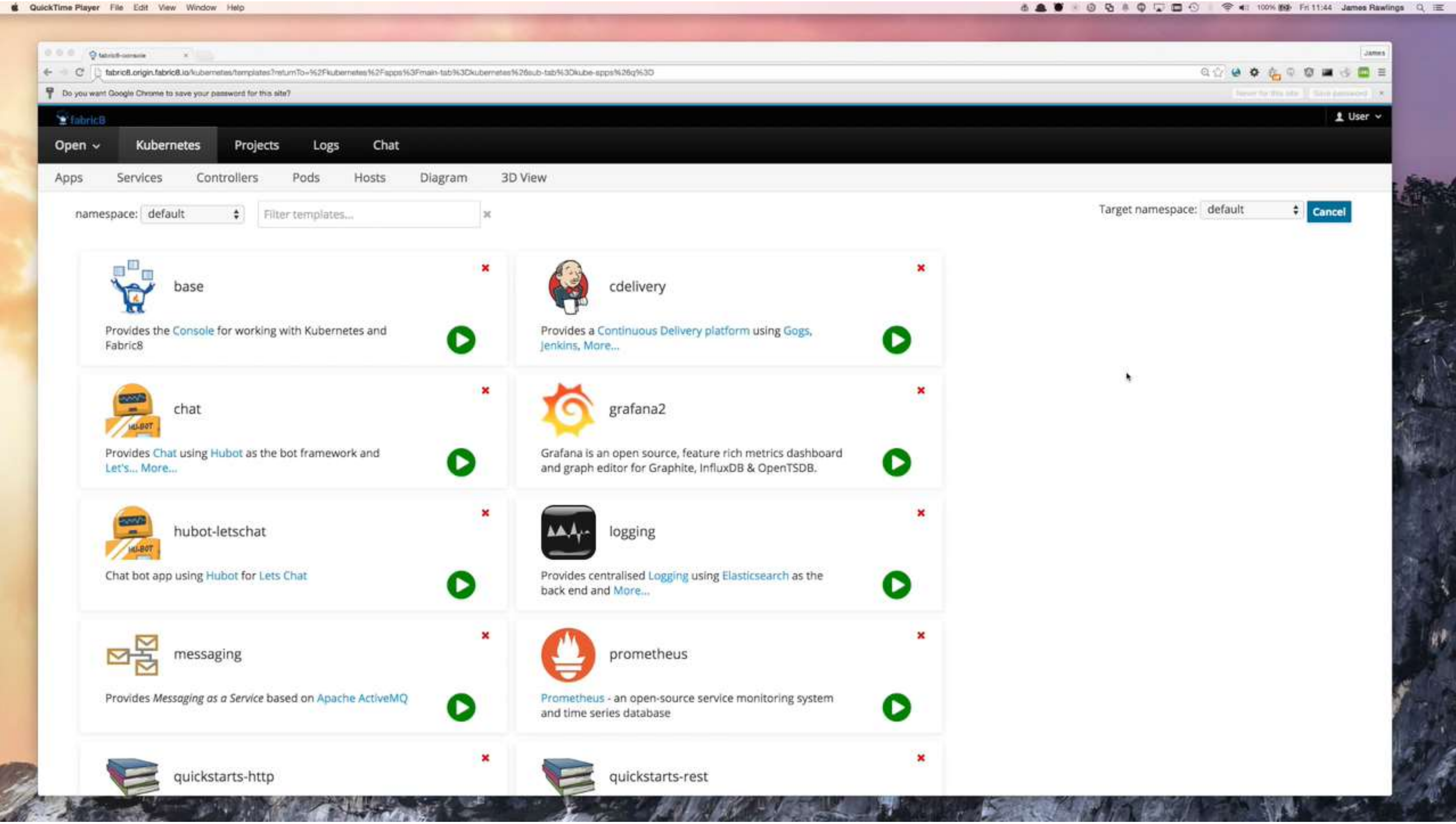
✓ canary release
2 minutes

» ✓ integration test
2 minutes

» ✓ stage
30 seconds

» ✓ approve
2 minutes

» promote
7 milliseconds


QuickTime Player File Edit View Window Help
Fri 11:44 James Rawlings
fabric8-console
fabric8.origin.fabric8.io/kubernetes/templates?returnTo=%2Fkubernetes%2Fapps%3Fmain-tab%3Dkubernetes%26sub-tab%3Dkube-apps%26q%3D
Do you want Google Chrome to save your password for this site?
fabric8
User
Open
Kubernetes
Projects
Logs
Chat
Apps
Services
Controllers
Pods
Hosts
Diagram
3D View
Cancel

namespace: default

Filter templates...

Target namespace: default

base

Provides the Console for working with Kubernetes and Fabric8

cdelivery

Provides a Continuous Delivery platform using Gogs, Jenkins, More...

chat

Provides Chat using Hubot as the bot framework and Let's... More...

grafana2

Grafana is an open source, feature rich metrics dashboard and graph editor for Graphite, InfluxDB & OpenTSDB.

hubot-letschat

Chat bot app using Hubot for Lets Chat

logging

Provides centralised Logging using Elasticsearch as the back end and More...

messaging

Provides *Messaging as a Service* based on Apache ActiveMQ

prometheus

Prometheus - an open-source service monitoring system and time series database

quickstarts-http

quickstarts-rest

MORE RESOURCES AND READINGS

- Fabric8 Guide

  http://fabric8.io/guide/overview.html

- Running WildFly on Fabric8 / OpenShift

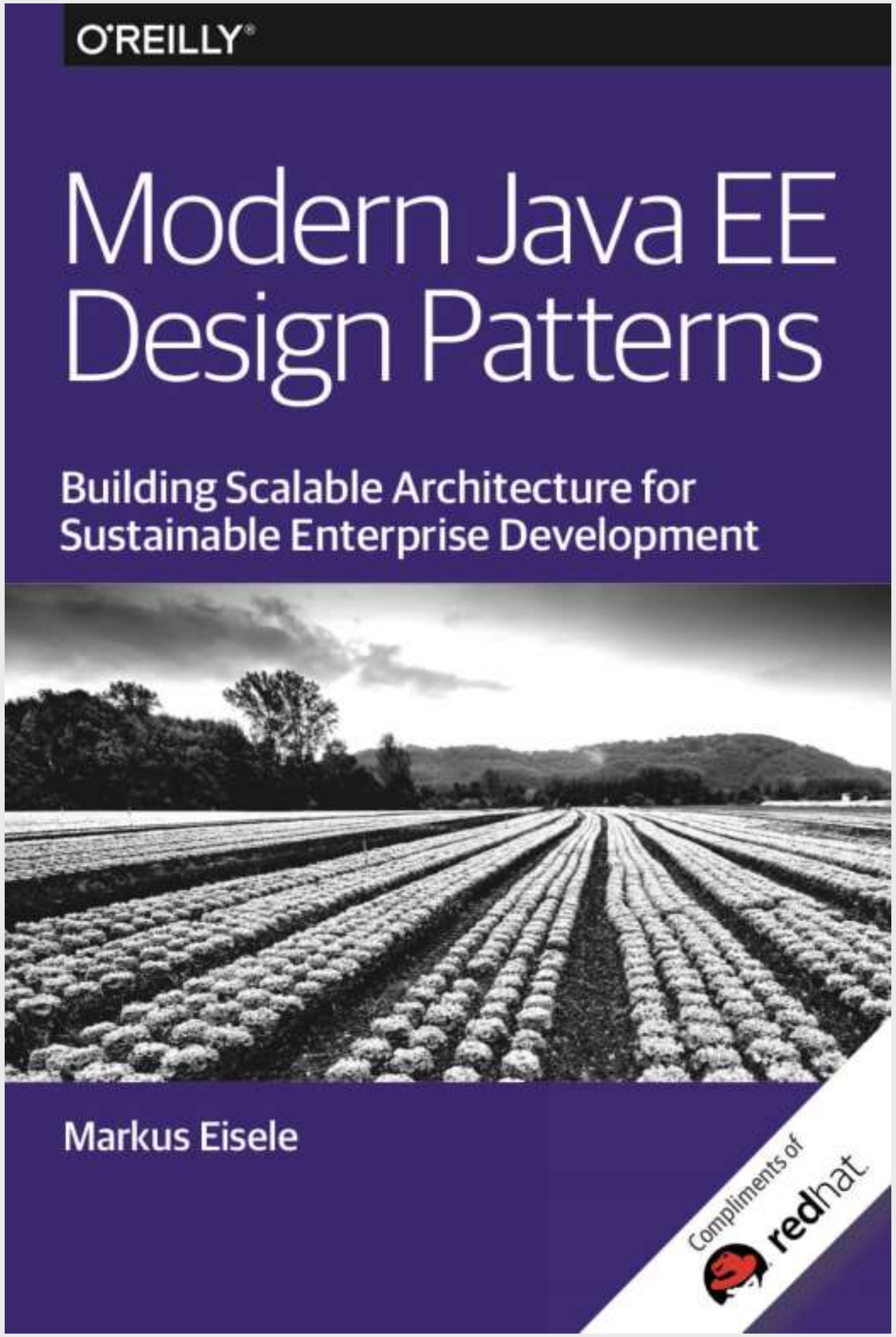
  http://blog.eisele.net/2015/07/running-wildfly-on-openshift-3-with-kubernetes-fabric8-on-windows.html

- Handy Resources for WildFly on Docker

  http://blog.eisele.net/2015/01/java-ee-docker-wildfly-and-microservices-on-docker.html

Follow:

- @myfear
- @jbossdeveloper
- @fabric8io


O'REILLY®
Modern Java EE Design Patterns
Building Scalable Architecture for Sustainable Enterprise Development
Markus Eisele
Compliments of redhat.

http://developers.redhat.com/promotions/distributed-javaee-architecture/

Q&A