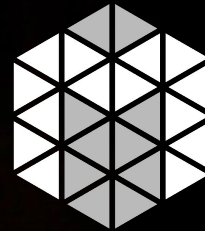




Using



MESOS

to drive **devops** at



GS SHOP

Founding Member,
Container Platform Team



GS SHOP

IT Innovation Center

2015 - Current



*Dev*Ops*, Cloud
Infrastructure,
Microservices,
Containers



vivekjuneja



MESOS

The DevOps Enabler



Happy families are all alike

**every unhappy family is
unhappy in its own way.**

幸福的家庭大抵相同，而不幸的家庭却各有其不幸。

LEO TOLSTOY

Productive teams are all alike
every unproductive team is
unhappy in its own way.



Productivity = Happy Teams



AGENDA

NOT A LONG AGO

1995 - 2015

BEGINNING OF THE CHANGE

2015 - 2016

ADOPTING THE CHANGE

2016 - 2017

THE ROAD AHEAD

2017 - 2019

NOT A LONG AGO

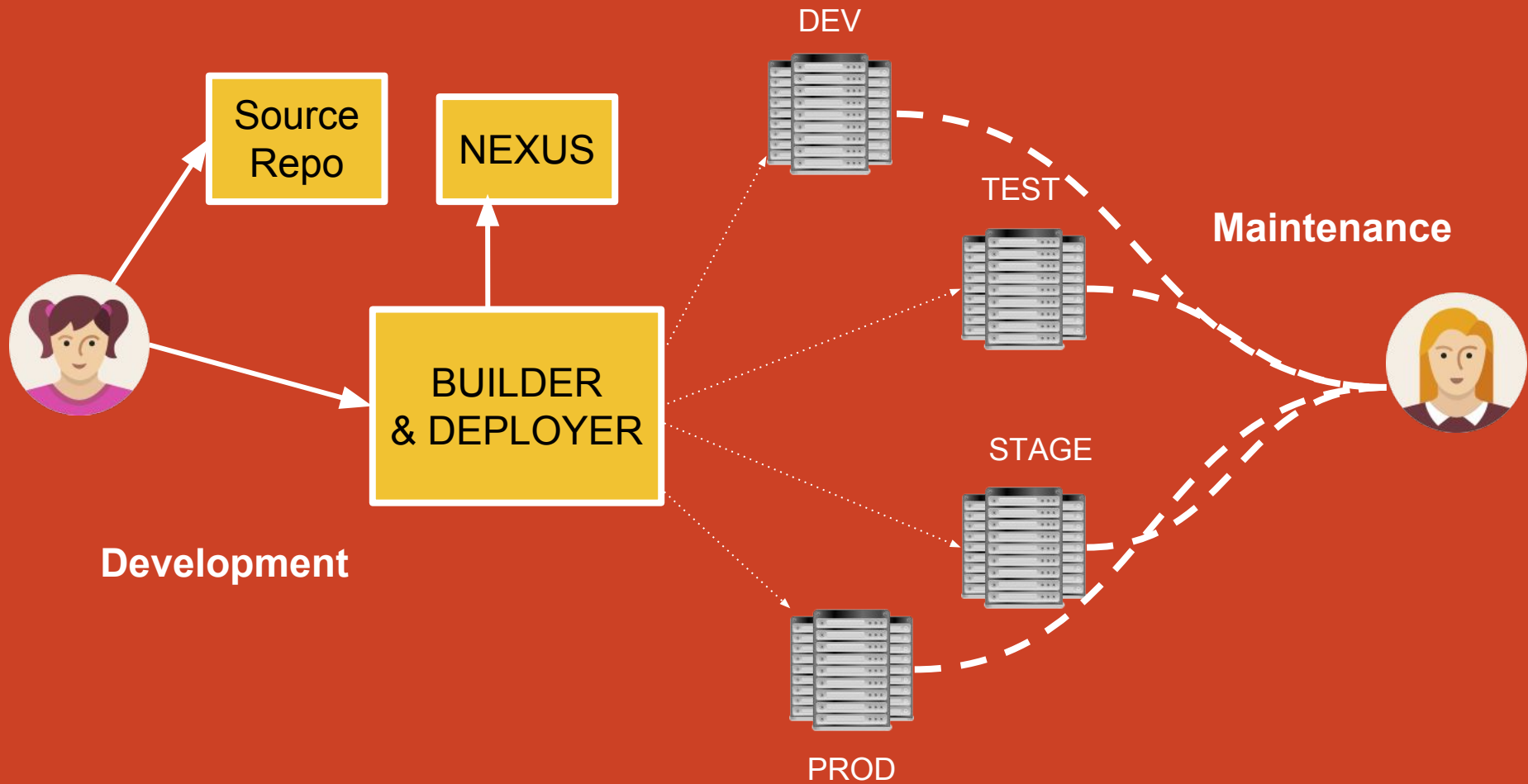
BEGINNING OF THE CHANGE

ADOPTING THE CHANGE

THE ROAD AHEAD



Build & Deploy



Build & Deploy

Deploy Frequency

7 days

Lead Time for Change

10 days

Per developer per week

3 changes

Changes per Deploy

10 changes



Introducing...

OPERATIONS*



DEVELOPER*



OPERATIONS*

Stable system
Minimal Changes
Control on changes

DEVELOPER*

New Features
Fast Changes
Quick rollout to Prod



Monolithic App

Simple
well-understood
Management Primitives

Minimal Moving
Parts

Service Management



Multi-Apps / Microservices

New and Complicated
Management Primitives

Too many
Moving Parts

Yawn (打哈欠) Driven Deployment



Yawn (打哈欠) Driven Deployment



Deploy Code at
3 AM to Production



NOT A LONG AGO

BEGINNING OF THE CHANGE

ADOPTING THE CHANGE

THE ROAD AHEAD



Know thy Issues





**We manage our
machines like
pets.**

**Pets get old, die
and it is sad**



**Each team invents
their own tools
and processes**

**It takes a long
time for
developers to get
feedback**





**Big Bang releases
that are treated
like religious
events**

**Not-my-problem
syndrome**

**Lack of Empathy
between roles**



Inspiration



Inverse Conway Maneuver





Who is

Inverse Conway Maneuver



Melvin Conway



Conway's Law

organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations





Conway's Law

organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations





Inverse Conway Maneuver

Design systems that impose
constructive constraints on the
teams to **change** the way they
communicate and manage





Inverse Conway Maneuver

Design systems that impose
constructive constraints on the
teams to **change** the way they
communicate and manage



O-ring theory of economics

Tasks of Production must be executed proficiently together in order for any of them to be of any value



Michael Kremer



Implications for DevOps

If all of this speculation turns out to be correct, and the O-Ring Theory does indeed apply to DevOps pipelines, then there are some interesting consequences:

- ★ A single weak link in your DevOps pipeline brings down your overall output dramatically. It's not enough to be good in one or two areas and mediocre everywhere else, you need to be good (or great) across the board.
- ★ Small differences in quality (i.e, in how quickly and accurately you perform each stage of your DevOps pipeline) quickly compound to make very large differences between the performance of the best-in-class and the rest. (Two orders of magnitude perhaps, as seen in the data from the State of DevOps Practice Survey?).
- ★ The better you already are, the more value you get from improving your weaknesses. Conversely, if you're fairly poor across the board, you won't get as high a return on investment on an improvement in one specific area as a company with a higher overall level would. These forces tend to lead to 'skill-matching' – fairly uniform levels of performance across the board in the various steps of a DevOps pipeline for a given organisation.



Adrian Colyer

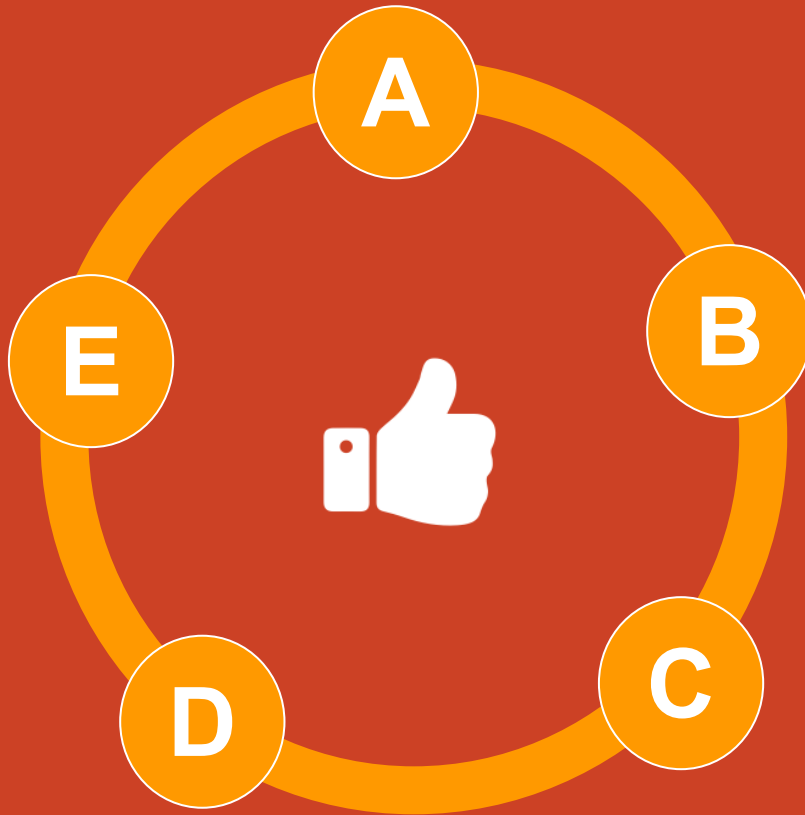


O-ring theory of ~~economics~~ devops

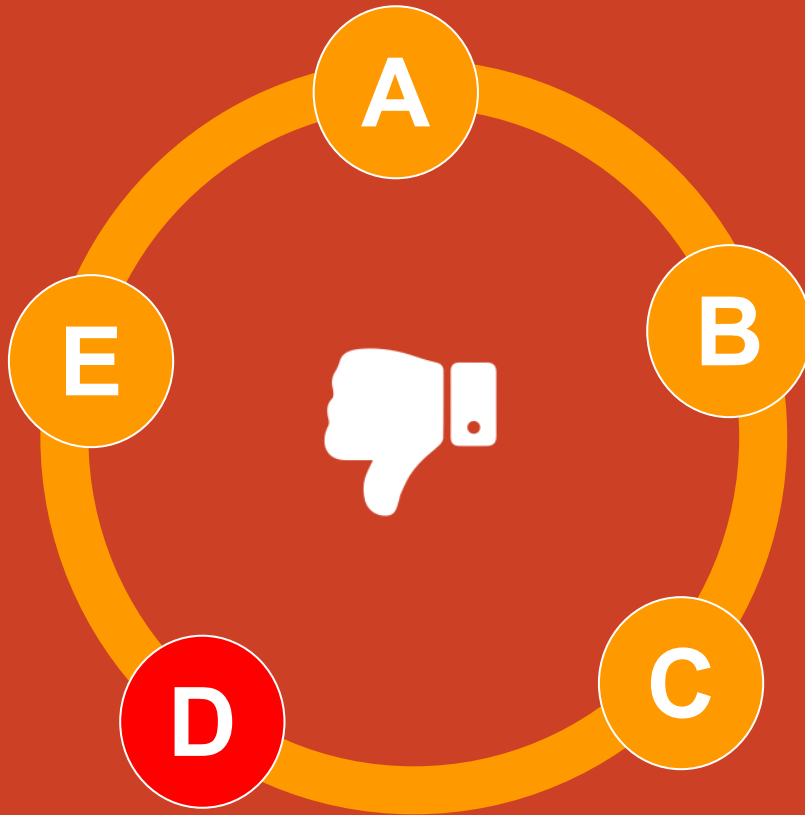
Implications for DevOps

If all of this speculation turns out to be correct, and the O-Ring Theory does indeed apply to DevOps pipelines, then there are some interesting consequences:

- ★ A single weak link in your DevOps pipeline brings down your overall output dramatically. It's not enough to be good in one or two areas and mediocre everywhere else, you need to be good (or great) across the board.
- ★ Small differences in quality (i.e, in how quickly and accurately you perform each stage of your DevOps pipeline) quickly compound to make very large differences between the performance of the best-in-class and the rest. (Two orders of magnitude perhaps, as seen in the data from the State of DevOps Practice Survey?).
- ★ The better you already are, the more value you get from improving your weaknesses. Conversely, if you're fairly poor across the board, you won't get as high a return on investment on an improvement in one specific area as a company with a higher overall level would. These forces tend to lead to 'skill-matching' – fairly uniform levels of performance across the board in the various steps of a DevOps pipeline for a given organisation.



O-ring theory of
devops



O-ring theory of
devops

Tenets



Tenets

Disposable Apps

**Measure and Log
everything**

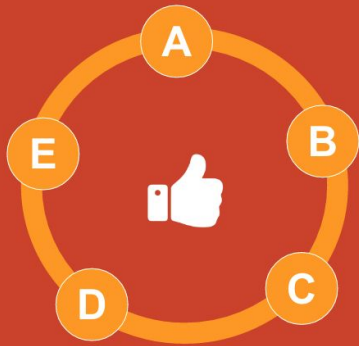


**Developer
Productivity**

**Automate Service
management primitives**

**Shared Multi-tenant
Infrastructure and Tooling**

Reverse Conway Maneuver



Apply !

Master Plan

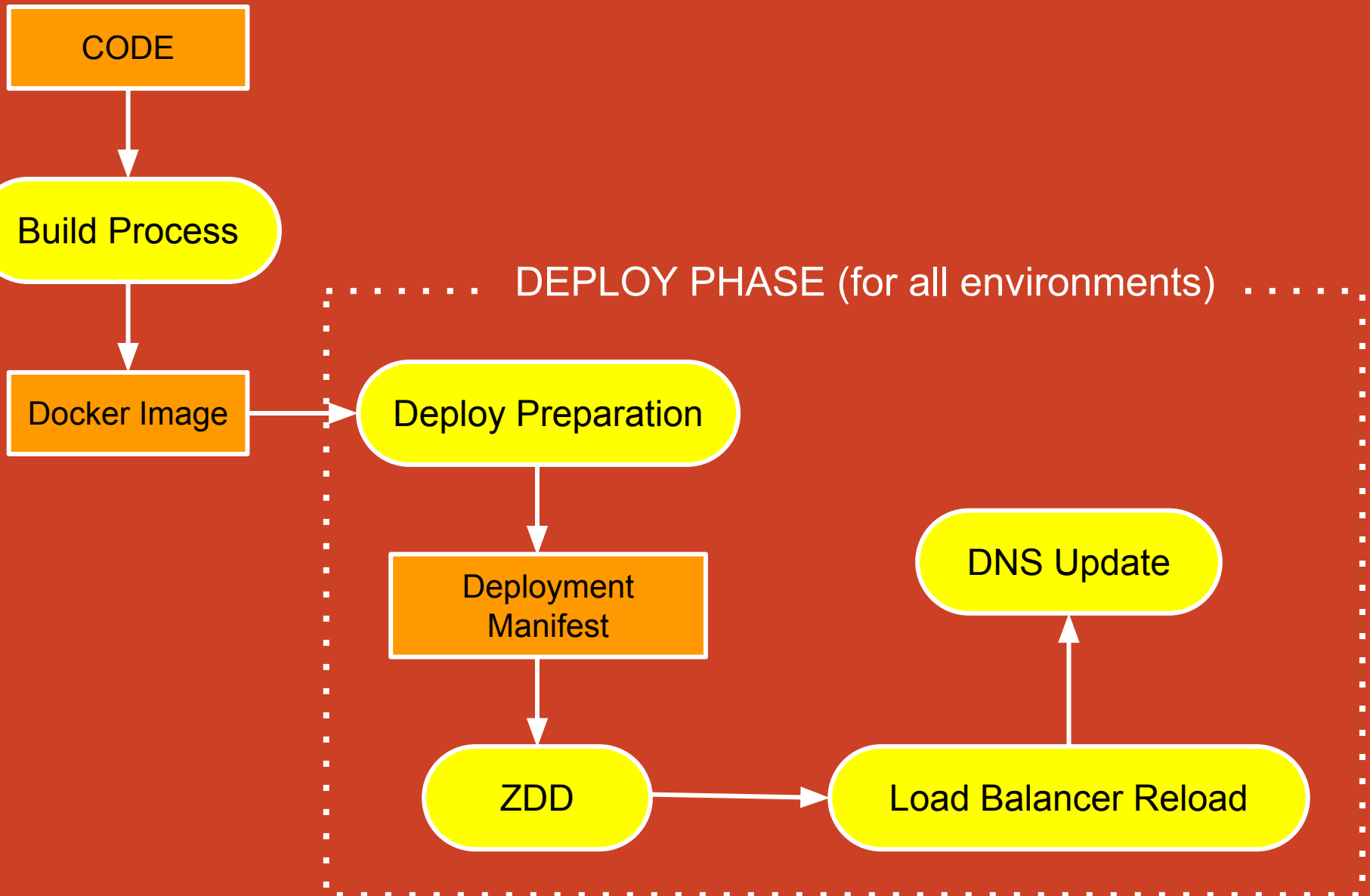
O-ring theory of
devops



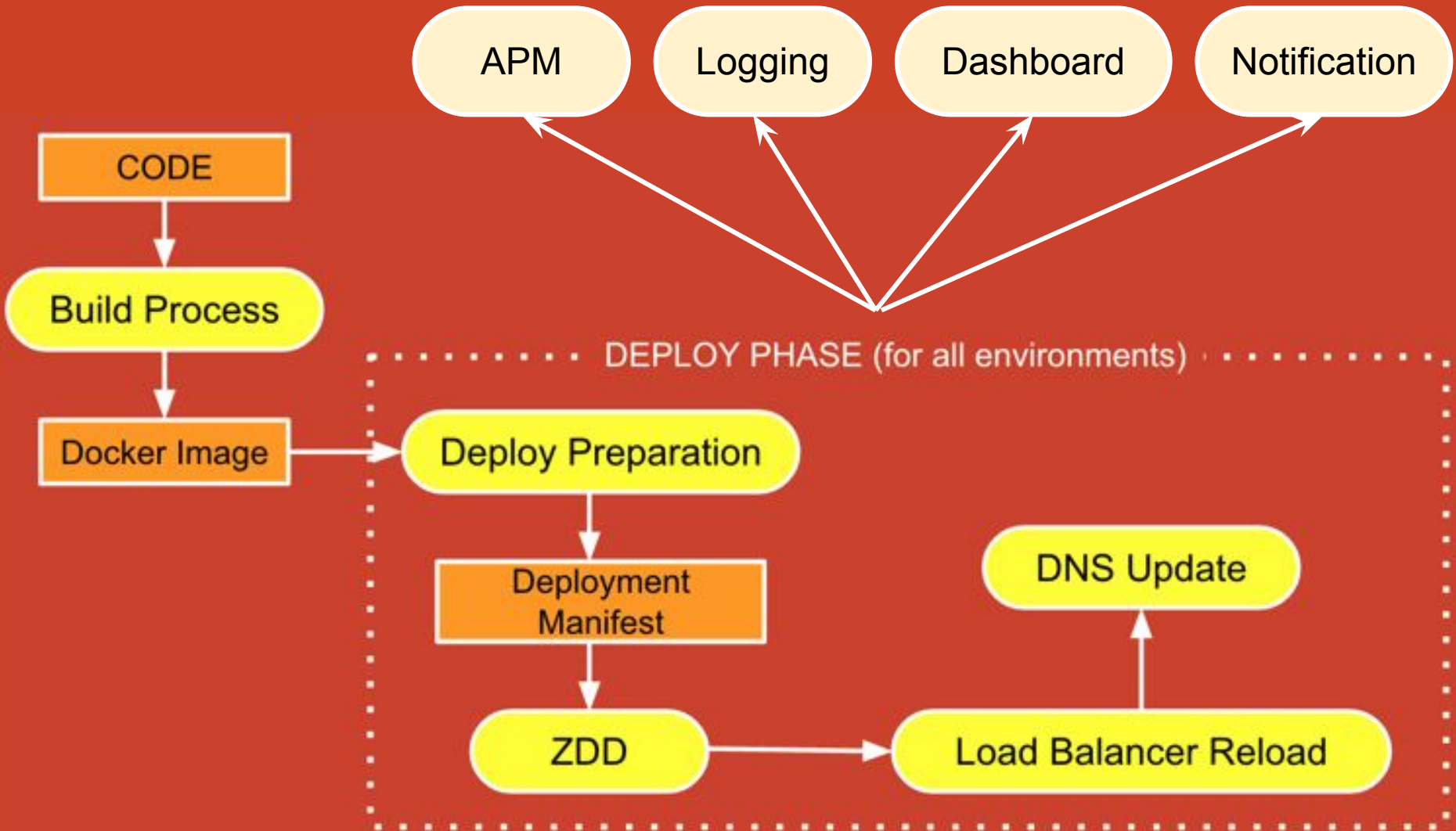
Service Delivery Platform

The building blocks for building reliable software at scale

End to End Workflow



End to End Workflow





```
"instances": "$runtime:instances$",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": ["$serviceport:tomcat$"],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffs": {
  "initial": 10,
  "max": 100,
  "multiplier": 1.5
},
"labels": {
  "HAPROXY_DEPLOYMENT_GROUP": "tomcat"
},
"ENV": {
  "COMMIT_SHA": "$build:commitSha",
  "BRANCH_NAME": "$build:branchName",
  "BUILD_ID": "$build:buildId",
  "DEPLOYMENT_ID": "$build:deploymentId",
  "PROJECT_NAME": "$build:projectName",
  "TRIGGER_NAME": "$build:triggerName",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAPROXY_0_VHOST": "<runtime:env>.<domain:name>",
  "HAPROXY_0_STICKY": "true",
  "HAPROXY_DEPLOYMENT_ALT_PORT": "<runtime:altserviceport>",
  "HAPROXY_DEPLOYMENT_GROUP": "<build:project>--<runtime:env>",
  "HAPROXY_APP_ID": "<build:project>--<runtime:env>"
}
```

One Deployment Manifest to rule them all



DEV

TEST

STAGE

PROD

```
"instances": "$runtime:instances$",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": ["$serviceport:tomcat$"],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAPROXY_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAPROXY_0_VHOST": "<runtime:env>.<domain:name>",
  "HAPROXY_0_STICKY": "true",
  "HAPROXY_DEPLOYMENT_ALT_PORT": "<runtime:altserviceport>",
  "HAPROXY_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAPROXY_APP_ID": "<build:project>-<runtime:env>"
}
```

Deployment Manifest

```
"instances": "$runtime:instances$",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": ["$serviceport:tomcat$"],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAPROXY_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAPROXY_0_VHOST": "<runtime:env>.<domain:name>",
  "HAPROXY_0_STICKY": "true",
  "HAPROXY_DEPLOYMENT_ALT_PORT": "<runtime:altserviceport>",
  "HAPROXY_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAPROXY_APP_ID": "<build:project>-<runtime:env>"
}
```



Template

```
build:
  id: ${id?}
  user: ${user?}
  tag: ${image_tag?}
  image: ${image?}
  commit: ${commit_hash?}
  branch: ${branch?}
  triggeredBy: ${build_user?}
  project: ${project_name?}
  jobName: ${builder?}
  deployId: ${deploy_id?}
  registry: ${registry_url?}

runtime:
  env: ${env?}
  instances: ${app_count?}
  altserviceport: ${bluegreen_altport?}
  project: ${project_name?}
```



```

"instances": "${runtime:instances$}",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": [{"serviceport:tomcat$}],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAProxy_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAProxy_0_VHOST": "<runtime:env>.<domain:name>",
  "HAProxy_0_STICKY": "true",
  "HAProxy_DEPLOYMENT_ALT_PORT": "<runtime:altServiceport>",
  "HAProxy_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAProxy_APP_ID": "<build:project>-<runtime:env>"
}

```



```

build:
  id: ${id?}
  user: ${user?}
  tag: ${image_tag?}
  image: ${image?}
  commit: ${commit_hash?}
  branch: ${branch?}
  triggeredBy: ${build_user?}
  project: ${project_name?}
  jobName: ${builder?}
  deployId: ${deploy_id?}
  registry: ${registry_url?}

runtime:
  env: ${env?}
  instances: ${app_count?}
  altServiceport: ${bluegreen_altport?}
  project: ${project_name?}

```

DEV

TEST

STAGE

PROD

```

"instances": "${runtime:instances$}",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": [{"serviceport:tomcat$}],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAProxy_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAProxy_0_VHOST": "<runtime:env>.<domain:name>",
  "HAProxy_0_STICKY": "true",
  "HAProxy_DEPLOYMENT_ALT_PORT": "<runtime:altServiceport>",
  "HAProxy_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAProxy_APP_ID": "<build:project>-<runtime:env>"
}

```

```

"instances": "${runtime:instances$}",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": [{"serviceport:tomcat$}],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAProxy_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAProxy_0_VHOST": "<runtime:env>.<domain:name>",
  "HAProxy_0_STICKY": "true",
  "HAProxy_DEPLOYMENT_ALT_PORT": "<runtime:altServiceport>",
  "HAProxy_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAProxy_APP_ID": "<build:project>-<runtime:env>"
}

```

```

"instances": "${runtime:instances$}",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": [{"serviceport:tomcat$}],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAProxy_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAProxy_0_VHOST": "<runtime:env>.<domain:name>",
  "HAProxy_0_STICKY": "true",
  "HAProxy_DEPLOYMENT_ALT_PORT": "<runtime:altServiceport>",
  "HAProxy_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAProxy_APP_ID": "<build:project>-<runtime:env>"
}

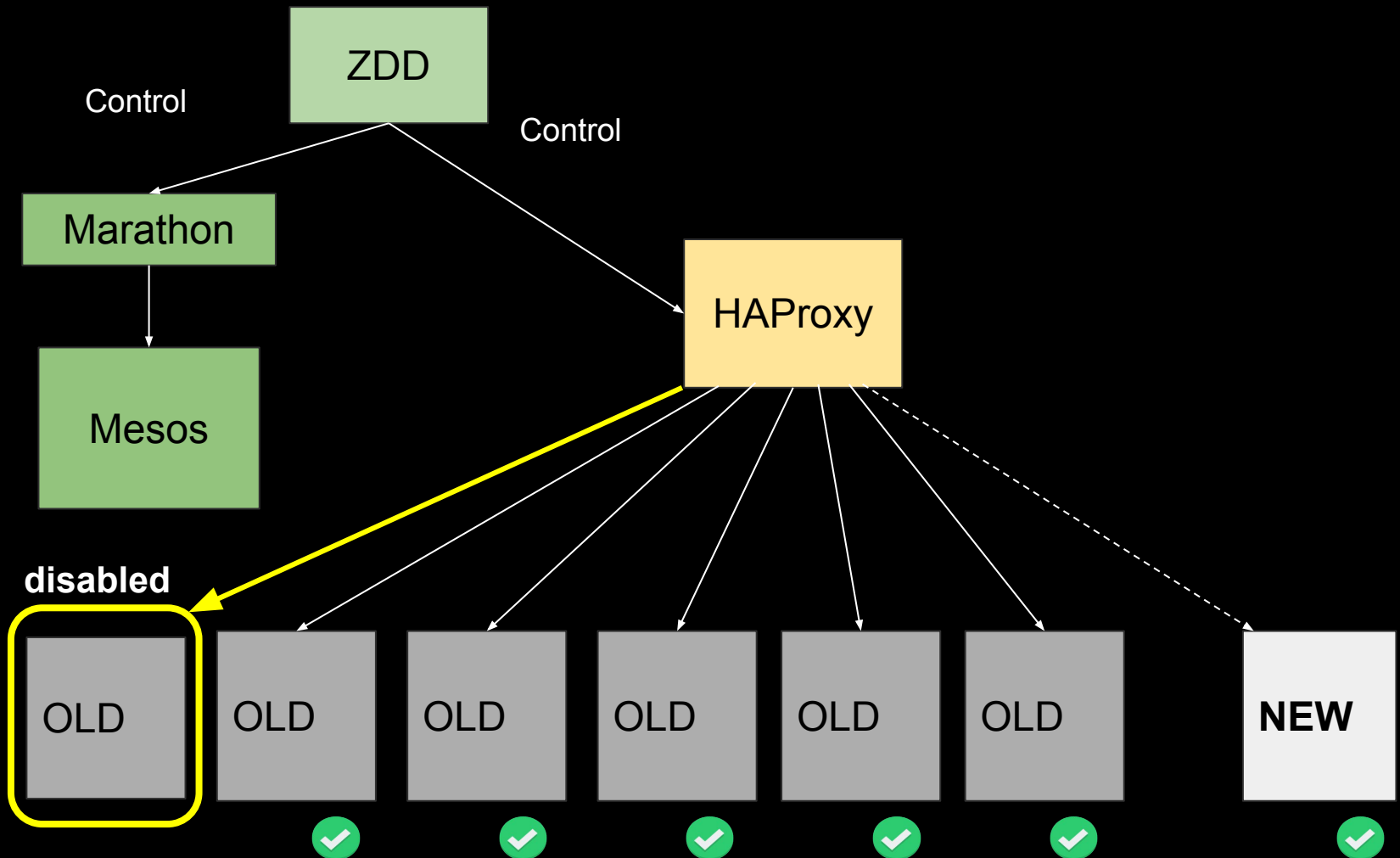
```

```

"instances": "${runtime:instances$}",
"cpus": 2,
"mem": 4096,
"disk": 0,
"ports": [{"serviceport:tomcat$}],
"executor": "",
"constraints": [],
"storeUrls": [],
"requirePorts": false,
"backoffSeconds": 1,
"labels": {
  "HAProxy_0_GROUP": "external",
  "ENV": "<runtime:env>",
  "COMMIT": "<build:commit>",
  "BRANCH": "<build:branch>",
  "BUILD_ID": "<build:id>",
  "DEPLOYID": "<build:deployId>|",
  "PROJECT": "<build:project>",
  "TRIGGERED_BY": "<build:triggeredBy>",
  "JOB_NAME": "<build:jobName>",
  "USER": "<build:user>",
  "HAProxy_0_VHOST": "<runtime:env>.<domain:name>",
  "HAProxy_0_STICKY": "true",
  "HAProxy_DEPLOYMENT_ALT_PORT": "<runtime:altServiceport>",
  "HAProxy_DEPLOYMENT_GROUP": "<build:project>-<runtime:env>",
  "HAProxy_APP_ID": "<build:project>-<runtime:env>"
}

```

Blue Green Deployment



Ideally

Zero Downtime Deployment

Blue Green Deployment



Deploy when
awake !

Notifications



jenkins BOT 4:58 PM ☆

Order-Front-DevInt-Docker-Build - #197 Success after 31 sec ([Open](#))
Building revision `29012` at branch `trunk`.
Initiated by user `kimsj`.



Marathon Event Bot BOT 4:58 PM

Deployment info

The deployment of `/pc-order-front-dev/logs/kibana` triggered the following steps:

1. RestartApplication

`USER : kimsj`

`BUILD_ID : 197`

`ENV : dev`



[Click here to access : /pc-order-front-dev/logs/kibana](#) Today at 4:58 PM



Groups

`/order-api-stage``/pc-order-front-stage``/pc-order-front-prod`**server-green**

Deployed Date : 2016-11-10T04:31:18.997Z

Basic Information

Service Location

Se

CPU	2
Memory(MB)	4096
Disk	0
Instances	4

Custom Dashboard

**HAPROXY****Metadata based Rollback****Comprehensive Health
Check uses Service
Discovery****Supports Multiple Data
Centers / Platform
Regions****Common API for
Developers for integration
with CI Server**

2016

State of DevOps Report

High-performing IT organizations report experiencing:



200x more frequent deployments



24x faster recovery from failures

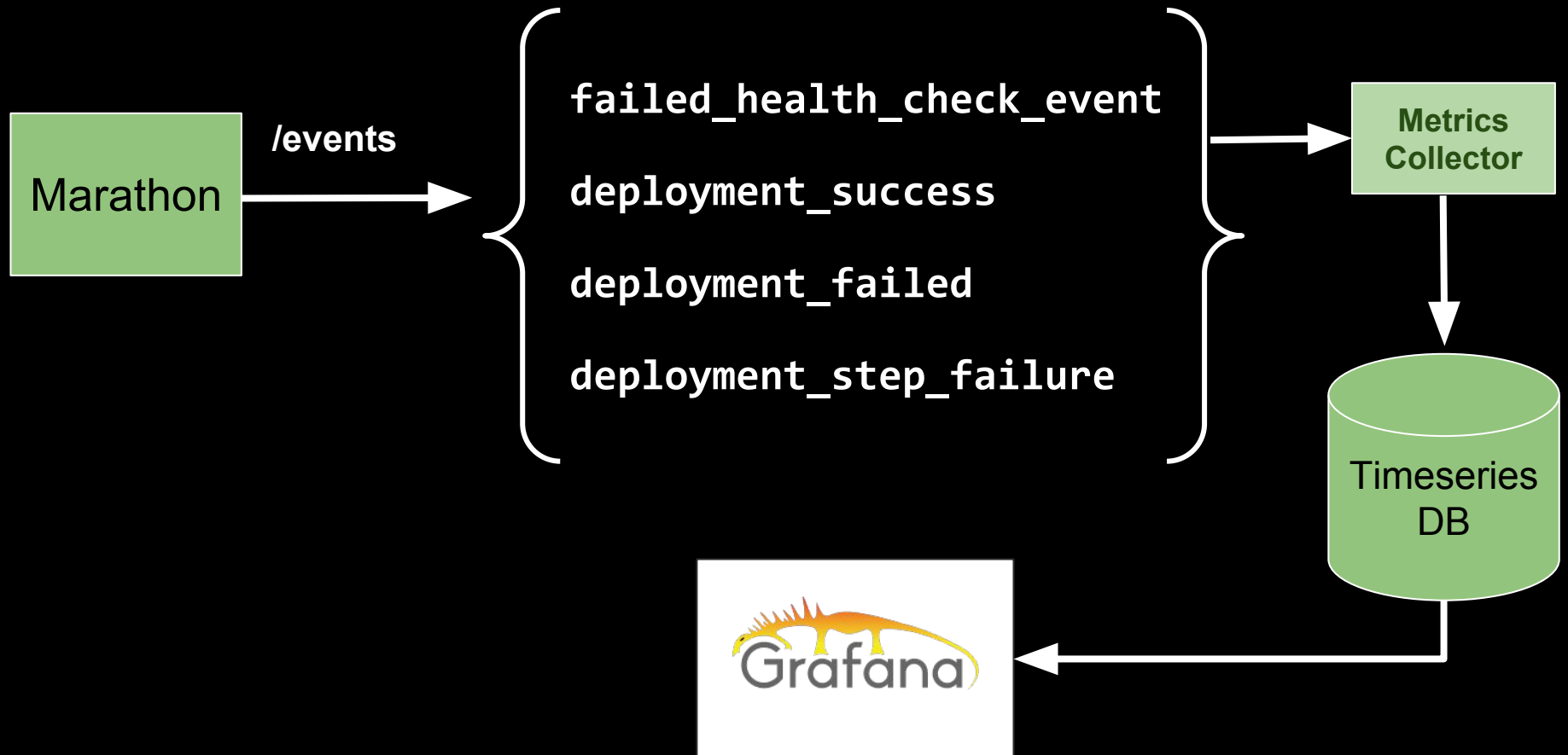


3x lower change failure rate

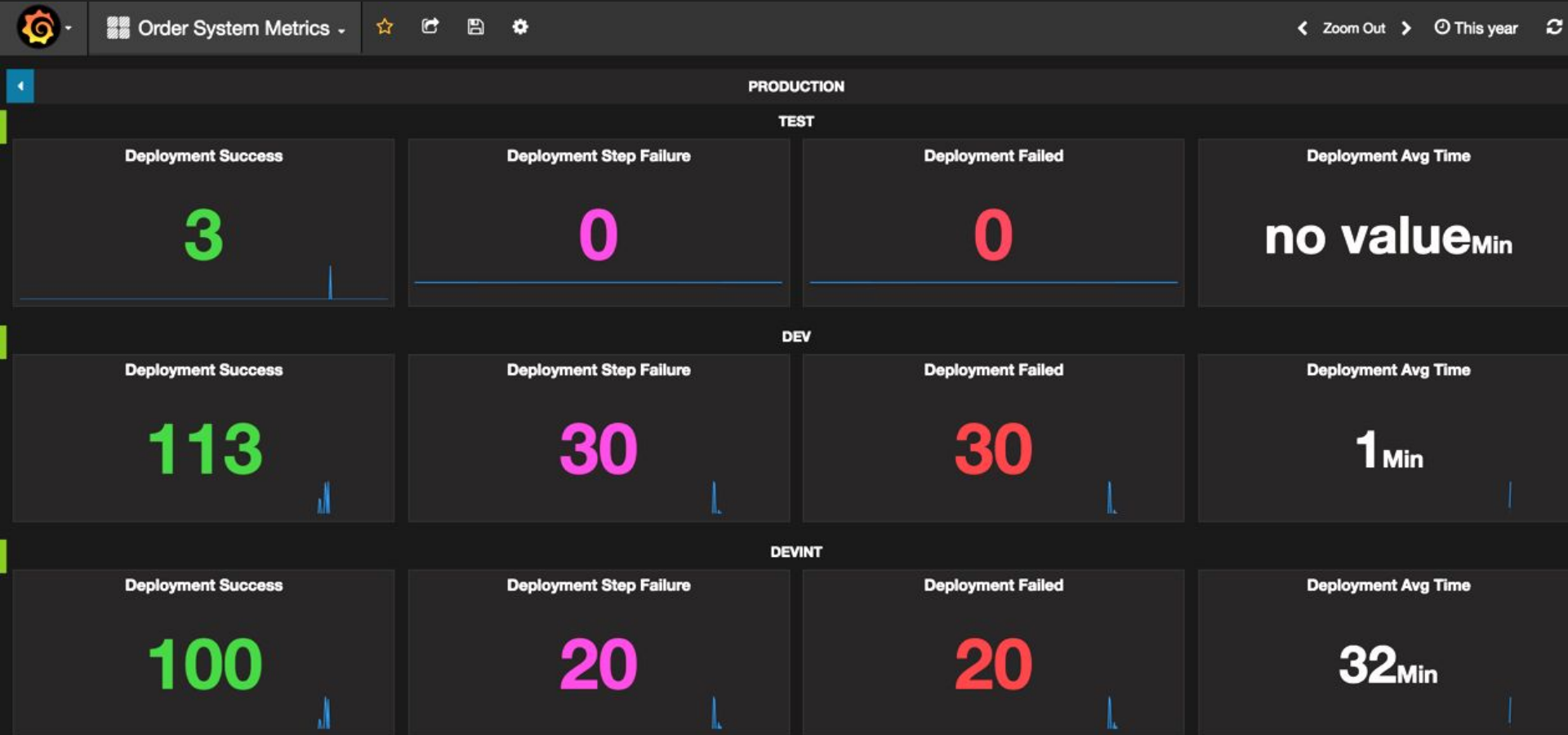


2,555x shorter lead times

Devops Metrics



Devops Metrics



Monitoring and Alerts

APM

**Container Platform Stack
Monitoring**

**Service Monitoring (container,
non-container)**

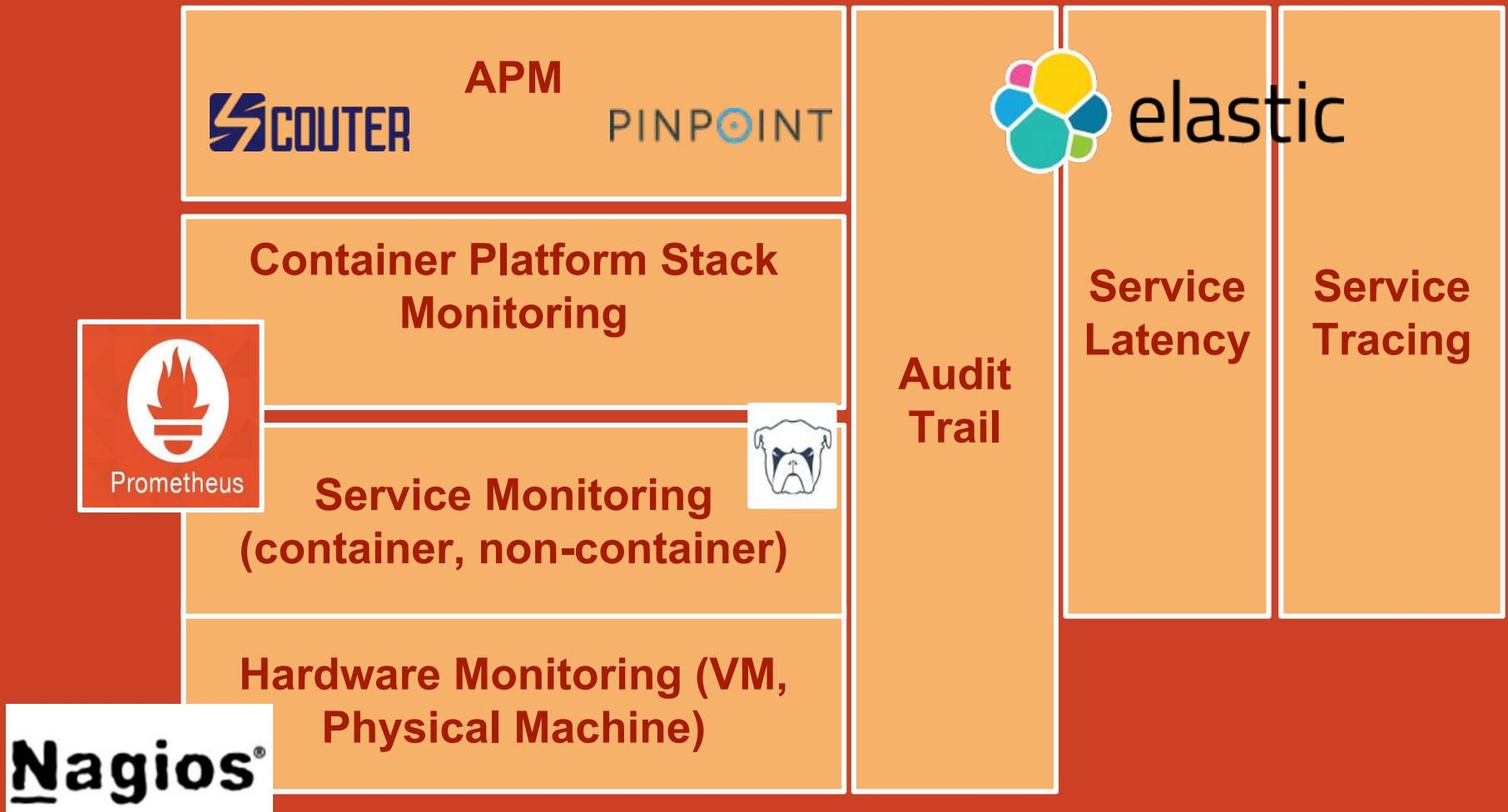
**Hardware Monitoring (VM,
Physical Machine)**

**Audit
Trail**

**Service
Latency**

**Service
Tracing**

Monitoring and Alerts



Monitoring and Alerts

Status

?

Number of Hosts: 39

Any Led ▾

All Host Groups ▾

Find hosts...

×

*▲	Host	%Cpu	%Mem	Status	Events
●		<div><div></div></div>	<div><div></div></div>	13 out of 14 services are available, 1 not monitored	289
●		<div><div></div></div>	<div><div></div></div>	14 out of 15 services are available, 1 not monitored	337
●		<div><div></div></div>	<div><div></div></div>	11 out of 12 services are available, 1 not monitored	297
●		<div><div></div></div>	<div><div></div></div>	11 out of 12 services are available, 1 not monitored	226
●		<div><div></div></div>	<div><div></div></div>	All 4 services are available	11
●		<div><div></div></div>	<div><div></div></div>	All 4 services are available	11
●		<div><div></div></div>	<div><div></div></div>	All 4 services are available	14
●		<div><div></div></div>	<div><div></div></div>	All 6 services are available	32
●		<div><div></div></div>	<div><div></div></div>	All 4 services are available	12
●		<div><div></div></div>	<div><div></div></div>	All 7 services are available	10
●		<div><div></div></div>	<div><div></div></div>	All 7 services are available	9
●		<div><div></div></div>	<div><div></div></div>	All 6 services are available	13
●		<div><div></div></div>	<div><div></div></div>	All 10 services are available	99
●		<div><div></div></div>	<div><div></div></div>	All 13 services are available	178
●		<div><div></div></div>	<div><div></div></div>	All 8 services are available	129
●		<div><div></div></div>	<div><div></div></div>	All 9 services are available	166
●		<div><div></div></div>	<div><div></div></div>	All 14 services are available	29

Monitoring and Alerts



SYSTEM - STATS ▾



◀ Zoom Out ▶

🕒 Nov 7, 2016 15:51:18 to Nov 7, 2016 16:11:21



Host ▾ ITSDP Service Type monitoring ▾ ITSDP Service Name cAdvsior ▾

System Containers

15

System Containers - Memory Usage

4.17 GiB

System Containers - CPU Usage

201%

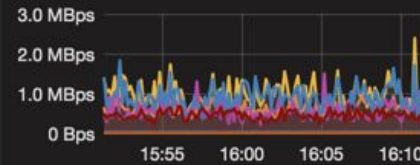
CPU Usage - monitoring



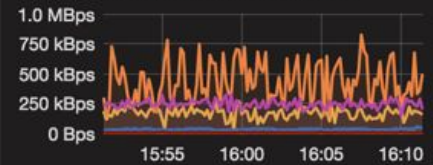
Memory Usage - monitoring



Network Rx - monitoring



Network Tx - monitoring



CPU Usage - All Services



Memory Usage - All Services

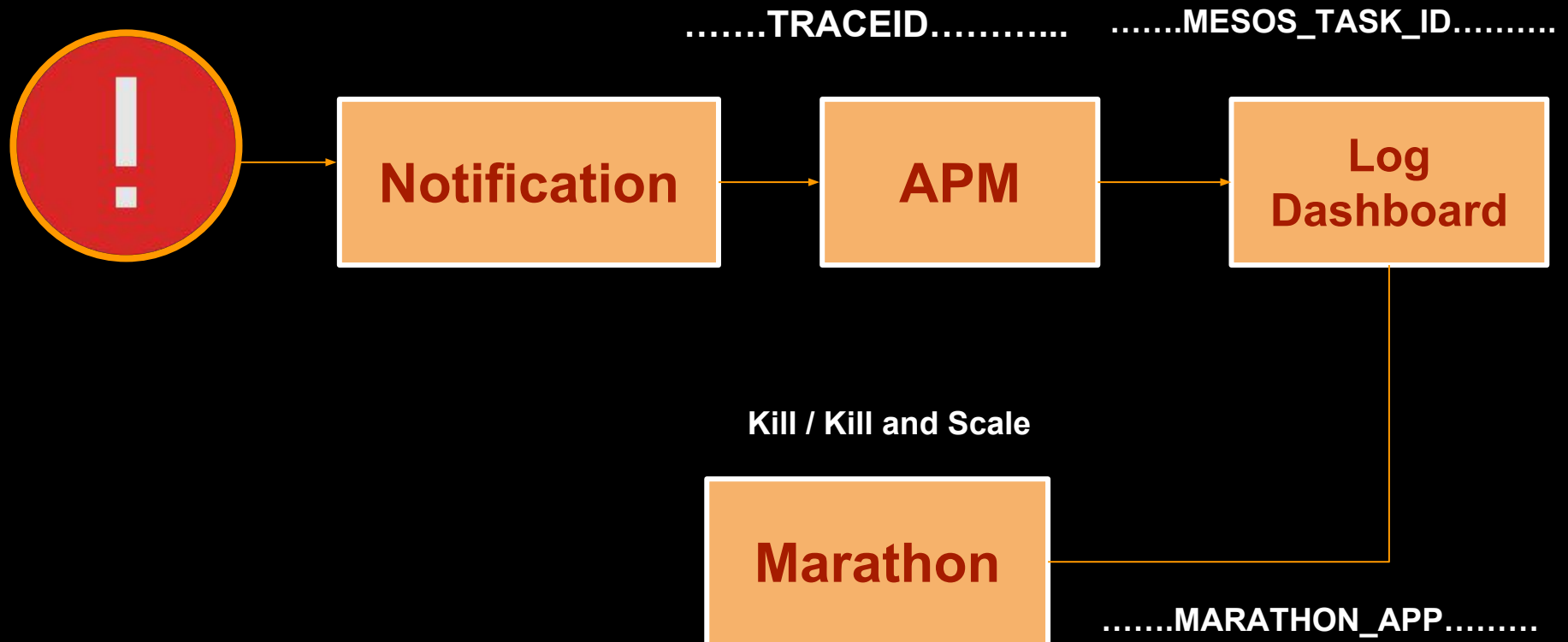


Network Rx - All Services



Network Tx - All Services





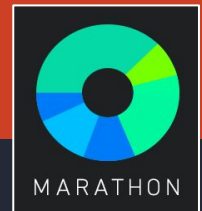
- ☐ tags
- ☒ taskid
- ☐ time
- ☐ traceid
- ☐ type

Q ☐ Micro Analysis of taskId (string)
Value

1. pc-order-front-prod_java-web-app_server-blue.7f680398-aabe-11e6-8fcf-4e39463eb257
2. pc-order-front-prod_java-web-app_server-blue.9786c9c9-aabe-11e6-8fcf-4e39463eb257
3. pc-order-front-prod_java-web-app_server-blue.f24da8fc-aabf-11e6-8fcf-4e39463eb257
4. pc-order-front-prod_java-web-app_server-blue.f24d81eb-aabf-11e6-8fcf-4e39463eb257



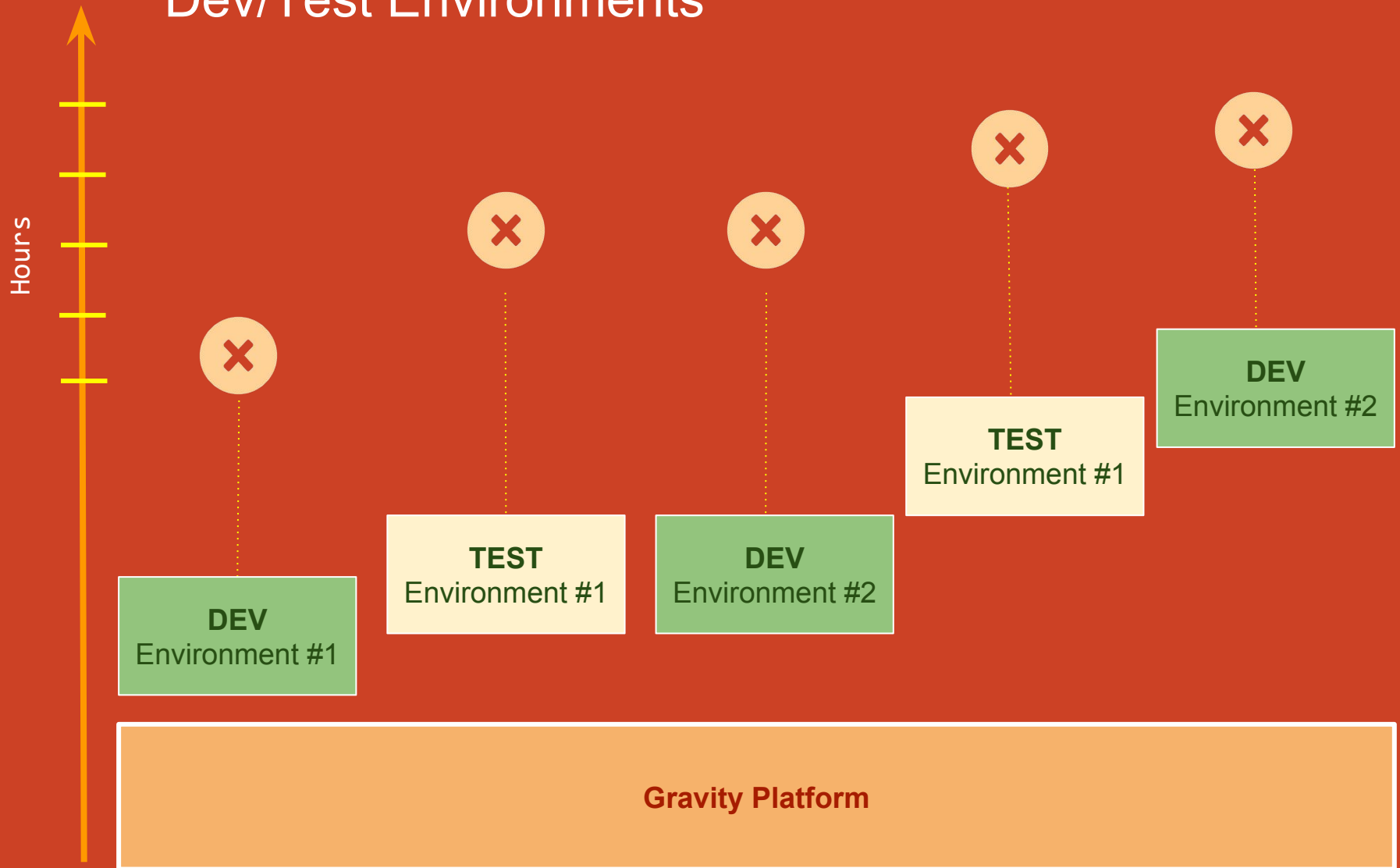
Fault Identification



ID	Health
<input checked="" type="checkbox"/> pc-order-front-prod_java-web-app_server-blue.7f680398-aabe-11e6-8fcf-4e39463eb257 mesos-slave-01.order.gravity.gsenext.com:31006	Healthy
<input type="checkbox"/> pc-order-front-prod_java-web-app_server-blue.9786c9c9-aabe-11e6-8fcf-4e39463eb257 mesos-slave-01.order.gravity.gsenext.com:31587	Healthy
<input type="checkbox"/> pc-order-front-prod_java-web-app_server-blue.f24d81eb-aabf-11e6-8fcf-4e39463eb257 mesos-slave-01.order.gravity.gsenext.com:31677	Healthy
<input type="checkbox"/> pc-order-front-prod_java-web-app_server-blue.f24da8fc-aabf-11e6-8fcf-4e39463eb257 mesos-slave-02.order.gravity.gsenext.com:31789	Healthy

Disposable Transient Dev/Test Environments

Fair Share Usage



Standardization

Worker Node

Prometheus

Marathon-LB

cAdvisor

HAproxy

Monit

Log
Forwarder

Mesos Agent

Docker

Master Node

cAdvisor

Monit

Log
Forwarder

Mesos
Master

Marathon

Platform Provisioning

PKG
REPOSITORY

FLEET

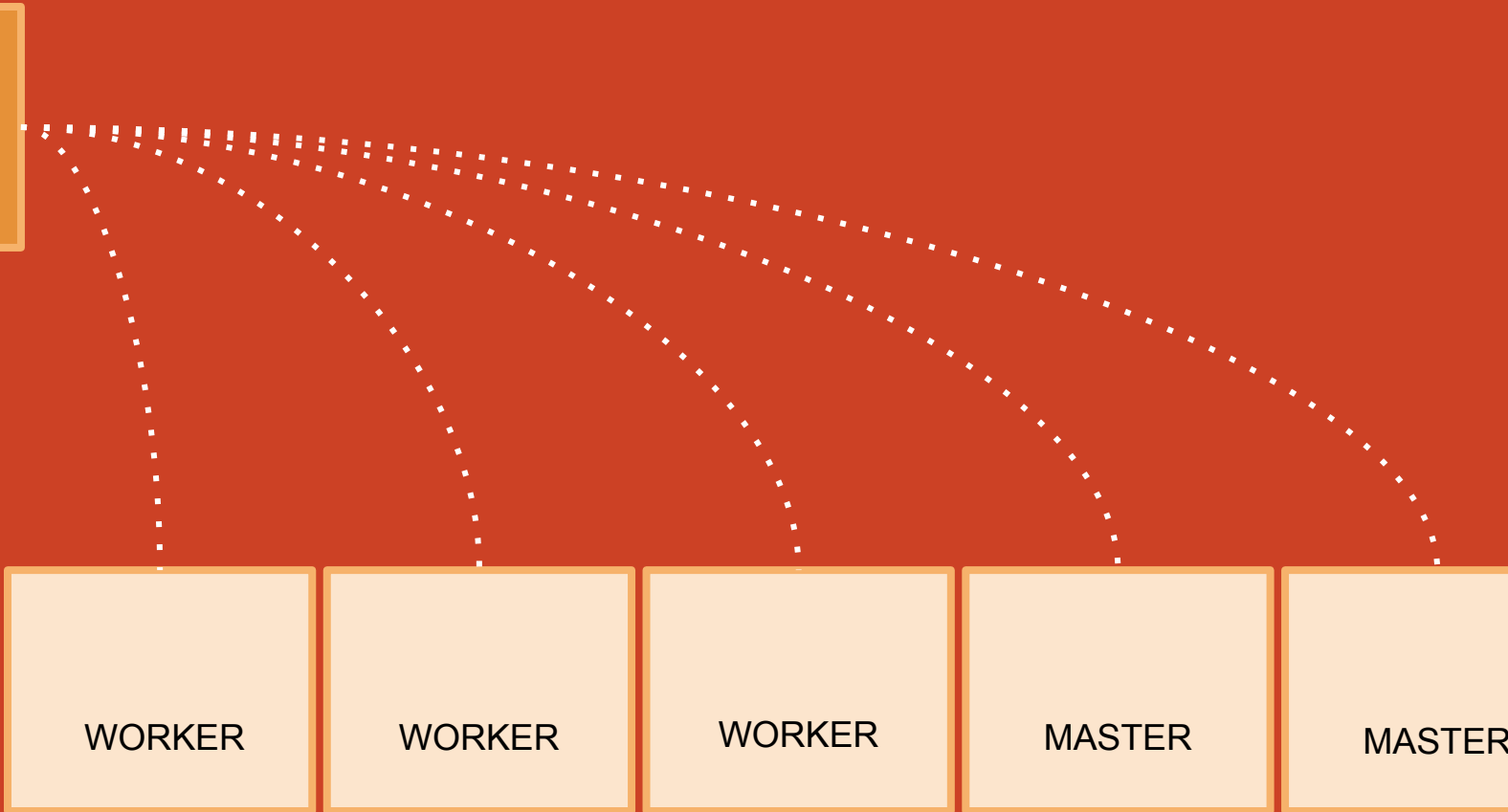
WORKER

WORKER


WORKER

MASTER

MASTER



Platform Provisioning


FLEET

WORKER

WORKER

WORKER

MASTER

.....

WORKER

WORKER

WORKER

MASTER

.....

WORKER


WORKER

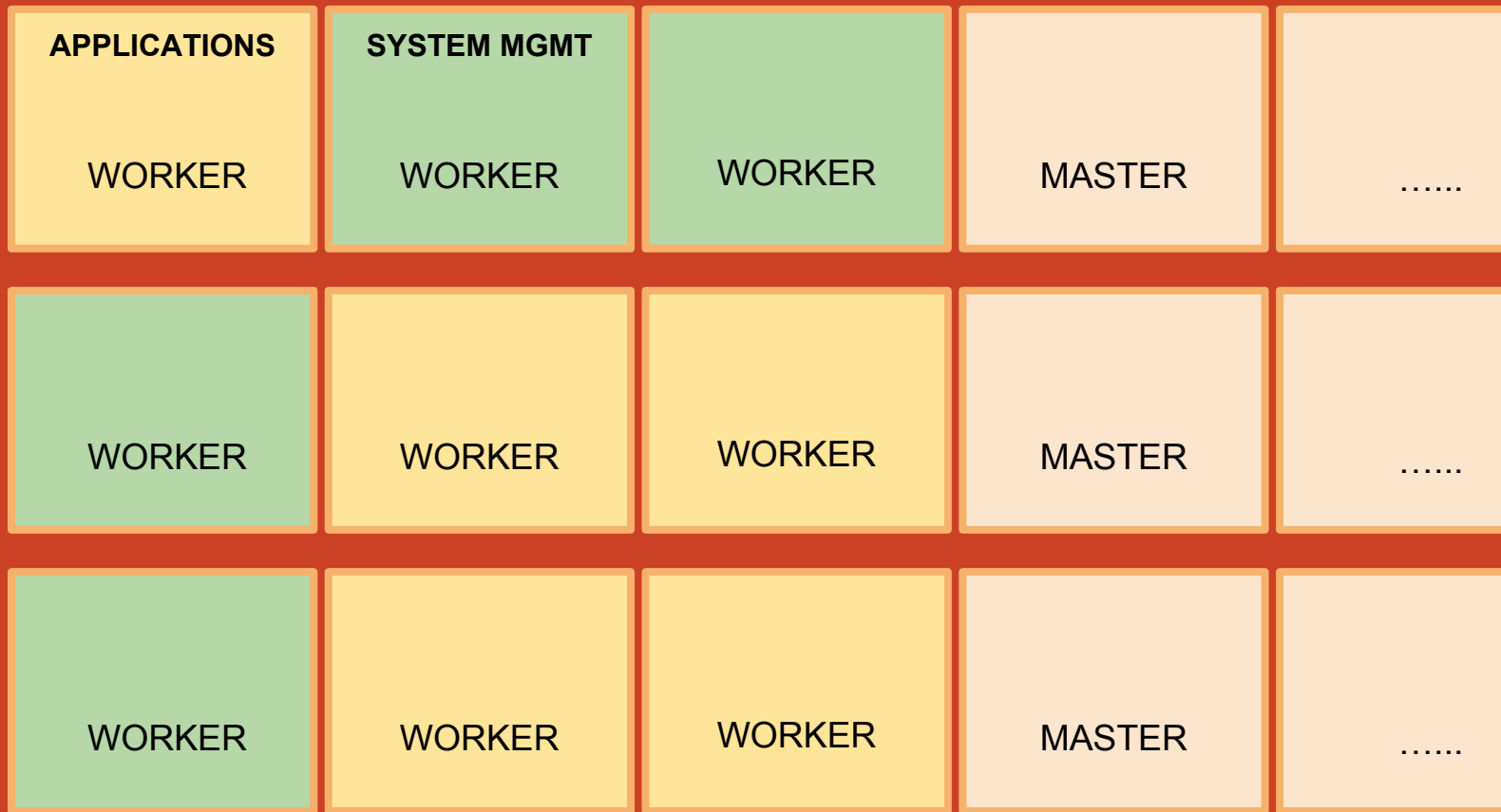
WORKER

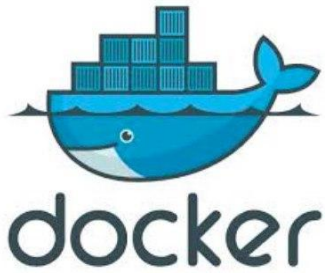
MASTER

.....

Platform Provisioning


FLEET





Jenkins



logstash



Kibana



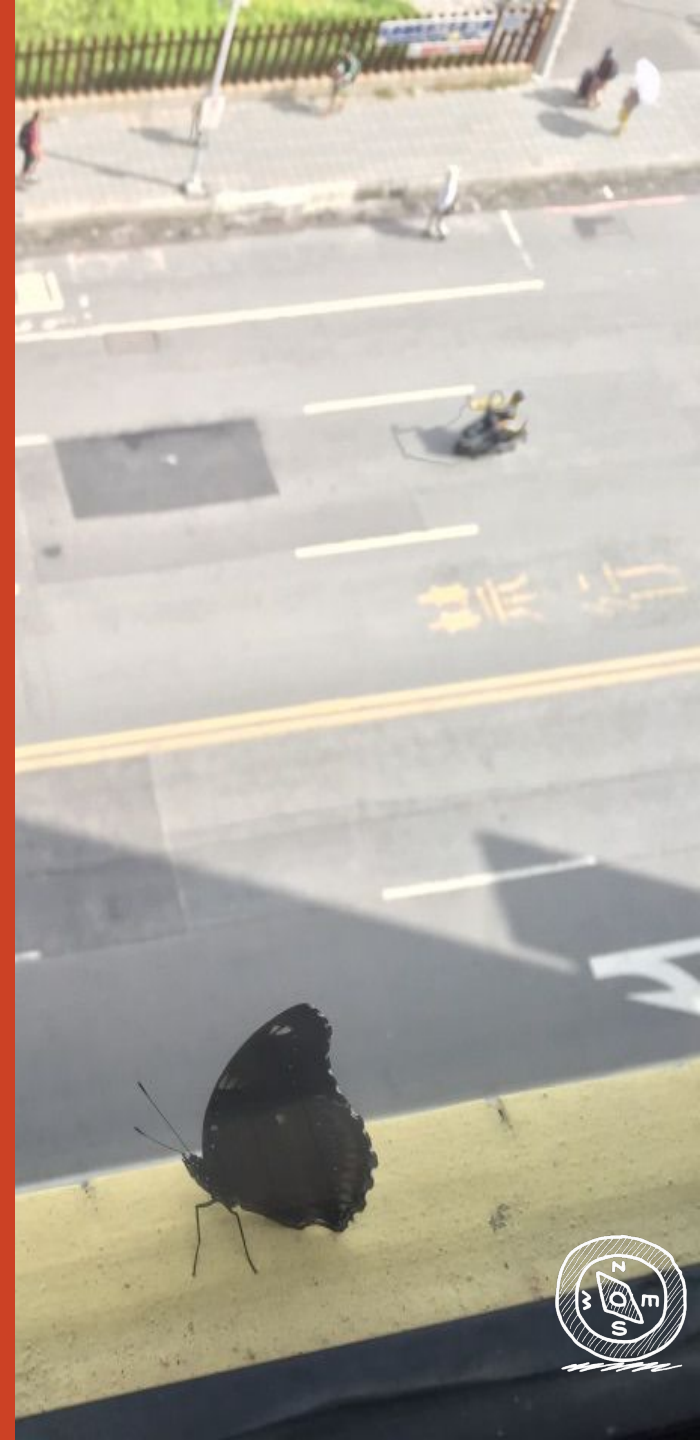
& many more

NOT A LONG AGO

BEGINNING OF THE CHANGE

ADOPTING THE CHANGE

THE ROAD AHEAD



How to Successfully Install Agile/DevOps in Asia

Posted by [Tsuyoshi Ushio](#) on Oct 22, 2016 | [1 Discuss](#)

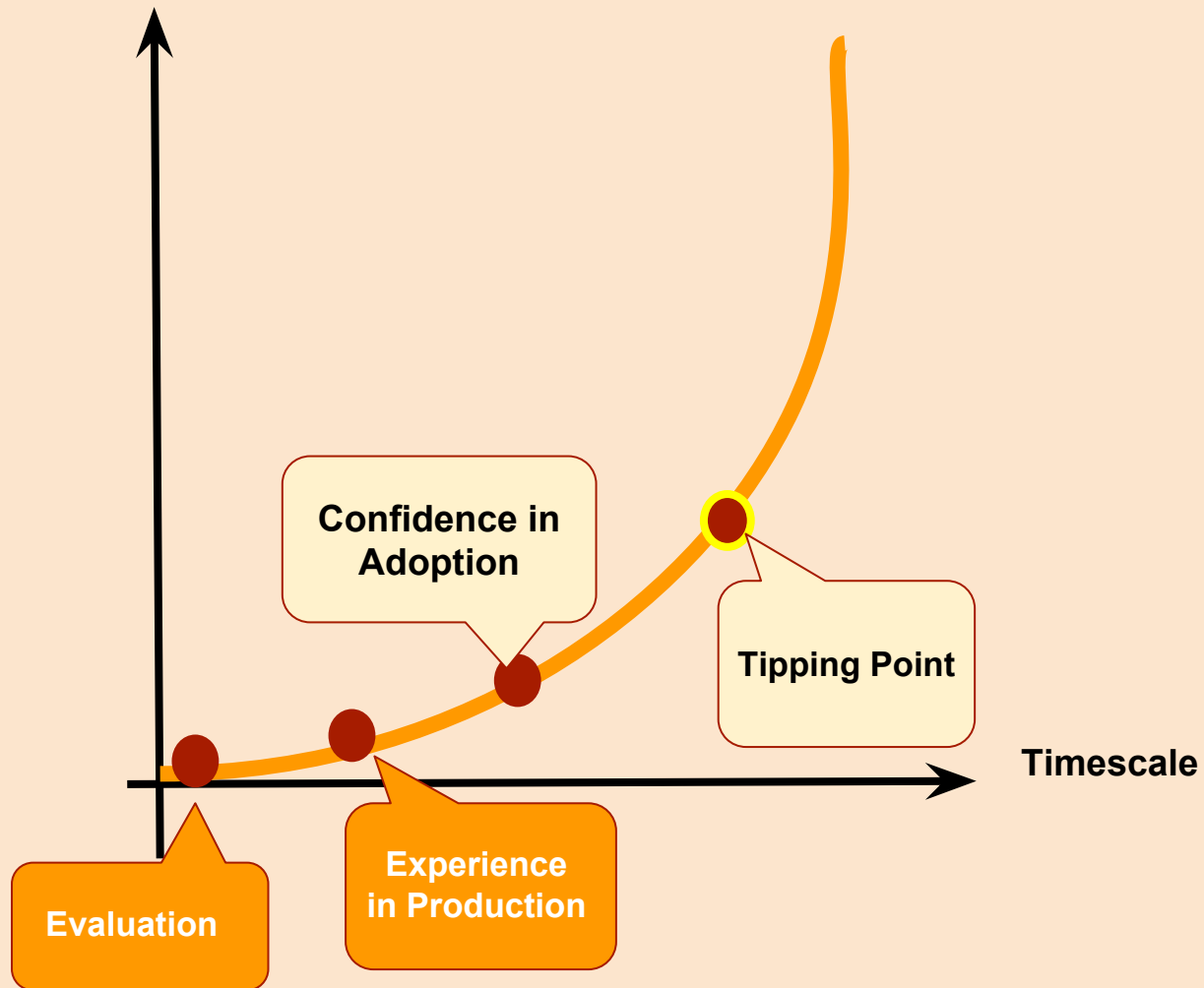
Share  |     

Key takeaways

- Install elements of western culture first
- Use Value Stream Mapping to help break the cultural barrier
- Get upper management involved
- Hackfest will help reduce the lead-in time
- Understand the impact of cultural differences

Change is hard !

**Maturity
with Devops**



Our Adoption Playbook



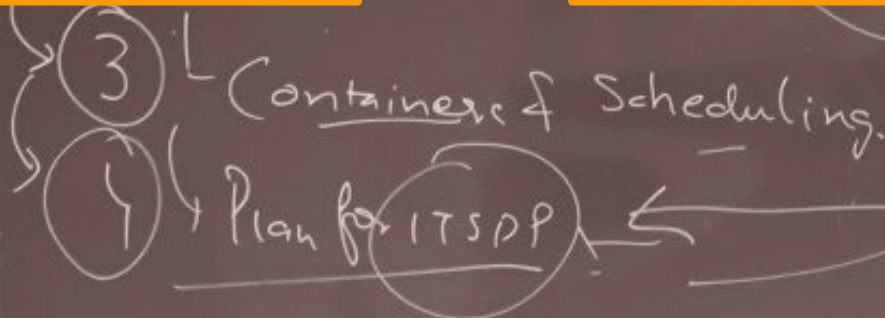
**Confidence in
Technology**



**Compare and
Contrast**



**Create new
Roles**





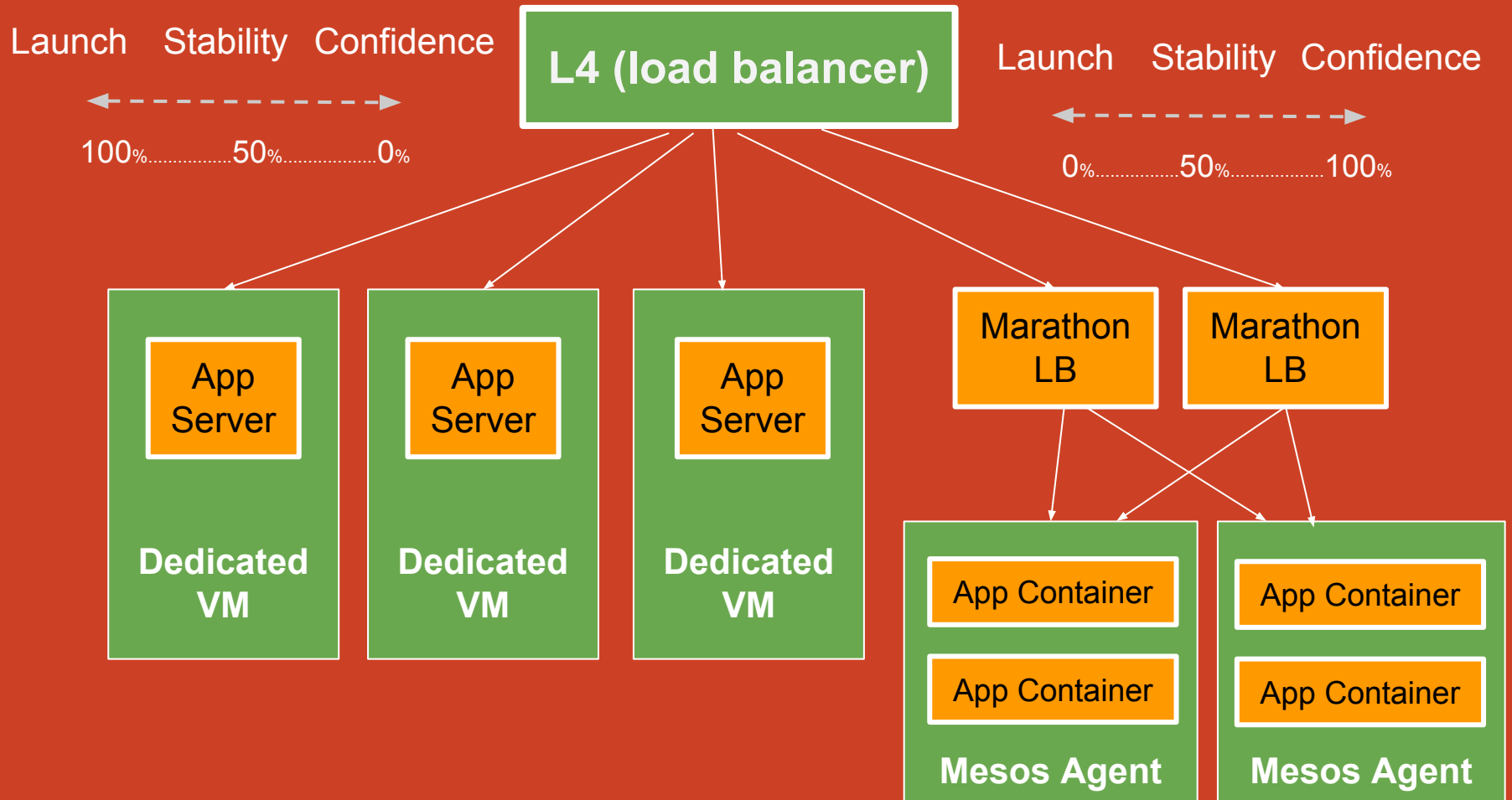
**Unified
Deployment**

**Common
Notifications**

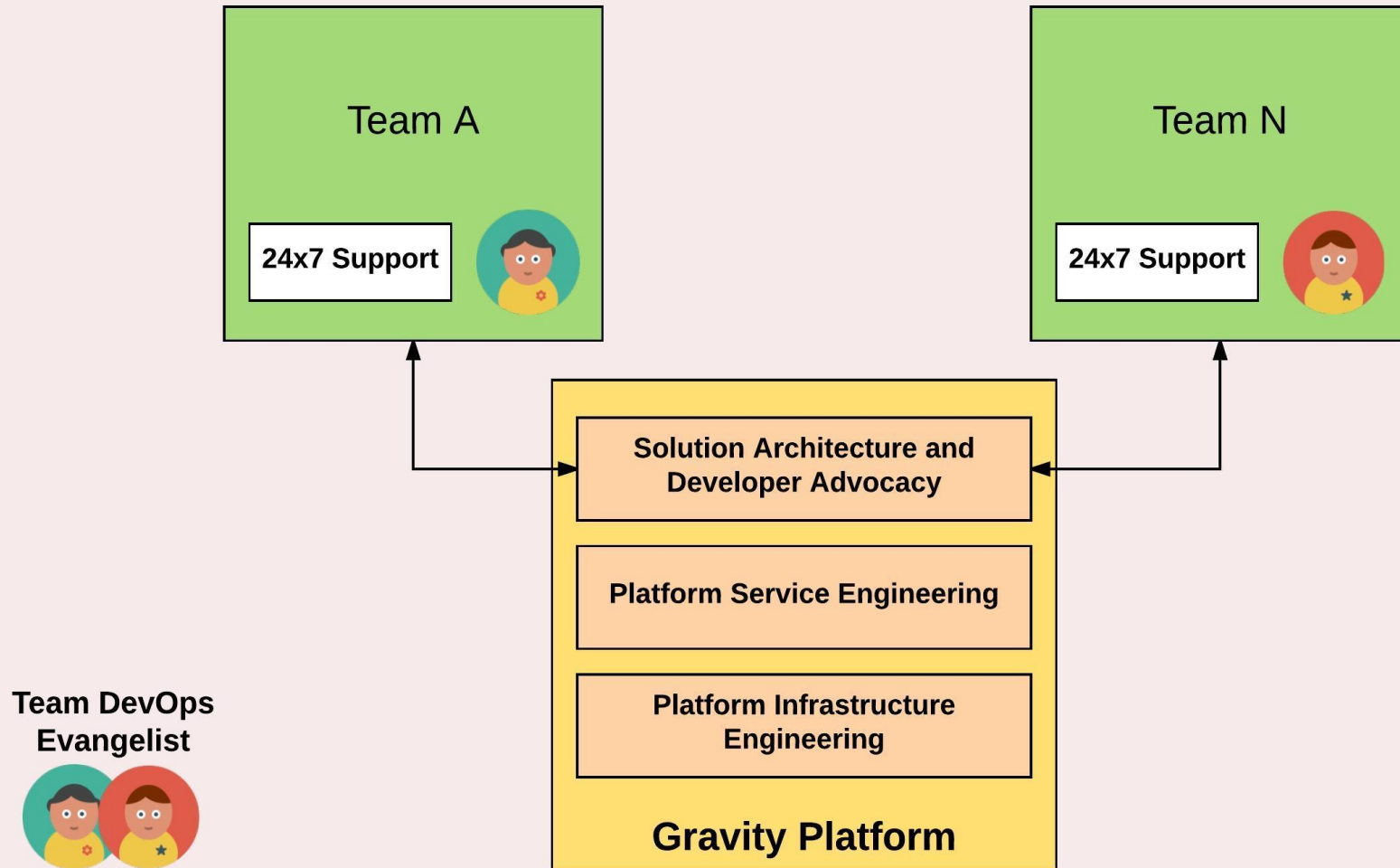
**Centralized
Logging**

**Consolidated
Monitoring**


Compare and Contrast



New Roles



OPERATIONS



Create Self Service
Automate Primitives
Shared Goal with Dev

DEVELOPER



Use Self Service
Ops friendly code
Shared Goal with Ops

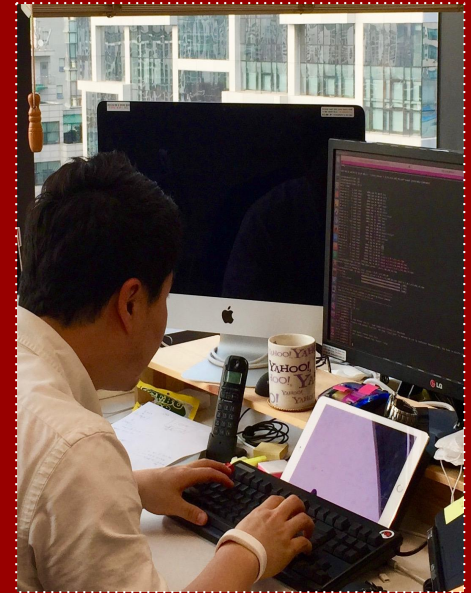
NEW SHARED GOALS



**Reduce Time from
Code checkin to
Production Release**



**Ensure Releases
can be performed
during normal
business hours**



**Reduce unplanned
work and increase
productivity**

1

Allocate VM to a Service



Let Software decide

Reality Check



**Less upfront capacity
allocation meetings, and
more work done !**

2

Availability and
Tolerance = manual
mgmt.



Let Software
decide

Reality Check



**Less manual intervention,
and more time to spend
on improving quality**

3

Time to Production
influenced by lot of
manual monotonous
work



Minimal Manual work
and increased
Self-service

Reality Check



Ops work made more
accessible to Devs via Self
Service

4

Limited reusability
and lack of standards
across teams



Standardize through
Containers and
Deployment
primitives

Reality Check



Reusability across
teams, and more time to
focus on innovation

1

**Less upfront capacity
allocation meetings, and more
work done !**

2

**Less manual intervention, and
more time to spend on
improving quality**

3

**Ops work made more accessible
to Devs. Ops spend more time
improving quality**

4

**Reusability across teams, and
more time to focus on
innovation**



*But DevOps
is also
about Architecture*

Architecture Constraints



**Ops friendly
development**



**Metrics friendly
development**



**Build systems
which are failure
aware**



**Everything is
distributed**

NOT A LONG AGO

BEGINNING OF THE CHANGE

ADOPTING THE CHANGE

THE ROAD AHEAD



Multi-Tenant Cluster

Mix workloads for increased efficiency

Share Common platform primitives

Performance and Isolation guarantees

Avoid Noisy neighbours

Resource Reservation

Isolated Container Registry

Isolated Load balancer and
Discovery

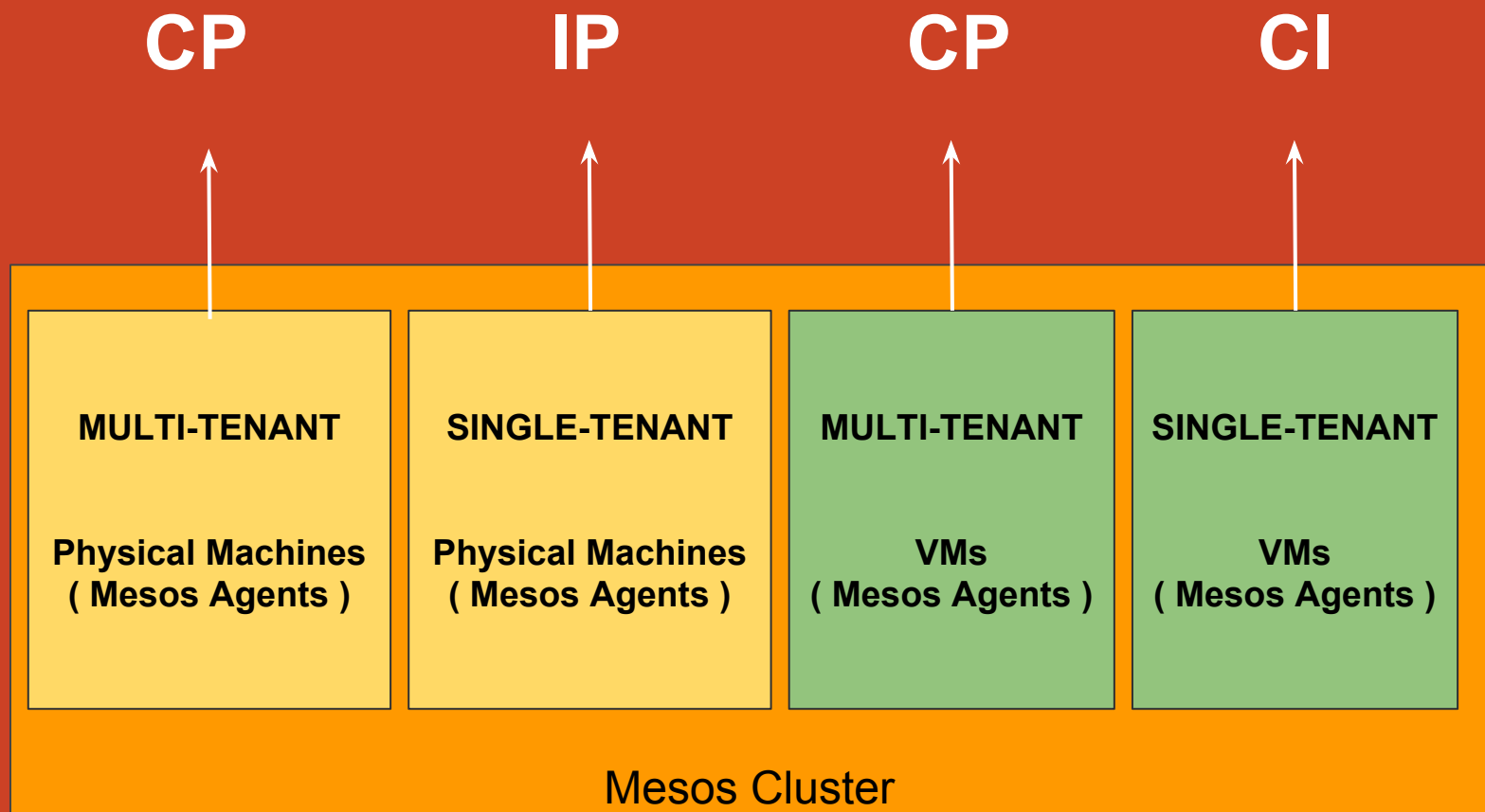
Mesos Agent Selection

Global Workload Allocation

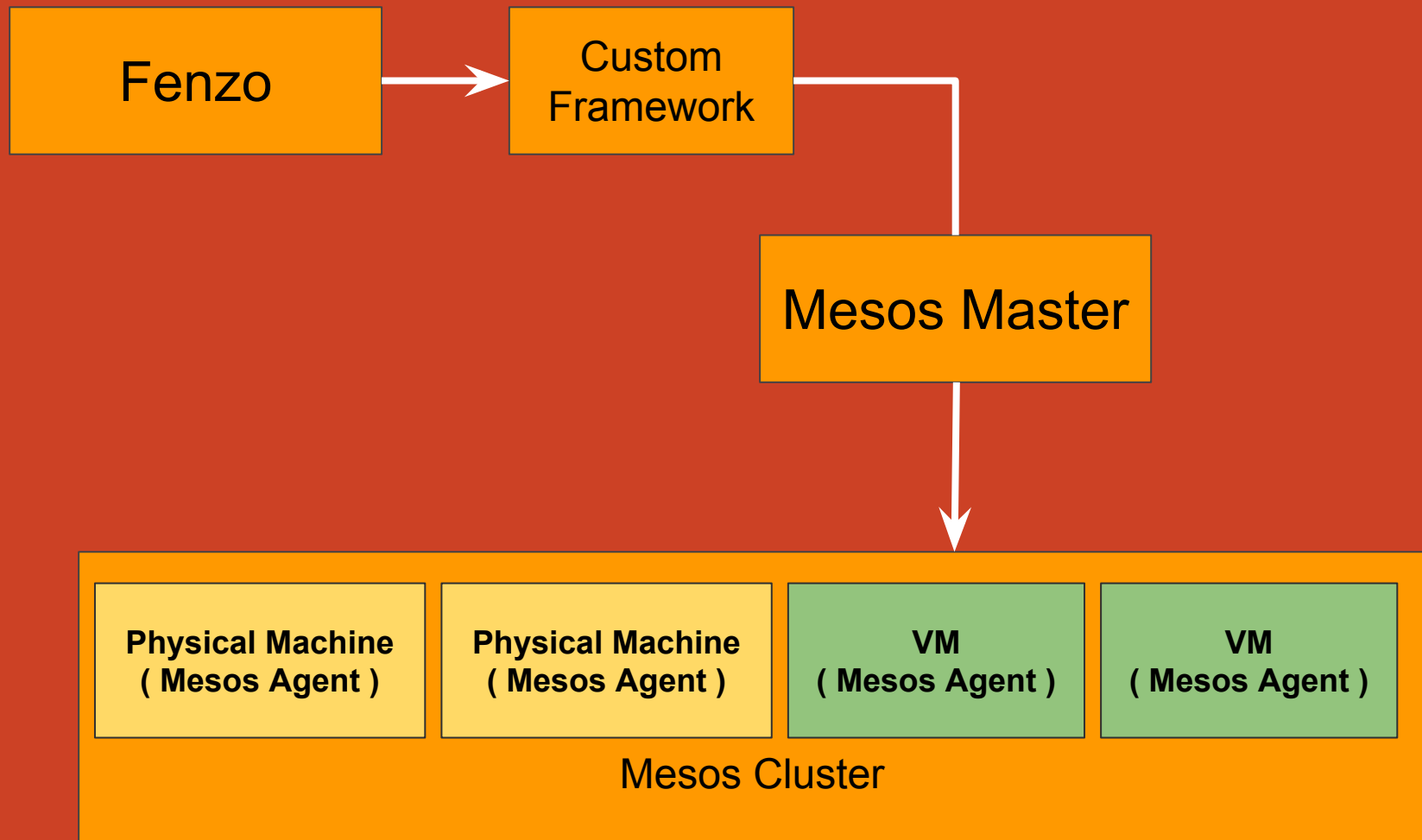
Not all workloads are same !



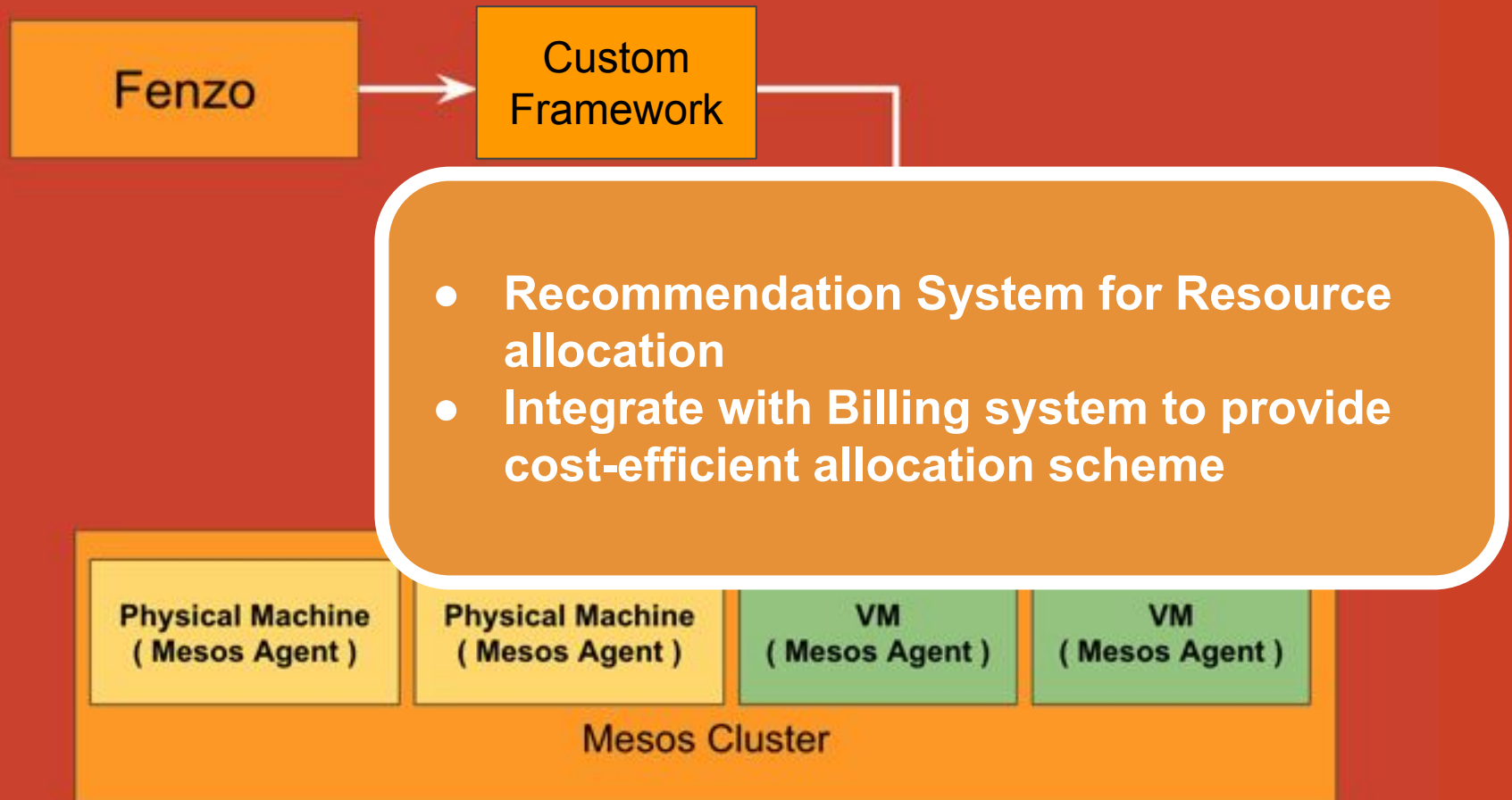
Global Workload Allocation



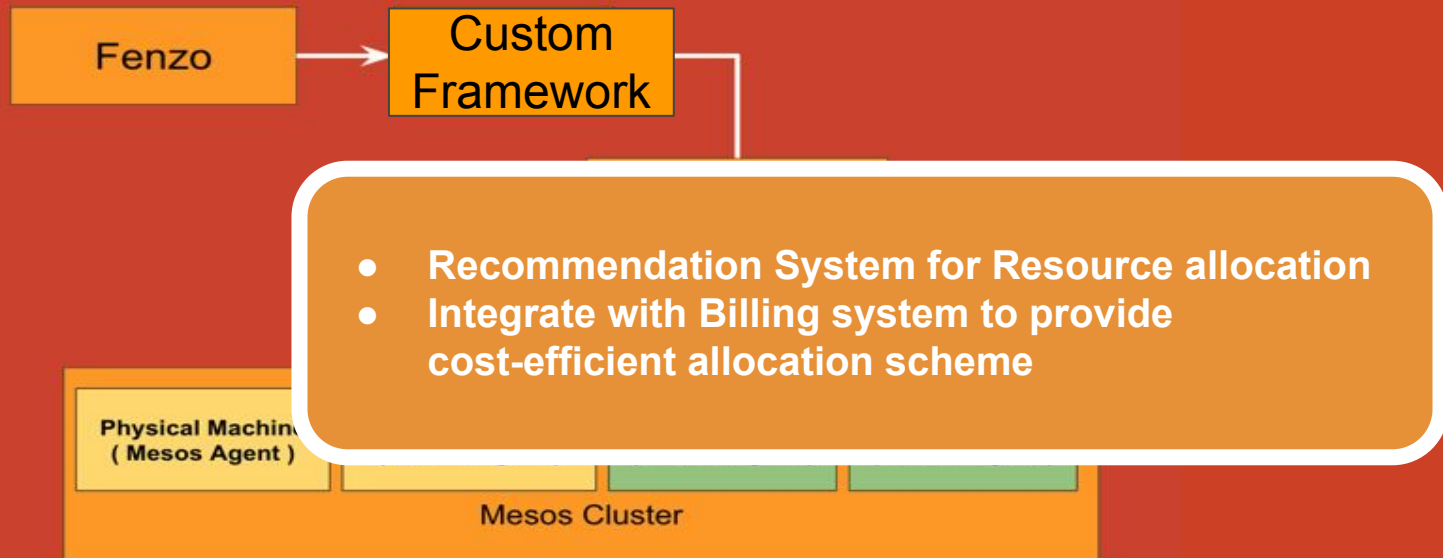
Global Workload Allocation



Global Workload Allocation

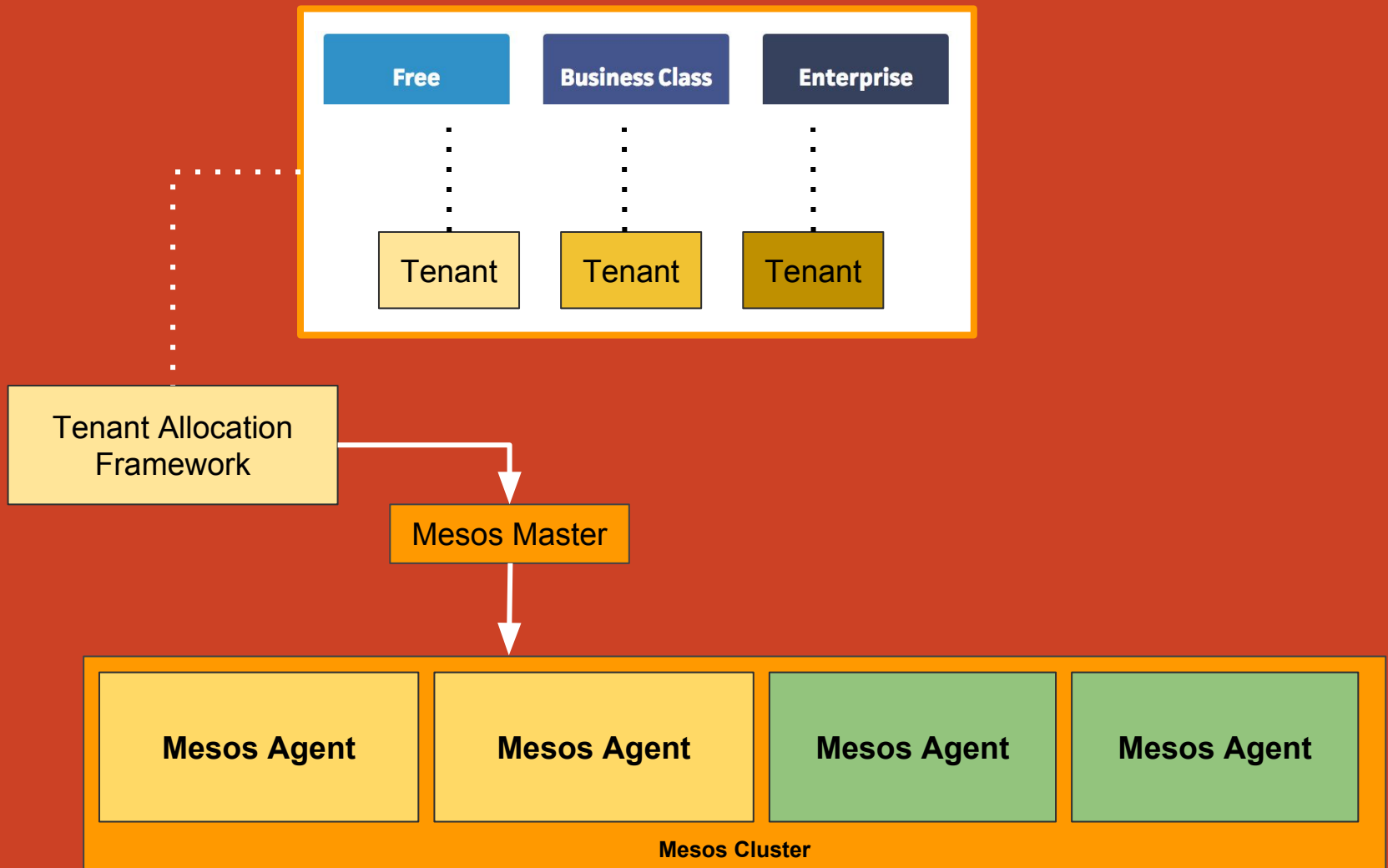


Global Workload Allocation



- Support more advanced bin-packing with Fenzo #2430 (mesosphere/marathon)

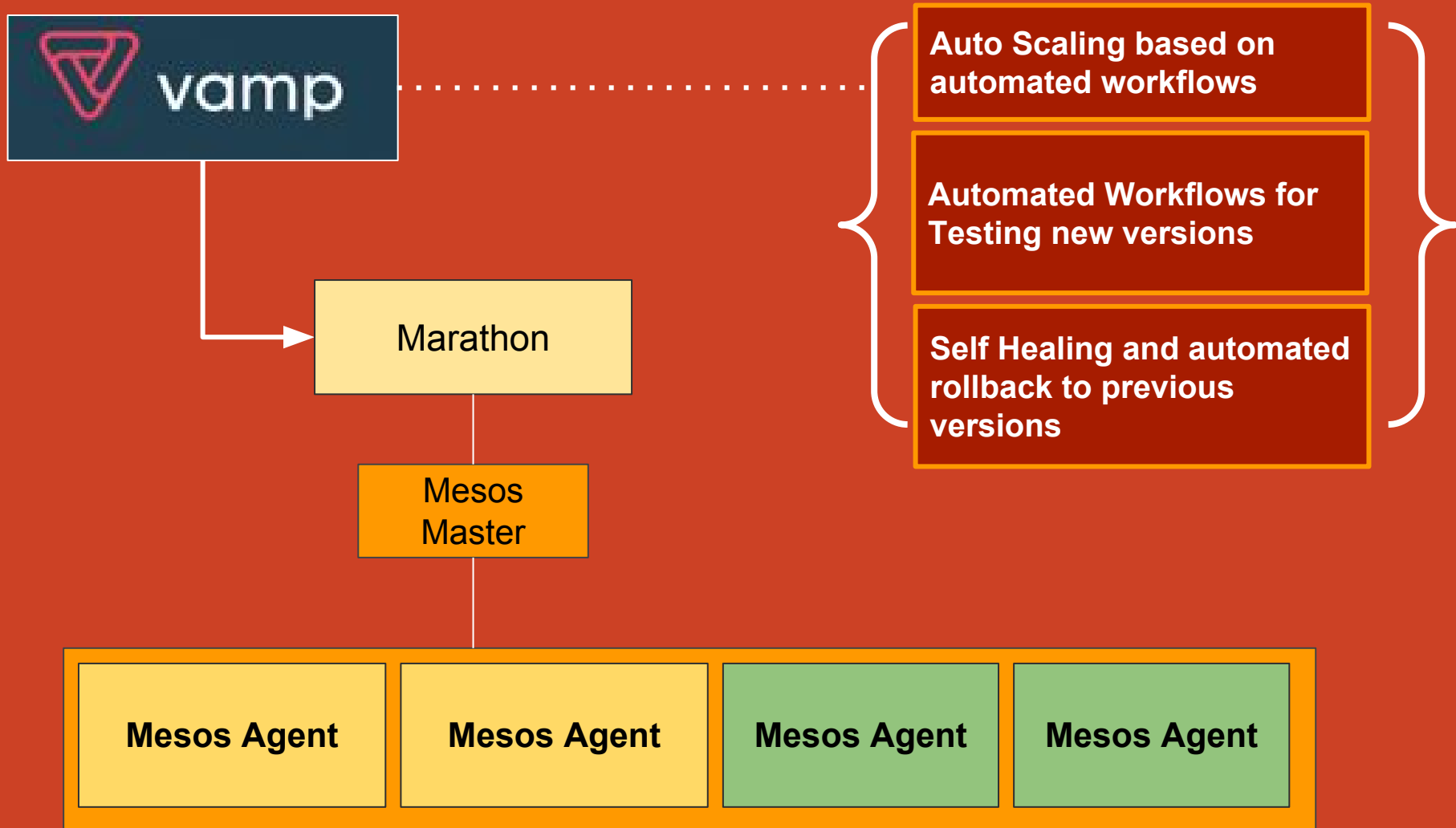
Application aware scheduling



Application aware scheduling



Canary Release and Architecture A/B



THE EARLY YEARS

BEGINNING OF THE CHANGE

ADOPTING THE CHANGE

THE ROAD AHEAD





It's still

1

DAY ONE



Change is possible.

Change is possible. ■



Microsoft



Linux



Microsoft joins **Linux Foundation** - *November 16, 2016*

Thanks

고맙습니다

谢谢

We ♥ Mesos community !

Questions ?



Thanks

고맙습니다

谢谢

We ♥ Mesos community !



Want to help us build this further ?

We are hiring !

