



- Continuous Deployment at Wealthfront eng blog: <http://eng.wealthfront.com/search/label/continuous%20deployment>
- Deployment Infrastructure for Continuous Deployment (David Fortunato) <http://eng.wealthfront.com/2010/05/deployment-infrastructure-for.html>
- Continuous Deployment at IMVU: Doing the impossible fifty times a day <http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/>
- In praise of continuous deployment: The WordPress.com story <http://toni.org/2010/05/19/in-praise-of-continuous-deployment-the-wordpress-com-story/>

Continuous Deployment

Eishay Smith

@eishay @wltheng
#leanstartup #qcon

@eishay @wltheng
#leanstartup #qcon

How much time does it take
your new line of code to meet
customers?

SIX MONTHS TO A YEAR .

More than a year ?

**How much time does it take
your new line of code to meet
customers?**

Six months to a year ?

More than a year

much time does

One to six months ?

ix months to a year

Two weeks to a month ?

One to six months ?

One day to two weeks?

Two weeks to a month

One hour to one day ?

One day to two weeks?

Less than ten minutes ?

One hour to one day ?

Two weeks to a month ?

One to six months ?

Six months to a year ?

More than a year ?

**How much time does it take
your new line of code to meet
customers?**



"Connect Investors with Outstanding Investment Managers"

- Web Based Financial Product
- Heavily Regulated
- Millions of dollars under management
- Deploying to production fifty times a day

What is Continuous Deployment?

- Continuous, successful and repeatable methodology to deploying code
- Automates all steps of taking checked in code and making it run on production servers, used by customers

What is a Startup ?

- Startup is a human organization on a quest
- The quest is to find a market fit between a product and customers
- Finding market fit is done through iteration and learning from customer behavior in the real world
- Market fit must be found before the startup runs out of cash

80/20

- Realize that 80% of what you build will go to waste
- Strive for 80% of the value for 20% of the cost
- Validate before investing more time

Release is a Marketing Concern

Deployment is Engineering's Job

Deployment is reducing code inventory

Deploy Fast Deploy Often

Dozens of Times a Day



Bucket or Hose



Bucket

Traditional Release Organization

Timeline

1-4 Weeks

Bucket

Traditional Release Organization

Timeline

1-4 Weeks





Timeline

1-4 Weeks



Development

- Software organized as a tree
- Engineers work on feature branches



Release Cut

- aka code freeze
- Features get integrated into trunk
- Trunk goes on production
- Trunk is the truth

Stage

Release train on regular cyclic period

QA

QA Ensures correctness of each train

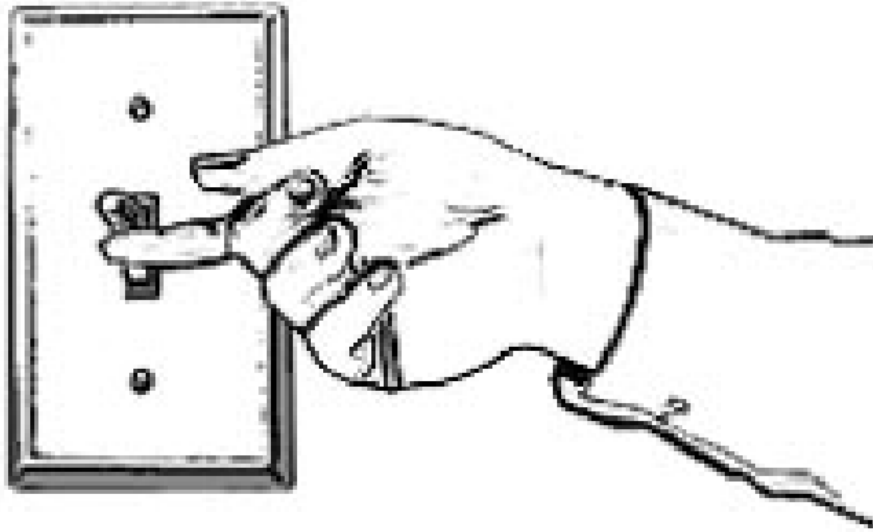
Fix P1 Bugs

P1

Integrate Patches

P1 Bugs get patched back into the train

Release !



Bucket

Traditional Release Organization

Timeline

1-4 Weeks





Bucket or Hose

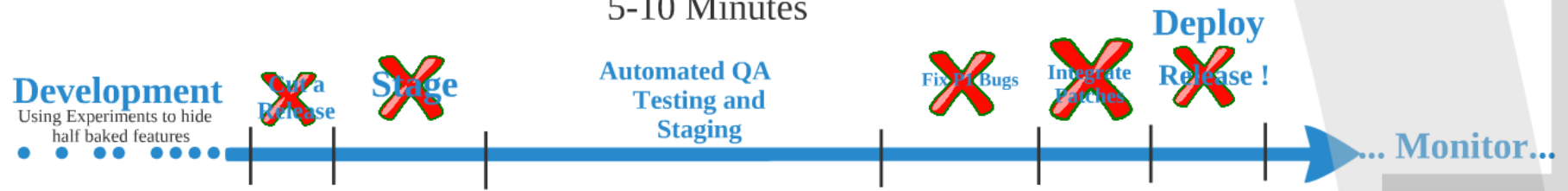


or Hose

Continuous Deployment

Timeline

5-10 Minutes





Timeline

5-10 Minutes

Automated QA

Development

Using Experiments to hide
half baked features



ent

hide

S



~~Cut a
Release~~

~~Sta~~

it a
ease

Stage

Automated QA Testing and Staging

Fix P1 Bugs

**In
P**



Bugs



Integrate
Patches



Rele

rate
nes

Release !



Deploy

Release !



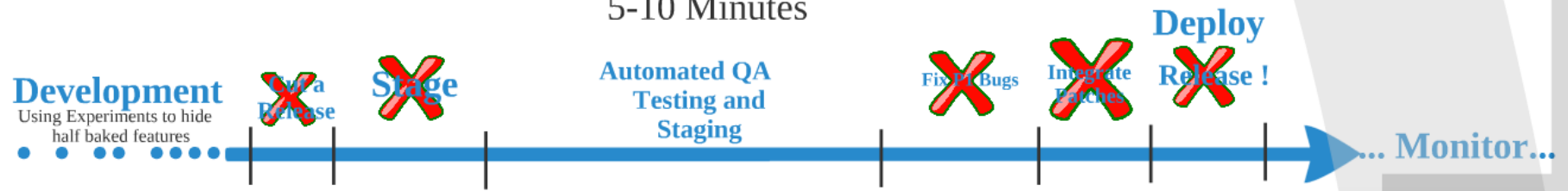


or Hose

Continuous Deployment

Timeline

5-10 Minutes





Bucket or Hose

Story Time

Based on a real life story



Based on a real life story









- Quick iterations
- Obsolete process, e.g. 'waiting a release'
- Reduce risk
- No throwing code over the wall

- Code in development matches production
- No need to switch branches



- Code in development matches production
- No need to switch branches

- Quick iterations
- Obsoletes process, e.g. "cutting a release"
- Reduce risk
- No throwing code over the wall





- Quick iterations
- Obsolete process, e.g. 'waiting a release'
- Reduce risk
- Not throwing code over the wall

- Code in development matches production
- No need to switch branches



20 minutes



Faster iteration lead to customer - product market fit

Faster iteration lead to
customer - product market fit

Story Time

Based on a real life story



Wealthfront's Architecture

- Service Oriented Arcitecture
- Vertical sharding
- Coordination using ZooKeeper
- Data interchange using JSON and Protobuf
- Data store in MySql, Redis and Voldemort
- Guice for IoC
- Everything uses the same platform © Kawala



Open
Source

Kawala

Functional Query Engine
Inspired by functional languages (Scala)

Wealthfront's Architecture

- Service Oriented Arcitecture
- Vertical sharding
- Coordination using ZooKeeper
- Data interchange using JSON and Protobuf
- Data store in MySql, Redis and Voldemort
- Guice for IoC
- Everything uses the same platform © Kawala

Scale and Availability

Typical Stack

- Clustered services storage
- Replicated databases (MySQL, Redis)
- Caching (e.g. memcached)
- Denormalized Data

Testing

- Hudson
- Selenium
- Static analysis (PMD, FindBugs)
- DbUnit
- In memory DB
- Enhanced jUnit Runners

Deploy Fast Deploy Often

Dozens of Times a Day

Release is a Marketing Concern
 Deployment is Engineering's Job
CapEx/OpEx is reducing order frequency

80/20
 • Realize that 80% of what you build will go to waste
 • Strive for 80% of the value for 20% of the cost
 • Validate before investing more time

What is a Startup ?
 • Startup is a human organization on a quest
 • The quest is to find a market fit between a product and customers
 • Finding market fit is done through iteration and learning from customer behavior in the real world
 • Market fit must be found before the startup runs out of cash

What is Continuous Deployment?
 • Continuous, successful and repeatable methodology to deploying code
 • Automates all steps of taking checked in code and making it run on production servers, used by customers

"Connect Investors with Outstanding Investment Managers"
 • Additional financial traction
 • More stable long term
 • Millions of dollars under investment
 • Deploying to production 4 times a day

How much time does it take your new line of code to meet customers?
 More than a year ?

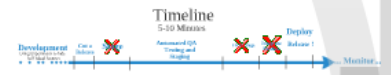


Bucket or Hose

Traditional Release Organization



Continuous Deployment



Story Time



Wealthfront's Architecture

- Service Oriented Architecture
- Vertical sharding
- Coordination using ZooKeeper
- Data interchange using JSON and Protobuf
- Data store in MySQL, Redis and Voldemort
- Guice for IoC
- Everything uses the same platform!

Scale and Availability

- Clustered services storage
- Replicated databases (MySQL, Redis)
- Caching (e.g. memcached)
- Denormalized Data

Testing

- E2E test
- Selenium
- Static analysis (PMD, FindBugs)
- DDD
- In Memory DB
- Embedded JUnit runners

Continuous Deployment

Enchay Smith

Continuous Deployment

Eishay Smith

@eishay @wltheng
#leanstartup #qcon

DEPLOYMENT MANAGER			
ZOOKEEPER		P-RUNNER	
Type	Interval	Version	State
PROD	15:34:00	21476	DEPLOYING
DEV	15:34:00	21476	DEPLOYING

Client	Version	State	Interval
PROD	21476	DEPLOYING	15:34:00
DEV	21476	DEPLOYING	15:34:00

Small, frequent commits

Small, frequent commits

Deploying either using the UI or more commonly using hash tags like "#release:UM" in the commit comment

Forward and Backward compatibility

Trivial reverts
Trivial rollbacks

Life of a Deployment

- Managed by a deployment manager
- State is propagated by Zookeeper
- Exponentially increase deployment group size
- Increased monitoring during deployment
- Self Test
- Automated rollbacks



Stable trunk

- No branch switch
- No merges
- No conflicts

Stable trunk

- No branch switch
- No merges
- No conflicts

Small, frequent commits

Stable trunk

Small, frequent commits

Deploying either using the UI or more commonly using hash tags like "#release:UM" in the commit comment

Stable trunk

Small, frequent commits

Deploying either using the UI or more commonly using hash tags like "#release:UM" in the commit comment

Forward and Backward compatibility

Stable trunk

Small, frequent commits

Deploying either using the UI or more commonly using hash tags like "#release:UM" in the commit comment

Forward and Backward compatibility

Trivial reverts

Stable trunk

Small, frequent commits

Deploying either using the UI or more commonly using hash tags like "#release:UM" in the commit comment

Forward and Backward compatibility

Trivial reverts

Trivial rollbacks

DEPLOYMENT MANAGER

ACTIVE DEPLOYMENTS

Type	Started	Revision	State	Cancel
ROBI	17:39:46	27478	BUILDING ↘	Cancel
HTF	17:37:14	27477	DEPLOYING ↘	65% done

htf0 [htf1](#) [htf2](#)

HUDSON

Build	Revision	State
Last	27478	SUCCESS
Last Completed	27478	SUCCESS
Last Successful	27478	SUCCESS

9 builds since the last failure

CLUSTER STATUS

Cluster	Release	Status	Version(#)	Updated
AA	aa0 aa1 aa2		26125(2)	17:37:24
AM	am0 am1 am2		27435(1)	17:37:34
BI	bi0 bi1		27467(1)	17:37:44
DM	dm0		27471(1)	17:37:54
HTF	htf0 htf1 htf2		27269(1)	17:38:04
IM	im0 im1 im2		27349(1)	17:38:14
KFE	kfe0 kfe1		27425(2)	17:37:14
NL	nl0 nl1		27144(1)	17:38:24
OD	od0 od1		26005(1)	17:38:34
PM	pm4 pm5 pm2 pm3 pm0 pm1 pm10 pm8 pm9 pm6 pm7		27222(11)	17:39:25
ROBI	robi0 robi1		27452(1)	17:38:44
SC	sc0 sc1 sc2 sc3		26588(1)	17:38:54
TF	tf0 tf1 tf2		26869(1)	17:39:04
UM	um0 um1 um2 um3		27439(1)	17:39:35
WL	wl0 wl1		26866(2)	17:39:14

LATEST COMMITS

	Revision	Committer	Message
✓	27478 ☒	david	
✓	27477 ☒	julien	
✓	27476 ☒	pascal	
✗	27475 ☒	eishay	
✗	27473 ☒	pascal	
✓	27471 ☒	eishay	
✓	27470 ☒	david	

RECENTLY COMPLETED RELEASES

Cluster	At	State
DM	15:21:29	RELEASED
UM	15:19:18	RELEASED
AM	15:18:53	ROLLED_BACK
UM	15:14:42	ROLLED_BACK
BI	15:10:07	RELEASED
ROBI	15:07:38	RELEASED

LATEST COMMITS

	Revision	Committer	Message
✓	27478 ☒	david	
✓	27477 ☒	julien	
✓	27476 ☒	pascal	
✗	27475 ☒	elshay	
✗	27473 ☒	pascal	
✓	27471 ☒	elshay	
✓	27470 ☒	david	

HUDSON

Build	Revision	State
<u>Last</u>	<u>27478</u>	SUCCESS
<u>Last Completed</u>	<u>27478</u>	SUCCESS
<u>Last Successful</u>	<u>27478</u>	SUCCESS

9 builds since the last failure





DEPLOYMENT MA

ACTIVE DEPLOYMENTS

Type	Started	Revision	State	Cancel
ROBI	17:39:46	<u>27478</u>	BUILDING ↘	<u>Cancel</u>
HTF	17:37:14	<u>27477</u>	DEPLOYING ↘	65% done

htf0 **htf1** htf2

CLUSTER STATUS

Cluster	Release	Status	Version(%)	Updated
AA	 	aa0 aa1 aa2	26125(2)	17:37:24
AM	 	am0 am1 am2	27435(1)	17:37:34

RECENTLY COMPLETED RELEASES

Cluster	At	State
DM	15:21:29	RELEASED
UM	15:19:18	RELEASED
AM	15:18:53	ROLLED_BACK
UM	15:14:42	ROLLED_BACK
BI	15:10:07	RELEASED
ROBI	15:07:38	RELEASED

CLUSTER STATUS

Cluster	Release	Status	Version(¶)	Updated
AA	 	aa0 aa1 aa2	26125(2)	17:37:24
AM	 	am0 am1 am2	27435(1)	17:37:34
BI	 	bi0 bi1	27467(1)	17:37:44
DM	 	dm0	27471(1)	17:37:54
HTF	 	htf0 htf1 htf2	27269(1)	17:38:04
IM	 	im0 im1 im2	27349(1)	17:38:14
KFE	 	kfe0 kfe1	27425(2)	17:37:14
NL	 	nl0 nl1	27144(1)	17:38:24
OD	 	od0 od1	26005(1)	17:38:34
PM	 	pm4 pm5 pm2 pm3 pm0 pm1 pm10 pm8 pm9 pm6 pm7	27222(11)	17:39:25
ROBI	 	robi0 robi1	27452(1)	17:38:44
SC	 	sc0 sc1 sc2 sc3	26588(1)	17:38:54
TF	 	tf0 tf1 tf2	26869(1)	17:39:04
UM	 	um0 um1 um2 um3	27439(1)	17:39:35
WL	 	wl0 wl1	26886(2)	17:39:14



Clus
DM
UM
AM
UM

DEPLOYMENT MANAGER

ACTIVE DEPLOYMENTS

Type	Started	Revision	State	Cancel
ROBI	17:39:46	27478	BUILDING ↘	Cancel
HTF	17:37:14	27477	DEPLOYING ↘	65% done

htf0 [htf1](#) [htf2](#)

HUDSON

Build	Revision	State
Last	27478	SUCCESS
Last Completed	27478	SUCCESS
Last Successful	27478	SUCCESS

9 builds since the last failure

CLUSTER STATUS

Cluster	Release	Status	Version(%)	Updated
AA	aa0 aa1 aa2		26125(2)	17:37:24
AM	am0 am1 am2		27435(1)	17:37:34
BI	bi0 bi1		27467(1)	17:37:44
DM	dm0		27471(1)	17:37:54
HTF	htf0 htf1 htf2		27269(1)	17:38:04
IM	im0 im1 im2		27349(1)	17:38:14
KFE	kfe0 kfe1		27425(2)	17:37:14
NL	nl0 nl1		27144(1)	17:38:24
OD	od0 od1		26005(1)	17:38:34
PM	pm4 pm5 pm2 pm3 pm0 pm1 pm10 pm8 pm9 pm6 pm7		27222(11)	17:39:25
ROBI	robi0 robi1		27452(1)	17:38:44
SC	sc0 sc1 sc2 sc3		26588(1)	17:38:54
TF	tf0 tf1 tf2		26869(1)	17:39:04
UM	um0 um1 um2 um3		27439(1)	17:39:35
WL	wl0 wl1		26866(2)	17:39:14

LATEST COMMITS

	Revision	Committer	Message
✓	27478 ☒	david	
✓	27477 ☒	julien	
✓	27476 ☒	pascal	
✗	27475 ☒	eishay	
✗	27473 ☒	pascal	
✓	27471 ☒	eishay	
✓	27470 ☒	david	

RECENTLY COMPLETED RELEASES

Cluster	At	State
DM	15:21:29	RELEASED
UM	15:19:18	RELEASED
AM	15:18:53	ROLLED_BACK
UM	15:14:42	ROLLED_BACK
BI	15:10:07	RELEASED
ROBI	15:07:38	RELEASED

Life of a Deployment

- Managed by a deployment manager
- State is propagated by Zookeeper
- Exponentially increase deployment group size
- Increased monitoring during deployment
- Self Test
- Automated rollbacks



Announce:
An animal lets the ZooKeeper know it woke up
and available

Un-Announce:
An animal lets the ZooKeeper know it is not
available

er




Announce:

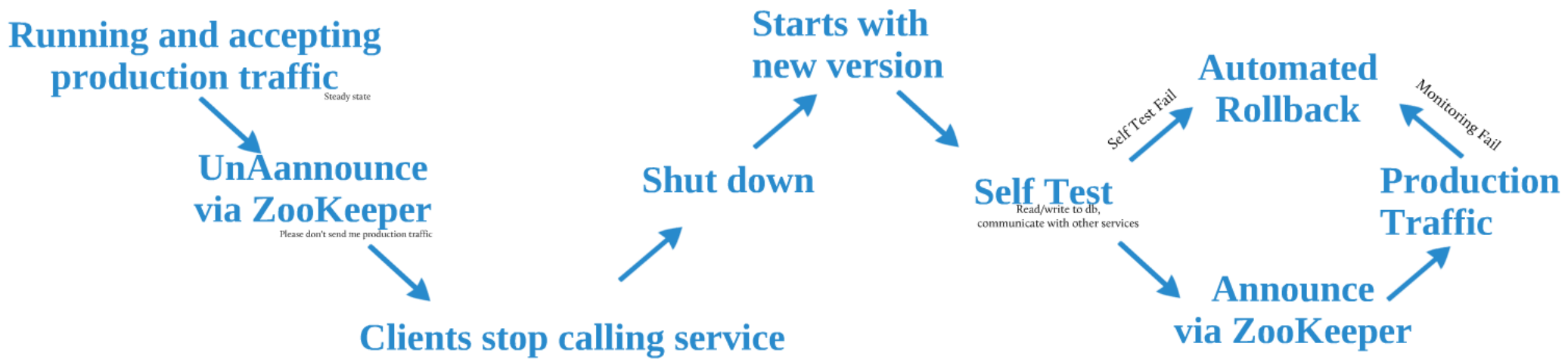
An animal lets the ZooKeeper know it woke up and available

Un-Announce:

An animal lets the ZooKeeper know it is not available

at group size

- Managed by a deployment manager
- State is propagated by Zookeeper 
- Exponentially increase deployment group size
- Increased monitoring during deployment
- Self Test
- Automated rollbacks





Running and accepting production traffic

Steady state



UnAannounce
via ZooKeeper

Production traffic

Steady state



UnAannounce via ZooKeeper

Please don't send me production traffic



nounce Keeper

n't send me production traffic



Shut down



Clients stop calling service



Shut down



**Starts with
new version**



Self

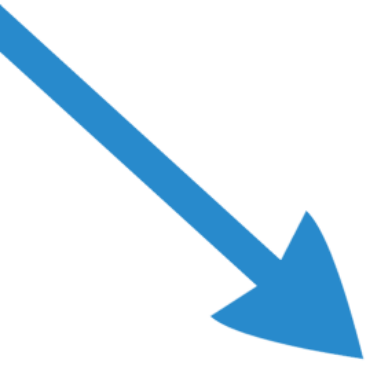
Self Test

Read/write to db,

communicate with other services

ices

group
Traffic



Announce
via ZooKeeper

ing Fail

Production Traffic

Self Test

Read/write to db,
communicate with other services


Self Test Fail

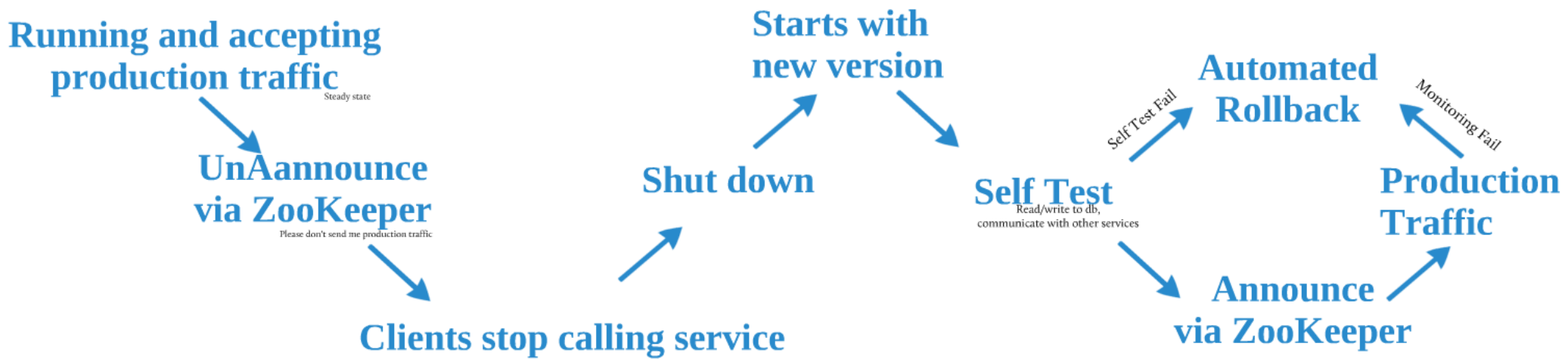
**Automated
Rollback**

Monitoring Fail

**Production
Traffic**

**Announce
via ZooKeeper**

- Managed by a deployment manager
- State is propagated by Zookeeper 
- Exponentially increase deployment group size
- Increased monitoring during deployment
- Self Test
- Automated rollbacks



How long does it take
to write code to meet
the needs of users?

Testing

- Hudson
- Selenium
- Static analysis (PM)
- DbUnit
- In memory DB
- Enhanced jUnit R

Continuous Deployment

Eishay Smith

@eishay @wltheng
#leanstartup #qcon

Forward and Backward compatibility
Trivial reverts
Trivial rollbacks



Life of a Deployment

- Managed by deployment manager
- State is propagated by Zookeeper
- Essentially no new deployment group size
- Increased availability during deployment
- Self Test
- Automated rollbacks



Immune System

Multi level monitoring



- Automatic rollback with the slightest suspicion of something wrong
- Automatically annotate graphs

Philosophy

Multi level monitoring

Production Self Test

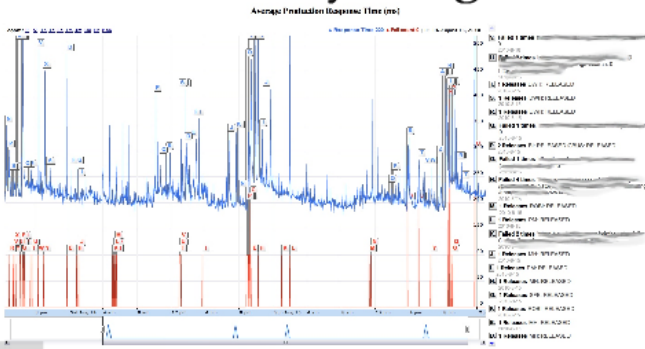
- Could be perceived as part of integration tests
- Checks a service does before it announce itself

Production Self Tests

- Could be perceived as part of integration tests
- Checks a service does before it announce itself

Engineering

We know everything !

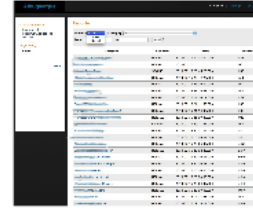
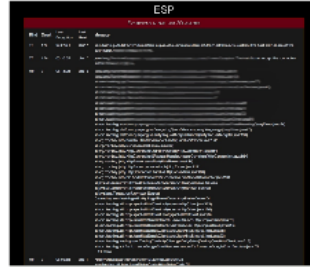
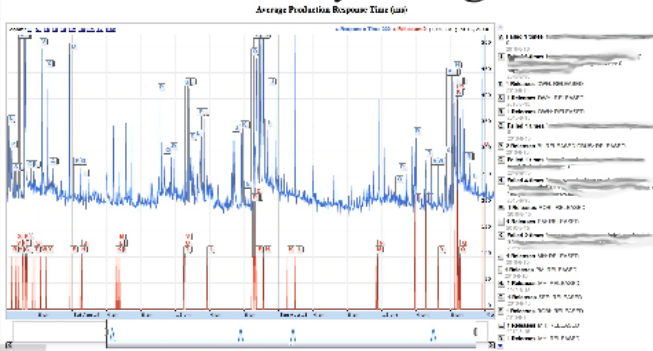


A screenshot of a terminal window with a black background and white text. The title bar says 'ESP'. The text inside appears to be system logs or configuration files, with some lines highlighted in red.

A screenshot of a data table or dashboard. It has a dark background with white text. The table has several columns and rows of data. The title bar says 'ESP'.

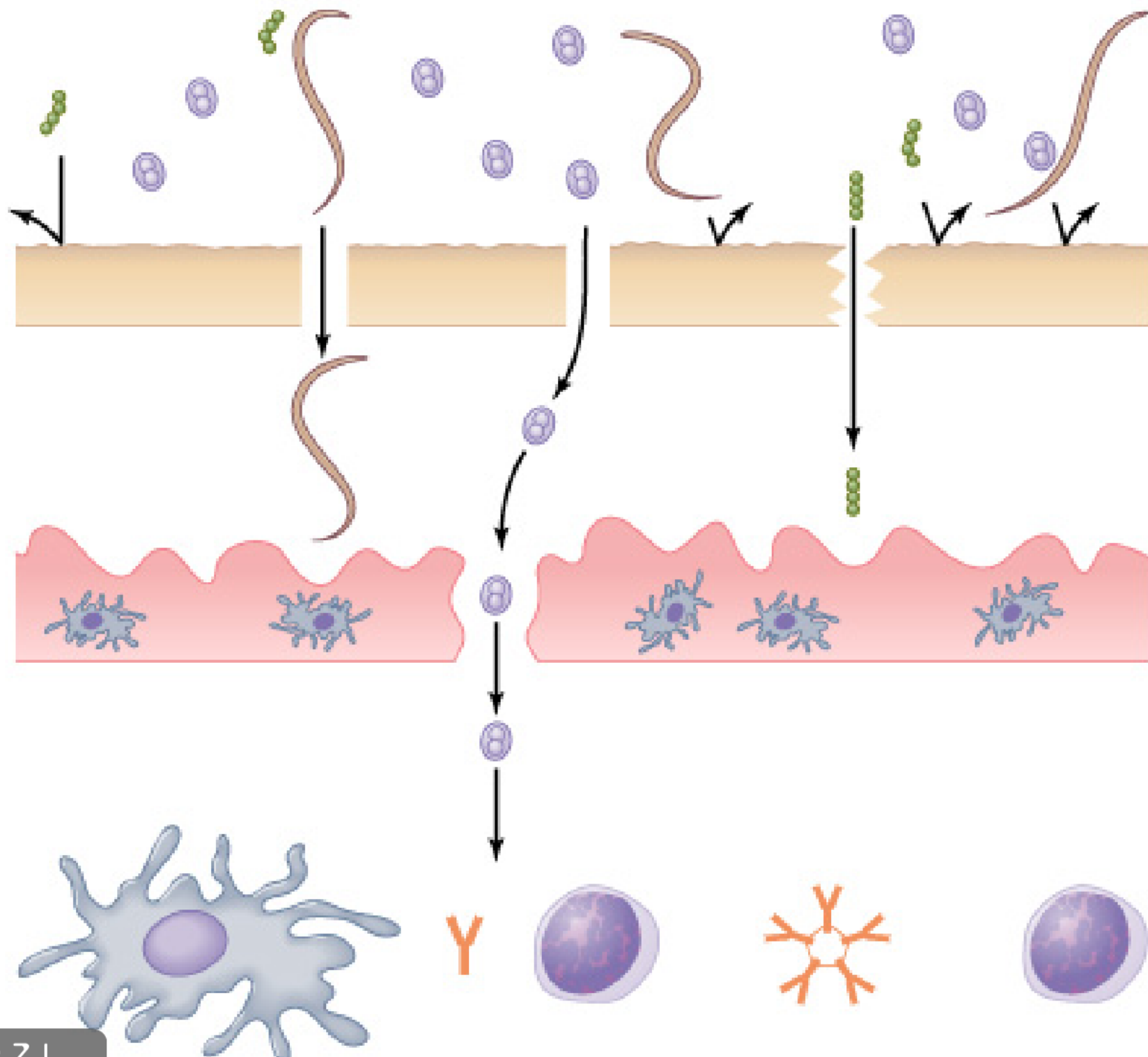
LIGHTNING

We know everything !



Business aka online KPI

Much much more
(can't disclose)



- Automatic rollback with the slightest suspicion of something wrong
- Automatically annotate graphs
- Prefer Business metrics
- Monitor Statistical deviation, not absolute values



Philosophy

- Lines of defense will ever be broken
- Design for swift mitigation

- methodology to deploying code
- Automates all steps of taking checked in code and making it run on production servers, used by customers

"Connect Investors with Outstanding Investment Managers"

- Web Based Financial Product
- Highly Regulated
- Millions of dollars under management
- Deploying to production 8 times a day

Wealthfront's Architecture

- Service Oriented Architecture
- Vertical sharding
- Coordination using ZooKeeper
- Data interchange using JSON and Protobuf
- Data store in MySQL, Redis and Voldemort
- Guice for IoC
- Everything uses the same platform!

Scale and Availability

- Clustered services storage
- Replicated databases (MySQL, Redis)
- Caching (e.g. memcached)
- Denormalized Data

Testing

- Mockito
- Selenium
- Static analysis (PMD, FindBugs)
- DSLs
- In memory DB
- Enhanced JUnit Runner

More than a year ?
 How much time does it take your new line of code to meet customers?

Continuous Deployment

Eishay Smith

Immune System

Multi level monitoring
 Production Self Test
 Engineering
 Business

- Automatic rollback with the slightest suspicion of something wrong
- Automatically generate graphs
- Dry: Restraints create
- Monitor Statistical deviations, not absolute values

Philosophy

- Lines of defense will ever be broken
- Design for swift mitigation

Continuous Integration

- Standard integration tests
- Running all services and shutting them down in a staging environment
- After each and every commit!
- No Broken Windows!
- Isolate new deployment in production
- Gradually shift load to fresh services

Test Driven Development

Only automated testing matters!

Test each commit!

If it isn't tested, it isn't finished or correct

Write testable code
 Embrace abstractions



Empowers engineers to change anything and scale the team

More cost effective than debugging

Facilitates continuous refactoring,

Attracts the right kind of engineers

TESTING

Only automated testing matters !

No QA Team

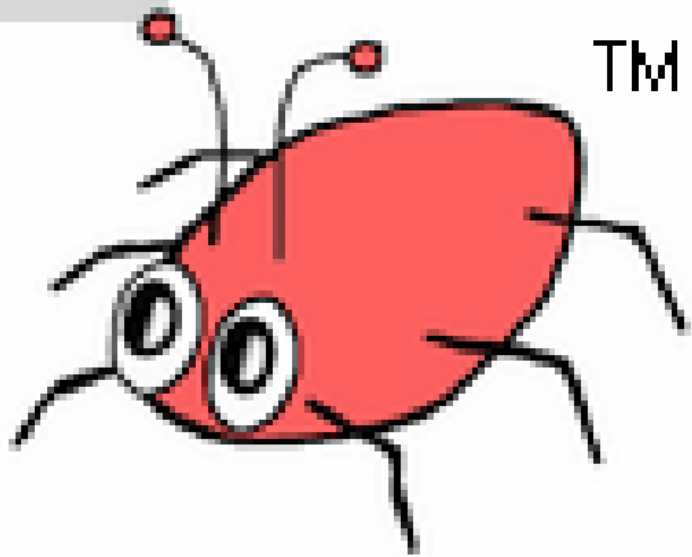
Test each commit!

If it isn't tested,
it isn't finished or correct

Write testable code

Embrace abstractions

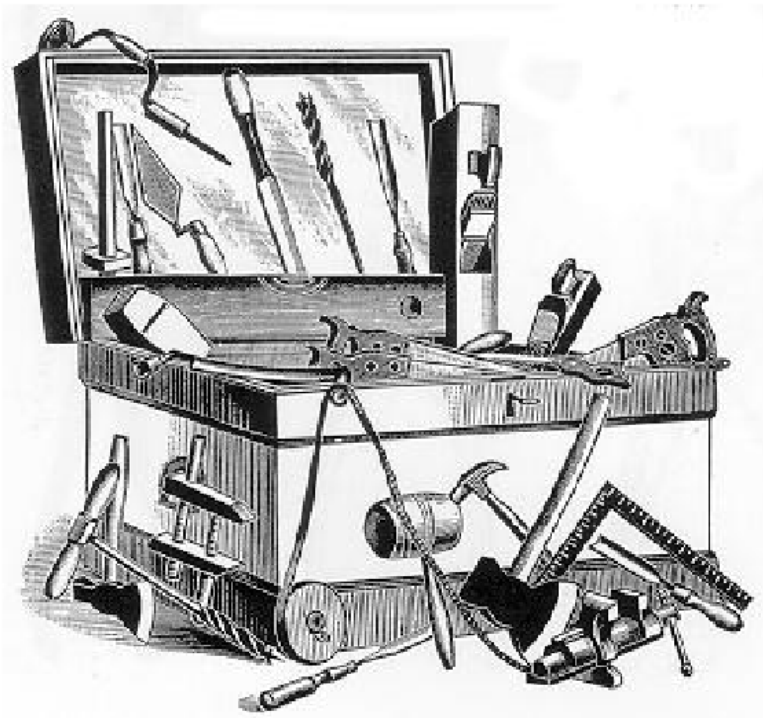
No QA Team



TM

Pmd

DON'T SHOOT THE MESSENGER



Empowers engineers to change anything
and scale the team

More cost effective than debugging

Facilitates continuous refactoring,
allows the code to get better with age

Principles

Attracts the right kind of engineers

your new line of code to meet customers?

- Testing**
- Manual
 - Automated
 - Static analysis (PMD, FindBugs)
 - TDD
 - Integration DB
 - Enhance JUnit Runners

Continuous Deployment

Esbay Smith



Immune System

- Multi level monitoring**
- Production Self Test**
- Engineering**
- Business**

- Accurate and fast with the highest perspective of searching logs
- Accurately measure graphs
- Order to create metrics
- Monitor historical decisions, not *done* or *done*

- Philosophy**
- Lines of defense will ever be broken
 - Design for swift mitigation

Continuous Integration

- Standard integration tests
 - Running all services and shutting them down in a staging environment
 - After each and every commit!
 - No Broken Windows!
- Inhibit new deployment in production
 - Gradually shift load to fresh services

Test Driven Development

- Only automated testing matters!**
- Test each commit!
- If it isn't tested, it isn't finished or correct
- Write testable code
- Embrace abstractions



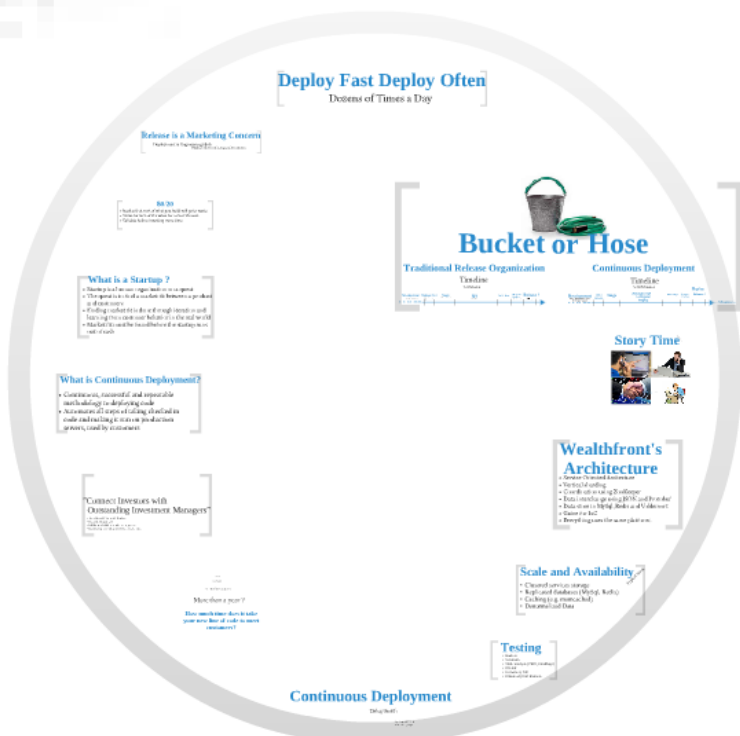
- Empowers engineers to change anything and scale the team
- More cost effective than debugging
- Facilitates continuous refactoring, allows the code to get better with age

Attracts the right kind of engineers

Culture

- Quality driven culture
- Visual and audio signals when anything is broken
- Accountability across the board

**Quality by design
Not by process**



Immune System

Continuous Integration

Test Driven Development

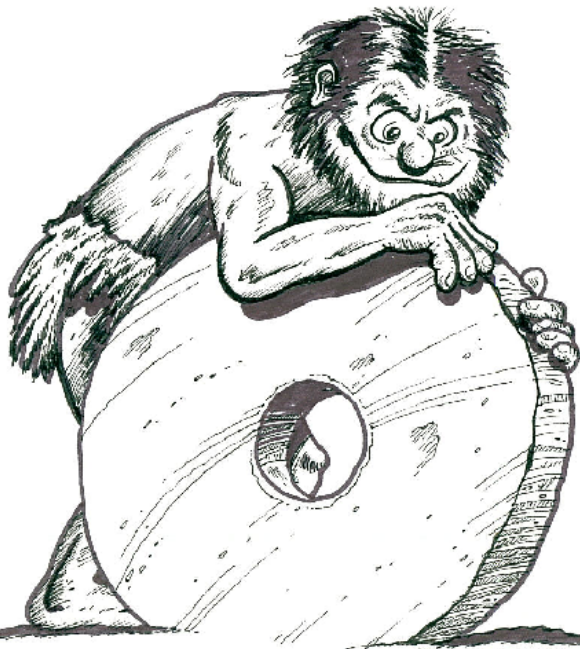
Only automated testing matters!

Empowers engineers to change anything and scale the team

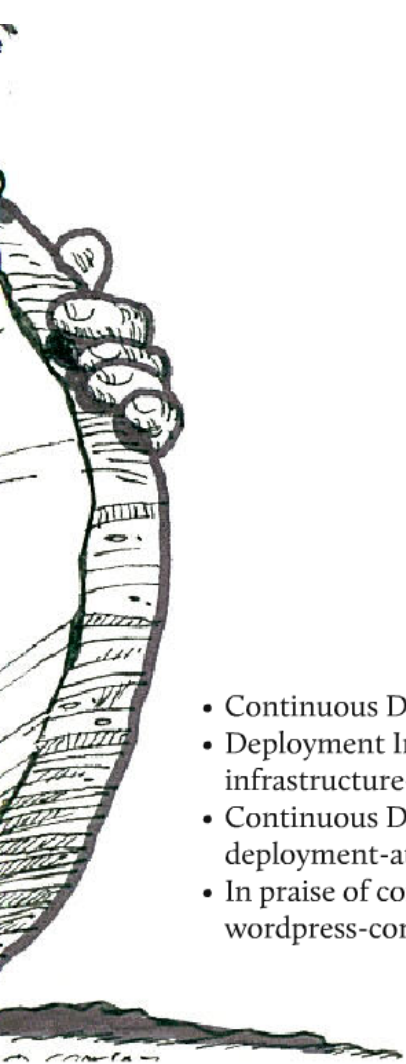
Facilitates continuous re-testing, allows the code to get better with age

Attracts the right kind of engineers

Culture



- Continuous Deployment at Wealthfront eng blog: <http://eng.wealthfront.com/search/label/continuous%20deployment>
- Deployment Infrastructure for Continuous Deployment (David Fortunato) <http://eng.wealthfront.com/2010/05/deployment-infrastructure-for.html>
- Continuous Deployment at IMVU: Doing the impossible fifty times a day <http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/>
- In praise of continuous deployment: The WordPress.com story <http://toni.org/2010/05/19/in-praise-of-continuous-deployment-the-wordpress-com-story/>



- Continuous Deployment at Wealthfront eng blog:<http://eng.wealthfront.com/search/label/continuous%20deployment>
- Deployment Infrastructure for Continuous Deployment (David Fortunato) <http://eng.wealthfront.com/2010/05/deployment-infrastructure-for.html>
- Continuous Deployment at IMVU: Doing the impossible fifty times a day <http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/>
- In praise of continuous deployment: The WordPress.com story <http://toni.org/2010/05/19/in-praise-of-continuous-deployment-the-wordpress-com-story/>