

# Controlling Your Environments using Infrastructure As Code

ConFoo Montréal - February 2016

Steve Mercier

# Who am I?



I participated in multiple software development projects:

- from very small (<10 person) to large (~400 persons)
- from a long time ago (>20 years) up to now
- from different angles/roles: Developer, Architect, PM, DevOps/BuildMaster, Software Release Manager
- I have seen software methods/processes come and go: Waterfall, RUP, OpenUP, XP, Scrum, Scaled Agile, etc.

## **But the essential remains:**

I believe that to produce good software, it takes good people + resources AND discipline/professionalism/focus!



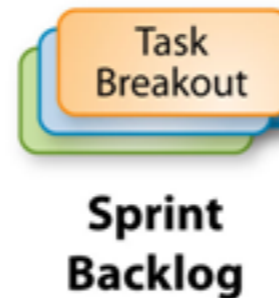
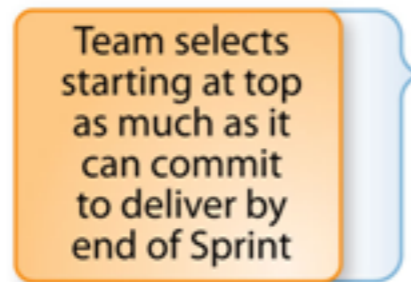
# Are You Agile?

Agile is not easy... and is not only about ceremonies + tools!

Inputs from Executives,  
Team, Stakeholders,  
Customers, Users



Every  
24 Hours



Sprint end date and team deliverable do not change



# Scrum Overview

Nothing on the previous slide concerning software context, where is the code delivered, running, performing...

Agile typically deals with this with the Definition Of Done concept, sadly overlooked way too often...

What would be a good  
measure of Agility  
anyway?

# Suggestion #1:

The capability to release  
(aka)

The frequency at which you are  
delivering software updates  
that add business value to your clients  
without breaking any previously  
delivered business value that you want  
to retain

# Suggestion #2:

The time to release  
(aka)

The time it takes you to deliver  
the smallest change/fix to your  
software in production



Given that:  
What prevents true  
agility?

What prevents you to **release**?

What prevents you to release **fast**?

# From my point of view: Lack of environments' management

True for all types of environments:  
DEV, QA, Staging, Production

# Platform dependencies? Do you manage them?

ex: Frameworks dependencies, external libraries dependencies, etc. Are they either never updated or are they continuously creating problem when you upgrade them?

# OS dependencies? Do you manage them?

How is the OS changed on your platform? Does it break your applications sometimes? Any logs/traces of those changes? Can they be rolled back in case of problems?

# Hardware/VM specs dependencies? Do you manage them?

What happens if available RAM/Disk/Network gets below what your applications need? Do you know what they need?

If you manage your environments, do you manage them manually?

Everything not automated  
reduces your Agility





Manual triggers can take a long time

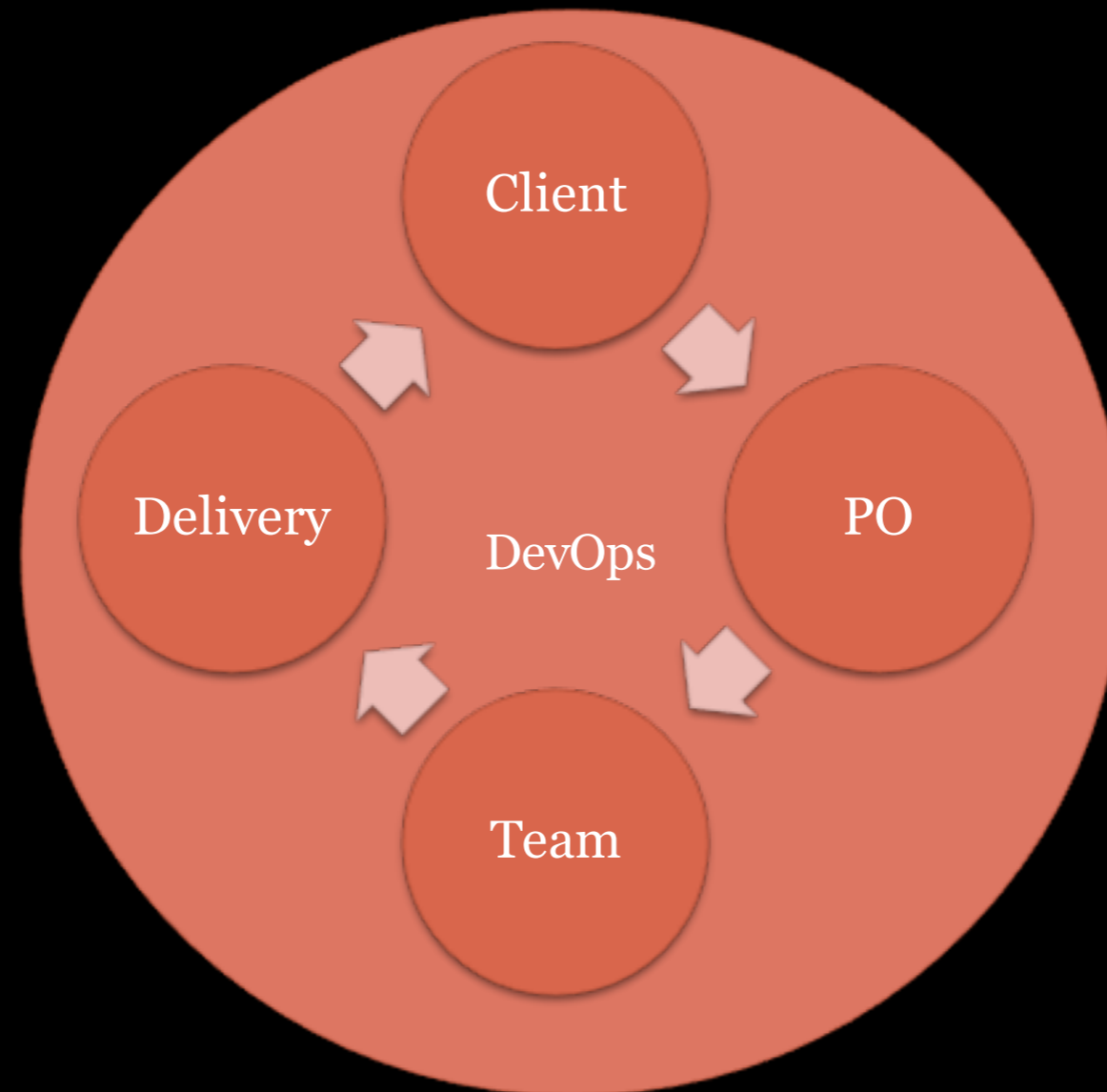
Manual = Time++;



As your manager might say

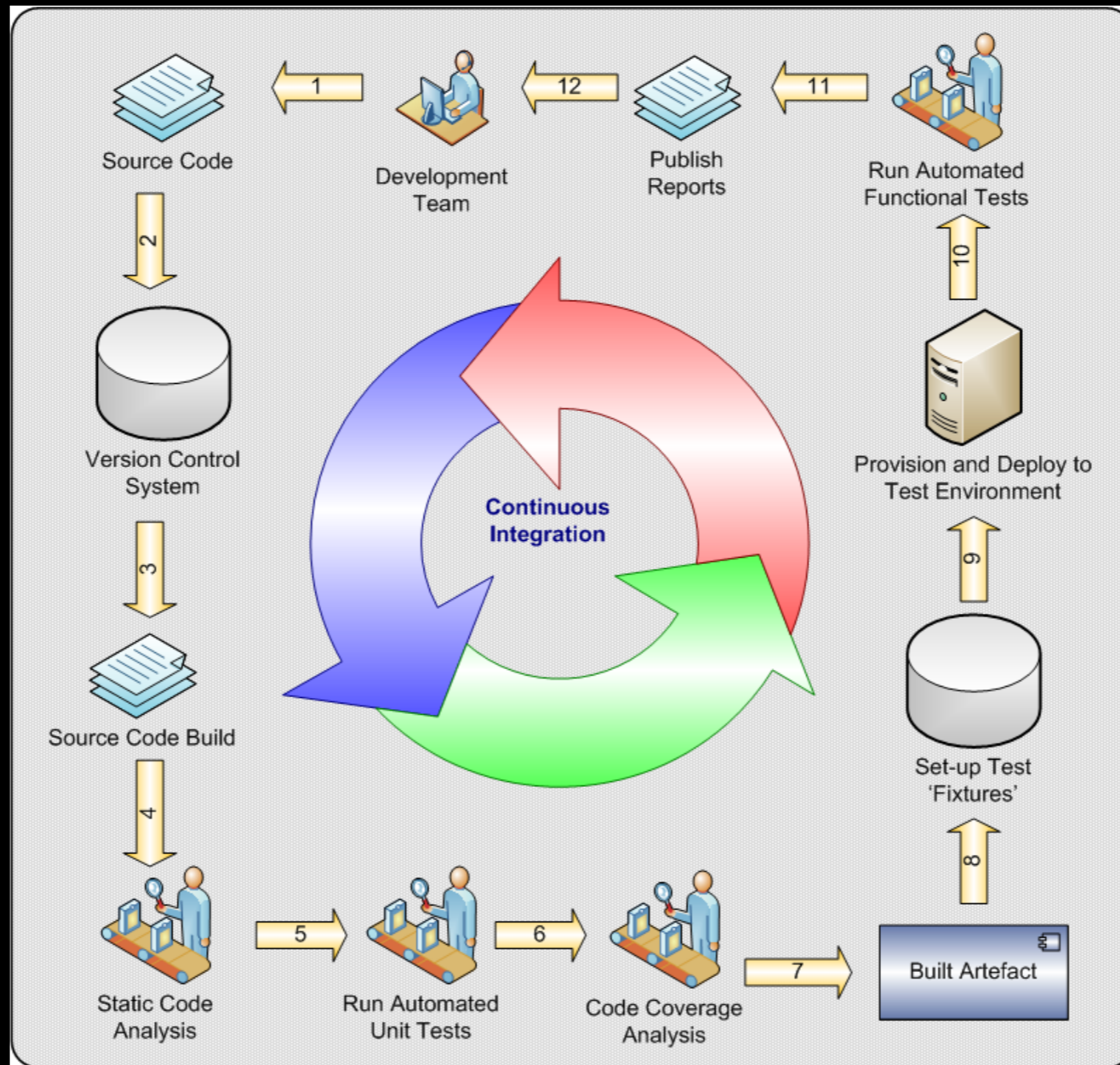
Time == Money();

# What techniques can reduce manual steps?



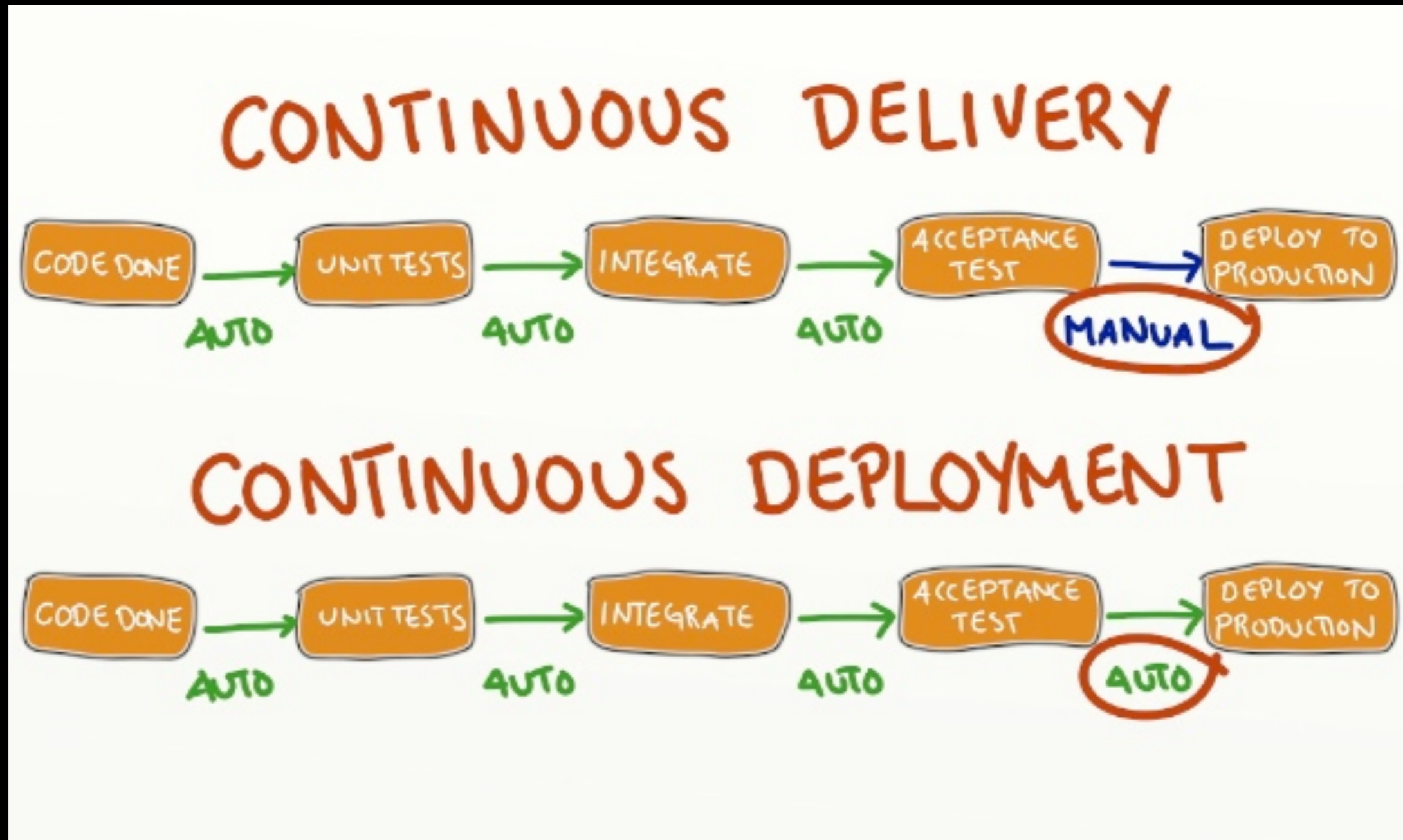
You could consider using DevOps:  
CI server, CD server, Infrastructure as Code to automate  
as much as possible your release process

# What techniques can reduce manual steps?



Continuous Integration

What techniques can reduce manual steps?



Continuous Delivery/Deployment

# What techniques can reduce manual steps?

## Puppet

```
node 'codecamp.ro' {  
  package { 'ruby'  
    ensure => 'latest'  
  }  
}
```

Are your Agile Demos  
done from unreleased  
software

(aka Works at my desk)



# SHIPPING IS A FEATURE

Your software must have it.

Your software must have it!



Demos from unreleased software /  
Difficulty to release in production



Involve Ops people in sprints  
Releasing in prod should be doable by  
anyone, anytime, using a single click  
(rollback is obviously a feature you will want!)

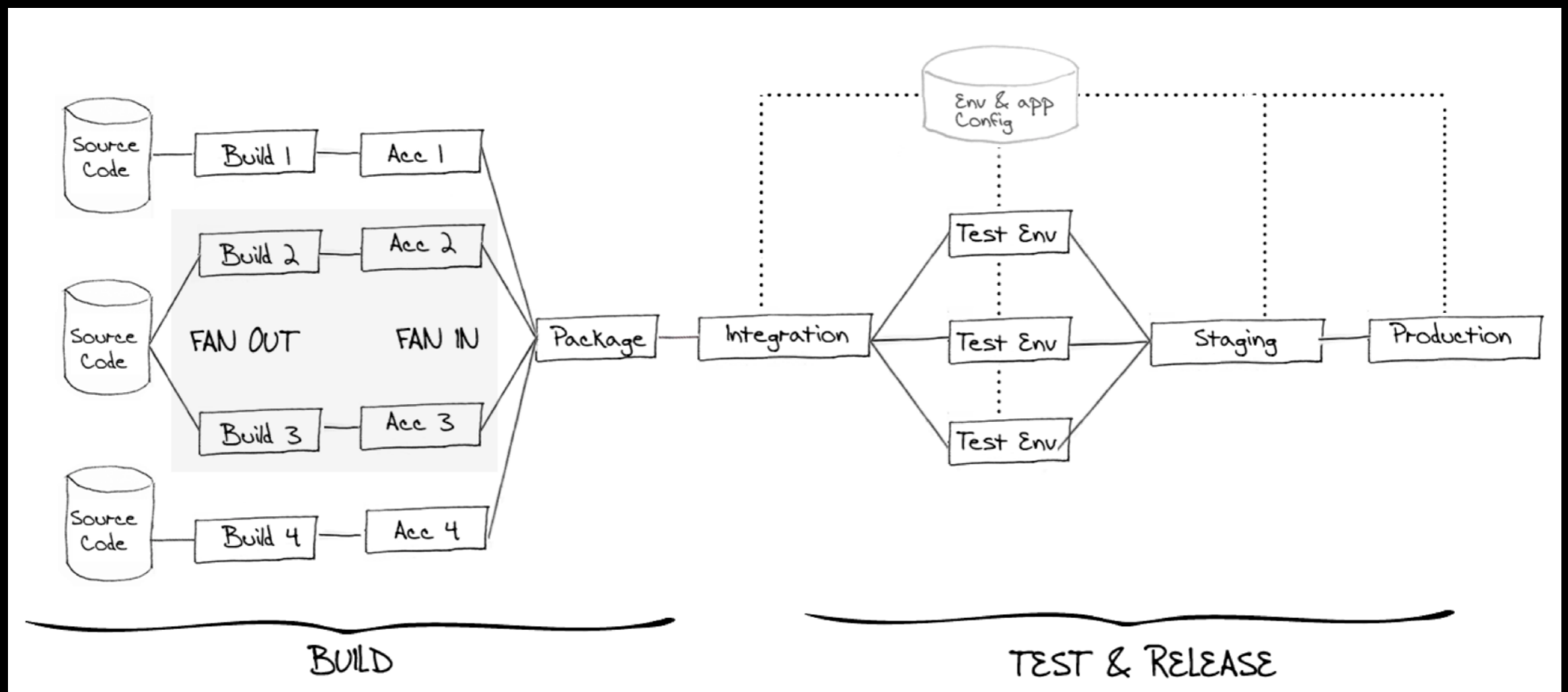
CD—Continuous  
Delivery/Deployment

# Continuous Delivery/Deployment

Always have a shippable version available for your customers

Ex: GO CD (from Thoughtworks - now Open Source)

You can integrate your CI servers (Jenkins) with a CD server



# Continuous Delivery/Deployment

A good practice to deploy gradually using cascaded environments:

- (Development)
- Test
  - Less resources, used mostly to validate business logic
- Staging/Pre-Production
  - More representative of the production environment. Can be used for load/performance testing. Typically uses a data set that is a copy of the Production data set.
- Production

# IaC—Infrastructure as Code

# Infrastructure as Code

Your code is under CM, but your infrastructure is typically not! It also needs to be versioned, tracked and automated!



# Infrastructure as Code

There are so many tools available. But essentially, keep **ALL** under source control, including what it takes to reproduce your production environment from **ZERO**

- How to restart a hardware environment/virtual machines
- How to install the platform on the machines
- How to install the applications on the platforms
- How to configure the whole stack
- The databases schemas and content
- **Everything!**

# Infrastructure as Code

If you are not convinced, think about your disaster recovery plan (you have one, right?)

What if your server room is destroyed by water/fire? (or the one from your cheap cloud provider...)

With IaC at least, the software part is covered in case of disaster



DevOps

# DevOps

- A Good app without an infrastructure to run it is useless...
- A superb, scalable infrastructure without an app adding business value to a customer is also useless...
- You need both!
- DevOps should not be a separate team! It serves to bridge the gap between development and operations teams.
- And if you are not convinced that DevOps cannot typically be a single person's responsibility...

# DevOps related tools

PERIODIC TABLE OF DEVOPS TOOLS (V1)																XebiaLabs Deliver Faster					
1 En O 12c															2 Fm Aws Amazon Web Services						
3 Os My MySQL	4 Os Gt Git															5 En Ch Chef	6 En Pu Puppet	7 Os An Ansible	8 En Sl Salt	9 Os Dk Docker	10 Pd Az Azure
11 En Mq MSSQL	12 Os Sv Subversion															13 Fr Ssh SSH	14 En Bl BladeLogic	15 Os Va Vagrant	16 Fr Tf Terraform	17 Os Rk rkt	18 Fm Hk Heroku
19 Os Pq PostgreSQL	20 Fr Mc Mercurial	21 Os Mv Maven	22 Os Gr Gradle	23 En Mr Meister	24 Os Jn Jenkins	25 Pd Bb Bamboo	26 Os Tr Travis CI	27 Fr Ar Archiva	28 Os Fn FitNesse	29 Fr Se Selenium	30 Os Gn Gatling	31 Pd Gd Deployment Manager	32 Os Sf SmartFrog	33 Fr Cb Cobbler	34 Os Bc Bcfg2	35 Os Kb Kubernetes	36 En Rs Rackspace				
37 Os Mg MongoDB	38 Fm Gh Github	39 Os Br Buildr	40 Os At ANT	41 Fm Bm BuildMaster	42 Fm Cs Codeship	43 Fm Sn Snap CI	44 Fm Cr CircleCI	45 Os Nx Nexus	46 Fr Cu Cucumber	47 Os Cj Cucumber.js	48 Fr Qu Qunit	49 Fr Cp Capistrano	50 Fr Ju JuJu	51 Os Rd Rundeck	52 Os Cf CFEngine	53 Fr Pk Packer	54 Fm Bx Bluemix				
55 En Db DB2	56 Fm Bb Bitbucket	57 Fm Qb QuickBuild	58 En Ub UrbanCode Build	59 Pd Ta Visual Build	60 Fm Tc TeamCity	61 Fm Sh Shippable	62 Os Cc CruiseControl	63 Os Ay Artifactory	64 Fr Ju JUnit	65 Fr Jm JMeter	66 Fr Tn TestNG	67 En Rd RapidDeploy	68 Fm Cy CodeDeploy	69 En Oc Octopus Deploy	70 Os No CA Nolio	71 En Eb ElasticBox	72 En Ad Apprenda				
73 Fr Cs Cassandra	74 En Hx Helix	75 Os Msb MSBuild	76 Os Rk Rake	77 Os Lb LuntBuild	78 Os Cu Continuum	79 Fm Ca Continua CI	80 Os Gu Gump	81 Os Ng NuGet	82 Os Ap Appium	83 En Xltv XL TestView	84 En Tc TestComplete	85 Os Go Go	86 En Ef ElectricFlow	87 En Xld XL Deploy	88 En Ud UrbanCode Deploy	89 Os Mo Mesos	90 Os Cf Cloud Foundry				

Share

Embed

Become Excellent!

Subscribe here!

91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Ls CA Service Virtualization	94 En Bm BMC Release Process	95 En Hp HP Codar	96 Pd Ex Excel	97 En Pl Plutora Release	98 En Sr Serena Release	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow
106 En Sp Splunk	107 Os Ki Kibana	108 Fm Nr New Relic	109 Os Ni Nagios	110 Os Gg Ganglia	111 Os Ct Cacti	112 Os Gr Graphite	113 Os Ic Icinga	114 Fm Sl Sumo Logic	115 Os Ls Logstash	116 Fm Lg Loggly	117 Os Gr Graylog	118 Os Sn Snort	119 Os Tr Tripwire	120 En Cy CyberArk

What could be potential  
solutions to deliver and  
faster?

1- Testing (TDD, BDD)

2- CI

3- CD

4- IaC

DEMO

# Demo content

- GO CD presentation
  - Application build pipeline
    - connected to GitHub for app code
    - running unit tests
  - Triggering Staging pipeline on success
    - using Vagrant + VirtualBox + Ansible to provision production like environment for system tests
    - connected to GitHub for IaC code
  - Triggering Deployment pipeline on success
    - using Ansible (Tower) to provision non VM production multiple environments



# Application Build/Unit tests

Continuous Integration pipeline  
Triggered on code changes



# Staging pipeline

For deployment + system testing

Triggered on new application integration OR new IaC code

# Deployment pipelines

Used to actually deploy the application into production  
triggered on successful staging pipelines OR manually when needed

Discussion  
How do you do it?

# Agile Values vs. Agile Practices

# CRAFTSMANSHIP MANIFESTO

<http://manifesto.softwarecraftsmanship.org/>

**NOT ONLY WORKING SOFTWARE,  
BUT ALSO WELL-CRAFTED SOFTWARE**  
**NOT ONLY RESPONDING TO CHANGE,  
BUT ALSO STEADILY ADDING VALUE**  
**NOT ONLY INDIVIDUALS AND INTERACTIONS,  
BUT ALSO A COMMUNITY OF PROFESSIONALS**  
**NOT ONLY CUSTOMER COLLABORATION,  
BUT ALSO PRODUCTIVE PARTNERSHIPS**

Agility is more than Agile values and ceremonies

It is acting as per the Agile values and producing software with related best practices



## BOY SCOUT RULE

Leave your code better than you found it.

# But where to start?

Every Agile cycle, try to improve on those issues, trying to automate everything you can, while having the conversation with your key stakeholders

Questions or  
comments?

Thanks!

**[softwarethatmattersdoneright.com](http://softwarethatmattersdoneright.com)**

+

**<http://ca.linkedin.com/in/stevemercier>**



Backup

# Install provisioning tools

Ex: Vagrant +  
VirtualBox

- vagrant init hashicorp/precise64
  - vagrant up
  - vagrant ssh
  - vagrant destroy

# Ex: See how using the Vagrant file, we can provision the platform

```
config.vm.box = "hashicorp/precise64"  
config.vm.network :forwarded_port, guest: 80, host: 4567  
config.vm.provision :shell, path: "bootstrap.sh"
```

```
#!/usr/bin/env bash
```

```
apt-get update  
apt-get install -y apache2  
if ! [ -L /var/www ]; then  
    rm -rf /var/www  
    ln -fs /vagrant /var/www  
fi  
apt-get install -y python3
```

# How to install the applications on the platforms

Ex: We can simply reuse the GIT clone using a shared folder between host and guest VM or use similar platform provisioning technique to perform a git clone in the VM after provisioning