Continuously deliver your puppet code with jenkins, r10k and git

Toni Schmidbauer

October 16, 2014

whoami

- SysAdmin@s-itsolutions.at
- toni@stderr.at
- stderr@jabber.org
- `http://stderr.at`
- `http://github.com/tosmi/puppetcamp2014`

Agenda

- A short story about configuration management
- What is continuous delivery
- Tools used to achieve continuous delivery
- DEMO
- Things to improve

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11 SPARC/i386, AIX, RHEL/CentOS 5/6/7)

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11 SPARC/i386, AIX, RHEL/CentOS 5/6/7)
- We've got around 1000 nodes in total

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11 SPARC/i386, AIX, RHEL/CentOS 5/6/7)
- We've got around 1000 nodes in total
- Before CM we had **strict** standards on how to manage these systems

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11 SPARC/i386, AIX, RHEL/CentOS 5/6/7)
- We've got around 1000 nodes in total
- Before CM we had **strict** standards on how to manage these systems
- The problem:
  count(team members) == count(standards)

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11 SPARC/i386, AIX, RHEL/CentOS 5/6/7)
- We've got around 1000 nodes in total
- Before CM we had **strict** standards on how to manage these systems
- The problem:
  count(team members) == count(standards)
- So configuration management is the solution to all our problems

The solution to all our problems

The solution to all our problems

- Broke our systems

WHY????

Problems with our old CM system

- Deployments sucked

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked
  - No Unittests
  - No acceptance tests

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked
  - No Unittests
  - No acceptance tests
- No immediate feedback if things where OK or **not**

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked
  - No Unittests
  - No acceptance tests
- No immediate feedback if things where OK or **not**
- Systems installed without CM are hard to bring under CM control

Problems with our old CM system

- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked
  - No Unittests
  - No acceptance tests
- No immediate feedback if things where OK or **not**
- Systems installed without CM are hard to bring under CM control
- Every system was a special case

So what's our solution?

Continuous delivery

- is a pattern for getting software from development to release

1

[1] Continuous Delivery: Jez Humble, David Farley

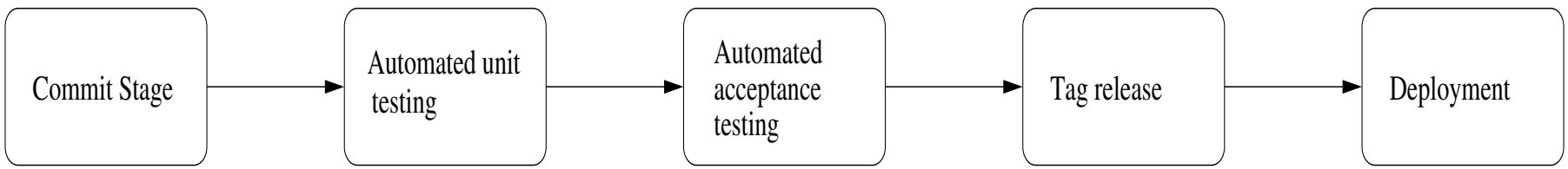
Continuous delivery

- is a pattern for getting software from development to release
- this pattern is called **the deployment pipeline**

1

[1] Continuous Delivery: Jez Humble, David Farley

The deployment pipeline



Why are we using continuous delivery

- When you automate your deployment,

Why are we using continuous delivery

- When you automate your deployment,
  - less mistakes will happen and the same mistake will not happen twice

Why are we using continuous delivery

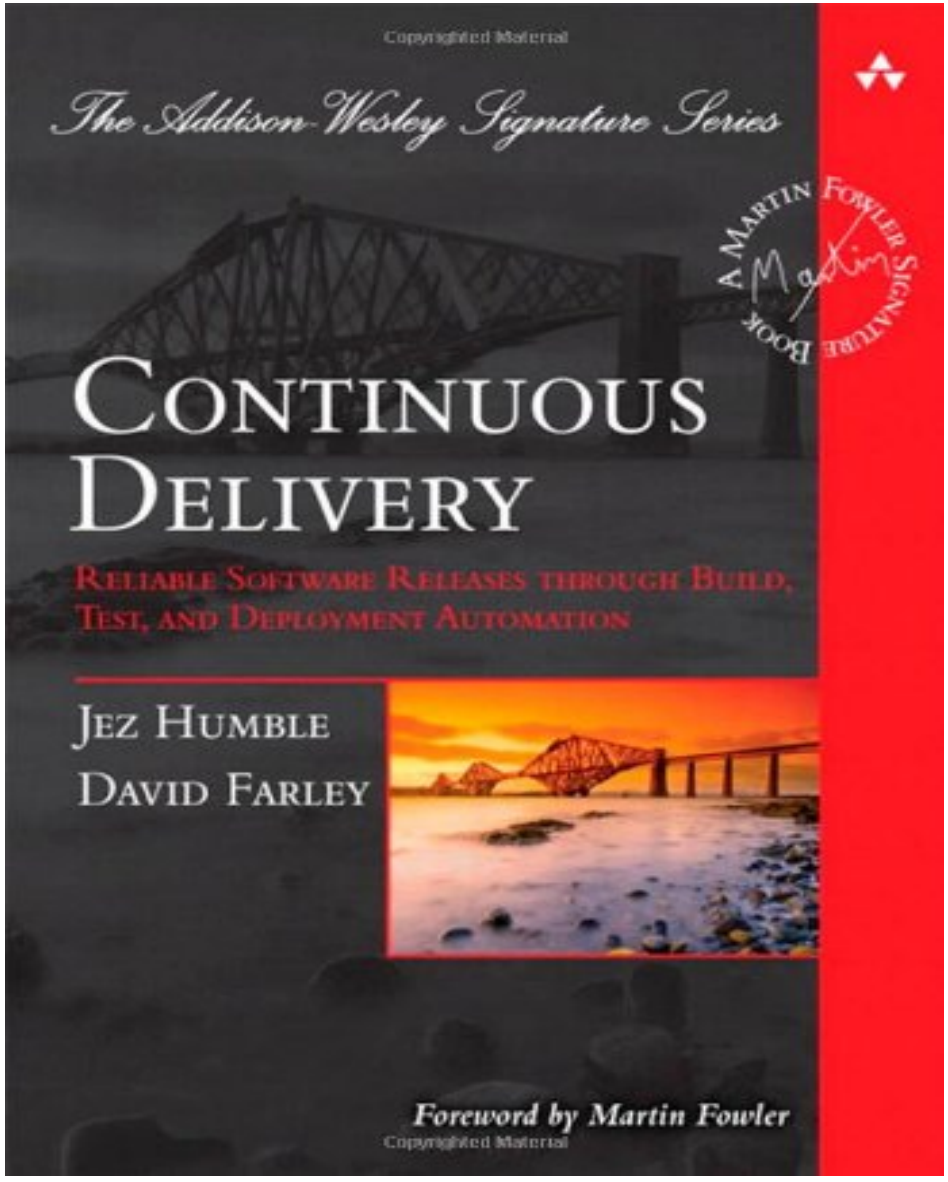
- When you automate your deployment,
  - less mistakes will happen and the same mistake will not happen twice
  - less mistakes means less stress when deploying

Why are we using continuous delivery

- When you automate your deployment,
  - less mistakes will happen and the same mistake will not happen twice
  - less mistakes means less stress when deploying
  - less stress means you are going to deploy more often

Why are we using continuous delivery

- When you automate your deployment,
  - less mistakes will happen and the same mistake will not happen twice
  - less mistakes means less stress when deploying
  - less stress means you are going to deploy more often
  - more deployments means a more flexible environment


Copyrighted Material
The Addison-Wesley Signature Series
A Martin Fowler Signature Book
CONTINUOUS DELIVERY
RELIABLE SOFTWARE RELEASES THROUGH BUILD, TEST, AND DEPLOYMENT AUTOMATION
JEZ HUMBLE
DAVID FARLEY
Foreword by Martin Fowler
Copyrighted Material

Tools to build a deployment pipeline

Jenkins

- Jenkins is an Open Source continuous integration server
- It's purpose is to execute and monitor jobs
- Jobs are shell scripts or any other thing that's executable and returns 0 on success
- Many plugins available to extend Jenkins (e.g. git, build-pipeline, monitor)
- You can **link jobs** together, thats our **pipeline**

Jenkins II


Build Pipeline
Run
History
Configure
Add Step
Delete
Manage
Pipeline
#38
#38 production_puppet_unittest
Oct 3, 2014 9:45:27 AM
4 min 29 sec
#3 production_puppet_acceptance
Oct 3, 2014 9:50:03 AM
2 min 26 sec
#38 production_puppet_deployment
Oct 3, 2014 9:52:38 AM
1 min 58 sec
Pipeline
#37
#37 production_puppet_unittest
Oct 2, 2014 8:40:05 PM
4 min 23 sec
#2 production_puppet_acceptance
Oct 2, 2014 8:44:35 PM
2 min 33 sec
#37 production_puppet_deployment
Oct 2, 2014 8:47:15 PM
3 min 54 sec
Pipeline
#36
#36 production_puppet_unittest
Oct 2, 2014 7:12:44 PM
4 min 24 sec
admin
#1 production_puppet_acceptance
Oct 2, 2014 7:17:15 PM
1 min 35 sec
production_puppet_deployment

Monitoring with Jenkins

GIT

- **git push** triggers the deployment pipeline

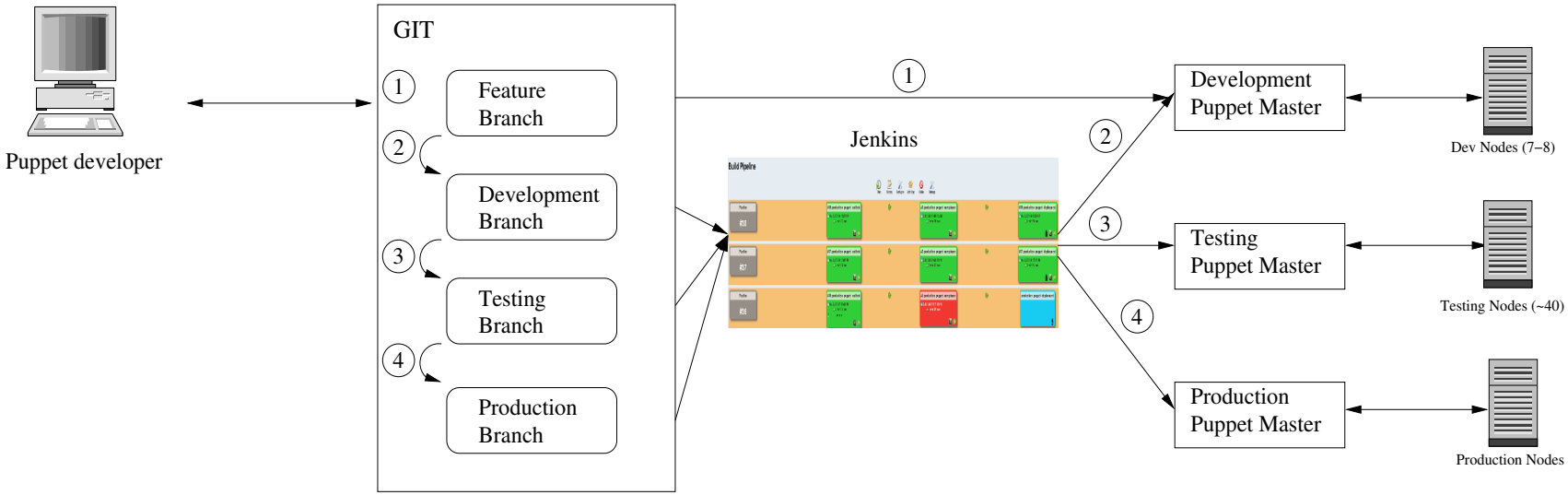
GIT

- **git push** triggers the deployment pipeline
- one central repository for internal modules
- gitolite for access control
- 3 main branches
  - development
  - testing
  - production
- feature branches for new site local modules
- hiera data is in the same repository

GIT repository layout

- `modules/`:
  - where r10k stores external (forge, github) modules
- `site/`:
  - site local modules, we do not want to share
- `hiera/`:
  - our hiera yaml files
- `Puppetfile`:
  - config file for r10k that specifies which external modules we need
- `Vagrantfile`:
  - To boot a development puppet environment on your local workstation

GIT workflow



1. Features Branches get automatically created on Puppet Master (Dynamic Environments)
2. Development Branch gets deployed on commit via Jenkins
3. Testing Branch gets deployed via GIT tag pushing to testing triggers a deployment
4. Production Branch gets deployed via GIT tag pushing to production triggers a deployment

It's all the same for Hiera yaml files!

r10k

- a tool to deploy puppet environments and modules
- every git branch gets deployed to a corresponding puppet environment
- it also downloads and installs modules from puppetforge or github
- in the current version (1.3.2) dependencies have to be managed manually

Example Puppetfile

```
forge 'forge.puppetlabs.com'

mod 'puppetlabs/ntp', '3.1.2'
mod 'puppetlabs/postgresql', '3.4.2'
mod 'puppetlabs/stdlib', '4.3.2'
mod 'puppetlabs/firewall', '1.1.3'
mod 'puppetlabs/apache', '1.1.1'
mod 'puppetlabs/lvm', '0.3.2'
mod 'nosolutions/tsm', '0.2.2'
mod 'saz/sudo', '3.0.6'
mod 'spiette/selinux', '0.5.4'

mod 'concat',
    :git => 'https://github.com/puppetlabs/puppetlabs-concat',
    :commit => 'feba3096c99502219043b8161bde299ba65e7b8a'
```

You are able to pin to a git tag / branch / commit hash

a word on testing

- you must have unit tests for your puppet code: **rspec-puppet**
- for acceptance tests there's **puppetlabs/beaker**
- you need to test everything to get most out of the build pipeline
- we test
  - internal puppet modules
  - hiera data
  - puppet configuration
- all internal modules are required to have rspec tests

rspec-puppet example

samplemodule/manifests/init.pp

```
class samplemodule ( $message = 'defaultmessage' ) {
  notify { 'samplemessage':
    message => "This is the sample module, my message is: $message",
  }
}
```

samplemodule/spec/classes/samplemodules_spec.rb

```
require 'spec_helper'

describe 'samplemodule', :type => :class do
  context 'with default parameters' do
    it { should contain_notify('samplemessage') }
  end
end
```

beaker example

```
1  require 'spec_helper_acceptance'

3  describe 'profiles::ossbase class' do
     it 'should work with no errors' do
5      apply_manifest('include profiles::ossbase', :catch_failures => true)
     end
7  end
```

DEMO

Do try this at home

- You need:
  - Vagrant from `http://vagrantup.com`
  - Virtualbox
  - Git client
- You have to run:
  - `git clone https://github.com/tosmi/puppetcamp2014.git`
  - `cd puppetcamp2014`
  - `vagrant up`
  - `vagrant ssh`

Things we have to improve

- We need more test Systems (CentOS/RHEL/Solaris/AIX?)
- We need more acceptance tests
- Once again use stages in production
- Our documentation

Things puppetlabs should improve

- Puppetlabs should package beaker as a rpm/deb whatever, gems suck in production

Summary

- Continuous Delivery is implemented via a deployment pipeline

Summary

- Continuous Delivery is implemented via a deployment pipeline
- Within the pipeline **everything** is automated

Summary

- Continuous Delivery is implemented via a deployment pipeline
- Within the pipeline **everything** is automated
- When everything is automated you need to test everything

Summary

- Continuous Delivery is implemented via a deployment pipeline
- Within the pipeline **everything** is automated
- When everything is automated you need to test everything
- When everything is automated and well tested deploying becomes easy

Summary

- Continuous Delivery is implemented via a deployment pipeline
- Within the pipeline **everything** is automated
- When everything is automated you need to test everything
- When everything is automated and well tested deploying becomes easy
- Tools we are using to implement a deployment pipeline
  - Jenkins: to execute jobs
  - GIT: version control
  - Gitolite: access control and authorization for GIT
  - r10k: install puppet modules / create puppet environments from branches
  - rspec-puppet: unittests for puppet
  - puppetlabs/beaker: acceptance tests for puppet

Thanks for your attention!