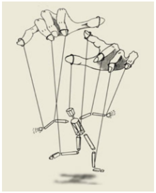


Continuously deliver your puppet code with jenkins, r10k and git

Toni Schmidbauer

October 3, 2014

whoami

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

Agenda

- A short story about configuration managment
- 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, 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, AIX, RHEL/CentOS 5/6/7)
- 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, AIX, RHEL/CentOS 5/6/7)
- Before CM we had **strict** standards on how to manage these systems
- The problem:
  count(teammembers) == count(standards)

A short story about configuration management (CM)

- We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11, AIX, RHEL/CentOS 5/6/7)
- Before CM we had **strict** standards on how to manage these systems
- The problem:
  count(teammembers) == 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 Unittest
  - 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 Unittest
  - No acceptance tests
- No immediate feedback if things where ok or **not**

Problems with our old CM system

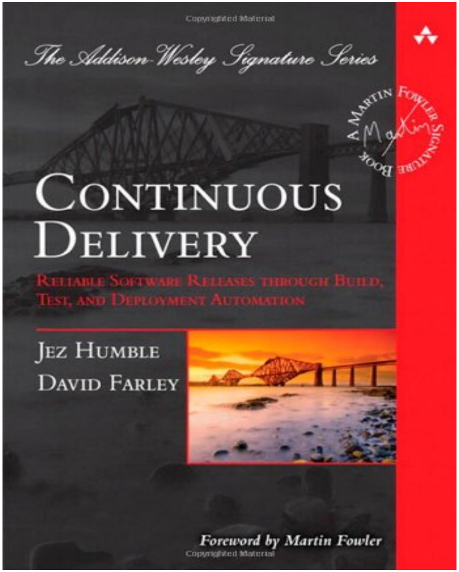
- Deployments sucked
  - Deployment via manual tagging and checkout, so mistakes happened
  - Deployment in stages, but we always had to cross our fingers
- Testing sucked
  - No Unittest
  - No acceptance tests
- No immediate feedback if things where ok or **not**
- Systems installed without CM are 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 Unittest
  - 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 whats our solution?

or: why should i care?



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

Continuous delivery

- is a pattern for getting software from development to release

1

[1] Continuous Delivery: Jez Humble, David Farley

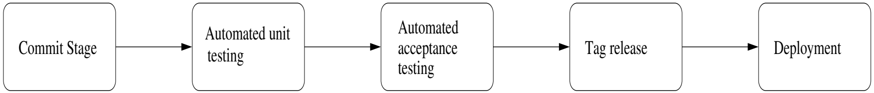
Continuous delivery

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

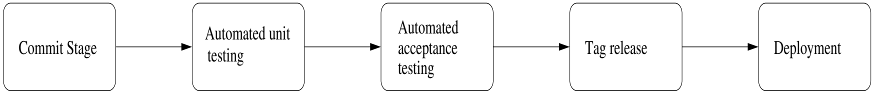
1

[1] Continuous Delivery: Jez Humble, David Farley

The deployment pipeline



The deployment pipeline



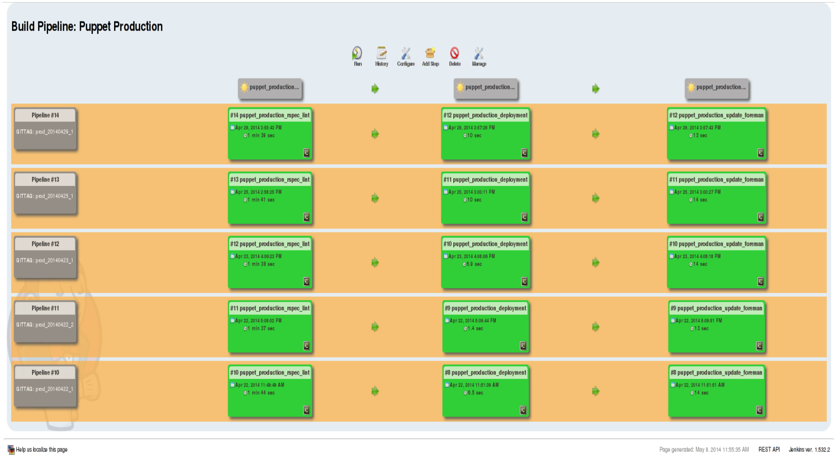
but the automated acceptance tests are currently missing in our setup, we will fix this with beaker (thanks puppetlabs)

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
- You can link jobs together, thats our pipeline
- Many plugins available to extend Jenkins (e.g. git, build-pipeline, Build Publisher Plugin, monitor)

Jenkins II



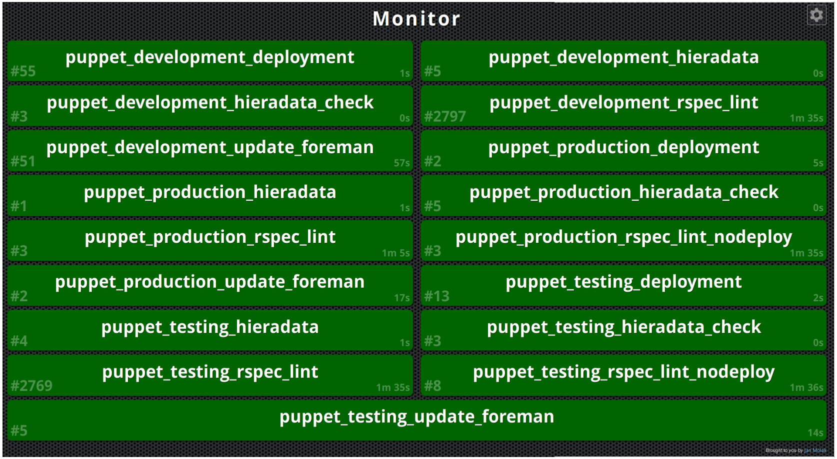
Build Pipeline: Puppet Production

	puppet_production...	puppet_production...	puppet_production...
Pipeline #14 GITTAG: prod_20140429_1	#14 puppet_production_rspec_lint Apr 29, 2014 3:55:43 PM 1 min 39 sec	#12 puppet_production_deployment Apr 29, 2014 3:57:28 PM 10 sec	#12 puppet_production_update_foreman Apr 29, 2014 3:57:43 PM 13 sec
Pipeline #13 GITTAG: prod_20140425_1	#13 puppet_production_rspec_lint Apr 25, 2014 2:58:25 PM 1 min 41 sec	#11 puppet_production_deployment Apr 25, 2014 3:00:11 PM 10 sec	#11 puppet_production_update_foreman Apr 25, 2014 3:00:27 PM 14 sec
Pipeline #12 GITTAG: prod_20140423_1	#12 puppet_production_rspec_lint Apr 23, 2014 4:06:23 PM 1 min 38 sec	#10 puppet_production_deployment Apr 23, 2014 4:08:06 PM 6.9 sec	#10 puppet_production_update_foreman Apr 23, 2014 4:08:18 PM 14 sec
Pipeline #11 GITTAG: prod_20140422_2	#11 puppet_production_rspec_lint Apr 22, 2014 5:08:02 PM 1 min 37 sec	#9 puppet_production_deployment Apr 22, 2014 5:09:44 PM 1.4 sec	#9 puppet_production_update_foreman Apr 22, 2014 5:09:51 PM 13 sec
Pipeline #10 GITTAG: prod_20140422_1	#10 puppet_production_rspec_lint Apr 22, 2014 11:49:49 AM 1 min 44 sec	#8 puppet_production_deployment Apr 22, 2014 11:51:39 AM 6.5 sec	#8 puppet_production_update_foreman Apr 22, 2014 11:51:51 AM 14 sec

Help us localize this page

Page generated: May 8, 2014 11:55:35 AM REST API Jenkins ver. 1.532.2

Monitoring with Jenkins



Monitor

puppet_development_deployment
#55 1s

puppet_development_hieradata
#5 0s

puppet_development_hieradata_check
#3 0s

puppet_development_rspec_lint
#2797 1m 35s

puppet_development_update_foreman
#51 57s

puppet_production_deployment
#2 5s

puppet_production_hieradata
#1 1s

puppet_production_hieradata_check
#5 0s

puppet_production_rspec_lint
#3 1m 5s

puppet_production_rspec_lint_nodeploy
#3 1m 35s

puppet_production_update_foreman
#2 17s

puppet_testing_deployment
#13 2s

puppet_testing_hieradata
#4 1s

puppet_testing_hieradata_check
#3 0s

puppet_testing_rspec_lint
#2769 1m 35s

puppet_testing_rspec_lint_nodeploy
#8 1m 36s

puppet_testing_update_foreman
#5 14s

Brought to you by Jan Molak

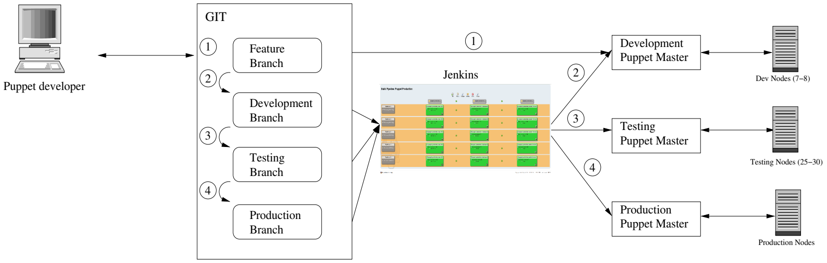
GIT

- One central repository managed with gitolite (access control for git) for internal modules
- 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, that we do not want to share
- hiera/: our hiera yaml files
- Puppetfile: config file for r10k that specifies which external modules we need

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 deploy to a puppet environment
- in the current version (1.3.2) dependencies have to be managed manually

a word on testing

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

rspec-puppet

- Ruby RSpec (unit tests) for puppet
- Every interal module must have rspec tests

```
require 'spec_helper'
describe 'linuxwochen2014' do
  let :facts { { :osfamily => 'RedHat' } }

  context 'ensure is set to absent' do
    let :params { { :ensure => 'absent'} }

    it do
      should contain_user('toni').with({
                                     'ensure' => 'absent',
                                     'uid'    => '4711',
                                     'gid'    => '100',
                                  })
    end

    it { should contain_package('emacs-nox').with_ensure('installed') }
    it { should contain_package('vim-enhanced').with_ensure('absent') }
    it { should contain_package('emacs-nox).that_comes_before('Package[vim-enhanced]') }
  end
end
```

DEMO

Thanks for you attention!