

Continuously deliver your puppet code with jenkins, r10k and git

Toni Schmidbauer

September 21, 2014

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

Agenda

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

A short story about configuration management (CM)

- ▶ We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11, AIX, RHEL 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 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 5/6/7)
- ▶ Before CM we had **strict** standards on how to manage these systems
- ▶ The problem:
 $\text{count}(\text{teammembers}) == \text{count}(\text{standards})$

A short story about configuration management (CM)

- ▶ We manage a very diverse environment of UNIX/Linux Systems (Solaris 10/11, AIX, RHEL 5/6/7)
- ▶ Before CM we had **strict** standards on how to manage these systems
- ▶ The problem:
 $\text{count}(\text{teammembers}) == \text{count}(\text{standards})$
- ▶ 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

- ▶ Systems installed without CM are hard to bring under CM control
- ▶ Every system is a special case
- ▶ In the beginning every problem was a CM problem
- ▶ CFEngine 2 based, no unit tests
- ▶ Deployment in stages, but we always had to cross our fingers
- ▶ Deployment via manual tagging and checkout, so mistakes happened
- ▶ We fixed the same mistake more than once

So whats our solution?
or: why should i care?

Copyrighted Material

The Addison-Wesley Signature Series



A MARTIN FOWLER SIGNATURE
BOOK
Martin

CONTINUOUS DELIVERY

RELIABLE SOFTWARE RELEASES THROUGH BUILD,
TEST, AND DEPLOYMENT AUTOMATION

JEZ HUMBLE
DAVID FARLEY



Foreword by Martin Fowler

Copyrighted Material

Continuous delivery

- ▶ is a pattern for getting software from development to release

1

Continuous delivery

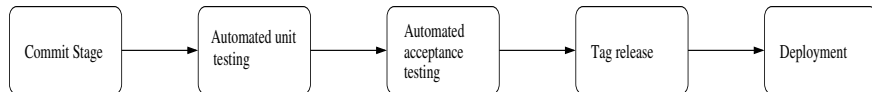
- ▶ is a pattern for getting software from development to release
- ▶ this pattern is called **the deployment pipeline**

1

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)

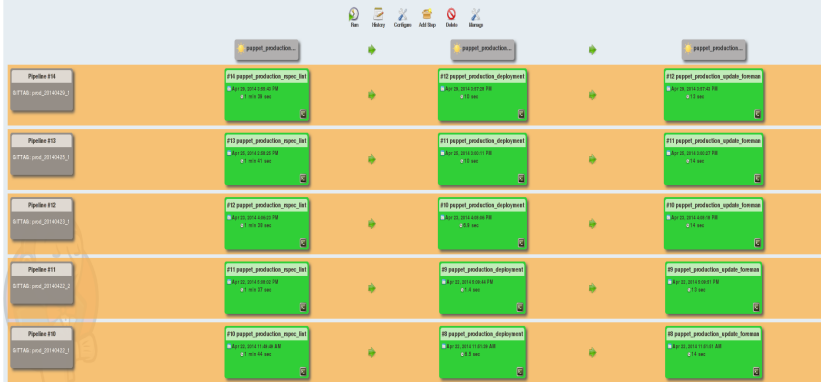
Tools to build a deployment pipeline

Jenkins

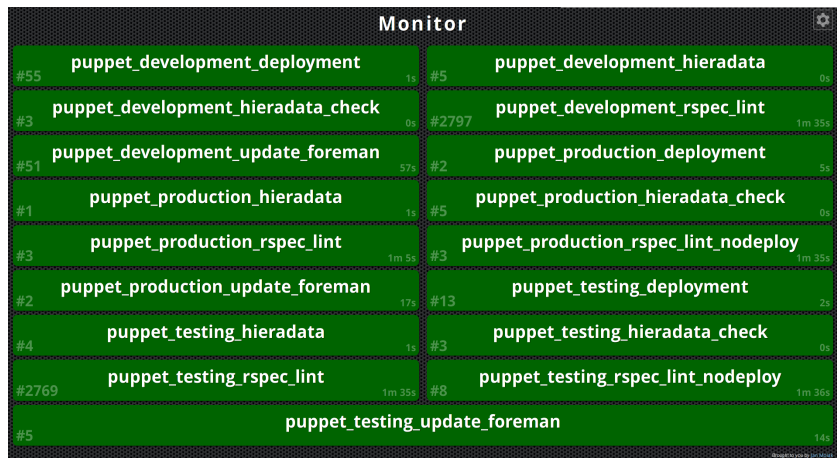
- ▶ Jenkins is an Open Source continuous integration server
- ▶ It's purpose is to execute and monitor jobs
- ▶ Jobs are shell scripts or any other thing that's executable and returns 0 on success
- ▶ You can link jobs together, thats our pipeline
- ▶ Many plugins available to extend Jenkins (e.g. git,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

Jenkins II

Build Pipeline: Puppet Production



Monitoring with Jenkins



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

rspec-puppet

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

```
1  require 'spec_helper'
2  describe 'linuxwochen2014' do
3    let :facts { { :osfamily => 'RedHat' } }
4
5    context 'ensure is set to absent' do
6      let :params { { :ensure => 'absent' } }
7
8      it do
9        should contain_user('toni').with({
10                                     'ensure' => 'absent',
11                                     'uid'     => '4711',
12                                     'gid'     => '100',
13                                     })
14      end
15
16      it { should contain_package('emacs-nox').with_ensure('installed') }
17      it { should contain_package('vim-enhanced').with_ensure('absent') }
18      it { should contain_package('emacs-nox').that_comes_before('Package[vim-enhanced]') }
19    end
20  end
```

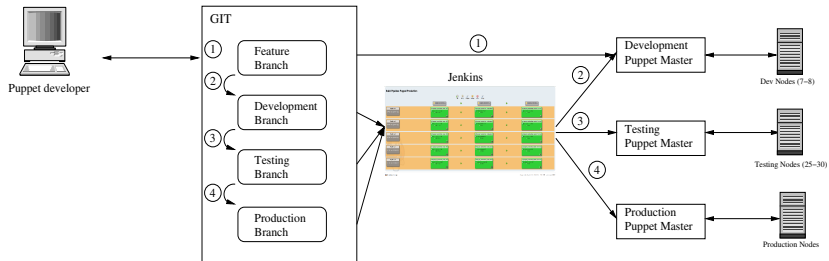
GIT

- ▶ One central repository managed with gitolite (access control for git)
- ▶ 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 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
a normal commit to the Testing branch only runs tests
- ④ Production Branch gets deployed via GIT tag
a normal commit to the Production branch only runs tests

It's all the same for Hiera yaml files!

- ▶ 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

DEMO

Thanks for you attention!