Continuous Integration
Continuous Integration

• What is Continuous Integration?
  • “In software engineering, continuous integration (CI) implements continuous processes of applying quality control — small pieces of effort, applied frequently.”

• What is Jenkins?
  • “Jenkins, previously known as Hudson, is an open source continuous integration tool written in Java.”
How It Used To Be

- Lack of automation
- Push me, pull you!
Getting Ready For Automated CI
Managing Change

- We want to keep revisions.
- But we also want to share.
- In fact, we want to share in complex ways.

- But when we share, shit breaks. :-(

Managing Change
Vincenzo Russo

• Development Team Lead
• He likes hats...
• He hates it when shit breaks.
• He also likes process.
• A little too much.
VCS

• Vincenzo likes Scrum
• That changes the way you think...

• What is “feature branching”?

• Um, we don't actually do that...

• We do feature forking!
VCS

- We use Gitorious instead of GitHub
Hardware In The Cloud

- So we are managing change...
- Next we need a place to build!
Miguel Jacq

- Systems Engineer
- Likes France.
- Hates working.
- Hates working so much, he makes tools that can work for him!
Managing The Servers

- Jenkins
- Madelon
- Libcloud / Linode API
- Puppet
- Nagios
- Munin
Why Jenkins for arbitrary tasks?

• Not just a CI tool
• Good for executing any remote script & monitoring result
• A useful audit trail
• Repeatable (useful for DR)
• Automatable
• Many notification options (iPhone, IRC, email)
Madelon?

- [http://github.com/mig5/madelon](http://github.com/mig5/madelon)
- Fabric-based implementation of Libcloud API
- Builds a new server, runs post 'hooks'
- Triggers the initial Puppetisation of a server
- Triggers automatic monitoring (Nagios/Munin)
Triggered from a Drupal webform!

New Server Deployment

Use this form to create a new server at Linode (using Madelon) and configure it (via Puppet) based on the appropriate roles.

The job will be passed through Jenkins, so that the progress may be viewed in the Console Output.

FQDN *

The FQDN (fully qualified domain name) of the new host. Eg. foo.example.com

Roles *

- apachesserver
- nginxserver
- dbserver
- proxyserver
- memcached_server
- mongodb_server
- monitor
- jenkins_server
- vpn_server

The roles for the new server.
Wrapping up

- Webform
  - Nginx
  - PHP-cgi
  - MySQL
  - Varnish

- Jenkins
- Madelon
- Linode API

- Puppet
  - Roles:
    - common
    - nginxserver
    - mysqlserver
    - varnishserver

- New Linode

- Nagios
  - Hostgroups:
    - common (RAM, disk, SSH, SMTP etc)
    - nginxservers
    - mysqlservers
    - varnishservers
Alasdair Cowie-Fraser

- Junior Developer
- Likes music.
- Not very clever.

- We need to stop him from breaking shit...

- Because that upsets Vincenzo!
We can manage change
We have a place to build
But we also have an Alasdair
Now we need a safe pair of hands!
• Jenkins is already doing large chunks of Mig's job – maybe he can do Alasdair's job too?
Webform once again, to make dev's life easier
Becomes a Jenkins job

Project name: Deploy_petsupermarket_master_branch

Description: deploy a new release of petsupermarket project from master branch

Source Code Management:
- Git
- Repositories: URL of repository: ssh://git@github.com/codeenigma/petsupermarket
- Branches to build: Branch Specifier (blank for default): master
Triggers a Fabric deployment

- Clones repo
- Checks out requested branch
- Creates database, settings.php, files dir
- Sets up symlinks (files dir lives outside build)
- Creates Drush alias, Drush crontab
- Creates Nginx vhost, restarts services
- Points 'Document Root' symlink to new build
- Commits relevant bits back into git (updated settings.php with db credentials etc)
First build in action

Console Output

Started by an SCM change
Checkout: /var/lib/jenkins/jobs/Deploy_petsupermarket_master_branch/workspace - hudson.remoting.LocalChannel@30b9c3
Using strategy: Default
Checkout: /var/lib/jenkins/jobs/Deploy_petsupermarket_master_branch/workspace - hudson.remoting.LocalChannel@30b9c3
Cloning the remote Git repository
Cloning repository
Patching upstream changes from ssh://git@github.com/codeenigma/petsupermarket
Commencing build of Revision 2d53efac9f689f8d52d9beee254e3197f538428 (origin/master)
Checking out Revision 2d53efac9f689f8d52d9beee254e3197f538428 (origin/master)
No change to record in branch origin/master

--- Cloning petsupermarket from github
[localhost] run: ssh-agent bash -c 'ssh-add; ssh -t -A dev1.codeenigma.com 'git clone git@github.com:codeenigma/petsupermarket~jenkins/petsupermarket_master_build_1"'
Cloning into /home/jenkins/petsupermarket_master_build_1...
[dev1.codeenigma.com] sudo: mv ~jenkins/petsupermarket_master_build_1 /var/www

Done.

--- This looks like the first build! We have some things to do...
--- Setting the live document root symlink
--- Making the shared files dir and setting symlink
--- Preparing the database
[dev1.codeenigma.com] /var/www/live.petsupermarket.master/db/petsupermarket.sql
--- Creating database dump /var/www/live.petsupermarket.master/db/petsupermarket.sql into petsupermarket
--- Creating database dump /var/www/live.petsupermarket.master/db/petsupermarket.sql into petsupermarket
--- Setting up an nginx vhost
[dev1.codeenigma.com] sudo: cp /etc/nginx/sites-available/dummy.conf /etc/nginx/sites-available/petsupermarket.conf
--- Adding htpasswd
[dev1.codeenigma.com] sudo: /usr/local/bin/nginx.default.conf master
--- Committing our settings.php change to preserve the db credentials
--- Committing our settings.php change to preserve the db credentials
[dev1.codeenigma.com] run: git commit -m "db url"
--- Committing our settings.php change to preserve the db credentials
[dev1.codeenigma.com] run: git push
What happens after the first build?

- Subsequent commits + push to that branch trigger new builds in Jenkins
- Fabric backs up site
- Clones a fresh build into /var/www
- Runs tests
- Runs drush updatedb
- Manipulates 'Document Root' symlink to point to new build
- Clears caches
ZERO-TOUCH DEPLOYMENT
Further Detail

Developer -> Github -> Jenkins

HTTP server (dev, stage or live) -> Real builds deployed with Jenkins

Real builds dir that each build links to

Shared assets

foobar_build_1

foobar_build_2

foobar_build_3

shared/files

The ghost's DocumentRoot which is a symlink pointing to the latest successful build -> live.foobar

Automatic Deployment tasks:
- Backup database
- Clone repo as foobar_build_n
- Switch to branch
- Move repo into /var/www
- Run tests
- Set site offline
- Run drush updatedb
- Set site online
- Switch 'live.foobar' DocumentRoot symlink to new build
- Restart services / cache
YOU BROKE THE BUILD!

This is Agnes and she's none too pleased with you, Bub. You broke the build. You should have known that you were making breaking changes, but you checked them in anyway.

If you start using continuous integration, Agnes won't have to come back.

www.YouBrokeTheBuild.com
Build failed? No problem!

- Usually happens early on in tests
- Existing site continues to function as it points to the last working build
- If it breaks too late in the process, simply set live symlink back to previous build
- Perhaps restore from the database backup taken prior to the broken build if it messed the data
Taking changes live?

- Merge or cherry-pick master branch to stage
- Triggers similar build on stage server
- Client approved? Cherry-pick/merge to prod
- Triggers similar build on prod server
- Takes database changes from dev > live because Jenkins build runs drush updatedb
How has this changed things?

- Automatic triggered build from git to live site sounds risky? Doesn't have to be
- Encourages and inspires devs to commit early, commit often
- Building good tools for Dev takes away the pain for both Dev and Op
- Jenkins notifications to IRC of successful builds + commit message keeps everyone informed!
- Emails and alerts about failed builds keeps everyone informed!
Fabric or Capistrano?

- Either tool is good
- Designed to solve same or similar problems
- (running remote tasks over SSH)
- Mig prefers Python to Ruby :P
- What Mig says goes (Mig wrote this slide..)
• Jenkins can build stuff.
• But Alasdair might still do something stupid. Who's keeping an eye on what he commits?
Steve Hunton

- Project Manager
- Likes order.
- Hates Scots.

- Which is unfortunate, since Alasdair is half-Scottish!
Peer Review

- Both Steve and Vincenzo like to keep an eye on poor Alasdair.
- They do so by:
  - Pull requests
  - Built feature branches
Peer Review

Reviewing merge request #1: I hacked core.

Please accept my hacking of core.

Requested by: mig5
Status: Open
Source repository: mig5:nokia_academy:master
Target repository: nokia_academy:master
Created at: November 10th, 2011 02:53

Commits that would be merged:

www/CHANGelog.txt (1/0)

1. DON'T HACK CORE!
3. 
4. 

Comments

Pushed new version 1
mig5 | a minute ago Edit comment

Add a new comment:
Comment (Use Markdown for formatting)

What the fuck were you thinking mig5? This is rubbish!

View the diff of what would change

Abuse the developer for their bad code below, as part of code review
Customers

- Lovely people
- Like it when things just work.
- Customers sign off on Stage
- We cherry-pick the changes to Production
- Tag and release (Jenkins again)
The Full Picture
The Full Picture
The Full Picture
The Full Picture

Jenkins
The Full Picture
The Full Picture
The Full Picture

Jenkins
The Full Picture

ETC. ETC.
The Full Picture
The Full Picture
The Full Picture
The Full Picture
The Full Picture

Jenkins
Steve Cowie

- Director of Operations
- Lacks empathy.
- Likes money. (He's a Scot.)
- Sends the bill!
Get Paid!
The Plug!

- Like what you see?
- Buy it from Code Enigma.
- Ask Mig to come and manage your servers – he'll find a way to avoid doing it!
Thanks for listening!