23 August 2012

Open up! Connecting systems and people

DRUPALCON

MUNICH 2012

20-24 AUGUST

Achieving Continuous Integration with Drupal

Achieving Continuous Integration with Drupal

Drupalcon Munich 2012

Barry Jaspan
barry.jaspan@acquia.com

The Evolution of a Drupal Developer

Stage 1:
Hacking code directly on the server.

PRO: It’s easy, fast, and efficient

- It’s the quickest and cheapest way to get started
- New features are visible immediately
- Bugs are easy to replicate
- Feedback is immediate
- No time-sucking “release engineering” overhead

CON: It’s a disaster (not) waiting to happen

- The slightest mistake causes a WSOD
- Encourages panic-driven engineering
- Impossible to demo without a release
- Nearly impossible to do with a team
- Basically, this never works beyond “install a new module”

Stage 2:
Develop locally, push to production

PRO: Safer, and works with teams

- Most obvious bugs never make it to production
- Normal software development process are possible
- Team development becomes possible
  - Version control to share and log changes
  - Everyone uses a local copy of the prod database

CON: Releases are slow and scary

- Release process

1. Announce a “feature freeze”
2. Merge everyone’s changes to release engineer’s machine
3. Test, find bugs, point fingers, fix bugs
4. Copy code to production servers
5. Manually upgrade the production environment
6. Find new bugs, scramble to fix or revert

Can we do better?

Stage 3:
Continuous Integration
Testing
Deployment

Fancy words for “good software engineering practice.”

Use a source code repository

- Step zero for good software engineering
- It mostly doesn’t matter which one you use
- Version everything
  - Code, tests, documentation, libraries, dependencies, …
- Keep it simple
  - Main branch and tags
  - Temporary, private feature branches
  - Release branches only for major changes

Integrate code constantly

- Break big features into small steps
- Automate the upgrade process
- Implement tests as you go
- Commit changes often
- Update from main branch daily
- Reduce integration conflicts, surface problems sooner

Sidebar:
Commits, Code Reviews, and Architecture

Automate testing

- “If it ain’t broke, test it anyway.”
  - Functionality: WSOD, unit, browser-based, ...
  - Performance
  - Upgrade
  - Start testing new things, don’t wait for time to “catch up”
- Run tests on every commit
  - No patch is done until all the tests pass
- Announce when the tests fail
- Deploy frequently to an accessible QA area
  - Stakeholders can always see current progress

Test in a clone of production

- "It worked on my machine!"
  - Drupal depends heavily on its environment
  - OS, Apache/MySQL/other daemons, PHP extensions, PEAR libraries, assorted packages
- "It worked on the testing database!"
  - Current (scrubbed) version of production database
- "Production has a different config..."
  - Manual server tweaks will always get lost
  - Unify and automate build of *all* environments
  - This is the part you probably do not (want to) do yourself

Automate deployment

- Deployment should be a push-button-and-relax operation
  - Tag each release for future reference or rollback
  - Snapshot databases for rollback
  - Deploy code to server(s)
  - Apply upgrade changes to database
    - No patch is done until the upgrade is automated

How do I get there?

"Continuous Integration is an attitude, not a tool." - James Shore

DIY

- Drush has commands for most operations
  - Code
    - `git tag && drush rsync @dev @test`
  - Files
    - `drush rsync @dev:%files @test:%files`
  - Database
    - `drush sql-sync @dev @test --sanitize`
    - `cat scrub.sql | drush @test sql-cli`
  - Testing
    - `drush @test test-run`
    - `drush @test ssh ./other-tests.sh`

But ... welcome to DevOps

- Git/SVN server
- Many scripts/Jenkins jobs
  - Deploy on commit
  - Copy DB/files for dev
  - Test on deploy
  - Tag and release on success
  - ... etc ...
- OS management
  - Server build
  - Security updates
- Multiple web vhosts
  - Domains, SSL, php.ini, ...
- Multiple databases
  - Manage credentials
- HA, memcached, Varnish, Tomcat, Jenkins, Solr, ...
- Scaling all of this
- Backups and restores
- 24/7 monitoring

Pay someone else

Acquia Cloud

PANTHEON

The Drupal Config/Content Staging Problem

- Configuration
  - The UI is *only* for prototyping!
  - Update functions
  - Features module
  - D8 Configuration Management Initiative
- Content
  - CI is about software development, not content management, but since someone *always* asks…
  - "Enterprise Drupal Application Scaling" by Michael Cooper, Drupal Watchdog, August 2012
  - Modules: Migrate, Node Import/Export, Deploy

23 August 2012

What did you think?

Please rate this session on the DrupalCon Munich website:

http://munich2012.drupal.org/node/409

Thank you!

Questions?

Barry Jaspan
barry.jaspan@acquia.com
Twitter: @bjaspan

Senior Architect
Acquia, Inc.

23 August 2012

What did you think?

Please rate this session on the DrupalCon Munich website:

http://munich2012.drupal.org/node/409

Thank you!