Practical continuous deployment

Who Am I?

- Steve Smith
- An Atlassian for 8+ years
- Original company sysadmin
- Developer for last 5 years
- Now working out of Amsterdam
- *Not* a professional speaker

Who Am I?

- Steve Smith
- An Atlassian for 8+ years
- Original company sysadmin
- Developer for last 5 years
- Now working out of Amsterdam
- ~~*Not* a professional speaker~~

Who are you?

- Who’s in the room? Devs, ops, mgmt?
- Please ask questions (or share your experiences), I’d like this to be a discussion, not a lecture.

What I've been up to...

- Spent 6 months converting our order systems to high-availability and continuous deployment.
- Why so long? Because the concept is straightforward, but it’s implications affect a lot of your organisation.

"Deployment"? "Delivery"?

- Continuous *integration* is continuous, automated build and test.
- Continuous *delivery* is the next obvious step; be continuously release-ready.
- Continuous *deployment* is the final step, the continuous delivery of software to production.

"Deployment"?
"Delivery"?

- Constant QA is the common theme.
- In practice there's a continuous spectrum of options, each organisation has different needs and constraints.
- But if you trust your testing and process you can adopt the level appropriate for you.

Why Continuous deployment?

- We want to release *features*, not “what ever happens to be done”
- Automation: Releasing is hard, automation makes it repeatable
- Remove organisational bottlenecks to releases

Stakeholder benefits

- To customers: You'll get your requested feature faster!
- To management: You'll get results faster and clearer progress.
- To devs: No more death-marches, mad-dashes, clean-up after releases.
- To admins: You know which change broke the system!

So how do you actually *do* it?

- Continuous deployment guides tend to focus on the high-level philosophy
- But how do you actually get a feature from a customer request to your servers?

Development workflow

- Continuous deployment implies a clearer development process.
- You need to know what is going out when you release, not a dump of the current state.
- Hence *release by feature*

Step 1: Track your requests

- Each feature/update request should have a unique ID.
- This allows tracking the state of a feature from request to deployment.
- Bug-trackers are a good choice for this.

Step 2: Work on this feature in a branch

- Create a branch for just this feature
- Name it after the feature request
  - Jira/Stash integration will do this
- The branch will be merged when complete
- You need a sane version control system
- We use git, Mercurial is good too

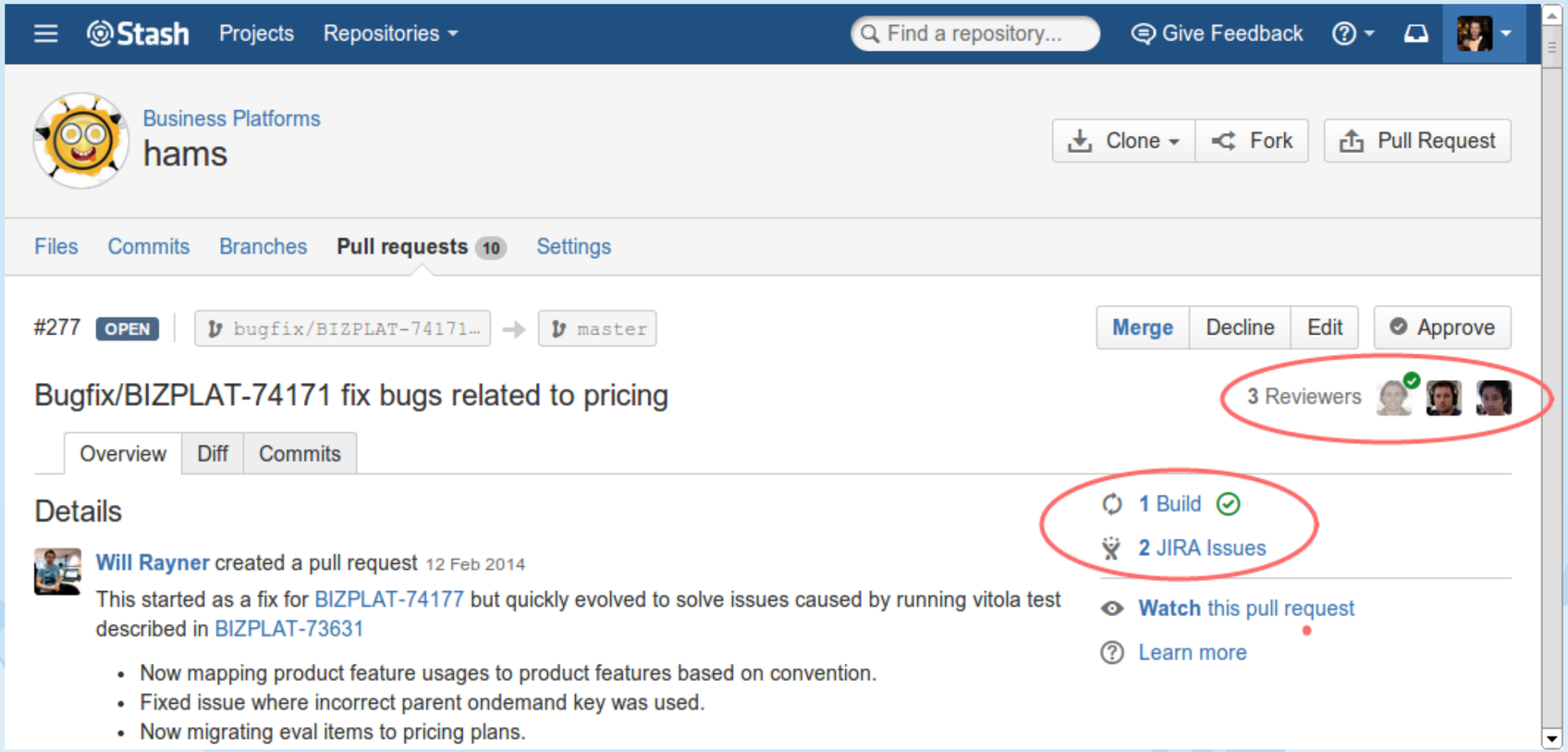
Step 3: Automatically test the branch

- Run a continuous integration tool that will automatically run tests against the branch.
- Features may not be merged until all tests are passing.
  - Stash has some features to support this.

Step 4: Code review

- No code may be merged to the release branch until reviewed by other members of the team.
- Team members have a responsibility to ensure quality.

Step 4.1: Stash testing integration



Step 5: Merge and release

- Once all reviews and tests are passed them merge to release branch
- At this point we have a separate Bamboo plan that performs a full release.

Step 6: Deploy to staging

- Allows testing of more advanced interactions and against production samples.
- More testing can occur at this point, including testing by humans.

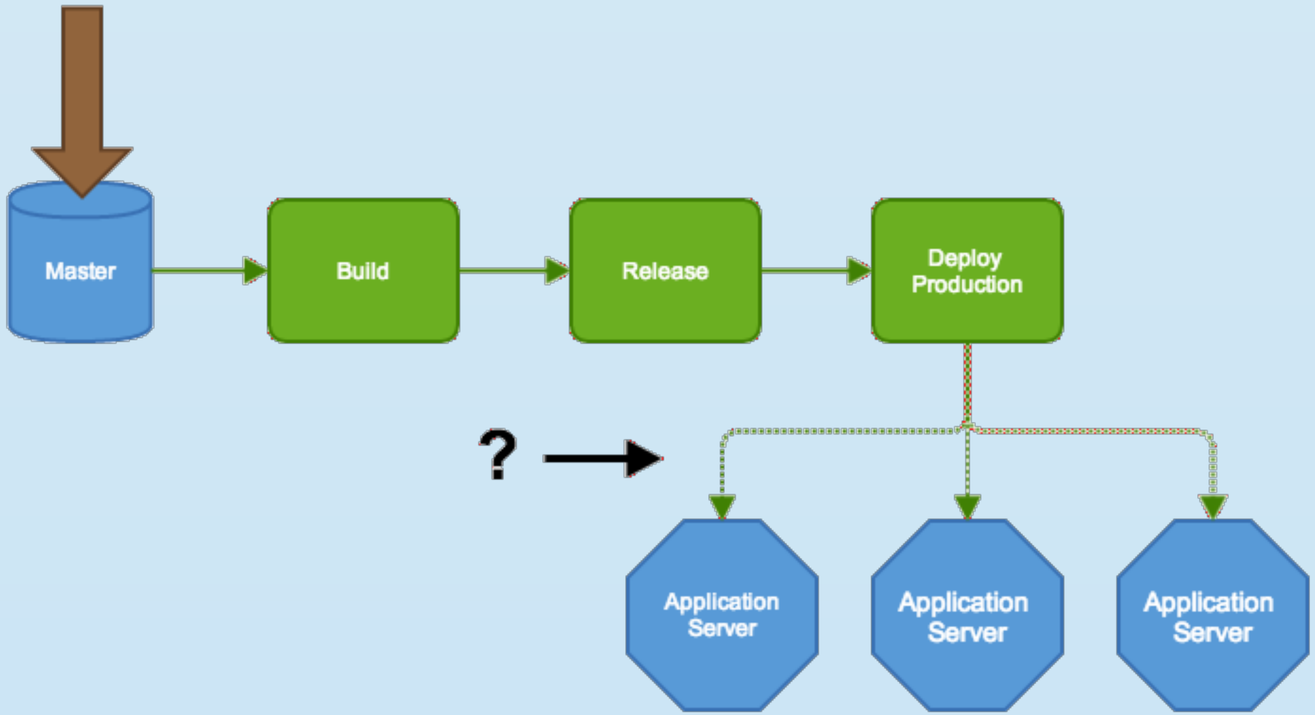
Step 7: Release to production

- Valid staging builds may be promoted up to production.

Segue: “Continuous downtime"?

- So if you’re doing all these releases, what about uptime?
- For public-facing service clustering/HA is important.
- Ideally you should be able to automate cluster configuration as part of the deployment

Last mile


Master
Build
Release
Deploy Production
?
Application Server
Application Server
Application Server

Practical issue

- How do you actually get releases onto your staging and production servers?
  - AKA “the last-mile problem”

Last mile - Puppet/Chef

- Puppet/Chef are *not* appropriate
  - .. if timing is critical
  - .. if cross-host coordination required

Last mile - DIY

- Roll your own
  - Bamboo SSH plugin + bash scripting
- Number of existing automation solutions
  - func, capistrano, SaltStack, Ansible, mcollective, Fabric…

Last mile - Direct Agent

- Bamboo (or other) agent per-node
  - SSH not required
  - Works for simple (single node) apps
  - Coordination is tricky

Last mile - Other Agents

- Agent-based frameworks
  - Powerful and flexible
  - Can parallelise deployments
  - Requires setup on all nodes
  - If you already have it setup then use it

Last mile

- SSH scripting
  - Requires management of SSH keys on agent
  - Bamboo SSH plugin
  - Scripting (Bash, Python, Ruby, etc.)
  - Automation frameworks (Ansible, SaltStack, Func, Fabric)

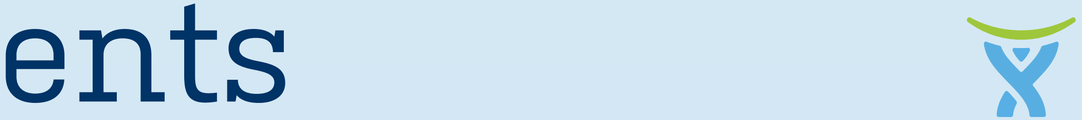
Last mile

- Our solution
  - Ansible for automation (explicit support for load-balancer integration)
  - Minimal requirements, SSH+Python
  - Bamboo pulls Ansible directly from their source repository
  - Ansible playbooks checked into git

Practical issue

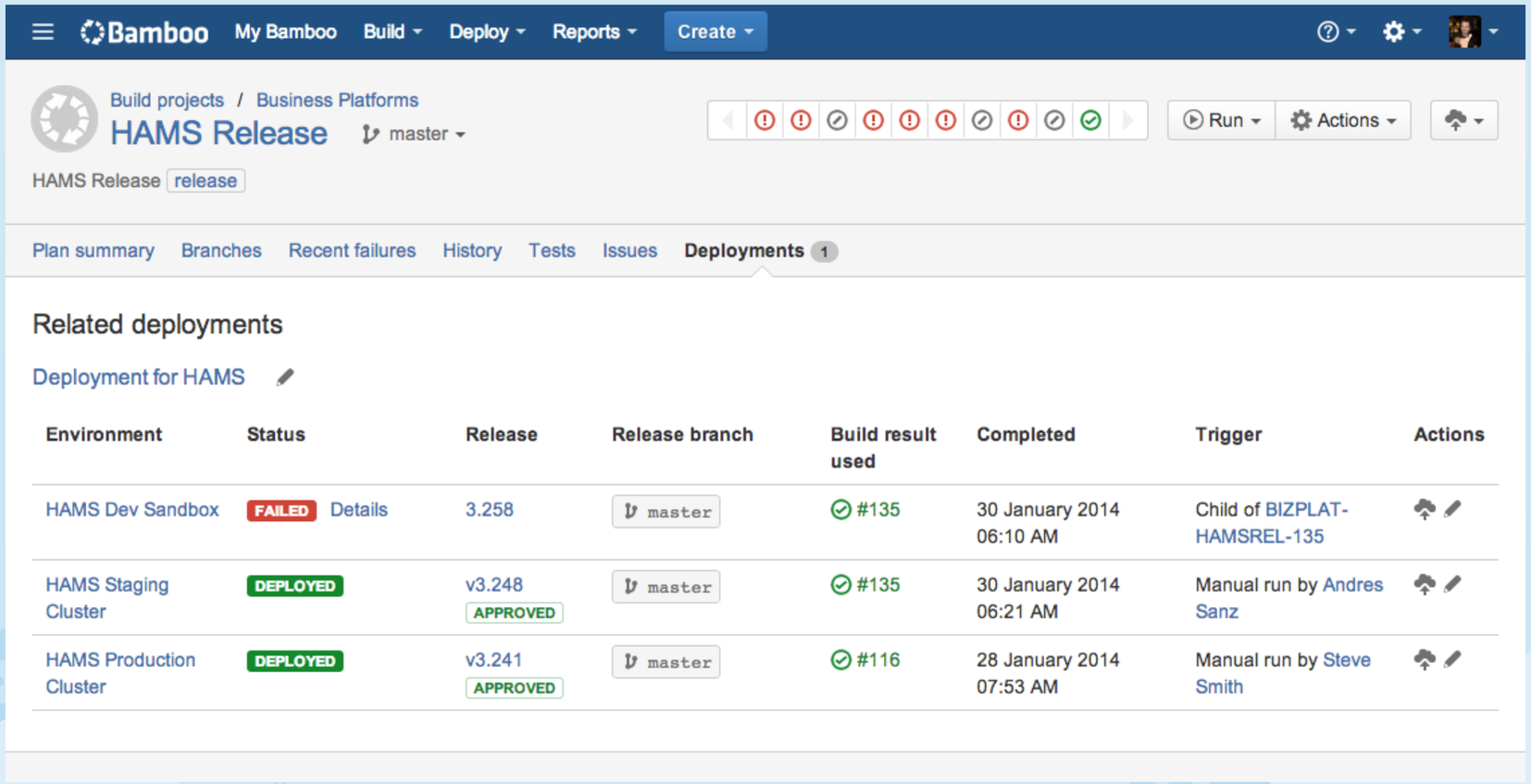
- How do you manage what has been released, and to where?
- How do you control who performs deployments?

Bamboo deployment environments



- The release build plan can be associated with certain environments
- Normal ones are dev, staging (QA) and production

Bamboo deployment environments



Bamboo deployment environments

- Environment has tasks, like a build plan
- Tasks perform the actual deployment
- Environments have permissions, limiting who may perform deployments
- Generates *releases*, which are deployed
- Has some nice integrations…

Bamboo deployment release

Deployment status

Environment	Status		Deployment result	Completed	Trigger	Actions
HAMS Dev Sandbox	Never deployed	now at 3.258				
HAMS Staging Cluster	SUCCESS		SUCCESS	30 January 2014 06:21 AM	Manual run by Andres Sanz	
HAMS Production Cluster	Never deployed	now at v3.241				

Commits tested by

The commits that were used to produce this release, were also built in the following build results

Build	Plan	Test results
#161	Business Platforms › HAMS	2667 passed
#135	Business Platforms › HAMS Release	No tests found
#1479	Business Platforms › HAMS Remote Tests	49 passed
#170	Business Platforms › HAMS Sonar	2202 passed

v3.248 details

Created
9 hours ago

Created by
Andres Sanz

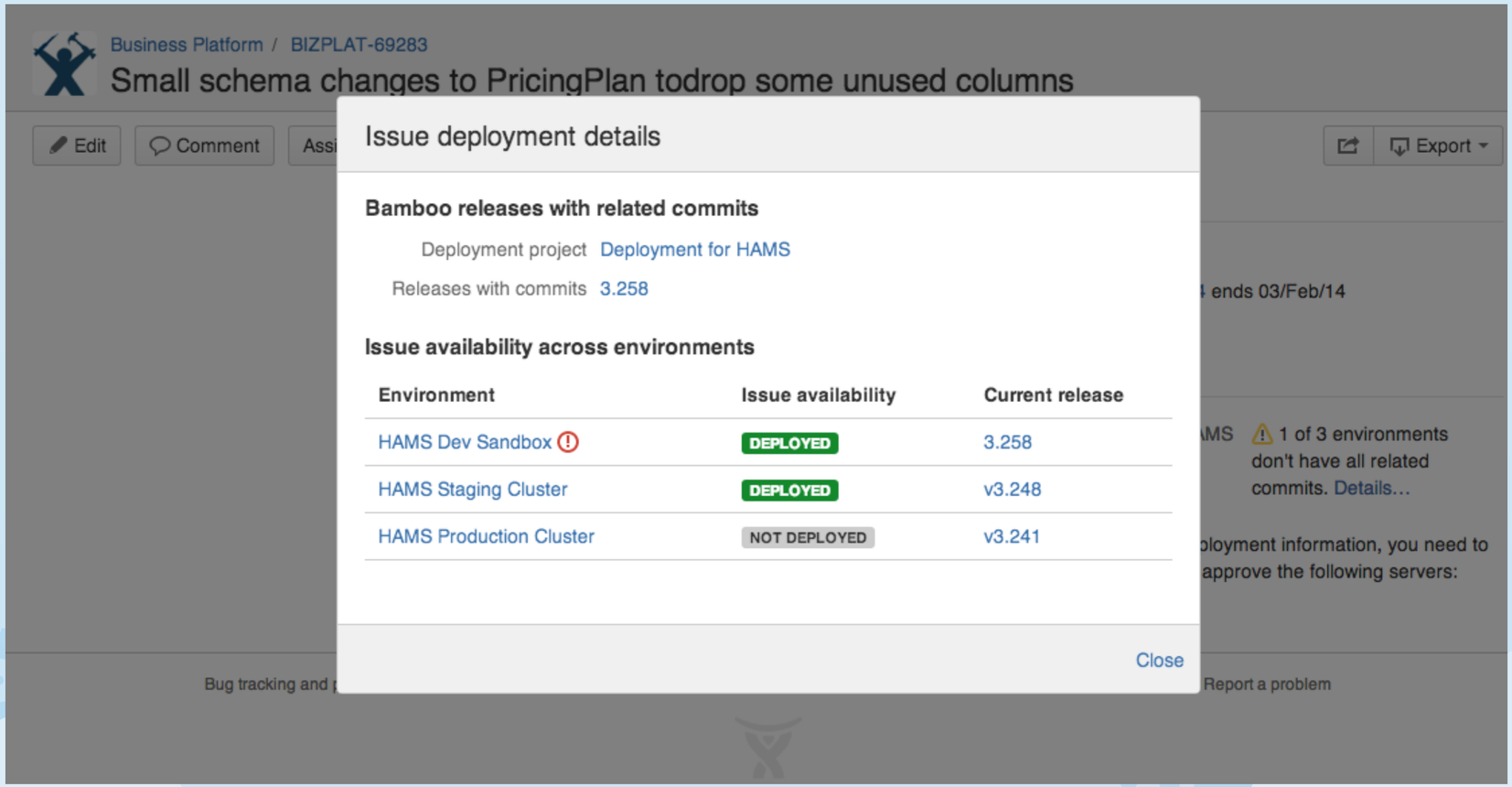
Reviewed

Deployment project
Deployment for HAMS

Artifacts provided by
#135 Business Platforms › HAMS Release

Release contents
Server Properties
Server War

Bamboo deployment JIRA integration



Procedural issues

- Where’s the oversight in all this?
- What about SoX, PCI, SEC requirements?
- Who is allowed to do releases?
- Who signs off?

Procedural issues

- Our solution - separate the infrastructure
  - Dedicated Bamboo server for business software
  - Dedicated agents for building
  - Separate, dedicated agents for deployment

Procedural issues

- Access controls
  - Build team/admins control the server
  - Business analysts define features
  - Devs code, review, merge and release
  - Features pushed to staging for BA review
  - BAs can promote releases to production

Questions?

Steve Smith

@tarkasteve

ssmith@atlassian.com

http://www.slideshare.net/tarkasteve/