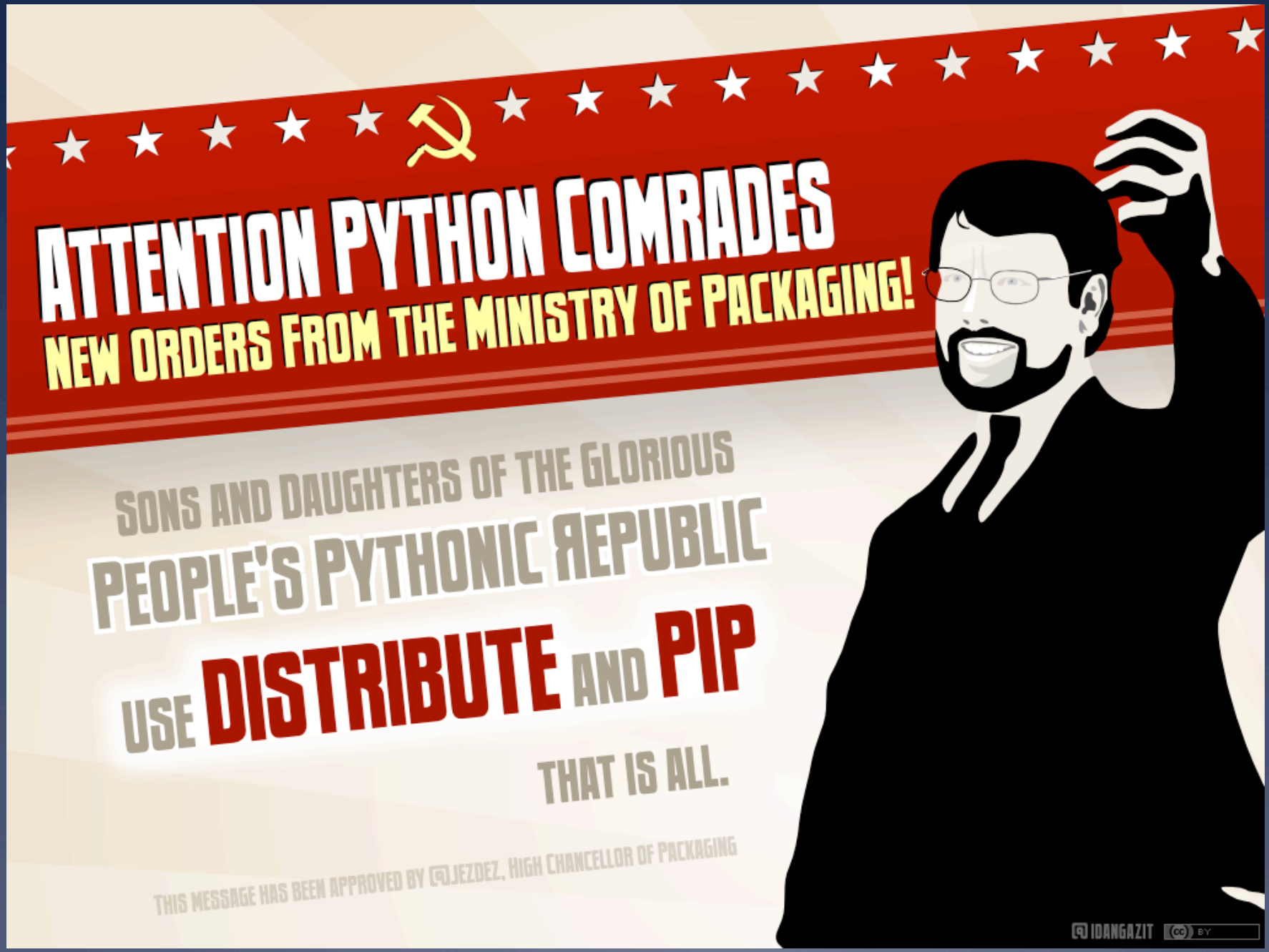

ATTENTION PYTHON COMRADES
NEW ORDERS FROM THE MINISTRY OF PACKAGING!
SONS AND DAUGHTERS OF THE GLORIOUS
PEOPLE'S PYTHONIC REPUBLIC
USE DISTRIBUTE AND PIP
THAT IS ALL.
THIS MESSAGE HAS BEEN APPROVED BY JEZDEZ, HIGH CHANCELLOR OF PACKAGING
IDANGAZIT
BY

Why not run all your tests, all the time, on all your platforms?

C. Titus Brown
titus@idyll.org

Outline

1. What & why – thinking about continuous integration
2. Just Use Hudson (with examples)
3. What about Buildbot?
4. Is there a better way? Architectural constraints.
5. What if …
6. Concluding thoughts & calls to action

What is continuous integration?

Build and test your project regularly and automatically.

- Makes sure your tests are run – even the slow ones on platforms your developers don't like.
- Detects cross-environment problems, versioning mismatches, etc. etc. as soon as possible.
- Mark Shuttleworth is also a fan, and *he's* been in space!

Digression: you know you need a build system and tests, right?

You must have:

* Version control
* Automated build/compile(/integrate/deploy) system
* Automated tests

…or else continuous integration doesn't make sense.

(And you're doomed.)

So, CI: It's not rocket science…

Ingredients:

1. shell script/batch file
2. cron job / scheduled task
3. Some notification system

⇒ Continuous integration!

…but it's not trivial either

- Configuration of build/tests
  - Build/test location and cleanup
  - Version control, caching and reset
  - Error reporting
- Management of configuration
  - Who decides what to build?
  - Cross-platform build? (Windows vs the world)
- Management of results and reporting
  - Centralized reporting
  - Data & stats mining of the reports over time
- Notification
  - E-mail, RSS, Twitter, cell phone, …
  - Test subsets ("Windows-only failures, of foo")

Software options

* Buildbot
* Hudson
* CruiseControl
* Bamboo
* Bitten
* Apycot
* Continuum
* Quickbuild
* …

Just Use Hudson

* hudson-ci.org
* Good for one project/one machine, on up.
* Supports multiple projects, multiple builders, multiple slaves, remote "push" of build info, e-mail notification, every VCS, Web configuration, XML-RPC interface, …
* …and it's the Python testing world's dirty little secret, because it's written in Java.

Hudson example

* Basic config
* Add output XML
* Add a slave

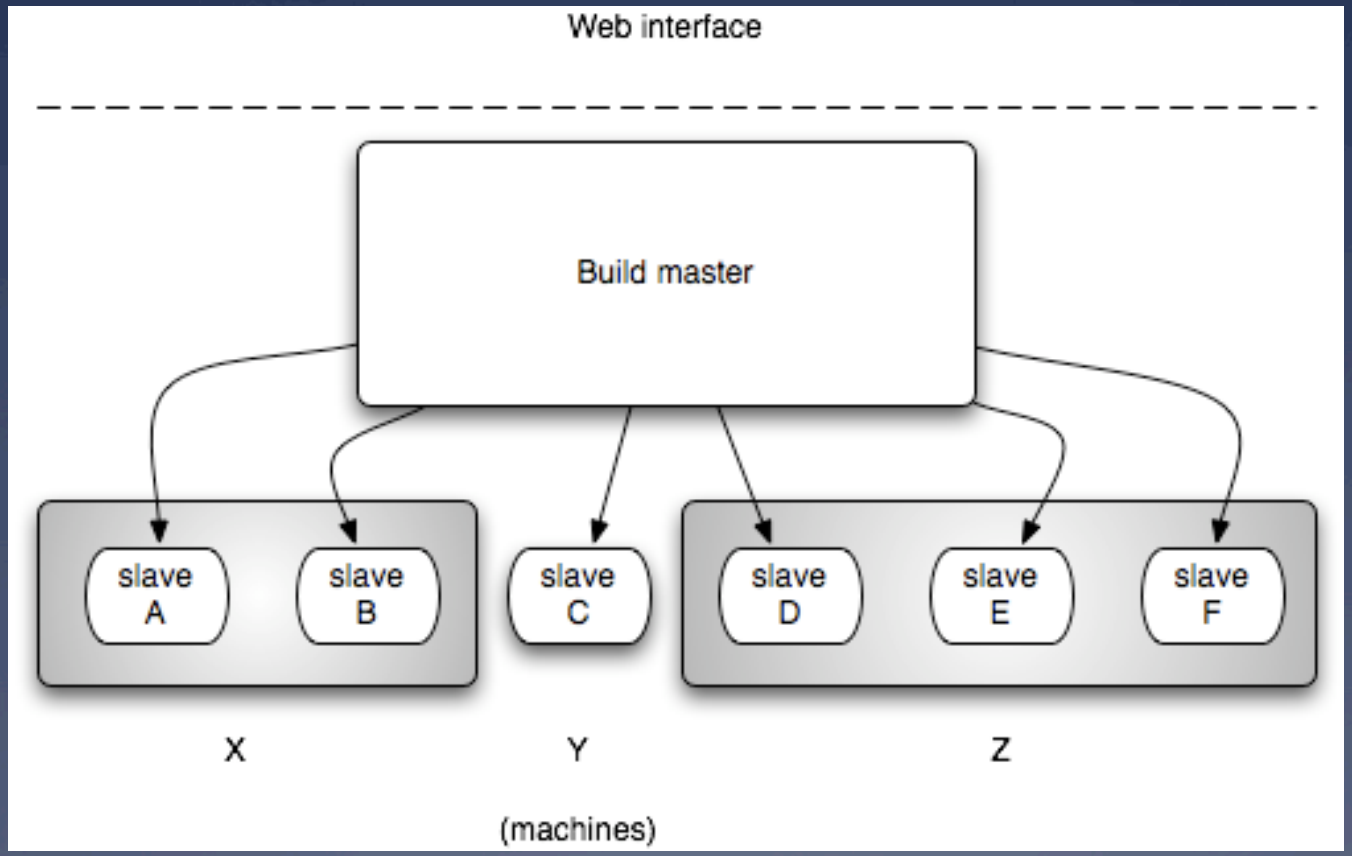
Hudson verdict

* Very friendly to install, configure, and use.
* Great for small-ish projects (like most of mine, and yours).
* Very saleable to non-geeks.
* Dirty secret of Python world: "Yeah, we used buildbot until recently. Then I switched us to Hudson and my life got a lot better."

What about Buildbot?

* What everyone thinks they should use until they actually try to use it. (…analogous to zope in 1999.)
* Very powerful, very configurable.
* Uses an (explicit) master/slave model designed around persistent connections.
* Requires a relatively high level of expertise for both configuration and maintenance.
* Reliability on Windows used to suck, but it may have improved recently with Twisted/Windows fixes.

Buildbot architecture



Buildbot verdict

- Frustratingly annoying to configure and maintain. (Although I still don't quite understand why.)
- Debugging buildbot setups makes me want to shoot myself – it's almost as bad as debugging e-mail or print queues.
- "Just use zc.buildout and collective.buildout" – now you have *two* problems.
- Twisted is a lifestyle choice that I have not personally made.
- **Unless you have a specific reason to go with buildbot, try Hudson first.**
- If you need it, you need it: excellent synchronization/ coordination, scalability, configurability.

Architectural constraints

Architecture doesn't prevent you from making arbitrary design decisions, but it *does* make some decisions **more natura**l than others.

Most continuous integration systems are built around a master/slave model: "one configuration, to rule them all."

What does this prevent? And what would a more decentralized, less tightly coupled model enable?

Introducing pony-build

(Image from headinjurytheater.com/Iocaine Powder game)

So I designed and wrote my own CI system.

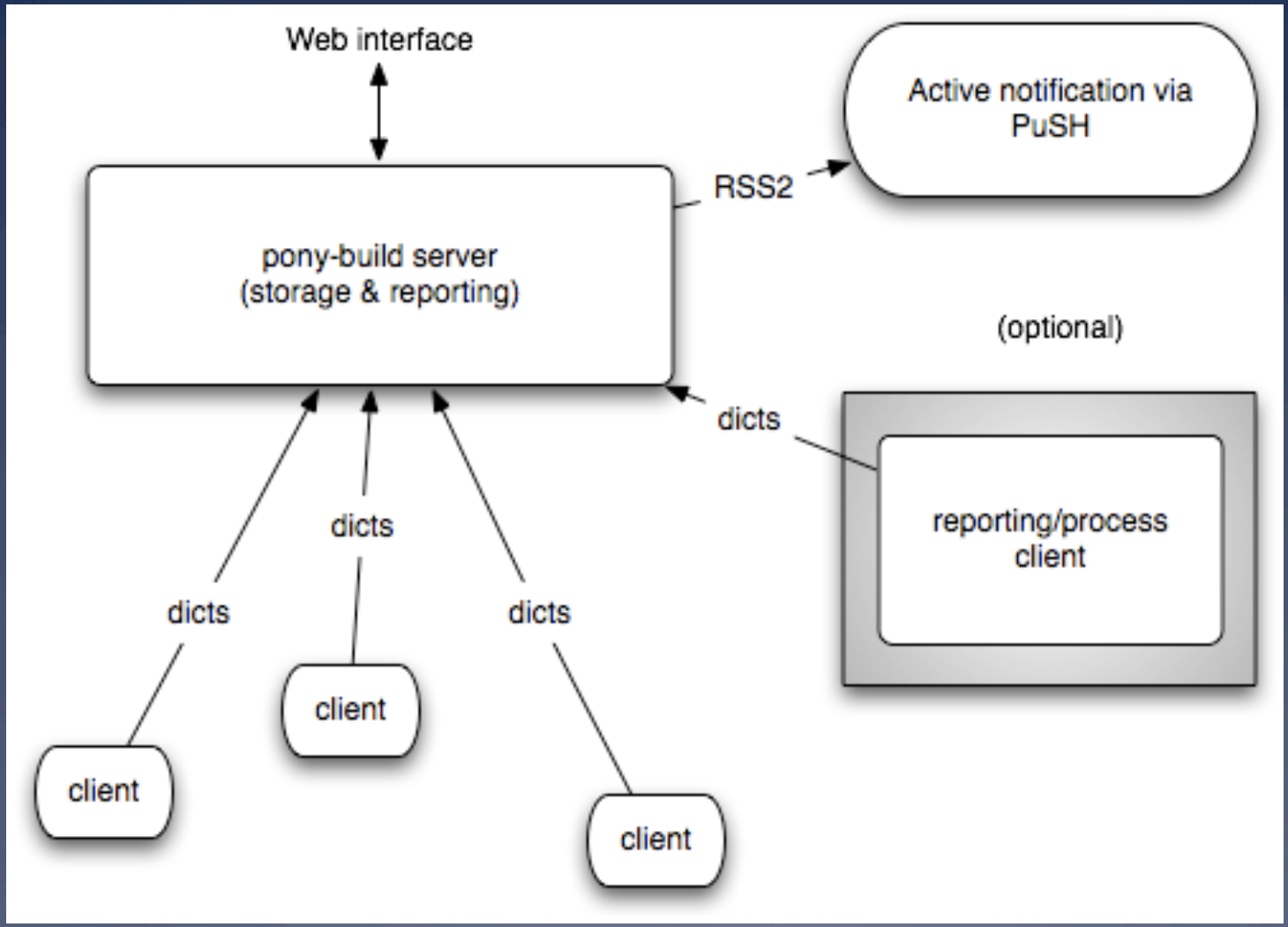
I don't know why it's called pony-build.

But everybody really *does* want a pony when it comes to continuous integration. So maybe that's why?

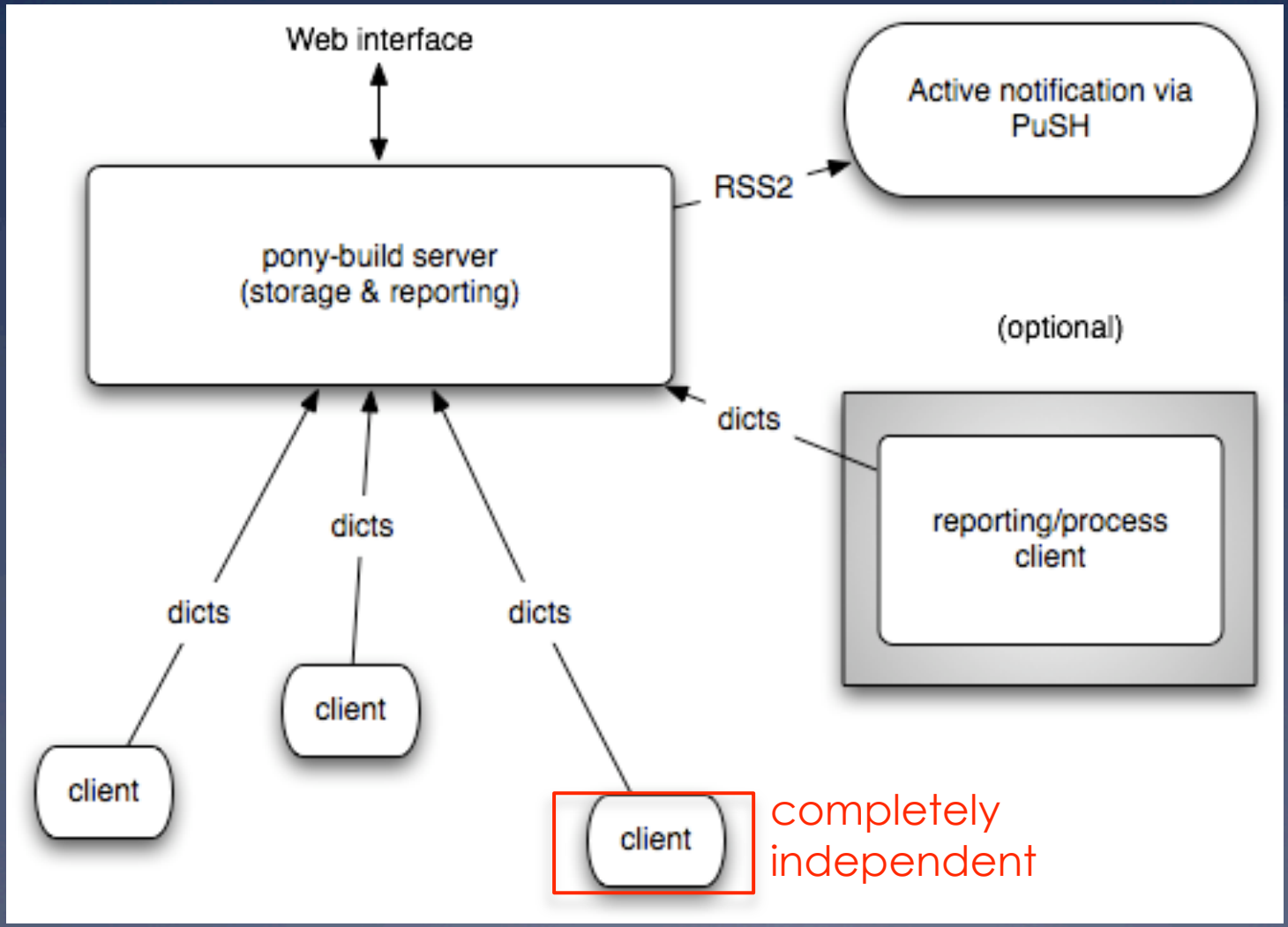
pony-build is my test-bed for CI ideas and implementation.

(Similar in concept to cmake/ctest/DART stack.)

pony-build architecture



pony-build architecture



pony-build niftiness?

* File and metadata upload.
* Built-in support for virtualenv and (soon) EC2.
* PubSubHubbub (PuSH) for active notation via webhooks.
* **Ridiculously trivial to set up on any platform - Python + single-file.**
* **Very simple to write new, custom build scripts – they are scripts.**
* **Very debuggable – they are scripts.**

…but pony-build sucks, too:

* Centralized coordination and synchronization is not natural to the pony-build model.
  * Prevents real-time monitoring/control of builds.
  * Build timeouts are tricky as well.
* Decoupling things complicates the mental model for inexperienced "users".
* Some good authentication model needs to be developed ("official" results?)
* Basic functionality exists & works, but ***not ready for general use***.

Let a thousand ponies whinny!

- Eric Holscher (and Jacob Kablan-Moss):

  "Hey, this pony-build thing is really trivial…. Screw Titus's crappy implementation, we're going to write something using Django instead, 'cause that's how we roll." (paraphrased)

  => pony_barn/devmason.org

- Awesome! (No, really – changing the world means convincing *others* to do your dirty work.)

- Interoperability might be a problem down the road. This is going to be my challenge, if anything.

What if…

(The results of titus drinking too much.)

E.T. phone home!

Wouldn't it be great if you shipped (some of) your tests with your package…

And people could run them…

And send the results to you automatically?

=> Diagnose version problems, requirements, and platform issues; debug installs.

(Inspired by cmake/ctest/DART)

(Hat tip to @illume. Shout-out to pandokia and testr.)

Meta-language for build/test

90% of Python software has some semi-custom build or test step.

(Usually it's 'test')

This is especially annoying for UNIX vs Windows, where program call functionality is different.

What about a simple cross-CI DSL for building?

Be careful: *debuggability* is paramount. How far do you go?

Build all of PyPI, all the time, on everything.

"Does this software work under Python 2.5, on Windows 7?"

"Can I download a binary package?"

And can I have all this done automatically?

From releases and dev branches?

And posted publicly?

YEEEEEEEAH!

(This is what convinced me to look beyond buildbot.)

See devmason.com, testrun.org, snakebite.org.

What about a leaderboard?

Have a standing competition with a public leaderboard, listing who has built & tested the most PyPI packages.

Post test stats/scores (cheesecake, coverage, …?) too.

What about cheaters? Reward people for finding ‘em.

(Hat tip to @jacobian and @gvwilson)

Automatic build/test on cloud

Life's too short to maintain build & test servers.

Why not pay Amazon, or Rackspace, or … to do it for you, at 10c a CPU-hour?

This should be a single configuration option in your CI system.

(Hint: Hudson has a plugin for Amazon EC2.)

Calls to action

- Stop creating, recommending, and using crappy software:
  - a.k.a. hard to install; hard to configure; built on unstable foundation; but otherwise functioning.
  - Look at your software through the eyes of an infant (i.e. me)
- Stop doing clever things in setup.py.
  - "Python's a great configuration language, until you suddenly realize, hey, I can program in it, too!" –GvR, paraphrased)
  - How about "python –m package.test"??
  - Or "python setup.py test" (in Distribute)?
- Let's standardize our build & test processes as much as possible, and then a bit more.

Acknowledgements

- Eric Henry, prototype; Owen Pierce; Trent Nelson
- UCOSP undergrad capstone, spring 2010:
  - Jack Carlson
  - Fatima Cherkaoui
  - Max Laite
  - Khushboo Shakya
- Jacob Kablan-Moss and Eric Holscher
- Brett Cannon and Jesse Noller (for general incitement)