Agile Software Development @ UC Davis

Adam Getchell

acgetchell@ucdavis.edu

https://github.com/acgetchell

http://www.linkedin.com/in/adamgetchell

ucdotnet@ucdavis.edu

TIF Ignite Session

March 28, 2012

University of California Davis

- Culture
- Collaboration
- Process
- Cloud
- Services

Culture

We are in a research institution whose output is academic publications – and yet, we do not apply the same processes of peer-review, independent verification, and publication to our software.

Publish!

We don't even fully consult with the intended users of our software before we design it!

And we rarely get their input while we're writing it!

- Not Invented Here syndrome
- Who fixes problems?

Peer Review!

= We are still working on the balance between central and departmental services

= Better Collaboration

Bug Fix costs

- The longer a bug sits, the harder it is to fix*
  - Compiler errors trivial
  - A few days ago easier
  - Months ago???
- The same applies to designing the application to begin with
- http://www.joelonsoftware.com/articles/fog0000000043.html

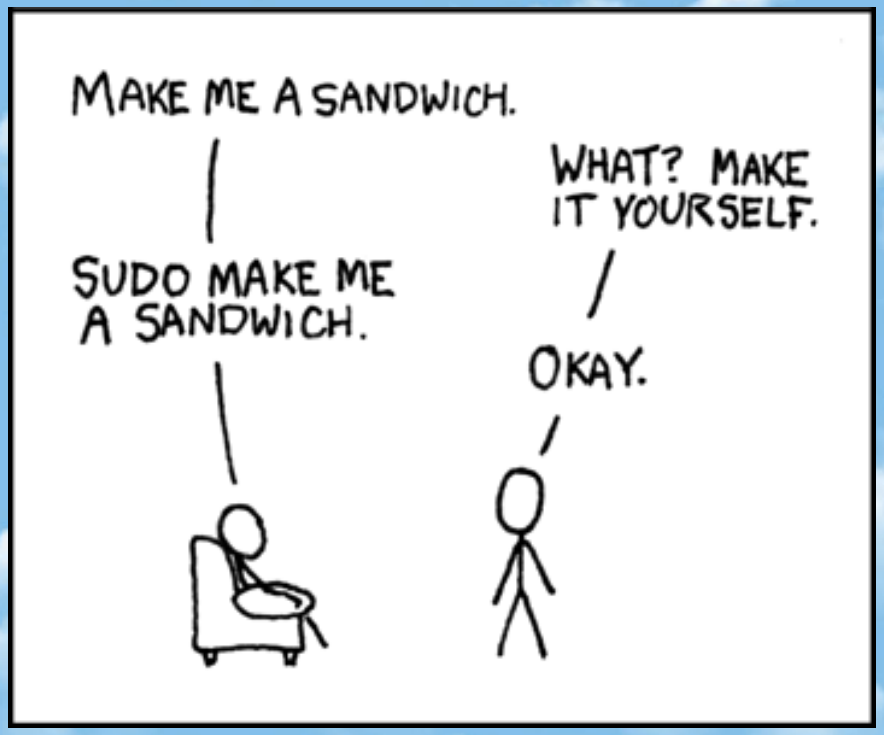
Collaboration

Get Programmers to collaborate by project, not reporting relationship

- We're Not Sharing Enough! (i.e. useless language/tool debates, service catalogs)
  - It's not about the language or tool
  - It's about usable, actionable Software Development Standards!
  - And documentation (LP)
    - Comments are NOT sufficient documentation
    - Test cases are NOT "executable" documentation
    - Need to know Why something was done, not How
  - And a Service Catalog
  - Using well-known APIs (RESTFul, OAuth, etc.)

Publish/Peer Review

Programmer (noun): An organism that can turn caffeine and alcohol into code*



http://xkcd.com/149/

http://xkcd.com/303/

* http://uncyclopedia.wikia.com/wiki/Programmer

Process

The Joel (Spolsky) Test:

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?
5. Do you fix bugs before writing new code
6. Do you have an up-to-date schedule
7. Do you have a spec?
8. Do programmers have quiet working conditions?
9. Do you use the best tools?
10. Do you have testers?
11. Do new candidates write code during interviews?
12. Do you do hallway usability testing?

Cloud

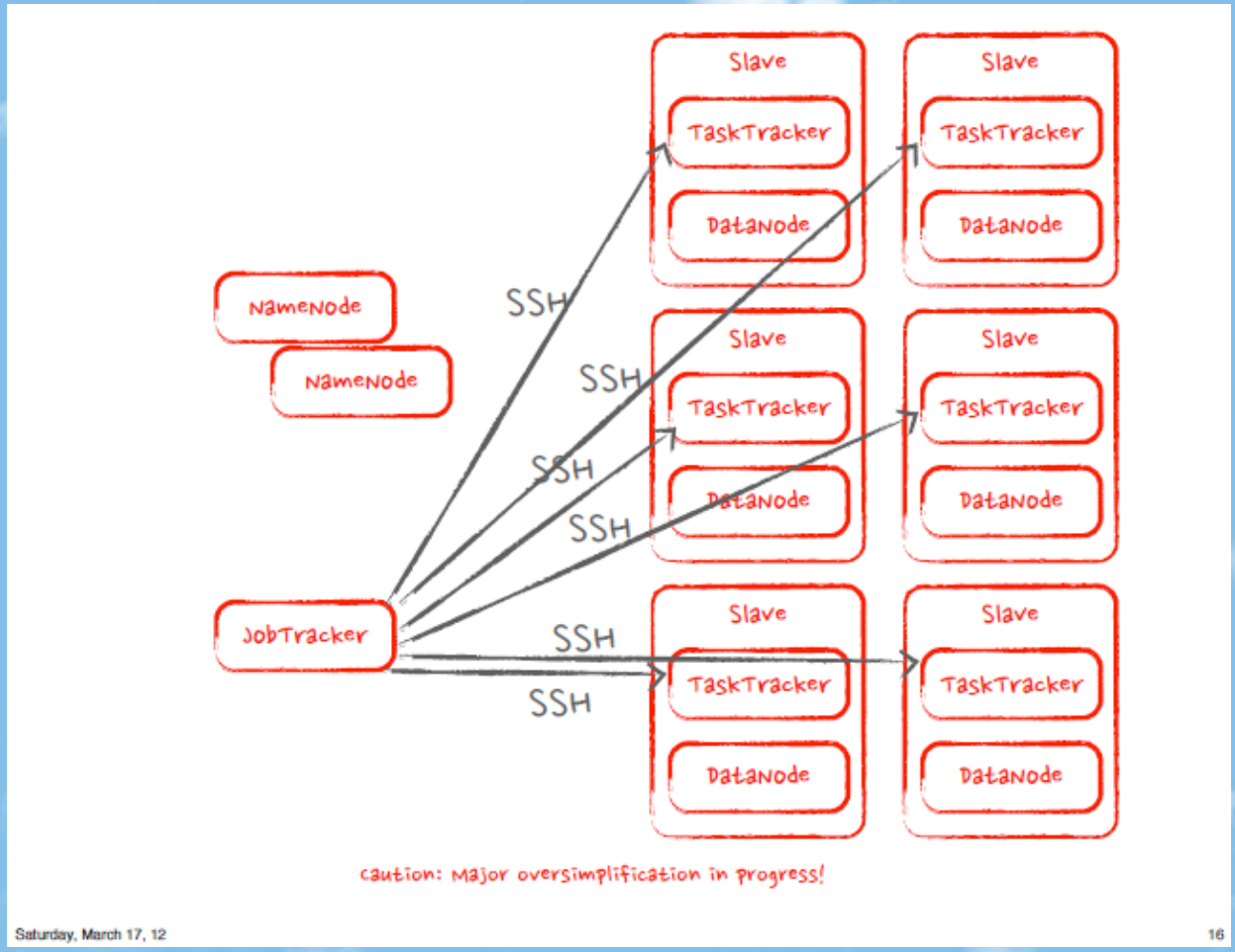
- Virtualization ≠ Cloud Services

-> DevOps = ??? (a combination of the above)

Cloud

caution: Major oversimplification in progress!

Cloud



Cloud

node = class of compute server to be instantiated in the cloud

```
(def small-node
   (node-spec
     :image {:os-family :ubuntu
             :os-version-matches "10.10"}
     :hardware {:min-cores 2 :min-ram 512}
     :network {:inbound-ports [22 80]}))
```

Cloud

server spec = abstraction over a set of actions to configure a server

```
(def hadoop
  (server-spec
     :phases
     {:configure
      (phase-fn
       (java/java :jdk)
       (hadoop/install :cloudera))}))
```

Cloud

```
(def task-trackers
  (group-spec "task-tracker"
      :extends [hadoop task-tracker]
      :node-spec big-node))

(def job-trackers
  (group-spec "job-tracker"
      :extends [hadoop job-tracker]
      :node-spec small-node))
```

Cloud

Action!

```
(converge {task-trackers 10
           job-trackers 1}
          :phase [:configure :start]
          :compute aws-ec2)
```

... or even...

```
(converge {task-trackers 5
           job-trackers 1}
          :phase [:configure :start]
          :compute virtualbox)
```

Cloud

crate function = schedules defines actions to be executed on the target nodes

```
(def-crate-fn authorize-key
  "Authorize a public key on the specified user."
  [user public-key-string
    & {:keys [authorize-for-user]}]
  ...
  (directory dir :owner target-user :mode "755")
  (file auth-file :owner target-user :mode "644")
  (exec-checked-script
   (format "authorize-key on user %s" user)
   (echo (quoted ~public-key-string)
         ">>" ~auth-file)))
```

Services

- Why do we buy cloud computing from Amazon, an online bookstore?
- Steve Yegge's Google Platform* rant:
  - All teams expose their data and functionality through service interfaces
  - Teams must communicate with each other through these interfaces
  - There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back doors whatsoever
  - It doesn't matter what technology is used
  - All service interfaces, without exception, must be designed from the ground up to be externalizable.
  - Anyone who doesn't do this will be fired

- http://news.ycombinator.com/item?id=3101876

Services

- Authentication
  - CAS is UC Davis only, difficult for “post-PC” devices
  - Shibboleth is hard to setup, hard to use, narrow in scope
  - OAuth 2.0 is the rest of the web
    - Google, Facebook, major cloud providers
    - Why not UC?
- Non-secured APIs (hint: OAuth 2.0)
  - SOAP is insecure without a shared secret
  - With a shared secret, it’s difficult to mange
- Authorization (API)
  - CatBert v4 is a web API for Authorization

Let’s fix it together!

Adam Getchell
acgetchell@ucdavis.edu
https://github.com/acgetchell
http://www.linkedin.com/in/adamgetchell
ucdotnet@ucdavis.edu

TIF Ignite Session

March 28, 2012

University of California Davis