PRACTICAL SOFTWARE ENGINEERING FUNDAMENTALS

Sam is an architect at Cerner. He leads a dedicated group of engineers focused on making web application development and deployment easier up and down the stack and back and forth through the ecosystem. Sam's background has always been in web development. Throughout his career, Sam has experienced web development using Java, Python (first love), and Ruby (current SO). Sam also had a fling with Node one Summer. He is a 2005 graduate of Kansas State University, enjoys EDM music when he works, likes Star Wars, and playing board games with a zombie theme.

http://engineering.cerner.com

10 years

Architect on HealtheIntent Application Infrastructure

Agile development - scrums, iterations

http://engineering.cerner.com

"The best things in life are free."

–COCO CHANEL

Encourage everyone to ask questions through out my presentation. If you ask a question, you get a t-shirt. I reserve the right not to answer your questions, but you can absolutely ask them.

WHAT YOU'LL SEE TODAY

- Every Day Best Practices / Tools / Skills
- Cerner DevAcademy
- Lots of pictures

SOFTWARE ENGINEERING

- "Software engineering is the **application** of a **systematic**, **disciplined**, **quantifiable approach** to the **design**, **development**, **operation**, and **maintenance** of software"

Margaret Hamilton, credited with coining the term software engineering, standing with the code she designed for the Apollo 11.

First quote on wikipedia from ACM

Second quote on wikipedia from IEEE Standard Glossary of Software Engineering Terminology

http://en.wikipedia.org/wiki/Software_engineering#mediaviewer/File:Margaret_Hamilton.gif

http://en.wikipedia.org/wiki/Software_engineering

THERE'S A LOT TO SOFTWARE ENGINEERING

Testing & CI

SCM

Monitoring

Documentation

Performance Profiling

Code Reviews

Resourcefulness

Debugging & Troubleshooting

Team Work

Automation

Ownership Cost

Issue Tracker

Globalization

Security

Passivity

Error Handling

Assessing Code

Fault Tolerance

Community

Sharpening Saw

REST APIs

Web

Mobile

…

In addition to learning about data structures, algorithm analysis, and operating systems, there is a common set of technical and soft skills that are crucial to being a good software engineer. In this talk, I will go over some of these skills, discuss their value, how they are useful, and what you can do to gain them. I know this all seems overwhelming, but we won't be going over all of these.

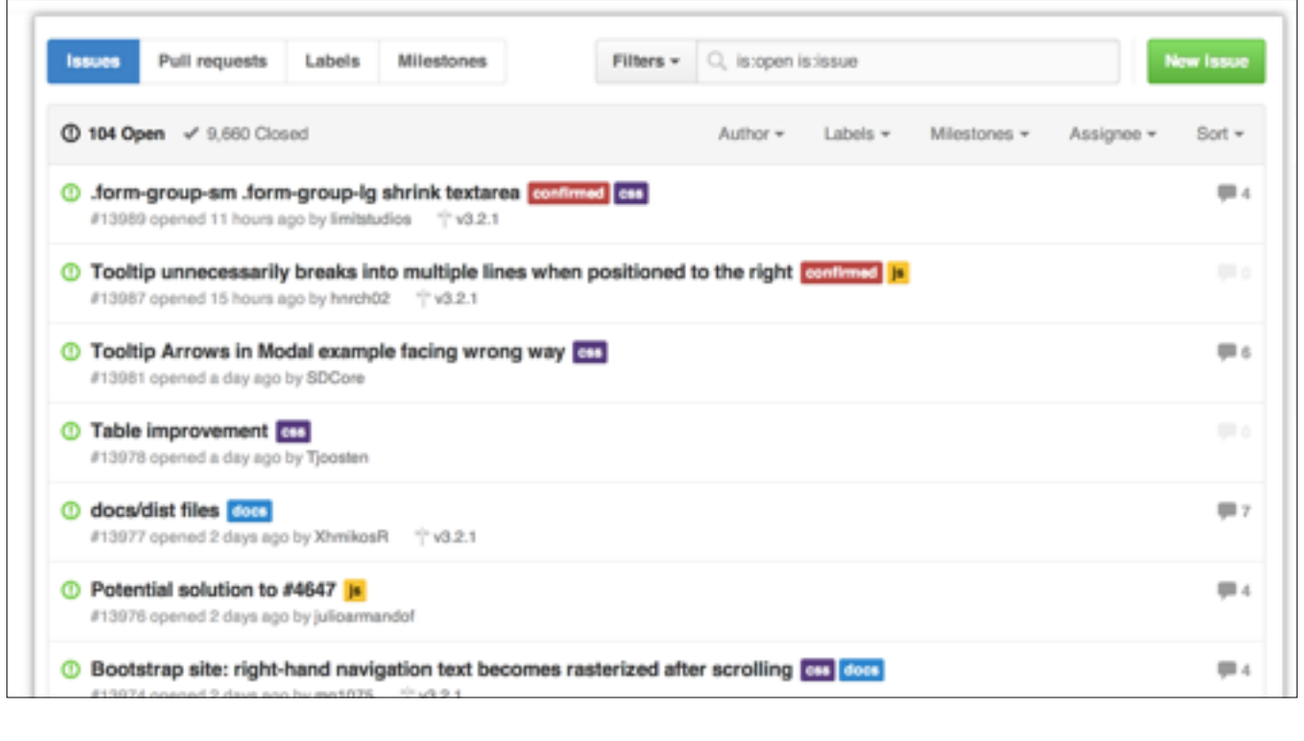
Overwhelming, so give you a break already.

Best material right from the beginning…Star Wars + Cats

ISSUE TRACKING

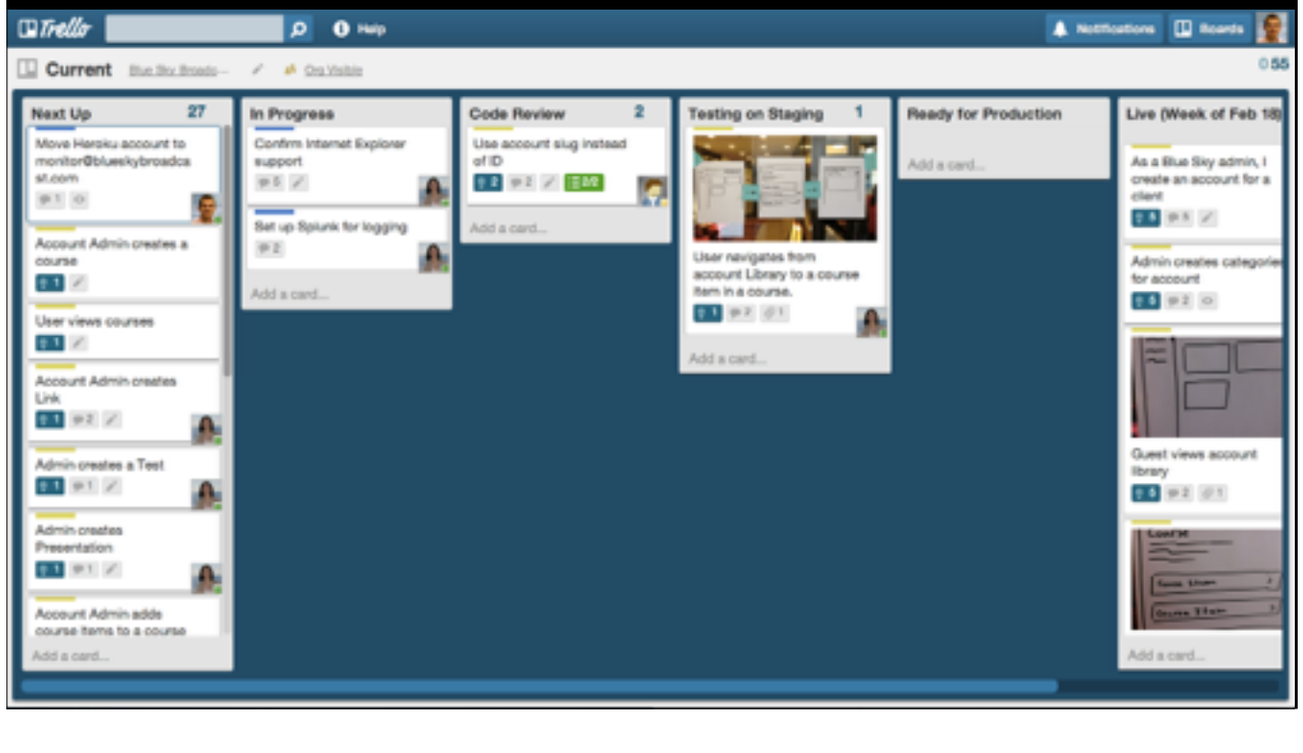
People should learn to use an issue tracker. Bugzilla, Jira, Github, whatever. Use one. Don't use excel, or keep things in a book or worst in your head. Using an issue tracker, not only to track issues, but to share knowledge and have conversations. We use jira, and this is where we document a lot of the decisionn. Of course, it also allows us to track progress and make it transparent to others.

GITHUB ISSUES



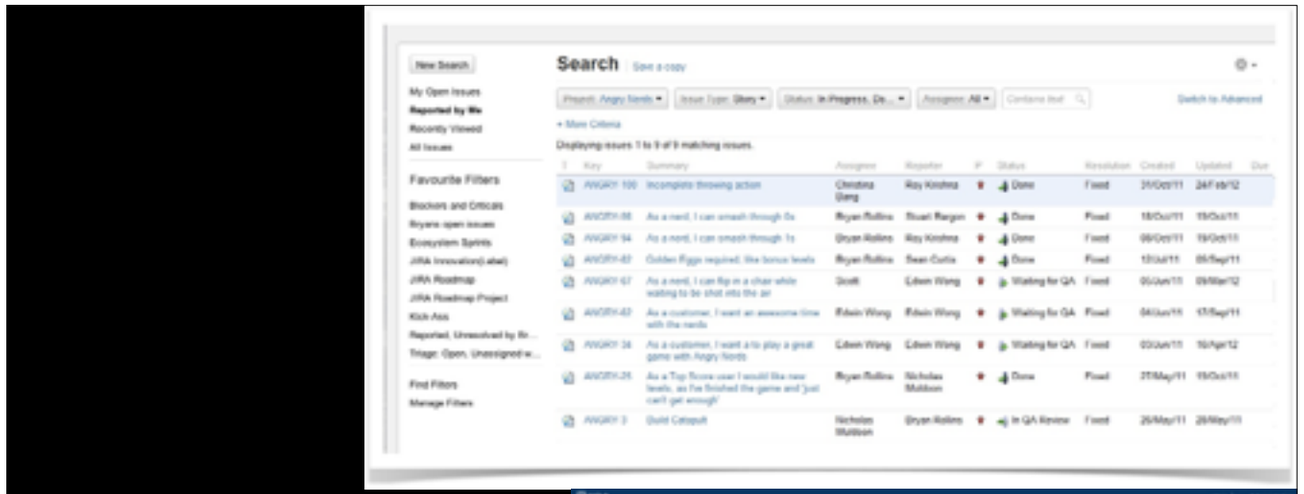
milestones, labels, comments, notifications, assignees, search,

TRELLO



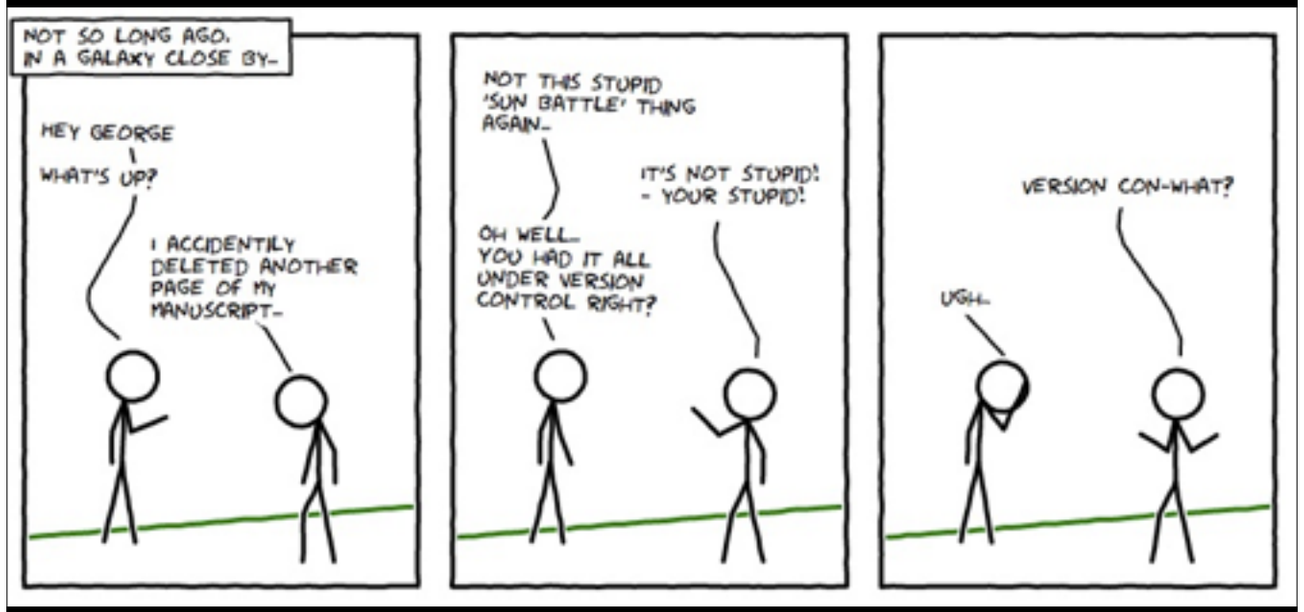
Kanban style - Cards and Lists

JIRA



Agile, scrums, planning, notification, assignments, due dates, statuses, conversation, attachments, description, category, priority

VERSION CONTROL SYSTEM (VCS)
AKA SOURCE CODE MANAGEMENT (SCM)



VCS/SCM - Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

people should know how to use Git, or at least Svn, CVS, or Mercurial. And when I say use, I don't mean they know what it is and how to checkout and commit, but know some pretty common SCM workflows like Gitflow [1] or Githubflow [2]. They should understand proper use of feature branches, hotfix branches, merging, and resolving conflicts as they would use in a day-to-day coding environment with multiple people.

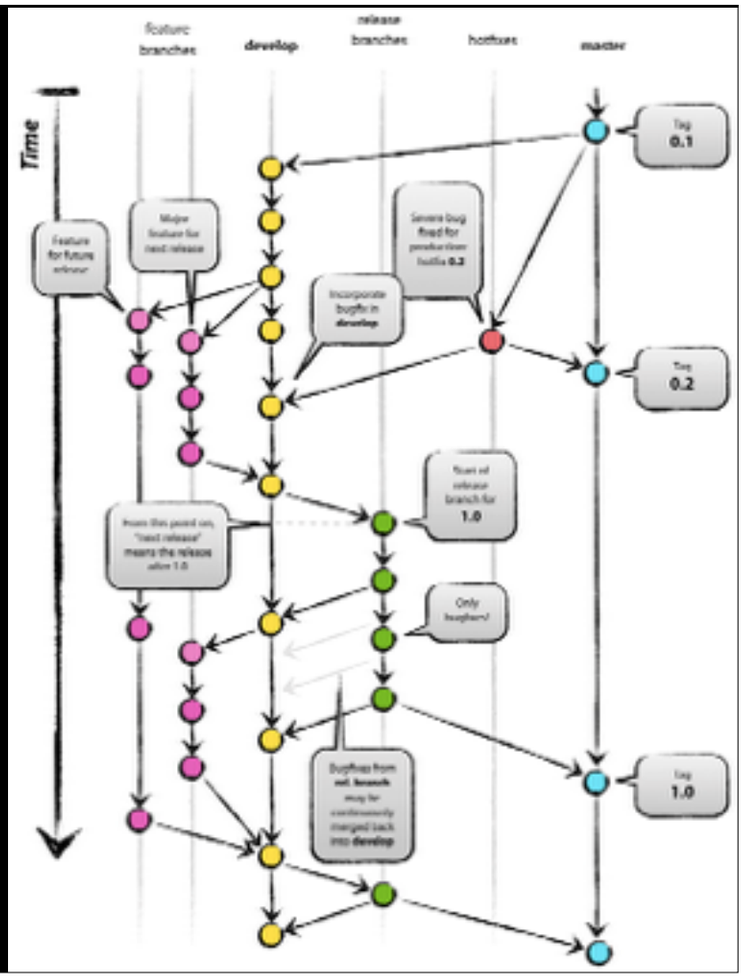
3 main ones out there, but really, git is the oprah of scm

GITHUB

Free for open source, pay for private repos, fork, clone, pull requests, merges, keyboard shortcuts, markdown parser, pages, wiki, diffs, syntax highlighting, API, hooks for jenkins, travis

Bitbucket is good too. It’s free, even for private repos, but everyone uses github, and you should keep it public.

GITFLOW



master branch

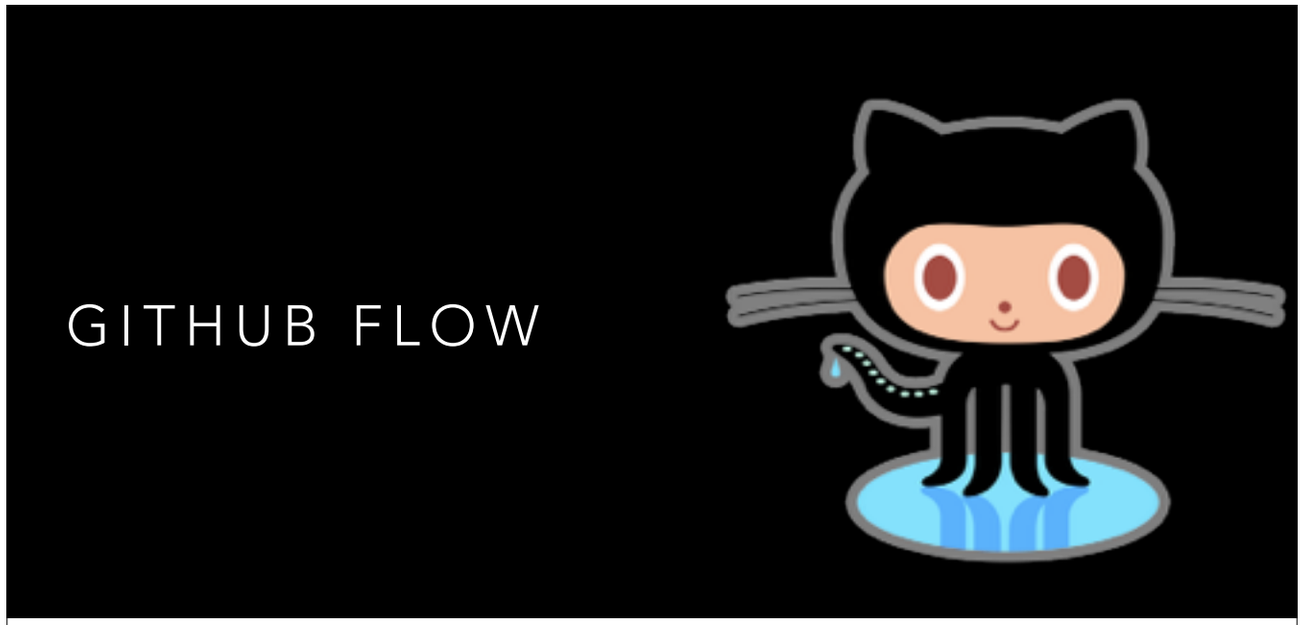
feature branch

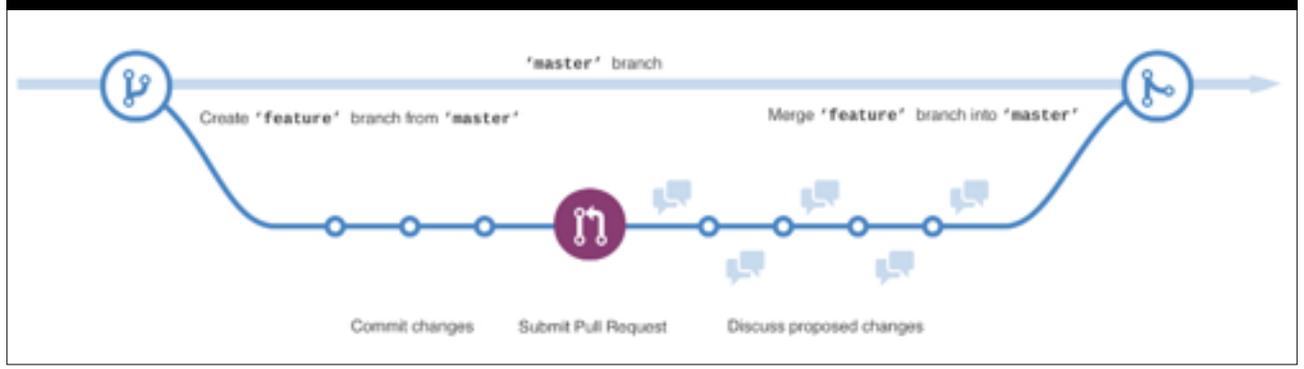
develop branch

release branch

hotfix branches

GITHUB FLOW





master branch

fork

feature branch

commit

Pull Request

Merge

"print statements are the best way to debug and troubleshoot errors!"

DEBUGGING

debugger **use it with** print statements

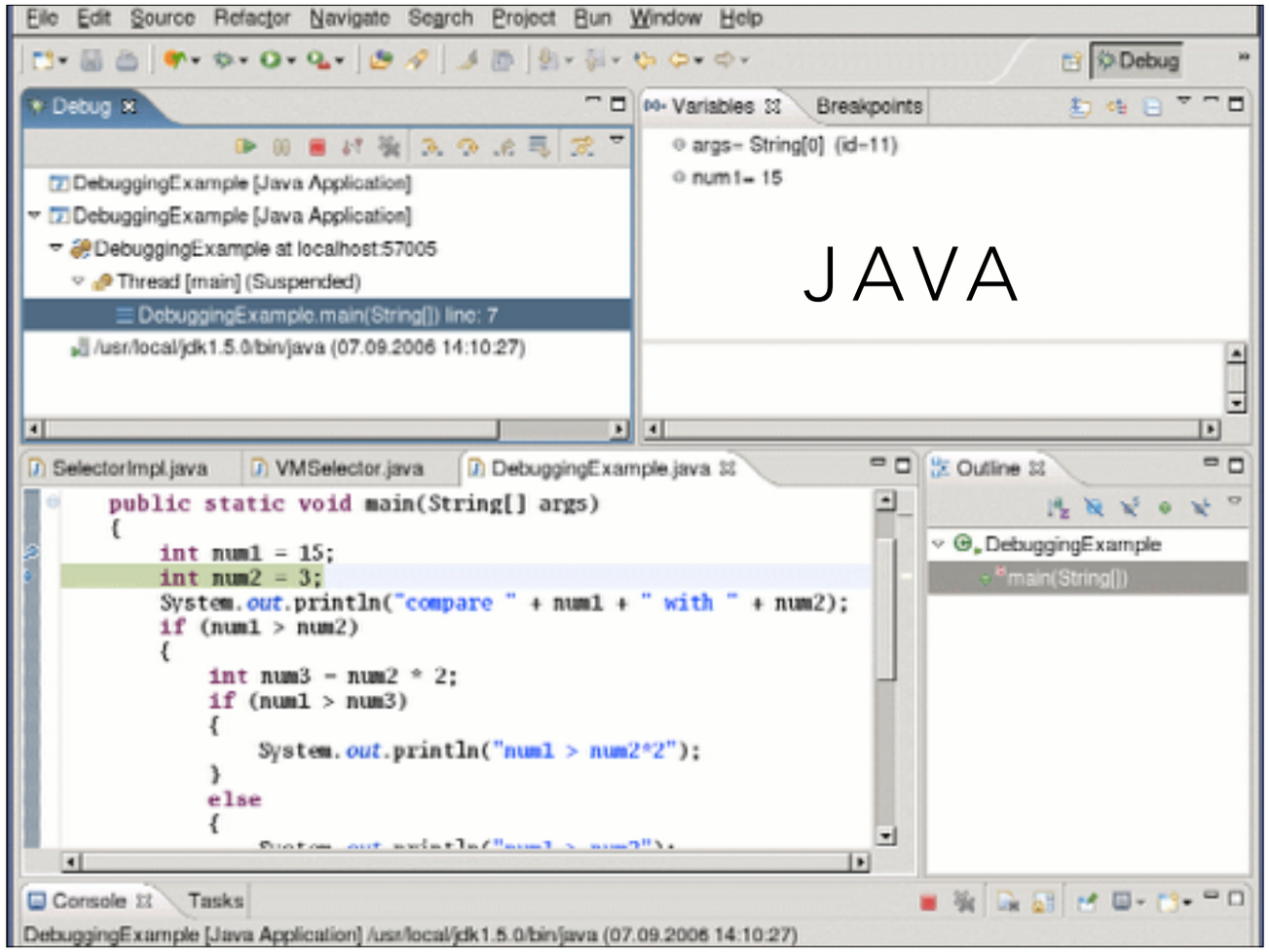
breakpoints

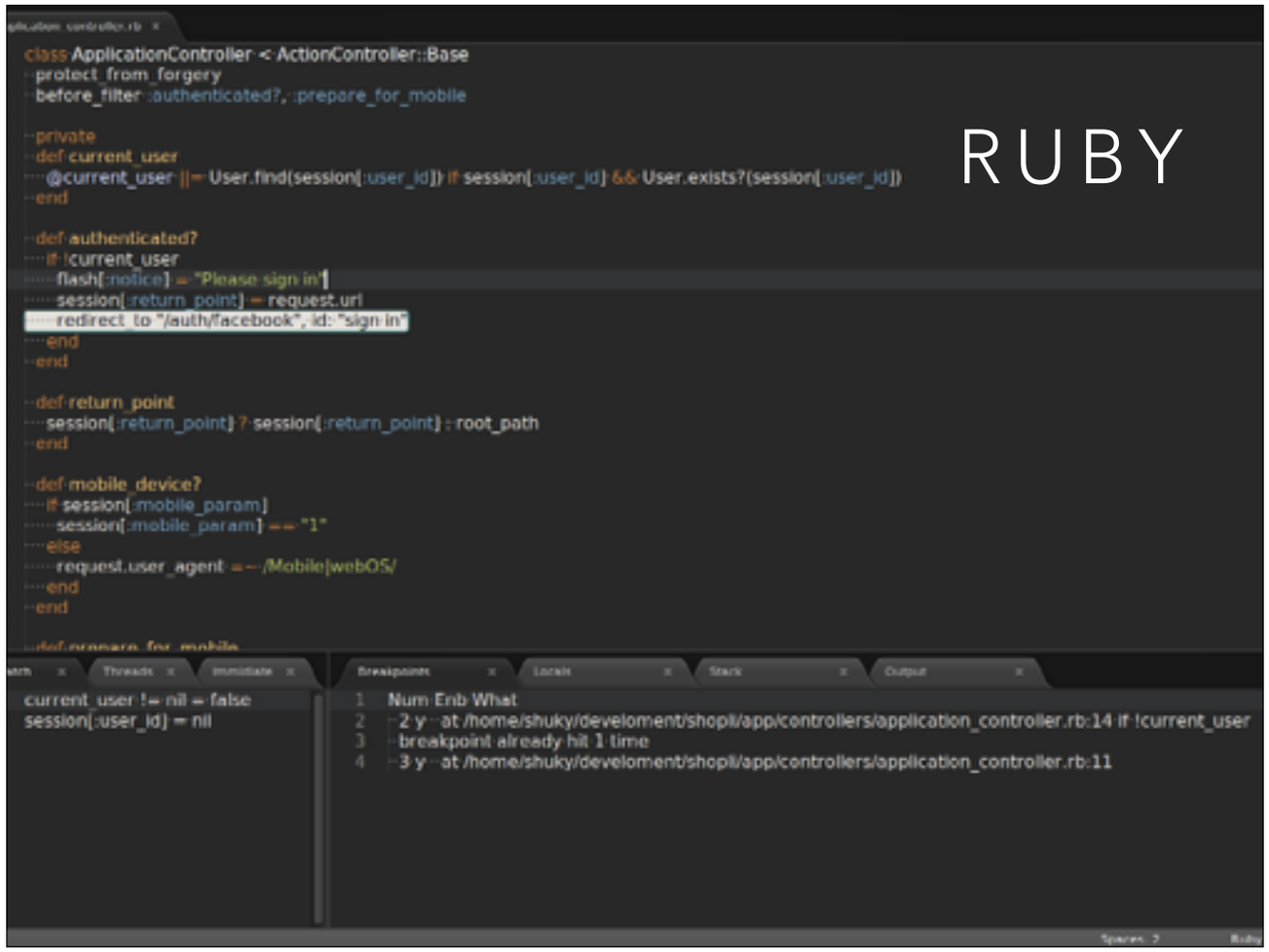
watch variables

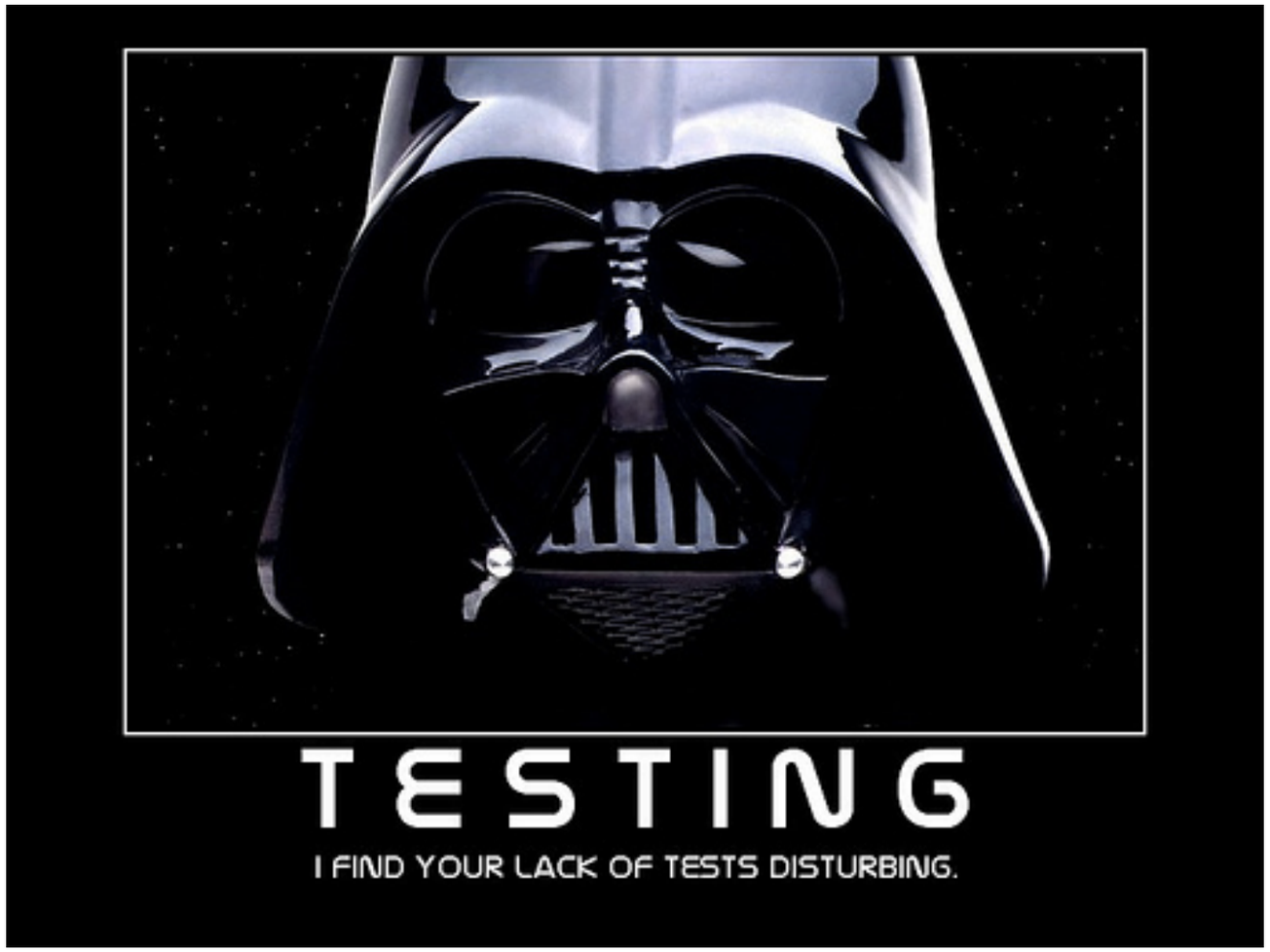
step through, into, over

conditional break points

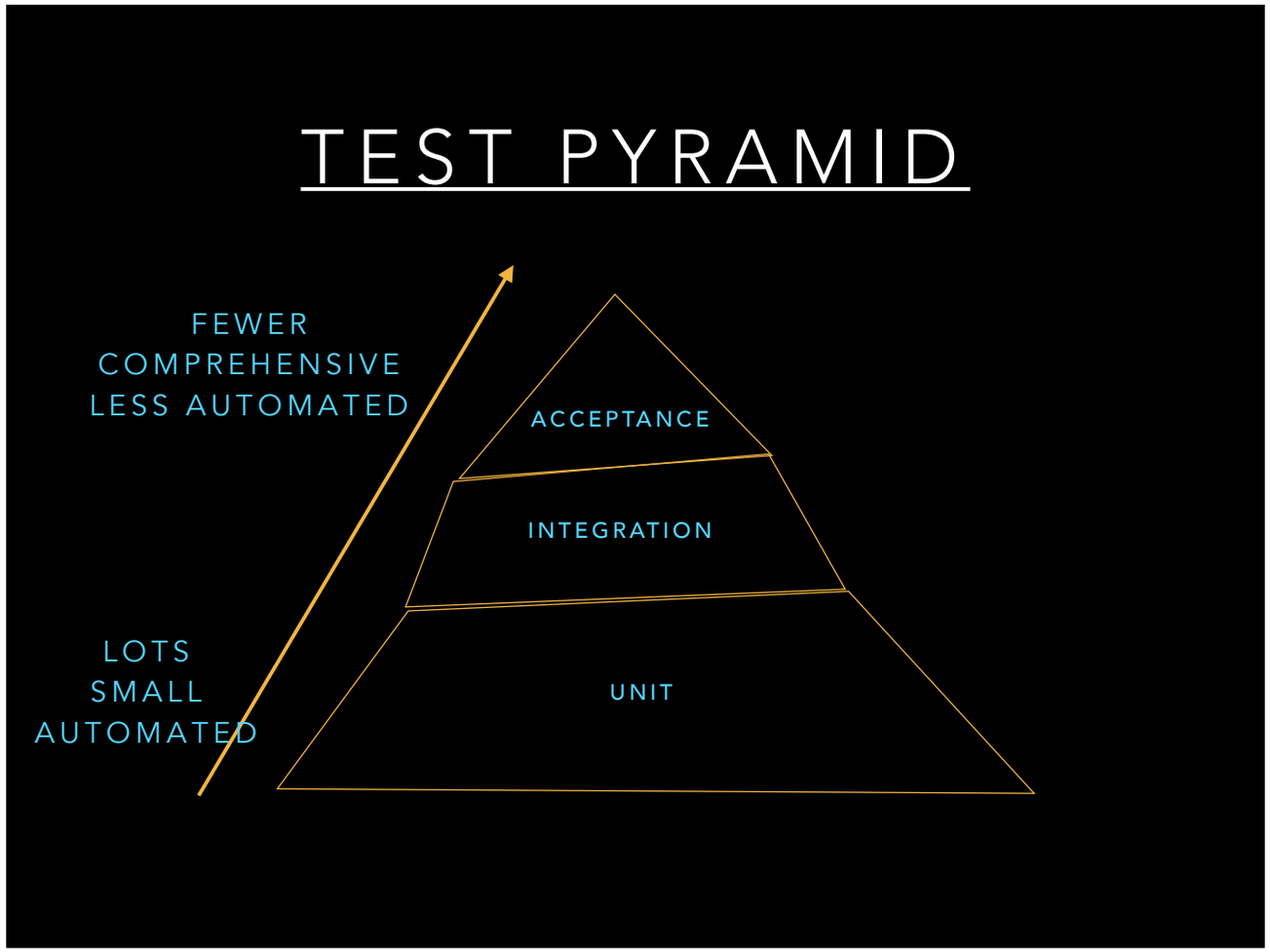
stack


File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Debug
DebuggingExample [Java Application]
DebuggingExample [Java Application]
DebuggingExample at localhost:57005
Thread [main] (Suspended)
DebuggingExample.main(String[]) line: 7
/usr/local/jdk1.5.0/bin/java (07.09.2006 14:10:27)
Variables
Breakpoints
args= String[0] (id=11)
num1= 15
JAVA
SelectorImpl.java
VMSelector.java
DebuggingExample.java
public static void main(String[] args)
{
int num1 = 15;
int num2 = 3;
System.out.println("compare " + num1 + " with " + num2);
if (num1 > num2)
{
int num3 = num2 * 2;
if (num1 > num3)
{
System.out.println("num1 > num2*2");
}
else
{
Outline
DebuggingExample
main(String[])
Console
Tasks
DebuggingExample [Java Application] /usr/local/jdk1.5.0/bin/java (07.09.2006 14:10:27)


class ApplicationController < ActionController::Base
protect_from_forgery
before_filter :authenticated?, :prepare_for_mobile
private
def current_user
@current_user ||= User.find(session[:user_id]) if session[:user_id] && User.exists?(session[:user_id])
end
def authenticated?
if !current_user
flash[:notice] = "Please sign in"
session[:return_point] = request.url
redirect_to "/auth/facebook", id: "sign in"
end
end
def return_point
session[:return_point] ? session[:return_point] : root_path
end
def mobile_device?
if session[:mobile_param]
session[:mobile_param] == "1"
else
request.user_agent =~ /Mobile|webOS/
end
end
Threads
Immediate
Breakpoints
Locals
Stack
Output
current_user != nil = false
session[:user_id] = nil
Num Enb What
2 y at /home/shuky/develoment/shopli/app/controllers/application_controller.rb:14 if !current_user
breakpoint already hit 1 time
3 y at /home/shuky/develoment/shopli/app/controllers/application_controller.rb:11
RUBY



unit, functional, integration are just some basic concepts, but people should understand why writing tests is not just upfront time, but how over the long run how it promotes quality, reduce upkeep time, as well as a peace of mind. Having a good set of tests, it gives me confidence that the code not only works now, but if someone goes and refactor, it will help to assure them that whatever changes they made, it still works. This shouldn't be one assignment thing, but rather an expectation that code have corresponding tests, and it tests adding meaningful value (that's the tricky part).



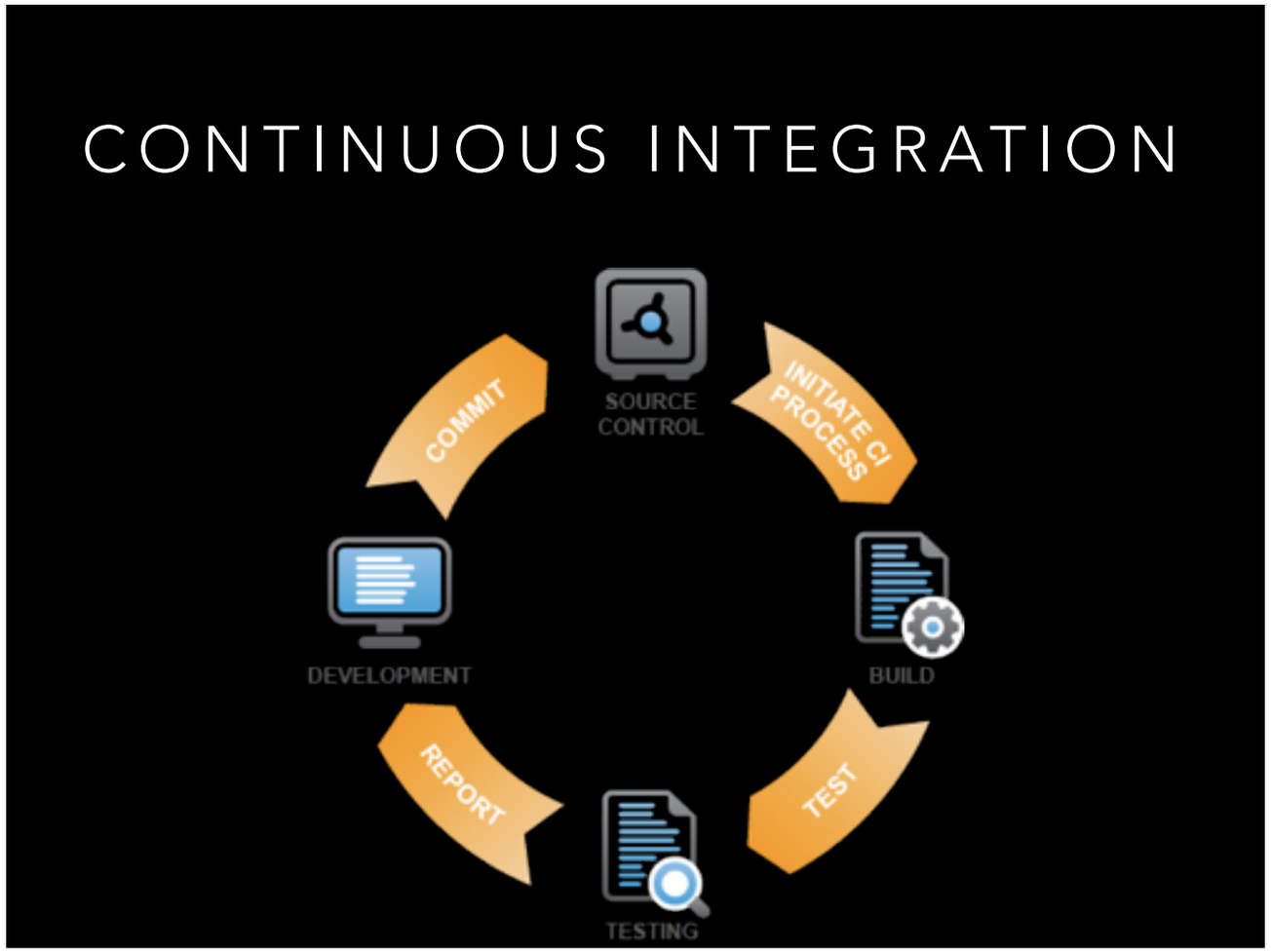
I drew this my self.

Automate everything you can

unit - single units of functionality

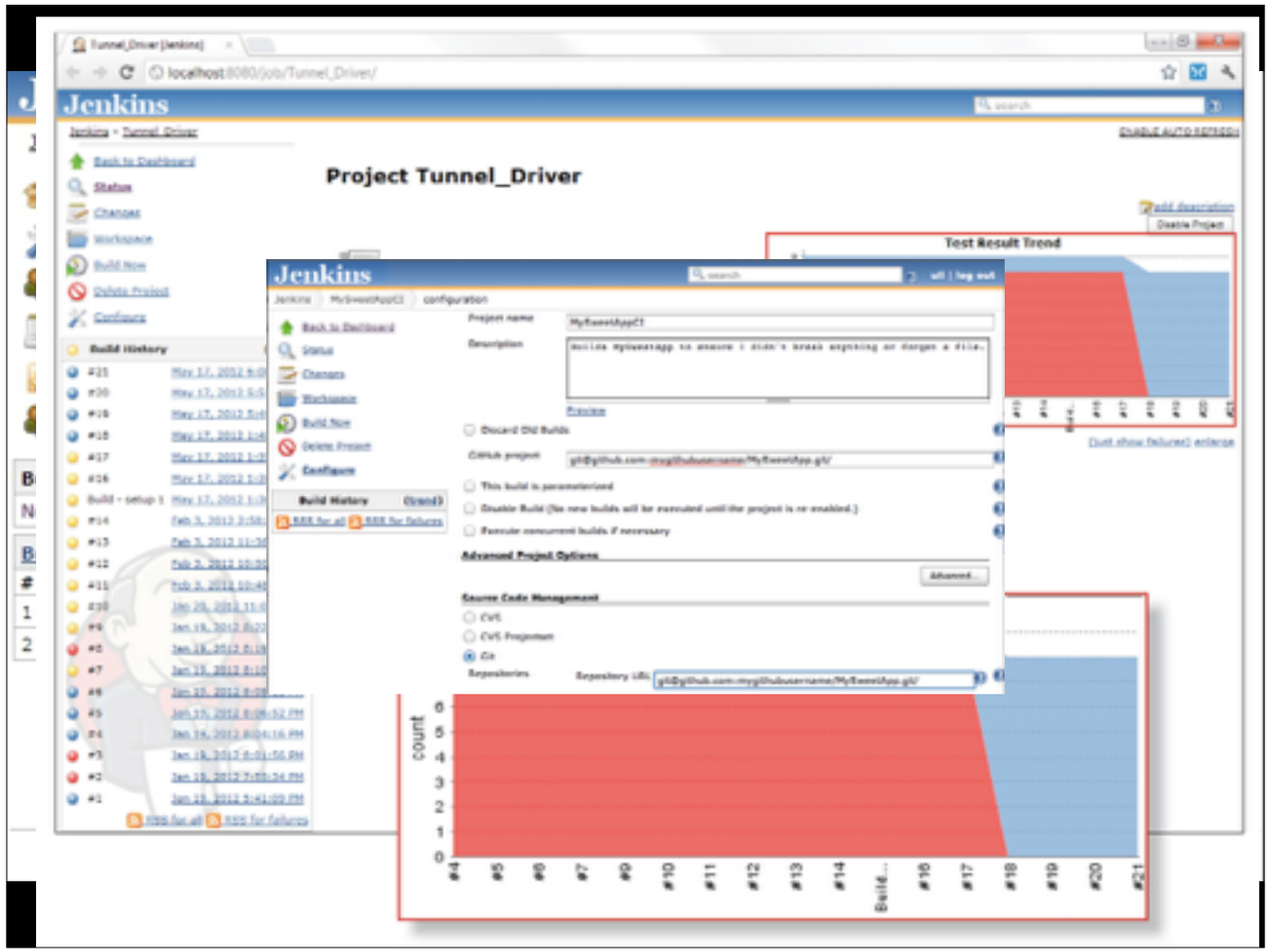
integration - testing when code is combined with other code…dependencies (functions, classes, apis, services, systems)

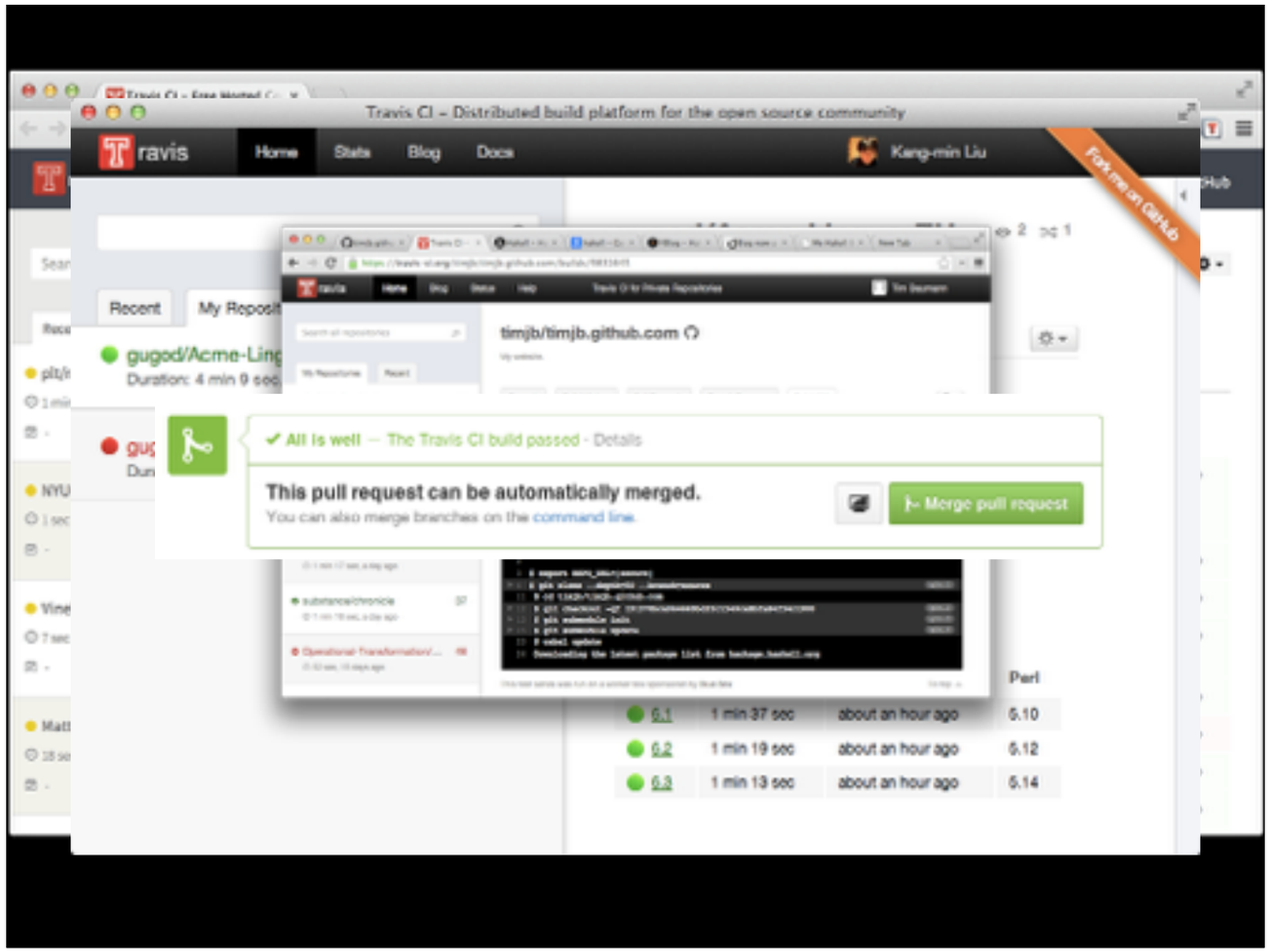
acceptance - success criteria met



Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

People should understand continuous integration, and have used it. They don't necessarily have to use it every single time with every single project, but they should understand the value of having and why they would or wouldn't want to have it. On large projects (like the Software Engineering class at the end of the curriculum) I would expect they use it, or have real good reasons why they don't.

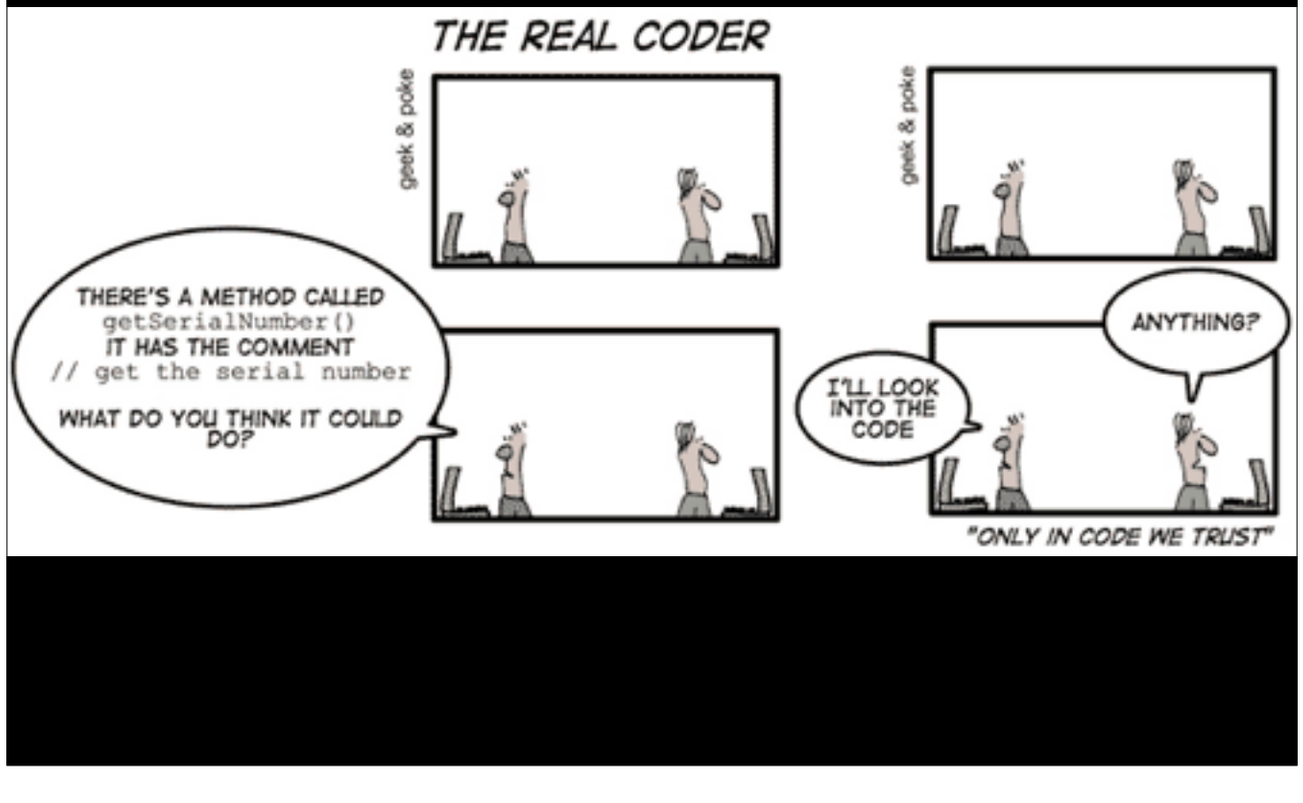

Jenkins
Project Tunnel_Driver
Test Result Trend
count
Build...



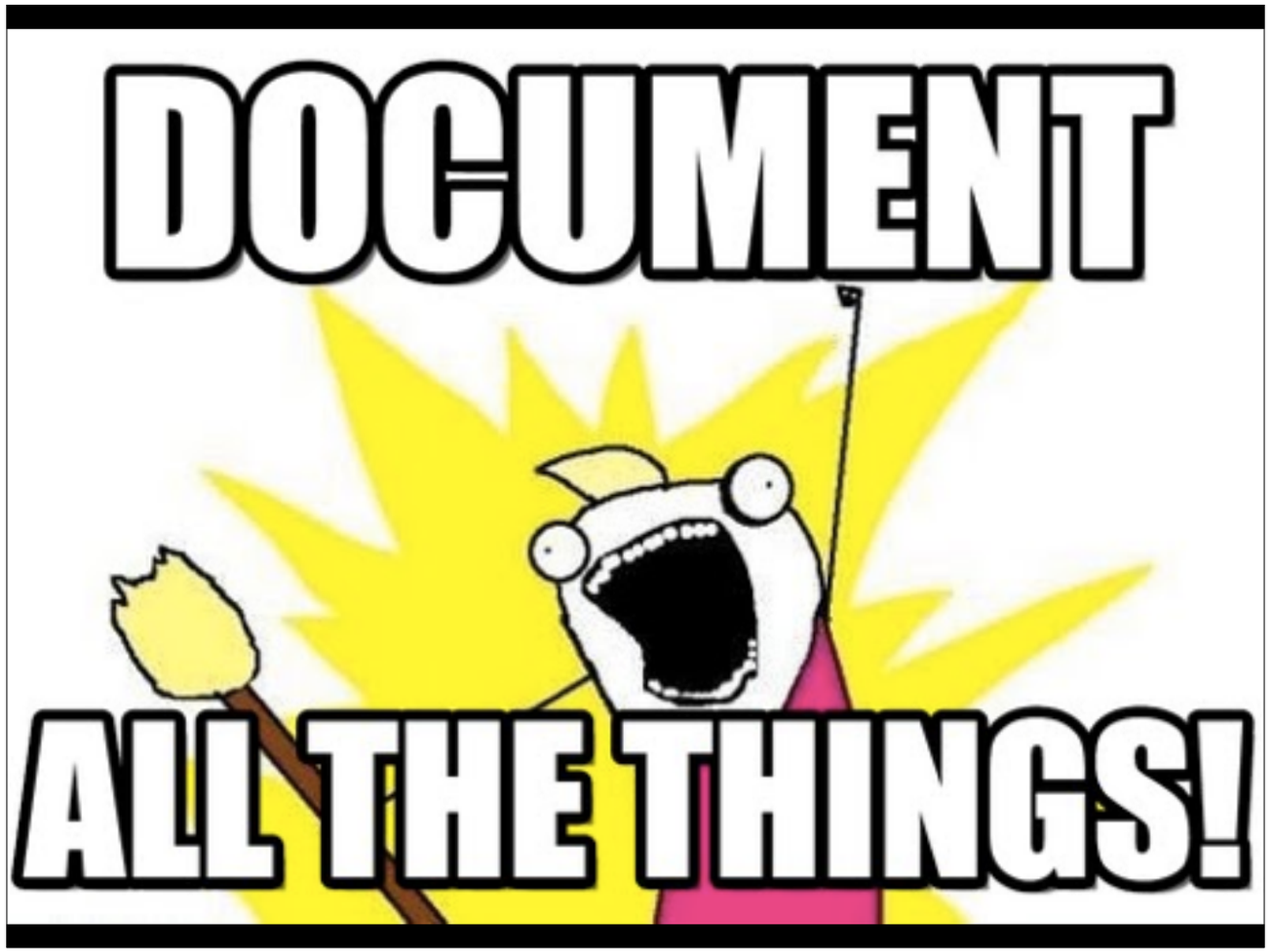
Free

Github integration

Easy to setup



People that can not only write code, but document the code and thoughts well (getFullName() /* gets the full name of person */ is not good documentation). Some of the best projects that I consume are projects with excellent documentation.

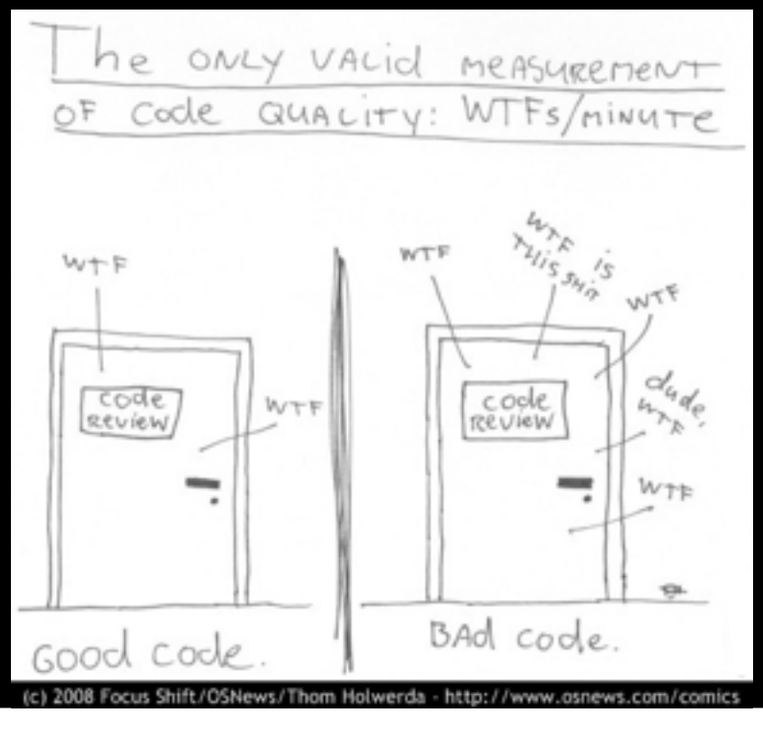


Not just javadoc, or code comments, but guides, tutorials, quickstarts, readmes, etc... my team strives to produce good meaningful documentation, and it's not just doc'ing up our code, but that we have a core value to document everything so that it doesn't have to repeated by others. So we try to have are decisions documented, and the reasoning behind it. We document and summarize meetings. We document design proposals and acceptance. We document retrospective. We believe that everything should be documented for yourself (future self) and others so decisions are transparent, and also it allows others to learn why.

- http://guides.rubyonrails.org/
- https://docs.djangoproject.com
- https://github.com/mbostock/d3/wiki
- http://docs.writethedocs.org/writing/beginners-guide-to-docs/

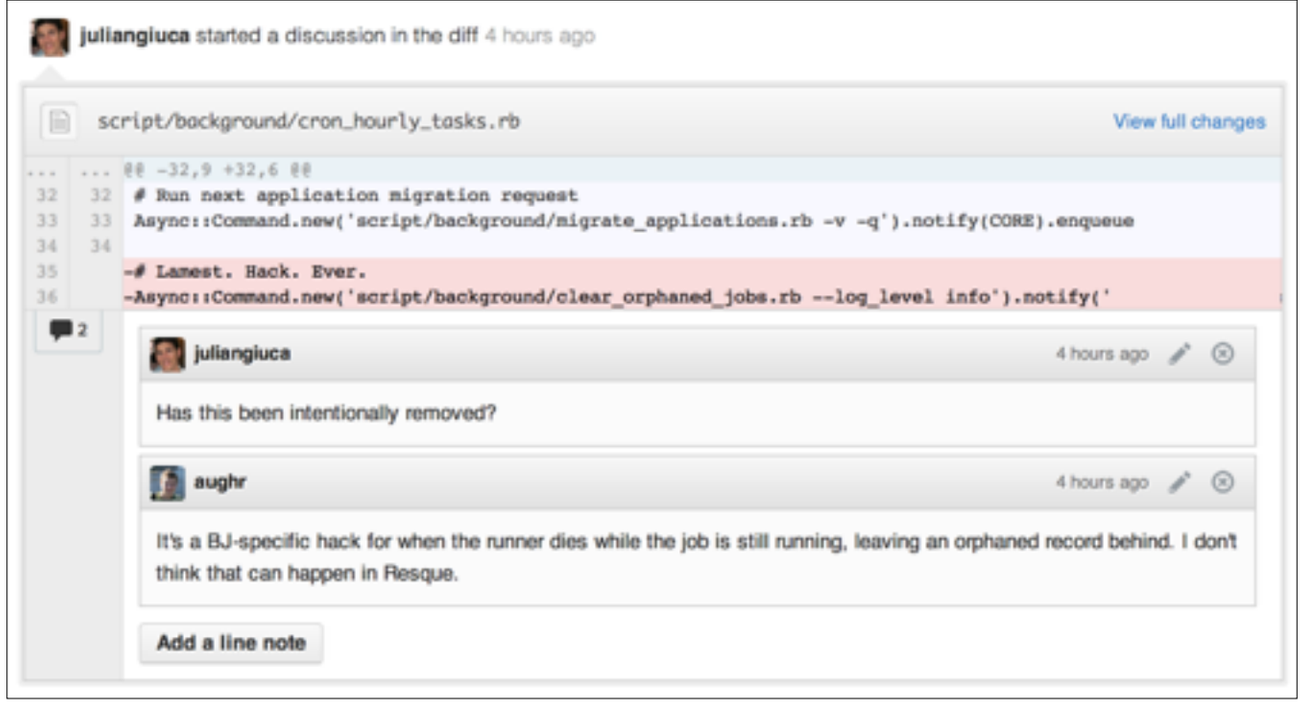
Examples

CODE REVIEWS



On my team, we have a policy that everyone reviews everyone's code and have to give consent that they're complete. We use it not only as a means of quality control, but also as a knowledge sharing mechanic. It allows the author to explain them self to everyone else the implementation decisions they made and why, and it allows everyone else to understand the code that the team is releasing and owning, which implicitly means they own.

GITHUB PULL REQUESTS



time line

inline

comments

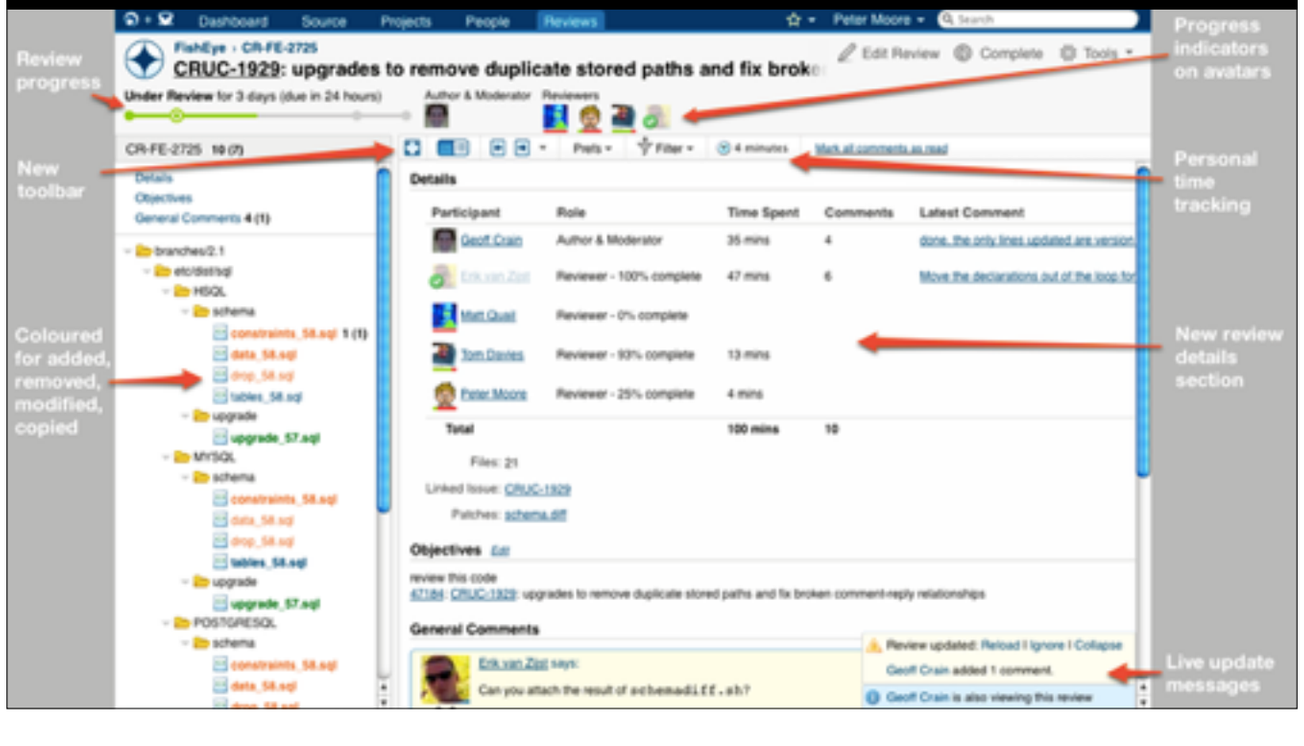
diff

files changes

all commits

notifications

CRUCIBLE



jira integration

completion status

due dates

git integration

list of reviewers

TROUBLESHOOTING

Students struggle with this at a system level. I think that most students don't have the exposure to a large system as a part of any assignments, and so when they have to debug, they can get away with simple print statements and walk through the code.

- logs
- linux
- bash, vim, grep, awk, sed
- debugger
- reproduce

- what changed last?
- google
- 1 hour rule
- logs, logs, logs

However, learning to use a debugger, navigate linux, leverage apache access logs, system logs, or recording requests for playback are things that they may not ever really need in order to debug their personal assignment or project. Those are skills that not only should they have experience with, but that at the very least they should be aware to consider those avenues. We have a policy on my team where people are expected to struggle on their own for 1 hour. If after an hour, they are still truly stuck, then others will help. However, what will usually happen is within that hour, they will utilize their troubleshooting skills and discover something in logs or debugging through the code or running through tests that will lead them to the next clue and the next clue to where they tend to figure it out for themselves. Obviously, google also greatly helps.

RESTFUL APIS

RESTFUL APIS

- Representational State Transfer
- HTTP
  - Methods - GET, POST, PUT, DELETE - Verbs
  - Status Codes
  - Headers
  - Media Type
- URIs - Resources - Nouns
- How I Explained REST to My Wife

Covered well in your Feb 17th Arch & Design 2

RESTful API - representational state transfer

http - GET, PUT, POST, DELETE

URIs, resources

stateless

cacheable

passive - semver

Resource	**POST** create	**GET** read	**PUT** update	**DELETE** delete
/dogs	create a new dog	list dogs	**bulk update dogs**	delete all dogs
/dogs/1234	**error**	show Bo	if exists update Bo **if not error**	delete Bo

- https://developer.github.com/v3/
- https://dev.twitter.com/rest/public
- https://www.yelp.com/developers/documentation
- https://developers.soundcloud.com/docs/api/guide


POSTMAN
Supercharge your API workflow
Build, test, and document your APIs faster. More than a million developers already do.

One thing I see new and experience engineers suffer is it's hard for them to adopt new technologies without bring along the habits of the old technology. I've seen java like code written in ruby, OOP concepts in functional programming, and synchronous code in async (promise, callback) frameworks. This is not inherently bad, but it increases barrier for standing on the shoulders of the community, which I think is not a productive endeavor.

Participate in the community

COMMUNITY

- Open Source
- Blogs
- Twitter
- Idioms & Code Conventions
- Conferences
- Meetups

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Ruby Heroes

I'D LIKE TO NOMINATE

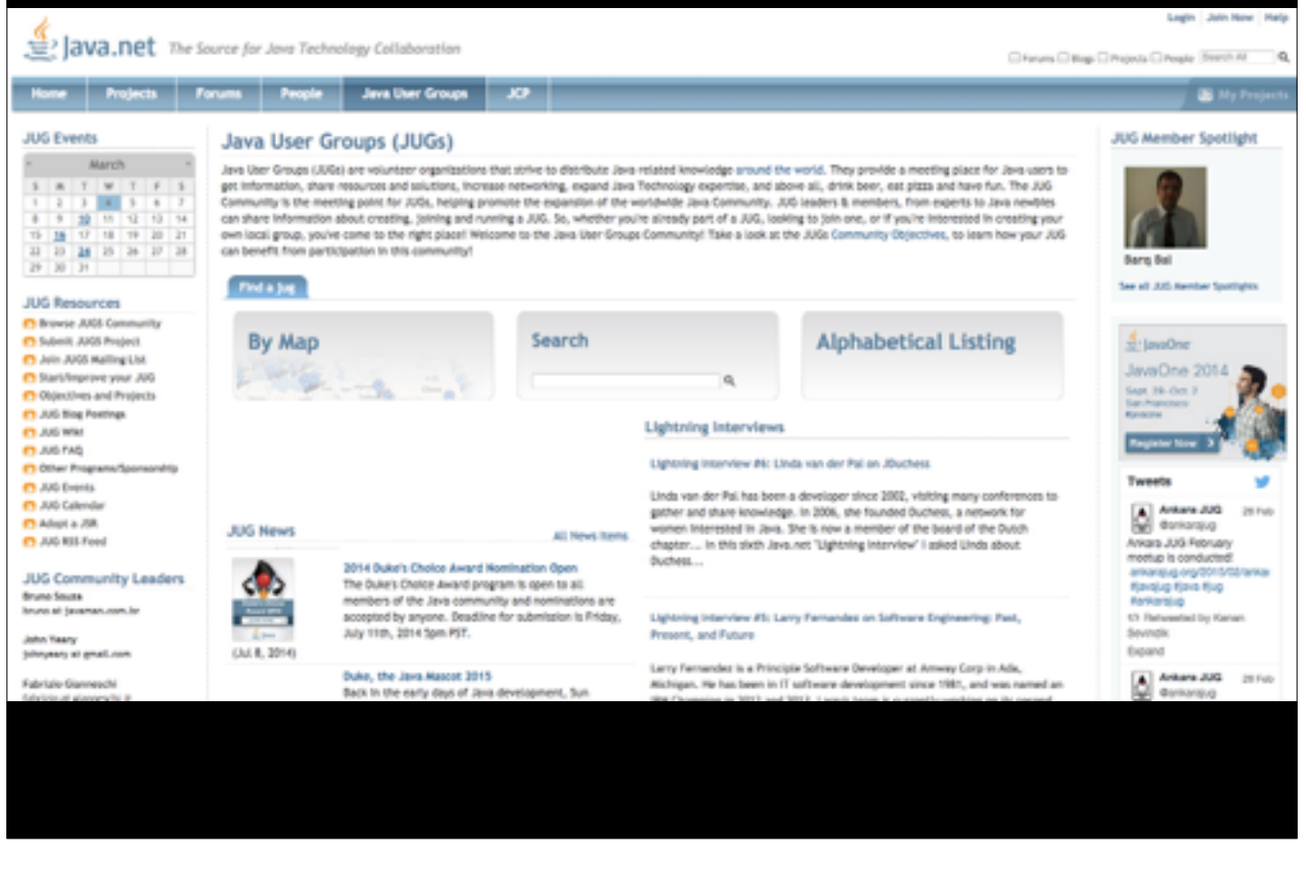
The *Ruby Hero Award* gives recognition to influential devs in the Ruby/Rails community and announces the winners at *RailsConf*.

We believe great work deserves recognition. The Ruby/Rails dev community is full of amazing contributors and unsung heroes who are busy producing educational content, developing plugins and gems, contributing to open-source projects, or putting on events to help developers learn and grow. Wherever they are, their work is too great to go unnoticed and we need your help to find them.

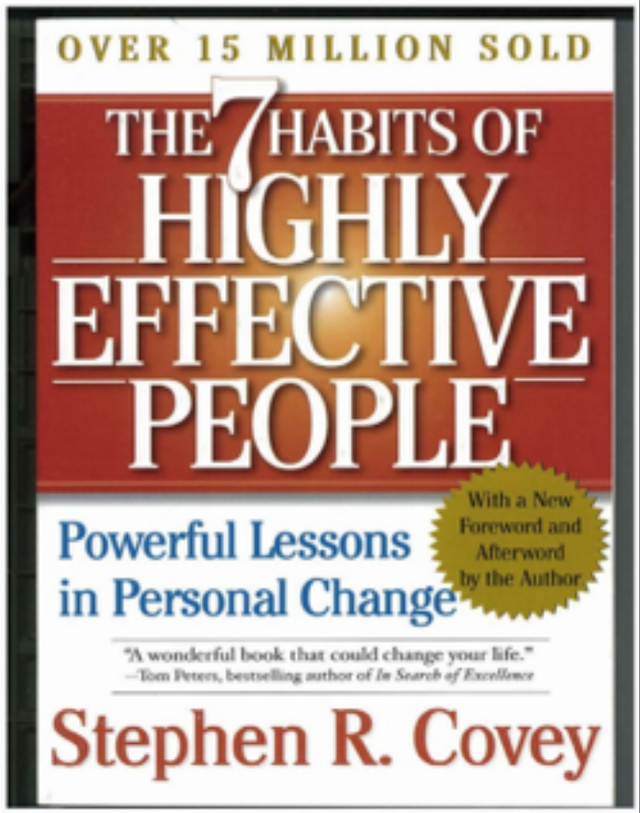
Submit your vote for the Ruby Hero Award above. Six will be chosen and pulled out from behind their code to receive an award at RailsConf.

Instructions

- **Nominate a worthy candidate**
- **Share link to get others voting**
- **Nominees get tallied**
- ***Previous Ruby Heroes* choose finalists**
- **Winners announced at *RailsConf***


Java.net
The Source for Java Technology Collaboration
Login | Join Now | Help
Home
Projects
Forums
People
Java User Groups
JCP
My Projects
JUG Events
March
JUG Resources
Browse JUGS Community
Submit JUGS Project
Join JUGS Mailing List
Start/Improve your JUG
Objectives and Projects
JUG Blog Postings
JUG Wiki
JUG FAQ
Other Programs/Sponsorship
JUG Events
JUG Calendar
Adopt a JSR
JUG RSS Feed
JUG Community Leaders
Bruno Souza
bruno at javaman.com.br
John Yeary
johnyeary at gmail.com
Fabrizio Giannecchi
Java User Groups (JUGs)
Java User Groups (JUGs) are volunteer organizations that strive to distribute Java related knowledge around the world. They provide a meeting place for Java users to get information, share resources and solutions, increase networking, expand Java Technology expertise, and above all, drink beer, eat pizza and have fun. The JUG Community is the meeting point for JUGs, helping promote the expansion of the worldwide Java Community. JUG leaders & members, from experts to Java newbies can share information about creating, joining and running a JUG. So, whether you're already part of a JUG, looking to join one, or if you're interested in creating your own local group, you've come to the right place! Welcome to the Java User Groups Community! Take a look at the JUGs Community Objectives, to learn how your JUG can benefit from participation in this community!
Find a Jug
By Map
Search
Alphabetical Listing
JUG News
All News Items
2014 Duke's Choice Award Nomination Open
The Duke's Choice Award program is open to all members of the Java community and nominations are accepted by anyone. Deadline for submission is Friday, July 11th, 2014 5pm PST.
(Jul 8, 2014)
Duke, the Java Mascot 2015
Lightning Interviews
Lightning Interview #4: Linda van der Pal on JBuchess
Linda van der Pal has been a developer since 2002, visiting many conferences to gather and share knowledge. In 2006, she founded Duchess, a network for women interested in Java. She is now a member of the board of the Dutch chapter... In this sixth Java.net "Lightning Interview" I asked Linda about Duchess...
Lightning Interview #3: Larry Fernandez on Software Engineering: Past, Present, and Future
Larry Fernandez is a Principle Software Developer at Amway Corp in Ada, Michigan. He has been in IT software development since 1987, and was named an
JUG Member Spotlight
Barış Bal
See all JUG Member Spotlights
JavaOne
JavaOne 2014
Register Now
Tweets
Ankara JUG
@ankarajug
Ankara JUG February meetup is conducted!
ankarajug.org/2015/02/anka
#javajug #java #jug
#ankarajug
Retweeted by Kenan Sevindik
Expand
Ankara JUG
@ankarajug

SHARPENING
THE SAW



The 7 Habits of Highly Effective People - Stephen Covey

Habit #7

Lumberjack working hard but going nowhere, saw is dull, too busy sawing to sharpen the saw.

"**Sharpening the saw** is shorthand for anything you do that isn't programming, necessarily, but (theoretically) makes you a better programmer."

–JEFF ATWOOD

- Read Blogs - https://cooperpress.com/
- Read Code - https://github.com/explore
- https://news.ycombinator.com/
- http://www.reddit.com/r/WatchPeopleCode

WHAT WE DIDN'T COVER

~~Testing & CI~~

~~SCM~~

Monitoring

~~Documentation~~

Performance Profiling

~~Code Reviews~~

Resourcefulness

~~Debugging & Troubleshooting~~

Team Work

Automation

Ownership Cost

~~Issue Tracker~~

Globalization

Security

~~Passivity~~

Error Handling

Assessing Code

Fault Tolerance

~~Community~~

~~Sharpening Saw~~

~~REST APIs~~

Web

Mobile

…

Lots of good material in this class. Learn it, but PRACTICE IT.

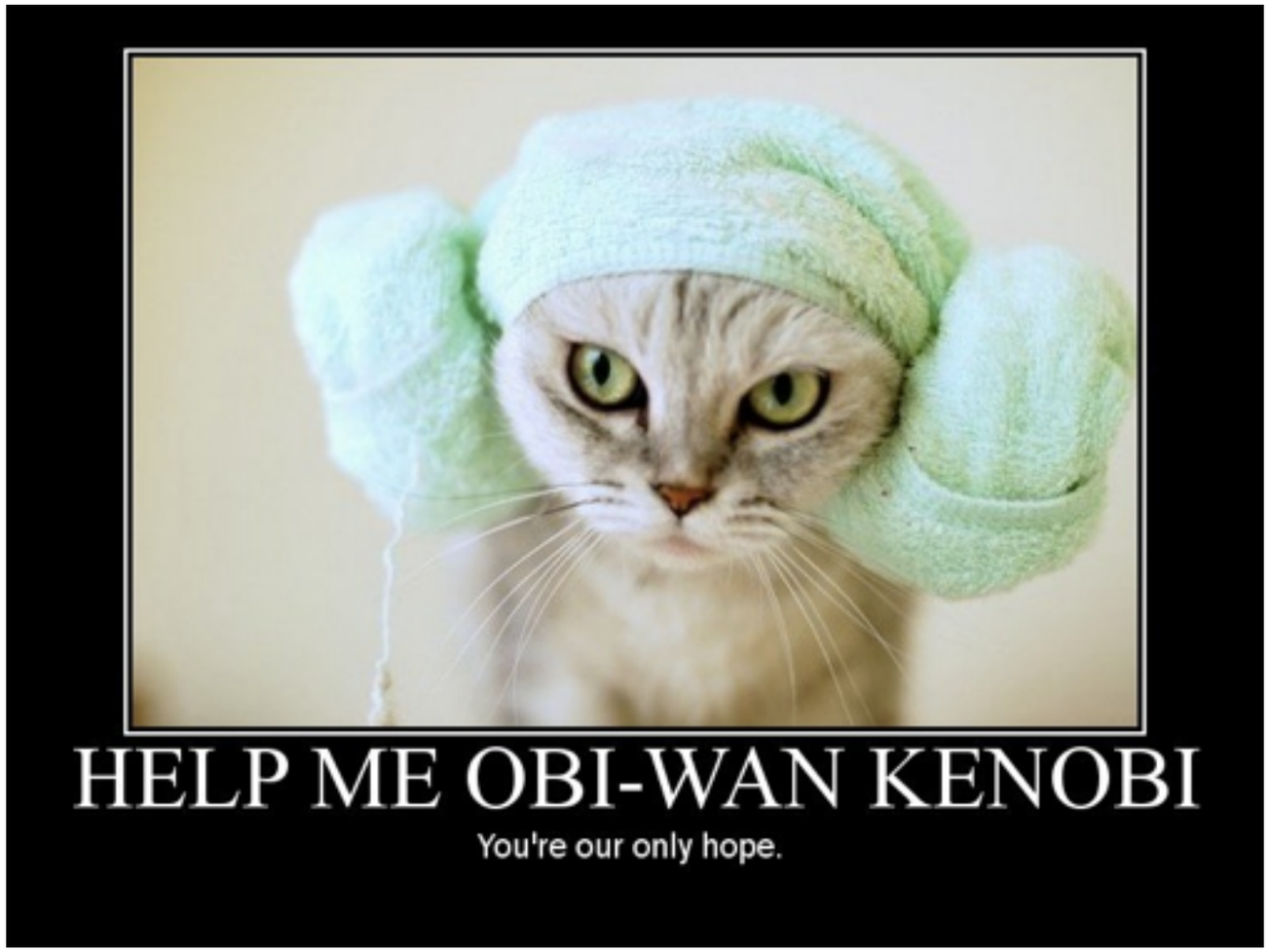
Globalization - I don't know if anyone coming out of school really even considers that their application will not be in English. There's i18n, l10n, bidi, understanding character encodings and character sets, and dealing with timezones
Security - authentication/authorization/auditing are 3 a's that is another application level concern. People should understand the difference between authn and authz. Preferably they've used OAuth, OpenId, but understanding encryption and ciphers. Auditing/Accounting is something more specific to health care (or banking), so maybe that's really essential.
Error handling - Learn proper error handling. When to handle it, when to let it bubble.
Automation - This is a soft skill, but finding candidates that strive for automation is an easy flag for a good engineer. The shows that they have the intrinsic motivation to make things better and never settle for the current things. Whether that's improving their deployment, code review process, or even submitting their timesheet. Seeing that drive, certainly peaks my interests in a candidate.
Ownership cost - This is a concept very few graduates understand because they've been trained from beginning of school career that once their work is done, test is passed, project is submitted, they're done. They don't consider that they should have optimized for the fact that they have to maintain that code base, and enhance it, and refactor it. They just get it to work to submit it and get a grade, and then it's off to the next assignment.
Monitoring - Another system level skill most students don't have exposure to in a class room. But people should know how to monitor the health and

BASICALLY CS490

I went through all the class material posted out there so far. Lots of good stuff. Some stuff I don’t see often, but still good to know what it is (UML, ADL). I don’t see any arch studio like work.

See a lot of REST, MVC, MapReduce, PubSub.

Learn emacs or vim - especially when you need to



Ok, help is on the way

DevAcademy is the new early career software engineer onboarding program designed to deliver customizable and relevant training to engineers that have recently joined Cerner.

3 components to it

What are DevEssentials?

DevEssentials is a component of DevAcademy. It refers to the in-class training portion of the program. This time focuses on the essential things every engineer must know as well as discovering the strengths and areas for improvement the associates have. This is done through a series of assessments, activities and evaluations. The goal is to gather information to ensure the training is maximized for each associate.

What is the DevCenter?

The DevCenter is a componet of DevAcademy. It is an engineering organization dedicated to mentoring early career software engineers on real Cerner projects. While in the DevCenter, new software engineers will be practicing tools, processes and practices. They will be demonstrating their skills on real work. Velocity engineers will also be gaining exposure to the types of projects completed by the hiring teams at Cerner. During the time in the DevCenter, the hiring teams will have the opportunity to interact with the software engineers, gaining visibility to their capabilities.

What are DevElectives?

DevElectives are a component of DevAcademy. It refers to a series of instructor led offerings covering a wide range of topics. The intent of these electives is to provide associates with just-in-time training to address business needs. There will be a rotating schedule of electives, associates will need to complete a certain amount of elective "credits" to complete the program. Managers will work closely with the associates to ensure the training selected is relevant

DEVACADEMY - STRUCTURE

Associate	Level	DevEssentials	DevCenter	DevElectives
Campus Hire	7	Required	Required	Required with Waivers
Returning Interns	7		Required with Waivers	
External Hire	7		Required	
	6		Required with Waivers	
	5	Opt-in		

DEVESSENTIALS

Testing & CI	Team Work	Assessing Code
SCM	Automation	Fault Tolerance
Monitoring	Ownership Cost	Community
Documentation	Issue Tracker	Sharpening Saw
Performance Profiling	Globalization	REST APIs
Code Reviews	Security	Web
Resourcefulness	Passivity	Mobile
Debugging & Troubleshooting	Error Handling	…

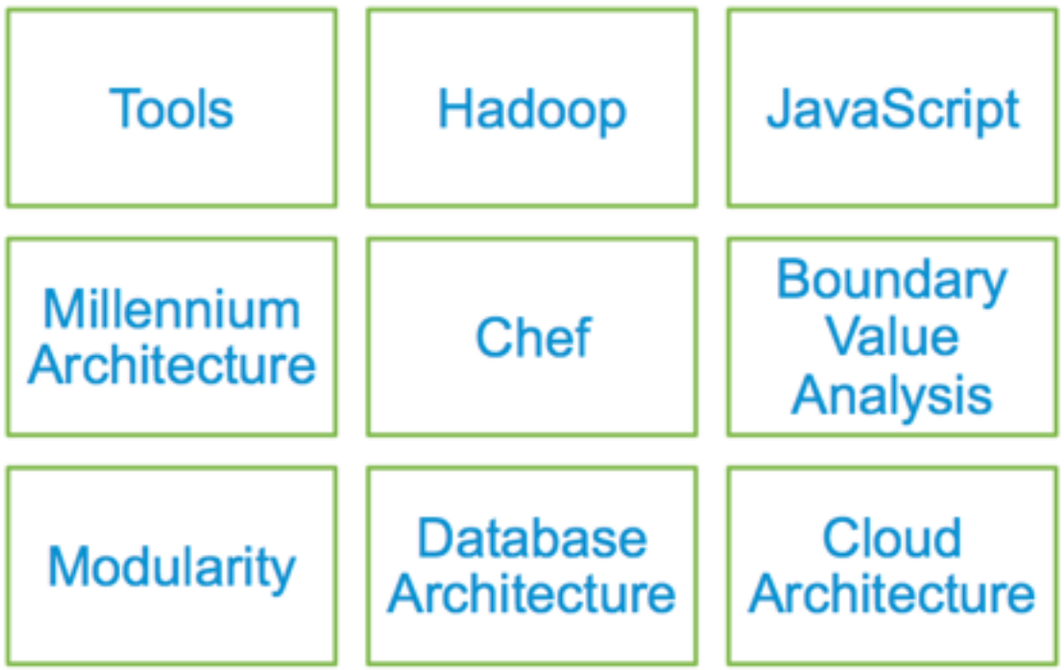
2 weeks

DEVCENTER

- 2-8 weeks
- Agile Development
- Real Projects - Simulated Environment
- Mentorship
- Graduation & Placement

DEVELECTIVES

ADVANCED TOPICS



Cerner Millennium+ 101 Training *5d Jared Fordham, Jim O'Keeffe

Cerner Millennium: Discern Explorer Intermediate * 3d Bob Ross

Cerner: Millennium Technical Troubleshooting* 4d (spread apart into 1d/w) Stephen Sedgwick

Software Globalization (Glocalization) Lab

Java

C#

3h Java: Andy Gifford, Joshua Davis

C#: John Bowen, Matt Strayhall

DevElectives: Boundary Analysis 3hr Dylan Buccieri

DevElectives: Chef 2d Jamie Scheker and Daniel Lloyd

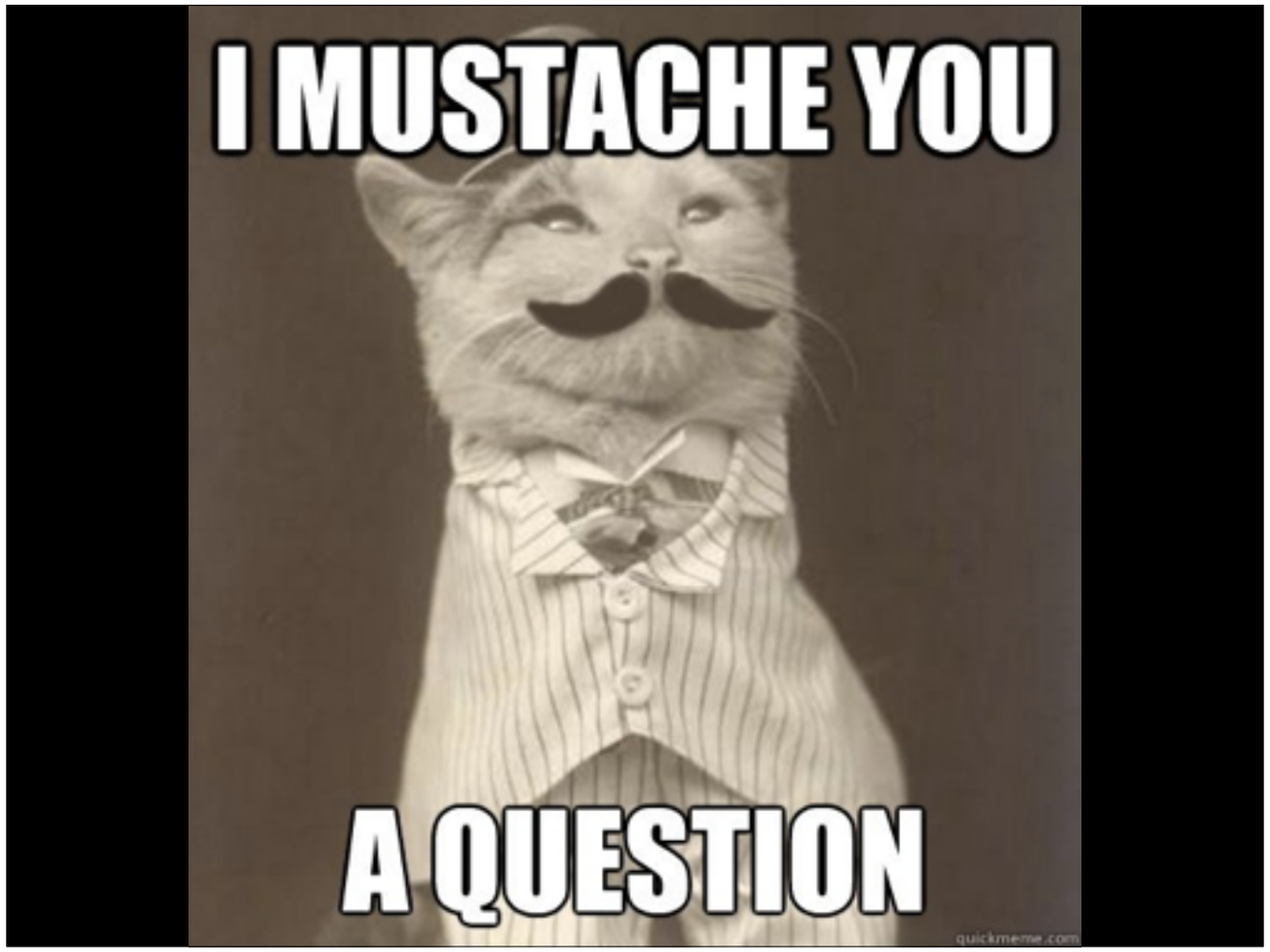
DevElectives: Cloud Platform Overview 3h Kevin Shekleton, Robert Farr, Carl Carl Chesser

DevElectives: Code Cache 2hr John Bennewitz, Brandon Heck

DevElectives: Git 201 3hr Josh Matcuk & Brian Keh

DevElectives: Hadoop Utilities 7hr Christian Duranleau, Ben Roling, Micah Whitacre, Swarnim Kulkarni, Cole Skoviak, Jeffrey Jestes

DevElectives: iBus Consumer Fundamentals 4hr Kris Klindworth and Mike Barnes



any questions?