Testing:

It's not just for your code!

Christopher H. Laco » claco@chrislaco.com » @claco

Follow along! http://chrislaco.com/slides/csc-testing.pdf

Shout out to our sponsors

Robert Half® Technology

roberthalftechnology.com

pluralsight hardcore developer training

pluralsight-training.net

O'REILLY® Spreading the knowledge of innovators.

oreilly.com

Your humble speaker

- Reformed Music Major Turned Nerd
- Hardware/Software/Network
- Large Auto Aftermarket Retailer
- Internet -> Old Old System
- First Order -> Oodles Per Hour
- .NET By Day. Perl By Night. Rails!
- The “Build Server Guy”

Inspiration

- Perl 5.6 - 2000 / CPAN - 1995 / Testers - 1999
- Ruby 1.8 - 2003 / RSpec - 2005 / GemTesters - 2010
- Test code, style, metrics, performance
- Tests uploaded across platforms / versions (at install time!)
- TAP - Test “Anything” Protocol; Perl, PHP, Ruby, C, etc
- Andy Lester (PETDANCE): Test Everything
- Great TeamCity Nightmare of 2011

Why are we here?

- Heard the word “testing”
- Food / Drink
- TDD / BDD / Agile / Red Green Refactor
- RSpec / Test::Unit / Cucumber / Other
- Rewrite! My Tests Still Pass!
- Testing as a process w/ feedback loops
- Software Quality / Craftsmanship

“Good News Everyone!”

We’re not talking about those!

Say what meow?

- Those things are awesome
- Developer centric ideas
- Focused on process
- Forgetting about product
- Product lifecycle is bigger than that
- Fail doesn't discriminate
- Lets talk about the other things

Why do we write tests?

- Measure code quality*
- Freedom to refactor / prevent regressions
- Document software interface
- Confirm expectations / requirements
- Code style / metrics
- Pass on knowledge
- Early warning system

Why not other things?

- Prevent non-software regressions
- Document environment
- Confirm data expectations
- Tests as an early warning system
- Monitor is just a scheduled test
- Manage problems that might not be preventable
- Troubleshoot production problems quicker
- Use familiar tools instead of scattered scripts

Uber Fast

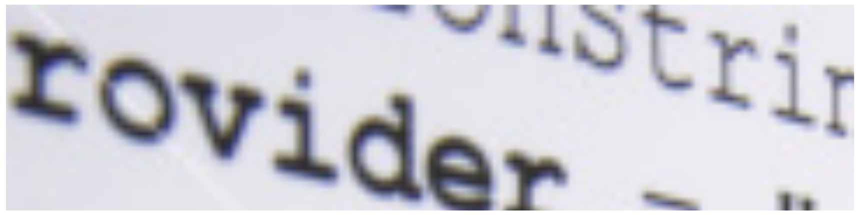
Follow Practices

100% Green

100% Coverage

Cured Cancer

Ended Global Warming



Environment

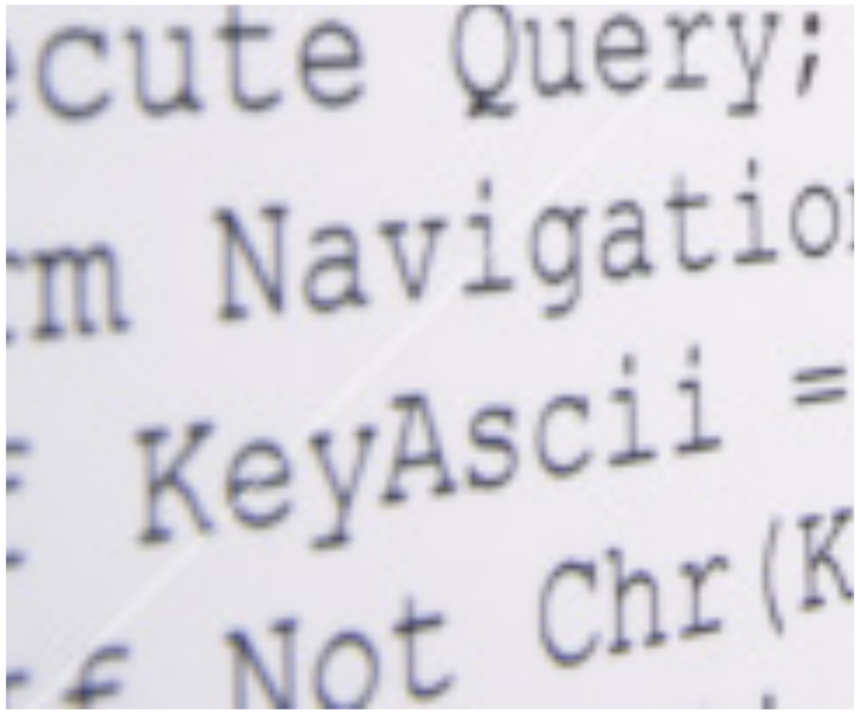
Works on my machine!

Data

Garbage In.
Garbage Out.

Deployment

Command finished successfuly.



Environments gone wrong

- Different Operating System Behavior - “MBP Bubble”
- Missing/Broken Connection Strings - machine.config/database.yml
- Missing/Broken Libraries - LibXML / LibXSLT / ImageMagick
- Missing/Expired License Keys - DLL Registration
- Connectivity Issues / DNS / Config Files
    - Local Firewall / DMZ / Internet API Access
- Email Delivery - Config/Filter/DNS Changes
- Server Upgrades - Managed Host Security Upgrades

From the trenches...

```
require 'YAML'
```

```
claco@mbp ~ $ irb
irb(main):001:0> require 'YAML'
 => true
```

```
claco@builds ~ $ irb
irb(main):001:0> require 'YAML'
LoadError: no such file to load -- YAML
	from (irb):1:in `require'
	from (irb):1
```

From the trenches...

```
Dir['lib/**/*.rb'].each {|f| require f}


touch a.rb; touch d.rb
mkdir b; touch b/c.rb


claco@mbp ~ $ irb
irb(main):001:0> Dir['**/*.rb']
 => ["a.rb", "b/c.rb", "d.rb"]

claco@builds ~ $ irb
irb(main):001:0> Dir['**/*.rb']
 => ["d.rb", "b/c.rb", "a.rb"]
```

From the trenches...

```
uninitialized constant A::BC (NameError)
```

```
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in
`gem_original_require'
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in `require'
from /usr/lib/ruby/gems/1.8/gems/activesupport-2.3.8/lib/active_support.rb:57
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in
`gem_original_require'
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in `require'
from /usr/lib/ruby/gems/1.8/gems/rails-2.3.8/lib/rails_generator.rb:31
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in
`gem_original_require'
from /usr/local/lib/site_ruby/1.8/rubygems/custom_require.rb:36:in `require'
from /usr/lib/ruby/gems/1.8/gems/rails-2.3.8/bin/rails:15
from /usr/bin/rails:19:in `load'
from /usr/bin/rails:19
```

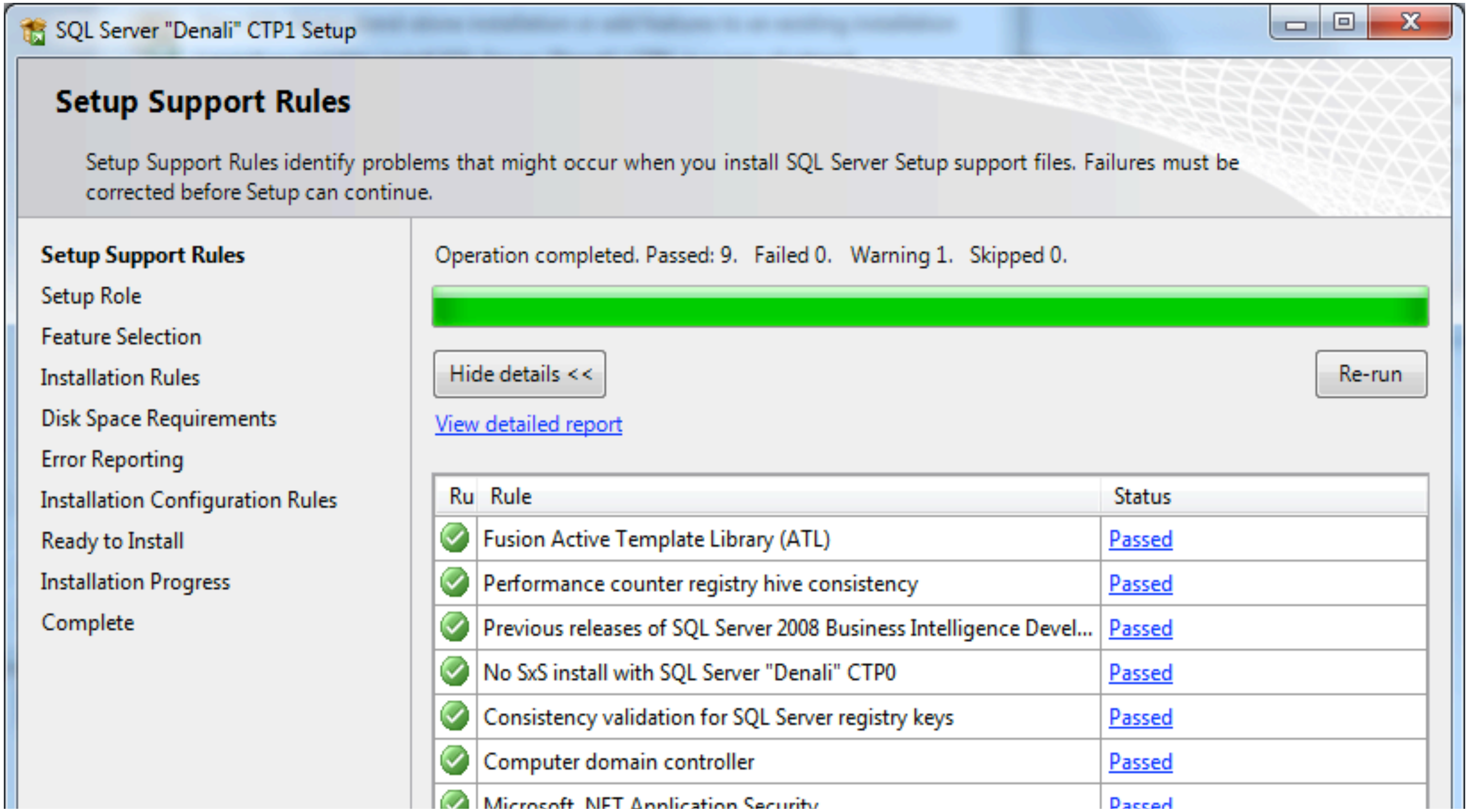
Test your environment!

- Run tests on different OS. Maybe two!
- Write a test connect to the database / internet
  - `rake tests:db:connect`
- Write a test to connect to external API
  - `rake tests:twitter:connect`
- Write a test to confirm emails get delivered
  - `rake tests:email:send_confirmation`

Use tests for troubleshooting!

- Installed a new server. Configured correctly?
  - Chef/Puppet Recipe downloads/runs tests
- Installed a new Load Balancer. Still connect to DB?
  - Run connection tests on all servers
- Reconfigured Postfix Filters. Mailers still work?
  - Run mailer tests on app servers
- God/Relic/Nagios/Client says the site is down!
  - Run various tests to verify environment / pinpoint issues

Examples in the wild



SQL Server "Denali" CTP1 Setup

Setup Support Rules

Setup Support Rules identify problems that might occur when you install SQL Server Setup support files. Failures must be corrected before Setup can continue.

Setup Support Rules
Setup Role
Feature Selection
Installation Rules
Disk Space Requirements
Error Reporting
Installation Configuration Rules
Ready to Install
Installation Progress
Complete

Operation completed. Passed: 9. Failed 0. Warning 1. Skipped 0.

Ru	Rule	Status
✓	Fusion Active Template Library (ATL)	Passed
✓	Performance counter registry hive consistency	Passed
✓	Previous releases of SQL Server 2008 Business Intelligence Devel...	Passed
✓	No SxS install with SQL Server "Denali" CTP0	Passed
✓	Consistency validation for SQL Server registry keys	Passed
✓	Computer domain controller	Passed
✓	Microsoft .NET Application Security	Passed

Examples in the wild

```
Scenario: Checking if apache is running
    When I ssh to "localhost" with the following credentials:
     | username | password  |
     | vagrant  | vagrant |
    And I run "ps -ef |grep http|grep -v grep"
    Then I should see "http" in the output

Scenario: Surf to apache
Given I go to "http://localhost:9000"
Then I should see "It works"
```

Examples in the wild

```
Starting Chef Run for new-server-node
Synchronizing cookbooks
Loading cookbooks [apache, mysql, server-tests]
Processing apache action install
Processing mysql action install
Processing server-tests action execute
```

```
$ knife ssh “node:app01” “sudo server-tests”
```

Data gone wrong

- Legacy Business System -> New Shiny WebApp
- Little / No / Different Constraints
- Bad data import into “good” schema or strict constraints
- Table of file names. Do they exist?
- “Required” fields which are still NULL / invalid format (Email)
- Complex Data Logic (When Does Widget X Deliver?)
    - Inventory + Qty + Date + Time + Weight + Zip + POM == Est. Delivery Date

From the trenches...

```
CREATE TABLE Legacy_DB_Widgets
(
  Id int NOT NULL,
  Name varchar(255) NOT NULL
)


CREATE TABLE Greenfield_Widgets_View
(
  Id int NOT NULL,
  Name varchar(255) NOT NULL,
  UNIQUE (Name)
)


SELECT ...
INTO Greenfield_Widgets_View
FROM Legacy_DB_Widgets
INNER JOIN (...5 other tables!)
```

```
Duplicate entry 'WidgetName' for key 4
```

From the trenches...

```
Error 0xc0047038: Data Flow Task: SSIS Error Code DTS_E_PRIMEOUTPUTFAILED.  The PrimeOutput method on component "Source 2 - name$" (128) returned error code 0xC02020C4.  The component returned a failure code when the pipeline engine called PrimeOutput(). The meaning of the failure code is defined by the component, but the error is fatal and the pipeline stopped executing.  There may be error messages posted before this with more information about the failure.
```

(SQL Server Import and Export Wizard)

From the trenches...

```
CREATE TABLE Attachments
(
  Id int NOT NULL,
  User int NOT NULL,
  FileName varchar(255) NOT NULL
)



Id     User       FileName
 1        1       derp-resume.pdf
```

http://example.com/users/derp/attachments/1

```
RESOURCE NOT FOUND
```

From the trenches...

```
user = User.new( :email => ‘bogons’ )
  -> ArgumentError “Email address is crap!”


def register
  user = User.new( :email => valid_form_data.email )
  user.save
end

def login
  if legacy.authenticate(user, pass)
    user = User.new( :email => legacy.email ) # “bogons”
  end
end
```

“Customer Derp called. They can’t login! Help!”

Test your data!

- Write a test to detect duplicate data
  - `rake tests:widgets:dups`
- Write a test to detect invalid data
  - `rake tests:emails:format`
- Write a test to detect if files exist
  - `rake tests:attachments:exist`
- Write a test to analyze data against logic
  - `rake tests:delivery:estimate widget=ABC1234`

Use tests for troubleshooting!

- Complex Import failed? Run the data tests.
- Image is missing! Run the file tests.
- Why doesn't X page show Y?. Run the logic tests.
- Run various tests to verify environment.
- Quick identification of problems.

Deployment gone wrong

- Tests Passed on the Build Server. So what?
- Deploy Without Errors. Does the site work?
- Cucumber tests confirm functionality
  - ...for Stakeholders...not Users
  - Not usually reusable
- QA can’t possibly test 100% on every release
- Customers are the first to find issues

From The Trenches...

```
yum install imagemagick
```

```
bundle install --deployment
```

```
claco@builds ~ $ bundle exec rails console
irb(main):001:0> require ‘rmagick’
LoadError: ImageMagick 6.0.so not found....
```

```
echo "/usr/lib" >> /etc/ld.so.conf.d/local.conf
ldconfig
```

Examples in the wild

```
$ cpan install App::Ack

  Running make test
  t/ack-binary.t ............. ok
  t/ack-ignore-dir.t ......... ok
  ...
  All tests successful.

  Running make install
  Installing /usr/local/bin/ack
```

Examples in the wild

```
$ cap deploy

  * executing `deploy'
  * executing `deploy:tests:environment'
  * executing `deploy:update_code'
  * executing `deploy:tests:specs'
  * executing `deploy:symlink'
```

Test Everything!

- Find a bug in code? Write a test.
- Find a bug in the deploy process? Write a test.
- Find a bug in the environment? Write a test.
- Find a bug in external data? Write a test.
- Researched same problem multiple times? Write a test. Pass it on.
- Tests are your todo list of things that can go wrong.
- Tests are your troubleshooting checklist when things go wrong.
- Tests make troubleshooting quicker and more accessible.

Thanks for participating!

Slides:	http://chrislaco.com/slides/csc-testing.pdf
Email:	claco@chrislaco.com
Twitter:	@claco