Continuous Integration

jingmi@gmail.com

mi.jing@jiepang.com

2012-07-02

Reference

- **Continuous Integration:** http://martinfowler.com/articles/continuousIntegration.html

Definition(1)

- The term 'Continuous Integration' originated with the Extreme Programming development process.

Definition(2)

- The practice of frequently integrating one's new or changed code with the existing code repository – should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.

Building a Feature with Continuous Integration(1)

- Clone codes to local machine
- Automated build
- All builds and tests without errors
- If other changes clash with your changes
- Commit to mainline
- Integration build

Building a Feature with Continuous Integration(2)

- A good team should have many correct builds a day.

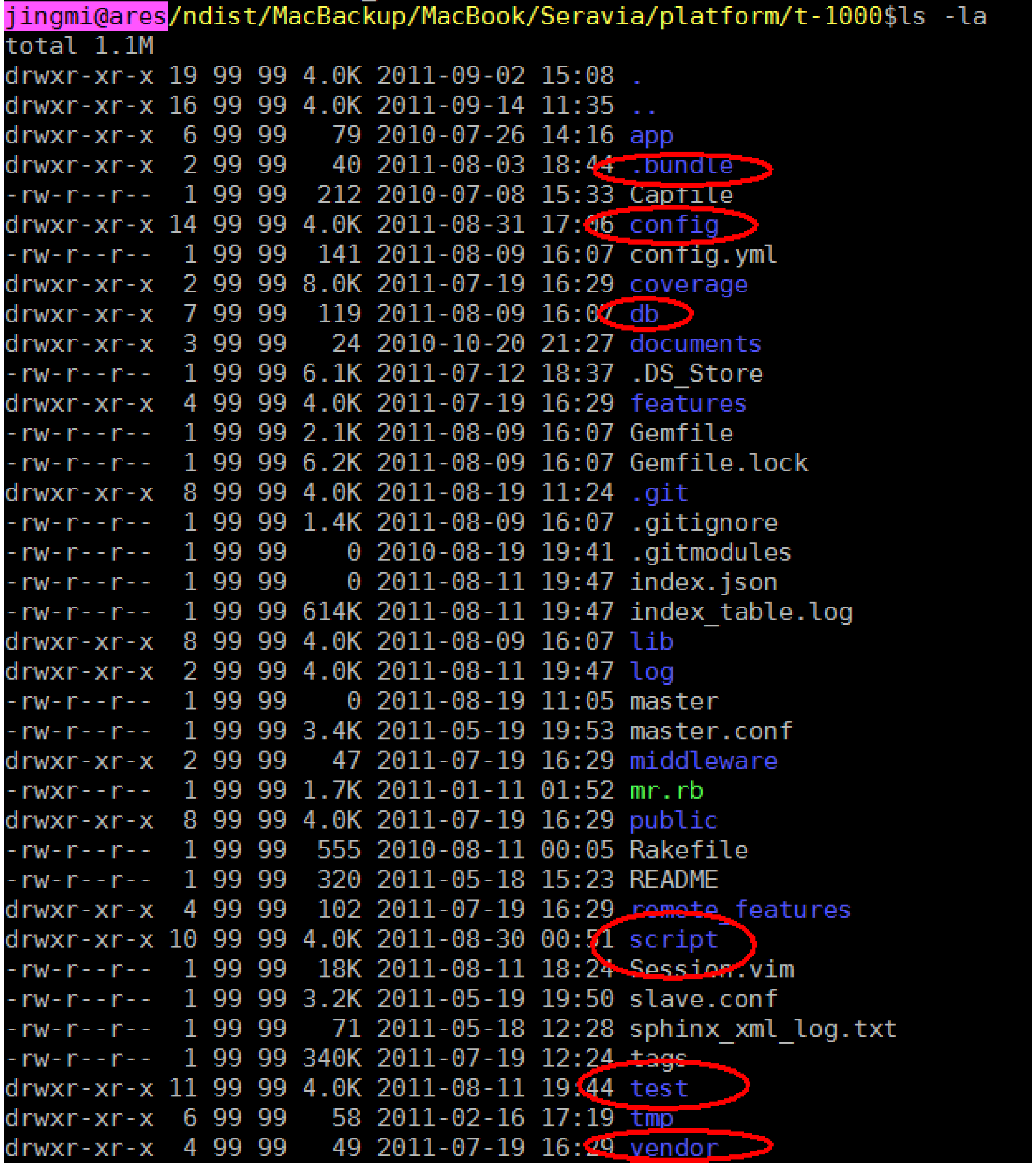
Practices of Continuous Integration(1)

- Repository

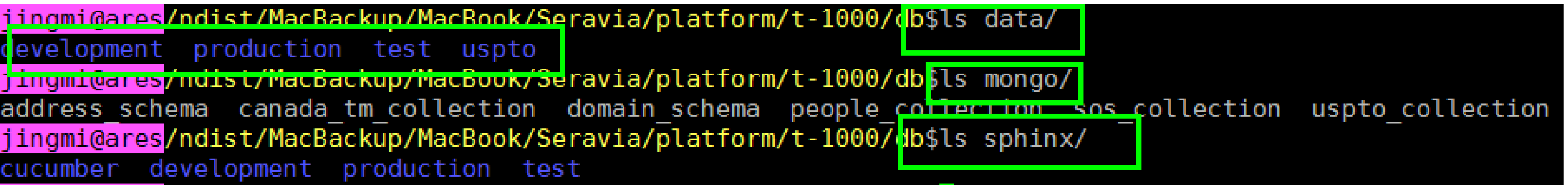
  Maintain a Single Source Repository

- Included:

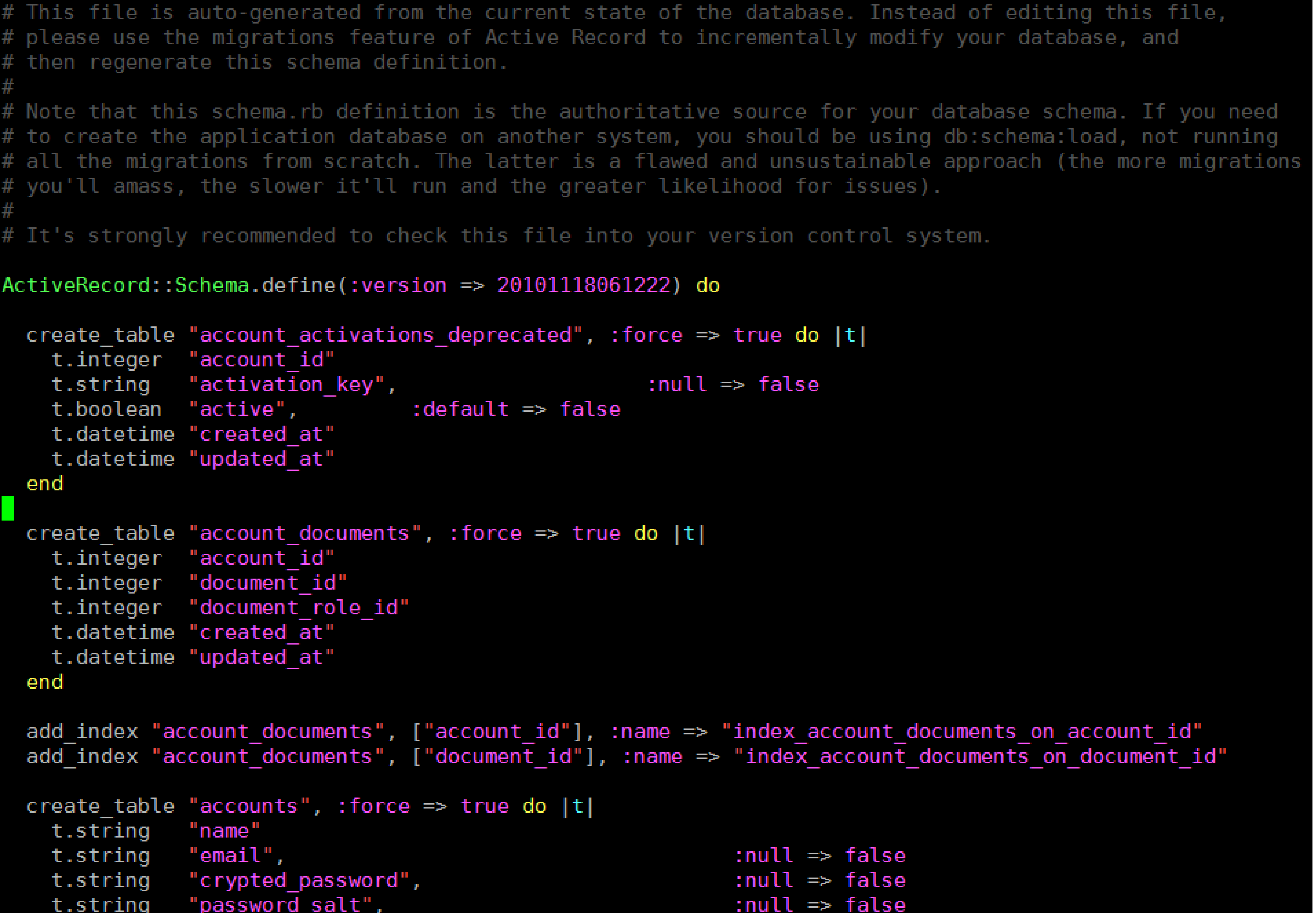
  test scripts, properties files, database schema, install scripts, and third party libraries



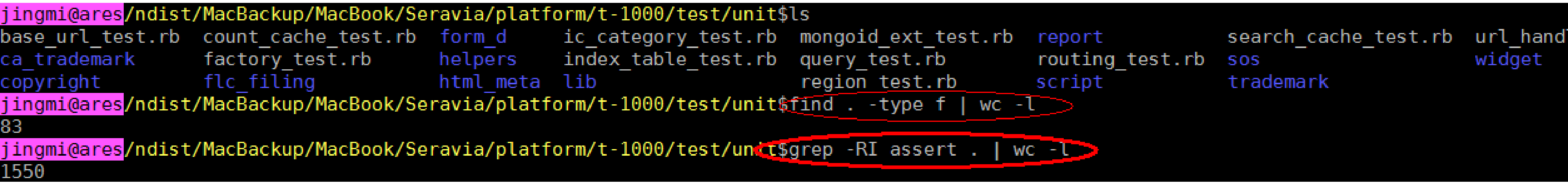
```
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000$ls -la
total 1.1M
drwxr-xr-x 19 99 99 4.0K 2011-09-02 15:08 .
drwxr-xr-x 16 99 99 4.0K 2011-09-14 11:35 ..
drwxr-xr-x  6 99 99   79 2010-07-26 14:16 app
drwxr-xr-x  2 99 99   40 2011-08-03 18:44 .bundle
-rw-r--r--  1 99 99  212 2010-07-08 15:33 Capfile
drwxr-xr-x 14 99 99 4.0K 2011-08-31 17:06 config
-rw-r--r--  1 99 99  141 2011-08-09 16:07 config.yml
drwxr-xr-x  2 99 99 8.0K 2011-07-19 16:29 coverage
drwxr-xr-x  7 99 99  119 2011-08-09 16:07 db
drwxr-xr-x  3 99 99   24 2010-10-20 21:27 documents
-rw-r--r--  1 99 99 6.1K 2011-07-12 18:37 .DS_Store
drwxr-xr-x  4 99 99 4.0K 2011-07-19 16:29 features
-rw-r--r--  1 99 99 2.1K 2011-08-09 16:07 Gemfile
-rw-r--r--  1 99 99 6.2K 2011-08-09 16:07 Gemfile.lock
drwxr-xr-x  8 99 99 4.0K 2011-08-19 11:24 .git
-rw-r--r--  1 99 99 1.4K 2011-08-09 16:07 .gitignore
-rw-r--r--  1 99 99    0 2010-08-19 19:41 .gitmodules
-rw-r--r--  1 99 99    0 2011-08-11 19:47 index.json
-rw-r--r--  1 99 99 614K 2011-08-11 19:47 index_table.log
drwxr-xr-x  8 99 99 4.0K 2011-08-09 16:07 lib
drwxr-xr-x  2 99 99 4.0K 2011-08-11 19:47 log
-rw-r--r--  1 99 99    0 2011-08-19 11:05 master
-rw-r--r--  1 99 99 3.4K 2011-05-19 19:53 master.conf
drwxr-xr-x  2 99 99   47 2011-07-19 16:29 middleware
-rwxr--r--  1 99 99 1.7K 2011-01-11 01:52 mr.rb
drwxr-xr-x  8 99 99 4.0K 2011-07-19 16:29 public
-rw-r--r--  1 99 99  555 2010-08-11 00:05 Rakefile
-rw-r--r--  1 99 99  320 2011-05-18 15:23 README
drwxr-xr-x  4 99 99  102 2011-07-19 16:29 remote_features
drwxr-xr-x 10 99 99 4.0K 2011-08-30 00:51 script
-rw-r--r--  1 99 99  18K 2011-08-11 18:24 Session.vim
-rw-r--r--  1 99 99 3.2K 2011-05-19 19:50 slave.conf
-rw-r--r--  1 99 99   71 2011-05-18 12:28 sphinx_xml_log.txt
-rw-r--r--  1 99 99 340K 2011-07-19 12:24 tags
drwxr-xr-x 11 99 99 4.0K 2011-08-11 19:44 test
drwxr-xr-x  6 99 99   58 2011-02-16 17:19 tmp
drwxr-xr-x  4 99 99   49 2011-07-19 16:29 vendor
```



```
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/db$ls data/
development  production  test  uspto
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/db$ls mongo/
address_schema  canada_tm_collection  domain_schema  people_collection  sos_collection  uspto_collection
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/db$ls sphinx/
cucumber  development  production  test
```



```
# This file is auto-generated from the current state of the database. Instead of editing this file,
# please use the migrations feature of Active Record to incrementally modify your database, and
# then regenerate this schema definition.
#
# Note that this schema.rb definition is the authoritative source for your database schema. If you need
# to create the application database on another system, you should be using db:schema:load, not running
# all the migrations from scratch. The latter is a flawed and unsustainable approach (the more migrations
# you'll amass, the slower it'll run and the greater likelihood for issues).
#
# It's strongly recommended to check this file into your version control system.

ActiveRecord::Schema.define(:version => 20101118061222) do

  create_table "account_activations_deprecated", :force => true do |t|
    t.integer  "account_id"
    t.string   "activation_key",                        :null => false
    t.boolean  "active",         :default => false
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  create_table "account_documents", :force => true do |t|
    t.integer  "account_id"
    t.integer  "document_id"
    t.integer  "document_role_id"
    t.datetime "created_at"
    t.datetime "updated_at"
  end

  add_index "account_documents", ["account_id"], :name => "index_account_documents_on_account_id"
  add_index "account_documents", ["document_id"], :name => "index_account_documents_on_document_id"

  create_table "accounts", :force => true do |t|
    t.string   "name"
    t.string   "email",                                           :null => false
    t.string   "crypted_password",                                :null => false
    t.string   "password_salt",                                   :null => false
```



```
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/test/unit$ls
base_url_test.rb   count_cache_test.rb  form_d      ic_category_test.rb  mongoid_ext_test.rb  report            search_cache_test.rb  url_hand
ca_trademark       factory_test.rb      helpers     index_table_test.rb  query_test.rb        routing_test.rb   sos                   widget
copyright          flc_filing           html_meta   lib                  region_test.rb       script            trademark
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/test/unit$find . -type f | wc -l
83
jingmi@ares/ndist/MacBackup/MacBook/Seravia/platform/t-1000/test/unit$grep -RI assert . | wc -l
1550
```

Practices of Continuous Integration(2)

- In Repository

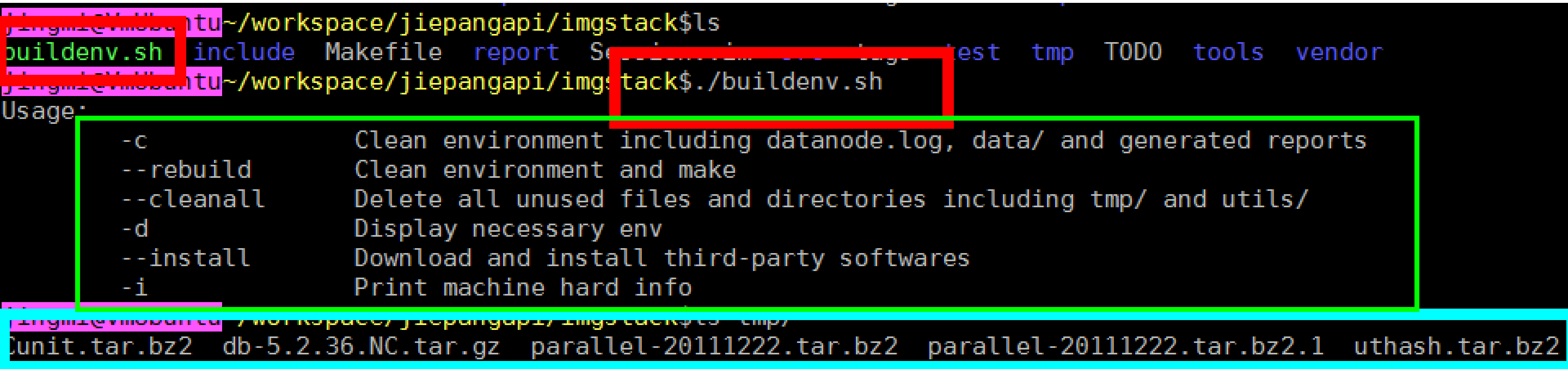
  Only a minimal amount of things should be on the virgin machine - usually things that are large, complicated to install, and stable. An operating system, Java development environment, or base database system are typical examples.

  Should be able to walk up to the project with a virgin machine, do a checkout, and be able to fully build the system.

  But nothing that you actually build.

Automate The Build(1)

- A common mistake is not to include everything in the automated build.
- Anyone should be able to bring in a virgin machine, check the sources out of the repository, issue a single command, and have a running system on their machine.



```
[illegible]tu~/workspace/jiepangapi/imgstack$ls
buildenv.sh  include  Makefile  report  Se[illegible]  [illegible]est  tmp  TODO  tools  vendor
[illegible]tu~/workspace/jiepangapi/imgstack$./buildenv.sh
Usage:
        -c              Clean environment including datanode.log, data/ and generated reports
        --rebuild       Clean environment and make
        --cleanall      Delete all unused files and directories including tmp/ and utils/
        -d              Display necessary env
        --install       Download and install third-party softwares
        -i              Print machine hard info
[illegible]
Cunit.tar.bz2  db-5.2.36.NC.tar.gz  parallel-20111222.tar.bz2  parallel-20111222.tar.bz2.1  uthash.tar.bz2
```

Automate The Build(2)

- So on a Java project we're okay with having developers build in their IDE, but the master build uses Ant to ensure it can be run on the development server.

Make Your Build Self-Testing

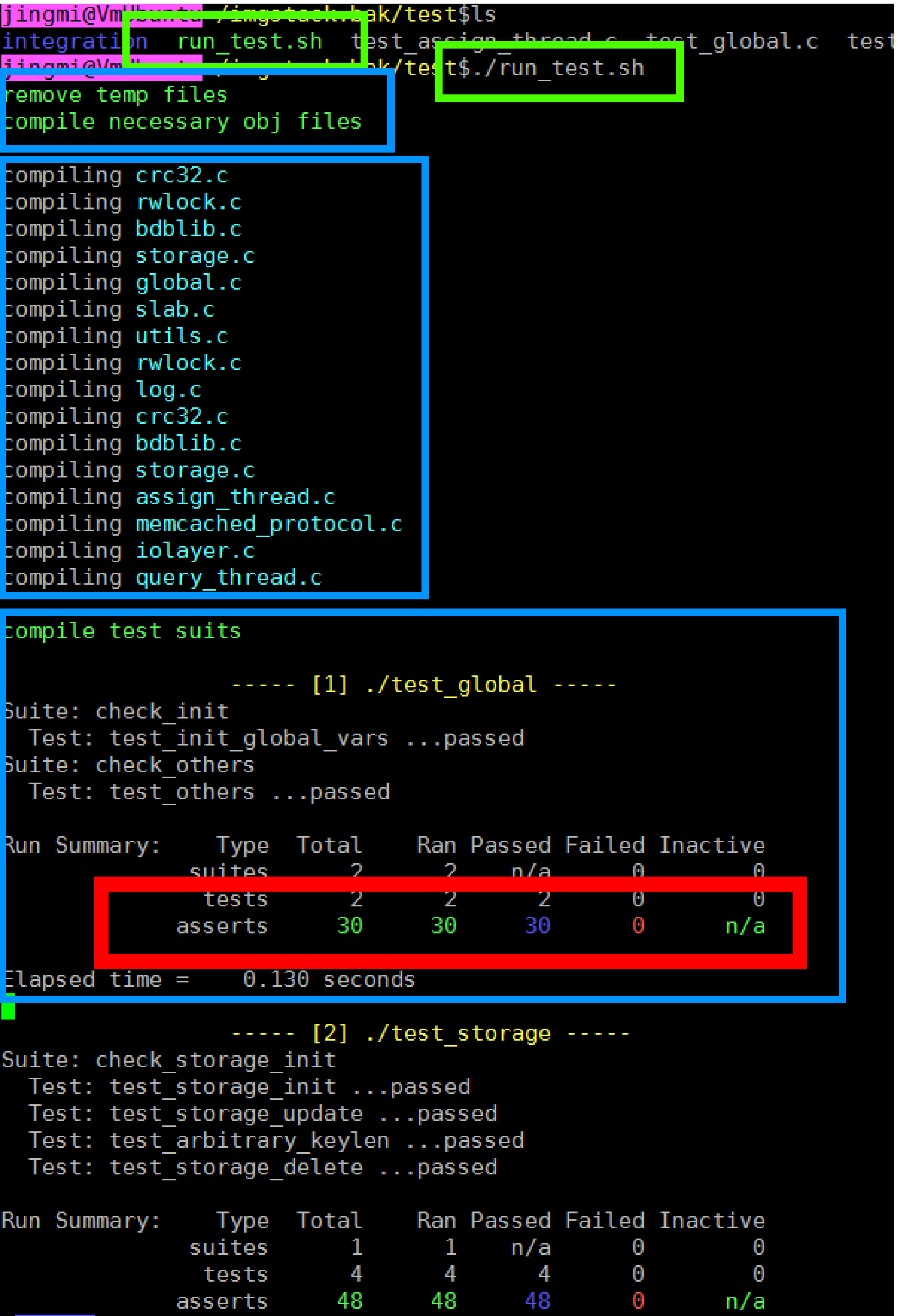
- Purpose of Testing

  TDD/XP make a point of writing tests before you write the code that makes them pass - in this mode the tests are as much about exploring the design of the system as they are about bug catching.

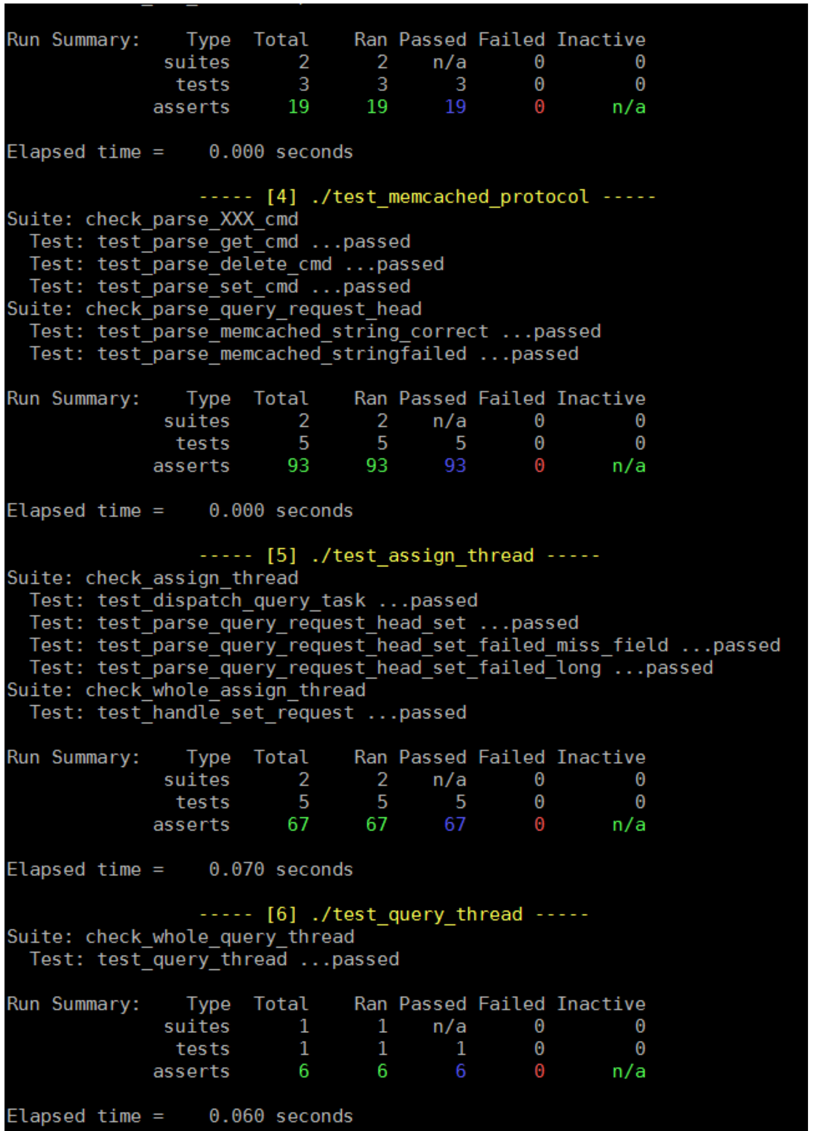
- Simple Command

  The tests need to be able to be kicked off from a simple command and to be self-checking.

Cunit(MyMod)

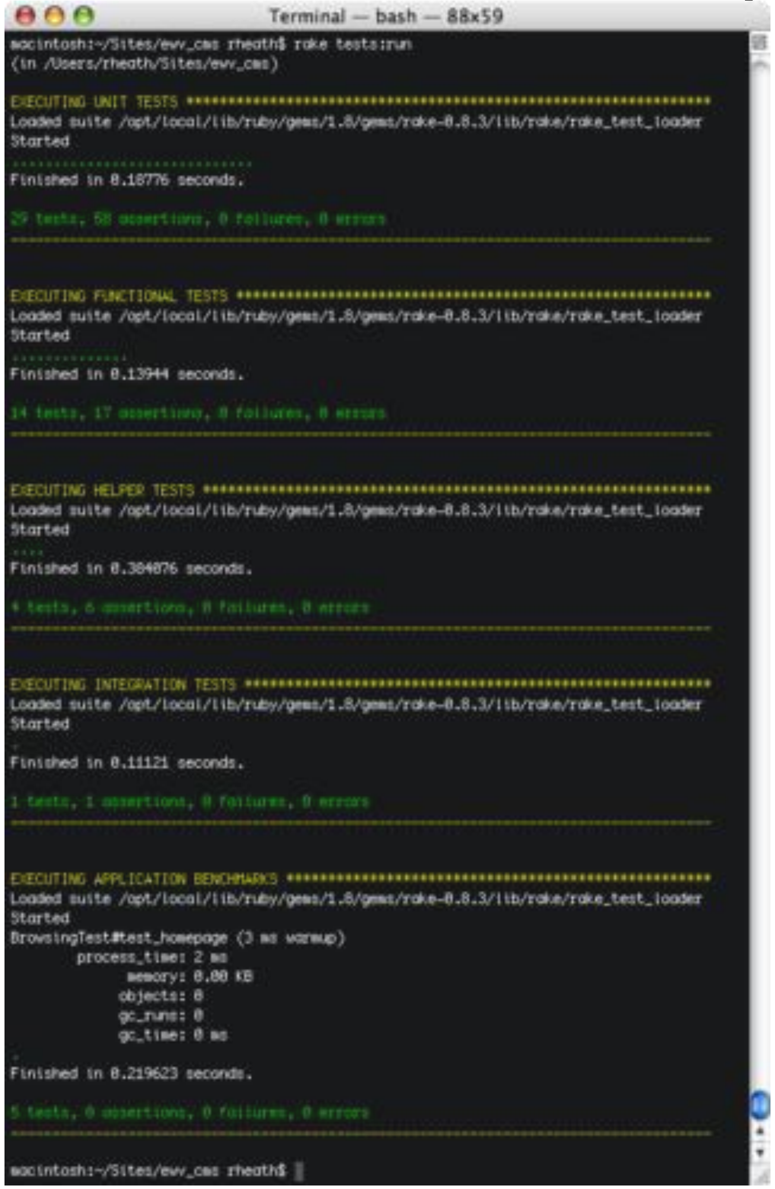


```
jingmi@VmUbuntu ~/imgstack.bak/test$ls
integration  run_test.sh  test_assign_thread.c  test_global.c  test
jingmi@VmUbuntu ~/imgstack.bak/test$./run_test.sh
remove temp files
compile necessary obj files

compiling crc32.c
compiling rwlock.c
compiling bdblib.c
compiling storage.c
compiling global.c
compiling slab.c
compiling utils.c
compiling rwlock.c
compiling log.c
compiling crc32.c
compiling bdblib.c
compiling storage.c
compiling assign_thread.c
compiling memcached_protocol.c
compiling iolayer.c
compiling query_thread.c

compile test suits

                 ----- [1] ./test_global -----
Suite: check_init
  Test: test_init_global_vars ...passed
Suite: check_others
  Test: test_others ...passed

Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      2       2    n/a      0        0
                tests      2       2      2      0        0
              asserts     30      30     30      0      n/a

Elapsed time =    0.130 seconds

                 ----- [2] ./test_storage -----
Suite: check_storage_init
  Test: test_storage_init ...passed
  Test: test_storage_update ...passed
  Test: test_arbitrary_keylen ...passed
  Test: test_storage_delete ...passed

Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      1       1    n/a      0        0
                tests      4       4      4      0        0
              asserts     48      48     48      0      n/a
```

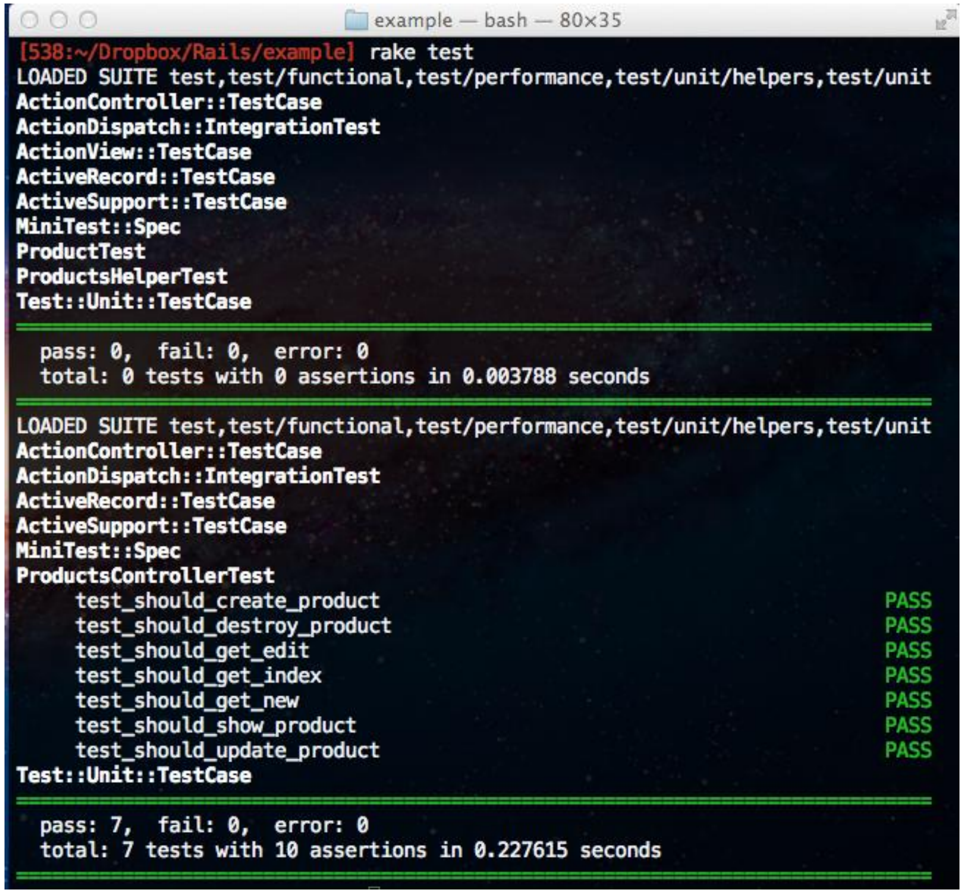


```
Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      2       2    n/a      0        0
                tests      3       3      3      0        0
              asserts     19      19     19      0      n/a

Elapsed time =    0.000 seconds

                 ----- [4] ./test_memcached_protocol -----
Suite: check_parse_XXX_cmd
  Test: test_parse_get_cmd ...passed
  Test: test_parse_delete_cmd ...passed
  Test: test_parse_set_cmd ...passed
Suite: check_parse_query_request_head
  Test: test_parse_memcached_string_correct ...passed
  Test: test_parse_memcached_stringfailed ...passed

Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      2       2    n/a      0        0
                tests      5       5      5      0        0
              asserts     93      93     93      0      n/a

Elapsed time =    0.000 seconds

                 ----- [5] ./test_assign_thread -----
Suite: check_assign_thread
  Test: test_dispatch_query_task ...passed
  Test: test_parse_query_request_head_set ...passed
  Test: test_parse_query_request_head_set_failed_miss_field ...passed
  Test: test_parse_query_request_head_set_failed_long ...passed
Suite: check_whole_assign_thread
  Test: test_handle_set_request ...passed

Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      2       2    n/a      0        0
                tests      5       5      5      0        0
              asserts     67      67     67      0      n/a

Elapsed time =    0.070 seconds

                 ----- [6] ./test_query_thread -----
Suite: check_whole_query_thread
  Test: test_query_thread ...passed

Run Summary:     Type  Total     Ran Passed Failed Inactive
               suites      1       1    n/a      0        0
                tests      1       1      1      0        0
              asserts      6       6      6      0      n/a

Elapsed time =    0.060 seconds
```

Ruby(rake test)

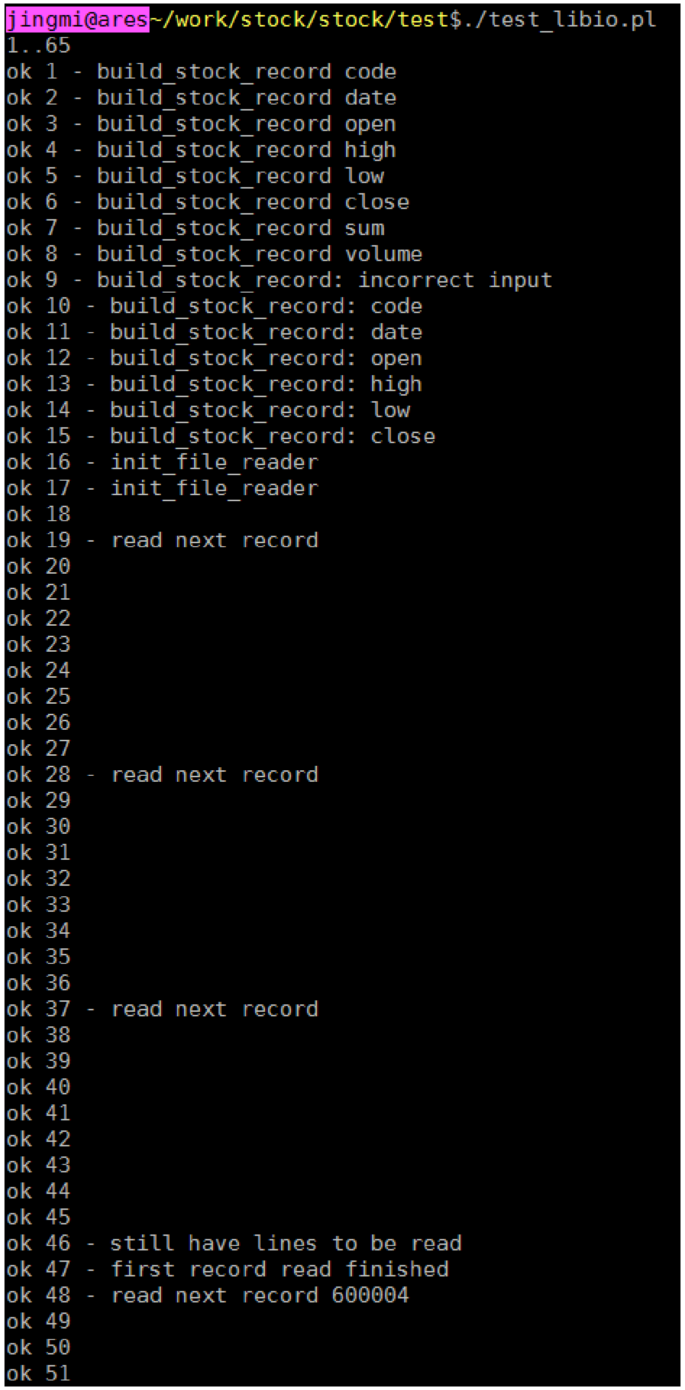


```
macintosh:~/Sites/evv_cms rheath$ rake tests:run
(in /Users/rheath/Sites/evv_cms)

EXECUTING UNIT TESTS ************************************************************
Loaded suite /opt/local/lib/ruby/gems/1.8/gems/rake-0.8.3/lib/rake/rake_test_loader
Started
.............................
Finished in 0.18776 seconds.

29 tests, 58 assertions, 0 failures, 0 errors
---------------------------------------------------------------------------------


EXECUTING FUNCTIONAL TESTS ******************************************************
Loaded suite /opt/local/lib/ruby/gems/1.8/gems/rake-0.8.3/lib/rake/rake_test_loader
Started
..............
Finished in 0.13944 seconds.

14 tests, 17 assertions, 0 failures, 0 errors
---------------------------------------------------------------------------------


EXECUTING HELPER TESTS **********************************************************
Loaded suite /opt/local/lib/ruby/gems/1.8/gems/rake-0.8.3/lib/rake/rake_test_loader
Started
....
Finished in 0.384076 seconds.

4 tests, 6 assertions, 0 failures, 0 errors
---------------------------------------------------------------------------------


EXECUTING INTEGRATION TESTS *****************************************************
Loaded suite /opt/local/lib/ruby/gems/1.8/gems/rake-0.8.3/lib/rake/rake_test_loader
Started
.
Finished in 0.11121 seconds.

1 tests, 1 assertions, 0 failures, 0 errors
---------------------------------------------------------------------------------


EXECUTING APPLICATION BENCHMARKS ************************************************
Loaded suite /opt/local/lib/ruby/gems/1.8/gems/rake-0.8.3/lib/rake/rake_test_loader
Started
BrowsingTest#test_homepage (3 ms warmup)
        process_time: 2 ms
              memory: 0.00 KB
             objects: 0
             gc_runs: 0
             gc_time: 0 ms
.
Finished in 0.219623 seconds.

5 tests, 0 assertions, 0 failures, 0 errors
---------------------------------------------------------------------------------

macintosh:~/Sites/evv_cms rheath$
```



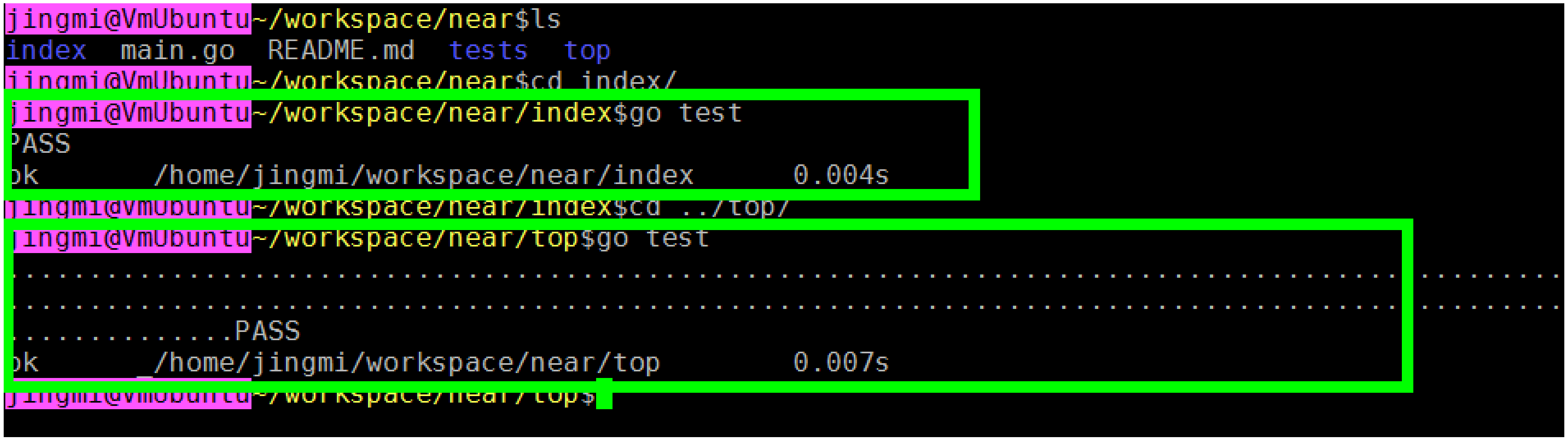
```
[538:~/Dropbox/Rails/example] rake test
LOADED SUITE test,test/functional,test/performance,test/unit/helpers,test/unit
ActionController::TestCase
ActionDispatch::IntegrationTest
ActionView::TestCase
ActiveRecord::TestCase
ActiveSupport::TestCase
MiniTest::Spec
ProductTest
ProductsHelperTest
Test::Unit::TestCase
==============================================================================
  pass: 0,  fail: 0,  error: 0
  total: 0 tests with 0 assertions in 0.003788 seconds
==============================================================================
LOADED SUITE test,test/functional,test/performance,test/unit/helpers,test/unit
ActionController::TestCase
ActionDispatch::IntegrationTest
ActiveRecord::TestCase
ActiveSupport::TestCase
MiniTest::Spec
ProductsControllerTest
     test_should_create_product                                             PASS
     test_should_destroy_product                                            PASS
     test_should_get_edit                                                   PASS
     test_should_get_index                                                  PASS
     test_should_get_new                                                    PASS
     test_should_show_product                                               PASS
     test_should_update_product                                             PASS
Test::Unit::TestCase
==============================================================================
  pass: 7,  fail: 0,  error: 0
  total: 7 tests with 10 assertions in 0.227615 seconds
==============================================================================
```

Perl(Unit test)



```
jingmi@ares~/work/stock/stock/test$./test_libio.pl
1..65
ok 1 - build_stock_record code
ok 2 - build_stock_record date
ok 3 - build_stock_record open
ok 4 - build_stock_record high
ok 5 - build_stock_record low
ok 6 - build_stock_record close
ok 7 - build_stock_record sum
ok 8 - build_stock_record volume
ok 9 - build_stock_record: incorrect input
ok 10 - build_stock_record: code
ok 11 - build_stock_record: date
ok 12 - build_stock_record: open
ok 13 - build_stock_record: high
ok 14 - build_stock_record: low
ok 15 - build_stock_record: close
ok 16 - init_file_reader
ok 17 - init_file_reader
ok 18
ok 19 - read next record
ok 20
ok 21
ok 22
ok 23
ok 24
ok 25
ok 26
ok 27
ok 28 - read next record
ok 29
ok 30
ok 31
ok 32
ok 33
ok 34
ok 35
ok 36
ok 37 - read next record
ok 38
ok 39
ok 40
ok 41
ok 42
ok 43
ok 44
ok 45
ok 46 - still have lines to be read
ok 47 - first record read finished
ok 48 - read next record 600004
ok 49
ok 50
ok 51
```

Go(Unit Test)



```
jingmi@VmUbuntu~/workspace/near$ls
index  main.go  README.md  tests  top
jingmi@VmUbuntu~/workspace/near$cd index/
jingmi@VmUbuntu~/workspace/near/index$go test
PASS
ok      /home/jingmi/workspace/near/index      0.004s
jingmi@VmUbuntu~/workspace/near/index$cd ../top/
jingmi@VmUbuntu~/workspace/near/top$go test
.........................................................................................
.........................................................................................
..............PASS
ok      /home/jingmi/workspace/near/top        0.007s
jingmi@VmUbuntu~/workspace/near/top$
```

Make Your Build Self-Testing(2)

- Tests don't prove the absence of bugs.

Make Your Build Self-Testing(3)

- How to test a distributed system that deployed on thousands of machines(such as GFS/BigTable/Hadoop)

Everyone Commits To the Mainline Every Day

- Communication

  Integration is primarily about communication.

- Frequent commits

  Frequent commits encourage developers to break down their work into small chunks.

- Monitor the mainline build.

  Developer needs to monitor the mainline build so they can fix it if it breaks.

Keep the Build Fast

- Rapid feedback

  The whole point of Continuous Integration is to provide rapid feedback.

- Two stage build

  The first stage would do the compilation and run tests that are more localized unit tests with the database completely stubbed out.

  The second stage build runs a different suite of tests that do hit the real database and involve more end-to-end behavior.

Test in a Clone of the Production Environment(1)

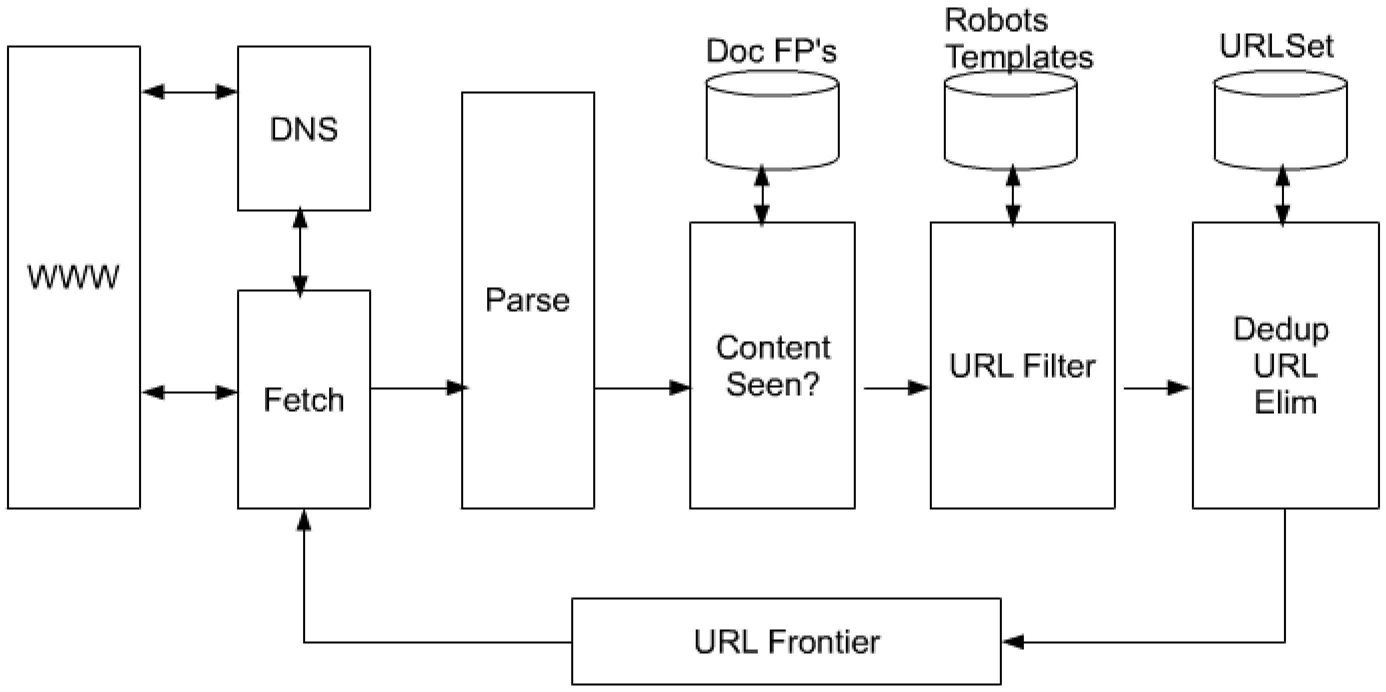
- Key of testing

  The point of testing is to flush out, under controlled conditions, any problem that the system will have in production.

  As a result it's common to have a very artificial environment for the commit tests for speed, and use a production clone for secondary testing.

Test in a Clone of the Production Environment(2)

- How to test a distributed crawler/spider system?



Everyone can see what's happening(1)

- Notification

  Ensure that everyone can easily see the state of the system and the changes that have been made to it.

- Communicate with mainline build

  One of the most important things to communicate is the state of the mainline build.

- Monitor

  The monitor of the physical build machine can show the status of the mainline build.

- Noise

  Often people like to make a simple noise on good builds, like ringing a bell.

Everyone can see what's happening(2)

- Grooming
- IPM(Iteration Planning Meeting)
- Board + Card(Trello)

Automate Deployment(1)

- Deploy Scripts

  A natural consequence of this is that you should also have scripts that allow you to deploy into production with similar ease.

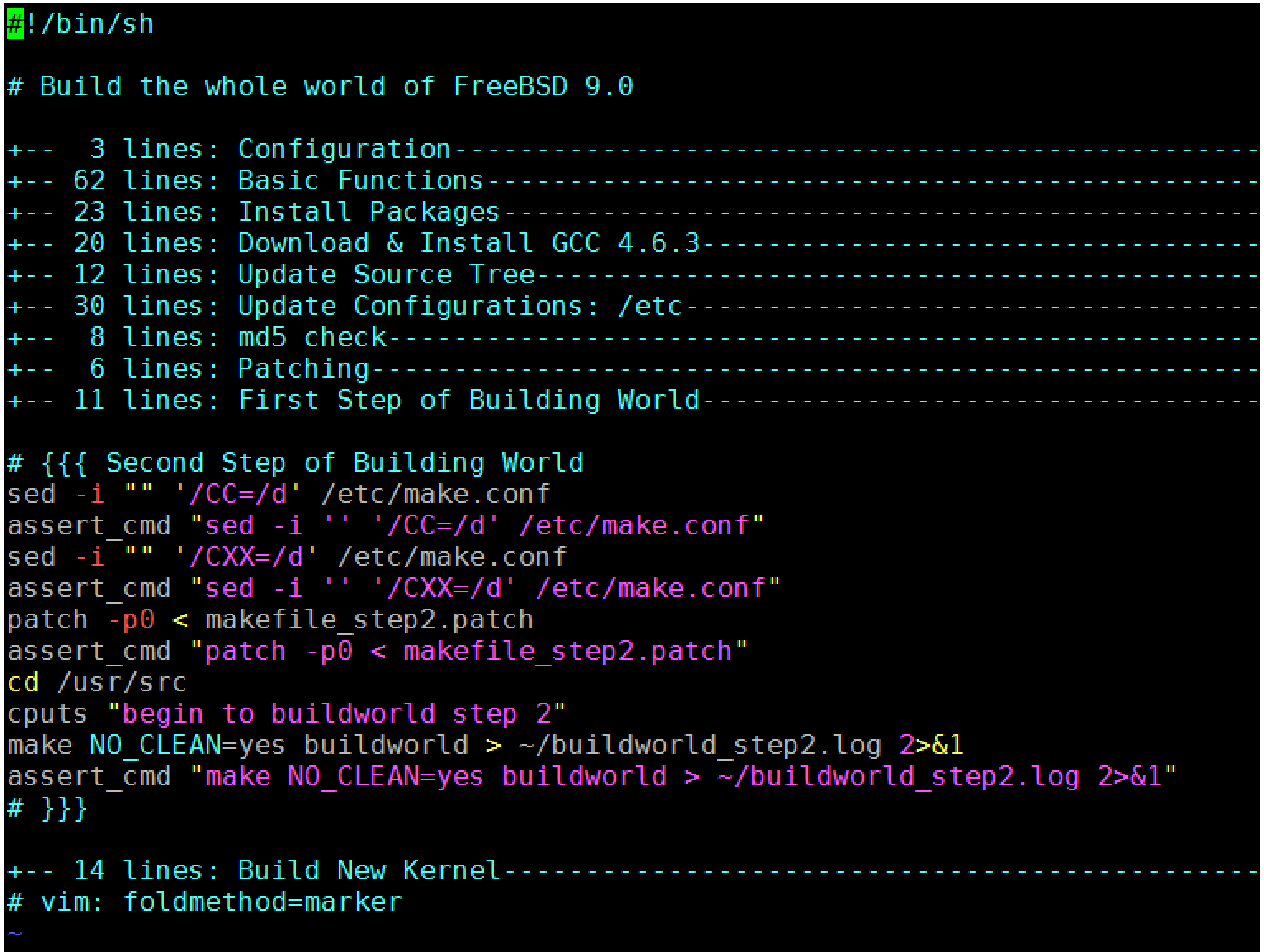
- Automated rollback

  If you deploy into production one extra automated capability you should consider is automated rollback.

- Rolling deployments

  The new software is deployed to one node at a time, gradually replacing the application over the course of a few hours.

Automate Deployment(2)



```
#!/bin/sh

# Build the whole world of FreeBSD 9.0

+--  3 lines: Configuration-----------------------------------------------
+-- 62 lines: Basic Functions---------------------------------------------
+-- 23 lines: Install Packages--------------------------------------------
+-- 20 lines: Download & Install GCC 4.6.3--------------------------------
+-- 12 lines: Update Source Tree------------------------------------------
+-- 30 lines: Update Configurations: /etc---------------------------------
+--  8 lines: md5 check---------------------------------------------------
+--  6 lines: Patching----------------------------------------------------
+-- 11 lines: First Step of Building World--------------------------------

# {{{ Second Step of Building World
sed -i "" '/CC=/d' /etc/make.conf
assert_cmd "sed -i '' '/CC=/d' /etc/make.conf"
sed -i "" '/CXX=/d' /etc/make.conf
assert_cmd "sed -i '' '/CXX=/d' /etc/make.conf"
patch -p0 < makefile_step2.patch
assert_cmd "patch -p0 < makefile_step2.patch"
cd /usr/src
cputs "begin to buildworld step 2"
make NO_CLEAN=yes buildworld > ~/buildworld_step2.log 2>&1
assert_cmd "make NO_CLEAN=yes buildworld > ~/buildworld_step2.log 2>&1"
# }}}

+-- 14 lines: Build New Kernel--------------------------------------------
# vim: foldmethod=marker
~
```

Automate Deployment(3)

- Capistrano
- Puppet

Benefits of Continuous Integration(1)

- Reduced Risk

  On the whole I think the greatest and most wide ranging benefit of Continuous Integration is reduced risk.

- Hard to Estimate

  The trouble with deferred integration is that it's very hard to predict how long it will take to do, and worse it's very hard to see how far you are through the process.

Benefits of Continuous Integration(2)

- Bugs

  Continuous Integrations doesn't get rid of bugs, but it does make them dramatically easier to find and remove.

  Bugs are also cumulative.

- Debugging

  You can also use diff debugging - comparing the current version of the system to an earlier one that didn't have the bug.

Introducing Continuous Integration

- It depends

  There's no fixed recipe here - much depends on the nature of your setup and team.

- Build Automated

  One of the first steps is to get the build automated.

- Automated Testing

  Introduce some automated testing into your build.

- Commit

  Try to speed up the commit build.