PB173 - Tématický vývoj aplikací v C/C++
Domain specific development in C/C++

Skupina: Aplikovaná kryptografie a bezpečné programování
https://is.muni.cz/auth/predmety/uplny_vypis?fakulta=1433;obdobi=6384;predmet=871304

Petr Švenda svenda@fi.muni.cz

CROCS
Centre for Research on Cryptography and Security

VERIFICATION OF LIBRARY

GPG verification – missing key problem

```
gpg --verify openssl-1.0.1i.tar.gz.asc    (soubor s podpisem, ne knihovna
                                          samotna, predpoklad existence openssl-1.0.1i.tar.gz)

GnuPG>gpg --verify openssl-1.0.1i.tar.gz.asc
gpg: Signature made 08/06/14 23:18:48 using RSA key ID 0E604491
gpg: Can't check signature: public key not found
```

- Problem: we don’t have key used to create this signature

GPG – find and download key

```
https://www.openssl.org/about/
Matt Caswell   matt@openssl.org       UK      0E604491

pgp.mit.edu
http://pgp.mit.edu:11371/pks/lookup?op=vindex&search=0x8657ABB260F
-> matt.asc

GnuPG>gpg --keyserver pgpkeys.mit.edu --recv-key 0E604491

GnuPG>gpg --import matt.asc
gpg: key 0E604491: public key "Matt Caswell <matt@openssl.org>" importe
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:  21  signed:   3  trust: 0-, 0q, 0n, 0m, 0f, 21u
gpg: depth: 1  valid:   3  signed:   0  trust: 3-, 0q, 0n, 0m, 0f, 0u
gpg: next trustdb check due at 2016-06-04
```

GPG verification – untrusted key problem

```
GnuPG>gpg --verify openssl-1.0.1i.tar.gz.asc
gpg: Signature made 08/06/14 23:18:48 using RSA key ID 0E604491
gpg: Good signature from "Matt Caswell <matt@openssl.org>"
gpg:                 aka "Matt Caswell <frodo@baggins.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 8657 ABB2 60F0 56B1 E519  0839 D9C4 D26D 0E6
```

- We have the key, but GPG doesn't know if it is trusted
- You need to set trust (and you will need your keypair for this operation)
  - `GnuPG>gpg --gen-key`
  - `...` generate your ultimately trusted key

GPG edit trust on OpenSSL key

```
GnuPG>gpg --edit-key 0E604491 trust
gpg (GnuPG) 1.4.7; Copyright (C) 2006 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

pub  2048R/0E604491  created: 2013-04-30  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  2048R/E3C21B70  created: 2013-04-30  expires: never       usage: E
[ unknown] (1). Matt Caswell <matt@openssl.org>
[ unknown] (2)  Matt Caswell <frodo@baggins.org>

pub  2048R/0E604491  created: 2013-04-30  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  2048R/E3C21B70  created: 2013-04-30  expires: never       usage: E
[ unknown] (1). Matt Caswell <matt@openssl.org>
[ unknown] (2)  Matt Caswell <frodo@baggins.org>
```

Please decide how far you trust this user to correctly **verify** other users' key
(by looking at passports, checking fingerprints from different sources, etc.)

1 = I don't know or won't say
2 = I **do NOT** trust
3 = I trust marginally
4 = I trust fully
5 = I trust ultimately
m = back to the main menu

Your decision? 5
Do you really want to **set** this key to ultimate trust? (y/N) y

pub 2048R/0E604491 created: 2013-04-30 expires: never usage: SC
trust: ultimate validity: unknown
sub 2048R/E3C21B70 created: 2013-04-30 expires: never usage: E
[unknown] (1). Matt Caswell <matt@openssl.org>
[unknown] (2) Matt Caswell <frodo@baggins.org>
Please note that the shown key validity is **not** necessarily correct
unless you restart the program.

GPG verification – finally correct

```
GnuPG>gpg --verify openssl-1.0.1i.tar.gz.asc
gpg: Signature made 08/06/14 23:18:48 using RSA key ID 0E604491
gpg: Good signature from "Matt Caswell <matt@openssl.org>"
gpg:                 aka "Matt Caswell <frodo@baggins.org>"
```

CONTINUOUS INTEGRATION


YES WE CODE

+ Catch

+  +  + 

=

build passing

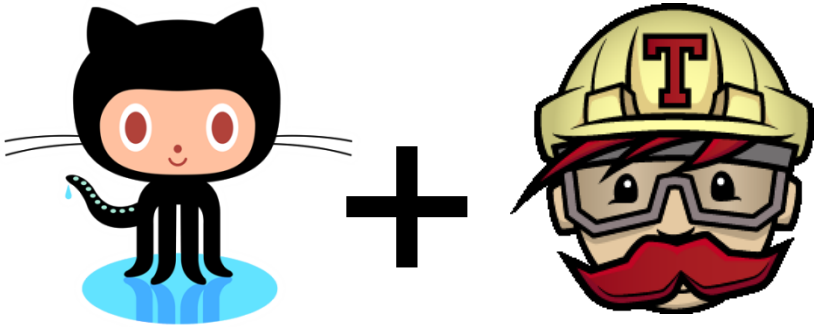
Klasický způsob vývoje

- Old-school styl vývoje
  - Zadání rozděleno do dílčích problémů oddělených rozhraními
  - Vývojáři pracují na separátních komponentách
    - Separátní větve v SVN repozitářích
    - Testy na úrovni dílčí komponenty
- Po nějaké době chceme provést spojení dílčích částí
  - Stačí jen Merge & Compile?
- “Integration hell” přichází
  - Rozhraní lehce upraveny
  - Sdílený kód modifikován
  - Chybné spojení během merge
  - …

Continuous integration (CI)

- Originálně navrženo pro Extreme Programming
  - Nyní široce využíváno
  - Složení více dobrých vývojových technik a postupů
  - Celý produkt je stále “připraven” (night builds)
- Hlavní větev je spojena, kompilována a automaticky testována i několikrát denně
  - CI server (Jenkins, Travis CI…)
  - Versioning system (SVN, GIT…)
  - Automatický build (make, Ant, Maven…)
  - Automatické testy (unit testy, integrační testy)
  - Dodatečná analýza (statická analýza, výkonostní testy…)
  - Prezentace výsledků (grafický web…)

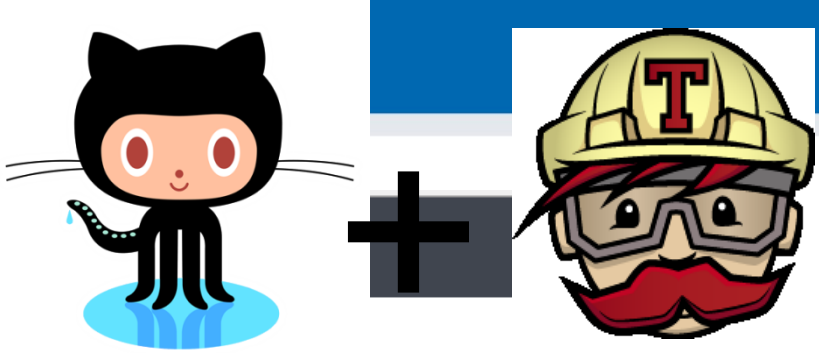
CI : GitHub + Travis CI



1. Create GitHub repository
   - Create Makefile for automated build
2. Sign into Travis-CI.org with GitHub account
3. Add New Repository (button +)
   - Your GitHub repositories are listed
   - Activate GitHub Webhook for target repo
4. Add .travis.yml file to root of your repo
   - Fill content according your language and build process
5. Trigger Travis build with Git push
   - Results available at https://travis-ci.org/ and by email

```
language: cpp
compiler:
  - gcc
  - clang
script: make
```

- http://docs.travis-ci.com/user/getting-started/



travis Home Blog Status Help Travis CI for Priv

Search all repositories

My Repositories Recent +

petrs/EACirc 7
32 sec
35 minutes ago

petrs/JCAlgTest 11
1 min 22 sec
about 15 hours ago

petrs/JCAlgTest

build passing

Automated testing tool for algorithms supported by particular smart card with JavaCard platform

Current | Build History | Pull Requests | Branch Summary

Build	Message	Commit	Duration	Finished
11	updated for Coverity scan	f368b51 (master)	1 min 22 sec	about 15 hours ago
10	updated global build.xml	f663dce (master)	7 sec	about 16 hours ago
9	updated build.xml	ccb7fff (master)	2 sec	about 16 hours ago
8	returned global build.xml	6eb1d9c (master)	3 sec	about 16 hours ago
7	default ant build	5f892b0 (master)	2 sec	about 17 hours ago
6	updated base dir in global build.xml	de1aace (master)	7 sec	about 17 hours ago
5	updated global build.xml	7fa6023 (master)	4 sec	about 17 hours ago
4	build file for AlgTest_JClient moved	a4b484c (master)	14 sec	about 17 hours ago

GitHub + Travis CI

+

- Continuous integration
  - Compile and run tests after every **`git push`**
- https://github.com/crocs-muni/git-travis-demo
- **`Makefile`**
  - Automatic compilation and test execution
  - Generic build script (no special dependency on GitHub)
- **`.travis.yml`**
  - Configuration file instructing GitHub to use Travis CI

Makefile

Basic variables

Build rule(s):

`all` – executes rules main and main-test

`test` – builds main-test and runs it

`main` – build regular program

Files to be build. Can be named (main.cpp) or automatic rule for all *.cpp files (used here)

```
# Makefile example
# Variables CC and CXX are automatically set on all UNIX systems.
CXXFLAGS=-Wall -Wextra
SOURCES_GEN=src/fact.cpp
# Source and object lists for main program
SOURCES_MAIN=$(SOURCES_GEN) src/main.cpp
OBJECTS_MAIN=$(SOURCES_MAIN:.cpp=.o)
# Source and object lists for testing binary
SOURCES_TEST=$(SOURCES_GEN) src/testing.cpp
OBJECTS_TEST=$(SOURCES_TEST:.cpp=.o)

# Most frequently used automatic variables:
# $@ (name of the target rule)
# $< (name of the first prerequisite)
# $^ (name of all the prerequisites)

# Target 'all' has 'main' and 'main-test' as dependencies.
all: main main-test

# Depends on main-test, runs the test program.
test: main-test
	./main-test

# Depends on all object files and main, links the final binary.
main: $(OBJECTS_MAIN)
	$(CXX) $(CXXFLAGS) -o $@ $^

# Depends on all object files and test, links the test binary.
main-test: $(OBJECTS_TEST)
	$(CXX) $(CXXFLAGS) -o $@ $^

# Automatic rule for all object files in build directory
%.o: %.cpp
	$(CXX) $(CXXFLAGS) -c -o $@ $<

clean:
	rm -fr $(OBJECTS_MAIN) $(OBJECTS_TEST)
```

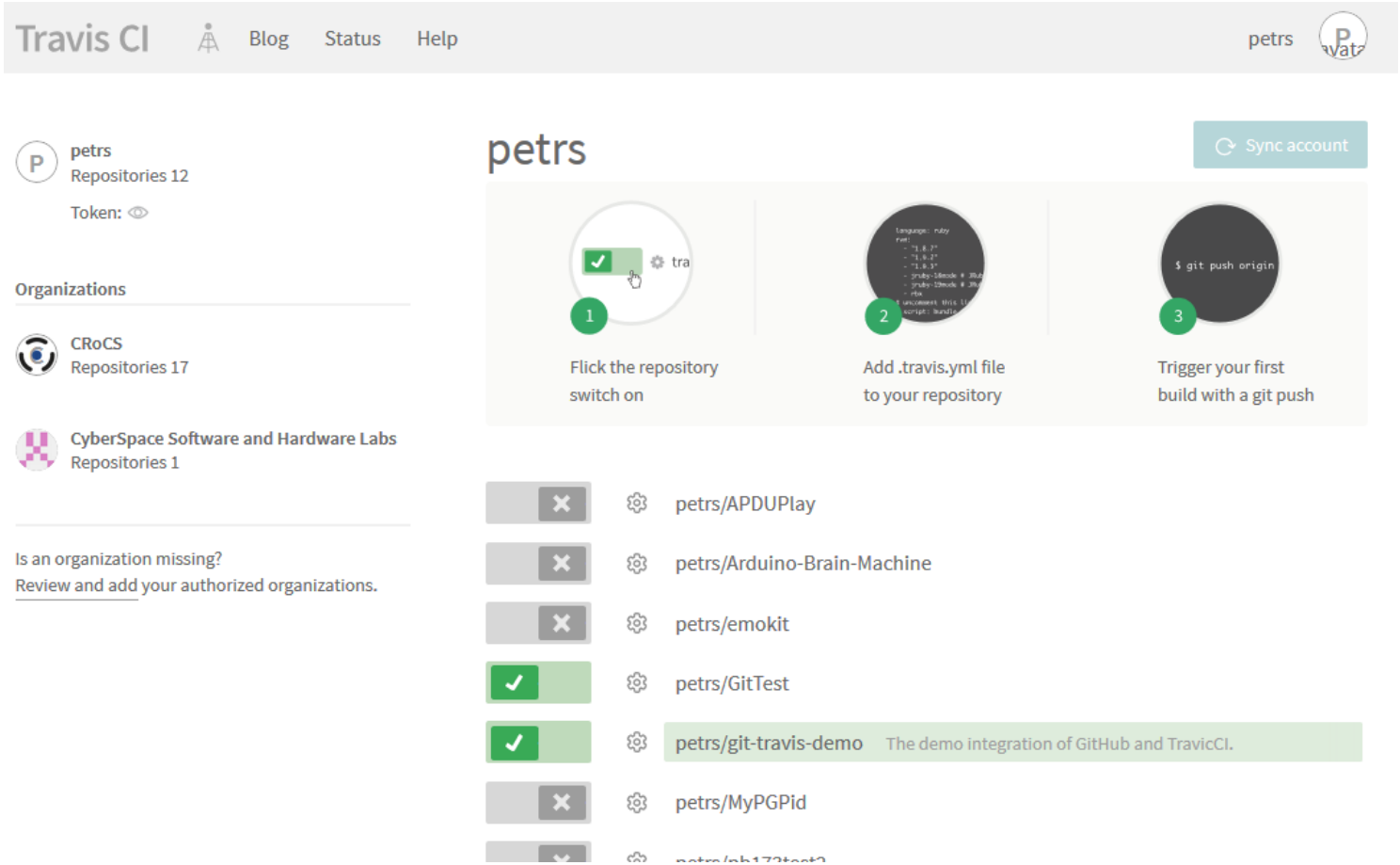
.travis.yml

```
# TravisCI build settings file
# For more info, see http://docs.travis-ci.com/user/getting-started/
# To validate your .travis.yml, go to http://lint.travis-ci.org/

# setting the project language
language: cpp

# setting compilers, do 2 separate sub-builds for gcc and clang
compiler:
  - gcc
  - clang

# script to run after build (run tests, etc.)
script: make all test
```

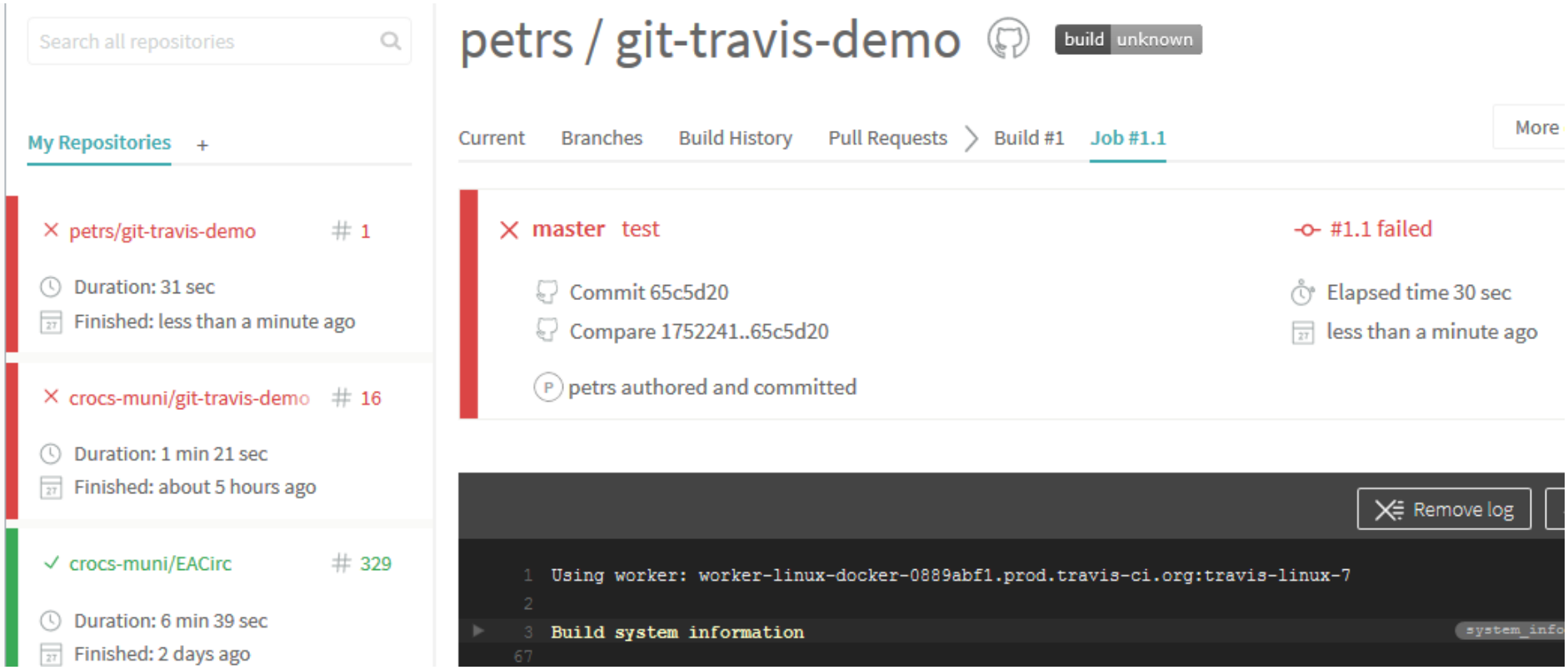
Practical assignment

1. Create Travis CI account (http://travis-ci.org)
2. Fork https://github.com/crocs-muni/git-travis-demo
3. Create local copy of forked repo
4. Enable git-travis-demo at travis-ci.org
5. Push updated files
6. Observe TravisCI build and response
7. Connect your repo (from last lecture) with TravisCI
   - Add tests (Catch)
   - Add Makefile (compile, run test)
8. Push into your repo and observe Travis CI

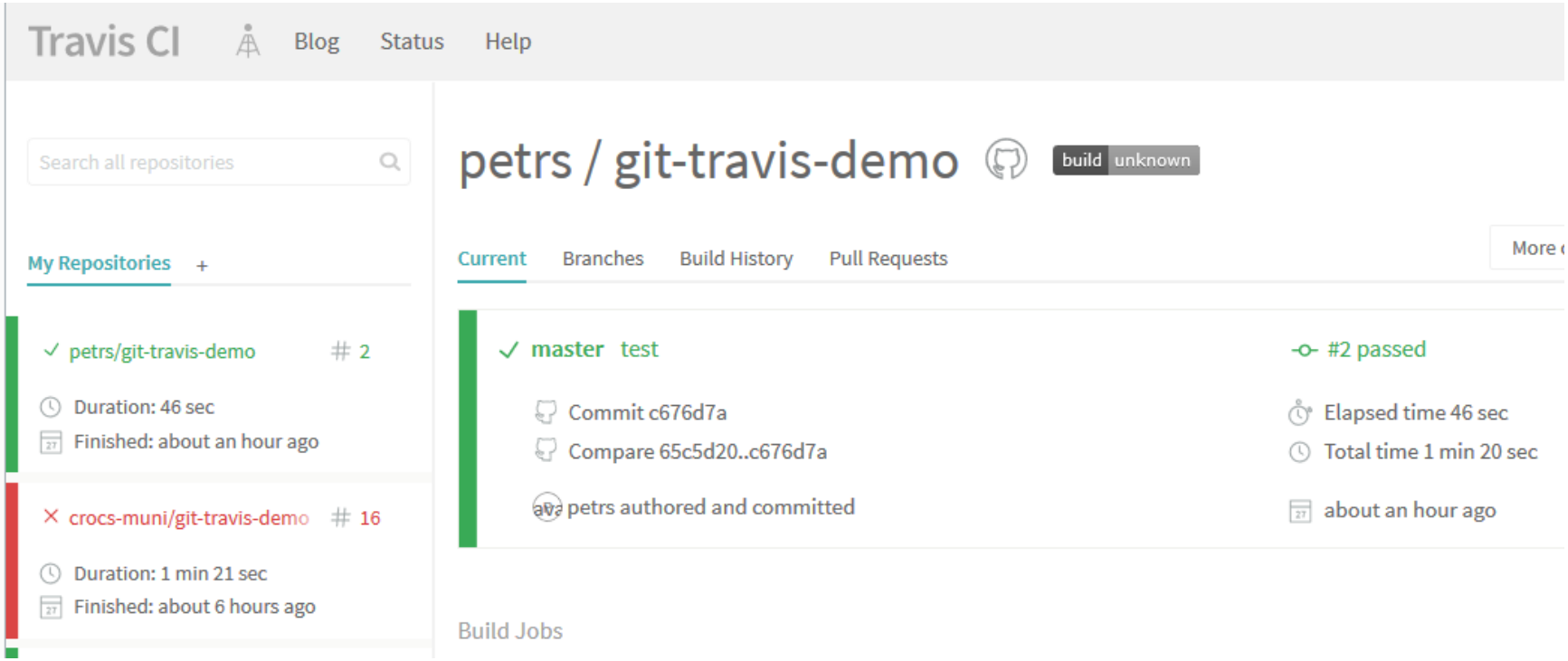
Enable GitHub repo in Travis GUI



Tests are initially failing



Fix tests to pass



Homework

- Write following simple unit tests:
  - file not exists or cannot be read/written into
  - encrypted blob was corrupted
  - wrong decryption key was used
  - test vectors for encryption and hashing
  - Use UT framework you like (Catch, QTest, UnitTest++, CxxTest…)
- Integrate your tests into Travis CI
- Best practices
  - http://blog.stevensanderson.com/2009/08/24/writing-great-unit-tests-best-and-worst-practises/
  - http://www.levelofindirection.com/journal/2010/12/28/unit-testing-in-c-and-objective-c-just-got-easier.html
- Code will be used later in architecture
  - will be used again and extended, so write it well ☺

Submissions, deadlines

- Upload application source codes as single zip file into IS Homework vault (Crypto - 2. homework (UT))
  - Finalized codes based on the discussions during lecture
  - Added unit tests
- DEADLINE 6.3. 12:00 (second part)
  - addition of unit tests
  - 0-5 points assigned

Questions?