C.E.C.I

Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Introduction to Scientific Software Deployment and Development

damien.francois@uclouvain.be

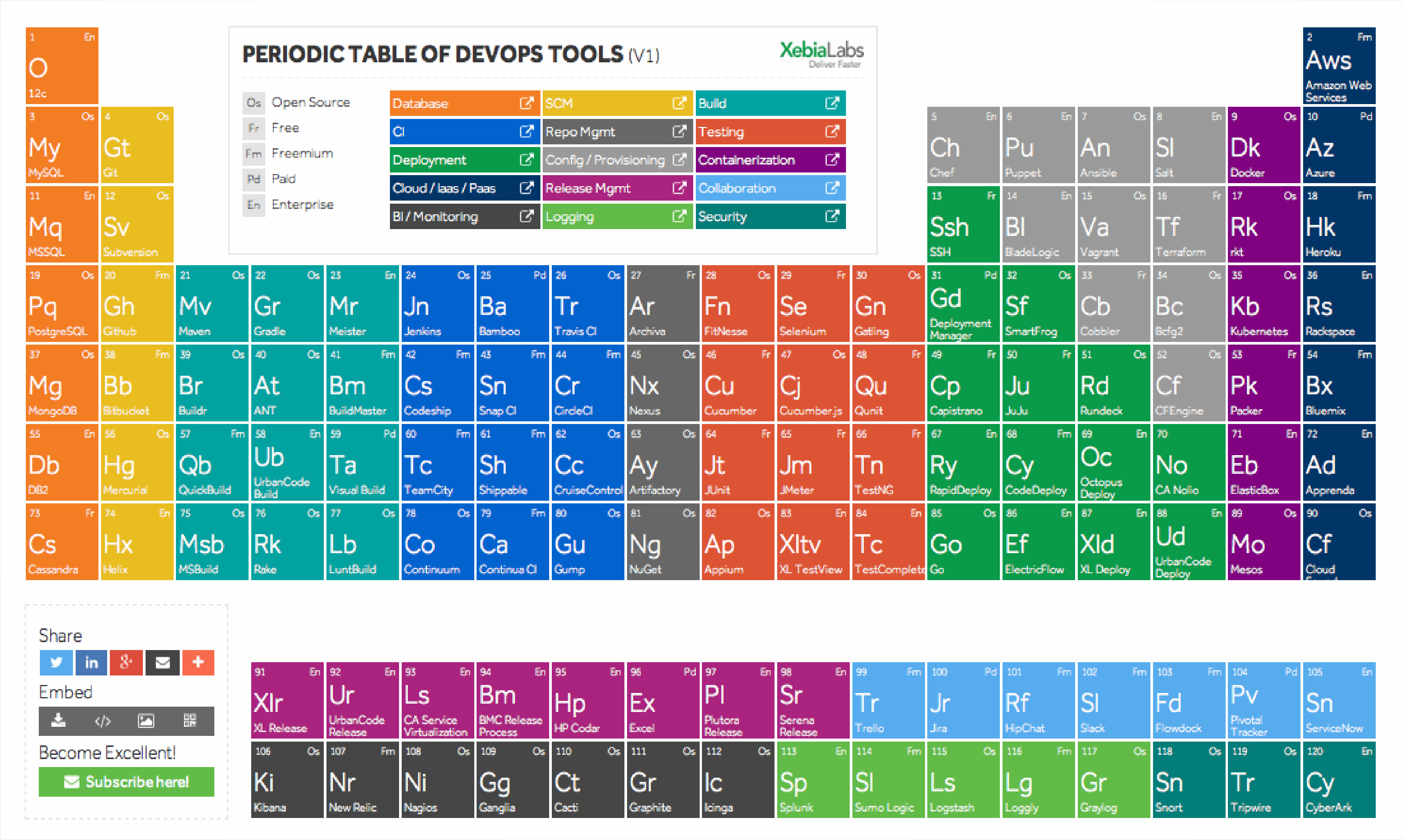
October 2018

http://www.ceci-hpc.be/training.html

UCL Université catholique de Louvain

INSTITUT DE CALCUL INTENSIF ET DE STOCKAGE DE MASSE

What is this?



PERIODIC TABLE OF DEVOPS TOOLS (V1)

XebiaLabs
Deliver Faster

Os	Open Source
Fr	Free
Fm	Freemium
Pd	Paid
En	Enterprise

Database	SCM	Build
CI	Repo Mgmt	Testing
Deployment	Config / Provisioning	Containerization
Cloud / Iaas / Paas	Release Mgmt	Collaboration
BI / Monitoring	Logging	Security

1 En O 12c																	2 Fm Aws Amazon Web Services
3 Os My MySQL	4 Os Gt Git											5 En Ch Chef	6 En Pu Puppet	7 Os An Ansible	8 En Sl Salt	9 Os Dk Docker	10 Pd Az Azure
11 En Mq MSSQL	12 Os Sv Subversion											13 Fr Ssh SSH	14 En Bl BladeLogic	15 Os Va Vagrant	16 Fr Tf Terraform	17 Os Rk rkt	18 Fm Hk Heroku
19 Os Pq PostgreSQL	20 Fm Gh Github	21 Os Mv Maven	22 Os Gr Gradle	23 En Mr Meister	24 Os Jn Jenkins	25 Pd Ba Bamboo	26 Os Tr Travis CI	27 Fr Ar Archiva	28 Os Fn FitNesse	29 Fr Se Selenium	30 Os Gn Gatling	31 Pd Gd Deployment Manager	32 Os Sf SmartFrog	33 Fr Cb Cobbler	34 Os Bc Bcfg2	35 Os Kb Kubernetes	36 En Rs Rackspace
37 Os Mg MongoDB	38 Fm Bb Bitbucket	39 Os Br Buildr	40 Os At ANT	41 Fm Bm BuildMaster	42 Fm Cs Codeship	43 Fm Sn Snap CI	44 Fm Cr CircleCI	45 Os Nx Nexus	46 Fr Cu Cucumber	47 Os Cj Cucumber.js	48 Fr Qu Qunit	49 Fr Cp Capistrano	50 Fr Ju JuJu	51 Os Rd Rundeck	52 Os Cf CFEngine	53 Fr Pk Packer	54 Fm Bx Bluemix
55 En Db DB2	56 Os Hg Mercurial	57 Fm Qb QuickBuild	58 En Ub UrbanCode Build	59 Pd Ta Visual Build	60 Fm Tc TeamCity	61 Fm Sh Shippable	62 Os Cc CruiseControl	63 Os Ay Artifactory	64 Fr Jt JUnit	65 Fr Jm JMeter	66 Fr Tn TestNG	67 En Ry RapidDeploy	68 Fm Cy CodeDeploy	69 En Oc Octopus Deploy	70 No CA Nolio	71 En Eb ElasticBox	72 En Ad Apprenda
73 Fr Cs Cassandra	74 En Hx Helix	75 Os Msb MSBuild	76 Os Rk Rake	77 Os Lb LuntBuild	78 Os Co Continuum	79 Fm Ca Continua CI	80 Os Gu Gump	81 Os Ng NuGet	82 Os Ap Appium	83 En Xltv XL TestView	84 En Tc TestComplete	85 Os Go Go	86 En Ef ElectricFlow	87 En Xld XL Deploy	88 En Ud UrbanCode Deploy	89 Os Mo Mesos	90 Os Cf Cloud

Share

Embed

Become Excellent!

Subscribe here!

91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Ls CA Service Virtualization	94 En Bm BMC Release Process	95 En Hp HP Codar	96 Pd Ex Excel	97 En Pl Plutora Release	98 En Sr Serena Release	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow
106 Os Ki Kibana	107 Fm Nr New Relic	108 Os Ni Nagios	109 Os Gg Ganglia	110 Os Ct Cacti	111 Os Gr Graphite	112 Os Ic Icinga	113 En Sp Splunk	114 Fm Sl Sumo Logic	115 Os Ls Logstash	116 Fm Lg Loggly	117 Os Gr Graylog	118 Os Sn Snort	119 Os Tr Tripwire	120 En Cy CyberArk

https://xebialabs.com/periodic-table-of-devops-tools/

Goal of this session:

“Give you access to the same tools
the professionals are using
for **developing** and **deploying** programs.”

Dev's toolkit :

1. Programming languages
2. Good practices/principles/style
3. Text editor
4. Source control management
5. Debuggers / Profilers
6. Databases
7. Packaging / Distributing tools
8. Comments and documentation
9. Tests
10. Licensing

1. Programming language

- Good reasons for choosing language X:
  - it offers useful paradigms for your problem
  - it offers high-level constructs/tools - e.g. for parsing arguments
  - it offers (directly or indirectly) useful libraries - e.g. for linear algebra
- Ok reasons for choosing language X:
  - standard in your community – easier to get accepted
- Bad reasons for choosing language X:
  - it runs fast – probably needs high skills to be fast
  - it is the language you already know

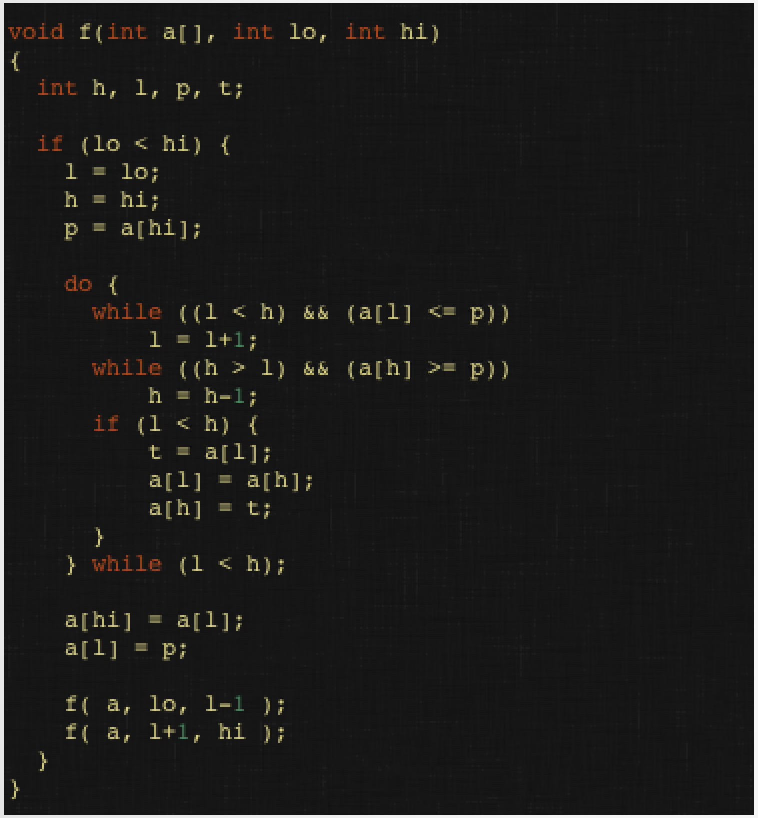
1. Programming language

Be aware of the 'other' paradigm...

Imperative – “Do this” BASIC, Assembly	Declarative – “I need this” SQL
Structured – Subroutines, scopes C, FORTRAN77 algorithms + data : good for explicit computing	Functional – Pure functions, lazy evaluation Haskell, Scala functions o functions : good for reasoning
Object-Oriented – Encapsulation, Inheritance, ... C++, Python objects + messages : good for modeling	Logic – Predicates and rules Prolog, Datalog facts + rules : good for searching

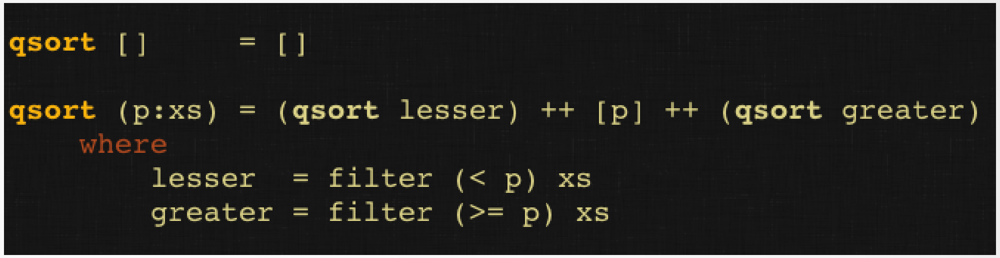
1. Programming language

C



```c
void f(int a[], int lo, int hi)
{
  int h, l, p, t;

  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] <= p))
          l = l+1;
      while ((h > l) && (a[h] >= p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    f( a, lo, l-1 );
    f( a, l+1, hi );
  }
}
```

Haskell



```haskell
qsort []     = []

qsort (p:xs) = (qsort lesser) ++ [p] ++ (qsort greater)
    where
        lesser  = filter (< p) xs
        greater = filter (>= p) xs
```

Purely functional
Static strong typing
Lazy evaluation

2. Good practices

- Write for humans, not for computers
- Use the appropriate language(s)
- Organize for change, and make incremental changes
- Plan for mistakes, automate testing
- Automate repetitive tasks
- Use modern source-code management system
- Document the design and purpose, not the implementation
- Optimize only when it works already

2. Good practices

Paul F. Dubois. 1999. Ten Good Practices in Scientific Programming. *Computing in Science and Engg.* 1, 1 (January 1999), 7-11. DOI=10.1109/MCISE.1999.743610 http://dx.doi.org/10.1109/MCISE.1999.743610

Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, et al. (2014) Best Practices for Scientific Computing. PLoS Biol 12(1): e1001745. doi:10.1371/journal.pbio.1001745

Dubois PF, Epperly T, Kumfert G (2003) Why Johnny can't build (portable scientific software). Comput Sci Eng 5: 83–88. doi: 10.1109/mcise.2003.1225867

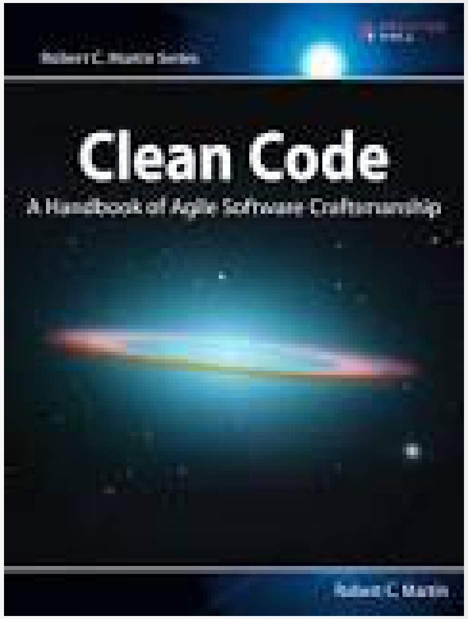
Prlić A, Procter JB (2012) Ten Simple Rules for the Open Development of Scientific Software. PLoS Comput Biol 8(12): e1002802. doi:10.1371/journal.pcbi.1002802

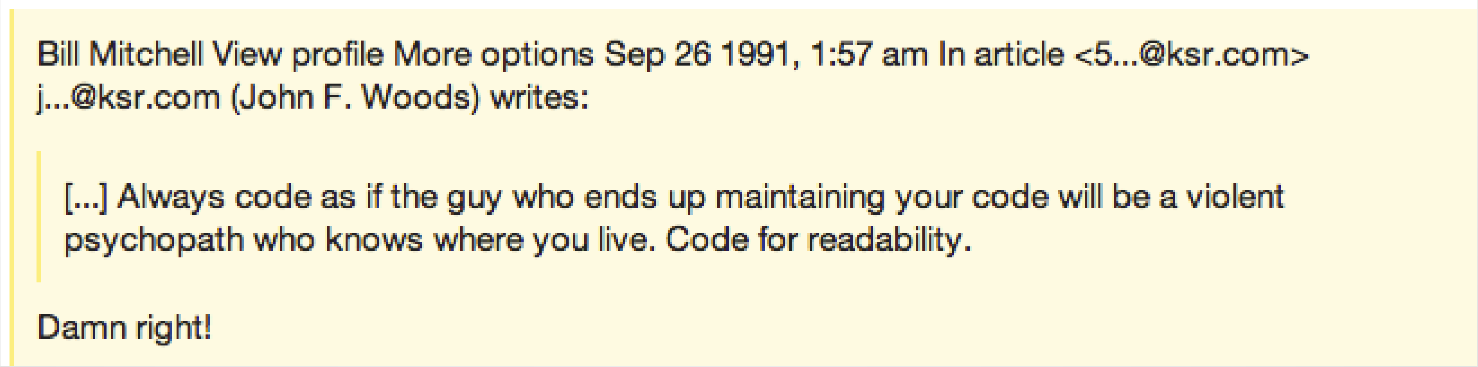
Victor R. Basili, Jeffrey C. Carver, Daniela Cruzes, Lorin M. Hochstein, Jeffrey K. Hollingsworth, Forrest Shull, Marvin V. Zelkowitz, "Understanding the High-Performance-Computing Community: A Software Engineer's Perspective," IEEE Software, vol. 25, no. 4, pp. 29-36, July/August, 2008

Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017) Good enough practices in scientific computing. PLoS Comput Biol 13(6): e1005510. https://doi.org/10.1371/journal.pcbi.1005510

2. Good coding principles

- Don't repeat yourself (DRY)
- Keep it simple, Stupid (KISS)
- One level of abstraction
- Single responsibility principle
- Separation of concern
- Avoid premature optimization
- Many others...



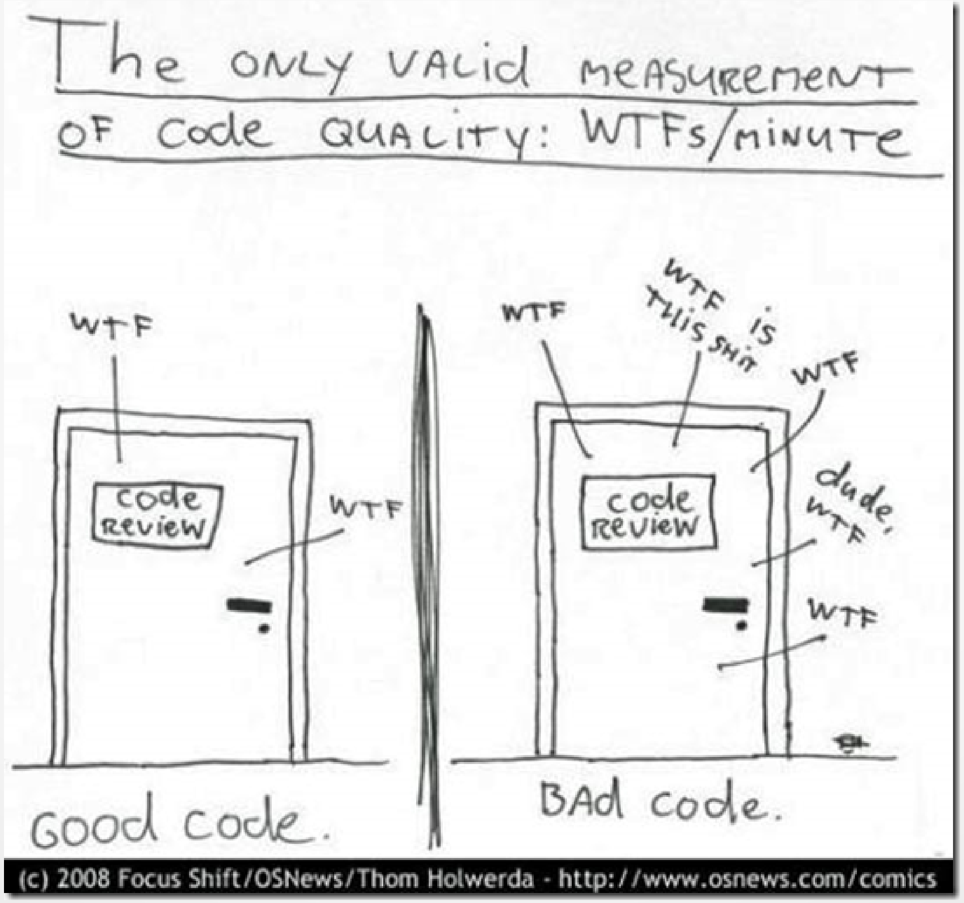

Bill Mitchell View profile More options Sep 26 1991, 1:57 am In article <5...@ksr.com> j...@ksr.com (John F. Woods) writes:

> [...] Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. Code for readability.

Damn right!

Clean Code: A Handbook of Agile Software Craftsmanship, R. C. Martin, Prentice Hall, 2008

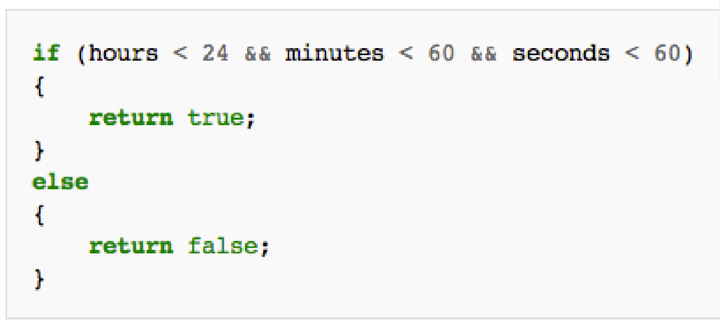
How to measure code quality



(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

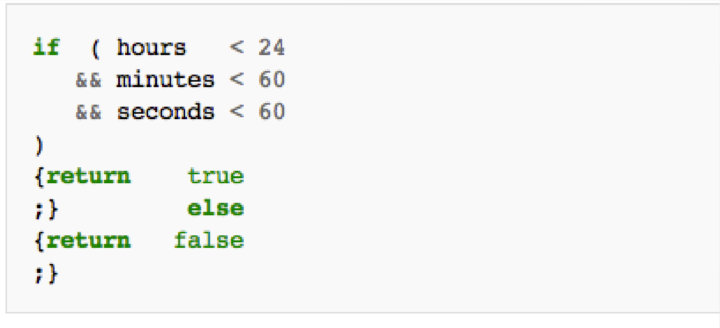
2. Good style

- Makes sure the code is readable by all
  - easily
  - quickly
- Depends on
  - the language
  - the project



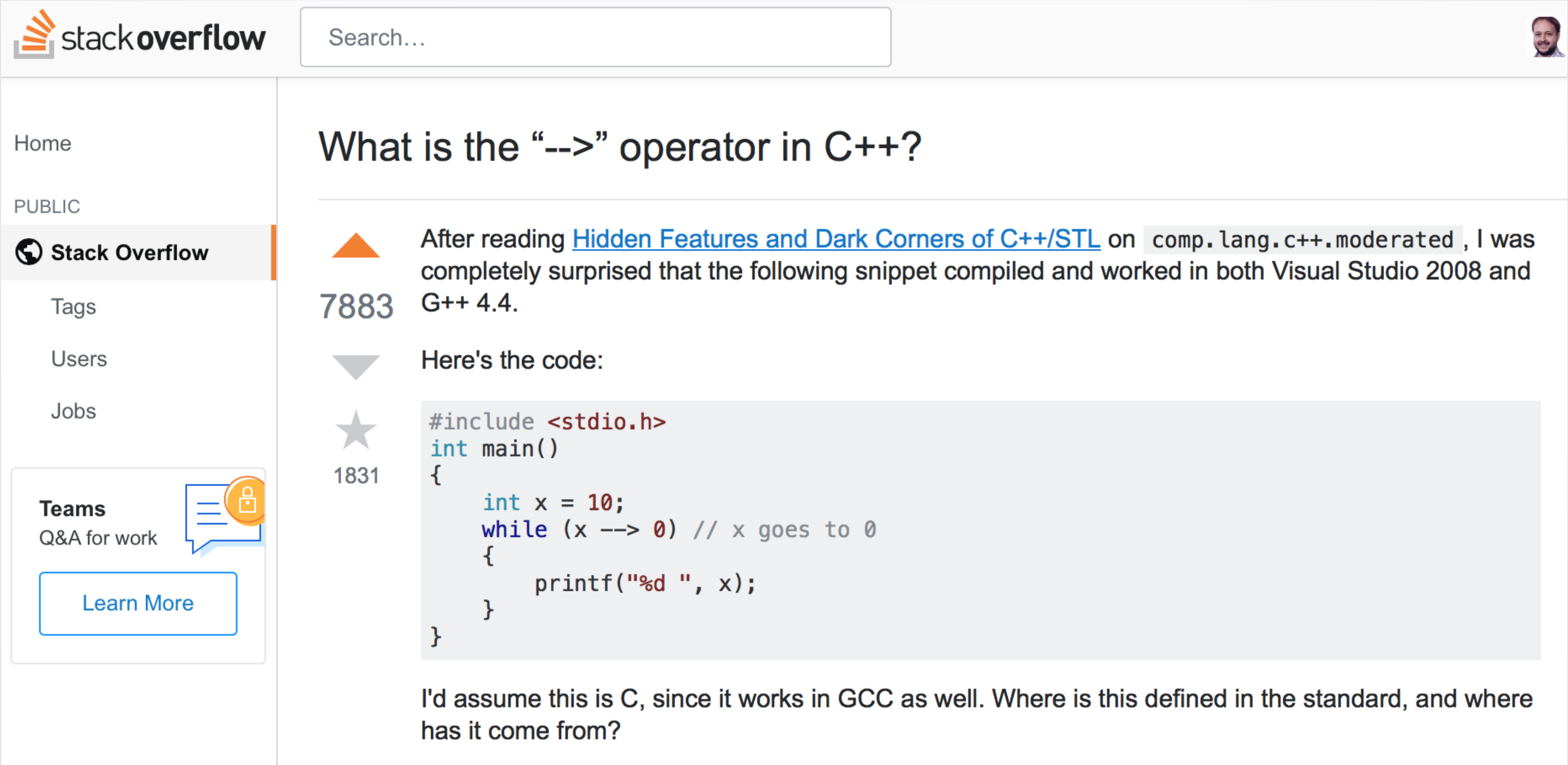
```
if (hours < 24 && minutes < 60 && seconds < 60)
{
    return true;
}
else
{
    return false;
}
```

vs



```
if  ( hours   < 24
   && minutes < 60
   && seconds < 60
)
{return    true
;}         else
{return   false
;}
```

2. Good style



What is the “-->” operator in C++?

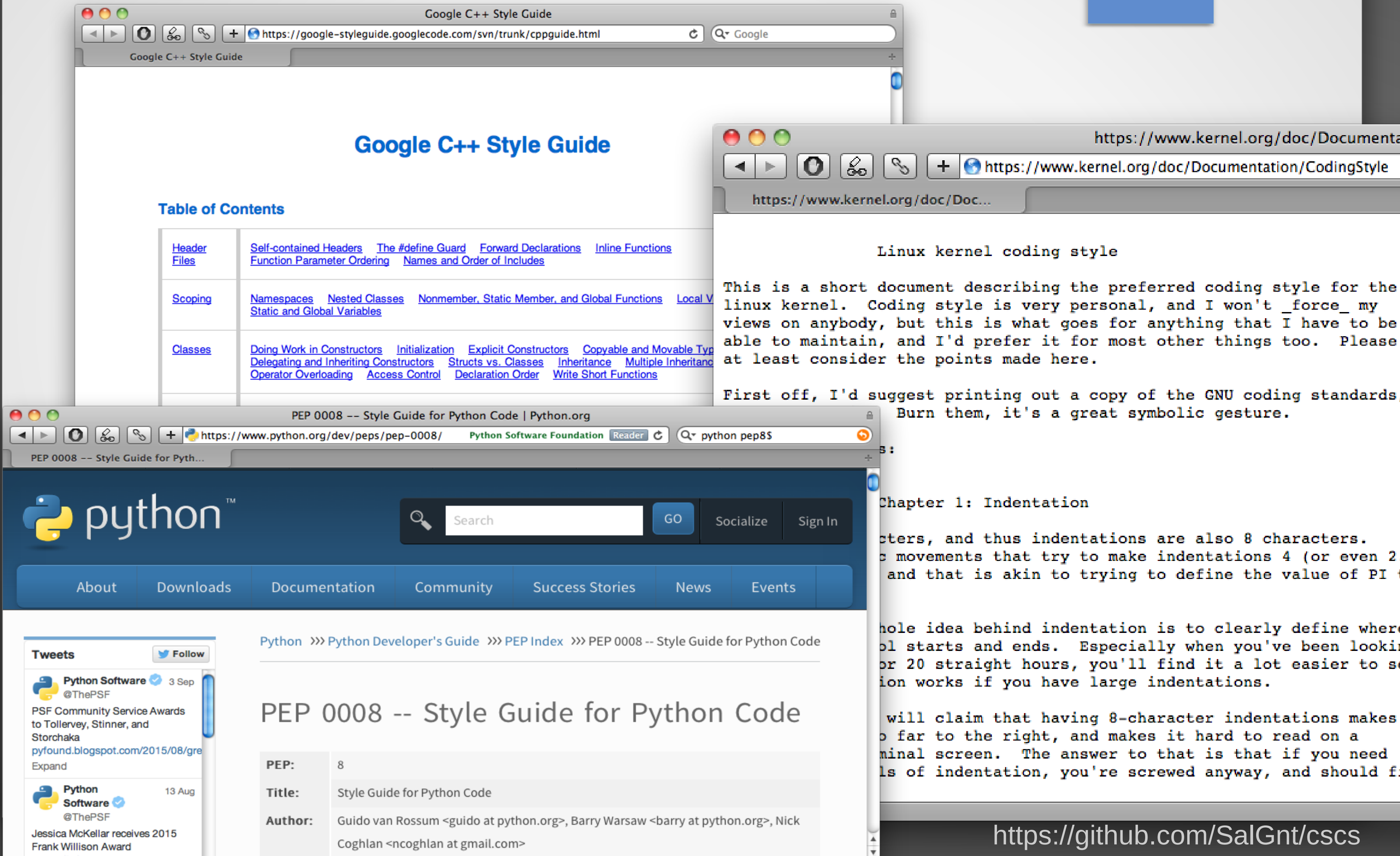
After reading Hidden Features and Dark Corners of C++/STL on `comp.lang.c++.moderated`, I was completely surprised that the following snippet compiled and worked in both Visual Studio 2008 and G++ 4.4.

Here's the code:

```
#include <stdio.h>
int main()
{
    int x = 10;
    while (x --> 0) // x goes to 0
    {
        printf("%d ", x);
    }
}
```

I'd assume this is C, since it works in GCC as well. Where is this defined in the standard, and where has it come from?

2. Good style



Google C++ Style Guide

https://google-styleguide.googlecode.com/svn/trunk/cppguide.html

Google C++ Style Guide

Table of Contents

Header Files	Self-contained Headers The #define Guard Forward Declarations Inline Functions Function Parameter Ordering Names and Order of Includes
Scoping	Namespaces Nested Classes Nonmember, Static Member, and Global Functions Local V Static and Global Variables
Classes	Doing Work in Constructors Initialization Explicit Constructors Copyable and Movable Typ Delegating and Inheriting Constructors Structs vs. Classes Inheritance Multiple Inheritanc Operator Overloading Access Control Declaration Order Write Short Functions

https://www.kernel.org/doc/Documenta

https://www.kernel.org/doc/Documentation/CodingStyle

https://www.kernel.org/doc/Doc...

```
                Linux kernel coding style

This is a short document describing the preferred coding style for the
linux kernel.  Coding style is very personal, and I won't _force_ my
views on anybody, but this is what goes for anything that I have to be
able to maintain, and I'd prefer it for most other things too.  Please
at least consider the points made here.

First off, I'd suggest printing out a copy of the GNU coding standards
    Burn them, it's a great symbolic gesture.

s:


Chapter 1: Indentation

cters, and thus indentations are also 8 characters.
c movements that try to make indentations 4 (or even 2
 and that is akin to trying to define the value of PI

hole idea behind indentation is to clearly define where
ol starts and ends.  Especially when you've been looki
or 20 straight hours, you'll find it a lot easier to se
ion works if you have large indentations.

 will claim that having 8-character indentations makes
o far to the right, and makes it hard to read on a
minal screen.  The answer to that is that if you need
ls of indentation, you're screwed anyway, and should f
```

PEP 0008 -- Style Guide for Python Code | Python.org

https://www.python.org/dev/peps/pep-0008/ Python Software Foundation Reader

python pep8$

PEP 0008 -- Style Guide for Pyth...

Tweets Follow

Python Software @ThePSF 3 Sep
PSF Community Service Awards to Tollervey, Stinner, and Storchaka
pyfound.blogspot.com/2015/08/gre
Expand

Python Software @ThePSF 13 Aug
Jessica McKellar receives 2015 Frank Willison Award

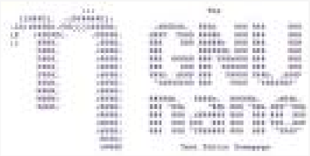
Python ››› Python Developer's Guide ››› PEP Index ››› PEP 0008 -- Style Guide for Python Code

PEP 0008 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

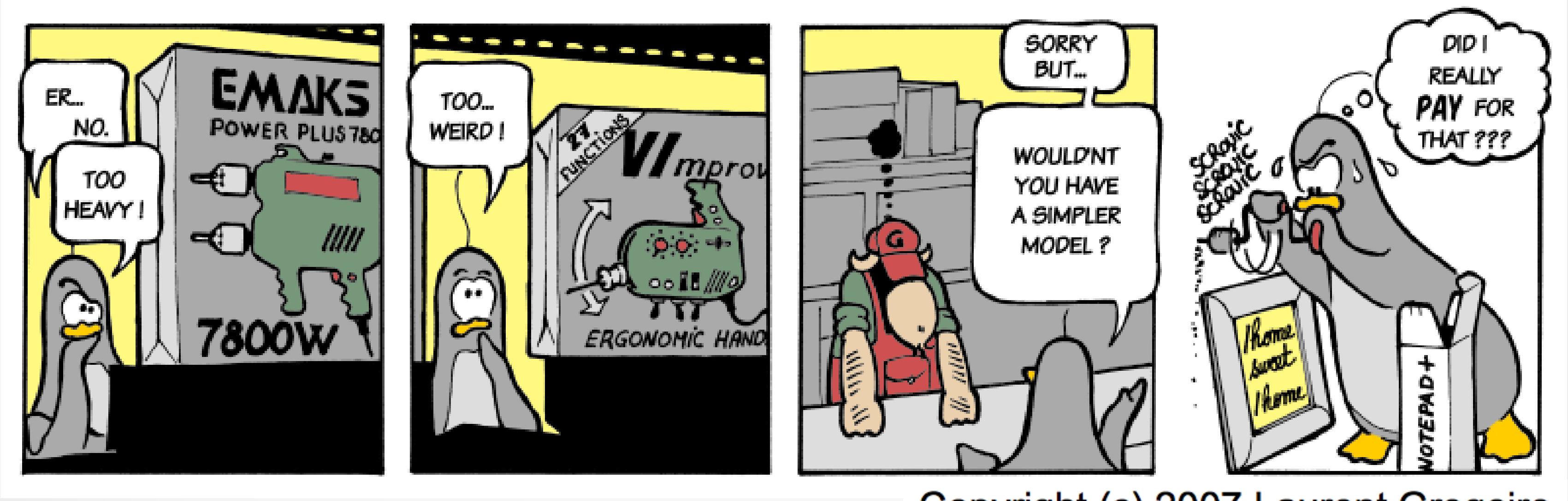
3. Text editor

- Some files are better edited directly on the clusters;



- If you prefer a graphical user interface, some good candidates are:
  - Sublime text: http://www.sublimetext.com/
  - Notepad++: https://notepad-plus-plus.org/
  - Text Wrangler: http://www.barebones.com/
  - Textmate: https://macromates.com/
  - Atom: https://atom.io/
- Choose one and learn it from inside out

3. Text editor



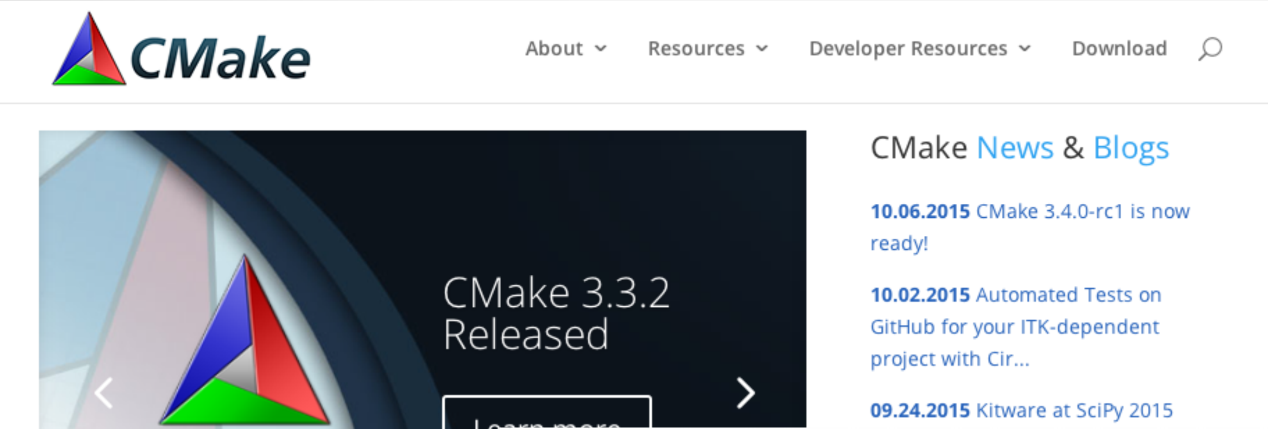
Copyright (c) 2007 Laurent Gregoire

Dev's toolkit :

1. Programming language
2. Good practices / Code Style Guides
3. Text editor / IDE
4. **Source control management**
5. **Debuggers / Profilers**
6. **Databases**
7. Packaging / Distributing tools
8. Comments and documentation
9. Tests
10. Licensing

Own dedicated sessions

7. Packaging Fortran/C/C++ code



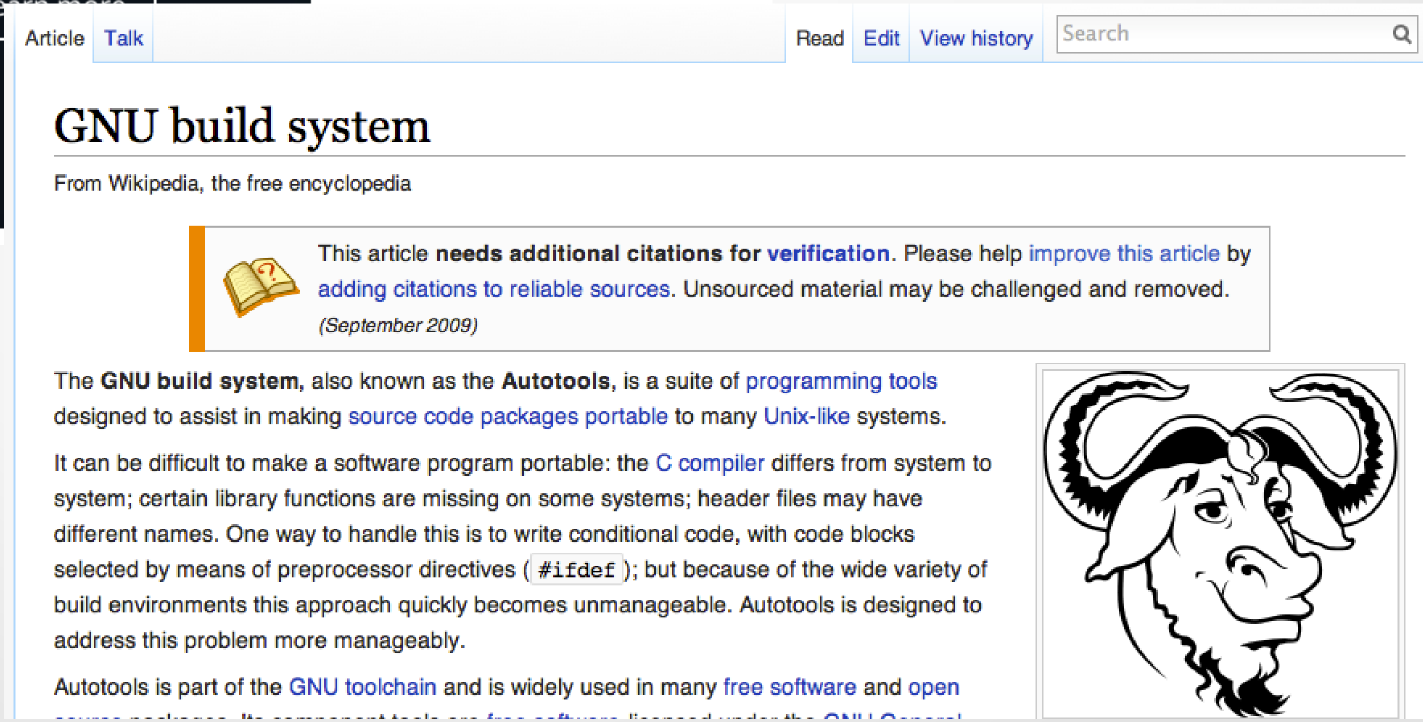
CMake News & Blogs

10.06.2015 CMake 3.4.0-rc1 is now ready!

10.02.2015 Automated Tests on GitHub for your ITK-dependent project with Cir...

09.24.2015 Kitware at SciPy 2015

Making sure it compiles on your laptop is not enough



GNU build system

From Wikipedia, the free encyclopedia

> This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(September 2009)*

The **GNU build system**, also known as the **Autotools**, is a suite of programming tools designed to assist in making source code packages portable to many Unix-like systems.

It can be difficult to make a software program portable: the C compiler differs from system to system; certain library functions are missing on some systems; header files may have different names. One way to handle this is to write conditional code, with code blocks selected by means of preprocessor directives (`#ifdef`); but because of the wide variety of build environments this approach quickly becomes unmanageable. Autotools is designed to address this problem more manageably.

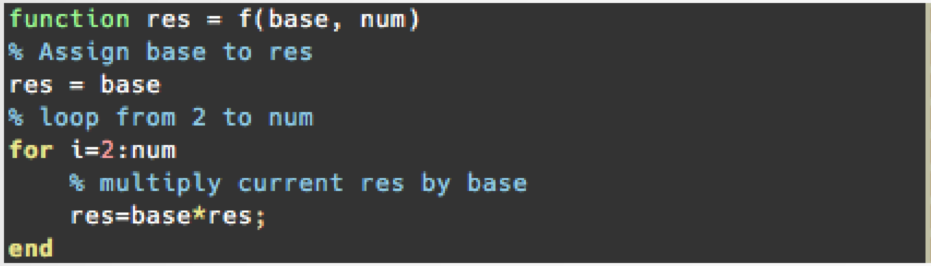
Autotools is part of the GNU toolchain and is widely used in many free software and open

It has to compile on all the clusters...

https://cmake.org/ , https://en.wikipedia.org/wiki/GNU_build_system

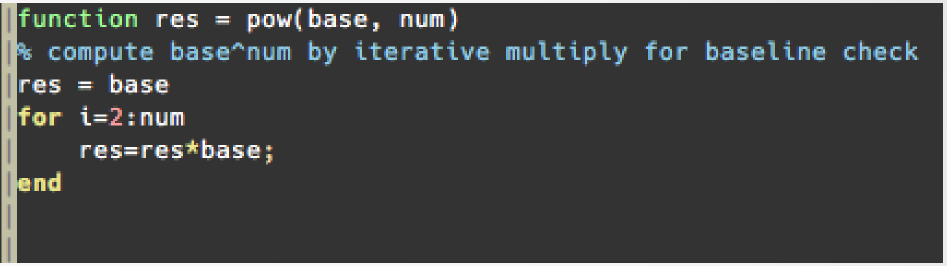
8. Comments / Documentation

Lots of useless comments



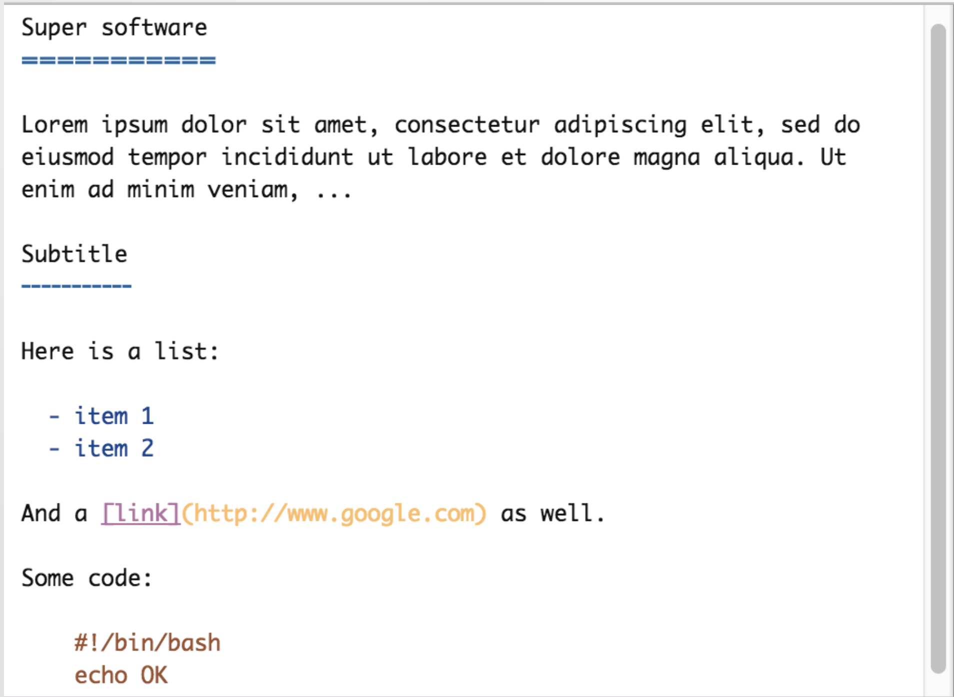
```
function res = f(base, num)
% Assign base to res
res = base
% loop from 2 to num
for i=2:num
    % multiply current res by base
    res=base*res;
end
```

Less comments but useful comments

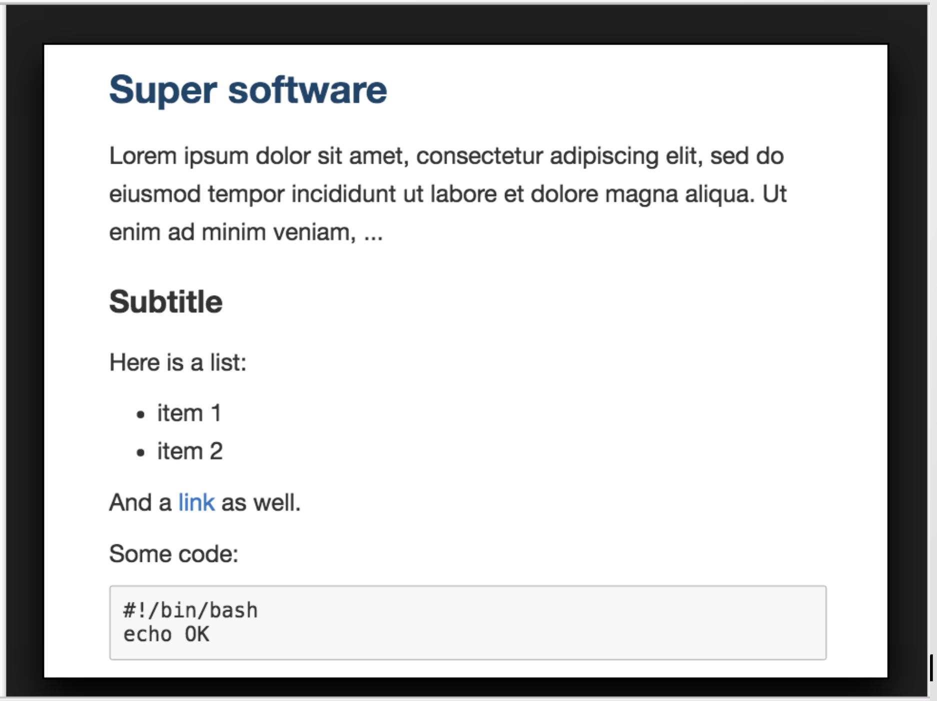


```
function res = pow(base, num)
% compute base^num by iterative multiply for baseline check
res = base
for i=2:num
    res=res*base;
end
```

Write doc in a lightweight markup language (Markdown, rst, etc.)



```
Super software
==============

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad minim veniam, ...

Subtitle
--------

Here is a list:

  - item 1
  - item 2

And a [link](http://www.google.com) as well.

Some code:

    #!/bin/bash
    echo OK
```



Super software

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, ...

Subtitle

Here is a list:

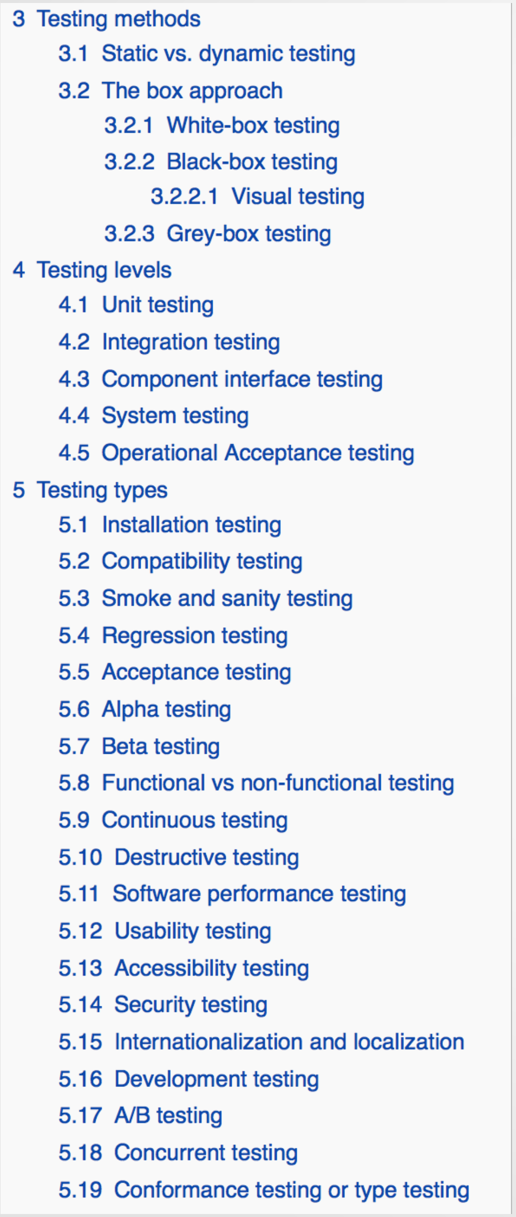
- item 1
- item 2

And a link as well.

Some code:

```
#!/bin/bash
echo OK
```

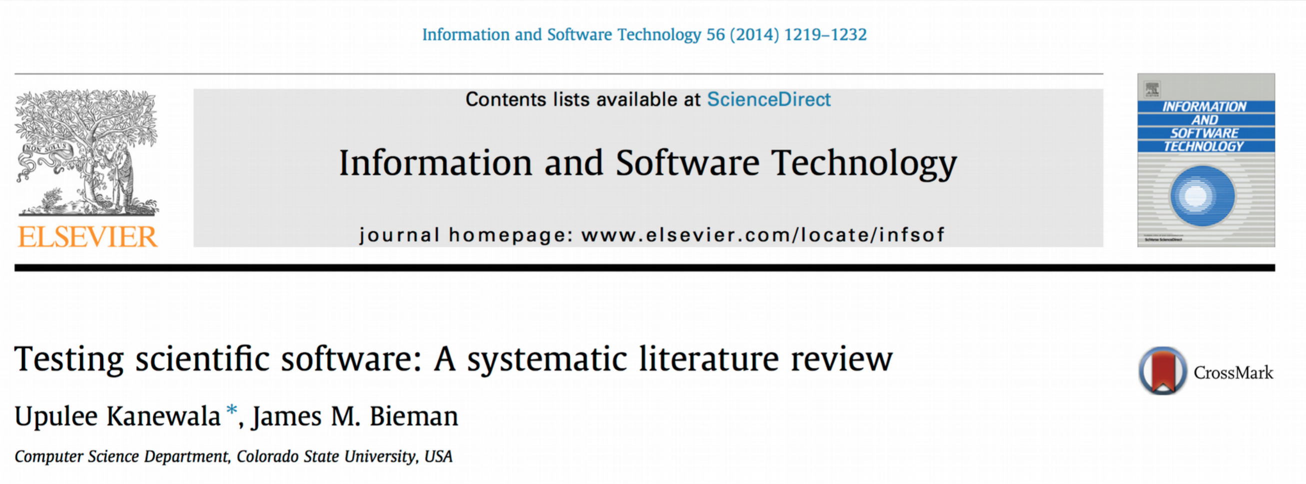
9. Tests - TDD



3 Testing methods
- 3.1 Static vs. dynamic testing
- 3.2 The box approach
  - 3.2.1 White-box testing
  - 3.2.2 Black-box testing
    - 3.2.2.1 Visual testing
  - 3.2.3 Grey-box testing

4 Testing levels
- 4.1 Unit testing
- 4.2 Integration testing
- 4.3 Component interface testing
- 4.4 System testing
- 4.5 Operational Acceptance testing

5 Testing types
- 5.1 Installation testing
- 5.2 Compatibility testing
- 5.3 Smoke and sanity testing
- 5.4 Regression testing
- 5.5 Acceptance testing
- 5.6 Alpha testing
- 5.7 Beta testing
- 5.8 Functional vs non-functional testing
- 5.9 Continuous testing
- 5.10 Destructive testing
- 5.11 Software performance testing
- 5.12 Usability testing
- 5.13 Accessibility testing
- 5.14 Security testing
- 5.15 Internationalization and localization
- 5.16 Development testing
- 5.17 A/B testing
- 5.18 Concurrent testing
- 5.19 Conformance testing or type testing



Information and Software Technology 56 (2014) 1219–1232

Contents lists available at ScienceDirect

Information and Software Technology

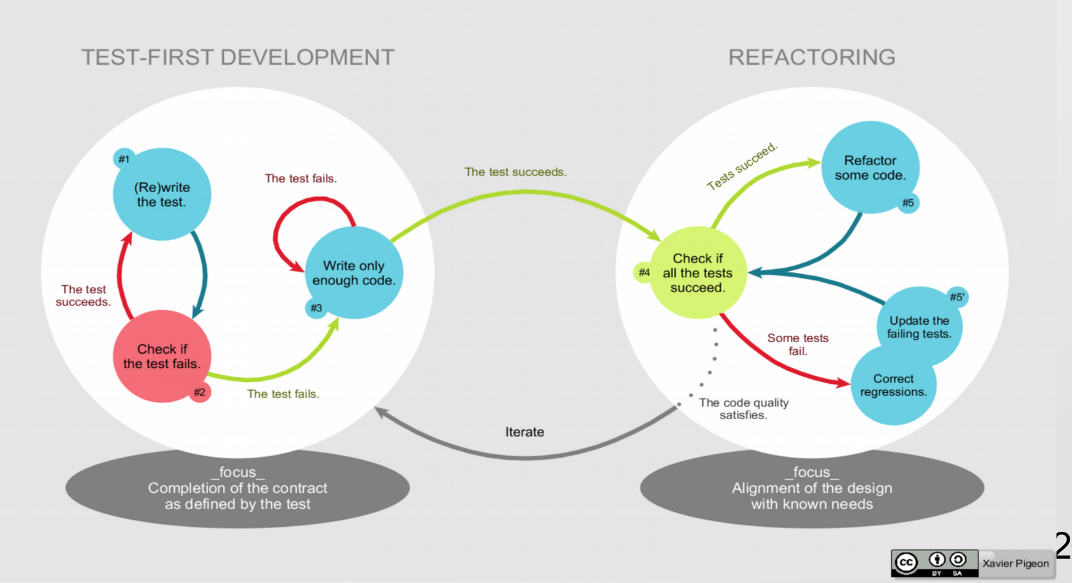
journal homepage: www.elsevier.com/locate/infsof

ELSEVIER

Testing scientific software: A systematic literature review

Upulee Kanewala *, James M. Bieman

Computer Science Department, Colorado State University, USA



http://dx.doi.org/10.1016/j.infsof.2014.05.006

10. Licensing your code: Why?

- **Commercial reason** :
  - you want to make money out of it – forbid distribution

    – forbid reverse engineering

- **Scientific reason** :
  - you want to it to be used and get citations
    - you need to allow usage, and/or modification, etc.
    - you require others to cite your work
  - you want to protect yourself from liability claims

10. Licensing your code: How?

- Choose a license type, e.g.
  - Apache License 2.0
  - BSD 3-Clause "New" or "Revised" license
  - BSD 2-Clause "Simplified" or "FreeBSD" license
  - GNU General Public License (GPL)
  - GNU Library or "Lesser" General Public License (LGPL)
  - MIT license
  - Mozilla Public License 2.0
  - Common Development and Distribution License
  - Eclipse Public License
- Copy/adapt the text
- Distribute a LICENSE file with your code

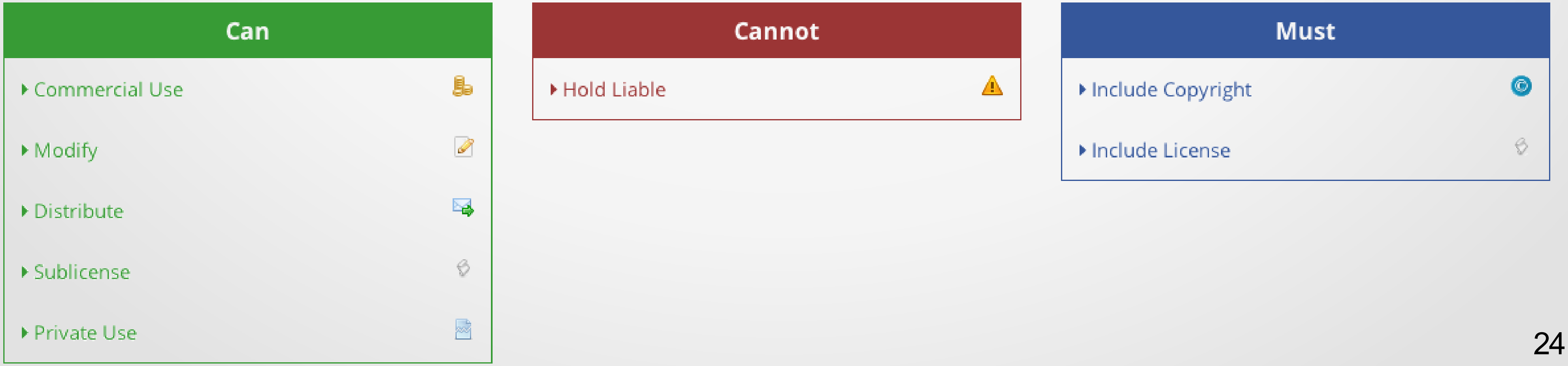
10. Licensing your code: MIT

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

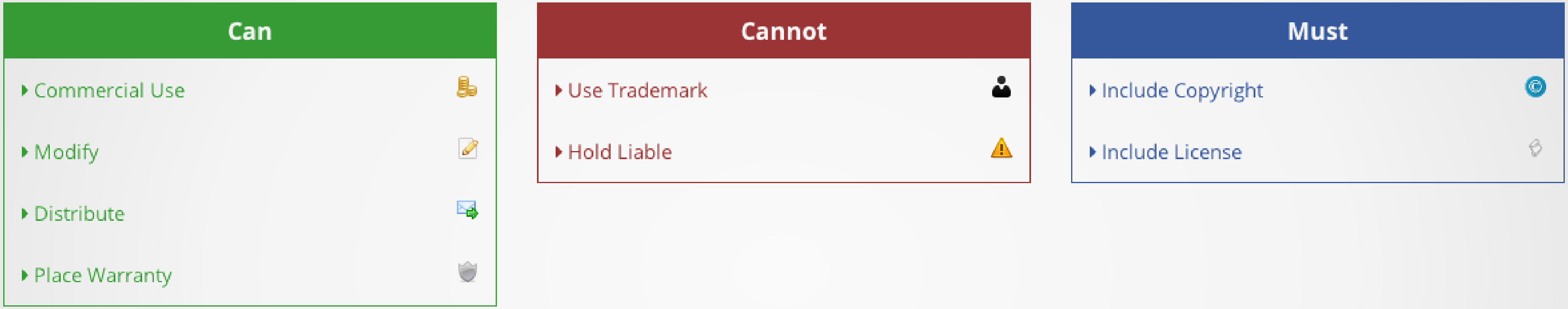
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



Can	Cannot	Must
Commercial Use	Hold Liable	Include Copyright
Modify		Include License
Distribute		
Sublicense		
Private Use		

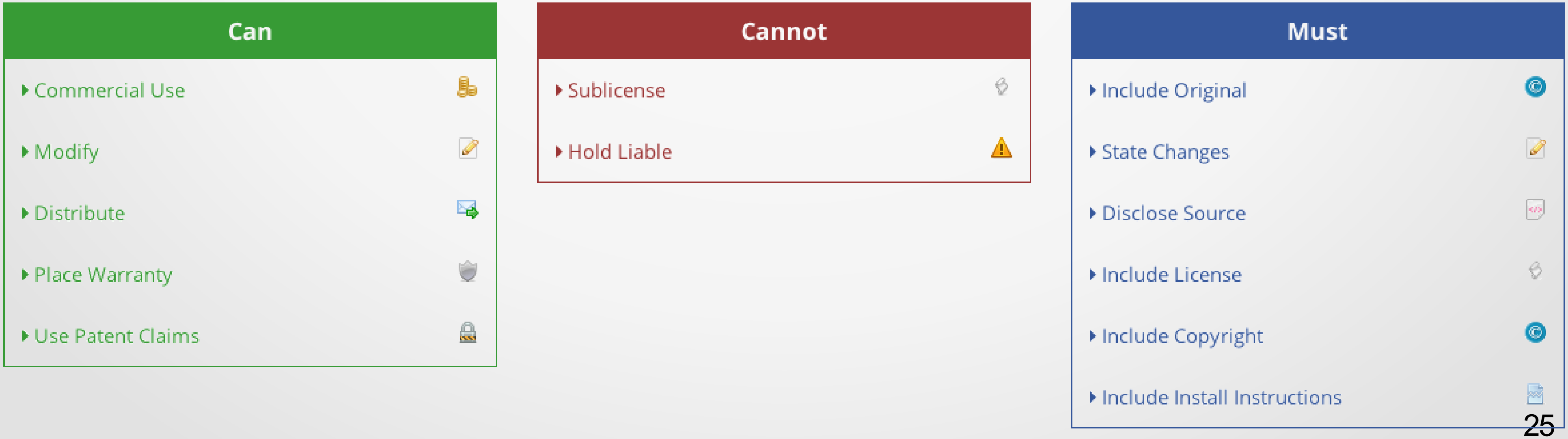
10. Licensing your code: BSD,GPL

BSD



Can	Cannot	Must
Commercial Use	Use Trademark	Include Copyright
Modify	Hold Liable	Include License
Distribute		
Place Warranty		

GPL



Can	Cannot	Must
Commercial Use	Sublicense	Include Original
Modify	Hold Liable	State Changes
Distribute		Disclose Source
Place Warranty		Include License
Use Patent Claims		Include Copyright
		Include Install Instructions

Reproduced from https://tldrlegal.com

Ops' toolkit :

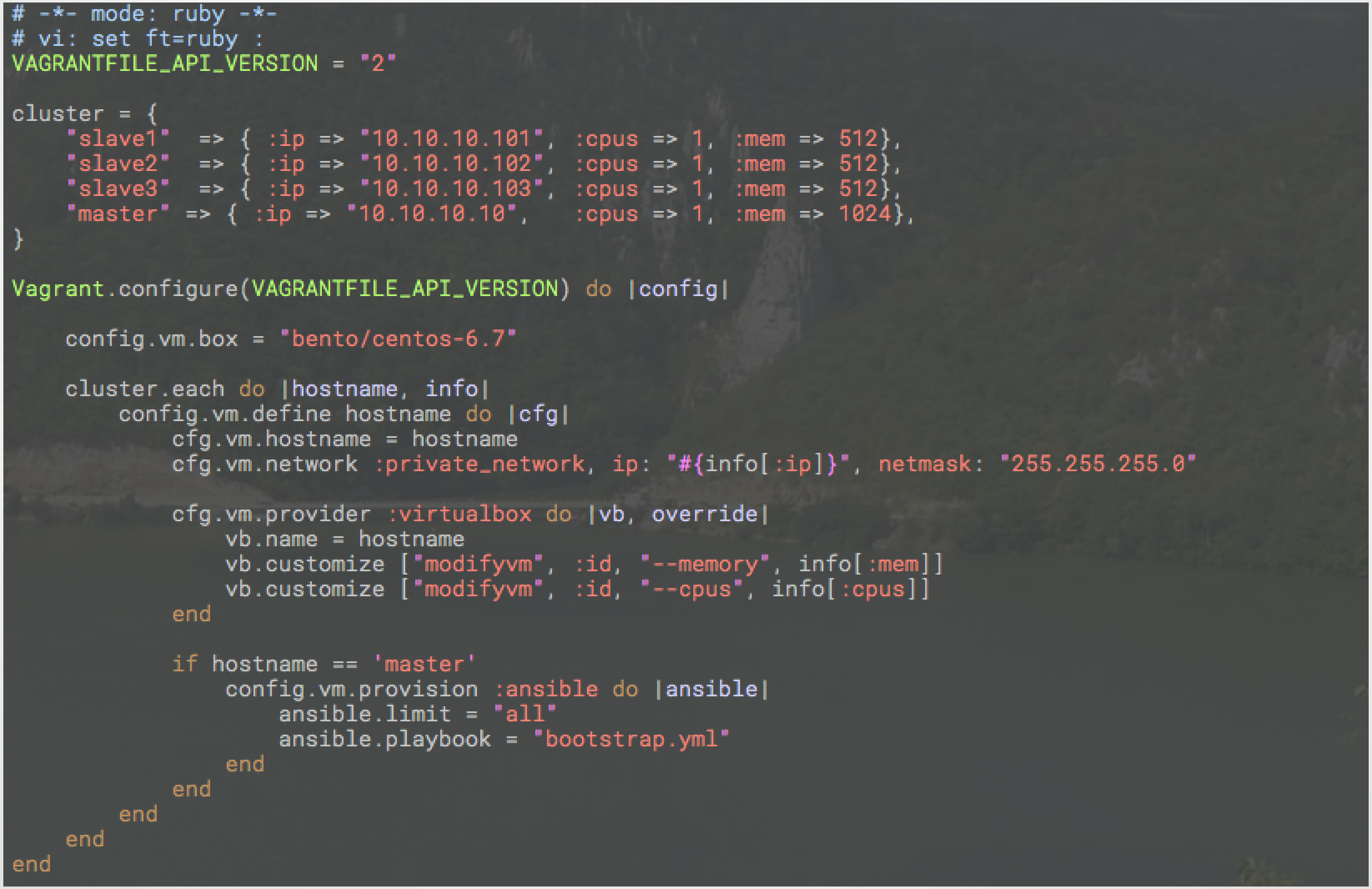
1. Virtualization platforms
2. Multi-host connexions
3. Configuration management
4. Installing
5. Automatic build tests
6. Monitoring

1. Virtualization

- Install on your laptop an environment similar to that of the cluster to test your workflow
- With
  - VirtualBox: https://www.virtualbox.org/
  - Vagrant: https://www.vagrantup.com/
- you can build a virtual cluster in one command:

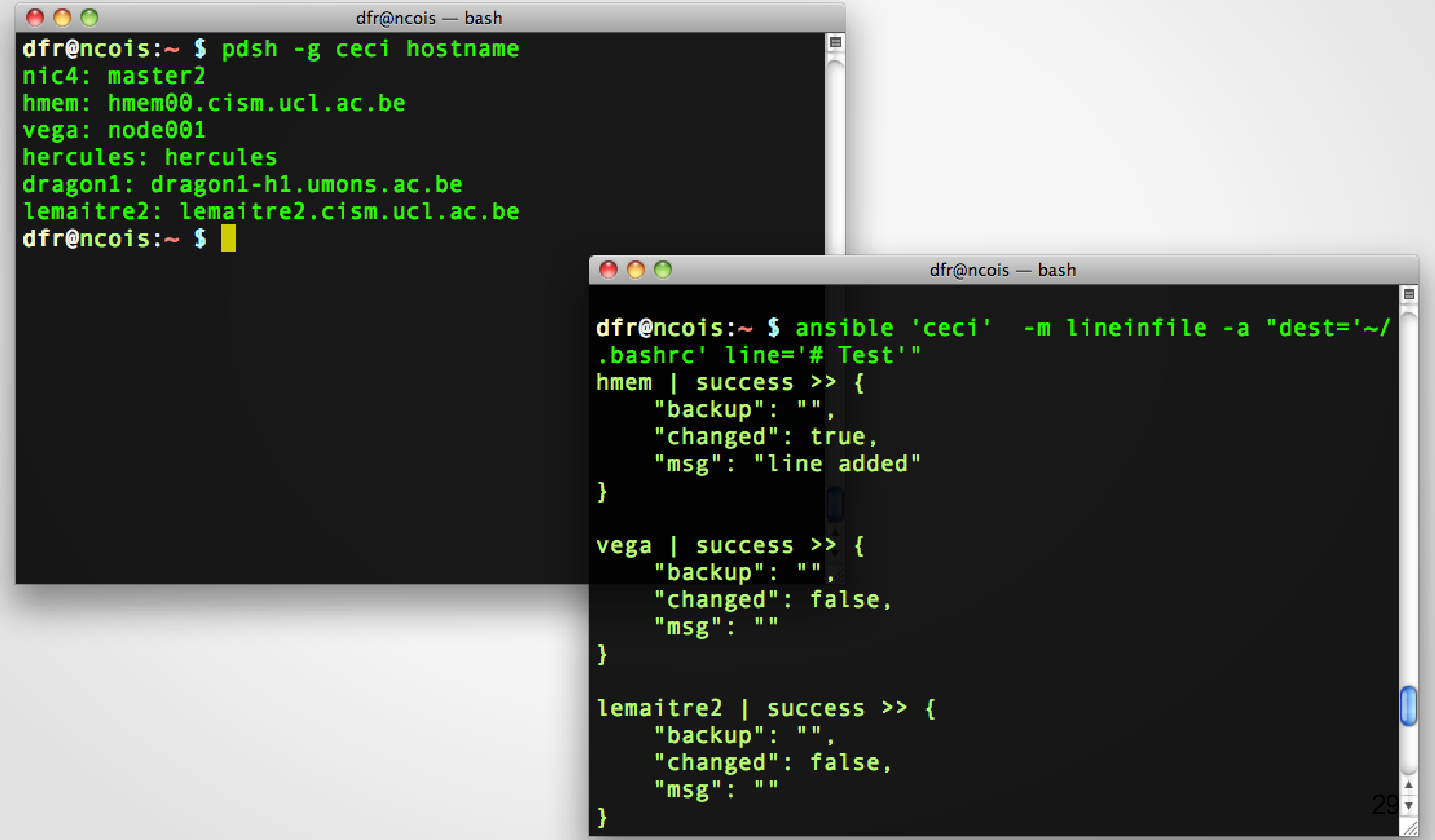
“vagrant up”

1. Virtualization



```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :
VAGRANTFILE_API_VERSION = "2"

cluster = {
    "slave1"  => { :ip => "10.10.10.101", :cpus => 1, :mem => 512},
    "slave2"  => { :ip => "10.10.10.102", :cpus => 1, :mem => 512},
    "slave3"  => { :ip => "10.10.10.103", :cpus => 1, :mem => 512},
    "master" => { :ip => "10.10.10.10",   :cpus => 1, :mem => 1024},
}

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

    config.vm.box = "bento/centos-6.7"

    cluster.each do |hostname, info|
        config.vm.define hostname do |cfg|
            cfg.vm.hostname = hostname
            cfg.vm.network :private_network, ip: "#{info[:ip]}", netmask: "255.255.255.0"

            cfg.vm.provider :virtualbox do |vb, override|
                vb.name = hostname
                vb.customize ["modifyvm", :id, "--memory", info[:mem]]
                vb.customize ["modifyvm", :id, "--cpus", info[:cpus]]
            end

            if hostname == 'master'
                config.vm.provision :ansible do |ansible|
                    ansible.limit = "all"
                    ansible.playbook = "bootstrap.yml"
                end
            end
        end
    end
end
```

2. Multi-host SSH

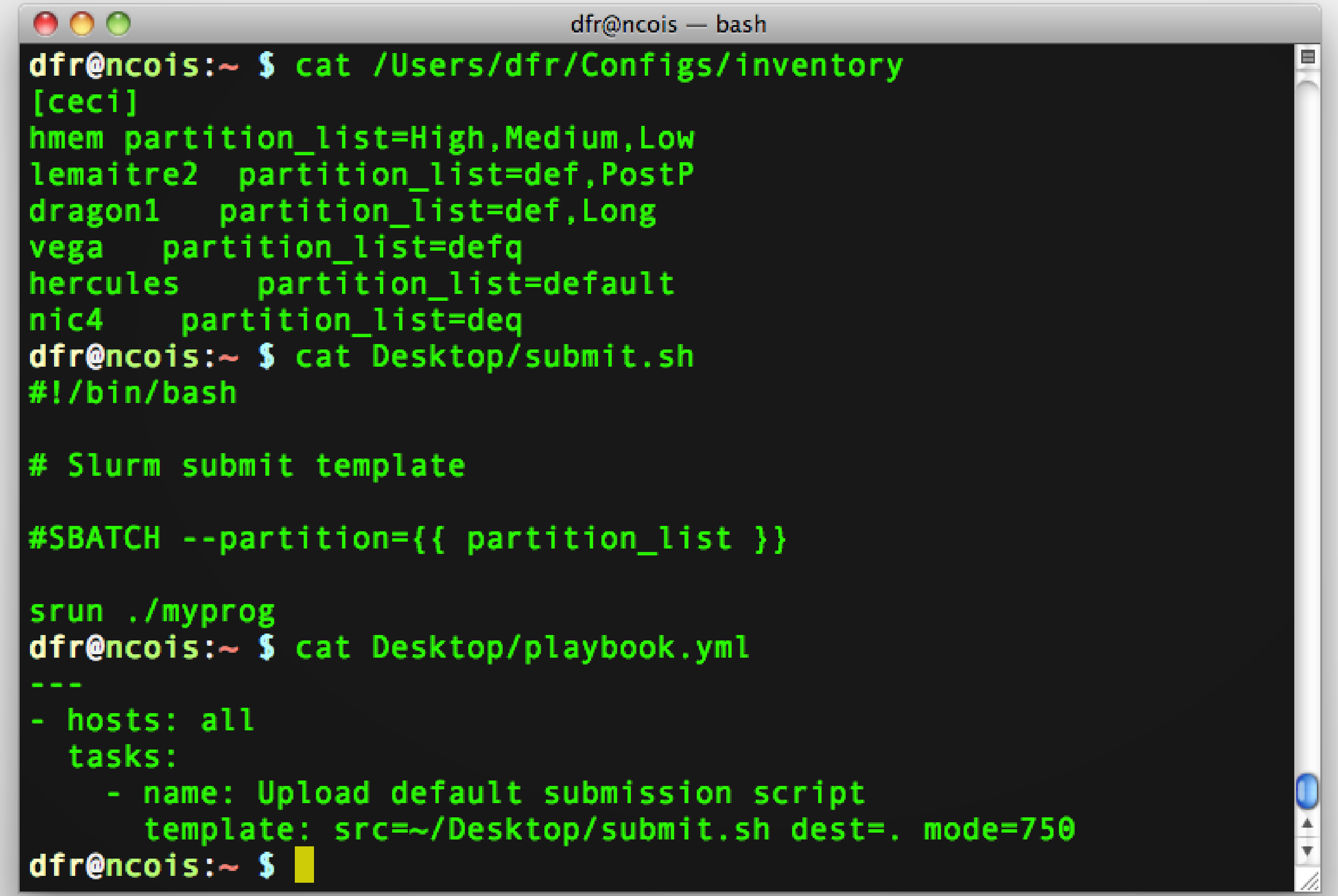


```
dfr@ncois:~ $ pdsh -g ceci hostname
nic4: master2
hmem: hmem00.cism.ucl.ac.be
vega: node001
hercules: hercules
dragon1: dragon1-h1.umons.ac.be
lemaitre2: lemaitre2.cism.ucl.ac.be
dfr@ncois:~ $ 
```

```
dfr@ncois:~ $ ansible 'ceci'  -m lineinfile -a "dest='~/
.bashrc' line='# Test'"
hmem | success >> {
    "backup": "",
    "changed": true,
    "msg": "line added"
}

vega | success >> {
    "backup": "",
    "changed": false,
    "msg": ""
}

lemaitre2 | success >> {
    "backup": "",
    "changed": false,
    "msg": ""
}
```

https://clustershell.readthedocs.io https://code.google.com/p/pdsh/, http://www.ansible.com/

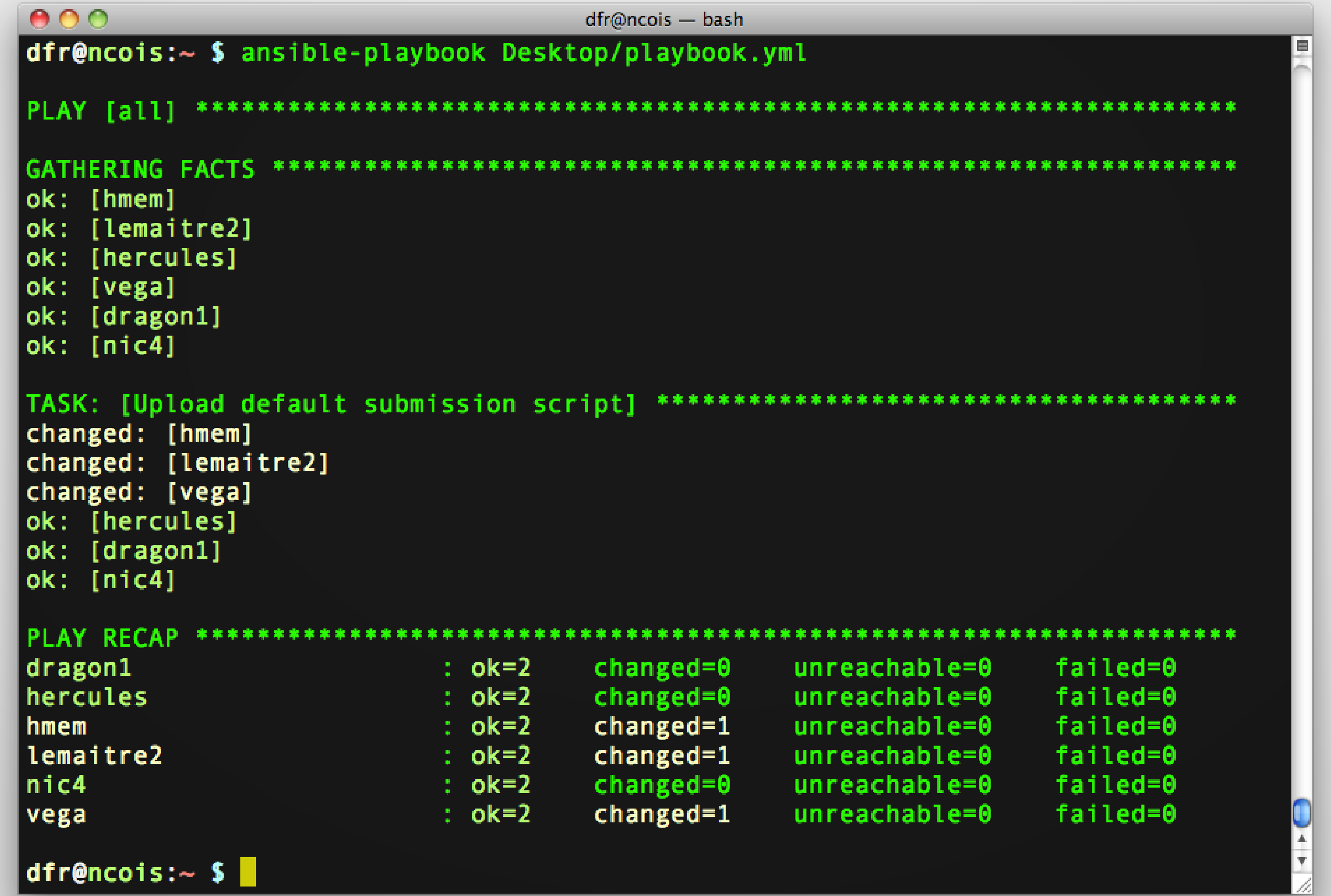
3. Configuration Management



```
dfr@ncois:~ $ cat /Users/dfr/Configs/inventory
[ceci]
hmem partition_list=High,Medium,Low
lemaitre2  partition_list=def,PostP
dragon1   partition_list=def,Long
vega   partition_list=defq
hercules    partition_list=default
nic4    partition_list=deq
dfr@ncois:~ $ cat Desktop/submit.sh
#!/bin/bash

# Slurm submit template

#SBATCH --partition={{ partition_list }}

srun ./myprog
dfr@ncois:~ $ cat Desktop/playbook.yml
---
- hosts: all
  tasks:
    - name: Upload default submission script
      template: src=~/Desktop/submit.sh dest=. mode=750
dfr@ncois:~ $ 
```

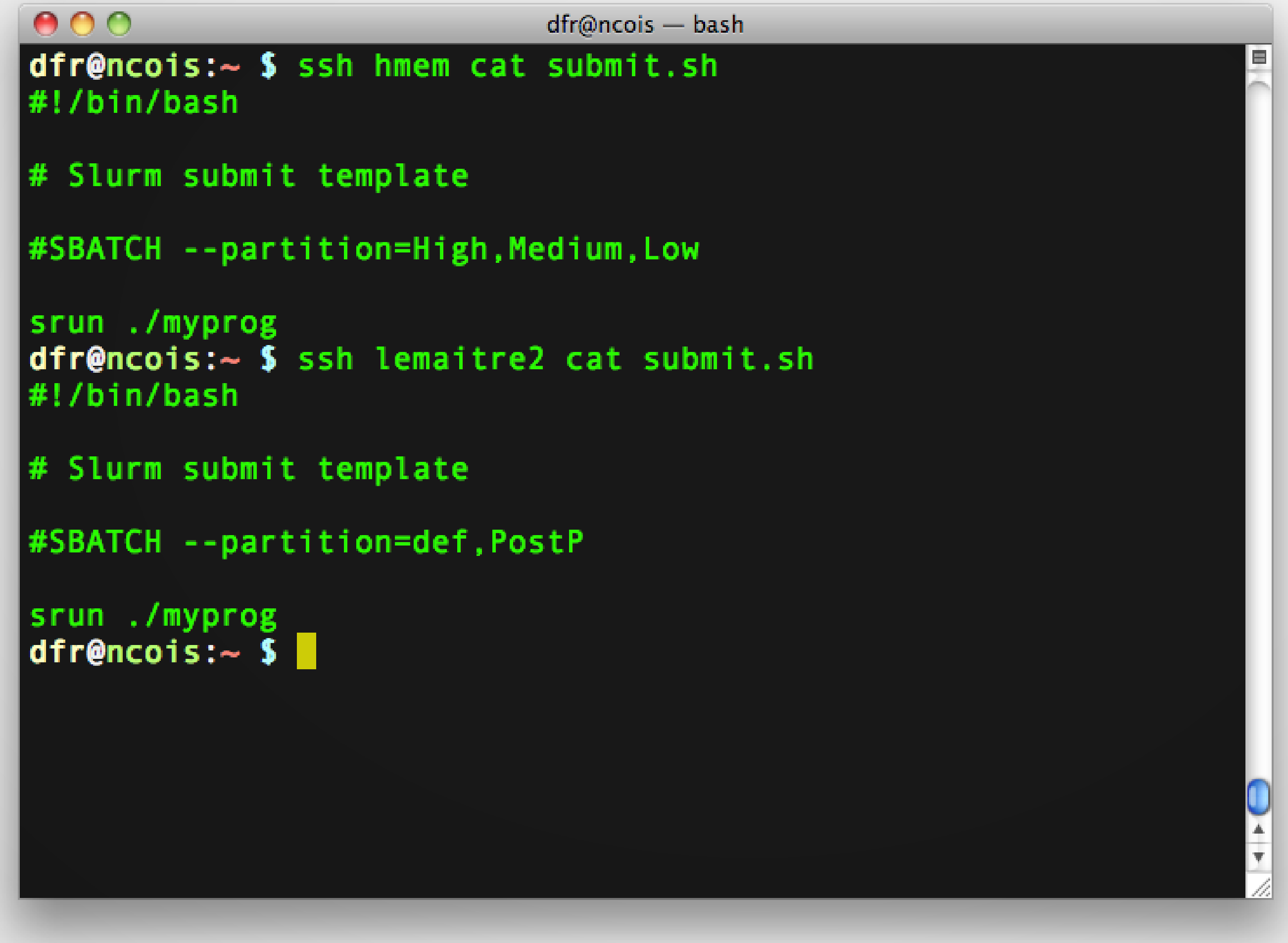
ANSIBLE

3. Configuration Management



```
dfr@ncois:~ $ ansible-playbook Desktop/playbook.yml

PLAY [all] ********************************************************************

GATHERING FACTS ***************************************************************
ok: [hmem]
ok: [lemaitre2]
ok: [hercules]
ok: [vega]
ok: [dragon1]
ok: [nic4]

TASK: [Upload default submission script] **************************************
changed: [hmem]
changed: [lemaitre2]
changed: [vega]
ok: [hercules]
ok: [dragon1]
ok: [nic4]

PLAY RECAP ********************************************************************
dragon1                    : ok=2    changed=0    unreachable=0    failed=0
hercules                   : ok=2    changed=0    unreachable=0    failed=0
hmem                       : ok=2    changed=1    unreachable=0    failed=0
lemaitre2                  : ok=2    changed=1    unreachable=0    failed=0
nic4                       : ok=2    changed=0    unreachable=0    failed=0
vega                       : ok=2    changed=1    unreachable=0    failed=0

dfr@ncois:~ $ 
```

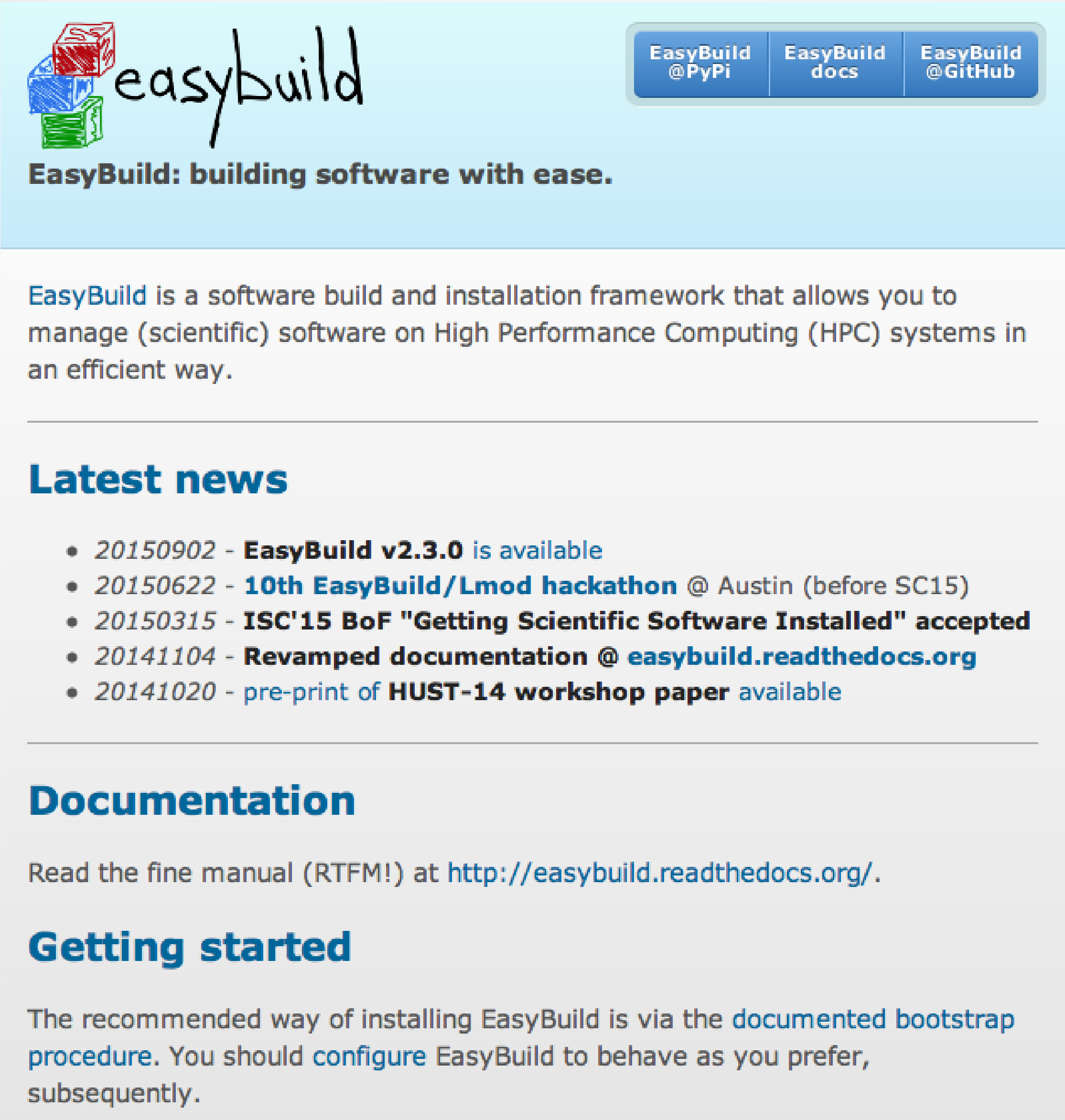
3. Configuration Management



dfr@ncois — bash

```
dfr@ncois:~ $ ssh hmem cat submit.sh
#!/bin/bash

# Slurm submit template

#SBATCH --partition=High,Medium,Low

srun ./myprog
dfr@ncois:~ $ ssh lemaitre2 cat submit.sh
#!/bin/bash

# Slurm submit template

#SBATCH --partition=def,PostP

srun ./myprog
dfr@ncois:~ $ 
```

4. Easy installing



EasyBuild: building software with ease.

EasyBuild is a software build and installation framework that allows you to manage (scientific) software on High Performance Computing (HPC) systems in an efficient way.

Latest news

- *20150902* - **EasyBuild v2.3.0** is available
- *20150622* - **10th EasyBuild/Lmod hackathon** @ Austin (before SC15)
- *20150315* - **ISC'15 BoF "Getting Scientific Software Installed" accepted**
- *20141104* - **Revamped documentation @ easybuild.readthedocs.org**
- *20141020* - pre-print of **HUST-14 workshop paper** available

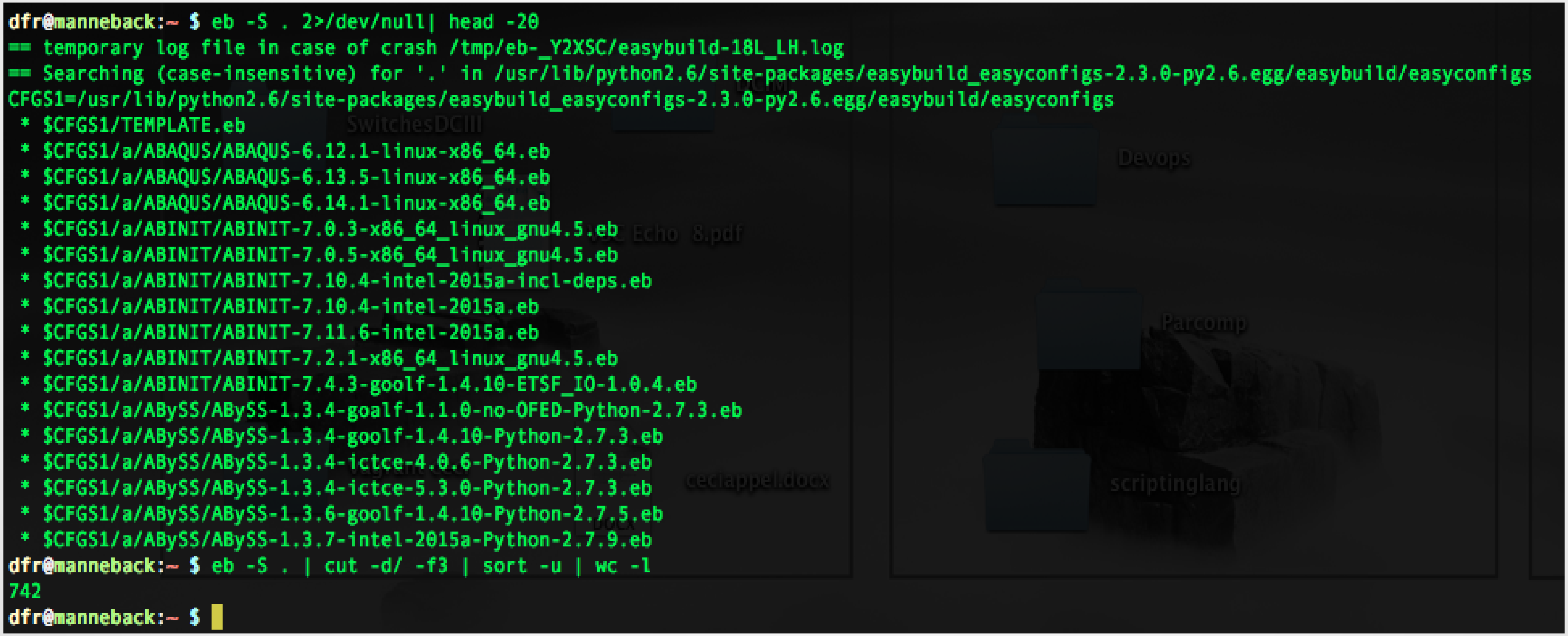
Documentation

Read the fine manual (RTFM!) at http://easybuild.readthedocs.org/.

Getting started

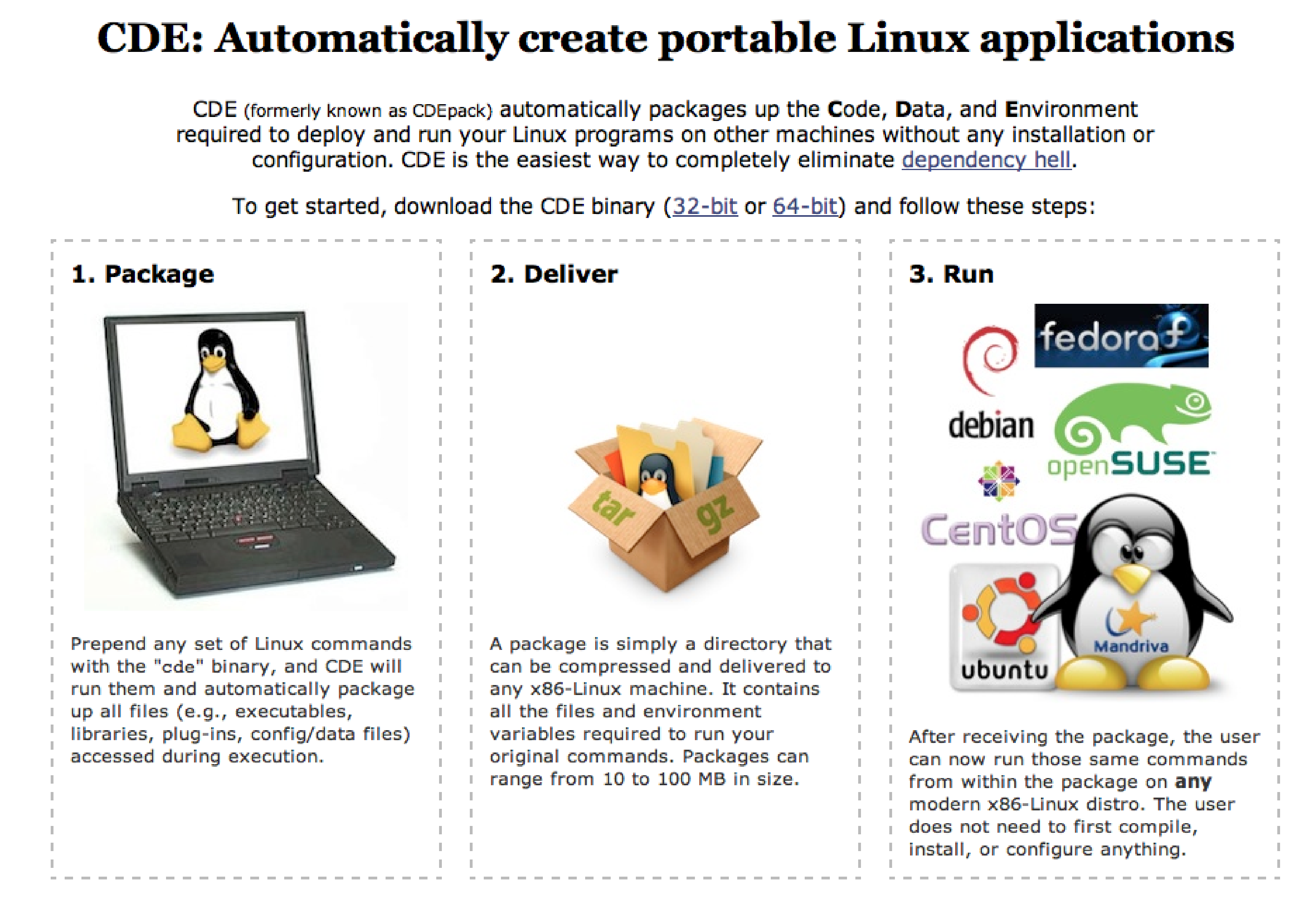
The recommended way of installing EasyBuild is via the documented bootstrap procedure. You should configure EasyBuild to behave as you prefer, subsequently.

4. Easy installing



```
dfr@manneback:~ $ eb -S . 2>/dev/null| head -20
== temporary log file in case of crash /tmp/eb-_Y2XSC/easybuild-18L_LH.log
== Searching (case-insensitive) for '.' in /usr/lib/python2.6/site-packages/easybuild_easyconfigs-2.3.0-py2.6.egg/easybuild/easyconfigs
CFGS1=/usr/lib/python2.6/site-packages/easybuild_easyconfigs-2.3.0-py2.6.egg/easybuild/easyconfigs
 * $CFGS1/TEMPLATE.eb
 * $CFGS1/a/ABAQUS/ABAQUS-6.12.1-linux-x86_64.eb
 * $CFGS1/a/ABAQUS/ABAQUS-6.13.5-linux-x86_64.eb
 * $CFGS1/a/ABAQUS/ABAQUS-6.14.1-linux-x86_64.eb
 * $CFGS1/a/ABINIT/ABINIT-7.0.3-x86_64_linux_gnu4.5.eb
 * $CFGS1/a/ABINIT/ABINIT-7.0.5-x86_64_linux_gnu4.5.eb
 * $CFGS1/a/ABINIT/ABINIT-7.10.4-intel-2015a-incl-deps.eb
 * $CFGS1/a/ABINIT/ABINIT-7.10.4-intel-2015a.eb
 * $CFGS1/a/ABINIT/ABINIT-7.11.6-intel-2015a.eb
 * $CFGS1/a/ABINIT/ABINIT-7.2.1-x86_64_linux_gnu4.5.eb
 * $CFGS1/a/ABINIT/ABINIT-7.4.3-goolf-1.4.10-ETSF_IO-1.0.4.eb
 * $CFGS1/a/ABySS/ABySS-1.3.4-goalf-1.1.0-no-OFED-Python-2.7.3.eb
 * $CFGS1/a/ABySS/ABySS-1.3.4-goolf-1.4.10-Python-2.7.3.eb
 * $CFGS1/a/ABySS/ABySS-1.3.4-ictce-4.0.6-Python-2.7.3.eb
 * $CFGS1/a/ABySS/ABySS-1.3.4-ictce-5.3.0-Python-2.7.3.eb
 * $CFGS1/a/ABySS/ABySS-1.3.6-goolf-1.4.10-Python-2.7.5.eb
 * $CFGS1/a/ABySS/ABySS-1.3.7-intel-2015a-Python-2.7.9.eb
dfr@manneback:~ $ eb -S . | cut -d/ -f3 | sort -u | wc -l
742
dfr@manneback:~ $ 
```

4. Easy installing



CDE: Automatically create portable Linux applications

CDE (formerly known as CDEpack) automatically packages up the **C**ode, **D**ata, and **E**nvironment required to deploy and run your Linux programs on other machines without any installation or configuration. CDE is the easiest way to completely eliminate dependency hell.

To get started, download the CDE binary (32-bit or 64-bit) and follow these steps:

1. Package

Prepend any set of Linux commands with the "cde" binary, and CDE will run them and automatically package up all files (e.g., executables, libraries, plug-ins, config/data files) accessed during execution.

2. Deliver

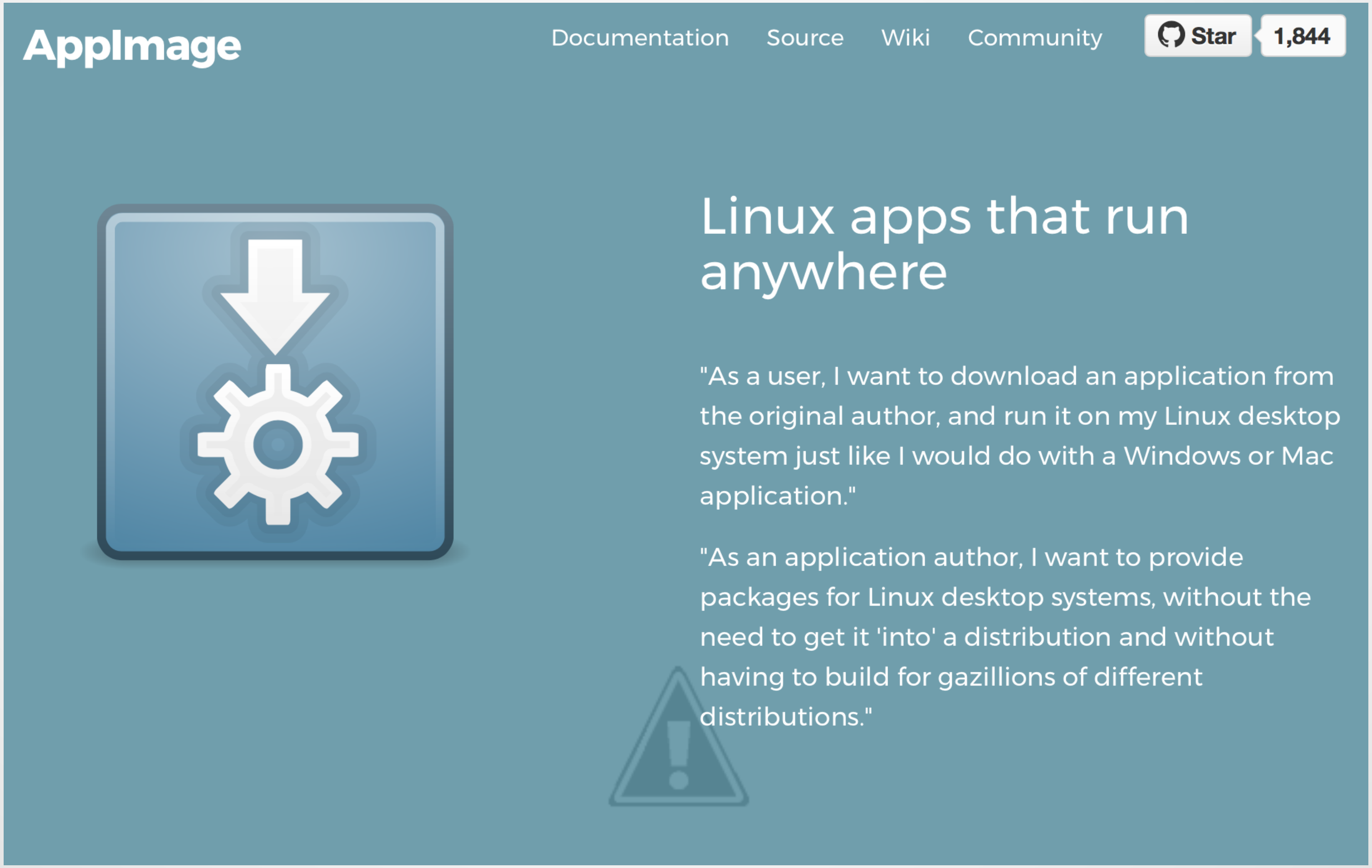
A package is simply a directory that can be compressed and delivered to any x86-Linux machine. It contains all the files and environment variables required to run your original commands. Packages can range from 10 to 100 MB in size.

3. Run

After receiving the package, the user can now run those same commands from within the package on **any** modern x86-Linux distro. The user does not need to first compile, install, or configure anything.

http://www.pgbovine.net/cde.html

4. Easy installing



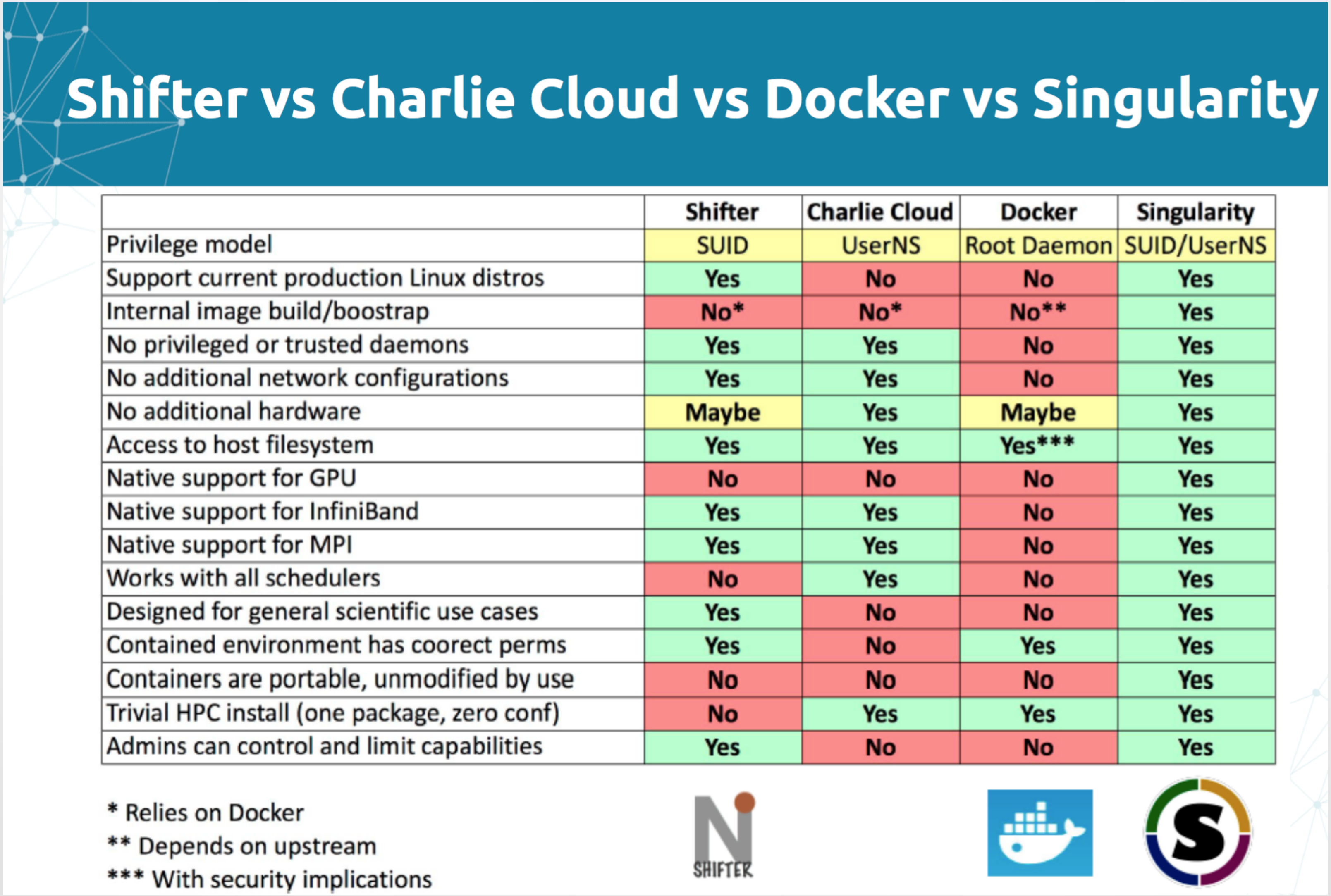
Linux apps that run anywhere

"As a user, I want to download an application from the original author, and run it on my Linux desktop system just like I would do with a Windows or Mac application."

"As an application author, I want to provide packages for Linux desktop systems, without the need to get it 'into' a distribution and without having to build for gazillions of different distributions."

http://appimage.org

4. Easy installing



Shifter vs Charlie Cloud vs Docker vs Singularity

	Shifter	Charlie Cloud	Docker	Singularity
Privilege model	SUID	UserNS	Root Daemon	SUID/UserNS
Support current production Linux distros	Yes	No	No	Yes
Internal image build/boostrap	No*	No*	No**	Yes
No privileged or trusted daemons	Yes	Yes	No	Yes
No additional network configurations	Yes	Yes	No	Yes
No additional hardware	Maybe	Yes	Maybe	Yes
Access to host filesystem	Yes	Yes	Yes***	Yes
Native support for GPU	No	No	No	Yes
Native support for InfiniBand	Yes	Yes	No	Yes
Native support for MPI	Yes	Yes	No	Yes
Works with all schedulers	No	Yes	No	Yes
Designed for general scientific use cases	Yes	No	No	Yes
Contained environment has coorect perms	Yes	No	Yes	Yes
Containers are portable, unmodified by use	No	No	No	Yes
Trivial HPC install (one package, zero conf)	No	Yes	Yes	Yes
Admins can control and limit capabilities	Yes	No	No	Yes

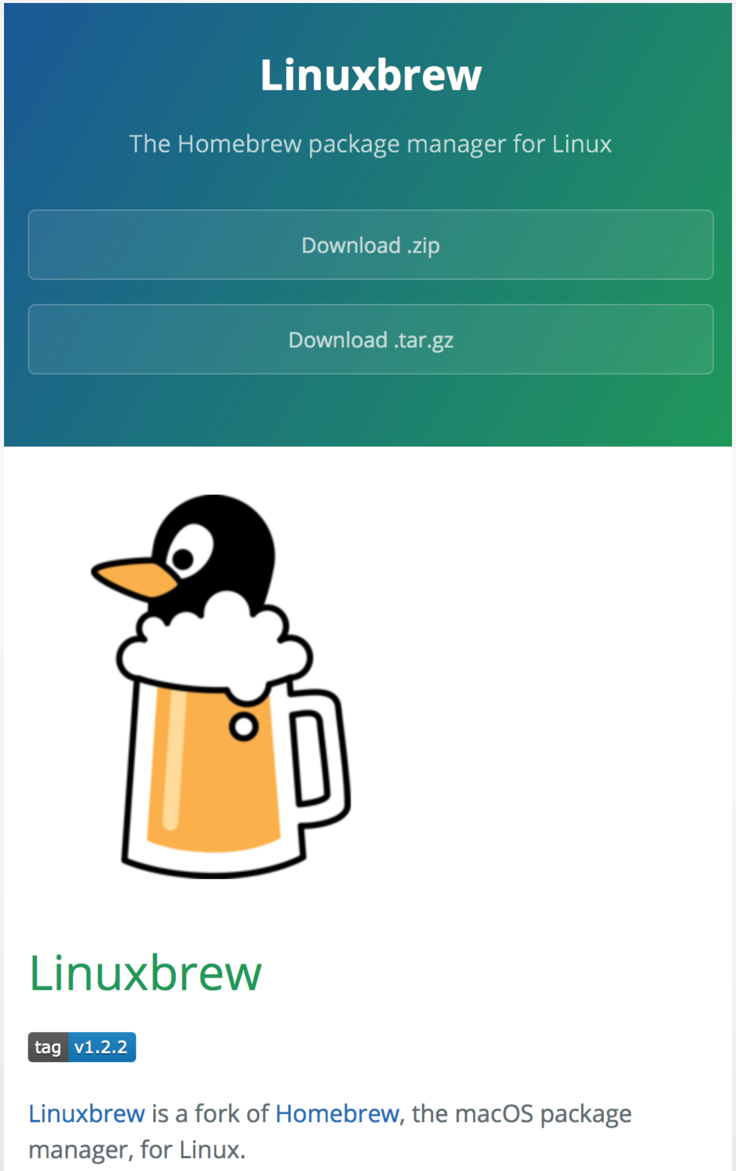
* Relies on Docker

** Depends on upstream

*** With security implications

http://www.pgbovine.net/cde.html

4. Easy installing



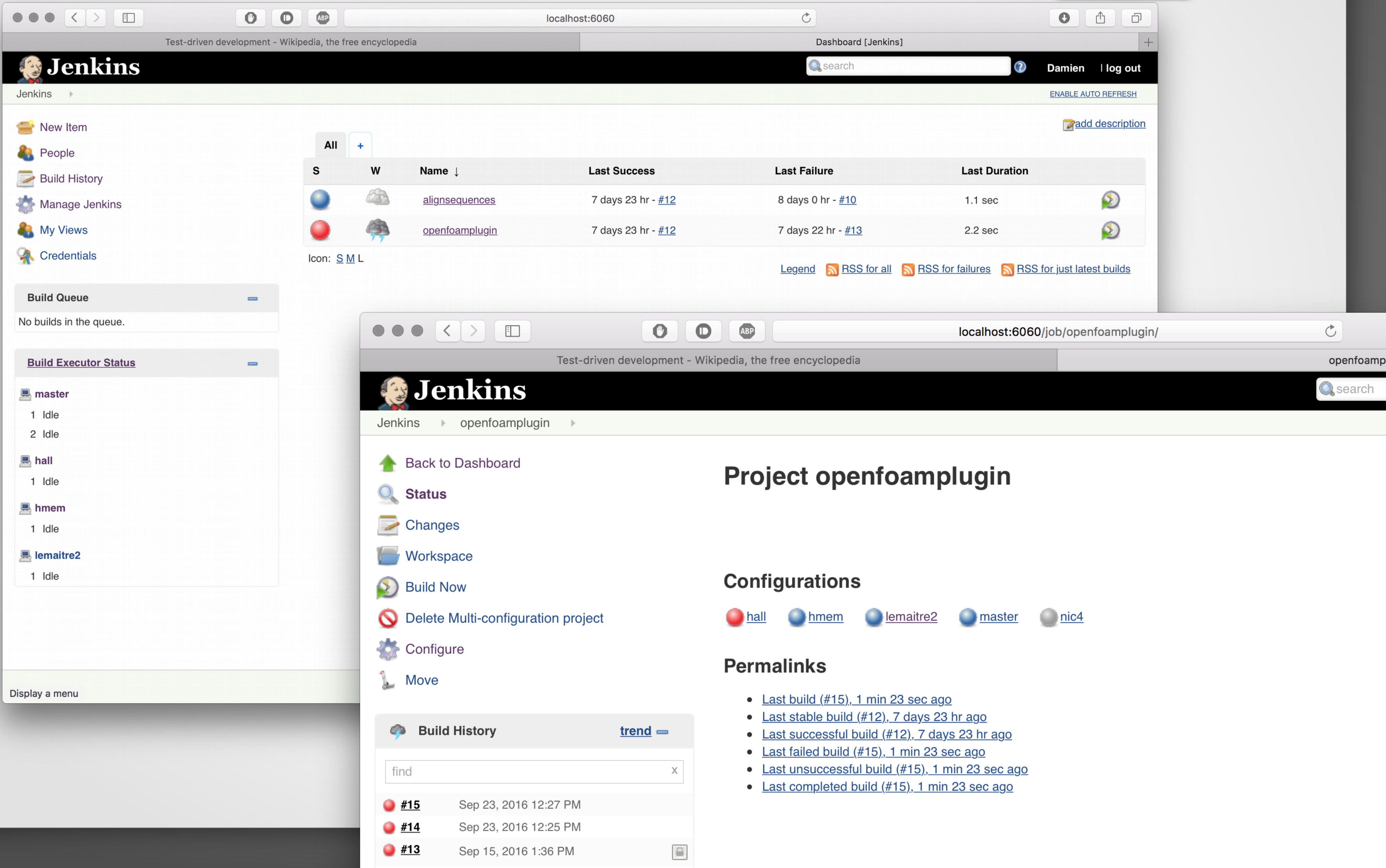
Linuxbrew

tag v1.2.2

Linuxbrew is a fork of Homebrew, the macOS package manager, for Linux.

http://linuxbrew.sh

5. Automatic build tests



localhost:6060

Test-driven development - Wikipedia, the free encyclopedia | Dashboard [Jenkins]

Jenkins

search | Damien | log out

Jenkins ▸ | ENABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials

add description

All | +

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☁	alignsequences	7 days 23 hr - #12	8 days 0 hr - #10	1.1 sec
●	⛈	openfoamplugin	7 days 23 hr - #12	7 days 22 hr - #13	2.2 sec

Icon: S M L

Legend | RSS for all | RSS for failures | RSS for just latest builds

Build Queue

No builds in the queue.

Build Executor Status

master	
1	Idle
2	Idle
hall	
1	Idle
hmem	
1	Idle
lemaitre2	
1	Idle

Display a menu

localhost:6060/job/openfoamplugin/

Test-driven development - Wikipedia, the free encyclopedia | openfoamp

Jenkins

search

Jenkins ▸ openfoamplugin ▸

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Multi-configuration project
- Configure
- Move

Project openfoamplugin

Configurations

● hall ● hmem ● lemaitre2 ● master ● nic4

Permalinks

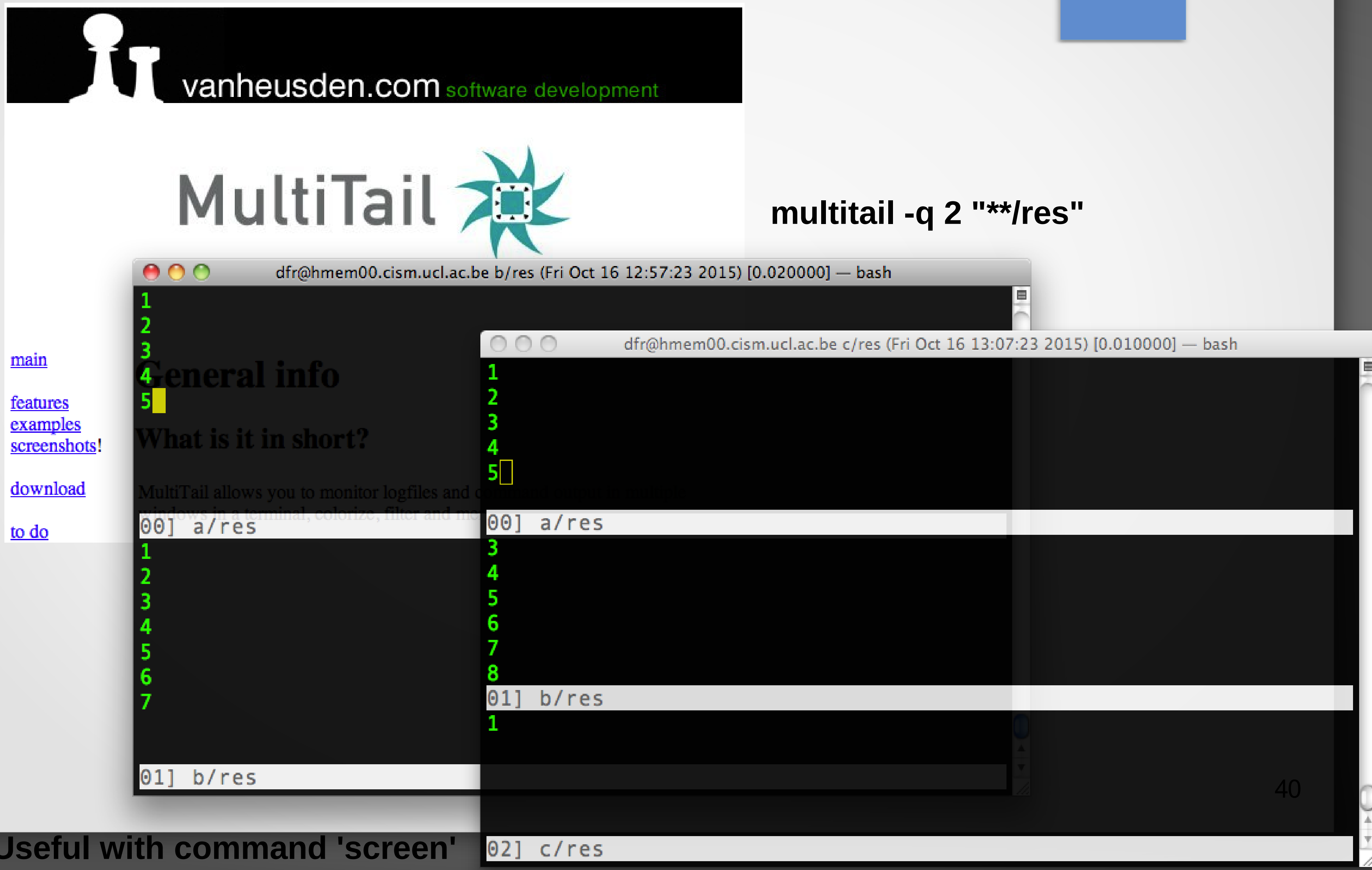
- Last build (#15), 1 min 23 sec ago
- Last stable build (#12), 7 days 23 hr ago
- Last successful build (#12), 7 days 23 hr ago
- Last failed build (#15), 1 min 23 sec ago
- Last unsuccessful build (#15), 1 min 23 sec ago
- Last completed build (#15), 1 min 23 sec ago

Build History trend

find x

● #15	Sep 23, 2016 12:27 PM
● #14	Sep 23, 2016 12:25 PM
● #13	Sep 15, 2016 1:36 PM

6. Monitoring



vanheusden.com software development

MultiTail

main

features
examples
screenshots!

download

to do

General info

What is it in short?

MultiTail allows you to monitor logfiles and c… windows in a terminal, colorize, filter and me…

dfr@hmem00.cism.ucl.ac.be b/res (Fri Oct 16 12:57:23 2015) [0.020000] — bash

```
1
2
3
4
5
00] a/res
1
2
3
4
5
6
7

01] b/res
```

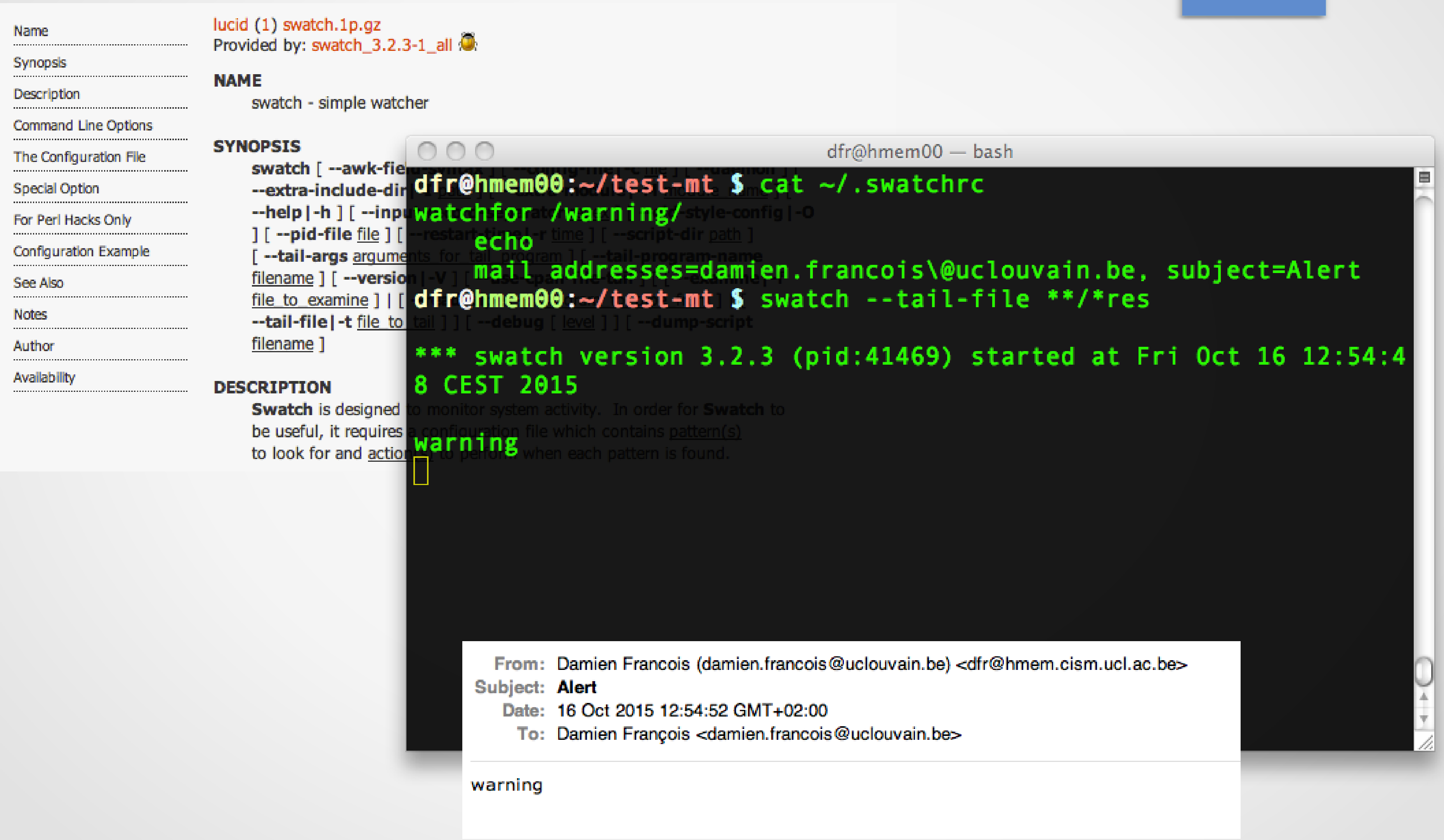
dfr@hmem00.cism.ucl.ac.be c/res (Fri Oct 16 13:07:23 2015) [0.010000] — bash

```
1
2
3
4
5
00] a/res
3
4
5
6
7
8
01] b/res
1




02] c/res
```

multitail -q 2 "**/res"

Useful with command 'screen'

6. Monitoring



- Name
- Synopsis
- Description
- Command Line Options
- The Configuration File
- Special Option
- For Perl Hacks Only
- Configuration Example
- See Also
- Notes
- Author
- Availability

lucid (1) swatch.1p.gz
Provided by: swatch_3.2.3-1_all

NAME
swatch - simple watcher

SYNOPSIS
swatch [**--awk-fie**
--extra-include-dir
--help|-h] [**--inpu**
] [**--pid-file** file] [
[**--tail-args** arguments for tail program] [--tail-program-name
filename] [**--version**
file to examine] | [
--tail-file|-t file to tail]] [--debug [level]] [--dump-script
filename]

DESCRIPTION
Swatch is designed to monitor system activity. In order for **Swatch** to be useful, it requires a configuration file which contains pattern(s) to look for and action(s) to perform when each pattern is found.

dfr@hmem00 — bash

```
dfr@hmem00:~/test-mt $ cat ~/.swatchrc
watchfor /warning/
    echo
    mail addresses=damien.francois\@uclouvain.be, subject=Alert
dfr@hmem00:~/test-mt $ swatch --tail-file **/*res

*** swatch version 3.2.3 (pid:41469) started at Fri Oct 16 12:54:4
8 CEST 2015

warning
```

From: Damien Francois (damien.francois@uclouvain.be) <dfr@hmem.cism.ucl.ac.be>
Subject: **Alert**
Date: 16 Oct 2015 12:54:52 GMT+02:00
To: Damien François <damien.francois@uclouvain.be>

warning

Useful with command 'screen'

Dev's toolkit :

1. Programming language
2. Good practices / Code Style Guides
3. Text editor / IDE
4. Source control management
5. Debuggers / Profilers
6. Databases
7. Packaging / Distributing tools
8. Comments and documentation
9. Tests
10. Licensing

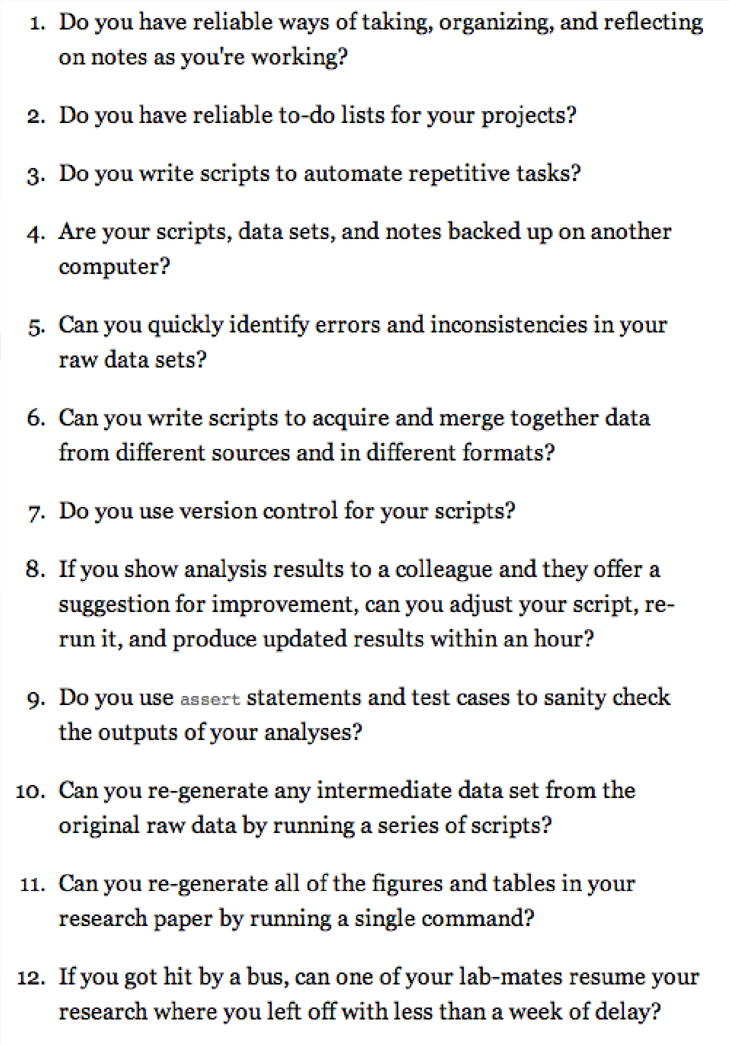
Ops' toolkit :

1. Virtualization platforms (Virtual box, Vagrant)
2. Multi-host connexions (pdsh, clustershell)
3. Configuration management/ (ansible)
4. Installing (easybuild)
5. Automatic build tests (jenkins)
6. Monitoring (multitail)

The 'Phillip' test

- 12 simple questions
- ordered by 'difficulty'
- measures quality of organization
- for research programming

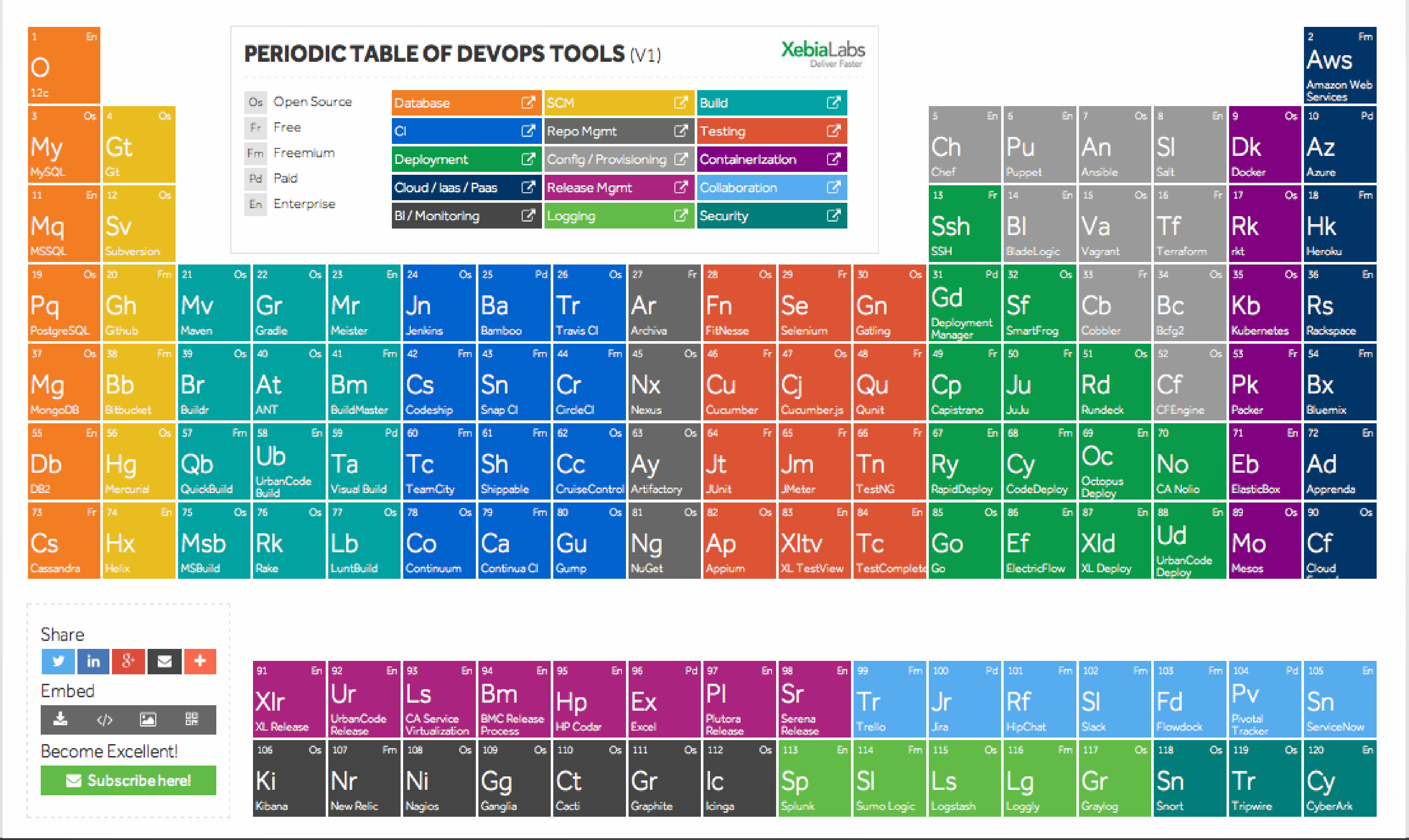
If you do not score at least a 7 there is room for improvement using the tools presented here



1. Do you have reliable ways of taking, organizing, and reflecting on notes as you're working?
2. Do you have reliable to-do lists for your projects?
3. Do you write scripts to automate repetitive tasks?
4. Are your scripts, data sets, and notes backed up on another computer?
5. Can you quickly identify errors and inconsistencies in your raw data sets?
6. Can you write scripts to acquire and merge together data from different sources and in different formats?
7. Do you use version control for your scripts?
8. If you show analysis results to a colleague and they offer a suggestion for improvement, can you adjust your script, re-run it, and produce updated results within an hour?
9. Do you use `assert` statements and test cases to sanity check the outputs of your analyses?
10. Can you re-generate any intermediate data set from the original raw data by running a series of scripts?
11. Can you re-generate all of the figures and tables in your research paper by running a single command?
12. If you got hit by a bus, can one of your lab-mates resume your research where you left off with less than a week of delay?

http://pgbovine.net/research-programming-workflow.htm

Work quicker & more reliably



PERIODIC TABLE OF DEVOPS TOOLS (V1)

XebiaLabs
Deliver Faster

Os	Open Source
Fr	Free
Fm	Freemium
Pd	Paid
En	Enterprise

Database	SCM	Build
CI	Repo Mgmt	Testing
Deployment	Config / Provisioning	Containerization
Cloud / Iaas / Paas	Release Mgmt	Collaboration
BI / Monitoring	Logging	Security

1 En O 12c																	2 Fm Aws Amazon Web Services
3 Os My MySQL	4 Os Gt Git											5 En Ch Chef	6 En Pu Puppet	7 Os An Ansible	8 En Sl Salt	9 Os Dk Docker	10 Pd Az Azure
11 En Mq MSSQL	12 Os Sv Subversion											13 Fr Ssh SSH	14 En Bl BladeLogic	15 Os Va Vagrant	16 Fr Tf Terraform	17 Os Rk rkt	18 Fm Hk Heroku
19 Os Pq PostgreSQL	20 Fm Gh Github	21 Os Mv Maven	22 Os Gr Gradle	23 En Mr Meister	24 Os Jn Jenkins	25 Pd Ba Bamboo	26 Os Tr Travis CI	27 Fr Ar Archiva	28 Os Fn FitNesse	29 Fr Se Selenium	30 Os Gn Gatling	31 Pd Gd Deployment Manager	32 Os Sf SmartFrog	33 Fr Cb Cobbler	34 Os Bc Bcfg2	35 Os Kb Kubernetes	36 En Rs Rackspace
37 Os Mg MongoDB	38 Fm Bb Bitbucket	39 Os Br Buildr	40 Os At ANT	41 Fm Bm BuildMaster	42 Fm Cs Codeship	43 Fm Sn Snap CI	44 Fm Cr CircleCI	45 Os Nx Nexus	46 Fr Cu Cucumber	47 Os Cj Cucumber.js	48 Fr Qu Qunit	49 Fr Cp Capistrano	50 Fr Ju JuJu	51 Os Rd Rundeck	52 Os Cf CFEngine	53 Fr Pk Packer	54 Fm Bx Bluemix
55 En Db DB2	56 Os Hg Mercurial	57 Fm Qb QuickBuild	58 En Ub UrbanCode Build	59 Pd Ta Visual Build	60 Fm Tc TeamCity	61 Fm Sh Shippable	62 Os Cc CruiseControl	63 Os Ay Artifactory	64 Fr Jt JUnit	65 Fr Jm JMeter	66 Fr Tn TestNG	67 En Ry RapidDeploy	68 Fm Cy CodeDeploy	69 En Oc Octopus Deploy	70 No CA Nolio	71 En Eb ElasticBox	72 En Ad Apprenda
73 Fr Cs Cassandra	74 En Hx Helix	75 Os Msb MSBuild	76 Os Rk Rake	77 Os Lb LuntBuild	78 Os Co Continuum	79 Fm Ca Continua CI	80 Os Gu Gump	81 Os Ng NuGet	82 Os Ap Appium	83 En Xltv XL TestView	84 En Tc TestComplet	85 Os Go Go	86 En Ef ElectricFlow	87 En Xld XL Deploy	88 En Ud UrbanCode Deploy	89 Os Mo Mesos	90 Os Cf Cloud
			91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Ls CA Service Virtualization	94 En Bm BMC Release Process	95 En Hp HP Codar	96 Pd Ex Excel	97 En Pl Plutora Release	98 En Sr Serena Release	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow
			106 Os Ki Kibana	107 Fm Nr New Relic	108 Os Ni Nagios	109 Os Gg Ganglia	110 Os Ct Cacti	111 Os Gr Graphite	112 Os Ic Icinga	113 En Sp Splunk	114 Fm Sl Sumo Logic	115 Os Ls Logstash	116 Fm Lg Loggly	117 Os Gr Graylog	118 Os Sn Snort	119 Os Tr Tripwire	120 En Cy CyberArk