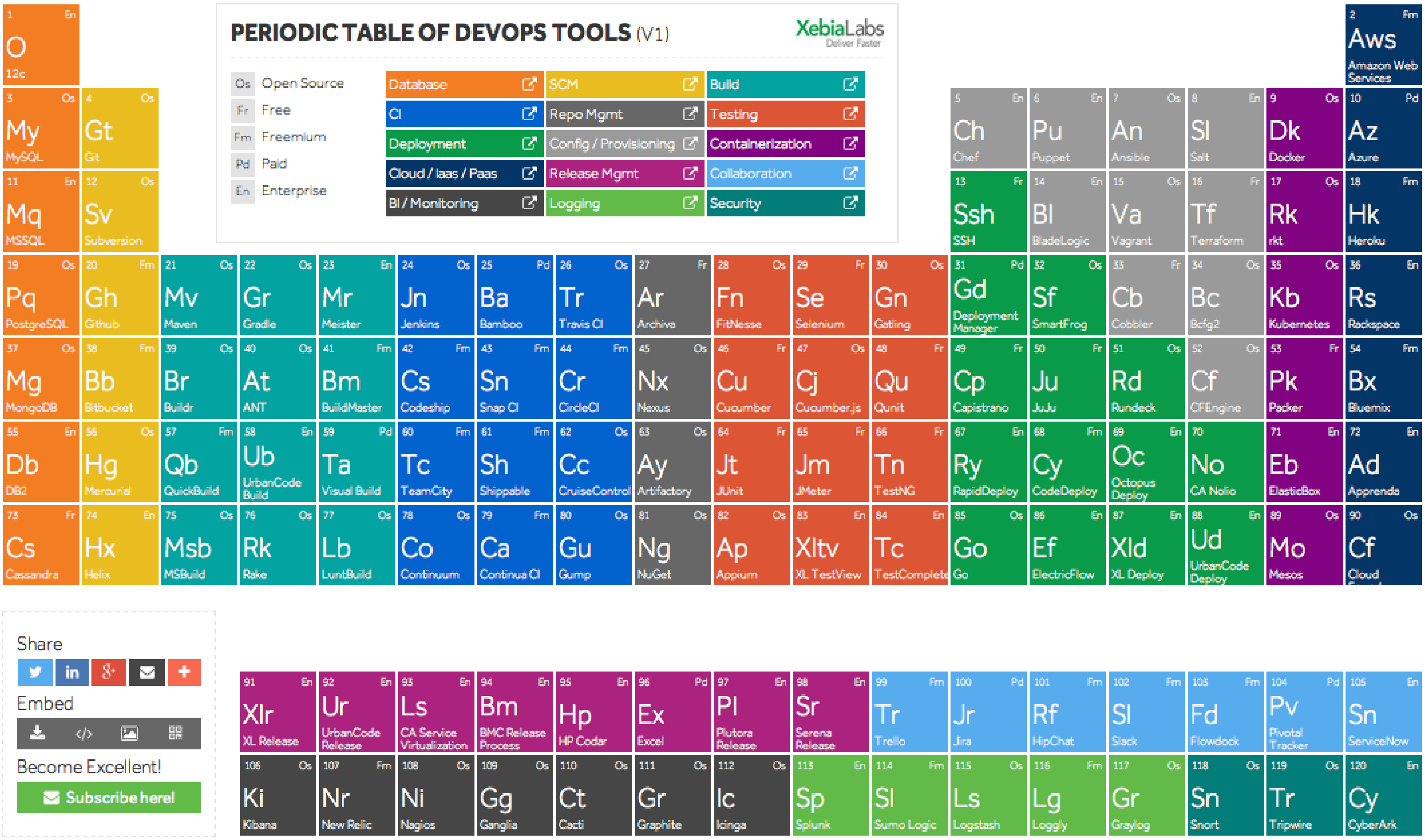
Introduction to Scientific Software Deployment and Development

damien.francois@uclouvain.be

October 2021

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



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

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

Share

Embed

Become Excellent!

Subscribe here!

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

Goal of this session:

“Promote the *tools*
the professionals are using for
developing and **deploying** programs,
to make them **correct**, **maintainable**, **shareable**, and **fast**,
efficiently.”

“...to make them **correct** and **maintainable**, …, *efficiently*”

Paul F. Dubois. 1999. **Ten Good Practices in Scientific Programming**. *Computing in Science and Eng.* 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

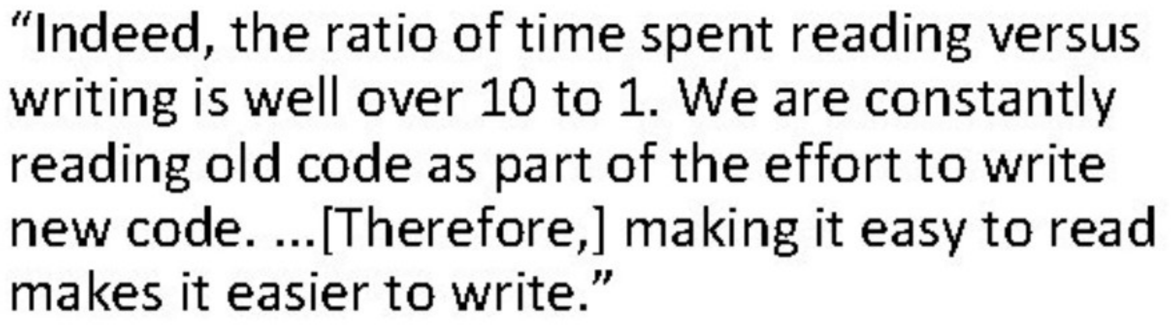
Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017) **Good enough practices in scientific computing**. *PLoS Comput Biol* 13(6): e1005510. https://doi.org/10.1371/journal.pcbi.1005510

“...to make them **correct** and **maintainable**, …, *efficiently*”

Follow programming good practices:

1. Write for humans, not for computers
2. Use the appropriate language
3. Organize for change and make incremental changes
4. Follow good coding principles
5. Plan for mistakes, automate testing
6. Use modern source-code management system
7. Document the design and purpose, not the implementation
8. Optimize only when it works already
9. Debug cleverly

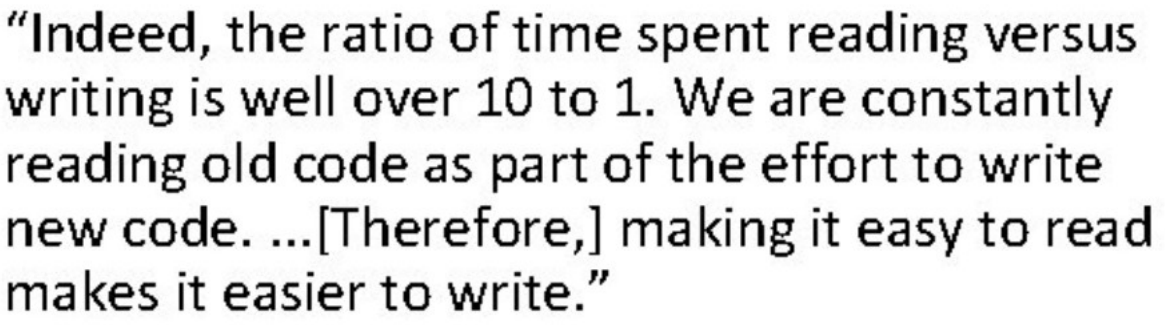
1. Write for humans, not for computers



> "Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write."

Robert C. Martin *Clean code A Handbook of Agile Software Craftsmanship*, 2009

1. Write for humans, not for computers



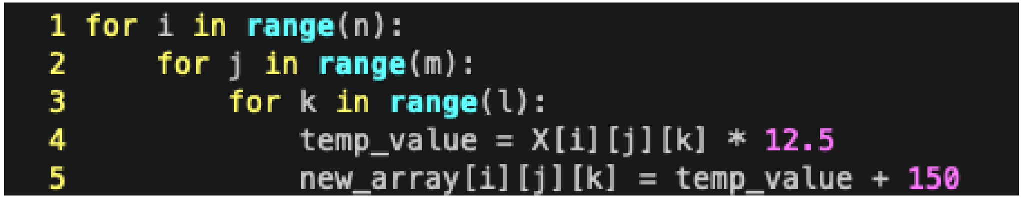
> “Indeed, the ratio of time spent reading versus writing is well over 10 to 1. We are constantly reading old code as part of the effort to write new code. ...[Therefore,] making it easy to read makes it easier to write.”

- Do not be afraid of being too explicit
- Choose names carefully
- Structure your code

Let the compiler do the writing for the computer

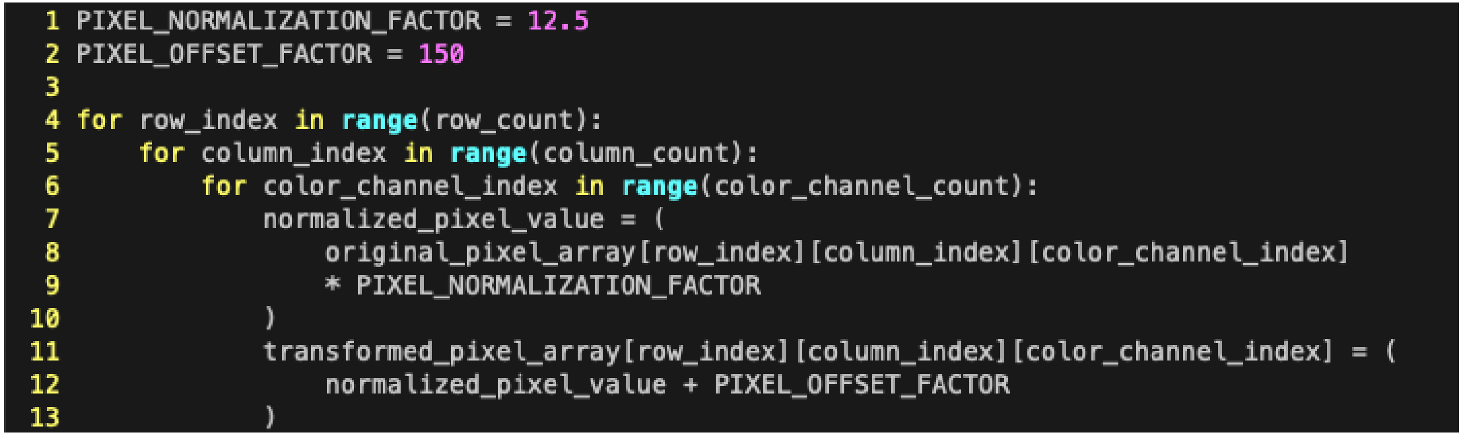
Robert C. Martin *Clean code A Handbook of Agile Software Craftsmanship*, 2009

1. Write for humans, not for computers



```python
for i in range(n):
    for j in range(m):
        for k in range(l):
            temp_value = X[i][j][k] * 12.5
            new_array[i][j][k] = temp_value + 150
```

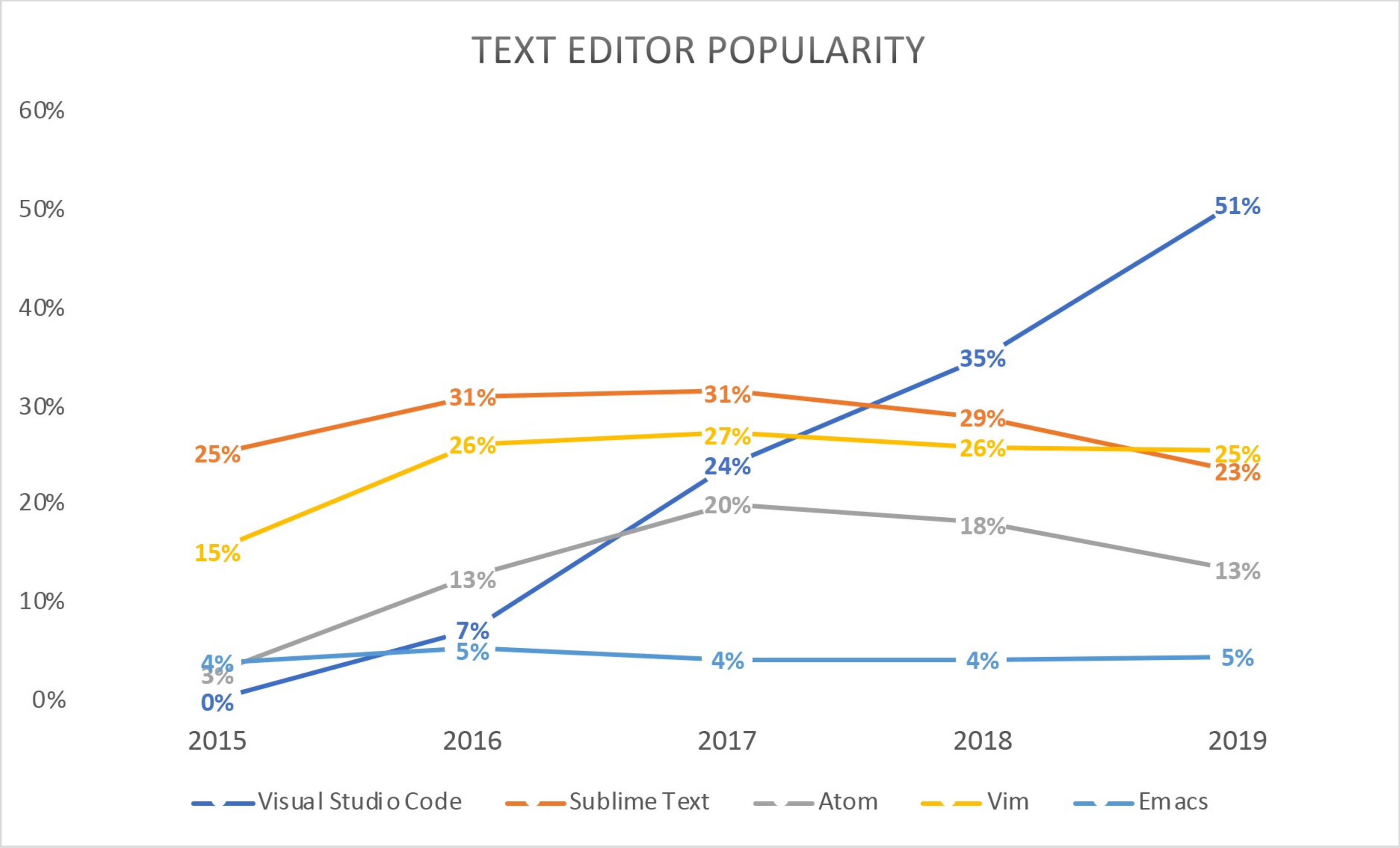
vs



```python
PIXEL_NORMALIZATION_FACTOR = 12.5
PIXEL_OFFSET_FACTOR = 150

for row_index in range(row_count):
    for column_index in range(column_count):
        for color_channel_index in range(color_channel_count):
            normalized_pixel_value = (
                original_pixel_array[row_index][column_index][color_channel_index]
                * PIXEL_NORMALIZATION_FACTOR
            )
            transformed_pixel_array[row_index][column_index][color_channel_index] = (
                normalized_pixel_value + PIXEL_OFFSET_FACTOR
            )
```

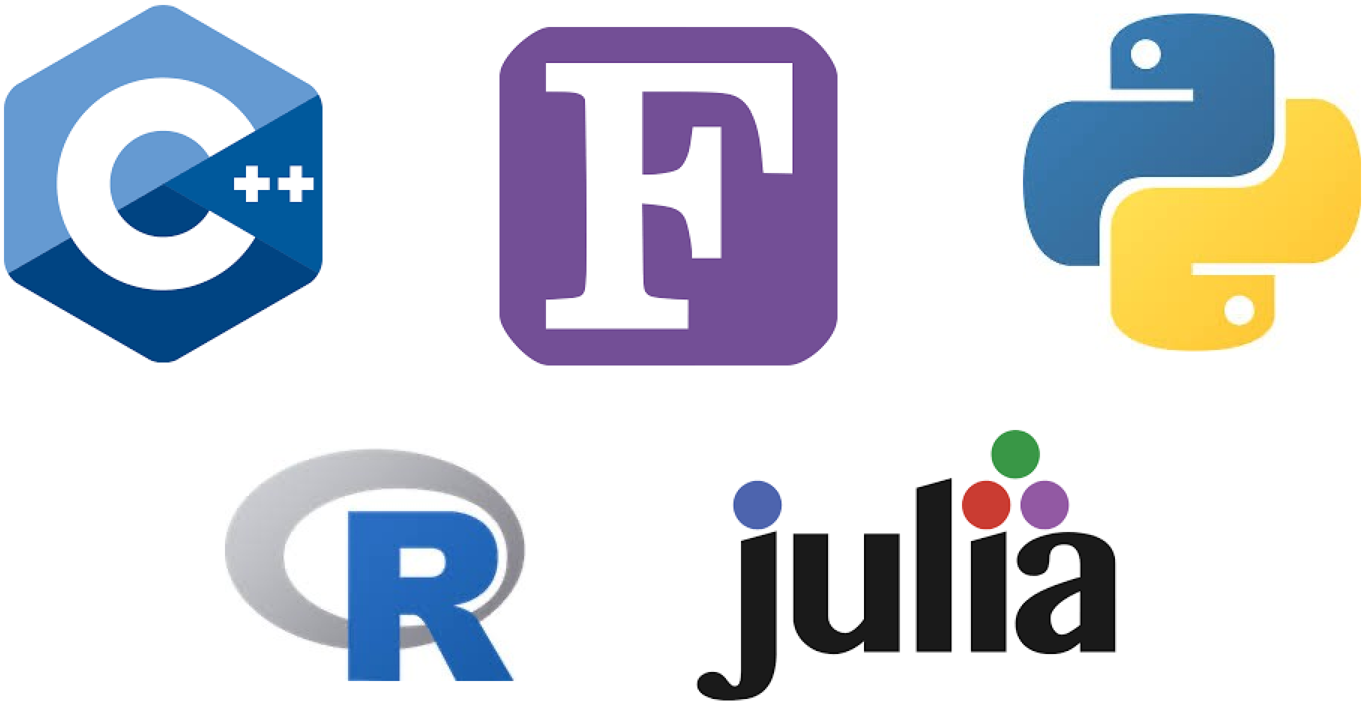
https://towardsdatascience.com/data-scientists-your-variable-names-are-awful-heres-how-to-fix-them-89053d2855be

1. Write for humans, not for computers (and learn your editor)



https://blog.robenkleene.com/2020/09/21/the-era-of-visual-studio-code/

2. Use the appropriate language

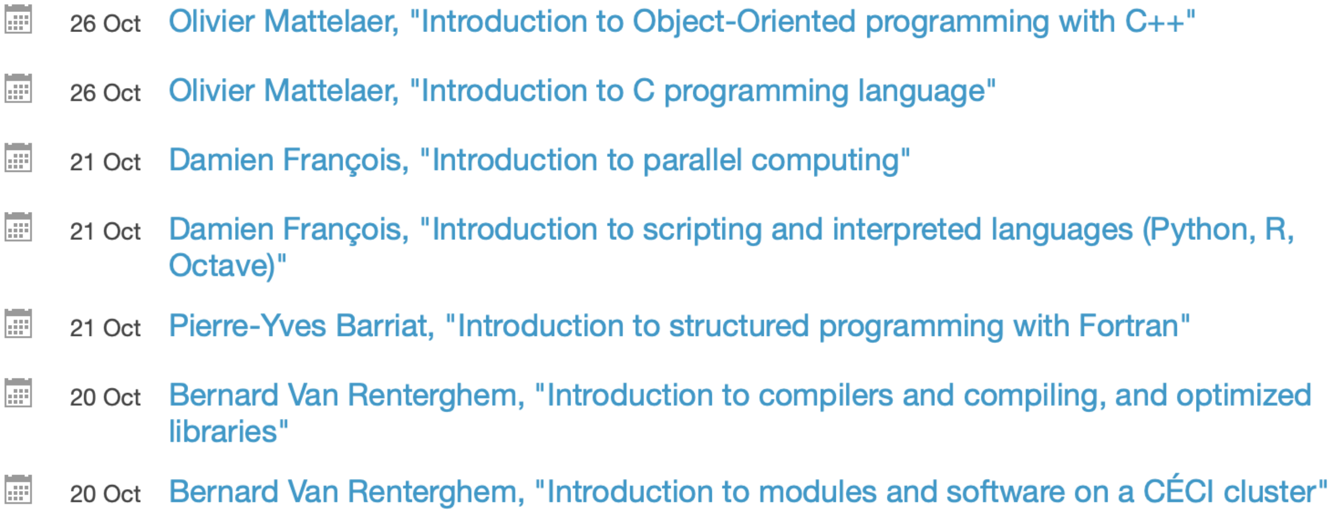


are all valid choices in a scientific context.

2. Use the appropriate language

What they have in common:

- Computation efficiency concern
- Optimized libraries available for linear algebra, signal processing, learning, etc.
- Support for parallel computing
- Extensions/libraries for using GPUs



26 Oct	Olivier Mattelaer, "Introduction to Object-Oriented programming with C++"
26 Oct	Olivier Mattelaer, "Introduction to C programming language"
21 Oct	Damien François, "Introduction to parallel computing"
21 Oct	Damien François, "Introduction to scripting and interpreted languages (Python, R, Octave)"
21 Oct	Pierre-Yves Barriat, "Introduction to structured programming with Fortran"
20 Oct	Bernard Van Renterghem, "Introduction to compilers and compiling, and optimized libraries"
20 Oct	Bernard Van Renterghem, "Introduction to modules and software on a CÉCI cluster"

2. Use the appropriate language

Haskell

Scala

“Functional programming”

Very close to mathematical formulation

Imposes constraints that make code less prone to bugs and easier to make parallel

Not very popular in HPC (yet)

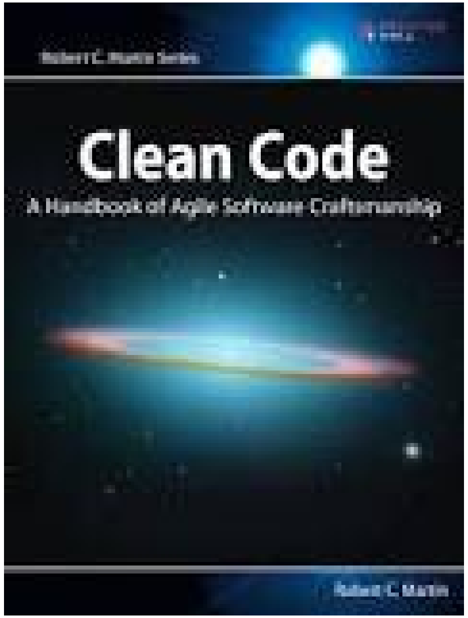
3. Organize for change and make incremental changes

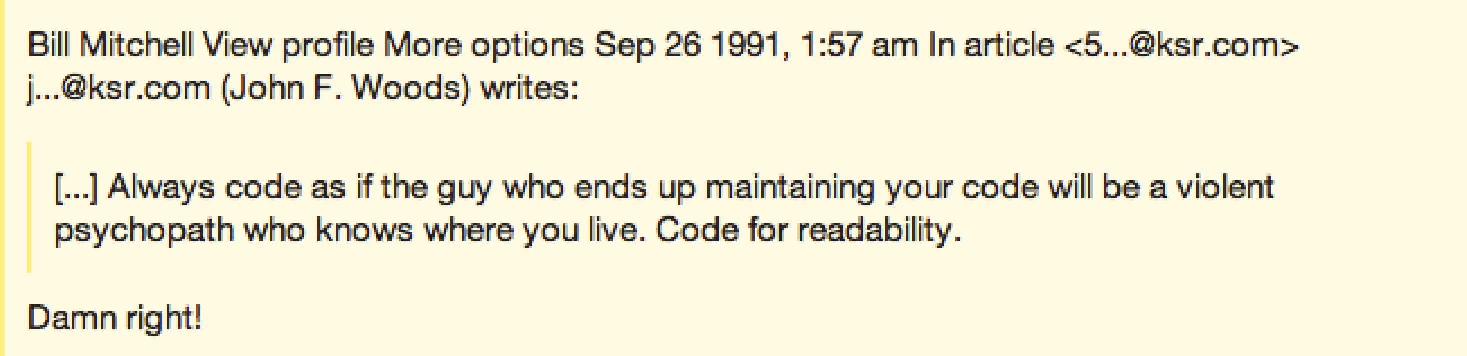
- Scientific software specifications are always changing
- Work from working state to another working state
- Document the changes and why they were made
- (And sometimes restart from scratch)

Keyword: **modularity:** small independent interchangeable building blocks (e.g. functions)

4. Follow 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
- Follow style guidelines
- Many others...





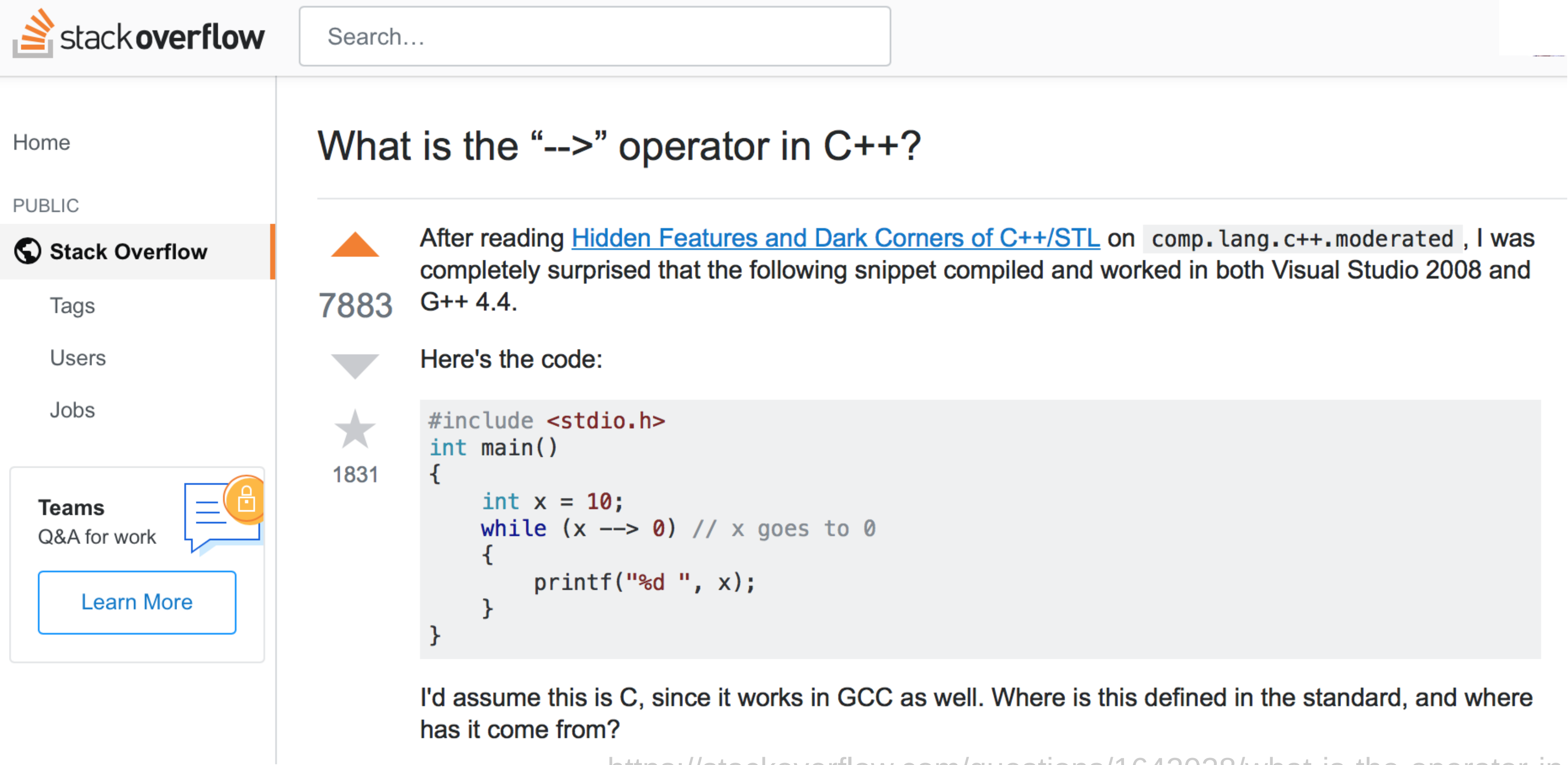
Bill Mitchell View profile More options Sep 26 1991, 1:57 am In article <5...@ksr.com> j...@ksr.com (John F. Woods) writes:

> [...] 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

4. Follow good coding principles and 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.

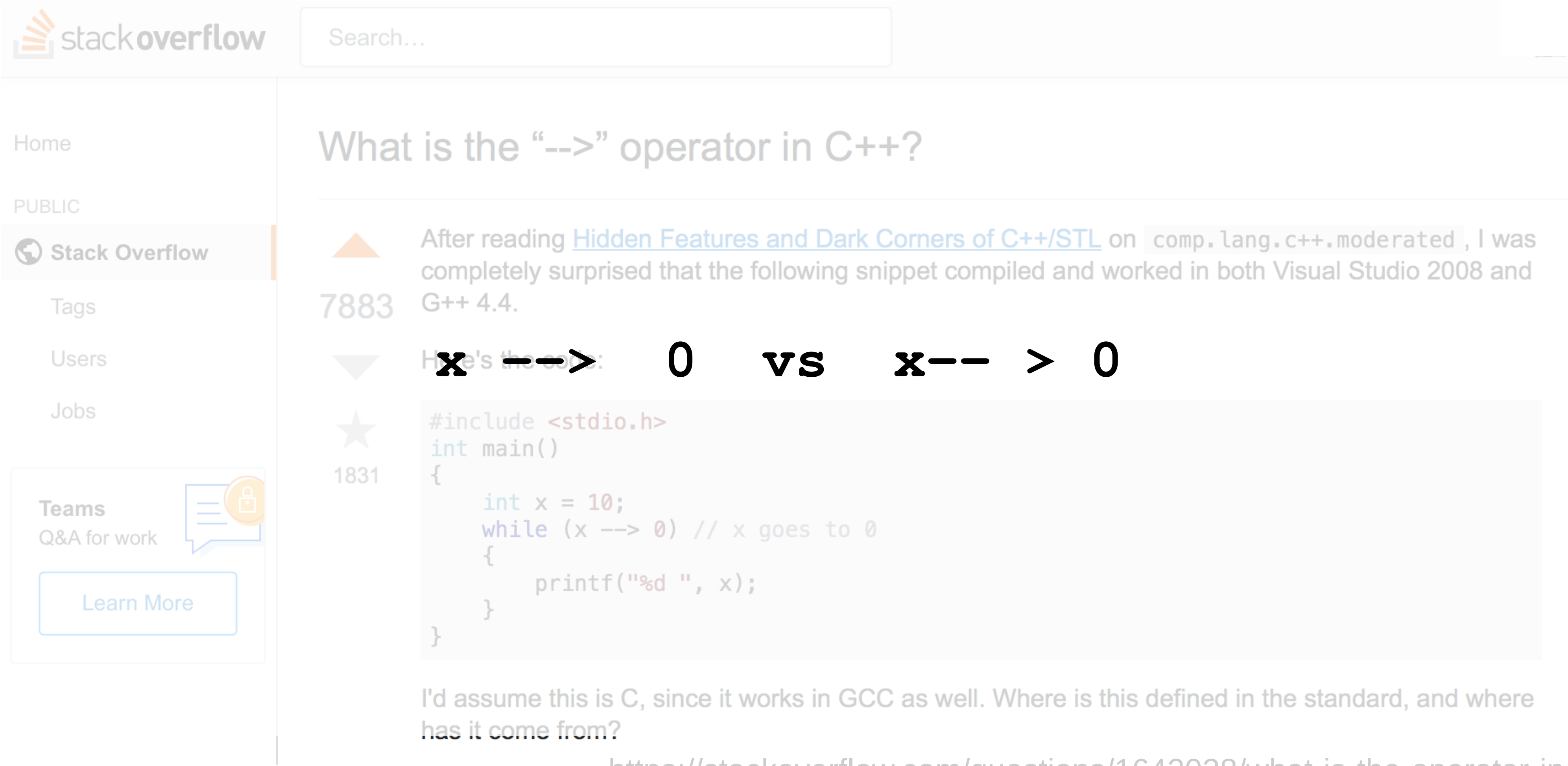
Here's the code:

```
#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?

https://stackoverflow.com/questions/1642028/what-is-the-operator-in-c

4. Follow good coding principles and 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:

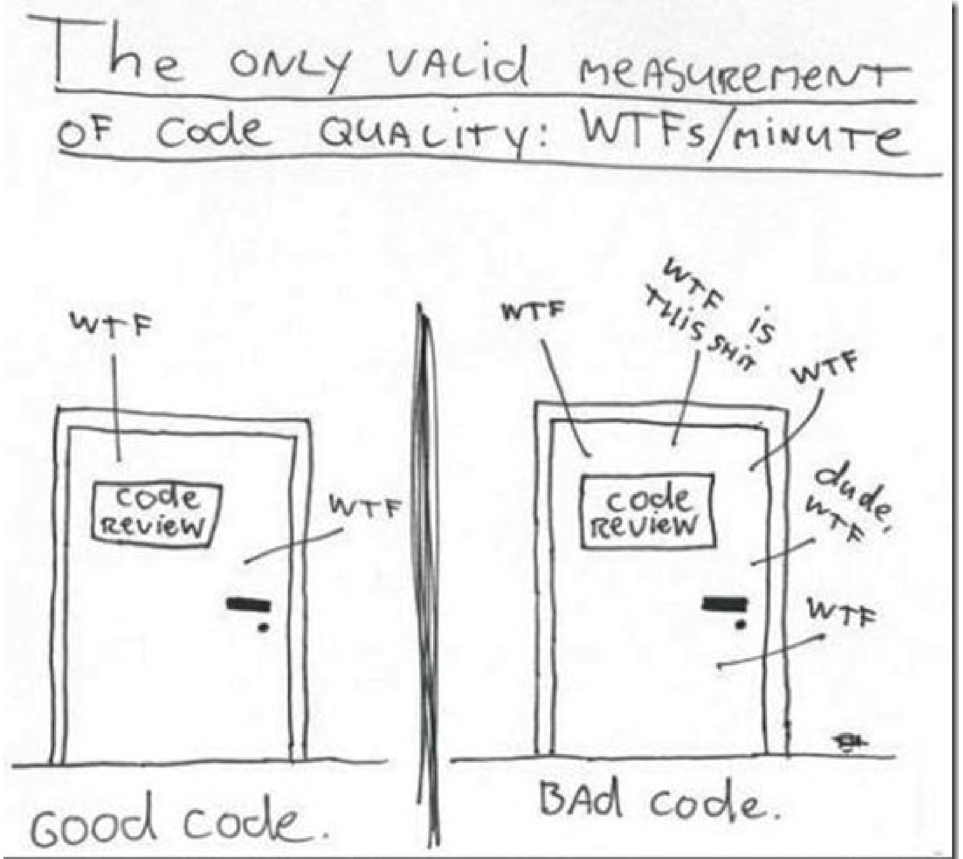
`x --> 0   vs   x-- > 0`

```
#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?

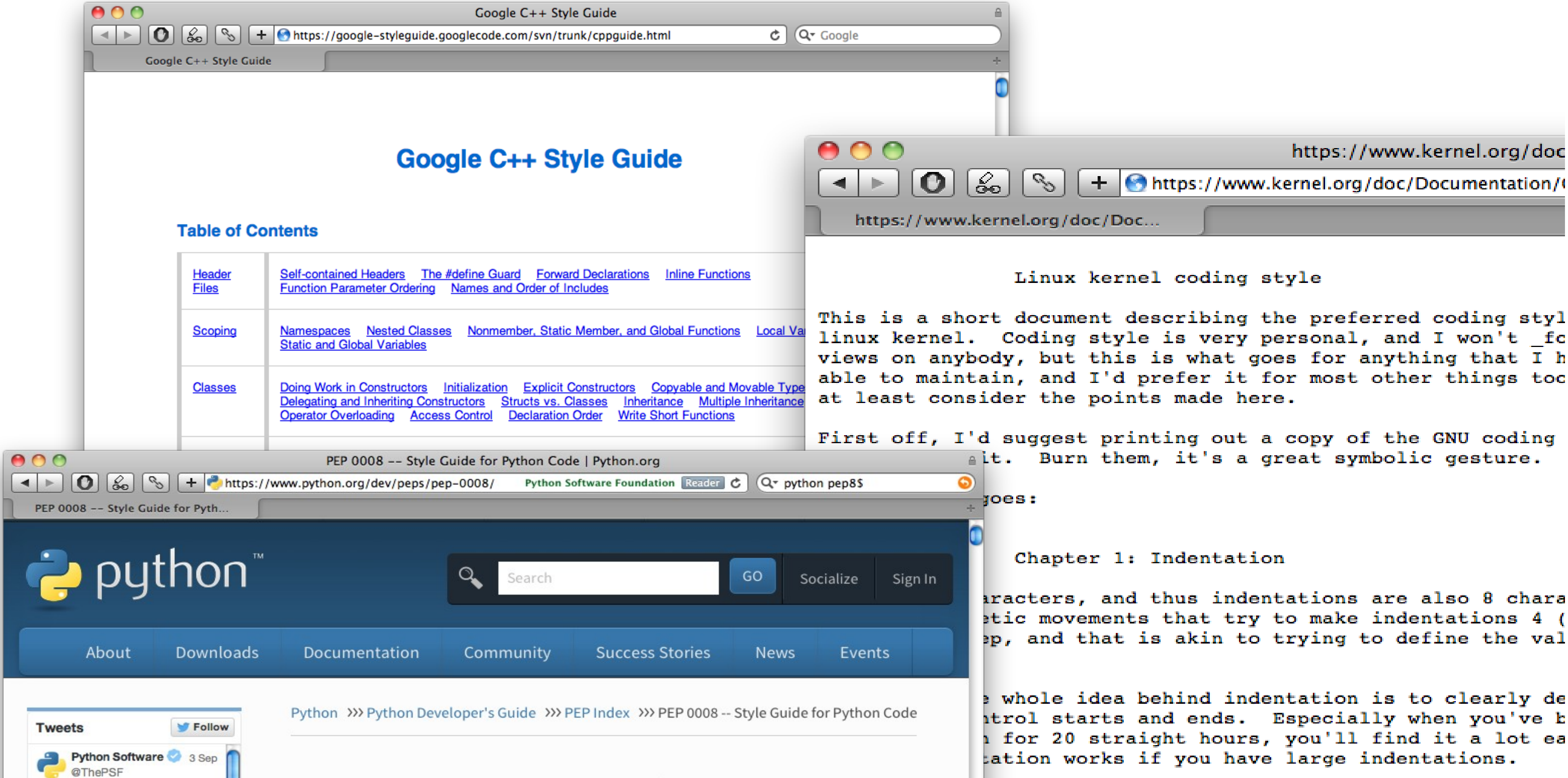
https://stackoverflow.com/questions/1642028/what-is-the-operator-in-c

4. Follow good coding principles and style



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

4. Follow good coding principles and 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 Va Static and Global Variables
Classes	Doing Work in Constructors Initialization Explicit Constructors Copyable and Movable Type Delegating and Inheriting Constructors Structs vs. Classes Inheritance Multiple Inheritance Operator Overloading Access Control Declaration Order Write Short Functions

https://www.kernel.org/doc

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

```
                    Linux kernel coding style

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

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

                 goes:


                    Chapter 1: Indentation

                 aracters, and thus indentations are also 8 chara
                 etic movements that try to make indentations 4 (
                 ep, and that is akin to trying to define the val


                 e whole idea behind indentation is to clearly de
                 ntrol starts and ends.  Especially when you've b
                 n for 20 straight hours, you'll find it a lot ea
                 ation works if you have large indentations.
```

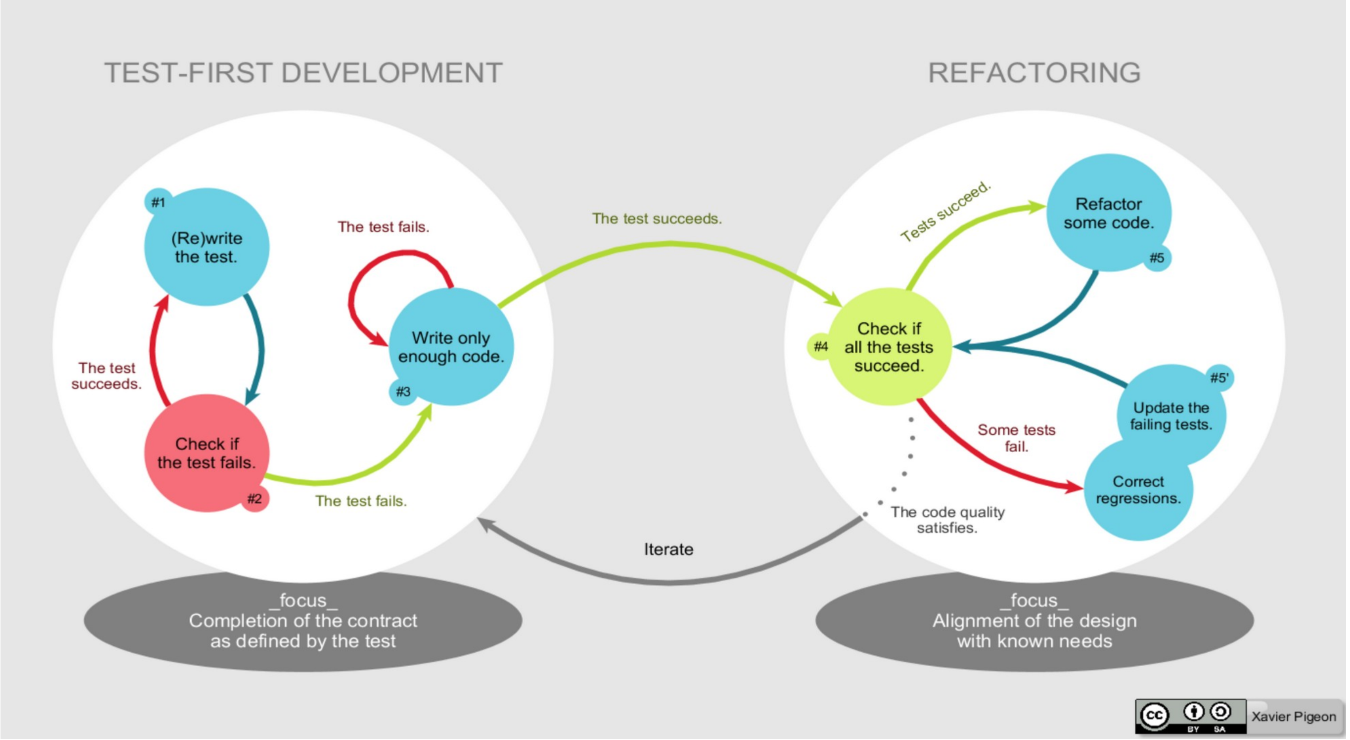
PEP 0008 -- Style Guide for Python Code | Python.org

https://www.python.org/dev/peps/pep-0008/

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

5. Plan for mistakes, automate testing; Test-driven development

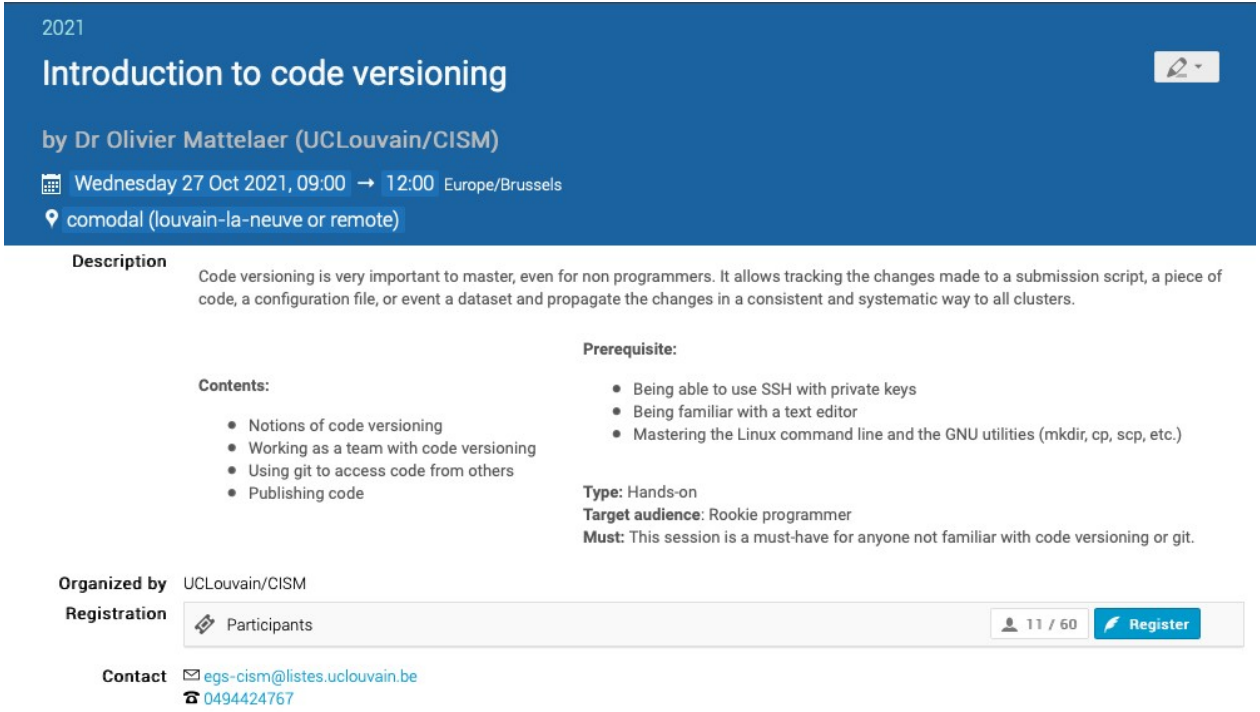
Write the tests before you even write the code



https://en.wikipedia.org/wiki/Test-driven_development

6. Use modern source-code management system

git for your code, papers, thesis, etc.



2021

Introduction to code versioning

by Dr Olivier Mattelaer (UCLouvain/CISM)

Wednesday 27 Oct 2021, 09:00 → 12:00 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description

Code versioning is very important to master, even for non programmers. It allows tracking the changes made to a submission script, a piece of code, a configuration file, or event a dataset and propagate the changes in a consistent and systematic way to all clusters.

Contents:

- Notions of code versioning
- Working as a team with code versioning
- Using git to access code from others
- Publishing code

Prerequisite:

- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)

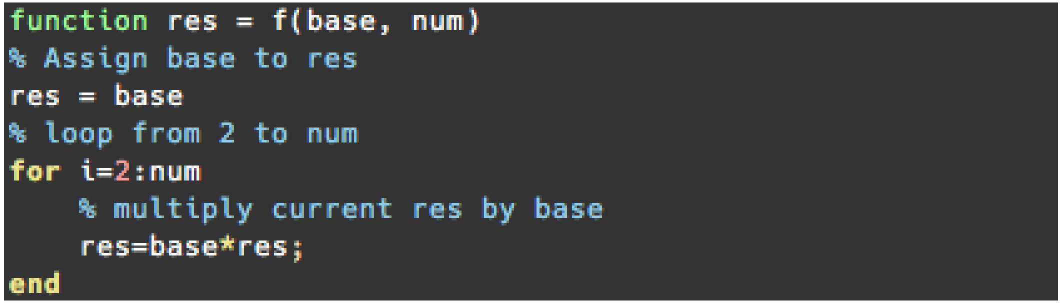
Type: Hands-on
Target audience: Rookie programmer
Must: This session is a must-have for anyone not familiar with code versioning or git.

Organized by UCLouvain/CISM

Registration Participants 11 / 60 Register

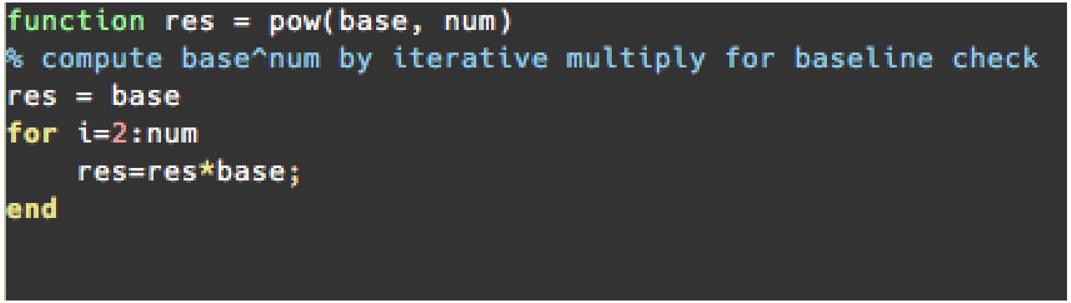
Contact egs-cism@listes.uclouvain.be
☎ 0494424767

7. Document the purpose and design, not the implementation



```
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
```

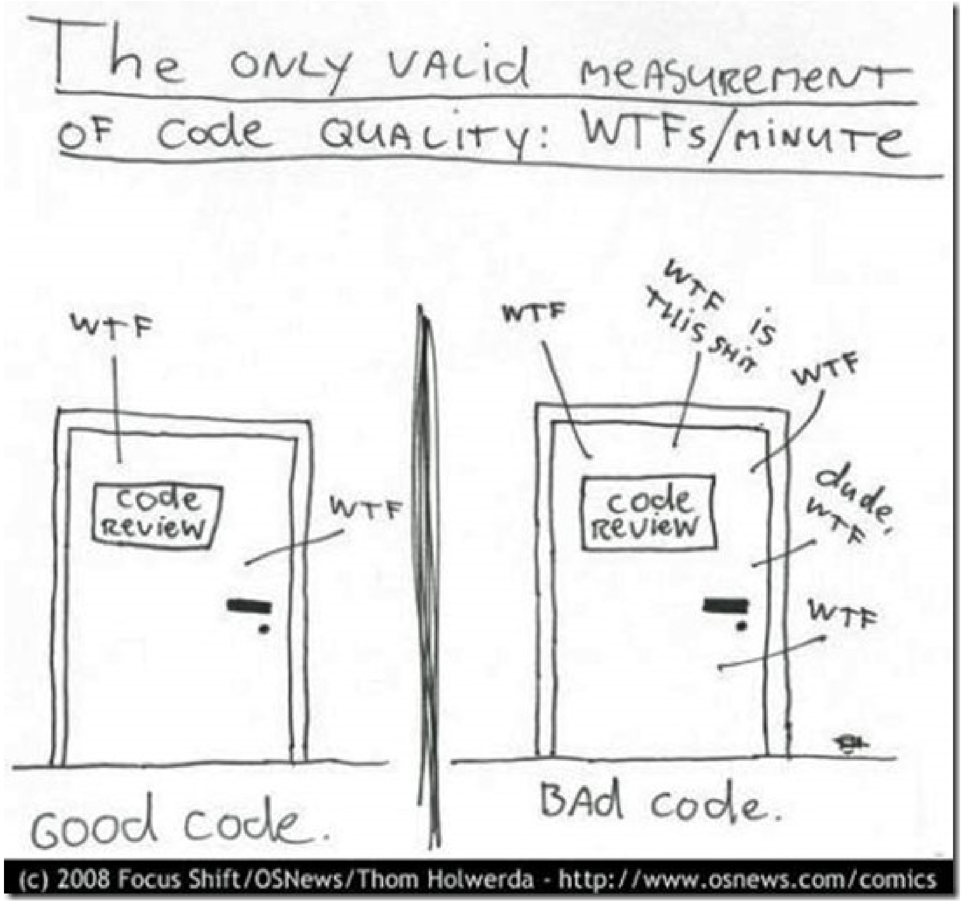
vs



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

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006561

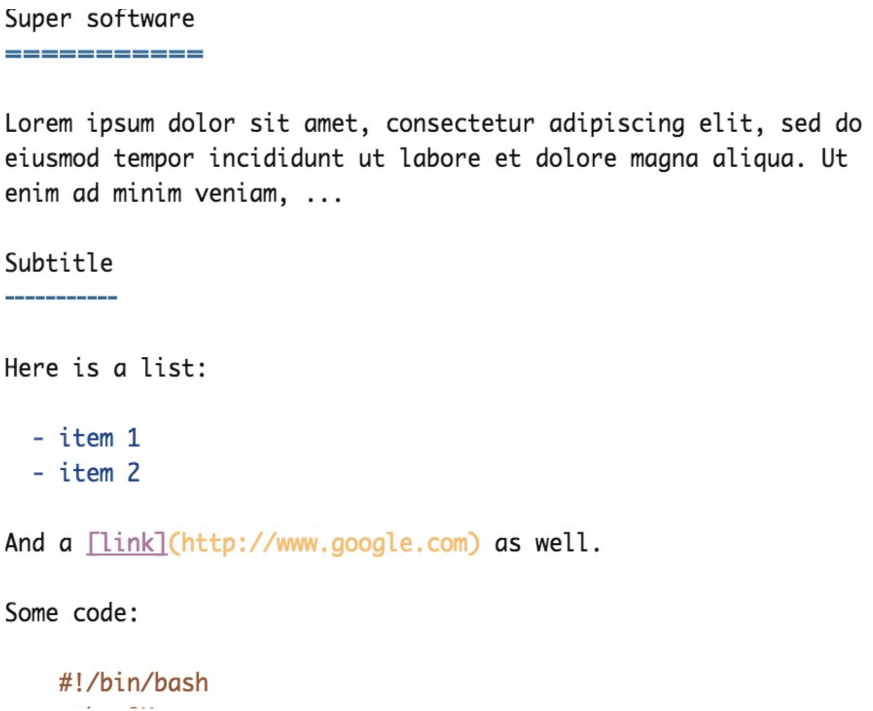
7. Document the purpose and design, not the implementation


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

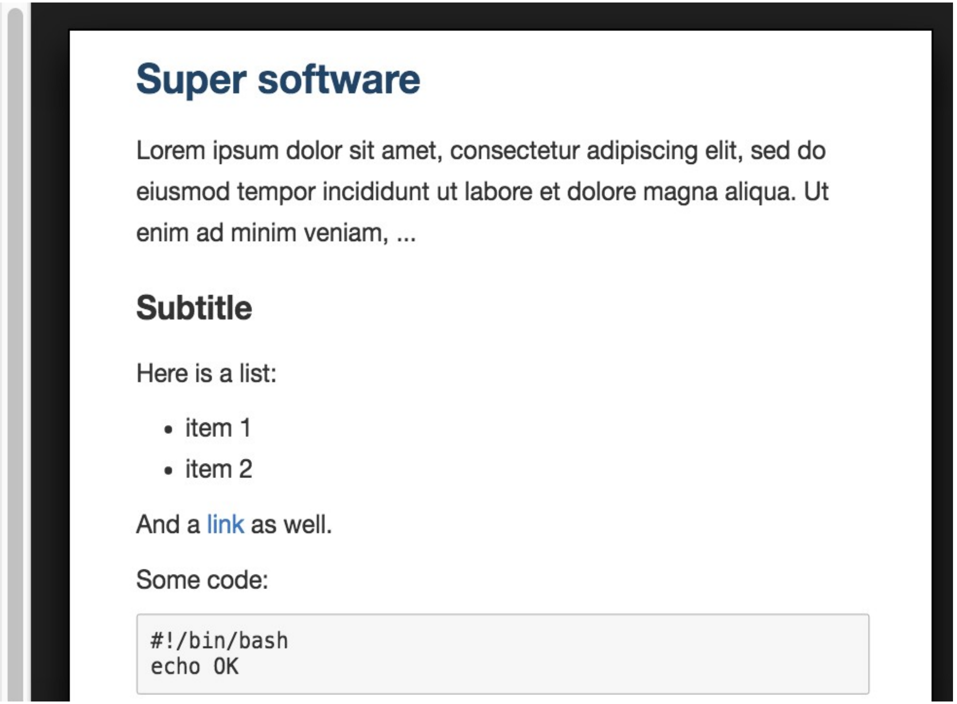
again

7. Document the purpose and design, not the implementation

Learn Markdown or RestructuredText



```
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
```



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
```

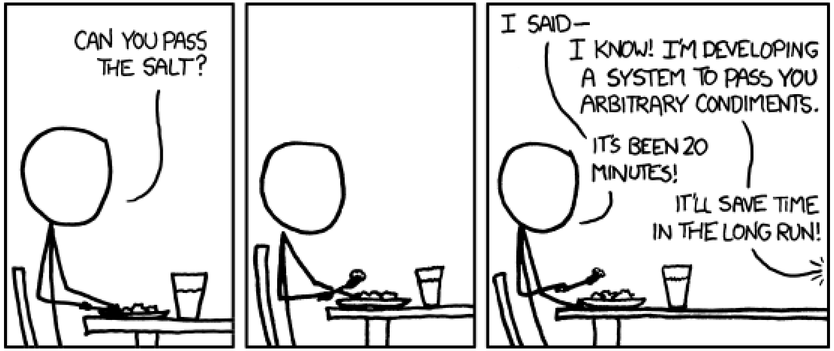
https://daringfireball.net/projects/markdown/

8. Optimize when it works already

- Do not try to make it fast when it is not working yet

(I can always make your code faster by commenting out some of it)

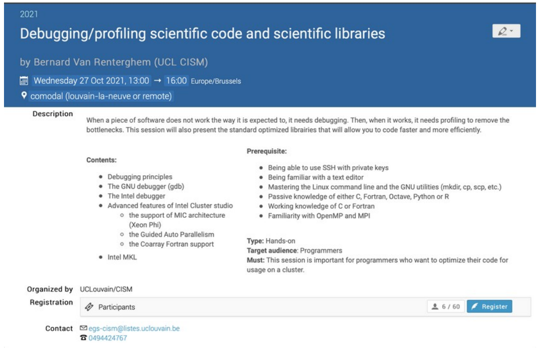
- Do not try to make it universal for future needs at the beginning


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

https://xkcd.com/974/

8. Optimize when it works already

Use a profiler



2021

Debugging/profiling scientific code and scientific libraries

by Bernard Van Renterghem (UCL CISM)

Wednesday 27 Oct 2021, 13:00 → 16:00 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description When a piece of software does not work the way it is expected to, it needs debugging. Then, when it works, it needs profiling to remove the bottlenecks. This session will also present the standard optimized librairies that will allow you to code faster and more efficiently.

Contents:

- Debugging principles
- The GNU debugger (gdb)
- The Intel debugger
- Advanced features of Intel Cluster studio
  - the support of MIC architecture (Xeon Phi)
  - the Guided Auto Parallelism
  - the Coarray Fortran support
- Intel MKL

Prerequisite:

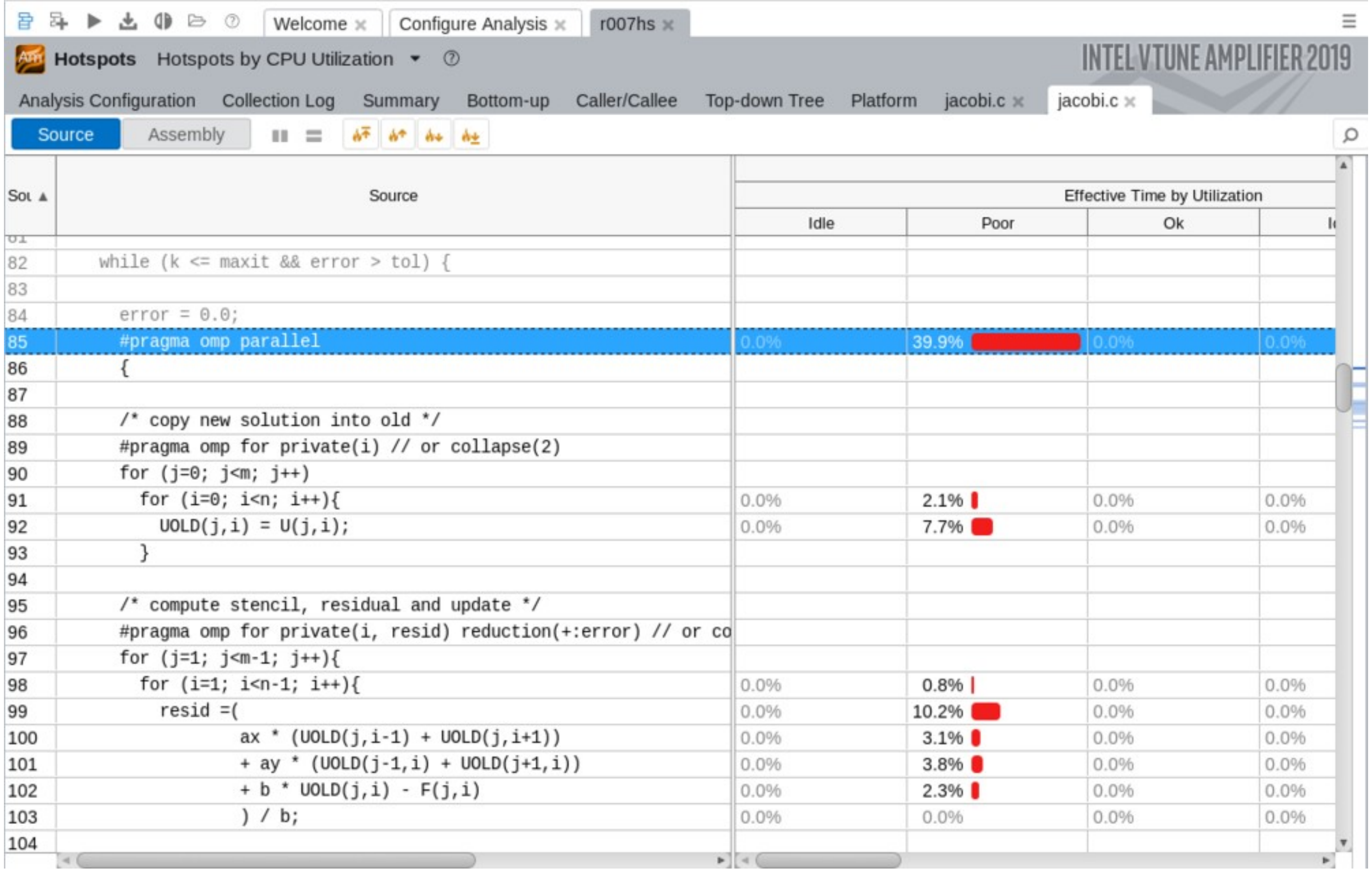
- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R
- Working knowledge of C or Fortran
- Familiarity with OpenMP and MPI

Type: Hands-on
Target audience: Programmers
Must: This session is important for programmers who want to optimize their code for usage on a cluster.

Organized by UCLouvain/CISM

Registration Participants 6 / 60 Register

Contact egs-cism@listes.uclouvain.be
☎ 0494424767

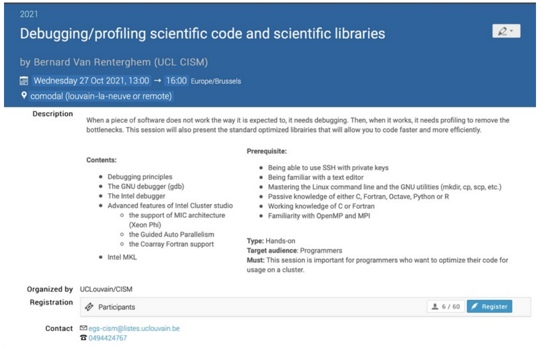


Sou ▲	Source	Effective Time by Utilization			
		Idle	Poor	Ok	I
82	while (k <= maxit && error > tol) {				
83					
84	error = 0.0;				
85	#pragma omp parallel	0.0%	39.9%	0.0%	0.0%
86	{				
87					
88	/* copy new solution into old */				
89	#pragma omp for private(i) // or collapse(2)				
90	for (j=0; j<m; j++)				
91	for (i=0; i<n; i++){	0.0%	2.1%	0.0%	0.0%
92	UOLD(j,i) = U(j,i);	0.0%	7.7%	0.0%	0.0%
93	}				
94					
95	/* compute stencil, residual and update */				
96	#pragma omp for private(i, resid) reduction(+:error) // or co				
97	for (j=1; j<m-1; j++){				
98	for (i=1; i<n-1; i++){	0.0%	0.8%	0.0%	0.0%
99	resid =(	0.0%	10.2%	0.0%	0.0%
100	ax * (UOLD(j,i-1) + UOLD(j,i+1))	0.0%	3.1%	0.0%	0.0%
101	+ ay * (UOLD(j-1,i) + UOLD(j+1,i))	0.0%	3.8%	0.0%	0.0%
102	+ b * UOLD(j,i) - F(j,i)	0.0%	2.3%	0.0%	0.0%
103	) / b;	0.0%	0.0%	0.0%	0.0%
104					

Incorporate benchmarks in your tests

9. Debug cleverly

Use a debugger



2021

Debugging/profiling scientific code and scientific libraries

by Bernard Van Renterghem (UCL CISM)

Wednesday 27 Oct 2021, 13:00 → 16:00 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description

When a piece of software does not work the way it is expected to, it needs debugging. Then, when it works, it needs profiling to remove the bottlenecks. This session will also present the standard optimized librairies that will allow you to code faster and more efficiently.

Contents:

- Debugging principles
- The GNU debugger (gdb)
- The Intel debugger
- Advanced features of Intel Cluster studio
  - the support of MIC architecture (Xeon Phi)
  - the Guided Auto Parallelism
  - the Coarray Fortran support
- Intel MKL

Prerequisite:

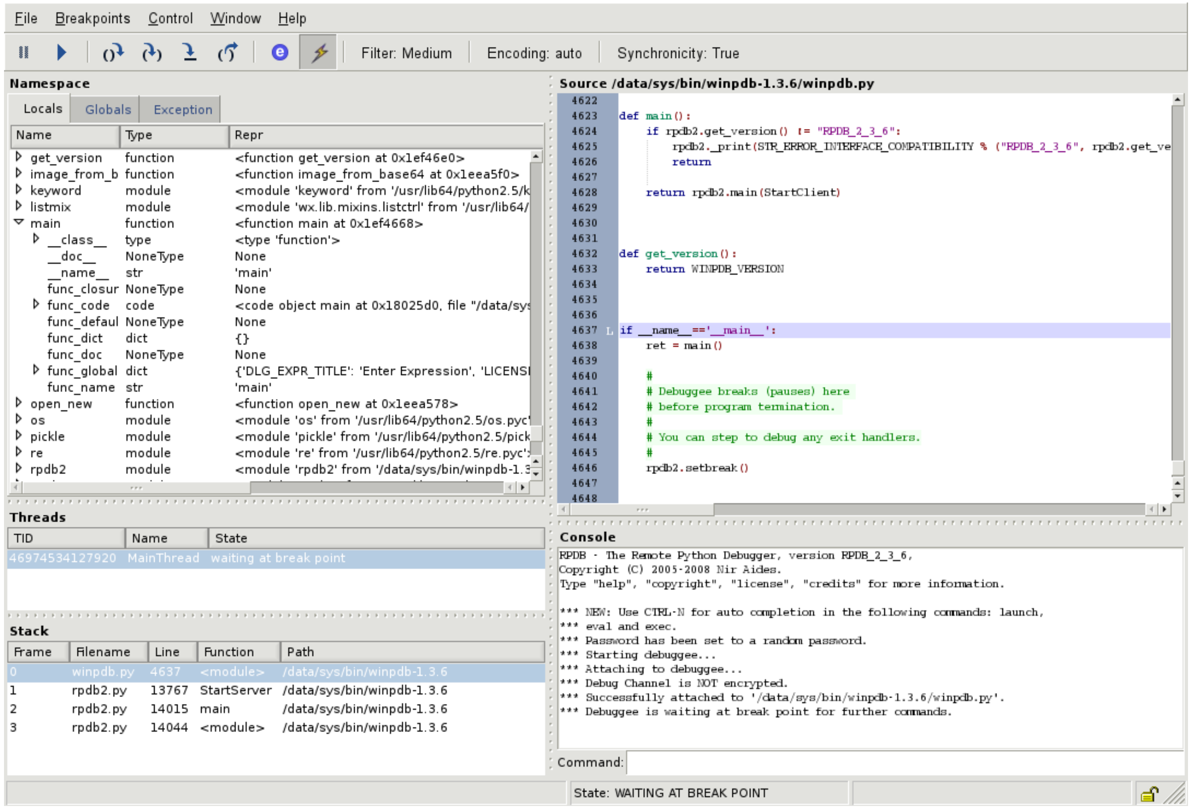
- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R
- Working knowledge of C or Fortran
- Familiarity with OpenMP and MPI

Type: Hands-on
Target audience: Programmers
Must: This session is important for programmers who want to optimize their code for usage on a cluster.

Organized by UCLouvain/CISM

Registration Participants 6 / 60 Register

Contact egs-cism@listes.uclouvain.be
☎ 0494424767



Filter: Medium | Encoding: auto | Synchronicity: True

Namespace

Locals | Globals | Exception

Name	Type	Repr
get_version	function	<function get_version at 0x1ef46e0>
image_from_b	function	<function image_from_base64 at 0x1eea5f0>
keyword	module	<module 'keyword' from '/usr/lib64/python2.5/k
listmix	module	<module 'wx.lib.mixins.listctrl' from '/usr/lib64/
main	function	<function main at 0x1ef4668>
__class__	type	<type 'function'>
__doc__	NoneType	None
__name__	str	'main'
func_closur	NoneType	None
func_code	code	<code object main at 0x18025d0, file "/data/sys
func_defaul	NoneType	None
func_dict	dict	{}
func_doc	NoneType	None
func_global	dict	{'DLG_EXPR_TITLE': 'Enter Expression', 'LICENSI
func_name	str	'main'
open_new	function	<function open_new at 0x1eea578>
os	module	<module 'os' from '/usr/lib64/python2.5/os.pyc'
pickle	module	<module 'pickle' from '/usr/lib64/python2.5/pick
re	module	<module 're' from '/usr/lib64/python2.5/re.pyc'>
rpdb2	module	<module 'rpdb2' from '/data/sys/bin/winpdb-1.3

Source /data/sys/bin/winpdb-1.3.6/winpdb.py

```

def main():
    if rpdb2.get_version() != "RPDB_2_3_6":
        rpdb2._print(STR_ERROR_INTERFACE_COMPATIBILITY % ("RPDB_2_3_6", rpdb2.get_ve
        return

    return rpdb2.main(StartClient)



def get_version():
    return WINPDB_VERSION



if __name__=='__main__':
    ret = main()

    #
    # Debuggee breaks (pauses) here
    # before program termination.
    #
    # You can step to debug any exit handlers.
    #
    rpdb2.setbreak()


```

Threads

TID	Name	State
46974534127920	MainThread	waiting at break point

Stack

Frame	Filename	Line	Function	Path
0	winpdb.py	4637	<module>	/data/sys/bin/winpdb-1.3.6
1	rpdb2.py	13767	StartServer	/data/sys/bin/winpdb-1.3.6
2	rpdb2.py	14015	main	/data/sys/bin/winpdb-1.3.6
3	rpdb2.py	14044	<module>	/data/sys/bin/winpdb-1.3.6

Console

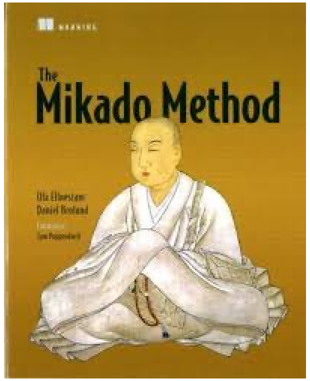
```
RPDB - The Remote Python Debugger, version RPDB_2_3_6,
Copyright (C) 2005-2008 Nir Aides.
Type "help", "copyright", "license", "credits" for more information.

*** NEW: Use CTRL-N for auto completion in the following commands: launch,
*** eval and exec.
*** Password has been set to a random password.
*** Starting debuggee...
*** Attaching to debuggee...
*** Debug Channel is NOT encrypted.
*** Successfully attached to '/data/sys/bin/winpdb-1.3.6/winpdb.py'.
*** Debuggee is waiting at break point for further commands.
```

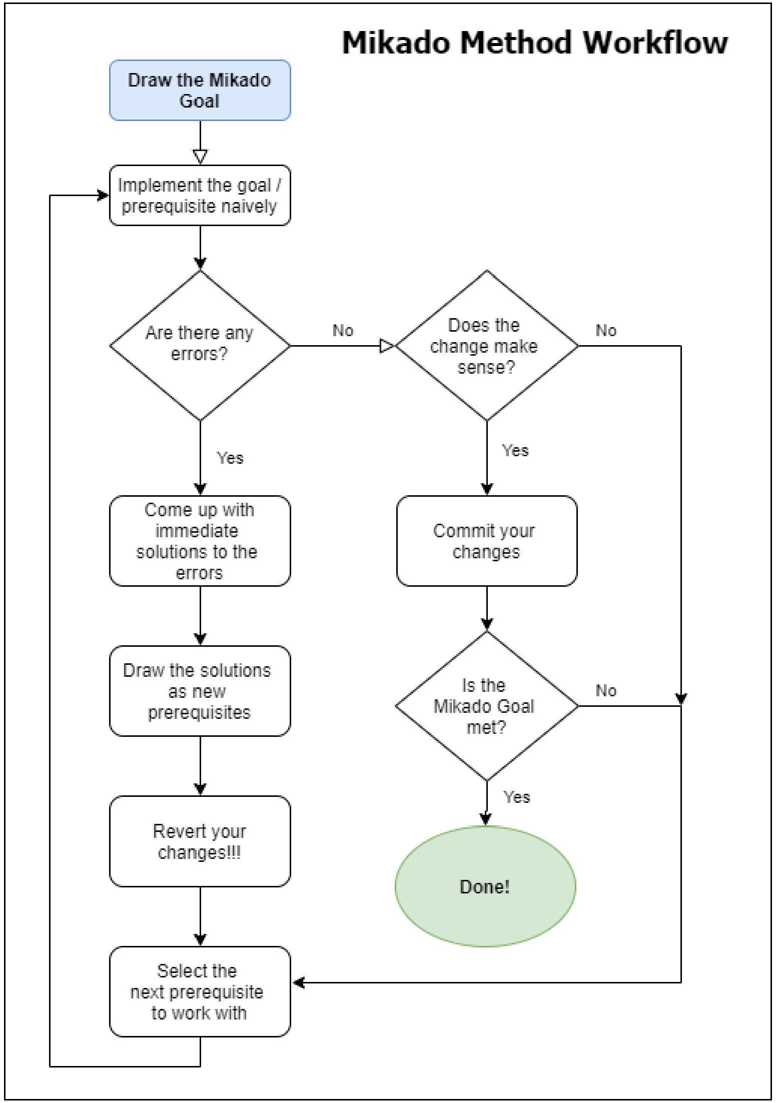
Command:

State: WAITING AT BREAK POINT

9. Debug cleverly

Use a method





Mikado Method Workflow

Draw the Mikado Goal

Implement the goal / prerequisite naively

Are there any errors?

No

Does the change make sense?

No

Yes

Yes

Come up with immediate solutions to the errors

Commit your changes

Draw the solutions as new prerequisites

Is the Mikado Goal met?

No

Yes

Revert your changes!!!

Done!

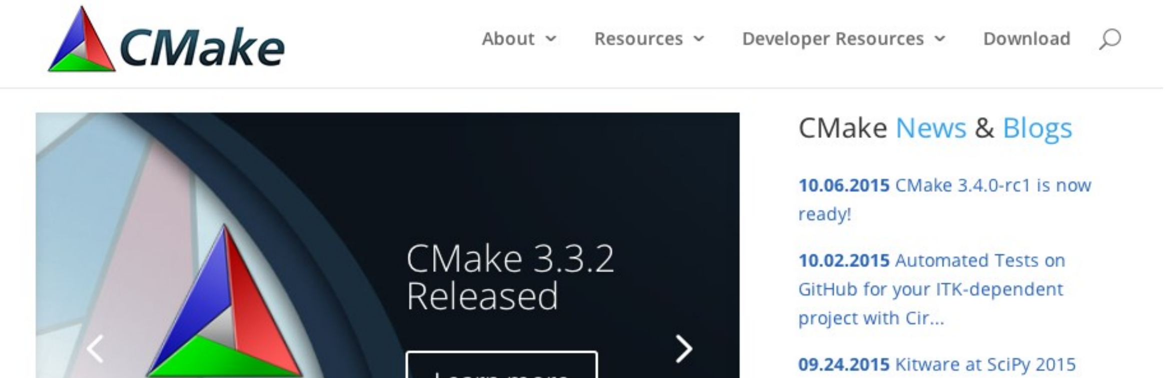
Select the next prerequisite to work with

The Mikado Method, O. Ellnestam & D. Brolund, Manning, 2014

“… to make them … **shareable** …, *efficiently*”

1. Automate the compiling process
2. Learn about containers
3. License your code

1. Automate the compiling process



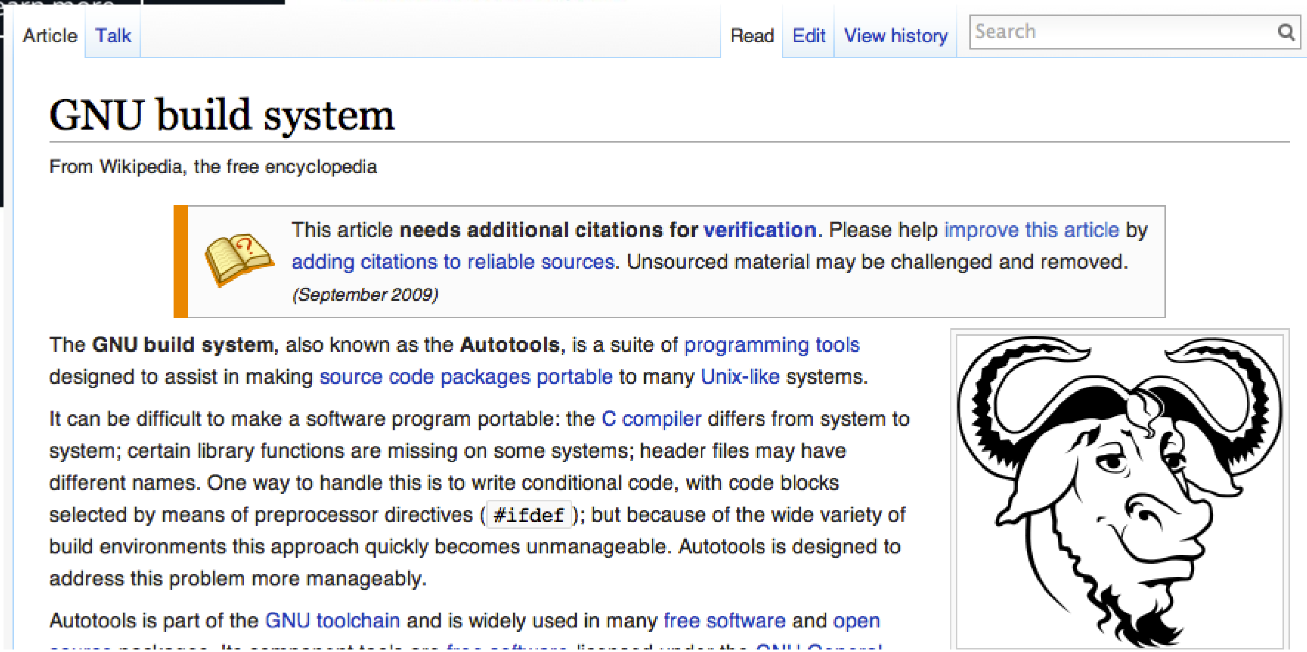
CMake News & Blogs

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)*

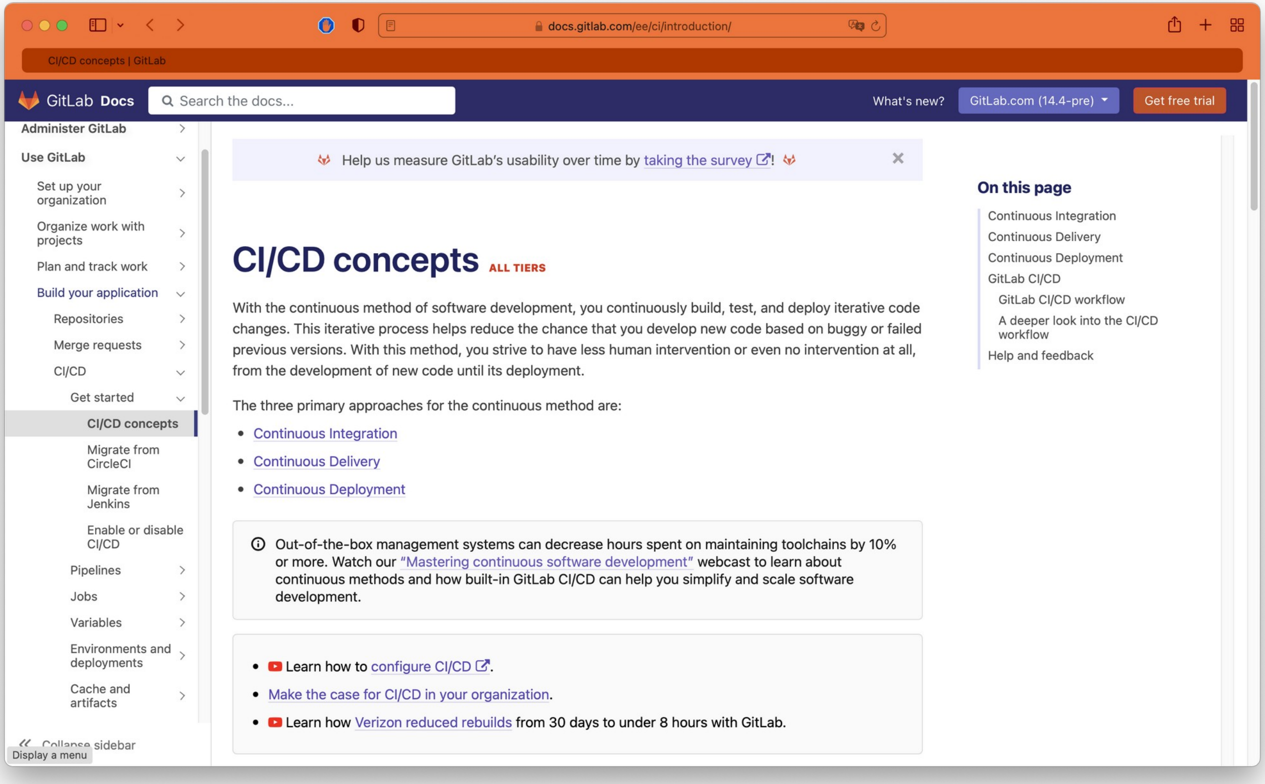
The **GNU build system**, also known as the **Autotools**, is a suite of programming tools designed to assist in making source code packages portable to many Unix-like systems.

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 source packages. Its component tools are free software licensed under the GNU General

It has to compile on all the clusters...

1. Automate the compiling process



Help us measure GitLab's usability over time by taking the survey!

CI/CD concepts ALL TIERS

With the continuous method of software development, you continuously build, test, and deploy iterative code changes. This iterative process helps reduce the chance that you develop new code based on buggy or failed previous versions. With this method, you strive to have less human intervention or even no intervention at all, from the development of new code until its deployment.

The three primary approaches for the continuous method are:

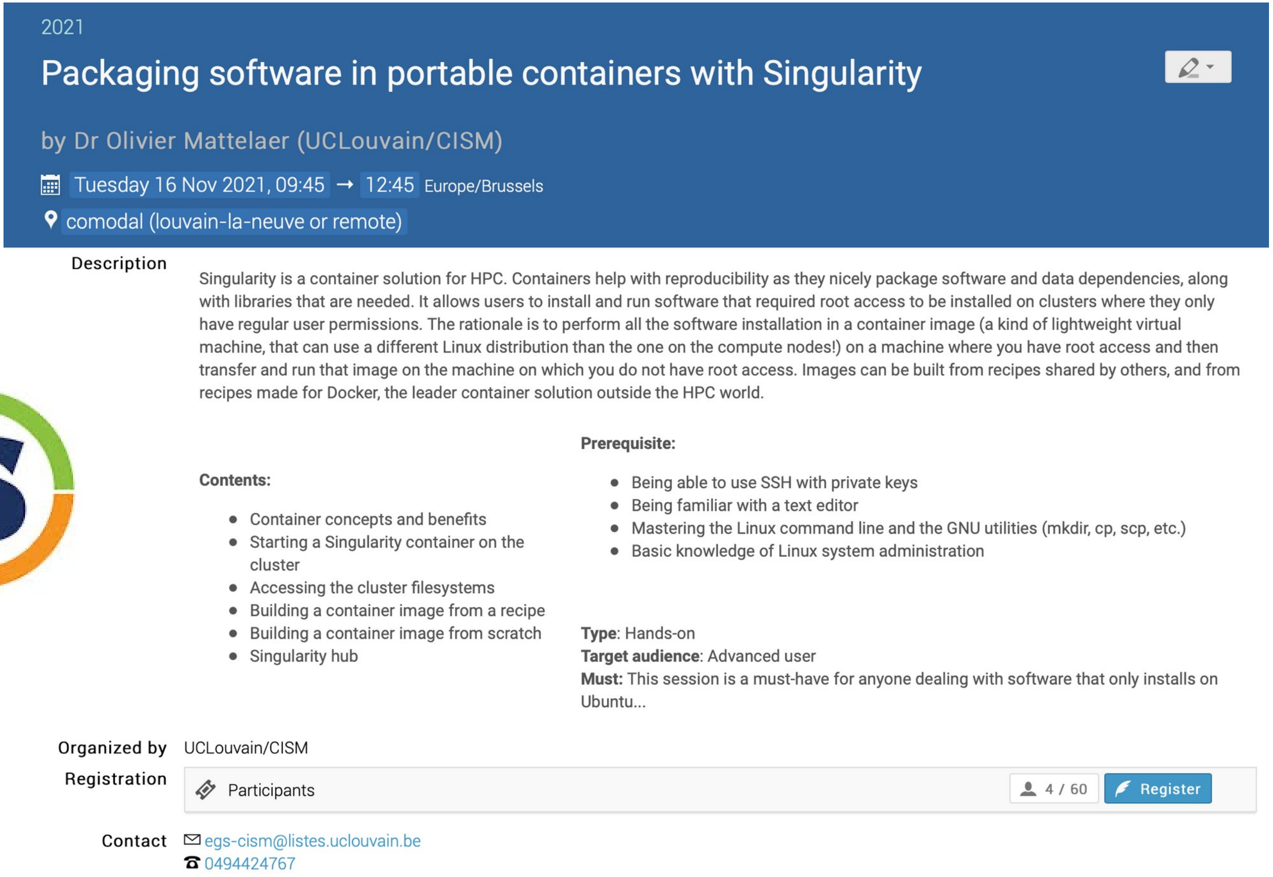
- Continuous Integration
- Continuous Delivery
- Continuous Deployment

Out-of-the-box management systems can decrease hours spent on maintaining toolchains by 10% or more. Watch our "Mastering continuous software development" webcast to learn about continuous methods and how built-in GitLab CI/CD can help you simplify and scale software development.

- Learn how to configure CI/CD.
- Make the case for CI/CD in your organization.
- Learn how Verizon reduced rebuilds from 30 days to under 8 hours with GitLab.

https://docs.gitlab.com/ee/ci/introduction/

2. Learn about containers



2021

Packaging software in portable containers with Singularity

by Dr Olivier Mattelaer (UCLouvain/CISM)

Tuesday 16 Nov 2021, 09:45 → 12:45 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description

Singularity is a container solution for HPC. Containers help with reproducibility as they nicely package software and data dependencies, along with libraries that are needed. It allows users to install and run software that required root access to be installed on clusters where they only have regular user permissions. The rationale is to perform all the software installation in a container image (a kind of lightweight virtual machine, that can use a different Linux distribution than the one on the compute nodes!) on a machine where you have root access and then transfer and run that image on the machine on which you do not have root access. Images can be built from recipes shared by others, and from recipes made for Docker, the leader container solution outside the HPC world.

Contents:

- Container concepts and benefits
- Starting a Singularity container on the cluster
- Accessing the cluster filesystems
- Building a container image from a recipe
- Building a container image from scratch
- Singularity hub

Prerequisite:

- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Basic knowledge of Linux system administration

Type: Hands-on
Target audience: Advanced user
Must: This session is a must-have for anyone dealing with software that only installs on Ubuntu...

Organized by UCLouvain/CISM

Registration

Contact egs-cism@listes.uclouvain.be
☎ 0494424767

3. License 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

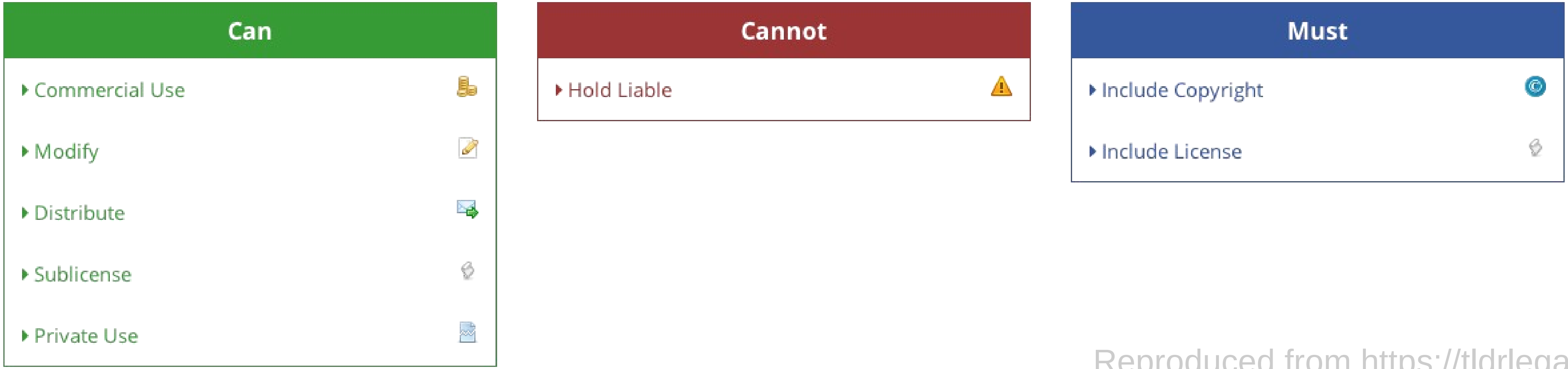
3. License your code: e.g. MIT

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 Softwarre.

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

- Commercial Use
- Modify
- Distribute
- Sublicense
- Private Use

Cannot

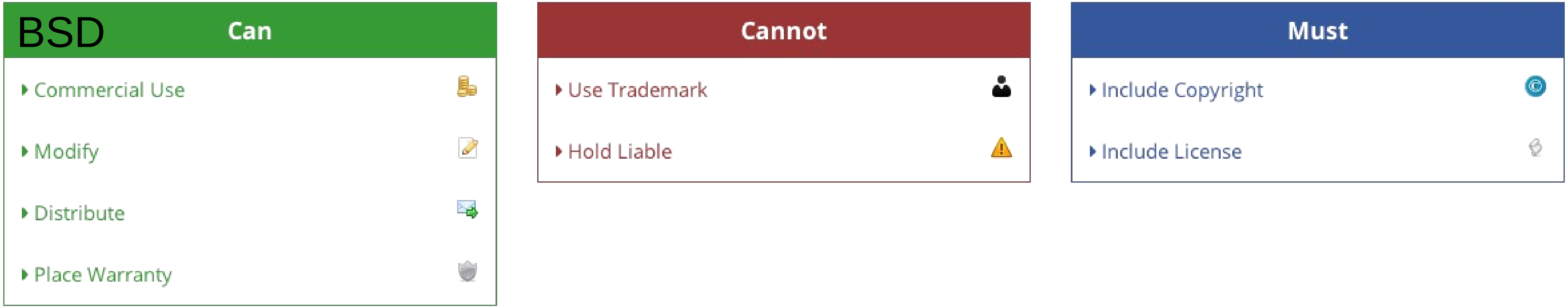
- Hold Liable

Must

- Include Copyright
- Include License

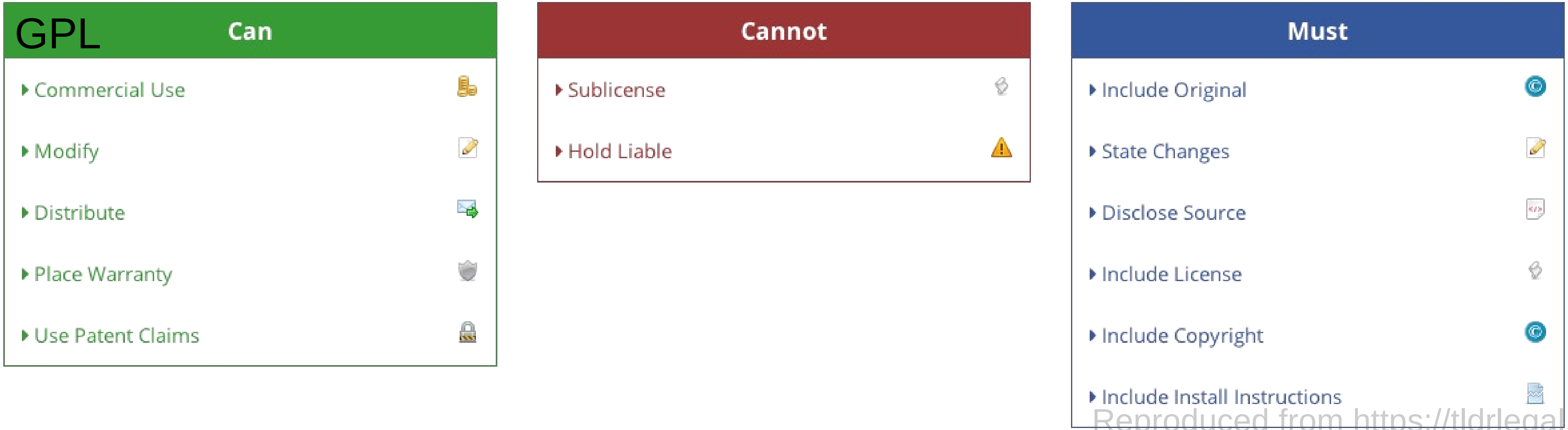
Reproduced from https://tldrlegal.com

3. License your code: e.g. BSD, GPL



BSD

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

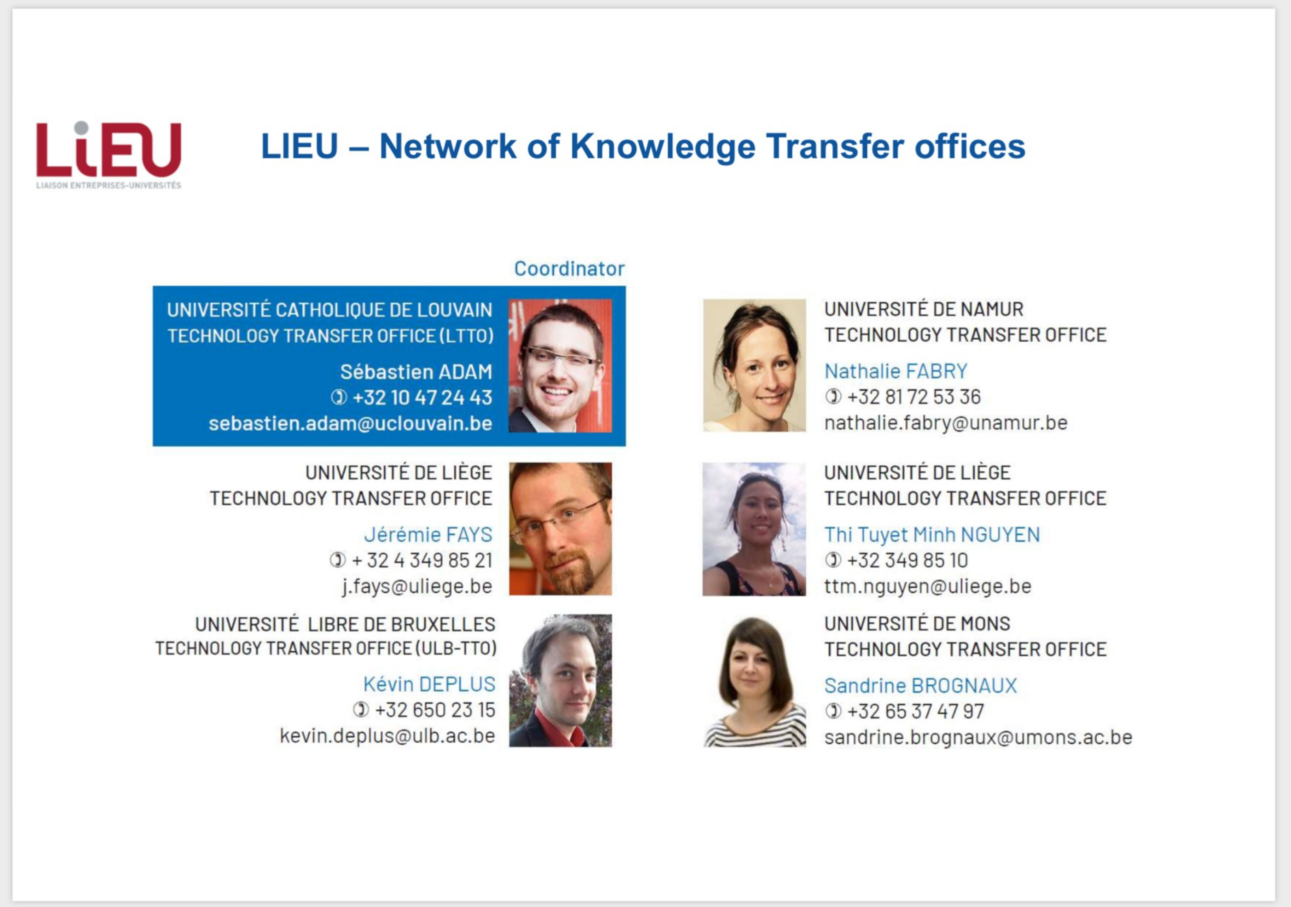


GPL

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

3. License your code: finding help



LiEU
LIAISON ENTREPRISES-UNIVERSITÉS

LIEU – Network of Knowledge Transfer offices

Coordinator

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
TECHNOLOGY TRANSFER OFFICE (LTTO)
Sébastien ADAM
☎ +32 10 47 24 43
sebastien.adam@uclouvain.be

UNIVERSITÉ DE NAMUR
TECHNOLOGY TRANSFER OFFICE
Nathalie FABRY
☎ +32 81 72 53 36
nathalie.fabry@unamur.be

UNIVERSITÉ DE LIÈGE
TECHNOLOGY TRANSFER OFFICE
Jérémie FAYS
☎ + 32 4 349 85 21
j.fays@uliege.be

UNIVERSITÉ DE LIÈGE
TECHNOLOGY TRANSFER OFFICE
Thi Tuyet Minh NGUYEN
☎ +32 349 85 10
ttm.nguyen@uliege.be

UNIVERSITÉ LIBRE DE BRUXELLES
TECHNOLOGY TRANSFER OFFICE (ULB-TTO)
Kévin DEPLUS
☎ +32 650 23 15
kevin.deplus@ulb.ac.be

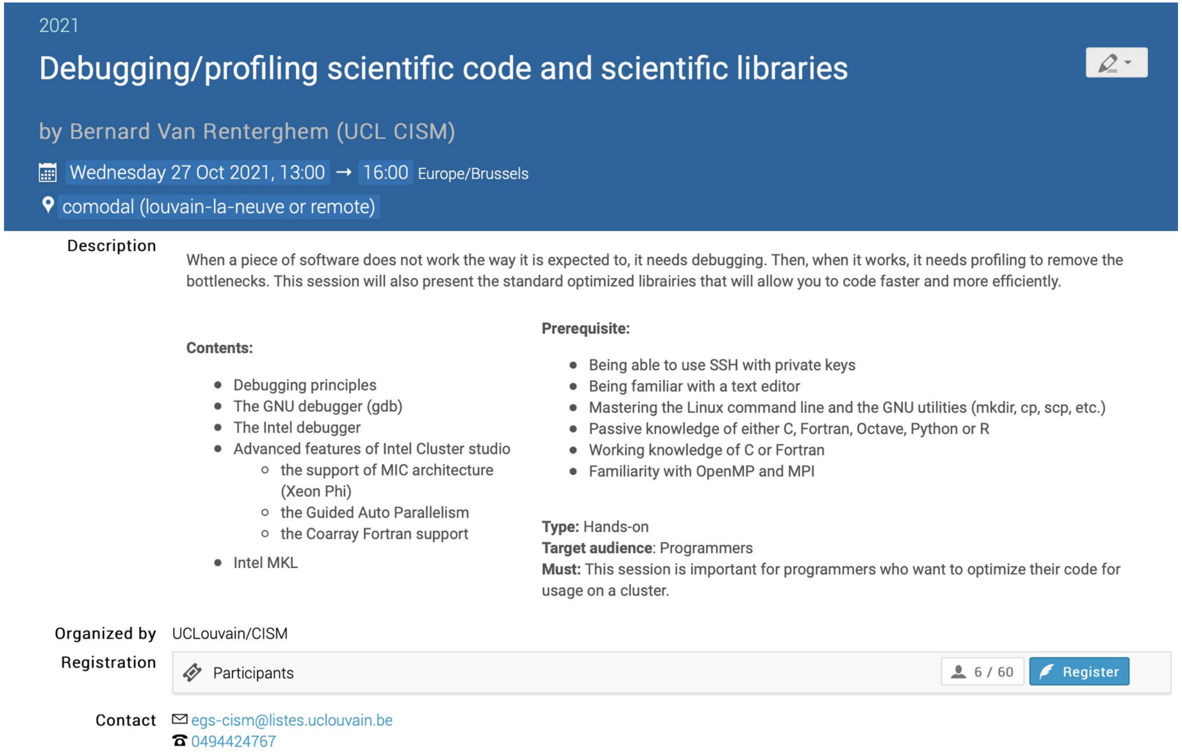
UNIVERSITÉ DE MONS
TECHNOLOGY TRANSFER OFFICE
Sandrine BROGNAUX
☎ +32 65 37 47 97
sandrine.brognaux@umons.ac.be

Slide credit: Sébastien ADAM

“… to make them … **fast** …, *efficiently*”

1. Use optimized libraries
2. Choose the right storage system
3. Think parallel from the start
4. Integrate checkpoint/restart from the start

1. Use optimized libraries



2021

Debugging/profiling scientific code and scientific libraries

by Bernard Van Renterghem (UCL CISM)

Wednesday 27 Oct 2021, 13:00 → 16:00 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description

When a piece of software does not work the way it is expected to, it needs debugging. Then, when it works, it needs profiling to remove the bottlenecks. This session will also present the standard optimized librairies that will allow you to code faster and more efficiently.

Contents:

- Debugging principles
- The GNU debugger (gdb)
- The Intel debugger
- Advanced features of Intel Cluster studio
  - the support of MIC architecture (Xeon Phi)
  - the Guided Auto Parallelism
  - the Coarray Fortran support
- Intel MKL

Prerequisite:

- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R
- Working knowledge of C or Fortran
- Familiarity with OpenMP and MPI

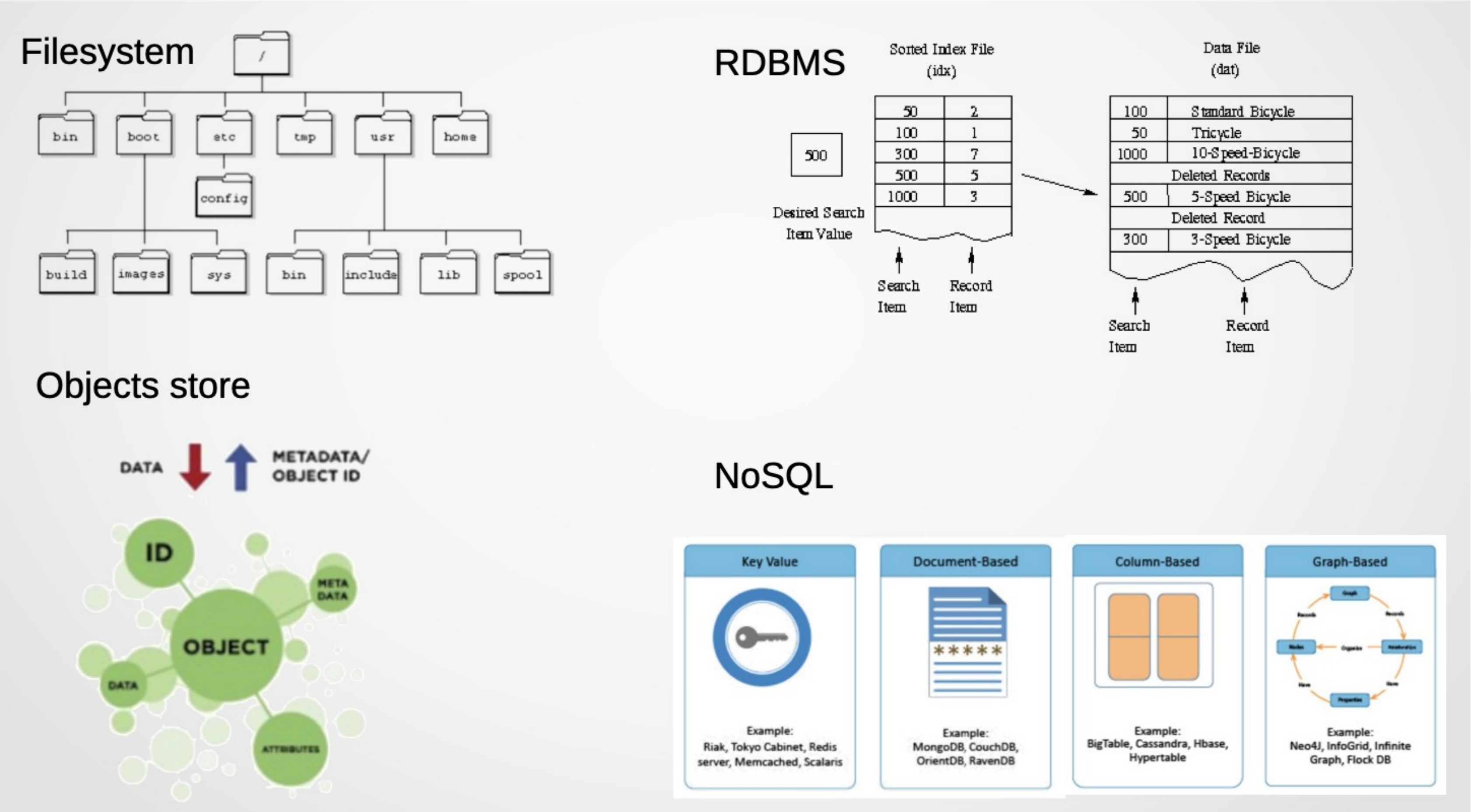
Type: Hands-on
Target audience: Programmers
Must: This session is important for programmers who want to optimize their code for usage on a cluster.

Organized by UCLouvain/CISM

Registration Participants 6 / 60 Register

Contact ✉ egs-cism@listes.uclouvain.be
☎ 0494424767

2. Choose the right storage system



Sorted Index File
(idx)

50	2
100	1
300	7
500	5
1000	3

Data File
(dat)

100	Standard Bicycle
50	Tricycle
1000	10-Speed-Bicycle
Deleted Records	
500	5-Speed Bicycle
Deleted Record	
300	3-Speed Bicycle

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

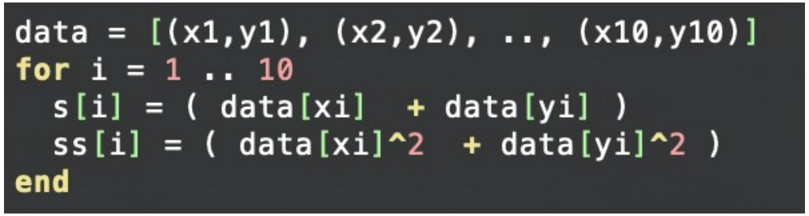
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Example:
BigTable, Cassandra, Hbase, Hypertable

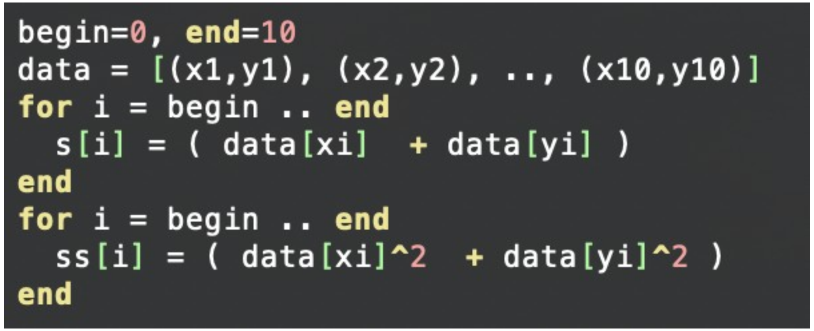
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

3. Think parallel from the start

1. Identify data flows and independent tasks
2. Make data decomposition easy
3. Make work decomposition easy

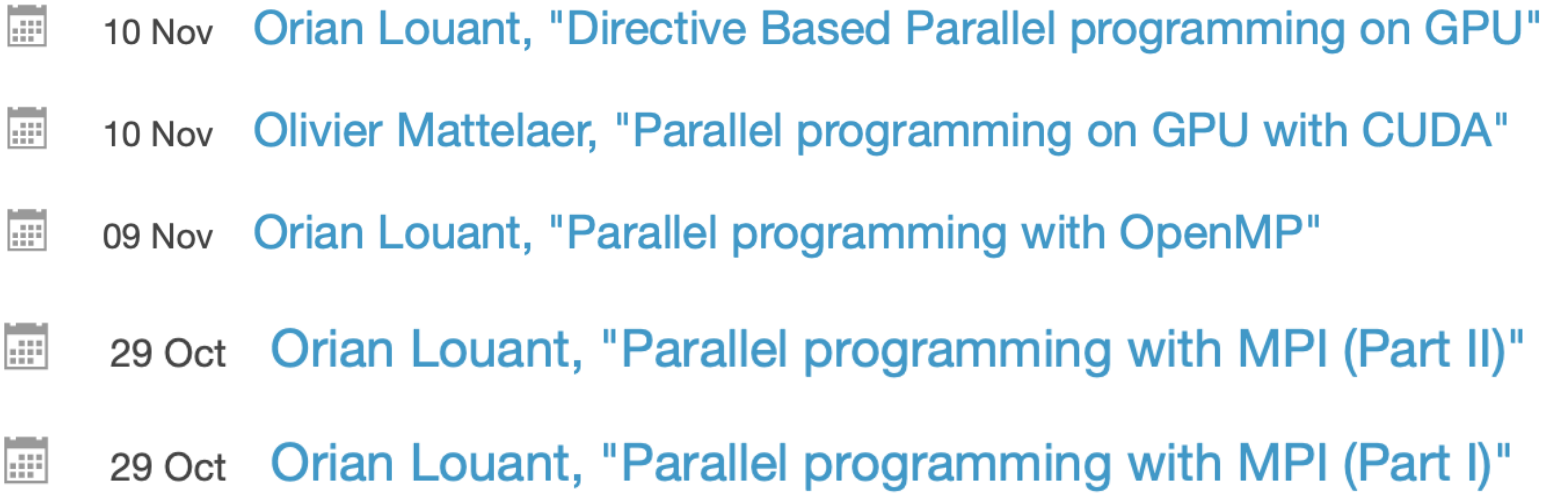


```
data = [(x1,y1), (x2,y2), .., (x10,y10)]
for i = 1 .. 10
  s[i] = ( data[xi]  + data[yi] )
  ss[i] = ( data[xi]^2  + data[yi]^2 )
end
```



```
begin=0, end=10
data = [(x1,y1), (x2,y2), .., (x10,y10)]
for i = begin .. end
  s[i] = ( data[xi]  + data[yi] )
end
for i = begin .. end
  ss[i] = ( data[xi]^2  + data[yi]^2 )
end
```

3. Think parallel from the start



10 Nov Orian Louant, "Directive Based Parallel programming on GPU"

10 Nov Olivier Mattelaer, "Parallel programming on GPU with CUDA"

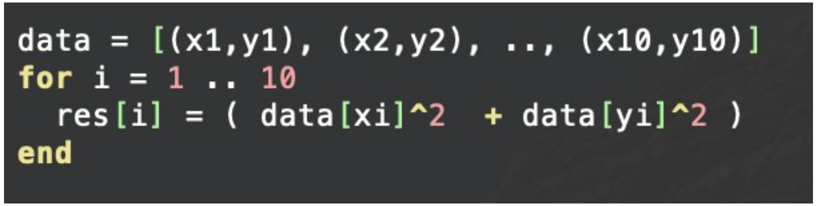
09 Nov Orian Louant, "Parallel programming with OpenMP"

29 Oct Orian Louant, "Parallel programming with MPI (Part II)"

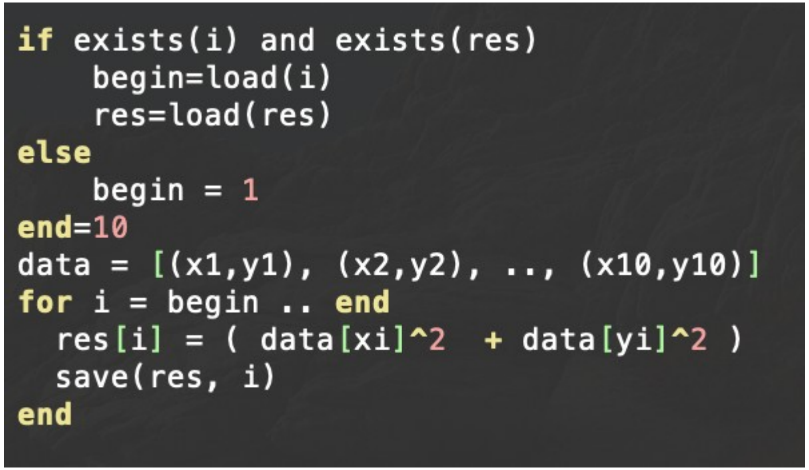
29 Oct Orian Louant, "Parallel programming with MPI (Part I)"

4. Integrate checkpoint/restart from the start

1. Allow starting from a non-initial state
2. Save variables to disk frequently



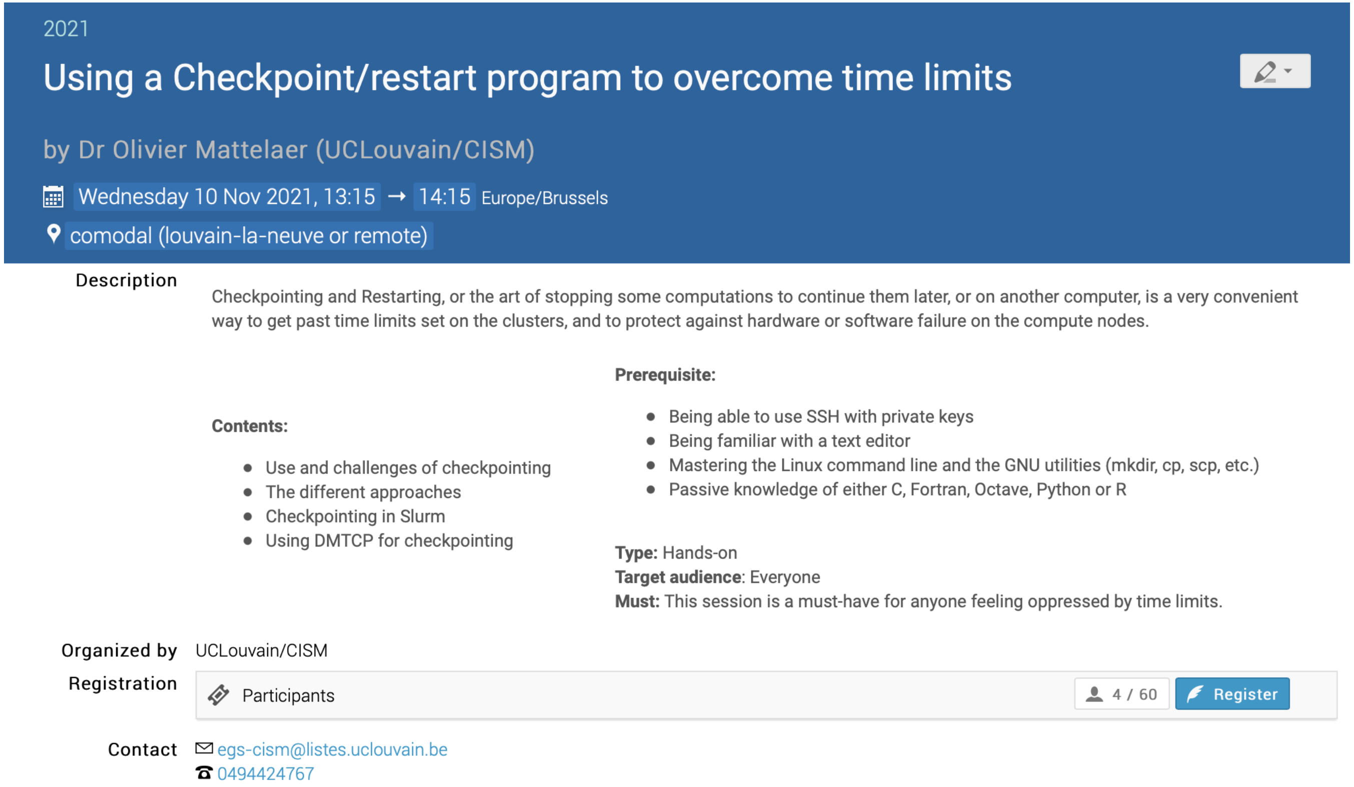
```
data = [(x1,y1), (x2,y2), .., (x10,y10)]
for i = 1 .. 10
  res[i] = ( data[xi]^2  + data[yi]^2 )
end
```



```
if exists(i) and exists(res)
    begin=load(i)
    res=load(res)
else
    begin = 1
end=10
data = [(x1,y1), (x2,y2), .., (x10,y10)]
for i = begin .. end
  res[i] = ( data[xi]^2  + data[yi]^2 )
  save(res, i)
end
```

4. Integrate checkpoint/restart from the start

Because sometimes your code will not be fast enough….



2021

Using a Checkpoint/restart program to overcome time limits

by Dr Olivier Mattelaer (UCLouvain/CISM)

Wednesday 10 Nov 2021, 13:15 → 14:15 Europe/Brussels

comodal (louvain-la-neuve or remote)

Description

Checkpointing and Restarting, or the art of stopping some computations to continue them later, or on another computer, is a very convenient way to get past time limits set on the clusters, and to protect against hardware or software failure on the compute nodes.

Contents:

- Use and challenges of checkpointing
- The different approaches
- Checkpointing in Slurm
- Using DMTCP for checkpointing

Prerequisite:

- Being able to use SSH with private keys
- Being familiar with a text editor
- Mastering the Linux command line and the GNU utilities (mkdir, cp, scp, etc.)
- Passive knowledge of either C, Fortran, Octave, Python or R

Type: Hands-on
Target audience: Everyone
Must: This session is a must-have for anyone feeling oppressed by time limits.

Organized by UCLouvain/CISM

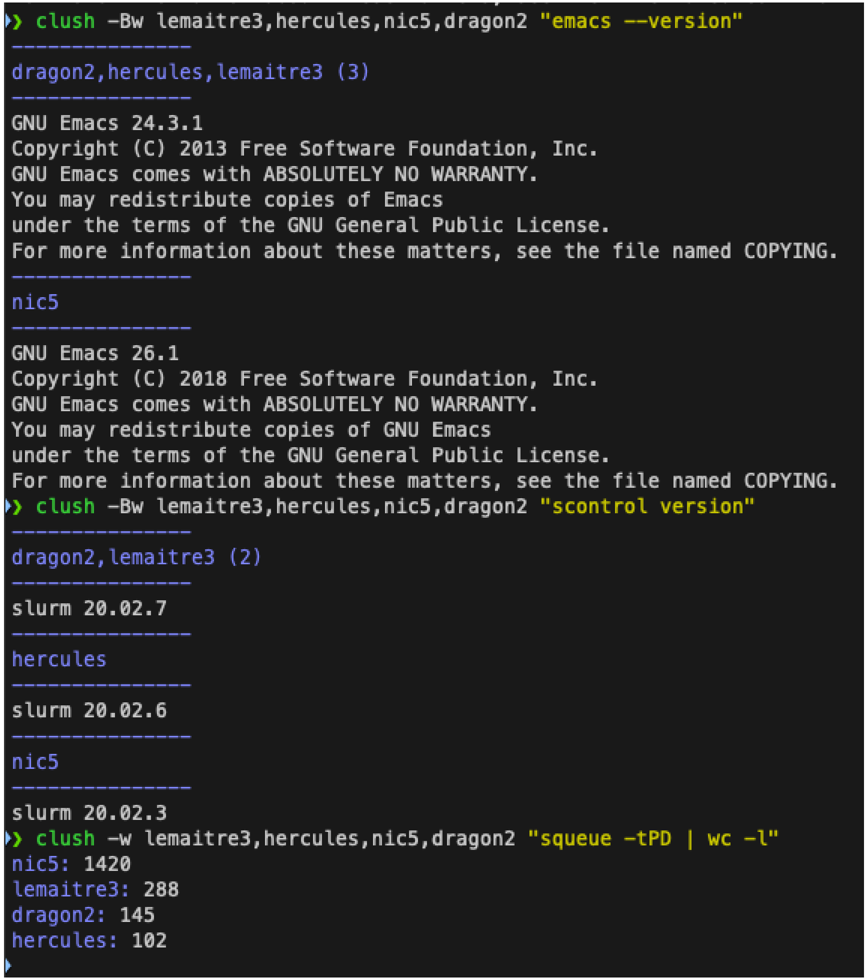
Registration

Contact
✉ egs-cism@listes.uclouvain.be
☎ 0494424767

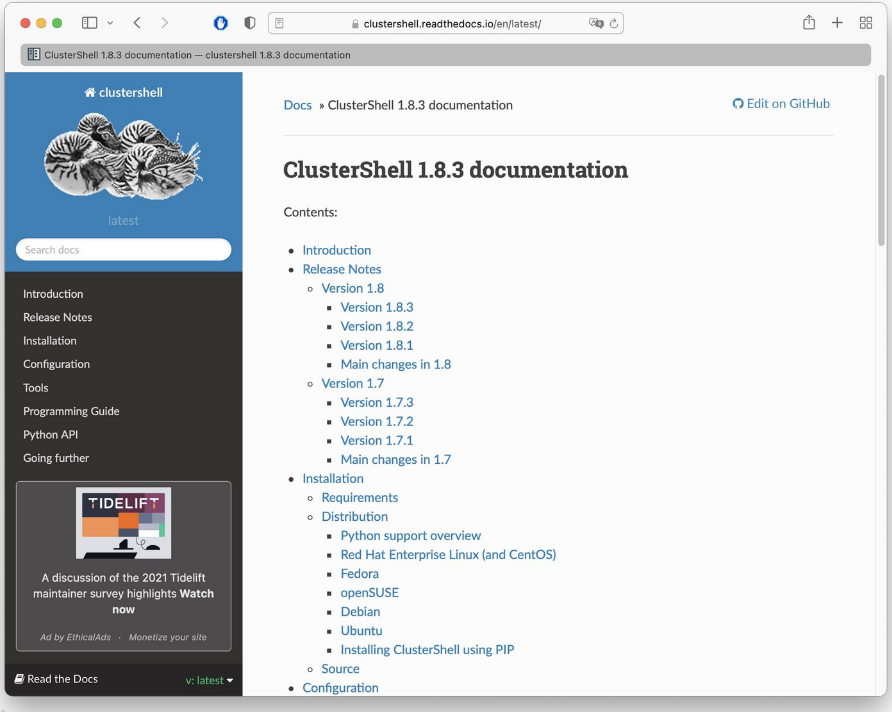
“ …, *efficiently*”

1. Perform “multi-host” SSH
2. Master configuration management
3. Use terminal multiplexing
4. Install software like a boss
5. BACKUPS!

1. Perform “multi-host” SSH



```
❯ clush -Bw lemaitre3,hercules,nic5,dragon2 "emacs --version"
---------------
dragon2,hercules,lemaitre3 (3)
---------------
GNU Emacs 24.3.1
Copyright (C) 2013 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
---------------
nic5
---------------
GNU Emacs 26.1
Copyright (C) 2018 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of GNU Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
❯ clush -Bw lemaitre3,hercules,nic5,dragon2 "scontrol version"
---------------
dragon2,lemaitre3 (2)
---------------
slurm 20.02.7
---------------
hercules
---------------
slurm 20.02.6
---------------
nic5
---------------
slurm 20.02.3
❯ clush -w lemaitre3,hercules,nic5,dragon2 "squeue -tPD | wc -l"
nic5: 1420
lemaitre3: 288
dragon2: 145
hercules: 102
❯
```



clustershell.readthedocs.io/en/latest/
ClusterShell 1.8.3 documentation — clustershell 1.8.3 documentation
clustershell
latest
Search docs
Introduction
Release Notes
Installation
Configuration
Tools
Programming Guide
Python API
Going further
TIDELIFT
A discussion of the 2021 Tidelift maintainer survey highlights Watch now
Ad by EthicalAds · Monetize your site
Read the Docs
v: latest
Edit on GitHub

Docs » ClusterShell 1.8.3 documentation

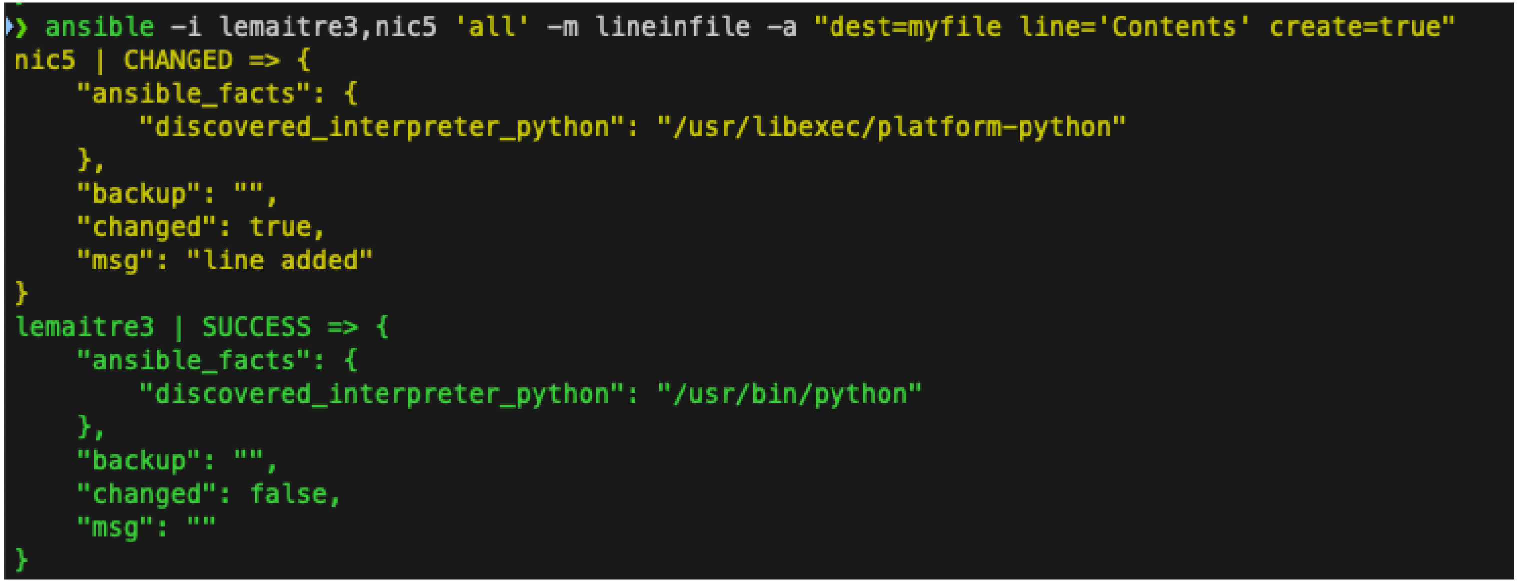
ClusterShell 1.8.3 documentation

Contents:

- Introduction
- Release Notes
  - Version 1.8
    - Version 1.8.3
    - Version 1.8.2
    - Version 1.8.1
    - Main changes in 1.8
  - Version 1.7
    - Version 1.7.3
    - Version 1.7.2
    - Version 1.7.1
    - Main changes in 1.7
- Installation
  - Requirements
  - Distribution
    - Python support overview
    - Red Hat Enterprise Linux (and CentOS)
    - Fedora
    - openSUSE
    - Debian
    - Ubuntu
    - Installing ClusterShell using PIP
  - Source
- Configuration

https://clustershell.readthedocs.io/en/latest/

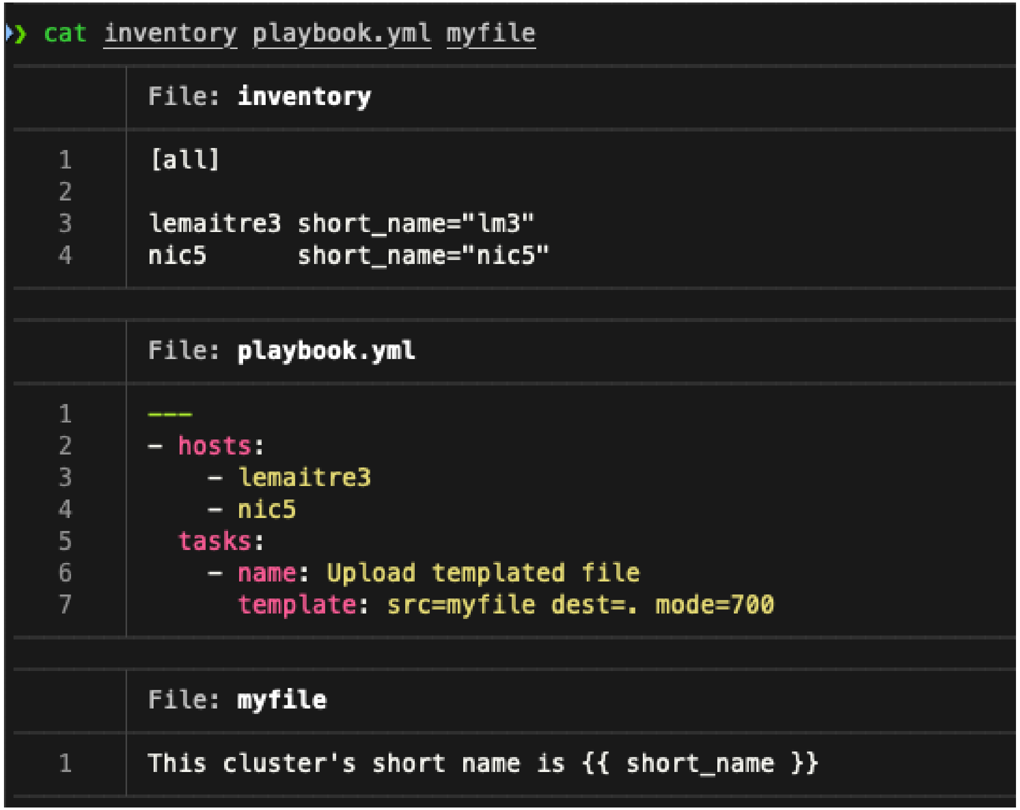
2. Master configuration management



```
❯ ansible -i lemaitre3,nic5 'all' -m lineinfile -a "dest=myfile line='Contents' create=true"
nic5 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "backup": "",
    "changed": true,
    "msg": "line added"
}
lemaitre3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "backup": "",
    "changed": false,
    "msg": ""
}
```

https://www.ansible.com

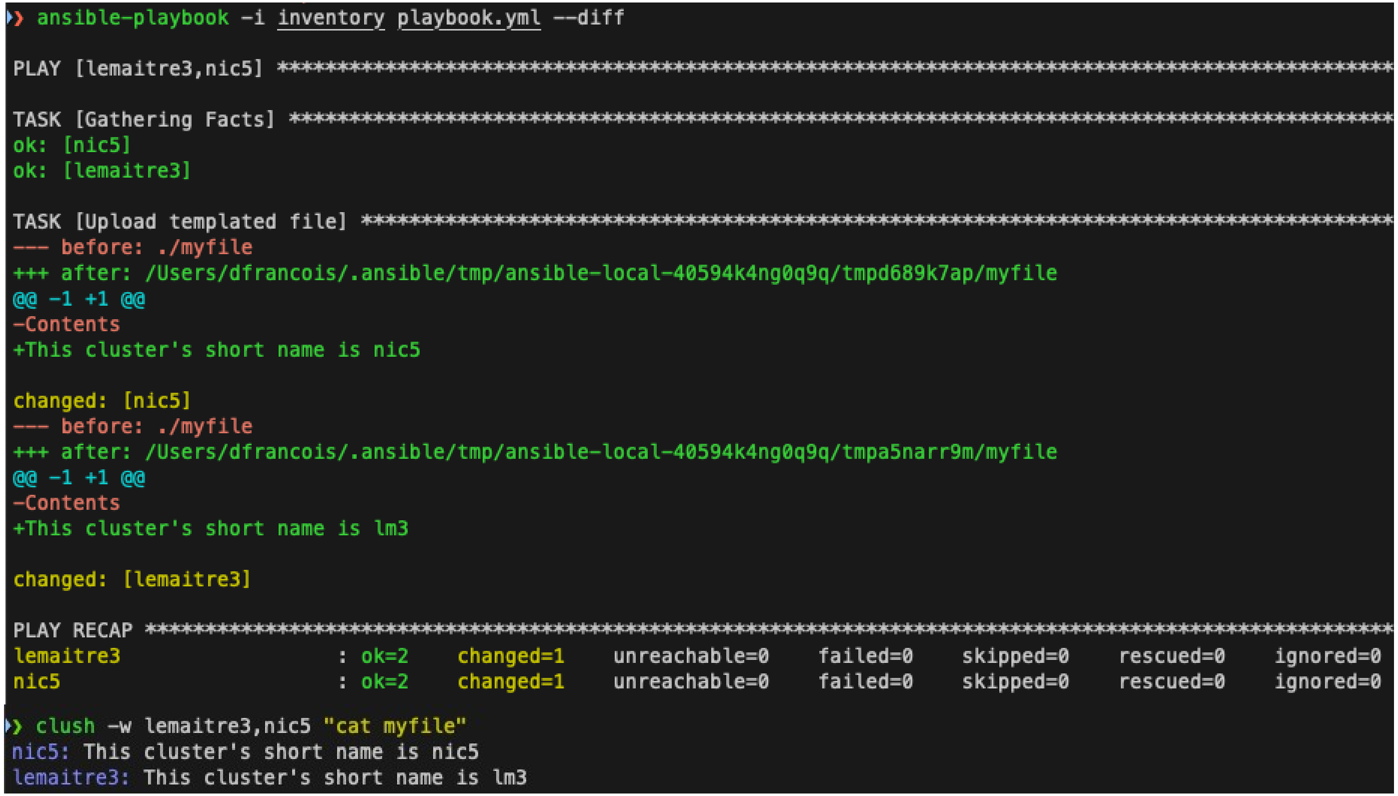
2. Master configuration management



```
❯ cat inventory playbook.yml myfile

File: inventory

[all]

lemaitre3 short_name="lm3"
nic5      short_name="nic5"


File: playbook.yml

---
- hosts:
    - lemaitre3
    - nic5
  tasks:
    - name: Upload templated file
      template: src=myfile dest=. mode=700


File: myfile

This cluster's short name is {{ short_name }}
```

https://www.ansible.com

2. Master configuration management



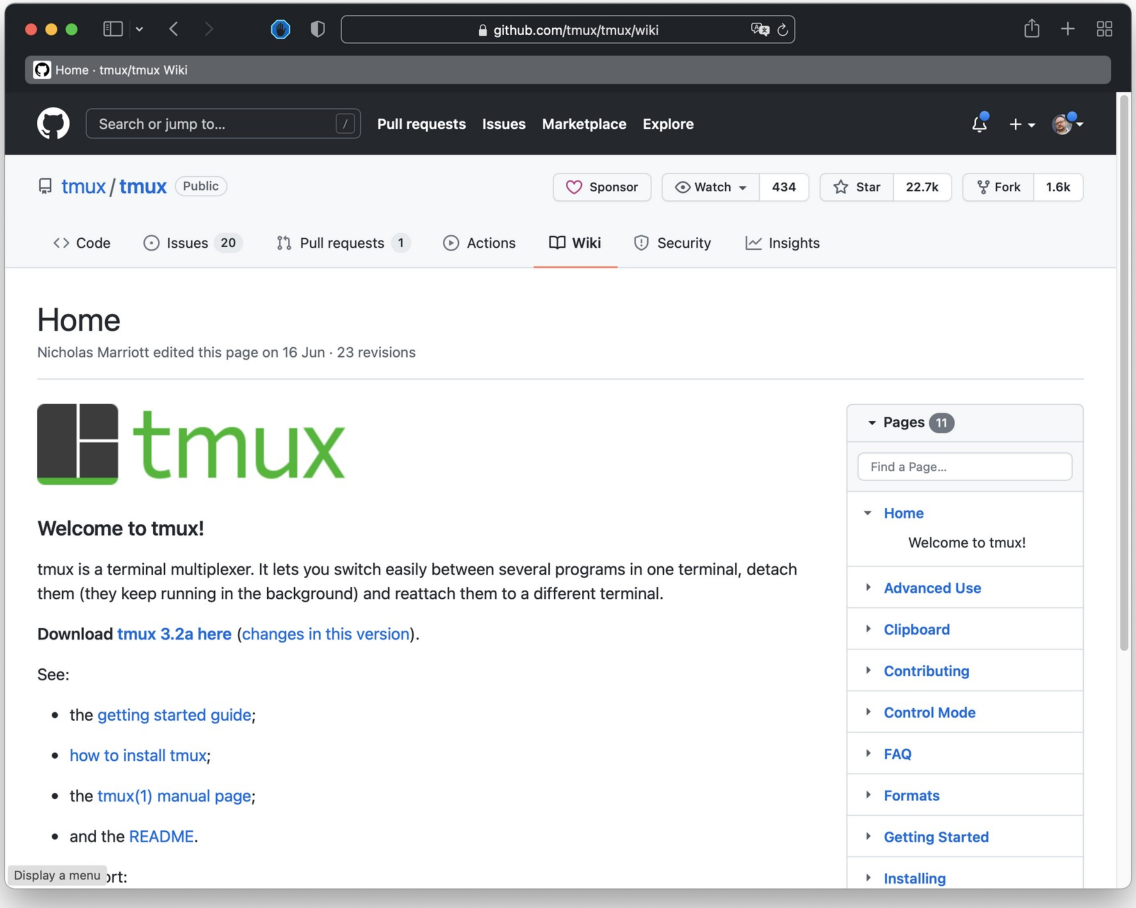
```
❯ ansible-playbook -i inventory playbook.yml --diff

PLAY [lemaitre3,nic5] *********************************************************************************************

TASK [Gathering Facts] ********************************************************************************************
ok: [nic5]
ok: [lemaitre3]

TASK [Upload templated file] **************************************************************************************
--- before: ./myfile
+++ after: /Users/dfrancois/.ansible/tmp/ansible-local-40594k4ng0q9q/tmpd689k7ap/myfile
@@ -1 +1 @@
-Contents
+This cluster's short name is nic5

changed: [nic5]
--- before: ./myfile
+++ after: /Users/dfrancois/.ansible/tmp/ansible-local-40594k4ng0q9q/tmpa5narr9m/myfile
@@ -1 +1 @@
-Contents
+This cluster's short name is lm3

changed: [lemaitre3]

PLAY RECAP ********************************************************************************************************
lemaitre3                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
nic5                       : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

❯ clush -w lemaitre3,nic5 "cat myfile"
nic5: This cluster's short name is nic5
lemaitre3: This cluster's short name is lm3
```

https://www.ansible.com

3. Use terminal multiplexing

Do not let SSH disconnections harm your workflow

(and much more)



Home

Nicholas Marriott edited this page on 16 Jun · 23 revisions

Welcome to tmux!

tmux is a terminal multiplexer. It lets you switch easily between several programs in one terminal, detach them (they keep running in the background) and reattach them to a different terminal.

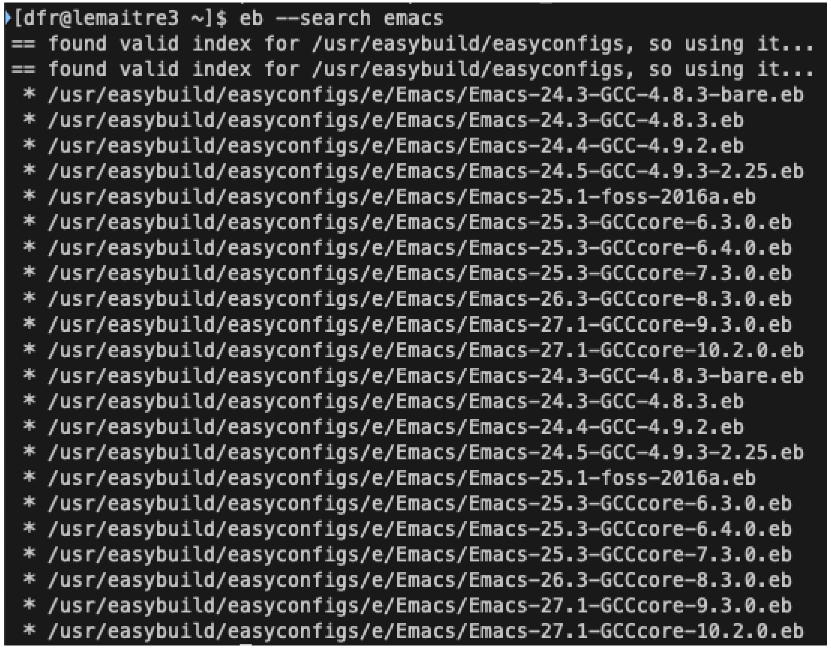
Download tmux 3.2a here (changes in this version).

See:

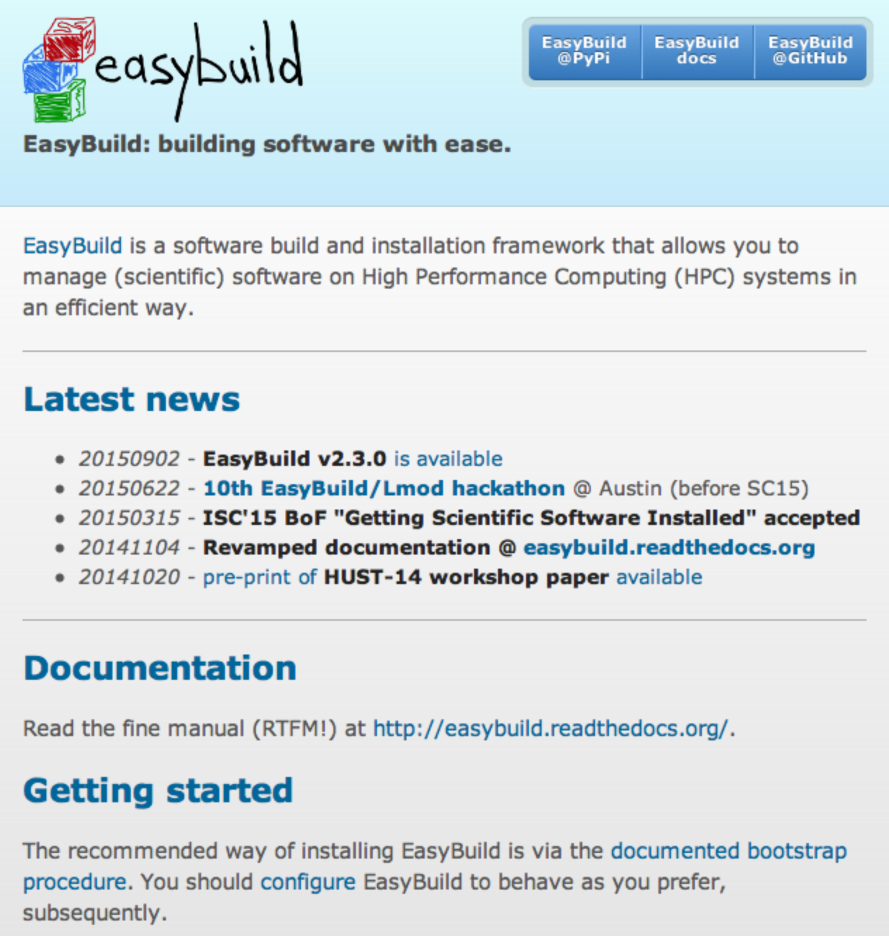
- the getting started guide;
- how to install tmux;
- the tmux(1) manual page;
- and the README.

https://github.com/tmux/tmux/wiki

4. Install software like a boss



```
[dfr@lemaitre3 ~]$ eb --search emacs
== found valid index for /usr/easybuild/easyconfigs, so using it...
== found valid index for /usr/easybuild/easyconfigs, so using it...
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.3-GCC-4.8.3-bare.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.3-GCC-4.8.3.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.4-GCC-4.9.2.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.5-GCC-4.9.3-2.25.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.1-foss-2016a.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-6.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-6.4.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-7.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-26.3-GCCcore-8.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-27.1-GCCcore-9.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-27.1-GCCcore-10.2.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.3-GCC-4.8.3-bare.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.3-GCC-4.8.3.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.4-GCC-4.9.2.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-24.5-GCC-4.9.3-2.25.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.1-foss-2016a.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-6.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-6.4.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-25.3-GCCcore-7.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-26.3-GCCcore-8.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-27.1-GCCcore-9.3.0.eb
 * /usr/easybuild/easyconfigs/e/Emacs/Emacs-27.1-GCCcore-10.2.0.eb
```



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

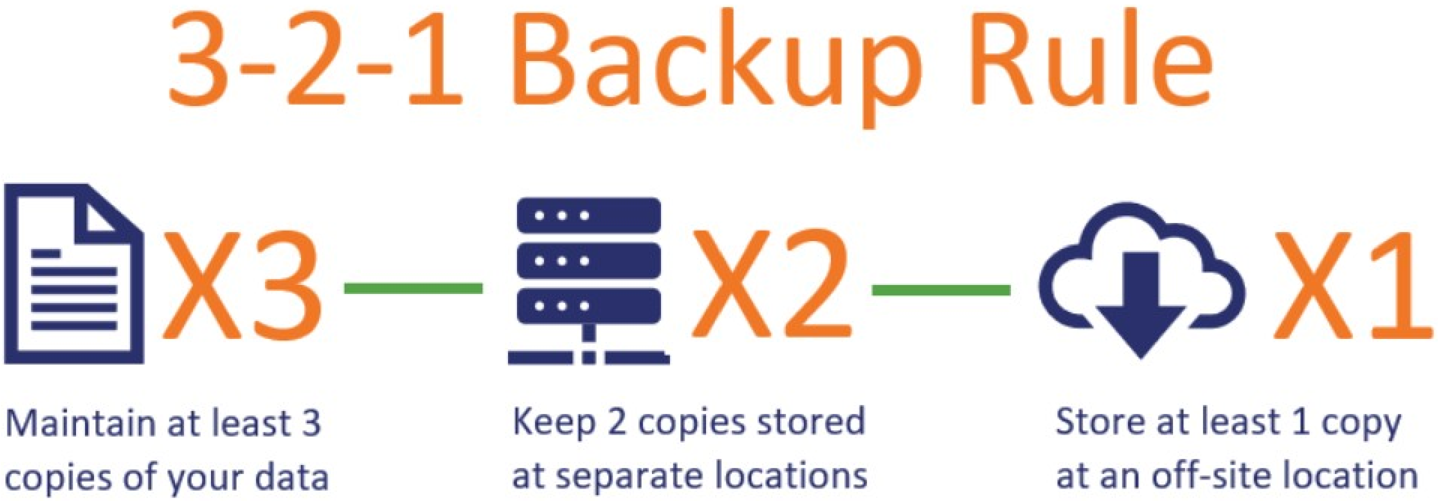
Read the fine manual (RTFM!) at http://easybuild.readthedocs.org/.

Getting started

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

https://easybuild.readthedocs.io/en/latest/

5. BACKUPS!!!



3-2-1 Backup Rule

X3 — X2 — X1

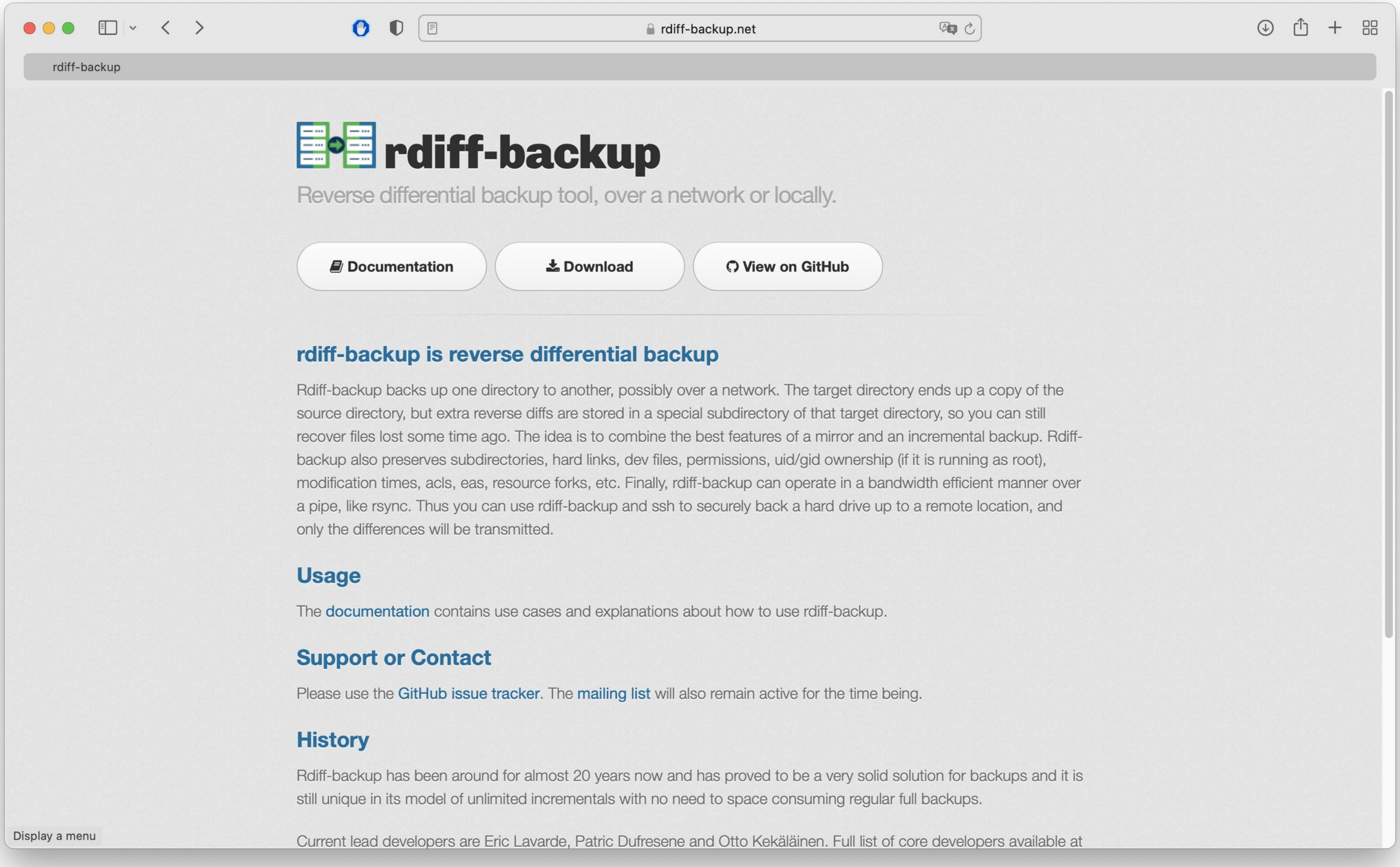
Maintain at least 3 copies of your data

Keep 2 copies stored at separate locations

Store at least 1 copy at an off-site location

https://securityboulevard.com/2020/05/3-2-1-backup-rule-the-rule-of-thumb-to-solve-your-data-loss-problems/

5. BACKUPS!!!



rdiff-backup

Reverse differential backup tool, over a network or locally.

rdiff-backup is reverse differential backup

Rdiff-backup backs up one directory to another, possibly over a network. The target directory ends up a copy of the source directory, but extra reverse diffs are stored in a special subdirectory of that target directory, so you can still recover files lost some time ago. The idea is to combine the best features of a mirror and an incremental backup. Rdiff-backup also preserves subdirectories, hard links, dev files, permissions, uid/gid ownership (if it is running as root), modification times, acls, eas, resource forks, etc. Finally, rdiff-backup can operate in a bandwidth efficient manner over a pipe, like rsync. Thus you can use rdiff-backup and ssh to securely back a hard drive up to a remote location, and only the differences will be transmitted.

Usage

The documentation contains use cases and explanations about how to use rdiff-backup.

Support or Contact

Please use the GitHub issue tracker. The mailing list will also remain active for the time being.

History

Rdiff-backup has been around for almost 20 years now and has proved to be a very solid solution for backups and it is still unique in its model of unlimited incrementals with no need to space consuming regular full backups.

Current lead developers are Eric Lavarde, Patric Dufresene and Otto Kekäläinen. Full list of core developers available at

https://rdiff-backup.net

This was:

“A short catalog of *tools*
the professionals are using for
developing and **deploying** programs,
to make them **correct**, **maintainable**, **shareable**, and **fast**,
efficiently.”

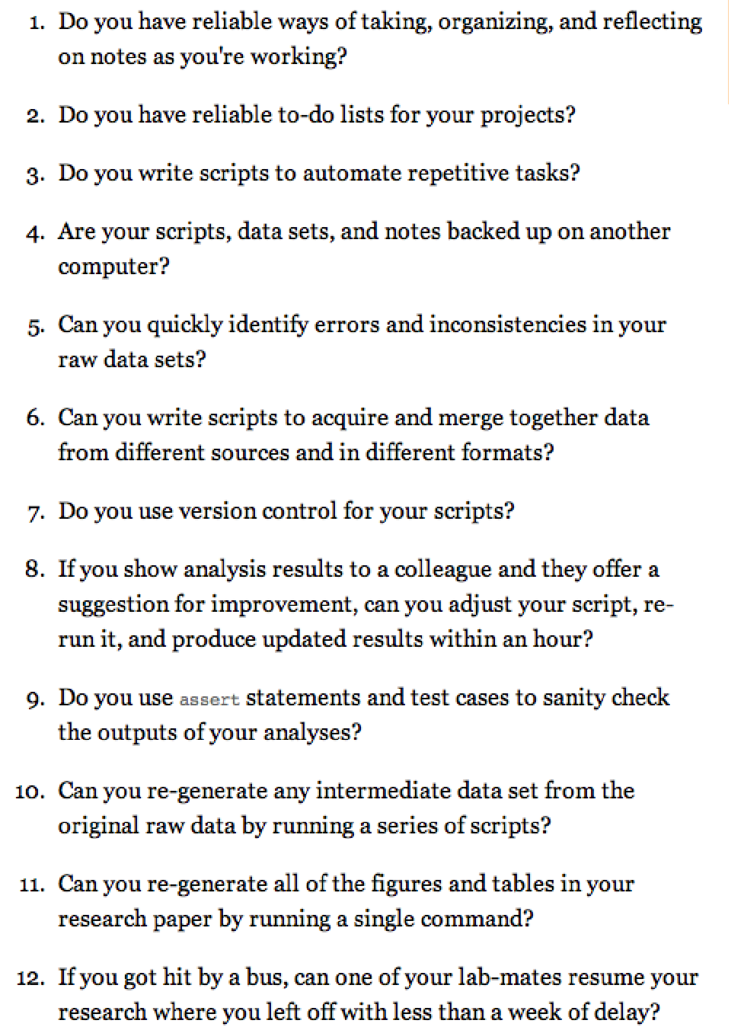
We discussed:

- good practices
- important choices
- useful tools
- practical references

The “Phillip test” (by Philip Guo)

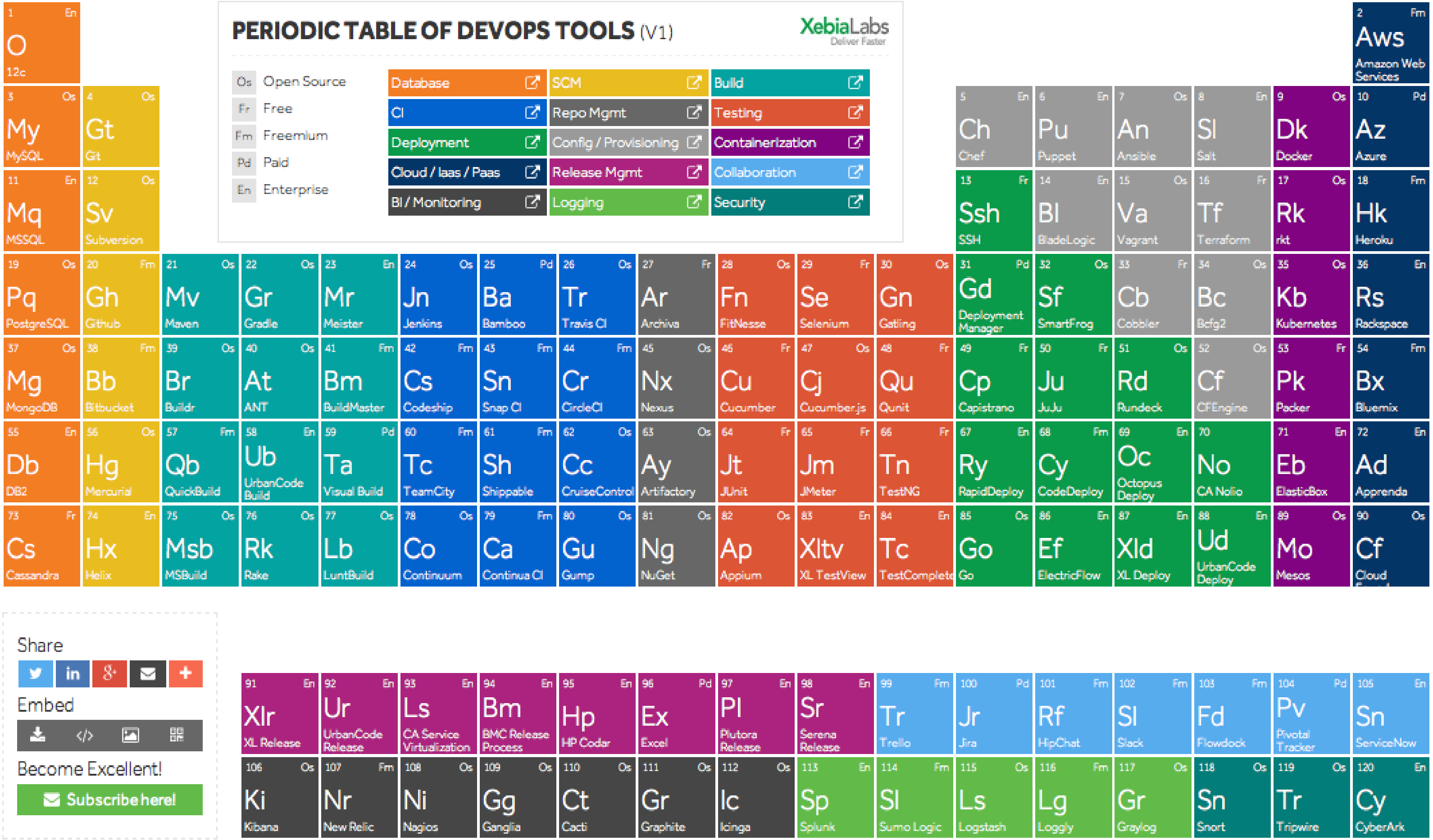
12 simple questions
ordered by 'difficulty'
measures quality of organization
for research programming

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?

Work faster & 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 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
			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

Share

Embed

Become Excellent!

Subscribe here!