

# Bu17dBot

And Continuous Integration  
RegCM4 experiences

S. Cozzini/A. Messina/G. Giuliani

Warning: Some slides/ideas  
stolen by Willie  
<willie@issdu.com.tw>

# Agenda

- How do we use BuildBot here?
- What is BuildBot?
- Practices of Continuous Integration
- Example for RegCM code

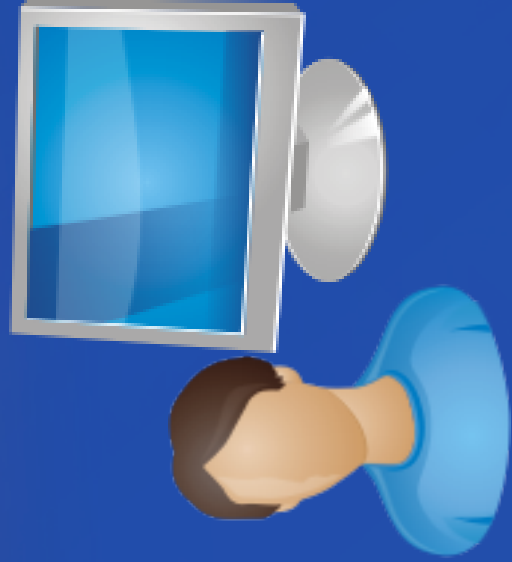
**How do we use  
ButtBot?**



2. Commit



3. Trigger



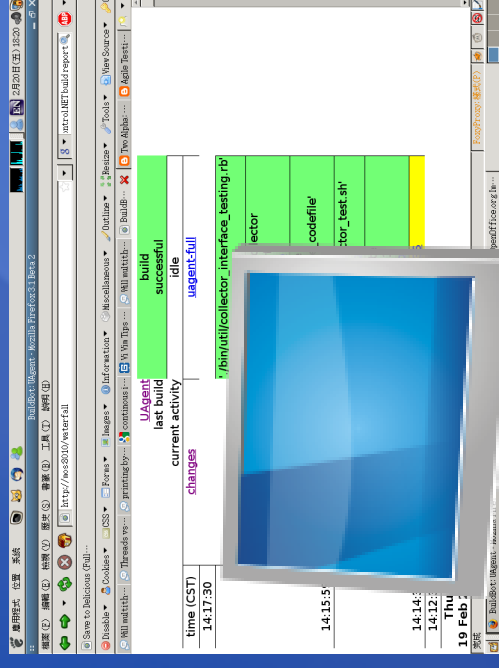
4. Wait & See



Email If Fail



1. make a change on RegCM



Subject: buildbot failure in ICTP BuildBot on  
regcm-conf=debug|mod=openmpi:1.4.3:intel:2011  
Date: Tue, 21 Feb 2012 22:20:39 +0100  
From: buildmaster@gforge.ictp.it  
To: amessina@ictp.it, ggiulian@ictp.it  
The Buildbot has detected a failed build on builder  
regcm-conf=debug|mod=openmpi:1.4.3:intel:2011 while building ICTP  
BuildBot.  
Full details are available at:  
<http://buildbot.ictp.it/builders/regcm-conf%3Ddebug%7Cmod%3Dopenmpi%3A1.4.3%3Aintel%3A2011>  
Buildbot URL: <http://buildbot.ictp.it/>  
Buildslave for this Build: argo-buildbot01  
Build Reason: The web-page 'force build' button was pressed by ''  
Build Source Stamp: 2750  
Blamelist:  
BUILD FAILED: failed compile  
sincerely,  
-The Buildbot

What is BuildBot?

# BuildBot

- ▶ A continuous integration system
- ▶ Help to automate the compile/test cycle
- ▶ Written in Python

# Screenshot

Home - Waterfall\_Grid\_T-Grid\_Console\_Builders\_Recent\_Builds\_Buildslaves\_Changesources - JSON\_API - About

## Waterfall

	last build	current activity	changes
CET			
18:17:50	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:17:31	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:17:07	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:16:56	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:15:34	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:15:24	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:15:06	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:14:55	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:14:34	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:14:05	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:13:50	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing
18:13:40	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile	idle	reqcm- conf=dm mod=openmpi:1.4.3:intel:2011 failed compile
18:13:29	reqcm- conf=debug mod=openmpi:1.4.3:intel:2011 failed Testing	building	reqcm- conf=empty mod=openmpi:1.4.3:intel:2011 failed Testing

waterfall\_help

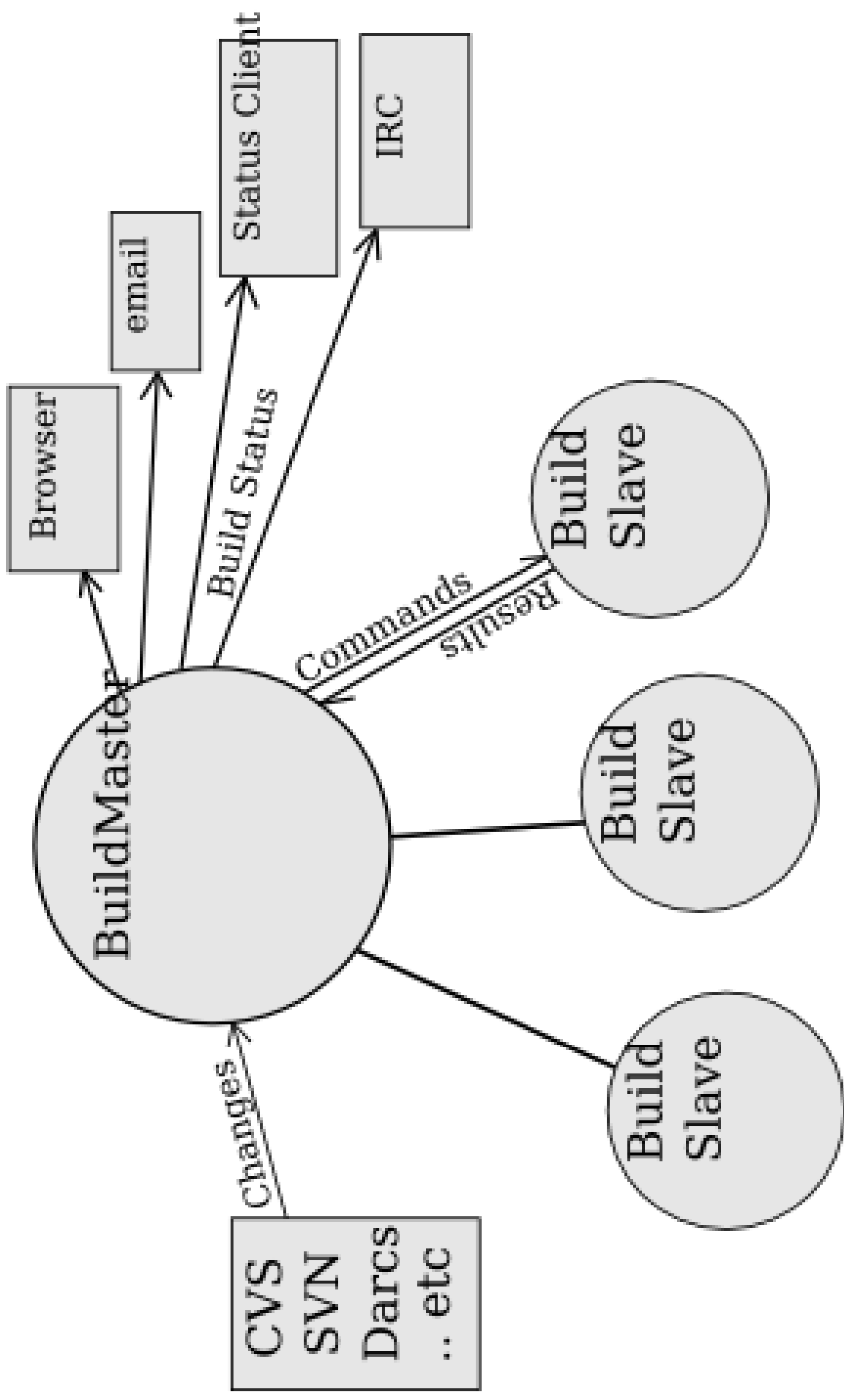


# Screenshot

The screenshot displays a web browser window with several tabs. The active tab is 'Buildbot'. The browser address bar shows 'www.carlosfau.com.ar/nqj/in'. The page content is a 'Grid View' of build jobs. The table below represents the data shown in the screenshot.

Job ID	Job Name	Status	Progress
2750	regcm-conf=REGRESSION band mod=openmpi:1.4.3:intel:2011 (waiting)		
2749	regcm-conf=REGRESSION band mod=openmpi:1.5.1:intel:2011 (waiting)		
2752	regcm-conf=REGRESSION band mod=openmpi:1.5.3:pqi:10.9 (waiting)		
2753	regcm-conf=REGRESSION clm mod=openmpi:1.4.3:intel:2011 (waiting)		
2754	regcm-conf=REGRESSION clm mod=openmpi:1.5.1:intel:2011 (waiting)		
	regcm-conf=REGRESSION clm mod=openmpi:1.5.3:pqi:10.9 (waiting)		
	regcm-conf=REGRESSION debug mod=openmpi:1.4.3:intel:2011 (waiting)		
	regcm-conf=REGRESSION debug mod=openmpi:1.5.1:intel:2011 (waiting)		
	regcm-conf=REGRESSION debug mod=openmpi:1.5.3:pqi:10.9 (waiting)		
	regcm-conf=REGRESSION empty mod=openmpi:1.4.3:intel:2011 (waiting)		
	regcm-conf=REGRESSION empty mod=openmpi:1.5.1:intel:2011 (waiting)		
	regcm-conf=REGRESSION empty mod=openmpi:1.5.3:pqi:10.9 (waiting)		
	regcm-conf=band mod=openmpi:1.4.3:intel:2011 (building)	failed compile	building
	regcm-conf=band mod=openmpi:1.5.1:intel:2011 (building)	failed compile	building
	regcm-conf=band mod=openmpi:1.5.3:pqi:10.9 (building)	failed compile	building
	regcm-conf=clm mod=openmpi:1.4.3:intel:2011 (building)	failed compile	building
	regcm-conf=clm mod=openmpi:1.5.1:intel:2011 (building)	failed compile	building
	regcm-conf=clm mod=openmpi:1.5.3:pqi:10.9 (building)	failed compile	building
	regcm-conf=debug mod=openmpi:1.4.3:intel:2011 (building)	failed compile	building
	regcm-conf=debug mod=openmpi:1.5.1:intel:2011 (building)	OK	OK
	regcm-conf=debug mod=openmpi:1.5.3:pqi:10.9 (building)	OK	OK
	regcm-conf=debug mod=openmpi:1.5.3:pqi:10.9 (building)	failed compile	building

# System architecture



# Similar projects

## ➤ Hudson (JAVA)

The screenshot shows the Hudson web interface in a Microsoft Internet Explorer browser window. The address bar displays <http://kohlsuke.sfbay/hudson/>. The main content area features a large 'Hudson' header and a navigation menu with links for 'New Job', 'Configure', and 'Reload Config.'. Below the header, there are two tables: 'Build Queue' and 'Build Executor Status'.

**Build Queue**

Job	Status
hudson	⊘
jaxb-ri	⊘

**Build Executor Status**

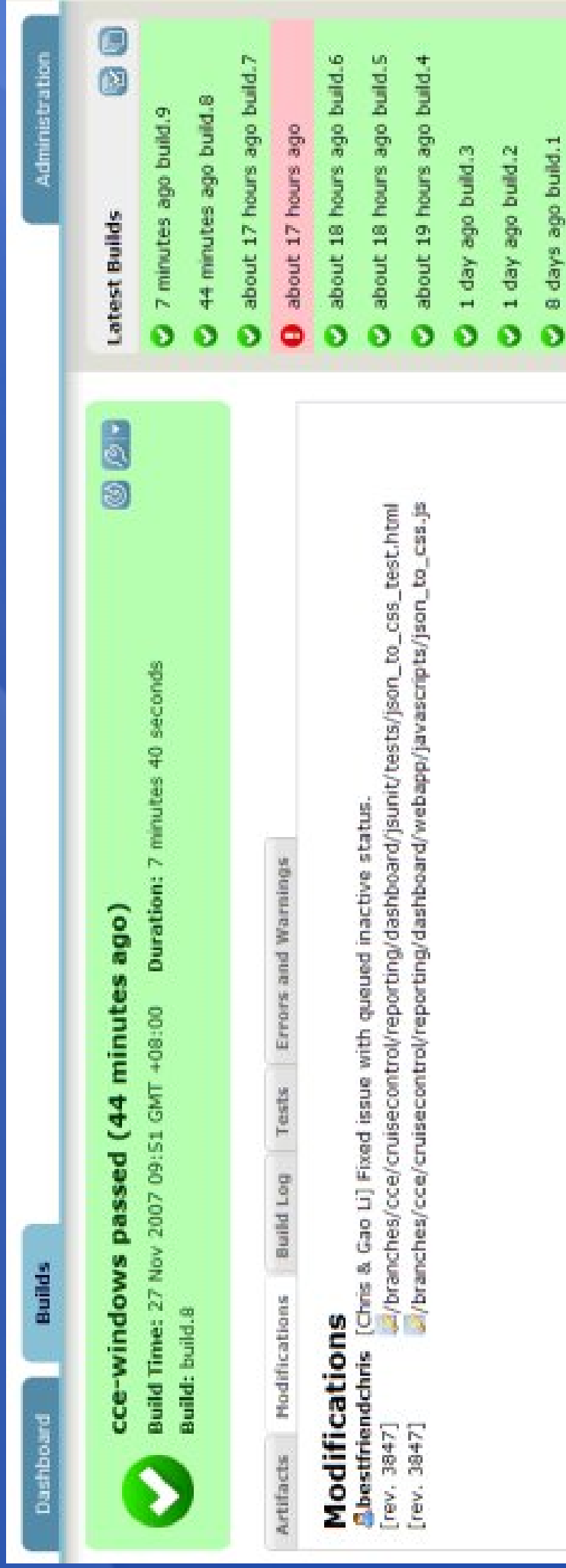
No.	Executor	Status
1	Idle	
2	Idle	
3	Building javanet-maven-repository-daemon #826	⊘
4	Building jaxb-ri #3181	⊘
5	Building glassfish #105	⊘
6	Idle	

Below the tables is a 'Job' table with columns for 'Job', 'Last Success', 'Last Failure', and 'Last Duration'. The table lists various jobs such as 'Common annotations', 'bsh', 'dtd-parser', 'fi', 'fi (weeklv)', 'glassfish', 'hudson', 'istack-commons', 'iapex', 'java-ws-xml\_community\_discussion\_updater', and 'java.net.acl\_processor'. Each job entry includes a status indicator (blue or red circle) and a refresh icon.

Job	Last Success	Last Failure	Last Duration
Common annotations	4 days (#16)	9 months (#3)	39 seconds
bsh	6 months (#11)	10 months (#2)	59 seconds
dtd-parser	6 months (#8)	N/A	1 minute
fi	28 days (#586)	1 month (#567)	7 minutes
fi (weeklv)	6 days (#53)	13 days (#52)	5 minutes
glassfish	4 hours (#104)	1 day (#88)	1 hour
hudson	4 minutes (#201)	N/A	1 minute
istack-commons	12 days (#19)	16 days (#5)	14 seconds
iapex	3 days (#55)	9 hours (#64)	1 minute
java-ws-xml_community_discussion_updater	4 minutes (#16146)	10 hours (#16125)	1 minute
java.net.acl_processor	18 hours (#162)	N/A	0 seconds

# Similar projects


## ➤ CruiseControl (JAVA)



The screenshot displays the CruiseControl (JAVA) dashboard. At the top, there are navigation tabs for "Dashboard", "Builds", and "Administration". The main content area features a large green box with a checkmark icon, indicating a successful build. Below this, the build details are shown: "cce-windows passed (44 minutes ago)", "Build Time: 27 Nov 2007 09:51 GMT +08:00", "Duration: 7 minutes 40 seconds", and "Build: build.8". To the right of this box are icons for "Builds" and "Administration". Below the build details, there are tabs for "Artifacts", "Modifications", "Build Log", "Tests", and "Errors and Warnings". The "Modifications" tab is active, showing a list of changes with commit hashes and file paths. At the bottom right, there is a "Latest Builds" section with a list of recent builds, each with a status icon (checkmark or warning) and a timestamp.




**Builds**

**Dashboard** **Builds** **Administration**











 **cce-windows passed (44 minutes ago)**  
Build Time: 27 Nov 2007 09:51 GMT +08:00    Duration: 7 minutes 40 seconds  
Build: build.8

**Artifacts** **Modifications** **Build Log** **Tests** **Errors and Warnings**

**Modifications**

-  bestfriendchris [Chris & Gao Li] Fixed issue with queued inactive status.  
[rev. 3847]  /branches/cce/cruisecontrol/reporting/dashboard/\_json\_to\_css\_test.html
- [rev. 3847]  /branches/cce/cruisecontrol/reporting/dashboard/webapp/javascripts/\_json\_to\_css.js

**Latest Builds**

-  7 minutes ago build.9
-  44 minutes ago build.8
-  about 17 hours ago build.7
-  about 17 hours ago
-  about 18 hours ago build.6
-  about 18 hours ago build.5
-  about 19 hours ago build.4
-  1 day ago build.3
-  1 day ago build.2
-  8 days ago build.1

# BuildBot Demo

# Practices of Continuous Integration

# What is CI

- A software development practice (requires no particular tooling)
- From Extreme Programming development process

# Wikipedia definition

In **software engineering**, continuous integration (CI) implements continuous processes of applying quality control on small pieces of effort, applied frequently. Continuous integration aims to improve the **quality of software**, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control **after** completing all development.



# Why use CI system

- Detect integration errors **ASAP**
  - Fix bugs made in last week: painful
  - Fix bugs made in last month: **very painful!!!**

# People thought . . .

- Before
  - It can't work (here).
  - Doing it won't make much difference.
- After
  - Yes we do that - how could you live without it?

# How to use CI system

1. Checkout source code from VCS
2. Modify code
3. Run self-testing code (ex. unit test)
4. Update and commit
5. Run self-testing code on an integration machine

# How to use CI system (cont.)

- Step 5. can be done automatically by CI system
- Errors is detected **rapidly**
- A good team should have many correct builds a day
- Bad builds do occur from time to time, but **should be quickly fixed**

# Continuous integration

- Enhance portability:
  - Different Build slaves for different HW/SW architectures
- Enhance/check numerical stability:
  - Different build slaves for different compiler options
- Enhance your own " ...ility"

# Practices of CI

- Maintain a single source repository
  - Put everything required for a build
- Automate the build
  - Build and launch in a single command
  - Need a virgin machine

# Practices of CI (cont.)

- Make your build self-testing
  - Testing can catch a lot of bugs – enough to be useful
- Everyone commits every day
  - Conflicts that stay undetected for weeks can be very hard to resolve

# Practices of CI (cont.)

- Every commit should build the mainline on an integration machine
  - Why: things still go wrong
    - Developers forget to run build before commit
    - Environmental differences
  - Manual build & CI system



# Practices of CI (cont.)

- Keep the build fast
  - The whole point of CI is to provide **rapid feedback**
  - **10 minutes** build

# Practices of CI (cont.)

- Keep the build fast (cont.)
  - Staged build (2<sup>nd</sup> build doesn't have the same 'stop everything' quality)

		<pre>./bin/util/collector_interface_testing.rb failed stdio</pre>
		<pre>'/etc/init.d/collector restart' stdio</pre>
		<pre>'rake setup:overwrite_codefile' stdio</pre>
		<pre>'./bin/util/run_collector_test.sh' stdio</pre>
		<pre>update r2466 stdio</pre>
		<b>Build 34</b>
15:32:07		
15:30:38		

# Practices of CI (cont.)

- Keep the build fast (cont.)
  - 2<sup>nd</sup> build fails: add another test into commit test suite
    - MySQL story
- Test in a clone of the production environment
  - Real tests without mocks

# Practices of CI (cont.)

- Make it easy for anyone to get the latest executable
  - Nightly build
- Everyone can see what's happening
  - BuildBot demo (countdown)
- Automate deployment
  - Also needs automated rollback

# Benefits of CI

- Reduce risk
  - Integration is a long and unpredictable process
- There's no long integration anymore
- Easier to find bugs and remove them
  - Quality of your test suits

# Where to start

1. Get the build automated
  - An automated nightly build is a fine step
2. Introduce some automated testing into your build
  - Try to identify the major areas

# Where to start

(cont.)

3. Try to speed up the `commit build`

- Magic **10** mins

# Buildbot for RegCM (1)

- Continuous integration:
  - Test all the compilers/libraries available on the ICTP HPC facility
  - Test all the option available:
    - ./configure --enable-clm
    - ./configure --enable-band
    - ./configure --enable-debug
  - Run some regression tests on any change...



# Buildbot for RegCM(2)

- Nightly bots to run a set of basic examples
  - RegCM\_root/Testing
- Aim:
  - Check if last modification broke numerical results
- Other things to check/do
  - Enable bots on branches
  - Enable bots on different HW

# BuildBot

And Continuous Integration  
RegCM4 experiences

S.Cozzini/A.Messina/G.Giuliani

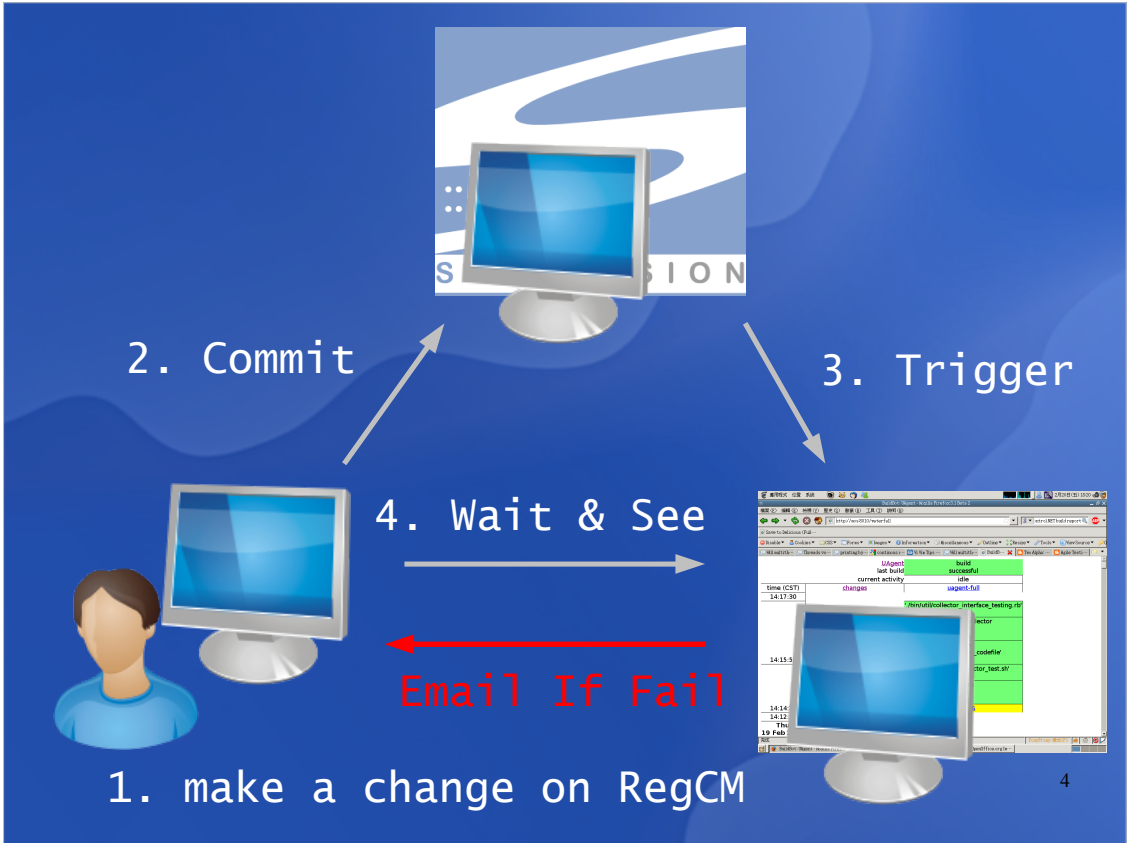
Warning: Some slides/ideas  
stolen by Willie  
<willie@issdu.com.tw>

1

# Agenda

- How do we use BuildBot here?
- What is BuildBot?
- Practices of Continuous Integration
- Example for RegCM code

# How do we use BuildBot?



Subject: buildbot failure in ICTP BuildBot on  
regcm-conf=debug|mod=openmpi:1.4.3:intel:2011  
Date: Tue, 21 Feb 2012 22:20:39 +0100  
From: buildmaster@gforge.ictp.it  
To: amessina@ictp.it, ggiulian@ictp.it  
The Buildbot has detected a failed build on builder  
regcm-conf=debug|mod=openmpi:1.4.3:intel:2011 while building ICTP  
BuildBot.  
Full details are available at:

<http://buildbot.ictp.it/builders/regcm-conf%3Ddebug%7Cmod%3Dopenmpi%3A1.4.3%3Aintel%3A2011>  
Buildbot URL: <http://buildbot.ictp.it/>  
Buildslave for this Build: argo-buildbot01  
Build Reason: The web-page 'force build' button was pressed by ''  
Build Source Stamp: 2750  
Blamelist:  
BUILD FAILED: failed compile  
sincerely,  
-The Buildbot

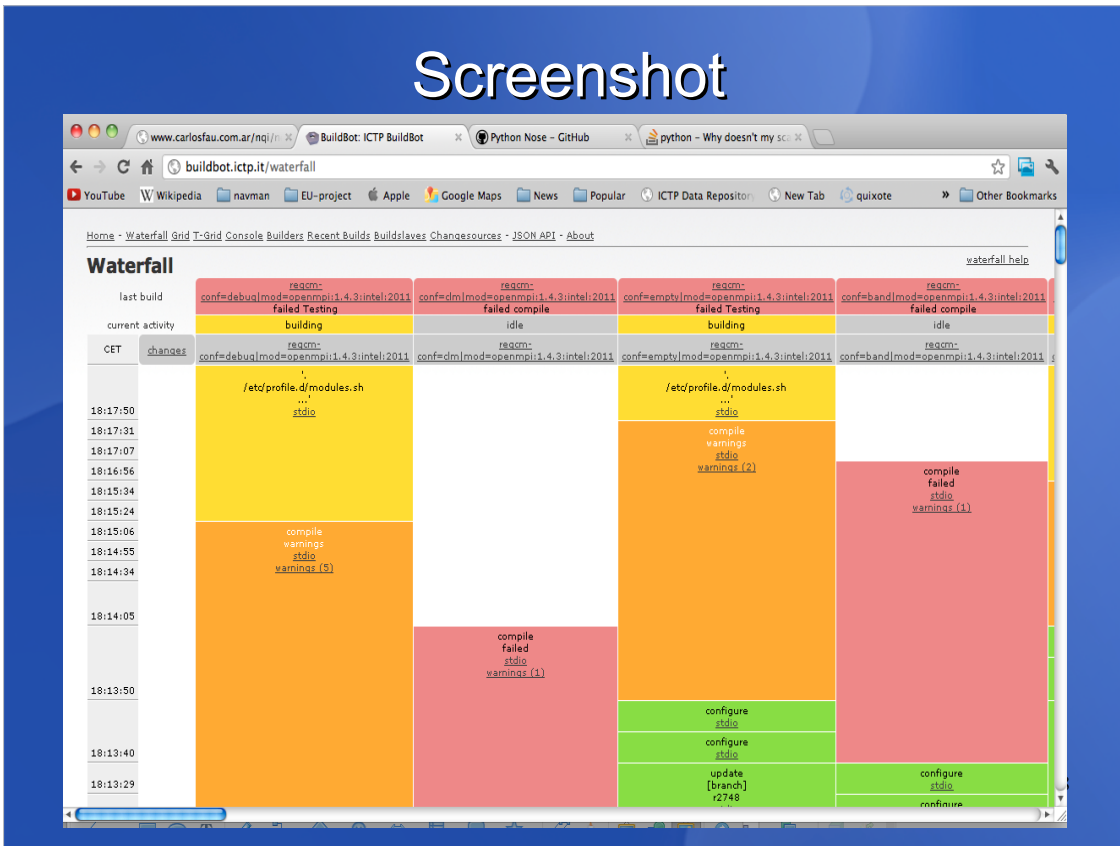
# What is BuildBot?

# BuildBot

- › A continuous integration system
- › Help to automate the compile/test cycle
- › Written in Python



# Screenshot



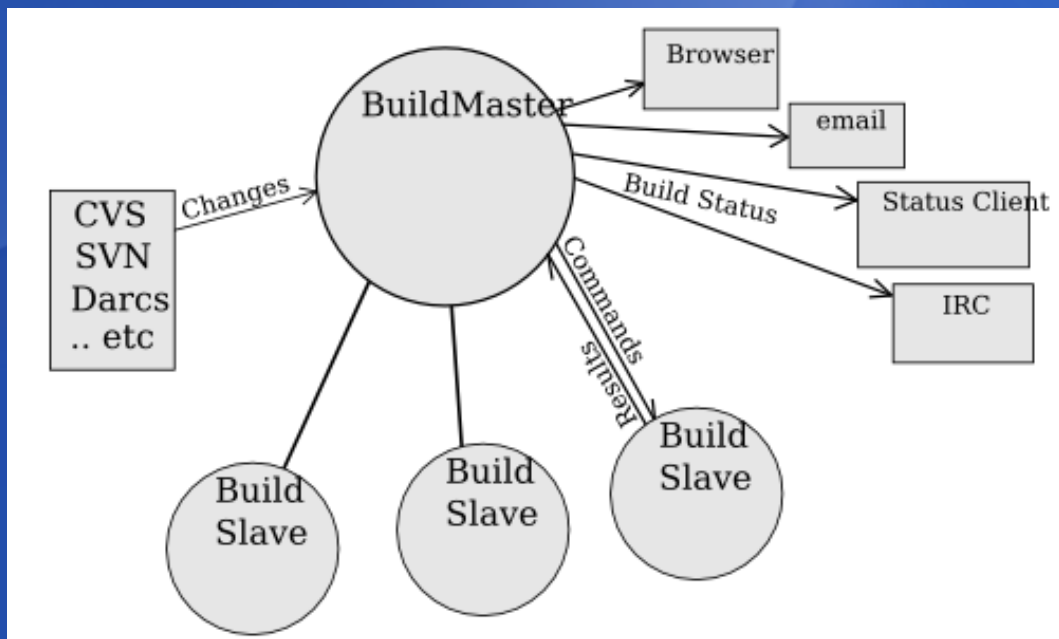
# Screenshot

CLI

Grid View

ICTP BuildBot	2750	2749	2752 in branches/regcm-core	2753 in branches/regcm-core	2754 in branches/regcm-core
regcm-conf=REGRESSION band mod=openmpi:1.4.3 intel:2011 (waiting)					
regcm-conf=REGRESSION band mod=openmpi:1.5.1 intel:2011 (waiting)					
regcm-conf=REGRESSION band mod=openmpi:1.5.3 ppc64le:10.9 (waiting)					
regcm-conf=REGRESSION clm mod=openmpi:1.4.3 intel:2011 (waiting)					
regcm-conf=REGRESSION clm mod=openmpi:1.5.1 intel:2011 (waiting)					
regcm-conf=REGRESSION clm mod=openmpi:1.5.3 ppc64le:10.9 (waiting)					
regcm-conf=REGRESSION debug mod=openmpi:1.4.3 intel:2011 (waiting)					
regcm-conf=REGRESSION debug mod=openmpi:1.5.1 intel:2011 (waiting)					
regcm-conf=REGRESSION debug mod=openmpi:1.5.3 ppc64le:10.9 (waiting)					
regcm-conf=REGRESSION empty mod=openmpi:1.4.3 intel:2011 (waiting)					
regcm-conf=REGRESSION empty mod=openmpi:1.5.1 intel:2011 (waiting)					
regcm-conf=REGRESSION empty mod=openmpi:1.5.3 ppc64le:10.9 (waiting)					
regcm-conf=band mod=openmpi:1.4.3 intel:2011 (building)			failed compile	failed compile	building
regcm-conf=band mod=openmpi:1.5.1 intel:2011 (building)			failed compile	failed compile	building
regcm-conf=band mod=openmpi:1.5.3 ppc64le:10.9 (building)			failed compile	failed compile	building
regcm-conf=clm mod=openmpi:1.4.3 intel:2011 (building)			failed compile	failed compile	building
regcm-conf=clm mod=openmpi:1.5.1 intel:2011 (building)			failed compile	failed compile	building
regcm-conf=clm mod=openmpi:1.5.3 ppc64le:10.9 (building)			failed compile	failed compile	building
regcm-conf=debug mod=openmpi:1.4.3 intel:2011 (building)	failed compile	failed compile	failed compile	failed compile	building
regcm-conf=debug mod=openmpi:1.4.3 intel:2011 (building)	failed compile	OK	OK	OK	building
regcm-conf=debug mod=openmpi:1.5.1 intel:2011 (building)	failed compile	OK	OK	OK	building
regcm-conf=debug mod=openmpi:1.5.3 ppc64le:10.9 (building)	OK	failed compile	OK	OK	building

# System architecture



# Similar projects

## ➤ Hudson (JAVA)

The screenshot shows the Hudson web interface in a Microsoft Internet Explorer browser window. The address bar shows the URL <http://kohsuke.sfbay.com/hudson/>. The page title is "Hudson".

On the left side, there are navigation links: "New Job", "Configure", and "Reload Config". Below these is the "Build Queue" section, which shows a list of jobs in the queue:

No.	Status
1	Idle
2	Idle
3	Building javanet-maven-repository-daemon #826
4	Building jaxb-ri #3181
5	Building glassfish #105
6	Idle

On the right side, there is a table of jobs with columns for "Job", "Last Success", "Last Failure", and "Last Duration". The jobs listed are:

Job	Last Success	Last Failure	Last Duration
Common annotations	4 days (#15)	9 months (#3)	39 seconds
bsh	6 months (#11)	10 months (#2)	59 seconds
dtd-parser	6 months (#8)	N/A	1 minute
fi	28 days (#586)	1 month (#567)	7 minutes
fi (weeklv)	6 days (#53)	13 days (#52)	5 minutes
glassfish	4 hours (#104)	1 day (#88)	1 hour
hudson	4 minutes (#201)	N/A	1 minute
istack-commons	12 days (#19)	16 days (#5)	14 seconds
jaxp	3 days (#55)	9 hours (#54)	1 minute
java-ws-xml-community-discussion-updater	4 minutes (#16146)	10 hours (#16125)	1 minute
java.net.acl.processor	18 hours (#152)	N/A	0 seconds

# Similar projects

› CruiseControl (JAVA)

Dashboard

**Builds**



**cce-windows passed (44 minutes ago)**

**Build Time:** 27 Nov 2007 09:51 GMT +08:00    **Duration:** 7 minutes 40 seconds

**Build:** build.8



Artifacts

**Modifications**

Build Log

Tests

Errors and Warnings

## Modifications

 **bestfriendchris** [Chris & Gao Li] Fixed issue with queued inactive status.

[rev. 3847]  /branches/cce/cruisecontrol/reporting/dashboard/jsunit/tests/json\_to\_css\_test.html

[rev. 3847]  /branches/cce/cruisecontrol/reporting/dashboard/webapp/javascripts/json\_to\_css.js

# BuildBot Demo

# Practices of Continuous Integration

# What is CI

- › A software development practice (requires no particular tooling)
- › From Extreme Programming development process



# Wikipedia definition

In **software engineering**, continuous integration (CI) implements continuous processes of applying quality control on small pieces of effort, applied frequently. Continuous integration aims to improve the **quality of software**, and to reduce the time taken to deliver it, by replacing the traditional practice of applying quality control **after** completing all development.

# Why use CI system

- › Detect integration errors  
**ASAP**
  - Fix bugs made in last week:  
painful
  - Fix bugs made in last  
month: **very painful!!!**

# People thought...

## › Before

- It can't work (here).
- Doing it won't make much difference.

## › After

- Yes we do that – how could you live without it?

# How to use CI system

1. Checkout source code from VCS
2. Modify code
3. Run self-testing code (ex. unit test)
4. Update and commit
5. **Run self-testing code on an integration machine**

## How to use CI system (cont.)

- › Step 5. can be done automatically by CI system
- › Errors is detected **rapidly**
- › A good team should have many correct builds a day
- › Bad builds do occur from time to time, but **should be quickly fixed**

# Continuous integration

- Enhance portability:
  - Different Build slaves for different HW/SW architectures
- Enhance/check numerical stability:
  - Different build slaves for different compiler options
- Enhance your own "...ility"

# Practices of CI

- › Maintain a single source repository
  - Put everything required for a build
- › Automate the build
  - Build and launch in a single command
  - Need a virgin machine

# Practices of CI (cont.)

- › Make your build self-testing
  - Testing can catch a lot of bugs – enough to be useful
- › Everyone commits every day
  - Conflicts that stay undetected for weeks can be very hard to resolve



# Practices of CI (cont.)

- › Every commit should build the mainline on an integration machine
  - Why: things still go wrong
    - Developers forget to run build before commit
    - Environmental differences
  - Manual build & CI system

# Practices of CI (cont.)

- › Keep the build fast
  - The whole point of CI is to provide **rapid feedback**
  - **10 minutes** build

# Practices of CI (cont.)

- › Keep the build fast (cont.)
  - Staged build (2<sup>nd</sup> build doesn't have the same 'stop everthing' quality)

15:32:07		./bin/util/collector_interface_testing.rb failed stdio
		./etc/init.d/collector restart' stdio
		'rake setup:overwrite_codefile' stdio
		./bin/util/run_collector_test.sh' stdio
		update r2466 stdio
15:30:38		Build 34

# Practices of CI (cont.)

- › Keep the build fast (cont.)
  - 2<sup>nd</sup> build fails: add another test into commit test suit
    - MySQL story
- › Test in a clone of the production environment
  - Real tests without mocks

# Practices of CI (cont.)

- › Make it easy for anyone to get the latest executable
  - Nightly build
- › Everyone can see what's happening
  - BuildBot demo (countdown)
- › Automate deployment
  - Also needs automated rollback<sup>28</sup>

# Benefits of CI

- › Reduce risk
  - Integration is a long and unpredictable process
- › There's no long integration anymore
- › Easier to find bugs and remove them
  - Quality of your test suits

# Where to start

1. Get the build automated
  - An automated nightly build is a fine step
2. Introduce some automated testing into your build
  - Try to identify the major areas

# Where to start (cont.)

3. Try to speed up the commit  
build

- Magic 10 mins



# Buildbot for RegCM (1)

- Continuous integration:
  - Test all the compilers/libraries available on the ICTP HPC facility
  - Test all the option available:
    - `./configure --enable-clm`
    - `./configure --enable-band`
    - `./configure --enable-debug`
  - Run some regression tests on any change...

## Buildbot for RegCM(2)

- Nightly bots to run a set of basic examples
  - RegCM\_root/Testing
- Aim:
  - Check if last modification broke numerical results
- Other things to check/do
  - Enable bots on branches
  - Enable bots on different HW