HYDRA: TOP-NOTCH CONTINUOUS INTEGRATION FOR DEMANDING PEOPLE

Ludovic Courtès
Service d’expérimentation & développement
Inria Bordeaux Sud-Ouest

3 November 2011
scenario 1: a peaceful development team...
scenario 1: ... software that works ...
scenario 1: ... and suddenly ...
scenario 1: who broke it?!
scenario 2: “works for me”

▶ user: It’s broken.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me.
scenario 2: “works for me”

- user: It’s broken.
- dev: Weird, it works for me. Which platform?
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64.
scenario 2: “works for me”

▶ **user**: It’s broken.
▶ **dev**: Weird, it works for me. Which platform?
▶ **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2.
scenario 2: “works for me”

► **user**: It’s broken.

► **dev**: Weird, it works for me. Which platform?

► **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec --disable-color.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec --disable-color.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec --disable-color.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, GDBM 1.8.2, GTK+ 2.12. Oh, and with --enable-color --enable-gui.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec --disable-color.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, GDBM 1.8.2, GTK+ 2.12. Oh, and with --enable-color --enable-gui.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec --disable-color.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, GDBM 1.8.2, GTK+ 2.12. Oh, and with --enable-color --enable-gui.
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, **BDB 4.2**. Et avec `--disable-color`.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, **GDBM 1.8.2**, GTK+ 2.12. Oh, and with `--enable-color --enable-gui`. 
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec `--disable-color`.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, GDBM 1.8.2, GTK+ 2.12. Oh, and with `--enable-color --enable-gui`. 
scenario 2: “works for me”

- **user**: It’s broken.
- **dev**: Weird, it works for me. Which platform?
- **user**: Darwin 10.4. On x86_64. With Apple GCC 4.0.1, GMP 4.2.4, MPFR 2.4.2, BDB 4.2. Et avec `--disable-color`.
- **dev**: Well, too bad. I rather test on GNU/Linux, i686, with GCC 4.6.2, GMP 4.2.1, MPFR 2.4.1, GDBM 1.8.2, GTK+ 2.12. Oh, and with `--enable-color --enable-gui`. 
why continuous integration is so great

1. continuous compilation/test
why continuous integration is so great

1. continuous compilation/test
2. quick feedback
why continuous integration is so great

1. continuous compilation/test
2. quick feedback
3. pristine build environment
why continuous integration is so great

1. continuous compilation/test
2. quick feedback
3. pristine build environment
4. different configurations under test
why continuous integration is so great

1. continuous compilation/test
2. quick feedback
3. pristine build environment
4. different configurations under test
5. build results published
the true cost of doing it yourself

```bash
while true
do
  rm -rf foo
  svn co svn+ssh://scm.gforge.inria.fr/svn/foo/trunk foo
  cd foo
  make
  sleep 300
done
```
while true
do
  rm -rf foo
  svn co svn+ssh://scm.gforge.inria.fr/svn/foo/trunk foo
  cd foo
  make 2>&1 > build-log || \\n    mail michel.cosnard@inria.fr -s "Build failed!" \\n    < build-log
  sleep 300
done
the true cost of doing it yourself

```bash
function svn_revision {
    ( cd "$1"
        svn info  | \n            grep '\^R.vision *: *\([0-9]\+\)$' | \n        sed -e's/[^0-9]//g' )
}
function svn_update {
    ( prev_rev="$(svn_revision $1)"
        ( cd "$1" ; svn update )
        new_rev="$(svn_revision $1)"
        test $prev_rev -ne $new_rev )
}
function update_html_report {
    local stamp="$(date -u +%Y%m%d-%H%M)"
    local test_log="$builddir/testsuite/log"
    local failure_rate="(tests not run)"
    gzip --best < "$2" > "build-$stamp.log.gz"
    scp "build-$stamp.log.gz" "$ssh_remote_dir/"
    if [ -f "$test_log" ]
        then
            failure_rate="$(test_failure_rate $test_log)"
            html
                gzip --best
                scp "test-$stamp.log.gz" "$ssh_remote_dir/"
            fi
    local result bgcolor
    if [ $1 -ne 0 ]
        then
            result="<b>FAIL</b>"
        else
            result="PASS"
        fi
    local rev="$(svn_revision $srcdir)"
    local html="<tr bgcolor="$bgcolor">
            <td><a href="$(gforge_revision_url foo $rev)
                "$rev</a></td>
            <td>$stamp</td>
            <td>$failure_rate</td>
            <td>
                $result [ <a href="build-$stamp.log.gz">build</a> |
                <a href="test-$stamp.log.gz">test</a> ]
            </td>
        </tr>"
    ed "$html_report" <<EOF
        /INSERT BUILD RESULTS
        +1
        i
        failure_rate="$(test_failure_rate $test_log)"
        html
            gzip --best
            scp "test-$stamp.log.gz" "$ssh_remote_dir/"
            wq
            < "$builddir/testsuite/log"
        EOF
    scp "$html_report" "$ssh_remote_dir/index.html"
}
```

L. Courtès — Hydra: continuous integration for demanding people
the true cost of doing it yourself

```bash
function svn_revision {
    ( cd "$1"
        svn info | \ 
        grep \^R\vision *: *\([^0-9]\+\)$' | \ 
        sed -e's/[^0-9]//g' )
}

function svn_update {
    ( prev_rev="$(svn revision $1)"
        ( cd "$1" ; svn update )
        new_rev="$(svn revision $1)"
        test $prev_rev -ne $new
    )
}

function update_html_report {
    local stamp="$(date -u +%Y%m%d-%H%M)"
    local test_log="$builddir/testsuite/log"
    local failure_rate="(tests not run)"
    gzip --best < "$2" > "build-$stamp.log.gz"
    scp "build-$stamp.log.gz" "$ssh_remote_dir/"
    if [ -f "$test_log" ]
    then
        failure_rate="$(test failure_rate $test_log)"
        gzip --best
        scp "test-$stamp.log.gz" "$ssh_remote_dir/"
    fi
    local result bgcolor
    if [ $1 -ne 0 ]
    then
        result="<b>FAIL</b>"
        bgcolor="lightred"
    else
        result="PASS"
        bgcolor="lightgreen"
    fi
    local rev="$(svn revision $srcdir)"
    local html="<tr bgcolor="$bgcolor">
        <td><a href="$(gforge revision_url foo $rev)">$rev</a></td>
        <td>$stamp</td>
        <td>$failure_rate</td>
        <td>
            $result
            [ <a href="build-$stamp.log.gz">build</a> | 
              <a href="test-$stamp.log.gz">test</a> ]
        </td>
    </tr>"
    ed "$html_report" <<EOF
/INSERT BUILD RESULTS
+E+1
i$html.
wq
EOF
    scp "$html_report" "$ssh_remote_dir/index.html"
}
```

lots of tools!

- BuildBot
- Jenkins (formerly Hudson)
- CruiseControl
- ...
- Hydra
suitability of a continuous integration tool

- languages & build systems?
- build notification & visualization?
- portability?
- reproducibility?
- scalability?
“Thanks to Hydra, my software projects shine more than ever!”

— Ludovic C.
Hydra, Nix, & you
http://nixos.org/hydra/

- **Nix**
  - build tool
  - focus on **controlling the build environment**

- **Hydra**
  - continuous integration tool based on Nix
Hydra, Nix, & you

http://nixos.org/hydra/

- **Nix**
  - build tool
  - focus on *controlling the build environment*

- **Hydra**
  - *continuous integration tool based on Nix*
  - web interface + email notification
  - GNU/Linux, FreeBSD, MacOS X, Cygwin
  - distributed builds
Hydra, Nix, & you

http://nixos.org/hydra/

- **Nix**
  - build tool
  - focus on **controlling the build environment**

- **Hydra**
  - continuous integration tool based on Nix
  - web interface + email notification
  - GNU/Linux, FreeBSD, MacOS X, Cygwin
  - distributed builds
  - SSH/root access to the build host **not needed**
Hydra, Nix, & you

http://nixos.org/hydra/

- **Nix**
  - build tool
  - focus on **controlling the build environment**

- **Hydra**
  - **continuous integration tool based on Nix**
  - web interface + email notification
  - GNU/Linux, FreeBSD, MacOS X, Cygwin
  - distributed builds
  - SSH/root access to the build host **not needed**
  - **manage the build environment by yourself!**
a project

---

Project runtime

Jobsets

This project has the following jobsets:

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Last evaluated</th>
<th>Success</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>hwloc-trunk</td>
<td>2011-11-02 15:08:54</td>
<td>100%</td>
<td>Evaluate</td>
</tr>
<tr>
<td>✗</td>
<td>marcel-trunk</td>
<td>2011-11-02 15:08:55</td>
<td>97%</td>
<td>Evaluate</td>
</tr>
<tr>
<td>✗</td>
<td>open-mx-master</td>
<td>2011-11-02 15:08:58</td>
<td>69%</td>
<td>Evaluate</td>
</tr>
<tr>
<td>✗</td>
<td>starpu-trunk</td>
<td>2011-11-02 15:09:00</td>
<td>40%</td>
<td>Evaluate</td>
</tr>
<tr>
<td>✓</td>
<td>tbx-trunk</td>
<td>2011-11-02 15:09:02</td>
<td>100%</td>
<td>Evaluate</td>
</tr>
</tbody>
</table>

[Create a new jobset]
a job set
a build
build inputs

Job runtime:hwloc-trunk:build build 10735

Build inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Revision</th>
<th>Store path</th>
</tr>
</thead>
<tbody>
<tr>
<td>tarball</td>
<td>Build output</td>
<td>runtime:hwloc-trunk:tarball build 10717</td>
<td></td>
<td>/nix/store/41nc8cw2lbxx6x373sp0dh hwloc-tarball-Gpre3949</td>
</tr>
<tr>
<td>system</td>
<td>String value</td>
<td>x86_64-linux</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nixpkgs</td>
<td>Subversion export</td>
<td><a href="https://svn.nixos.org/repos/nixpkgs/trunk/">https://svn.nixos.org/repos/nixpkgs/trunk/</a></td>
<td>30123</td>
<td>/nix/store/fi09x9lxlnw0lihr9nja nixpkgs-r30123</td>
</tr>
<tr>
<td>hydraConfig</td>
<td>Git checkout</td>
<td>git://scm.gforge.inria.fr/hydra-recipes/hydra-recipes.git</td>
<td>b7ae2f1fd5a8e5dfa38cf59e9f4a8239fa6c93</td>
<td>/nix/store/bfyg3dlyh3kmjdJSkff03s git-export</td>
</tr>
</tbody>
</table>

Changes since previous build: runtime:hwloc-trunk:build build 10621

<table>
<thead>
<tr>
<th>Input</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>tarball</td>
<td>10735 to 10735</td>
</tr>
</tbody>
</table>
### Build inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Revision</th>
<th>Store path</th>
</tr>
</thead>
<tbody>
<tr>
<td>tarball</td>
<td>Build output</td>
<td><code>runtime:hwloc-trunk:tarball build 10717</code></td>
<td></td>
<td><code>/nix/store/4lnc8cw2lbxx6x373p0dh   hwloc-tarball-Qpre3949</code></td>
</tr>
<tr>
<td>system</td>
<td>String value</td>
<td>&quot;x86_64·linux&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nixpkgs</td>
<td>Subversion</td>
<td><a href="https://svn.nixos.org/repos/nix/nixpkgs/trunk/">https://svn.nixos.org/repos/nix/nixpkgs/trunk/</a></td>
<td>30123</td>
<td><code>/nix/store/fidig9lxlnxihr9mjcx   nixpkgs-r30123</code></td>
</tr>
<tr>
<td>hydraConfig</td>
<td>Git checkout</td>
<td>git://scm.gforge.inria.fr/hydra-recipes/hydra-recipes.git</td>
<td>b7ae2f1f1d5a8e5dfe38cf59e9f4a8239fa6c93</td>
<td><code>/nix/store/byl93dlyh3kmdj5kff03s   git-export</code></td>
</tr>
</tbody>
</table>

**Changes since previous build:** `runtime:hwloc-trunk:build build 10621`

<table>
<thead>
<tr>
<th>Input</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>tarball</td>
<td>10735 to 10735</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>nixpkgs</td>
<td>29979 to 30112</td>
</tr>
<tr>
<td>hydraConfig</td>
<td>59feld to b7ae2</td>
</tr>
<tr>
<td>nixpkgs</td>
<td>30077 to 30123</td>
</tr>
<tr>
<td>hydraConfig</td>
<td>59feld to b7ae2</td>
</tr>
</tbody>
</table>
view of a successful job set

**View** runtime: hwloc result **10717** (hwloc-1.4a1-1)

*Finished building on 2011-10-30 02:04:47.*

**Tarball and Manual**

- Manual
- [Source distribution hwloc-1.4a1-1.tar.bz2](details, contents)
- [Source distribution hwloc-1.4a1-1.tar.gz](details, contents)

**NixOS GNU/Linux Packages**

- [One-click install of Nix package hwloc-1.4a1-1](help, contents)
- [Nix closure of path /nix/store/a7mkvkgbo6szfl79lxzdhc1dmr4i3i-hwloc-1.4a1-1](help)

**Code Coverage Report**

- Code coverage analysis report
a failed build

Job runtime:hwloc-trunk:build build 9701

Build ID: 9701
Nix name: hwloc-1.4a1-1
Status: Build failed (with result)
System: x86_64-darwin
Duration: 33s finished at 2011-10-13 09:23:29
Logfile: Available (raw, tail)

Add to release: [Apply]

[Clone this build]

Build products

Failed build produced output. Click here to inspect the output. [help]

Changes

<table>
<thead>
<tr>
<th>Last successful build</th>
<th>First broken build</th>
<th>This build</th>
</tr>
</thead>
</table>

| [ log diff ] |
build input changes that led to a failure

**Build products**

*Failed build produced output. Click here to inspect the output. [help]*

**Changes**

<table>
<thead>
<tr>
<th>Last successful build</th>
<th>First broken build</th>
<th>This build</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nixpkgs</td>
<td>29703 to 29801</td>
</tr>
<tr>
<td>hwlocSrc</td>
<td>3924 to 3932</td>
</tr>
<tr>
<td>hydraConfig</td>
<td>985352 to 0a092a</td>
</tr>
<tr>
<td>nixpkgs</td>
<td>29750 to 29801</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Log</th>
</tr>
</thead>
</table>

config.status: creating doc/doxygen-config.cfg
config.status: creating utils/Makefile
config.status: creating hwloc.pc
config.status: creating tests/Makefile
config.status: creating tests/Linux/Makefile
config.status: creating tests/Linux/gather/Makefile
config.status: creating tests/xml/Makefile
config.status: creating tests/ports/Makefile
config.status: creating tests/rename/Makefile
build environments vs. reproducibility

- versions of the dependencies
- compiler
- compilation options, and those of dependencies
- miscellaneous (locale, timezone, etc.)
- paths
build environments vs. reproducibility

- versions of the dependencies
- compiler
- compilation options, and those of dependencies
- miscellaneous (locale, timezone, etc.)
- paths

- I/path/to(headers
  - L/path/to/lib
  - $CPATH
  - $LIBRARY_PATH

- $LD_LIBRARY_PATH
- $XML_CATALOG_FILES
- $GUILE_LOAD_PATH
- $CLASSPATH
build environments vs. reproducibility

- versions of the dependencies
- compiler
- compilation options, and those of dependencies
- miscellaneous (locale, timezone, etc.)
- paths

```
-I/path/to/headers $CPATH
-L/path/to/lib $LIBRARY_PATH
$LD_LIBRARY_PATH
RPATH RUNPATH
```
build environments vs. reproducibility

- versions of the dependencies
- compiler
- compilation options, and those of dependencies
- miscellaneous (locale, timezone, etc.)
- paths

-\texttt{-I/path/to/headers} \quad \$\texttt{CPATH}
-\texttt{-L/path/to/lib} \quad \$\texttt{LIBRARY\_PATH}
  \$\texttt{LD\_LIBRARY\_PATH}
  \texttt{RPATH}
  \texttt{RUNPATH}
  \$\texttt{PYTHONPATH}
  \$\texttt{CLASSPATH}
  \$\texttt{XML\_CATALOG\_FILES}
  \$\texttt{PERL5LIB}
  \$\texttt{GUILE\_LOAD\_PATH}
build environments vs. reproducibility

- versions of the dependencies
- compiler
- compilation options, and those of dependencies
- miscellaneous (locale, timezone, etc.)
- paths

-Ahem, reproducible builds?
build environment control without Nix

Hello

Qt4

Which compiler? Which libc?
build environment control without Nix

which compiler? which libc?
and now your favorite admin upgrades the system...
which compiler? which libc?

and now your favorite admin upgrades the system...

Segmentation fault
uncontrolled build environment = untraceable failures

which compiler? which libc?

and now your favorite admin upgrades the system...

Segmentation fault
build environment control with Nix

... down to the compiler’s compiler!

¹Except for the kernel, the hardware, the direction of the wind, etc.
build environment control with Nix

... down to the compiler’s compiler!

- build environment **entirely under control**\(^1\)
- all the build inputs are known to Nix

\(^1\)Except for the kernel, the hardware, the direction of the wind, etc.
hello, world!
a job set

---

**Jobset** runtime:hwloc-trunk

---

**Most recent builds**

<table>
<thead>
<tr>
<th>#</th>
<th>Job</th>
<th>Release Name</th>
<th>System</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>10744 runtime:hwloc-trunk:build_opensuse110_x86_64</td>
<td>hwloc-1.4a1-1</td>
<td>x86_64-linux</td>
<td>2011-10-30 05:10:37</td>
</tr>
<tr>
<td>✓</td>
<td>10738 runtime:hwloc-trunk:build_fedora13_x86_64</td>
<td>hwloc-1.4a1-1</td>
<td>x86_64-linux</td>
<td>2011-10-30 05:05:54</td>
</tr>
<tr>
<td>✓</td>
<td>10741 runtime:hwloc-trunk:build_fedora2_i386</td>
<td>hwloc-1.4a1-1</td>
<td>x86_64-linux</td>
<td>2011-10-30 05:04:07</td>
</tr>
<tr>
<td>✓</td>
<td>10737 runtime:hwloc-trunk:coverage</td>
<td>hwloc-1.4a1-1</td>
<td>x86_64-linux</td>
<td>2011-10-30 02:11:19</td>
</tr>
<tr>
<td>✓</td>
<td>10735 runtime:hwloc-trunk:build</td>
<td>hwloc-1.4a1-1</td>
<td>x86_64-linux</td>
<td>2011-10-30 02:10:23</td>
</tr>
</tbody>
</table>

**Status**

<table>
<thead>
<tr>
<th>Job</th>
<th>i686-linux</th>
<th>x86_64-darwin</th>
<th>x86_64-linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>build</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>build_fedora13_x86_64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>build_fedora2_i386</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>build_opensuse110_x86_64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>coverage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>debian50_i386</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tarball</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

---

L. Courtès — Hydra: continuous integration for demanding people
writing a Hydra job set in Nix

```nix
{ 
  tarball = ...;
  build = ...;
  coverage = ...;
  deb = ...;
}
```
writing a Hydra job set in Nix

```plaintext
{
    tarball <= ...;
    build <= ...;
    coverage <= ...;
    deb <= ...;
}
make dist
make && make check...
make CFLAGS=-cov coverage...
qemu... checkinstall...
```
writing Hydra jobs in Nix

tarball =

let pkgs = import nixpkgs { }; in
pkgs.releaseTools.sourceTarball {
  name = "hello";
  src = helloSrc;
};

L. Courtès — Hydra: continuous integration for demanding people
writing Hydra jobs in Nix

tarball =
{ nixpkgs , helloSrc }

let pkgs = import nixpkgs { } ; in
pkgs.releaseTools.sourceTarball {
  name = "hello" ;
  src = helloSrc ;
};

checkout of the distro

checkout of Hello
writing Hydra jobs in Nix

```nix
build =

{nixpkgs, tarball, system}:

let pkgs = import nixpkgs { inherit system; }; in
  pkgs.releaseTools.nixBuild
  { name = "hello";
    src = tarball;
    buildInputs = [ pkgs.qt4 ];
    configureFlags = "--with-colors";
    diskImage = vmTools.diskImages.debian50x86; }
```

result of tarball job

x86_64-linux, or x86_64-darwin, etc.
writing Hydra jobs in Nix

```
build =
{ nixpkgs, tarball, system }

let pkgs = import nixpkgs { inherit system; }; in
pkgs.releaseTools.nixBuild {
  name = "hello";
  src = tarball;
  buildInputs = [ pkgs.qt4 ];
};
```

explicit dependency
writing Hydra jobs in Nix

```nix
build =
    { nixpkgs, tarball, system }

let pkgs = import nixpkgs { inherit system; }; in
pkgs.releaseTools.nixBuild {
  name = "hello";
  src = tarball;
  buildInputs = [ pkgs.qt4 ];
  configureFlags = "--with-colors";
};
```
writing Hydra jobs in Nix

coverage =

{ nixpkgs, tarball, system }:

let pkgs = import nixpkgs { inherit system; }; in
pkgs.releaseTools.coverageAnalysis {
  name = "hello";
  src = tarball;
  buildInputs = [ pkgs.qt4 ];
  configureFlags = "--with-colors";
};
deb = 
  { nixpkgs, tarball, system }:

let pkgs = import nixpkgs { inherit system; }; in pkgs.releaseTools.debBuild {
  name = "hello";
  src = tarball;

  configureFlags = "--with-colors";
  diskImage = vmTools.diskImages.debian50x86_64;
};
Jobset runtime:hwloc-trunk

Information

[Edit|Clone]

Description: hwloc, 'trunk'
Nix expression: runtime/hwloc/release.nix in INPUT hydraConfig
Enabled: Yes
Enable email notification: Yes
Email override: 
Number of builds to keep: 3
Last checked: 2011-11-02 14:00:15, no errors

Inputs

<table>
<thead>
<tr>
<th>Input name</th>
<th>Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>hwlocSrc</td>
<td>Subversion export</td>
<td><a href="http://svn.open-mpi.org/svn/hwloc/trunk">http://svn.open-mpi.org/svn/hwloc/trunk</a></td>
</tr>
<tr>
<td>hydraConfig</td>
<td>Git checkout</td>
<td>git://scm.gforge.inria.fr/hydra-recipes/hydra-recipes.git</td>
</tr>
<tr>
<td>nixpkgs</td>
<td>Subversion export</td>
<td><a href="https://svn.nixos.org/repos/nix/nixpkgs/trunk/">https://svn.nixos.org/repos/nix/nixpkgs/trunk/</a></td>
</tr>
<tr>
<td>system</td>
<td>String value</td>
<td>&quot;x86_64-linux&quot;, 'i686-linux', 'i86_64-darwin'</td>
</tr>
<tr>
<td>tarball</td>
<td>Build output</td>
<td>tarball</td>
</tr>
</tbody>
</table>
“But Autotools suck, CMake rocks.”

— Anonymous (and cavalier) attendee.
building with CMake

```python
build =
{ system, nixpkgs, helloSrc }:

let pkgs = import nixpkgs { inherit system; }; in
pkgs.releaseTools.nixBuild {
  name = "hello-cmake";
  src = helloSrc;
  cmakeFlags = "-DFOO_BAR=on";
  buildInputs = [ pkgs.cmake pkgs.qt4 ];
};
```
“All these C-ish things suck anyway; Java is the way.”

— Anonymous attendee.
building with Ant

build =
{ nixpkgs, helloSrc }:

let pkgs = import nixpkgs { }; in
pkgs.releaseTools.antBuild {
    name = "hello-ant";
    src = helloSrc;
    antTargets = [ "jar" "test" ];

    buildInputs = with pkgs; [ ant jdk ];
};
building with Maven

```haskell
build =
  { nixpkgs, helloSrc }:

  let pkgs = import nixpkgs { }; in
  pkgs.releaseTools.mvnBuild {
    name = "hello-maven" ;
    src = helloSrc;

    doJavadoc = true;
    doCheckstyle = true;
    buildInputs = with pkgs; [ maven2 jdk ];
  };
```
building things differently

▶ **customize** existing build functions (**nixBuild**, etc.)

▶ **write** your own build function
building things differently

- **customize** existing build functions (*nixBuild*, etc.)
  - define `configureFlags`, `makeFlags`, etc.
  - define `configurePhase`, `buildPhase`, etc. (shell snippets)
  - ...

- **write** your own build function
  - `pythonBuild` anyone?
introduction
features
practice
closing
Continuous integration pacifies developers!
Continuous integration pacifies developers!

- quick feedback to developers
- source + binaries + doc snapshots for users
- tests of various configurations
summary

Continuous integration pacifies developers!

- quick feedback to developers
- source + binaries + doc snapshots for users
- tests of various configurations
- available today!
Hydra @ Inria: what’s missing

- production-ready!
Hydra © Inria: what’s missing

- not quite production-ready
Hydra @ Inria: what’s missing

- not quite production-ready
- no private/proprietary projects yet
Hydra @ Inria: what’s missing

- not quite production-ready
- no private/proprietary projects yet
- no GPUs, no webcams, no USB percolator (yet)
Hydra @ Inria: what’s missing

- not quite production-ready
- no private/proprietary projects yet
- no GPUs, no webcams, no USB percolator (yet)
- some of your dependencies might be missing
Hydra @ Inria: what’s missing

▶ not quite production-ready
▶ no private/proprietary projects yet
▶ no GPUs, no webcams, no USB percolator (yet)
▶ some of your dependencies might be missing
▶ no blame list, no favorite Jenkins plug-in here
▶ ...

L. Courtès — Hydra: continuous integration for demanding people
Hydra © Inria: what’s already there

- small & growing compile farm: 12 x86_64 cores
Hydra © Inria: what’s already there

- small & growing compile farm: 12 x86_64 cores
- GNU/Linux, MacOS X, and FreeBSD
Hydra © Inria: what’s already there

- small & growing compile farm: 12 x86_64 cores
- GNU/Linux, MacOS X, and FreeBSD
- kernel & distributed tests in virtual machines
- virtual build environments: Debian, Fedora, openSuSE, etc.
Hydra © Inria: what’s already there

- small & growing compile farm: 12 x86_64 cores
- GNU/Linux, MacOS X, and FreeBSD
- kernel & distributed tests in virtual machines
- virtual build environments: Debian, Fedora, openSuSE, etc.
- 6.5 projects being built (C/C++/Fortran, Autotools/CMake)
Hydra @ Inria: what’s already there

- small & growing compile farm: 12 x86_64 cores
- GNU/Linux, MacOS X, and FreeBSD
- kernel & distributed tests in virtual machines
- virtual build environments: Debian, Fedora, openSuSE, etc.
- 6.5 projects being built (C/C++/Fortran, Autotools/CMake)
- ...
getting started

1. email sed-bordeaux@inria.fr
getting started

1. email sed-bordeaux@inria.fr
2. add your recipes to
   http://gforge.inria.fr/projects/hydra-recipes
getting started

1. email sed-bordeaux@inria.fr
2. add your recipes to http://gforge.inria.fr/projects/hydra-recipes
3. watch your project at http://hydra.bordeaux.inria.fr/
getting help

- copy/paste from http://gforge.inria.fr/projects/hydra-recipes
- read the fine manual at http://nixos.org/hydra/
- email sed-bordeaux@inria.fr