Outline

- Introduction
- **Specifying and configuring tests**
- Running tests
- **Obtain results**
- Upcoming new features
- **Getting help**
Introduction

- Continuous Integration (CI)
- why Continuous Integration
- definitions
- CI system overview
Introduction: **Continuous Integration (CI)**

- Continuous integration is a software engineering practice in which changes in a software code are immediately tested and reported.

- The goal is to provide rapid feedback helping identifying defects introduced by code changes as soon as possible.

- Issues detected early on in development are typically smaller, less complex and easier to resolve.
Introduction: why Continuous Integration

- Bad habits in code development can break your code...
  ...or someone else's code!
Introduction: why Continuous Integration

- Sometime also good practice in code development can lead to some hidden bug...
Introduction: why Continuous Integration

- The more code you write without testing, the more paths you have to check for errors.
  - Keep on a straight path with proper code testing.
Introduction: definitions

- **CI phase**
  - Is a single activity executed by the CI system
  - Examples are: checkout, build, install, ...

- **CI workflow**
  - Is a collection of CI phases
  - There are CI workflows used to test experiment code standalone, to update reference files used by CI tests, and so on

- **CI build**
  - Is the job that executes a specific CI workflow on a build node
Introduction: definitions

• Unit test
  • Is a test that verifies a single “unit” or logical concept of the system
  • Is fully automated (you want be able to run the test in an automatic procedure)
  • Is independent (you want be able to run the test in parallel)
  • Runs fast (you want quick feedback)
  • Is trustworthy (if the test fails you know that the code is broken)
  • It is good practice to have at least one unit test for each functionality of the system
Introduction: definitions

• **CI test or integration test**
  • Is the logical extension of unit test
  • Tests the behavior of “combined units” of the system
  • It verifies that the (major) parts of the system work well together
  • You can have:
    • **Regression test:** runs existing CI tests against modified code to make sure that what used to work doesn't break
    • **Reproducibility test:** makes sure that running the code using the same input always generate the same output
    • **Back-compatibility test:** make sure that new code is able to access data files produced by previous code releases
    • **Validation test:** make sure that new code produce meaningful results

• **Test suite**
  • Is a collection (suite) of CI tests
Introduction: **CI system overview**

**Fermilab Central Build Service**
- CI scripts run on selected Jenkins slaves
- CI scripts execute CI phases defined by CI workflow
- CI build triggered:
  - on demand by user
  - by commit into repository
  - by a cronjob

**CI web server**
- Server 1:
  - CI web app exp1
  - CI web app exp2
  - CI web app exp3
- Server 2:
  - CI web app exp1
  - CI web app exp2
  - CI web app exp3
- Redirector
- CI web app send queries to the CI DB through the CI web server. The info are shown by the CI web app

**DataBase Service**
- CI DB exp1
- CI DB exp2
- CI DB exp3
Specifying and configuring tests

- CI workflow configuration
- Integration test configuration
- CI validation and grid support
The CI workflow configuration drives all the actions to be performed by the CI build.

- The CI workflow configuration has three basic sections:
  - **default**: global section, selects the CI workflow to use, set the report mailing list, proxy VO, ...
  - **workflow**: defines *personality*, list of CI tests to run, list of CI phases, code modules to test, CI validation config file (grid_cfg), ...
  - **personality**: defines what each CI phase does

---


---

Excerpt of the CI workflow configuration

```
[default]
workflow             = ${LAR_WORKFLOW:-defaultwf}
notify_email_to   = larsoft_build@fnal.gov
proxy_vo             = /fermilab/uboone
build_db_uri        = http://dbweb6.fnal.gov:8080/LarCI/app

[defaultwf] ### workflow section
experiment    = LArSoft uBooNE DUNE LArIAT ArgoNeuT
qualifier         = ${LAR_QUALS:-e14:prof}
ci_test_lists    = quick_test_uboonecode quick_test_dunetpc
quick_test_lariatsoft quick_test_argoneutcode
personality    = mrb
proxy_flag     = true
phases          = _eval_n checkout_x_module
build unit_test install ci_tests
grid_cfg         = ${LAR_GRIDWFCFG:-cfg/grid_workflow.cfg}
modules        = larsoft ... uboonecode dunetpc lariatsoft argoneutcode

[mrb] ### personality section
# define what the CI phases do:
# _eval_n: setup the code environment
... #checkout: instruction to checkout the code
... #build: instruction to build the code
... # unit_test: instruction to run unit tests
... # install: instruction to install the code
... #ci_tests: instruction to run the CI tests
```
Integration test configuration

- The CI test configuration file is a INI-formatted file
  - It is parsed using the `ConfigParser` python module
- It is located in the experiment code repository at: `<exp code>/test/ci/ci_tests.cfg`
- It has three type of sections
  - `[DEFAULT]` is used to define global variables that can be used in the test section
  - `[test <testname>]` is used to configure the test `<testname>`
  - `[suite <suitename>]` is used to specify a collection of tests to run in the same job
Integration test configuration


- Basic layout for *ci_tests.cfg* file

  # Definition for a named 'testA' that uses the output from 'testB'
  [test testA]
  script=$<PRODUCT>_DIR/test/testA.sh
  args= -a qualA -b qualB
  requires= testB

  # Definition for 'testB'
  [test testB]
  script=$<PRODUCT>_DIR/test/testB.sh
  ...

  # Definition for 'testC'
  [test testC]
  script = $PRODUCT_DIR/test/testC.sh
  ...

  # Definition of test suite 'test_suiteA'
  [suite test_suiteA]
  testlist = testA testC
Integration test configuration


- Basic layout for ci_tests.cfg file

  # Definition for a named 'testA' that uses the output from 'testB'
  [test testA]
  script=$<PRODUCT>_DIR/test/testA.sh
  args= -a qualA -b qualB
  requires= testB

  # Definition for 'testB'
  [test testB]
  script=$<PRODUCT>_DIR/test/testB.sh
  ...

  # Definition for 'testC'
  [test testC]
  script = $PRODUCT_DIR/test/testC.sh
  ...

  # Definition of test suite 'test_suiteA'
  [suite test_suiteA]
  testlist = testA testC

Set dependencies between tests
Pre-requisites will run first

“testB” will run as part of the suite because it is declared as dependency of “testA”
CI validation and grid support


- Validation tests usually require thousands of events
  - For this purpose the grid can help to get the job done
- The CI allows to build a specific version of the code (tag, branch, …) and to use it to run jobs on the grid
- Data produced by the CI validation are stored in a configurable dCache area for further analysis
  - Also the code tarball and job logs are stored in dCache
- Provides stats about job usage resources
- Send an email report when the CI validation is complete and results are available
- Ability to track jobs using POMS
  https://cdcvs.fnal.gov/redmine/projects/prod_mgmt_db/wiki
  (Production Operations Management Service) a service which will assist the Production Teams and the Analysis groups of the experiments in their scientific computational work
CI validation and grid support

The CI validation phase has its own configuration file to set up.

- It lives in `lar_ci/cfg/grid_workflow.cfg`
- It consists of two types of sections:
  - **[global]** section that defines the experiment workflow
  - **[<stage>]** section that specifies stage properties

---

```ini
[global]
  stages_phase_1 = gen g4 detsim reco1 reco2 ana
  njobs_phase_1 = 5
  nevents_per_job_phase_1 = 2
  stages_phase_2 = merge
  njobs_phase_2 = 1
  validation_process = uBooNE calorimeter validation
  validation_function = calorimeter_validation
  ci_dcachedir = /pnfs/uboone/scratch/users/vito/CI_tests_grid/
  max_log_size = 20971520
  notify_grid_email_to = vito@fnal.gov
  POMS_CAMPAIGN_ID = 55

[gen]
  FHICL = prod_muminus_0.1-2.0GeV_isotropic_uboone.fcl
  expected_lifetime = 50m
  memory = 2000MB
  disk = 20GB
  executable = lar
  arguments = --rethrow-all
  output_filename = prodgenie_brb_nu_cosmic_uboone_gen.root
  output_to_transfer = prodgenie_brb_nu_cosmic_uboone_gen.root

[g4]
  FHICL = standard_g4_uboone.fcl
  expected_lifetime = 50m
  memory = 2000MB
  disk = 20GB
  executable = lar
  arguments = --rethrow-all
  input_from_stage = gen
  input_filename = prodgenie_brb_nu_cosmic_uboone_gen.root
  output_filename = prodgenie_brb_nu_cosmic_uboone_g4.root
  output_to_transfer = prodgenie_brb_nu_cosmic_uboone_g4.root

[detsim]
  FHICL = standard_detsim_uboone.fcl
  expected_lifetime = 300m
```
CI validation and grid support


---

**stages**

**#jobs, #events/job**

**dCache area** used to store exp code used to be validated, data, plots and logs produced by the experiment workflow

**mailing list** to report the completion of the CI validation phase

---

```bash
[global]
stages_phase_1 = gen g4 detsim reco1 reco2 ana
njobs_phase_1 = 5
nevents_per_job_phase_1 = 2
stages_phase_2 = merge
njobs_phase_2 = 1
validation_process = uBooNE calorimeter validation
validation_function = /pnfs/uboone/scratch/users/vito/CI_tests_grid/
max_log_size = 2 097 1520
notify_grid_email_to = vito@fnal.gov
POMS_CAMPAIGN_ID = 55

[gen]
FHICL = prod_muminus_0.1-2.0GeV_isotropic_uboone.fcl
expected_lifetime = 50m
memory = 2000MB
memory = 20GB
disk = lar
executable = --rethrow-all
arguments = prodgenie_brb_nu_cosmic_uboone_gen.root
output_filename = prodgenie_brb_nu_cosmic_uboone_gen.root
output_to_transfer =

[g4]
FHICL = standard_g4_uboone.fcl
expected_lifetime = 50m
memory = 2000MB
memory = 20GB
disk = lar
executable = --rethrow-all
arguments = gen
input_from_stage
input_filename = prodgenie_brb_nu_cosmic_uboone_gen.root
output_filename = prodgenie_brb_nu_cosmic_uboone_g4.root
output_to_transfer =

[detsim]
FHICL = standard_detsim_uboone.fcl
expected_lifetime = 300m
```
CI validation and grid support


```
[g4]
FHICL = standard_g4_uboone.fcl
expected_lifetime = 50m
memory = 200GB
disk = lar
executable = dBhso1
arguments = --rethrow-all
data = prodgenie_brh_nu_cosmic_uboone_gen.root
output_to_transfer = prodgenie_brh_nu_cosmic_uboone_gen.root

[detsim]
FHICL = standard_detsim_uboone.fcl
expected_lifetime = 300m
memory = 200GB
disk = lar
executable = dBhso1
arguments = --rethrow-all
data = prodgenie_brh_nu_cosmic_uboone_gen.root
output_to_transfer = prodgenie_brh_nu_cosmic_uboone_gen.root
```

Stage FHiCL file

Stage executable

Stage jobs resources: expected lifetime, memory, disk
Running tests


• There are three methods to test the code
  • **Automatic trigger of a CI build**
    • *Occurs when users* `git push` *in the develop branch of the repository*
    • Tests the develop branch of LArSoft + experiment
    • Default CI workflow consist of:
      • checkout, build, unit tests (174), CI tests (29 quick CI tests)
  • **Manual trigger of a CI build**
    • Developers can run a script to trigger a CI build to test specific branches
      • Developer can specify branches repository by repository
      • Default branch is “develop”
  • **NOTE: all code used for tests must be committed and pushed to the central repositories (a security requirement)**
• **Running tests locally**
  • The code can be tested locally to verify that it doesn't break anything
  • This can be achieved using the `test_runner` script
    see https://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki for more details
Obtaining results

- The CI web application components
- Which information you can get
- Email report
Obtaining results

- The CI web application is a web interface that easily provides information about the status of the code.
It is available at the URL: [http://lar-ci-history.fnal.gov/LarCI/app](http://lar-ci-history.fnal.gov/LarCI/app)

- In the next few slides we will walk through the CI web interface
- I'll describe:
  - its different components
  - the available information you can access

### Obtaining results

- **In-line documentation:**

  ![Multiplatform Continuous Integration for LArCI](image)

  **Link to wiki pages with description of the CI web application components**

<table>
<thead>
<tr>
<th>Build</th>
<th>Start Time</th>
<th>Build Type</th>
<th>checkout</th>
<th>build</th>
<th>unit_test</th>
<th>install</th>
<th>cl_tests</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2017-04-26 21:24:55.528732</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2017-04-26 21:24:55.528732</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Progress Legend:**
- Running
- Pending
- Succeeded
- Warning
- Failed
- Skipped
Obtaining results

- **CI Build details**

- Hovering the mouse on the “Build” box you will get a tooltip that shows:
  - Trigger reason (T:)
  - Workflow (W:)
  - Personality (P:)

---

V. Di Benedetto

LArSoft Workshop
Obtaining results

- Checkout details

- Hovering the mouse on the “checkout” box you will get a tooltip that shows:
  - repository name
  - git description revision
Obtaining results

- Unit test details

- Hovering the mouse on the “unit_test” box you will get a tooltip that shows:
  - Unit tests stats:
    - total number;
    - succeeded;
    - failed;
    - skipped.
Obtaining results

- CI tests details

- Hovering the mouse on the “ci_test” box you will get a tooltip that shows:
  - CI tests stats:
    - total number;
    - succeeded;
    - warning;
    - failed;
    - Skipped.

- Summary of CI tests status.
Obtaining results: CI tests view

- The “CI tests view”
- Which information you can get here

To access the CI tests view, click on the `ci_tests` box.
Obtaining results: CI tests view

Phase ci_tests Runtime History

Select date range:
From: mm/dd/yyyy
To: mm/dd/yyyy
Update | Clear

Phase: ci_tests

# Test Name
1. ci binary run regression test_larsoft
2. ci binary run regression test_larsoft
3. ci binary run regression test_dasmed
4. ci binary run regression test_dasmed
5. ci binary run regression test_protobuf
6. ci binary run regression test_usbencode
7. ci binary run regression test_dasmed
8. ci binary run regression test_dasmed
9. ci binary run regression test_protobuf
10. ci binary run regression test_usbencode
11. ci binary run regression test_dasmed
12. ci binary run regression test_dasmed
13. ci binary run regression test_protobuf
14. ci binary run regression test_usbencode
15. ci binary run regression test_dasmed
16. ci binary run regression test_dasmed
17. ci binary run regression test_protobuf
18. ci binary run regression test_usbencode
19. ci binary regression test_dasmed
20. ci binary regression test_protobuf
21. ci binary regression test_usbencode
22. ci binary regression test_dasmed
23. ci binary regression test_dasmed
24. ci binary regression test_protobuf
25. ci binary regression test_usbencode
26. ci binary regression test_dasmed
27. ci binary regression test_dasmed
28. ci binary regression test_protobuf
29. ci binary regression test_usbencode
30. ci binary regression test_dasmed
Obtaining results: CI tests view

Zoom out the view to easily see details

Select date range:
From: 03/14/2017
To: 04/24/2017

Phase ci_tests Runtime History

Number of Tests started

Show Series:
- Number of Tests
- Succeeded
- Failed
- Warning
- Skipped

Phase: ci_tests

```
# ci_tests.log
```

Started: 04/17/2017 13:05:19
Finished: 04/17/2017 13:16:15. Exit code: 0

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci_bilanco_R21_regression_test_lariatsoft</td>
<td>14</td>
</tr>
<tr>
<td>ci_mergeana_regression_test_dune25it</td>
<td>15</td>
</tr>
</tbody>
</table>
Obtaining results: CI tests view

Navigation bar to easily navigate through CI phases
Obtaining results: CI tests view

- Select date range:
  - From: 03/14/2017
  - To: 04/24/2017

- Show Series:
  - Number of Tests
  - Succeeded
  - Failed
  - Warning
  - Skipped

- Time picker to select data range

- Phase ci_tests Runtime History
  - Wall Clock Time (s)
  - Dates: 19 Mar, 26 Mar, 02 Apr, 09 Apr, 16 Apr

- Number of Tests started
  - Dates: 19 Mar, 26 Mar, 02 Apr, 09 Apr, 16 Apr
  - 2017/04/08 05:17:25:
    - Number of Tests: 29
    - Succeeded: 29
    - Failed: 3
    - Warning: 2

- Logs:
  - Started: 04/17/2017 13:05:19
  - Finished: 04/17/2017 13:16:15
  - Exit code: 0

- Tests:
  - ci_blrco_regression_test_larsoft
  - ci_blrco_regression_test_larsoft
  - ci_mergeana_regression_test_dune3kit
Obtaining results: CI tests view

Marker to indicate the selected CI build i.e. this is the data point belonging to the CI build clicked on the main page of the CI web application
Obtaining results: CI tests view

Select date range:
From: 03/14/2017
To: 04/24/2017
Update Clear

Show Series:
- Number of Tests
- Succeeded
- Failed
- Warning
- Skipped

Phase ci_tests Runtime History

Number of Tests started

Check box to select the series to show
Obtaining results: CI tests view

Range selector to zoom in the graph

Phase ci_tests Runtime History

Show Series:
- Number of Tests
- Succeeded
- Failed
- Warning
- Skipped

Number of Tests started

2017/04/08 05:17:25:
Number of Tests: 29
Succeeded: 29
Failed: 3
Warning: 2
Obtaining results: CI tests view

- Access specific CI test logs and stats

Each CI phase has its own log

To access ci test details, click on the specific ci test link
Obtaining results: CI tests details

This page provides:
- Graphs that show resources usage
- stdout and stderr logs
- Backtrace log in case the test crashes
- Statistics like: memory peak (max RSS), %CPU, elapsed time, ...
- Each statistic is a link to the associated graph
Obtaining results: CI tests details

- Graph of RSS memory peak: uboonecode g4 stage as an example
Obtaining results: CI tests details

- Graph of RSS memory peak: uboonecode reco stage 1 as an example
Obtaining results: CI tests view

- The list of CI tests is sorted by status severity
- The CI test status corresponds to a color: red=failure; orange=warning; green=succeeded
- Each CI test link provides logs for the test

<table>
<thead>
<tr>
<th>#</th>
<th>Test Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ci_reco_regression_test_dune35t</td>
</tr>
<tr>
<td>2</td>
<td>ci_g4_regression_test_dune35t</td>
</tr>
<tr>
<td>3</td>
<td>ci_g4_regression_test_dunefd</td>
</tr>
<tr>
<td>4</td>
<td>ci_g4_regression_test_protodune</td>
</tr>
<tr>
<td>5</td>
<td>ci_reco_regression_test_dunefd</td>
</tr>
<tr>
<td>6</td>
<td>ci_reco_regression_test_protodune</td>
</tr>
<tr>
<td>7</td>
<td>ci_blreco_RUN1_regression_test_lariatsoft</td>
</tr>
<tr>
<td>8</td>
<td>ci_blreco_RUN2_regression_test_lariatsoft</td>
</tr>
<tr>
<td>9</td>
<td>ci_detsim_regression_test_dune35t</td>
</tr>
<tr>
<td>10</td>
<td>ci_detsim_regression_test_dunefd</td>
</tr>
<tr>
<td>11</td>
<td>ci_detsim_regression_test_protodune</td>
</tr>
<tr>
<td>12</td>
<td>ci_detsim_regression_test_uboonecode</td>
</tr>
<tr>
<td>13</td>
<td>ci_g4_regression_test_uboonecode</td>
</tr>
<tr>
<td>14</td>
<td>ci_gen_regression_test_dunefd</td>
</tr>
<tr>
<td>15</td>
<td>ci_gen_regression_test_protodune</td>
</tr>
<tr>
<td>16</td>
<td>ci_gen_regression_test_uboonecode</td>
</tr>
<tr>
<td>17</td>
<td>ci_mergeana_regression_test_dune35t</td>
</tr>
<tr>
<td>18</td>
<td>ci_mergeana_regression_test_dunefd</td>
</tr>
<tr>
<td>19</td>
<td>ci_mergeana_regression_test_protodune</td>
</tr>
<tr>
<td>20</td>
<td>ci_mergeana_regression_test_uboonecode</td>
</tr>
<tr>
<td>21</td>
<td>ci_mergeana_regression_test_uboonecode</td>
</tr>
<tr>
<td>22</td>
<td>ci_reco1_regression_test_uboonecode</td>
</tr>
<tr>
<td>23</td>
<td>ci_reco2D_RUN1_regression_test_lariatsoft</td>
</tr>
<tr>
<td>24</td>
<td>ci_reco2D_RUN2_regression_test_lariatsoft</td>
</tr>
<tr>
<td>25</td>
<td>ci_reco2_regression_test_uboonecode</td>
</tr>
<tr>
<td>26</td>
<td>ci_reco_regression_test_argoneutcode</td>
</tr>
<tr>
<td>27</td>
<td>ci_sim_regression_test_argoneutcode</td>
</tr>
<tr>
<td>28</td>
<td>ci_slicer_RUN1_regression_test_lariatsoft</td>
</tr>
<tr>
<td>29</td>
<td>ci_slicer_RUN2_regression_test_lariatsoft</td>
</tr>
</tbody>
</table>
Obtaining results: Unit test view

- The “Unit test view”
- Which information you can get here

To access the unit tests view, click on the **unit_tests** box
Obtaining results: Unit test view

Logs

Stats

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Start Time</th>
<th>Duration (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>04/17/2017 12:45 45</td>
<td>6.77</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 45</td>
<td>1.26</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 39</td>
<td>21.90</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 44</td>
<td>11.19</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 42</td>
<td>1.18</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 39</td>
<td>26.17</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 45</td>
<td>2.80</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 48</td>
<td>0.55</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 30</td>
<td>1.72</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 39</td>
<td>1.65</td>
</tr>
<tr>
<td>...</td>
<td>04/17/2017 12:45 44</td>
<td>6.95</td>
</tr>
</tbody>
</table>
Obtaining results: CI validation view

- uBooNE calorimeter validation as an example
Obtaining results: CI validation view

- **uBooNE calorimeter validation as an example**

  Progress bars show the number of events available for each stage, this is based on the number of successful jobs.

- **Experiment workflow stages**

  - The CI validation can process a workflow with as many stages as needed.
  - The stages can be grouped together in the same grid job to minimize I/O and improve grid job efficiency.

  Hovering the mouse over the stages box a tooltip shows the status of that stage jobs.
Obtaining results: CI validation view

- uBooNE calorimeter validation as an example

- CI validation phase inherits the CI functionality to upload and show plots

- This provides the user an easy access to CI validation results

- If something looks suspicious the user have access to the job outputs for a further analysis
Obtaining results: CI validation view

uBooNE calorimeter validation

<table>
<thead>
<tr>
<th>Build</th>
<th>Start Time</th>
<th>Build type</th>
<th>gen</th>
<th>gl</th>
<th>dataim</th>
<th>rec1</th>
<th>rec2</th>
<th>run</th>
<th>merge</th>
<th>Progress Legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>lar_ci_test255 (ubool8)</td>
<td>2017-04-12 22:30:33.985825</td>
<td>sll5.c10:profi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Successful</td>
</tr>
</tbody>
</table>

- By clicking on a stage box more info are available
  - jobs stats which include: resident memory peak, elapsed time, file size
  - job status details

Test: recol

Jobs stats plots for each stage

Jobs status details for each stage
Obtaining results: email report

• Use different mailing lists according to the CI build status.

• Possible CI build status are:
  • **successful**: any issue reported
  • **warning**: the code run properly, but some check on the output is not successful
  • **failed**: something is not working properly, it can be a failure in the build of the code, or running unit/integration tests

• There is still the possibility to send a report whatever is the status of the CI build

• Each experiment can choose a set of mailing lists to receive email reports according to the CI build status
• The mailing list can also be a list of individual user email
• Users that trigger a manual CI build will receive an email report
Obtaining results: email report

- Provides quick information in the subject;
- details about the CI build;

CI build of defaultwf workflow for slf6 Succeeded

Experiment: LArSoft uBooNE DUNE LArIAT ArgoNeuT
Code revision: develop
Build type: e10:prof
Platform: slf6
Workflow: defaultwf
Personality: mrb
Jenkins project: lar_ci, build 58
Build slave: buildservice002.fnal.gov
Trigger: git push on develop branch
For more details about this CI build status and to access the logs, please, see the web monitoring application. To access more logs see the Jenkins project web page.

Test Suite: quick_test_uboonecode quick_test_dunetpc quick_test_lariatsoft quick_test_argoneutcode
Start 1: ci_gen_regression_test_uboonecode
Start 2: ci_g4_regression_test_uboonecode
Start 3: ci_defsim_regression_test_uboonecode
Start 4: ci_reco1_regression_test_uboonecode
Start 5: ci_reco2_regression_test_uboonecode
Start 6: ci_reco3_regression_test_uboonecode
Start 7: ci_gen_regression_test_dune35t
Start 8: ci_g4_regression_test_dune35t
Start 9: ci_defsim_regression_test_dune35t
Start 10: ci_reco_regression_test_dune35t
Start 11: ci_reco3_regression_test_dune35t
Start 12: ci_gen_regression_test_duned
Start 13: ci_g4_regression_test_duned
Obtaining results: email report

- Support “warning” status for CI tests;
- details about CI tests that are not successful

CI build of defaultwf workflow for slf6 Warning at phase ci_tests

<table>
<thead>
<tr>
<th>CI Test Status</th>
<th>Error Message</th>
<th>Required Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ci reco2D RUN1 regression_test_larlatsoft</td>
<td>Differences in products names</td>
<td>Request new reference files</td>
</tr>
<tr>
<td>ci reco2D RUN2 regression_test_larlatsoft</td>
<td>Differences in products names</td>
<td>Request new reference files</td>
</tr>
</tbody>
</table>

Warning means: experiment code runs successful, but some check on the output is not successful.

For more details about the warning phase, check the cl_tests phase logs

Experiment: LArSoft uBooNE DUNE LArIAT ArgoNeuT
Code revision: develop
Build type: e10:prof
Platform: slf6
Workflow: defaultwf
Personality: mrb
Jenkins project: lar_ci, build 54
Build slave: buildservice002.fnal.gov
Trigger: git push on develop branch
For more details about this CI build status and to access the logs, please, see the web monitoring application.
To access more logs see the Jenkins project web page.

Test Suite: quick_test_uboonecode quick_test_dunetpc quick_test_larlatsoft quick_test_argoneutcode
   Start 1: ci_gen_regression_test_uboonecode
Upcoming new features

- **CI workflow reorganization**
  - each experiment code will be tested independently each other, but still depend on the build/test of LArSoft
  - CI build will test only code from experiments that can be potentially affected by a given commit
  - CI email report will provide independent info for each experiment
  - Support to get email report for individual CI tests

- **CI Web App reorganization**
  - LArSoft and experiment code test information are shown in dedicated tabs

- **Add support for memory profiling to intercept memory leak**
Getting help

• The redmine wiki documentation is available at:
  LArCI: http://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki

• The CI web application provides tooltips and online documentation

• To request new features open a SNOW ticket
  Scientific Computing Services > Scientific Production Processing > Continuous Integration

• For support/question send an email to: ci_team@fnal.gov

• As last resort, talk to LArSoft Team
Thank you for your attention