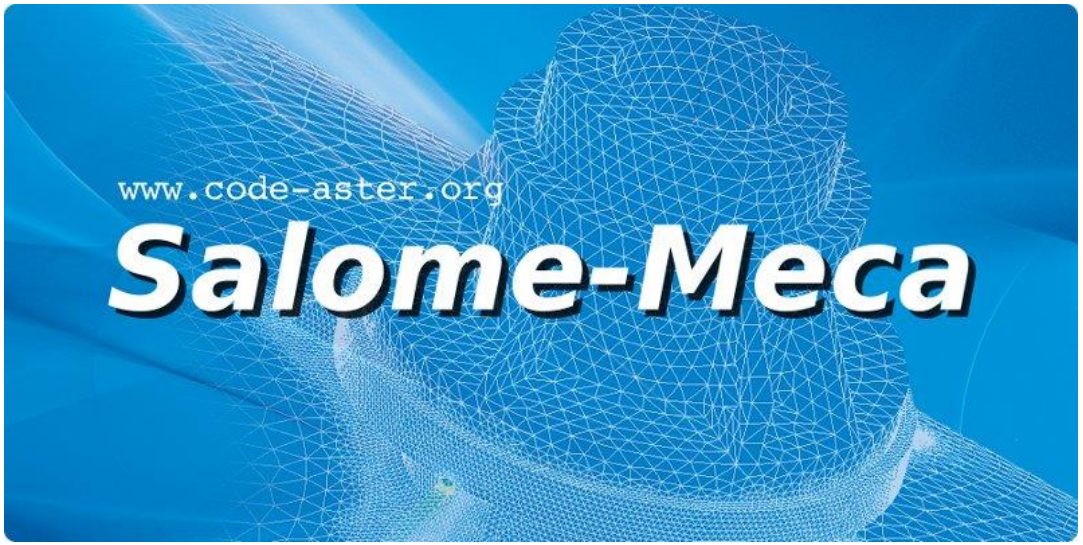
Development in code_aster
Integration process



Code_Aster, Salome-Meca course material

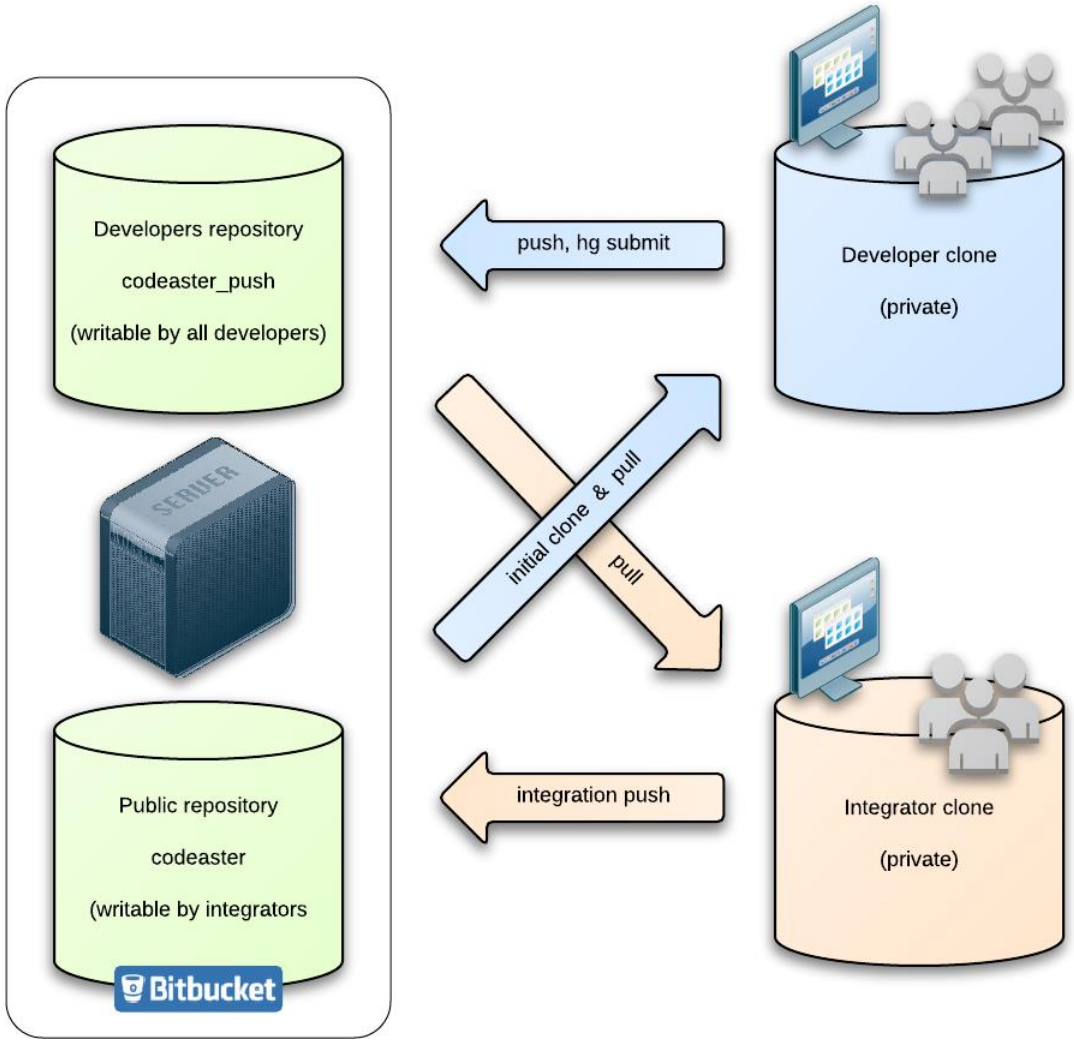
Repositories Map

What does *integration* mean?

- Your private development will become officially published into the main public repository

Workflow

- Your changesets are exchanged through a common repository writable by all developers
- An integrator pulls your changes and checks them
- If the tests were passed with success, your changesets are pushed to the public repository (also mirrored on Bitbucket)



https://bitbucket.org/code_aster

Prepare your repository

One problem = one changeset

Keep intermediate revisions. *Restrictions: all revisions must compile!*

Fold meaningless changesets (typo, comments…) with another.

All the revisions contributing a same issue must be contiguous.

Commit message

In english (published on bitbucket).

Do not copy the title of the issue (the problem).

The message must describe what it was done (the solution).

Start with the issue number the changeset solves: [#012345] (or [bb0107] for bitbucket ticket).

Submit a linear history

Multiple (~10) branches merged during a week could be too complicated to understand.

Use `hg rebase`, not `hg merge`

Prepare your source files

Respect of the coding conventions

Checked by *aslint.*

You must fix (E)rror and error by (C)onvention messages.

You should remove (W)arnings and be carefull about (I)nformation messages.

Request for integration

Check that the submitted revisions are children of the last public revision.

Check the issues status: must be ‘valide_eda’.

Local compilation and installation.

Checking of the source files.

Run of a short list of testcases.

Steps to a definitive integration

Merge with other developers.

Manual code review (too many warnings, clumsy programming…).

Run of all the testcases from the src repository on the main development server.

Differed: Run of the testcases on other supported architectures.

Live demo

Always start a new development from the last public reference

```
hg log –rev reference
hg update reference
```

Check each branch with *aslint*

```
aslint –hg refe
```

Use rebase to submit several developments

```
hg rebase –s REV1 –d REV2
hg up
```

Let’s go

```
hg submit
```

End of presentation

Is something missing or unclear in this document?

Or feeling happy to have read such a clear tutorial?

Please, we welcome any feedbacks about Code_Aster training materials.

Do not hesitate to share with us your comments on the Code_Aster forum dedicated thread.