Einstein Toolkit Web infrastructure overview

Steve Brandt (LSU), Roland Haas (NCSA)

Other sources of information

- the ET wiki at https://docs.einsteintoolkit.org/et-docs/Services is the **authoritative** source of information on ET web infrastructure
- current ET repos eg https://bitbucket.org/einsteintoolkit/einsteinbase/admin
- README files in bitbucket / github / etc repos
- minutes of the calls https://docs.einsteintoolkit.org/et-docs/Phone_Call_Minutes
- the mailing list users@einsteintoolkit.org searchable on ET website: https://www.einsteintoolkit.org/support.html
- the mailing list maintainers@einsteintoolkit.org to which the maintainers are subscribed (its archive is **private**). This lists accepts **posts** from **anyone**.

Components of ET web infrastructure

The Einstein Toolkit, being a community project involving members that are at geographically distant institutions in different time zones, relies on a collection of web services to distribute information and code.

- the main website: https://www.einsteintoolkit.org hosted at LSU
- the mailing list(s) users@einsteintoolkit.org hosted at LSU
- the wiki https://docs.einsteintoolkit.org hosted at LSU
- the issue tracker https://trac.einsteintoolkit.org hosted at LSU
- the tutorial server https://etkhub.ndslabs.org hosted at NCSA
- git code repositories hosted on bitbucket https://bitbucket.org/einsteintoolkit
- svn code repositories https://svn.einsteintoolkit.org hosted at LSU
- the Jenkins instance https://build.barrywardell.net/ hosted at NCSA
- the gitter chat https://gitter.im/EinsteinToolkit
- the github orignization https://github.com/einsteintoolkit used with gitter

Jenkins continuous integration system

- Jenkins builds the Einstein Toolkit and runs the testsuites whenever a commit to the "master" branches is detected.
- the reason we are permissive with commits since it will catch buggy commits.
- status updates: test@einsteintoolkit.org
- admin interface and details of builds: https://build.barrywardell.net/
  - owned by Barry Wardell
  - ask for account from Ian Hinder
  - only NCSA slave is active, runs on Nebula VM system
  - ask Roland Haas for access to Nebula
  - VMs run Ubuntu 16.04.2 LTS and have 2 cores and 3.5GB of RAM
- request access to build slave like VM from Ian Hinder: login.barrywardell.net

Four VMs working in concert

- build.barrywardell.org: Jenkins "master"
- one anonymous "slave" VM runs the tests
- et.barrywardell.org: listens to changes in ET repos, updates super-repository
- login.barrywardell.org: mimics "slave" VM for digging down in case of failures

Repos and docker images:

- https://bitbucket.org/einsteintoolkit/einsteintoolkit - Git super-repository for the Einstein Toolkit source code.
- https://bitbucket.org/ianhinder/cactusjenkins - Scripts used by the ET Jenkins build server
- https://bitbucket.org/ianhinder/et-jenkins-slave - docker image for build slave
- https://bitbucket.org/ianhinder/ncsajenkins - Scripts to set up a build node on Ubuntu, currently used on a Nebula VM
- [RH] no documentation for et.barrywardell.org

Jenkins continuous integration system

- Jenkins slave VM is described in https://bitbucket.org/ianhinder/ncsajenkins
- Docker image in slave VM is in https://bitbucket.org/ianhinder/et-jenkins-slave
- Jenkins scripts inside of Docker are in https://bitbucket.org/ianhinder/cactusjenkins
- Jenkins currently runs Ubuntu 16.04 LTS
- uses generic.cfg to build
- raw logs of build eg here https://build.barrywardell.net/job/EinsteinToolkit/lastCompletedBuild/consoleText
-

https://build.barrywardell.net/job/EinsteinToolkit/

- main ET build reports, build__1_2.log and build__2_1.log are 1 and 2 rank test results
- console logs (life if currently running) https://build.barrywardell.net/job/EinsteinToolkit/lastCompletedBuild/consoleText
- Failures point to commit introducing issue, eg: https://build.barrywardell.net/job/EinsteinToolkit/1340/

ET super-repository

provides a single git repository tracking all ET changes for the benefit of Jenkins

- git repos a submodules
- svn repos (ExternalLibraries) via git-svn copies then submodules

"master" branch needs to be updated when a new git repo is added to einsteintoolkit.th by adding the new repo as submodule

"release" branch needs to be updated for each release by updating the submodules, updating the git-svn repos and updating the submodules. [RH: this is currently broken]

This is probably the most complex setup we use in the ET web infrastructure anywhere

Benefits

- unified view of ET repos lets Jenkins track down exact commit introducing failure
- super-repository useful for using git-bisect to track down bugs in ET repos
- working right now

Downsides

- git submodules are complex
- manual interaction required when repos are added / removed from einsteintoolkit.th
- (currently) single point of failure with only one knowledgeable maintainer

Maybe use GetComponents and give up on failure tracking

ET tutorial server

First point of contact for potential users trying out the Einstein Toolkit

Tutorial is a jupyter notebook maintained in https://github.com/nds-org/jupyter-et.

Initially using web-platform in www.ndslabs.org backed up by VMs on TACC's JetStream. Currently down due to unacceptable use. In process of setting up Jupyter Hub server at https://etkhub.ndslabs.org using github.com for authentication.

Whitelisting of users by Steve or Roland.

Current setup provided by Craig Willis (staff @ NCSA), using OpenStack, Kubernetes, Docker.

Access to nds-org repo granted by Craig Willis (via Roland or Steve).

Tutorial VM

- 4 cpus, 8 GB of RAM **shared among users**
- **no persistent data**, all data removed when jupyter instance ends (4 days)
- access (user, admin) through Steve or Roland
  - admin access (for ET maintainers): ubuntu@etkhub.ndslabs.org:whitelists/users.txt
  - need signup page on ET website, notify maintainers@einsteintoolkit.org
  - need warning at tutorial server website
  - policy on when / how to remove inactive users.
- Access to NCSA VM control panel through Roland or Gabrielle.

https://github.com/nds-org/jupyter-et/blob/master/CactusTutorial.ipynb

- instructions must work both online and offline
- commit only instructions not results (clear all output)
- example is TOV star evolution and plots

Einstein Toolkit repositories

majority of repositories owned by the einsteintoolkit or cactuscode bitbucket organizations

- code repositories
- master thornlist: https://bitbucket.org/einsteintoolkit/manifest
- website: https://bitbucket.org/einsteintoolkit/www/
- ET seminars: https://bitbucket.org/einsteintoolkit/seminars/
- release testsuite results: https://bitbucket.org/einsteintoolkit/testsuite_results

SVN repos for ExternalLibraries and old tutorials

- https://svn.einsteintoolkit.org
- access granted by Steve or Peter

Non ET owned repos:

- Carpet (Erik Schnetter): https://bitbucket.org/eschnett/carpet/
- Kranc (Ian Hinder): https://github.com/ianhinder/Kranc
- Simfactory (ET maintainers): https://bitbucket.org/simfactory/simfactory2
- GetComponents (ET maintainers): https://github.com/gridaphobe/CRL
- tutoria (NDS labs)l: https://github.com/nds-org/jupyter-et

ET tutorial and website hosting

- https://github.com/nds-org/jupyter-et
- https://github.com/stevenrbrandt/et-websites

Maintainers have write access to all and admin access to the bitbucket and github organizations

Einstein Toolkit bitbucket repo layout

- owned by einsteintoolkit or cactuscode organizations
  - ET administrator group has "admin" access
  - ET developers group has "write" access
  - managed via https://bitbucket.org/account/user/einsteintoolkit/groups/
- no issue tracker, https://trac.einsteintoolkit.org is used for all of ET (except Kranc)
- no wiki, we use https://docs.einsteintoolkit.org
- code repos must send commit emails to commits@einsteintoolkit.org using bitucket's Email service
  - Bitbucket is whitelisted for that mailing list
  - github currently has no similar functionality
- code repo housekeeping
  - master branch is current development
  - ET_YYYY_MM is release branch
  - ET_YYYY_MM_vV is release tag
- suggested commit / branch layout
  - name own branches $USER/branchname
  - one commit per change
  - commits should not span thorns
  - prefix commit message by "Thornname: "
  - make commit message useful on its own
    - trac goes away
    - bitbucket will go away
    - you will go away
- **all (non-trivial and non-documentation)** changes need review
  - create a bitbucket pull request
  - create a trac ticket and set its state to "review"
  - main authors are exempt in their own project (Ian: Kranc, Erik: Carpet, Eloisa: CT_MultiLevel, …) but may still request review
  - **should improve code, not block progress**
  - can alway revert changes when bugs are found
- https://docs.einsteintoolkit.org/et-docs/How_to_Review_a_Patch

Einstein Toolkit github repo layout

GitHub

- owned by einsteintoolkit organization
  - ET administrator group are "owners"
  - ET developers group has "write" access
  - managed via https://github.com/orgs/EinsteinToolkit/people
- no issue tracker, https://trac.einsteintoolkit.org is used for all of ET (except Kranc)
- no wiki, we use https://docs.einsteintoolkit.org
- code repos cannot currently send emails
  - CRL and Kranc are only code repo so far
  - no repos owned by einsteintoolkit exist
- GitHub authentication used by
  - https://gitter.im/einsteintoolkit
  - https://ekthub.einsteintoolkit.org

Einstein Toolkit chat channels

- two chat channels available
  - IRC: #cactus on irc.oftc.net (abandoned)
  - Gitter: https://gitter.im/EinsteinToolkit
- intended to provide channel for real time questions
  - in the past offered "virtual office hours" but had little success
- IRC channel no longer in use, Frank, Erik are operators on the channel
- Gitter offers web-based chat and an IRC bridge
  - text only chat, no images embeddable
  - code snippets and images can be shared via github's gists feature
  - used for Working Groups as well
  - currently very little traffic
  - gitter channel name is tied to einsteintoolkit github organization
  - admin status on github inherited on gitter

Einstein Toolkit release process (the plan)

https://docs.einsteintoolkit.org/et-docs/Release_Process

Long-term planning

A few months or weeks before the release:

- Decide it is time to make another release
- Choose features which are going to be included, and those which won't be included
- Choose a tentative date
- Begin discussions on the mailing list, reminding people to look at test cases, review patches etc.

One or two weeks before the release

- Set up a wiki planning page for the release
- Ask for volunteers, assign tasks to people
- Review thorns for code quality, documentation, test cases
- Decide on a thorn list
- Decide on a list of release-critical systems
- Collect list of new features, newsworthy items, and acknowledgements for release announcement on wiki
- verify that gallery runs and example parfiles still work with new release
- Choose a concrete date
- Announce date publicly

One or two days before the release

- Have a telecon with the release team, discuss (and modify if necessary) release process and responsibilities
- Get a list of all repositories that are involved (including repositories for tools such as GetComponents)
- Publicly declare a freeze on all involved thorns and tools
- Ensure that any manually-generated files are consistent (i.e. regenerate McLachlan, WeylScal4 and EinsteinExact from their source with the current version of Kranc, generate the Cactus autoconf script, the Cactus loop macros etc)
- Test all thorns on all systems, collect status reports on wiki
- verify that new users tutorial still works
- Update release planning wiki page with peoples', thorns', and systems' statuses
- Set up a (smaller) technical release team who will be available all day on the day of the release
- Draft release announcement

Einstein Toolkit release process (the plan)

https://docs.einsteintoolkit.org/et-docs/Release_Process

The release

- Have the technical release team meet in the same room
- Briefly re-check status of thorns and machines
- Possibly disable all outside write access to all critical repositories
- CREATE THE RELEASE BRANCH
  - svn: svn copy https://<repository-server>/<repository-path>/trunk https://<repository-server>/<repository-path>/branches/ET_2011_05
  - git: git push origin origin:refs/heads/ET_2011_05
- Update simfactory.org https://svn.cct.lsu.edu/repos/numrel/simfactory2/www/
- Create a release branch in master git repo at https://bitbucket.org/einsteintoolkit/einsteintoolkit via "git checkout -b ET_XXXX_YY ; sed -i 's/master/ET_XXXX_YY/' ; git add .gitmodules ; git commit -m '.gitmodules: Create release branch"
- Update ET Jenkins build for release branch https://build-test.barrywardell.net/job/EinsteinToolkitReleased/configure and trigger a build
- Possibly update version numbers in stable/development branches
- Update component lists to point to new stable version
- Check out release branch on all systems, re-run all quick tests
- Update documentation on web sites (set up fresh copies of pdfs/htmls)
- Update tutorials on web sites (version numbers, urls)
- Update documenation and examples of simfactory and Kranc to refer to the current release
- Re-enable write access to all repositories
- Finalise release announcement
- ANNOUNCE: users@einsteintoolkit.org, {news|users}@cactuscode.org, Jennifer Claudet <jennifer@cct.lsu.edu> for AllCCT, http://hyperspace.uni-frankfurt.de/, http://astro-sim.org/, HPCWire

After the release

- Watch mailing lists for problem reports
- Use released version to repeat a few production runs
- Update this page with new lessons learned

Einstein Toolkit release process (reality)

https://docs.einsteintoolkit.org/et-docs/Release_Process

- we release about every 6 months, unless core release team cannot dedicate time
  - current release months: September, March
- about N months before the release, check on new thorns to be included
  - thorns cannot be included "just before" the release
  - frequent, periodic prods and a dedicated champion help to include a thorn
- produce a baseline status of test failures on current clusters
  - simfactory's distribute script
  - weed out retired clusters
  - update existing cluster simfactory files
- discuss outstanding tickets that are release critical, flag with "ET_YYYY_MM" milestone on trac
- assign tasks / tickets / thorns to volunteers
  - volunteer persons
  - discuss during ET calls
- check on remaining failures
  - only affects few clusters, known fragile test
  - fix failures if fixable in timely manner
  - document failure if cause understood
- draft release statement
  - show to at least one other person
- announce release
  - update website
  - test tutorial
  - test on typical OS using VMs / docker images
    - Linux: Ubuntu, Debian, Fedora
    - MacOS: MacPorts & Homebrew
    - Windows: Ubuntu Linux subsystem
- document what went wrong on wiki
- repeat in 6 months