CODE-RADE a user centric code delivery system for science

Sean Murray — Center for High Performance Computing
Bruce Becker — CSIR, Meraka

Introduction

COntinuous Integration and **DE**livery of **R**esearch **A**pplications in a **D**istributed **E**nvironment (CODE-RADE). The researcher is faced with a paradox. There is a large amount of computing resources available, and a rich, well maintained, interoperable infrastructure of services **however** getting applications onto it requires far too much effort.

CODE-RADE solves this by lowering the barrier to entry to grid or cloud infrastructure, or a single HPC site. Application experts are then able to prove to resource providers that the application will run on the sites execution environment. It very importantly allows easy managing the lifecycle of applications across multiple versions, architectures and configurations. This ensures that once an application is certified, its available on as many sites as possible, and verified to work. This has the knock on effect of promoting collaboration between researcher, research software engineer, and infrastructure providers. The people doing the work get citeable work as a DOI is created for a verified, peer reviewed, CVMFS release that is tied the git hashes.

CODE-RADE has 4 principles:

- **Cross platform** : It builds and test artifacts for an arbitrary set of targets, promoting diversity in computing platforms and ensures proper optimisations and application portability
- **Atomic** : There is fine grained control over dependencies, versions, and targets and relevant actions are taken on each event in the CODE-RADE cycle.
  All these events are communicated to all willing listeners via slack.
- **Community** : No restriction is placed on the applications that can be integrated, anyone is free contribute; applications (dev or testing), resources(human or hardware), code review, etc.
- **Automated** : Heavy reliance on automated agents to reduce bias, lead time. Humans are removed as far as possible.

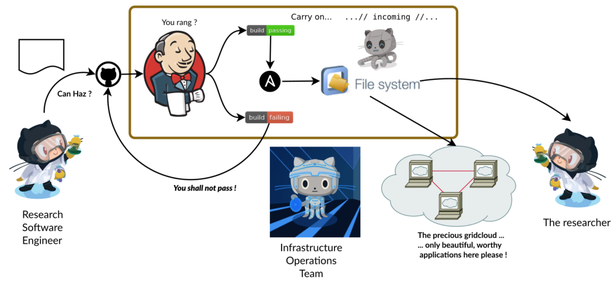
Technology

- Components
  - Version control at Github github.com/SouthAfricaDigitalScience
  - Continuous Integration (Jenkins and Travis) ci.sagrid.ac.za
  - Automated Delivery from successful Jenkins build to (CVMFS) /cvmfs/code-rade.africa-grid
  - Waffle.io is used to coordinate all human tasks, integrating with slack, and linked to github issues.
  - Slack is used extensively for communication via numerous bots to notify on all steps, success/failure/progress.
  - Open discussions at discourse.sci-gaia.eu
- Workflow Each software package has 3 scripts.
  - Build : User and/or expert-defined means to produce executable applications or libraries
  - Test : Infrastructure operator-defined targets provide a means to ensure viability
  - Deliver : Ensure that once the tests pass, the application is available at all sites.
- All builds are inside docker containers so a user could reproduce the constant integration environment locally to test themselves, or attempt to reproduce someone else's results using the exact same software.
- Docker images are publicly available at quay.io/AAROC
- Releases get a DOI, after review.

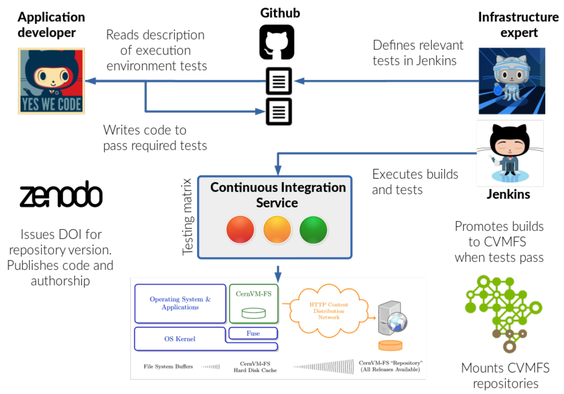
Actors

Researcher Research Software Engineer Infrastructure operations Automated Agents

Workflow



Deployment



Architecture

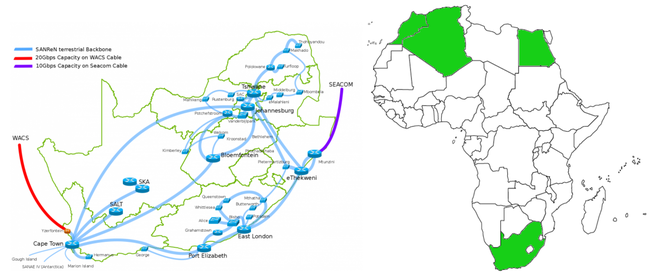
We build the full stack assuming nothing but the base operating system, similar to many hpc sites. We are therefore not dependent on anything local but an operational operating system and our cvmfs mount.

- One can add as many target configurations to the build matrix as can be reliably simulated.
- Currently we have defined three axes :
  - Architecture x86_64, ARM.
  - Operating System CentOS 6,7 Ubuntu 14.04 16.10
  - Site specific - an axis encapsulating the special nature of a site, its interconnects, compiler, local libraries, and a catch all option of generic.

Each application is built for a combination of these dimensions, typically 1x1x4 With different version of complier, mpi implementations, python versions the matrix of builds and grow rapidly. For example NumPy has: 1(ARCH)x1(SITE)x4(OS)x4(PYTHON_VERSION)x3(GCC_VERSION)x4(NUMPY_VERSION)=192

Where can all this can run

University of Free State, University of Cape Town, University of Witwatersrand, Center for High Performance Computing, University of Johannesburg, Moroccan National Grid, Algerian Regional Network, Zewail City of Science and Technology Egypt, and anywhere else prepared to mount the cvmfs repository. We are also integrated into EGI, via AfricaArabia NGI.



Repositories

Numerous pieces of software have been incorporated. The versions are very much dependent on what people want to run. We cover all fields from bioinformatics, astronomy, physics,

- base software from readline, zlib, pcre.
- languages/compilers : python gcc 4.9.4 5.4 6.3
- infrastructure : openpbs, torque.
- hep : root 5.34 6.08 6.10, fastjet, geant4 9 10.1 10.3, pythia 6 8, Fermionic Molecular Dynamics, herwig, hepmc.
- bioinformatics : clustal-omega hmmer oases velvet
- chemistry : autodocksuite gromacs quantum-espresso

The full listing is available at both www.africa-grid.org and ci.sagrid.ac.za

Reproducibility

- Reproducing scientific results.
  One should be able reproduce **exact** workflow with the **exact expression** of the application The entire repository history is kept, allowing one to relate **each successful build** to a CVMFS repository version.
- Reproducing the executable application.
  The **expression** of the application should be reproducible by different build systems The means for expressing the application should be citeable. We this via Zenodo.

Summary and Future

Making the best use of computational resources comes down to applications. We have built a cross-platform, automated system for integration and delivery of applications that :

- Deliver functional, tested, reproducible software to a site
- Recognise and reward the work of research software engineers.
- Depends on only an operating system and cvmfs installed.

Looking forward we would like to simplify the overheads in defining new jobs and versions of software, and provide more varied architecture vectors, thereby expanding where it can run.

Acknowledgements

University of Cape Town, UFS-UV University of the Free State, CSIR our future through science, CHPC, MaGrid Moroccan National Grid, ARN Algerian Research Network, Zewail City Established 2000 Inaugurated 2011

Further Information

github.com/AAROC/CODE-RADE
africa-arabia-roc.slack.com
Sean Murray at murrays@cern.ch
Bruce Becker at bbecker@csir.co.za