Continuous Integration and Continuous Delivery at FRIB

Martin Konrad
Control System Engineer

MICHIGAN STATE UNIVERSITY

U.S. DEPARTMENT OF ENERGY | Office of Science

This material is based upon work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University. Michigan State University designs and establishes FRIB as a DOE Office of Science National User Facility in support of the mission of the Office of Nuclear Physics.

Outline

- Continuous Integration
- Continuous Delivery
- Why use Continuous Delivery?
- Implementation at FRIB

Continuous Integration Principles

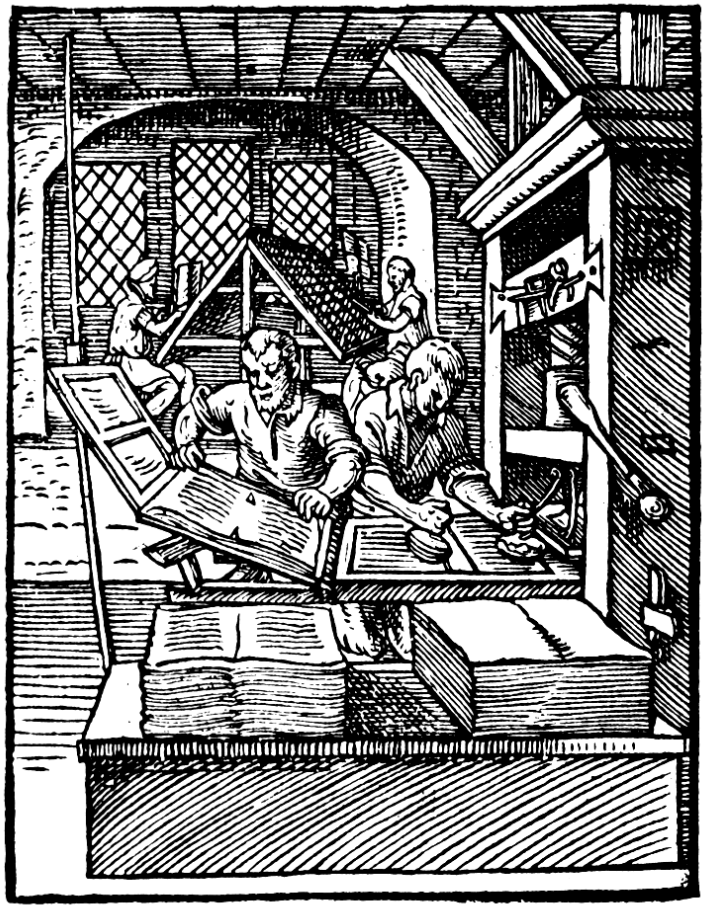
- Maintain a code repository
- Automate the build
- Make the build self-testing
- **Merge changes into a shared mainline several times a day**
- Every commit to mainline should build
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Continuous Delivery

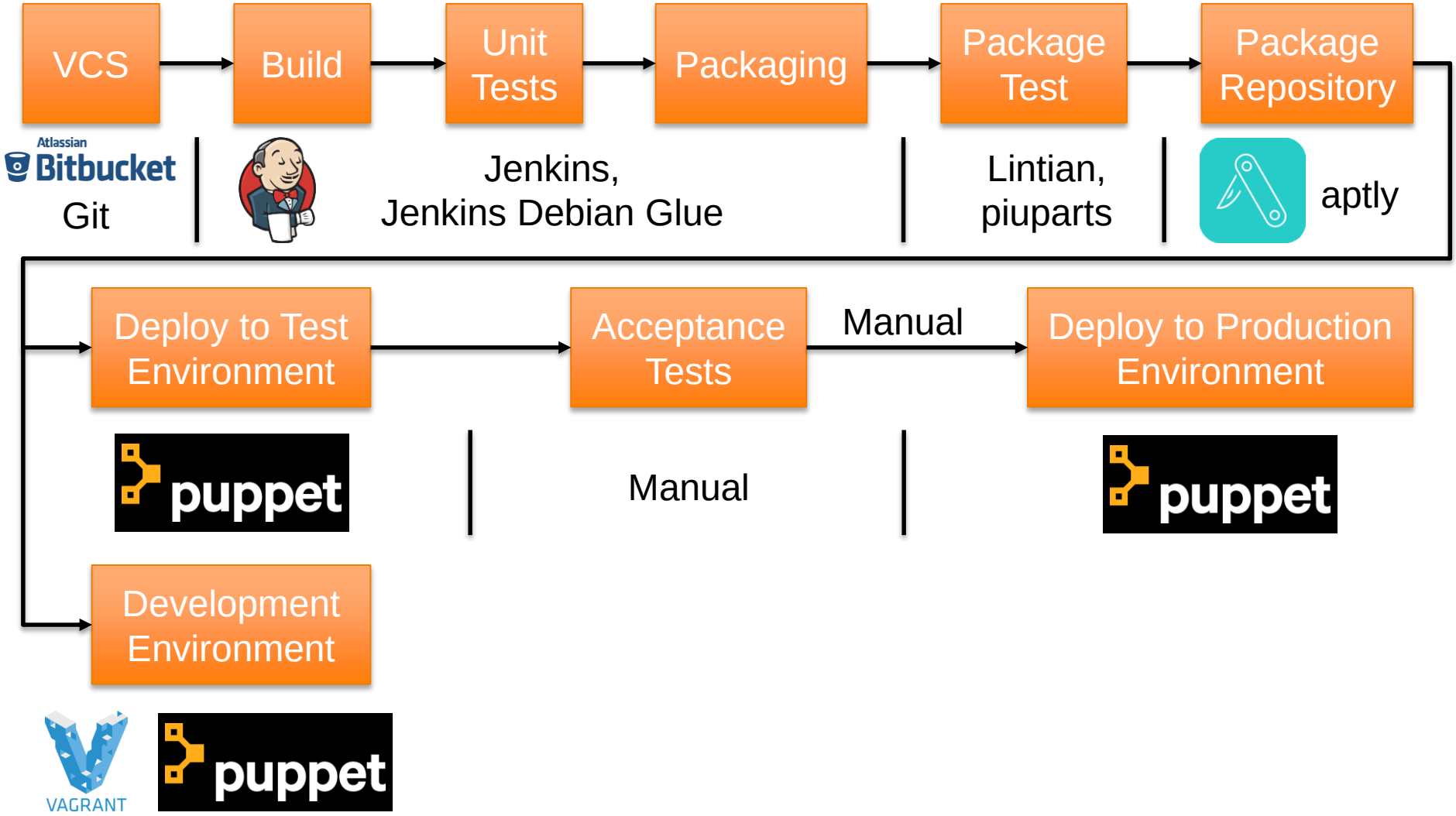
- Continuous *Deployment*
  - Continuous Integration
  - Automatically deploy after each change
- Continuous *Delivery*
  - Continuous Integration
  - Automatically build a candidate after each change that could *potentially* be deployed
  - Deployment process is automated but requires approval (e. g. one-click deployment or merge into a release branch to deploy)

Why use Continuous Delivery?

- Overall we do not expect to save a significant amount of development time, but…
- Allows faster turn-around times
- Helps to catch issues before code is deployed to production system
- Full traceability
- No risk of breaking anything (you can always roll back)
- ➡ Facilitates team work



Continuous Delivery at FRIB



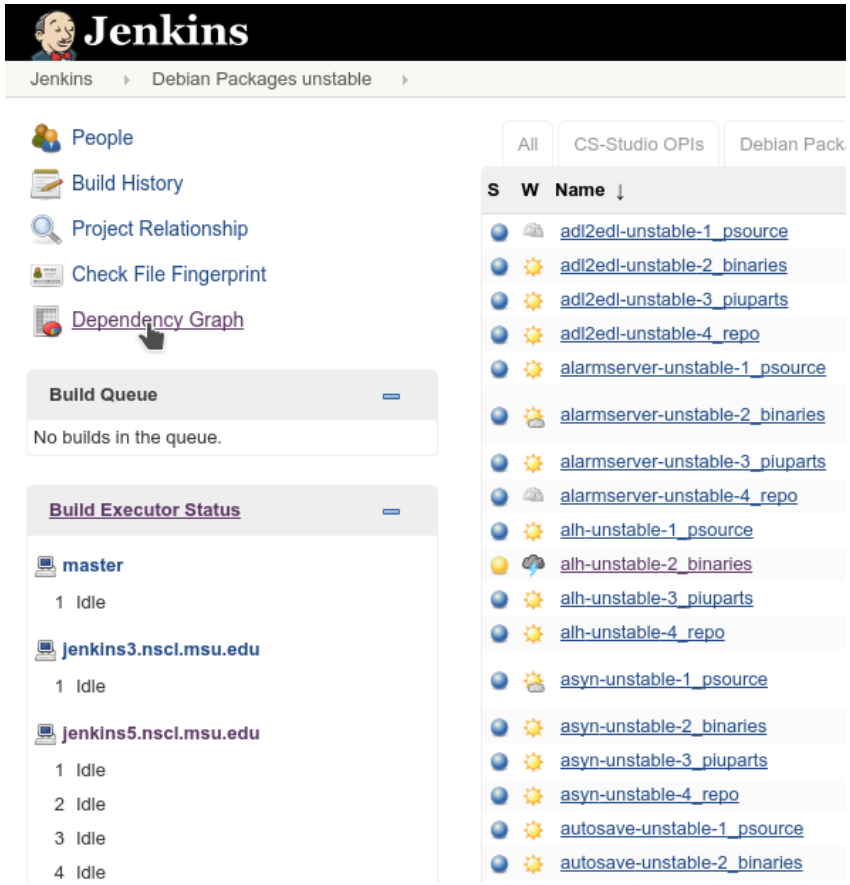
Managing Jenkins

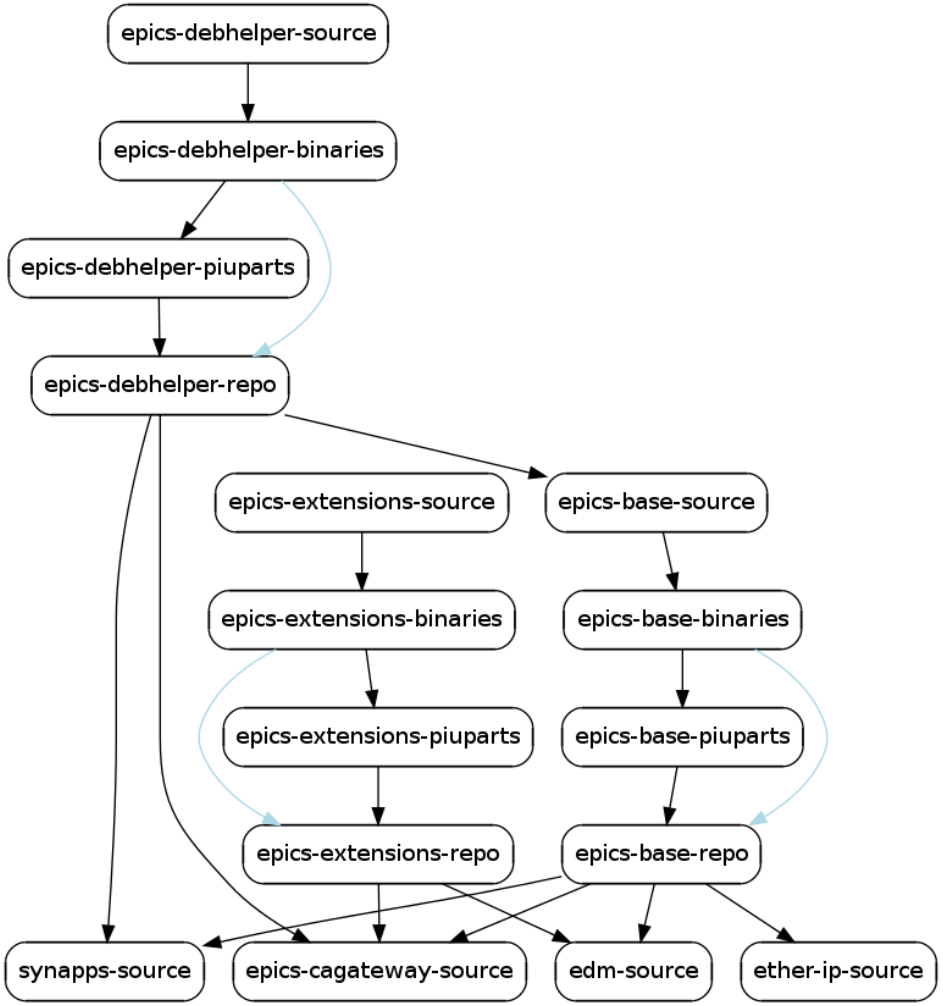
- Jenkins master and build slaves are managed by Puppet
- Jenkins jobs are automatically generated using Jenkins Job Builder
  - Input: short YAML descriptions of the jobs + job templates
  - Output: Jenkins jobs created/changed through API
  - Puppet runs Jenkins Job Builder periodically
- Automation makes sure
  - We can easily add more build nodes/jobs
  - All build machines are exactly the same
  - All jobs of a family (e. g. Debian package jobs) are using the same rules

Dependencies between Debian Packages I

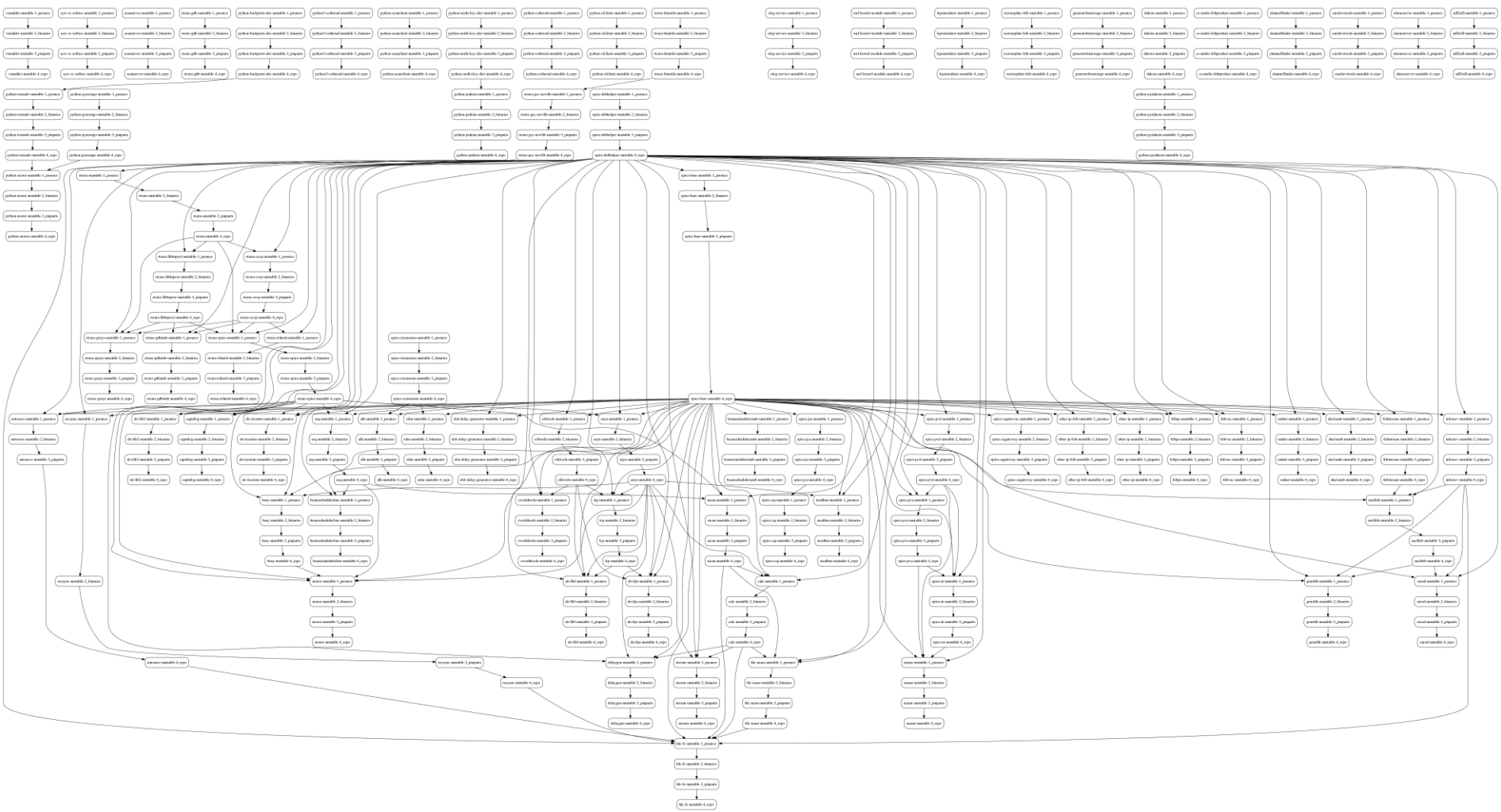
- FRIB's Jenkins cluster has >700 jobs
- When a library is updated we want to automatically rebuild software that depends on it to see if it builds fine against the latest version
- Additional "FRIB" script extracts build dependencies from repositories and translates them into Jenkins triggers
  - Backward dependencies ("depends on") are translated into forward dependencies ("triggers") automatically
  - Puppet automatically runs this script before running Jenkins Job Builder
- A graphical representation of the dependencies is available on the Jenkins web GUI

Dependencies between Debian Packages II





Dependencies between Debian Packages III



Summary

- FRIB is successfully following continuous delivery principles
  - Build and deployment has been automated
  - Manual approval is required before software gets deployed
- Following standards pays off
- Helps improving software quality
- Enables us to follow a more agile development approach

URLs

- Jenkins: https://jenkins-ci.org
- Jenkins Debian Glue: http://jenkins-debian-glue.org
- Jenkins Job Builder: http://ci.openstack.org/jenkins-job-builder/
- Lintian: http://lintian.debian.org
- Package Installation, Upgrading and Removal Testing Suite: http://piuparts.debian.org
- Puppet: http://puppetlabs.com
- Puppet modules and Vagrant files for EPICS: http://stash.frib.msu.edu