Continuous Delivery and Deployment of EPICS IOCs

Martin Konrad
High Performance Controls Team Leader

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.

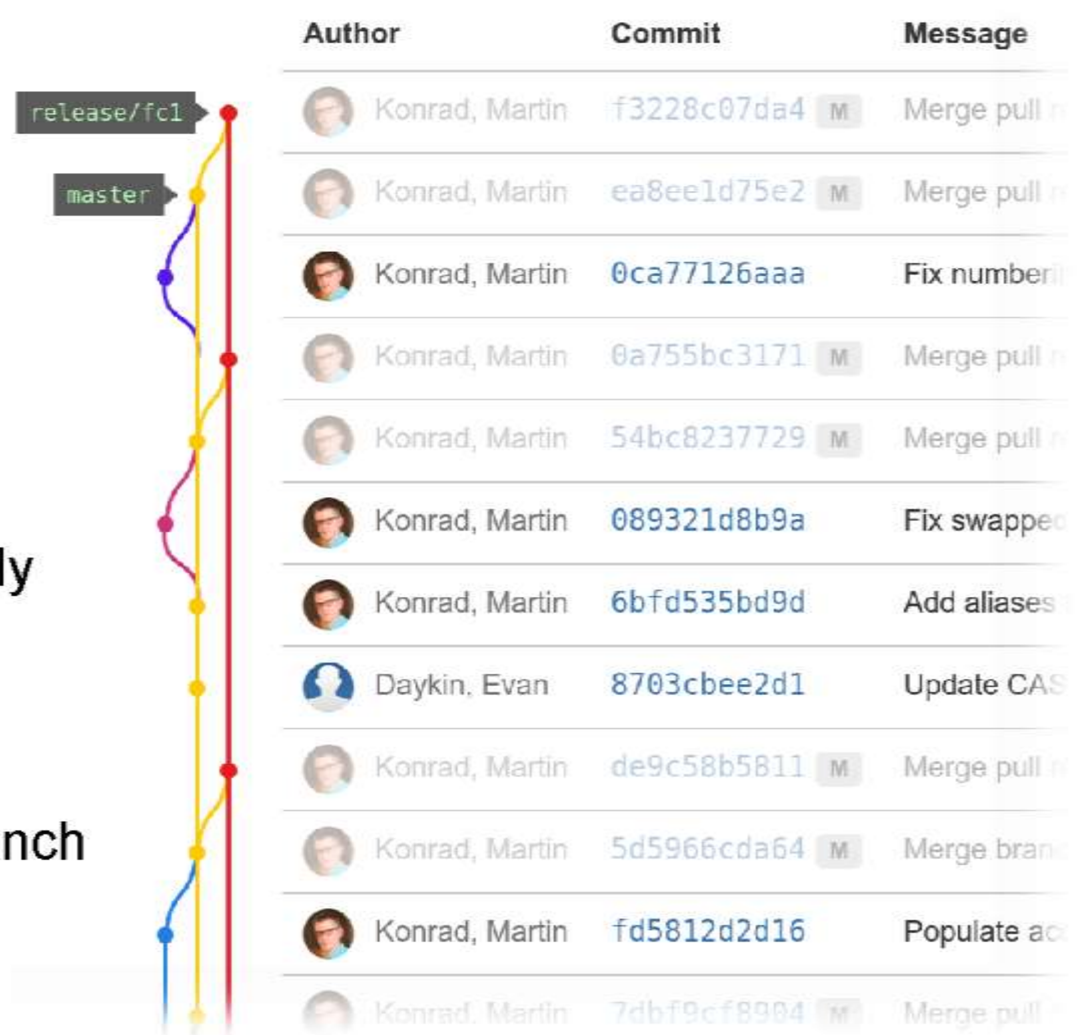
FRIB Controls Environment

Device	Interface to IOC	IOC Runs On	Quantity
Power Supplies, RF Amplifiers, Vacuum Gauges/Pumps,...	Ethernet (TCP with text protocol)	Virtual Machine	Thousands
LLRF Controllers	Ethernet (UDP)	Virtual Machine	~350
MPS Controllers	Ethernet (UDP)	Virtual Machine	~50
MTCA.4 Systems	PCIe	MTCA CPU (Intel)	~25
PLCs	Ethernet	Virtual Machine	~20 processors
Timing Master/Receiver	PCI	cPCI CPU (Intel)	2

- Almost all IOCs run on virtual machines in the data center
  - Improves availability
  - Reduces hardware cost and maintenance burden
  - Resources can be assigned flexibly
- All IOC machines run Debian GNU/Linux 8

Source Code is Under Revision Control

- All control system code is stored in a central Git version control system (VCS)
  - Development happens on feature branches
  - Merge to “master” branch when feature is complete
  - “master” gets deployed to test environment automatically
  - Release branch gets deployed to production environment
  - Branch permissions prevent accidental push to “release” branch
    - » Pull requests are enforced



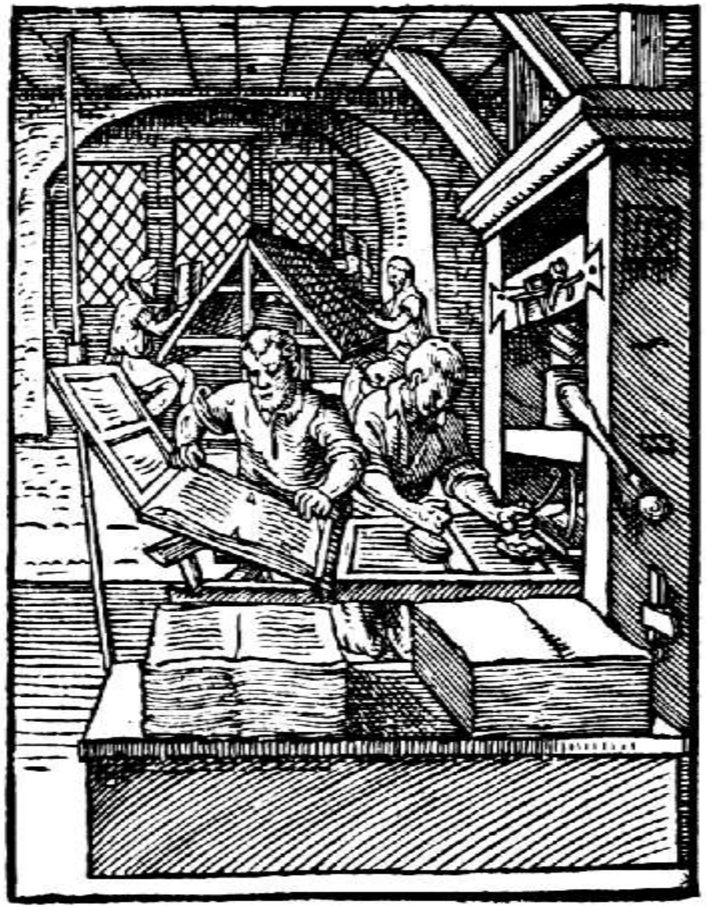
Continuous Delivery vs. Continuous Deployment

- Continuous *Deployment* (used with FRIB test environment)
  - Continuous Integration
  - Automatically deploy after each change
- Continuous *Delivery* (used with FRIB production environment)
  - 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)

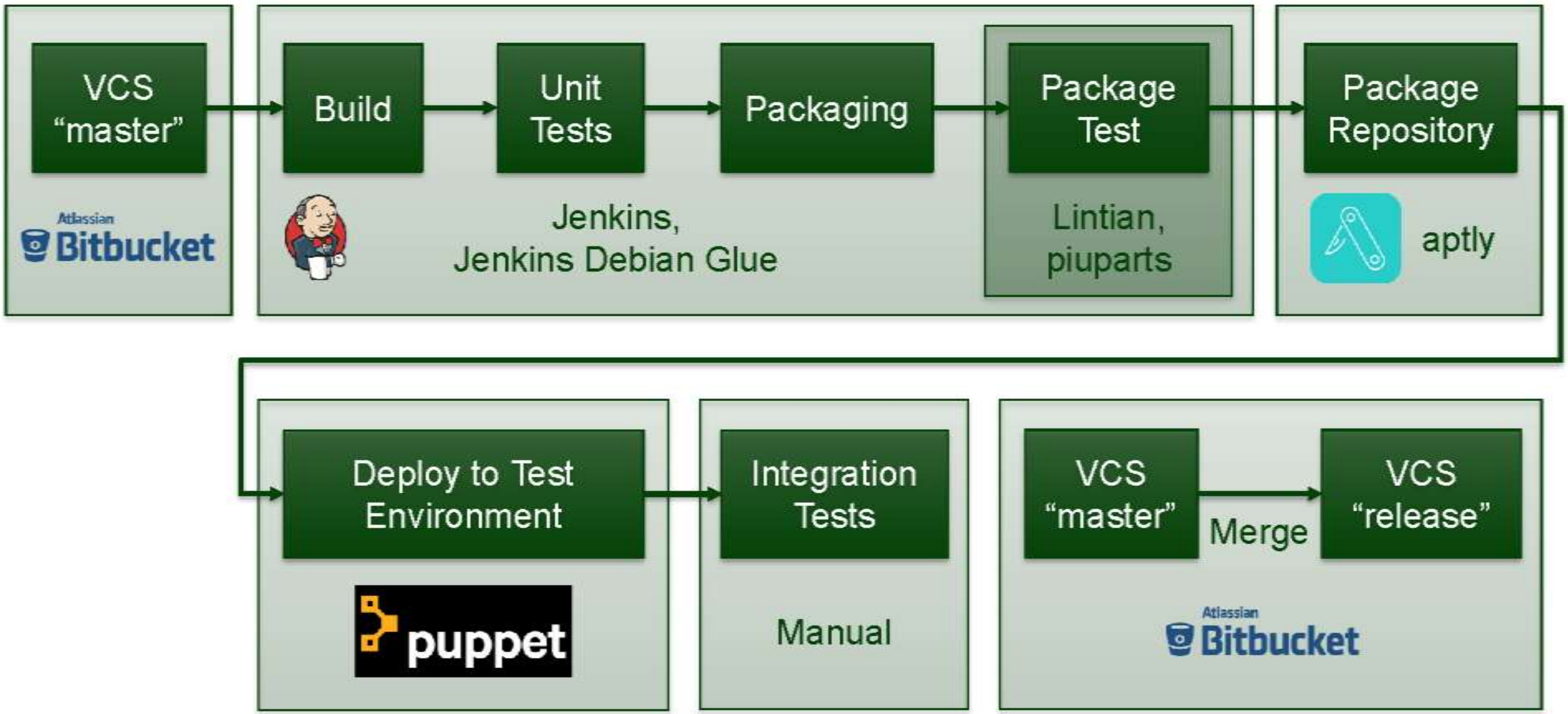
Why use Continuous Delivery?

- Overall we do not expect to save a significant amount of development time, but…
- Allows faster turn-around times
- More predictable (reproducible)
- Helps to catch issues before code is deployed to production system
- Full traceability
- Less risk of breaking something (we can always roll back)

➔ Facilitates team work

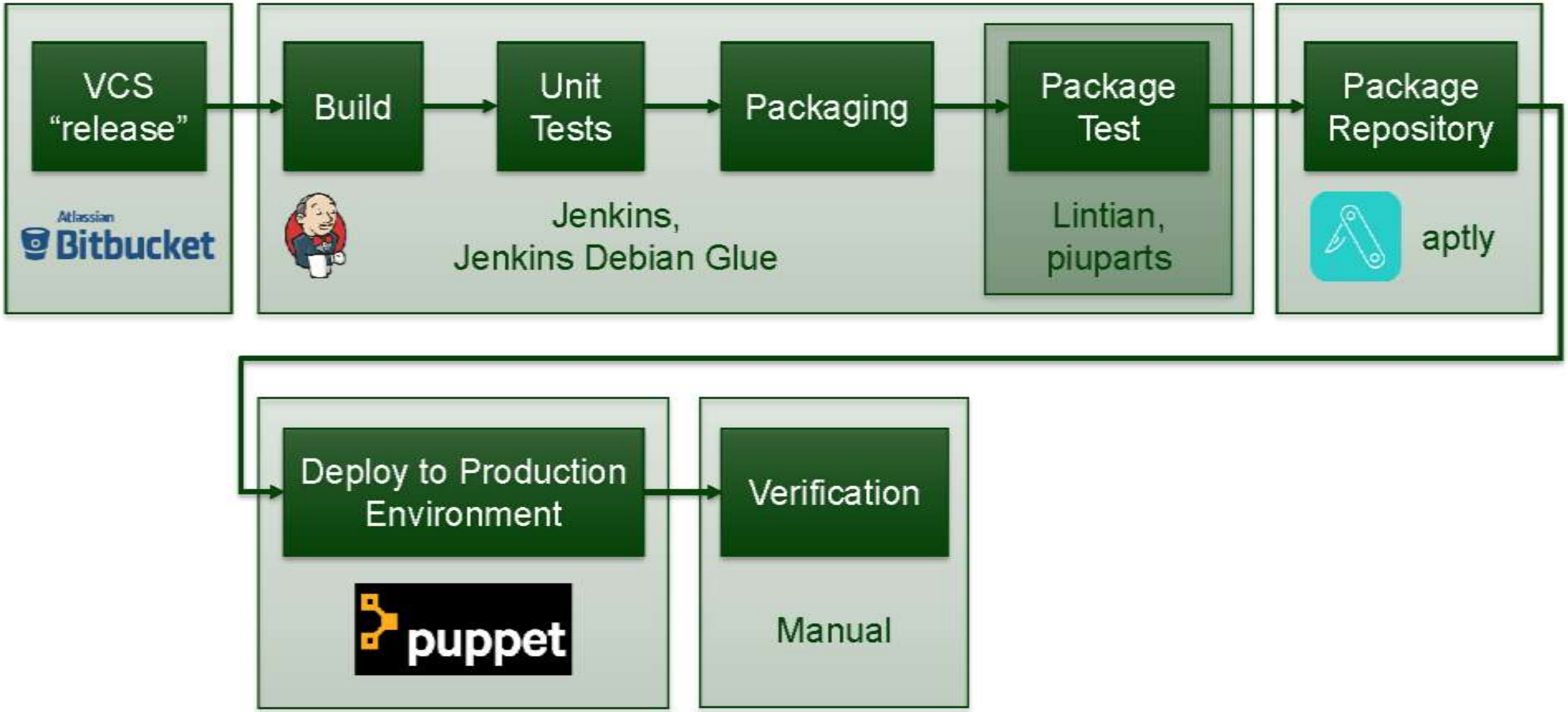


Continuous Delivery Pipeline for Test Environment



- Merge to "release" branch initiates deployment to production system

Continuous Delivery Pipeline for Production Environment



- Requires manual decision to deploy, but fully automatic from there

Deploying IOCs with Puppet: Motivation

- The FRIB approach
  - Deploy EPICS base and support modules as Debian packages
  - Build IOCs on the target machine
    - » Allows tweaking of IOC database in the production environment
- Challenges
  - Hundreds of IOCs, maintained by multiple engineers
    - » Consistency is important
  - Wide variety of IOCs require flexible deployment solution
  - Steps for setting up an FRIB IOC evolve over the years
  - Typical problems include
    - » New revision of IOC database gets pulled from Git repo but IOC maintainer forgets to restart IOC
    - » New version of support module gets deployed but IOC doesn't get rebuild
    - » Out of disk space due to missing logrotate configuration for procServ log files

EPICS Soft-IOC Puppet Module

- Features
  - IOC directory can come from any source
  - Automatically builds and restarts IOC if something has changed
  - Runs IOCs as a daemon with
    - » systemd
    - » System-V-style init scripts
  - Provides access to IOC shell via procServ
  - Supports multiple IOCs on the same machine
  - By default runs IOC process with limited user privileges
  - Rotates procServ log files
  - Lots of configuration options including
    - » Setting environment variables like EPICS_CA_MAX_ARRAY_BYTES
    - » Managing autosave directories
    - » CA security configuration

Example

```
$iocbase = '/epics/iocs'

package { 'epics-asyn-dev':
  ensure => latest,
}

class { 'epics_softioc':
  iocbase => $iocbase,
}

vcsrepo { "${iocbase}/vacuum-ioc":
  ensure   => latest,
  provider => git,
  source   => 'git://example.com/vacuum-ioc.git',
}

epics_softioc::ioc { 'vacuum-ioc':
  ensure    => running,
  enable    => true,
  bootdir   => 'iocBoot/iocvacuum',
  subscribe => [
    Package['epics-asyn-dev'],
    Vcsrepo["${iocbase}/vacuum-ioc"],
  ],
}
```

Install support packages

Ensure EPICS Base, procServ etc. are installed

Configure IOC process (use multiple of these sections to run multiple IOCs on the same machine)

Example [2]

- Use facility-wide defaults to reduce typing

```
Epics_softioc::Ioc {
  ensure     => running,
  enable     => true,
  log_server => 'logserver.example.com',
}

epics_softioc::ioc { 'vacuum-ioc':
  bootdir   => 'iocBoot/iocvacuum',
  subscribe => [
    Vcsrepo["${iocbase}/vacuum-ioc"],
    Package['epics-stream-dev'],
  ],
}
```

Experience

- Works very smoothly
- Saves quite some time when upgrading many IOCs at the same time
- For most use cases we rebuild and restart IOCs automatically after upgrading database files or support modules
  - Thus we always know that we are running the latest version
    - Avoids surprises when an IOC needs to be restarted later
  - It took a while until all engineers were comfortable with this behavior

Summary

- FRIB uses
  - Continuous Deployment with test environment
  - Continuous Delivery with production environment
- Libraries are being build as Debian packages on CI server
- IOCs are being build on the target machine
- EPICS Soft-IOC Puppet module automates deployment of IOCs
  - It's generic (no FRIB-specific functionality)
  - It's free software
    - » https://forge.puppet.com/mark0n/epics_softioc
    - » https://github.com/frib-high-level-controls/mark0n-epics_softioc