Stop Breaking The Trunk: A step by step guide to a CI Setup
Software Architect, Konrad Technologies
Chris Roebuck
Software Architect
Physicist
Team Working “Idealist”
LabVIEW Cheerleader

chris@konrad-technologies.co.uk
Continuous Integration / Deployment

**Continuous Integration (CI)** is a development practice that requires developers to **integrate** code into a shared code repository frequently. Each check-in or commit is then verified by automated tests (linting, unit tests) and production build artefacts.

**Continuous Deployment (CD)** is a similar methodology except that code that passes the automated testing is automatically deployed to a live production environment.
Constituent Parts of a CI Setup

1. Check In Changes
2. Fetch Changes
3. Build
4. Test
5. Fail or Succeed
6. Notify Success or Failure
Source Code Control

Source Code Control (SCC) is a version control system designed to track changes in source files during the development of a piece of software.
Version Control Systems

Local Version Control
- Checkout
- File
- Version Database
  - Version 3
  - Version 2
  - Version 1

Examples: RCS

Centralized Version Control
- Computer A
  - File
- Central VCS Server
  - Version Database
    - Version 3
    - Version 2
    - Version 1
- Computer B
  - File

Examples: CVS, Subversion, VSS
Distributed Version Control

- Developer makes changes
- Developer adds new files
- Developer Commits locally
- Developer pushes to server

Examples: Git, Hg
Multiples Developers & Binary Files

Hatching a catastrophe?

- Binary Files are a pain
- LVLIBs can cause sleepless nights
- Feature branches and good communication are your friend
- Get a highly visual client
- Don’t panic!
Many clients and hosting solutions

- Bitbucket
- GitHub
- GitLab
- SourceTree
- TortoiseSVN
- Azure DevOps
Hook Scripts

- Triggers to perform additional actions
  - Pre-Commit
  - On Commit
  - On Push
  - On Pull

- Subversion allows server-side hooks
- Git allows client or server-side hooks
Build Tools
Chris’ CI Setup – Client Side
Commit Hook

### PRE-COMMIT.sh

```bash
#!/bin/sh

cmd.exe /C C:\Users\christopherroe\buck\Documents\CI_Main.vi -- C:\Users\christopherroe\buck\Documents\test" "TRUE" "FALSE" "TRUE"

"C:\Users\christopherroe\buck\Documents\test\Demo.lvproj" "C:\Users\christopherroe\buck\Documents\test\Demo.cfg" "My Application"
```

### On-Commit.bat

```bash
labview-cli
```

Server Computer

Version Database

Version 3

Version 2

Version 1

Computer A

File

Version Database

Version 3

Version 2

Version 1

Computer B

File

Version Database

Version 3

Version 2

Version 1

On-Commit.bat

"C:\Users\christopherroe\buck\Documents\CI_Main.vi" -- C:\Users\christopherroe\buck\Documents\test" "TRUE" "FALSE" "TRUE"

"C:\Users\christopherroe\buck\Documents\test\Demo.lvproj" "C:\Users\christopherroe\buck\Documents\test\Demo.cfg" "My Application"
CLI Helper

LabVIEW Settings
- Path to LabVIEW project file
- Path to VI Analyzer configuration
- Build Specification
- Path to VI Package Build Specification

Options
- Run VI Analyzer Tests
- Run VI Tester Tests
- Execute Build Specification
- Build VI Package

Start
Exit
Command Line Interface

- Build EXE or Package
- VI Analyzer
- VI Tester
Testing

Static Code Analysis – VI Analyzer

Functional Analysis – VI Tester
Building

Build Specification – EXE

Build Specification - VIP
Continuous Integration

LabVIEW Integration & Local Repo Hooks
Chris’ CI Setup - Building and Packaging
Bamboo Build Agent

OnPush.bat

labview-cli
"C:\Users\christopherroebuck\Documents\ci-tool\CI_Main.vi" --
"C:\Users\christopherroebuck\Documents\test" "TRUE" "FALSE" "TRUE" "C:\Users\christopherroebuck\Documents\test\Build Demo.lvproj"
"C:\Users\christopherroebuck\Documents\test\Demo.cfg" "My Application"
Command Line Interface

VI Analyzer

VI Tester

Build EXE or Package
Continuous Integration

Atlassian Bamboo Build Server & LabVIEW
The tools I used
Questions?

chris@konrad-technologies.co.uk