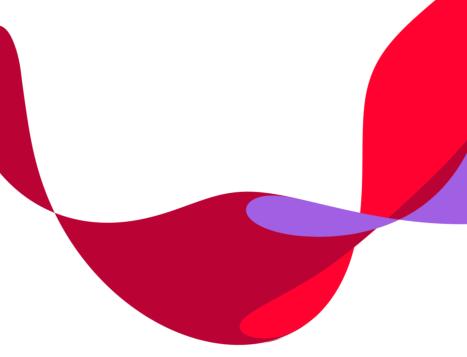
Zero to hero: bootstrapping control system development at a new facility



Jure Varlec

Senior Developer / Tech Lead

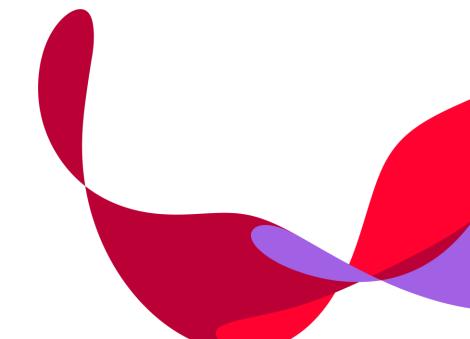
jure.varlec@cosylab.com





We adapt to your EPICS environment ...

... but that's not always possible.





Sometimes, there is no existing EPICS environment

- A brand-new facility, or a full upgrade of an existing one.
- No EPICS build and deployment tools.
- No controls team, or no EPICS experience.
- No controls or device network.
- No git repositories.

In time, these will all be set up. But the facility can't afford to wait until then!





Aims

- Smooth transition from nothing to full infrastructure.
- Easy onboarding of Cosylab developers.
 - Managing variable workload of long projects.
 - Also helps with facility staff training.
- Clear version and release management.
 - Early deployment will be running a mixture of development versions.
- High assurance that software working in development will also work when deployed.

Means

- Vanilla EPICS build system.
 - Mostly standard module structure.
 - Everyone knows this.
- Heavy reliance on git describe and tags.
 - To discern between released versions and development versions.
- Containers for development and deployment.
- Testing with the deployment environment.
- Support deployment without containers.
 - There are cases when they don't make sense.



A big giant ball of tar

- The main deliverable is a single tarball:
 - Source code
 - Documentation
 - Build scripts
- Versioned using git describe.
 - Additionally, all custom modules are themselves versioned this way.
 - No semantic versioning, release notes contain everything staff needs to know.
- Contains entire git repos of modules.
 - Everything is delivered, no bus factor.
 - Straightforward to transition to git later on.

epics-environment-v4-14-gdb9df8c.tar.gz base modules asyn autosave CONFIG_SITE.local RELEASE.local build.sh Containerfile README.md RELEASE NOTES.md



Build results

- Containerfile builds a development image and a deployment image.
- RELEASE.local and build.sh are generated.
- CONFIG_SITE.local sets install locations for documentation and GUIs.

```
/opt/<facility>
```

```
— epics
```

```
base
```

```
— modules
```

```
module_docs
```

- ├── eng_gui
- └── plc_code

epics-environment-v4-14-gdb9df8c.tar.gz

- base
- modules
 - asyn
 - autosave
 - ----
 - ├── CONFIG_SITE.local
 - └── RELEASE.local
- ├── build.sh
- ├── Containerfile
- README.md
- L___ RELEASE_NOTES.md



Structure of a module

- Support library with everything important
 - Databases
 - Support code
 - Engineering GUIs
 - Documentation
 - PLC code
 - iocsh scripts
- Sample app: developer's tool
- Docs, GUIs, and PLC code are maintained together with EPICS code.
 - Installed using FILE_TYPE mechanism.

MyFancyModule

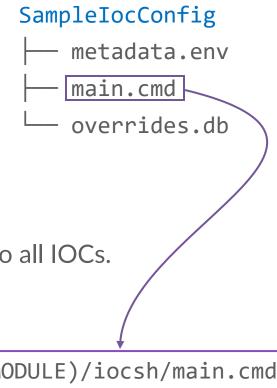
- -- configure
- -- MyFancyModuleSup
 - -- Db
 - -- src
 - -- doc
 - |-- gui
 - |-- plc
 - `-- iocsh
- -- MyFancyModuleSampleApp
 - -- src
- -- iocBoot
 - -- iocMyFancyModuleSample



Deployment

- GenericIoc: one app linking (nearly) all modules.
- IOC configuration: a directory with startup files.
 - No compiling to instantiate an IOC.
 - Amenable to templating.
- Starting an IOC from the config directory:
 - iocRunner GenericIoc
 - podman run <opts> <image> iocRunner GenericIoc
- iocRunner sets up the environment and things common to all IOCs.
- IOCs managed by systemd and procServ.

iocshLoad("\$(MYFANCYMODULE)/iocsh/main.cmd", "<MACROS>")
dbLoadRecords("./overrides.db", "<MACROS>")





Summary

- Developing a control system from nothing to everything is an exercise in change management.
 - Things should start simple so as **not to overwhelm** facility staff.
 - As available infrastructure grows, **processes need to adapt**.
- Start by delivering **tarballs**, transition to git **repositories**.
- **Containers** allows running IOCs **manually** in the beginning, transitioning to **service management**.
- Developing EPICS code, PLC code, GUIs, and documentation **in the same repository** reduces headaches considerably.
- **Continuous integration** allows using the same artifacts for testing and deployment.
 - Run CI at Cosylab first, set it up at the facility later.
- Release management should be such that **releases are just a formality**.



Thank you.

Jure Varlec

jure.varlec@cosylab.com



