EPICS Development and deployment at ESS

Timo Korhonen
ICS Chief Engineer
European Spallation Source ERIC
Date

Overview

- Introduction
- Context of ESS
  - The project, organisation, collaborations
- What would we like to achieve, and why?
  - Project duration, facility lifetime considerations
- **e**SS **e**PICS **e**nvironment, aka **e3**
  - Basic concepts and structures
- Deployment of IOCs
  - From code to functional controllers
- Summary

Introduction

- In the 2019 deployment meeting @ITER, I presented slides from Jeong Han Lee
  - I did not do a good job – I think the core message was lost
    - Those slides contain a lot of good material, I recommend to take a look
  - This time I concentrate on the concepts only – details to be discussed separately
- A lot of things has happened since
  - We have forked even more from the PSI implementation
    - Dropped some architectures that we had no chance of supporting or understanding (notably VxWorks)
    - “require” developed further, version numbering has been revised, etc.
  - e3 has been augmented with a deployment tool
    - Not a subject of this talk, but a crucial (and very nice) tool;
  - There is also a “help desk” for support and requests
  - We have first commissioning rounds behind us
    - over 500 e3-based IOCs deployed on over 200 hosts, number is growing.
- The system has matured, but there is still a lot of development going on

The ESS Context

- ESS is a pretty large project
  - 12 member countries, cost to completion ~3000 MEUR (1800+800+500)
  - Green field, starting from the ground (no existing infrastructure, institute, facility,…)
- Building the organization from scratch has been a long and winding road.
  - New people, many first time in an accelerator or scientific project
  - The organisation has also been a moving target – new things come up on the way
  - Growth has been fast
    - Hard to keep consistency and coherence in working practices
    - Skills are very different – different backgrounds
  - We have to continuously adapt to new things
    - EPICS developments – new features
    - Changing environment in the project / institution

High-level goals

Oversimplified, please bear with me.

- Support on-boarding of new people, under time pressure
  - Standardization of work practices
    - To enable sharing of developments and mobility of people
  - Simplification of IOC composition
    - Provide a common baseline to get quickly started
  - Separation of concerns: control system integration vs. software engineering
    - To enable people to concentrate on the tasks at hand and utilize the skill that they bring in
- Adapt to the changes in the surrounding environment without compromising quality
  - Agility
    - Keep up with the developments in EPICS (7), in IOC and on the service layer
    - Frequent small updates instead of a “big bang”
    - “no IOC left behind” policy – new features (e.g., Channel Finder enhancements) easy to add
  - Quality control, system-wide
    - CI/CD builds, known sources of modules, dedicated support team

Central concepts of e3

- The environment is centrally provided for all (production) systems/IOCs
  - EPICS Core release, in multiple (but limited number of) versions
  - E3 Core team is responsible for composition and provision
    - Gatekeeper of what modules are installed in production
- IOC is configured start-up time
  - No code compilation for an individual IOC
    - (*when did **you** last compile the Linux kernel to add or update drivers?* 😉)
  - Modules (asyn, StreamDevice, Area Detector,…)are loaded with a require *module name* [version] in startup script
    - Further configuration done with startup snippets
    - And of course in database templates and substitutions
- IOCs become “codeless” – startup scripts plus database configuration
- Module updates (in an individual IOC) do not require compilation
  - “service” modules can be updated without asking the IOC developer to re-compile
    - IOC restart and simple functionality test is sufficient

Implementation

A selection of details – nowhere near a complete description

- EPICS resources are provided as a combination of
  - Core release
  - A selection of modules, possibly in different versions
  - Method for loading modules with module dependencies pre-organized
    - “require <module>” command in startup script
  - Modules and libraries in a tree structure, pre-compiled
- Production versions provided in a shared file system tree
    - Typically NFS (CEPH in consideration for the future.)
    - Locally built trees are also possible but not allowed/supported in production
- More reading: introduction, download links and install instructions: http://e3.pages.esss.lu.se/index.html
  - Should build without issues on fairly recent Linux distributions (Red Hat, Debian)

User (IOC developer) side

- Figure out what the IOC needs to do, decide the platform
  - aka “system requirements”
  - Embedded/MTCA, PC-based, fully virtualized
- Select appropriate modules for the task from the e3 module list
  - If no appropriate module has been provided:
    - Ask the e3 support team to add one from the community, or
    - Develop your own; ask e3 team to integrate when it is ready for production
    - For testing of new combinations in the field, a “sandbox” method is provided (“cell mode”)
  - No scourging of modules or drivers from the network!
- Configure the device support and drivers
- Write the database templates to implement the functionality
- Test, and deploy
  - …and in parallel, write a lot of documentation required by ESS
    - Well, it is not that much after all IMHO…the next guy who takes over the maintenance will appreciate.

Pros and cons

A selection

+ IOC developers do not have to worry about code details
  + Unless they want to, or are developing new code
+ Deployed software is easier to manage (uniformity)
+ Updates are easy (there is always some pain, but frequent updates keep the pain small.)
+ System is (relatively) easy to keep up to date and add new services
- A dedicated team required to provide the resources
  - This is both a pro and con: experts are not easy to find (or to educate), OTOH the experts can concentrate on what they can the best. (we have also quite active user participation.)
- Customization needed for module integration
  - Dependency handling, patching (which can be a pro as well)
- Loneliness… 😢
  - Most sites still prefer to recompile each (type of) IOC…

Summary

- e3 tries to address the goals and challenges of ESS
- Development has taken some time, even when we started from the example of PSI
- It was not straightforward to get buy-in from IOC developers
  - Organisational goals are often not seen or recognized by the individual developers
  - There were several turns in development, sometimes contradictory
- We have now reached a stable state
  - Manifested by the stability in operation, developer buy-in and drastic reduction of issues and complaints
    - It is almost too silent…
- Not all work is done however.
  - We would like to reduce or remove the need to create e3-specific wrappers and just use the EPICS build system. (seems feasible even if not straightforward; details need still to be worked out)
  - Module (re)organisation is under consideration
    - Dilemma: how to deal with modules that provide generic functionality vs. those for specific device or application integration.
    - There are some promising ideas brewing.

Acknowledgements

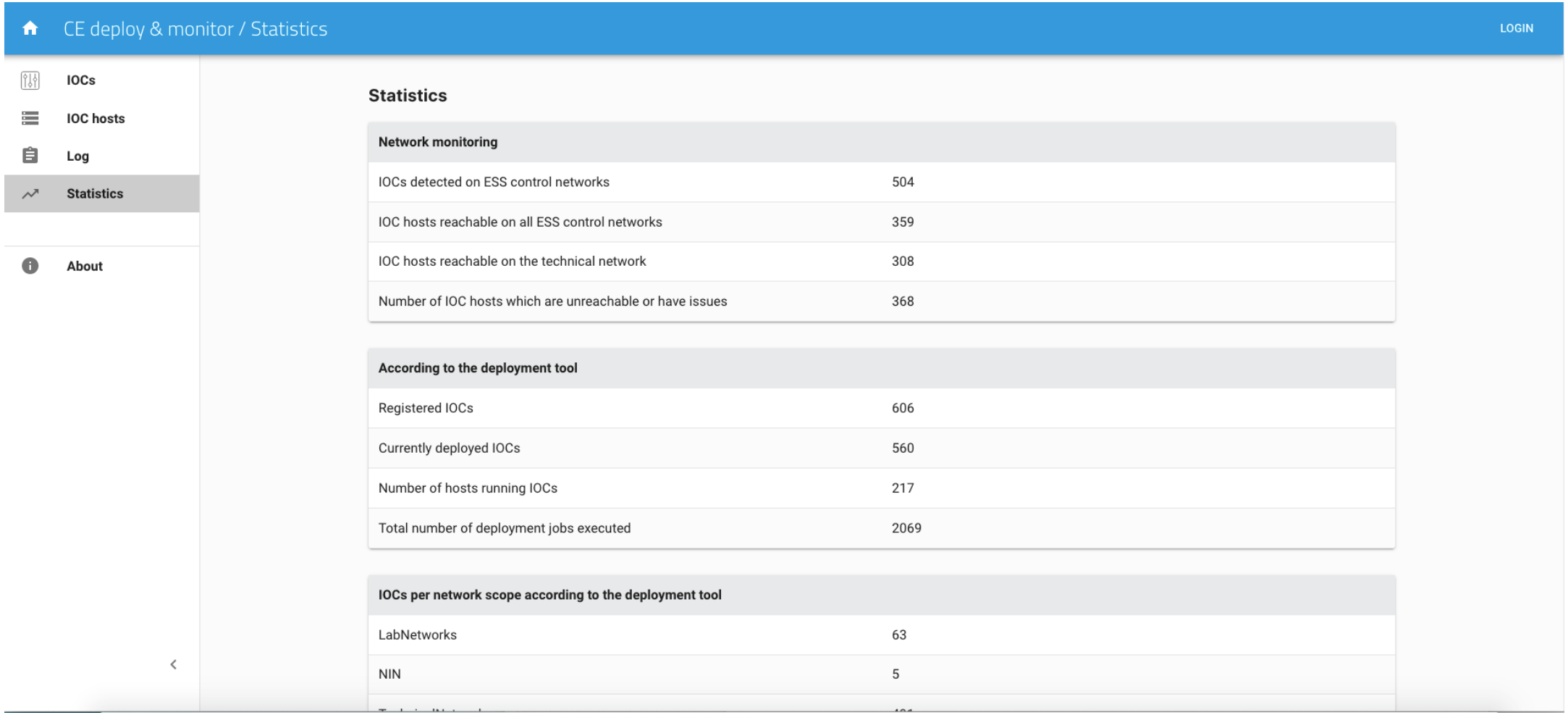
- The ESS e3 team core members during development:
  - Simon Rose, Anders Lindh Ohlsson, Krisztian Löki, Wayne Lewis (Osprey DCS)
- Alumni:
  - Jeong Han Lee (aka Han)
  - Klemen Strnisa, Niklas Claesson (Cosylab), first EEE, modelled after PSI and later became e3
  - Benjamin Bertrand, supporting EEE deployment (now at MaxIV)
- Father of the concept and implementer of the PSI version
  - Dirk Zimoch (PSI)
  - plus other colleagues at PSI

- Without the effort and skills of all these people we would not be where we are.

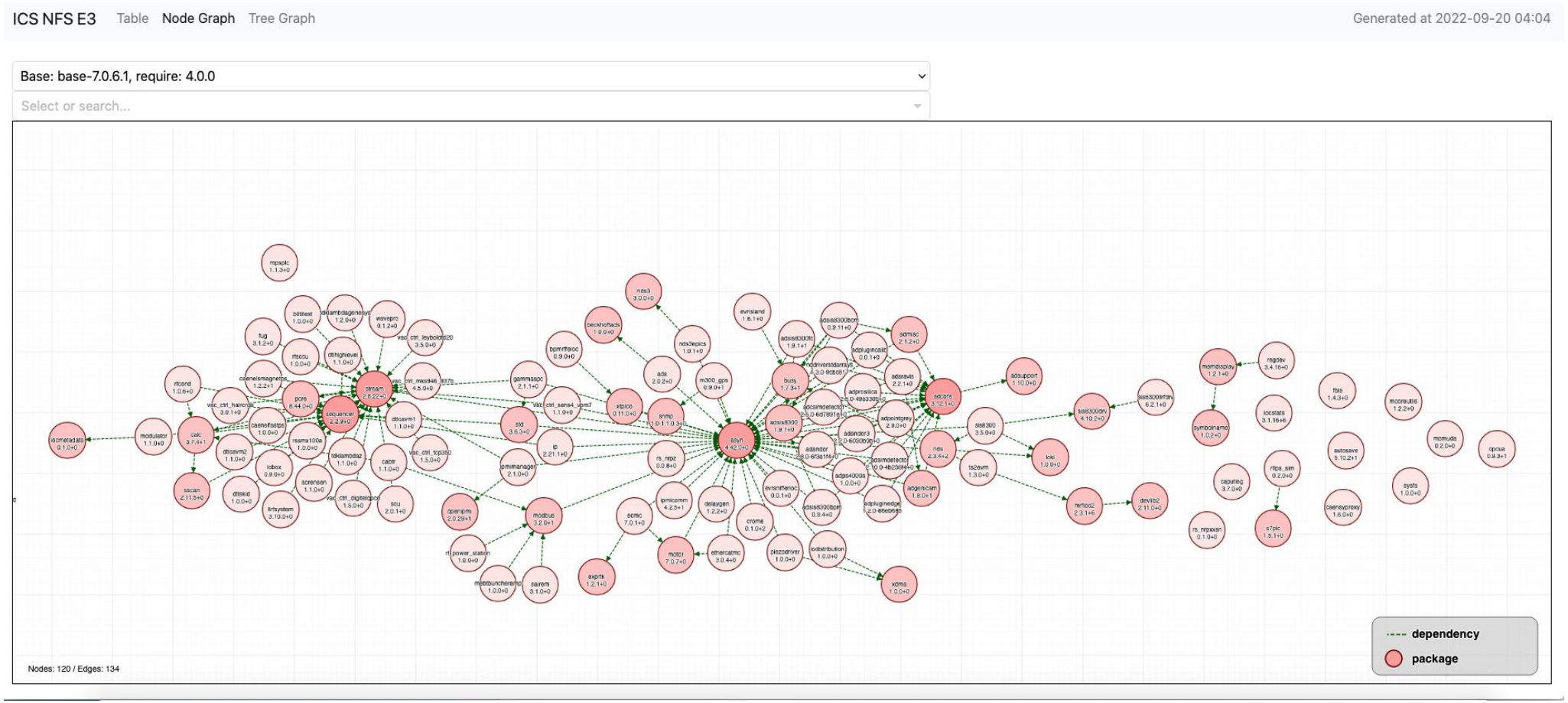
Additional material

Deployment status as of today

(we are in a shutdown + installation period, many IOCs offline)



Installed modules and their dependencies
(for each available version)



Installed modules and their dependencies
(table view, including build pipeline status)