Run control status and plans

Pierre Lasorak

Summary

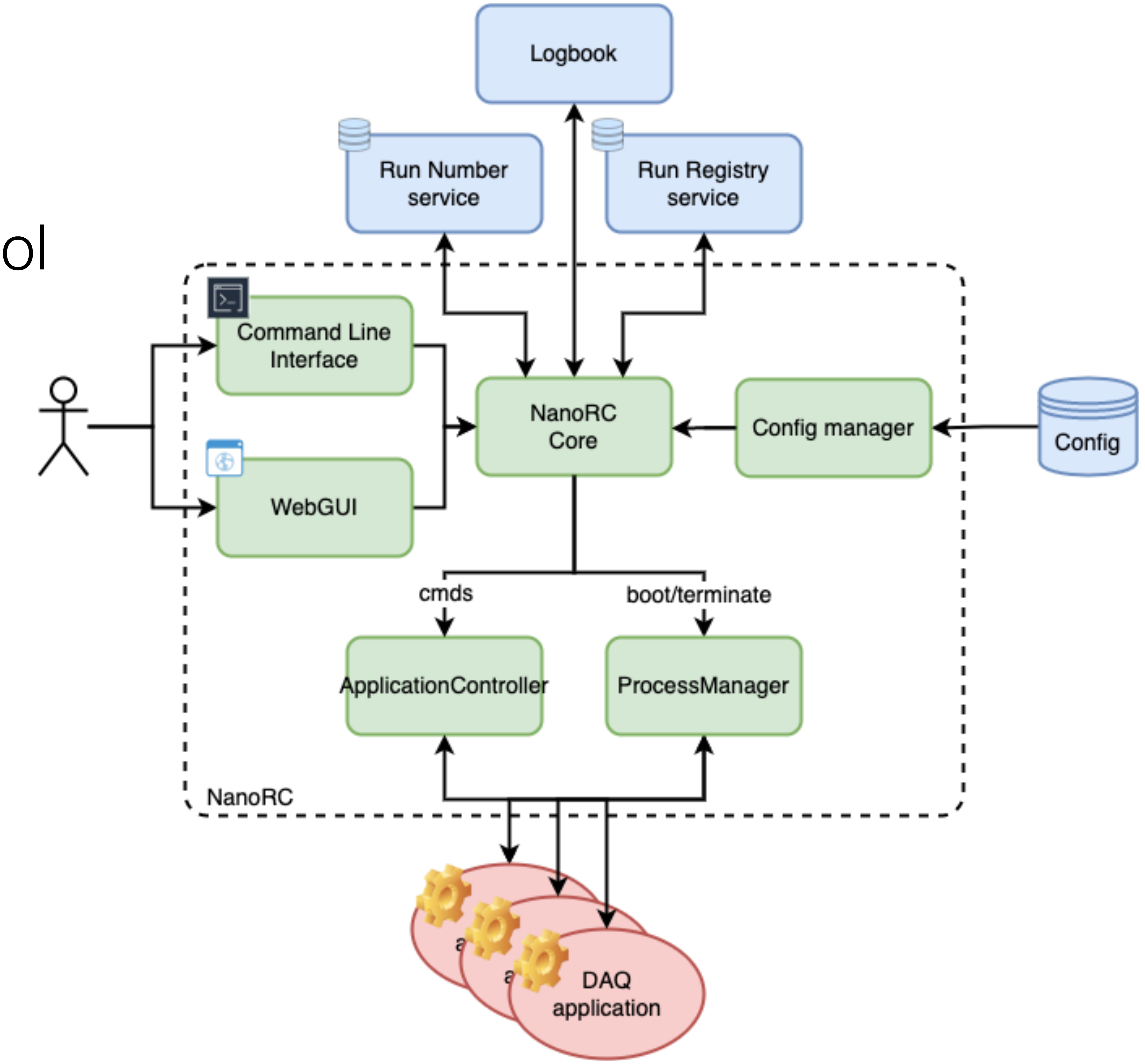
- Overview of NanoRC
- Future of the run control
- Supervisor

NanoRC

- The prototype run control is used to control the DAQ
- In operation at EHN1 (ProtoDUNE-VD, ProtoDUNE-HD and the 2 coldboxes), for TOAD test stand and ICEBERG
- Enable interactions with DAQ applications
- Interact with external DAQ services
- Testing ground and learning platform for some CCM functionality

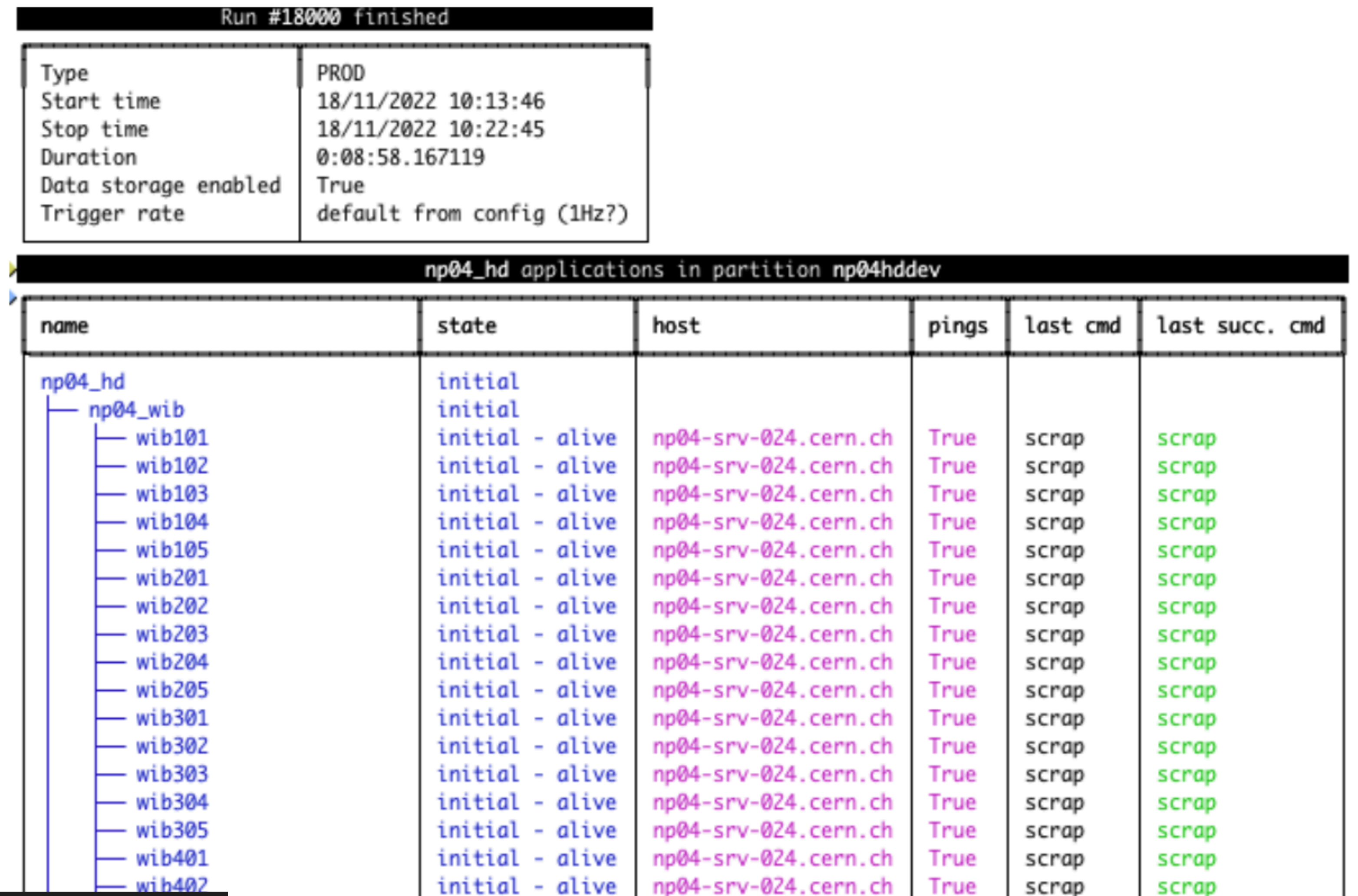
NanoRC architecture

- Written in Python
- "Tries" to implement the division of labour that will be used in the real control system
  - User interactions
  - Process management
  - DAQ applications & services interactions

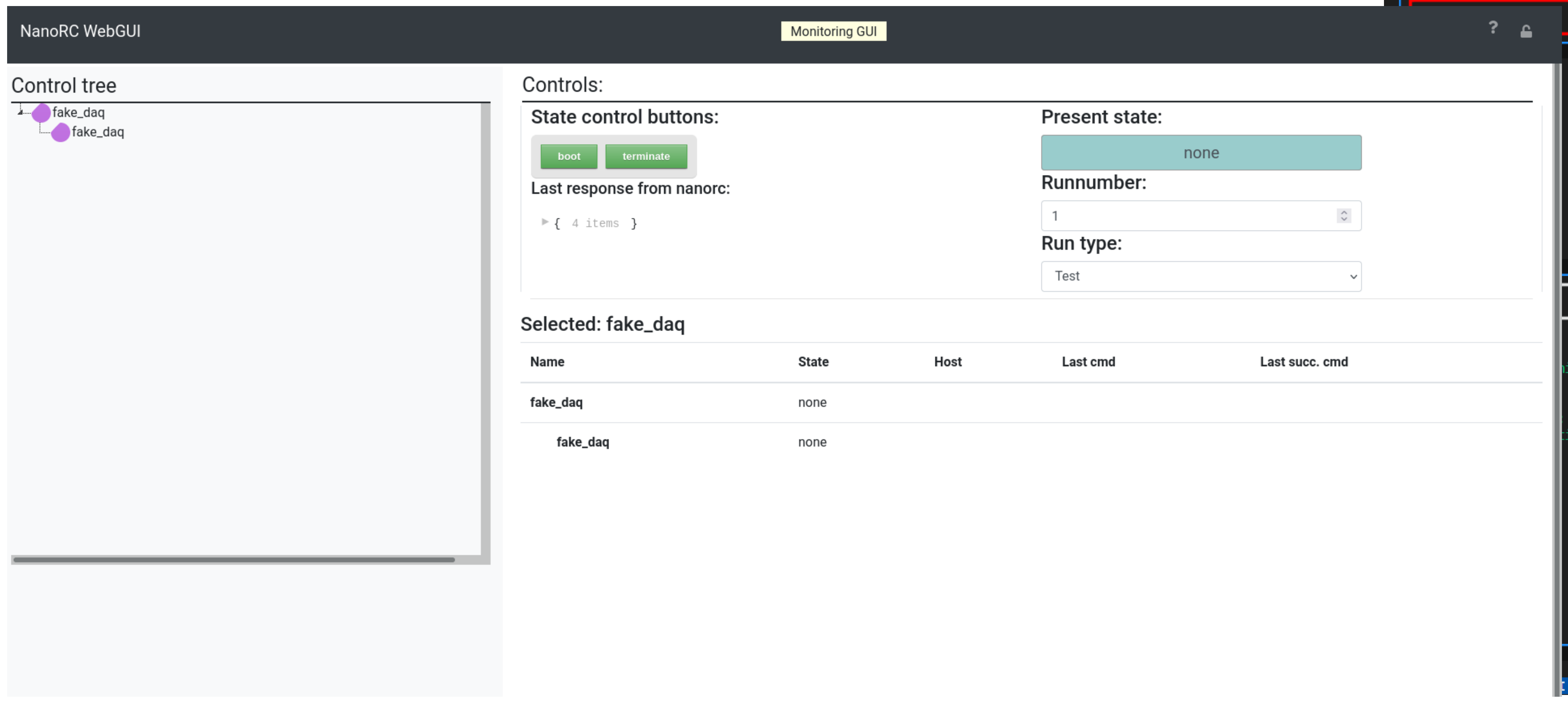


NanoRC
User interactions

- 3 ways to interact with the run control
  - Command line (shell-like): default
  - Textual user interface (click-in-terminal)
  - Graphical user interface (browser-based)



```
                Run #18000 finished
┌──────────────────────┬────────────────────────────┐
│ Type                 │ PROD                       │
│ Start time           │ 18/11/2022 10:13:46        │
│ Stop time            │ 18/11/2022 10:22:45        │
│ Duration             │ 0:08:58.167119             │
│ Data storage enabled │ True                       │
│ Trigger rate         │ default from config (1Hz?) │
└──────────────────────┴────────────────────────────┘
                              np04_hd applications in partition np04hddev
┌────────────────────────────┬─────────────────┬─────────────────────┬───────┬──────────┬────────────────┐
│ name                       │ state           │ host                │ pings │ last cmd │ last succ. cmd │
├────────────────────────────┼─────────────────┼─────────────────────┼───────┼──────────┼────────────────┤
│ np04_hd                    │ initial         │                     │       │          │                │
│ └── np04_wib               │ initial         │                     │       │          │                │
│     ├── wib101             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib102             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib103             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib104             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib105             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib201             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib202             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib203             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib204             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib205             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib301             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib302             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib303             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib304             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib305             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib401             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│     ├── wib402             │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-024.cern.ch│ True  │ scrap    │ scrap          │
│                          d │ initial         │                     │       │          │                │
│          np04srv028card0   │ initial - alive │ np04-srv-028        │ True  │ scrap    │ scrap          │
│          np04srv028card1   │ initial - alive │ np04-srv-028        │ True  │ scrap    │ scrap          │
│          np04srv029card0   │ initial - alive │ np04-srv-029        │ True  │ scrap    │ scrap          │
│          np04srv029card1   │ initial - alive │ np04-srv-029        │ True  │ scrap    │ scrap          │
│                            │ initial         │                     │       │          │                │
│          0                 │ initial - alive │ np04-srv-002.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│          4srv0280          │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│          4srv0281          │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│          4srv0290          │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│          4srv0291          │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
│          v0280             │ initial - alive │ np04-srv-028        │ True  │ scrap    │ scrap          │
│          v0281             │ initial - alive │ np04-srv-028        │ True  │ scrap    │ scrap          │
│          v0290             │ initial - alive │ np04-srv-029        │ True  │ scrap    │ scrap          │
│          v0291             │ initial - alive │ np04-srv-029        │ True  │ scrap    │ scrap          │
│                            │ initial - alive │ np04-srv-018.cern.ch│ True  │ scrap    │ scrap          │
└────────────────────────────┴─────────────────┴─────────────────────┴───────┴──────────┴────────────────┘
```



Control tree

- fake_daq
  - fake_daq

Controls:

State control buttons:

Last response from nanorc:

▸ { 4 items }

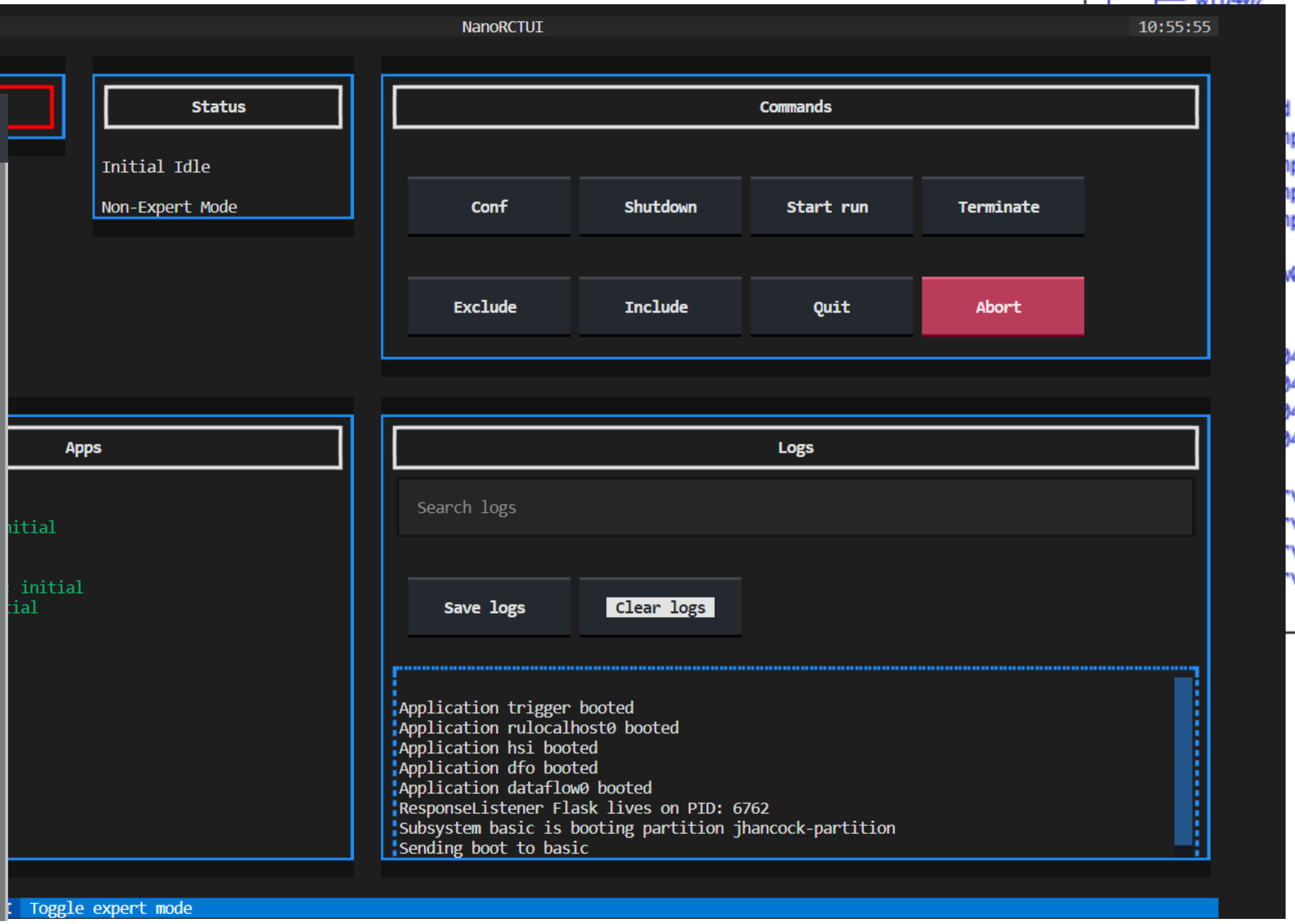
Present state: none

Runnumber: 1

Run type: Test

Selected: fake_daq

Name	State	Host	Last cmd	Last succ. cmd
fake_daq	none			
fake_daq	none			



Initial Idle

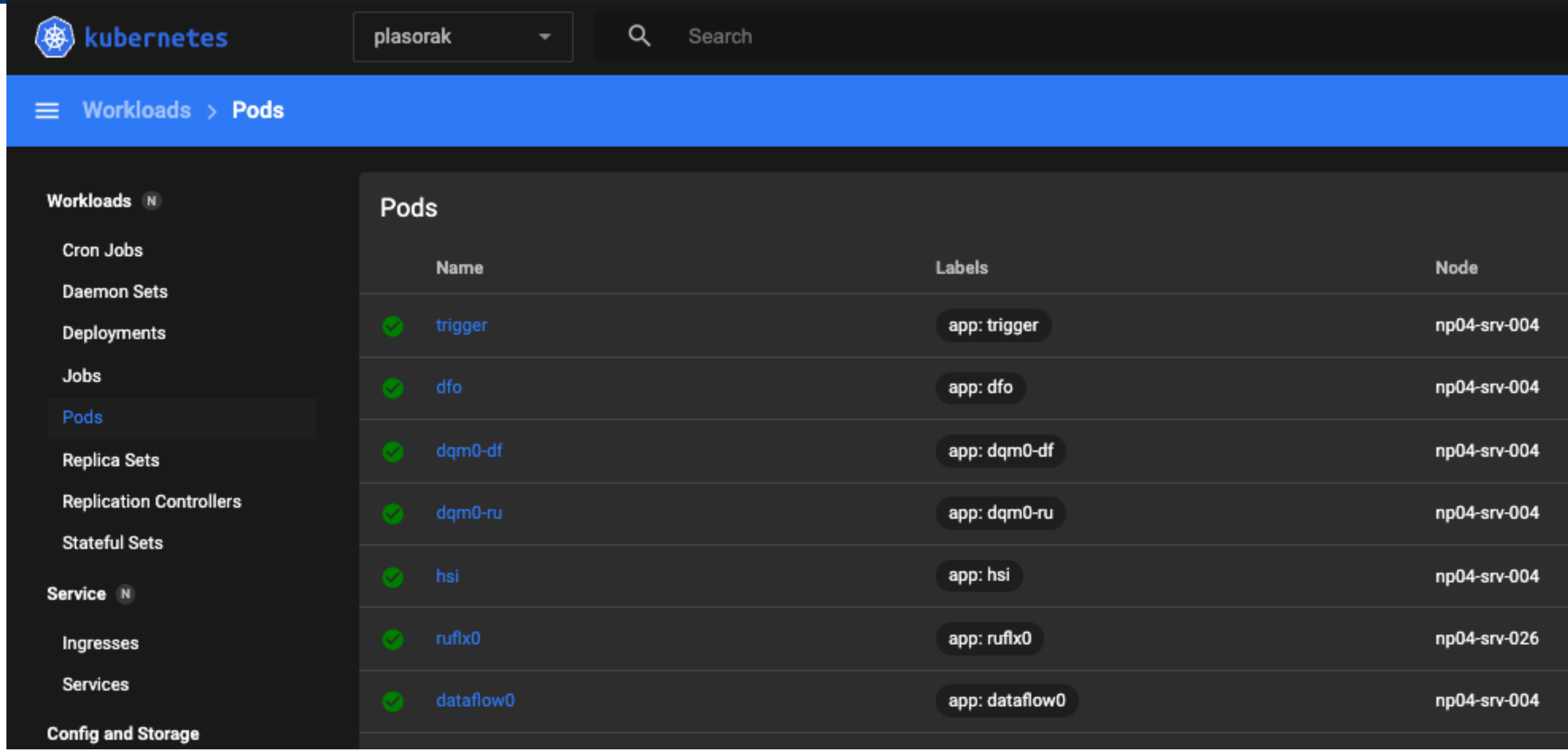
Non-Expert Mode

```
Application trigger booted
Application rulocalhost0 booted
Application hsi booted
Application dfo booted
Application dataflow0 booted
ResponseListener Flask lives on PID: 6762
Subsystem basic is booting partition jhancock-partition
Sending boot to basic
```

GUI, from Zbynek Kral   TUI, from Jonathan Hancock

NanoRC
Process management

- NanoRC has 2 process managers
  - SSH-based
    - SSH on a host where the application should run and execute the daq_application binary
  - Kubernetes-based
    - Container orchestration tool from Google, used in data centres
    - Advantages: flexibility, reliability, use of containers and extensive configurability

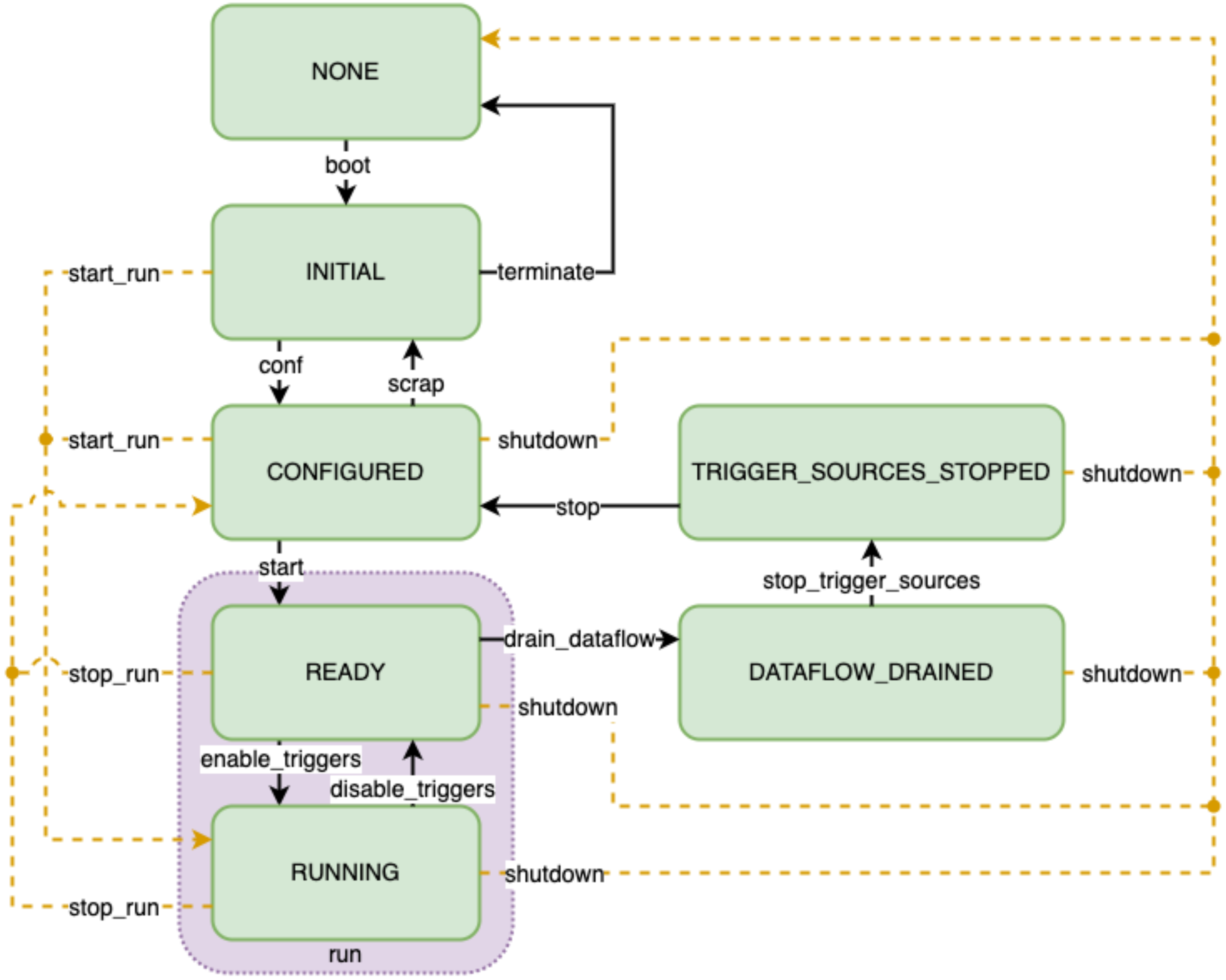


Pods

	Name	Labels	Node
✓	trigger	app: trigger	np04-srv-004
✓	dfo	app: dfo	np04-srv-004
✓	dqm0-df	app: dqm0-df	np04-srv-004
✓	dqm0-ru	app: dqm0-ru	np04-srv-004
✓	hsi	app: hsi	np04-srv-004
✓	ruflx0	app: ruflx0	np04-srv-026
✓	dataflow0	app: dataflow0	np04-srv-004

NanoRC
Application interactions

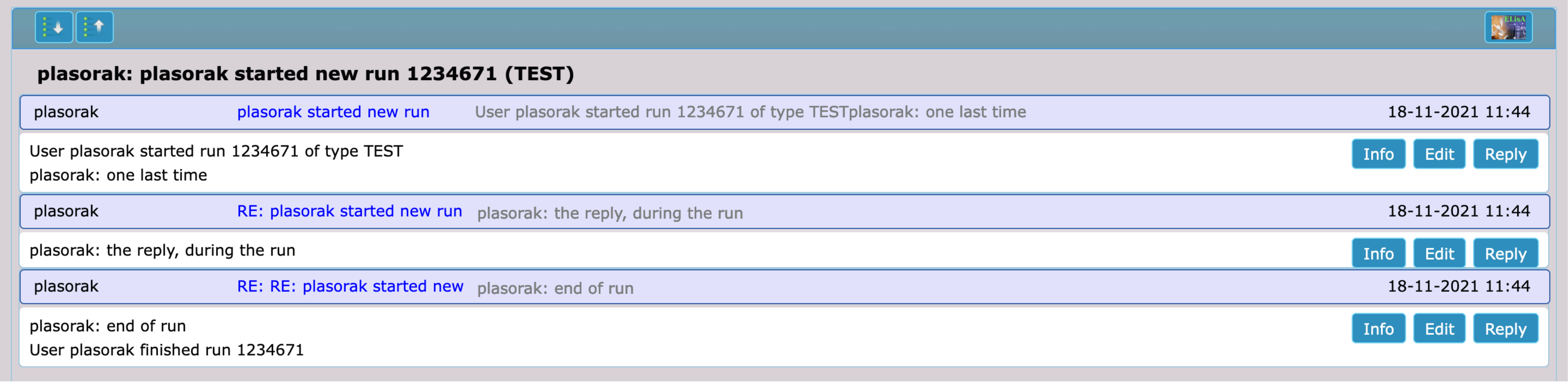
- DAQ Applications are REST-APIs
  - “HTTP servers” that respond to POST commands containing JSON data
    - Predefined schema
  - DAQ applications execute the FSM transitions described in the JSON data
  - DAQ applications acknowledge and reply to the run control after executing the transition



Finite state machine for DUNE-DAQ applications

NanoRC
Microservices interactions

- DAQ microservices provide, at EHN1
  - Run numbers (continuously increasing number started after the ProtoDUNE 1 runs number)
  - Run configuration archiving
  - Electronic logbook
- NanoRC interacts with each of these microservices via interfaces at EHN1 (production nanorc)
- Working on packaging these microservices for all the test stand to integrate them



plasorak: plasorak started new run 1234671 (TEST)

plasorak   plasorak started new run   User plasorak started run 1234671 of type TESTplasorak: one last time   18-11-2021 11:44

User plasorak started run 1234671 of type TEST
plasorak: one last time

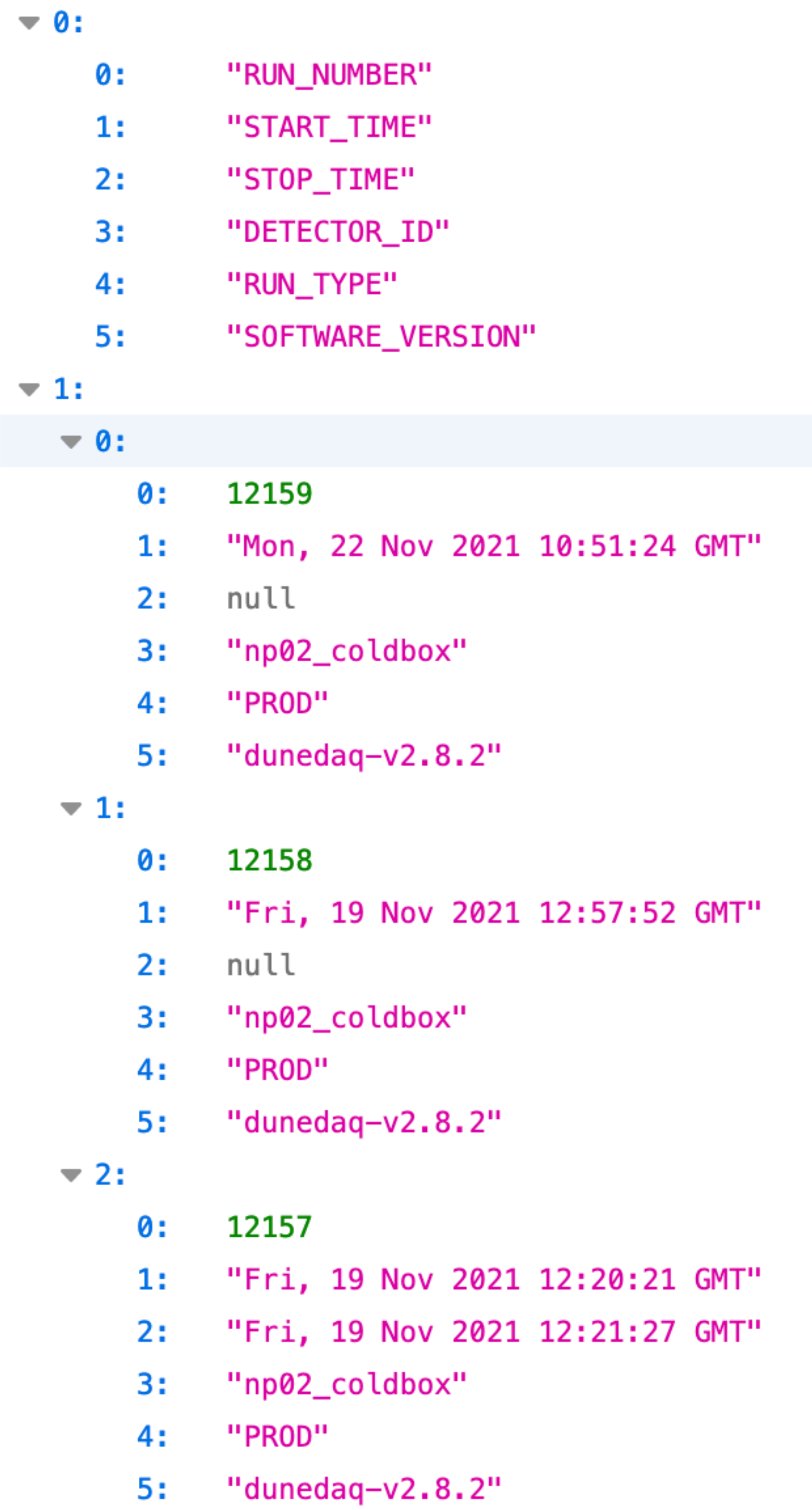
plasorak   RE: plasorak started new run   plasorak: the reply, during the run   18-11-2021 11:44

plasorak: the reply, during the run

plasorak   RE: RE: plasorak started new   plasorak: end of run   18-11-2021 11:44

plasorak: end of run
User plasorak finished run 1234671

CERN ELisA logbook facility

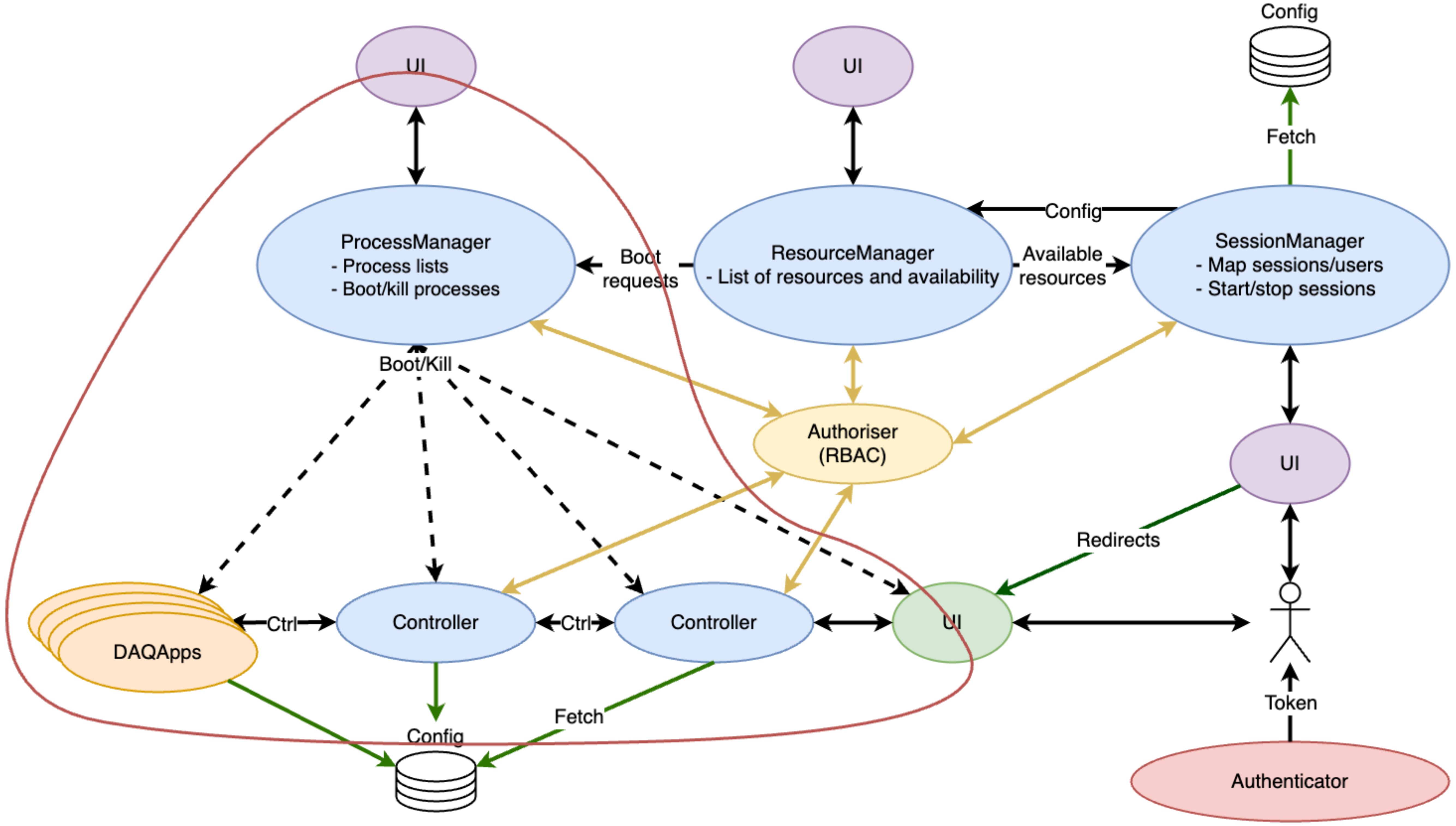


```
▼ 0:
    0:    "RUN_NUMBER"
    1:    "START_TIME"
    2:    "STOP_TIME"
    3:    "DETECTOR_ID"
    4:    "RUN_TYPE"
    5:    "SOFTWARE_VERSION"
▼ 1:
  ▼ 0:
      0:  12159
      1:  "Mon, 22 Nov 2021 10:51:24 GMT"
      2:  null
      3:  "np02_coldbox"
      4:  "PROD"
      5:  "dunedaq-v2.8.2"
  ▼ 1:
      0:  12158
      1:  "Fri, 19 Nov 2021 12:57:52 GMT"
      2:  null
      3:  "np02_coldbox"
      4:  "PROD"
      5:  "dunedaq-v2.8.2"
  ▼ 2:
      0:  12157
      1:  "Fri, 19 Nov 2021 12:20:21 GMT"
      2:  "Fri, 19 Nov 2021 12:21:27 GMT"
      3:  "np02_coldbox"
      4:  "PROD"
      5:  "dunedaq-v2.8.2"
```

Configuration archiving

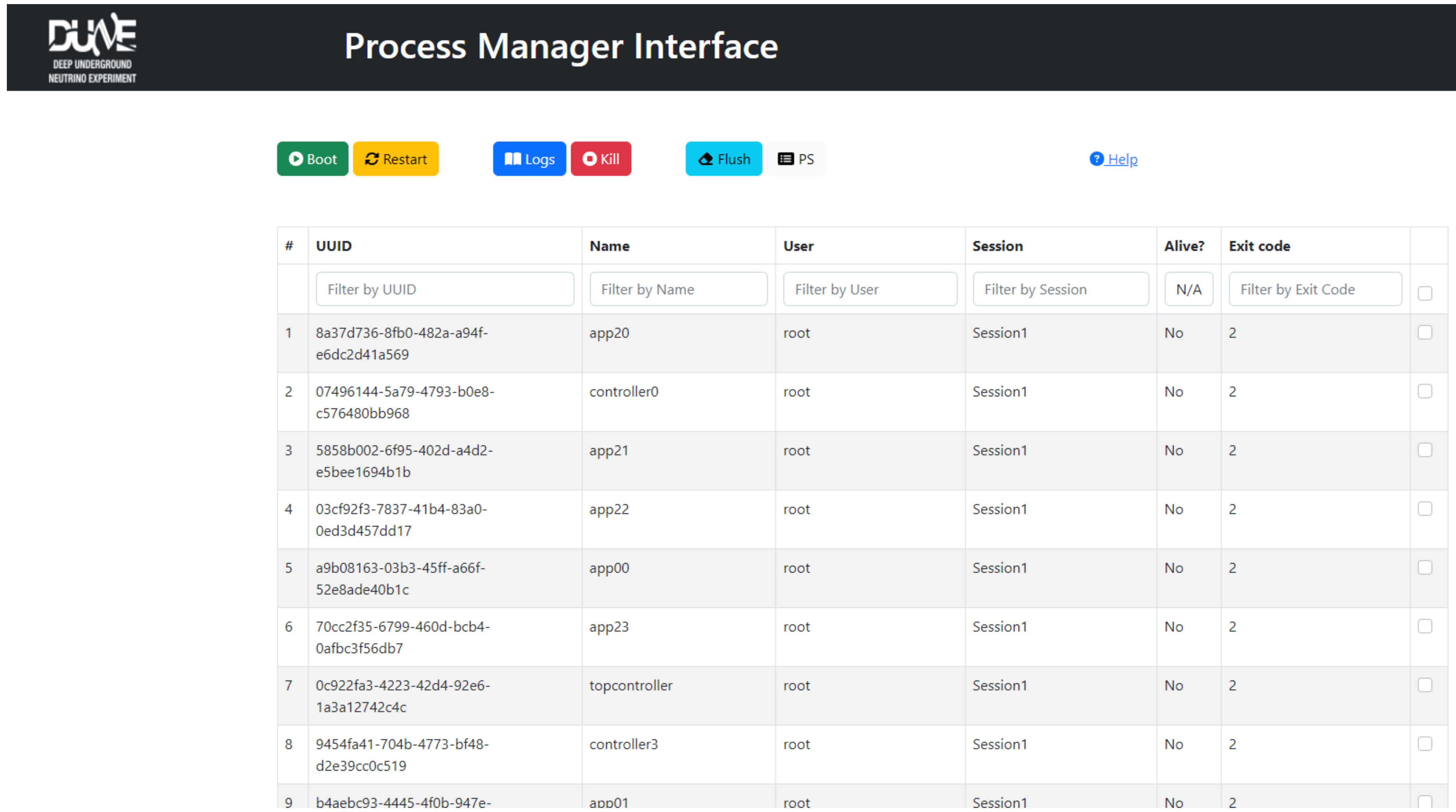
New run control

- Couple of issues with NanoRC
  - Monolithic/not distributed
  - Not able to handle multiple users interacting with it at the same time
  - Accept and handle automated actions
  - No sub-transitions
- Need a new run control
  - In-house project (no licence needed…)
  - Code-named drunc (Dune RUN Control)
  - Python
  - Uses explicit separation (i.e. different processes) between
    - UIs
    - Controllers
    - Process manager



New run control

- The run control is written gRPC (Google's Remote Procedural Calls)
  - Allows network function calls with predefined message formats (protobuf) and some error handling
  - Able to handle "stream" messages
    - Used for notifier systems
- Now fully containerised
- Implements a mock authorisation/authentication system
- Started from Process Manager and UIs
  - Able to replicate the functionality from nanorc (SSH, but K8s envisaged soon)
  - UI written in React by a colleague from Prague University
- Controller able to control subcontrollers



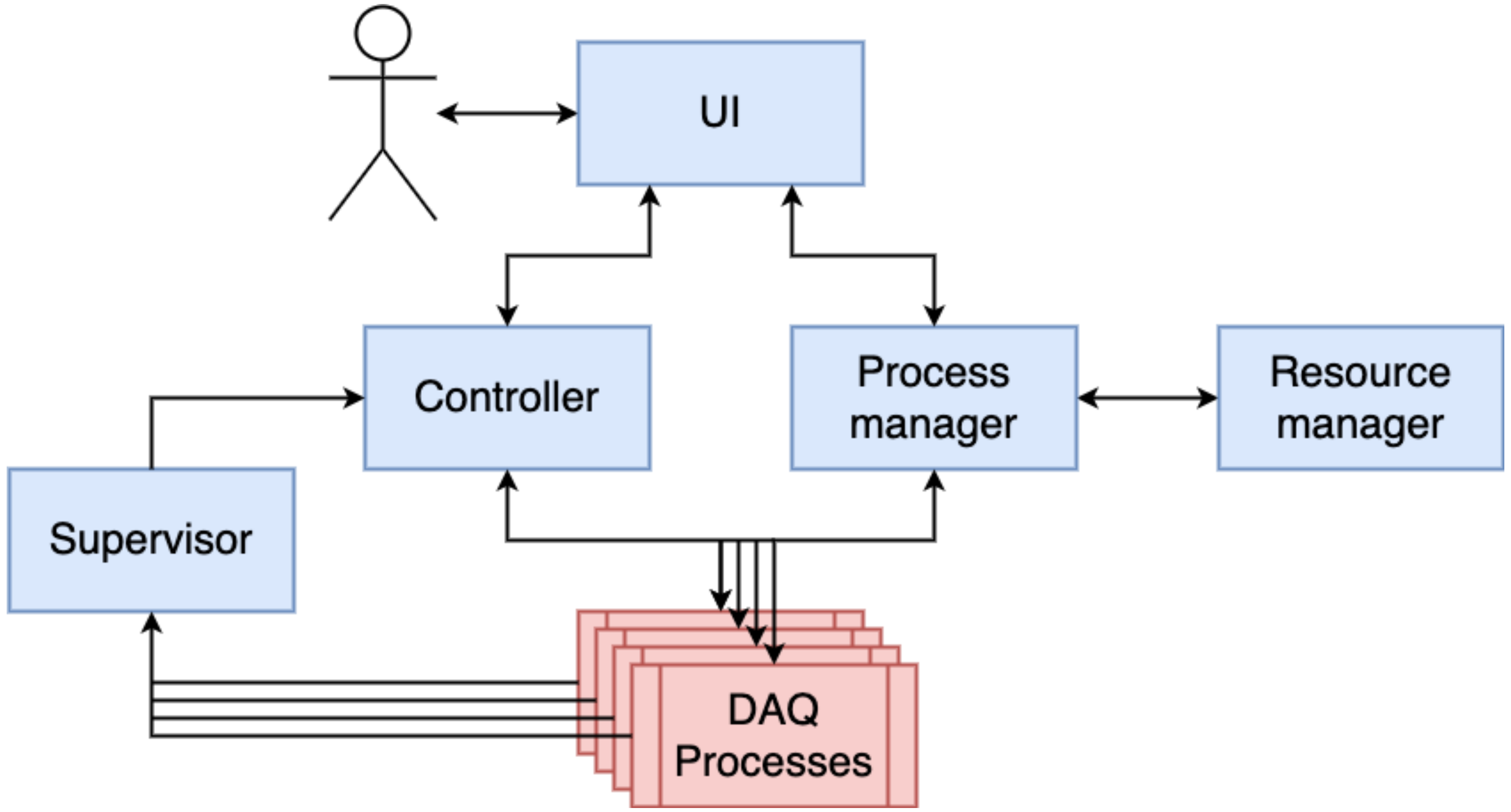
DEEP UNDERGROUND NEUTRINO EXPERIMENT
Process Manager Interface
Boot
Restart
Logs
Kill
Flush
PS
Help

#	UUID	Name	User	Session	Alive?	Exit code	
	Filter by UUID	Filter by Name	Filter by User	Filter by Session	N/A	Filter by Exit Code	□
1	8a37d736-8fb0-482a-a94f-e6dc2d41a569	app20	root	Session1	No	2	□
2	07496144-5a79-4793-b0e8-c576480bb968	controller0	root	Session1	No	2	□
3	5858b002-6f95-402d-a4d2-e5bee1694b1b	app21	root	Session1	No	2	□
4	03cf92f3-7837-41b4-83a0-0ed3d457dd17	app22	root	Session1	No	2	□
5	a9b08163-03b3-45ff-a66f-52e8ade40b1c	app00	root	Session1	No	2	□
6	70cc2f35-6799-460d-bcb4-0afbc3f56db7	app23	root	Session1	No	2	□
7	0c922fa3-4223-42d4-92e6-1a3a12742c4c	topcontroller	root	Session1	No	2	□
8	9454fa41-704b-4773-bf48-d2e39cc0c519	controller3	root	Session1	No	2	□
9	b4aebc93-4445-4f0b-947e-	app01	root	Session1	No	2	□

Process Manager
Zbynek Kral

Command interface

- Drunc will enable many users to connect to the same controller
  - Able to see what is happening in the system
  - Handle expert and shifter interactions
- Supervisor interface
  - The supervisor is a process that uses monitoring/DQM to act automatically on the system to recover it
  - Fundamental for reaching our very high uptime requirement (99% overall)
  - Run control will treat the Supervisor as a command interface



Conclusion

- Using nanorc successfully to record data at EHN1 and for the coldboxes
  - Integrated with DAQ services running at CERN
  - Packaging the services to enable production-like running in other places (FNAL)
- Developing a new run control
  - Most of the functionality is expected to be ready in a year