Microservices Status Report

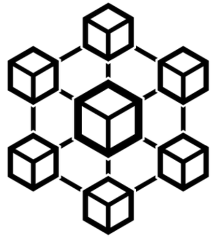
Technical Remarks Edition

Gerhard Brandt, Jonas Schmeing, Marvin Geyik, Maren Stratmann,
Adrian Miemczyk, Dominik Schlothane, Wolfgang Wagner (Wuppertal)
Daniele Dal Santo (Bern) Matthias Wittgen (SLAC)
Benedikt Vormwald (CERN) Hava Schwartz (SCIPP)

ITk TDAQ Session
ATLAS ITk Week
7 March 2023

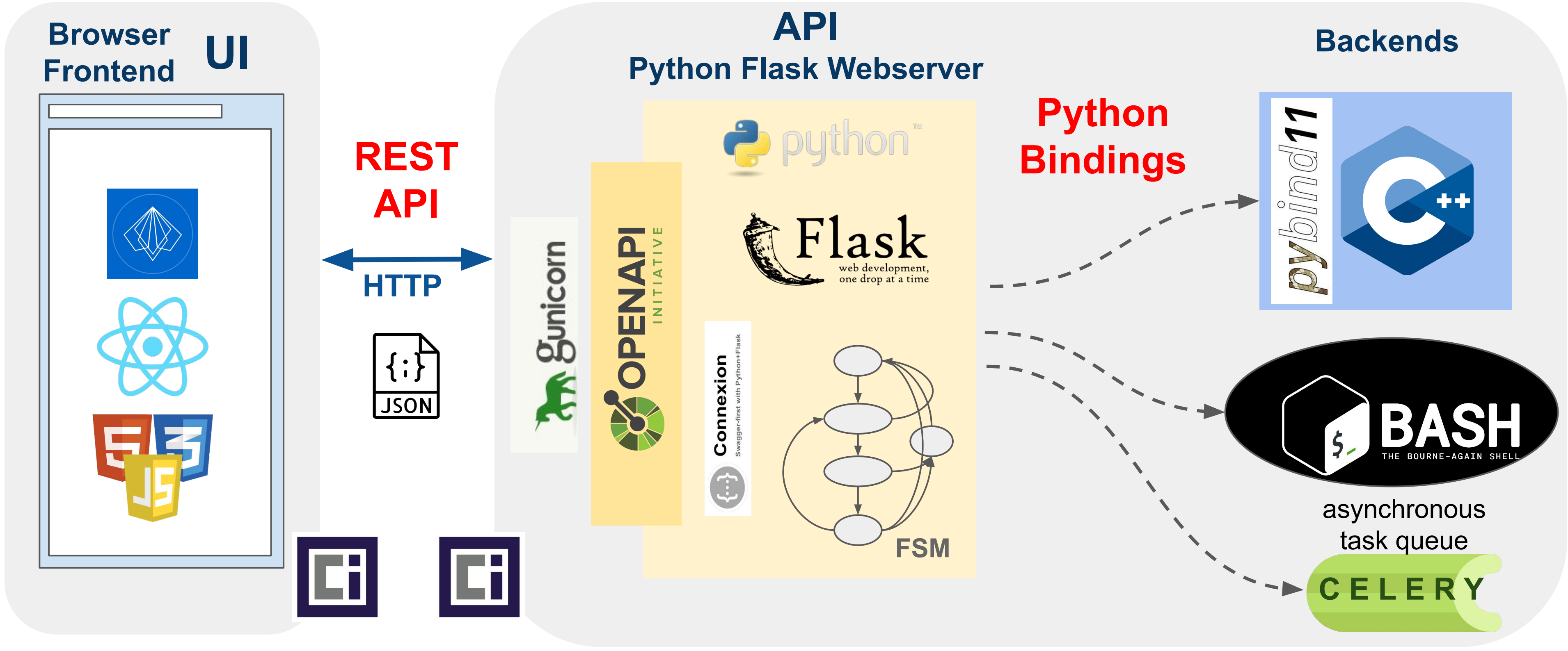
Reminder: What are the "microservices"?

- New top-level ITk **online software** UI framework designed from scratch.
- Partition system in small web servers that do
  "one thing and one thing well": microservices
- This effort situates itself *upstream* of **all** other sw development for Phase II
- Aim to provide top level online operation interface.
  - ***No*** time critical tasks (in the DAQ / TTC context).
  - ***No*** duplication of existing functionality.
- Learn from and improve on features of current ID online software.
- Part of ITk Pixel Systemtest LLS FDR.



https://microservices.io/

Architecture of a Microservice



Leveraging Microservices Similarity

- Same basic directory tree for every microservice.
- Enables lateral maintenance between microservices.
- Still decoupled to allow independent development / updating.
- Matrix (not yet used in software management) can be used to track status / progress.
- Ideally automate with script which can also format, lint, analyse semantics (security issues etc.).

- Developers still need to learn the API/UI stack once.

Microservices			template	dashboard	optoboard	felix	daqapi	configdb	resultviewer	GALE
Frontend		ui/								
		package.json								
	Development	tasks.sh								
	Containerization	Dockerfile								
		docker-compose								
	Deployment	...								
OpenAPI		openapi/								
		openapi.yml								
Python	project	pyproject.toml								
		itk_demo_{service}/								
	Flask	app.py								
	FSM	states.py								
	Celery	celery.py								
		tasks.py								
	Backend	backend.py								
	Development	tasks.sh								
	Containerization	Dockerfile								
		docker-compose								
	CI	gitlab-ci.yml								
	Deployment	...								

Containerization and GitLab CI/CD

- **Microservices have fully embraced containerization.**
- Goal: largely remove need for local software building / installation at system test sites.
- "Only" need to adjust local configuration "Infrastructure-as-Code".
- Configurations can live in private repos in `/deployments` GitLab subgroup.
- Need to develop **`Dockerfiles`**, **`docker-compose.yml`** and **`gitlab-ci.yml`**
- Ideal workflow still under development.
  - Using cascade of bash scripts.
  - Probably heavily use `docker compose`

- Take full advantage of CERN GitLab Premium DevOps.
- Automatic builds using CI (Continuous Integration).
- Use GitLab Runners to build images
  - CERN shared (w. cvmfs mounts).
  - dedicated servers for testing.

KB0005851

- Images stored directly in GitLab
- For third-party tools
  - Preference for official images.
  - Copy to CERN harbor registry.

registry.cern.ch

- CD (Continuous Deployment): Working on automated deployment of testing environment based on emulator.
- Local building + running possible during development:
  - **`tasks.sh`** script

Continuous Integration Details

- CI pipelines defined in `.gitlab-ci.yml` KB0003690
- Use kaniko builder image from CERN Linux Service https://gitlab.cern.ch/ci-tools/docker-builder
- On every push to Git repo :
  - build image
  - tag with commit hash SHA, push
  - tag also with :latest
- On every tag of Git repo:
  - build image
  - tag with Git tag
- Image naming convention:
  - `{repo registry URL}/{repo name}-{suffix}:{tag}`

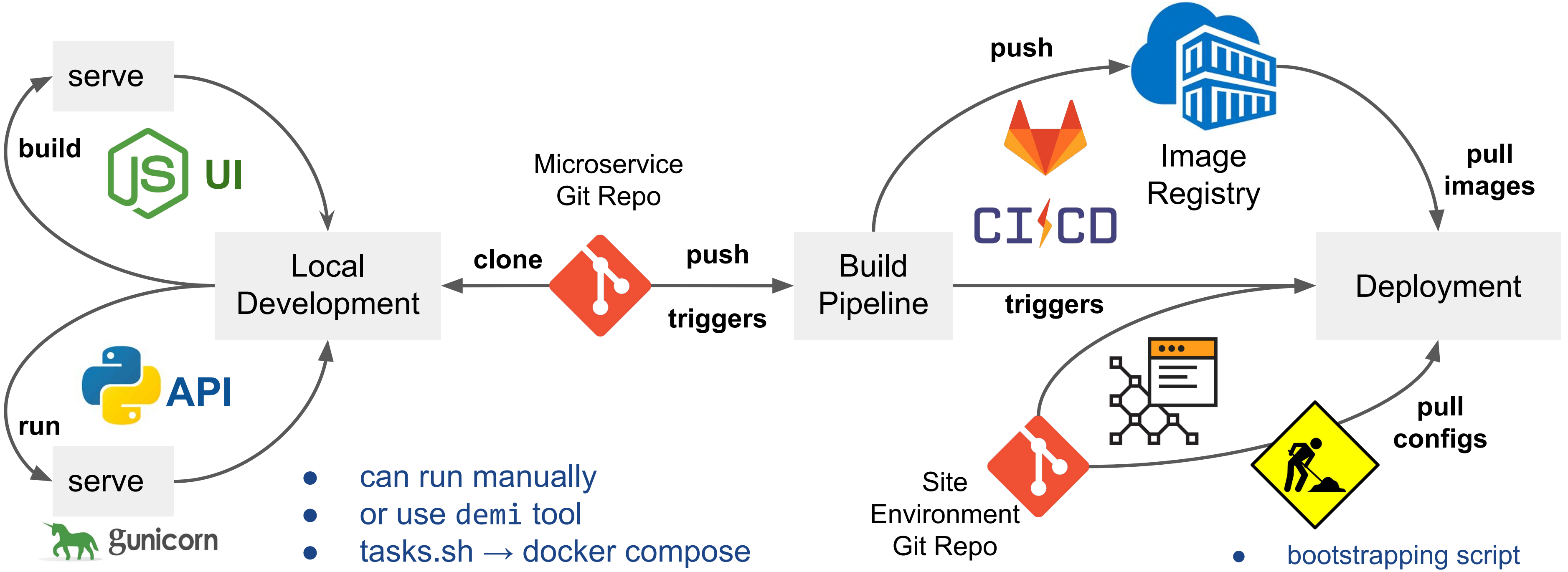
can be used as local short tag

Example:

`gitlab-registry.cern.ch/atlas-itk-pixel-systemtest/itk-demo-sw/itk-demo-configdb/itk-demo-configdb-ui:00ffe95e56adfe15de7a93c4d4dd712b53712386`

- Typically two images per microservice
  - `{repo name}-api` Flask server
  - `{repo name}-ui` Frontend
- Backends usually built in extra images eg. `containers/felix-container`

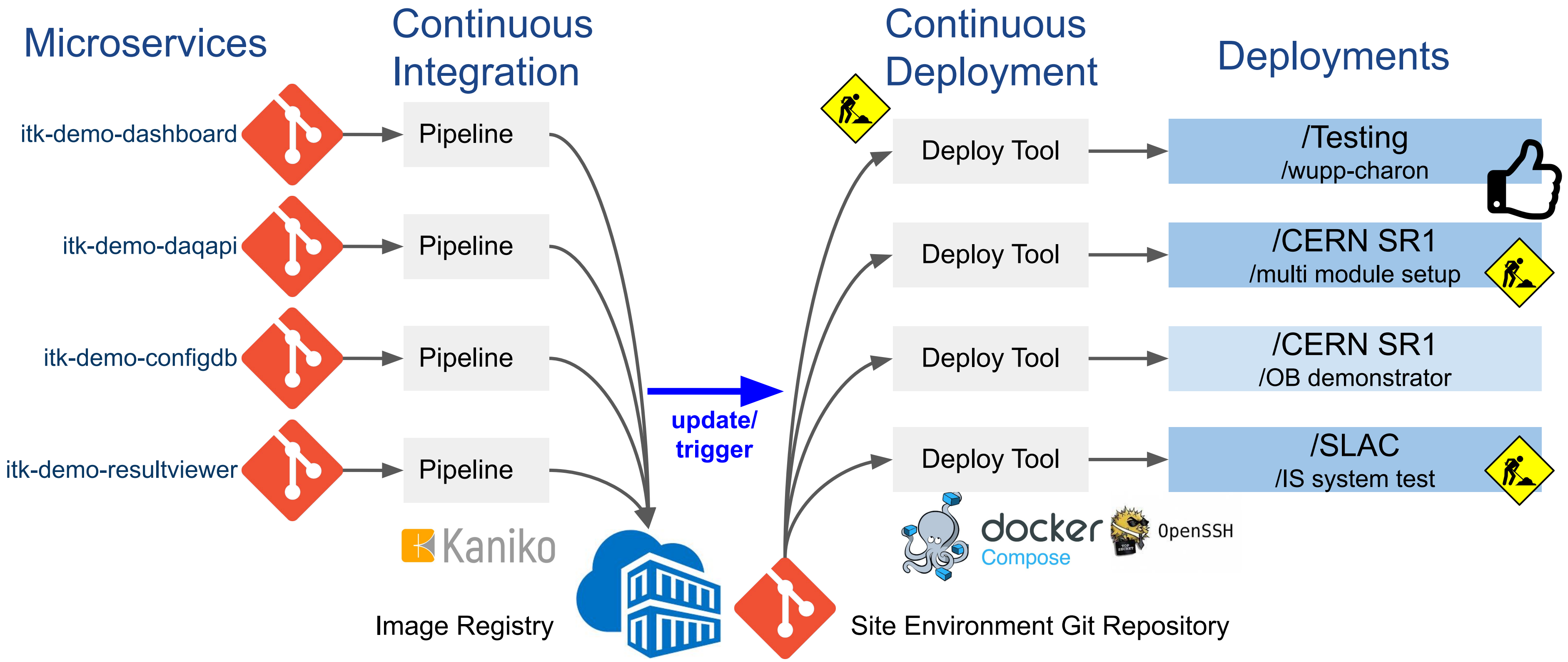
Microservices Development: DevOps + GitOps Workflow



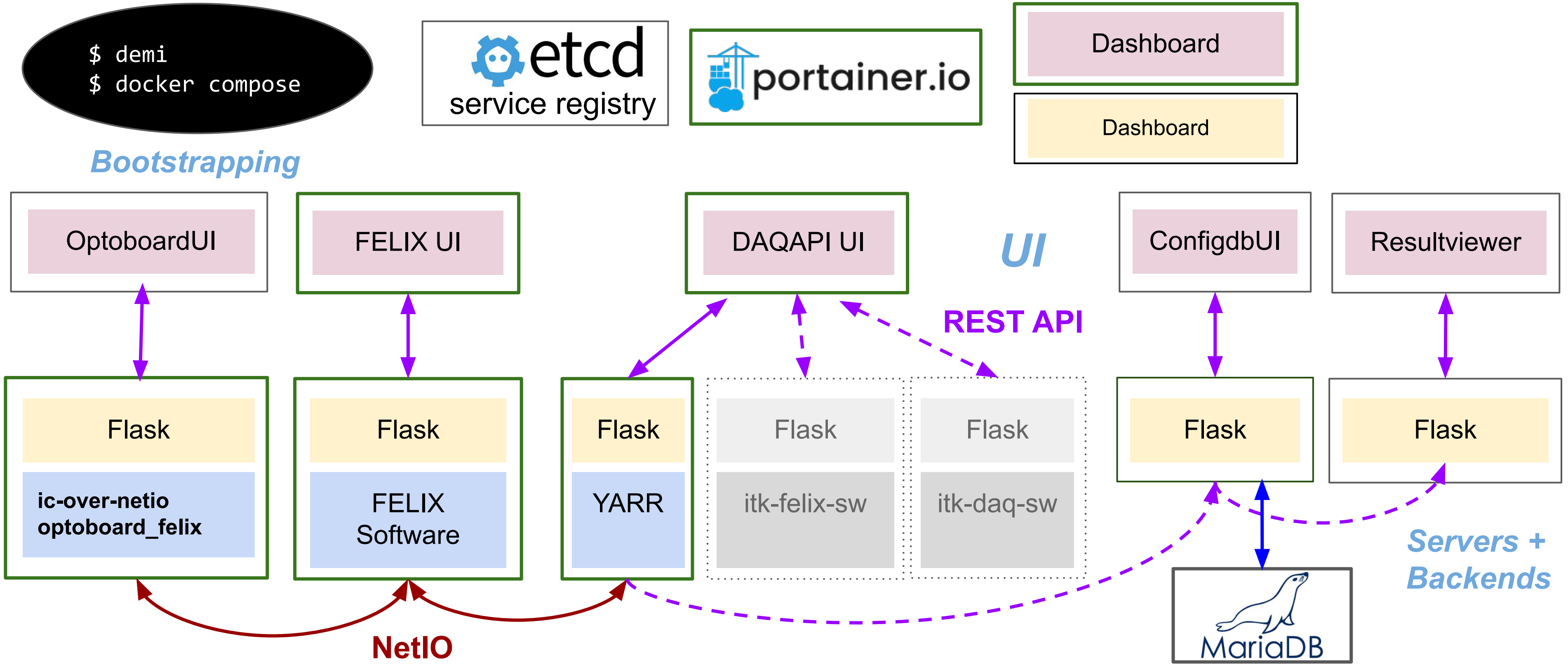
- can run manually
- or use `demi` tool
- tasks.sh → docker compose
- (option for dockerless deployment)

- bootstrapping script
- CD → docker compose
- I-a-C/GitOps tool?

Microservices + Site Deployment Workflow



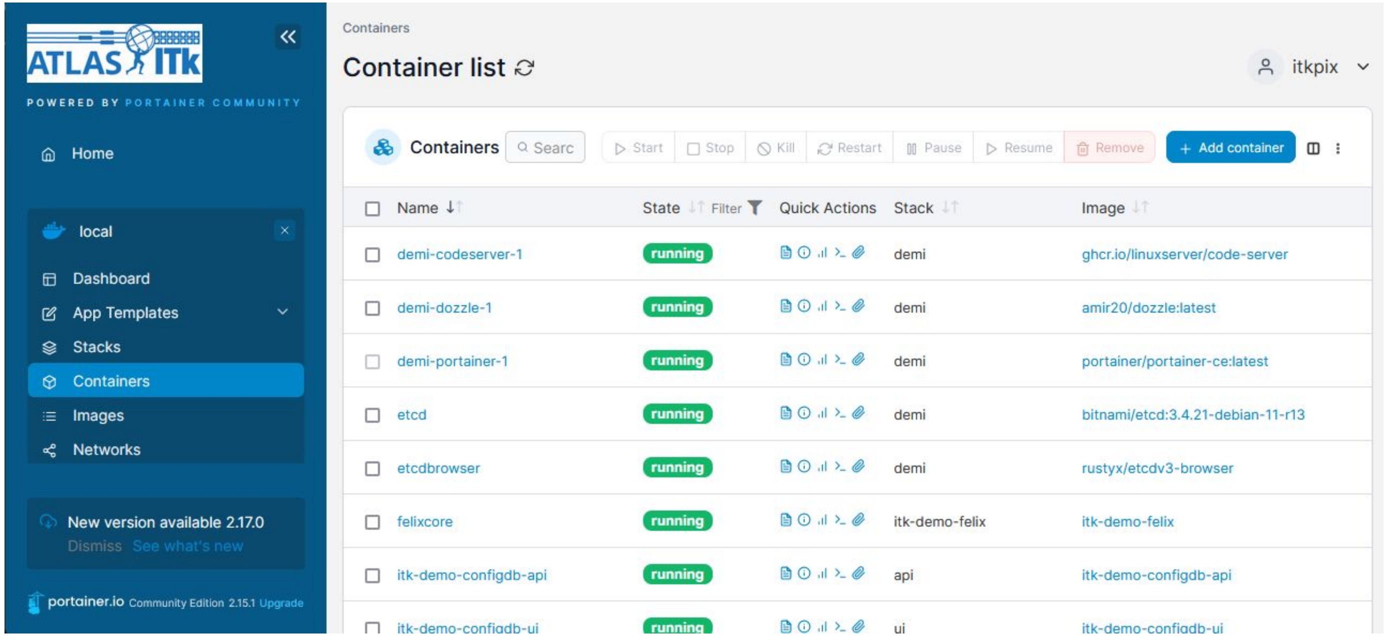
Overview over containerized SR1 deployment



Portainer - Container Management Platform

portainer.io

- Open Source third-party tool to manage Docker containers.
- Possible to start / stop / update / inspect and log in to containers.
- Comprehensive generic tool useful in addition to custom microservice UIs
- Stand-in for dedicated ITk UIs



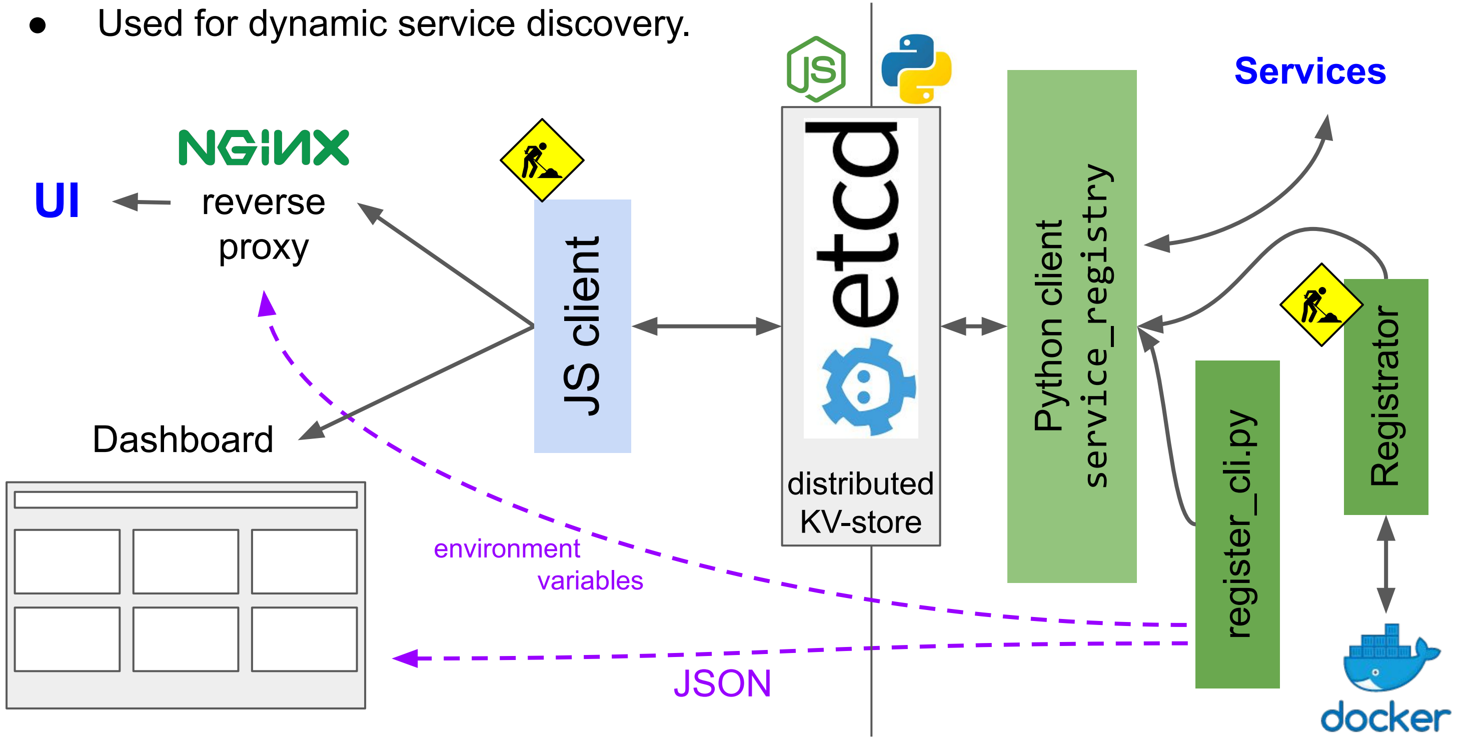
New version available 2.17.0
Dismiss See what's new

Name	State	Quick Actions	Stack	Image
demi-codeserver-1	running		demi	ghcr.io/linuxserver/code-server
demi-dozzle-1	running		demi	amir20/dozzle:latest
demi-portainer-1	running		demi	portainer/portainer-ce:latest
etcd	running		demi	bitnami/etcd:3.4.21-debian-11-r13
etcdbrowser	running		demi	rustyx/etcdv3-browser
felixcore	running		itk-demo-felix	itk-demo-felix
itk-demo-configdb-api	running		api	itk-demo-configdb-api
itk-demo-configdb-ui	running		ui	itk-demo-configdb-ui

Portainer running in SR1

Service Registry

- Single source of truth for all runtime configuration.
- Used for dynamic service discovery.



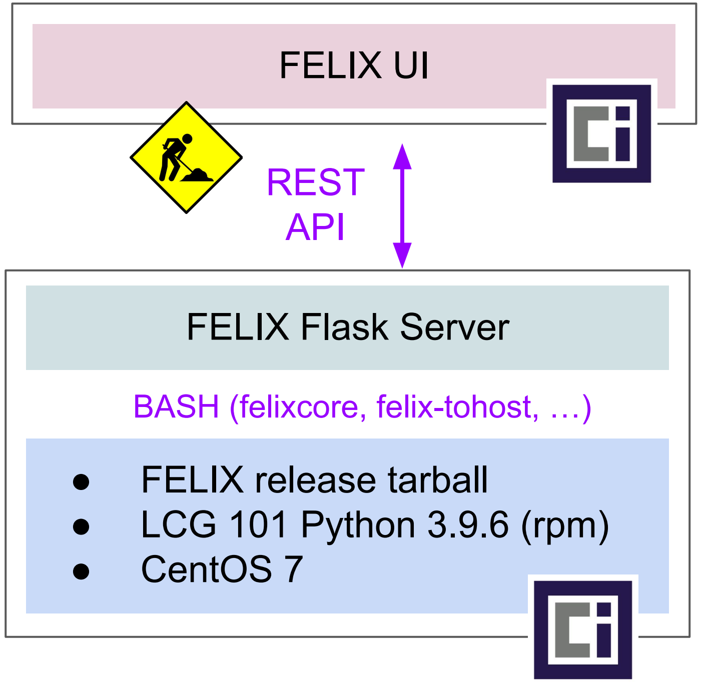
- Currently services registered at startup via CLI (scripted).
- **Registrator** will constantly inspect Docker and do health checks to keep registry up-to-date.
- Services query via Python client.
- Frontends query via JS client.

Status of FELIX Software Container

itk-demo-felix

- Based on special `felix-bin-container` using latest FELIX binary releases

  https://gitlab.cern.ch/atlas-itk-pixel-systemtest/itk-demo-sw/containers/felix-bin-container
- In discussion with FELIX group to move to FELIX
  https://gitlab.cern.ch/atlas-tdaq-felix/felix-image
- Shell tests successful (→ Portainer screenshots)
  - configured FELIX and elinks
- Under discussion how to manage felixcore / felix-star processes inside container
  - s6 init system https://skarnet.org/software/s6/
  - supervisord http://supervisord.org/
  - restart container from GUI?
- GUI needs overhaul to reflect functionality.

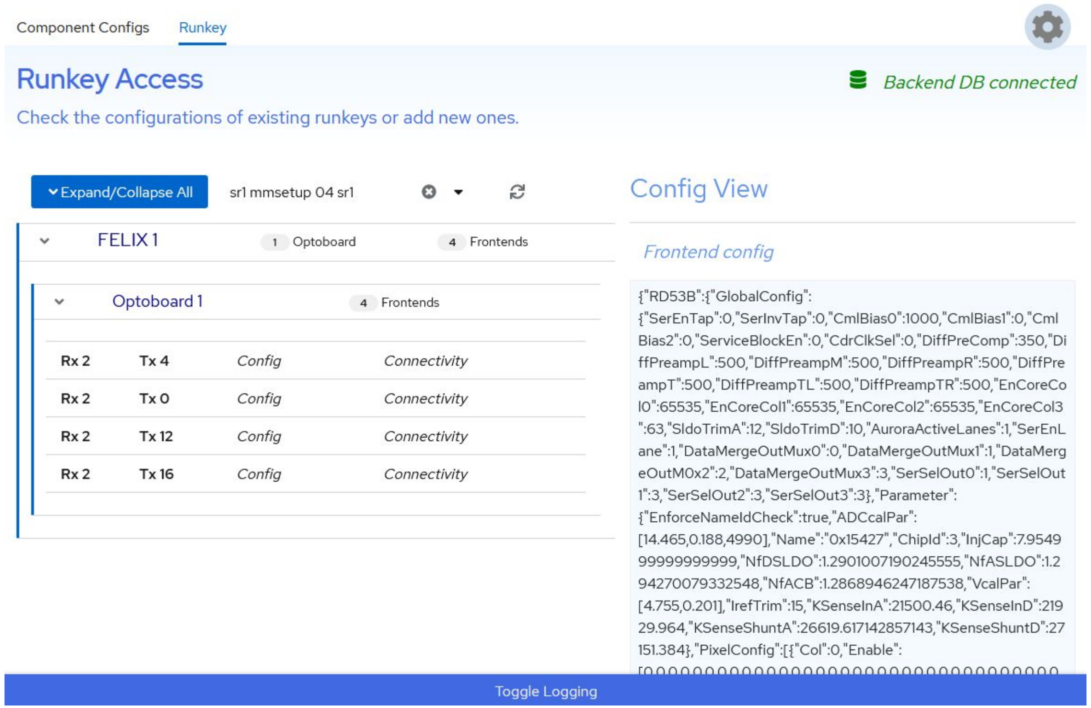


ConfigDb Microservice Status

itk-demo-configdb

- ConfigDb stores Runkeys (sets of configurations) and scan results.
- For FDR using a simple “matrix” data model with one data path for each FE to SWROD.
- Implemented Backends for FDR: SQLite and **MariaDb**
- Bootstrapping being finalized.

- ***Next version of ConfigDb with a more flexible configuration and connectivity data model currently under development →***



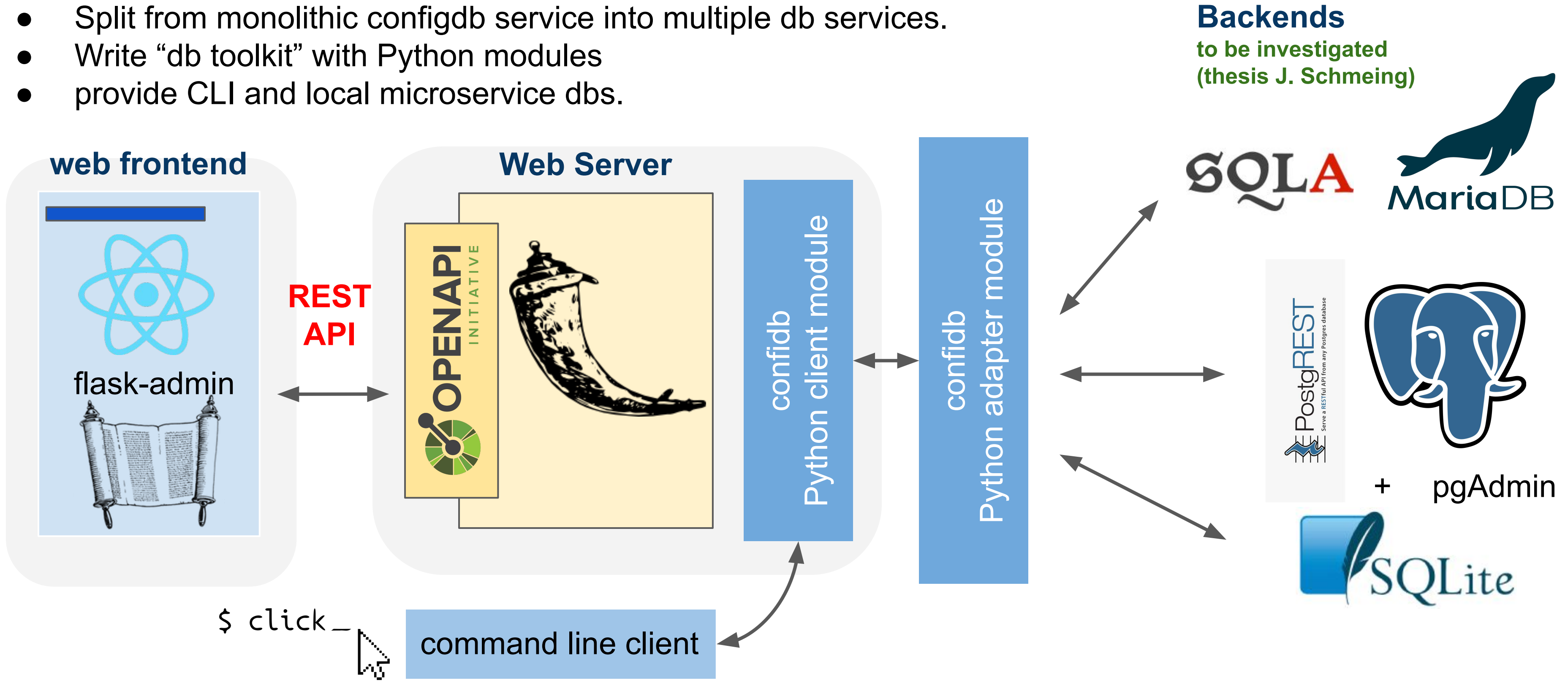
Rx 2	Tx 4	Config	Connectivity
Rx 2	Tx 0	Config	Connectivity
Rx 2	Tx 12	Config	Connectivity
Rx 2	Tx 16	Config	Connectivity

{"RD53B":{"GlobalConfig":{"SerEnTap":0,"SerInvTap":0,"CmlBias0":1000,"CmlBias1":0,"CmlBias2":0,"ServiceBlockEn":0,"CdrClkSel":0,"DiffPreComp":350,"DiffPreampL":500,"DiffPreampM":500,"DiffPreampR":500,"DiffPreampT":500,"DiffPreampTL":500,"DiffPreampTR":500,"EnCoreCol0":65535,"EnCoreCol1":65535,"EnCoreCol2":65535,"EnCoreCol3":63,"SldoTrimA":12,"SldoTrimD":10,"AuroraActiveLanes":1,"SerEnLane":1,"DataMergeOutMux0":0,"DataMergeOutMux1":1,"DataMergeOutM0x2":2,"DataMergeOutMux3":3,"SerSelOut0":1,"SerSelOut1":3,"SerSelOut2":3,"SerSelOut3":3},"Parameter":{"EnforceNameIdCheck":true,"ADCcalPar":[14.465,0.188,4990],"Name":"0x15427","ChipId":3,"InjCap":7.954999999999999,"NfDSLDO":1.2901007190245555,"NfASLDO":1.294270079332548,"NfACB":1.2868946247187538,"VcalPar":[4.755,0.201],"IrefTrim":15,"KSenseInA":21500.46,"KSenseInD":21929.964,"KSenseShuntA":26619.617142857143,"KSenseShuntD":27151.384},"PixelConfig":[{"Col":0,"Enable":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Runkey Access tab in ConfigDb UI.

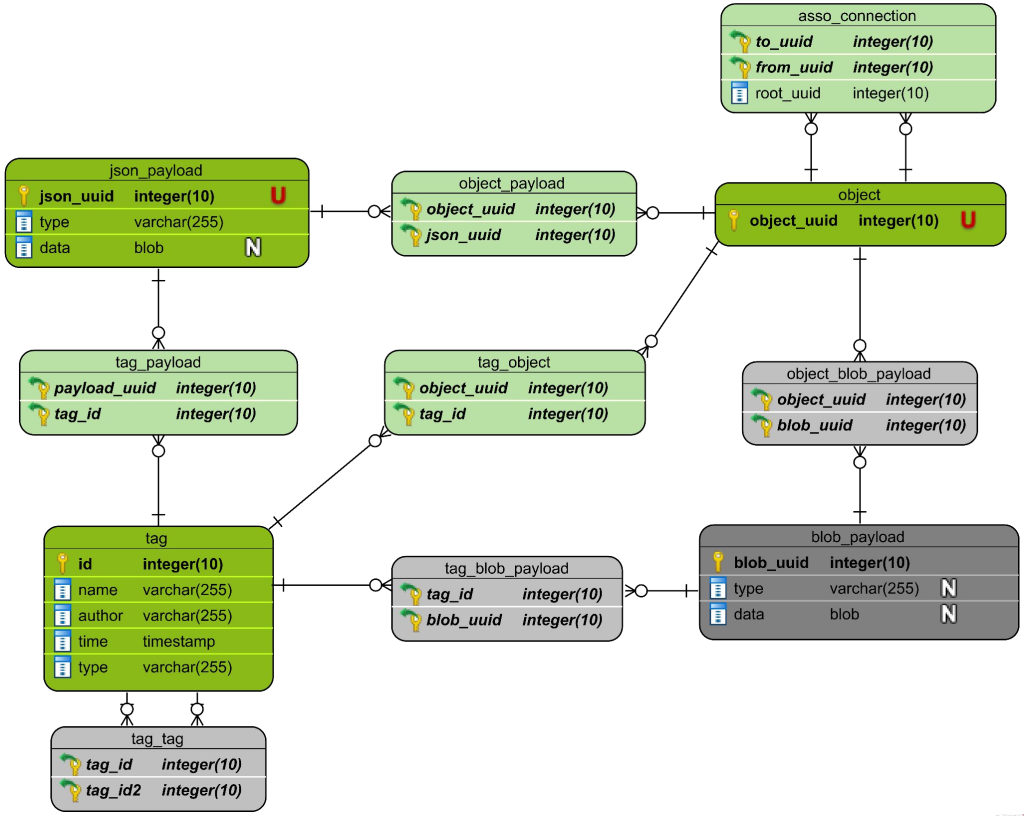
ConfigDb Under Development

- Split from monolithic configdb service into multiple db services.
- Write “db toolkit” with Python modules
- provide CLI and local microservice dbs.

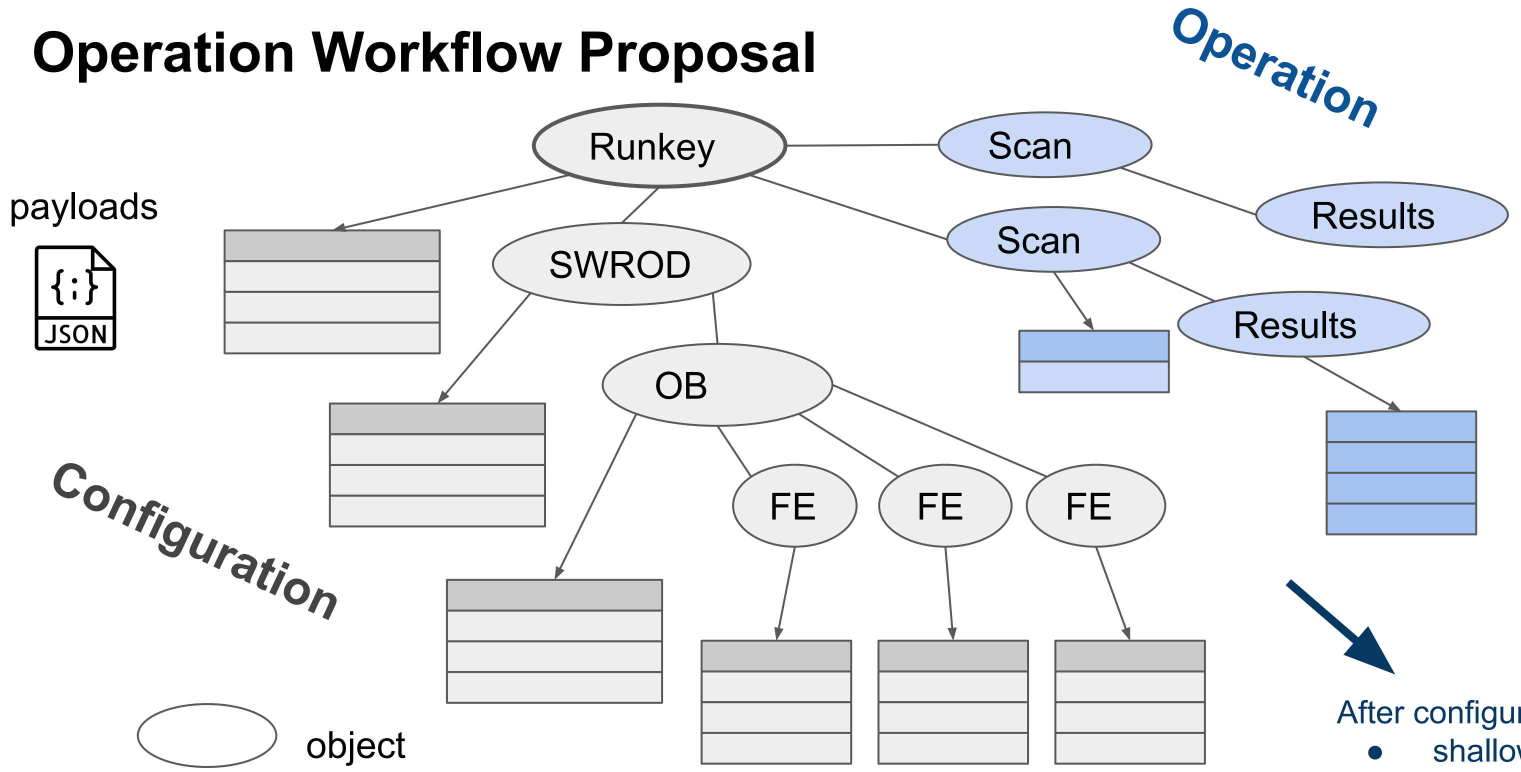


Current Schema Proposal

- Decouple functional blocks using UUIDs as primary keys.
  → Possibility to separate into multiple databases or point at external databases (LocalDb, itkdb).
- Generic objects.
- JSON payloads.
  - Encode binary blobs in base64.
- Tag system to flexibly label everything.
- Connectivity becomes simply standard association table between objects.
- Runkey is root object of traversable tree(s).
- Design started from current ID Pixel CoralDb schema.



Operation Workflow Proposal



tags can point at objects for labelling, grouping and to point at reference objects

▨ immutable info (eg. module id, metadata, parents)

☐ mutable info (eg. FE config, results)

After configuration changing scan:

- shallow-copy to obtain next runkey
- copy-on-write when generating/ editing next runkey

Summary

- The Microservices project has spent a lot of effort on containerization and implementing CI/CD.
  - Effort is starting to pay off - don’t need build images locally anymore.
- Integration efforts progressing but several dots still need to be connected.
  - Feasibility of running FELIX and DAQ Software in containers demonstrated.
- Development has started on the next version of a flexible database for ITk connectivity, configuration and result storage.
- Automated Bootstrapping / easy deployment at testing sites still major construction site.
  - Have temporary solution based on cascade of BASH scripts.
  - Need to develop a light-weight container orchestration solution adapted to our specific use case.

Ressources

Meeting	Bi-Weekly Wednesday 1700 CEST.
Egroup	**atlas-itk-demo-sw-devel**
Mattermost	https://mattermost.web.cern.ch/itkpixel ○ /itk-demo-sw-users ○ /itk-demo-sw-devel
GitLab	https://gitlab.cern.ch/atlas-itk-pixel-systemtest/itk-demo-sw ○ also hosts Wiki ○ PyPi and npm package registries. ○ OCI image registry
JIRA	https://its.cern.ch/jira/projects/ITKDEMOSW

https://gitlab.cern.ch/groups/atlas-itk-pixel-systemtest/itk-demo-sw/-/wikis/resources

NASA Open-Source Science Initiative
TOPS
Open Source Used by NASA (A Small Sampling!)
SQLAlchemy
NumPy
scikit-image
image processing in python
blender
git
OSGeo Project
RAILS
QGIS
NGINX
Ubuntu
node JS
SUBVERSION
Flask
scikit learn
BASH
THE BOURNE-AGAIN SHELL
APACHE
RabbitMQ

NASA at FOSDEM 2023 - similar selection of Open Source tools

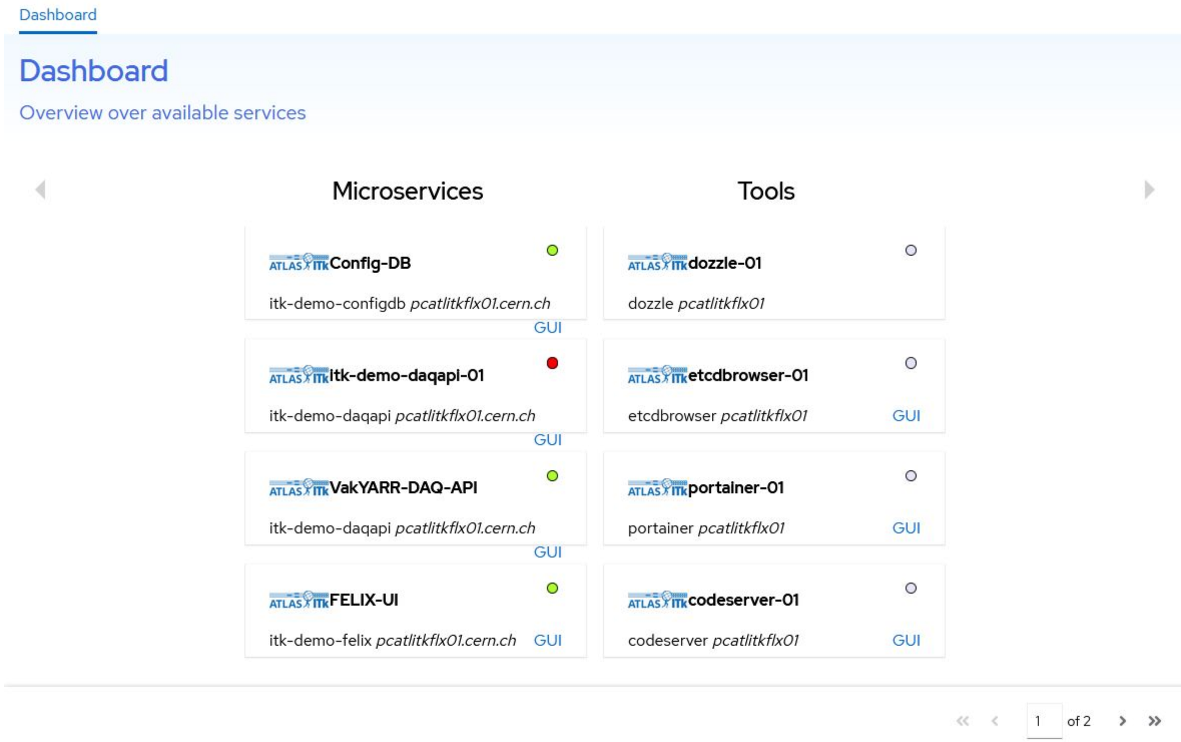
B A C K U P

Dashboard Microservice

itk-demo-dashboard

- Top-level page for system test operation.
- Populated from services in service-registry.
  - Microservices and third-party tools.
  - Currently fed by JSON file → Implement service-registry JS client
- Summary information cards.
  - Health indicator.
- Links to UI(s).
  - Individual microservice UIs will offer more fine-grained control.
- For expert / admin operations and debugging tools like Portainer, Flower, PhpMyAdmin … are linked.

- Layout being overhauled to be more comprehensive


Dashboard

Dashboard

Overview over available services

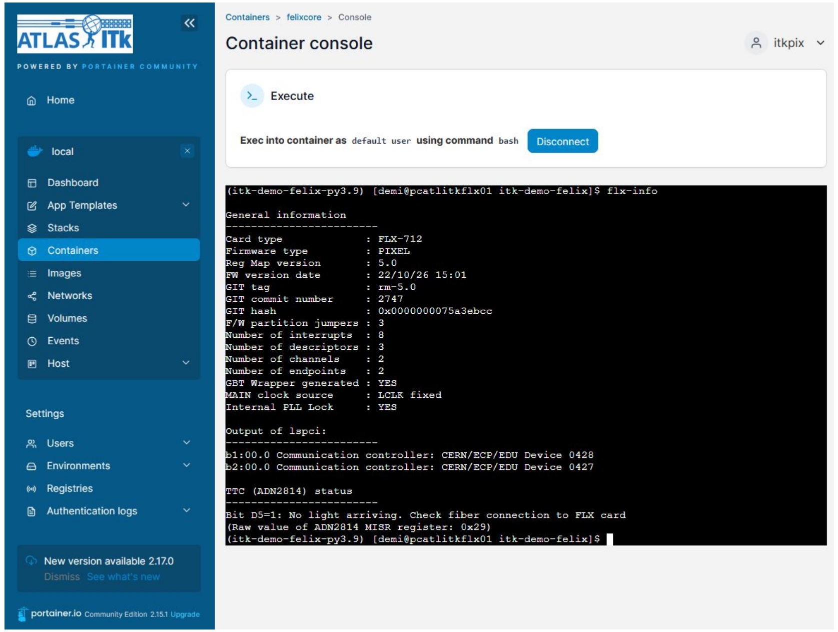
Microservices	Tools
Config-DB ● — itk-demo-configdb *pcatlitkflx01.cern.ch* GUI	dozzle-01 ○ — dozzle *pcatlitkflx01*
Itk-demo-daqapi-01 ● — itk-demo-daqapi *pcatlitkflx01.cern.ch* GUI	etcdbrowser-01 ○ — etcdbrowser *pcatlitkflx01* GUI
VakYARR-DAQ-API ● — itk-demo-daqapi *pcatlitkflx01.cern.ch* GUI	portainer-01 ○ — portainer *pcatlitkflx01* GUI
FELIX-UI ● — itk-demo-felix *pcatlitkflx01.cern.ch* GUI	codeserver-01 ○ — codeserver *pcatlitkflx01* GUI

1 of 2

Example of Dashboard running in SR1.

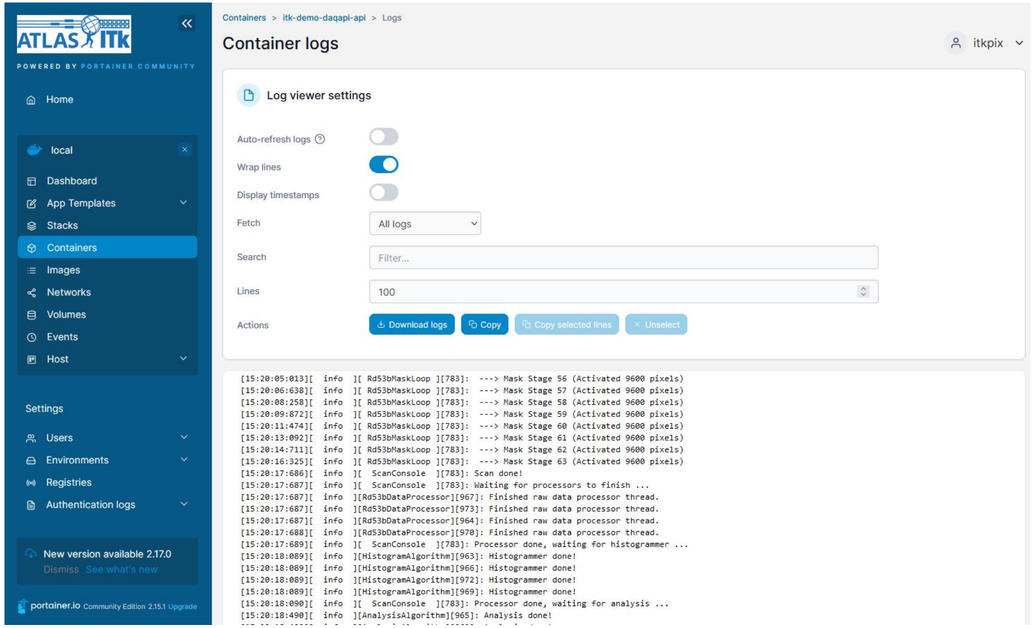
Portainer - More Operation Examples

portainer.io



Exec into container as `default user` using command `bash`

```
(itk-demo-felix-py3.9) [demi@pcatlitkflx01 itk-demo-felix]$ flx-info

General information
------------------------
Card type              : FLX-712
Firmware type          : PIXEL
Reg Map version        : 5.0
FW version date        : 22/10/26 15:01
GIT tag                : rm-5.0
GIT commit number      : 2747
GIT hash               : 0x0000000075a3ebcc
F/W partition jumpers : 3
Number of interrupts  : 8
Number of descriptors : 3
Number of channels    : 2
Number of endpoints   : 2
GBT Wrapper generated : YES
MAIN clock source     : LCLK fixed
Internal PLL Lock     : YES

Output of lspci:
------------------------
b1:00.0 Communication controller: CERN/ECP/EDU Device 0428
b2:00.0 Communication controller: CERN/ECP/EDU Device 0427

TTC (ADN2814) status
------------------------
Bit D5=1: No light arriving. Check fiber connection to FLX card
(Raw value of ADN2814 MISR register: 0x29)
(itk-demo-felix-py3.9) [demi@pcatlitkflx01 itk-demo-felix]$
```



Auto-refresh logs: off
Wrap lines: on
Display timestamps: off
Fetch: All logs
Search: Filter...
Lines: 100

```
[15:20:05:013][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 56 (Activated 9600 pixels)
[15:20:06:638][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 57 (Activated 9600 pixels)
[15:20:08:258][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 58 (Activated 9600 pixels)
[15:20:09:872][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 59 (Activated 9600 pixels)
[15:20:11:474][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 60 (Activated 9600 pixels)
[15:20:13:092][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 61 (Activated 9600 pixels)
[15:20:14:711][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 62 (Activated 9600 pixels)
[15:20:16:325][  info  ][ Rd53bMaskLoop ][783]:  ---> Mask Stage 63 (Activated 9600 pixels)
[15:20:17:686][  info  ][  ScanConsole  ][783]: Scan done!
[15:20:17:687][  info  ][  ScanConsole  ][783]: Waiting for processors to finish ...
[15:20:17:687][  info  ][Rd53bDataProcessor][967]: Finished raw data processor thread.
[15:20:17:687][  info  ][Rd53bDataProcessor][973]: Finished raw data processor thread.
[15:20:17:687][  info  ][Rd53bDataProcessor][964]: Finished raw data processor thread.
[15:20:17:688][  info  ][Rd53bDataProcessor][970]: Finished raw data processor thread.
[15:20:17:689][  info  ][  ScanConsole  ][783]: Processor done, waiting for histogrammer ...
[15:20:18:089][  info  ][HistogramAlgorithm][963]: Histogrammer done!
[15:20:18:089][  info  ][HistogramAlgorithm][966]: Histogrammer done!
[15:20:18:089][  info  ][HistogramAlgorithm][972]: Histogrammer done!
[15:20:18:089][  info  ][HistogramAlgorithm][969]: Histogrammer done!
[15:20:18:090][  info  ][  ScanConsole  ][783]: Processor done, waiting for analysis ...
[15:20:18:490][  info  ][AnalysisAlgorithm][965]: Analysis done!
```

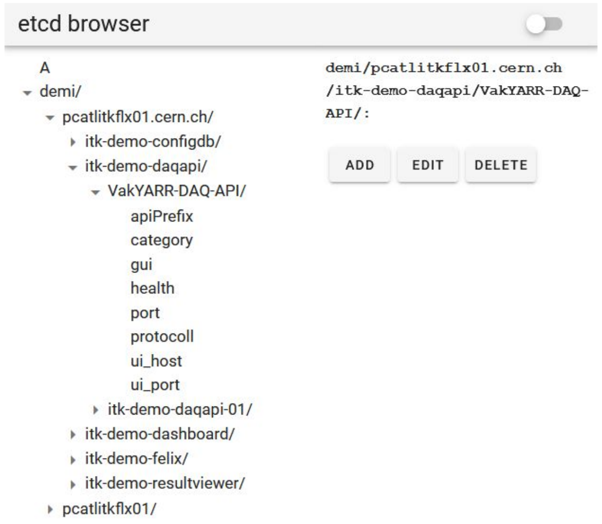
Example of operating felixcore (flx-info) in container via web terminal.

Example of VakYARR scan log output inspection. Digital Scan on ITkPixV1 mm setup (no merging).

Other Third-Party Tools Examples

demi container tools up

- Want to provide complete operations and development environment in the browser
- No more VNC

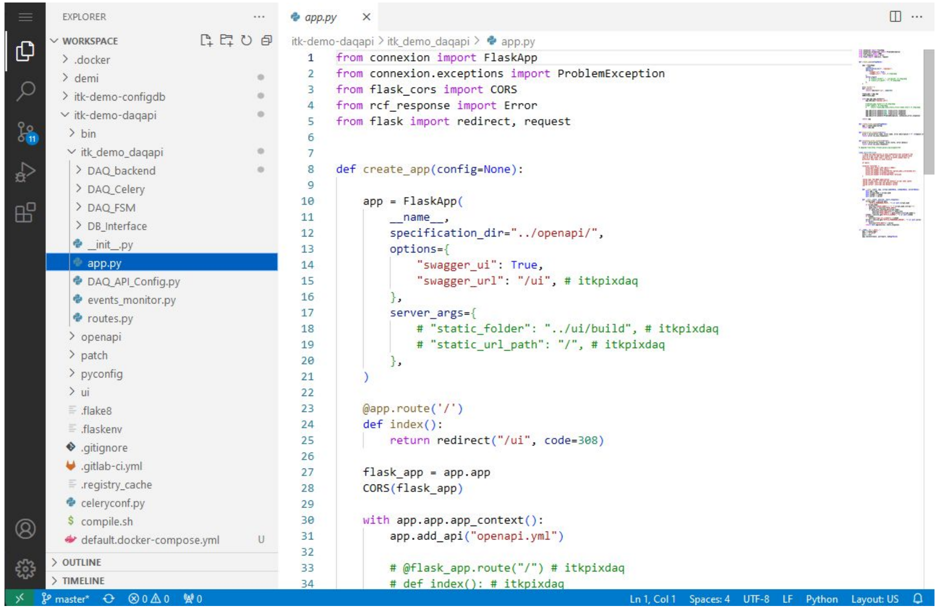


etcd browser

```
A
▾ demi/
    ▾ pcatlitkflx01.cern.ch/
        ▸ itk-demo-configdb/
        ▾ itk-demo-daqapi/
            ▾ VakYARR-DAQ-API/
                apiPrefix
                category
                gui
                health
                port
                protocoll
                ui_host
                ui_port
            ▸ itk-demo-daqapi-01/
        ▸ itk-demo-dashboard/
        ▸ itk-demo-felix/
        ▸ itk-demo-resultviewer/
    ▸ pcatlitkflx01/
```

demi/pcatlitkflx01.cern.ch/itk-demo-daqapi/VakYARR-DAQ-API/:

etcd (service-registry) Browser

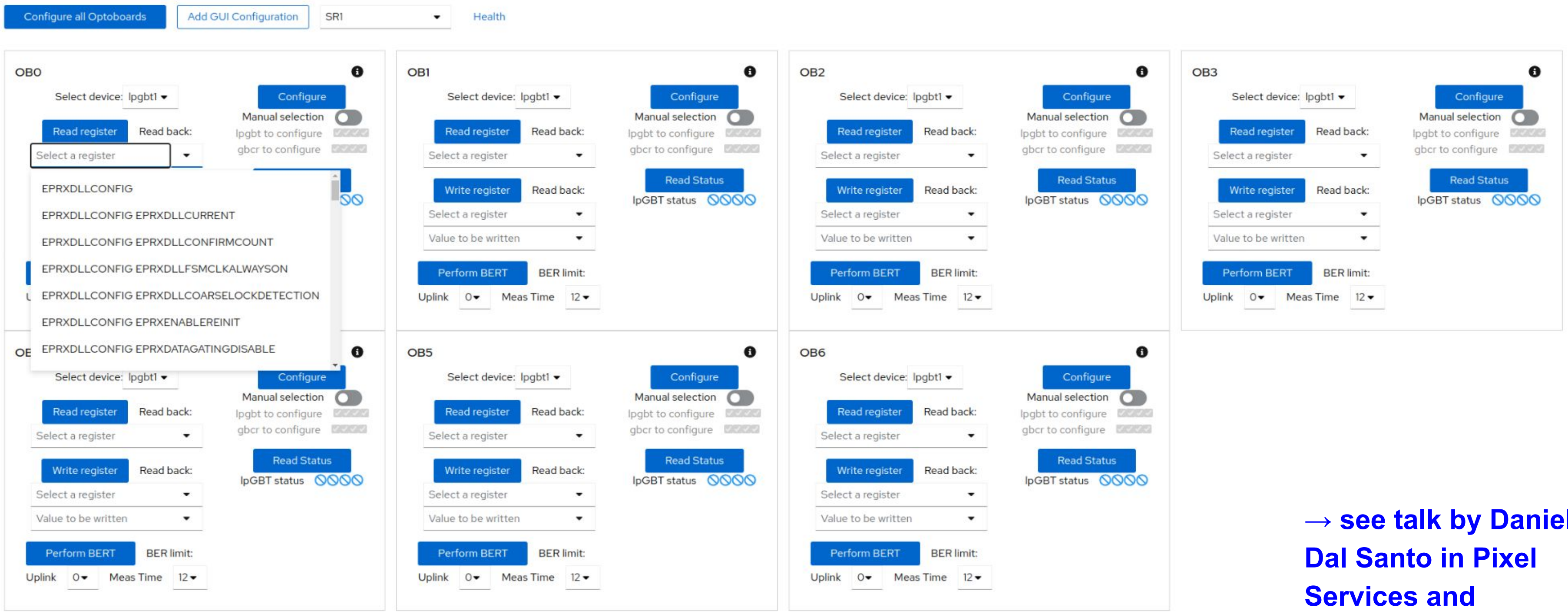


```python
from connexion import FlaskApp
from connexion.exceptions import ProblemException
from flask_cors import CORS
from rcf_response import Error
from flask import redirect, request


def create_app(config=None):

    app = FlaskApp(
        __name__,
        specification_dir="../openapi/",
        options={
            "swagger_ui": True,
            "swagger_url": "/ui", # itkpixdaq
        },
        server_args={
            # "static_folder": "../ui/build", # itkpixdaq
            # "static_url_path": "/", # itkpixdaq
        },
    )

    @app.route('/')
    def index():
        return redirect("/ui", code=308)

    flask_app = app.app
    CORS(flask_app)

    with app.app.app_context():
        app.add_api("openapi.yml")

        # @flask_app.route("/") # itkpixdaq
        # def index(): # itkpixdaq
```

Codeserver (VSCode) DeMi workspace text editor

Optoboard Microservice UI Status

itk-demo-optoboard



→ see talk by Daniele Dal Santo in Pixel Services and Electronics

Optoboard Microservice Status in SR1

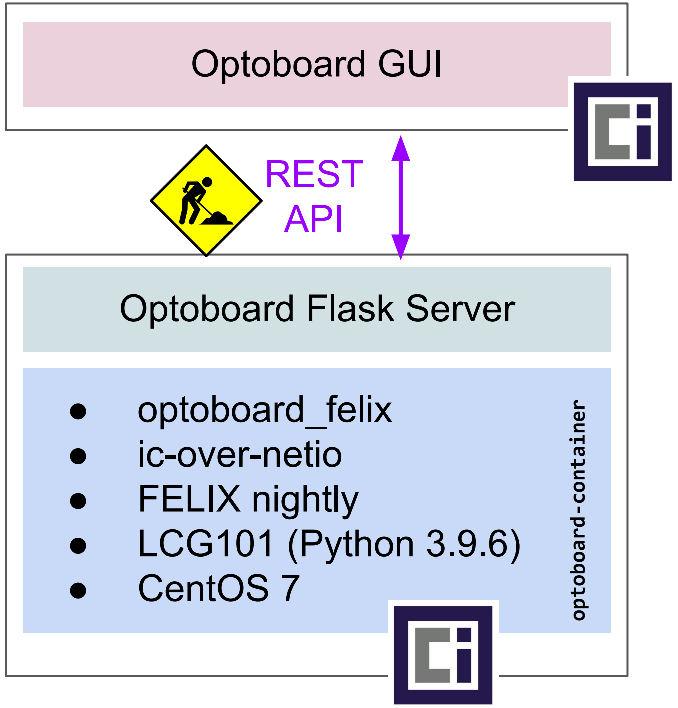
itk-demo-optoboard

- Built on special image `optoboard-container` with all necessary software installed.

  https://gitlab.cern.ch/atlas-itk-pixel-systemtest/itk-demo-sw/containers/optoboard-container

- Tests in SR1
  - Configuration files copied to container
  - Talks ok to felixcore over netio
  - Can read OB status ok
  - *OB Configuration currently unstable, crashes*
  - Plan to try with felix-star and local HEAD of `optoboard_felix`
- Need to try UI with API
- Need to store configurations in configdb

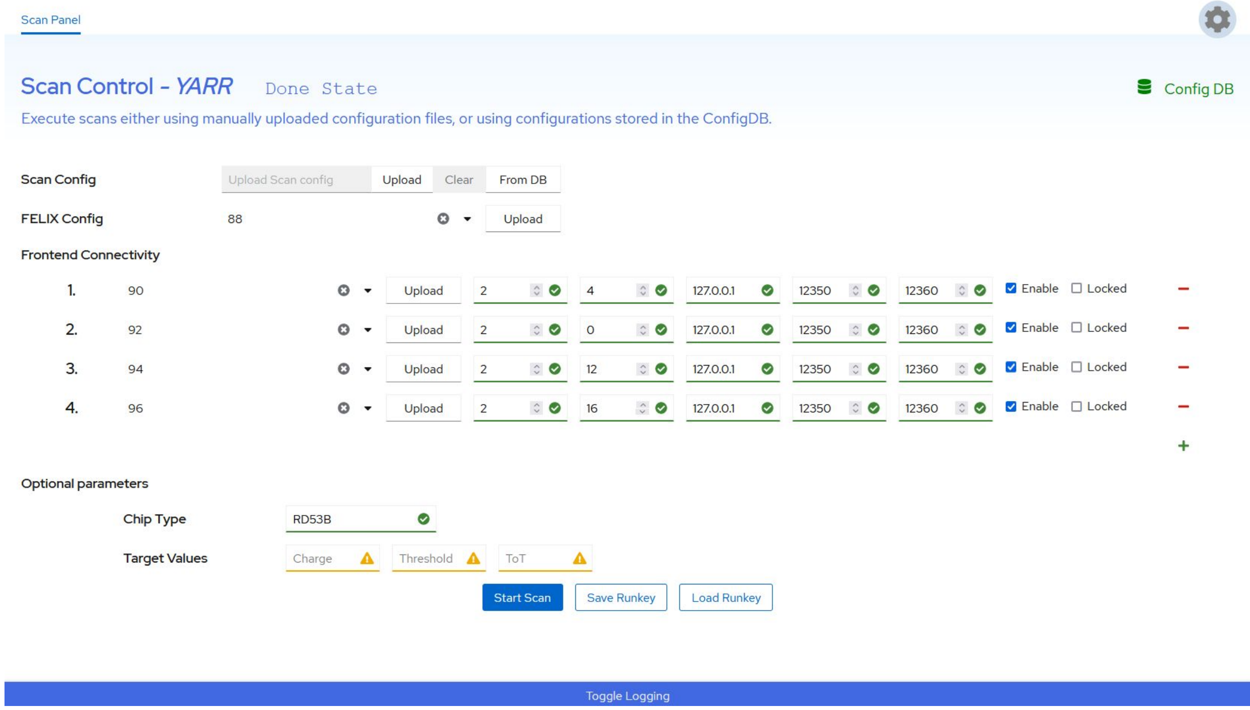
→ **see talk by Daniele Dal Santo in Pixel Services and Electronics**



itk-demo-daqapi

DAQAPI Microservice UI

- Single UI to control different DAQ software flavors currently in existence.
- Backend consists of python bindings (eg. pyYARR) with common adapter.
- Runkeys (complete sets of configuration data) can be
  - Edited by hand using local files.
  - Used with the ConfigDB.



Scan Panel

Scan Control - *YARR*   Done State   Config DB

Execute scans either using manually uploaded configuration files, or using configurations stored in the ConfigDB.

Scan Config: Upload Scan config | Upload | Clear | From DB

FELIX Config: 88 | Upload

Frontend Connectivity

1.	90	Upload	2	4	127.0.0.1	12350	12360	✓ Enable ☐ Locked
2.	92	Upload	2	0	127.0.0.1	12350	12360	✓ Enable ☐ Locked
3.	94	Upload	2	12	127.0.0.1	12350	12360	✓ Enable ☐ Locked
4.	96	Upload	2	16	127.0.0.1	12350	12360	✓ Enable ☐ Locked

Optional parameters

Chip Type: RD53B

Target Values: Charge | Threshold | ToT

DAQAPI with "VakYARR" backend running in SR1

itk-demo-daqapi

Status of DAQAPI Microservice Container

- Works well with YARR emulator.
  - useful for local and automated testing.
- Real Scan works from shell
  - not yet ran from UI.

Backends to be added:

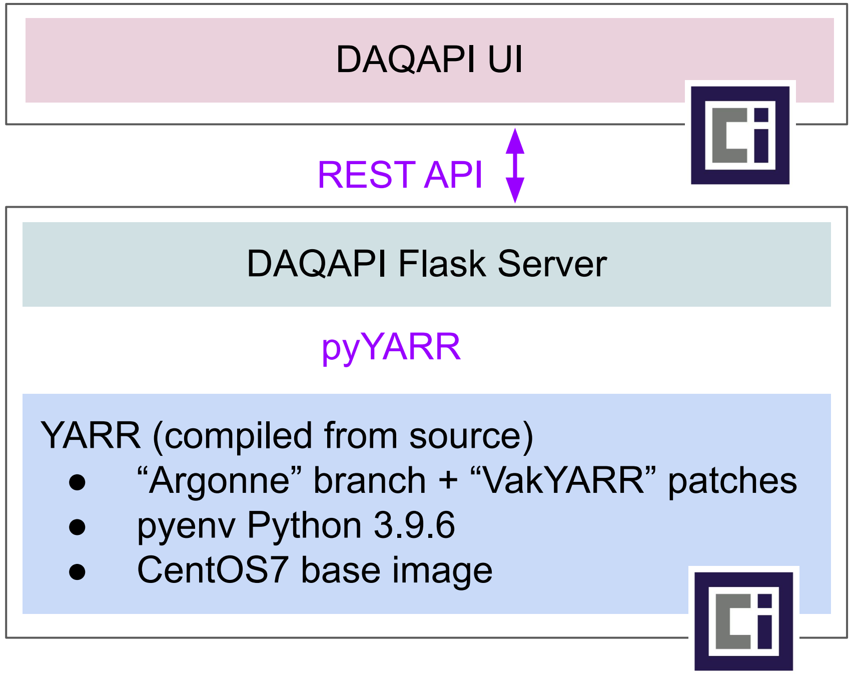
- itk-felix-sw
  - Implementation similar to optoboard container.
- itk-daq-sw

  → see talk by Vakhtang in ITk TDAQ Session
- RCE

  → see talk by SLAC

Other Plans:

- Option to read connectivity from JSON file.
- Factor out runkey editor in dedicated microservice.



Result Viewer Microservice Status

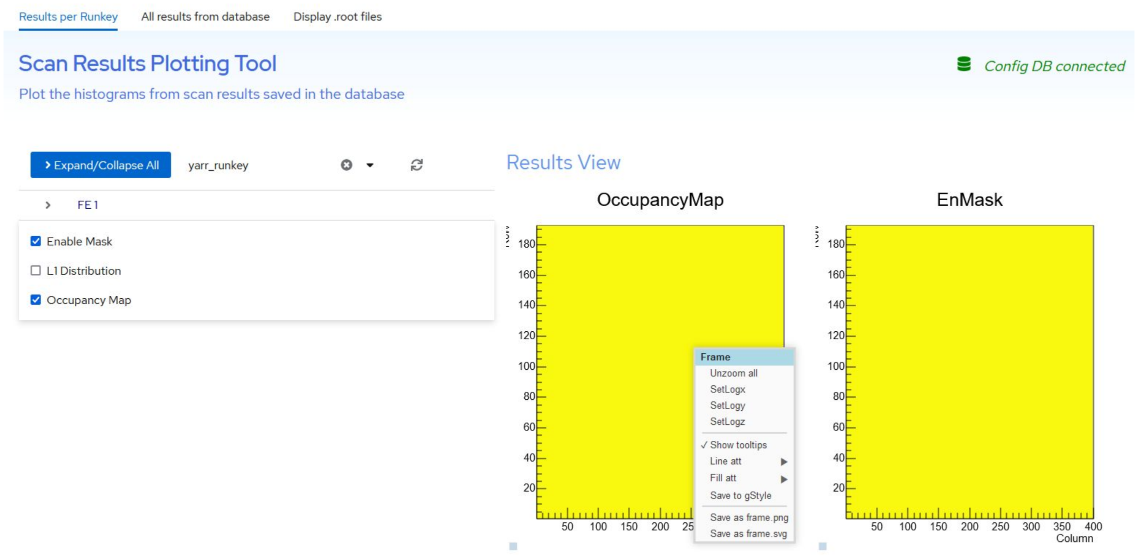
itk-demo-resultviewer

- Allows inspection of scan results (histograms in JSON format).
- Currently based on JSRoot

JSROOT 7.3.0

- Other plotting backends possible.
- Plots generated client-side and fully interactive.
- Results can be read from ConfigDb or uploaded as ROOT file.

- API based on slim Python container (no backend bindings).


Results per Runkey | All results from database | Display .root files

Scan Results Plotting Tool

Plot the histograms from scan results saved in the database

Config DB connected

Expand/Collapse All   yarr_runkey

FE 1

- [x] Enable Mask
- [ ] L1 Distribution
- [x] Occupancy Map

Results View

Resultviewer microservice showing emulated "digital scan" results.