

Run Control Plans

Goal

Provide a means for developers to run their applications in a run-control-like environment

- Only hard requirement for march-release
- Best effort towards engineering for reliability given time constraint
 - ↳ Implemented for $O(10)$ scale, not $O(100)$
 - ↳ Any incompatibilities between march-release and final release are written off as technical debt and will need to be adapted

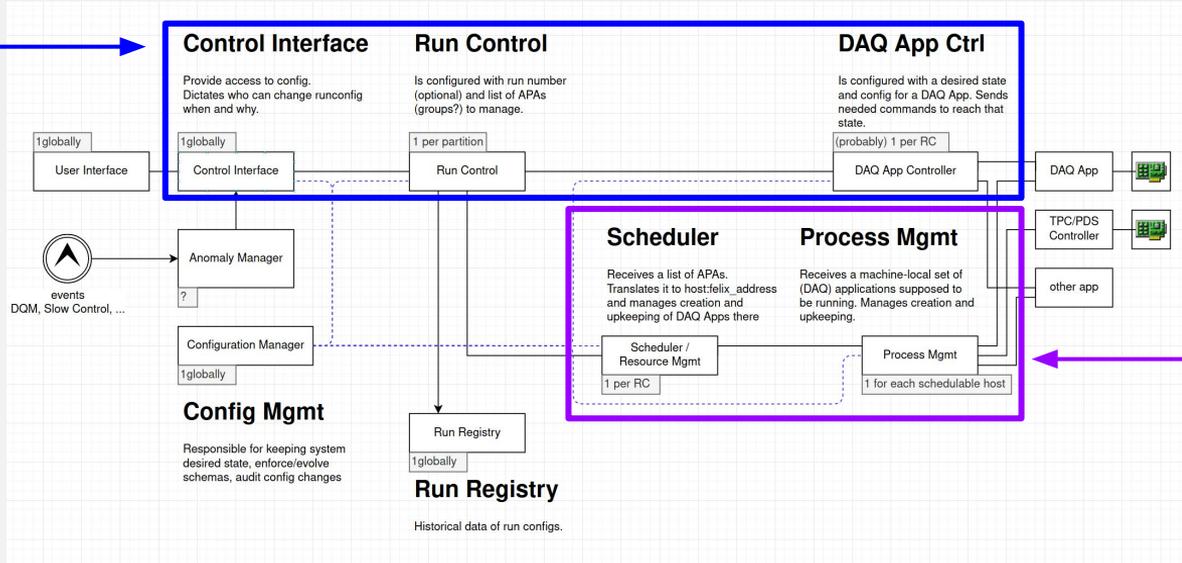
- First, the high overall experiment uptime goal requires DAQ to be stringently designed for reliability, fault tolerance, and redundancy, criteria that aim to reduce overall downtime.

todo list

1. Set up base technical stack (so we have something to run DAQ controllers on)
2. Make it easy to use for expected usage scenarios
 - a. Local deployment (your laptop)
 - b. Semi-professional deployment (your own cluster of pc's)
3. Re-implement control loops for DAQ Manager/Run Control/...
 - a. Steal code from last demo (but without any helper libs)
 - b. Bottom-up, DAQ Manager first

DONE

NOT DONE



Setup documentation

Initial intended usage scenarios

- Run locally
- Run on your own cluster of machines

How do we make this easy

- Ansible playbooks
- Infrastructure management absent, assumed to be bare metal
 - ↳ We could opt for terraform manifests (for CERN OpenStack?) as an inbetween
- Docker-compose for local one-machine setups
 - ↳ To remind devs to think distributed, all components can actually run on just one (logical) machine

```
→ DUNE-RC-RC git:(march-rc0) ✗ make docker.start
docker-compose -f docker/docker-compose.yml up
Creating docker_dune-rc-march-ru-1_1 ... done
Creating docker_dune-rc-march-gp-1_1 ... done
Creating docker_dune-rc-march-gp-2_1 ... done
Creating docker_dune-rc-march-ru-3_1 ... done
Creating docker_dune-rc-march-ru-2_1 ... done
Creating docker_dune-rc-march-gp-3_1 ... done
Attaching to docker_dune-rc-march-ru-1_1, docker_dune-rc-march-gp-1_1, docker_dune-rc-march-ru-2_1, docker_dune-rc-march-ru-3_1, docker_dune-rc-march-gp-3_1, docker_dune-rc-march-gp-2_1
```

```
→ DUNE-RC-RC git:(march-rc0) ✗ make docker.ansible
docker-compose -f docker/docker-compose.yml run --rm ansible -i /mnt/ansible/hosts.yaml /mnt/ansible/playbook.yaml
Creating docker_ansible_run ... done

PLAY [all] *****

TASK [Gathering Facts] *****
ok: [dune-rc-march-gp-2]
ok: [dune-rc-march-gp-1]
ok: [dune-rc-march-ru-1]
ok: [dune-rc-march-ru-2]
```

Process manager & scheduler

First iteration: supervisord + custom

→ Scrapped, supervisord not designed to run under a scheduler

Second iteration: nomad + consul

→ Very very close as well to our requirements

↳ Consul: raft implementation

↳ Nomad: process scheduler

* Largely supports our various labeling (except for anti-labels, which can be lived with for now)

→ Designed to work together

→ Solid track record

Apart from advantages that the previous demo has, I believe this to be a solid alternative.

DAQ App Manager & config

To instruct the DAQ state machines, configs are needed for each state transition

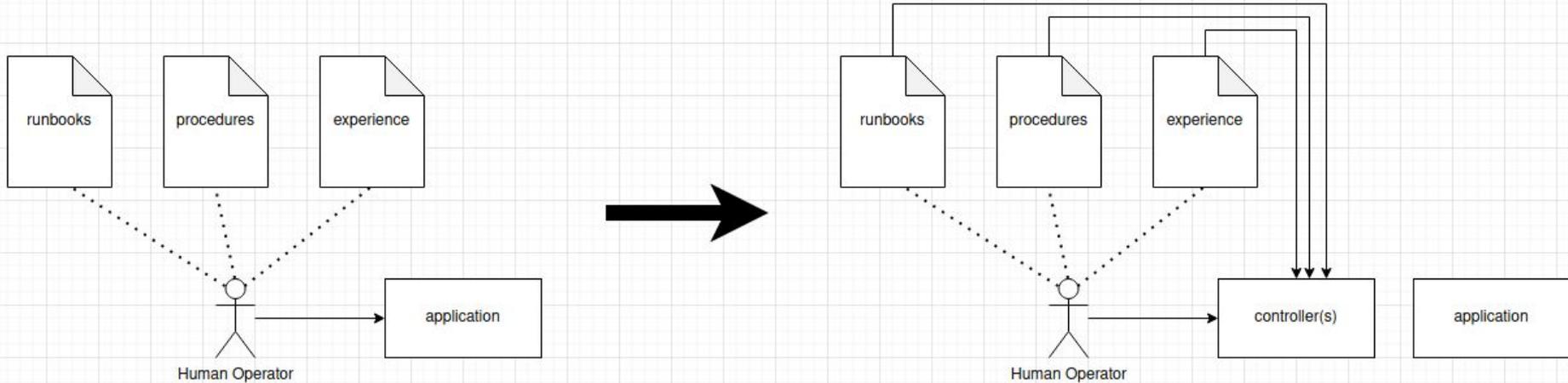
Couple options on where responsibility lies to supply configs

- Outside RC - configs are already present on schedulable locations
 - ↳ Only works for static setups, unsuitable for future system
- Scheduler - after scheduling decision made, run some 'setup' job
 - ↳ Either in-code, or (preferably) a callable service
- DAQ App manager
 - ↳ Makes sense since it is the piece of code actually needing the configs to make forward progress
 - ↳ Either in-code, or (preferably) a callable service

On control loops

Subtle aspect of run control design that makes this proposal superior to alternative technical stacks

Besides reliability arguments, there is also a large ops-related advantage, called the Operator Pattern.



Operators

Example: prometheus

→ Installed using 'kubectl apply https://.../manifest.yaml'

Overview

The Prometheus Operator provides [Kubernetes](#) native deployment and management of [Prometheus](#) and related monitoring components. The purpose of this project is to simplify and automate the configuration of a Prometheus based monitoring stack for Kubernetes clusters.

```
balerion ~ kubectl get prometheus
NAME                                VERSION  REPLICAS  AGE
prometheus-operator-prometheus     v2.18.2  1         193d
balerion ~
```

```
spec:
  alerting:
    alertmanagers:
      - apiVersion: v2
        name: prometheus-operator-alertmanager
        namespace: default
        pathPrefix: /
        port: web
    baseImage: quay.io/prometheus/prometheus
    enableAdminAPI: false
    externalUrl: https://prometheus.cms11test.juravenator.dev
```

```
balerion ~ kubectl get servicemonitor auto-devops-scrape-port-http -oyaml | yq .spec
{
  "endpoints": [
    {
      "port": "http"
    }
  ],
  "jobLabel": "component",
  "namespaceSelector": {
    "any": true
  },
  "selector": {
    "matchLabels": {
      "prometheus-operator-scrape-port": "http"
    }
  }
}
```