Combining cloud-native workflows with HTCondor jobs

A Kubernetes Operator for HTCondor

Clemens Lange (CERN)

HTCondor Workshop Autumn 2020
24th September 2020

Hello :-)

- I’m a CERN research physicist working on the CMS experiment
- Among other responsibilities, leading one the CMS new physics analysis groups
- Interested in making physics analysis workflows reproducible and reusable with the help of software images (Docker) and Kubernetes

Workflow “languages”

1. Capture software	2. Capture commands	3. Capture workflow
Individual analysis stages in an executable way (including all dependencies)	How to run the captured software?	How to connect the individual analysis steps?

Several tools under investigation and used by (smaller) groups

REANA CWL implementation (HEP-focussed)

Luigi

HTCondor DAGMan

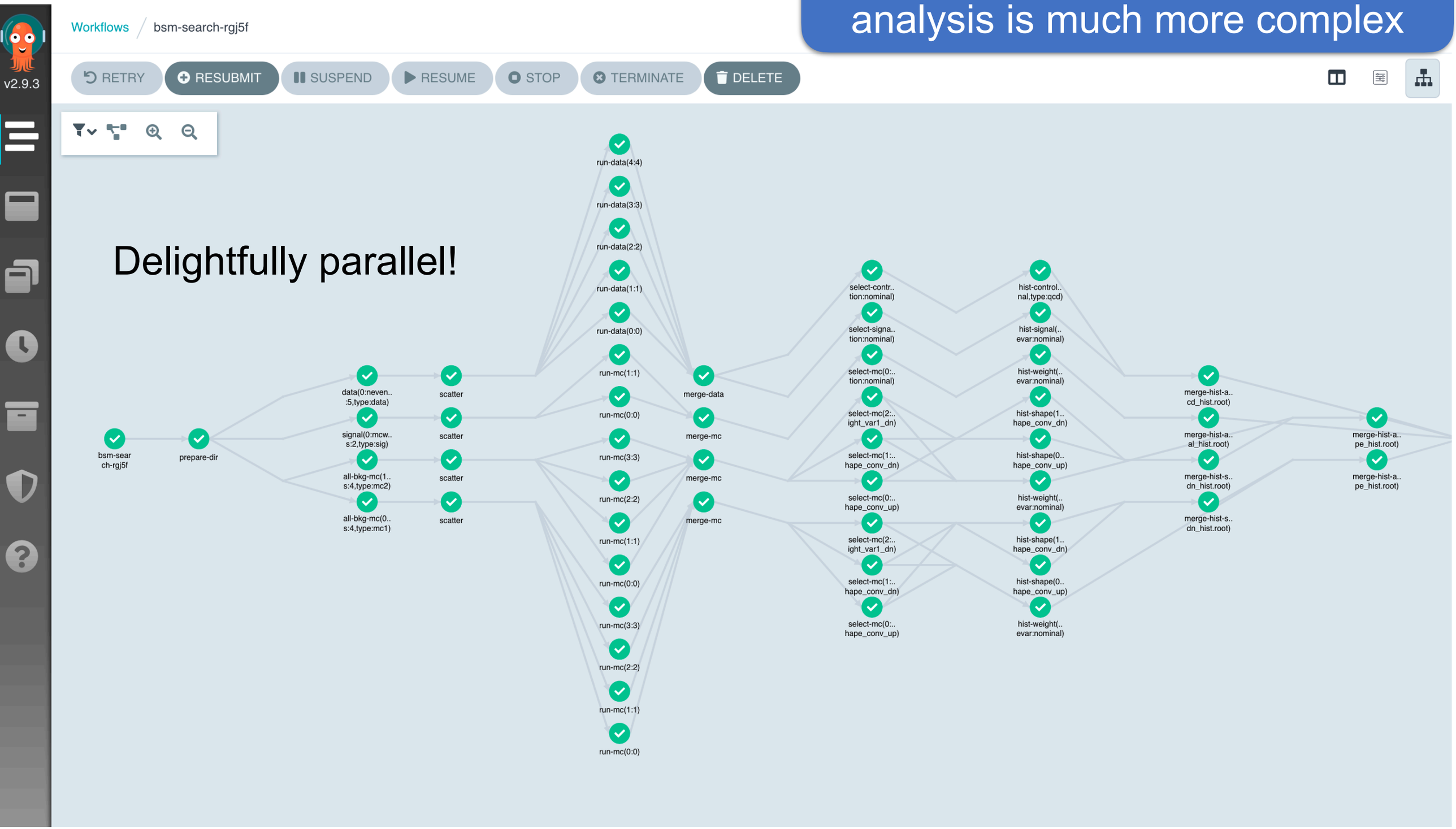
Yadage (HEP-specific)

Can we do it cloud-native (i.e. with Kubernetes)?

Argo Workflows

Example workflow

Demo only — a realistic physics analysis is much more complex



Based on https://github.com/reanahub/reana-demo-bsm-search

Life's unfair

I have a Kubernetes cluster, but with little resources

Too small to run realistic physics analysis workflows

Relative size not even to scale…

Can I use your cores?

My cluster

kubernetes

16 cores

The CERN batch cluster

HTCondor

High Throughput Computing

230,000 cores

970 TB memory

HTCondor operator

Idea: introduce HTCJob Custom Resource Definition (CRD)
➜ extend the Kubernetes API

```yaml
apiVersion:
apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: htcjobs.htc.cern.ch
spec:
  group: htc.cern.ch
  names:
    kind: HTCJob
    listKind: HTCJobList
    plural: htcjobs
    singular: htcjob
  scope: Namespaced
```

Mimic Kubernetes Jobs (reflect status Running/Failed/Succeeded)

```yaml
status:
  properties:
    active:
      type: integer
    failed:
      type: integer
    succeeded:
      type: integer
    clusterID:
      type: string
    jobIDs:
      items:
        type: string
      type: array
    uniqID:
      type: integer
```

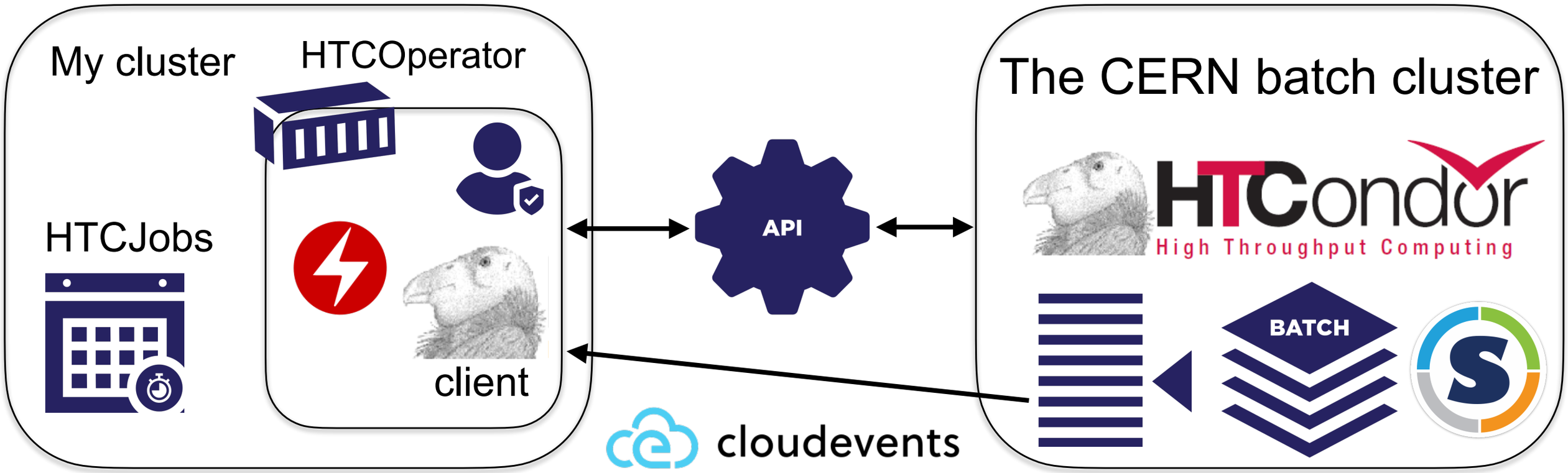
Operator acts on custom resource using a control loop:

- (Re)Submit jobs to HTCondor
- Reflect current job status in Kubernetes
- Take care of data transfer into cluster (spool jobs)

Work by **Tadas Bareikis** (Bioinformatics student at Vilnius University) and me

Implementation

- Operator SDK makes it easy to get a Kubernetes operator implemented und running
- Built Docker container that contains **HTCondor client** – also knows about **authentication** via Kerberos (using secrets)
- HTCondor Operator installed into this container – also translates image/job spec into **singularity** exec script
- Individual jobs can additionally notify operator via CloudEvents



Demo

Replay at https://www.youtube.com/watch?v=RIHT5MjQFqw

Concluding remarks

- Kubernetes CustomResourceDefinition combined with an Operator allows to manage and scale out jobs to HTCondor
  - Heavy use of HTCondor Python API
- “Operator” concept extremely powerful for this purpose
  - Can transparently be plugged into any Kubernetes-based tool

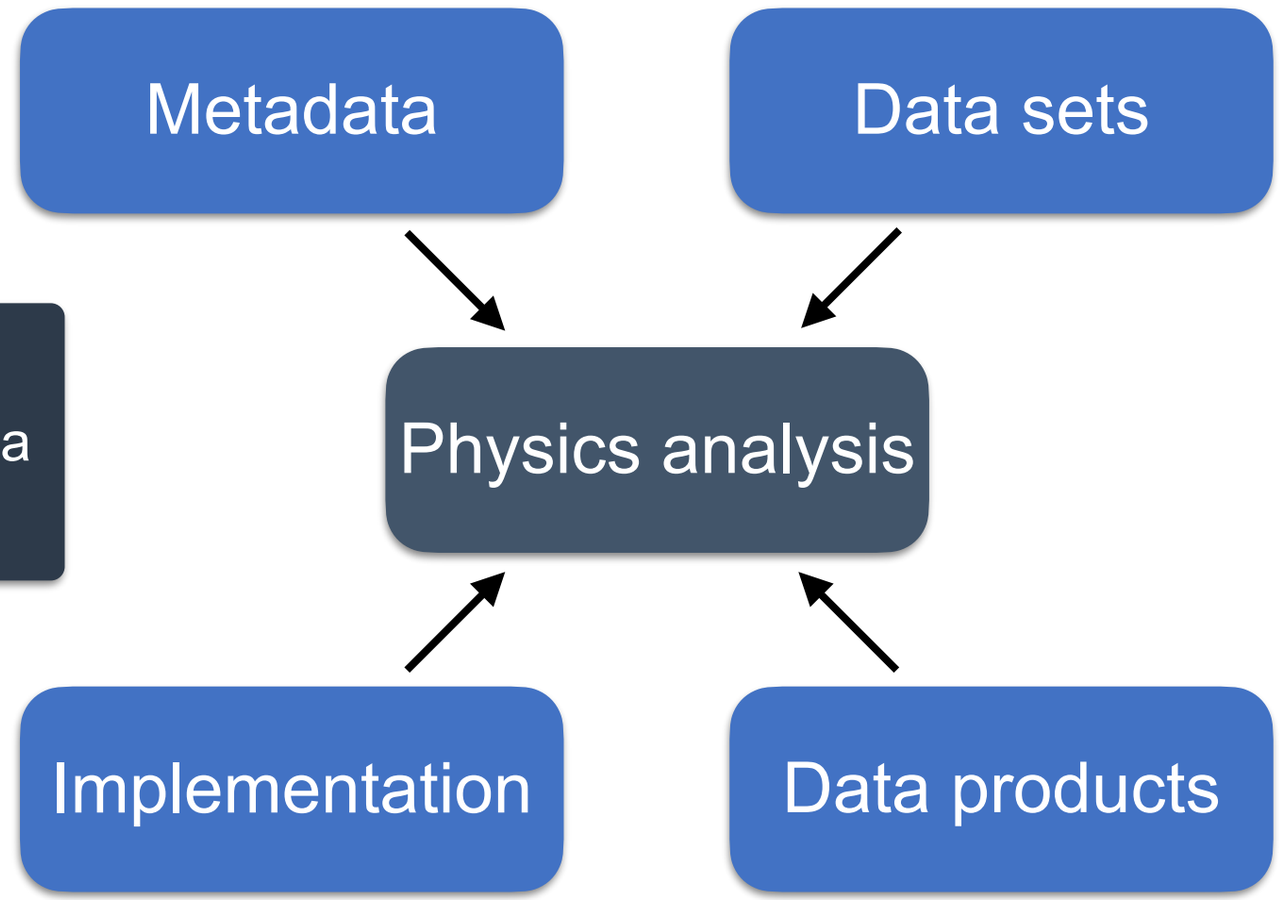
- Next steps:
  - Make HTCOperator more flexible/feature-complete
  - Apply same concept to also use the grid (WLCG)?


CERN

High-energy physics workflows

- Largest part of typical high-energy physics workflows is automated already
- Challenge: implement/use automated workflows for high-level physics analyses

These address a physics question, often resulting in a paper publication



- Software version control system and images are essential tools in this context

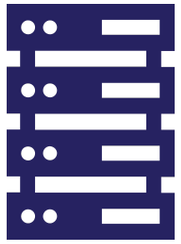
git + docker

Lots of inputs to be handled/keep track of!

My Kubernetes cluster

Provisioned with OpenStack with
CERN plugins/customisations

kubernetes v1.18.2



4 nodes à
4 cores w/ 8 GB RAM

Managed via GitOps
using Argo CD

300 GB S3 object storage
300 GB CephFS block storage

Secrets encrypted using SOPS
w/ Barbican modification
Deployed using KSOPS plugin
with Argo CD

Argo + HTCJobs

Argo can manage any kind of Kubernetes resources:

```
- name: generate-batch
  inputs:
    parameters:
    - name: type
    - name: nevents
    - name: njobs
  resource:
    action: create
    successCondition: status.succeeded == {{inputs.parameters.njobs}}
    failureCondition: status.failed > 0
    manifest: |
      apiVersion: htc.cern.ch/v1alpha1
      kind: HTCJob
```

➜ Can move the long-running steps to HTCondor!

Mind: no need to use Argo, can use bare Kubernetes HTCJob resource