Stimela & Dask & Kubernauts: Cloud Workflow Management for Radio Astronomy Pipelines

Oleg Smirnov (Rhodes University & SARAO)

+**Simon Perkins** (Rhodes), **Jonathan Kenyon** (Rhodes & SARAO), **Landman Bester** (SARAO & Rhodes), **Sphesihle Makhathini** (Wits U. & Rhodes)

https://ratt.center

RATT

Centre For Radio Astronomy Techniques & Technologies • Rhodes University

NRF National Research Foundation

SARAO South African Radio Astronomy Observatory

Pipelines In The Cloud?

- Vera C Rubin Observatory (LSST) Science Platform on Google cloud:
    - 100 PB of data
    - "[Cloud] not cheap but great value for money"
    - "From a technical perspective, use of commodity computing is a no-brainer"

      - Frossie Economou @ADASS 2021, Rubin Science Platform on Google: the story so far
- "SKA/AWS Call For Proposals For AstroCompute In The Cloud", 2015
    - Two reports/papers in 2017 (Processing public pulsar astronomy data in the Amazon Cloud, Calibration of LOFAR data on the cloud) but no follow-up
    - 2021: Development of a high throughput cloud-based data pipeline for 21cm cosmology
    - **If you use the cloud to just spin up an old-school compute node, then all you have is an inconveniently remote compute node with a rapidly ticking $$$ meter**
- The ML/data science community uses the cloud heavily in novel ways
- 2022: nobody is doing SKA precursor radio imaging in the cloud
    - why not?

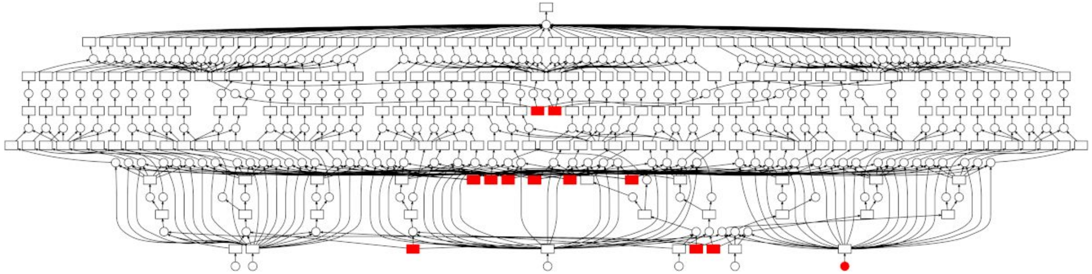
Awkward Chickens / Messy Eggs

- Data ingress/egress (mitigated by an AWS/GCE/... data centre near you)
- Awkward data formats
  - The Measurement Set is such a spectacularly successful piece of 1990s technology that we're still locked into it in 2022
  - Ill-adapted for parallel processing or even basic multithreading
  - Needs a filesystem volume (e.g. EBS), not suitable for cheaper cloud storage such as S3
- Awkward software stacks
  - The best MeerKAT images today were produced by heterogenous pipelines, using a combination of software packages from different groups
  - No e2e solution except CASA, but this doesn't serve our needs
  - **Where is the new software going to come from if you can't get astronomers to test it?**
  - **How do you get them to test it until you have a complete cloud-based solution?**
- Awkward and complex "thick/thin" workflows
  - And awkward orchestration
- Very difficult to control costs, particularly in a development environment!

https://dask.org

dask

- Numpy-like arrays and dataframes implementing parallel and/or out-of-core computations
- Construct computation graph, then (lazily) evaluate it on demand
- Very popular in the PyData/Big Data/ML community
    - e.g. Pangeo (https://pangeo.io/) geoscience
- Achieves HPC performance comparable to traditional C/C++/MPI
- Since the intrinsic computation parallelism is encoded in the graph, clever schedulers can take advantage of this to distribute the compute



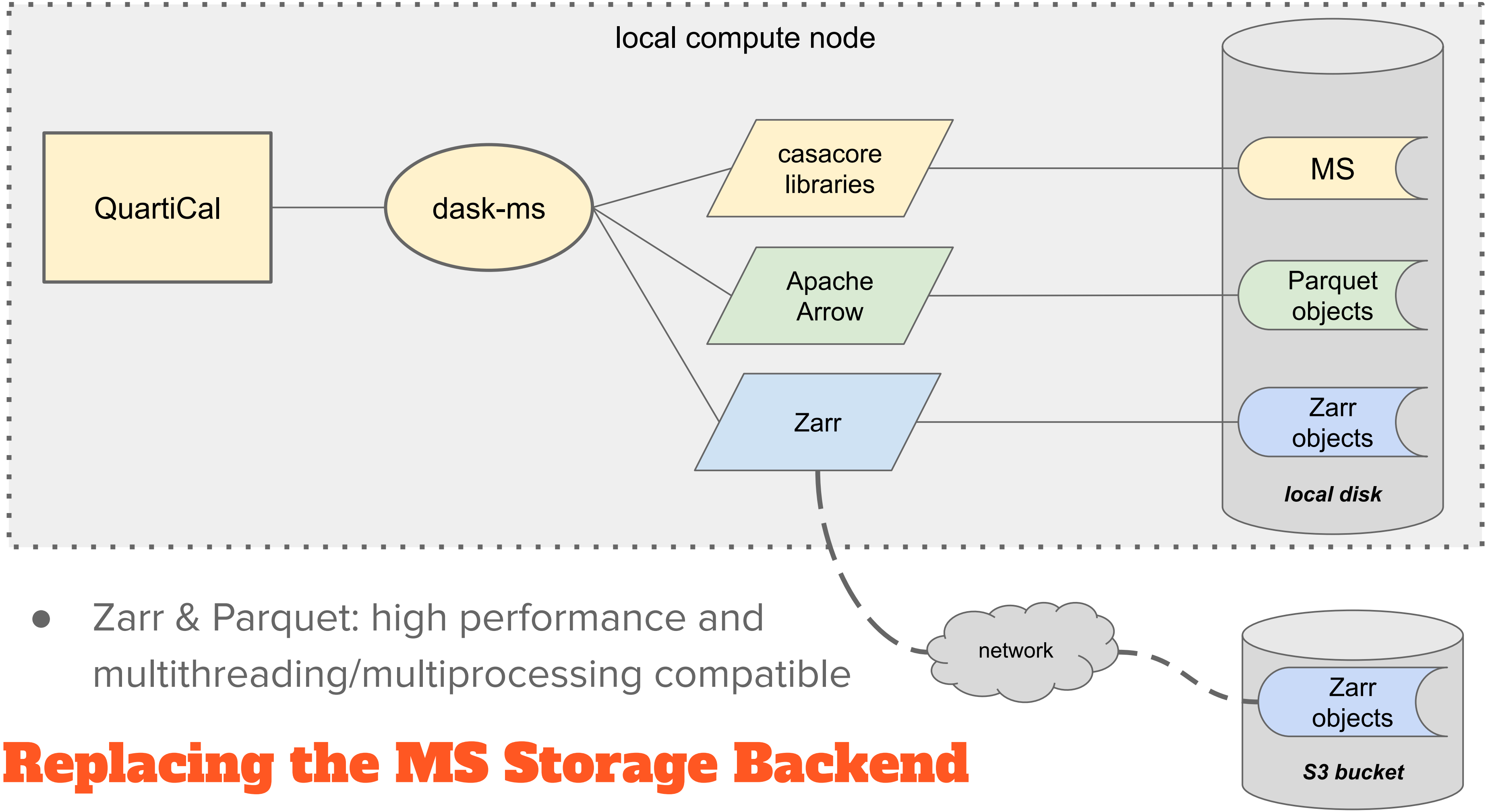
dask-ms

https://github.com/ratt-ru/dask-ms

- Originally started by S. Perkins as a **dask** interface to the Measurement Set
- Attach to an MS, access it as dask arrays, and write all your computation in terms of dask array operations
- Backend for a number of RATT & SARAO products:
  - **QuartiCal** (J. Kenyon) https://github.com/ratt-ru/QuartiCal: calibration suite
  - **pfb-clean** (L. Bester) https://github.com/ratt-ru/pfb-clean: imager
  - **tricolour** (B. Hugo, S. Perkins) https://github.com/ratt-ru/tricolour: flagger
  - **xova** (S. Perkins) https://github.com/ratt-ru/xova: visibility data averaging, including BDA
  - **crystalball** (S. Perkins) https://github.com/caracal-pipeline/crystalball: DFT-based model predict
  - **shadeMS** (O. Smirnov, I. Heywood) https://github.com/ratt-ru/shadeMS: plotting & visualization

  - *a.k.a. most of the moving parts for a selfcal pipeline…*

"Dask-ms is the best documentation for the Measurement Set!" – T. Molteno

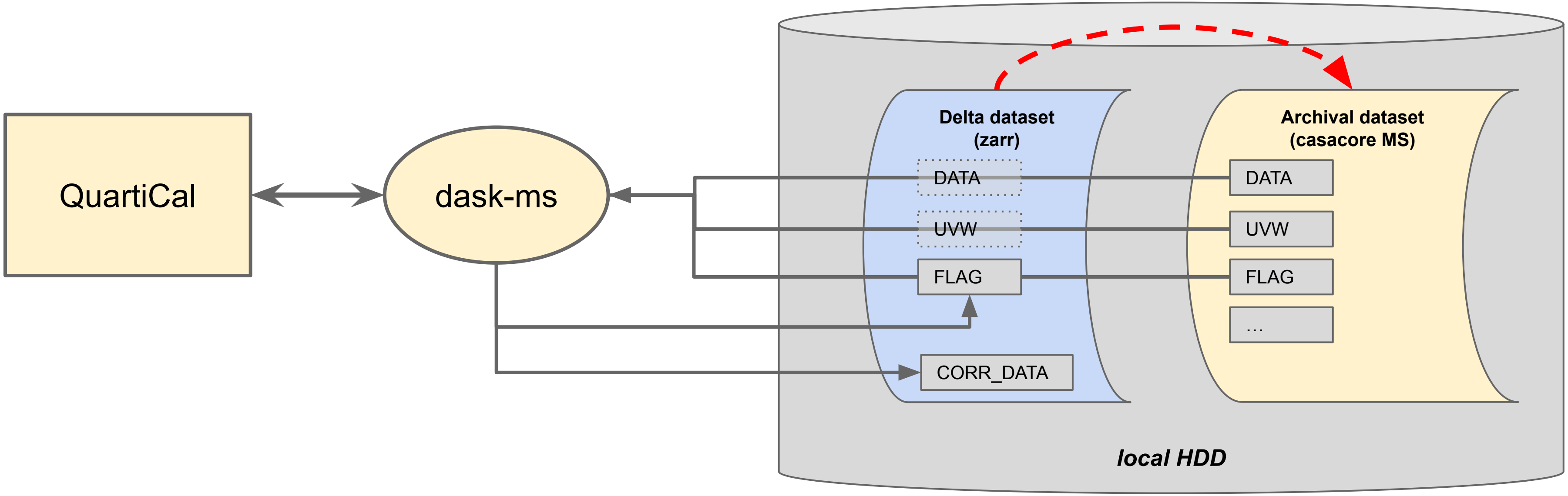


- Zarr & Parquet: high performance and multithreading/multiprocessing compatible

Replacing the MS Storage Backend

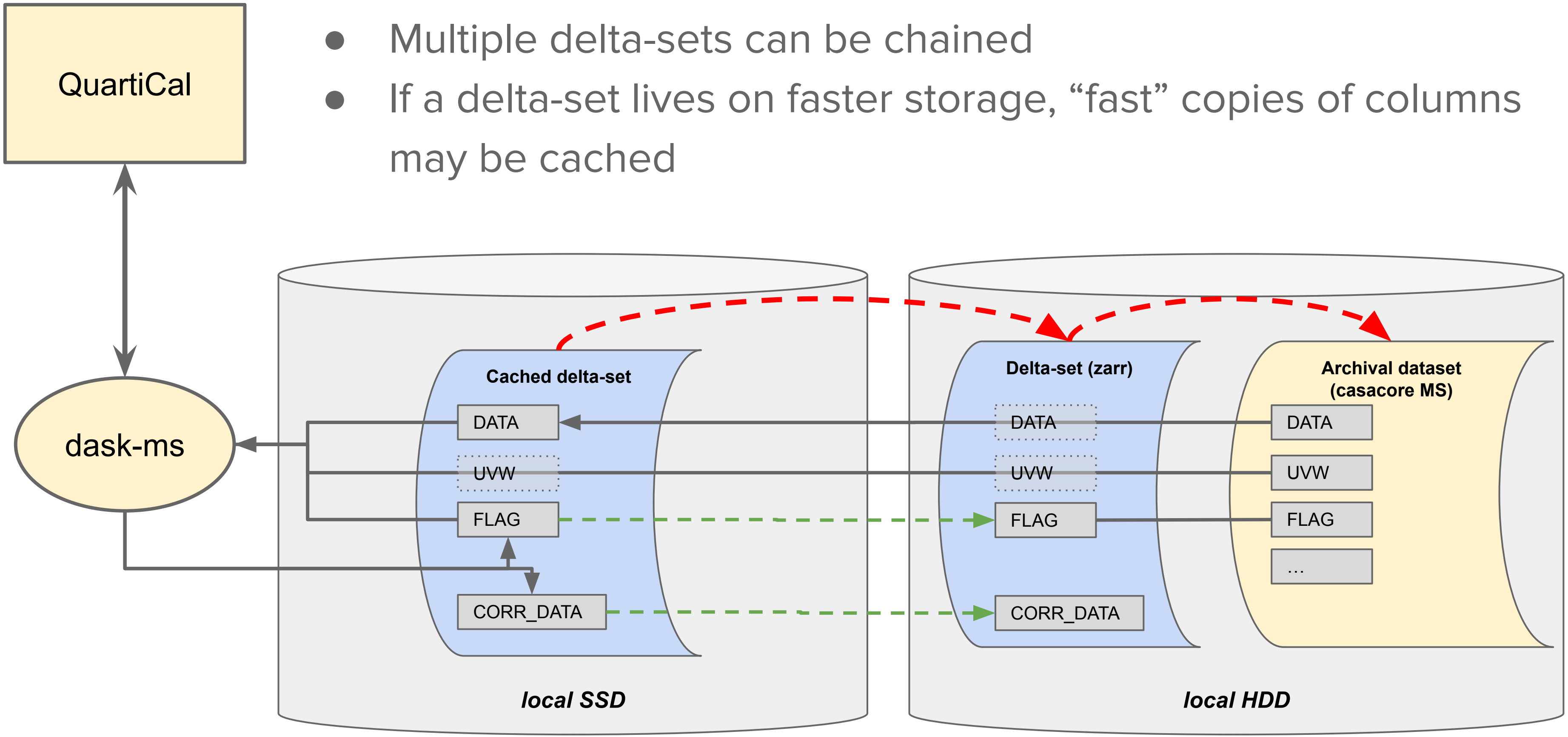
Dataset Versioning (RSN)

- Archival dataset lives in **readonly** form somewhere (e.g. as casacore MS)
- Delta dataset (delta-set) has a logical link to the archival dataset
  - Only stores new columns (*deltas*) that have been written out (as zarr objects)
  - Unchanged columns pulled directly and transparently from archival dataset



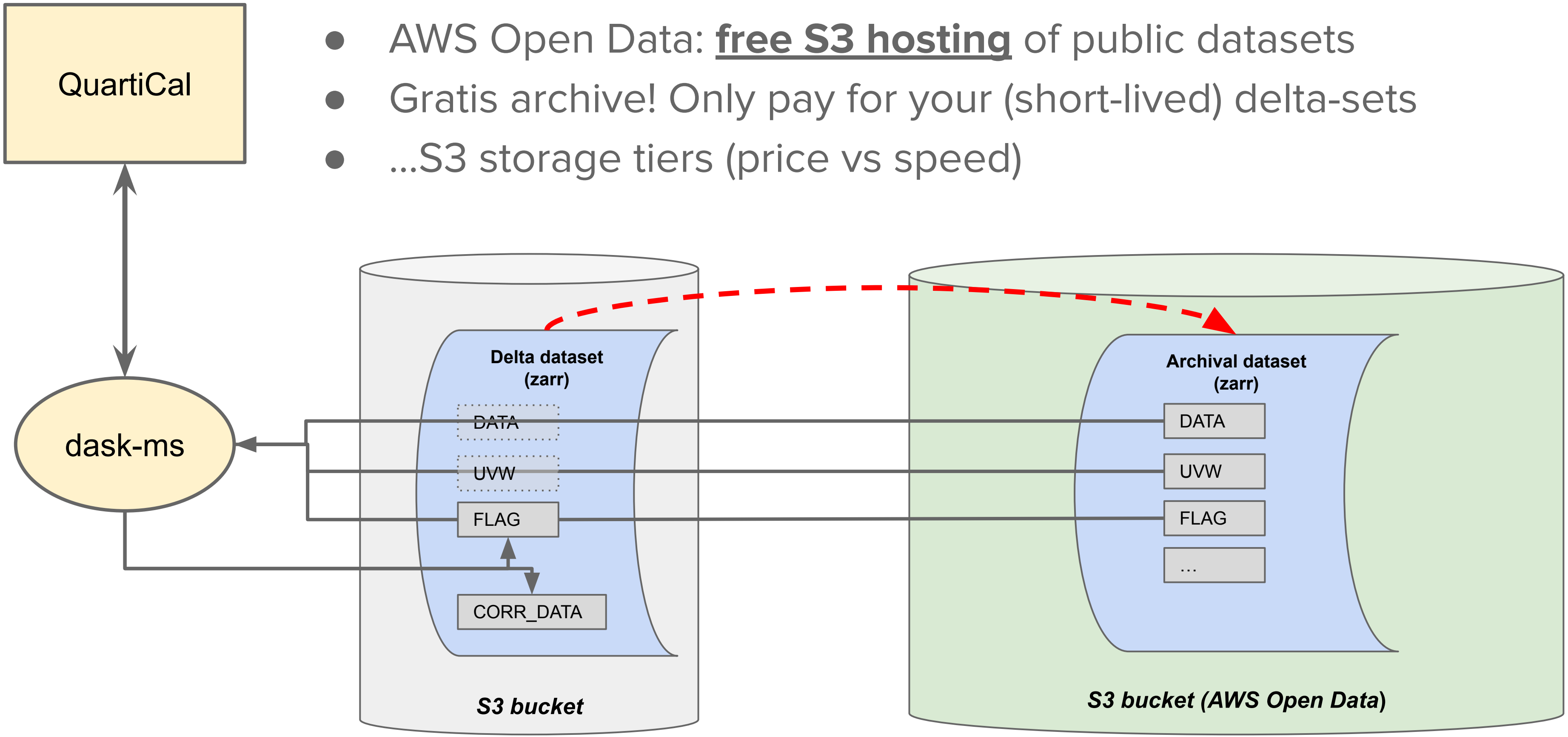
Dataset Caching (RSN)

- Multiple delta-sets can be chained
- If a delta-set lives on faster storage, “fast” copies of columns may be cached



AWS Data Deployment

- AWS Open Data: **free S3 hosting** of public datasets
- Gratis archive! Only pay for your (short-lived) delta-sets
- …S3 storage tiers (price vs speed)



Software Stack Transition

- dask-ms based applications don’t care if they’re running on a local node with an old-school casacore MS, or on the cloud with an S3 backend
- In local mode, QuartiCal etc. are being put to routine use today, reducing MeerKAT and VLA data
    - by astronomers i.e. testers!
- We can concentrate on developing cloud-based pipelines, while the components are being tested & validated extensively by others
    - “Most of the moving parts for a selfcal pipeline” are available, so…

kubernetes aka K8s

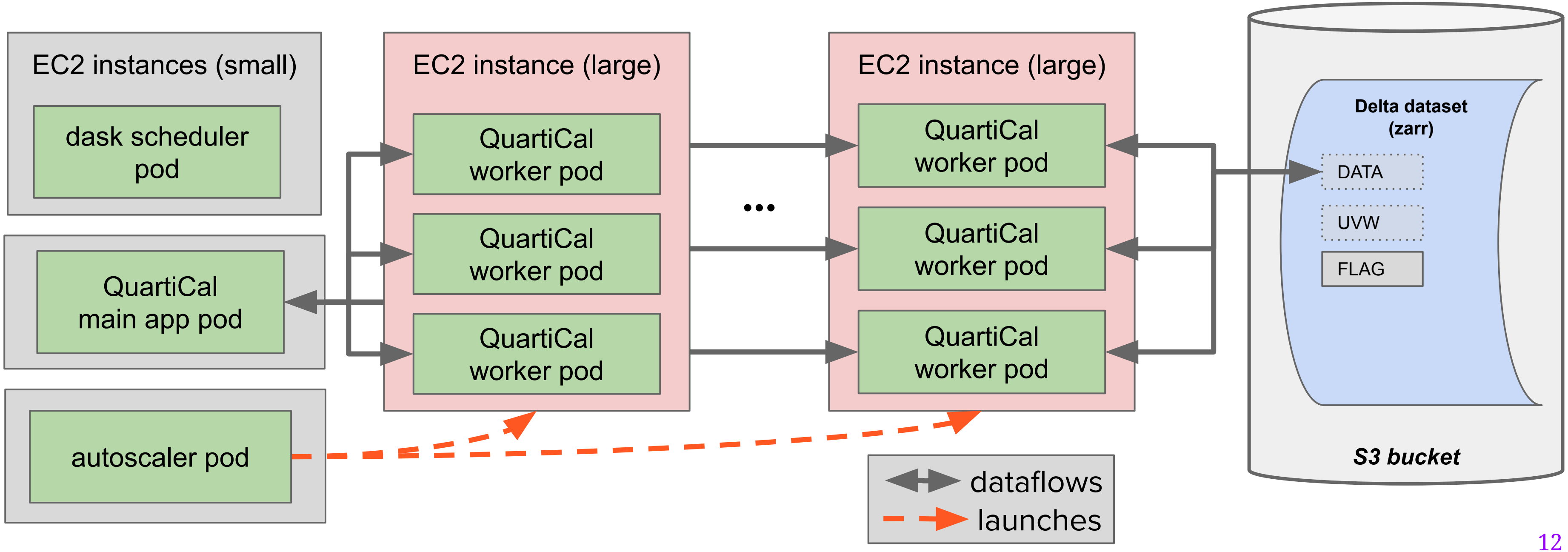
https://kubernetes.io/

- Open source container orchestration system created by Google
  - Now maintained by Cloud Native Computing Foundation (CNCF)
  - Industry standard
  - (see Rubin Science Platform, slide 1)
- Provides services for scheduling ***pods*** (containers) on compute nodes to create highly available, scalable, distributed applications
- Implementations available at all levels:
  - **microk8s** on Linux, runs a local k8s cluster
  - Run k8s cluster in the cloud:
    - Amazon Elastic Kubernetes Service (**EKS**)
    - Google Kubernetes Engine (**GKE**)
    - Azure Kubernetes (**AKS**)
  - Same RESTful API on all implementations

Dask Kubernetes

https://kubernetes.dask.org

- **dask-kubernetes** schedules dask applications as a set of pods
- An **autoscaler** pod launches more EC2 instances to meet application pod requirements, and scales them down once the application finishes
- Resolves the thick/thin conundrum

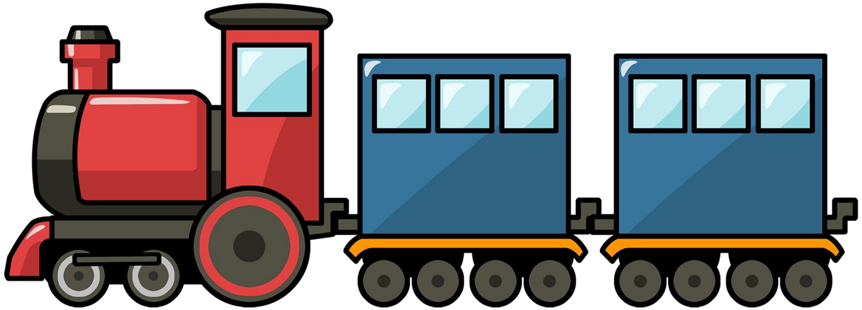


Kubernauts

HashiCorp Terraform

- Kubernauts: RATT/SARAO project to support cloud-based pipeline infrastructure
    - a set of scripts and (developing) best practices
- **Infrastructure as Code** (IAC)
- Define desired AWS + K8s state in Hashicorp Configuration Language (HCL)
- Terraform resolves **drift** between desired and actual state
- Desired state maintained in github
- Observed state maintained in **Terraform Cloud**
- Pushes to github trigger drift resolution within Terraform Cloud

Stimela: All aboard



- Stimela originally developed by S. Makhathini (2018 PhD @Rhodes) as a Docker-based application scripting framework
- Provides Docker images ("cabs") for all major astronomy software packages, and a Python API to chain cabs together into "recipes"
- A recipe is a sequence of steps, each step is a cab (application)
- Stimela takes care of running containers and passing/validating parameters
- Stimela2: next-gen rewrite

Stimela2: One YaML To Rule Them All

- Recipes specified using a concise YaML syntax
- Composable: recipes can be nested as steps within recipes
- Conditionals, loops, scatter/gather
- Formula language for parameter evaluation

```yaml
# actual recipe steps
steps:
  restoreflags:
    info: restore an original flag version before starting calibration
    cab: flagman
    params:
      mode: restore
      name: =recipe.init-flags
    skip: =not IFSET(recipe.init-flags)

  flagsummary:
    info: report initial flagging statistics
    cab: flagsummary
    params:
      spw: =recipe.casa-spw

  image-0:
    info: construct initial model image using the DATA column
    _use: lib.steps.wsclean.image
    params:
      prefix: =recipe.image-prefix
      column: DATA
      auto-threshold: 3
      auto-mask: 7
    skip: =recipe.init-model.enable
    tags: [never, init_model]

  selfcal-d1:
    info: first round of delay selfcal
    recipe: selfcal-cc-delay
    params:
      label: "{info.suffix}"
      predict:
        # use init selfcal model if set, otherwise use output of image-0
        prefix: =IF(recipe.init-model.enable, recipe.init-model.prefix, steps.image-0.prefix)
        nchan: =IF(recipe.init-model.enable, recipe.init-model.nchan, steps.upsample-0.nchan)
        size: =IF(recipe.init-model.enable, recipe.init-model.size, steps.image0-0.size)
        scale: =IF(recipe.init-model.enable, recipe.init-model.scale, steps.image-0.scale)

  selfcal-d2:
    info: second round of delay selfcal
    recipe: selfcal-cc-delay
    params:
      label: "{info.suffix}"
      predict:
         prefix: =previous.output-model-prefix
```

Composability

Top level recipe:

```
_include:
  - lib-137.yaml
```

...

followed by:

```
selfcal-d2:
  info: second round of delay selfcal
  recipe: selfcal-cc-delay
  params:
    label: "{info.suffix}"
    predict:
      prefix: =previous.output-model-prefix
```

lib-137.yaml:

```
selfcal-cc-delay:
  info: runs a basic step of delay selfcal (predict-solve-image)

  inputs:
    label:
      dtype: str
      info: label of this selfcal step, e.g. "1", "1p", etc.
    ms:
      aliases: [(flagman).ms, (flagsummary).ms, (wsclean).ms, (cubical).data.ms, (quartical).input_ms.path]
      info: MS name
    flags:
      restore:
        aliases: [restoreflags.name]
        info: flagversion to restore to before starting. Set to none/empty to skip flag restore.
        default: =root.init-flags
        required: false
      save:
        aliases: [saveflags.name]
        info: flagversion to save to after calibration. Set to none/empty to skip flag save.
        default: 'selfcal_{current.label}'
        required: false
    predict:
      # pull in standard image parameters
      _use: lib.params.imaging.base
      _scrub: '*.dtype'
      enable:
        info: enables the predict step
        dtype: bool
        default: true
```

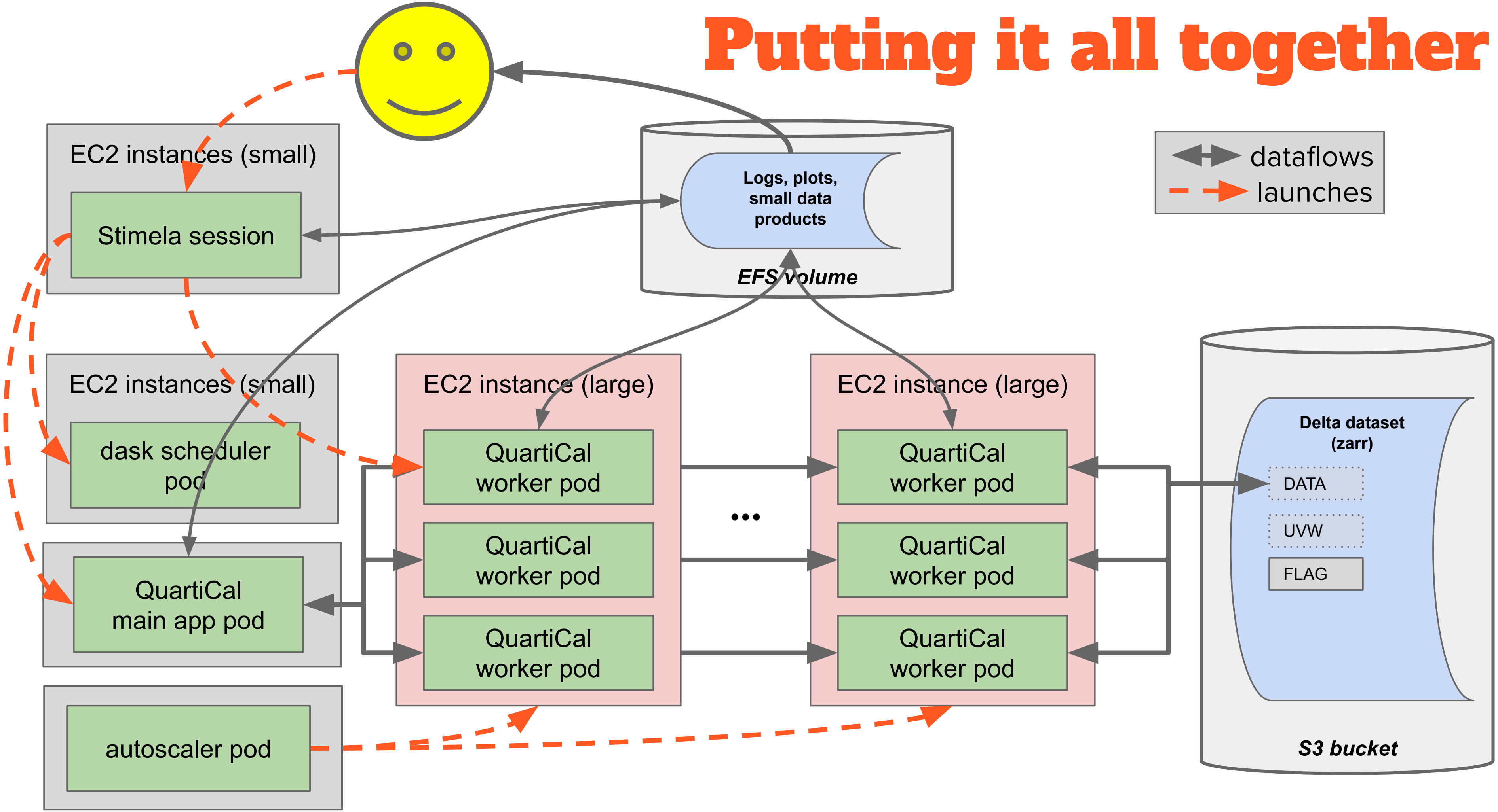
...

followed by:

```
steps:
  restoreflags:
    info: restores an original flag version before starting calibration
    cab: flagman
    params:
      mode: restore
    skip: =not IFSET(recipe.flags.restore)

  predict:
```

Putting it all together



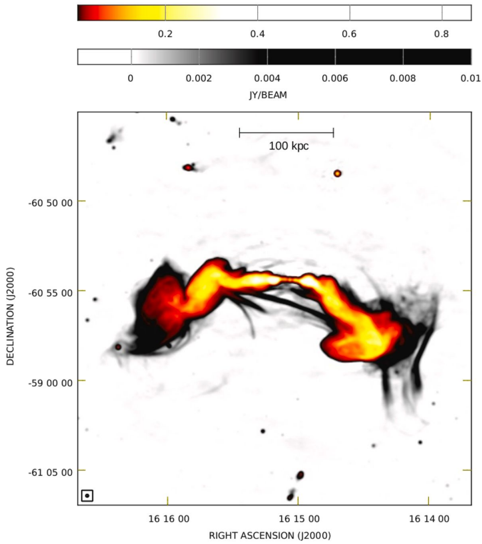
Some (Future) Killer Apps

- Calibration & imaging challenge/contest
  - Raw data of MeerKAT ESO-137 observation from Ramatsoku et al. 2020 will be made available via AWS Open Data on S3
  - Along with an e2e Stimela recipe, from raw data to final image
  - Run for yourself (on AWS, or on your own hardware) and try to do better?
  - "Recipe contest" not "imaging contest"
- Towards full reproducibility
  - Don't just publish a paper, publish data and recipes!
- In conclusion: passengers welcome (please mind, train is under construction)

https://github.com/caracal-pipeline/stimela2

A&A 636, L1 (2020)

Letter to the Editor



Collimated synchrotron threads linking the radio lobes of ESO 137-006

M. Ramatsoku[1,2], M. Murgia[2], V. Vacca[2], P. Serra[2], S. Makhathini[1], F. Govoni[2], O. Smirnov[1,3], L. A. L. Andati[1], E. de Blok[7,5,6], G. I. G. Józsa[3,1,4], P. Kamphuis[9], D. Kleiner[2], F. M. Maccagni[2], D. Cs. Molnár[2], A. J. T. Ramaila[3], K. Thorat[8] and S. V. White[1,4]