ETH zürich

KOF

Workflow and Workload Management with Airflow & Ray

Stéphane Bisinger

29th January 2026, Zürich

About me

KOF

Stéphane Bisinger

- Software Engineer
- Worked at Homegate, siroop, and many more
- Web development: Backend, Frontend, Fullstack
- Infrastructure: Serverless Architectures, Kubernetes
- Extreme Programming, TDD, Continuous Delivery
- At KOF since 2022
    - Data import
    - Coding support and architecture
    - Infrastructure development and maintenance

KOF Swiss Economic Institute

- The KOF sees itself as a **bridge builder** between **research and society** and as an **exchange platform for economists**
- Fulfills so-called **national tasks** that have been assigned to it by the **federal government**
- Some of the main products from KOF are:
  - Quarterly updated macroeconomic forecasts for the Swiss economy. (R)
  - Business surveys of Swiss companies. (R)
  - The Nowcasting Lab https://nowcastinglab.org (MATLAB, Python)
  - …and many other Surveys, Forecasts, and Indicators https://kof.ethz.ch

Table of Content

KOF

VM-based architecture

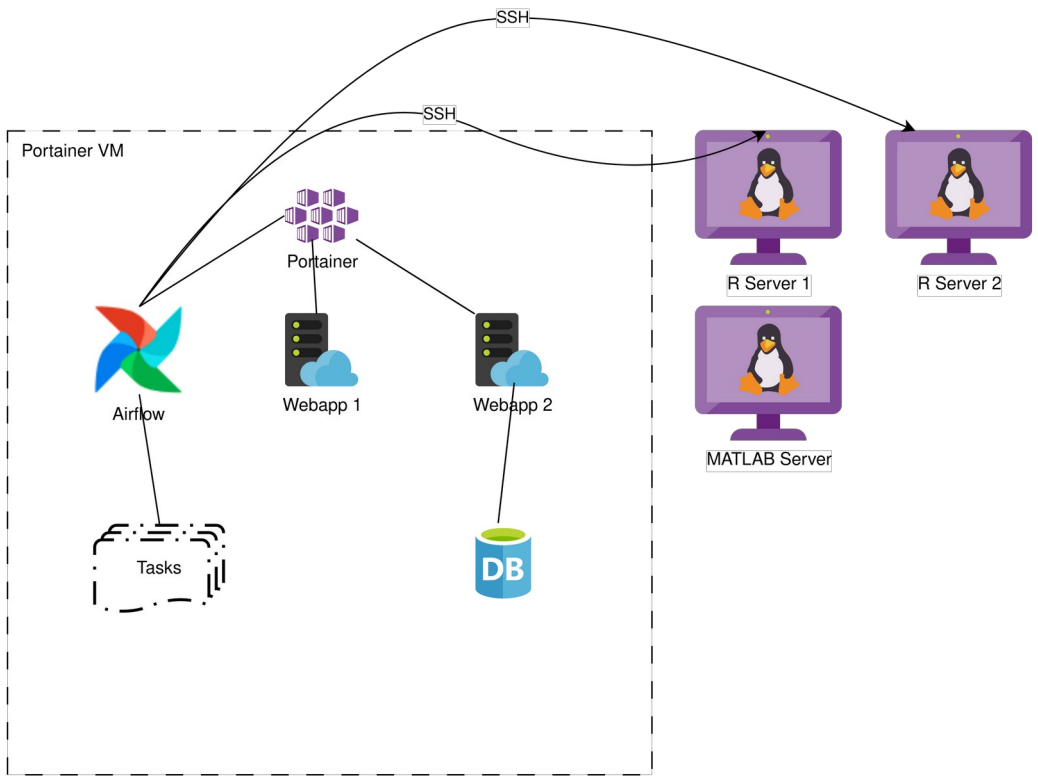
Typical workloads at KOF

KOF

- Data import
  - Read an API or website to check if new data we are interested in is available
  - Download the data and parse the format
  - Save it in our internal database
- Seasonal Adjustment
  - When data is updated, calculate and store the seasonally adjusted variant of a time series
- Data aggregations
- Run prediction models
  - Load the required data
  - Calculate the prediction with Fancy Models
  - Store the output of the model

VM-based architecture

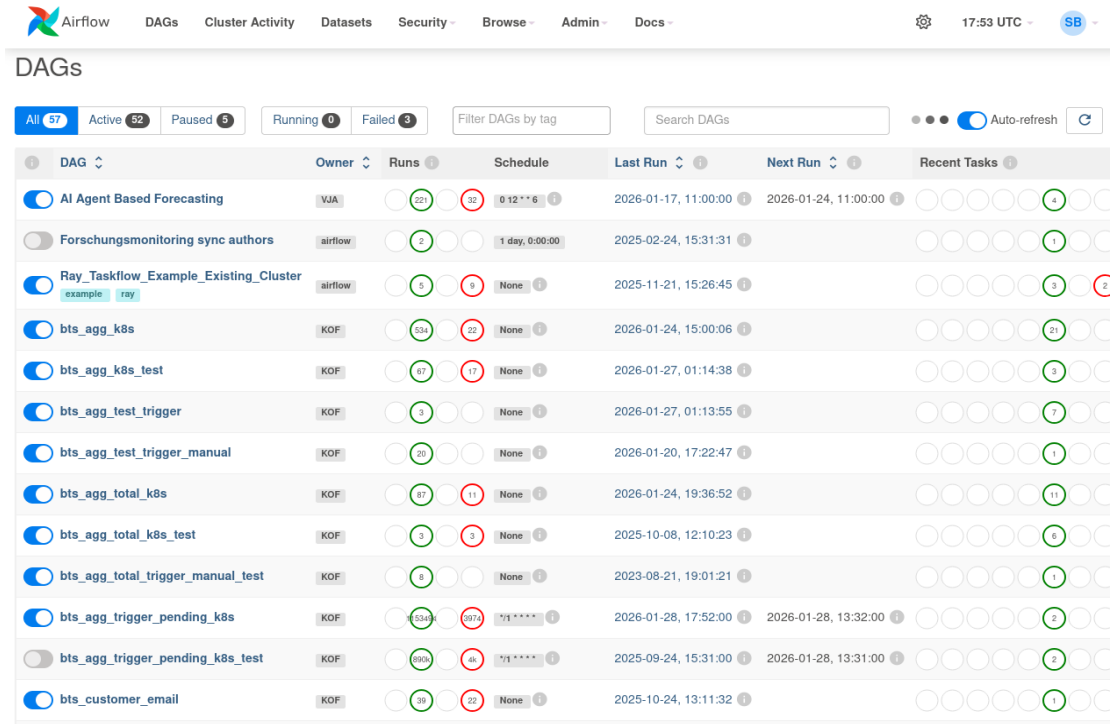
KOF

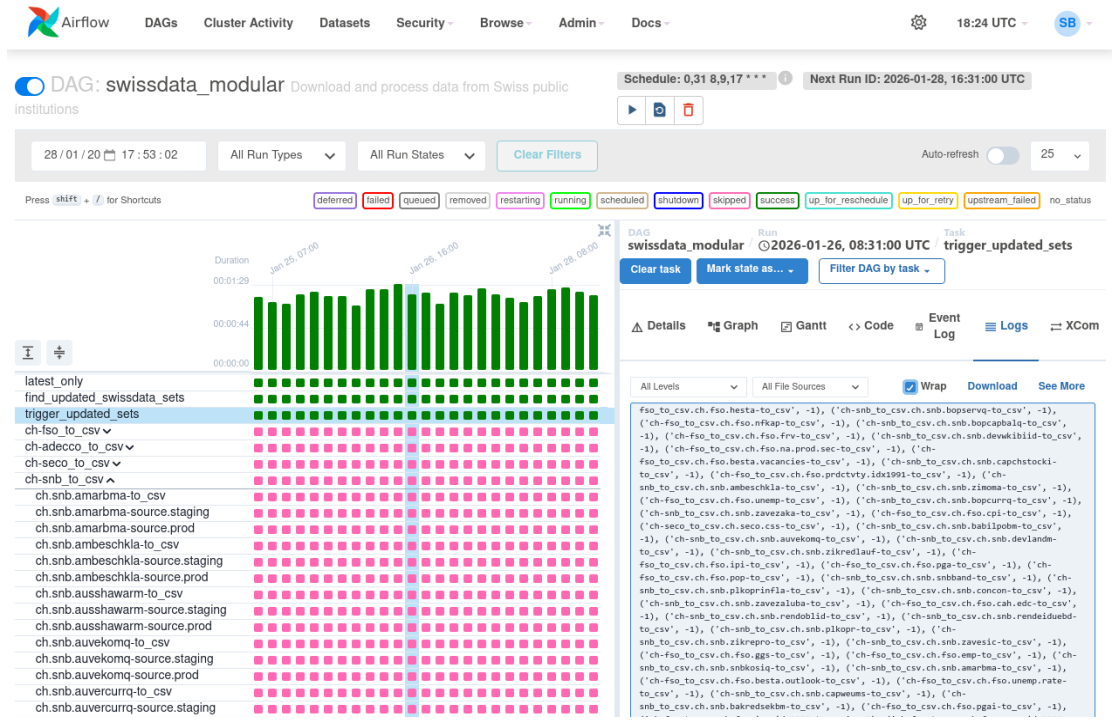


- There's a few VMs dedicated to running R and MATLAB programs
- A VM would host the docker host, with portainer as a UI
- Various web apps and other applications would run here as containers
- Airflow also runs as a container
  - It will launch simple R tasks as a standalone container
  - It will connect to other specific VMs for complex tasks requiring more computing resources

Airflow

KOF





Airflow is cron on steroids

Run tasks on a schedule. Keep track of runs and their status and logs.

- Interval-based scheduling
- Event-based scheduling
- Custom timetables
- ...and more!

Running R tasks

KOF

Dockerfile

```
FROM rocker/r-ver:3.6.0 as deps

RUN apt-get update && apt-get install -y \
                libpq-dev \
                libcurl4-openssl-dev \
                libxml2-dev \
                ...

RUN mkdir /tasks
COPY tasks/ /tasks/

# Installed internal packages used by Airflow scripts
RUN install2.r --repos registry.kof.ethz.ch kofpkg1 kofpkg2 kofpkg3


FROM deps

COPY .Rprofile /home/kofbaser/.Rprofile
COPY entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh

WORKDIR /home/kofbaser

ENTRYPOINT [ "/entrypoint.sh" ]
```

entrypoint.sh

```
#! /bin/bash

if [[ ! -f /tasks/$1.R ]]; then
    echo not a task file, doing whatever
    exec $@
else
    echo running /tasks/$1.R ${@: 2}
    Rscript /tasks/$1.R ${@: 2}
fi
```

Running R tasks

KOF

swissdata.py

```python
from datetime import datetime
from airflow import DAG
from airflow.models import Variable
from airflow.operators.docker_operator import DockerOperator

default_args = {
        'start_date': datetime(2017, 3, 20)
}

dag = DAG('swissdata', description='Run swissdata',
          schedule_interval = '0,31 8,9,17 * * *',
          default_args = default_args, catchup=False)

pg_password = Variable.get("docker_pg_password")

with dag:
    source_swissdata = DockerOperator(
        task_id = 'source_swissdata',
        image = 'registry.ethz.ch/kof/kofdocker/kofbase:0.2.0',
        user='kofbaser',
        volumes = [
            'swissdata-output:/swissdata',
            'kofbase-tasks:/tasks'
        ],
        docker_url = 'unix:///var/run/docker.sock',
        environment = {
            'PG_PASSWORD': pg_password,
            'PG_HOST': 'db.kof.ethz.ch'
        },
        command = 'source_swissdata'
    )

```

business_tendency_survey.py

```python
import datetime
from airflow import DAG
from airflow.contrib.operators.ssh_operator import SSHOperator



default_args = {
  'start_date': datetime.datetime(2019, 11, 12)
}

dag = DAG(
  'bts_agg_prod',
  description='Runs business tendency survey aggregation [PROD]',
  schedule_interval = None,
  default_args = default_args,
  catchup = False
)

with dag:
  aggregate = SSHOperator(
    task_id='aggregate',
    ssh_conn_id='rcalc_prod',
    command="""Rscript -e 'library(kofbts); runAggregation( \
      "{{dag_run.conf.survey_name}}", \
      {{dag_run.conf.run_of_analysis}}, \
      {{dag_run.conf.year}}, \
      {{dag_run.conf.month}})'"""
  )

```

VM-based architecture: Pros and Cons

KOF

Pros

- Computations move away from personal computers, onto dedicated infrastructure that doesn’t require a user sitting in front of the computer
- With Airflow, regularity and timing can be automated.
- Automatic notification in case of errors
- Log history of runs

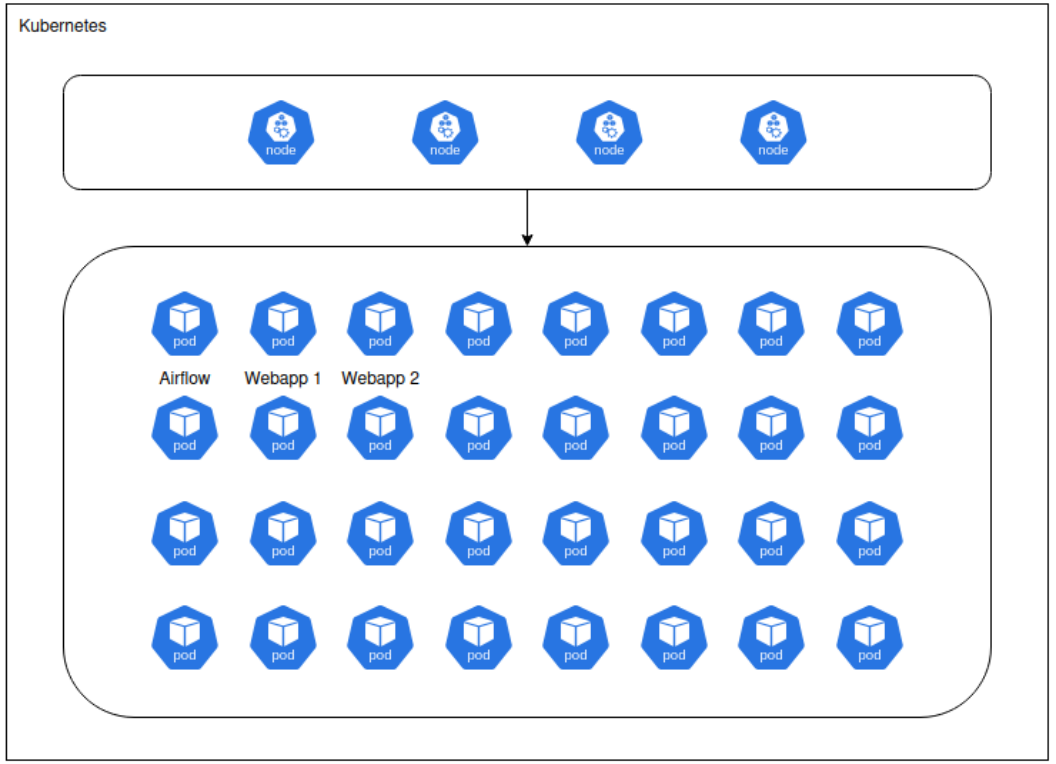
Cons

- VMs are always running, whether they are needed or not
- Servers need to be installed with all necessary dependencies (and managed)
- Favors big scripts that do it all, rather than modular and parametrizable tasks
- Parallelism is not really encouraged
- Infrastructure management is fragmented and complex

Kubernetes

Kubernetes architecture

KOF



- The architecture is simply a kubernetes cluster: no special cases, Kubernetes manages workloads, routing, scaling, etc.
- The VMs are now just Kubernetes Nodes
- Everything is containerized: no specialized servers that require specific libraries or tooling.
- Airflow tasks are now run as pods

Running R tasks – now with renv!

KOF

Dockerfile

```
ARG R_VERSION=4.3.1
FROM docker.io/rocker/r-ver:${R_VERSION}

# libssl-dev must be installed first to work in older R versions. It might get
# downgraded to the correct version with the follow-up command.
RUN apt-get update \
  && apt-get install -y libssl-dev \
  && apt-get install -y \
  libpq-dev \
  libcurl4-openssl-dev \
  libxml2-dev \
  ... \
  && rm -rf /var/lib/apt/lists/*

# Only renv needed
RUN install2.r renv --repos https://packagemanager.posit.co/cran/latest

# renv volume must be mounted to /renv in container for renv cache
ENV RENV_PATHS_PREFIX_AUTO=TRUE
ENV RENV_PATHS_CACHE=/renv/cache

COPY entrypoint.sh /entrypoint.sh
COPY run_task.R /
COPY Rprofile /home/kofbaser/.Rprofile
RUN chmod +x /entrypoint.sh

WORKDIR /home/kofbaser

ENTRYPOINT [ "/entrypoint.sh" ]

```

entrypoint.sh

```
#! /bin/bash
if [ "$1" = "run_task" ]; then
  echo running task $3 in package $2
  Rscript /run_task.R $2 $3
elif [[ ! -f /tasks/$1/run.R ]]; then
  echo not a task file, doing whatever
  exec $@
else
  echo running /tasks/$1/run.R
  cd /tasks/$1
  R -e "renv::restore(); source('run.R')"
fi
```

run_task.R

```
args <- commandArgs(trailingOnly = TRUE)
package_name <- args[[1]]
command <- args[[2]]

o <- renv::refresh()
renv::install(package_name)
tasks_dir <- system.file("tasks", package = package_name)
# packages can have their own tasks in inst/tasks/<task>.R
script <- paste0(tasks_dir, "/", command, ".R")
message(sprintf("Executing script: %s", script))
sys.source(script, envir = topenv())
```

Running R tasks

KOF

swissdata.py

```python
set_name = "ch.fso.frv"
environment = "prod"
db_hosts = {"prod": "db.kof.ethz.ch"}
pg_password = Variable.get("prod_db_password")

with dag:
    source_swissdata = KubernetesPodOperator(
        namespace="airflow-worker",
        image="registry.kof.ethz.ch/images/kofrenv:latest",
        task_id=f"{set_name}-source.{environment}",
        in_cluster=True,
        get_logs=True,
        env_vars=k8s_env_vars(
            {
                "PG_PASSWORD": pg_password,
                "PG_HOST": db_host(environment),
                "SWISSDATA_SET": set_name,
            }
        ),
        volume_mounts=k8s_volume_mounts({"renv-cache": "/renv"}),
        volumes=k8s_volumes({"renv-cache": "/renv"}),
        arguments=["run_task", "kofpkg1", "source_swissdata"],
    )
```

business_tendency_survey.py

```python
pg_password = Variable.get("prod_db_password")

with dag:
    aggregate = KubernetesPodOperator(
        namespace="airflow-worker",
        task_id="aggregate",
        image="registry.kof.ethz.ch/images/kofrenv:4.5.0-1",
        in_cluster=True,
        get_logs=True,
        env_vars=k8s_env_vars(
            {
                "pg_password": pg_password,
                "nr_cores": "4",
                "survey": "{{dag_run.conf.survey_name}}",
                "run": "{{dag_run.conf.run_of_analysis}}",
                "year": "{{dag_run.conf.year}}",
                "month": "{{dag_run.conf.month}}",
            }
        ),
        volume_mounts=k8s_volume_mounts({"renv-cache": "/renv"}),
        volumes=k8s_volumes({"renv-cache": "/renv"}),
        container_resources=k8s.V1ResourceRequirements(
            requests={"memory": "24Gi", "cpu": "4000m"},
            limits={"memory": "48Gi", "cpu": "12000m"},
        ),
        arguments=["run_task", "kof_business_tendency_survey", "aggregate"],
    )
```

Kubernetes architecture: Pros and Cons

KOF

Pros

- Infrastructure is now standardized and managed through Helm Charts (Infrastructure-as-Code)
- Everything can be scaled automatically on-demand
- Monitoring everything is much simpler
- All the R-code is in the R package, not spread over multiple projects.
- Usage is simplified
- Encourages splitting into smaller modular tasks
- Can run any type of workload with the same architecture

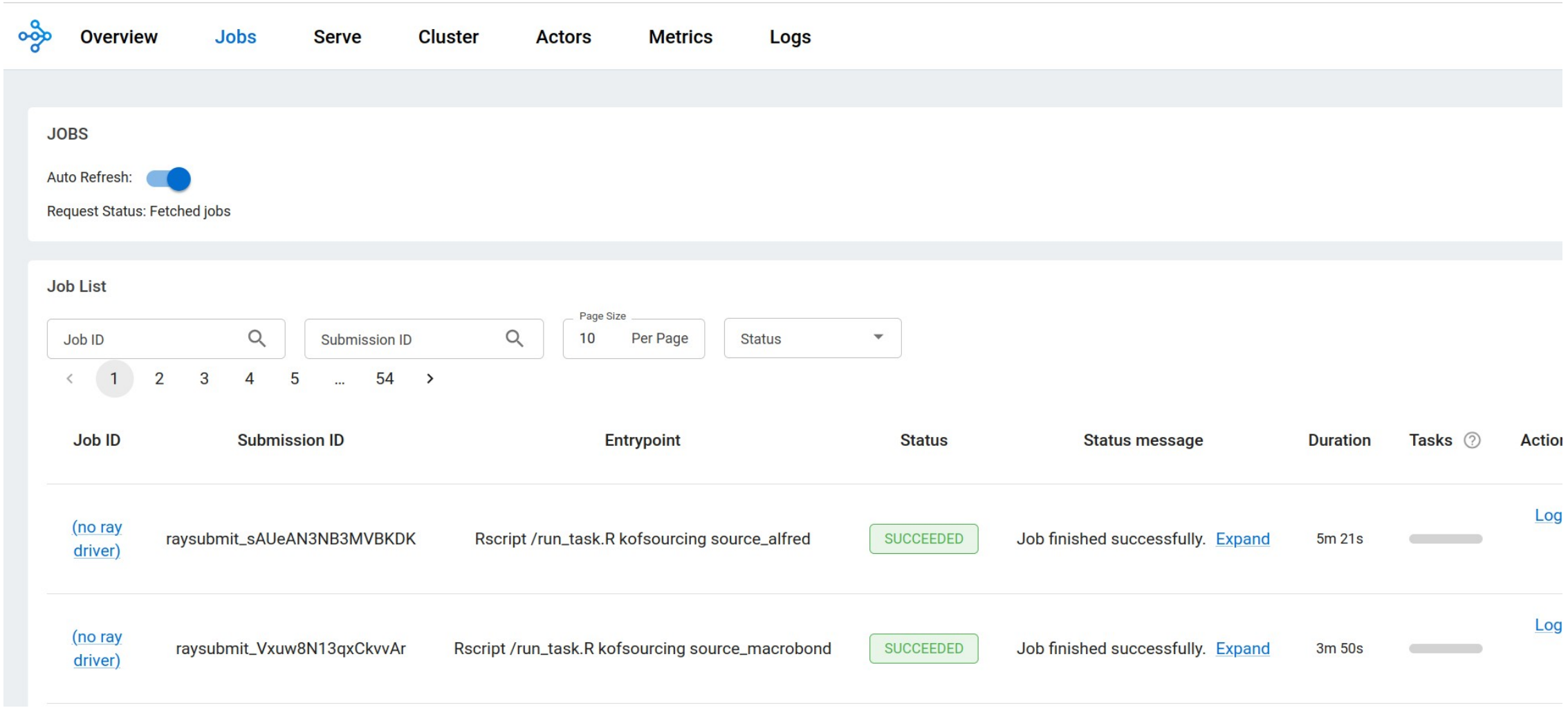
Cons

- Kubernetes management is a complex skill that requires time to acquire
- Resource limits must be set to avoid greedy tasks taking over all the available resources
- Small tasks have a relatively high overhead

Ray

Ray

KOF


Overview
Jobs
Serve
Cluster
Actors
Metrics
Logs
JOBS
Auto Refresh:
Request Status: Fetched jobs
Job List
Job ID
Submission ID
Page Size
10
Per Page
Status
1
2
3
4
5
...
54
Job ID
Submission ID
Entrypoint
Status
Status message
Duration
Tasks
Actio
(no ray driver)
raysubmit_sAUeAN3NB3MVBKDK
Rscript /run_task.R kofsourcing source_alfred
SUCCEEDED
Job finished successfully. Expand
5m 21s
Log
(no ray driver)
raysubmit_Vxuw8N13qxCkvvAr
Rscript /run_task.R kofsourcing source_macrobond
SUCCEEDED
Job finished successfully. Expand
3m 50s
Log

Ray

KOF

Ray is an open source unified framework for scaling AI and Python applications.

It provides a simple, universal API for building distributed applications that can scale from a laptop to a cluster.

What's Ray?

Ray simplifies distributed computing by providing:

- **Scalable compute primitives:** Tasks and actors for painless parallel programming
- **Specialized AI libraries:** Tools for common ML workloads like data processing, model training, hyperparameter tuning, and model serving
- **Unified resource management:** Seamless scaling from laptop to cloud with automatic resource handling

Ray cluster definition

KOF

ray.yaml

```yaml
head:
  rayVersion: {{ $vars.RAY_VERSION }}
  enableInTreeAutoscaling: true
  autoscalerOptions:
    idleTimeoutSeconds: 300
    imagePullPolicy: Always
worker:
  image:
    repository: registry.kof.ethz.ch/images/ray-kofrenv
    tag: {{ $tag }}
  rayStartParams:
    resources: '"{\"r\": 1}"'
  replicas: 0
  minReplicas: 0
  maxReplicas: 8
  volumes:
    - name: renv-cache
      persistentVolumeClaim:
        claimName: renv-cache
  volumeMounts:
    - mountPath: /renv
      name: renv-cache
```

- Ray allows us to define worker nodes with specific capabilities
- Normally intended for nodes with GPUs available rather than CPU. But we use it to add different programming languages!
- Workers are scaled automatically as tasks come in, and the same worker is reused until all tasks are done. Pay the overhead only once!
- Workers are based off the usual images + the ray image.

Executing an Airflow task on Ray

KOF

macrobond_ray.py

```python
from ray_provider.operators import SubmitRayJob


with dag:
    source_macrobond = SubmitRayJob(
        task_id="source_macrobond",
        conn_id="kuberay",
        entrypoint="Rscript /run_task.R kofpkg1 source_macrobond",
        runtime_env={
            "env_vars": {
                "https_proxy": "http://proxy.ethz.ch:3128",
                "DB_HOST": "db.kof.ethz.ch",
                "DB_PASSWORD": "{{var.value.prod_pg_password}}",
                "MACROBOND_API_PASSWORD": "{{var.value.mb_api_password}}",
            }
        },
        num_cpus=0,
        memory=0,
        resources={
            "r": 1,
        },
        outlets=[Dataset("macrobond")],
    )
```

- The Ray provider makes the integration with Airflow very simple to achieve
- By defining our own entrypoint, we’re able to execute arbitrary things in whatever programming language => R!
- Airflow still tracks task execution and status.

Alternative approach: rpy

KOF

ray_helper.py

```python
import ray


def ray_remote_r_task(*, env_vars=None):
    if env_vars is None:
        env_vars = {}

    @ray.remote(resources={"r": 1}, env_vars=env_vars)
    def run_r_task(package, task):
        import rpy2.robjects as robjects
        from rpy2.robjects.packages import importr

        r = robjects.r
        r.options(**{"renv.consent": True})
        base = importr("base")
        renv = importr("renv")

        renv.install(package, prompt=False)
        tasks_dir = base.system_file("tasks", package=package)

        script = base.paste0(tasks_dir, "/", task, ".R")
        print(f"Executing script: {script}")
        base.sys_source(script, envir=r.topenv())

    return run_r_task
```

- It is possible to go through the python sdk to execute R code
- However: who wants to write R code in python?
- Due to some technicality, this code is actually not reusable and must be copy/pasted in each Airflow task

Ray: Pros and Cons (so far)

KOF

Pros

- Automatic scaling of workers based on the amount of tasks
- Reduces overhead of starting/stopping containers
- Great capabilities to further distribute ML/AI models
- Flexible enough to allow arbitrary code execution – can be used with any language

Cons and open challenges

- Logs are not in Airflow anymore, need to be retrieved via specific logging infrastructure
- Keeping worker images up-to-date can be a challenge
- With great power, comes great complexity

Questions / Demo

ETH Zürich
KOF Institut

Leonhardstrasse 21
8092 Zürich
www.kof.ethz.ch

Insights empowering decisions