Aviso

Enabling notifications for meteorological workflows across HPC and Cloud data centres

C. Iacopino, J. Hawkes, T. Quintino, B. Raoult

ECMWF

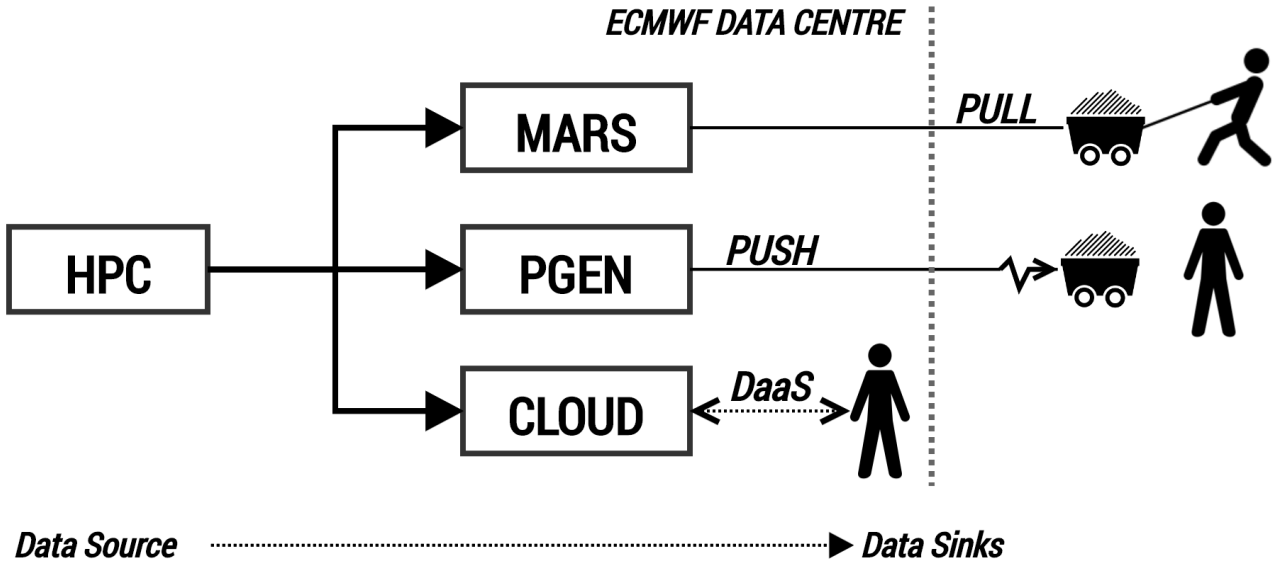
claudio.iacopino@ecmwf.int

ECMWF virtual workshop: Weather and climate in the cloud - 09-02-2021

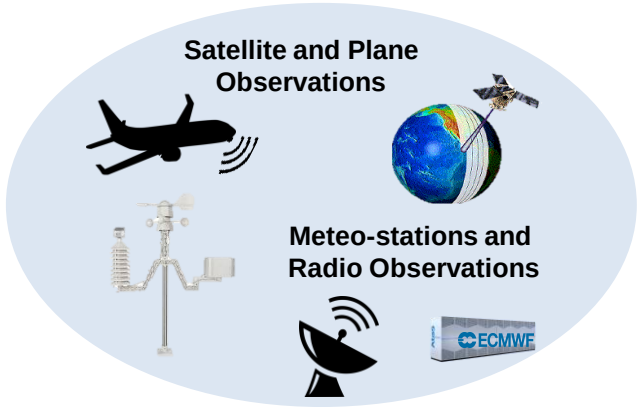
ECMWF

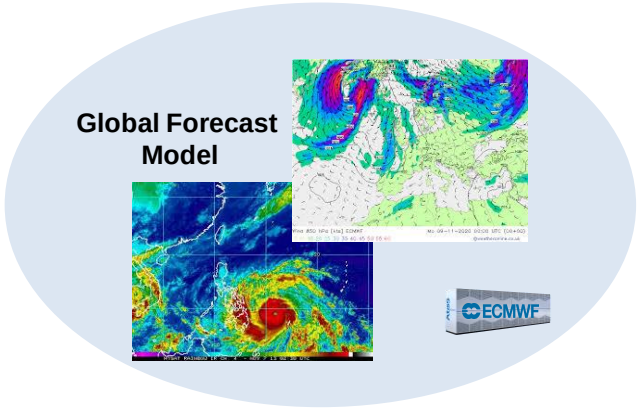
Cloud Data-as-a-Service

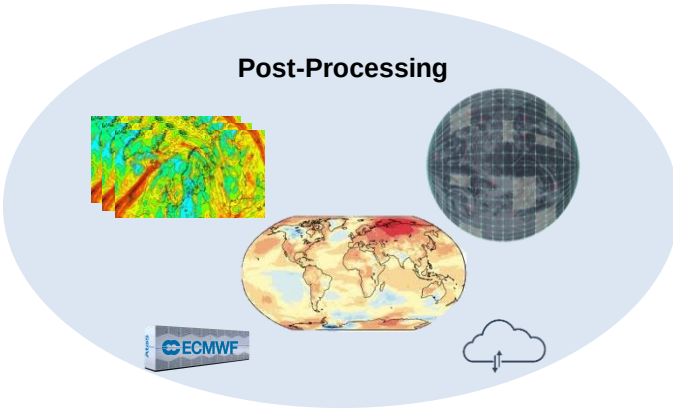
- Raw data output and disseminated products increasing in size
  - 5km ensemble ~1PiB of raw data in 1 hour
  - Transferring data out of the centre becoming prohibitive
  - Cloud data model: **bring the computing to the data**



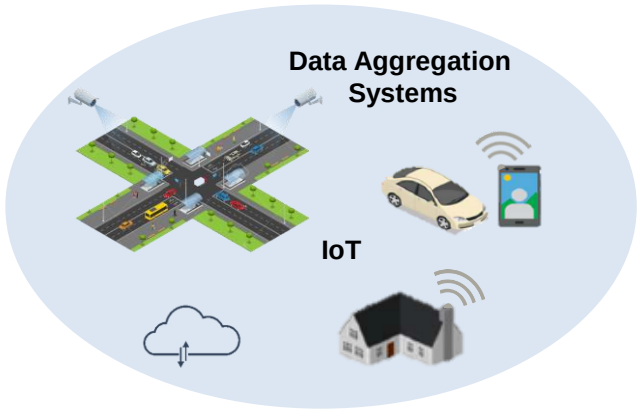
Enabling multi-domain workflows across data centres (HPC + Clouds)

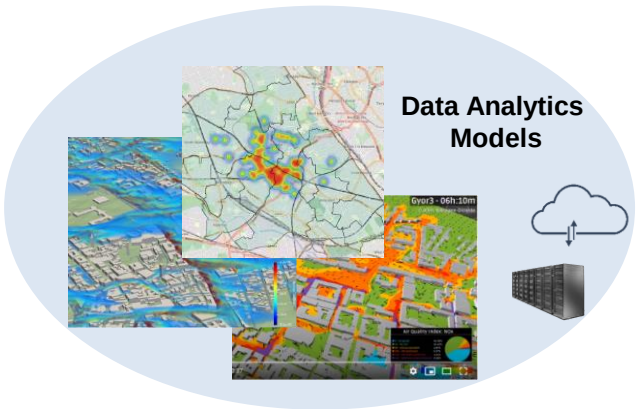


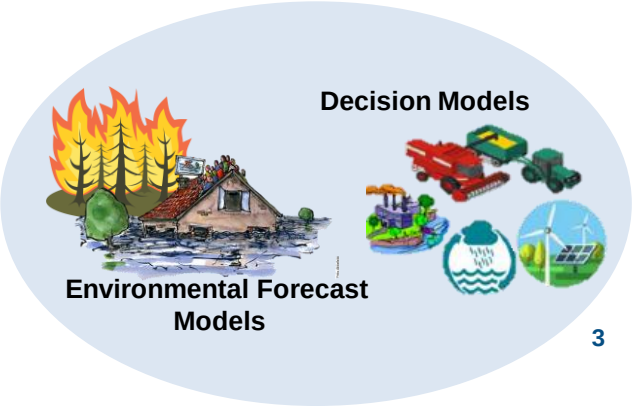




Model – To – Model
Notification of data availability is crucial



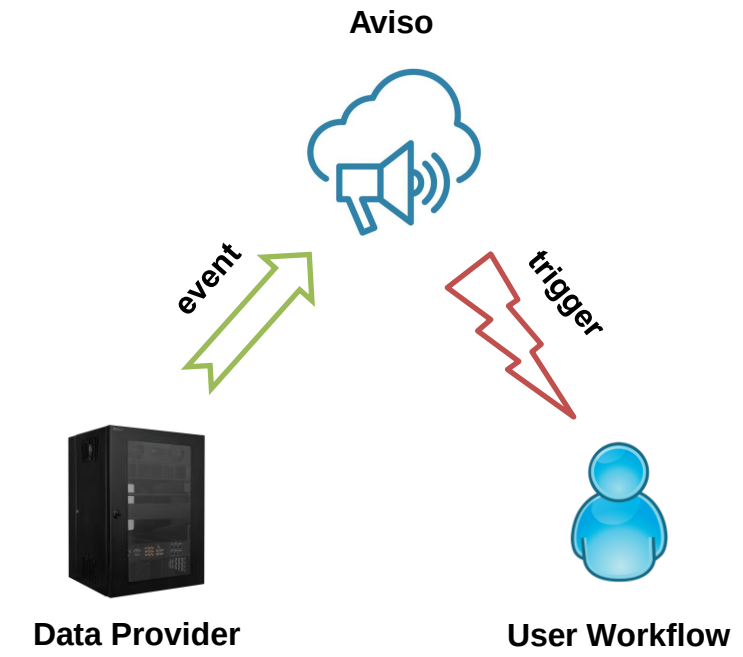




Aviso – Aims

Aviso is a software developed at ECMWF with the aim of:

- Notifying for **data availability**
- Meant for **automated systems**
- Support **When** <this> **… Do** <that> …

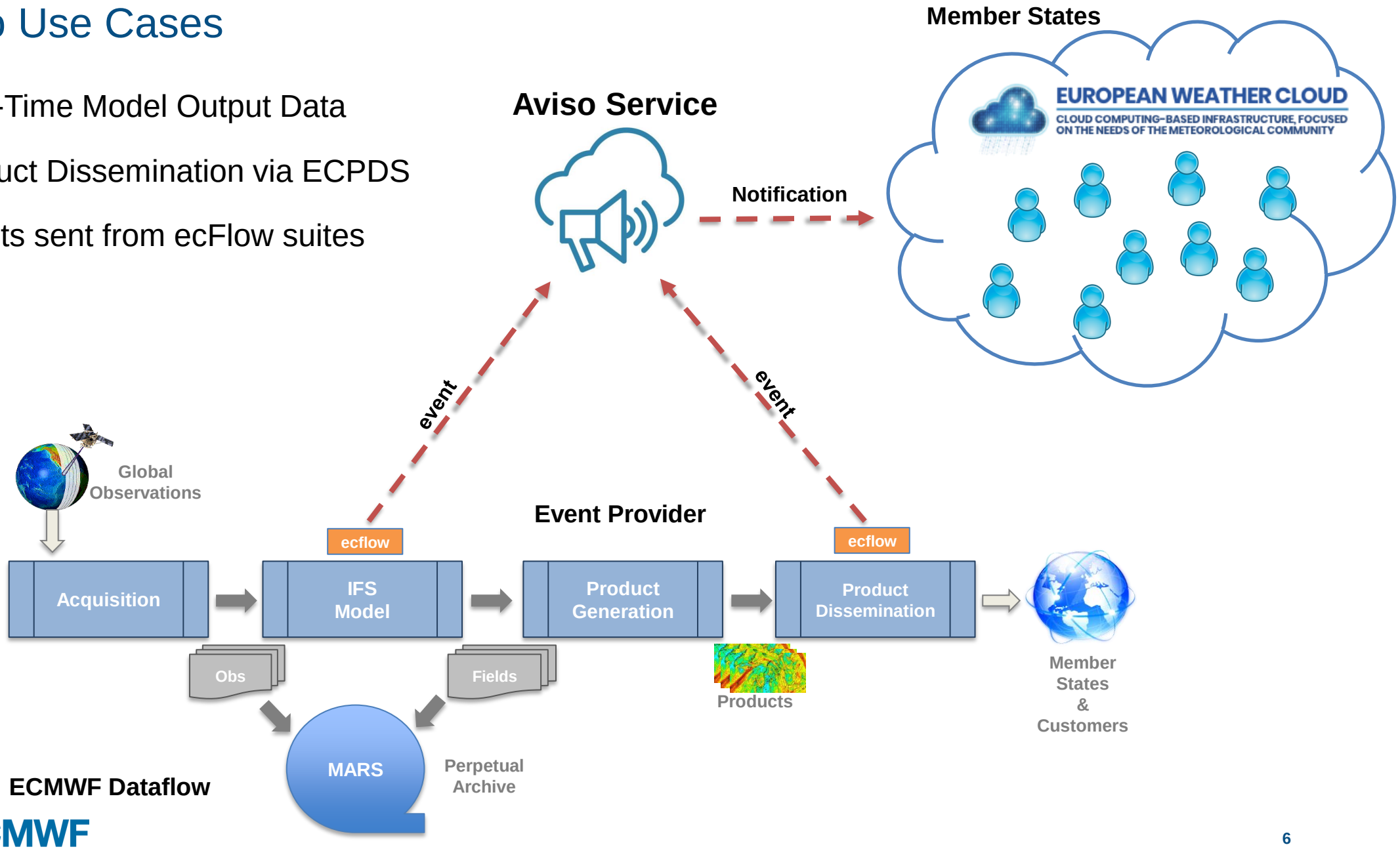


Aviso – Rationale

- **Domain-Specific** notification system:
  - Speaks **Meteorological** language (MARS DSL)
    - Understand our metadata
    - Richer experience for the user
  - **Protocol Agnostic:**
    - We can develop interfaces for MQTT, AMQP, RabbitMQ, Kafka, **CloudEvents**, etc..
- **Persistent State:**
  - Consistency
  - Transactionality
  - Reliability (notification history)
- Leverage on modern **Cloud technologies**:
  - Highly-available, high-throughput
  - Scalable & Distributed
  - Lightweight client

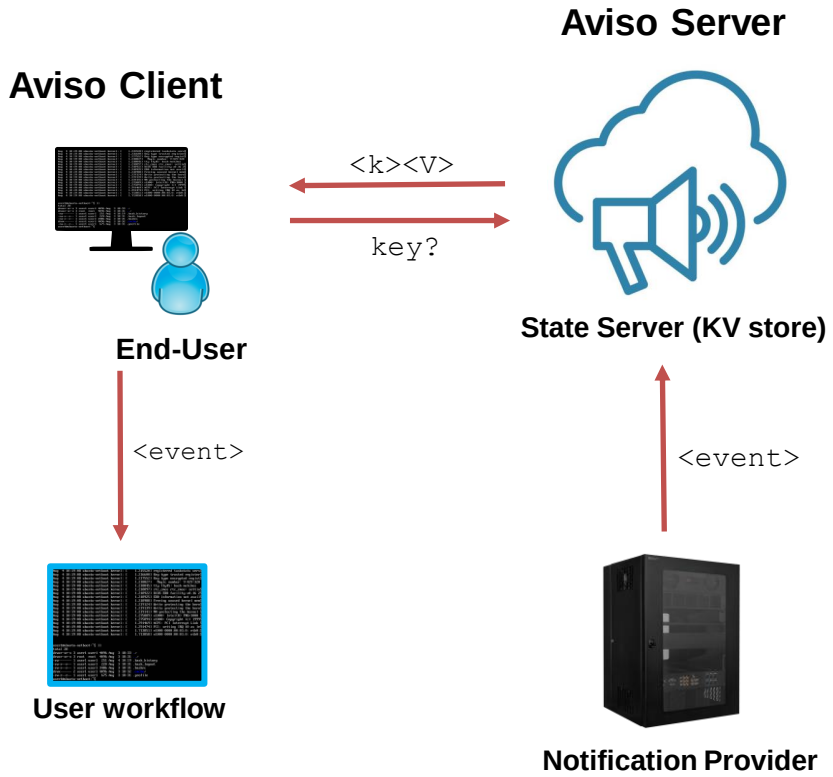
Aviso Use Cases

- Real-Time Model Output Data
- Product Dissemination via ECPDS
- Events sent from ecFlow suites



General Workflow

- Server: Microservices architecture based on a Key-Value Store
- Client: CLI application or Python API
- Users:
  - End-Users
  - Notification Providers
- Workflow:
  1. End-User subscribes to an event and programs the trigger.
  2. Aviso client polls for changes.
  3. A notification provider submits a notification.
  4. The subscriber is notified with a new event.
  5. The event triggers user's workflow.

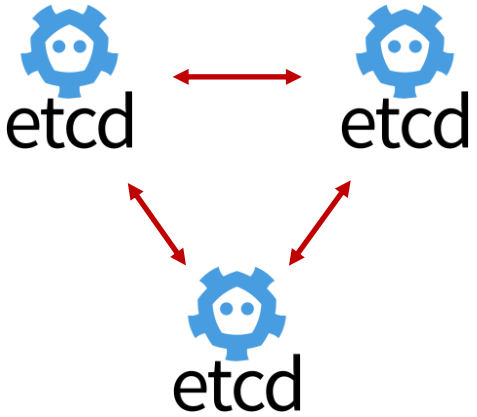


Key : Product's metadata
Value : Product's location

Aviso Server – Under the hood

etcd cluster

- Strongly consistent, distributed key-value store
- Leader election based on the Raft consensus algorithm
- Gracefully tolerates machine failure and network partition
- Provides gRPC and RESTful interfaces
- Benchmarked at 10,000s of writes per second
- Adopters:
  - kubernetes uses it as service discovery, storing cluster state and configuration
  - openstack. has it as configuration storage and distributed locking
  - CoreDNS uses it as optional backend
  - ROOK uses it as orchestration engine
  - M3 (metric platform for Prometheus) uses it for rule storage



Benchmark for a 3-size cluster

N. keys	N. clients	Write QPS	Latency x req.
100,000	1000	50,100	20 ms

Aviso Client - Command Line Interface

- Listen to events and program a trigger

```
aviso listen my_listener.yaml
```

my_listener.yaml

```
listeners:
  - event: dissemination
    request:
      destination: FOO
      stream: enfo
      step: [1,2,3]
    triggers:
      - type: command
        command: my_script.sh
        environment:
          STEP: ${request.step}
```

- Listen to a specific time interval in the past

```
aviso listen --from 2020-10-20T00:00:00.0Z --to 2020-10-21T00:00:00.0Z
```

- Test with local user-defined notifications

```
aviso notify event=mars,class=od,date=20190810,domain=g,expver=1,step=1,stream=enfo,time=0 --test
```

Aviso Client as a Python API

Aviso Client can be used as a Python API. Below an example:

1

2

3

4

```
from pyaviso import NotificationManager
from ecmwfapi import ECMWFService

# define function to be called
def do_something(notification):
    mars_server = ECMWFService("mars")
    request = notification["request"]
    request.update({
        "type": "fc",
        "levtype": "sfc",
        "param": 167.128})
    mars_server.execute(request, "my_data.grib")

# define the trigger
trigger = {"type": "function", "function": do_something}

# create an event listener request that uses that trigger
request = {"stream": "enfo", "date": 20190810, "time": 0}
listeners = {"listeners": [{"event": "mars", "request": request, "triggers": [trigger]}]}


# run it
aviso = NotificationManager()
aviso.listen(listeners=listeners)
```

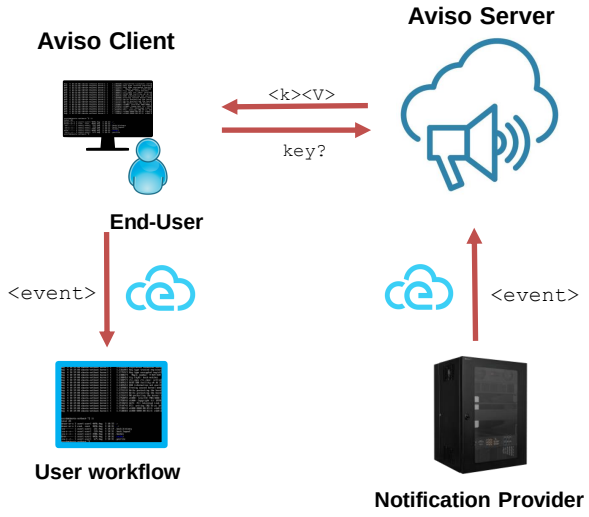
CloudEvents

CloudEvents is a specification seeking to describe event data in a common way

- **Consistency** – Reuse event handling logic for different event source
- **Accessibility** – SDKs for Go, JavaScript, Java, C#, Ruby, and Python to build common libraries, tooling, and infrastructure for delivering event data across environments.
- **Portability** - Interoperability among different cloud platforms. Implemented by most cloud vendors.

```
{
   "specversion" : "1.0",
   "type" : "aviso",
   "source" : "https://...",
   "id" :  "0c02fdc5-148c-43b5-b2fa-cb1f590369ff",
   "datacontenttype": "application/json",
   "data": {
      "event": "mars",
      "request": {…},
      "location": "…"
    }
}
```

CloudEvents in Aviso



CloudEvents as Aviso notifications

Notification Provider

Aviso Server

Key :ec/diss/foo/E1/od/g/1/enfo/20190810/0/1
Value :s3://storage.ecmwf.europeanweather.cloud/ecpds/E3S1

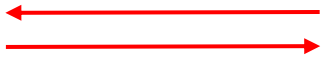
POST *https://aviso.ecmwf.*

```
{
  "type" : "aviso",
  "specversion" : "1.0",
  "source" : "/ecpds/emos",
  "id" : "0c02fdc5-148c-43b5-b2fa-cb1f590369ff",
  "datacontenttype" : "application/json",
  "data" : {
    "event" : "dissemination",
    "request" : {
      "destination" : "FOO",
      "target": "E1",
      "class": "od",
      "date" : "20190810",
      "domain" : "g",
      "expver" : "1",
      "step" : "1",
      "stream" : "enfo",
      "time" : "0" },
    "location":"s3://storage.ecmwf.europeanweather.cloud/ecpds/E3S1"
  }
}
```

CloudEvents as Aviso notifications

```
listeners:
  - event: dissemination
    request:
      destination: FOO
    triggers:
        - type: post
          protocol:
            type: cloudevents
            url: https://my.endpoint.com/
```

Aviso Server



Aviso Client

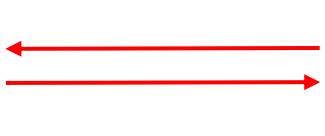
Key :ec/diss/foo/E1/od/g/1/enfo/20190810/0/1
Value :s3://storage.ecmwf.europeanweather.cloud/ecpds/E3S1

Request Key Prefix : ec/diss/foo

CloudEvents as Aviso notifications

```
listeners:
  - event: dissemination
    request:
      destination: FOO
    triggers:
        - type: post
          protocol:
            type: cloudevents
            url: https://my.endpoint.com/
```

Aviso Server



Aviso Client

End-User Endpoint

Key :ec/diss/foo/E1/od/g/1/enfo/20190810/0/1
Value :s3://storage.ecmwf.europeanweather.cloud/ecpds/E3S1

CloudEvents used as:

- protocol from notifiers to the server
- protocol from server to clients

POST *https://my.endpoint.com/*

```
{
  "type" : "aviso",
  "specversion" : "1.0",
  "source" : "https://aviso.ecmwf.int
  "id" : "6e364719-4880-49e9-adaf-c2f3334f771d",
  "datacontenttype" : "application/json",
  "data" : {
    "event" : "dissemination",
    "request" : {
      "destination" : "FOO",
      "target": "E1",
      "class": "od",
      "date" : "20190810",
      "domain" : "g",
      "expver" : "1",
      "step" : "1",
      "stream" : "enfo",
      "time" : "0" },
   "location":"s3://storage.ecmwf.europeanweather.cloud/ecpds/E3S1"
  }
}
```

ECMWF

Conclusions

- Aviso is a scalable notification system designed for high-throughput
- Automatic triggering of user-defined workflows
- ECMWF data availability of real-time forecast data or derived products
- CloudEvents compliant notifications for interoperability across clouds

Current status

- Aviso Service is currently pre-operational and open to users of the European Weather Cloud
- **> 300K** notifications a day among real-time and product dissemination
- First users: Met Norway and FMI

Beta version released on https://github.com/ecmwf/aviso and **PyPI**

Current Projects

ECMWF internal development

- Aviso: a scalable notification system for ECMWF data availability
- Polytope: a distributed API for data access and manipulation

European projects

- LEXIS - Complex weather and climate workflows over 3 HPCs and 2 Cloud systems
- HiDALGO - On-demand data access for Global Challenges applications over multiple HPC sites

ECMWF

European Commission

These projects have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 825532 and 824115.