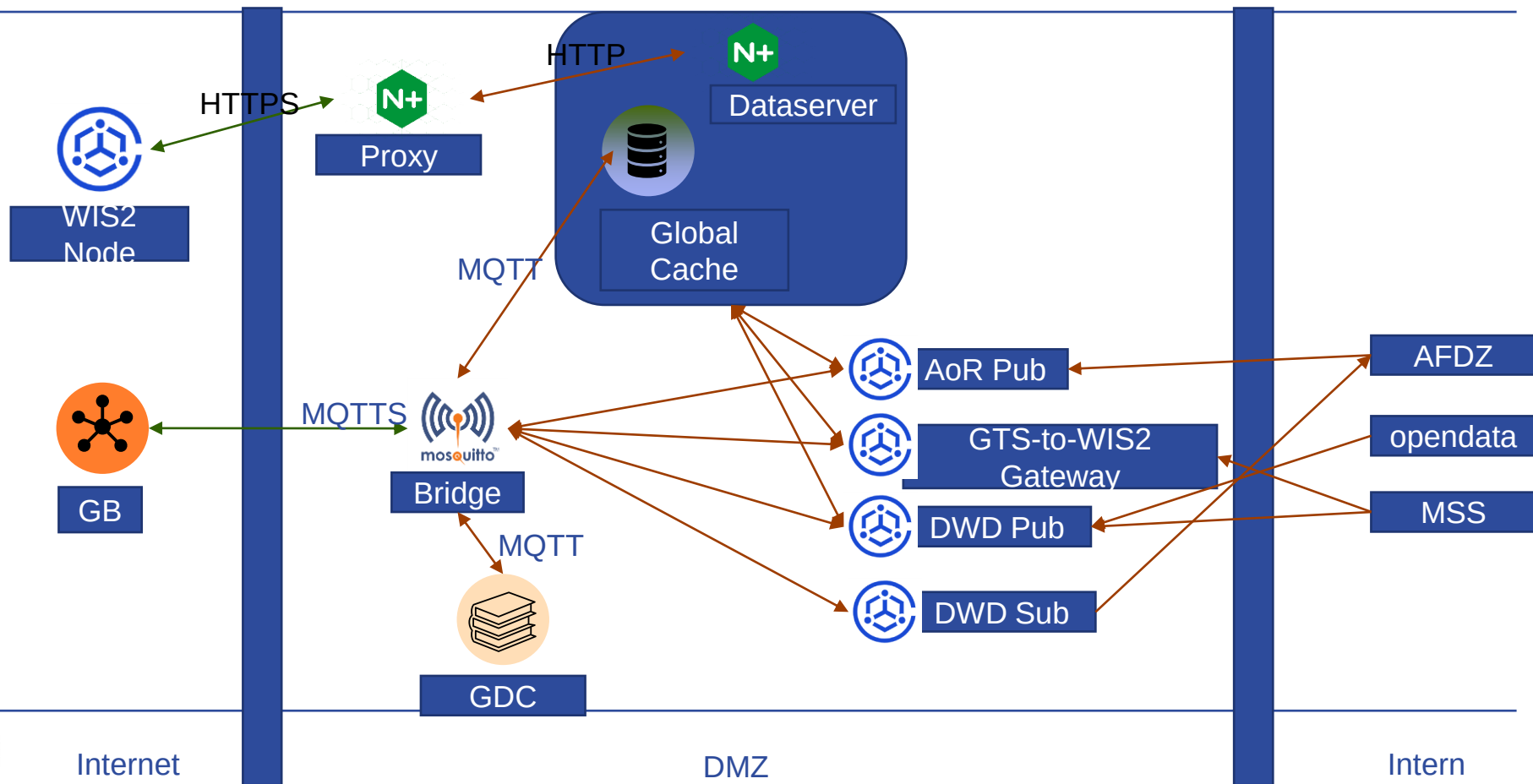


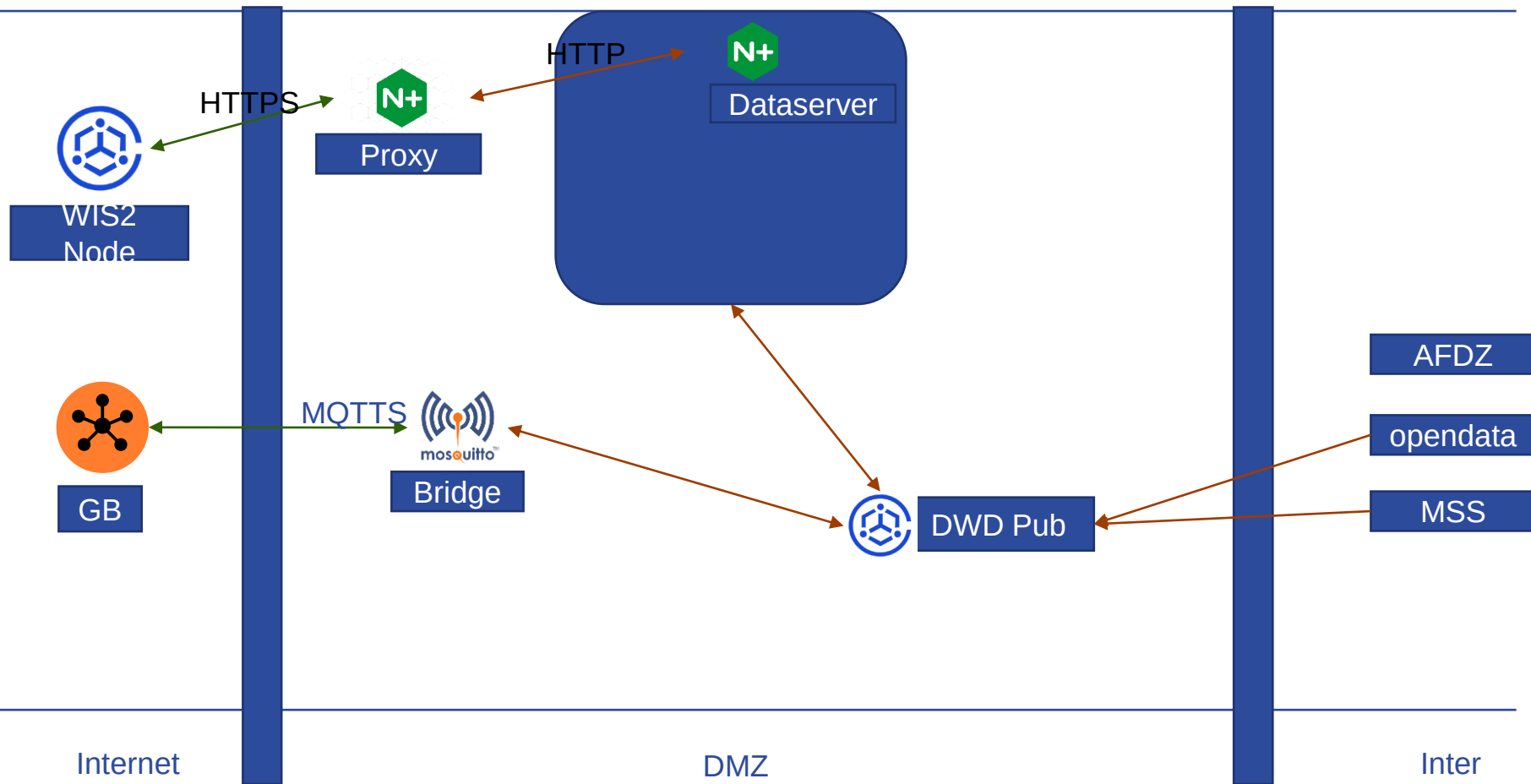


WIS2Node Implementation in Germany

Overview – WIS2 @ DWD



Overview – WIS2Node

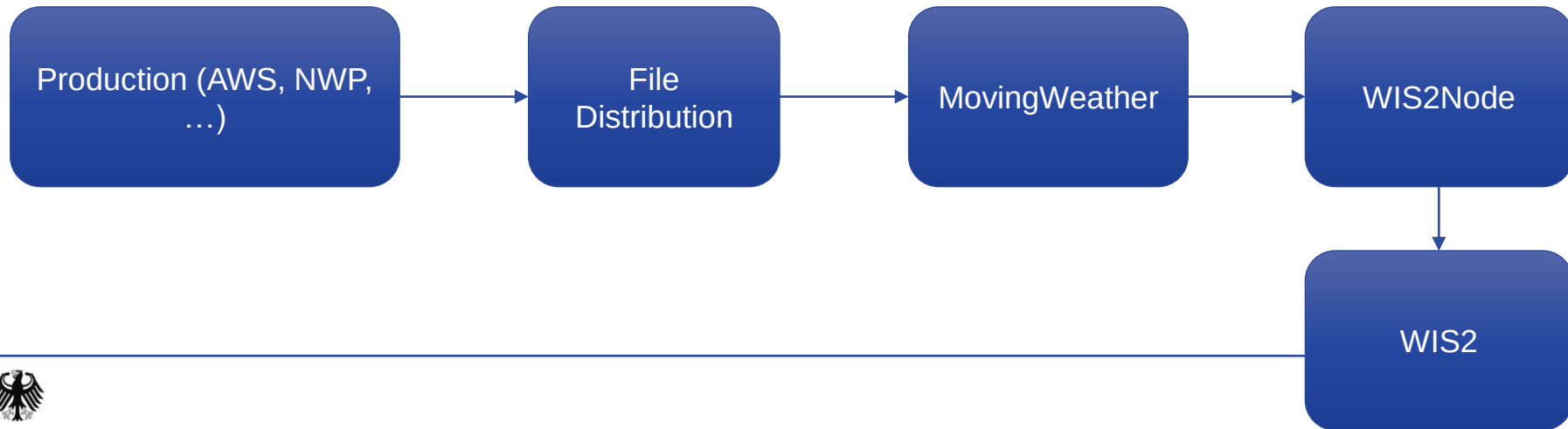


DWD Pub - The Path of a File

WIS1:

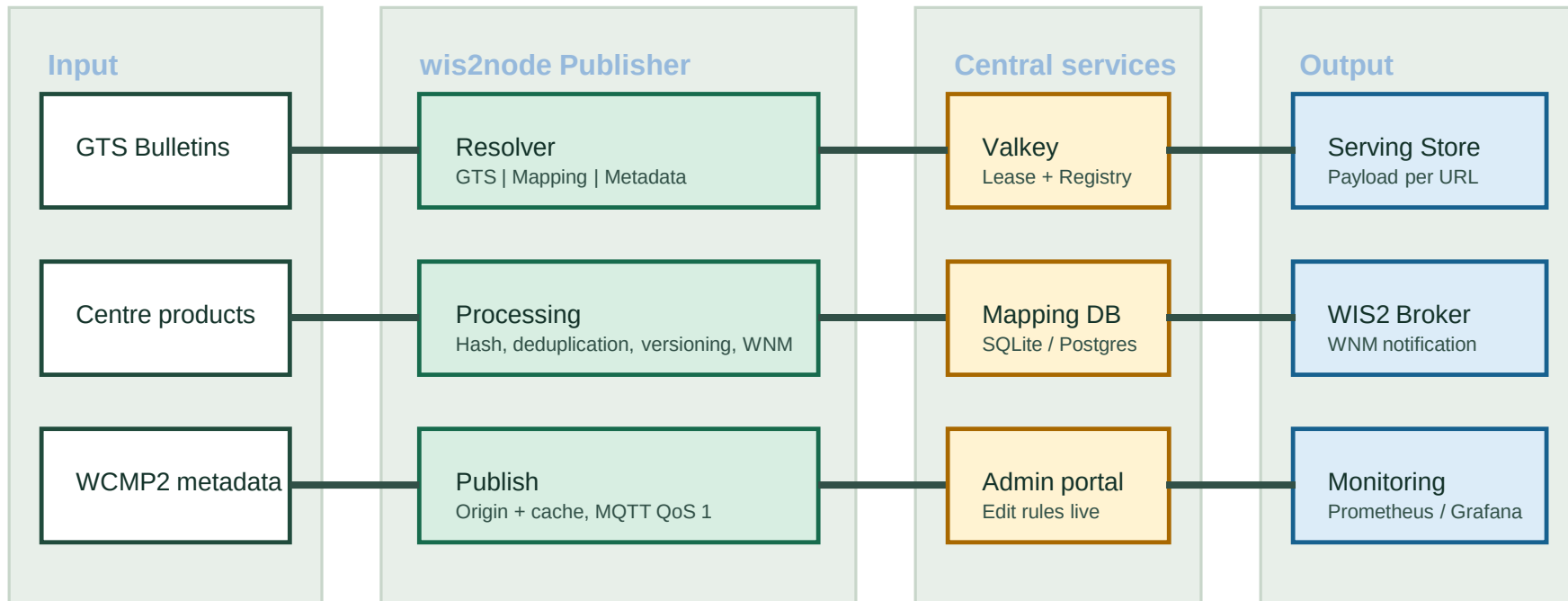


WIS2:



DWD Pub - System Overview

Different inputs, one controlled and observable delivery path

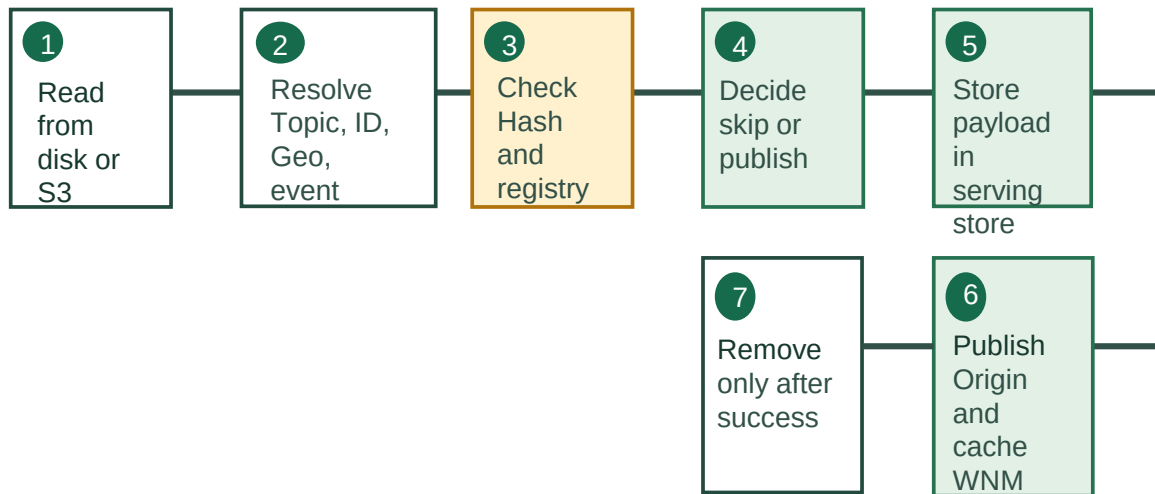


Core rule: persist the payload before sending the notification.



What happens to a file?

The path is deliberately sequential: simple error handling and safe reprocessing.



Reprocessing is safe: identical content is skipped; new content receives a new immutable URL.

- same filename + same content → skip
 - same filename + new content → new version, "update" event
 - new filename → publish
-
- WIS2Box overrides the old file (update planned)

- ➔ <https://wis2.dwd.de/metadata-editor/>
- ➔ Web-based editor, freely available on the internet
- ➔ Business side creates metadata
- ➔ Metadata file is copied to our wis2node and is published (own container)



WIS2 Metadata Editor

Create and manage WCMP2 compliant metadata for the WIS2 Global Discovery Catalogue

How to Use This Editor

1. **Create:** Fill out the metadata form with your dataset information
2. **Validate Form:** Use the "Validate Form" button to check for missing required fields
3. **WIS2 Validate:** Use the "WIS2 Validate" button to validate metadata against official WCMP2 standards
4. **Export:** Download your metadata as JSON using "Export As JSON"
5. **Submit:** For metadata published by DWD: Send the downloaded JSON file to wis@dwd.de for publication

 **Note:** This editor does not save automatically. Remember to download your JSON before leaving the page!

→ Rules:

→ filename pattern → topic, metadata, geometry, event

→ Management of rules via web portal:

→ validation

→ priority

→ Publishers load changes within roughly 60 seconds

```
"Z__C_EDZW_*synop_bufr_GER*": {  
  "event": "canonical",  
  "relPath": "observations",  
  "topic": "origin/a/wis2/de-dwd/data/core/weather/surface-based-observations/synop",  
  "metadata_id": "urn:wmo:md:de-dwd:weather.observations",  
  "geometry": {  
    "Polygon": {  
      "west": 5.87,  
      "south": 47.27,  
      "east": 15.04,  
      "north": 55.06  
    }  
  },  
  "datetime": "BUFR"  
},
```

→ Rules:

→ filename pattern → topic, metadata, geometry, event

WIS2 Node Mapping Admin

admin [Change Password](#) [Logout](#)

Mapping Rules

Users

Mapping Rules

de-dwd (10/10 enabled) ▼

[Copy all](#)

[Paste rules...](#)

[Add Rule](#)

Rules are matched with `fnmatch` against the incoming filename. They are listed in evaluation order and **the last matching rule wins** — move a rule *down* to give it precedence over the ones above it. Publishers pick edits up within about a minute; no restart needed.



fc_*

CANONICAL

NO CACHE

[Copy](#)

[Edit](#)

[Disable](#)

[Delete](#)

Topic: `origin/a/wis2/de-dwd/data/core/weather/prediction/forecast/medium-range/probabilistic/global`

Metadata ID: `urn:wmo:md:de-dwd:icon-eps.ALL`

Rel. path: `forecast`

Datetime: `fc_YYYYmmdd%hicle_XXX`

Geometry: Bounding box (W -180, S -90, E 180, N 90)

priority 0 · last changed by seed · 2026-08-24 09:47 UTC



Z__C_EDZW_*synop_bufnr_GER*

CANONICAL

[Copy](#)

[Edit](#)

[Disable](#)

[Delete](#)

Topic: `origin/a/wis2/de-dwd/data/core/weather/surface-based-observations/synop`

Metadata ID: `urn:wmo:md:de-dwd:weather.observations`

Rel. path: `observations`

Datetime: `BUFR`

Geometry: Bounding box (W 5.87, S 47.27, E 15.04, N 55.06)

priority 1 · last changed by seed · 2026-08-24 09:47 UTC

→ Rules:

→ filename pattern → topic, metadata,
geometry, event

Edit Rule

Centre ID *

de-dwd

Type a new value to start a new centre.

Filename pattern *

Z_C_EDZW_*synop_bufr_GER*

fmatch glob: * any run, ? one char, [abc] a set.

Topic *

origin/a/wis2/de-dwd/data/core/weather/surface-based-observations/synop

The part after /a/ also becomes the `data_id` prefix.

Metadata ID *

urn:wmo:md:de-dwd:weather.observations

Relative path

observations

Prefix for the stored object key. Leave empty to store at the root.

Datetime spec

BUFR

BUFR reads the time from the message; a template slices it out of the filename.

Geometry

☐ None ☐ Point ☒ Bounding box ☐ GeoJSON

Paste GeoJSON...

west

5.87

south

47.27

east

15.04

north

55.06

Event

canonical

☒ Cacheable

☒ Enabled

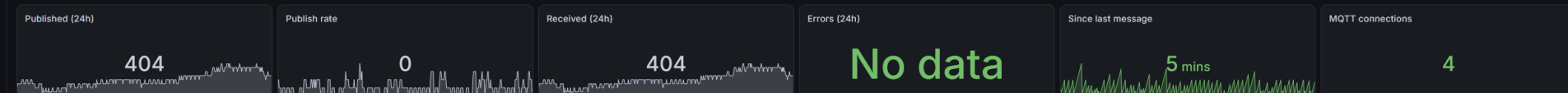
Cancel

Save

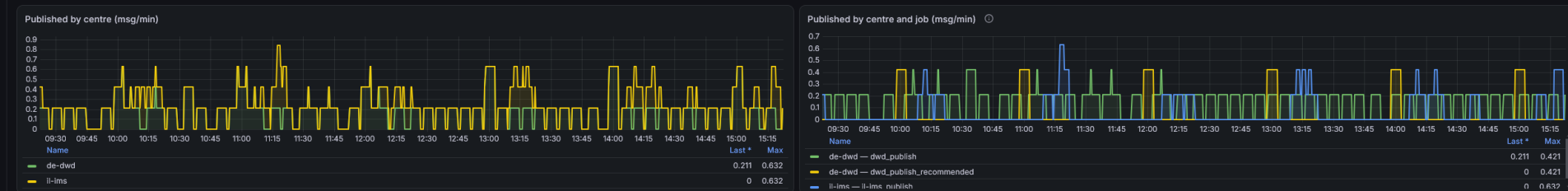


DWD Pub - Making Operations Visible

Health



Throughput per centre



Staleness and errors



DWD Pub - Making Operations Visible

Health

Published (24h)

1 Mil

Publish rate

663

Received (24h)

1 Mil

Errors (24h)

No data

Since last GTS message

7 s

MQTT connections

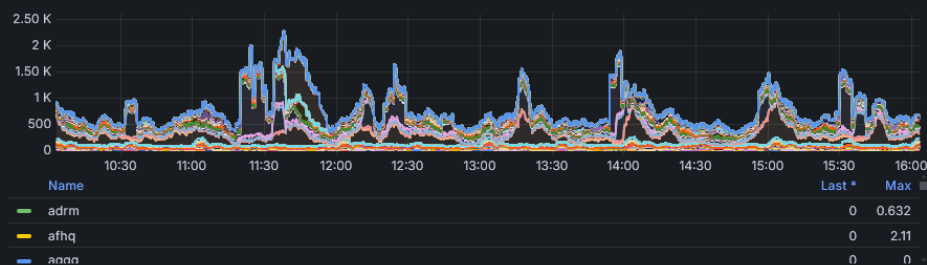
4

Throughput

Published vs received (msg/min)



Published by CCCC (msg/min)



Errors and backlog

Errors (msg/min)

No data

Unpublished backlog (received - published, 1h)



DWD Pub - Making Operations Visible

Deutscher Wetterdienst
Wetter und Klima aus einer Hand



→ Prometheus metrics



→ Alertmanager for alerting



wis2node-gateway /etc/prometheus/wis2node-rules.yml

WIS2-GTS-Gateway No Data Published

WIS2-GTS-Gateway Metrics Absent

WIS2-GTS-Gateway Ingest Stalled

WIS2-GTS-Gateway Error Rate

wis2node-metadata /etc/prometheus/wis2node-rules.yml

WIS2 Metadata No Daily Refresh

wis2node-nodes /etc/prometheus/wis2node-rules.yml

WIS2 Node de_dwd No Data Published Core

WIS2 Node de_dwd No Data Published Recommended

WIS2 Node il_ims No Data Published

WIS2 Node Metrics Absent

WIS2 Node Error Rate

wis2node-process /etc/prometheus/wis2node-rules.yml

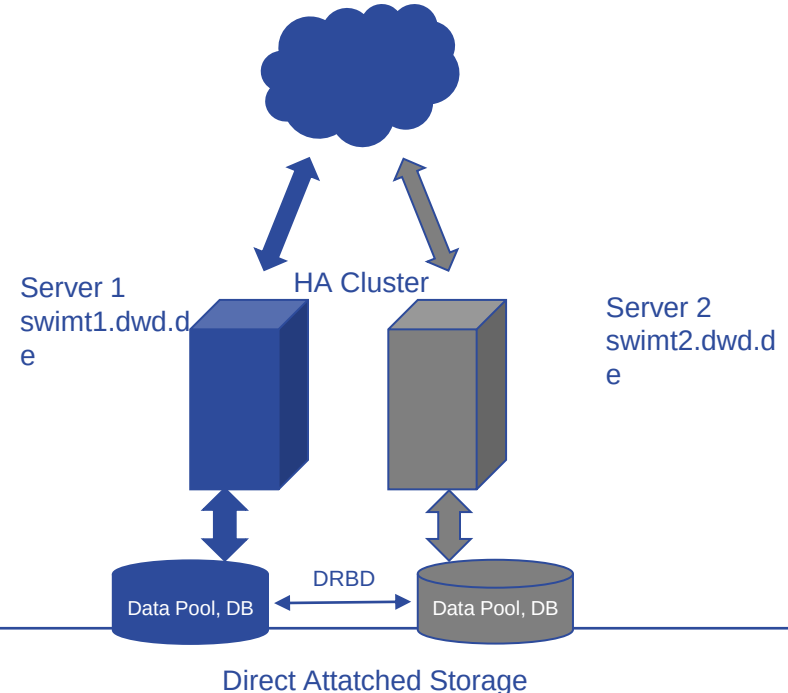
WIS2 Publisher Broker Disconnected

WIS2 Publisher Memory High



- ➔ DWD also operates a GC & GDC
- ➔ All WIS2 components share resources like broker & data server
 - ➔ Node directly publish a cache message
 - ➔ Origin and cache points to the same data file

- ➔ One software stack for GTS-to-WIS2 bridge and node
- ➔ High availability:
 - ➔ Currently: Pacemaker
 - ➔ Future: Kubernetes



HA: Pacemaker

→ two identical server (active-passiv)

→ pacemaker/corosync + DRBD

→ Problem:

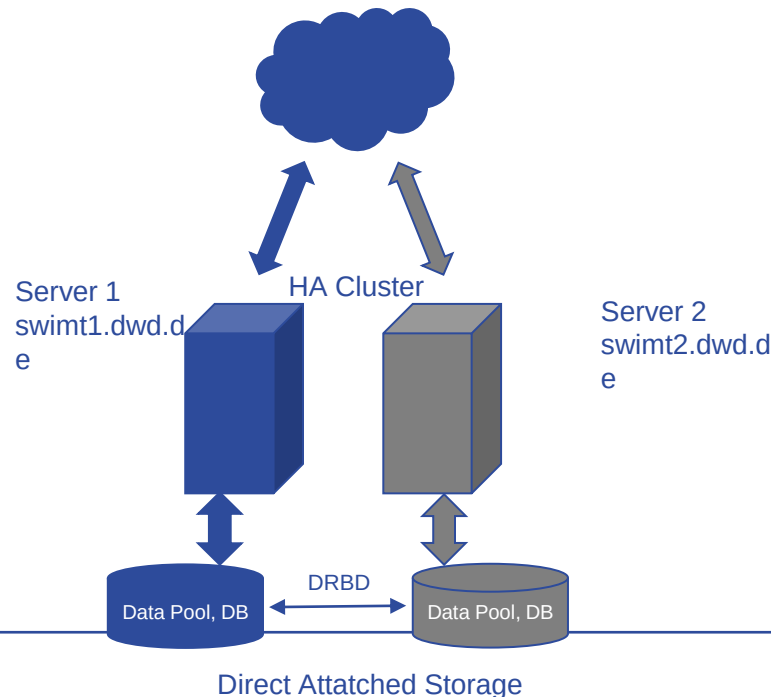
→ WIS2 services runs as containers

→ Pacemaker do not work well with containers

→ Long-live, stateful services
vs

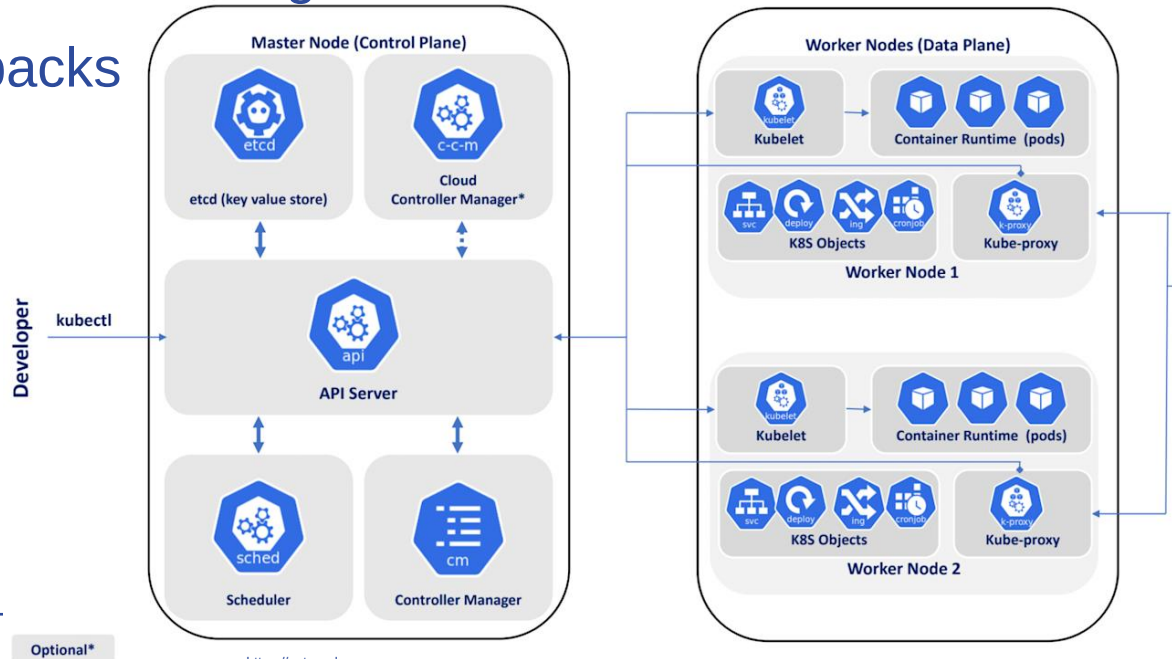
→ Short-live stateless containers

→ No health checks



HA: Kubernetes

- $N > 3$ nodes, standard hardware
- horizontal scaling and autoscaling
- Zero downtime & rollbacks
- Load balancing
- Health checks



Thank you for your attention!