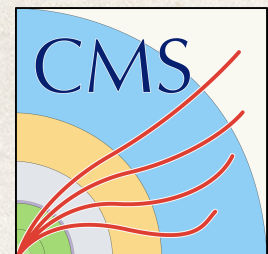
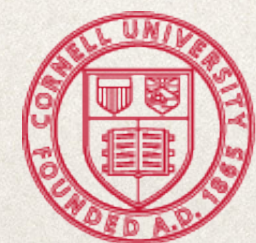


The Evolution of CMS Monitoring infrastructure

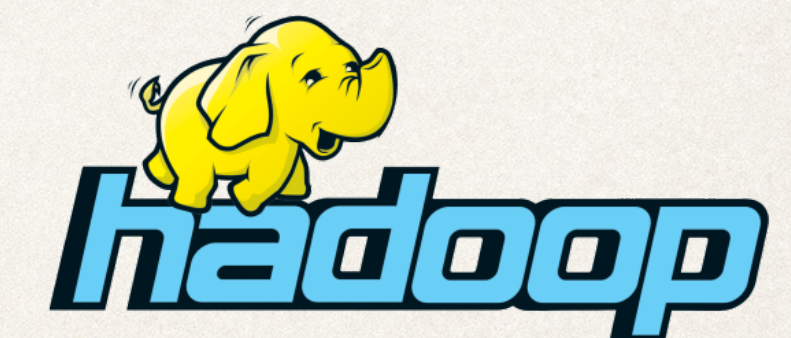
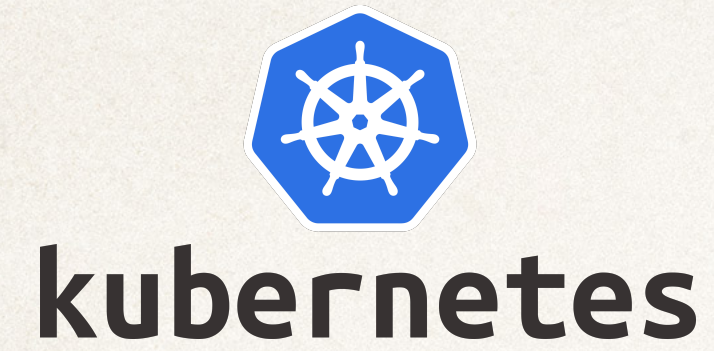
Christian Ariza, Valentin Kuznetsov, Federica Legger, Rahul Indra, Nikodemas Tuckus, Ceyhun Uzunoglu



vCHEP 2021














Cornell University



CMS Monitoring

CMS Monitoring provides comprehensive information about CMS services and metrics.
We support Data Injection, Data Storage and Data Access into CERN [MONIT](#) infrastructure.

<p>Prometheus</p>  <p>HA1 HA2</p>	<p>Alerts</p> <ul style="list-style-type: none"> AlertManager (HA1) Dashboard (HA1) AlertManager (HA2) Dashboard (HA2)	<p>Dashboards</p> 	<p>Slack</p> 
<p>Jira Tickets</p> 	<p>SNOW Tickets</p> 	<p>Documentation</p> 	<p>Contact Us</p> 

CMS Monitoring infrastructure



Complexity
of software

Volume
of data

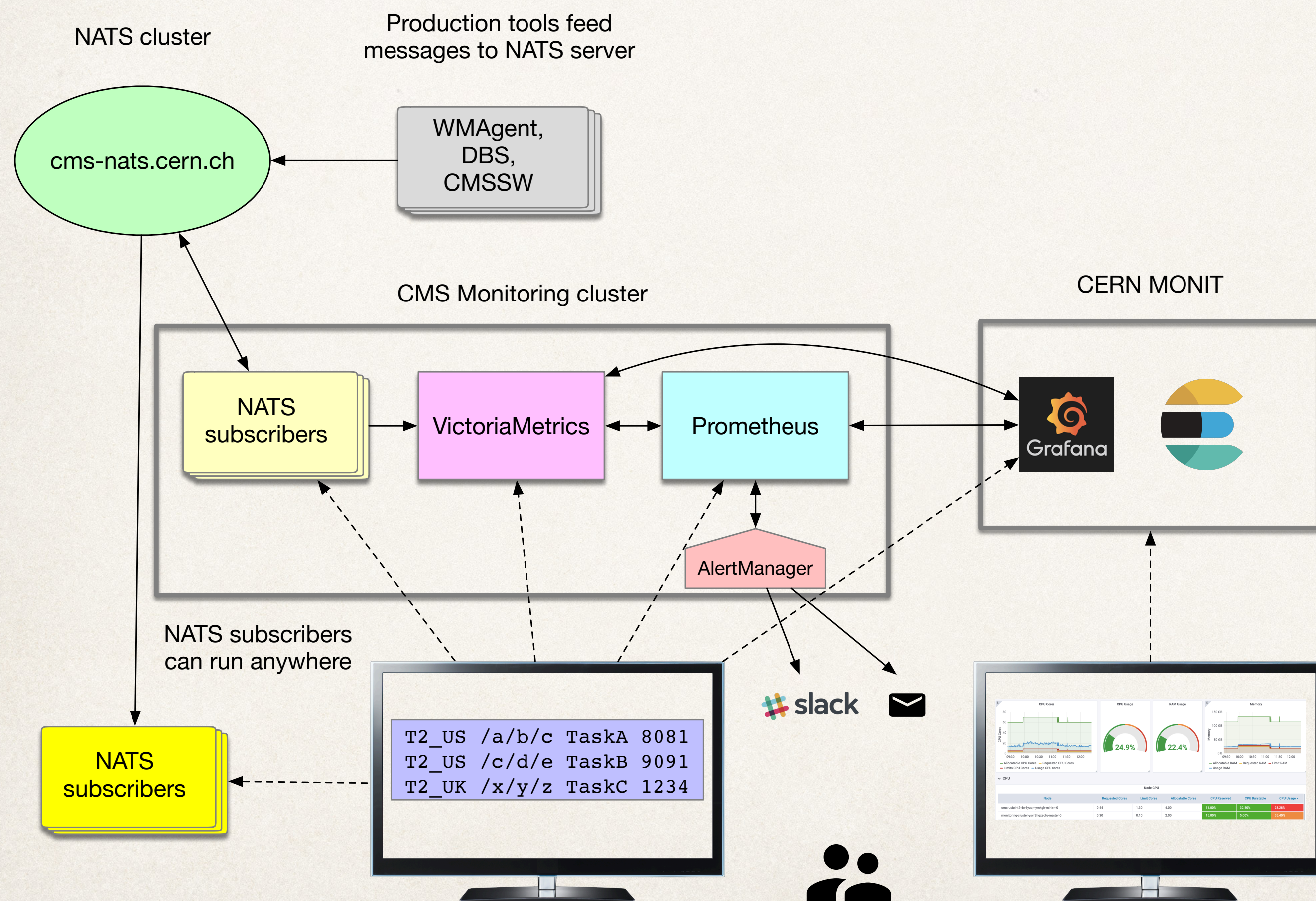
Variability
of infra

Velocity
of collection

- ❖ 5 k8s clusters
- ❖ 500B data points
- ❖ we monitor:
 - ❖ 200+ nodes / services on VMs
 - ❖ 100+ services on k8s
 - ❖ 150+ alert rules
- ❖ 100 prod dashboards, 500+ overall
- ❖ 30 TB of ES data
- ❖ 25 TB of compressed data on HDFS
- ❖ 3.5M msg/hour to AMQ brokers with 7.5kHz rate

Monitoring has
become a data
analytics platform

Initial infrastructure (2 years in production)



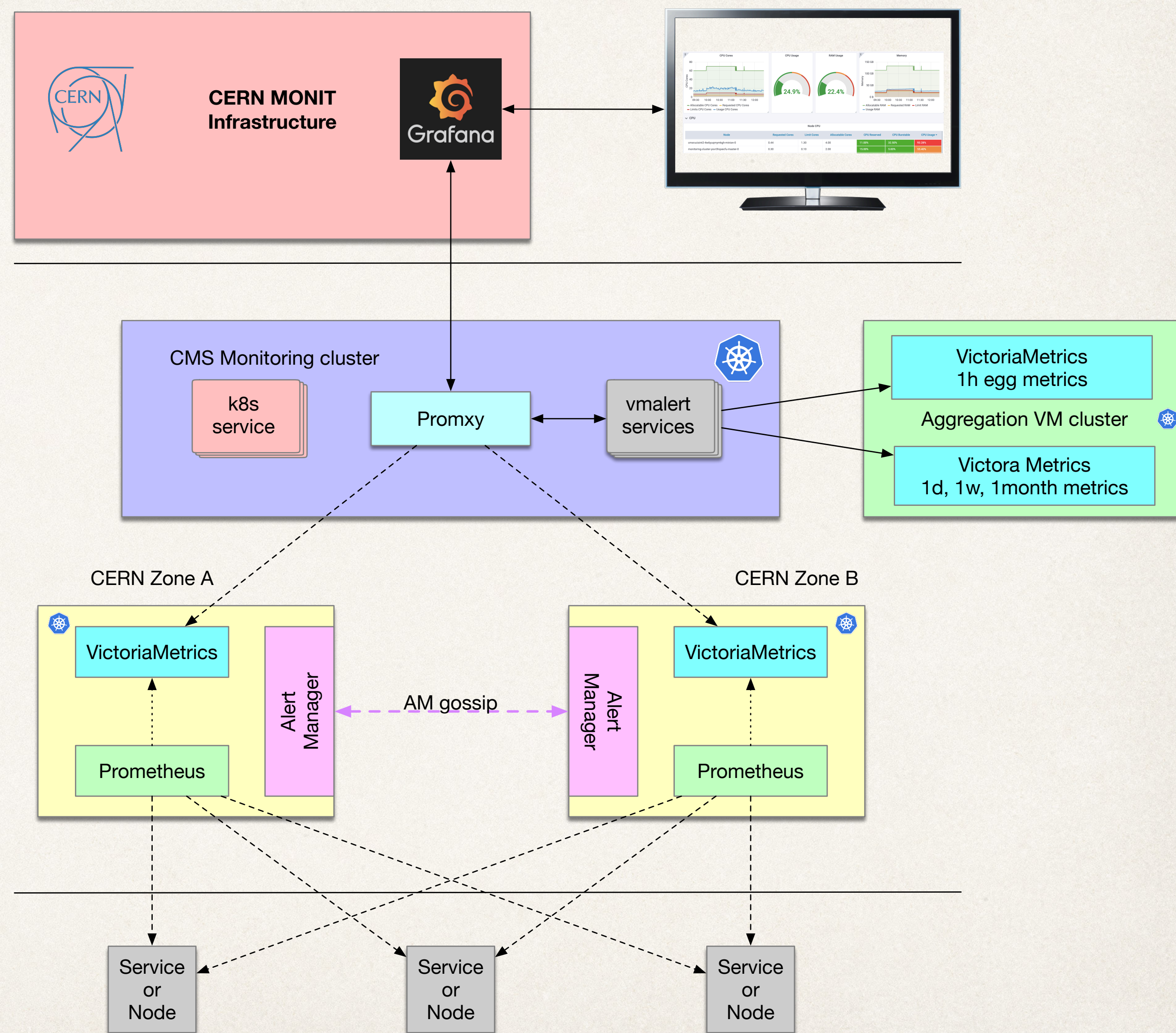
- ❖ In addition to CERN MONIT we setup two k8s clusters
 - ❖ monitoring cluster to run Prometheus, VictoriaMetrics, etc.
 - ❖ NATS cluster for real-time monitoring
- ❖ We used VictoriaMetrics as back-end storage for our Prometheus with retention policy 30 days
- ❖ We setup AlertManager to handle all our alerts

Learned lessons along the way

- ❖ Two-side sword problem: expressiveness vs concise information
 - ❖ annotations can be useful to indicate root of the problem but their excess may overwhelm users
- ❖ Monitoring became an analytical tool to address various problems
- ❖ Data vs Infrastructure vs Visualization separation is a big benefit
- ❖ Develop easy to use CLI tools
 - ❖ manage alerts, annotations, query data in various data-sources, etc.
- ❖ Work with open-source developers via Slack channels, GitHub issues, etc.

New infrastructure

- ❖ We moved to High-Availability mode by setting individual k8s clusters in different zones for our main monitoring stack
- ❖ Prometheus scrapes the metrics and stores them in VictoriaMetrics (short vs long-term storage)
- ❖ All aggregated records redirected to separate instance of VictoriaMetrics on dedicated k8s cluster
- ❖ We use Promxy to access all VictoriaMetrics servers
- ❖ AlertManagers relies on Hashicorp's member list to gossip information about notifications and silences
- ❖ CLI tools: SSB/GGUS parsers, alert, annotation, token management, data access to MONIT, etc.

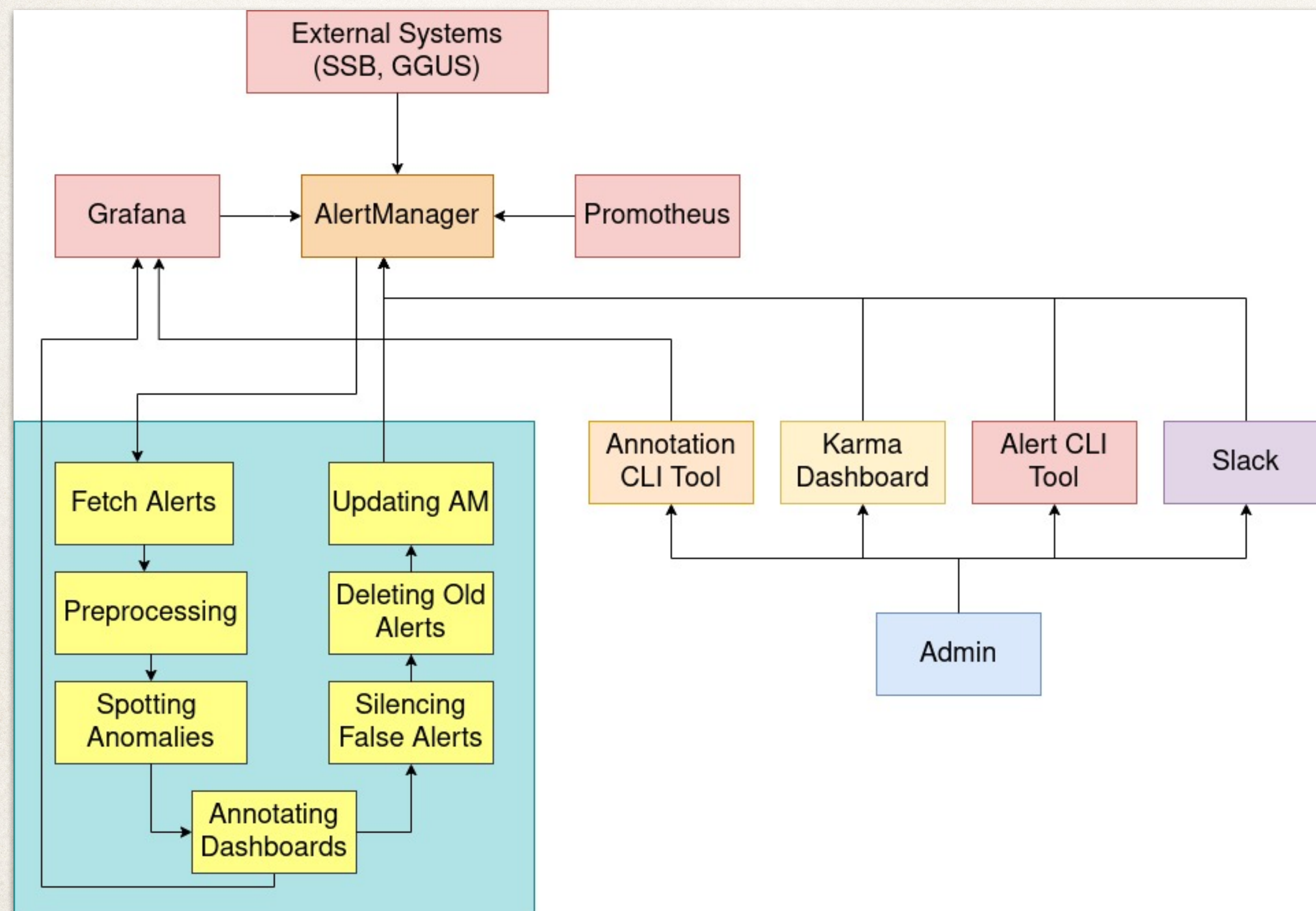


Further improvements

- ❖ High-Availability provides fault-tolerant mode of operations
 - ❖ we can upgrade our k8s cluster independently without any interruption in data processing and availability for end-users
- ❖ We are considering to start data portioning across HA clusters
 - ❖ node metrics go to cluster X, k8s metrics go to cluster Y, service metrics go to cluster Z
 - ❖ use different resource allocations for flexible retention policies
- ❖ Data relabeling and long storage of dynamically aggregated records
 - ❖ route aggregated records to dedicated VictoriaMetrics instance(s)

CMS Monitoring services

Intelligent alert management system



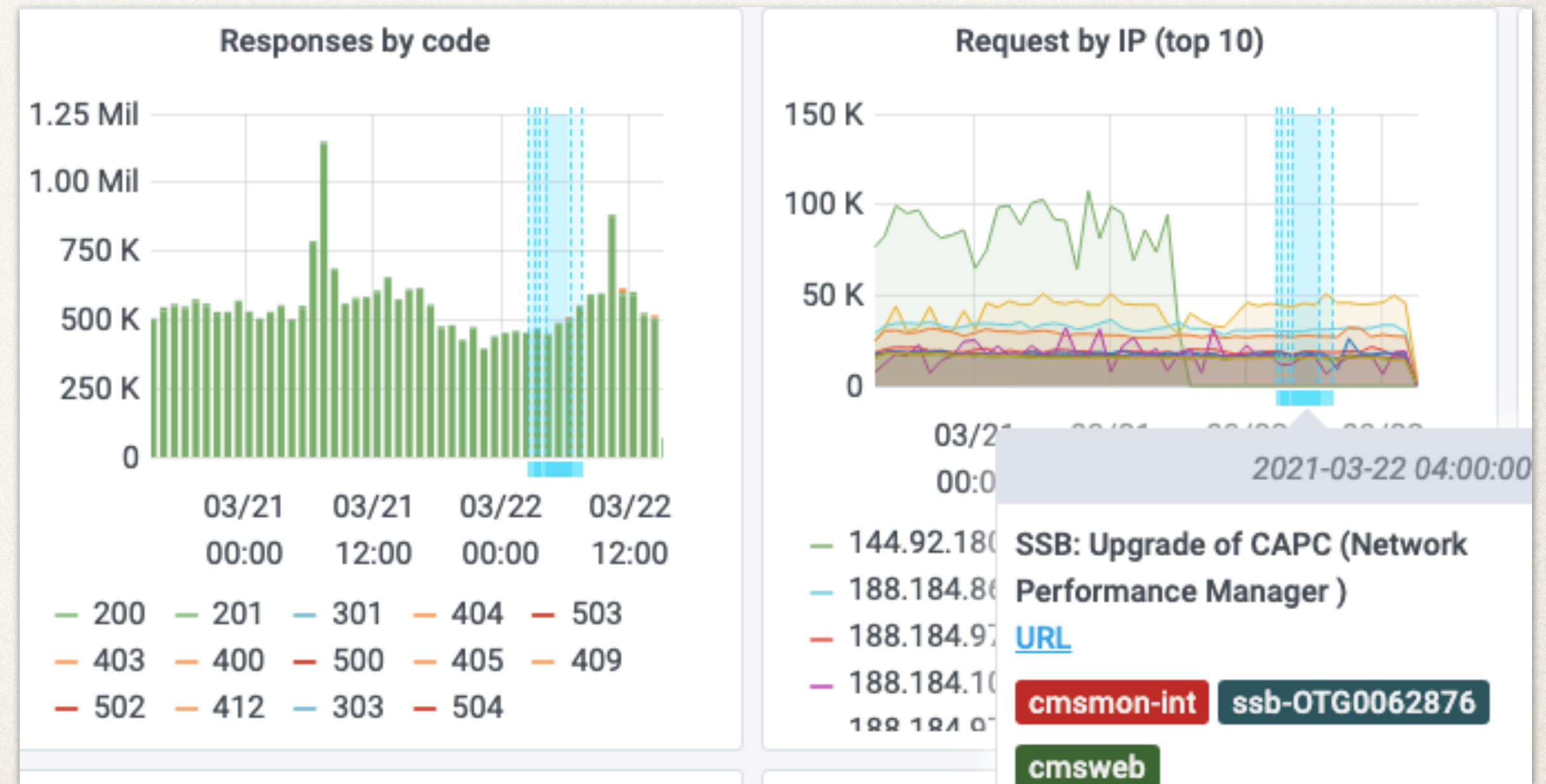
- ❖ Custom made service
 - ❖ parse GGUS and SSB tickets
 - ❖ add intelligent processing (anomalies, silencing, tagging)
 - ❖ annotate Grafana dashboards
- ❖ k8s deployment close to Alert Manager
- ❖ Data and experiment agnostic
- ❖ Can be deployed with any Prometheus/AlertManager stack

Intelligent alert management system

- ❖ Expandable pipeline loop
- ❖ Full integration with AlertManager
- ❖ Process Labels and Annotations
- ❖ Fetch \Rightarrow Preprocess \Rightarrow Filter \Rightarrow Keyword matching \Rightarrow Add annotation \Rightarrow ML Box \Rightarrow Push \Rightarrow Delete silence
- ❖ Currently works with GGUS and SSB services

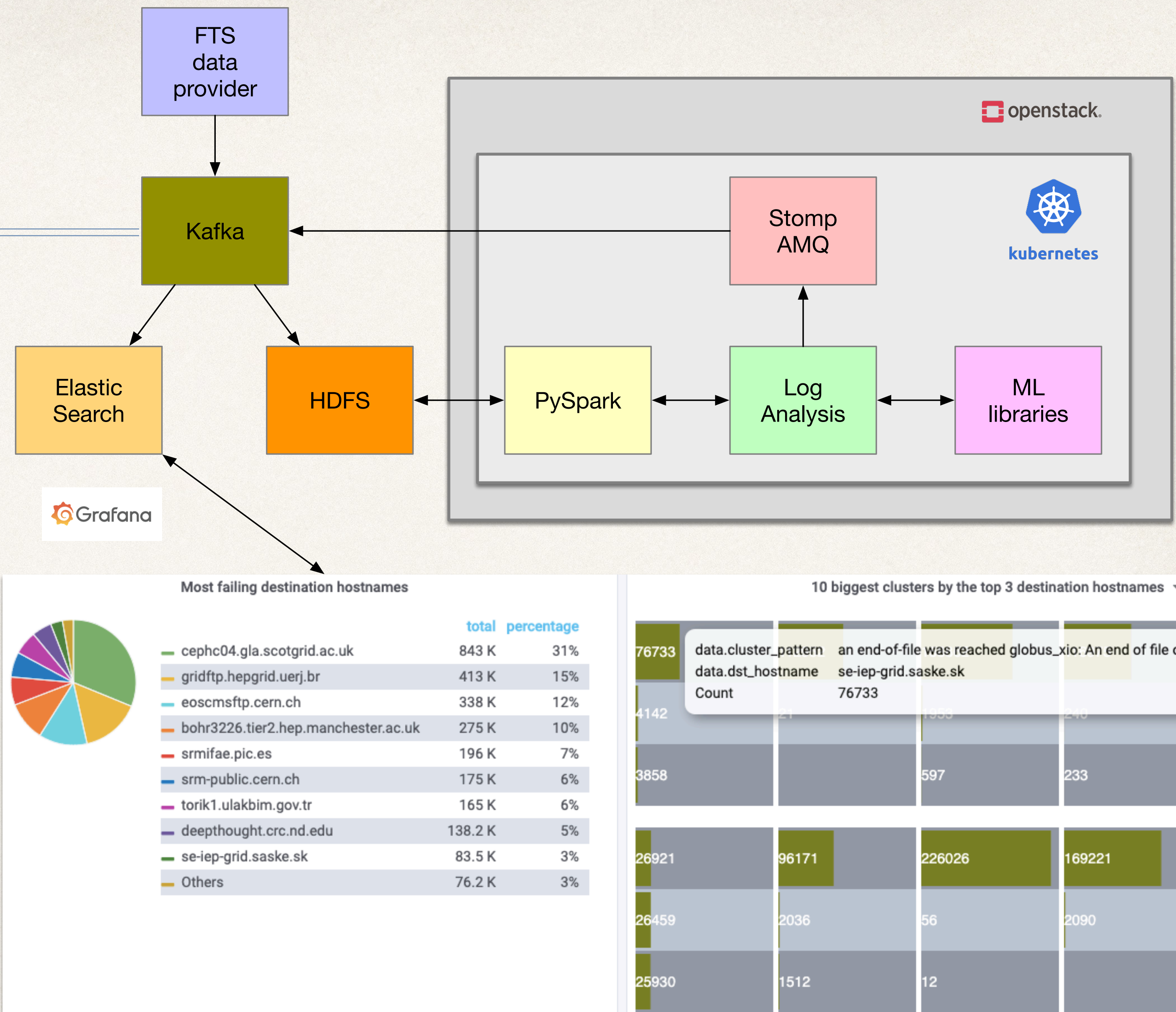
```
var processedData []models.AmJSON
a := pipeline.DeleteSilence(pipeline.Silence(
    pipeline.PushAlert(
        pipeline.MlBox(
            pipeline.AddAnnotation(
                pipeline.KeywordMatching(
                    pipeline.Filter(
                        pipeline.Preprocess(
                            pipeline.FetchAlert()))))))))

for d := range a {
    processedData = append(processedData, d)
}
```



FTS Log analysis

- ❖ Develop pipeline to perform clustering of FTS errors
- ❖ Data are processed by Sqoop job on HDFS
- ❖ We apply Levenhstein distance for clustering or can use word2vec ML approach
- ❖ Data shipped back to ES via StompAMQ
- ❖ Use Grafana for data visualization

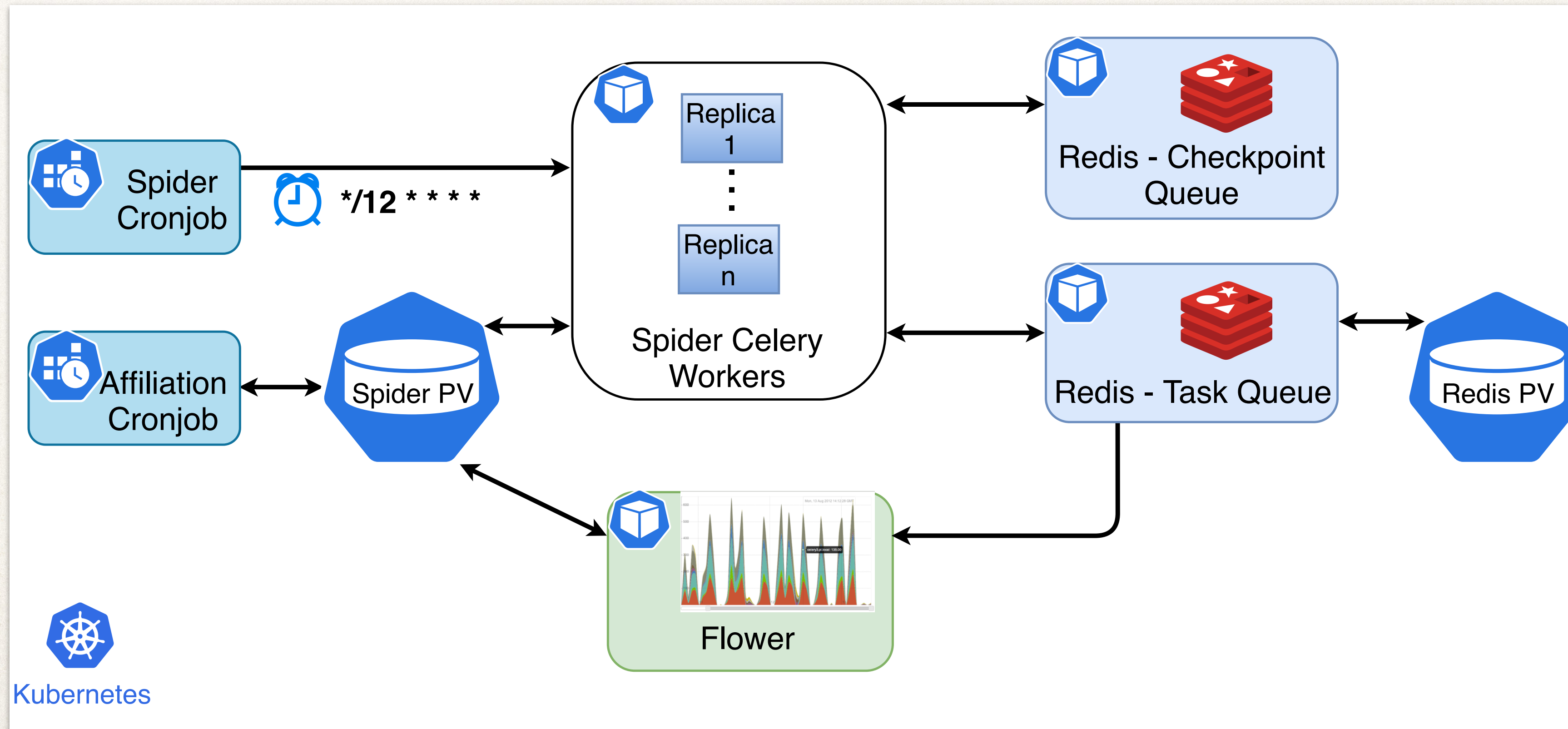


CMS Monitoring CLI tools

- ❖ We developed all CMS Monitoring CLI tools using GoLang
 - ❖ fast, native concurrency, static executables, easy maintenance, work on all major architectures and OSes, no installation or setup is required
 - ❖ majority of off-shell monitoring tools written in Go: Prometheus, VictoriaMetrics, InfluxDB, etc.
- ❖ GGUS, SSB parsers, annotation and alert managers, NATS sub/pub, monit tool to fetch data from MONIT data-sources (ES, InfluxDB), token manager for CERN SSO authentication, etc.
 - ❖ all tools available on `/cvmfs/cms.cern.ch/cmsmon`
- ❖ We also successfully use Prometheus and VictoriaMetrics tools (e.g. amtool, vmagent) in CMS workflows

CMS Spider

HTCondor schedds \Rightarrow Processing \Rightarrow ElasticSearch



Rumble service for HDFS queries

```
for $doc in json-file(  
  "/project/monitoring/archive/wmarchive/raw/metric/2021/01/01/*"  
)  
where $doc."data"."meta_data"."jobstate" eq "failed"  
group by $task := $doc."data"."task"  
where count($doc) ge 1  
return {  
  "task name": $task,  
  "count": count($doc)  
}
```

Rumble HTTP end-point

HDFS+Spark



- ❖ Compose JSON queries over HDFS data
 - ❖ flexible JSONiq Query Language
- ❖ Rumble server (deployed on k8s) can process user requests
- ❖ Hide complexity of HDFS and Spark workflows
- ❖ Python and Go clients are available as well as ability to use Jupyter notebooks

CMS Operational Intelligence

- ❖ Use proper architecture and tools, apply K.I.S.S. principle, e.g. you may not need a fancy ML if simple regex matching can work for your use-case
- ❖ Work on architectural choices and independently test your infrastructure, services, tools, etc.
- ❖ Automate as much as possible, e.g. from git tag (via CI/CD) to k8s deployment
- ❖ Web dashboards are great for visualization but CLI tools can help you manage your services and infrastructure at scale
- ❖ Data schema, validation, common notations, and standards are the keys to success
- ❖ It is possible to manage 500B data points, several k8s clusters just with 1 FTE
- ❖ Work with WLCG and Operational Intelligence groups on common use-cases and provide feedback of various R&D efforts, see Panos' [OpInt talk](#) at vCHEP

Towards observability platform

