Migration of CMSWeb to k8s

Valentin Kuznetsov, Cornell University

pre-GDB meeting

CMS vision for k8s

- CMS envision to gradually move its services to k8s infrastructure
- So far we identified few use-cases: migration of CMSWeb cluster (this talk), setup of Rucio and CMS Monitoring services on k8s infrastructure
- We also see advantage of using k8s for testing various services, perform R&D studies with new technologies and works towards end-to-end deployment procedure on provided infrastructure
- We should not be constrained by particular technology choices and adapt k8s deployment across possible cloud providers
- We are interested in an R&D towards k8s federation, building and managing serverless workloads, FaaS, etc.

CMSWeb cluster

- CMSWeb cluster is a set of VMs to serve large pool of CMS data-services in one common place
  - internally we maintain 3 clusters: production, pre-production and dev one where we mostly use m2.large, m2.xlarge, r2.xlarge, m2.2xlarge and m2.3xlarge flavors
- It uses Apache frontend to perform user authentication and authorization based on X509 certificates
- Dozens of services run on pre-commissioned VMs with chosen hardware configuration to serve service needs, e.g. we use SSD drives for various DBs, VMs with large RAM for memory hungry applications, etc.
  - all machines are puppetized
- We use RPM based deployment procedure with custom shell script and run it on monthly basis for new releases
- One CMSWeb operator performs upgrades, maintain the cluster, apply security patches, etc.

CMSWeb cluster details

Hardware Resources by services				
Service	**# Nodes[*]**	**RAM**	**VCPUs**	**Attached Volumene Disk [**]**
AlertsCollector	1	58.6 GB	32	4 TB
ACDCServer	1	58.6 GB	32	4 TB
DAS	2	117.2 GB	64	No volumes attached.
DBS	5	145 GB	40	1,5 TB
DBSMigration	5	145 GB	40	1,5 TB
CrabCache	1	58.6 GB	32	No volumes attached.
CrabServer	5	145 GB	40	1,5 TB
ConfDB	1	58.6 GB	32	No volumes attached
CouchDB	3	175.8 GB	96	4 TB
DMWMMon	5	145 GB	40	1,5 TB
DQMGUI	2 dedicated	58.6 GB	32	3 TB
Exporters	20 (~all nodes)			
Frontend	6	87.6	44	900
MongoDB	2	117.2 GB	64	No volumes attached.
PhEDEx	5	145 GB	40	1,5 TB
PopDBWeb	5	145 GB	40	1,5 TB
ReqMgr2	5	145 GB	40	1,5 TB
SiteDB	5	145 GB	40	1,5 TB
T0ReqMon	1	58.6 GB	32	4 TB
T0WMADataSvc	5	145 GB	40	1,5 TB
VictorWeb	5	145 GB	40	1,5 TB
WorkQueue	1	58.6 GB	32	3 TB
ReqMon	2	117.2 GB	64	4 TB

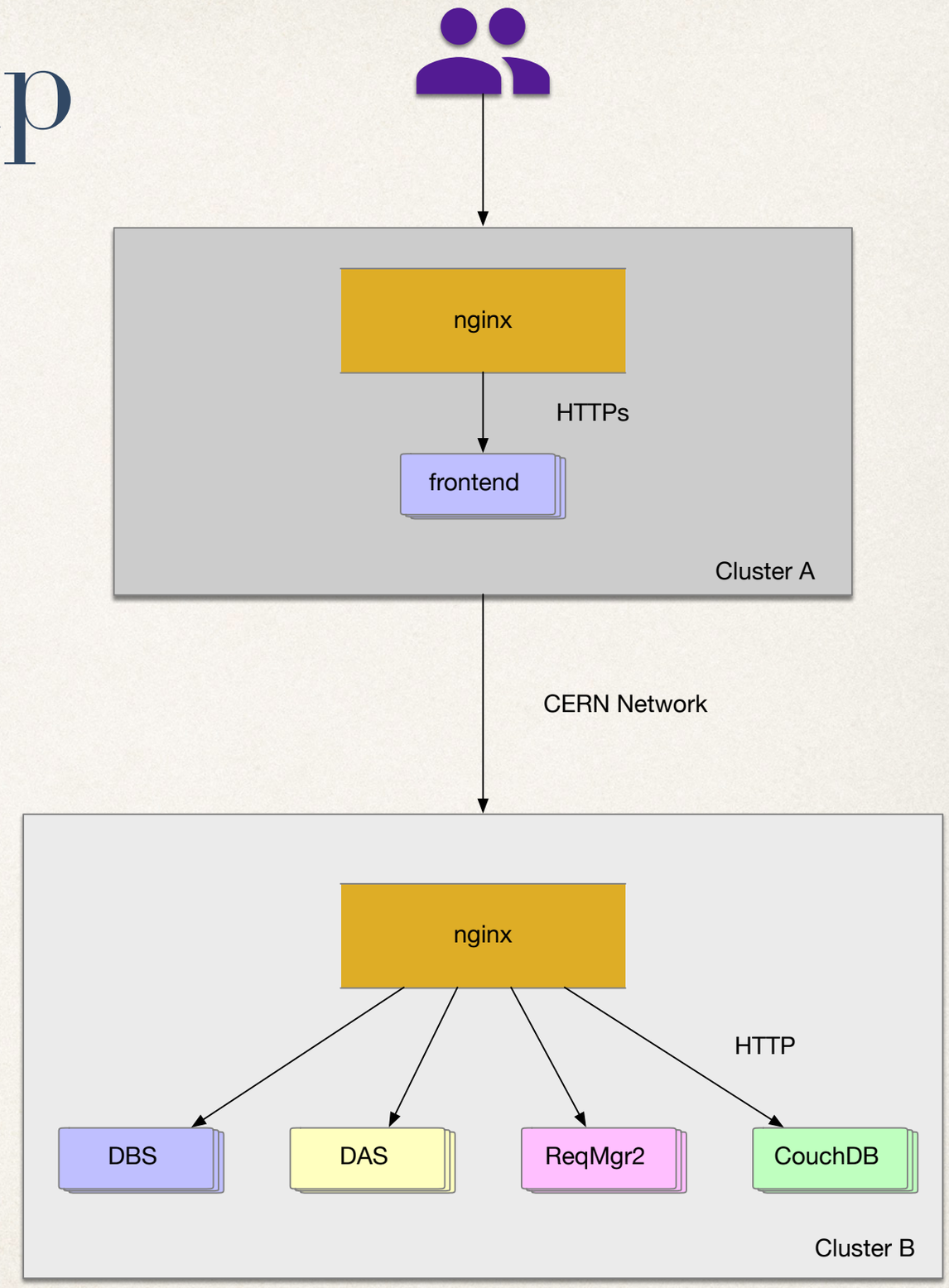
- we use m2.large, m2.xlarge, r2.xlarge, m2.2xlarge and m2.3xlarge flavors
- In total: 310 VCPUs, 641 GB of RAM, 73 TB of disk space, 13 backends, 6 frontends, 22 nodes divided across prod / preprod / test clusters
- services runs on different VMs, we manually allocate resources in a similar way as k8s does

CMSWeb migration to k8s

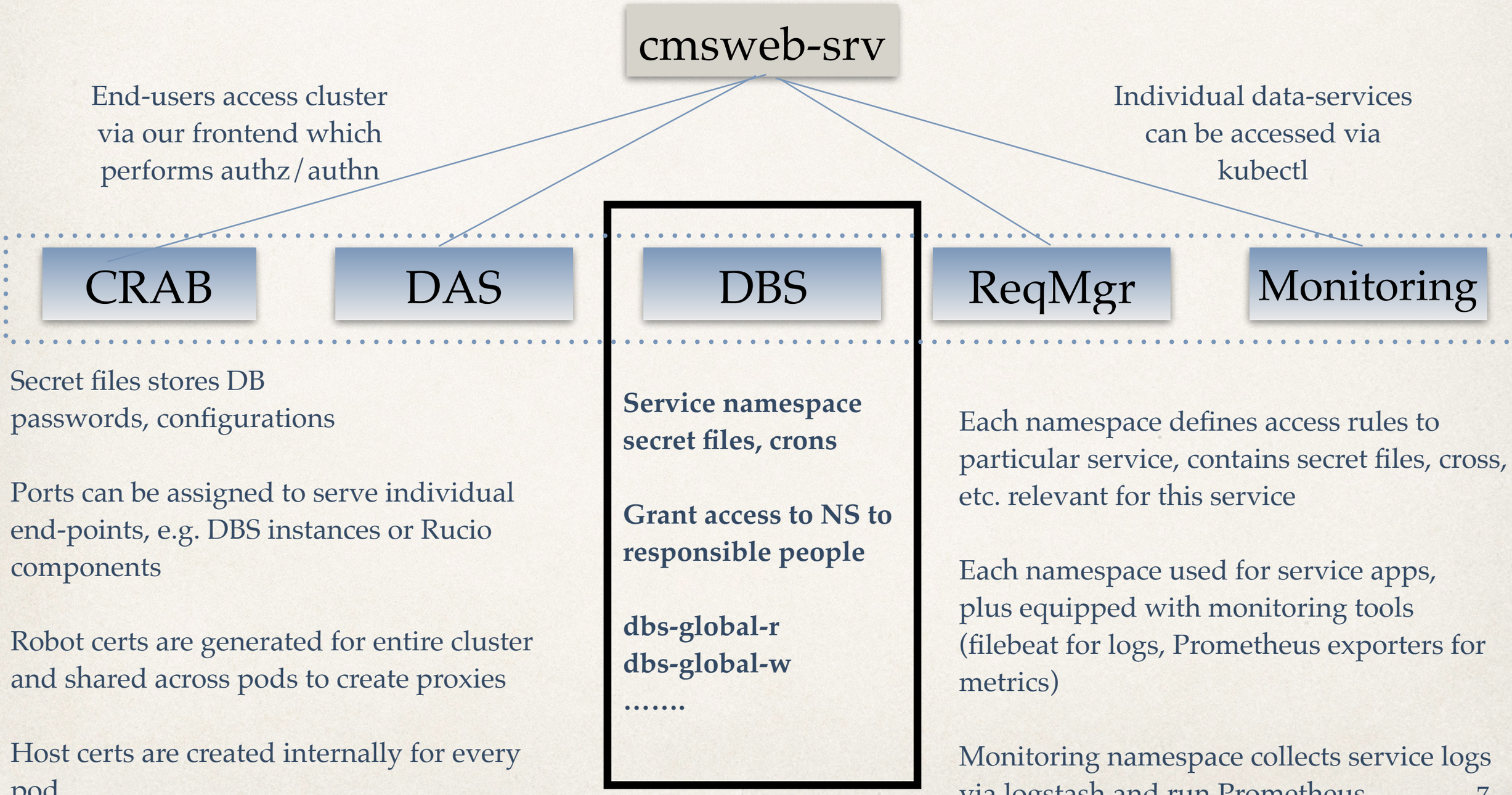
- Started early 2018 with k8s evaluation, gradually acquired the knowledge, and stayed in constant interaction with CERN IT to obtain necessary k8s functionality
  - TLS support was added in mid 2018, switch from traefik to nginx, etc.
- Migration process was started in 2019
  - obtained dedicated quota 400 VCPUs, 1TB RAM and 100TB disk quota on open stack
  - designed cluster architecture; performed CMS k8s training; completed documentation (end-to-end deployment procedure) for CMS operators and developers
  - maintained whole codebase in CMSKubernetes repository
- We reached the phase of full deployment procedure and now in integration phase which involves training, validation tests, cross services interactions, resources allocation, etc.

CMSWeb k8s setup

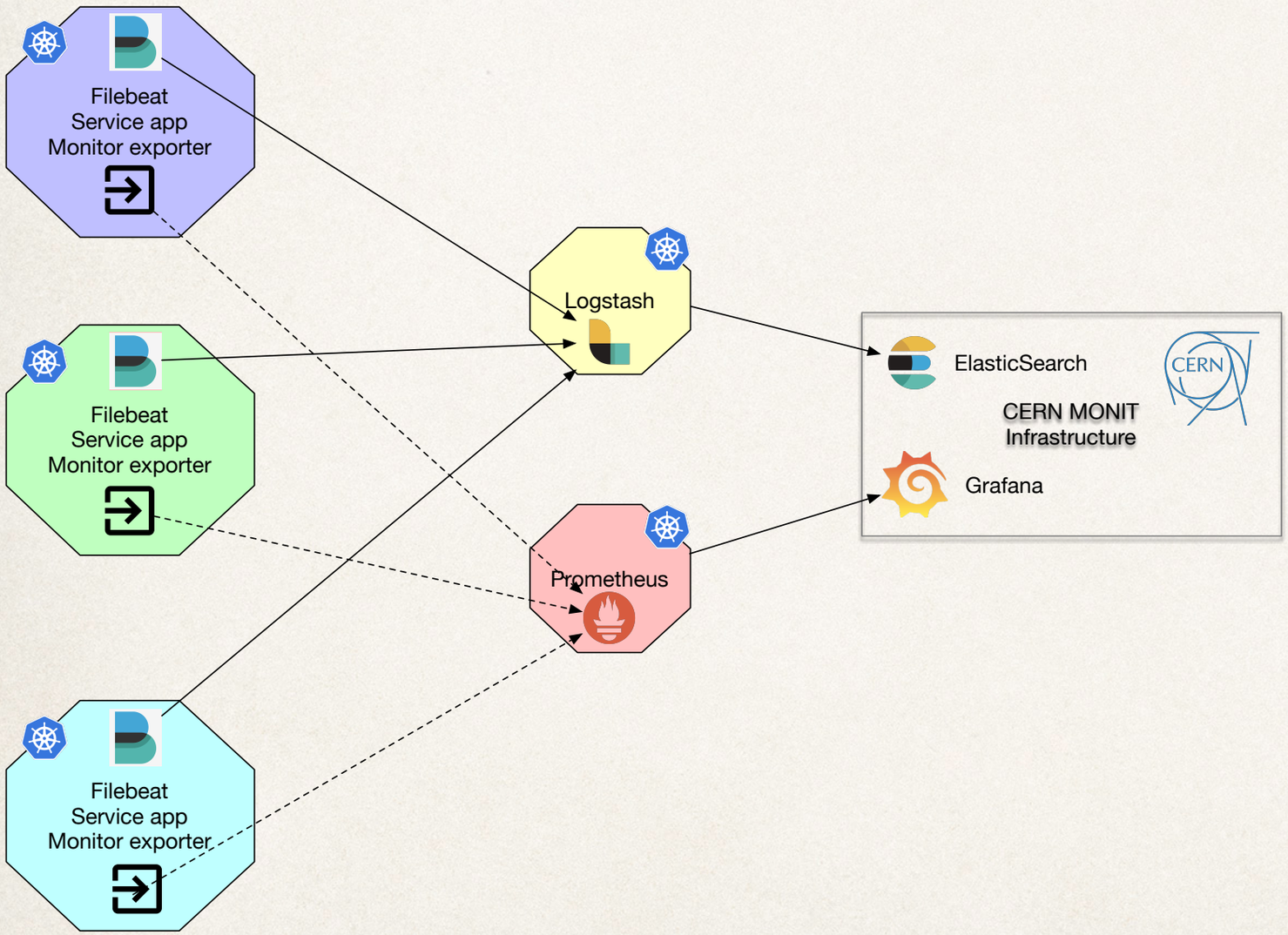
- Due to existing constrains (our FE auth/authz schema) we settled on the following configuration
  - frontend cluster to host our apache frontends (they contain explicit redirect rules to BE domain)
  - service cluster to host our services
  - inter cluster communication is strictly restricted via IP whitelists
  - we explored host networking option too, but it didn't provide proper scaling
- We use three tier architecture: nginx load balancer with TLS passthrough, cmsweb frontend for authentication and cmsweb backend services to complete user requests



CMS data-services/namespaces



cmsweb k8s monitoring



- Every cmsweb data-service produces logs in
  - `/data/srv/logs/<srv>`
- The `monitor.sh` scripts contains `filebeat` part which will scrape logs (configurable via `filebeat.yml` configuration file) and send them to `logstash` service
- Logstash service collects all filebeats and send them to CERN MONIT
- A new end-point in monic-timber ES is under validation

CMSWeb k8s benefits

- Simplification of deployment cycle
  - CMS developers can be fully engaged in end-to-end deployment of their services
  - decouple monthly release upgrades into service release schedule
  - fast rollback to previous release in case of service errors
- Reduction of operational costs
  - cmsweb operators no longer need to maintain monolithic deployment
- Service maintenance and auto-scaling
- Service isolation and delegation of responsibilities

Outstanding issues

- **Proper definition of service resource limits**
  - should be defined via integration tests, e.g. how many pods to allocated for data-service, what are the min/max CPU/RAM requirements for it, etc. **Need more flexible auto-scalers.**
- **Persistent storage** for cmsweb services (CouchDB, CrabCache, DQMGUI, cmsweb logs)
  - input from k8s experts: keep DBs out of k8s (probably keep CouchDB out of the cluster)
  - limited shred storage possibilities
- **Very steep learning curve** there are a lot of new concepts, terminology, planing
  - lack of end-to-end examples within clouddocs.web.cern.ch
- **Debugging k8s issues** so far requires lots of knowledge, easy to miss namespace, or make a typo in manifest files, etc.
- **We lack of production operational experience with k8s**, need to build and share it
- **Agree on level of support**

Suggestions

- Improve documentation, e.g. provide reproducible end-to-end examples of k8s deployments
- Provide more training and guidance, e.g. which tools/middleware/concepts to use (traefik vs nginx, Cephfs vs Block storage, hemp vs k8s, etc.)
  - good practices: namesapces, nginx vs traefik, secrets vs configmap, etc.
- Identify priorities based on experiment needs
- Limited user quota on OpenStack constrains user adaptation of new technology, e.g. to deploy a single cluster user required to have at least 2 nodes (master+minion) out of 5
  - Increase number of instances to allow users to have more nodes within their quota
  - if possible double RAM quota and increase VCPUs

Our areas of interests

- Database as a Service (aaS)
- Autoscalers with custom metrics, e.g. based on Prometheus metrics
- k8s persistent volumes, maintenance and scaling
- F2F k8s training, both at experiment and IT level
- Federated k8s
- Common k8s auth layer
- CRD (Custom Resource Definition), OperatorHub, work towards deployment simplification
- Dynamic provisioning of k8s clusters at CMS sites
- On-demand analysis facilities