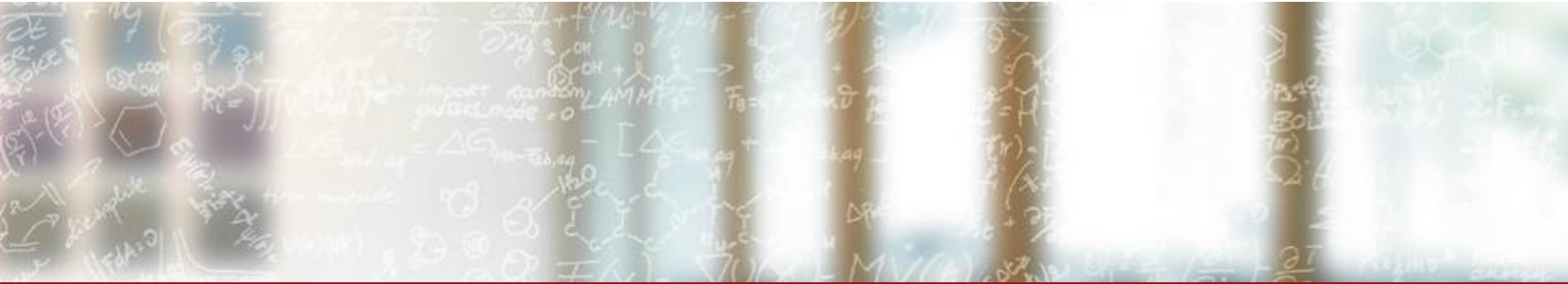




CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



dCache on Kubernetes

HEPiX – University of Oklahoma (Norman, OK, USA)

Elia Oggian, System Engineer, CSCS

November 05, 2024

Table of Contents

1. dCache evolution at CSCS
2. Current status
 - Overview
 - Storage backends
3. dCache on Kubernetes
 - Architecture
 - Docker image
 - Helm Chart
 - HA Zookeeper + HA PostgreSQL
 - Fetch CRL
 - Challenges
4. dCache deployment with ArgoCD
5. Future developments
6. Demo (dCache Upgrade)



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

dCache evolution at CSCS

dCache evolution at CSCS

- Pre SAN
 - SE nodes directly attached to Storage Systems (in pairs)
 - This allowed, with minor config changes, to bring up pools of node-A to node-B in case of failure of node-A (and vice versa)
- SAN
 - All SE nodes were able to see all LUNs and bring up pools of a failed node to any other node
- SAN + GPFS (2017)
 - All SE nodes were part of a GPFS cluster
 - Pool data was stored in a GPFS filesystem, any pool was able to be started on any node
- Ceph + Kubernetes (2021)
 - Ceph as storage backend (RBD)
 - dCache running on a Kubernetes cluster (Helm Chart)
 - Core domains, doors and pools can move across nodes



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Current status

Current status (Overview)

- 24 Physical Nodes
 - AMD EPYC 64c, 256GB, 2x 100G, RHEL 9.2
- RKE2 Kubernetes Cluster (v1.29)
- 5 dCache v9.2 instances running on the same K8S cluster
 - DEV (CSCS)
 - TDS (CSCS)
 - WLCG (Worldwide LHC Computing Grid – T2 for ATLAS + CMS + LHCb)
 - CTA (Cherenkov Telescope Array)
 - SKA (Square Kilometre Array)
- ~ 300 dCache Pools (48 TiB each)
- Ceph Backend (EC 4+2)
- Postgres Operator (Crunchy)
 - S3 backups (ceph + tape)
 - HA
- Zookeeper HA

Current status (Storage backend - NARET)



- Ceph Reef (v18.2.4)
 - 29 PiB RAW HDD space (1836 OSDs)
 - 700 TiB RAW NVMe space (224 OSDs)
- 3 Monitor Nodes
 - MON + MGR + MDS
 - 2x 25G NIC
- 51 OSD Nodes
 - 36x 18TB (JBOD)
 - 2x 7.68TB NVMe SSD (RocksDB + WAL)
 - 1x 14TiB + 1x 3.84TiB NVMe SSD (Fast device class)
 - 2x 100G NIC
- 3 RGW Nodes (S3)
 - HAProxy Load Balancer
 - 2x 100G NIC
- Erasure Coding (EC) 4+2
 - 66.41% efficiency
 - Max 2 host failures
- 2 RBD images per pool
 - 48 TiB data on HDD
 - 2 GiB metadata on NVMe

Current status (Storage backend - TOM)



- Ceph Quincy (v17.2.7)
 - 11 PiB RAW HDD space (552 OSDs)
- 3 Monitor Nodes
 - MON + MGR + MDS
 - 2x 25G NIC
- 23 OSD Nodes
 - 24x 22TB (JBOD)
 - 1x 6.4TB NVMe SSD (RocksDB + WAL)
 - 2x 100G NIC
- Erasure Coding (EC) 4+2
 - 66.41% efficiency
 - Max 2 host failures
- 1 RBD images per pool
 - 48 TiB data on HDD



CSCS

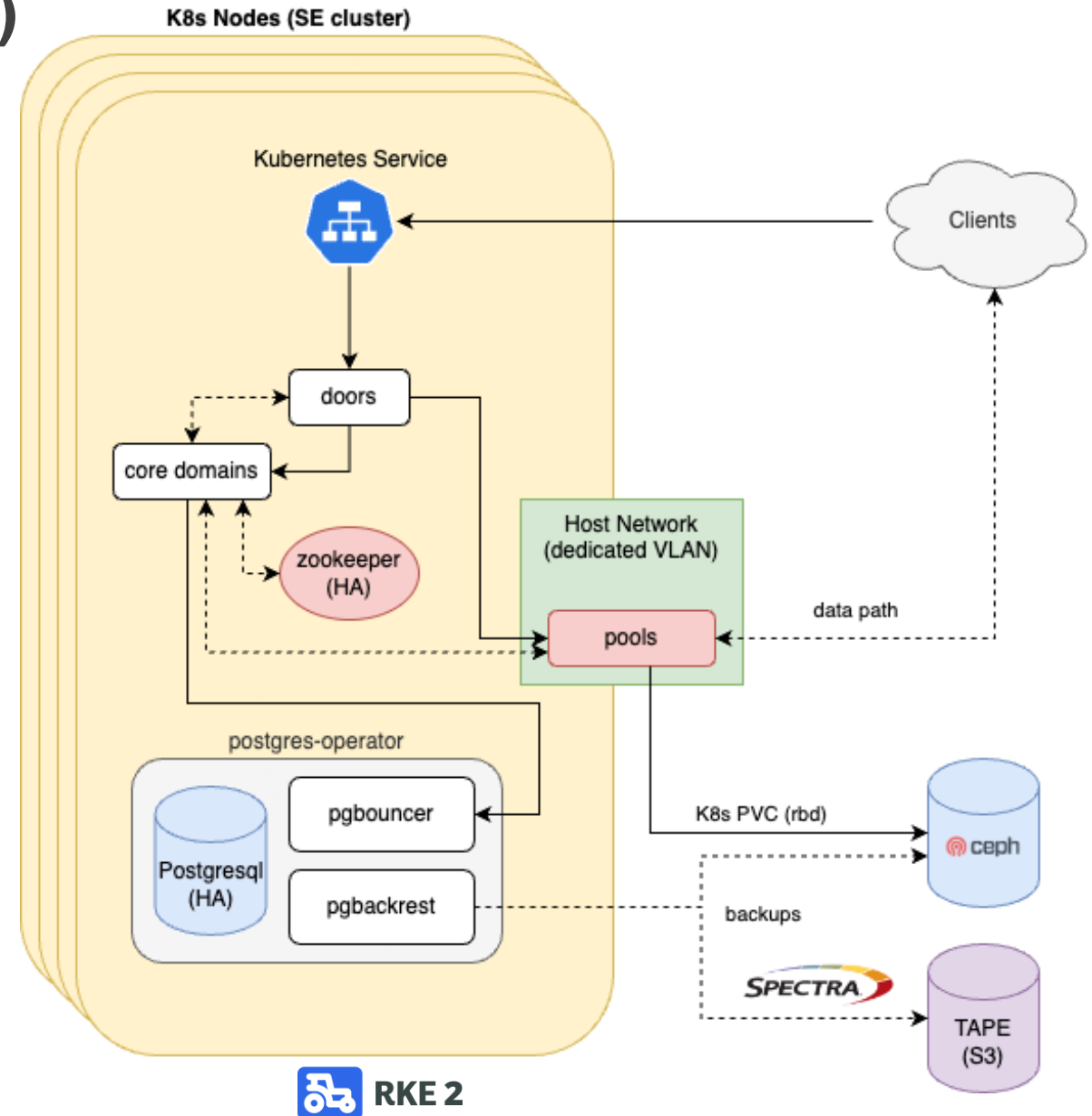
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

dCache on Kubernetes

dCache on Kubernetes (Architecture)

- Pool pods with HostNetwork
 - Public IP
- NGINX Ingress Controller
 - TCP services for doors
 - Ingress for HTTPS services
- Cilium CNI (IPv4 + IPv6)
- Node failover in case of failure
- Logs and metrics collection
 - Filebeat + Metricbeat



dCache on Kubernetes (Docker Image)



- Based on [cern/alma9-base](#) image
- Download specific version of dCache release (.tgz)
- Install Java OpenJDK
- Install WLCG and CTA packages (LSC + VOMS files)
- Unpack dCache
- Install S3 HSM Plugin
- Copy entrypoint script
 - Filter local IP addresses
 - Replace environment variables in configuration files
 - Exec Java VM (run dCache)

dCache on Kubernetes (Helm Chart)

- Kubernetes resources
 - StatefulSet (replica 1)
 - ConfigMap (layout.conf)
 - Secret (certificate)
 - Service (for core cells and doors)
 - PersistentVolumeClaim
- Extras
 - Global ConfigMap (dcache.conf)
 - Certificate for Ingress
- Chart Values
 - Global dCache Configuration
 - List of core domains and doors
 - List of pools

```
pools:  
- name: pool-k8s-dev-hsm-dteam01  
  vo: dteam  
  hsm: true  
  configmap:  
    data:  
      layout.conf: |  
        [pool-k8s-dev-hsm-dteam01]  
        [pool-k8s-dev-hsm-dteam01/pool]  
        pool.name=pool-k8s-dev-hsm-dteam01  
        pool.path=pool-k8s-dev-hsm-dteam01  
        pool.size=32000000000  
        pool.wait-for-files=${pool.path}/data  
  volume:  
    mountPath: "/pool/pool-k8s-dev-hsm-dteam01"  
    storageClassName: "ceph-naret-se-wlwg"  
    size: 32Gi  
  statefulset:  
    hostNetwork: true  
  service: {}  
  
- name: pool-k8s-dev-hsm-dteam02  
...  
...
```

dCache on Kubernetes (Fetch CRL)

- Docker image is built daily to get latest CA packages
- CronJob runs each 4h in the cluster on the latest image
 - Saves the CRLs in a Persistent Volume (PV)
- RWX Persistent Volume (CephFS)
 - Read-Only mounted on all the pools and domains (/etc/grid-security/certificates)

dCache on Kubernetes (HA Zookeeper + HA PostgreSQL)

- Bitnami Zookeeper Helm Chart

- HA with replica 3
- <https://artifacthub.io/packages/helm/bitnami/zookeeper>



- Crunchy Postgres Operator

- HA with replica 3
- Manage multiple PostgresCluster resources
- HA DB connectivity (pgbouncer)
- Scheduled backups to S3 endpoints (pgbackrest)
 - Ceph RGW
 - Spectra Logic BlackPearl
- <https://access.crunchydata.com/documentation/postgres-operator/latest>



dCache on Kubernetes (Challenges)

- Tested different network implementations
 1. BGP at CNI level (calico), pool pods with public IPs
 2. LoadBalancer Service with public IP for each pool (with hundreds of ports 🌀)
 3. Host Network on pool pods (current)
- Network issues/workarounds
 - Filter Kubernetes IPs and loopback addresses at startup (entrypoint script)
 - Reverse DNS resolution not working → Custom coreDNS configuration
- Storage
 - Slow metadata operations will kill the pool
 - Separate metadata PVC on NVMe backend
 - Symlinking **meta** directory with an init container (config option available in dCache 10)

dCache on Kubernetes (more challenges)

- Domain startup order
 - Core domain must start before the others
 - Single core domain cell with a Service pointing to it
 - initContainer on all other cells to wait for the core svc to be ready
- dCache Core address
 - By default the pod IP gets registered in Zookeeper, but the IP changes at pod restart
 - Set fixed core address using this option on the core cell
 - Dorg.dcache.net.localaddresses = dcache.ns.svc.cluster.local
- Read DB password from Secret
 - Use ENV variable and substitute it at startup (fix available in dCache 10.0)

dCache on Kubernetes (even more challenges)

- Restart Pods on changes to ConfigMap mounted with subPath
 - <https://github.com/stakater/Reloader> to the rescue!
- Pod limits have negative effects on performances
 - CPU limits
 - Throttling causes a lot of context switches (impacting performances, no benefits)
 - Memory limits
 - Can cause the pods to get OOM Killed (if misconfigured)
 - Dramatically impacts performances (still under investigation)
- Kubelet counts active page cache against memory.available of the node
 - This can cause unnecessary node drains (pods get restarted)
 - Reserved more memory for the system to avoid exceeding the threshold



CSCS

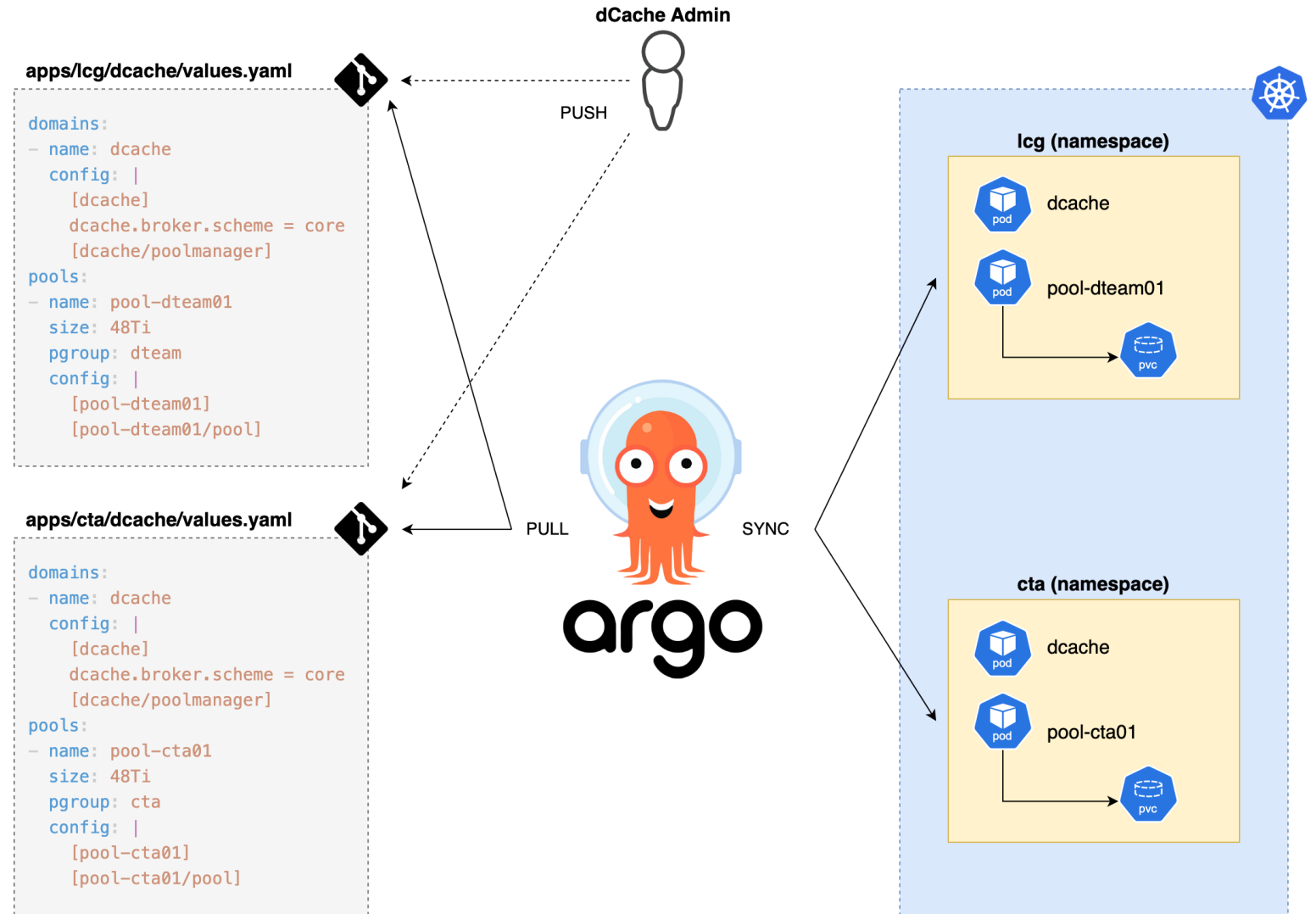
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

dCache deployment with ArgoCD

dCache deployment with ArgoCD

- GitOps Model
 - Pull requests for changes
- Declarative
- Replicable
- State enforcement
- Self-Healing





CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

Future developments

Future developments

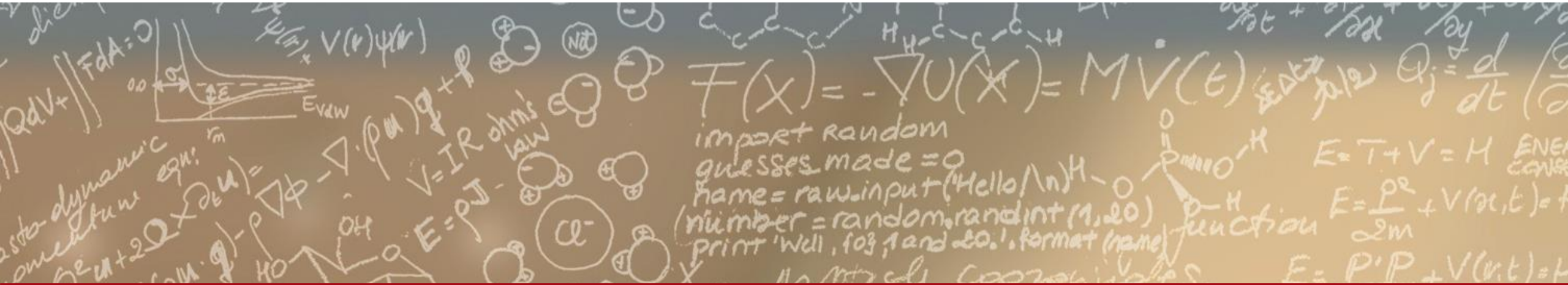
- Move to official dCache image (most likely on dCache 10.2)
- Single Statefulset of Pools with n replicas
 - One statefulset per VO
 - Rolling deployment
- Improve overall robustness and flexibility
- Simplify Helm Chart values (reduce repetition)
- Improve Monitoring
 - Prometheus exporter for dCache metrics
 - DB monitoring (pgmonitor)



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Thank you for your attention.