Automated Provisioning of Virtual Workstations and Servers for a Broad Audience of Researchers¶

By Roberto Fabbretti DCSR UNIL

UNIL | Université de Lausanne

Division de Calcul et Soutien à la Recherche

Members

Ewan Roche

Etienne Orliac

Emmanuel Jeanvoine

Philippe Jacquet

Thierry Lombardot

Marco Barroso

UNIL | Université de Lausanne

Context

- UNIL hosts 7 faculties
  - Faculté de théologie et de sciences des religions (FTSR)
  - Faculté de droit, des sciences criminelles et d'administration publique
  - Faculté des lettres
  - Faculté des sciences sociales et politiques (SSP)
  - Faculté des hautes études commerciales (HEC)
  - Faculté de biologie et de médecine (FBM)
  - Faculté des géosciences et de l'environnement (GSE)

  Only 2 of which are heavy HPC consumers

HPC Ressources @UNIL

- DCSR
  - Curnagl (new)
    - 72 nodes
    - 3456 cores
    - 32 TB RAM
    - 16 GPU
    - 1.5 PB GPFS storage
  - Wally (very old)
    - 96 nodes
    - 1536 cores
    - 6.1 TB RAM
    - 346 TB BeeGFS storage
  - Jura (air gapped cluster for sensitive data)
    - 15 nodes
    - 1000 cores
    - 10 TB Ram
    - 100 TB SSD BeeGFS storage

- FGSE
  - Achilles
    - 592 cores
    - 3.8 TB RAM
    - 8 GPU
    - 120 TB storage
  - Octopus
    - 428 cores
    - 6 TB RAM
    - 288 GPU
    - 308 TB Beegfs storage

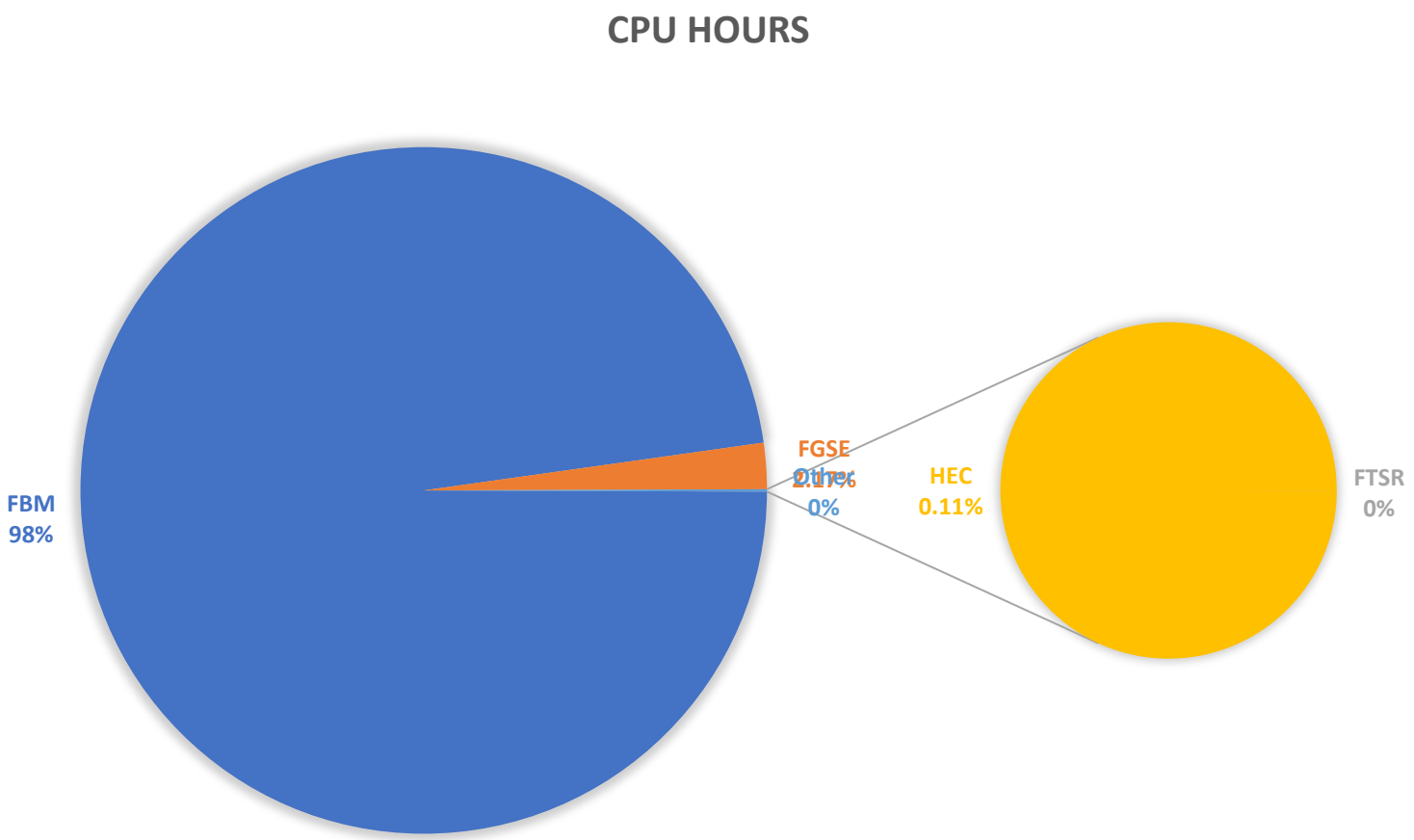
- SIB
  - Biomed IT (Openstack for sensitive data)
    - 400 cores
    - 4.6 TB RAM
    - 2 GPU
    - 100 TB WekaIO SSD storage

UNIL | Université de Lausanne

DCSR ressources used by faculty

Computational resources

Faculty	CPU usage
FBM	97.72%
FGSE	2.17%
HEC	0.11%
FTSR	0.00%



How to provide “cost effective” scientific computing

- For some users command line, scripting in bash or python is considered an insurmountable obstacle.
- Some users need an interactive system while exploring solution spaces
- Some software are only available for Windows systems
- Some users need a full IDE with configurations compatible with the clusters

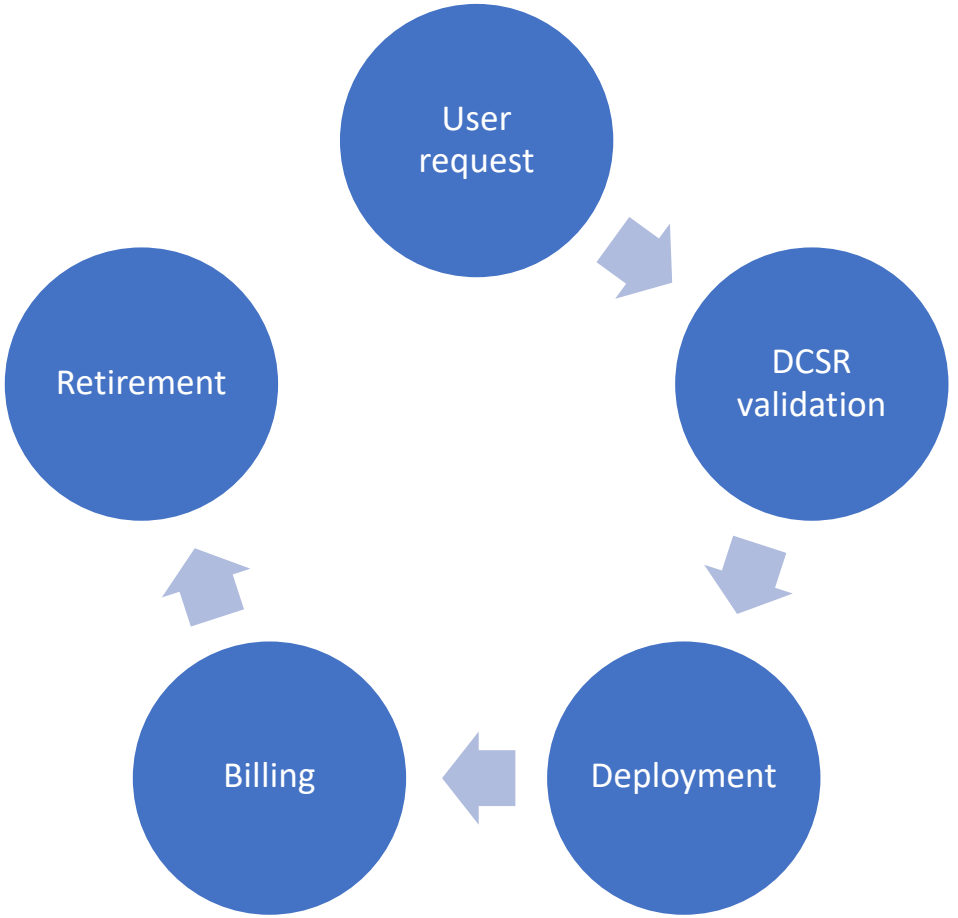
The goal is to prevent the creation of mini datacenters under the desks of the researchers which usually remain idle

Proposed solution

- Provide virtual workstations and servers hosted centrally
- Create a procedure to automate the deployment of the above mentioned virtual systems
- Project based automated billing for the user resources consumption
- Encrypted filesystems for system images and user data to manage sensitive data
- Per project isolated subnets

VM lifecycle

The only manual intervention is the validation of the funding



Technical outline

- Vmware hypervisor
- Vmware horizon VDI infrastructure
- Commvault Hedvig filesystem to host VM images repository
- Weka.io NVMe based filesystem to host research data
- Scality S3 object store as tiering and snapshotting solution for Weka filesystems

Storage back ends

- 3 storages infrastructures with different purposes of use
  - VM image storage --> Hedvig
  - Fast and secure data processing --> WekaIO
  - Large fat and secure data repository --> Scality
- Common features
  - All software defined
  - Independent of the underlying hardware
  - Run on a Vanilla CentOS
  - Perpetual licenses
  - Benefit on the performance evolution of the underlying hardware
  - Can be deployed on premise or on cloud infrastructures
  - Have been designed from scratch to be massively scalable in respect of:
    - storable data volume
    - number of files
  - Support workflow automation through REST APIs
  - Support encryption
  - Have enterprise grade support with large commercial or government clients operating at massive scale
- With our foreseen budgets, even with several dozen of PBs and dozen of billions of files we will never be able to go out of the technical comfort zone of these storage systems

Hedvig filesystem <-- VMs boot image

- Massively scale out storage cluster (3 --> 1K+ storage nodes)
  - Multi-site synchronous data replication (1-6 sites)
  - Strong consistency, partitioning tolerant (CAP theorem)
  - 2 tiers HDD/SSD nodes and/or full SSD nodes
- Virtual volumes (server side) ->
  - DC/Rack aware Replication/Compression/Deduplication
  - HD/SSD cache or Pin to Flash
  - Quota management
  - Instant metadata-based snapshots & writable clones
  - REST API first management (+GUI & CLI)
- Client-side agents (stateless)
  - Present data on NFS v3 v4 / iSCSI / S3 protocols
  - Manage local NVMe caches
    - Local node metadata copy
    - Local node deduplication cache
    - Local node fast data read cache
  - KMIP based client-side data encryption
  - Support Docker Swarm & CSI for Kubernetes

- CAVEATS
  - Massively parallel but limited BW per data stream
  - Distributed over 3 datacenters --> limited minimum latency
  - Caching optimized for Read access
  - No write commit until data is written on at least 2 storage nodes

- Scalability
  - Starting with 6 storage nodes dispatched symmetrically on 3 datacenters (150 TB raw HDD)
  - Connected on a leaf/spine network (2x25Gb bounded/2x40GB bounded)
  - Will grow as needed by adding HDD/SSDs in nodes then adding new HDD or full flash nodes
  - Several TBs of fast NVMe client-side cache per client node (Samsung PM1735, 8 GB/s read , 3.5 GB/s write, 1.5M IOPS)
- Data efficiency
  - Each VM and VDI instance receive its own image
  - OS and common applications are heavily globally deduplicated (80 % to 90%)
  - Since deduplication is done client side before encryption, encrypted data is also globally deduplicated and compressed
  - Most of OS and applications (high Read low Wright) data is kept in the read client-side NVMe cache
- Security
  - Tenants & Volumes based data partitioning
  - Tenants & Volumes based data encryption on flight and at rest
  - Tenant specific role-based access & KMS
- Manageability
  - Automation workflows via REST API
  - Tenant based quota management
  - Resizable thin provisioned volumes
  - No downtime upgrade
- Multi-Platforms integration
  - One storage fits all less storage platforms lower admin/operation time
  - Vmware: optimized NFS, VAAI plugin, Hedvig mgmt plugin for Vcenter
  - Open Stack: Cinder volumes plugin backed by Hedvig virtual vol & functionalities
  - Docker Swarm: docker pull hedvig/hedvig-proxy multi-site replicated persistent volumes in a dev/ops environment
  - Kubernetes: Vanilla 1.13-1.19 , OpenShift 4.1-4.6 , D2iQ Konvoy

WekaIO filesystem <-- datasets processing

- Massively scale out storage cluster (6 --> 1K+ storage nodes)
  - Strong consistency and availability (CAP theorem)
  - 2 tiers: local NVMe Flash <--> external S3 storage
  - N+2 or N+4 data integrity protection
- Virtual volumes (server side) ->
  - Tenant and Volume Quota management
  - File or volume-based snapshots
  - S3 replicable snapshoots for DR
  - Present data on NFS / SMB / S3 / HDFS protocols
  - REST API first management (+GUI & CLI)
- Client-side agents
  - Kernel module
  - Present a fast POSIX compliant filesystem
  - Verify data integrity
  - KMS/KMIP based end to end data encryption
    - On flight
    - At rest
    - XTS-AES 512 bit (256 bit effective)
- Support SMB encryption
- Support CSI plugging for Kubernetes

- Scalability
  - Starting with 8 storage nodes (196 TB raw SSD)
  - 3x 8TB NVMe flash per node
  - Connected on/with 100 GbE Mellanox cards and Switches
  - Will grow as needed by adding SSDs in nodes then adding new nodes
- Performance
  - SPEC SFS2014 score on a 6 nodes cluster
  - Based on an ESG 2020 report (to be confirmed on our infrastructure!)
    - Engineering Design = 2000 with 0.26 ms overall Response Time
    - Database = 4480 with 0.34 ms overall Response Time
    - Software Build = 5700 with 0.48 ms overall Response Time
    - Video Streams 6800 with 1.56 ms overall Response Time
- Security
  - Tenants & Volumes based data partitioning
  - Secure mount authorization via authentication token
  - Tenants & Volumes based data encryption on flight and at rest
  - Tenant specific role-based access & KMS
- Manageability
  - Configurable automated tiering on S3 storage
  - Automation workflows via REST API
  - Tenant and Volume based quota management
  - Resizable thin provisioned volumes

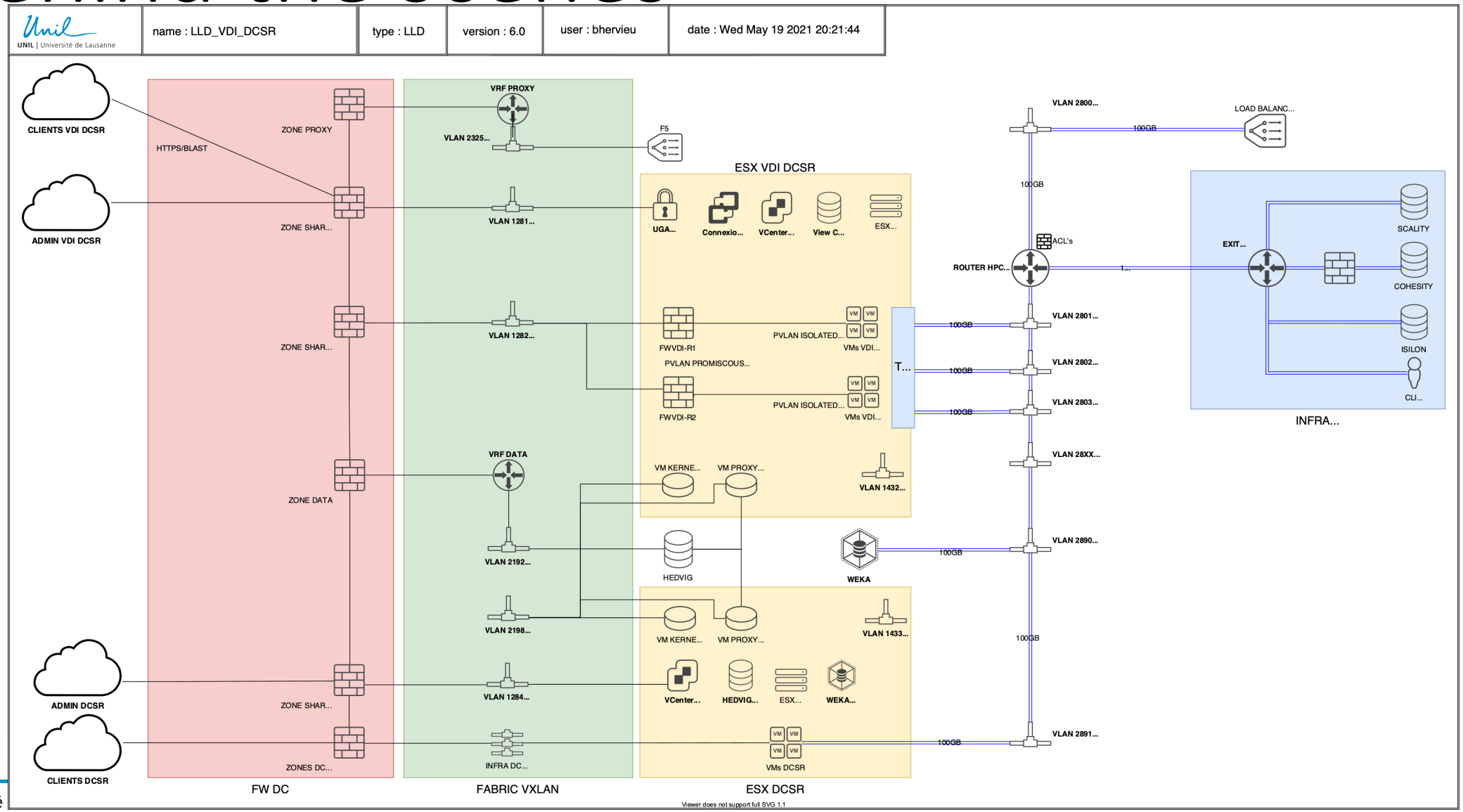
Scality object storage <-- datasets repository

- Massively scale out storage cluster (3 --> 1K+ storage nodes)
  - Multi-site synchronous erasure coding and/or data replication
  - Strong consistency on non-replicated infrastructures (CAP theorem)
  - Either full HD or full Flash
  - Storage nodes + protocol nodes architecture
  - Adding storage node increases the storage capacity
  - Adding protocol nodes increases the data BW
- Storage nodes->
  - Host a sorted massively scalable DHT
  - No central metadata server
  - Data lookup based on the CHORD P2P protocol
  - REST API / CLI and GUI based management
  - Cross region replication with Amazon and other S3 infrastructures
- Protocol nodes
  - Present data on NFS v3 v4 / SBM v2 / S3 protocols
  - KMIP based client-side data encryption

- CAVEATS
  - Massively parallel but limited BW per data stream
  - Distributed over 3 datacenters --> limited minimum latency
  - Caching optimized for Read access
  - No write commit until a quorum of distributed write is acknowledged

- Scalability
  - Starting with 12 storage nodes dispatched symmetrically on 3 datacenters (2PB raw HDD)
  - Connected on a leaf/spine network (2x25Gb bounded/2x40GB bounded)
  - Will grow as needed by adding HDD in nodes then adding nodes
- Data durability
  - 3 sites synchronous erasure coding
  - 12 data + 12 parity erasure coding scheme
  - Ability to loss a full datacenter plus any two servers on the remaining datacenters
  - Provides about 15 9 data durability (at the price of 100% data overhead)
- Security
  - Tenants & bucket-based data partitioning
  - Support POSIX rights for NFS and Windows ACL for SMB
  - S3 supports: ACL, Amazon IAM for Users, Groups and Policies
  - Tenants & Volumes based data encryption on flight and at rest
  - Tenant specific role-based access
  - KMS/KMIP based bucket encryption
- Manageability
  - Automation workflows via Ansible and REST API
  - Tenant based quota management
  - No downtime for upgrades (can even take down a full site without service interruption)
- Multi-Platforms integration
  - One storage fits all --> less storage platforms --> lower admin/operation time
  - S3 tiering end point for: WekaIO, backup infrastructure, ELK infrastructure
  - Storage silo: large scientific datasets
  - Docker Swarm + Kubernetes : host docker image repositories

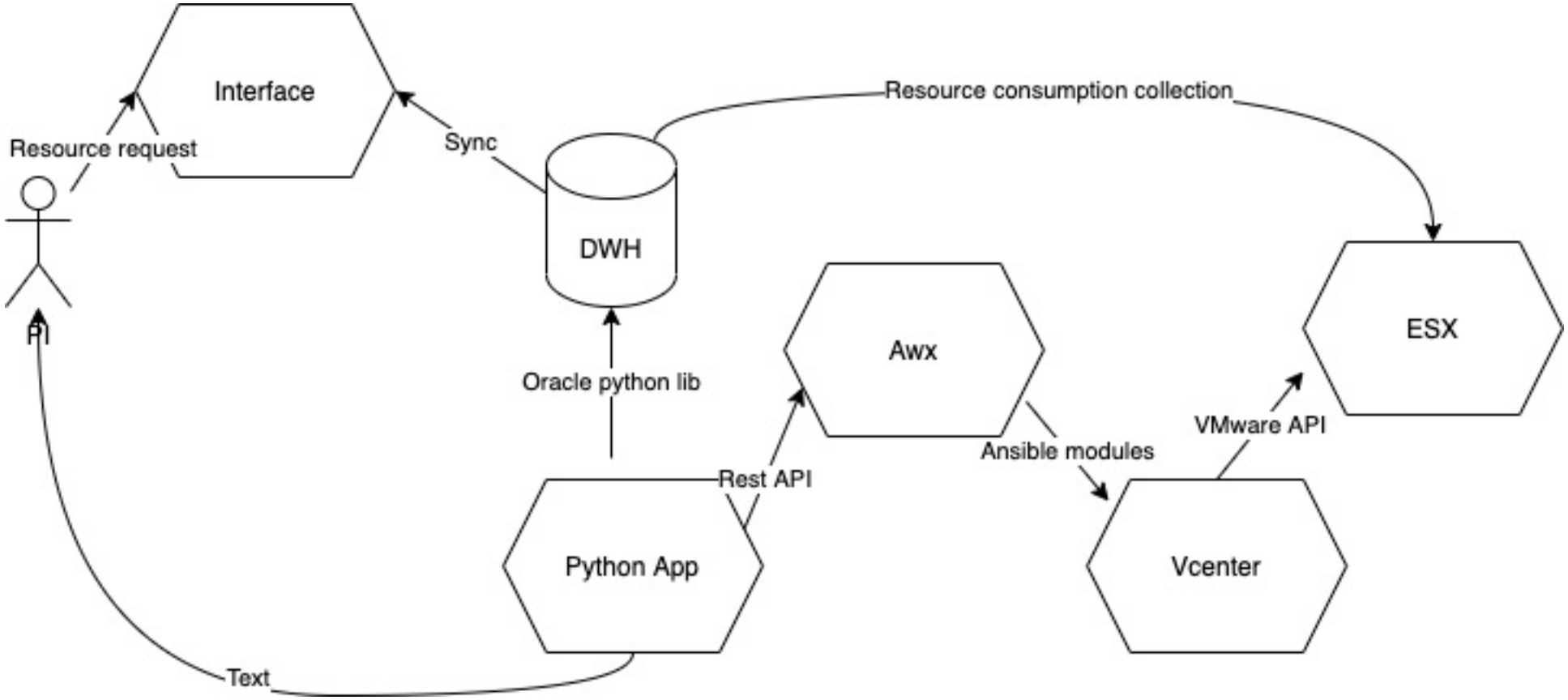
Behind the scenes



User interface

- One interface for requesting all DCSR infrastructure resources
  - The same project can contain storage VM and VDI workstations
  - Automated notification of resource availability
  - PI can manage user access to his resources
  - For SNF and EU projects reference can be found in the bill
  - Links to UNIL DMP generator

Deployment process



What works

- Automatic deployment of VM
- Simple access from a browser or from Vmware horizon client
- CentOS and Windows 10 systems
- Billing per project
- Soon
  - Windows 2019 server VM
  - Encrypted VM images + data filesystems
  - VM isolation with Vmware NSX

What could be better

- Vmware + 100 Gb has issues
  - Packet loss + network instability
- Weka severely impacted from Vmware network issues
  - On a physical server we achieve > 3 GB/s
  - On VM we cannot use SR-IOV + DPDK and therefore are stuck with performances around 1 GB/s
- Users very often need support to install apps or customise setup
- “Cost effectiveness”