

Scale-out With Coffea: Coffea-Casa Analysis Facility

Mat Adamec, Ken Bloom, Oksana Shadura

University of Nebraska, Lincoln

Garhan Attebury, Carl Lundstedt, John Thiltges

University of Nebraska, Holland Computing Center

Brian Bockelman

Morgridge Institute



Building blocks for designing Coffea-Casa



Modern authentication (AIM/OIDC), tokens, macaroons

Efficient data delivery and data management technologies

Columnar analysis and support new python ecosystem

Modern deployment and integration techniques

Support for object storage

Efficient data caching solutions

Easy integration with existing HPC/HTC resources

CMS Analysis Facility @ T2_US_Nebraska



Authorized CMS Users Only!

To login into the Coffea-Casa Analysis Facility, you will need to get a CMS OAuth token.

To get a token you need to a) be member of CMS and b) register with the OAuth service at: [CMS-Auth.web.cern.ch](https://cms-auth.web.cern.ch)

Useful Links

[Coffea-Casa Support Page](#) [Coffea-Casa Docs](#)

News

Watch here for announcements!

Authorized CMS Users Only:
Sign in with CMS SSO

Pain points and Deploying a Coffea-Casa

Modern authentication (AIM/OIDC), tokens, macaroons

Efficient data delivery and data management technologies

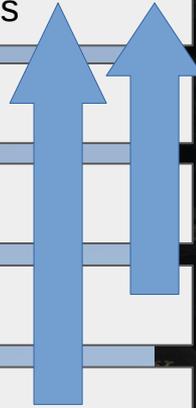
~~Columnar analysis and support new python ecosystem~~

Modern deployment and integration techniques

Support for object storage

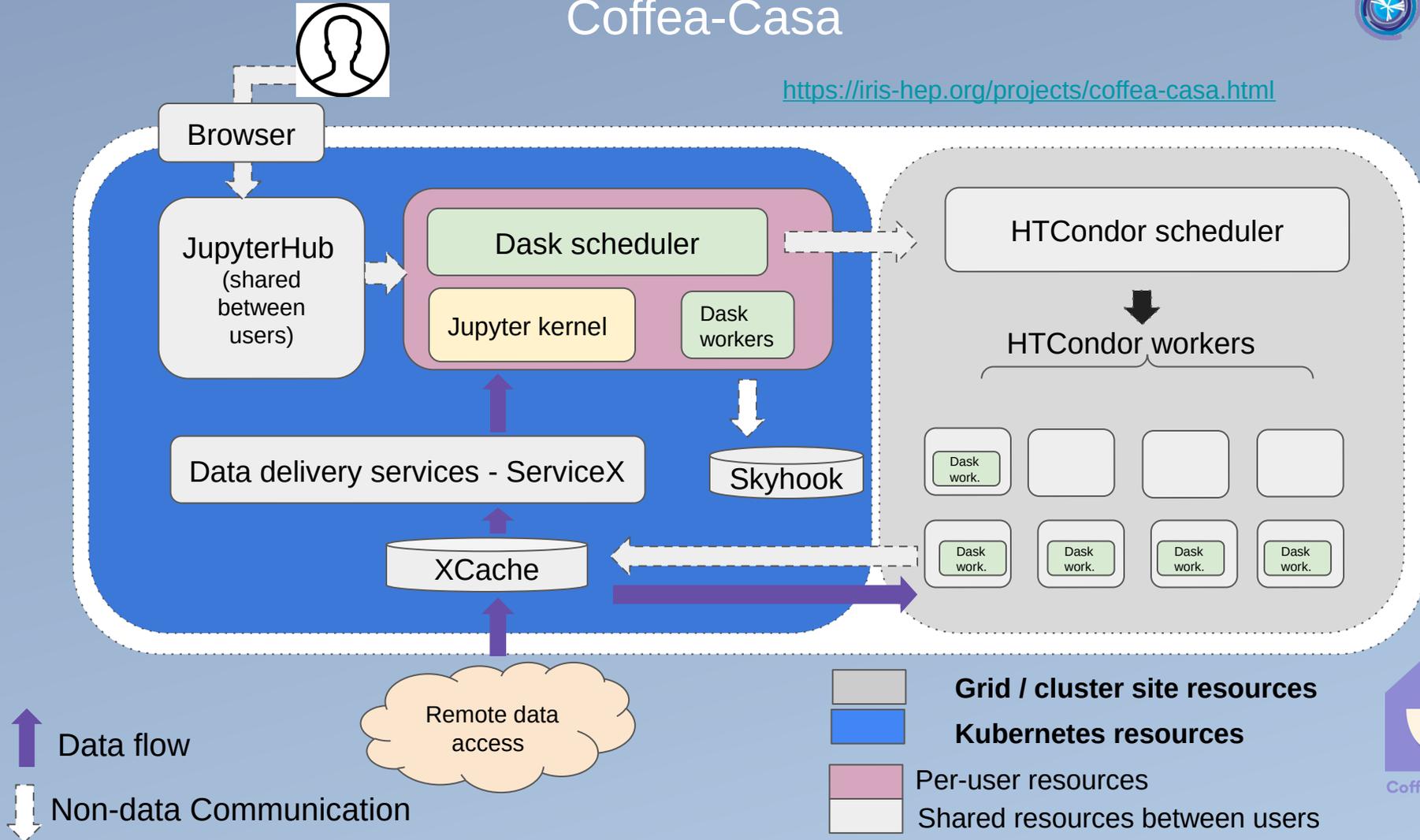
Efficient data caching solutions

Easy integration with existing HPC/HTC resources

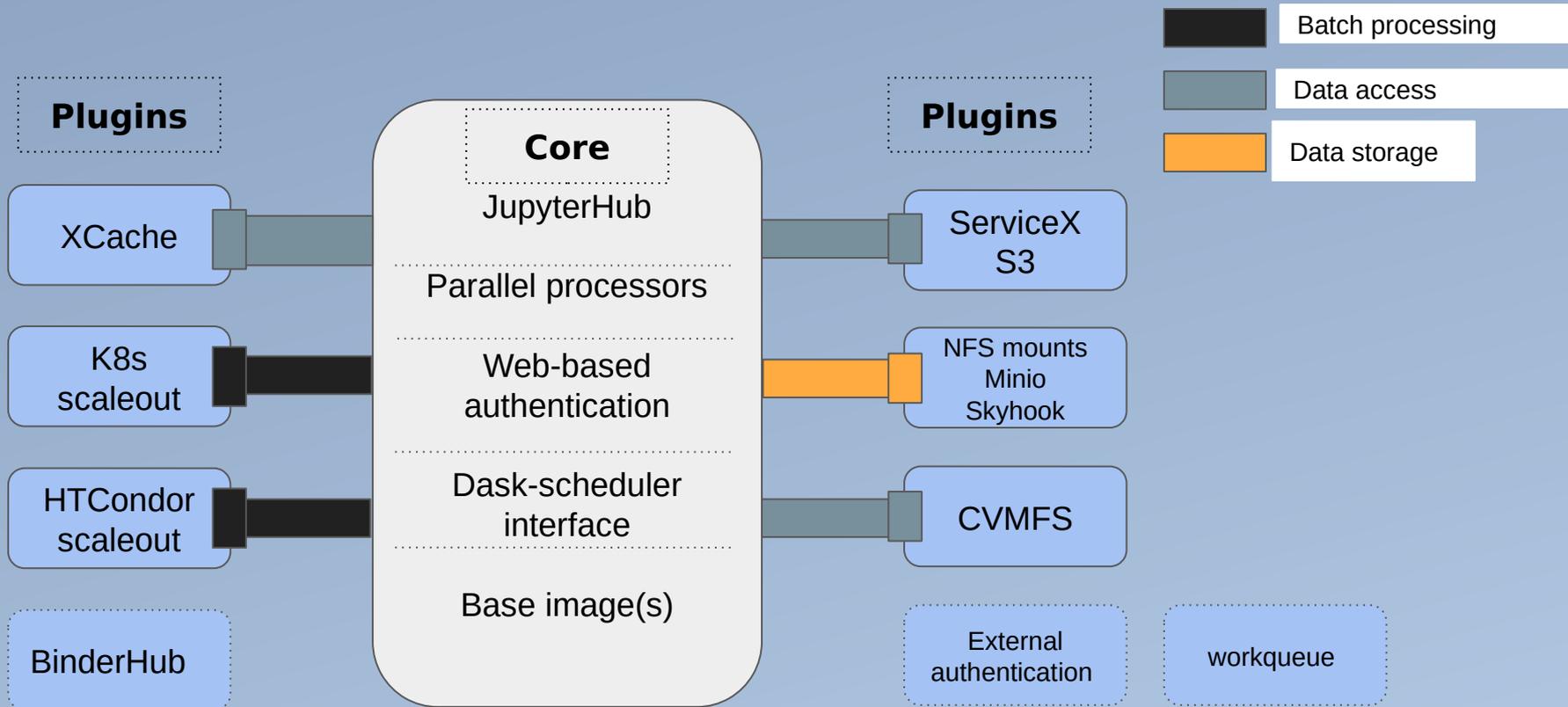


Coffea-Casa

<https://iris-hep.org/projects/coffea-casa.html>



Coffea-Casa: Aiming for Remote Adoption



[Coffea-casa team]

Building blocks: Authentication Tools

Jupyterhub allows for a variety of authentication methods and we inherit this functionality.

Using OAuth we can select an OIDC service to manage users for us.

Dummy authentication is also useful for spinning up test instances.

Each instance must be registered and secrets about that client have to be available in the instance. We seal these secrets so we can store them in git encrypted.

```
values:
  hub:
    extraEnv:
      #OAUTH_CALLBACK_URL: https://cmsaf-jh.unl.edu/hub/oauth_callback
      OAUTH2_AUTHORIZE_URL: https://cms-auth.web.cern.ch/authorize
      OAUTH2_TOKEN_URL: https://cms-auth.web.cern.ch/token
      OAUTH_CLIENT_ID:
        valueFrom:
          secretKeyRef:
            name: jhub-cms-auth
            key: client_id
      OAUTH_CLIENT_SECRET:
        valueFrom:
          secretKeyRef:
            name: jhub-cms-auth
            key: client_secret
```



Welcome to **cms**

Sign in with

Your X.509 certificate

CERN SSO

Not a member?

Apply for an account

You have been successfully authenticated as

CN=Carl

**Lundstedt,CN=514102,CN=clundst,OU=Users,OU=Organic
Units,DC=cern,DC=ch**

<https://cms-auth.web.cern.ch>
<https://cilogon.org/oauth2/register>

Building blocks: OIDC demo



INDIGO IAM for cms

PERSONAL

- Manage Approved Sites
- Manage Active Tokens
- View Profile Information

DEVELOPER

- Self-service client registration
- Self-service protected resource registration



Welcome!

This is the INDIGO Identity and Access Management (IAM) service.

Scope

new scope

- openid
- profile
- email
- address
- phone
- offline_access
- wlwg
- eduperson_scoped_affiliation
- eduperson_entitlement
- storage.read/
- storage.modify/
- storage.create/
- ssh-keys
- compute.read
- compute.cancel
- wlwg.groups

OAuth scopes this client is allowed to request

Grant Types

- authorization code
- client credentials
- implicit
- password
- redelegation
- refresh
- device
- token exchange

NYI Response Types

- code
- token
- id_token
- token id_token
- code id_token
- code token
- code token id_token

values:

```
hub:
  extraEnv:
    #OAUTH_CALLBACK_URL: https://cmsaf-jh.unl.edu/hub/oauth_callback
    OAUTH2_AUTHORIZE_URL: https://cms-auth.web.cern.ch/authorize
    OAUTH2_TOKEN_URL: https://cms-auth.web.cern.ch/token
    OAUTH_CLIENT_ID:
      valueFrom:
        secretKeyRef:
          name: jhub-cms-auth
          key: client_id
    OAUTH_CLIENT_SECRET:
      valueFrom:
        secretKeyRef:
          name: jhub-cms-auth
          key: client_secret
```

Redirect URI(s)

- https://
- https://coffea-dev.casa/hub...
- https://cmsaf-jh-dev.unl.ed...
- http://localhost:8000/hub/o...

Opedata Instance:

```
GenericOAuthenticator:
  scope:
    - openid
    - email
    - org.cilogon.userinfo
  Authenticator:
    allowed_groups:
      - CO:COU:OpenData Facility:members:active
```

Building blocks: Secrets (KubeSeal)

```
apiVersion: bitnami.com/v1alpha1
kind: SealedSecret
metadata:
  name: mysecret
  namespace: mynamespace
spec:
  encryptedData:
    foo: AgBy3i40JSWK+PiTySYZZA9r043cGDEq.....
```

```
1  apiVersion: bitnami.com/v1alpha1
2  kind: SealedSecret
3  metadata:
4    creationTimestamp: null
5    name: jhub-cms-auth
6    namespace: cmsaf-dev
7  spec:
8    encryptedData:
9      client_access_key: AgDAiw07oV/7lyHBstZx0JQpXnf1b5Q9ZHKRIEwf12bFw4f1W68AvhAinxrWq0rMprJVJUoVOMFCCJ3c0w2ojDYzmKal
10     client_id: AgBkc7gx4Vr8Wsn3LmxWinuTG92L/bT+/mBbmtbf1IW0usZk8Mj18B9JMNf+JxzHJO+yU3MNFwk+UhxNo45j1XdweNs9KZqWEZc
11     client_secret: AgARi2XcJMHZiyXdLDxZhU8siwIPI256/V3aiU6owVb9R5D1KSk26+4iljxRwXhPZARXKkq1QHly4y99oVDgcsLkeA7wWay7
12  template:
13    metadata:
14      creationTimestamp: null
15      name: jhub-cms-auth
16      namespace: cmsaf-dev
17    type: Opaque
```

Building blocks: Token Management

Tokens are signed strings that allow for various functionality. The pieces of Casa using/needing tokens are HTCondor and XCache access.

We want to abstract access authentication away from the user so things 'just work'.

We also want to obscure any x509 certs as needed.

Token creation takes place in a custom script that's shipped with the base Coffea-casa codebase.

Sample xcache_token payload:

```
location T2_US_Nebraska  
identifier 46a...  
cid name:cms-jovyan  
cid activity:DOWNLOAD  
cid path:/store  
cid before:2023-04-24T13:59:47Z
```

Condor POOL Password vs. Token

Pool passwords are master keys for your cluster and needed to sign any tokens.

The POOL password is stored in K8s but NOT available in any user's environment.

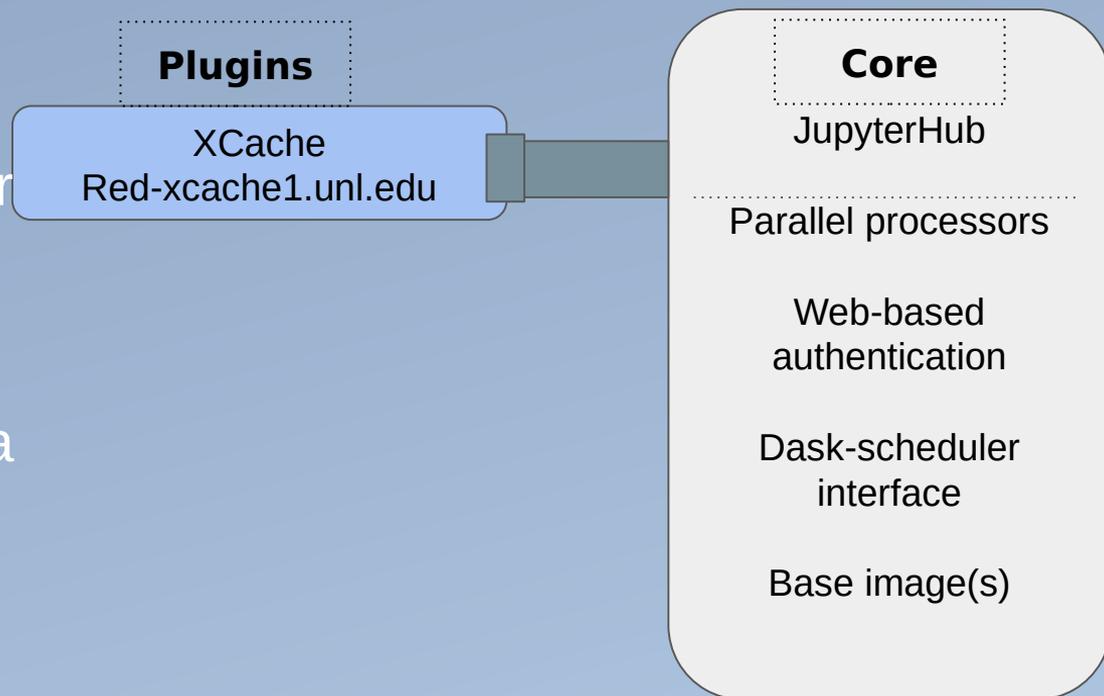
The user only gets a user token to submit jobs to the local queue, whether that queue is a remote resource or a K8s service.

Building blocks: Data Delivery (Stand-alone XCache)

red-xcache1 has 88TB of cache space and allows a cms coffea user access to the CMS data namespace.

Although the casa issues the user a token the mechanism for access is x509 “under the bonnet”.

Full K8s native Xcache 'almost there'.



Building blocks: Data Delivery (Skyhook)



Skyhook bumps into a limitation for scaling out to a cluster as it requires access to the underlying CEPH filesystem. This doesn't easily cross network boundaries. Our skyhook enabled instances are limited to internal Condor worker/resources.

Core

JupyterHub

Parallel processors

Web-based authentication

Dask-scheduler interface

Base image(s)

[\[Coffea-casa team\]](#)

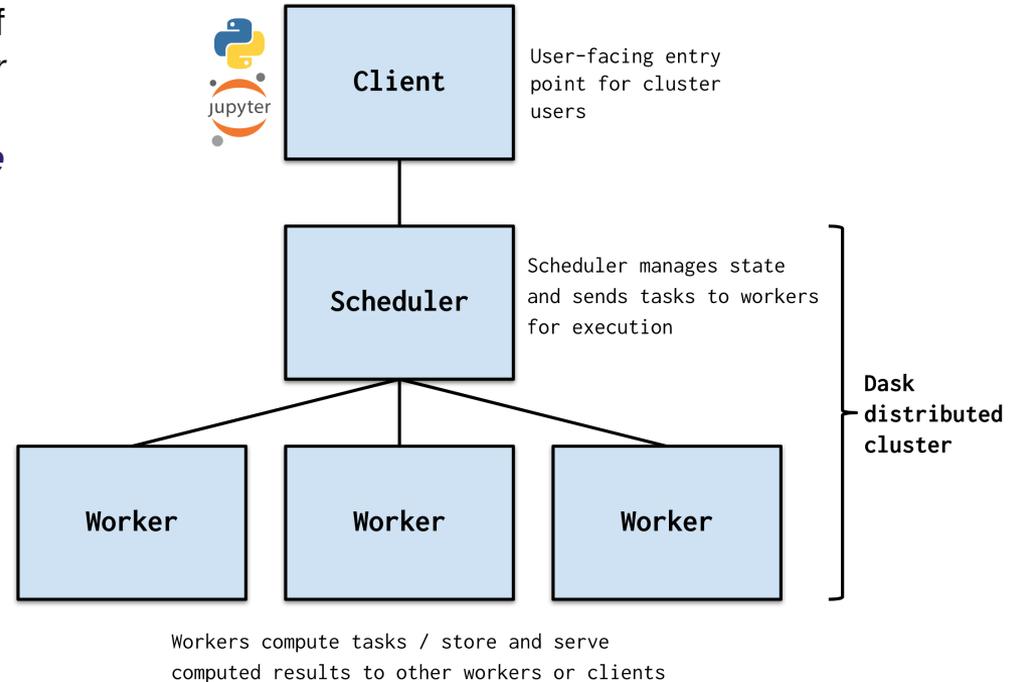
Building blocks: Dask



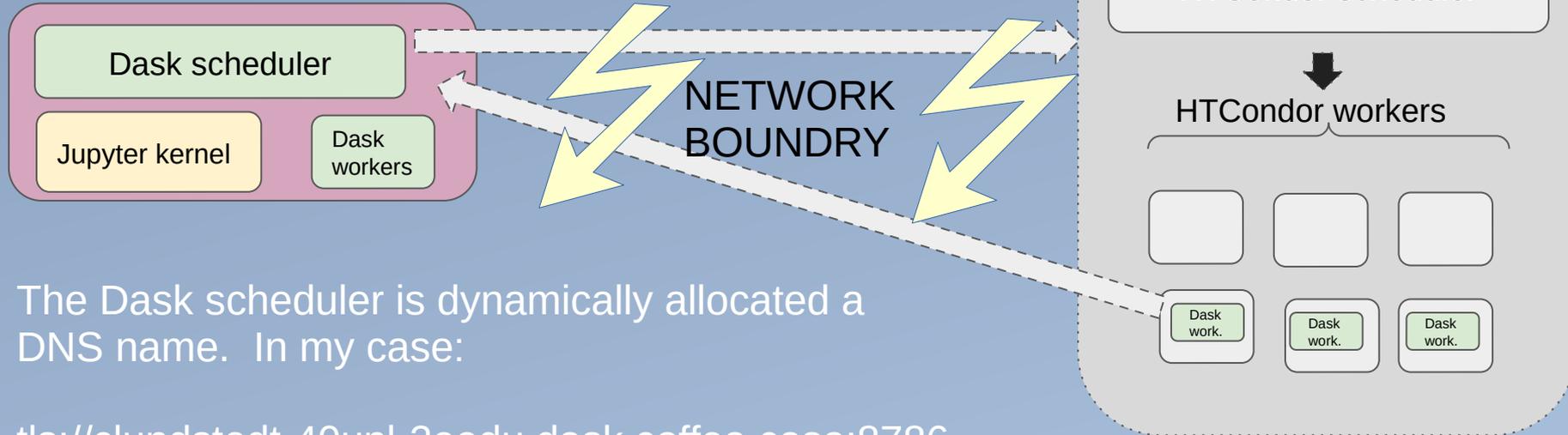
- Dask provides a task-management computational framework in Python based on the manager-worker paradigm
- Dask exposes lower-level APIs letting to build custom systems for in-house applications
- Integrates with HPC clusters, running a variety of schedulers including SLURM, LSF, SGE and *HTCondor* via “*dask-jobqueue*”
- ***This allows us to create a user-level interactive system via queueing up in the batch system***



Dask can be used inside Jupyter or you can simply launch it through Jupyter and connect directly from your laptop



Implementing Dask in Coffea-Casa



The Dask scheduler is dynamically allocated a DNS name. In my case:

[tls://clundstedt-40unl-2eedu.dask.coffea.casa:8786](https://clundstedt-40unl-2eedu.dask.coffea.casa:8786)

This address is passed to the HTCondor/Dask workers to report back to.

Kubernetes manages this traffic for each instance using a traefik service spun up in each instance

At the DNS service (GoDaddy in this case):

Type ?	Nom ?	Données ?	TTL ?	🗑️	✎	
<input type="checkbox"/>	A	@	129.93.183.36	600 secondes	Supprimer	Modifier
<input type="checkbox"/>	A	*.dask	129.93.183.35	1 heure	Supprimer	Modifier
<input type="checkbox"/>	A	*.dask-worker	129.93.183.35	1 heure	Supprimer	Modifier
<input type="checkbox"/>	A	opendataaf-servicex-aod-minio.servicex	129.93.183.35	1 heure	Supprimer	Modifier
<input type="checkbox"/>	A	opendataaf-servicex-minio.servicex	129.93.183.35	1 heure	Supprimer	Modifier
<input type="checkbox"/>	A	opendataaf-servicex.servicex	129.93.183.35	1 heure	Supprimer	Modifier



Two IPs were allocated to this instance, coffea-opendataaf.casa



Implementing Dask in Coffea-Casa

The Coffea-casa class extends the Dask job-queue to automatically configure the job queue environment. Users need know NO Condor or service names.

Service providers/admins can tune the size of workers here.

```
jobqueue:
  coffea-casa:

  # Dask worker options, taken from https://github.com/dask/dask-jobqueue/tree/master/dask_jobqueue
  cores: 2          # Total number of cores per job
  memory: "6 GiB"  # Total amount of memory per job
  processes: null  # Number of Python processes per jobs
  worker-image: "coffeateam/coffea-casa-analysis:latest"

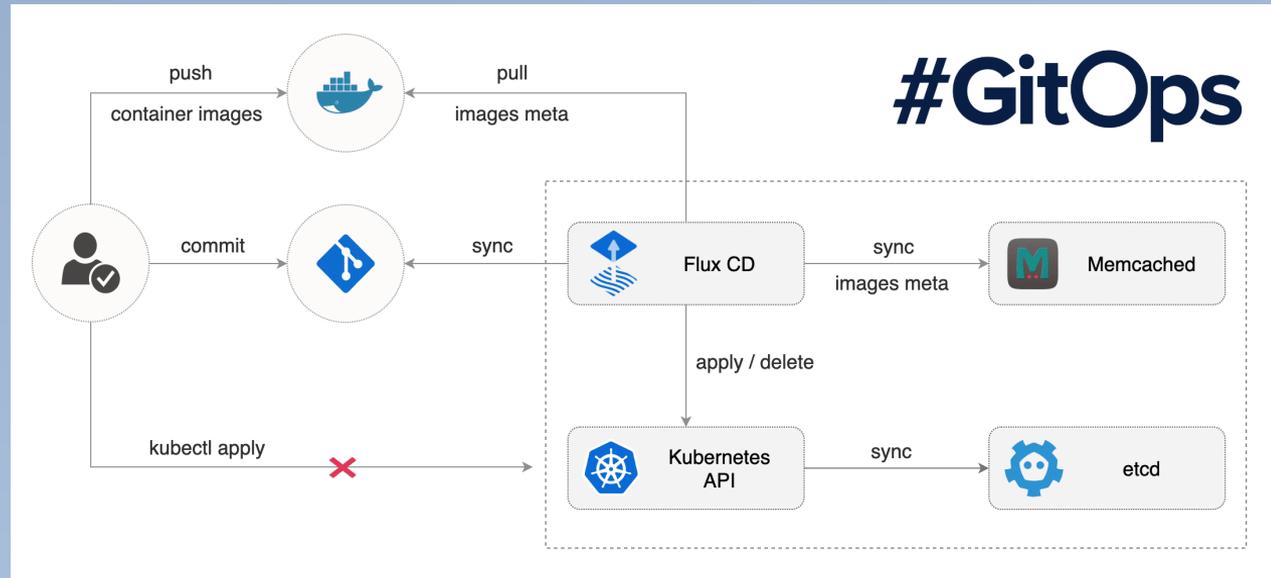
  # Communication settings
  interface: null  # Network interface to use like eth0 or ib0
  death-timeout: 60 # Number of seconds to wait if a worker can not find a scheduler
  local-directory: null # Location of fast local storage like /scratch or $TMPDIR
  extra: []
```

```
## Job extra settings (HTCondor ClassAd)
job_config["job_extra"] = merge_dicts(
    {
        "universe": "docker",
        "docker_image": worker_image or dask.config.get(f"jobqueue.{cls.config_name}.worker-image")
    },
    {
        #"container_service_names": "dask,nanny",
        #"dask_container_port": DEFAULT_CONTAINER_PORT,
        #"nanny_container_port": nanny_port,
        "container_service_names": "dask",
        "dask_container_port": DEFAULT_CONTAINER_PORT,
    },
    {"transfer_input_files": files},
    {"encrypt_input_files": files},
    {"transfer_output_files": ""},
    {"when_to_transfer_output": "ON_EXIT"},
    {"should_transfer_files": "YES"},
    {"Stream_Output": "False"},
    {"Stream_Error": "False"},
    {"+DaskSchedulerAddress": external_ip_string},
    job_kwargs.get(
        fig.get(f"jobqueue.{cls.config_name}.job-extra")
```

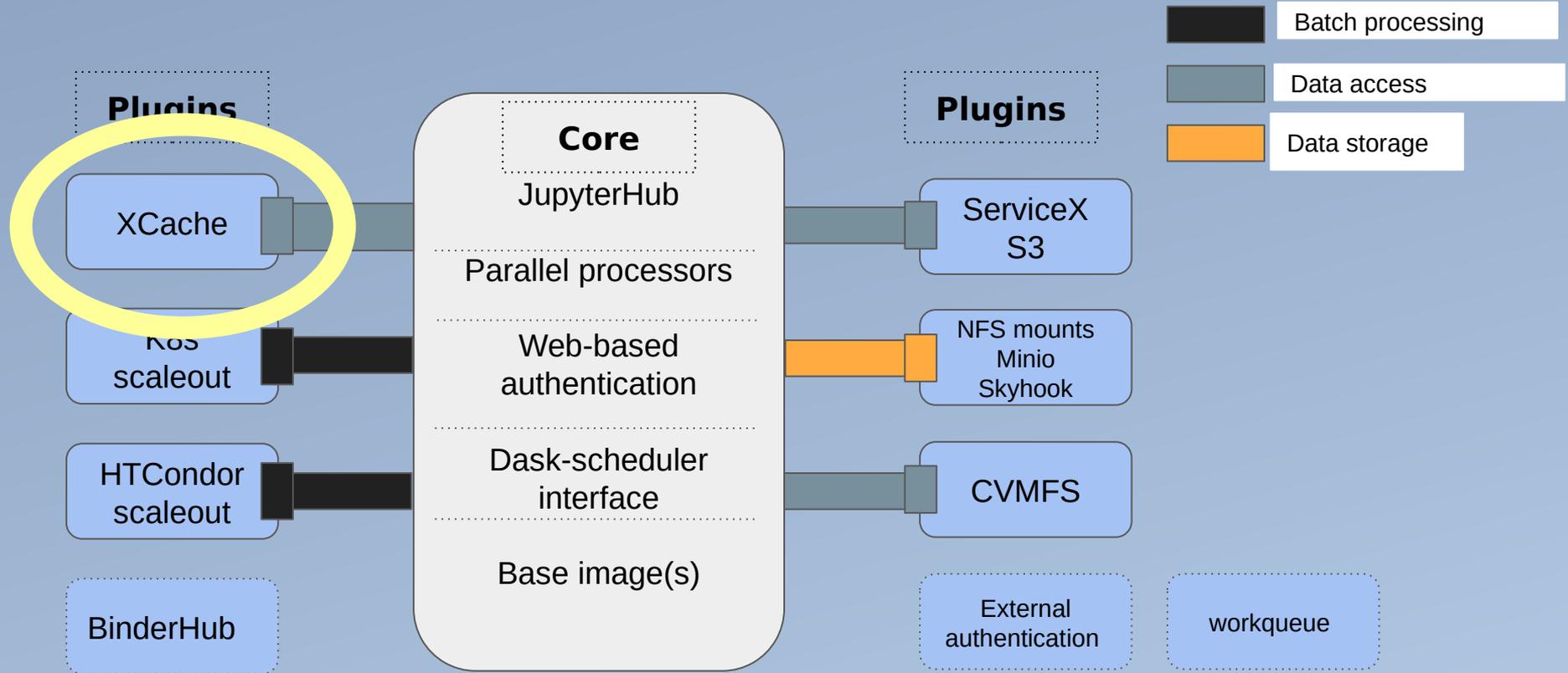
GitOps and Flux

Flux is a service on K8s that checks the state of the cluster vs. what is requested in the charts/yaml files. The state of the cluster is defined in files stored in git, flux reconciles the state with the demand.

All Coffea-casa administration is done via git.

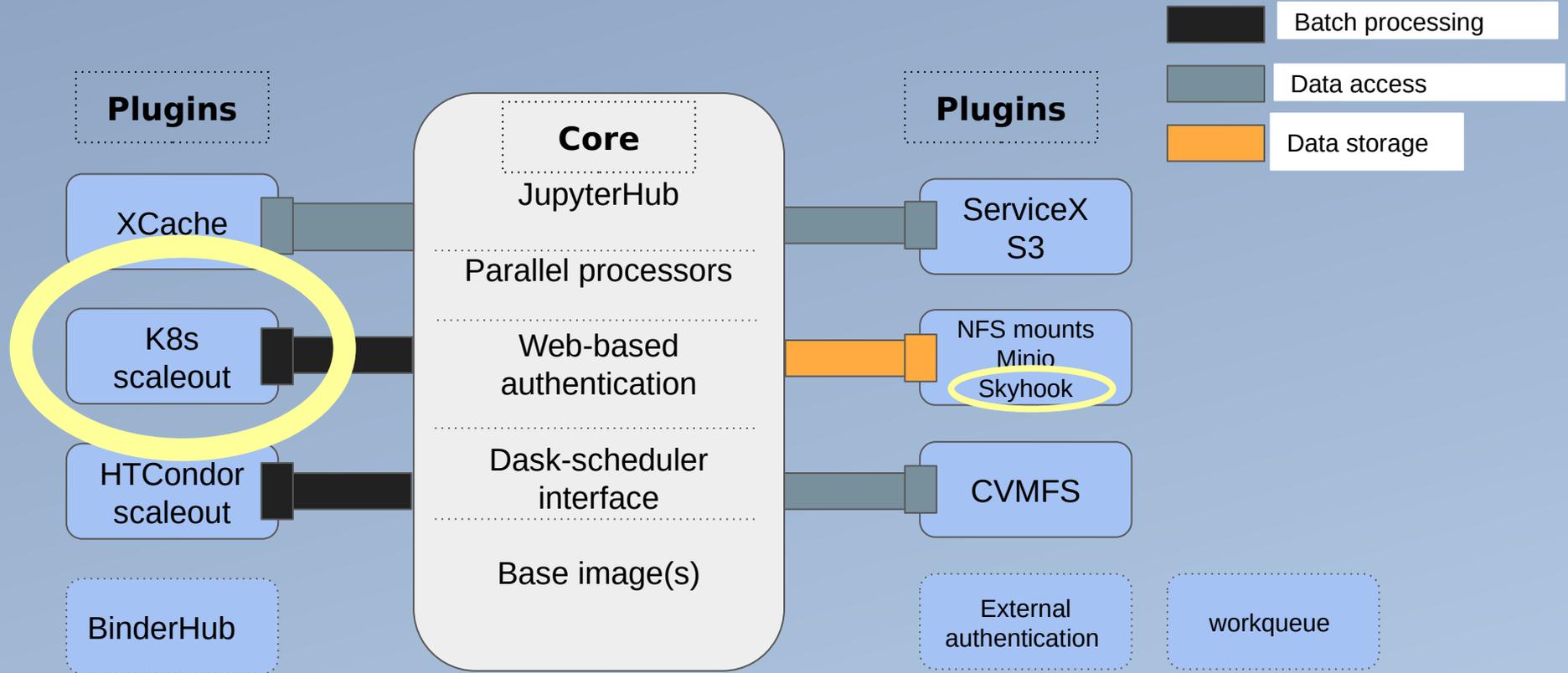


Further extensions/improvements for Casa



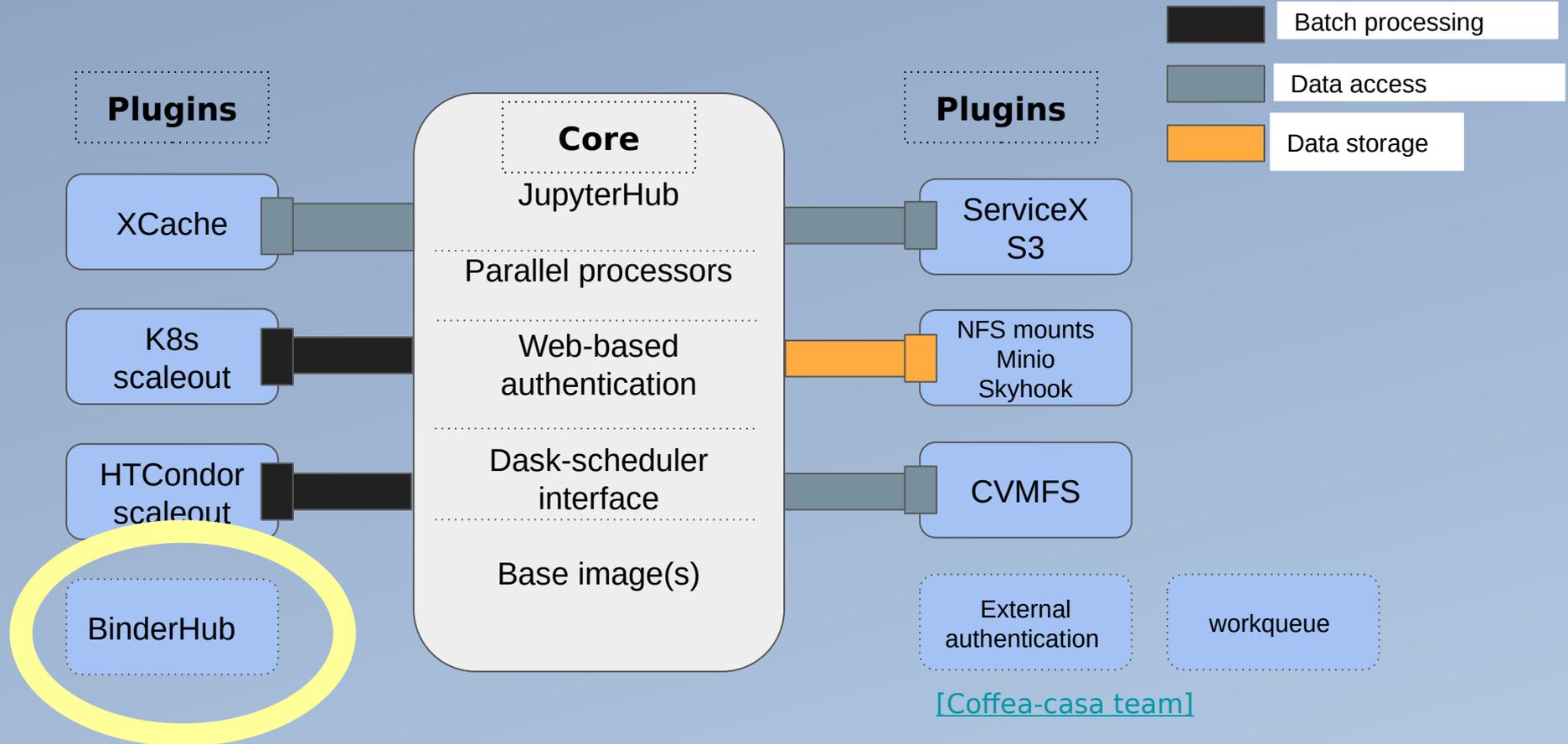
[Coffea-casa team]

Further extensions/improvements for Casa

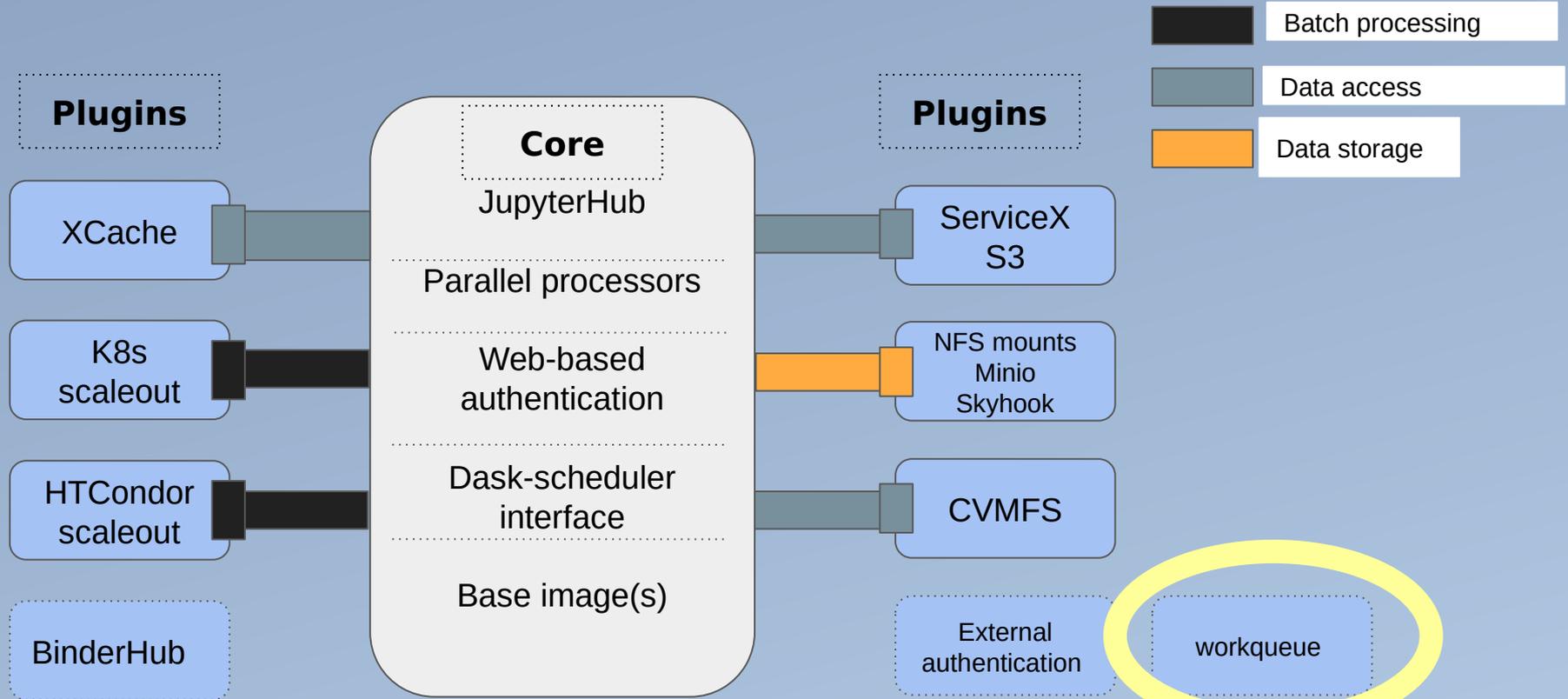


[Coffea-casa team]

Further extensions/improvements for Casa

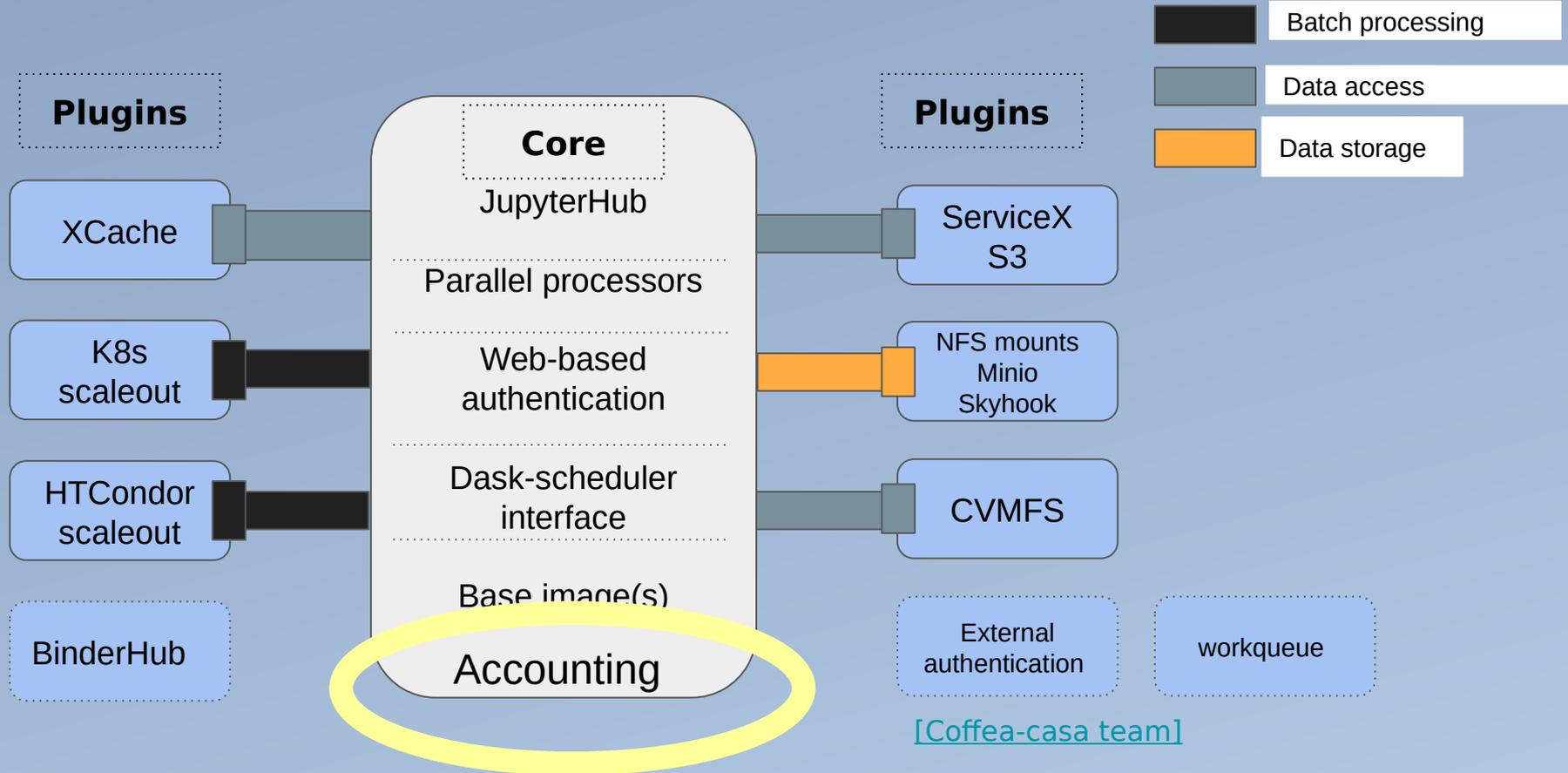


Further extensions/improvements for Casa

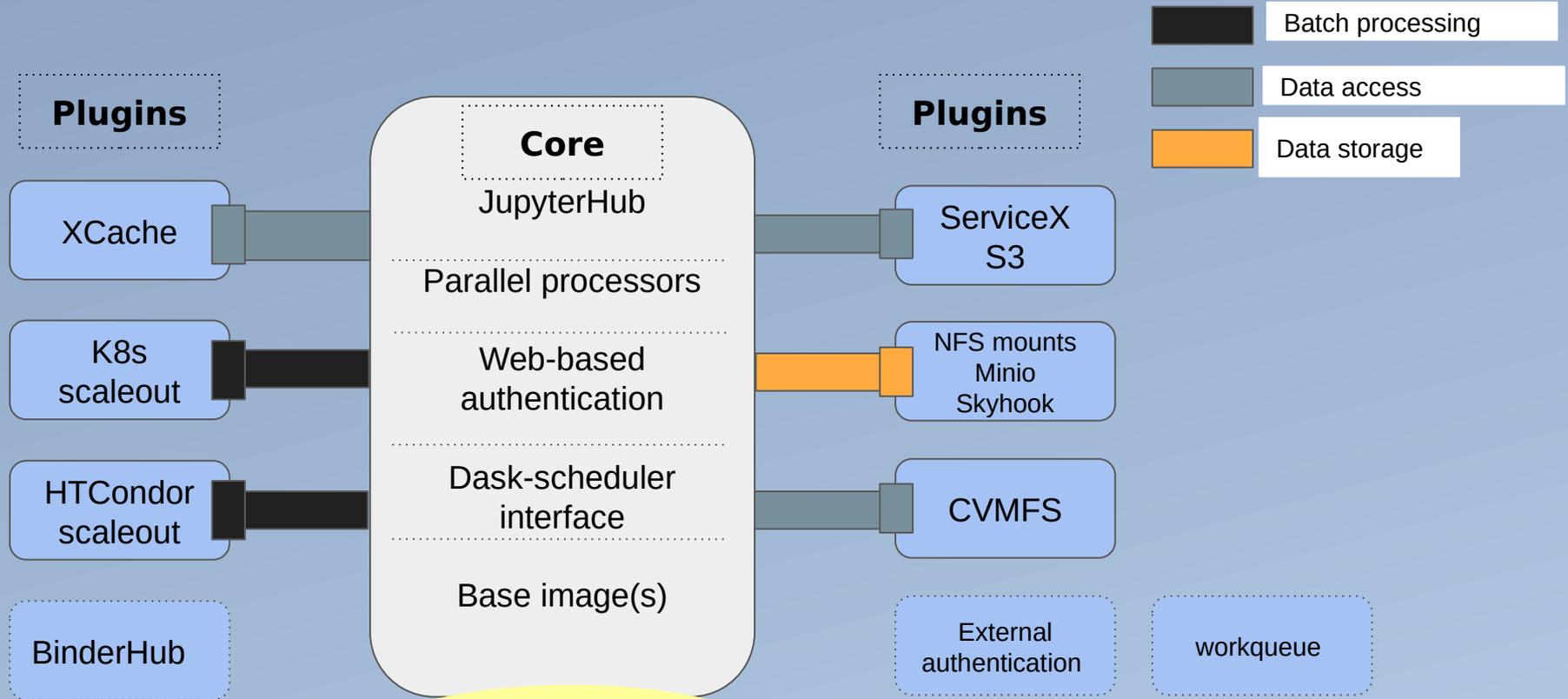


[Coffea-casa team]

Further extensions/improvements for Casa



Further extensions/improvements for Casa



Casa in Minikube

[Coffea-casa team]

Tools we use to admin: Jupyterhub Native



jupyterhub Home Token Admin clundstedt@unl.edu

User ▲	Admin ▼	Last Activity ⇅	Running (0) ⇅
<input type="text" value="Add Users"/>			<input type="button" value="Stop All"/> <input type="button" value="Shutdown Hub"/>
clundstedt@unl.edu	admin	29 minutes ago	<input type="button" value="start server"/> <input type="button" value="edit user"/>
garhan.attedbury@unl.edu	admin	Never	<input type="button" value="start server"/> <input type="button" value="edit user"/> <input type="button" value="delete user"/>
jthiltges@unl.edu	admin	Never	<input type="button" value="start server"/> <input type="button" value="edit user"/> <input type="button" value="delete user"/>
oksana.shadura@cern.ch	admin	8 hours ago	<input type="button" value="start server"/> <input type="button" value="edit user"/> <input type="button" value="delete user"/>

Tools we use to admin: Lens



The screenshot displays the Kubernetes Lens interface. On the left is a navigation sidebar with categories: Cluster, Nodes, Workloads (selected), Configuration, and Network. The main area shows a list of 22 pods in the 'opendataaf-prod' namespace. The pods are controlled by either a 'ReplicaSet' or a 'DaemonSet'. A detailed view for a pod is shown at the bottom, featuring three donut charts for CPU, Memory, and Pods usage.

Name	Namespace	Containers	Restarts	Controlled By	Node
autohttps-579565c5cb-fkpl2	opendataaf-prod	● ● ●	0	ReplicaSet	red-
continuous-image-puller-2dvs9	opendataaf-prod	● ● ● ●	2	DaemonSet	red-
continuous-image-puller-2xtjd	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-4jmdq	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-5jxm9	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-9cn6p	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-bqclq	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-dhkr6	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-f8nss	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-gm5tf	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-gzcts	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-h8s4w	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
continuous-image-puller-t7h2r	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
hub-6864665ff5-zqxcr	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
opendataaf-servicex-code-gen-86bdc5bf-5pvpg	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
opendataaf-servicex-did-finder-cernopendata-cb4d4d949-fs...	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
opendataaf-servicex-minio-779f56cb64-h4sbw	opendataaf-prod	● ● ● ●	0	DaemonSet	red-
opendataaf-servicex-postgresql-0	opendataaf-prod	● ● ● ●	0	DaemonSet	red-

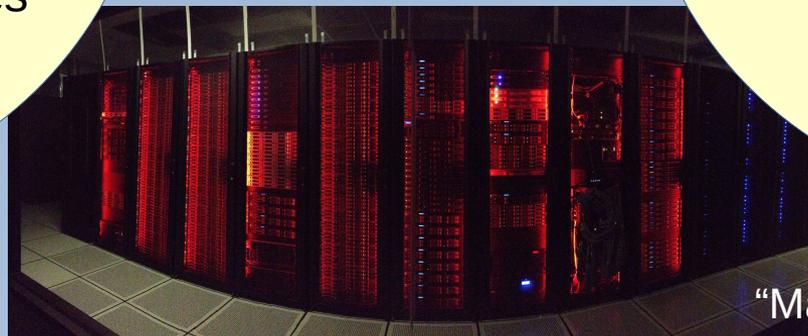
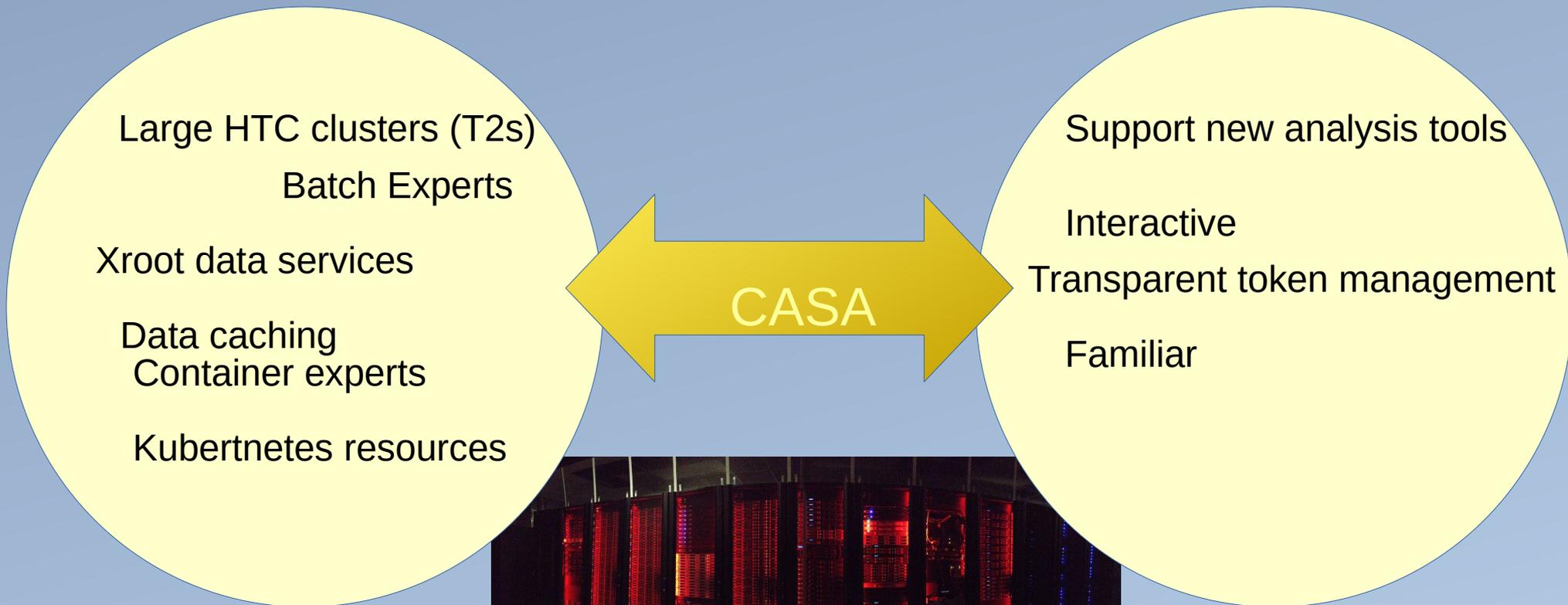
Resource	Usage	Requests	Limits	Allocatable Capacity	Capacity
CPU	0.00	61.00	68.70	344.00	344.00
Memory	443.1Gi	398.0Gi	437.0Gi	1.3Ti	1.3Ti
Pods	500	-	-	-	1760

Scale-out With Coffea: Coffea-Casa Analysis Facility

Questions?



Designing Coffea-Casa Assets vs. Demands



“May you live in an interesting age”