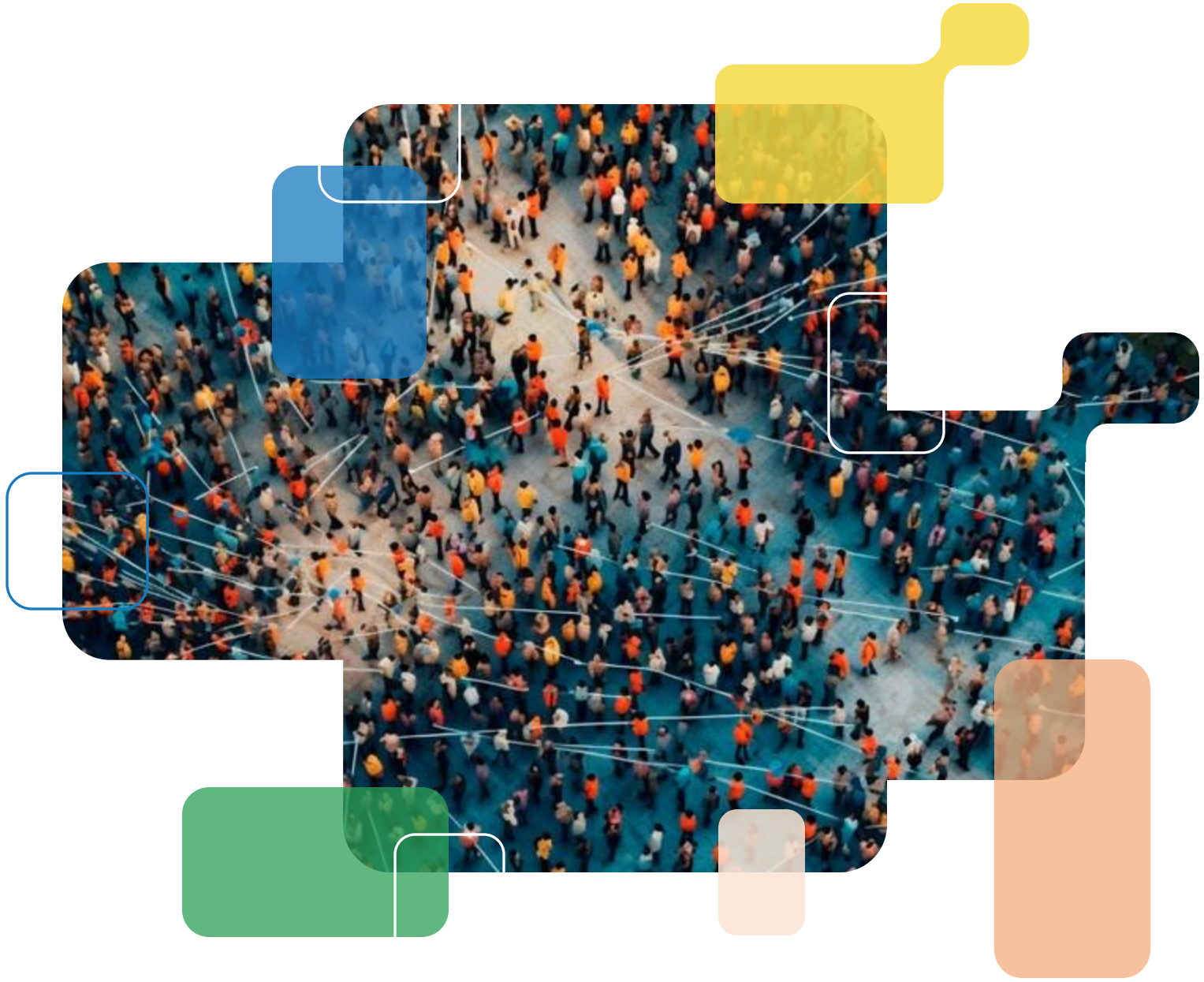
Improving data movement and findability on HPC ecosystems

Mher Kazandjian (Ekogo Computing)

Manuel Torres Rodriguez (SURF)

Marco Verdicchio (ex-SURF)

iRODS UGM 2026

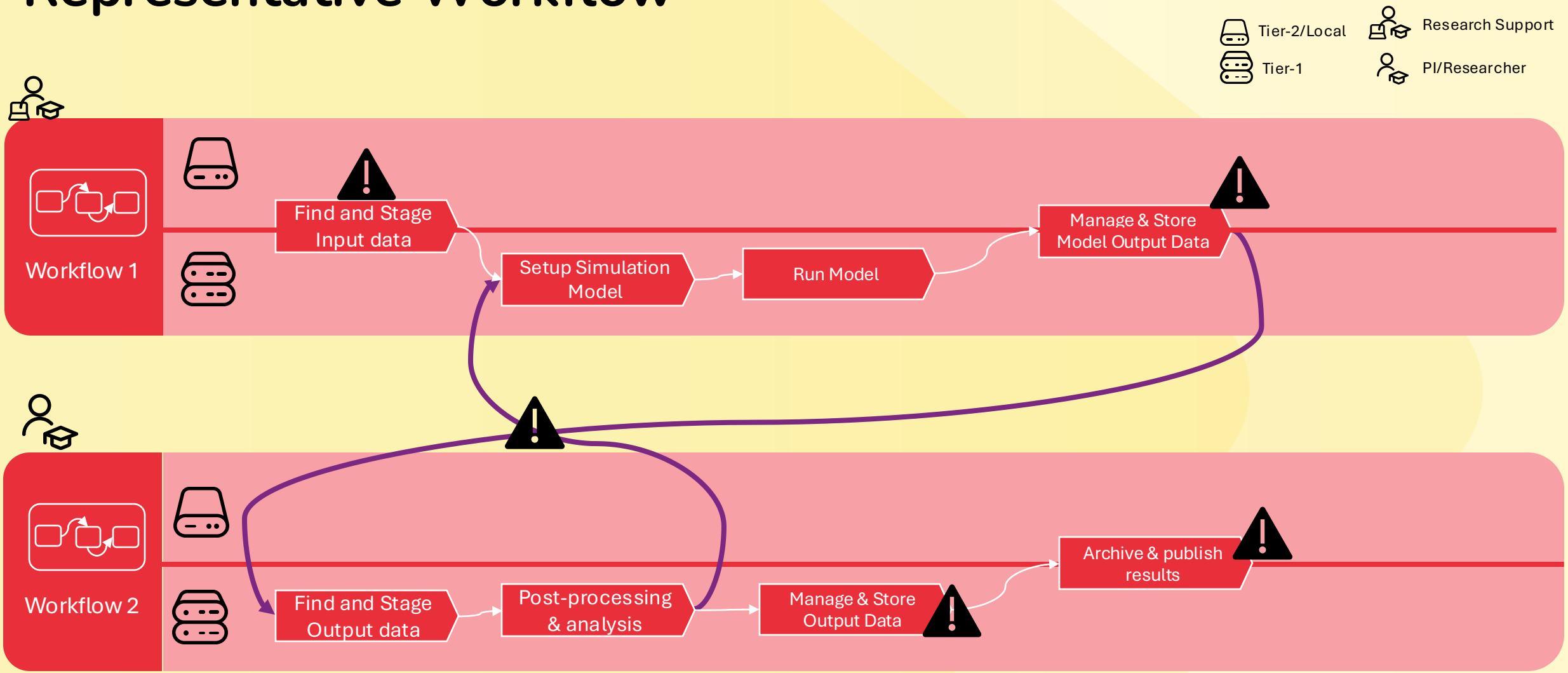


SURF

Ekogo Computing SRL

2026-07-01

Representative Workflow



Current:
Data management consumes ~30% personnel time

Future:
Data management consumes ~0%

Representative Workflow

Tier-2/Local
Research Support
Tier-1
PI/Researcher

Work

Work

For context

1 FTE ~ 200kEU

0.3 FTE ~ 60kEU

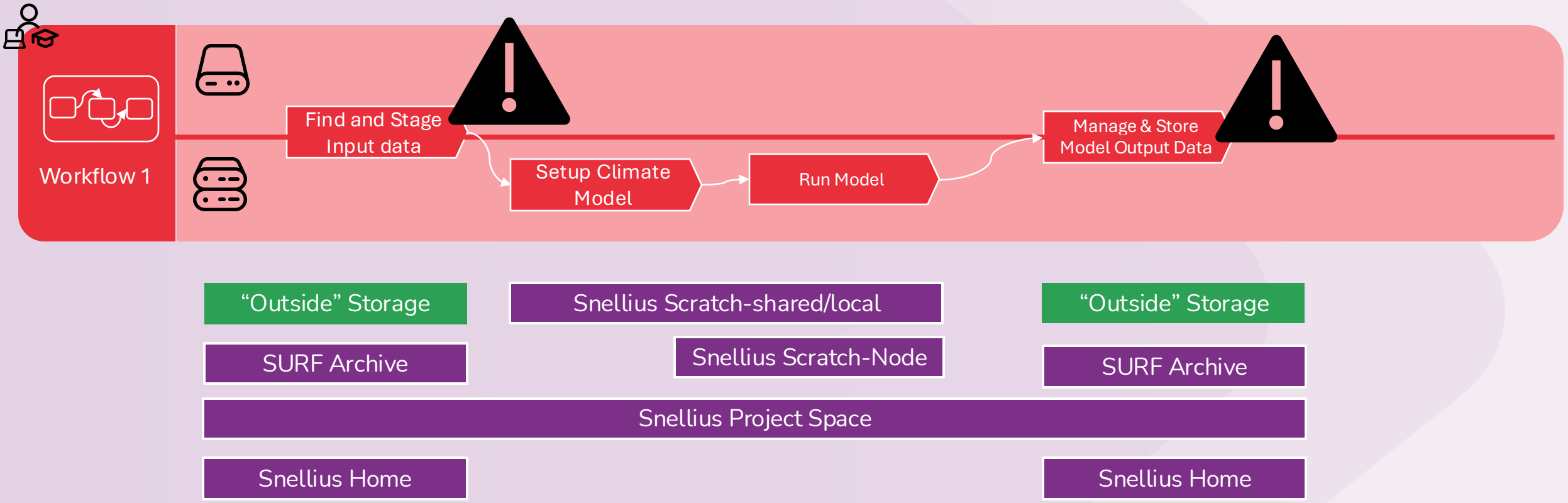
1000 SBU ~ 21 EU

2.8 M-SBU

Current:
Data management consumes
~30% personnel time

Future:
Data management
consumes ~0%

Storage and data movement complexities



Commands:

- Outside Storage to Snelliuis: rsync, scp, rclone
- Inside Snellius: ls/mv/cp/rm
- SURF Archive: dals/daget/darelease

Storage Specificities:

- Scratch-Node: Data stored expires after job completion
- Scratch-local: Data local to node
- Project Space: Data sharable with other users
- SURF Archive: Data needs to be staged

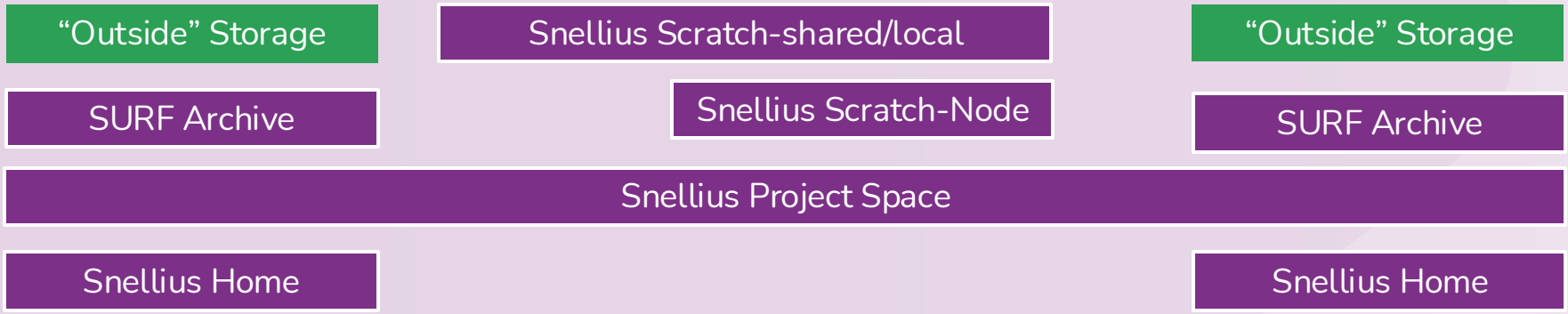
Storage and data movement complexities

```
#!/bin/bash
#SBATCH --job-name=stage
#SBATCH --nodes=1
#SBATCH --cpus-per-task=1
#SBATCH --partition=staging

daget -r /archive/rdmworkflow/dataset

rsync -PrlHvtpog /archive/rdmworkflow/dataset /gpfs/work2/1/rdmworkflow/
```

```
DATASET_DEST="${TMPDIR}/foo"
PROJECT_DEST="/projects/myproj/results/${SLURM_JOB_ID}"
cp -r /projects/myproj/datasets/foo "${DATASET_DEST}"    # foo lands in s
my_sim.py -i "${DATASET_DEST}" -o "${DATASET_DEST}_out" --param 15    # ou
mkdir -p "${PROJECT_DEST}"                                            # fi
cp -vr "${DATASET_DEST}_out" "${PROJECT_DEST}/"
cp job.sh "${PROJECT_DEST}/job.sh"
cp slurm-${SLURM_JOB_ID}{.out,.err} "${PROJECT_DEST}/job.sh/"
```



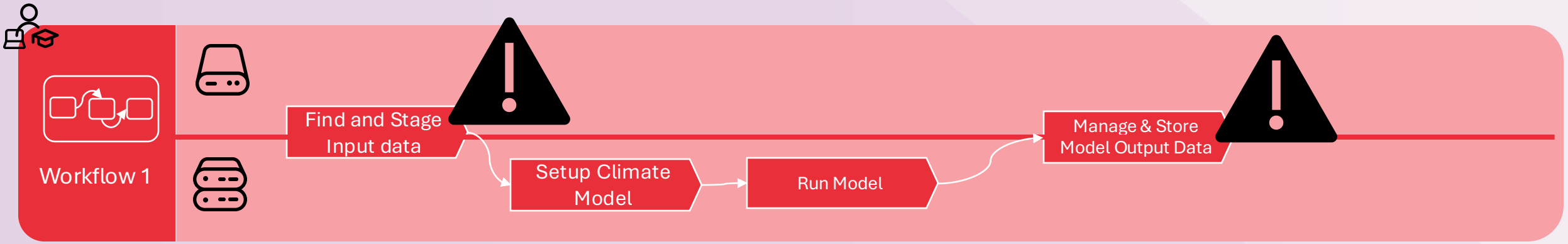
Commands:

- Outside Storage to Snelliuis: rsync, scp, rclone
- Inside Snellius: ls/mv/cp/rm
- SURF Archive: dals/daget/darelease

Storage Specificities:

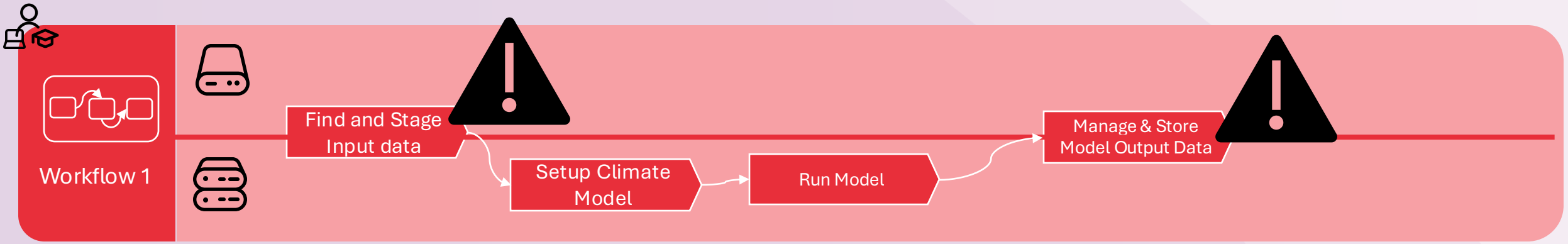
- Scratch-Node: Data stored expires after job completion
- Scratch-local: Data local to node
- Project Space: Data sharable with other users
- SURF Archive: Data needs to be staged

Storage and data movement complexities



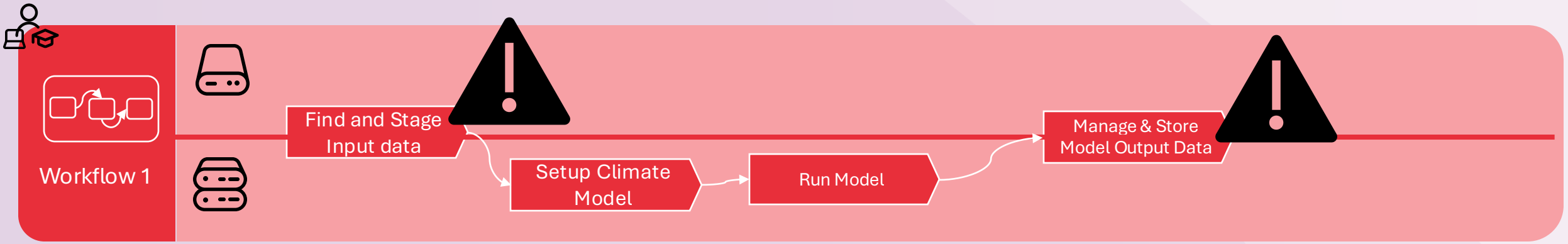
Can we hide storage complexity and let the system manage data movement?

Prof of Concept



Containerized iRODS solution to abstract data movement and storage complexities

Prof of Concept



Containerized iRODS solution to abstract data movement and storage complexities

FLOW is **Fast**

Design for HPC/AI systems from the ground up

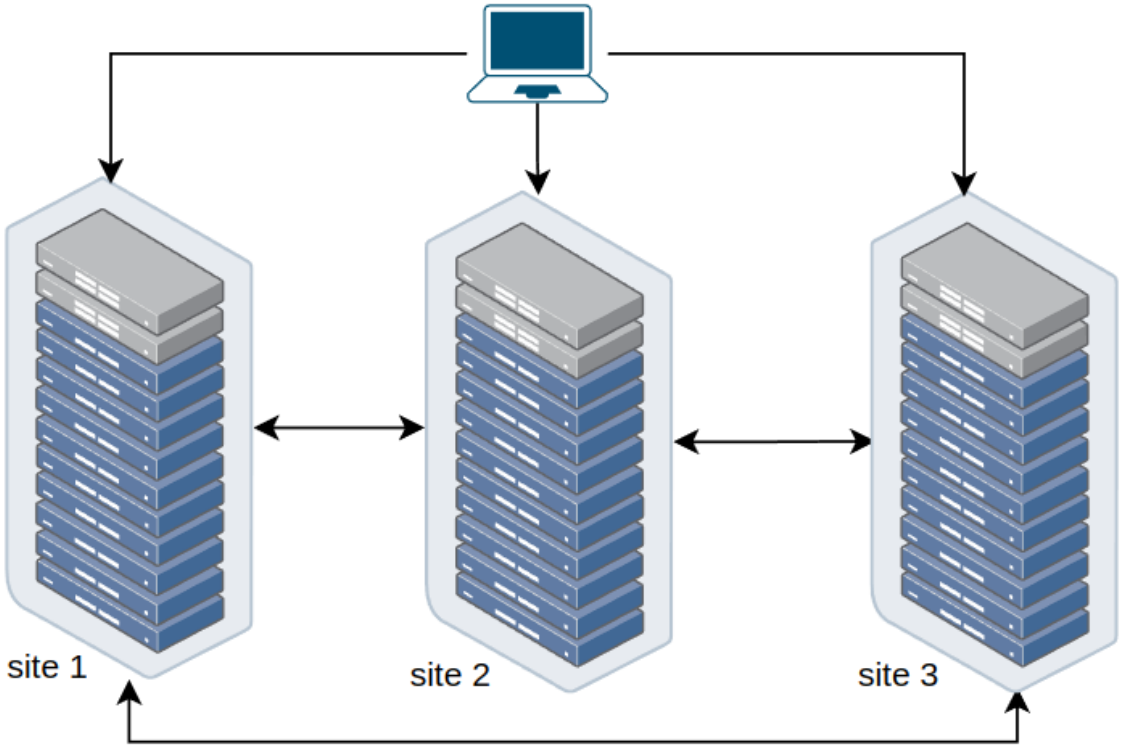
- Data transfers over infiniband by default
- measured 10+GB/s transfers (with room for improvement when if tuned for production)

Building blocks: Abstraction

Hide complexity from the user

Ideal setup: Ambiteous goal

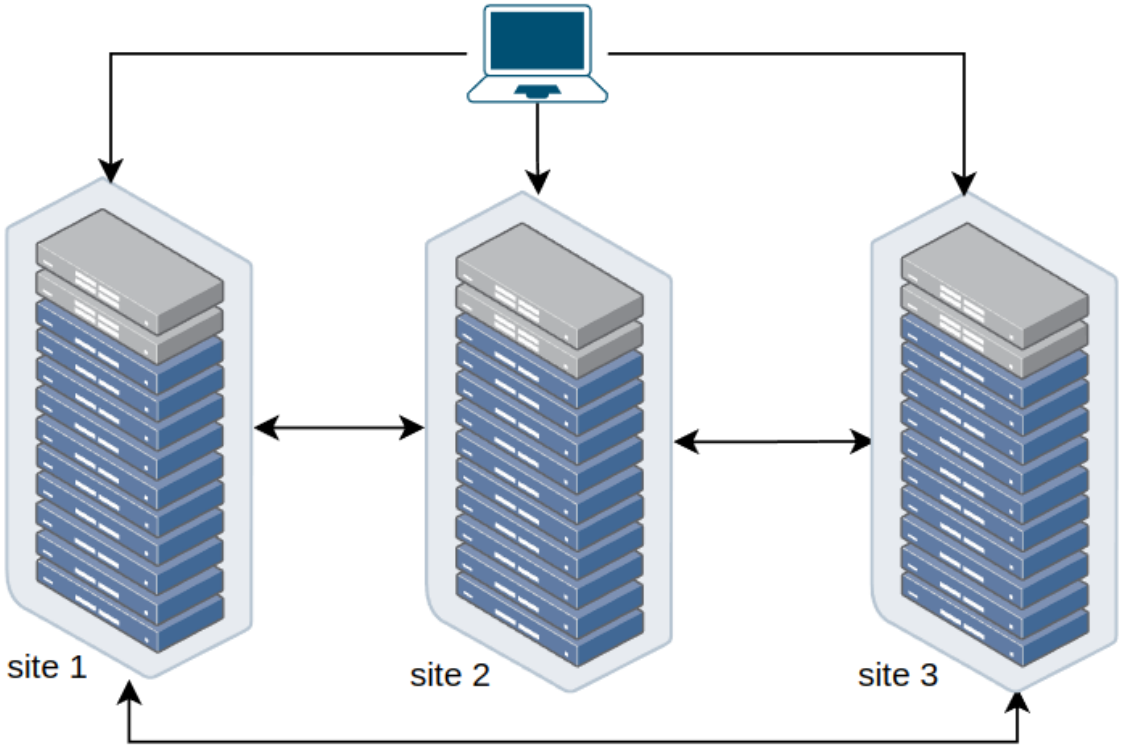
As a user:
- manage my data and run simulations without worrying much about where the data is
- fast data movements and less know how on ops of data movement (efficiently)



Challenges:
- Multi-site deployments => multi-site IT coordination [**nightmare**]
- Every site is different, develop adapters to homogenize or dedicate each site for a particular task [doable – worth the time investment for large projects]

Ideal setup: Ambiteous goal

As a (power) user (local research IT supporting the PI of a project):
- I want to avoid talking to the site admins
- I'd like to deploy the RDM solution myself



Challenges:

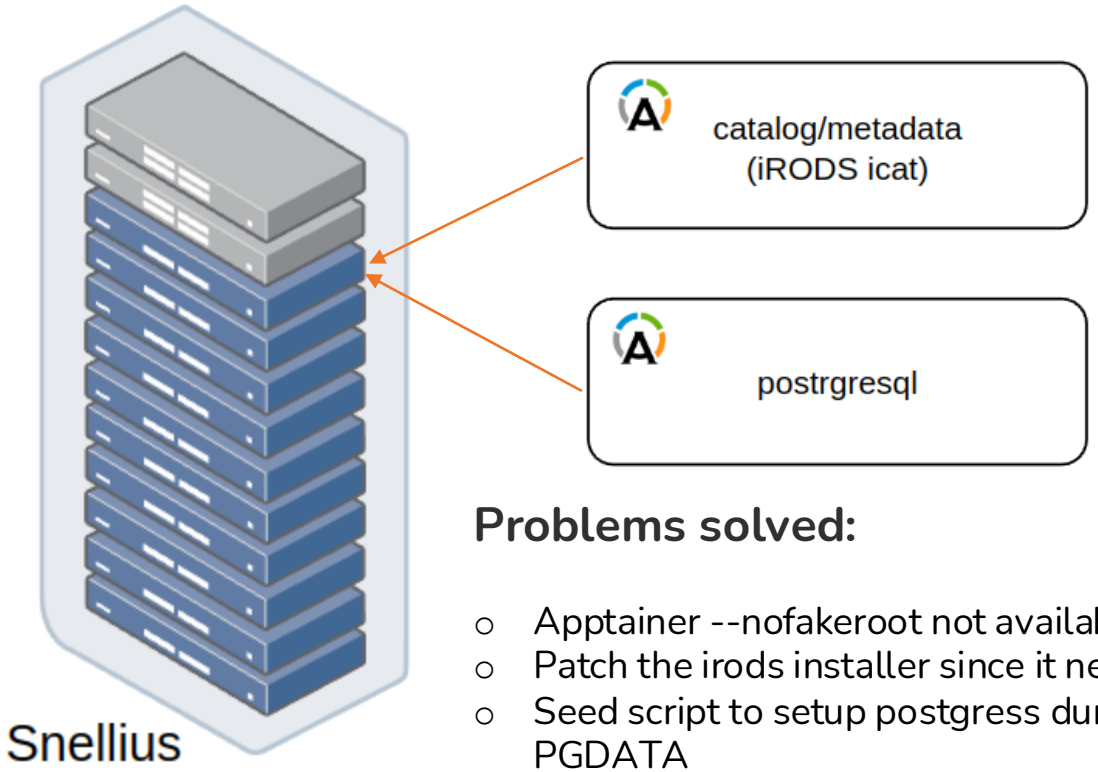
- Can I run iRODs an unpriviledged user in a robust way?

Practical first step:

- Apptainer by design allows running containers as an unpriviledged user
- Try to run iRODS server + postgresql

Single Site – single node

Run iRODS as a non-priv user on a compute node:
- produce a reusable apptainer .sif image
- demonstrate re-usability



Question:

- How easy/hard is it to run iRODS on a compute node?

Practical first step:

- Work in a sandbox and troubleshoot
- Craft the apptainer .def file

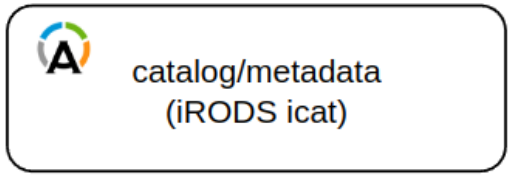
Problems solved:

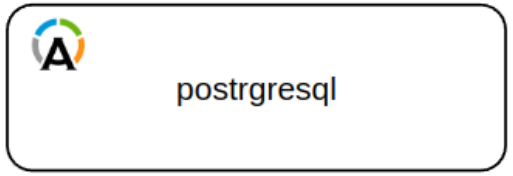
- Apptainer --nofakeroot not available on all systems (need to work around that)
- Patch the irods installer since it needs to os.chown and os.lchown
- Seed script to setup postgress during the first launch, postgress can run as the HPC user as long as it owns PGDATA
- No systemd, no problem – use supervisord (needed to manage the services -> add robustness)

https://servicedesk.surf.nl/wiki/spaces/WIKI/pages/30660184/Snellius

Single Site – single node

Run iRODS as a non-priv user on a compute node:
- smoke test






```
apptainer instance start --writable-tmpfs --no-mount hostfs \
  --no-mount tmp --hostname $(hostname -f) \
  --bind ~/demo/pgdata:/var/lib/pgsql/15/data \
  ~/hpcrdm.sif hpcrdm-demo
```

```
INFO:    Instance stats will not be available - DBUS_SESSION_BUS_ADDRESS is not set
WARNING: Not virtualizing pid namespace by configuration

INFO:    instance started successfully
```

```
└─ ~ $ apptainer exec instance://hpcrdm-demo whoami
mkazandjian
```

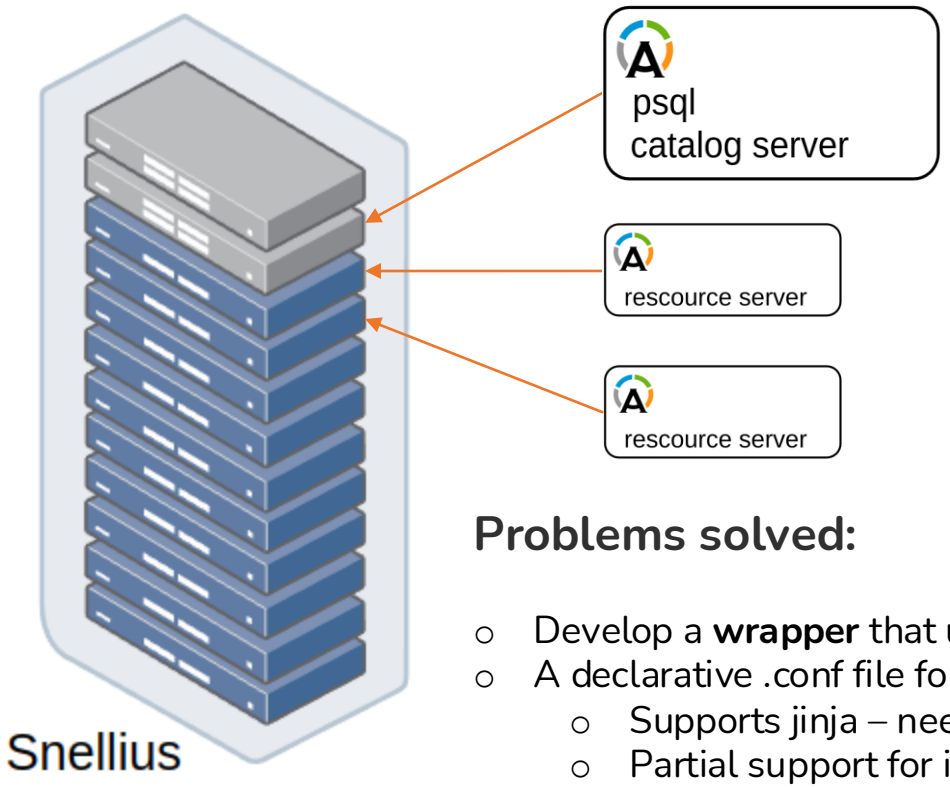
```
└─ ~ $ apptainer exec instance://hpcrdm-demo bash -lc 'IRODS_HOST=localhost ils'
/tempZone/home/rods:
```

Problems solved:

- Apptainer --nofakeroot not available on all systems (need to work around that)
- Patch the irods installer since it needs to os.chown and os.lchown
- Seed script to setup postgress during the first launch, postgress can run as the HPC user as long as it owns PGDATA

Single Site – multi-node

Run iRODS on multiple compute nodes:
- icat + postgres on one compute node (e.g with access to the archive or some special storage)
- one resc server on each compute node



Question:

- Can this be achieved in an easy to use way? (without having to execute many commands?
- Users usually are happy to use s3 commands but complain about icommands

Practical first step:

- Develop a wrapper that minimizes the effort of deploying the setup
- Try to mimic s3 sub-commands / verbs
- Make it easy to configure the deployment – add sshd to supervisord to allow to run ansible* inside the instances

Problems solved:

- Develop a **wrapper** that understands slurm (make use of slurm env vars to discover hosts in a job)
- A declarative .conf file for the deployment
  - Supports jinja – needed for slurm integration and rendering sections for compute nodes
  - Partial support for irods specific configs such as: zone info, admin pass, storage tiering config

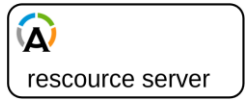
*development was done with ansible, salt works too

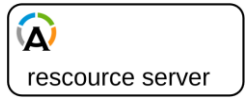
Single Site – multi-node

Minimal requirements:
- 4 line configuration file (yaml – easier to eyeball and read compared to json)
- the .sif image
- the hpcrdm package (pip installable)


```yaml
project: 01-irods-minimal

servers:
  - name: icat-01
    role: provider
```



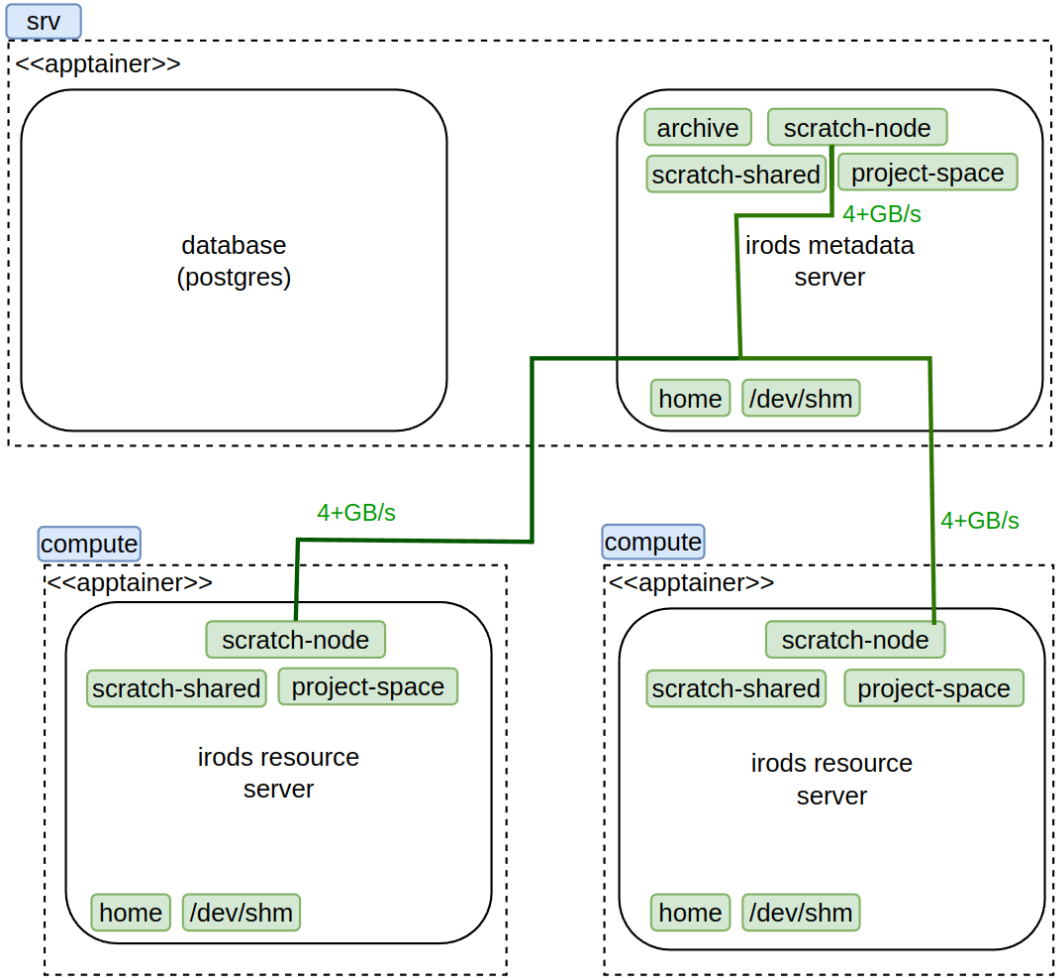



```yaml
project: 03-irods-cluster

servers:
  - name: icat-01
    role: provider
    hostname: node-01.example.com
    resources:
      - name: demoResc
        vault_path: ~/workspaces/hpcrdm/03-irods-cluster/vaults/demoResc
  - name: resc-01
    role: consumer
    hostname: node-02.example.com
    resources:
      - name: resc01Resc
        vault_path: ~/workspaces/hpcrdm/03-irods-cluster/vaults/resc01Resc
      - name: resc01ArchiveResc
        vault_path: ~/workspaces/hpcrdm/03-irods-cluster/vaults/resc01ArchiveResc
  - name: resc-02
    role: consumer
    hostname: node-03.example.com
    resources:
      - name: resc02Resc
        vault_path: ~/workspaces/hpcrdm/03-irods-cluster/vaults/resc02Resc
```

Single Site – multi-node

What is deployed:
- icat + psql (with access to the tape archive)
- two compute nodes




```
└─ ~/.../examples/01-snellius $ hpcrdm flow deploy --n-compute=2
Deploy log: /gpfs/home5/mkazandjian/projects/surf/hpcrdm/hpcrdm/examples
SLURM blocks detected in config -- allocating nodes ...
[2026-06-28 20:54:42] Submitting 3 SLURM job(s) for 3 total node(s) ...
[2026-06-28 20:54:42]   group icat        : 1 node(s) on staging
[2026-06-28 20:54:42]   group compute-01  : 1 node(s) on rome
[2026-06-28 20:54:42]   group compute-02  : 1 node(s) on rome
```

```
JOBID    PARTITION NAME              STATE   NODELIST
-------------------------------------------------------
24275537 rome      hpcrdm-icat       RUNNING tcn505
24275541 rome      hpcrdm-compute-02 RUNNING tcn507
24275540 rome      hpcrdm-compute-01 RUNNING tcn506
```

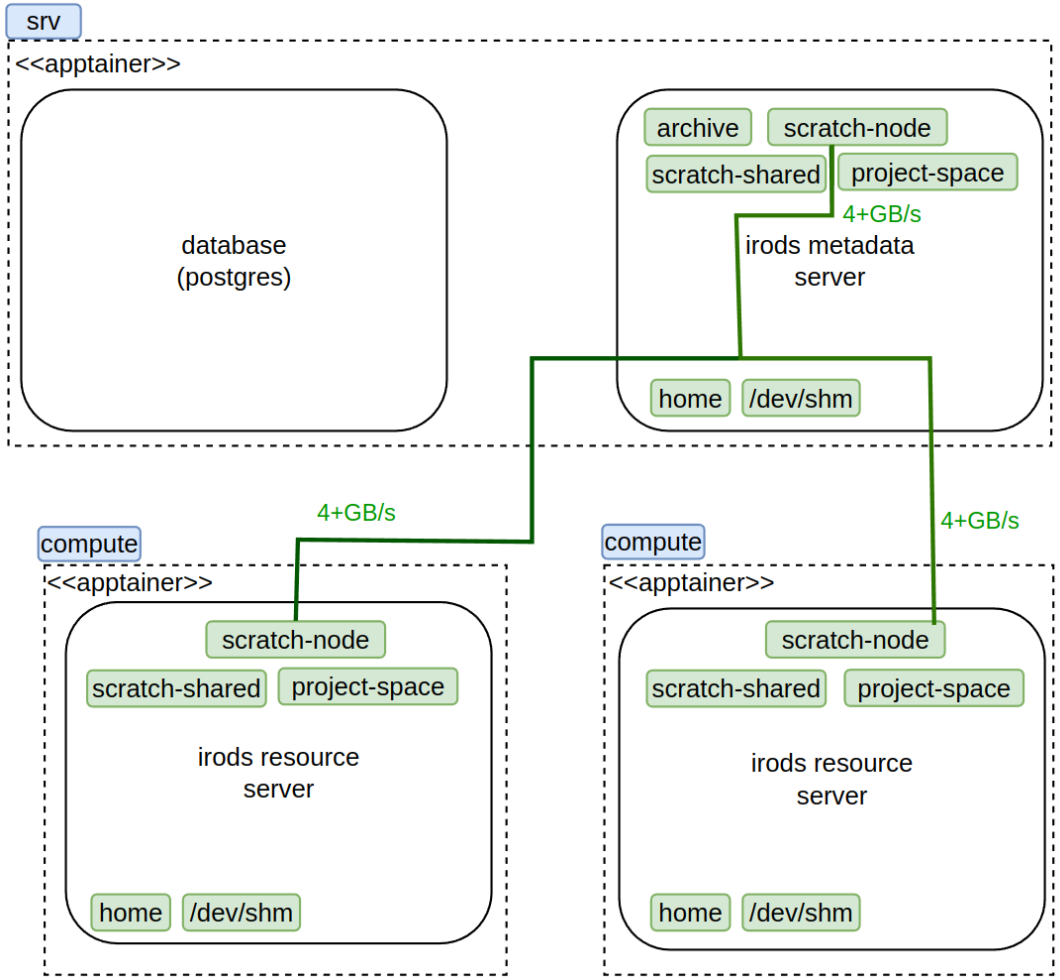
```
All health checks passed.
[2026-06-28 20:57:22] All health checks passed.
  log: /gpfs/home5/mkazandjian/projects/surf/hpcrdm/hpcrdr

Starting RDMA transfer agents ...
  node01 (srv5): agent started on port 7200, C RDMA on por
  node02 (tcn505): agent started on port 7200, C RDMA on p
  node03 (tcn506): agent started on port 7200, C RDMA on p

Cluster is running. Ctrl+C to stop and cancel SLURM jobs.
```

Single Site – multi-node

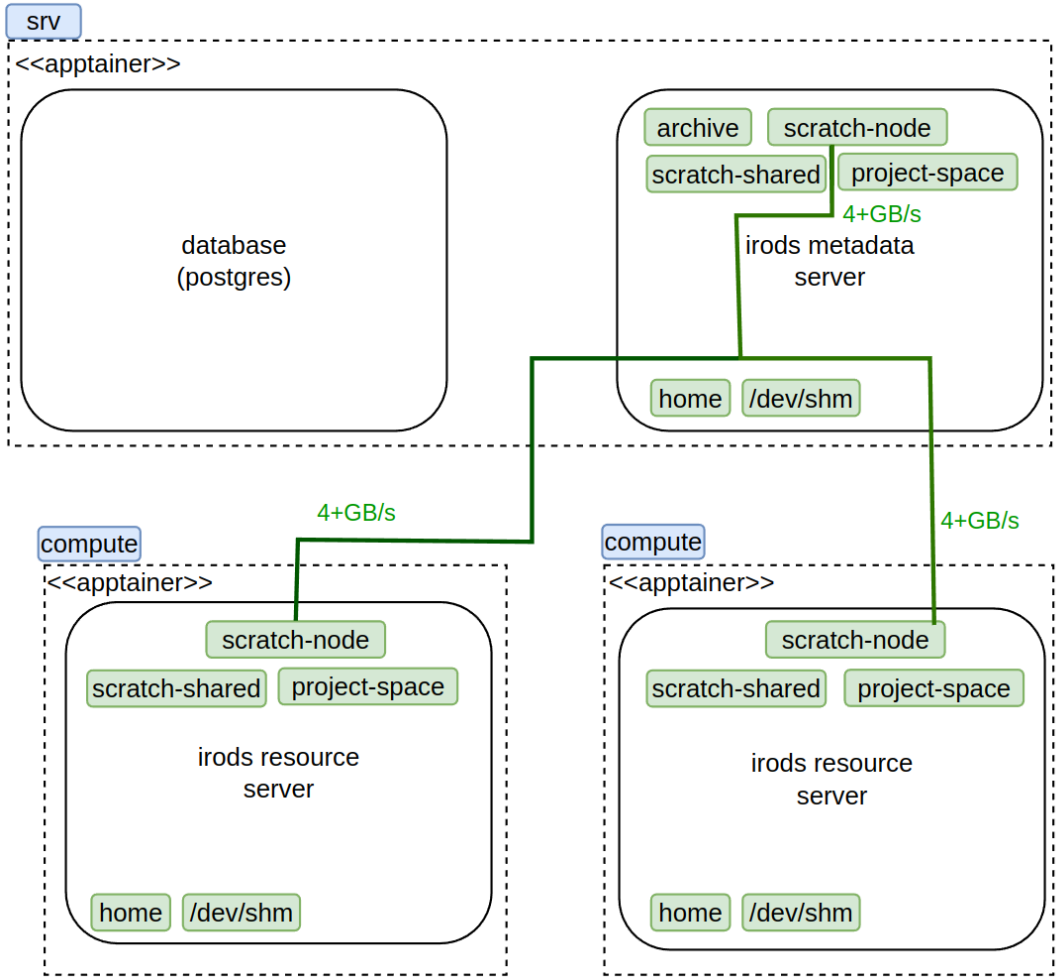
actions:
- list objects




```
$ hpcrdm flow ls
/tempZone/home/rods:
```

Single Site – multi-node

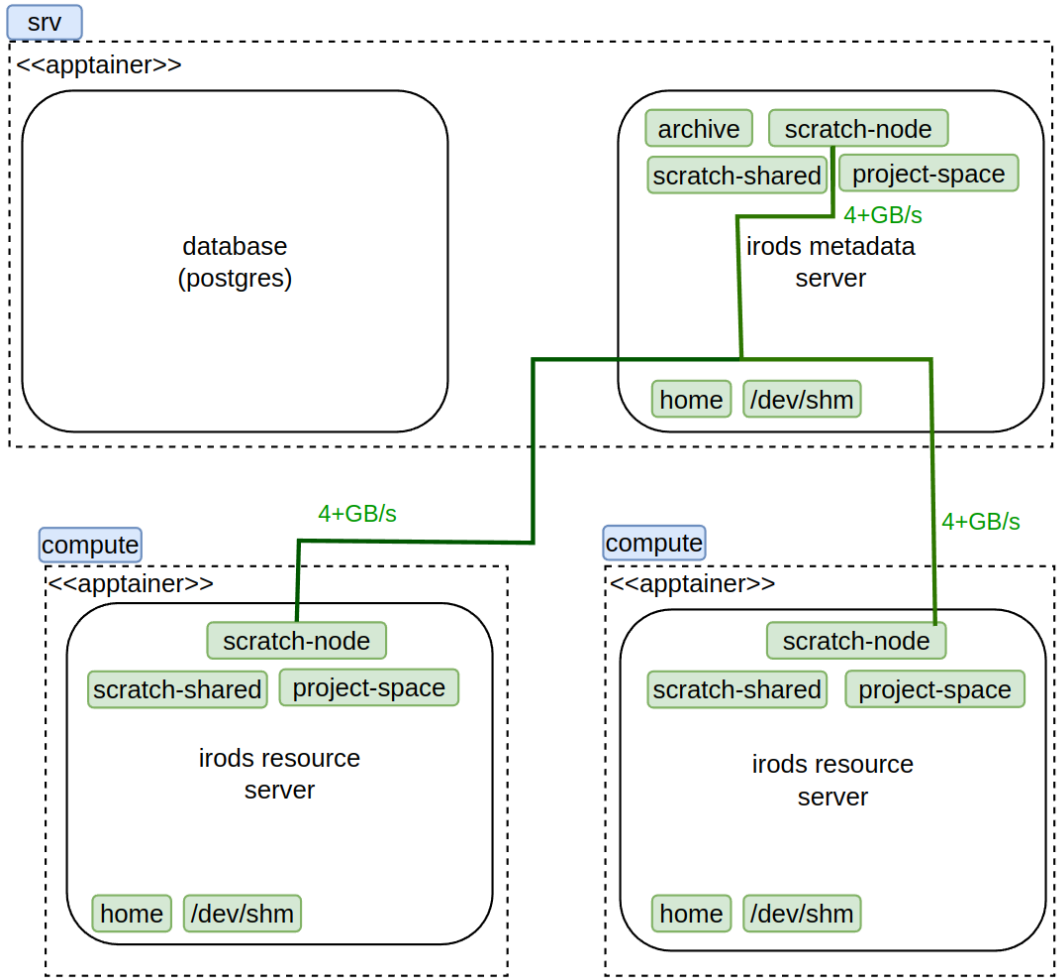
What is deployed:
- list resources




```
$ hpcrdm flow stores
  node01ArchiveResc          srv5    /archive/mkazandjian/irods_vaul
  node01HomeResc             srv5    /home/mkazandjian/irods_vault/n
  node01ProjectSpaceResc     srv5    /gpfs/work2/1/rdmworkflow/irods
  node01RamdiskResc          srv5    /dev/shm/irods_vault/node01
  node01ScratchNodeResc      srv5    /scratch-node/mkazandjian.24275
  node01ScratchSharedResc    srv5    /scratch-shared/mkazandjian/iro
  node02HomeResc             tcn505  /home/mkazandjian/irods_vault/n
  node02ProjectSpaceResc     tcn505  /gpfs/work2/1/rdmworkflow/irods
  node02RamdiskResc          tcn505  /dev/shm/irods_vault/node02
  node02ScratchNodeResc      tcn505  /scratch-node/mkazandjian.24275
  node02ScratchSharedResc    tcn505  /scratch-shared/mkazandjian/iro
  node03HomeResc             tcn506  /home/mkazandjian/irods_vault/n
  node03ProjectSpaceResc     tcn506  /gpfs/work2/1/rdmworkflow/irods
  node03RamdiskResc          tcn506  /dev/shm/irods_vault/node03
  node03ScratchNodeResc      tcn506  /scratch-node/mkazandjian.24275
  node03ScratchSharedResc    tcn506  /scratch-shared/mkazandjian/iro
```

Single Site – multi-node

What is deployed:
- put/get




```
$ hpcrdm flow put 1GB.bin
RDMA transfer 1GB.bin -> srv5:/scratch-node/mkazandjian.24275819/i
Transfer complete: 1048.6 MB in 1.21s (868.5 MB/s)
Registering /tempZone/home/rods/1GB.bin -> /scratch-node/mkazandji
ssh -o ControlMaster=auto -o ControlPath=/tmp/hpcrdm-ssh-cm-mkazan
ne/home/rods/1GB.bin'"'"''
    1GB.bin    sha2:2ocoHJ+ats74+TYpNfT8hk25RgbVIhJhSJTxJTRhp2I=
Done.
```

```
$ hpcrdm flow ls
/tempZone/home/rods:
  1GB.bin
```

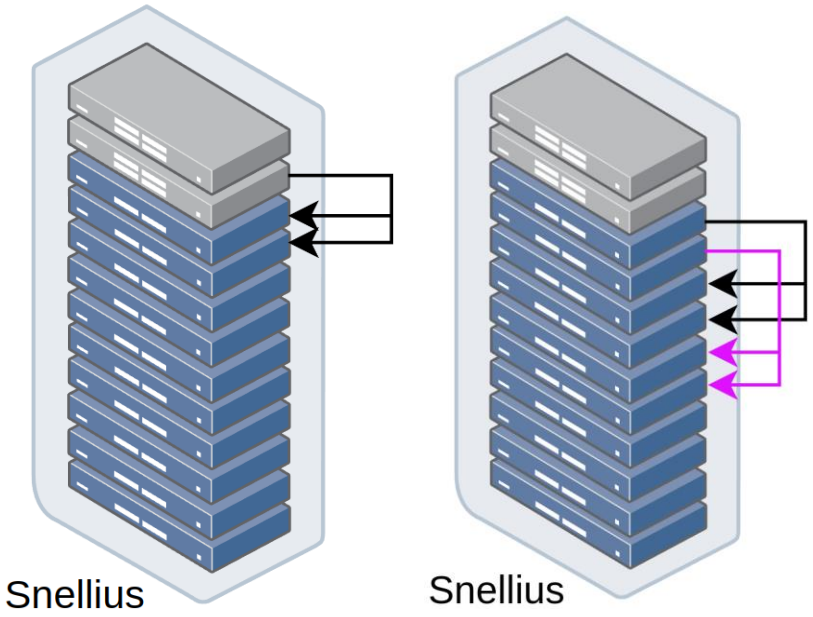
```
$ hpcrdm flow get 1GB.bin 1GB.bin.out
RDMA fetch srv5:/scratch-node/mkazandjian.24275819/irods_vault/1GB
Transfer complete: 1048.6 MB in 1.76s (596.1 MB/s)
```

Single Site – multi-node - summary

What has been achieved:
- solid apptainer container based, slurm aware deployment
- unprivileged user managed (admins are needed only for a systemwide deployment)
- cluster agnostic
- wrapper to facilitate deployment
- persistent deployment (clean stop , start, resume)

HPC workflows: Performance demo

A fast network is what makes an HPC system possible:
- optional support for RDMA transfers as a sidecar
- support MPI like broadcast / scatter / gather paradigms



Question:

- How much bandwidth can one squeeze out?
- What use cases?
  single dataset - same code - parametric sweep
  - different algorithms

How:

```
$ hpcrdm flow repl /tempZone/home/rods/mydata \
    -R '*RamdiskResc' \
    --tree --fanout 2 -P 8
```

Results:

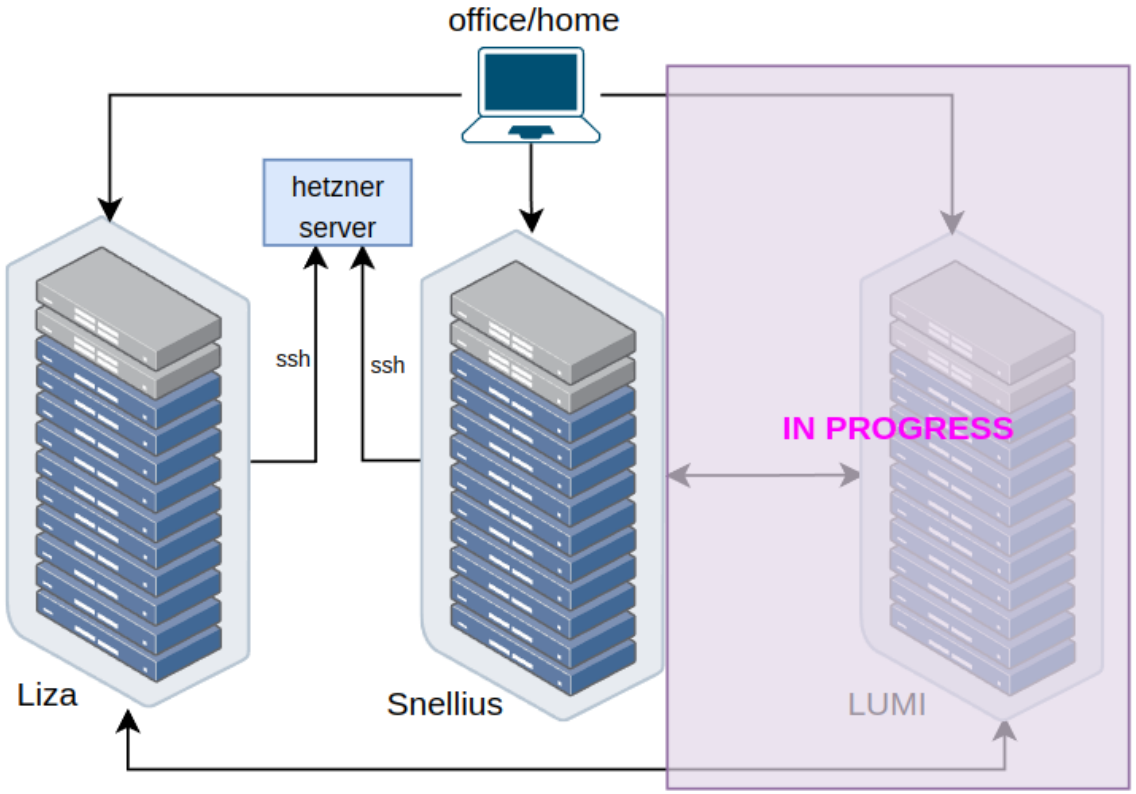
```
Broadcast output:
  round 1:  n0 -> n1   n0 -> n2                                          3.20 s
  round 2:  n0 -> n3   n0 -> n4   n1 -> n5   n1 -> n6   n2 -> n7        3.27 s
  ------------------------------------------------------------------------------
  total wall                                                             6.50 s
  delivered  : 7 × 12 GiB = 84 GiB
  AGGREGATE  : 12.93 GiB/s
  checksums  : ALL OK (7/7 nodes)
```

Room for optimization:

- With a proper admin tuned RDMA configs, mainly allow for large pages (currently 4K) line speed (200Gbit/s) can be acheived

Towards the Ideal setup: Two zone federation (as a user)

As a (power) user (local research IT supporting the PI of a project):
- I already have access to two platforms: experimental + tier 1 supercomputer
- I assume the two zones can not communicate over ssh



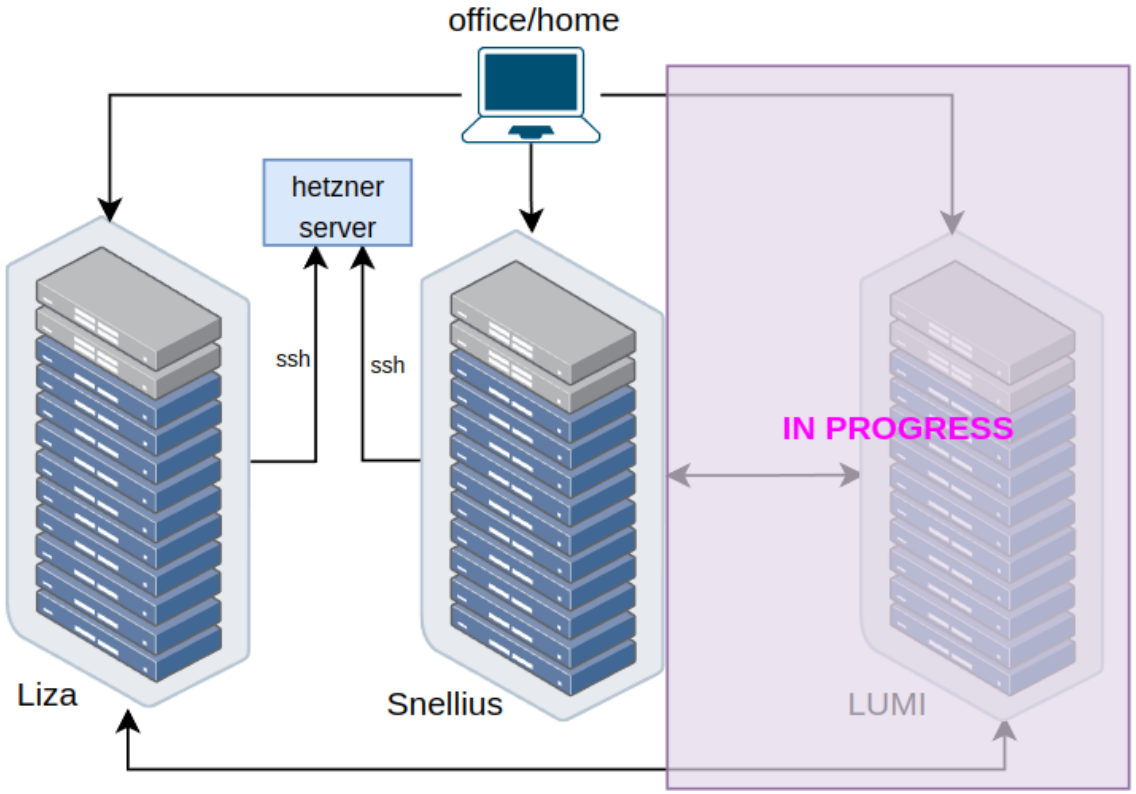
Challenges:
- Juggle ssh tunnel (no big deal)
- Good enough for workflows (as a user)
- [Goal] Achieve listing objects / metadata
- get/put require changing the envs

Practical first step:

- Experiment with the forward and reverse tunnels
- Experiement with the iRODS zone configs

Towards the Ideal setup: Two zone federation (as a user)

As a (power) user (local research IT supporting the PI of a project):
- I already have access to two platforms: experimental + tier 1 supercomputer
- I assume the two zones can not communicate over ssh




```
$ hpcrdm flow ls /snellius/home/exchange
/snellius/home/exchange:
  snellius_seed.txt
```

```
$ hpcrdm flow ls /liza/home/exchange
/liza/home/exchange:
  from_snellius.dat
  liza_seed.txt
```

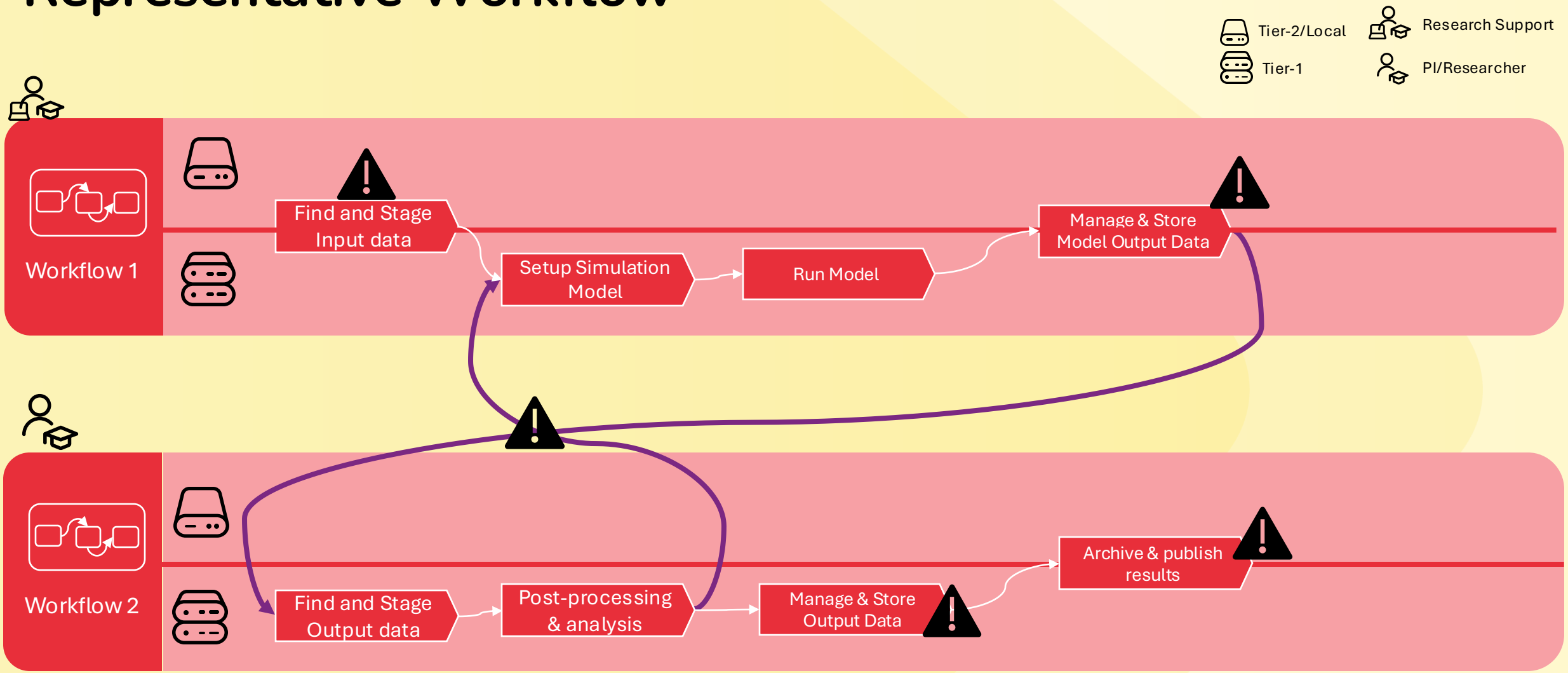
Snellius

```
$ hpcrdm flow lz
  ZONE      TYPE    ENDPOINT
  liza      remote  localhost:16247 (tunnel-client)
* snellius  local   (local)
```

Liza

```
$ hpcrdm flow lz
  ZONE      TYPE    ENDPOINT
  snellius  remote  localhost:26247
* liza      local   (local)
```

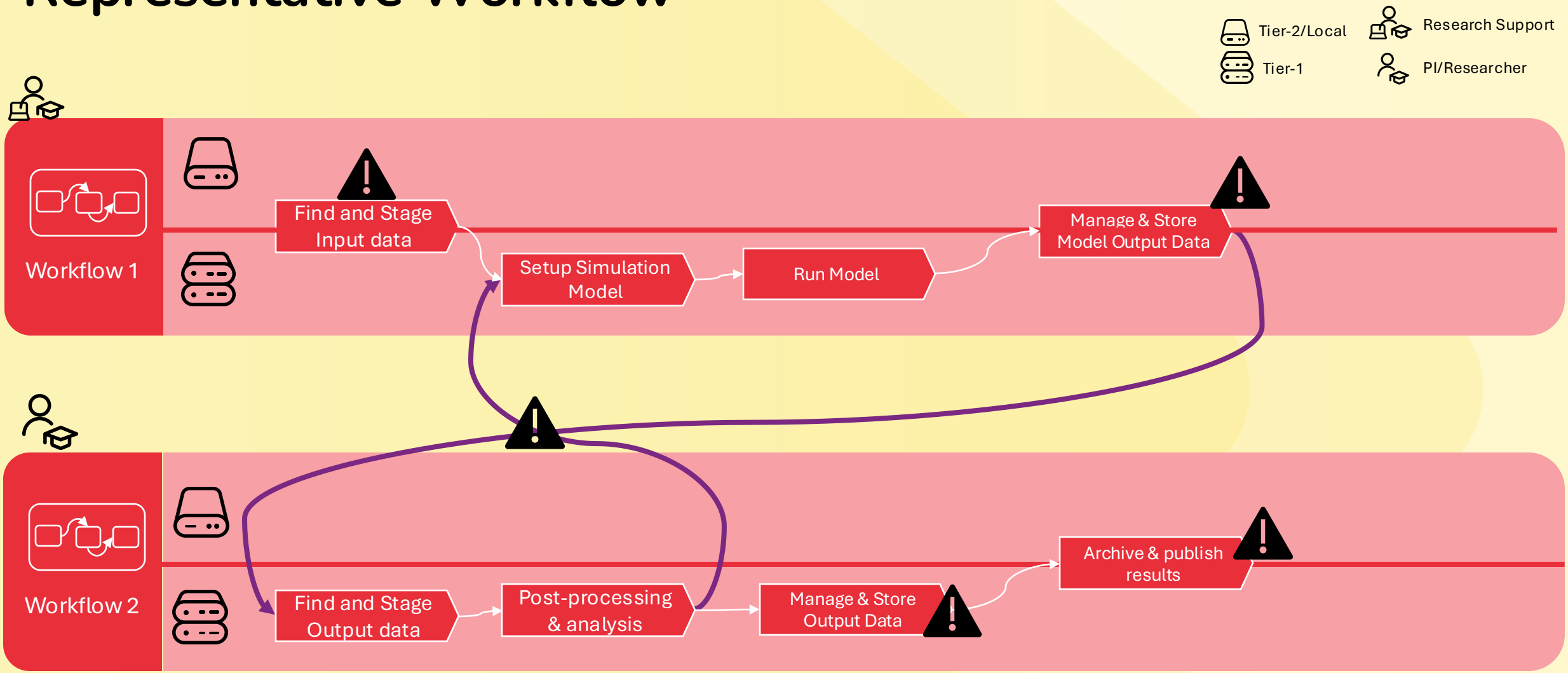
Representative Workflow



Current:
Data management consumes ~30% personnel time

Future:
Data management consumes ~0%

Representative Workflow



Current:
Data management consumes ~30% personnel time

Future:
Data management consumes ~0%

Good Research Data Management begins with the right mindset

Foundation

- How to achieve (or get close to) that 0% effort in data management?
  - This project is an attempt towards identifying workflows and abstracting data movements
  - Awareness and training – encourage users to opt-in
  - A solid MCP server – alpha testing underway
  - A nice / smart / and fast UI – experimenting (TUI + web app)

Future work

- We are looking for projects / volunteers (free support) to test “FLOW”
- We’d like to build a repository of Tier-2/1/0 sites to have as many zones as possible
  - Main challenge: Singularity / Apptainer custom configs (few hours of work per site)
  - SSH restrictions on compute nodes (need to replace ssh’ing by apptainer exec everywhere)
- Under development: scalable and elastic transport mesh to route transfers efficiently

Thank you!

- **Interested in evaluating FLOW?**
  - **Will be released publicly at the end of 2026**

email

github

Moving data across Snellius Storage Spaces

```
#!/bin/bash
#Set job requirements
#SBATCH --job-name=sim
#SBATCH --nodes=2
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=128
#SBATCH --partition=rome
#SBATCH --constraint=scratch-node
#SBATCH --mem=64G
#SBATCH --time=01:00:00

DATASET_DEST="${TMPDIR}/foo"
PROJECT_DEST="/projects/myproj/results/${SLURM_JOB_ID}"
cp -r /projects/myproj/datasets/foo "${DATASET_DEST}"    # foo lands in scratch-node's $TM
my_sim.py -i "${DATASET_DEST}" -o "${DATASET_DEST}_out" --param 15    # output lands in scr
mkdir -p "${PROJECT_DEST}"                                            # final destination f
cp -vr "${DATASET_DEST}_out" "${PROJECT_DEST}/"
cp job.sh "${PROJECT_DEST}/job.sh"
cp slurm-${SLURM_JOB_ID}{.out,.err} "${PROJECT_DEST}/job.sh/"

```

Moving data across Snellius Storage Spaces

Conta

```
#!/bin/bash
#Set job requirements
#SBATCH --job-name=sim
#SBATCH --nodes=2
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=128
#SBATCH --partition=rome
#SBATCH --constraint=scratch-node
#SBATCH --mem=64G
#SBATCH --time=01:00:00

INPUT_COLL="/tempZone/home/rods/06-job-flow/source"
OUTPUT_DEST="/path/to/intermediate/or/output/data/in/scratch-node/${SLURM_JOB_ID}/output"
GATHER_COLL="/logical/path/to/gather/collection/${SLURM_JOB_ID}"
DATASET_DEST=$(hpcrdm flow mirror "${INPUT_COLL}")
python3 my_sim.py -i "${DATASET_DEST}" -o "${OUTPUT_DEST}" --block-size 640 &
hpcrdm flow watch "${OUTPUT_DEST}" --gather "${GATHER_COLL}"
```

Admin Perspective

What does the user need to execute? - - - User space (Zero admin)

How it look like to deploy this system wide?

- include the files below in the container (bind is ok too)
- adjust core.py and server_config.json (standard irods admin work – DMS team quite falimiar with such changes)

Configurations needed and maintenance?

- once a year polish of the following two files (270 and 186 LOC respectively)

`metadata_stage.py` `surfDA.py`