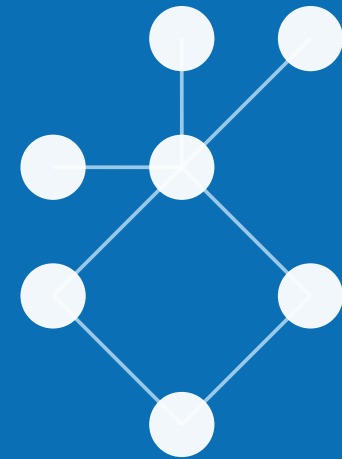


Secure HPC use: what can you do?

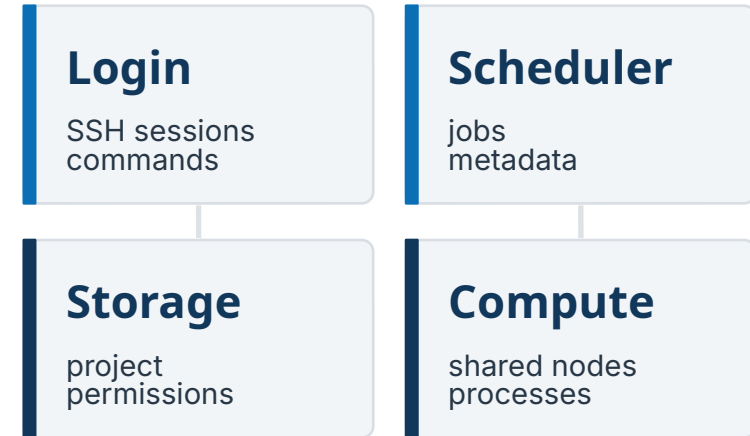
Credentials, SSH, data, software and incident reporting



Shared system

- Shared responsibility: HPC centre + user
- Shared with other users – beyond your own project
- Shared filesystems and login / compute nodes
 - File access is usually controlled by UNIX permissions and project groups
- Process information may be visible to other users
 - Application names and command-line parameters can reveal information
- Scheduler job / metadata may be visible (e.g. squeue)
 - Even job names can leak customer, project or dataset information
- Consult the HPC documentation and system support

ONE SYSTEM, MANY USERS



VISIBLE METADATA

```
$ squeue
JOBID  NAME                USER
81452  sens-analysis      alice
```

Assume names and command lines are not secret unless the system explicitly protects them.

Credentials and secrets

- More than passwords: tokens, SSH keys, certificates, Kerberos, OIDC, ...
- Use MFA where available
- Secrets also include DB passwords, S3 keys and remote-storage credentials
 - Do not embed them in Git, command lines, container images, notebooks or shared files
 - Use an approved secrets-management solution (e.g. Vault / OpenBao) where available
- Do not share accounts or credentials – request another account instead
- If a credential is exposed: revoke / rotate it – do not just delete the file

WRONG

```
$ python run.py --api-token abc123
```

BETTER

```
$ export API_TOKEN=abc123  
# application reads $API_TOKEN
```

BETTER STILL

```
$ install -m 600 /dev/null ~/.secrets/api_token  
# application reads the protected file
```

BEST FOR MANAGED WORKFLOWS

Use an approved secret store: Vault / OpenBao

Exposure compromises the credential. Revoke / rotate it; deleting the copy does not make it safe again.

Secure SSH access

- Check the hostname before connecting – avoid typos / the wrong site
- Verify the host key; do not ignore host-key-change warnings
- Use your own account – one identity per person
- Protect private keys with a passphrase and restrictive file permissions
- Prefer separate keys for different sites / security domains
- Use ssh-agent for convenience; avoid agent forwarding unless needed
- Use MFA or short-lived SSH certificates where the site provides them
- Lock your workstation; close sessions you no longer need

STOP AND VERIFY why the key changed before updating known_hosts.

```
$ ssh login.hpc.example
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION @
@ HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
It is also possible that a host key has just been changed.
ED25519 fingerprint: SHA256:7gX4K0nFv2mL8qQ1uY9pR3cW6aZ5sT0j...
```

Data storage: choose the right place

HOME

PROJECT

SCRATCH

OBJECT

ARCHIVE

- Different storage systems have different purposes
- Classify data before upload: personal data, trade secrets, regulated / export-controlled data
- Storage areas differ in:
 - lifetime / purge policy, quotas, sharing model, availability and performance
- Agree / verify with the provider:
 - where data may be stored, retention, access rights; data-processing agreement where relevant
- Scratch is usually temporary and regularly purged
- Redundancy (RAID / erasure coding) is not backup
- Transfer final results to organisational storage
- Consult documentation and request the appropriate setup

Lifetime

Is the area persistent, purged after N days, or tied to project lifetime?

Protection

Is it backed up? Snapshotted? Replicated only? What restore service exists?

Policy

Is this data class allowed here? What contractual / regulatory rules apply?

Do not let scratch become the only copy of something you cannot reproduce.

Data access and protection

- Review project / group membership; remove people who no longer need access
- UNIX permissions: owner, group, others
- Use extended ACLs where supported / appropriate
- Check permissions after copying or moving data
- Encrypt sensitive data when required
 - in transit and at rest; client-side before upload where appropriate
 - do not store decryption keys together with encrypted data
- Check documentation / ask about:
 - backups / snapshots, retention, administrator access, object-storage exposure / public links
- Know your organisation's rules for confidential / customer data

PERMISSIONS

```
$ umask 077
$ chmod 2770 /project/acme
$ chmod -R o-rwx /project/acme
$ setfacl -m u:bob:rX /project/acme/results
$ getfacl /project/acme/results
```

ENCRYPTION AT REST

```
$ age-keygen -o key.txt
$ tar c dataset | age -r age1qxxxx... >
dataset.tar.age
```

```
# decrypt only where / when needed
```

Encryption helps only if key handling and access control are also sound.

Data lifecycle

- Minimise: upload only what the job actually needs
- Pseudonymise or anonymise before upload where appropriate
- Verify integrity of important transfers
- Intermediate files, checkpoints and logs may also be sensitive
- After the job: clean up temporary copies
 - scratch, \$TMPDIR, node-local disks and application temporary files
- Transfer required final results to organisational storage
- Follow retention / deletion requirements



TRANSFER + INTEGRITY

```
$ rsync -av --partial -e ssh ./dataset/ \
  alice@login.hpc.example:/project/acme/dataset/

$ sha256sum results.tar.zst
```

JOB-PRIVATE TEMP + CLEANUP

```
WORK=$(mktemp -d "${TMPDIR:-/tmp}/job.XXXXXX")
trap 'rm -rf "$WORK"' EXIT
umask 077
```

Pseudonymised personal data is still personal data unless it is truly anonymised.

Software supply chain

- Know where software came from
- Prefer trusted sources:
 - centre-provided modules, EESSI, trusted repositories, trusted / curated container registries
- For external software:
 - use official upstream sources; verify checksums / signatures; pin versions / digests; keep dependencies updated
- For containers:
 - trusted source; lean runtime image; separate build / runtime stages; static analysis; regular rebuilds
- AI agents run with your privileges and can read accessible files
 - They may also send data to a remote LLM / service

TRUST CHAIN

1 Source

Official upstream • centre modules • EESSI • trusted registry

2 Verify

Checksum / signature • pinned digest • vulnerability / secret scan

3 Run

Minimal runtime • least privilege • rebuild when dependencies change

AI agent = code execution + your file access + usually a network connection.

If something looks wrong

- Stop using the affected credential / service when appropriate
- Do not quietly clean up
- Preserve information and logs
- Contact the HPC security or support contact promptly
- Know the confidential reporting channel before you need it
- Early reporting limits the damage
- Your organisation may have its own incident-reporting obligations

Lost laptop

Could contain SSH keys, tokens or cached credentials.

Compromised machine

Private keys or active sessions may need revocation.

Suspected phishing

Report even if you are unsure whether anything was entered.

Misplaced / exposed file

Access may need to be contained and audited quickly.

The security team would rather investigate a false alarm early than discover a real incident late.