Webinar series: Different aspects of EESSI

5 Mondays in a row April-May 2026

https://eessi.io/docs/training-events/2026/webinar-series-2026Q2/

- Introduction to EESSI - **recording available**
- Building software on top of EESSI + contributing to EESSI - **recording available**
- Using EESSI for Continuous Integration -**recording available**
- Introduction to CernVM-FS - **recording available**
- **Using EESSI as a base for a central software stack *(today)***

More info→

Q&A via EESSI Slack

Please join the `#webinar-series-2026q2` channel in the EESSI Slack for questions and discussion

Step 1) **Join the EESSI Slack,** see “Slack channel” link at https://eessi.io

Step 2) **Join `#webinar-series-2026q2` channel** in EESSI Slack

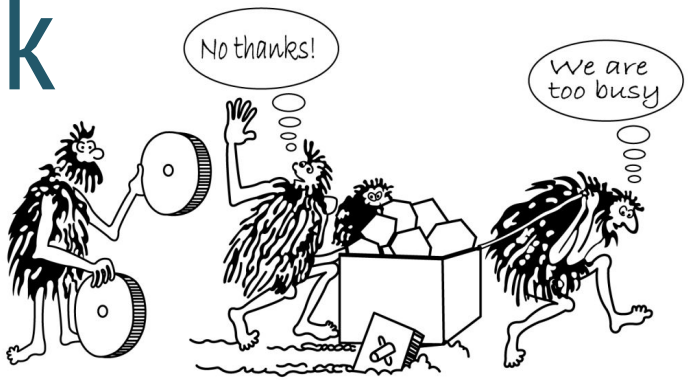
(https://join.slack.com/t/eessi-hpc/shared_invite/zt-2wg10p26d-m_CnRB89xQq3zk9qxf1k3g)

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

MultiScale

Using EESSI as a base for a central software stack

Mon 01 Jun 2026



Speakers: Bob Dröge, Caspar van Leeuwen

Slides: Bob Dröge, Caspar van Leeuwen, Pedro Santos Neves

University of Groningen (NL), SURF (NL)

Helpful knowledge

- You have watched the previous episodes in the webinar series
    - We will use **EESSI**, **EESSI-extend**, **EasyBuild**, **CernVM-FS**
- You know how to make **EESSI** available on your system
- You know how to build a software stack with **EasyBuild**
- Nice to have:
    - You are familiar with containers and **Lmod**

- Need a refresher? Check out past session materials 👇
    - Introduction to EESSI
    - EESSI, EasyBuild & EESSI-extend
    - CernVM-FS

Agenda

- Introduction
- Approach 1: Site builds on top of EESSI in a shared FS
- Approach 2: Leveraging all of EESSI for site builds
- Special cases: rebuilds, licensed software
- Site configuration
- Discussion / Future work

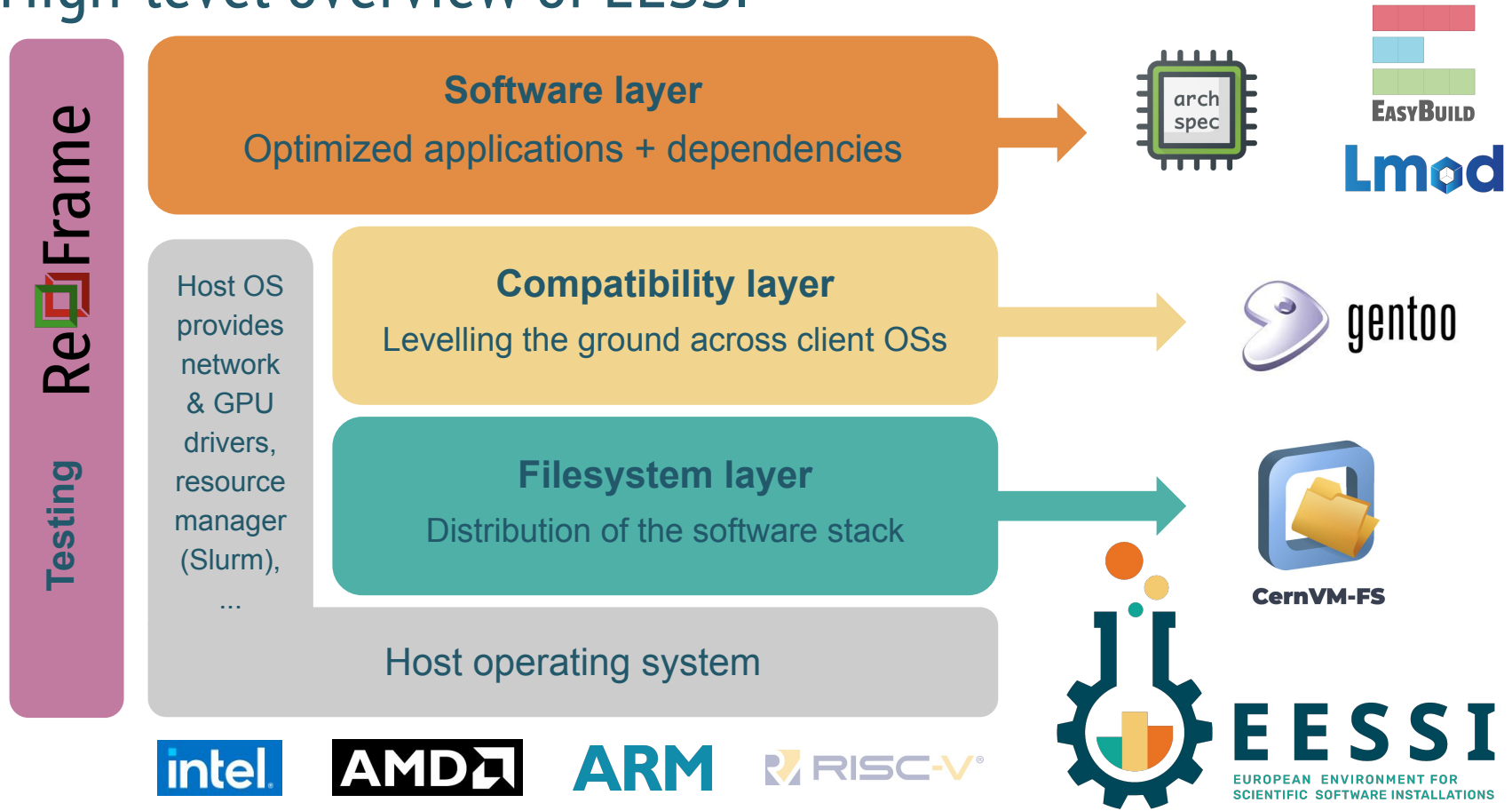
EESSI in a nutshell

- European Environment for Scientific Software Installations (EESSI)
- **Shared repository of (optimized!) scientific software installations**
- Uniform way of providing software to users, regardless of the system they use!
- Should work on any Linux OS (+ WSL, macOS via Lima) and system architecture
- From laptops and personal workstations to HPC clusters and cloud
- Support for different CPU (micro)architectures, interconnects, GPUs, etc.
- **Focus on performance, automation, testing, collaboration**

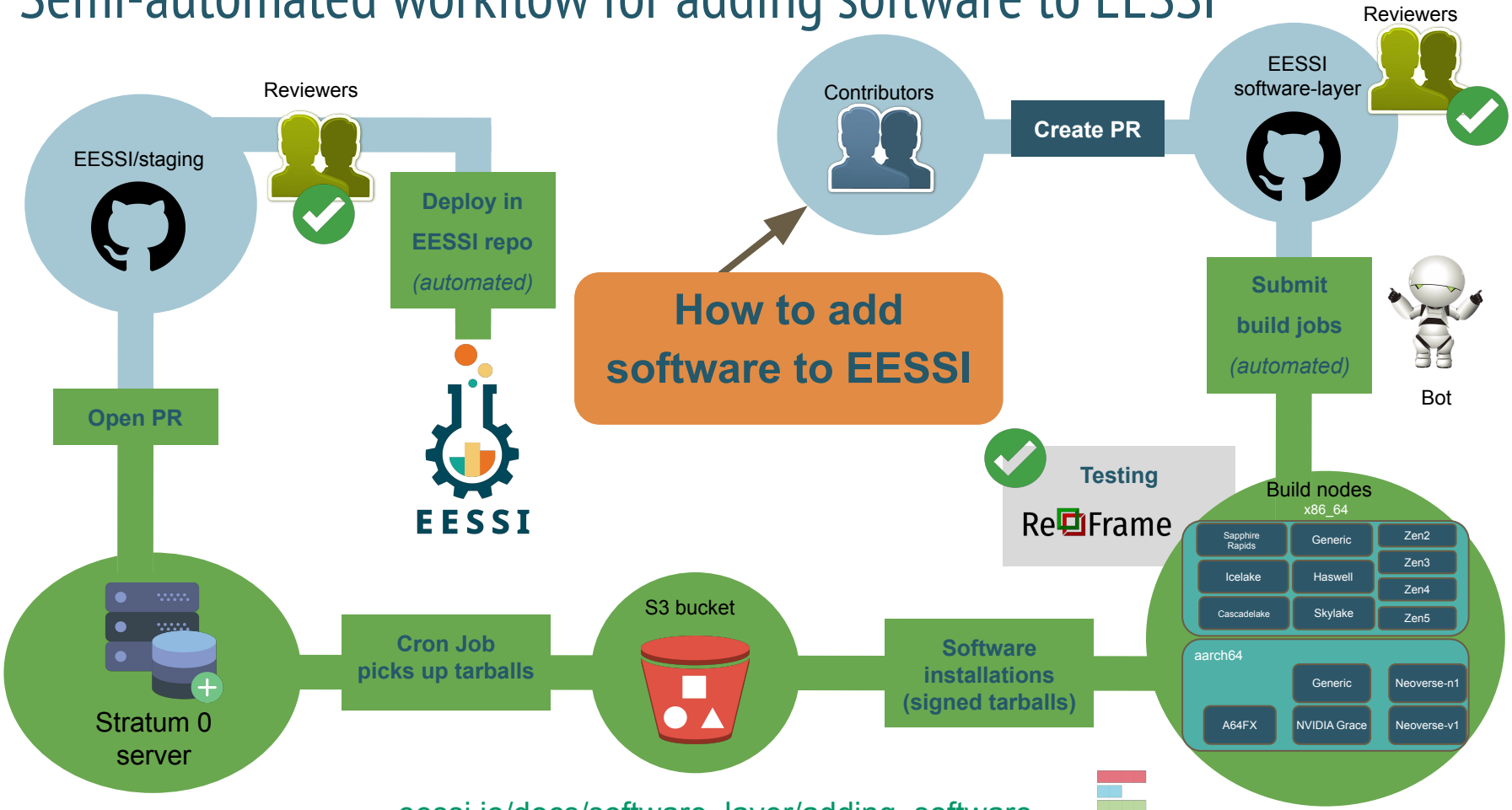
https://eessi.io

https://eessi.io/docs

High-level overview of EESSI



Semi-automated workflow for adding software to EESSI

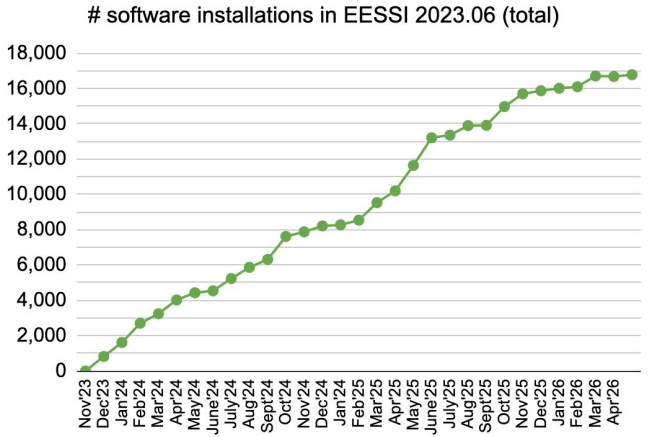


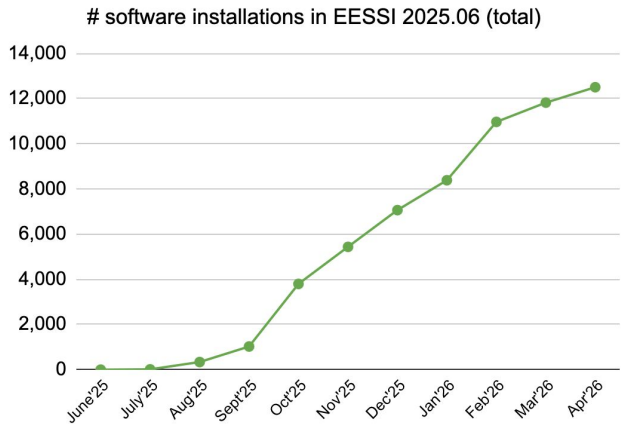
eessi.io/docs/software_layer/adding_software

Overview of available software

- Two EESSI versions available **2023.06** and **2025.06**
  - **2023.06** is the default (for now)
  - New software added to **2025.06** Using recent compiler toolchains: `foss/2024b` and `foss/2025[a,b]`
- See 👉 **Introduction to EESSI** materials for all the details
- eessi.io/docs/available_software/overview

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

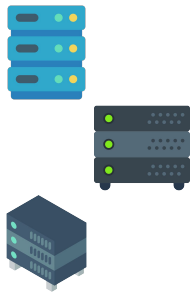




Motivation for this session

More autonomy as a site:

- Need **proprietary software** with closed licenses
  - Cannot be distributed through EESSI
  - Possibly limited to a subset of users in the system
- Need software that is not (yet) available through EESSI
  - Extra flexibility in deploying software fast
- **Custom installations** tuned for a particular system



(icons via https://www.flaticon.com/authors/smashicons)

Approach 1: Site builds on top of EESSI in a shared FS

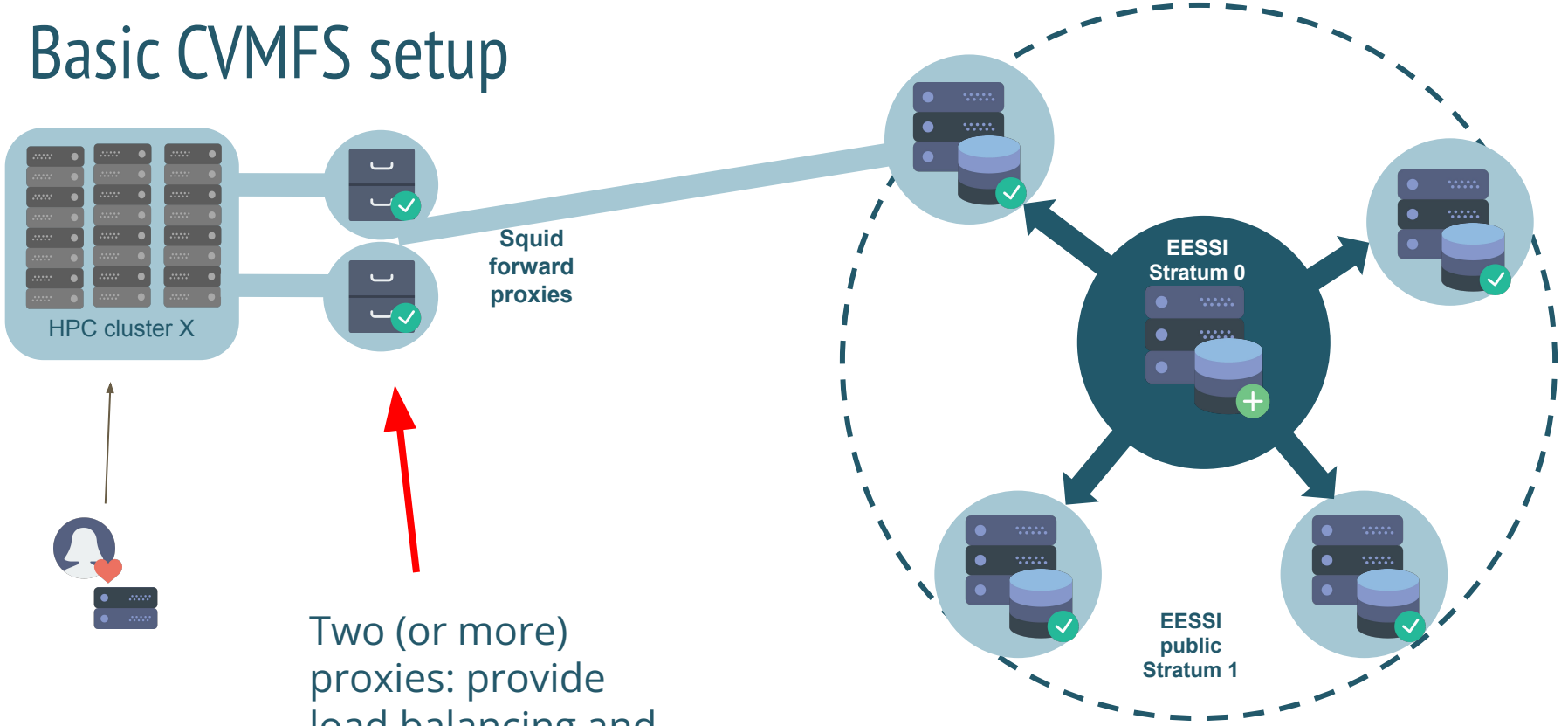
Pros

- Easy: no additional setup or knowledge needed
- Automatically installs in an architecture-specific prefix, that is automatically added to your `$MODULEPATH` at runtime

Cons

- Manual procedure (less scalable) / need to create your own automation
- Shared FS typically doesn't give the best startup performance

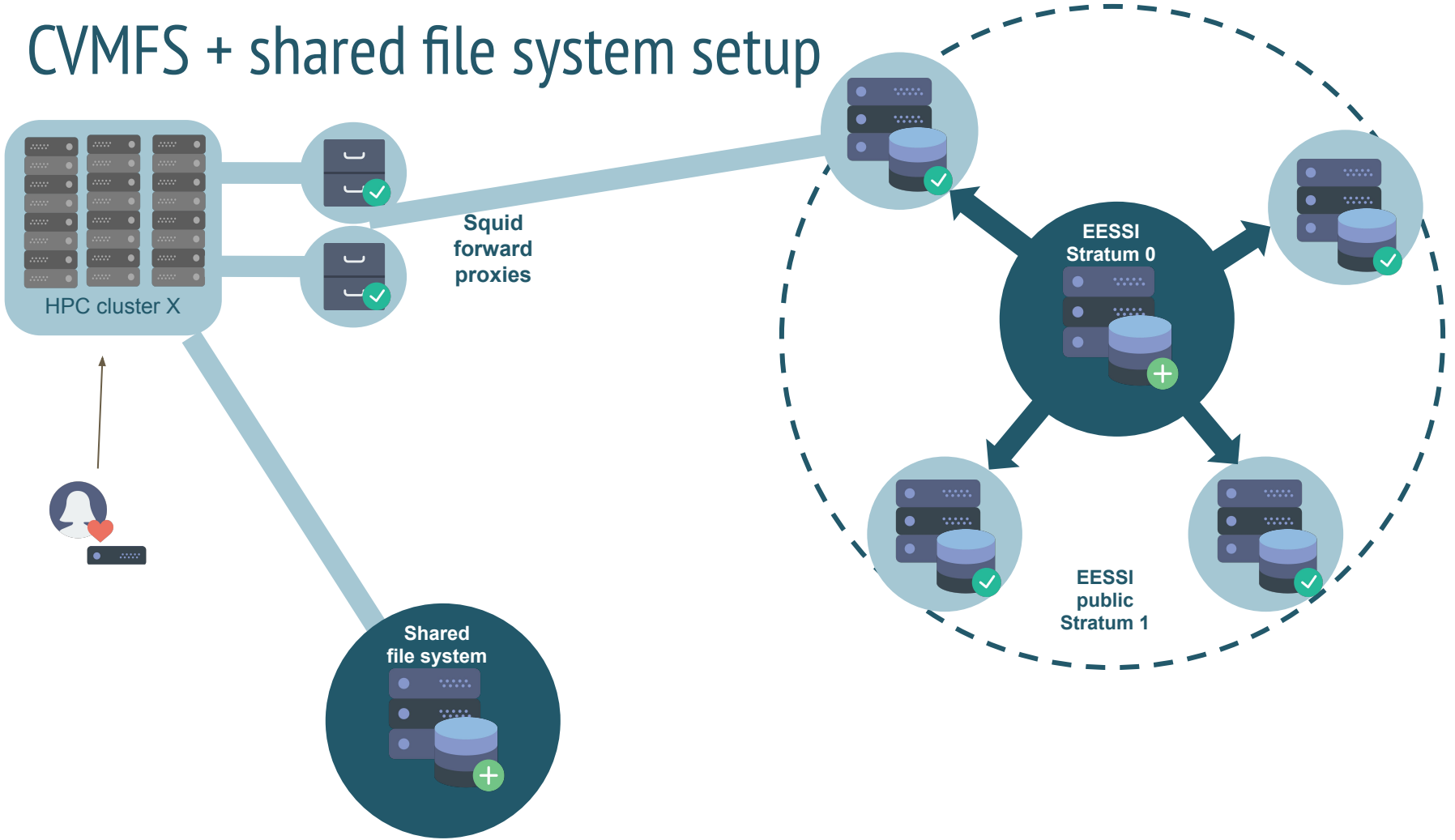
Basic CVMFS setup



Two (or more) proxies: provide load balancing and redundancy

(icons via https://www.flaticon.com/authors/smashicons)

CVMFS + shared file system setup



(icons via https://www.flaticon.com/authors/smashicons)

EESSI-extend: site installations

- **EESSI-extend** allows for user, group/project, site installations on top of EESSI
- We will focus on site installations:
  - `export EESSI_SITE_INSTALL=1`
  - Option 1: install in `/cvmfs/software.eessi.io/host_injections/`
    - CVMFS variant symlink `EESSI_HOST_INJECTIONS`
    - Symlink target determined in your CVMFS config (default: `/opt/eessi`)
    - See https://www.eessi.io/docs/site_specific_config/host_injections/
  - Option 2 (recommended):
    - `export EESSI_SITE_SOFTWARE_PREFIX=/my/preferred/site/path`
  - `module load EESSI/<EESSI_VERSION>`
  - `module load EESSI-extend`

EESSI-extend: site installations

WARNING:

- `EESSI-extend` does native optimization, and installs in a prefix specific to your host architecture
- You'll have to do installations on *every node type* you want the module to be available on

Demo scenario

University of Groningen "Hábrók" cluster

- CernVM-FS already installed, EESSI is available
- Empty directory for site installations
- Unprivileged user account for doing the site installations
- **EESSI-extend** for local stack
- **OS:** AlmaLinux 9.8
- **CPU architecture:** `x86/amd/zen3`

Demo: EESSI-extend & shared filesystem

Multiscale

EESSI

Prepare the software installation prefix

```
[bob@merel1 ~]$ sudo mkdir /software
[bob@merel1 ~]$ sudo chown f115372:f115372 /software
[bob@merel1 ~]$
```

Demo: EESSI-extend & shared filesystem

Configure EESSI for site installations

```
[f115372@merel1 ~]$ export EESSI_SITE_INSTALL=1
[f115372@merel1 ~]$ export EESSI_SITE_SOFTWARE_PREFIX=/software
[f115372@merel1 ~]$ module load EESSI/2025.06
Module for EESSI/2025.06 loaded successfully (requires '--force' option to
unload or purge)

Lmod is automatically replacing "2023.01" with "EESSI/2025.06".

Module for EESSI/2025.06 loaded successfully (requires '--force' option to
unload or purge)
[f115372@merel1 ~]$ module load EESSI-extend
-- Using /tmp/$USER as a temporary working directory for installations, you can
override this by setting the environment variable WORKING_DIR and reloading the
module (e.g., /dev/shm
is a common option)
-- To create installations for EESSI, you _must_ have write permissions to
/software/versions/2025.06/software/linux/x86_64/amd/zen3
-- You may wish to configure a sources directory for EasyBuild (for example, via
setting the environment variable EASYBUILD_SOURCEPATH) to allow you to reuse
existing sources for
packages.
```

Demo: EESSI-extend & shared filesystem

Multiscale

EESSI

Let's check a few things before we start installing...

```
[f115372@merel1 ~]$ echo $EESSI_SITE_SOFTWARE_PATH
/software/versions/2025.06/software/linux/x86_64/amd/zen3

[f115372@merel1 ~]$ which eb
/cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen3/software/EasyBuild/5.3.0/bin/eb

[f115372@merel1 ~]$ eb --show-config
#
# Current EasyBuild configuration
# (C: command line argument, D: default value, E: environment variable, F: configuration file)
#
allow-loaded-modules                      (E) = EasyBuild, EESSI-extend
buildpath                                 (E) = /tmp/f115372/easybuild/build
bwrap-installpath                         (E) = /tmp/f115372/easybuild/bwrap
containerpath                             (E) = /tmp/f115372/easybuild/containers
cuda-sanity-check-error-on-failed-checks  (E) = True
debug                                     (E) = True
experimental                              (E) = True
fail-on-mod-files-gcccore                 (E) = True
filter-deps                               (E) = binutils, bzip2, DBus, flex, gettext, gperf, help2man,
intltool, libreadline, makeinfo, ncurses, NVPL, ParMETIS, util-linux, XZ, zlib
filter-env-vars                           (E) = LD_LIBRARY_PATH
hooks                                     (E) =
/cvmfs/software.eessi.io/versions/2025.06/init/easybuild/eb_hooks.py
ignore-osdeps                             (E) = True
installpath                               (E) = /software/versions/2025.06/software/linux/x86_64/amd/zen3
...
```

Demo: EESSI-extend & shared filesystem

Multiscale

EESSI

Install some software! 🚀

```
[f115372@merel1 ~]$ eb cowsay-3.04.eb
== Temporary log file in case of crash /tmp/eb-xsg89v2a/easybuild-njcc_lo3.log
== found valid index for
/cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen3/software/EasyBuil
d/5.3.0/easybuild/easyconfigs, so using it...
== Running parse hook for cowsay-3.04.eb...


...


== COMPLETED: Installation ended successfully (took 2 secs)
== Results of the build can be found in the log file(s)
/software/versions/2025.06/software/linux/x86_64/amd/zen3/software/cowsay/3.04/easybuild/e
asybuild-cowsay-3.04-20260529.155525.log.bz2
== Running post-easyblock hook...

== Build succeeded for 1 out of 1 (total: 4 secs)
== Summary:
   * [SUCCESS] cowsay/3.04
== Temporary log file(s) /tmp/eb-xsg89v2a/easybuild-njcc_lo3.log* have been removed.
== Temporary directory /tmp/eb-xsg89v2a has been removed.
```

Demo: EESSI-extend & shared filesystem

Multiscale

EESSI

Install some software! 🚀

```
[f115372@merel1 ~]$ eb attr-2.5.2-GCCcore-14.3.0.eb
== Temporary log file in case of crash /tmp/eb-ohalf52w/easybuild-zyoesfhm.log
== found valid index for
/cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen3/software/EasyBuil
d/5.3.0/easybuild/easyconfigs, so using it...
== Running parse hook for attr-2.5.2-GCCcore-14.3.0.eb...


...


== COMPLETED: Installation ended successfully (took 31 secs)
== Results of the build can be found in the log file(s)
/software/versions/2025.06/software/linux/x86_64/amd/zen3/software/attr/2.5.2-GCCcore-14.3
.0/easybuild/easybuild-attr-2.5.2-20260529.155904.log.bz2
== Running post-easyblock hook...

== Build succeeded for 1 out of 1 (total: 33 secs)
== Summary:
   * [SUCCESS] attr/2.5.2-GCCcore-14.3.0
== Temporary log file(s) /tmp/eb-ohalf52w/easybuild-zyoesfhm.log* have been removed.
== Temporary directory /tmp/eb-ohalf52w has been removed.
```

Demo: EESSI-extend & shared filesystem

Use the software 🎉🐄

```
[f115372@merel1 ~]$ module avail

---- /software/versions/2025.06/software/linux/x86_64/amd/zen3/modules/all ----
   attr/2.5.2-GCCcore-14.3.0    cowsay/3.04

---- /cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen3/modules/all
----
   Abseil/20240722.0-GCCcore-13.3.0
   absl-py/2.1.0-GCCcore-13.3.0
   ...

[f115372@merel1 ~]$ module load cowsay/3.04
[f115372@merel1 ~]$ cowsay "EESSI keeps the clusters moo-ving."
 _____________________________________
< EESSI keeps the clusters moo-ving. >
 -------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Build container

- Host isolation, very minimal container
  - Prevent builds from picking up system libraries
  - Fixed/predictable build environment
- Use the `eessi_container.sh` from `software-layer-scripts`
  - `eessi_container.sh -b $EESSI_SITE_SOFTWARE_PREFIX`
  - In the container:
    - `export EESSI_SITE_INSTALL=1`
    - `export EESSI_SITE_SOFTWARE_PREFIX=/my/preferred/site/path`
    - `module load EESSI/<EESSI_VERSION>`
    - `module load EESSI-extend/<EESSI_VERSION>-easybuild`
    - `eb <easyconfig>`

See: https://www.eessi.io/docs/getting_access/eessi_container/

Approach 2: Leveraging all of EESSI for site-builds

Pros

- Highly automated
- Scalable to many architectures & installations
- List of software you've build in a GitHub repo
- Share maintenance on automation with EESSI community
- End-user look & feel very similar to EESSI

Cons

- More setup time
- Requires more extensive knowledge (CVMFS, build bot / GitHub App, object store)
- More hardware resources (CVMFS infra, bot infra)
- More components (software/hardware) to maintain

Leveraging all of EESSI for site-builds

The goal:

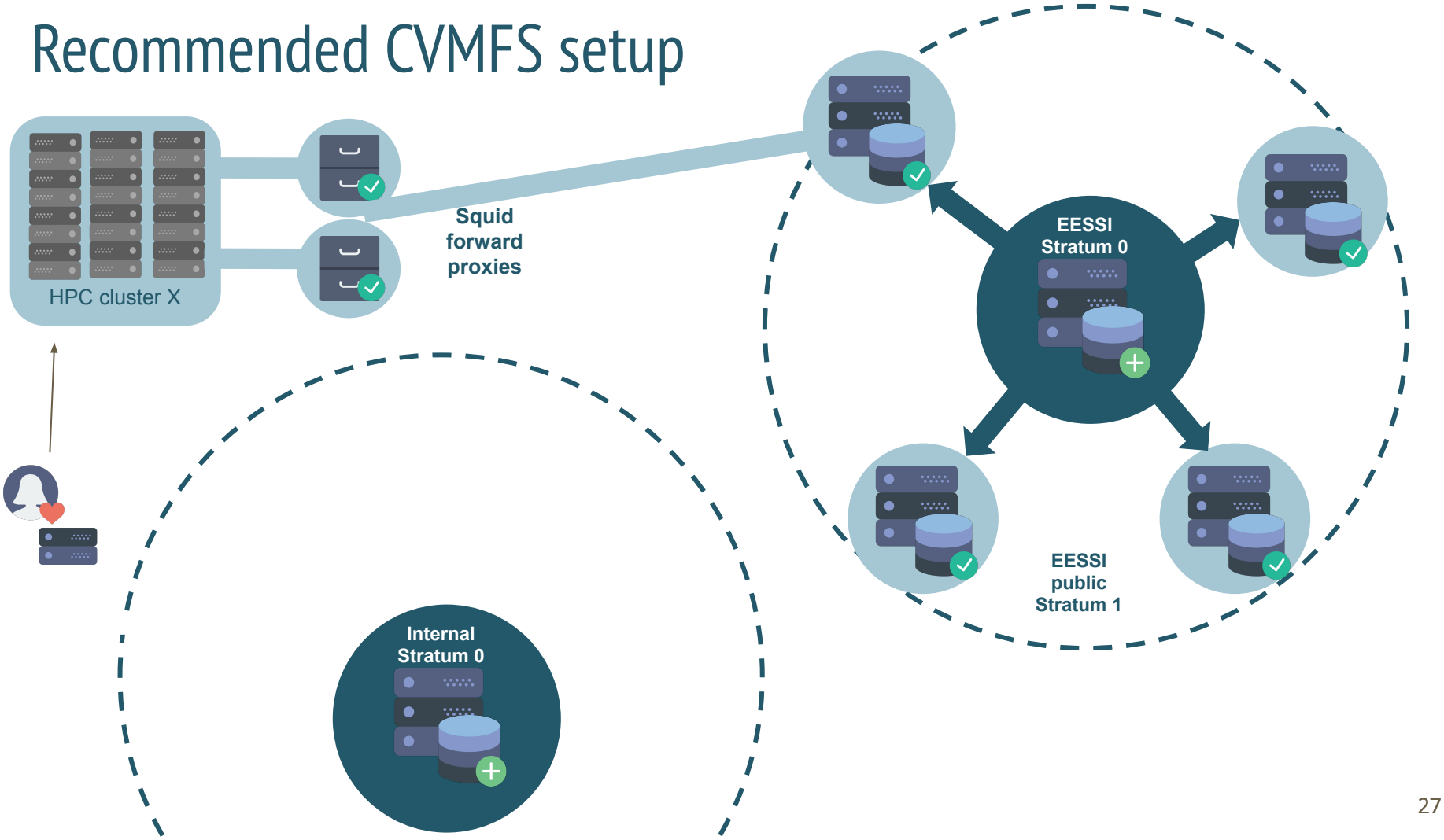
- Build using dependencies from EESSI…
- in a CVMFS repository of our own…
- with the EESSI build bot…
- using the same build scripts EESSI uses (from `EESSI/software-layer-scripts`)

Leveraging all of EESSI for site-builds

What we need:

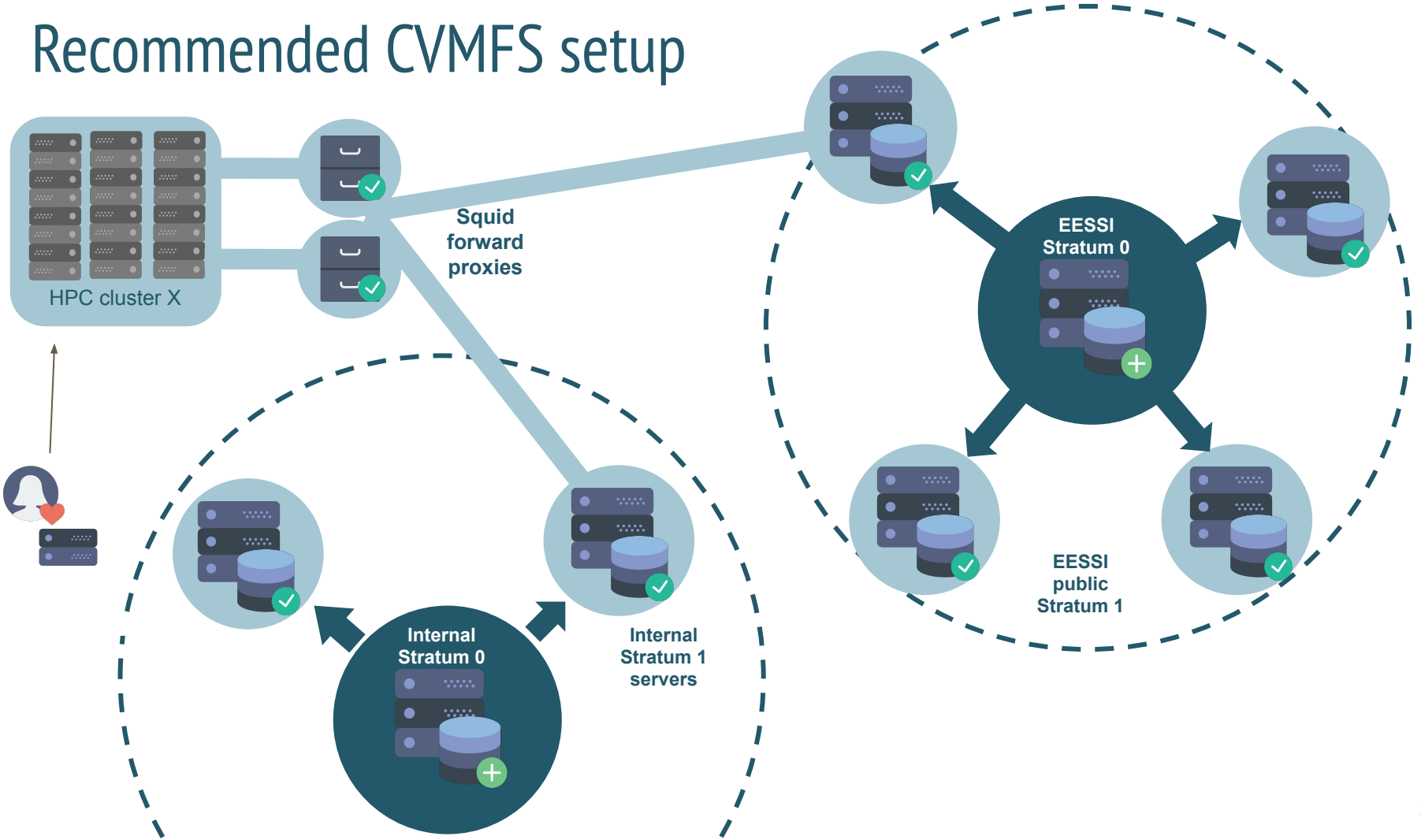
- Site specific CVMFS infrastructure: a Stratum 0, Stratum 1, proxies, clients
- One or more instances of the EESSI build bot, typically deployed on the cluster(s) we want to build for
    - We want to build natively, i.e. build on a zen4 node for a zen4 target
- A bucket in an AWS S3-compatible object store
- A GitHub repository to list easystack files with software we want to build ‘on top’ of EESSI
- Optionally: an automated procedure to ingest the tarballs

Recommended CVMFS setup



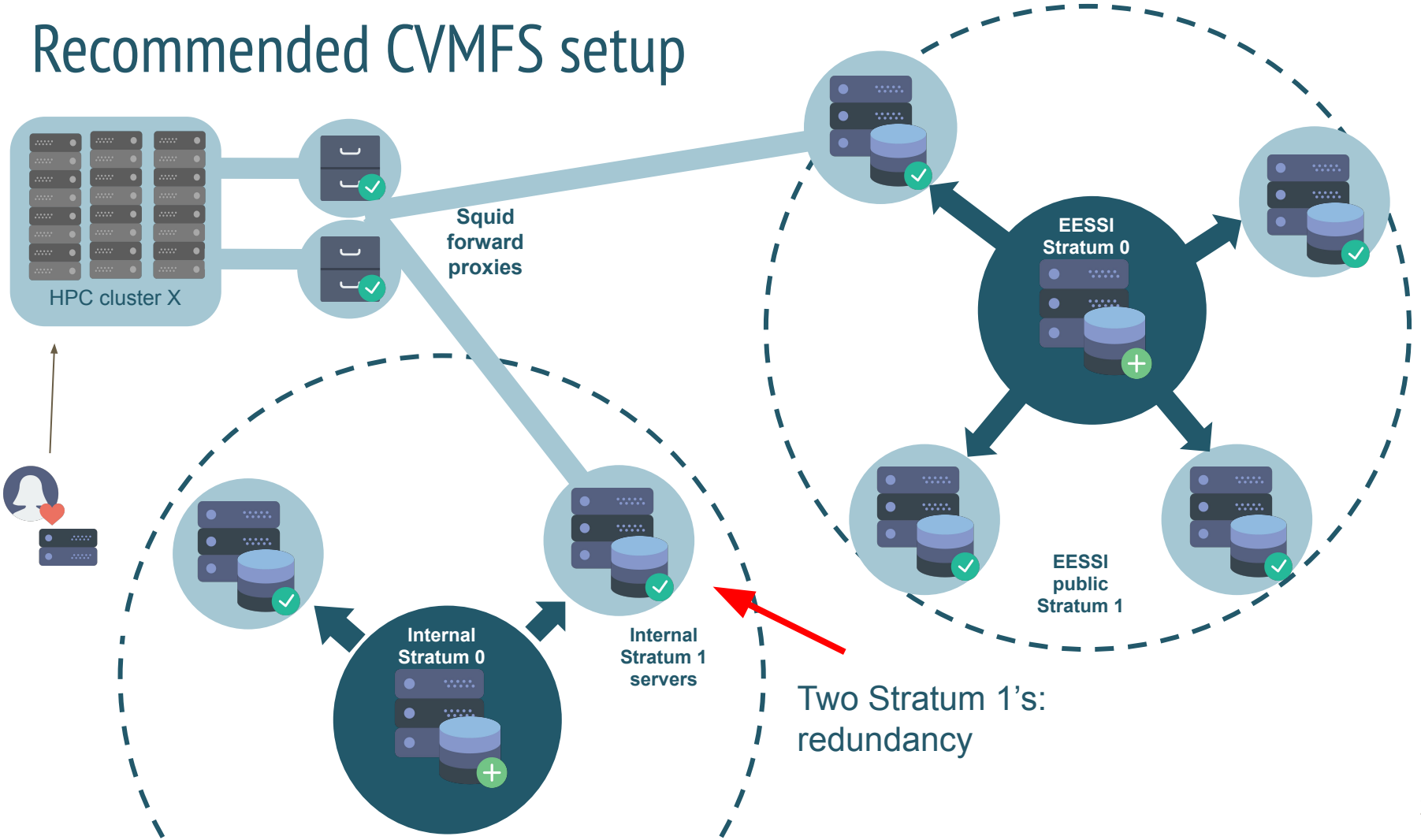
(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup



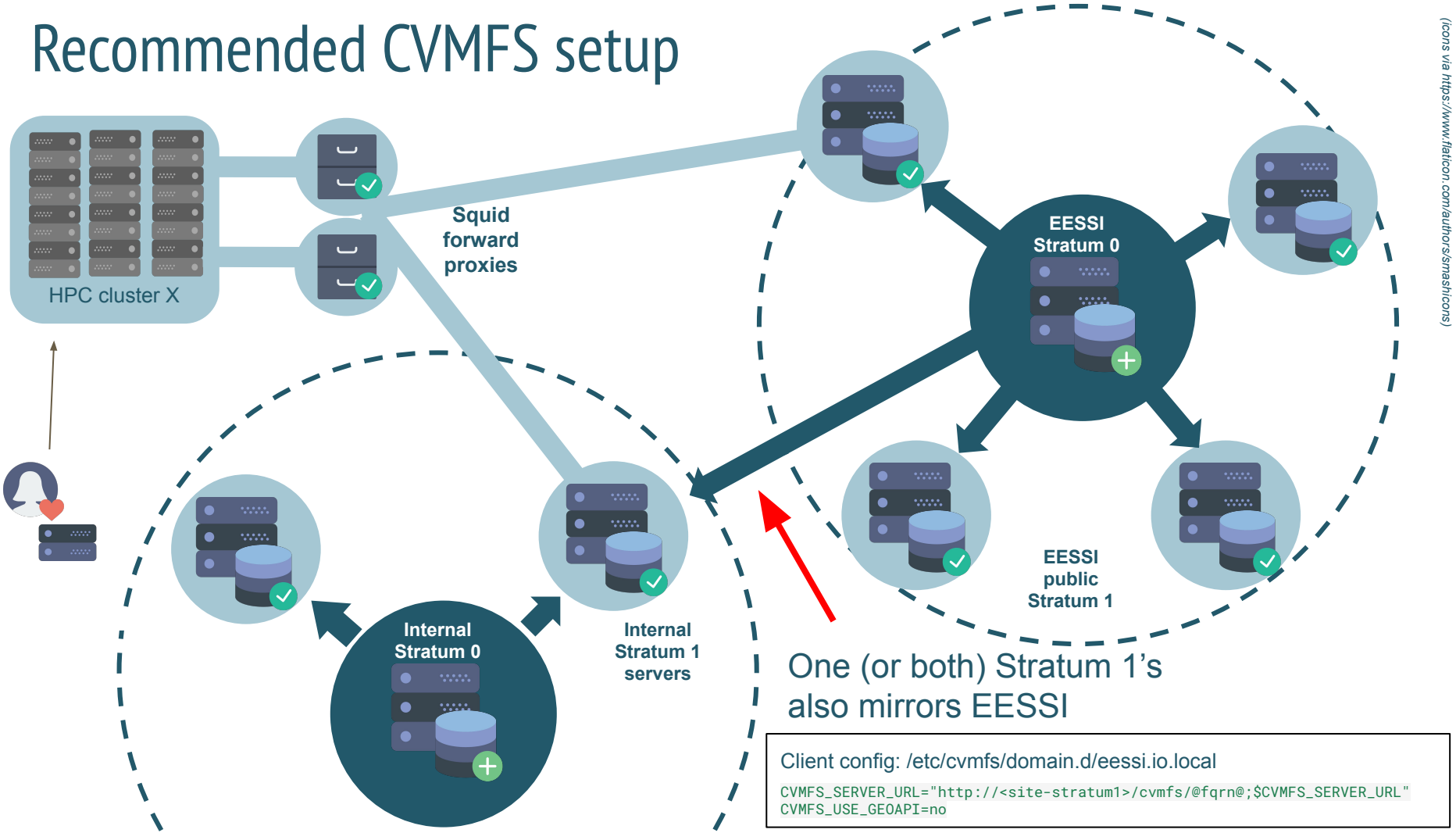
(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup

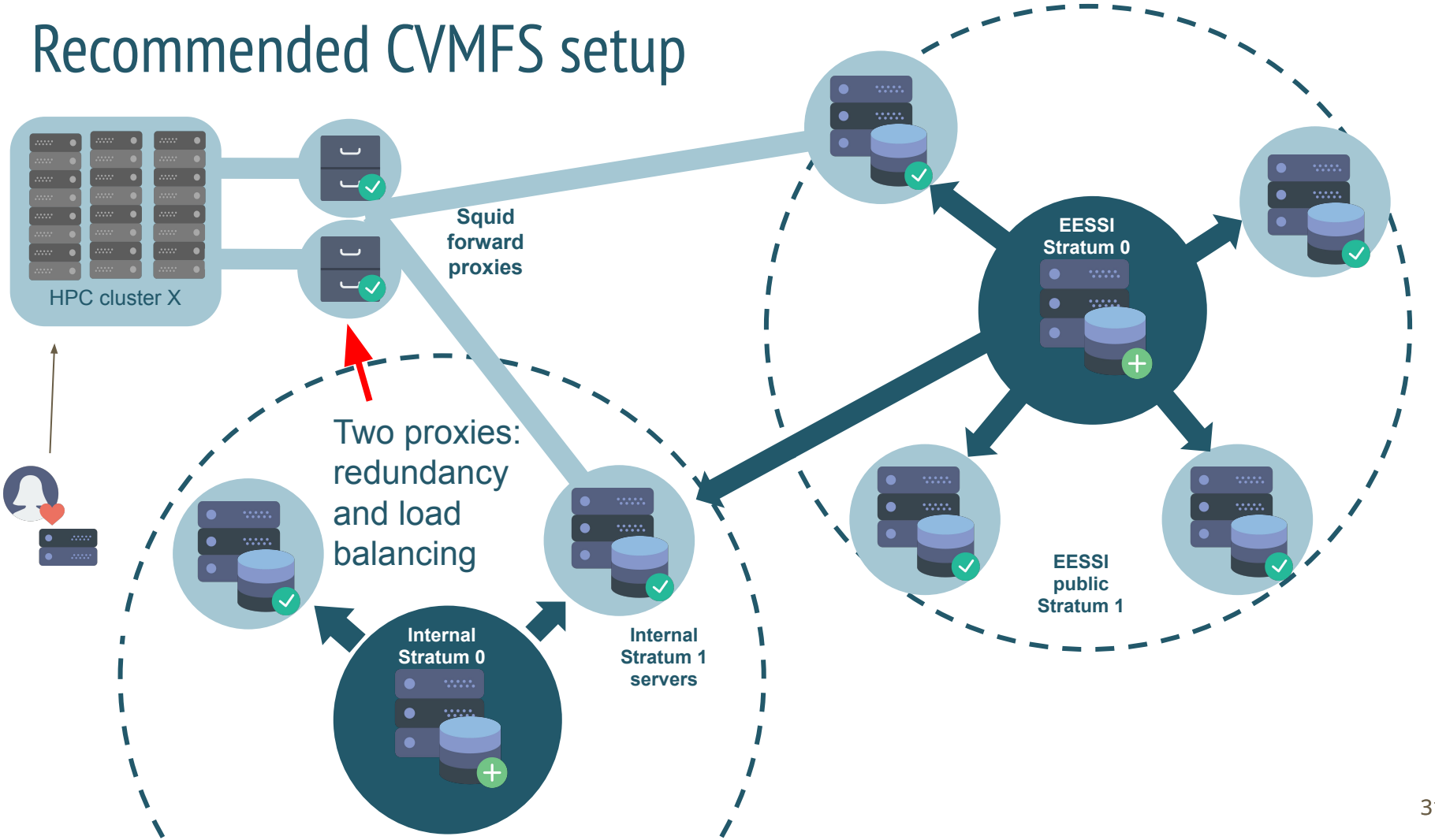


(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup



Recommended CVMFS setup



(icons via https://www.flaticon.com/authors/smashicons)

Setting up a site-specific Stratum 0

For this demo, I've set up a repo `software.caspar.nl`

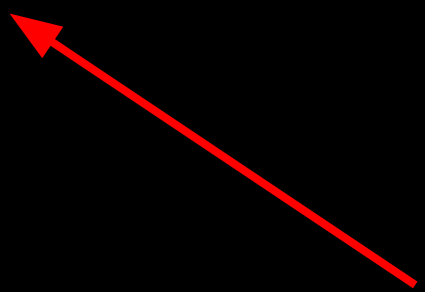
- Install `cvmfs` and `cvmfs-server` packages
- Create the repo, owned by root (we'll get back to this!): `sudo cvmfs_server mkfs -o root software.caspar.nl`
- Make sure the `apache2` (Ubuntu/Debian/...) or `httpd` (CentOS/RHEL/Fedora/...) service is running
- Note: `software.caspar.nl` is not a URL, it's a CVMFS repo 'name'.
    - One would commonly use the same name for DNS (no DNS for this demo)

https://cvmfs-contrib.github.io/cvmfs-tutorial-2021/02_stratum0_client/#21-setting-up-the-stratum-0

Setting up a site-specific Stratum 0

Demo

```
cvanleeuwe@cvmfss0:~$ cvmfs_server list | grep caspar\.nl
software.caspar.nl (stratum0 / local)
```



Lists all repositories served by this machine

Setting up a site-specific Stratum 0

Demo

```
cvanleeuwe@cvmfss0:~$ mount | grep caspar\.nl
software.caspar.nl on /var/spool/cvmfs/software.caspar.nl/rdonly type fuse
(ro,nodev,relatime,user_id=0,group_id=0,default_permissions,allow_other)
overlay_software.caspar.nl on /cvmfs/software.caspar.nl type  overlay
(ro,nodev,relatime,lowerdir=/var/spool/cvmfs/software.caspar.nl/ rdonly,upperdir=
/var/spool/cvmfs/software.caspar.nl/ scratch/current,workdir=/var/spool/cvmfs/sof
tware.caspar.nl/ofs_workdir,uuid=on,xino=off,userxattr)
```

An overlay FS combining a read-only dir .../rdonly/ with a writable dir ../scratch/current/ to which open transactions will write

Setting up a site-specific Stratum 0

Demo

```
cvanleeuwe@cvmfss0:~$ ls -al /srv/cvmfs/software.caspar.nl
total 36
drwxr-xr-x   3 root root 4096 May 28 17:45 .
drwxr-xr-x   5 root root 4096 May 27 14:19 ..
-rw-r--r--   1 root root   68 May 27 14:19 .cvmfs_master_replica
-rw-r--r--   1 root root  600 May 28 17:45 .cvmfspublished
-rw-r--r--   1 root root 9216 May 28 17:45 .cvmfsreflog
-rw-r--r--   1 root root  411 May 27 14:25 .cvmfswhitelist
drwxr-xr-x 259 root root 4096 May 27 14:19 data
```

This is where CVMFS stores its actual data.
Can be changed by making /srv/cvmfs a symlink

Setting up a site-specific Stratum 0

Demo

```
cvanleeuwe@cvmfss0:~$ ls -al /cvmfs/software.caspar.nl
total 15
drwxr-xr-x 1 root root 4096 May 28 17:45 .
drwxr-xr-x 4 root root 4096 May 27 14:19 ..
-rw-r--r-- 1 root root 1100 May 27 14:30 .cvmfsdirtab
-rw-r--r-- 1 root root   48 May 27 14:19 new_repository
drwxr-xr-x 2 root root 4096 May 28 17:45 versions
```

The overlay FS presents us with a current view of the repository, as it would be visible on the client (after all transactions are published)

Setting up a site-specific Stratum 1 / proxies

Won't do in my demo setup, but essential for redundancy, load balancing, etc

- Set up two Stratum 1 instances *and* at least two proxies
  - Configure your clients with a proxy group:

    `CVMFS_HTTP_PROXY="<proxy1>|<proxy2>"` => Causes clients to randomly select one of the two proxies.
  - You can reuse the same proxies you use for `software.eessi.io`

https://cvmfs-contrib.github.io/cvmfs-tutorial-2021/03_stratum1_proxies/

Setting up a site-specific Stratum 1 / proxies

Won't do in my demo setup, but essential for redundancy, load balancing, etc

- If you're short on machines:
  - You *could* set up 1 Stratum 1 and also list your Stratum 0 in `CVMFS_SERVER_URL`
  - OR: set up two Stratum 1 instances, no proxies for site repo (you'll still need proxies for EESSI...)
    - set `CVMFS_USE_GEOAPI="no"` and `"CVMFS_HTTP_PROXY=DIRECT"` on your clients
    - Configure half your clients with `CVMFS_SERVER_URL="<instance_1>;<instance_2>"` and half with `CVMFS_SERVER_URL="<instance_2>;<instance_1>"` for load balancing
  - If you also have a private Stratum 1 for `software.eessi.io`, you could reuse the same machine to serve one of your site Stratum 1 instances.

https://cvmfs-contrib.github.io/cvmfs-tutorial-2021/03_stratum1_proxies/

Configuring your local clients

- Install the `cvmfs` package
- Read public key on Stratum 0 and put it in `/etc/cvmfs/keys/caspar.nl/software.caspar.nl.pub`
- Minimal repository config: `/etc/cvmfs/config.d/software.caspar.nl.conf`
- Add repository to `CVMFS_REPOSITORIES` in `/etc/cvmfs/default.local`
- For this demo, I connect *directly* to the Stratum 0. That is not recommended for anything but a test setup.

https://cvmfs-contrib.github.io/cvmfs-tutorial-2021/02_stratum0_client/#22-setting-up-a-client

Configuring your local clients

Multiscale

EESSI

Demo

```
cvanleeuwe@debugcvmfs:~$ cat /etc/cvmfs/default.local
CVMFS_REPOSITORIES=software.caspar.nl
CVMFS_CLIENT_PROFILE=single
CVMFS_QUOTA_LIMIT=10000
```

Configuring your local clients

Demo

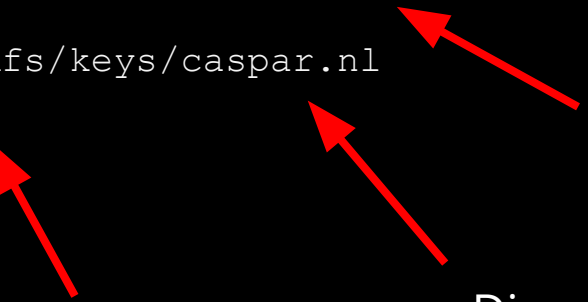
```
cvanleeuwe@debugcvmfs:~$ cat /etc/cvmfs/keys/caspar.nl/software.caspar.nl.pub
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApbgFB1tB3vW3pgdpj+cv
KQcQNgNA26ZJjma4fwuUNOkkTVgEb+TMDzn+NtPOY8rIKCrBkQf9Z7/sEA/gJHT0
9uOiFleWjbQT9ZVLeraDzLNY6DSUkYMeWWPFysmELga/pz3H6sTySCjDKKsS6Yb+
+bRVlba7lkIdUpqCjUW/Ce9MNG36S/cfX+J/s2Q03ptQlstOPYwe1QKaybttAEVC
dMKfuRy0lr8x8akDNsoWC1B4SIRN+sM3AiyRF5M2kxvdd5DfX7Z2cJ8s2wWUDgFL
b2jdtleW3oEx0zUojcpeNt11GOH72rEahBuKSoOUFw1/gaWULH9epAaL/bgNA94v
UwIDAQAB
-----END PUBLIC KEY-----
```

Public key, as obtained from `/etc/cvmfs/keys` on the CVMFS Stratum 0

Configuring your local clients

Demo

```
cvanleeuwe@debugcvmfs:~$ cat /etc/cvmfs/config.d/software.caspar.nl.conf
CVMFS_SERVER_URL="http://145.38.193.197/cvmfs/@fqrn@"
CVMFS_USE_GEOAPI="no"
CVMFS_KEYS_DIR=/etc/cvmfs/keys/caspar.nl
CVMFS_HTTP_PROXY=DIRECT
```



URL for Stratum 0

Connect directly to Stratum 0 / 1, not via proxy (demo setup only!)

Directory where to find the key for this repo

Setting up an EESSI build bot: what is it?

- EESSI bot starts jobs and runs scripts when triggered by GitHub events
  - Can start jobs with any script
  - Replace or adapt the EESSI `bot/build.sh` with your site's build script
- Store tarballs where appropriate (shared file system?)

eessi-bot-aws bot commented 2 days ago • edited

New job on instance `eessi-bot-mc-aws` for CPU micro-architecture `x86_64-amd-zen2` for repository `eessi.io-2023.06-software` in job dir `/project/def-users/SHARED/jobs/2025.05/pr_1101/65821`

date	job status	comment
May 28 13:58:07 UTC 2025	submitted	job id `65821` awaits release by job manager
May 28 13:59:03 UTC 2025	released	job awaits launch by Slurm scheduler
May 28 14:06:48 UTC 2025	running	job `65821` is running
May 28 14:22:08 UTC 2025	finished	▸ 😁 SUCCESS *(click triangle for details)*
May 28 14:22:08 UTC 2025	test result	▸ 😁 SUCCESS *(click triangle for details)*
May 28 18:42:53 UTC 2025	uploaded	transfer of `eessi-2023.06-software-linux-x86_64-amd-zen2-17484414780.tar.gz` to S3 bucket succeeded

Setting up an EESSI build bot: what is it?

- The existing bot has a few requirements
  - Access to a Slurm cluster and permissions to start jobs
  - GitHub App and GitHub repository to communicate through
  - A shared filesystem to write to

- Note: GitLab compatibility is actively worked on

Setting up an EESSI build bot instance

EESSI build bot

- Set up a smee channel to relay GitHub events (one per bot instance!): https://smee.io/new . Note down the URL!
- Run a smee client that connects to that channel

https://github.com/EESSI/eessi-bot-software-layer

Setting up an EESSI build bot instance

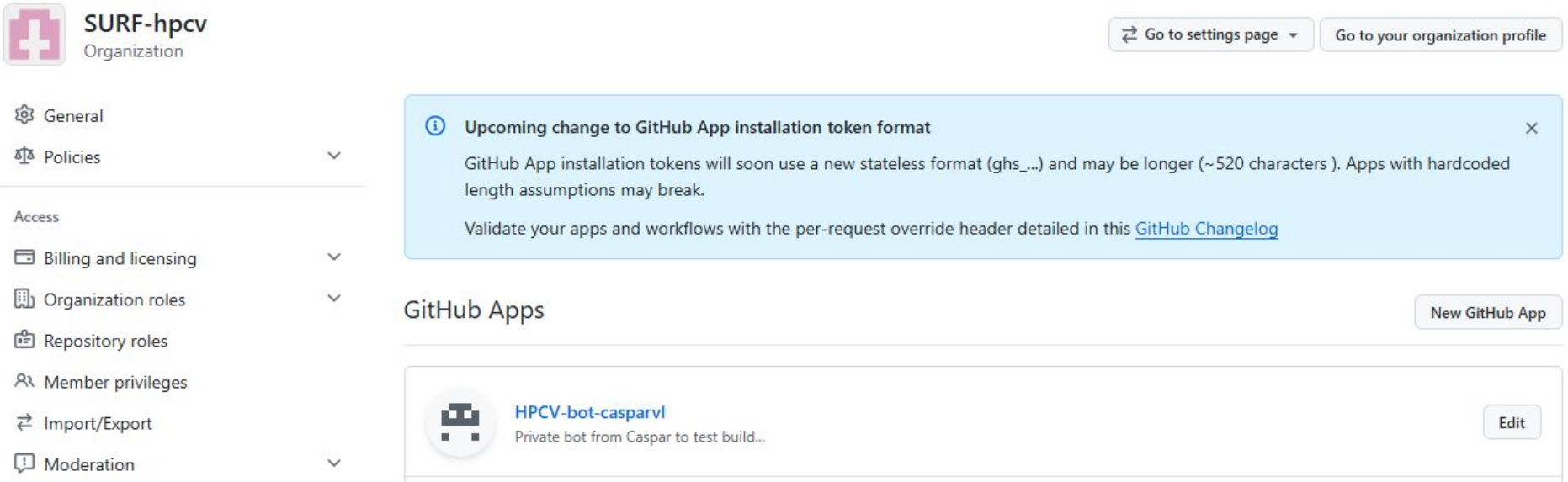
EESSI build bot

- Register a GitHub App
  https://github.com/EESSI/eessi-bot-software-layer#step-2-registering-a-github-app
  - If you have a GitHub organization (i.e github.com/org) go to the Settings of that org => developer settings => Github Apps
  - Webhook URL: the smee channel you created
  - Secret: create a secret to verify the webhook sender
  - Set relevant permissions
  - Subscribe to events this app should act on
  - Click “Create GitHub app”, the create a private key. The bot instance (=a GitHub App) will use this to identify itself to GitHub.

https://github.com/EESSI/eessi-bot-software-layer

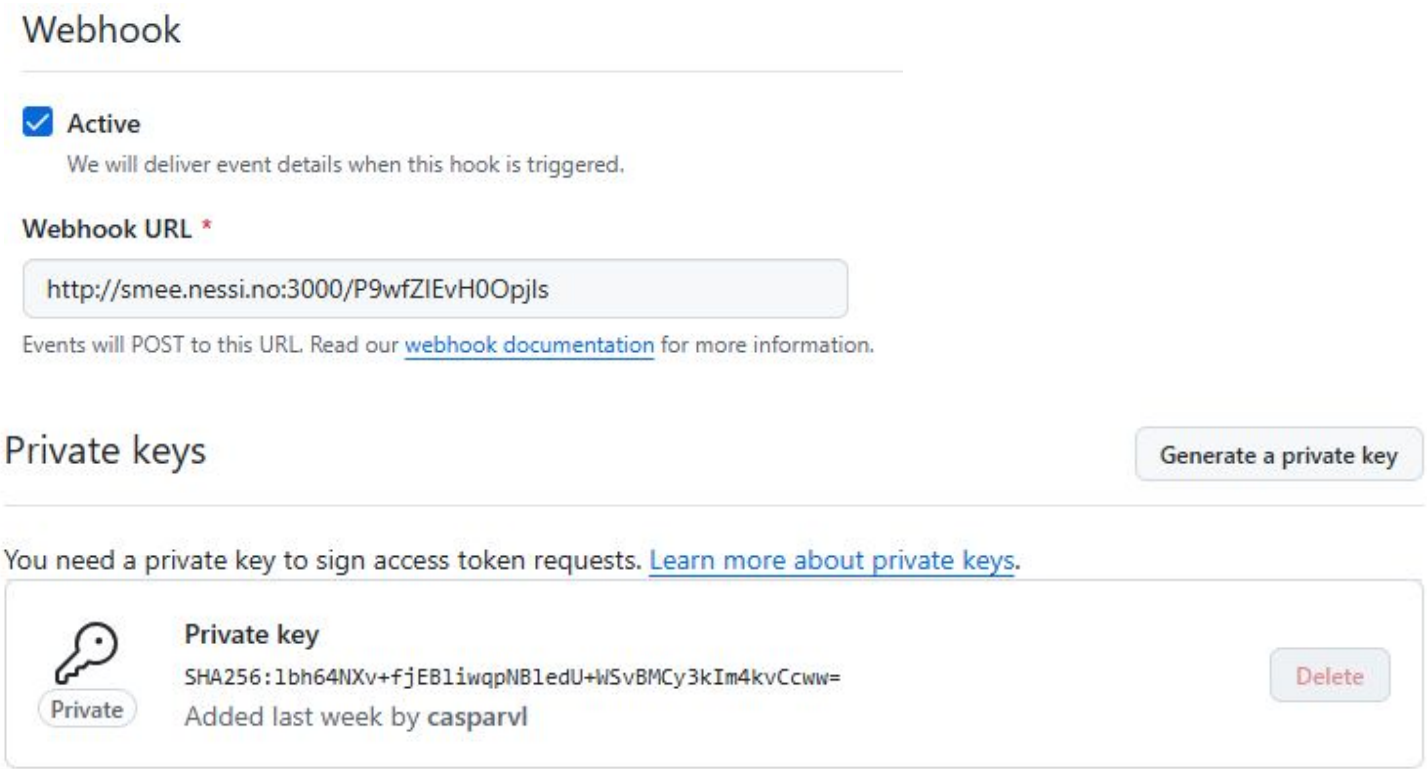
Setting up an EESSI build bot instance

EESSI build bot: register app



Setting up an EESSI build bot instance

EESSI build bot: register app



Setting up an EESSI build bot instance

EESSI build bot: register app

Permissions

User permissions are granted on an individual user basis as part of the User authorization flow.
Read our permissions documentation for information about specific permissions.

Changes to permissions will be applied to all future installations. Current users will be prompted to accept any changes and enable the new permissions on their installation.

Repository permissions 2 selected 1 mandatory

Repository permissions permit access to repositories and related resources.

Issues Selected

Issues and related comments, assignees, labels, and milestones. Learn more.

Access: Read and write

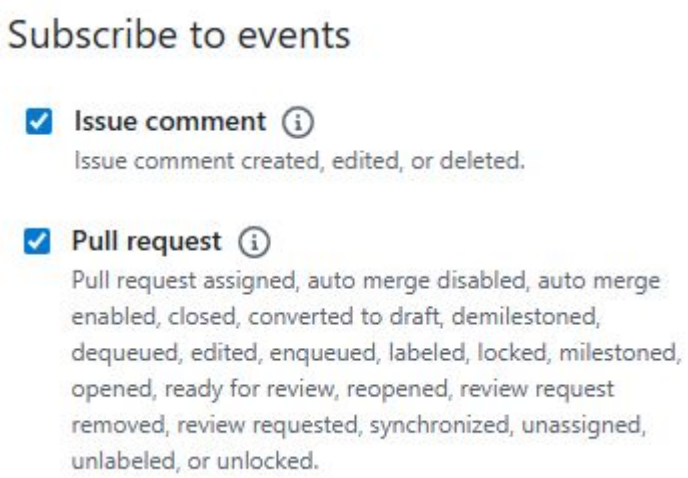
Pull requests Selected

Pull requests and related comments, assignees, labels, milestones, and merges. Learn more.

Access: Read and write

Setting up an EESSI build bot instance

EESSI build bot: register app



Setting up an EESSI build bot instance

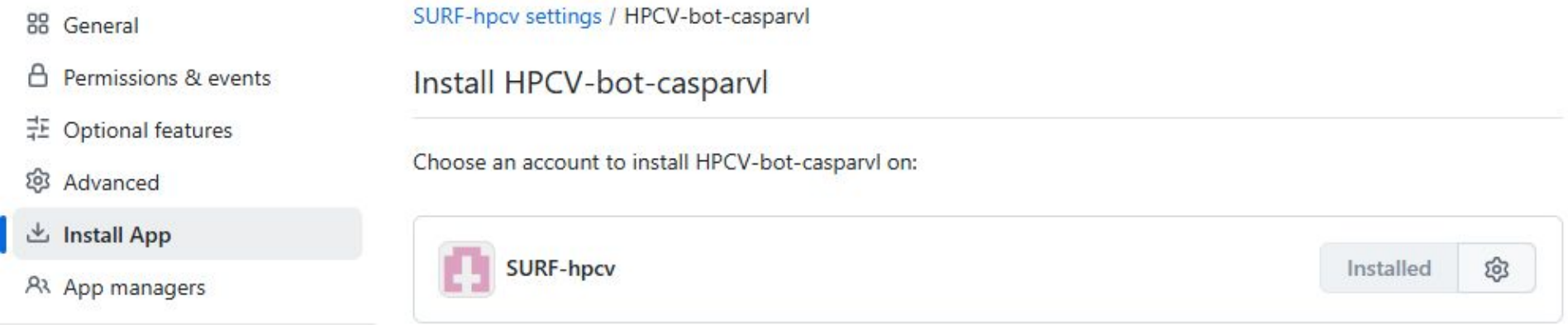
EESSI build bot

- Install the GitHub App into a repository https://github.com/EESSI/eessi-bot-software-layer#step-3-installing-the-github-app-into-a-repository
    - Select which repositories from the organization you want to install it on - typically the software-layer-like repository

https://github.com/EESSI/eessi-bot-software-layer

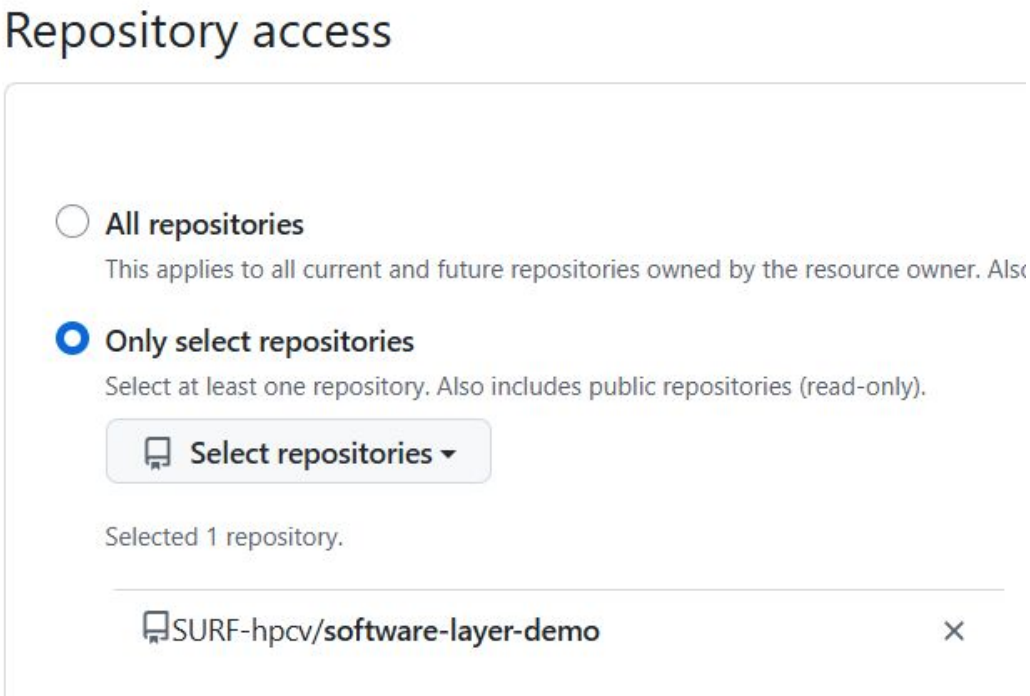
Setting up an EESSI build bot instance

EESSI build bot: install app on repo



Setting up an EESSI build bot instance

EESSI build bot: install app on repo



Setting up an EESSI build bot instance

EESSI build bot

- Install the EESSI bot on a machine
  - Should be a machine that can submit jobs through SLURM (login node, service node, etc)
  - Bot instance itself is very lightweight - building happens in jobs
  - Clone https://github.com/EESSI/eessi-bot-software-layer
  - Set up a virtual env and install the `requirements.txt`
  - Install `aws` and `jq` commands
  - Put in your environment: `GITHUB_APP_SECRET_TOKEN=<webhook secret>`
  - Configure the bot's `app.cfg` https://github.com/EESSI/eessi-bot-software-layer#step-54-create-the-configuration-file-appcfg
  - Optionally: create a ReFrame configuration file to run the EESSI test suite in the test step of any install https://github.com/EESSI/eessi-bot-software-layer#step-6-creating-a-reframe-configuration-file-for-the-test-step-only-needed-when-building-for-the-eessi-software-layer

https://github.com/EESSI/eessi-bot-software-layer

Setting up an EESSI build bot instance

EESSI build bot: install app on machine

- I typically create a small script to load my environment
  - `GITHUB_TOKEN` is not used anymore (but needs to be defined)
  - N.B. the secret below is fake :)

```
[casparl@int5 bot-instance]$ cat demo_env.sh
#!/bin/bash

source eessi_bot_hpcv_casparvl_venv/bin/activate
export GITHUB_TOKEN="foo"
export
GITHUB_APP_SECRET_TOKEN='f82452sdafa234509fuafagf230fx8ad98ag7sdvdfa89067dsfasdf
sadfad07fasdf8907asdf098asdfasdfasdf087asdf908a7sdf0asdf987adsfsadfasdfa'
```

Setting up an EESSI build bot instance

EESSI build bot: install app on machine

- Configure `app.cfg`, e.g.
  - app & installation IDs of the GH app
  - partitions you have
  - location of private key (created for GH app)
  - (shared FS) location to store jobs, logs, etc
  - which GH users have build/deploy permissions
  - bucket names, S3 endpoint URL
  - location of (CVMFS) repo configs

https://github.com/EESSI/eessi-bot-software-layer

Setting up an EESSI build bot instance

EESSI build bot: install app on machine

- CVMFS repo configurations
  - There are essentially three files per repo (same files as you use to configure you clients!)
  - Stored in a single tar-file

```
" tar.vim version v32
" Browsing tarfile
/gpfs/home4/casparl/HPCV_Software/bot-instance/ repos/surf.nl-cfg_files.tgz
" Select a file with cursor and press ENTER

default.local
surf.nl/
surf.nl/software.surf.nl.pub
surf.nl.conf
```

Setting up an EESSI build bot instance

EESSI build bot: install app on machine

- CVMFS repo configurations
  - Add repo to `repos.cfg`

```
[software.caspar.nl-2025]
repo_name = software.caspar.nl
repo_version = 2025.06
config_bundle = caspar.nl-cfg_files.tgz
config_map = {
"caspar.nl/software.caspar.nl.pub":"/etc/cvmfs/keys/caspar.nl/software.caspar.nl
.pub", "default.local":"/etc/cvmfs/default.local",
"caspar.nl.conf":"/etc/cvmfs/domain.d/caspar.nl.conf"}
container = docker://ghcr.io/eessi/build-node:debian12
```

Maps files from your config_bundle tarball to where they will be bind-mounted in build container

Build container to use

Bucket in an object store

Won't go into detail, but...

- Make bucket: `aws s3 mb s3://<bucket_name>`
- Consider setting a bucket policy that restricts access to whitelisted IP range
- Consider automatic cleanup policies
- Create token, store corresponding `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` in environment, or in `$HOME/.aws/credentials`

```
[casparl@int5 bot-instance]$ cat demo_env.sh
#!/bin/bash

…
# You can also store these in $HOME/.aws/credentials
export AWS_ACCESS_KEY_ID='SIFGYHW8867SFHG89DF'
export AWS_SECRET_ACCESS_KEY='CysPFjsdi8SDs6f0sdkFJS9s0df7GD9sFsfDS092'
```

GitHub repo to list easystack files

- A GitHub repository to list software we want to build ‘on top’ of EESSI in easystack files
- Similar to `EESSI/software-layer`
  - the same `bot/build.sh`
  - the same directory structure for where easystack files are placed
    `easystacks/<site_cvmfs_repo_name>/<EESSI_VERSION>/`
  - the same easystack naming scheme
    `eessi-<EESSI_VERSION>-eb-<eb_version>-<anything>.yml`

Example:

https://github.com/SURF-hpcv/software-layer-demo/pull/1

(Automated) ingestion on Stratum 0

A script on the Stratum 0 that

- Checks for new tarballs in the bucket
- Downloads new tarballs to the Stratum 0
- Checks the signature (optional)
- Ingests them & updates the CVMFS catalog files & updates Lmod caches
  - Can be done (soon) by leveraging `filesystem-layer/scripts/ingest-tarball.sh` *
- Moves/removes tarballs from the bucket so they only get ingested once
- Example at https://github.com/SURF-hpcv/stratum0_ingestion/blob/main/ingest_tarballs.sh

* Lmod cache updates require `software.eessi.io` to be available on your Stratum 0 (recommended) or to point it to a working Lmod installation using `LMOD_LIBEXEC_DIR`

(Automated) ingestion on Stratum 0

Making software.eessi.io available on your Stratum 0 *without* enabling the automounter

- `cvmfs_server` can't function properly with automounter enabled, so simply running `sudo cvmfs_config setup` is *not* the way to go
- Instead, we mount the repo manually (first the `cvmfs-config` repo, then `software.eessi.io`)
  - `sudo mkdir -p /cvmfs/{cvmfs-config.cern.ch ,software.eessi.io}`
  - `echo "cvmfs-config.cern.ch /cvmfs/cvmfs-config.cern.ch cvmfs defaults 0 0" >> /etc/fstab`
  - `echo "software.eessi.io /cvmfs/software.eessi.io cvmfs defaults 0 0" >> /etc/fstab`
  - `sudo systemctl daemon-reload`
  - `sudo mount -a`

(Automated) ingestion on Stratum 0

Note:

Upstream, we use https://github.com/EESSI/filesystem-layer/blob/main/scripts/automated_ingestion/ingest_bundles.py , but that is likely unnecessarily complex for sites (requires a staging repository, additional approval step, etc)

Intermezzo: nested CVMFS catalogs

- CVMFS repositories use nested catalogs for storing metadata
- Catalogs should not become too large
  - Rule of thumb: between 1k and 200k items (files/directories)
- For the EESSI repository we make a nested catalog for every software installation
  - Done using a `.cvmfsdirtab` file, see documentation and the EESSI file
- You can make one for your CVMFS repository based on the EESSI file
  - Your prefix may be slightly different
  - Some paths, e.g. for the compatibility layer, can be removed
- CVMFS includes functionality for automatically managing nested catalogs

All prep work done! Let's start...

EESSI build bot: start the three bot processes

- Smee client:

```
[casparl@int4 bot-instance]$ cat start_smee_client.sh
#!/bin/bash
singularity pull docker://deltaprojects/smee-client
singularity run smee-client_latest.sif --url
http://smee.nessi.no:3000/P9wfZlEvH0Opjls
[casparl@int4 bot-instance]$ ./start_smee_client.sh
FATAL:   Image file already exists: "smee-client_latest.sif" - will not
overwrite
Forwarding http://smee.nessi.no:3000/P9wfZlEvH0Opjls to http://127.0.0.1:3000/
Connected http://smee.nessi.no:3000/P9wfZlEvH0Opjls
```

All prep work done! Let's start...

EESSI build bot: start the three bot processes

- Event handler

```
[casparl@int4 bot-instance]$ source load_eessi_bot_env.sh
(eessi_bot_hpcv_casparvl_venv) [casparl@int4 bot-instance]$ cd
eessi-bot-software-layer/
(eessi_bot_hpcv_casparvl_venv) [casparl@int4 eessi-bot-software-layer]$
./event_handler.sh
Configuration check: PASSED
EESSI bot for software layer started!
app is listening on port 3000
logging in to
/home/casparl/HPCV_Software/bot-instance/eessi-bot-software-layer/eessi_bot_even
t_handler.log
```

All prep work done! Let's start…

EESSI build bot: start the three bot processes

- Job manager

```
[casparl@int4 bot-instance]$ source load_eessi_bot_env.sh
(eessi_bot_hpcv_casparvl_venv) [casparl@int4 bot-instance]$ cd
eessi-bot-software-layer/
(eessi_bot_hpcv_casparvl_venv) [casparl@int4 eessi-bot-software-layer]$
./job_manager.sh
Configuration check: PASSED
job manager just started, logging to
'/home/casparl/HPCV_Software/bot-instance/eessi-bot-software-layer/eessi_bot_job
_manager.log', processing job ids ''
```

All prep work done! Let's start...

Trigger a build on my PR that adds `Biopython-1.86-gfbf-2025b.eb` to `/cvmfs/software.caspar.nl`

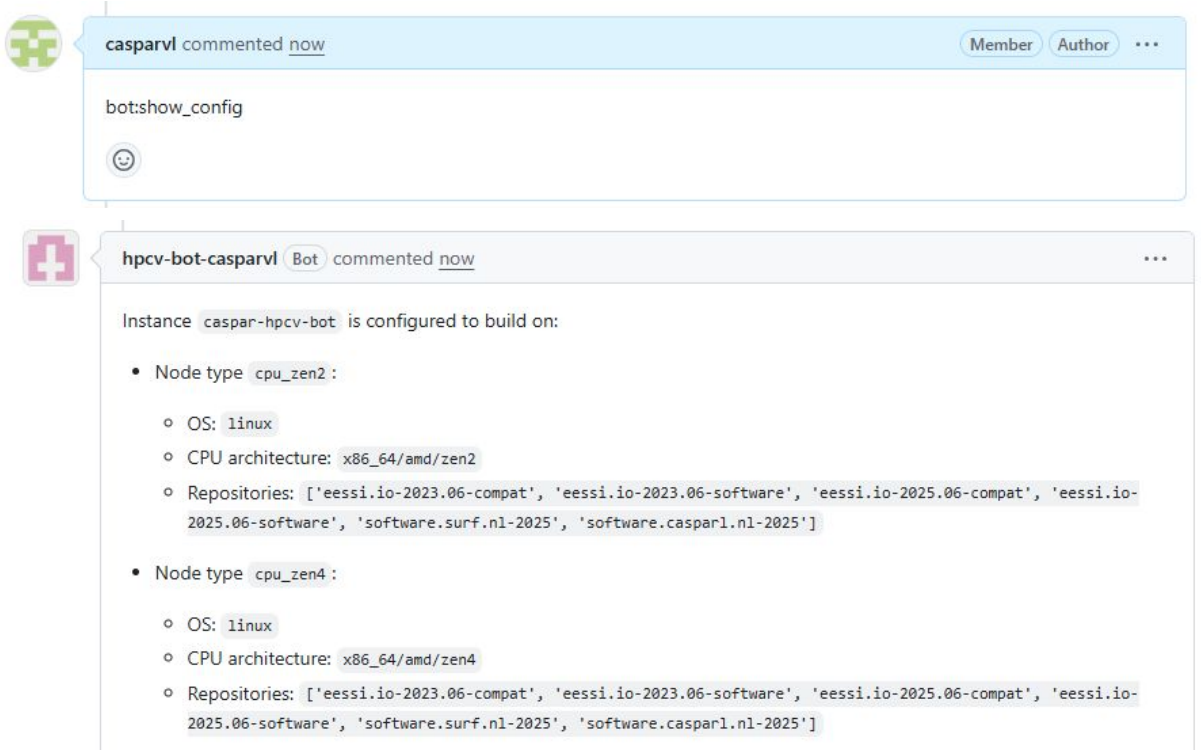
(https://github.com/SURF-hpcv/software-layer-demo/pull/2)

easystacks/software.caspar.nl/2025.06/eessi-2025.06-eb-5.3.0-2025b.yml

```
...   @@ -0,0 +1,2 @@
1   + easyconfigs:
2   +   - Biopython-1.86-gfbf-2025b.eb
```

All prep work done! Let's start...

First, check that the bot responds / is configured correctly



casparvl commented now — Member, Author

bot:show_config

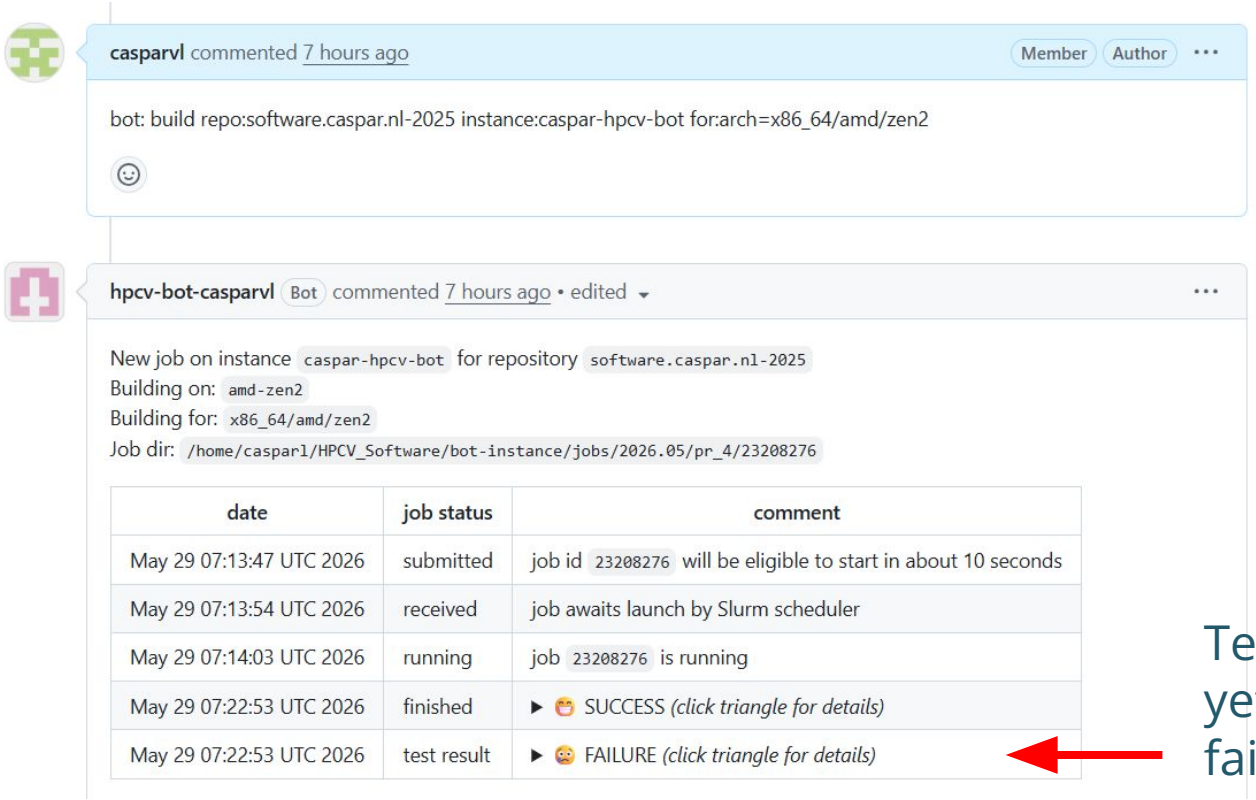
hpcv-bot-casparvl Bot commented now

Instance `caspar-hpcv-bot` is configured to build on:

- Node type `cpu_zen2`:
  - OS: `linux`
  - CPU architecture: `x86_64/amd/zen2`
  - Repositories: `['eessi.io-2023.06-compat', 'eessi.io-2023.06-software', 'eessi.io-2025.06-compat', 'eessi.io-2025.06-software', 'software.surf.nl-2025', 'software.casparl.nl-2025']`
- Node type `cpu_zen4`:
  - OS: `linux`
  - CPU architecture: `x86_64/amd/zen4`
  - Repositories: `['eessi.io-2023.06-compat', 'eessi.io-2023.06-software', 'eessi.io-2025.06-compat', 'eessi.io-2025.06-software', 'software.surf.nl-2025', 'software.casparl.nl-2025']`

All prep work done! Let's start...

Trigger a build (instance name = `app_name` as configured in `app.cfg`)



casparvl commented 7 hours ago — Member — Author

bot: build repo:software.caspar.nl-2025 instance:caspar-hpcv-bot for:arch=x86_64/amd/zen2

hpcv-bot-casparvl Bot commented 7 hours ago • edited

New job on instance `caspar-hpcv-bot` for repository `software.caspar.nl-2025`
Building on: `amd-zen2`
Building for: `x86_64/amd/zen2`
Job dir: `/home/casparl/HPCV_Software/bot-instance/jobs/2026.05/pr_4/23208276`

date	job status	comment
May 29 07:13:47 UTC 2026	submitted	job id `23208276` will be eligible to start in about 10 seconds
May 29 07:13:54 UTC 2026	received	job awaits launch by Slurm scheduler
May 29 07:14:03 UTC 2026	running	job `23208276` is running
May 29 07:22:53 UTC 2026	finished	▶ SUCCESS *(click triangle for details)*
May 29 07:22:53 UTC 2026	test result	▶ FAILURE *(click triangle for details)*

Test step doesn't work yet for site builds, ignore failure for now

All prep work done! Let's start...

Deploy:

casparvl added the bot:deploy label 7 minutes ago

May 28 10:57:45 UTC 2026	finished	▶ SUCCESS *(click triangle for details)*
May 28 10:57:45 UTC 2026	test result	▶ FAILURE *(click triangle for details)*
May 28 11:38:14 UTC 2026	uploaded	transfer of `eessi-2025.06-software-linux-x86_64-amd-zen2-17799658480.tar.zst` to S3 bucket succeeded

All prep work done! Let's start…

On the CVMFS Stratum 0, run our automated ingestion script:

```
cvanleeuwe@cvmfss0:~$ ./ingest_tarballs.sh
Found 1 tarball(s) to process.
=== Processing
eessi-2025.06-software-linux-x86_64-amd-zen2-17799658480.tar.zst ===
Downloading tarball...
…
Ingesting tarball
/data/cvmfs_s0_storage/staged_tarballs/eessi-2025.06-software-linux-x86_6
4-amd-zen2-17799658480.tar.zst to software.caspar.nl...
…
Generating the nested catalogs…
…
Remounting newly created repository revision
Nested catalogs for software.caspar.nl have been created!
Ingest succeeded for
eessi-2025.06-software-linux-x86_64-amd-zen2-17799658480.tar.zst.
```

Aaand action!

Let's try to use our new module from the client

```
cvanleeuwe@debugcvmfs:~$ export
EESSI_SITE_SOFTWARE_PREFIX=/cvmfs/software.caspar.nl
cvanleeuwe@debugcvmfs:~$ source
/cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash
Module for EESSI/2025.06 loaded successfully
EESSI has selected x86_64/amd/zen2 as the compatible CPU target for
EESSI/2025.06
EESSI did not identify an accelerator on the system
(for debug information when loading the EESSI module, set the environment
variable EESSI_MODULE_DEBUG_INIT)
```

Aaand action!

Let's try to use our new module from the client

```
{EESSI/2025.06} cvanleeuwe@debugcvmfs:~$  module load Biopython/1.86-gfbf-2025b
{EESSI/2025.06} cvanleeuwe@debugcvmfs:~$  python -c 'import Bio.motifs; print(Bio.motifs.__path__)'
['/cvmfs/software.caspar.nl/versions/2025.06/software/linux/x86_64/amd/zen2/software/Biopython/1.86-gfbf-2025b/lib/python3.13/site-packages/Bio/motifs']
{EESSI/2025.06} cvanleeuwe@debugcvmfs:~$ which python
/cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen2/software/Python/3.13.5-GCCcore-14.3.0/bin/python
```

The goal, revisited

- Build using dependencies from EESSI… ✅
- in a CVMFS repository of our own… ✅
- with the EESSI build bot… ✅
- using the same build scripts EESSI uses (from software-layer-scripts)
  - Almost, these changes are in the process of being upstreamed
    https://github.com/EESSI/software-layer-scripts/pull/239 🕐

Special cases: licensed software

- Licensed software that all of your users are allowed to access
  - Use appropriate firewall settings for your Stratum servers
- Licensed software that only a group of your users are allowed to use
  - Restricting access to subset of users by setting `CVMFS_CLAIM_OWNERSHIP=no` (preserves ownership) provides *some* security (but can be overridden from e.g. a container...)
  - Alternatively, you could still do these on a shared filesystem
- Installations could still require sites/users to point to a license file/server
- Often shipped as binaries: may require patching with patchelf

Special cases: software rebuilds

- Put your easystack in a `rebuilds` subdirectory `<gitrepo>/easystacks/software.caspar.nl/<EESSI_VERSION>/rebuilds/`
  - This triggers the EESSI build scripts to pass `--rebuild` to EasyBuild
- Trigger your build as usual

Special cases: removing software

- Start a transaction & remove software:
  - `sudo cvmfs_server transaction <repo>`
  - `rm -r /cvmfs/<repo_name>/versions/2025.06/software/<os>/<arch>/<vendor>/<micro-arch>/software/<software_name>/<software_version>`
  - `rm -r /cvmfs/<repo_name>/versions/2025.06/software/<os>/<arch>/<vendor>/<micro-arch>/module/all/<software_name>/<software_version>.lua`
- Make sure to update the caches
  - filesystem-layer/scripts/update_lmod_caches.sh `/cvmfs/<repo_name>/versions/2025.06 /cvmfs/software.eessi.io/versions/2023.06/compat/linux/x86_64/usr/share/Lmod/libexec/update_lmod_system_cache_files`

Use the oldest version for max compatibility

- Finish transaction
  - `sudo cvmfs_server publish -m "Remove software XYZ and update caches"`

Site configuration: relevant EESSI variables

- `EESSI_SITE_SOFTWARE_PREFIX`
  - Ensures that the EESSI module will find your site installations
- `EESSI_MODULE_STICKY`
  - Makes the EESSI module sticky if set to 1
- `EESSI_MODULE_FAMILY_NAME`
  - Assigns a Lmod family name to the EESSI module
  - Useful if you offer different (incompatible) software stacks

- For Lmod itself, see https://lmod.readthedocs.io/en/latest/090_configuring_lmod.html

Site configuration using EESSI's Lmod

- Create a file in `/etc/profile.d` that does the following:
  - Set the path to your site software prefix:

    export `EESSI_SITE_SOFTWARE_PREFIX=/cvmfs/software.caspar.nl`
  - Source the EESSI init script:

    `/cvmfs/software.eessi.io/versions/2025.06/init/lmod/bash`
  - Adjust the **Lmod** configuration to your liking

Site configuration using site's Lmod

- Set of scripts in `/etc/profile.d`
  - `00-modulepath.sh`: installed by Lmod package
  - `01-local_lmod.sh`: site-specific Lmod environment variables, set `$MODULEPATH`
  - `z01_StdEnv.sh`: define the standard environment/module (see Lmod documentation)
- Use your own `StdEnv` meta module for setting up the environment

Demo: Hábrók site configuration

- Using our own Lmod installation and /software from the first demo

```
[bob@merel1 ~]$ cat /etc/profile.d/01-local_lmod.sh
# Only run these settings once
if [ -z "$__Init_Habrok_Modules" ]; then
    export __Init_Demo_Modules=1

    export EESSI_SITE_SOFTWARE_PREFIX=/software
    export MODULEPATH=/cvmfs/software.eessi.io/init/modules
    export LMOD_CASE_INDEPENDENT_SORTING="yes"
    export LMOD_PAGER="None"
    export LMOD_AVAIL_EXTENSIONS="no"
    export LMOD_DISABLE_SAME_NAME_AUTOSWAP="yes"
    # Use cache for load operations
    export LMOD_CACHED_LOADS=yes
    # Prevent user caches from being generated
    export LMOD_SHORT_TIME=86400
fi
```

Demo: Hábrók site configuration

- Using our own Lmod installation and /software from the first demo

```
[bob@merel1 ~]$ cat /etc/profile.d/z01_StdEnv.sh
# https://lmod.readthedocs.io/en/latest/070_standard_modules.html
if [ -z "$__Init_Default_Modules" ]; then
   export __Init_Default_Modules=1;

   ## ability to predefine elsewhere the default list
   LMOD_SYSTEM_DEFAULT_MODULES=${LMOD_SYSTEM_DEFAULT_MODULES:-"EESSI/2025.06"}
   export LMOD_SYSTEM_DEFAULT_MODULES
   module --initial_load --no_redirect restore
else
   module refresh
fi
```

Demo: Hábrók site configuration

- Now we log in as a regular user:

```
Last login: Fri May 29 20:02:52 CEST 2026 on pts/4
Module for EESSI/2025.06 loaded successfully

[p123456@merel1 ~]$ module avail

---- /software/versions/2025.06/software/linux/x86_64/amd/zen3/modules/all ----
   attr/2.5.2-GCCcore-14.3.0    cowsay/3.04

----
/cvmfs/software.eessi.io/versions/2025.06/software/linux/x86_64/amd/zen3/modules
/all ----
   Abseil/20240722.0-GCCcore-13.3.0
   absl-py/2.1.0-GCCcore-13.3.0

...

------------- /cvmfs/software.eessi.io/init/modules -------------
   EESSI/2023.06 (D)    EESSI/2025.06 (L)
```

Future work

- Provide documentation for building a stack on top of EESSI

- EasyBuild hooks customize installations
  - Very useful for site-specific configurations
  - Example: add path to a site license file, add/change a configuration option
  - Site-specific hooks on top of the EESSI hooks not supported *yet* by EESSI-extend

- Offer more module naming schemes
  - Manually rebuilding the module files of an existing stack is tricky

Discussion items

- Need for upstream automated ingestion script?
- What is still missing or would make your life easier?
- How to deal with site installations that are also added to EESSI later on?
- How long do we keep older EESSI versions available, and how to deal with this as a site?