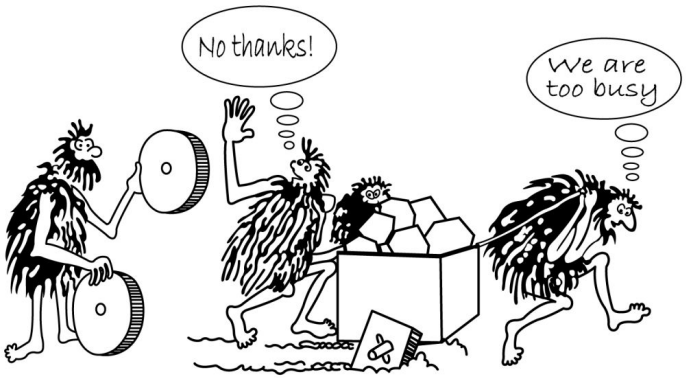
EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

Multiscale

Using EESSI as the base for a system stack

Mon 02 Jun 2025



Speakers: Bob Dröge, Pedro Santos Neves

University of Groningen (NL)

Webinar series: Different aspects of EESSI

5 Mondays in a row May-June 2025

https://eessi.io/docs/training/2025/webinar-series-2025Q2

- Introduction to EESSI - **slides+recording available!**
- Introduction to CernVM-FS - **slides+recording available!**
- Introduction to EasyBuild - **slides+recording available!**
- EESSI for CI/CD - **slides+recording available!**
- Using EESSI as the base for a system stack *(today)*

More info and registration →

Helpful knowledge

- You have watched the previous episodes in the webinar series
  - We will use **EESSI**, **EESSI-extend**, **EasyBuild**, **CVMFS**
- 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**

Agenda

- Short recap of important topics from previous webinars
- File system design
- **EESSI-extend**
- Build workflow + **demos**
  - Container
  - Scripts and easystacks
  - Bot
- Site configuration
- Important optimizations and special cases (GPU builds, licensed software, rebuilds)

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**

EESSI
EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

https://eessi.io

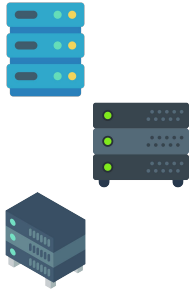
https://eessi.io/docs

Major goals of EESSI

- **Avoid duplicate work** (for researchers, HPC support teams, sysadmins, ...)
  - Tools that automate software installation process (EasyBuild, Spack) are not sufficient anymore
  - Go beyond sharing build recipes => work towards a shared software stack
- Providing a truly **uniform software stack**
  - Use the (exact) same software environment everywhere
  - **Without sacrificing performance** for “mobility of compute” (like is typically done with containers/conda)
- Facilitate HPC training, development of (scientific) software, ...

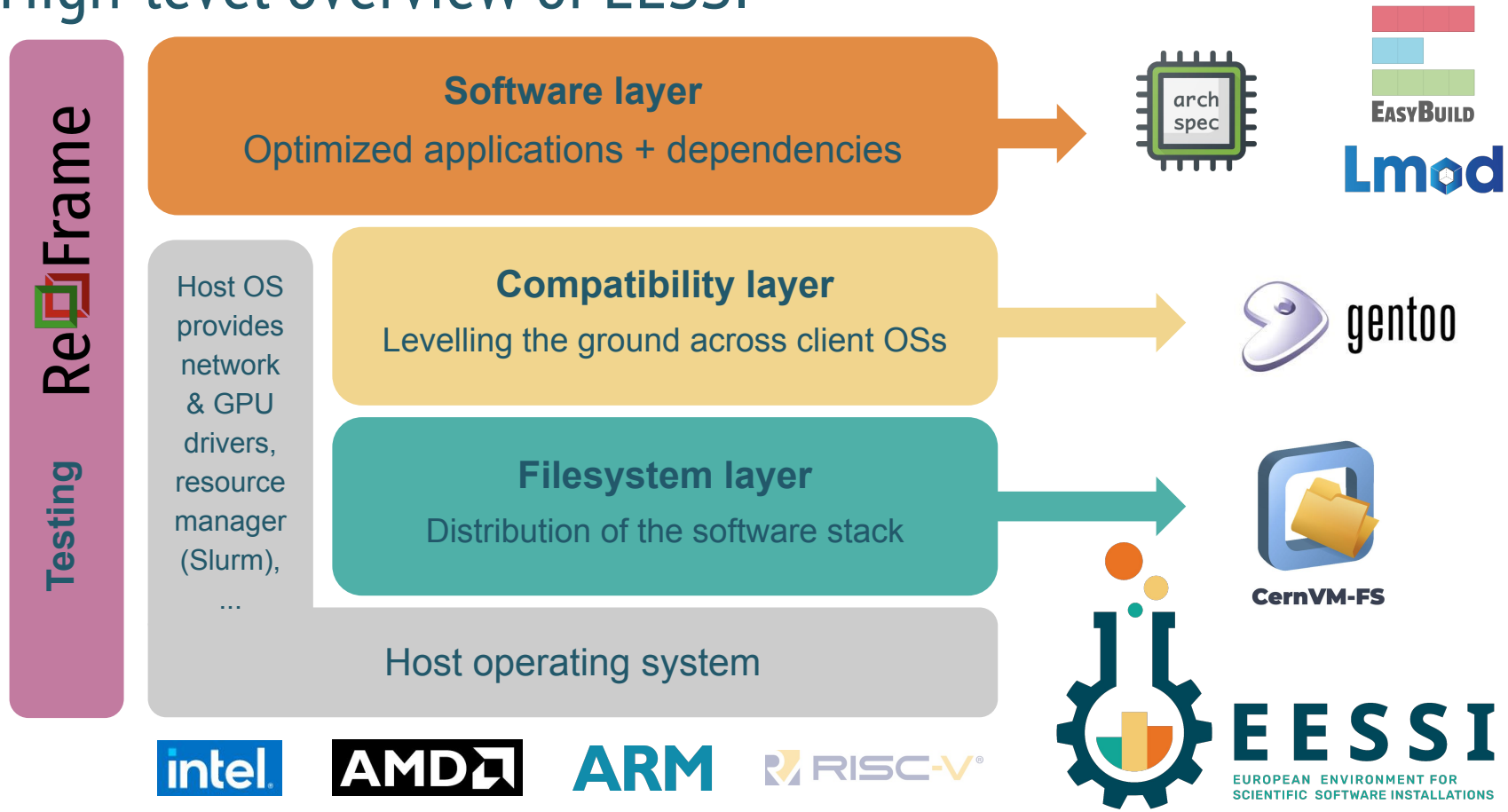
Motivation for this session

- 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

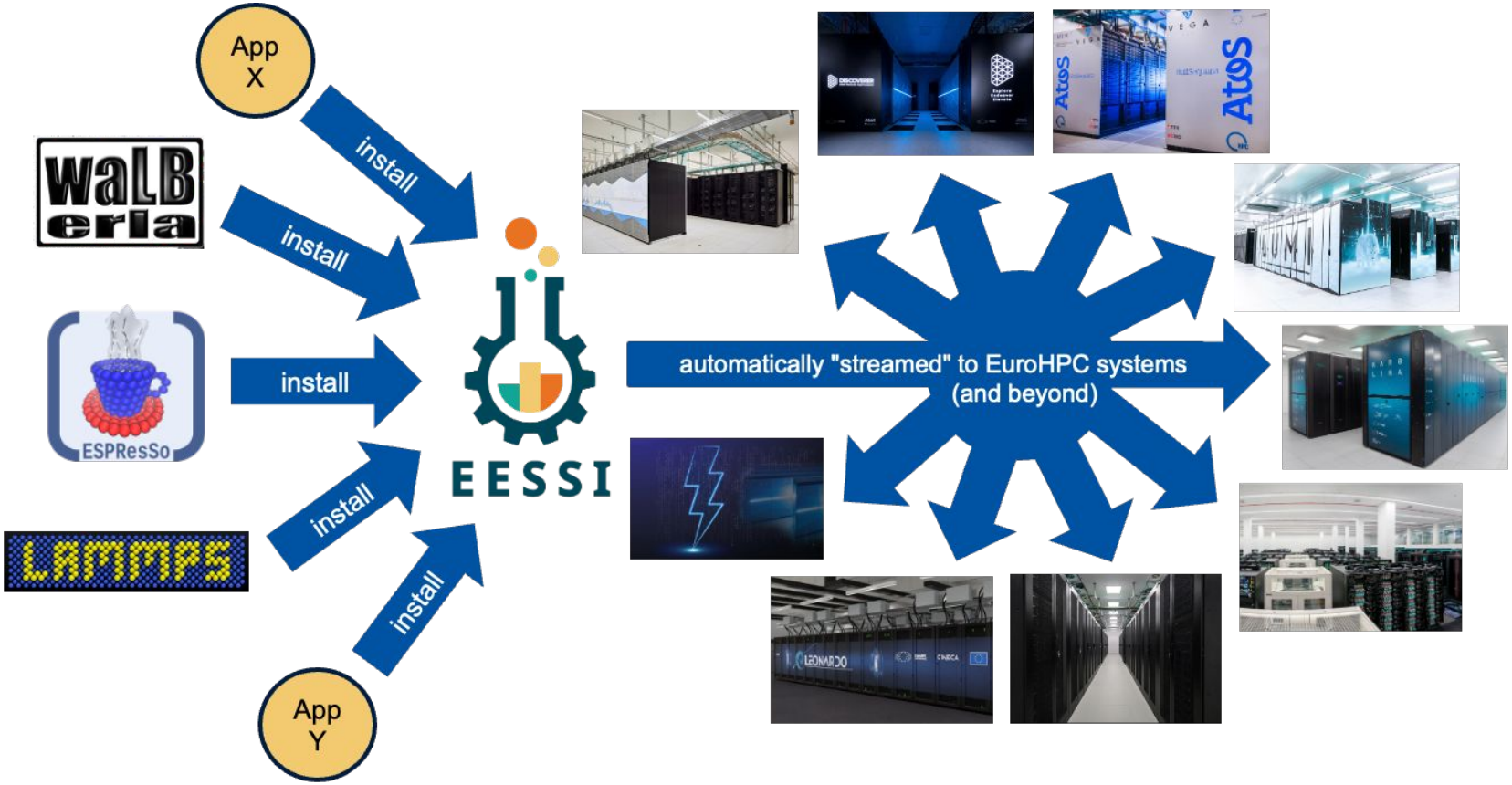


(icons via https://www.flaticon.com/authors/smashicons)

High-level overview of EESSI



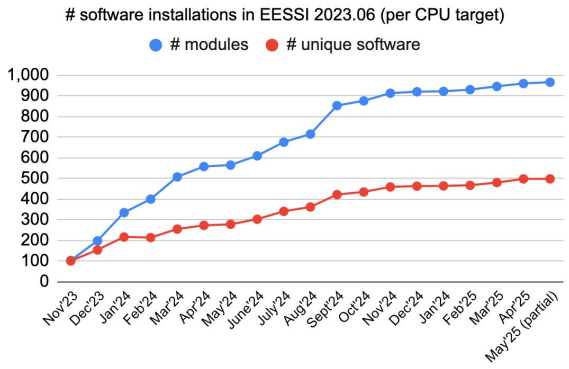
EESSI as a shared software stack

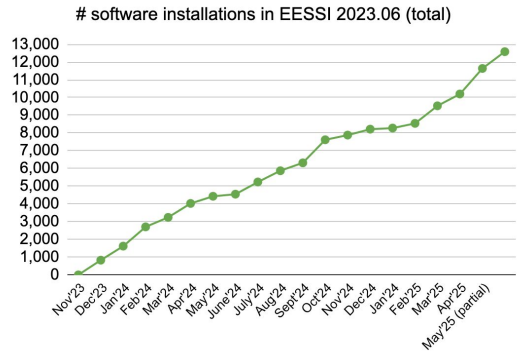


Overview of available software

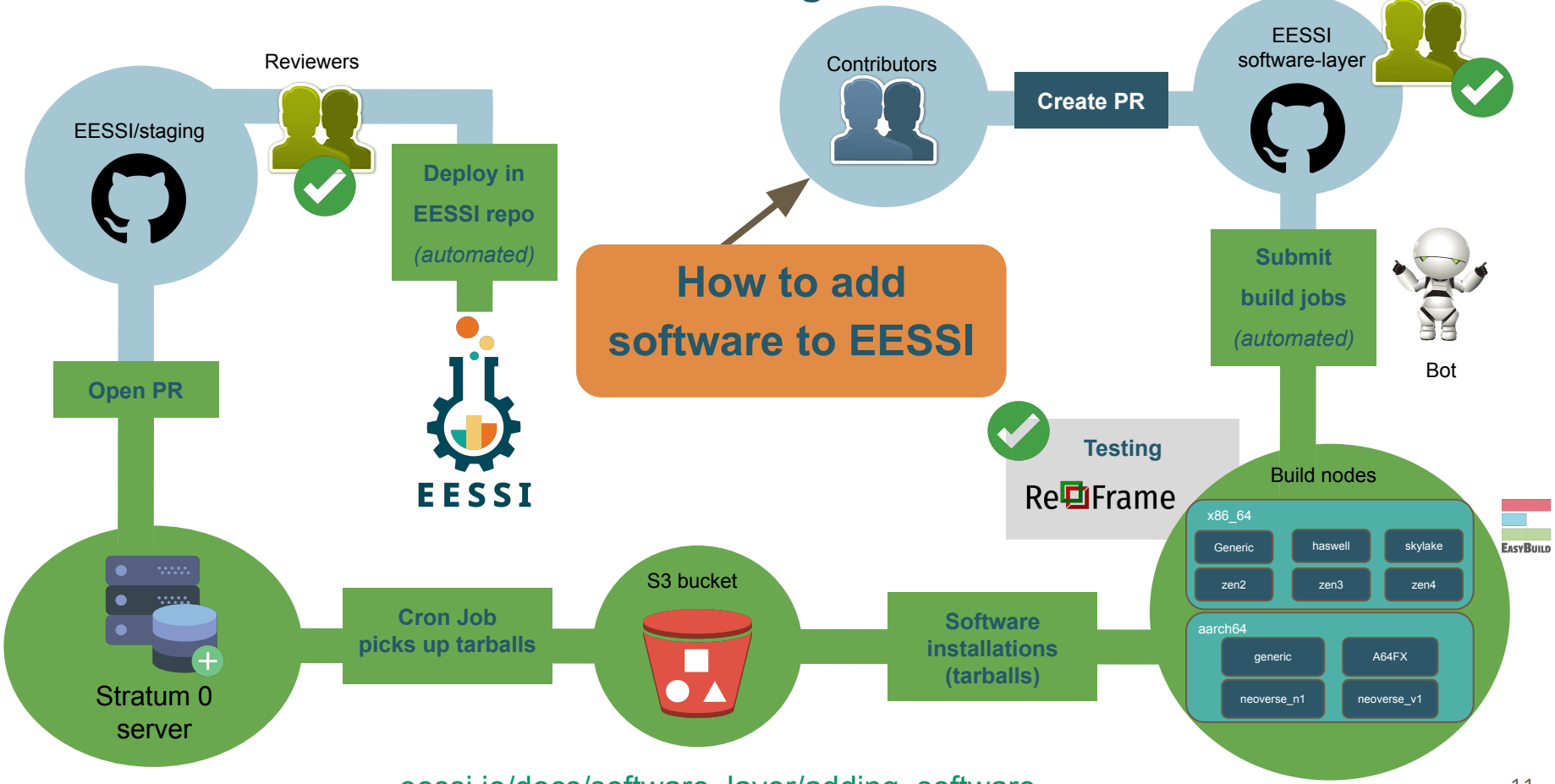
Currently ~900 software software installations available per CPU target via software.eessi.io CernVM-FS repository; increasing every day

- Over 450 different software packages
- Excl. extensions: Python packages, R libraries
- Including ESPResSo, GROMACS, LAMMPS, OpenFOAM, PyTorch, R, QuantumESPRESSO, TensorFlow, waLBerla, WRF, …
- eessi.io/docs/available_software/overview
- Using recent compiler toolchains: currently focusing on `foss/2023a` and `foss/2023b`



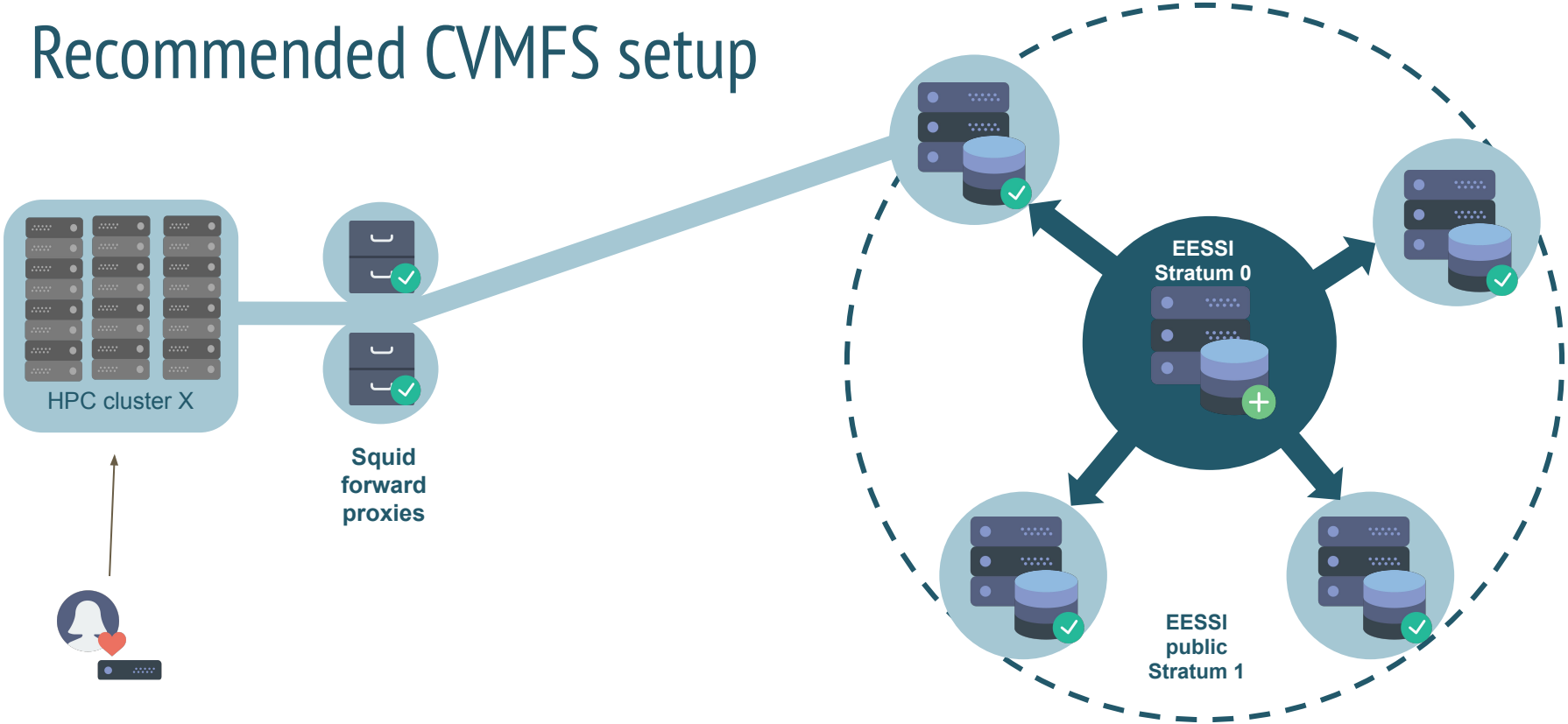


Semi-automated workflow for adding software to EESSI



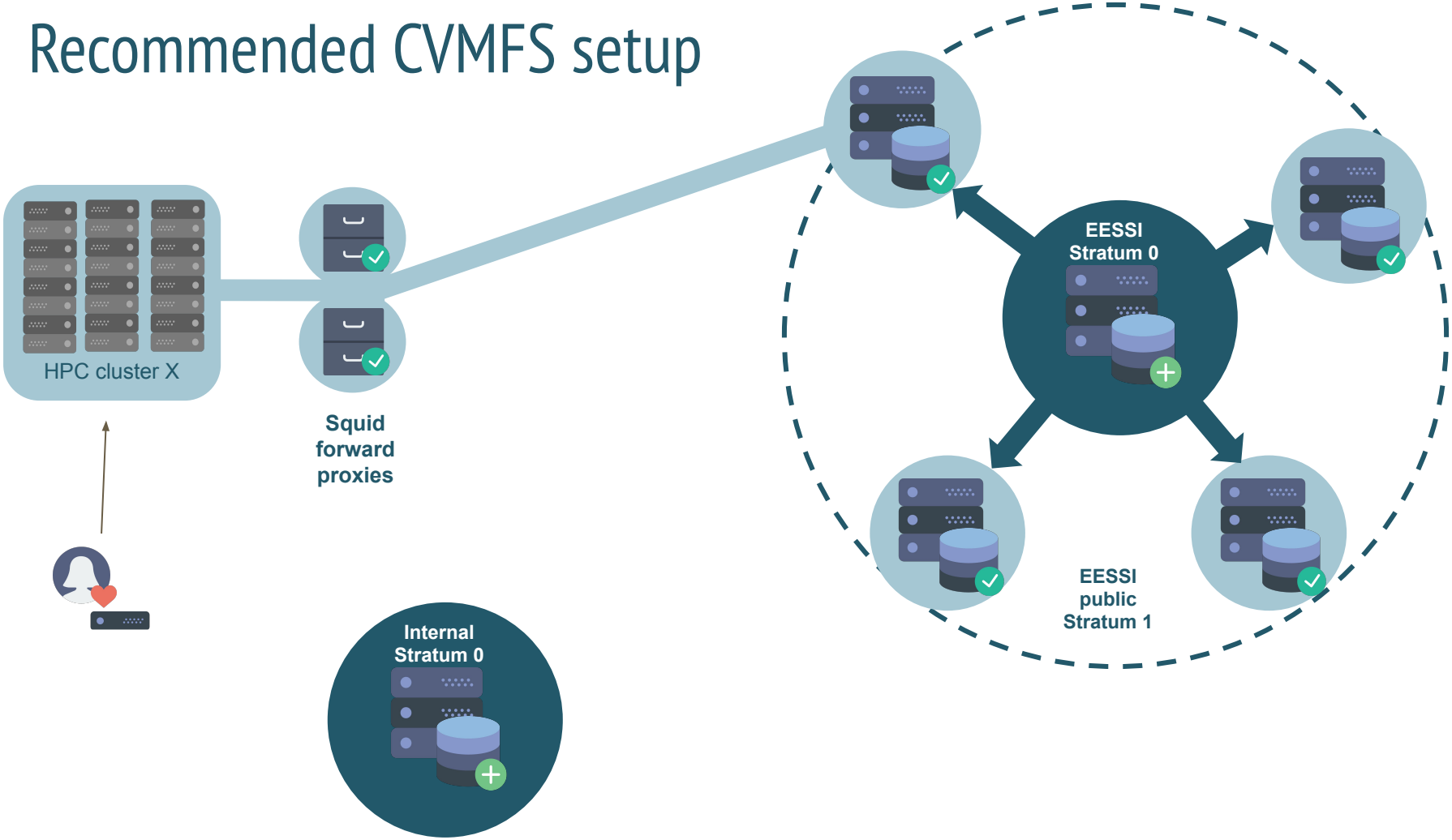
eessi.io/docs/software_layer/adding_software

Recommended CVMFS setup



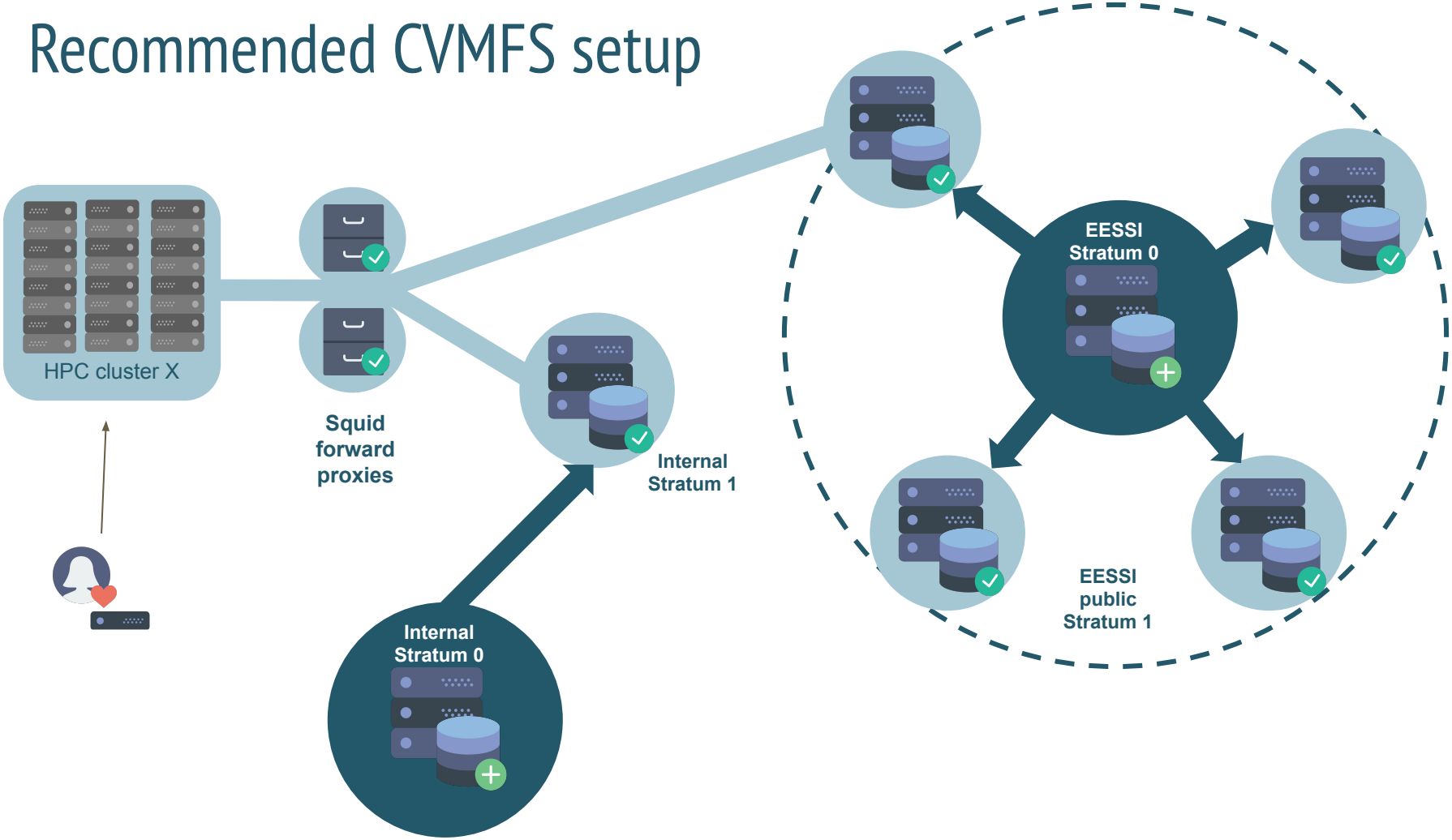
(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup



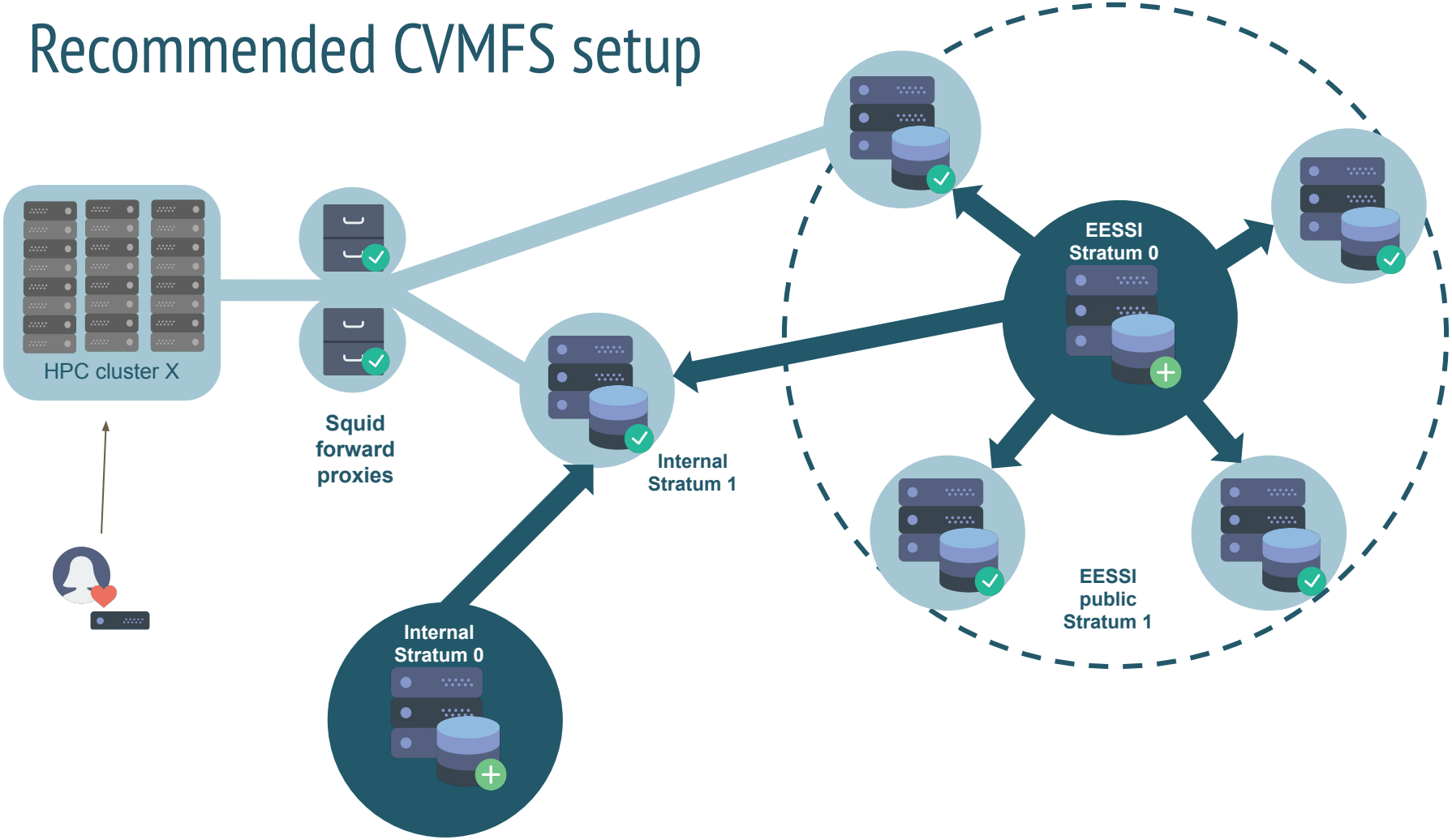
(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup



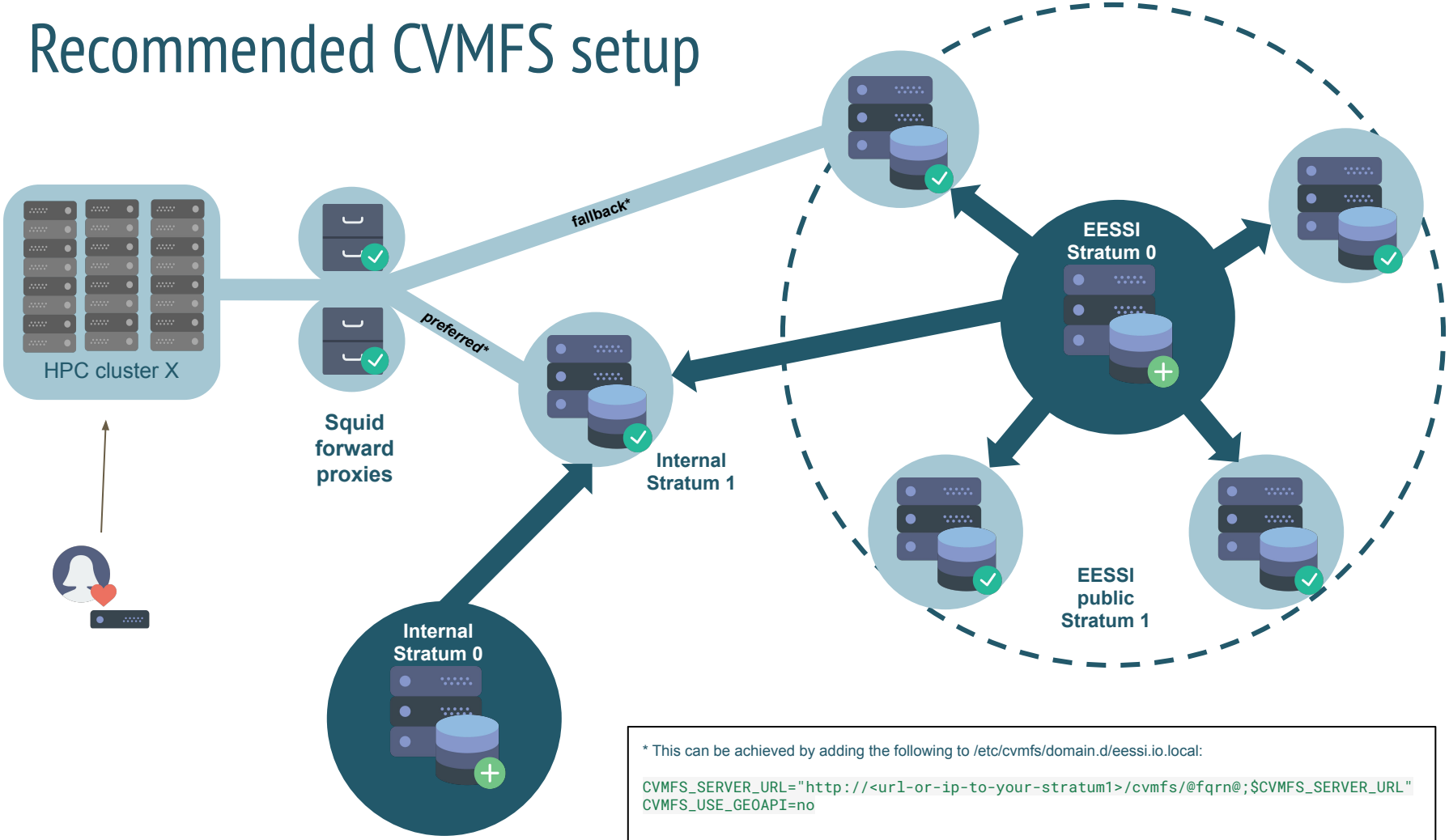
(icons via https://www.flaticon.com/authors/smashicons)

Recommended CVMFS setup



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

Recommended CVMFS setup

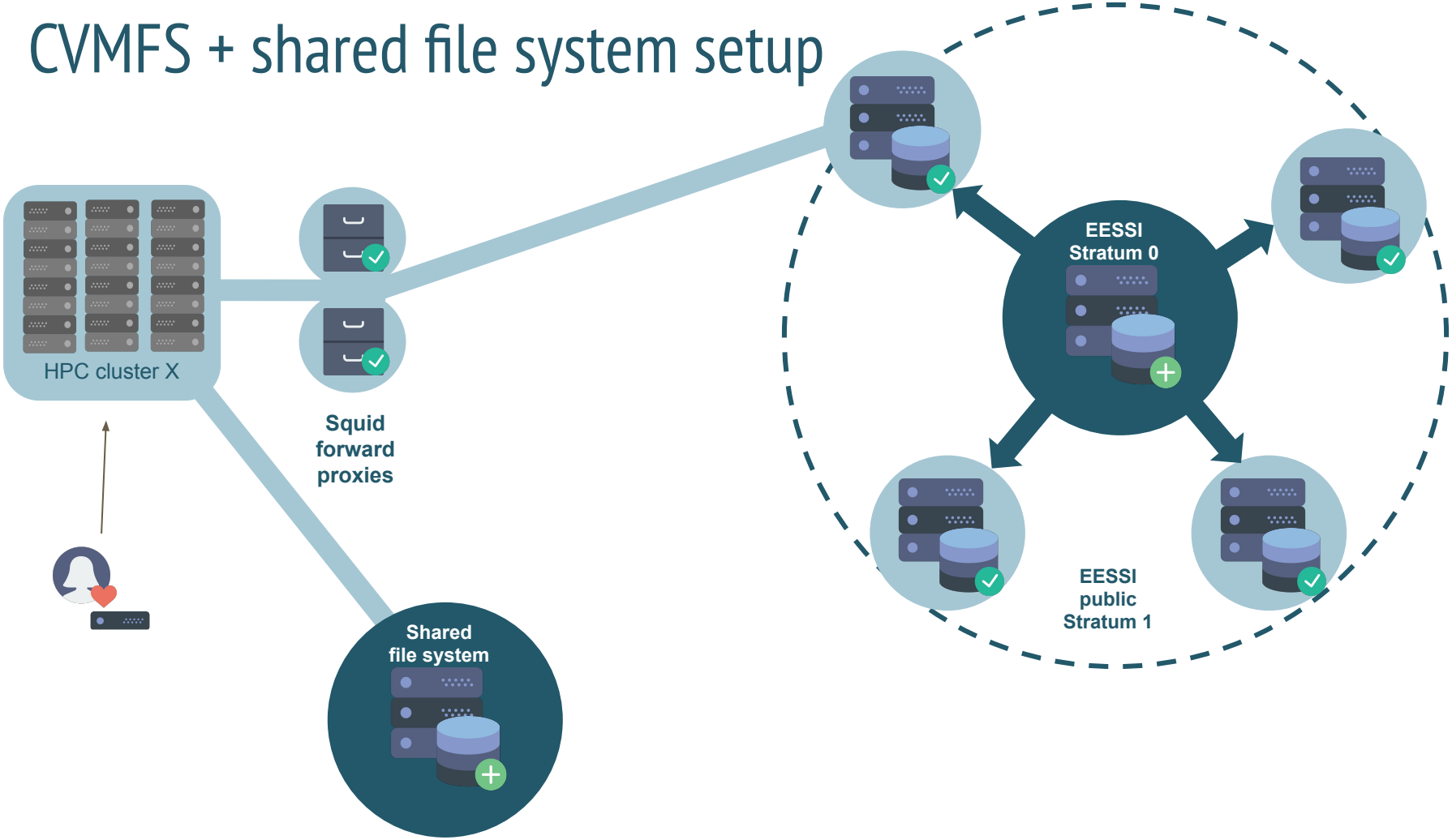


(icons via https://www.flaticon.com/authors/smashicons)

* This can be achieved by adding the following to /etc/cvmfs/domain.d/eessi.io.local:

```
CVMFS_SERVER_URL="http://<url-or-ip-to-your-stratum1>/cvmfs/@fqrn@;$CVMFS_SERVER_URL"
CVMFS_USE_GEOAPI=no
```

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`
  - `module load EESSI-extend`
- Installations will go into `/cvmfs/software.eessi.io/host_injections/<version>/…`
  - This path is automatically added to `$MODULEPATH`, so new installations are found
  - Variant symlink, default value: `/opt/eessi`
  - **Make sure you point the symlink to your software stack in your CVMFS config!**
    `EESSI_HOST_INJECTIONS=/cvmfs/my.repo.tld`
  - See https://www.eessi.io/docs/site_specific_config/host_injections/

Demo scenario

Starting from a barebones virtual machine

- CernVM-FS can be installed, EESSI is available, empty local stack
- Requires admin rights (`sudo` to install and configure CernVM-FS)
- Using the EESSI **build container**
- **EESSI-extend** for local stack
- **OS:** Rocky Linux 9.5
- **CPU architecture:** `x86/amd/zen2`

Demo 1: EESSI-extend & shared filesystem

- Cheatsheet: https://hackmd.io/@Ab_EvqFWSKm7fYsHfgWWkA/S1G27yPMge
- Documentation: https://www.eessi.io/docs/using_eessi/building_on_eessi/

Build container

- Host isolation, very minimal container
  - Prevent builds from picking up system libraries
- Fixed/predictable build environment
- Use any machine as build host, without requiring special privileges
- FUSE can be used to provide writable overlays
  - Allows you to install to `/cvmfs` (or another read-only location)

EESSI build container

- Built as Docker image
  - Dockerfiles: https://github.com/EESSI/filesystem-layer/tree/main/containers
  - Run with Apptainer/Singularity
- Based on Debian 11
- Script to run the container
  - https://github.com/EESSI/software-layer/blob/2023.06-software.eessi.io/eessi_container.sh
  - Many parameters, e.g. for bind mounting additional paths
  - Configuration file can be used for mounting (read-only or read-write) additional CVMFS repositories

EESSI build container: technical details

- Read-only CVMFS repositories are directly mounted under /cvmfs
  - `--fusemount "container:cvmfs2 software.eessi.io /cvmfs/software.eessi.io"`
- Writable repositories are mounted under `/cvmfs_ro`, and then made writable by adding a writable overlay using fuse-overlayfs
  - `--fusemount "container:cvmfs2 demo.eessi.eu /cvmfs_ro/demo.eessi.eu"`
  - `--fusemount "container:fuse-overlayfs -o lowerdir=/cvmfs_ro/demo.eessi.eu -o upperdir=/tmp/demo.eessi.eu/overlay-upper -o workdir=/tmp/demo.eessi.eu/overlay-work /cvmfs/demo.eessi.eu"`
  - The upper directory is bind mounted from the host and will be used for storing changes
    - New installations will end up here
- Something similar could be done for writable shared file systems
  - Advantage: makes the build process more transactional

Add your CVMFS repository to the EESSI container

- Make a directory containing a repos.cfg

```
[webinar-demo]
repo_name = demo.eessi.eu
config_bundle = repos.tgz
config_map = { "demo.eessi.eu/demo.eessi.eu.pub":
"/etc/cvmfs/keys/eessi.eu/demo.eessi.eu.pub",
"demo.eessi.eu/eessi.eu.conf":"/etc/cvmfs/domain.d/eessi.eu.conf",
"demo.eessi.eu/default.local":"/etc/cvmfs/default.local" }
```

- Prepare the `repos.tgz` containing the files listed in your `config_map`
  - Store it in the same directory as `repos.cfg`
- Point `$EESSI_REPOS_CFG_DIR_OVERRIDE` to this directory
- Run `eessi_container.sh` with `-r webinar-demo,access=rw`

Demo 2: EESSI build container

- We use the following GitHub repositories:
  - https://github.com/EESSI/stack-on-top-of-eessi-demo
  - https://github.com/EESSI/software-layer

- Cheatsheet: https://hackmd.io/@Ab_EvqFWSKm7fYsHfgWWkA/S1G27yPMge

Start building software

- Create a job script that calls the build container with some build script
- The build script itself should perform the following steps:
  - Initialize the EESSI environment
  - Configure your environment
  - Load EESSI-extend with the right settings (`EESSI_SITE_INSTALL=1`)
  - Run EasyBuild with an easyconfig or easystack file
  - Create a tarball containing the installation (optional, but recommended)
- The EESSI scripts are quite specific to the EESSI workflow and repository
  - Can be used as a good starting point

Creating tarballs

- This step is not strictly necessary depending on installation location!
  - Still recommended, as it makes the installation more transactional
- Find software, module files, additional scripts in the overlay upper directory
- Keep/create desired directory hierarchy (matching CVMFS repository)
- Bundle files into tarball with specific name
  - The overlay upper directory is scanned for changes / new installations
  - The tarball itself is created from the real installation prefix (`/cvmfs/…`)
- Store it on a shared file system or object store

Ingesting tarballs

- Tarball can be ingested on the Stratum 0 using `gunzip -c <tarball> | sudo cvmfs_server ingest -t - -b /` demo.eessi.eu
  - `-t -` reads the tarball from `stdin`
  - `-b /` ingests it to the root of the repository, update accordingly if you used a different prefix
- You may want to use a wrapper script to do this
  - Can also do additional checks / operations (e.g. updating the Lmod cache, as shown later)

Demo 3: full build and ingestion to CVMFS

- We use the following GitHub repositories:
  - https://github.com/EESSI/stack-on-top-of-eessi-demo
  - https://github.com/EESSI/software-layer

- Cheatsheet: https://hackmd.io/@Ab_EvqFWSKm7fYsHfgWWkA/S1G27yPMge

Build bot: automated builds

- 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

Build bot: automated builds

- 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
- Future work
  - GitLab compatibility
  - Even more flexibility
  - Even more quality of life improvements

(Semi-)automated tarball ingestions

- Deploy tarballs to Stratum 0 and ingest them to the repository
  - Cron job that picks up new tarballs
  - May be wise to retain a manual element for ingestion

- GitHub/GitLab event -> Bot can be run on a login node that is also a CVMFS publisher node

Site configuration for EESSI without Lmod

- Create a file in `/etc/profile.d` that does the following:
  - Source the EESSI init script
  - Adjust the **Lmod** configuration to your liking
  - Add the path to an Lmod cache for modules built on top of EESSI

Site configuration for EESSI with 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 4: site configuration

- We use the following GitHub repositories:
  - https://github.com/EESSI/stack-on-top-of-eessi-demo
  - https://github.com/EESSI/software-layer

- Cheatsheet: https://hackmd.io/@Ab_EvqFWSKm7fYsHfgWWkA/S1G27yPMge

Important optimizations: 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 will probably be slightly different
- Alternatively, you can include an empty `.cvmfscatalog` file in the root of every software installation directory
  - Use an EasyBuild hook or do this when creating/ingesting the tarball
- CVMFS includes functionality for automatically managing nested catalogs

Important optimizations: Lmod cache for added modules

- Lmod RC file for configuring the cache can be generated using https://github.com/EESSI/software-layer/blob/2023.06-software.eessi.io/create_lmodrc.py
  - Call this at the end of your build script if it needs to be updated
- The cache itself should be updated when ingesting new software
  - Use a simple ingestion wrapper script that generates the cache and ingests the tarball
  - Loop through the directories of the different CPU targets, call `update_lmod_system_cache_files` for each of them
  - See https://github.com/EESSI/filesystem-layer/tree/main/scripts

EESSI

Multiscale

Special cases: GPU builds

- GPU drivers need to be found in the `host_injections` directory
  - See https://www.eessi.io/docs/site_specific_config/gpu/#exposing-nvidia-gpu-drivers
- Full CUDA SDK needs to be installed
  - Only runtime libraries can be redistributed
  - https://www.eessi.io/docs/site_specific_config/gpu/#installing-full-cuda-sdk-optional
- Builds currently end up in the CPU prefix
  - Will cause issues for nodes with the same CPU but different GPUs
  - Planning to enhance EESSI-extend to install them to a separate prefix

Special cases: licensed software

- Could go into a separate CVMFS repository
- Use appropriate firewall settings for your Stratum servers
- Restrict access to subset of users by setting `CVMFS_CLAIM_OWNERSHIP=no` (preserves ownership)
- Installation could still require sites to point to a license file/server

Special cases: software rebuilds

- Prevent or minimise disruption of ongoing jobs
- Transactional approach with container, overlays, tarballs make this possible
- EESSI-extend sets `$EASYBUILD_READ_ONLY_INSTALLDIR=1`
  - Protects you from accidentally writing to existing installations
  - Not possible to restore write permissions in the container
  - Workaround by using Apptainer's `--fakeroot` option
  - Not ideal, looking for better solutions

Future work

- Provide documentation for building a stack on top of EESSI
- Better procedure for software rebuilds
- 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
- Enhance EESSI-extend for GPU builds on top of EESSI
  - Install them to a subdir for the corresponding accelerator / compute capability
- Offer more module naming schemes
  - As a site you could already try rebuilding all the modules with another MNS

EESSI

EUROPEAN ENVIRONMENT FOR SCIENTIFIC SOFTWARE INSTALLATIONS

Website: eessi.io

GitHub: github.com/eessi

Documentation: eessi.io/docs

Blog: eessi.io/docs/blog

Join the EESSI Slack

YouTube channel: youtube.com/@eessi_community

Paper (open access): doi.org/10.1002/spe.3075

EESSI support portal: gitlab.com/eessi/support

Bi-monthly online meetings (1st Thu, odd months, 2pm CE(S)T)

MultiXscale

Web page: multixscale.eu

Facebook: MultiXscale

Twitter: @MultiXscale

LinkedIn: MultiXscale

Co-funded by the European Union

EuroHPC Joint Undertaking

KEMIJSKI INŠTITUT

UNIVERSITAT DE BARCELONA

Universität Stuttgart

UNIVERSITAS BERGENSIS

SORBONNE UNIVERSITÉ

Université de Toulouse

Consiglio Nazionale delle Ricerche

MAX-PLANCK-GESELLSCHAFT

HPC Now!

university of groningen

SURF

JÜLICH Forschungszentrum

GHENT UNIVERSITY

BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación

LEONARDO

iit ISTITUTO ITALIANO DI TECNOLOGIA

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and countries participating in the project under grant agreement No 101093169.