EESSI

EUROPEAN ENVIRONMENT FOR
SCIENTIFIC SOFTWARE INSTALLATIONS

EESSI - behind the scenes

January 19th 2021

https://github.com/EESSI/docs/tree/master/talks/20210119_EESSI_behind_the_scenes

Agenda

EESSI

behind the scenes

- (Very) high-level overview of EESSI
- Focus on “behind-the-scenes” aspects
- Tools & workflows used for building EESSI pilot repository
- Current status of different EESSI layers
- Goals and plans going forward

EESSI in a nutshell

- **European Environment for Scientific Software Installations** (EESSI, pronounced as “easy”)
- Collaboration between different European partners in HPC community
- Goal: building a **common scientific software stack for HPC systems & beyond**
- “Grass roots” project, fueled by a lack of time to do a proper job at installing scientific software and the desire for collaborating on something useful (+ having beers together)

Scope & goals

- **Shared repository of (optimized!) scientific software installations**
- Avoid duplicate work across HPC sites
- Uniform way of offering software to users, regardless of system they use
- Should work on any Linux OS (+ macOS) and system architecture
    - From laptops and personal workstations to HPC clusters and cloud
    - Support for different CPUs, interconnects, GPUs, etc.
- **Focus on performance, automation, testing, collaboration**

EESSI partners & interested parties

Founding partners:

TU Delft

VU University Amsterdam

university of groningen

TU/e Eindhoven University of Technology

SURF

UNIVERSITY OF TWENTE.

UNIVERSITY OF CAMBRIDGE

GHENT UNIVERSITY

DELL Technologies

Extensive interest from (HPC) community:

HPC Now!

Microsoft Azure

CSCS

CERN

aws

FRED HUTCH™

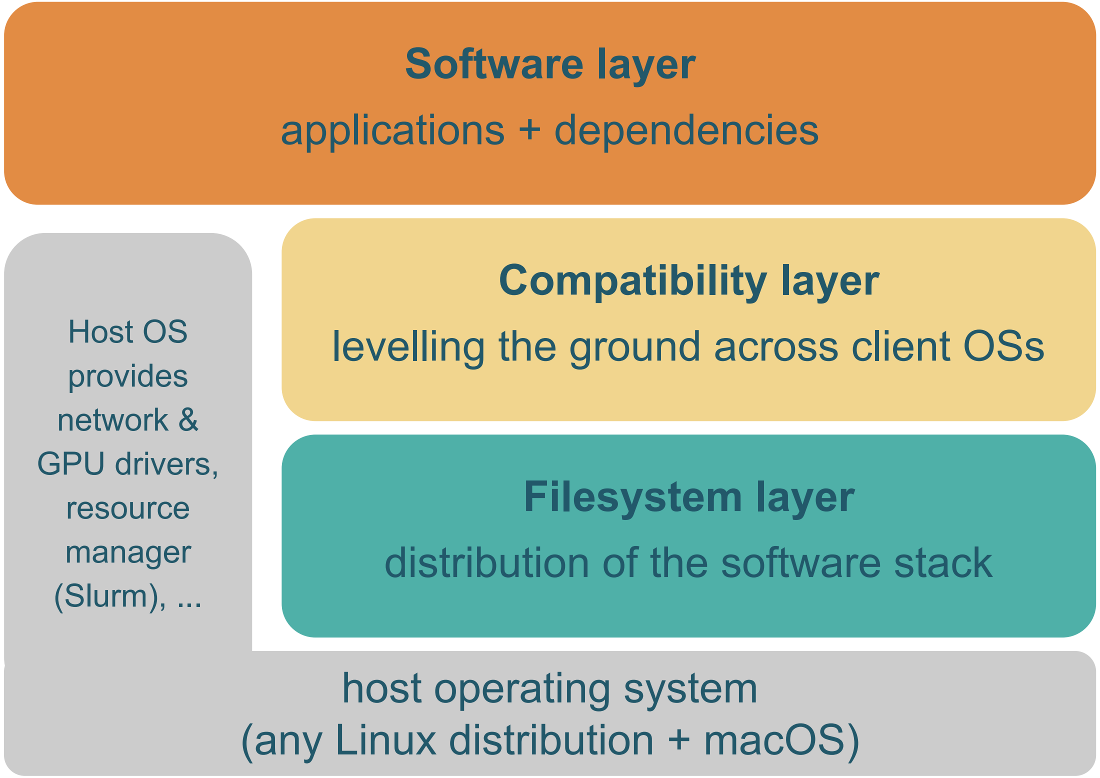
JÜLICH Forschungszentrum

UiO : University of Oslo

HPC.NRW

EESSI layers

EESSI
behind the scenes



Software layer

applications + dependencies

Compatibility layer

levelling the ground across client OSs

Filesystem layer

distribution of the software stack

Host OS provides network & GPU drivers, resource manager (Slurm), ...

host operating system
(any Linux distribution + macOS)

Heavily inspired by

compute canada

software stack

EESSI is powered by FOSS (1/2)

easybuild

- **Installation tool for scientific software**
- Optimises for build host (by default)
- Supports over 2,000 different pkgs

https://easybuild.io/eum

arch spec

- Python library
- **Detects processor type**
- Check compatibility with host CPU

https://github.com/archspec

Lmod

- **Environment modules tool** (in Lua)
- Intuitive access to software installations
- Multiple versions side-by-side

https://lmod.readthedocs.io

gentoo

https://wiki.gentoo.org/wiki/Project:Prefix

- Gentoo: Linux distribution, installs from source
- Prefix subproject: **install packages in <prefix>**
- Supports x86_64, Arm64, POWER, …
- Supports both Linux and macOS

CernVM-FS

https://cernvm.cern.ch/fs

- **Software distribution service** (software *installations*, not packages!)
- Scalable, read-only, globally distributed filesystem
- Served by web servers (HTTP only), no firewall issues
- Originally build for Large Hadron Collider (LHC) project at CERN

ReFrame

- Regression testing framework for HPC
- Tests are implemented as Python classes
- **Verify correctness**
- **Evaluate performance**

https://reframe-hpc.rtfd.io

EESSI is powered by FOSS (2/2)

ANSIBLE

- Tool for automation and configuration management
- Using “playbooks” (YAML)
- **Used to automate deployment of filesystem and compatibility layer**

https://www.ansible.com

- Singularity: popular container runtime for HPC
- Own container image format
- Also consumes Docker containers
- **Used to fully control build environment for compatibility and software layer + (optionally) to let clients access EESSI**

https://sylabs.io/singularity

Terraform

https://www.terraform.io

- “Infrastructure as code” tool
- Creating & managing cloud instances
- Declarative configuration files in custom DSL (HashiCorp Configuration Language - HCL)
- **Planning to use this for creating on-demand build/test nodes in AWS/Azure/...**

CLUSTER IN THE CLOUD

- CLI tool to easily create disposable Slurm clusters in the cloud
- Supports AWS, Oracle, Google cloud (Azure not yet supported)
- Leverages Ansible and Terraform in the background
- **Planning to use this to set up (heterogenous) Slurm clusters for building and testing software**

https://github.com/clusterinthecloud

Every layer solves a specific problem

- We only rely on client OS for things we can’t / shouldn’t provide ourselves
- CernVM-FS in filesystem layer is used to *distribute* software installations
- Compatibility layer levels the ground across different client OSs
  - Minimal “OS”, incl. glibc, python3, curl, binutils, make, patch, fonts, ...
- Software layer hosts scientific software + dependencies, optimized for specific CPU microarchitectures

Client operating system + CPU architecture

- We only rely on client OS for:
  - Kernel, drivers (interconnect, GPU, …), GPFS & co, scheduler (Slurm)
  - CernVM-FS (or Singularity)
- ***Not*** used from client OS: compiler, tools, libraries (incl. glibc)
- We aim to support:
  - Variety of CPU families: **x86_64, Arm 64-bit,** POWER, RISC-V
  - Client OS: **Any Linux distro (incl. WSL in Windows)** + macOS

Client operating system + CPU architecture

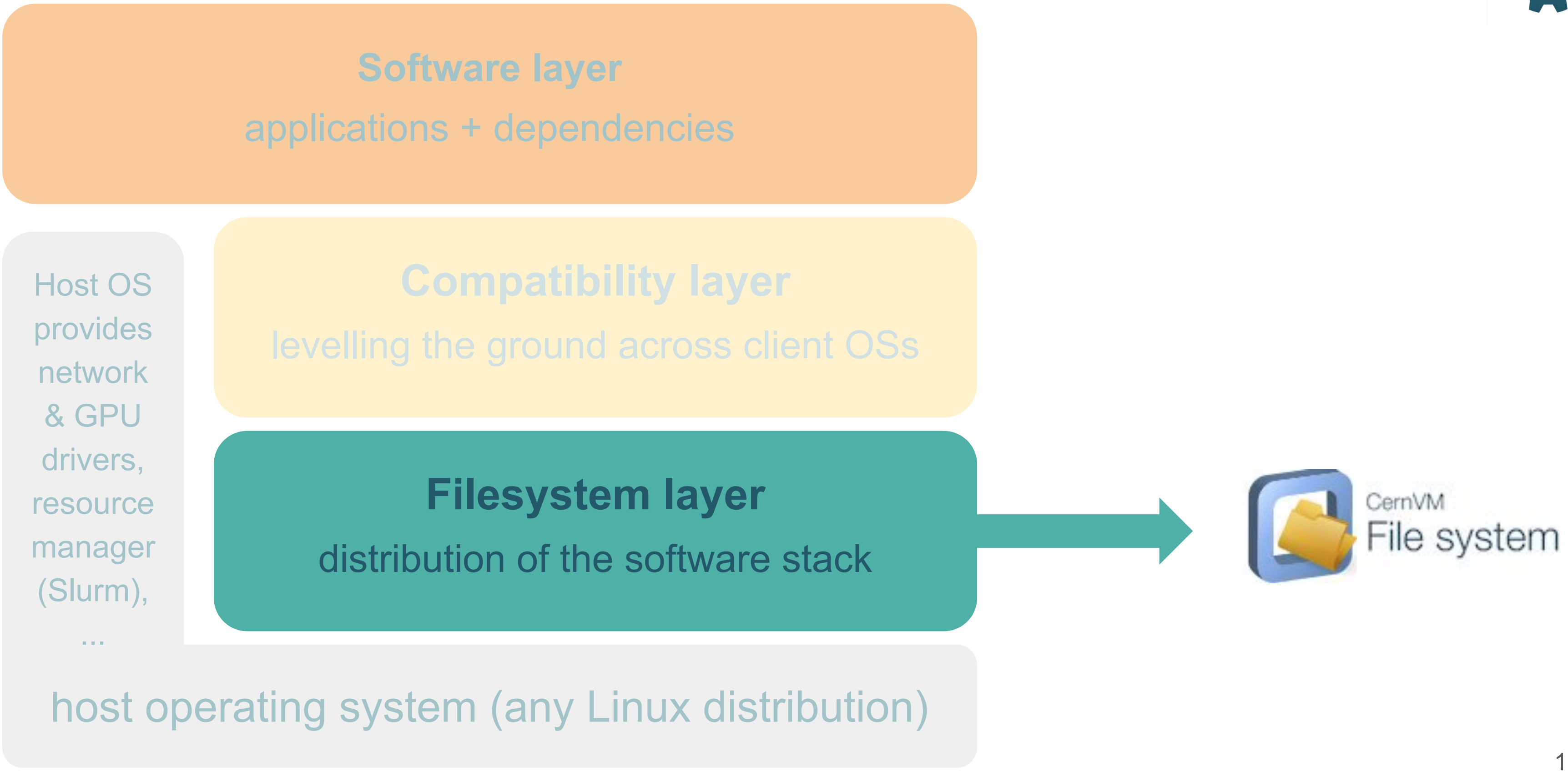
Current status

EESSI
behind the scenes

- Client operating systems
  - Tested on various Linux distributions (CentOS 7+8, Ubuntu, …)
  - Also working well in Windows Subsystem for Linux (WSL)
  - Native macOS support is WIP for both Intel + M1 (CernVM-FS works)

- CPU architectures
  - **[OK]** x86_64: `generic` (SSE2), Intel Haswell/Skylake, AMD Rome (Zen2)
  - **[OK]** Arm64: `generic` (armv8-a), Graviton2 (AWS), ThunderX2
  - **[WIP]** POWER: not working yet, problems with bootstrapping Gentoo Prefix
  - RISC-V: no (capable) hardware yet :)

EESSI filesystem layer



Software layer

applications + dependencies

Compatibility layer

levelling the ground across client OSs

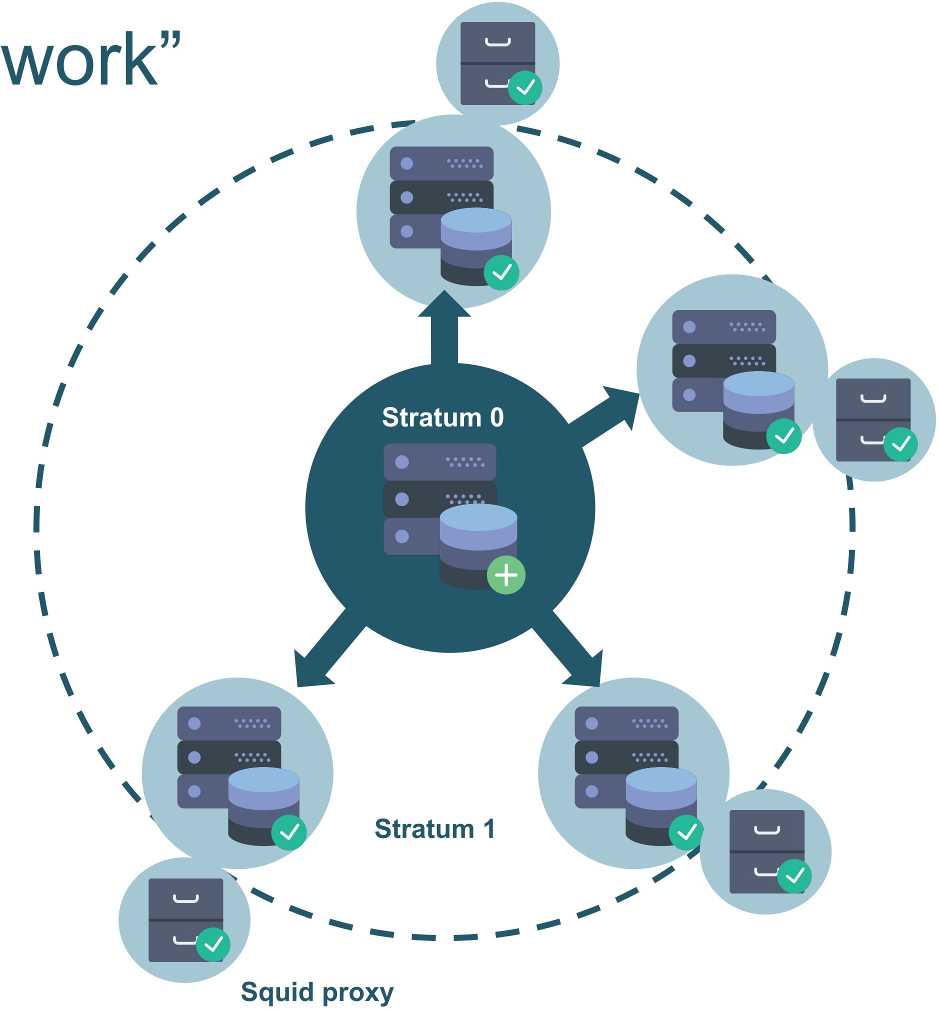
Filesystem layer

distribution of the software stack

Host OS provides network & GPU drivers, resource manager (Slurm), ...

host operating system (any Linux distribution)

EESSI “network”



CernVM File system

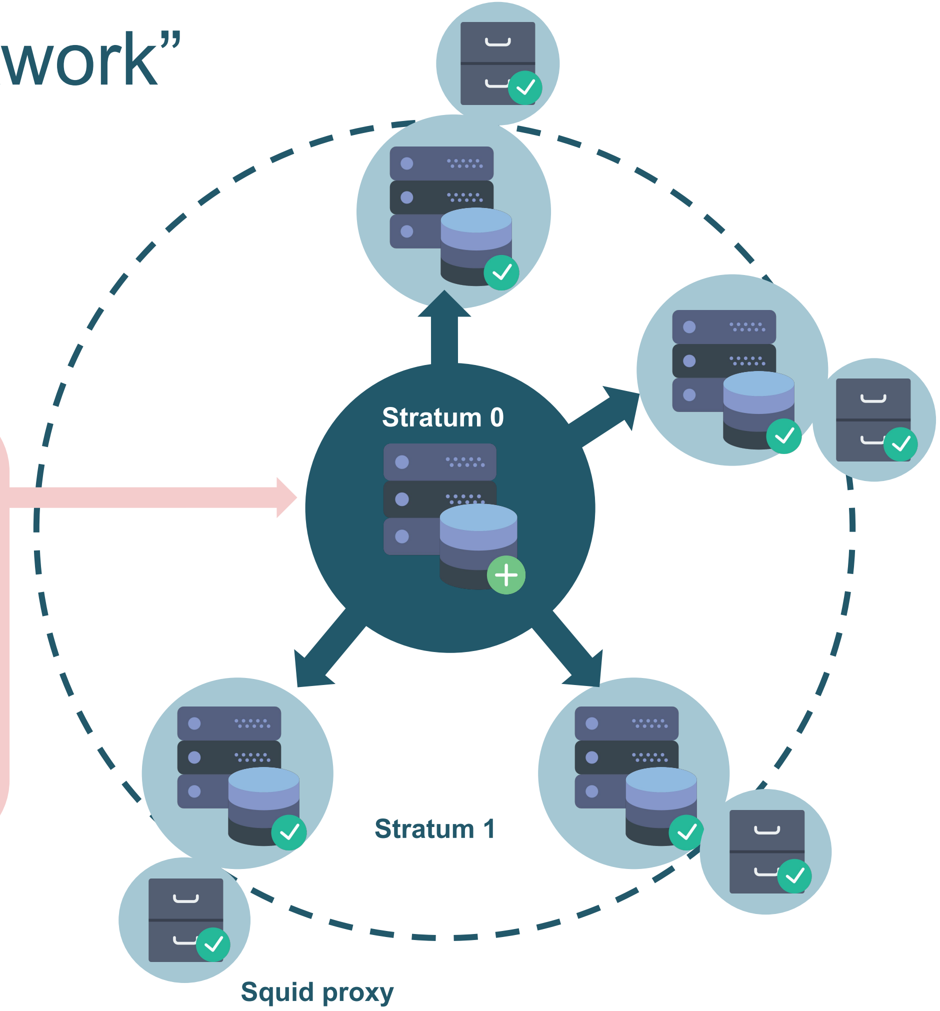
(icons via https://www.flaticon.com/authors/smashicons)

EESSI “network”

CernVM File system

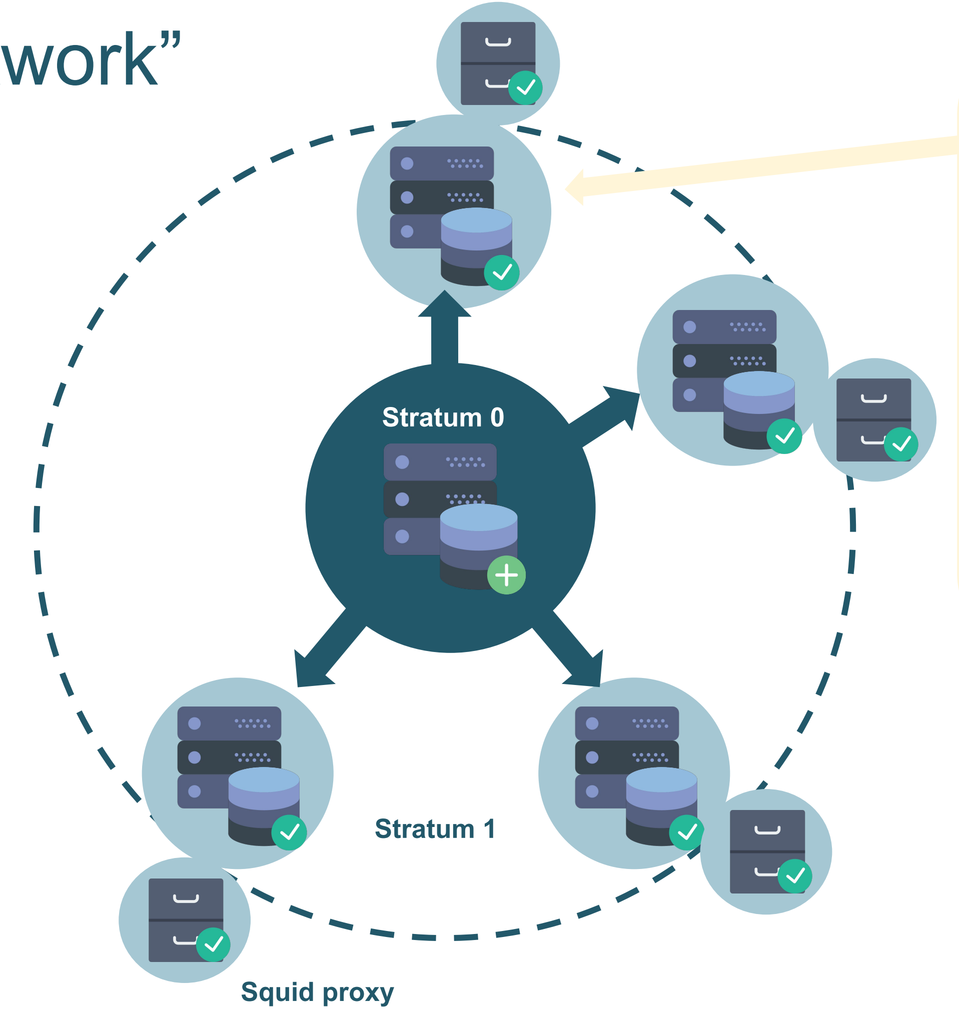
Stratum 0

- Central server
- **Only one**
- Hosts CVMFS “repositories”
- New software is added here
- Usually restricted access



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

EESSI “network”

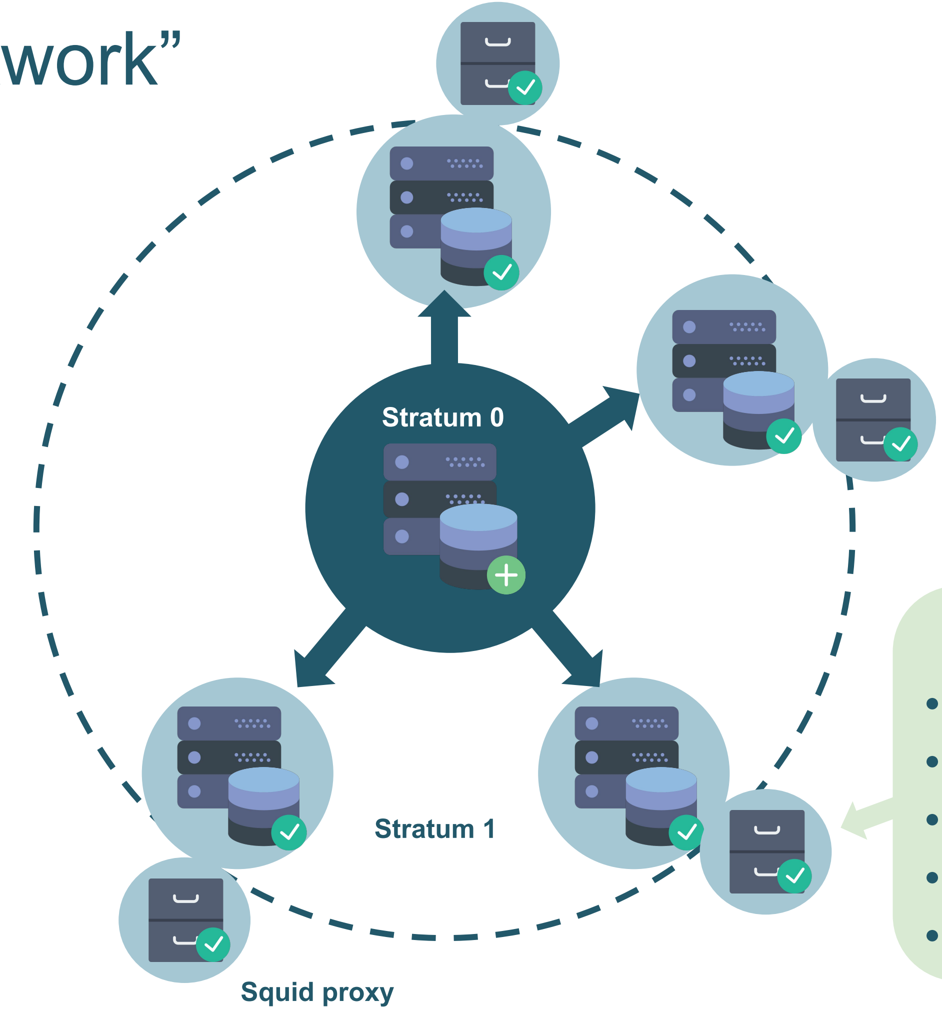


Stratum 1

- CernVM-FS “replica”
- Full copy of repositories (mirror)
- **Read-only**
- **Multiple servers**
- Geographically distributed
- Standard web server (HTTP)
- Reduce load on Stratum 0
- Improve reliability
- Clients *may* connect here

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

EESSI “network”



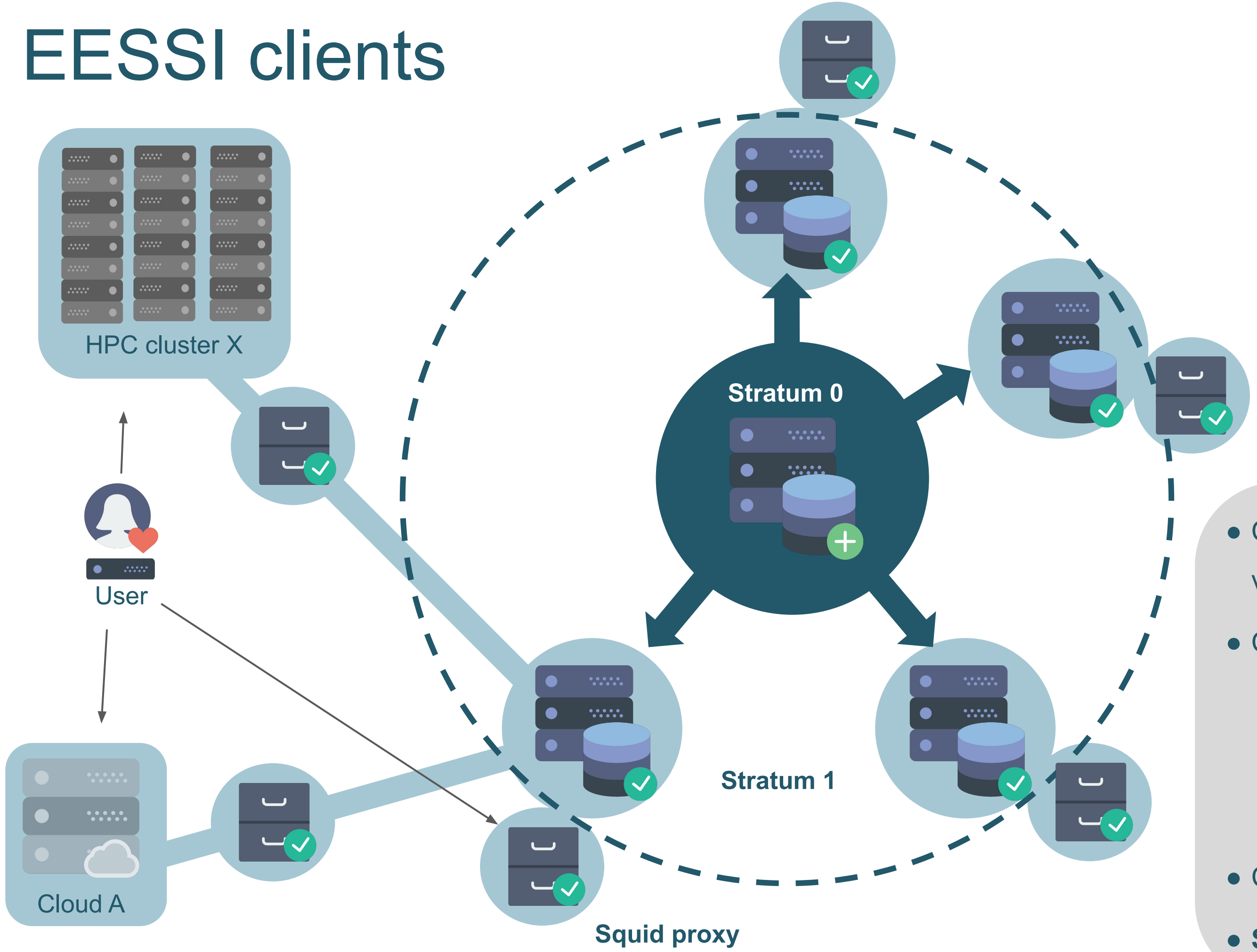
CernVM File system

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

Squid proxy

- Reverse proxy for Stratum 1 servers
- Caching to improve client I/O performance
- Reduce load on Stratum 1 servers
- Can be used for load balancing
- Clients usually connect to one of these

EESSI clients



- Clients access software stack via (squid proxy of) a Stratum 1
- Clients include:
  - Laptops
  - Personal workstations
  - HPC clusters
  - Cloud instances
- Clients also use local filesystem cache
- **Same software stack everywhere!**

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

EESSI: filesystem layer

Current status

- Initial CernVM-FS setup by Bob Dröge at Univ. of Groningen (NL)
  - Stratum 0 server: 2-core VM, 8GB memory, 1TB disk, Ubuntu 18.04
  - Stratum 1 + proxy: 4-core VM, 8GB, 80GB disk *(too small!)*, CentOS 7
  - ~60GB of disk space used for EESSI pilot repository
  - Additional proxy for RUG in separate VM (same specs as Stratum 1)
- Additional Stratum 1 server set up at Univ. of Oslo by Axel Rosén
  - 2-core VM, Intel Haswell, 8GB memory, 100GB disk, CentOS 7
- **Additional Stratum 1 servers welcome!**
- Primary contact: Bob Dröge (`@bedroge`)

EESSI: filesystem layer

Current status

- Ansible playbook available for deployment + configuration of CernVM-FS (Stratum 0, Stratum 1, squid proxy)
- Clients need to:
  - Install `cvmfs-config-eessi` package (Linux: rpm, deb, macOS: pkg), available via https://github.com/EESSI/filesystem-layer/releases
  - Create minimal local configuration file (`/etc/cvmfs/default.local`)
- See https://github.com/EESSI/filesystem-layer
  - Incl. GitHub Actions workflows for testing all playbooks on different Linux distros
- Detailed instructions in `README.md` *(should be moved to docs!)*

EESSI: filesystem layer

Client configuration via config repository

- CVMFS config repo for distributing all configuration to clients
  - This repository resembles the directory structure of `/etc/cvmfs`
  - Contents are automatically generated using an Ansible playbook
  - Config repo itself gets configured using `cvmfs-config-eessi` package
  - Packages are automatically generated using Github Actions
- Issue: clients can only use ONE config repo…
  - Possible solution: https://github.com/EESSI/filesystem-layer/issues/58
- Alternative: manual configuration of repos. Another package?

EESSI: filesystem layer

Client containers

- Preconfigured Docker image for clients
  - Available via Docker Hub: https://hub.docker.com/r/eessi/client-pilot
- Built using script and Dockerfiles from https://github.com/EESSI/filesystem-layer/tree/master/containers
- Can be easily run with Singularity (using `--fusemount`)
  - See documentation at https://eessi.github.io/docs/pilot
  - Note that each container instance will use its own cache!
  - Workaround: see https://github.com/EESSI/filesystem-layer/issues/37

EESSI: filesystem layer

Problems, goals and TODOs

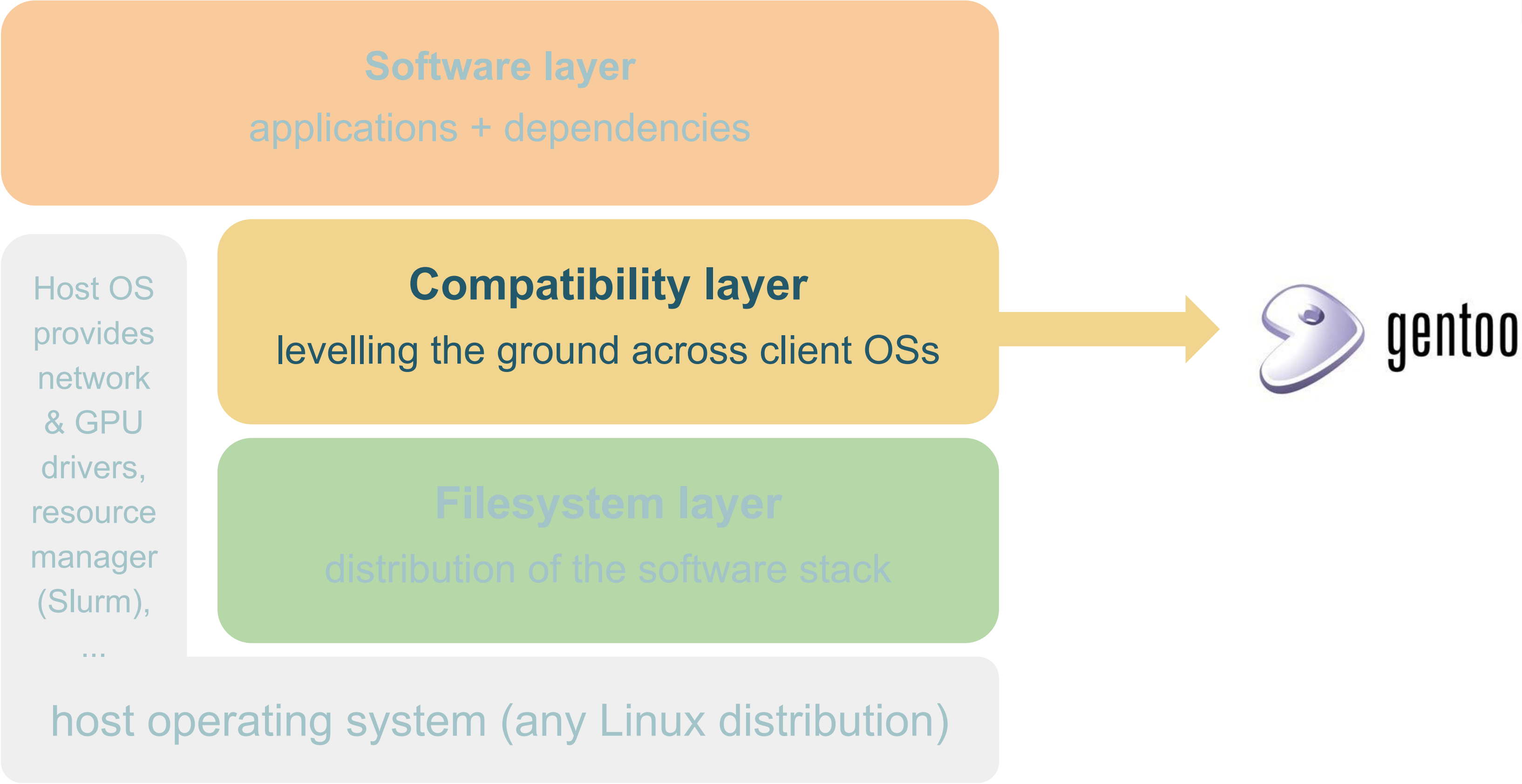
- CernVM-FS duplicates number of open files
  - Problem when client or build node has strict ulimit settings
- Ingesting data into Stratum 0 is cumbersome
  - Tarballs created on build nodes are currently ingested manually using “`cvmfs_server ingest`”
- **Workflow of ingesting + publishing should be automated!**
  - Via GitHub Actions + GitHub App (bot)?

EESSI: filesystem layer

Problems, goals and TODOs

- Stratum 0 requires key management (periodic signing of whitelist)
  - Yubikeys?
- More documentation needed:
  - Native access for clients by installing and configuring CernVM-FS
  - Setting up alien cache (air-gapped systems, multi-node jobs via containers)
- Monitoring/dashboard of infrastructure and (usage) statistics
- Add Stratum 1 servers to cloud (AWS, Azure)
  - maybe also move Stratum 0?

EESSI: compatibility layer



Software layer

applications + dependencies

Compatibility layer

levelling the ground across client OSs

Filesystem layer

distribution of the software stack

Host OS provides network & GPU drivers, resource manager (Slurm), …

host operating system (any Linux distribution)

EESSI: compatibility layer

Leveling the ground across client OSs

- A complete system layer without kernel and drivers
- Built via Gentoo Prefix
- Built once for each CPU “family” (`x86_64`, `aarch64`, `ppc64le`)
- Separate compatibility layer for Linux and macOS clients
- Provides core “OS” dependencies like OpenSSL and glibc
- Updated infrequently during production (only security updates)

EESSI: compatibility layer

Leveling the ground across client OSs

```
$ ls /cvmfs/pilot.eessi-hpc.org/2020.12
compat  init  software

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/compat
linux

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/compat/linux
aarch64  x86_64

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/compat/linux/x86_64
bin  etc  lib  lib64  run  sbin  stage1.log  stage2.log  stage3.log  startprefix  tmp  usr  var

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/compat/linux/x86_64/{bin,usr/bin}
<standard OS tools, like bash, bzip2, cat, gzip, make, patch, rm, ...>

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/compat/linux/x86_64/lib64
ld-linux-x86-64.so.2 libc.so.6 libm.so.6 libmvec.so.1 libnsl.so.1 libpthread.so.0 librt.so.1 ...
```

EESSI: compatibility layer

Installation

- Ansible playbook that does the entire installation
  - https://github.com/EESSI/compatibility-layer/tree/master/ansible/playbooks
  - Install Prefix (inside a Singularity container), add overlay, install packages (architecture-specific sets), do configuration (e.g. symlinks to host files)
  - Uses a slightly customized Prefix bootstrap script
  - We use/keep our own copy of an available Prefix snapshot
  - Github Action that runs it in a Docker container with a prebuilt Prefix
- Run once for every CPU family (`x86_64`, `aarch64`, `ppc64le`, ...)

EESSI: compatibility layer

Installation

- Gentoo overlay: https://github.com/EESSI/gentoo-overlay
  - Customized ebuild files (Gentoo packages) and profile/build settings
  - For Lmod, archspec, rdma-core (Infiniband), opa-psm2 (Omnipath), ...
  - Custom *package sets* for EESSI,
    see https://github.com/EESSI/gentoo-overlay/tree/master/etc/portage/sets
  - One common set, additional sets of specific type of OS + CPU family

EESSI: compatibility layer

Status

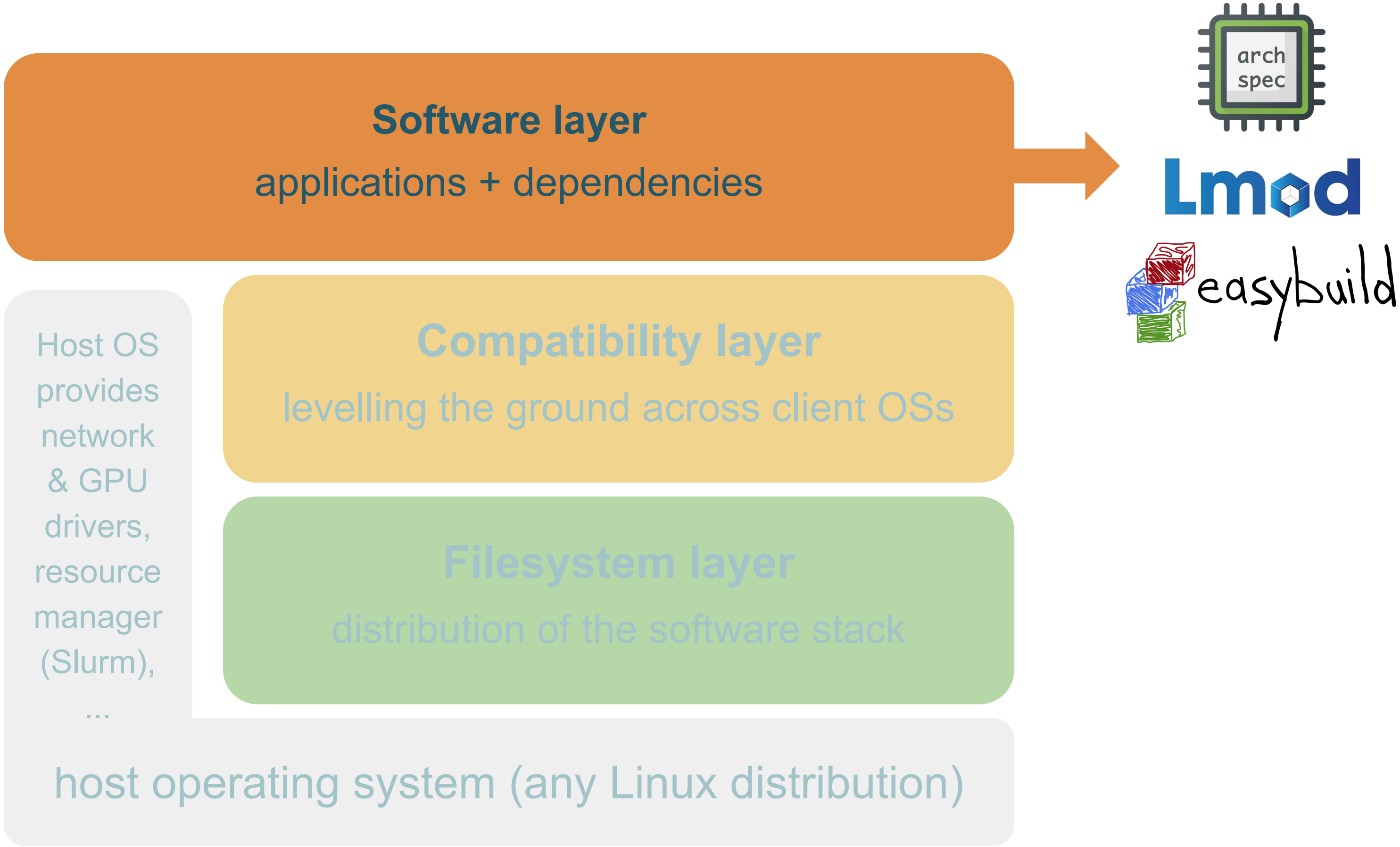
- What we have usually works well
- Not completely identical across architectures and operating systems, e.g. some libraries are x86_64 only (!)
- Every now and then we need to:
    - Sync the bootstrap script with upstream changes
    - Try/use a newer snapshot, and store it on our server
- Primary contact: Peter Stol (`@peterstol`)

EESSI: compatibility layer

Goals

- Needs more automation for building, but that’s coming...
- Initially based on a fork of Compute Canada’s overlay, but we don’t really use their files, so we can/should probably “defork”...
- POWER support is WIP: https://bugs.gentoo.org/show_bug.cgi?id=755551
- Security monitoring: https://www.gentoo.org/support/security/

EESSI: software layer



Software layer

applications + dependencies

Compatibility layer

levelling the ground across client OSs

Filesystem layer

distribution of the software stack

Host OS provides network & GPU drivers, resource manager (Slurm), ...

host operating system (any Linux distribution)

EESSI: software layer

Are we there yet?

- Built using EasyBuild + Lmod + archspec
- Built for specific CPU microarchitectures (Haswell, Rome, Graviton2)
- Built on top of compatibility layer (avoid libraries from host OS!)
- Built using a Singularity container to have full control of build environment
- Provides the module tree via Lmod (Lua only, Tcl to be added)
- Scripted, but we hope to fully automate the process of adding software...

EESSI: software layer

Singularity build container

- Similar to client container, but with fuse-overlayfs
  - https://hub.docker.com/r/eessi/fuse-overlay
  - built with scripts in https://github.com/EESSI/software-layer
- Weird issues with Singularity + newer versions of fuse-overlayfs
  - https://github.com/containers/fuse-overlayfs/issues/232
  - Current workaround: use very old version...
- Minimal container to prevent builds from picking up host libraries
- `fuse-overlayfs` adds a writable overlay on top of `/cvmfs`
- Added files end up in an “upper” directory, mounted from the host

EESSI: software layer

Build script

https://github.com/EESSI/software-layer/blob/master/EESSI-pilot-install-software.sh

- Rudimentary bash script which configures and runs EasyBuild to install a (limited) set of applications in EESSI pilot repo
- Intention is to start using new “easystack files” feature soon, see https://docs.easybuild.io/en/latest/Easystack-files.html
- Can easily be executed on different microarchitectures
- Resulting “upper” dir is tarred and ingested into the repository

EESSI: software layer

Init script

https://github.com/EESSI/software-layer/tree/master/init

- Init script sets up environment to use software provided by EESSI
- Uses archspec to detect best fit for host CPU microarchitecture
- Falls back to `{x86_64,aarch64}/generic` if needed
- Separate init script for using EESSI in Magic Castle (cluster-in-cloud)

```
$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/init
Magic_Castle  README.md  bash  eessi_environment_variables  eessi_software_subdir_for_host.py
```

EESSI: software layer

Status and goals

- Included software: Bioconductor, GROMACS, OpenFOAM, TensorFlow
- Example scripts available for each: https://github.com/EESSI/eessi-demo
- Several upstream fixes done for `aarch64`, `ppc64le`, ...
- Building on cloud (AWS, Azure) for specific CPU microarchitectures can be troublesome (no guarantees w.r.t. specific CPUs)
- We would like to also add GPU-capable installations
  - Need to check how we can adhere to the NVIDIA EULA w.r.t. redistributing...
- Primary contact: Kenneth Hoste (`@boegel`)

EESSI: software layer

```
$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software
aarch64  x86_64

[kehoste@generoso ~]$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/aarch64/
generic  graviton2  thunderx2

[kehoste@generoso ~]$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64
amd  generic  intel

[kehoste@generoso ~]$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/amd
zen2

[kehoste@generoso ~]$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel
haswell  skylake_avx512
```

EESSI: software layer

```
$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel/haswell
modules  software

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel/haswell/modules/all
foss GCCcore GROMACS OpenFOAM R R-bundle-Bioconductor TensorFlow

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel/haswell/software
foss GCCcore GROMACS OpenFOAM R R-bundle-Bioconductor TensorFlow

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel/haswell/software/OpenFOAM
8-foss-2020a  v2006-foss-2020a

$ ls /cvmfs/pilot.eessi-hpc.org/2020.12/software/x86_64/intel/haswell/software/TensorFlow
2.3.1-foss-2020a-Python-3.8.2
```

EESSI: software layer

Testing

- Intention is to test installed software with ReFrame
- Different types of tests: smoke tests, small/medium/large tests, scaling, …
- Until now: mostly discussions…

  https://github.com/EESSI/software-layer/wiki/Brainstorm-meeting-(Dec-16th-2020)--testing
- How do we implement tests while not hardcoding
- Primary contact: Caspar van Leeuwen (`@casparvl`)

EESSI: documentation

https://github.com/eessi/docs - https://eessi.github.io/docs

- Documentation is built with MkDocs (https://www.mkdocs.org)
  - Theme: *Material* (https://squidfunk.github.io/mkdocs-material)
- Makefile is provided to make things easy
  - “`make`” to build docs
  - “`make test`” to run tests (check for broken references, etc.)
  - “`make preview`” to serve rendered docs locally
- Documentation is updated automatically when PR is merged
- GitHub Actions takes care of running tests + deploying updated docs
- Primary contact: Robbert Eggermont (`@robberteggermont`)

EESSI @ GitHub

- All code, documentation, TODOs, etc. are in GitHub repositories
- GitHub organisation: https://github.com/EESSI
- One repository per layer + `gentoo-overlay` (part of compat layer)
- Additional repos for: documentation, meetings (notes), demo scripts
- Ideally all TODOs have a corresponding open issue…
- Projects are useful to organize work: https://github.com/orgs/EESSI/projects
  - We should start “themed” projects (like Automation, Testing, Monitoring, Security, …)?

Git workflow

- All changes to master branch done via pull requests (PRs)
- Pull requests should be reviewed + merged by someone else
  - “Two pairs of eyes” policy
  - Never merge your own pull request!
  - Few people per repository have merge permissions (can be extended)
- For people unfamiliar with git, see:

  https://github.com/EESSI/meetings/tree/master/other/git-training-20200703

Available resources for EESSI

- AWS (EC2): large amounts of sponsored credits provided
    - Can be used for Stratum 1, build nodes, testing, …
    - Supported: Linux + **macOS**, Intel + AMD + **Arm64** (Graviton 2)
- Microsoft Azure: sponsorship pending…
    - Intended usage: Stratum 1, build nodes, testing, ...
- OSU Open Source Lab (https://osuosl.org):

  POWER9 instances via OpenStack
- Individual resources (like Terje's Mac Mini with M1)

EESSI organisation

- Monthly update meeting: every first Thursday of the month, 2pm CET
- Focused meetings on specific topics in between monthly meetings
- Aiming for monthly revision of EESSI pilot repository
  - Fixing known problems, changes motivated by testing, etc.
- Informal discussions via Slack (focused channels)
- All members can be reached via mailing list: `eessi@list.rug.nl`