EBRAINS

Co-funded by the European Union

ESD: Build Paths

Eric Müller

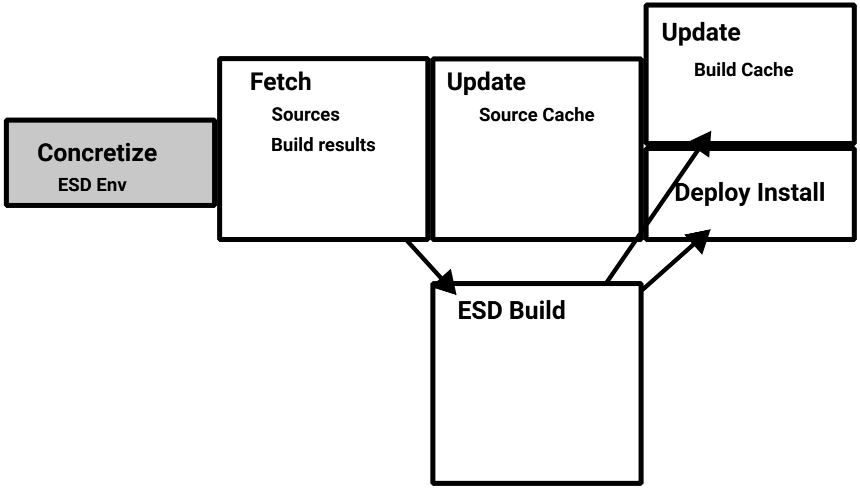
2024-11-28

Co-funded by the European Union

Motivation

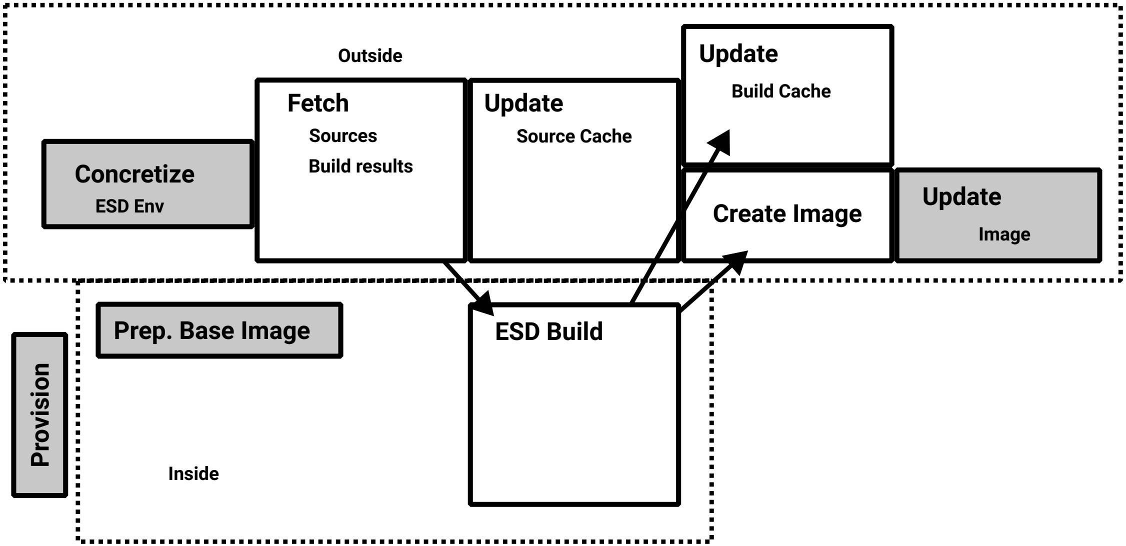
- Usage of the ESD…
  - in the lab,
  - on your laptop,
  - on HPC,
  - to run a (your) service.
- implies different build paths depending on
  - “target”-specific optional components (e.g., lab)
  - optimization
  - dependencies on underlying base install/image

(Full) Build Process for the Lab



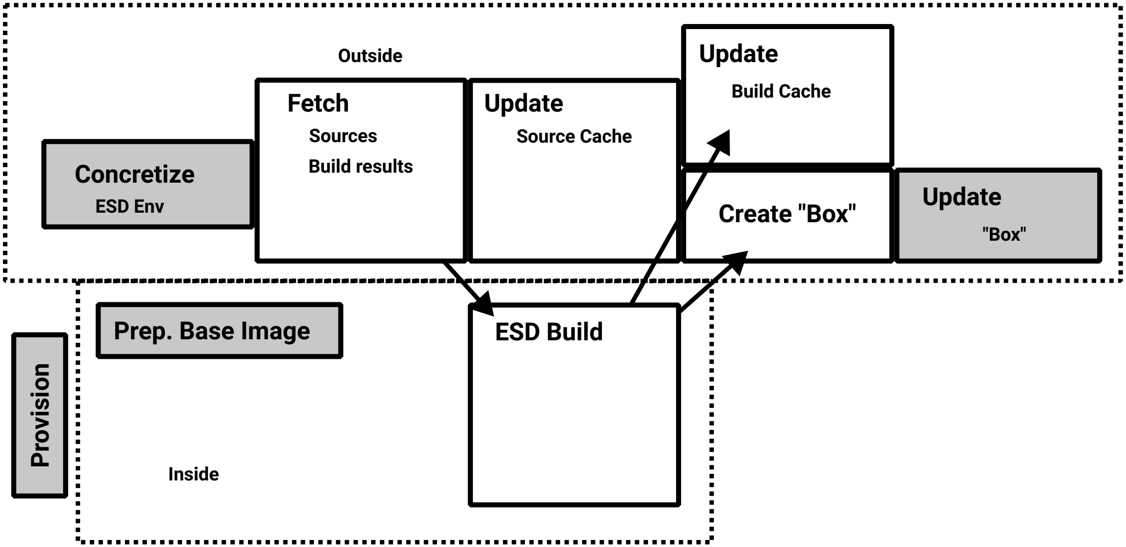
Co-funded by the European Union

(Full) Build Process for Container Images



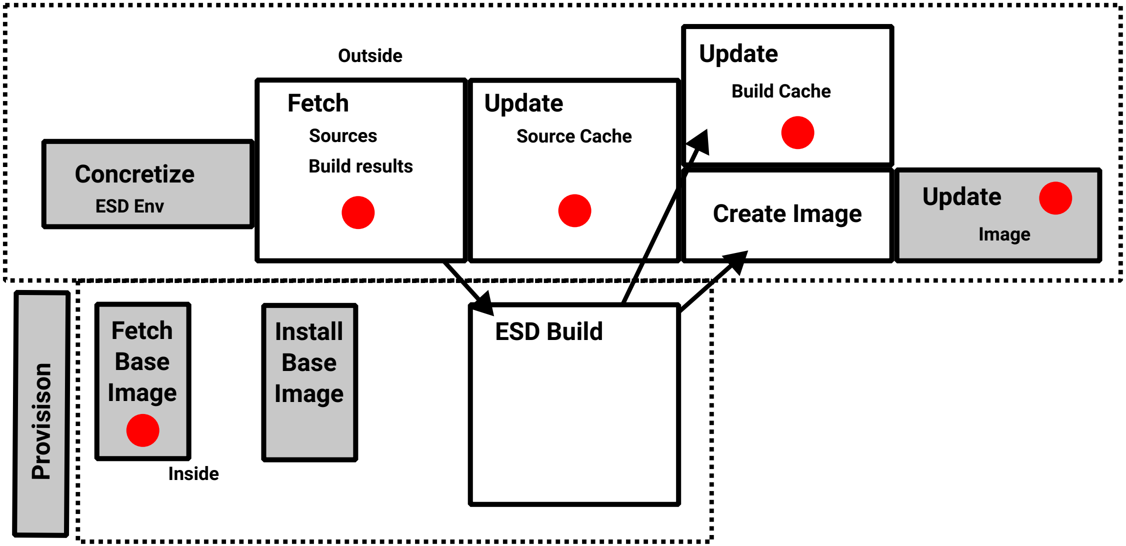
Co-funded by the European Union

(Full) Build Process for Virtual Machines



Co-funded by the European Union

(Full) Build Process for HPC Optimized Images



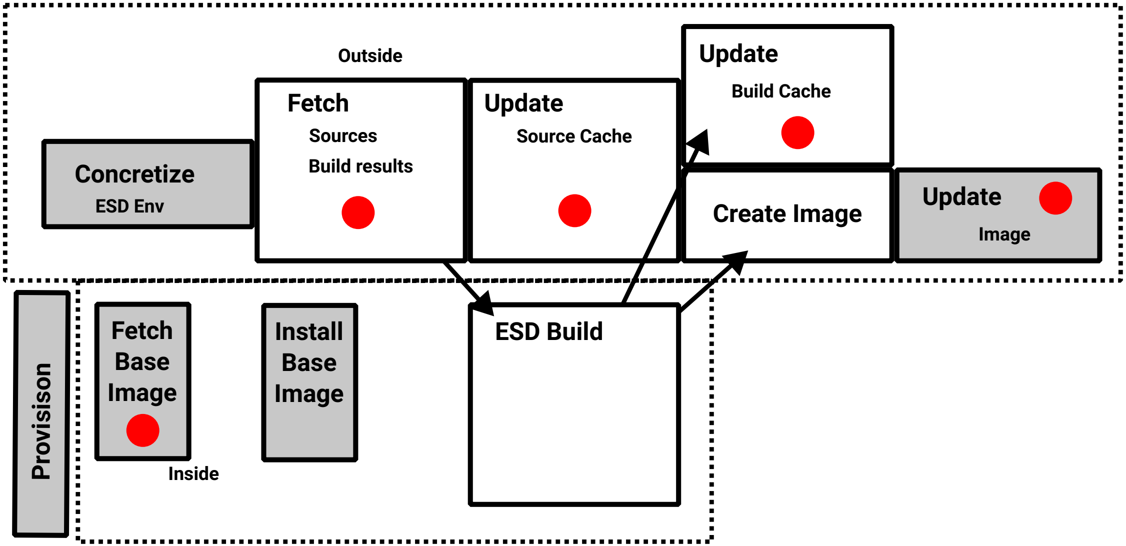
Co-funded by the European Union

Cacheable Entities and Build Persistence

- `spack`
  - sources
  - build results / `spack install` buildcache
- Images:
  - Base image / base image contents?
  - Final image
- Filesystem deployments
  - r/w distributed FS
  - separate write/deployment from r/o mount

Co-funded by the European Union

(Full) Build Process for HPC Optimized Images



“Outside” & “Inside” Environments

- gitlab runners
  - OCI runtime-based container
  - could be different. . .
- image build envs:
  - “`chroot`”
  - container runtime
  - (thick) VM
- HPC builds:
  - online and offline build resources
- user CLI

Encapsulation Levels

“Pure” Userspace (Type “0”)

- `mount --bind`
- `SECCOMP` or `ptrace()`
- (Type I requires mount namespace)

User namespace (Type II)

- “unprivileged”[a] or “rootless”
- requires user namespaces (`unshare(int)`)

[a] fails on unprivileged gitlab docker runners

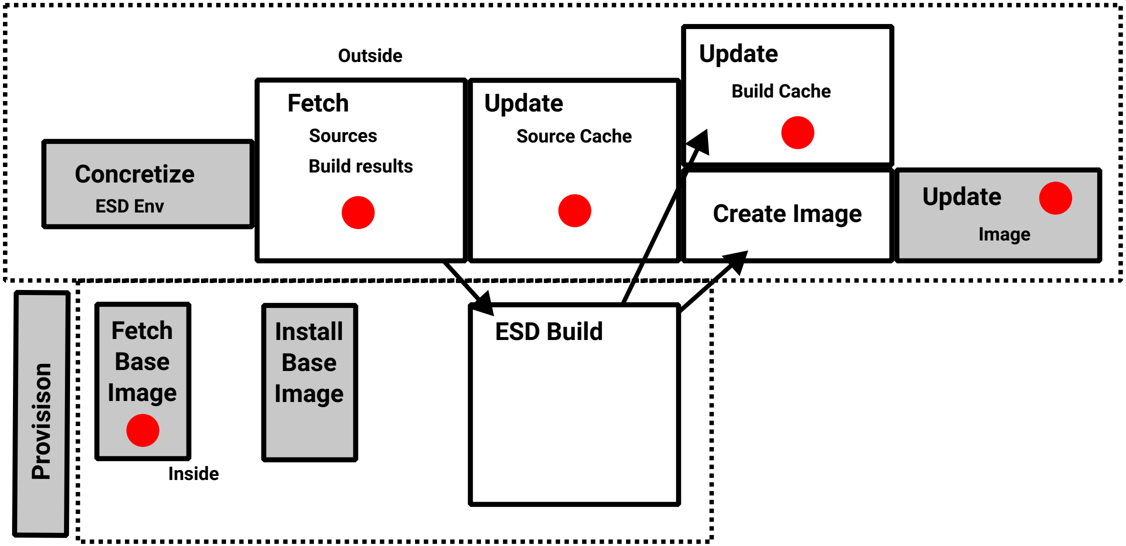
Classic OCI (Type III)

- not needed for software builds
- syscall-requiring package managers (`chown()`)...
- useful for certain service deployments

“Pure Userspace” is basically a Type 0 that doesn’t require mount namespace “Zero-consistency root emulation for unprivileged container image build”, Priedhorsky et al. arXiv:2405.06085 [cs.DC].

Co-funded by the European Union

(Full) Build Process for HPC Optimized Images



Additional Tools

- Getting “inside”
  - `proot` — Type 0
  - `charliecloud` — Type I
  - `apptainer` — Type II
  - `buildah` — Type II
- image file and OCI registry things
  - `oras` — handling artifacts in OCI registries
  - `skopeo` — handling OCI images and image repos
- PoC for modular/image ESD builds:
  `https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/yashchiki`

Co-funded by the European Union

Co-funded by the European Union

Misc Aspects

- Build path parallelism
  - `spack env depfile`
  - local, CI, scalable resources (`unicore`/`pyslurm`?)
  - per-package build requirements (e.g., memory per core)
- Filesystem recommendations for building software and images
  - `/dev/shm` recommended for server-type machines
  - avoid distributed fileystems (GPFS et al., NFS also not too much fun)
  - FUSE-mounted ones $\rightarrow$ depends (overlayfs etc. is fine)

Summary

- Modular concept from the definition of the ESD (set of toplevel packages to a deployment)
- PoC implementation for image builds via “yashchiki”[1]

Next steps:

- CI flow of (laptop) image operation on development branch
- modularization of build “paths” to run all mentioned combinations in CI
- Hands-on: Exploring container image builds on HPC/JUSUF

[1] `https://gitlab.ebrains.eu/ri/tech-hub/platform/esd/yashchiki`