Hewlett Packard Enterprise

BOF EXTENDING SOFTWARE TOOLS FOR CSM

Harold Longley HPE, Ryan Haasken HPE,
Graham van Heule LANL, Alden Stradling LANL,
Brian Friesen NERSC, Doug Jacobsen NERSC
Manuel Sopena Ballesteros CSCS, Miguel Gila CSCS

CUG 2023

AGENDA

HPE

LANL

NERSC

CSCS

OPEN FORUM

HPE

- Harold Longley, harold.longley@hpe.com
- Ryan Haasken, ryan.haasken@hpe.com

CRAY CLI FRAMEWORK FROM REST API SPECIFICATION

```
user@ncn> cray auth login --username UserWithAdminRole
Password:
user@ncn> cray --help
Usage: cray [OPTIONS] COMMAND [ARGS]...
  Cray management and workflow tool
Options:
  --version  Show the version and exit.
  --help     Show this message and exit.
Commands:
  init     Initialize/reinitialize the Cray CLI
```

- Documentation convention is that if the admin role is required for cray CLI or sat CLI, then the command prompt will use `hostname#` rather than `user@hostname>`
- Linux account and Keycloak authentication are different credentials

Management services which have API specifications

Groups:

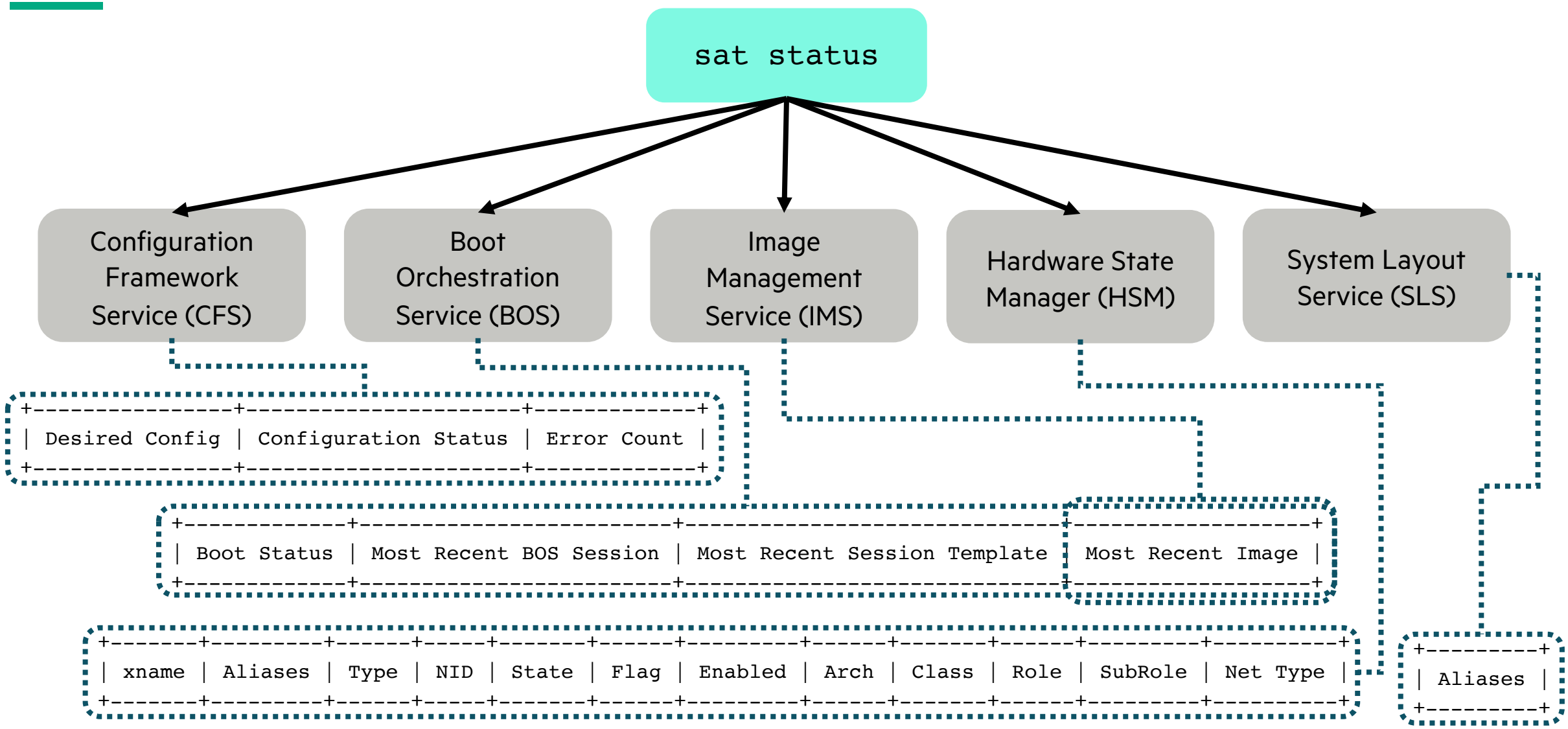
artifacts	Manage artifacts in S3
auth	Manage OAuth2 credentials for the Cray CLI
badger	Badger Service API
bos	Boot Orchestration Service
bss	Boot Script Service API
capmc	Cray Advanced Platform Monitoring and Control API
cfs	Configuration Framework Service
config	View and edit Cray configuration properties
cps	Content Projection Service
crus	Compute Rolling Upgrade Service
fas	Firmware Action Service
hsm	Hardware State Manager API
ims	Image Management Service
nmd	Node Memory Dump Service
scsd	System Configuration Service
sls	System Layout Service
uas	User Access Service
vnid	Virtual Network Identifier Daemon

SAT COMMANDS

sat auth	Authenticate to the API gateway and save the token	sat k8s	Report on Kubernetes replicasets that have co-located replicas
sat bmccreds	Set BMC Redfish access credentials	sat nid2xname	Translate node IDs to node xnames
sat bootprep	Prepare to boot nodes with images and configurations	sat sensors	Report current sensor data
sat bootsys	Boot or shutdown the system (compute nodes, application nodes, and management nodes)	sat setrev	Set HPE Cray EX system revision information
sat diag	Launch diagnostics on the HSN switches and generate a report	sat showrev	Print revision information for the HPE Cray EX system
sat firmware	Report firmware version	sat slscheck	Perform a cross-check between SLS and HSM
sat hwhist	Report hardware component history	sat status	Report node status across the HPE Cray EX system
sat hwinv	Give a listing of the hardware of the HPE Cray EX system	sat swap	Prepare HSN switch or cable for replacement and bring HSN switch or cable into service
sat hwmatch	Report hardware mismatches for processors and memory	sat xname2nid	Translate node and node BMC xnames to node IDs
sat init	Create a default SAT configuration file		

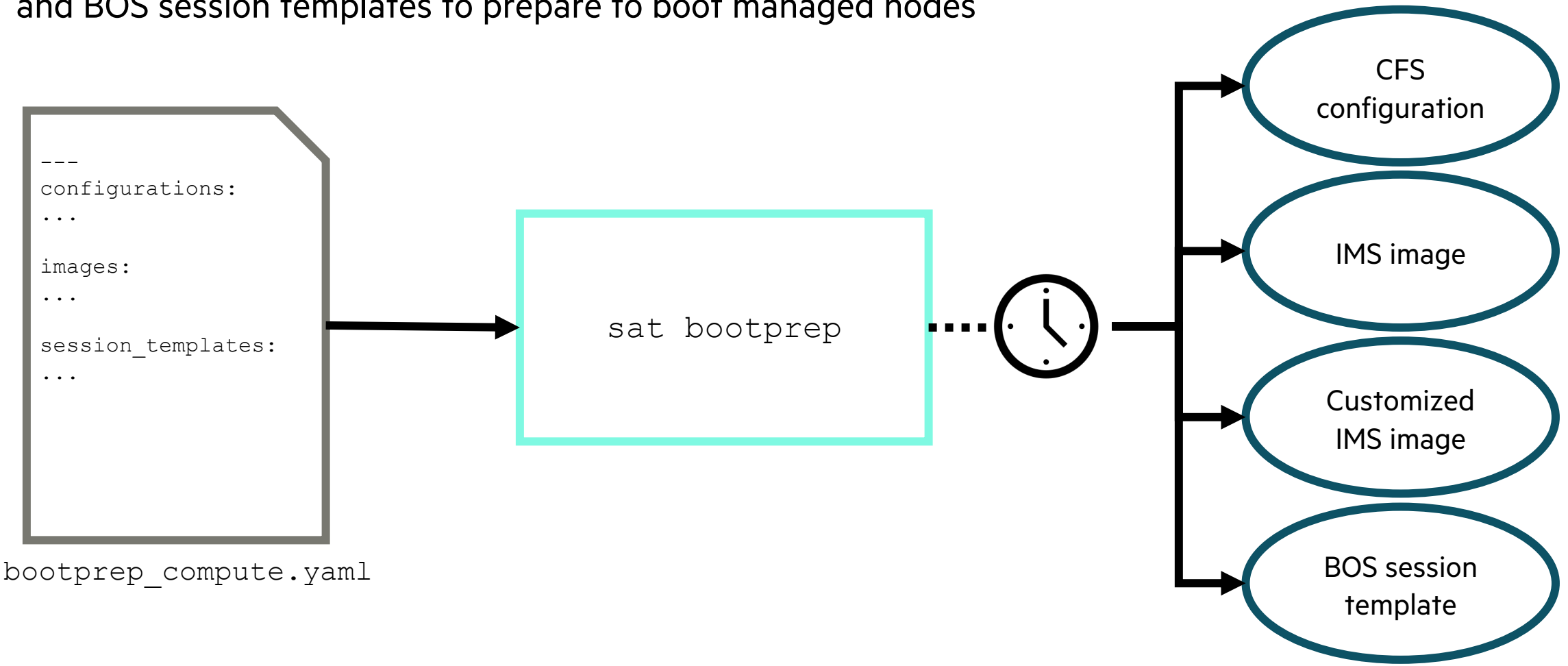
Newest SAT commands

SAT STATUS API REQUESTS



SAT BOOTPREP HIGH-LEVEL DIAGRAM

- `sat bootprep` automates the creation of CFS configurations, IMS images, and BOS session templates to prepare to boot managed nodes



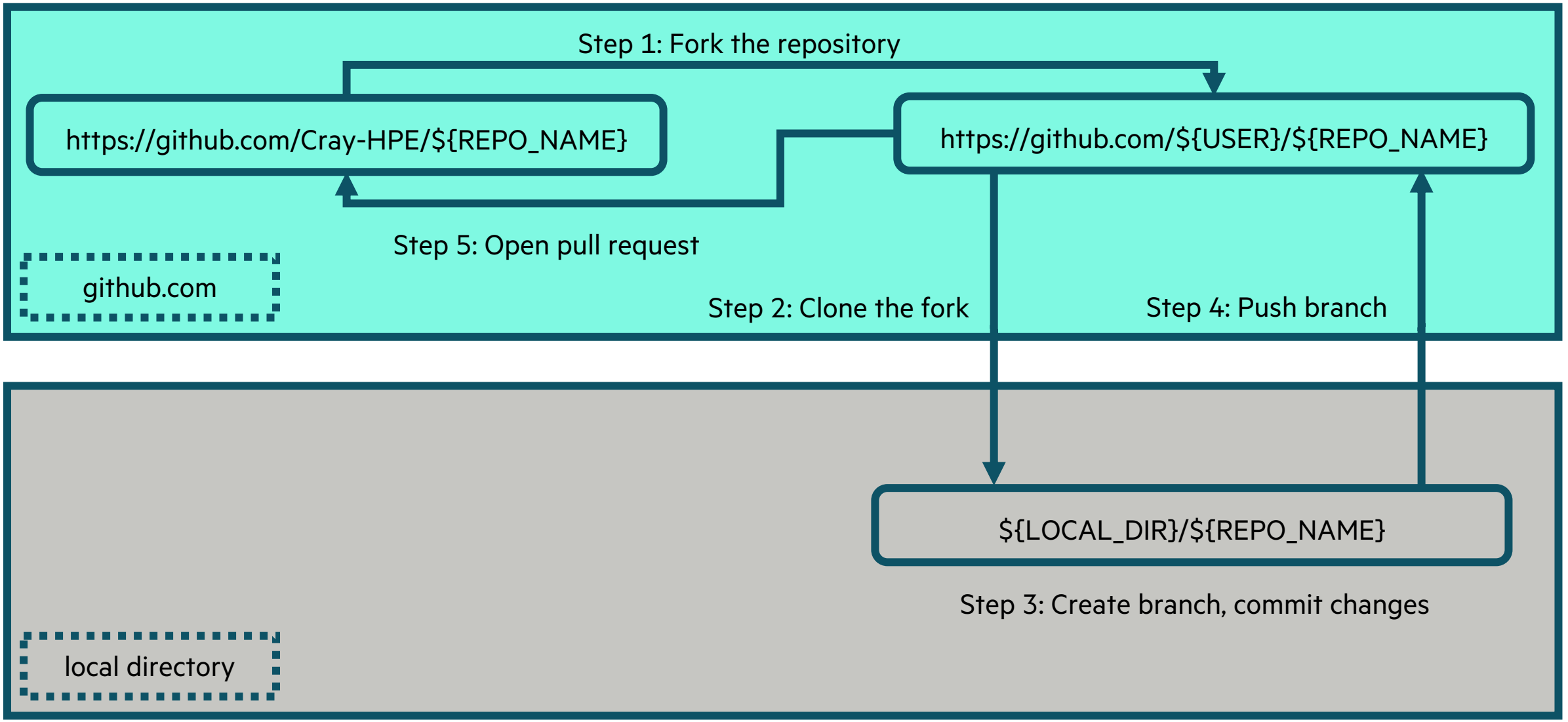
CSM OPEN SOURCE

- CSM is open source under the MIT License
- Code is available on public GitHub under the Cray-HPE organization
  - https://github.com/Cray-HPE
  - https://github.com/Cray-HPE/csm
  - https://github.com/Cray-HPE/docs-csm
  - https://github.com/Cray-HPE/uan
  - https://github.com/Cray-HPE/docs-uan
- The following documentation is a helpful starting point
  - https://github.com/Cray-HPE/community
  - https://github.com/Cray-HPE/community/blob/main/governance.md
  - https://github.com/Cray-HPE/community/blob/main/contributors/guide/pull-requests.md

SAT OPEN SOURCE

- SAT is open source under the MIT License
- Code is available on public GitHub under the Cray-HPE organization
  - Primary repository for the sat CLI written in Python: https://github.com/Cray-HPE/sat
  - Podman wrapper script written in Bash: https://github.com/Cray-HPE/sat-podman
  - An important library used by sat CLI: https://github.com/Cray-HPE/python-csm-api-client
- The following documentation is a helpful starting point
  - https://github.com/Cray-HPE/sat/blob/integration/CONTRIBUTING.md
  - https://github.com/Cray-HPE/sat/blob/integration/docs/developer/README.md

EXTERNAL PARTICIPANT WORKFLOW DIAGRAM



LANL

- Graham van Heule, grahamvh@lanl.gov
- Alden Stradling, stradling@lanl.gov

- Live demo of shasta wrapper
  - Scripting to simplify the administration of HPE Cray EX Systems
  - https://github.com/hpc/shasta_wrapper

NERSC

- Brian Friesen, bfriesen@lbl.gov
- Doug Jacobsen, dmjacobsen@lbl.gov

- Accelerating node boots with systemd
  - See attached PDF

CSCS

- Manuel Sopena Ballesteros, manuel.sopena@cscs.ch
- Miguel Gila, miguel.gila@cscs.ch

- Live demonstration of manta
  - See attached PDF
  - https://github.com/eth-cscs/manta

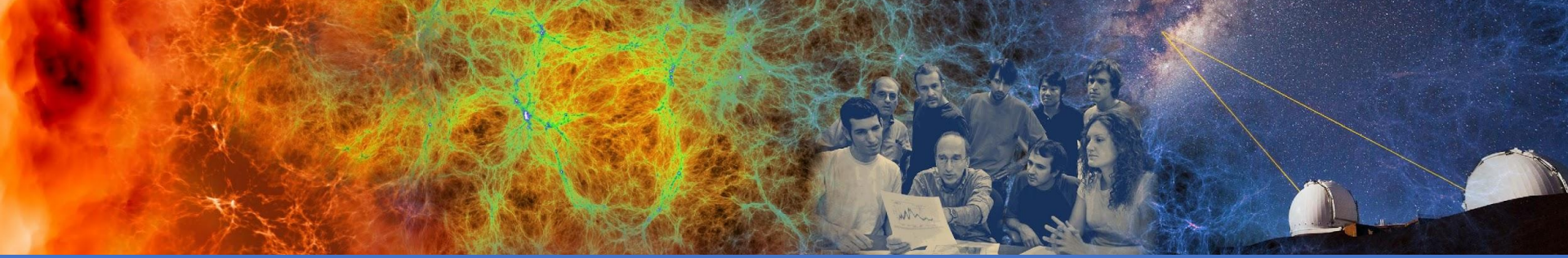
THANK YOU

harold.longley@hpe.com

Some CSM enhancements at NERSC

CUG 2023

Brian Friesen
National Energy Research Scientific Computing Center
Lawrence Berkeley National Laboratory
May 8, 2023



Accelerating node boots with systemd

Problem statement

- **Managed node boot times are too slow.**
  - Compute nodes and UANs on Perlmutter took ~50 minutes from `cray capmc xname_on` to fully configured.
  - Contributing factors:
    - POST
    - Some Ansible plays scale poorly on a system with ~5000 nodes
    - Some node personalization activities that could be moved to image customization

Current state

- UAN/compute node boot times reduced to ~10 minutes.
- Several Ansible plays moved from node personalization to image customization.
- Several other plays were rewritten altogether for better scalability - e.g., CHN IP address assignment
- **Several plays were converted from Ansible to RPMs containing a systemd service file with associated configuration files**

Example: provisioning GPUs using systemd

```
[Unit]
Description=Load NVIDIA GPU drivers
[Service]
Type=oneshot
ExecStart=/bin/bash -c "depmod -v -w $(uname -r)"
ExecStart=/bin/bash -c "modprobe nvidia ${NVIDIA_KMOD_PARAMS}"
ExecStart=/bin/bash -c "modprobe nvidia-uvm"
ExecStart=/bin/bash -c "modprobe nvidia-drm nvidia-modeset"
ExecStart=/bin/bash -c "nvidia-persistenced --persistence-mode"
ExecStart=/bin/bash -c "nvidia-smi -c ${GPU_COMPUTE_MODE}"
ExecStart=/bin/bash -c "modprobe gdrdrv"
ExecStart=/bin/bash -c "/usr/sbin/create_gdrdrv_device.sh gdrdrv"
```

Example: provisioning GPUs using systemd

```
muller:login01:~ # cat
/usr/lib/systemd/system/cray_nvidia_ld_drivers.service.d/n
vidia-kmod-params.conf
[Service]
Environment=NVIDIA_KMOD_PARAMS="NVreg_RestrictProfilingToA
dminUsers=0 NVreg_DeviceFileMode=0666"
```

Example: provisioning GPUs using systemd

```
muller:login01:~ # cat
/usr/lib/udev/rules.d/99-nvidia-gpu.rules
SUBSYSTEM=="pci", DRIVER=="nvidia", TAG+="systemd",
ENV{SYSTEMD_WANTS}="cray_nvidia_ld_drivers.service"
```

Drawbacks and considerations

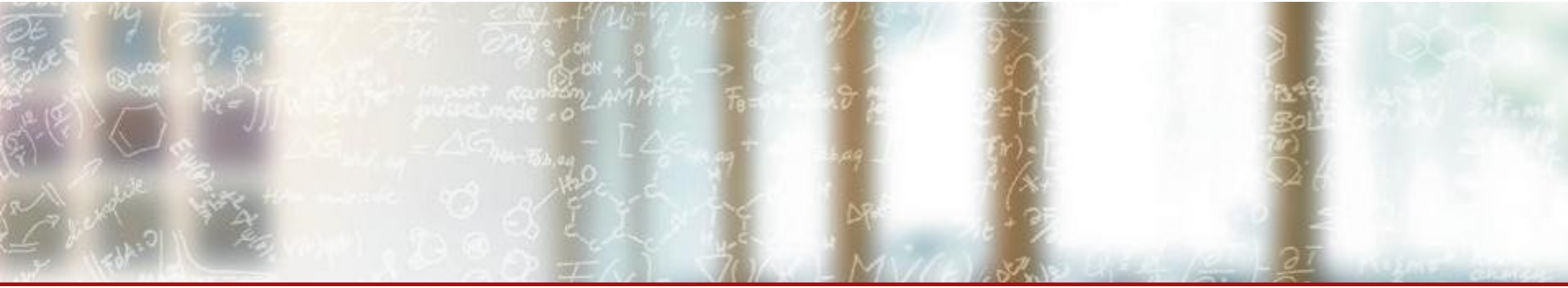
- Mixing Ansible with RPMs+systemd adds complexity to image recipes and node configurations
  - Configurations now come from 2 places instead of 1
  - Node personalization becomes more complex
    - a systemd service waits for a file to appear, and the file is touched by an Ansible role when it completes
- Minimizing complexity most likely means moving (almost) entirely to pure Ansible or pure RPM+systemd

Other thoughts

- Ansible is still useful for several things
  - Building images (speed is nice, but not critical)
  - Injecting secrets during node personalization
- udev is great for distinguishing diverse nodes types that boot the same image

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Manta CLI for CSM

CUG 23

Manuel Sopena Ballesteros (manuel.sopena@cscs.ch), CSCS

May 08, 2023

Table of Contents

CSCS

Cray System Management (CSM)

Cray System Management (CSM)

CFS

BOS

Note: This talk is based on CSM 1.2

Configuration Framework Service (CFS)

Ansible

CFS layer

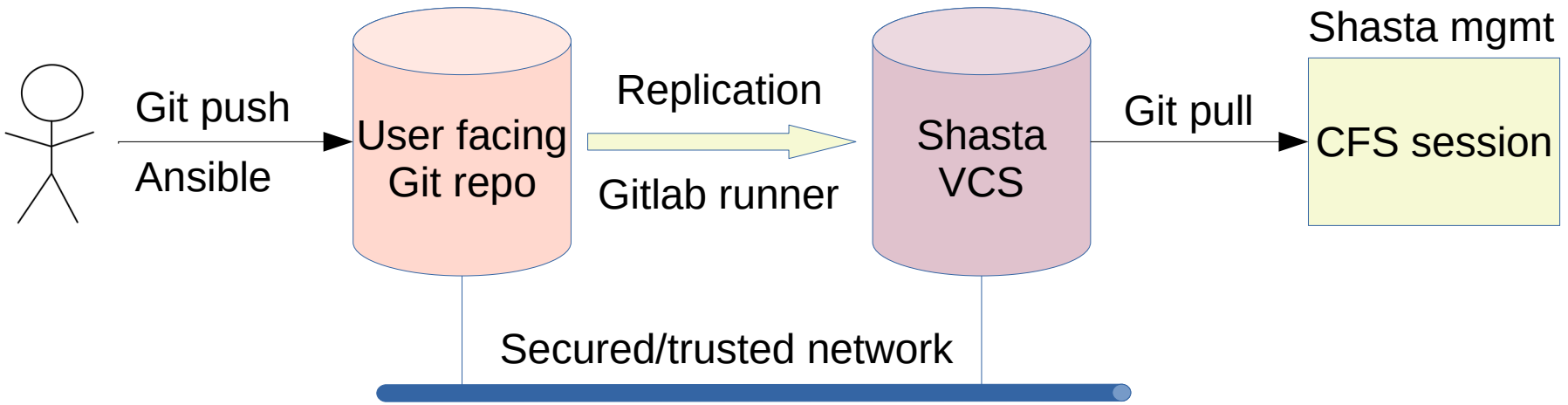
CFS configuration

CFS session

CFS layer

- Divided in 2 parts (description and the ansible playbooks)
- A CFS layer description is a “pointer” to an ansible playbook in a git repository
- Two **target definitions** (node image creation or configuring a running node)
- Granularity at the ansible task level
- Each target definition is identified using Boolean cray_cfs_image ansible variable
- For PSI they should be able to build their own layers to deploy extra stuff on top of the existing ones provided by CSCS

CFS layer (Ansible playbook)



CFS layer (definition)

```
{
    "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cscs-config-management.git",
    "commit": "2d6770fb165a6e252e957ee6ab8398e898b6deae",
    "name": "cscs-integration-2.0.44",
    "playbook": "site.yml"
}
```

CFS layer (definition)

Git repo url to CFS layer code or ansible playbook

```
{
    "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cscs-config-management.git",
    "commit": "2d6770fb165a6e252e957ee6ab8398e898b6deae",
    "name": "cscs-integration-2.0.44",
    "playbook": "site.yml"
}
```

Ansible playbook

Commit id

CFS configuration

- A set of CFS layers
- Layers execution order is sorted by “layer id”
- CFS configuration are related to a cluster only through a CFS session

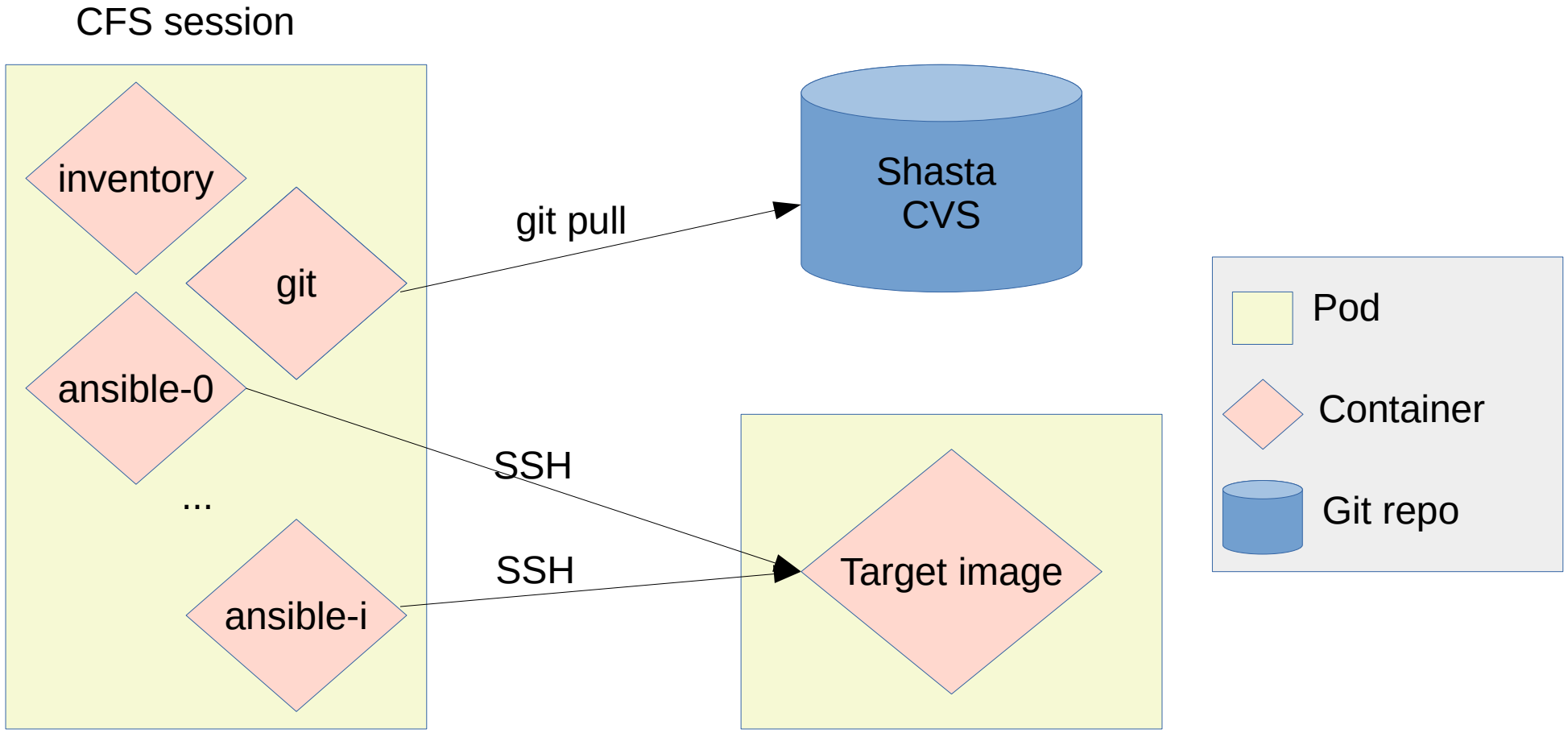
CFS configuration


```
{
   "layers": [
     {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cos-config-management.git",
       "commit": "60535ef51c0d8baafeaeffd8b6060ed0bd80ab52",
       "name": "cos-integration-2.0.40",
       "playbook": "site.yml"
     },
     {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/slurm-config-management.git",
       "commit": "87cfbfaa9f24b0e4dcd701bd373091ed922225f8",
       "name": "slurm-integration-0.2.2",
       "playbook": "site.yml"
     },
     {
       "cloneUrl": "https://api-gw-service-nmn.local/vcs/cray/cscs-config-management.git",
       "commit": "2d6770fb165a6e252e957ee6ab8398e898b6deae",
       "name": "cscs-integration-2.0.44",
       "playbook": "site.yml"
     }
   ]
}
```

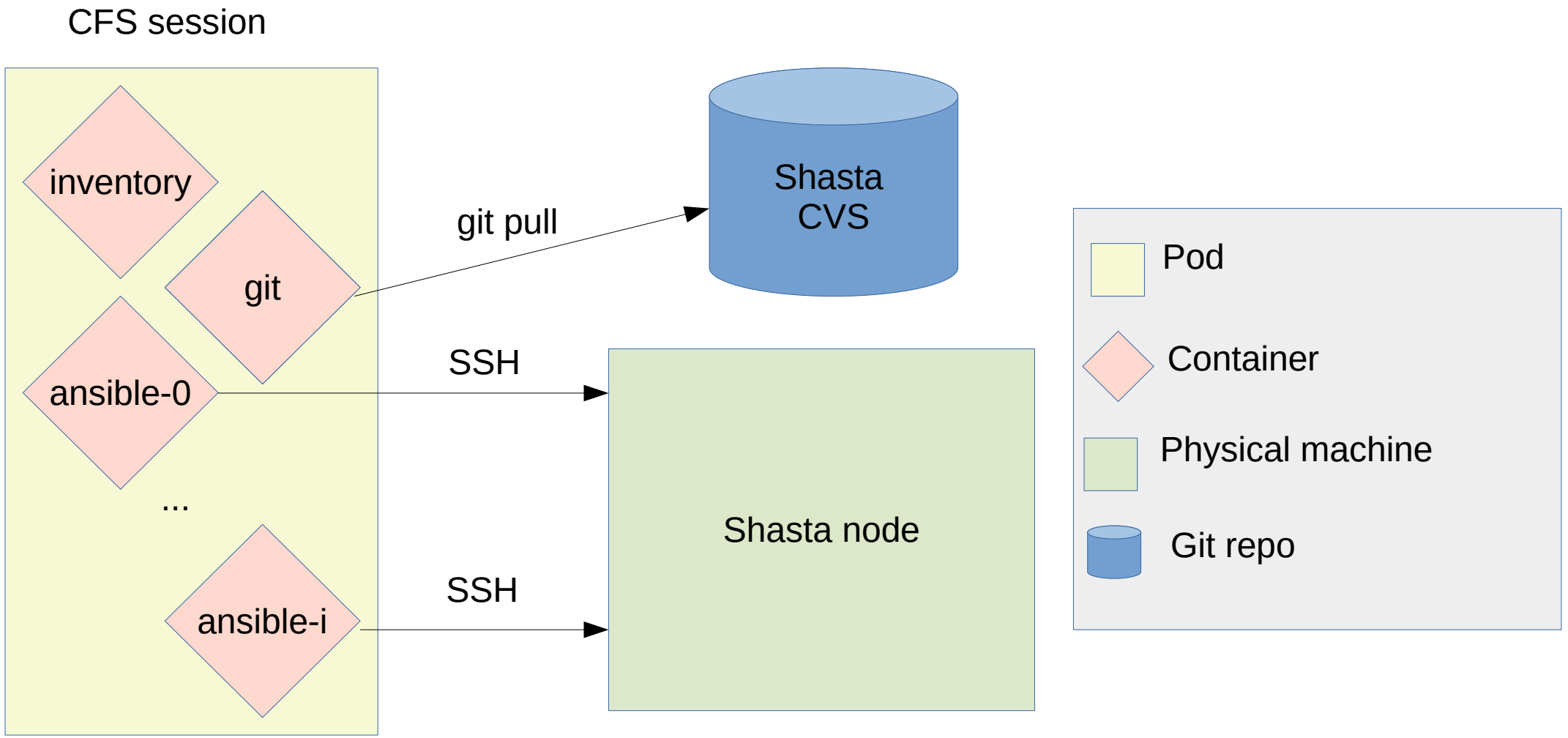
CFS session

- A CFS session is an instance of a CFS configuration
- A CFS session can run on any **target definition**
- A CFS session is orchestrated by Kubernetes
- Kubernetes job instantiates a pod with a set of containers
- A subset of these containers runs the CFS layers (e.g. ansible-0, ansible-1, … ansible-i)

CFS session target image



CFS session target dynamic



Boot Orchestrator Service (BOS)

Configures nodes boot params so they use the right image (BSS)

Reboots nodes associated with a session template (CAPMC)

Overall

Pros:

Improves communication and collaboration across the site

Persists ansible logs

Logs the history of operations done in Shasta

Cons:

Engineers are less productive

Hard to find information

Information is scattered across different systems

Manta

Background

CSM Pros

- Improves communication and collaboration across the site
- Persists ansible logs
- Logs the history of operations done in Shasta

CSM Cons

- Engineers are less productive
- Hard to find information
- Information is scattered across different systems

Initial Solution

- Bash scripts as a wrapper for cray and kubectl CLIs
- Difficult to scale and extend functionalities based on bash

Manta Overview

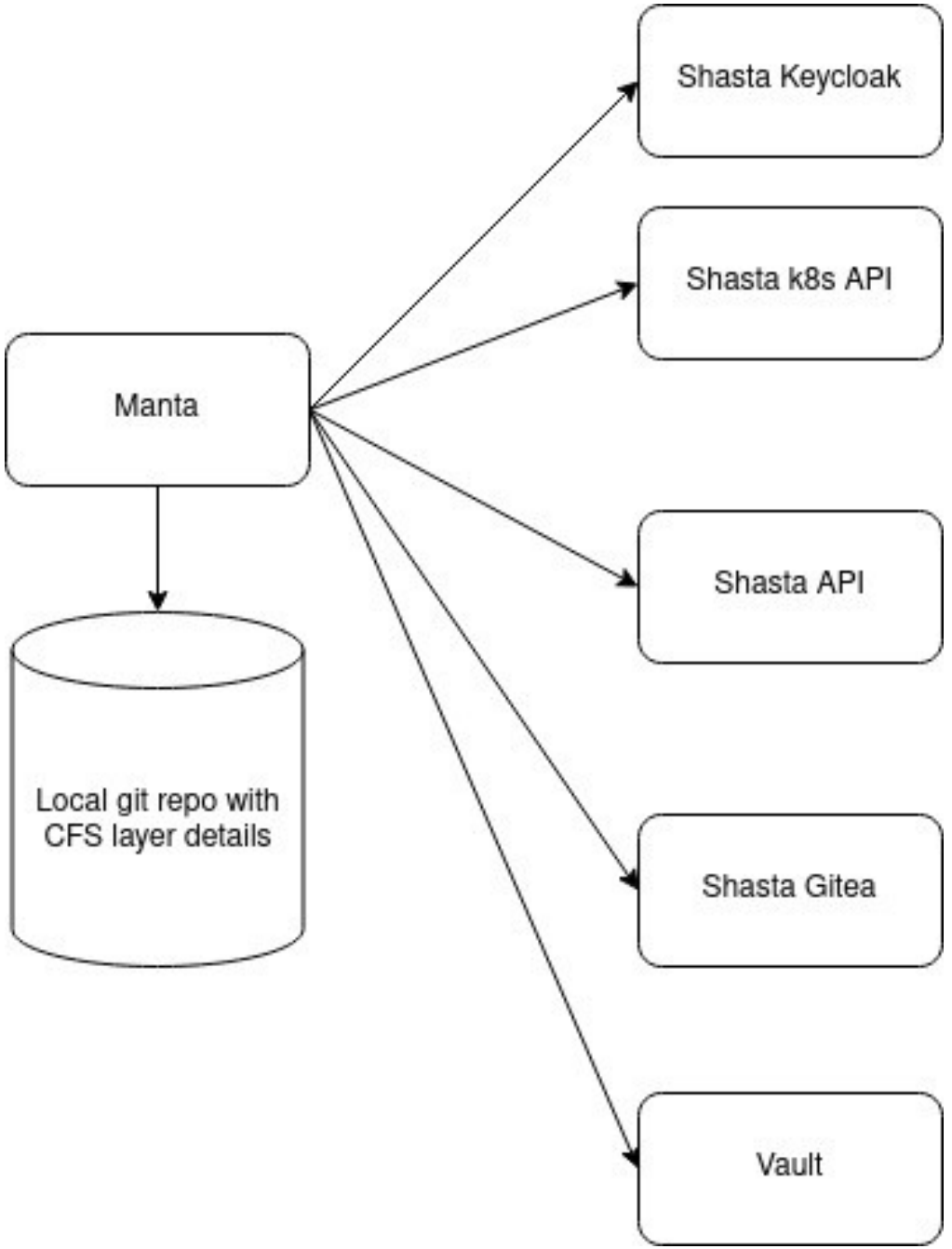
Opinionated Shasta CLI

Ergonomics

Aggregate information from different systems

Binary less

Manta Overview



Manta Overview - Features

- Get information from CSM (CFS, HSM, nodes, etc.)
- Get CFS session logs
- Create CFS configurations
- Create CFS session target definition and image
- Create BOS session template/sessions
- Connect to node’s console
- Configure clusters from file
- Start/Shutdown/Reboot nodes

Manta Overview - Security

- All users using Manta need to have admin role in Keycloak Shasta realm
- K8s and Gitea credentials are stored in Hashicorp Vault
- Hashicorp vault authorization through approle

Manta Overview - Configuration

Configuration file ~/.config/manta/config

```
base_image_id = "4bf91021-8d99-4adf-945f-46de2ff50a3d"
socks5_proxy = "socks5h://127.0.0.1:1080"
shasta_base_url = "https://api.cmn.alps.cscs.ch/apis"
keycloak_base_url = "https://api.cmn.alps.cscs.ch/keycloak"
gitea_base_url = "https://api.cmn.alps.cscs.ch/vcs"
k8s_api_url = "https://10.252.1.12:6442"
vault_base_url = "https://hashicorp-vault.cscs.ch:8200"
vault_role_id = "b15517de-cabb-06ba-af98-633d216c6d99"
hsm_group = "psi-dev"
```

CSM (keycloak) token ~/.cache/manta/http

Manta Overview – Node configuration

1) Check local repositories have all changes committed
2) Check most recent commit id exists in CSM VCS
3) Check nodes are available to run CFS session (Nodes in power state “ready” and no CFS sessions scheduled)

DEMO

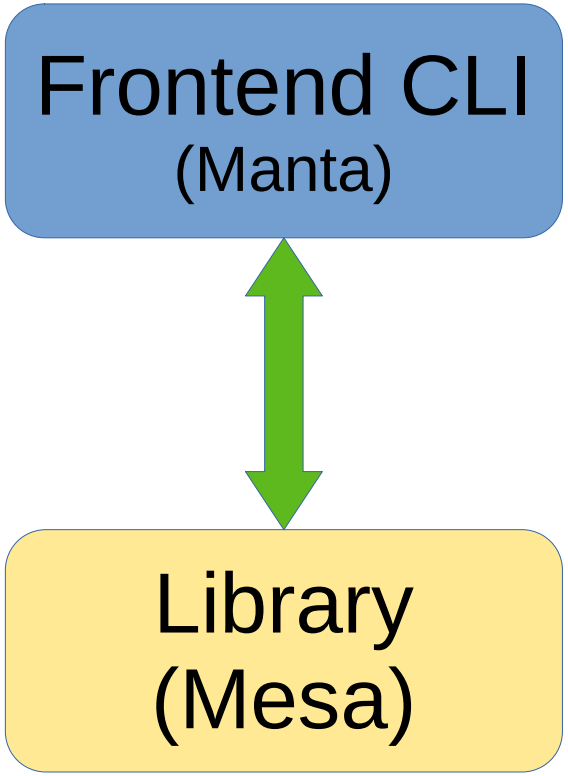
Common commands

- Note: manta cli already has a built in help: `manta --help` or `manta <subcommand> --help`
- List 5 most recent sessions created:
  - `manta get session --limit 5`
- Get node status:
  - `manta get nodes`
- Shutdown a node:
  - `manta apply node off --force <xname>`
- Note: xname example x1004c7s0b0n0
- Build an configuration, then a session, update nodes boot parameters to use new image, reboot them and run a session to finish configuration:
  - `manta apply cluster --file </path/to/sat/file>`
- Create a configuration and a session based on a local repo:
  - `manta apply session --repo-path </path/to/my/local/repo> --name <my-session-name>`
- Get session logs:
  - `manta log <session-name>`

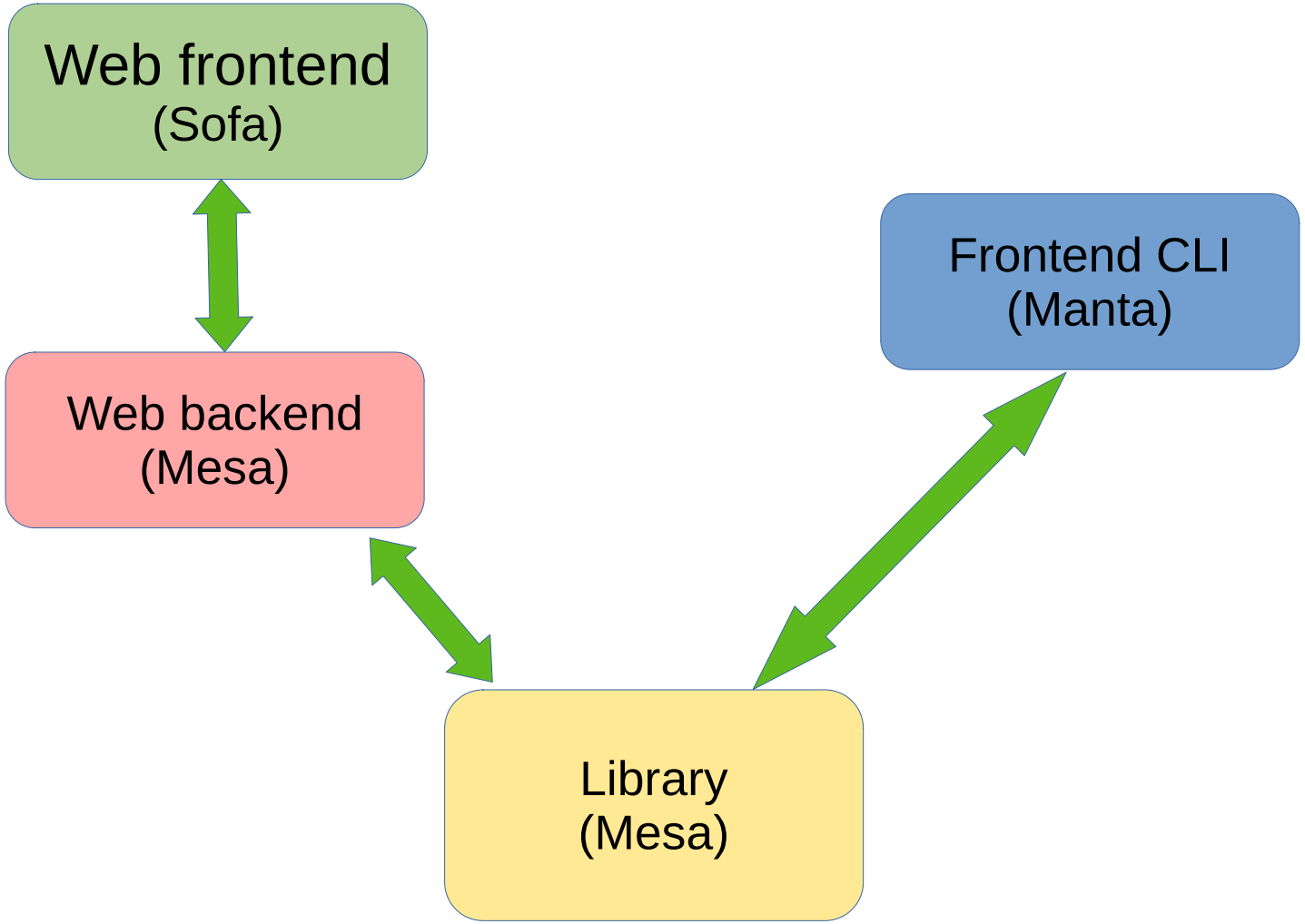
Current limitations

- Can’t create BOS session template for UANs
- Can’t create repos in Shasta git repo

Future Work



Future Work

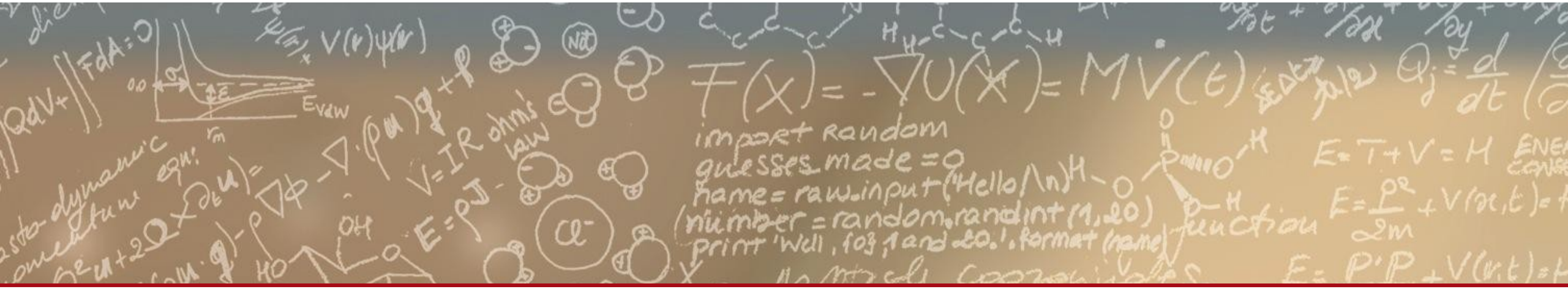

Web frontend
(Sofa)
Web backend
(Mesa)
Frontend CLI
(Manta)
Library
(Mesa)

Future Work

- Logging
- Security
- Scripting
- Rust library (mesa)
- Web backend (tabla)
- Create a web frontend (sofa)
- Delete/clean CSM data
- Terraform providers?
- Multi-region management?
- Authorization?
- Portability/Accessibility?

Acknowledge

- Mark Klein
- Miguel Gila
- Peter Tiernan
- Hussein Harake
- Derek Feichtinger
- Hussein Nasser
- Marc Caubet
- Hans-Nikolai Viessmann
- Elsa Germann



Thank you for your attention.

Manuel Sopena Ballesteros (manuel.sopena@cscs.ch)

manta src: **https://github.com/eth-cscs/manta**