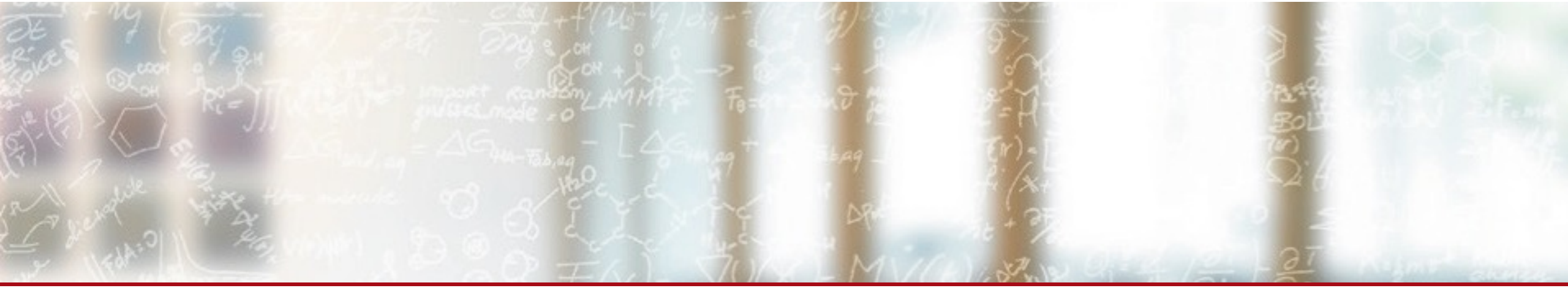
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



A Look at Some Scalable HPC Configuration Management Tools

HPC-CH

Miguel Gila, CSCS

May 19, 2022

Disclaimer

- I’m covering only the system-related part

- I won’t go very deep technical details, but I’m happy to answer questions or show how things are done

A bit of history

Early 2010's: Monte Rosa/Todi and Piz Daint/Piz Dora

- Back then we used a mix of tools, some standard, some proprietary (and weird):
  - High-density Cray supercomputers:
    - **XTopview** was a system based on symlinks (believe it or not) that allowed engineers to configure systems based on groups or hardware addresses/names.
    - It was simple to operate, and didn't provide a lot of functionality
  - High-density IBM supercomputers:
    - Set of scripts to customize initramfs and OS image, pretty much like XTopview
  - Low-density generic systems:
    - RCS + SSH
    - CFengine2
    - Puppet

Late 2010's: Piz Daint+

- At the time Cray replaced their proprietary CM tooling with Ansible… but in a very particular way:
  - Image management is done with a proprietary tool based on chroot
  - Configuration was to each node at boot time and launched "locally" on each node
  - This is fast but can be troublesome if thousands of nodes are pulling RPMs at boot

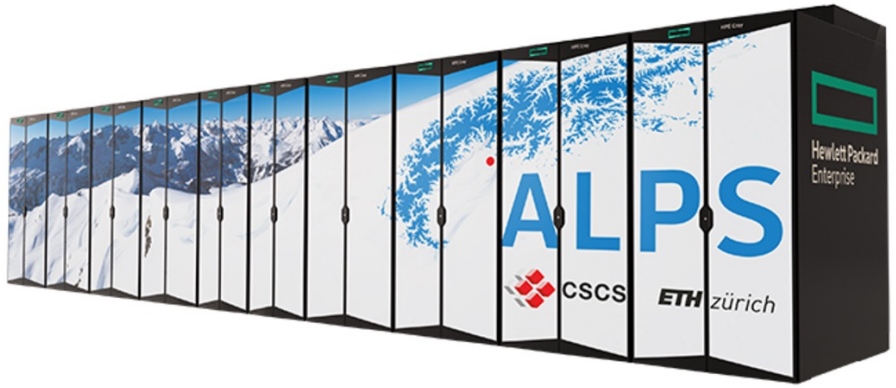
- Non-high-density Cray systems (CS line) evolved
  - From Cray ACE (Advanced Cluster Engine) | image-based
  - To Bright CM | image-based

- We used a variety of tools on other non-Cray systems:
  - Puppet
  - Ansible
  - XCAT

Early 2020's: Alps

- With the HPE Cray EX systems, HPE shifted to ~standard tooling
  - Images could be baked using standard tools (now Kiwi-NG, eventually Packer)
  - Ansible is applied on thousands of nodes (or images) using a self-scaling set of Kubernetes deployments
  - Eventually *cloud-init* will be used to do some basic configuration at boot time (hostname, etc.)

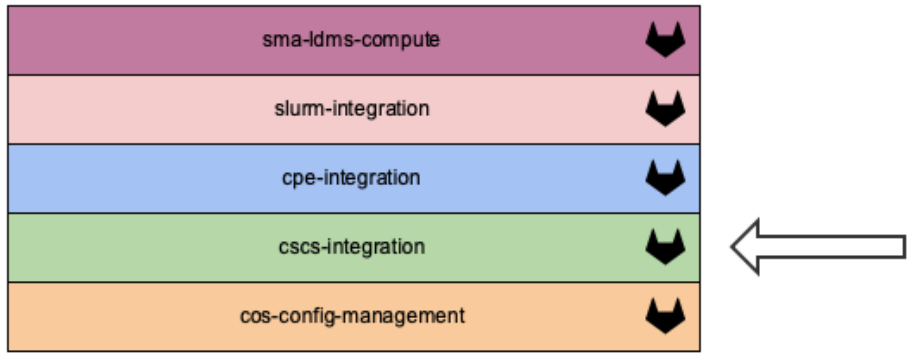
- We adapted the tooling to
  - build immutable images with Ansible
  - configure nodes with Ansible upon boot



- On non HPE Cray systems, we are taking a similar approach using Packer and Ansible

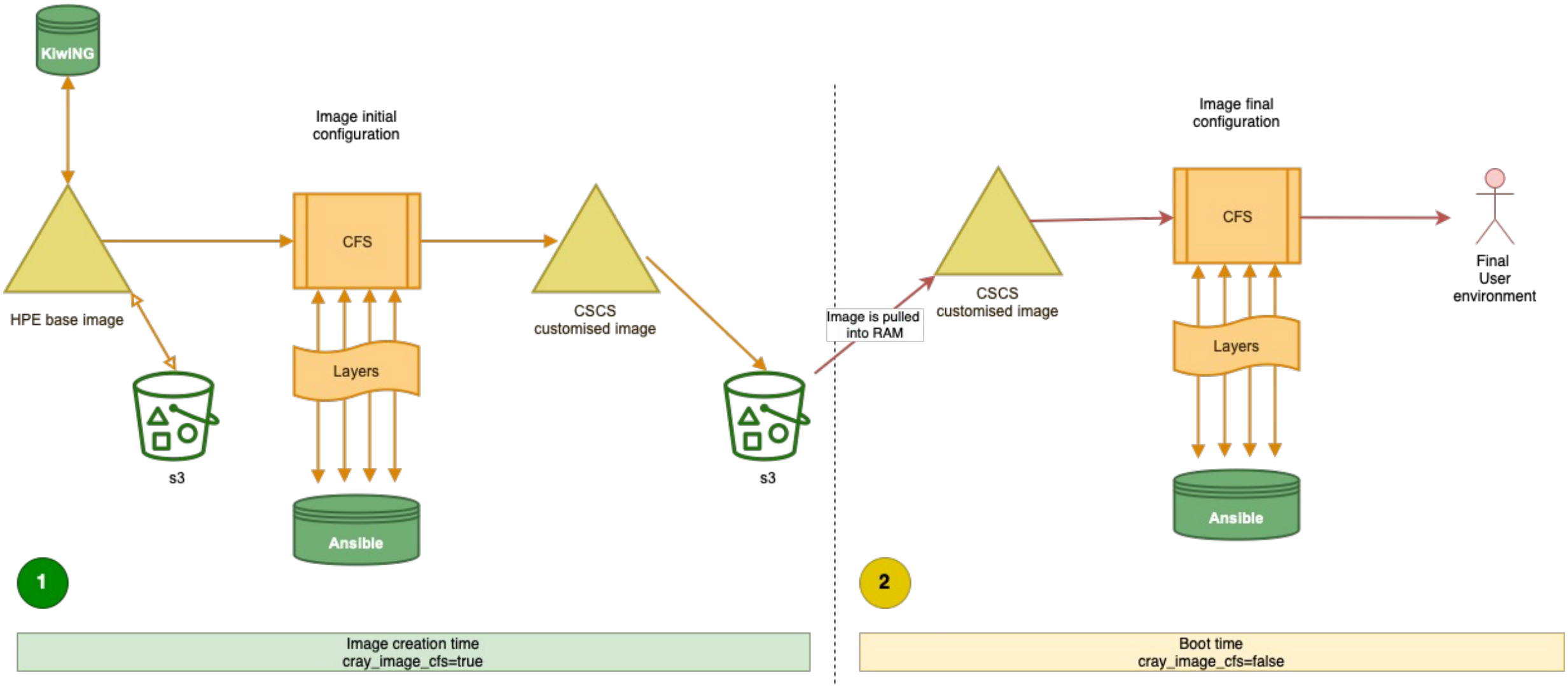
Image creation and node configuration in Alps

- Ansible code and cluster values are divided in layers that are applied sequentially:
  - COS (Cray OS)
  - Slurm
  - CSCS (common roles)
  - SMA (infrastructure telemetry)
  - WLCG



- Each layer can be owned by a different team with different scope and responsibilities:
  - Role of platform admin
  - Role of tenant admin
  - Role of infrastructure admin

Alps CM workflow



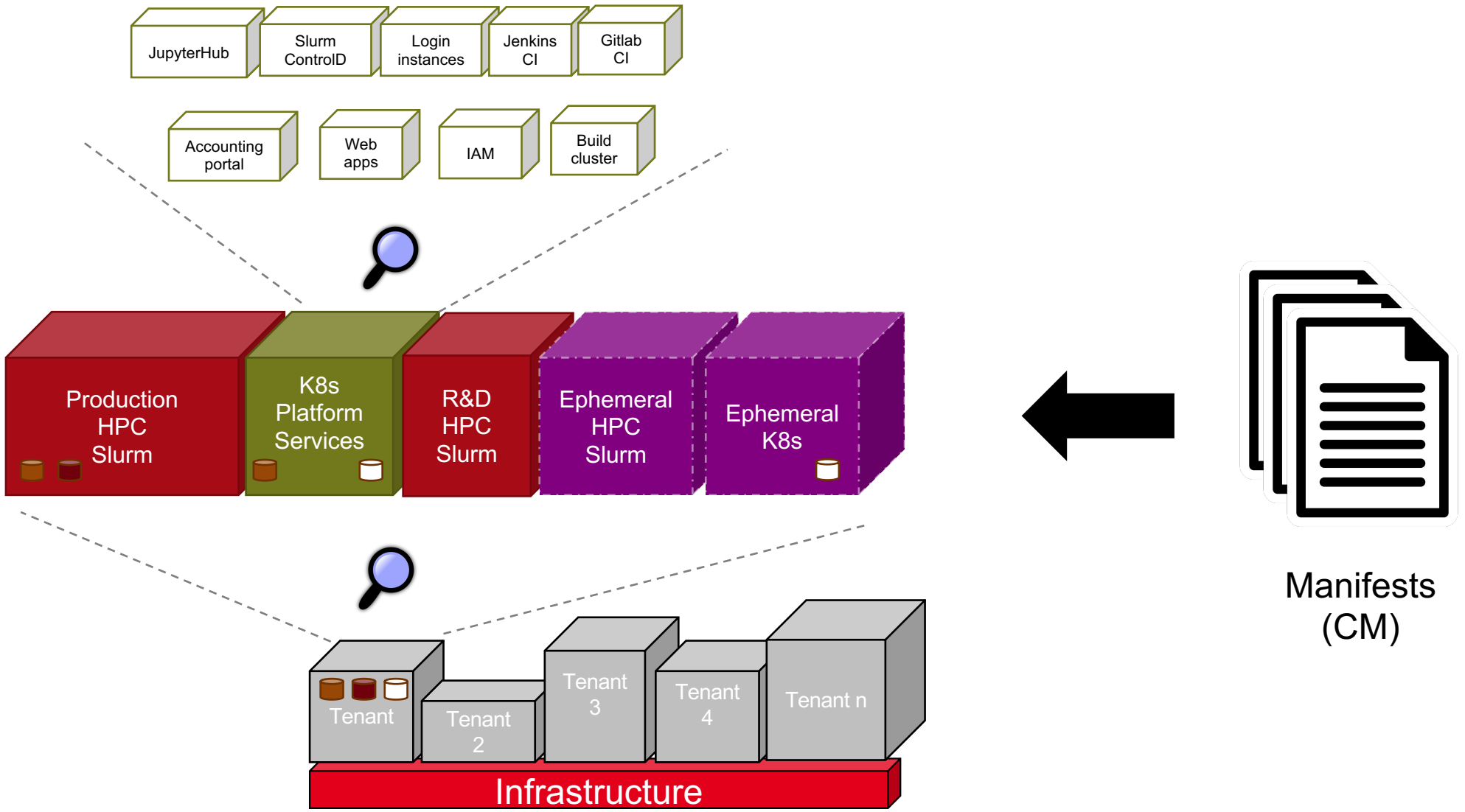
Kubernetes

- We configure our Kubernetes clusters in multiple ways, but the idea is to converge towards a similar method:
  - Bake node images with Packer and Ansible
  - Configure nodes with cloud-init (hostname, etc.) and Ansible (everything, including k8s)

- Configuring platforms (applications and services) using manifests (HELM) is great. For multi-HELM platforms, we **could** use https://github.com/Cray-HPE/loftsman. Q: Are here additional, better tools? Terraform?

- Configuration management one layer up: what about services that are multi-cluster-type (Slurm/Torque + Kubernetes-based)

With multi-tenancy things can get really complicated



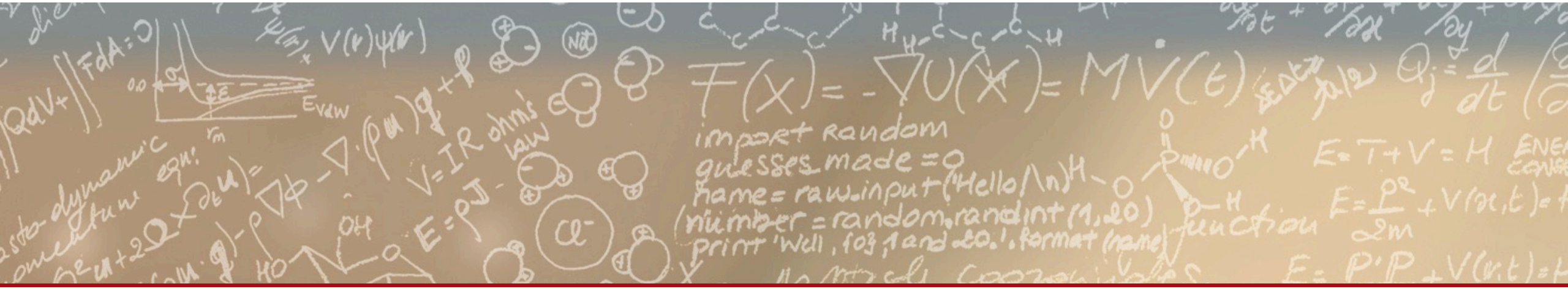
Looking forward, personal conclusion

Looking forward

- Configuring HPC systems at scale is hard
- Configuring platforms in Kubernetes can be easy… or very hard, depending on needs
- Configuring multi-tenant systems at scale is really, really hard
- Ansible doesn't scale very well past a few hundreds of nodes
- A lot of the developments in this area seem to be converging towards the same technology choices done by the cloud providers:
  - Bake complete images using the CM tool of your choice
  - Set host-related configs at boot with cloud-init

I personally think that…

- Any modern HPC or non-HPC configuration management should be done using standard tools
- Using manifests to configure environments (like on Kubernetes) is the way to go
- Revision control and release management are super important, to me it's the only way to manage rolling-updates on large systems
  - Git-ops and CI/CD workflows
  - RPMs, container images and immutable images
- For a centre like CSCS the toolset should be unique and as simple as possible
- Start small and grow slowly. Test everything, even stupidly small changes



Thank you for your attention.

Backup slides

Cray EX – Multi-tenancy roles