Re-imagining management methods for distributed and clustered systems

Kraken + Layercake

Video: LA-UR-21-24586

Slides: LA-UR-21-24563

J. Lowell Wofford
Kevin Pelzel
Travis Cotton

Los Alamos National Laboratory

Los Alamos NATIONAL LABORATORY

NNSA National Nuclear Security Administration

TRIAD National Security, LLC

Who We Are

- J. Lowell Wofford (presenter)
  - Scientist, LANL HPC-Design.
- Kevin Pelzel (co-author)
  - Scientist, LANL HPC-Environments.
- Travis Cotton (co-author)
  - Scientist, LANL HPC-Systems.

The Problem

People are building lots of clusters…

- High-performance computing
- VM clusters & Private (or Public) Clouds
- Container Orchestration (kube, shift, swarm, oh my!)
- Storage & data clusters (ceph, mongo, etcd,...)
- ML/AI & Data analytics
- Application & Service clusters

People are building clusters out of lots of things…

The hardware landscape is evolving:

- Different kinds of processors
- Different kinds of baseboard management
- Different kinds of boot processes
- Different kinds of networks
- …

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

Home

Technologies

Los Alamos Takes New HPE Apollo 80 System on a Test Drive

September 1, 2020

LOS ALAMOS, N.M., Sept. 1, 2020— Los Alamos National Laboratory recently began using HPE Apollo 80 Systems from Hewlett Packard Enterprise (HPE) that feature F some of the most demanding p

Scalable Clusters Make HPC R&D Easy as Raspberry Pi

November 13, 2017

LOS ALAMOS, N.M., Nov. 13, 2017 — A quest to help the systems s community work on very large supercomputers without having to actu on them has spawned an affordable, scalable system using thousand inexpensive Raspberry Pi nodes. It brings a powerful high-performanc computing testbed to system-software developers, researchers, and who lack machine time on the world's fastest supercomputers.

"It's not like you can keep a petascale machine around for R&D work scalable systems software," said Gary Grider, leader of the High Perf Computing Division at Los Alamos National Laboratory, home of the T supercomputer. "The Raspberry Pi modules let developers figure out write this software and get it to work reliably without having a dedicate testbed of the same size, which would cost a quarter billion dollars an megawatts of electricity."

Grider conceived the ne solution to provide the s software community wit inexpensive testbed of s scale to the largest supercomputers being d Designed and built by BitScope and distributed by SICORP, the Pi Cluster Modules enable developers to build, scale and test software before launching it on Trinity, Crossroads, and other

Nvidia Serves Up Its First Arm Datacenter CPU 'Grace' During Kitchen Keynote

By Tiffany Trader

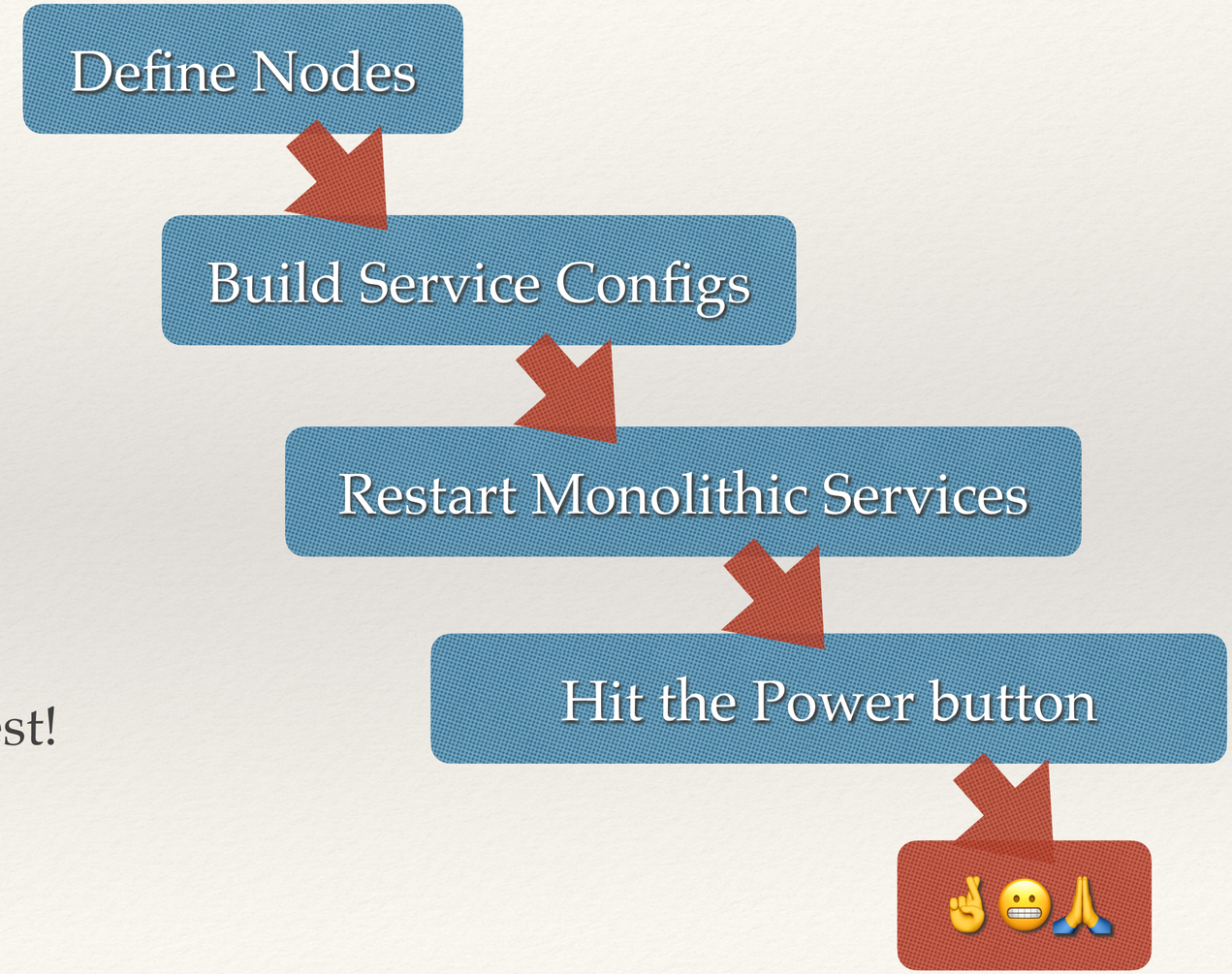
April 12, 2021

Today at Nvidia's annual spring GPU Technology Conference (GTC), held virtually once more due to the pandemic, the company unveiled its first ever Arm-based CPU, called Grace in honor of the famous American programmer Grace Hopper. The announcement of the new Arm CPU follows Nvidia's 2019 declaration of intent to fully embrace Arm and its September 2020 bid to acquire Arm for $40 billion.

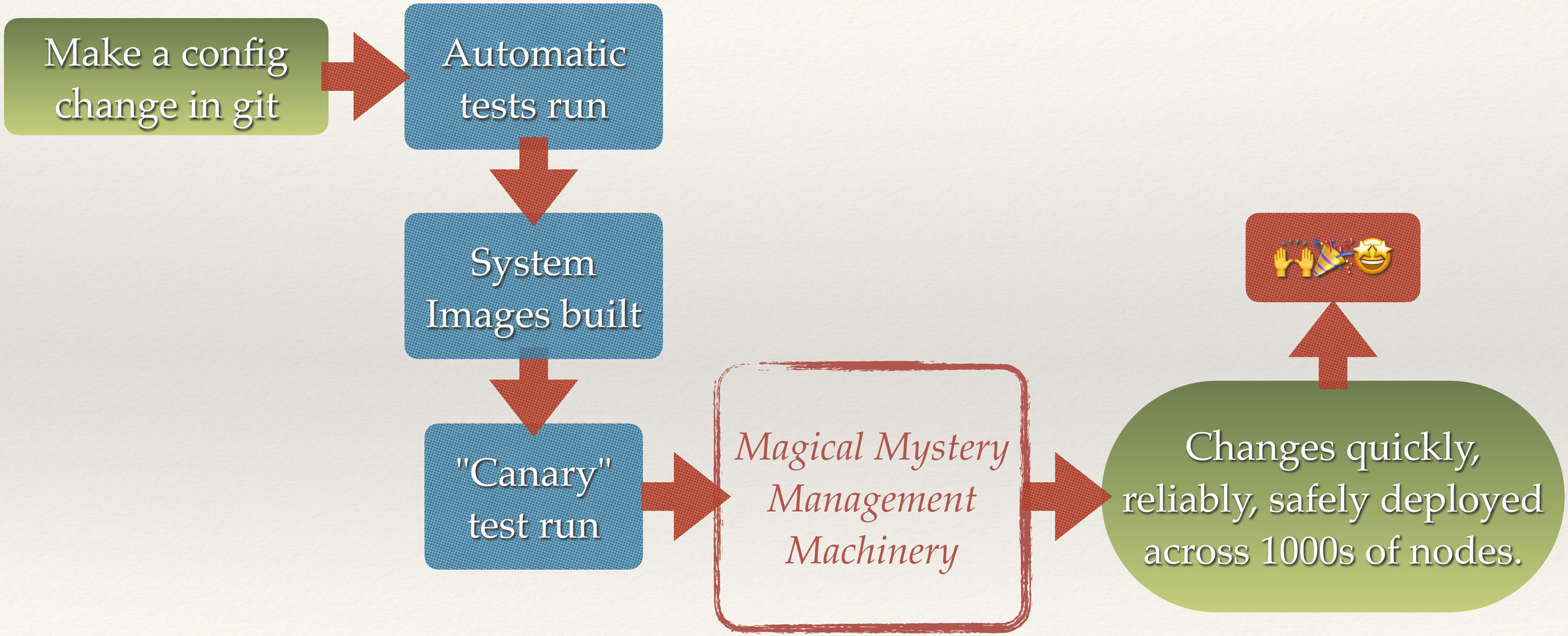
Grace is expected to debut in 2023 with two HPC centers leading the way. The Swiss National Supercomputing Centre (CSCS) and the U.S. Department of Energy's Los Alamos National Laboratory are the first to announce plans to build Grace-powered supercomputers in partnership with HPE and Nvidia.

But how we mange them hasn't changed.

- Maintain some configuration store (probably in a database).
- Use configuration store to create misc. configuration files.
- Use configuration files to control classic, monolithic services.
- Power everything on and hope for the best!



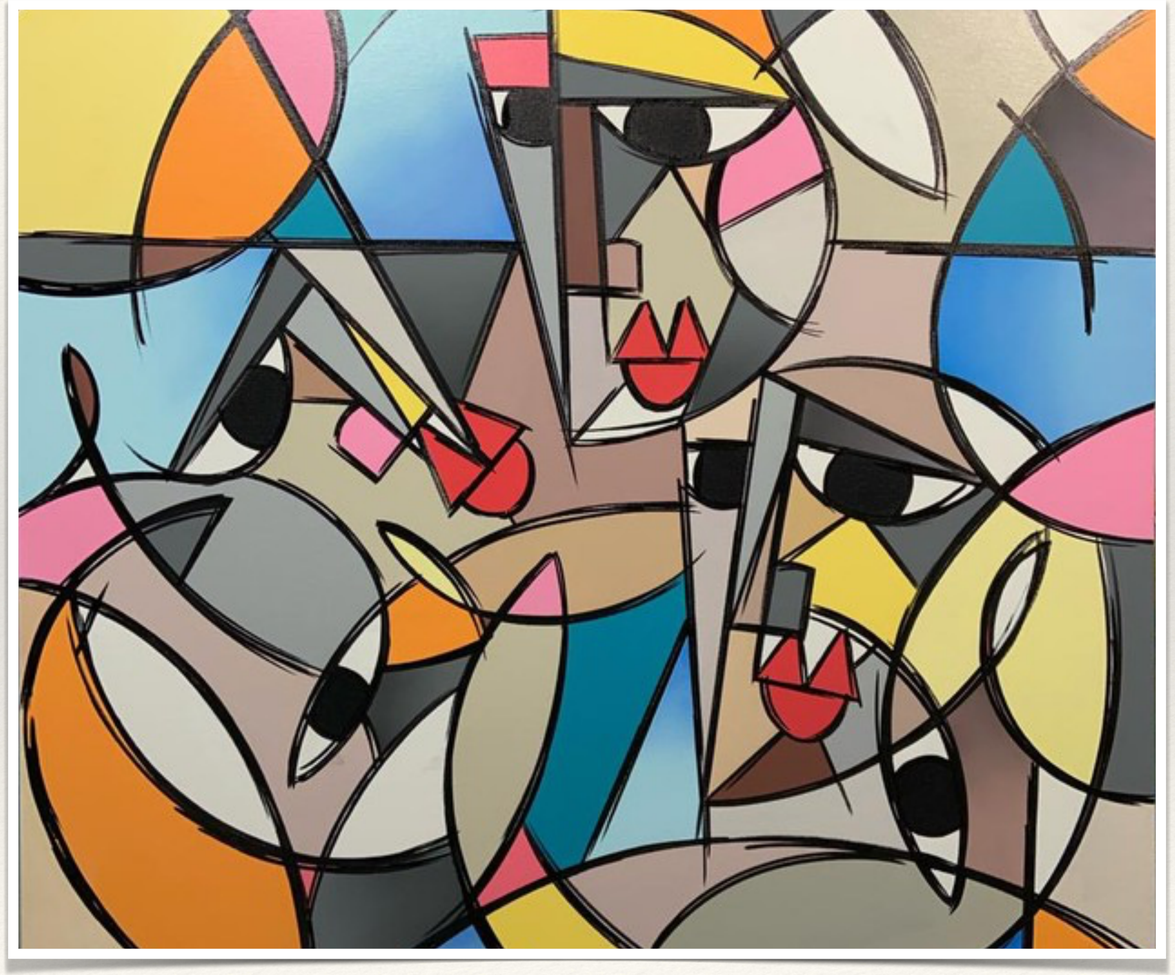
Imagine a world…



A Re-Think in Three Inspirations

Inspiration #1: Container Orchestration

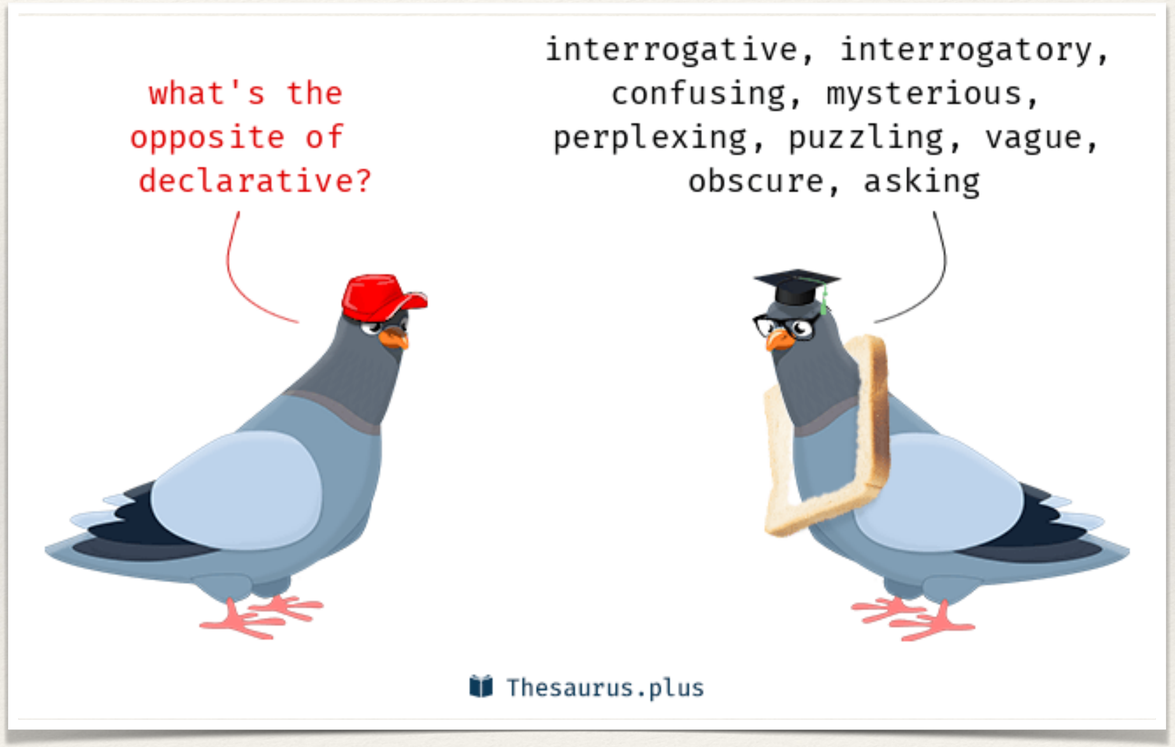
- **Continuous Enforcement:** System state should be enforced continuously.
  - Example: If a service gets out-of-line, detect it and fix it.
- **Hardware Abstraction:** Components should be abstracted from the services they provide.
  - Example: A pod should care what it provides, not where it runs.



Abstraction Faces by Julien Raynaud

Inspiration #2: Configuration Management

- **Centralized Configuration:** Centralize and organize system configuration.
  - Example: CM states should enforce local system states.
- **Declarative Administration:** say what you want, not how to get it.
  - Example: Describe the functions a system should have (e.g. "roles"), not how to create those functions.
  - Bonus: naturally makes systems idempotent.



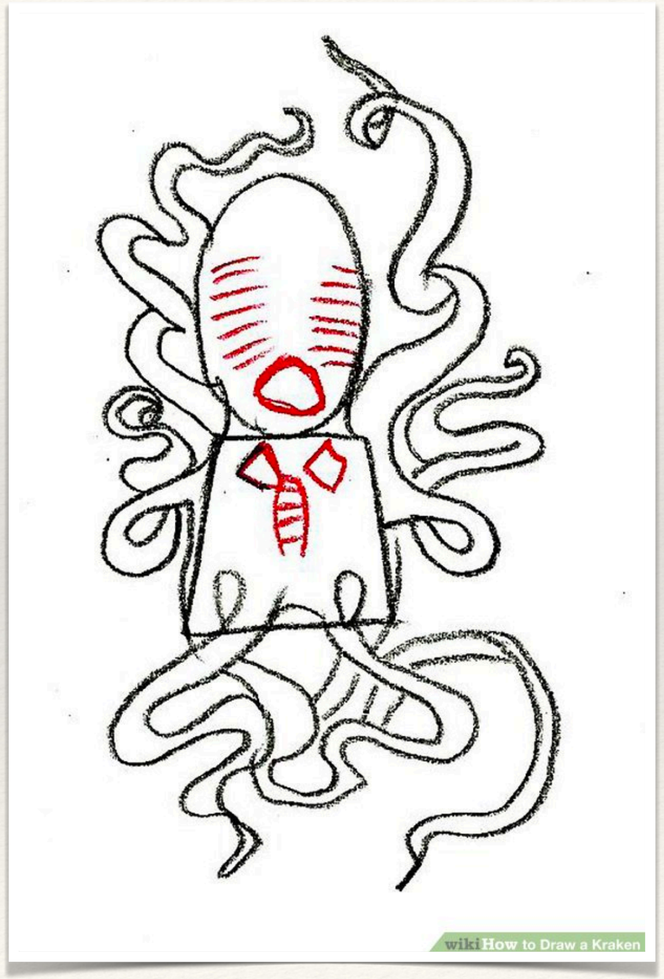
Inspiration #3: API-Driven Design

- **Layer Abstractions:** Build systems in layers.
  - Example: Define clear boundaries in functionality, abstract outside details.
- **Well-defined Interactions:** Prescribe interactions, not actions.
  - Example: One component shouldn't tell another how to do its job, only how to interact.

Illustration by Nayane de Souza Hablitzel

A rough sketch

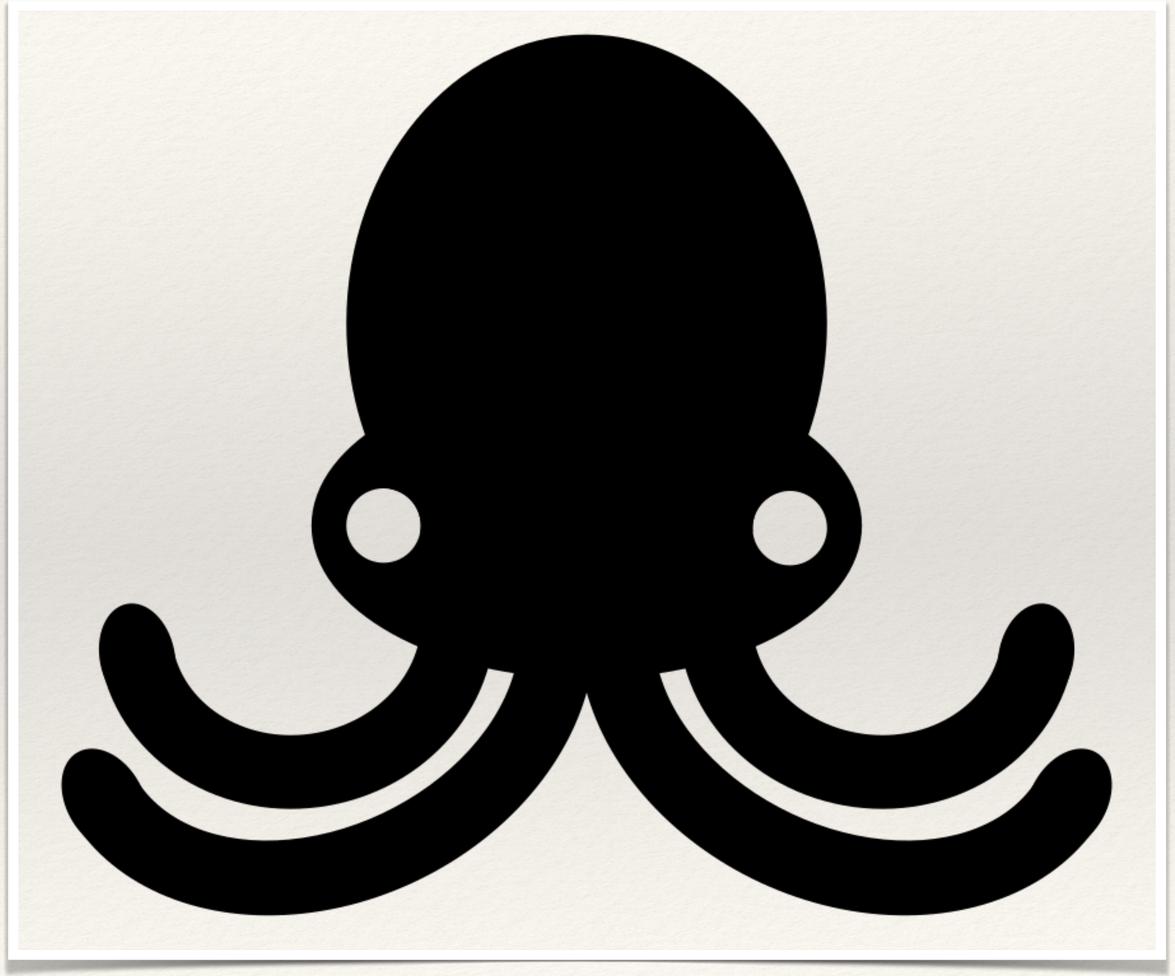
- **Centralize** where system state lives.
- Configure systems **declaratively**.
- Enforce states **continuously**.
- **Abstract** hardware from its uses.
- Clearly delineate **layers** of functionality.
- Provide simple APIs to allow components to **interact**.



Kraken + Layercake

The Kraken Framework

- Kraken is a framework for building automation tools that are:
  - **Distributed**: Able to manage automation workflows across many systems.
  - **Scalable**: Able to scale to the demands of workloads like HPC.
  - **Declarative**: Say what, not how.
  - **Modular**: Can create tools for a variety of use-cases based on included modules.
  - **Always-on**: Continuously enforces states across a distributed system.



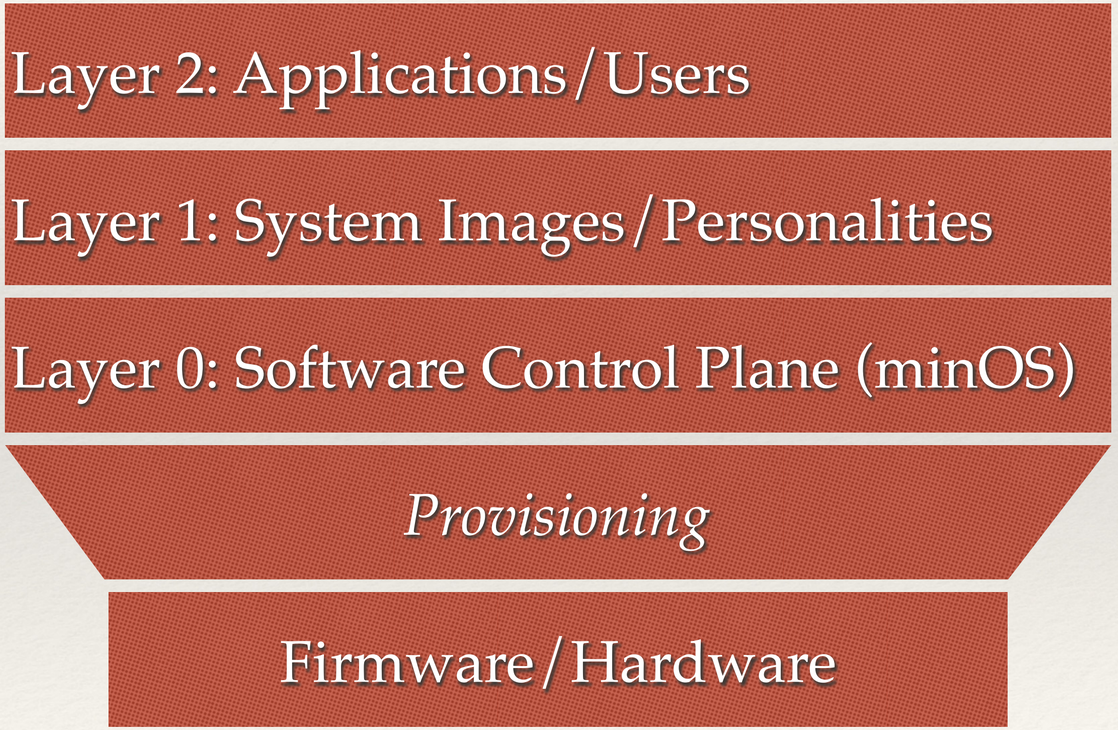
https://kraken-hpc.io

Layercake: The basics

- Written in Go; Based on Kraken.
- Inherits from Kraken:
  - Declarative
  - Continuous state enforcement
  - 100% Modular
- Adds:
  - System abstraction layers (hence the name)
  - Centralized state; stateless nodes.

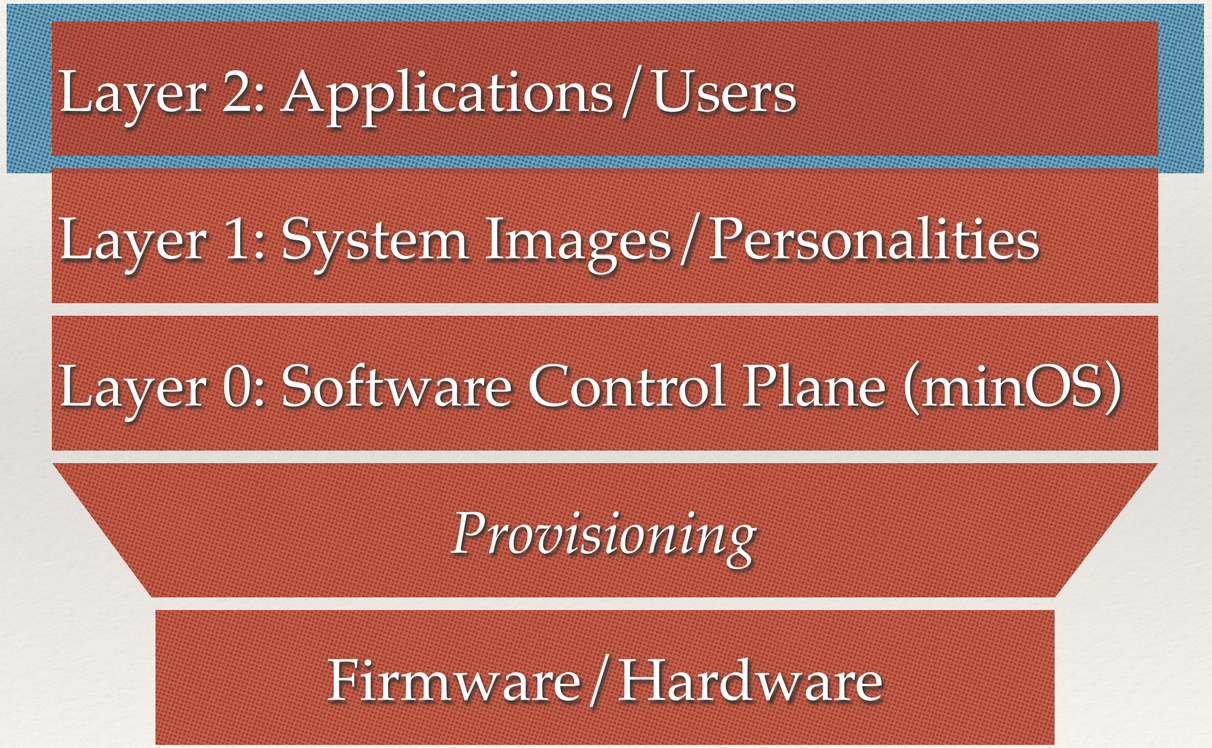
Systems as Layers

- Think of systems in layers.
- Changing a higher layer affect a lower layer.
  - E.g. Change Applications without changing System Images.
  - E.g. Change System Images without changing Software control plane.
- Layers and their interactions should be modular.



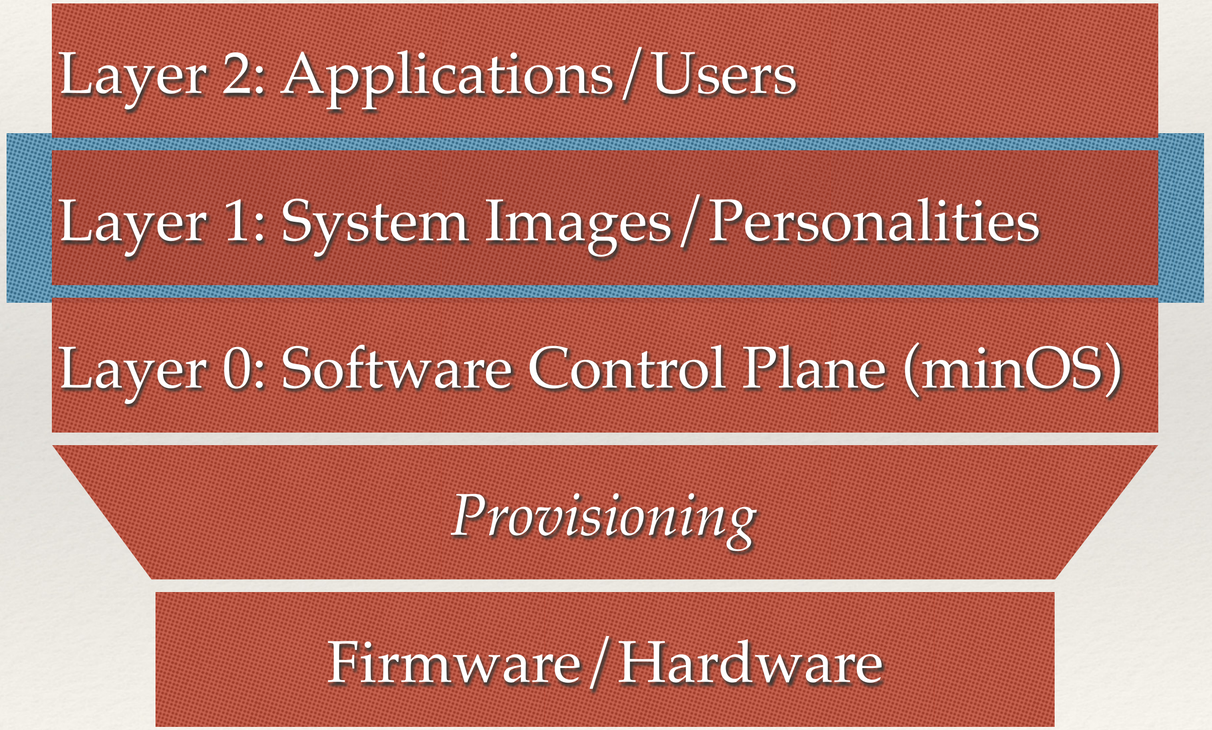
Application & Users

- Focused on Applications, users, and their needs, not systems.
- Keep applications containerized & abstracted from system images as much as possible.
- Particulars vary significantly based on the system type.



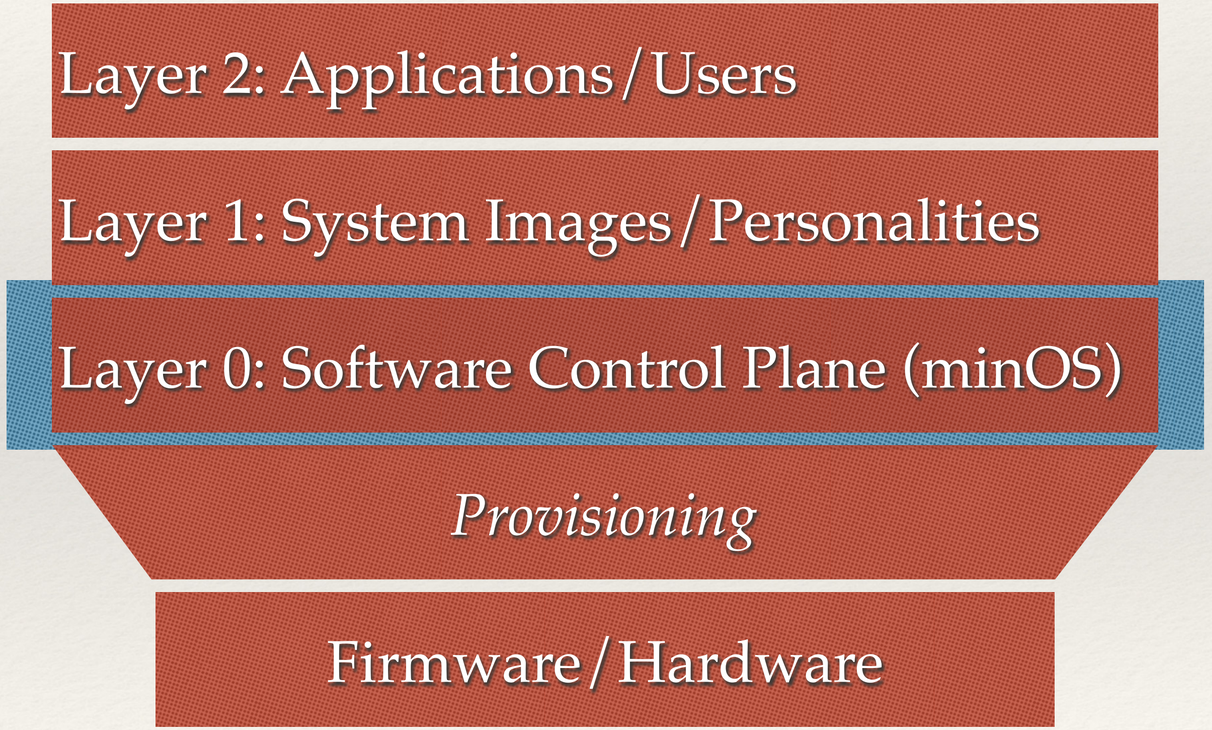
System Images & Personalities

- System Images determine available functionality for Layer 2.
  - An HPC cluster might run Slurm.
  - A Container Orchestration might run k3s.
- Don't be picky about image formats & specifications.
  - Just tell us where to find it and how to run it.



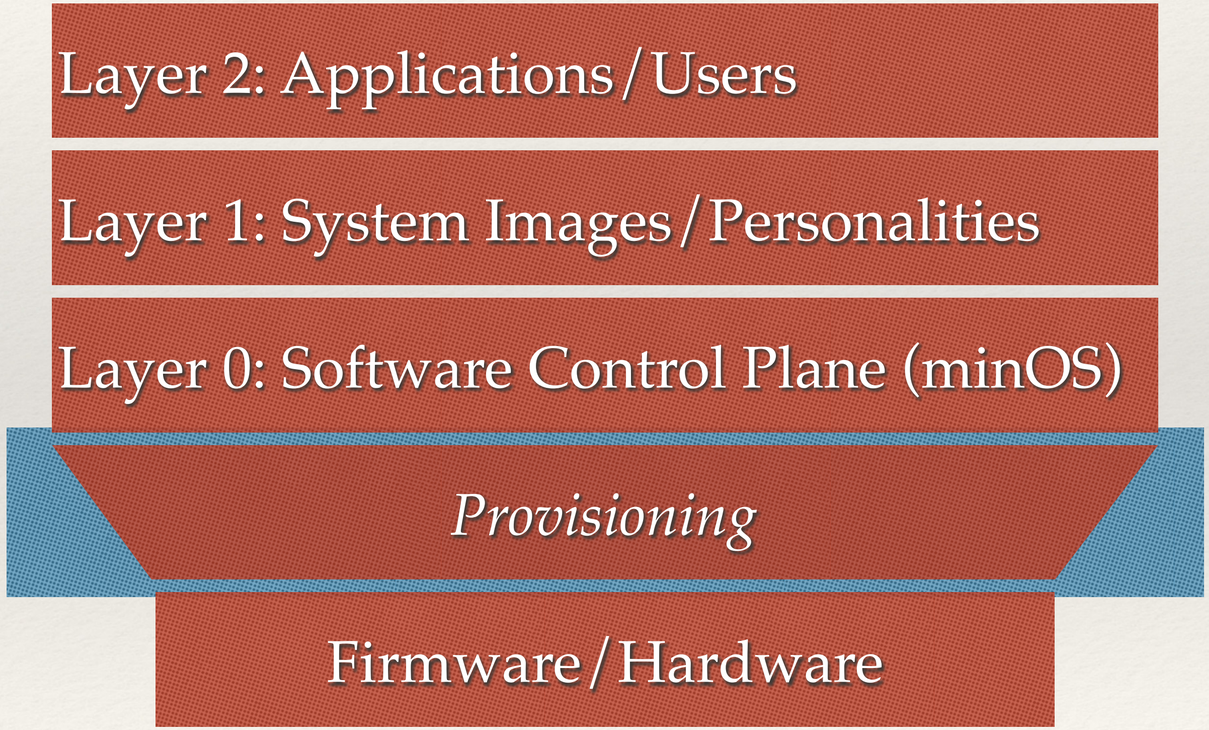
The Minimal OS (minOS)

- Software abstraction layer between system images (personalities) & hardware.
  - The system image should have to worry about how it was provisioned.
  - Changing a system image/personalities shouldn't require hardware interaction.
- In general: Try not to re-provision.



Provisioning

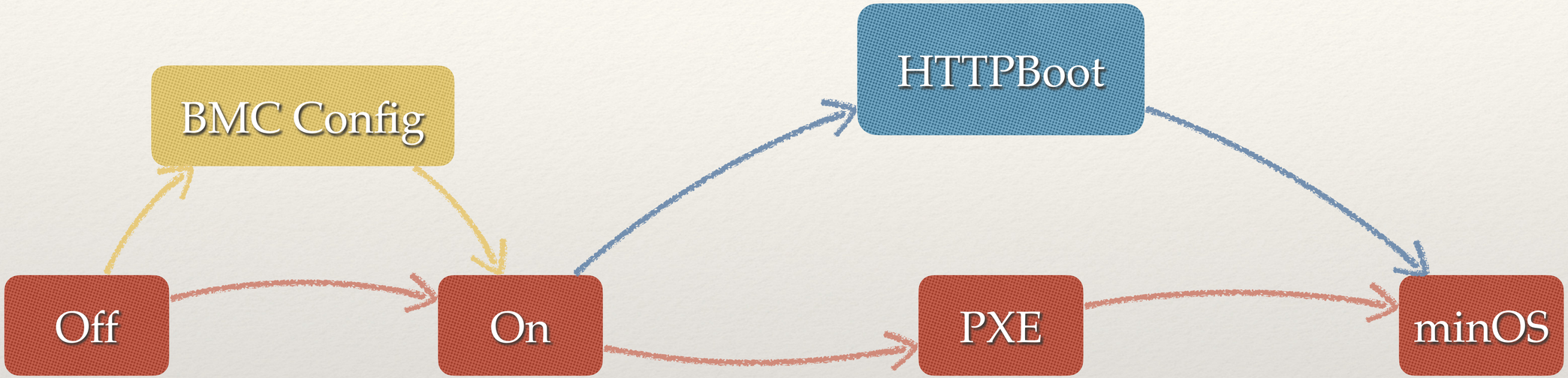
- We don't get to dictate hardware & firmware.
  - (well, maybe firmware...)
- Try not to make assumptions about how to get to Layer 0.
- Allow for many different paths to reach the minOS.



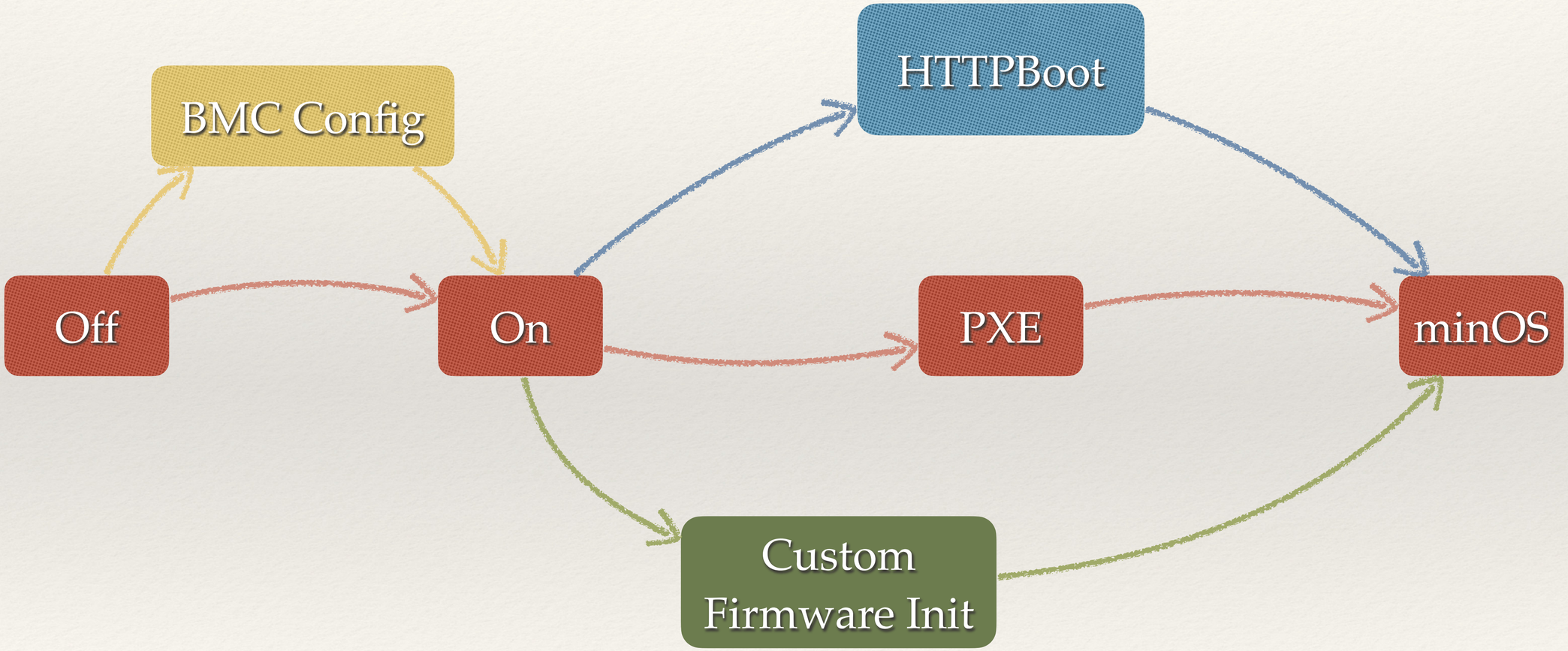
Keeping things modular

Keeping things modular

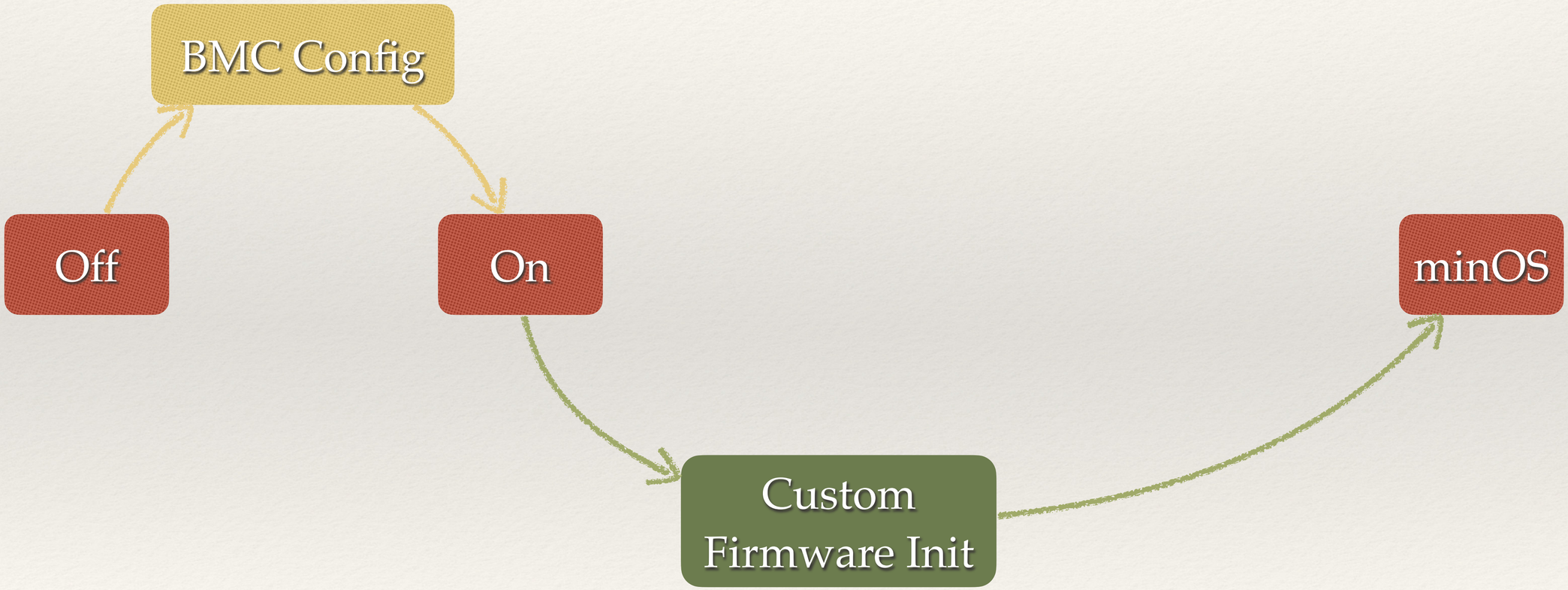
Keeping things modular



Keeping things modular



Take the path you need…



Reference implementation

- Power control through multiple mechanisms for many physical & virtual systems.
- Standard, integrated PXE stack.
  - All required services in one binary.
  - Several alt-stacks in dev.
- All-Golang minOS (u-root* based).
  - Tiny (~10M!), yet powerful.
- System image building via special tooling + Dockerfile.
  - But we can handle many different image formats.
- System image loading via ImageAPI service & Ceph RBD.
  - Loads system images privileged containers.
  - Supports everything from microservice containers to systemd.

* https://u-root.org

Demo time!

Demo

Zero to minOS

- Installing Layercake
- Running Layercake
- Node definitions
- Boot a single node to minOS

```
[root@moonlight ~]# banner ml001 console

#     #  #         ####    ####     #
##   ##  #        #   ##  #   ##   ##
# # # #  #        #  # #  #  # #  # #
#  #  #  #        # #  #  # #  #    #
#     #  #        ##   #  ##   #    #
#     #  #        #    #  #    #    #
#     #  #######   ####    ####   #####


 #####   #######  #     #   #####   #######  #       #######
#     #  #     #  ##    #  #     #  #     #  #       #
#        #     #  # #   #  #        #     #  #       #
#        #     #  #  #  #   #####   #     #  #       #####
#        #     #  #   # #        #  #     #  #       #
#     #  #     #  #    ##  #     #  #     #  #       #
 #####   #######  #     #   #####   #######  #######  #######

[root@moonlight ~]# telnet gb001 7001
Trying 192.168.0.1...
Connected to gb001.
Escape character is '^]'.
```

```
[root@moonlight ~]# banner Layercake

#         #    #    #  #######  ######   #####     #    #    #  #######
#        # #    #  #   #        #     #  #     #   # #   #  #   #
#       #   #    # #    #        #     #  #         #   #  # #    #
#       #     #    #    #####    ######   #  ####   #     ##     #####
#       #######    #    #        #   #    #     #  #######  #    #
#       #     #    #    #        #    #   #     #  #     #  #  #   #
#######  #     #    #    #######  #     #   #####   #     #  #    #  #######

[root@moonlight ~]#
```

```
#     #  ###  ######   #######  #######
#     #   #   #     #  #        #     #
#     #   #   #     #  #        #     #
#     #   #   #     #  #####    #     #
 #   #    #   #     #  #        #     #
  # #     #   #     #  #        #     #
   #     ###  ######   #######  #######

[root@moonlight ~]#
```

```
[root@moonlight ~]# banner working window

#     #  #######  ######   #    #  ###  #     #   #####
#  #  #  #        #     #  #    #   #   #     #  #     #
#  #  #  #        #     #  #  #    #   #  #  #  #
#  #  #  #        #  #####  ###     #   #  # #  #  #  ###
#  #  #  #        #  #   #    #  #     #   #   ## #     #
#  #  #  #        #  #    #   #   #    #   #    ## #     #
 ## ##   #######  #     #  #    #  ###  #     #   #####

#     #  ###  #     #  ######   #######  #     #
#  #  #   #   ##    #  #     #  #     #  #  #  #
#  #  #   #   # #   #  #     #  #     #  #  #  #
#  #  #   #   #  #  #  #     #  #     #  #  #  #
#  #  #   #   #   # #  #     #  #     #  #  #  #
#  #  #   #   #    ##  #     #  #     #  #  #  #
 ## ##   ###  #     #  ######   #######   ## ##

[root@moonlight ~]#
```

Video Link: http://bit.ly/lisa21_video_wofford_demo1

Demo

Zero to Cluster

- Layercake in k3s
- The Kraken Dashboard
- Layercake + Ansible
- Boot entire Cluster to system images
- Self-healing

```
[root@moonlight ~]# banner ML001

#     #  #         ####    ####     #
##   ##  #        #   ##  #   ##   ##
# # # #  #        #  # #  #  # #  # #
#  #  #  #        # #  #  # #  #    #
#     #  #        ##   #  ##   #    #
#     #  #        #    #  #    #    #
#     #  #######   ####    ####   #####

[root@moonlight ~]# telnet gb001 7001
Trying 192.168.0.1...
Connected to gb001.
Escape character is '^]'.
```

```
[root@moonlight layerk8s]# banner layercake + kubernetes

#         #    #    #  #######  ######    #####     #    #    #  #######
#        # #    #   #   #       #     #  #     #   # #   #   #   #
#       #   #    # #    #       #     #  #         #   #  #  #    #
#       #     #    #    #####   ######   #        #     #  ##     #####
#       #######    #    #       #   #    #        #######  #  #   #
#       #     #    #    #       #    #   #     #  #     #  #   #  #
#######  #     #    #    #######  #     #   #####   #     #  #    #  #######


   #
   #
 #####
   #
   #



#    #  #     #  ######   #######  ######   #     #  #######  #######  #######   #####
#   #   #     #  #     #  #        #     #  ##    #  #           #     #        #     #
#  #    #     #  #     #  #        #     #  # #   #  #           #     #        #
###     #     #  ######   #####    ######   #  #  #  #####       #     #####     #####
#  #    #     #  #        #        #   #    #   # #  #           #     #              #
#   #   #     #  #        #        #    #   #    ##  #           #     #        #     #
#    #   #####   #        #######  #     #  #     #  #######     #     #######   #####

[root@moonlight layerk8s]#
```

Video Link: http://bit.ly/lisa21_video_wofford_demo2

Demo

System Images to HPC

- Build a Slurm image
- Roll an update
- See that the update is stateful
- Run a (very simple) Slurm job

```
Every 1.0s: sinfo                                moonlight: Mon May 10 22:18:24 2021

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite    130  down* ml[001-130]
```

```
[root@moonlight layerk8s]# banner layercake + slurm

#          #     #     #  #######  ######    #####      #     #    #  #######
#         # #     #   #   #        #     #  #     #    # #    #   #   #
#        #   #     # #    #        #     #  #          #   #   #  #    #
#       #     #     #     #####    ######   #         #     #  ###    #####
#       #######     #     #        #   #    #         #######  #  #    #
#       #     #     #     #        #    #   #     #   #     #  #   #   #
#######  #     #     #     #######  #     #   #####   #     #  #    #  #######


   #
   #
 #####
   #
   #


 #####   #        #     #  ######   #     #
#     #  #        #     #  #     #  ##   ##
#        #        #     #  #     #  # # # #
 #####   #        #     #  ######   #  #  #
      #  #        #     #  #   #    #     #
#     #  #        #     #  #    #   #     #
 #####   #######   #####   #     #  #     #

[root@moonlight layerk8s]#
```

Video Link: http://bit.ly/lisa21_video_wofford_demo3

Demo

But wait…

…we don't all want HPC!

- Assign a new image to specific nodes
- Roll nodes into k3s agents

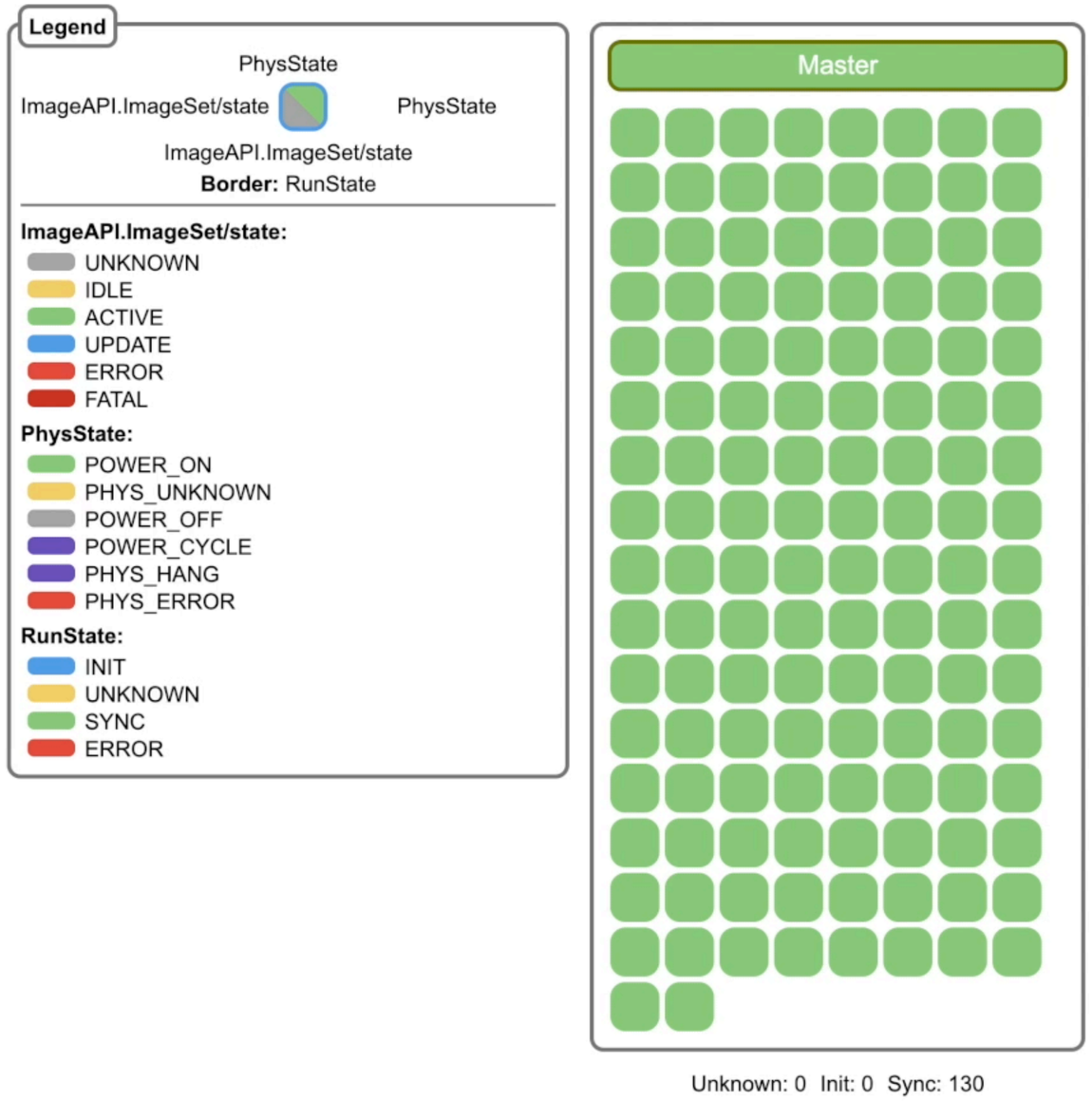
```
Every 1.0s: sinfo                                  moonlight: Mon May 10 23:20:44 2021

PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite    130   idle ml[001-130]
```

```
Every 2.0s: kubectl get nodes                      moonlight: Mon May 10 23:20:43 2021

NAME        STATUS   ROLES                  AGE    VERSION
moonlight   Ready    control-plane,master   3d8h   v1.20.6+k3s1
```

```
[root@moonlight layerk8s]# banner beyond hpc

######   #######  #     #  #######  #     #  ######
#     #  #        #   #   #     #  ##    #  #     #
#     #  #         # #    #     #  # #   #  #     #
######   #####      #     #     #  #  #  #  #     #
#     #  #          #     #     #  #   # #  #     #
#     #  #          #     #     #  #    ##  #     #
######   #######    #     #######  #     #  ######


#     #  ######    #####
#     #  #     #  #     #
#     #  #     #  #
#######  ######   #
#     #  #        #
#     #  #        #     #
#     #  #         #####

[root@moonlight layerk8s]#
```

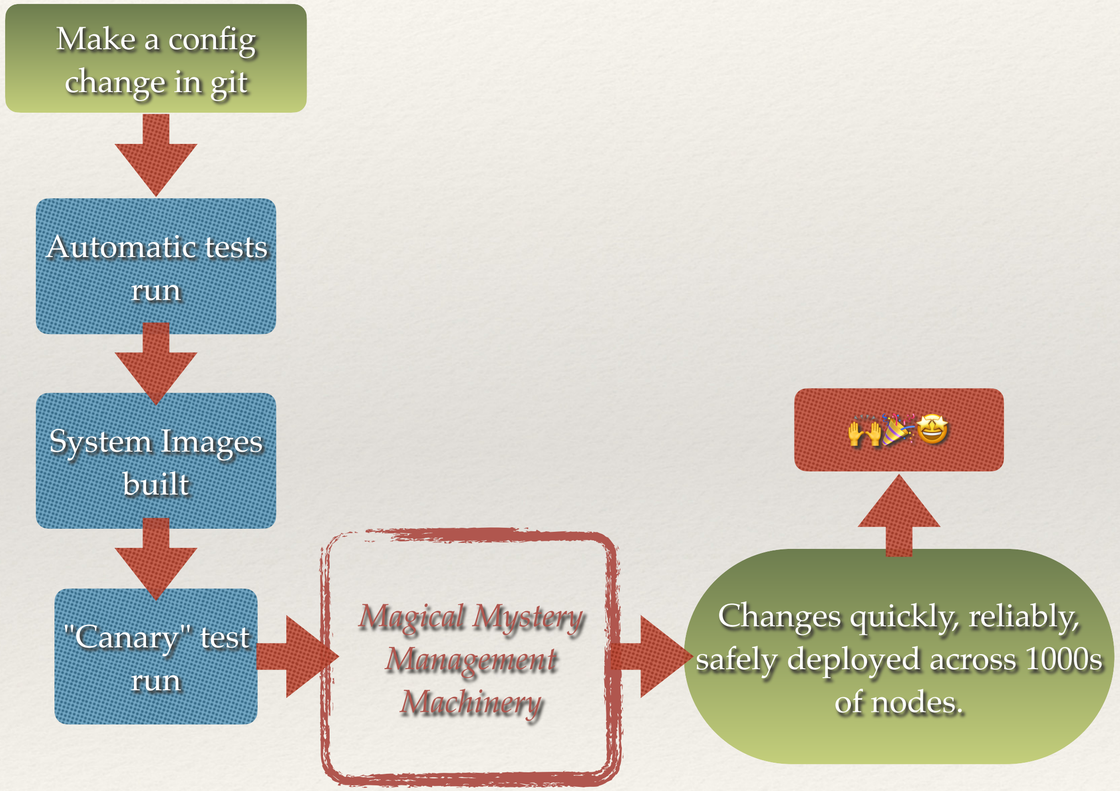
Video Link: http://bit.ly/lisa21_video_wofford_demo4

Kraken



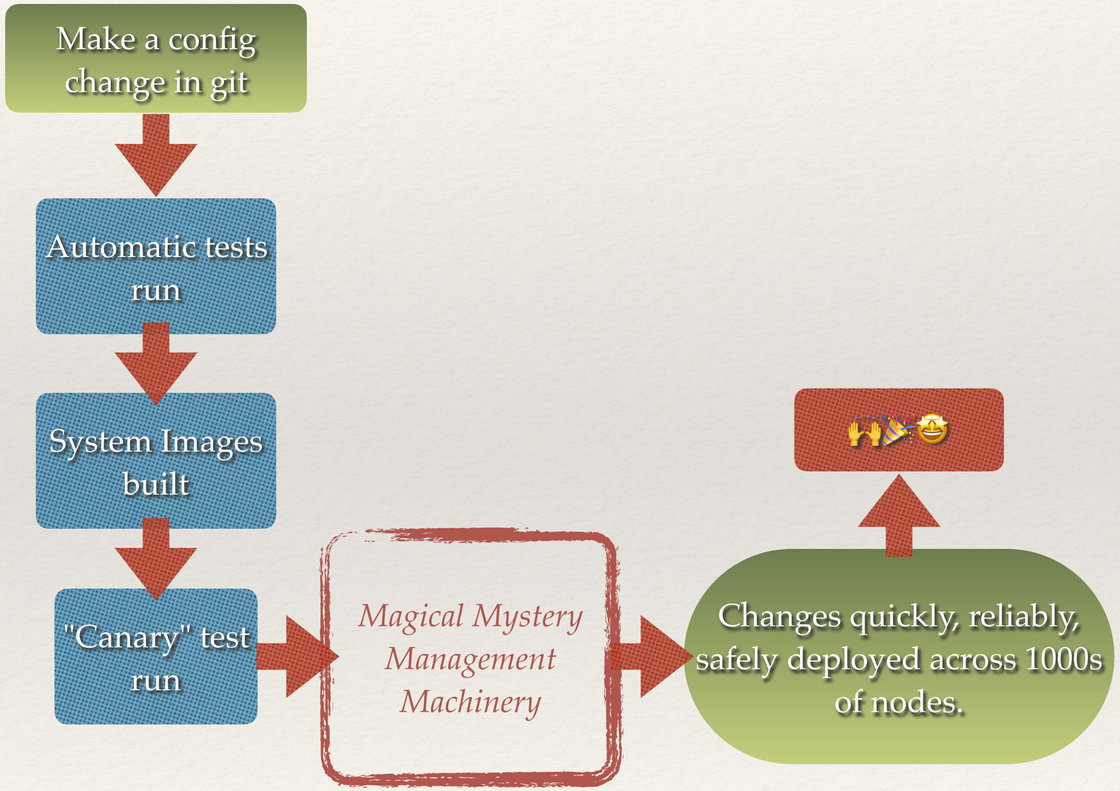
Let's recap.

- We argued we need better tooling for distributed and clustered systems.
- We derived some guiding principles.
- We talked a little about the theory of Kraken/Layercake.
- Then we actually played with it a bit.
- **Is it the "Magical Mystery Management Machinery" in this diagram?**



Let's recap.

- **Is it the "Magical Mystery Management Machinery" in this diagram?**
- We think we're headed there.
  - It meets the requirements in theory.
  - And already powerful in practice.
- It's still got some "rough edges."
- We hope you'll join us in getting it the rest of the way...



Thanks for listening!

References

- **Design Inspiration**
  - Allen, Benjamin S., Ezell, Matthew A., Jacobsen, Douglas, Lueninghoener, Cory, Peltz, Paul, Roman, Eric, & Wofford, J. Lowell. (2020). Modernizing the HPC System Software Stack. Presented at the SC (SC20), Atlanta, GA: Zenodo. http://doi.org/10.5281/zenodo.4324415. arXiv preprint arXiv:2007.10290. Presentation (movie): http://bit.ly/mhsss-syspros20.
  - Peltz Jr., Paul, & Wofford, Lowell. (2018). Next-Generation Cluster Management Architecture and Software. Presented at the SC18 (HPCSYSPROS18), Dallas, TX: Zenodo. http://doi.org/10.5281/zenodo.3552990
- **Kraken HPC Project**
  - https://kraken-hpc.io/ - Kraken & related projects
  - https://github.com/kraken-hpc/kraken - The Kraken framework
  - https://github.com/kraken-hpc/kraken-layercake - The Kraken/Layercake tools
- **Kraken Theory & Architecture**
  - Wofford, J. L. (2021). Designing a scalable framework for declarative automation on distributed systems. arXiv preprint arXiv:2104.13263 (2021).
  - Wofford, J. L. (2020). Kraken: A framework for declarative automation of distributed systems. 2020 NSF-CAC Keynote. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-20-23959