FBInstance to MetaInstance

Our Journey from Long-Running Mutable to Immutable Instances

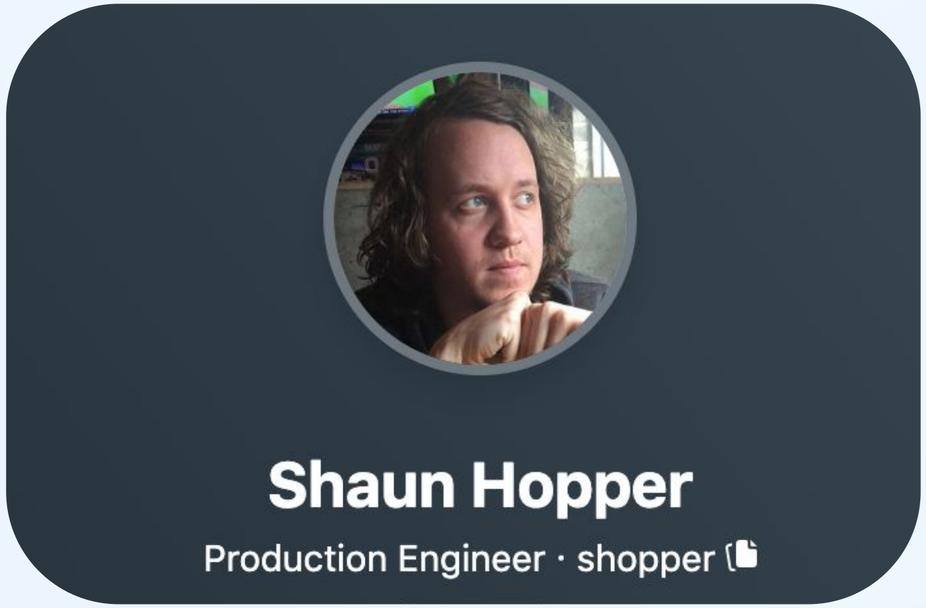
Shaun Hopper
Production Engineer

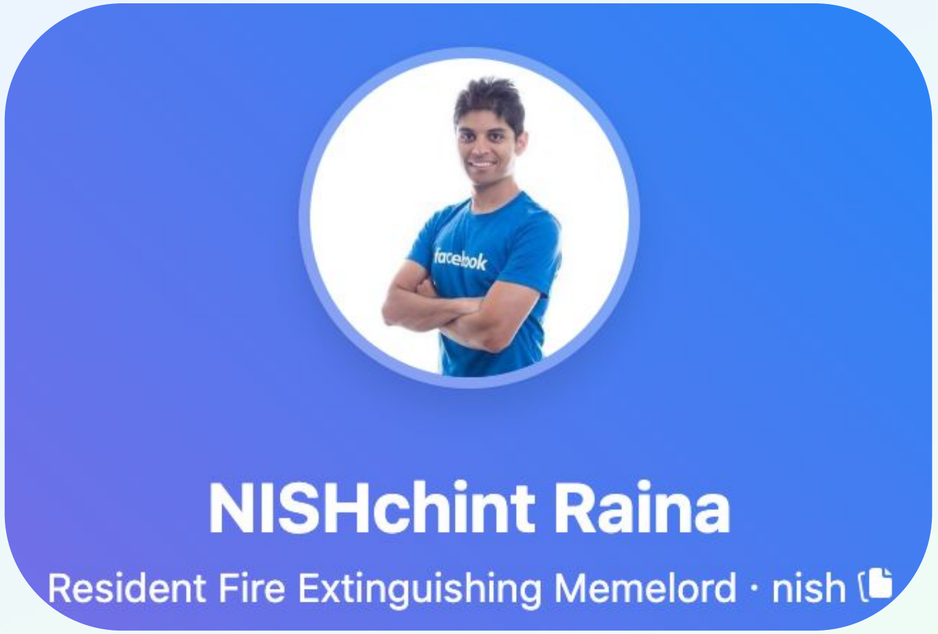
NISHchint Raina
Production Engineer

Meta

Who are we?

- Cloud Foundation
- ~5 Engineers building the compute platform
- The rest of the company uses it





Why are we here?

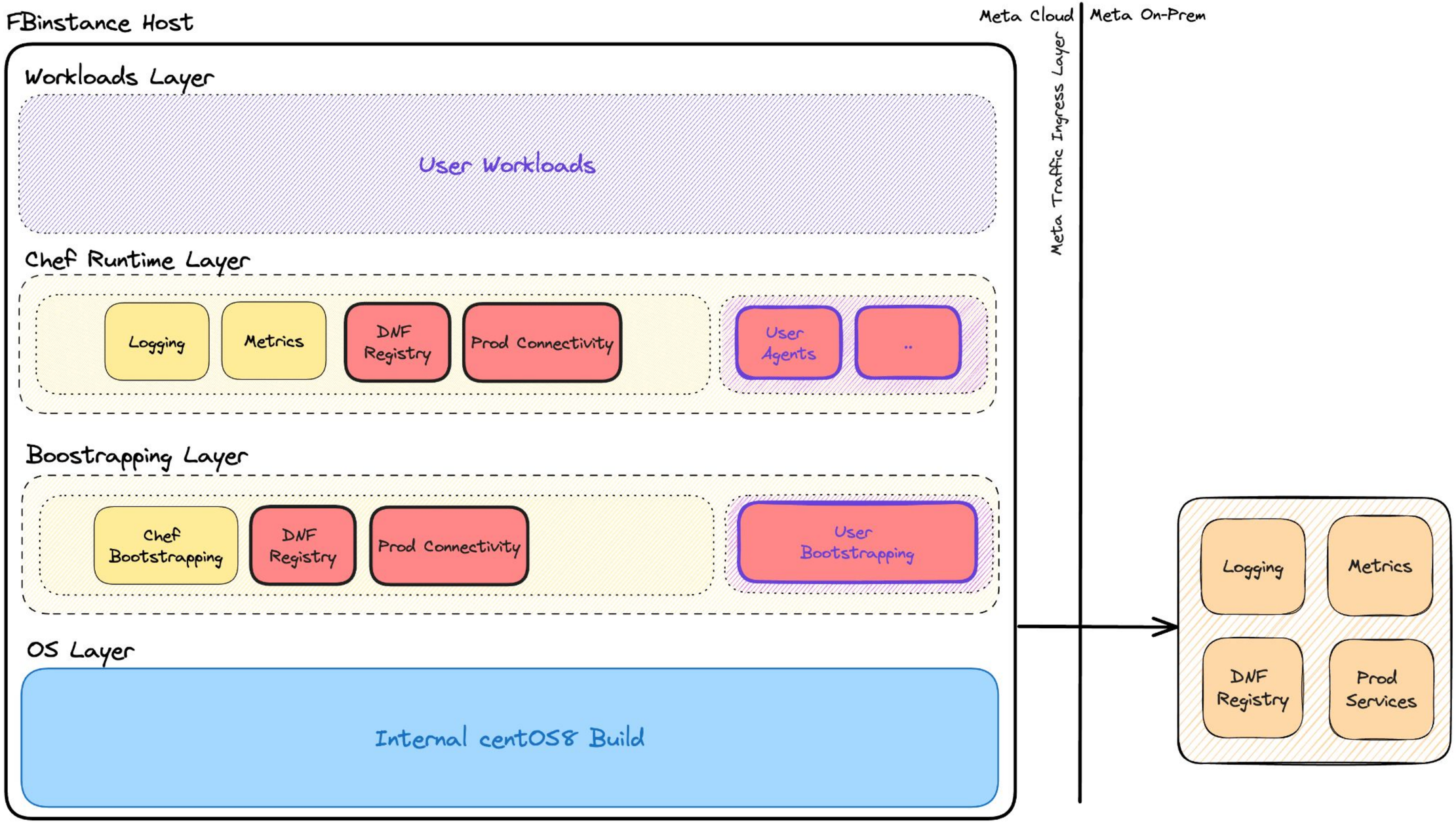
The story begins with fbinstance

fbinstance

- Cloud was new to Facebook.
- Needed a common platform that looked like a Facebook host
  - CentOS-based
  - Chef
  - DNF registries
- Wanted to reuse and extend familiar tooling
- Needed ability to talk to services on-prem

What did an fbinstance host look like?

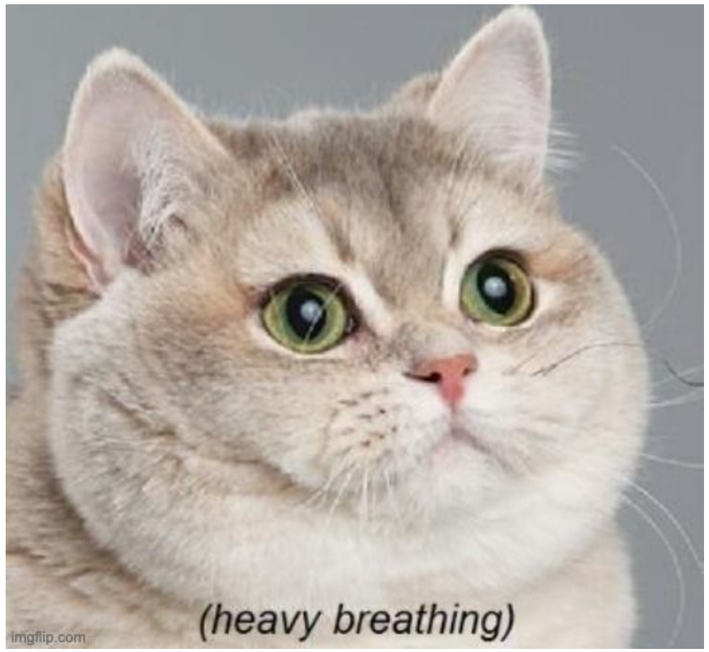
FBINSTANCE


FBinstance Host
Workloads Layer
User Workloads
Chef Runtime Layer
Logging
Metrics
DNF Registry
Prod Connectivity
User Agents
..
Boostrapping Layer
Chef Bootstrapping
DNF Registry
Prod Connectivity
User Bootstrapping
OS Layer
Internal centOS8 Build
Meta Cloud
Meta On-Prem
Meta Traffic Ingress Layer
Logging
Metrics
DNF Registry
Prod Services

fbinstance had problems

- Operational Burdens with Chef
- Dependency Management
- Tight Infrastructure-as-Code (IaC) Coupling
- Reliance on On-Prem Network Connectivity

fbinstance problems

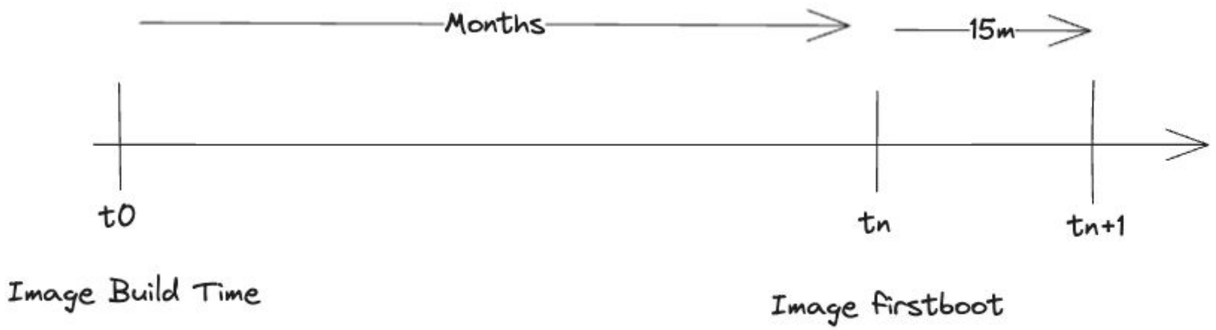


Operational Burdens with Chef

- Had no support for staged rollouts of changes, and couldn’t roll back
- One recipe breaking could hold up rest of the deployment
- Existing testing functionality was partially supported and inadequate for non-homogenous hosts
- Always playing catch-up with internal chef cookbooks
  - Many cookbooks written came with on-prem assumptions
  - Cookbooks in state of frequent changes
  - Only had support for centos, we needed to support for other distros

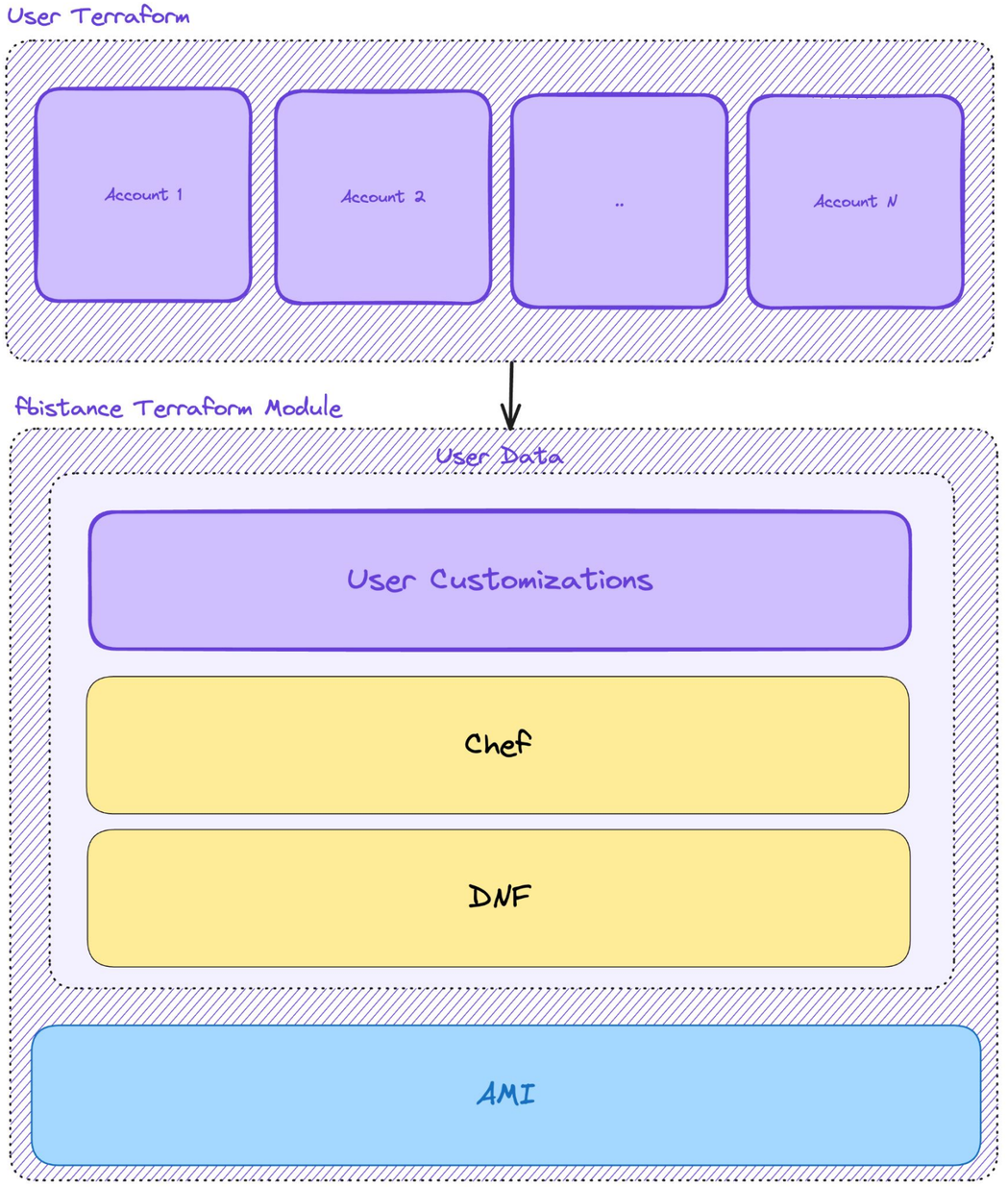
Dependency Management

- RPM dependencies were installed at multiple stages of the hosts lifecycle, increasing conflicts.
  - Internal CentOS build time
  - AMI build time
  - Bootstrap time
  - Chef runtime
- Testing AMIs in CI/CD had limited reproducibility due to chef’s constantly moving nature.
- The "15 minute upgrade problem” vs "6 month upgrade problem"



Infra-as-Code (Terraform)

- The ability to bootstrap fbinstance depended on using our custom terraform modules.
- Users did their customizations to fbinstance through terraform.
- We couldn’t control when terraform was actually run in every account.



On-prem Connectivity Reliance

- In order to talk to on-prem services, everything previously mentioned had to work flawlessly.
- Compliance and security requirements to talk to on-prem are strict.
- Continuously updating hosts required talking to on-prem.


IMAGINED WITH AI
IMAGINED WITH AI

So where do we go from here?

We knew where we wanted to be

We knew we wanted immutability

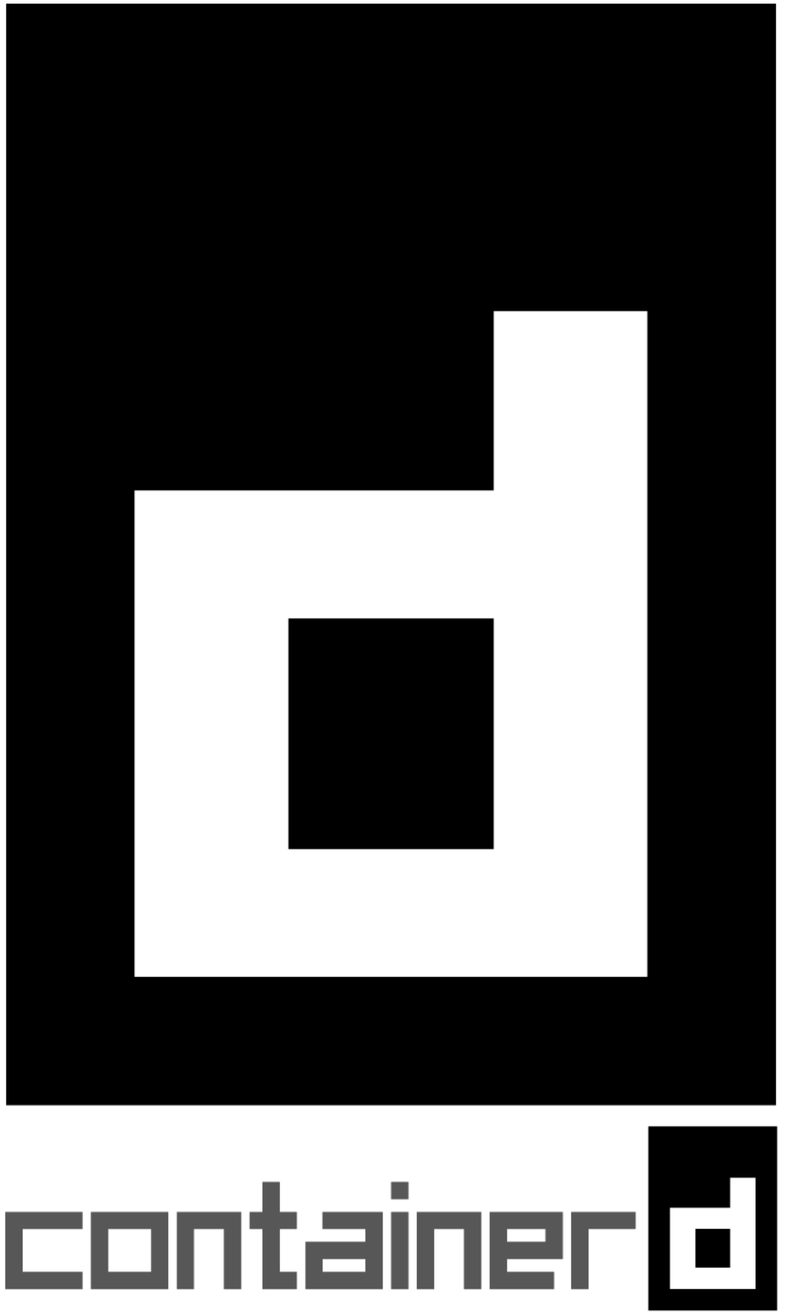
- All bootstrapping dependencies are baked into the image.
- Once bootstrapped, it stays unchanged during the course of its lifetime.
- Updating the instance requires deploying a new build.
- Instead of chef, we rely on CI/CD and shorter running instances.

We knew we wanted to support multiple distros

- Beyond generic compute, support a broader set of use cases (HPC, Kubernetes).
- A consistent way to:
  - Bootstrap a host.
  - Package, fetch, install and run our platform daemons.

So how can we get there?


systemd


containerd

Why systemd + containerd?

systemd + containerd

- Systemd provide a consistent bootstrap story & requires minimal changes across distros.
- OCI containers + containerd keeps our platform binaries portable.
- Instead of maintaining binaries per distro, we maintained a single container per binary.

Could we pull it off?

- OCI Container tooling was maturing due to ongoing investments in supporting kubernetes.
- It had put us into an immutable first mindset.
- We could consolidate the teams efforts around containerizing our platform.

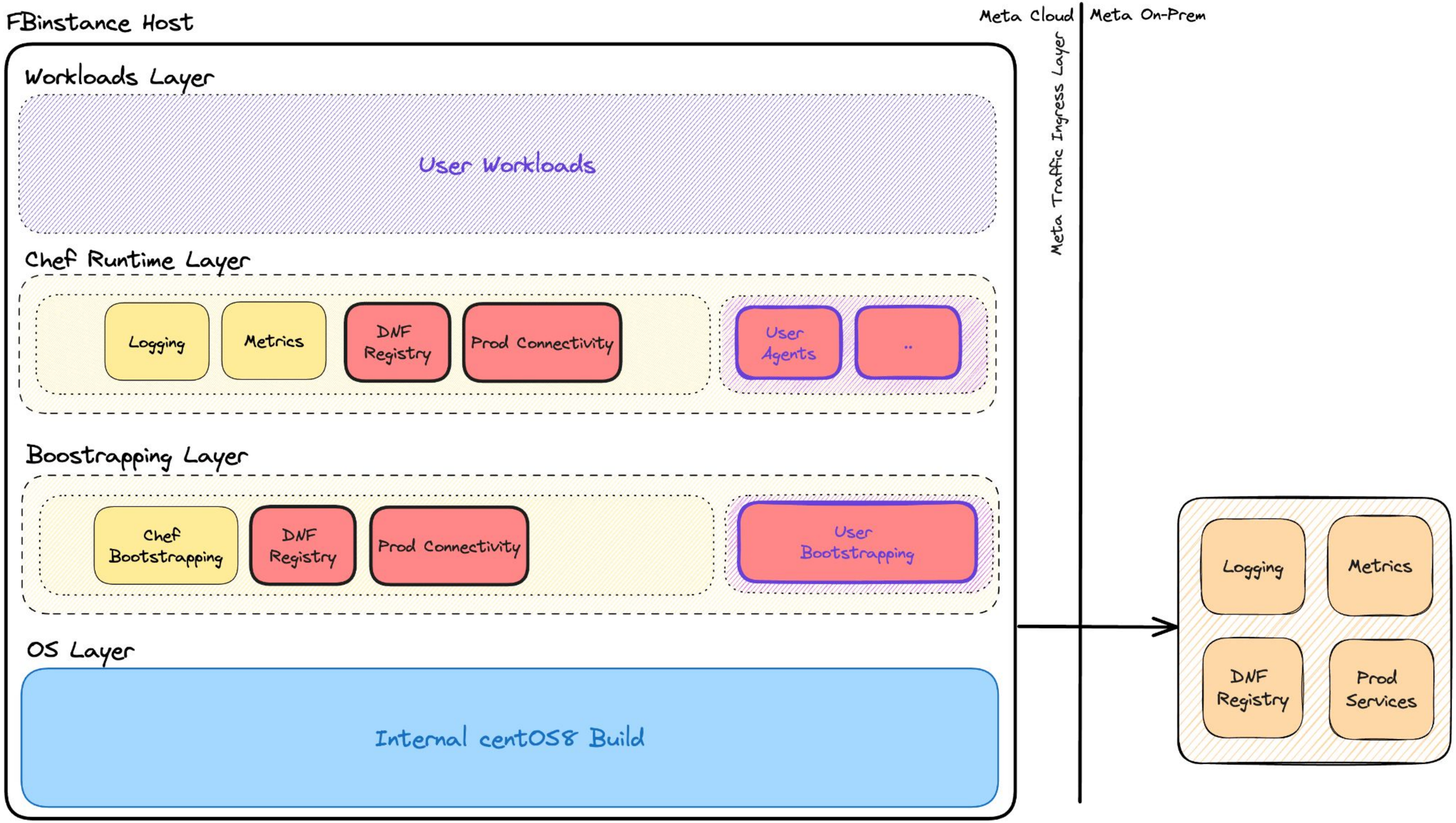
We could

We called it… Metainstance

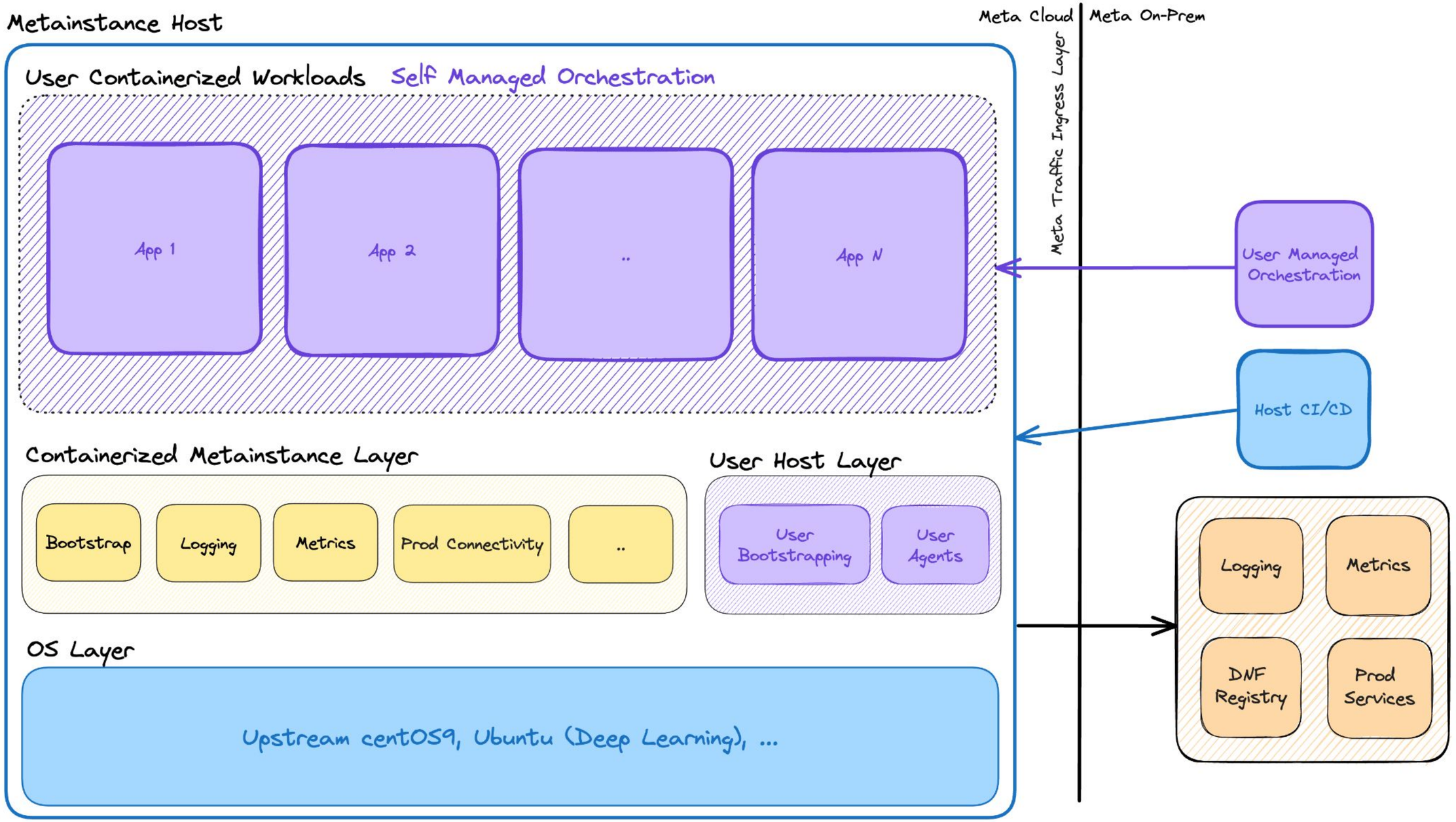
Building Metainstance

- We stripped down the functionality of what chef does on the host to near 0.
- We put systemd in charge of bootstrapping the host.
- We put all platform daemons into their own containers.

METAINSTANCE



METAINSTANCE

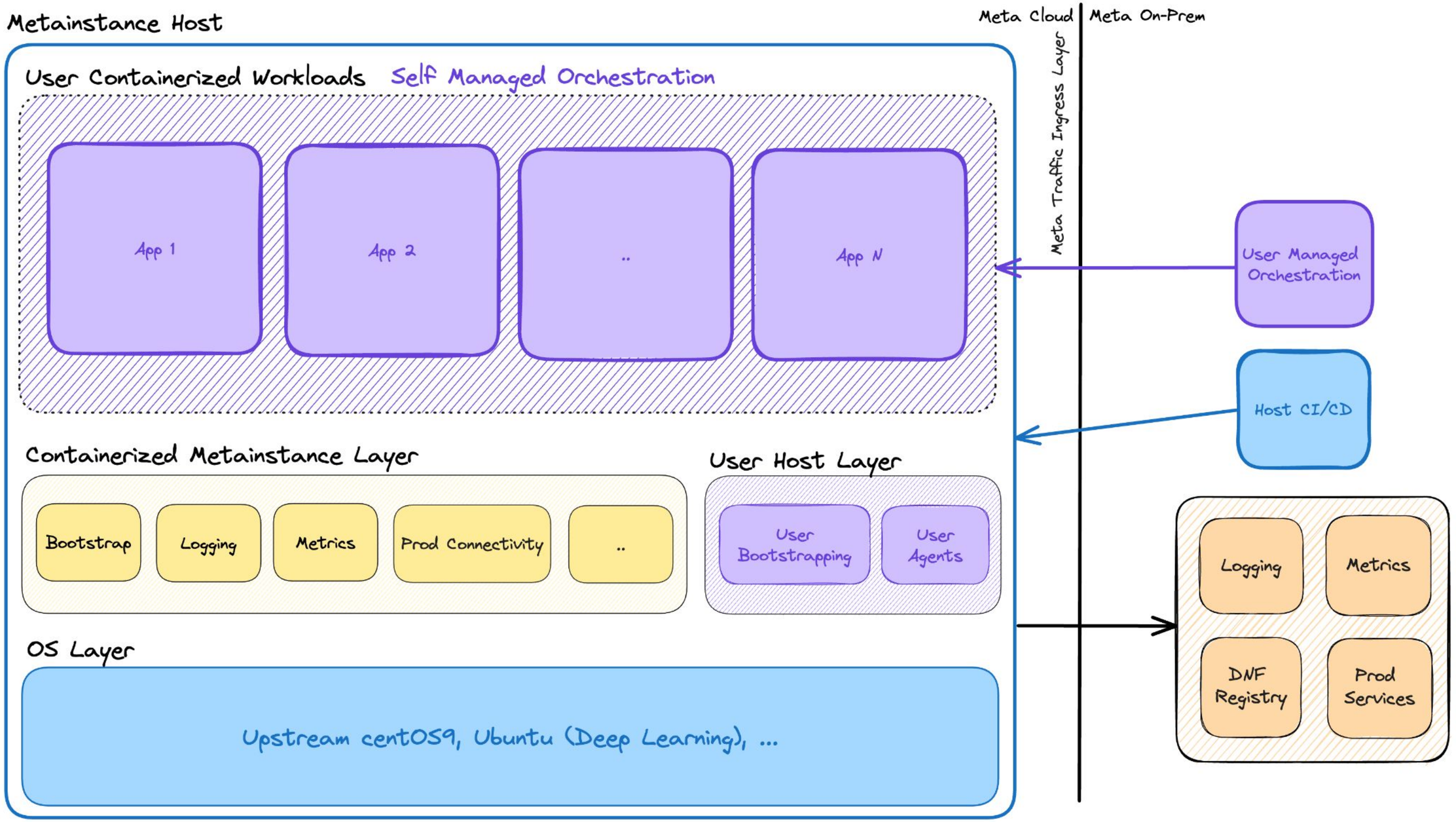


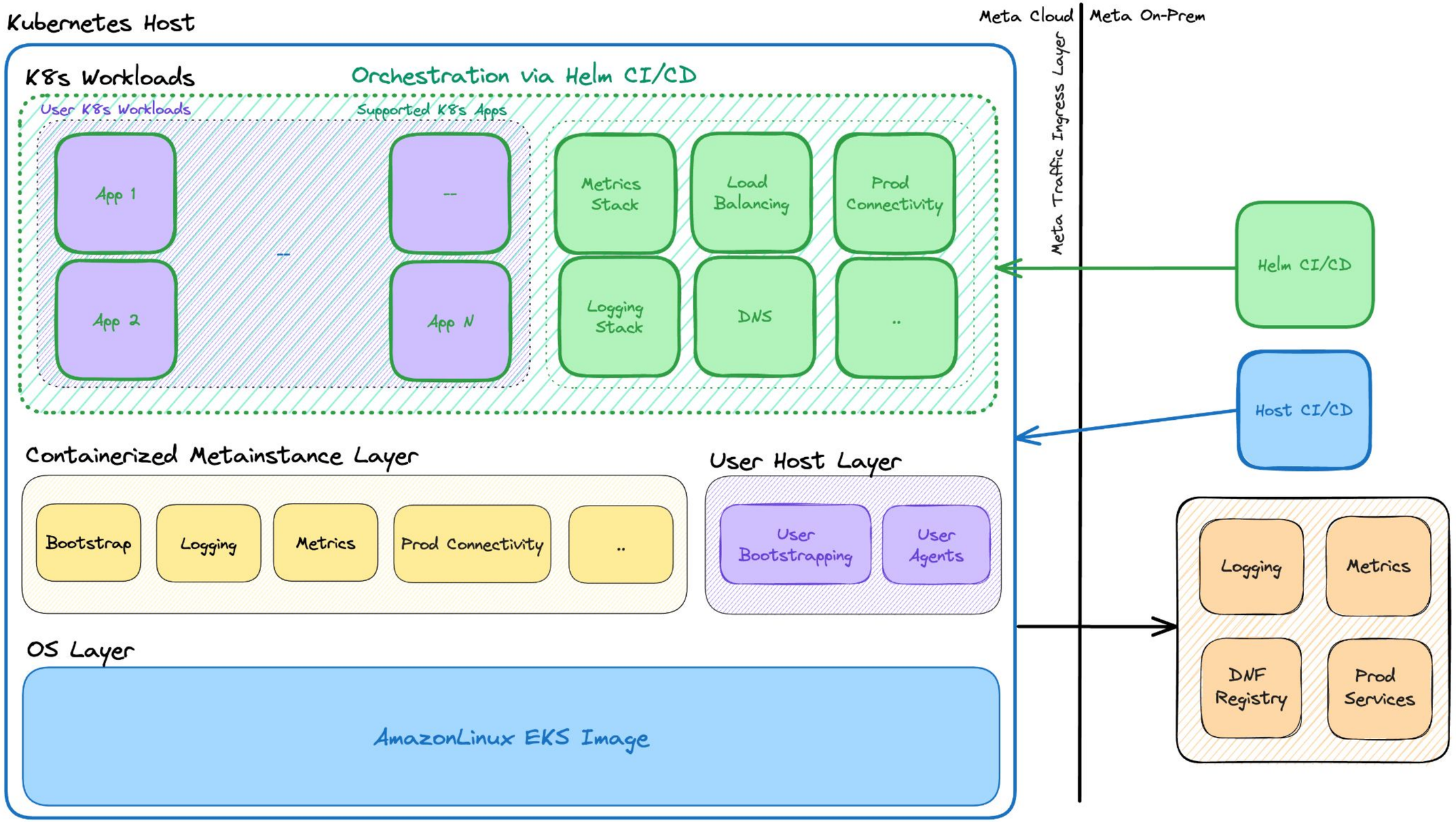
What did it give us?

Portability

- It scaled our ability to support multiple host distros for specific use cases.
  - We could now support Ubuntu (HPC) and Amazon Linux (EKS) in addition to CentOS.
- Our platform daemons could run in kubernetes too.

METAINSTANCE




Kubernetes Host
K8s Workloads
Orchestration via Helm CI/CD
User K8s Workloads
Supported K8s Apps
App 1
App 2
App N
Metrics Stack
Load Balancing
Prod Connectivity
Logging Stack
DNS
Containerized Metainstance Layer
Bootstrap
Logging
Metrics
Prod Connectivity
User Host Layer
User Bootstrapping
User Agents
OS Layer
AmazonLinux EKS Image
Meta Cloud
Meta On-Prem
Meta Traffic Ingress Layer
Helm CI/CD
Host CI/CD
Logging
Metrics
DNF Registry
Prod Services

Predictability

- Instance runtime became predictable.
- We could deterministically catch failures at CI/CD test time.
- Rolling back became possible, just redeploy an older version.

Simplicity

- We could lean more on industry standards.
- We didn’t have to support multiple stacks of bespoke tooling.
  - We leveraged existing AMI/Container tooling for building & deployment.
  - We invested less into the ecosystem of chef tooling.
- Leaner Terraform Module.

But it wasn’t all great

- Supporting separate distros meant wrangling differences in systemd/cloud-init versions.
- New learning curves:
  - Docker became a barrier to entry.
  - Bash isn’t as expressive as ruby (chef).
  - Frequent CI/CD for hosts was a new concept.

Takeaways

- Moving host configuration to build time results in fewer alerts and incidents, more sleep!
- Most failures become CI pipeline issues, less urgency to respond.
- If you have to support functionality in kubernetes anyway, you may as well use the same containers on your instances.
- By containerizing the entire platform, integrating new functionality (security agents, telemetry agents, etc) becomes easy. Just add it to the MetaInstance layer.

Questions?

THANK YOU FOR YOUR TIME


Meta