Immutable Infrastructure

<<< Shift Left <<<

Rethinking Configuration in the Age of Easy Redeployment

RackN, Inc

February, 2018

Note: Graphics mainly from http://pexels.com

Your Humble Presenter

I'm all about automating infrastructure.

Involved in Open Ops Software:

Digital Rebar Project
Kubernetes ClusterOps SIG
OpenStack Board

Rob Hirschfeld (aka @zehicle)
Co-Founder of RackN
rob@rackn.com

RackN

Storytime! “Self-Bootstrapping Kubernetes”

Kubecon in Nov 2017 we created this demo

Simple “immutable” Idea:

1) **In Memory Boot Machines**
2) **Install Docker**
3) **Elect Leader**
4) **Run Kubeadm on Leader**
5) **Run Kubeadm on Remainder**

But….it’s shockingly hard to maintain.

Dependencies breaks the installation

And they are constantly changing.

Storytime! “Self-Bootstrapping Kubernetes”

So, while it’s pretty cool,
it’s not “real” immutability

Presentation & Demo
https://youtu.be/OowxF6GqK4I

Why is configuration fragile?

Why is ~~configuration~~ mutation fragile?

RackN

But... I ❤ Infrastructure as Code?! Sorry. Mutability adds complexity

Traditional "build-in place" approaches

- **Have hidden dependency graphs**
- **Create variation between environments**
- **Are harder to "lock down" due to config**

AND OMG... updates and patches are even harder

- **Idempotent operations are difficult**
- **Roll backward is next to impossible!**
- **Creating indeterminate state**

But... I ❤ Infrast as code?!
Sorry. Mutab adds complexity

Traditional "build-in place" approaches

- Have hidden dependency graphs
- Create variation between environments
- Are harder to "lock down" due to config

Let's lock it down!

AND OMG... updates and patches are even harder

- **Idempotent operations are difficult**
- **Roll backward is impossible**
- **Creating indeterminate state**

What is Immutable Infrastructure?

RackN

Pre-deploy configured

What is ~~Immutable~~ Infrastructure?

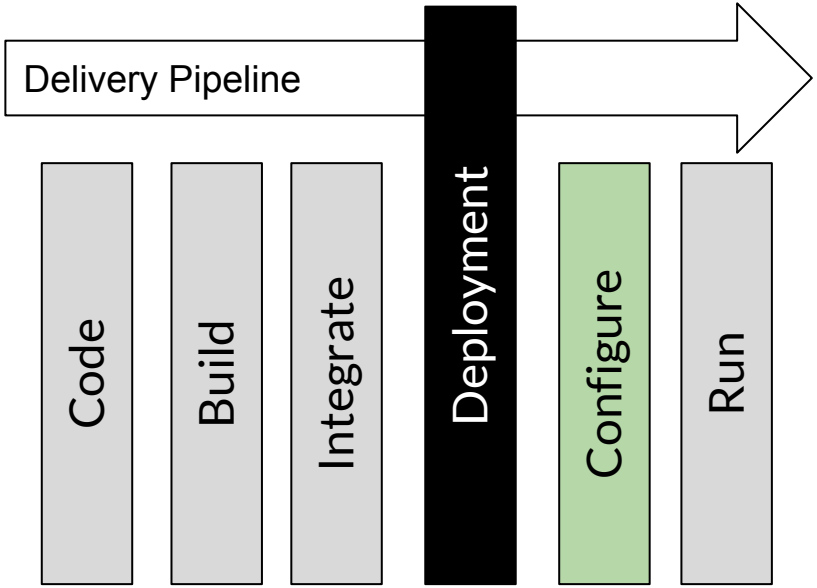
RackN

Traditional Deploy and Configure

System is configured *in situ* from a least common denominator baseline.

This can be "immutable-like" under the right conditions.

We'll come back to that...

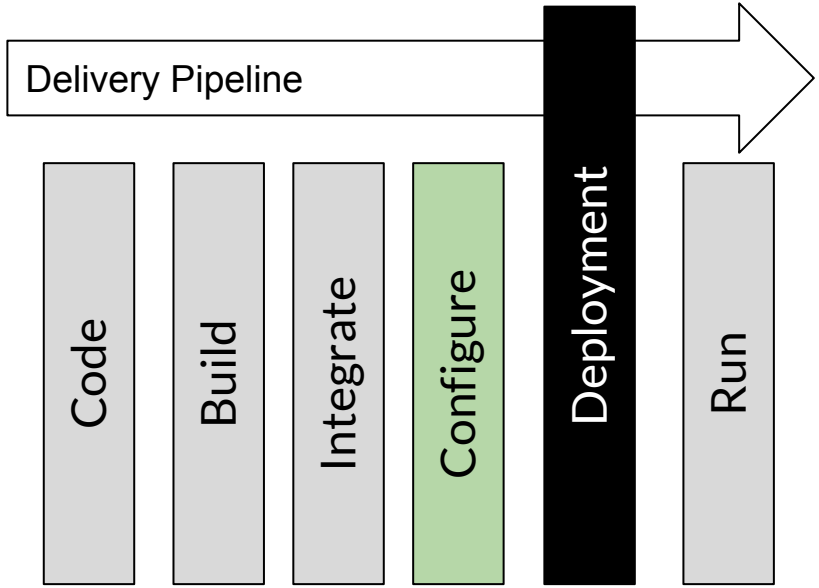


RackN

Shifting Configuration BEFORE Deployment

In our ideal delivery pipeline, configuration is *before* deployment.

Running systems are delivered as a complete runnable unit for deployment.

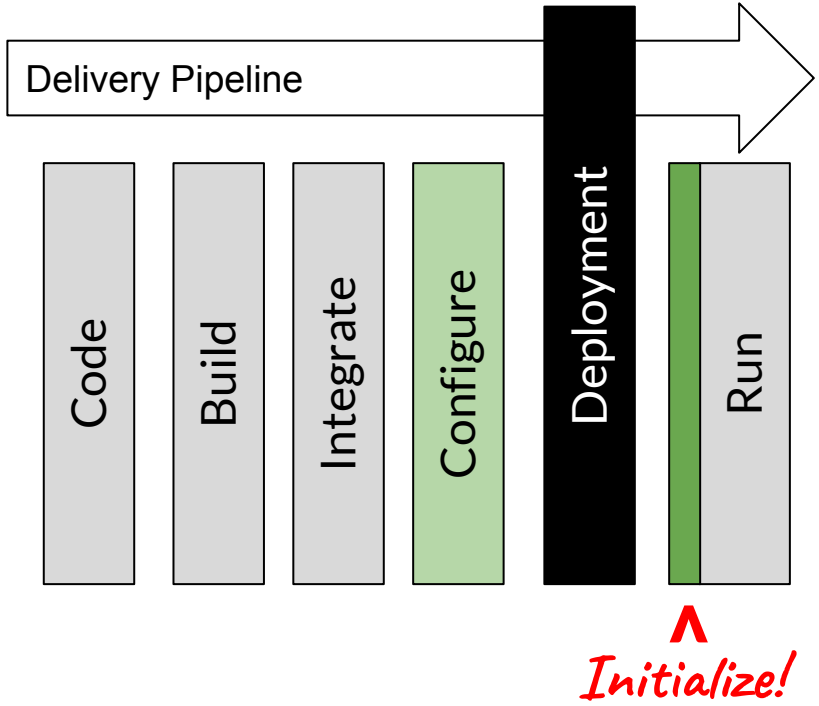


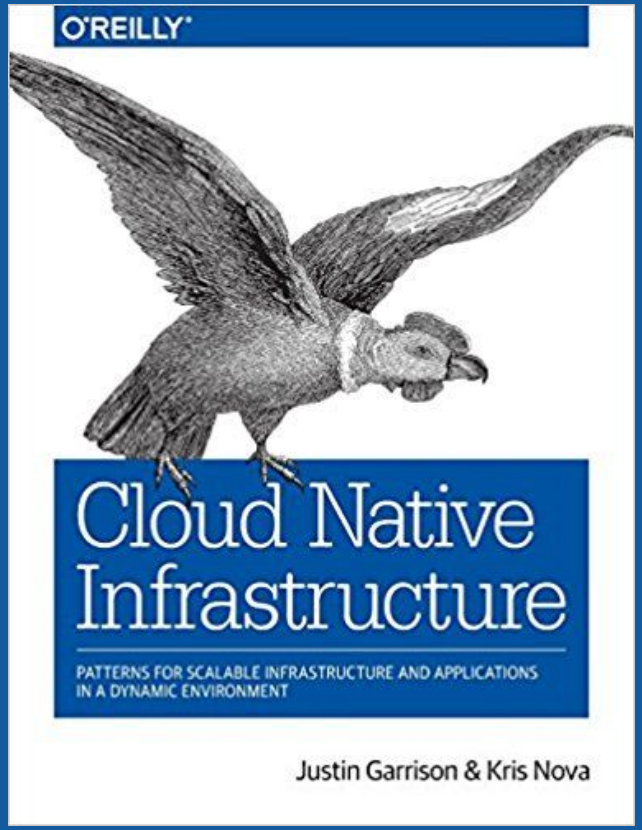
Shifting Configuration BEFORE Deployment

In reality, it's very hard to create a distinct artifact for every running instance; instead, we create incremental versions.

So we do *some* initialization of the reusable versioned instance.

Cloud init is the most commonly known pattern for this.





RackN

Cloud Native Infrastructure

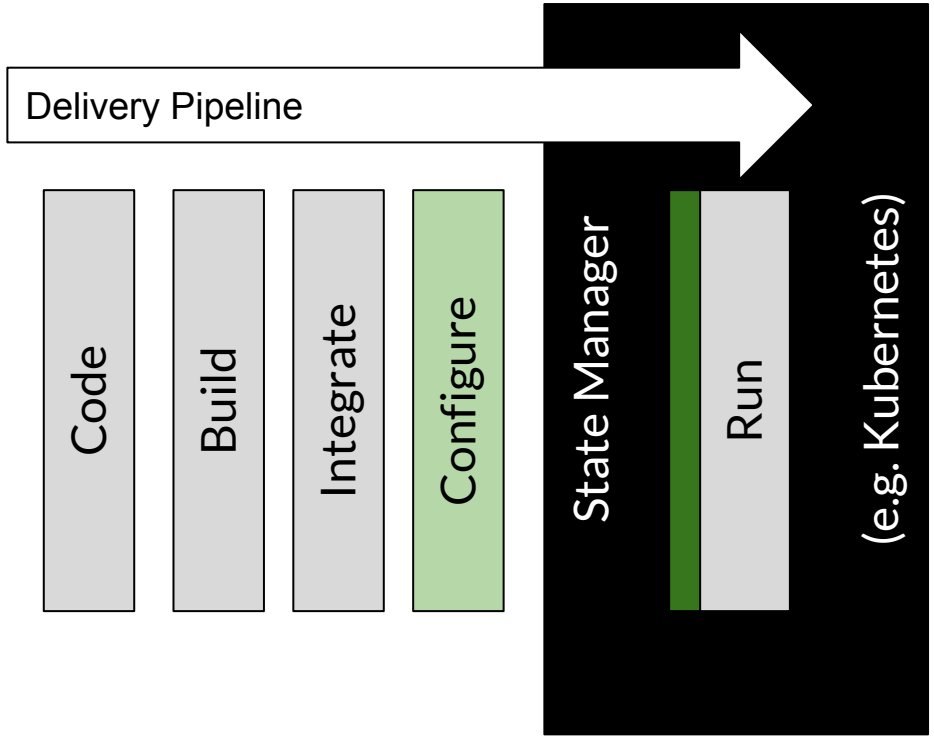
CNIbook.info
Justin Garrison & Kris Nova

“Infrastructure as software”

Which Enables... Delegating Operations

If you can make your artifacts immutable then you can delegate management of them to a platform like Kubernetes.

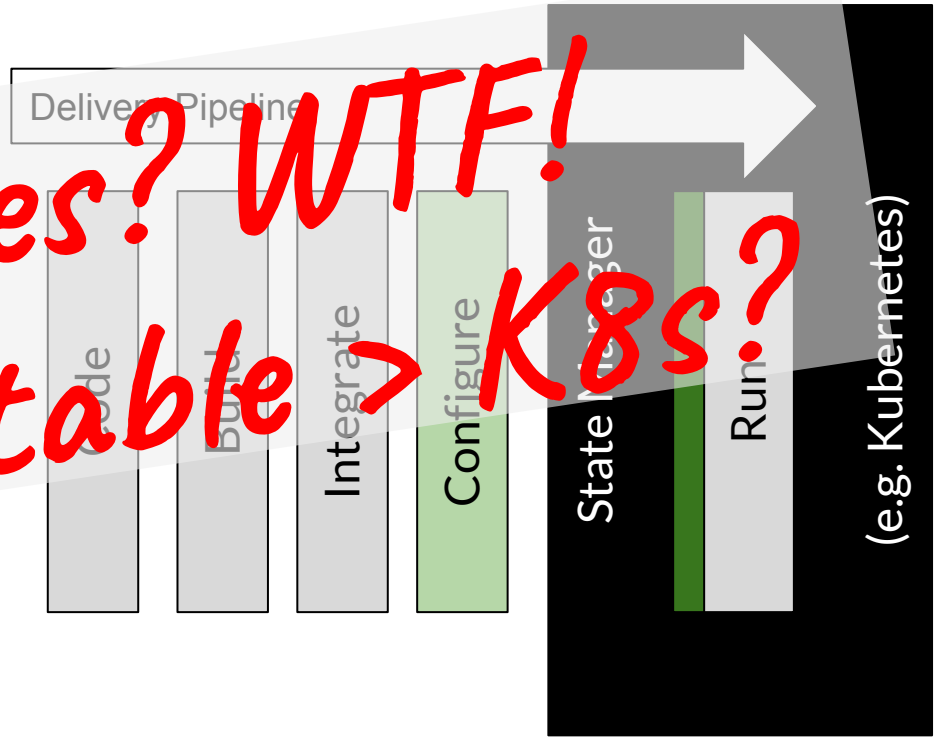
Kubernetes does not configure infrastructure. It maintains state based on a manifest.



Which Enables... Delegating Operations

If you can make your artifacts immutable then you can delegate management of them to a platform like Kubernetes.

Kubernetes does not configure infrastructure. It maintains state based on a manifest.



Kubernetes? WTF!
Is Immutable > K8s?

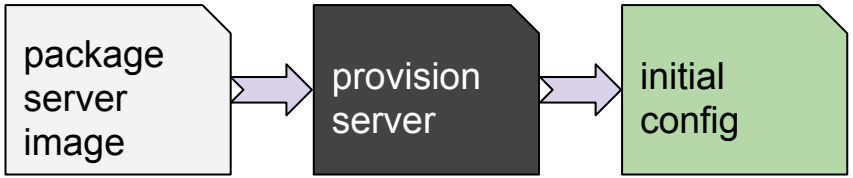
Immutable is a DevOps Pattern

<<< Shift Left & Create/Delete

Immutability <<< Shifting Left

RackN

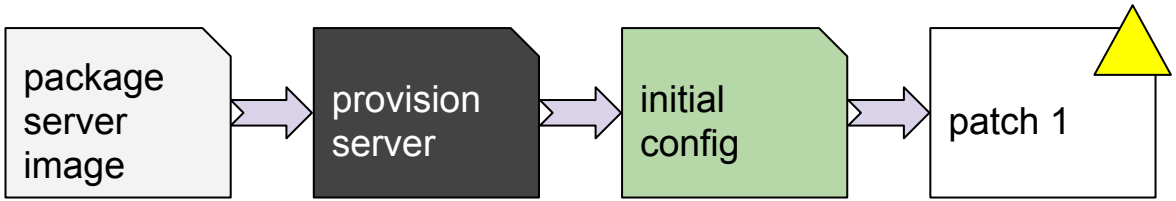
The Problem



Immutability <<< Shifting Left

RackN

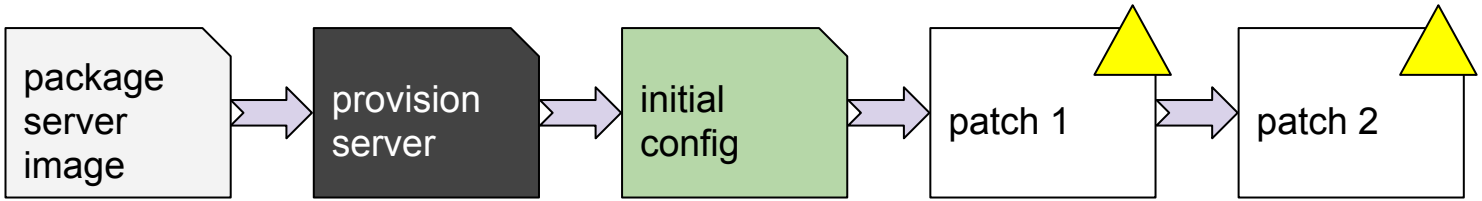
The Problem



Immutability <<< Shifting Left

RackN

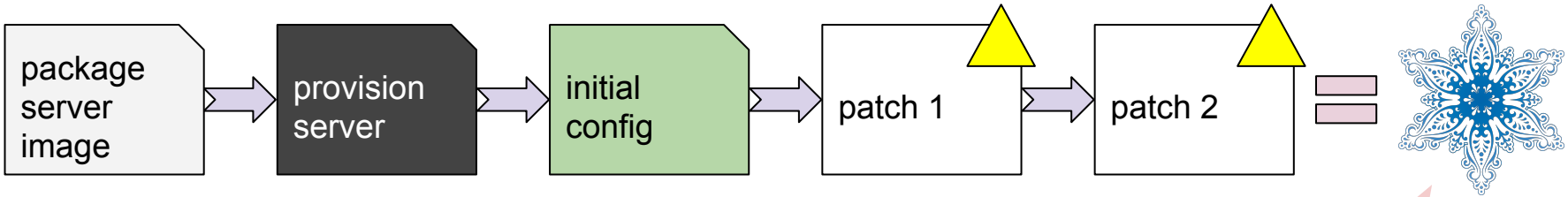
The Problem



Immutability <<< Shifting Left

RackN

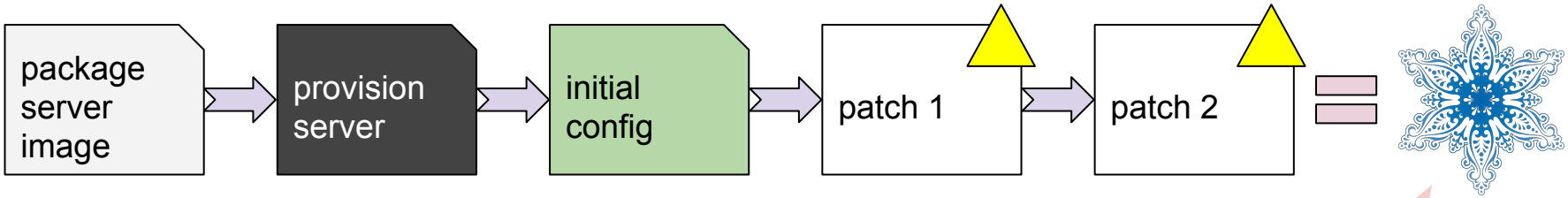
The Problem



Immutability <<< Shifting Left

RackN

The Problem



What Madness?

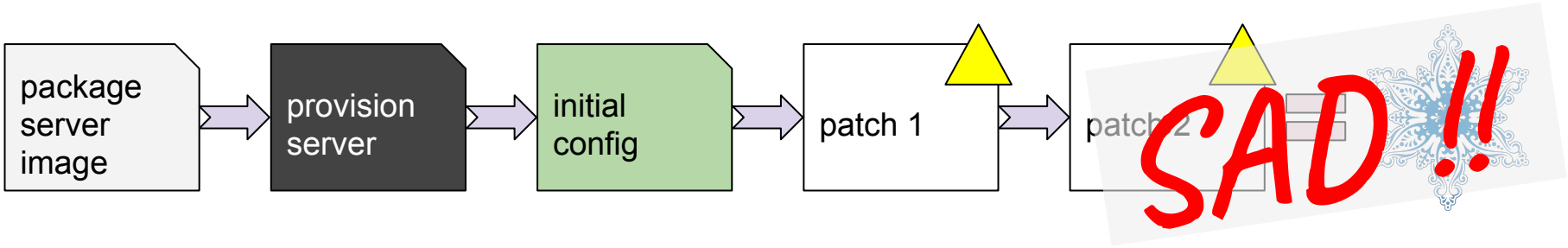
- We have to maintain root access
- Patches assume system state
- Patches create dependency graphs
- Coordination? Should we halt work?
- Drift is inevitable!

the madness doesn't stop at patch 2!

Immutability <<< Shifting Left

RackN

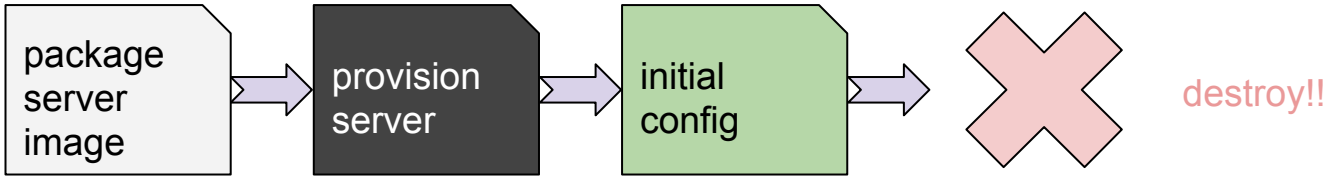
The Problem



Immutability <<< Shifting Left

RackN

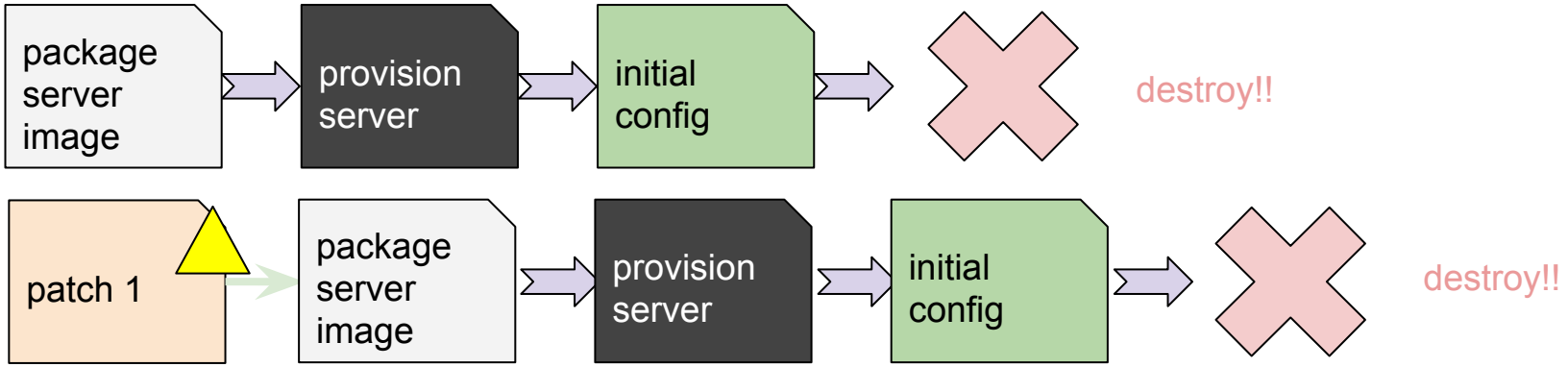
Apply cloud and container lessons to our Bare Metal …



Immutability <<< Shifting Left

RackN

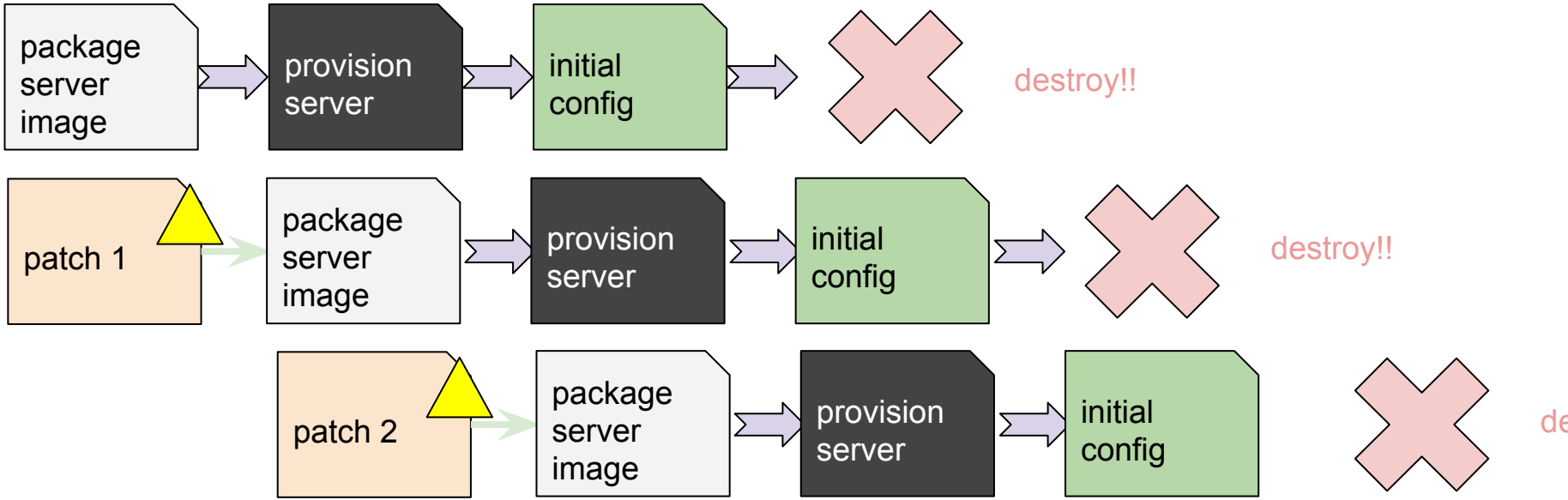
Apply cloud and container lessons to our Bare Metal ...



Immutability <<< Shifting Left

RackN

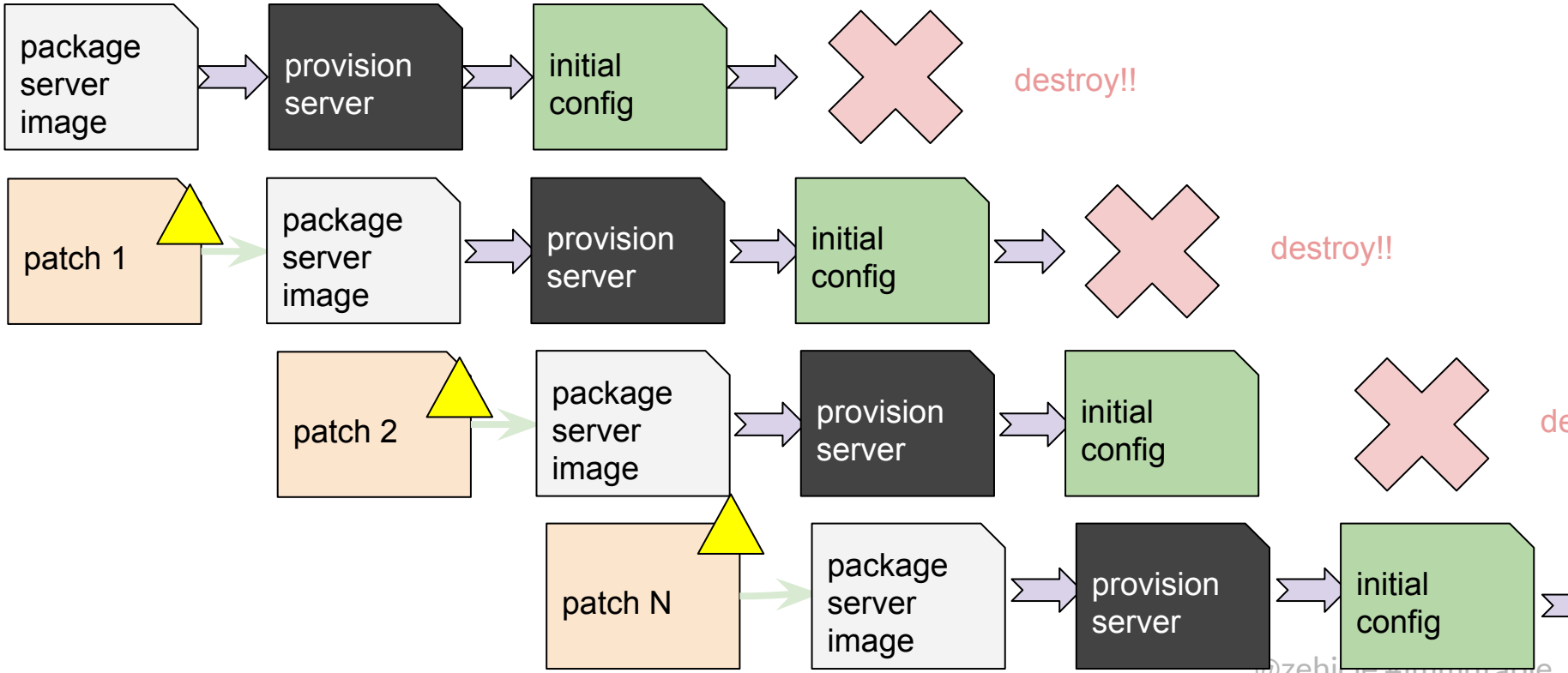
Apply cloud and container lessons to our Bare Metal …



Immutability <<< Shifting Left

RackN

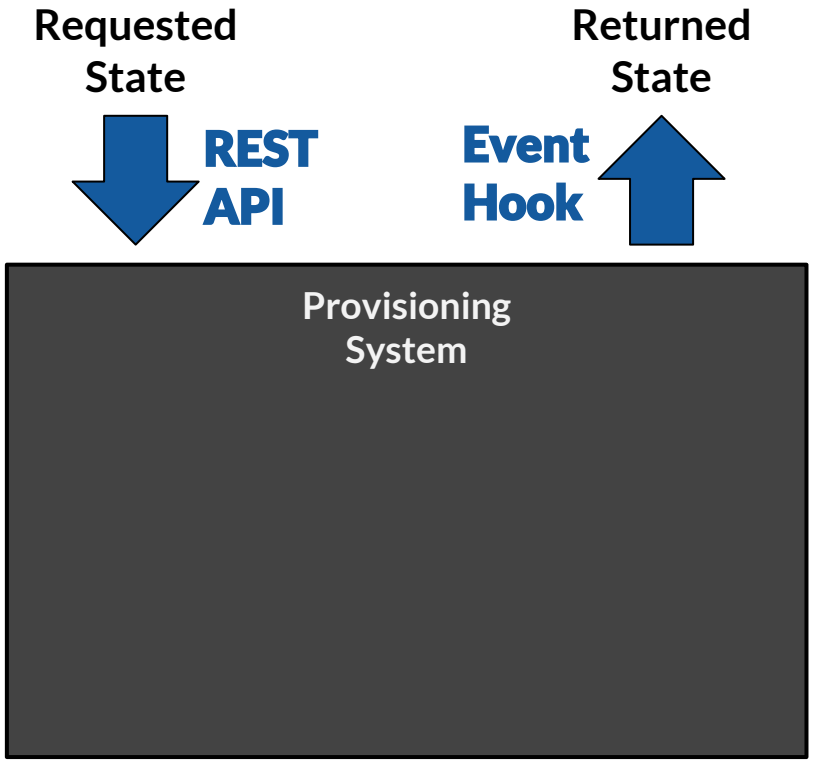
Apply cloud and container lessons to our Bare Metal …



Cloud like behavior …

Cloud-like Integration and Staged Workflow

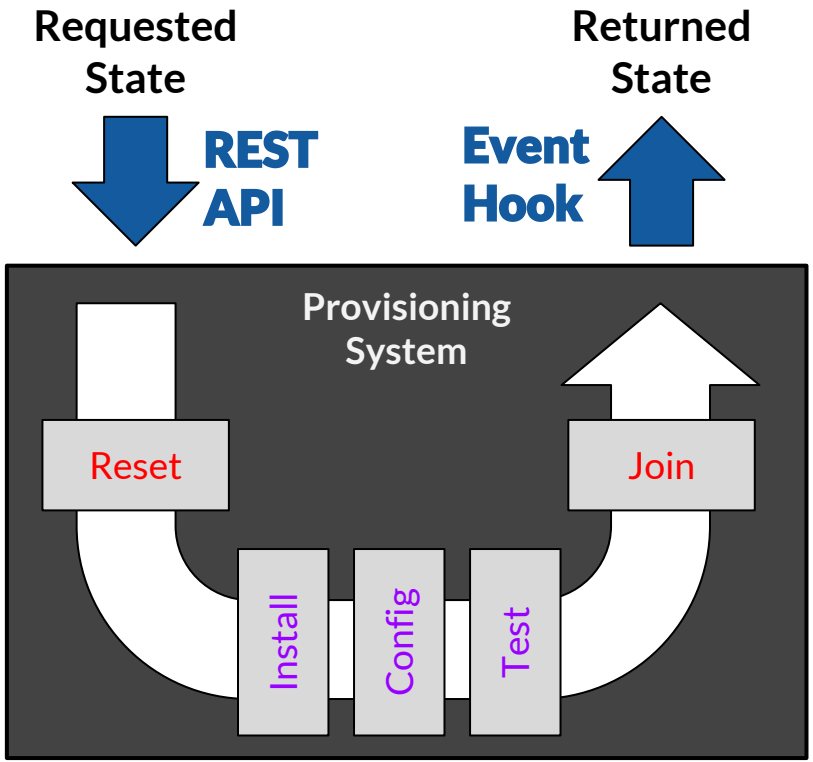
RackN



Immutable Provisioning systems treat infrastructure as a black box

Cloud-like Integration and Staged Workflow

RackN



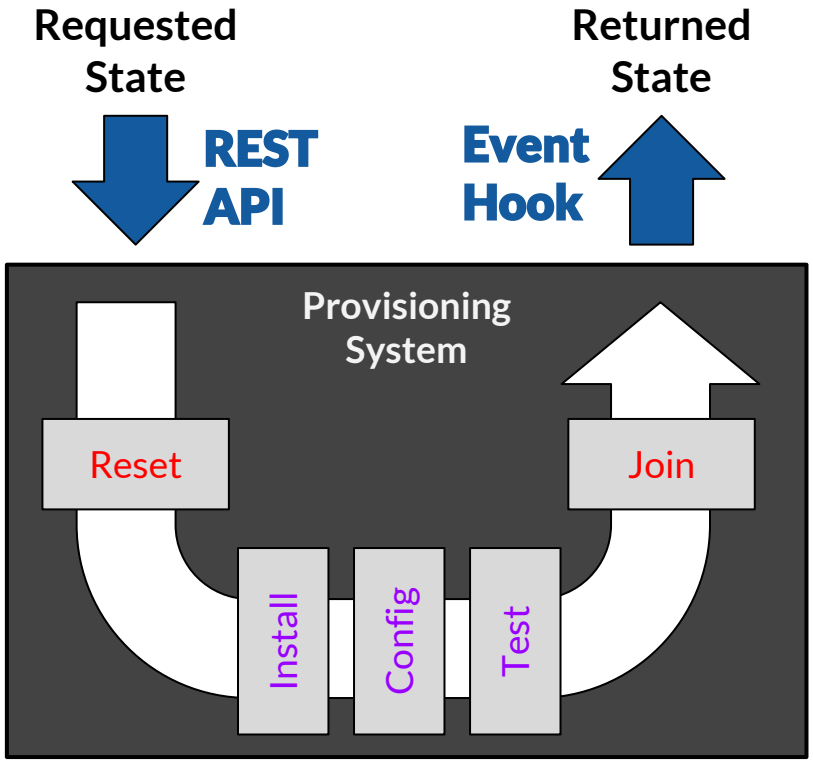
Immutable Provisioning systems treat infrastructure as a black box

Provision requests are for a system state with optional parameters.

The intermediate changes to achieve the state are *not exposed* to the requester.

Cloud-like Integration and Staged Workflow

RackN



Immutable Provisioning systems treat infrastructure as a black box

Provision requests are for a system state with optional parameters.

The intermediate changes to achieve the state are *not exposed* to the requester.

No hidden operations!

REMEMBER: Operators of the provisioning system require high transparency, stages and control.

Immutable Patterns

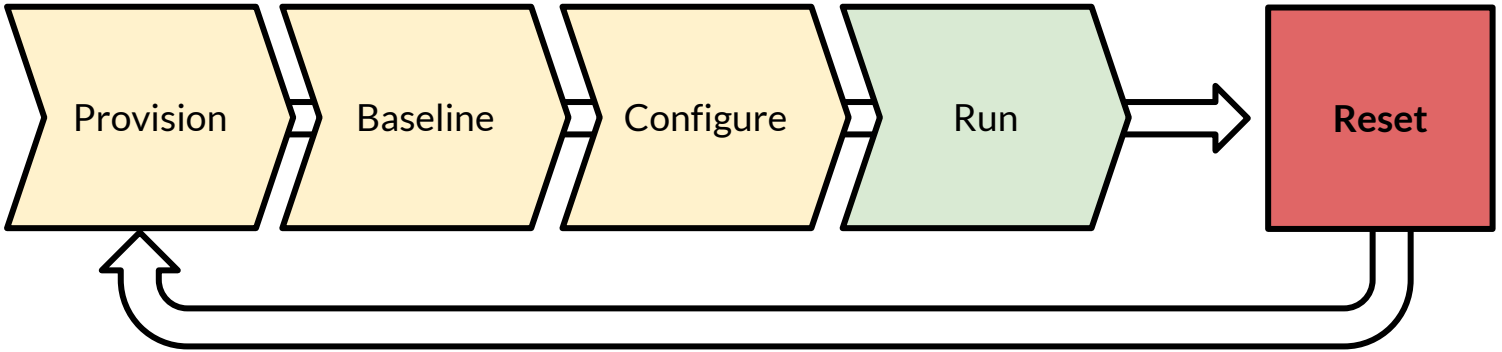
1) Baseline + Configuration
2) Live Boot + Configuration
3) Image Deploy

1: Baseline + Configuration

RackN

Benefit: Easiest to achieve with current tools, Safer than Patching
Challenge: Lots of Post-Configuration, Not Really "Immutable", Slow

Instead of relying on patches, rely on starting from a pristine image



Additional Reference https://thenewstack.io/immutable-hardware-ops-hygiene-security-efficiency/

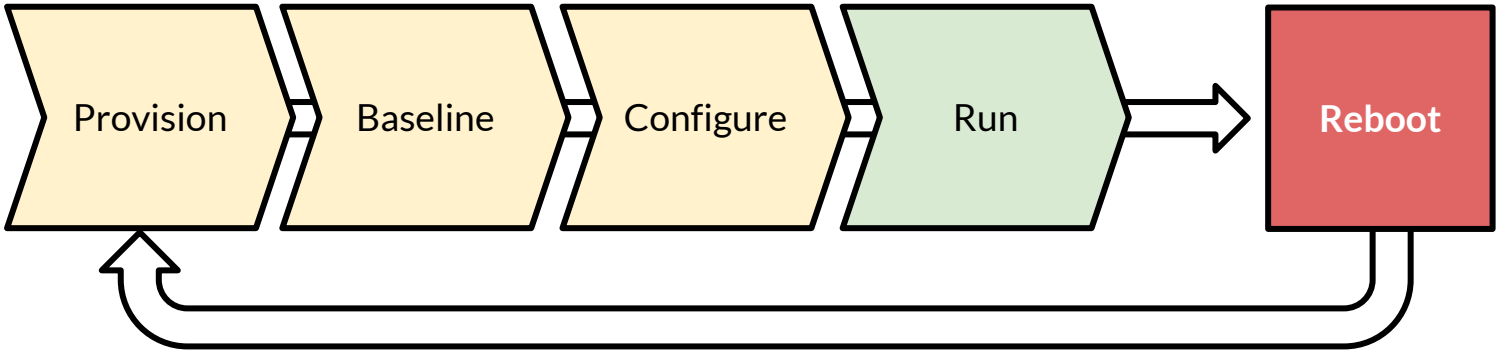
2: Live Boot + Configuration

RackN

Benefit: Fast reset times, forces good behavior

Challenge: Provisioning becomes critical path, still have dependency graph

Like #1 but clean-up is simply a reboot. Favors smaller footprint O/S.

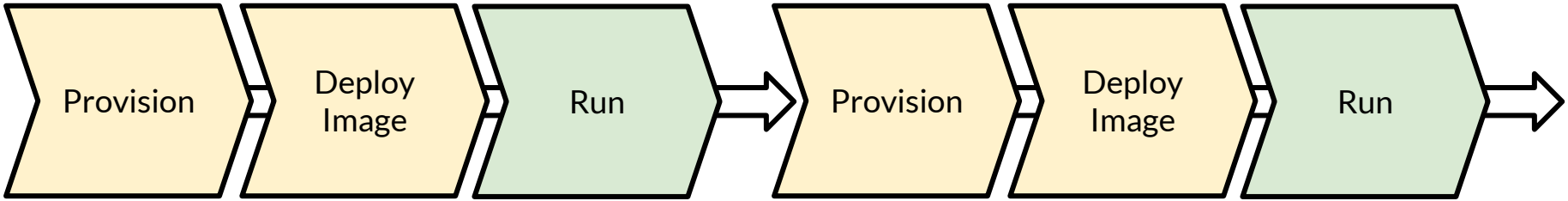


3: Image Deploy

RackN

Benefit: Shorter time to ready, highly controlled (“shift left”), rollback
Challenge: Harder to create and deploy images

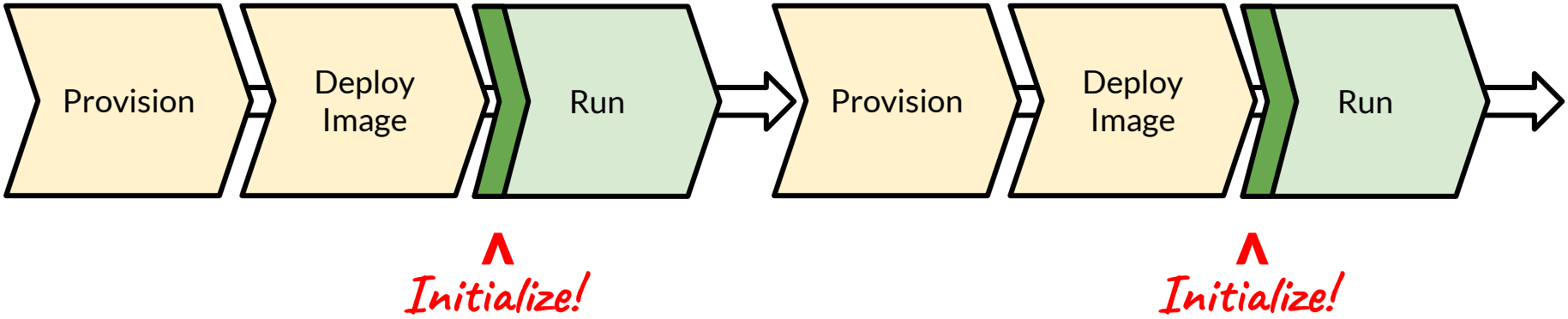
Image is deployed from source instead of Baseline + Configure



3: Image Deploy

Benefit: Shorter time to ready, highly controlled (“shift left”), rollback
Challenge: Harder to create and deploy images

Image is deployed from source instead of Baseline + Configure



So… Let's talk Image Creation

RackN

Ideally in an automation build process.

You DO THE CONFIGURATION on a live system (so you still need configuration tools) and then capture the image into a portable format.

Tools like Hashicorp Packer, Image Builder, WBIC or raw images are used to create source files (e.g. AMI, OVS).

So… Let's talk Image Creation

RackN

Ideally in an automation build process.

You DO THE CONFIGURATION on a live system (so you still need configuration tools) and then capture the image into a portable format.

Tools like Hashicorp Packer, Image Builder, VMDK or raw images are used to create source files (e.g. AMI, OVS).

That sounds like a lot of work & really slow!

Yes, But...

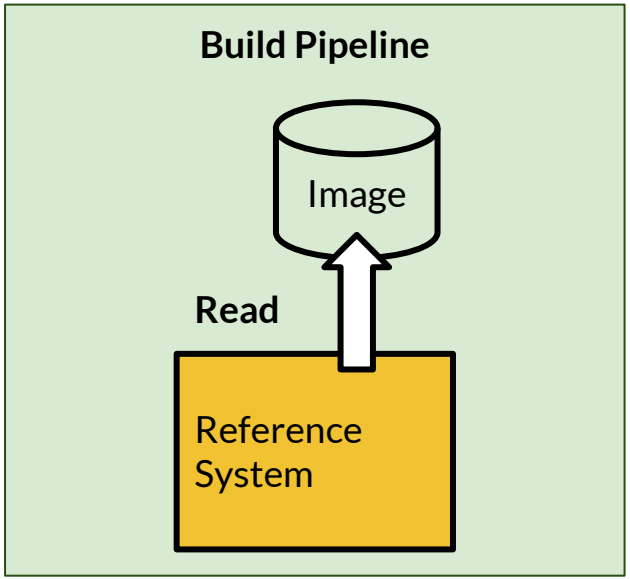
It's faster, safer & more scalable.

RackN

Immutable Demo

RackN

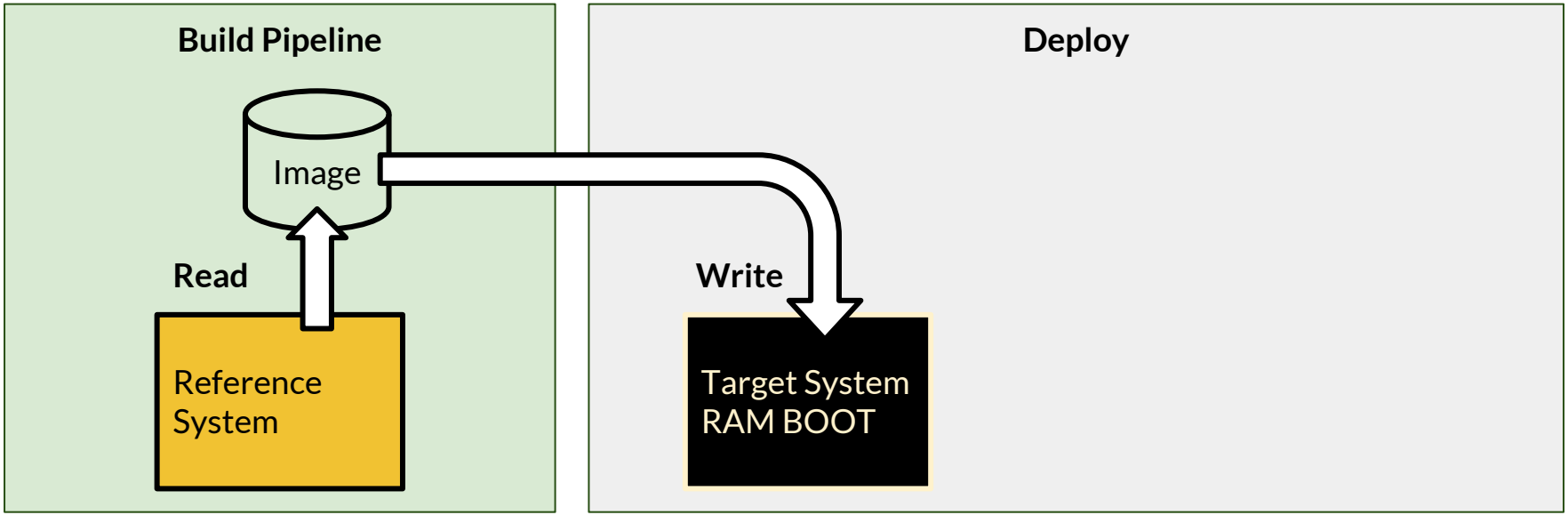
Prep: Image is pre-created from reference system.



Immutable Demo

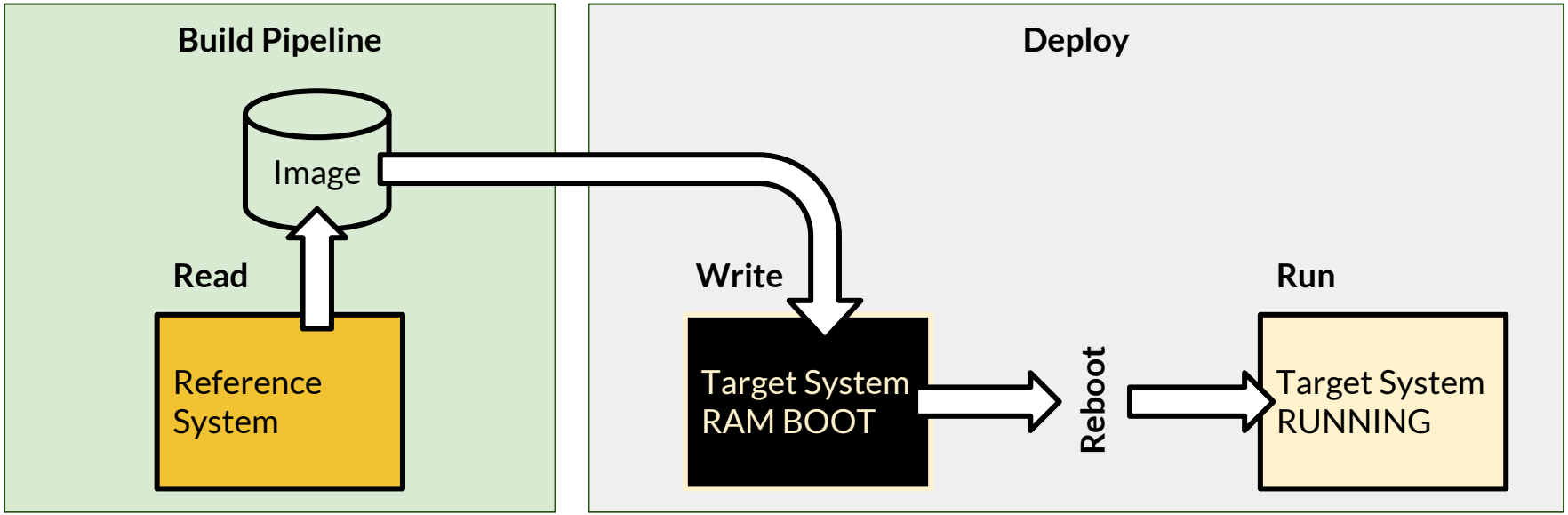
Prep: Image is pre-created from reference system.

Stage: Boot RAM image and write image to disk(s)



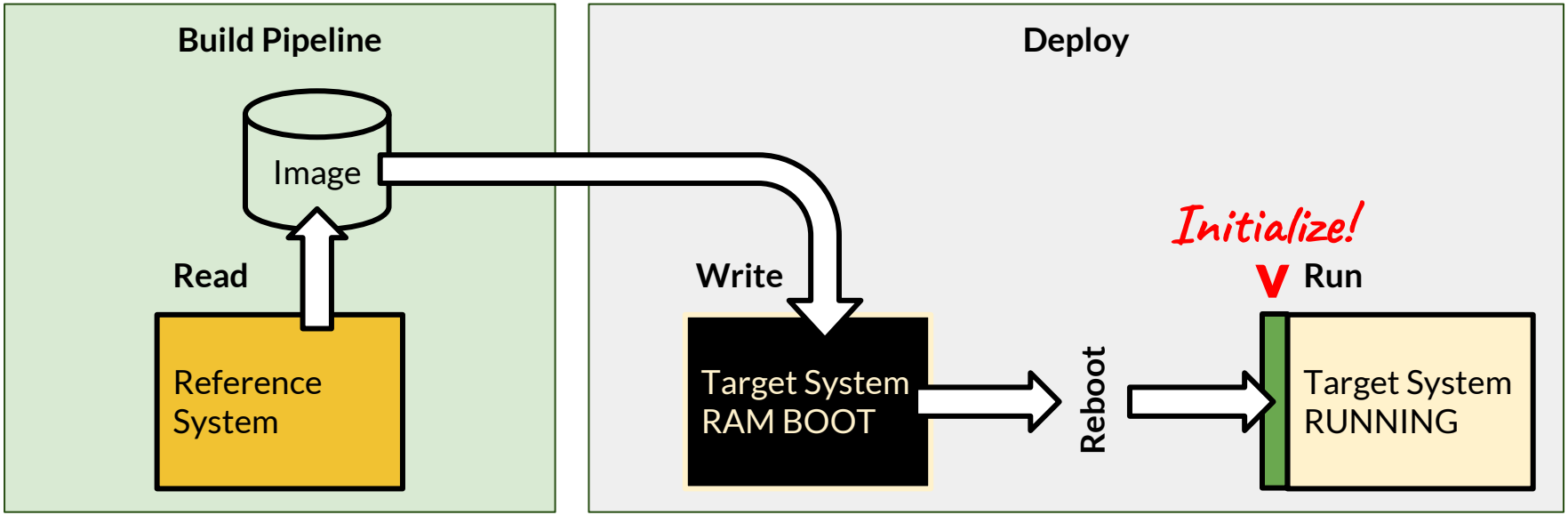
Immutable Demo

Prep: Image is pre-created from reference system.
Stage: Boot RAM image and write image to disk(s)
Deploy: Reboot and run



Immutable Demo

Prep: Image is pre-created from reference system.
Stage: Boot RAM image and write image to disk(s)
Deploy: Reboot and run



Thank you!

Questions?

Interested in IMMUTABLE METAL?

It's complicated, but we can get you there.

Start at http://portal.rackn.io

- Quickstart takes about 30 minutes
- Use your own hardware, VirtualBox or Packet.net account
  - `use "RACKN100" on Packet.net for credit`

RackN