


# From Push to Pull

## Managing Mutable Infrastructure at Scale

Holly Micham-Mooneyham, Cisco Meraki

 hmooneyh@cisco.com

 @TrailsThroughTheSystem@hachyderm.io



# Defining Some Terms

## Immutable Infrastructure

- Deploy components
- **Replace** them when things change
- Manage the churn

## Mutable Infrastructure

- Deploy components
- **Update** them when things change
- Manage the drift

# How NOT to Build a Better Mousetrap



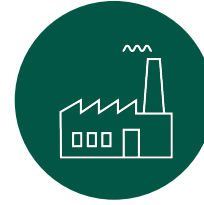
**How did we even get here?**



**What do we actually need?**



**What's the minimum viable product?**



**How do we keep momentum?**

# Who are we?



# When I joined Meraki in 2015...

We had **98 engineers**

We had **405 physical servers**

Meraki was based around  
**mutable** infrastructure

# Today in 2023...

We have **more than 10x** the engineers

We have **more than 10x** the compute

Meraki is still based around  
**mutable** infrastructure

# Why Mutable Infrastructure?

Wouldn't immutable be better at scale?



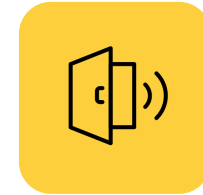
## Social Change is Hard

- People like their workflows
- Requires building consensus



## We Can't Change the Past

- Pragmatic decisions got us here
- Technical debt is expensive



## Change takes Time

- Competing priorities
- Can't stop all non-SRE work




**We want things to be great. But we have to be good at what we're doing now.**

# From Push to Pull

## Managing Mutable Infrastructure at Scale

Holly Micham-Mooneyham, Cisco Meraki

 hmooneyh@cisco.com


 @TrailsThroughTheSystem@hachyderm.io

# How did we even get here?







- 
- 1. Change is constant**
  - 2. Needs evolve**
  - 3. Complexity is inevitable**

“Why This Stuff is Hard” by Lorin Hochstein, SRECon Americas 2023



# Evolution of a Deploy System



**Capistrano**

**pre-2016**



**ANSIBLE  
+ Capistrano**

**late 2017**



**ANSIBLE  
+ Jenkins**

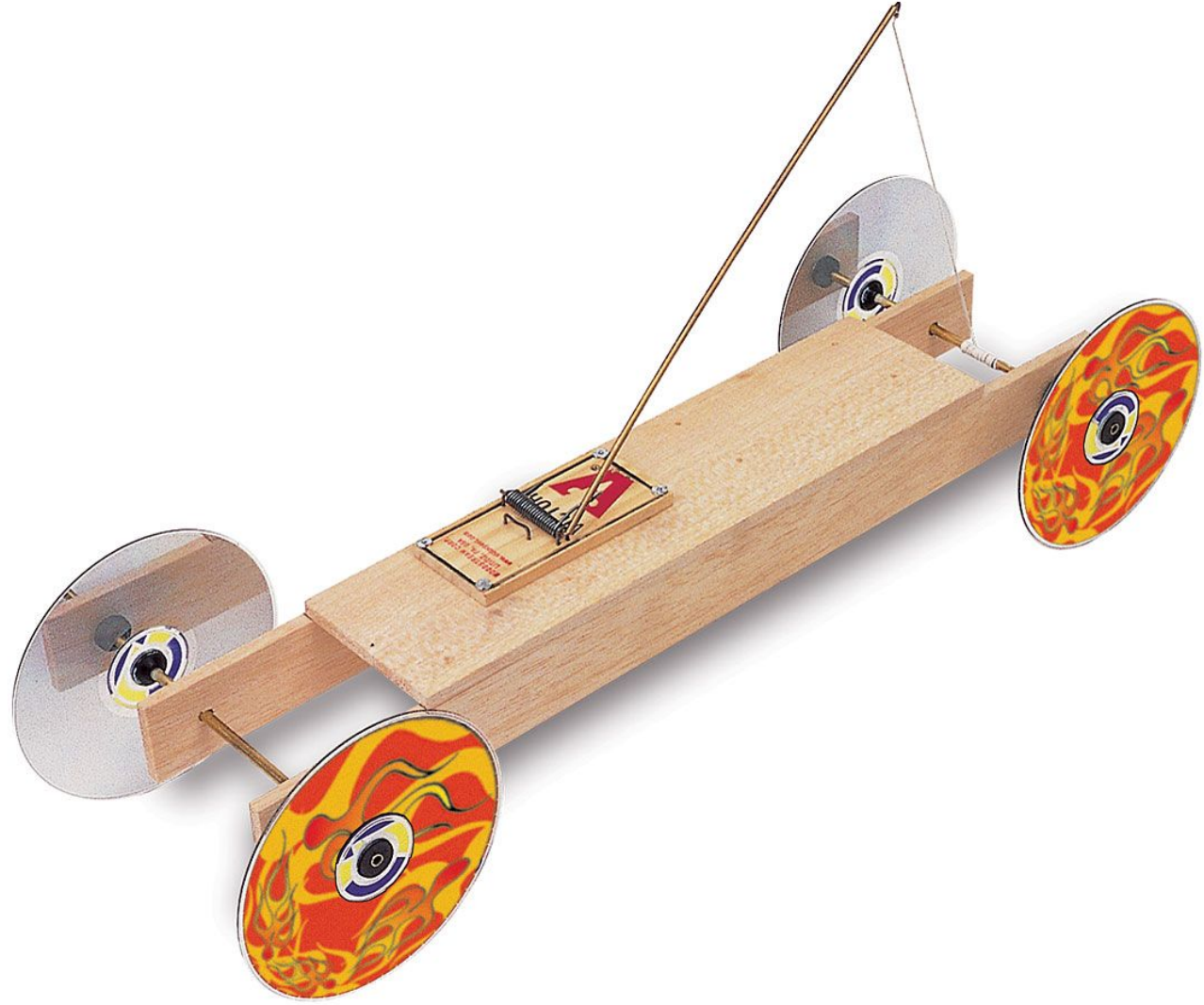
**early 2019**



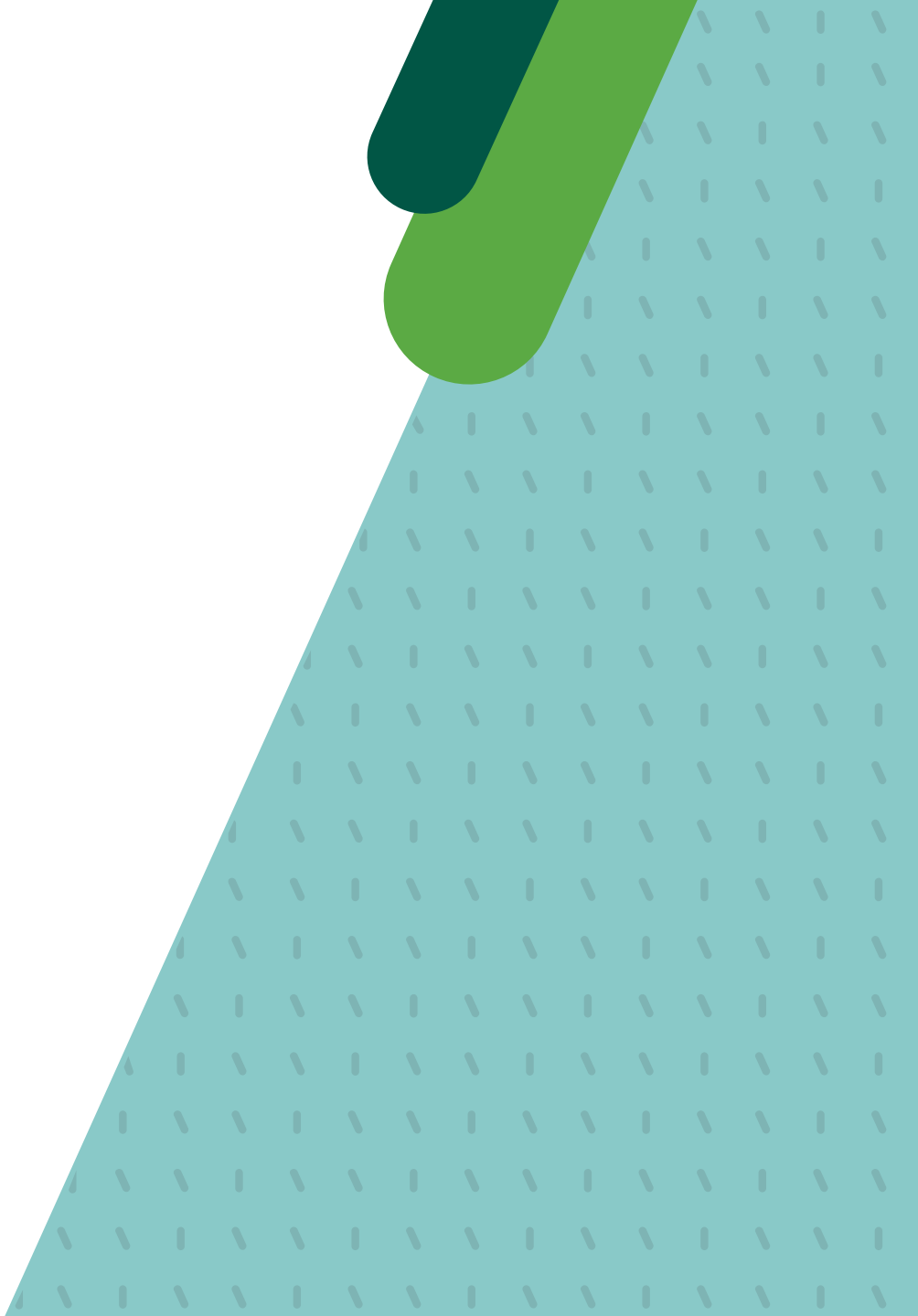
**ANSIBLE  
+ Gitlab**

**mid 2020**





# The Problem of Constraints



# What about Existing Systems?

Too much focus on Kubernetes and feature gaps around mutable infrastructure

Challenges integrating with our technical-debt laden systems, especially inventory

The search for the “perfect” system that does not exist


# Push Systems as a Mousetrap

## Push Systems have Benefits

- Fine grained control
- Natural prioritization
- Developers can coordinate

## They also Don't Scale

- Synchronous locking of all resources
- Unreproducible production state
- Vulnerable to drift



**“If SRE need the lock for any reason there’s a very good chance that you’ll have to reschedule”**

# Error Handling

```
10847 Failure Reasons:
10848 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10849 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10850 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10851 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10852 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10853 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10854 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10855 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10856 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10857 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10858 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10859 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10860 [REDACTED] failed on task 'apply new iptables rules': '[u'bundle', u'exec', u'./script/update_iptables', u'--no-prompt']'
10861 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10862 [REDACTED] failed on task 'deploy_release : get current svn revision': 'Timeout (62s) waiting for privilege escalation prompt: '
10863 Sunday 11 June 2023 16:08:13 -0700 (0:35:31.179) 0:55:11.822 *****
10864 =====
```



# What do we actually need?





# Breaking Change Budget

# Breaking Change Budget

## Earned By

- Addressing pain
- Fixing high-interest technical debt
- Providing net-new features
- Building trust

## Spent on

- Social change
- Changing people's workflows
- Removing constraints

# User Experience Interviews

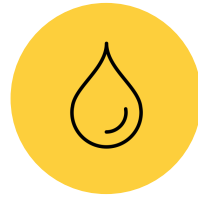
# Interview Goals



**Discover our  
users needs**



**Map out  
constraints**




**Understand  
current pain**




**Make our customers  
voices heard**

# What's the actual problem?



**“We need to scan  
containers during or before  
deployment.”**



~~**“We need to scan  
containers during or before  
deployment.”**~~

**“We need to prevent  
introducing new  
vulnerabilities.”**



# Interview Methodology



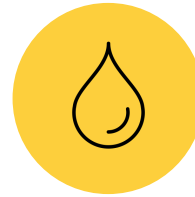
## **4 questions about current experience**

2 Positive questions  
2 Negative questions



## **2 questions about requirements**

To discover descriptive needs



## **2 implementation questions**

To help inform our tooling choices



## **1 open ended question**

To ensure our users voices are heard



# Learnings

## Features

- Rollout strategies
- Feature flags
- Visibility

## Constraints

- Time zones
- Technology choices
- Exceptional cases

## Pain

- Reproducibility
- Agility/speed
- Synchronicity
- Documentation

# Unavoidable Constraints

# Guideposts

DevOps Research and Assessment Report: Elite



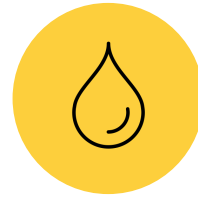
## Frequency

On-demand



## Lead Time

< 1 hour



## Time to Restore

< 1 hour



## Change Failure Rate

0%-15%

# Putting it all together



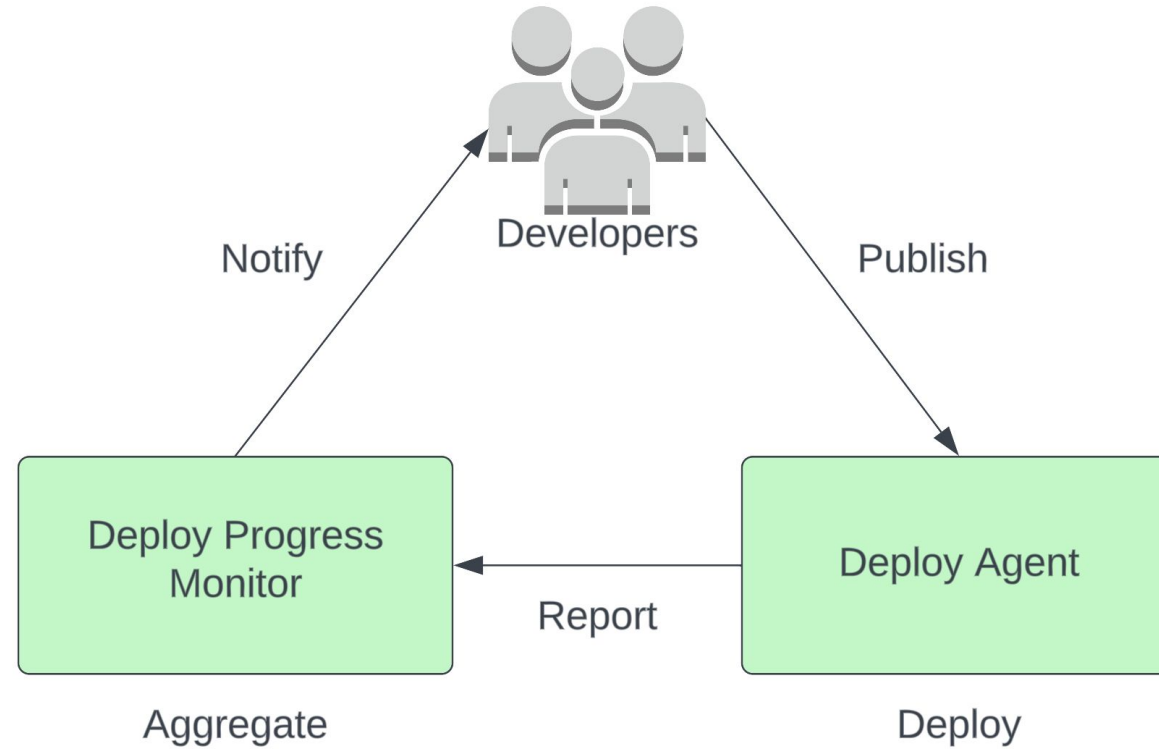
# Firm Design Decisions

Eventual consistency

Declarative state

Core toolchains

# Earliest Diagram



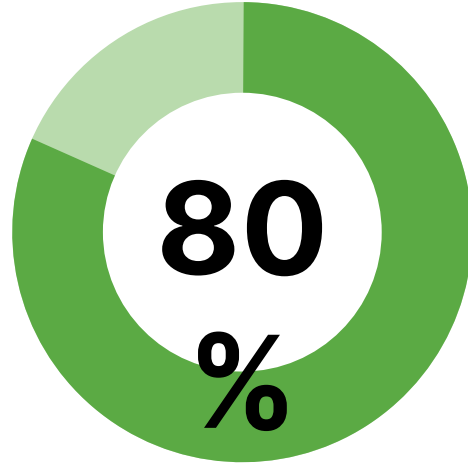
# Flexibility

What people do with the system

How people interact with the system

Support the unpaved road

# The 80/20 Rule



**Pave the road that handles  
80% of our traffic**

**Make sure there is a road for  
the other 20%**



# What's the minimum viable product

# MVP as a process

# Use Cases Supported

**1**

Early Experiments

**10%**

Alpha

**40%**

Open Beta

**80%**

General Availability



# Let's build a system

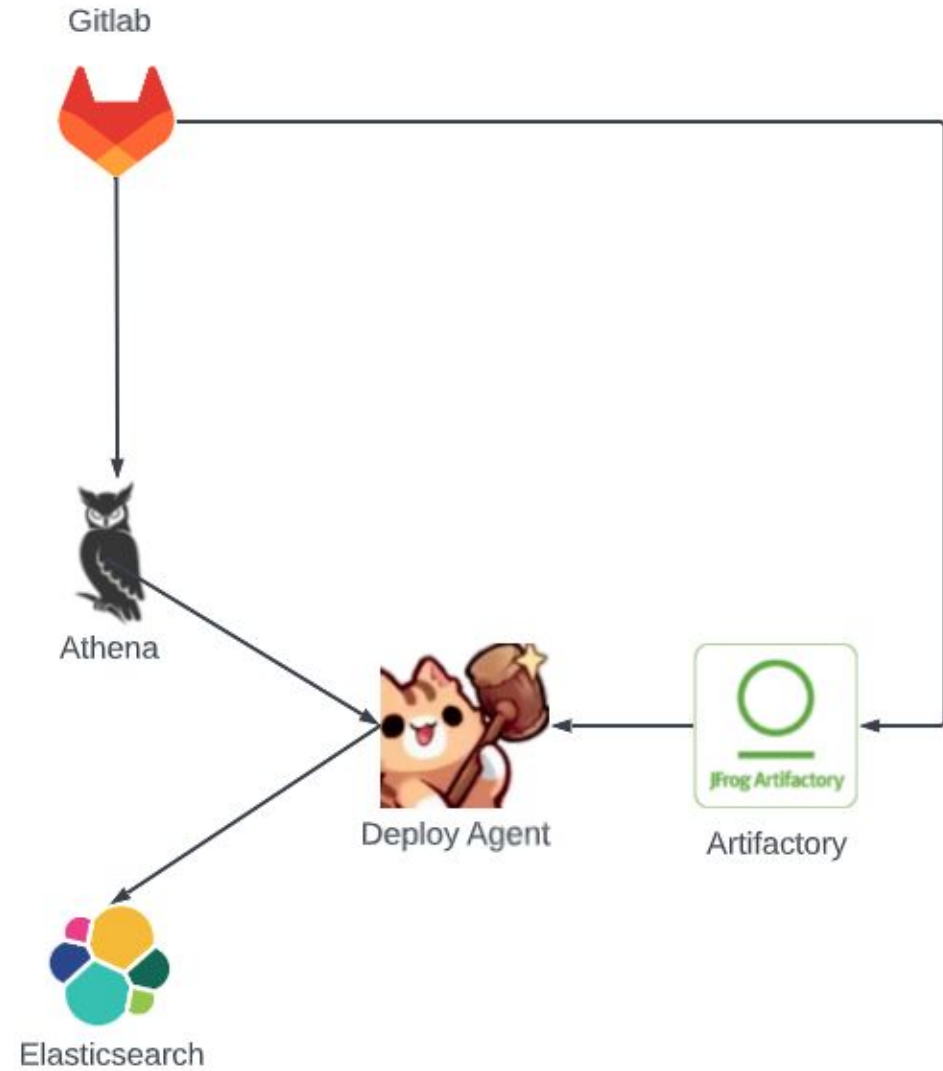
# Early Experiments

# Athena



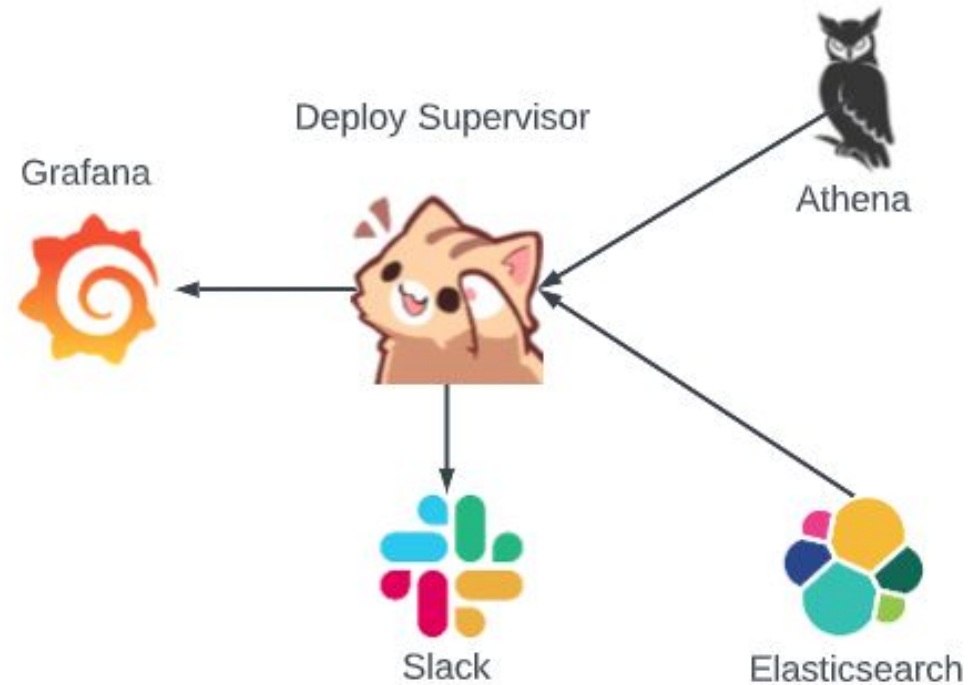
# Deploy Agent

1. Read Intended State
2. Make it So
3. Report State



# Deploy Supervisor

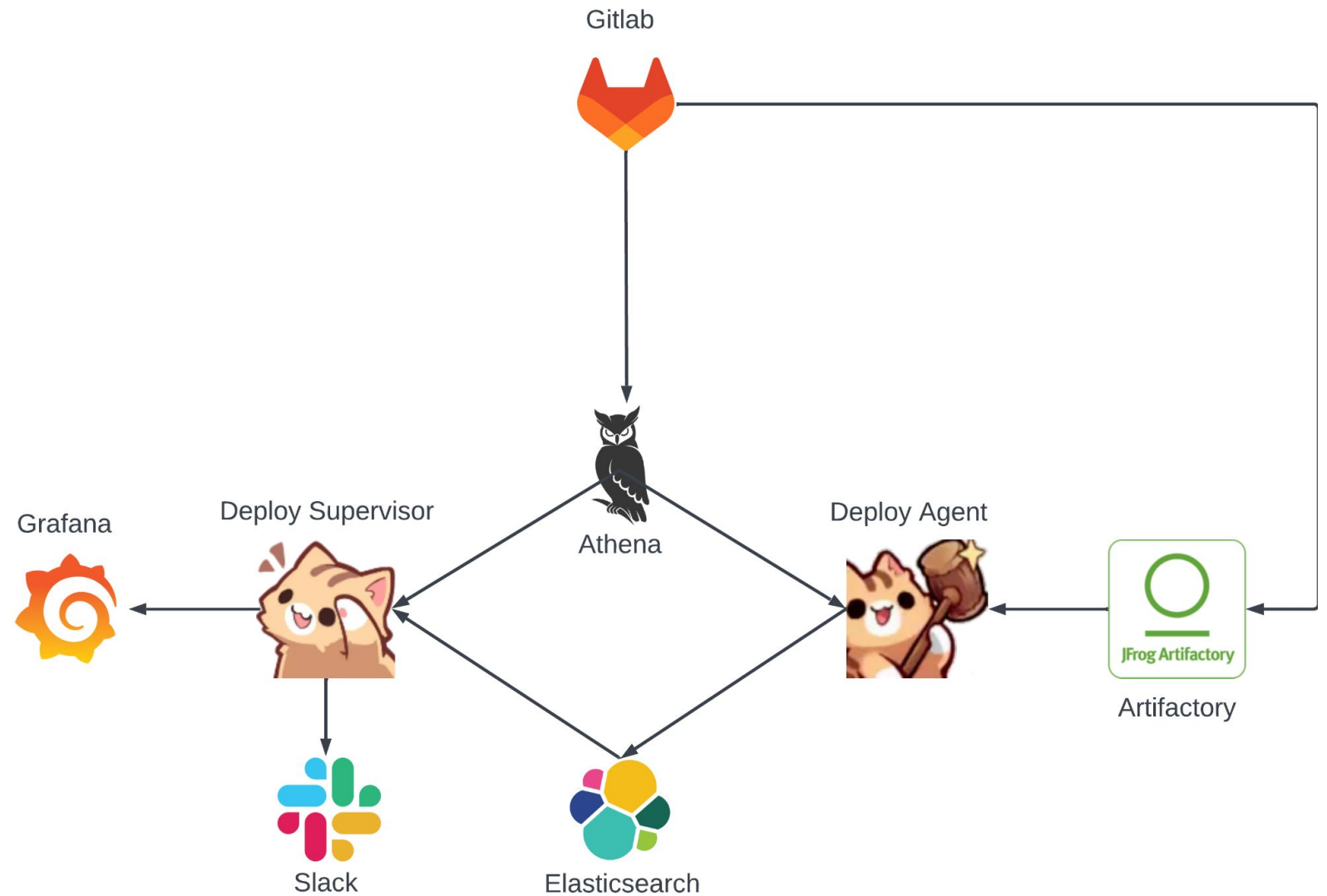
1. Read Intended State
2. Read Current State
3. Report to Users



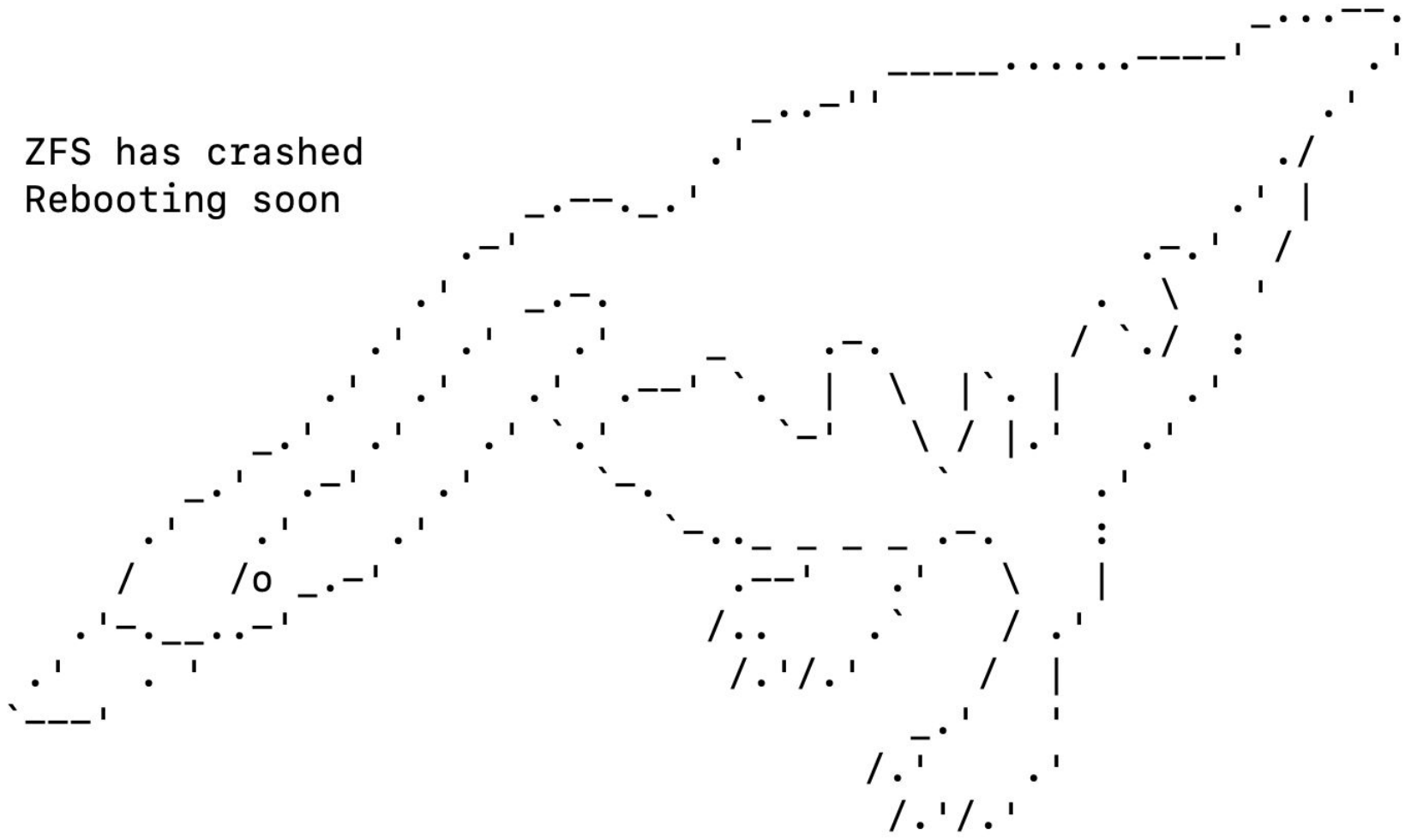


# Refactor #1

- Sustainability
- Hygiene
- Future-proofing



ZFS has crashed  
Rebooting soon





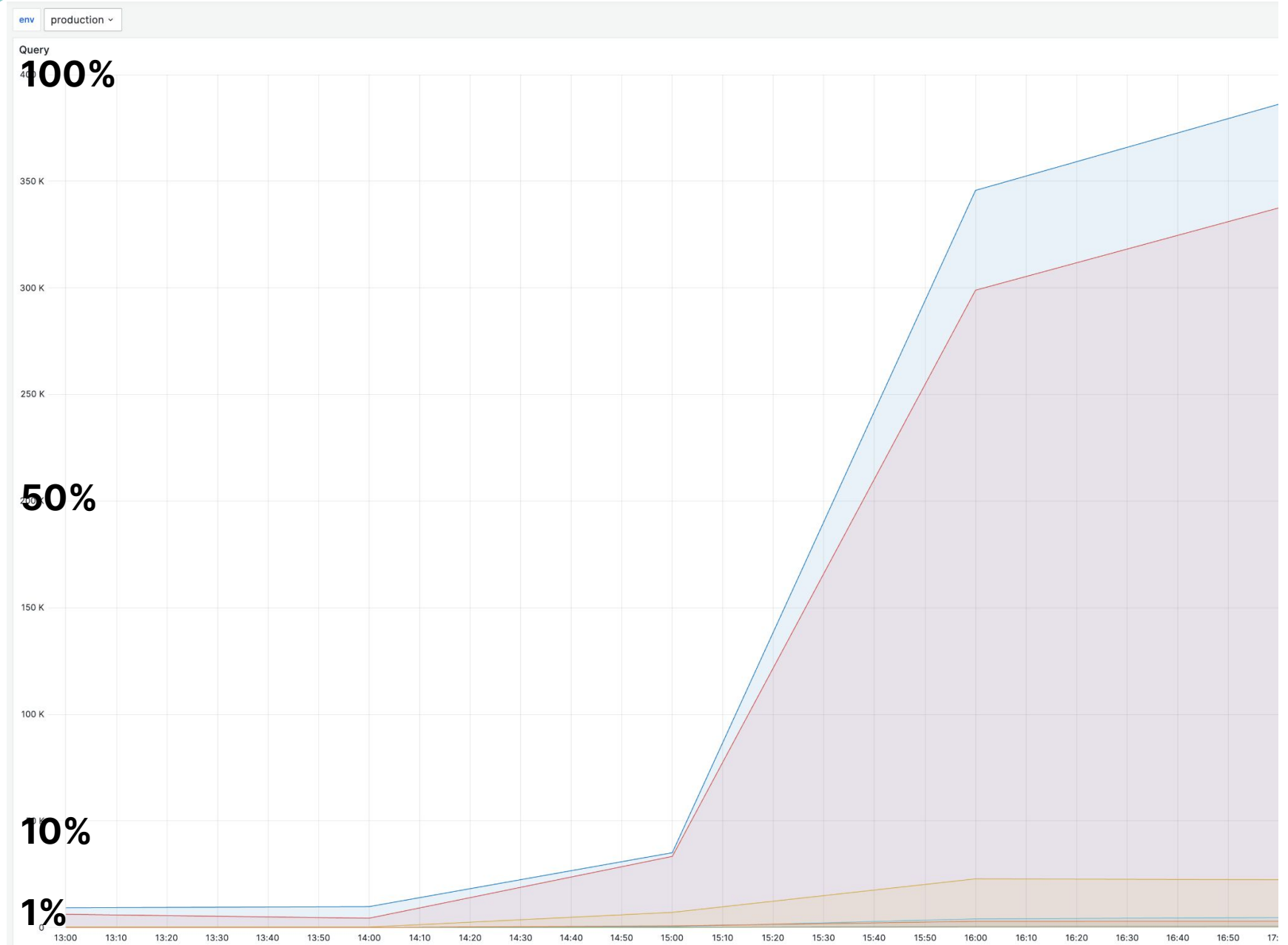
# Alpha



**“FYI we're firing up the new  
deploy system and it's  
about to put a bunch of load  
on Athena.”**

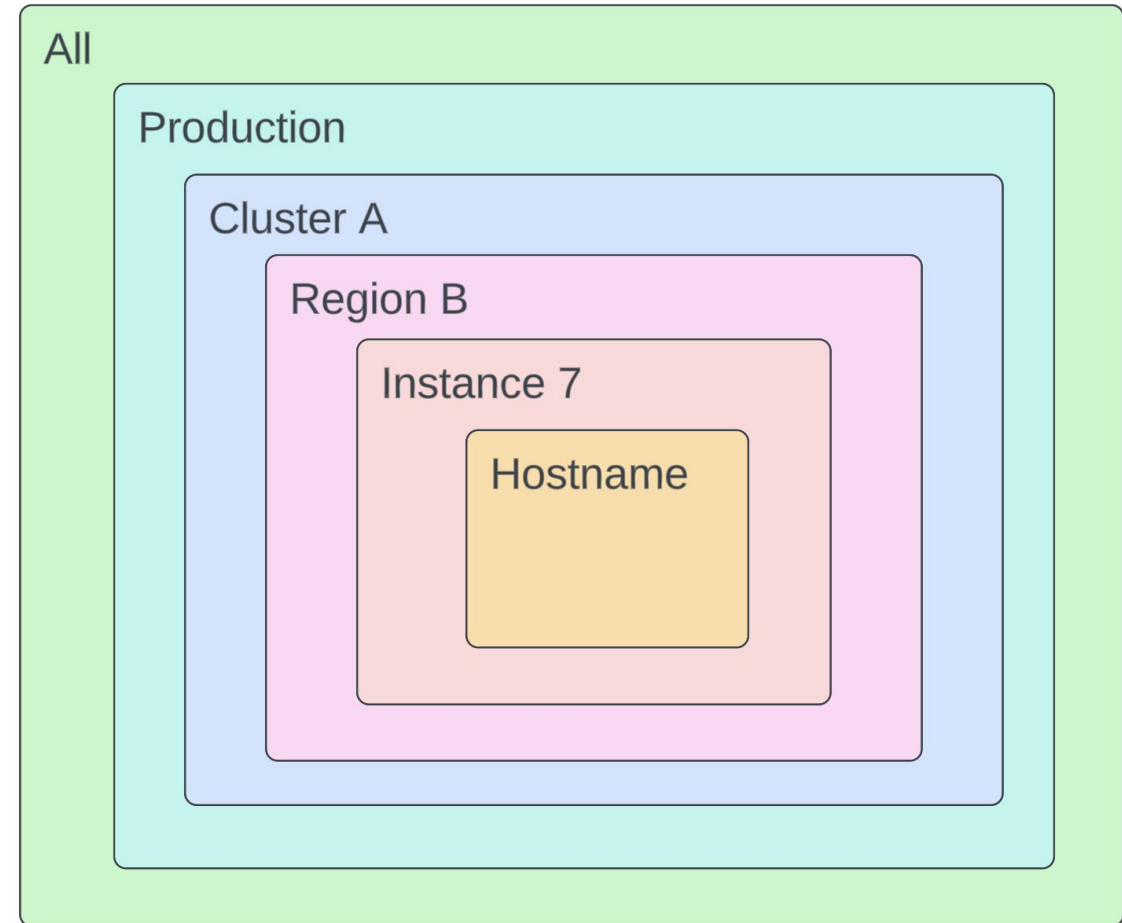


# Athena Load



# Targeted Deployments

- #1 *descriptive* need identified during customer interviews
- 3 competing use cases
- completely rewritten 4 times
- has the most tests of any part of the system



# Paved vs Unpaved Roads



# Deploy API

## Anatomy of a Deploy Script

### FIELD\_ORDER

Constant defining custom Athena selectors

### lookup\_\*

Param: target

Return the value of a custom Athena selector for a given target

Return: String

### version

Return the installed version on this machine

Return: String

### check\_consistency

Does the running version match the installed version?

Return: Boolean

### converge

Param: version

Reconfigure this machine to the specified version

### verify

Does the service work after deployment?

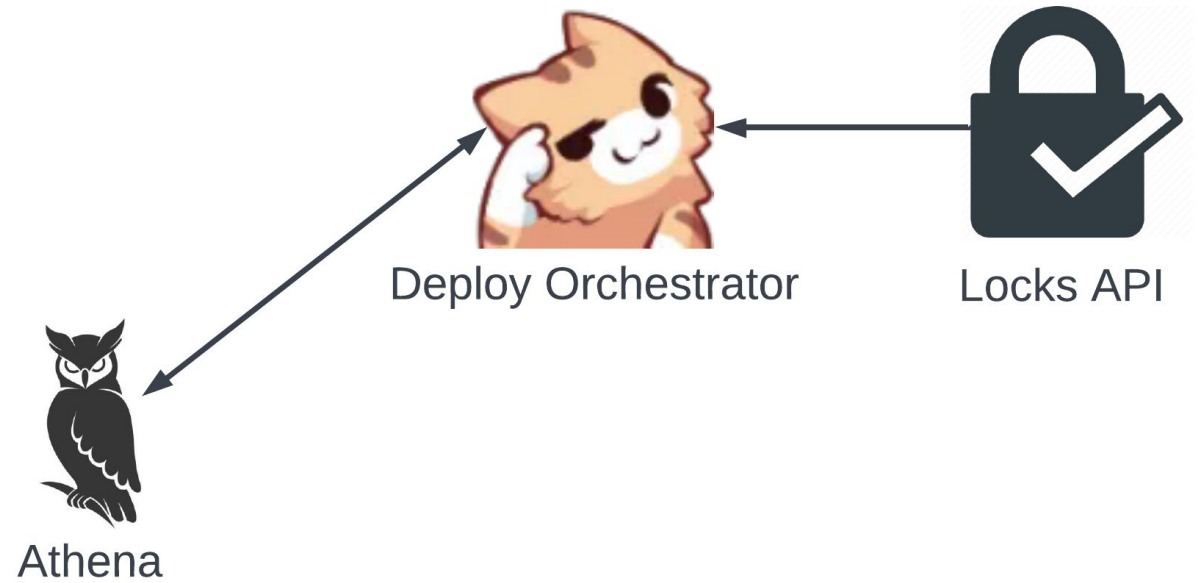
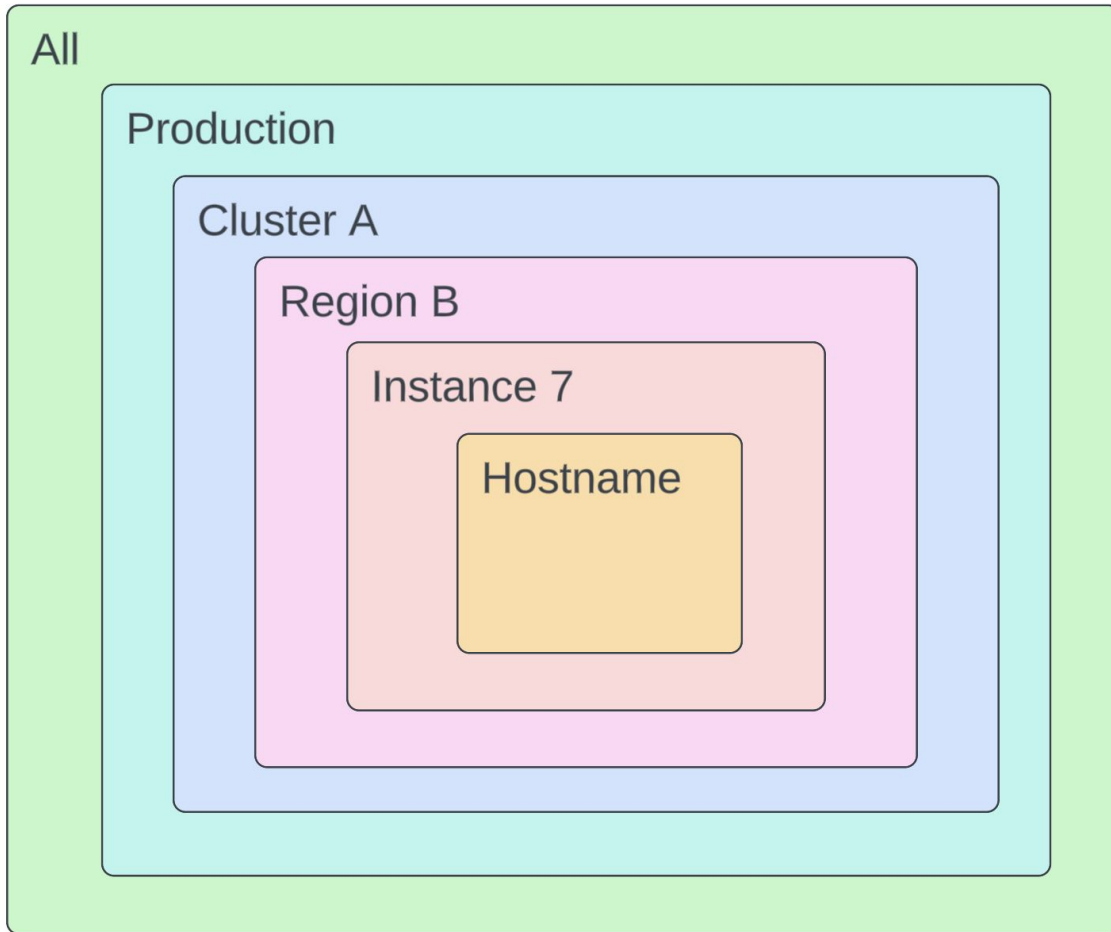
Return: Boolean

# Beta



# Avoiding complexity creep

# The Evolution of a Feature



# Blessings in disguise

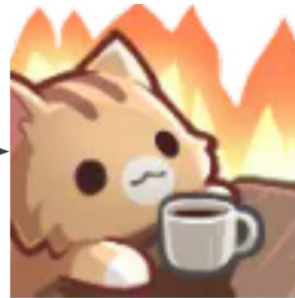
# GA and Beyond

# Host owner vs service owner

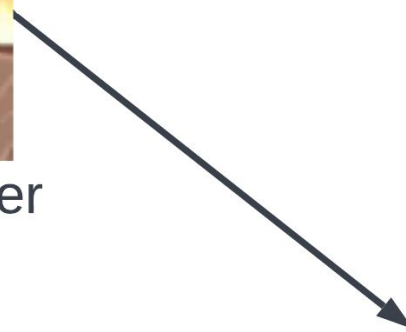
# Deploy Verifier

- Did everything break?
- Roll it back

NewRelic



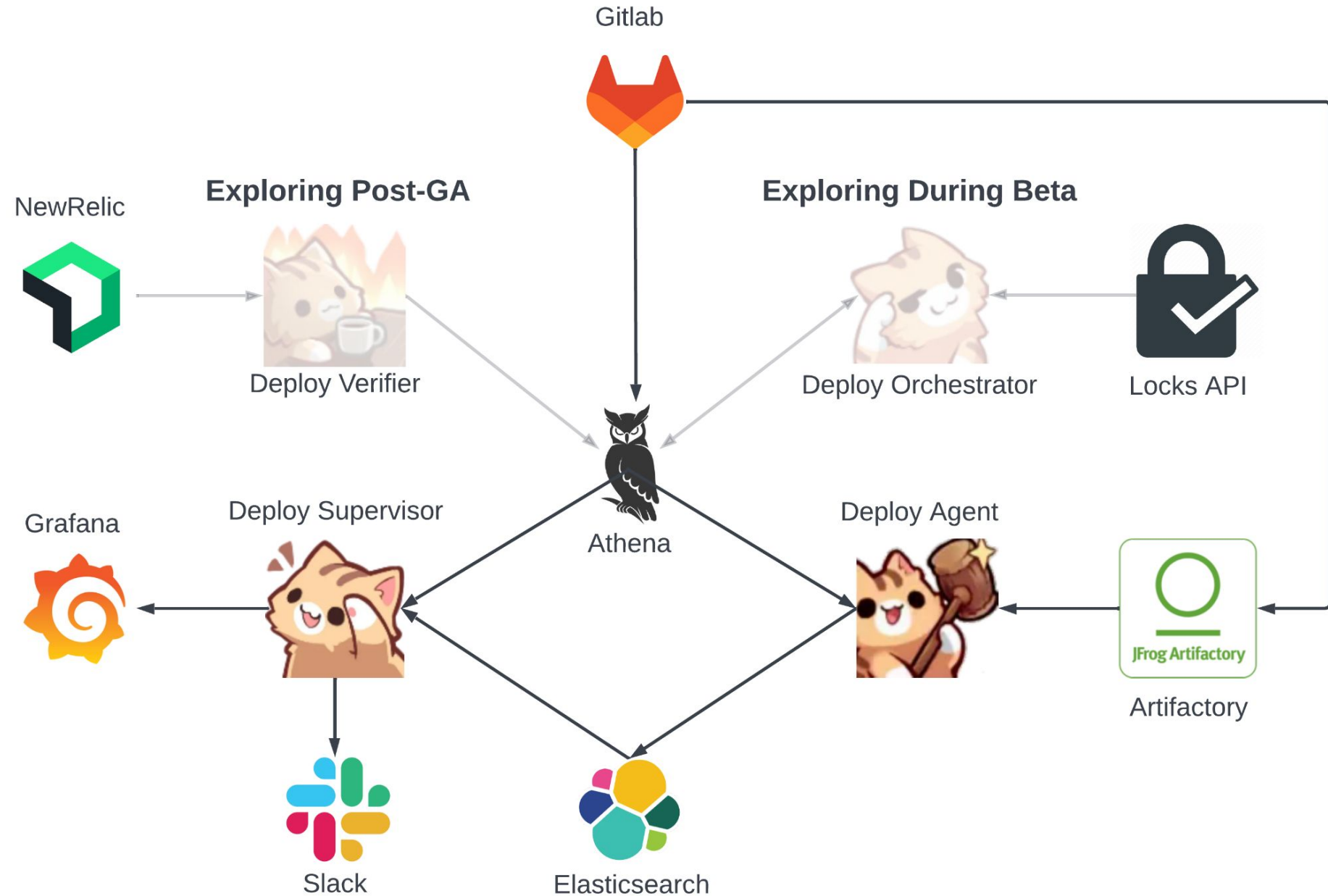
Deploy Verifier



Athena



# A Full System



# In Review



# How NOT to Build a Better Mousetrap



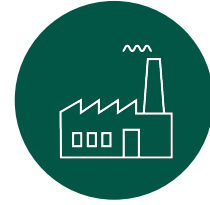
**How did we even get here?**



**What do we actually need?**



**What's the minimum viable product?**



**How do we keep momentum?**

# Building Stuff is Exciting!



# Tools for Making a Great System

**Question your constraints:** Understand how you got here, and then step outside your local perspective.

---

**Learn your breaking change budget:** Spend it in places where your constraints stop you from building what you need.

---

**Interview your customers:** Cut through the noise of prescriptive needs and learn their descriptive needs.

---

**MVP-as-a-process:** Identify your riskiest assumptions and continuously test with the minimal investment.

# Thank you!