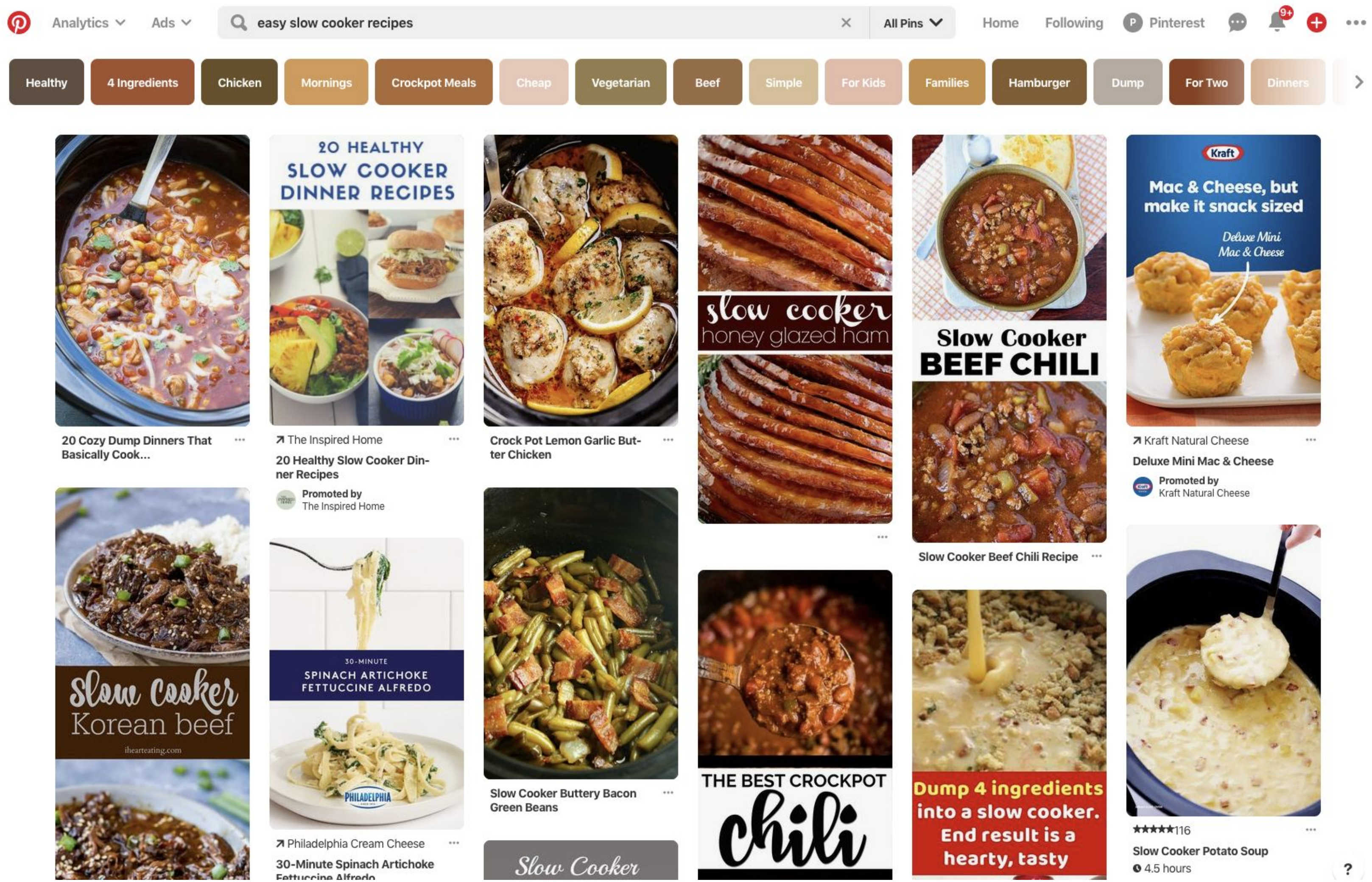
Invisible OS and Platform Upgrades

Adam McKenna, Site Reliability Engineer, Pinterest
@deathtocss
/in/admckenna
Pronouns: He, his



20 Cozy Dump Dinners That Basically Cook...

The Inspired Home

20 Healthy Slow Cooker Dinner Recipes

Promoted by
The Inspired Home

Crock Pot Lemon Garlic Butter Chicken

Kraft Natural Cheese

Deluxe Mini Mac & Cheese

Promoted by
Kraft Natural Cheese

Slow Cooker Beef Chili Recipe

Philadelphia Cream Cheese

30-Minute Spinach Artichoke Fettuccine Alfredo

Slow Cooker Buttery Bacon Green Beans

★★★★★116

Slow Cooker Potato Soup

4.5 hours

Our mission

To bring everyone the inspiration to create a life they love.

Core SRE @ Pinterest

- Overall Uptime
- Internal Services
- Tech Debt / “Ownerless” services

- Linux user since 1993, Sysadmin since 1998
- DevOps practitioner since 2012

Defining our problem

Software goes out of date

- **OS/Container Runtimes**
  - Ubuntu
    - 12.04, 14.04 EOL
  - Docker
    - Quarterly Updates
  - Windows
    - Windows 7 EOL (Pinterest not affected)

- **Language Runtimes**
  - Python
    - 2.x to 3.x
  - Java
    - Oracle to OpenJDK
    - Java version updates
  - Go
  - C++

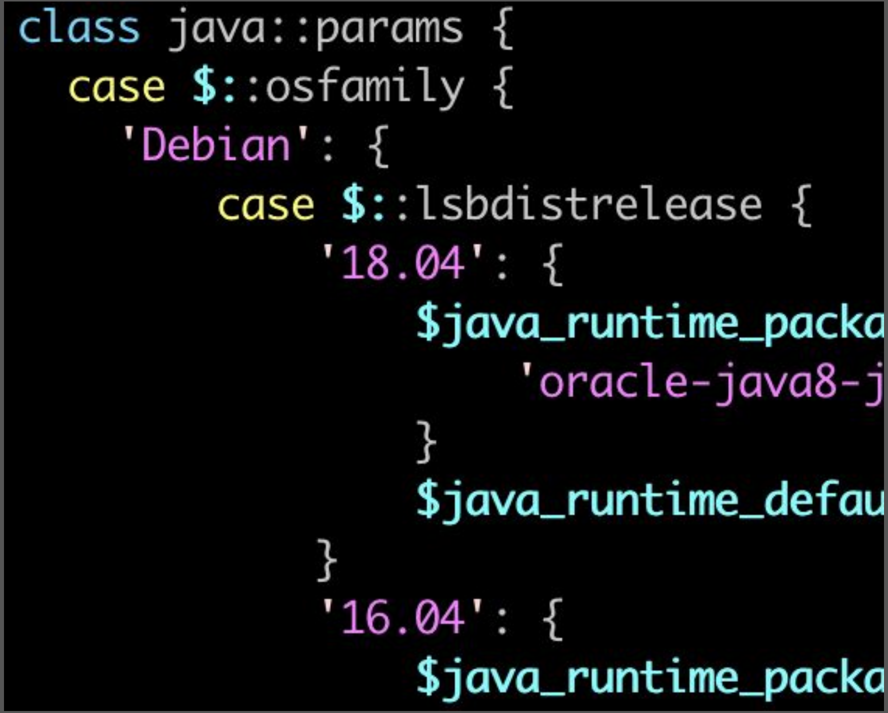
We Must Upgrade!

The business needs:

- **Retain support**
  - Security bug fixes
- **Access to new features**
  - New EC2 Instance Families/Generations
  - Hardware support
  - Performance
  - New storage technologies (Instance store -> EBS)
- **“Containerization War Stories” https://www.usenix.org/conference/srecon18americas/presentation/wong**

Developers want

- **Access to new packages / features**
  - Updated runtimes (Go, Python, Java, etc.)
  - GPU Drivers for AI/ML workloads
- **Platform Efficiency**
  - Desire to avoid supporting multiple OS or runtime versions



```
class java::params {
  case $::osfamily {
    'Debian': {
        case $::lsbdistrelease {
            '18.04': {
                $java_runtime_packa
                    'oracle-java8-j
                }
                $java_runtime_defau
            }
            '16.04': {
                $java_runtime_packa
```

But we hates upgrading!

- **Complexity**
    - Hundreds of microservices across tens of thousands of hosts
- **Service owners**
    - Don’t like downtime.
    - Migration work is generally not a preferred task
- **Developers**
    - Consistent API/ABI
    - Tests written for specific versions of language runtimes

Solution:

Automated Canary Analysis

Canary Analysis

Image credits: Michael Sonnabend (flickr)
https://pxhere.com/en/photo/1412043

What is ‘Canary Analysis’?

- Compare the behavior and performance of two clusters - a canary (test) cluster and a control cluster.
- Has inherent risk, but lower risk than full rollout of bad code.

https://en.wikipedia.org/wiki/Risk_matrix

Canary Analysis

Requirements and Goals

- **Automate and normalize the most mundane migration tasks**
  - Depend on successful integration and acceptance testing
  - Automate looking at charts, comparing metrics
  - Normalize - Process can be applied to many similar systems.

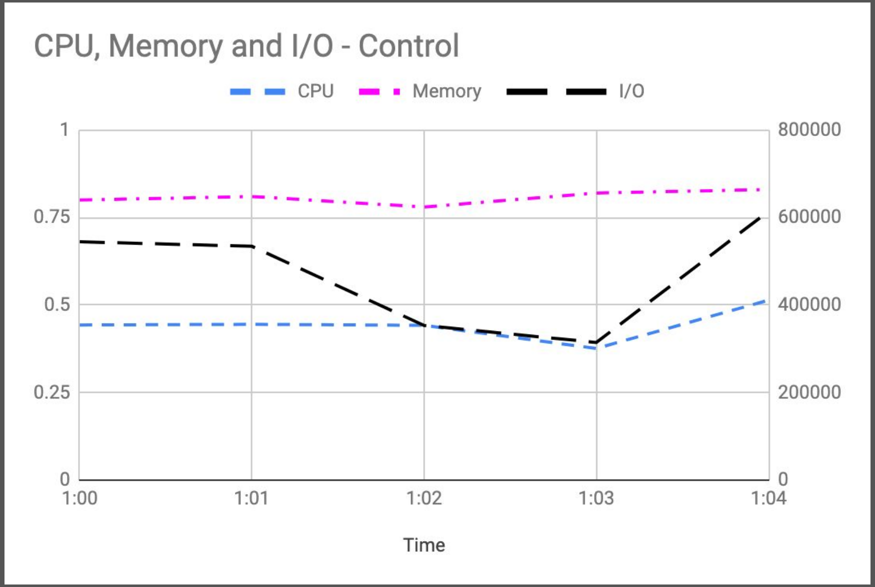
- **Make migrations 'invisible' to service owners**
  - Address reliability concerns; Services are analyzed at the cluster level.
  - Can be run by an operator, freeing up developer time
  - No more 'finger crossing'

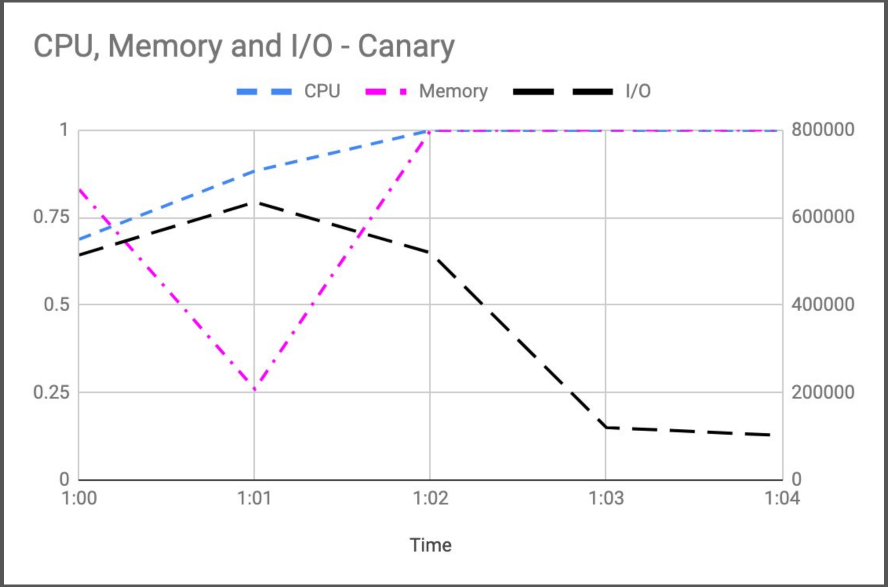
Typical migration ‘toil’ pattern

1. New Image is released
2. Service owner updates Image ID in deploy system
3. Operator triggers rolling cluster upgrade
4. Humans watch charts and cross fingers
5. Did things go OK? Good!
6. Did things not go OK? Bad! Rollback! Outage?

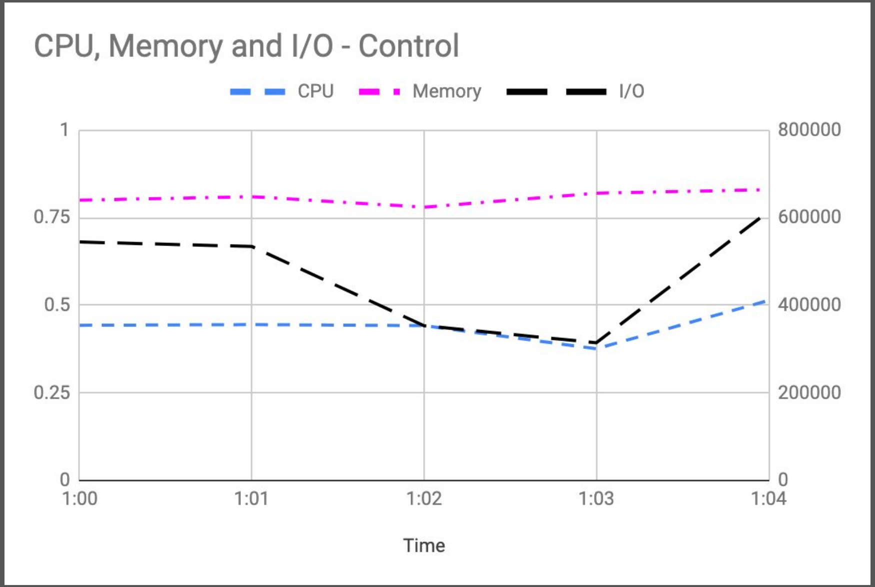
“Looks good to me!”

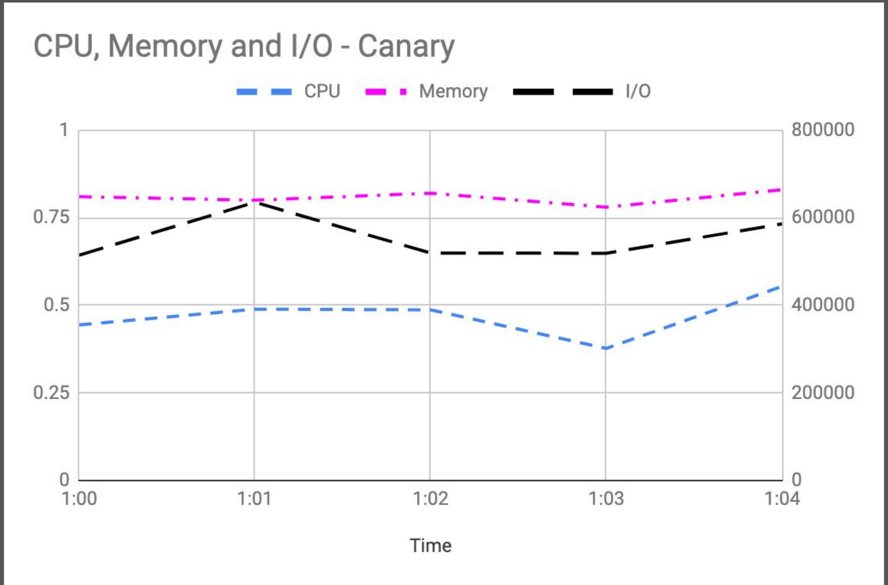
Can you make a Go / No-Go decision based on these two charts?





How about these?





Well, Let’s ask the service owners!

The human factor

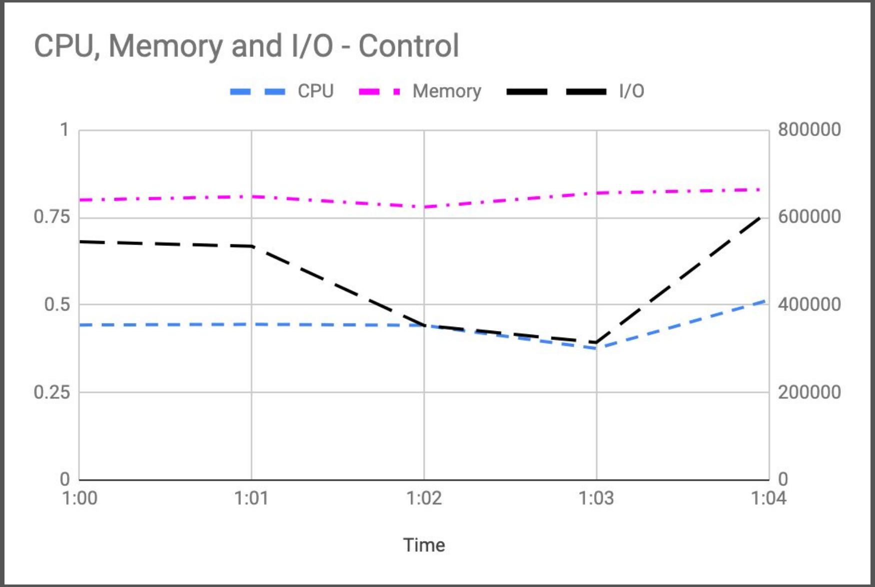
What might these Service Owners think?

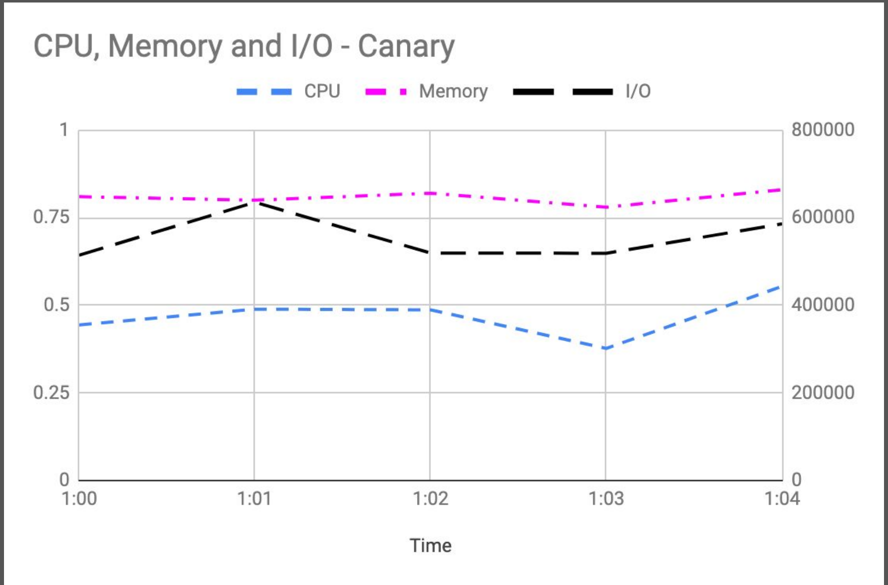
Gene

- Extremely Risk Averse
- Hates change
- His service has 99.999999% uptime, but runs on Ubuntu 10.04

Gene’s call

“No way. Look at that I/O line! Let’s meet on Monday and figure out what’s going on.”



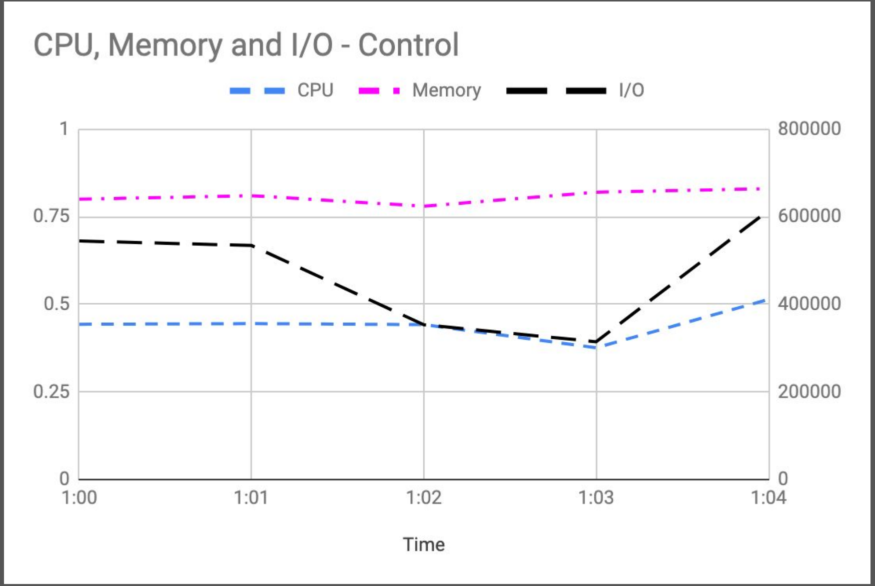


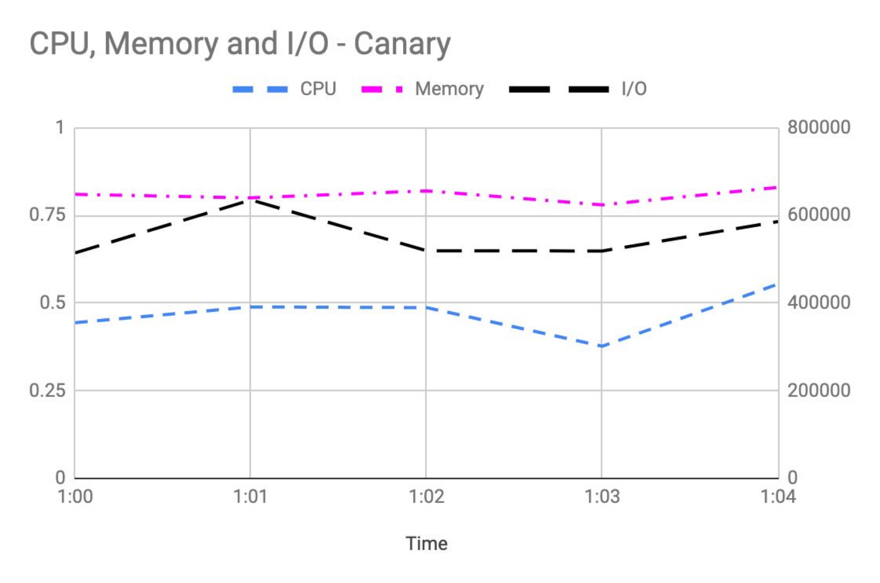
Angus

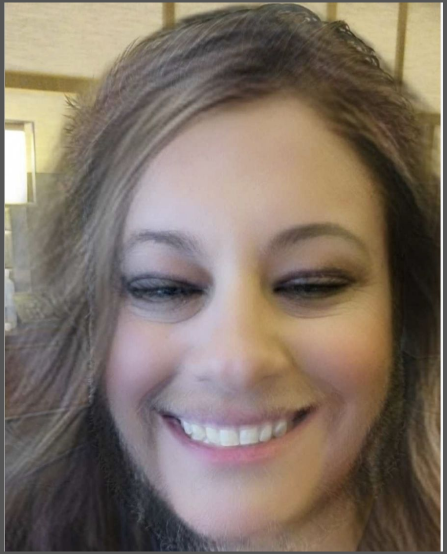
- Last job was at a Bitcoin startup
- Likes to “Move fast and break stuff”
- Yells “TONIGHT WE TEST IN PROD” at least once a day

Angus’s call

“LGTM. SHIP IT!”





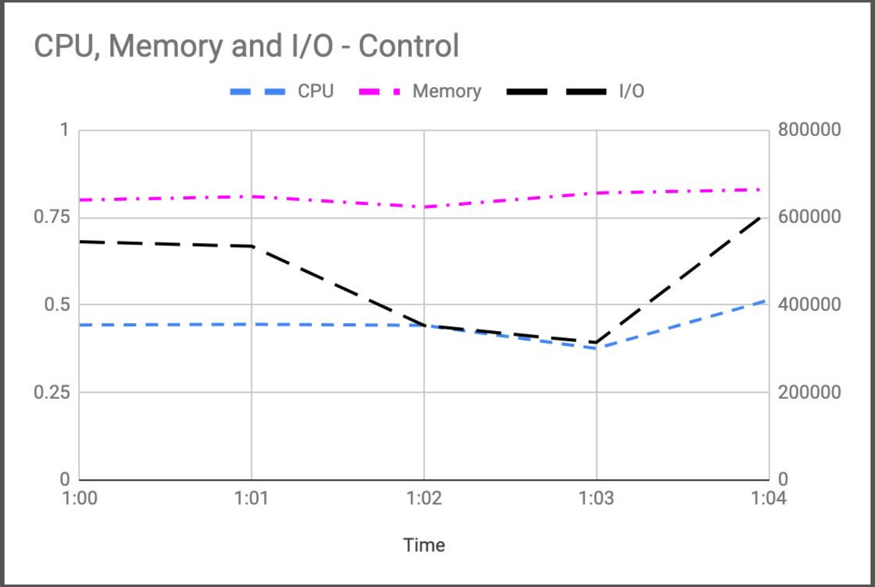


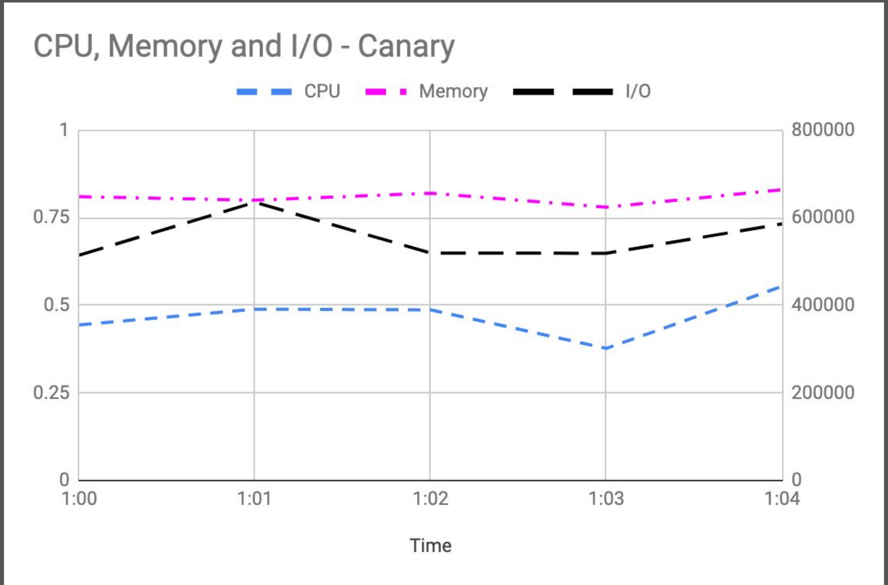
Carol

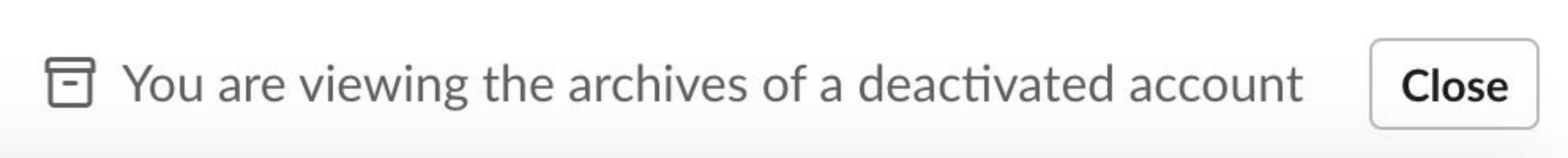
- Has 100% test coverage on all her code
- Is researching statistical models to improve fault detection
- Analysis is her middle name

Carol's call

"I'll need a couple hours to run some additional tests against the new hosts. Stand by."







You are viewing the archives of a deactivated account

Image credit: svgsilh.com

Karen

- **The employee who left the company**
- **All of the knowledge they didn’t write down about their service left with them!**

A better way - Automate!

Benefits of automated Canary Analysis

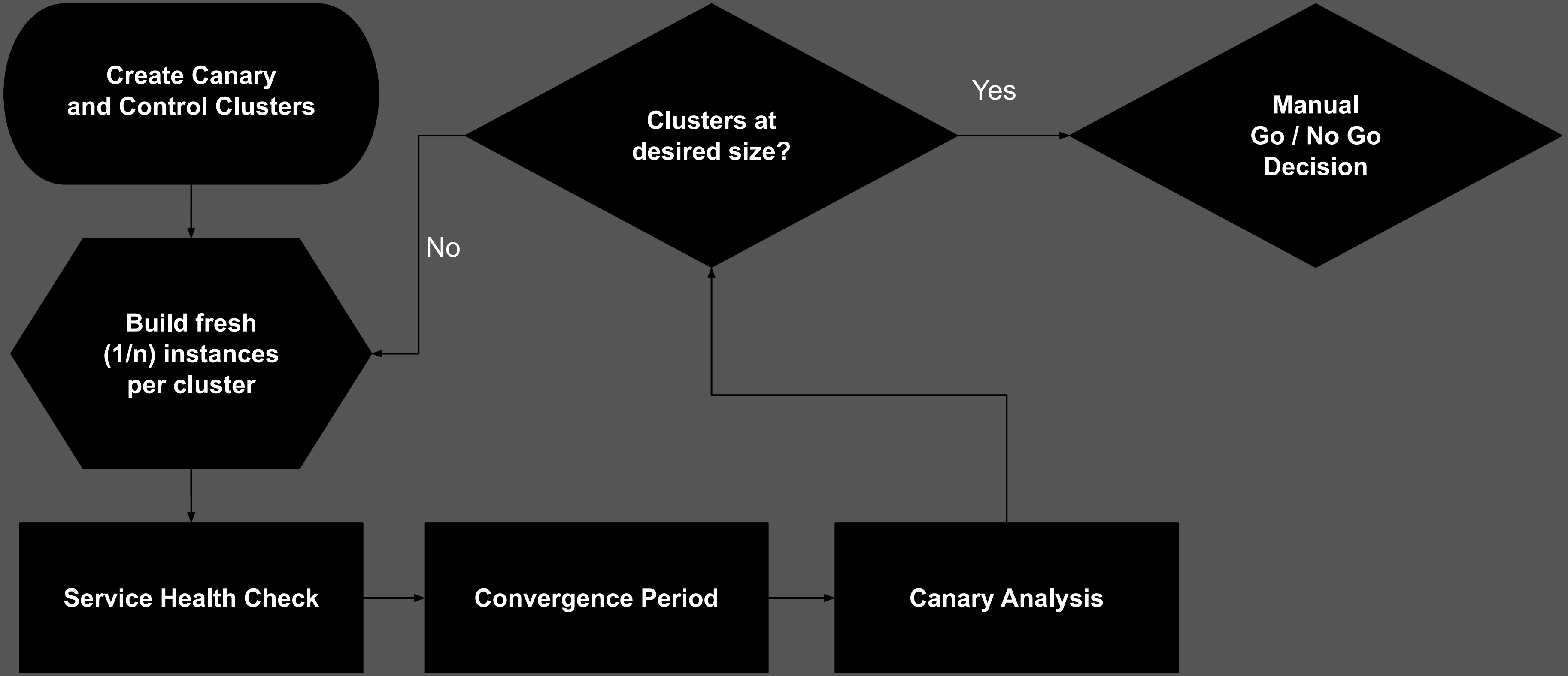
- **A software-defined, repeatable process for validation of new code**
- **Accumulated knowledge remains in the system as code, rather than leaving with employees**
- **Can be generalized/normalized with sane defaults**
- **(Optionally) continue to provide manual go/no-go approval (You know, for Gene)**
- **Bad results? Update the model! (In source control!)**

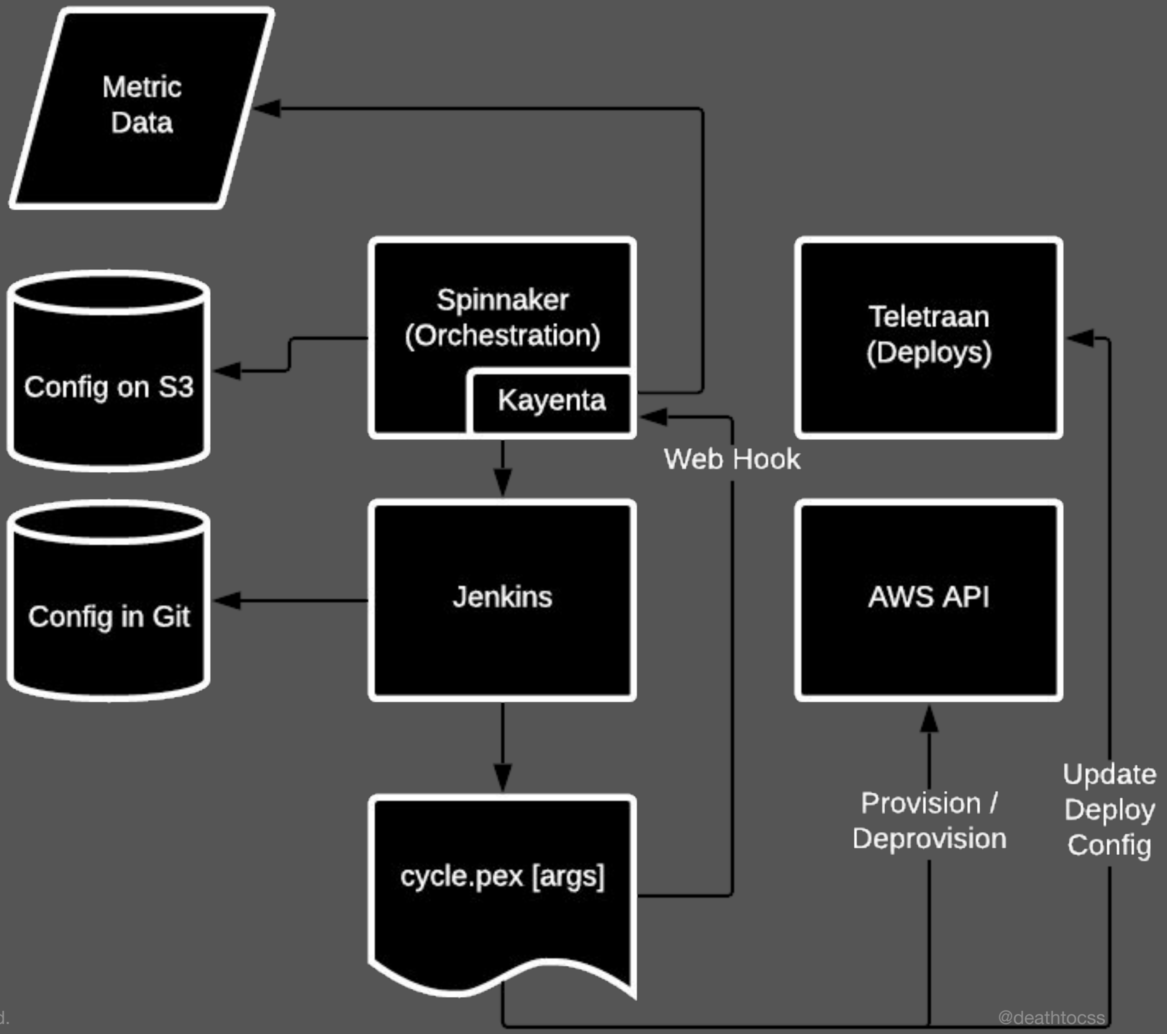
How it works

Components

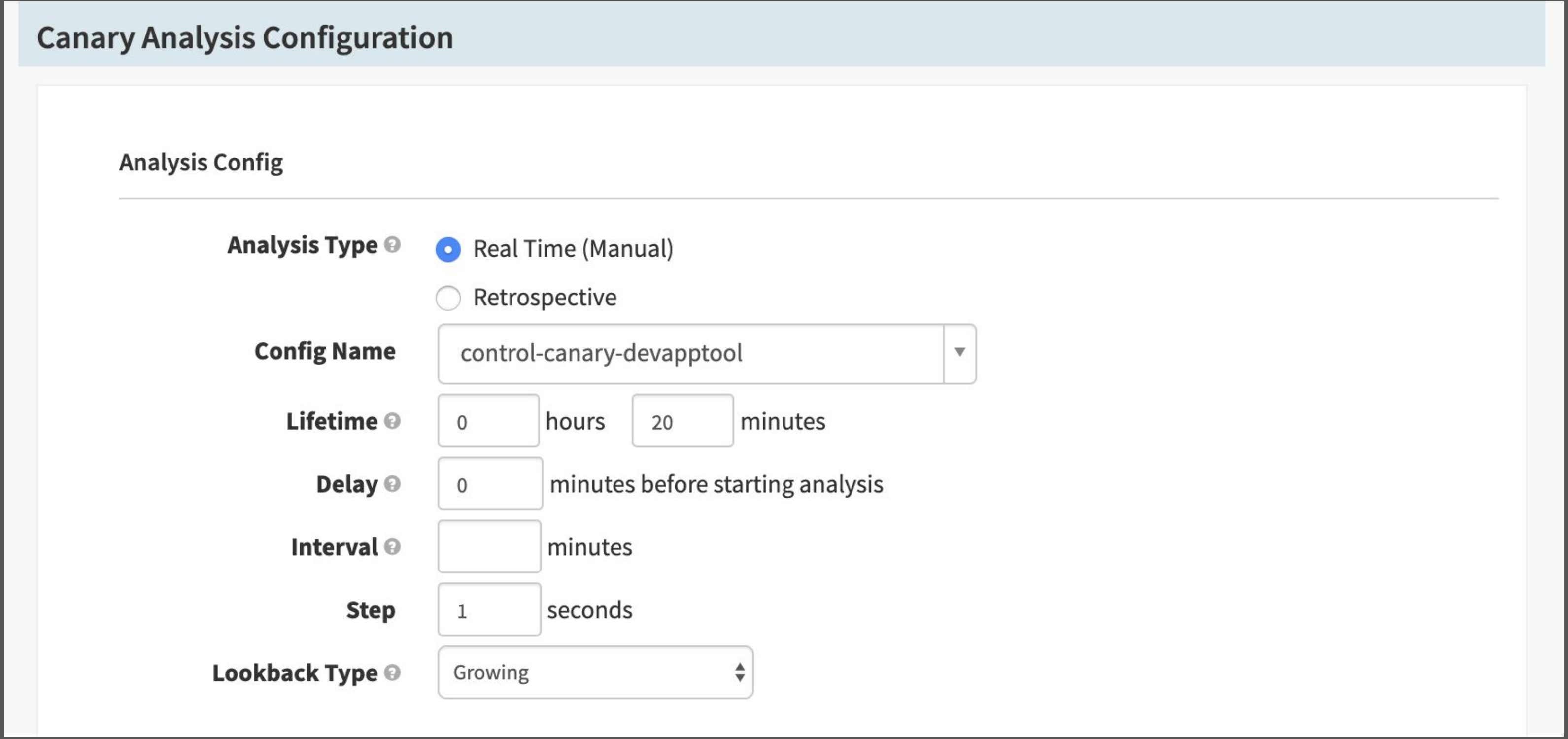
- Spinnaker + Kayenta
  - Workflow orchestration
    - https://www.spinnaker.io/
- Jenkins
  - Script execution
    - https://jenkins.io/
- Teletraan
  - Pinterest’s deploy system
    - https://github.com/pinterest/teletraan
- Docker
  - Execution environment
    - https://www.docker.io
- Python
  - Script logic implementation
    - https://www.python.org

Simplified Process Flow




Metric Data
Spinnaker (Orchestration)
Kayenta
Teletraan (Deploys)
Config on S3
Web Hook
Config in Git
Jenkins
AWS API
cycle.pex [args]
Provision / Deprovision
Update Deploy Config

Configuring the Pipeline



Canary Analysis Configuration

Analysis Config

Analysis Type ● Real Time (Manual)

○ Retrospective

Config Name: control-canary-devapptool

Lifetime: 0 hours 20 minutes

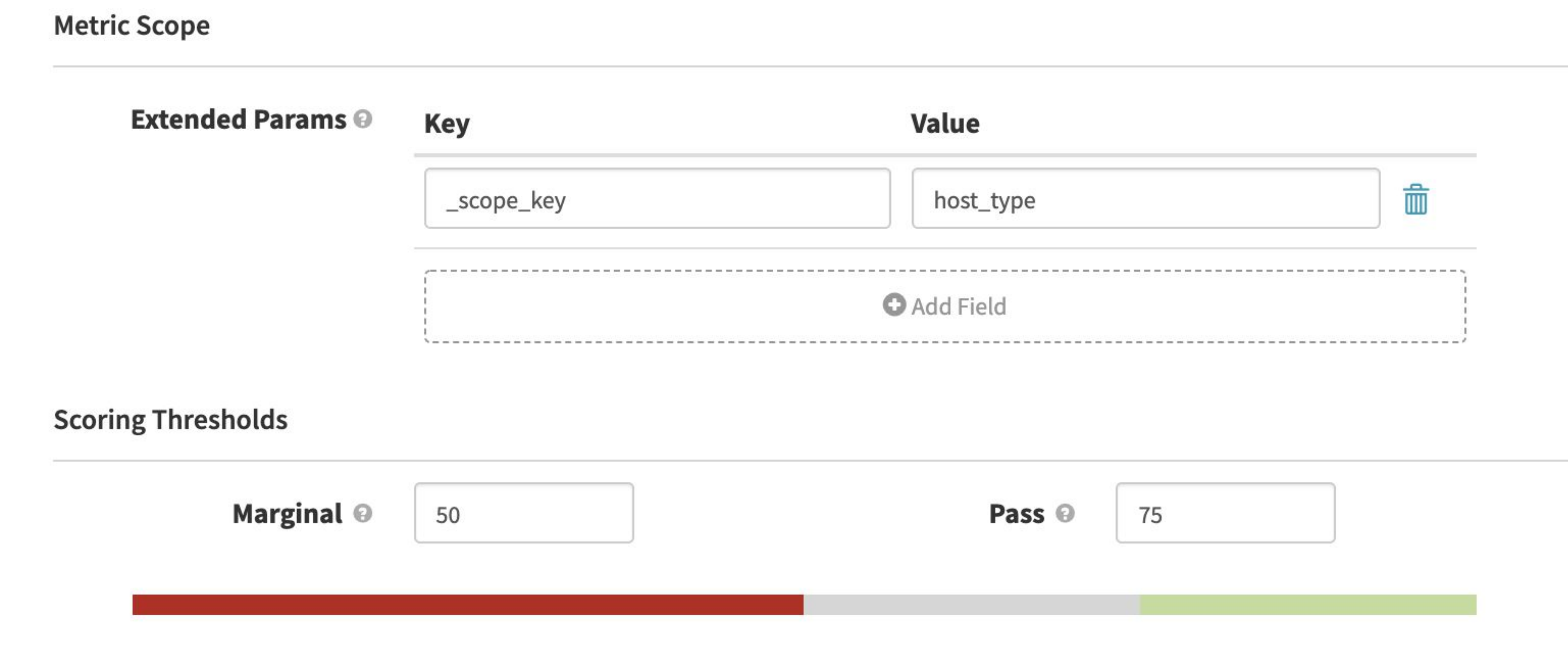
Delay: 0 minutes before starting analysis

Interval: minutes

Step: 1 seconds

Lookback Type: Growing

Configuring Scoring



Metric Scope

Extended Params

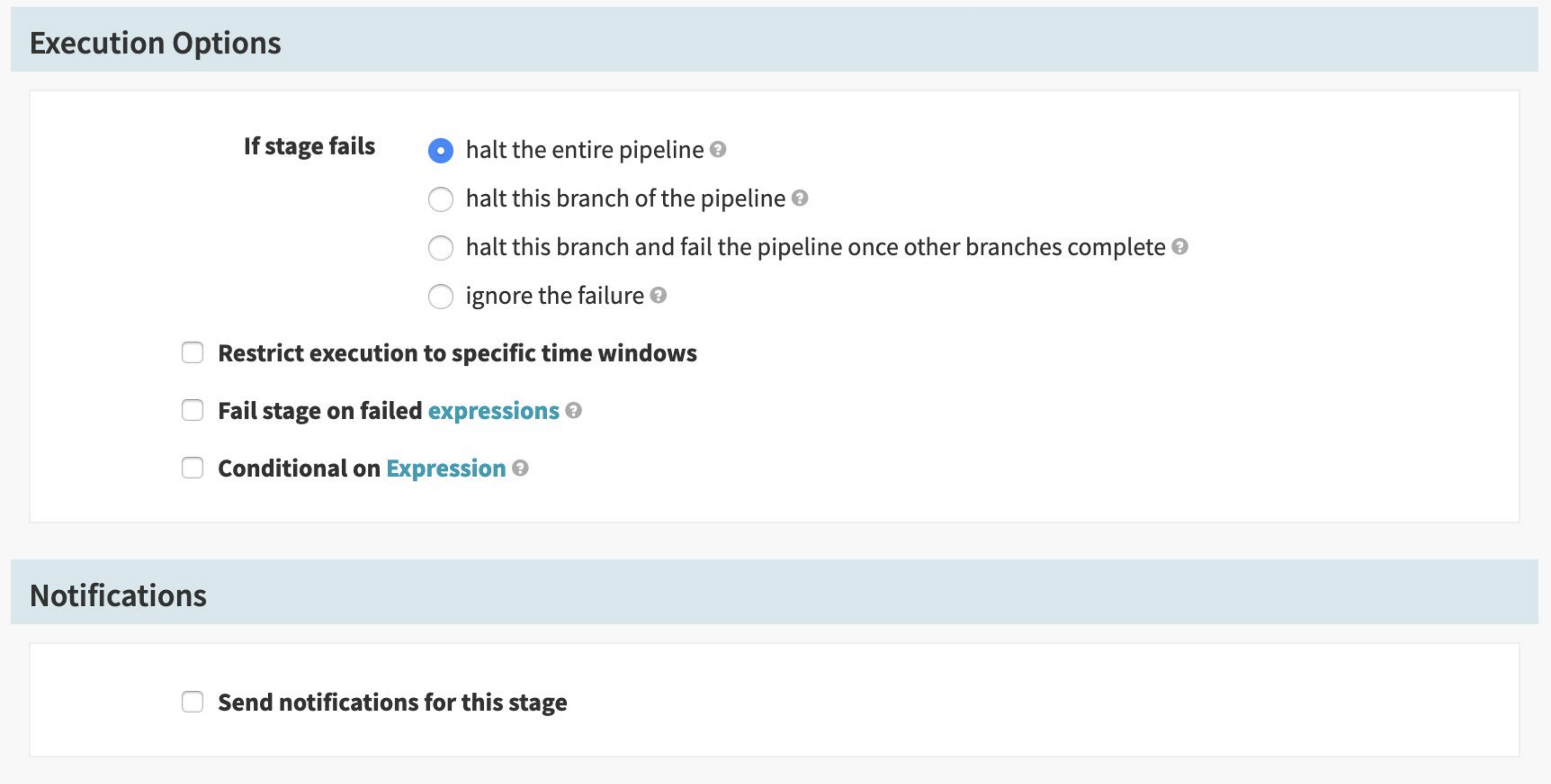
Key	Value
_scope_key	host_type

Scoring Thresholds

Marginal: 50

Pass: 75

Execution Options



Execution Options

If stage fails

- ● halt the entire pipeline
- ○ halt this branch of the pipeline
- ○ halt this branch and fail the pipeline once other branches complete
- ○ ignore the failure

☐ **Restrict execution to specific time windows**

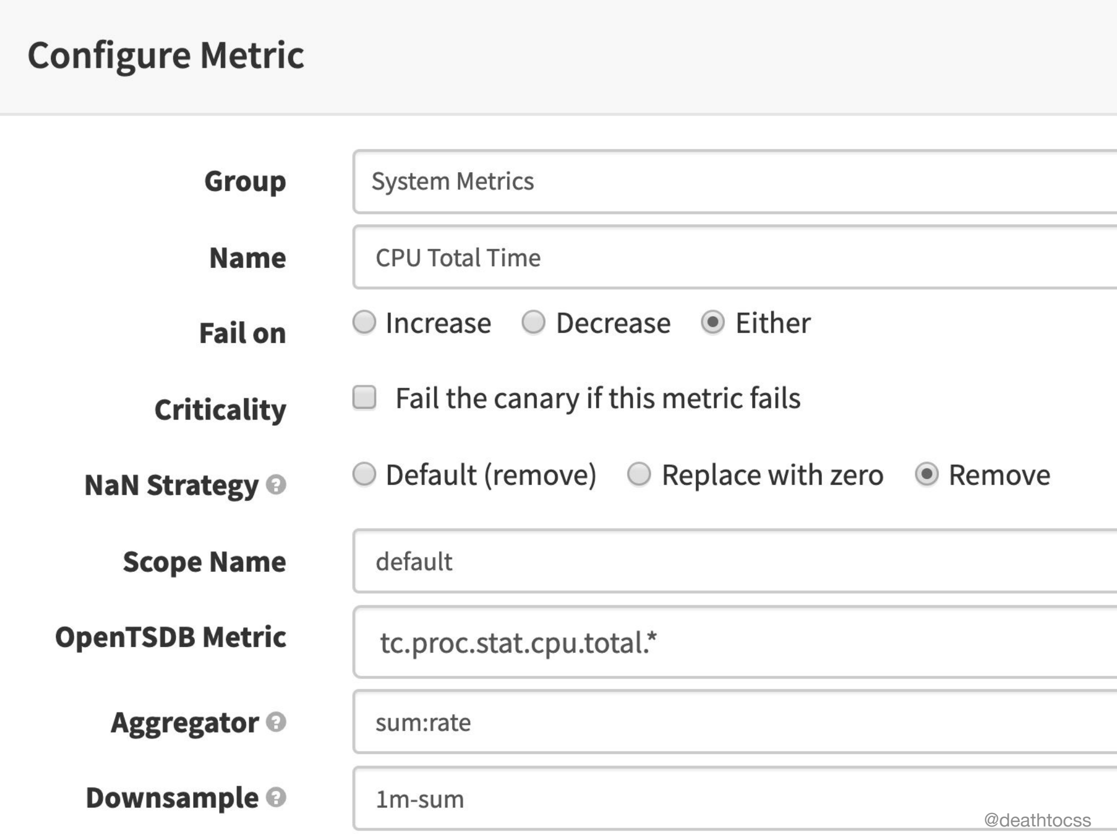
☐ **Fail stage on failed expressions**

☐ **Conditional on Expression**

Notifications

☐ **Send notifications for this stage**

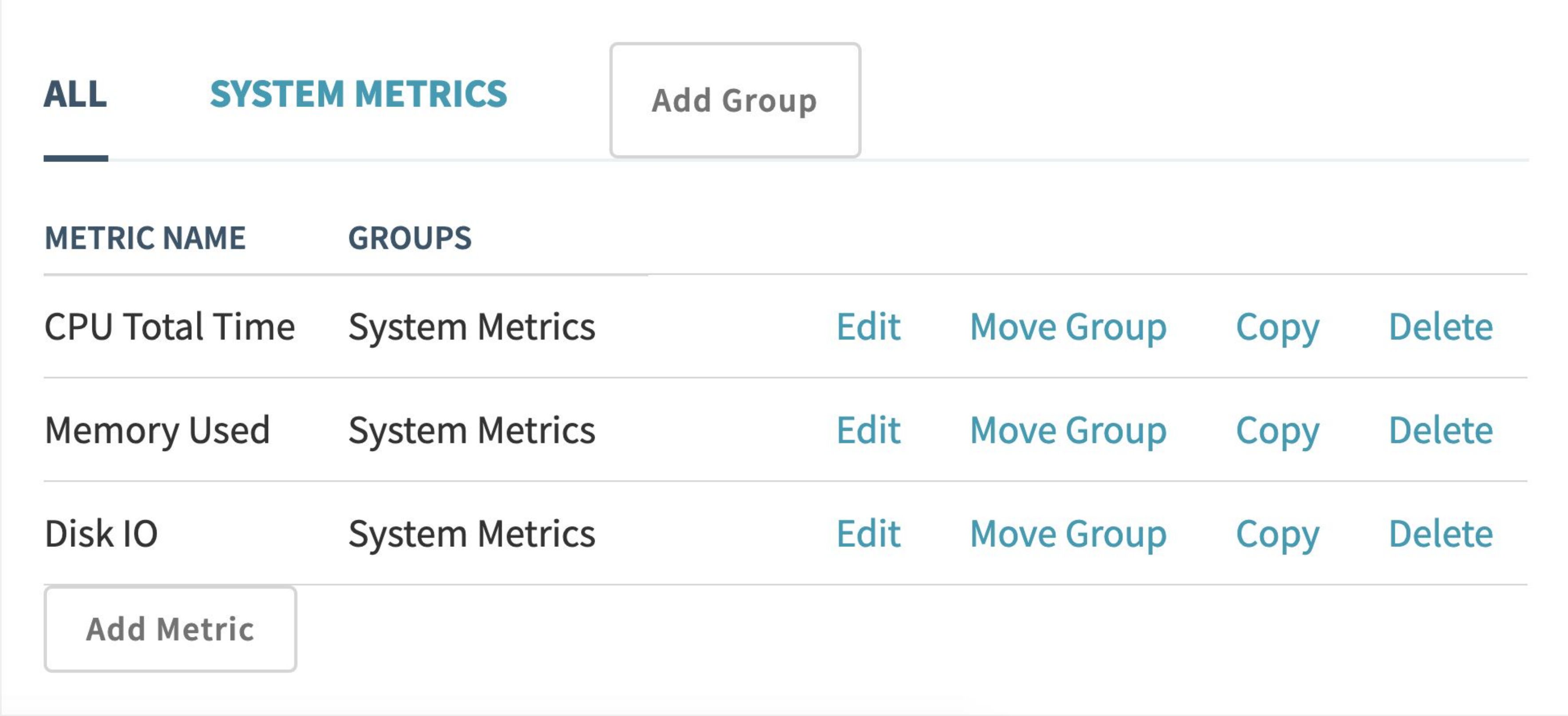
Adding a Metric



Configure Metric

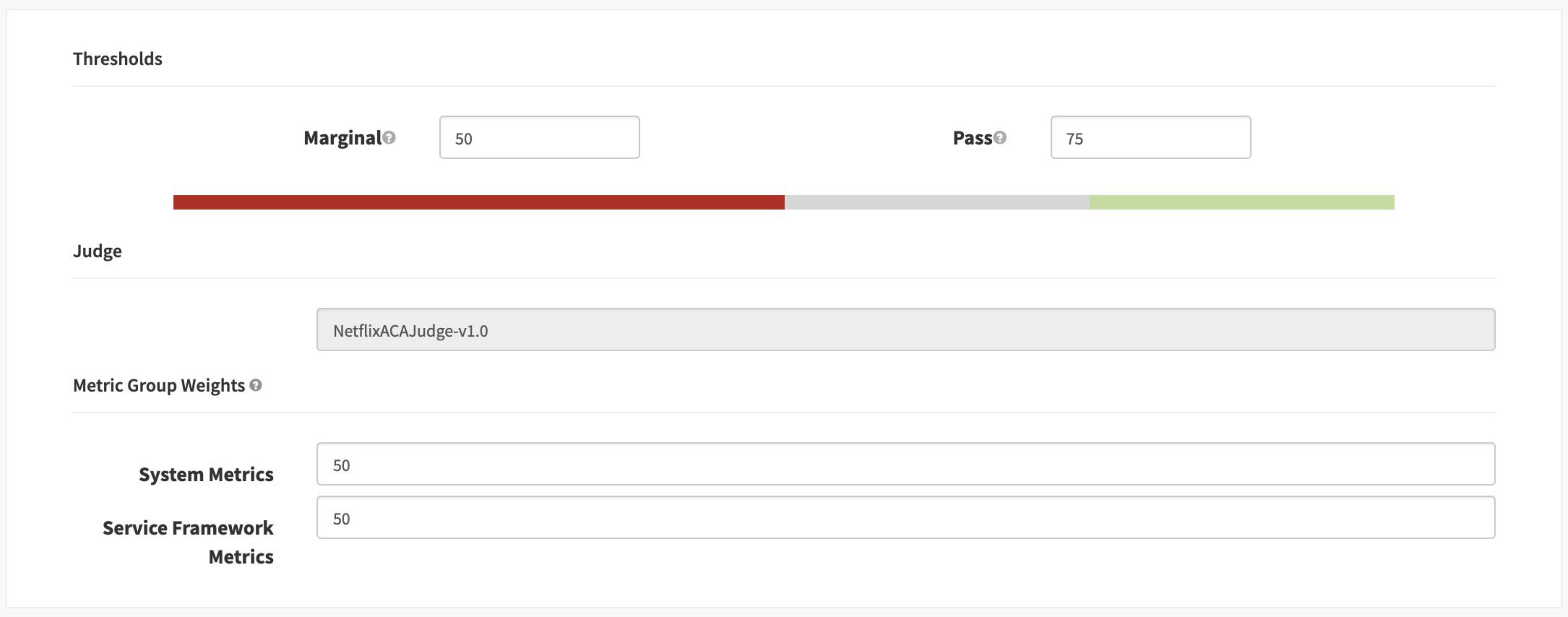
Group	System Metrics
Name	CPU Total Time
Fail on	○ Increase ○ Decrease ● Either
Criticality	□ Fail the canary if this metric fails
NaN Strategy ?	○ Default (remove) ○ Replace with zero ● Remove
Scope Name	default
OpenTSDB Metric	tc.proc.stat.cpu.total.*
Aggregator ?	sum:rate
Downsample ?	1m-sum

Metrics Groups



METRIC NAME	GROUPS				
CPU Total Time	System Metrics	Edit	Move Group	Copy	Delete
Memory Used	System Metrics	Edit	Move Group	Copy	Delete
Disk IO	System Metrics	Edit	Move Group	Copy	Delete

Metric Thresholds / Weights



Thresholds

Marginal: 50

Pass: 75

Judge

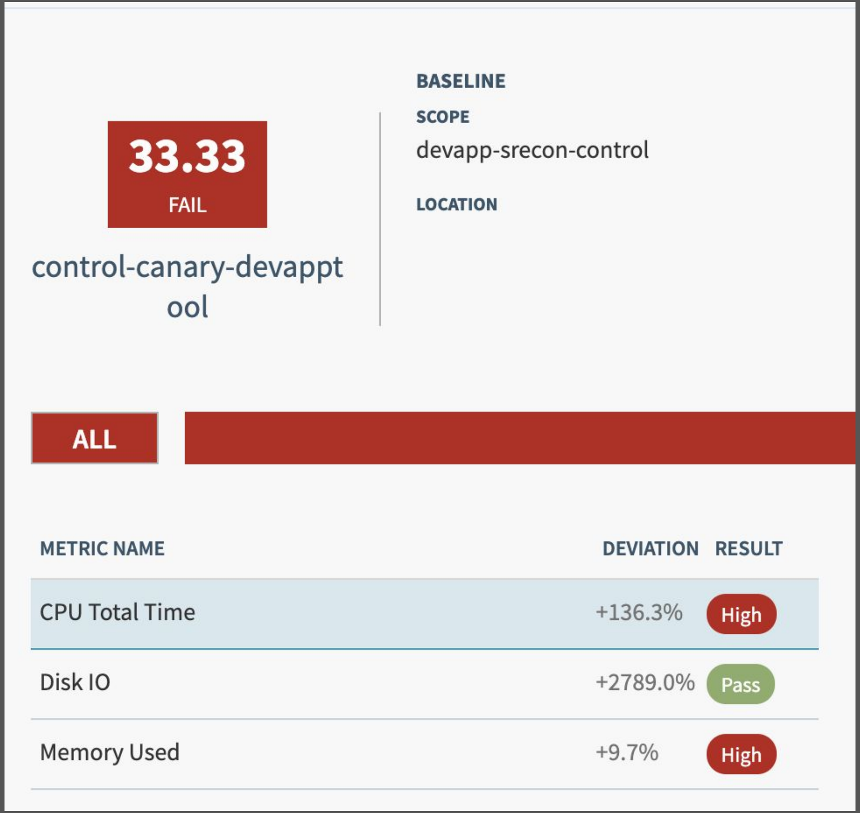
NetflixACAJudge-v1.0

Metric Group Weights

System Metrics: 50

Service Framework Metrics: 50

Canary Output



33.33

FAIL

control-canary-devapptool

BASELINE

SCOPE

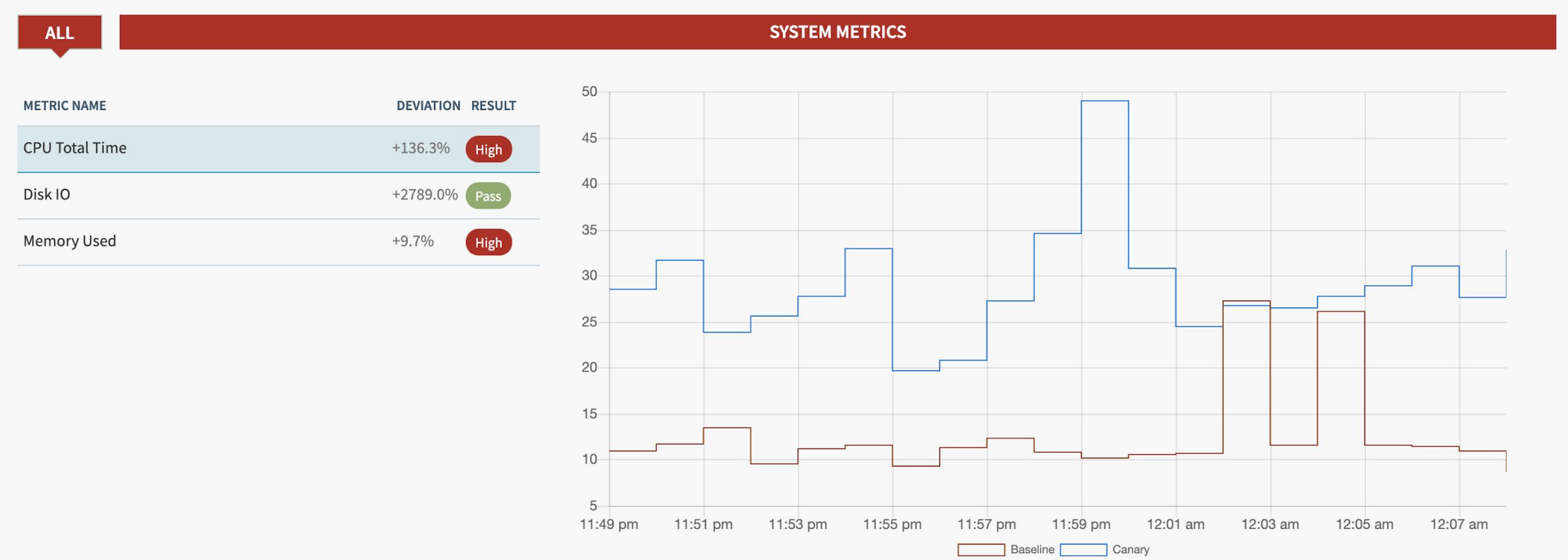
devapp-srecon-control

LOCATION

ALL

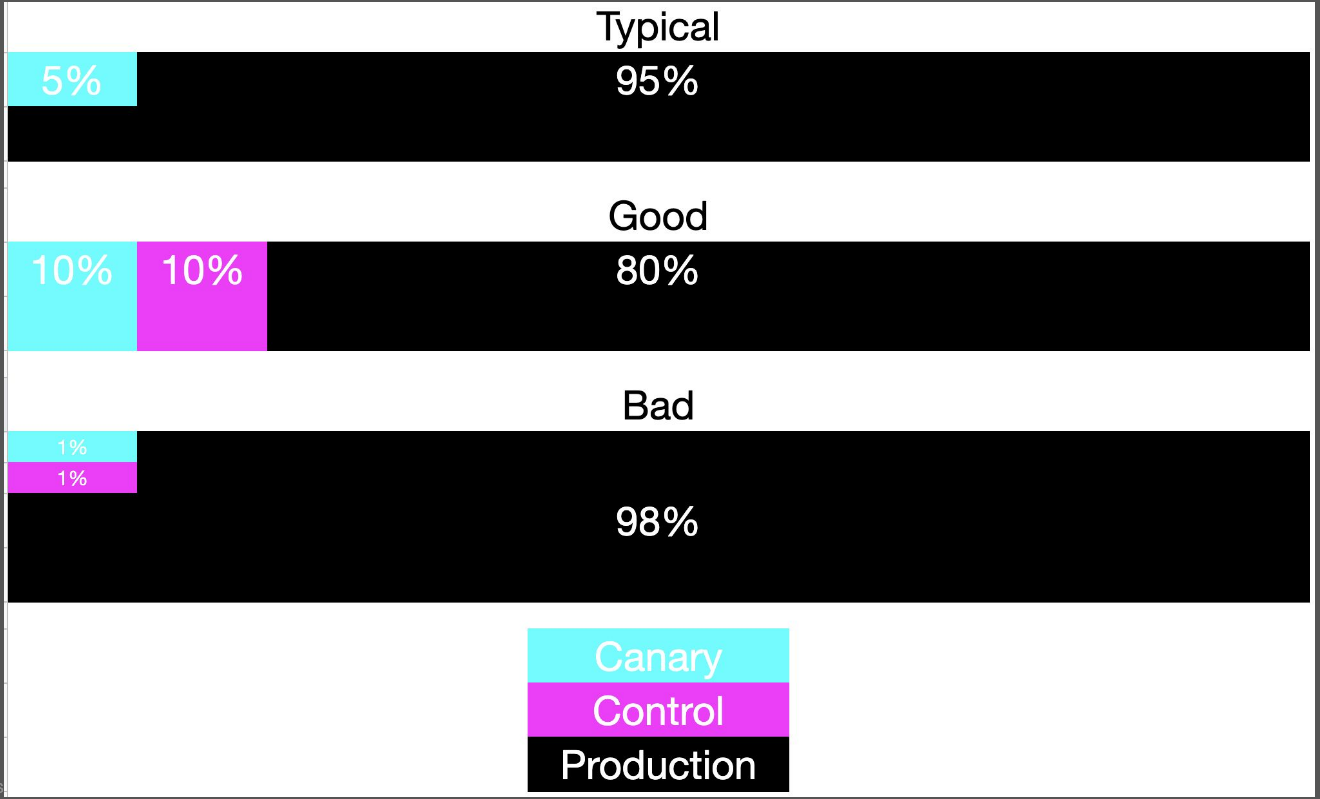
METRIC NAME	DEVIATION	RESULT
CPU Total Time	+136.3%	High
Disk IO	+2789.0%	Pass
Memory Used	+9.7%	High

Canary Output



METRIC NAME	DEVIATION	RESULT
CPU Total Time	+136.3%	High
Disk IO	+2789.0%	Pass
Memory Used	+9.7%	High

Sizing your clusters



Overall Effort

- Originally planned as a six-month effort for 2-3 FTE
- Actual effort was ~1 year due to various delays
- Cycle.py - ~700 lines of python
- Kayenta - had to write our own OpenTSDB plugin

Future Plans

- Integration of additional Machine Learning to Canary process
- Direct integration into our build system
- Additional tooling to improve user experience

Additional Resources

- **Kayenta:** https://cloud.google.com/blog/products/gcp/introducing-kayenta-an-open-automated-canary-analysis-tool-from-google-and-netflix
- **Spinnaker Canary Analysis doc:** https://www.spinnaker.io/setup/canary/
- **Waze presentation on Canary Analysis:** https://cloud.google.com/blog/products/devops-sre/canary-analysis-lessons-learned-and-best-practices-from-google-and-waze
- **Netflix presentation:** https://medium.com/netflix-techblog/automated-canary-analysis-at-netflix-with-kayenta-3260bc7acc69

We’re hiring! Come work with us!

Scan me

hiring-srecon@pinterest.com

Questions?