

Maintaining Reliability with Canary Testing

Pavlos Ratis (@dastergon)
Site Reliability Engineer, HolidayCheck

SRE Munich Meetup
September 25, 2018

Outline

- What is canary testing?
- Why it is useful?
- How to evaluate canaries?
- Pitfalls in the evaluation process
- Where do I begin?

Dickerson's Hierarchy of Service Reliability

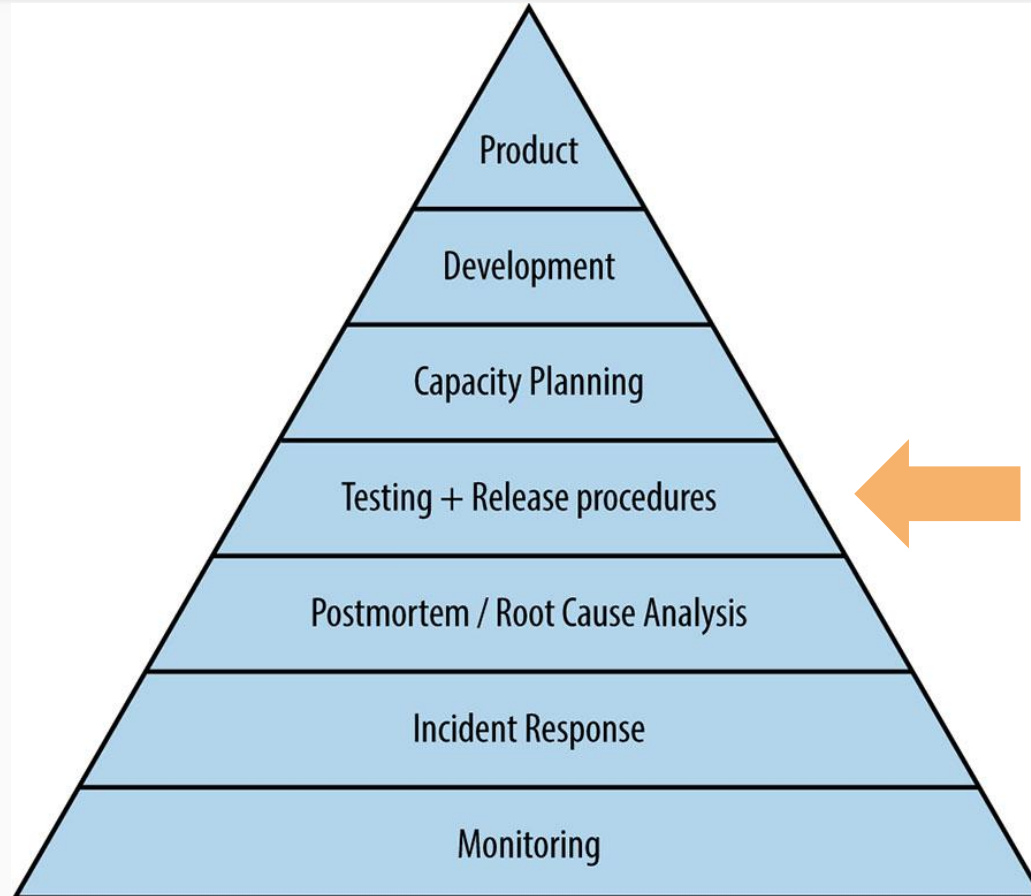
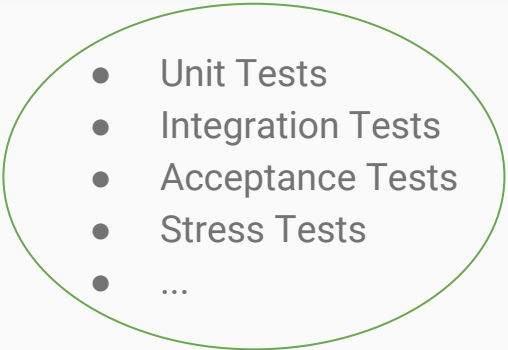
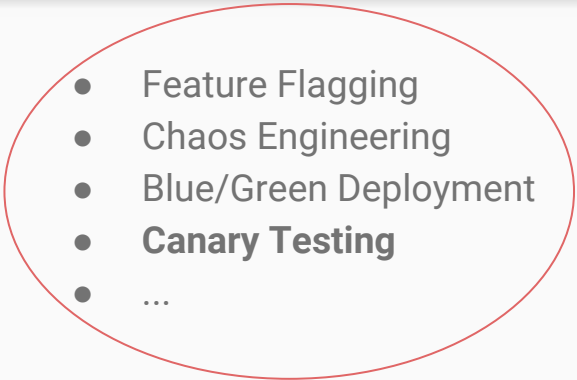


Image source:
Site Reliability Engineering:
How Google Runs Production Systems

Pre-Production Testing Production testing

- 
- Unit Tests
 - Integration Tests
 - Acceptance Tests
 - Stress Tests
 - ...

Comfort zone

- 
- Feature Flagging
 - Chaos Engineering
 - Blue/Green Deployment
 - **Canary Testing**
 - ...

Where magic happens

Canary in the coal mine

Historically used to detect gas in coal mines.

The idea first proposed by John Scott Haldane, in 1913 or later.^[1]

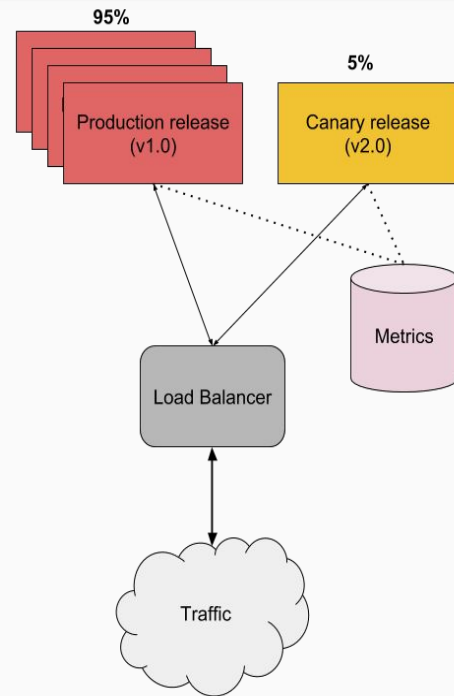


Image from: <http://coachellavalleyweekly.com/canary-in-a-coal-mine/>

[1]: "JS Haldane, JBS Haldane, L Hill, and A Siebe: A brief resume of their lives". *South Pacific Underwater Medicine Society Journal*. 29(3). ISSN 0813-1988. OCLC 16986801. Retrieved 2008-07-12.

Canary Testing

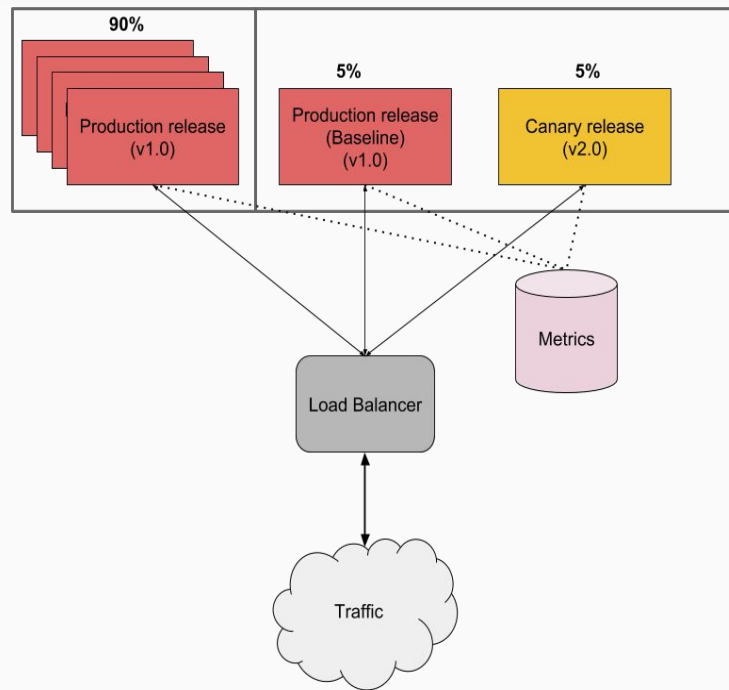
- New releases are deployed incrementally to a small subset of users.
- Stages
 1. Release
 - a. Gather data
 2. Evaluate canary
 - a. Compare metrics
 3. Verdict
 - a. Proceed to rollout on success
 - b. Proceed to rollback on bad behaviour



Traffic distribution

- **It depends!**
- Gradual releases (i.e., 5% increase every 3 hours)
- We need representative comparisons

Rule of thumb: Have a large set of production servers serving traffic normally, and have a small subset for production release baseline and canary.



Sampling

- Internal users (dogfooding)
- Random users
- Sophisticated user selection (i.e., country, activity)
- Combination of the above

Benefits

- Early warning system
- Reduces the risk
- Reliable software releases

Downsides

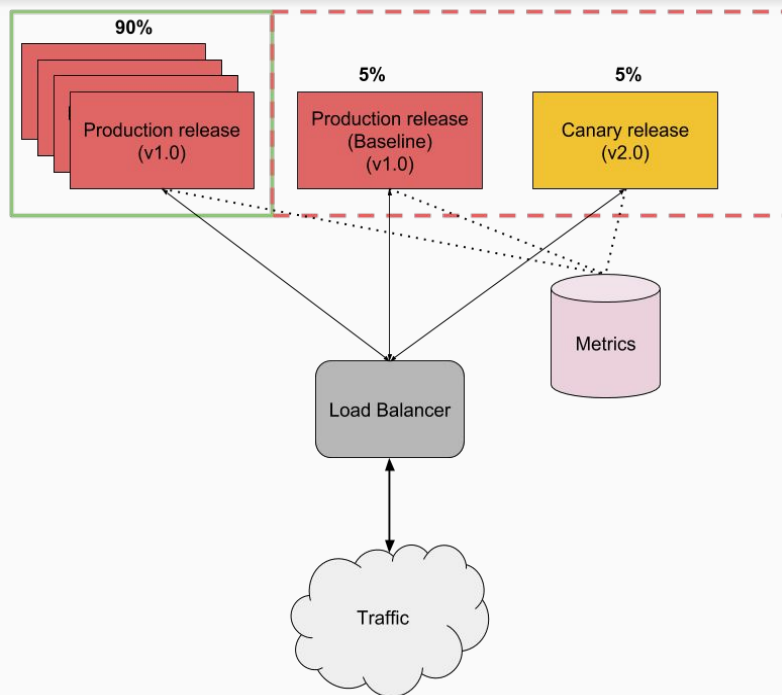
- Not the easiest task to put into practice initially
- Quite a few considerations before rolling out
- Requires time investment to implement properly

Pitfalls in Canary Releases

- Database changes
- Configuration changes
- Distributed monoliths
- Complexity in managing multiple versions

Canary Evaluation

- Identifies the reliability of the canary by comparing critical metrics for the specific release



Canary Evaluation Prerequisites

- Reproducible builds
- Instrumentation (for metrics)
- Have a rollback plan

Measure the impact

- Manually
 - Checking dashboards, graphs and logs
- Semi-automatic
 - Implementing supporting tools that are incorporated in rollout tools
- Automatic
 - Automated evaluation integrated as a service or in the CI pipeline
 - **Bonus points:** Automated rollback

Measure the impact (cont.)

Manual

- Operational toil for SREs
- Not reliable
- Bias
- Hard to declutter noise and outliers

Semi-Automatic

- It might still require some operational work
- Easier to implement
- Good for ad-hoc solutions

Automatic

- Requires time investment in the beginning
- Reduces the amount of operational work
- Increases productivity for developers and SREs
- Can be generalised for many services

What to measure

- Health checks during deployment (short circuit)
- Incoming network connections (short circuit)
- Success rate (HTTP 2xx)
- Error rate (HTTP 5xx)
- Latency distribution (90p, 95p, 99p)
- Load Average
- CPU utilization
- Memory leaks
- Quality

Considerations

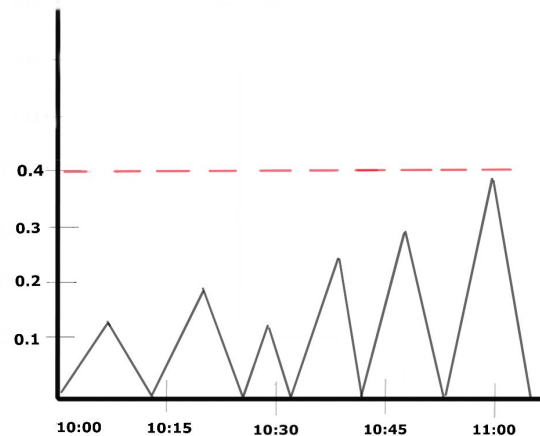
- Velocity for new releases
- Canary lifespan
- Amount of traffic
- New hardware
- Time (day vs night)
- Caches (cold vs hot)
- Different regions (eu-west vs us-west)
- Seasonality
- Diversity of metrics

Potential Issues

- Heterogeneous comparisons
- Overfitting
- False positives/negatives
- Trust

Overfitting

- Hand-tuning thresholds based on bounds observed in dashboard graphs is a bad idea
- Have adequate historical data
- Need to generalise
- Need to find better ways to classify

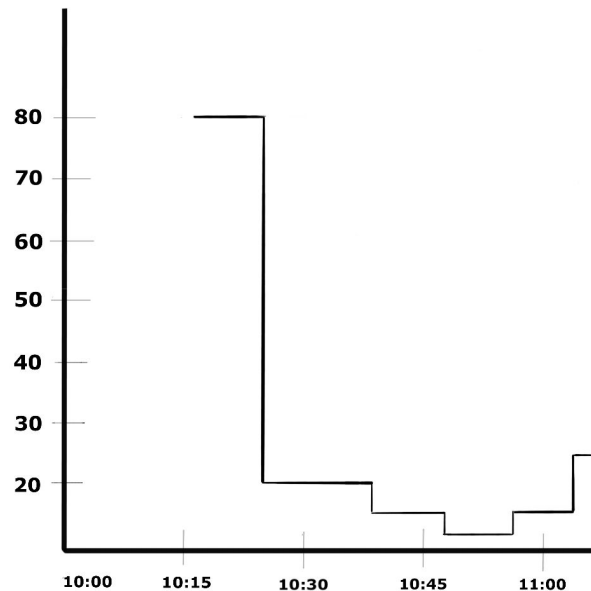


False Positives/Negatives

- Have adequate historical data from your baseline
 - Don't just look in the past 1 or 2 weeks.
- Think about your models.
 - What metrics are really meaningful to compare?
- Beware of outliers
- Reconsider importance of your comparisons
 - Error rate vs Systems metrics vs Latency

Caches

- Warmup caches if necessary.
- Wait for certain amount of time before start comparing.



Seasonality

- Christmas holidays, New Year's Eve, Black Friday or any other public event will affect your metrics
- High variance.
- Difficult problem. Requires thorough investigation.
- Start with moving averages.
- Investigate different anomaly detection algorithms.
 - Example: [Anomaly Detection algorithm by Twitter for big events and public holidays.](#)

Latency

- User perception of our product changes based on the timeliness of the response
- Factors that affect latency:
 - Network congestion
 - Memory overcommitment
 - Swapping
 - Garbage Collection pauses
 - Reindexing
 - Context Switching
 - ...
- Averages vs Percentiles
 - Averages are misleading, they hide outliers
 - We are interested in the “long tail”
 - Percentiles enable us to understand the distribution
- The bell curve is not representative

Latency (cont.)

- **Catch:** Canary has an average latency of **70ms**.
 - **Reality:** 99p: 99% of values are less than 800ms, 1% ≥ 800 ms latency.
- **Catch:** Canary latency **should not** exceed **10% above the average**.
 - **Reality:** When the amount of traffic is pretty low, or if we have heavy outliers, we will have false positives.
- **Catch:** Canary latency **should not** be **more than two standard deviations**.
 - **Reality:** In high variance (i.e., during peak season), it will give false positives.

Anomaly Detection in Time Series

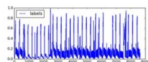
Time Series Anomaly Detection Detection of Anomalous Drops with Limited Features and Sparse Examples in Noisy Highly Periodic Data

Dimitrios T. Shipmon, Isaac M. Gurevitch, Paul M. Pisch, Steve Edwards
Google, Inc.
Cambridge, MA, USA
{dts@cs.cmu.edu, {isagurev,psp}@google.com,
stev@cs.cmu.edu}

Abstract—Google uses continuous streams of data from industry partners in order to deliver results to users. Unexpected drops in traffic can be an indication of an underlying issue and may be an early warning that remedial action may be necessary. Detecting such drops in non-trivial because streams are variable and noisy, with enough regular spikes that many different aspects in traffic data. We investigated the question of whether or not we can predict anomalies in these data streams. Our goal is to allow machine learning and statistical approaches to classify anomalous drops in periodic, but noisy, traffic patterns. Since we do not have a large body of labeled examples to directly apply supervised learning for anomaly classification, we approached the problem in two parts. First we used TensorFlow to train our various models including RNNs, RNNs, and LSTM, to perform regression and predict the expected value in the time series. Secondly we created anomaly detection rules that compared the actual values to predicted values. Since the problem requires finding anomalous patterns, rather than just short delays or momentary increases in the data, our two models were trained on continuous streams of activity rather than just single points. We used multiple combinations of our models and rules and found that using the intersection of our two anomaly detection methods proved to be an effective method of detecting anomalies on almost all of our models. In the process we also found that not all data sets follow our experimental assumptions, so our data stream had no periodicity, and therefore no time-based model could predict it.

Keywords—Anomaly Detection; Anomaly Detection; Anomaly Detection; Drop; Neural Networks; Regression; Neural Networks; Long short-term memory

A. About the Data
We looked at 14 different sets of data that were used at 5 minute intervals. This means that each hour had 12 data points, and each day had 288. Our goal is to detect anomalies such as a decrease in daily traffic, but not unexpected decreases in the throughput volume, while not detecting anomalies on regular low or high values. The only two attributes of this dataset were a unit timestamp and the two or second value, with a sample every 5 minutes. Although there are no accurate markings of anomalies within this data, there does not mean that they do not exist, only that we had no information of the at the beginning of this project.



B. About the Problem

When there are clear labels for anomalies data a binary classifier can be built to predict anomalies and non-anomalies points. For these problems there exist a plethora of techniques to choose from, including clustering analysis, isolation forests, and classifiers built using artificial neural networks. These have all shown promise in the field of anomaly detection [1]. The first two of these techniques not only require labels for training, but also are more effective when there are many features. For these reasons they are not as useful on time series data, especially when there are not other features that we have access to.

Neural networks can effectively predict periodic time series data, so can simple techniques such as Fourier series. However, because that treats an anomaly as very hard on the data, such problems potentially require its own model. Some problems have many features to work with or need to occur at a data point, while others, like ours, have few features and look for continuous, rather than point anomalies.

We trained a separate model for each data stream since we found no significant correlation between each of the data streams. Our models trained predictions that we compared to the actual data stream values using our anomaly detection rules to determine if the current point is anomalous. Although all of our models and metrics are done in an offline environment, they are all approaches that are adaptable to online training, prediction, and anomaly detection. The adaptability of our models is a result of our machine learning models and anomaly detection rules only requiring either the past or current point. Additionally, to make sure our models

Automatic Anomaly Detection in the Cloud Via Statistical Learning

Jordan Hochenbaum, Owen S. Vallis, Arun Kejariwal
Twitter Inc.

ABSTRACT

Performance and high availability have become increasingly important drivers, amongst other drivers, for user retention in the context of web services such as social networks, and mail search. Diagnostic and endogenous factors often give rise to anomalies, making it very challenging to maintain high availability, while also delivering high performance. Given that server-oriented architectures (SOA) typically have a large number of services, each such service forms a large set of metrics, automatic detection of anomalies is a non-trivial task. Although there exists a large body of prior research in anomaly detection, existing techniques are not as effective in the context of social network data, owing to the inherent anomaly and trend components in the time series data. To this end, we developed two novel statistical techniques for automatically detecting anomalies in cloud infrastructure data. Specifically, the techniques employ statistical learning to detect anomalies in both applications, and system metrics. Statistical decomposition is employed to filter the trend and seasonal components of the time series, followed by the use of robust statistical measures – median and median absolute deviation (MAD) – to accurately detect anomalies, even in the presence of seasonal cycles. We demonstrate the efficacy of the proposed techniques from three different perspectives, viz., applying planning, user behavior, and operational learning. In particular, we used prediction data for evaluation, and we report Precision, Recall, and F-measures in each case.

1. INTRODUCTION

The data is characterized by the increasing volume (in the order of analysis), and the velocity of data generation [1], [2]. It is reported that the median size of Big Data has doubled from the current median size of \$1.1 billion [3] to \$3.7 billion by 2017. In recent years [1], IBM Corporation stated, “A major factor behind the expansion of the digital universe is the growth of machine-generated data, increasing from 1.6% of the digital universe in 2001 to over 40% in 2010” [4]. In the context of social networks, members of Twitter Data is key to building an engaging social network; in a similar fashion, analysis of Machine Data is key to building an efficient and performant underlying cloud computing platform.

In the Twitter Data use case, the ability to detect anomalies in both presence [5], as well as in long term repetition [6]. To this end, several enterprise-wide monitoring

ing initiatives [7], have been undertaken. Likewise, there has been an increasing emphasis on developing techniques for detection, and even more analysis, of performance issues in the cloud [8], [9], [10], [11], [12], [13].

A lot of research has been done in the context of anomaly detection in various domains such as, but not limited to, statistics, signal processing, finance, econometrics, manufacturing, and networking [14], [15], [16]. In a recent survey paper [Chandak et al.], highlighted that anomalies are considered to occur [17] and touched the following:

A data measure might be a contextual anomaly in a given context, but an statistical data measure (in terms of behavioral attributes) could be considered normal in a different context. This property is key to identifying contextual and behavioral anomalies for a contextual anomaly detection and response.

Detection of anomalies in the presence of seasonality, and an underlying trend – which are both characteristics of the time series data of social networks – is non-trivial. Figure 1 illustrates the presence of both positive and negative anomalies – corresponding to the circled data points – in time series data obtained from prediction. From the figure we note that the time series has a very complex seasonality, and that there are multiple modes within a seasonal period. Existing techniques for anomaly detection (inspired in depth in Section 3) are not amenable to such data, as well as the inherent complexity. In this end, we developed novel techniques for automated anomaly detection in the cloud for statistical learning. In particular, the main contributions of the paper are as follows:

1) First, we propose novel statistical learning based techniques to detect anomalies in the cloud. The proposed techniques can be used to automatically detect anomalies in time series data of both application metrics such as Twitter Post (TP) and system metrics such as CPU utilization etc. Specifically, we propose the following:

• Seasonal Hybrid ESD (S-HESD). This technique employs time series decomposition to determine the seasonal component of a given time series. S-HESD then applies ESD [18], [21], as well as the resulting time series to detect the anomalies.

• Seasonal Hybrid ESD (S-HESD). In the case of some time series (observed from production) we observed a relatively high percentage of false positives. To address such cases, coupled with the fact that mean and standard deviation (used by ESD) are highly sensitive to a large number of anomalies [21], [22], we extended S-HESD to use the robust median absolute median [23] and median absolute deviation (MAD) to detect anomalies [24]. Com-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are made for personal or classroom use, and are not for commercial purposes. Copyright 2017 ACM XXXXX-XX-XXXXX, \$16.00.

¹Dimitrios Shipmon and Isaac Gurevitch were summer interns at Google when they did this work.

Shipmon, D.T., Gurevitch, J.M., Piselli, P.M. and Edwards, S.T., 2017. Time Series Anomaly Detection; Detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. arXiv preprint arXiv:1708.03665.

Hochenbaum, J., Vallis, O.S. and Kejariwal, A., 2017. Automatic anomaly detection in the cloud via statistical learning. arXiv preprint arXiv:1704.07706.

Verdict

1. How much deviation is tolerable?
2. Evaluation:
 - a. Pass or Fail
 - b. Cumulative Score with thresholds
 - c. Both a. and b.



Build Trust

- Start small.
- Accept the fact that you will have false positives.
- Don't overdo it with the comparisons. (less is more)
- Have a pair of eyes in verification initially.
- Experiment with different models.
- Iterate often to improve the accuracy.
- Don't neglect your SLOs

Getting Started

- Metrics collection:
 - Stackdriver
 - Prometheus and Influxdb
- Evaluation:
 - Spinnaker with Kayenta
 - Kapacitor (Influxdb)
 - Kubervisor

Summary

- Canary testing
 - is important to maintain the reliability levels
 - can be applied to any size of infrastructure
- Never neglect the evaluation stage. Many factors to consider!
- Keep a minimal amount of metrics comparisons per evaluation
 - Not all metrics are important
- Start small, then, iterate for better accuracy

Further Reading

- [Testing Microservices, the sane way](#)
- [How release canaries can save your bacon - CRE life lessons](#)
- [Canary Analysis Service](#)
- [Automated Canary Analysis at Netflix with Kayenta](#)
- [Canarying Well: Lessons Learned from Canarying Large Populations](#)
- [Introducing practical and robust anomaly detection in a time series](#)
- ["How NOT to Measure Latency" by Gil Tene](#)

Thank you!

@dastergon

<https://dastergon.gr>

<https://speakerdeck.com/dastergon>

<https://github.com/dastergon/awesome-sre>

Real World SRE by Nat
Welch (Packt Publishing)

