

# Hadoop Inspector

Ken Farmer  
William Farmer

August 14, 2015

## **Abstract**

This project attempts to act as a health report for your Hadoop cluster. At its core, this tool acts as a Hadoop Testing Framework. Tests are either in the form of rules or checks, and either flag the table for review if a rule fails, or throws a warning if a check fails. Tests can be scheduled, and the incremental output can be viewed through the local frontend. This allows for the user to view the health and status of their database over time.

## 1 Background

Data quality problems have plagued analytical systems for twenty years: continually appearing in the top four reasons for project failure.

In this space data quality problems loom large - a small defect that could be safely ignored or forgotten in the transactional world hamper queries and cause users to question our credibility for months.

The advent and innovation in Big Data and Data Science has not diminished this challenge. On Hadoop specifically:

Our data generally lacks any enforced constraints to ensure data validity. We are adding data faster than ever, with less time to research upstream and ETL pipeline issues. We are building vast systems, sometimes with hundreds of thousands of tasks being defined. We often have democratized access to our clusters - with dozens of different people adding data.

Additionally, in these large clusters most teams struggle to comply with policies and other requirements, whether regulatory, corporate or defined by their own teams. These might define general data retention requirements, or specific requirements for individual tables. They might define table naming conventions, security requirements, or stats aging & collection requirements.

## 2 Objective

Hadoop-Inspector is being built because we believe that the complexity of a large, constantly loaded cluster defies an unmanaged approach or QA testing in the development process. It requires something more like an automobile assembly line: continuous quality control (QC) that can take into account undocumented changes from upstream sources, accidental changes to production, changes that bypass QA, etc. And it shouldn't be limited to traditional quality tests, but should be able to test for compliance with policies as well.

## 3 Checks

Every test is simply a script that lives in a type-specific folder. Rules and Warnings exist separately.

Each script returns a JSON-encoded object to `stdout` that meets the following specification.

---

```
1 {
2   "name": "Name of test",
3   "violations": 9000,
4   "Output": "Test Specific output. Either JSON or String"
5 }
```

---

### 3.1 Rules

Rules are strict rules about the cluster. They should never be ignored, or disobeyed, and any violation of a set rule results in a violation. Ideally, a “healthy” cluster should have no rule violations.

An example rule would be that a specific column only contains integers.

### 3.2 Warnings

Checks are suggestions about specific tables, or databases. These are a lot more fluid, and a warning from a check does not necessarily indicate an issue with the database, but rather that something new may have happened.

For example, a check that examines the average number of a column may throw a warning if an entry is too far from the mean. This isn't an indication that the data is bad, just that it *may* be bad.

## 4 Results

All results can be viewed through the front end. This provides a way to analyze your cluster's health, as well as its current status.

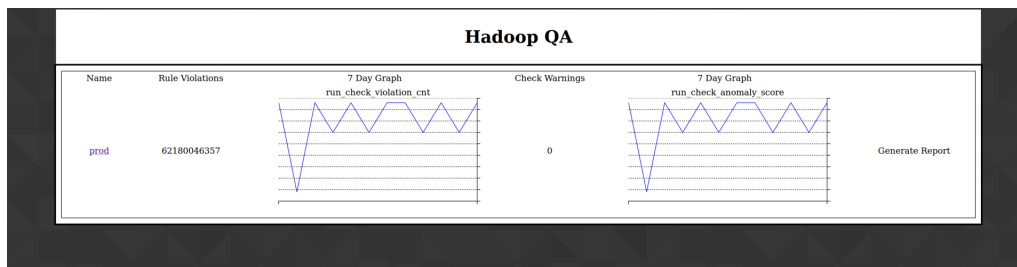


Figure 1: Index

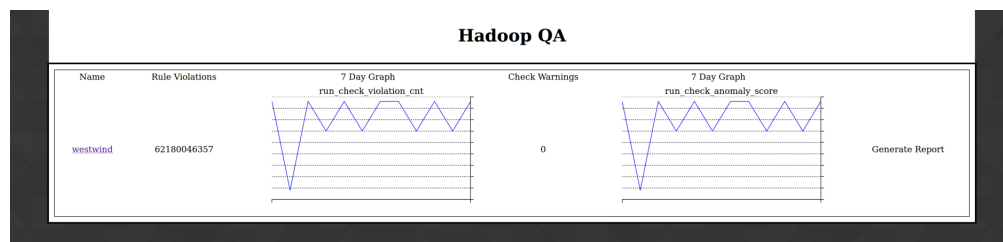


Figure 2: Environment View

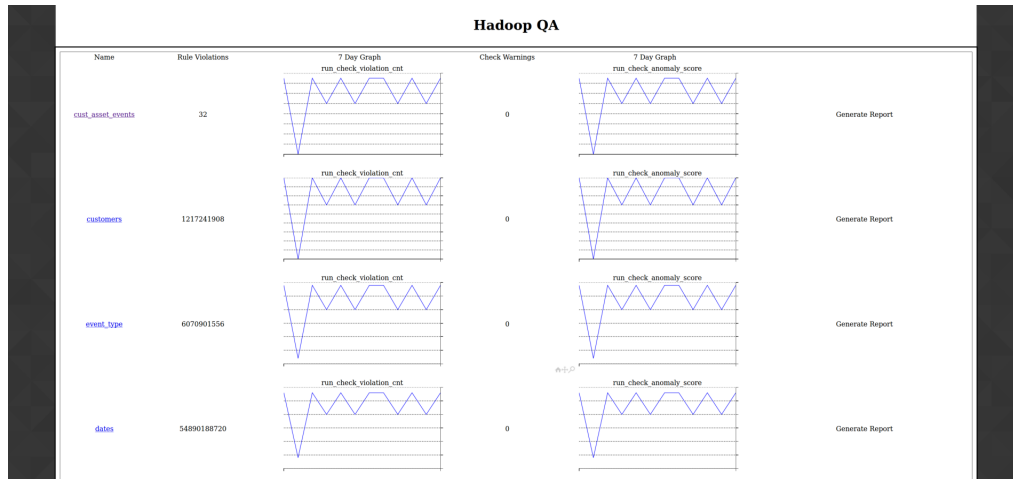


Figure 3: Database View

## 5 Current Status

Our initial focus has been on building a demo to help us validate ideas, and build some of our UIs. This includes:

- **hadoopinspector-demogen** - Can generate 50,000+ check results against a hypothetical user hadoop environment
- **server** - Runs a website that allows the user to analyze these demo results
- **report** - Produces a pdf check result summary report