influxdata®

AN INFLUXDATA CASE STUDY

How Vera C. Rubin Observatory Uses Time-Stamped Data to Understand the Universe Better

Dr. Angelo Fausti

Software Engineer, Vera C. Rubin Observatory

NOVEMBER 2019 (REVISION 1)

Company in brief

Vera C. Rubin Observatory's mission is to constantly survey the Southern Sky (skies above the southern hemisphere) for 10 years to collect 500 petabytes of image data to enable astronomers to make new discoveries and answer many questions about the universe. The survey is known as the Legacy Survey of Space and Time (LSST). They are currently building an 8.4 meter optical telescope in Chile. The entire system includes a modern observatory, a telescope, a camera, and an automated data processing system. It is designed to take about 1,000 images of the sky every night. It is essentially "the world's largest digital camera" in which one image covers 40 times the area of the Moon.

They plan on measuring the properties of 37 billion stars and galaxies. The survey will show changes in the sky over time and will in essence create the first motion picture of the universe. They desire to better understand: dark matter and dark energy, the solar system, the changing sky, and the Milky Way's structure and formation.

Rubin Observatory will produce alerts for stars, planets, solar system objects, asteroids, galaxies and other objects that change position or brightness every 60 seconds. The entire Southern Sky will be imaged every three nights. Rubin Observatory will provide a thousand-fold increase in capability over current facilities, dramatically advancing our knowledge of the universe.

All alerts will amount to 10 million alerts, 1,000 pairs of exposures and 15 terabytes of data per night. There will be worldwide alerts within 60 seconds of detected changes in the sky. They plan to create a huge dataset which will provide the properties of billions of stars and galaxies, and astronomers will be able to use it to further society's knowledge of the universe.

The idea for the project started in 1998. In 2008, they started the construction of the telescope's mirror with private funds, and in 2014, official construction started with funds from the National Science Foundation (NSF) and the Department of Energy (DOE). This international endeavor includes 22 universities, observatories and other institutions around the world. Rubin Observatory is primarily funded by the US — by the National Science Foundation (NSF) and the Department of Energy (DOE). NSF has their National Optical-Infrared Astronomy Research Laboratory (NSF's OIR Lab) whose mission is to help groundbreaking discoveries in astrophysics through state-of-the-art observatories. Rubin Observatory is one of their five programs. The Association of Universities for Research in Astronomy (AURA) is a consortium of 47 US-based institutions and 3 international affiliates that operate observatories for the NSF and for NASA. AURA operates Rubin Observatory for the NSF under a cooperative agreement.

NSF provided $473 million for the construction of the observatory, telescope and software. Through the Office of High Energy Physics in the Office of Science, the DOE provided $168 million to the construction of the camera. Rubin Observatory also received $30 million in 2008 from Charles Simonyi and Bill Gates for the construction of the mirror.

In 2019, they started testing and assembling the telescope in Chile. In the current schedule, Rubin Observatory will start operations by October 2022 and begin the ten-year survey.

Datasets will be available through LSST's Education and Public Outreach (EPO) online portal. They will be offering tools and activities for educators, scientists, science centers and the general public to engage, explore and discover. In 2005, the US Congress ruled NASA has to analyze and track all hazardous near-Earth objects (NEO's) that are equal or greater than 140 meters in diameter. LSST will be able to contribute to this mandate and help NASA work towards completion of this requirement.

Case overview

Rubin Observatory's operations are complex and run by an international team. They already had a database in place, but realized they needed a tool that better addressed their needs. As they will be accumulating 500 petabytes of time-stamped data that includes application/infrastructure metrics, sensor data, and time-stamped astronomical data, a combination of three different and distinct types of data, it made sense to look at purpose-built time series databases. By using InfluxData's TICK Stack, they better understand their telescope, camera, observatory and data. Through trial and error, they have configured InfluxDB, the purpose-built time series database, to best suit their needs. The worldwide team understands their data through improved analytics, alerts and dashboards.

The business challenge

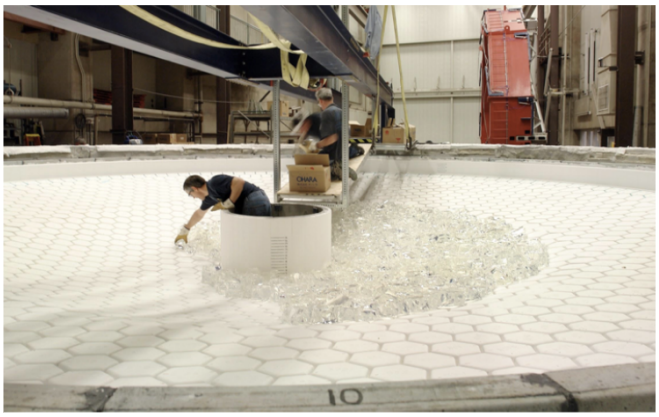
LSST wants to create a modern observatory including a telescope, a camera and software to control the telescope. Additionally, they need to process all data collected. The creation of the telescope and the camera presented interesting challenges as it is the first of its scale. The construction of the Rubin Observatory, the integration of the different parts of the telescope and the verification and validation process are really challenging.

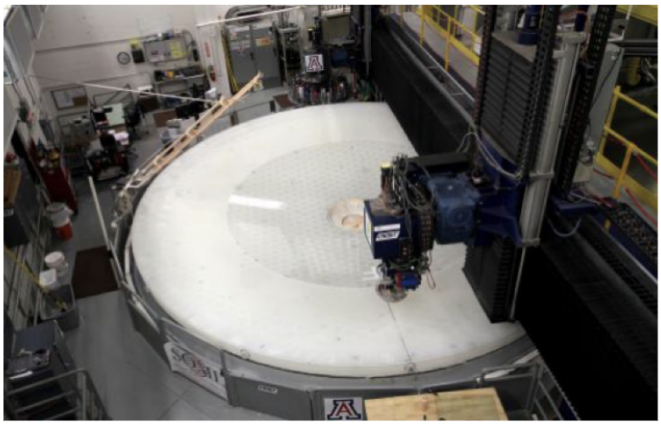
The telescope and the mirror

The jumping-off point for the telescope's design was the Dark Matter Telescope suggested by Roger Angel and his team at the University of Arizona in 2000. The LSST has a unique three-mirror design which provides a very wide view. It is so wide that it will be able to survey the entire sky in three nights. The LSST is located on the Cerro Pachon ridge in northern-central Chile, the telescope is about 60 miles inland, and their base facility is in the town of La Serena.

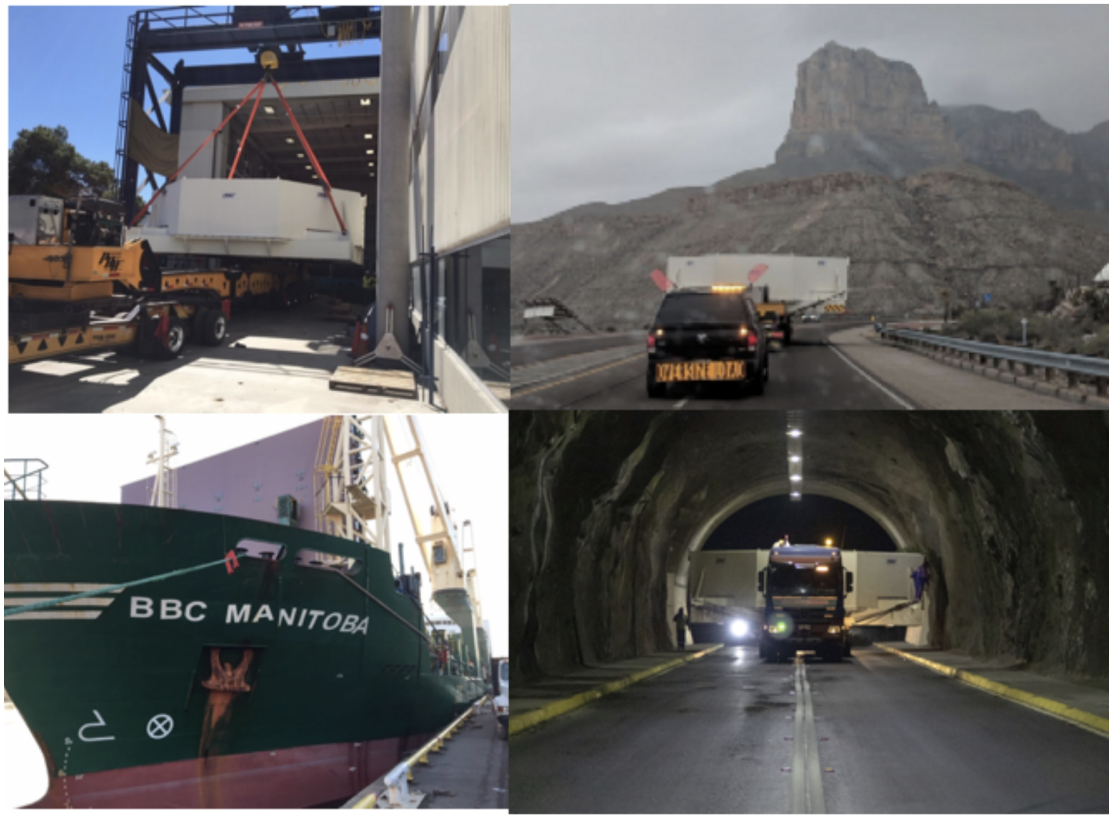
There are two types of optical telescopes: refracting and reflective. Refracting telescopes use convex lenses to gather and focus light which is used to form an image that is magnified for the viewer. Reflecting telescopes primarily use concave mirrors to gather light. There's often a secondary mirror used in reflecting telescopes to bring the light to a detector — human eyes are an example of a detector. For larger telescopes, there's often an electronic device used as the detector.

While LSST's mirror is not the largest, the primary mirror is 8.4 meters wide, but what is unique about its design is that the two optical surfaces that make the primary and tertiary mirrors are built in a single piece of glass. The reason for that is to make a very compact and fast telescope to take as many pictures of the sky as possible. The mirror's construction began in 2015. It was built and polished in LSST's Mirror Lab in Tucson, Arizona. It took about five years for the team to polish the final mirror's surfaces. The mirror was stored in a warehouse at the Tucson Airport for a few years. It was back at the lab at the beginning of 2019 for final optical testing.





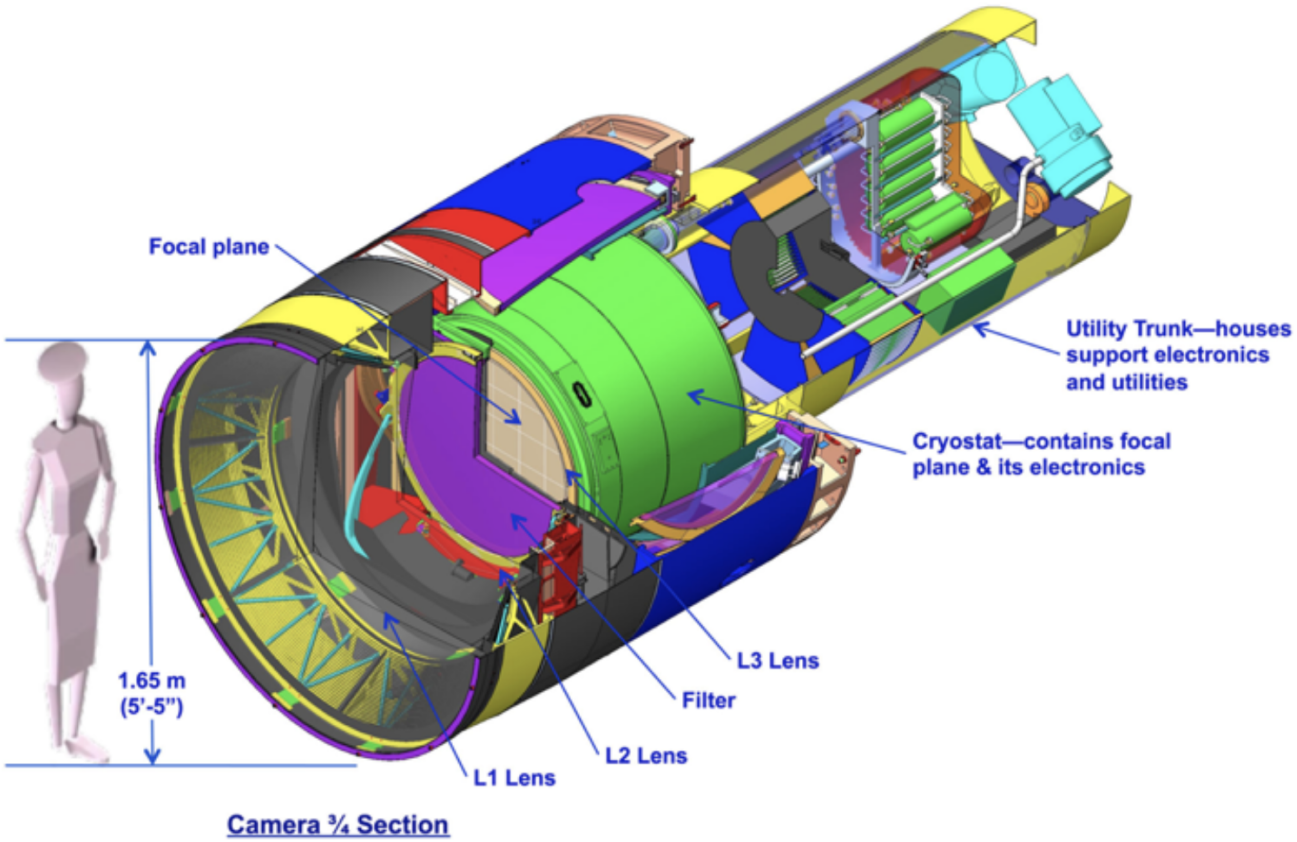
The mirror weighs around 17 tons and requires 54 vacuum pads to lift and move it. By early 2019, it was on its way to Chile. Getting to the observatory's site presented interesting challenges. It crossed the Panama Canal on a general cargo ship. On its way to Cerro Pachon, where the observatory is located, there was a tunnel where the mirror barely fit through, and to get to the top of the mountain, it was pulled by two trucks.



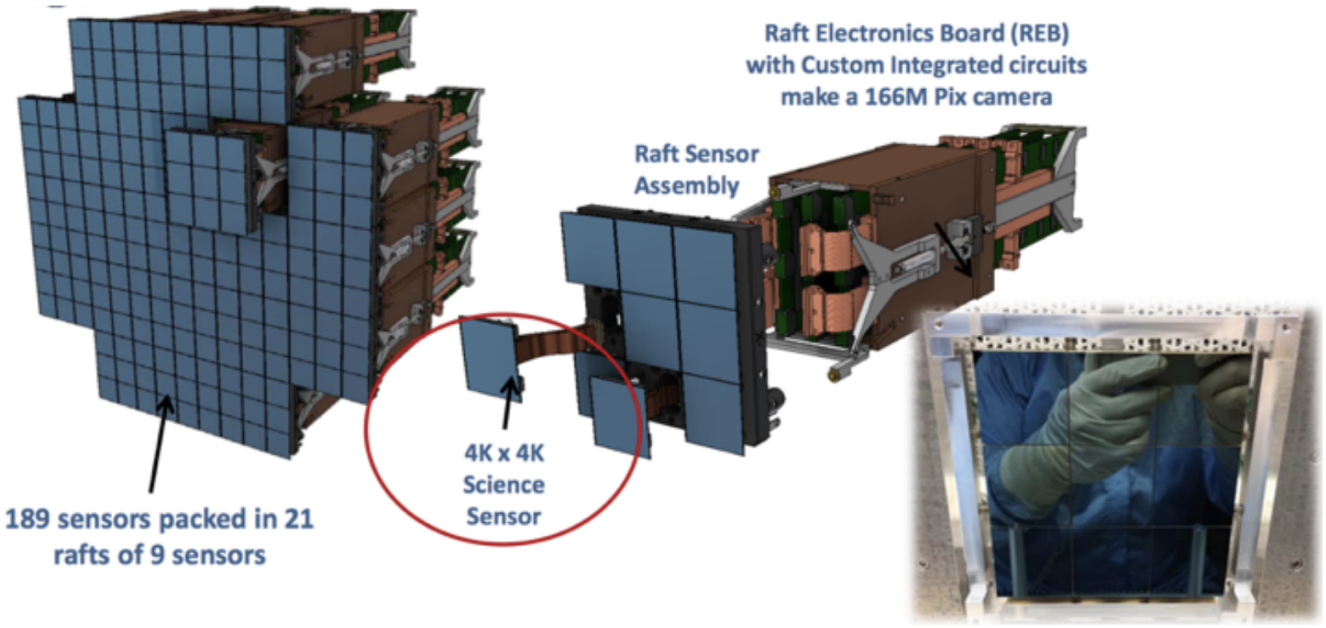
The mirror cell, the structure that supports the mirror, has 156 force actuators to compensate for deformations on the optical surface during observations — that's called active optics. Each actuator has a few sensors which produce data at 50Hz that need to be constantly analyzed and recorded.

The camera

Rubin Observatory created the world's largest digital camera for their telescope; it is a 3.2 gigapixel camera. It is the size of a small car and weighs around 6,200 pounds. As seen in the diagram below, before reaching the focal plane the image is corrected by a system of lenses and then recorded by a mosaic of 189 Charge-coupled devices (CCDs), which are used in digital cameras to measure light. LSST's camera has 189 CCD's in a mosaic pattern.

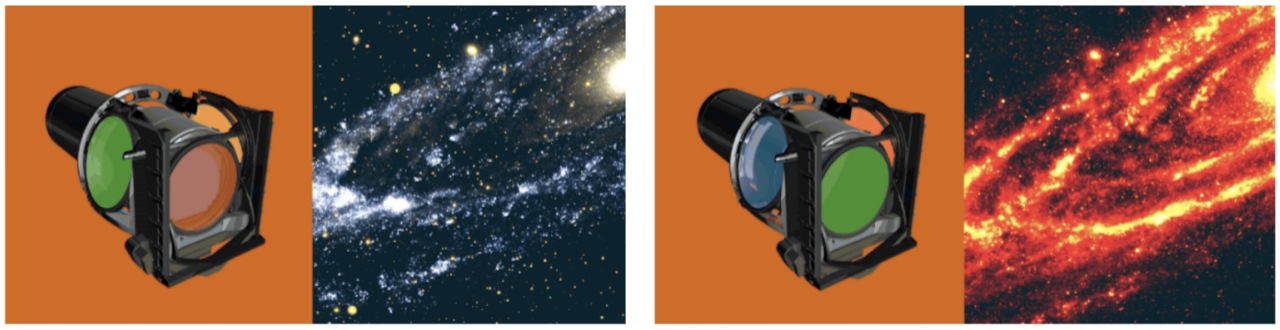


Each CCD takes a 16 megapixel image, and the images are arranged on 21 platforms or “rafts”. Every raft consists of 3x3 sensors or 9 CCD’s. Every CCD takes photos which are 4K by 4K pixels.



They need to keep the camera cool to avoid thermal deformation of the camera’s body due to the heat generated by the electronics. The Camera cooling system helps to keep every raft’s temperature at around -100 Celsius and prevent noise in every sensor.

The camera has a filter-changing mechanism that was built in France. By analyzing the light in different wavelengths, astronomers can understand the physical properties of stars and galaxies. As shown below, by applying different filters, astronomers can better analyze the same galaxy. Using the green filter, different attributes of the same stars and galaxies are more visible. The filter-changing mechanism was created in France.



The telescope

Rubin Observatory's team likes to think of the telescope as a discovery machine. It weighs 350 tons, moves quickly and has to be able to stop and settle in just four seconds. For every position, the Telescope Camera takes two exposures, 15 seconds long each, and then the Telescope points to the next position in the sky and repeats that about 1,000 times to collect about 20TB of image data every night. The Telescope Mount Assembly was built by engineers and scientists in Spain.

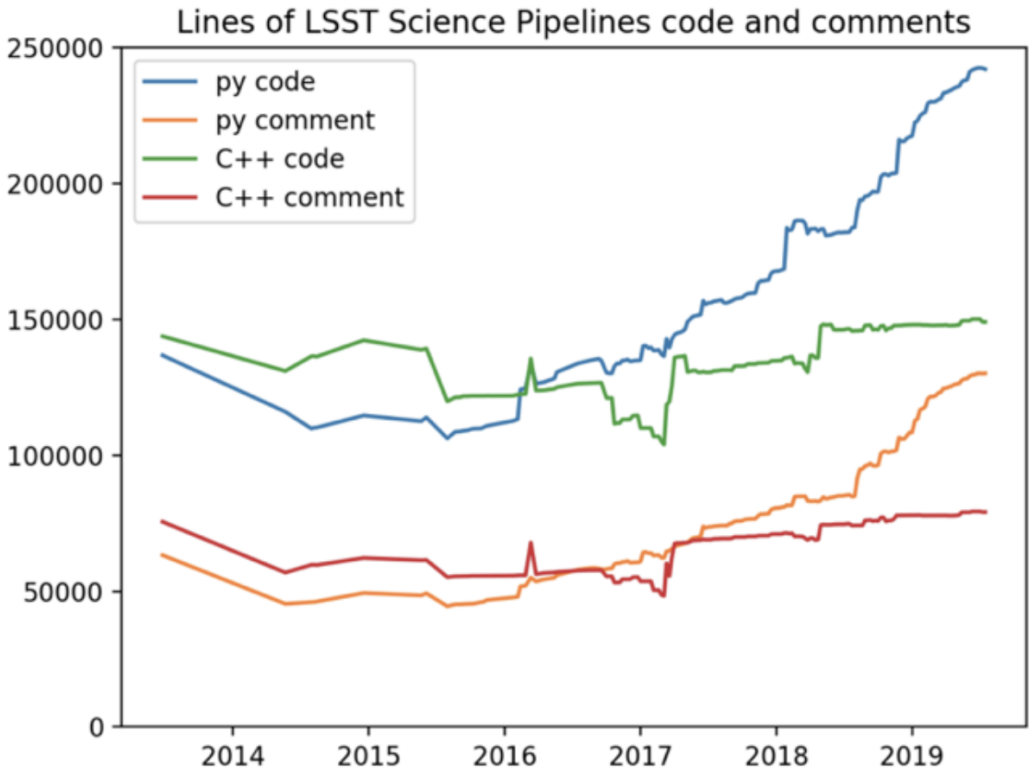
The Telescope Mount Assembly

The technical challenge

To measure the properties of every object detected in the images, they are developing the LSST Science Pipelines that will enable optical and near-infrared astronomy in the big data era by collecting and processing the data about the images. The software stack is open source with the core code written in C++. The rest of the stack is predominantly written in Python. The stack will be used to process the data daily and to produce annual releases and will be a big lift to the community that can use the stack to analyze the data. The Rubin Observatory will also provide services to facilitate data access and processing through a Jupyter-notebook-based environment.

In addition to astronomical data, Rubin observatory produces large amounts of engineering data that is critical to understand how the different parts of the telescope and the observatory are functioning during operations and if the software algorithms are doing the right thing. Here starts their journey using time series data, metrics and monitoring.

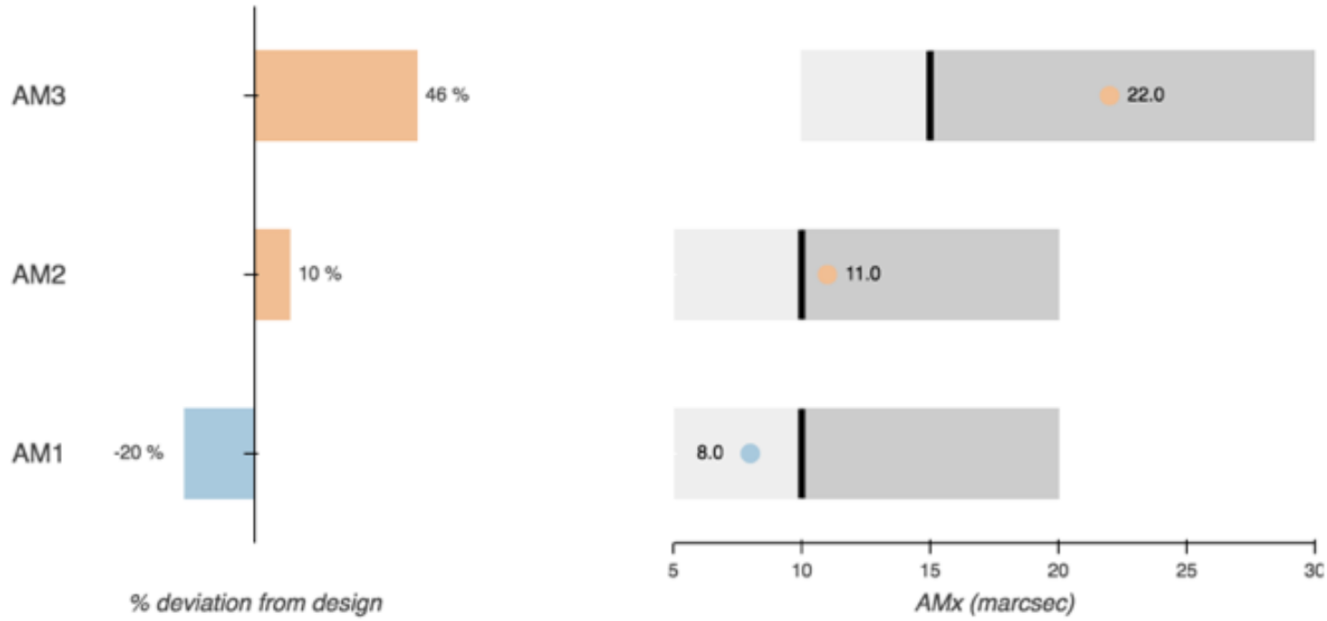
Prior to having a time series database, they were already tracking their data over time. Since 2014, they have tracked C++ and Python (py) codes and comments. They wanted to be able to quantify if they were doing the right thing. Before using InfluxDB, in 2016, Dr. Angelo Fausti and his team created their Science Quality and Software Harness System (SQuaSH), to start tracking their data over time.



Application monitoring

Even prior to using a time series database, they were already tracking metrics over time. Since 2016, they have been tracking code changes in the LSST Science Pipelines to understand how these changes in the software impact the scientific results.

The Rubin Observatory has science requirements to guide them through construction. For every metric they are held against, they have a minimum goal, a stretch goal and a design requirement. The following diagram depicts three metrics for astrometric performance: AM1, AM2 and AM3. The right panel indicates the minimum requirement (dark grey), the design requirement (black line), the stretch goal (light gray) and the actual values for each metric (points). The left panel shows how far the actual values deviate from the design requirements.



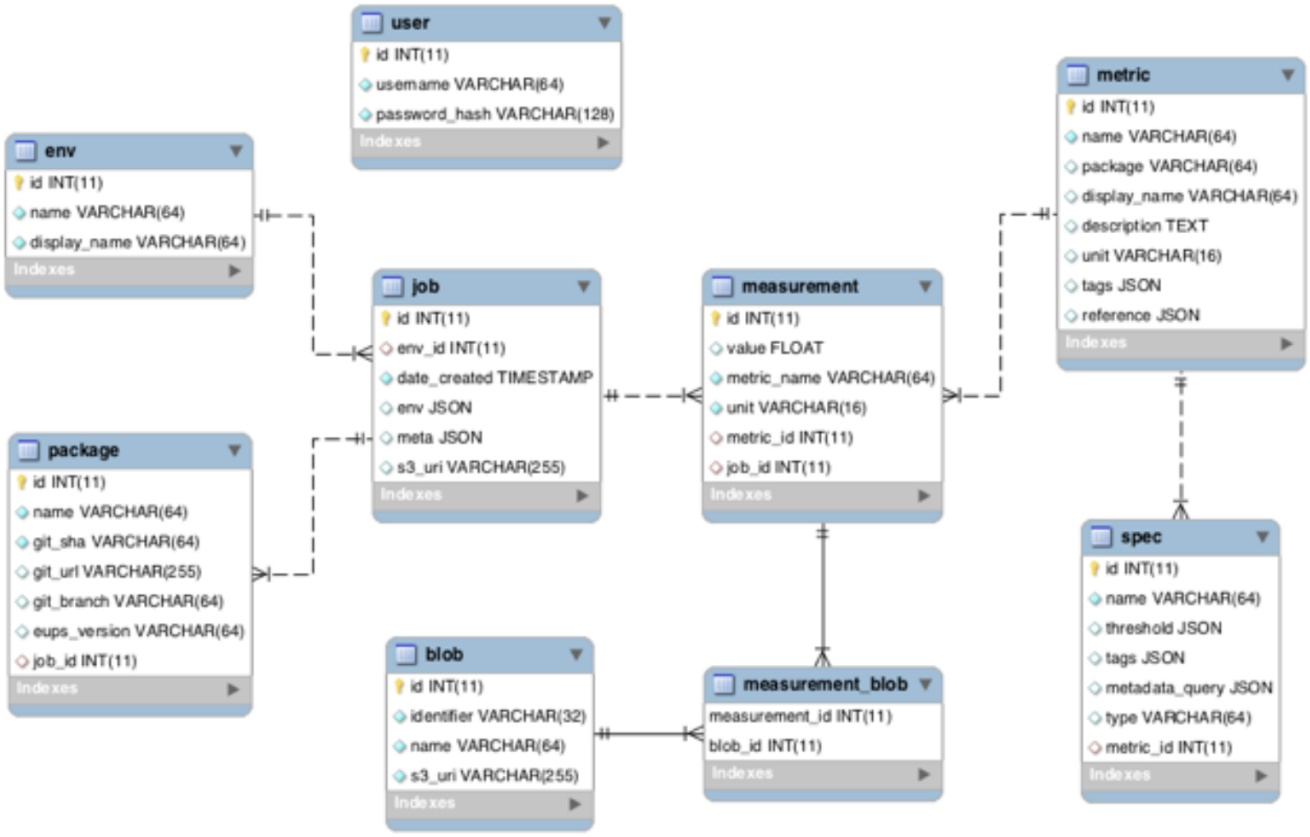
The SQuaSH system

Starting in 2016, they used their Science Quality and Software Harness System (SQuaSH) system to test the Rubin Observatory Science Pipelines' software daily as part of their Continuous Integration (CI) system. After every build, a verification system runs their code on fixed datasets. By repeating the verification step, they generate time series data that helps them to understand the impact of code changes in the scientific results.

If the Science Pipeline code or a configuration parameter changes, they can analyze the time series data to understand if it's a regression or an improvement. They collect metrics about their science algorithms and specifically care about storing metrics to analyze them later.

In addition to the metric name and its value, their verification system stores semantic information about the metrics, such as description, unit, reference to relevant documentation, arbitrary tags as well as thresholds or range specifications that help them to test the metric values.

The first implementation of SQuaSH used a relational database. The figure below shows the database schema:



- Used MySQL
- Created dashboards to monitor how code changes impacted the scientific results
- Provided developers and managers visibility into the software development process

The SQuaSH UI consists of a set of interactive dashboards to track metrics with time. From the SQuaSH UI, the team could go back into the continuous integration (CI) job that produced the results or inspect the software packages that changed, going straight to the GitHub commit associated with a particular metric excursion.

SQuaSH worked great for small data volumes, but they also need to monitor code changes for the entire lifetime of the project. Including the 10-year survey, they will have 15+ years worth of data and thousands of metrics. They knew they couldn't stick with their MySQL implementation as they had already spent too much time developing the SQuaSH UI.

The Rubin Observatory was already collecting time series data. As shown below, they were tracking metrics to determine how well they were doing over time compared with their requirements.

IoT

Rubin Observatory needs to store telemetry data from the numerous sensors in the mirror, camera, telescope and other subsystems of the observatory. They call that the Engineering Facility Database (EFD).

Telemetry streams combine physical or electrical data with a timestamp. Although it's not technologically feasible to include data from every Rubin component in the EFD, the goal is to capture as much useful data as possible and make it available to the scientists, engineers, and technicians who need it to make sure everything is working properly.

The primary EFD runs at the mountain in Chile and is used by the observers for real-time monitoring. Because the computing resources are limited there, they decided to store data for the past 30 days only and replicate the EFD data to a second instance that runs at the US Data Facility where the historical data is stored and is available to the project staff for analysis, such as trend analysis.

The solution

When developing SQuaSH back in 2016, Fausti was not aware of time series databases. As he points out, “relational databases are generic tools, but not necessarily the right tool to solve your specific problem”. He considered implementing a time series database in MySQL but realized that he would waste time and money and probably not implement a performant and feature-complete solution. Dr. Fausti decided to find an existing tool to address his needs.

The team used DB-Engines to help them to pick and compare time series databases. They considered and tested InfluxData’s InfluxDB platform and other solutions: Honeycomb, Datadog, Graphite, OpenTSDB, and Prometheus with Grafana. Dr. Fausti still remembers his manager’s “metric” for success: “If you need more than three days to make it work, it isn’t the right solution for you”. They ultimately picked InfluxDB, the purpose-built time series database.

Why InfluxDB?

In less than three days, the Rubin Observatory's team had a prototype pushing their data into InfluxDB, and they were analyzing and visualizing it with Chronograf. They started playing with Kapacitor for alert notifications. They found the InfluxDB platform easy to use and flexible. They value the fact that it is open source and made a few contributions to the code base.

The InfluxDB platform provided them with the flexibility that they needed. In choosing InfluxDB as their solution, the team valued that InfluxQL's syntax is similar to SQL, as they were already familiar with it and the fact that InfluxDB provides an HTTP API to interact with the database.

They have re-architected SQuaSH using the InfluxDB platform. They also have customized Chronograf to suit their needs. They use the open source JavaScript library React.JS to create UI components. As astronomers often work late at night in the observatory, they like Chronograf's dark mode which is much easier to read. Chronograf has enabled astronomers to analyze different datasets, depending on the specific project.

A feature that the Rubin Observatory's team relies on are Chronograf's annotations. Dr. Fausti points out that in SQuaSH, they want to understand the causation behind changes in their metrics, and it's great being able to add annotations on the time series to provide context for future reference. From time to time, management has meetings where they look at the charts and make annotations in Chronograf to better understand what has changed.

They have created dashboards and alert rules using Kapacitor. All alert notifications get pushed out through Slack. As Dr. Fausti points out, "this is great because we get notifications on Slack, and in the same channel, we discuss what happened and how to fix the problem".

Getting started

To adopt InfluxDB in SQuaSH, the first step was to understand time series database concepts and how the data is organized in measurements, fields and tags. Dr. Fausti points out that once you realize how to translate your problem to InfluxDB concepts, then you can build a data model that works for you, and there's probably different ways to do that.

When migrating from MySQL to InfluxDB, Fausti and team took the time to ensure the team understood how to use the new platform. They started by showing a table with data they were familiar with and applied InfluxData's terminology. The table consisted of data from a verification package:

Time	ci_id	ci_dataset	instrument	filter	visit	ccdnum	AssociationTime	numNewDiaObjects	numUnassociatedDiaObjects
2018-12-07T14:25:59Z	67	CI-HiTS2015	DECAM	g	411371	56	0.84	116	22
2018-12-07T14:33:21Z	67	CI-HiTS2015	DECAM	g	411371	60	0.82	68	285
2018-12-07T14:41:00Z	67	CI-HiTS2015	DECAM	g	411420	10	0.7	131	0
2018-12-07T14:48:32Z	67	CI-HiTS2015	DECAM	g	411420	5	0.65	175	0
2018-12-07T14:55:55Z	67	CI-HiTS2015	DECAM	g	419802	10	1.19	267	94
2018-12-07T15:03:16Z	67	CI-HiTS2015	DECAM	g	419802	5	1.24	315	136
2018-12-08T14:27:42Z	68	CI-HiTS2015	DECAM	g	411371	56	0.84	116	22
2018-12-08T14:35:09Z	68	CI-HiTS2015	DECAM	g	411371	60	0.81	68	285
2018-12-08T14:42:32Z	68	CI-HiTS2015	DECAM	g	411420	10	0.67	131	0
2018-12-08T14:49:53Z	68	CI-HiTS2015	DECAM	g	411420	5	0.64	175	0
2018-12-08T14:57:22Z	68	CI-HiTS2015	DECAM	g	419802	10	1.17	267	94
2018-12-08T15:04:58Z	68	CI-HiTS2015	DECAM	g	419802	5	1.21	315	136

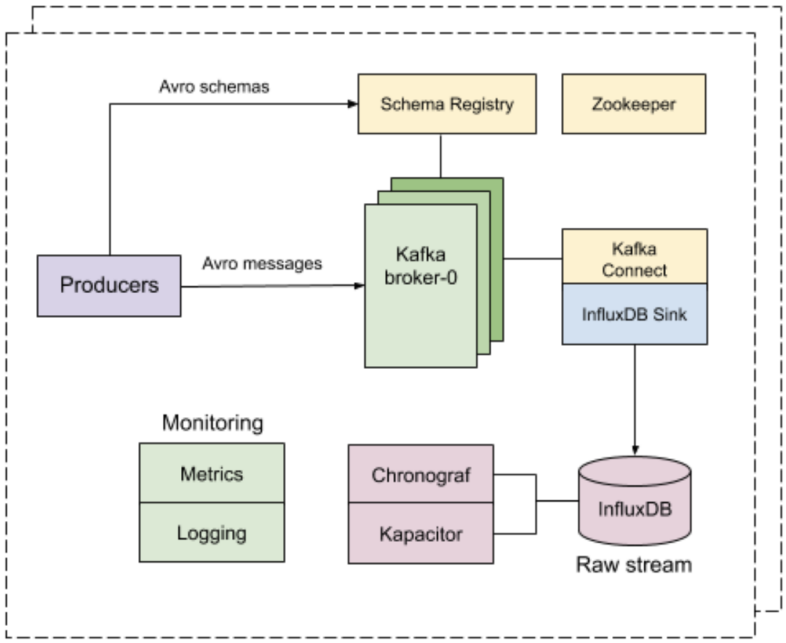
They use the following analogy to explain InfluxDB concepts:

- A measurement in InfluxDB is similar to a table.
- A field in InfluxDB is the actual metric and event data.
- A tag in InfluxDB can be used for any metadata associated with the metric or the job that produced it

Another important difference between tags and fields is that tags are indexed in the database while fields are not. The team were shown examples of Line Protocol using their own data. Dr. Fausti also ensured his team understood more advanced topics like series cardinality. Cardinality all depends on how tags have been designed. By adopting InfluxDB to solve the SQuaSH problem, Dr. Fausti had time to work on other problems, and that's how he got involved in the EFD project.

From the experience using the InfluxDB platform in SQuaSH and the fact that the InfluxDB platform was designed with the IoT use case in mind, InfluxDB was a perfect fit for the EFD problem. However, the EFD presents interesting challenges given the large number and variety of systems to monitor in the Rubin Observatory.

Technical architecture

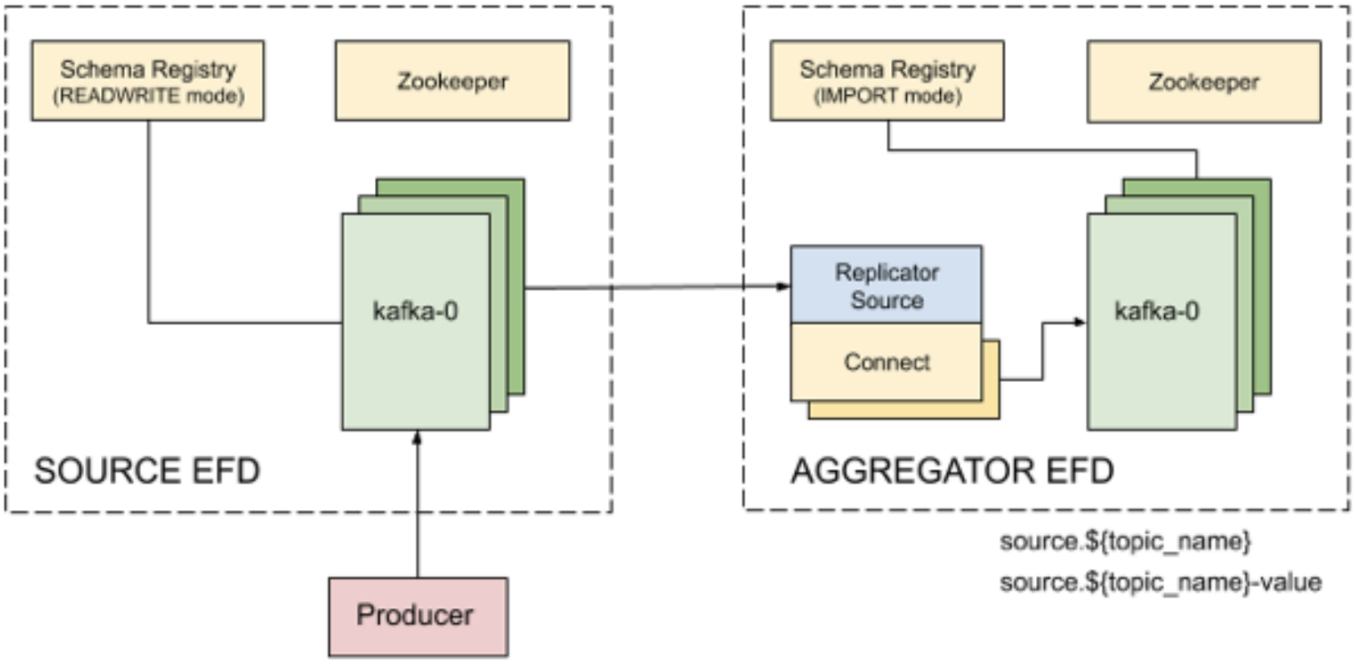


Shared Architecture between SQuaSH and the EFD

They use their own client to collect metrics and events and then the data is pushed to Kafka. The client is dynamic, as it automatically detects new topics in data distribution service (DDS). DDS is the communication middleware used on their edge devices and sensors. They parse the DDS IDL in real time which describes the topic schema, transforms it to the other schemas, and uploads it to the Kafka schema registry. Their Kafka brokers have a 24-hour retention policy. If there is any downtime with InfluxDB, they are able to recover the data from Kafka.

They use the InfluxDB Sink Connector to write data to InfluxDB; the connector converts their data from Apache Avro to InfluxDB. They appreciate that it's free; however, they ran into issues. Initially, the connector wasn't working well, but they worked through their challenges and were able to contribute back to Apache Avro. To manage the connector and make it dynamic, they developed a utility called Kafka Connect Manager.

Data replication architecture



The EFD data needs to be replicated from the Observatory in Chile to the US Data Facility, currently at the National Center for Supercomputing Applications (NCSA). They initially used the Kafka Replicator Connector for that but are looking at open-source alternatives to solve this problem. At the US Data Facility the EFD data is going to be stored for the entire lifetime of the project. Most of the data streams don't need to be stored in full resolution. For that, they implemented a Kafka aggregator based on the Faust Python Stream Processing library.

Results

Rubin Observatory's team learned best practices along the way. They determined that having fewer series with more points in their InfluxDB instance is better than having more series with fewer points as it leads to better performance. They learned that as the number of tags increases, so does the time series cardinality. For example, they realized they need to record the job ID as a field instead of a tag to avoid cardinality issues.

Dr. Angelo Fausti and his team took the time to research and understand InfluxDB concepts and how to better design their databases. While they got the initial prototype running quickly, they wanted to make sure that in the long run, they were using the tools correctly. Starting by understanding the InfluxDB concepts such as measurements, tags, fields and Line Protocol helped immensely. They needed to make sure that their existing tools and processes would work with InfluxDB's concepts.

> *"You can design your database in several different ways. It's just a matter of trying and seeing which one works best for you."*

Dr. Angelo Fausti, Software Engineer, Vera C. Rubin Observatory

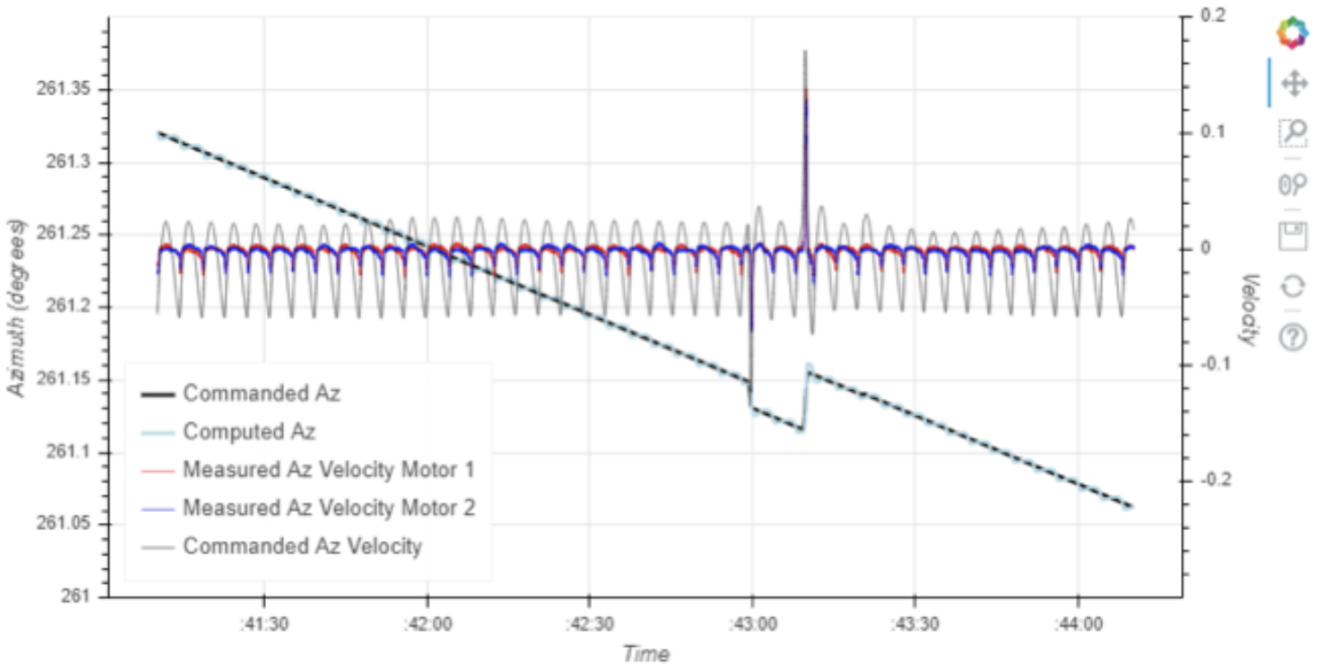
Rubin Observatory is also using InfluxDB to monitor their infrastructure which is running at the Observatory and at the Data Facility. It's being used to monitor their hardware — Dr. Fausti values that they use the same technology stack across various applications.

Fausti expands to say they are using machine learning (ML) for analysis of astronomical data. An application of ML is to classify the astronomical objects detected in the images. "We'll have so many images with a large number of objects on them. ML will help us to classify those objects and understand the data better," said Dr. Angelo Fausti, who realizes there will likely be additional uses for machine learning.

His team is using InfluxDB to monitor their Kubernetes deployments with the default dashboard provided by InfluxData. For some of their deployments, they use a lightweight version of Kubernetes — Kubes (K3's). Dashboards were important — on both their on-premises deployments using K3s and Google Cloud Platform, they were useful to understand how the cluster is behaving.

Advanced analysis

Rubin Observatory's team uses the asynchronous Python client (aioinflux) to perform advanced analysis in Jupyter notebooks. Jupyter notebooks are used to create and share data with team members that contain code, equations, visualizations and text. Fausti's colleague, Simon Krughoff, was able to compare the computed and commanded velocity of the telescope's motor. Krughoff needed to be able to combine time series data points together. The combination of InfluxDB and Jupyter notebooks made this analysis possible.

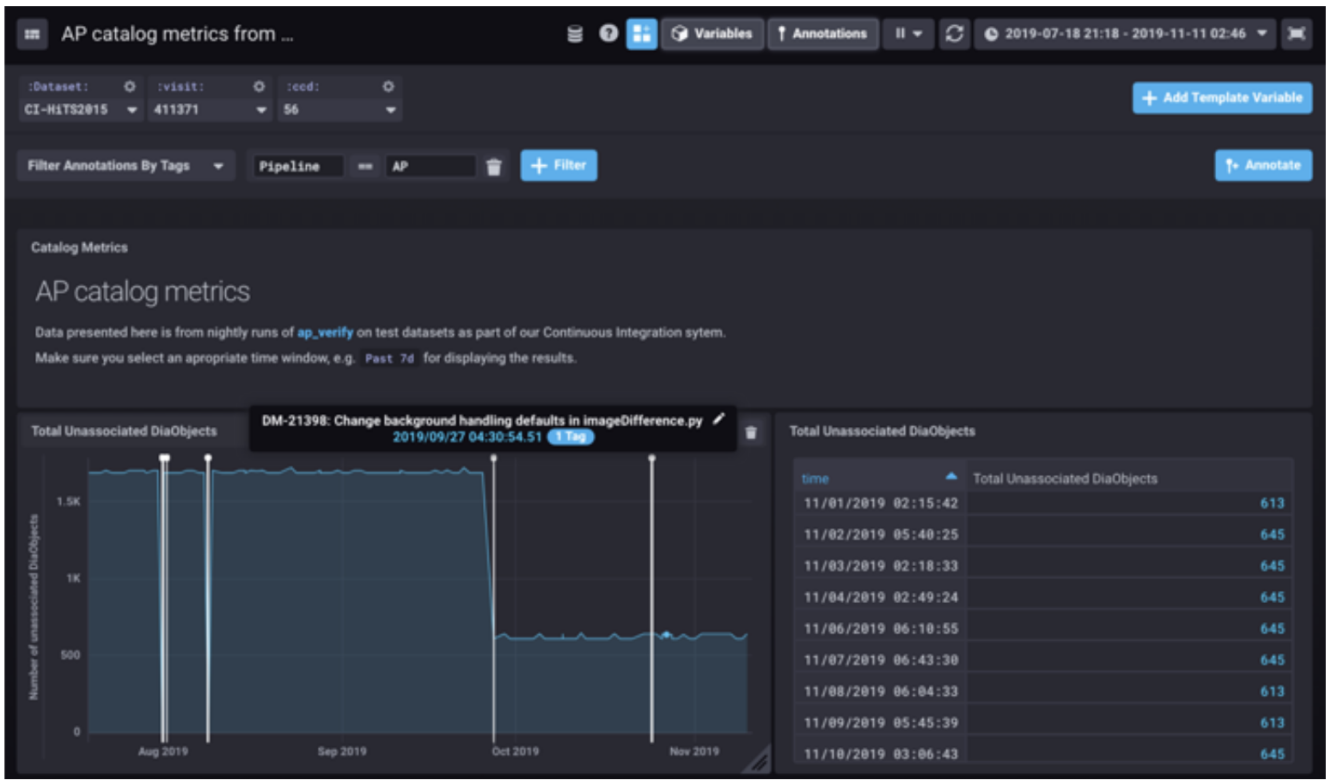


Chronograf dashboards

They use Chronograf to understand their various systems. As their teams are distributed, being able to look at the same dashboards is very helpful. As their data volumes grow, they need to analyze and view their data in more efficient ways.

Annotations

Being able to answer questions like “What happened on September 27th?” has become invaluable. As previously mentioned, Rubin Observatory’s team likes being able to add annotations to their Chronograf dashboards. They are able to add notes directly to dashboards to explain changes in the data. Fausti’s boss, Frossie Economou, Technical Manager, believes that annotations are more important than the data itself as it helps explain the data and the results.



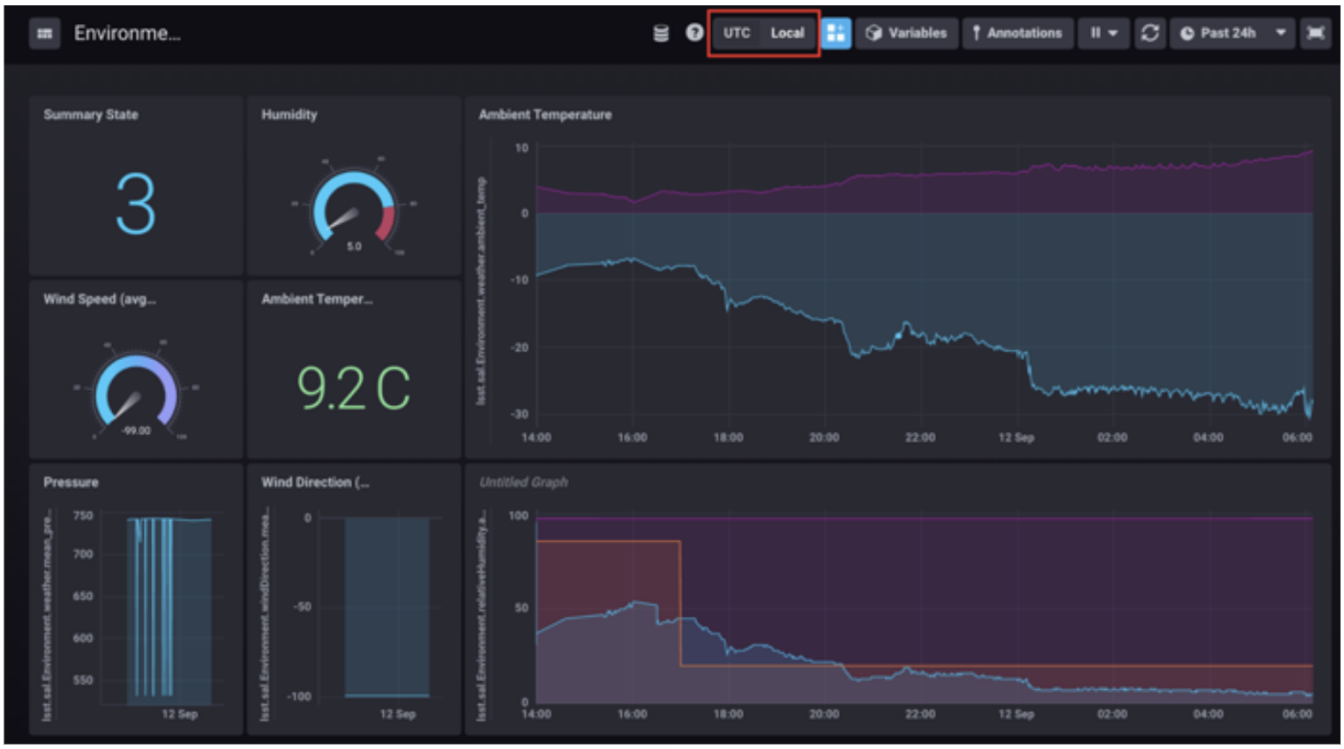
CCD monitoring

In this dashboard, Rubin Observatory's team is able to view data from:

- Six temperature sensors on the electronic board
- Two sensors on the integrated ASPIC

Weather Station in Chile

In Chronograf, they can change the timezone from UTC to local time. This is key as they have people working in different timezones. They can all look at the same dashboard and use Chronograf's timezone toggle.



Summary State

Rubin Observatory has a dashboard showing an overall summary of all systems running at the observatory.

- 1: Disabled
- 2: Enabled
- 3: Fault
- 4: Offline
- 5: Standby

“Chronograf gives us the flexibility to create dashboards that suit our needs [in] different situations.”

Tiago Ribeiro, Scheduler Scientist, Vera C. Rubin Observatory

As Dr. Fausti points out, Ribeiro had never used Chronograf before. Ribeiro went to Chile to use the Auxiliary Telescope and started playing with Chronograf. He created some of these dashboards in one afternoon.

Alerting with Kapacitor

Based on analytics performed with InfluxDB and Jupyter notebooks, Fausti and team has alerts sent to Slack via Kapacitor using the Slack event handler. All sensors are pushing information to the team through multiple streams and from multiple physical locations. Having all of the alerts sent to Slack has been great for Rubin Observatory's team as it creates a lot of good discussion. It allows people to follow the current status of everything, and they are able to make comments.

> *"Kapacitor has been critical in sounding the alarm when the system is not behaving correctly."*

Simon Krughoff, Science Lead, Vera C. Rubin Observatory

Stress testing latency in InfluxDB

As discussed, their mirror cells have 156 sensors and they are collecting sensor data from about 120 different measurements at 50 hertz. They were concerned about whether Kafka and InfluxDB would be able to handle the data stream. They were unsure if they could read data from InfluxDB while also writing to InfluxDB. They needed to measure the end-to-end latency of the system.

They note the time a message was sent. After it goes through their entire system, they'd also have the time it was written to InfluxDB. The latency was the difference between the write and send times. They determined latency was low, at around .06 seconds. They were writing about 100,000 points per minute into InfluxDB. They noticed little bumps as latency increased. Fausti points out that these are okay as they signaled when they were querying the data into the database. They saw the system would recover quickly. Therefore, they knew their systems would be able to handle large data streams, especially from the mirror cell's subsystem with the high rate of 50 hertz.

What's next?

Rubin Observatory's team is excited to try InfluxDB 2.0. Having taken Flux training at InfluxDays San Francisco 2019, Fausti enjoyed the training and is eager to start using the new query language. Some of their team are concerned about switching from InfluxQL to Flux, yet Fausti was eager to point out that the capabilities of Flux will address some of their current challenges.

They've found that downsampling and data retention is made much easier with Flux. They are able to downsample data at different levels of resolution depending on the age of the data. They have some data in its raw format for a week; for a month, it's downsampled to a degree; and data older than a year is downsampled even more. Their retention policy deletes old data as it hits a certain age threshold. They are playing with the statistical methods built into Flux which Fausti describes as being similar to Holt-Winters' predictions for machine learning.

“Flux is much more flexible because it is not just a query language...The ability to combine data from different sources is great.”

Dr. Angelo Fausti, Software Engineer, Vera C. Rubin Observatory

As soon as there’s more feature parity, they will start implementing the newest version of InfluxDB. As Chronograf’s annotations are vital to their operations, they want to make sure they continue being able to use this functionality.

They are continuing to test their solutions with increasing data volumes and variety. This will be especially important as they assemble the telescope’s subsystems. Fausti mentions that their team is still determining which resources are essential and how much raw data they need to store for the duration of the experiment.

Prior to using Kubernetes for deployments, they used tools like Terraform. They use Kubernetes to automate EFD deployments. The team is switching to ArgoCD which Angelo Fausti feels is a better continuous delivery tool. Dr. Fausti likes that all configurations, secrets, etc. are stored in a GitHub repository. Rather than doing manual changes directly to deployments, they first do it on GitHub, and then ArgoCD handles the synchronization. He values their UI as they can see all of their Kubernetes objects and their health statuses. If needed, his team can delete all of them from the UI. Dr. Fausti enables the team to manage their own deployments without them needing to know tools like Terraform. As he explains, some tools are difficult for people to use if they aren’t using them daily.

As the experiment continues, there will be new challenges for Dr. Fausti and his team to address. They plan to use InfluxDB for the observatory electronic log system where observers can make comments. They are also considering using InfluxDB to store all astronomical data. The team views all stars, planets, and other astronomical objects as sensors, as they will be collecting data about each object for the entire 10-year survey. As the data about all astronomical objects will change over the 10 years, Rubin Observatory's cohort hopes to track all changes in InfluxDB.

About InfluxData

InfluxData is the creator of InfluxDB, the leading time series platform. We empower developers and organizations, such as Cisco, IBM, Lego, Siemens, and Tesla, to build transformative IoT, analytics and monitoring applications. Our technology is purpose-built to handle the massive volumes of time-stamped data produced by sensors, applications and computer infrastructure. Easy to start and scale, InfluxDB gives developers time to focus on the features and functionalities that give their apps a competitive edge. InfluxData is headquartered in San Francisco, with a workforce distributed throughout the U.S. and across Europe. For more information, visit influxdata.com and follow us @InfluxDB.

influxdata®

Try InfluxDB

Get InfluxDB

Contact us for a personalized demo influxdata.com/get-influxdb/