DigitalOcean

akass@ + dmi@

Building a Robust Analytics Platform

with an open-source stack

What's coming up:

1) DigitalOcean - a company background
2) Data @ DigitalOcean
3) The Big Data Tech Stack @ DO
4) Use-cases + Demo

What is DigitalOcean?

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- ***4 years old***

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- *4 years old*
- ***12 Data Centers globally***

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- *4 years old*
- *12 Data Centers globally*
- ***Over 30 Million VMs created***

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- *4 years old*
- *12 Data Centers globally*
- *Over 30 Million VMs created*
- ***In 196 Countries***

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- *4 years old*
- *12 Data Centers globally*
- *Over 30 Million VMs created*
- *In 196 Countries*
- ***700K Developer Users***

What is DigitalOcean?

A Cloud Hosting Company for Software Developers.

- *4 years old*
- *12 Data Centers globally*
- *Over 30 Million VMs created*
- *In 196 Countries*
- *700K Developer Users*
- ***30K Developer Teams***

Data @ DO - Participating Teams

- Data & Analytics ("DnA" - Analysts + Data Scientists/Engineering)
  - ***(Alex & Dao live here)***
- Platform/Infrastructure Engineering
- Product Engineering
- Security

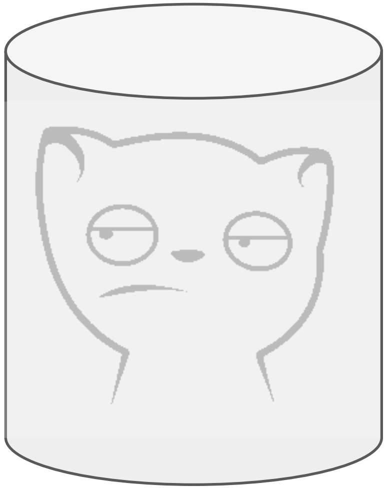
Data @ DO

- **Product Usage**

Data @ DO - **Product Usage**

- *Product Revenue Forecasting*
- *Churn Prediction*
- *User Segmentation*
- *Beta product uses and product cannibalization*
- *Support Team Efficacy*

Data @ DO - **Product Usage, v1.0**



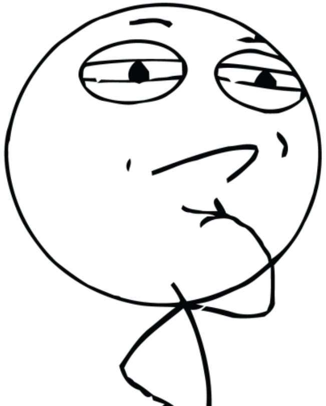
Pain Point 1: General Data Architecture

Huge, unwieldy SQL Tables → Slooooow, monolithic, unadaptive

Pain Point 2: Upstream Dependencies

E.g. Monthly Invoicing → Revenue analytics done on monthly basis

Data @ DO - **Product Usage, v2.0**



Revision 1: General Data Architecture

Microservices + Kafka pass application-level events →
Faster and more robust, but ***teams must build their own consumers***.

Revision 2: Active Downstream Consumption

E.g. Granular Billable Events → Daily, hourly, even near-RT processing of revenue for ingestion into analysis

digitalocean.com

Data @ DO

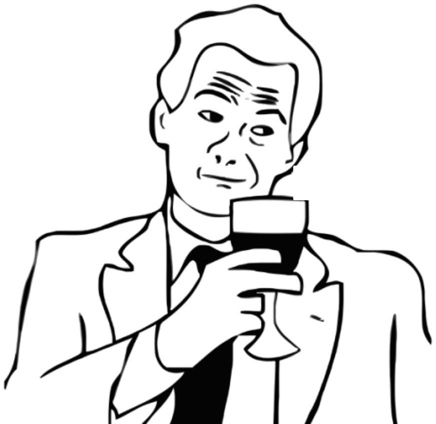
- Product Usage
- **Sales/Marketing Leads**

Data @ DO - **Sales/Marketing Leads**

Problem:

- *User behavioral data lives in AWS Redshift*

- *User metadata lives in MySQL on DO's cloud*



Data @ DO - **Sales/Marketing Leads**

Problem:

- *User behavioral data lives in AWS Redshift*
- *User metadata lives in MySQL on DO's cloud*

Solution:

- *Migrate warehousing to our own cloud so that all data stays on-premise*

Data @ DO

- Product Usage
- Sales/Marketing Leads
- **Infrastructure**

Data @ DO - **Infrastructure**

Problem - nay - <u>Conundrum</u>:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

VMs

Data @ DO - **Infrastructure**

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

VMs

Data @ DO - **Infrastructure**

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

Nodes

Hypervisors

VMs

Data @ DO - **Infrastructure**

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

Nodes

Hypervisors

VMs

Data @ DO - Infrastructure

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

Nodes

PDU

Hypervisors

VMs

S.M.A.R.T.ctl

Data @ DO - **Infrastructure**

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*

Nodes

PDU

Hypervisors

VMs

Data @ DO - **Infrastructure**

S.M.A.R.T.ctl

Problem - nay - Conundrum:

- *Every 5 minutes, our entire active VM fleet is polled for OS and HW data using Prometheus and other in-house scraping solutions*
- ***Significant scale (too big for RDBMS), inherent silos***

Nodes

PDU

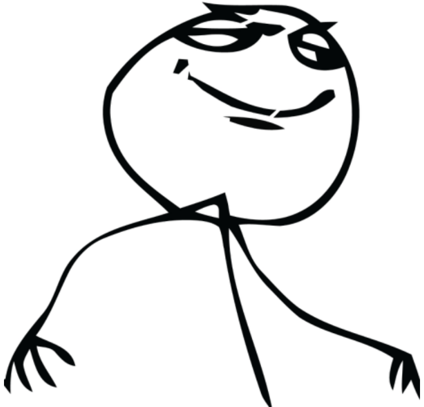
To recap:

- Product Data in MySQL is **slow** and **isolated**
- Sales/Marketing data are **isolated** in different warehouses
- Infra data are **prohibitively large** and **isolated**

We need to reimagine how we process and store *everything*.

We need to reimagine how we process and store *everything*.

We need to reimagine how we process and store *everything*.

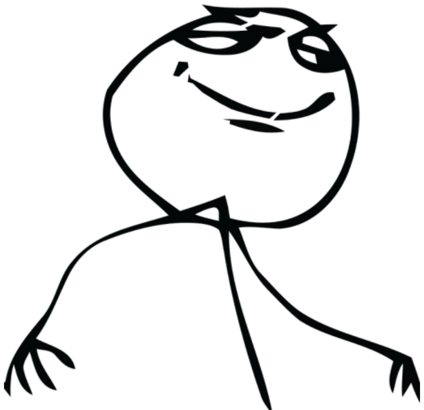


We need to reimagine how we process and store *everything*.

digitalocean.com

What is DigitalOcean?

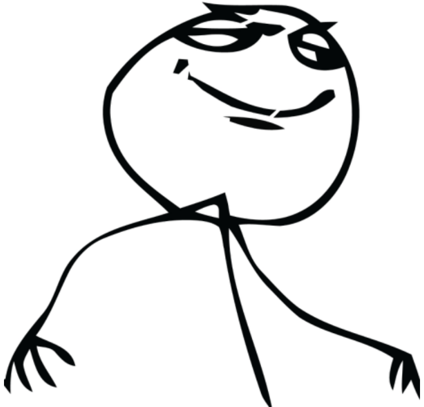
A Cloud Hosting Company for Software Developers.



digitalocean.com

What is DigitalOcean?

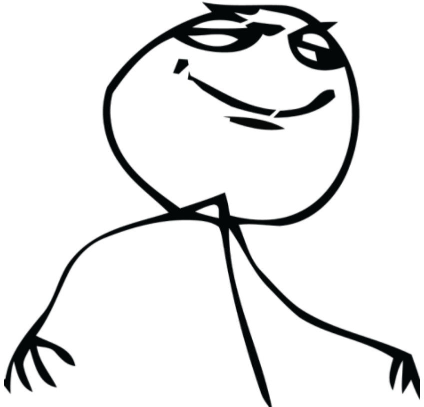
A Cloud Hosting Company = We have cloud infrastructure.



What is DigitalOcean?

A Cloud Hosting Company = We have cloud infrastructure.

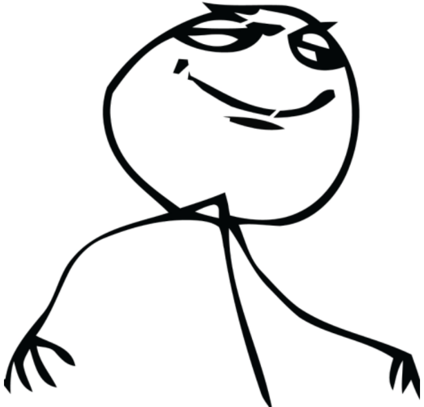
= We can build our own big data environment.



What is DigitalOcean?

A Cloud Hosting Company = We have cloud infrastructure.

= We can build our own big data environment.



digitalocean.com

The DO Big Data Stack

The DO Big Data Stack

MESOS

elasticsearch

jupyter

kafka

presto

Spark

The DO Big Data Stack

MESOS

1. **Distributed Systems Management**

The DO Big Data Stack

MESOS

1. Distributed Systems Management
2. **Parallel Compute**

Spark

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. **Standardized Compute Environments**

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. **Distributed Data Warehousing**

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. Distributed Data Warehousing
5. **Distributed Streaming Events System**

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. Distributed Data Warehousing
5. Distributed Streaming Events System
6. **Massively Parallel Query Engine**

The DO Big Data Stack

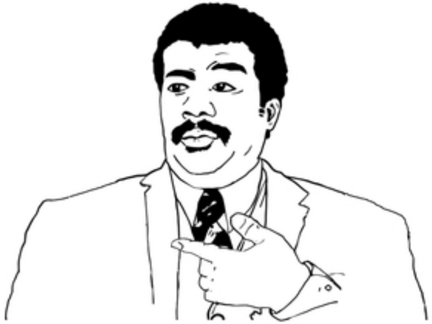
1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. Distributed Data Warehousing
5. Distributed Streaming Events System
6. Massively Parallel Query Engine
7. **Unifying IDE**

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. Distributed Data Warehousing
5. Distributed Streaming Events System
6. Massively Parallel Query Engine
7. Unifying IDE
8. **Logging at Scale**

The DO Big Data Stack

1. Distributed Systems Management
2. Parallel Compute
3. Standardized Compute Environments
4. Distributed Data Warehousing
5. Distributed Streaming Events System
6. Massively Parallel Query Engine
7. Unifying IDE
8. Logging at Scale

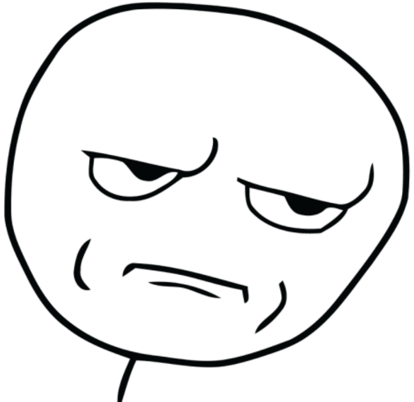


“Buzz.”

– Hive

“Buzz.”

– Hive



Use Case 1: A New ETL Pipeline for Support Ticket Events

Measuring Customer Satisfaction (CSAT)
Integration with Internal Ticketing
More transparency for Support Team

Use Case 1: A New ETL Pipeline for Support Ticket Events

Subscribe to new Kafka topic

```
akass@[redacted]$ python3 print_sample_event.py -t lifecycleevents
b'\n1\n$21c19a1f-3f3f-497c-a0d6-ba197c5bbfe6\x12\x06\x08\x8f\xa1\xaa\xc0\x05\x18\xfe\x01j\x
1e\x10\xe6\x88B\x18\xac\xaajR\x14\x10\xe6\x88B\x1a\x06\x08\xd1\xf3\xda\xba\x05"\x06\x08\x8f
\xa1\xaa\xc0\x05'
```

Use Case 1: A New ETL Pipeline for Support Ticket Events

Subscribe to new Kafka topic

Prototype reading of Kafka data within a Spark-enabled Jupyter Notebook

In [5]:
```
import lifecycle_events_pb2
from protobuf_to_dict import protobuf_to_dict

event = lifecycle_events_pb2.Event()
```

In [6]:
```
def decoder(s):
    """ Decode the object as bytes"""
    if s is None:
        return None
    return event.FromString(s)
```

Use Case 1: A New ETL Pipeline for Support Ticket Events

Subscribe to new Kafka topic

Prototype reading of Kafka data within a Spark-enabled Jupyter Notebook

Update Spark Docker container to include new modules and configurations, if necessary

```
&& tar xf protobuf-$PROTOBUF_VERSION.tar.gz

COPY autogen.sh /protobuf-$PROTOBUF_VERSION/

RUN cd /protobuf-$PROTOBUF_VERSION && ./autogen.sh \
&& ./configure --prefix=/usr && make install \
&& cd / && rm /protobuf-$PROTOBUF_VERSION.tar.gz \
&& ./Anaconda3-$ANACONDA_VERSION-Linux-x86_64.sh -b -p /opt/anaconda3 \
&& rm Anaconda3-$ANACONDA_VERSION-Linux-x86_64.sh \
&& pip install protobuf==3.0.0 kafka-python protobuf3-to-dict
```

```
$ cat spark-defaults.conf | grep docker
spark.mesos.executor.docker.image
docker.internal.digitalocean.com/platform/spark:1929b8d
```

Use Case 1: A New ETL Pipeline for Support Ticket Events

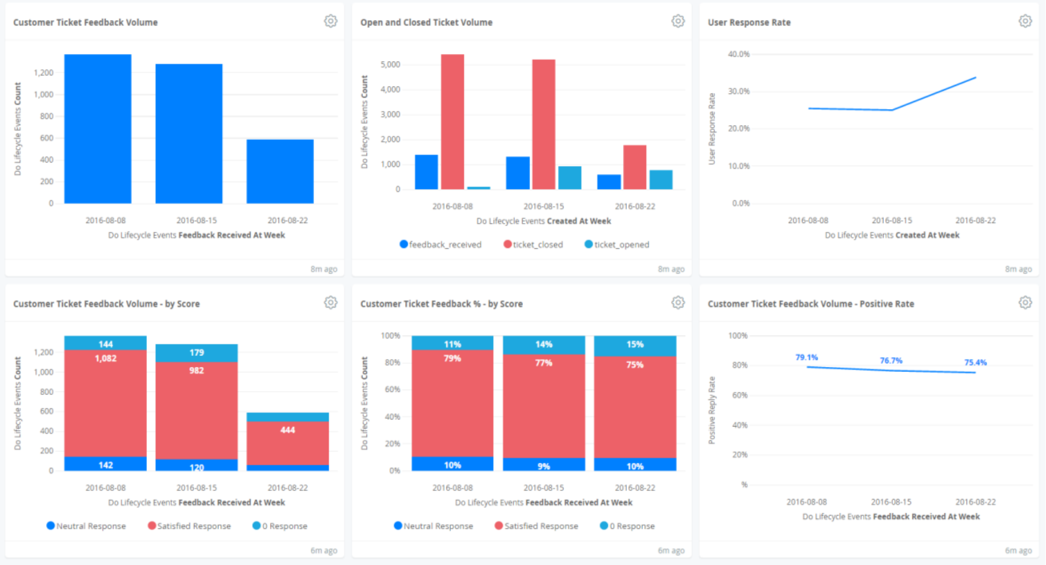
- Subscribe to new Kafka topic
- Prototype reading of Kafka data within a Spark-enabled Jupyter Notebook
- Update Spark Docker container to include new modules and configurations, if necessary
- **Write Spark script & deploy onto Mesos**

Active Frameworks

Find...

ID ▼	Host	User	Name	Role	Principal	Active Tasks	CPUs	GPUs	Mem	Disk	Max Share
...975e-94a0cddebd60-1849	test-dna-kafka-lifecycle.nyc3.internal.digitalocean.com	root	product_lifecycle_etl	spark		1	8	0	4.4 GB	0 B	0.150%

Use Case 1: A New ETL Pipeline for Support Ticket Events



Subscribe to new Kafka topic

Prototype reading of Kafka data within a Spark-enabled Jupyter Notebook

Update Spark Docker container to include new modules and configurations, if necessary

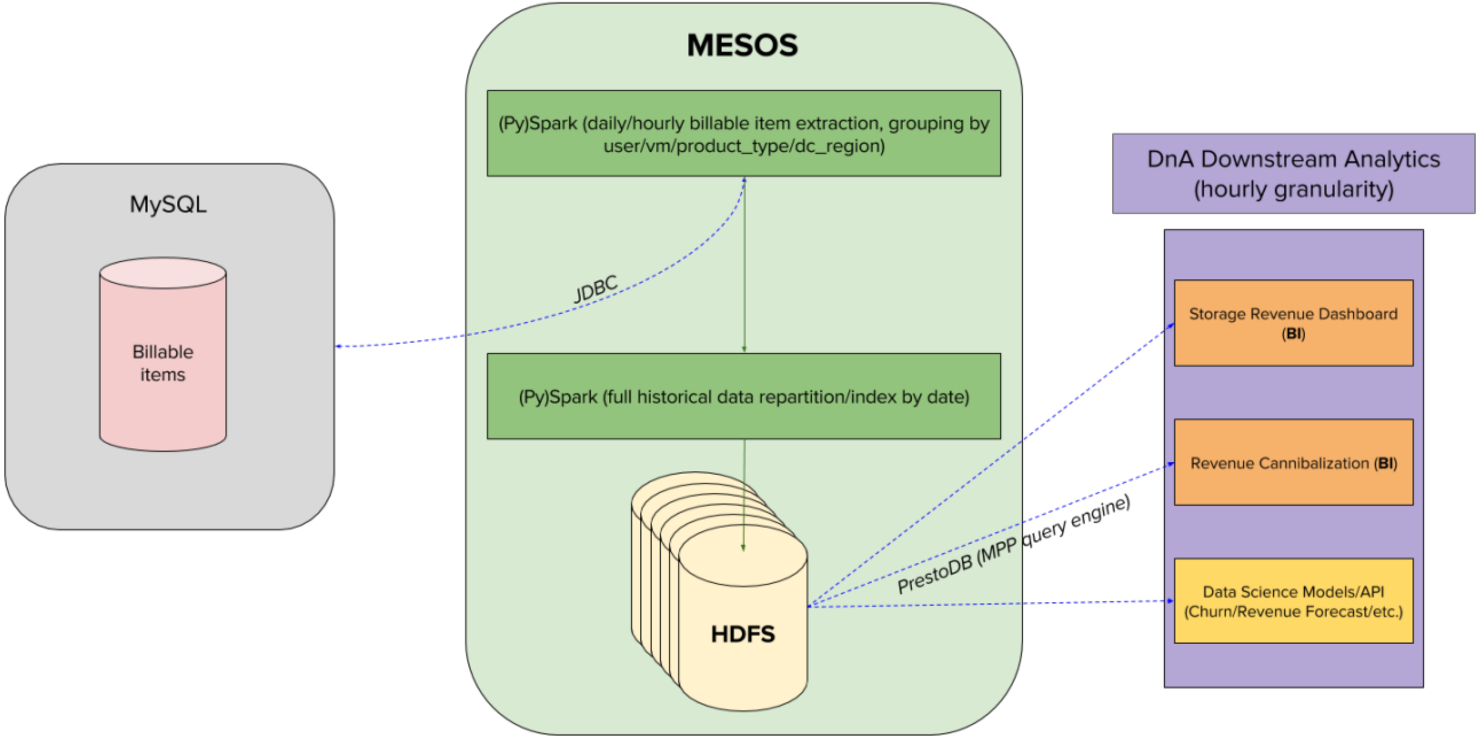
Write Spark script & deploy onto Mesos

Analyze!

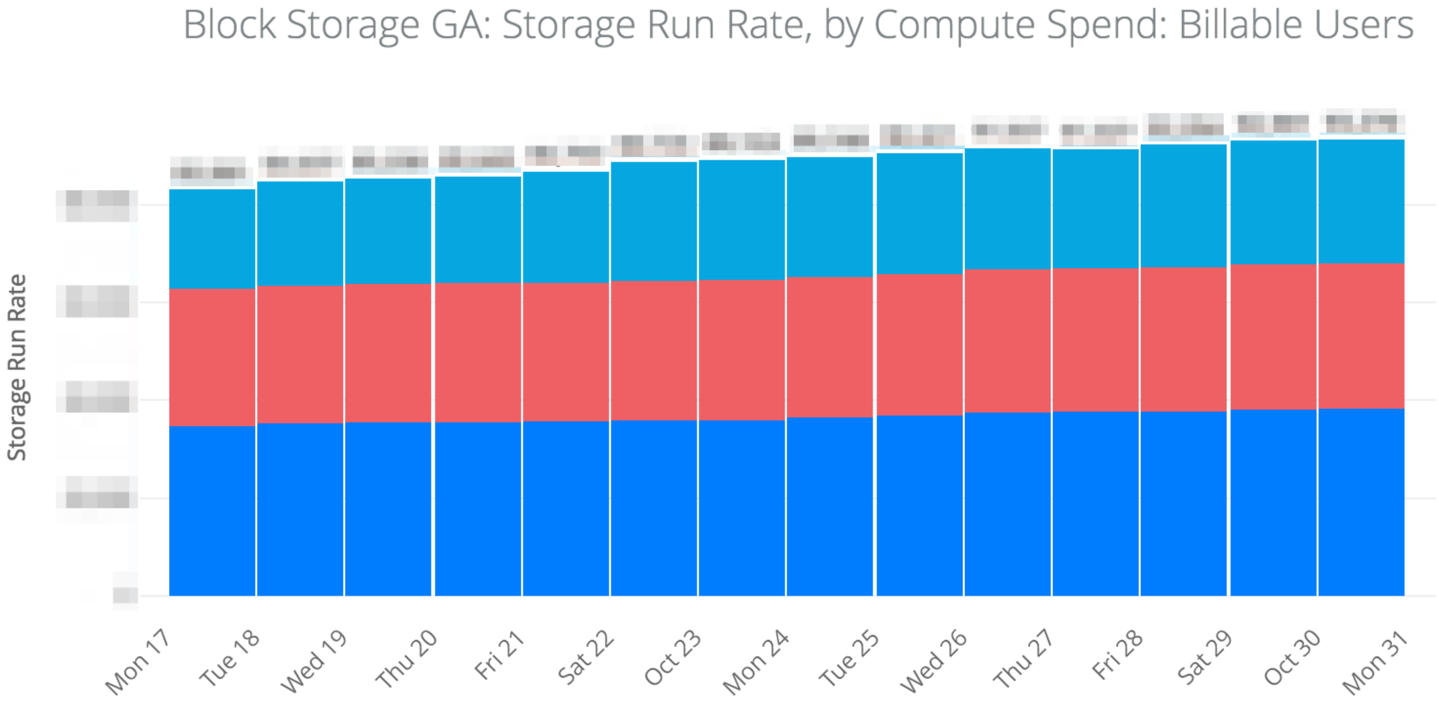
Use Case 2: Billable Data, from Months to Minutes

Reminder:
Pain Point 2: Upstream Dependencies E.g. Monthly Invoicing → Revenue analytics done on monthly basis
Revision 2: Active Downstream Consumption Goal → Increase temporal granularity to **daily, hourly, even near-RT** processing of revenue for ingestion into analysis

Use Case 2: Billable Data, from Months to Minutes



Use Case 2: Billable Data, from Months to Minutes



Use Case 3: Power Failure Detection + PDU Clustering

Real-World problem:

- **many** different drive models out in fleet…

Use Case 3: Power Failure Detection + PDU Clustering

Real-World problem:

- **many** different drive models out in fleet…

- how to predict performance stability/failures with consistency?

Use Case 3: Power Failure Detection + PDU Clustering

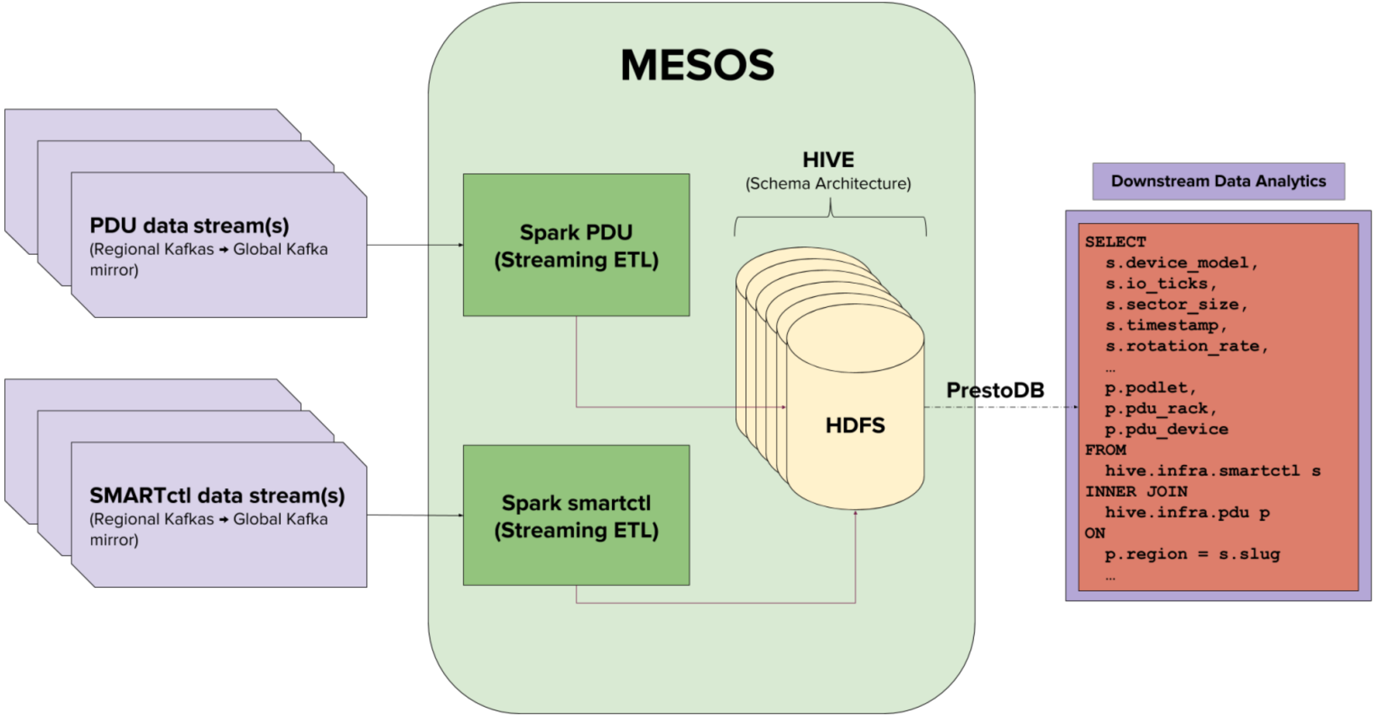
Solution:

1) Measure PDU patterns & fluctuations on a ***podlet/rack/device level***

2) Join to smartctl data to get vendor/drive-specific metadata

3) Mine for outliers to help DC teams react quicker to anomalies

Use Case 3: Power Failure Detection + PDU Clustering

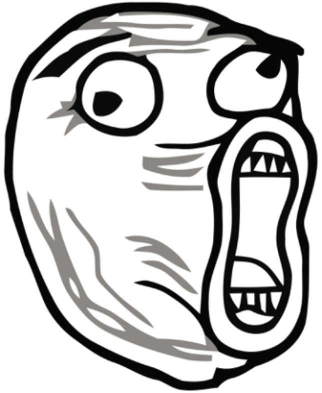


Use Cases in Action:

Let’s have a DEMO...

Use Cases in Action:

Let’s have a DEMO...



Lessons learned from work thus far...

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively
- Spark image containerization → consistency across Mesosphere

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively
- Spark image containerization → consistency across Mesosphere
- Multiple Docker images → developmental flexibility

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively
- Spark image containerization → consistency across Mesosphere
- Multiple Docker images → developmental flexibility
- Presto DB → significantly improved querying efficiency:
  - Naturally worked well with parallel data-stores on HDFS
  - Also improved RDBMS data retrieval through parallelization

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively
- Spark image containerization → consistency across Mesosphere
- Multiple Docker images → developmental flexibility
- Presto DB → significantly improved querying efficiency:
  - Naturally worked well with parallel data-stores on HDFS
  - Also improved RDBMS data retrieval through parallelization
- Laying pipelines to connect Kafka to Spark to HDFS was challenging;

Lessons **Learned** (a small sample thus far)

- Jupyter was indispensable for prototyping Spark + Kafka interactively
- Spark image containerization → consistency across Mesosphere
- Multiple Docker images → developmental flexibility
- Presto DB → significantly improved querying efficiency:
  - Naturally worked well with parallel data-stores on HDFS
  - Also improved RDBMS data retrieval through parallelization
- Laying pipelines to connect Kafka to Spark to HDFS was challenging; **tuning** everything was often even harder!

Recapping…

Recapping…

1) Consolidation/Centralization of data from across DO

Recapping…

1) Consolidation/Centralization of data from across DO

   Anyone can build reports/analyses

Recapping…

1) Consolidation/Centralization of data from across DO

   Anyone can build reports/analyses

2) Faster Reporting, Better Granularity

Recapping…

1) Consolidation/Centralization of data from across DO

   Anyone can build reports/analyses

2) Faster Reporting, Better Granularity
3) Tight coupling with the other engineering groups

Recapping…

1) Consolidation/Centralization of data from across DO

   Anyone can build reports/analyses

2) Faster Reporting, Better Granularity

3) Tight coupling with the other engineering groups

4) Modern stack allows massive scale with small headcount

Gazing into the **Future**...

Future Work

- Hardware failure detection forecasting to help the DC team perform predictable maintenance.

Future Work

- Hardware failure detection forecasting to help the DC team perform predictable maintenance.

Future Work

- Hardware failure detection forecasting to help the DC team perform predictable maintenance.

- Better inform hardware acquisition costs with detailed machine performance metrics.

Future Work

- Hardware failure detection forecasting to help the DC team perform predictable maintenance.

- Better inform hardware acquisition costs with detailed machine performance metrics.

- Profile hypervisors by usage to optimize how VMs are allocated to particular racks.

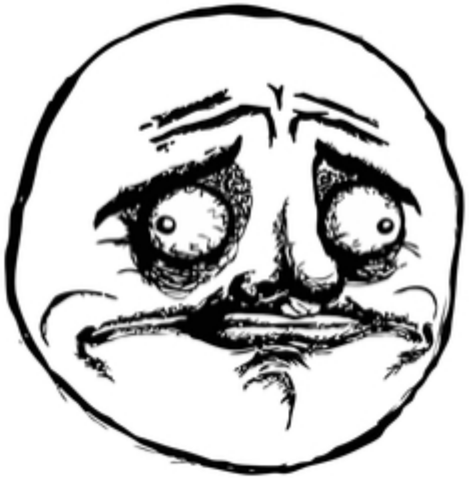
Future Work

- Hardware failure detection forecasting to help the DC team perform predictable maintenance.

- Better inform hardware acquisition costs with detailed machine performance metrics.

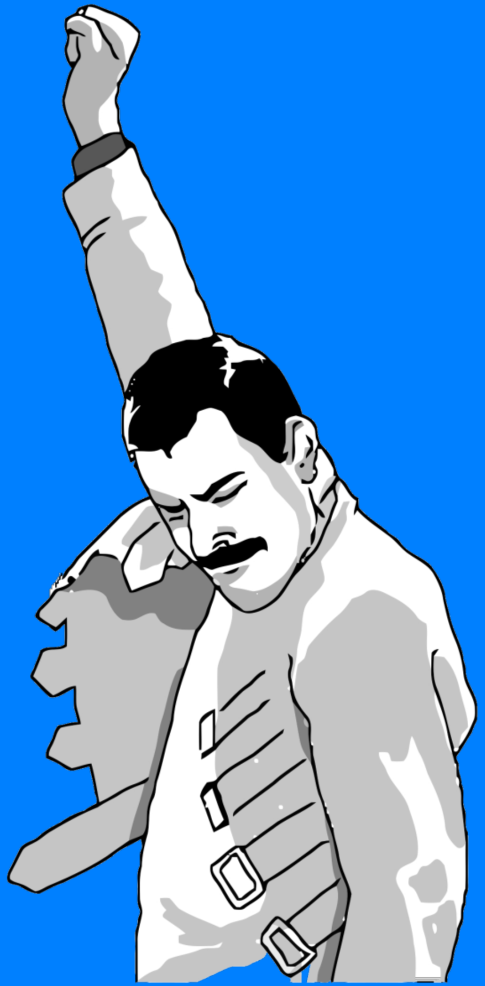
- Profile hypervisors by usage to optimize how VMs are allocated to particular racks.

- Beta-test/prototype/dogfood future products.



Questions?

Thank you!



DigitalOcean