Building highly reliable data pipelines @ Datadog

Reliability is the probability that a system will produce correct outputs up to some given time **t**.

Source: E.J. McClusky & S. Mitra (2004). "Fault Tolerance" in Computer Science Handbook 2ed. ed. A.B. Tucker. CRC Press.

Highly reliable data pipelines

1. Architecture

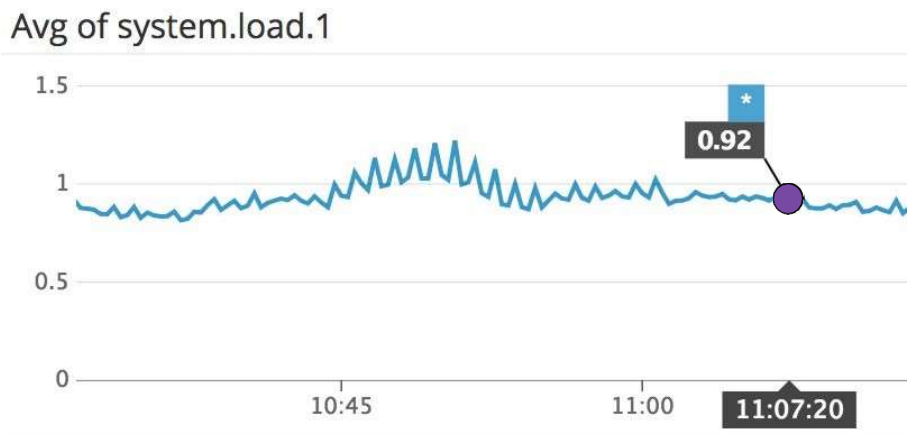
Highly reliable data pipelines

1. Architecture
2. Monitoring

Highly reliable data pipelines

1. Architecture
2. Monitoring
3. Failures handling

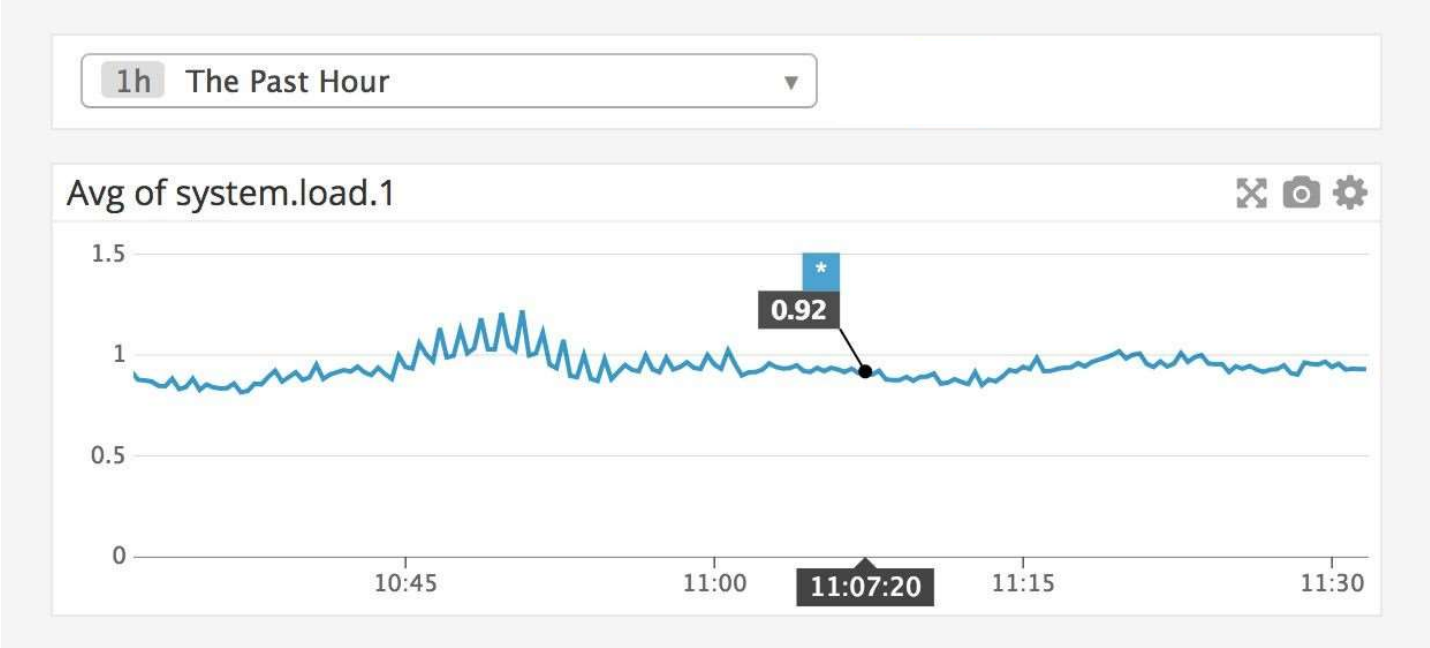
Historical metric queries



Time series data

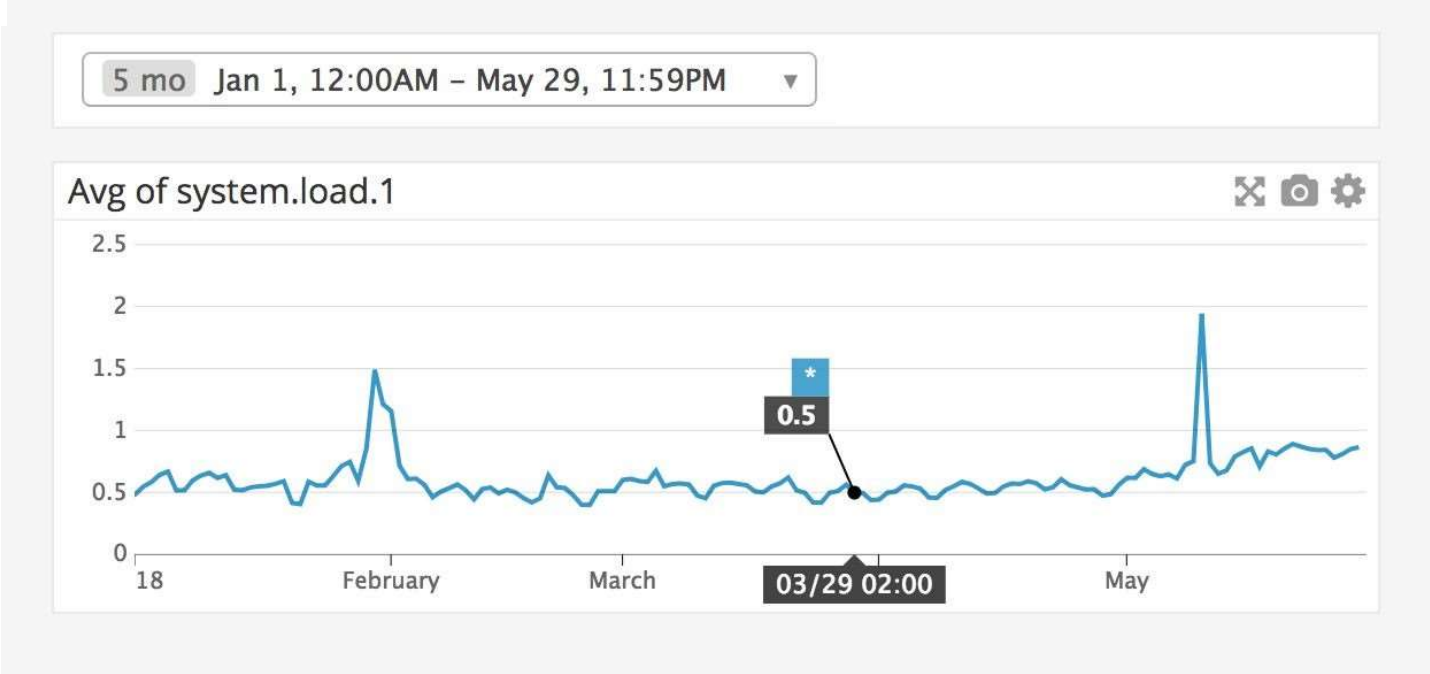
metric	system.load.1
timestamp	1526382440
value	0.92
tags	host:i-xyz,env:dev,...

Historical metric queries



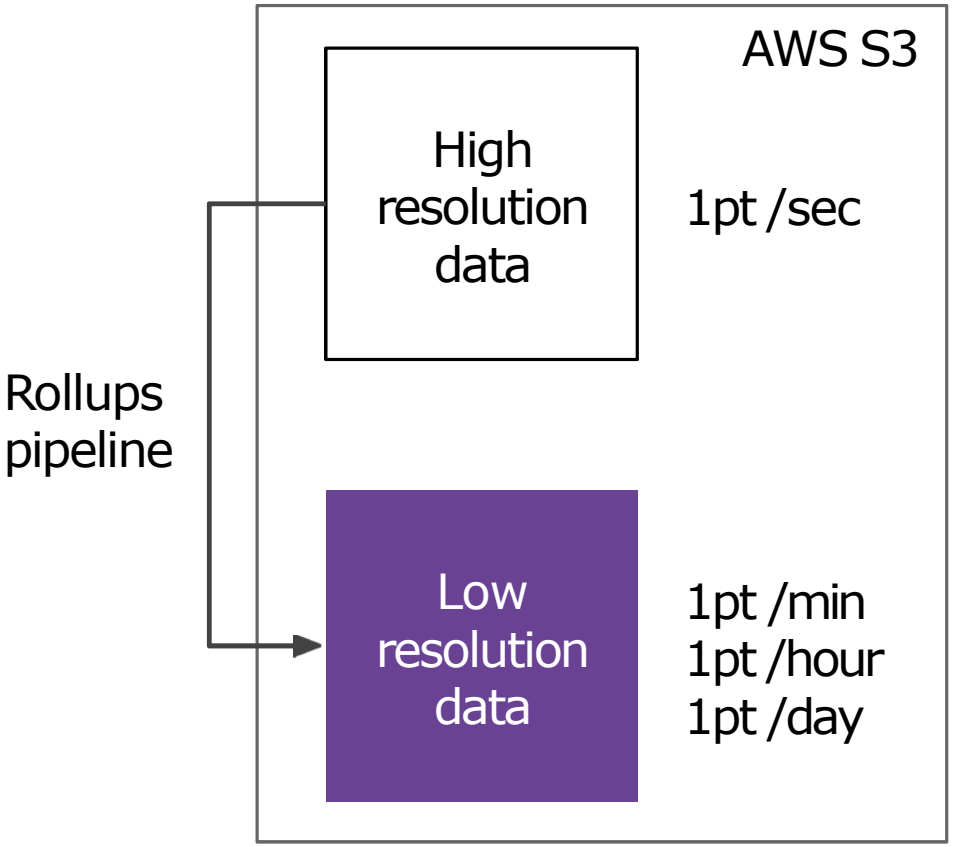
1 point/second

Historical metric queries



1 point/day

Historical metric queries

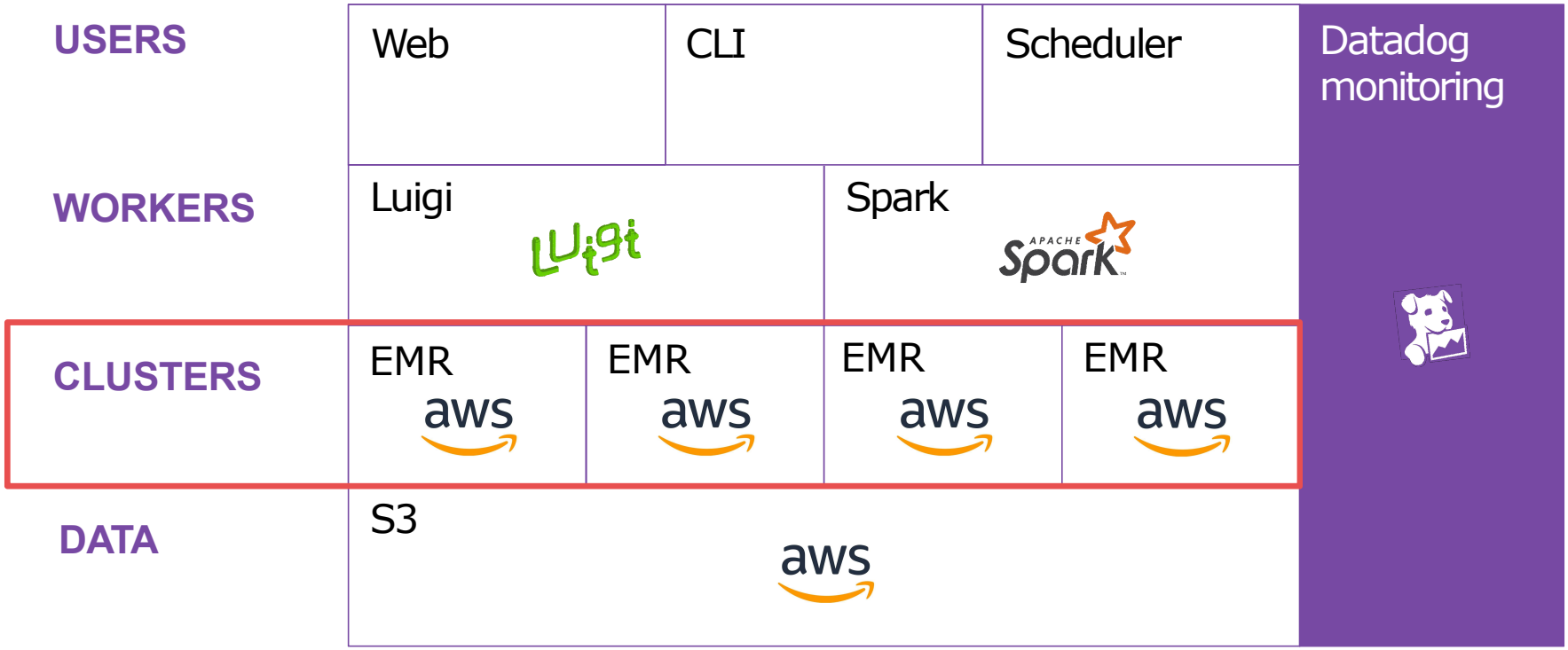


- Runs once a day.
- Dozens of TBs of input data.
- Trillions of points processed.

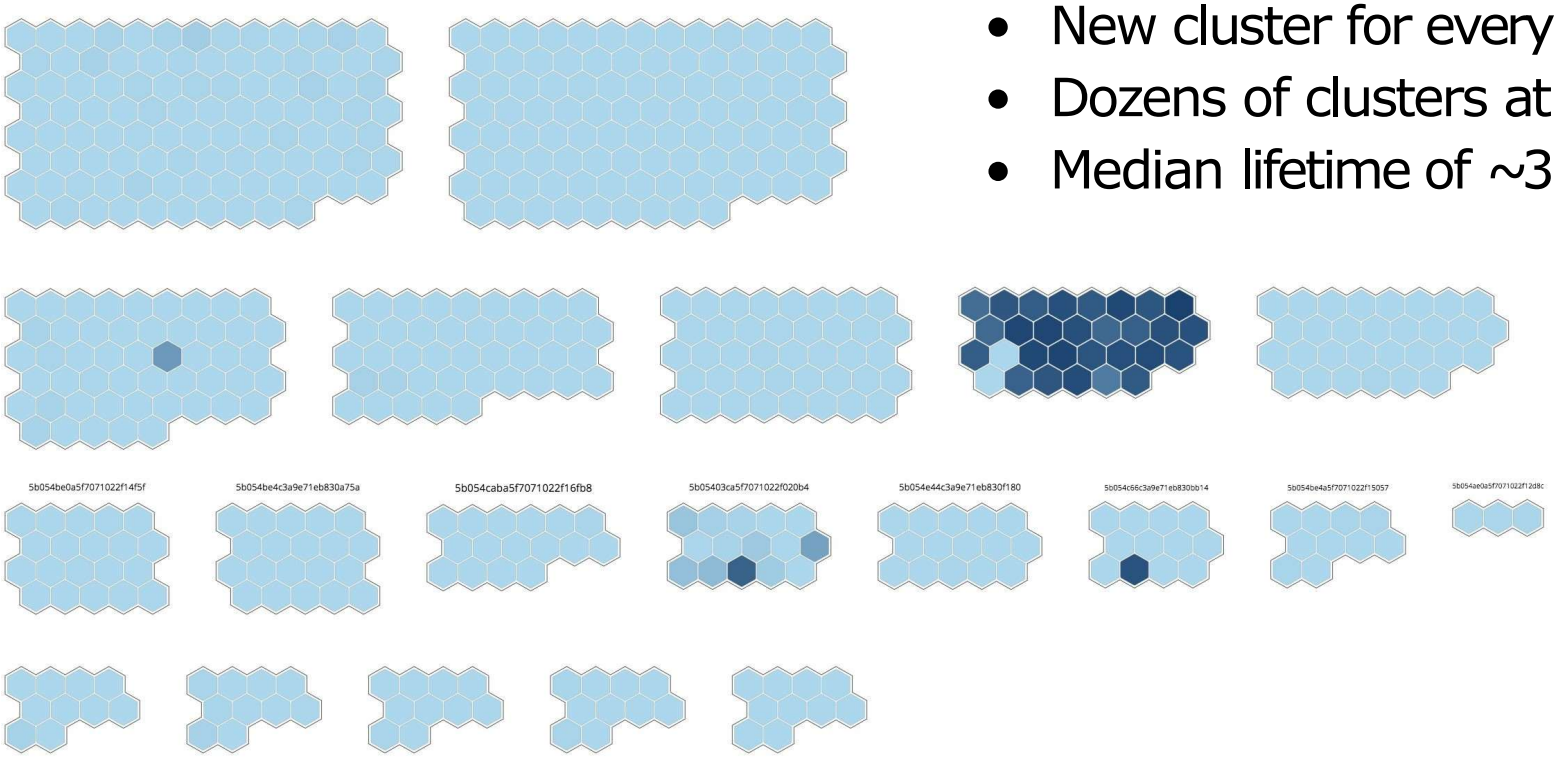
Highly reliable data pipelines

1. Architecture
2. Monitoring
3. Failures handling

Our big data platform architecture

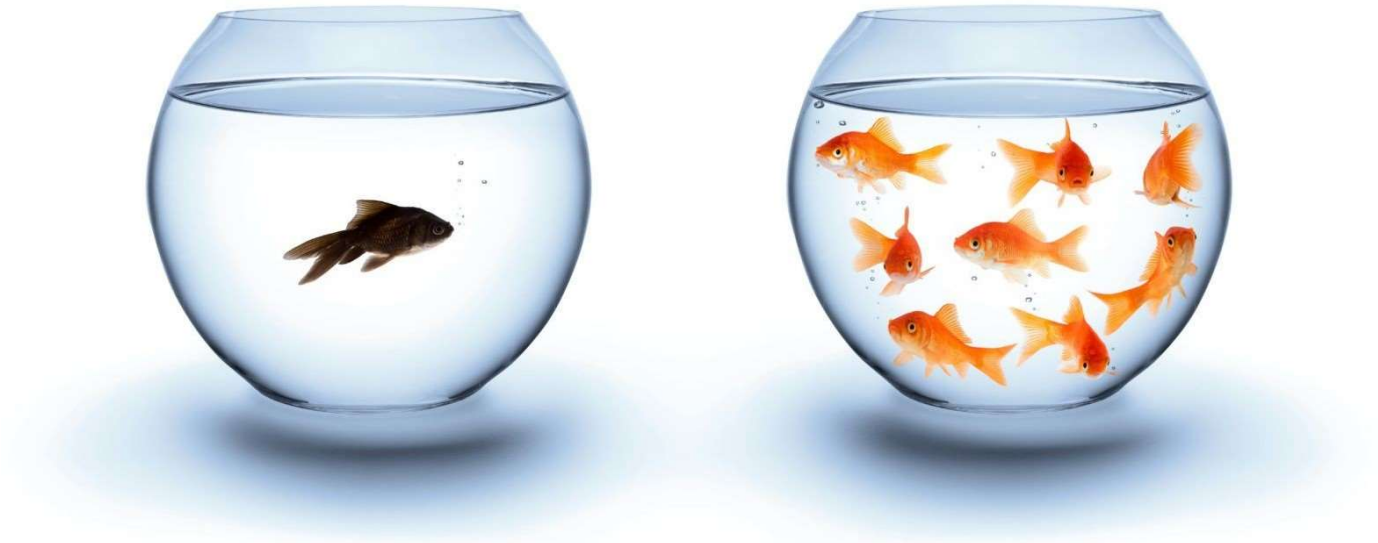


Many ephemeral clusters



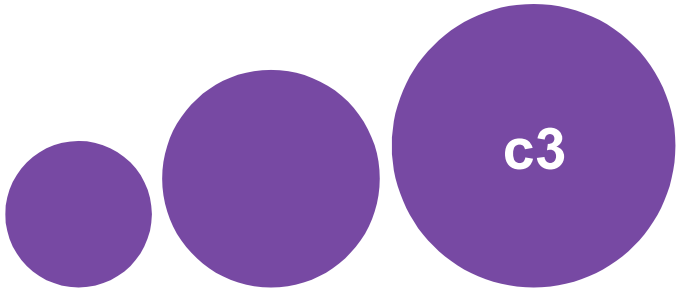
- New cluster for every pipeline.
- Dozens of clusters at a time.
- Median lifetime of ~3 hours.

Total isolation

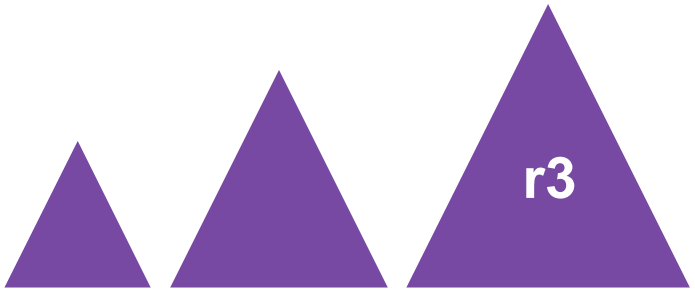


We know what is happening and why.

Pick the best hardware for each job

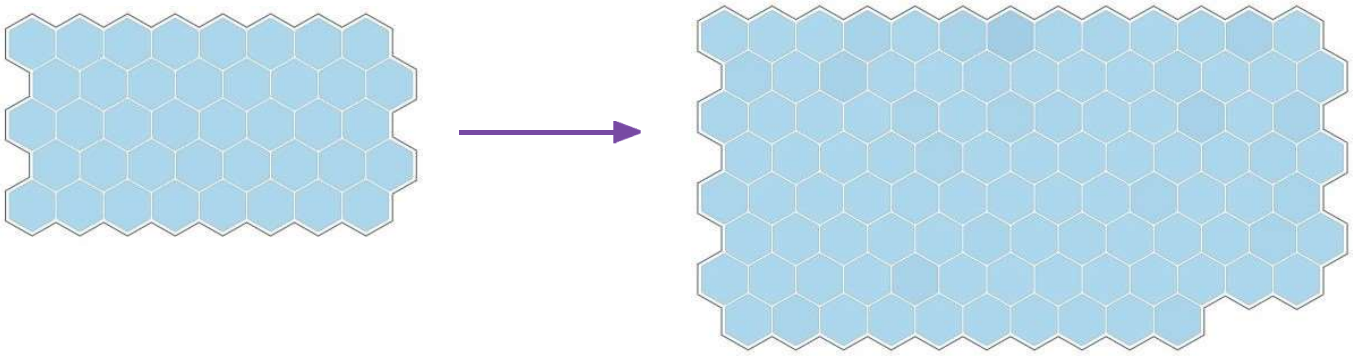


For CPU-bound jobs



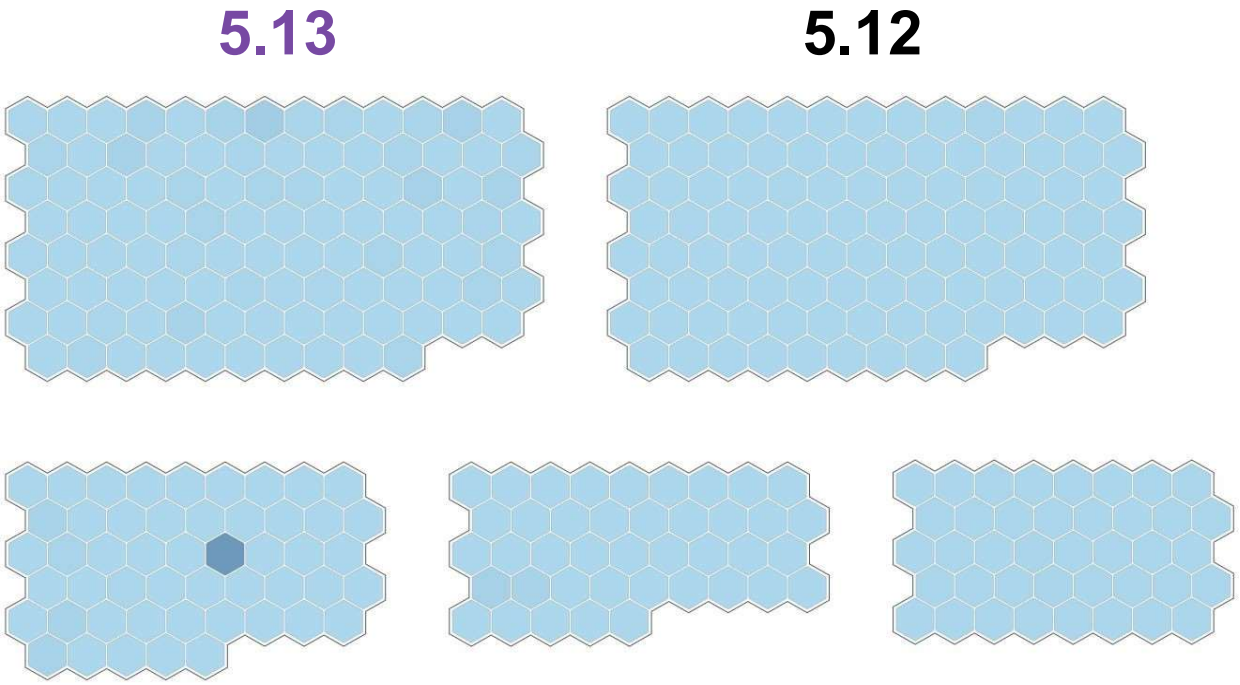
For memory-bound jobs

Scale up/down clusters



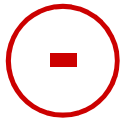
- If we are behind.
- Scale as we grow.
- No more waiting on loaded clusters.

Safer upgrades of EMR/Hadoop/Spark



Spot-instance clusters

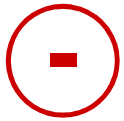
Ridiculous savings
(up to 80% off the on-demand price)



Nodes can die at any time

Spot-instance clusters

Ridiculous savings
(up to 80% off the on-demand price)



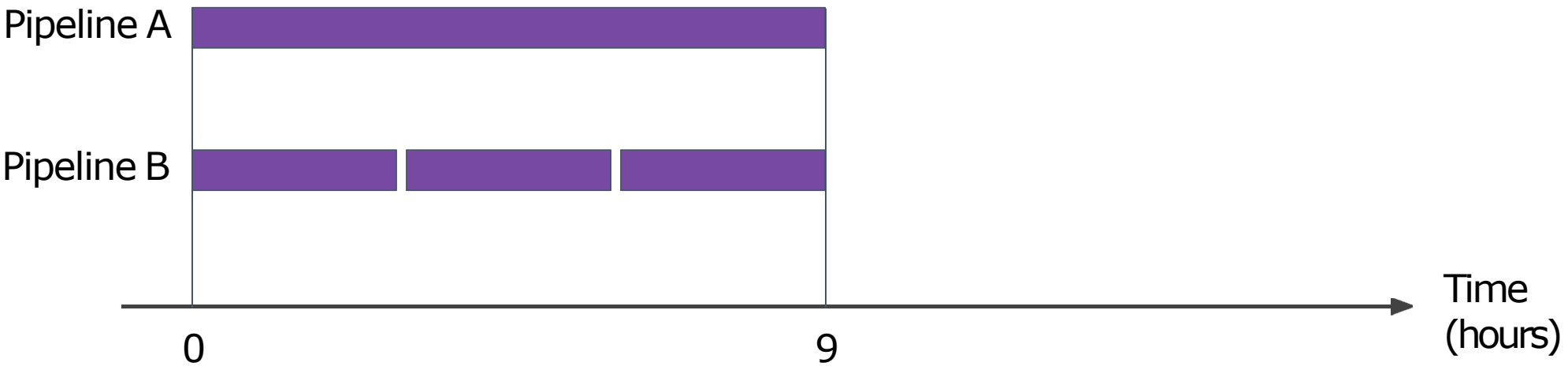
Nodes can die at any time

How can we build **highly reliable** data pipelines with **instances killed randomly** all the time?

No long running jobs

- The longer the job, the more work you lose on average.
- The longer the job, the longer it takes to recover.

No long running jobs



No long running jobs

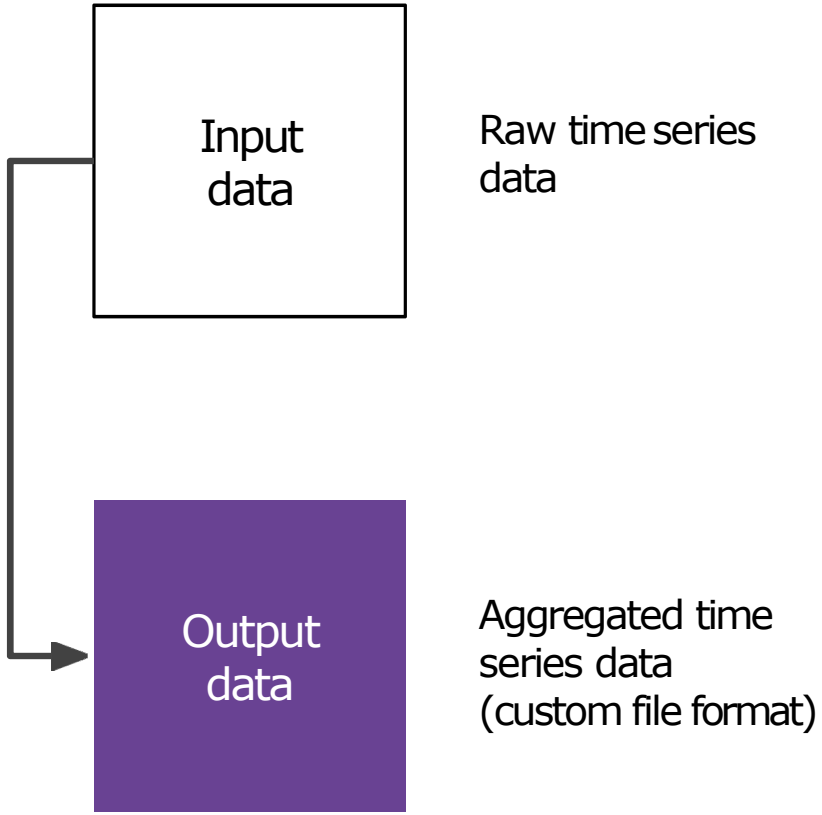
Break down jobs into smaller pieces

Vertically - persist intermediate data between transformations.

Horizontally - partition the input data.

Example

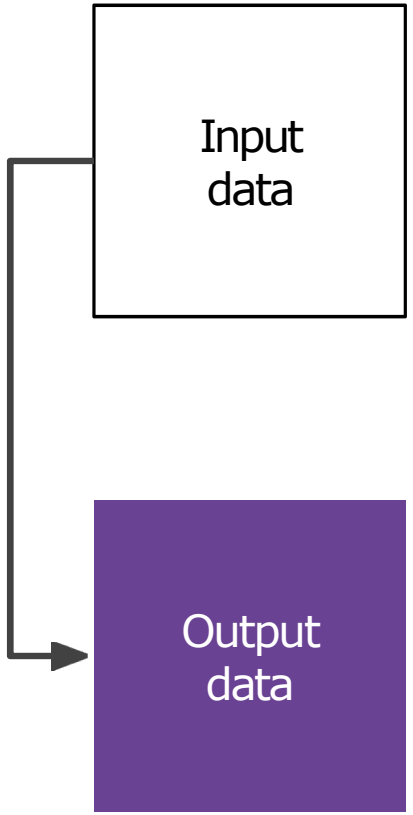
Rollups pipeline



Example

Rollups pipeline

1. **Aggregate** high resolution data.
2. **Store** the aggregated data in our custom file format.

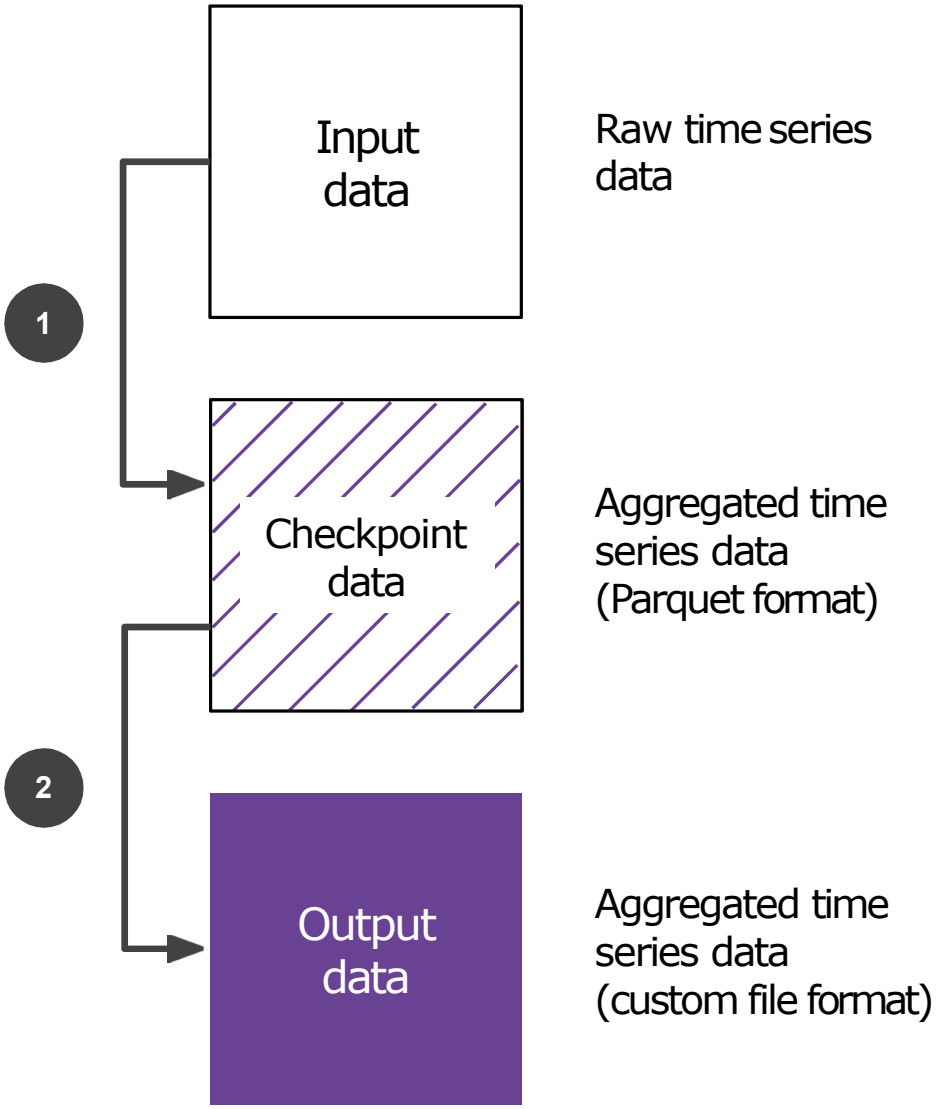


Raw time series data

Aggregated time series data (custom file format)

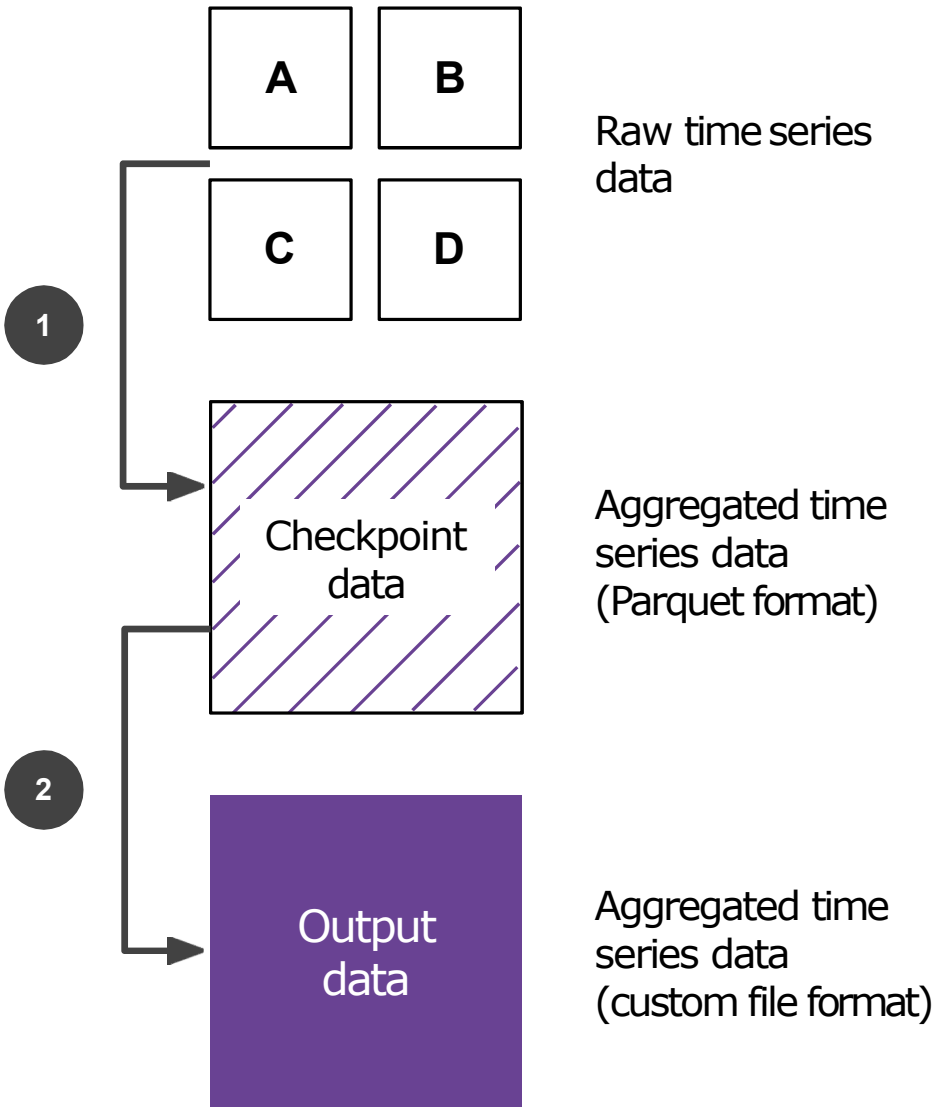
Example
Vertical split

1. **Aggregate** high resolution data.
2. **Store** the aggregated data in our custom file format.



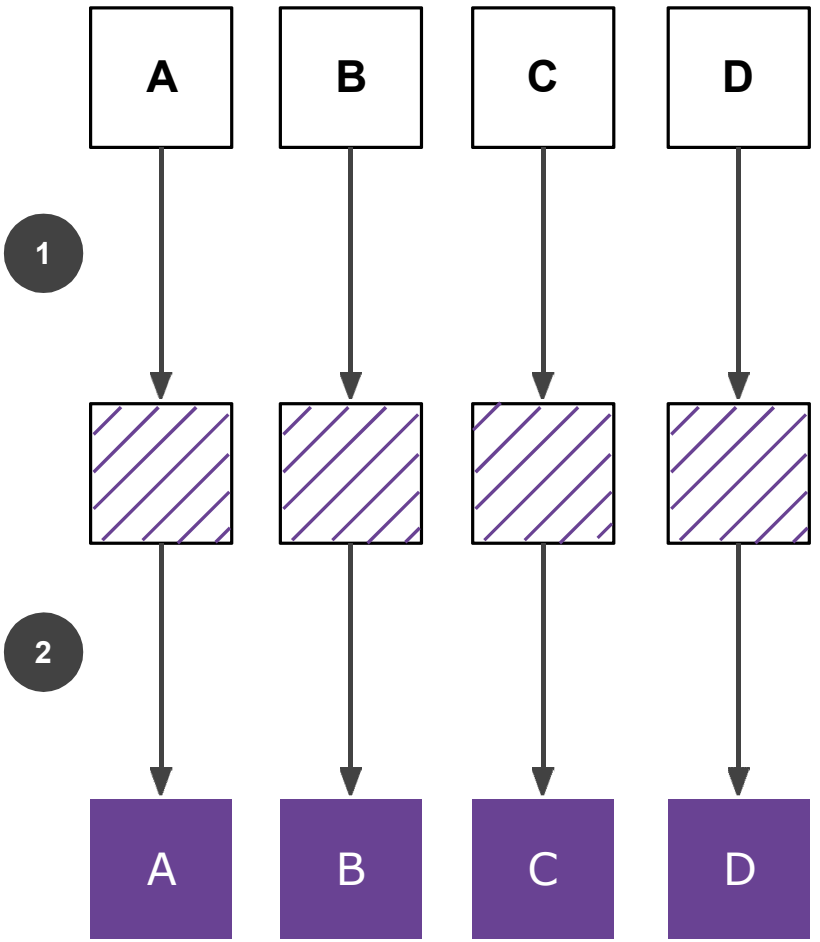
Example
Horizontal split

1. **Aggregate** high resolution data.
2. **Store** the aggregated data in our custom file format.

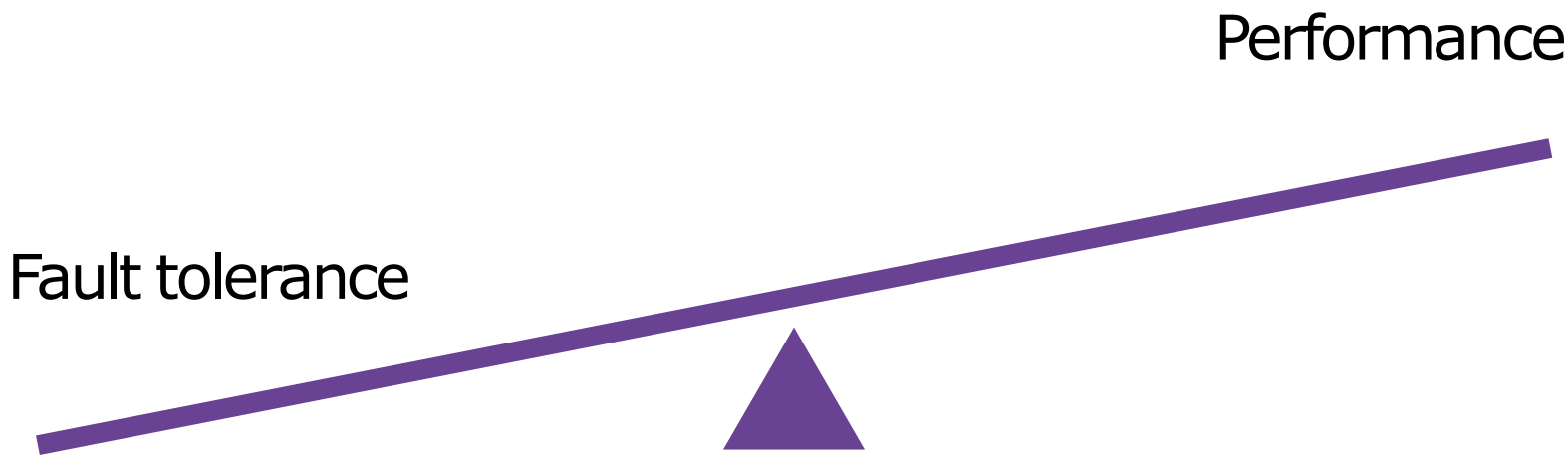


Example
Horizontal split

1. **Aggregate** high resolution data.
2. **Store** the aggregated data in our custom file format.



Break down jobs into smaller pieces



Lessons

- Many clusters for better isolation.
- Break down jobs into smaller pieces (no longer than ~3 hours).
- Trade-off between performance and fault tolerance.

Highly reliable data pipelines

1. Architecture
2. Monitoring
3. Failures handling

Reliability is the probability that a system will produce correct outputs **up to some given time t**.

Reliability is the probability that a system will produce **correct outputs** up to some given time t.

Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?

We monitor actively 3 types of metrics:

- Data lags metrics.
- Cluster health metrics.
- Job health metrics.

Monitoring data pipelines

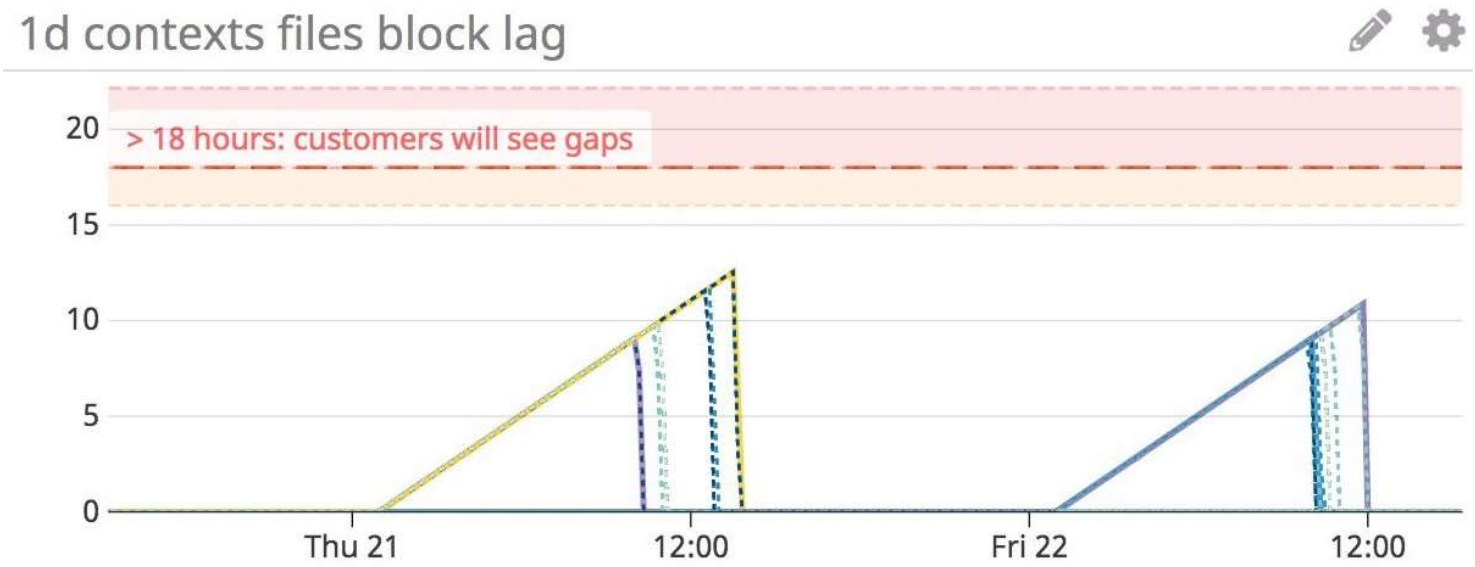
1. Is the data pipeline going to finish before the deadline?

We monitor actively 3 types of metrics:

- **Data lags metrics.**
- Cluster health metrics.
- Job health metrics.

Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?



Monitoring data pipelines

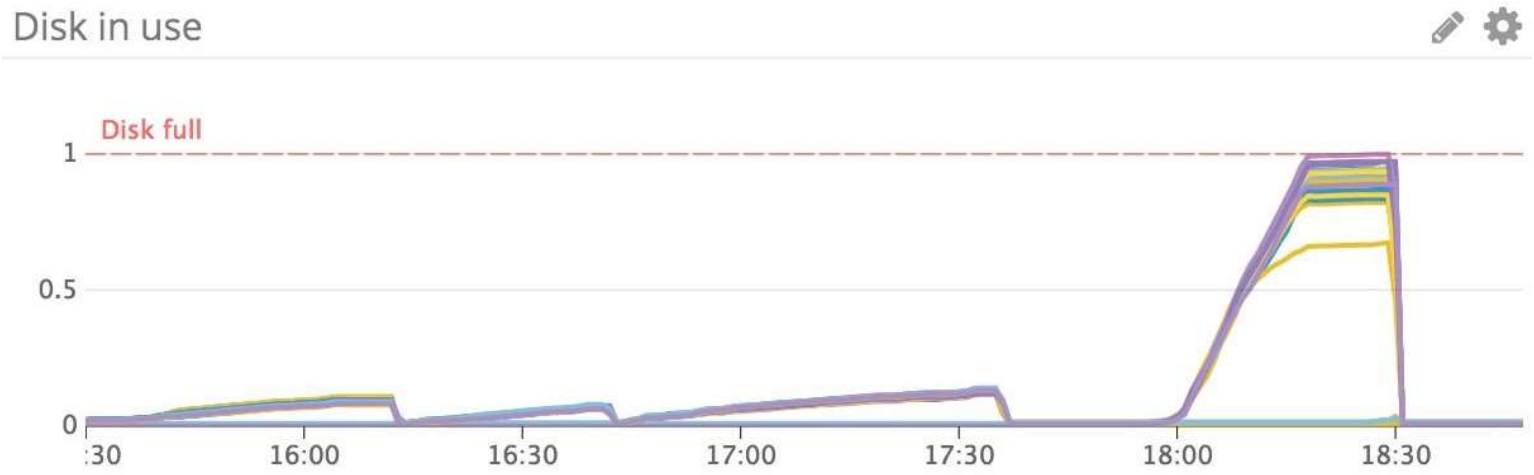
1. Is the data pipeline going to finish before the deadline?

We monitor actively 3 types of metrics:

- Data lags metrics.
- **Cluster health metrics.**
- Job health metrics.

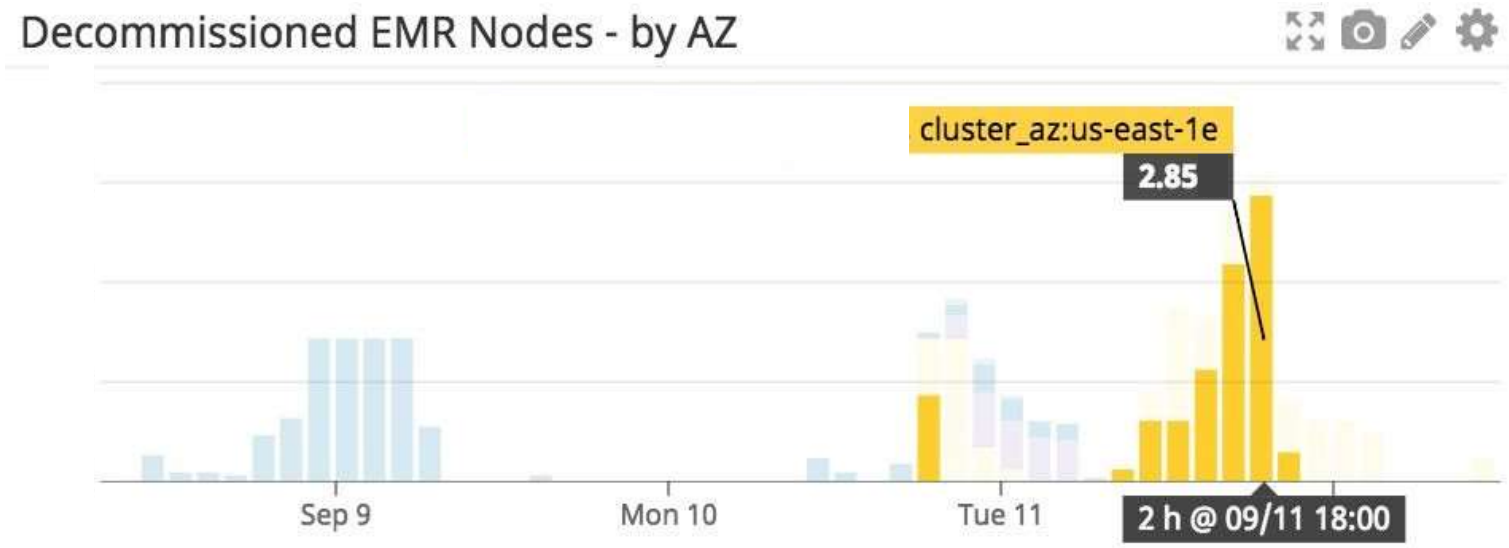
Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?



Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?



Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?

We monitor actively 3 types of metrics:

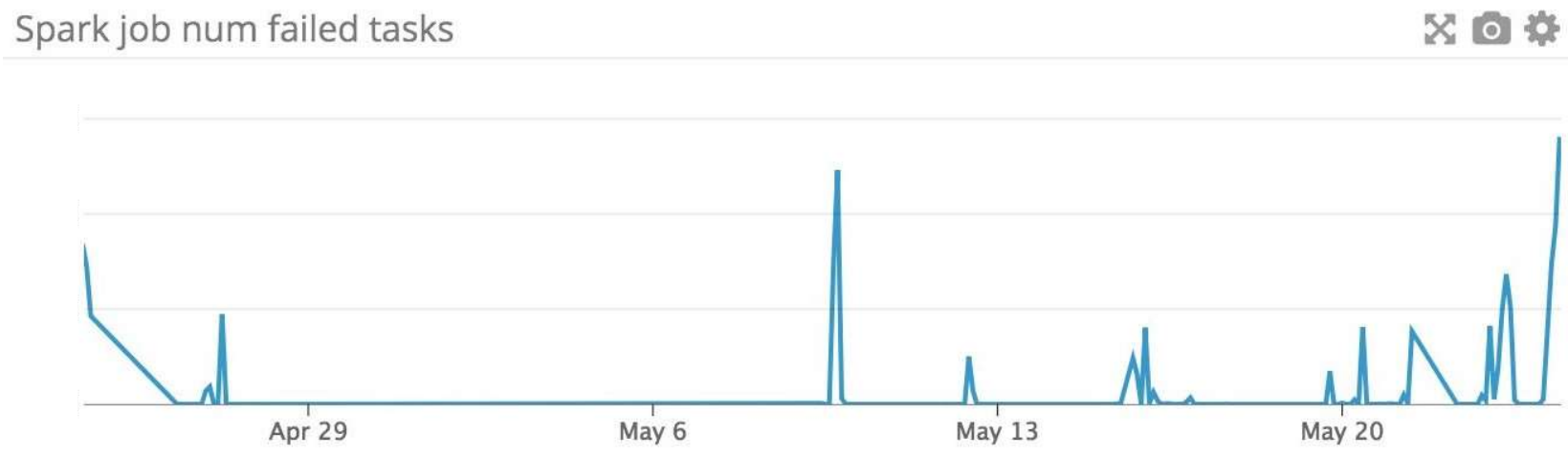
- Data lags metrics.
- Cluster health metrics.
- **Job health metrics.**

Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?

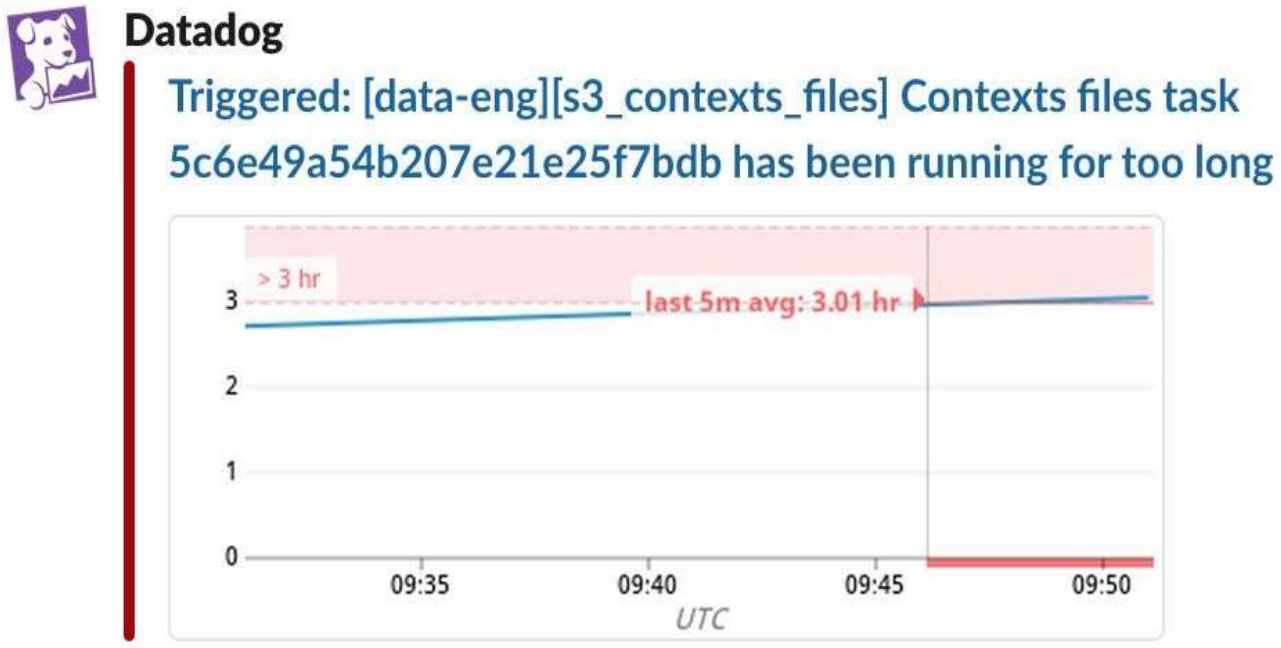
Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?



Monitoring data pipelines

1. Is the data pipeline going to finish before the deadline?



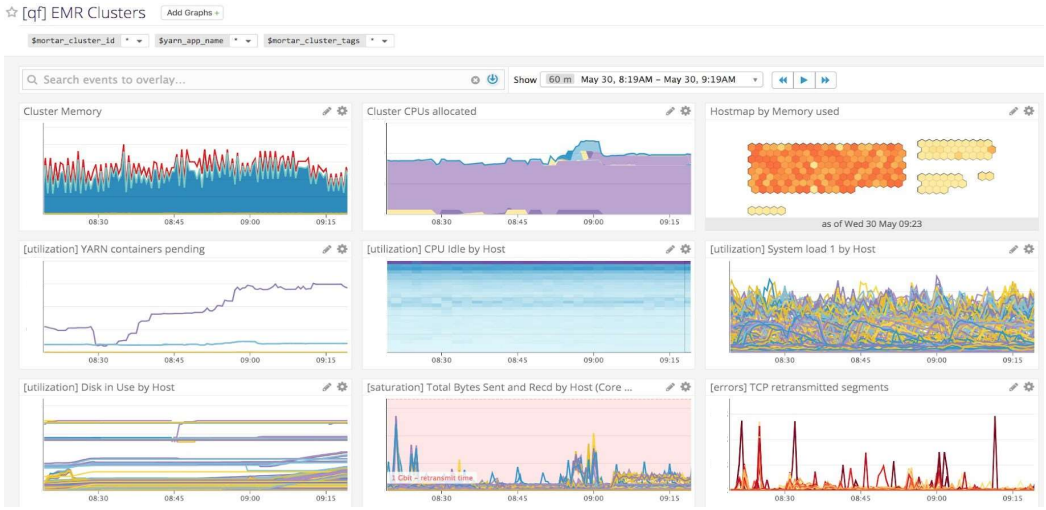
Monitoring data pipelines

2. Is the data produced correct?

- Add custom counters throughout the pipelines.
  - Count records.
  - Count duplicates.
  - Count records that can't join.
- Ad hoc checks on the output data.

Lessons

- Monitoring =will we finish before t? +is the data correct?
- Measure, measure and measure!
- Alert on meaningful and actionable metrics.

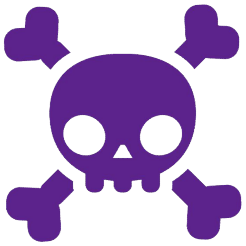


Highly reliable data pipelines

1. Architecture
2. Monitoring
3. Failures handling


BRACE YOURSELF
FAILURES ARE COMING

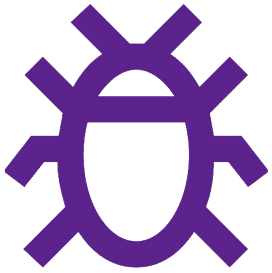
Data pipelines will break



Hardware failures

Increasing volume of data

Upstream delays



Bad code changes

Data pipelines will break

1. **Recover fast**

   We want to fix the issues ASAP.

2. **Degrade gracefully**

   We want to limit the customer-facing impact.

Recover fast

- No long running job.
- Switch from spot to on-demand clusters.
- Increase cluster size.
- Easy ways to rerun jobs (not always trivial!).

Example: rerun the rollups pipeline

s3://bucket/

- 2018-01
- 2018-02
- 2018-03
- 2018-04
- 2018-05

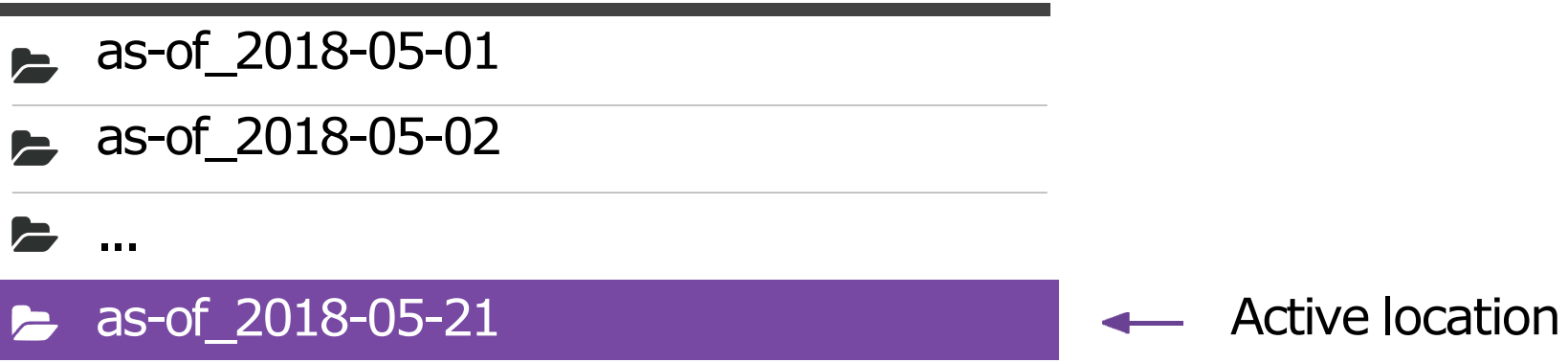
Example: rerun the rollups pipeline

s3://bucket/2018-05/

- as-of_2018-05-01
- as-of_2018-05-02
- ...
- as-of_2018-05-21

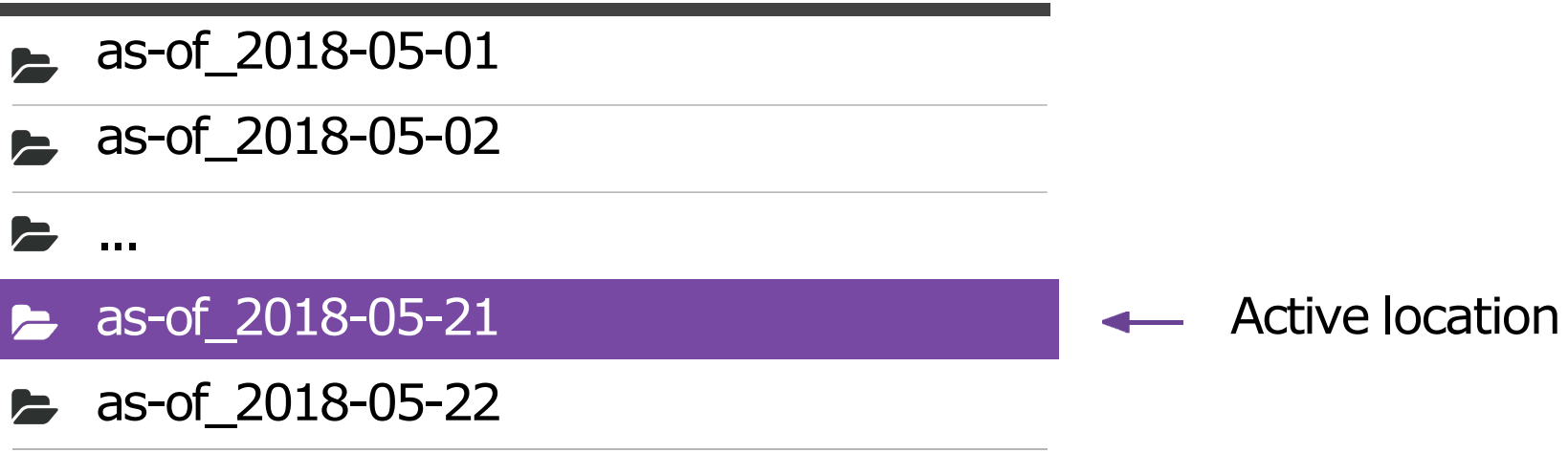
Example: rerun the rollups pipeline

s3://bucket/2018-05/



Example: rerun the rollups pipeline

s3://bucket/2018-05/

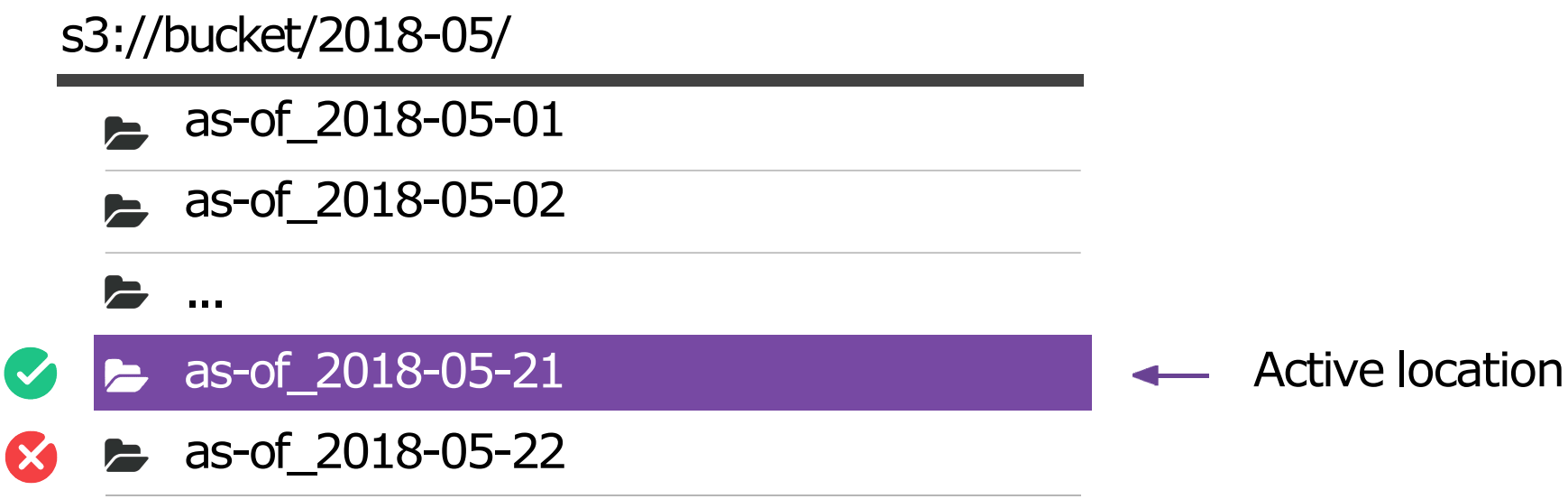


Example: rerun the rollups pipeline

s3://bucket/2018-05/

- as-of_2018-05-01
- as-of_2018-05-02
- ...
- as-of_2018-05-21
- as-of_2018-05-22 ← Active location

Example: rerun the rollups pipeline



Example: rerun the rollups pipeline

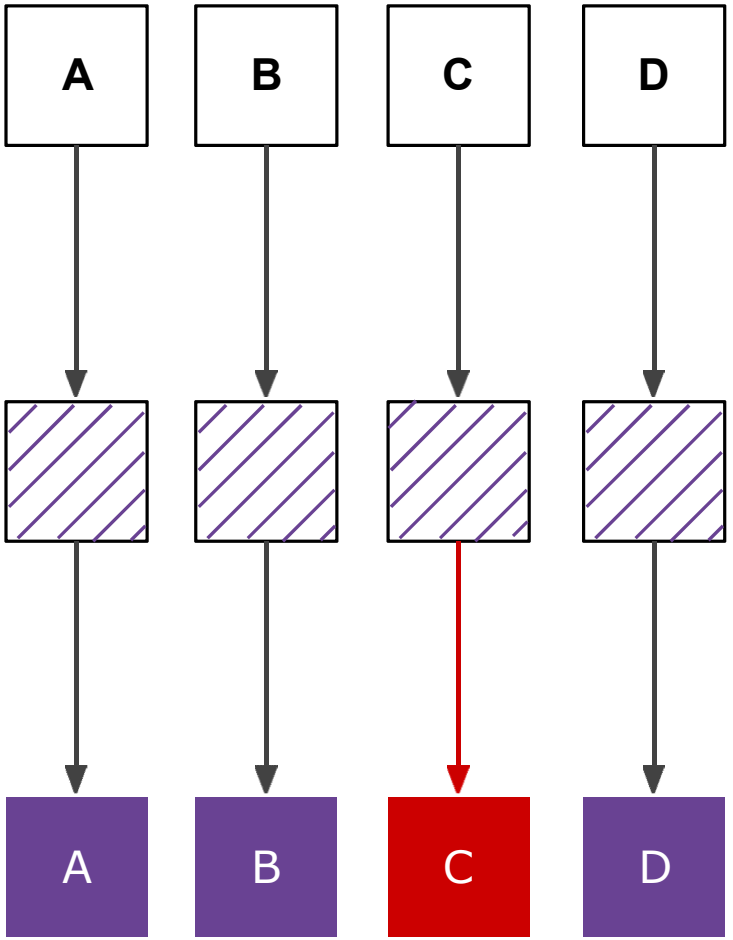
s3://bucket/2018-05/

- as-of_2018-05-01
- as-of_2018-05-02
- ...
- as-of_2018-05-21
- as-of_2018-05-22
- as-of_2018-05-22_run-2 ← Active location

Degrade gracefully

- Isolate issues to a limited number of customers thanks to horizontal sharding.

- Keep the functionalities operational at the cost of performance/accuracy.



Lessons

- Think about potential issues ahead of time.
- Have knobs ready to recover fast.
- Have knobs ready to limit the customer facing impact.

Conclusion

Building highly reliable data pipelines

Conclusion

Building highly reliable data pipelines

- Know your time constraints.

Conclusion

Building highly reliable data pipelines

- Know your time constraints.
- Break down jobs into small survivable pieces.

Conclusion

Building highly reliable data pipelines

- Know your time constraints.
- Break down jobs into small survivable pieces.
- Monitor cluster metrics, job metrics and data lags.

Conclusion

Building highly reliable data pipelines

- Know your time constraints.
- Break down jobs into small survivable pieces.
- Monitor cluster metrics, job metrics and data lags.
- Think about failures ahead of time and get prepared.

Thanks !

We’re hiring!

qf@datadoghq.com
https://jobs.datadoghq.com