Resilient Predictive Data Pipelines

Sid Anand (@r39132)

GOTO Chicago 2016

About Me

Work [ed | s] @

Co-Chair for

Committer &
PPMC on

Report to

Motivation

Why is a Data Pipeline talk in this Always Available Track?

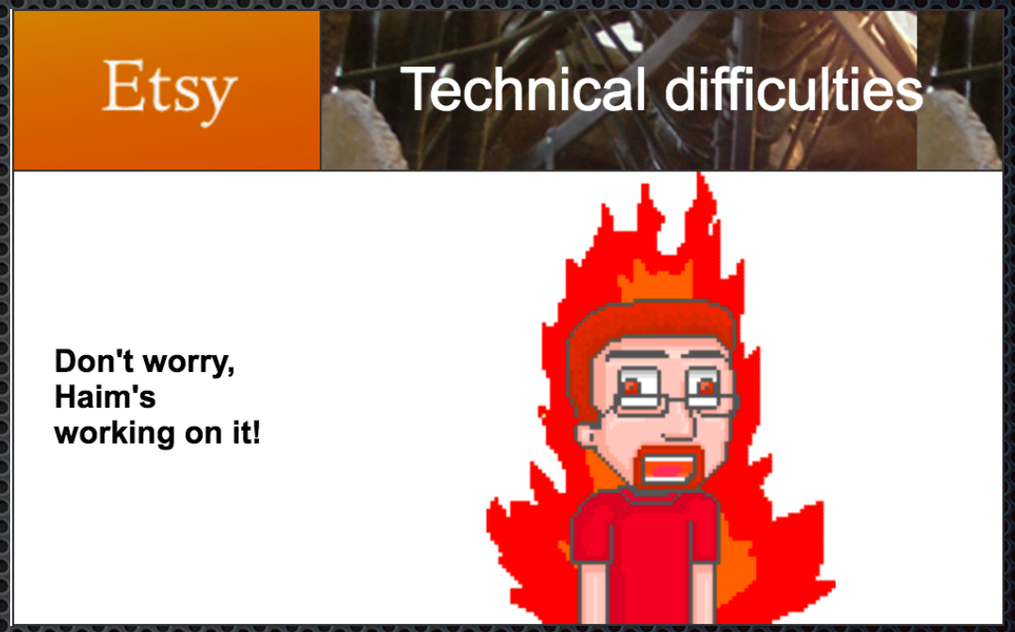
Motivation

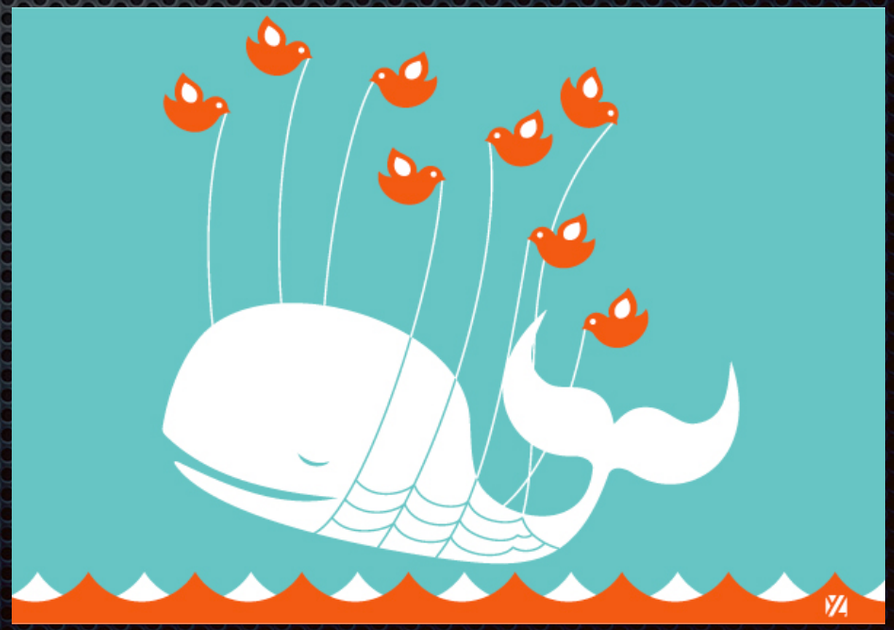
Always On work has traditionally focused on the availability of **Serving Systems** :

- Synchronous or Semi-Synchronous
- Often Transactional
- Latency-sensitive

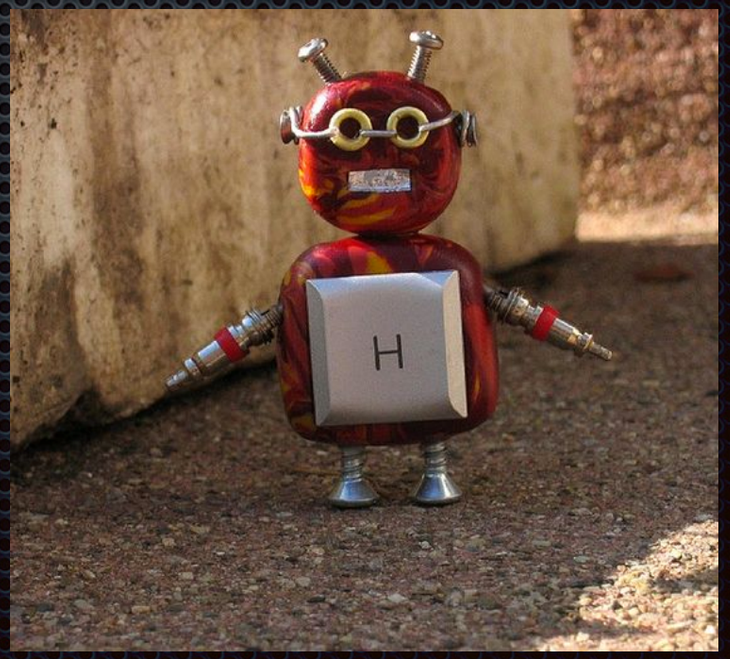
HA Goals of Serving Systems

Outages are Big News Items!





And sometimes your failures become your brand!



Motivation

Always On work has traditionally focused on the availability of **Serving Systems** :

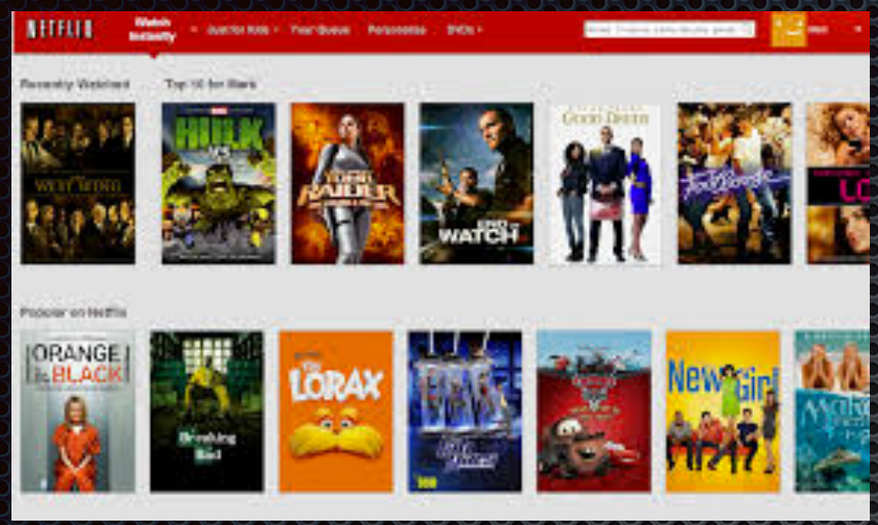
- Synchronous or Semi-Synchronous
- Often Transactional
- Latency-sensitive

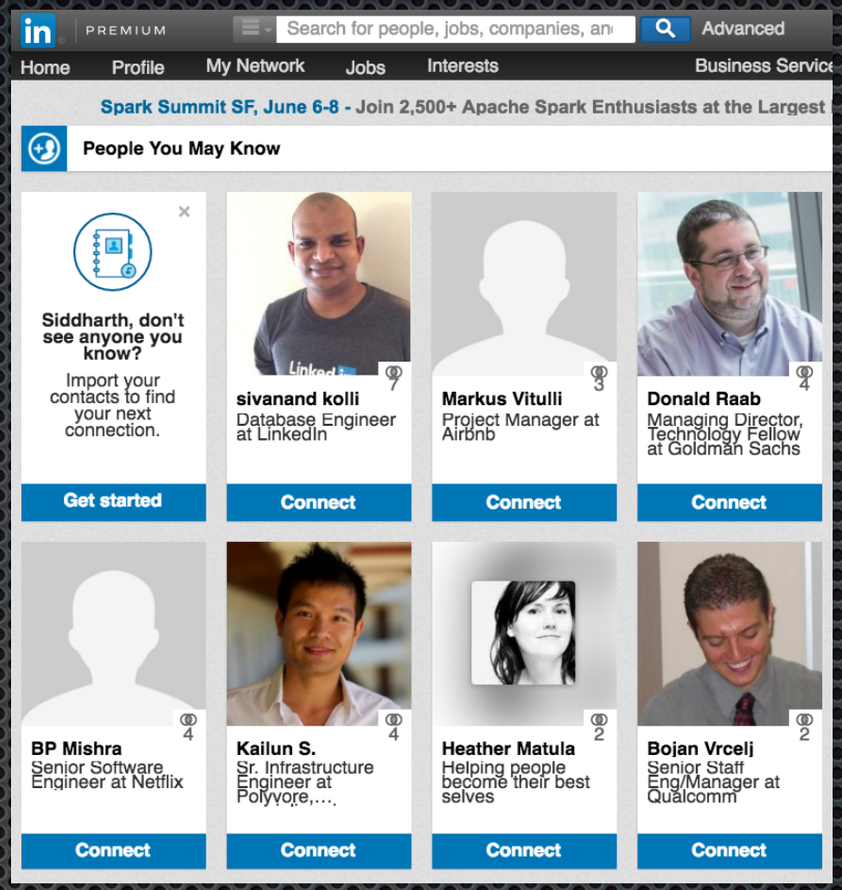
Motivation

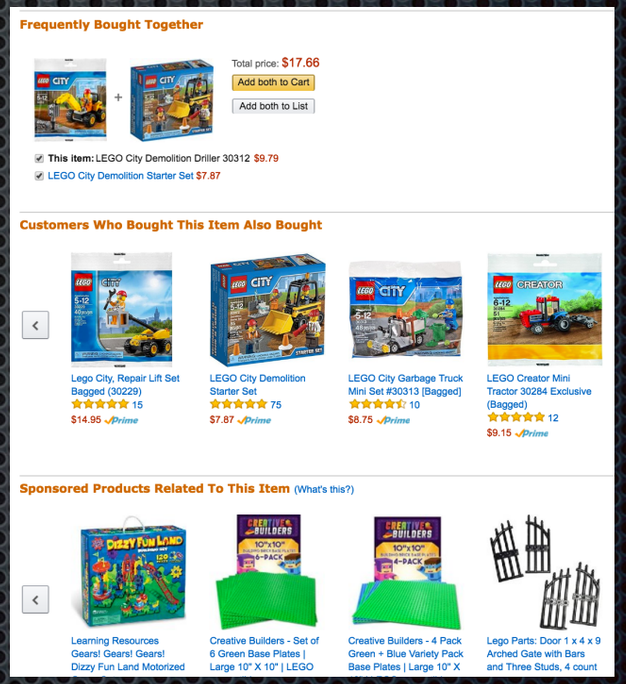
Arguably, the more valuable parts of online services are driven by **Data Flow Systems** (**a.k.a. Data Pipelines**):

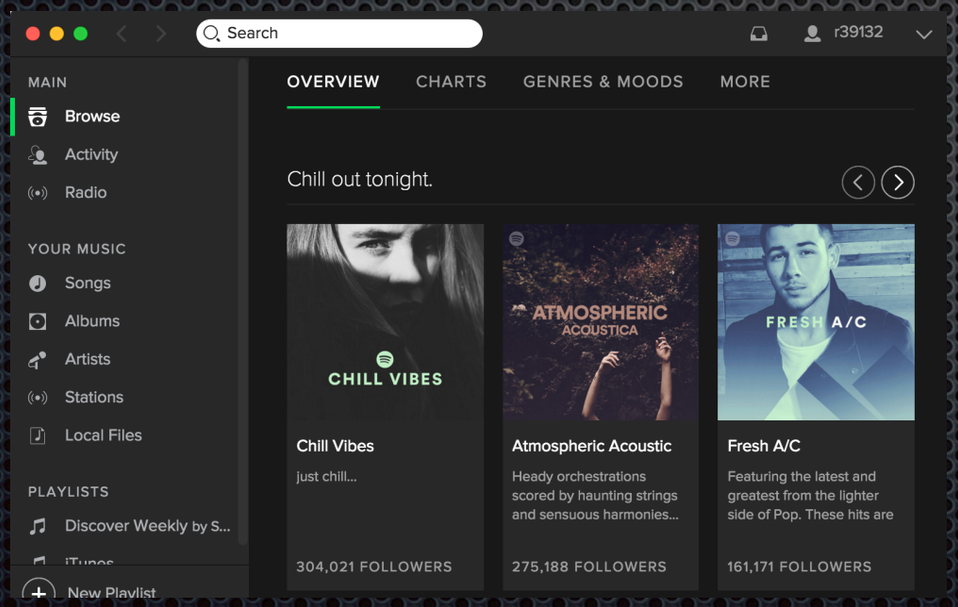
- Asynchronous
- Throughput-sensitive

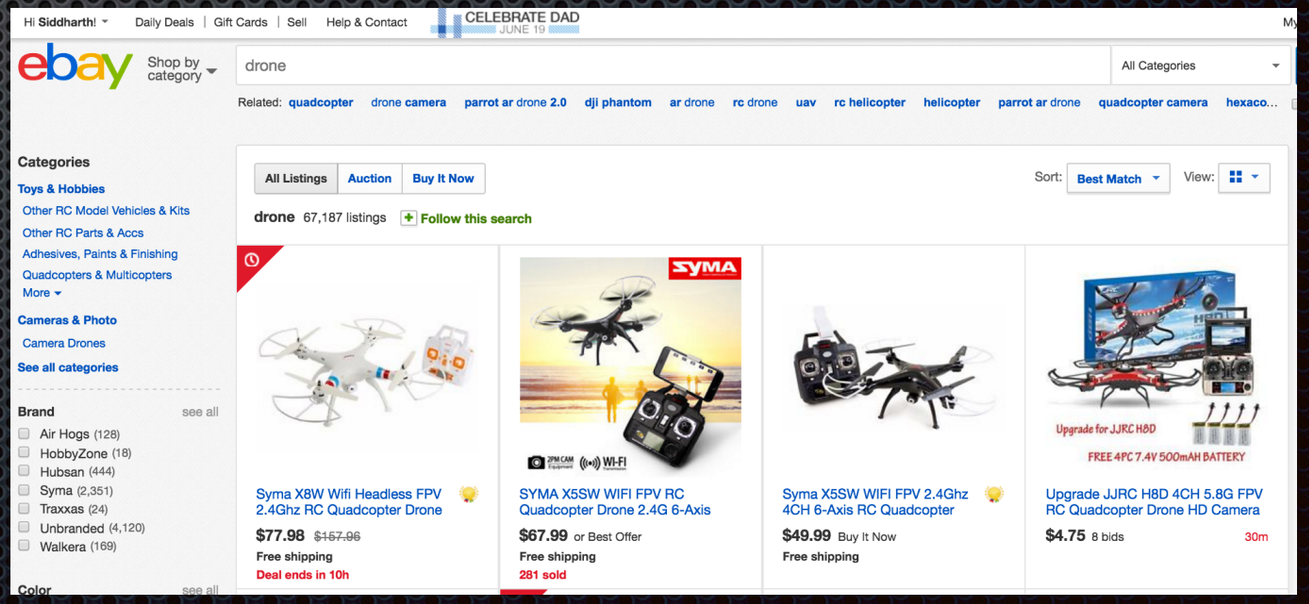
Data Products











Serving + Data Pipelines

A business's viability is based on its ability to

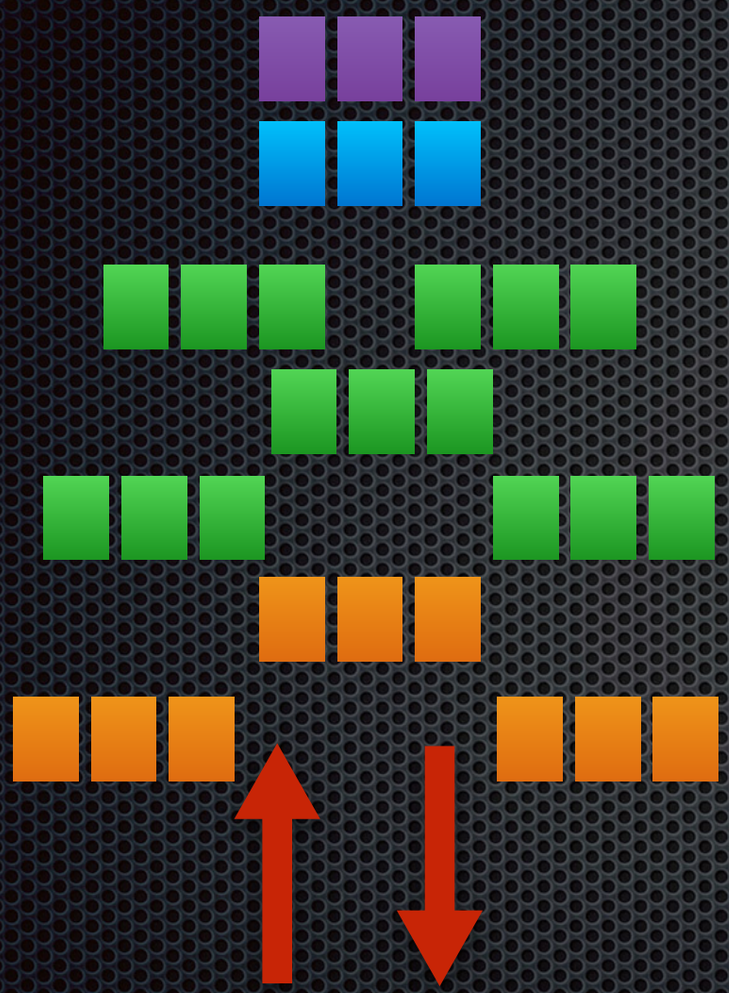
- keep the site up (Always-On Serving Architectures) &
- maintain engagement (views & clicks) with customers (Always-On Data Pipelines)

This talk is about Always On Data Pipelines!

Serving + Data Pipelines

Serving

Data Pipeline



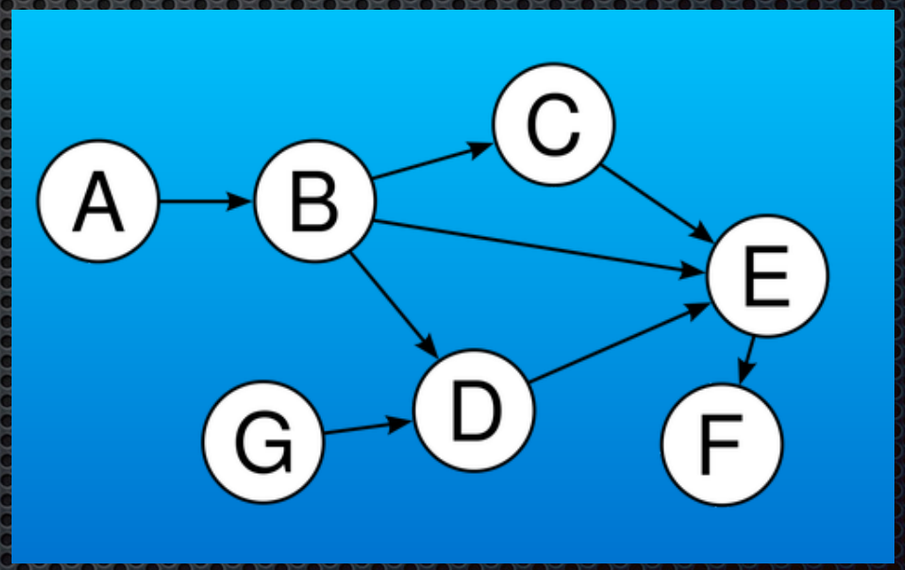
FE Load Balancers

Web Servers

Microservice Layer

Data Layer
(DB, Search, Caching, Graph DB, Object Store)

DAGs + Scheduler + Distributed Computation Engine

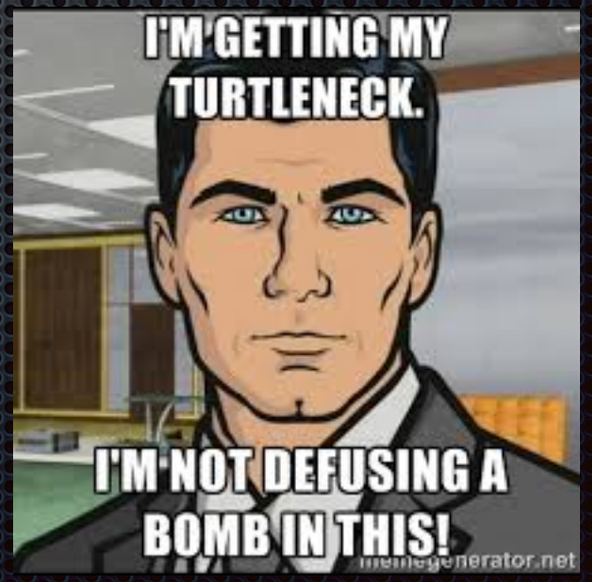


Data Integration Layer

Data Pipeline Challenges

Data Pipeline Challenges

Problem 1 : The Blast Radius Problem



The Blast Radius Problem

- A developer introduces a bug in Data Pipeline **Job 1**
- Data Pipeline **Job 1** reads **Data A** & writes **Data B**

The Blast Radius Problem

- A developer introduces a bug in Data Pipeline **Job 1**
- Data Pipeline **Job 1** reads **Data A** & writes **Data B**
- Data Pipeline **Job 2** reads **Data B** & writes **Data C**

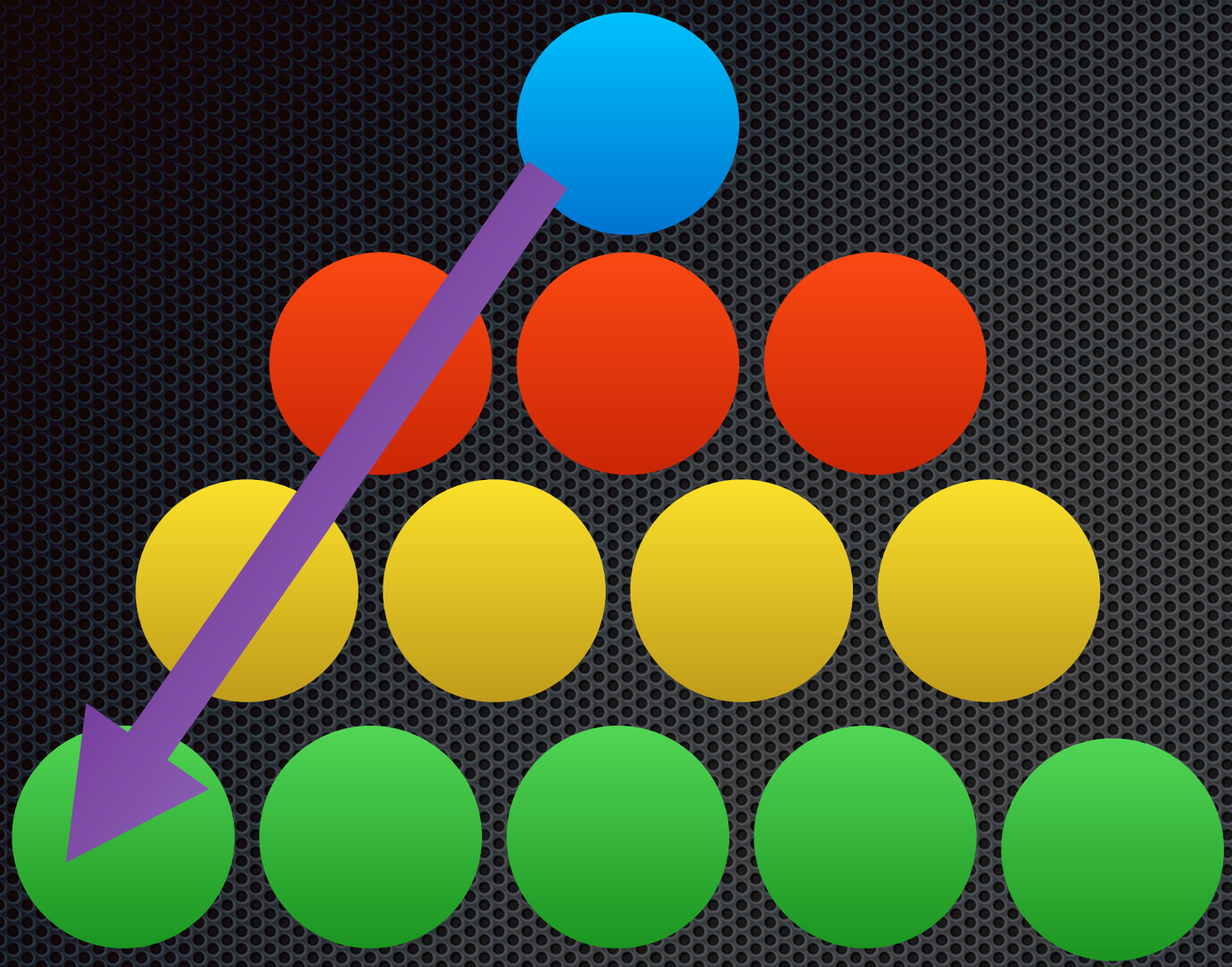
The Blast Radius Problem

- A developer introduces a bug in Data Pipeline **Job 1**
- Data Pipeline **Job 1** reads **Data A** & writes **Data B**
- Data Pipeline **Job 2** reads **Data B** & writes **Data C**
- Data Pipeline **Job 3** reads **Data C** & writes **Data D** to a Serving System DB

The Blast Radius Problem

- A developer introduces a bug in Data Pipeline **Job 1**
- Data Pipeline **Job 1** reads **Data A** & writes **Data B**
- Data Pipeline **Job 2** reads **Data B** & writes **Data C**
- Data Pipeline **Job 3** reads **Data C** & writes **Data D** to a Serving System DB
- **Serving System 4** reads **Data D, where the bug is discovered!**

The Blast Radius Problem

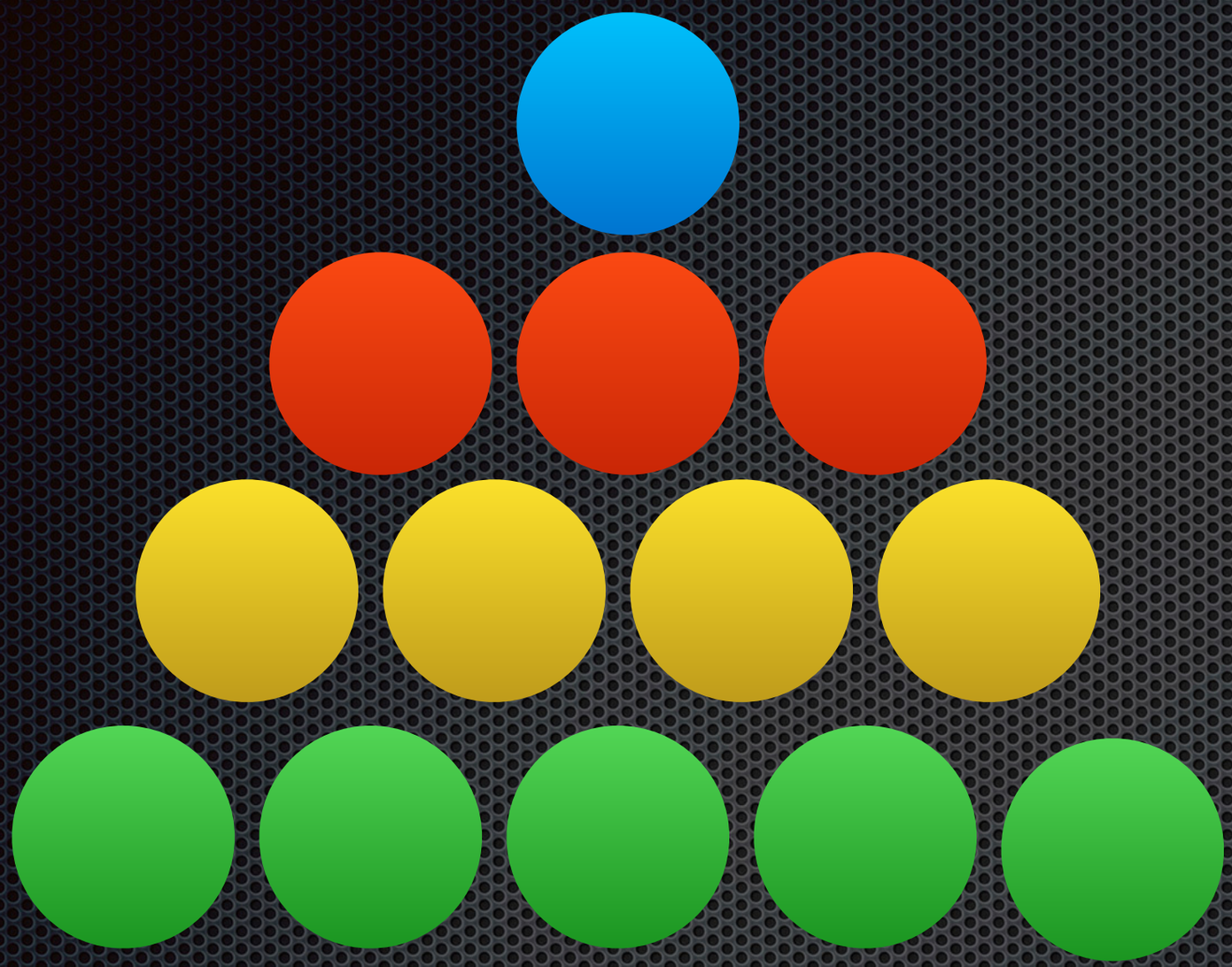


- The previous diagram only shows one path of a tree
- The reality is much worse
- For each data set produced, there are multiple consuming jobs and hence multiple bad downstream outputs

The Blast Radius Problem

An acute pain point

The Blast Radius Problem



Job 1

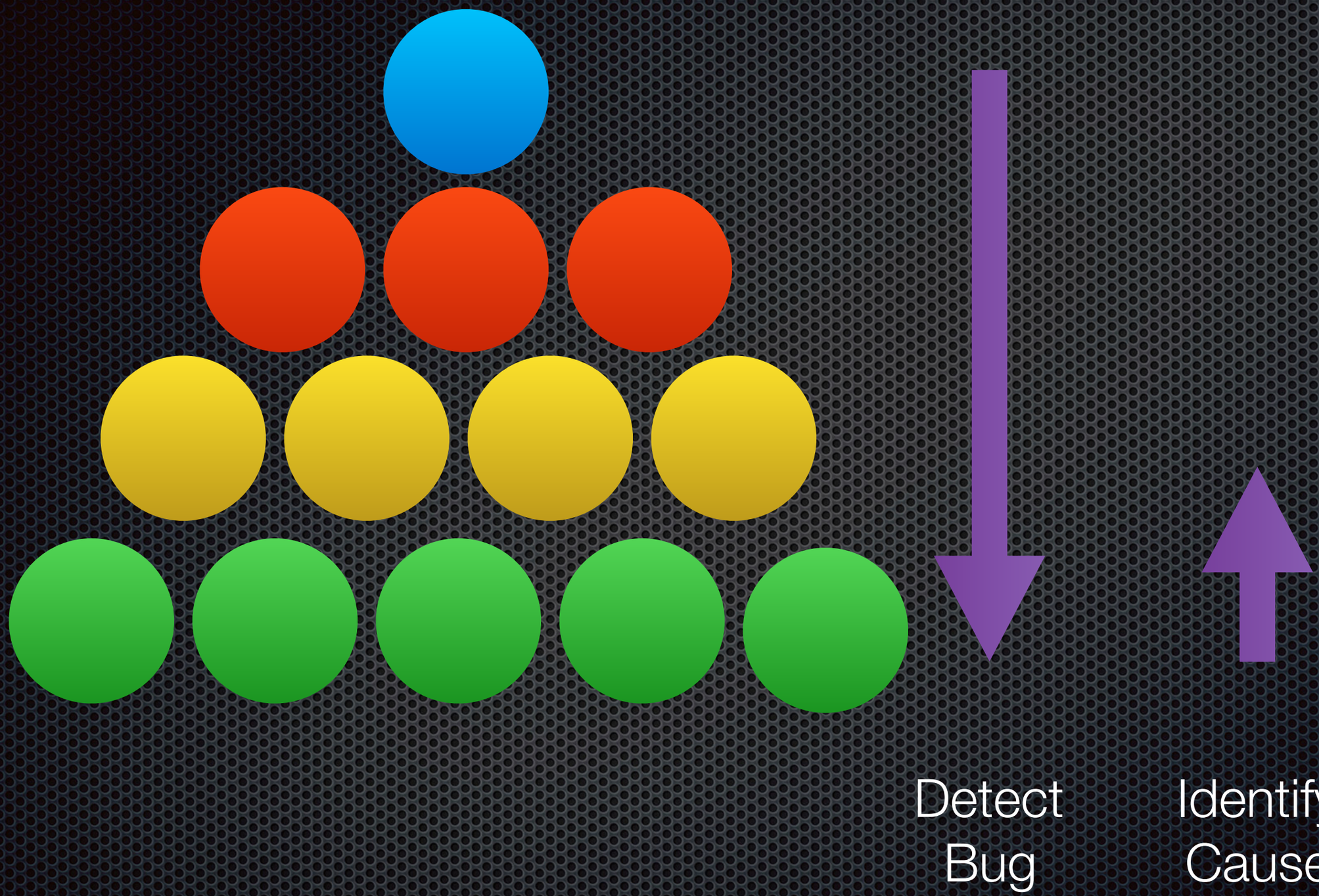
Job 2

Job 3

Serving System

Detect
Bug

The Blast Radius Problem



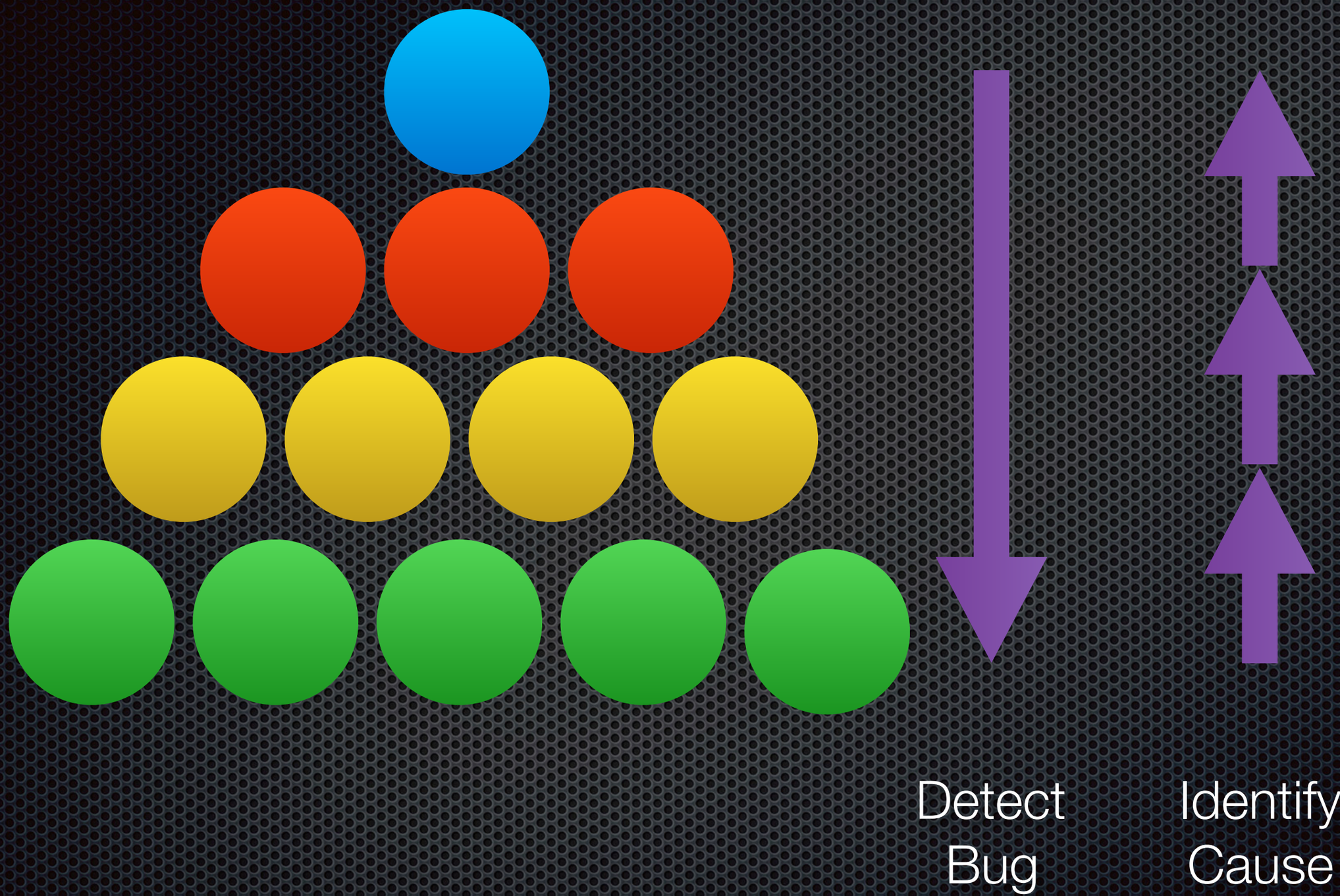
Identify Cause



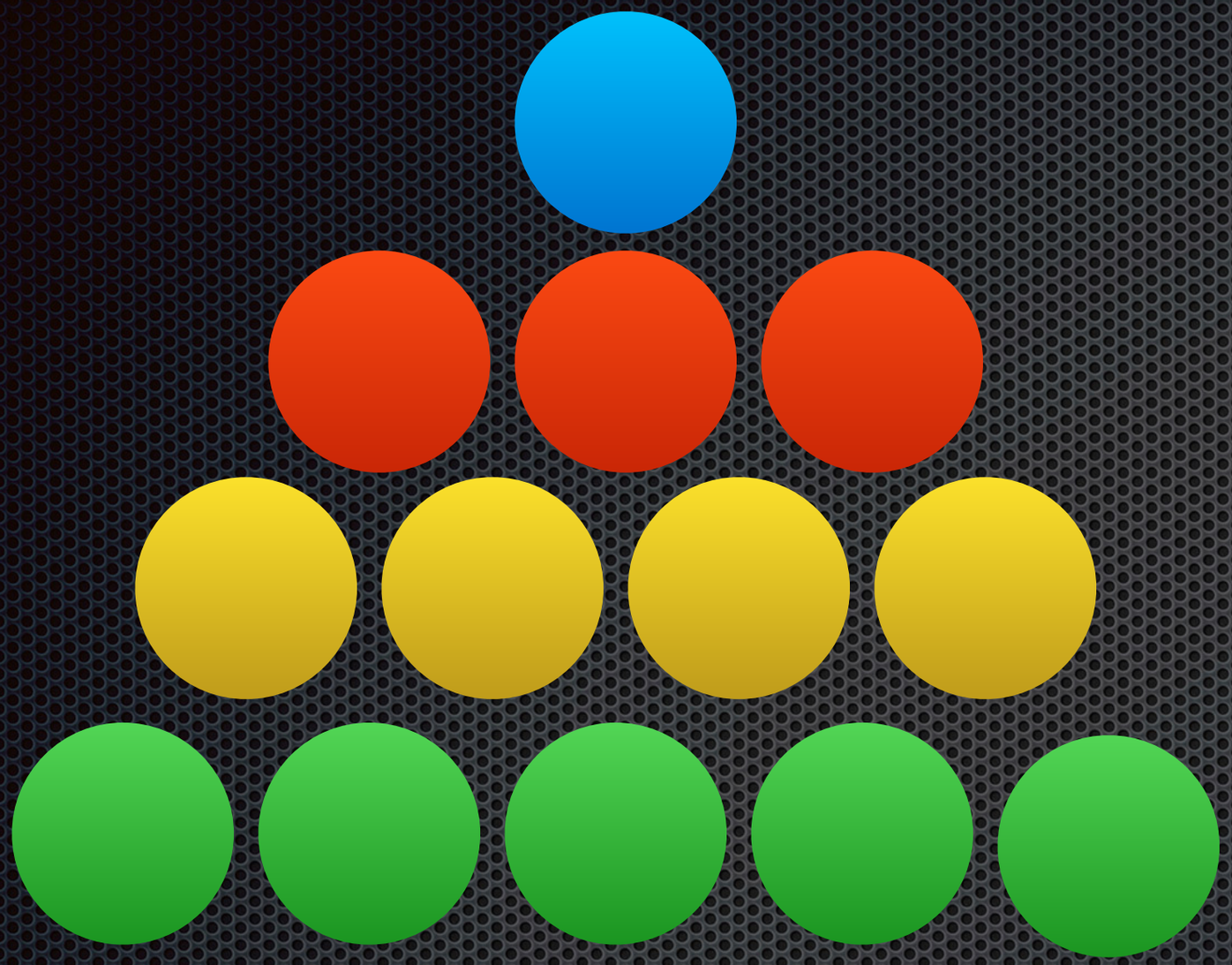





The Blast Radius Problem



The Blast Radius Problem

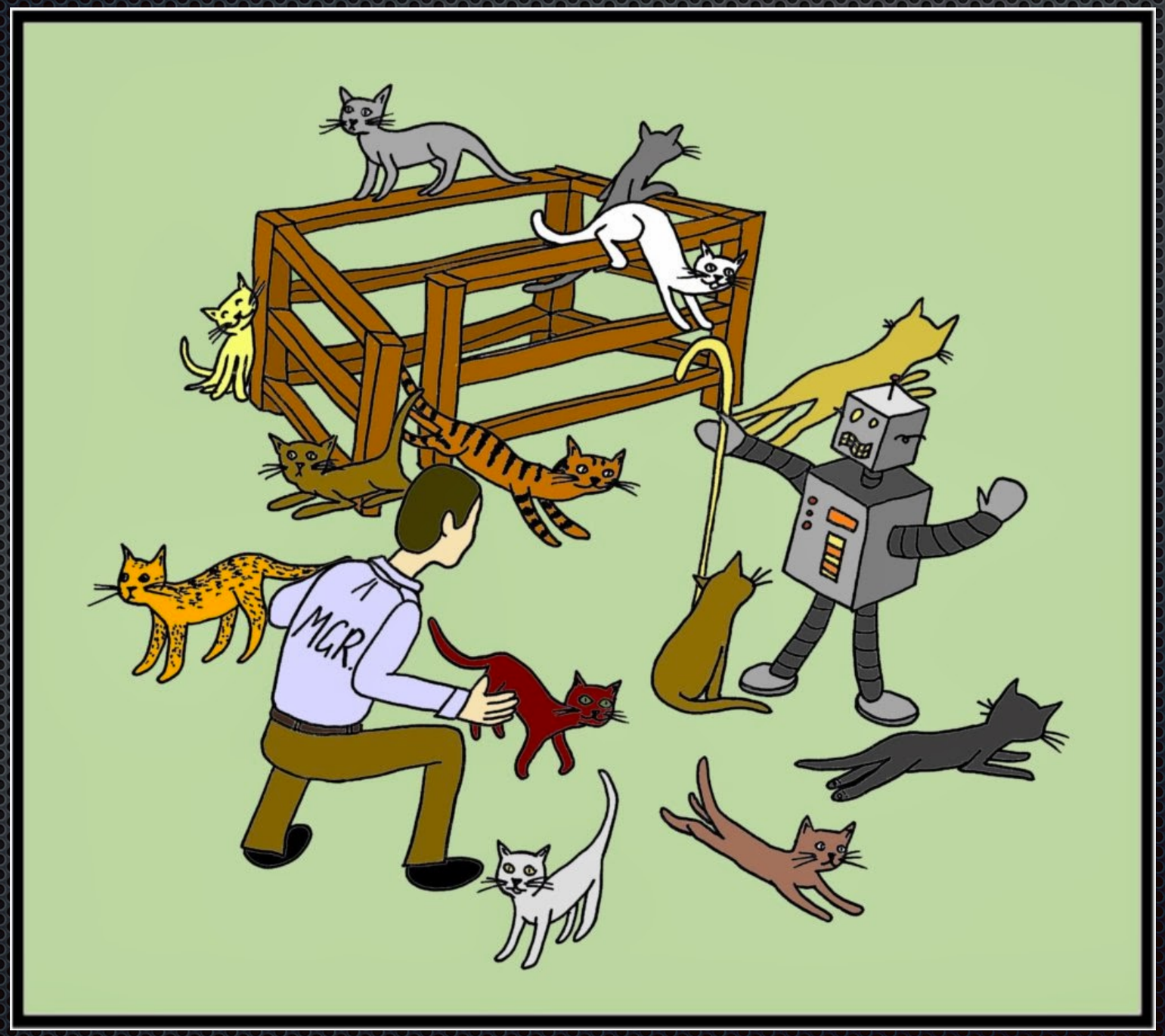


Detect Bug

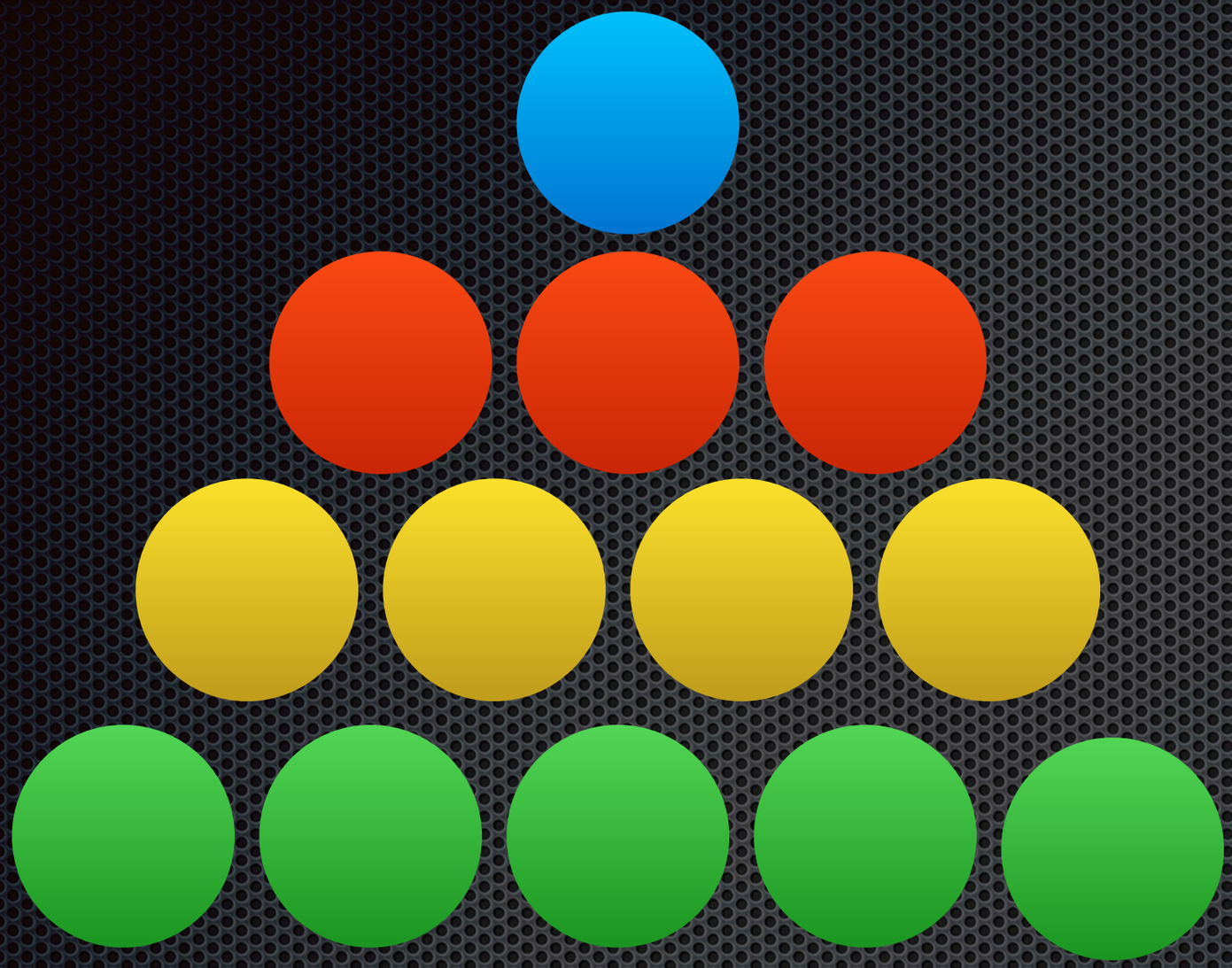
Identify Cause

Deploy a Fix

Rollout a Fix & Rerun all Downstream Jobs in the Affected Time Window



The Blast Radius Problem



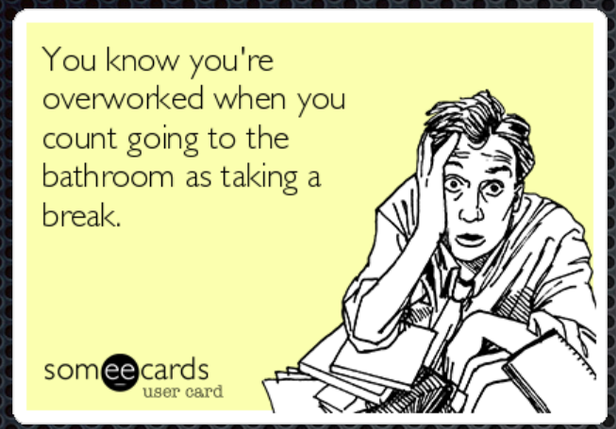
Detect Bug

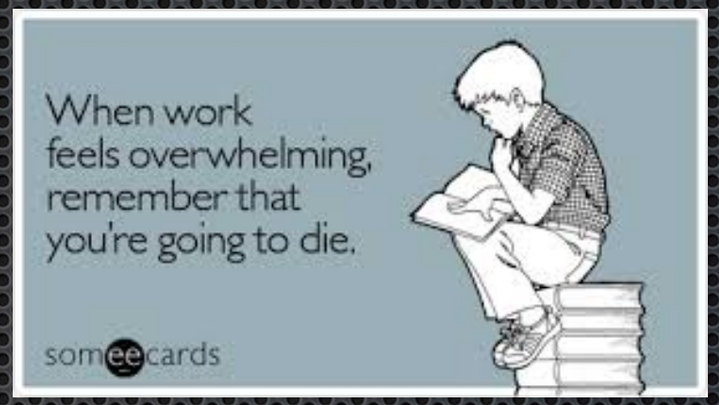
Identify Cause

Re_Run All Jobs over a Time Window

Take Aways?

- The cost in people, time, and morale for a Data Pipeline bug is high and they can occur frequently.



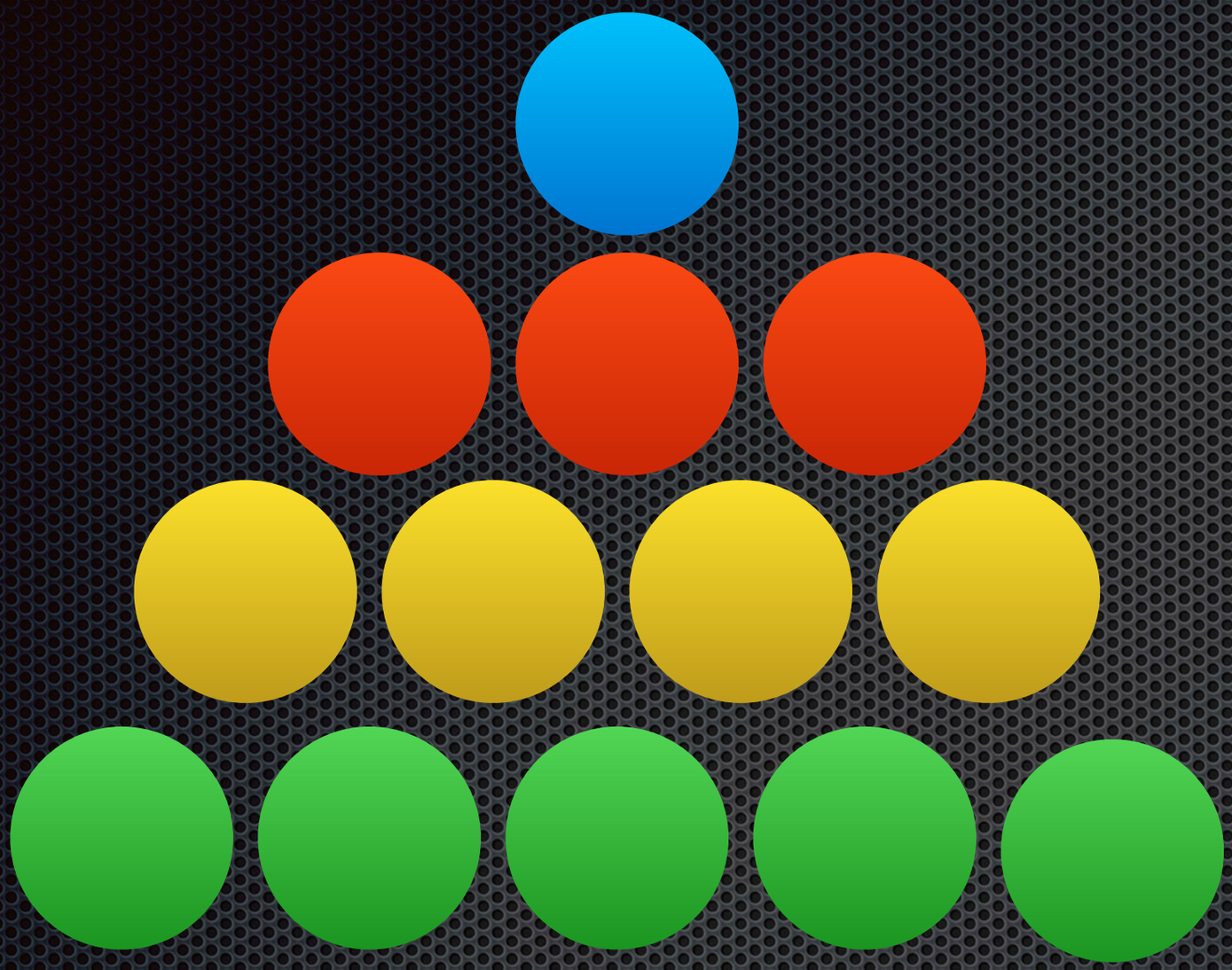


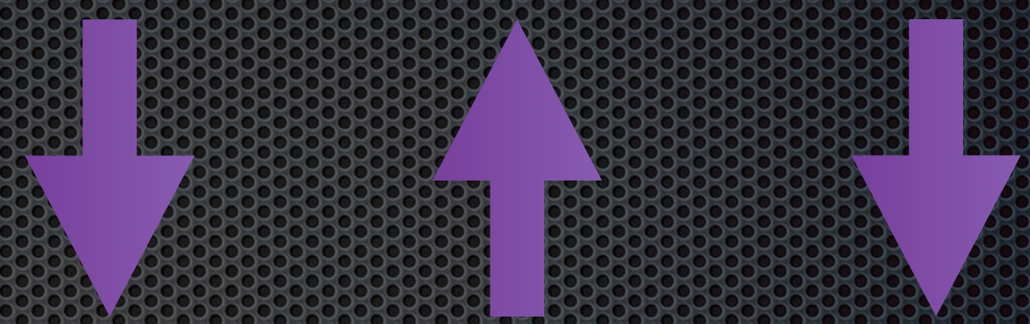
- In most areas of software, testing is invaluable, less so in data pipelines

- Data Pipeline bugs can be due to a logic problem **or** bad input data!

- Best Option : Detect & Rollback/Fix Forward

The Blast Radius Solution





Detect Bug

Identify Cause

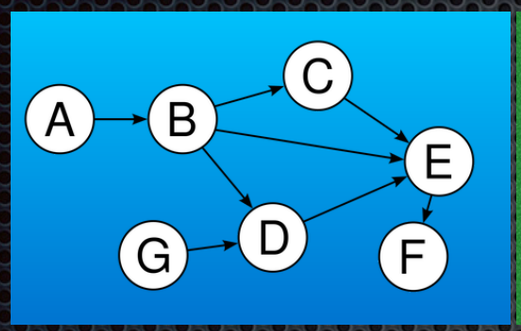
Re_Run 1 Job

Data Pipeline Challenges

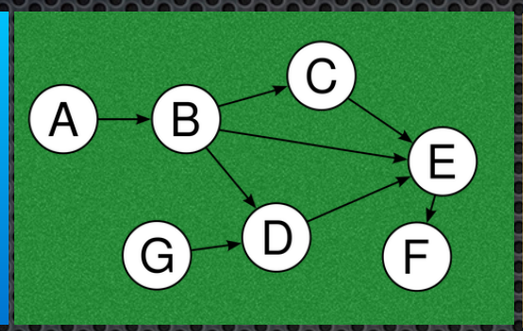
Timeliness

Timeliness

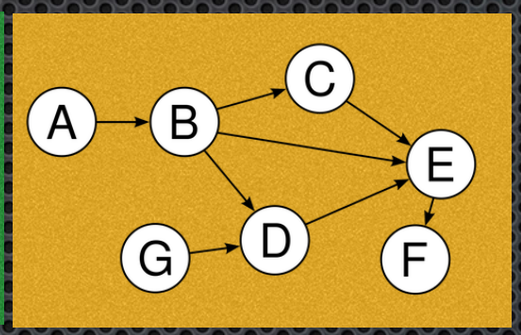
Definition : job = workflow = DAG of tasks



Job 1



Job 2

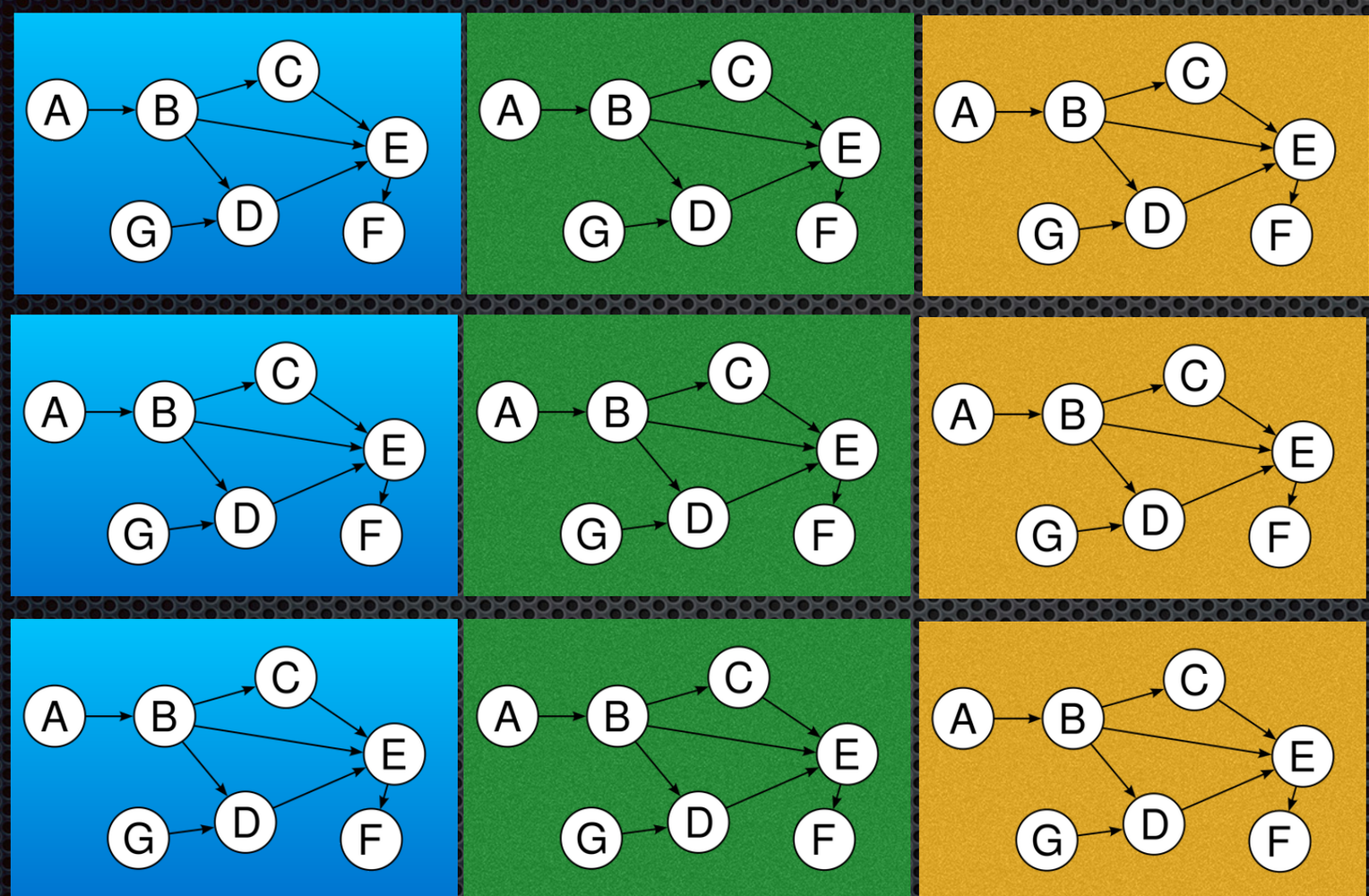


Job 3

Job 3's output is pushed to a serving system

Timeliness

Consider the Daily Run Schedule below:

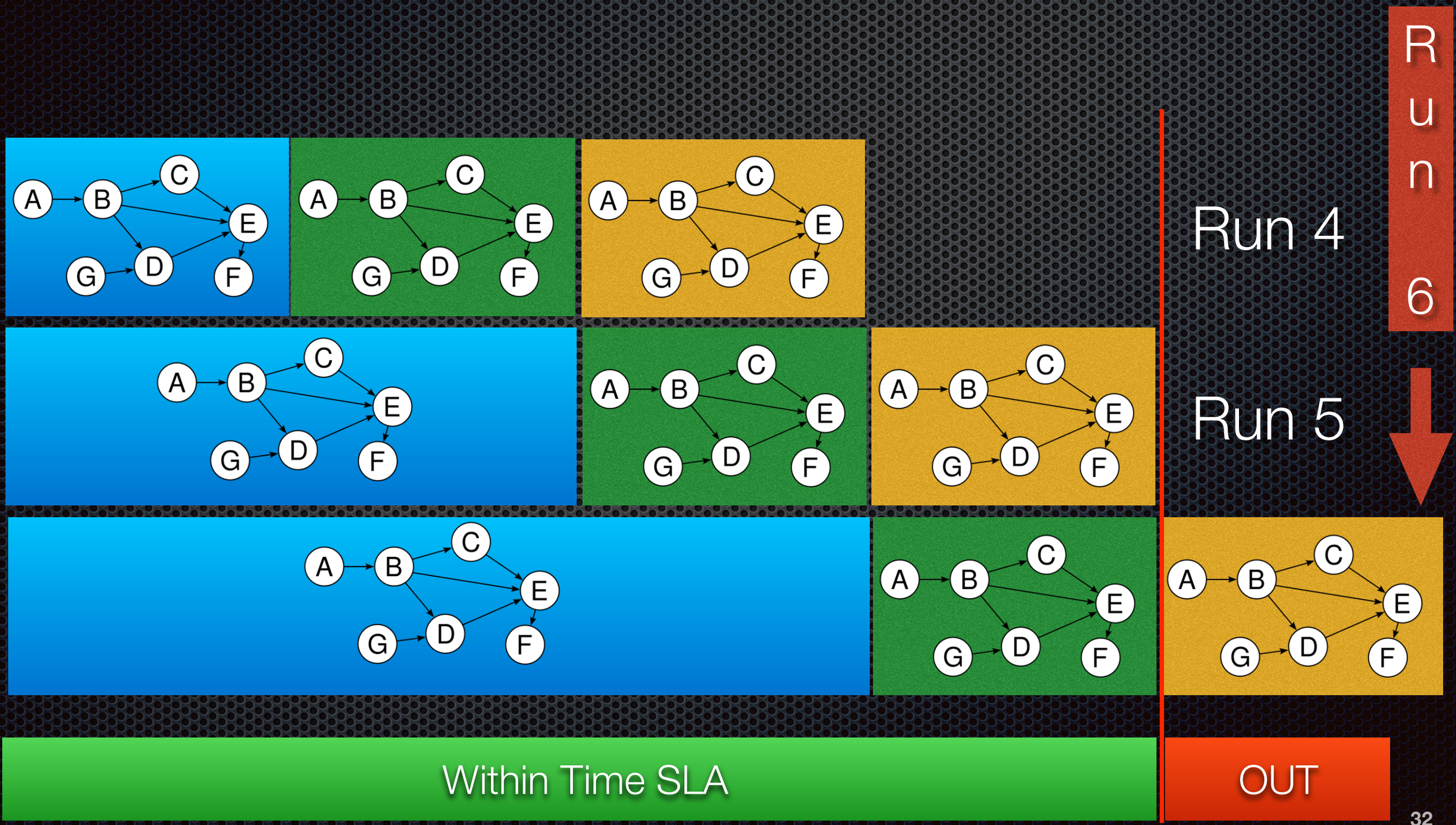


Run 1 : Monday

Run 2 : Tuesday

Run 3 : Wednesday

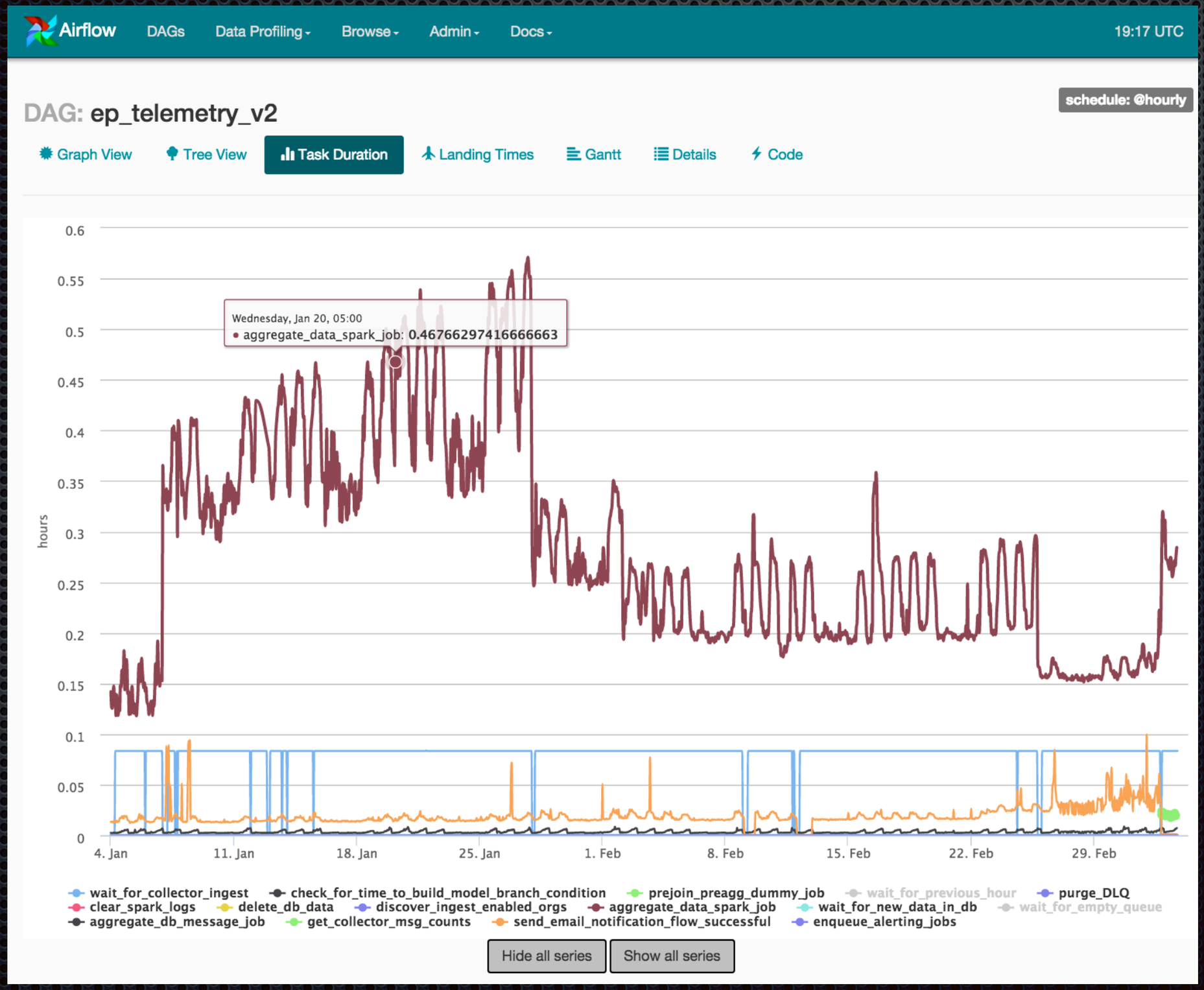
Timeliness



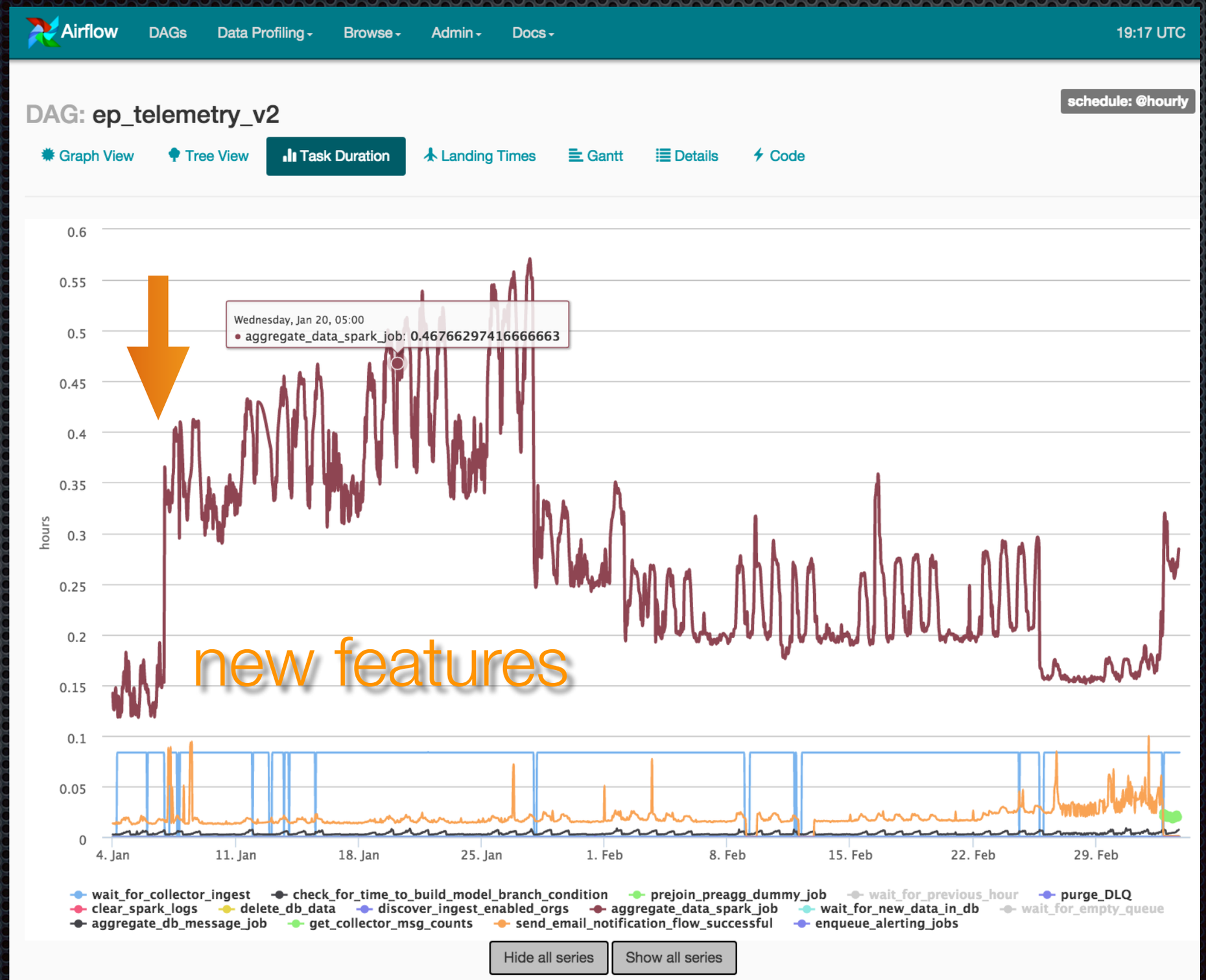
Timeliness

Why do jobs get slower?

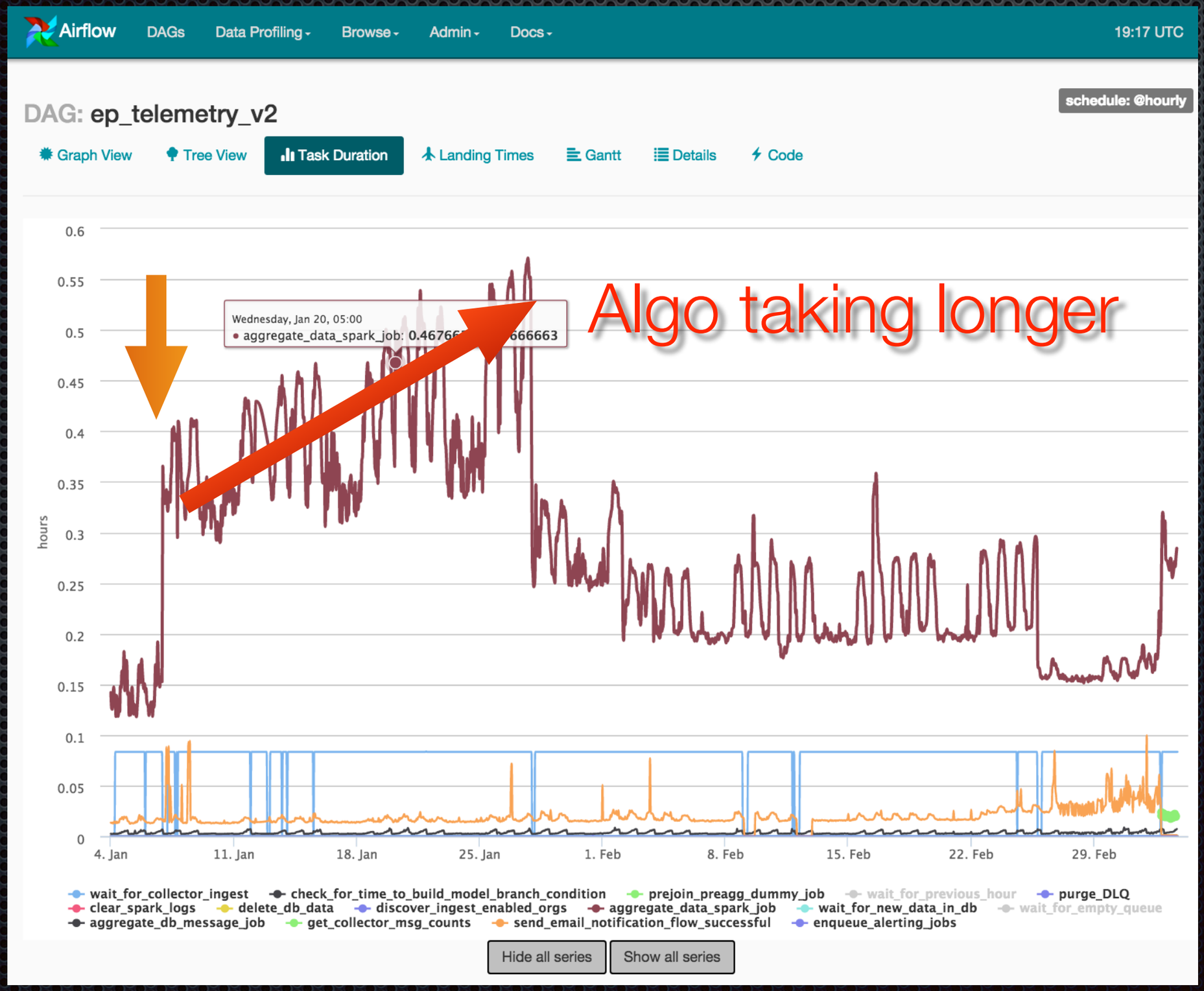
Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
schedule: @hourly
DAG: ep_telemetry_v2
Graph View
Tree View
Task Duration
Landing Times
Gantt
Details
Code
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766297416666663
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

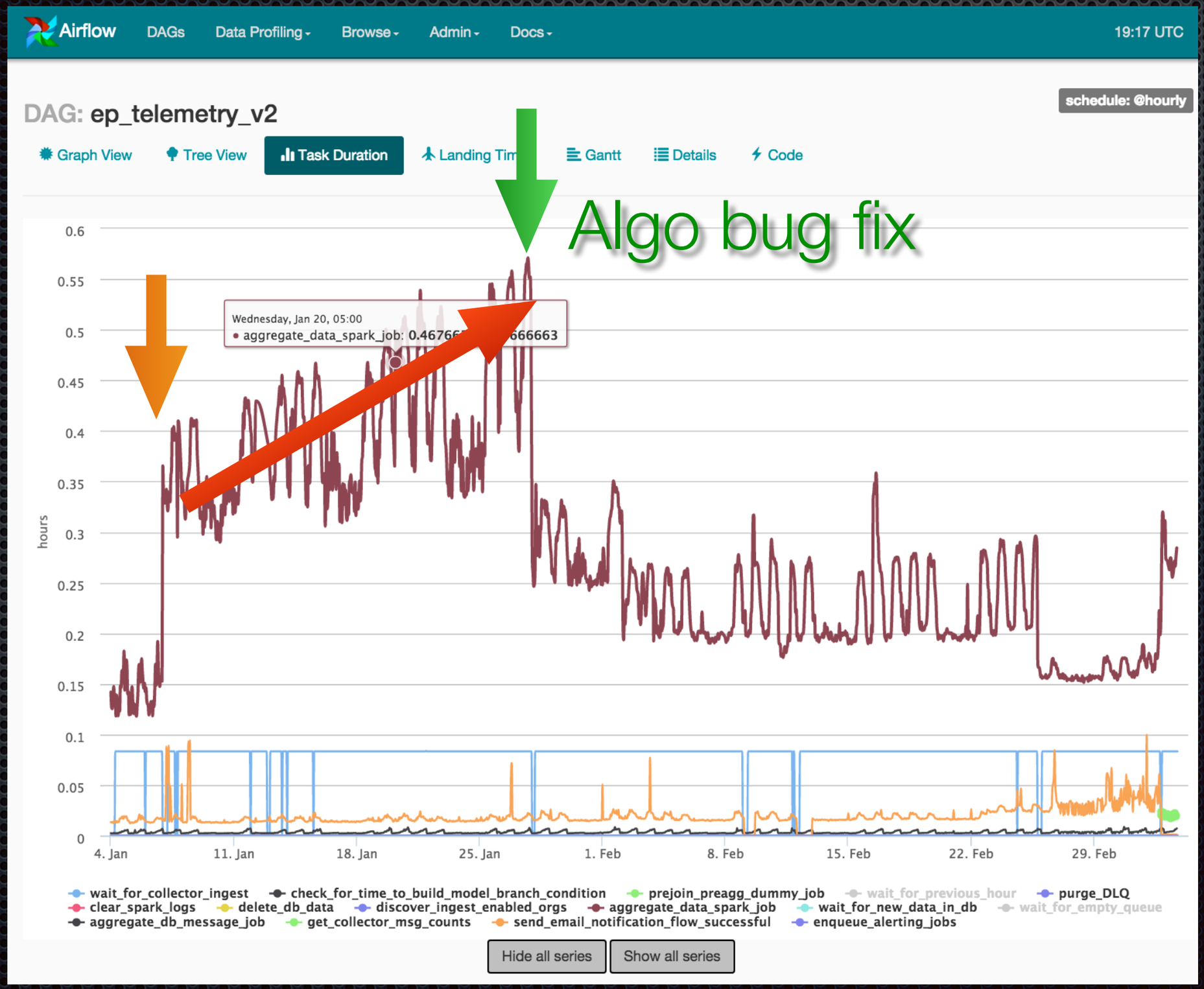
Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
schedule: @hourly
DAG: ep_telemetry_v2
Graph View
Tree View
Task Duration
Landing Times
Gantt
Details
Code
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766297416666663
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
new features
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

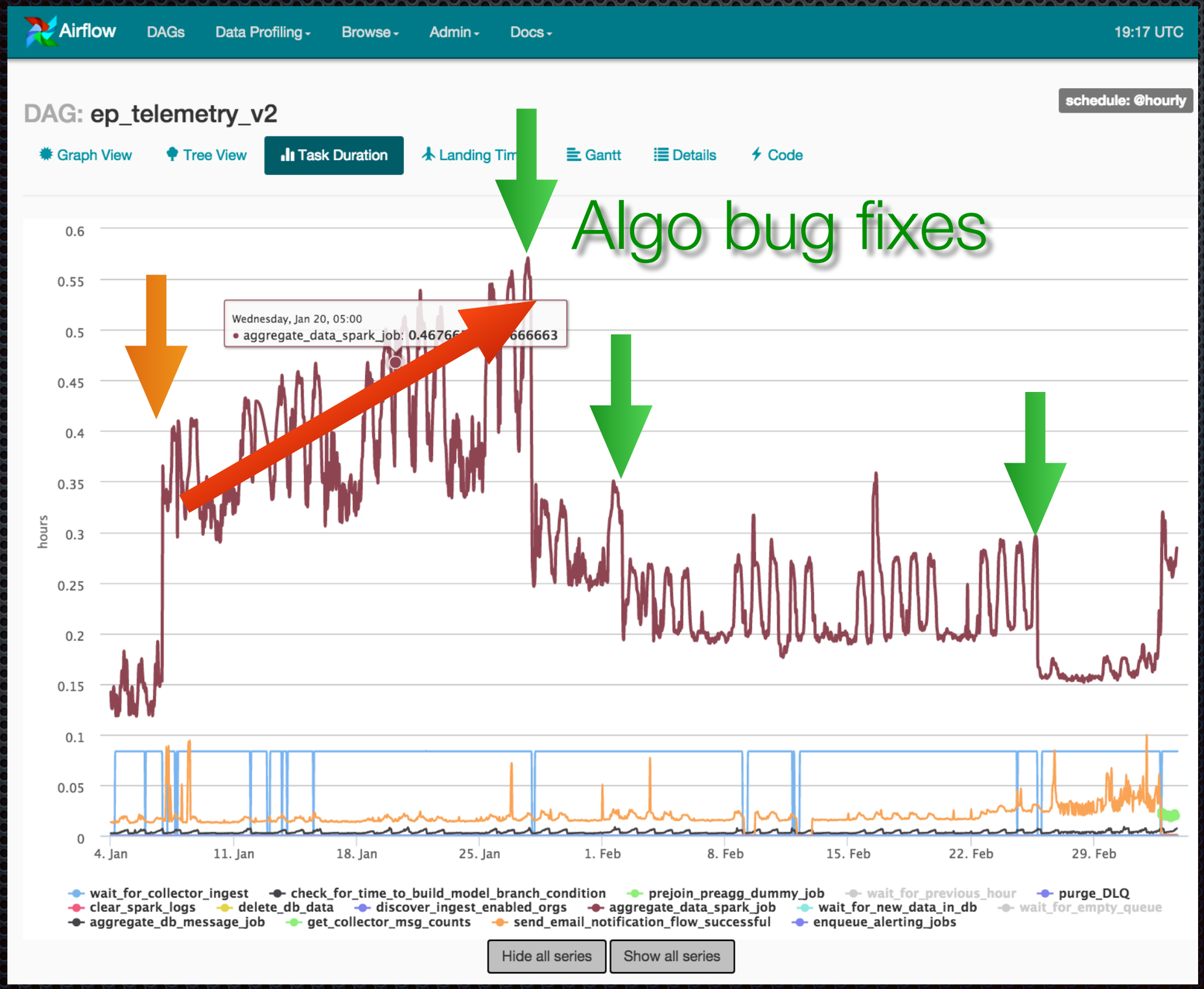
Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
DAG: ep_telemetry_v2
schedule: @hourly
Graph View
Tree View
Task Duration
Landing Times
Gantt
Details
Code
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766
666663
Algo taking longer
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

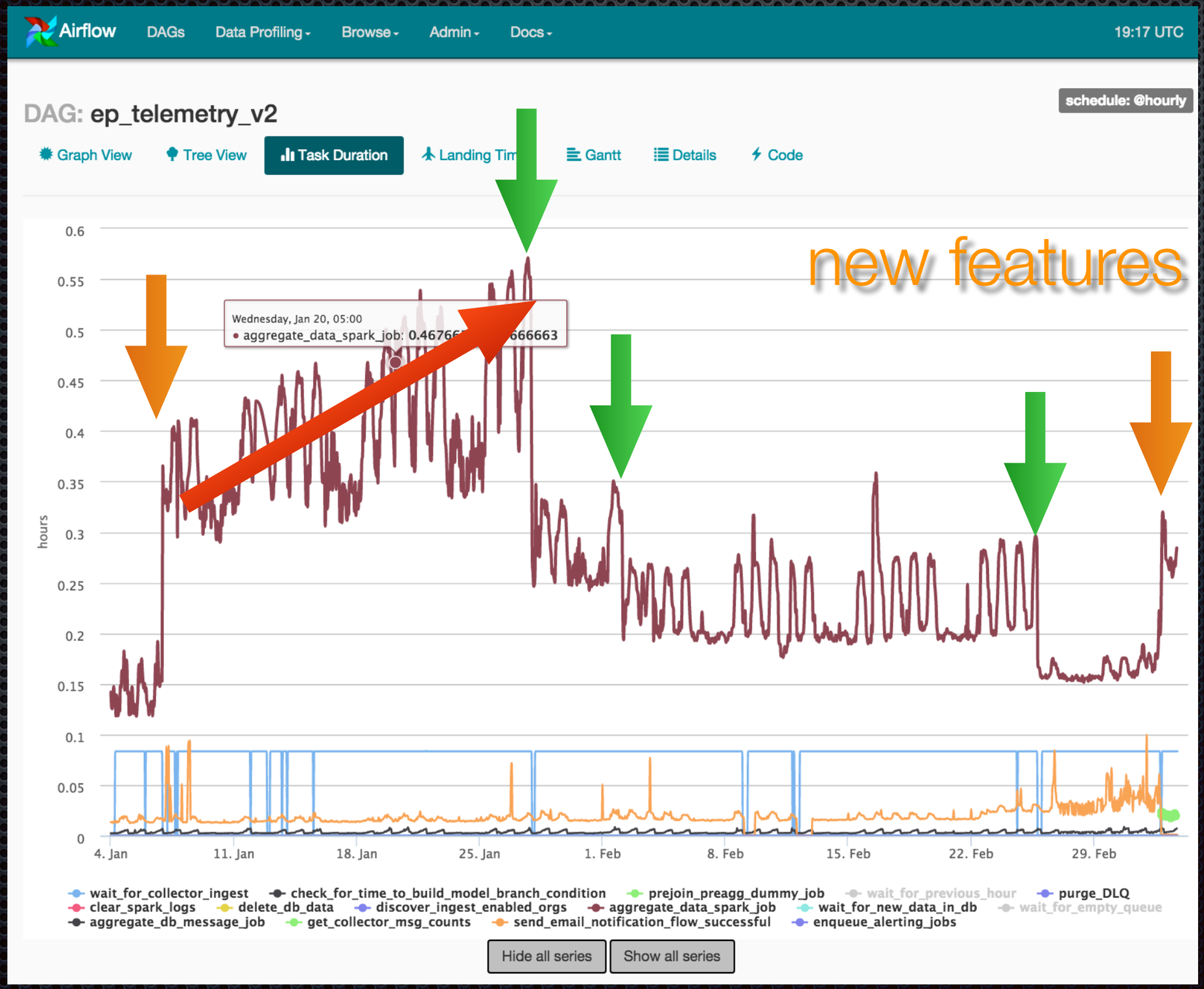
Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
schedule: @hourly
DAG: ep_telemetry_v2
Graph View
Tree View
Task Duration
Landing Tim
Gantt
Details
Code
Algo bug fix
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766
666663
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
DAG: ep_telemetry_v2
schedule: @hourly
Graph View
Tree View
Task Duration
Landing Tim
Gantt
Details
Code
Algo bug fixes
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766
666663
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

Timeliness


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
19:17 UTC
schedule: @hourly
DAG: ep_telemetry_v2
Graph View
Tree View
Task Duration
Landing Tim
Gantt
Details
Code
new features
Wednesday, Jan 20, 05:00
aggregate_data_spark_job: 0.46766
666663
hours
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
4. Jan
11. Jan
18. Jan
25. Jan
1. Feb
8. Feb
15. Feb
22. Feb
29. Feb
wait_for_collector_ingest
check_for_time_to_build_model_branch_condition
prejoin_preagg_dummy_job
wait_for_previous_hour
purge_DLQ
clear_spark_logs
delete_db_data
discover_ingest_enabled_orgs
aggregate_data_spark_job
wait_for_new_data_in_db
wait_for_empty_queue
aggregate_db_message_job
get_collector_msg_counts
send_email_notification_flow_successful
enqueue_alerting_jobs
Hide all series
Show all series

Take Aways?

- Data Science & Engineering work is a virtuous cycle of adding features (and the like) + tuning performance
- Latency does matter (a bit)

Design Goals

Desirable Qualities of a Resilient Data Pipeline

Desirable Qualities of a Resilient Data Pipeline

Correctness	Operability
Timeliness	Cost

Desirable Qualities of a Resilient Data Pipeline

Correctness

- Data Integrity (no loss, etc…)
- Expected data distributions

Operability

- Fine-grained Monitoring & Alerting of Correctness & Timeliness SLAs
- Quick Recoverability

Timeliness

- All output within time-bound SLAs

Cost

- Pay-as-you-go

Quickly Recoverable

- Bugs happen!
- Bugs in Predictive Data Pipelines have a large blast radius
- Optimize for MTTR

Maintainability

MTTR Optimized

Versus

MTBF Optimized

More info here: http://ti.arc.nasa.gov/projects/ishem/Papers/ONeill_Maintainability.doc

Implementation

Using AWS to meet Design Goals

SQS

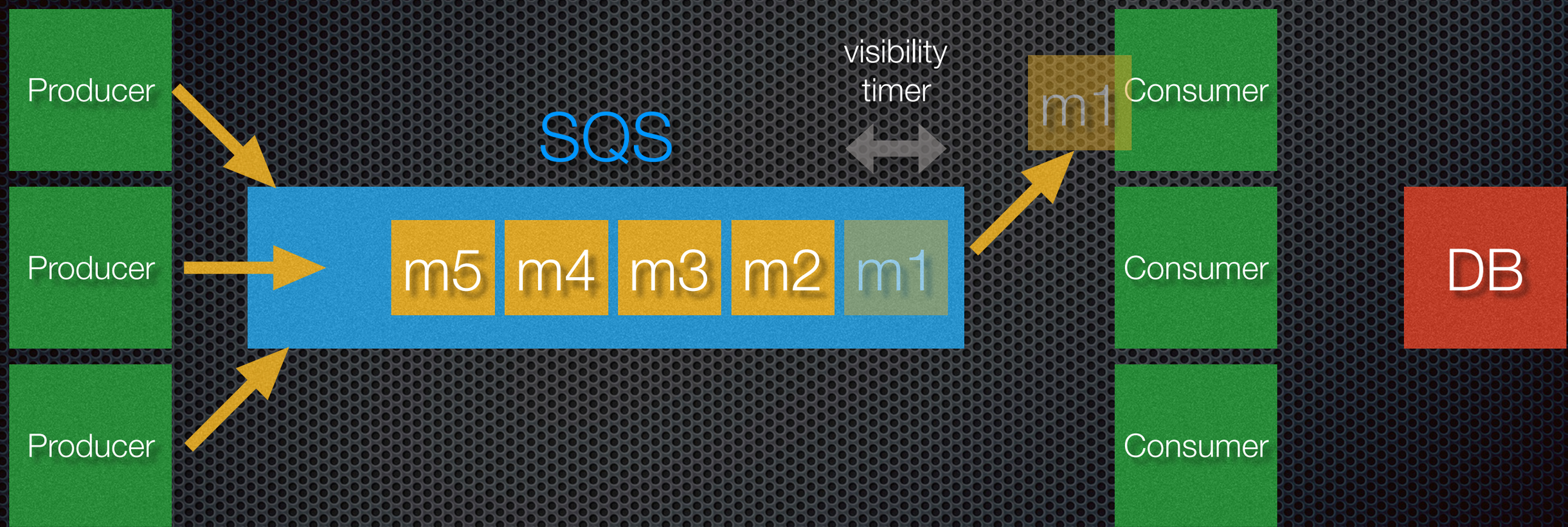
Simple Queue Service

SQS - Overview

- AWS's low-latency, highly scalable, highly available message queue
  - Infinitely Scalable Queue (though not FIFO)
  - Low End-to-end latency (generally sub-second)
  - Pull-based

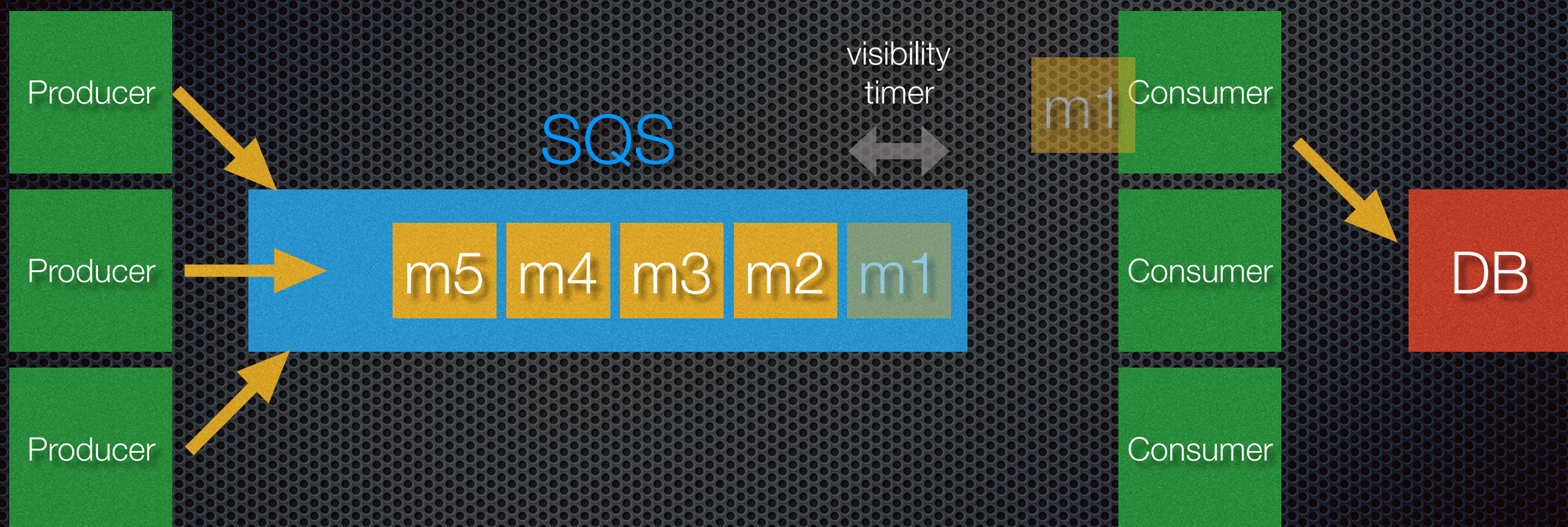
SQS - Typical Operation Flow

Step 1: A consumer reads a message from SQS. This starts a visibility timer!



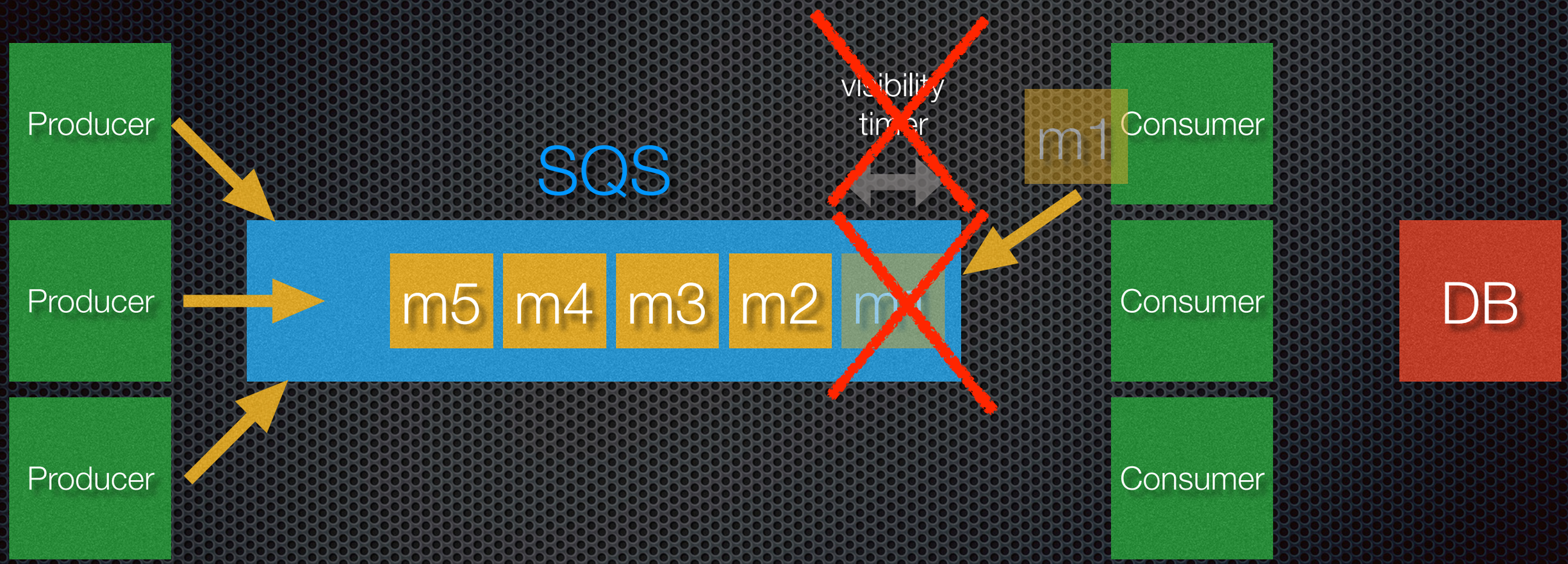
SQS - Typical Operation Flow

Step 2: Consumer persists message contents to DB



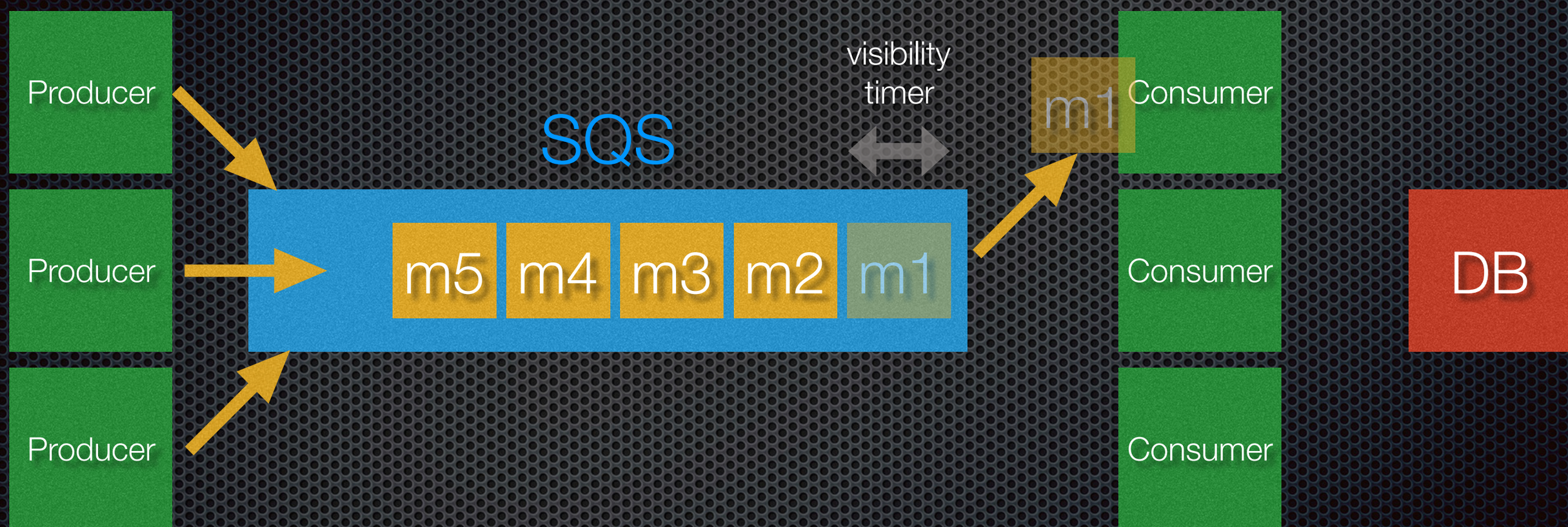
SQS - Typical Operation Flow

Step 3: Consumer ACKs message in SQS



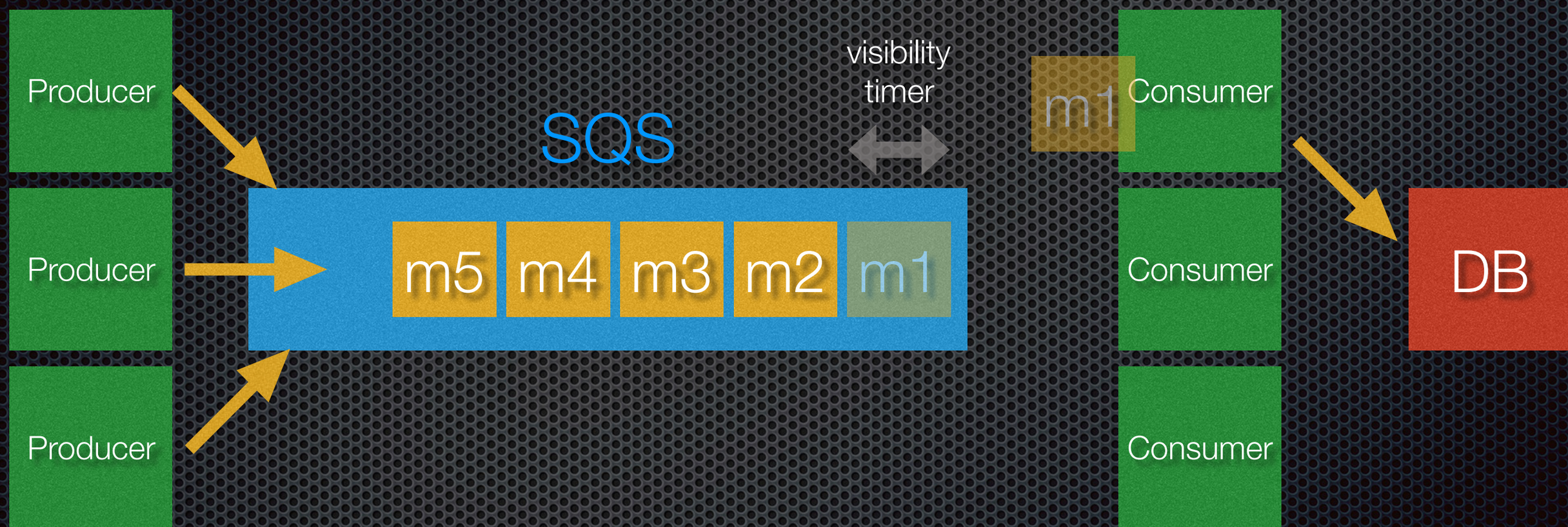
SQS - Time Out Example

Step 1: A consumer reads a message from SQS



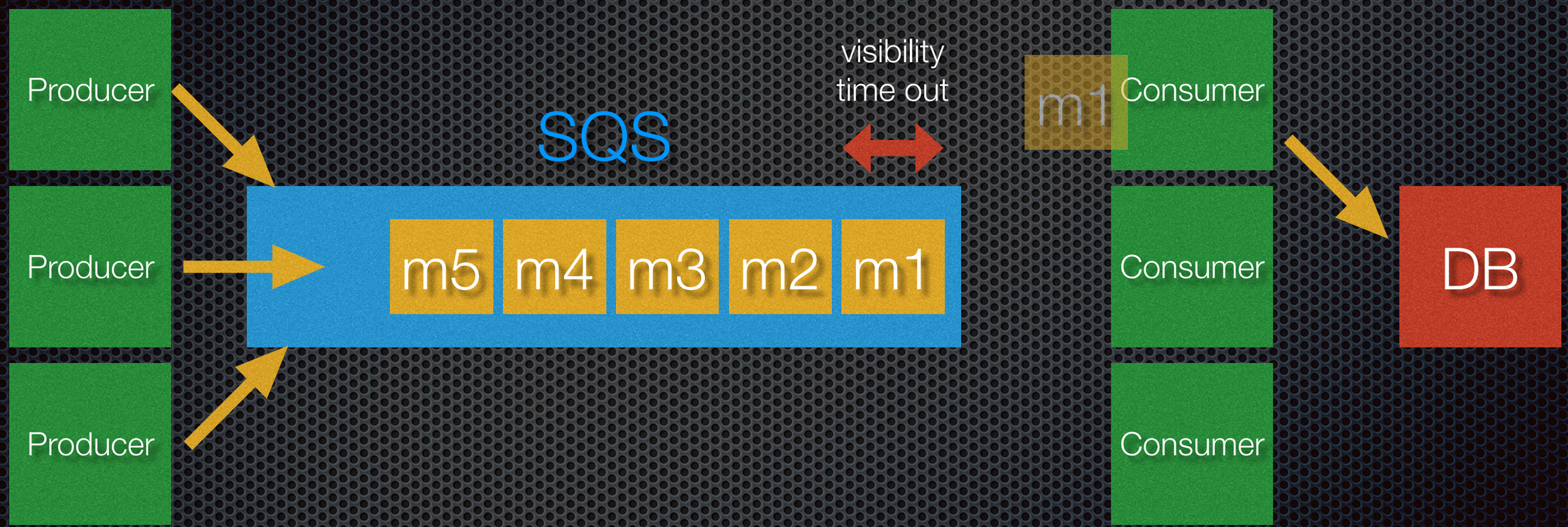
SQS - Time Out Example

Step 2: Consumer attempts persists message contents to DB



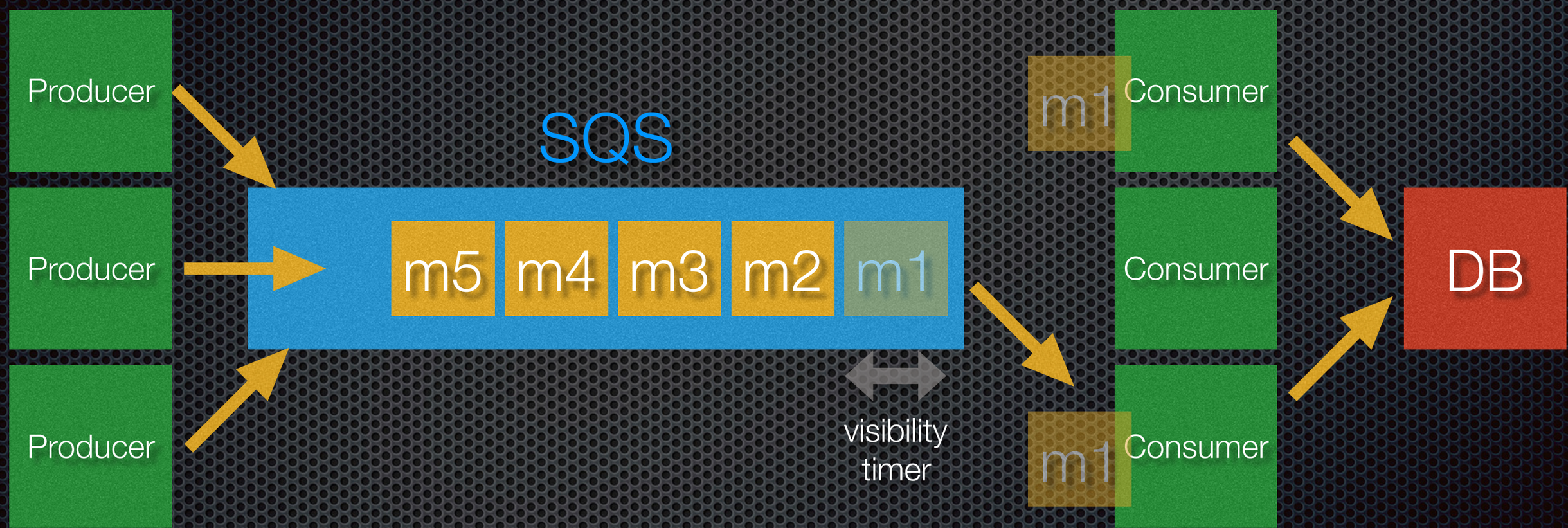
SQS - Time Out Example

Step 3: A Visibility Timeout occurs & the message becomes visible again.



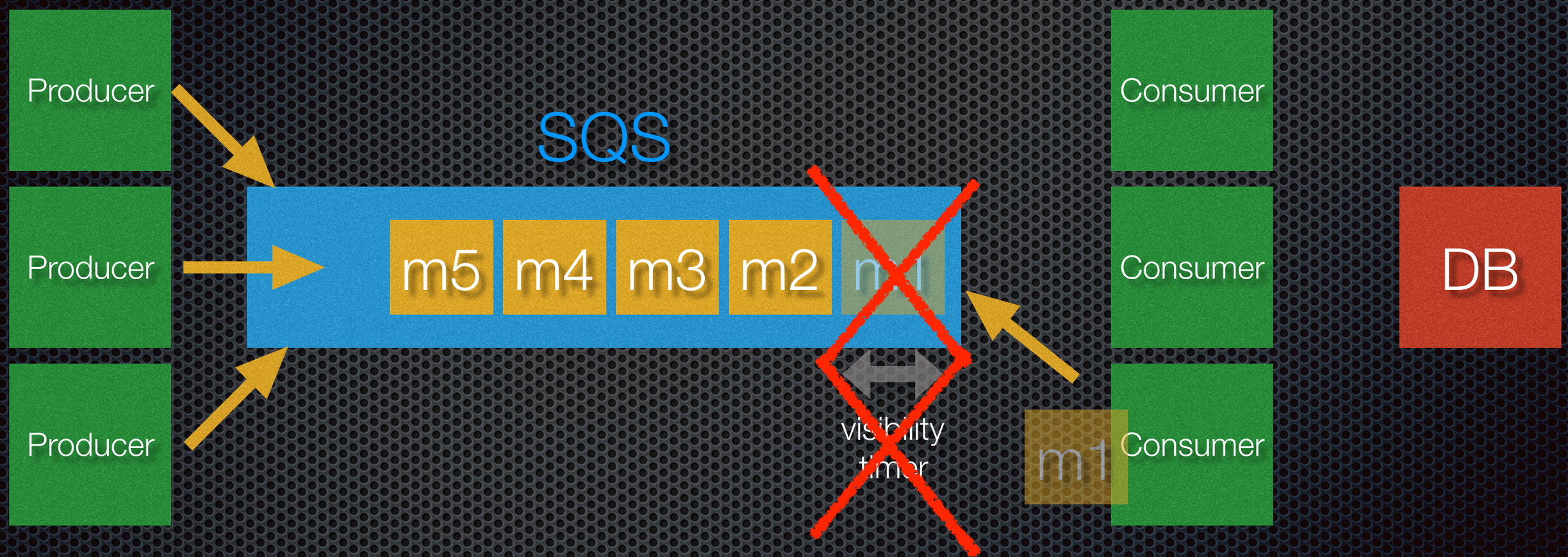
SQS - Time Out Example

Step 4: Another consumer reads and persists the same message

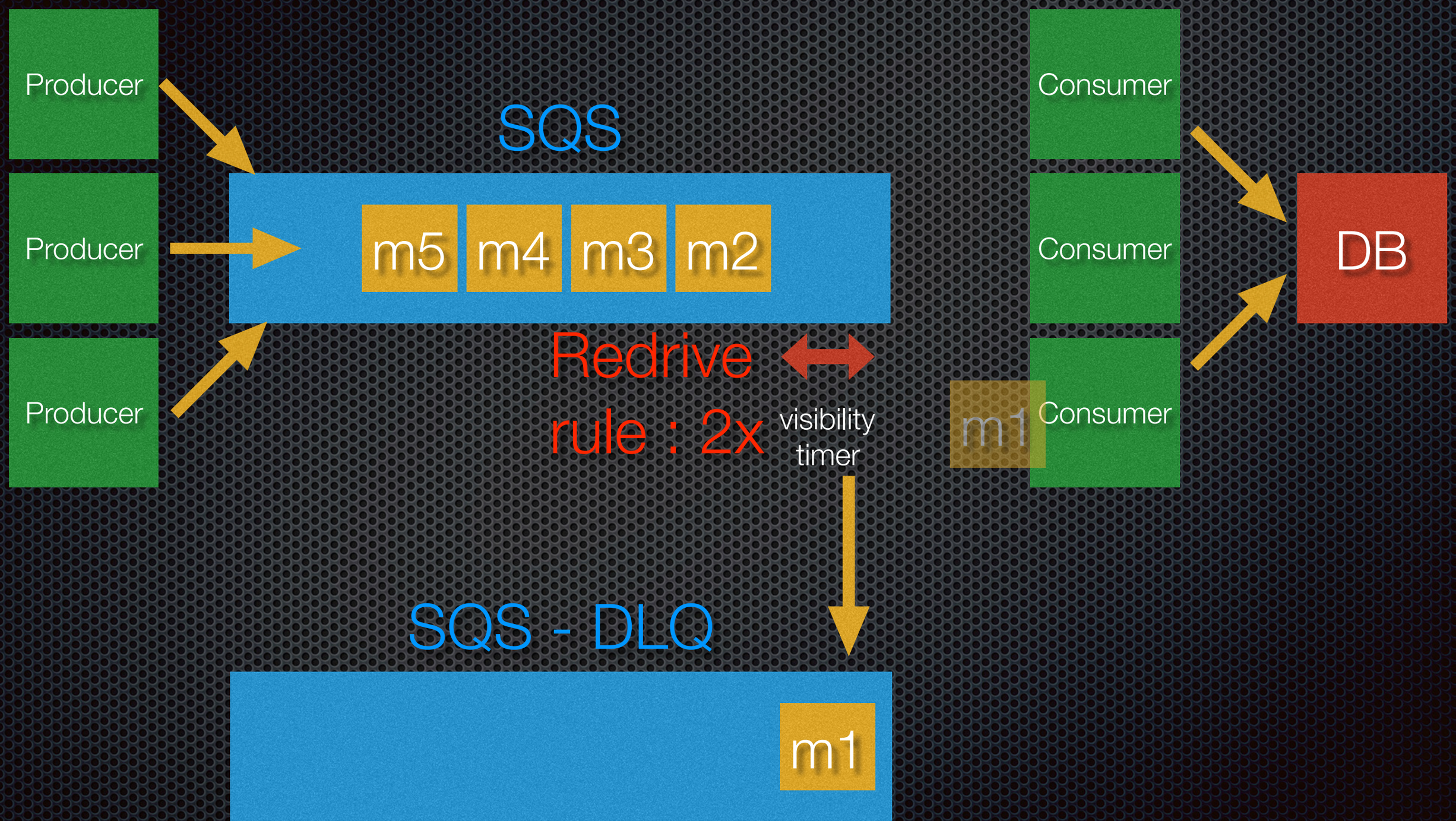


SQS - Time Out Example

Step 5: Consumer ACKs message in SQS



SQS - Dead Letter Queue



SNS

Simple Notification Service

SNS - Overview

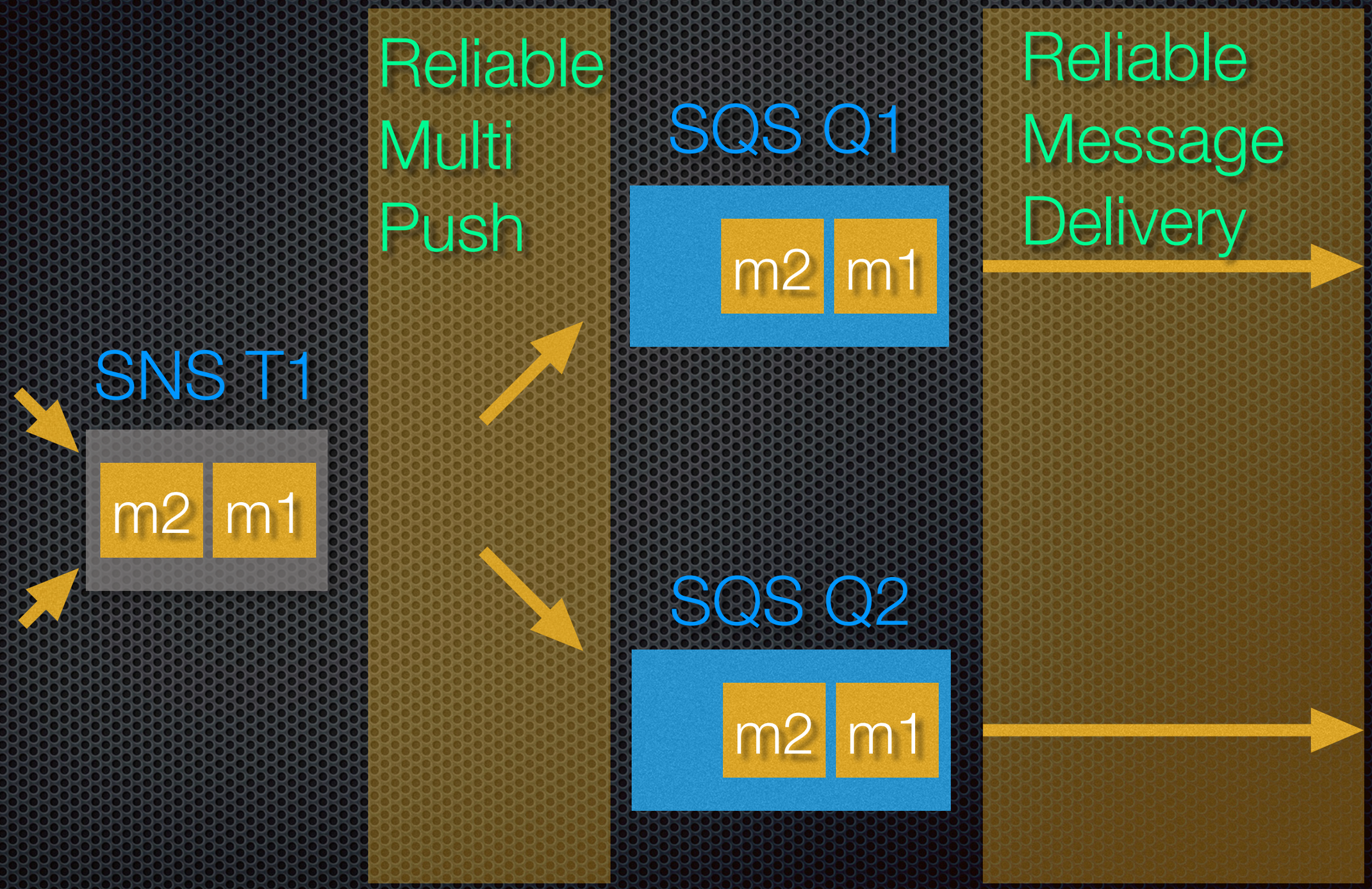
- Highly Scalable, Highly Available, Push-based Topic Service

  - Whereas SQS is pull-based, SNS is push-based
    - There is no message retention & there is a finite retry count
    - **No** Reliable Message Delivery

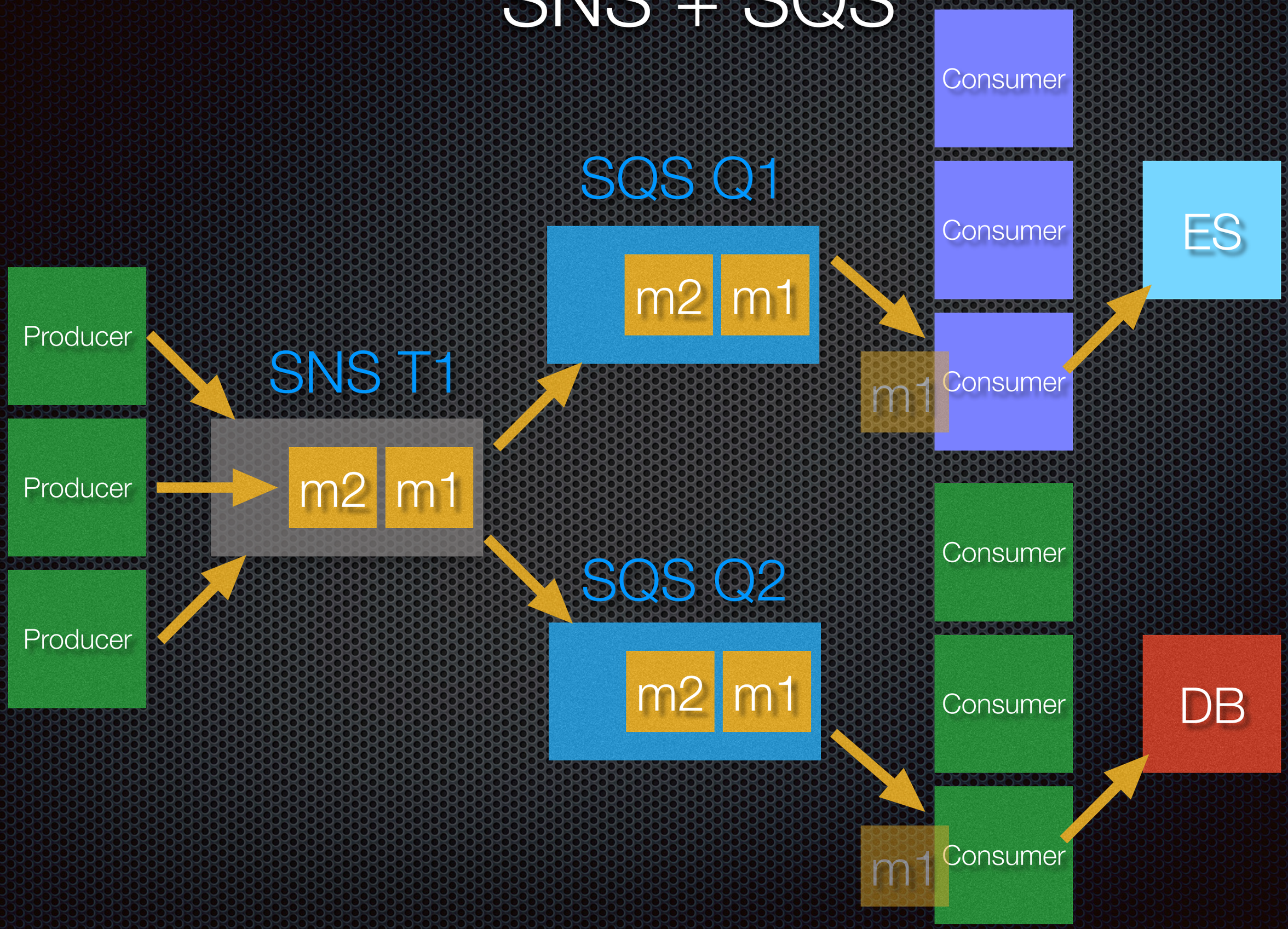
  - Whereas SQS ensures each message is seen by at least 1 consumer
    - SNS ensures that each message is seen by every consumer
    - Reliable Multi-Push

Can we work around this limitation while getting Reliable Multi-push?

SNS + SQS Design Pattern



SNS + SQS


Producer
Producer
Producer
SNS T1
m2
m1
SQS Q1
m2
m1
SQS Q2
m2
m1
Consumer
Consumer
Consumer
m1
Consumer
Consumer
Consumer
m1
Consumer
ES
DB

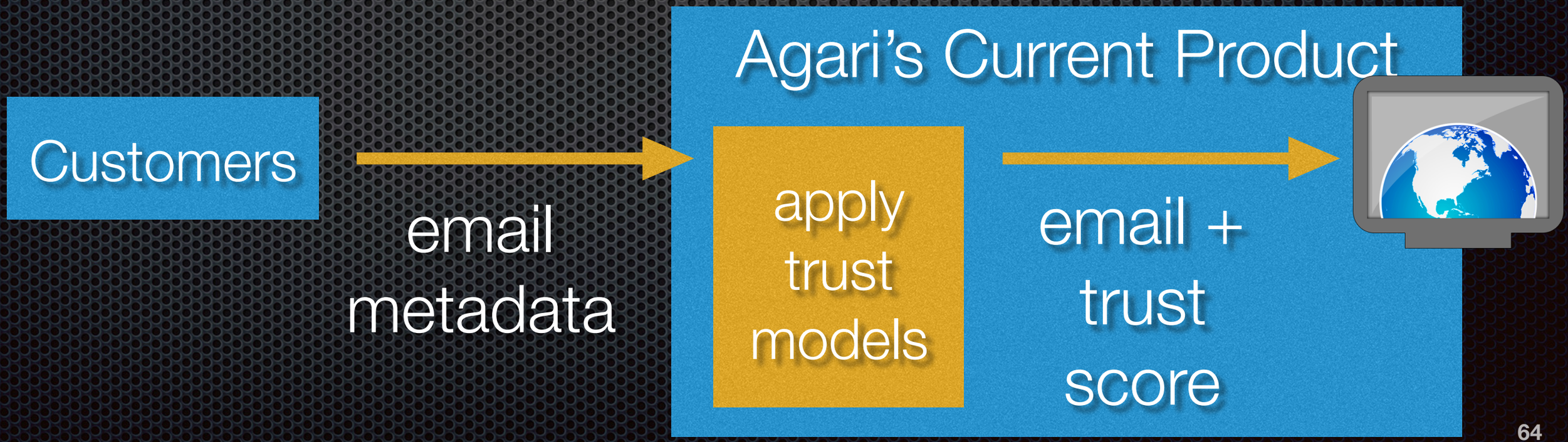
Batch Pipeline Architecture

Putting the Pieces Together

But First ....

What Does Agari Do?

What Does Agari Do?



What Does Agari Do?

Enterprise Customers

email metadata

Agari's Future Product

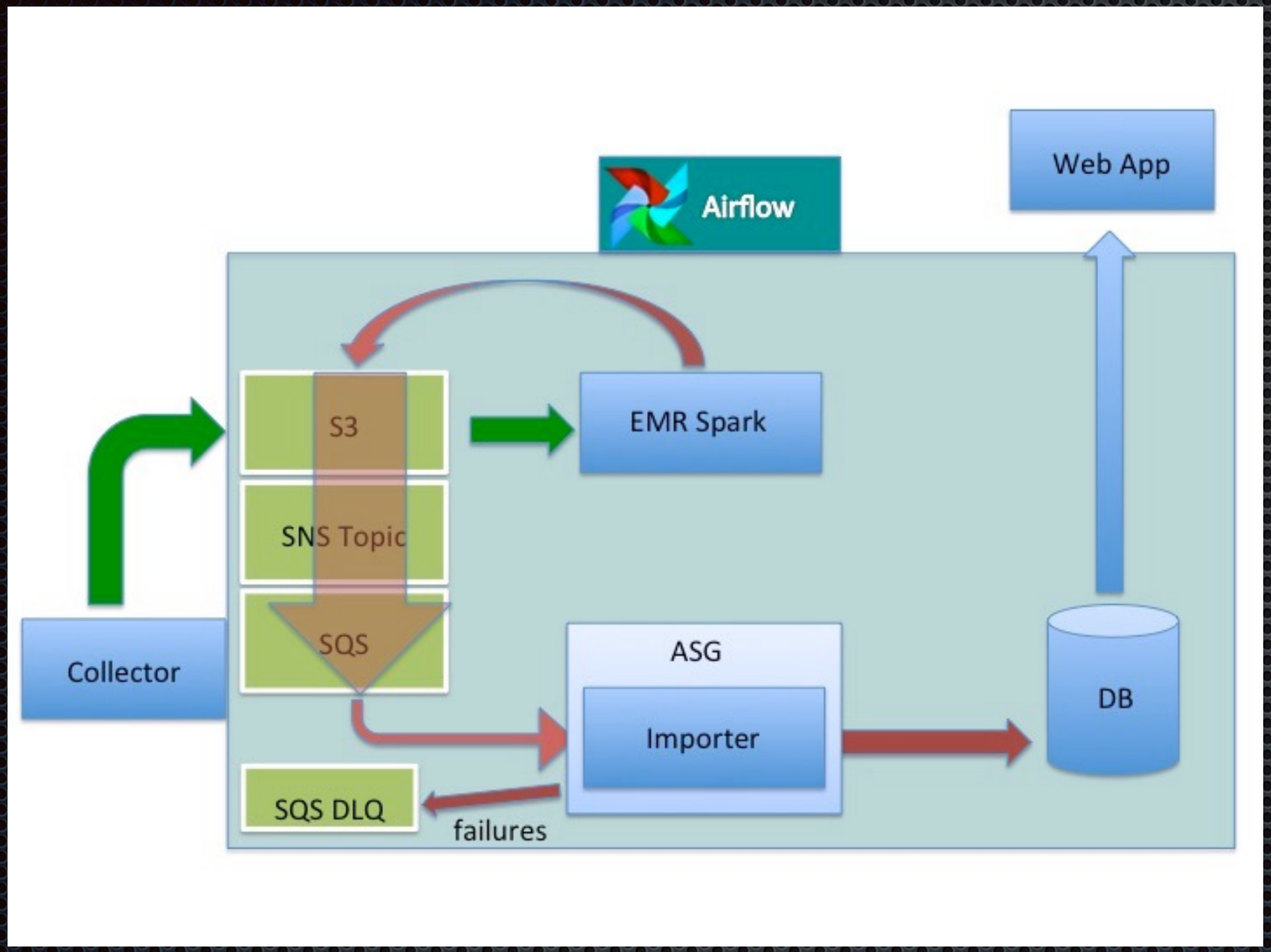
apply trust models

email md + trust score

Batch Pipeline Architecture

Putting the Pieces Together

Batch Architecture



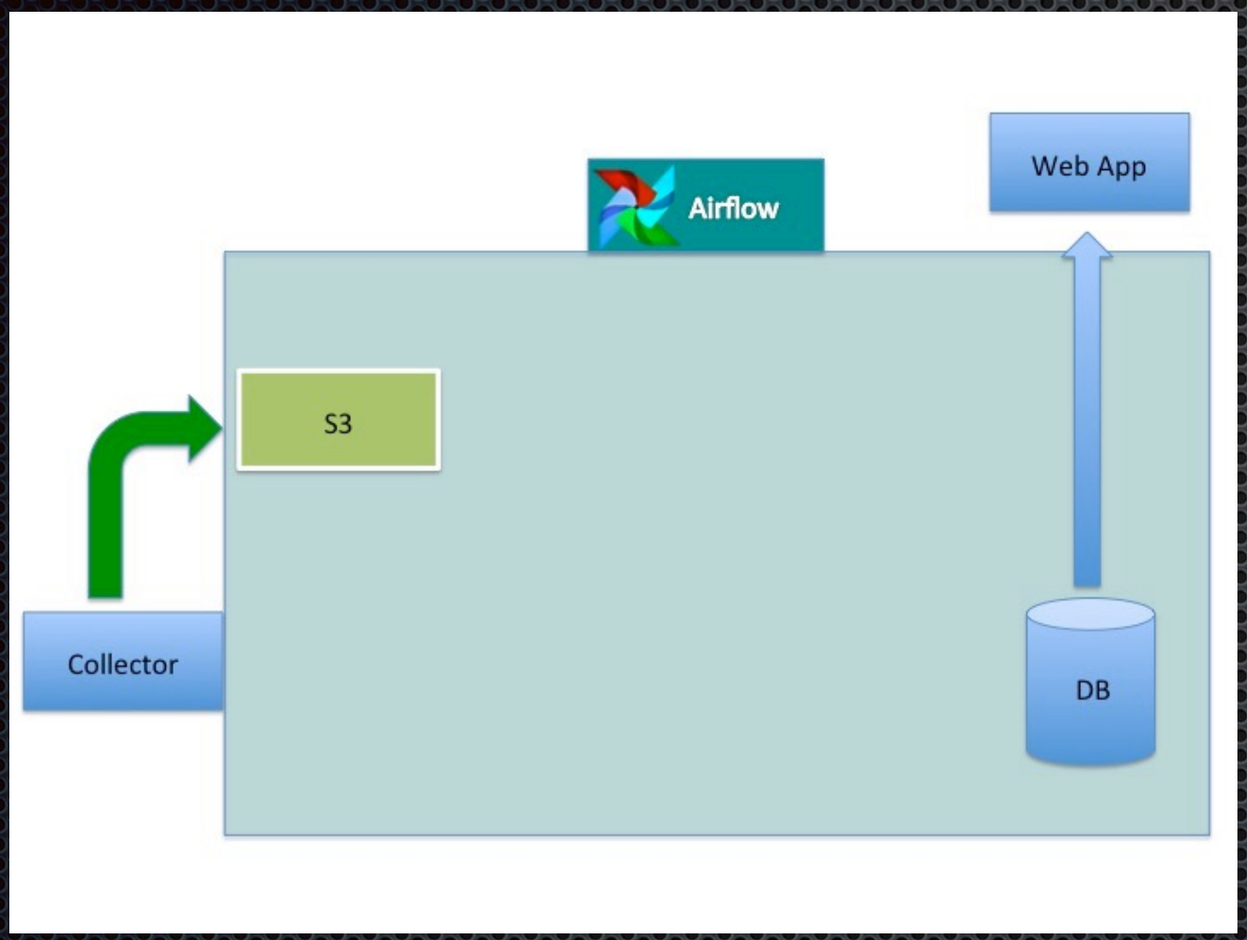
- S3 to hold all source & computed data (**Avro**)
- EMR Spark for scoring + summarization
- Apache Airflow for hourly job scheduling
- SNS+SQS for messaging
- ASG Importer to import
- WebApp in Ruby-on-Rails

Tackling Cost & Timeliness

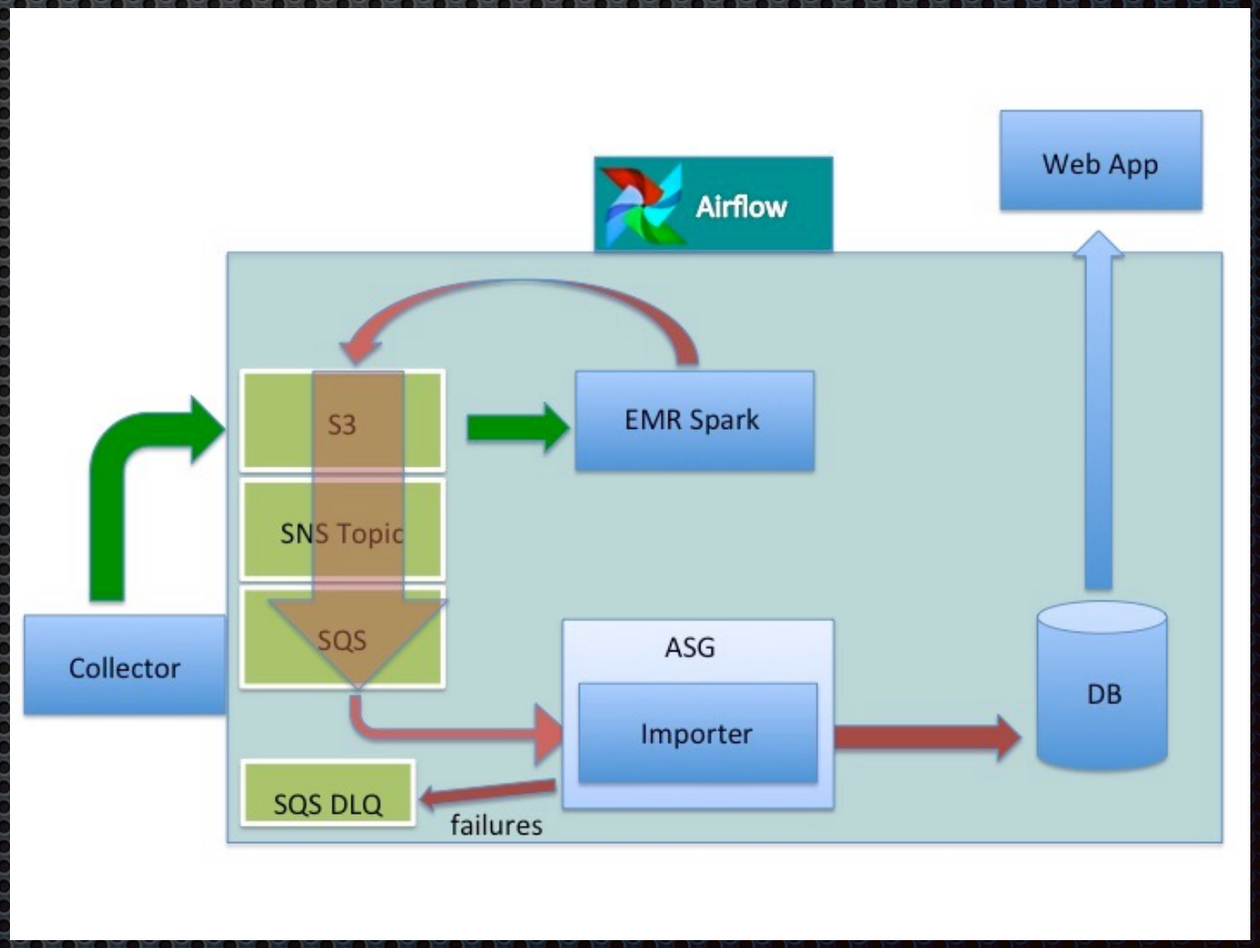
Leveraging the AWS Cloud

Tackling Cost

Between Hourly Runs



During Hourly Runs

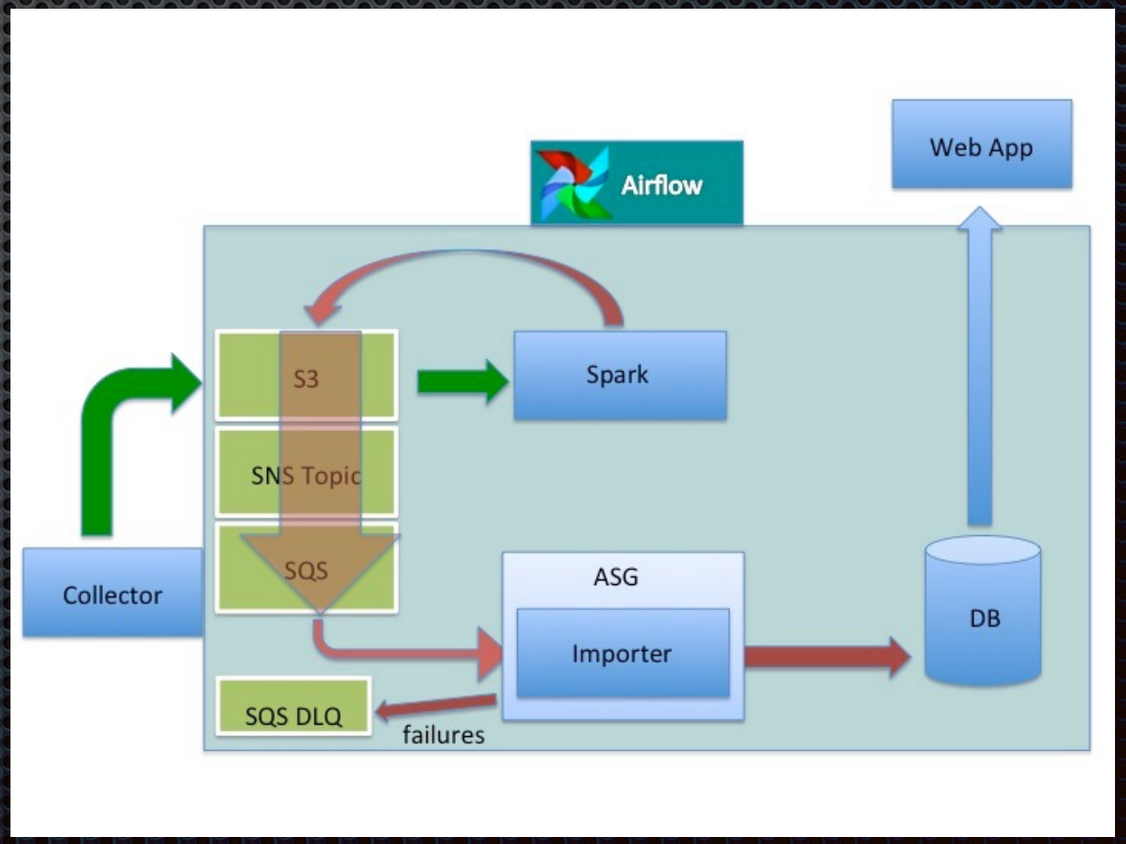


Tackling Timeliness

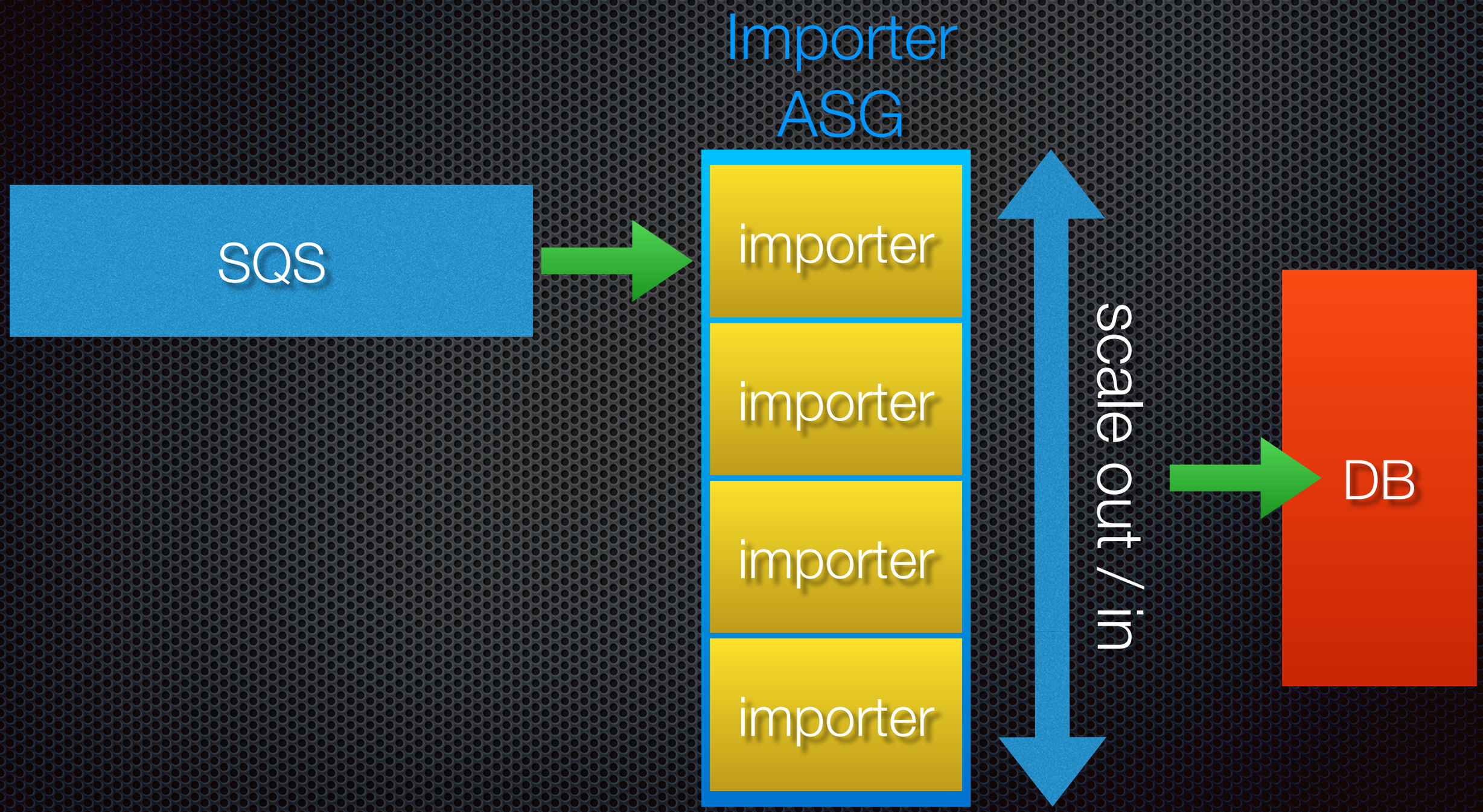
Auto Scaling Group (ASG)

ASG - Overview

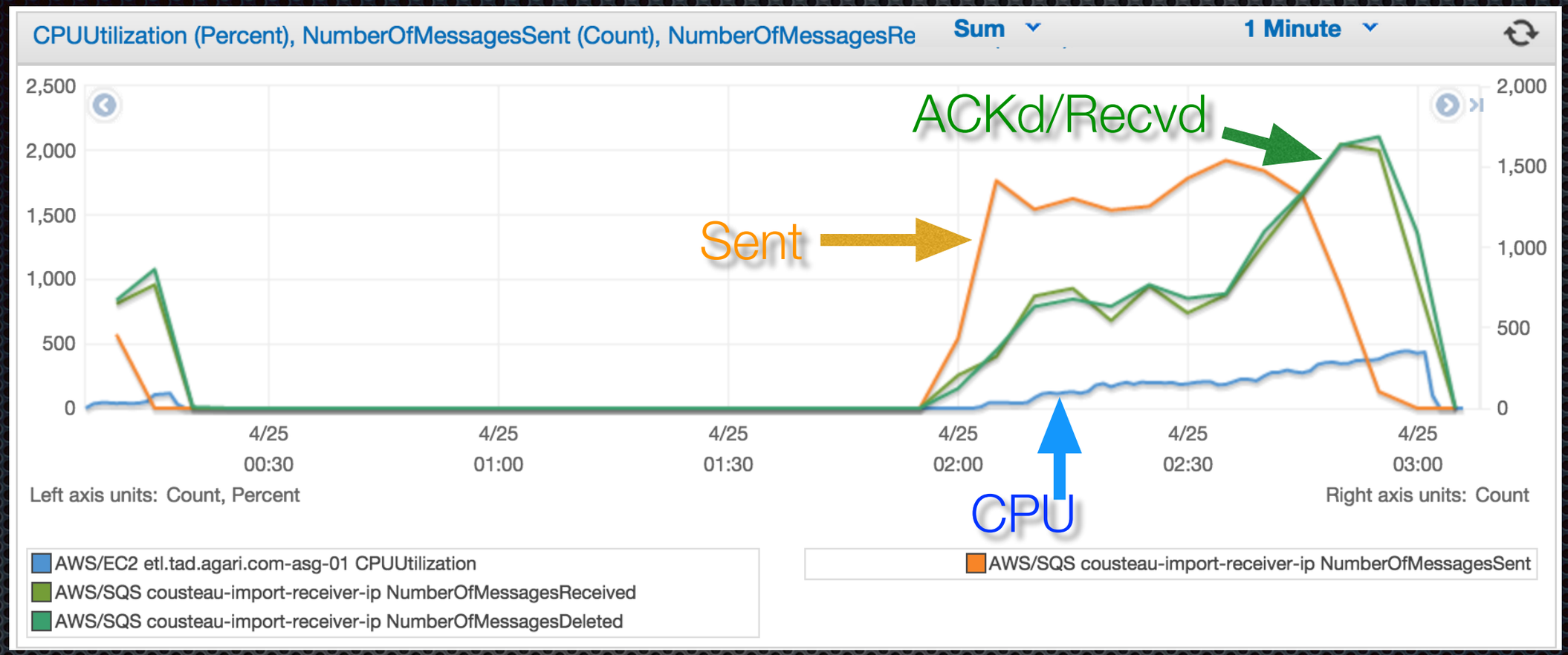
- What is it?
  - A means to automatically scale out/in clusters to handle variable load/traffic
  - A means to keep a cluster/service of a fixed size always up

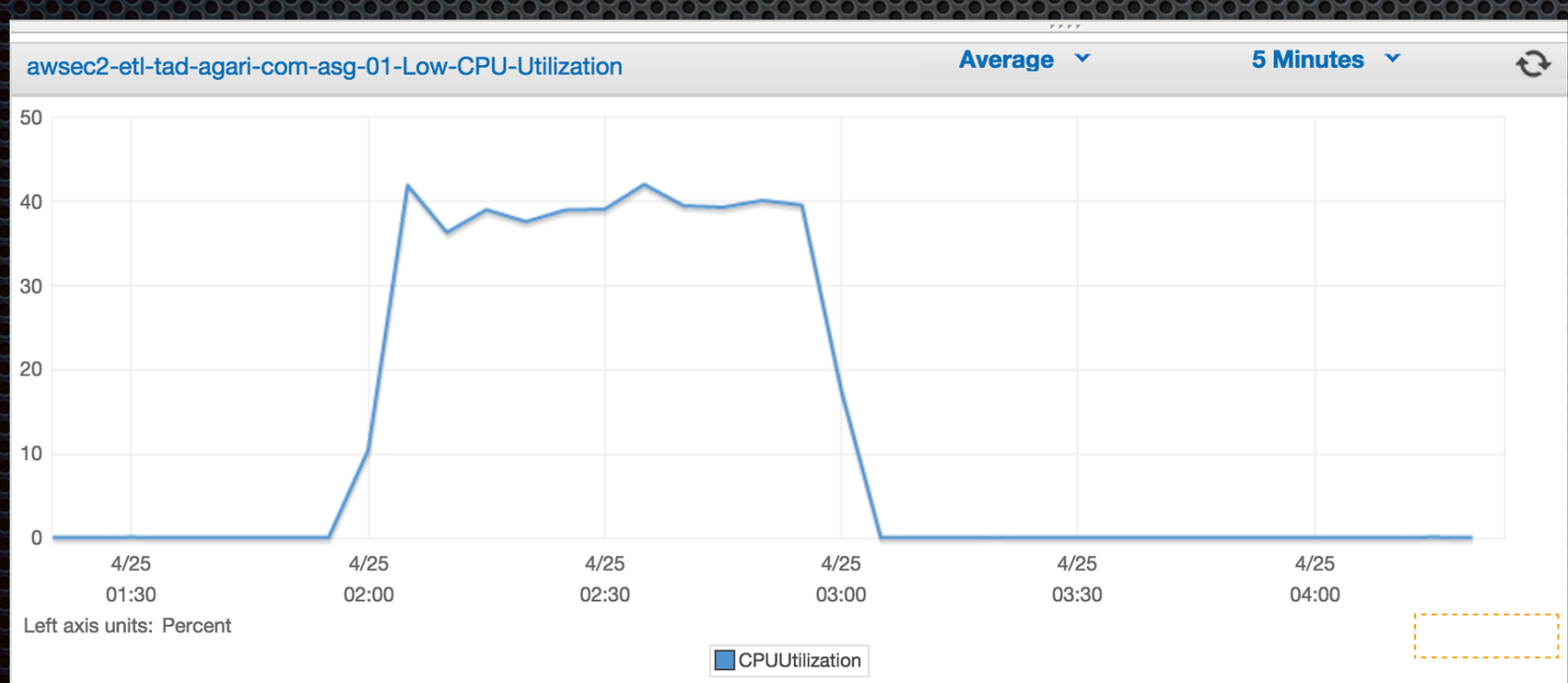


ASG - Data Pipeline



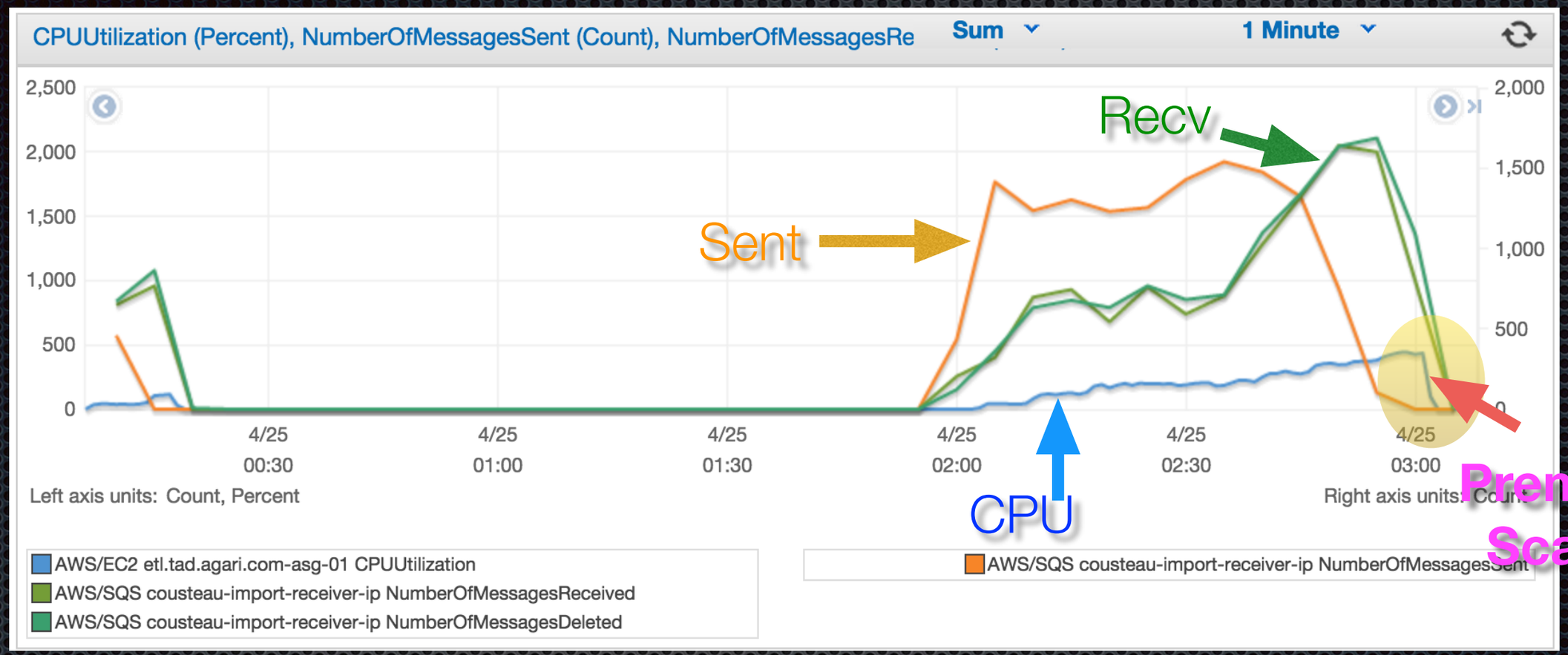
ASG : CPU-based





CPU-based auto-scaling is good at scaling in/out to keep the average CPU constant

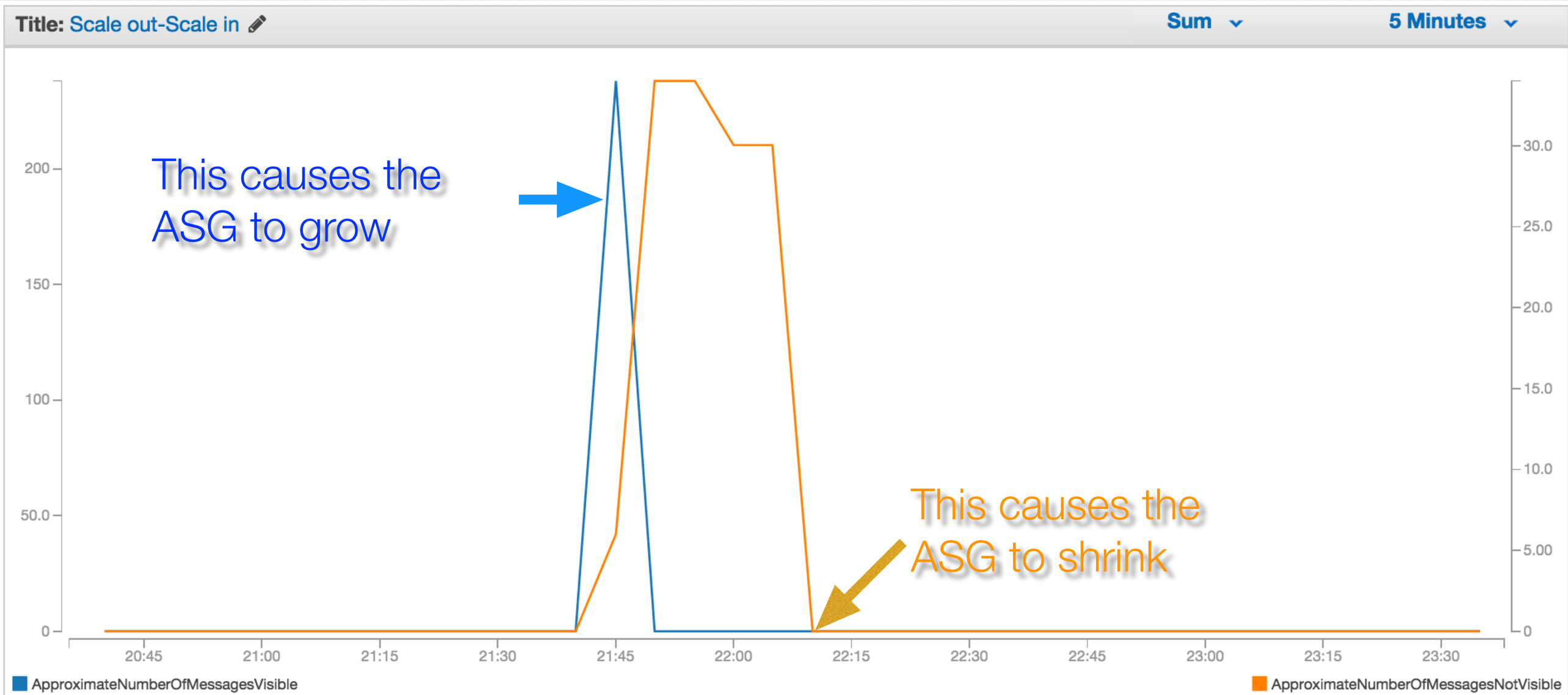
ASG : **CPU**-based



Premature Scale-in:

- The CPU drops to noise-levels before all messages are consumed
- This causes scale in to occur while the last few messages are still being committed

ASG : **Queue**-based



Scale-out: When Visible Messages > 0 (a.k.a. when queue depth > 0)

Scale-in: When Invisible Messages = 0 (a.k.a. when the last in-flight message is ACK'd)

Desirable Qualities of a Resilient Data Pipeline

Correctness

Operability

Timeliness

- ASG
- EMR Spark

Cost

- ASG
- EMR Spark

Tackling Operability & Correctness

Leveraging Tooling

Tackling Operability : Requirements

- A simple way to author and manage workflows
- Provides visual insight into the state & performance of workflow runs
- Integrates with our alerting and monitoring tools

Apache Airflow

Workflow Automation & Scheduling

Apache Airflow - Authoring DAGs

Airflow: Author DAGs in Python! No need to bundle many config files!

Airflow DAGs Data Profiling Browse Admin Docs 18:57 UTC

DAG: ep_telemetry_v2 schedule: @hourly

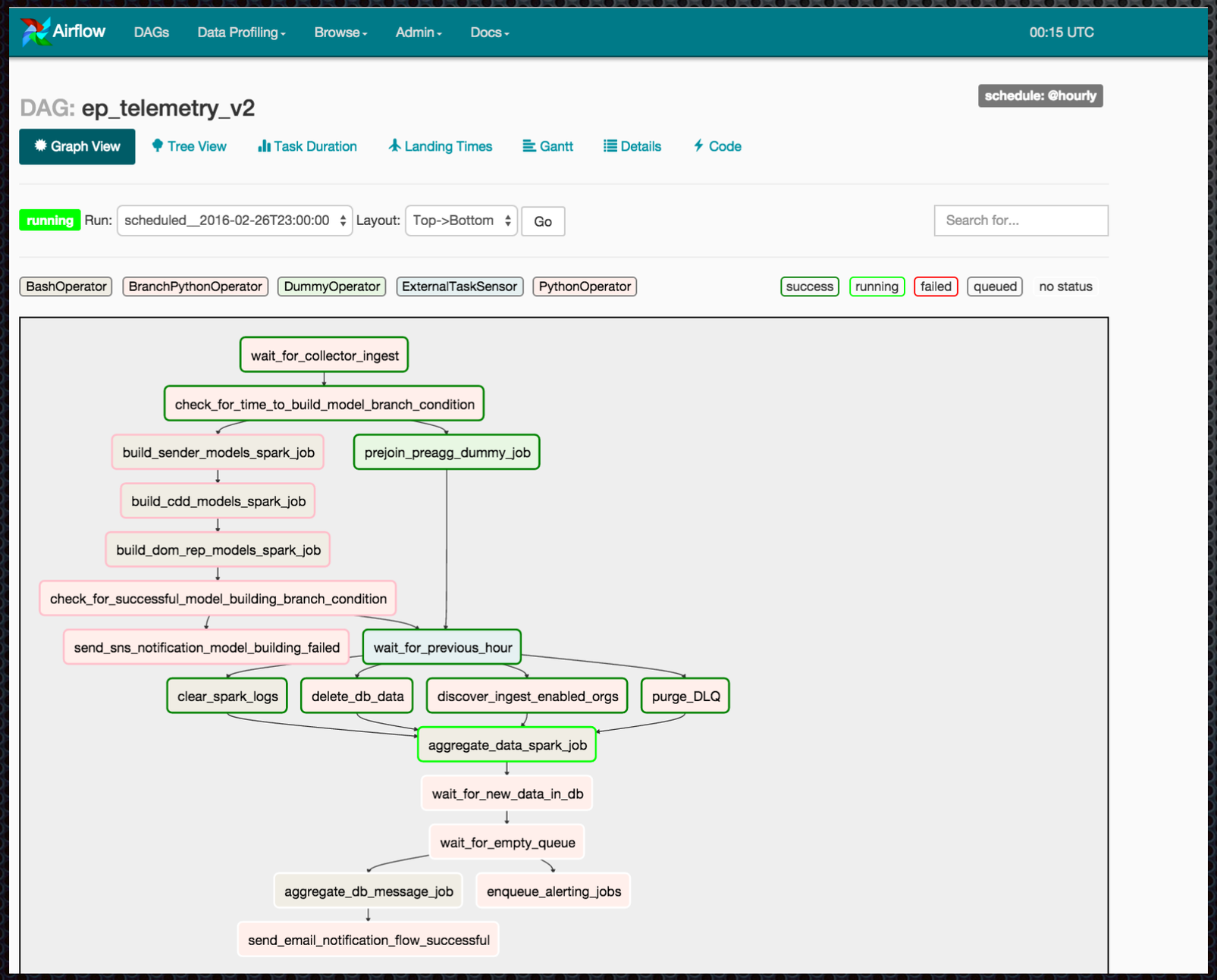
Graph View Tree View Task Duration Landing Times Gantt Details Code

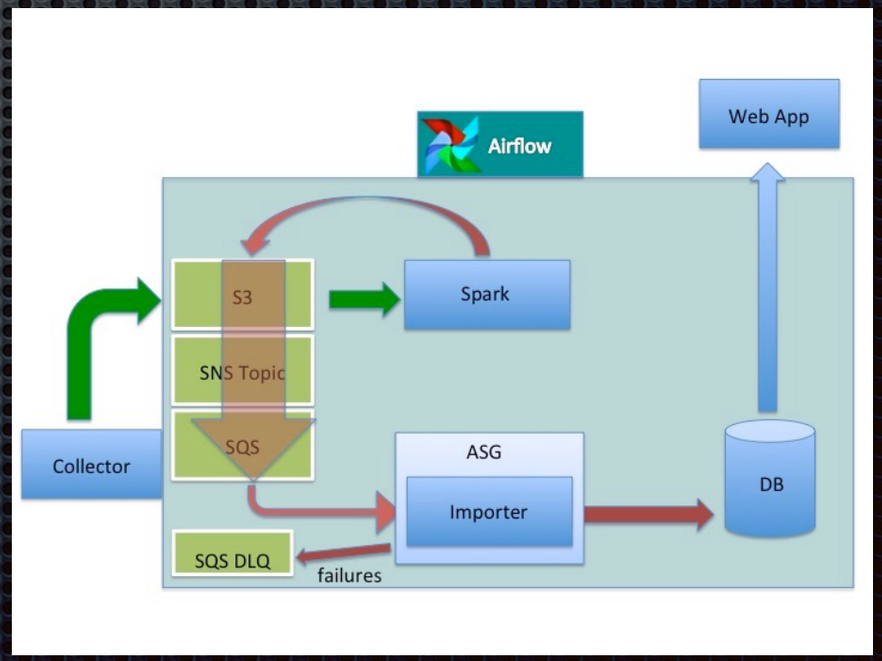
ep_telemetry_dag.py

```python
from airflow import DAG, utils
from airflow.operators import *
from datetime import date, datetime, time, timedelta
from ep_telemetry_pipeline_utils import *

now = datetime.now()
now_to_the_hour = now.replace(hour=now.time().hour, minute=0, second=0, microsecond=0)
START_DATE = now_to_the_hour + timedelta(hours=-3)
DAG_NAME = 'ep_telemetry_v2'
ORG_IDS = get_active_org_ids_string()

default_args = {
    'owner': 'sanand',
    'depends_on_past': False,
    'pool': 'ep_data_pipeline',
    'start_date': START_DATE,
    'email': [import_ep_pipeline_alert_email_dl],
    'email_on_failure': import_airflow_enable_notifications,
    'email_on_retry': import_airflow_enable_notifications,
    'retries': 1,
    'retry_delay': timedelta(seconds=120),
    'priority_weight': import_airflow_priority_weight
}
dag = DAG(DAG_NAME, schedule_interval='@hourly', default_args=default_args, sla_miss_callback=sla_a


# Operator DAG Design
# 0. Purge DLQ
# 1.a Get Collector counts for validation
# 1.b Delete db data
# 2.  Generate Parquet files -- this also consumes new upload files and moves them into archived
# 2.a. Get counts of messages by org
# 3. Run Model Building
# 4. Run Aggregate Spark Job -- this generates aggregate data from Parquet files
# 5. Check SQS for new messages -- if no messages, branch --> send email
# 6. Check the DB for new rows within exec_date - 1d <--> exec_date
# 7. Wait for SQS to drain
# 8. Send an email that the flow completed successfully

# Operator : Wait for S3 delays related to collector_ingest copies
wait_for_collector_ingest = PythonOperator(
    task_id='wait_for_collector_ingest',
    provide_context=True,
    python_callable=wait_for_collector_ingest,
    dag=dag)
```

Apache Airflow - Authoring DAGs

Airflow: Visualizing a DAG





Apache Airflow - Managing DAGs

Airflow: It's easy to manage multiple DAGs

Airflow | DAGs | Data Profiling | Browse | Admin | Docs | 00:31 UTC

DAGs

Show 10 entries

Search:

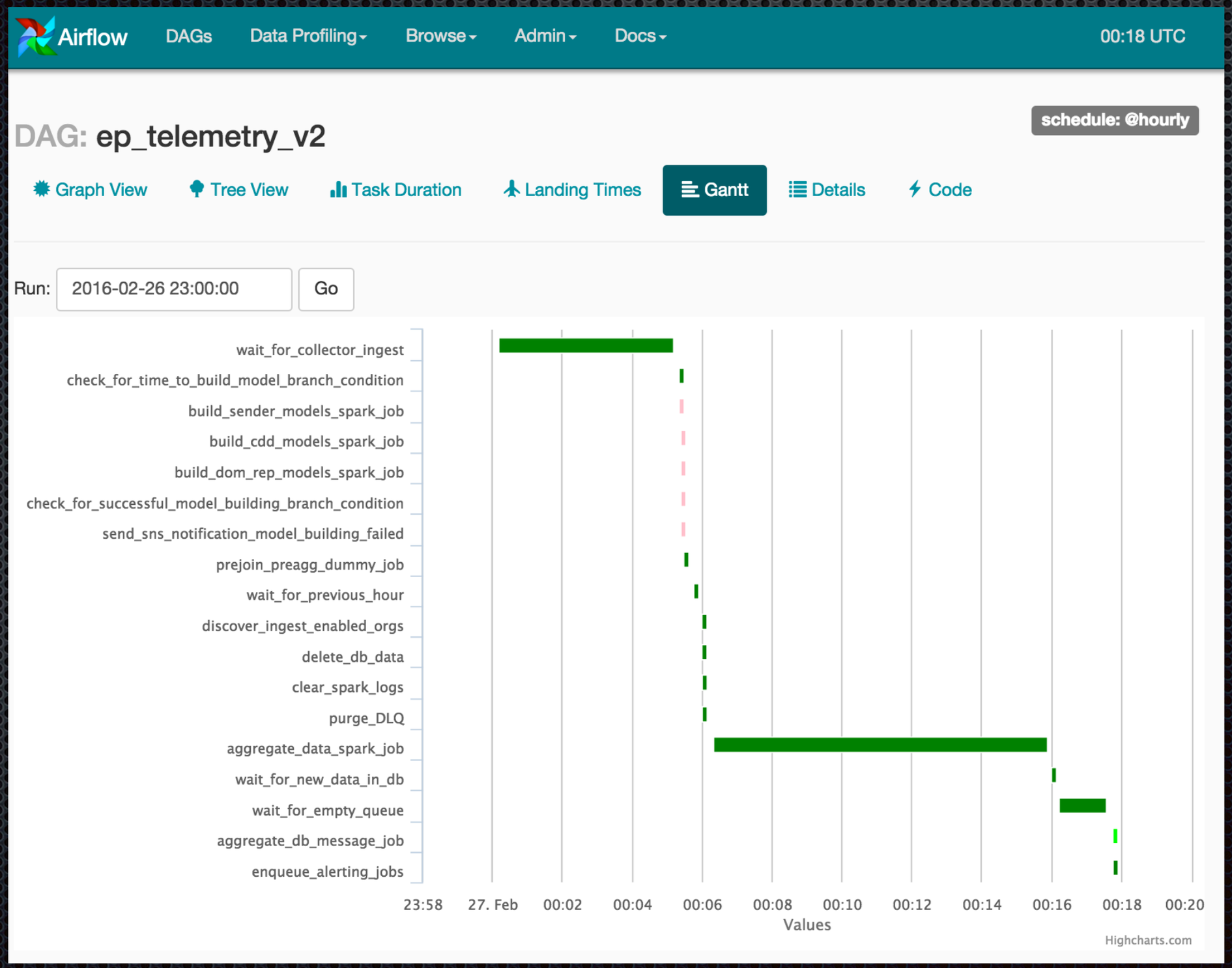
		DAG	Schedule	Owner	Statuses	Links
	On	db_backup_v1	0 4 * * *	aflury	25	
	On	emr_forwarders	0 8 * * *	kmandich	118	
	On	emr_model_building	0 1 * * *	kmandich	235	
	Off	ep_model_building_v1	0 1 * * *	sanand	253	
	Off	ep_reload_data	None	sanand	2	
	On	ep_telemetry_v2	@hourly	sanand	30671 7	
	Off	feeback_report	1 day, 0:00:00	kmandich	72	
	On	generate_spoofs	1 day, 0:00:00	kmandich	258 1 1	
	On	refresh_asn_data	0 10 * * *	kmandich	292	
	On	refresh_dns_cache	0 10 1,15 * *	kmandich		

Showing 1 to 10 of 10 entries

Previous 1 Next

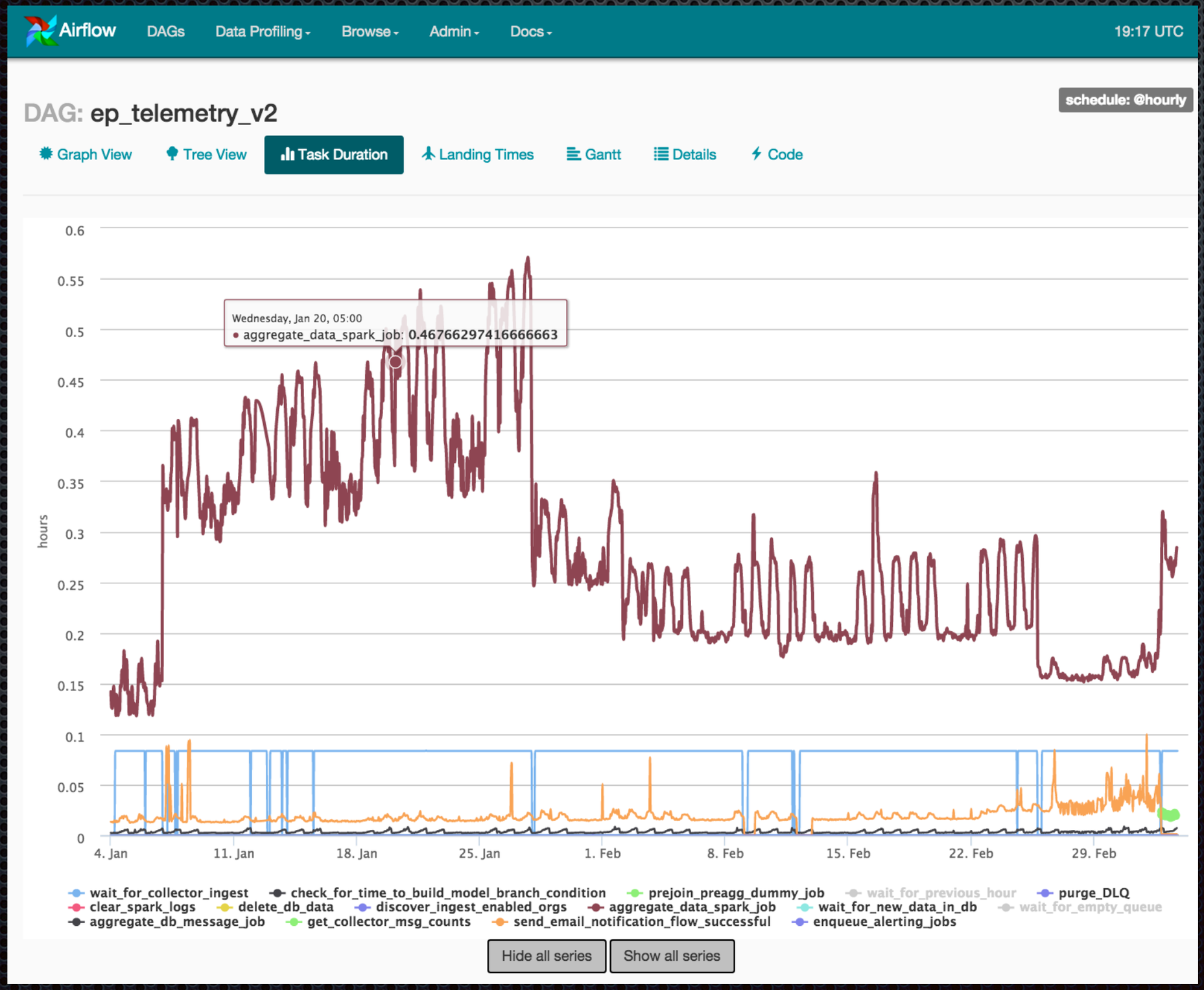
Apache Airflow - Perf. Insights

Airflow: Gantt chart view reveals the slowest tasks for a run!



Apache Airflow - Perf. Insights

Airflow: Task Duration chart view show task completion time trends!



Apache Airflow - Alerting

Slack

Agari
sid

STARRED
airflow
analysis
eng
eng-ep
ep-ops
ep-real-time-alerting
scotfree, kevin, wforr…

#ep-ops A channel with info that can help you resolve VictorOps alerts 13 Search

Today

Airflow BOT 10:00 AM
EP_STAGE - ep_telemetry_v2 SLA Miss for task `send_email_notification_flow_successful on 2016-02-26T18:00:00`

sid 10:02 AM
So, we need to catch up here.. any reason we don't want to just mark success for the many hours it is behind?

Airflow BOT 11:00 AM
EP_STAGE - ep_telemetry_v2 SLA Miss for task `send_email_notification_flow_successful on 2016-02-26T19:00:00`

Airflow BOT 5:27 PM
ep_telemetry_v2 on etl-00.ep-old.prod.agari.com completed `2016-02-03 00:00:00` with High Discrepancies

February 3rd

Airflow BOT 8:48 PM
ep_telemetry_v2 on workflow-00.ep.stage.agari.com completed `2016-02-26 03:00:00` with 1 DLQs : Sample Exception

Agari Timeline | Settings | Reports @sanand

Teams Users
On-call Engaged
Teammates All Users

Andrew Flury @aflury Enterprise Protect, everyone
Christopher Haag @chaag Enterprise Protect, everyone
Spencer Sun @ssun everyone, healthchecks.io

post an update...

More Info Feb 26, 2016 17:00:02 PST

TYPE	airflow
STATE	INFO
TIME	Feb 26, 2016 17:00:02 PST
ENTITY	airflow/telemetry/SLA_miss
MSG	EP_STAGE - ep_telemetry_v2 SLA M...

More Info Feb 26, 2016 16:00:04 PST

TYPE	airflow
STATE	INFO
TIME	Feb 26, 2016 16:00:04 PST
ENTITY	airflow/telemetry/SLA_miss
MSG	EP_STAGE - ep_telemetry_v2 SLA M...

More Info Feb 26, 2016 15:41:40 PST

TYPE	airflow
STATE	INFO

All Yours Team

Triggered : 0
Acked : 0
Resolved

#102 ALERT - Prod EP Pipeline Discrepancy: workflow-00.ep.prod.agari.com/agari.log [Agari Data Inc]

Time	Feb 26, 2016 13:...
Email	ALERT - Prod EP ...
State	OK
Resolved By	aflury

#101 airflow/telemetry/SLA_miss

Time	Feb 26, 2016 13:0...
API	airflow/telemetry...
State	INFO
Resolved By	SYSTEM

VictorOps Connected Download the mobile app → Get Help

Apache Airflow - Correctness

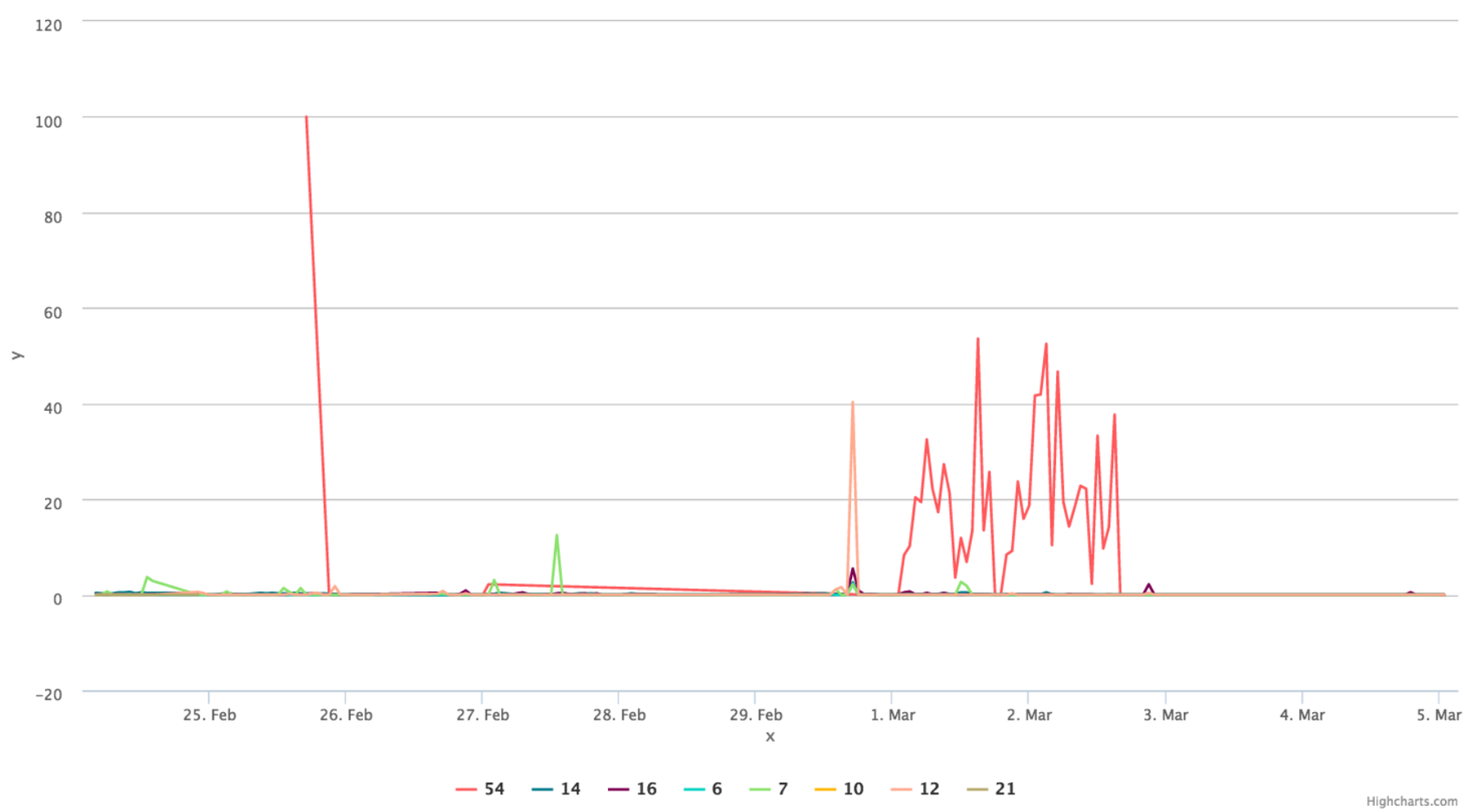
Airflow DAGs Data Profiling Browse Admin Docs 03:07 UTC

Hourly Discrepancy Percentage (By Org)

SQL

```
SELECT organization_id as series, execution_date as x , discrepancy_percentage as y FROM telemetry_data_discrepancy where execution_date
```

Chart



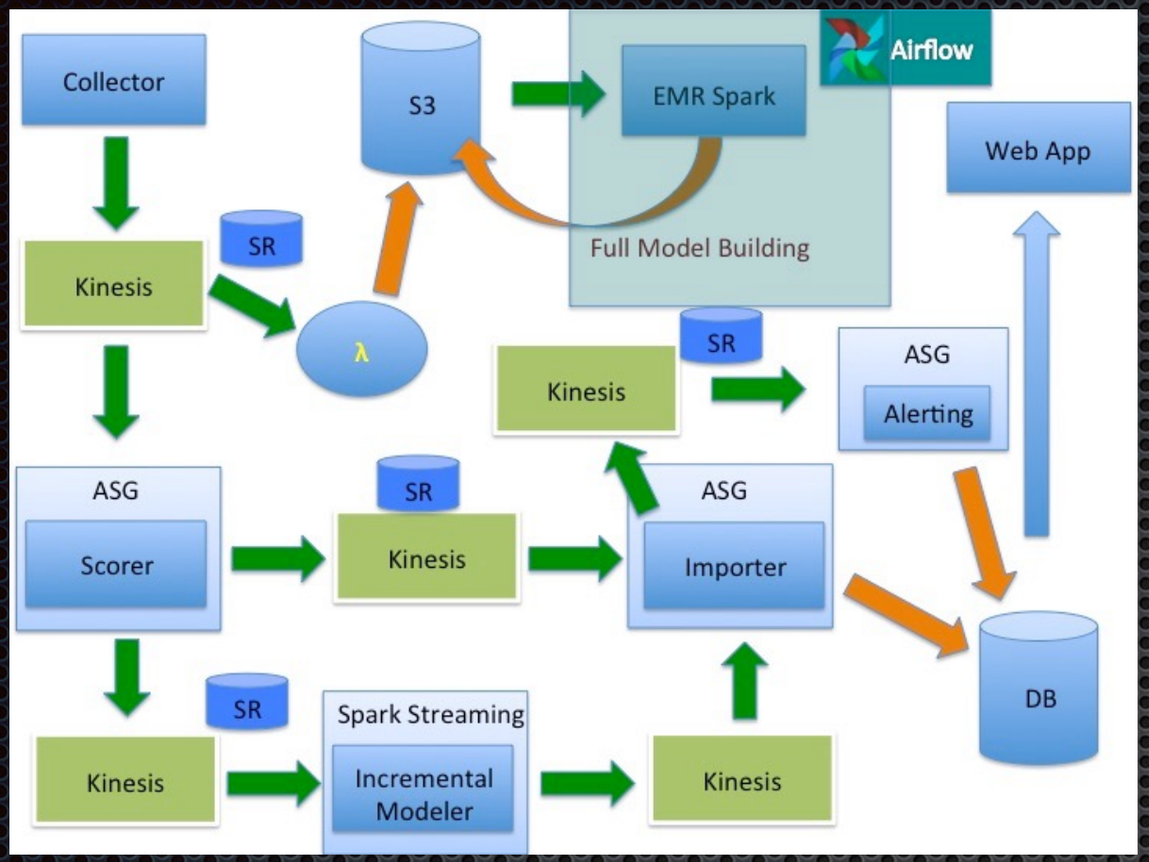
Desirable Qualities of a Resilient Data Pipeline

Correctness	Operability
Timeliness	Cost

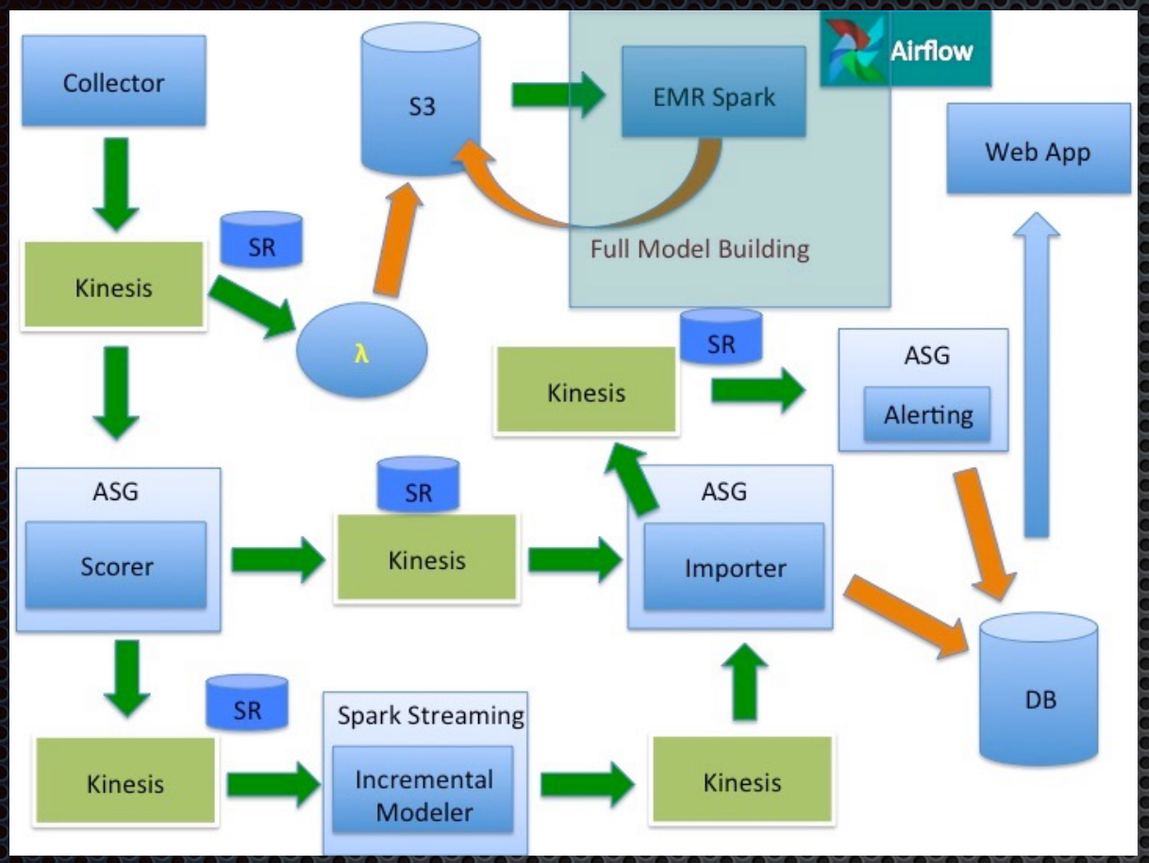
Near-Real Time Data Pipelines

Stream Processing @ Agari

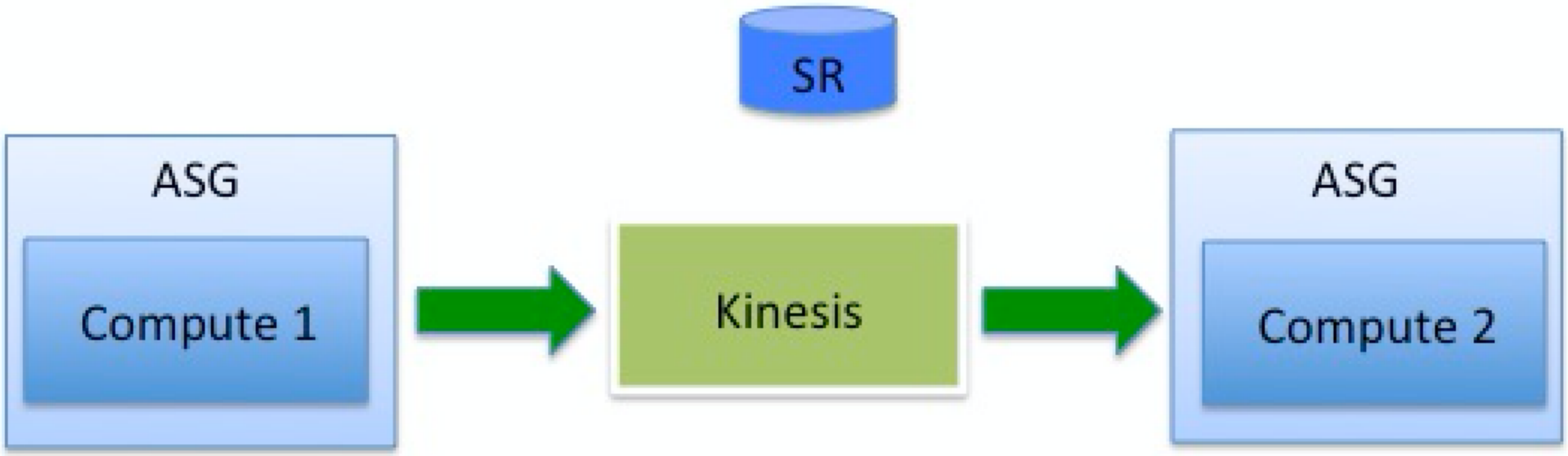
NRT Architecture



NRT Architecture



NRT Architecture



- The Architecture is composed of repeated patterns of :
  - ASG-based compute
  - Kinesis streams (i.e. AWS' managed "Kafka")
  - Lambda-based Avro Schema Registry

Avro Schema Registry

Avro Schema Storage

What is Avro?

A self-describing (schema'd) serialization format

```
{
  "namespace":"com.agari.ep.collector.model",
  "type":"record",
  "doc":"This Schema describes the server-side configuration of Agari's Enterprise-Protect Collector",
  "name":"collector_config",
  "fields":[
    {"name": "email_log_enabled", "type": "boolean"},
    {"name": "email_log_interval_seconds", "type": ["int", "null"]},
    {"name": "email_log_bucket_name", "type": "string"},
    {"name": "phone_home_interval_seconds", "type": "int"},
    {"name": "phone_home_sns_topic_ARN", "type": "string"},
    {"name": "config_pull_interval_seconds", "type": "int"},
    {"name": "receiver_netblocks", "type": ["null", {"type": "array", "items": "string"}]},
    {
      "name": "connecting_ip",
      "type": [
        "null",
        {
          "type": "record",
          "name": "connecting_ip_record",
          "fields": [
            {
              "name": "received_header_index",
              "type": "int"
            },
          ]
        ......
```

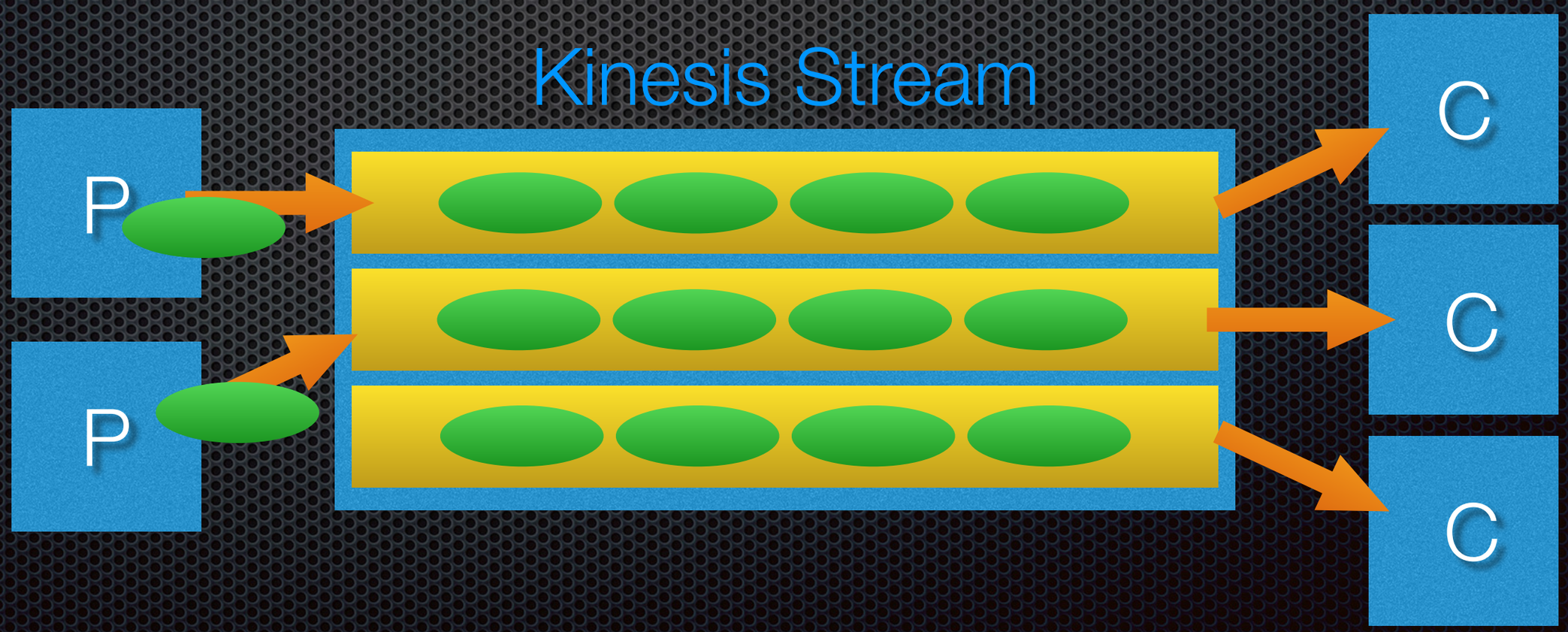
What is Avro?

- Typically, the schema is stored in the same file as the data it represents
  - In HDFS, where files are typically large, the schema overhead is negligible

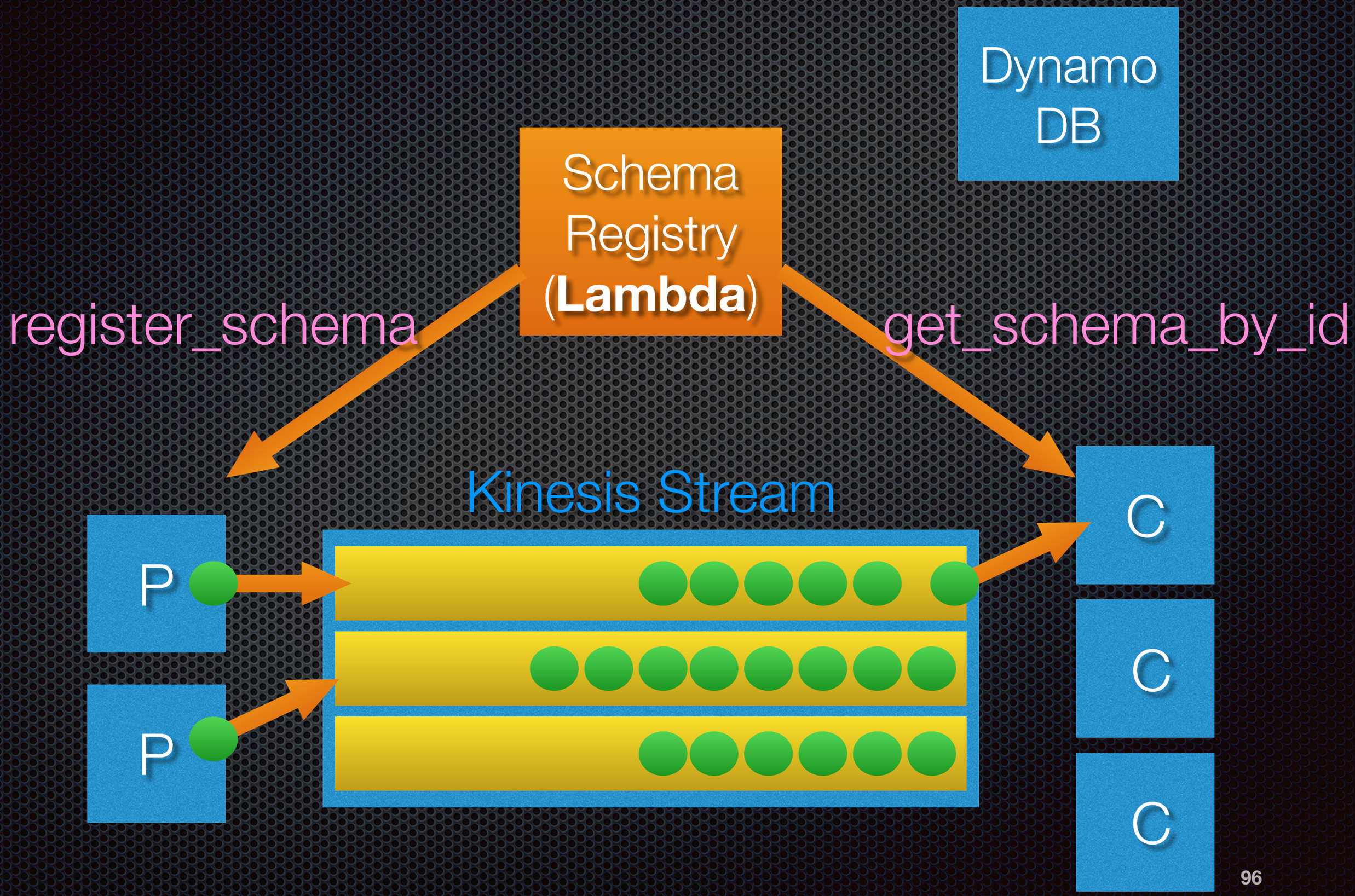
Schema Registry

- In streaming, where each record may be sent individually, the schema will be the majority of the data transmitted!
-  is a fat message

Can we be smarter?



Schema Registry



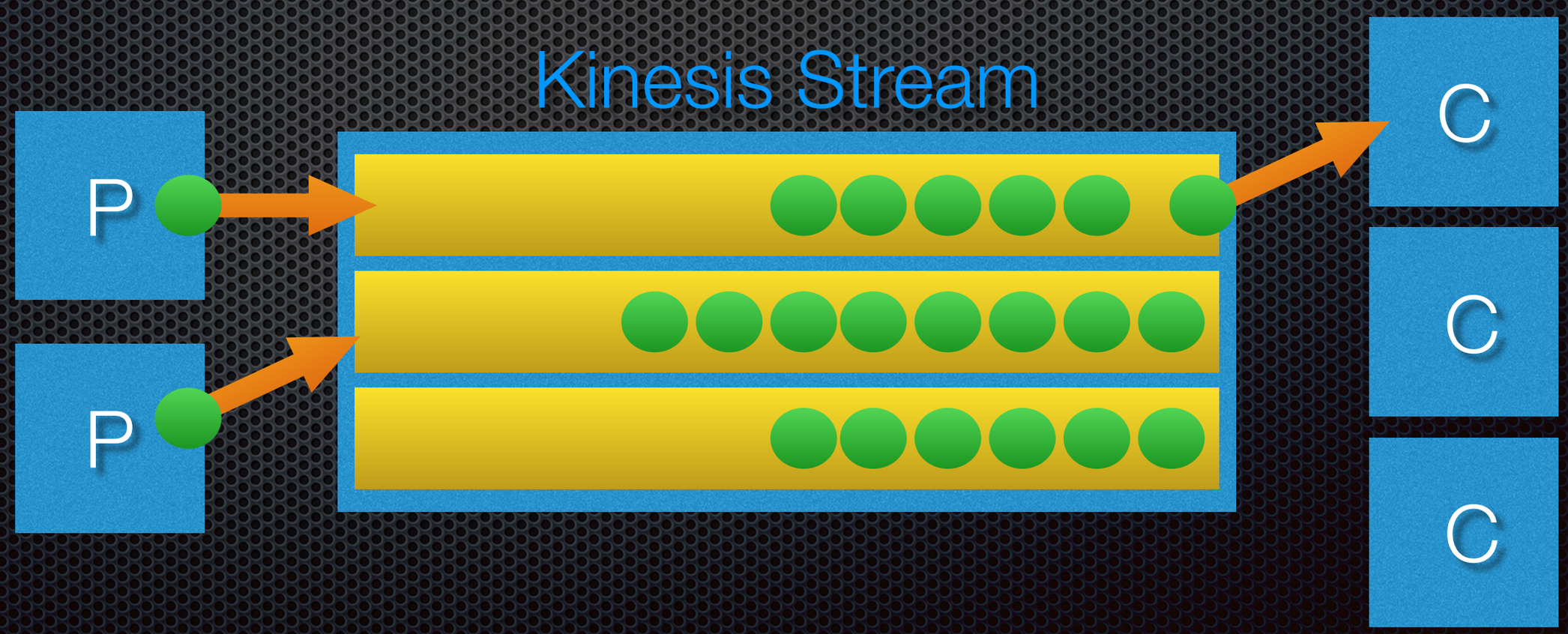
What is AWS Lambda?

- AWS-hosted code execution environment (Python, Node, Java, Ruby)
- You upload some code & specify a simple memory and CPU profile (e.g. medium CPU, 256 GB memory)
- The code will get a new version (e.g. v2)
- Code Rollback as easy as setting $LATEST alias to a previous version (e.g. $LATEST=v1)

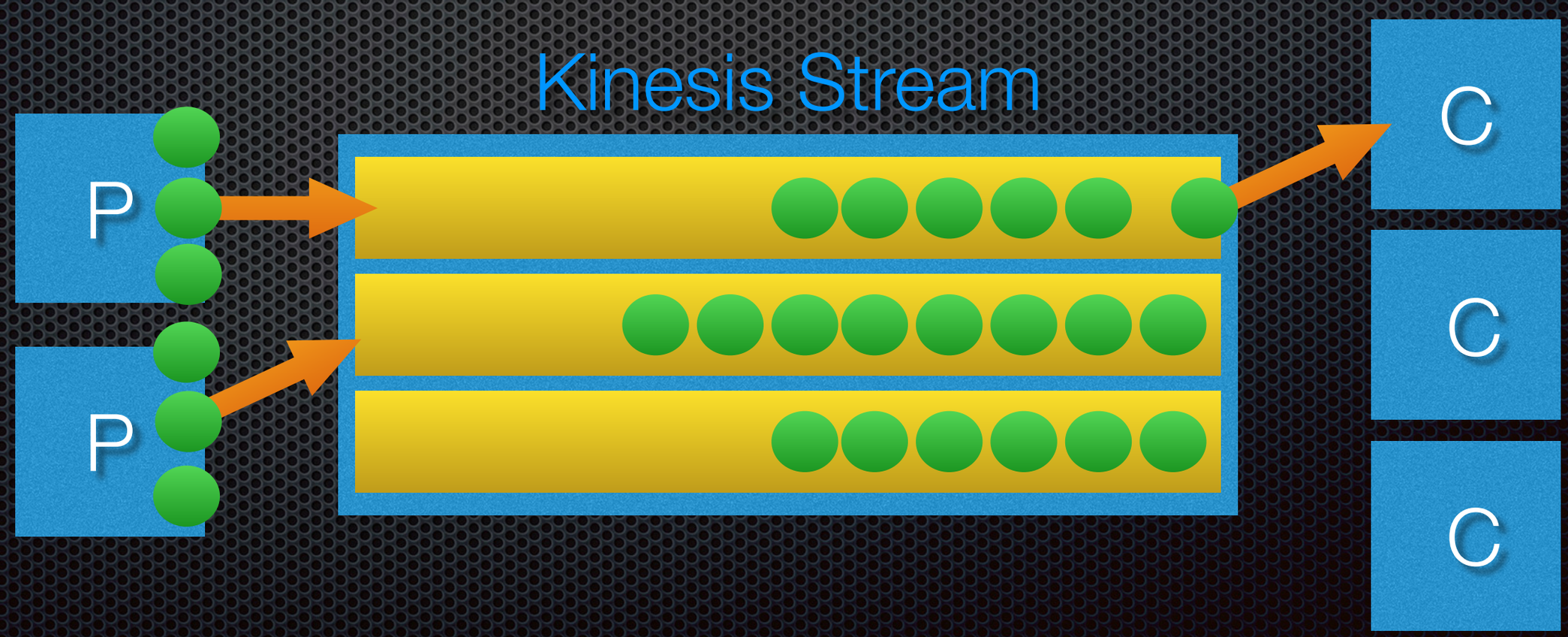
Elastic Stream Processing

How Do We Handle Increasing Traffic?

Elastic Stream Processing

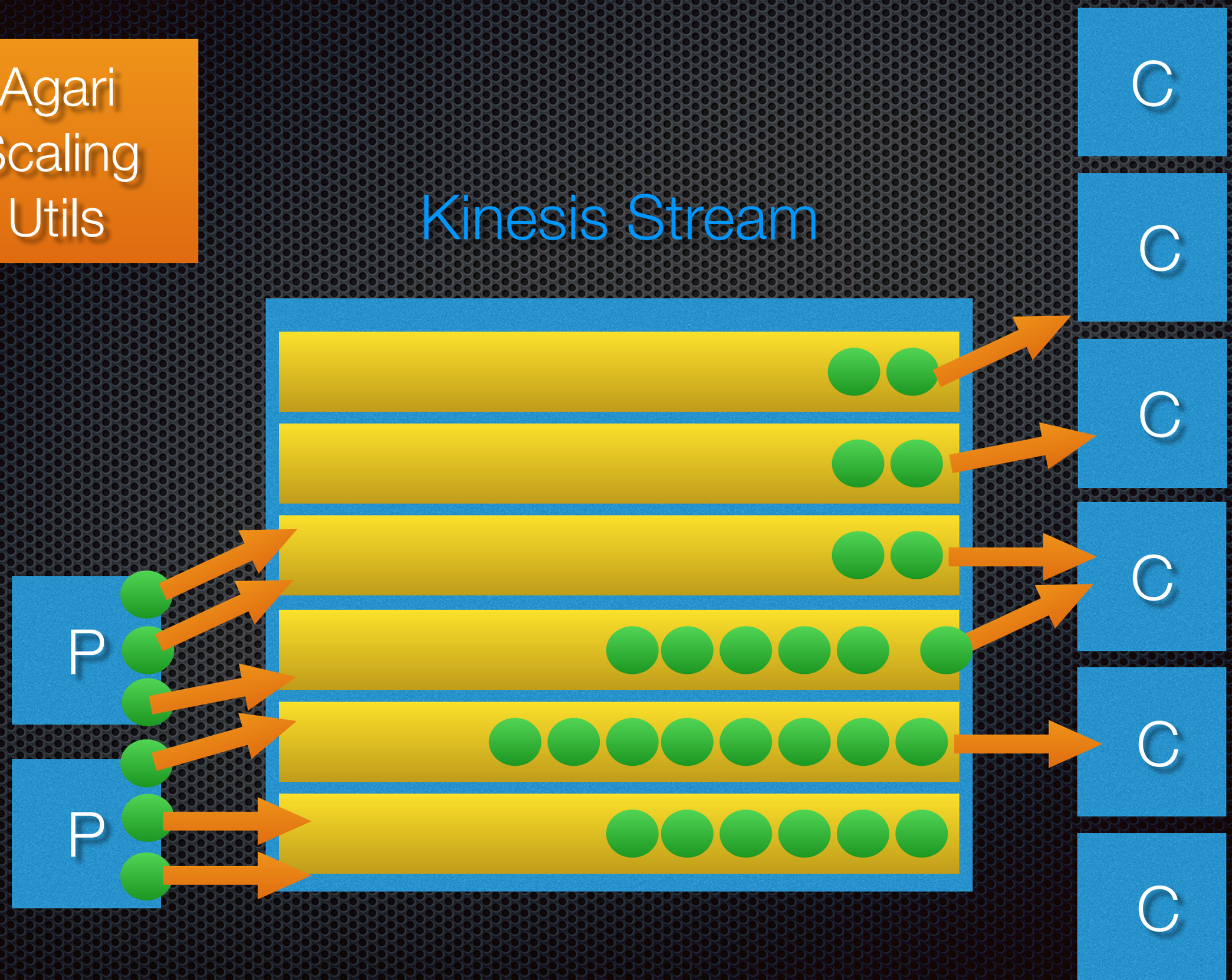


Elastic Stream Processing



Elastic Stream Processing

Agari Scaling Utils



Open Source Plans

In late Q2/early Q3, we plan to open-source our cloud tools for :

- Avro Schema Registry &
- Agari (Kinesis+ASG) scaling tools

To be notified, follow @AgariEng & @r39132

Questions? (@r39132)