Teaching an old DAG new tricks

...

Migrating a decade old pipeline to Airflow

Outline

Cloud native deployment

- Cloud native deployment
- Multi-repo DAG management
- Manage Airflow Variables with code through Terraform
- Airflow monitoring best practices with Datadog and Pagerduty

Airflow Migration

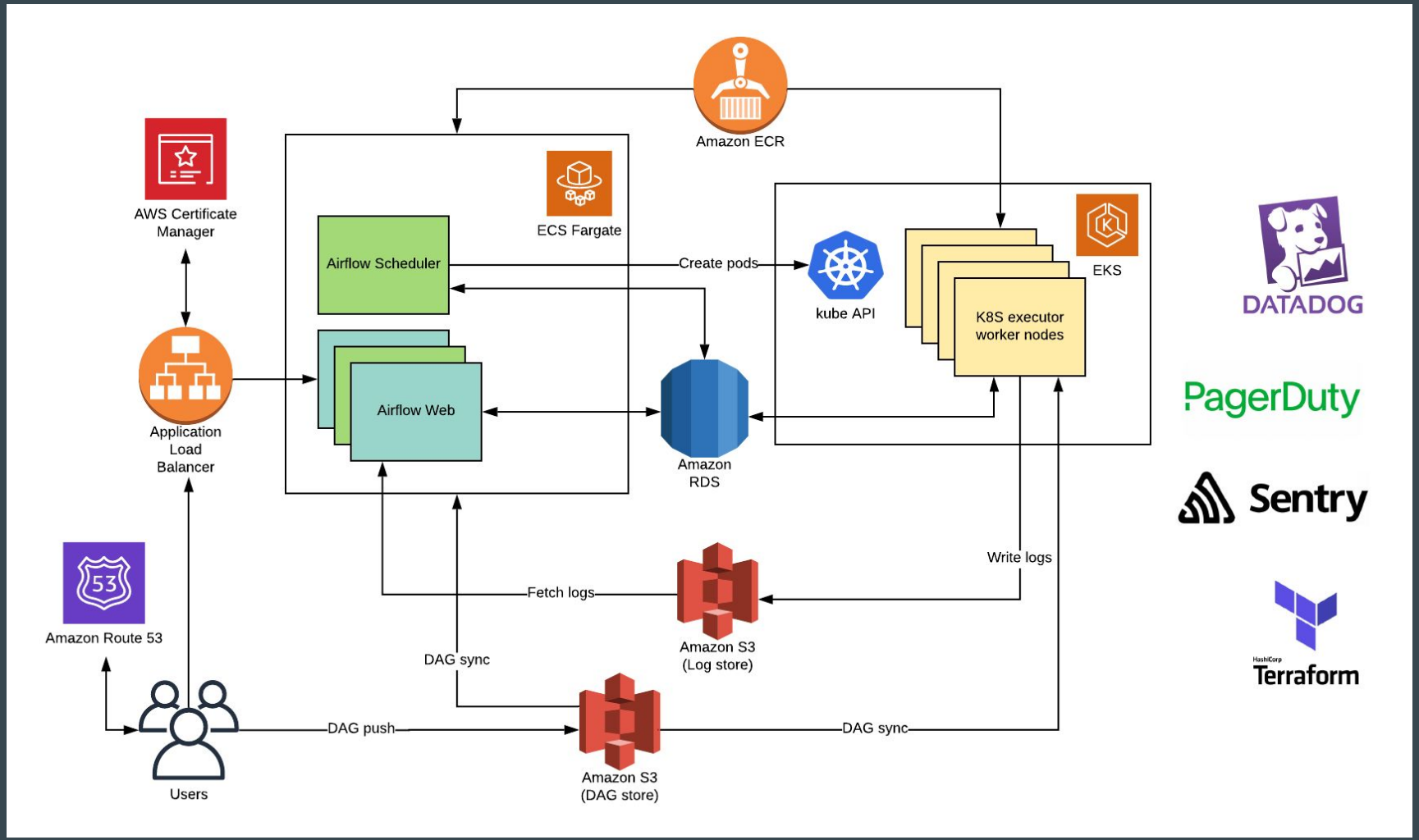
- Simulate production run to surface issues early
- Plan and execute with incremental deliverables

Scribd is moving to the cloud

https://tech.scribd.com/blog/2019/building-the-library.html

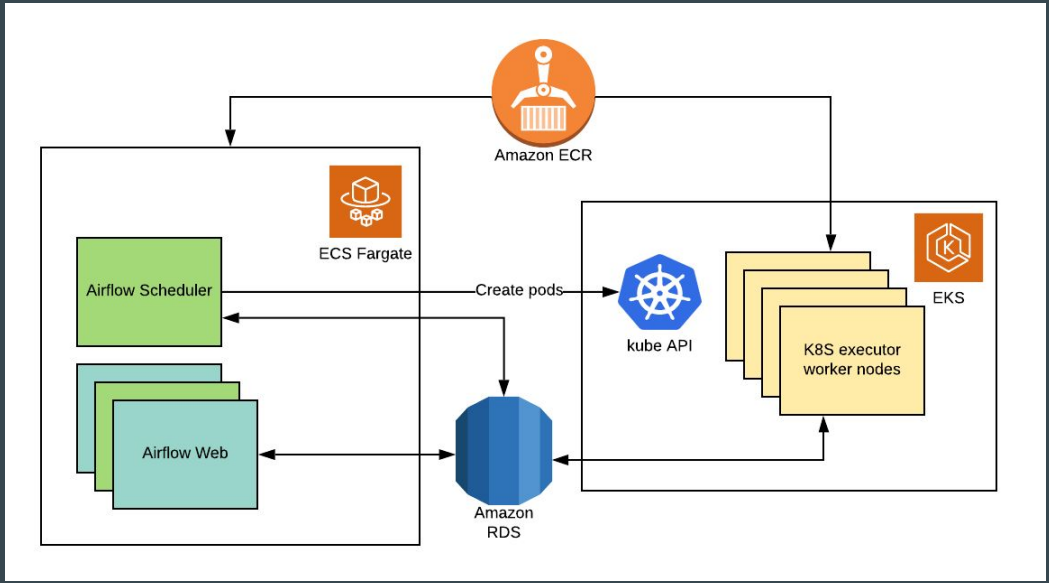
Cloud native Airflow

- Use managed service whenever possible
- Separation of stateless compute and stateful data store
- Separation of infrastructure (Airflow cluster) and application (DAG)
- Separation of environments
- Automate Infrastructure provisioning with code
- Running on development branch of Airflow for latest improvements and bug fixes


Amazon ECR
AWS Certificate Manager
ECS Fargate
Airflow Scheduler
Create pods
kube API
EKS
K8S executor worker nodes
Application Load Balancer
Airflow Web
Amazon RDS
Amazon Route 53
Fetch logs
Write logs
Amazon S3 (Log store)
DAG sync
Users
DAG push
Amazon S3 (DAG store)
DAG sync
DATADOG
PagerDuty
Sentry
HashiCorp
Terraform

ECS and EKS?!

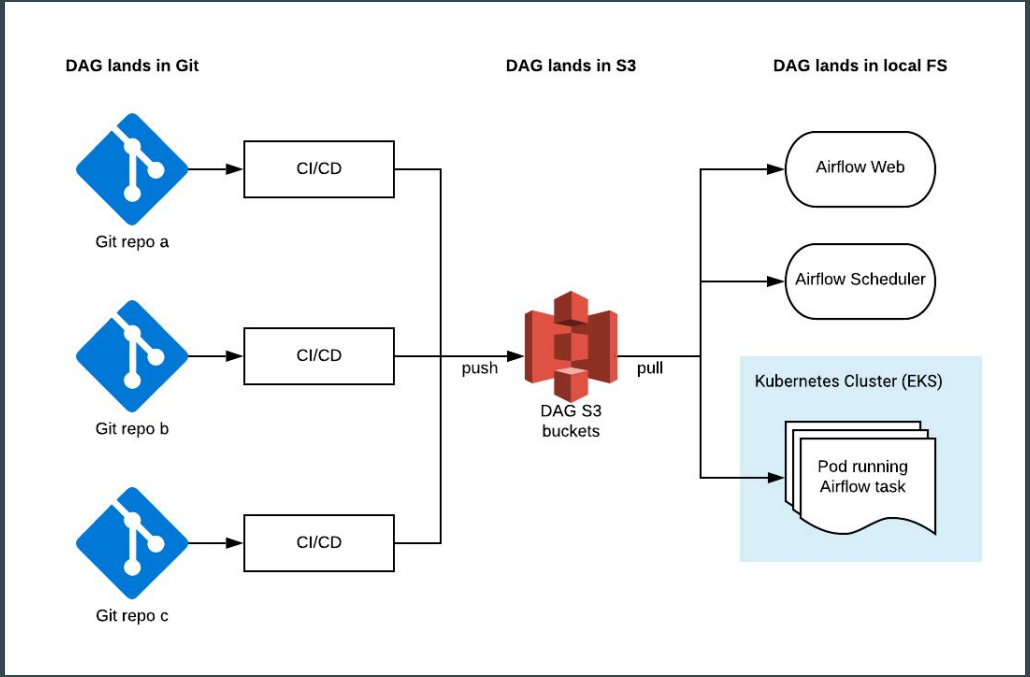
- Different crash zones
- Reduce maintenance burden with ECS fargate



Out of cluster Kubernetes executor support for EKS

- Kubernetes Python client doesn't work well with EKS
- API token generated by aws-iam-authenticator expires about every 14 minutes
- Python client fix backported to Airflow: https://github.com/apache/airflow/pull/5731

Develop DAGs across multiple repos



https://tech.scribd.com/blog/2020/breaking-up-the-dag-repo.html

DAG sync daemon

- Background daemon written in Golang with small CPU and memory footprint
- Single binary ready to run in any environment
- File list and checksums are cached in memory to minimize network and disk IO
- DAG release gets picked up within seconds
    - Future plan to use S3 event notification to make it near realtime
- Expose operational metrics as prometheus format through HTTP
    - DAG Update/Delete/Create statistics
    - Time spent on DAG sync
    - Daemon uptime

Project Github: https://github.com/scribd/objinsync

Manage Variables with Terraform

We use variables to templatize a lot of things

- IAM roles for Databricks clusters
- Glue catalog id
- EC2 Instance profile ARN
- Application Jar release version
- ...

```
{"assume_role_arn":"arn:aws:iam::1234567:role/automated
-job-role","glue_catalogid":"2234567","instance_profile
_arn":"arn:aws:iam::3234567:instance-profile/foo","inst
ance_profile_arn":"arn:aws:iam::4234567:instance-profil
e/databricks-jobs-dev-profile"}
```

Airflow Terraform Provider

```
locals {
  team_a_remote_state = data.terraform_remote_state.team_a.outputs
  dev_vars = {
    "team_a_cluster" = {
      "assume_role_arn"      = local.team_a_remote_state.databricks_role_arn,
      "glue_catalogid"       = local.team_a_remote_state.glue_catalogid,
      "instance_profile_arn" = local.team_a_remote_state.databricks_instance_profile_arn,
    }
  }
}

provider "airflow" {
  variables_output_path = "development.variables.json"
}

resource "airflow_variable" "development" {
  for_each = local.dev_vars
  key      = each.key
  value    = jsonencode(each.value)
}
```

Airflow Terraform Provider

- Project Github: https://github.com/houqp/terraform-provider-airflow
- Experimental branch using Airflow Go client:
  - https://github.com/houqp/terraform-provider-airflow/tree/openapi
  - https://github.com/apache/airflow-client-go/pull/1

Monitor Airflow with Datadog

Datadog agent as sidecar container within ECS

```
datadog_container = {
  name      = "datadog-agent",
  image     = "datadog/agent:latest",
  essential = true,
  environment = [
    { name  = "DD_API_KEY",
      value = var.datadog_api_key
    },
    { name  = "DD_TAGS",
      value = "env:${local.env} application:airflow"
    },
    { name  = "DD_DOGSTATSD_TAGS",
      value = "[\"env:${local.env}\", \"application:airflow\"]"
    },
    { name  = "ECS_FARGATE",
      value = "true"
    },
  ]
  portMappings = [
    { protocol      = "tcp"
      containerPort = 8125
    }
  ]
```

Statsd config for scheduler

```
# monitoring
{ name  = "AIRFLOW__SCHEDULER__STATSD_ON"
  value = "True"
},
{ name  = "AIRFLOW__SCHEDULER__STATSD_HOST"
  value = "127.0.0.1"
},
{ name  = "AIRFLOW__SCHEDULER__STATSD_PORT"
  value = "8125"
},
{ name  = "AIRFLOW__SCHEDULER__STATSD_PREFIX"
  value = "airflow"
},
```

Monitor Airflow with Datadog

- Synchronize ALB, RDS, S3, ECS and EKS Cloudwatch metrics to Datadog using Terraform (https://github.com/scribd/terraform-aws-datadog)

HashiCorp Terraform | Registry

Search for modules

Browse Publish Sign-in

aws

datadog

AWS

Terraform module for setting up AWS Datadog integration

Version 1.2.0

Published July 1, 2020 by scribd

Module managed by jim80net

Total provisions: 1,247

Source Code: github.com/scribd/terraform-aws-datadog (report an issue)

Examples

Provision Instructions

Copy and paste into your Terraform configuration, insert the variables, and run `terraform init` :

```
module "datadog" {
  source  = "scribd/datadog/aws"
  version = "1.2.0"
  # insert the 1 required variable here
}
```

Airflow

Edit Widgets +

1h The Past Hour

Search... | dev

$env dev

Overview 18

Scheduler 19

Web 11

RDS 28

DAG sync 6

ECS 4

EKS 13

S3 2

Airflow

Edit Widgets

1h The Past Hour

Search... | dev | $env dev

RDS DML throughput

RDS select/s

RDS select latency

RDS insert/s

RDS insert latency

RDS delete/s

RDS delete latency

RDS update/s

RDS update latency

RDS billed bytes used

RDS billed read IO per second

RDS billed write IO per second

DAG sync 6

ECS 4

EKS 13

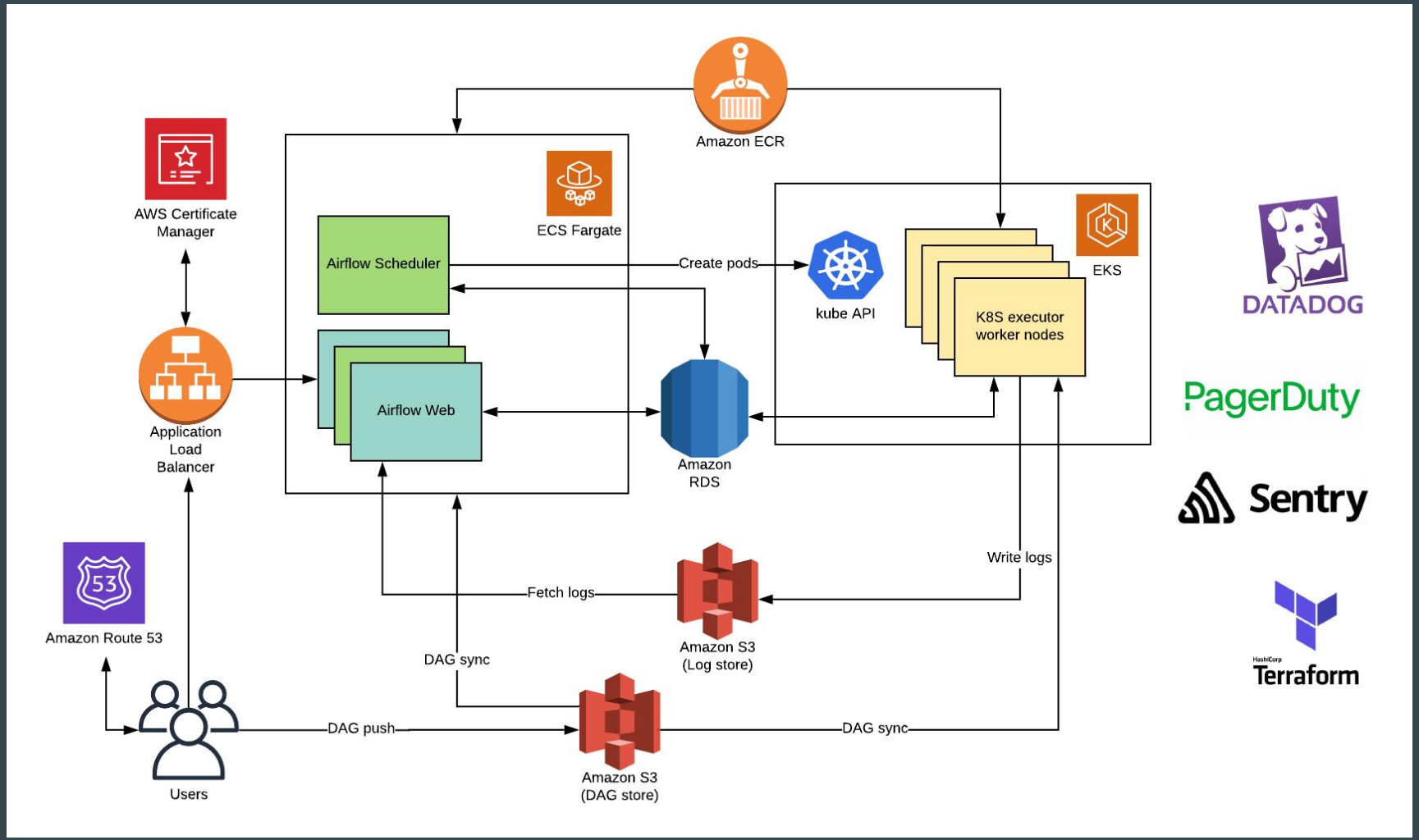
S3 2

Incident response with Pagerduty

- Paging for infrastructure incidents
  - Through Datadog monitors
- Paging for application incidents
  - Pagerduty event emitted from Airflow for
    - Task failures
    - SLA misses
    - Adhoc events

Integration with Pagerduty

	When **all** conditions are met • `payload.group` contains `pdservice:airflow` • `payload.group` contains `env:prod`	• Route to Airflow • Then stop processing
	When **all** conditions are met • `payload.group` contains `env:preprod` • `payload.group` contains `pdservice:airflow`	• Route to Airflow • Set severity to warning • Then stop processing
	When **all** conditions are met • `payload.group` contains `env:dev` • `payload.group` contains `pdservice:airflow`	• Route to Airflow • Set severity to warning • Then stop processing
	When **all** conditions are met • `payload.source` equals `dag_nightly`	• Route to Nightly • Then stop processing


Amazon ECR
AWS Certificate Manager
ECS Fargate
Airflow Scheduler
Create pods
kube API
EKS
K8S executor worker nodes
Application Load Balancer
Airflow Web
Amazon RDS
Amazon Route 53
Fetch logs
Write logs
Amazon S3 (Log store)
DAG sync
Users
DAG push
Amazon S3 (DAG store)
DAG sync
DATADOG
PagerDuty
Sentry
HashiCorp
Terraform

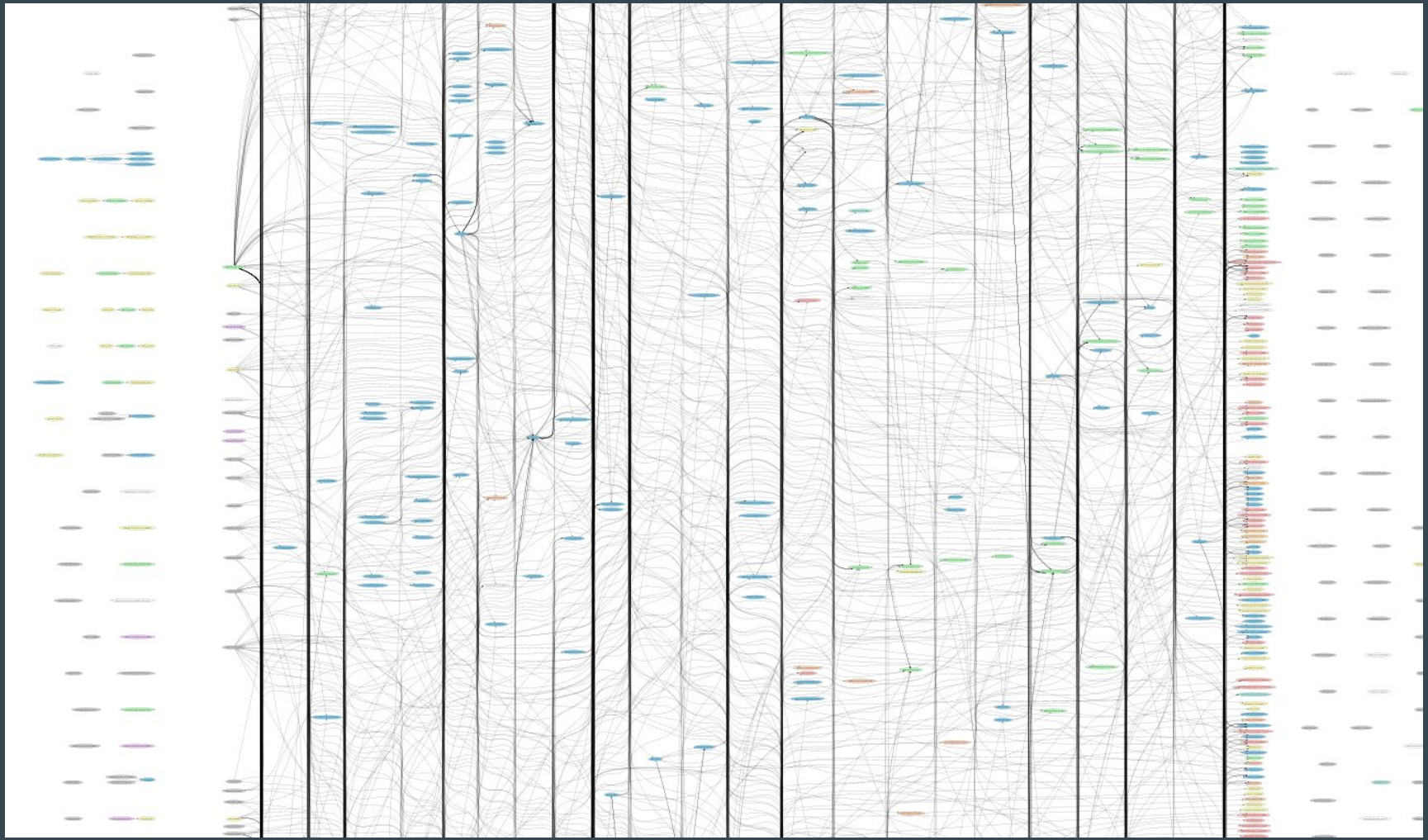
Migration

A decade old data pipeline

- In house workflow orchestration system called Datapipe
- First commit dates back to 2010
- 1500+ tasks with 1200+ of them in a single DAG
- Depend on features not supported by Airflow out of the box
- Data storage: HDFS, S3, Kafka, MySQL, Redis, ES
- Compute: Hive, Implala, Spark 1, Spark 2, Ruby

A brave new world

- Orchestrated through Airflow
- Data storage: S3 with Delta lake, Kafka, RDS, ElasticCache
- Compute: Spark 3 (Databricks)



Simulate production run early

- Automation to transpile Ruby DSL to Airflow DAG
  - Each task is a dummy operator that sleeps to simulate a run
  - Task sleep time calculated based off Avg runtime recorded by in-house system
- Scheduler was able to handle this DAG out of the box

503

Service Unavailable

How to render a 1500+ tasks DAG in Airflow

- It takes a long time to generate and render a 100MB page (tree view)
- Optimizations:
  - Avoid serialize the whole ORM object
  - Remove unnecessary if statements
  - Serialize JSON as string to be parsed with JSON.parse in the frontend
  - ...
  - https://github.com/apache/airflow/pull/7492
- Reduced page size by more than 10X
- Improved page load time by 5X

Airflow DAGs Security Browse Admin Docs About 2020-07-10, 00:28:26 UTC

To the cloud, with incremental deliverables

- Incremental daily sync for new data lake in S3
  - Wrote a mini Python parser in Ruby
- Move ad-hoc read-only interactive queries
- Trim the dependency graph
- Move output phase of the pipeline to unblock external services
- Move remaining of the pipeline

About me (QP Hou)

Engineer at Scribd's Core Platform team

New Airflow committer

Maintainer and contributor of many other open-source projects

You can find me at:

- Airflow slack and mailing list
- https://about.houqp.me

Closing

- Truly a team effort within different engineering teams at Scribd
  - Driven by Platform Engineering
    - Core platform team
    - Data engineering team
- Embrace the open-source community
  - 41 PRs merged into upstream Airflow, many more to come
- Openings: https://www.scribd.com/about/engineering