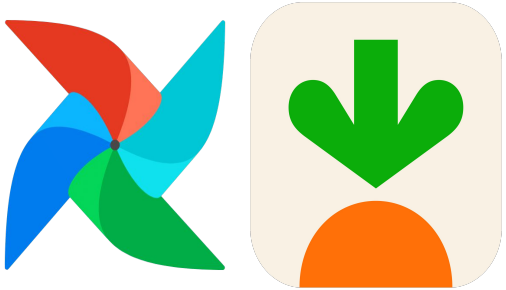
Scaling Airflow for Data Productivity at Instacart



Anant Agarwal

Software Engineer, Instacart

About Me

- Been with Instacart ~4 years
- Worked across different platform teams building systems, frameworks and libraries along the way.
- Live in SF Bay Area
- https://www.linkedin.com/in/anantag/

instacart

Agenda

- Part 1 – Instacart's Airflow Journey
- Part 2 – How we use Airflow at Instacart
- Part 3 – Key Takeaways

instacart

Instacart's Airflow Journey

instacart

State of Airflow (earlier)

- A combination of legacy Airflow clusters, used by multiple teams.
- No central team to manage them!
- Older versions
- Limited visibility and monitoring
- Scalability issues.

State of Airflow (now)

1. One Central Airflow Cluster
    a. Managed by the Data Infra team
    b. Airflow 2.7.3 with upgrades every 6 months
    c. Infra as Code!
    d. Support for multiple use-cases and remote executors.
    e. High auditability with compliance controls.
2. All new workloads and use-cases onboard to the Central cluster.
3. Migration of workloads on legacy cluster - one set of workloads every quarter.

State of Airflow (future)

1. One Central Airflow cluster across the company.
2. New clusters only on a case by case basis.
3. Legacy clusters removed and all workloads migrated.

Use Cases powered by Central Airflow at Instacart

dbt pipelines

Snowflake <> Delta pipelines

GSheets, Jira, Incidents etc. data movement

Non-dbt pipelines

How we use Airflow at Instacart

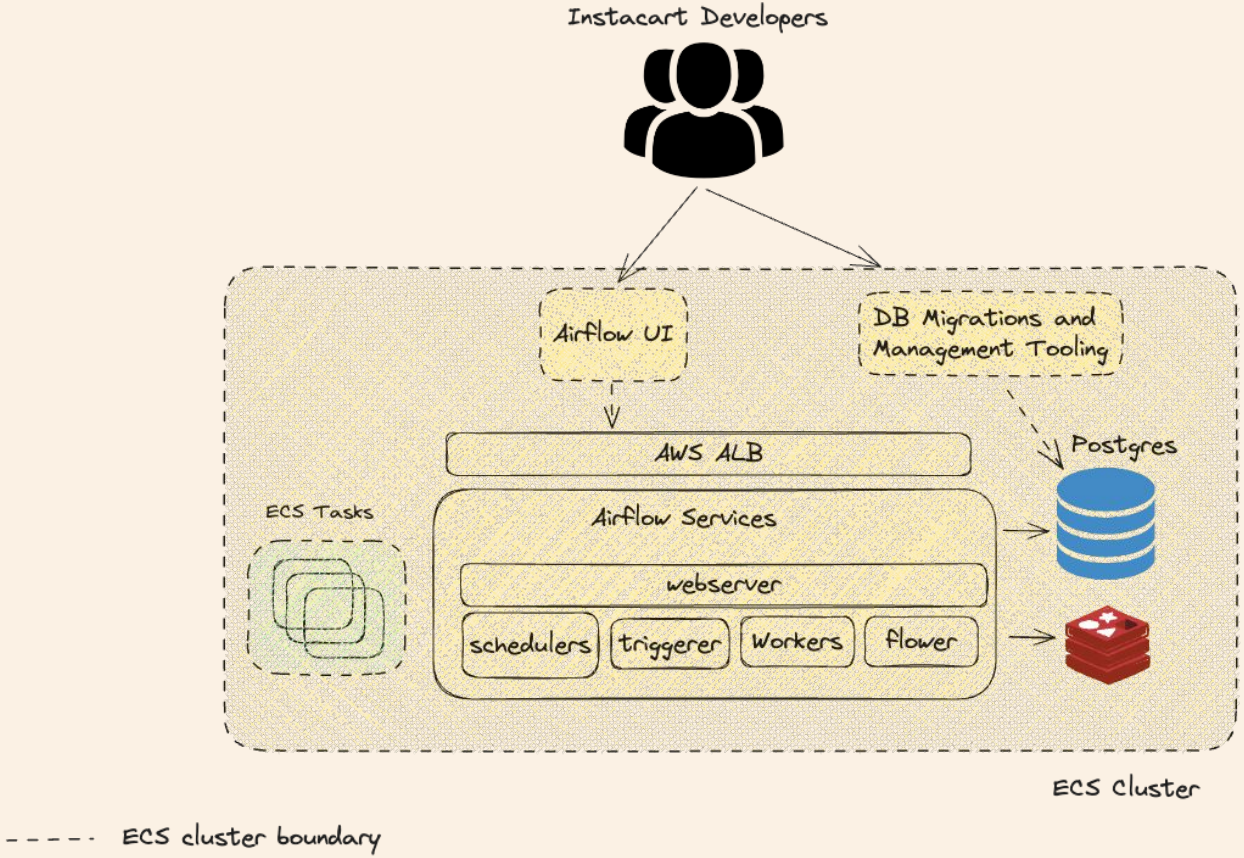
instacart

Central Airflow Usage by the numbers

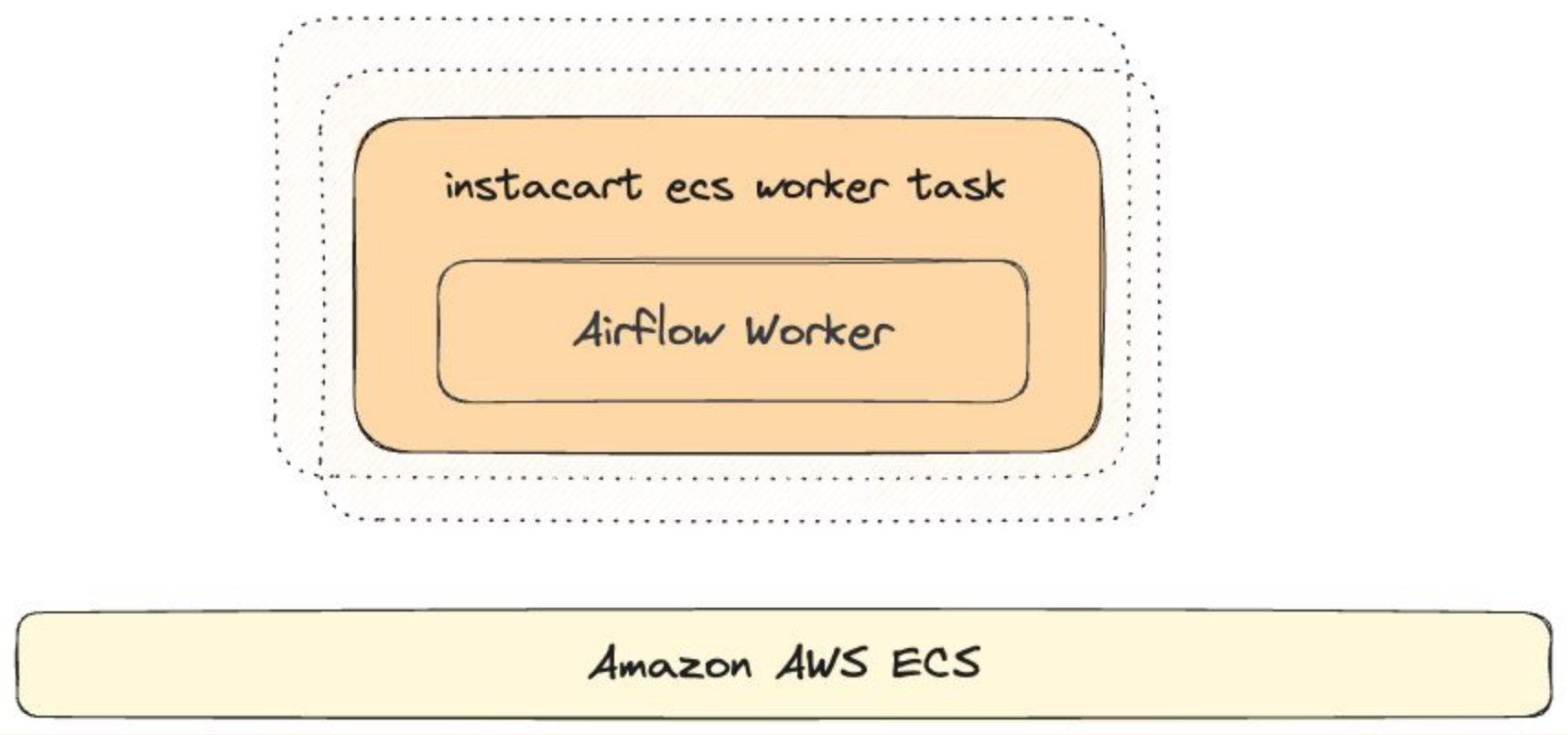
- ~3 years in production
- ~2200 DAGs
- ~16M tasks/month
- <3 critical incidents over the past 2 years
- Snowflake, Databricks, dbt, Custom ECS Operator
- Airflow version upgrades every 6 months.
- 99.5% task success rate
- Separate Production and Development environments

instacart

Airflow Deployment at Instacart



Airflow Worker Deployment at Instacart



Abstract away DAG Configurations

```yaml
owner: anant.agarwal
start_date: 2022-01-04
schedule_interval: 0 * * * *
catchup: false
description: Example job that fail due to failed pretest condition
totem_system: data-productivity-data-pipelines
notify_on_failure:
  slack:
    team_channel: bot-data-infra-productivity-non-critical
    description: Testing example that will fail please pause after testing.
  pause_dag_on_error: true
deploy_env: local, dev, prod
models:
  - name: my_dbt_fail_pretest_model
    sensors:
      - sensor_always_pass
    custom_dependencies:
      - name: my_dbt_fail_pretest_model
        depends_on:
          - test_always_fail

```

Abstract away DAG Configurations

test_gsheet.pipeline.yml

```yaml
owner: anant.agarwal
start_date: 2023-06-20
schedule_interval: "30 * * * *"
description: A pipeline for testing the gsheet to sf dag
deploy_env: local, prod
notify_on_failure:
  slack:
    team_channel: bot-data-infra-productivity-critical
sheets:
  - Sheet1
table_prefix: gs
gsheet_url: "https://docs.google.com/spreadsheets/d/1jmdyVN5XxERAhTLfyk3IO-r6ipqLsRpAzGmHgTlOMXA/edit#gid=0"
```

Abstract away DAG Configurations

ingest_jira_to_snowflake.pipeline.yml

```yaml
owner: anant.agarwal
start_date: 2023-10-06
schedule_interval: "20 10 * * *"
description: Pull data from Jira into Snowflake
deploy_env: prod
notify_on_failure:
  opsgenie:
    responders: IC-SF-DataEng-Platform-Orchestration
    priority: P3
    description: Please check runbook https://instacart.atlassian.net/wiki/spaces/DATA/pages/3932356704/care-to-sf+Runbook
tasks:
  - airflow_task_name: care-to-sf
    isc_task_name: care-to-sf.data-eng
    command: python /app/bin/ingest_jira_to_snowflake.py
    task_properties:
      timeout: 90m
```

Abstract away DAG Configurations

example_delta_to_snowflake_sync.pipeline.yml

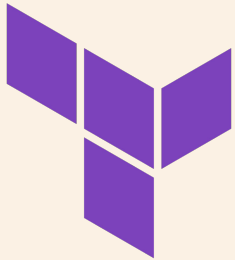
```yaml
owner: anant.agarwal
totem_system: team-3-2396   # Compute Frameworks
start_date: 2024-03-26
schedule_interval: null
description: Example pipeline to demonstrate how the delta to snowflake sync is configured
deploy_env: local, dev, prod
notify_on_failure:
  slack:
    team_channel: bot-data-infra-productivity-non-critical
tags: examples
tasks:
  - name: delta_to_snowflake_sync
    type: spark
    operator:
      databricks_connection_id: DATABRICKS_CONNECTION_ID_DELTA_TO_SNOWFLAKE_SYNC
      operator_params:
        type: delta_to_snowflake_sync
        delta_to_snowflake_sync_new_cluster_defaults: example_cluster
        delta_to_snowflake_sync_timeout_seconds: 3600
        delta_path: "{{ file_env_var('instacart_ads_dags/scheduling/examples/example_env_params.yml', 'delta_path') }}"
        snowflake_table_fq: "{{ file_env_var('instacart_ads_dags/scheduling/examples/example_env_params.yml', 'snowflake_table_fq') }}"
        snowflake_connection_id: snowflake_connection_id_delta_to_snowflake_sync
        key_columns: [ identity_key ]
        stage: instadata.public.delta_to_snowflake_stage_dev
```

Abstract away DAG Configurations

```python
from datetime import datetime, timedelta

from airflow import DAG
from airflow.operators.bash import BashOperator
from utils.config import AirflowConfig
from utils.notifications.opsgenie import create_opsgenie_failure_callback

# Dag that should always pass. Alerts should be triggered when this dag fails.
with DAG(
    "current_build",
    description="Print the current build loaded to the worker",
    schedule_interval="*/5 * * * *",
    start_date=datetime(2022, 6, 10),
    default_args={
        "owner": "Data Platform Orchestration",
        "on_failure_callback": create_opsgenie_failure_callback(
            responder_team="IC-SF-DataEng-Platform-Orchestration",
            priority="P1",
        ),
    },
    catchup=True,
    is_paused_upon_creation=False,
) as dag:

    t1 = BashOperator(
        task_id="print_current_build",
        bash_command="cat /app/current-build",
        retries=AirflowConfig.task_retry,
        retry_delay=timedelta(seconds=AirflowConfig.task_retry_delay),
    )
```

Terraform-ized Airflow

- Infrastructure as code!
- Easy to scale and configure resources
- Easy to spin up new clusters
- Consistency across DEV and PROD environments.
- Auditability and Compliance controls

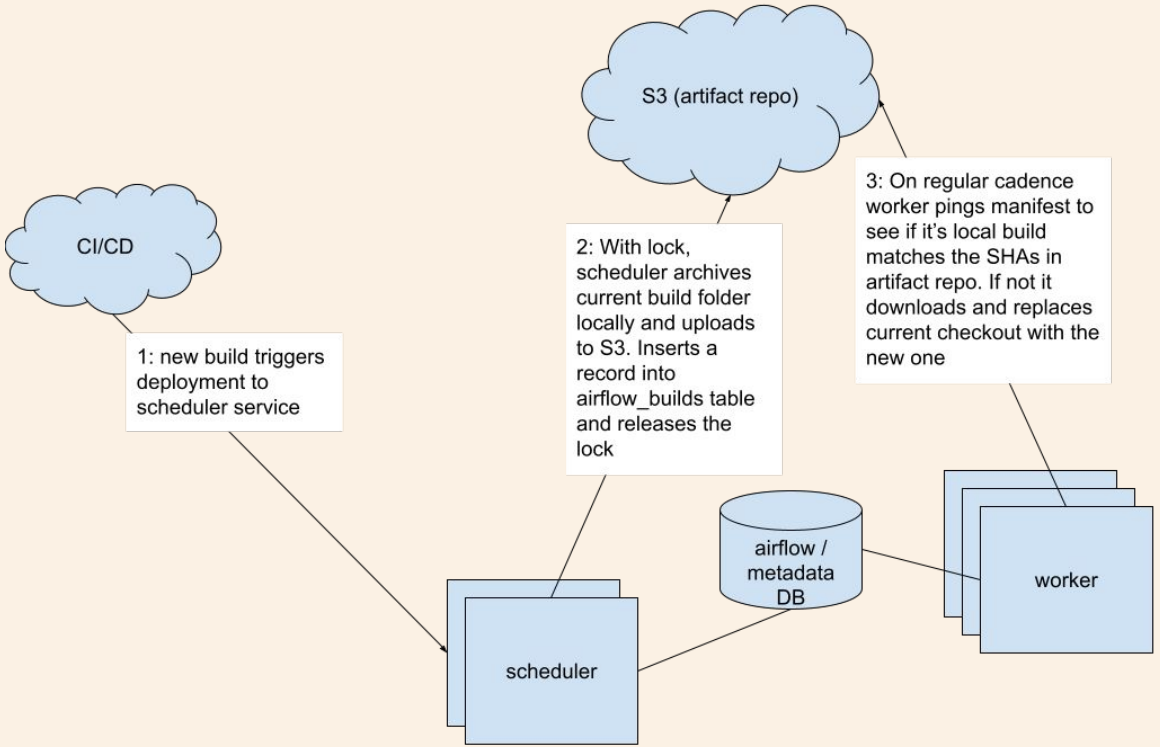


HashiCorp
Terraform

```
module "airflow2-test" {
  count                           = var.environment == "development" ? 1 : 0
  source                          = "github.com/instacart/terraform-airflow?ref=v1.20.3"
  environment                     = var.environment
  domain                          = var.domain
  tags                            = module.tags
  cluster_name                    = "infra"
  has_autoscaling                 = false
  name                            = "airflow2-test"
  app_name                        = "airflow2-playground"
  create_db_parameter_group       = true
  pgbouncer_security_group        = module.pgbouncer-defaults.security_groups
  rds_database_name               = "airflow2test"
  target_group_deregistration_delay = var.target_group_deregistration_delay
  webserver_dashed_name           = true
  proxy_webserver                 = true
  loadbalancer_security_group_ids   = { "LoadBalancer Security Group" : data.terraform_remote_state.ecs-clusters.outputs.data_eng_ecs_

ecs_cluster_names = [
    "infra-dev"
  ]
```

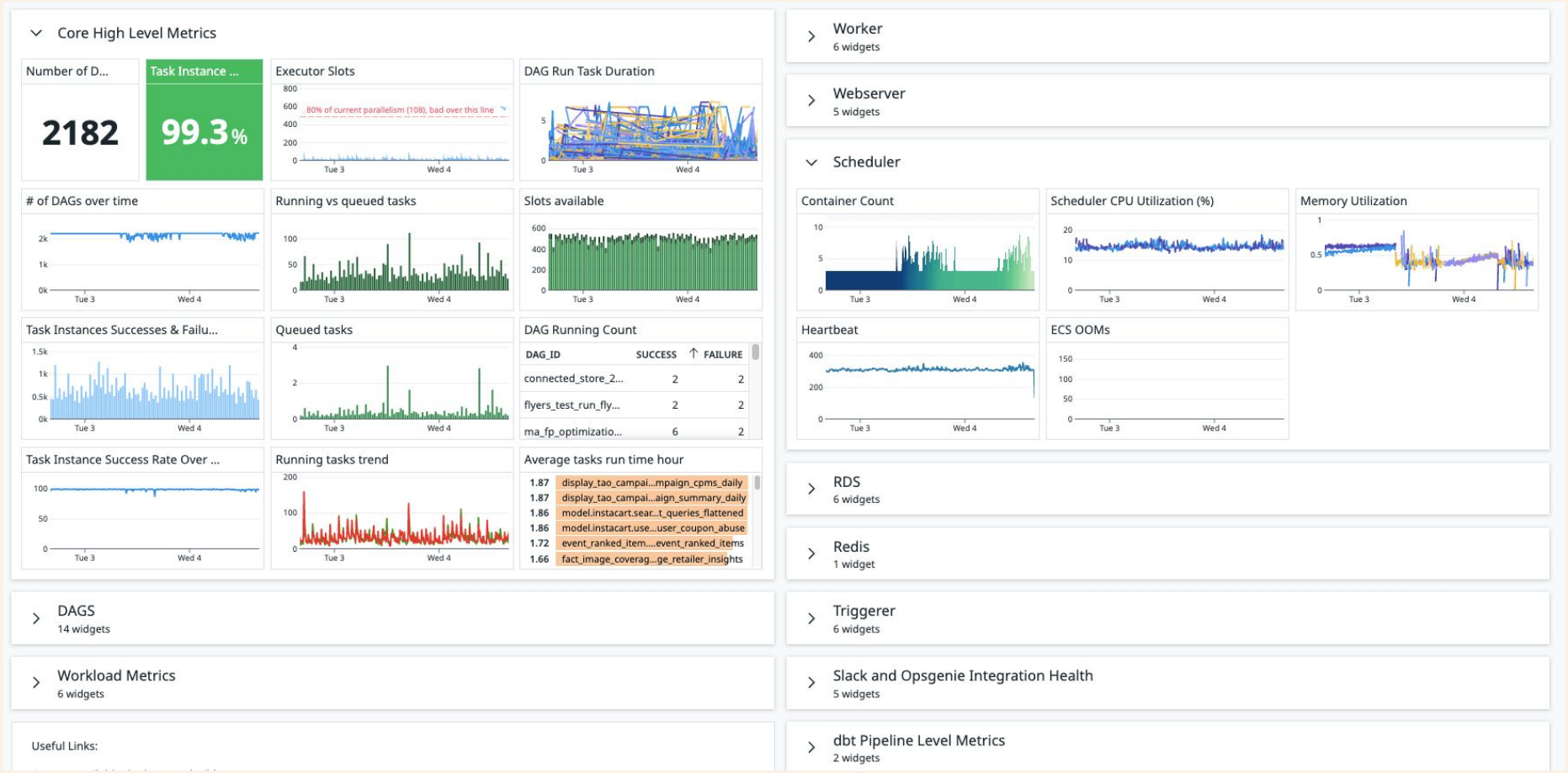
```
  rds_primary_instance_class      = var.rds_primary_instance_class
  rds_primary_is_multi_az         = var.rds_primary_is_multi_az
  rds_primary_skip_final_snapshot = var.rds_primary_skip_final_snapshot
  engine_version                  = var.engine_version

  redis_node_type             = var.redis_node_type
  redis_number_cache_clusters = var.redis_number_cache_clusters

  # replica config
  create_replica_rds_instance = var.create_replica_rds_instance
  rds_replica_instance_class  = var.rds_replica_instance_class
  create_replica_pgbouncer    = var.create_replica_pgbouncer

  webserver_cores = var.webserver_cores
  webserver_rammb = var.webserver_rammb
  webserver_nodes = var.webserver_nodes

  worker_cores           = var.worker_cores
  worker_rammb           = var.worker_rammb
  worker_nodes           = var.worker_nodes
  worker_has_autoscaling = false
  worker_min_capacity    = 2
  worker_max_capacity    = 2

  scheduler_cores = var.scheduler_cores
  scheduler_rammb = var.scheduler_rammb
  scheduler_nodes = var.scheduler_nodes

  datadog_monitor_notification_list = ["@something"]
}
```

Development Friendly CI/CD



Monitoring and Alerting

- Monitoring as a first class citizen!
- Airflow Platform Monitoring
- DAG Monitoring
- Providing abstractions to create monitoring resources easily


Core High Level Metrics
Number of D...
2182
Task Instance ...
99.3%
Executor Slots
80% of current parallelism (108), bad over this line
DAG Run Task Duration
# of DAGs over time
Running vs queued tasks
Slots available
Task Instances Successes & Failu...
Queued tasks
DAG Running Count
DAG_ID SUCCESS FAILURE
connected_store_2... 2 2
flyers_test_run_fly... 2 2
ma_fp_optimizatio... 6 2
Task Instance Success Rate Over ...
Running tasks trend
Average tasks run time hour
1.87 display_tao_campai...mpaign_cpms_daily
1.87 display_tao_campai...aign_summary_daily
1.86 model.instacart.sear...t_queries_flattened
1.86 model.instacart.use...user_coupon_abuse
1.72 event_ranked_item....event_ranked_items
1.66 fact_image_coverag...ge_retailer_insights
Tue 3
Wed 4
DAGS
14 widgets
Workload Metrics
6 widgets
Useful Links:
Worker
6 widgets
Webserver
5 widgets
Scheduler
Container Count
Scheduler CPU Utilization (%)
Memory Utilization
Heartbeat
ECS OOMs
RDS
6 widgets
Redis
1 widget
Triggerer
6 widgets
Slack and Opsgenie Integration Health
5 widgets
dbt Pipeline Level Metrics
2 widgets

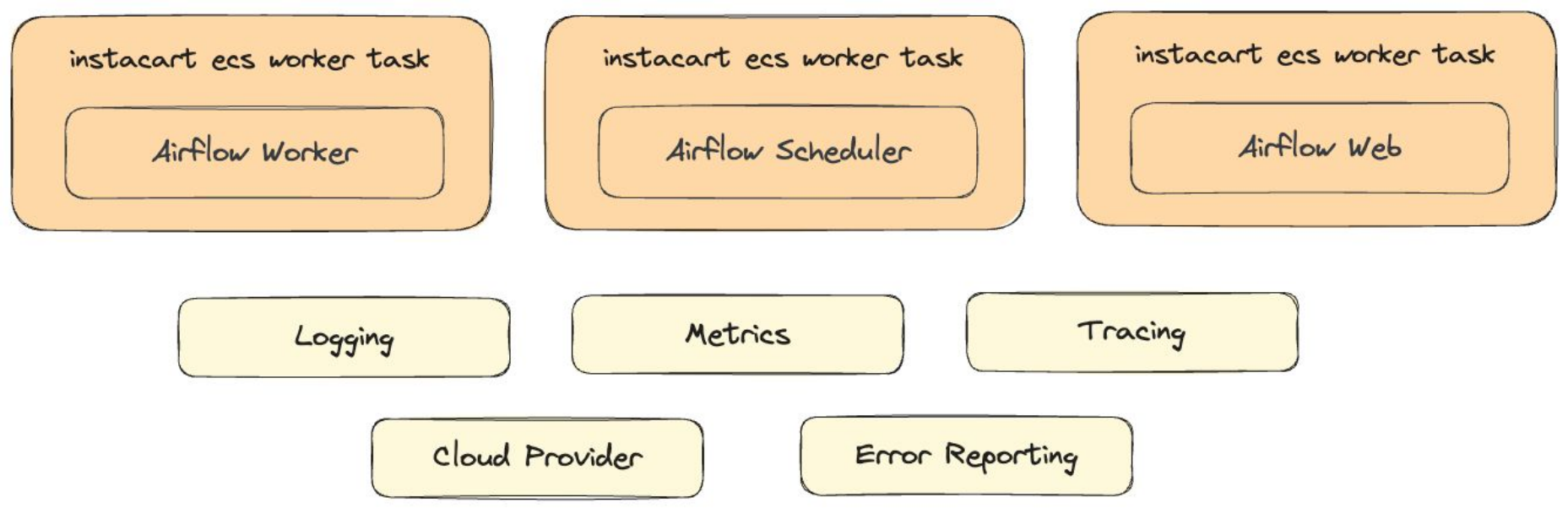
```
resource "datadog_monitor" "airflow_scheduler_heartbeat" {
  name    = "[Data Platform Airflow] Scheduler No Heartbeat"
  type    = "query alert"
  query   = "sum(last_10m):avg:airflow.scheduler_heartbeat{environment:production,ecs_cluster:data-eng}.as_count() <= 0"
  message = <<-EOT
    service
    https://isc.fernet.io/services?page=1&per_page=50&query%5Benvironment%5D=production&query%5Bservice%5D=data-platform-airflow

    UI
    https://data-platform-airflow.icprivate.com/home

    Runbook https://instacart.atlassian.net/wiki/spaces/DATA/pages/3438412138/Airflow+monitor+and+runbook
    @opsgenie-IC-SF-DataEng-Platform-Orchestration-P1
  EOT

  tags = ["team:data-eng-platform"]

  monitor_thresholds {
    critical          = 0
    critical_recovery = 1
  }

  notify_audit        = false
  require_full_window = false
  notify_no_data      = true
  renotify_interval   = 0
  include_tags        = true
  no_data_timeframe   = 10
  priority            = 1
}
```

Key Takeaways

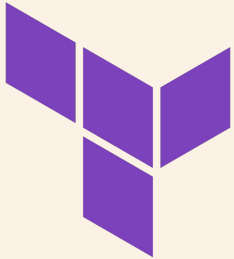
Our Recommendations!

instacart

Key Takeaway 1: Building Abstractions



Key Takeaway 2: Terraforming Resources



HashiCorp
Terraform

+

=

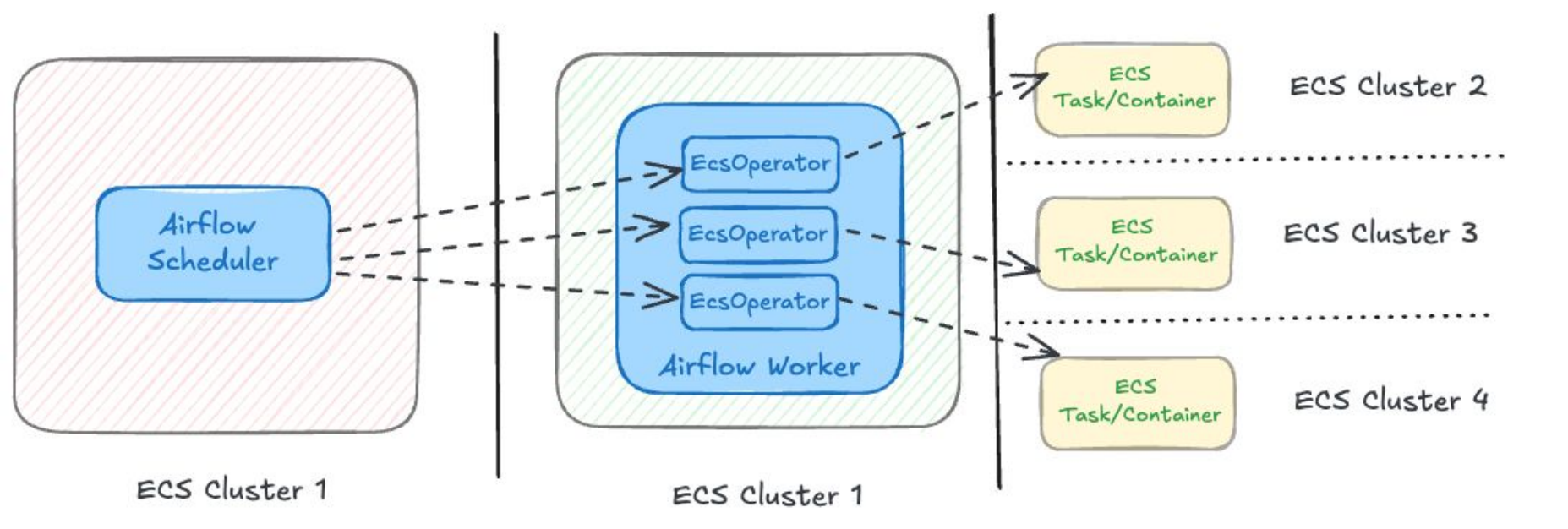
DATADOG

Key Takeaway 3: YML DAGs & Remote Executors!

ingest_jira_to_snowflake.pipeline.yml

```yaml
owner: anant.agarwal
start_date: 2023-10-06
schedule_interval: "20 10 * * *"
description: Pull data from Jira into Snowflake
deploy_env: prod
notify_on_failure:
  opsgenie:
    responders: IC-SF-DataEng-Platform-Orchestration
    priority: P3
    description: Please check runbook https://instacart.atlassian.net/wiki/spaces/DATA/pages/3932356704/care-to-sf+Runbook
tasks:
  - airflow_task_name: care-to-sf
    isc_task_name: care-to-sf.data-eng
    command: python /app/bin/ingest_jira_to_snowflake.py
    task_properties:
      timeout: 90m

```

Key Takeaway 3: DAG abstractions & Remote Workers!



Questions?

- Find me on Airflow's community slack
- https://www.linkedin.com/in/anantag/

instacart