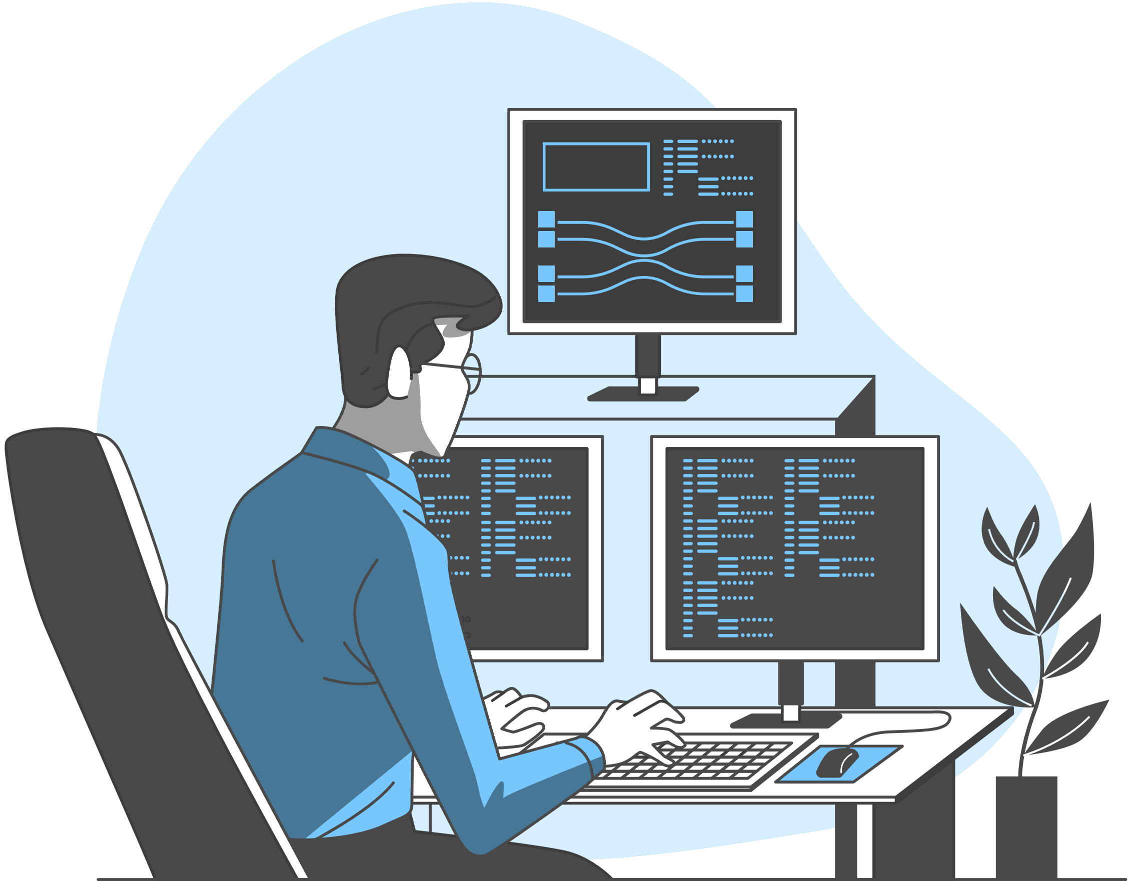
Welcome!



Where my data engineers at?

Have you heard?

is the new oil!

is the new gold!

E stands for Extracting

About me and what I do

bestseller.com

VERO MODA

.OBJECT

ONLY

VILA CLOTHES

SELECTED / FEMME

Y.A.S

JACQUELINE de YONG

PIECES

NOISY MAY

mama;licious

ONLY PLAY

ONLY CARMAKOMA

VERO MODA CURVE

ONLY & SONS PASSIONATE JEANS MAKERS

JACK&JONES

SELECTED / HOMME

KIDS ONLY

LMTD

name it

Lil'Atelier

JACK&JONES JUNIOR

ONLY PLAY GIRLS

Solving data governance with Airflow and AWS ECS Anywhere

Residency and governance

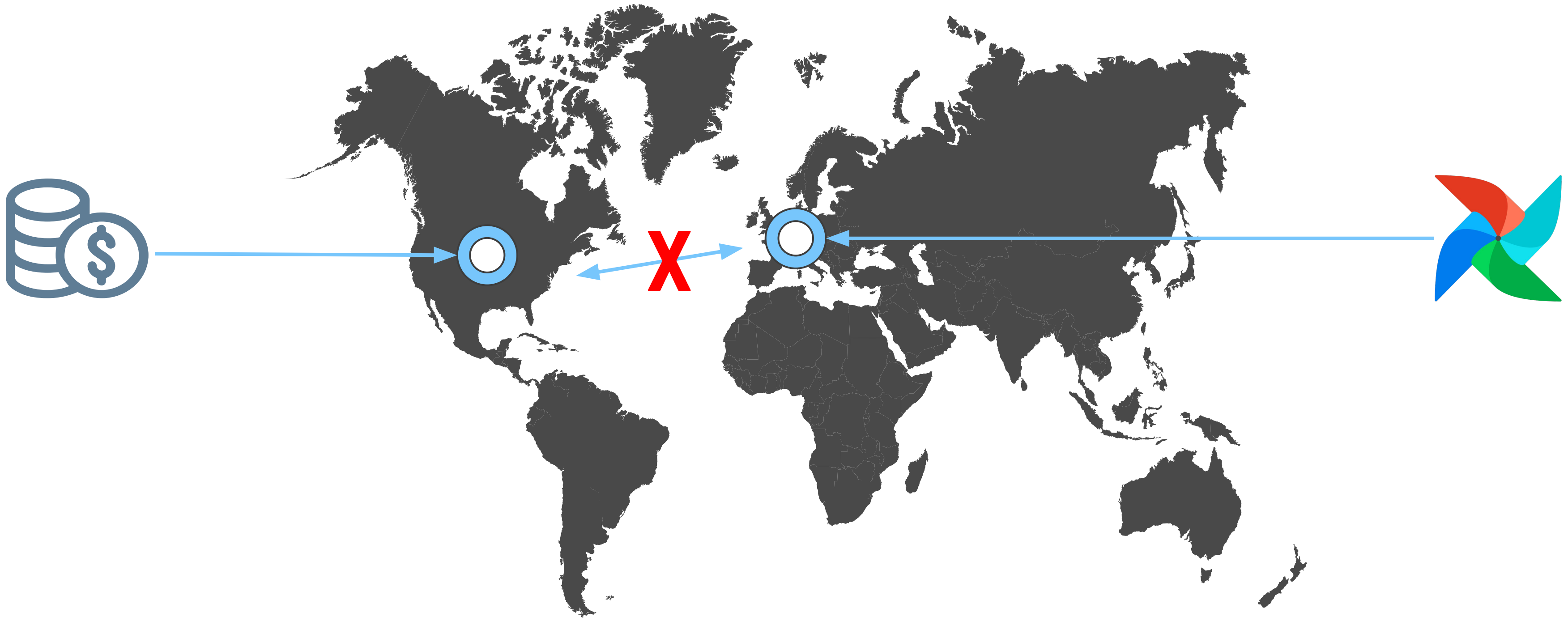
Residency reh·zi·duhn·see	Where something lives (resides)
Governance guh·vr·nuhns	How that "something" is shaped and managed

Problem statement

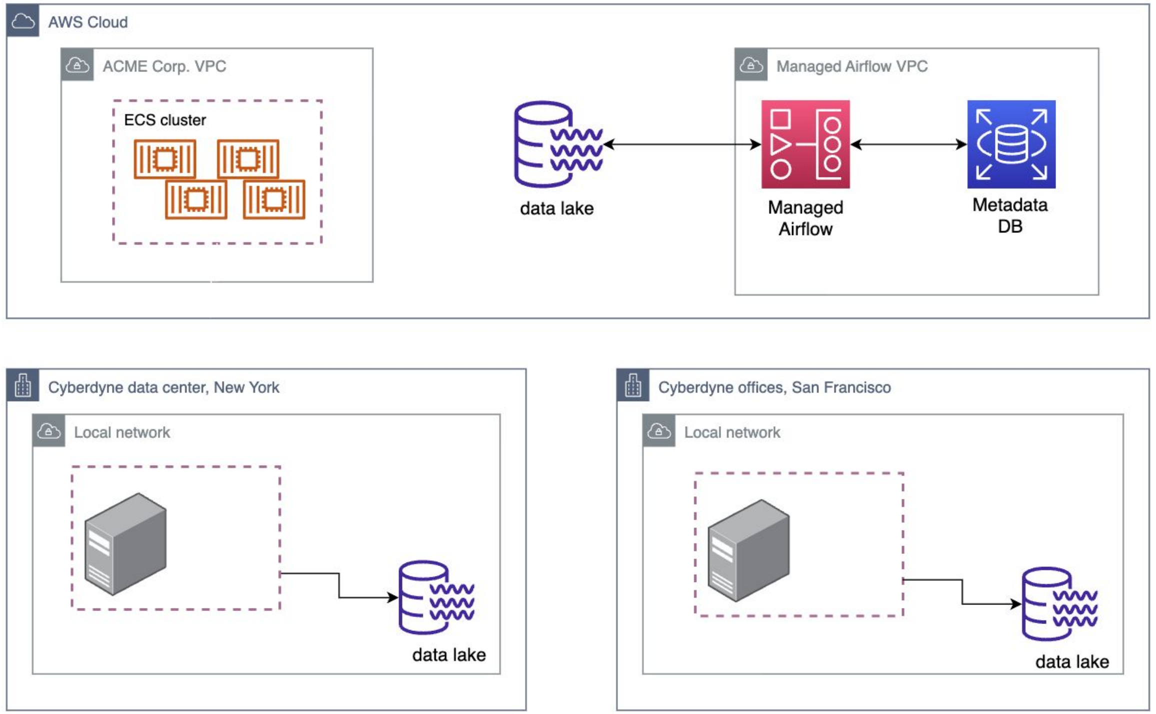
ACME acquired a competitor in the US.

ACME acquired a competitor in the US and now they have a problem with paying Christmas bonuses.

Problem statement



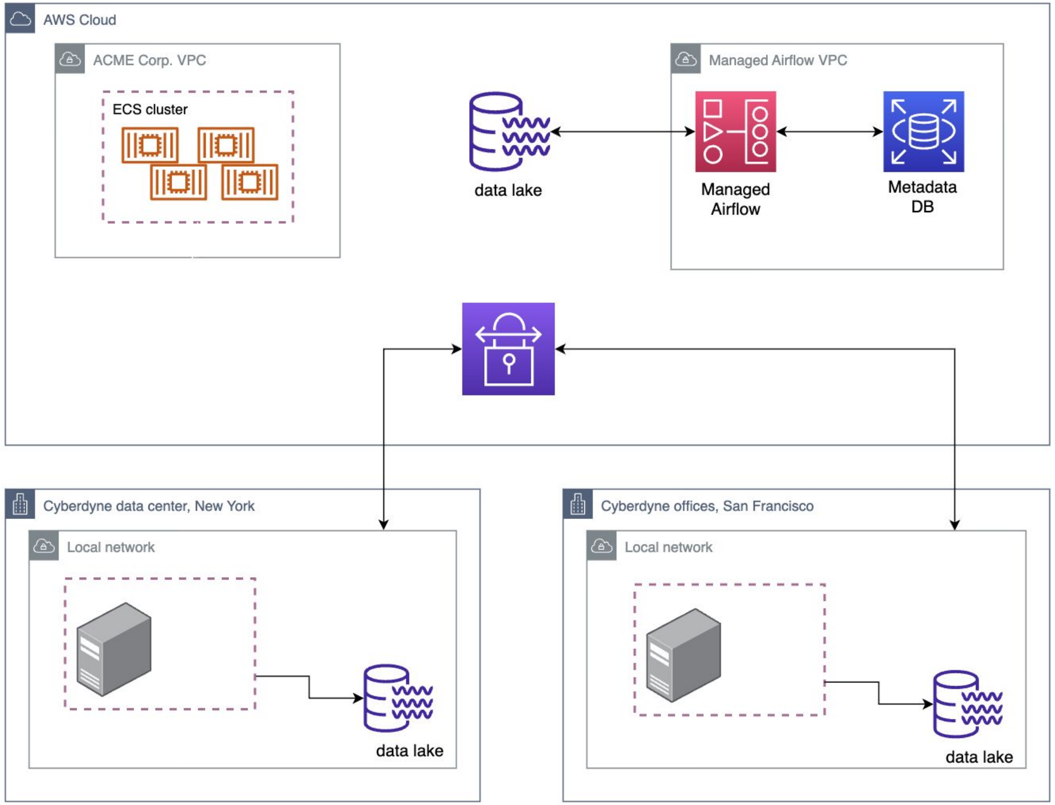
Architecturally, we are in this situation



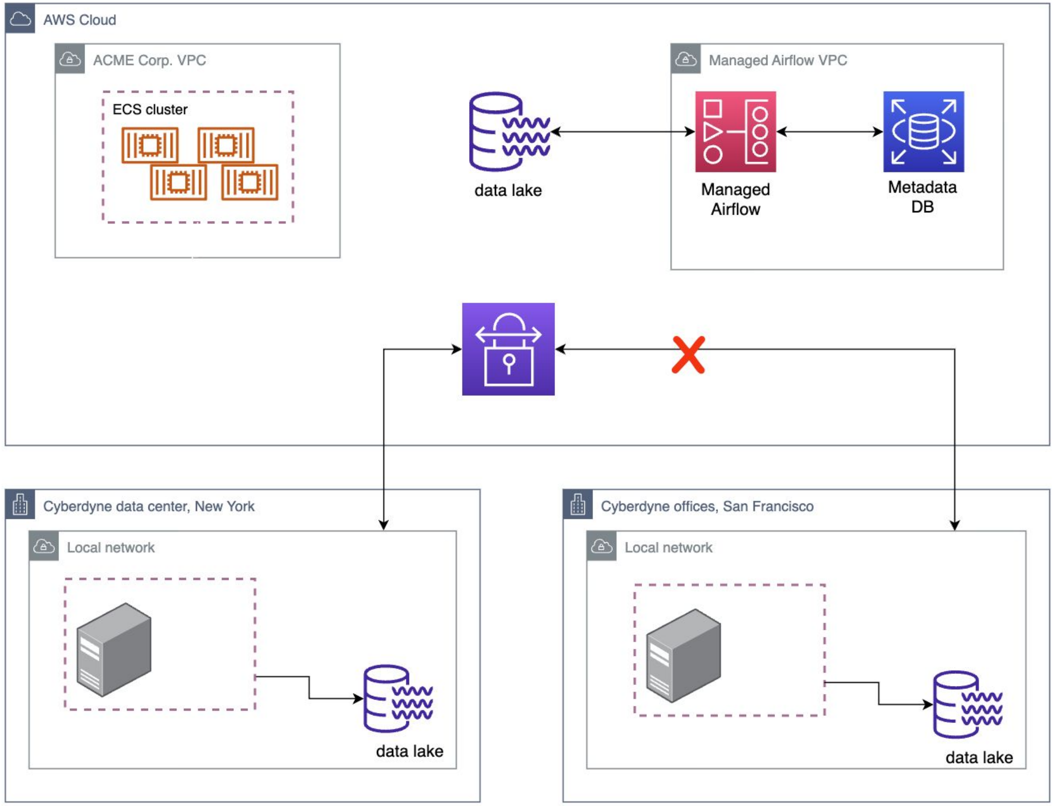
How would you solve the problem?

Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

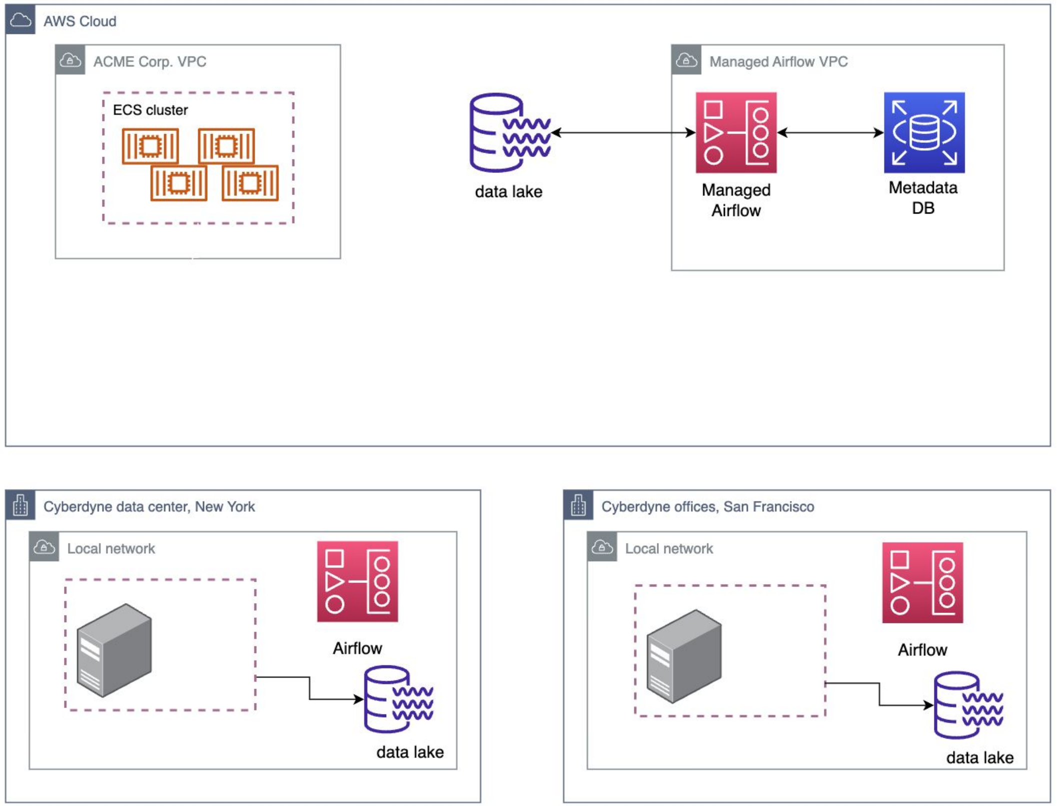
It’s all about trade-offs



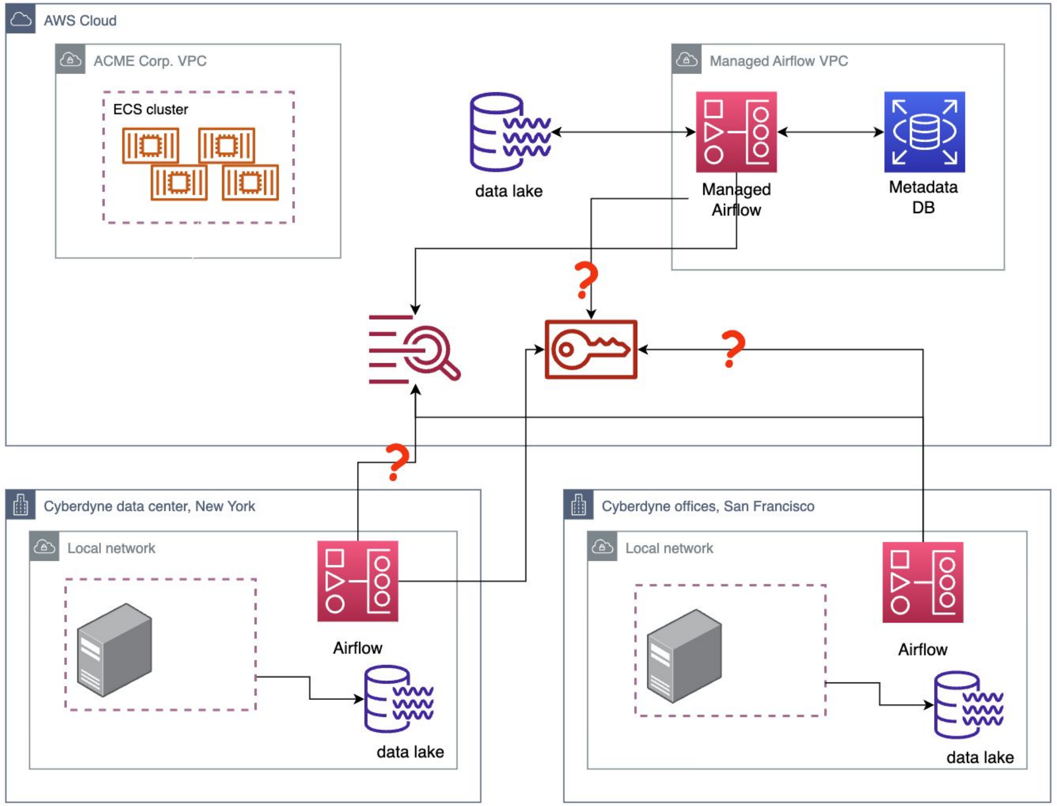
It's all about trade-offs



It’s all about trade-offs



It's all about trade-offs



“People that made the most money in a **gold rush** were **selling shovels**, not digging gold”

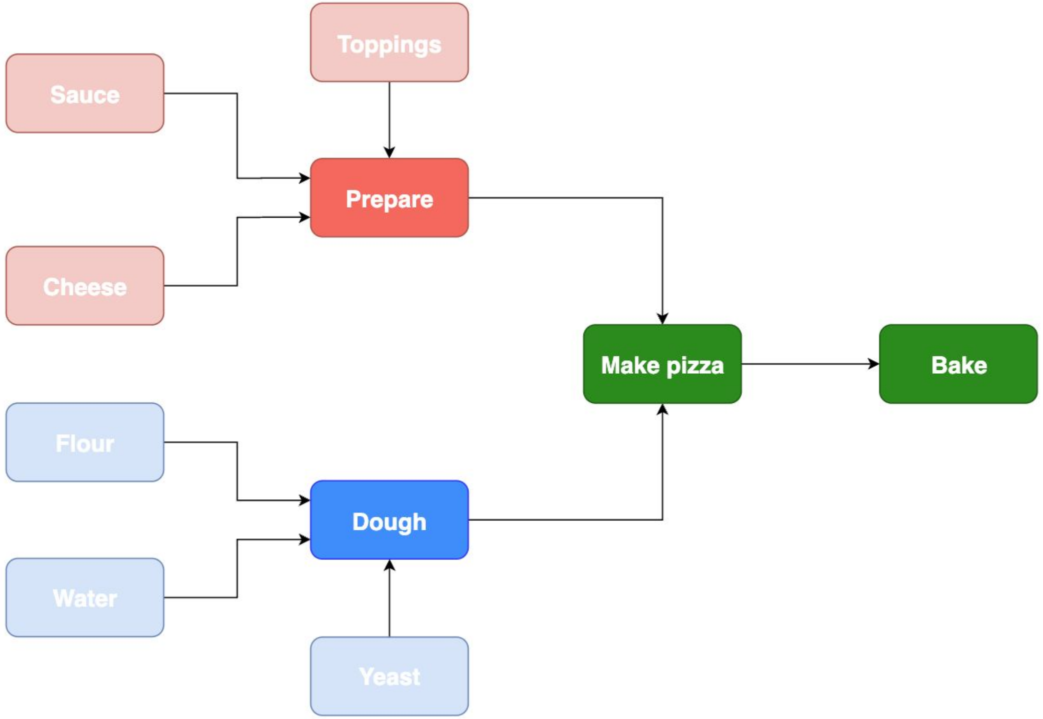
AWS ECS and Apache Airflow

AWS ECS

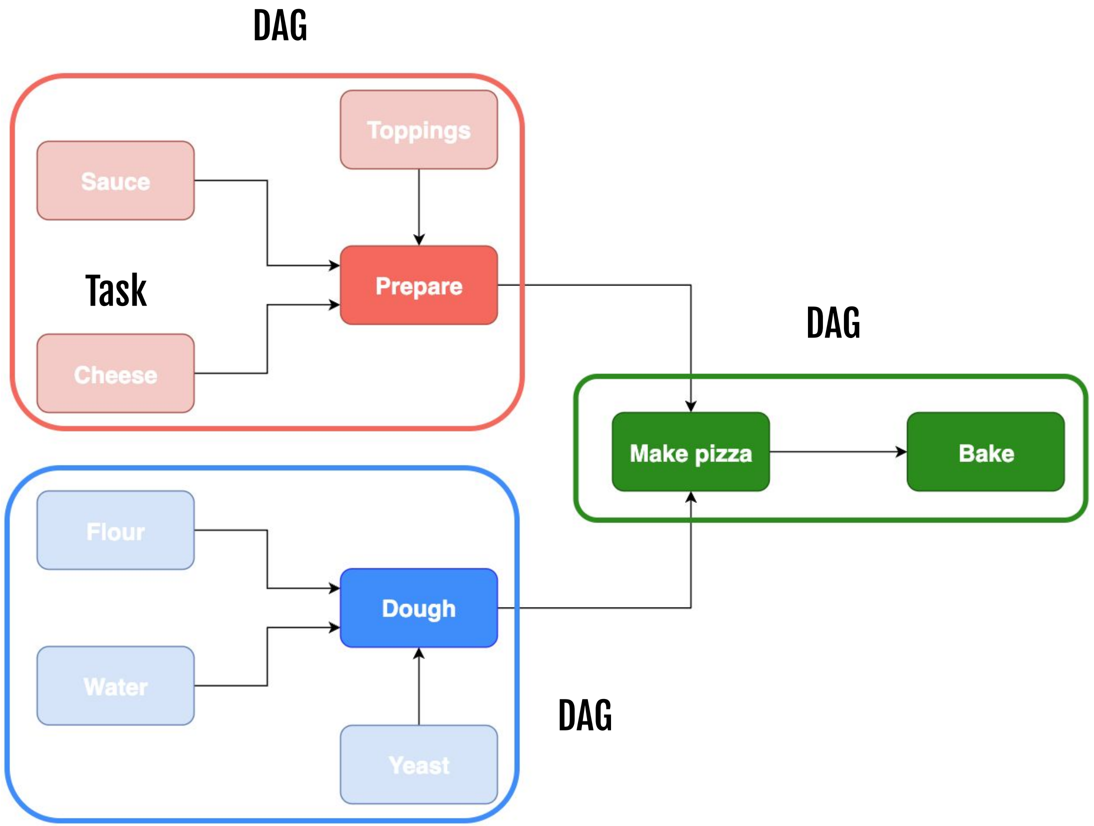
AWS ECS in the real world

Apache Airflow

Workflows and dependencies



Workflows and dependencies

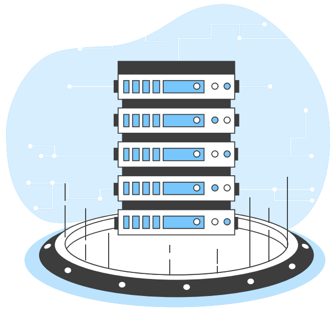


Batteries included!

Operators

```
with DAG("my_dag"):
    results = RedshiftSQLOperator(sql="SELECT * from ...")
    email = EmailOperator(to="admin@example.com", subject="Redshift results")

    results >> email
```

AWS ECS and Apache Airflow together



AWS ECS Anywhere

ECS Anywhere schedules and runs containers on your infrastructure.

Airflow ECS Operator

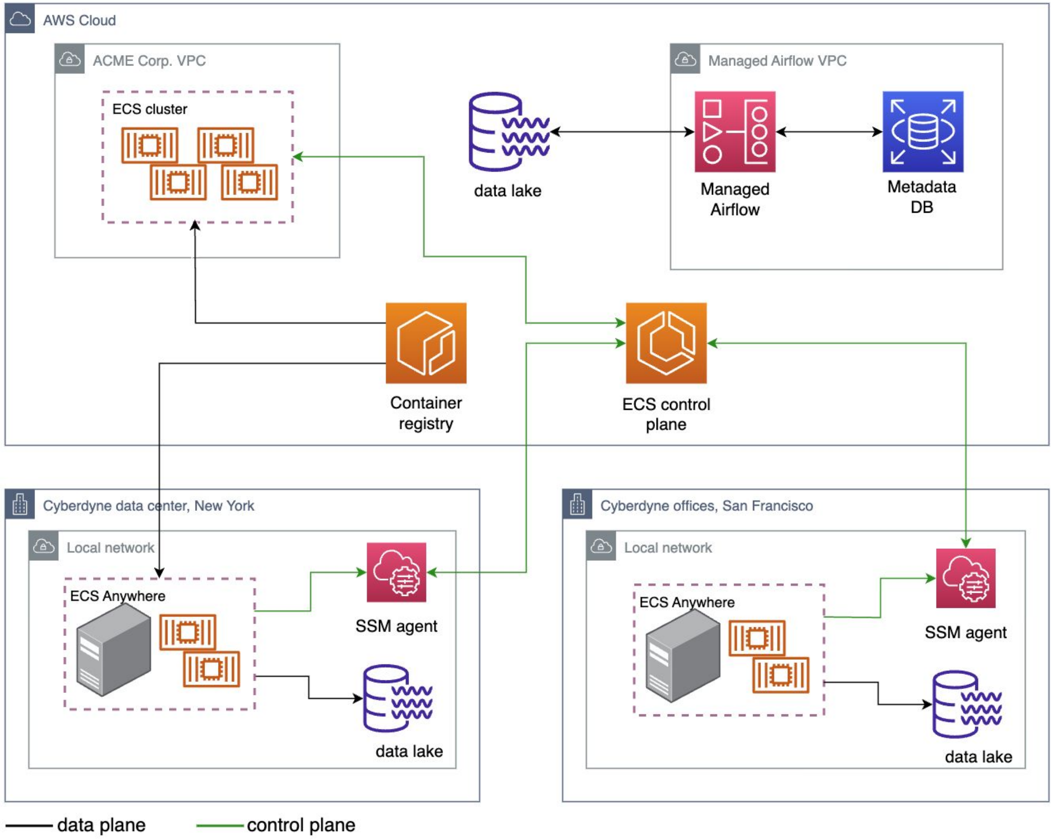
ECS Operator allows you to schedule and run DAGs as containers on ECS

Airflow ECS operator

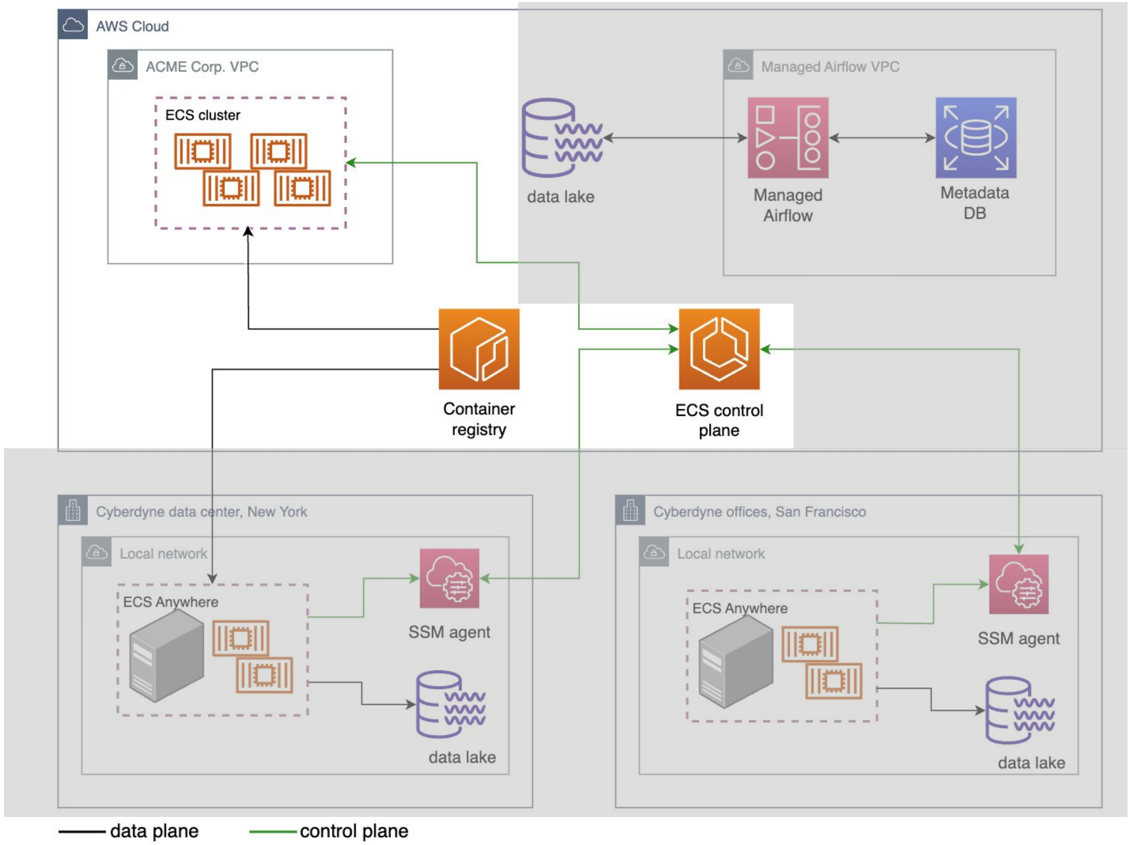
```
task = EcsRunTaskOperator(
    task_id="my_task_id",
    dag=dag,
    cluster="my_cluster",
    task_definition="task_definition:2",
)
```

task == container

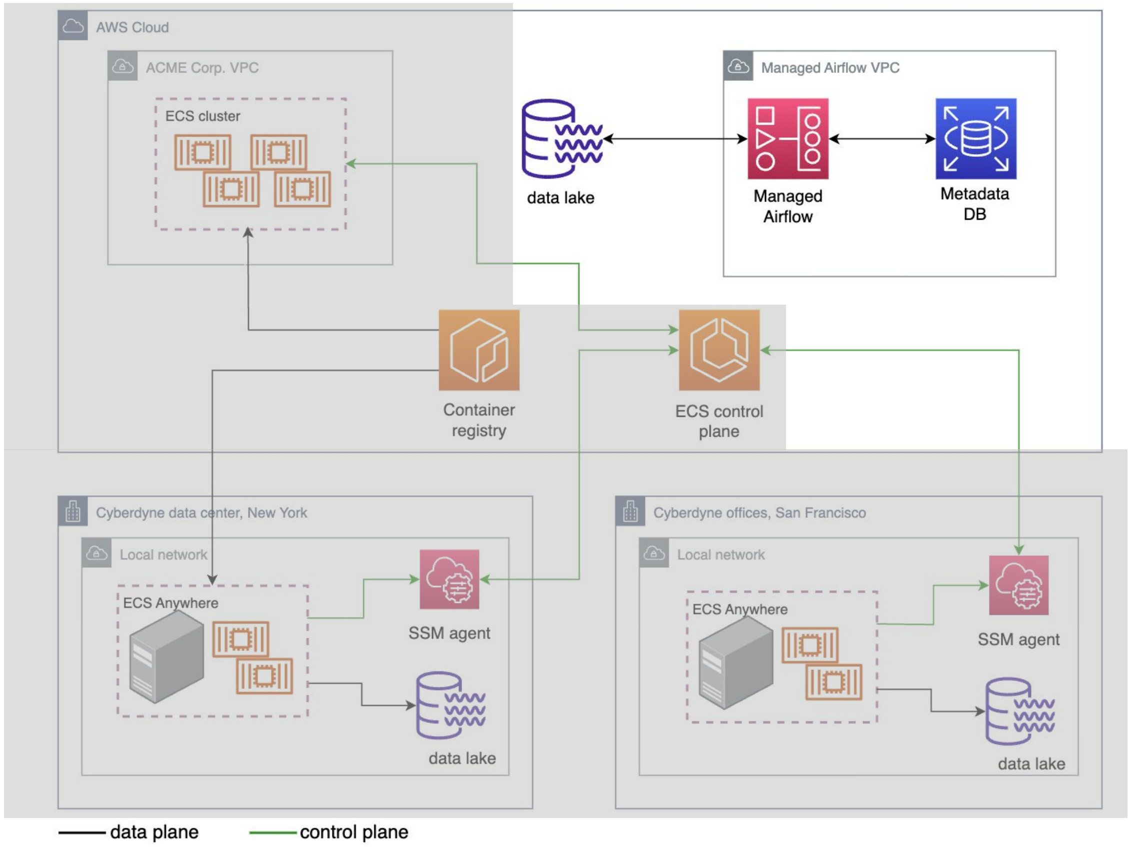
AWS ECS Anywhere



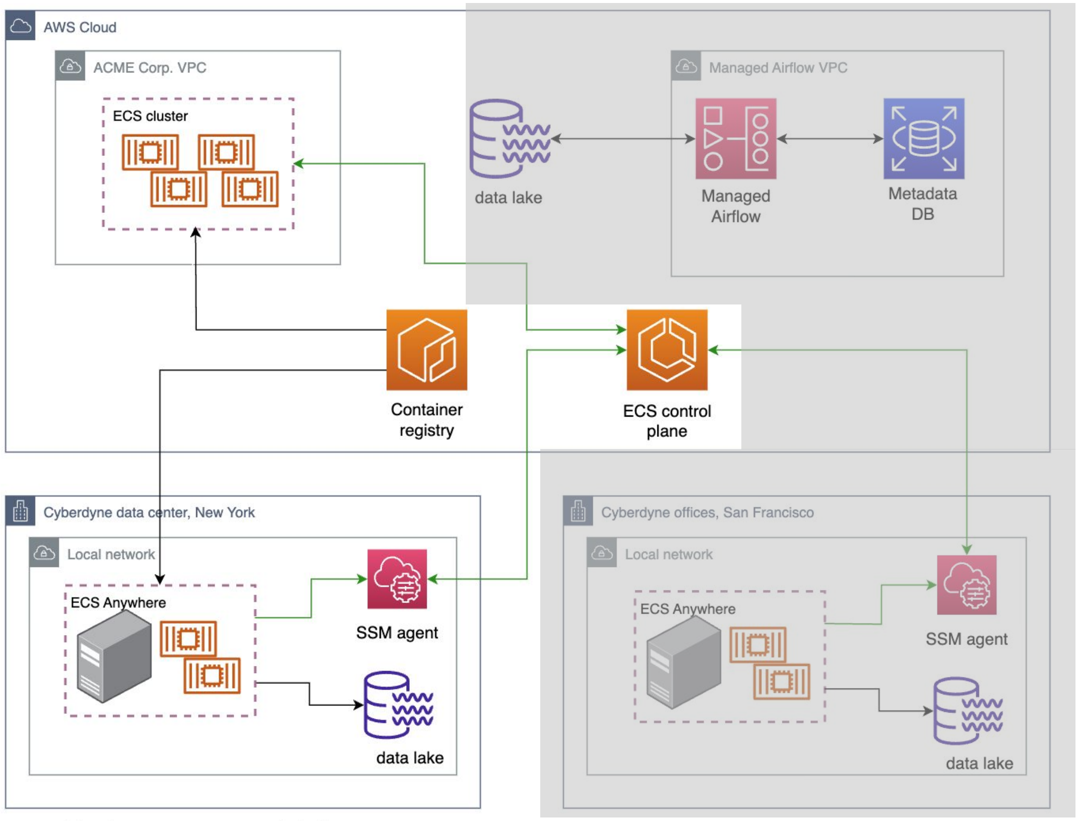
AWS ECS


AWS Cloud
ACME Corp. VPC
ECS cluster
Managed Airflow VPC
data lake
Managed Airflow
Metadata DB
Container registry
ECS control plane
Cyberdyne data center, New York
Local network
ECS Anywhere
SSM agent
data lake
Cyberdyne offices, San Francisco
Local network
ECS Anywhere
SSM agent
data lake
data plane
control plane

Managed Airflow

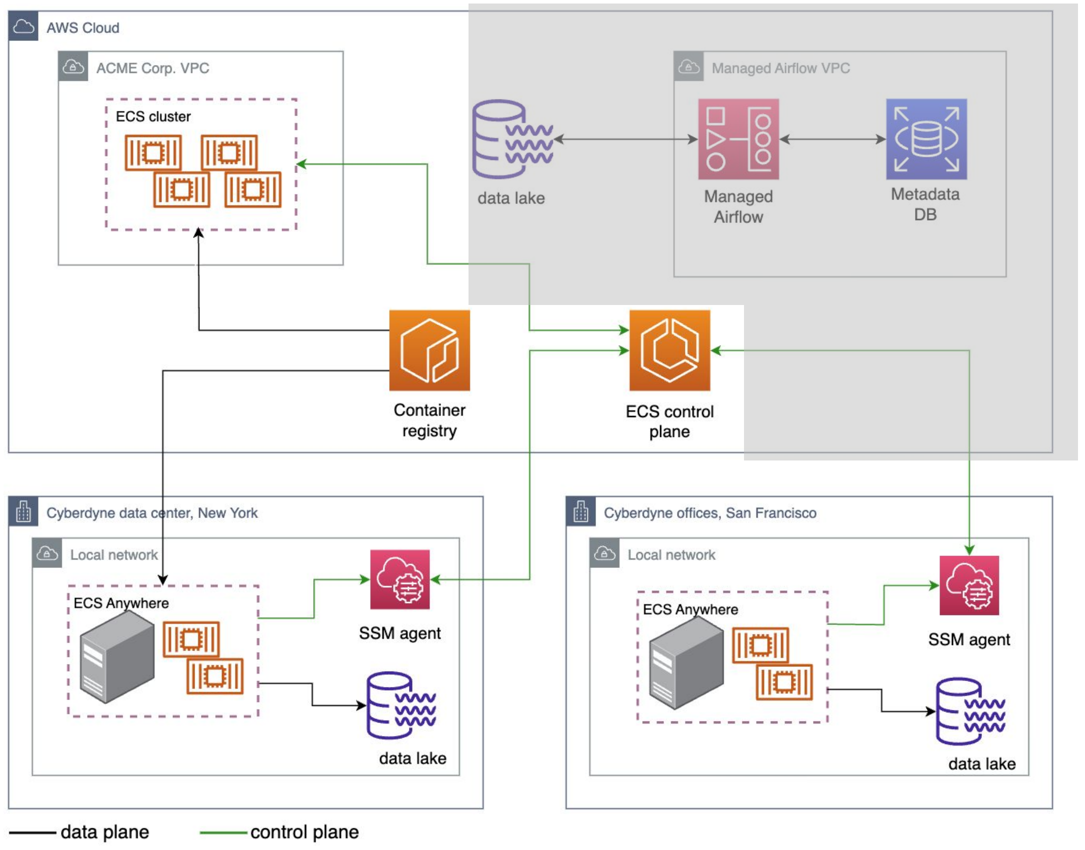


AWS ECS Anywhere including New York



—— data plane —— control plane

AWS ECS Anywhere including New York and San Francisco



Demo time!

01

Show ETL scripts

Christmas bonus and amount of taxes to be paid.

02

Run ETL scripts standalone

To verify they work.

03

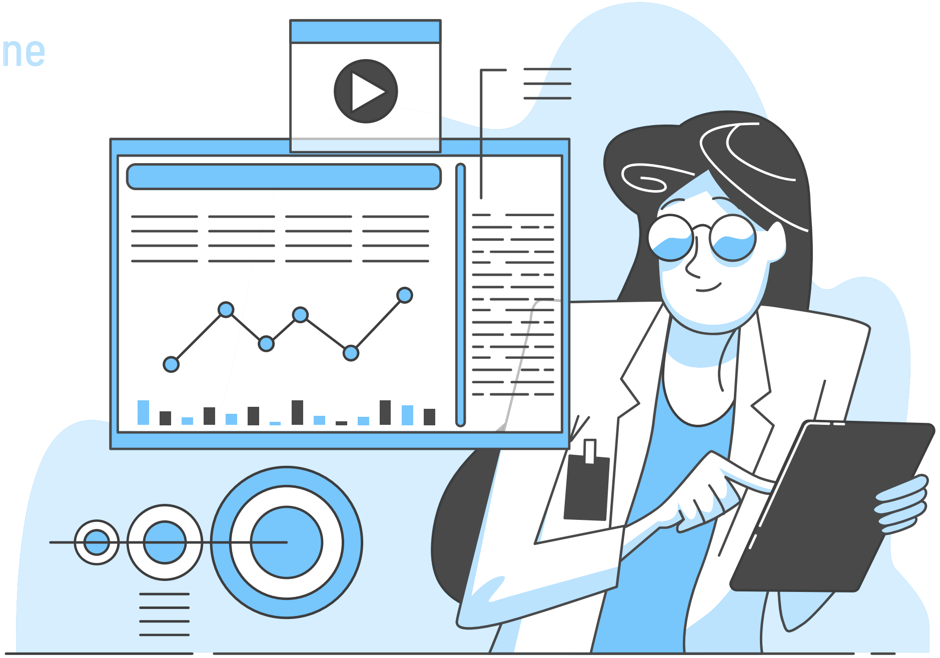
Containerize them

So they are ready to run on ECS.

04

Orchestrate with Airflow

For fun and profit!



Bonus calculation ETL script

```python
def create_salary_with_bonus(employee_data: List[Dict], result: str = "../salary_with_bonus.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = create_salary_with_bonus(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

Bonus calculation ETL script

```python
def create_salary_with_bonus(employee_data: List[Dict], result: str = "../salary_with_bonus.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = create_salary_with_bonus(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

Bonus calculation ETL script

```
def create_salary_with_bonus(employee_data: List[Dict], result: str = "../salary_with_bonus.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = create_salary_with_bonus(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

Tax calculation ETL script

```python
def calculate_total_tax(employee_data: List[Dict], result_filename: str = "../tax_amount.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = calculate_total_tax(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

Tax calculation ETL script

```
def calculate_total_tax(employee_data: List[Dict], result_filename: str = "../tax_amount.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = calculate_total_tax(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

Tax calculation ETL script

```python
def calculate_total_tax(employee_data: List[Dict], result_filename: str = "../tax_amount.csv") -> str:
  pass

def upload_to_s3(file_path: str, bucket_name: str, s3_folder_name: str = "") -> bool:
  pass

def main():
  employee_data = read_data(args.input_file)

  local_file = calculate_total_tax(employee_data, args.output_file)
  upload_to_s3(local_file, bucket_name=args.s3_bucket, s3_folder_name=args.s3_folder)
```

01

Show ETL scripts

Christmas bonus and amount of taxes to be paid.

02

Run ETL scripts standalone

To verify they work.

03

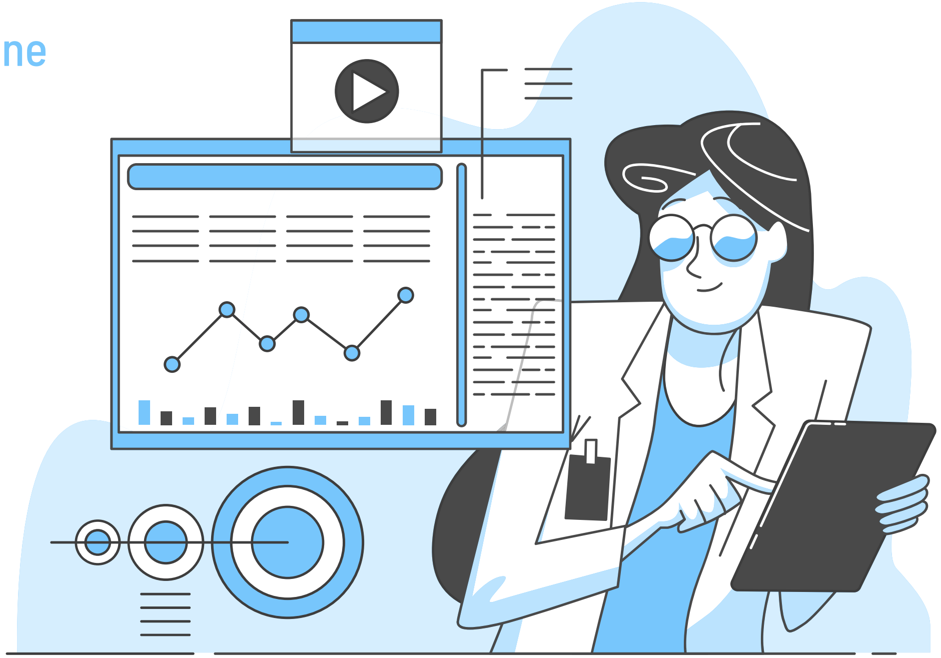
Containerize them

So they are ready to run on ECS.

04

Orchestrate with Airflow

For fun and profit!



Run tax script standalone

```
python docker/scripts/tax_amount.py \
  --input_file=docker/employee_data.csv \
  --output_file=docker/tax_amount.csv \
  --s3_bucket=mwaa-ecs-anywhere-bucket \
  --s3_folder=tax
```

```
aws s3 ls mwaa-ecs-anywhere-bucket/tax/
2023-06-16 10:30:58         20 tax_amount.csv
```

Run bonus script standalone

```
python docker/scripts/christmas_bonus.py \
  --input_file=docker/employee_data.csv \
  --output_file=docker/salary_with_bonus.csv \
  --s3_bucket=mwaa-ecs-anywhere-bucket \
  --s3_folder=bonus
```

```
aws s3 ls mwaa-ecs-anywhere-bucket/bonus/
2022-06-16 10:31:26       1712 salary_with_bonus.csv
```

01

Show ETL scripts

Christmas bonus and amount of taxes to be paid.

02

Run ETL scripts standalone

To verify they work.

03

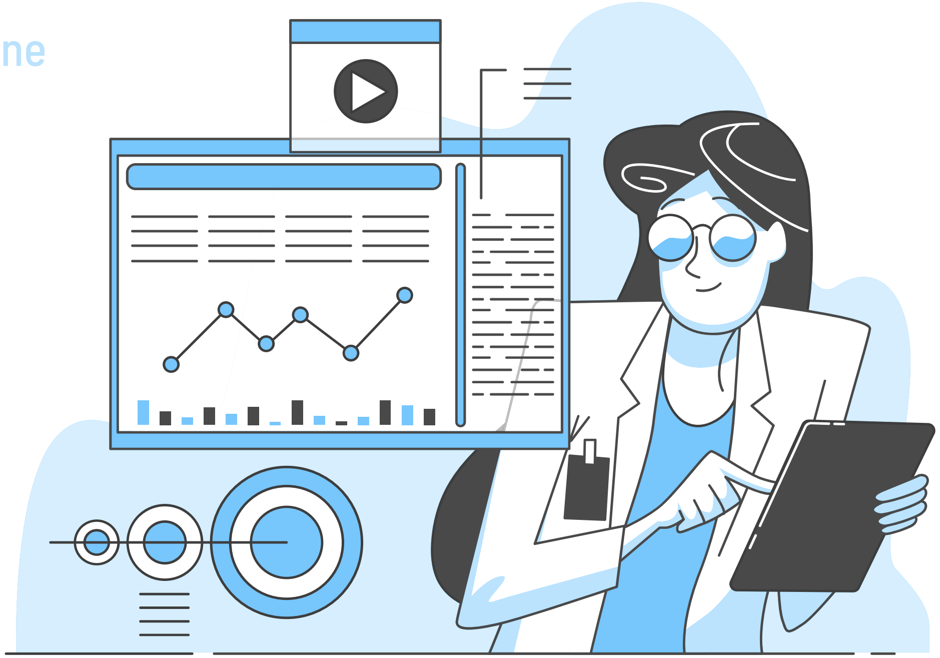
Containerize them

So they are ready to run on ECS.

04

Orchestrate with Airflow

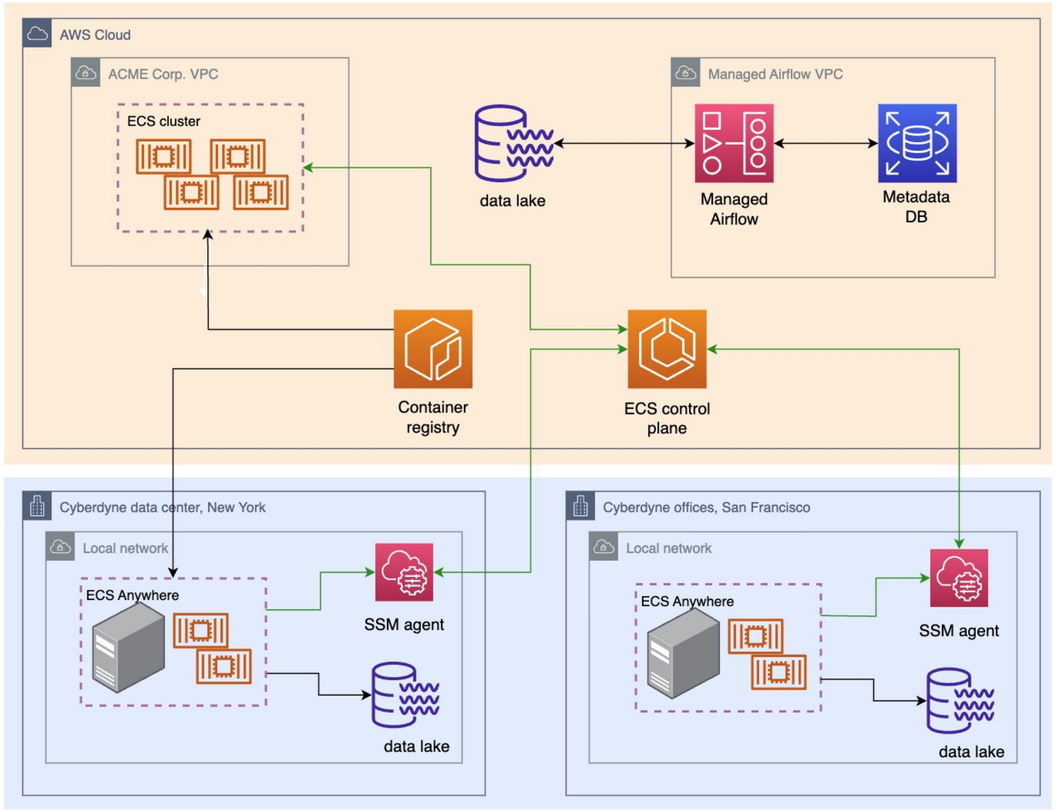
For fun and profit!



Containerize those ETL scripts

```
FROM  python:3.9.16-slim

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip3 --no-cache-dir install -r requirements.txt

COPY etl_scripts/*.py /app/
```

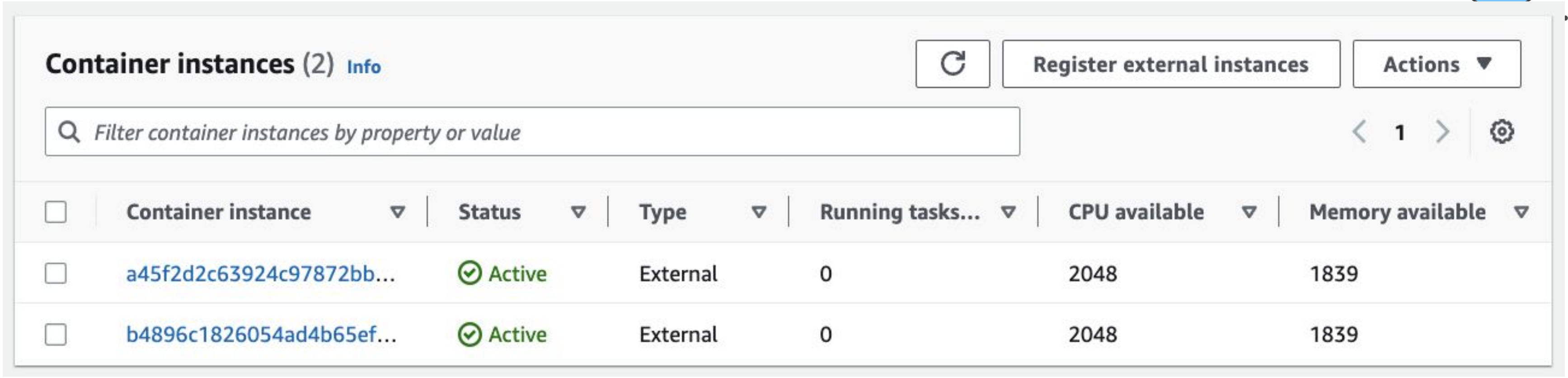
```
docker push ACCOUNT_ID.dkr.ecr.eu-central-1.amazonaws.com/mwaa-ecs-anywhere-repo:1
The push refers to repository [ACCOUNT_ID.dkr.ecr.eu-central-1.amazonaws.com/mwaa-ecs-anywhere-repo]
...
1: digest: sha256:5aeb31d4bbd4fa7083dfcd2b917b5adc7b6746ee2689992ebb8b689e1c841241 size: 2203
```



🟧 AWS infrastructure

🟦 existing infrastructure simulated with Vagrant

Vagrant VMs as external ECS instances


Register external instances
Actions
Filter container instances by property or value
1

Container instances (2) Info

Container instance	Status	Type	Running tasks...	CPU available	Memory available
a45f2d2c63924c97872bb...	Active	External	0	2048	1839
b4896c1826054ad4b65ef...	Active	External	0	2048	1839

Run bonus task (container)

```
aws ecs run-task \
  --cluster "cluster" \
  --count 1 \
  --launch-type EXTERNAL \
  --task-definition mwaa-ecs-anywhere-christmas-bonus:5
```

Check bonus task (container) logs

```
2023-06-16 10:33:02 - INFO - Running on host 'bonus.host'
2023-06-16 10:33:02 - INFO - Reading CSV file '/data/employee_data.csv'
2023-06-16 10:33:02 - INFO - Creating the yearly salary with bonus report at '/tmp/salary_with_bonus.csv'
2023-06-16 10:33:05 - INFO - Uploading local file '/tmp/salary_with_bonus.csv' to 's3://mwaa-ecs-anywhere-
bucket/bonus/salary_with_bonus.csv'
```

Check bonus task (container) logs

```
2023-06-16 10:33:02 - INFO - Running on host 'bonus.host'
2023-06-16 10:33:02 - INFO - Reading CSV file '/data/employee_data.csv'
2023-06-16 10:33:02 - INFO - Creating the yearly salary with bonus report at '/tmp/salary_with_bonus.csv'
2023-06-16 10:33:05 - INFO - Uploading local file '/tmp/salary_with_bonus.csv' to 's3://mwaa-ecs-anywhere-
bucket/bonus/salary_with_bonus.csv'
```

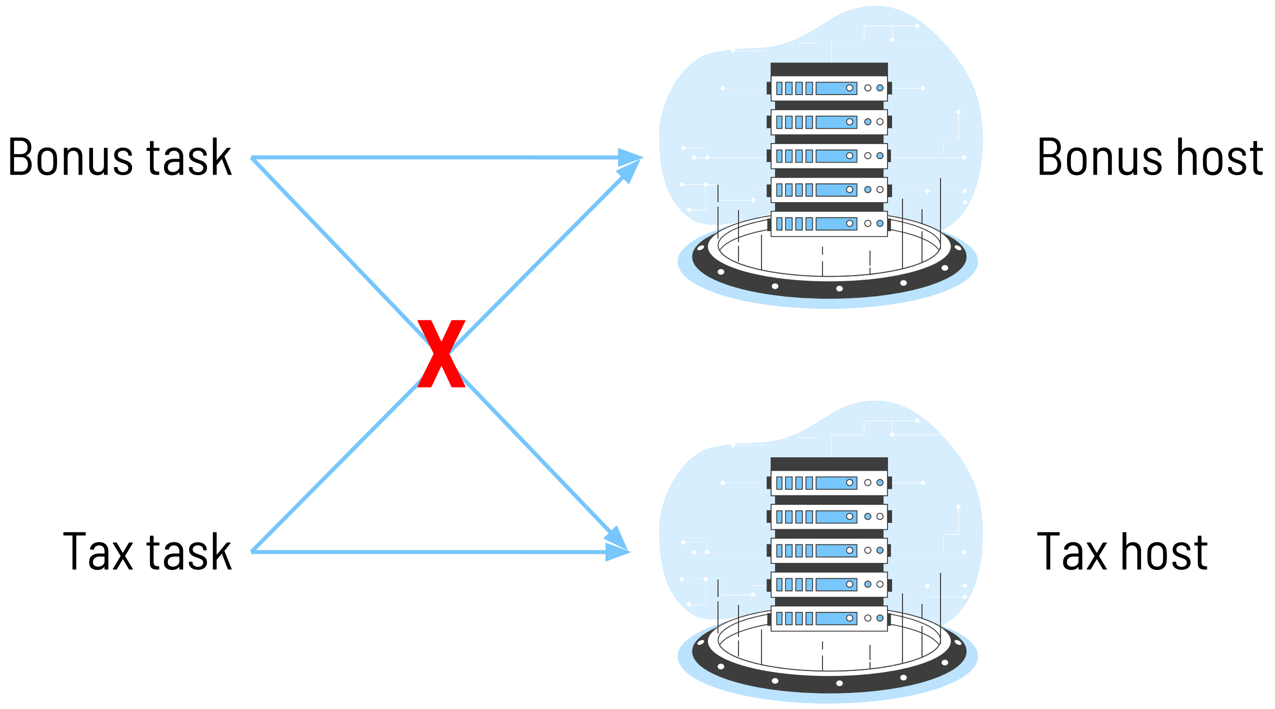
Run bonus task (container)

```
aws ecs run-task \
  --cluster "cluster" \
  --count 1 \
  --launch-type EXTERNAL \
  --task-definition mwaa-ecs-anywhere-christmas-bonus:5
```

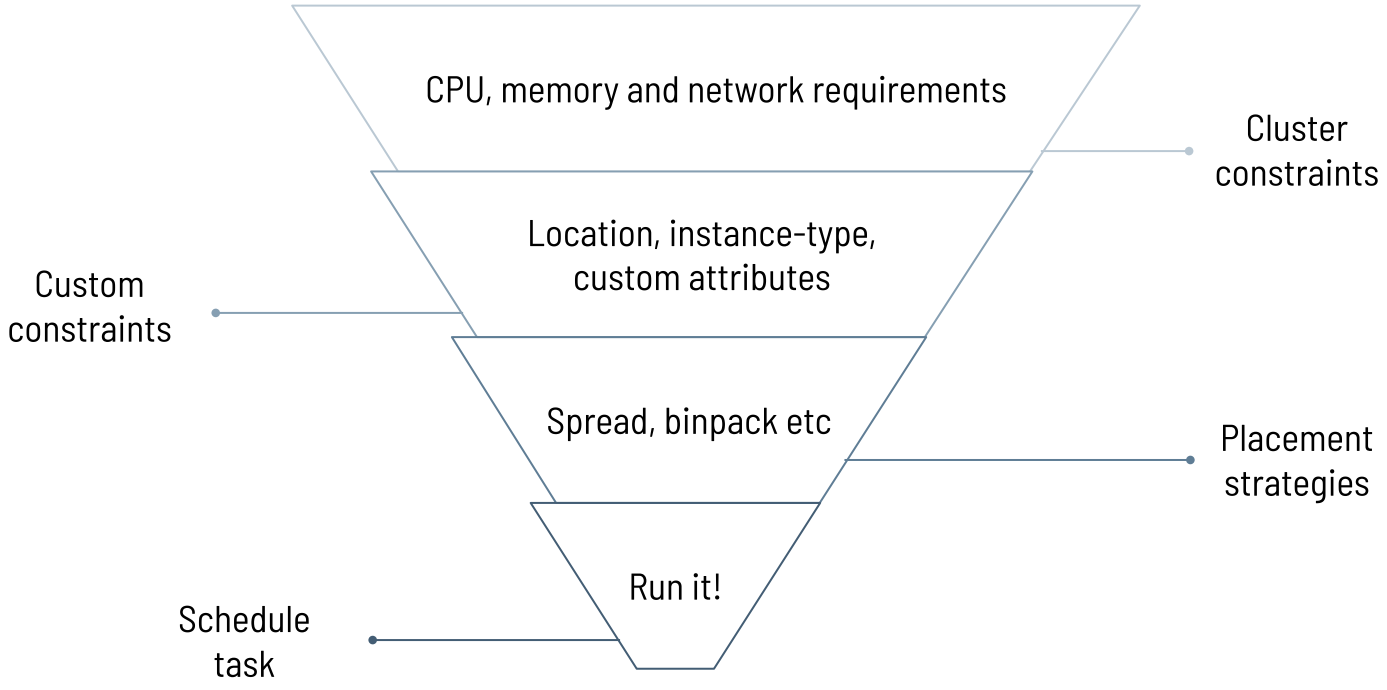
Check bonus task (container) logs again

```
2023-06-16 10:35:19 - INFO - Running on host 'tax.host'
2023-06-16 10:35:19 - INFO - Reading CSV file '/data/employee_data.csv'
2023-06-16 10:35:19 - INFO - Creating the yearly salary with bonus report at '/tmp/salary_with_bonus.csv'
2023-06-16 10:35:22 - INFO - Uploading local file '/tmp/salary_with_bonus.csv' to 's3://mwaa-ecs-anywhere-
bucket/bonus/salary_with_bonus.csv'
```

Cross-instance scheduling



ECS task scheduling



ECS container instance custom attributes

```
aws ecs put-attributes \
  --cluster cluster \
  --attributes name=purpose,value=bonus,targetId=BONUS_INSTANCE_ID
```

```
aws ecs put-attributes \
  --cluster cluster \
  --attributes name=purpose,value=tax,targetId=TAX_INSTANCE_ID
```

ECS container instance custom attributes

```
aws ecs put-attributes \
  --cluster cluster \
  --attributes name=purpose,value=bonus,targetId=BONUS_INSTANCE_ID
```

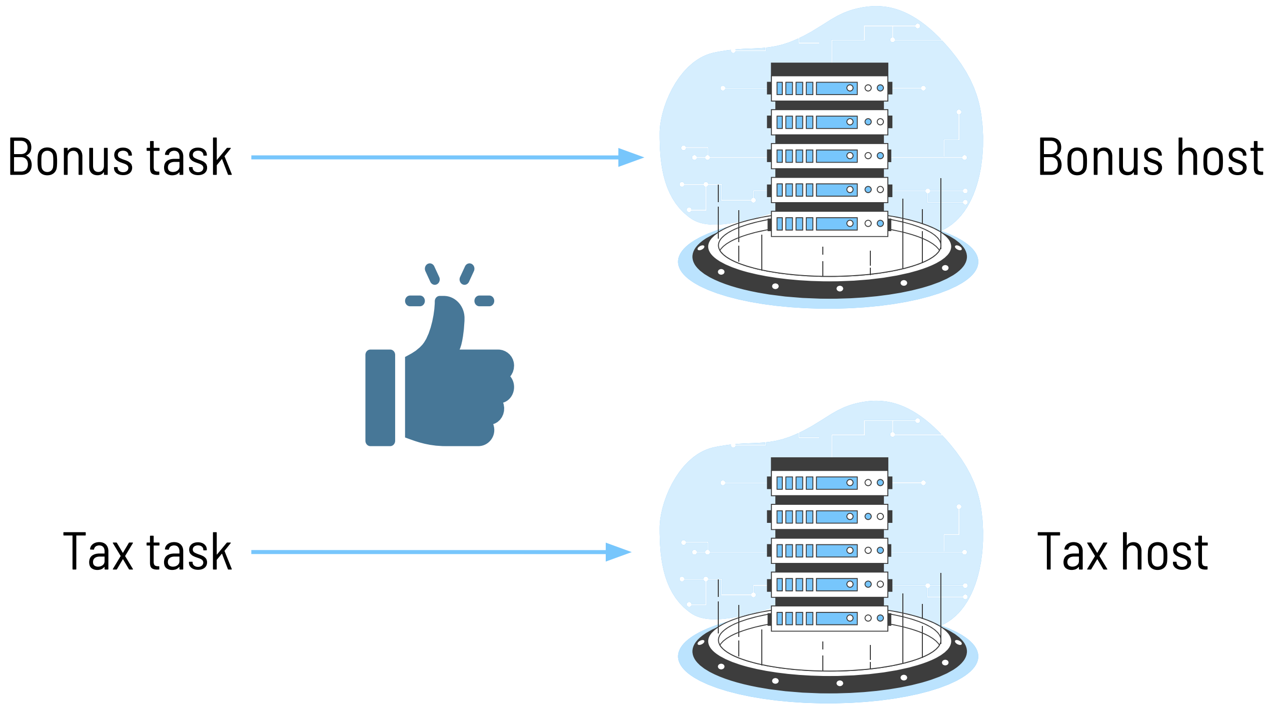
```
aws ecs put-attributes \
  --cluster cluster \
  --attributes name=purpose,value=tax,targetId=TAX_INSTANCE_ID
```

Run tasks with custom attributes

```
aws ecs run-task --cluster "cluster" \
  --count 1 \
  --launch-type EXTERNAL \
  --task-definition mwaa-ecs-anywhere-christmas-bonus:5 \
  --placement-constraints type="memberOf",expression="attribute:purpose==bonus"
```

```
aws ecs run-task --cluster "cluster" \
  --count 1 \
  --launch-type EXTERNAL \
  --task-definition mwaa-ecs-anywhere-yearly-tax:5 \
  --placement-constraints type="memberOf",expression="attribute:purpose==tax"
```

Proper scheduling



01

Prepare ETL scripts

Christmas bonus and amount of taxes to be paid.

02

Run ETL scripts standalone

To verify they work.

03

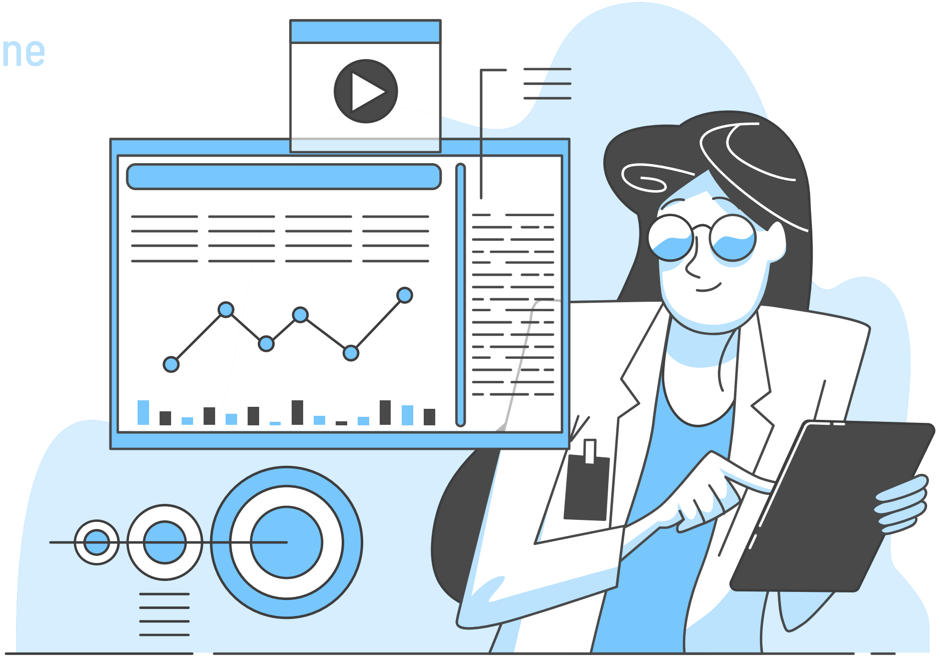
Containerize them

So they are ready to run on ECS.

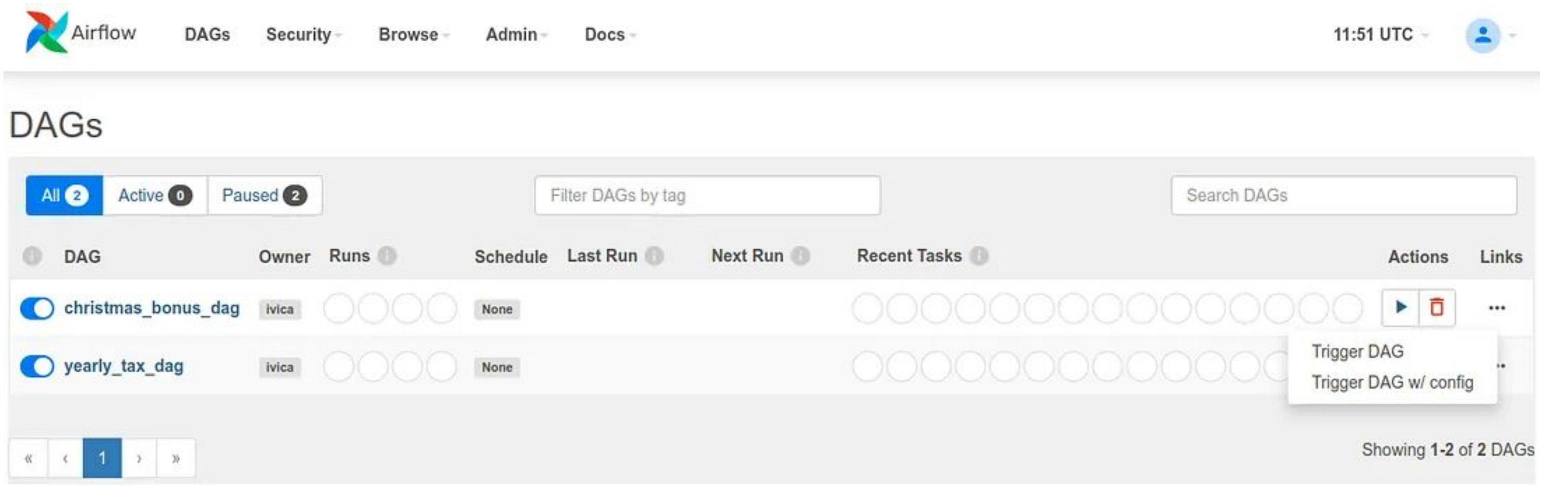
04

Orchestrate with Airflow

For fun and profit!



Airflow UI with DAGs



Airflow DAGs Security Browse Admin Docs 11:51 UTC

DAGs

All 2 | Active 0 | Paused 2

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
christmas_bonus_dag	ivica		None					•••
yearly_tax_dag	ivica		None					

Showing 1-2 of 2 DAGs

Our good friend, Airflow ECS operator

```python
task = EcsRunTaskOperator(
    task_id="my_task_id",
    dag=dag,
    cluster="my_cluster",
    task_definition="task_definition:2",
)
```

Christmas bonus DAG

```
with DAG("christmas_bonus_dag") as dag:
    bonus = EcsRunTaskOperator(
        task_id="calculate_christmas_bonus",
        dag=dag,
        cluster="cluster",
        task_definition="mwaa-ecs-anywhere-christmas-bonus:5",
        launch_type="EXTERNAL",
        placement_constraints=[
            {
                "type": "memberOf",
                "expression": "attribute:purpose==bonus"
            }
        ]
    )
```

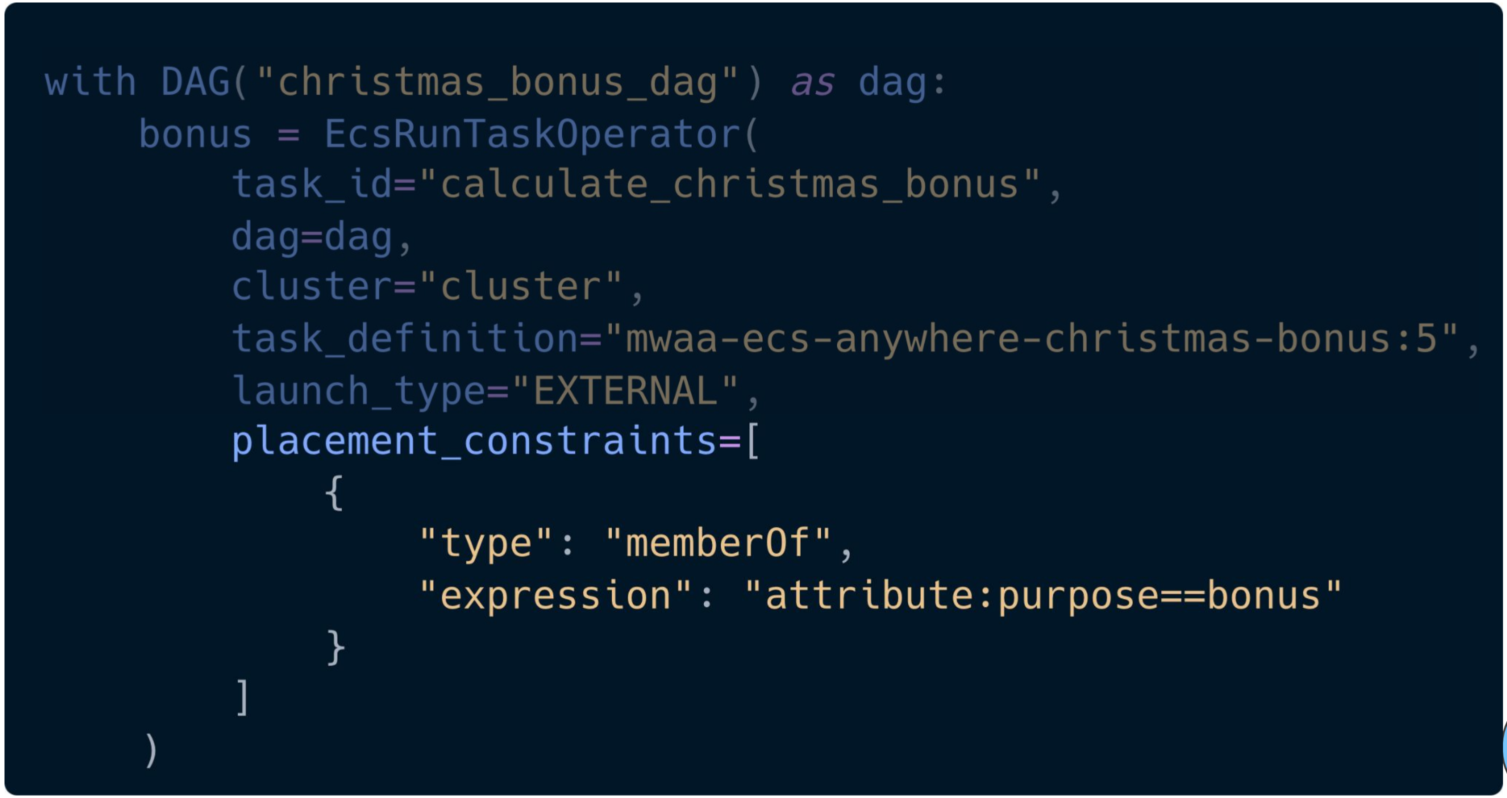
Christmas bonus DAG

```
with DAG("christmas_bonus_dag") as dag:
    bonus = EcsRunTaskOperator(
        task_id="calculate_christmas_bonus",
        dag=dag,
        cluster="cluster",
        task_definition="mwaa-ecs-anywhere-christmas-bonus:5",
        launch_type="EXTERNAL",
        placement_constraints=[
            {
                "type": "memberOf",
                "expression": "attribute:purpose==bonus"
            }
        ]
    )
```

Christmas bonus DAG

```
with DAG("christmas_bonus_dag") as dag:
    bonus = EcsRunTaskOperator(
        task_id="calculate_christmas_bonus",
        dag=dag,
        cluster="cluster",
        task_definition="mwaa-ecs-anywhere-christmas-bonus:5",
        launch_type="EXTERNAL",
        placement_constraints=[
            {
                "type": "memberOf",
                "expression": "attribute:purpose==bonus"
            }
        ]
    )
```

Christmas bonus DAG

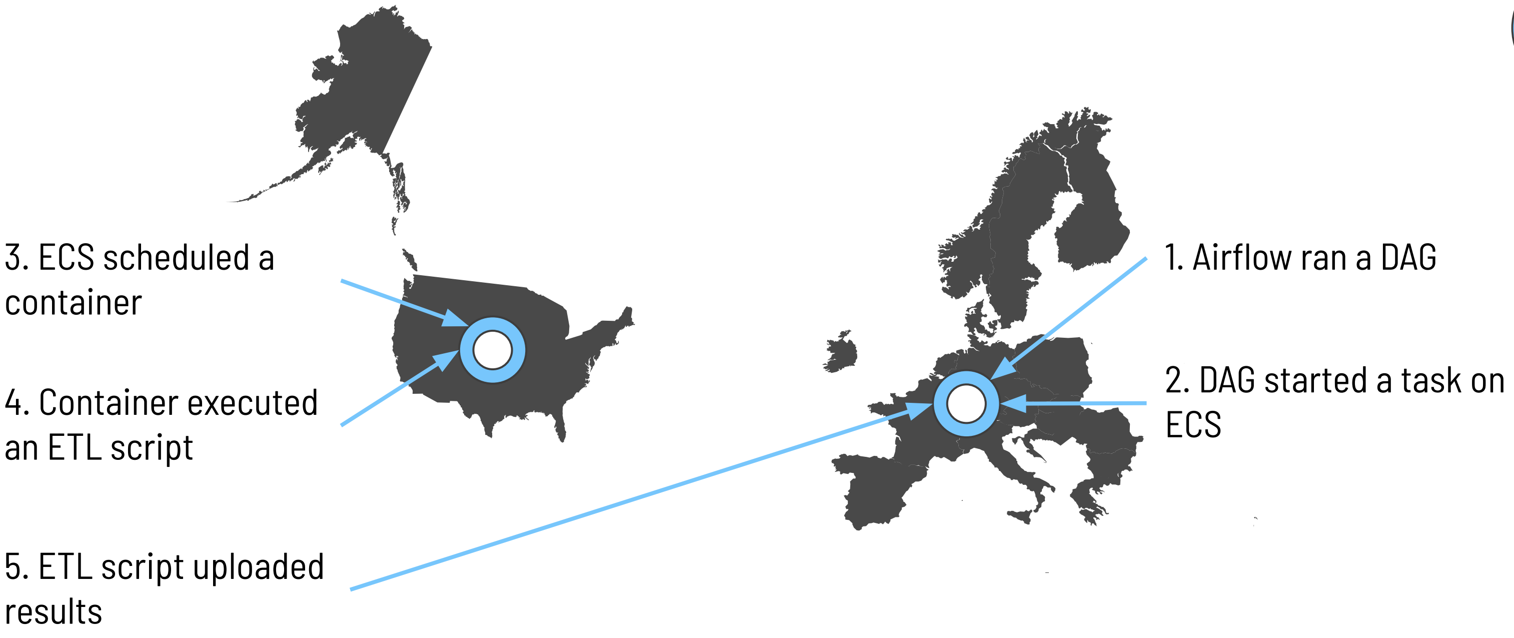


```python
with DAG("christmas_bonus_dag") as dag:
    bonus = EcsRunTaskOperator(
        task_id="calculate_christmas_bonus",
        dag=dag,
        cluster="cluster",
        task_definition="mwaa-ecs-anywhere-christmas-bonus:5",
        launch_type="EXTERNAL",
        placement_constraints=[
            {
                "type": "memberOf",
                "expression": "attribute:purpose==bonus"
            }
        ]
    )
```

Christmas bonus DAG execution

```
--------------------------------------------------------------------------------
2023-06-16 10:41:24 - INFO - Starting attempt 1 of 1
--------------------------------------------------------------------------------
2023-06-16 10:41:25 - INFO - Running ECS Task - mwaa-ecs-anywhere-christmas-bonus:5 - on cluster cluster
2023-06-16 10:41:25 - INFO - ECS Task started
2023-06-16 10:41:25 - INFO - ECS task ID is: a7a24ef8c88649619abce42d25efd7f1
2023-06-16 10:41:37 - INFO - ECS Task stopped check status
2023-06-16 10:41:37 - INFO - ECS Task has been successfully executed
2023-06-16 10:41:37 - INFO - Marking task as SUCCESS. dag_id=christmas_bonus_dag
2023-06-16 10:41:37 - INFO - Task exited with return code 0
```

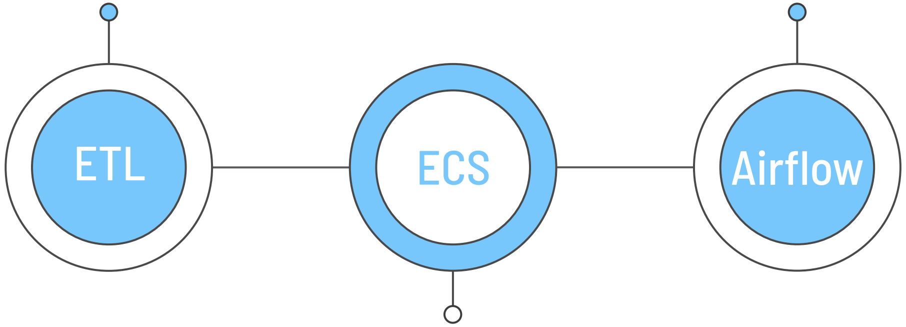
What happened just now?


1. Airflow ran a DAG
2. DAG started a task on ECS
3. ECS scheduled a container
4. Container executed an ETL script
5. ETL script uploaded results

Recap

Unavoidable and necessary

Containers are boosters 10 years later



Use the right tool for the job and understand trade-offs

Technical solutions often have non-technical problems, and vice versa

Thanks!

Do you have any questions?