CIRCLE

Airflow Blockchain Use Case: Testing, GitOps and Learnings

Nathaniel Rose

Nate Rose

- Staff Engineer @ Circle
- Data Platform
- @naterose2 𝕏
- ex-Microsoft & ex-Ripple

Agenda

- What is Circle
- Our Data Platform Problem
- The Proposal
- Automating CICD
- QA Testing DAGs
- Managing Vars & Configs
- Improvements & Astronomer

CIRCLE DEVELOPERS

What is Circle?

USDC

Dollar stability

Stable value

Used internationally

*Regulated

+

Blockchain efficiency

24/7 near-instant settlement

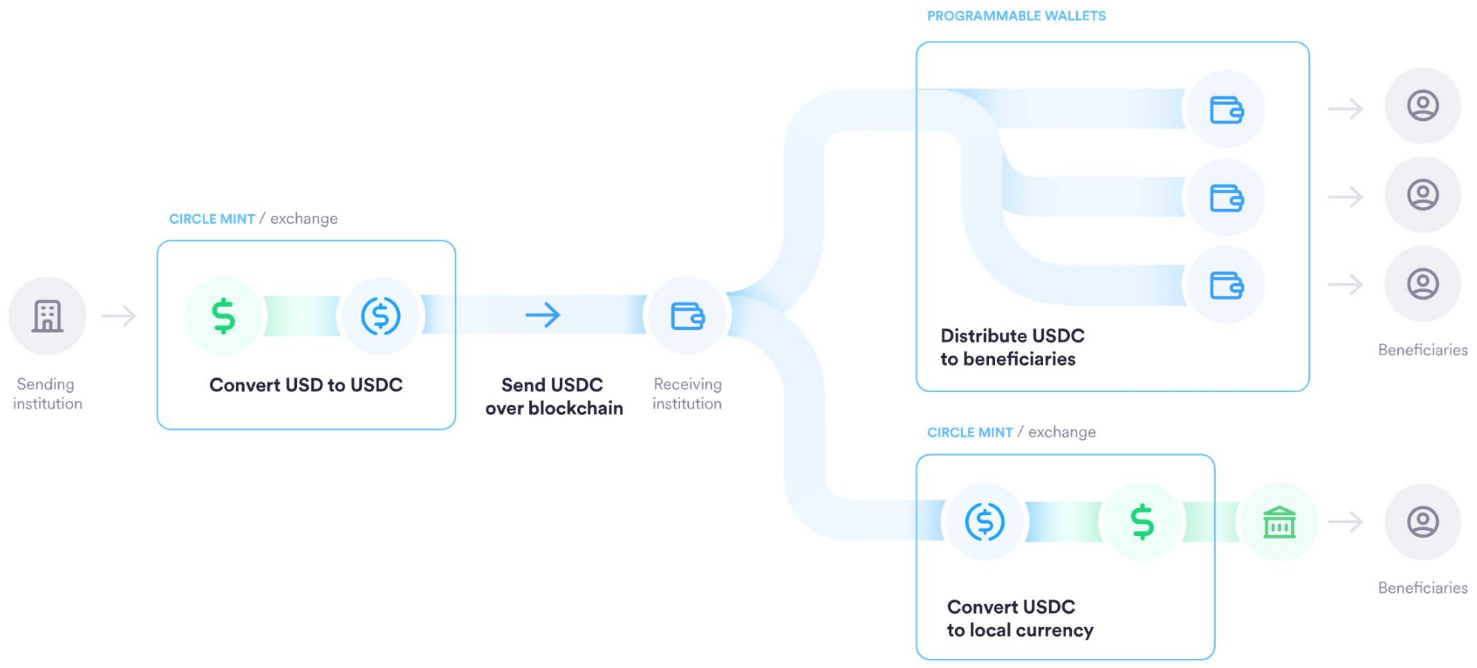
Fewer intermediaries

Lower costs

Real-time traceability

*A list of Circle's licenses can be found here

What We Do


CIRCLE MINT / exchange
Sending institution
Convert USD to USDC
Send USDC over blockchain
Receiving institution
PROGRAMMABLE WALLETS
Distribute USDC to beneficiaries
Beneficiaries
CIRCLE MINT / exchange
Convert USDC to local currency
Beneficiaries

Circle's full-stack platform for building on-chain

Programmable Wallets

Flexible, secure wallet infrastructure for blockchain use cases.

Smart Contract Platform

End-to-end tooling for creating, deploying, and managing smart contracts.

Automate funds flows, tokenize assets, and connect with DeFi.

Cross-Chain Transfer Protocol (CCTP)

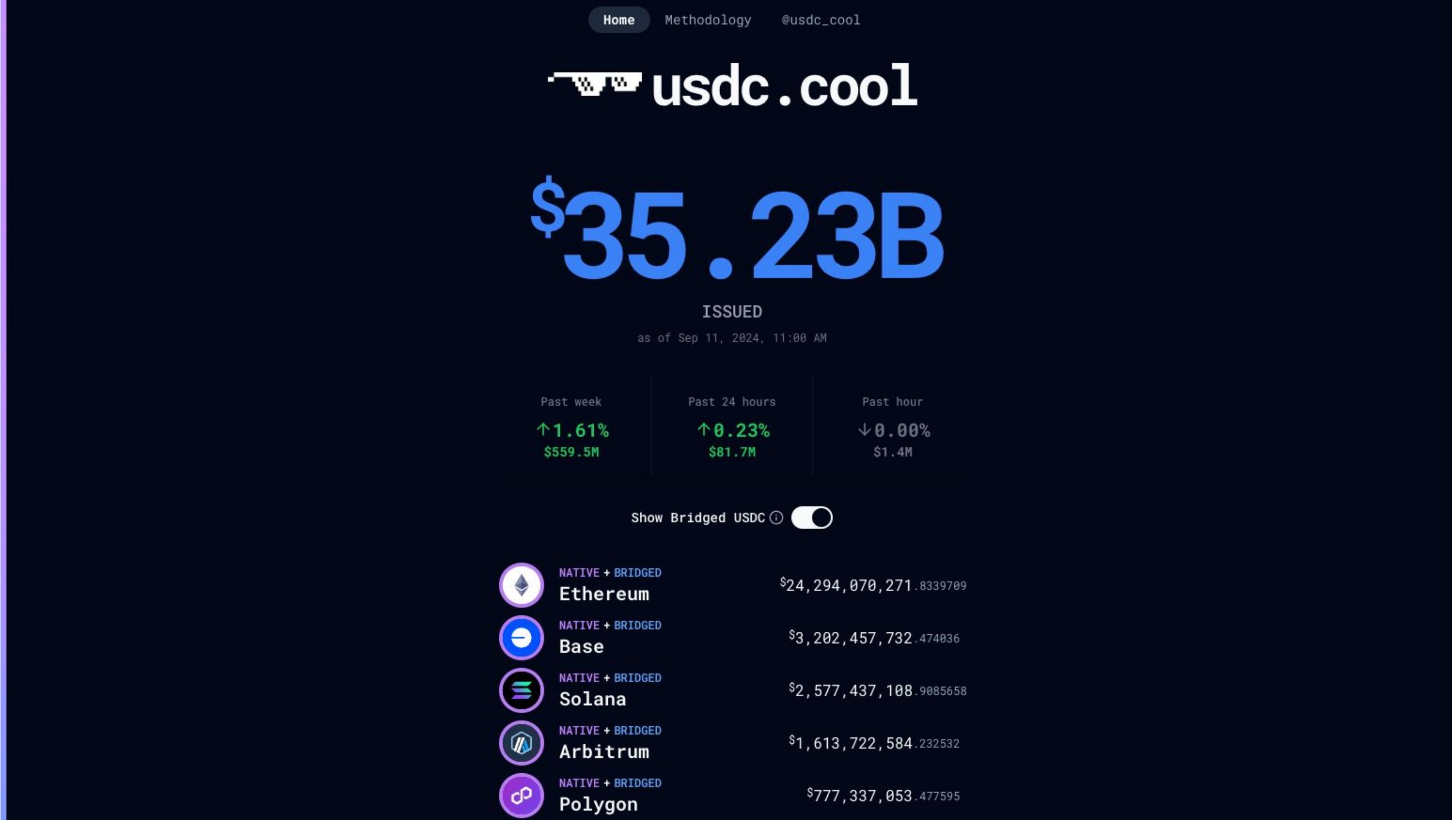
Move USDC securely across supported blockchains for digital asset swaps, seamless user deposits, and on-demand treasury rebalancing.

USDC

US dollar stablecoin that's fully reserved and redeemable 1:1 for USD, powering payments and financial services in your app.

EURC

Euro stablecoin that's fully reserved and redeemable 1:1 for euro, powering payments and financial services in your app.


Home   Methodology   @usdc_cool

usdc.cool

$35.23B

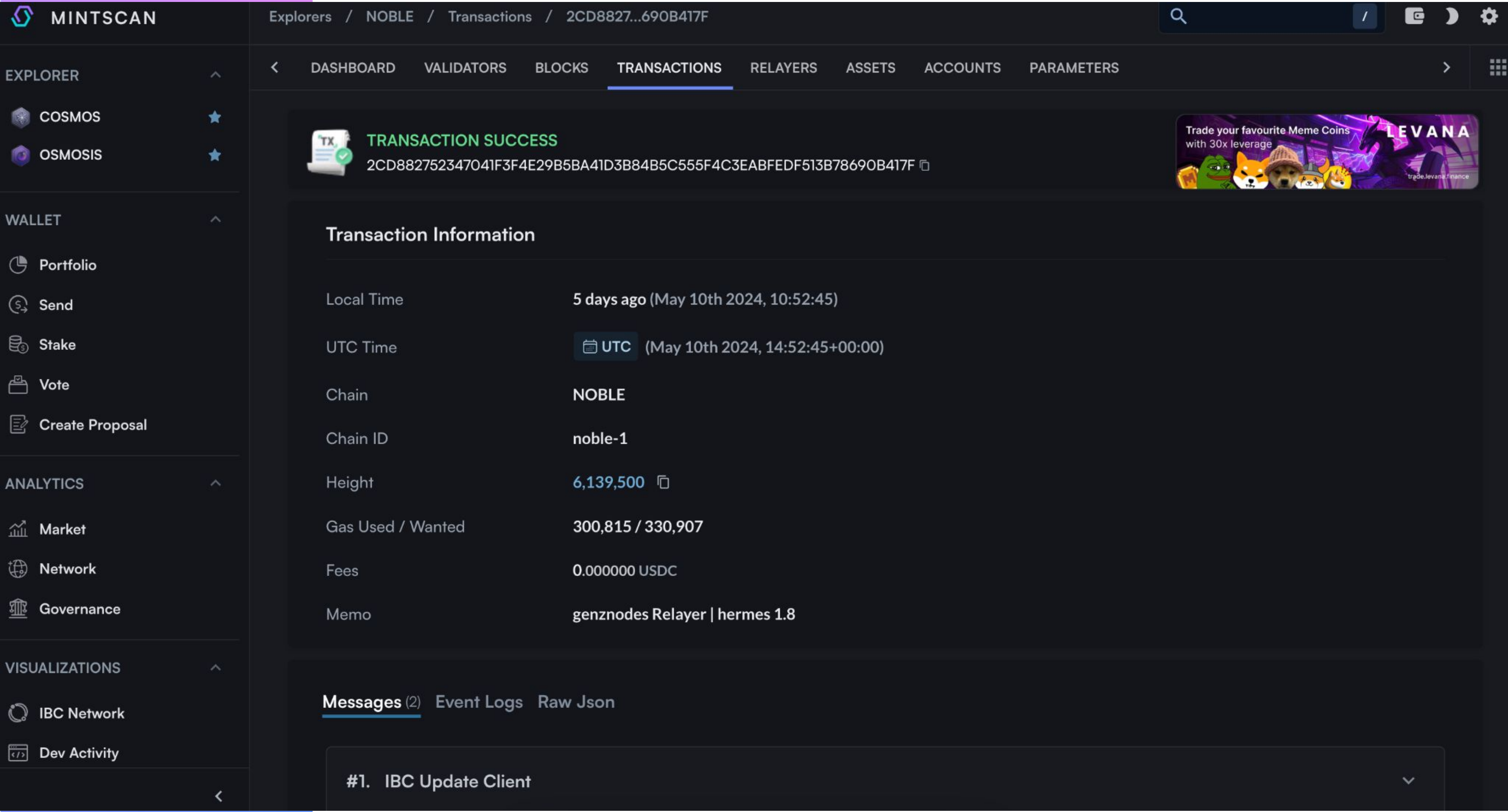
ISSUED

as of Sep 11, 2024, 11:00 AM

Past week	Past 24 hours	Past hour
↑1.61%	↑0.23%	↓0.00%
$559.5M	$81.7M	$1.4M

Show Bridged USDC ⓘ

NATIVE + BRIDGED Ethereum	$24,294,070,271.8339709
NATIVE + BRIDGED Base	$3,202,457,732.474036
NATIVE + BRIDGED Solana	$2,577,437,108.9085658
NATIVE + BRIDGED Arbitrum	$1,613,722,584.232532
NATIVE + BRIDGED Polygon	$777,337,053.477595



MINTSCAN

Explorers / NOBLE / Transactions / 2CD8827...690B417F

EXPLORER
COSMOS
OSMOSIS

WALLET
Portfolio
Send
Stake
Vote
Create Proposal

ANALYTICS
Market
Network
Governance

VISUALIZATIONS
IBC Network
Dev Activity

DASHBOARD VALIDATORS BLOCKS TRANSACTIONS RELAYERS ASSETS ACCOUNTS PARAMETERS

TRANSACTION SUCCESS
2CD882752347041F3F4E29B5BA41D3B84B5C555F4C3EABFEDF513B78690B417F

Trade your favourite Meme Coins with 30x leverage LEVANA trade.levana.finance

Transaction Information

Local Time	5 days ago (May 10th 2024, 10:52:45)
UTC Time	UTC (May 10th 2024, 14:52:45+00:00)
Chain	NOBLE
Chain ID	noble-1
Height	6,139,500
Gas Used / Wanted	300,815 / 330,907
Fees	0.000000 USDC
Memo	genznodes Relayer | hermes 1.8

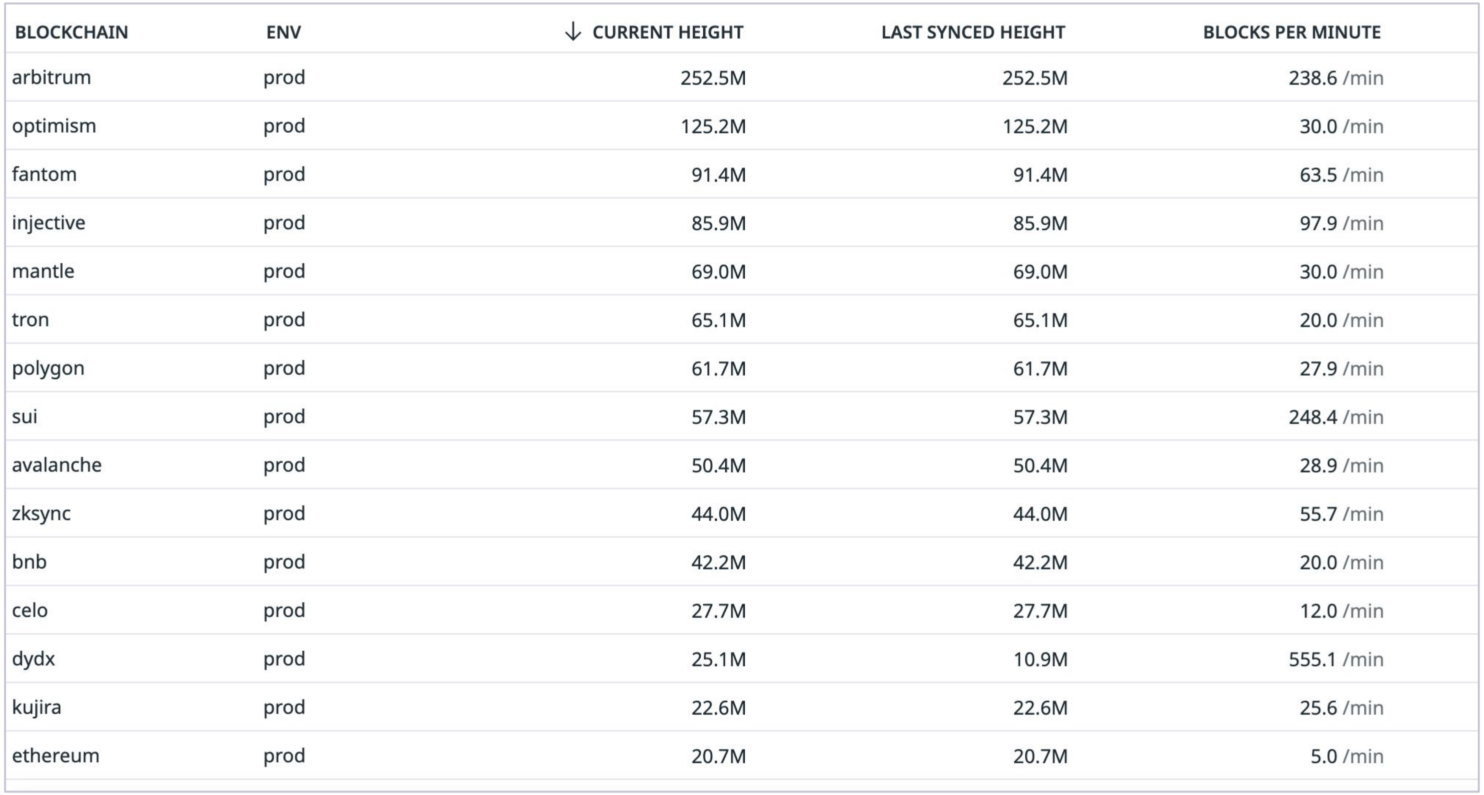
Messages (2) Event Logs Raw Json

#1. IBC Update Client

Data Engineering @ Circle

Scale of our Orchestration

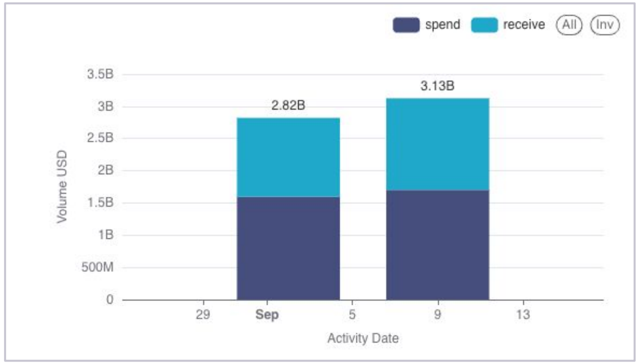
- Total Data Ingested Per Minute: 1.42 GB (approximately)
- Total Data Ingested Per Hour: 85.02 GB (approximately)
- Total Data Ingested Per Day: 2.04063 TB (approximately)
- Total Data Ingested Per Month: 61.21901 TB (approximately)

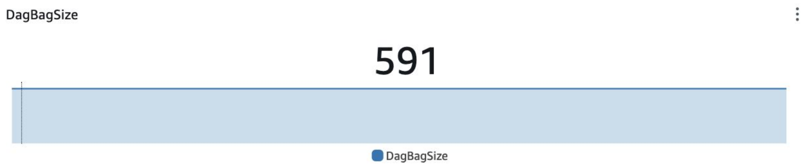


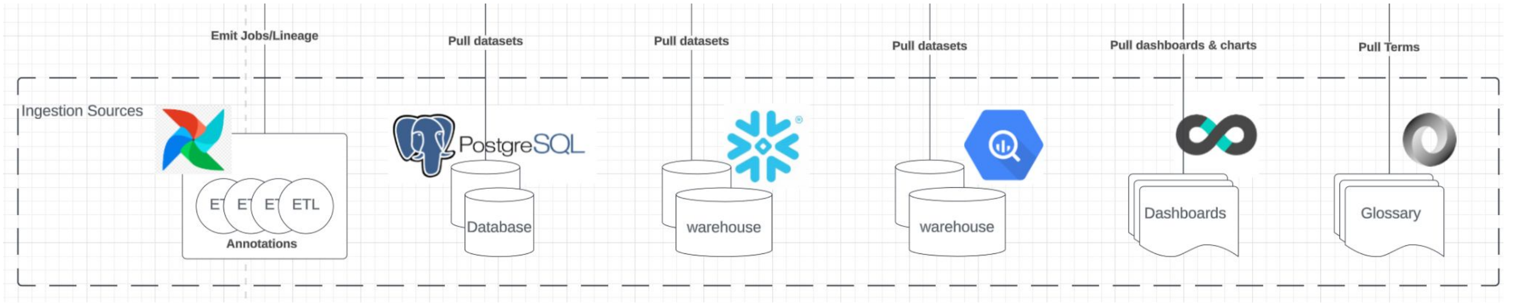
BLOCKCHAIN	ENV	↓ CURRENT HEIGHT	LAST SYNCED HEIGHT	BLOCKS PER MINUTE
arbitrum	prod	252.5M	252.5M	238.6 /min
optimism	prod	125.2M	125.2M	30.0 /min
fantom	prod	91.4M	91.4M	63.5 /min
injective	prod	85.9M	85.9M	97.9 /min
mantle	prod	69.0M	69.0M	30.0 /min
tron	prod	65.1M	65.1M	20.0 /min
polygon	prod	61.7M	61.7M	27.9 /min
sui	prod	57.3M	57.3M	248.4 /min
avalanche	prod	50.4M	50.4M	28.9 /min
zksync	prod	44.0M	44.0M	55.7 /min
bnb	prod	42.2M	42.2M	20.0 /min
celo	prod	27.7M	27.7M	12.0 /min
dydx	prod	25.1M	10.9M	555.1 /min
kujira	prod	22.6M	22.6M	25.6 /min
ethereum	prod	20.7M	20.7M	5.0 /min

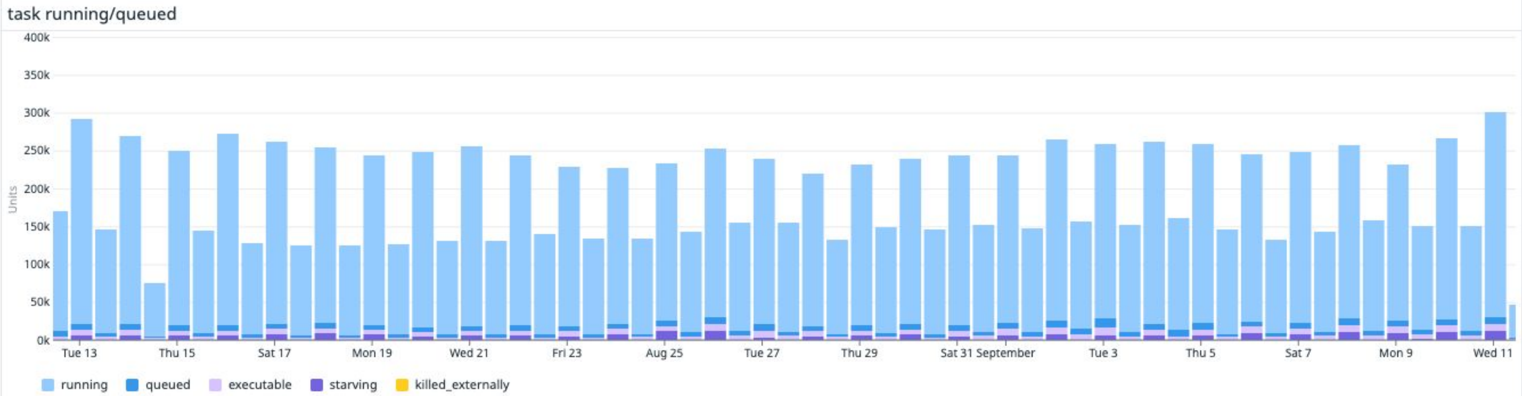
Data Engineering @ Circle

Scale of our Orchestration

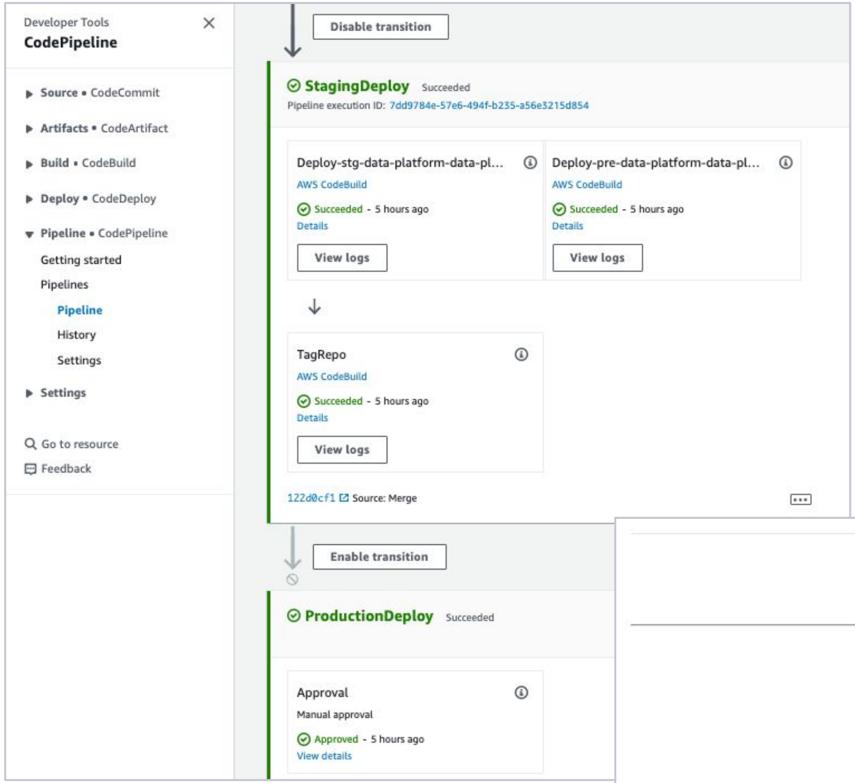








Problems



Developer Tools
CodePipeline

- Source • CodeCommit
- Artifacts • CodeArtifact
- Build • CodeBuild
- Deploy • CodeDeploy
- Pipeline • CodePipeline
  - Getting started
  - Pipelines
    - Pipeline
    - History
    - Settings
- Settings

Go to resource
Feedback

Disable transition

StagingDeploy Succeeded
Pipeline execution ID: 7dd9784e-57e6-494f-b235-a56e3215d854

Deploy-stg-data-platform-data-pl...
AWS CodeBuild
Succeeded - 5 hours ago
Details
View logs

Deploy-pre-data-platform-data-pl...
AWS CodeBuild
Succeeded - 5 hours ago
Details
View logs

TagRepo
AWS CodeBuild
Succeeded - 5 hours ago
Details
View logs

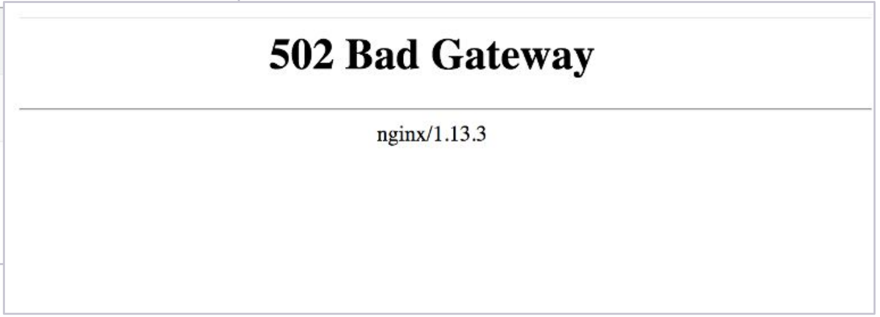
122d0cf1 Source: Merge

Enable transition

ProductionDeploy Succeeded

Approval
Manual approval
Approved - 5 hours ago
View details

More visibility into approvals and centralization of our code pipelines



502 Bad Gateway

nginx/1.13.3

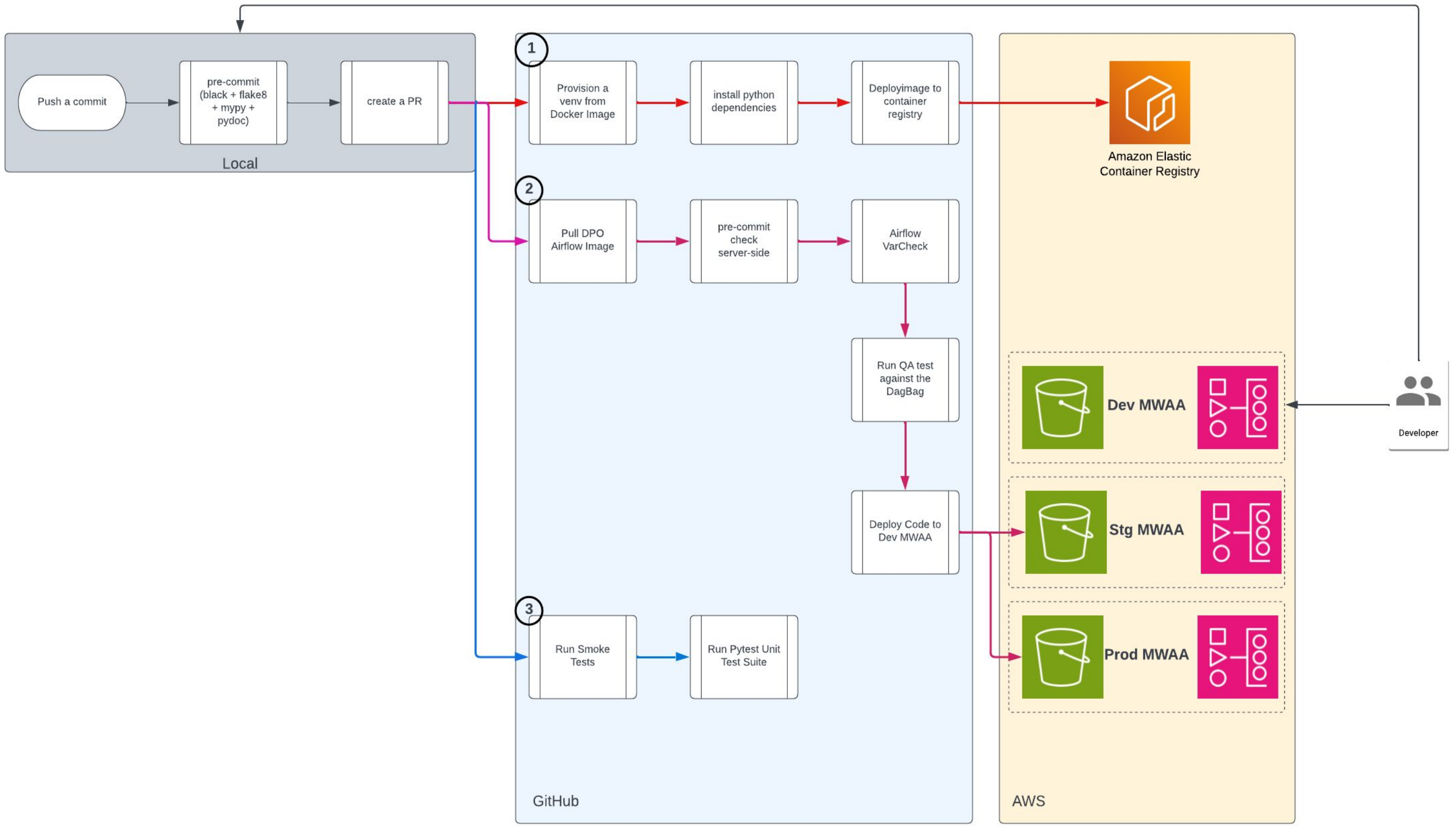
Improved ways to assure our environment configuration parity across MWAA environments

Reduce frequency of Web Server outages

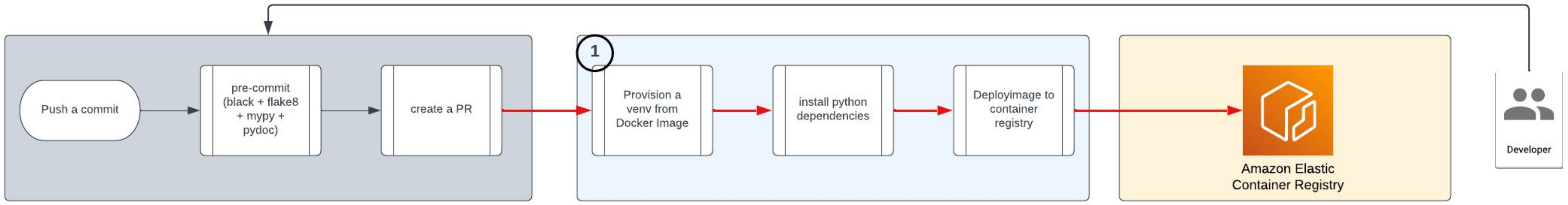
Objective

- Centralize the process of deployment, approvals and other key development cycle pillars to Github Actions in our Orchestration repo.
- Introduce a static testing framework to enable quality assurance, operator, smoke and core component testing to increase due diligence on Airflow upgrades
- Automate the continuous deployment of airflow.cfg , critical airflow environment variables, source code DAGs.

Architecture Proposal


Push a commit
pre-commit (black + flake8 + mypy + pydoc)
create a PR
Local
1
Provision a venv from Docker Image
install python dependencies
Deployimage to container registry
Amazon Elastic Container Registry
2
Pull DPO Airflow Image
pre-commit check server-side
Airflow VarCheck
Run QA test against the DagBag
Deploy Code to Dev MWAA
3
Run Smoke Tests
Run Pytest Unit Test Suite
GitHub
Dev MWAA
Stg MWAA
Prod MWAA
Developer
AWS

Pipeline 1 - Circle Airflow Image

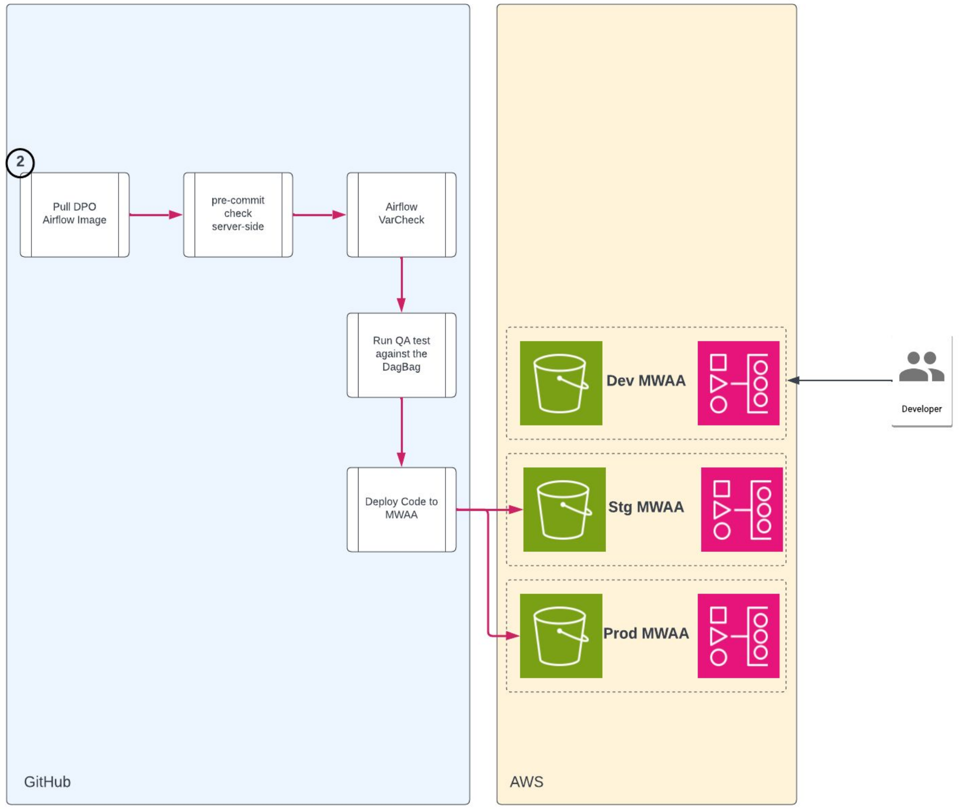


Push a commit
pre-commit (black + flake8 + mypy + pydoc)
create a PR
1
Provision a venv from Docker Image
install python dependencies
Deployimage to container registry
Amazon Elastic Container Registry
Developer

- Triggers on changes to **requirements.txt**
- **Installs our common python wheel and dependencies** on a docker image for use in pipeline and locally
- **Versioned to your pull request**
- Solely used for development purposes and environment mobility outside of MWAA

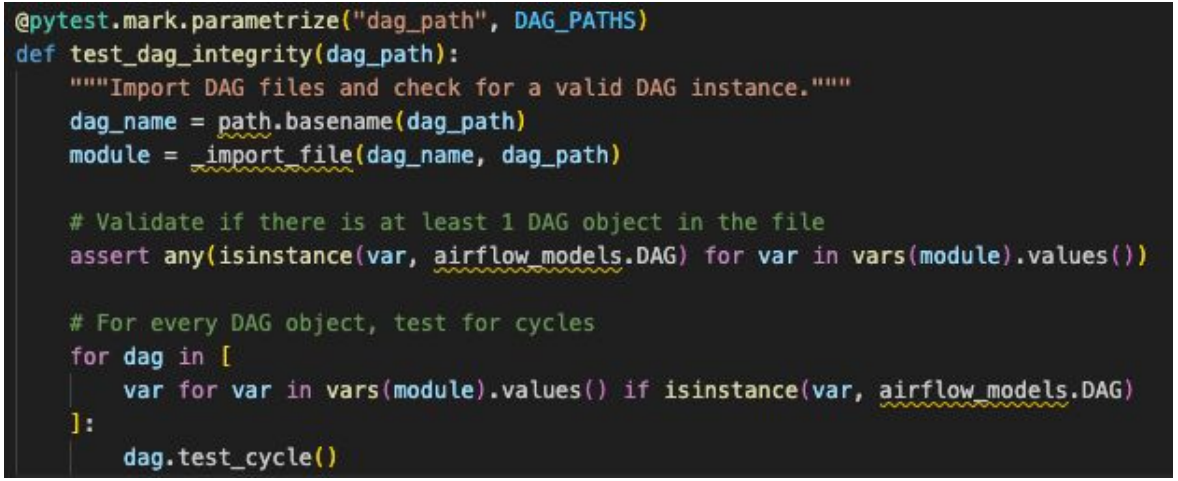
Pipeline 2 - CICD

- Triggers on changes to our **DAG src** commits
- Check for **linting using pre-commit server side** task
- Verify variables added are in all stages
- **Quality assurance test** to check the dags for configs
- **Deploy code to** respective MWAA S3 buckets



Quality Assurance Test

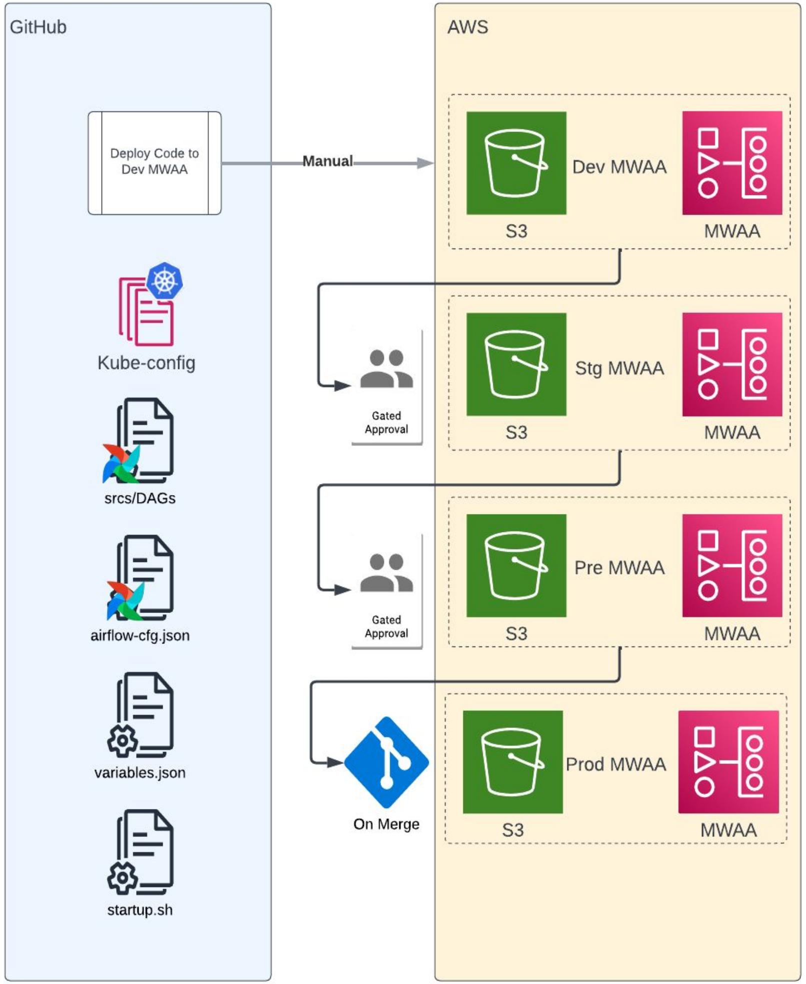
- Checks all DAGs in the dagbag for case conditions.
- Example 1: Check that all DAG Owners are set
- Example 2: Check the duration to parse the DAG
- Example 3: Check if `dag_id == file_name`



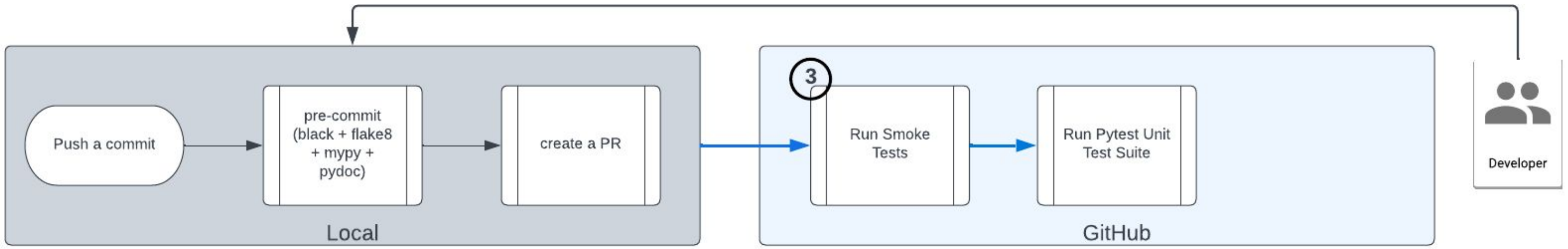
```python
@pytest.mark.parametrize("dag_path", DAG_PATHS)
def test_dag_integrity(dag_path):
    """Import DAG files and check for a valid DAG instance."""
    dag_name = path.basename(dag_path)
    module = _import_file(dag_name, dag_path)

    # Validate if there is at least 1 DAG object in the file
    assert any(isinstance(var, airflow_models.DAG) for var in vars(module).values())

    # For every DAG object, test for cycles
    for dag in [
        var for var in vars(module).values() if isinstance(var, airflow_models.DAG)
    ]:
        dag.test_cycle()
```

Continuous Deployment

- Devs still have to **manually add DAGs** to Dev MWAA S3
- **Gated approval** on deployment to staging and pre-prod MWAA
- On merge our environment config deploys to Prod MWAA
- **Startup script** installs variables and airflow.cfg into environments only when we invoke the update through the console. (Have the option to invoke through code)



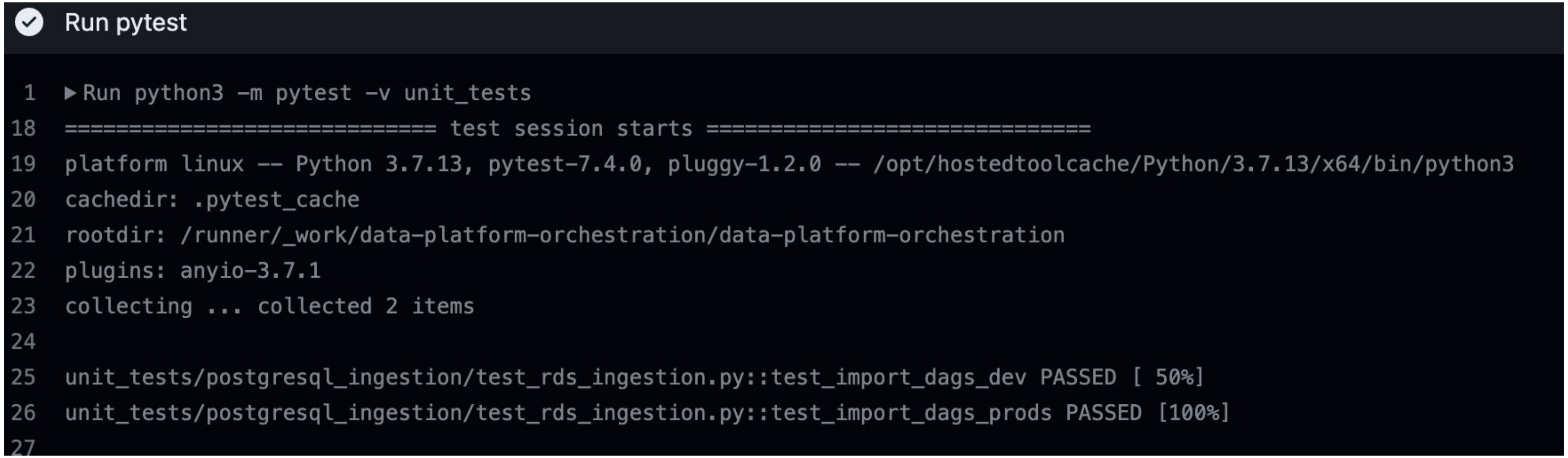
Pipeline 3 - Airflow Orchestration Repo Test Suite


Push a commit
pre-commit (black + flake8 + mypy + pydoc)
create a PR
Local
3
Run Smoke Tests
Run Pytest Unit Test Suite
GitHub
Developer

- Triggers on changes to **specific DAGs (boilerplate)**
- Pulls Airflow developer image to map dependencies to incoming changes
- Smoke tests may include **verifying datahub lineage, additional logic on import**
- Unit Tests **assure our core base operators** function upon each update

PyTest Unit Test Suite

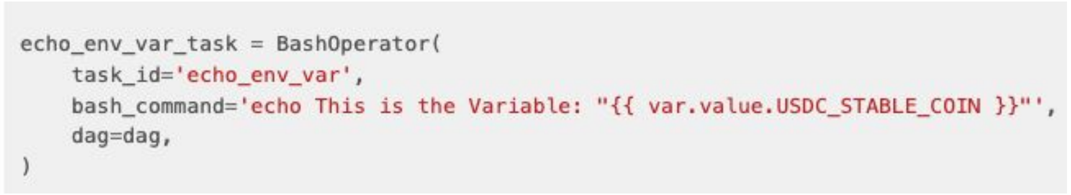
- Checks for import errors in the scheduler
- If dag magical condition is met then pass
- Mocking to meet the needs of the test case
- Currently completed with RDS type dags



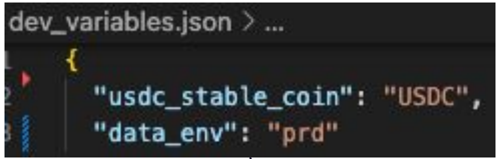
Run pytest

```
▶ Run python3 -m pytest -v unit_tests
============================== test session starts ===============================
platform linux -- Python 3.7.13, pytest-7.4.0, pluggy-1.2.0 -- /opt/hostedtoolcache/Python/3.7.13/x64/bin/python3
cachedir: .pytest_cache
rootdir: /runner/_work/data-platform-orchestration/data-platform-orchestration
plugins: anyio-3.7.1
collecting ... collected 2 items

unit_tests/postgresql_ingestion/test_rds_ingestion.py::test_import_dags_dev PASSED [ 50%]
unit_tests/postgresql_ingestion/test_rds_ingestion.py::test_import_dags_prods PASSED [100%]
```

Environment Variables

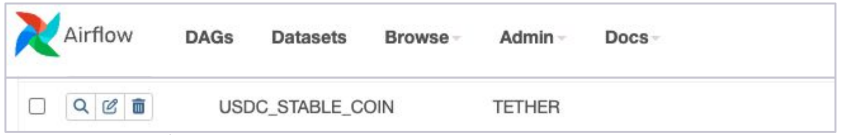


```
echo_env_var_task = BashOperator(
    task_id='echo_env_var',
    bash_command='echo This is the Variable: "{{ var.value.USDC_STABLE_COIN }}"',
    dag=dag,
)
```



dev_variables.json > ...

```
{
  "usdc_stable_coin": "USDC",
  "data_env": "prd"
```

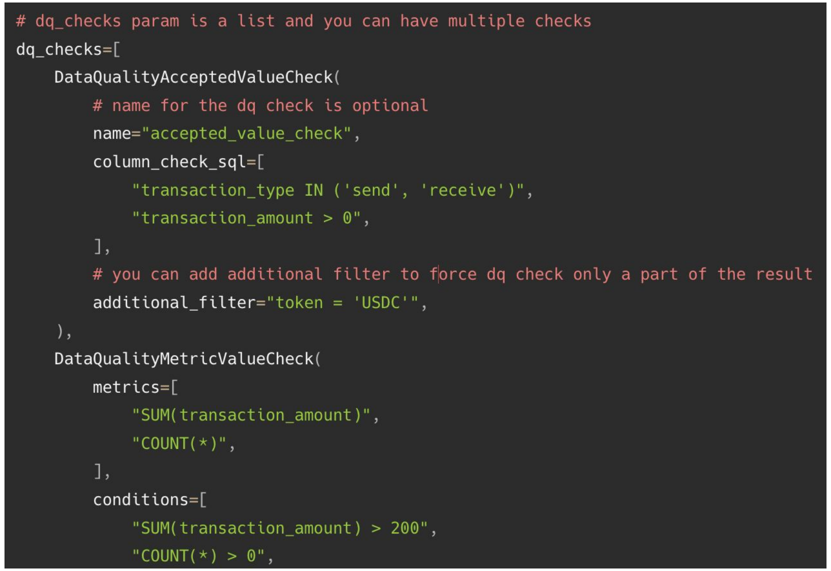


USDC_STABLE_COIN	TETHER	

CICD Var Value	UI Console Var Value	Output
USDC	TETHER	USDC
USDC	None	USDC
None	USDC	None
None*	USDC*	USDC

Learnings: Data Quality Framework

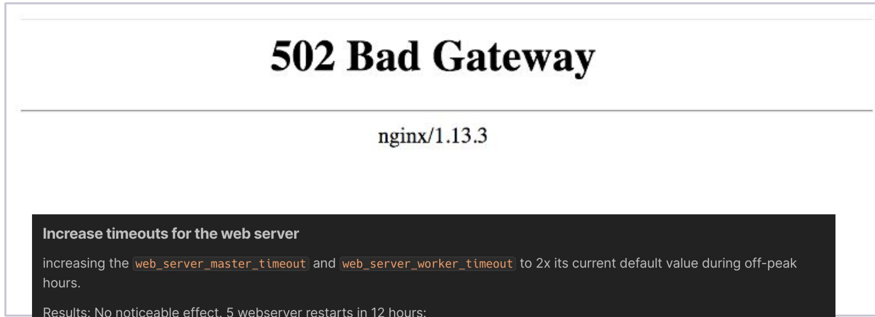
- Create a temp table for the insert query result
- Run dq check on the temp table
- If check passes, will merge the temp table into temp table, then delete the temp table
- If check fails, fail the task and log the failure reason and temp table name. (temp table will be deleted in 7 days)
- Example DQ Checks:
  - `DataQualityDuplicateCheck`
  - `DataQualityAcceptedValueCheck`
  - `DataQualityMetricValueCheck`



```
# dq_checks param is a list and you can have multiple checks
dq_checks=[
    DataQualityAcceptedValueCheck(
        # name for the dq check is optional
        name="accepted_value_check",
        column_check_sql=[
            "transaction_type IN ('send', 'receive')",
            "transaction_amount > 0",
        ],
        # you can add additional filter to force dq check only a part of the result
        additional_filter="token = 'USDC'",
    ),
    DataQualityMetricValueCheck(
        metrics=[
            "SUM(transaction_amount)",
            "COUNT(*)",
        ],
        conditions=[
            "SUM(transaction_amount) > 200",
            "COUNT(*) > 0",
```

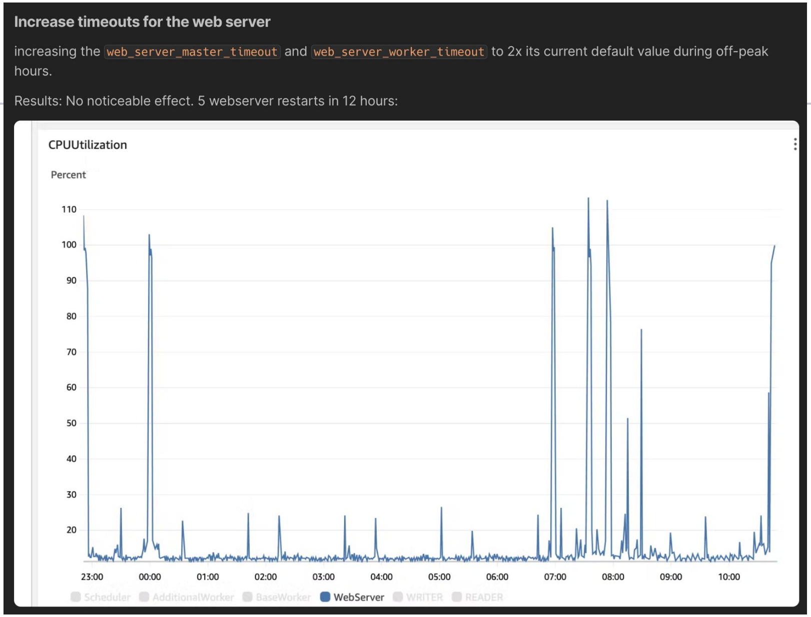
Learnings: Web Server Outages

- Scale Up?
- Increase connection Pool size
- Set `schedule_after_task_execution` to `False`
- Stagger DAG Runs
- Use a connection pooling service like PgBouncer
- Set `celery.pool` equal to `solo`
- MWAA source code was causing WebServer to crash and required an immediate upgrade to a later airflow version.
- **Upgrade to a later stable version**



502 Bad Gateway

nginx/1.13.3



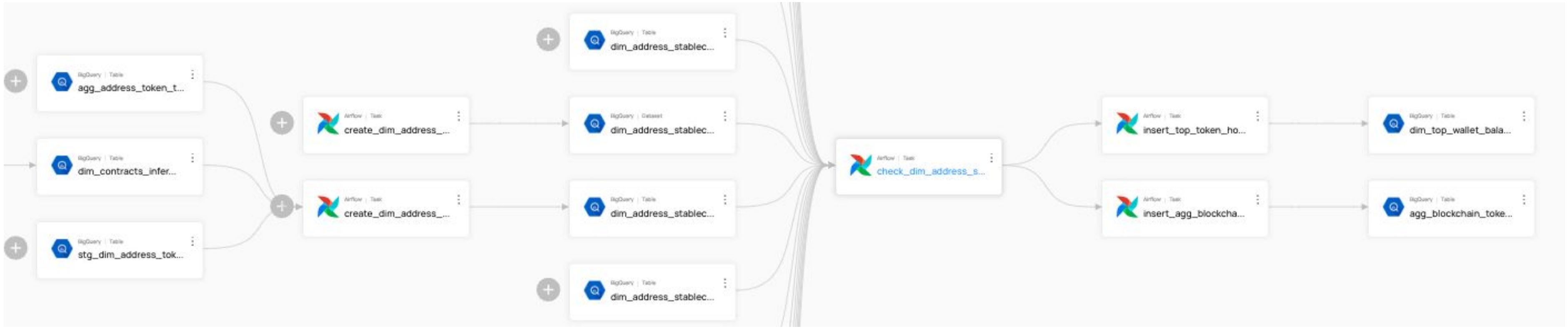
Increase timeouts for the web server

increasing the `web_server_master_timeout` and `web_server_worker_timeout` to 2x its current default value during off-peak hours.

Results: No noticeable effect. 5 webserver restarts in 12 hours:

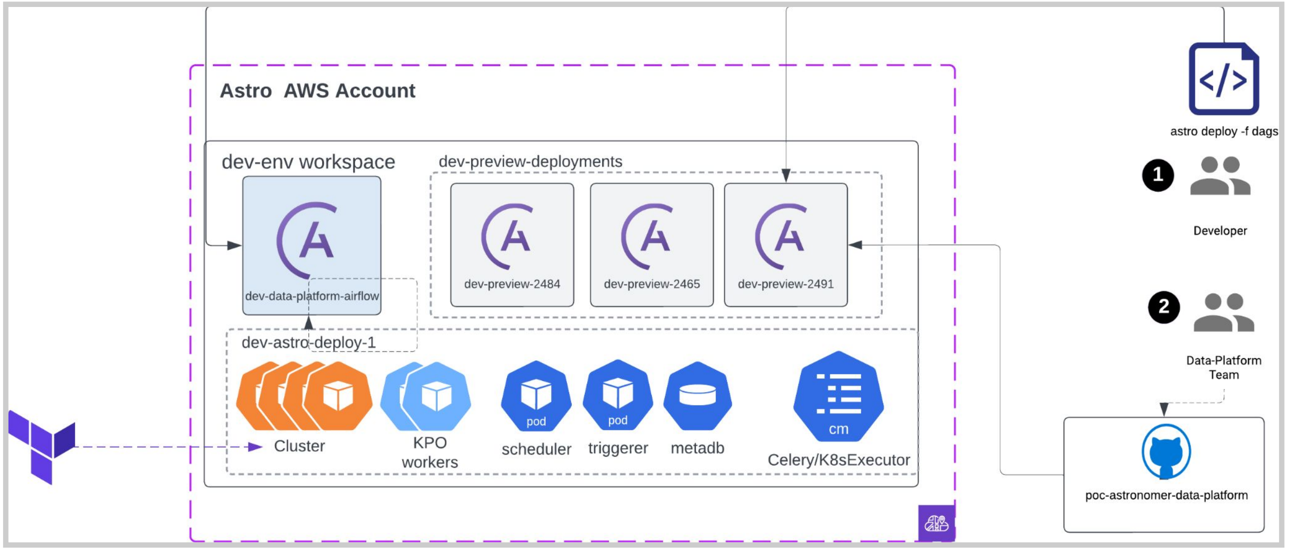
Learnings: Airflow + Data Hub & Lineage

- Notify downstream dependencies for pipeline failures
- Enforce lineage checks for our pipelines
- Capture node graph of our interconnected data platform
- Better discoverability with tagging and labeling scheme

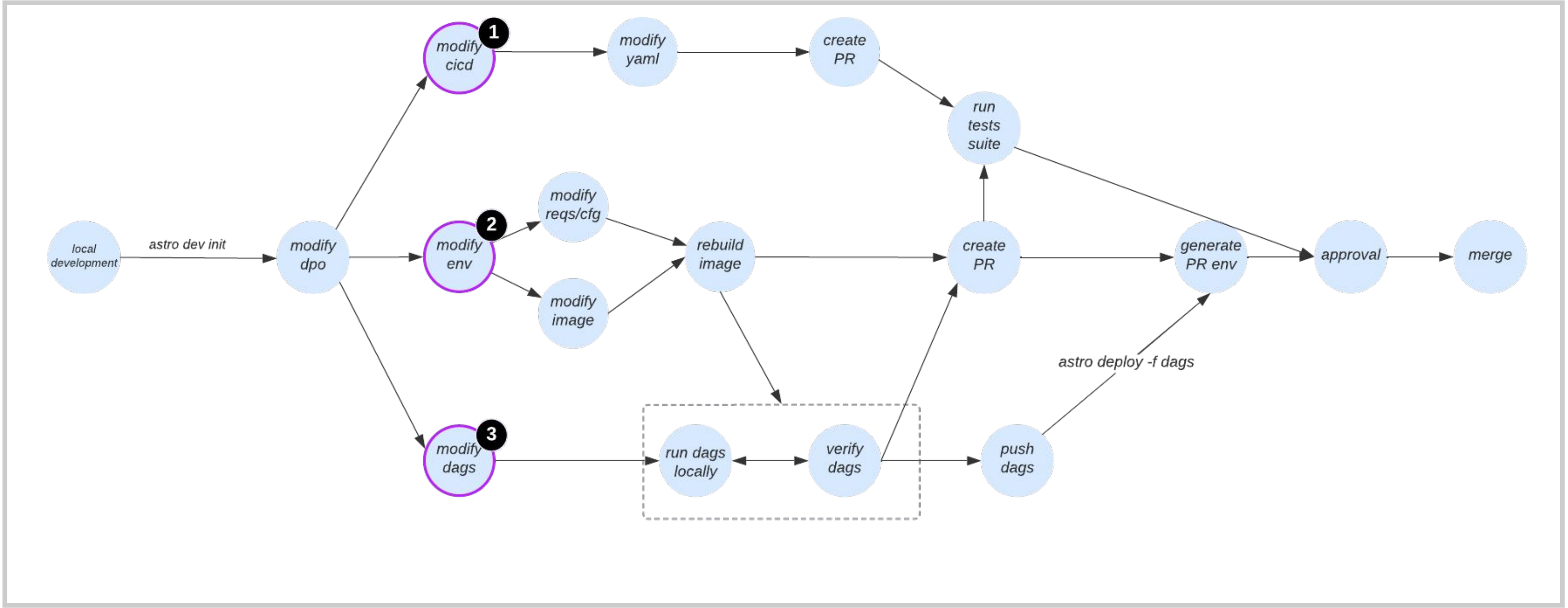


Exploring Astronomer for Managed Airflow

- Kubernetes Executor
- Managing Airflow Environments using Docker Images
- Ephemeral Airflow Environments
- Local Airflow Development
- **Reduced Environments & Cost Effective**

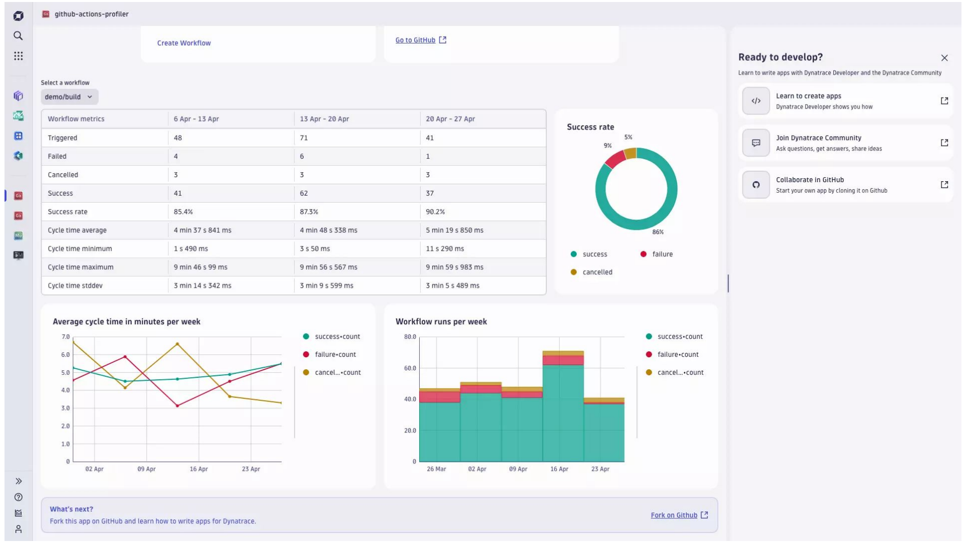


Exploring Astronomer for Managed Airflow



Future Work & Other Tricks

- Exploring Astronomer
- Expanded Unit Testing for Circle maintained Operators
- Variable race condition
- CICD Observability
- Lineage & DataHub
- **“Dataset” States & Backfill intelligence**


github-actions-profiler

Create Workflow

Go to GitHub

Select a workflow

demo/build

Workflow metrics	6 Apr - 13 Apr	13 Apr - 20 Apr	20 Apr - 27 Apr
Triggered	48	71	41
Failed	4	6	1
Cancelled	3	3	3
Success	41	62	37
Success rate	85.4%	87.3%	90.2%
Cycle time average	4 min 37 s 841 ms	4 min 48 s 338 ms	5 min 19 s 850 ms
Cycle time minimum	1 s 490 ms	3 s 50 ms	11 s 290 ms
Cycle time maximum	9 min 46 s 99 ms	9 min 56 s 567 ms	9 min 59 s 983 ms
Cycle time stddev	3 min 14 s 342 ms	3 min 9 s 599 ms	3 min 5 s 489 ms

Ready to develop?

Learn to write apps with Dynatrace Developer and the Dynatrace Community

Learn to create apps
Dynatrace Developer shows you how

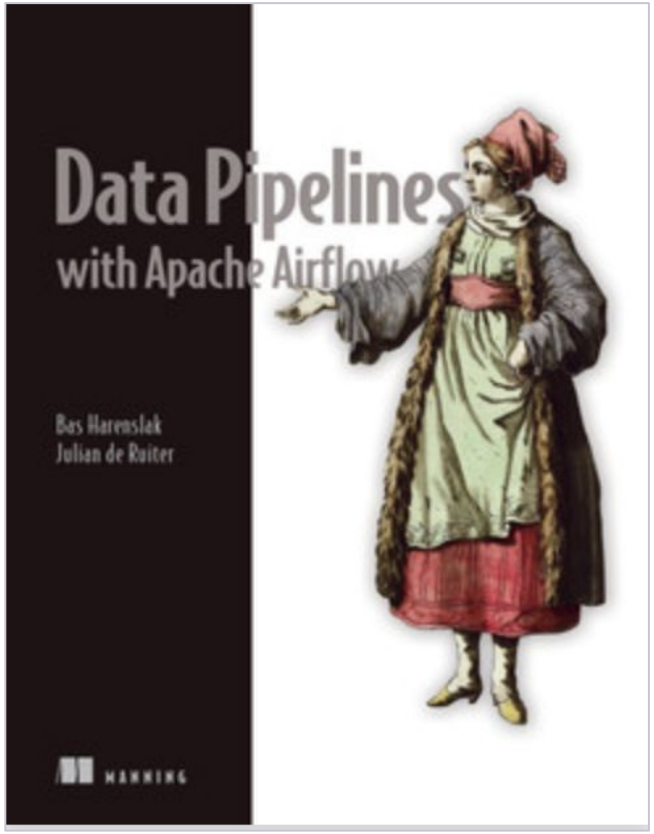
Join Dynatrace Community
Ask questions, get answers, share ideas

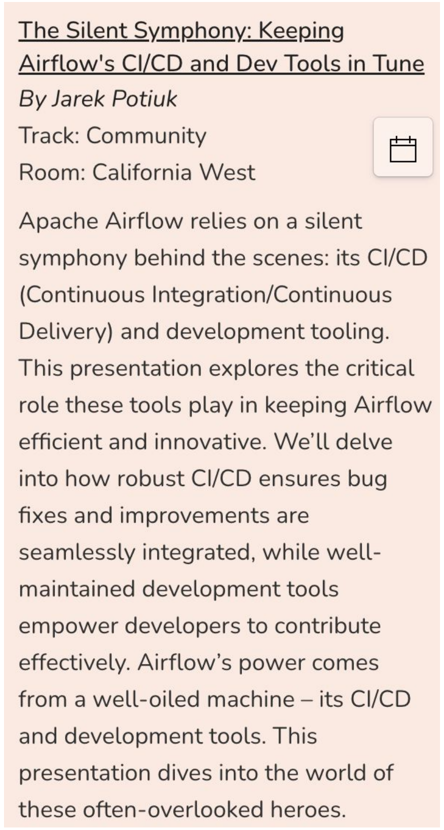
Collaborate in GitHub
Start your own app by cloning it on Github

What's next?
Fork this app on GitHub and learn how to write apps for Dynatrace.

Fork on Github

Additional Resources




The Silent Symphony: Keeping Airflow's CI/CD and Dev Tools in Tune

By Jarek Potiuk

Track: Community

Room: California West

Apache Airflow relies on a silent symphony behind the scenes: its CI/CD (Continuous Integration/Continuous Delivery) and development tooling. This presentation explores the critical role these tools play in keeping Airflow efficient and innovative. We'll delve into how robust CI/CD ensures bug fixes and improvements are seamlessly integrated, while well-maintained development tools empower developers to contribute effectively. Airflow's power comes from a well-oiled machine – its CI/CD and development tools. This presentation dives into the world of these often-overlooked heroes.