



## Semiconductor (Chip) Design Workflow Orchestration with Airflow

Dheeraj Turaga | Senior Staff Engineer @ Qualcomm  
Nicholas Redd | Senior Engineer @ Qualcomm

3.0



# Semiconductor (Chip) Design Workflow Orchestration With Airflow

Dheeraj Turaga | Senior Staff CAD Engineer @ Qualcomm  
Nicholas Redd | Senior Engineer @ Qualcomm





# Agenda

- About Us
- Life Before Airflow
- Airflow @ Qualcomm
  - Dynamic Celery Workers
  - Multiple Executors: Celery + Edge
  - Multi DC Architecture
- Contributions
- Methodologies

# About Us - Qualcomm

We make CPUs!

Qualcomm Oryon CPU – Our Snapdragon SOC Lineup :



Compute >

The industry leader in performance and power efficiency.



Mobile >

Transforming performance with the world's fastest mobile CPU.



Automotive >

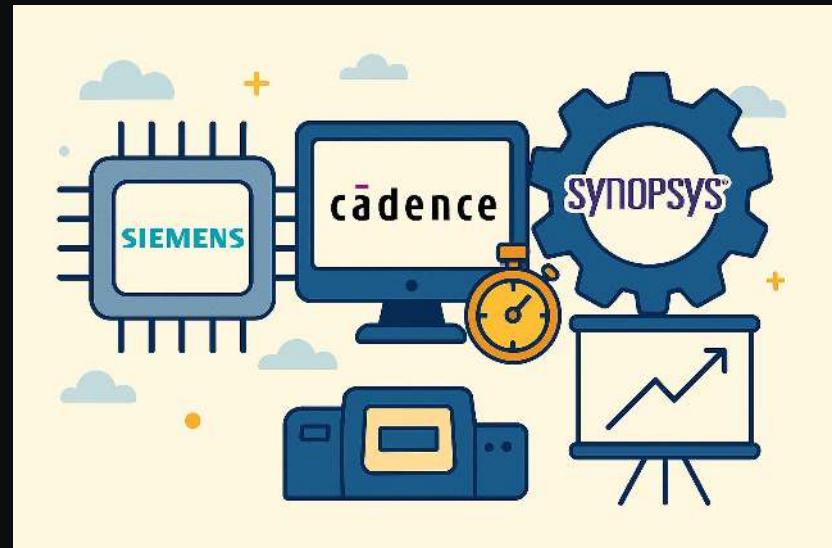
Unlocking a new era of automotive intelligence.

# About Us - EDA Tools

*EDA tools are specialized software used to automate and manage the complex processes of chip design and verification.*

EDA tools demand:

- **Massive Compute:** tasks can run 7+ days, consume >256 GB RAM, generate 100s of GBs of output.
- **Rigid Vendor Constraints:** strict licensing, environment setup, and version dependencies
- **Scale of Orchestration:** Thousands of jobs must be scheduled and managed daily across multiple sites and compute grids.



# EDA in Docker?

**Requires Root Privileges** → Security risk on multi-user clusters.

**No Native Scheduler Integration** → Hard to enforce resource limit policies.

**OverlayFS Overhead** → Slows down I/O-heavy EDA flows (millions of small files).

**Limited Shared Memory & Ulimits** → Manual tweaks needed; admins restrict flags.

**Complex HPC Networking** → MPI, Infiniband need extra config.

**Unstable Host Identity** → Breaks FlexNet licensing.



# SLURM/LSF + Singularity

*SLURM/LSF = Job Scheduler , Singularity = HPC-friendly container*

**Runs as Non-Root** → Secure for shared HPC environments.

**Native Scheduler Hooks** → Accurate resource tracking & fair-share scheduling.

**Direct Host Filesystem Access** → No overlay penalty; ideal for PDKs & scratch.

**Inherits Host Limits** → /dev/shm and ulimits work out-of-the-box.

**Seamless HPC Features** → MPI, GPUs, Infiniband supported natively.

**Preserves Host Identity** → Stable for floating license servers.

# Life Before Airflow

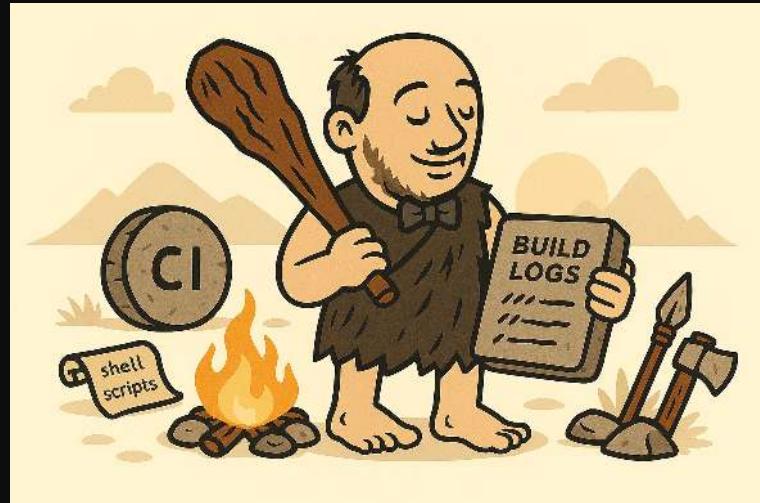
**"Before there was automation, before there was orchestration...  
there was chaos.  
And before there was chaos... there was Jenkins."**



# Life Before Airflow - Jenkins

The Bad:

- **Unstable Executor Nodes:** Frequent failures (resource, network, software, grid) → cascading regressions failures & delays.
- **Fragile Infrastructure:** Node failures require manual recovery; high maintenance overhead.
- **Siloed Freestyle Workflows:** Hard to share or generalize methodologies across teams.
- **No Version Control for Jobs:** Configurations live only on Jenkins host → risk of loss, no audit trail.
- **Poor Scalability for Complex Pipelines:** Designed for simple CI/CD; struggles with large distributed workflows.



# Life Before Airflow - Jenkins

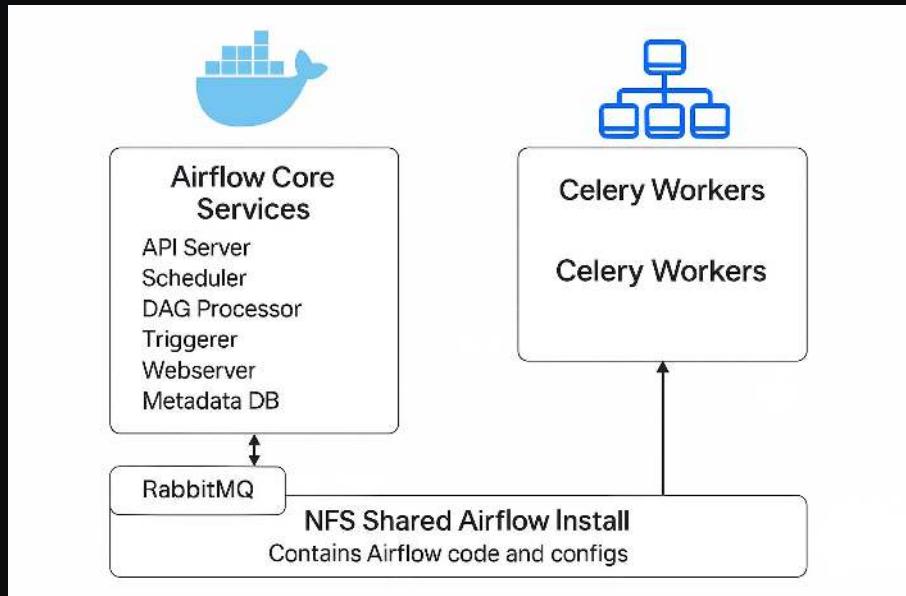
The Good:



# Airflow @ QCOM

## Our Airflow Instance:

- Core services run in Docker container
- Airflow install/code/configs shared via NFS
- Celery workers run as LSF jobs



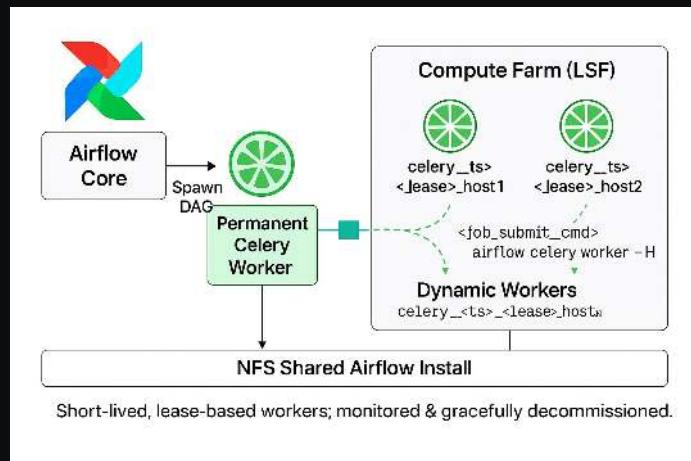
# Dynamic Celery Workers

## Spawning Celery Workers:

- Single always-on worker leases temporary workers on multi-day lease
- Temp workers created/shutdown via DAG:

```
job --cpus 8 --mem 64GB -- airflow celery worker -H
celery_<timestamp>_<lease_hrs>_<hostname>
```

- We contributed this for remote celery worker management!



>Add CLI command to remove all queues from Celery worker ✓ area:providers provider:celery

#56195 by dheerajturaga was merged yesterday • Approved

Enhance Celery CLI with Worker and Queue Management Features ✓ area:providers provider:celery

#51257 by dheerajturaga was merged on Jun 1 • Approved

Add worker\_umask to celery provider.yaml ✓ area:providers provider:celery

#51218 by dheerajturaga was merged on Jun 1 • Approved

```
Usage: airflow celery [-h] COMMAND ...

Start celery components. Works only when using CeleryExecutor. For more information, see https://airflow.apache.org/docs/apache-airflow/stable/executor/celery.html

Positional Arguments:
  COMMAND
    add-queue      Subscribe Celery worker to specified queues
    flower         Start a Celery Flower
    list-workers   List active celery workers
    remove-queue   Unsubscribe Celery worker from specified queues
    shutdown-all-workers
    shutdown-worker Request graceful shutdown of all active celery workers
    stop-worker    Request graceful shutdown of celery workers
    stop           Stop the Celery worker gracefully
    worker         Start a Celery worker node
```

# How do we scale up from here???

Challenges:

- Data Center is FULL -> Need more compute
- Compute availability is spread across many DCs
- Need to access Emulation resources
- Need to run on hosts hooked up to custom silicon

Can a single Airflow instance handle  
EVERYTHING?



# Multi Executors: Celery + Edge3

**Celery** → Local jobs in the same data center

**Edge** → Remote jobs & special hardware, anywhere!

## Why it rocks:

- ✓ One Airflow to rule them all
- ✓ Global compute orchestration
- ✓ No more DC handcuffs

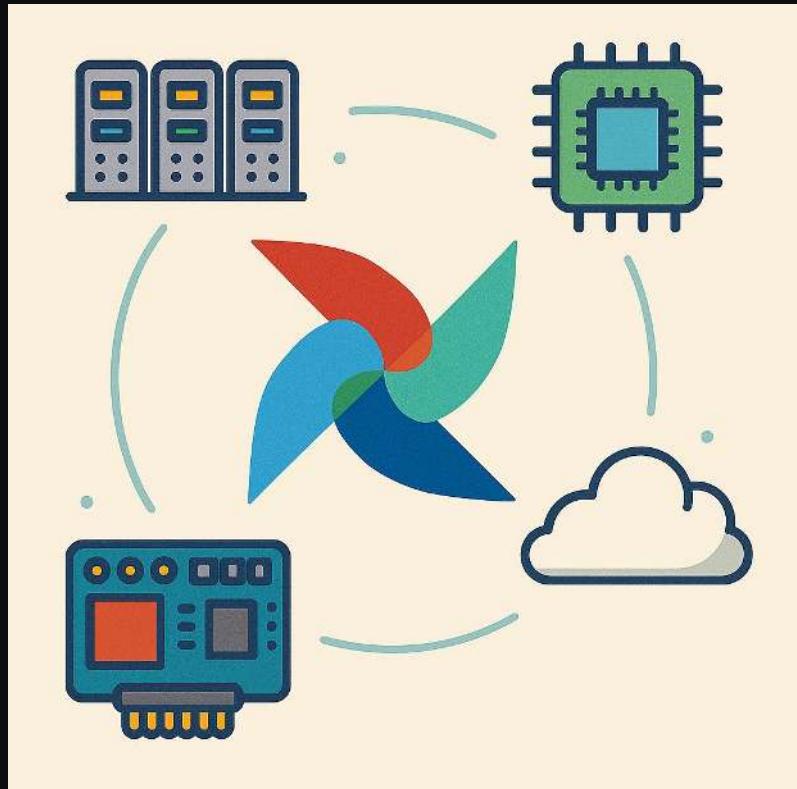
## How we do it:

- Permanent Edge Workers in each DC (always-on gateways)
- Dynamic Edge Workers spin up on demand
- Smart Routing: Celery for local, Edge for remote
- Load Balancing across DCs

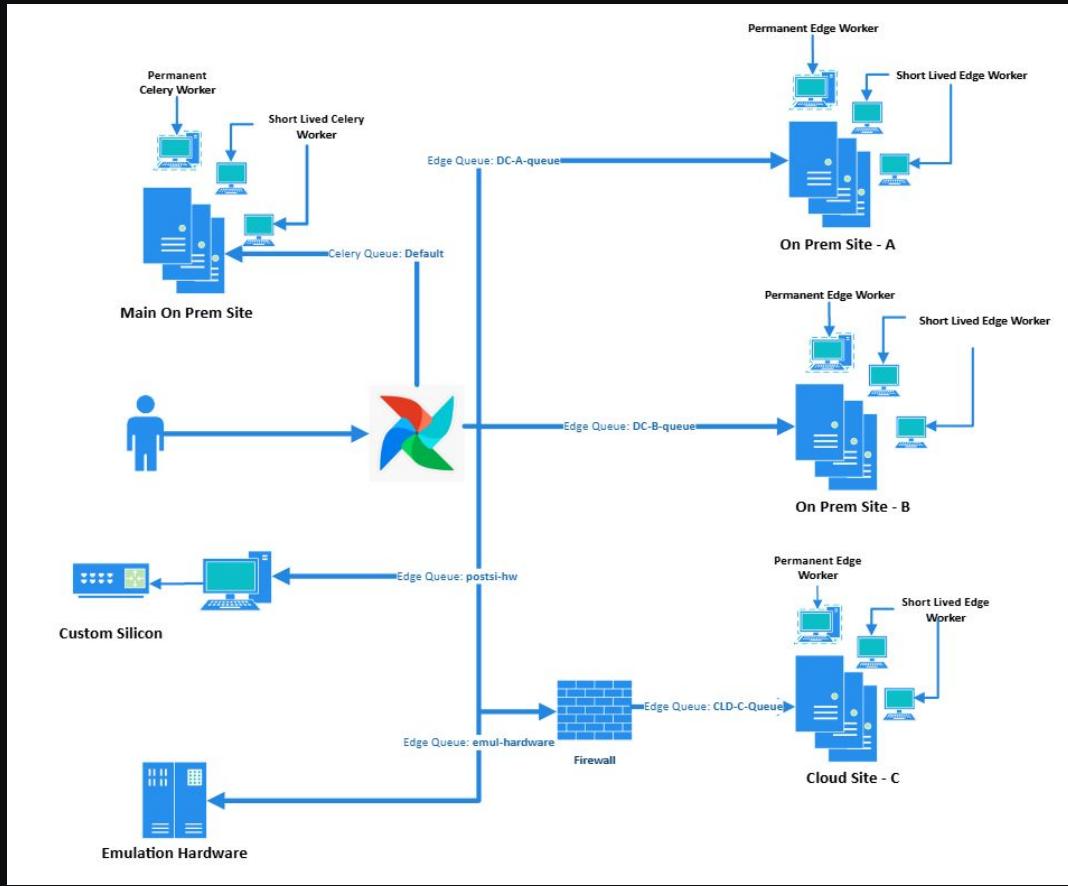
## Use Cases:

Pre-Silicon: Build on HPC, run on emulators

Post-Silicon: Validate on custom boards

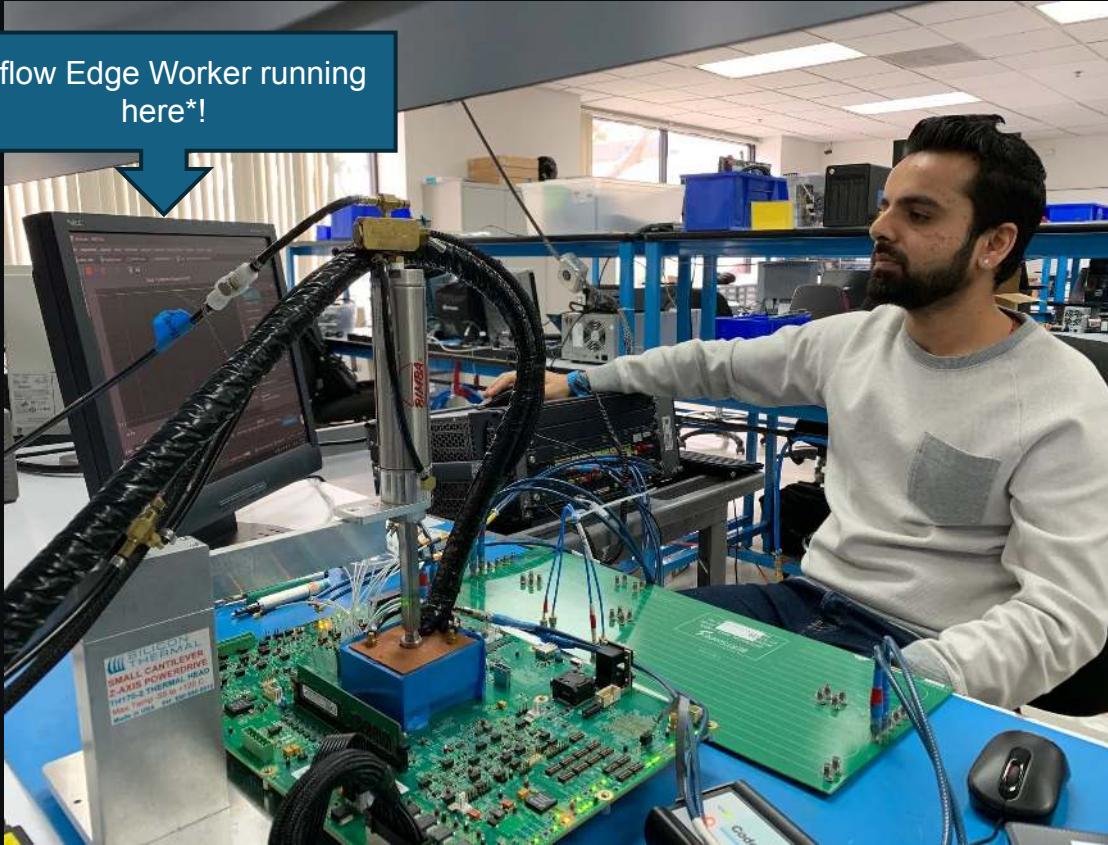


# Airflow @ QCOM – Multi-DC Architecture



# Airflow @ QCOM – Custom Silicon

Airflow Edge Worker running  
here\*!



\*This is a representative image only.

# Edge/Celery Contributions

- >Add revoke\_task implementation to EdgeExecutor for task queued timeout support ✓ area:providers provider:edge  
#56240 by dheerajturaga was merged 3 days ago • Approved
- Add CLI command to remove all queues from Celery worker ✓ area:providers provider:celery  
#56195 by dheerajturaga was merged 3 days ago • Approved
- Add yellow hover background to enter maintenance icon ✓ area:providers provider:edge  
#55631 by dheerajturaga was merged 3 weeks ago • Approved
- Add shutdown-all-workers command to Edge CLI ✓ area:providers provider:edge  
#55626 by dheerajturaga was merged 3 weeks ago • Approved
- Add queue management UI buttons for Edge workers ✓ area:providers provider:edge  
#55625 by dheerajturaga was merged 3 weeks ago • Approved
- feat: Add delete button for offline edge workers ✓ area:providers provider:edge  
#55529 by dheerajturaga was merged 3 weeks ago • Approved
- feat: Add shutdown button for edge workers with confirmation dialog ✓ area:providers provider:edge  
#55513 by dheerajturaga was merged 3 weeks ago • Approved
- Add confirmation dialog for exit maintenance action in Edge Worker Page ✓ area:providers provider:edge  
#55400 by dheerajturaga was merged 3 weeks ago • Approved
- Add worker maintenance mode functionality to Edge3 provider UI ✓ area:providers provider:edge  
#55301 by dheerajturaga was merged last month • Approved
- Fix EdgeWorker multiprocessing pickle error on Windows ✓ area:providers provider:edge  
#55284 by dheerajturaga was merged last month • Approved
- Add queue and remove queue cli commands for EdgeExecutor ✓ area:providers provider:edge  
#53505 by dheerajturaga was merged on Jul 19 • Approved
- Edge list worker cli command to list active job metrics ✓ area:providers provider:edge  
#51720 by dheerajturaga was merged on Jun 14 • Approved

- Extend command column in the edge.job table to accomodate more chars ✓ area:providers provider:edge  
#51716 by dheerajturaga was merged on Jun 14 • Approved
- Enhance Celery CLI with Worker and Queue Management Features ✓ area:providers provider:celery  
#51257 by dheerajturaga was merged on Jun 1 • Approved
- Add worker\_umask to celery provider.yaml ✓ area:providers provider:celery  
#51218 by dheerajturaga was merged on Jun 1 • Approved
- Support For Edge Worker in Daemon Mode ✓ area:providers provider:edge  
#50425 by dheerajturaga was merged on May 11 • Approved
- Ability to request shutdown of a remote edge worker ✓ area:providers provider:edge  
#50278 by dheerajturaga was merged on May 6 • Approved
- Extend Edge Worker CLI commands operate on remote edge workers ✓ area:providers provider:edge  
#49915 by dheerajturaga was merged on May 2 • Approved
- Minor doc fix in edge\_executor ✓ area:providers kind:documentation provider:edge  
#49755 by dheerajturaga was merged on Apr 25 • Approved

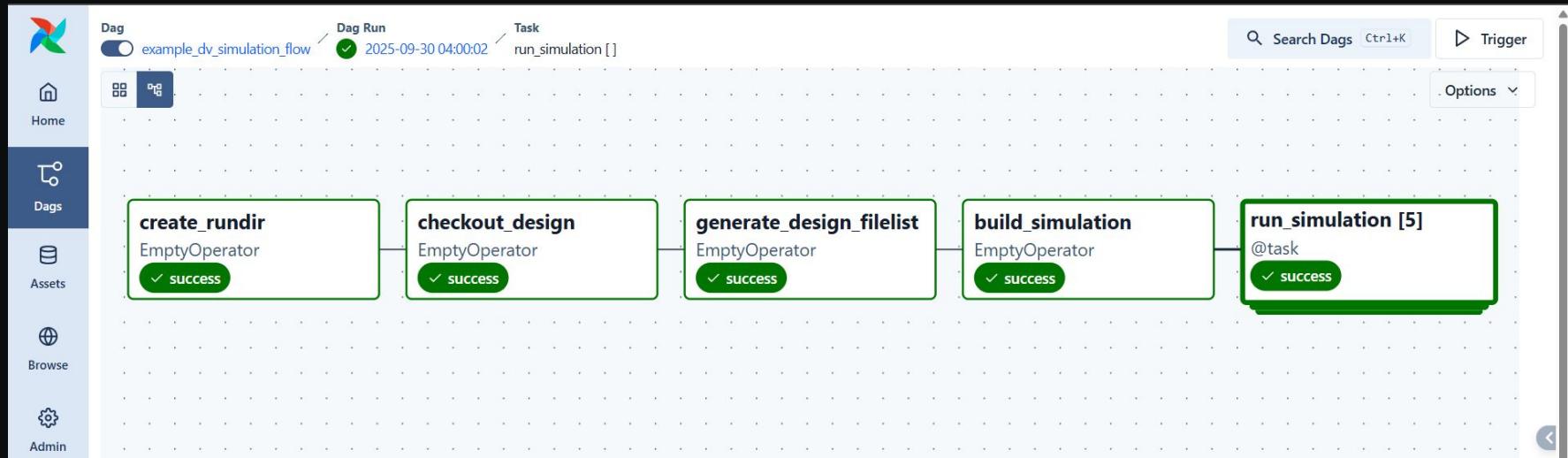
View all our contributions here:



# Methodologies

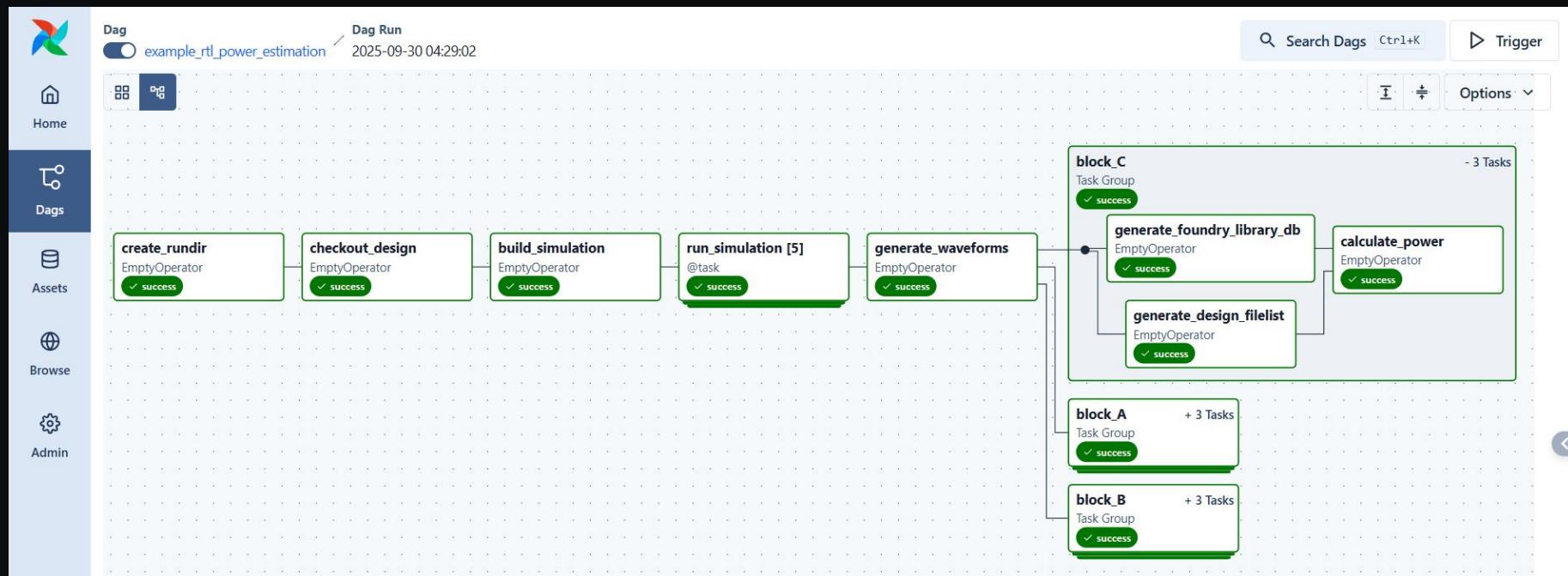
# Chip Design Workflow - Examples (DV)

*Design Verification (DV) is debugging hardware before it's built — to catch costly mistakes early.*



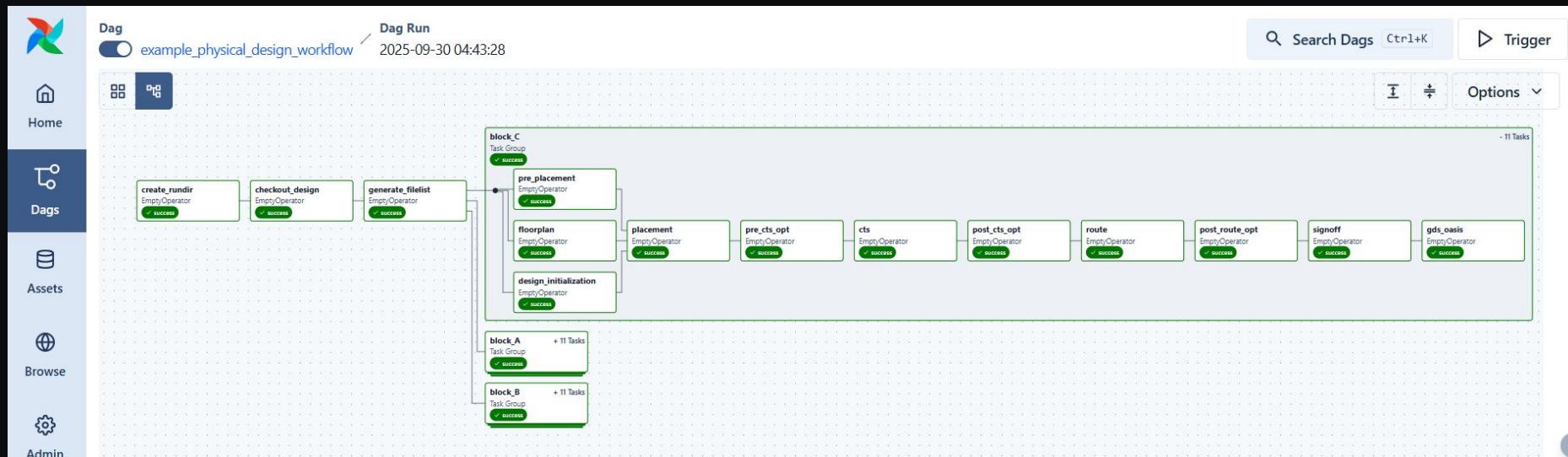
# Chip Design Workflow – Examples (Power Estimation)

*RTL power estimation is simulating the chip's power efficiency using its design plans before fabrication.*



# Chip Design Workflow - Examples (Physical Design)

*Physical Design is the process of converting a chip's logical design into a layout that can be manufactured on silicon..*



# Pytest Plugin for Airflow

Unit testing an Airflow codebase:

- 'airflow dags list-import errors' is not enough!
- No official Pytest support 😞
- Custom Pytest plugin -> [apache-airflow-devel-common](#) -> [pytest\\_plugin.py source code](#)

```
@pytest.mark.db_test
@pytest.mark.smoke
@ pytest.mark.execution_timeout(3000)
def test_integrity(dags_path):
    """
    Replaces `airflow dags list-import-errors`
    """
    dag_bag = DagBag(
        dag_folder=dags_path, include_examples=False, read_dags_from_db=False
    )
    assert not dag_bag.import_errors, "DAG import errors are present :("
    slowpokes = [
        s for s in dag_bag.dagbag_stats if s.duration > duration(seconds=10)
    ]
    assert not slowpokes, "Slowpokes are present :(
```

# Pytest Plugin for Airflow

```
@pytest.mark.db_test
@pytest.mark.execution_timeout(300)
def test_a_dag(dag_maker, session, default_args):
    """
    Test a DAG!
    """
    DAG_CFG = {
        # something relevant
    }
    with dag_maker(
            dag_id="test_a_dag",
            bundle_name="dags-folder",
            default_args=default_args,
            params=BASE_PARAMS(DAG_CFG),
            session=session,
    ):
        ...
        dr = dag_maker.create_dagrun(run_type=DagRunType.MANUAL)
        for ti in dr.get_task_instances(
                session=session
        ): # Mimic the specific order the scheduling would run the tests.
            ti.run(session=session)
            assert ti.state == TaskInstanceState.SUCCESS, f"{ti} failed!"
        dr.update_state(session=session)
        assert dr.state == DagRunState.SUCCESS, f"{dr} failed!"
    # more asserts
```

```
class TestCustomOperator:

    @pytest.mark.db_test
    def test_template_fields(self, create_task_instance_of_operator, session):
        ti = create_task_instance_of_operator(
            CustomOperator,
            # Templatized fields
            bash_command='echo "{{ dag_run.dag_id }}"',
            env={"FOO": "{{ ds }}"},
            cwd="{{ task.dag.folder }}",
            # Other parameters
            dag_id="test_template_fields_dag",
            task_id="test_template_fields_task",
            session=session,
        )
        context = ti.get_template_context(session=session)
        ti.render_templates(context=context)
        task: CustomOperator = ti.task
        assert task.bash_command.endswith('echo "test_template_fields_dag"')
        assert task.cwd == Path(__file__).absolute().parent.as_posix()
```

# Custom Shell Operators

## Shell Operators:

- Source global shell .rc file
- Modify SubprocessHook to pass entire stdout/output log path
- Custom pre/post-processing via pre\_execute/post\_execute
- Enables smart retries + scraping XComs from shell output

```
class SubprocessHook(BaseHook): <- class CustomHook(SubprocessHook)
    ...
    def run_command(
        self,
        command: list[str],
        env: dict[str, str] | None = None,
        output_encoding: str = "utf-8",
        cwd: str | None = None, <- Add optional output_log
    ) -> SubprocessResult:
        ...

class BashOperator(BaseOperator): <- class CustomOperator(BashOperator)
    ... <- def execute() uses CustomHook
    def __init__(
        self,
        *,
        bash_command: str | ArgNotSet, <- Prepend env setup commands
        env: dict[str, str] | None = None,
        append_env: bool = False,
        output_encoding: str = "utf-8",
        skip_on_exit_code: int | Container[int]
        cwd: str | None = None,
        **kwargs,
    ) -> None:
```

# DAG Templates

## Generalizing Workflows:

- DAG Factories but very different!
- Use symlinks to DAG templates + YAML dictionary configuration
- Enforce methodology with a unit test!
- .airflowignore:
  - \*/templates/
  - \*/\*\_template.py

```
my_dag/
└── cfg.yml
└── dag.py  -> templates/dag_template.py
└── templates/
    └── dag_template.py
```

```
@pytest.mark.db_test
@pytest.mark.smoke
def test_symlink_methodology(dags_path):
    """
    Enforce my rules!
    """

    for p in dags_path.rglob("*"):
        if p.is_symlink():
            # check if broken link
            # check for adjacent cfg.yml
```

# Questions?

Dheeraj Turaga

turaga@qti.qualcomm.com



in

Nick Redd

redd@qti.qualcomm.com



in

Like what we do?  
**COME JOIN US!  
WE ARE HIRING**



Airflow

Infrastructure &  
Automation Engineer –  
Silicon Design

Apply Here!



Qualcomm  
Snapdragon