Skip Tasks to Make your Debugging Easy

Challenges, Solutions, Lessons Learned

Howie Wang, Software Engineer, Apple
Airflow Summit 2022

THIS IS NOT A CONTRIBUTION

Agenda

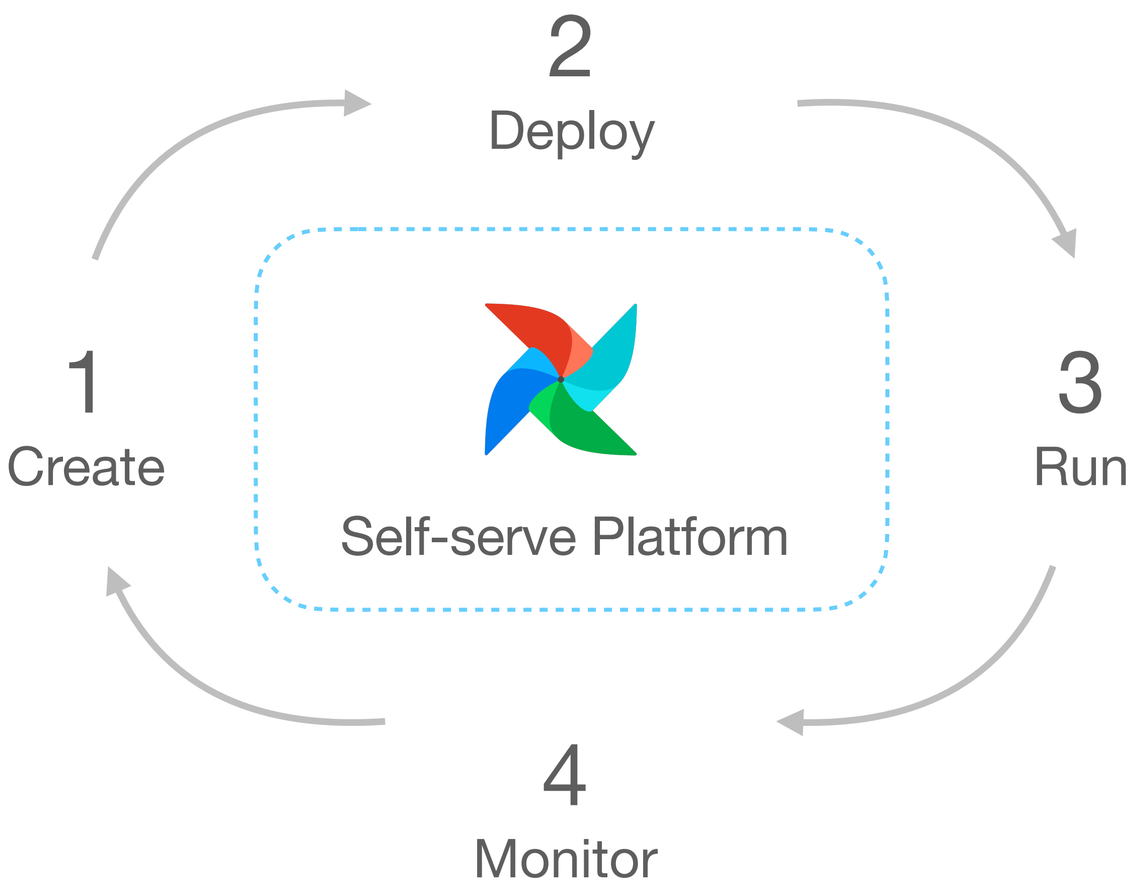
Use Case

Challenges

Solutions & Tradeoffs

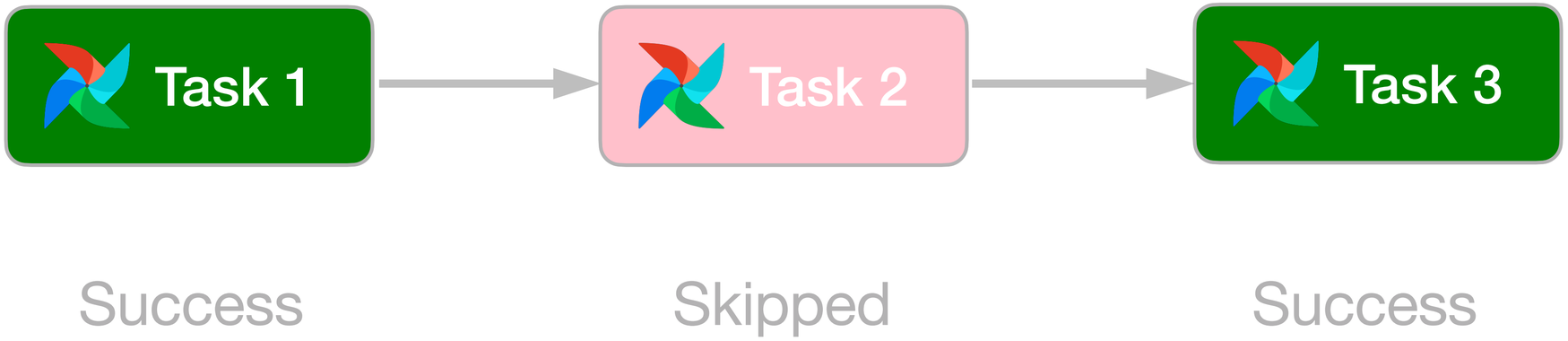
Lessons Learned

Self-serve Data Platform



Skip Task Use Case

Skip Task Use Case



Challenges

Must retain the dependencies between non-skip tasks

Must not force user to modify DAG structure

Must allow user to skip multiple tasks at once

Must minimize impact to Airflow Core

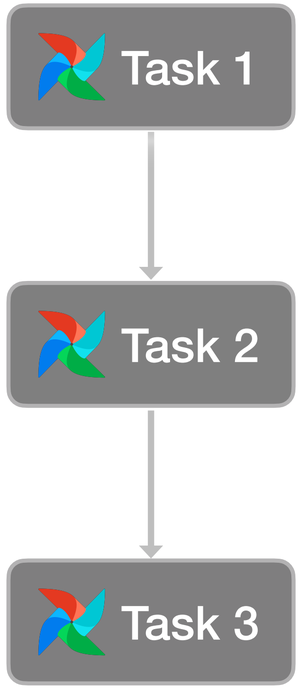
Option 1: Trigger DAG then mark SUCCESS

Pros

- Marking SUCCESS is supported via Airflow UI/API

Cons

- Does not retain dependencies between tasks
- Hassle when there are multiple tasks to skip
- Cannot mark task as SKIPPED



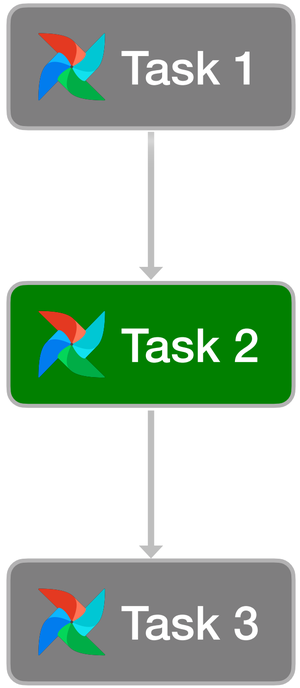
Option 1: Trigger DAG then mark SUCCESS

Pros

- Marking SUCCESS is supported via Airflow UI/API

Cons

- Does not retain dependencies between tasks
- Hassle when there are multiple tasks to skip
- Cannot mark task as SKIPPED



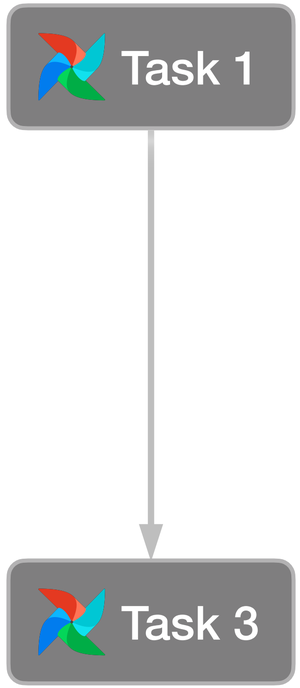
Option 2: Remove to-skip tasks from DAG

Pros

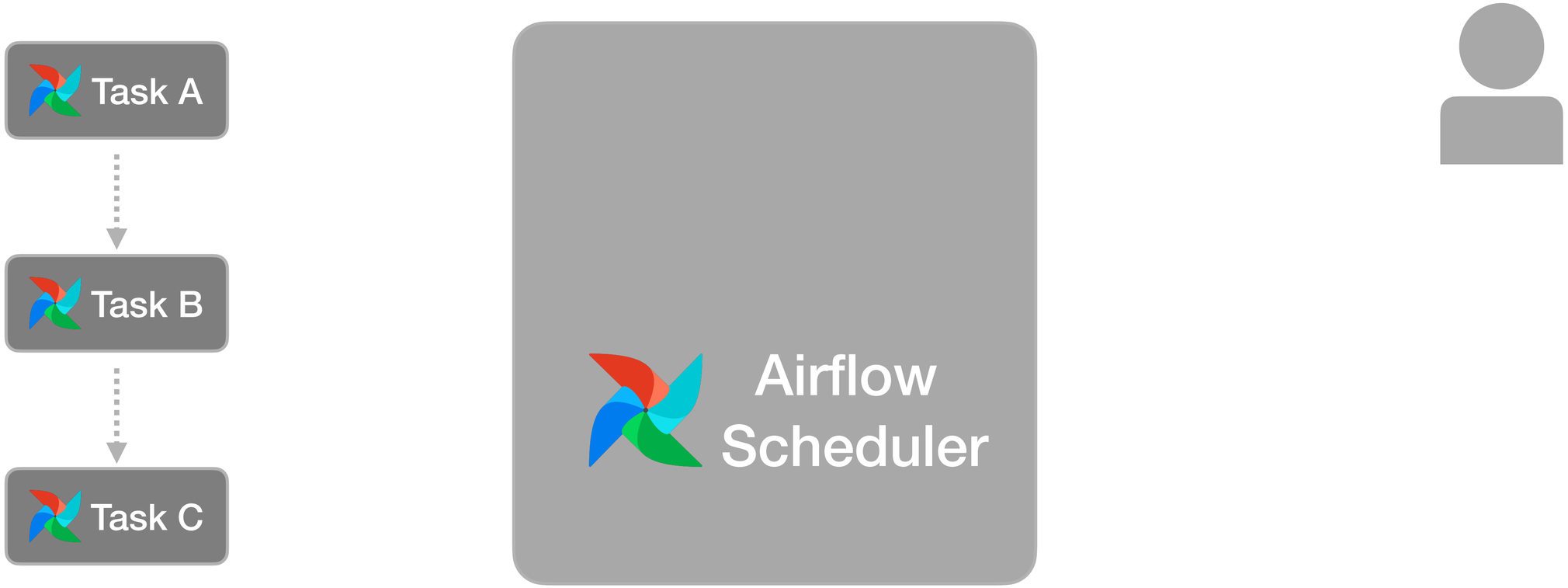
- CAN retain dependencies between tasks

Cons

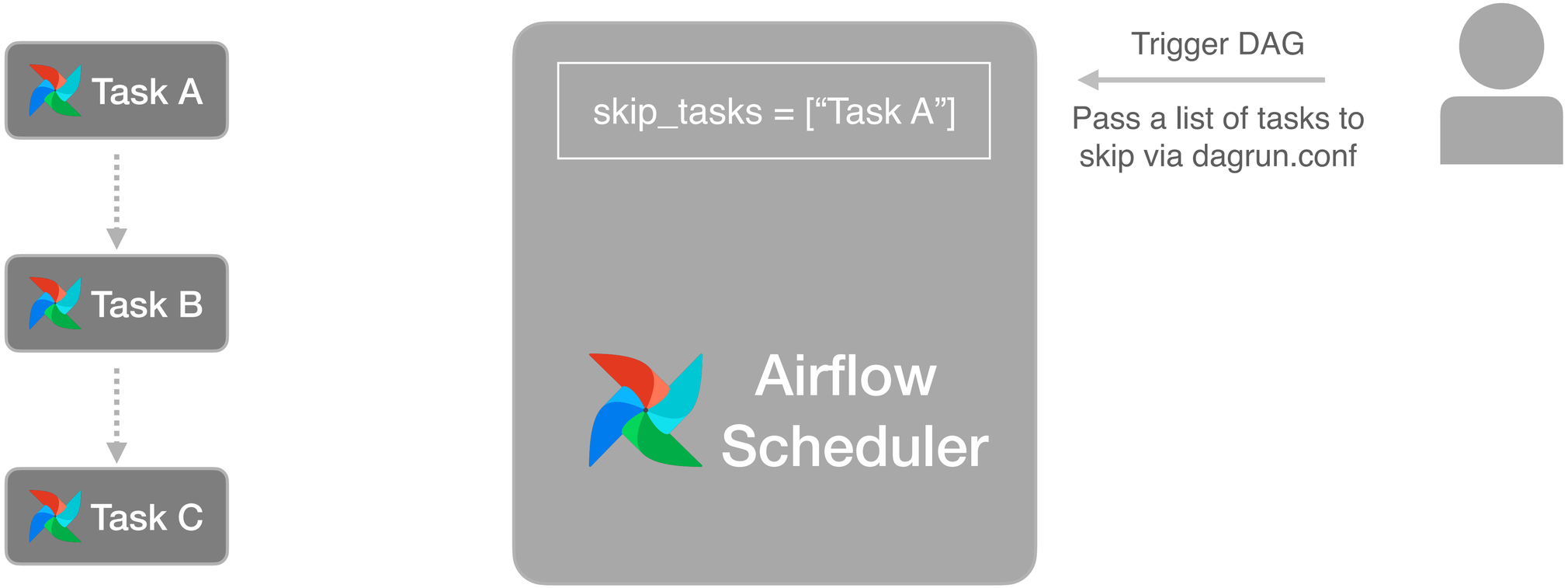
- Require user to modify DAG structure
- Tasks are removed instead of skipped



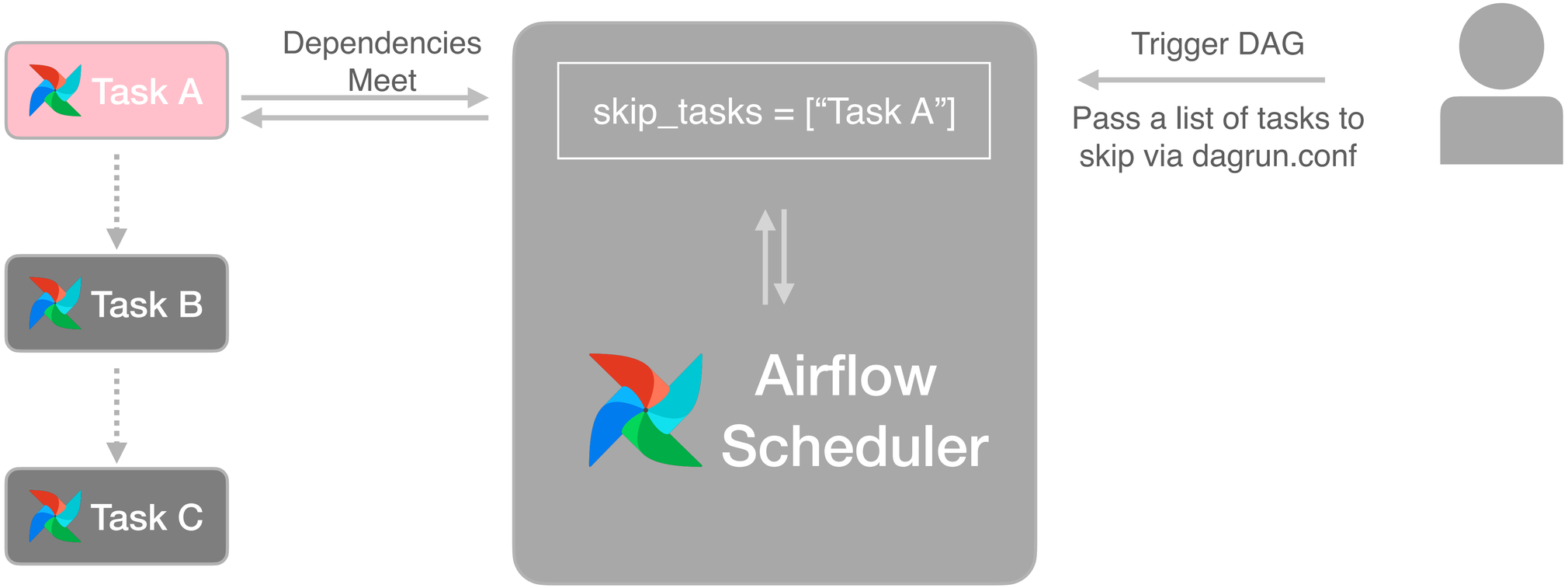
Option 3: Do not schedule tasks to skip



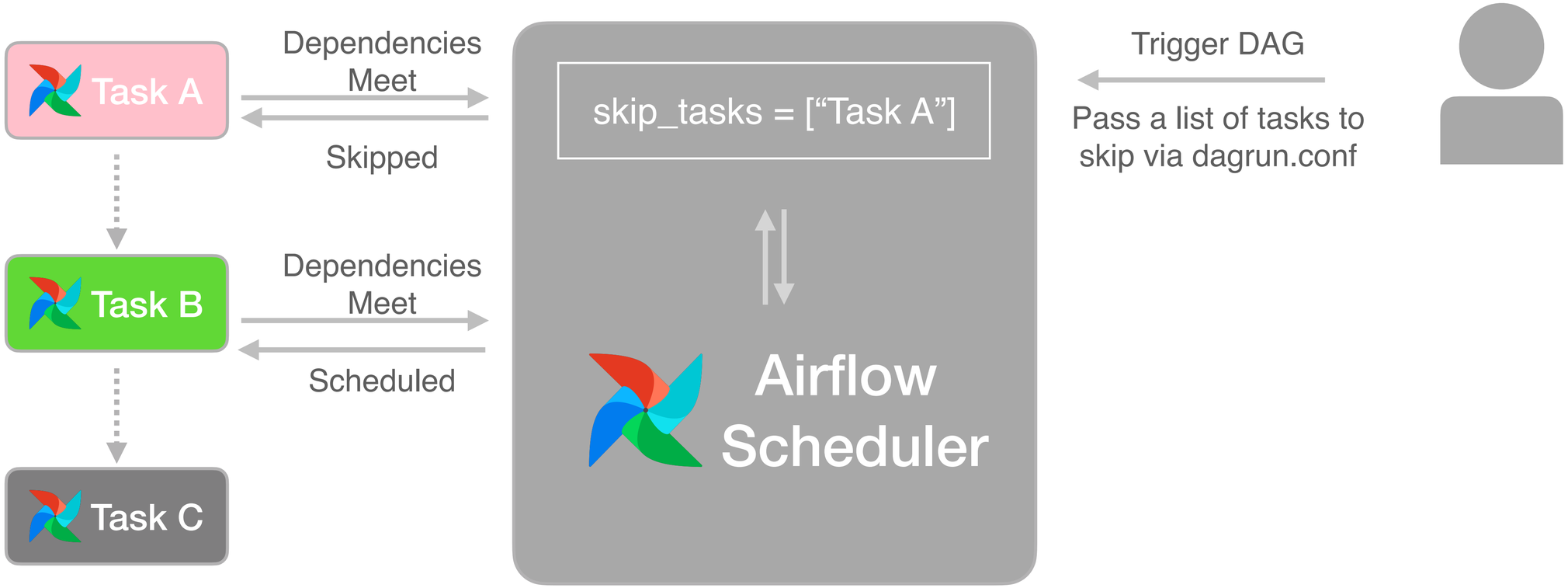
Option 3: Do not schedule tasks to skip



Option 3: Do not schedule tasks to skip



Option 3: Do not schedule tasks to skip



Option 3: Do not schedule tasks to skip

Pros

- CAN retain dependencies between tasks
- CAN easily specify multiple tasks to skip
- No need to modify or redeploy existing DAG

Cons

- Requires change in Airflow Core code (scheduler)
- Couples Scheduler with Dagrun.conf
- Unforeseen performance impact

Option 4: Skip Tasks with pre_execute

```
with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_success',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```

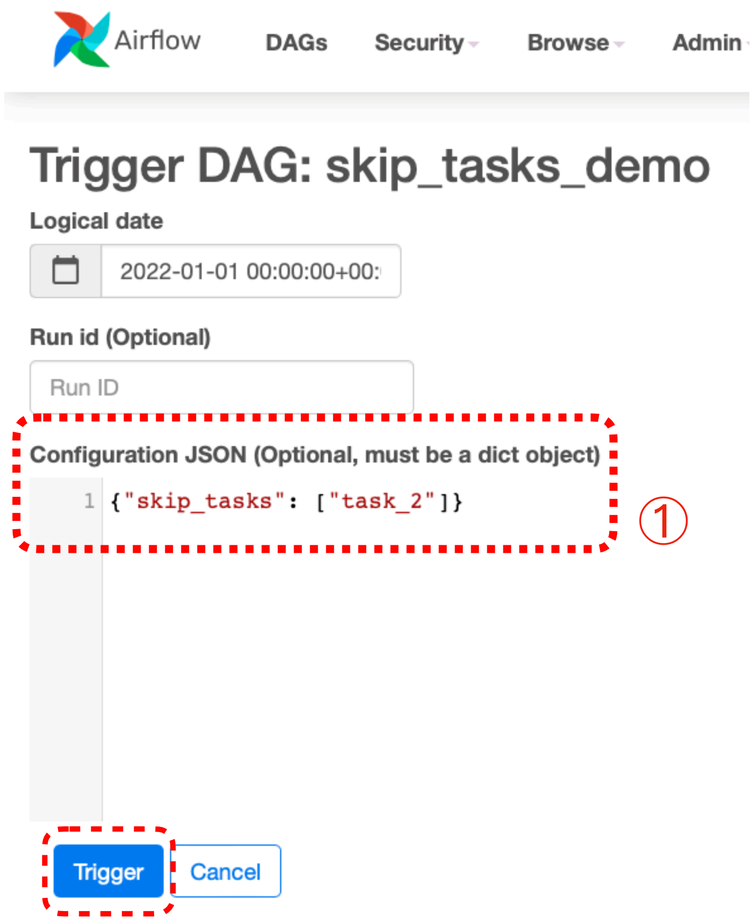
Option 4: Skip Tasks with pre_execute

```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        raise AirflowSkipException()
```

②

```python
with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_success',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```

③



①

Option 4: Skip Tasks with pre_execute

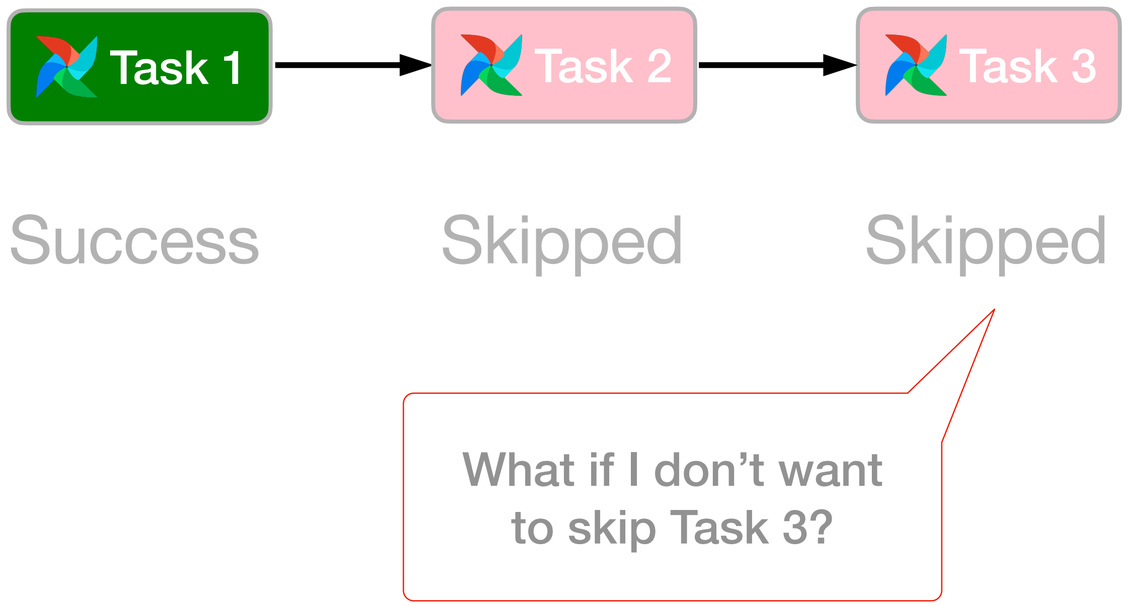
```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        raise AirflowSkipException()
```

```python
with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_success',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```

Option 4: Skip Tasks with pre_execute

```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        raise AirflowSkipException()
```

```python
with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_success',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```



Option 4: Skip Tasks with pre_execute

```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        raise AirflowSkipException()

with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_done',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```

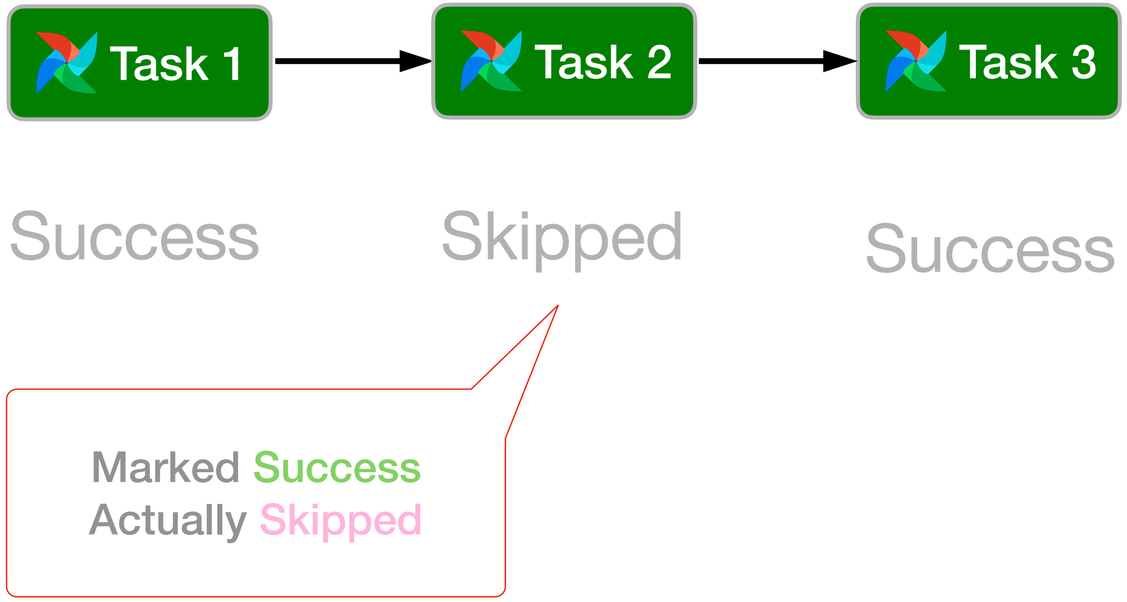
Option 4: Skip Tasks with pre_execute

```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        raise AirflowSkipException()

with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_done',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```

What if I still want to use all_success?

Option 4: Skip Tasks with pre_execute

```python
# callable for "pre_execute"
def skip_if_specified(context):
    task_id = context['task'].task_id
    conf = context['dag_run'].conf or {}
    skip_tasks = conf.get('skip_tasks', [])
    if task_id in skip_tasks:
        ti = context['dag_run'].get_task_instance(task_id)
        ti.set_state(State.SUCCESS)
        raise AirflowException()

with DAG(
    'skip_tasks_demo'
    default_args={
        'trigger_rule': 'all_success',
        'pre_execute': skip_if_specified
    },
    start_date=datatime(1976, 4, 1)
) as dag:
    t1 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_1"'
    )
    t2 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_2"'
    )
    t3 = BashOperator(
        task_id='task_1',
        bash_command='echo "this is task_3"'
    )
    t1 >> t2 >> t3
```



Option 4: Skip Tasks with pre_execute

Pros

- CAN retain dependencies between tasks
- CAN easily specify multiple tasks to skip
- Scalable (easy to manage)
- No impact to Airflow Scheduler
- Compatible with both Airflow 1 & 2

Cons

- With Kubernetes Executor, a pod will still launch for skipped task

Lesson Learned

Don’t add burden to Airflow Scheduler

Marking task as Skipped may cause side effect

Airflow’s devlist is a great resource