Traps and misconceptions of running reliable workloads

Bartosz Jankiewicz
Cloud Composer: Apache Airflow in Google Cloud

Bartosz Jankiewicz
Engineering Manager
Lesson 1

Can I always run tasks in Apache Airflow in reliable manner?
# Sources of failures

<table>
<thead>
<tr>
<th>External</th>
<th>Internal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originate from dependent services, latency, inconsistent data, network connectivity</td>
<td>Intrinsic to our setup, our code running in Apache Airflow, our actions</td>
</tr>
</tbody>
</table>
Lesson 2

Apache Airflow is a distributed system but you need to understand how to use the redundancy.
Some of single points of failures

- Network
- Metadata database
- Celery worker
What about redundancy?
Redundancy can help only when a component is stateless or its state can be recovered.

Typically executor process failure leads to task instance failure. To alleviate this, tasks should be configured with retries.

Running tasks in deferrable mode makes the tasks stateless from Airflow perspective.
Lesson 3

How can you improve availability of metadata database?
Database failures happen

- Too many tasks
- Badly written DAGs
- Sensors
- Hardware failures
Some countermeasures

- Scale up
- Defer
- Limit parallelism
- Change schedule

- Increase poll interval
- Defer
- Change architecture

- Use Variables with care
- Jinja template can reduce db calls

- Highly available database
Watch out!

Database scaling is not a free lunch!
More DB resources → Bigger load → More data

- Longer maintenance
- Cost
More effective solutions

- Schedule tasks more evenly
- Spread load across more Airflow instances
- Use Variables with care + Jinja templates
- Highly available database
Lesson 4

How can I make sure my Python code runs uninterrupted?
What can disrupt your task?

- Other scripts were running on the same host
- Maintenance operations on VMs
- Database being overwhelmed
- Network latency
- Dependant service failure
# Quick recap of Airflow executors

## Local Executors
- SequentialExecutor
- LocalExecutor

## Remote Executors
- CeleryExecutor
- KubernetesExecutor
- CeleryKubernetesExecutor
Quick summary of actions

- Isolate tasks e.g. with K8s executor
- Don't overload metadata database
- Run tasks in deferrable mode
- Plan your maintenance windows
What if everything fails?

Have a plan 😎
Questions?

Optionally share some contact info like email, blog or social media handles.