Hochschule Karlsruhe
University of
Applied Sciences

Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Vorlesung 6 – ETL – Airflow

Foto: Quirin Leppert

Themenübersicht

Data Integration

- Data Formats (csv, XML, json)
- **Extract, Transform, Load**
- Object Relation Mapper (ORM)
- **Staging**

Data Processing

- Relationale Datenbanken
- nicht-relationale Datenbanken
- Resource Description Framework (RDF)
- Ontologien
- Data Warehouse

Data Modelling

- Serialisierung
- OPC UA
- MQTT
- Pub/Sub
- **Data pipelines**
  - **Apache Airflow** (Fortsetzung)
  - gRPC

Web-Service Architektur

- Front-End
- Backend for Frontend (BFF)
- Micro Services
- Docker Container

Security

- Security ist wichtig in allen Phasen der Softwareentwicklung und Datenbereitstellung.

Übung

- Erstellung eines Daten-Modells einer prozesstechnischen Anlage
- Statische Daten
- Dynamische Daten
- Auswertung der Daten

Eigenschaften einer Production Pipeline

- Staging of data
- Building idempotent data pipelines
- Building atomic data pipelines

Quelle: Data Engineering with python, Paul Crickard, Packt, Oktober 2020

Eigenschaften einer Production Pipeline

Staging of data

- Intermediate database where all the data integration and transformation processes are run prior to the loading of the data into the data warehouse.
- Snapshots von Datenbanken zu bestimmten Zeitpunkten
- Nachvollziehbarkeit von Auswertungen in der Vergangenheit
- Testing und Debugging

Quelle: Data Engineering with python, Paul Crickard, Packt, Oktober 2020; Data Warehouse Systems, Vaisman, Zimányi, Springer, 2014

Eigenschaften einer Production Pipeline

Building idempotent data pipelines

+ **idempotent**
- Data pipelines, die mehrfach ausgeführt werden können, ohne das Ergebnis zu verändern.
- Mathematische Definition mit dem Objekt *a* und einer Verknüpfung •:
  $a \bullet a = a$
+ Vorteile
- Bei Abbruch der Ausführung einer Data pipeline kann die Ausführung einfach erneut angestoßen werden.

Realisierung

+ Guard Clauses
- Abfrage, ob eine Datenbank/Tabelle/Zeile bereits vorhanden ist, bevor diese erzeugt/geschrieben wird.
+ Erzeugung einer neuen Partition/eines neuen Index bei jedem Durchlauf der Pipeline, z.B. durch Nutzung eines Zeitstempels als Suffix

Quelle: Data Engineering with python, Paul Crickard, Packt, Oktober 2020; https://www.red-gate.com/hub/product-learning/flyway/creating-idempotent-ddl-scripts-for-database-migrations

Eigenschaften einer Production Pipeline

Building atomic data pipelines

+ Nutzung von Transaktionen
+ Schlägt im Verlauf der Durchführung einer Transaktion ein Befehl fehl, so können alle Befehle der Transaktion zurückgenommen werden (roll back).

+ Ausführung einer Transaktion mit dem MSSQL
- Explizite Nutzung einer Transaktion

```
BEGIN TRANSACTION;
DELETE FROM HumanResources.JobCandidate
WHERE JobCandidateID = 13;
COMMIT;
```

- Ausführen eines Rollback
  Zwei Zeilen werden in die Datenbank eingefügt. Durch Ausführung des Befehls `ROLLBACK` wird die Einfügung rückgängig gemacht.

```
CREATE TABLE ValueTable (id INT);
BEGIN TRANSACTION;
        INSERT INTO ValueTable VALUES(1);
        INSERT INTO ValueTable VALUES(2);
ROLLBACK;
```

Quelle: Data Engineering with python, Paul Crickard, Packt, Oktober 2020

Installation von ubuntu Linux als VM unter Windows

Installation von VirtualBox

- Virtual Box ist ein Player für virtuelle Maschinen
- https://www.chip.de/downloads/VirtualBox_23814448.html
- Virtual Box installieren und starten.

VM herunterladen

- Ordner UbuntuDesktop von https://cloud.pathconnect.de/s/2fKgba54kJBgQ9n herunterladen.
- Entpacken der zip-Datei

VM zu VirtualBox hinzufügen und VM starten

Quelle: Anleitung und Bereitstellung der VM durch Moritz Busch

- Anmelden mit user `student` und passwort `Dataengeneering21`
- **In der VM ist airflow bereits installiert.**
- Terminal öffnen und mit „airflow standalone“ die Anwendung starten
- Danach im Browser „localhost:8080“ aufrufen
- Benutzername admin, Passwort wird im Terminal ausgegeben

- **Alternativ** kann die VM auch selbst erstellt werden, siehe Anleitung von Moritz Busch https://cloud.pathconnect.de/s/2fKgba54kJBgQ9n

Einführung in Linux (1/4)

Administrator / root

+ `root` ist unter Linux das Account mit Administratorrechten
+ In der Praxis wird das root-Account nicht genutzt.
  Stattdessen wird von einem normalen Benutzeraccount der Befehl `sudo` genutzt.
  Dies entspricht einer Administrator-shell unter Windows.
+ Zur Bearbeitung von Nutzern, die berechtigt sind, `sudo` zu nutzen, wird der Befehl `sudo visudo` genutzt.

Editoren unter Linux

+ Verbreitet sind
- vi
- vim
- nano
- emacs

Editor vi

+ weitere Informationen siehe https://de.wikipedia.org/wiki/Vi
+ vi bietet keine graphische Oberfläche, entwickelt 1976
+ vi ist praktisch auf jedem Unix- und Linux-System installiert.
+ Bei der Installation der `git-bash` unter Windows wird vi (oder vim) ebenfalls installiert.
+ Die wichtigsten Befehle
- `i`
  Wechseln in den Modus zum Einfügen von Zeichen
- `Esc`-Taste
  Wechsel in den Modus für die Befehlseingabe
- `:w` (Enter)
  Speichern der Datei
- `:q` (Enter)
  Verlassen des Editors
- `:q!` (Enter)
  Verlassen des Editors ohne Speichern von Änderungen

Einführung in Linux (2/4)

Installation von Software (Distribution ubuntu)

- Die Installation von Software ist abhängig von der Linux-Distribution
- `sudo apt-get install <package name>`

oder

- Nutzung der graphischen Benutzeroberfläche `Synaptic Package Manager`
- Für komplexere Installationen (z.B. airflow) dedizierte Anleitung beachten.

Einführung in Linux (3/4)

Datei-Handling

+ Nutzung einer graphischen Oberfläche

oder

+ Befehle in der `bash` (bourne-again shell)
+ Die folgenden Befehle können auch unter Windows in der git bash ausgeführt werden
+ `df`
  Auflistung eingebundener Laufwerke
+ `df -h`
  Besser lesbare Ausgabe
+ `ls`
  Auflistung von Dateien/Verzeichnisse
+ `ls --help`
  Hilfe für ls
+ `ls -ahl`
  Erweiterte Informationen
+ `ls /`
  Auflisten der Dateien im Pfad / (oberster Pfad)
+ `ls ~`
  Auflisten der Dateien im Home-Verzeichnis
+ `cd`
  Wechsel des Verzeichnisses (standardmäßig ins Home-Verzeichnis)
+ `cd ~`
  Wechsel in das Home-Verzeichnis
+ `cp test.txt ~/tmp/`
  Kopieren der Datei test.txt in das Verzeichnis ~/tmp

Einführung in Linux (4/4)

Datei-Handling (2)

+ `ssh`
  Login auf einem Remote-Rechner über Secure-Shell
+ `scp`
  Sicheres, verschlüsseltes Kopieren von Dateien von einem Rechner auf einen anderen Rechner
+ `find . -name "test*"`
  Suche aller Dateien im aktuellen Verzeichnis und allen Unterverzeichnis, deren Name mit test beginnt.
+ `find . -name "test*" | xargs grep hallo`
  Suche nach dem String „hallo“ in allen Dateien, deren Dateiname mit test beginnt.

Data pipelines mit Apache Airflow
Installation unter Linux

Windows

- \+ lokale Installation nicht möglich
- \+ Docker-Installation fehlgeschlagen

Linux (ubuntu 21.04)

- \+ lokale Installation siehe https://airflow.apache.org/docs/apache-airflow/stable/start/local.html

```
# Airflow needs a home. `~/airflow` is the default, but you can put it # somewhere else if you prefer (optional)
export AIRFLOW_HOME=~/airflow

# Install Airflow using the constraints file
AIRFLOW_VERSION=2.2.0 PYTHON_VERSION="$(python --version | cut -d " " -f 2 | cut -d "." -f 1-2)"
# For example: 3.9
CONSTRAINT_URL=https://raw.githubusercontent.com/apache/airflow/constraints-${AIRFLOW_VERSION}/constraints-${PYTHON_VERSION}.txt
# For example: https://raw.githubusercontent.com/apache/airflow/constraints-2.2.0/constraints-3.9.txt
pip install "apache-airflow==${AIRFLOW_VERSION}" --constraint "${CONSTRAINT_URL}"

# The Standalone command will initialise the database, make a user,
# and start all components for you.
airflow standalone

# Visit localhost:8080 in the browser and use the admin account details
# shown on the terminal to login.
# Enable the example_bash_operator dag in the home page
```

Quelle: https://airflow.apache.org/docs/apache-airflow/stable/start/local.html

Installation der Datenbank postgresql

Linux (ubuntu 21.04)

+ siehe https://www.postgresql.org/download/linux/ubuntu

```
# install posgresql, Version 12 wird mit ubuntu mitgeliefert
sudo apt-get install posgresql-12

# start des postgresql-services
sudo systemctl start postgresql@12-main

# Verbindung zum postgresql-Server aufbauen
sudo su - postgres

# start der postgres shell
Psql

# Anlegen eines User-accounts
CREATE user userMeinName WITH PASSWORD 'meinPasswort';

# Anlegen einer Datenbank
CREATE DATABASE HKAdb1;
GRANT ALL PRIVILEGES ON DATABASE HKAdb1 to userMeinName;

# Test, ob der postgresql-Server reagiert
sudo pg_isready
```

Quelle: https://www.postgresql.org/download/linux/ubuntu

Installation der Administrationsoberfläche für postgresql pgAdmin 4

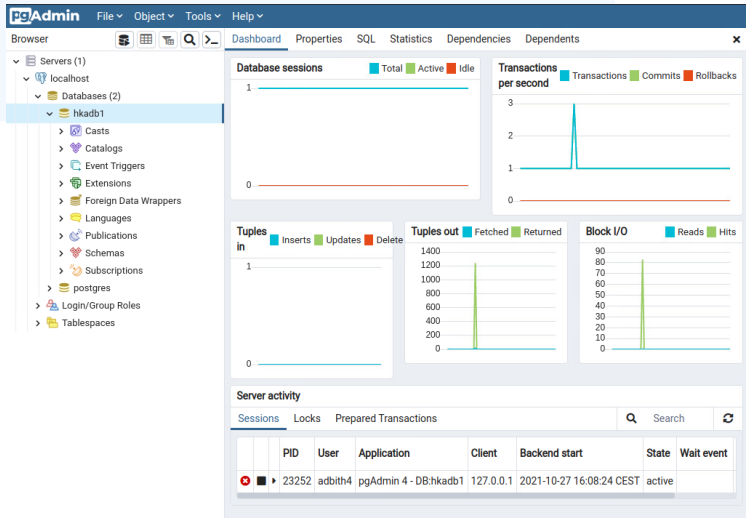
Linux (ubuntu 21.04)

+ siehe https://www.pgadmin.org/download/pgadmin-4-apt/

```
# Setup the repository
# Install the public key for the repository (if not done previously):
sudo curl https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo apt-key add

# Create the repository configuration file:
sudo sh -c 'echo "deb https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" >
 /etc/apt/sources.list.d/pgadmin4.list && apt update'

# Install pgAdmin
# Install for desktop mode only:
sudo apt install pgadmin4-desktop
```



Quelle: https://www.pgadmin.org/download/pgadmin-4-apt/

Installation des mssql-Servers als docker image

Linux (ubuntu 21.04)

+ Siehe https://docs.microsoft.com/de-de/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15&pivots=cs1-bash

```
# Herunterladen des docker images
sudo docker pull mcr.microsoft.com/mssql/server:2019-latest

# Starten des docker containers und setzen des Passworts für den Administrator sa
sudo docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=YourStrong@Passw0rd" \
   -p 1433:1433 --name sql1 -h sql1 \
   -d mcr.microsoft.com/mssql/server:2019-latest

# Anzeige laufender docker-Container
sudo docker ps -a
# Der MSSQL ist auf Port 1433 ansprechbar.

# Testen der Verbindung und Anlegen einer Datenbank
sqlcmd -S localhost -U sa -P YourStrong@Passw0rd -q 'CREATE DATABASE HKAx'
```

Quelle: https://docs.microsoft.com/de-de/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15&pivots=cs1-bash

Installation des Treibers für den mssql-Server

Linux (ubuntu 21.04)

- Siehe https://docs.microsoft.com/de-de/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver15#ubuntu17

```
sudo su
curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add -

#Ubuntu 21.04
curl https://packages.microsoft.com/config/ubuntu/21.04/prod.list > /etc/apt/sources.list.d/mssql-release.list

exit
sudo apt-get update
sudo ACCEPT_EULA=Y apt-get install -y msodbcsql17
# optional: for bcp and sqlcmd
sudo ACCEPT_EULA=Y apt-get install -y mssql-tools
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

Quelle: https://docs.microsoft.com/de-de/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver15#ubuntu17

Einrichten der Datenbank numPets für den mssql-Server

Linux (ubuntu 21.04)

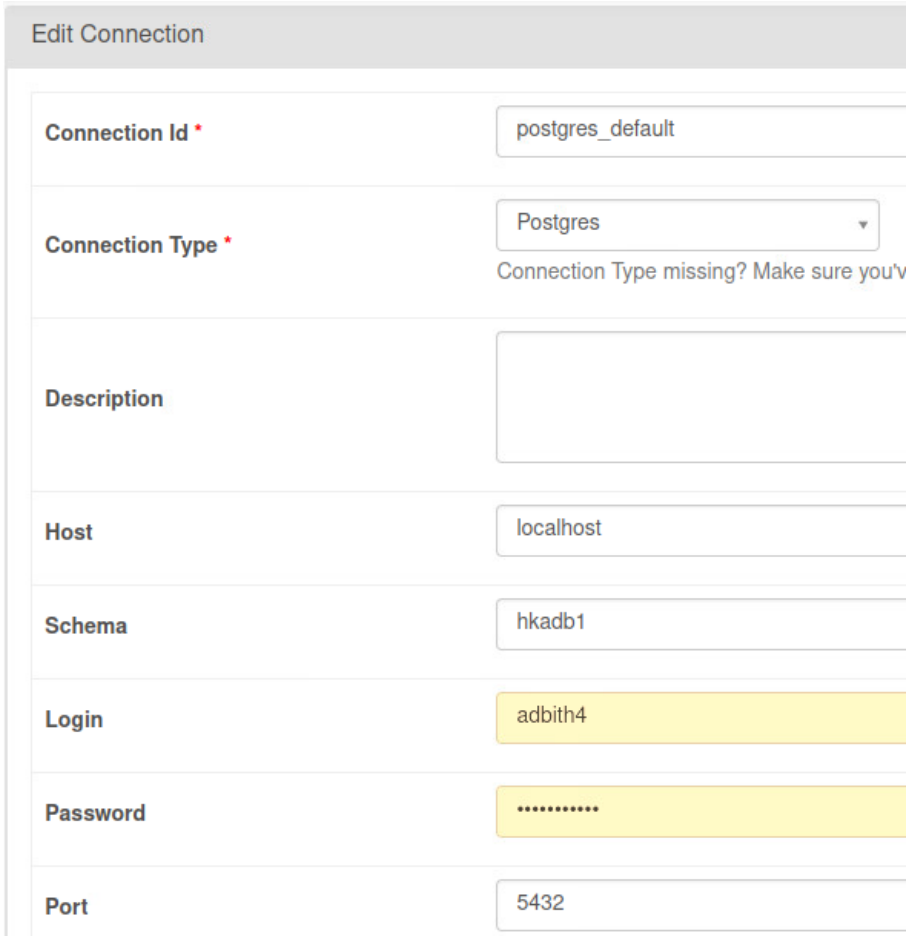
+ Verbindung mit dem mssql-Server aufnehmen
- `sqlcmd -S localhost -U sa -P YourStrong@Passw0rd`

```
# Erstellen der Datenbank
USE master;
GO
CREATE DATABASE numPets
GO

# Erstellen der Datenbanktabelle
CREATE TABLE num_pets(
 Id INT PRIMARY KEY IDENTITY (1,1),
 NumPets int
)
GO
```

Konfiguration der postgresql-Connection in airflow

- Zugriff auf localhost:8080
- Admin → Connections
- Editieren des postgres_default-Eintrags
- Schema: Datenbankname (hkadb1)



Data pipelines mit Apache Airflow

Debuggen eines DAGs

- Übung postgres_operator_dag.py
- Erstellen, Füllen und Abfrage einer Datenbanktabelle eines postgresql-Server mit `PostgresOperator`
- Am Ende der DAG-python-Datei folgendes ergänzen:

```
if __name__ == "__main__":
        from airflow.utils.state import State
        dag.clear(dag_run_state=State.NONE)
        dag.run()
```

- In der bash folgende Variable setzen:

```
export AIRFLOW__CORE__EXECUTOR=DebugExecutor
```

- Anschließend z.B. mit Visual Studio Code den DAG debuggen

Quelle: https://airflow.apache.org/docs/apache-airflow/stable/executor/debug.html

Data pipelines mit Apache Airflow

Nutzung eines performanten Database Backends

- Die Informationen über die Ausführung der Tasks wird einer Datenbank gespeichert.
- In der einfachsten Installation wird SQLite genutzt.
- Für eine performante Nutzung sollte eine Datenbank wie PostgresSQL oder MSSQL genutzt werden.

Quelle: https://airflow.apache.org/docs/apache-airflow/stable/howto/set-up-database.html

Data pipelines mit Apache Airflow

Parallele Ausführung der Tasks / Remote Execution

+ Bei der einfachsten Installation wird der `SequentialExecutor` genutzt. Tasks werden nacheinander ausgeführt, auch wenn der DAG eine parallele Ausführung zulassen würde.
+ Auf einem Rechner kann eine Parallelisierung mit Hilfe des `LocalExecutor` erreicht werden.

Für die Ausführung in einem Cluster stehen verschiedene Möglichkeiten zur Verfügung, u.a.

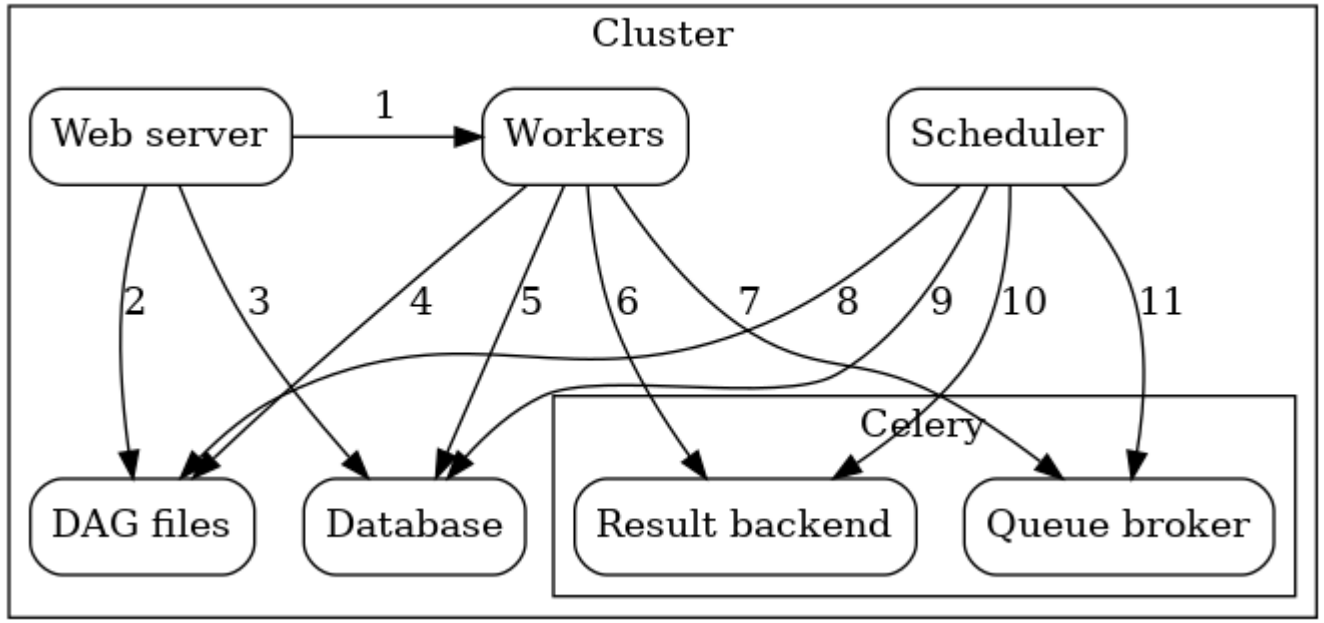
+ `Kubernetes Executor`
- Verteilung der Arbeit auf mehrere Docker Container
+ `Celery Executor`
- Verteilung der Tasks mit Hilfe von `Celery`. `Celery` nutzt AMQP (z.B. RabbitMQ) zur internen Kommunikation

Quelle: https://airflow.apache.org/docs/apache-airflow/stable/executor/index.html

Data pipelines mit Apache Airflow
Celery Executor

Architektur

- `Worker` führen den Task mit Hilfe von `Celery` aus.
- `Scheduler` fügt Tasks zur Queue hinzu.
- `Database` (backend) enthält Informationen über den Status der Tasks, DAGs, etc.
- Workers are statically configured and are running all the time.

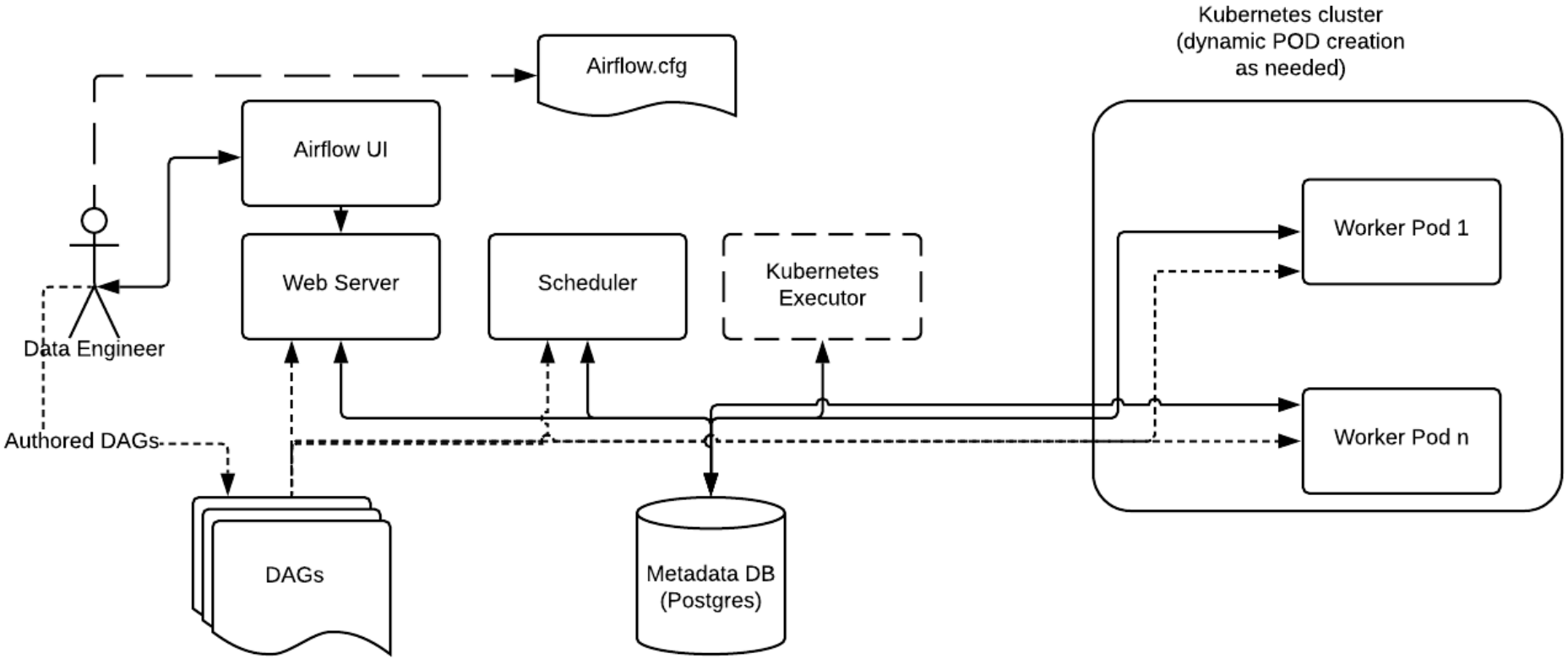


Quelle: https://airflow.apache.org/docs/apache-airflow/stable/executor/celery.html

Data pipelines mit Apache Airflow
Kubernetes Executor

Architektur

- \+ Worker Pods are only spun up when required for task execution.



Quelle: https://airflow.apache.org/docs/apache-airflow/stable/executor/kubernetes.html

Data pipelines mit Apache Airflow

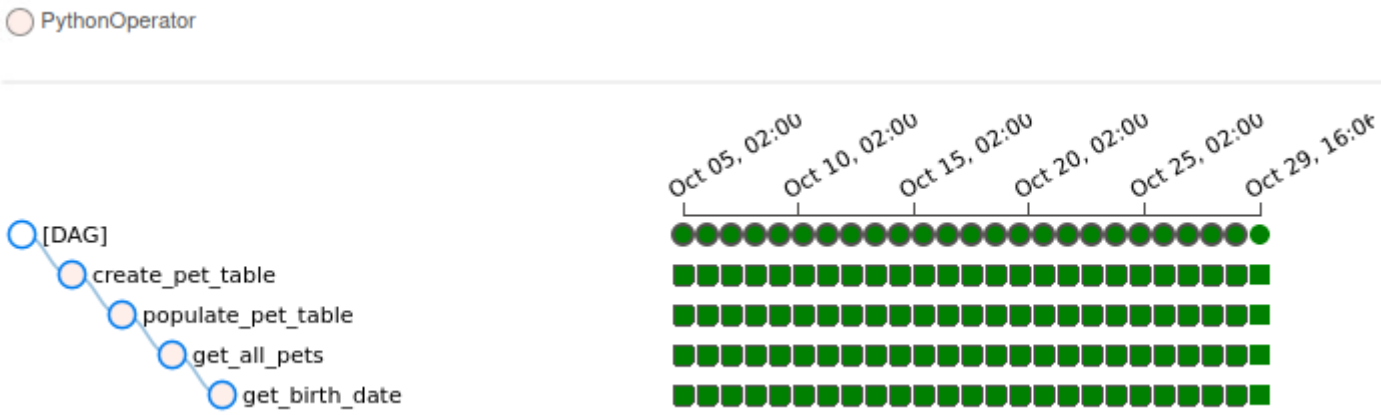
Übergabe von Daten zwischen Tasks

- Rückgabewerte und Übergabeparameter dürfen nur kleine Datenmengen enthalten (z.B. wenige Schlüsselwerte-Paare).
- Größere Datenmengen (z.B. ein pandas-Dataframe) müssen für die Übergabe in einer Datenbank oder auf einer Festplatte gespeichert werden.

Quelle: https://www.astronomer.io/guides/airflow-passing-data-between-tasks

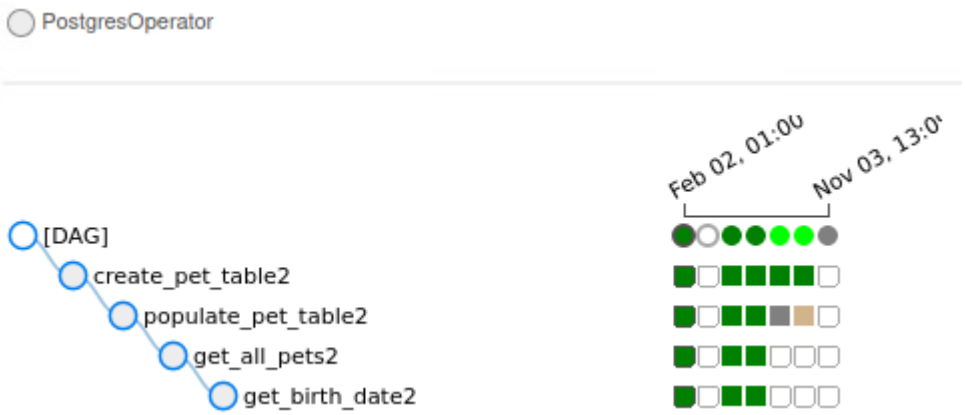
Data pipelines mit Apache Airflow

- Übung 21-access-postgres-pythonoperator.py
- Erstellen, Füllen und Abfrage einer Datenbanktabelle eines postgresql-Server mit `PythonOperator`



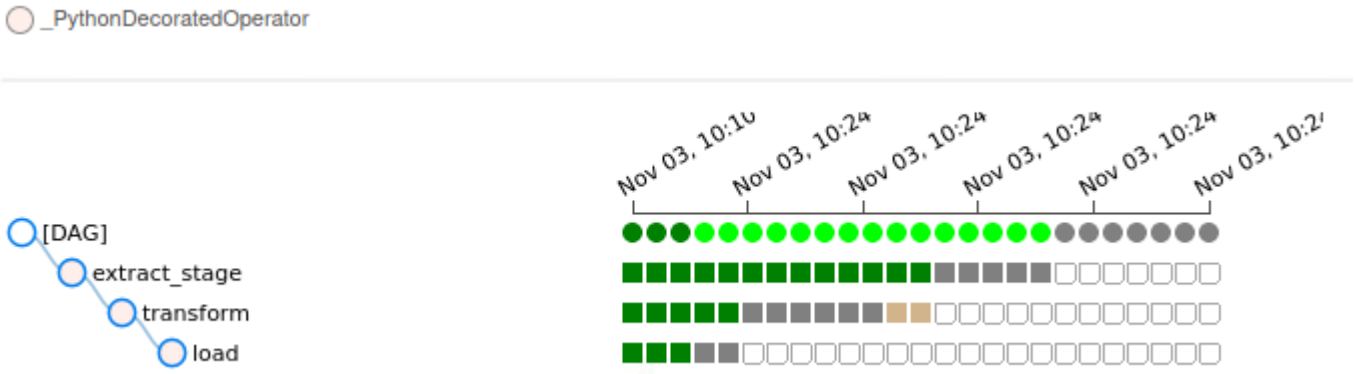
Data pipelines mit Apache Airflow

- Übung 22-postgres-operator-dag.py
- Erstellen, Füllen und Abfrage einer Datenbanktabelle eines postgresql-Server mit `PostgresOperator`



Data pipelines mit Apache Airflow

+ Übung 23-postgres-operator-dag-api.py
+ Nutzung des Taskflow APIs
- `extract_stage`
  Auslesen der Daten aus einer postgresql-Datenbank und „stagen“ der als pandas dataframe
- `transform`
  Einlesen des pandas dataframe und Ermittlung der Anzahl der Einträge
- `load`
  Speichern der Anzahl der Anzahl in einer Tabelle eines mssql-Servers



+ In der sqlcmd-shell können die Einträge ausgelesen werden:
- `SELECT * FROM num_pets`
  `GO`

Übungsaufgabe 8

Ausführungszeit for-loop gegenüber numpy

- **Ziel**
  Vergleich der Ausführungszeit einer for-Schleife gegenüber numpy
- **Vorgehensweise**
  Berechnen Sie eine Millionen Mal den Sinus einer Zahl plus Addition einer Zufahlszahl, z.B. mit

```
tsize=1000000
result=[]
for i in range(tsize):
    result.append(0.23*np.sin(2*np.pi*i)+random.random())
```

- Berechnen Sie ähnliche Werte mit numpy (np), z.B. mit

```
values=np.arange(tsize)
result2=0.23*np.sin(2*np.pi*values)+np.random.rand(tsize,)
```

- Messen und vergleichen Sie die Ausführungszeiten.
- Erstellen Sie eine einseitige Präsentationsfolie, auf der Sie das Ziel, die Vorgehensweise, ihr Ergebnis und ihre Schlussfolgerung skizzieren.

Übungsaufgabe 9

AMQP mit airflow

+ **Ziel**
  Empfangen und speichern Sie Daten von einem AMQP-Client in einer Datenbank mit Hilfe von airflow.
+ **Vorgehensweise**
  Starten Sie ähnlich wie in Übung 13-AMQP-RABBITMQ-WQ-SQL einen producer (01-new-task.py), der eine zufällige Anzahl von Nachrichten erzeugt und in eine Queue schreibt. Die Anzahl der Nachrichten soll im Intervall [10,530] liegen.
+ Schreiben Sie einen DAG, der
- in einem Task `receiveAMQP` (z.B. mit einem `PythonOperator`)
  die Nachrichten aus der Queue ausliest,
  als pandas dataframe speichert und
  den Dateinamen zurückgibt,
- in einem zweiten Task `transform` den pandas dataframe ausliest und die Anzahl der erhaltenen Nachrichten zurückgibt
- und in einem dritten Task `load` die Anzahl der Nachrichten in einer Datenbank speichert.

+ Erstellen Sie eine einseitige Präsentationsfolie, auf der Sie das Ziel und die Vorgehensweise skizzieren.

Hochschule Karlsruhe
University of
Applied Sciences

www.h-ka.de