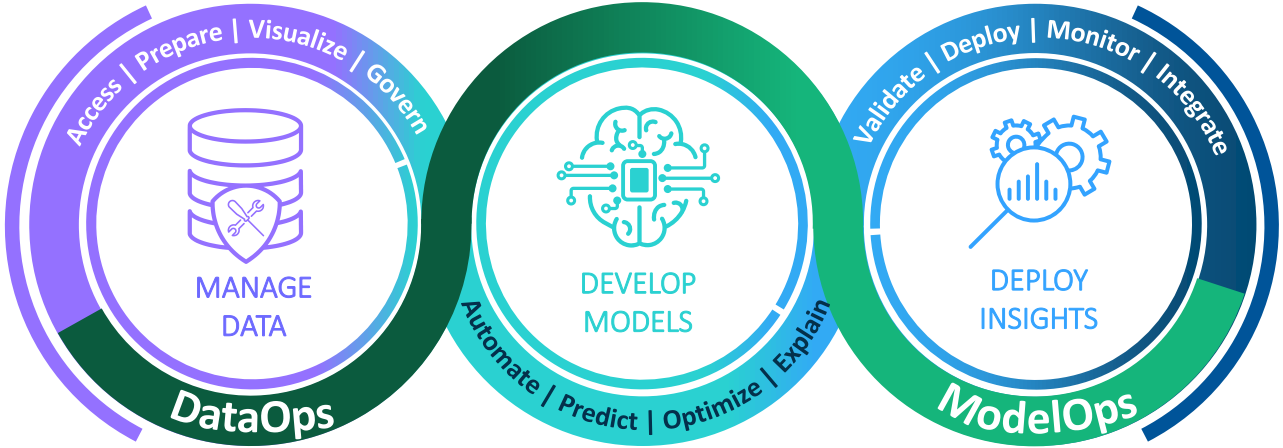
SAS und DevOps…

…und ein Blick auf das SAS Studio in SAS Viya

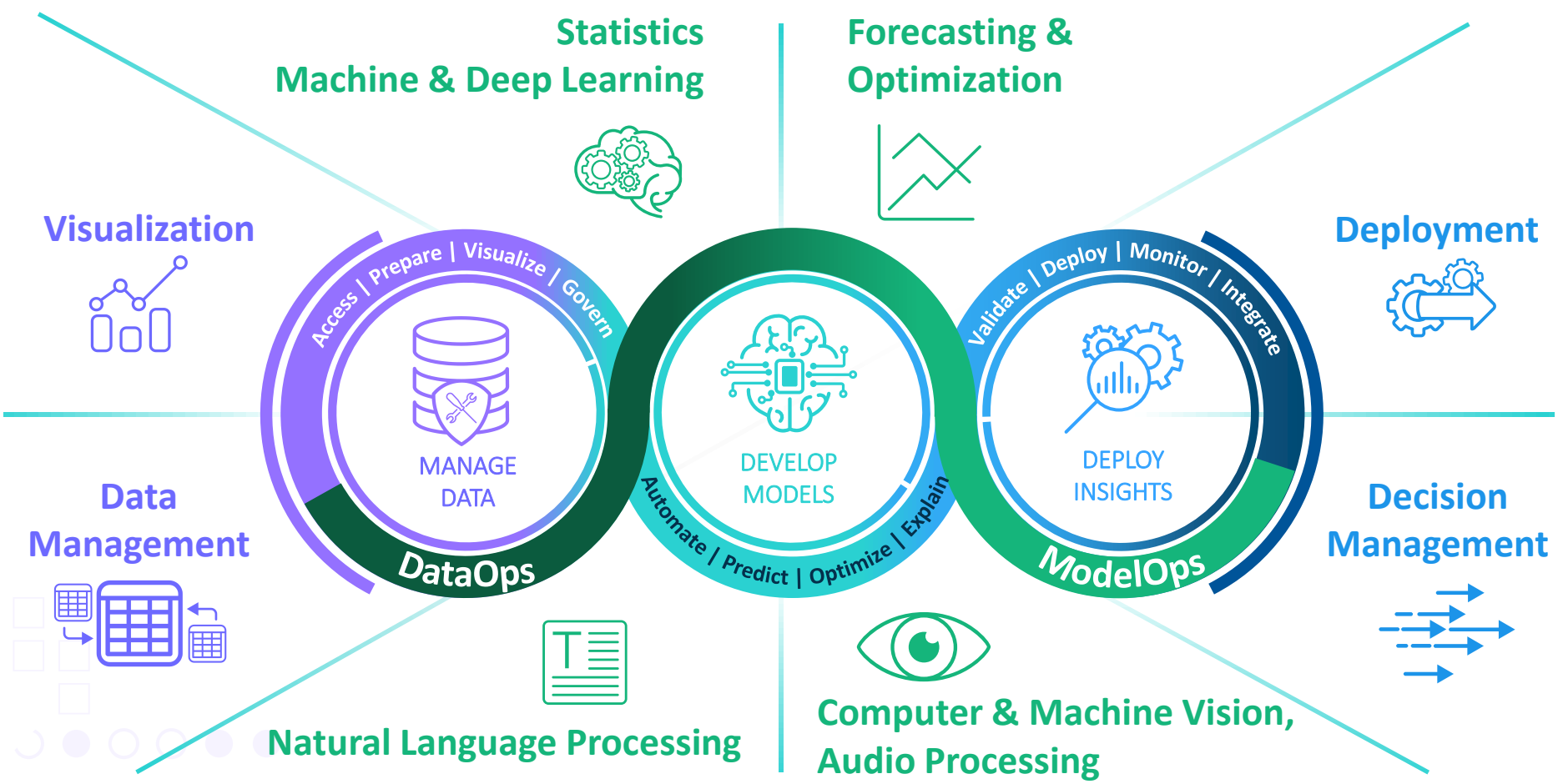
Ops, I did it .. again

Analytics Lifecycle

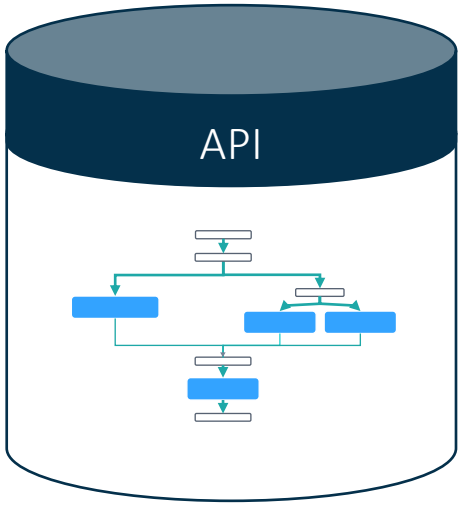
QUESTION



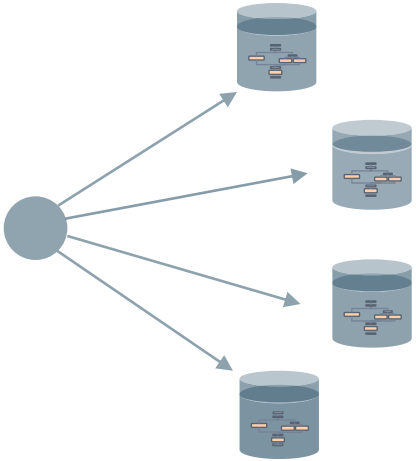
DECISION



CONTAINERIZE



DEPLOY



INTEGRATE

8 wichtige Eigenschaften von SAS Viya

SAS Viya – Analytik All-in-one

Output und Ergebnisse

SAS in Action

Governance

1 Analyseergebnisse in **Dashboards** aufbereiten und bereitstellen

2 **Big Data Analytics** – Mit NLP, Computer Vision, Machine/DeepLearning für neue Anforderungen gerüstet sein

```
%let id = SASClub22;
proc transpose;
proc report;
proc summary;
data _NULL_ ;
```

3 **SAS Programmierung** und bestehende Auswertungen weiterhin nutzen und ausbauen

4 Integration von SAS und **Open Source**

5 Erhöhung der Produktivität und Einbeziehen von unterschiedlichen Benutzergruppen mit **NoCode/LowCode**

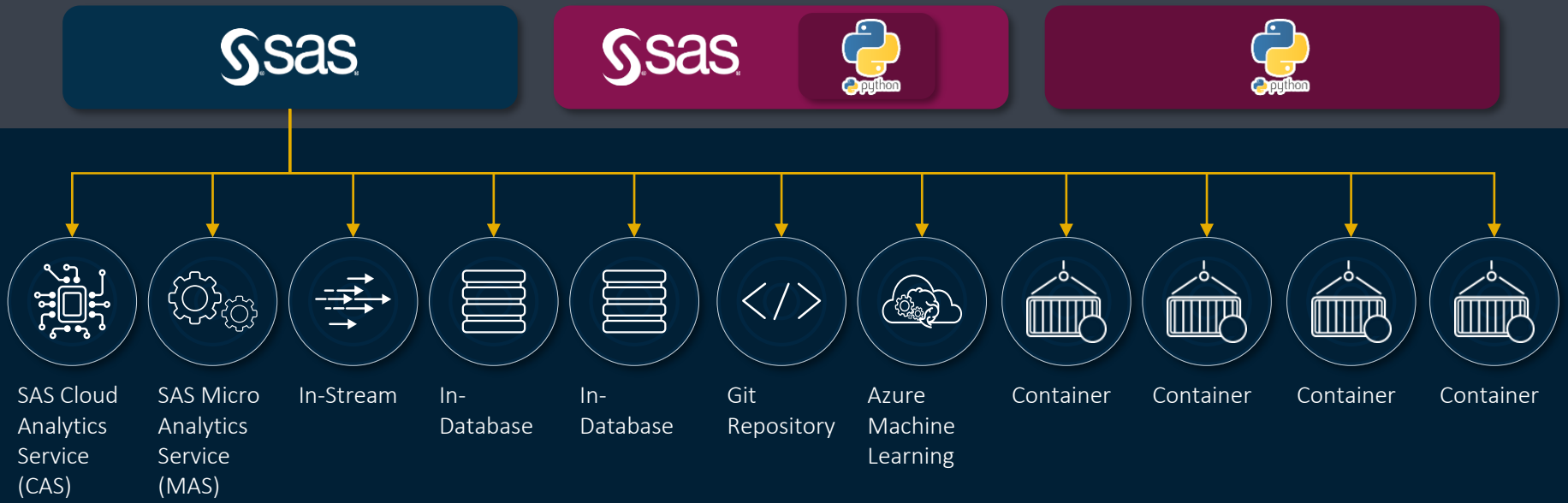
6 **Analyse-Code** strukturieren, dokumentieren, verwalten und wiederverwenden

7 Unterstützung des statistischen Lebenszyklus: **Data Governance, Model Governance**

8 **DevSecOps** Einbeziehung von Schutzbedarf

Publishing Destinations

Model source



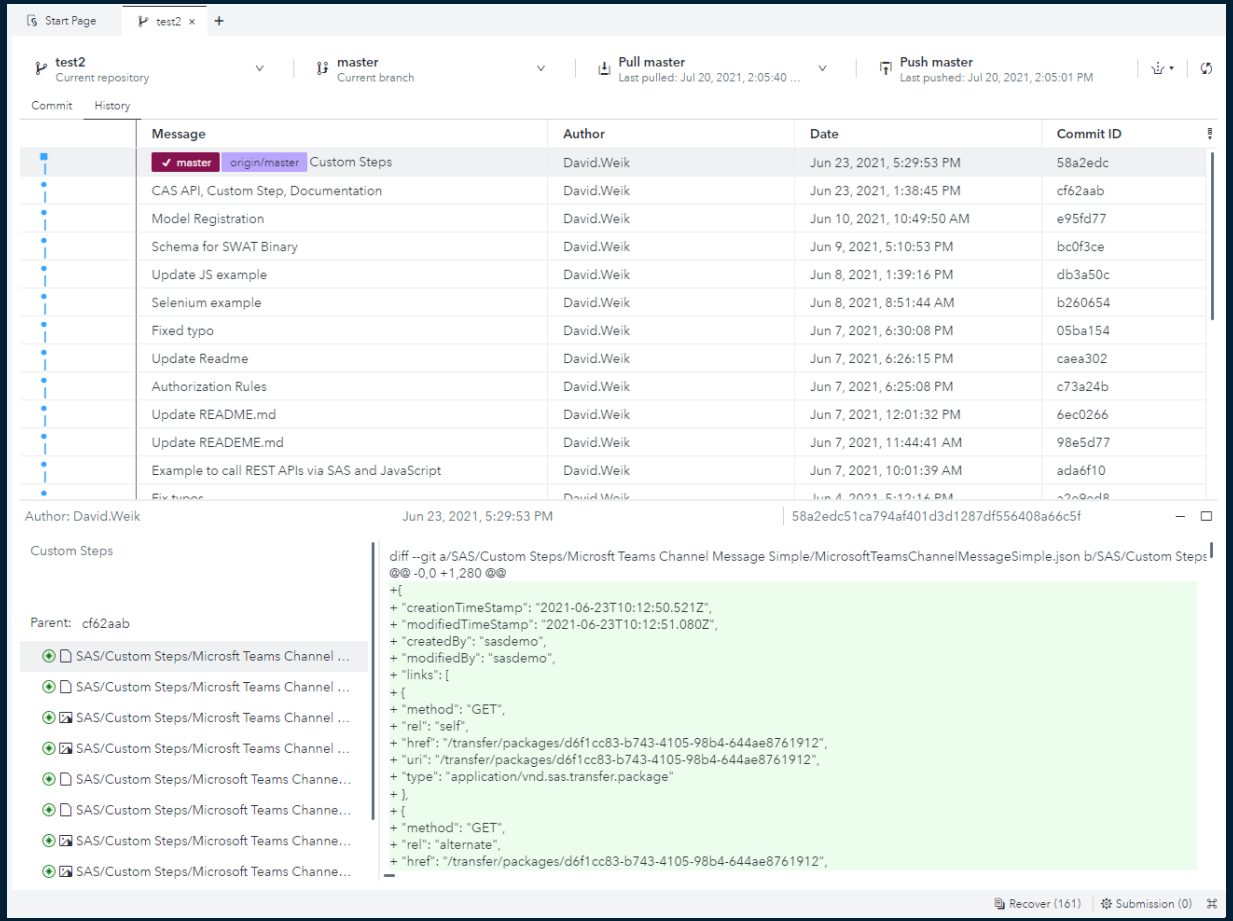
https://go.documentation.sas.com/doc/en/mdlmgrcdc/v_017/mdlmgrug/n1nrmfpk1ysdt6n1a3ysrcljhuuy.htm

Versionierung

Kollaboration & Nachvollziehbarkeit

- Integration mit Git für:
  - SAS Code Dateien
  - Python Code Dateien
  - Custom Steps
  - Flows
  - Modelle
  - Entscheidungen
  - Beliebige andere Dateien wie z.B. csv, txt, …

Git Oberfläche



- GIT_ Functions
- GIT_BRANCH_CHKOUT Function
- GIT_BRANCH_DELETE Function
- GIT_BRANCH_MERGE Function
- GIT_BRANCH_NEW Function
- GIT_CLONE Function
- GIT_COMMIT_FREE Function
- GIT_COMMIT Function
- GIT_COMMIT_GET Function
- GIT_COMMIT_LOG Function
- GIT_DELETE_REPO Function
- GIT_DIFF_FILE_IDX Function
- GIT_DIFF_FREE Function
- GIT_DIFF Function
- GIT_DIFF_GET Function
- GIT_INDEX_ADD Function
- GIT_INDEX_REMOVE Function
- GIT_PULL Function
- GIT_PUSH Function
- GIT_RESET_FILE Function
- GIT_RESET Function
- GIT_STATUS_FREE Function
- GIT_STATUS Function
- GIT_STATUS_GET Function
- GIT_VERSION Function

Git im Data Step

Warum klicken wenn ich programmieren kann!

24 Funktionen um mit Git Code basiert arbeiten zu können

```
data _null_;
 rc= git_pull(
   "your-local-repository",
   "ssh-user-name",
   "ssh-password",
   "ssh-public-key",
   "ssh-private-key");
 rc= git_push(
   "your-local-repository",
   "ssh-user-name",
   "ssh-password",
   "ssh-public-key",
   "ssh-private-key");
run;
```

Linting für SAS

Einheitliches SAS-Code Bild

```
{
    "noEncodedPasswords": true,
    "hasDoxygenHeader": true,
    "hasMacroNameInMend": true,
    "hasMacroParentheses": true,
    "indentationMultiple": 2,
    "lowerCaseFileNames": true,
    "maxLineLength": 80,
    "noNestedMacros": true,
    "noSpacesInFileNames": true,
    "noTabIndentation": true,
    "noTrailingSpaces": true
}
```

Testen von SAS Code

Community-Getriebenes Test-Framework

- mp_assert - generic assertion
- mp_assertcols - Asserts the existence (or not) of certain columns
- mp_assertcolvals - Asserts the existence (or not) of particular column values
- mp_assertdsobs - Asserts the existence (or not) of dataset observations
- mp_assertscope - Compares before/after to detect scope leakage in a SAS Macro

GitOps

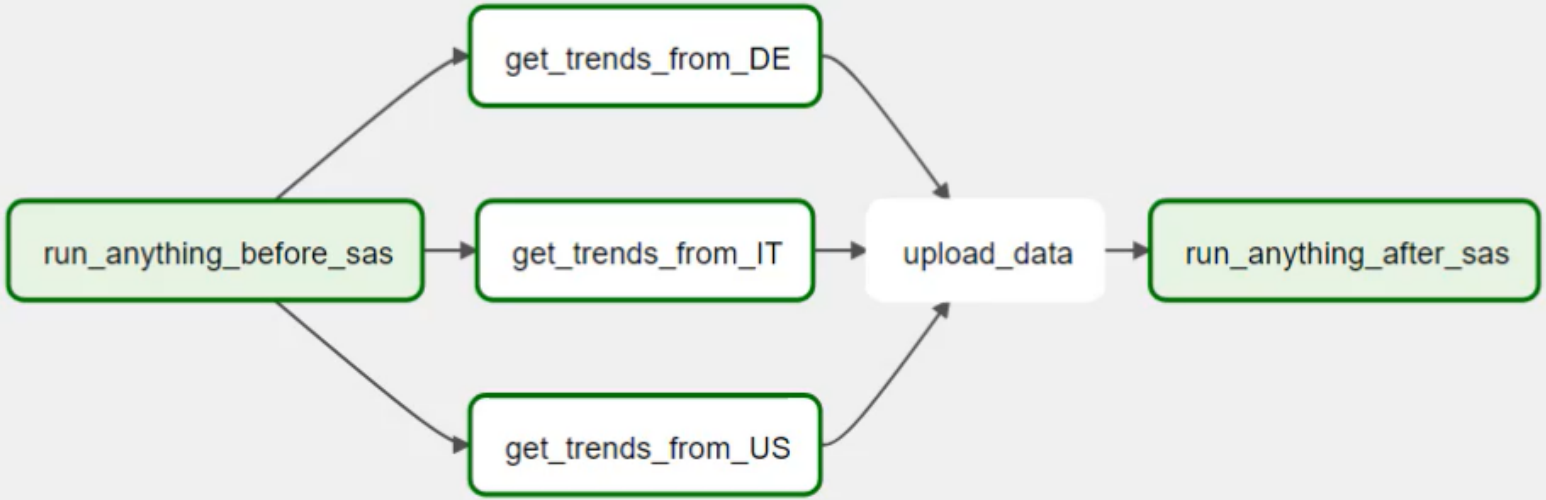
.gitlab-ci.yml 2.67 KiB

Open in Web IDE | Replace | Delete

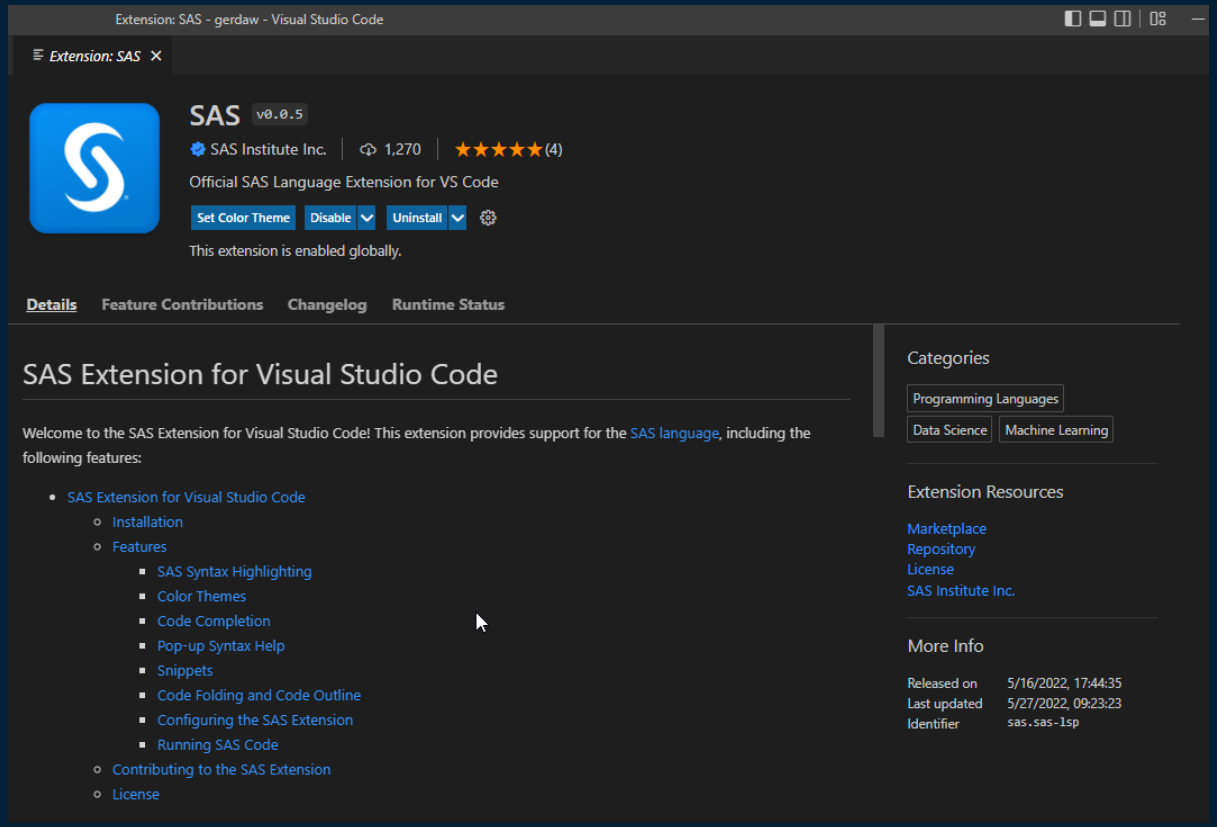
```yaml
# This file is a template, and might need editing before it works on your project.
# To contribute improvements to CI/CD templates, please follow the Development guide at:
# https://docs.gitlab.com/ee/development/cicd/templates.html
# This specific template is located at:
# https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Getting-Started.gitlab-ci.yml

# This is a sample GitLab CI/CD configuration file that should run without any modifications.
# It demonstrates a basic 3 stage CI/CD pipeline. Instead of real tests or scripts,
# it uses echo commands to simulate the pipeline execution.
#
# A pipeline is composed of independent jobs that run scripts, grouped into stages.
# Stages run in sequential order, but jobs within stages run in parallel.
#
# For more information, see: https://docs.gitlab.com/ee/ci/yaml/index.html#stages

# Use 'pipelines for merge requests'
# Pipeline is run every time you make changes to the source branch for a merge request
# https://docs.gitlab.com/ee/ci/pipelines/merge_request_pipelines.html
workflow:
  rules:
    #- if: $CI_PIPELINE_SOURCE == 'merge_request_event'
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH

stages:          # List of stages for jobs, and their order of execution
  - build
  - test
  - deploy

build-job:       # This job runs in the build stage, which runs first.
  stage: build
  script:
    - echo "Running SAS Studio Flow..."
    #- echo "Compiling the code..."
    #- echo "Compile complete."
    #- pwd
    #- echo CI_PROJECT_DIR=$CI_PROJECT_DIR
    #- python3 main.py --sas-endpoint http://server.demo.sas.com --user sasdemo --password Orion123 --flow $CI_PROJECT_DIR/summary-flow.flw
    #- python3 -m pip install requests
    - python3 main.py --sas-endpoint $_sas_endpoint --user $_user --password $_password --flow $CI_PROJECT_DIR/summary-flow.flw
  artifacts:
    name: "$CI_PIPELINE_ID-$CI_JOB_NAME"
    paths:
      - "*.log.txt"
      - "*.list.txt"
```

GitOps Visuell

Orchestrierung über Scheduler



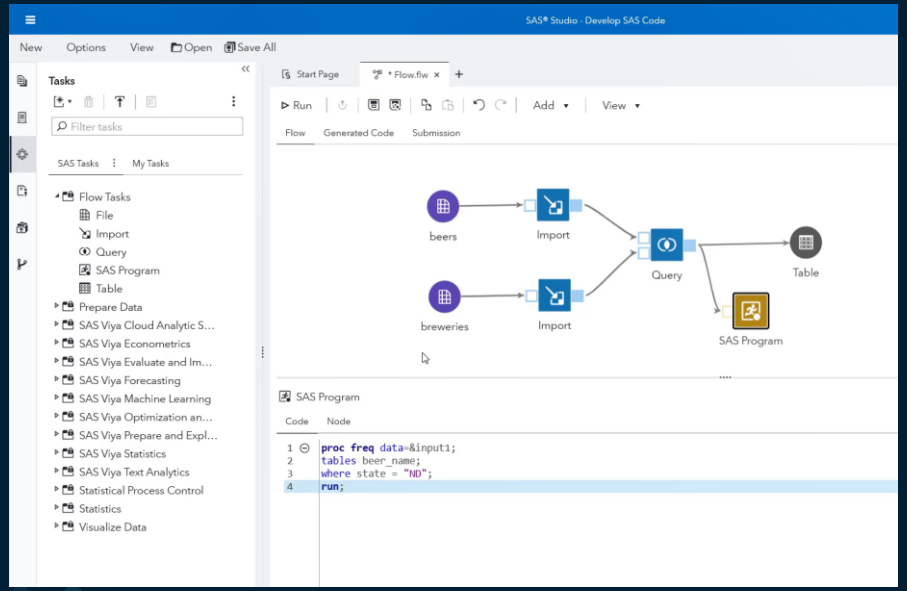
SAS Erweiterung für VS Code



SAS Studio

SAS Studio

Vereinheitlichter Flow Designer auf SAS Viya



Die Konvergenz von SAS Enterprise Guide und SAS Data Integration Studio richtet sich an Programmierer, Datenanalysten und ETL-Entwickler.

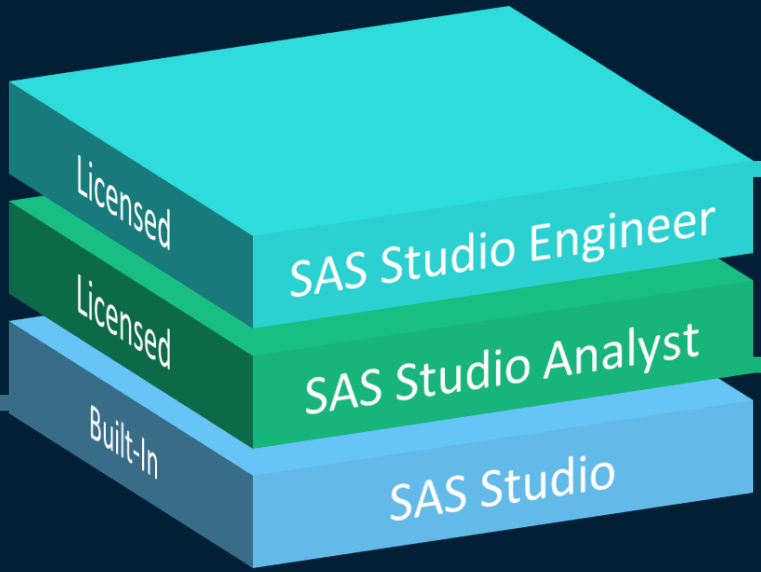
- Funktionale Äquivalenz
- Funktionsgleichheit und neue Funktionen
- Umstellung von SAS 9.4

SAS Studio – Rollenbasierte Nutzung

Integrierte Entwicklungsumgebung für Analytik und Datenmanagement

Erstellen wiederverwendbarer fortschrittlicher Datenverwaltungsabläufe für Bereitstellung, Synchronisierung oder weitere Aktivitäten

Engineer



Analyst

Analysieren, Aufbereiten und Bereitstellen von Daten für Analysezwecke

SAS Studio

Code, Queries bauen, Tasks & Basis Flows

SAS Studio

Übersicht über die Features der drei Stufen

SAS STUDIO (PROGRAMMIERER)

- Code Editor
- GIT Integration
- Tasks, Snippets und Queries
- Datei Import
- Flow Designer inklusive Table, File, Import, SAS Program, Query, Sort

SAS STUDIO ANALYST

SAS Studio (Programmierer)

Erweiterter Flow Designer inklusive

- Data Movements (CRUD)
- Data Preparation for Analytics
- Custom Steps & Prompts
- Data Quality

SAS STUDIO ENGINEER

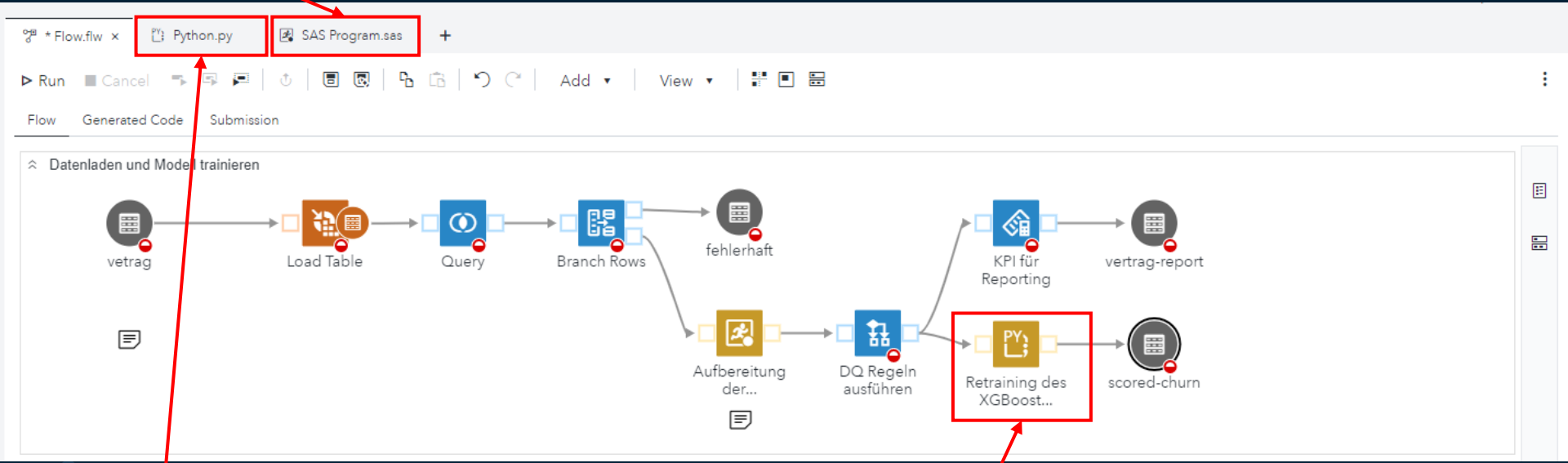
SAS Studio Analyst

- Information Catalog
- In-DB Technologies
  Teradata, Hadoop, Spark
- Erweiteter Flow Designer inklusive ETL spezifische Flow Steps & Optionen
- Code Optimization

SAS Studio

Kombination von visuellen Prozessflüssen mit Code-Editoren

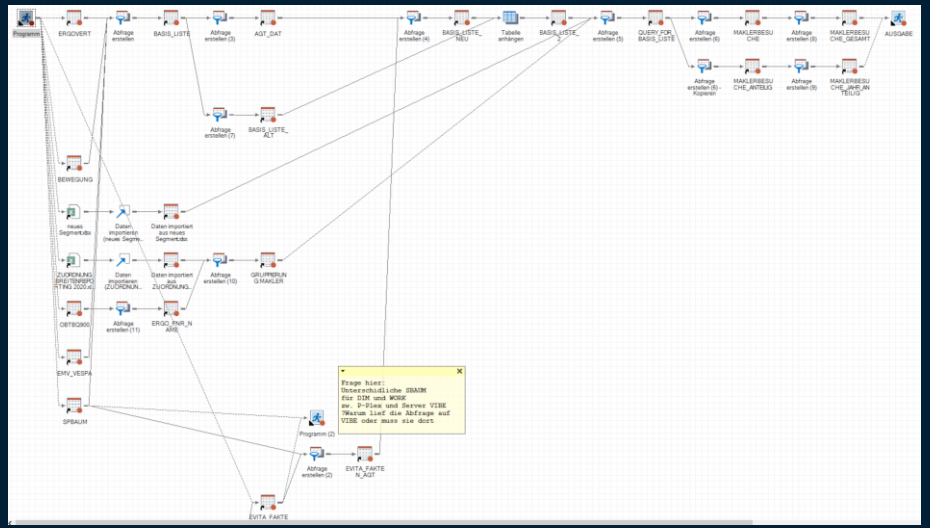
SAS Code Editor



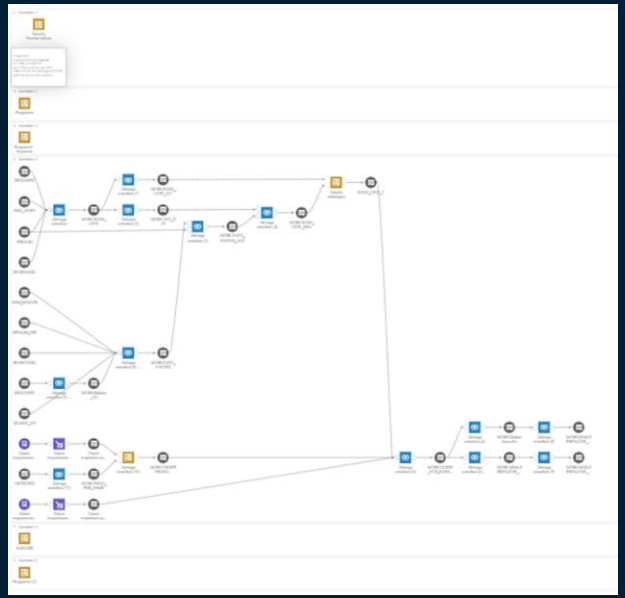
Python Code Editor

Python Programm als Schritt im Flow

SAS Enterprise Guide



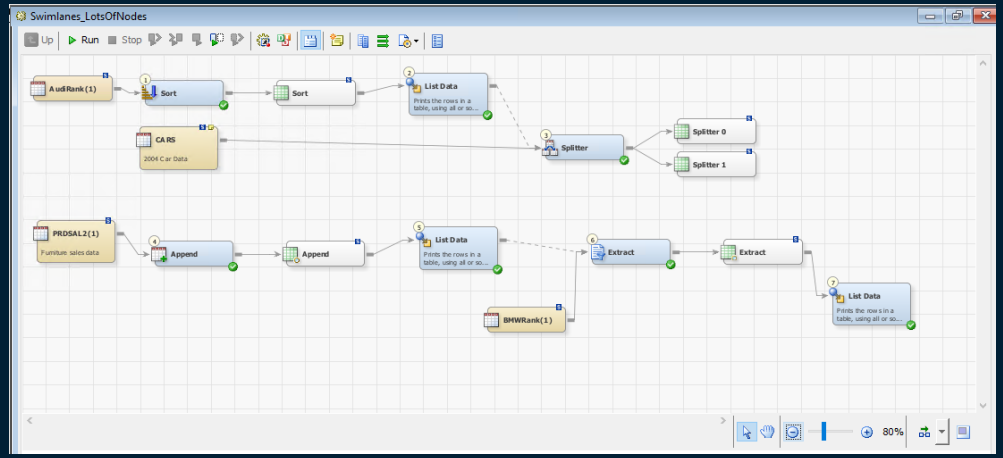
SAS Studio



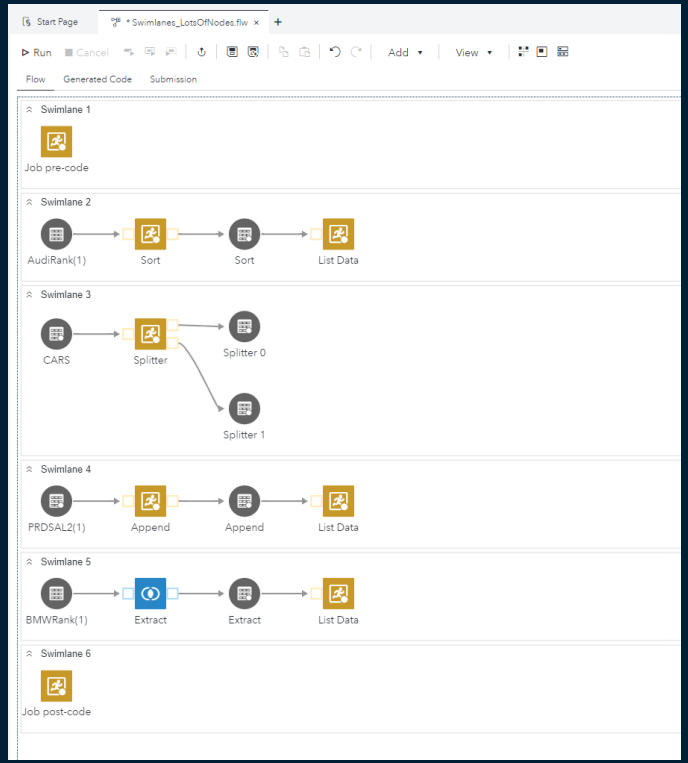
SAS 9

SAS Viya

SAS Data Integration Studio



SAS Studio



SAS 9

SAS Viya

Demo

sas.com