

Scheduling für SAS Viya auf Kubernetes mit externem Scheduler

Alexey Vodilin, Product Manager
Data & Analytics Engineering

Development process

Develop

Orchestrate / Schedule / Monitor

Development process

Develop

Orchestrate / Schedule / Monitor



Development process

Develop

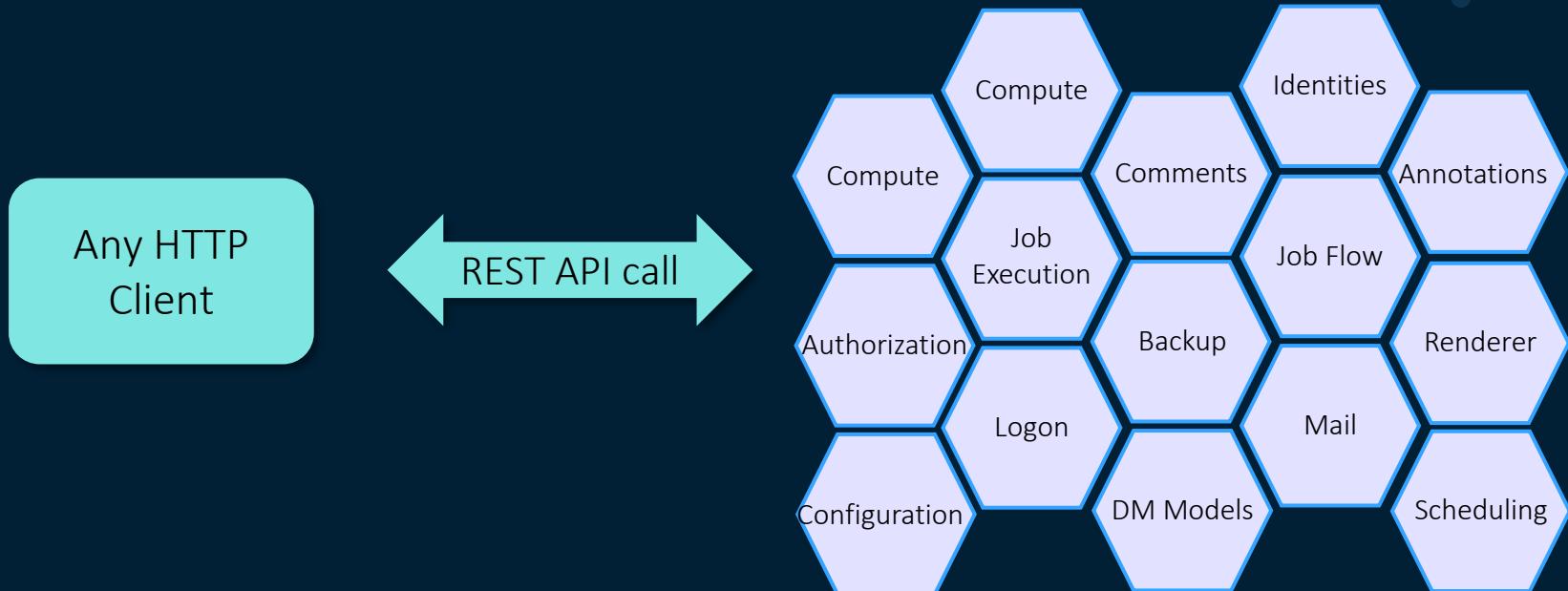
Orchestrate / Schedule / Monitor



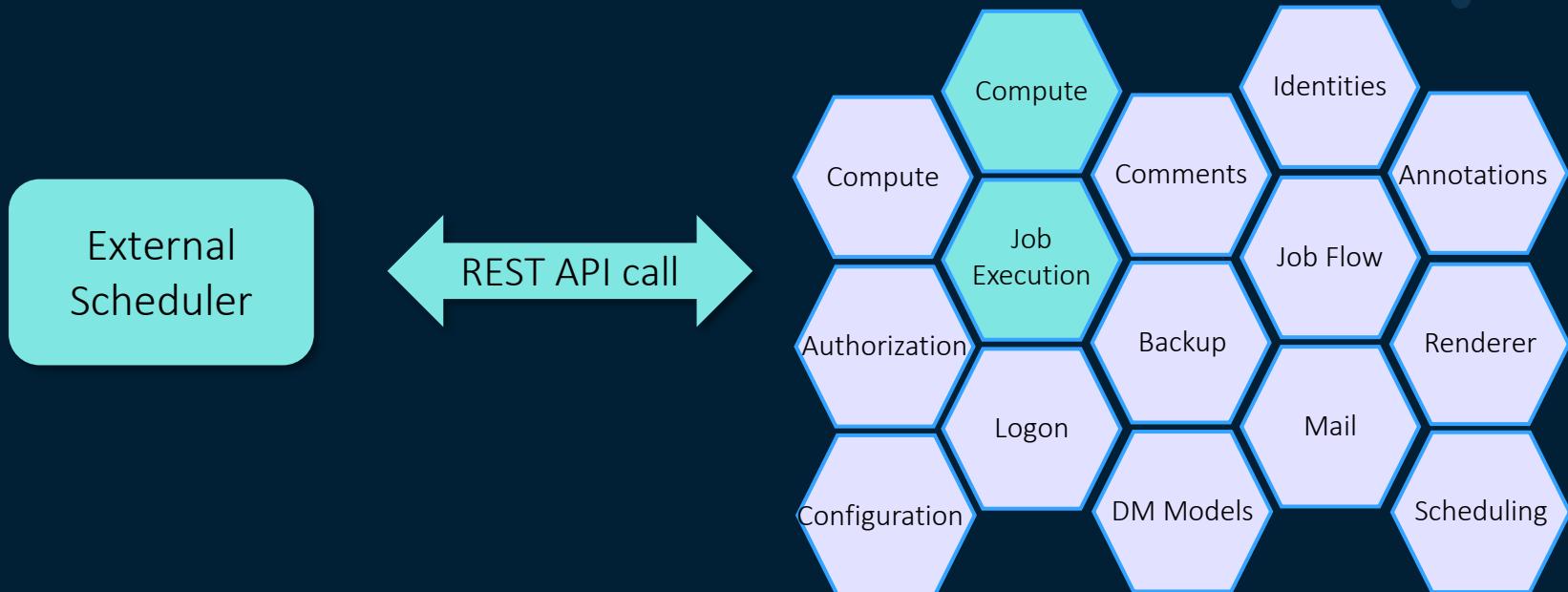
External tools



Viya REST APIs



Viya REST APIs



Viya REST API – documentation

developer.sas.com

APIs by category

Information about specific API request

Examples for different languages

The screenshot shows a browser window displaying the SAS Viya REST API documentation. The URL in the address bar is `developer.sas.com/apis/rest/Compute/?python#execute-sas-code-in-a-session`. The page content is as follows:

SAS Viya REST APIs

- Getting Started
- Analytics Insights
- Visualization
- Compute
- Data Management
- Decision Management
- Core Services
- Automated Machine Learning
- Support And Feedback

Compute

- CAS Management
- Job Definitions
- Job Execution
- Compute** (selected)
- Usage Notes
- Operations

Execute SAS code in a session

POST /sessions/{sessionId}/jobs

Executes SAS code in the specified session. Code is always submitted asynchronously. URLs are returned that contain endpoints. The Location header contains the URI of the job resource. You might submit the code directly in a request or as a reference to a File service resource.

Parameters

Name	In	Type	Required	Description
sessionId	path	string	true	Specifies the ID of the session.
body	body	jobRequest	true	Specifies the job submission request.

Responses

Status	Meaning	Description	Schema
201	Created	A job was created.	job
400	Bad Request	The request was invalid.	errorResponse
404	Not Found	No session exists at the requested path.	Inline

Response Schema

Status Code 400

Error

Code samples

shell javascript python go

```
import requests
headers = {
    'Content-Type': 'application/vnd.sas.compute.job.request+json',
    'Accept': 'application/vnd.sas.compute.job+json'
}

r = requests.post("https://example.com/compute/sessions/{sessionId}/jobs",
                  headers=headers)

print(r.json())
```

Body parameter

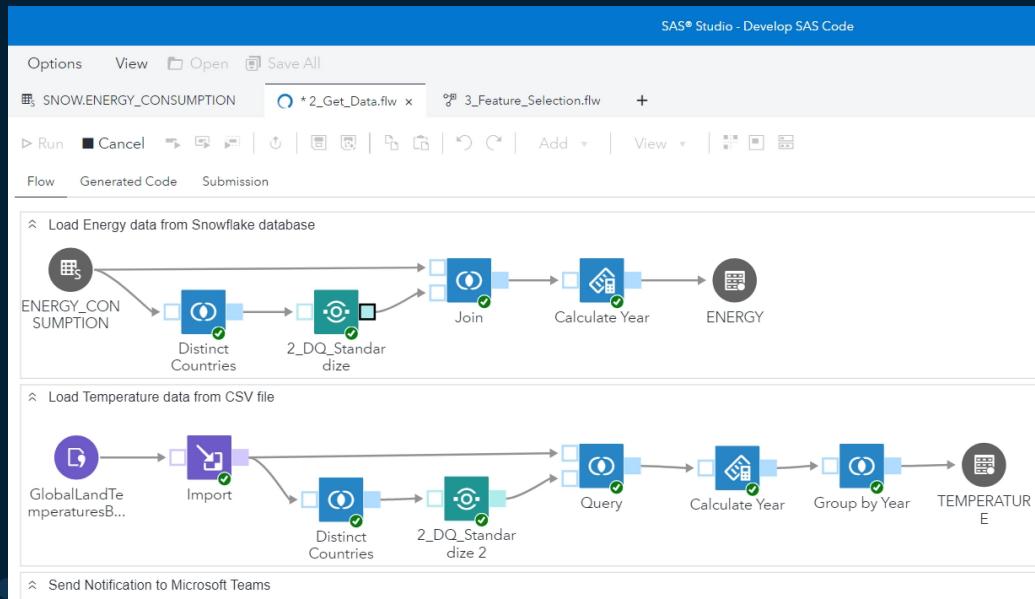
```
{
  "version": 2,
  "name": "MyOptions",
  "description": "Getting the options for my session.",
  "code": [
    "proc options;",
    "run;"
  ],
  "attributes": {
    "resetLogLineNumbers": false
  }
}
```

APIs for
artifacts
execution

Supported SAS Studio artifacts

All major artifacts are available for execution via API

Flows



Programs and Jobs

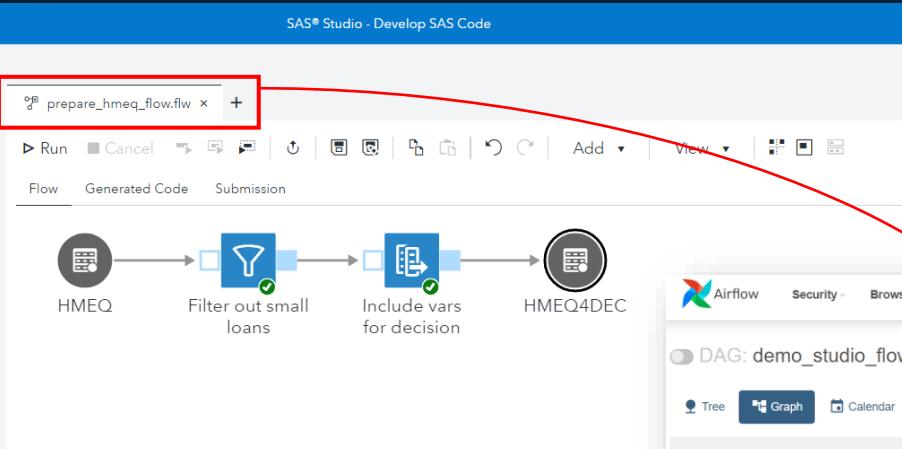
The screenshot shows the SAS Studio interface with a code editor displaying a SAS program:

```
/* _STARTMACROCODE_ */
/
/ SASStudio initialization file for SAS workspace connections
/
/*
/* Get the Git version */
DATA _NULL_;
GITVERSION = GIT_VERSION();
CALL SYMPUT('_GITVERSION', GITVERSION);
RUN;
libname airdemo "/appdata/nfs/users/data/airflowdemo";
%MACRO resolveHomeDirectory;
%GLOBAL _USERHOME;
%LOCAL _HOMEVAR;
%IF (%$YSSCP=WIN) %THEN
%DO;
%LET _HOMEVAR=USERPROFILE;
%END;
%ELSE
%DO;
%LET _HOMEVAR=HOME;
%END;
%IF _HOMEPATH=%$V$C$F$C%$V$GET(& HOMEVAR));

```

The code editor has tabs for "Code" and "Preview". The left pane shows the SAS Explorer tree with nodes like "SAS Content", "Decision Repository", and "Public". The right pane shows the code with syntax highlighting and line numbers.

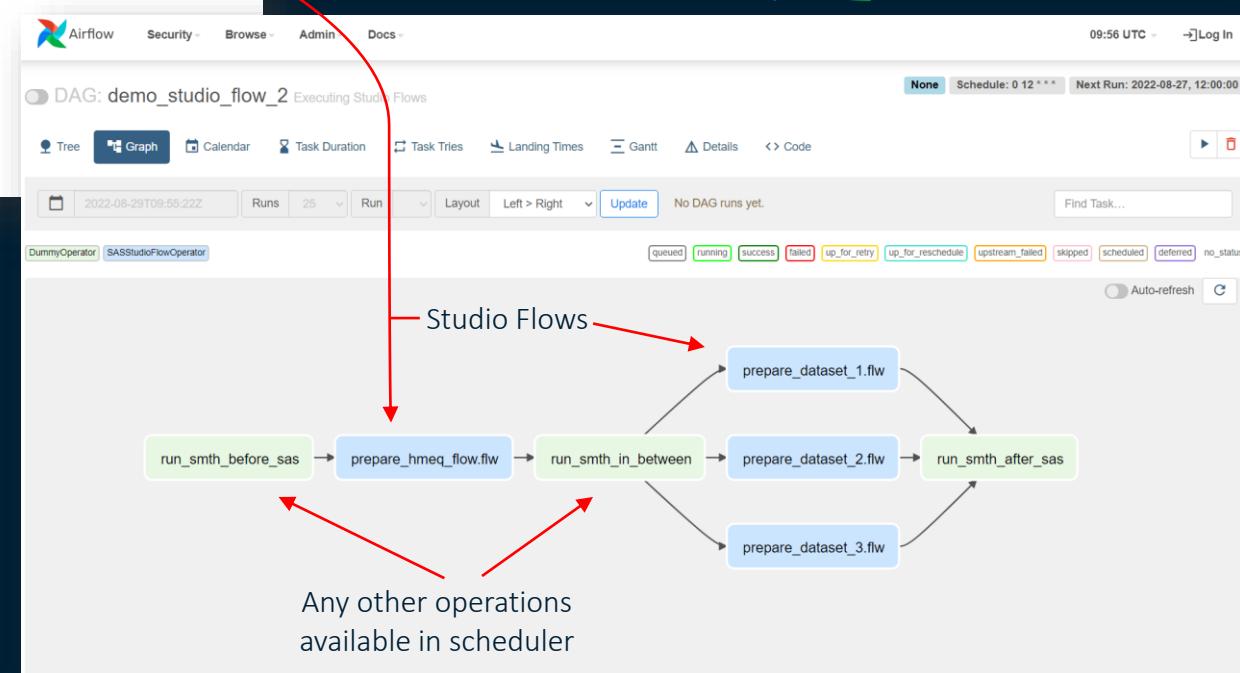
SAS Studio Flow



Exec status & log



Airflow DAG



Example of SAS logs in Airflow

Airflow Security Browse Admin Docs

```
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 70    %macro web_list_catalogs(library);
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 71
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 72    %macro web_list_catalogs(library);
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 73    %let library=%upcase(&library);
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 74      proc sql ;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 75      create table work.catalogs as select memname as Catalog, memtype as
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 76      Type, engine as Engine from sashelp.vmember where
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 77      libname=&library" and memtype="CATALOG";
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 78      run;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 79      quit;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 80      title "Catalogs in &library";
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 81
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 82      proc print data=work.catalogs;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 83      run;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 84    %mend;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 85
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 86    %macro web_replay_grseg(catalog,entry);
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 87    proc greplay nofs igout=&catalog;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 88    replay &entry;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 89    run;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 90    %mend;
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO -
[2022-08-25, 08:22:51 UTC] {logging_mixin.py:109} INFO - source: 91
```

Summary

- SAS artifacts can be used with external scheduler/orchestrator
- Integration is not limited to a particular tool, but rather generic
- All APIs are documented and available on developer.sas.com

Thank you