

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

 **sas** Studio

Development process

Develop

 **sas** Studio

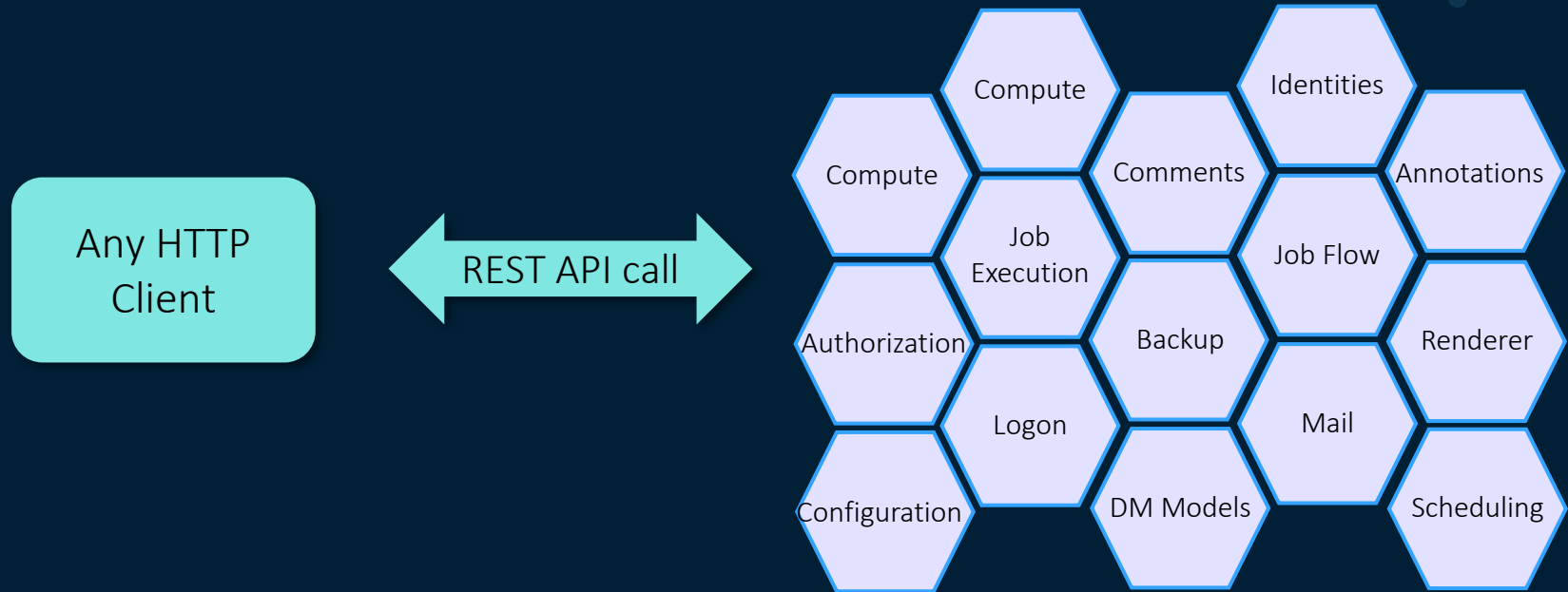
Orchestrate / Schedule / Monitor

 **sas** JFS

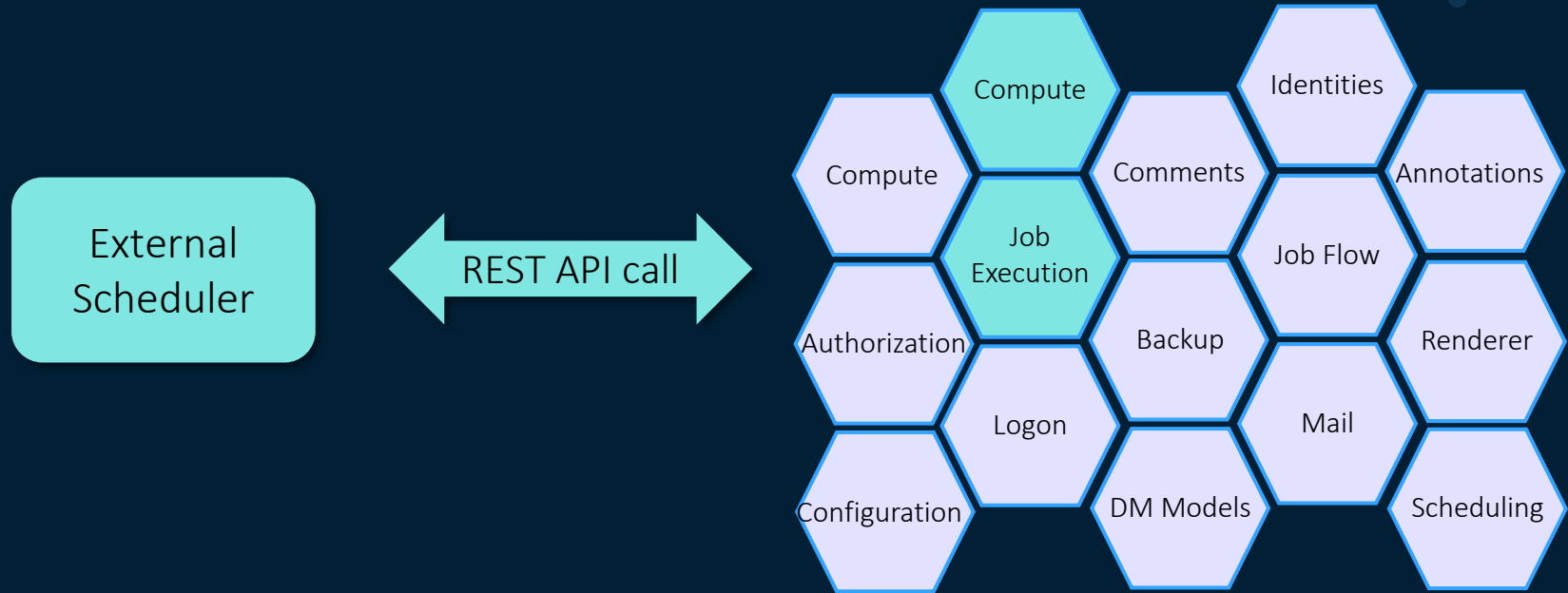
External tools



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 the SAS Viya REST API documentation page for the endpoint `POST /sessions/{sessionId}/jobs`. The page is titled "Execute SAS code in a session" and provides details about the request and response.

APIs by category: The left sidebar shows a list of API categories. The "SAS Viya REST APIs" category is highlighted, and the "Compute" sub-category is selected.

Information about specific API request: The main content area displays the endpoint `POST /sessions/{sessionId}/jobs` and its description: "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: The table below lists the parameters for the request.

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: The table below lists the possible responses from the API.

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: The status code 400 is shown, along with an "Error" section.

Examples for different languages: The right sidebar shows code samples for different languages. The "python" tab is selected, showing a Python example using the `requests` library to submit a job request.

```
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',
print(r.json())
```

Body parameter: The right sidebar also shows a JSON example for the body parameter.

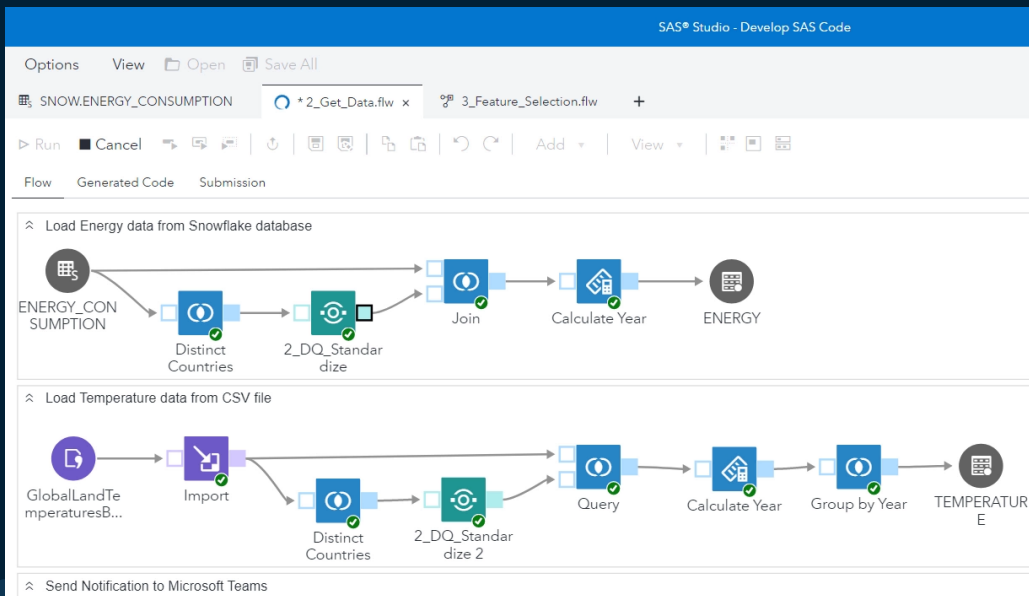
```
{
  "version": 2,
  "name": "MyOptions",
  "description": "Getting the options for my session.",
  "code": [
    "proc options;",
    "run;"
  ],
  "attributes": {
    "resetLogLineNumbers": false
  }
}
```

APIs for artifacts execution

Supported SAS Studio artifacts

All major artifacts are available for execution via API

Flows



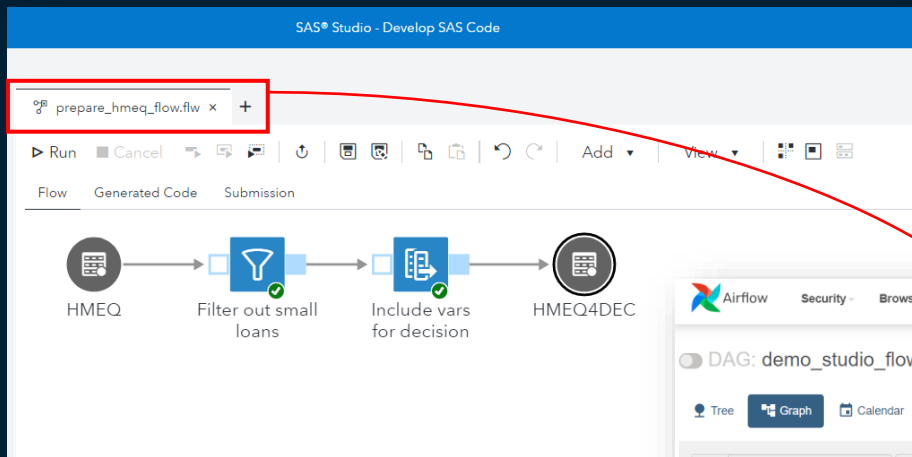
Programs and Jobs

The screenshot displays the SAS Studio interface with a SAS program. The 'Explorer' pane on the left shows a tree view of the workspace, including 'My Favorites', 'Folder Shortcuts', 'SAS Server', 'SAS Content', 'Conversational Flows', 'Decision Repository', 'ESP Projects', 'Model Repositories', 'Products', 'Projects', 'Public', 'Demos', 'Airflow', 'get_trends', 'get_trends.flw', 'Google-Trends.step', 'Tests', 'xyz_archive', and 'Users'. The main editor shows a SAS program with the following code:

```
1 /* _STARTMACROCODE_ */
2 /*-----*/
3 /
4 / SASstudio initialization file for SAS workspace connections
5 /
6 /*-----*/
7
8 /* Get the Git version */
9 DATA _NULL_;
10   GITVERSION = GIT_VERSION();
11   CALL SYMPUT('_GITVERSION', GITVERSION);
12 RUN;
13
14 libname airdemo "/appdata/nfs/users/data/airflowdemo";
15
16 %MACRO resolveHomeDirectory;
17   %GLOBAL _USERHOME;
18   %LOCAL _HOMEVAR;
19
20   %IF (&SYSSCP=MIN) %THEN
21     %DO;
22       %LET _HOMEVAR=USERPROFILE;
23     %END;
24   %ELSE
25     %LET _HOMEVAR=HOME;
26   %END;
27
28   %LET _USERHOME= %SYSFUNC(%SYSGET(%&_HOMEVAR));
29
```

The interface includes a menu bar (New, Options, View, Open, Save All), a toolbar with various icons, and a sidebar with 'Run', 'Generated Code', and 'Submission' tabs.

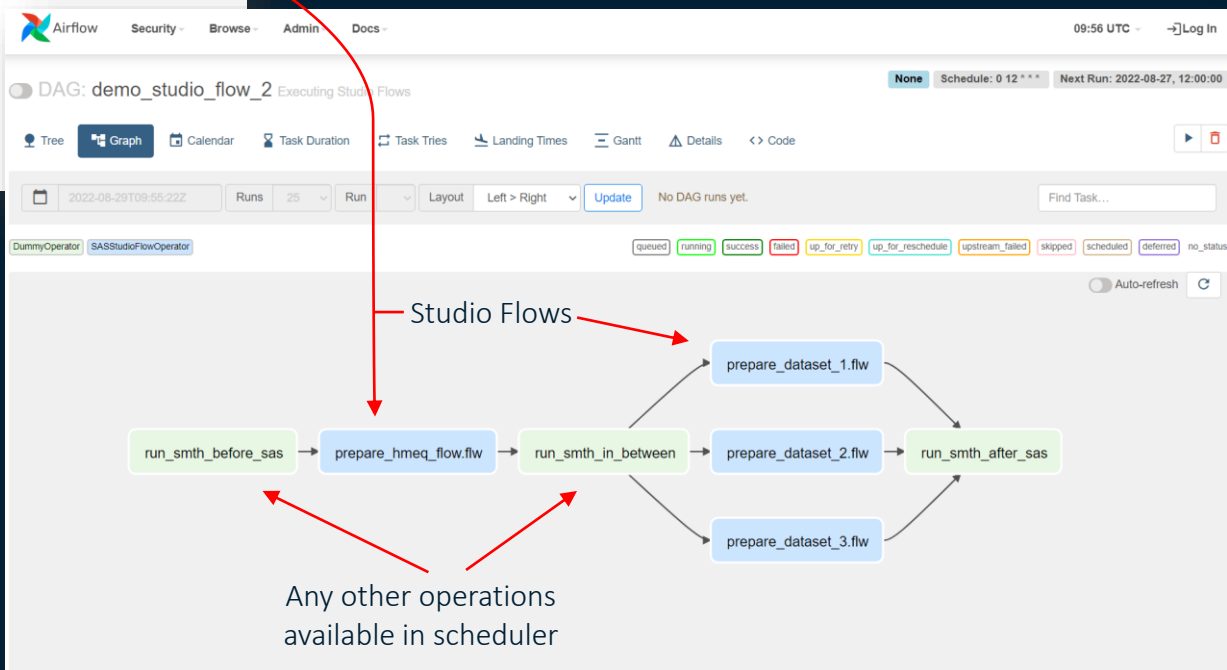
SAS Studio Flow



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	menu;
[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	source: 72	%macro web_list_catalogs(library);
[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	source: 74	proc sql ;
[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	source: 76	Type, engine as Engine from sashelp.vmember where
[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	source: 78	run;
[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	source: 80	title "Catalogs in &library";
[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	source: 82	proc print data=work.catalogs;
[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	source: 84	%mend;
[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	source: 86	%macro web_replay_grseg(catalog,entry);
[2022-08-25, 08:22:51 UTC]	{logging_mixin.py:109}	INFO	source: 87	proc replay nofs igout=&catalog;
[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	source: 89	run;
[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	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