

Leveraging APIs in a Multilingual World: Run SAS Jobs from your R Program

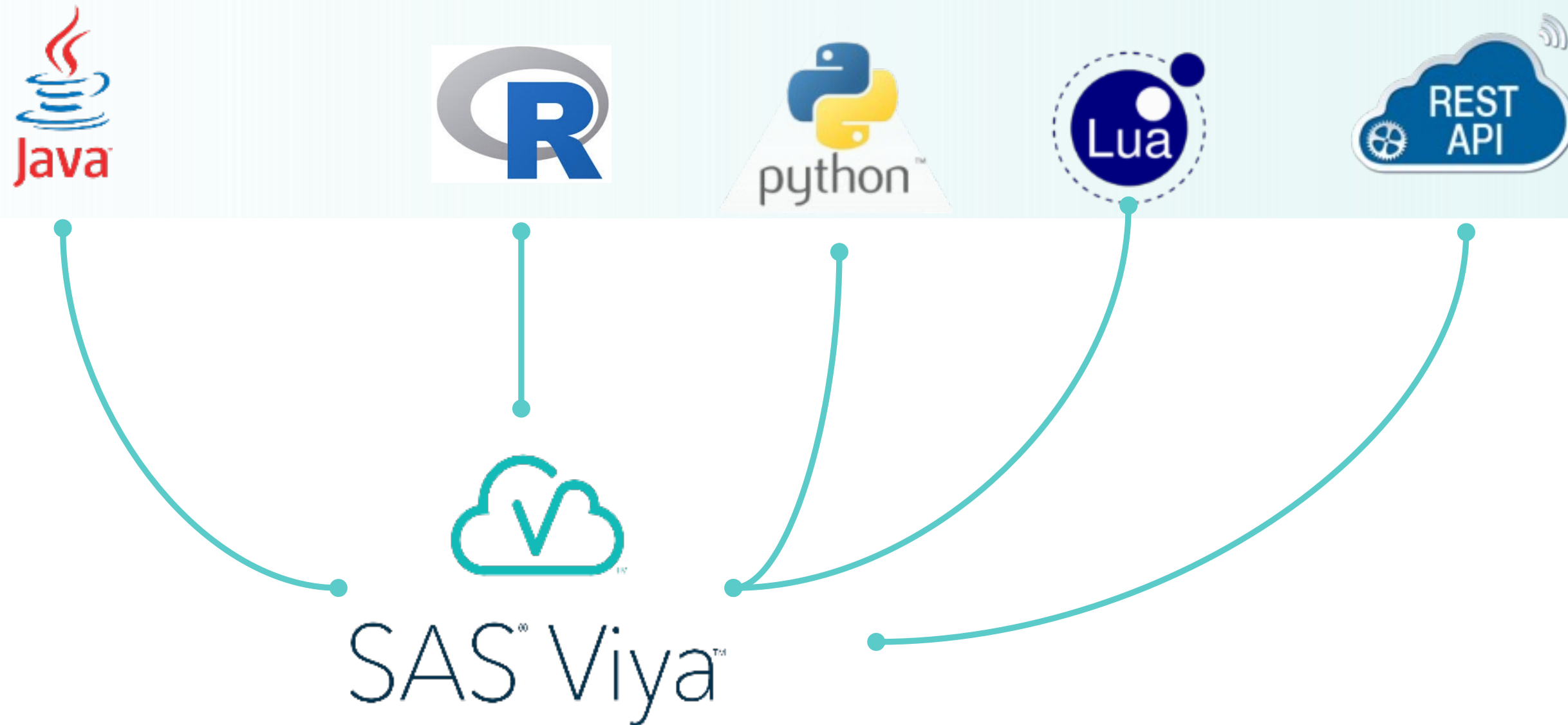
Samiul Haque
Jim Box



Scenario

- A data science team primarily working on R, wants to leverage SAS macro libraries.
- R programmers in a Biostatistics team want to leverage PROC MIXED.
- Trigger SDTM / ADAM automation process from R Shiny App.
- HEOR team using R for cohort analysis wants to seamlessly execute downstream analysis written in SAS.
- ML team wants to automate the cohort building process with data refresh, leveraging existing cohort definition.
- Run SAS Jobs from a quality dashboard

SAS Integration with External tool



API Integration allows **Open Source** to invoke SAS analytics actions

What is REST API?

- API: Application Programming Interface.
Mechanism (and set of rules) for communication between different software components and programming languages
- REST: Representation State Transfer
- An architectural style / guiding principle for developing API. E.g.,:
- Uniform Interface
- Client- Server Design
- Stateless

Why REST API ?

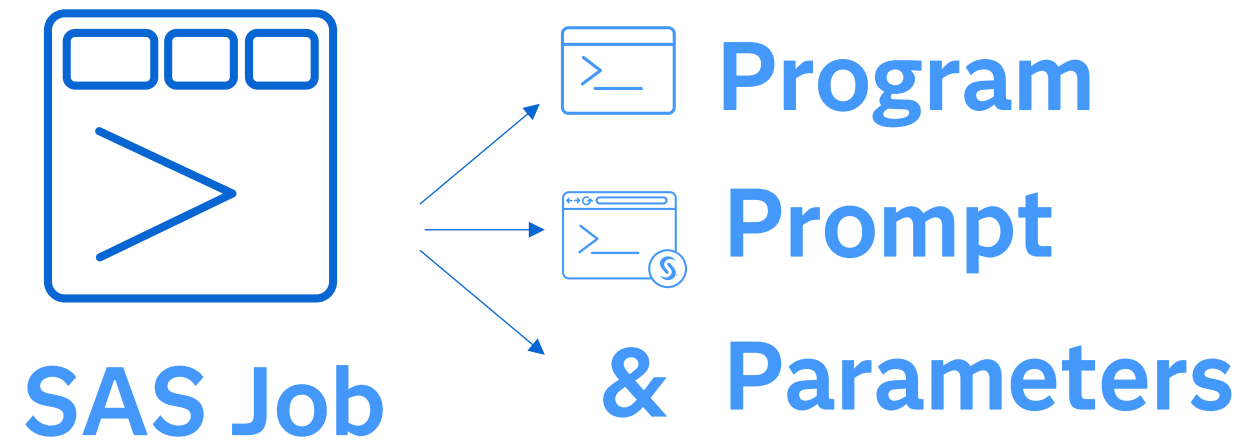
- "SAS Viya REST APIs are based on resource-oriented URLs, use HTTP authentication and HTTP verbs, and return HTTP response codes to indicate API errors. With SAS Viya REST APIs, you can create and access SAS resources using any client technology."

[-SAS Viya REST API Developers Guide](#)

Why REST API ?

- Create and access SAS resources from any client software / platform.
- Automation and efficiency, build business processes around SAS resources.
- Code libraries that can leverage different programming languages.
- Build custom solution on top of Viya
- Use REST APIs of external software to ingest data or decision
Example: EDC data ingested through REST APIs.

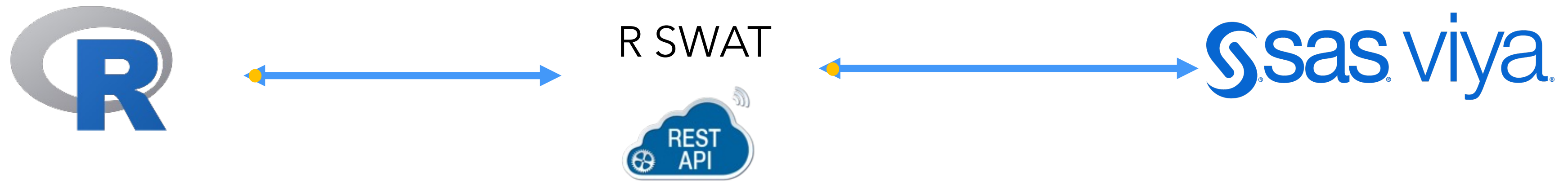
What Is SAS Job?



- All Job have a Job Submit URI
- Can be submitted through REST APIs
- Parameters can be passed as macro variables
- Can be scheduled

SAS Job is executed using the SAS Job Execution Web Application. It is a web-based client used to create, manage and execute jobs.

Executing SAS Job from R



What Is SAS Job?

Run | Show Log

Program Preview

```
1 cas;
2 caslib _ALL_ assign;
3 proc sort data=sashelp.cars out=tmp;
4   by &byvar.;
5 run;
6
7 proc means data=tmp;
8   var msrp;
9   by &byvar.;
10  output out=result;
11 run;
12
13 *promote result and CARS tables to PUBLIC CASLIB;
14
15 proc casutil;
16   droptable casdata='msrp_means' incaslib='PUBLIC' quiet;
17   droptable casdata='CARS' incaslib='PUBLIC' quiet;
18   droptable casdata='msrp_VA' incaslib='PUBLIC' quiet;
19   *promote makes the scope of tables available across all authenticated sessions;
20   load data=work.result casout='msrp_means' outcaslib='PUBLIC' promote;
21   load data=sashelp.cars casout='cars' outcaslib='PUBLIC' promote;
22 run;
23
24 data public.msrp_VA(promote=yes);
25   set public.msrp_means;
26   byvar="&byvar.";
27   byvarvalue=&byvar.;
28   keep byvar byvarvalue _STAT_ MSRP;
29 run;
30
31 *job output as JSON - to be leveraged by API Calls;
32
33 proc json out=_webout nosastags pretty;
34   export result;
35 run;
```

>>

Parameters

+ New parameter

Default value:

execute

Required: ☒

[Remove](#)

Name:

_OUTPUT_TYPE

Field type:

Character

Default value:

json

Required: ☒

[Remove](#)

Name:

byvar

Field type:

Character

Default value:

origin

Required: ☒

[Remove](#)

Job Parameters

What Is SAS Job?

▶ Run |   | ↶ ↷ |  Show Log

Program Preview

```
1 cas;  
2 caslib _ALL_ assign;  
3 ⊖ proc sort data=sashelp.cars out=tmp;  
4   by &byvar.;  
5 run;  
6  
7 ⊖ proc means data=tmp;  
8   var msrp;  
9   by &byvar.;  
10  output out=result;  
11 run;  
12  
13 *promote result and CARS tables to PUBLIC CASLIB;  
14  
15 ⊖ proc casutil;  
16   droptable casdata='msrp_means' incaslib='PUBLIC' quiet;  
17   droptable casdata='CARS' incaslib='PUBLIC' quiet;  
18   droptable casdata='msrp_VA' incaslib='PUBLIC' quiet;  
19   *promote makes the scope of tables available across all authenticated sessions;  
20   load data=work.result casout='msrp_means' outcaslib='PUBLIC' promote;  
21   load data=sashelp.cars casout='cars' outcaslib='PUBLIC' promote;  
22 run;  
23  
24 ⊖ data public.msrp_VA(promote=yes);  
25   set public.msrp_means;  
26   byvar="&byvar.";  
27   byvarvalue=&byvar.;  
28   keep byvar byvarvalue _STAT_ MSRP;  
29 run;  
30  
31 *job output as JSON - to be leveraged by API Calls;  
32  
33 ⊖ proc json out=_webout nosastags pretty;  
34   export result;  
35 run;
```

>>

Properties

Name:

job_means

Location:

/Public

File type:

Job definition

Description:

Details

Created:

Jan 25, 2024, 4:04:38 PM

Created by:

sasadm

Modified:

Jan 31, 2024, 11:26:04 AM

Modified by:

sahaqu

Job submit

https://
com/SASJobExecution/?
_program=%2FPublic%2Fjob_means

Job Submit URL

REST API Call from R

```
headers = c(  
  'Authorization' = 'Basic c2FzLmNsaTo=',  
  'Content-Type' = 'application/x-www-form-urlencoded'  
)
```

```
body = list(  
  'grant_type' = 'password',  
  'username' = paste0(uname),  
  'password' = paste0(pass)  
)
```

```
response <- POST(url = "https://sashls2.eastus.cloudapp.azure.com/SASLogon/oauth/token", body = body, add_headers(headers), encode = 'form')
```

```
response_df=fromJSON((content(response, 'text')))
```

```
print(response_df)
```

```
headers = c(  
  'Authorization' = paste0("Bearer ", response_df$access_token, " ") )
```

REST API Call from R

```
headers = c(
  'Authorization' = paste0("Bearer ", response_df$access_token, " ") )

#Run a Pre-Defined Job in SAS Viya using REST API this can also be submitted as Asynchronous call
res <- POST(url = "https://sashls2.eastus.cloudapp.azure.com/SASJobExecution/?_program=%2FPublic%2Fjob_means", add_headers(headers))

result_df=fromJSON((content(res, 'text')))
print(result_df)

res <- POST(url = "https://sashls2.eastus.cloudapp.azure.com/SASJobExecution/?_program=%2FPublic%2Fjob_means&byvar=make", add_headers(headers))
```

Summary

- REST API endpoints enables integration of SAS with any technology
- Web app developers can leverage governance, quality while using latest features available through open-source packages
- Complete documentation is available on developer.sas.com

