

Intelligence Features of SAS Viya Clinical Data Science Platform

From LLM to Agentic AI



LLM Powered Chatbots

Parametric knowledge of models used to respond to queries

RAG Chatbots

Non-parametric knowledge supplemented with user prompts

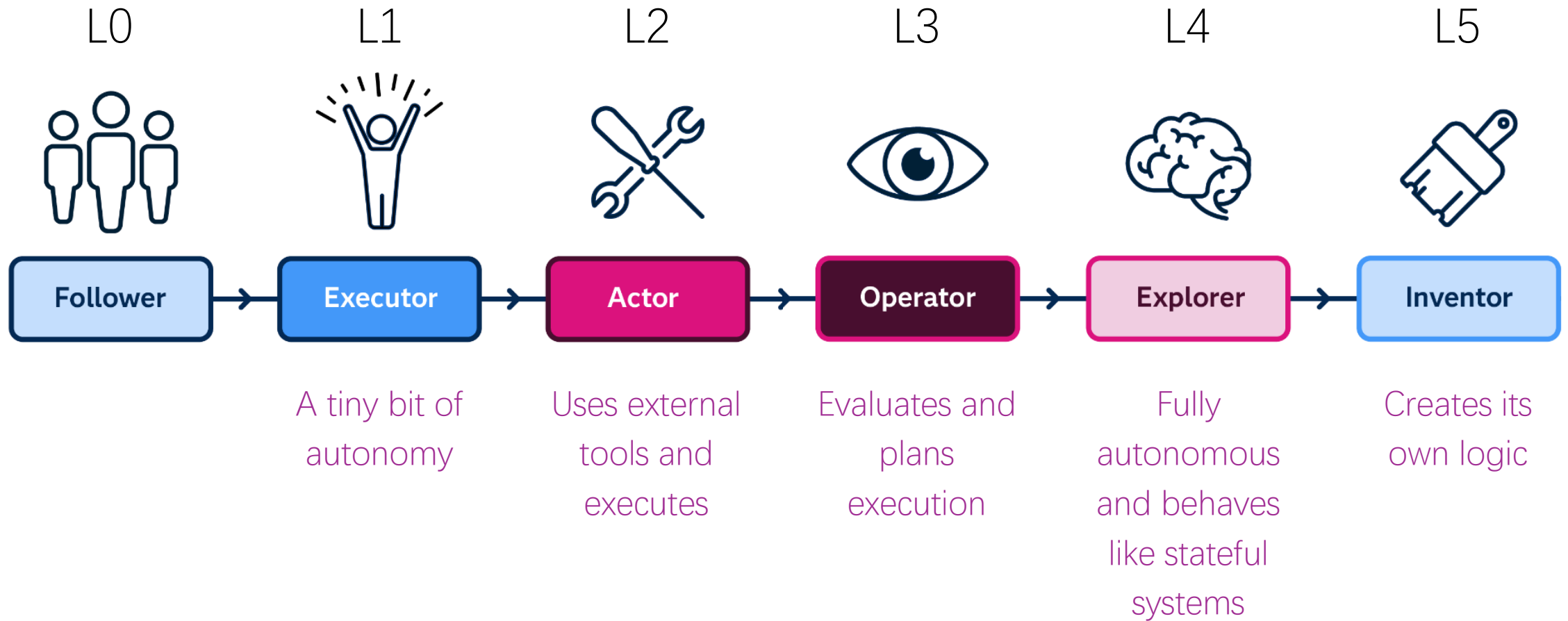
AI Agents

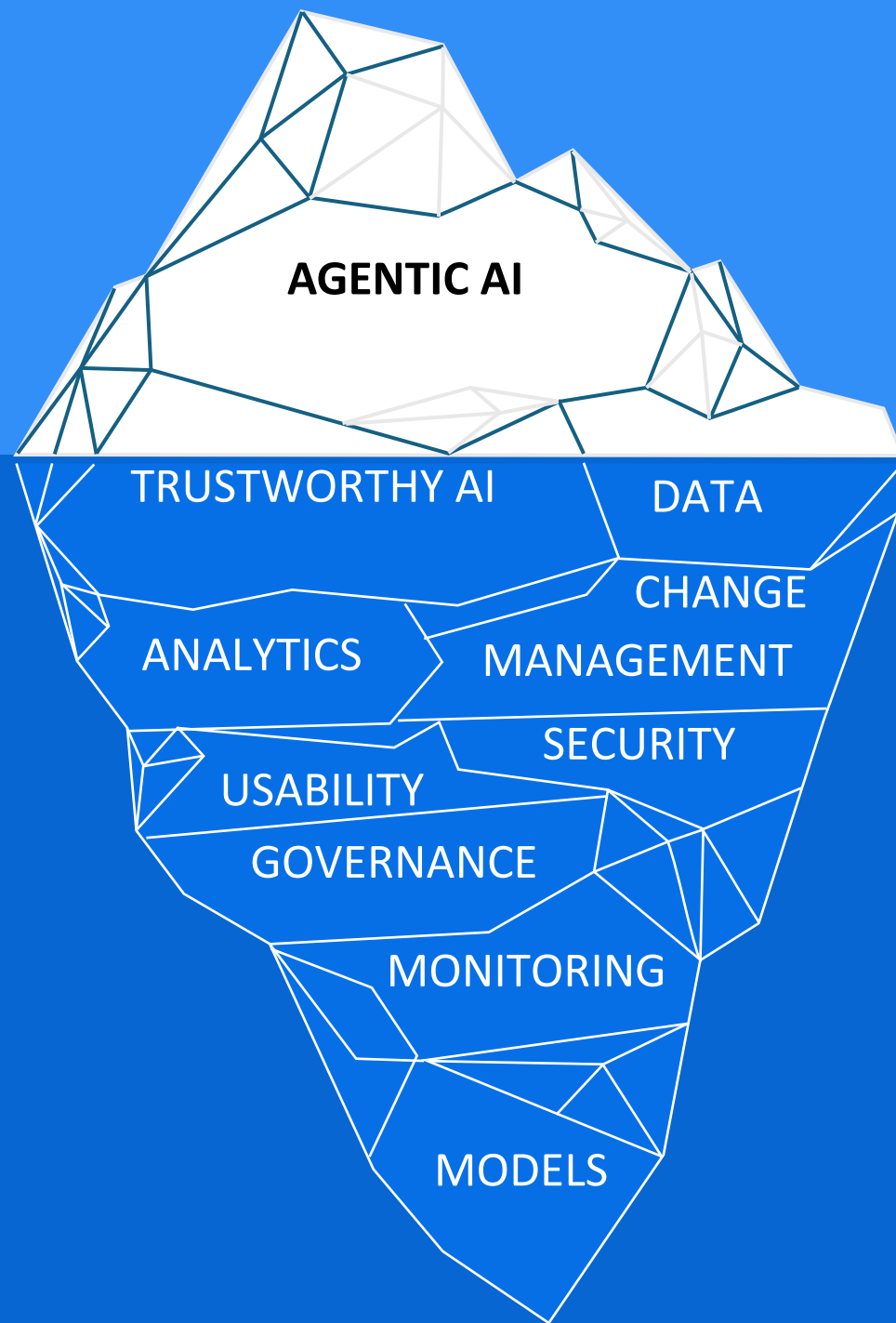
LLMs with tools use capabilities and advanced reasoning and planning capabilities

Agentic Systems

System architecture consisting of multiple tools, AI agents, and components

6 Levels of Agentic AI Behavior





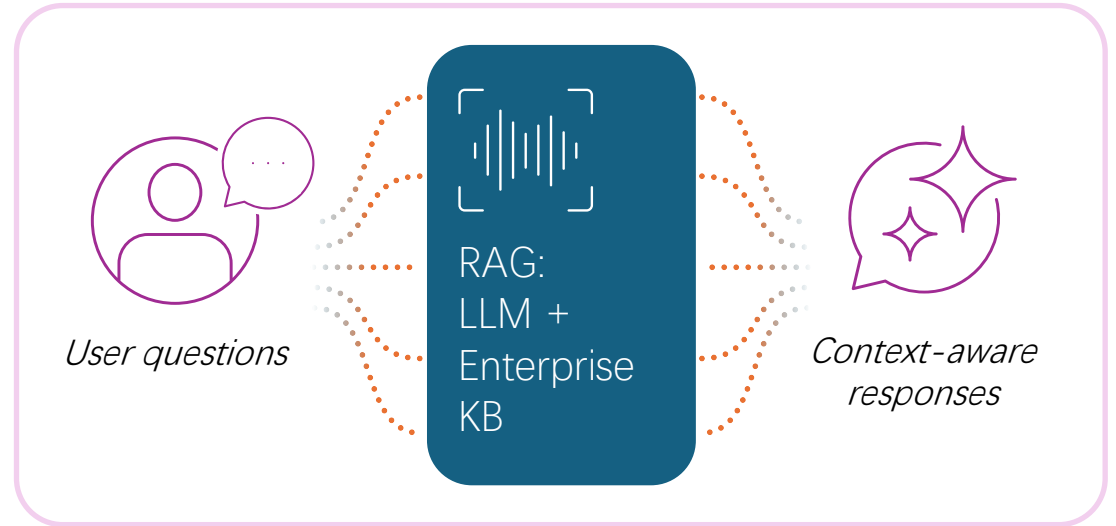
Enter RAG:

The Enterprise data: Knowledge base (1 of 2)

Retrieval-Augmented Generation

= search + generate

so outputs are **grounded** in your enterprise knowledge base



Why RAG matters to business:

Factuality & Trust

Model cites the source;
reduces hallucinations

Freshness

Pulls latest policies/specs
at answer time—no
retraining

Governance

Keeps enterprise access
controls & auditing in the
loop

Flexible

Not locked to a specific
AI model

Enter Agents:

The Enterprise Game-Changer (2 of 2)

AI Agents = reason + act

so systems don't just answer – they can decide and act



Why Agents matter to Business:

Autonomy & Action

Move from answering to acting — trigger workflows, updates, or decisions automatically

Personalization

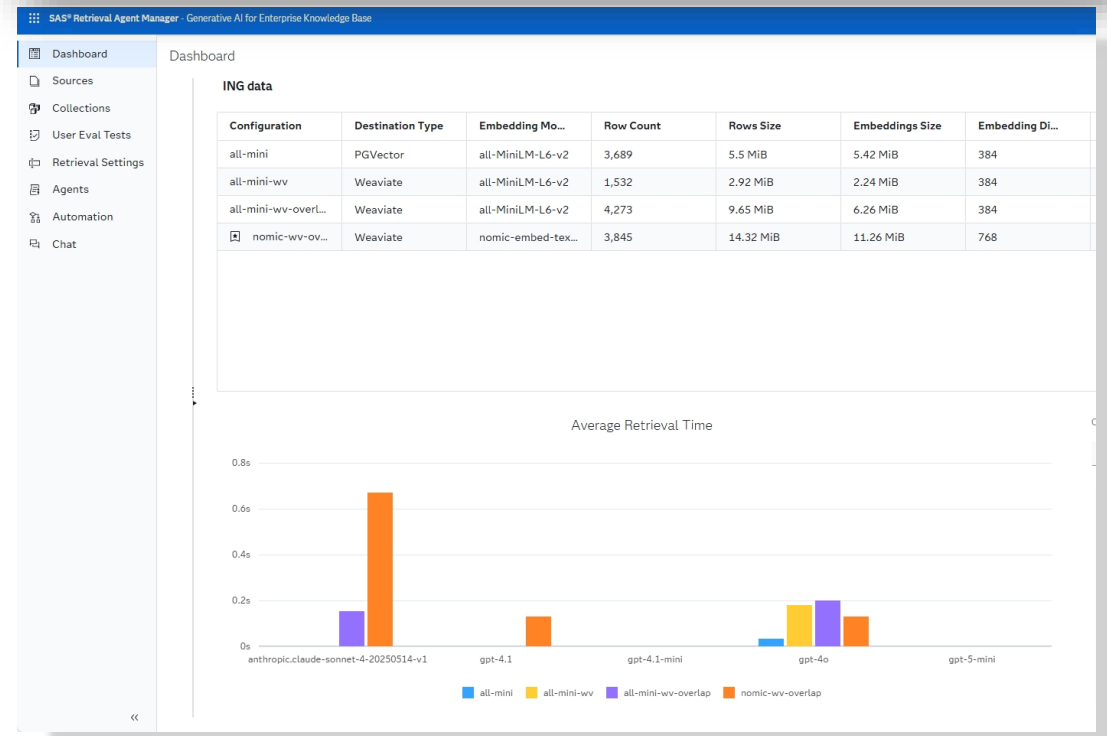
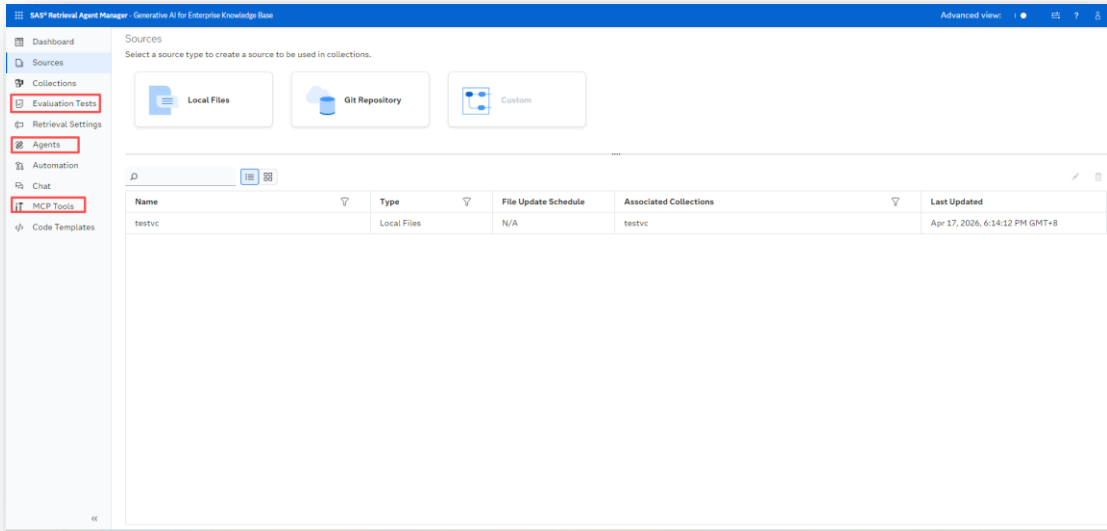
Tailor responses as per Organizational context

Adaptability

Evaluate results, adjust strategies and improve in dynamic environments

Governance

Keeps enterprise access controls & auditing in the loop



SAS Viya RAM

RAG

Agent

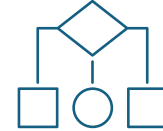
MCP

Key Benefits



Time to value

- No-code Agents with MCP tools
- No-code, Intuitive UX-based RAG



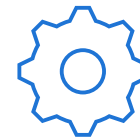
Fit for purpose

- Full support for Chat, APIs, and Agents
- Configurable resource limits
- BYO Gen AI services



Answers You Can Trust

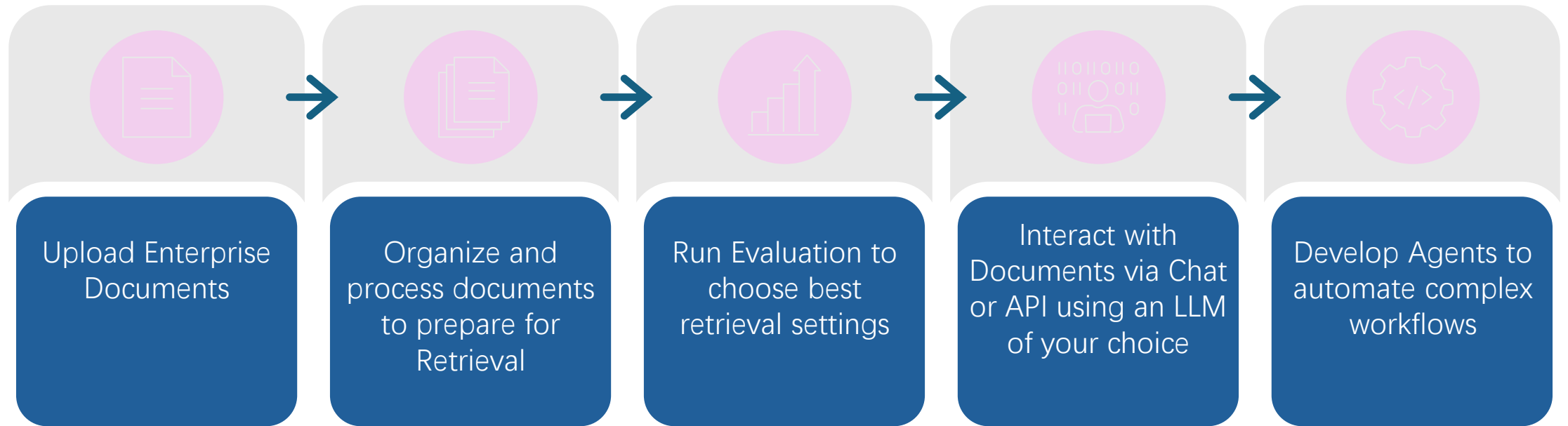
- Human in the loop workflow
- Agent* and RAG Evaluation
- Citations



Operational Resiliency

- Agent Control & Observability
- Automated deployments/updates
- Automation for handling new data

RAM: Workflow



Cloud:

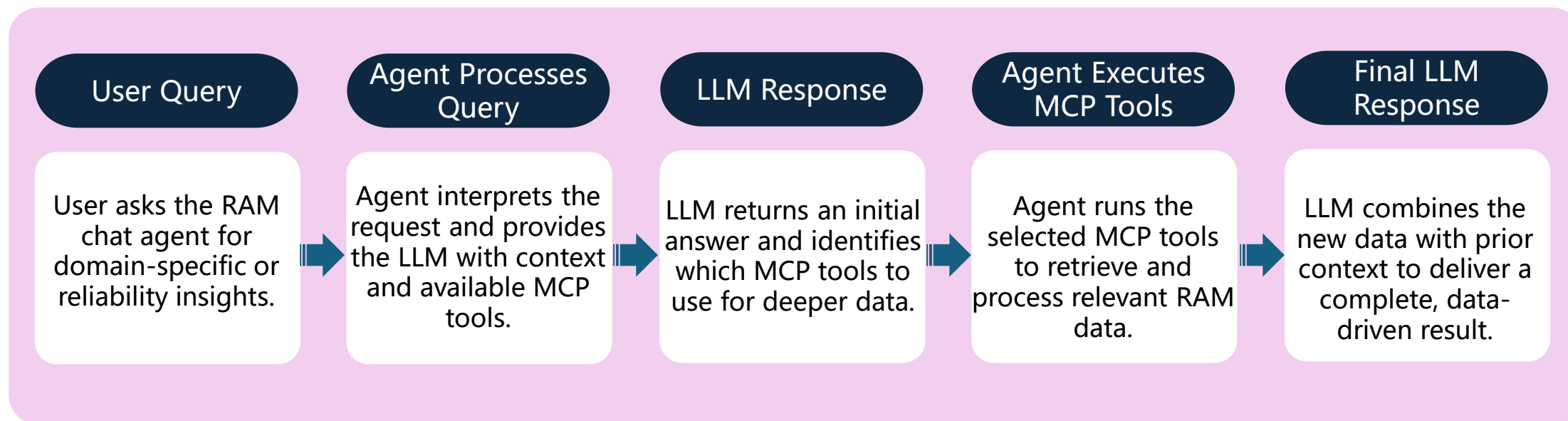


On-prem:

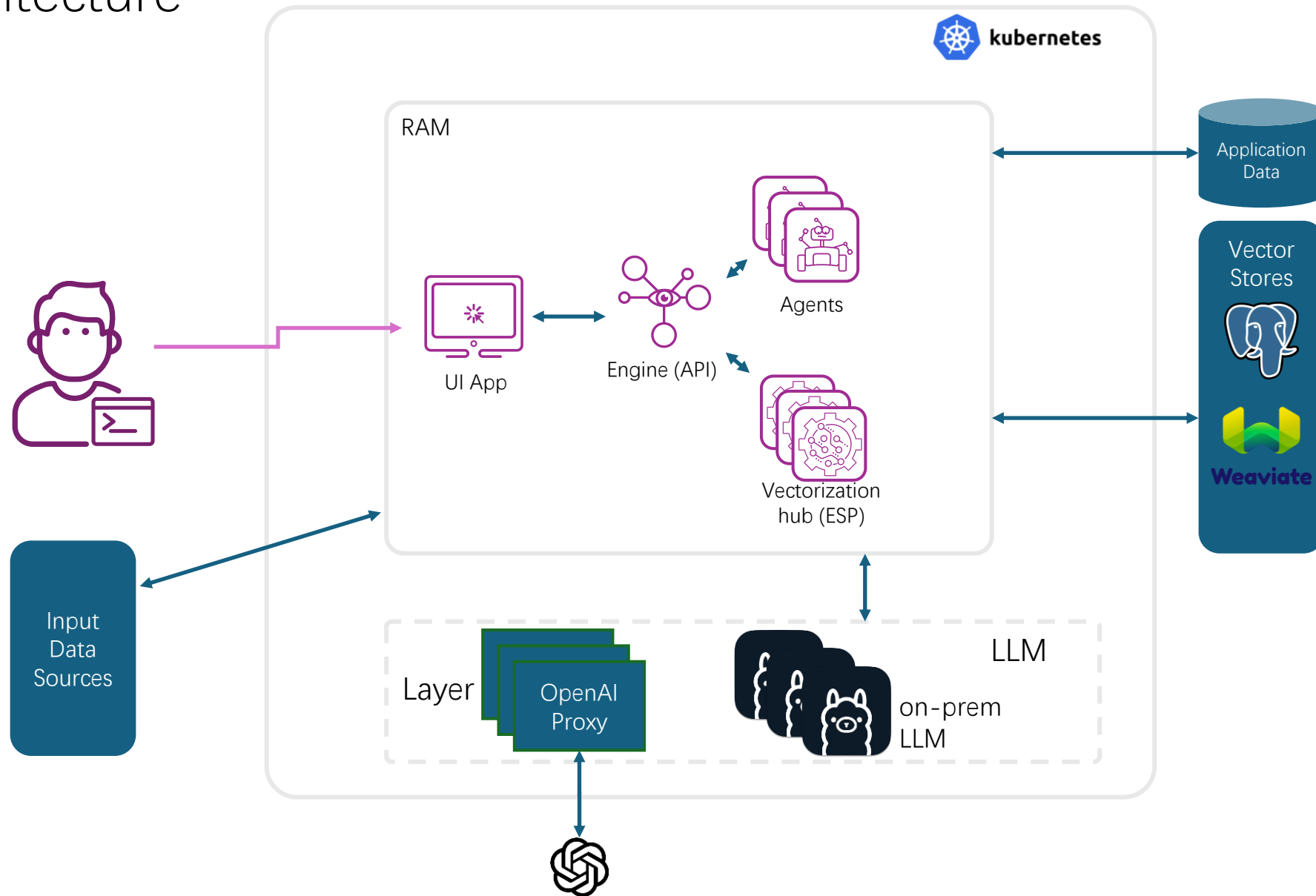


Deployment options

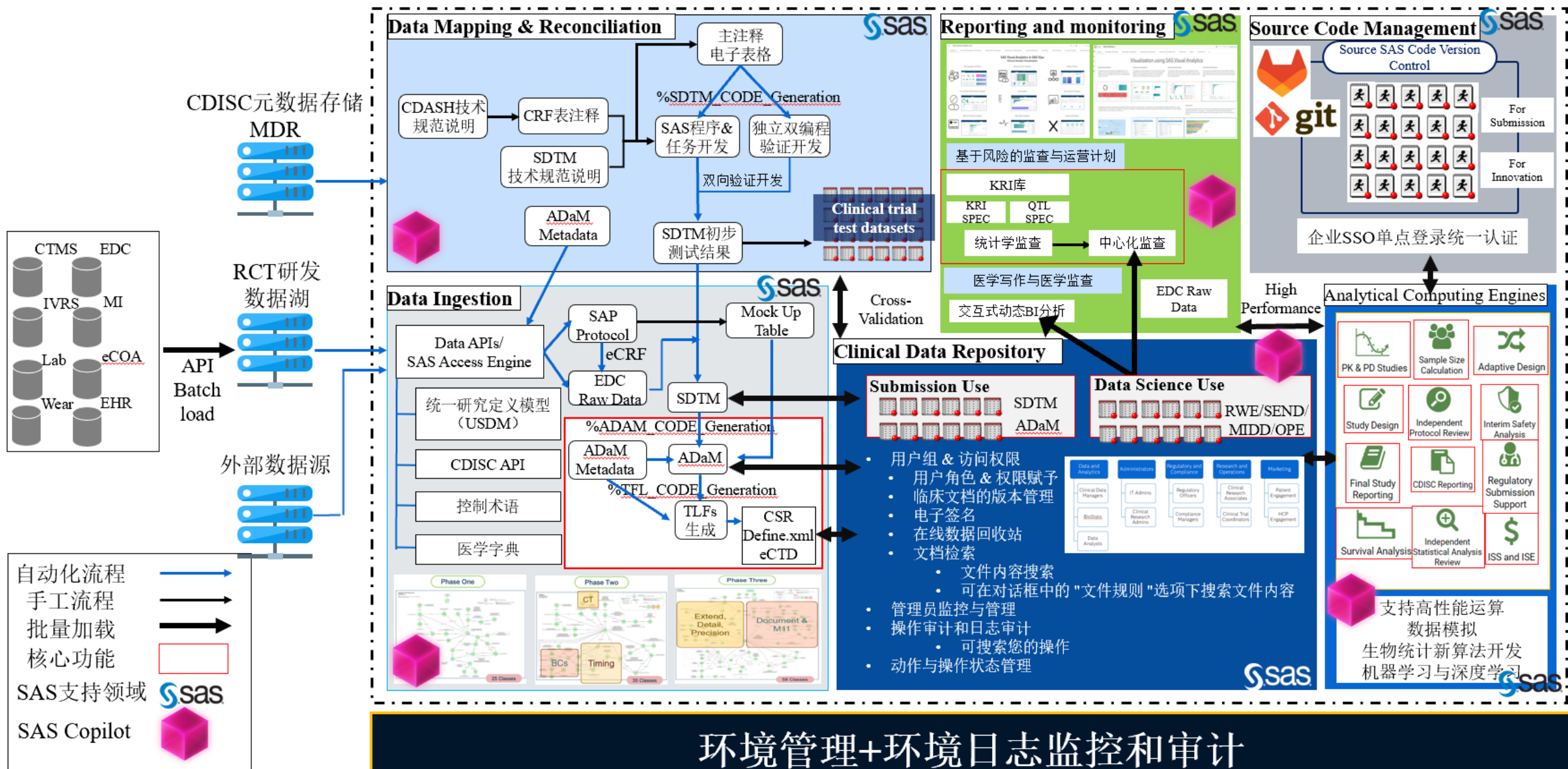
Agent workflow with MCP Integration



RAM: Architecture



现代化临床试验数据科学平台功能架构



User Benefits



Access to any data

Enterprise wide governed data access to leading cloud data platforms, relational and non-relational databases and popular file formats



Powerful data flows

Powerful reliable and transparent data pipelines for a single pane of glass view. ELT/ETL type of processing.



Open-Source Integration

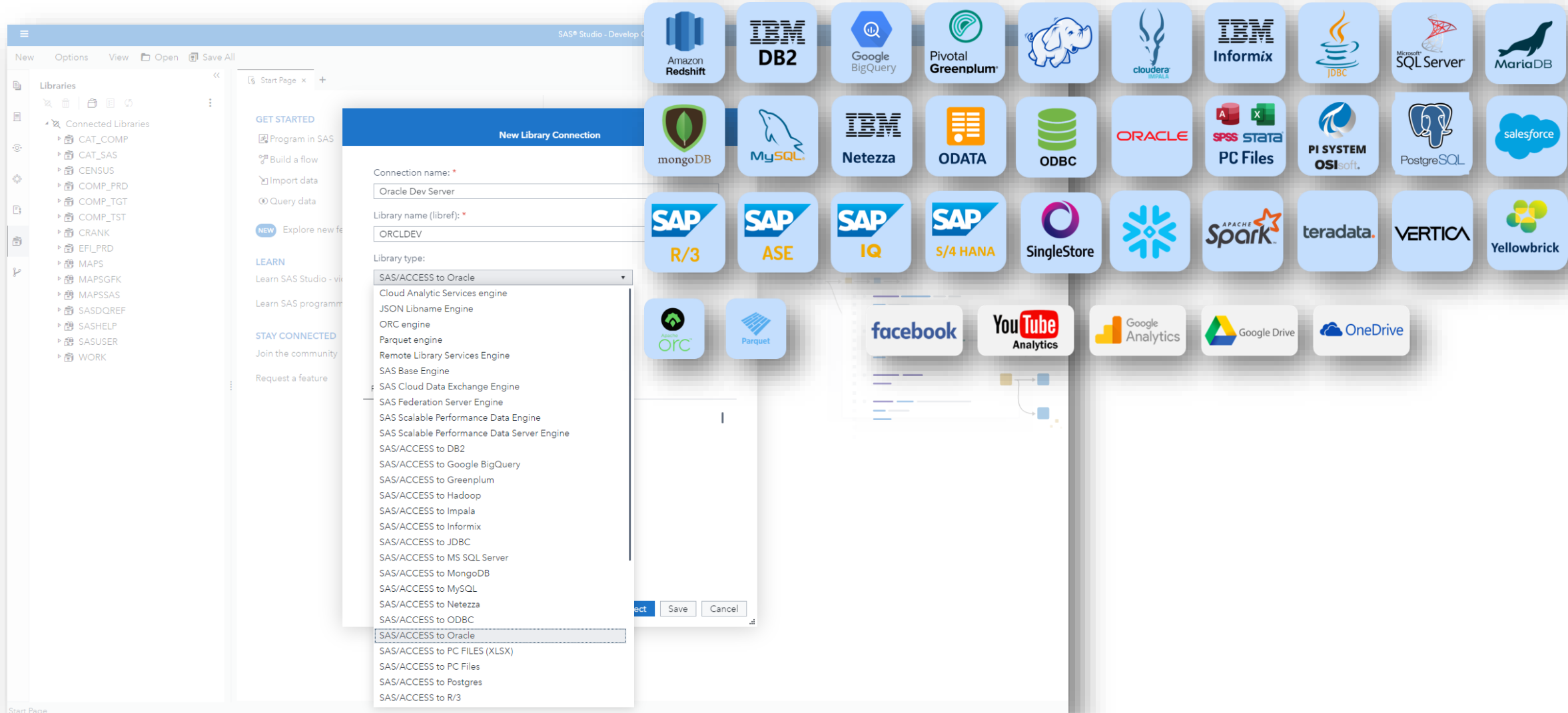
Bring Python code assets in governed enterprise workflows. Wrap up siloed utilities in reusable and shareable visual steps.



Seamless collaboration

Efficiently manage and orchestrate your project assets by using advanced Git integration.

80+ Connectors



Wide collection of Flow steps



- Branch Rows
- Calculate Columns
- Filter Rows
- Insert Rows
- Manage Columns
- Mask Data
- Query
- Rank Data
- Remove Duplicates
- Select Random Sample
- Sort
- Split Columns
- Stack Columns
- Transpose Data
- Union Rows

Data Quality

- Clean Data
- Match Codes
- Parse Data

Examine Data

- Characterize Data
- Describe Missing Data
- List Data
- List Table Attributes



- Bar Chart
- Bar-Line Chart
- Box Plot
- Bubble Map
- Bubble Plot
- Choropleth Map
- Heat Map
- Histogram
- Line Chart
- Pie Chart
- Scatter Map
- Scatter Plot
- Series Map
- Series Plot
- Text Map



- Canonical Correlation
- Cluster Variables
- Coin Toss Simulation
- Combinations
- Compute Similarities and Distances
- Correlation Analysis
- Correspondence Analysis
- Dice Roll Simulation
- Distribution Analysis
- Factor Analysis
- Multidimensional Preference Analysis
- One-Way Frequencies
- Permutations
- Poker Hand Probability
- Same Birthday Probability
- Summary Statistics
- t Tests
- Table Analysis

Machine Learning

- Fast k-Nearest Neighbors
- Moving Window Principal Component Analysis

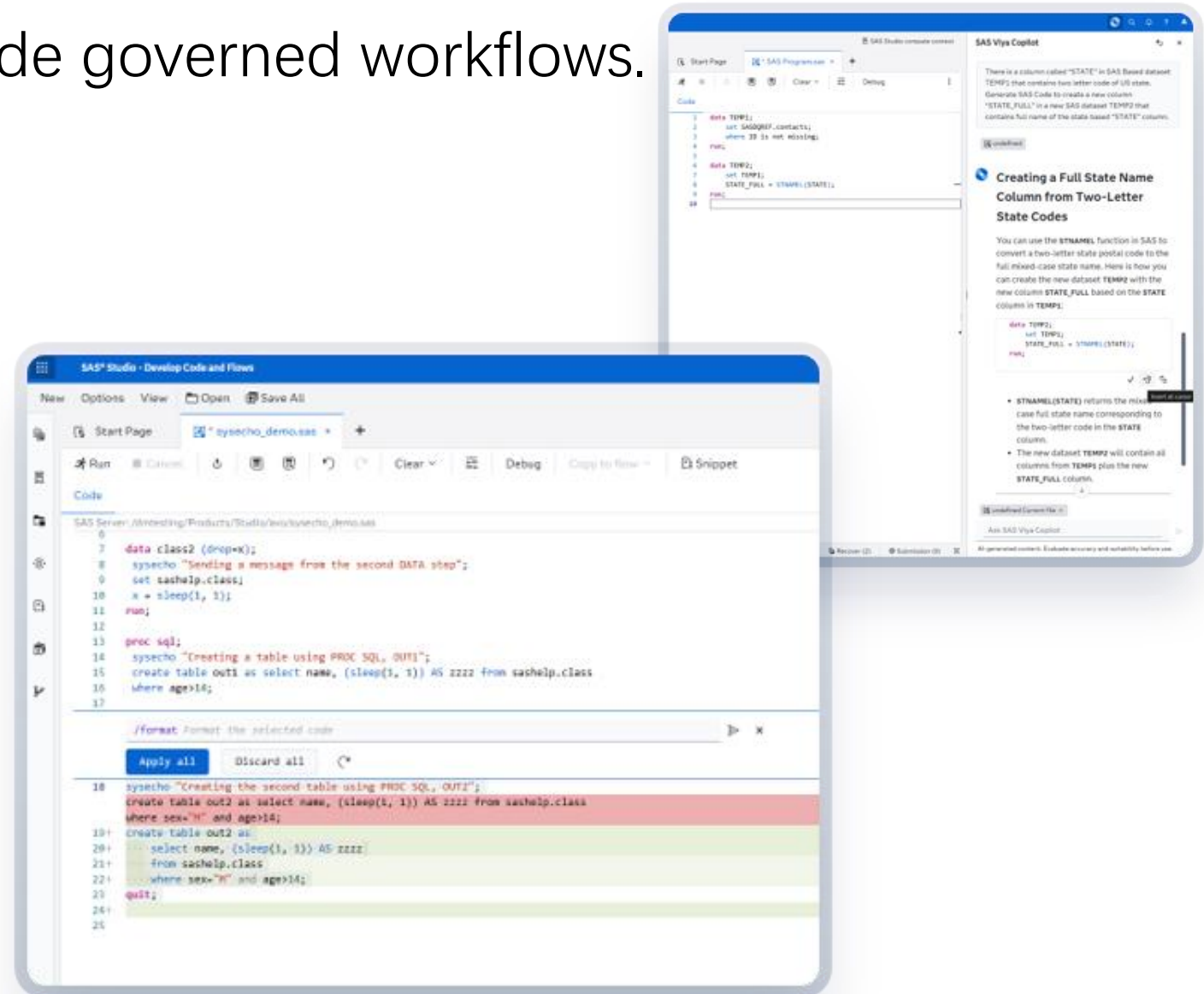
Manage Models

- Register Python Model
- Register SAS Model

SAS Viya Copilot in SAS Studio

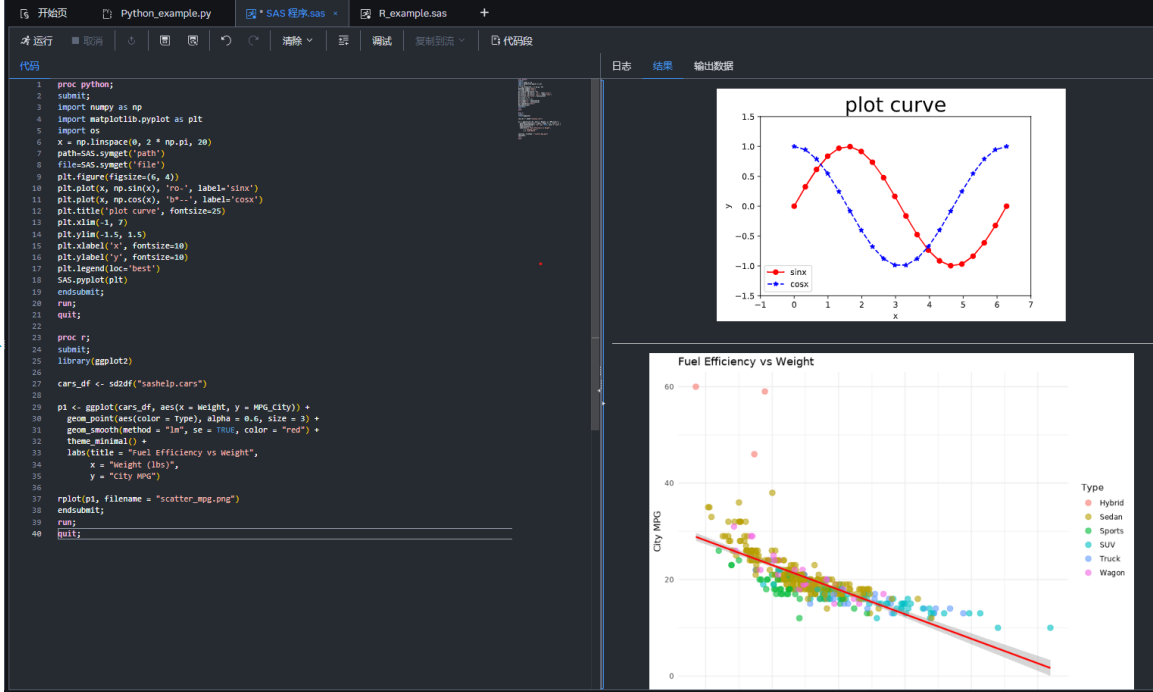
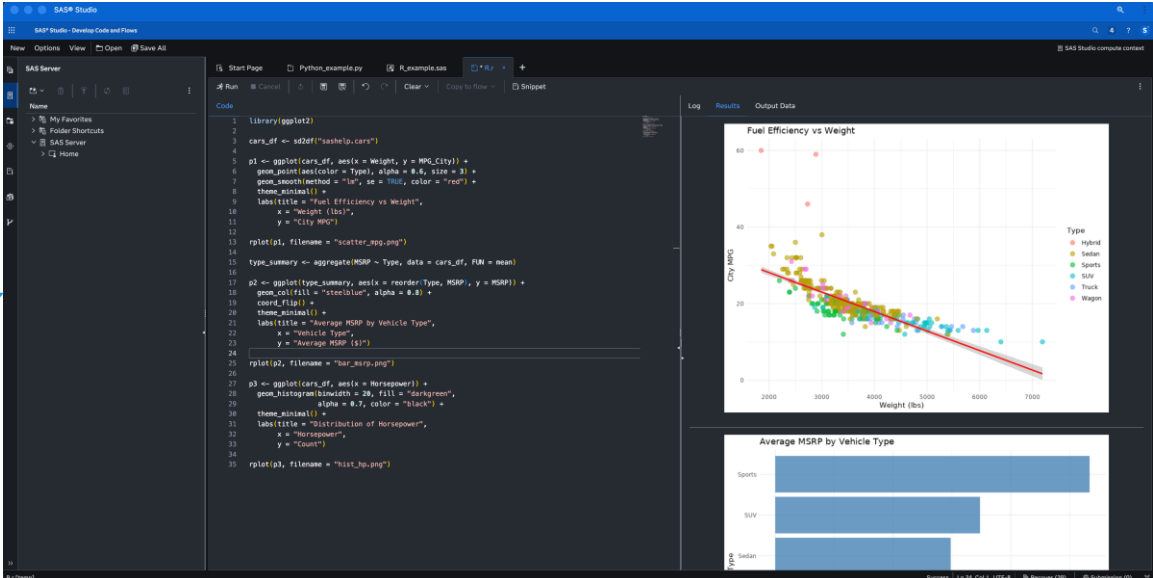
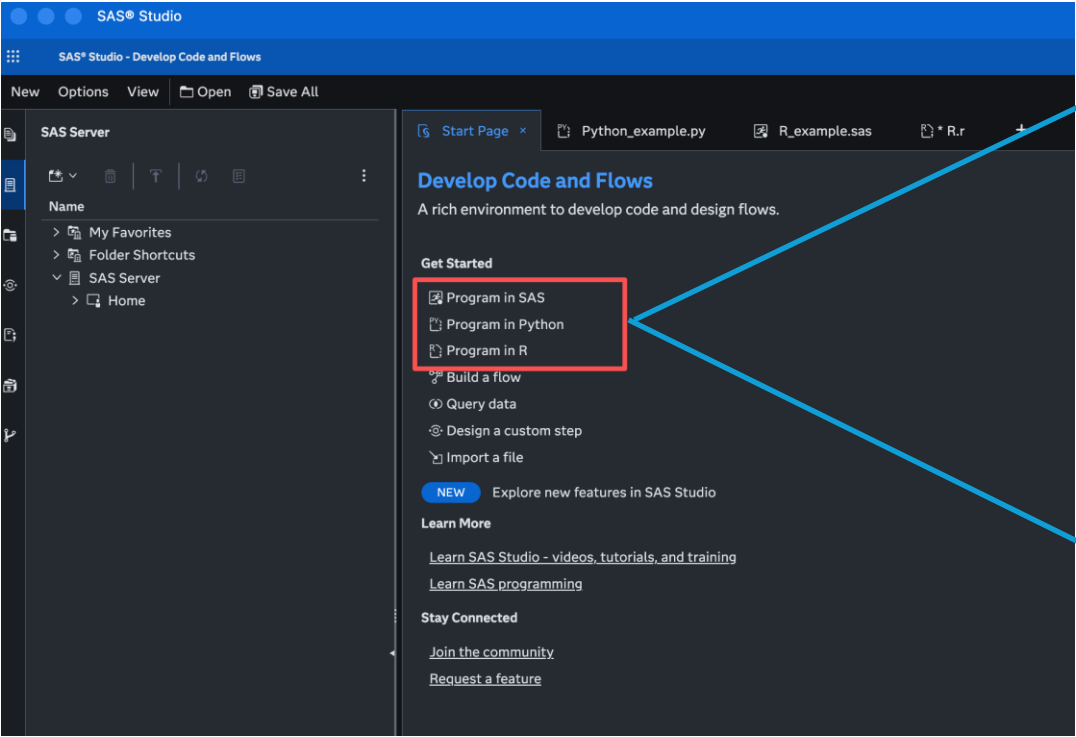
Embedding productivity inside governed workflows.

- **Context-aware by design—not generic code generation:** Understands the active program, flow, and session context to provide relevant, reviewable suggestions
- **Built for enterprise governance from the start:** Operates within SAS Viya security, permissions, and policy controls—no bypassing guardrails
- **Accelerates everyday developer workflows:** Reduces time spent interpreting legacy code, drafting boilerplate logic, and iterating safely



SAS & Python & R integration

Python & R Code Editor



Custom steps

Design

The screenshot displays the SAS Studio interface, specifically the 'Develop Code and Flows' workspace. The left pane shows the 'Control Library' with various UI components categorized under 'COMMON' and 'DATA'. The main workspace is divided into two panels: 'Design' and 'Execution'.

Design Panel (Left):

- Control Library:** Lists components like Check Box, Color Picker, Date and Time Picker, Drop-down List, File or Folder Selector, Link, List, Numeric Stepper, Radio Button Group, Section, Text, Text Area, Text or Numeric Field, Column Selector, Input Table, New Column, and Output Table.
- Definition Tab:** Contains the following fields:
 - Table that contains the text that needs translating:** A dropdown menu.
 - Table that will contain the translated text:** A dropdown menu.
 - Enter your DeepL API token:** A text field with the placeholder 'Token or macro variable storing the token'.
 - Select your DeepL subscription type:** A dropdown menu set to 'DeepL API Free'.
 - The DeepL API requires language codes to be provided as ISO 639-1 codes, e.g. DE is German.** (Instructional text)
 - Select the column containing source language:** A dropdown menu with 'Add a character column'.
 - Select column containing text needing translation:** A dropdown menu with 'Add a character column'.
 - Translated column name:** A text field containing '_tt_text'.
 - Select the language to translate text into:** A dropdown menu set to 'German'.

Execution Panel (Right):

- Flow:** A sequence of steps: 'Reviews from website' (input), 'Filter out recent' (filter), 'Translate Text' (custom step), and 'Load Translations' (output).
- Generated Code:** Shows the code generated for the 'Translate Text' step.
- Submitted Code and Results:** Displays the results of the step execution.
- Translate Text Step Details:** A detailed view of the 'Translate Text' step, showing the same configuration fields as the Design panel, but with the 'Node' tab selected. The 'Node' tab shows the step's configuration in a more compact, execution-oriented format.

The bottom status bar indicates the file path: 'File: /Public/avo/Translate review from website.flw' and the submission count: 'Submission (0)'.

APIs

Open API for Code Generation for Flows and Programs

SAS Studio Development

SAS Studio API

[Endpoints](#) [Details](#) [Authentication](#) [Usage Notes](#) [Error Codes](#)

Root

Gets the API object

Get

The API object contains a set of links for the supported endpoints.

CodeGeneration

Generates code for an item at the given location

Post

Given the location of a SAS program or the Studio flow file, return the code that SAS Studio would run.

Request Samples

Code Samples

Payload

Python ▾

```
1 import requests
2
3 url = "http://example.com/studioDevelopment/code"
4
5 payload = {
6     "reference": {
7         "type": "content",
8         "path": "string",
9         "mediaType": "application/vnd.sas.dataflow"
10    },
11    "initCode": True,
12    "wrapperCode": True
13 }
14 headers = {
15     "accept": "application/vnd.sas.publish.code.generation.result",
16     "authorization": "Bearer <access-token-goes-here>",
17     "content-type": "application/json"
18 }
19
20 response = requests.post(url, json=payload, headers=headers)
21
22 print(response.text)
```

Thank You