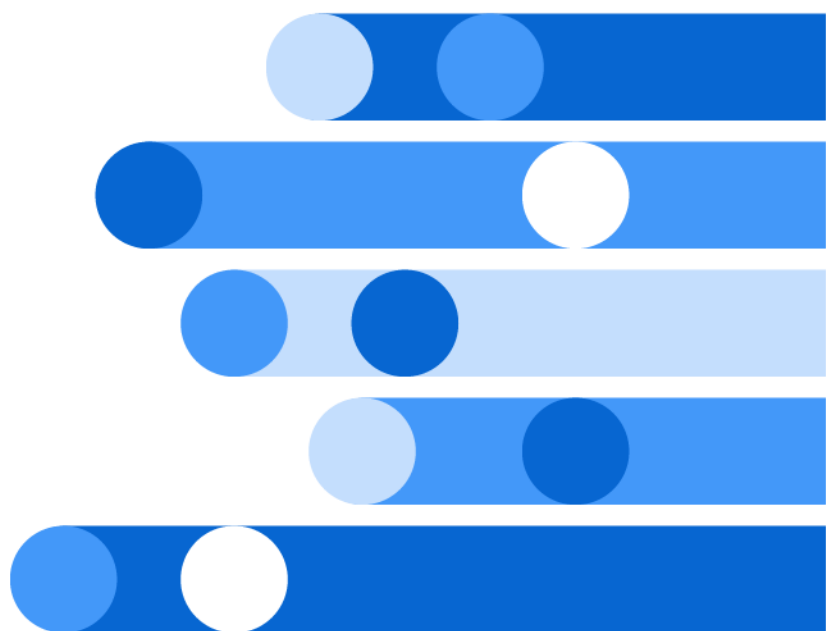




SAS[®] Viya[®] Workbench: Getting Started



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2026. *SAS® Viya® Workbench: Getting Started*. Cary, NC: SAS Institute Inc.

SAS® Viya® Workbench: Getting Started

Copyright © 2026, SAS Institute Inc., Cary, NC, USA

All Rights Reserved. Produced in the United States of America.

For a hard copy book: No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

For a web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

The scanning, uploading, and distribution of this book via the Internet or any other means without the permission of the publisher is illegal and punishable by law. Please purchase only authorized electronic editions and do not participate in or encourage electronic piracy of copyrighted materials. Your support of others' rights is appreciated.

U.S. Government License Rights; Restricted Rights: The Software and its documentation is commercial computer software developed at private expense and is provided with RESTRICTED RIGHTS to the United States Government. Use, duplication, or disclosure of the Software by the United States Government is subject to the license terms of this Agreement pursuant to, as applicable, FAR 12.212, DFAR 227.7202-1(a), DFAR 227.7202-3(a), and DFAR 227.7202-4, and, to the extent required under U.S. federal law, the minimum restricted rights as set out in FAR 52.227-19 (DEC 2007). If FAR 52.227-19 is applicable, this provision serves as notice under clause (c) thereof and no other notice is required to be affixed to the Software or documentation. The Government's rights in Software and documentation shall be only those set forth in this Agreement.

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

July 2026

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

v_003-P1:workbenchgs

Contents

Chapter 1 / Using SAS Viya Workbench	1
About SAS Viya Workbench	1
Open SAS Viya Workbench	2
Create a Workbench	2
Understanding the Status of a Workbench	4
Cloning the Examples Repository	5
Changing the Compute Resources	6
Delete a Workbench	7
Switch Integrated Development Environment	7
Managing Storage	8
Stop a Workbench	14
Setting the Theme	15
Accessing the SAS Documentation	15
Chapter 2 / Working with Git and SSH	17
Using Git from a Workbench	17
Cloning a Public GitHub Repository without Setup	17
Using SSH in Workbench	18
Configuring Global Git Settings	19
Message: "Fatal: Detected Dubious Ownership"	19
Pushing Files to a Remote Git Repository	20
Chapter 3 / Working in Visual Studio Code	21
Open a Workbench in Visual Studio Code	21
Accessing SAS Viya Workbench in Visual Studio Code	22
Run a Python Example	24
Run an Example SAS Program	25
Access SAS Libraries	28
Opening a Terminal in Visual Studio Code	29
Select a SAS Color Theme	30
Create a SAS Notebook	30
Using the AutoExec Feature in Visual Studio Code	32
Chapter 4 / Working in Jupyter Notebook	33
Open a Workbench in Jupyter Notebook	33
Accessing Your Workbench in Jupyter Notebook	34
Opening a Terminal in Jupyter Notebook	35
Run a Python Example	35
Upload Data in Jupyter Notebook	36
Chapter 5 / Working in Jupyter Lab	37
Open a Workbench in Jupyter Lab	37
Accessing Your Workbench in Jupyter Lab	38
Run a Python Example	39

Upload Data in Jupyter Lab	40
Opening a Terminal in Jupyter Lab	41
Chapter 6 / Working in SAS Enterprise Guide	43
Open a Workbench in SAS Enterprise Guide	43
Accessing Your Workbench in SAS Enterprise Guide	44
Chapter 7 / Working with SAS Jobs	47
What Are SAS Jobs?	47
Jobs Page in SAS Viya Workbench	48
Create a Job Definition	48
Submit a Job	50
sas-workbench-execute Command	51
Troubleshooting: SAS Jobs	55
Appendix 1 / Security Considerations	57
Uploading and Downloading Data	57
Storing User Credentials	57
Appendix 2 / Troubleshooting Tips and Frequently Asked Questions	59
What Are the Differences between SAS Viya Workbench and SAS Viya Platform?	59
Does SAS Viya Workbench Support SASPy?	60
What Happens if the Workbench Exceeds the Allocated Quota Storage?	60
How to Reconnect after SAS Viya Workbench Times Out	61
Troubleshooting: Enabling Third-Party Cookies for SAS Viya Workbench	62
Troubleshooting: Possible Error When Using Both the SWAT and SASVIYA Packages	64
Appendix 3 / Third-Party Licenses and Information	65
Overview	65
Jupyter	66
Jupyter Lab	66
OpenVSCoDe Server	67
Matplotlib	68
scikit-learn	68
seaborn	69
Teradata	70

Using SAS Viya Workbench

<i>About SAS Viya Workbench</i>	1
<i>Open SAS Viya Workbench</i>	2
<i>Create a Workbench</i>	2
<i>Understanding the Status of a Workbench</i>	4
<i>Cloning the Examples Repository</i>	5
<i>Changing the Compute Resources</i>	6
<i>Delete a Workbench</i>	7
<i>Switch Integrated Development Environment</i>	7
<i>Managing Storage</i>	8
Understanding Storage in SAS Viya Workbench	8
Create a Volume	9
Using a Home Volume	10
Manage Volumes	12
Network File System (NFS) Storage	12
Create a Snapshot	13
Create a Volume from a Snapshot	14
<i>Stop a Workbench</i>	14
<i>Setting the Theme</i>	15
<i>Accessing the SAS Documentation</i>	15

About SAS Viya Workbench

SAS Viya Workbench is a cloud-based application that is designed for developers and data scientists. Users can access SAS Viya Workbench through a browser on the desktop.

With SAS Viya Workbench, you can run SAS 9 code and SAS Viya analytics within the same environment, without having to install either SAS 9 or SAS Viya. The multi-language architecture enables you to natively run SAS Analytics in other

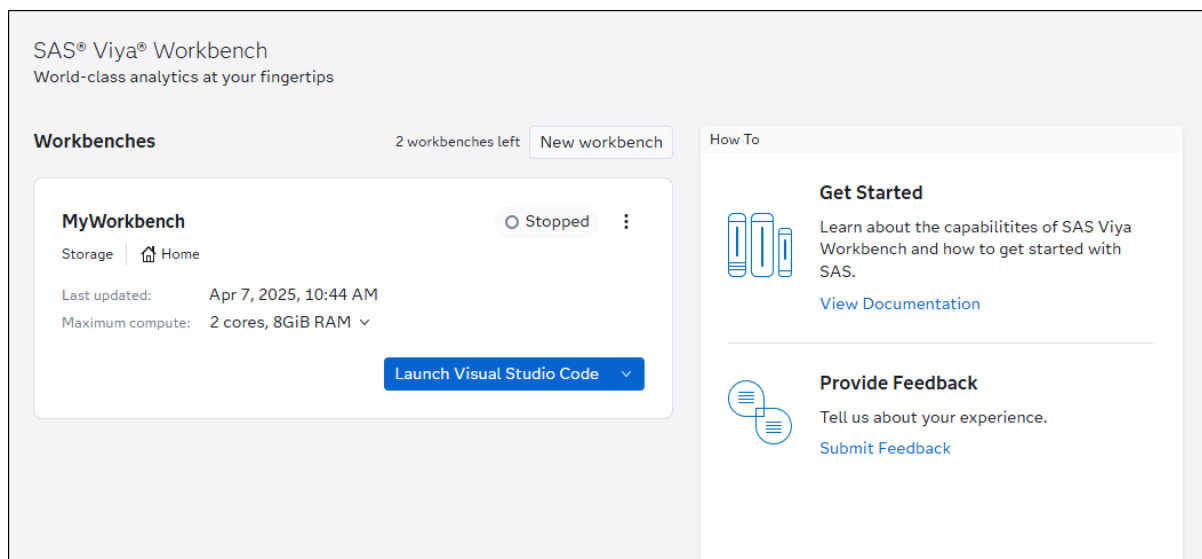
programming languages such as Python and R. SAS Viya Workbench is for SAS programmers who want to access SAS in VS Code and for Python or R programmers who use VS Code or Jupyter. For a list of SAS procedures supported in SAS Viya Workbench, see [SAS Viya Workbench Procedures by Name and Capability](#).

In SAS Viya Workbench, users can manage their own environment without additional IT support. Users can also choose the compute power to match the needs of the project.

Open SAS Viya Workbench

To open SAS Viya Workbench, check your email for login instructions.

From the home page, you can create new workbenches and manage existing workbenches.



Create a Workbench

- 1 Click **New workbench**.
- 2 Enter the workbench name.
- 3 From the **Maximum compute** drop-down list, select the number of cores to use. You can choose from 2, 4, 8, or 16 cores.

Note: If your SAS Viya Workbench administrator enabled GPU support, a GPU option is also available.

- 4 From the **Primary volume** drop-down list, select the storage location. To create a volume, see [“Create a Volume” on page 9](#).

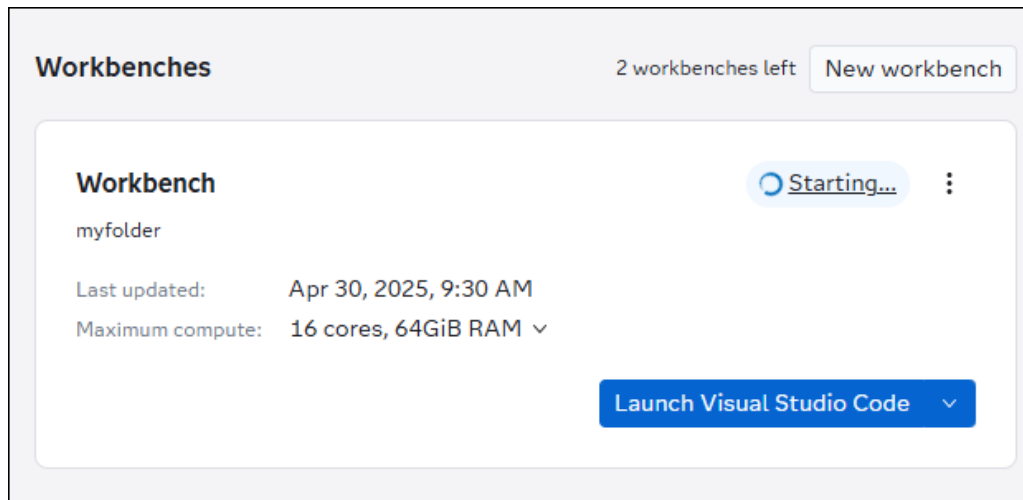
TIP The default volume is called myfolder.

- 5 Enter the path for the mounting folder for the workbench. This folder is automatically created in the `/workspaces/` folder.
- 6 Add any additional volumes that you want associated with this workbench. You can also define a home volume.
- 7 To use environment variables to initialize your workbench session, click **New**. In the Add Environment Variable window, enter the name of the environment variable and the value. Then click **OK**.

Examples of environment variables that you might want to set include SAS_LOG_CFG_PATH and SAS_LOG_OUTPUT_DIR.

- 8 Click **OK**.

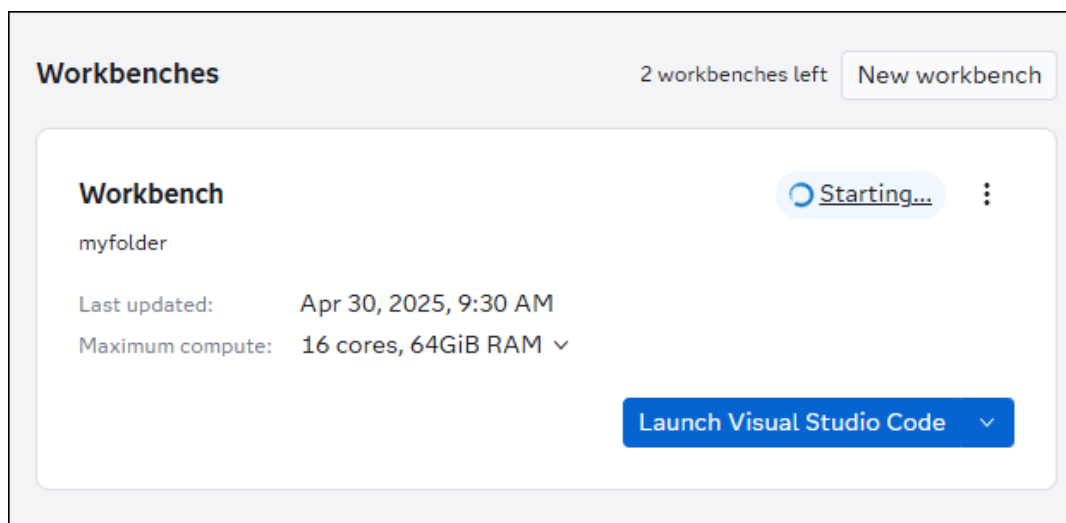
The new workbench is now available from the home page.



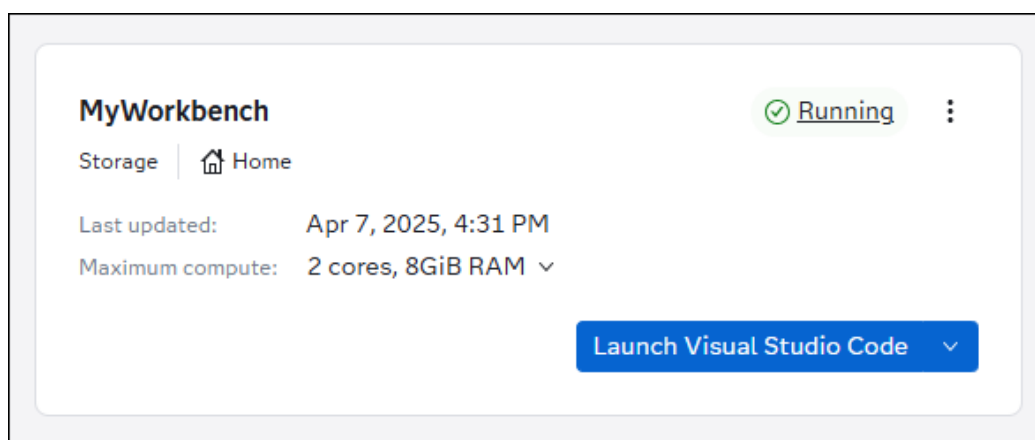
Understanding the Status of a Workbench

From the SAS Viya Workbench home page, you can view the status of any existing workbenches.

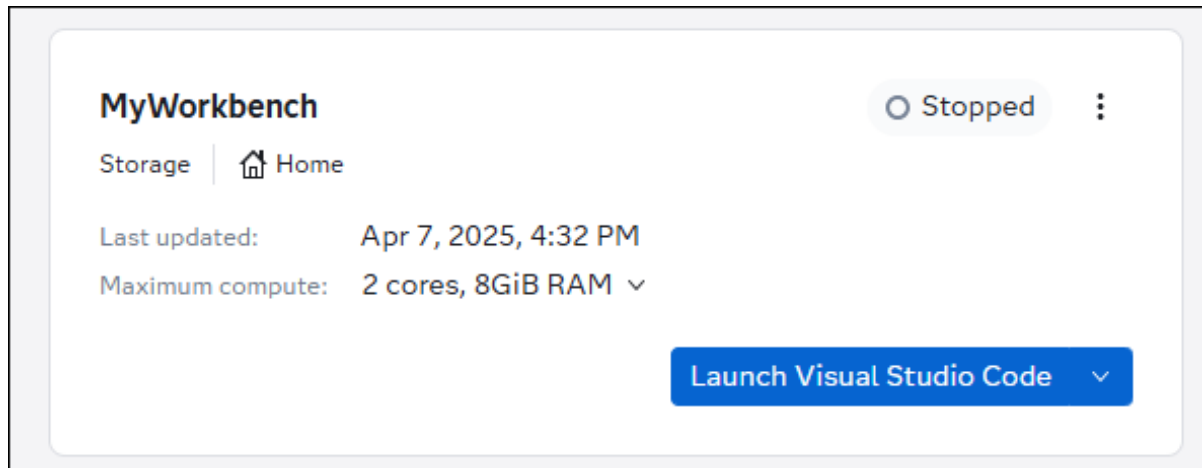
- Starting - This status appears after you click **Start** or after you open the workbench in your IDE.



- Running - The workbench is running.



- Stopped - The workbench is stopped. To stop a workbench, click  and select **Stop workbench**.



From the home page, you can also see whether the workbench has any errors. To get more details about the error, hover your cursor over the Error icon. A brief summary of the error appears in the pop-up. Click **View details in log** for more information.

To view the status of all workbenches, use the Activity Log.

Cloning the Examples Repository

The workbench starts when you open it in the integrated development environment. By default, the workbench is empty.

This documentation assumes that you have cloned the SAS Viya Workbench examples from [GitHub](#).

To clone these examples and make them available in your workbench session:

- 1 Open the workbench in your integrated development environment.
- 2 Open a terminal and enter the following command:

```
git clone https://github.com/sassoftware/sas-viya-workbench-examples.git
```

You do not need to sign in to GitHub to copy the examples to your workbench.

Now, the workbench contains these folders:

- **data** contains the data sources for the examples. For information about each data source, see README.md.
- **python** contains the example Python notebooks. To see the notebooks organized by algorithm, by data set, and by library, see README.md.
- **sas** contains SAS code for the examples. To see the programs organized by SAS procedure and SAS data set, see README.md.

Changing the Compute Resources

When you initially create a workbench, you specify the maximum number of compute resources. This number sets the limits for a workbench.

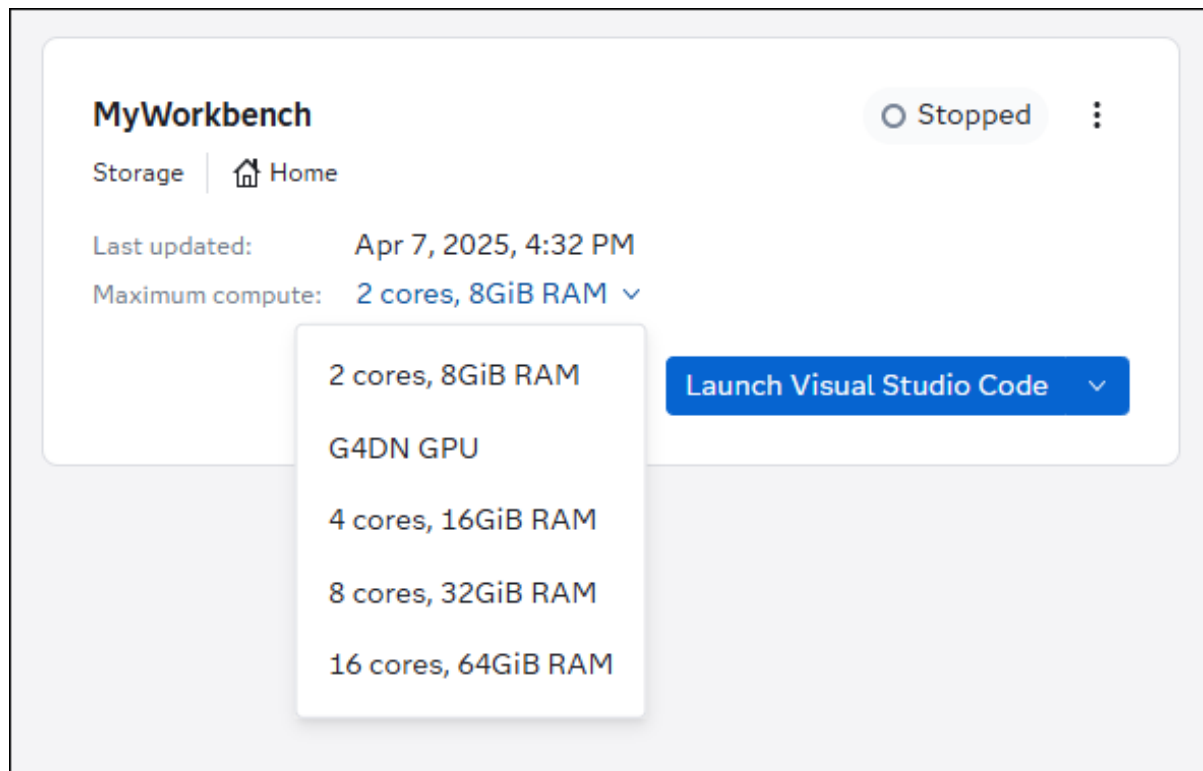
To change the number of compute resources, click  next to the **Maximum compute** field.

You can choose from these options:

- **2 cores, 8GiB RAM**
- **G4DN GPU**

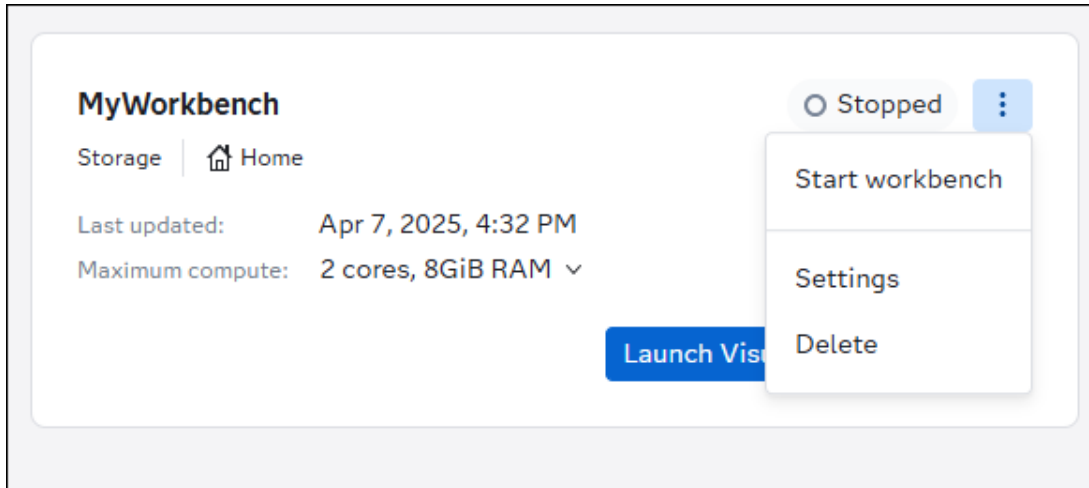
.....
Note: This option is available only if your administrator enabled GPU-support when onboarding SAS Viya Workbench.
.....

- **4 cores, 16GiB RAM**
- **8 cores, 32 GiB RAM**
- **16 Cores, 64 GiB RAM**



Delete a Workbench

- 1 On the home page, click  for the workbench that you want to delete and select **Delete**.



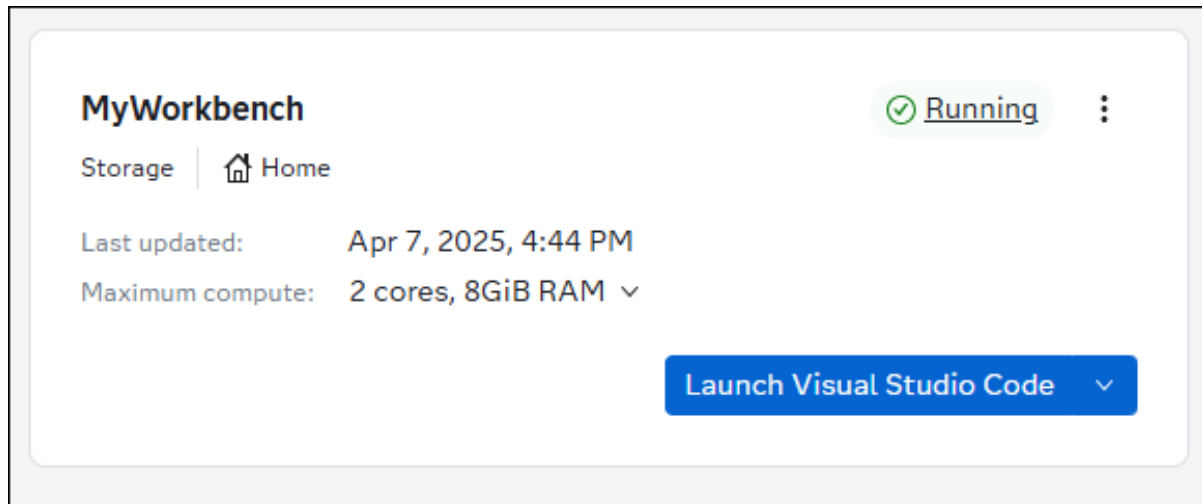
- 2 When prompted, click **Delete**.

The workbench is no longer available from the home page.

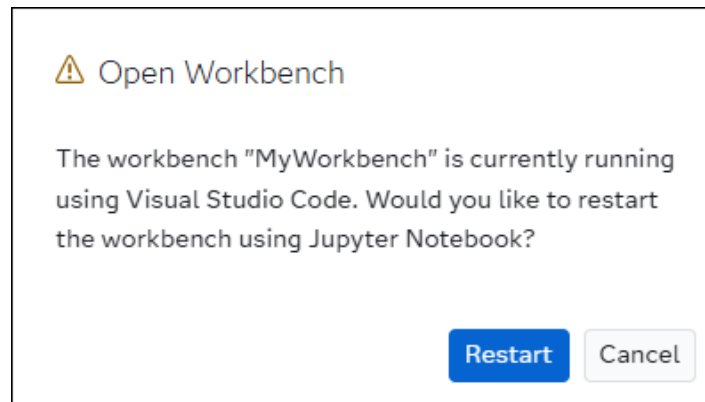
Switch Integrated Development Environment

A workbench can be open in only one development environment at a time.

From the home page of SAS Viya Workbench, you can easily switch development environments for your workbench. In this example, the workbench is currently running in VS Code.



To switch to another development environment, click  and select the new development environment. If the workbench is currently running in another environment, you are prompted to restart the workbench, so it can open in the new environment.



Managing Storage

Understanding Storage in SAS Viya Workbench

A workbench can be associated with one or more volumes. Having multiple volumes associated with a workbench makes it possible for you to easily access data from a variety of sources.

SAS Viya Workbench enables you to create snapshots of volumes. These snapshots enable you to create a backup of a storage location and to create new volumes quickly using existing snapshots. Snapshots enable you to make changes

to the workbench and if necessary, roll back to the previous version of the workbench.

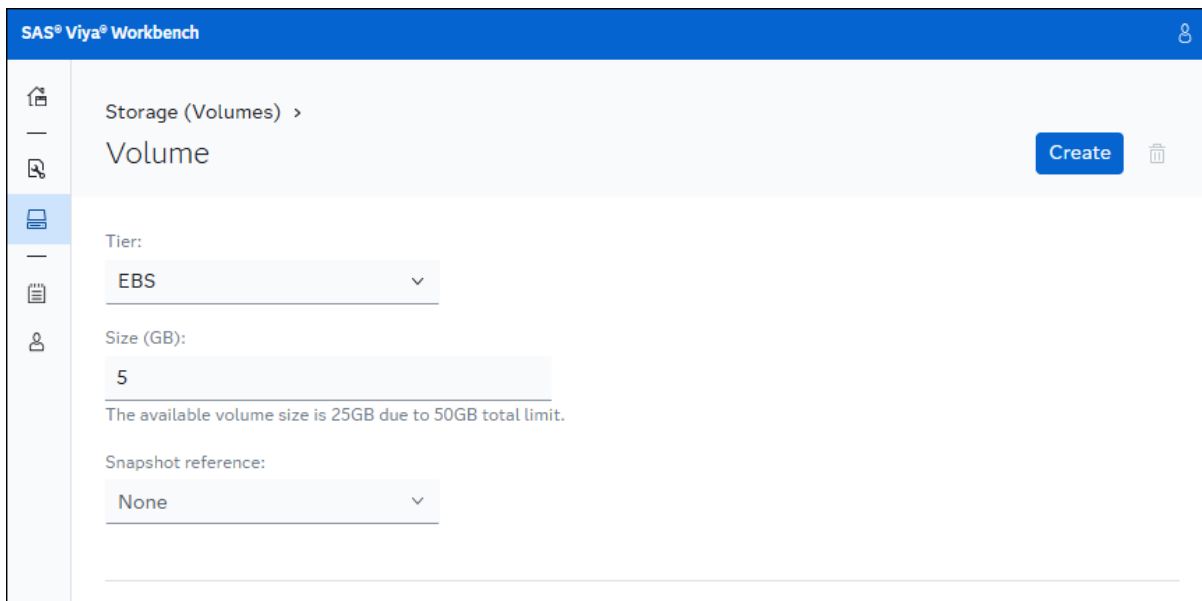
Create a Volume

When you create a workbench, you can select the primary volume and any additional volumes for the workbench.

To view the list of existing volumes, click  in the left navigation pane of SAS Viya Workbench.

To create a volume:

- 1 Click **New volume**.
- 2 Enter a name for the volume.
- 3 Select the tier (EBS or EFS) for the workbench.
- 4 Select the size (GB) of the workbench.
- 5 (Optional) Select the snapshot reference for the workbench. The new volume uses the settings from this snapshot reference.
- 6 Click **Create**.



SAS® Viya® Workbench

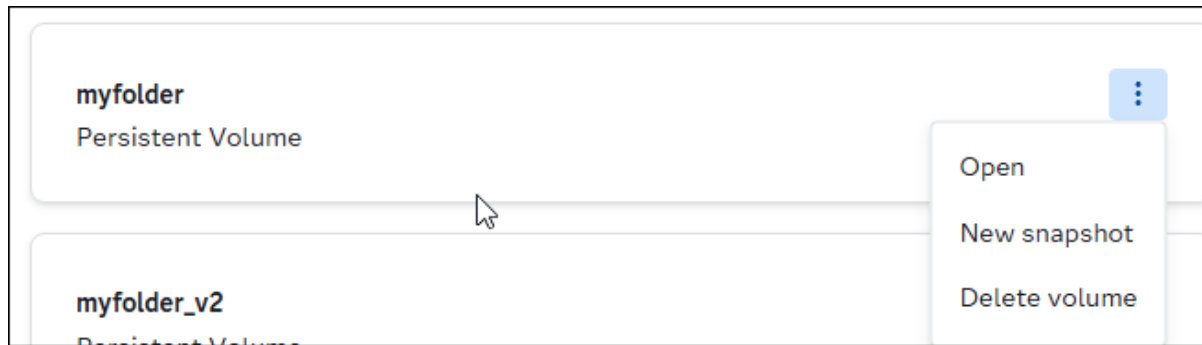
Storage (Volumes) > Volume Create

Tier:
EBS

Size (GB):
5
The available volume size is 25GB due to 50GB total limit.

Snapshot reference:
None

To delete a volume from the **Volume** tab, click  and select **Delete volume**.



Note: When you delete a volume, any snapshots that are associated with that volume are not deleted. For example, you might back up a volume by creating a snapshot and then delete the volume. Because the snapshot still exists, you can create a new volume from that snapshot.

Using a Home Volume

Benefits of Using a Home Volume

In SAS Viya Workbench, a home volume enables you to share extensions and configuration settings across workbenches. The home volume also includes your autoexec file.

Suppose you have a home volume and you install the SAS extension in Visual Studio Code for a workbench. Any configuration settings for that extension are saved in the home volume, so these configuration settings are available for all workbenches that use the home volume. If you do not use a home volume to save your configurations and settings for an extension, you have to specify these settings for every workbench where you use this extension.

SAS Viya Workbench administrators can allocate a home volume for users. SAS Viya Workbench users can access this home volume from a terminal session in their IDE. "\$HOME" and "\$SAS_USER" both point to `/home/sas` for all workbenches.

Allocating the home volume for the first time can take several minutes. After a home volume is created, all subsequent workbenches automatically use the home volume.

What Is the Difference between a Volume and a Home Volume?

When you create a workbench, you must specify a volume. This location saves information about the workbench and any data, code, or other files.

The home volume is optional. However, using a home volume means that you can share extensions and configuration settings across workbenches.

A home volume contains information for multiple workbenches and therefore, multiple storage locations can reside in a home volume. For example, you have two workbenches, Workbench_A and Workbench_B. Workbench_A uses **myfolder** as its volume. Workbench_B uses **sales** as its volume. Both **myfolder** and **sales** are in the home volume.

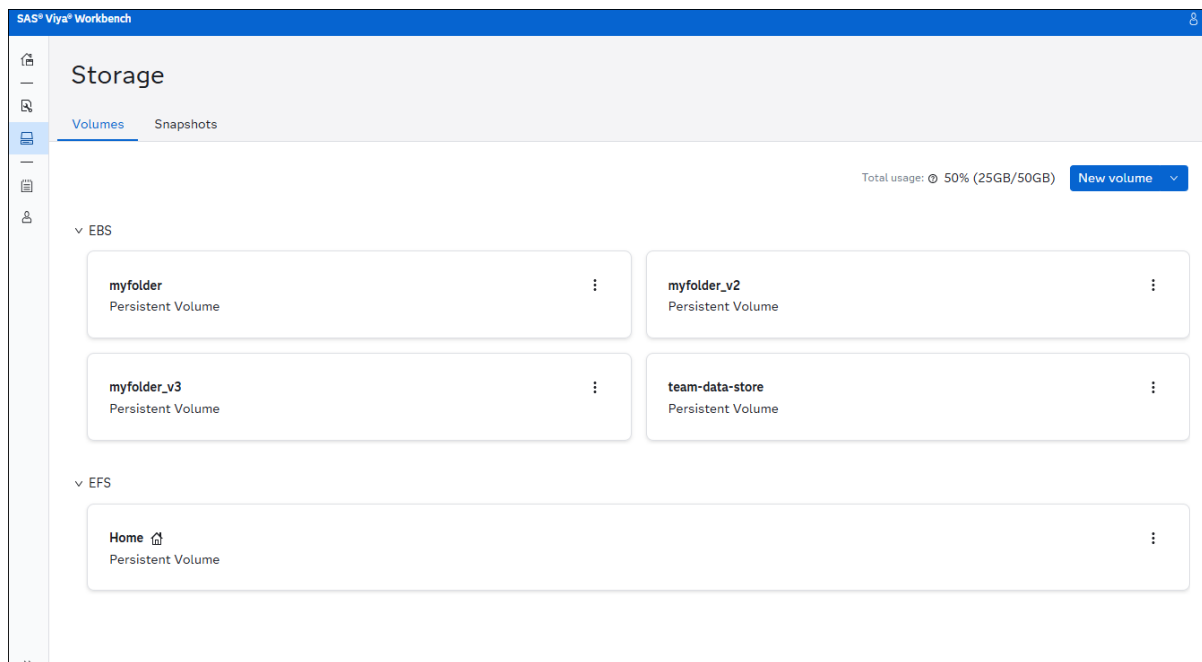
Create a Home Volume

- 1 In SAS Viya Workbench, click .
- 2 On the **Storage** page, select **New volume** ⇒ **New home volume**.

Note: You can have only one home volume. If a home volume already exists, that volume is deleted when you create the new home volume. All configurations and settings in the existing home volume are deleted.

- 3 Specify the size for the home volume.
- 4 Click **Create**.

The home volume now appears on the **Volumes** tab.



In this example, myfolder myfolder_v2, myfolder_v3, and team-data-store are volumes for a workbench. Home is the home volume.

IMPORTANT Workbenches that are created after you create the home volume automatically use the home volume. Workbenches that are created before you create the home volume do not use the home volume. You must delete and re-create any existing workbenches for those workbenches to use the home volume.

How to Access the Home Volume

The home volume is mounted to `/home/sas`. You can view the home volume by using a terminal session in your IDE.

Manage Volumes

Workbench users can create a volume when the user creates a workbench.

To view a list of volumes at your site, click  and click the **Volumes** tab.

Volumes can be defined by the end user or an administrator. The home volume (also referred to as the home directory) is named Home. For more information, see [“What Is the Difference between a Volume and a Home Volume?” on page 10](#).

From the **Volumes** page, you can complete these tasks:

- edit the settings of the volume. Click  and then **Open** to modify the name of the storage location or the size. The size is 5 GB. You can increase the size of the volume, but you cannot decrease the size. You can also add or remove a snapshot reference.
- delete a volume. Click  and then **Delete volume**.
- create a snapshot.

Network File System (NFS) Storage

You can mount an NFS directly to SAS Viya Workbench.

- 1 From the Microsoft Marketplace or AWS console, record the IP address for the SAS Viya Workbench data plane NAT gateway.
- 2 In your Azure or AWS subscription, add the IP address to the allowlist of NFS servers.

- 3 Email this information to the SAS Viya Workbench Ops Team (workbenchops@sas.com):
 - the host name / IP address for the NFS server
 - the NFS share name
 - the subdirectory for SAS Viya Workbench

By default, users are assigned a GUID of 001. Your administrator can use the admin panel in SAS Viya Workbench to override the GUID for a specific user.

Create a Snapshot

To create a snapshot of an existing volume:

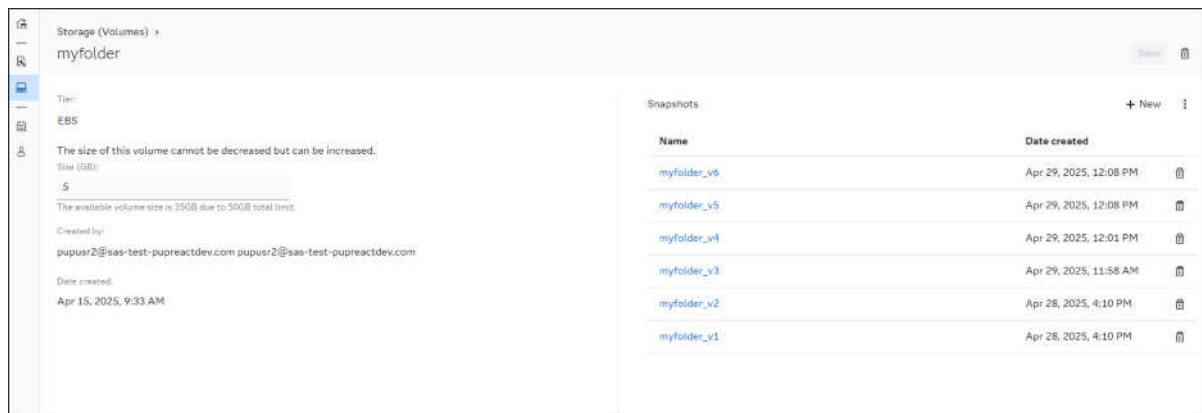
- 1 Click  and click the **Volumes** tab.
- 2 Click  and then **New snapshot**.

The snapshot now appears on the **Snapshots** tab.

Storage			
Volumes		Snapshots	
Filter		Delete	
☐	Name	Date created	
☐	myfolder_v1	Apr 28, 2025, 4:10 PM	
☐	myfolder_v2	Apr 28, 2025, 4:10 PM	
☐	myfolder_v3	Apr 29, 2025, 11:58 AM	
☐	myfolder_v4	Apr 29, 2025, 12:01 PM	

To view the snapshots associated with a specific volume:

- 1 Open the **Volumes** tab.
- 2 For the volume that you want to view, click  and then **Open**.



Create a Volume from a Snapshot

- 1 Click  and click the **Snapshots** tab.
- 2 Click the name of the snapshot.
- 3 On the *name-of-snapshot* (Snapshot) page, click **New volume from snapshot**.

The new volume is created using the settings from the selected snapshot.

Stop a Workbench

To stop a workbench, click  and select **Stop workbench** on the home page of SAS Viya Workbench.

Note: Closing your browser or the development environment (Jupyter or VS Code) does not stop the workbench. The workbench continues running. You can view the status of your workbench from the home page of SAS Viya Workbench. For more information, see [“Understanding the Status of a Workbench” on page 4](#).

Setting the Theme

You can change the appearance of the SAS Viya Workbench interface by using the **Theme** setting. The theme specifies the collection of colors, graphics, and fonts that appear in the application.

In SAS Viya Workbench, you can choose from these themes:

Dark

Presents an all-dark color palette for a better low-light experience.

High Contrast

Presents a black background with high-contrast foreground elements to meet the needs of users with low vision.

Light

Presents a light and clean color palette. The Light theme is the default application theme.

Accessing the SAS Documentation

To access the SAS documentation, click **View documentation** on the home page of SAS Viya Workbench.

If you are in your development environment (VS Code or Jupyter), you can access the documentation from the README.md file.

Working with Git and SSH

<i>Using Git from a Workbench</i>	17
<i>Cloning a Public GitHub Repository without Setup</i>	17
<i>Using SSH in Workbench</i>	18
<i>Configuring Global Git Settings</i>	19
<i>Message: "Fatal: Detected Dubious Ownership"</i>	19
<i>Pushing Files to a Remote Git Repository</i>	20

Using Git from a Workbench

The `git` command line tool comes preinstalled on your workbench. You must perform some setup steps to create a local copy of a repository, make changes, and submit your changes to the original repository. These setup steps might vary based on your Git host. For example, to submit changes to a GitHub repository, you must first have a GitHub account. However, you do not need to perform any setup steps to simply copy a repository to your workspace. To skip the setup process and copy a repository of Workbench examples, see ["Cloning a Public GitHub Repository without Setup"](#) on page 17.

Cloning a Public GitHub Repository without Setup

When you clone a repository, you create a local copy of all the files and associated metadata in the repository. You can clone a repository by running the following command:

```
git clone https://path/to/repository.git
```

For example, the following command clones a set of examples that you can use with your workbench:

```
git clone https://github.com/sassoftware/sas-viya-workbench-examples.git
```

You do not need to be signed in to GitHub to copy the examples to your workbench.

Using SSH in Workbench

Secure Shell (SSH) is a communication protocol that leverages public key encryption to establish a secure communication channel between a client and server. By setting up SSH on your workbench, you establish an identity that can be used to authenticate to a server. You can set up SSH credentials on your workspace in the same way as you would on any Unix-like operating system. The following steps show one way to set up SSH. Exact setup steps vary based on your site's security policies, but these steps show the common structure of how you can generate an SSH key for workbench:

- 1 Open the terminal for your chosen editor.
- 2 Run `ssh-keygen`. Respond yes to the prompt to accept the default file location for the identity or provide an alternate path. Respond to the prompt to define a password to use the identity or press Enter to leave it passwordless.
- 3 (Optional) Copy your public key to the server that you want to authenticate.

The key-generating step can vary according to your site's security policy. One common variation is to select a different encryption scheme, like ED25519, which you can specify using `ssh-keygen -t ed25519` in the key-generating step.

The file location is an important consideration for workbenches. By default, the file is created in `~/.ssh`. The `~/.ssh` folder corresponds to the home directory, so it is available in all of your workbenches and you have to perform the setup only once.

Copying your public key to a server can be done in many ways. For a web application, like GitHub, you can often find an interface that enables you to copy and paste the contents of your public key file, such as by following the instructions on this page: [Adding a new SSH key to your GitHub account](#). Alternatively, to authenticate as a user on another Unix-like machine, you can copy the identity with the following command: `ssh-copy-id -i /home/sas/.ssh/id_ed25519 user@host`. The `host` option corresponds to the host machine that you want to connect to. The `user` option corresponds to user that you want to sign in as on the host machine.

Note: Familiarize yourself with your site's security policy before creating an SSH identity.

Configuring Global Git Settings

For your commits to be valid, Git requires that you configure your user name and email. You can set your user name and email by running the following two commands:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

No output is written to the terminal when the command succeeds. You can confirm the configuration by running `git config --global --list` and ensuring that the `user.email` and `user.name` are the values that you expect.

By specifying the `--global` option when you set the configuration values, the settings apply to all your Git commands unless overwritten by a local configuration. Your global configurations are written to the `.gitconfig` file in your home directory. Because it is a part of your home directory, the settings persist for all workbench instances.

Message: "Fatal: Detected Dubious Ownership"

Some Git commands result in the following message:

```
fatal: detected dubious ownership in repository at path
To add an exception for this directory, call:
git config --global --add safe.directory path
```

The solution is to submit the command proposed in the message. Specifically, add the repository as a safe path by running the following command:

```
git config --global --add safe.directory path
```

The message is produced as a consequence of how workbench manages file ownership. Git raises the issue whenever the user submitting Git commands is different from the user that owns the files of the Git repository. By default, the user that submits Git commands is the `sas` user. Files, however, are nominally owned by a proxy user indicated by a UID. The files are owned by the proxy user even if the `sas` user ran the commands to produce the files.

Pushing Files to a Remote Git Repository

Once you have a local Git repository, you can track changes to files and push the changes to a remote repository, such as GitHub.

- 1 Make changes to the files in your repository.
- 2 Use `git add filename` to add a file to the staging area. Repeat for all files that you want to add.
- 3 Use `git commit` to commit all staged files to your local repository.
- 4 Use `git push <remote> <branch>` to push your changes from your local branch to a remote branch.

Before you can push changes, however, the following things must be true:

- The remote repository must exist.
- You must be authorized to push to the remote repository.
- Your local repository must know how to access the remote repository.

The first two items vary based on who is providing your remote Git repositories. For example, for GitHub, you can use the web user interface to create and fork repositories. To authenticate to GitHub, you must have an account and a way of authenticating from the command line to your account, such as by setting up SSH or GitHub tokens. The process is different for different Git hosts. See the [GitHub documentation](#), talk to your Git administrator, or consult [the Git handbook](#) for information about those items.

To ensure that your local repository knows about the remote repository, you might need to perform extra steps to add the remote. Often, the remote is defined automatically when you clone the repository. By convention, the remote is called "origin" and uses the protocol and URL that you used to clone the repository. However, you might need to modify the remote definitions if you do not want to push to the remote that you cloned from. You can add, remove, or get information about remotes by using the `git remote` commands. The following `git remote` commands are useful for managing remotes:

- `git remote --verbose` lists the name of each remote and its URL. Each remote is often listed twice, once for where it fetches changes from and once for where it pushes changes to. The fetch and push remotes often use the same URL.
- `git remote add nameURL` adds a remote to the list of known remotes. It uses the protocol and name that you specify in the URL portion.
- `git remote remove name` removes a remote from the list of known remotes.

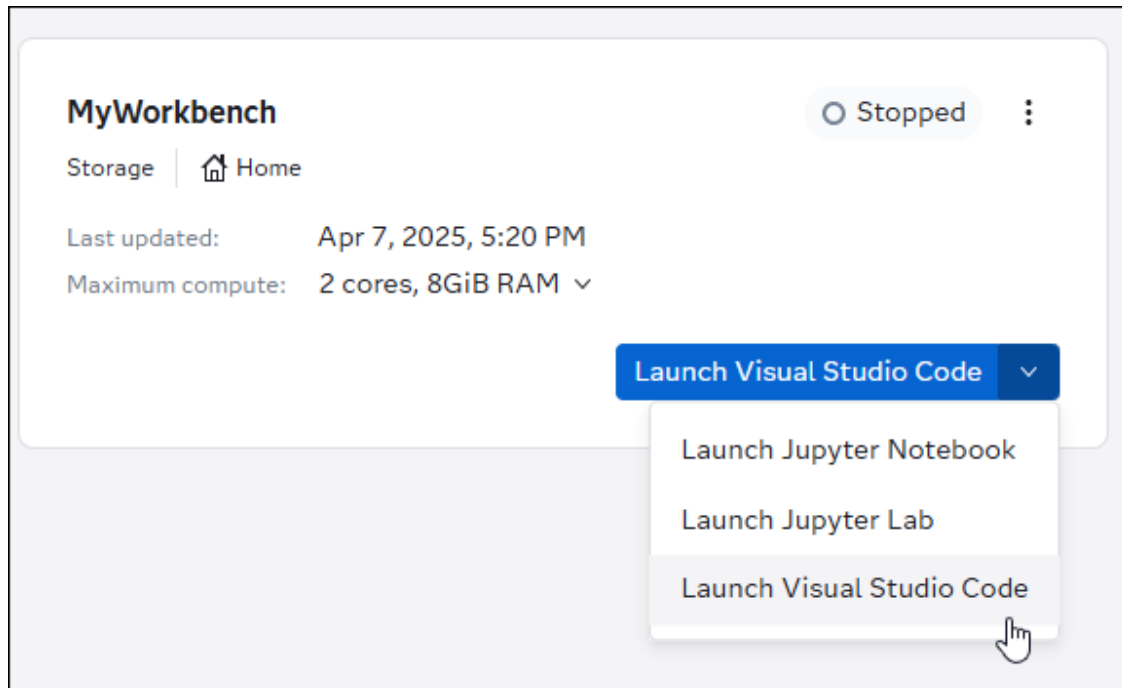
For more information about using the `git remote` command, see the [Git remote documentation](#).

Working in Visual Studio Code

<i>Open a Workbench in Visual Studio Code</i>	21
<i>Accessing SAS Viya Workbench in Visual Studio Code</i>	22
<i>Run a Python Example</i>	24
<i>Run an Example SAS Program</i>	25
<i>Access SAS Libraries</i>	28
<i>Opening a Terminal in Visual Studio Code</i>	29
<i>Select a SAS Color Theme</i>	30
<i>Create a SAS Notebook</i>	30
<i>Using the AutoExec Feature in Visual Studio Code</i>	32

Open a Workbench in Visual Studio Code

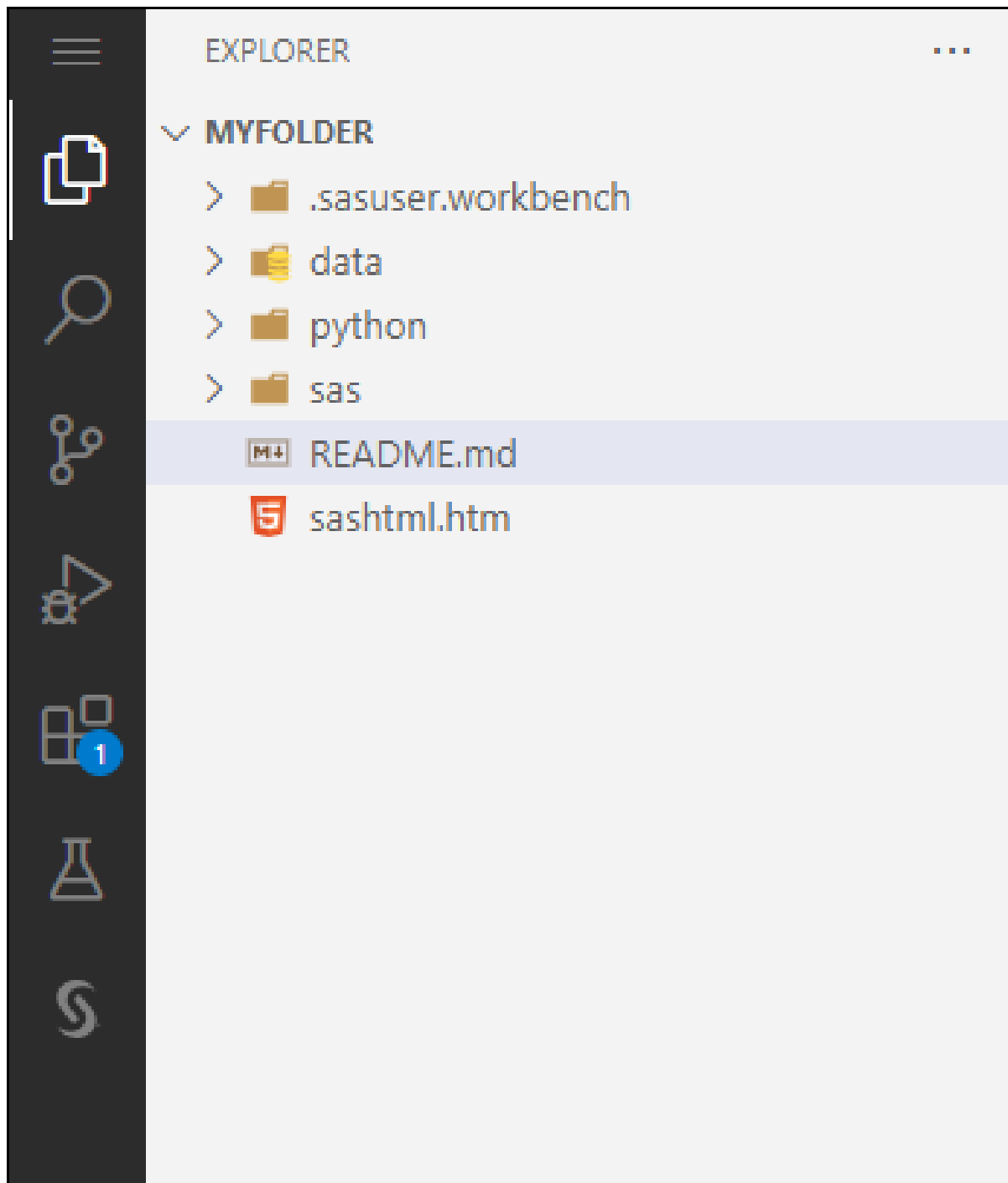
You can open the workbench from the home page. From the drop-down list, select **Launch Visual Studio Code**.



Note: In Visual Studio Code, only English is currently supported.

Accessing SAS Viya Workbench in Visual Studio Code

When you open the workbench in Visual Studio Code, the Explorer displays the folder that contains the workbench. MYFOLDER is the storage location that you specified when you created the workbench.



After you clone the examples repository, you should see these folders:

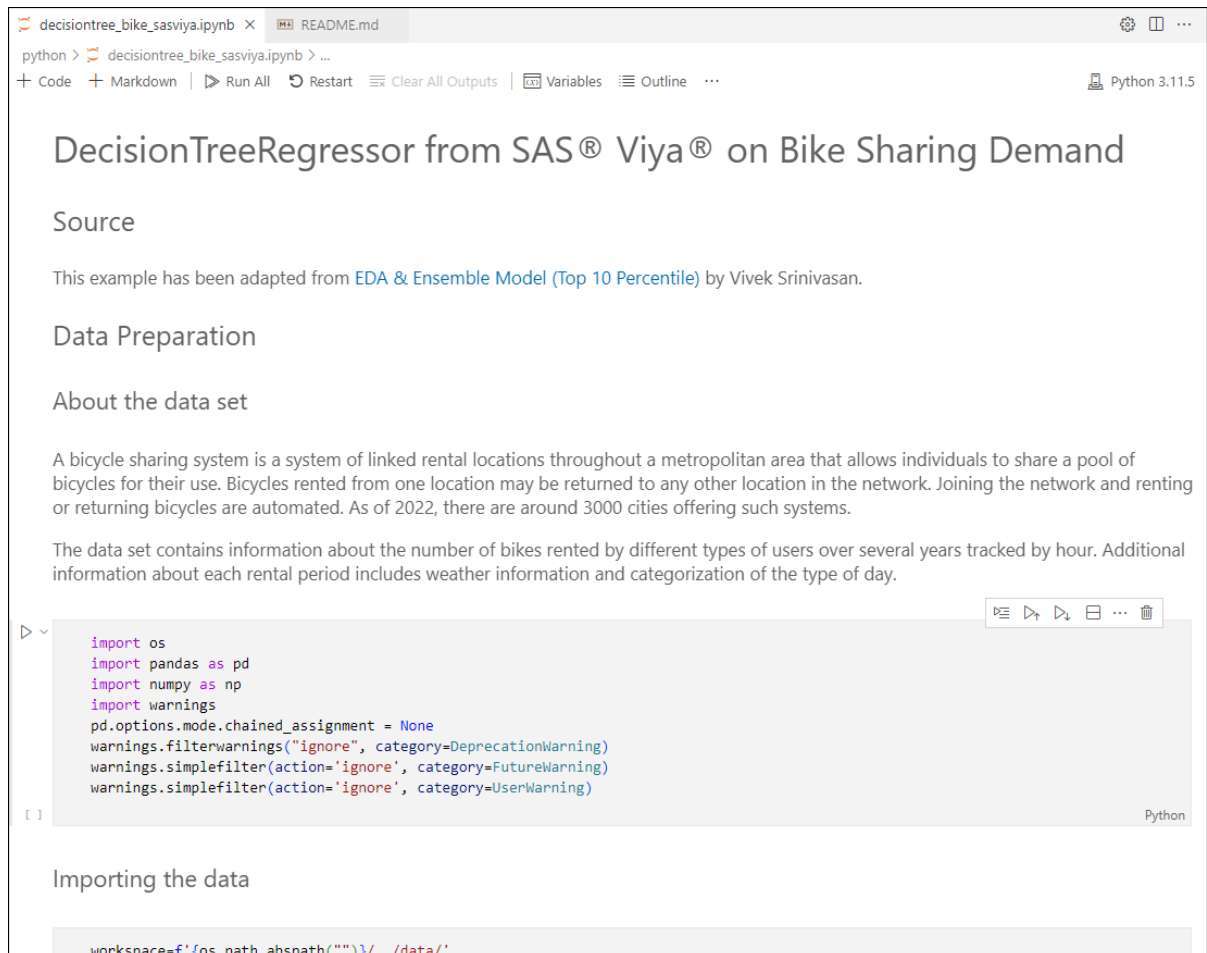
- **data** contains the data sources for the examples. For information about each data source, see README.md.
- **python** contains the example Python notebooks. To see the notebooks organized by algorithm, by data set, and by library, see README.md.
- **sas** contains SAS code for the examples. To see the programs organized by SAS procedure and SAS data set, see README.md.

For more information, see [“Cloning the Examples Repository” on page 5](#).

Run a Python Example

SAS Viya Workbench provides access to several example notebooks. To view an example, expand the **python** folder and double-click the file that you want to open. For this example, double click **decisiontree_bike_sasviya.ipynb**.

The entire example opens in the VS Code workspace.



To run the entire example, click **Run All**. The output for each code block appears after the individual cell.

Sample of the first few rows

[5] bikeData.head(5) Python

...

	date	season	year	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cc
0	1/1/2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	
1	1/1/2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	
2	1/1/2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	
3	1/1/2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	
4	1/1/2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	

Variable data types

[4] bikeData.dtypes Python

...

date	object
season	int64
year	int64
month	int64
hour	int64
holiday	int64
weekday	int64
workingday	int64
weather	int64
temp	float64
atemp	float64

To execute an individual cell, click the arrow on the left of the cell. You can also use the icons on the right to run the code by line, execute previous and subsequent cells, and additional options.

Sample of the first few rows

hikeData.head(5) Python

...

	date	season	year	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cc
0	1/1/2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	
1	1/1/2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	
2	1/1/2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	
3	1/1/2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	
4	1/1/2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	

Run an Example SAS Program

SAS Viya Workbench provides access to several example SAS programs. To view a SAS program, expand the **sas** folder and double-click the file that you want to open. For this example, double click **classification_heart.sas**.

The entire program opens in the VS Code Workspace.

```

1  /*****
2  EXAMPLE:  classification_heart.sas
3  DATA:    heart_disease.csv
4  DESCRIPTION: The data set contains measurements on 304 patients, consisting
5              of factors that potentially indicate the presence or absence of
6              heart disease.
7  PURPOSE:  In this example, we will show different binary classification
8              modeling techniques to predict the heart disease.
9  SOURCE:   Adapted from
10             "Heart Disease prediction Random forest Classifier"
11             https://www.kaggle.com/code/mruanova/heart-disease-prediction-random-forest-classifier
12             Mau Rua
13  *****/
14
15
16  title 'Predicting heart disease using different modeling techniques';
17
18  /*****
19
20  Loading the heart_disease data.
21
22  *****/
23
24  options nosource;
25  proc import
26      datafile="&WORKSPACE_PATH./data/heart_disease.csv"
27      out=heart_disease dbms=csv replace;

```

To run the entire program, click . To run a code block, select the code that you want to run and click .

Any output appears in a **Result** tab. The SAS log is available from the **Output** pane.

The screenshot shows the VS Code interface with the SAS extension. The editor displays the SAS code file `classification_heart.sas`, and the output window shows the results of running the program.

SAS Code:

```

1 /*****
2 EXAMPLE:   classification_heart.sas
3 DATA:     heart_disease.csv
4 DESCRIPTION: The data set contains me
5             of factors that potentia
6             heart disease.
7 PURPOSE:   In this example, we will
8             modeling techniques to p
9 SOURCE:     Adapted from
10            "Heart Disease predictio
11            https://www.kaggle.com/c
12            Mau Rua
13 *****/
14
15
16 title 'Predicting heart disease using
17
18 /*****
19
20 Loading the heart_disease data.
21
22 *****/
23
24 options nosource;
25 proc import
26     datafile="&WORKSPACE_PATH./data/he
27     out=heart_disease dbms=csv replace

```

Output:

Predicting heart disease using different modeling techniques
Portion of heart_disease data

Obs	target	sex	age	trestbps	chol	thalch	oldpeak
1	No heart disease	Male	63	145	233	150	
2	With heart disease	Male	67	160	286	108	
3	With heart disease	Male	67	120	229	129	
4	No heart disease	Male	37	130	250	187	
5	No heart disease	Female	41	130	204	172	

Predicting heart disease using different modeling techniques
Percentage of heart disease in the data

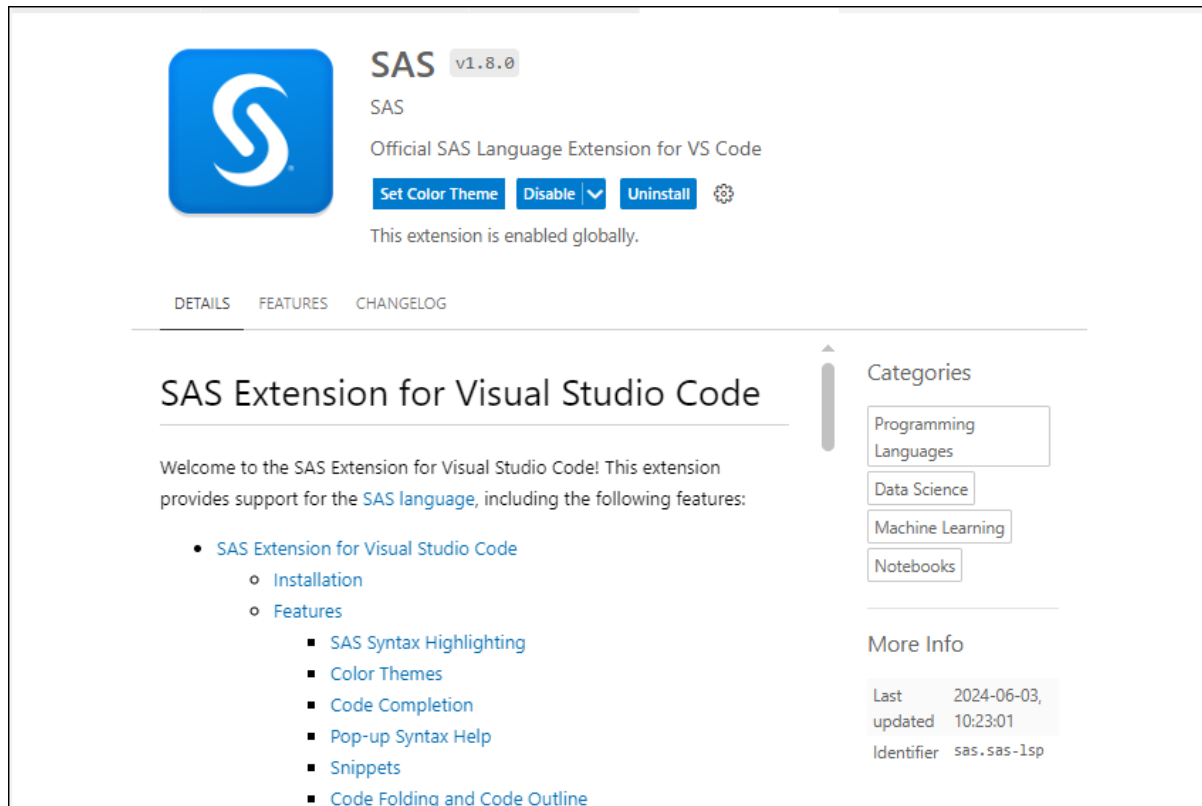
Output Log:

```

NOTE: DATA statement used (Total process time):
NOTE: PROCEDURE SGPLOT used (Total process time):
      real time          0.71 seconds
17
The SAS System
Thursday, June 13, 2024 08:53:00 AM
      cpu time           0.09 seconds
NOTE: There were 10 observations read from the data set WORK.ALLMETHODS.
542
543 title;
544 ;*';*';*/run;quit;ods html5 close;

```

For more information about how to work with your SAS code in VS Code, see the documentation for the SAS Extension for Visual Studio Code.

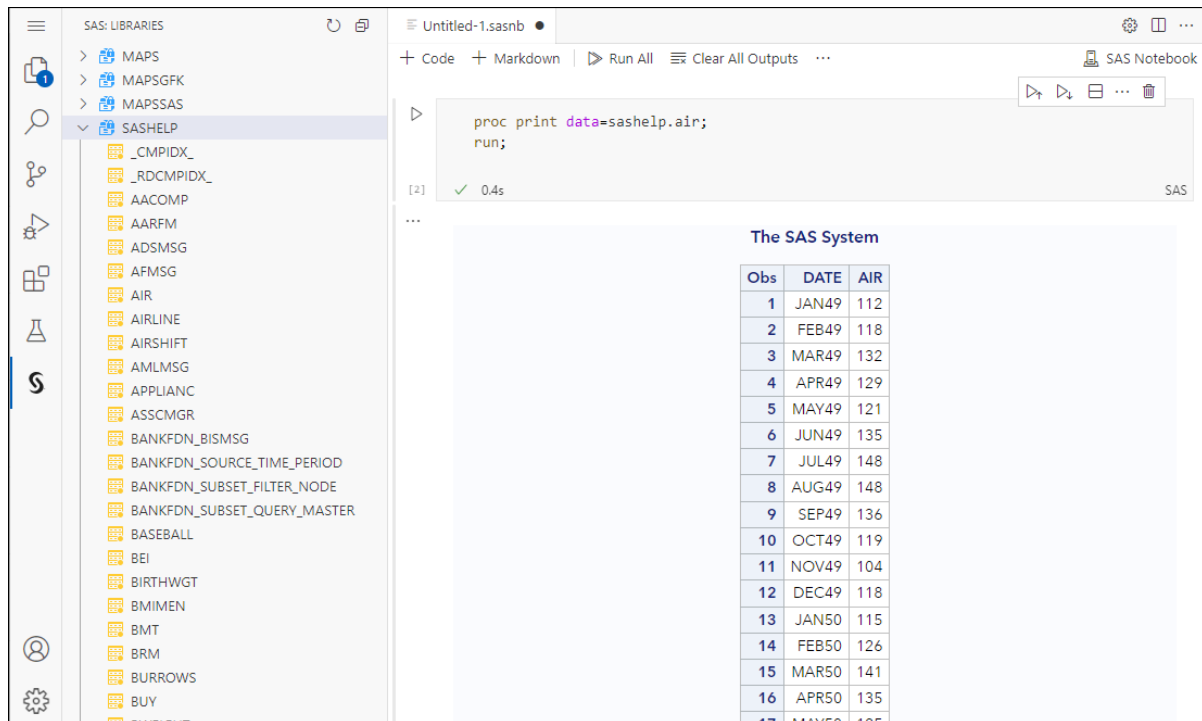


Access SAS Libraries

These SAS libraries are available through SAS Viya Workbench:

- MAPS
- MAPSGFK
- MAPSSAS
- SASHELP
- SASUSER
- WORK

To access SAS libraries, click .



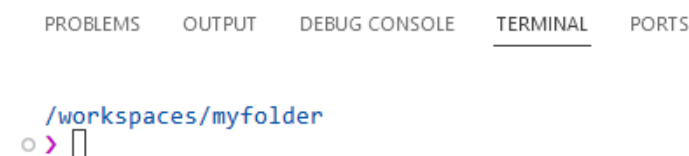
Expand a SAS library to see the data sets that you can point to from your SAS programs or Python notebook.

Opening a Terminal in Visual Studio Code

SAS Viya Workbench includes a full development environment, including the ability to invoke shell commands.

To open a terminal in Visual Studio Code, click  and select **Terminal** ⇒ **New Terminal**.

By default, the integrated terminal opens a Bash shell in the directory that you indicated as your mounting folder when you created the workbench.



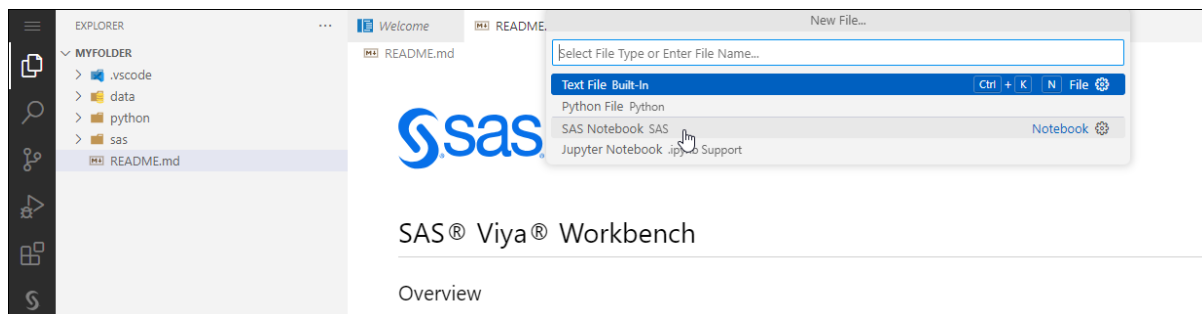
Select a SAS Color Theme

SAS has developed three themes: SAS Light, SAS Dark, and SAS High Contrast, which is recommended for accessibility.

- 1 Click  and select **File** ⇒ **Preferences** ⇒ **Theme** ⇒ **Color Theme**
- 2 Select the SAS color theme.

Create a SAS Notebook

To create a SAS notebook, click  and select **File** ⇒ **New File**. For the file type, select **SAS Notebook**.



After you run a section of SAS code, the output appears directly below the code block.

The screenshot shows a SAS Notebook with a code editor and an output area. The code editor contains the following SAS code:

```

accuracy = &&acc_test_&currentProc;
output;
procname = "&currentProc";
type = "test";
accuracy = &&acc_test_&currentProc;
output;
%end;

run;
proc sgplot data=allMethods;
vbar procname / response=accuracy group=type nostatlabel datalabel
              groupdisplay=cluster dataskin=pressted;
xaxis display=(nolabel);
yaxis grid;
run;
%mend;

title2 'Compare accuracy across all 5 procedures';
%CreateComparison;

title;

```

The output area displays the results of the SAS code. It includes a table titled "Predicting heart disease using different modeling techniques" and a chart titled "Predicting heart disease using different modeling techniques".

Predicting heart disease using different modeling techniques
Portion of heart_disease data

Obs	target	sex	age	trestbps	chol	thalch	oldpeak	ca	cp	exang	slope	thal	restecg	fbs
1	No heart disease	Male	63	145	233	150	2.3	0	3	0	0	1	0	1
2	With heart disease	Male	67	160	286	108	1.5	3	0	1	1	2	0	0
3	With heart disease	Male	67	120	229	129	2.6	2	0	1	1	0	0	0
4	No heart disease	Male	37	130	250	187	3.5	0	2	0	0	2	1	0
5	No heart disease	Female	41	130	204	172	1.4	0	1	0	2	2	0	0

Predicting heart disease using different modeling techniques
Percentage of heart disease in the data

Heart Disease Percentage

To view the SAS log, click and select **Change Presentation**.

From the drop-down list, select **application/vnd.sas.compute.log.lines SAS Log Renderer**.

The screenshot shows the SAS Notebook interface with a dropdown menu open. The dropdown menu is titled "Select mimetype to render for current output" and contains the following options:

- application/vnd.sas.compute.log.lines Currently Active
- SAS Log Renderer (SAS.sas-lsp)
- application/vnd.sas.ods.html5
- SAS HTML Renderer (SAS.sas-lsp)

The code editor shows the following SAS code:

```

length procname $16. type $8.;
%do i = 1 %to %sysfunc(countw(&allprocs));
  %let currentProc = %scan(&allprocs,&i);
  procname = "&currentProc";

```

Now the SAS log is displayed under each code block.

```

        accuracy = &&&acc_test_&currentProc;
        output;
    %end;
run;
proc sgplot data=allMethods;
    vbar procname / response=accuracy group=type nostatlabel datalabel
        groupdisplay=cluster dataskin=priest;
    xaxis display=(nolabel);
    yaxis grid;
run;
%mend;

title2 'Compare accuracy across all 5 procedures';
%CreateComparison;

title;

```

[1] ✓ 39.8s

```

... 545 title;footnote;ods _all_ close;
546 ods graphics on;
547 ods html5 style=HTMLEncore options(bitmap_mode='inline' svg_mode='inline');
NOTE: Writing HTML5 Body file: sashtml1.htm
548 %let _SASPROGRAMFILE = %nrquote(%nrstr(Untitled-1.sasnb));
549 /*****
550 EXAMPLE:      classification_heart.sas
551 DATA:        heart_disease.csv
552 DESCRIPTION:  The data set contains measurements on 304 patients, consisting
553               of factors that potentially indicate the presence or absence of
554               heart disease.
555 PURPOSE:      In this example, we will show different binary classification
556               modeling techniques to predict the heart disease.
557 SOURCE:       Adapted from
558               "Heart Disease prediction Random forest Classifier"
559               https://www.kaggle.com/code/muannova/heart-disease-prediction-random-forest-classifier

```

Using the AutoExec Feature in Visual Studio Code

Using the SAS Extension in Visual Studio Code, you can set up an autoexec file that runs when a session starts. Any changes made to the autoexec file require you to close and reopen your session to take effect.

For more information, see [SAS Autoexec Settings](#) in the *SAS Extension for Visual Studio Code Documentation*.

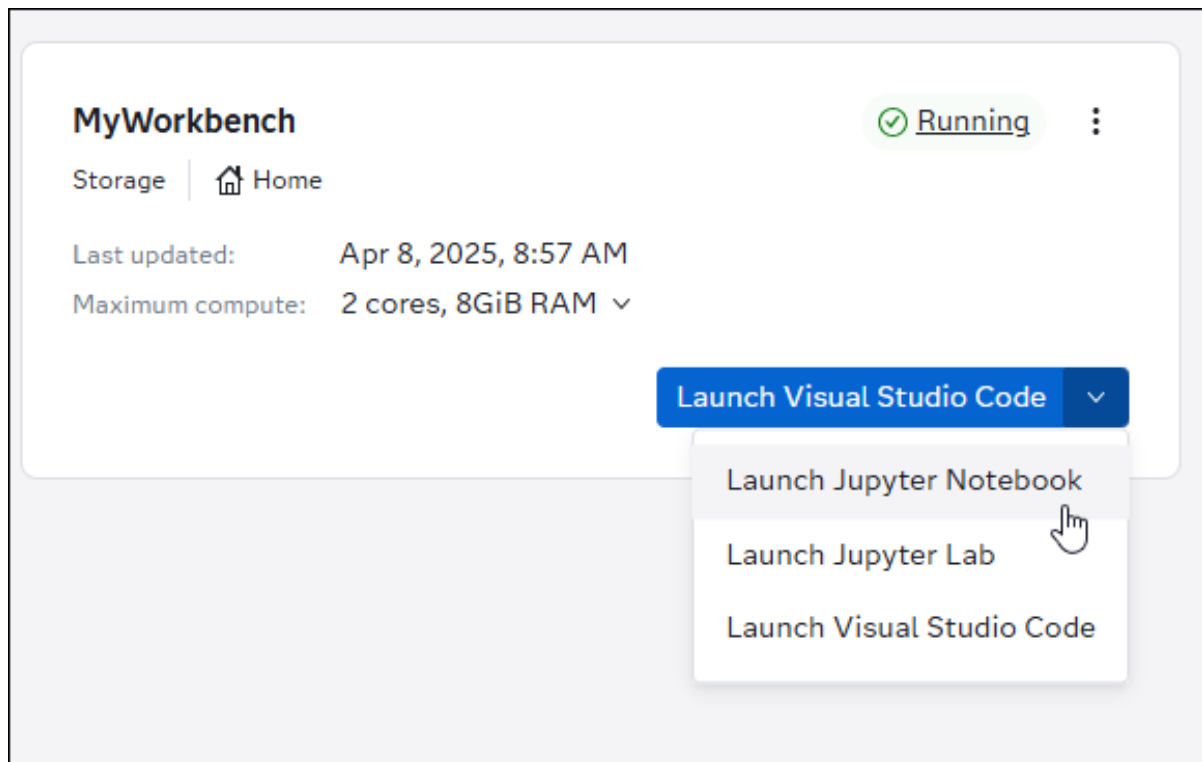
Save your autoexec file to the home directory (`/home/sas`) to share the settings across multiple workbenches.

Working in Jupyter Notebook

<i>Open a Workbench in Jupyter Notebook</i>	33
<i>Accessing Your Workbench in Jupyter Notebook</i>	34
<i>Opening a Terminal in Jupyter Notebook</i>	35
<i>Run a Python Example</i>	35
<i>Upload Data in Jupyter Notebook</i>	36

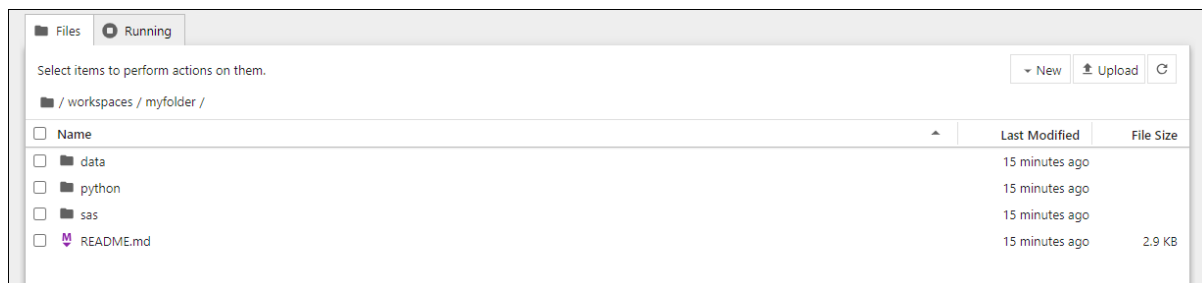
Open a Workbench in Jupyter Notebook

You can open the workbench from the home page. From the drop-down list, select **Launch Jupyter Notebook**.



Accessing Your Workbench in Jupyter Notebook

When you open the workbench in Jupyter Notebook, the **Files** tab displays the folder that contains the workbench. The storage location, **myfolder**, is what you specified when you created the workbench.



After you clone the examples repository, you should see these folders:

- **data** contains the data sources for the examples. For information about each data source, see README.md.
- **python** contains the example Python notebooks. To see the notebooks organized by algorithm, by data set, and by library, see README.md.

- **sas** contains SAS code for the examples. To see the programs organized by SAS procedure and SAS data set, see README.md.

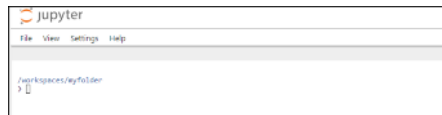
For more information, see [“Cloning the Examples Repository”](#) on page 5.

Opening a Terminal in Jupyter Notebook

SAS Viya Workbench includes a full development environment, including the ability to invoke shell commands.

To open a terminal, select **New** ⇒ **Terminal**.

By default, the terminal opens a Bash shell in the directory that you indicated as your mounting folder when you created the workbench.



Run a Python Example

SAS Viya Workbench provides access to several example notebooks. To view an example, double click the **python** folder and double-click the file that you want to open.

To run the entire example, select **Run** ⇒ **Run All Cells**.

To run selected text, use the options in the **Run** menu.

The output for a code block appears under that cell.

Importing the data

```
[2]: workspace=f'{os.path.abspath("")}/../data/'
bikeData = pd.read_csv(workspace + "bike_sharing_demand.csv")
bikeData.shape
```

[2]: (17379, 16)

Sample of the first few rows

```
[3]: bikeData.head(5)
```

	date	season	year	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	1/1/2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	1/1/2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	1/1/2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	1/1/2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	1/1/2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Variable data types

```
[4]: bikeData.dtypes
```

```
[4]: date          object
season          int64
year            int64
month           int64
hour            int64
holiday          int64
weekday          int64
workingday       int64
weather          int64
temp            float64
atemp           float64
humidity         float64
windspeed       float64
casual           int64
registered       int64
```

Upload Data in Jupyter Notebook

From the workbench's home page in Jupyter Notebook, click **Upload**. Select the data source saved on your local computer and click **Open**. The data source now appears as an item in the list.

The screenshot shows the Jupyter Notebook file browser interface. At the top, there are tabs for 'Files' and 'Running'. Below the tabs, a message says 'Select items to perform actions on them.' To the right of this message are buttons for 'New', 'Upload', and a refresh icon. The main area displays a list of files and folders under the path '/ workspaces / myfolder /'. The list has columns for 'Name', 'Last Modified', and 'File Size'. The items listed are: 'data' (folder, 18 minutes ago), 'python' (folder, 18 minutes ago), 'sas' (folder, 18 minutes ago), 'banking.csv' (file, 1 minute ago, 11 KB), and 'README.md' (file, 18 minutes ago, 2.9 KB).

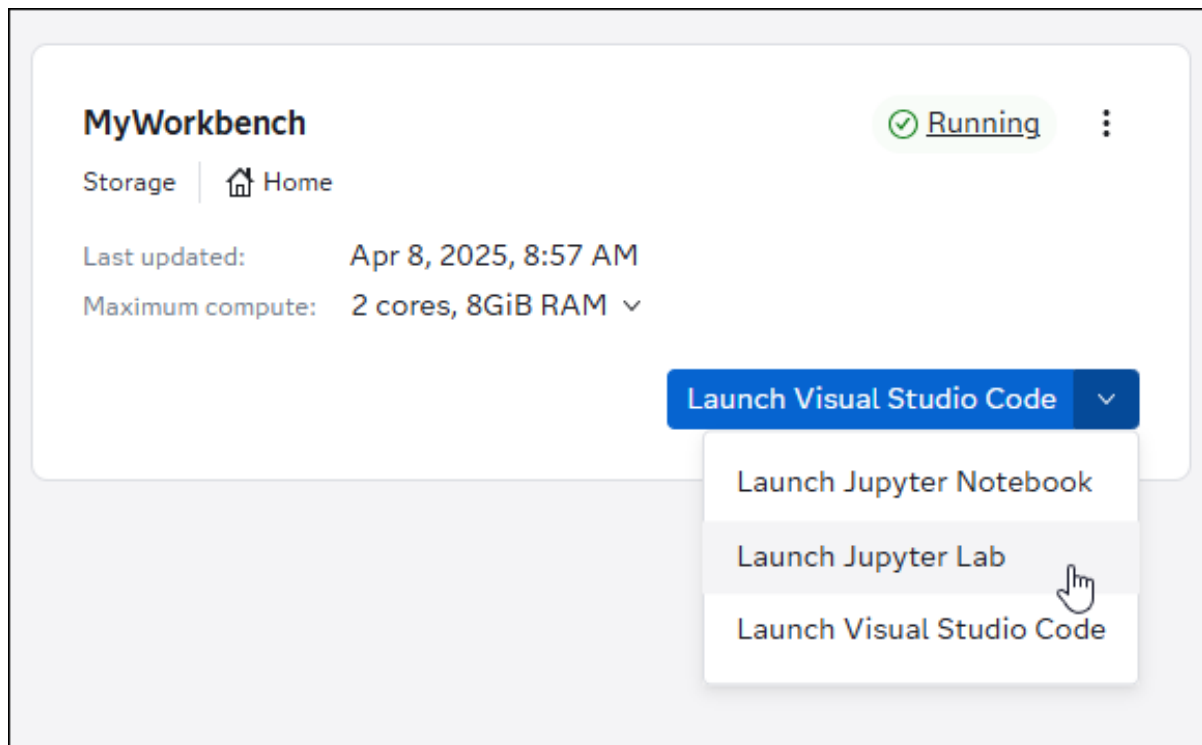
Name	Last Modified	File Size
data	18 minutes ago	
python	18 minutes ago	
sas	18 minutes ago	
banking.csv	1 minute ago	11 KB
README.md	18 minutes ago	2.9 KB

Working in Jupyter Lab

<i>Open a Workbench in Jupyter Lab</i>	37
<i>Accessing Your Workbench in Jupyter Lab</i>	38
<i>Run a Python Example</i>	39
<i>Upload Data in Jupyter Lab</i>	40
<i>Opening a Terminal in Jupyter Lab</i>	41

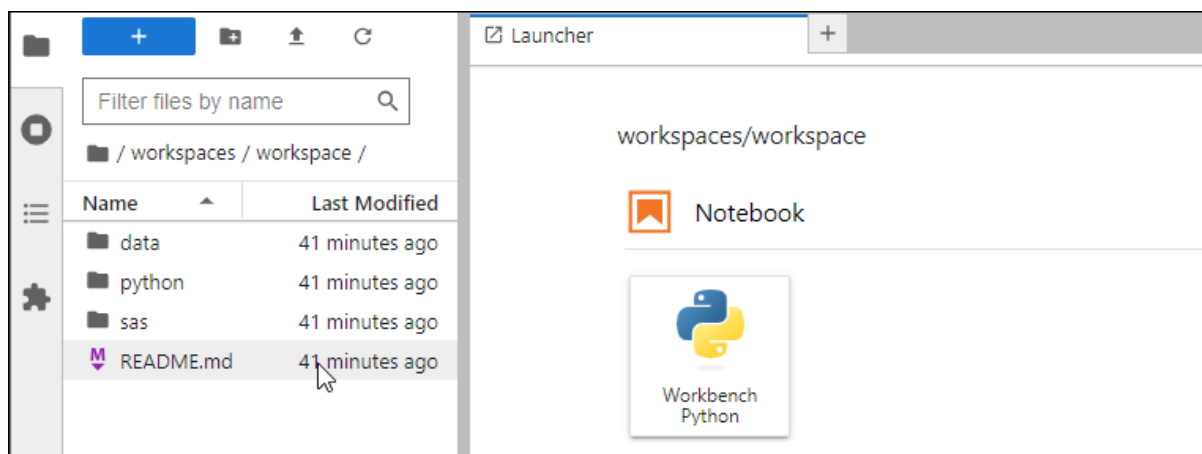
Open a Workbench in Jupyter Lab

You can open the workbench from the home page. From the drop-down list, select **Launch Jupyter Lab**.



Accessing Your Workbench in Jupyter Lab

When you open a workbench in Jupyter Lab, the **File Browser** tab lists the contents of the workbench.



After you clone the examples repository, you should see these folders:

- **data** contains the data sources for the examples. For information about each data source, see README.md.
- **python** contains the example Python notebooks. To see the notebooks organized by algorithm, by data set, and by library, see README.md.
- **sas** contains SAS code for the examples. To see the programs organized by SAS procedure and SAS data set, see README.md.

For more information, see [“Cloning the Examples Repository” on page 5](#).

Run a Python Example

The examples repository for SAS Viya Workbench provides access to several example notebooks. To view an example, double click the **python** folder and double-click the file that you want to open.

To run the entire example, select **Run** ⇒ **Run All Cells**. Use the options in the **Run** menu to run individual cells.

The output appears directly under the code block.

decisiontree_bike_sasviya.ipyn X

bikeData.shape

[2]: (17379, 16)

Sample of the first few rows

[3]: bikeData.head(5)

	date	season	year	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	winds
0	1/1/2011	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	
1	1/1/2011	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	
2	1/1/2011	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	
3	1/1/2011	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	
4	1/1/2011	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	

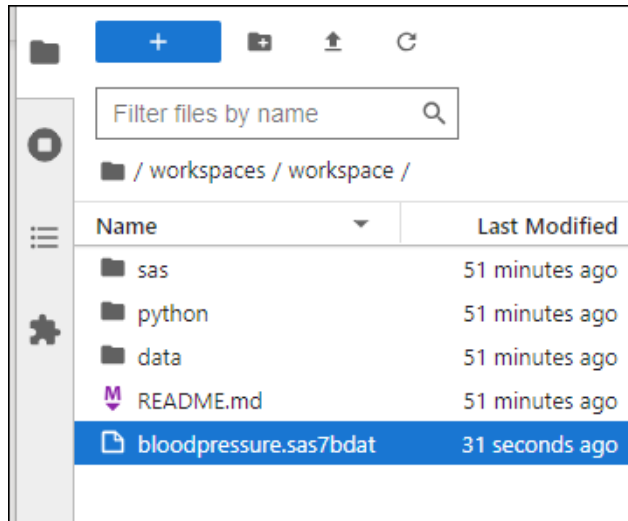
Variable data types

[4]: bikeData.dtypes

```
[4]: date          object
     season        int64
     year          int64
     month         int64
     hour          int64
     holiday       int64
     weekday       int64
     workingday    int64
     weather       int64
     temp          float64
     atemp         float64
     humidity      float64
     windspeed     float64
     casual        int64
     registered    int64
     count         int64
     dtype: object
```

Upload Data in Jupyter Lab

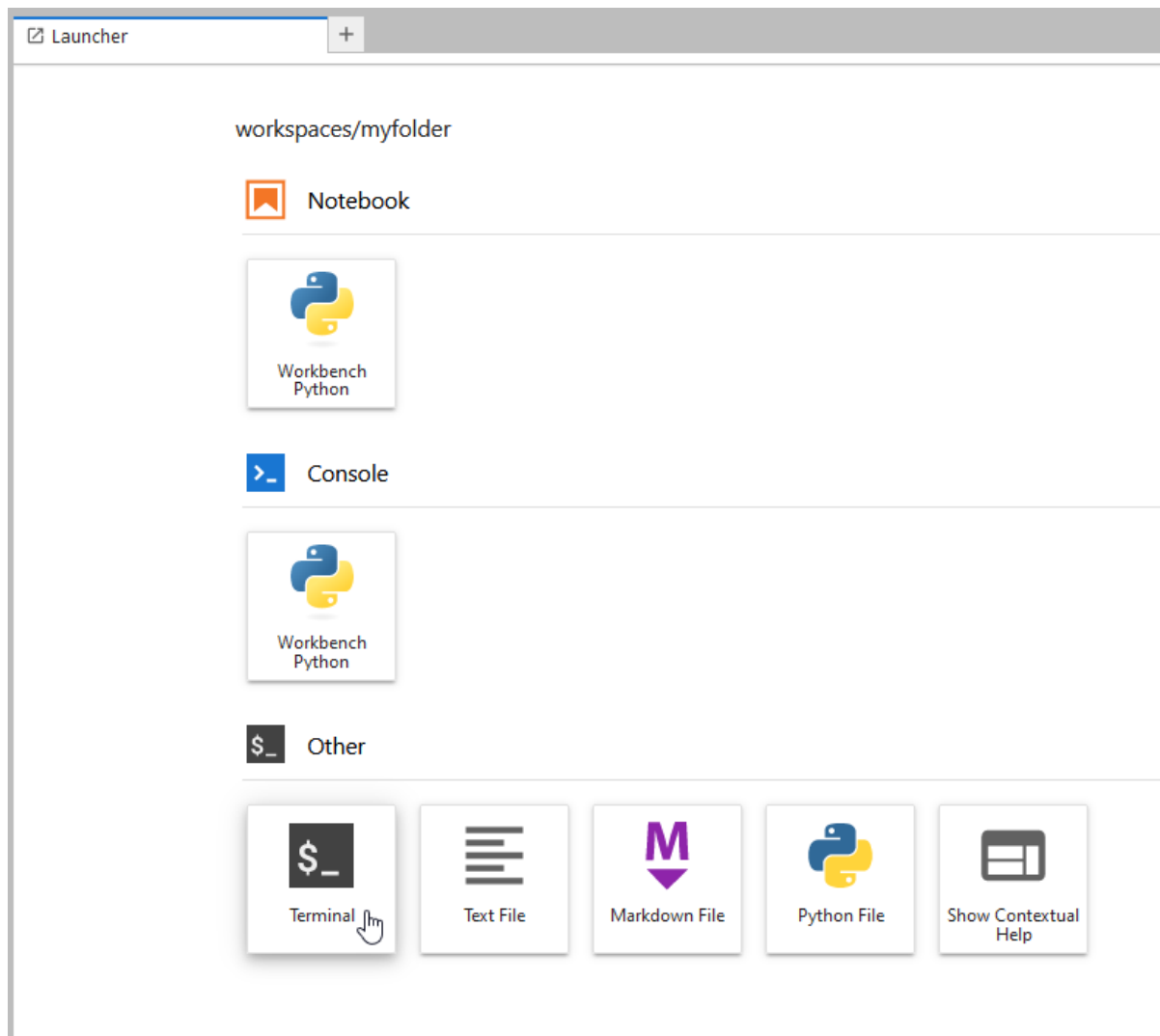
To upload a data source, click . The uploaded data now appears in the **File Browser**.



Opening a Terminal in Jupyter Lab

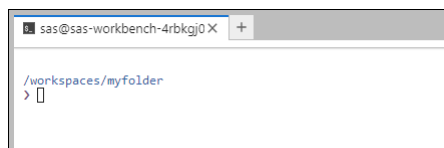
SAS Viya Workbench includes a full development environment, including the ability to invoke shell commands.

You can open a terminal from Jupyter Lab's launcher page. If you do not see a launcher page, select **File** ⇒ **New Launcher**.



From the launcher page, you can open a terminal by selecting the **Terminal** tile.

By default, the integrated terminal opens a Bash shell in the directory that you indicated as your mounting folder when you created the workbench.

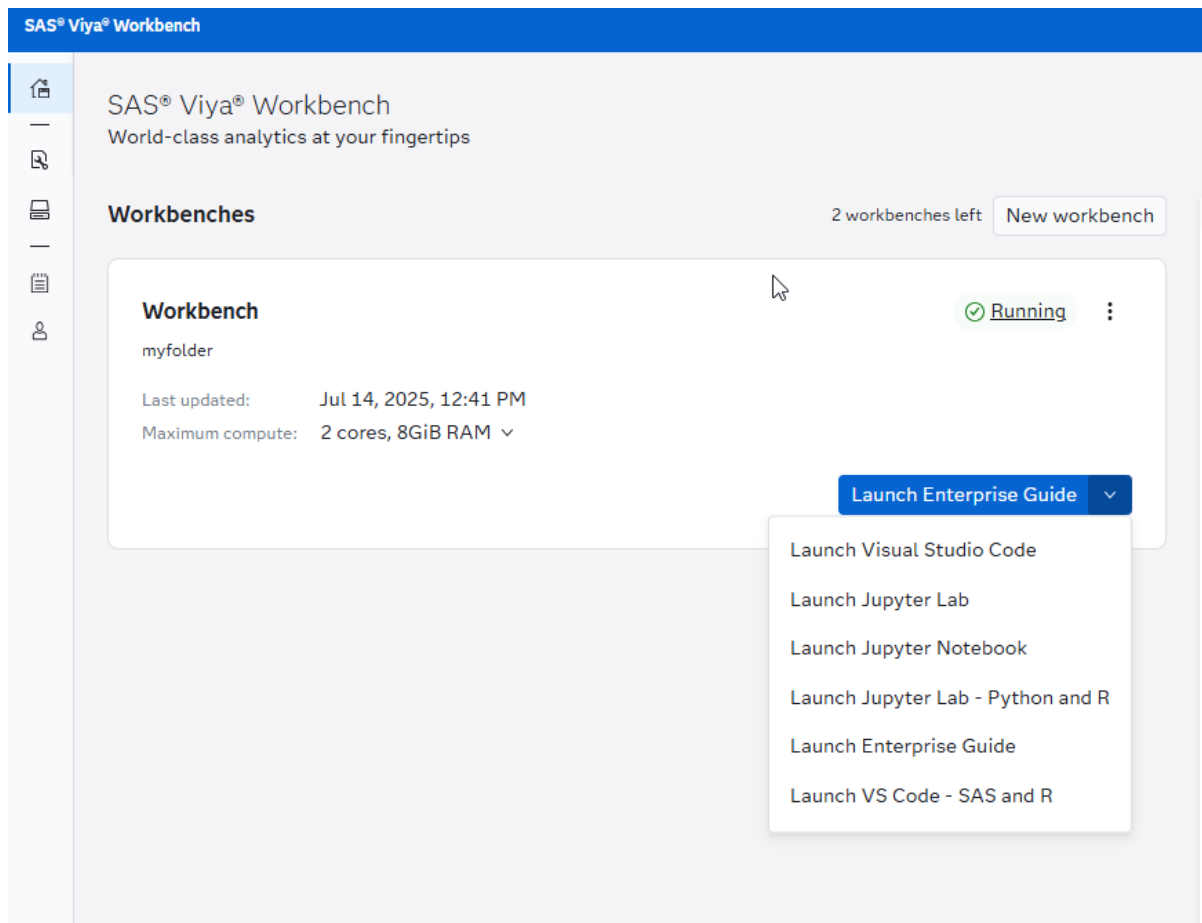


Working in SAS Enterprise Guide

<i>Open a Workbench in SAS Enterprise Guide</i>	43
<i>Accessing Your Workbench in SAS Enterprise Guide</i>	44

Open a Workbench in SAS Enterprise Guide

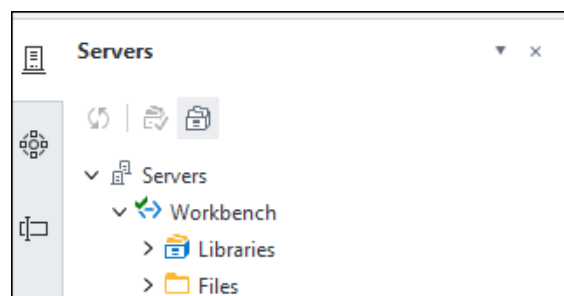
You can open the workbench from the home page. From the drop-down list, select **Launch Enterprise Guide**.



For more information, see [“Defining a SAS Viya Workbench Server Connection”](#) in *SAS Enterprise Guide: User’s Guide*.

Accessing Your Workbench in SAS Enterprise Guide

After you define a workbench connection, the workbench appears in the **Servers** tab. You can access your SAS libraries and files.



For more information about how to manage your workbench connection, see
“Defining a SAS Viya Workbench Server Connection” in *SAS Enterprise Guide: User’s
Guide*.

Working with SAS Jobs

<i>What Are SAS Jobs?</i>	47
<i>Jobs Page in SAS Viya Workbench</i>	48
<i>Create a Job Definition</i>	48
<i>Submit a Job</i>	50
<i>sas-workbench-execute Command</i>	51
Format for the sas-workbench-execute Command	51
Options for sas-workbench-execute Command	51
Examples: sas-workbench-execute Command	53
<i>Troubleshooting: SAS Jobs</i>	55
View the Job Logs	55
Why Did I Get an Error if the Job Ran?	55
What Does a Waiting Status Mean?	56

What Are SAS Jobs?

You can use jobs for web reporting, performing analytics, building web applications, and delivering content to clients such as SAS Studio. This client works with jobs that contain SAS code. These jobs can access any SAS data source or external file and create tables, files, or other data targets that are supported by SAS.

A SAS job consists of a program and its definition.

- The program contains the SAS, Python, or R code that you want to run.
- The job definition includes information such as the job name, the author, and the creation date and time.

In SAS Viya Workbench, you can create a job definition and submit existing jobs.

Jobs Page in SAS Viya Workbench

To create a job, view your existing SAS jobs, or submit a job, click .

The Jobs page contains two tabs:

- **Job Definitions** lists all the jobs that you have created or have permission to edit. You can also create a job definition from this tab.
- **Job Submissions** shows a history of your submitted jobs and the status of these jobs. From this tab, you can view the logs for your submission and resubmit a job using the parameters that were specified when you last submitted the job.

Note: Your SAS Viya Workbench administrator can view all SAS jobs and their status from these two tabs.

Create a Job Definition

- 1 To open the Jobs page, click .

- 2 On the **Job Definitions** tab, click **New**.

Note: To edit a job definition, click the name of the job definition. The same options are available. However, there is no job versioning. You can have only one job definition with the same name at a time.

- 3 Enter the name of the job definition.
- 4 In the **Command** line, enter the command to run the job.

Here are some example commands:

- `sas-workbench-execute /workspaces/myfolder/sales.sas`. For more information about the options available for this command, see [“sas-workbench-execute Command” on page 51](#).
- `python /workspaces/myfolder/sales.py`
- Here is an example of how to use bash and Python to create a job definition:

```
a=Workbench
echo "Hello $a"
```

```
echo "print('Hello from Python $a')" > /workspaces/myfolder/hello.py
python3 /workspaces/myfolder/hello.py
```

IMPORTANT You must use the full path to specify the file that contains the code for the SAS job. To determine the full path for a file, hover your pointer over the file name in the integrated development environment for your workbench.

- 5 Specify the submission settings for the job. These fields are filled with default values. You can change these settings for the job definition.

Note: Jobs have the same settings available as the workbench. Click **Copy from** to copy the settings from your workbench or existing job definition.

- From the **Maximum compute** drop-down list, select the number of cores to use. You can choose from 2, 4, 8, or 16 cores.
- From the **Primary volume** drop-down list, select the storage location. To create a volume, see [“Create a Volume” on page 9](#).

TIP The default volume is called myfolder. When you run a job, the resulting logs are stored in the selected storage. Typically, EBS storage volumes cannot be used by both a job and a workbench at the same time.

- Enter the path for the mounting folder for the workbench. This folder is automatically created in the `/workspaces/` folder.
 - Add any additional volumes that you want associated with this workbench. You can also define a home volume.
 - Select the execution environment for your job. The services and capabilities available for SAS jobs depend on your environment.
 - To use environment variables in your job definition, click **New**. In the Add Environment Variable window, enter the name of the environment variable and the value. Then click **OK**.
- 6 (Optional) Specify a schedule for job submissions. These scheduling options are available:
 - **None** - By default, no schedule is created and the job definition runs only when you submit it in SAS Viya Workbench.
 - **Simple recurrence** - You can specify whether the job runs daily or weekly and the interval for each submission. You can also specify the time of day in which the job should run.

The options in this list are based on your current time zone. For example, if your current time zone is Eastern Standard Time in the United States, then GMT-11 is your current time versus GMT minus 11 hours. If your current time is 10:00am EST and you select **GMT-11**, the job runs at 11:00pm EST.

- **Cron expression** - Use a cron expression to submit the job.

IMPORTANT Currently, SAS Viya Workbench supports only the UTC time zone for cron jobs.

- 7 Save the job definition.

Note: You must save the job definition before you can submit it.

Submit a Job

- 1 To open the Jobs page, click  .

- 2 Submit the job definition.

- On the **Job Definitions** tab, click the name of the job definition. On the page that contains the details for that job definition, click **Submit**.
- If you previously submitted a job, click **Resubmit** on the Job Submissions page.

- 3 By default, the job uses the compute resources and environment variables specified in the job definition. To use these defaults, click **OK** in the Submit Job window.

You can use the options in the Submit Job window to modify the maximum compute resources and environment variables for this specific job submission. These settings are used for this run only and are not saved to the job definition.

After you submit a job, the job status appears in the right pane. Common status messages are In Progress, Complete, or Error. In some cases, you might see a Waiting message. For more information, see [“What Does a Waiting Status Mean?” on page 56](#).

sas-workbench-execute Command

Format for the sas-workbench-execute Command

When you create a job definition, you can specify one or more `sas-workbench-execute` commands to submit SAS code to a compute server. This command creates a new session in the specified compute server and submits the program to the new session. After the command completes, the SAS log is appended to the resulting job log.

You can run the `sas-workbench-execute` command using the full path to the file that contains the SAS code or by providing the SAS code inline.

- To refer to a file, use this format: `sas-workbench-execute <options> / path/to/file_to_run.sas`. For example, `sas-workbench-execute / workspaces/myfolder/sales.sas` submits the `sales.sas` program as a job.
- To include the SAS code inline, use this format: `sas-workbench-execute <options> ' <inline sas code to run> '`. Here is an example: `sas-workbench-execute 'proc print data=sashelp.class; run;'`.

Options for sas-workbench-execute Command

Here are some additional options for the `sas-workbench-execute` command:

Table 7.1 Options for `sas-workbench-execute`

Option	Type	Required	Description
<code>--autoexec</code>	array [string]	false	Specifies the SAS code to execute during session initialization. You can specify this option multiple times to provide the content of the autoexec file or the path to the file that contains the autoexec code. For more information, see createSession on developer.sas.com .
<code>--option</code>	array [string]	false	Specifies the SAS system options to apply to the session for the SAS compute server. You can specify this option multiple times in the command line.

Option	Type	Required	Description
			For more information, see createSession on developer.sas.com .
--endpoint	array [string]	false	Specifies the URL of the SAS compute server. If the endpoint is not specified, the value of the WORKBENCH_COMPUTE_ENDPOINT environment variable is used. If neither are specified, the value of the PUP_COMPUTE_HOST is used. Note: The --endpoint and --serverId options are generally used together to run <code>sas-workbench-execute</code> against a SAS compute server that is not part of SAS Viya Workbench. You do not need to use these options if you are running the <code>sas-workbench-execute</code> command against the default compute server for SAS Viya Workbench or from the terminal in SAS Viya Workbench.
--ignore-code-server-connection-profile	array [string]	false	Specifies that the server does not use the values for --endpoint, --serverId, --option, and --autoexec that are in the <code>\$HOME/.local/share/code-server/Machine/settings.json</code> file. Values for --endpoint and --serverId specified using the command line override any values in the settings.json file. If additional --option and --autoexec values are specified on the command line, these values are appended to the values read from the settings.json file.
--serverId	array [string]	false	Specifies the ID of the SAS compute server. The default value is 0001.
--variable	array [string]	false	Specifies the SAS Macro variables to set for the SAS compute server session. For more information, see createJob on developer.sas.com.
--listing	array [string]	false	Prints the job listing to stdout after job completion.
--results	array [string]	false	Saves the job results to the specified directory after job completion.

Examples: sas-workbench-execute Command

Table 7.2 Examples: The sas-workbench-execute Command

Explanation	Example Code
Example of multiple options with inline code at the end surrounded by single quotation marks.	<pre>sas-workbench-execute --option fullstimer --option fmterr 'proc print data=sashelp.class; run;'</pre>
Example of multiple options with a SAS program instead of inline code.	<pre>sas-workbench-execute --option fullstimer --option fmterr /workspaces/myfolder/sales.sas</pre>
Example of setting an option requiring a name value pair	<pre>sas-workbench-execute --option 'validvarname any' 'data _null_; "Total Amount\$"n=12345; put "Total Amount\$"n=; run;'</pre>
Example where the --autoexec option points to a temporary file. Note: The values specified in the file run after the statements in the default autoexec.sas in <code>\$HOME/.local/share/code-server/Machine/settings.json</code> .	<pre>sas-workbench-execute --option fullstimer --option fmterr --autoexec /tmp/autoexec.sas /workspaces/myfolder/sales.sas</pre>
Example using an autoexec line rather than a file. Note: The values specified in the --autoexec option take precedence over the default autoexec.sas in <code>\$HOME/.local/</code>	<pre>sas-workbench-execute --option fullstimer --option fmterr --autoexec 'options pagesize=80;' 'proc options; run;'</pre>

Explanation	Example Code
<code>share/code-server/Machine/settings.json</code> .	
Example using the <code>--variable</code> option to assign SAS macro variables.	<pre>sas-workbench-execute --variable myvar='Macro var set from cmd line' --variable myvar2='and another' 'data _null_; put "&myvar &myvar2"; run;'</pre>
Example of specifying a unique SAS results folder for a job run.	<pre>dt='date +%Y%m%d%H%M%S' sas-workbench-execute --results /workspaces/myfolder/ job_results_\${dt} 'proc print data=sashelp.sales; run;'</pre>
Example of including SAS results listing after the log output.	<pre>sas-workbench-execute --option fullstimer --listing 'proc print data=sashelp.sales; run;'</pre>
Multi-line example using bash and <code>sas-workbench-execute</code> .	<pre>sas-workbench-execute 'proc options; run;' echo 'STARTING JOB1.SAS' sas-workbench-execute /workspaces/myfolder/job1.sas echo 'STARTING JOB2.SAS' sas-workbench-execute /workspaces/myfolder/job2.sas</pre>
Note: Each SAS program runs in a different SAS server session, so these programs do not share temporary resources, such as SAS Work and variable. These resources are shared if the programs run interactively in the same SAS Viya Workbench session.	
Alternative example so that you can run both SAS programs in the same SAS server session.	Note: This is an alternative to the multi-line example using bash and <code>sas-workbench-execute</code>. The first code block is a modification of the <code>job1.sas</code> program and references the <code>job2.sas</code> program in the same SAS Viya Workbench session. <pre>/* Use %include to call job2.sas at the end of job1.sas */ %global job1; %let job1='This is job1'; data _null_; put "&job1";</pre>

Explanation	Example Code
	<pre>put "&SYS_COMPUTE_SESSION_ID"; run; %include "&WORKSPACE_PATH/job2.sas";</pre> Here is the code to create the job definition: <pre>/* Now run job1.sas as part of the job run */ sas-workbench-execute 'proc options; run;' echo 'STARTING JOB1.SAS' sas-workbench-execute /workspaces/myfolder/job1.sas</pre>
Example using the <code>--ignore-code-server-connection-profile</code> which means that the command disregards the settings in <code>\$HOME/.local/share/code-server/Machine/settings.json</code>.	<pre>sas-workbench-execute --option fullstimer --ignore-code-server-connection-profile 'proc options; run;'</pre>

Troubleshooting: SAS Jobs

View the Job Logs

To view the log for a SAS job, open the **Job Submission** tab. For the specific job, click **View Logs**. You can also write the logs as text to the storage location used by the job.

Why Did I Get an Error if the Job Ran?

An error state can appear if a job process exists with a nonzero exit code.

What Does a Waiting Status Mean?

When you submit a job, SAS Viya Workbench displays the job status (for example, In Progress, Complete, or Error). If you see a status of Waiting, the workbench and job definition are competing for resources. To run the SAS job, first stop the current workbench.

Appendix 1

Security Considerations

<i>Uploading and Downloading Data</i>	57
<i>Storing User Credentials</i>	57

Uploading and Downloading Data

While working with SAS Viya Workbench, the upload and download functionality of your files is provided by your integrated development environment. SAS Viya Workbench does not have any restrictions on the file type or file size. SAS Viya Workbench does not enforce any speed limits. Remember to follow the security guidelines designed by your company.

Storing User Credentials

With SAS Viya Workbench, users can store their database credentials in a file.

- User access to an external relational database requires that the user credentials are stored in a file. The AUTHDOMAIN= option is not supported.
- Credentials can be stored in the user's home directory (autoexec file, INI file) or in a LIBNAME statement in a program. For more information, see [“LIBNAME Statement Examples for PostgreSQL” in Accessing External Data Sources with SAS®Viya Workbench](#).
- SAS Viya Workbench does not provide a warning or error if your credentials are not encrypted.

When working with SAS Viya Workbench, remember to follow the security guidelines for your company.

Appendix 2

Troubleshooting Tips and Frequently Asked Questions

<i>What Are the Differences between SAS Viya Workbench and SAS Viya Platform?</i>	59
<i>Does SAS Viya Workbench Support SASPy?</i>	60
<i>What Happens if the Workbench Exceeds the Allocated Quota Storage?</i>	60
<i>How to Reconnect after SAS Viya Workbench Times Out</i>	61
<i>Troubleshooting: Enabling Third-Party Cookies for SAS Viya Workbench</i>	62
Error Loading Webview	62
Chrome: Updating Your Third-Party Cookie Allowlist	63
Mozilla Firefox: Updating Your Third-Party Cookie Allowlist	63
Microsoft Edge: Updating Your Third-Party Cookie Allowlist	63
<i>Troubleshooting: Possible Error When Using Both the SWAT and SASVIYA Packages</i>	64

What Are the Differences between SAS Viya Workbench and SAS Viya Platform?

Here are some notable differences between SAS Viya Workbench and SAS Viya platform:

- A libref enables data source connections instead of a caslib.
- The PRESERVE_COL_NAMES=YES option is required to enable SAS Visual Analytics and SAS Machine Learning procedures to write to Oracle. This option can be specified as LIBNAME option or a data set option. PRESERVE_COL_NAMES=YES is optional when writing to Oracle with other procedures.

- [“Amazon Redshift: Functional Differences between SAS Viya Workbench and SAS Viya” in *Accessing External Data Sources with SAS®Viya Workbench*](#)
- [“Oracle: Functional Differences between SAS Viya Workbench and SAS Viya Platform” in *Accessing External Data Sources with SAS®Viya Workbench*](#)
- [“PostgreSQL: Functional Differences between SAS Viya Workbench and SAS Viya Platform” in *Accessing External Data Sources with SAS®Viya Workbench*](#)
- [“Snowflake: Functional Differences between SAS Viya Workbench and SAS Viya Platform” in *Accessing External Data Sources with SAS®Viya Workbench*](#)
- [“Microsoft SQL Server: Functional Differences between SAS Viya Workbench and SAS Viya Platform” in *Accessing External Data Sources with SAS®Viya Workbench*](#)

Does SAS Viya Workbench Support SASPy?

SASPy is not installed with SAS Viya Workbench. You can install SASPy; however, this functionality has not been tested, and SAS Viya Workbench does not pre-configure anything for use with SASPy.

What Happens if the Workbench Exceeds the Allocated Quota Storage?

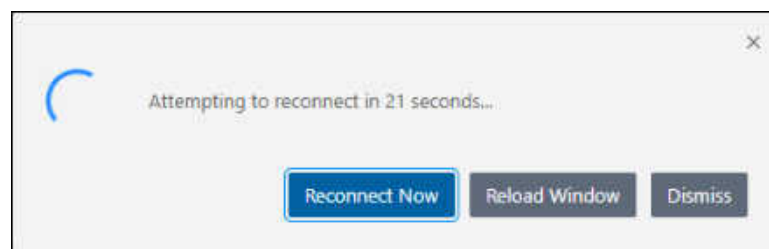
By default, each storage location has 5 GB of storage. You can change this value when you create the storage location. You can always increase the amount of storage but you cannot decrease it.

If you exceed this amount of storage, the workbench automatically restarts. If your workbench restarts, refresh the SAS Viya Workbench home page.

How to Reconnect after SAS Viya Workbench Times Out

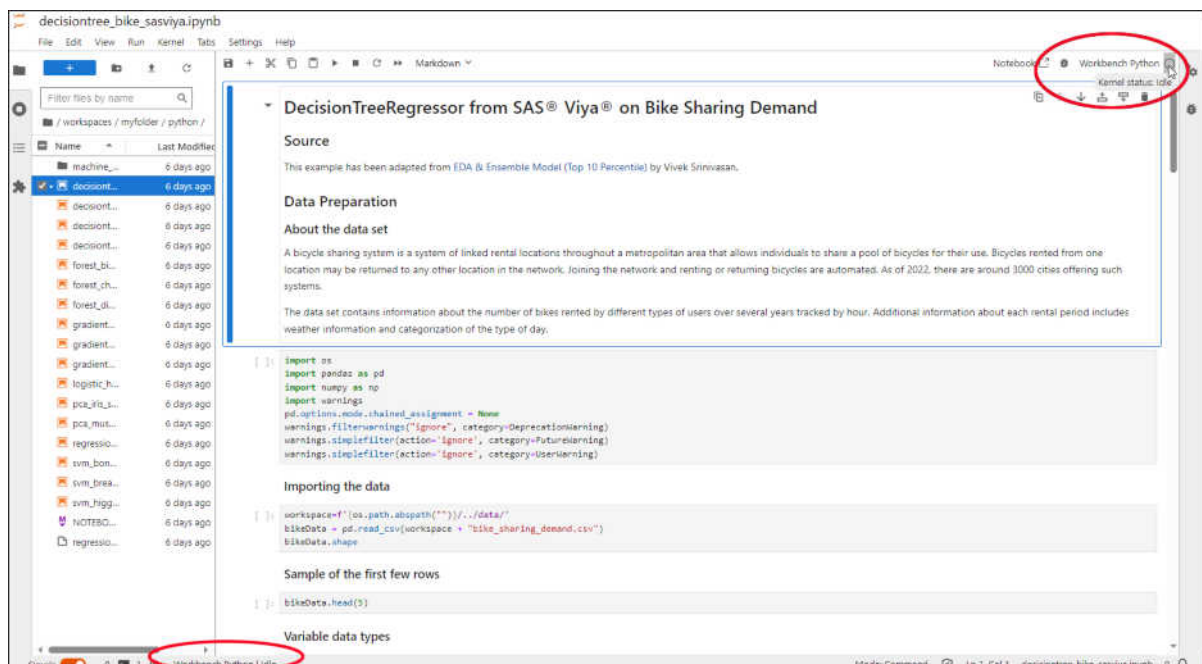
SAS Viya Workbench times out after 30 minutes of inactivity.

If you are using SAS Viya Workbench in VS Code, the following reconnect window appears after 30 minutes.

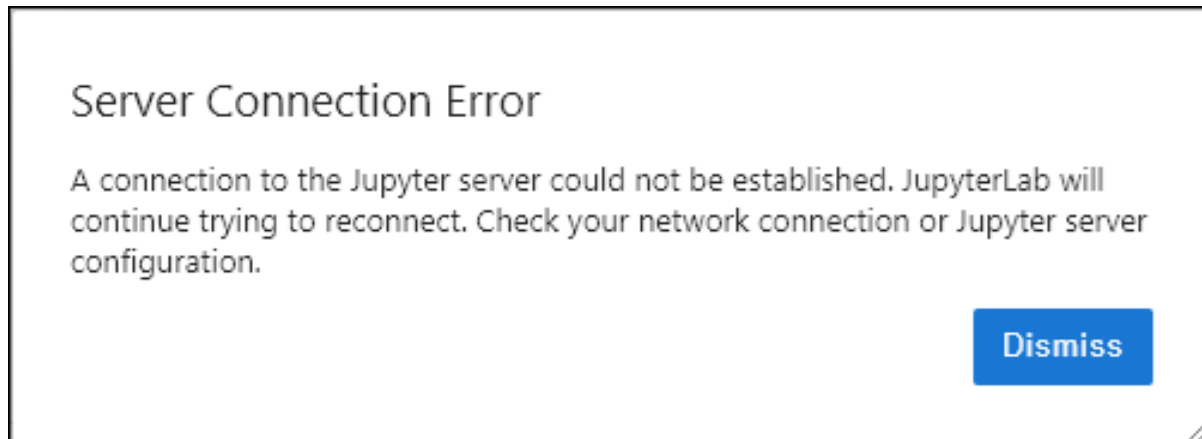


Do not click **Reconnect Now**. Instead, close the browser tab that contains SAS Viya Workbench and use the URL provided by SAS to reopen SAS Viya Workbench.

If you are using SAS Viya Workbench in Jupyter Notebook or Jupyter Lab, an idle status might appear in the workspace.



You might also see a server connection error after 30 minutes in Jupyter.

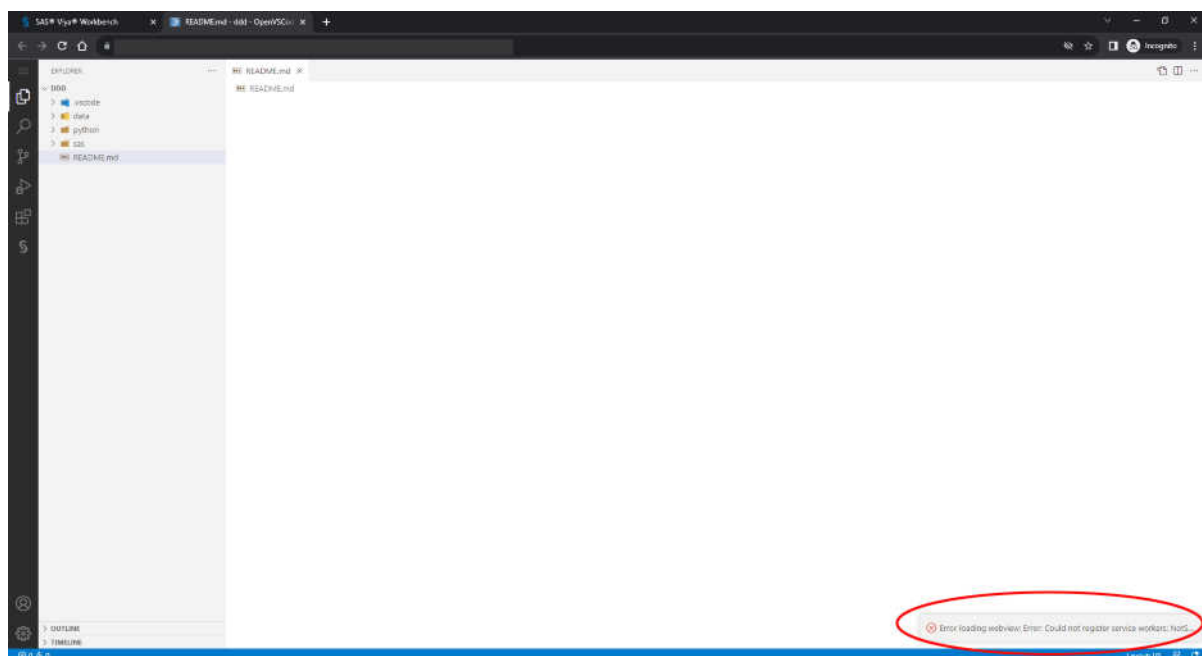


To reopen SAS Viya Workbench, dismiss this message and refresh your browser window.

Troubleshooting: Enabling Third-Party Cookies for SAS Viya Workbench

Error Loading Webview

If you have blocked third-party cookies in your browser or you are using an incognito window, you might see an error when you open a workbench in VS Code.



To resolve this issue, you need to enable third-party cookies from SAS Viya Workbench.

Chrome: Updating Your Third-Party Cookie Allowlist

Note: The option names might differ depending on the version of your browser.

- 1 In Chrome, open your settings.
- 2 In the left navigation pane, click **Privacy and security**.
- 3 Expand the section labeled **Third-party cookies**.
- 4 Click **Add** next to **Sites that always use cookies**.
- 5 In the **Site** field, add the URL for SAS Viya Workbench. Click **Add**.

Mozilla Firefox: Updating Your Third-Party Cookie Allowlist

Note: The option names might differ depending on the version of your browser.

- 1 In a Firefox browser, open your settings.
- 2 In the left navigation pane, click **Privacy & Security**.
- 3 Next to the **Enhanced Tracking Protection** heading, click **Manage Exceptions**.
- 4 In the **Address of website** field, enter the URL for SAS Viya Workbench. Click **Add Exception**.
- 5 Click **Save Changes**.

Microsoft Edge: Updating Your Third-Party Cookie Allowlist

- 1 In an Edge browser, open your settings.
- 2 In the left navigation pane, click **Privacy, search, and services**.

- 3 Under the **Tracking prevention** heading, click **Exceptions**.
- 4 On the **Tracking prevention exceptions** page, click **Add a site**.
- 5 In the **Site** field, enter the URL for SAS Viya Workbench. Click **Add**.

Troubleshooting: Possible Error When Using Both the SWAT and SASVIYA Packages

When using the SAS APIs for coding in Python, you might see an error when using both the SAS Scripting Wrapper for Analytics Transfer (SWAT) and SASVIYA package for Python. (The SWAT package is used to communicate with a SAS CAS server.) This error occurs because SWAT and SASVIYA use an overlapping set of internal settings, and SWAT needs to access those settings before SASVIYA is initialized.

To fix this error, import the SWAT package before you import the SASVIYA package.

Appendix 3

Third-Party Licenses and Information

<i>Overview</i>	65
<i>Jupyter</i>	66
<i>Jupyter Lab</i>	66
<i>OpenVSCode Server</i>	67
<i>Matplotlib</i>	68
<i>scikit-learn</i>	68
<i>seaborn</i>	69
<i>Teradata</i>	70

Overview

Using SAS Viya Workbench requires access to some third-party software.

This section contains information about any third-party code included with your SAS software, including applicable third-party software notices and/or additional terms and conditions. Additional third-party information is available at [Third-Party Notices and Information for SAS Viya](#).

Jupyter

On the web: <https://github.com/jupyter/notebook/blob/main/LICENSE>

BSD 3-Clause License

- Copyright (c) 2001-2015, IPython Development Team
- Copyright (c) 2015-, Jupyter Development Team

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1 Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- 2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3 Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Jupyter Lab

On the web: <https://github.com/jupyterlab/jupyterlab/blob/main/LICENSE>

Copyright (c) 2015-2022 Project Jupyter Contributors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1 Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- 2 Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3 Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Semver File License

=====

The semver.py file is from <https://github.com/podhmo/python-semver>, which is licensed under the "MIT" license. See the semver.py file for details.

OpenVSCode Server

On the web: <https://github.com/gitpod-io/openvscode-server/blob/main/LICENSE.txt>

MIT License

Copyright (c) 2015 - present Microsoft Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Matplotlib

On the web: <https://matplotlib.org/stable/users/project/license.html>

Matplotlib uses only BSD compatible code, and its license is based on the PSF license.

scikit-learn

On the web: <https://github.com/scikit-learn/scikit-learn/blob/main/COPYING>

BSD 3-Clause License

Copyright (c) 2007-2023 The scikit-learn developers.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

seaborn

On the web: <https://github.com/mwaskom/seaborn/blob/master/LICENSE.md>

Copyright (c) 2012-2021, Michael L. Waskom All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation, and/or other materials provided with the distribution.

Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Teradata

License Information for ICU 65.1

COPYRIGHT AND PERMISSION NOTICE (ICU 58 and later)

Copyright © 1991-2023 Unicode, Inc. All rights reserved.

Distributed under the Terms of Use in <https://www.unicode.org/copyright.html>.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that either (a) this copyright and permission notice appear with all copies of the Data Files or Software, or (b) this copyright and permission notice appear in associated Documentation.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.