

Advanced RPA Design and Development

v4.0



Lesson 16

Version Control

Use the Studio Git Integration to

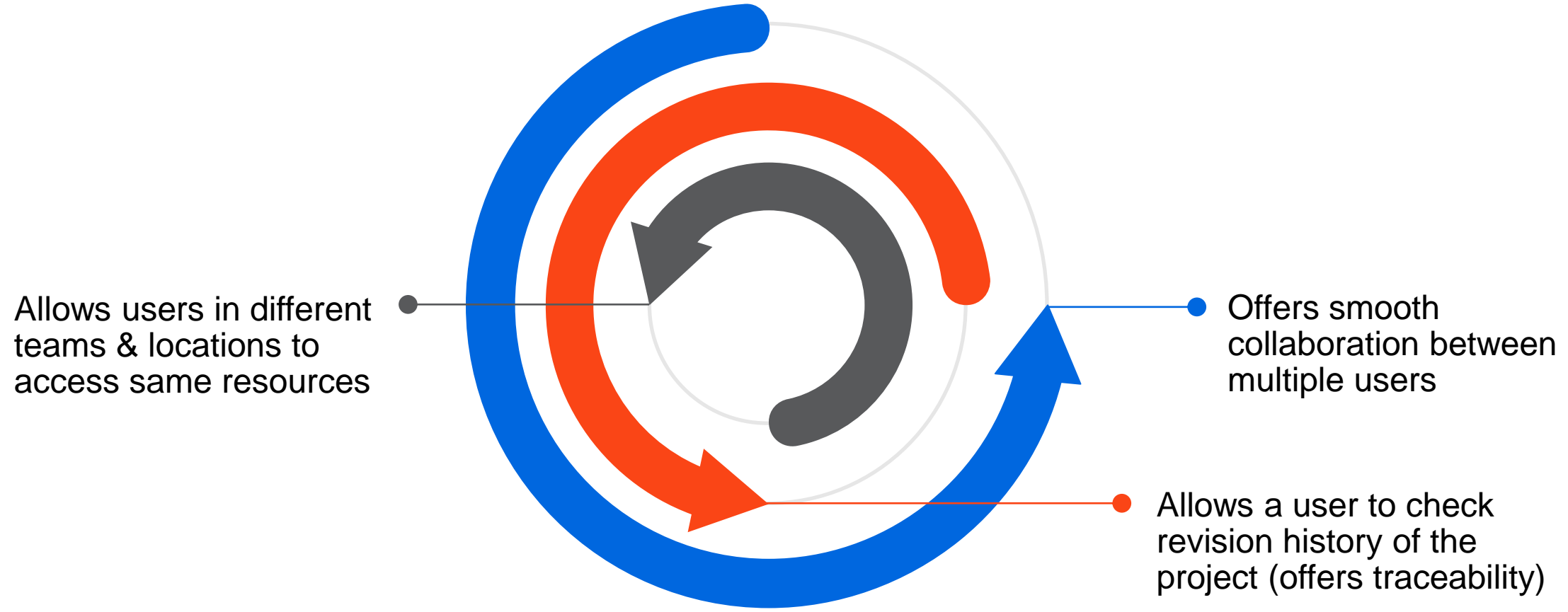
1. Add a project
2. Clone a repository
3. Commit
4. Push
5. Use show changes and solve conflicts
6. Create and Manage branches

Learning Objectives

1. Define version control system (VCS) and its purpose
2. Analyze the benefits and challenges of using version control systems
3. Describe how Git works
4. Identify the difference between Git and SVN
5. Create a GitHub Repository
6. Use the Git Init option to add a project to the new local Git repository
7. Identify how to clone a repository and commit changes to a local Git Repository
8. Identify how to change the last commit
9. Identify how to undo changes and push the final changes to the remote repository

Introduction to Version Control

Version control is used to track and manage any changes in the project.



Automation projects are connected to version control systems through the Team tab in Studio. The version control systems are:

GIT

GIT is a distributed version-control system for tracking changes in source code during software development.

- **Clone Repository:** Clone a remote GIT repository
- **Copy to GIT:** Copy the current project to an existing GIT repository
- **GIT Init:** Add the current project to a new local GIT repository
- **Disconnect:** Disconnect the current project from source control
- **Change Signature:** Change commit signature

TFS

TFS is the source code management established by Microsoft, used for the project and release management.

- **Open from TFS:** Open a project from a TFS repository
- **Add to TFS:** Add the current project to TFS source control
- **Manage TFS Online:** Go to the web management interface
- **Disconnect:** Disconnect the current project from source control

SVN

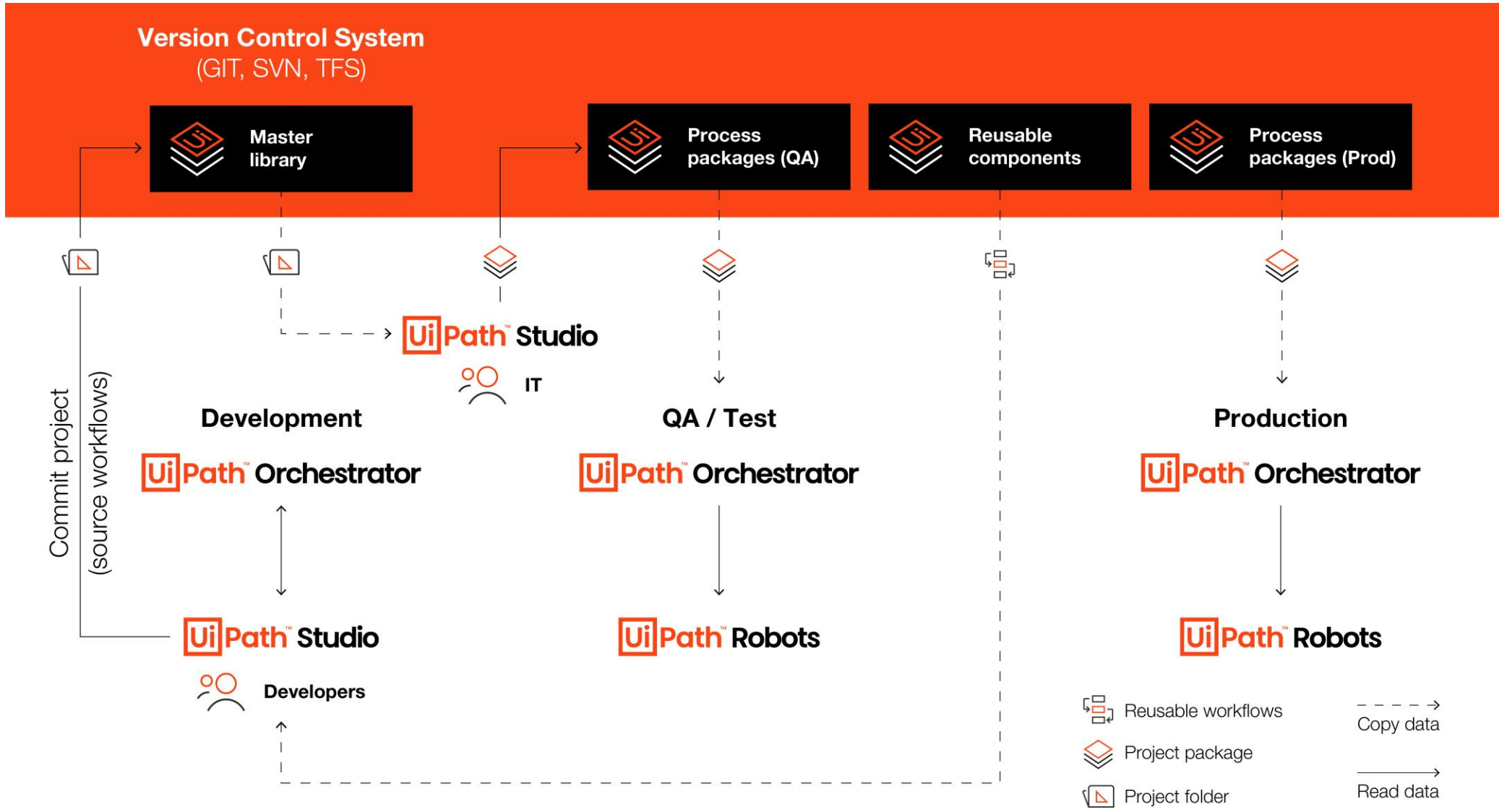
SVN is a software versioning and revision control system distributed as open source.

- **Open from SVN:** Open a project from an SVN repository
- **Add to SVN:** Add the current project to SVN source control
- **Disconnect:** Disconnect the current project from source control
- **Change Credentials:** Change remote repository credentials

What are version control systems?

1. A version control system (VCS), also known as a source control system or revision control system, is a **software tool that helps developers manage changes to their source code and other files.**
2. It provides a structured approach to
 - Track
 - Document
 - Control different versions of files
 - Enabling collaboration
 - Facilitating team workflows
 - Ensuring code integrity
3. UiPath projects are typically developed using external version control systems to manage source code and track changes over time.
4. You can use any version control system that suits your requirements, such as Git, SVN (Subversion), or TFS (Team Foundation Server)
5. UiPath projects are usually stored as collections of files, including workflow files (.xaml), configuration files, and other supporting files.

What are version control systems?



Benefits of Version Control System

History and Version Tracking

Collaboration and Teamwork

Branching and Merging

Code Integrity and Backup

Traceability and Auditing

Experimentation and Rollbacks

Code Reviews and Continuous
Integration

History and Version Tracking

Version control systems keep a complete history of all changes made to files, allowing developers to track and view the evolution of the codebase

Collaboration and Teamwork

Version control systems facilitate collaboration among team members working on the same project.

Multiple developers can work on different branches, make changes, and merge them back into the main codebase

Branching and Merging

Allow developers to create branches, which are independent lines of development. Branches enable the isolation of new features or experimental changes from the main codebase, providing a safe space for development without affecting the stability of the main branch. Branches can be merged back into the main branch once the changes are tested and approved

Code Integrity and Backup

Maintain code integrity by documenting, reviewing, and validating changes before integration.

They minimize the risk of errors and serve as backups by storing the project's complete history for easy reversion to previous versions

Traceability and Auditing

Provide traceability by associating each change with relevant information, such as the reason for the change, associated issues or tickets, and the person responsible.

This traceability aids in auditing, compliance, and helps in understanding the context behind specific modifications

Experimentation and Rollbacks

Developers can experiment with new ideas, features, or improvements without fear of damaging the existing codebase.

If an experiment doesn't work out as expected, it's easy to roll back to a previous known good state and continue from there

Code Reviews & Continuous Integration

Seamlessly integrate with code review and continuous integration/delivery tools.

They facilitate systematic code reviews, improving code quality and knowledge sharing.

By setting up continuous integration workflows, code changes can be automatically built, tested, and deployed upon each commit, streamlining the development process

What are version control systems?

Irrespective of the version control system type, project files are stored on a server where you upload your completed work from your local machine.

The choice between a centralized version control system like SVN and a distributed version control system like Git impacts the process of committing changes.

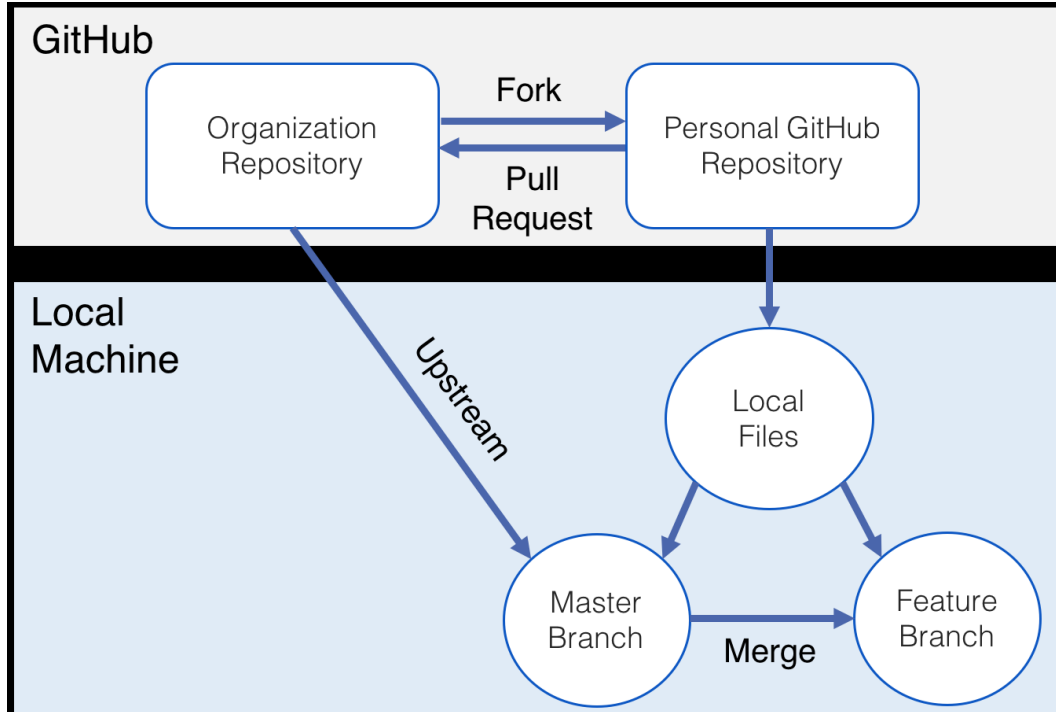
- Git, TFS, and SVN are the version control systems that integrate with UiPath Studio. Users establish the connection to a **version control system at the project level**
- To manage your connections, access Studio, go to the **Backstage view**, and click on the **Team** tab
- The **Add to Source Control button** in the status bar offers shortcuts to
 - Git Init
 - Copy to Git
 - Add to TFS
 - Add to SVN

Note: You cannot connect a project to Git, TFS and SVN at the same time.

Resources

Topic	Link
About Version Control	https://docs.uipath.com/studio/standalone/2022.10/user-guide/about-version-control
Managing Projects with Git	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Managing Projects with SVN	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-svn
Managing Projects with TFS	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-tfs
GitHub - Official Documentation	https://help.github.com/en

Git Overview



Git is an open-source distributed version control system, empowers users to efficiently and collaboratively manage source code and project files.

It actively tracks changes to files, enabling easy switching between different versions, branch creation for experimentation, merging of changes, and seamless collaboration.

Git facilitates both local and remote work, making it convenient for individual developers and teams alike.

It offers numerous benefits, including version history, streamlined rollback, conflict resolution, and seamless integration with popular development platforms.

Its flexibility, speed, and robustness have established Git as the de facto standard for version control in the software development industry.

Additionally, Git provides excellent support for branching, merging, and repository history rewriting, with the added advantage of the widely-used pull request feature that facilitates efficient code review and collaboration within teams.

As the most widely adopted version control system globally, Git has solidified its position as the modern standard in software development.

Here is a basic overview of how Git works:

1. Create a repository (project) with a Git hosting tool (e.g. GitHub)
2. Copy or clone the repository to your local machine
3. Add a file to your local repository and commit (save) the changes locally
4. Push your changes to the remote repository
5. Pull the file to local repository and make changes, and then commit and push the file
6. Create a branch (alternative), make a change, commit the change
7. Open a pull request (propose changes to the master branch)
8. Merge your branch to the master branch

Comparing SVN and Git Side-by-Side

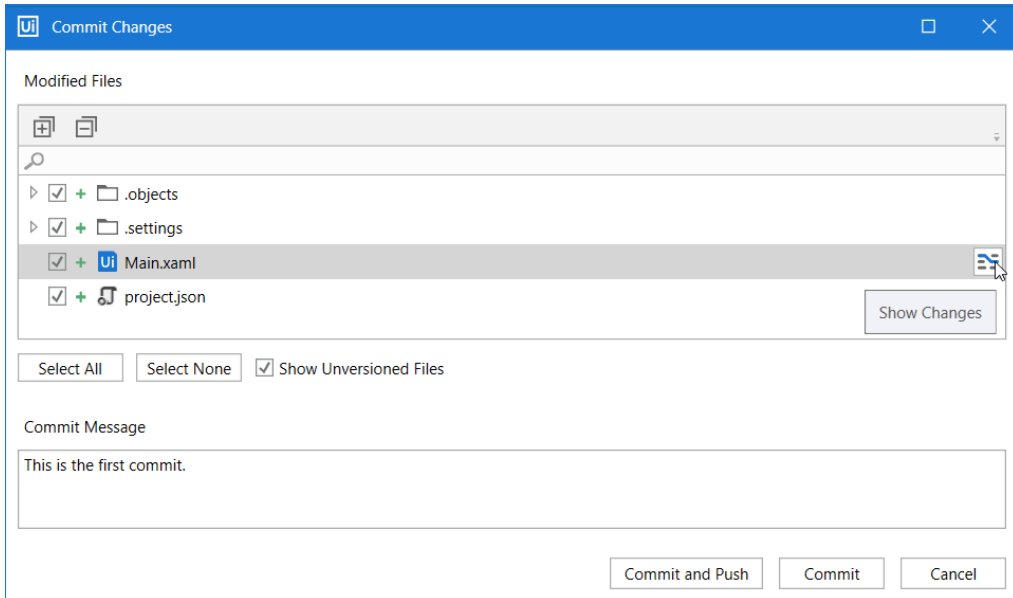
SVN (Subversion) and Git are both popular version control systems used in software development. While they serve similar purposes, they differ in their underlying architecture, workflow models, and features. Here's a comparison of SVN and Git:

	SVN	GIT
Architecture	SVN is a centralized version control system, where a central repository stores the entire history and versions of files. Developers typically check out a working copy from the central repository, make changes, and commit them back.	Git is a distributed version control system, where each developer has a complete local copy of the repository. This allows developers to work independently and commit changes locally before synchronizing them with remote repositories.
Branching and Merging	SVN supports branching and merging, but it follows a copy-modify-merge approach. Each branch is essentially a copy of the trunk or another branch, and developers merge changes back to the main branch manually.	Git provides powerful branching and merging capabilities. Branches are lightweight and can be created and merged easily. Git allows for various branching strategies (such as feature branches, release branches, etc.) and provides efficient merging mechanisms.
Performance	SVN performs well for smaller repositories and projects with fewer files. However, as the repository size grows, SVN's centralized architecture can lead to slower operations, especially for tasks like history traversal.	Git is designed to handle large repositories efficiently, thanks to its distributed nature. Most operations are performed locally, making Git faster for common tasks such as committing, branching, merging, and switching between branches.
Offline Work	SVN requires a network connection to the central repository for most operations. Working offline is limited, as you need network access to commit changes or access the repository's full history.	Git enables full offline work since each developer has a complete local repository copy. Developers can commit changes and access the repository's history without network connectivity.
Community and Ecosystem	SVN has a mature user base and a well-established ecosystem. It has been around for a longer time, and there are many tools and integrations available for SVN.	Git has gained immense popularity and has a larger user community. It has a vast ecosystem with a wide range of tools, services, and integrations built around it.

Using Git in Studio – Add a Project

1. Sign-in to GitHub (www.github.com)
2. Click Create repository
3. Copy the repository URL
4. Go UiPath Studio Home tab and select Team
5. **GIT Init** to add the current project to a new local GIT repository
6. Choose a location for our new local GIT repository
7. The Commit Changes window is now displayed, and we're prompted to select the project files that we want to add or that have suffered changes since our last commit. As this is our first commit, we can select all, type a commit message, and click Commit and Push to add all the folders along with any changes made to the remote repository on GitHub as well.
8. Since we aren't adding the project and changes to just our local GIT repository, but also to the remote one, we are now prompted to add our remote repository. Let's type in the name and paste in the URL from GitHub.
9. Click Add and Save

Add a Project to GIT

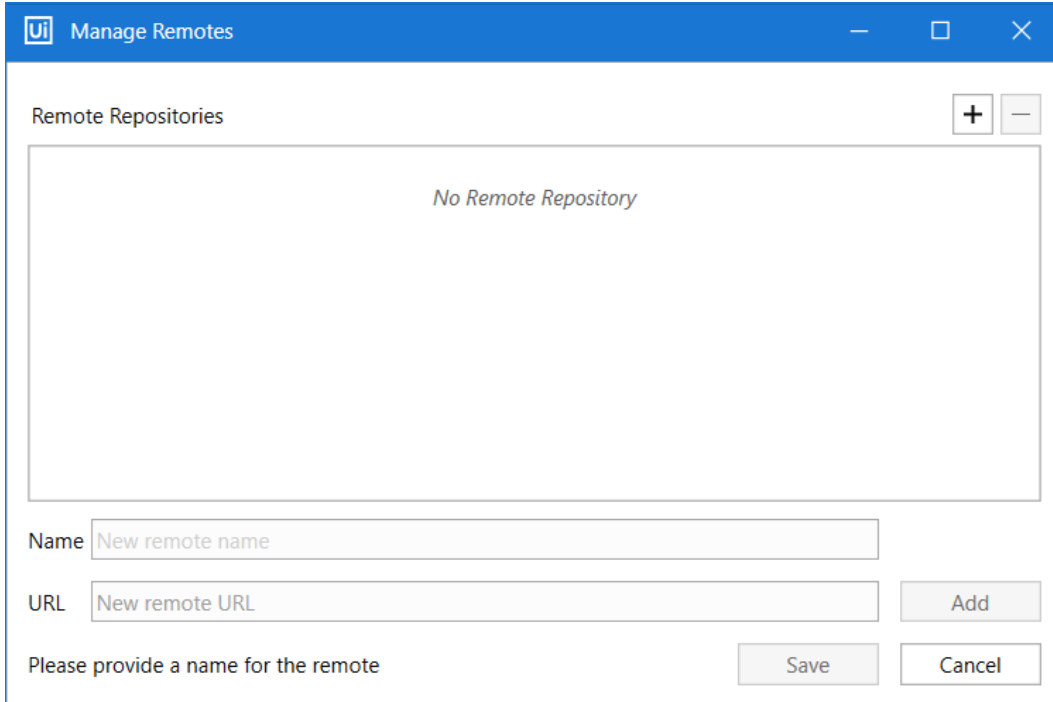


When a project is added to GIT, the context menu in the **Project** panel includes GIT-specific options. For more information, see [Context Menu Options for GIT](#).

The **GIT Init** feature adds the current project to a local GIT repository. Access the command from the **Team** tab, or the status bar.

1. Create or open a project in Studio. Click the **Start** tab > **Team**. The **Team** tab is displayed.
2. Click the **GIT Init** button, and then select a path where the repository should be initialized. The location may be the same as the project or the parent folder. The **Commit changes** window opens.
3. The Modified Files section shows the project's files that are to be added to the Git repo. Clear the box next to the ones that you don't want to add or use Select All, Select None.
4. Select the Show Unversioned Files box to add unversioned files to the list.
5. Write a Commit Message.
6. Click the Commit button to commit the changes to the local Git repository.

Committing and Pushing to GIT

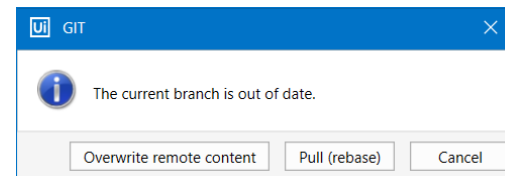


1. From the same Commit Changes window, click the Commit and Push button to commit the changes and push them to the remote repository. This Manage Remotes window is displayed. The window is also available from the status bar

2. In the **Name** section, add the name of the remote repository

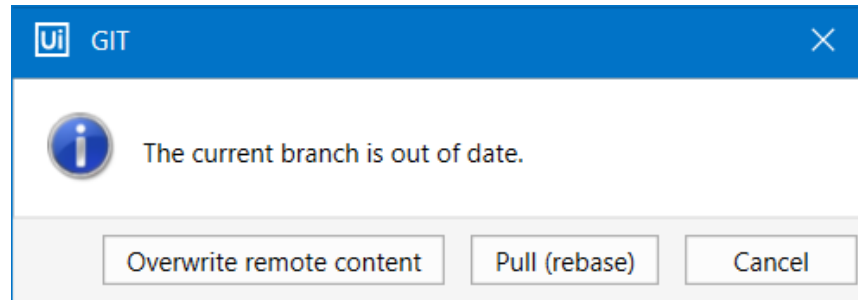
3. In the **URL** section, add the remote URL

If you want to make modifications to the added repositories, simply click an entry, change the name and URL, then click the **Update** button. When you're done click **Add**, then **Save**. The below message box opens. This means that the local repository is not synchronized with the remote one.



Committing and Pushing to GIT

If you want to make modifications to the added repositories, simply click an entry, change the name and URL, then click the **Update** button. When you're done click **Add**, then **Save**. The below message box opens. This means that the local repository is not synchronized with the remote one.

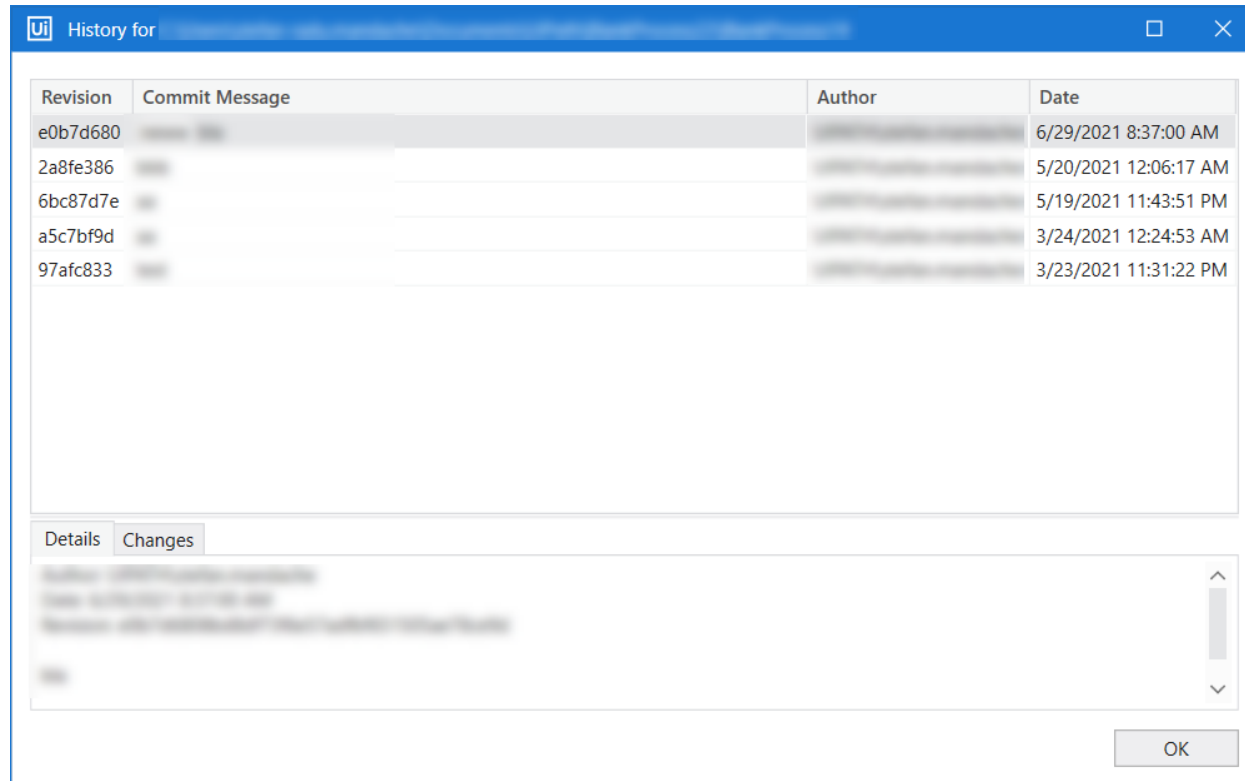


- Click the Overwrite remote content button to push the local versions of files to the remote repository and overwrite the files there
- Click the Pull (rebase) button to pull the remote files and rebase the current branch
- Click the Cancel button to discard the whole operation

The number of unpushed changes, and newly added files are visible in the status bar. Click the icon to open the Commit Changes window, or the icon to push changes

Note: If you edit a file from a project added to source control in an external editor, the change is visible in the Project panel and the status bar only after you click Refresh in the Project panel

Viewing the Commit History



To view the commit history for a project or for a specific file or folder in a project

- Right-click the project node, a file, or folder in the Project panel
- Select Show History

This opens the History window which displays a list of existing revisions for the selected file, folder, or project.

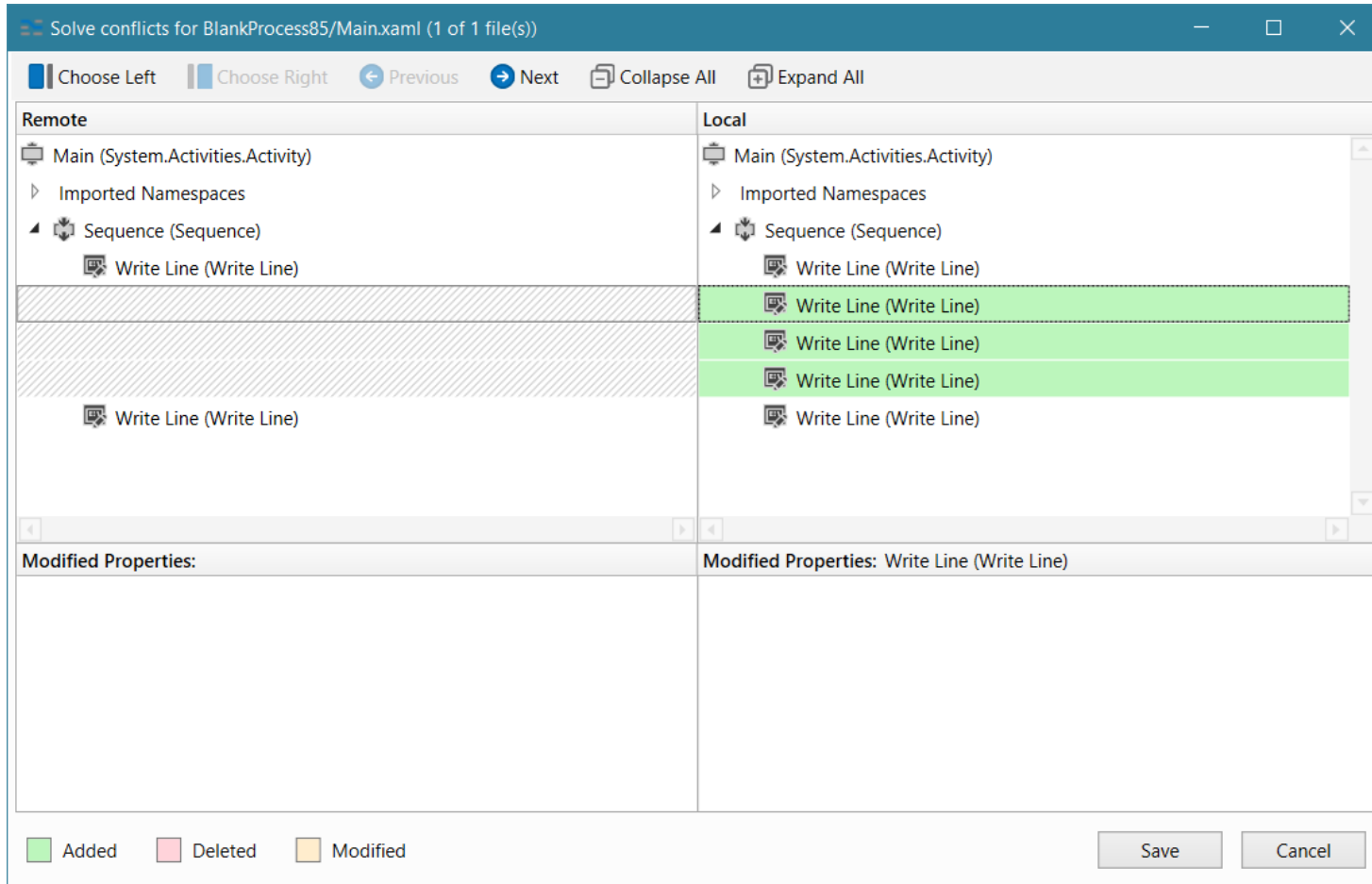
For each commit, the commit hash, message, author, and date are displayed in a table on the upper part of the window. You can view more information about a selected commit in the Details and Changes tabs on the lower part of the window

Comparing Two Versions of a File

To compare two versions of the same file:

- If you opened the history for a file, right-click a commit in the History window, and then select **Compare with Previous, Compare with Local, or Compare with Latest**
- If you opened the history for a folder or project, select a commit in the History window, and then, in the Changes tab, double-click a file to compare it with its previous version.

Solving Conflicts



GIT integration with Studio comes with a feature for solving conflicts that may occur when performing the Rebase or Push command, found in the Commit Changes window.

Whenever Studio detects a conflict between the local file and the one found in the remote repository, the Solve conflicts window is displayed.

The window is similar to [File Diff](#), showing the differences between the **Remote** version of the file and the **Local** version.

Solving Conflicts






The following table describes the options available in the **Solve conflicts** window.

Option	Description
Choose Left	Select the left file representing the file in the remote repository to push.
Choose Right	Select the right file representing the file in the local repository to push.
Save	Click Save after choosing the left or right file.
Cancel	Cancel the operation and exit the Solve conflicts window.
Previous	Navigates to the previous change in the compared files.
Next	Navigates to the next change in the compared files.
Collapse All	Collapses all nodes in the <code>.xaml</code> files.
Expand All	Expands all nodes in the <code>.xaml</code> files.

Using Git in Studio

acme login - UiPath Studio Community

GIT

-  **Clone Repository**
Clone a remote GIT repository.
-  **Copy to GIT**
Copy the current project to an existing GIT repository.
-  **GIT Init**
Add the current project to a new local GIT repository. The repository location can be the same as the project folder or a parent folder.
-  **Disconnect**
Disconnect current project from source control.
-  **Change signature**
Change commit signature

Once you have connected to Git (can be just a local repository)

These are the available options under the BackStage Team tab



Open

Close

Start

Tools

Templates

Team

Settings

Help

Authenticating to GIT

Authentication methods in Studio differ in accordance with the methods used for cloning a GIT repository, either

- HTTPS
- SSH

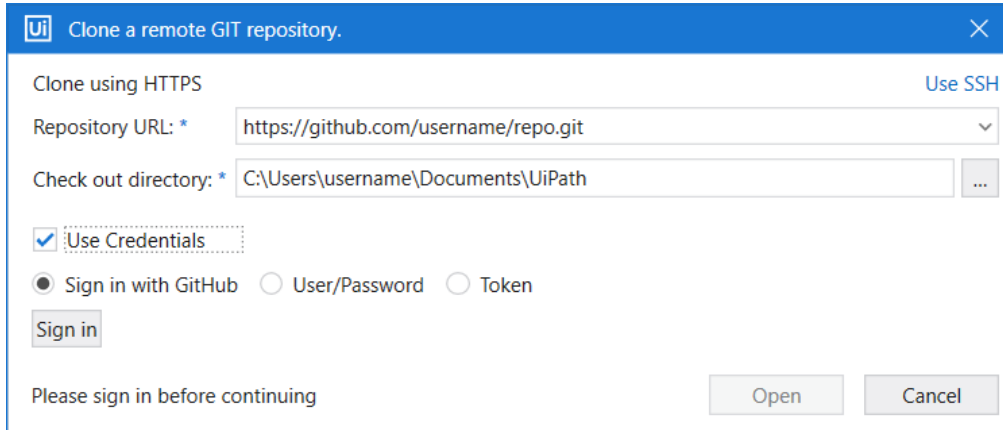
Note:

- The GIT credentials you provide in Studio are stored in the Windows Credential Manager
- The GIT integration with Studio currently **supports two-factor authentication only for GitHub if you authenticate by signing in to the service**. For other tools, use the SSO authentication method with a personal token, or the basic access authentication method.

We detail the steps for authenticating to a GitHub repository, but the Git integration in Studio is not limited to just this service

Authenticating to GIT over HTTPS

When cloning a remote GIT repository or copying the current project to an existing GIT repository using HTTPS for the first time, you must provide your GIT credentials. These credentials must be entered in the Use Credentials fields:



UiPath Clone a remote GIT repository. X

Clone using HTTPS Use SSH

Repository URL: *

Check out directory: *

Use Credentials

Sign in with GitHub User/Password Token

Please sign in before continuing

You can authenticate using the following options:

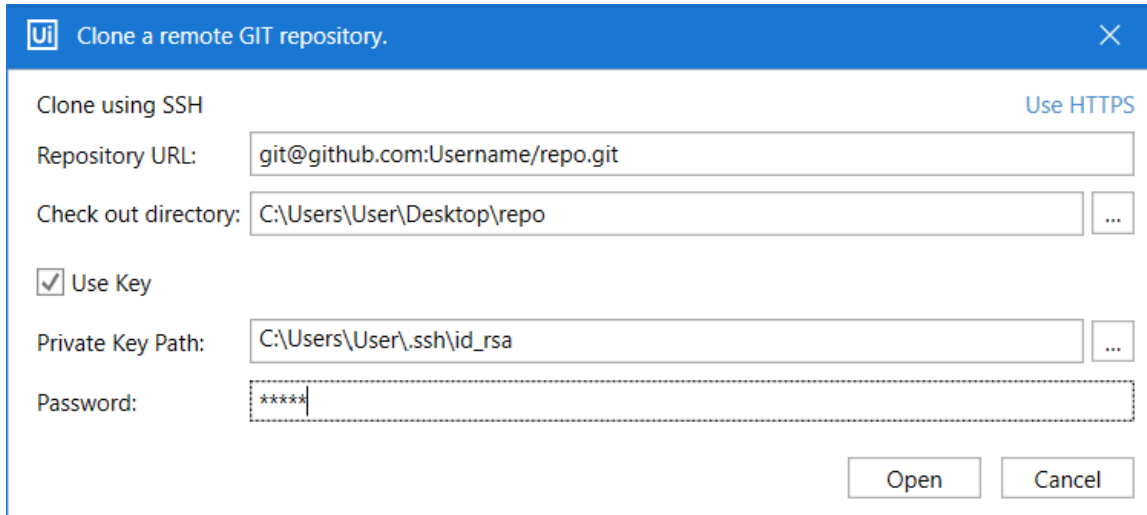
- **Sign in with GitHub** - Sign in with your GitHub account.
- **User/Password** - Enter your user and password
- **Token** - Enter your user and personal access token

Follow the steps to generate a GIT token for your GitHub repository

Important: The **Sign in with GitHub** option is available only for repositories hosted on github.com and requires the [UiPath GitHub App](#) to be installed in your organization or account.

Authenticating to GIT over SSH

When cloning a repository or copying the current project to an existing GIT repository using SSH for the first time, you have the option of using a private key:



Ui Clone a remote GIT repository. X

Clone using SSH [Use HTTPS](#)

Repository URL:

Check out directory: ...

Use Key

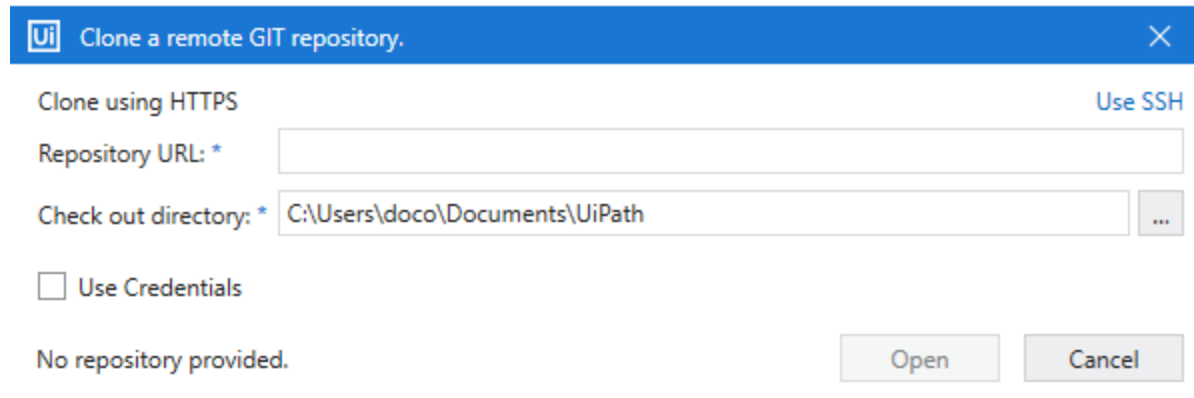
Private Key Path: ...

Password:

Add the Private Key Path and the Password, and then click Open to clone your remote GIT repository.

You will need to generate a SSH key for your GitHub repository.

Cloning a Remote GIT Repository



After cloning a GIT repository to a local working directory, the .git subdirectory is created containing the necessary GIT metadata.

The metadata includes subdirectories for objects, refs, and template files. In addition, a HEAD file is also created, which points to the currently checked out commit.

1. In the Team tab, select Clone Repository. The Clone a remote GIT repository. window is displayed.
2. Select either Use HTTPS or Use SSH
3. Type in the Repository URL, and choose an empty Check out directory
4. Select Use Credentials / Use Key and configure authentication (either sign in with GitHub, enter user and password, enter user and token for HTTPS, or enter private key path and password for SSH)
5. Click Open, Studio opens the project in the Designer panel
6. In the Open window, select a project.json file to open in Studio

Chapter 17:

Version Control Systems Integration in Studio

Using GIT in Studio



Resources

Topic	Link
Adding a Project to Git - UiPath Studio Guide Learn more about adding a Studio project to Git.	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Copying a Project to Git - UiPath Studio Guide Learn more about copying a project to Git.	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Disconnecting from Git - UiPath Studio Guide Learn more about disconnecting from Git.	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Committing and Pushing to Git - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Changing the Last Commit - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Undoing Pending Changes - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Authenticating to Git - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Solving Conflicts - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git#solving-conflicts

Creating and Managing Branches

Branching allows you to work on different versions of your project simultaneously, making it easier to collaborate with team members and experiment with new features without affecting the main codebase.

We will be covering the following topics

1. Create Branches
2. Switch between them
3. Merge changes back into the main branch

Creating and Managing Branches

To access the **add and manage branches** from the Manage Branches window, either

- right-click the project node or a file in the Project panel and select Manage Branches, or
- use the branch menu in the status bar

Creating a Branch

Step1 : Open the Manage Branches window

The screenshot displays the UiPath Studio interface for a project named 'Lists_and_Dictionaries_Demo'. The 'Project' panel on the left shows a context menu open over the project node, with the 'Manage Branches' option highlighted in red. The main workspace shows a workflow diagram with two activities: 'Create New Dictionary' and 'Build collection Cities US'. The 'Create New Dictionary' activity is configured with 'Save to' as 'Cities' and 'Value to save' as 'new Dictionary(of Strir'. The 'Build collection Cities US' activity is configured with 'First Item' as '"New York"' and 'Next Items' as '3 items in Collection'. The Properties panel on the right shows the 'Misc' section with 'DisplayName' set to 'Main'. The status bar at the bottom indicates the user is 'priyanka.sogani@uipath.co...' and the current branch is 'main'.

In the Project panel, right-click the project node or contained file or use the branch menu in the status bar and choose the option "Manage Branches"

Creating a Branch

Step 2 : Manage Branches Window

Ui Manage Branches

Current Branch: main

Branch Name	Author	Date	Commit Message
main	Costin David	5/10/2023 11:46:53 AM	testg
origin/main	Costin David	5/10/2023 11:46:53 AM	testg

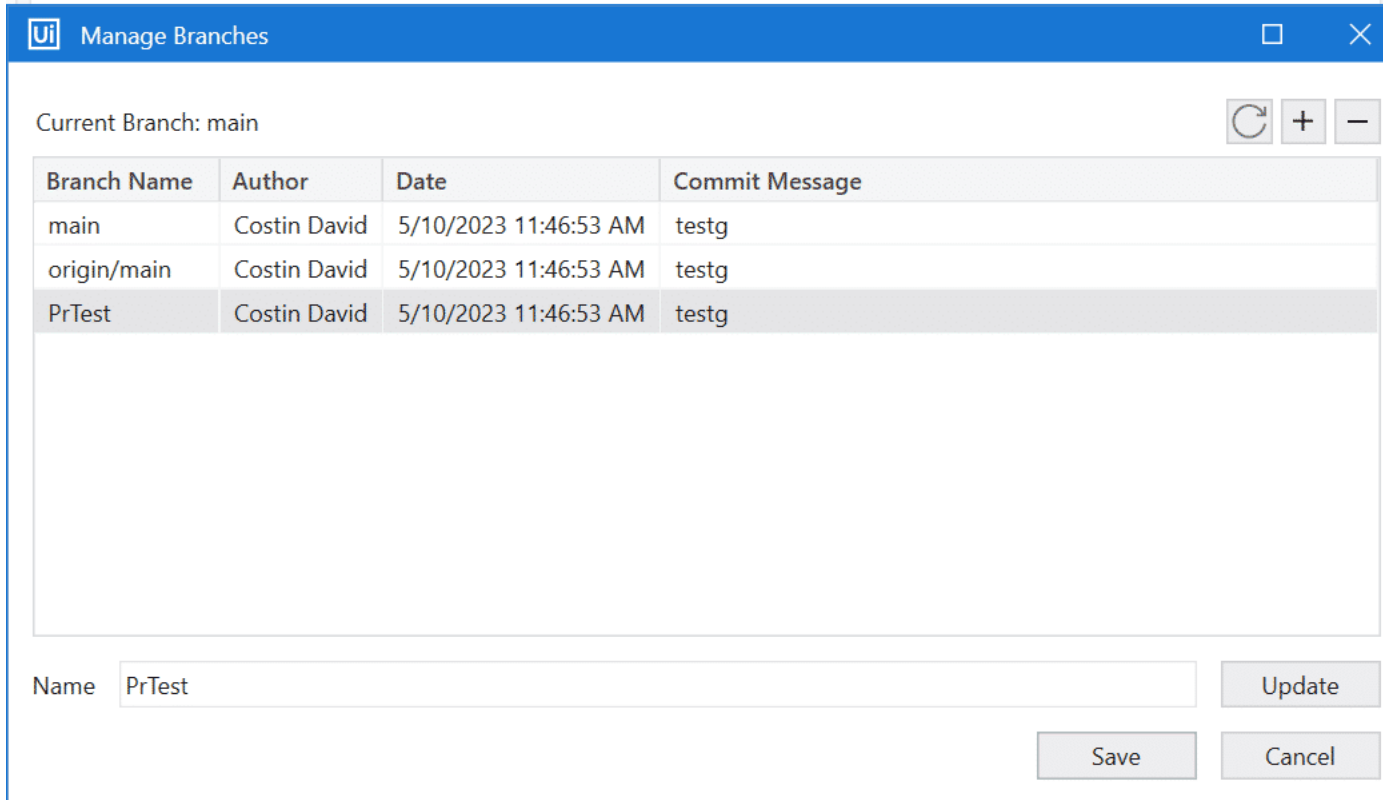
Name

Update Save Cancel

The Manage Branches window displays.

Creating a Branch

Step 3 : Add a New Branch



UiPath Manage Branches

Current Branch: main ↻ + -

Branch Name	Author	Date	Commit Message
main	Costin David	5/10/2023 11:46:53 AM	testg
origin/main	Costin David	5/10/2023 11:46:53 AM	testg
PrTest	Costin David	5/10/2023 11:46:53 AM	testg

Name Update

Save Cancel

Click on the plus button to add a branch.

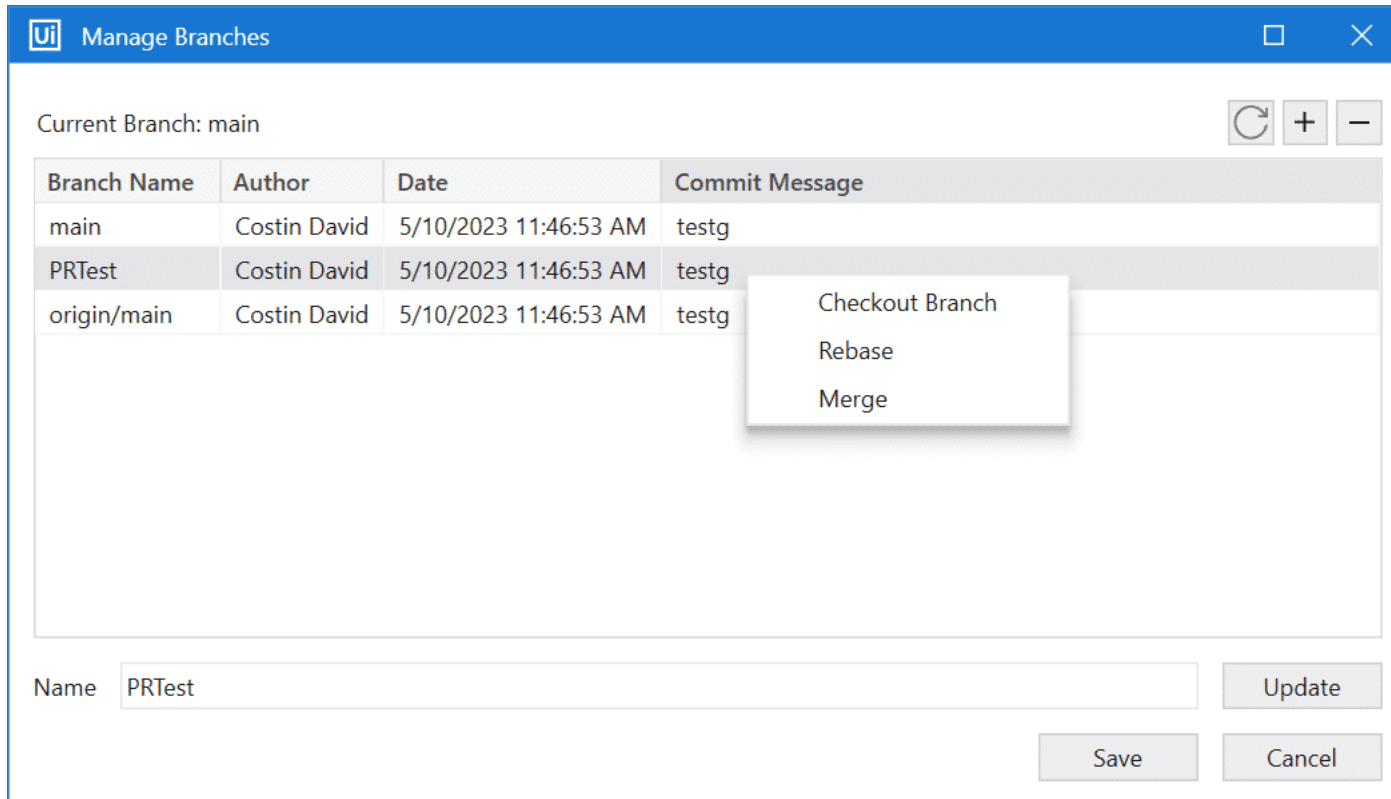
Provide a name for the branch in the Name section.

Select "Create branch for main" and then save.

The system will add the branch to the list.

Creating a Branch

Step 4 : Display Options



Current Branch: main

Branch Name	Author	Date	Commit Message
main	Costin David	5/10/2023 11:46:53 AM	testg
PRTTest	Costin David	5/10/2023 11:46:53 AM	testg
origin/main	Costin David	5/10/2023 11:46:53 AM	testg

Context Menu Options:

- Checkout Branch
- Rebase
- Merge

Name: PRTTest

Buttons: Update, Save, Cancel

When you right-click any branch, the options for Git branches are displayed:

- The "Checkout branch" option switches to the selected branch
- The "Rebase" option rebases the current branch onto the selected branch
- The "Merge" option merges the selected branch into the current branch

Classroom Exercise



Connect and manage automation projects using GIT with GitHub

Resources

About Version Control - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/about-version-control
Managing Projects with Git - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-git
Managing Projects with SVN - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-svn
Managing Projects with TFS - UiPath Studio Guide	https://docs.uipath.com/studio/standalone/2022.10/user-guide/managing-projects-tfs