PUBLIC
2021-07-15

Git Service for the Neo Environment

THE BEST RUN SAP

Content

1 SAP Git Service

Store and version source code in Git repositories.

The SAP Git service lets you store and version the application source code. It is based on Git, the widely used open-source system for revision management of source code that facilitates distributed and concurrent large-scale development workflows.

Git is a widely used open source system for revision management of source code that facilitates distributed and concurrent large-scale development workflows.

You can use any standard compliant Git client to connect to the SAP Git service. Many modern integrated development environments, including but not limited to SAP Web IDE, provide tools for working with Git. There are also native clients available for many operating systems and platforms.

Environment

This service runs in the Neo environment.

Features

Records differences between versions	Only the differences between versions are recorded allowing for a compact storage and efficient transport.
Cost-effective and simple	Create and merge branches supporting a multitude of development styles. Git is widely used and supported by many tools and is highly distributed. A clone of a repository contains the complete version history.
Operations on local repository clone	Perform almost all operations locally and thus very fast and without need to be permanently online. Only required when synchronizing with the Git service.

Restrictions

The SAP Git service is a dedicated service for source code versioning.

While Git can manage and compare text files very efficiently, it was not designed for processing large files or files with binary content, such as libraries, build artifacts, multimedia files (images or movies), or database backups. Consider using the document service or some other suitable storage service for storing such content.

To ensure best possible performance and health of the service, the following restrictions apply:

- The size of an individual file cannot exceed 20 MB. Pushes of changes that contain a file larger than 20 MB are rejected.
- The overall size of the bare repository stored in the SAP Git service cannot exceed 500 MB.
- The number of repositories per subaccount is not currently limited. However, SAP may take measures to protect the SAP Git service against misuse.

Third-Party Notice

The SAP Git service makes use of the Git-Icon-1788C image made available by Git (https://git-scm.com/downloads/logos) under the Creative Commons Attribution 3.0 Unported License (CC BY 3.0) http://creativecommons.org/licenses/by/3.0 .

Related Information

Managing Repositories [page 4]
Working with Repositories [page 9]
Security [page 13]
Best Practices [page 16]
Git
SAP Web IDE
What is Document Service

1.1 Managing Repositories

In the SAP BTP cockpit, you can create and delete Git repositories, as well as lock and unlock repositories for write operations. In addition, you can monitor the current disk consumption of your repositories and perform garbage collections to clean up and compact repository content.

Related Information

Create Repositories [page 5]
Change the State of Repositories [page 7]
Delete Repositories [page 7]
Clean Repositories [page 8]

1.1.1 Create Repositories

In the SAP BTP cockpit, you can create Git repositories for your subaccounts.

Prerequisites

You must be an administrator of the subaccount.

Context

> **i** Note
>
> To create a repository for the static content of an HTML5 application, see Create an HTML5 Application.

Procedure

1. Log on to the SAP BTP cockpit, and select the subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Choose *New Repository* and enter the following data.

   Data for Repository Creation

Field	Entry
Name	(Mandatory) A unique name starting with a lowercase letter, followed by digits and lowercase letters. The name is restricted to 30 characters.
Description	(Optional) A descriptive text for the repository. You can change this description later on.
Create empty commit	An initial empty commit in the history of the repository. This might be useful if you want to import the content of another repository.

4. Choose *OK*.
5. To navigate to the details page of the repository, click its name.

Results

The URL of the Git repository appears under *Source Location* on the detail page of the repository. You can use this URL to access the repository with a standard-compliant Git client. You cannot use this URL in a browser to access the Git repository.

Related Information

Navigate in the Cockpit
Create an HTML5 Application
Assign Developers to a Repository [page 6]

1.1.1.1 Assign Developers to a Repository

Permissions for Git repositories are granted based on the subaccount member roles that are assigned to users. To grant a subaccount member access to a Git repository, assign one of these roles: *Administrator*, *Developer*, or *Support User*.

Prerequisites

- You must be an administrator of the subaccount.
- New members to be added must already have SAP user IDs.

Context

For details about the permissions associated with the individual roles, see Security [page 13].

Procedure

1. Log on to the SAP BTP cockpit.
2. Assign members to subaccounts and define their roles.

   Make sure that you assign at least one of these roles: Administrator, Developer, or Support User.

Related Information

Managing Member Authorizations in the Neo Environment
User and Member Management

1.1.2 Change the State of Repositories

In the SAP BTP cockpit, you can change the state of a Git repository temporarily to READ ONLY to block all write operations.

Prerequisites

You must be an administrator of the subaccount.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. In the list of Git repositories, locate the repository you want to work with and follow the link on the repository's name.
3. On the details page of the repository, choose *Set Read Only*.

Results

The state flag of the repository changes from *ACTIVE* to *READ ONLY* and all further write operations on this repository are prohibited.

> **i** Note
>
> To unlock the repository again and allow write access, choose *Set Active* on the details page of the repository.

1.1.3 Delete Repositories

In the SAP BTP cockpit, you can delete a Git repository unless it is associated with an HTML5 application. In this case, delete the HTML5 application.

Prerequisites

You must be an administrator of the subaccount.

Context

> ⚠ Caution
>
> Be very careful when using this command. Deleting a Git repository also permanently deletes all data and the complete history. Clone the repository to some other storage before deleting it from the SAP Git service in case you need to restore its content later on.
>
> For more information, see Clone Repositories [page 10].

Procedure

1. Log on to the SAP BTP cockpit, and select the appropriate subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to delete.
4. From the *Actions* column, choose the delete icon ( ).
5. Confirm that you want to delete the repository.

1.1.4 Clean Repositories

In the SAP BTP cockpit, you can trigger a garbage collection for a repository to clean up unnecessary objects and compact the repository content aggressively.

Prerequisites

You are an administrator of the subaccount.

Context

Perform this operation from time to time to ensure the best possible performance for all Git operations. The SAP Git service also automatically runs normal garbage collections periodically.

> ℹ Note
>
> This operation might take a considerable amount of time and may also impact the performance of some Git operations while it is running.

Procedure

1. Log on to the SAP BTP cockpit, and select the subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and click the repository's name.
4. On the details page of the repository, choose *Garbage Collection*.

Results

The garbage collection runs in the background. You can use the Git repository without restrictions while the process is running.

1.2 Working with Repositories

Git supports many commands for working with repositories.

We assume that you are familiar with Git concepts, and that you have access to a suitable Git client, for example, SAP Web IDE for performing Git operations.

If you are new to Git, we strongly recommend that you read a text book about it, and consult the Best Practices guide before using the service.

Related Information

Using Source Control (Git) in SAP Web IDE
Best Practices [page 16]

1.2.1 Determine the Repository URL

The URL of the Git repository is shown under *Source Location* on the details page of the repository. Use this URL to access the repository using a Git client.

Prerequisites

In the subaccount where the repository resides, you must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and follow the link on the repository's name.
4. In the *Source Location* panel, copy the link labeled *Git Repository URL*.

1.2.2 Clone Repositories

You need to clone the Git repository of your application to your development environment.

Procedure

1. In the cockpit, copy the link to the Git repository of your application.
   a. Log on with a user who is a subaccount member to the SAP BTP cockpit.
   b. From the navigation area, choose *Applications* > *HTML5 Applications*.
   c. Click your newly created application.
   d. Switch to the *Versioning* tab.
   e. Under *Source Location*, copy the link that points to the Git repository of your application.
2. You can use any Git client to clone a repository from the SAP Git service. To use the Git command line client, execute these steps:
   a. Enter the following command:

      ```
      $ git clone <repository URL>.
      ```

   b. If prompted, enter your SCN user ID and password.

1.2.3 Fetch Changes from Repositories

The Git `fetch` operation transfers changes from the remote repository to your local repository.

Prerequisites

- You must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.
- You have cloned the repository to your workspace, see Clone Repositories [page 10].

Context

Refer to the SAP Web IDE documentation if you want to fetch changes to SAP Web IDE. Otherwise, see the documentation of your Git client to learn how to fetch changes from a remote Git repository.

Procedure

1. Execute a `fetch` or a `pull` command with your Git client.
2. Authenticate yourself with your SAP ID credentials.

Related Information

SAP Web IDE

1.2.4 Push Changes to Repositories

The Git `push` operation transfers changes from your local repository to a remote repository.

Prerequisites

- You must be a subaccount member who is assigned the role *Administrator* or *Developer*.
- You have already committed the changes you want to push in your local repository.
- You have ensured that the e-mail address in the push commit matches the e-mail address you registered with the SAP ID service.

Context

Refer to the SAP Web IDE documentation if you want to push changes from SAP Web IDE. Otherwise, see the documentation of your Git client to learn how to push changes to a remote Git repository.

Procedure

1. Execute a `push` command with your Git client.

2. Authenticate yourself with your SAP ID credentials.

Related Information

SAP Web IDE

1.2.5 Browse the Content of Repositories

The SAP Git service offers a web-based repository browser that allows you to inspect the content of a repository.

Prerequisites

In the subaccount where the repository resides, you must be a subaccount member who is assigned the role *Administrator*, *Developer*, or *Support User*.

Context

The repository browser gives read-only access to the full history of a Git repository, including its branches and tags as well as the content of the files. Moreover, it allows you to download specific versions as ZIP files.

The repository browser automatically renders `*.md` Markdown files into HTML to make it easier to create documentation.

Procedure

1. Log on to the SAP BTP cockpit, and select the required subaccount.
2. From the navigation area, choose *Repositories* > *Git Repositories*.
3. Select the repository you want to work with and follow the link on the repository's name.
4. In the *Source Location* panel, click the *Repository Browser* link.

1.2.6 Find Commits of a Given User

You can find the commits of a given user containing the user's name and e-mail address.

Procedure

1. Clone the Git repositories of the account to which the user had write access.

   For more information, see Determine the Repository URL [page 9] and Clone Repositories [page 10].
2. On each of the Git repositories, execute the following commands:

   ```
   $ git log --author <email address of the user>
   ```

   ```
   $ git log --committer <email address of the user>
   ```

1.3 Security

Access to the SAP Git service is protected by SAP BTP roles and granted only to subaccount members.

Restrictions

You can't host public repositories or repositories with anonymous access on the SAP Git service.

Authentication

Authentication for the SAP Git service is performed against the configured platform identity provider. The following providers are supported:

- SAP ID Service
  Users can use their SAP ID Service credentials to authenticate and access the SAP Git service.
- Custom Identity Authentication tenant
  The SAP Git service supports basic authentication against a custom Identity Authentication tenant that is configured as a platform identity provider. If a subaccount is configured to use a custom Identity Authentication tenant as the platform identity provider as described in Platform Identity Provider, then basic authentication is done against that custom Identity Authentication tenant.
  You can add members to a subaccount from the SAP ID Service as well. However, these users can't be used for basic authentication. This is because the security service doesn't support mixed use of custom platform Identity Authentication tenant users and SAP ID service users for basic authentication in the same subaccount. For more information, see the notes on basic authentication in Authentication.

If a custom Identity Authentication tenant is configured as the platform identity provider to grant Git permissions to users in this tenant, assign the respective roles to the respective user IDs (Pxxxxxx).

For more information on platform scopes, see Platform Scopes.

If you've configured your Identity Authentication tenant to act as a proxy to the corporate identity provider by following the steps described in Configure Trust with Corporate Identity Provider and Choose a Corporate Identity Provider as Default, make sure that SAP BTP receives the SAML 2.0 attributes with exactly the following names:

- `first_name`: First name of the user
- `last_name`: Last name of the user
- `mail`: E-mail of the user

To achieve this, configure your corporate identity provider to send SAML 2.0 attributes with exactly the same names as mentioned above in the SAML 2.0 assertion. For more information, see Configure the User Attributes Sent to the Application.

> **i Note**
>
> Before the SAP Git service supported custom Identity Authentication tenants (until October 25, 2018), only users of the SAP ID Service could perform basic authentication when executing Git commands.

Permissions

The permitted operations depend on the subaccount member role of the user.

Read access is granted to all users who are assigned the *Administrator*, *Developer*, or *Support User* role. These users are allowed to do the following:

- Clone a repository
- Fetch commits and tags

Write access is granted to all users who are assigned the *Administrator* or *Developer* role. These users are allowed to do the following:

- Create repositories.
- Push commits
- Push tags

  > **i Note**
  >
  > If the repository is associated with an HTML5 application, pushing a tag defines a new version for the HTML5 application. The version name is the same as the tag name.

- Create new remote branches.
- Push commits authored by other users (forge author identity).

Only users who are assigned the *Administrator* role are allowed to do the following:

- Delete repositories
- Run garbage collection on repositories

- Lock and unlock repositories
- Delete remote branches
- Delete tags
- Push commits committed by other users (forge committer identity)
- Forcefully push commits, for example to rewrite the history of a Git repository
- Forcefully push tags, for example to move the version of an HTML5 application to a different commit

For more information, see .

You can also use custom roles to grant permissions for using the SAP Git service. For more information on custom roles, see Manage Custom Platform Roles.

Related Information

Managing Member Authorizations in the Neo Environment
Create a Version

1.3.1 Data Protection and Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

> **i** Note
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source.

Handle personal data with care. You as the data controller are legally responsible when processing personal data.

If you need to know which repositories contain Git commits of a given user that contain the user's name and e-mail address, see Find Commits of a Given User [page 13].

If you need help with this, open a ticket on BC-NEO-GIT as described in 1888290. Please indicate the user's e-mail address and the account where Git repositories reside to which this user had write access.

If you need to anonymize a user's e-mail address and name from a given Git repository this requires rewriting the history of the Git repository. This will change the IDs of all affected commits and their successor commits. For more information,

If you intend to delete a subaccount or terminate your contract you can export the Git repositories by cloning them. For more information, see Determine the Repository URL [page 9] and Clone Repositories [page 10].

Related Information

Data Protection and Privacy

1.4 Best Practices

Following best practices can help you get started with Git and to avoid common pitfalls.

If you are new to Git, we strongly recommend that you read a text book about Git, search the Internet for documentation and guides, or get in touch with the large worldwide community of developers working with Git.

- Before creating a new branch, perform a fetch or pull command.
  In general, base your work on the most recent changes of your co-workers. Otherwise, you might have to resolve conflicts before you can push changes.
- Create a new local branch for every new feature or bug-fix you want to start.
- Derive new local branches from `origin/master`
- Write meaningful commit messages.
  A commit message should describe **why** you are doing a particular change rather than describing **what** you did. The **why** helps your co-workers understand the intention and consequences of your change. The **what** on the other side can easily be deduced from the source code, that is, from the difference between the old and new version.
- Commit early and often.
  While developing larger features, create periodic checkpoints in form of commits, for example, when you are experimenting with different possible solutions. If you do something wrong, you can easily go back to the last known good commit and start over. Before pushing the feature, squeeze the checkpoints into one final commit.
- Create small commits that are easy to digest.
  A commit that touches thousands of lines of code is likely to be much more difficult to understand and integrate than a commit that changes only a tiny piece of code. For example, it is usually considered bad practice to refactor your code and implement a new feature in one and the same commit.
- Avoid pushing patches that break your code.
  Git supports amendments to commits that have not yet been pushed to the SAP Git service. This means you can incrementally correct and improve your changes before you push them.
- Never rewrite a commit that has already been pushed to the SAP Git service.
  Your co-workers might already have fetched that commit to their local repositories and based their work on it. Instead, push a new commit.

  > **i** Note
  >
  > The only valid exception to this guideline is if you accidentally pushed a secret, for example, a password, to the SAP Git service.

- Don't create dependencies on changes that have not yet been pushed.
  While Git provides some powerful mechanisms for handling chains of commits, for example, interactive rebasing, these are usually considered to be for experienced users only.
- Do not push binary files.
  Git efficiently calculates differences in text files, but not in binary files. Pushing binary files bloats your repository size and affects performance, for example, in clone operations.

- Store source code, not generated files and build artifacts.
  Keep build artifacts in a separate artifact repository because they tend to change frequently and bloat your commit history. Furthermore, build artifacts are usually stored in some sort of binary or archive format that Git cannot handle efficiently.
- Periodically run garbage collection.
  Trigger a garbage collection in the SAP BTP cockpit from time to time to compact and clean up your repository. Also run garbage collection regularly for repositories cloned to your workplace. This will minimize the disk usage and improve the performance of common Git commands.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

THE BEST RUN SAP