Git Server Proxy for Repos (Github Enterprise, Bitbucket Server, Gitlab self-managed)

Private Preview

Document revision 2.0
Mar 27, 2023

Revision History

Date	Updates
4/13/2020	0.1 Initial documentation for git server proxy
5/13/2020	0.3 Incorporated review by documentation team
5/18/2020	0.4 Updated to include Bitbucket Server
6/19/2020	0.5 Gitlab support with updated init script
9/24/2020	0.6 Added Azure firewall to diagram and network requirements. Name cluster with do_not_use in name.
10/12/2020	Moved notebook into Gist to reduce cut & paste and spacing errors with Python.
11/19/2020	0.7 Updated instructions to reference self-serve git proxy notebook.
2/11/2021	0.8 Updated troubleshooting steps
3/15/2021	0.9 Insert additional troubleshooting error
6/8/2021	1.0 Updated with more concise instructions and updated enablement notebook.
3/27/2023	2.0 Updated with for updated notebook v0.0.19
5/08/2023	2.1 Updated with for updated notebook v0.0.20 to fix Nginx compatibility

Current Status

Git Proxy (Preview Closed - Exception based enrollment only)

ENABLEMENT NOTEBOOK NOT INCLUDED IN THIS DOCUMENTATION. IF YOU ARE EXISTING GIT PROXY CUSTOMER, CONTACT YOUR SOLUTION ARCHITECT FOR ENABLEMENT NOTEBOOK.

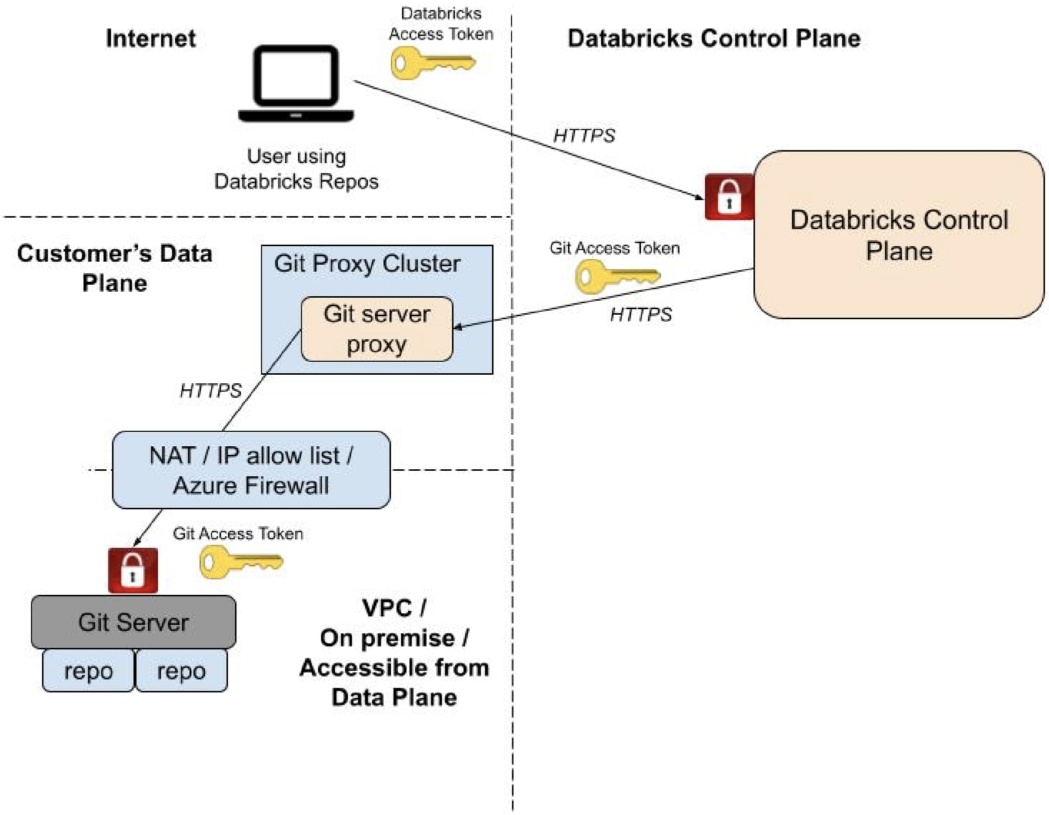
Introduction

Repos are folders whose contents are versioned together by syncing them to a Git repository. Repos supports collaboration through Git versioning, componentization of notebooks and their related code/data, and integration into execution workflows. See the Repos documentation for more information. Note that Repos currently can only contain Databricks notebooks and sub-folders, unless you have signed up for the "Files in Repos". The linked Git repository may contain other files, but they won't appear in the Databricks workspace.

By default, Repos can synchronize only with the public Github, Bitbucket, Gitlab and Azure Devops cloud services. If you host your own git server with GitHub Enterprise (GHE), Bitbucket Server(BBS), or Gitlab Self-managed you must use an additional Databricks feature called the Git server proxy to provide Databricks access to your Git server. Your Git server must be accessible from your workspace's virtual network.

IMPORTANT: The Git server proxy feature is currently available as *Private Preview*.

The Git server proxy works by proxying Git commands from the Databricks Control Plane to a cluster running in your workspace's Data Plane. In this document, the cluster is also referred to as your *proxy cluster*. A proxy server program runs on the proxy cluster to receive Git commands from the Databricks Control Plane and forward them to your git server instance. The diagram below illustrates the overall system architecture. For details, see the system architecture section.



This document includes which steps are required to enable the proxy feature for your Databricks workspace, configure your proxy cluster, and create repos from your privately-hosted repositories.

Prerequisites and planning

Before enabling the proxy, consider the following prerequisites and planning tasks:

- The Git server proxy requires that your workspace should have the Databricks Repos feature enabled
- You must make your Git server instance accessible from your Databricks workspace's Data Plane VPC, enable HTTPS, and enable personal access tokens on your Git server. See "Prepare your git server instance".
- You must create a proxy cluster and turn on a feature flag. See "Run the enablement notebook".

Step 1/ Prepare your Git server instance

Configure your Git server instance:

- Allow the proxy cluster's driver node to access your Git server. Your enterprise Git server likely has an allow list of IP addresses from which access is allowed.
    - First, associate a static outbound IP address for traffic that originates from your proxy cluster. You can do this by either of the following configurations:
        - AWS: Proxy traffic through a NAT gateway.
        - Azure: Use Azure Firewall or egress appliance.
        - GCP: Configure NAT gateway through subnetwork
    - Add the IP address from the previous step to your Git server's allow list.
- Configure your Git server instance to allow HTTPS transport.
    - If using Github Enterprise see Which remote url should I use.
    - If using Bitbucket then on the Bitbucket server administration page select the server settings. On this page in the HTTP(S) SCM hosting section enable the checkbox labeled HTTP(S) enabled.
- Configure your Git server instance to allow authentication by personal-access tokens
    - If using Github Enterprise then see creating a personal access token.

  - If using Bitbucket Server then go to Manage account > Account settings > Personal access tokens.
  - If using Gitlab self-managed then go Personal access tokens

Step 2/ Run the enablement notebook

Along with this document, you should have received an attached notebook to enable and set up the Git proxy cluster. If you don't have the notebook, please contact your Customer Success Engineer or Solutions Architect.

Please import the notebook in your Databricks workspace and follow the instructions on the notebook. You must be an admin on the workspace who has access rights to create a cluster.

A "Run All" of the enablement notebook will do the following things:

- Write a shell script to DBFS that is used as a cluster-scoped init script on the proxy cluster.
- Create a single node cluster named `dp_git_proxy`, which does not auto-terminate. You can customize the cluster's name in the enablement notebook.
- Enables a feature flag that controls whether Git requests in Repos are proxied via the cluster.

Note:

- Be aware of some security implications of the cluster configuration.
- By default, all users in the workspace are given "attach" permissions to the created cluster. You can control access to the cluster as you see fit. Users need "attach" permissions to use Repos in this setup.
- Running an additional long-running cluster to host the proxy software will incur extra DBUs. To minimize costs, we use a single node cluster with an inexpensive node type. However, feel free to customize the options to suit your needs.
- In typical configurations, most normal non-administrative users in your workspace can write to the DBFS location that is specified for the init script. Because other users can access this location, any user who can write to this DBFS location can replace the init script with their own version. Note for AWS customers: for better security, either lock down the init script using an S3 object lock or host the init script in an S3 bucket that has restricted role-based access control. If you have questions about either of these approaches, contact Databricks Support. Unfortunately this is not yet supported in Azure but it is currently on the roadmap.

Step 3/ Validate your Git server configuration

To validate your Git server configuration, you can try cloning a repo hosted on your private Git server on the proxy cluster. A successful clone means that you have successfully prepared your Git server instance.

There is a "Validate Git server setup" section in the enablement notebook that performs a simple clone, commit and push. You can attach the enablement notebook to the proxy cluster and run this cell, or perform Git operations after using SSH to log into the cluster.

Step 4/ Create proxy-enabled repos

After users configure their Git credentials, no further steps are required to create or synchronize repos. To configure credentials and create a repo in Repos, see the product documentation. The Git server proxy (Private Preview) features add support for GitHub Enterprise Server, Bitbucket Server, and Gitlab self-managed.

Troubleshooting

Checklist for common pitfalls

- Ensure that your proxy cluster is running.
- Ensure that your Repos users have "attach to" permissions on the proxy cluster.
- Ensure that your Repos users have set up their Git credentials.
- Validate your Git server configuration.
- Follow the instructions and run the debugging notebook. If you are unable to debug the issue, it can be useful for Databricks support to have the outputs of these cells. You can export and send the enablement notebook as a DBC archive.

Inspect logs on the proxy cluster

The file at `/databricks/db_repos_proxy/daemon.log`
 on the proxy cluster contains logs that are useful for debugging purposes.

The log file should start with the line "`Data-plane proxy server binding to ('', 8000)`…" If it does not, this means that the proxy server did not start properly. Try restarting the cluster, or delete the cluster you created and try running the enablement notebook again.

After this line, you should see log statements for each Git request initiated by a Git operation in Repos. For example:

```
do_GET: https://server-address/path/to/repo/info/refs?service=git-upload-pack
```

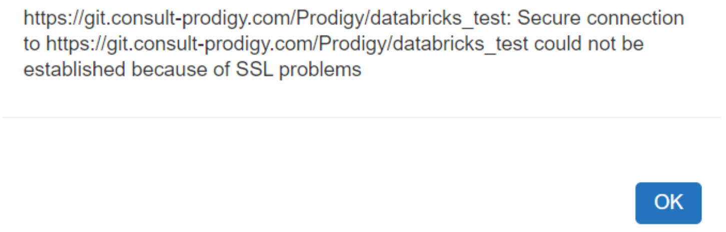
```
10.139.0.25 - - [09/Jun/2021 06:53:02] "GET
/server-address/path/to/repo/info/refs?service=git-upload-pack HTTP/1.1" 200
```

Any error logs written to this file can be useful to help you or Databricks support debug any issues.

Common error messages and their resolution:

1/ Secure connection could not be established because of SSL problems

You might see the following error:



https://git.consult-prodigy.com/Prodigy/databricks_test: Secure connection to https://git.consult-prodigy.com/Prodigy/databricks_test could not be established because of SSL problems

This likely means you are using a repository that needs special SSL certificates. Check the content of the /databricks/db_repos_proxy/daemon.log file on the proxy cluster. If it says that certificate validation failed, then you need to add the certificate of certificate authority into the system certificates chain. To do that you need to extract root certificate (via browser, or other way), upload it to DBFS, and update the enablement notebook like the highlighted instruction below.

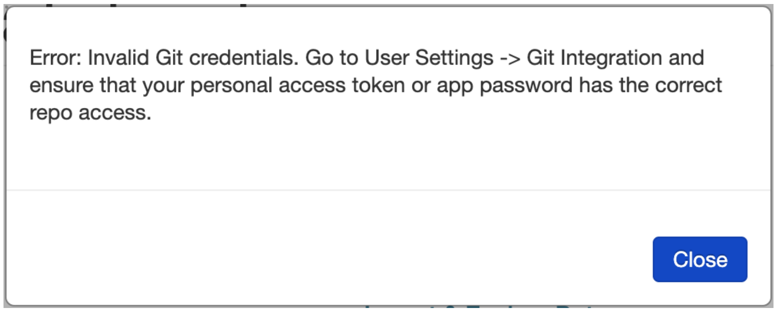
Unset

```
# Setup Systemd
cat > /etc/systemd/system/db_repos_proxy.service <<EOF
[Service]
Type=simple
# Either set ENABLE_SSL_VERIFICATION to false
# Or set CA_CERT_PATH to /FileStore/myCA.pem
Environment=ENABLE_SSL_VERIFICATION=true CA_CERT_PATH=''
ExecStart=/databricks/python3/bin/python3 -u
/databricks/db_repos_proxy/db_repos_proxy.py
StandardInput=null
StandardOutput=file:/databricks/db_repos_proxy/daemon.log
StandardError=file:/databricks/db_repos_proxy/daemon.log
Restart=always
RestartSec=1
```

2/ Failure to clone repository with error “Missing/Invalid Git credentials”

Ensure that you have configured your Git credentials in User Settings by following these instructions.

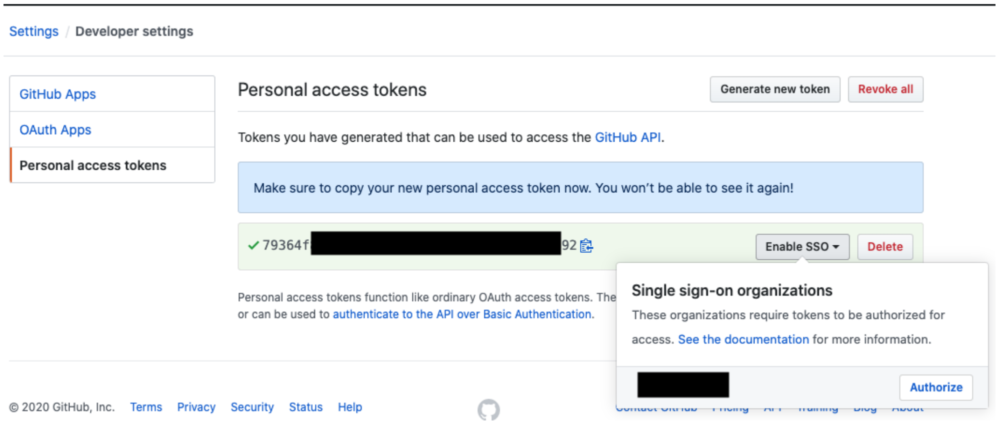
You might see the following error:



Error: Invalid Git credentials. Go to User Settings -> Git Integration and ensure that your personal access token or app password has the correct repo access.

If your organization is using SAML SSO, make sure the token has been authorized (this can be done from your git server personal access token management page)

a. Github Enterprise



Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your new personal access token now. You won’t be able to see it again!

✓ 79364f[redacted]92

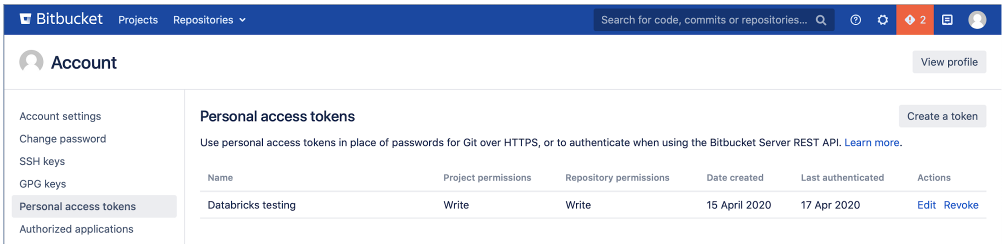
Personal access tokens function like ordinary OAuth access tokens. The
or can be used to authenticate to the API over Basic Authentication.

Single sign-on organizations

These organizations require tokens to be authorized for access. See the documentation for more information.

[redacted]

b. Bitbucket server je



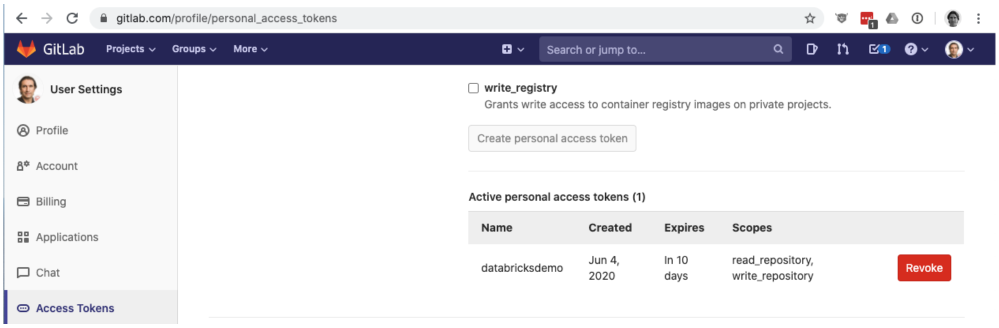
Account

Personal access tokens

Use personal access tokens in place of passwords for Git over HTTPS, or to authenticate when using the Bitbucket Server REST API. Learn more.

Name	Project permissions	Repository permissions	Date created	Last authenticated	Actions
Databricks testing	Write	Write	15 April 2020	17 Apr 2020	Edit Revoke

c. Gitlab self-managed



☐ write_registry
Grants write access to container registry images on private projects.

Create personal access token

Active personal access tokens (1)

Name	Created	Expires	Scopes	
databricksdemo	Jun 4, 2020	In 10 days	read_repository, write_repository	Revoke

Another thing to check is that all users have "can attach" permissions to the git proxy cluster.

Security Architecture

Our primary design principle is that only authenticated and authorized users with sufficient credentials can successfully send Git commands to your private Git server. The following access permissions and credentials are required for users to access repositories on your git server instance:

- **Network access to the Databricks control plane.** Network access to the Databricks control plane can be tightened by restricting access to your organization's VPN and/or an IP allowlist / access list.

- **Valid Databricks credentials.** The user can authenticate to the Databricks control plane. Your operations team can set ACLs on Databricks repos themselves to restrict access to repos to a subset of users.

- **Cluster "attach" permission to the proxy cluster.** You can restrict "attach" permissions on the proxy cluster to only those users or groups who require access to your git server instance. See security implications for more information.

- **Valid git server token.** Users who can connect to your git server instance still require valid git server credentials to access repositories.

Potential new vulnerabilities may be possible via the proxy cluster from users who are granted "attach" permissions to this cluster. This is an unmanaged service whose access control is managed by customers. Databricks recommends granting only trusted users attach permissions to this cluster.

The vulnerabilities include the ability to:

- **Access git server instance to try to exploit vulnerabilities.** Users could write code that uses network access to the IP address git server instance to try to exploit vulnerabilities. This potentially allows exploiting vulnerabilities in that server code. However, users can't access repositories without valid git server credentials, such as a valid personal access token.

- **Connect to other IP addresses accessible from your data plane.** The ability to connect to other IPs addresses in your data plane is unlikely to be a new attack vector because if the user already has attach permissions to at least one cluster in your data plane, they already can connect to arbitrary IPs addresses, including a location which the proxy cluster has egress access to in your data plane.

Databricks takes the security of your network, data, and credentials seriously. We are available to discuss security requirements or implications of the git server proxy with your operations team.

Security implications: Standard Clusters are recommended only for a single trusted user

The enablement notebook creates a Single Node cluster, which is a Standard Cluster with zero workers. The Git server proxy requires the cluster be a Standard Cluster and not a High-concurrency Cluster. Standard clusters do not isolate multiple users sharing the cluster. One user can access the files, data, and even Linux processes that were created by another user. (Refer to documentation for more details on Cluster configuration)

When the proxy cluster is configured as a Standard Cluster or otherwise without process isolation, be aware that this results in potential security risks from users with "attach" permissions to the cluster:

1. **Packet inspection.** Linux tools like `ptrace` could inspect the proxy server process on the proxy cluster. This could allow a sophisticated user/attacker to sniff tokens in the HTTP headers.

2. **Proxy server replacement.** A user/attacker could replace the proxy server code with a malicious version that listens on the proxy port. Like the Linux API vectors, this exposes tokens sent in HTTP headers.

As a consequence, the proxy and Standard Clusters are recommended only for a single trusted user. All users in a single workspace will share a single proxy cluster. We recommend only giving a single administrator the "can manage" permission. Users can have the "can attach" permission.

Frequently Asked Questions

What are the security implications of the git server proxy?

The most important things to know:

- Proxying does not affect the security architecture of your Databricks control plane.
- In the Private Preview release, the proxy could open new attack vectors for privileged users. See “Architecture” and “Security implications: Standard Clusters are recommended only for single users”.

Is all repos-related Git traffic routed through the proxy cluster, even for public Git repos?

Yes. In the Private Preview release, the workspace does not differentiate between proxied and non-proxied repos. This may change in a future release.

Does the Git proxy feature work with other git server providers?

Yes, Databricks supports Github Enterprise, Bitbucket Server, and Gitlab self-managed.

Do Repos support GPG signing of commits?

No.

Do Repos support SSH transport for Git operations?

No, only HTTPS is supported.

Is the use of a non-default HTTPS port on the Git server supported?

Currently, the enablement notebook assumes that your Git server uses the default HTTPS port 443. Please contact Databricks support if your Git server uses a non-default port number.

Can you share one proxy for multiple workspaces or do you need one proxy cluster per workspace?

You need one proxy cluster per workspace.

Does the proxy work with legacy single-notebook versioning?

No, the proxy does not work with legacy single-notebook versioning. Users must migrate to Repos versioning.

Can Databricks hide git server URLs that are proxied? Could users enter the original git server URLs rather than proxied URLs?

Yes to both questions. Users do not need to adjust their behavior for the proxy. With the current proxy implementation, all Git traffic for Repos is routed through the proxy. Users enter the normal Git repo URL such as `https://git.company.com/org/repo.git`.

How often will users work with the Git URLs?

Typically a user would just add the Git URL when creating a new repo or checking out an existing repo that they have not already checked out.

Does the feature transparently proxy authentication data to the git server?

Yes, the proxy uses the user account's git server token to authenticate to the git server.

Is there Databricks access to git server code?

The Databricks service accesses the Git repository in the git server service using the user-provided credential and synchronizes any code files in the repository with the repo. Access is restricted by the permissions attributed to the user-provided personal access token.