BridgeLink Version History: User Guide

For BridgeLink WebAdmin

Table of Contents

About This Guide

Version History brings Git-based version control into BridgeLink. It tracks, commits, and pushes changes to your channels, code templates, libraries, and global scripts to a remote Git repository, so you get a full audit trail of every change and can restore a previous version without leaving WebAdmin. This guide is for BridgeLink administrators who configure Version History and for anyone who reviews history, compares revisions, or restores a channel or code template. No Git experience is required; this guide walks through each screen and task.

Version History works by tracking a folder in a remote Git repository alongside your BridgeLink configuration. Every channel, code template, library, and set of global scripts you save can be written to the repository, automatically or on demand, and every write becomes a Git commit. You can then view a line-by-line diff between any two revisions and restore an entity to a prior revision from inside WebAdmin.

Requirements

- BridgeLink 26.6.0 or later.
- See **Availability**, below, for licensing details.
- A remote Git repository that your BridgeLink server can reach (for example on GitHub, GitLab, or a self-hosted Git server), and credentials for it: an SSH private key or an HTTPS username and personal access token.

Availability

Version History is open source and part of the core BridgeLink product. It is included in all editions at no additional cost, and no license entry is required. It appears automatically in WebAdmin; there is no separate package to find or install through a marketplace.

Whether the **Version History** navigation item, its **Settings** tab, and its buttons throughout WebAdmin appear for a given user depends on the permissions your administrator has assigned to that user's role.

Configuration

Configure Version History from **Settings**.

1. Sign in to **WebAdmin** as an administrator.
2. Click **Settings** in the left navigation.
3. Click the **Version History** tab.

The **Version History** settings tab has three sections: **Enable**, **Git**, and **Auto Commit**. The **Git** and **Auto Commit** sections only appear once **Enable** is set to **Yes**.

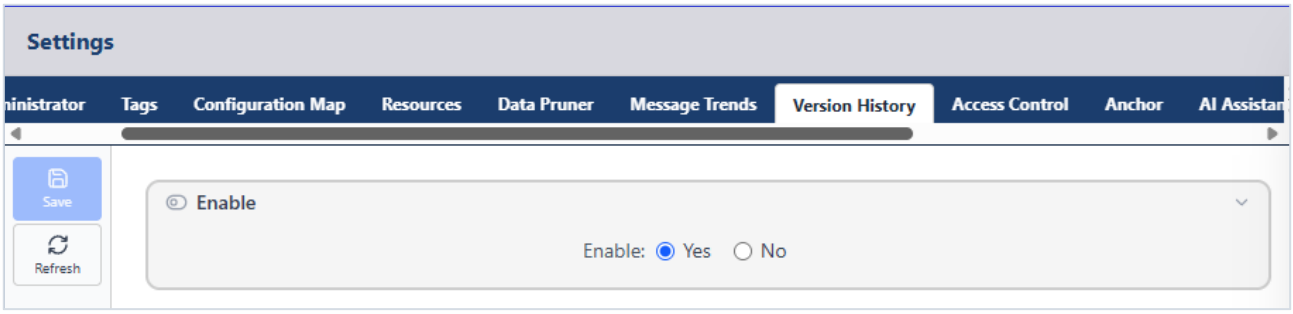

Figure 1: the Version History settings tab's Enable section.

Enable

Setting	What it does	Default
Enable	Master switch for the entire Version History feature. Select **Yes** to turn tracking on, or **No** to turn it off.	No

> **Note:** Set **Enable** to **Yes** and click **Save** before configuring **Git** or **Auto Commit**. Both sections stay hidden until the feature is enabled.

Git

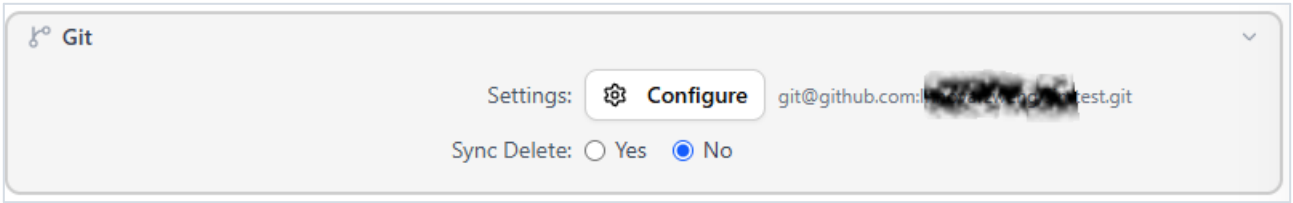

Figure 2: the Version History settings tab's Git section.

1. In the **Git** section, click **Configure**. The **Git Settings** dialog opens.
2. In **Remote Repository URL**, type the URL of your Git repository, for example `git@github.com:user/repo.git` for SSH or `https://github.com/user/repo.git` for HTTPS.
3. In **Branch Name**, type the branch Version History commits to, for example `main`.

4. Choose an **Authentication Type**: **SSH** or **HTTPS (Personal Access Token)**.
5. Fill in the credentials for the authentication type you chose, described in the tables below.
6. Click **Validate** to confirm BridgeLink can reach the repository with the credentials you entered.
7. Click **Save**.

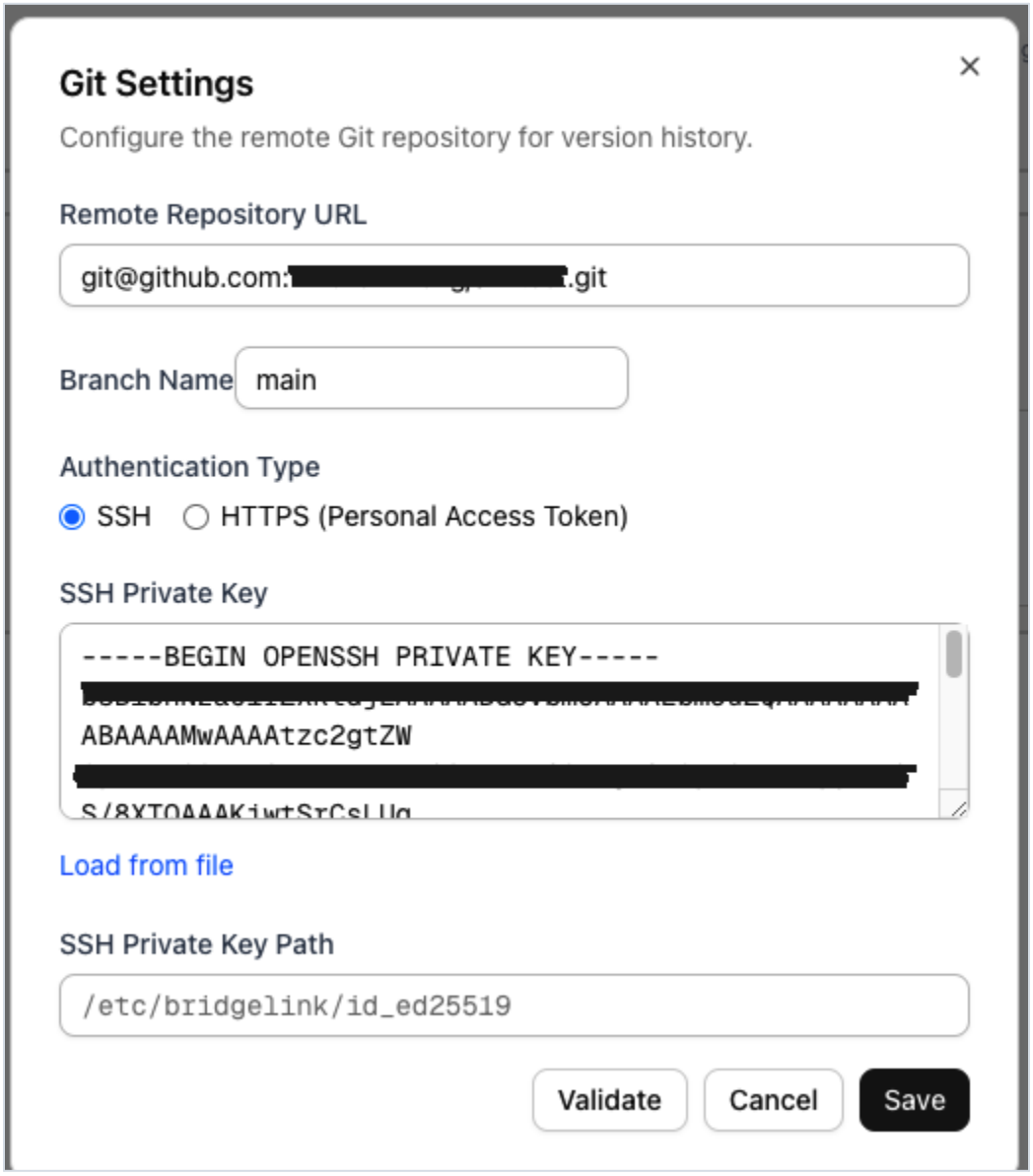

Figure 8: the Git Settings dialog, opened from the Git section's Configure button.

SSH authentication fields:

Field	What it does
SSH Private Key	Paste the private key directly. It must start with `-----BEGIN`.

Field	What it does
Load from file	Link that loads a key file from your computer into the **SSH Private Key** field.
SSH Private Key Path	A server-local file path to the private key, used instead of pasting the key. If both this and **SSH Private Key** are set, the pasted key wins.

HTTPS authentication fields:

Field	What it does
Username	The Git username for HTTPS authentication.
Personal Access Token	The token used as the HTTPS password. Generate it in your Git provider's developer settings, with the minimum scopes needed to push to the repository.
HTTPS Credentials Path	A server-local file path to a credentials file containing the username and token, used instead of typing them inline. If both this and the inline fields are set, the inline values win.

Back on the **Version History** settings tab, the **Git** section also has:

Setting	What it does	Default
Sync Delete	When set to **Yes**, deleting a channel or code template also removes its file from the Git repository on the next commit.	No

Auto Commit

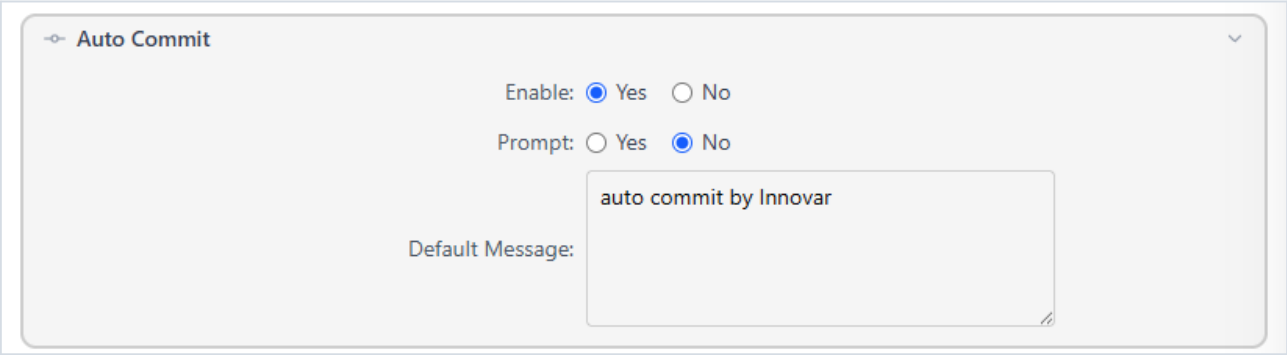

Figure 3: the Version History settings tab's Auto Commit section.

Setting	What it does	Default
Enable	Automatically writes and commits configuration changes to Git as soon as they are saved in BridgeLink.	No

Setting	What it does	Default
Prompt	When set to **Yes**, WebAdmin asks for a commit message every time it commits. When set to **No**, the message in **Default Message** is used automatically.	`No`
Default Message	The commit message used when **Auto Commit** is on and **Prompt** is off.	`(empty)`

When **Auto Commit** and **Prompt** are both on, saving a channel opens a **Commit Channel** dialog with a **Commit message** field, pre-filled with your default message as a placeholder. Leave it blank to use the default message, or type a different one, then click **Commit & Push**.

Even with **Auto Commit** off, you can always commit and push manually: from the **Local Changes** tab described below, or from the **Commit & Push** and **Save Libs** buttons covered later in this guide.

Click **Save** on the **Version History** settings tab to apply any changes on this page.

Using Version History

Once Version History is enabled, click **Version History** under **Build** in the left navigation, alongside **Channels**, **Code Templates**, and **Global Scripts**. The **Version History** page has four tabs: **Commits**, **Local Changes**, **Files**, and **Status**.

Commits Tab

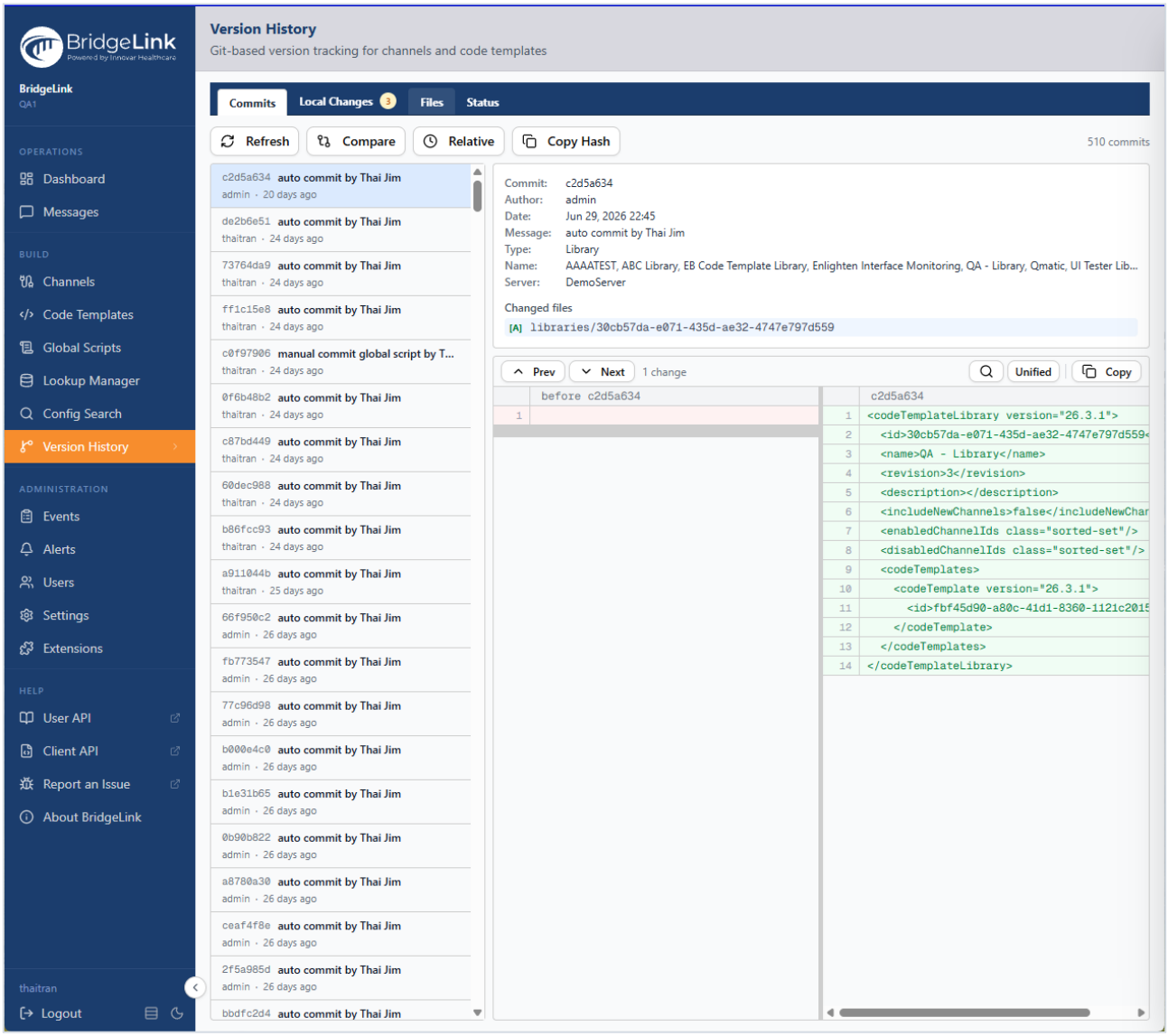

Figure 6: the Version History page's Commits tab, showing the commit list and diff viewer.

The **Commits** tab lists every commit in the repository, newest first, up to 1,000.

1. Click a commit in the list to view its details and changed files on the right.
2. Click a file under **Changed files** to load its diff.

3. Use **Prev** and **Next** in the diff toolbar to jump between changed sections, or click **Unified** to switch between the side-by-side and unified diff views.

Toolbar controls:

Control	What it does
Refresh	Reloads the commit list.
Compare	Switches to compare mode, where you pick any two commits and view the differences between them. You can also Ctrl-click (Cmd-click on macOS) two commits in the list to jump straight into compare mode.
Relative / **Exact**	Toggles commit timestamps between relative text (for example `18 hours ago`) and exact dates.
Copy Hash	Copies the selected commit's hash to the clipboard.

If a changed file belongs to a channel or code template that still exists in BridgeLink, an **Open in Editor** button opens it directly.

Local Changes Tab

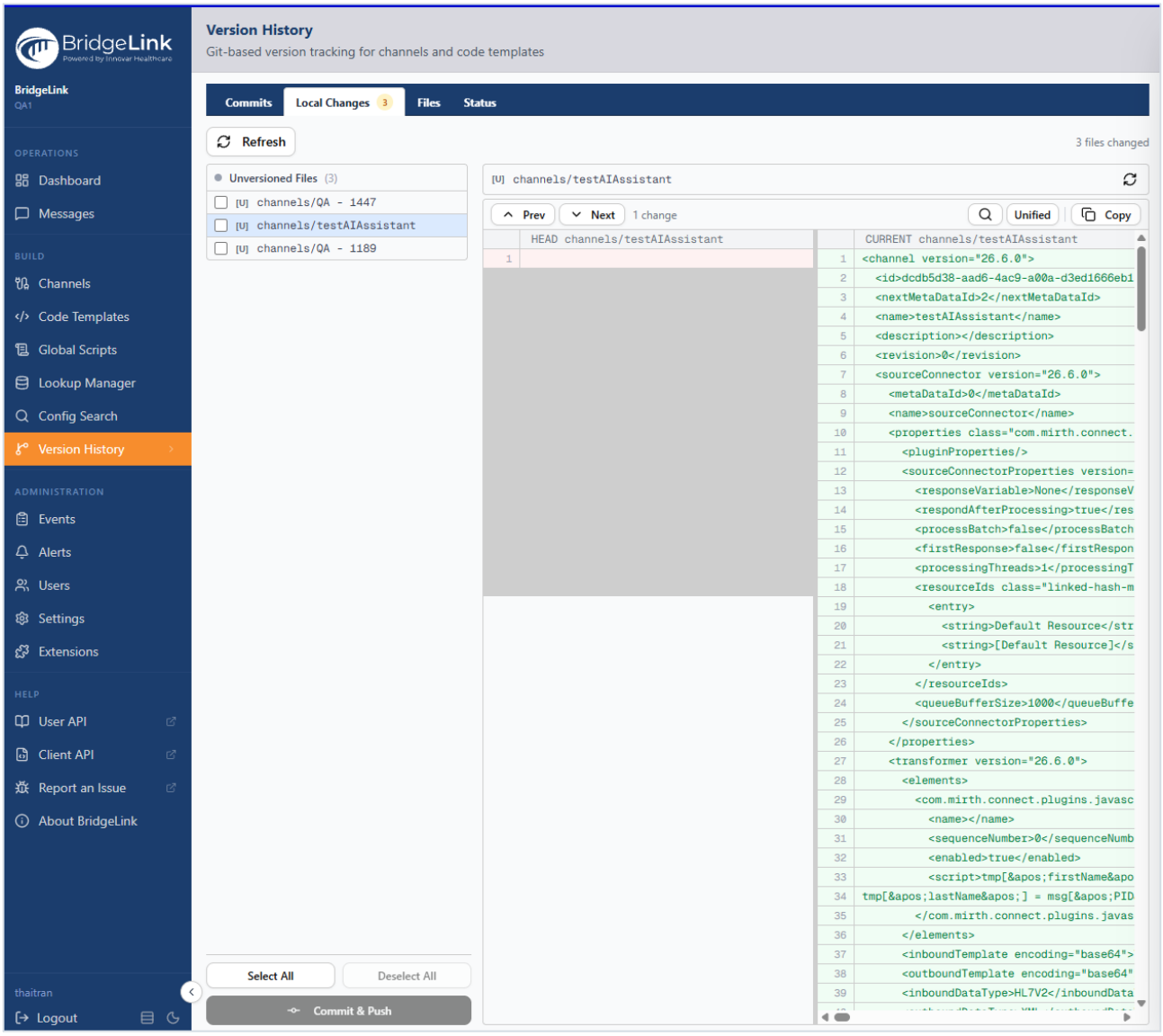

Figure 5: the Version History page's Local Changes tab, showing unversioned files and a diff.

The **Local Changes** tab lists files that have changed on disk but have not yet been committed, grouped under headings such as **Changed** (modified or deleted files) and **Unversioned Files** (new files not yet tracked). Each file carries a status tag: `[M]` modified, `[D]` deleted, or `[U]` untracked. A badge on the tab shows the pending count.

1. Select the checkbox next to each file you want to commit, or click **Select All** (**Deselect All** clears the selection).
2. Click a file to preview its diff on the right.
3. Click **Commit & Push**.
4. In the **Commit message** field, type a message describing the change.
5. Click **Commit & Push** to commit and push the selected files, or **Cancel** to close the dialog without committing.

Right-click a file for more options, including **Discard Changes** (restores the file to its last committed state) and **Open in Editor**.

Files Tab

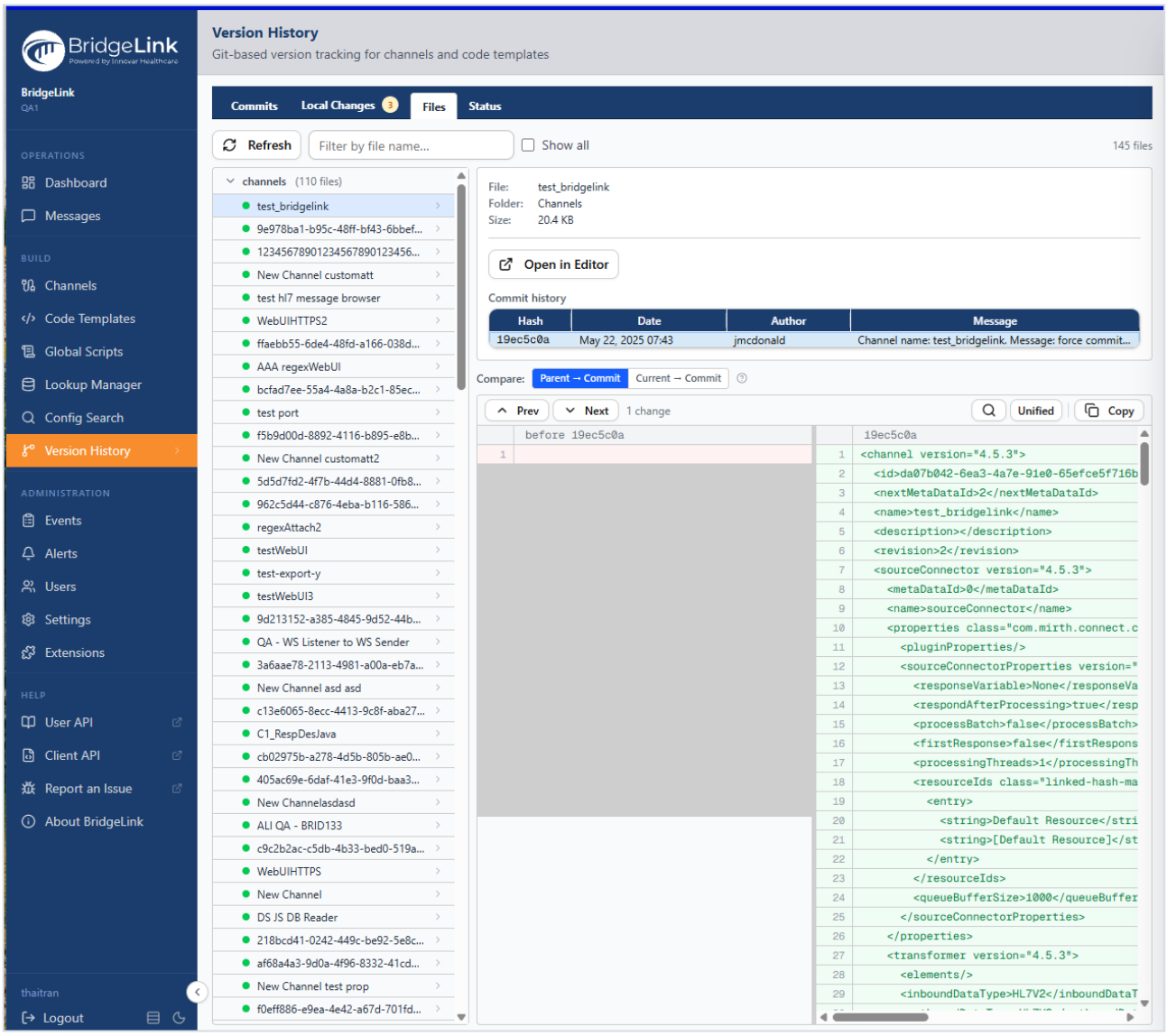

Figure 4: the Version History page's Files tab, showing a file's commit history and diff.

The **Files** tab lists every file tracked in the repository, grouped by folder: `channels`, `codetemplates`, `libraries`, and `globalscripts`.

1. Type in the **Filter by file name** field to search for a file, or select **Show all** to include folders BridgeLink does not manage.
2. Click a file to load its **Commit history** and the diff for its most recent commit on the right.
3. Click a different commit in that file's history to view that revision's diff. Comparison buttons above the diff let you choose what it is measured against: **Parent → Commit**

(the previous commit), **Current → Commit** (the version currently on the server), or **Last Saved → Commit** (the most recently committed version), the last of which only appears when the selected commit is not the most recent one.

4. If the file has two or more commits, click **Compare** next to **Commit history** to switch into compare mode and pick any two of its revisions to diff against each other.
5. Right-click a commit and select **Restore to this revision** to roll the file back to that revision. The option reads **Already at this revision** and is disabled once the working copy already matches.

Restoring a channel, code template, or library updates it immediately on the BridgeLink server, and the change then appears on the **Local Changes** tab so you can commit it. If you restore a code template that no longer exists in BridgeLink, WebAdmin asks you to choose which library to restore it into.

Status Tab

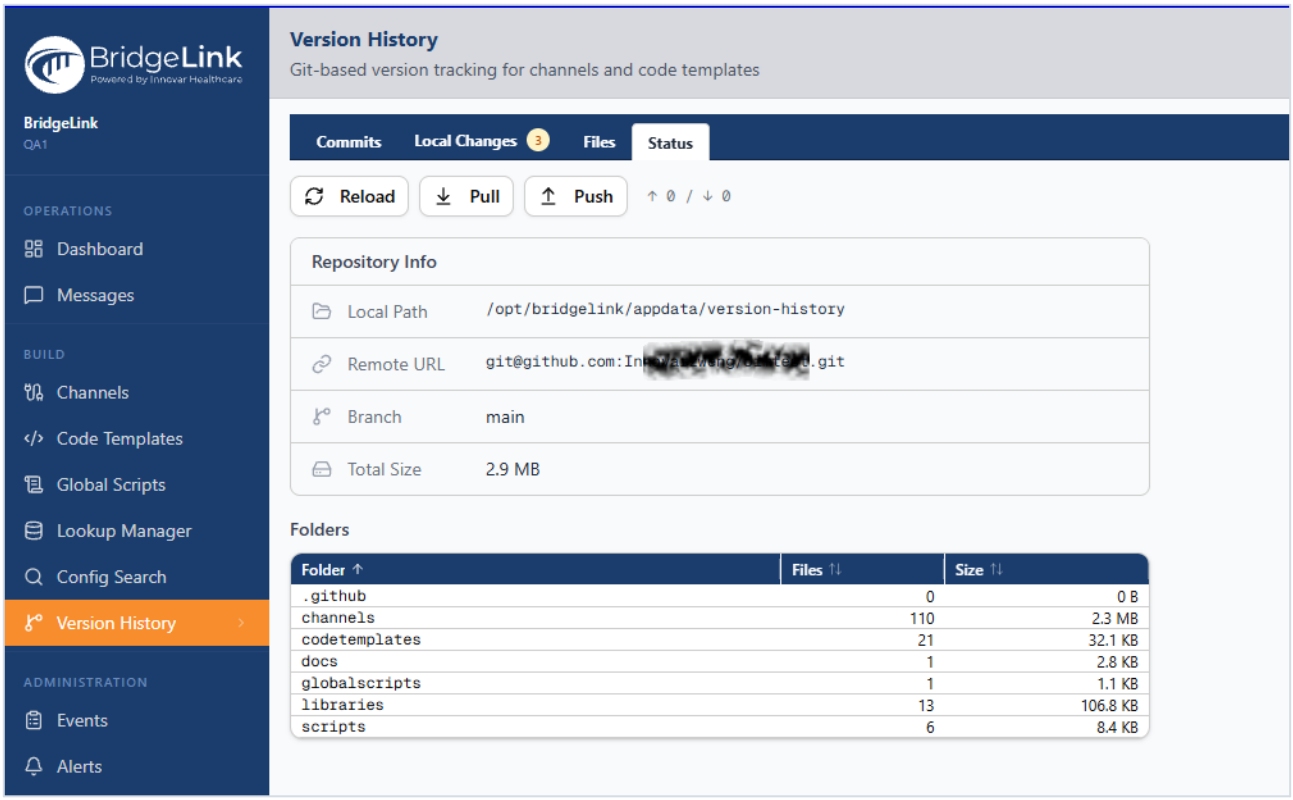

Figure 7: the Version History page's Status tab, showing repository info, folder breakdown, and remote sync controls.

The **Status** tab summarizes the state of the Git repository in a **Repository Info** card:

Field	What it does
Local Path	The directory on the server where the repository is stored.
Remote URL	The remote repository URL.

Field	What it does
Branch	The branch Version History is tracking.
Total Size	The size of the local repository on disk.

A **Folders** table below breaks the repository down by folder, showing a file count and size for each.

The tab also offers:

Control	What it does
Reload	Fetches from the remote repository and refreshes the ahead and behind commit counts, shown next to the buttons as a counter in the form `↑ 0 / ↓ 0`.
Pull	Pulls the latest changes from the remote. Merge conflicts are resolved automatically in favor of the remote version; your own unpushed commits are kept.
Push	Pushes your local commits to the remote.

Channel History

Every channel has its own **Version History** tab in the channel editor, alongside **Summary**, **Source**, **Destinations**, and **Scripts**.

1. Open the channel for editing.
2. Click the **Version History** tab.

This tab lists every commit for that channel and supports the same diff, restore, and compare actions as the **Commits** tab, plus:

Control	What it does
Show	Limits the number of commits loaded: `5`, `10`, `20`, `50`, `100`, `200`, or `500`. Defaults to `10`.
Restore	Restores the channel to the selected revision. Disabled once the working copy already matches that revision.

When you select a commit, comparison buttons above the diff let you choose what it is measured against: **Parent → Commit** (the previous commit), **Current → Commit** (the version currently on the server), or **Last Saved → Commit** (the most recently committed version), the last of which only appears when the selected commit is not the most recent one.

1. To restore a channel, select a commit and click **Restore**.
2. Confirm in the dialog that appears. The selected revision loads into the channel editor and is saved to the server immediately.
3. WebAdmin then offers to commit and push the restore to Git. Click **Commit & Push** to do so, or **Cancel** to leave it as a pending change on the **Local Changes** tab.

> **Note:** A restore is not complete in Git until you either accept that final **Commit & Push** offer or commit it yourself later from the **Local Changes** tab.

Importing a Channel from the Repository

1. On the **Channels** page, click the **Import** dropdown.
2. Select **Import from Repo**. The **Import Channel From Repo** dialog lists every channel tracked in the repository.
3. Type in the filter field to narrow the list by name or ID, if needed.
4. Select the checkbox next to each channel to import.
5. Click **Import**, or **Cancel** to close without importing.

Importing a single channel opens it in the **New Channel** editor for review; nothing is saved until you click **Save** there. Importing more than one channel at a time creates them immediately. Channels that already exist in BridgeLink, matched by channel ID, are listed but cannot be selected.

Code Templates

Code template libraries and templates each have their own history in the repository.

1. On the **Code Templates** page, select a library or template.
2. Click **History**. The history dialog for that item opens, named after it.

This dialog supports the same diff, compare, and restore controls as channel history, plus a **Commit & Push** button that commits the current state of that single template. Restoring a template updates it on the server and in the repository's working tree; restoring a template that no longer exists in BridgeLink prompts you to pick a target library.

To commit and push the entire library structure, meaning all libraries and templates, at once:

1. On the **Code Templates** page, click **Save Libs**.
2. If prompted, type a message in the **Commit message** field. (With **Auto Commit** on and **Prompt** off, this happens silently with the default message instead.)
3. Click **Commit & Push**, or **Cancel**.

To import code templates from the repository:

1. On the **Code Templates** page, click the **Import** dropdown, then **Import from Repo**.
2. Select the templates to import. Expand a library to see its templates, or use **Select All**.
3. Click **Next: Select Library**.
4. Choose the BridgeLink library to import the templates into.
5. Click **Import**.

Templates that already exist in BridgeLink by ID are shown but cannot be selected.

Global Scripts

The four global scripts, **Deploy**, **Undeploy**, **Preprocessor**, and **Postprocessor**, are tracked together as a single history.

1. On the **Global Scripts** page, click **History**. The **Global Scripts History** dialog opens.
2. Select a commit, then use the **Script** selector to switch between the four scripts.
3. Use the **Compare** selector to choose what the diff is measured against: **Current → Commit** or **Parent → Commit**.
4. Click **Restore** to load a commit's scripts into the editor.

> **Note:** Restoring here only loads the scripts into the editor. Click **Save** on the **Global Scripts** page afterward to persist them to the server.

To commit and push the current global scripts:

1. On the **Global Scripts** page, click **Commit & Push**.
2. If prompted, type a message in the **Commit message** field, then click **Commit & Push**, or **Cancel**.

Troubleshooting

Symptom	Likely cause	What to do
Validate fails in the **Git Settings** dialog	Incorrect repository URL, branch name, or credentials	Confirm the URL and branch name, and re-enter the SSH key or HTTPS token.
Changes are not appearing in the repository	**Enable** is set to **No**, or **Auto Commit** is off and nobody has committed manually	Set **Enable** to **Yes** on the **Version History** settings tab, and either turn on **Auto Commit** or commit manually from the **Local Changes** tab.
A restored code template cannot be matched to its library automatically	The template's original library no longer exists in BridgeLink	Choose a different existing library in the prompt that appears, so the template can be restored into it.

Support

Contact Innovar Healthcare support at `support@innovarhealthcare.com` for help with Version History. Include:

- Your BridgeLink version, and the Version History settings in use (**Enable**, **Auto Commit**, and whether SSH or HTTPS authentication is configured).
- The **Remote URL** and **Branch** shown on the **Status** tab. Do not include credentials.
- The exact error message from **Validate**, **Commit & Push**, **Pull**, or **Push**, if one appeared.
- The relevant Server Log entries, if available.