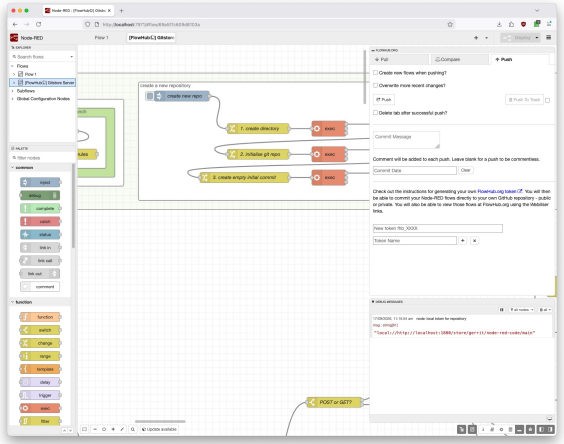
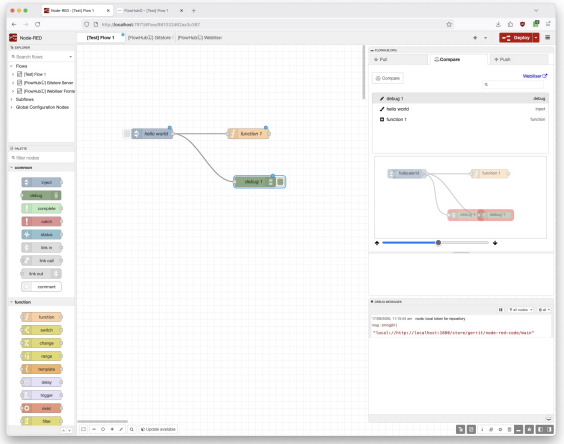
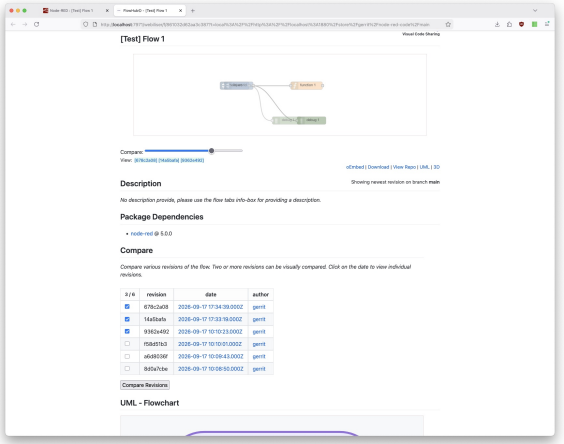
FLOWHUB







Installation Guide

Introduction

FlowHub.org provides *visual* change management (aka version control) on top of GitHub. FlowHub Local (aka FlowHubⓁ) is a local-first server for the FlowHub.org nodes, replacing the GitHub requirement.

FlowHubⓁ provides a local gitstore for managing change made to flows, locally, no internet connection is required. FlowHubⓁ is used in conjunction with the FlowHub.org nodes but provides a local gitstore, replacing GitHub.

FlowHubⓁ has a client-server architecture with both client and server being installed in Node-RED instances. The client and server are both Node-RED node packages that can be installed into any Node-RED instance.

FlowHubⓁ is designed to be installed on a central instance of Node-RED and multiple clients connect and communicate with that installation of FlowHubⓁ.

This solution was inspired by the Local First Conf initiative.

Purpose

Integrated Visual Change Management in Node-RED.

Providing a local git store for versioning flow tabs independently of the Node-RED instance or flows.json file. It is a local-first setup, complete with wheels for storing versioned flow code locally. No internet connection required.

The FlowHubⓁ "gitstore" summed up:

- Is based on git and follows gits' original design philosophy of being an offline version control system that eventually synchronises with an external git server.
- Can be used in the same way as git, since FlowHubⓁ utilises conventional git commands, no modifications or extensions are made nor installed to the git protocol or storage mechanism.
- Interacts with the Git database directly without cloning or checkout out the repository. All repository directories managed by FlowHubⓁ contain a single `.git` directory.
- Multiple git repositories can be created and independently maintained via tokens.
- Seamless and integrated exchanging of flows between instances of Node-RED.

The basic workflow supported by FlowHubⓁ is pulling flows from a repository, comparing local changes to the those in a git repository and pushing changes to git repositories.

Dependencies

The guide uses a docker container to demonstrate the installation of the server flow. The complete docker command is:

```
docker run -it --rm -p 0.0.0.0:7971:1880 -v $(pwd)/data:/data -v $(pwd)/repos:/repos --network fubar --r
```

Assumed is:

- the `git` command is installed, that is the case for the docker container,
- the existence of a 'repos' directory which will contain multiple git repositories, in individual directories.

An extra directory is required for storing git repositories and their contents. In this example, this will be the `/repos` directory *inside the docker container* but locally, outside of the container, this will the 'repos' directory in the current directory. This directory is configurable.

The FlowHubⓁ gitstore inside a docker container solution is recommended as it provides isolation, separation of concerns and nothing else needs installing.

For simplicity, this guide will install the client and the server in the same Node-RED instance. Normally FlowHubⓁ gitstore should be run as a separate Node-RED instance within the local network allowing Node-RED installations access.

About this Installation Guide

This guide provides screenshots and step descriptions for installing the FlowHubⓁ storage (gitstore) flows. It also demonstrates using the FlowHub.org nodes, in conjunction with FlowHubⓁ server, to initiate the versioning of a flow tab.

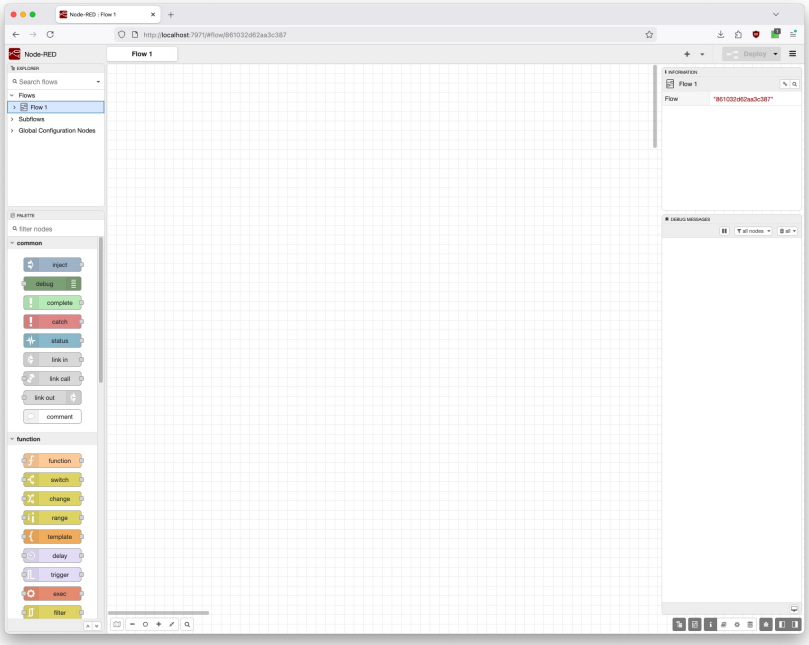
To begin with, the `flowhub-local-server` flow that manages the `gitstore` will be installed. It provides the APIs endpoints for the integrated Node-RED FlowHub nodes and also the Webiliser frontend endpoints.

For the Webiliser a second flow is required to provide all the Javascript frontend libraries. This flow is called `flowhub-local-cdn-assets`.

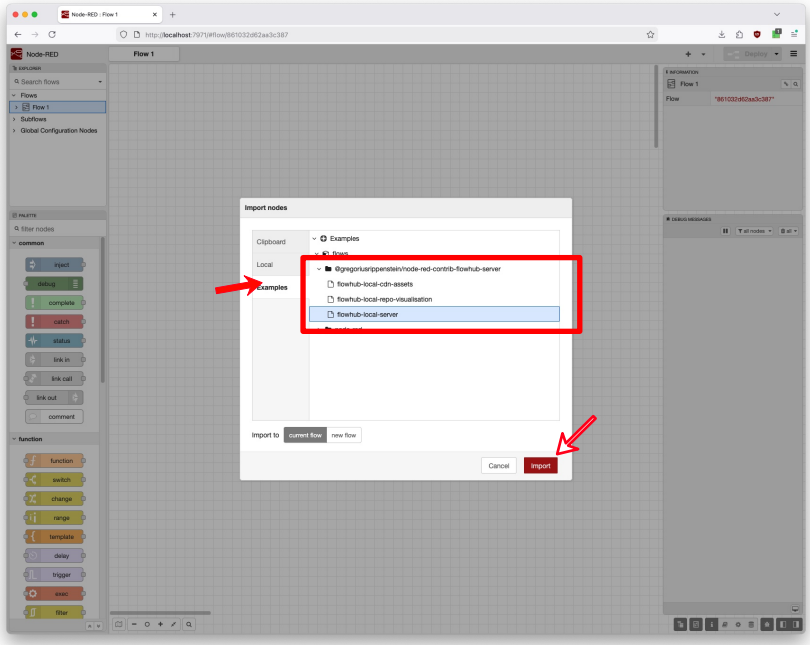
The guide should be used electronically, it has not been designed to be printed - safe trees, not electrons.

Guide Version 1.0

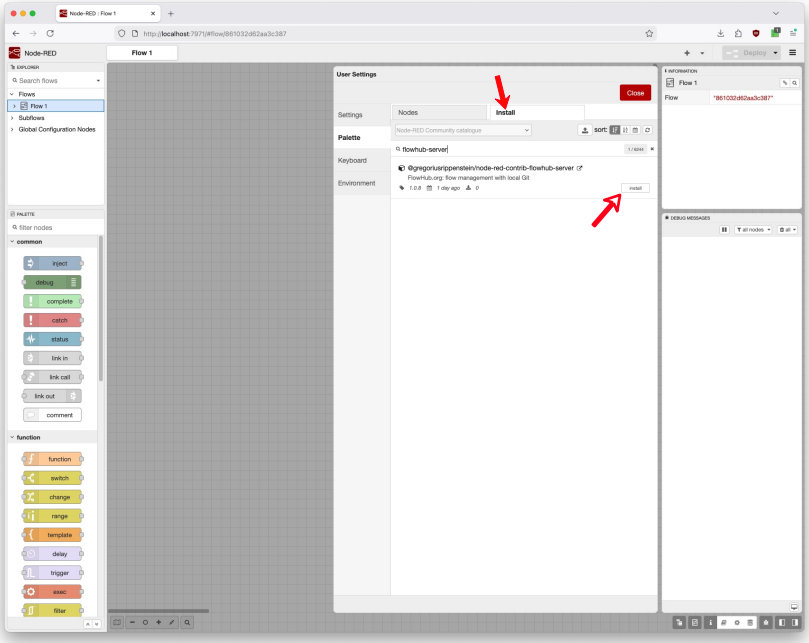
1



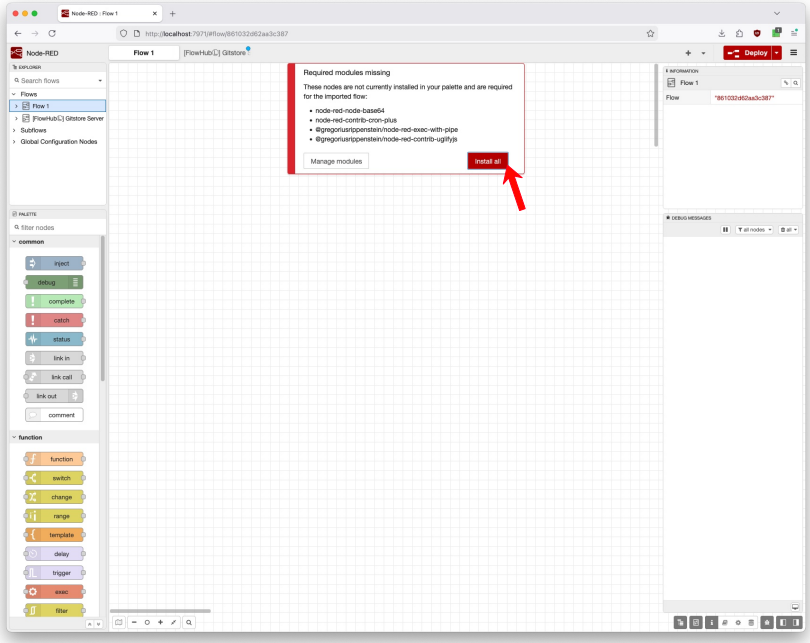
3


Import nodes
Clipboard
Local
Examples
Examples
@gregoriusrippenstein/node-red-contrib-flowhub-server
flowhub-local-cdn-assets
flowhub-local-repo-visualisation
flowhub-local-server
Import to
current flow
new flow
Cancel
Import

2


User Settings
Close
Settings
Nodes
Install
Palette
Keyboard
Environment
flowhub-server
@gregoriusrippenstein/node-red-contrib-flowhub-server
FlowHub.org: flow management with local Git

4


Required modules missing
These nodes are not currently installed in your palette and are required for the imported flow:
node-red-node-base64
node-red-contrib-cron-plus
@gregoriusrippenstein/node-red-exec-with-pipe
@gregoriusrippenstein/node-red-contrib-ugliyis
Manage modules
Install all

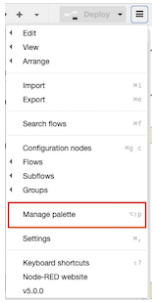
Step 1

Open Node-RED instance, in this case, version 5.0.0.

FlowHub Local gitstore and FlowHub nodes are also compatible with Node-RED v4 and v3.

Step 2

Open the Palette Manager via shortcut (⌥⇧p) or via the hamburger menu (≡) 👉



Switch to the Install tab and search for `flowhub-server`, from there install the node-red-contrib-flowhub-server package.

The package contains no plugins nor nodes, it only provides three example flows that implement the gitstore, the Webiliser and the repository visualisation.

Step 3

Import gitstore server flow.

Open the Flow Import Dialog (⌘i or via ≡), click on `Examples` in the sidebar and then open the flow example list for `flowhub-server` package.

Select and import the example flow `flowhub-local-server`. This provides the core of the gitstore functionality required by FlowHub local.

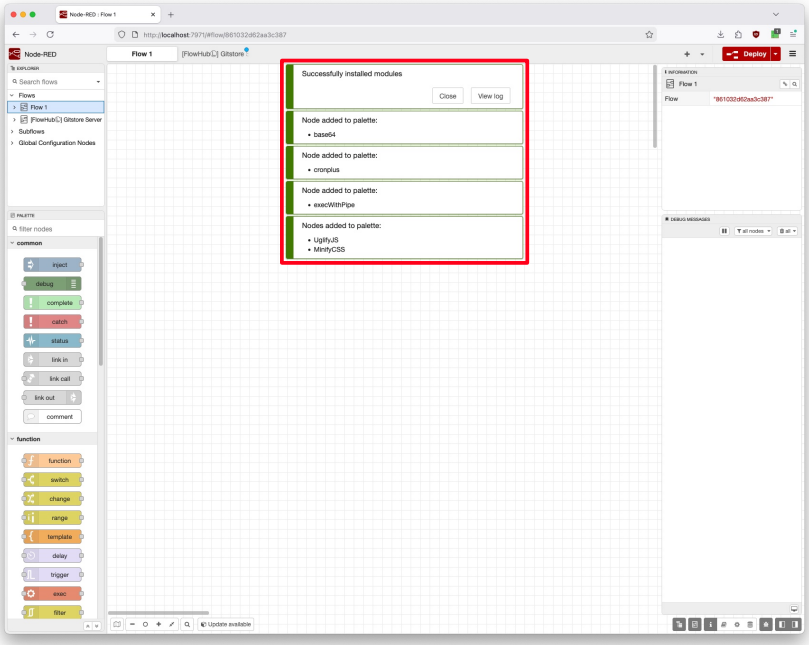
Step 4

Install flow node dependencies.

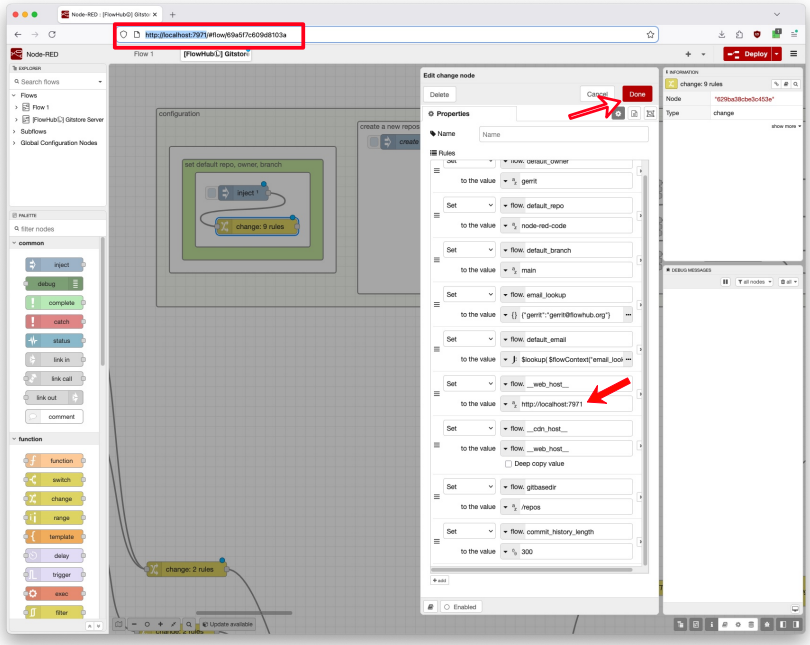
The flow has four other node package dependencies. These are the base64 node, cronplus node, the uglifyjs nodes which provide on-the-fly minification of JS and CSS content and finally the execWithPipe node that allows payload content to be piped to an exec command.

Use the "Install All" button to automate the installation process.

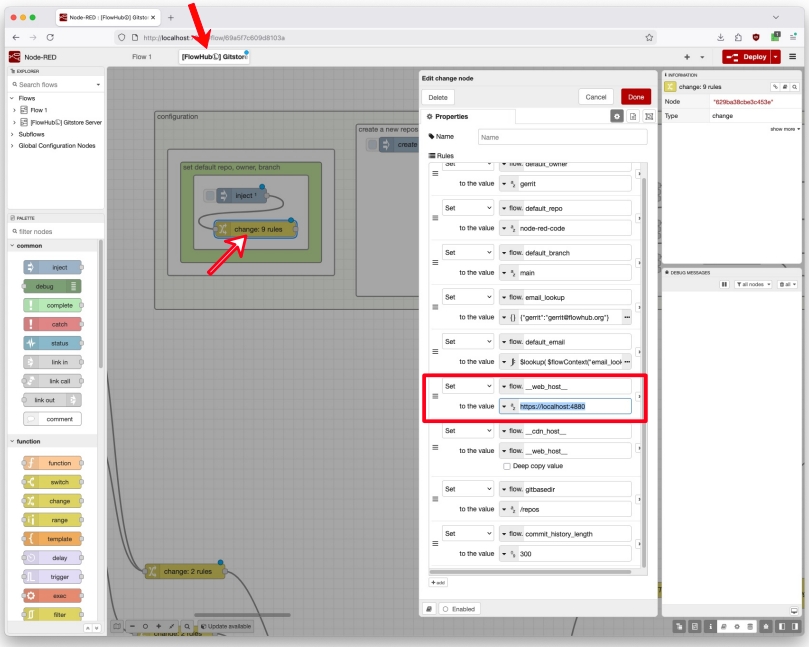
5


Successfully installed modules
Close
View log
Node added to palette:
• base64
Node added to palette:
• cronplus
Node added to palette:
• execWithPipe
Nodes added to palette:
• UglifyJS
• MinifyCSS

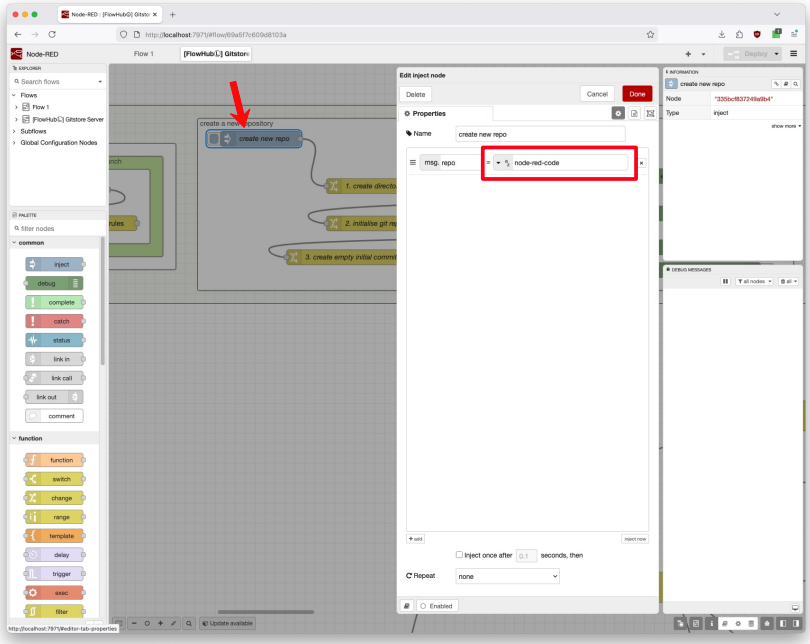
7


http://localhost:7971/#flow/69a57c609d8103a
Edit change node
Delete
Cancel
Done
Properties
Name
Rules
to the value
gerrit
default_repo
node-red-code
default_branch
main
email_lookup
default_email
__web_host__
http://localhost:7971
__cdn_host__
Deep copy value
gitbasedir
/repos
commit_history_length
300
change: 9 rules
change: 2 rules

6


Edit change node
Delete
Cancel
Done
Properties
Name
Rules
to the value
gerrit
default_repo
node-red-code
default_branch
main
email_lookup
default_email
__web_host__
https://localhost:4880
__cdn_host__
Deep copy value
gitbasedir
/repos
commit_history_length
300
change: 9 rules
change: 2 rules

8


Edit inject node
Delete
Cancel
Done
Properties
Name
create new repo
msg. repo
node-red-code
Inject once after
seconds, then
Repeat
none

Step 5

All dependent node packages have been successfully installed.

Step 6

Configuration of gitstore server flow.

Click on the flow tab `[FlowHubⓁ] Gitstore Server`, double click on the change node to open the configuration options in an edit panel.

Here the `__web_host__` configuration has to be set to the hostname of the Node-RED installation. Other configurations can also be altered but first it is important to ensure the gitstore installation works.

The `gitbasedir` configuration defaults to `/repos` which might need changing. This example is based on a docker container, the `/repos` directory has been mapped as a volume which provides a local directory as target.

Step 7

Replace the web host URL of the Node-RED installation and then press done to apply those changes.

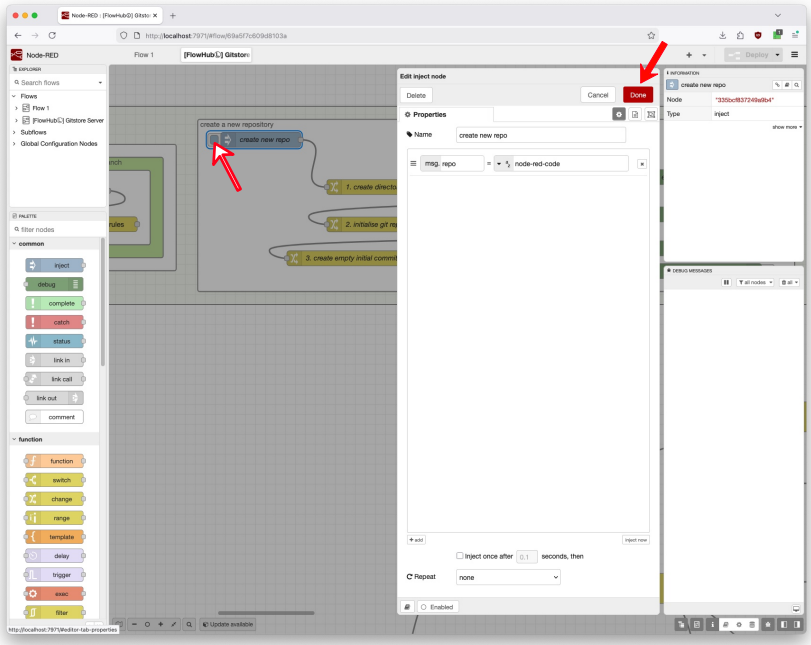
Step 8

Create initial Git repository.

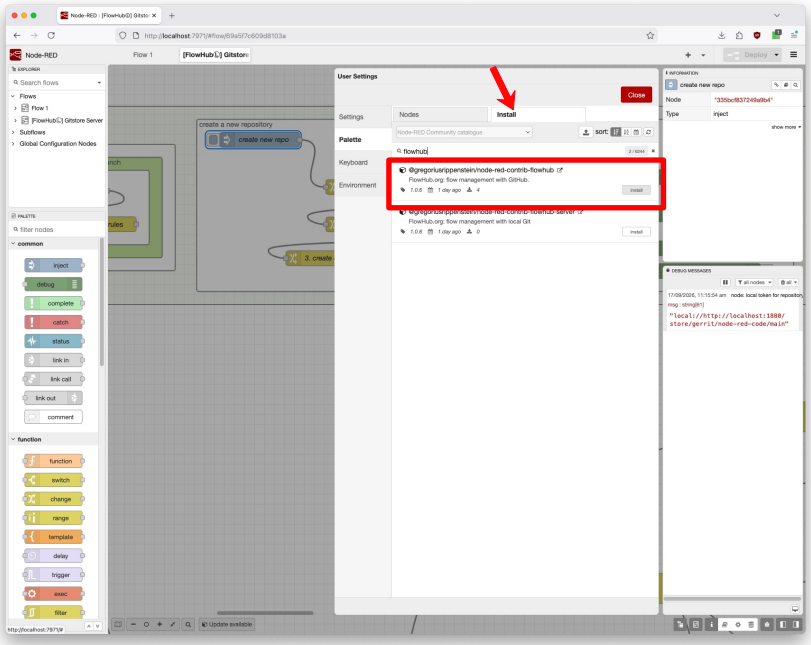
Creating an initial Git repository using the Create Repository button. For that, open the Edit Panel for the Inject button and then keep the default name `node-red-code`. Here alternative names can be used to create new repositories for other projects.

The intention is to have multiple directories for multiple git repositories.

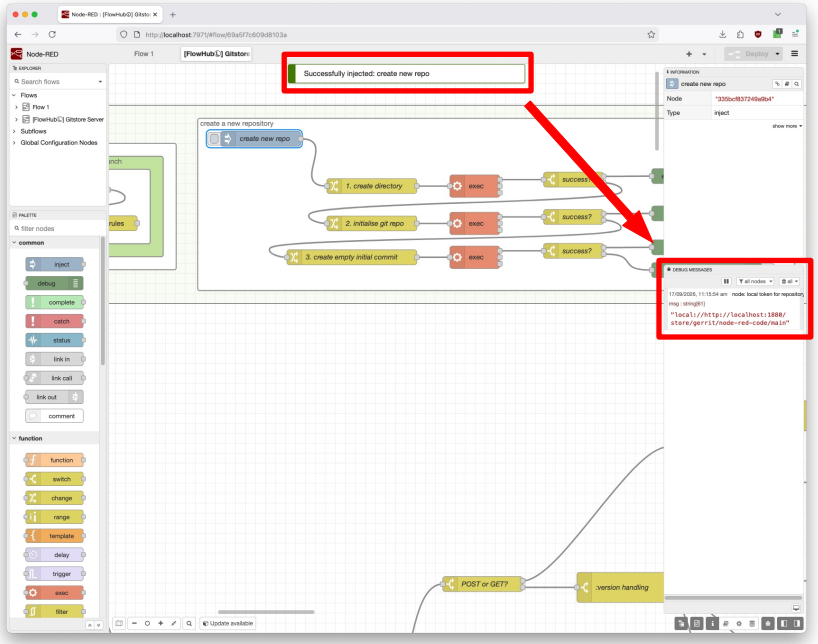
9


Edit inject node
Delete
Cancel
Done
Properties
Name
create new repo
msg. repo
node-red-code
Inject once after
seconds, then
Repeat
none
Enabled

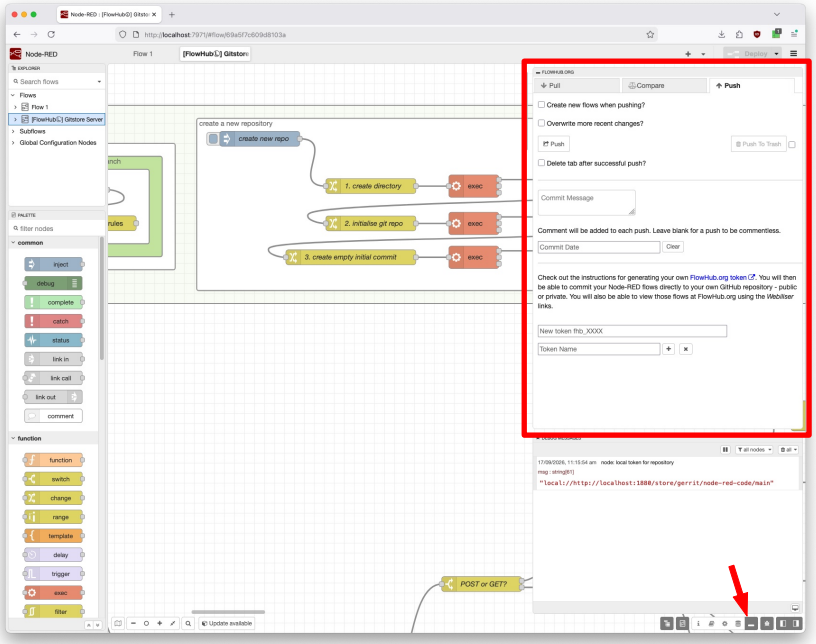
11


User Settings
Close
Settings
Palette
Keyboard
Environment
Nodes
Install
flowhub
@gregoriusrippenstein/node-red-contrib-flowhub
FlowHub.org: flow management with GitHub.
@gregoriusrippenstein/node-red-contrib-flowhub-server
FlowHub.org: flow management with local Git

10


Successfully injected: create new repo
create a new repository
create new repo
1. create directory
2. initialise git repo
3. create empty initial commit
exec
success?
POST or GET?
version handling

12


Pull
Compare
Push
Create new flows when pushing?
Overwrite more recent changes?
Push
Push To Trash
Delete tab after successful push?
Commit Message
Comment will be added to each push. Leave blank for a push to be commentless.
Commit Date
Clear
Check out the instructions for generating your own FlowHub.org token. You will then be able to commit your Node-RED flows directly to your own GitHub repository - public or private. You will also be able to view those flows at FlowHub.org using the Webilster links.
New token fhb_XXXX
Token Name

Step 9

Creating an empty git repository.

Confirm the new repository settings by pressing done and then press the inject button to create the directory inside the `/repos` directory with the name of `node-red-code`.

The directory now contains an initialised git repository with a Git database.

Step 10

Local token creation.

After successfully applying and creating the Git repository, a local token, is shown in the debug panel. This local token will be used to access the Git repository that has just been created.

Step 11

Install FlowHub.org nodes package.

To use that token, we have to install the flowhub.org node package called `node-red-contrib-flowhub`.

This is not the server package we have already installed that server package so now we'll install the client to access the repositories.

The FlowHub.org node package contains two nodes and a sidebar panel for providing three central operations: pull, compare and push.

Step 12

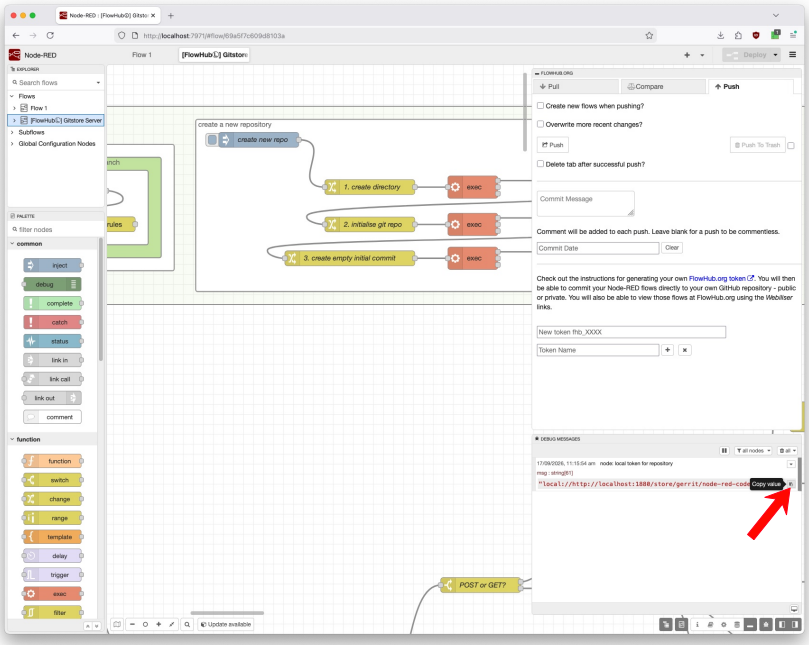
FlowHub nodes sidebar panel.

Once the FlowHub.org nodes have been installed, the FlowHub panel will be shown. The panel is used to:

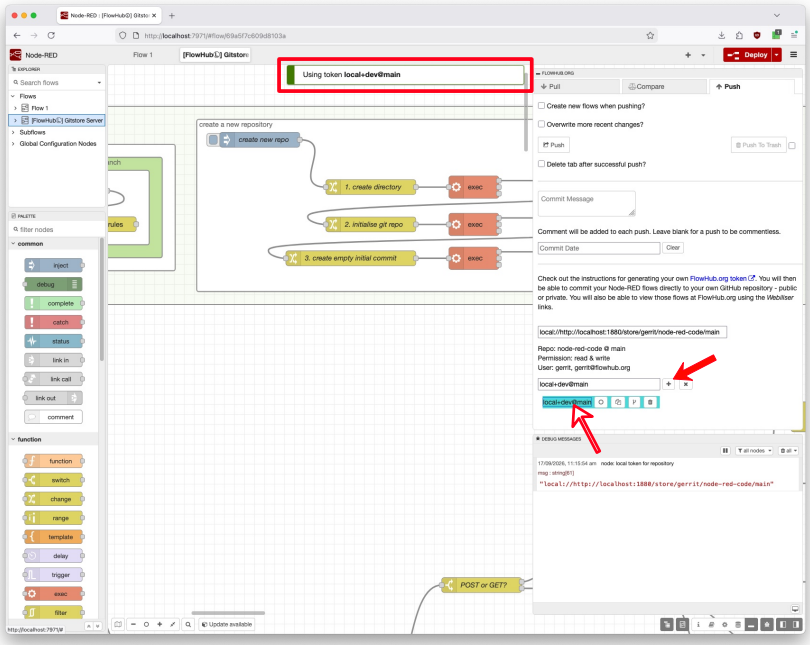
- configure new tokens to be used with the FlowHub nodes,
- visually compare changes made to flows that are under version control,
- pull and push flows from git repositories.

The bottom half of the push tab, remote and local tokens are managed. Here the generated token from step 10 will be added.

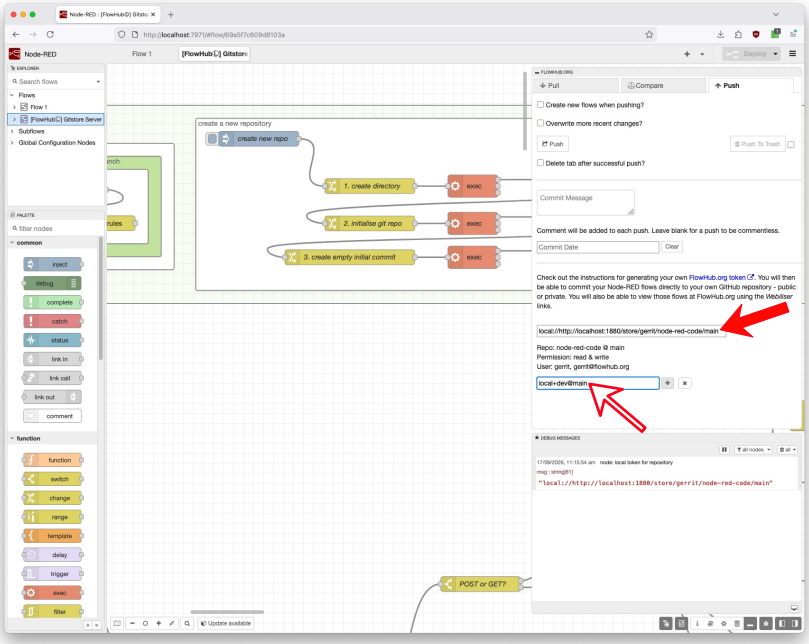
13



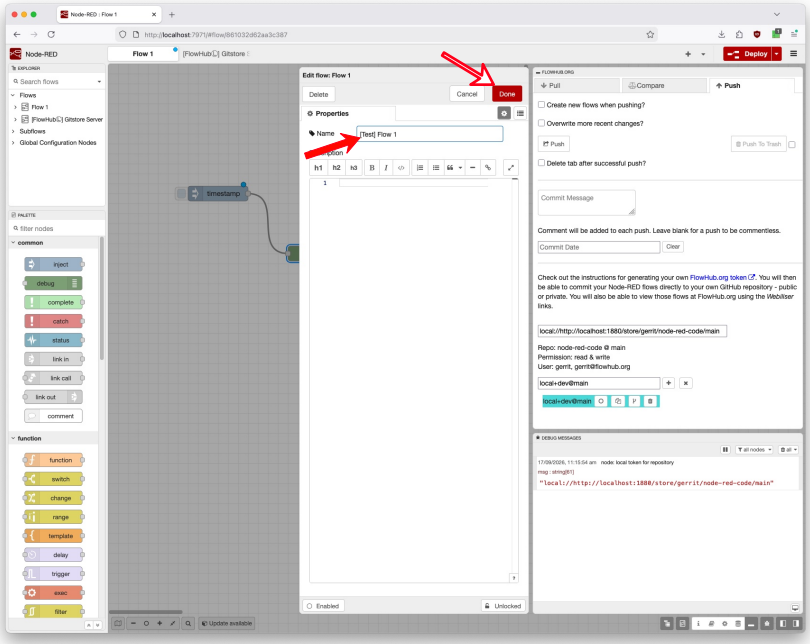
15



14



16



Step 13

Copy the local token for the git repository generated during step 10.

Step 14

Installing the local token.

Paste the token into the FlowHub node field. This is located in under the `Push` tab of the FlowHub.org panel.

If the token is correct then it will verify that with the FlowHub local server and display the details of that token.

Step 15

Naming the local token.

Add a token name, this case `local+dev@main`. Add the token to the token list by clicking on the plus (✚) button next to the name field.

Click on the token name in the token list to activate the token. The token is then highlighted by a aqua blue background and a notification is shown in Node-RED.

Activating the token means that all other actions (push, pull and compare) are done in the context of that token. Each token represents a git repository and a specific branch inside that repository.

There is no naming convention for tokens. In this example the token will be used for local development using the main branch and with that, it's called `local+dev@main`.

Step 16

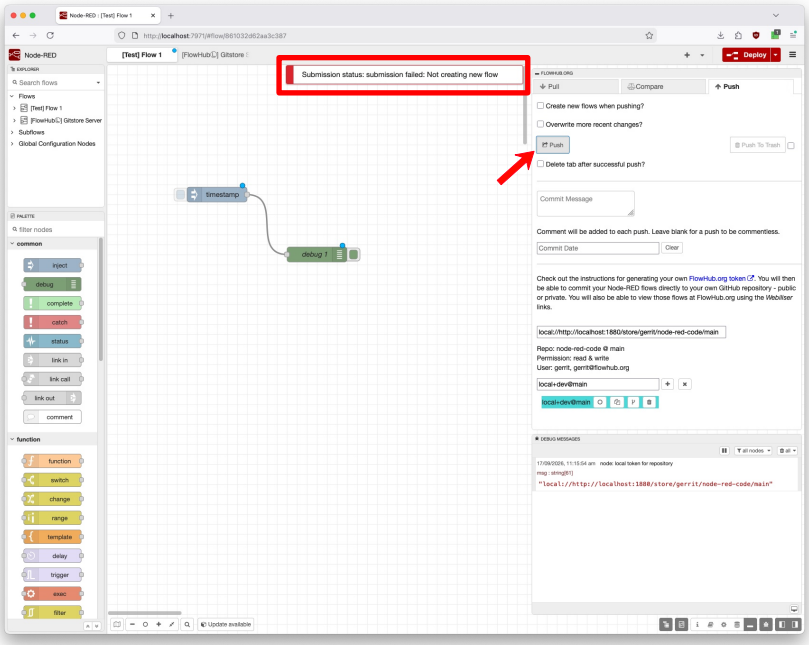
Adding first flow to the git repository.

Double click on the flow tab `Flow 1` to open edit panel. Rename the flow to `[Test] Flow 1`.

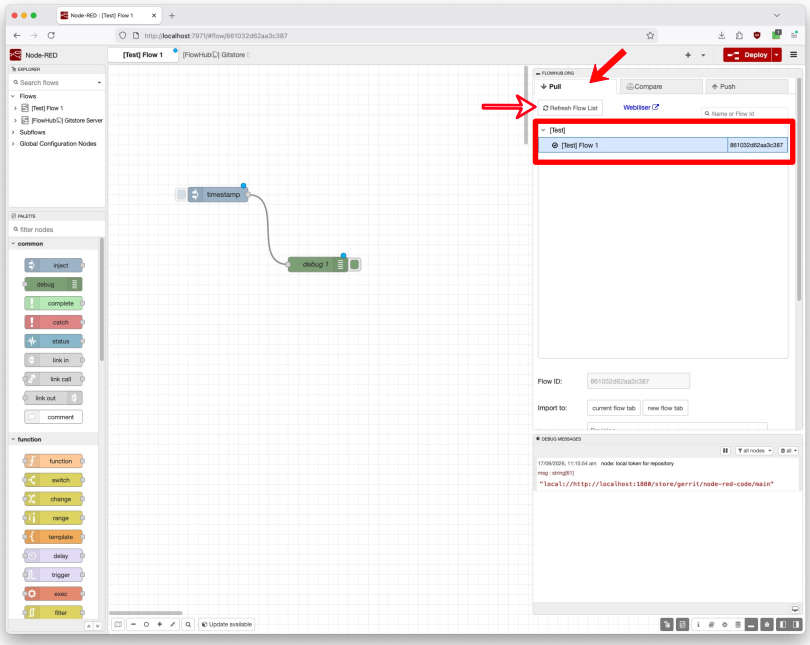
Apply the name change using the 'Done' button.

Flows named using square brackets are grouped by the group name. `[Groupname] Flow 1` and `[Groupname] Flow 2` would be collapsed into a single group of `[Groupname]`.

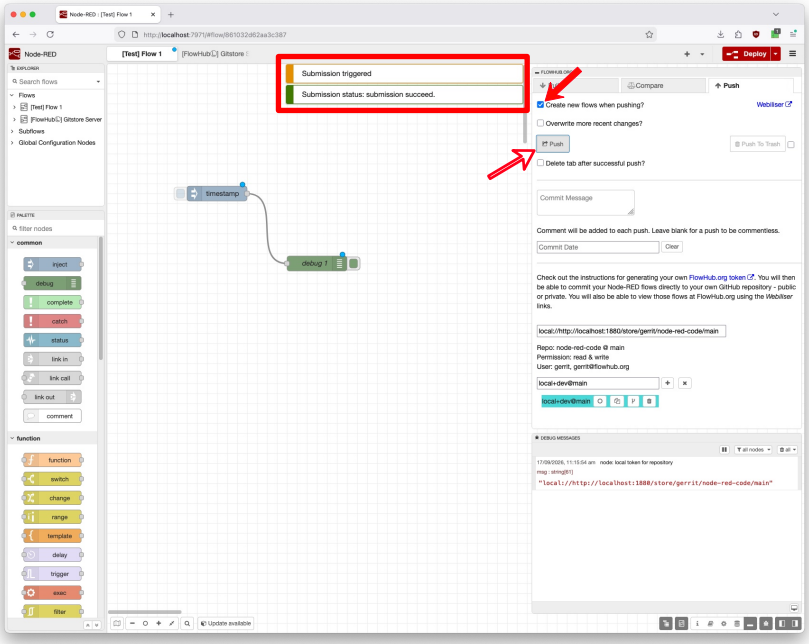
17


Submission status: submission failed: Not creating new flow
Push
timestamp
debug 1

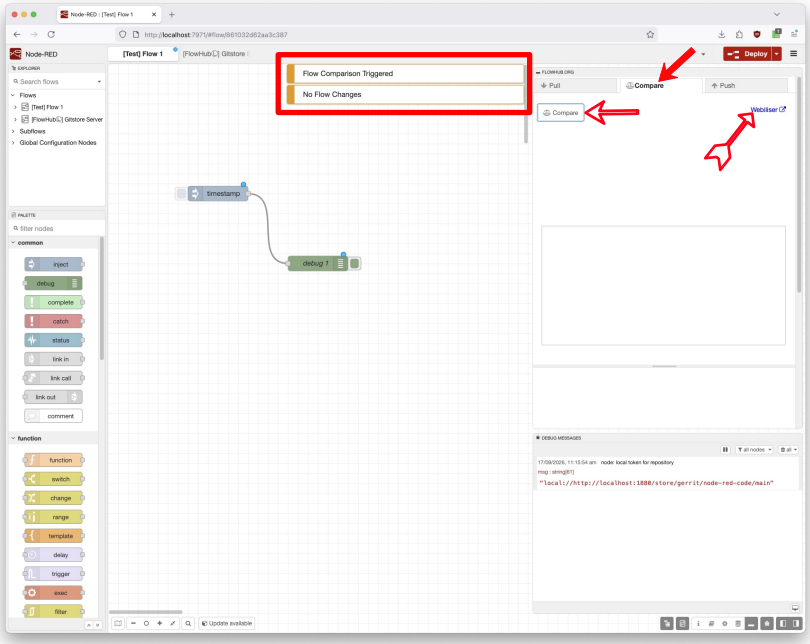
19


Pull
Refresh Flow List
Webliser
[Test]
[Test] Flow 1
timestamp
debug 1

18


Submission triggered
Submission status: submission succeed.
Create new flows when pushing?
Push
timestamp
debug 1

20


Flow Comparison Triggered
No Flow Changes
Compare
Webliser
timestamp
debug 1

Step 17

Here we push the flow as we've created it to the git repository, to the git store. The initial push will fail because the flow hasn't been created yet in the git repository.

The FlowHub.org nodes prevent flows from being created without explicitly user consent. This prevents flows from being added to incorrect git repositories.

Step 18

To give consent that flow should be created, we check the checkbox "Create new flows when pushing" and press the `Push` button.

That then creates the flow code inside the Git repository that we created in step 10. And that's because we're using the token that we generated in step 10.

Step 19

Click on the pull tab and refreshing the list. So when you go to the pull tab, it will either have an initial list of external flows or it will be empty.

Refreshing that list will now generate a list of just one flow and that is the test flow that we've just pushed.

Step 20

Comparing changes from within the Node-RED editor.

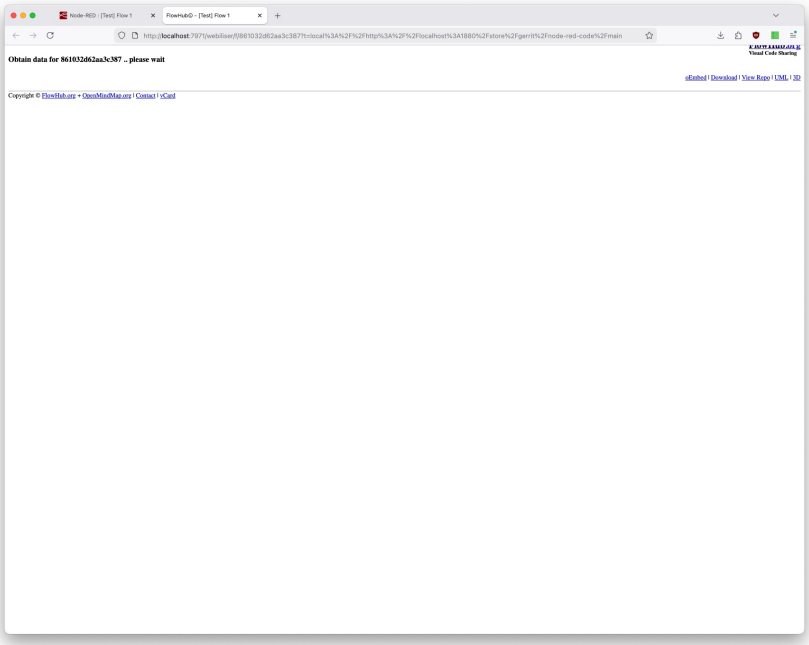
In step 18 we created and pushed our first flow to the change management system, the git store. In step 19 we pulled and checked that that flow was stored in the change management.

In step 20 now we compare our changes in the Node-RED editor to those in the Change Management System and for that we click on the compare tab and then we click on the compare button.

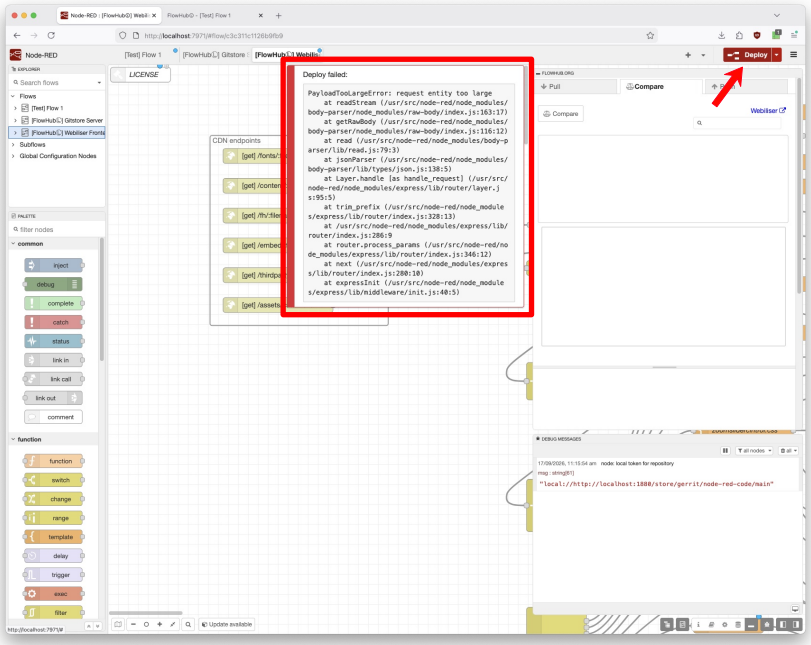
There are no changes as we have not made any changes to the flow in our in the Node-RED editor. What also appears is the webiliser link which is a link to further information on the flow.

We click on the webiliser link.

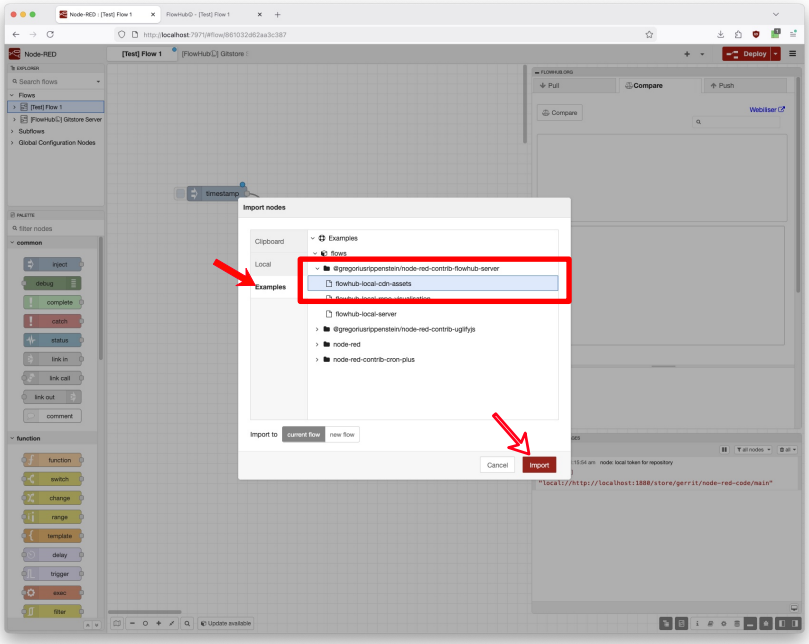
21


Obtain data for 861032d62aa3c387 .. please wait

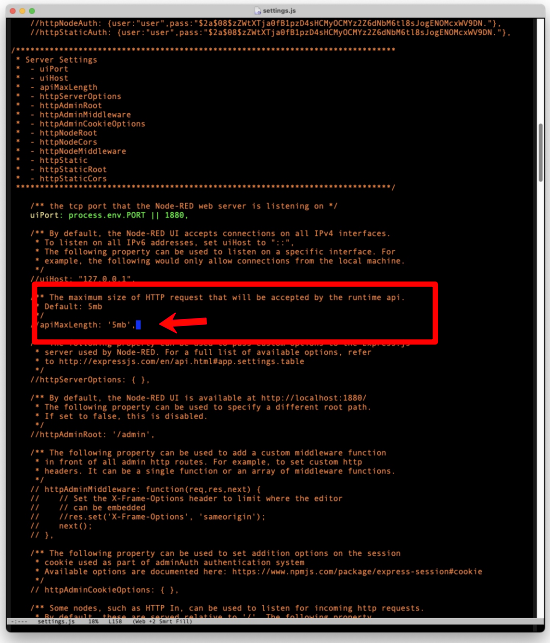
23


Deploy failed:
PayloadTooLargeError: request entity too large

22


Import nodes
Examples
flowhub-local-cdn-assets
Import

24


apiMaxLength: '5mb',

Step 21

Broken Webiliser Page.

When we click on the Webiliser link for the first time, it shows an empty page. That is because the contents, or the front-end code for the Webiliser has not yet been imported. To do that, we need to install a further flow that provides the frontend (Javascript and CSS) assets.

Step 22

Installing the CDN assets for the Webiliser.

Open the Flow Import Dialog (⌘i or via ≡), click on `Examples` in the sidebar and then open the flow example list for `flowhub-server` package.

Click on the flow named `flowhub-local-cdn-assets` and import that flow using the import button.

Step 23

Flow too large for deployment.

The flow cannot be imported as it is too large. Node-RED has a default limit of 5 megabytes (MB) for the size flows. The CDN assets flow contains all the Javascript and CSS required by the Webiliser and that comes to a total of 20 MB.

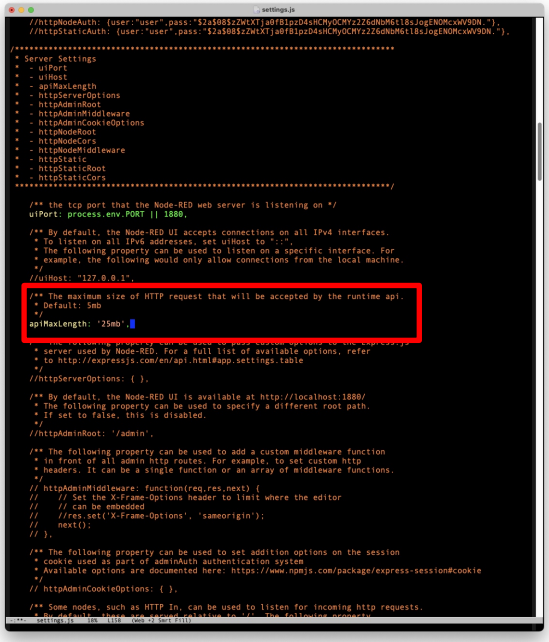
To increase the limit to 25 MB, we need to edit the `settings.js` file of the Node-RED installation.

Step 24

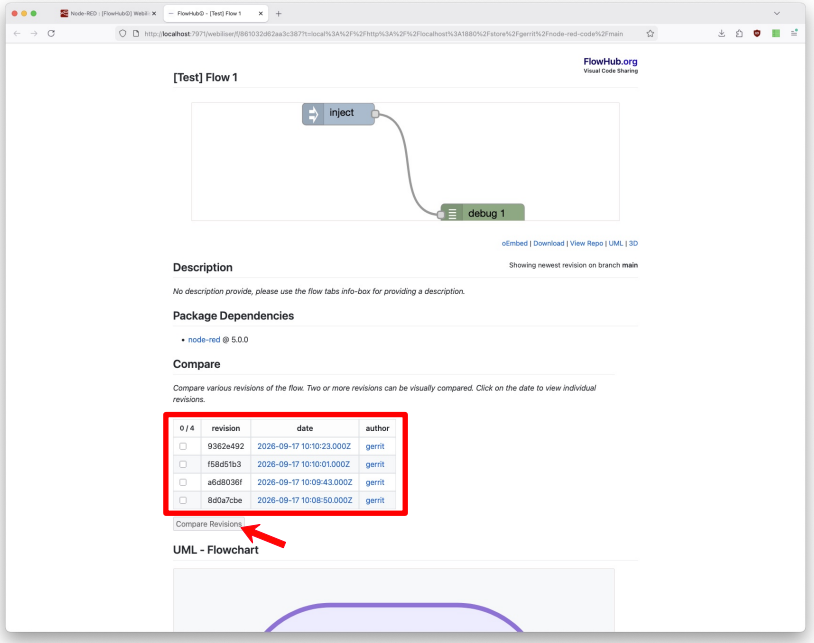
Open and edit the settings.js file.

In the settings.js, search for `apiMaxLength` and uncomment the line `//apiMaxLength: '5mb'` by removing the `//` at the beginning of the line.

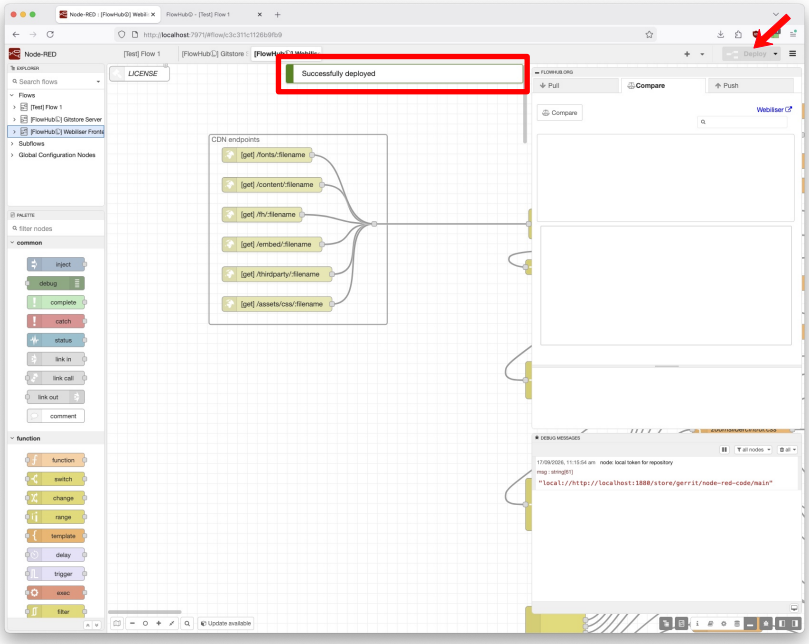
25


settings.js
/** Server Settings
* - uiPort
* - uiHost
* - apiMaxLength
* - httpServerOptions
* - httpAdminRoot
* - httpAdminMiddleware
* - httpAdminCookieOptions
* - httpNodeRoot
* - httpNodeCors
* - httpNodeMiddleware
* - httpStatic
* - httpStaticRoot
* - httpStaticCors
*/
/** the tcp port that the Node-RED web server is listening on */
uiPort: process.env.PORT || 1880,
/** By default, the Node-RED UI accepts connections on all IPv4 interfaces.
* To listen on all IPv6 addresses, set uiHost to "::",
* The following property can be used to listen on a specific interface. For
* example, the following would only allow connections from the local machine.
*/
//uiHost: "127.0.0.1",
/** The maximum size of HTTP request that will be accepted by the runtime api.
* Default: 5mb
*/
apiMaxLength: '25mb',
/** By default, the Node-RED UI is available at http://localhost:1880/
* The following property can be used to specify a different root path.
* If set to false, this is disabled.
*/
//httpAdminRoot: '/admin',
/** The following property can be used to add a custom middleware function
* in front of all admin http routes. For example, to set custom http
* headers. It can be a single function or an array of middleware functions.
*/
// httpAdminMiddleware: function(req,res,next) {
//   // Set the X-Frame-Options header to limit where the editor
//   // can be embedded
//   //res.set('X-Frame-Options', 'sameorigin');
//   next();
// },
/** The following property can be used to set addition options on the session
* cookie used as part of adminAuth authentication system
* Available options are documented here: https://www.npmjs.com/package/express-session#cookie
*/
// httpAdminCookieOptions: { },
/** Some nodes, such as HTTP In, can be used to listen for incoming http requests.

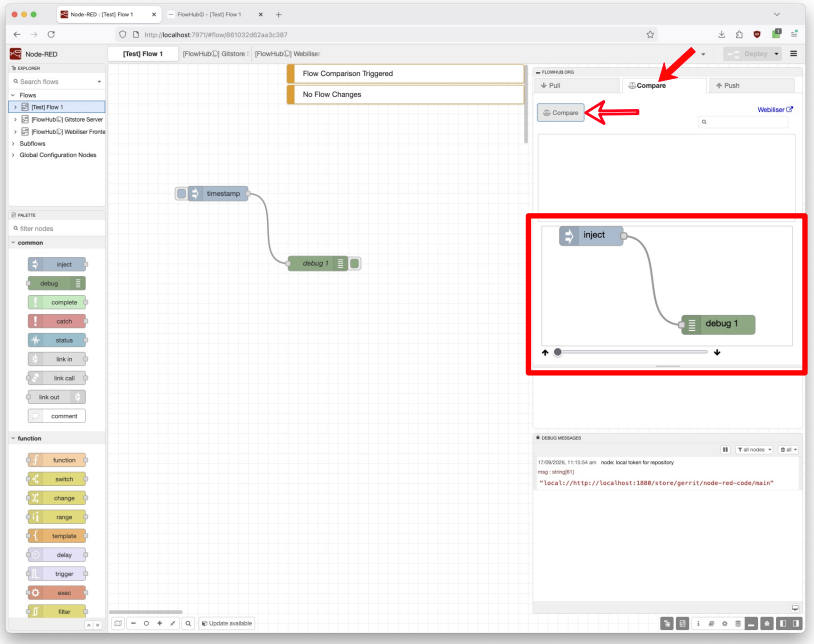
27


[Test] Flow 1
inject
debug 1
FlowHub.org
Visual Code Sharing
oEmbed | Download | View Repo | UML | 3D
Description
Showing newest revision on branch main
No description provide, please use the flow tabs info-box for providing a description.
Package Dependencies
node-red @ 5.0.0
Compare
Compare various revisions of the flow. Two or more revisions can be visually compared. Click on the date to view individual revisions.
0 / 4
revision
date
author
9362e492
2026-09-17 10:10:23.000Z
gerrit
f58d51b3
2026-09-17 10:10:01.000Z
gerrit
a6d8036f
2026-09-17 10:09:43.000Z
gerrit
8d0a7cbe
2026-09-17 10:08:50.000Z
gerrit
Compare Revisions
UML - Flowchart

26


Node-RED
Successfully deployed
CDN endpoints
[get] /fonts/:filename
[get] /content/:filename
[get] /fh/:filename
[get] /embed/:filename
[get] /thirdparty/:filename
[get] /assets/css/:filename
Compare
Webliser
"local://http://localhost:1880/store/gerrit/node-red-code/main"

28


Flow Comparison Triggered
No Flow Changes
timestamp
inject
debug 1
Compare
Webliser
"local://http://localhost:1880/store/gerrit/node-red-code/main"

Step 25

Save the settings.js file.

In the settings.js, search for `apiMaxLength` and uncomment the line `//apiMaxLength: '5mb'` by removing the `//` at the beginning of the line.

Replace the 5mb with 25mb so that the entire line becomes:

```
apiMaxLength: '25mb'
```

Save that change and restart Node-RED for that change to take effect.

Step 26

Redeploy the flow in Node-RED.

In Node-RED, after restarting it, we deploy the CDN assets flow again, and this time it should successfully deploy.

Now all the frontend assets are installed for the Webiliser to work.

Step 27

Reload the Webiliser tab.

Now we go back to the Webiliser tab and reload that tab. And now we see what the Webiliser actually looks like. It contains a visualised version of the flow. It contains a description of the flow. Which is take from the flow tabs information.

We are showing the package dependencies of the flow. Also shown is a list of flow versions of the flow that have been pushed to our change management system.

Finally at the bottom there is a UML representation of the flow as well for informational purposes.

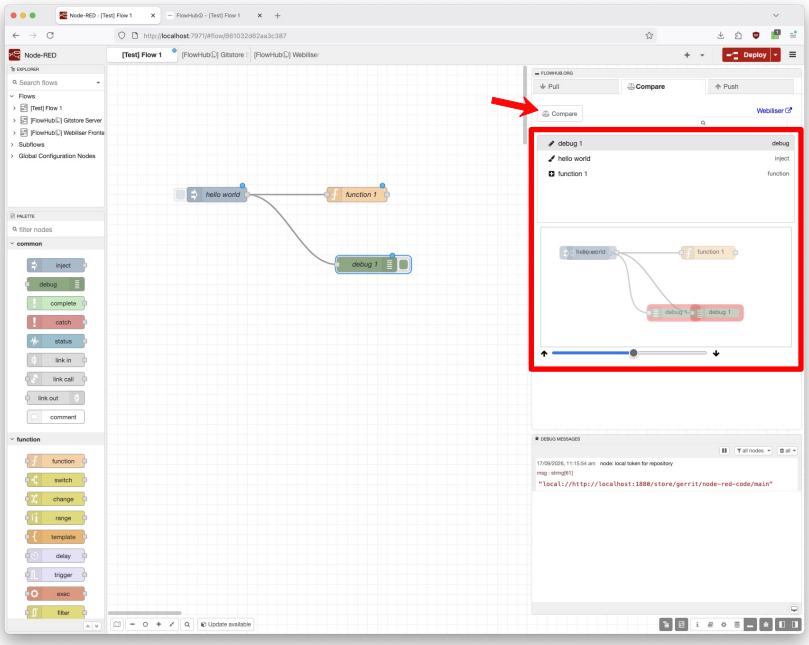
Step 28

Visual comparison inside Node-RED.

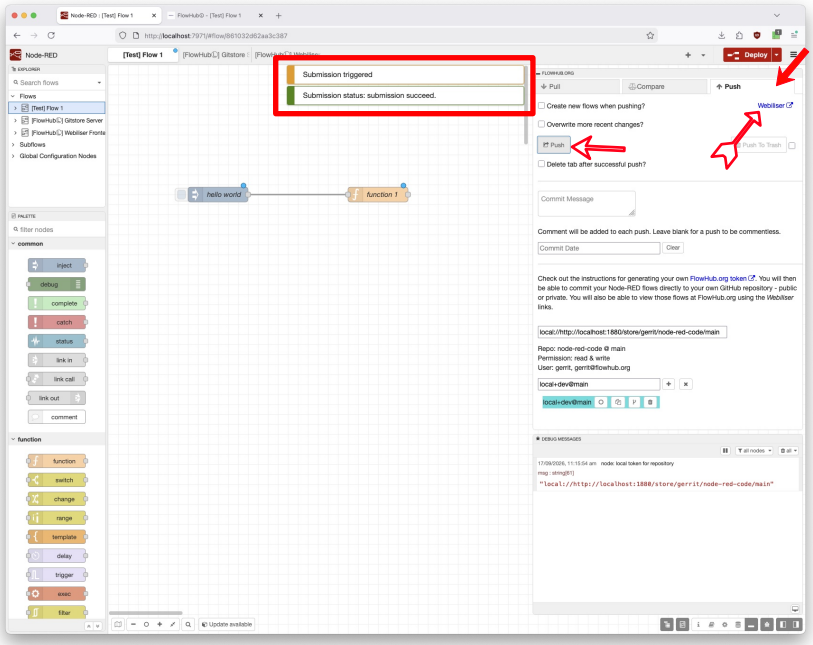
We go back to the compare button inside the Node-RED editor and see there is an image generated which wasn't there when we compared the flow in step 20.

The CDN assets are also used for the Node-RED internal visualisation of flow differences.

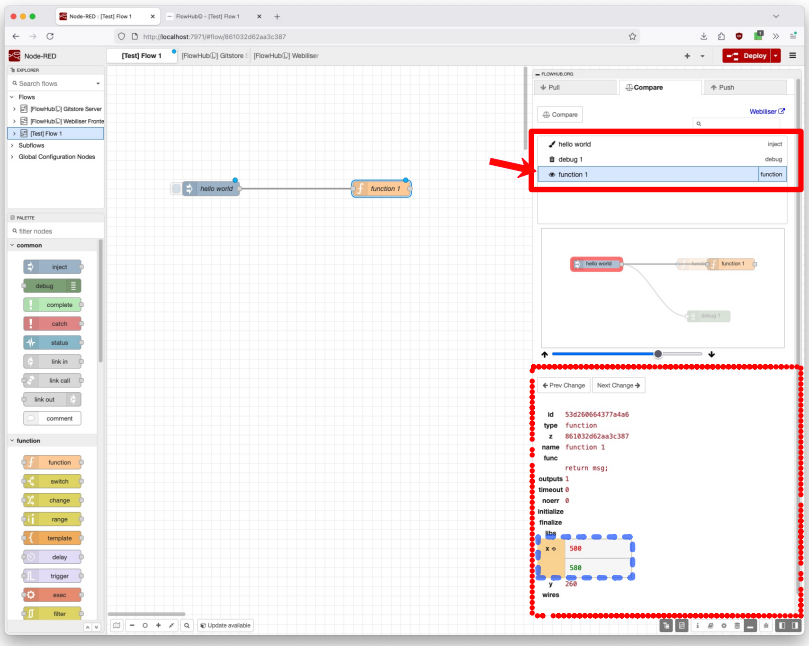
29



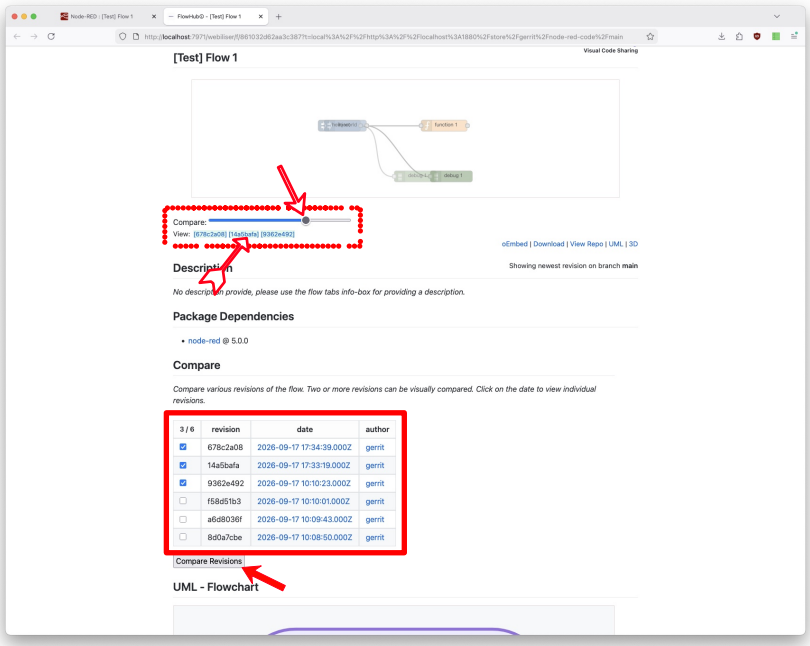
31



30



32



Step 29

Iconification of changes - semantic, visual or both.

Three changes are made to the flow: adding a function node, modifying the name of the inject node and editing the debug node: logging the entire message, not only `msg.payload`.

Clicking on the compare button, we see that these changes are listed in the top half. Each node that has changed is displayed. Each has a different icon to identify the type of change made:

- Pencil (🖉) for nodes that have been semantically changed, meaning their change can modify the workings of the flow.
- Paint brush (🖌) for nodes that have both visual and semantic changes - here the inject node.
- Add (🞣) for nodes that were added - here the function node.

These changes are pushed to the repository using the push button (step not shown).

Step 30

Textual representation of comparison.

Further changes are made to the flow. First the function node is moved to the right, the debug node is removed.

Comparing this change, two more icons are shown:

- Eye (👁) for the visual change made to the function node having been moved.
- Trash (🗑) can for the debug node that was removed.

The inject node is assigned a paint brush (🖌) icon because the wire to the debug node was removed. Removing a wire is consider a change that may or may not have semantic effects.

The visual comparison shows the vanishing debug node and the move of the function node.

Also shown is the textual changes made to the function node (dot rectangle). The exact changes are highlighted by the dashed rectangle. In this case, the `x` attribute went from 500 to 580, implying a move to the right of the node.

There is also an undo (⟲) symbol to allow the change to be reverted. Not all changes can be reverted, but some can.

Step 31

Updating the gitstore with changes.

First click on the push tab, then click on the push button. This pushes the changes to the gitstore server and then to the git repository. Once the change has been stored to the git repository, a notification of success is shown.

Also the Webiliser link underneath the Push tab is updated and clicking on it brings up the Webiliser page for the flow.

Step 32

Comparing flow revisions in the Webiliser.

Multiple revisions of a flow can be compared. Check the first three and click on the compare revision button.

The revision slider is shown is then shown (dotted rectangle) and sliding between revisions can be done using the slider. Also all revisions are displayed as buttons under the slider so that a revision can be shown directly.