


Git Integration

Table of Contents

- [Overview](#)
- [Git Setup and Configuration](#)
- [Creating a Git Repository](#)
- [Cloning a Git Repository](#)
- [Opening an Existing Git Repository](#)
- [Working with Changes](#)
- [Committing Changes](#)
- [Pull, Push, and View Repository Status](#)
- [Git Command Output](#)
- [Switching the Active Repository](#)
- [Deleting a Repository](#)
- [Authentication for Private Online Repositories](#)

 Git integration for Profound.js was added in version 4.10.0.

Overview

Git integration allows you to manage Git repositories within the Visual Designer. Repositories can be created using the Designer, or cloned from GitHub or other online sources. The Designer can track changes to the repository, show diffs, make commits, and more.

Git Setup and Configuration

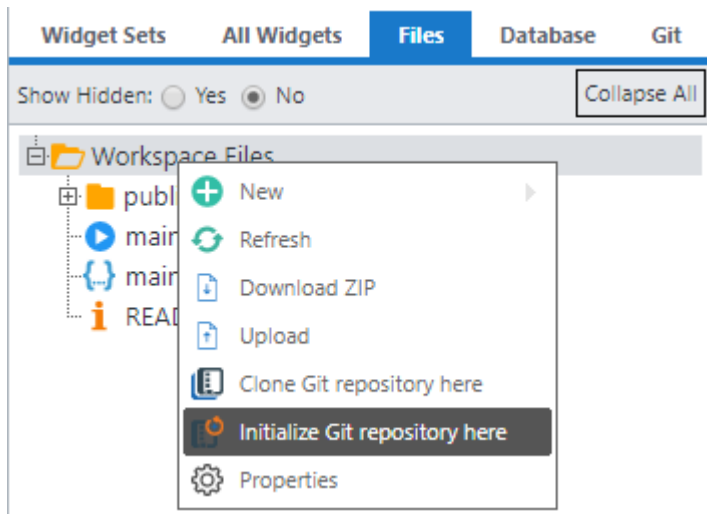
 This section only applies to the Profound.js version of the Visual Designer. On NodeRun.com, Git setup/configuration is taken care of for you.

The Designer uses your system's Git command. If Git is not installed on the system, or not found on your system's PATH environment variable, Git integration features will not be available. If Git is installed in a location not included in the PATH environment variable, the `gitPath` configuration option can be used to tell the Designer where to locate it.

Git requires that you configure your user display name and email address before you can commit changes. See [here](#) for first time Git setup instructions. Git configuration commands can be run in your favorite command shell, or using the Designer's built-in terminal emulator.

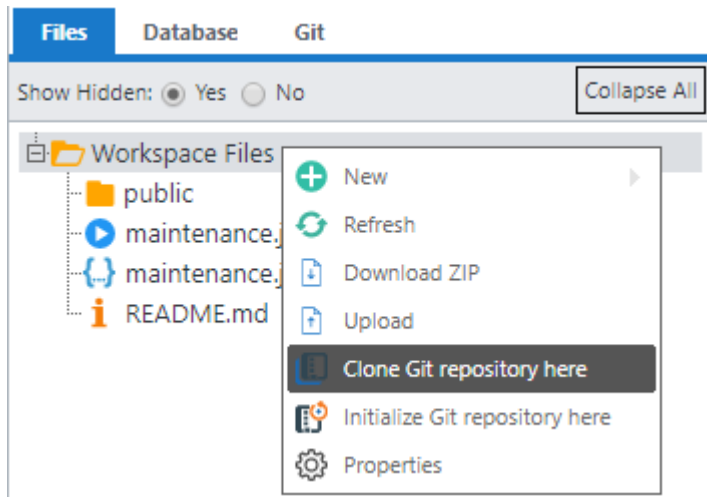
Creating a Git Repository

To create a Git repository, right-click a directory in the Files tree and select [Initialize Git repository here](#).

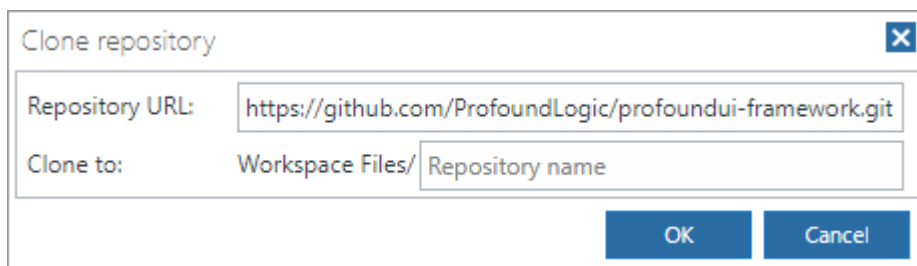


Cloning a Git Repository

To clone a Git repository, right-click the directory where you would like the repository to go and select [Clone Git repository here](#).



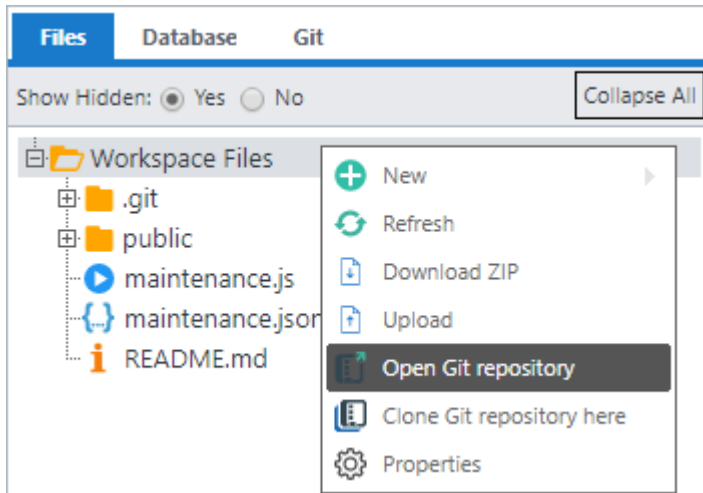
On the Clone Repository dialog, enter a repository URL and optional directory name:



The URL can be given in any format accepted by the Git command – i.e. HTTP/HTTPS, SSH, or a directory path on the local file system. If the 'Clone To' directory name is omitted, the name of the repository will be used as the directory name.

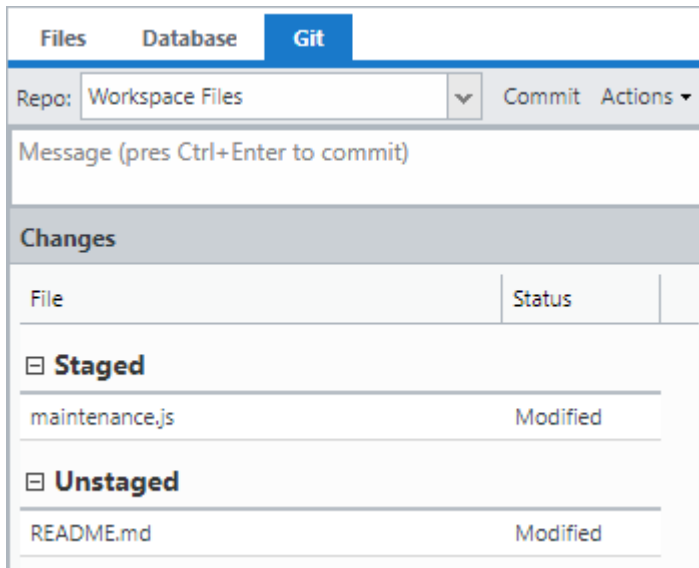
Opening an Existing Git Repository


If you already have a Git repository on the file system, you can open it by right-clicking the directory and selecting [Open Git repository](#).



Working with Changes

The Designer will automatically monitor the repository's working directory and index for changes and display them on the Git Tab:



 Entries are grouped into Staged and Unstaged changes. Multiple items can be selected on the Git tab by shift+click or ctrl+click. Multi-select works only within the same (Staged or Unstaged) group of items.

To stage changes, right-click an item under Unstaged and select [Stage Changes](#).

Files Database **Git**

Repo: Workspace Files Commit Actions

Message (pres Ctrl+Enter to commit)

Changes

File	Status
Staged	
maintenance.js	Modified
Unstaged	
README.md	Modified

- Open File
- Open Changes
- Stage Changes**
- Discard Changes

To unstage changes, right-click an item under Staged and select [Unstage Changes](#).

Files Database **Git**

Repo: Workspace Files Commit Actions

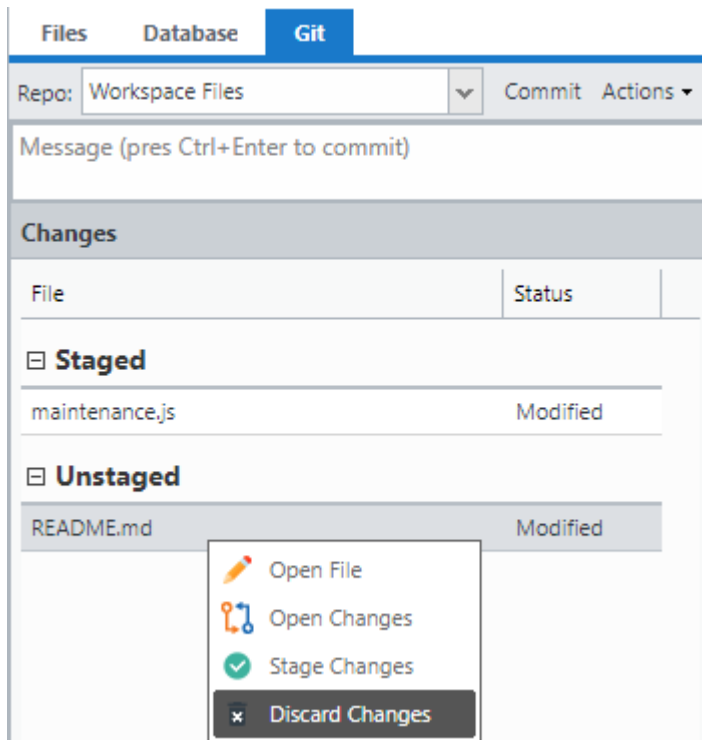
Message (pres Ctrl+Enter to commit)

Changes

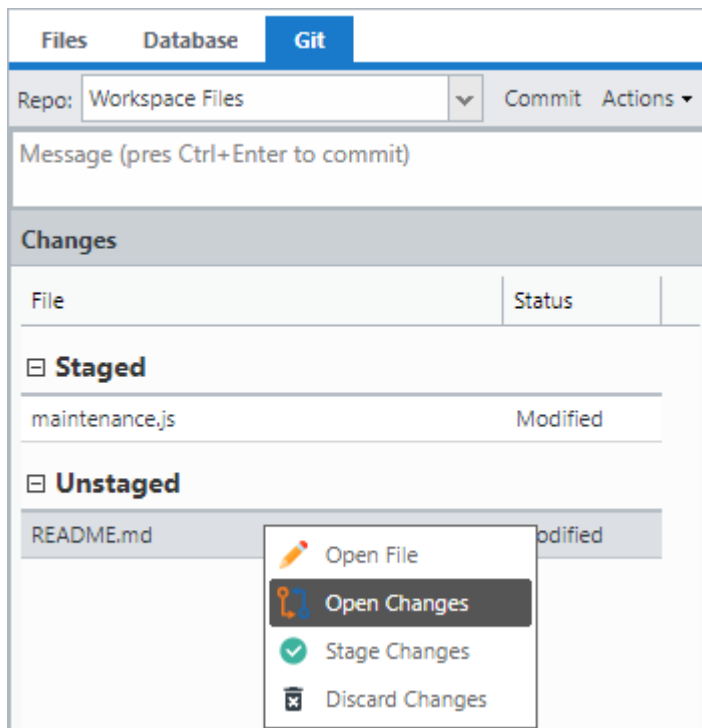
File	Status
Staged	
README.md	Modified
maintenance.js	Modified

- Open File
- Open Changes
- Unstage Changes**

To discard changes, right-click an item under Unstaged and select [Discard Changes](#).



To view changes, right-click an item under either group and select [Open Changes](#).



If the item is for a new file, the file will open in a normal editor. This is the same as selecting [Open File](#). If the item is for a modified file, then a split view will open that displays the differences.

```
1 function maintenance() {
2   pjs.defineDisplay("display", "maintenance.json");
3
4
5   while (!btnExit) {
6     loadGrid();
7     display.main.execute();
8     if (btnAdd) addRecord();
9     else processGrid();
10  }
11
12  // Load current set of records
13  function loadGrid() {
14-   var sql = "SELECT productLine, textDescription FROM productlines";
15    display.grid.replaceRecords(pjs.query(sql));
16  }
17
18  // Process any interaction with the grid
19  function processGrid() {
20    display.grid.readChanged();
21    if (!pjs.endOfData()) {
22      if (iconDelete) deleteRecord(productLine);
23      else if (iconEdit) editDetail(productLine);
24      else if (iconView) viewDetail(productLine);
25    }
26  }
27
28  // Prompt to delete and then delete the record
29  function deleteRecord(productLine) {
30    pjs.prompt("Delete record: " + productLine);
31    if (pjs.confirm("Delete record: " + productLine)) {
32      deleteRecord(productLine);
33    }
34  }
35
36  // Prompt to edit and then edit the record
37  function editDetail(productLine) {
38    pjs.prompt("Edit record: " + productLine);
39    if (pjs.confirm("Edit record: " + productLine)) {
40      editDetail(productLine);
41    }
42  }
43
44  // Prompt to view and then view the record
45  function viewDetail(productLine) {
46    pjs.prompt("View record: " + productLine);
47    if (pjs.confirm("View record: " + productLine)) {
48      viewDetail(productLine);
49    }
50  }
51
52  // Prompt to exit and then exit the application
53  function btnExit() {
54    pjs.prompt("Exit application");
55    if (pjs.confirm("Exit application")) {
56      btnExit();
57    }
58  }
59
60  // Prompt to add and then add a record
61  function addRecord() {
62    pjs.prompt("Add record");
63    if (pjs.confirm("Add record")) {
64      addRecord();
65    }
66  }
67
68  // Prompt to process and then process a record
69  function processGrid() {
70    pjs.prompt("Process record");
71    if (pjs.confirm("Process record")) {
72      processGrid();
73    }
74  }
75
76  // Prompt to load and then load records
77  function loadGrid() {
78    pjs.prompt("Load records");
79    if (pjs.confirm("Load records")) {
80      loadGrid();
81    }
82  }
83
84  // Prompt to display and then display records
85  function display() {
86    pjs.prompt("Display records");
87    if (pjs.confirm("Display records")) {
88      display();
89    }
90  }
91
92  // Prompt to define and then define display
93  function defineDisplay() {
94    pjs.prompt("Define display");
95    if (pjs.confirm("Define display")) {
96      defineDisplay();
97    }
98  }
99
100 }
101
102 // Prompt to define and then define display
103 function defineDisplay() {
104   pjs.prompt("Define display");
105   if (pjs.confirm("Define display")) {
106     defineDisplay();
107   }
108 }
109
110 // Prompt to load and then load records
111 function loadGrid() {
112   pjs.prompt("Load records");
113   if (pjs.confirm("Load records")) {
114     loadGrid();
115   }
116 }
117
118 // Process any interaction with the grid
119 function processGrid() {
120   display.grid.readChanged();
121   if (!pjs.endOfData()) {
122     if (iconDelete) deleteRecord(productLine);
123     else if (iconEdit) editDetail(productLine);
124     else if (iconView) viewDetail(productLine);
125   }
126 }
127
128 // Prompt to delete and then delete the record
129 function deleteRecord(productLine) {
130   pjs.prompt("Delete record: " + productLine);
131   if (pjs.confirm("Delete record: " + productLine)) {
132     deleteRecord(productLine);
133   }
134 }
135
136 // Prompt to edit and then edit the record
137 function editDetail(productLine) {
138   pjs.prompt("Edit record: " + productLine);
139   if (pjs.confirm("Edit record: " + productLine)) {
140     editDetail(productLine);
141   }
142 }
143
144 // Prompt to view and then view the record
145 function viewDetail(productLine) {
146   pjs.prompt("View record: " + productLine);
147   if (pjs.confirm("View record: " + productLine)) {
148     viewDetail(productLine);
149   }
150 }
151
152 // Prompt to exit and then exit the application
153 function btnExit() {
154   pjs.prompt("Exit application");
155   if (pjs.confirm("Exit application")) {
156     btnExit();
157   }
158 }
159
160 // Prompt to add and then add a record
161 function addRecord() {
162   pjs.prompt("Add record");
163   if (pjs.confirm("Add record")) {
164     addRecord();
165   }
166 }
167
168 // Prompt to process and then process a record
169 function processGrid() {
170   pjs.prompt("Process record");
171   if (pjs.confirm("Process record")) {
172     processGrid();
173   }
174 }
175
176 // Prompt to load and then load records
177 function loadGrid() {
178   pjs.prompt("Load records");
179   if (pjs.confirm("Load records")) {
180     loadGrid();
181   }
182 }
183
184 // Prompt to display and then display records
185 function display() {
186   pjs.prompt("Display records");
187   if (pjs.confirm("Display records")) {
188     display();
189   }
190 }
191
192 // Prompt to define and then define display
193 function defineDisplay() {
194   pjs.prompt("Define display");
195   if (pjs.confirm("Define display")) {
196     defineDisplay();
197   }
198 }
199
200 }
201
202 // Prompt to define and then define display
203 function defineDisplay() {
204   pjs.prompt("Define display");
205   if (pjs.confirm("Define display")) {
206     defineDisplay();
207   }
208 }
209
210 // Prompt to load and then load records
211 function loadGrid() {
212   pjs.prompt("Load records");
213   if (pjs.confirm("Load records")) {
214     loadGrid();
215   }
216 }
217
218 // Process any interaction with the grid
219 function processGrid() {
220   display.grid.readChanged();
221   if (!pjs.endOfData()) {
222     if (iconDelete) deleteRecord(productLine);
223     else if (iconEdit) editDetail(productLine);
224     else if (iconView) viewDetail(productLine);
225   }
226 }
227
228 // Prompt to delete and then delete the record
229 function deleteRecord(productLine) {
230   pjs.prompt("Delete record: " + productLine);
231   if (pjs.confirm("Delete record: " + productLine)) {
232     deleteRecord(productLine);
233   }
234 }
235
236 // Prompt to edit and then edit the record
237 function editDetail(productLine) {
238   pjs.prompt("Edit record: " + productLine);
239   if (pjs.confirm("Edit record: " + productLine)) {
240     editDetail(productLine);
241   }
242 }
243
244 // Prompt to view and then view the record
245 function viewDetail(productLine) {
246   pjs.prompt("View record: " + productLine);
247   if (pjs.confirm("View record: " + productLine)) {
248     viewDetail(productLine);
249   }
250 }
251
252 // Prompt to exit and then exit the application
253 function btnExit() {
254   pjs.prompt("Exit application");
255   if (pjs.confirm("Exit application")) {
256     btnExit();
257   }
258 }
259
260 // Prompt to add and then add a record
261 function addRecord() {
262   pjs.prompt("Add record");
263   if (pjs.confirm("Add record")) {
264     addRecord();
265   }
266 }
267
268 // Prompt to process and then process a record
269 function processGrid() {
270   pjs.prompt("Process record");
271   if (pjs.confirm("Process record")) {
272     processGrid();
273   }
274 }
275
276 // Prompt to load and then load records
277 function loadGrid() {
278   pjs.prompt("Load records");
279   if (pjs.confirm("Load records")) {
280     loadGrid();
281   }
282 }
283
284 // Prompt to display and then display records
285 function display() {
286   pjs.prompt("Display records");
287   if (pjs.confirm("Display records")) {
288     display();
289   }
290 }
291
292 // Prompt to define and then define display
293 function defineDisplay() {
294   pjs.prompt("Define display");
295   if (pjs.confirm("Define display")) {
296     defineDisplay();
297   }
298 }
299
300 }
```

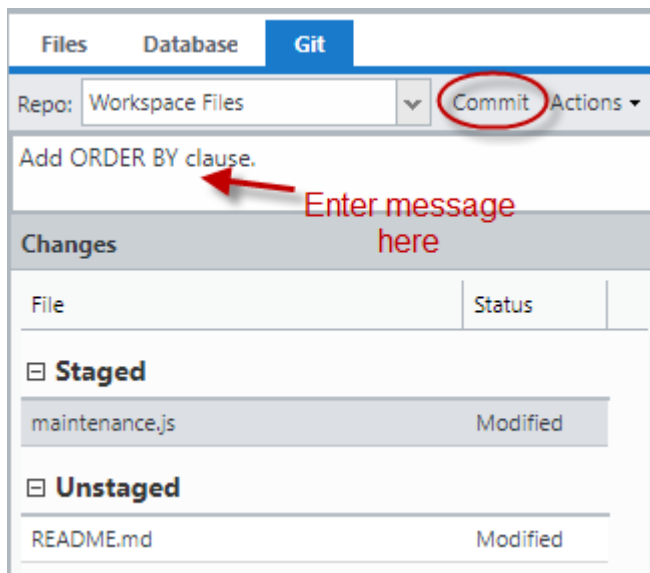
If viewing a diff in the working tree (Unstaged), then edits can be made on the right-side. If viewing a diff in the index (Staged), then the diff view is read-only. The buttons at the top of the editor panel can be used to toggle side-by-side vs line by line view.

Committing Changes

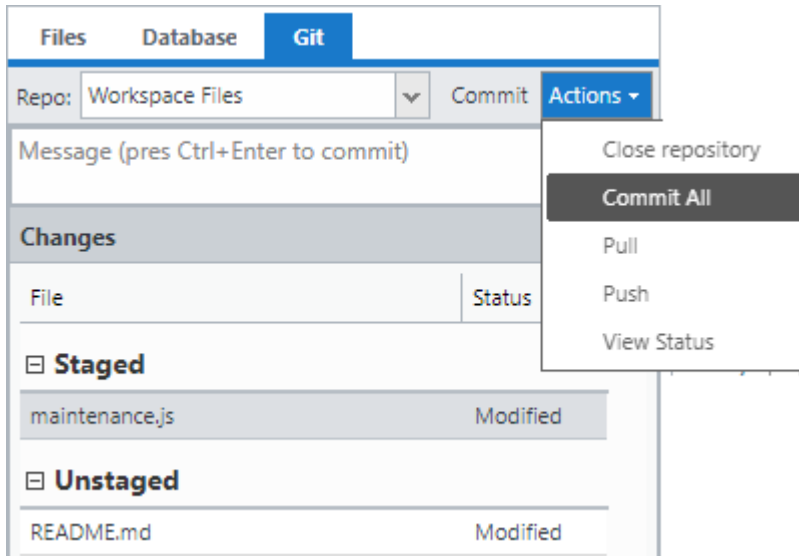
There are two ways to commit changes:

- Commit staged changes only
- Commit all changes

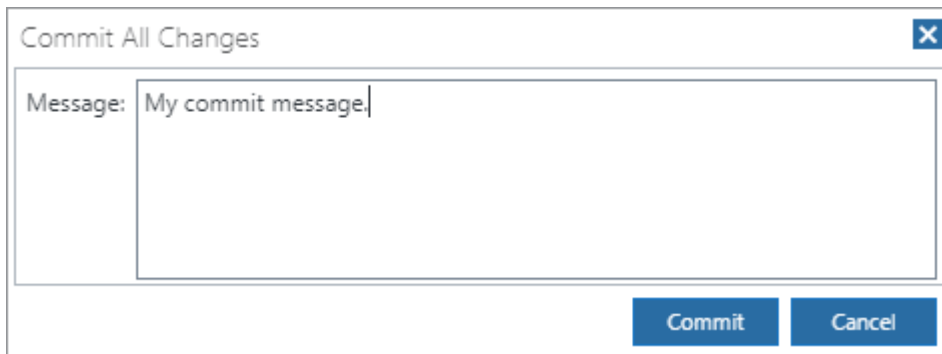
When there are staged changes, a message box and Commit button will appear at the top of the Git panel. To commit staged changes, enter a commit message and press ctrl+Enter or click **Commit**.



To all changes (both Staged and Unstaged), use the **Commit All** option on the **Actions** drop down:

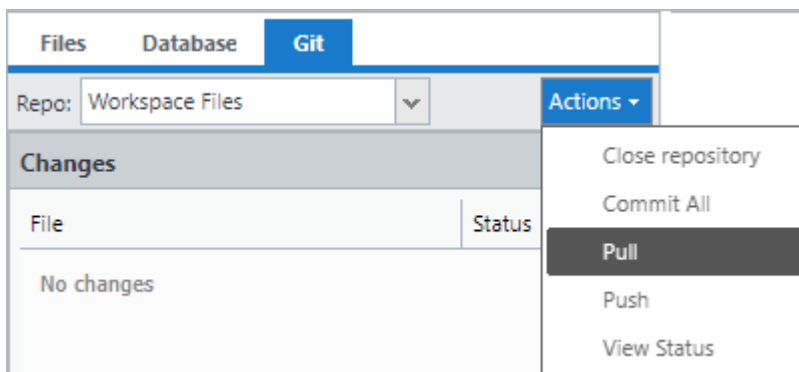


Enter your message in the Commit All Changes dialog and click [Commit](#).



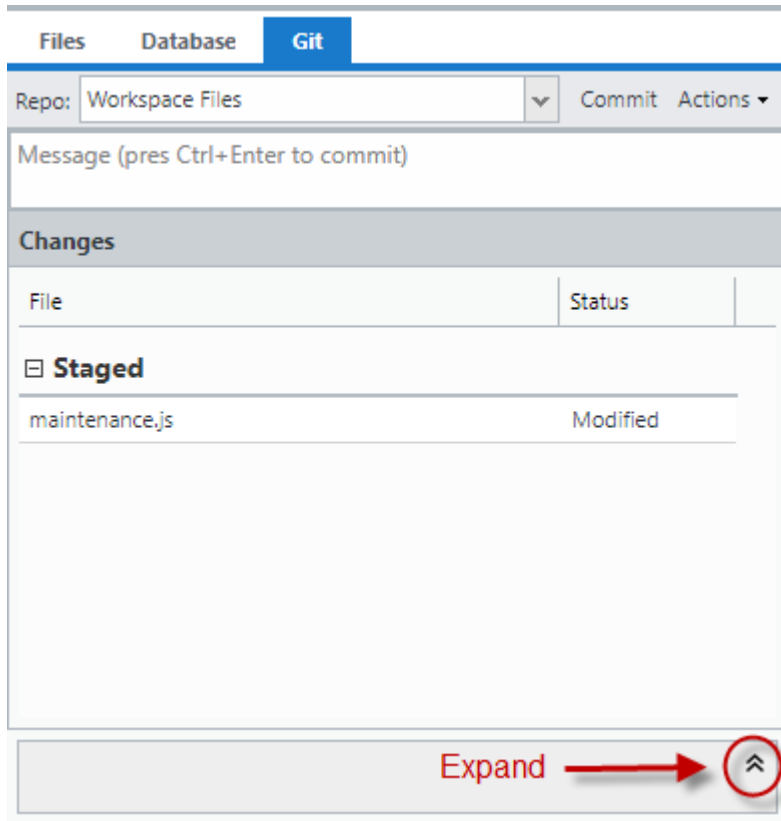
Pull, Push, and View Repository Status

You can pull/push to/from a remote and view repository status using the options on the [Actions](#) menu.

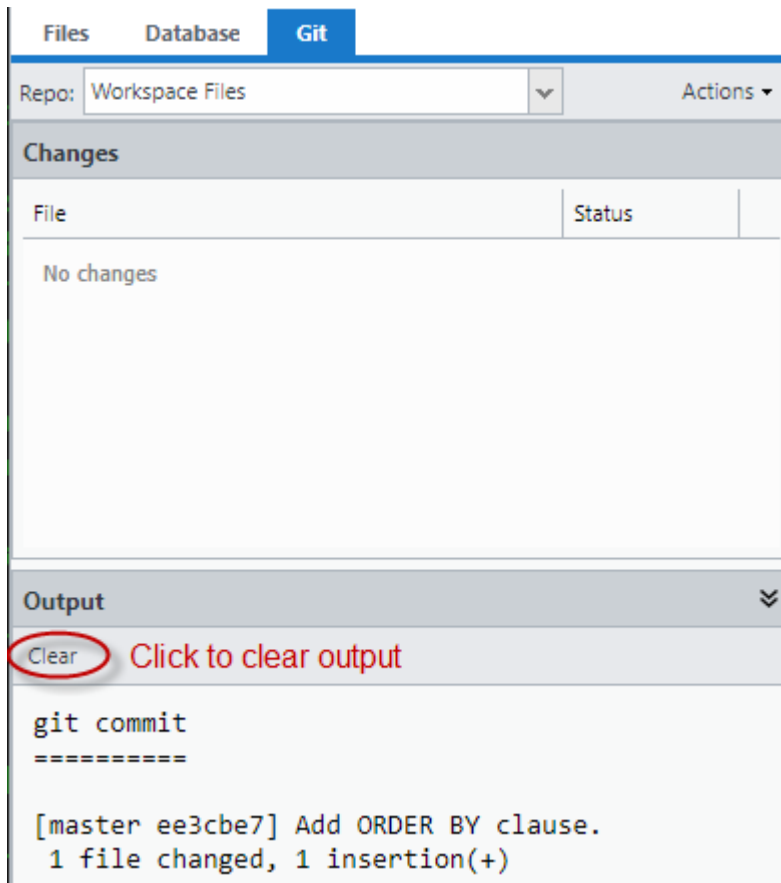


Git Command Output

Output from Git for all actions is displayed in the output panel at the bottom of the Git tab. If the panel is collapsed, click on the arrow icon to expand it:



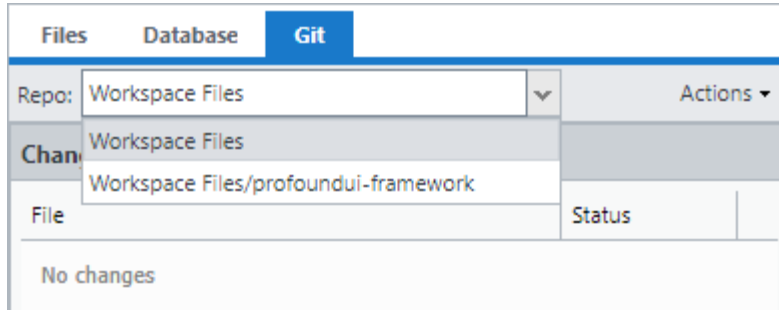
Output from all commands run during your Designer session is retained. Click the **Clear** button at the top of the output panel to clear previous messages.



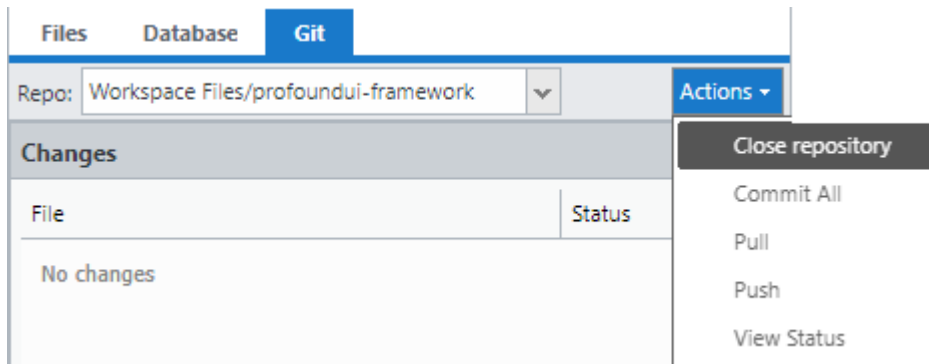
The output panel will automatically expand if there is an error when running a command.

Switching the Active Repository

The Designer stores a list of repositories that you have worked with. To change the active repository, use the drop down box at the top of the Git tab.



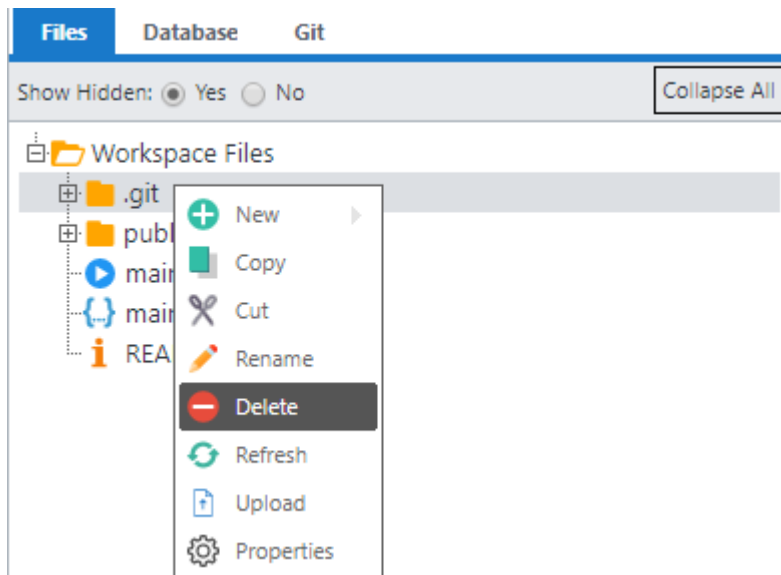
If you no longer need to work with a repository and want to remove it from the list, use the **Close** option on the **Actions** menu.



This option does not delete the repository, it simply removes it from your list of repositories to work with.

Deleting a Repository

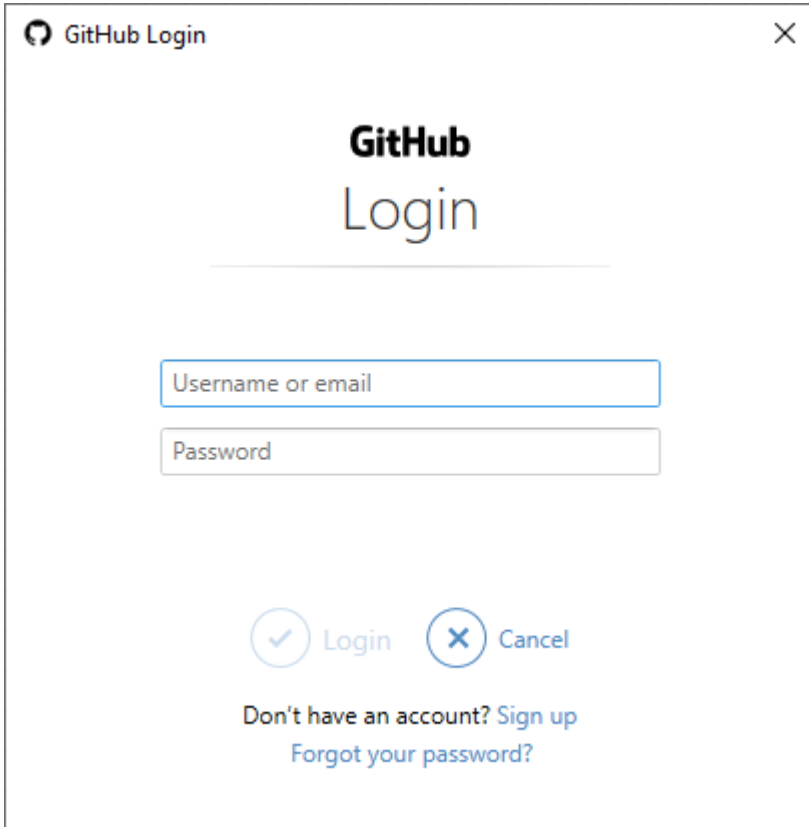
Git repositories are stored in a hidden directory named '.git'. To delete a repository using the Files tab, select **Show Hidden = Yes**, right-click the .git directory, and select **Delete**.



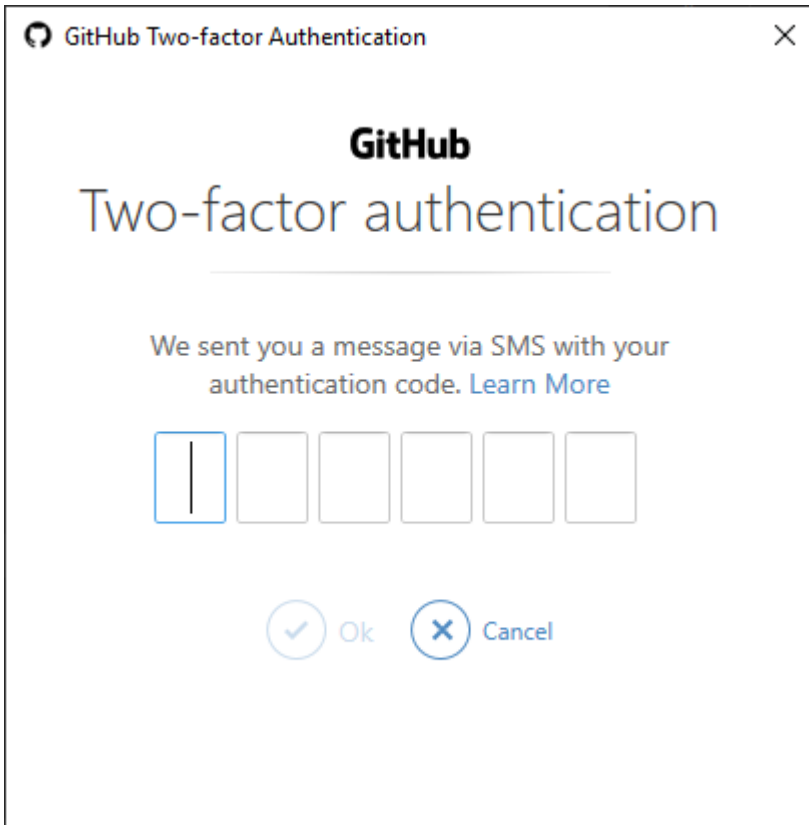
Authentication for Private Online Repositories

To clone and to pull/push to/from online (i.e. GitHub) repositories that require authentication, it's recommended to use SSH keys. See [here](#) for instructions on setting up SSH key authentication with GitHub. You can use your favorite command shell or the Designer's built-in terminal emulator to run the commands for creating/inspecting keys.

Profound.js on Windows also supports authentication over HTTP/HTTPS for GitHub, including 2-factor authentication. This is not supported on other platforms. Git for Windows will automatically prompt for authentication as needed.



The screenshot shows a dialog box titled "GitHub Login" with a close button in the top right corner. The main heading is "GitHub Login". Below the heading are two input fields: "Username or email" and "Password". At the bottom, there are two buttons: "Login" (with a checkmark icon) and "Cancel" (with an 'X' icon). Below the buttons, there are two links: "Don't have an account? Sign up" and "Forgot your password?".



The screenshot shows a dialog box titled "GitHub Two-factor Authentication" with a close button in the top right corner. The main heading is "GitHub Two-factor authentication". Below the heading, there is a message: "We sent you a message via SMS with your authentication code. [Learn More](#)". Below the message are six input boxes for the authentication code, with the first box containing a vertical line. At the bottom, there are two buttons: "Ok" (with a checkmark icon) and "Cancel" (with an 'X' icon).

