Lianja 11 Team Development

Overview

Team development in software engineering refers to the structured process of forming, managing, and optimizing a group of software engineers and related roles (e.g., testers, product managers, designers) to collaboratively design, develop, test, and deliver software products or systems. This process covers both the *human* and *technical* aspects of collaboration and aims to enhance productivity, code quality, and delivery efficiency.

Team Collaboration in Lianja refers to the integrated tools and workflows that allow multiple developers to work together efficiently on the same codebase within the **Lianja App Builder.** Lianja have built powerful collaboration features into Lianja 11.

Here's a breakdown of how teams can collaborate effectively during development using Lianja App Builder:

Version Control Integration

Git and GitHub

- **Built-in Git support** in Lianja App Builder allows for cloning, committing, pushing, pulling, branching, and merging directly from the Lianja App Builder IDE.
- Teams can use **GitHub repositories** to collaborate and manage pull requests (PRs), track issues, and host wikis.

Git will be enabled if you have installed it. You can install it like this:

Windows

Go to https://git-scm.com/download/win and download the installer.

Linux Ubuntu / Debian-based

```shell
sudo apt update
sudo apt install git
```

Linux Fedora

```shell
sudo dnf install git
```

Linux CentOS / RHEL

```shell
sudo yum install git
```

Arch Linux / Manjaro

```shell
sudo pacman -S git
```

Verify installation

Shell

git --version

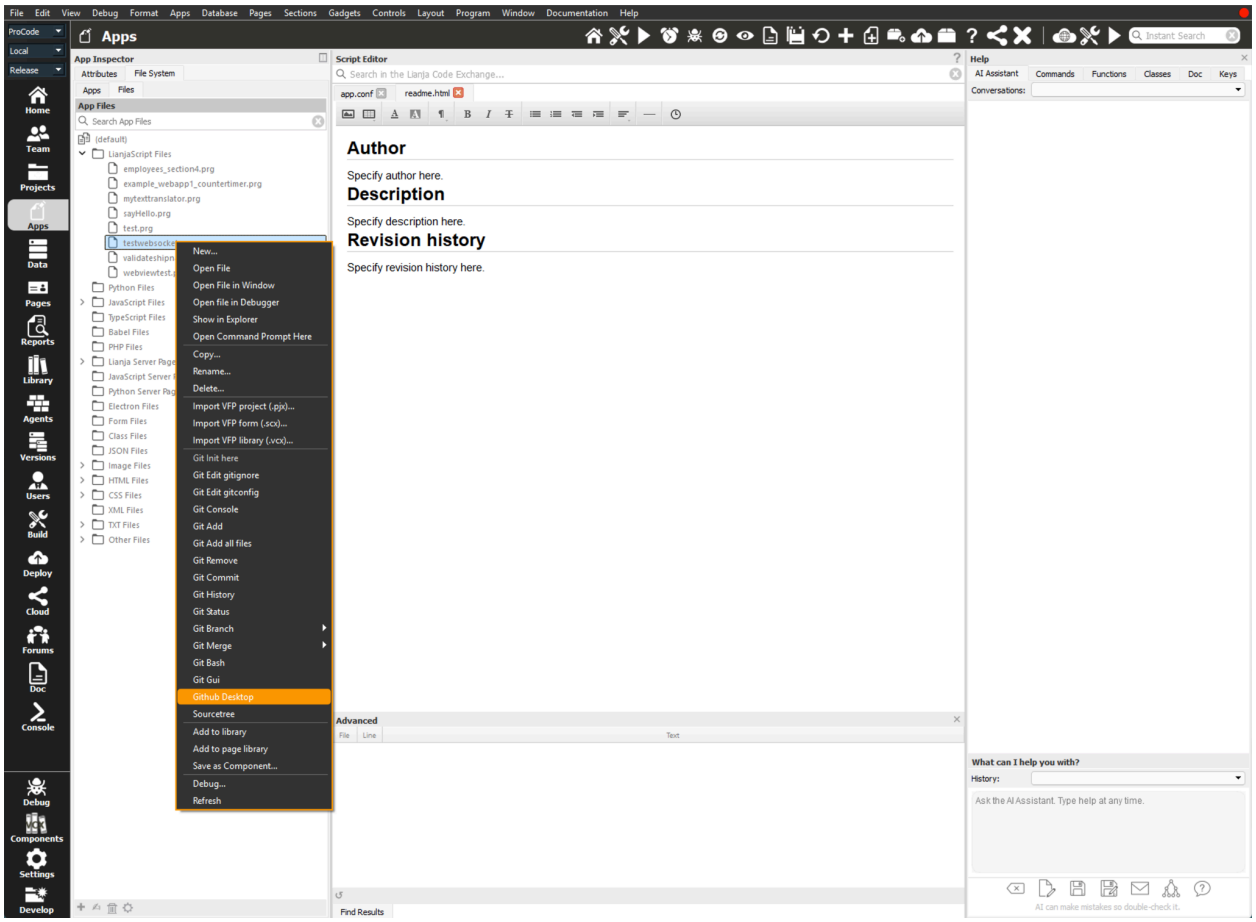
MacOS

Download and install using homebrew:

Shell

brew install git

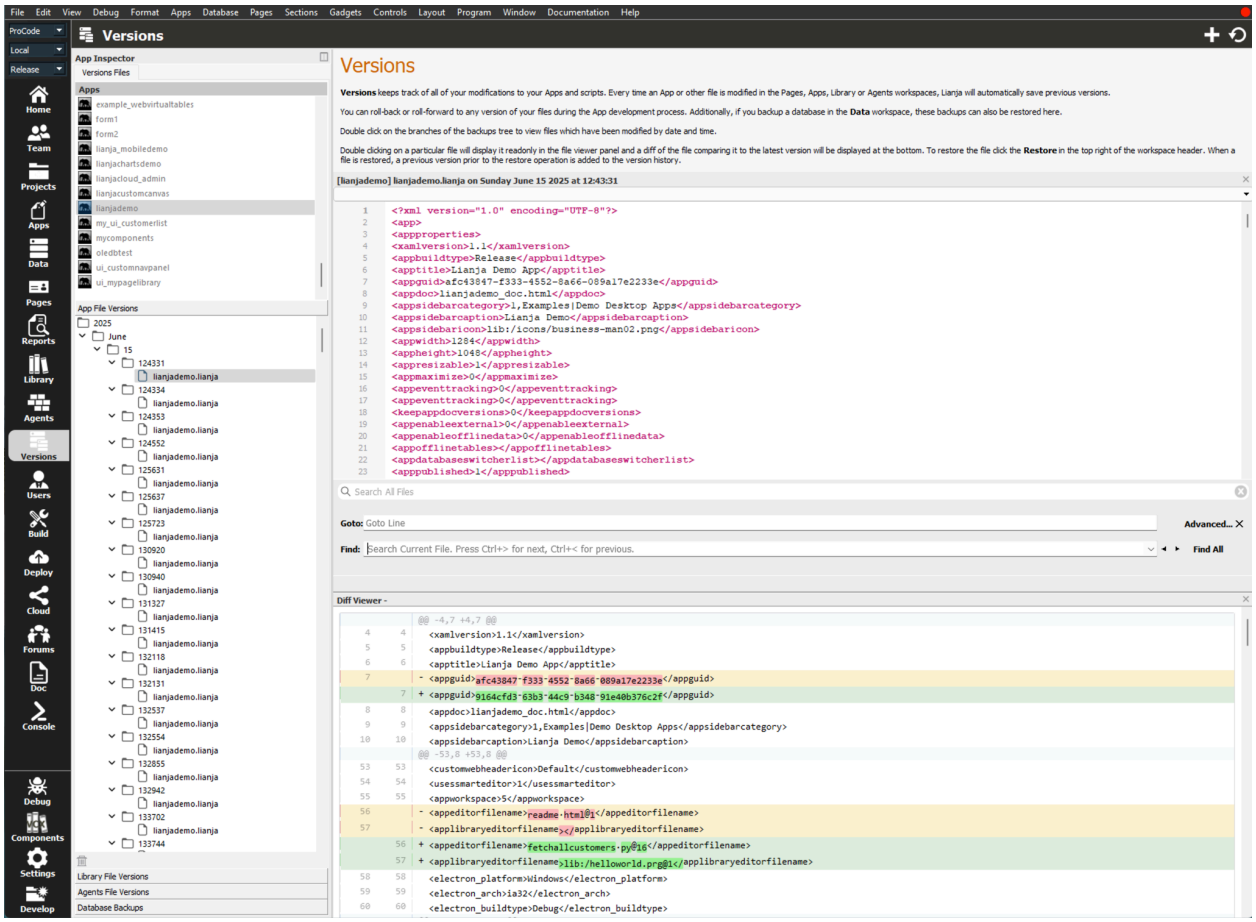
Right click in any of the file trees (Apps, Library or Agents workspaces) and you can work directly with git.



If you have a github account and have installed Github Desktop and/or Sourcetree then these can be run from the context menu in any of the editors; Apps, Library or Agents. You can create repos for your Apps and Library code and share these with other team members.

Built-in Version Management

- Lianja App Builder has built-in support for file versioning.

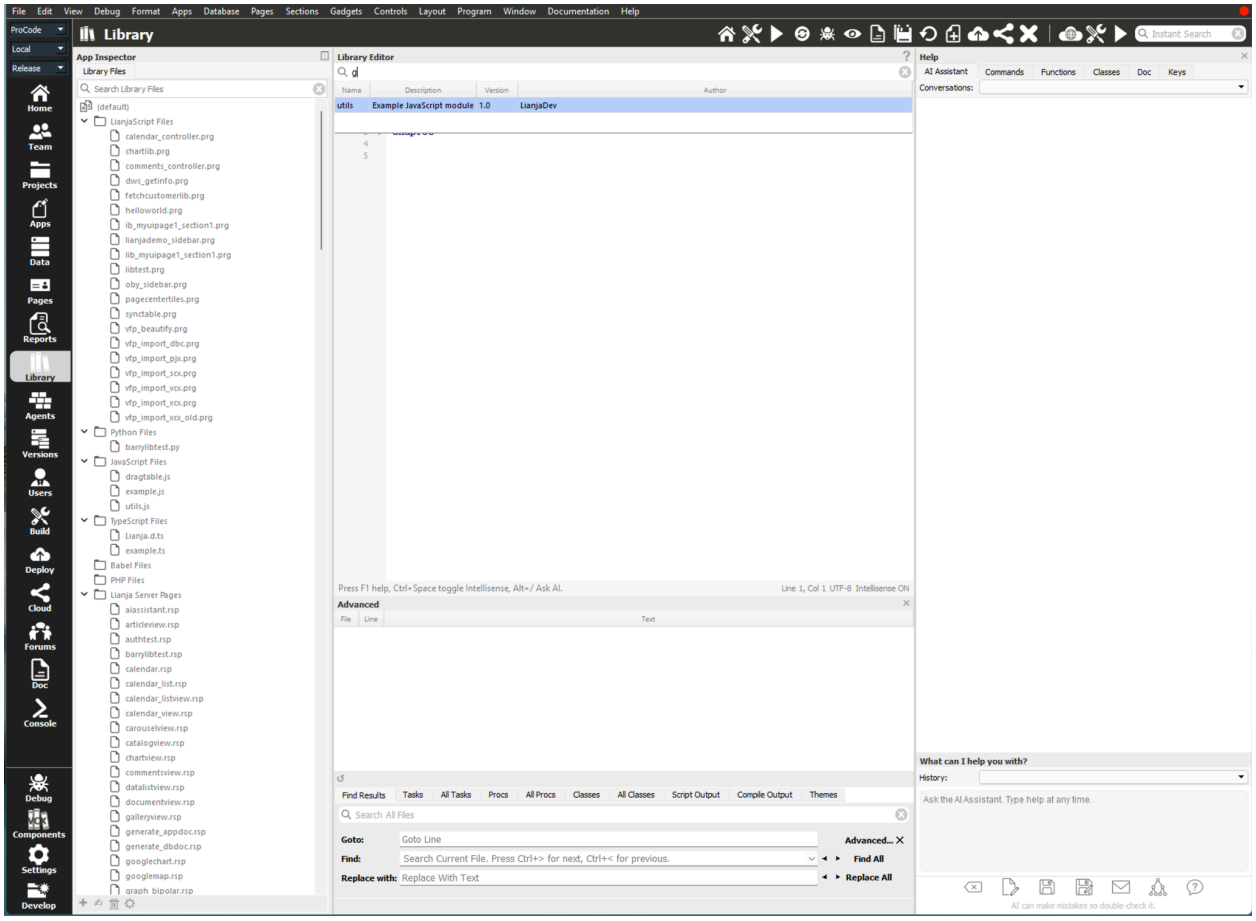


Lianja Code Exchange

The Lianja **Code Exchange** is a collaborative code repository built into Lianja 11.

- Share your **Modules** with teammates.
- Upload, discover and download LianjaScript, Python and JavaScript modules directly from the script editor.

When editing LianjaScript, Python, JavaScript or TypeScript files, you can save them into the Lianja Code Exchange on github. You can also have your own private Code Exchange which is not shareable by others and specific to your own github account and code repo.



LianjaScript modules

In teams or large projects, modules make it easy to work on separate parts of the codebase without stepping on each other's toes.

A **LianjaScript module** contains **Functions**, **Procedures** and **Variables**. They provide Object Oriented encapsulation for LianjaScript code so that loading a library does not "pollute" the namespace and create potential problems due to name clashes.

Use the LianjaScript require() function to dynamically load a library and reference its public variables and procedures/functions in an OO manner: object.property, object.method().

LianjaScript modules should also have a manifest file associated with them which is the same name as the module with a .json extension. This is used when searching for library modules from within the editor search bar.

Example manifest file mylibrary.json:

```JSON
{
	"name":"mylibrary",
	"displayname":"mylibrary.prg",
	"description":"An example LianjaScript module.",
	"publisher":"LianjaDev",
	"version":"1.0",
	"tags":"#module,#lianjascript"
}
```

Example LianjaScript module:

```JavaScript
// mylibrary.prg
public myvar = 10
proc helloworld()
  ? "Hello World"
endproc
// end of mylibrary.prg
```

You can import it and use it like this:

```JavaScript
local mylib = require("mylibrary.prg")
? mylib
? mylib.myvar
mylib.helloworld()
```

Python modules

A **Python module** is simply a file containing Python code, usually with a `.py` extension, that defines **functions**, **classes**, **variables**, or **executable code**.

Think of a module as a **toolbox** — instead of writing everything in one giant script, you organize your logic into separate, reusable units.

As with LianjaScript modules, when editing Python files you can save them into the Lianja Code Exchange on github. You can also have your own private Code Exchange which is not shareable by others.

LianjaScript modules should also have a manifest file associated with them which is the same name as the module with a .json extension. This is used when searching for library modules from within the editor search bar.

Example manifest file mypylibrary.json:

```JSON
{
	"name":"mypylibrary",
	"displayname":"mypylibrary.py",
	"description":"An example Python module.",
	"publisher":"LianjaDev",
	"version":"1.0",
	"tags":"#module,#python"
}
```

Example Python module:

```Python
# utils.py
def greet(name):
    return f"Hello, {name}"
```

You can import and use it like this:

```Python
import utils

print(utils.greet("Alice"))  # → Hello, Alice
```

JavaScript/TypeScript Modules

A **Lianja JavaScript or TypeScript module** is a reusable piece of code containing functions, objects, or classes — defined in its own file that can be **imported into other JavaScript files** using Lianja's `require()` function.

Modules are the **building blocks** of any Lianja Web or Mobile application, and they allow you to break your code into smaller, more manageable, and logically separate units.

As with LianjaScript modules, when editing JavaScript files you can save them into the Lianja Code Exchange on github. You can also have your own private Code Exchange which is not shareable by others.

JavaScript modules should also have a manifest file associated with them which is the same name as the module with a .json extension. This is used when searching for library modules from within the editor search bar.

Example JavaScript Module:

```javascript
var myfunc = function() {
   Lianja.writeOutput("myfunc() was called")
};
var myfunc2 = function() {
   Lianja.writeOutput("myfunc2() was called")
};
module.exports.myfunc = myfunc;
module.exports.myfunc2 = myfunc2;
```

You can import it and use it like this:

```javascript
var utils = require("utils.js");
utils.myfunc()
utils.myfunc2()
```

How do I share modules with others?

You can share modules written in LianjaScript, Python or JavaScript just by clicking the “Share” icon in the page header of the editor.

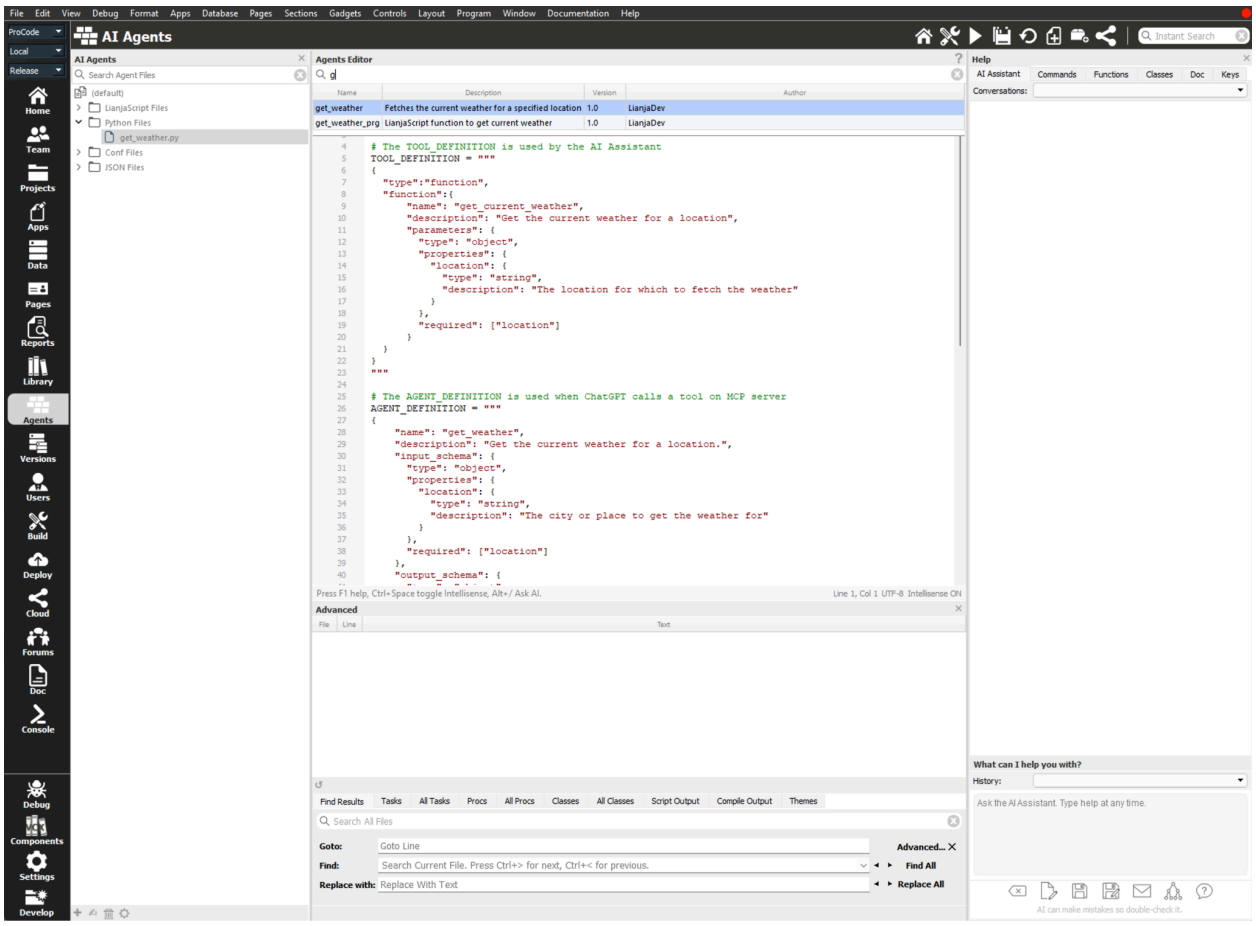
You will be prompted for missing manifest entries and also your Personal Access Token for uploading to github. This Personal Access Token is set up on github for each of your private repos. For the built in repos, namely LianjaCodeX and LianjaAgentX these can only be modified by the Lianja Dev Team.

Lianja AI Agents Exchange

The **Lianja Agents Exchange** is a collaborative AI agents repository built into Lianja 11.

- Share your **Agents** with teammates.

- Upload, discover and download LianjaScript and Python agents directly from the Agents workspace.



Creation and editing of custom AI Agents can be done with minimum to no coding whatsoever just by asking in the AI Assistant.

How do I share agents with others?

Agents can be uploaded into the Agents repo in github. This makes them discoverable by others so that developers can share agents.

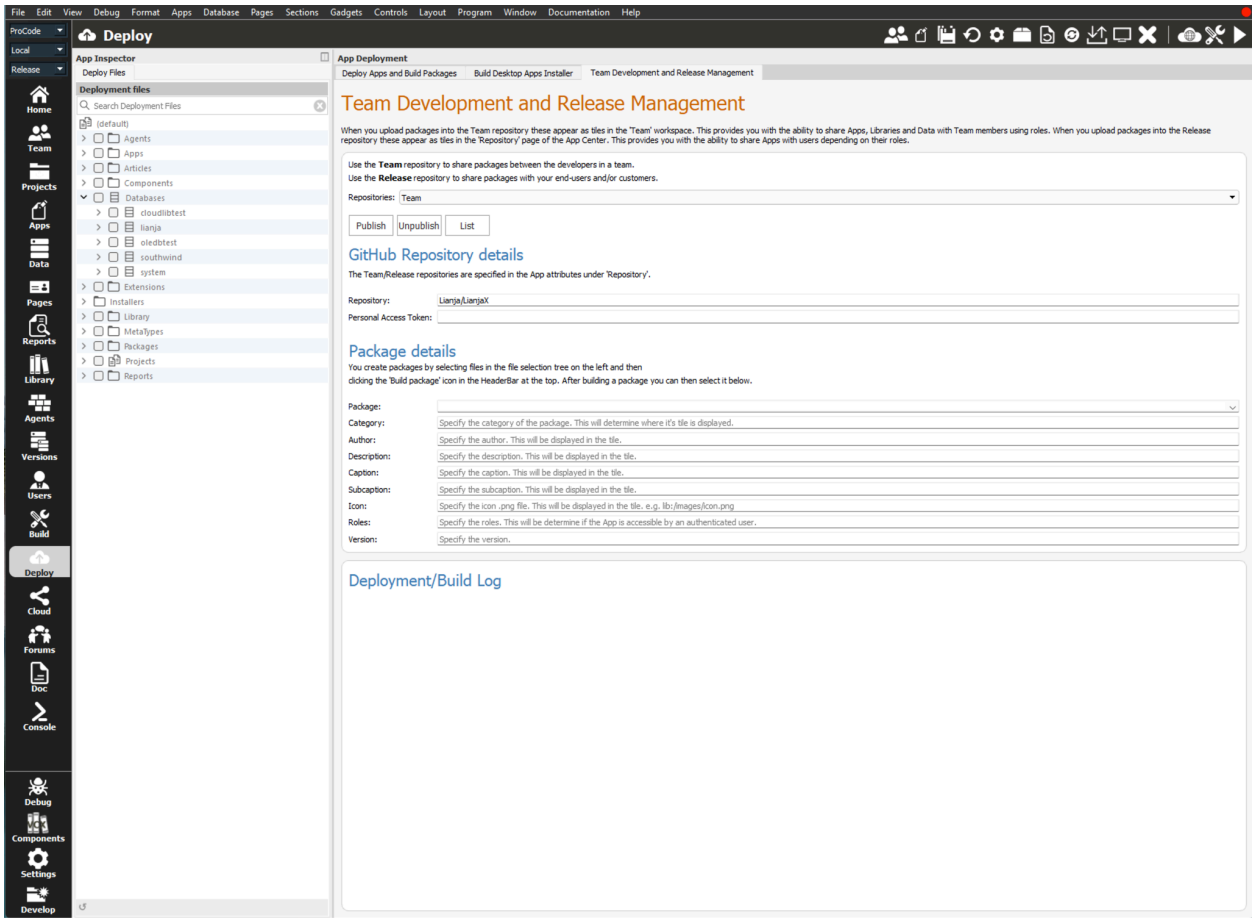
You can search for agents in the Agents repo directly from the editor in the “Agents” workspace. As you type in the agents SearchBar any agents that have #tags or descriptions containing the text you type will be displayed.

Team Workspace

Via the **Lianja App Builder Teams Workspace**, you can:

- Share **Apps** and **libraries** between Team Members.

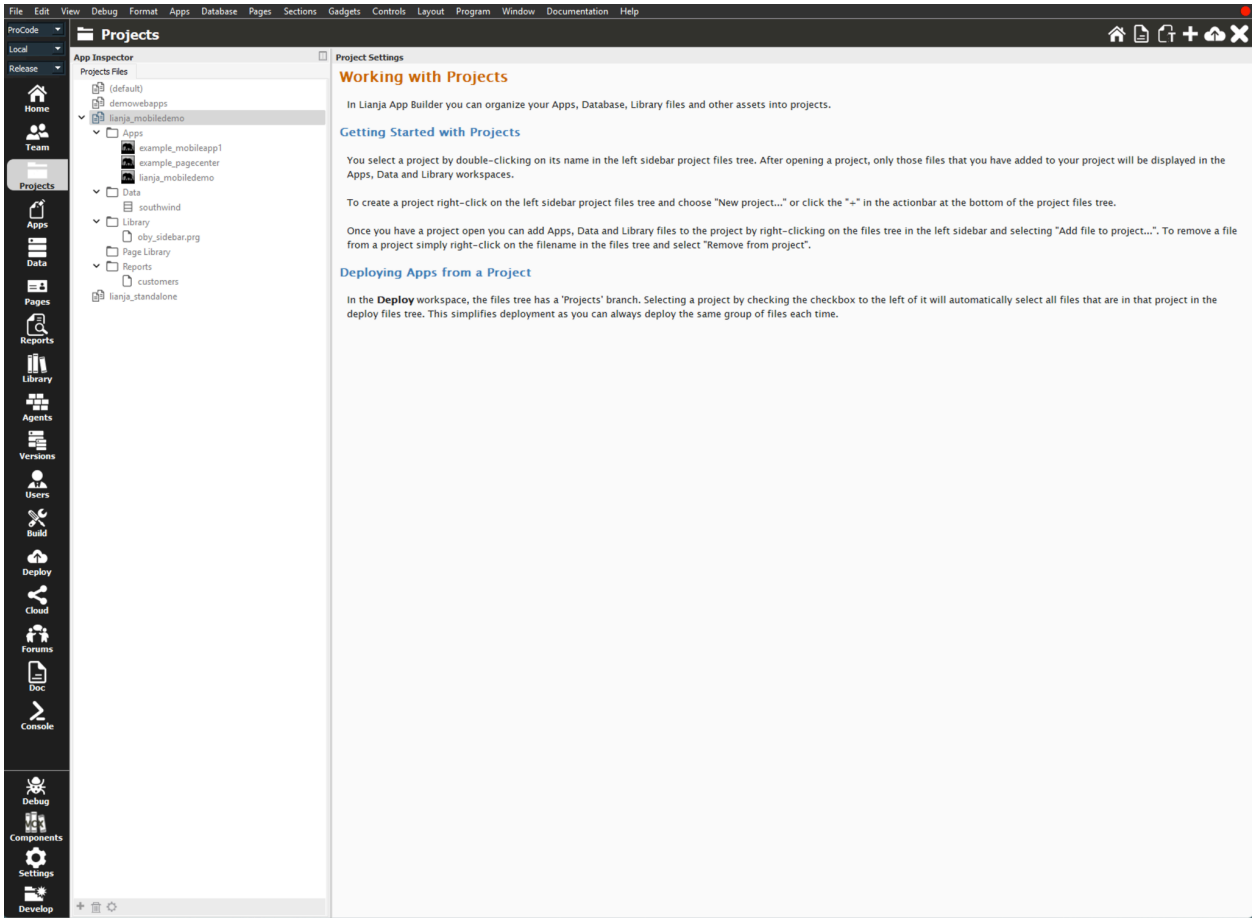
Create and manage Lianja packages in the “Deploy” workspace and upload them into your Team github repo (Development or Release).



Project Management

Lianja App Builder allows organizing large applications into **projects**:

- Team members can focus on specific projects/apps/modules allocated to them by the team leader.



Use projects in conjunction with github and the Lianja Code Exchange to assign **maintainers** for code libraries that you use in your Apps.