Using GitHub for Rally Apps (Windows Version)

SOURCE DOCUMENT (must have a rallydev.com email address to load and edit)

Introduction

Rally has a working relationship with GitHub to enable customer collaboration in creating and evolving apps.

The purpose of this document is to help you understand the relevant workflows and get set up to to use GitHub to collaborate with Rally in creating and evolving apps. Note that this document covers the setup and workflows for the Windows user. If you will be using Mac OSX, please consult the **Using GitHub for Rally apps (Mac Version)** instead.

This document is divided into two sections: **Getting Started**, and **Understanding Developer GitHub Workflows with Rally Apps**:

- In the **Getting Started** section, you'll create a GitHub account, and install the necessary software on your machine.
- In the **Understanding Developer GitHub Workflows with Rally Apps** section, you'll learn to use GitHub and your installed software to support the development workflows: **maintaining an existing app**, and **creating a new app**

So, let's get started!

Getting Started

To access the public repositories, you'll need to have a GitHub account. In addition, there are two software applications that you'll need to install on your Windows machine in order to interact with the GitHub repositories. These applications are named Git, and SmartGit.

Creating a GitHub Account

We recommend that you use a single account for all of your interaction with GitHub. So if you already have a personal GitHub account, use it. Any repository (project) can be stored beneath a personal or organization account at GitHub. Elevated account membership in an Owner Team can also be granted, imparting elevated privileges to create and administer repositories on behalf of a group.

To create a GitHub account, visit http://github.com and choose the **Signup and Pricing** menu item, followed by clicking the **Create a free account** button. Fill out the form, and click the **Create an account** button.

Installing and Configuring Git

Now that you have a GitHub account, you'll need to install and configure an application named Git on your machine. Git is an open source version control system that runs on your local machine, and communicates with the GitHub repositories. To download and install Git, follow the instructions provided on the Set Up Git page on the GitHub site, skipping the **Set Up SSH Keys** section. The sections that you'll need to perform are **Download and Install Git, Set Up Your Info**, and **Celebrate**:
http://help.github.com/win-set-up-git/

Tip: Please make note of the **passphrase** that you created in this section, as you'll need it later.

Installing and Running SmartGit

As you are aware from installing and configuring Git via the help.github.com instructions above, it has a command-line interface. The core Git tool ships with only the sparest of graphical user interfaces. To make the process of interacting with repositories easier, we recommend that you download and install the SmartGit for Windows application. SmartGit was chosen for this workflow, because we believe it provides the easiest possible on-ramp to using Git and GitHub on Windows. However, SmartGit requires the purchase of a license for commercial use. We recommend that you use the free trial at first and if you like it, acquire the appropriate license. To install SmartGit follow the instructions on their website: http://www.syntevo.com/smartgit

After installing SmartGit, go ahead and run it so that you can get the **Setup SmartGit** wizard steps out of the way. We'll point out a couple of steps here where the information is either especially important or not supplied by default. The first such step is shown in Figure 1, where the **Git Executable** supplied in the text field should default to the same one that you installed in the **Installing and Configuring Git** section. If this is not the case, use the **Choose** button to navigate to it.

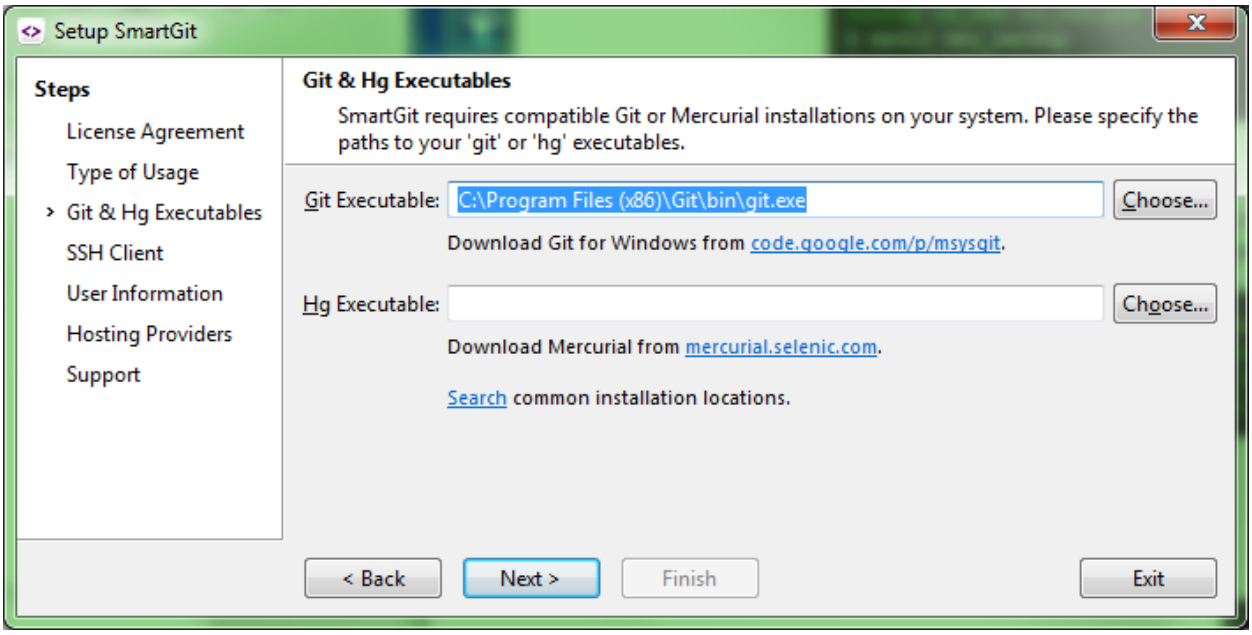

*Figure 1. The **Setup SmartGit** wizard **Git & Hg Executables** step*

In the **SSH Client** step shown in Figure 2, choose the **Use SmartGit as SSH Client** radio button.

*Figure 2. The **Setup SmartGit** wizard **SSH Client** step*

In the **Hosting Providers** step shown in Figure 3, choose **GitHub** and enter your GitHub **User Name** and **Password**. To verify that you supplied the correct information, click the **Login at GitHub** button.

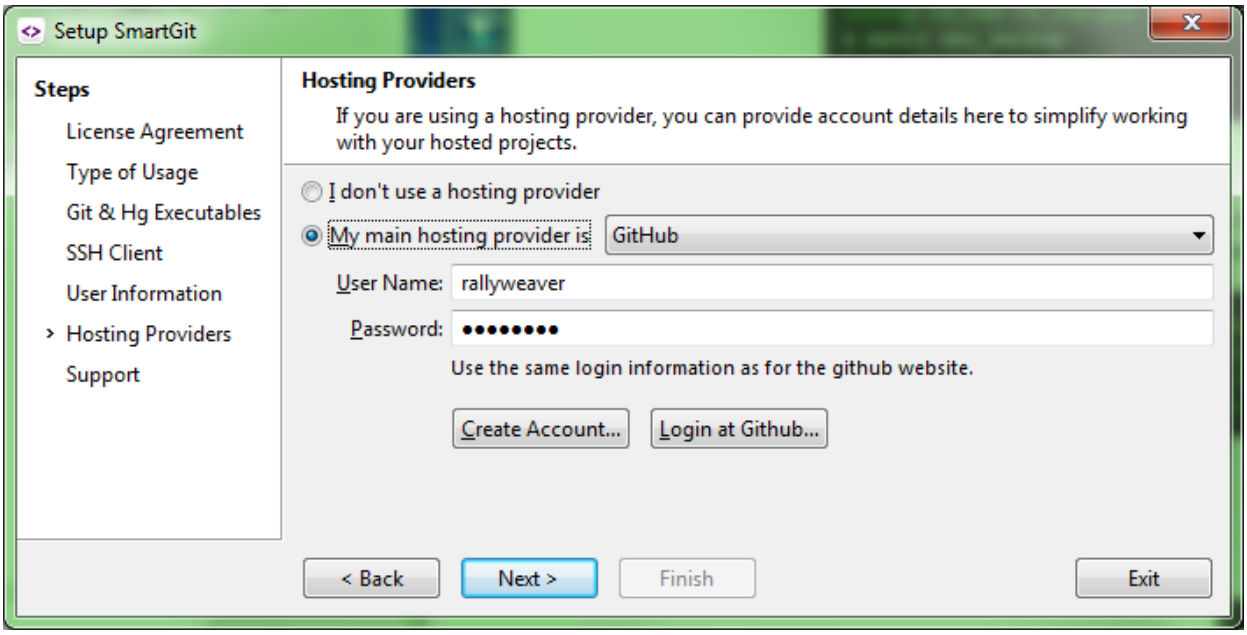

*Figure 3. The **Setup SmartGit** wizard **Hosting Providers** step*

In the **Master Password** dialog shown in Figure 4, go ahead and create a master password.

Tip: Please make note of the **master password** that you created in this section, as you'll need it later.

*Figure 4. The **Master Password** dialog*

When you're finished with the Setup SmartGit wizard, the dialog shown in Figure 5 should appear. Go ahead and click the **Close** button, as we don't need to perform the functionality offered in this dialog just yet.

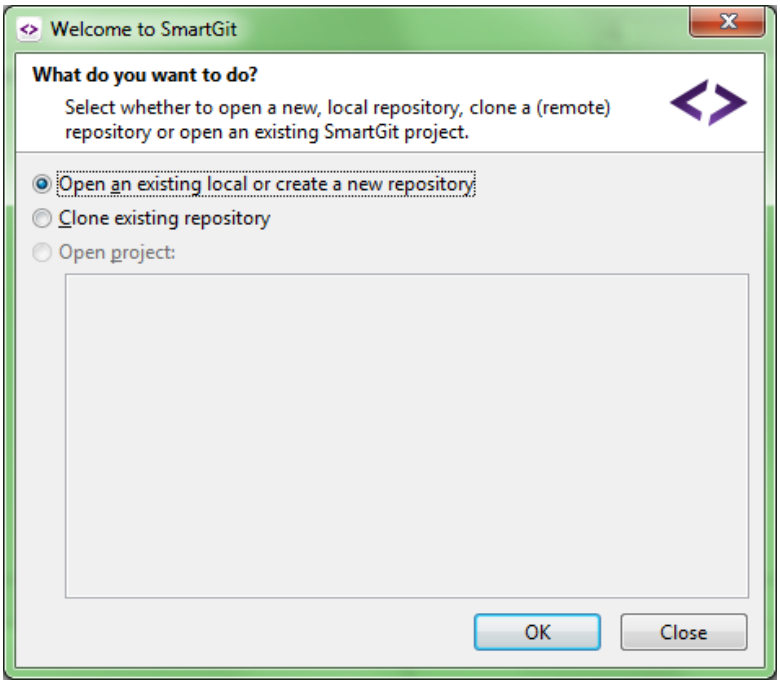

*Figure 5. The **Welcome to SmartGit** dialog*

Note: Though we'll use the SmartGit for Windows to interact with the Rally repositories, command line curious users can find additional information on Git's command line interface at: http://progit.org/book/

Congratulations. You are now prepared to participate in the development workflows. The following section covers these workflows, walking you through the steps required in each.

Understanding Developer GitHub Workflows with Rally Apps

There are two main workflows. They are:

1. creating a new app
2. maintaining an existing app

Let's take a high-level look at the **creating a new app** workflow first:

Creating a New App

This workflow is appropriate when you want to create a new app that uses an existing app or ***app template*** as a starting point. The idea here is that you'll fork a copy of an existing app or app template that becomes the primary repository for your new app. Modifications made to the application may be committed and synchronized directly to this primary repository. Here are the steps that you'll follow:

1. Fork the repository that contains the app or app template

2. Rename the project to a more meaningful name
3. Clone the new repository to your local machine
4. Modify the application
5. Commit and sync to the fork
6. (optional) Submit a request for the new app to be added to the Rally Apps library

Now we'll take a high-level look at the **maintaining an existing app** workflow:

Maintaining an Existing App

This workflow is appropriate when you want to make an update to an existing app to which you don't have write access. The idea here is that you'll make your modifications in your copy of the main repository for the app, and then submit a request to the maintainer of the main repository to apply your submitted modifications. Here are the steps that you'll follow:

1. Fork the repository that contains the app
2. (renaming the project is not applicable in this workflow)
3. Clone the new repository to your local machine
4. Modify the application
5. Commit and sync to your fork
6. Submit a pull request for inclusion into the original repository

Notice that the steps in these workflows are very similar, so we'll treat them as one set of steps, pointing out the differences as appropriate. To make these steps in both of the workflows easier to visualize, Figure 6 depicts them graphically:

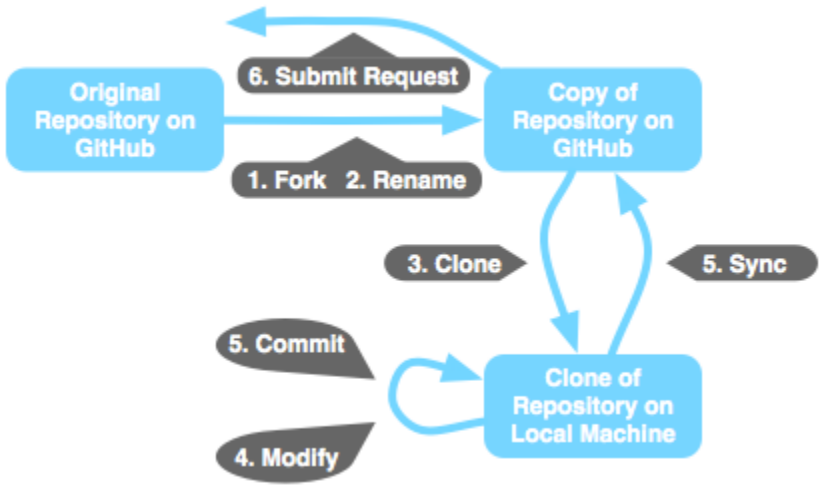

Figure 6. The GitHub for Rally Apps workflows

Diving into the Details of Both Workflows

We'll elaborate on each of the steps in these workflows, beginning with forking the original repository:

1. Forking the Original Repository

When you ***fork*** a GitHub repository, a copy of that repository is created on the GitHub server. Before performing the fork, you must first navigate to the repository that contains the app that you want to modify. You can do this in a number of ways, such pasting a GitHub URL supplied by a Rally GitHub administrator into your browser. Note that all of the Rally approved apps will be located in the RallyApps account on GitHub, which is located at the following URL: https://github.com/RallyApps

When you are on the desired repository page, click the **Fork** button as shown in Figure 7 below to perform the fork operation.

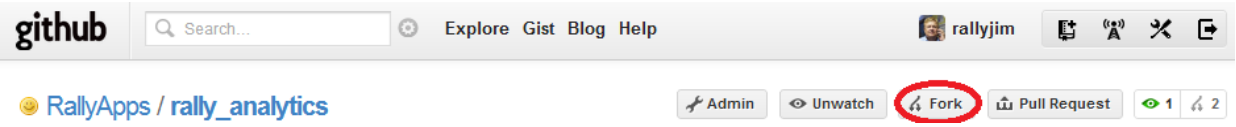

*Figure 7. The **Fork** button on the repository page*

If the dialog shown in Figure 8 appears, select your personal account as the one to which you want to fork the repository.

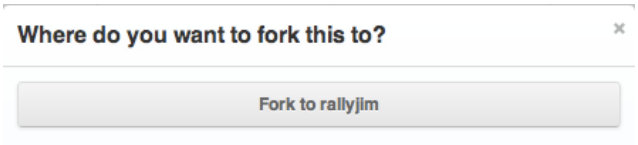

*Figure 8. The **Fork** dialog on the repository page*

Now that you've forked the repository, a copy of that repository exists on the GitHub server but is associated with your account. Note that you may navigate to this new repository in subsequent sessions by choosing it from the **Your Repositories** box on your GitHub home page as shown in Figure 9:

*Figure 9. Selecting a repository from the **Your Repositories** box*

For more information about forking a repository, consult the corresponding GitHub Help page: http://help.github.com/fork-a-repo/

The next step will be to clone the new repository on your local machine so that you can modify

the application.

2. Renaming the Project (if creating a new app)

If **creating a new app**, you need to rename the new repository you just created (via the fork) in the previous step. To do this, click the **Admin** button on the GitHub repository page shown in Figure 10.

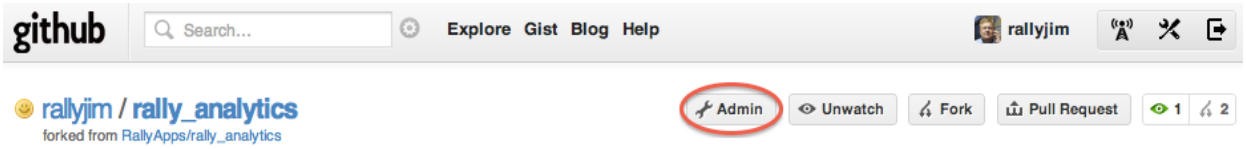

*Figure 10. The **Admin** button on the GitHub repository page*

Then, from the GitHub repository admin page shown in Figure 11, enter a new **Repository Name** and click the **Rename** button.

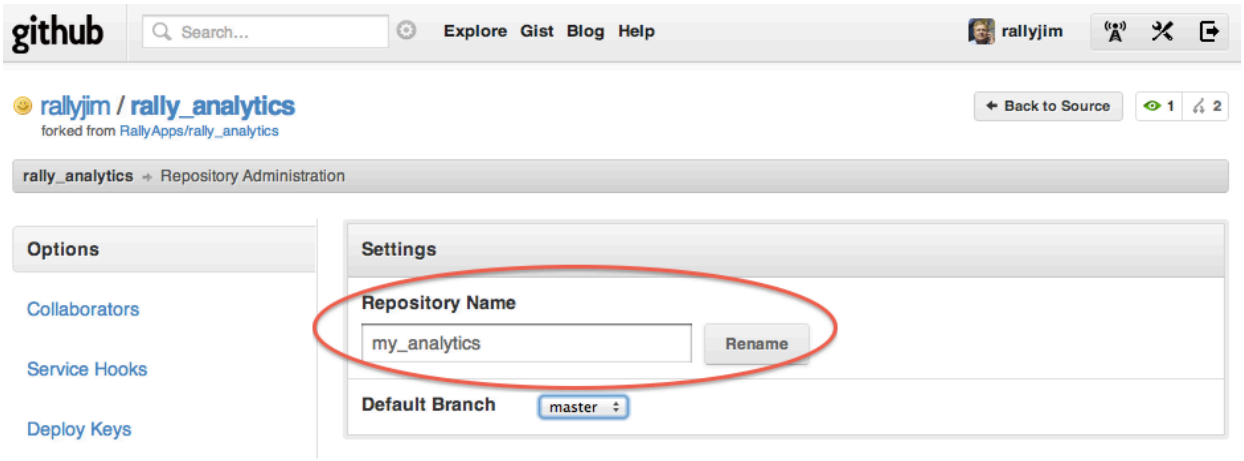

Figure 11. The GitHub repository admin page

3. Cloning the New Repository

When you ***clone*** a repository, you are making a copy of the repository that will reside on your local machine. Note that there is a **Read+Write access** button on the repository's page as shown in Figure 12.

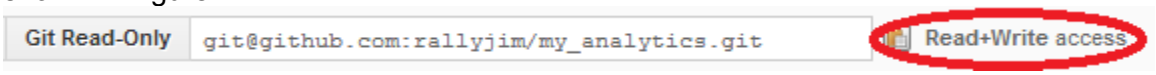

*Figure 12. The **Read+Write access** button on the repository page*

After clicking the **Read+Write access** button, the URL for obtaining read/write access to the repository will be copied to the clipboard. You'll then startup SmartGit, select **Project | Clone** from the menu, and paste the URL from the clipboard into the **Repository URL** field as shown in Figure 13.

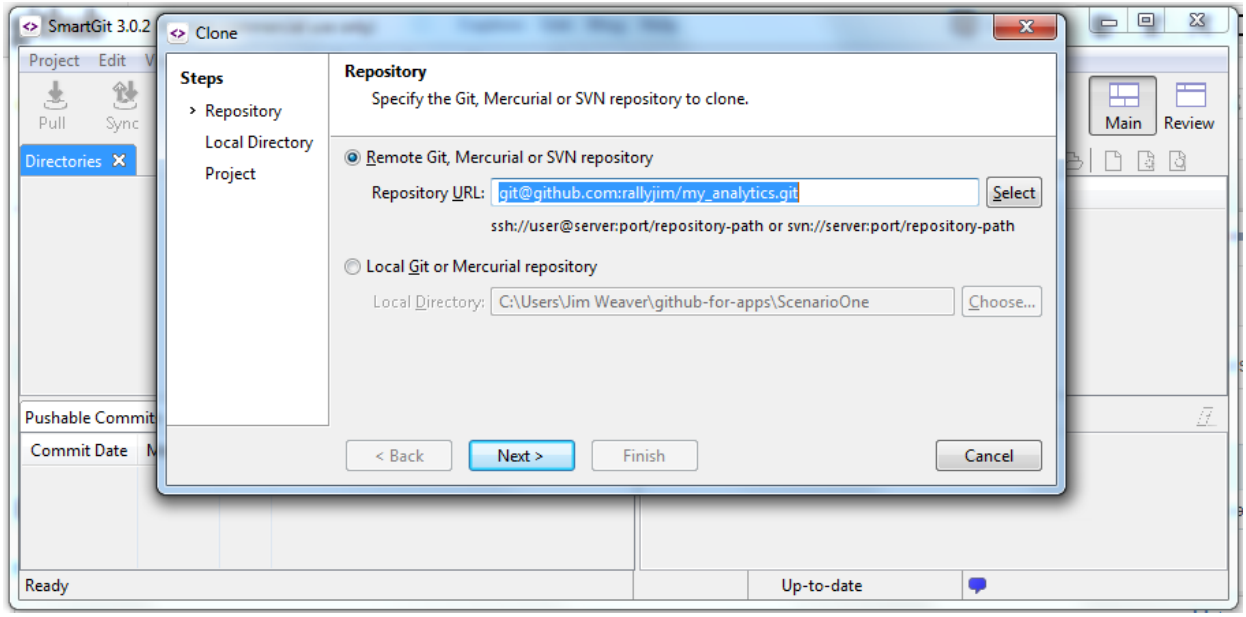

*Figure 13. The **Repository** step of the **Clone** wizard*

Clicking the **Next** button may produce the dialog shown in Figure 14, prompting you for the **master password**.

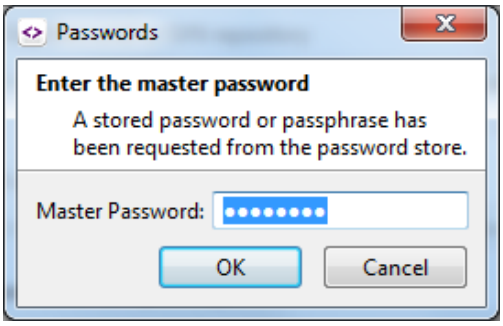

Figure 14. Entering the Master Password

Enter the **master password** that you created in the **Installing and Running SmartGit** section and click the **OK** button.

At this point, the **SSH Authentication** dialog shown in Figure 15 may appear. Enter the **Private Key File** and **passphrase** that you created in the **Installing and Configuring Git** section, click the **Store passphrase** checkbox, and click the **Login** button.

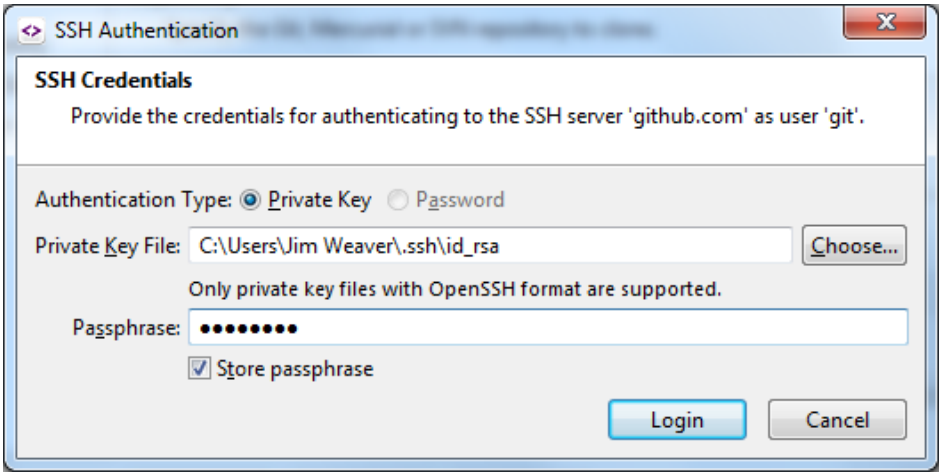

Figure 15. Entering SSH Credentials

In the **Local Directory** step of the **Clone** wizard shown in Figure 16, specify the location in which you'd like the clone to be created.

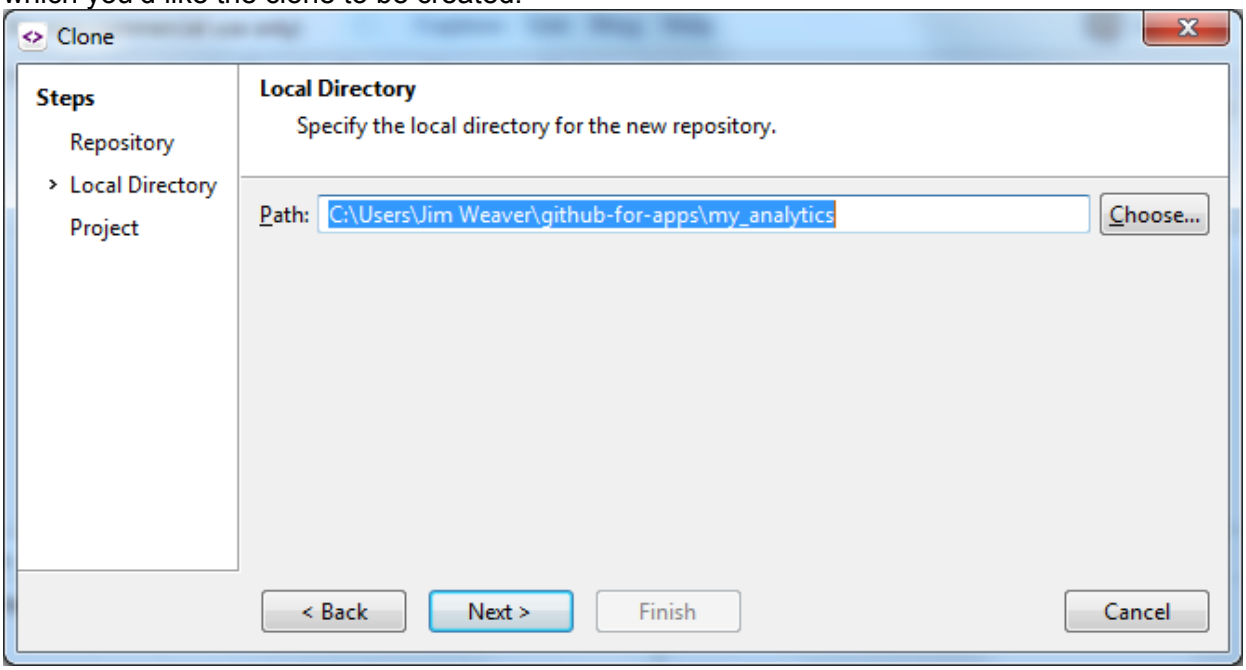

*Figure 16. The **Local Directory** step of the **Clone** wizard*

In the **Project** step of the **Clone** wizard shown in Figure 17, specify a new SmartGit **Project Name** for the newly cloned repository.

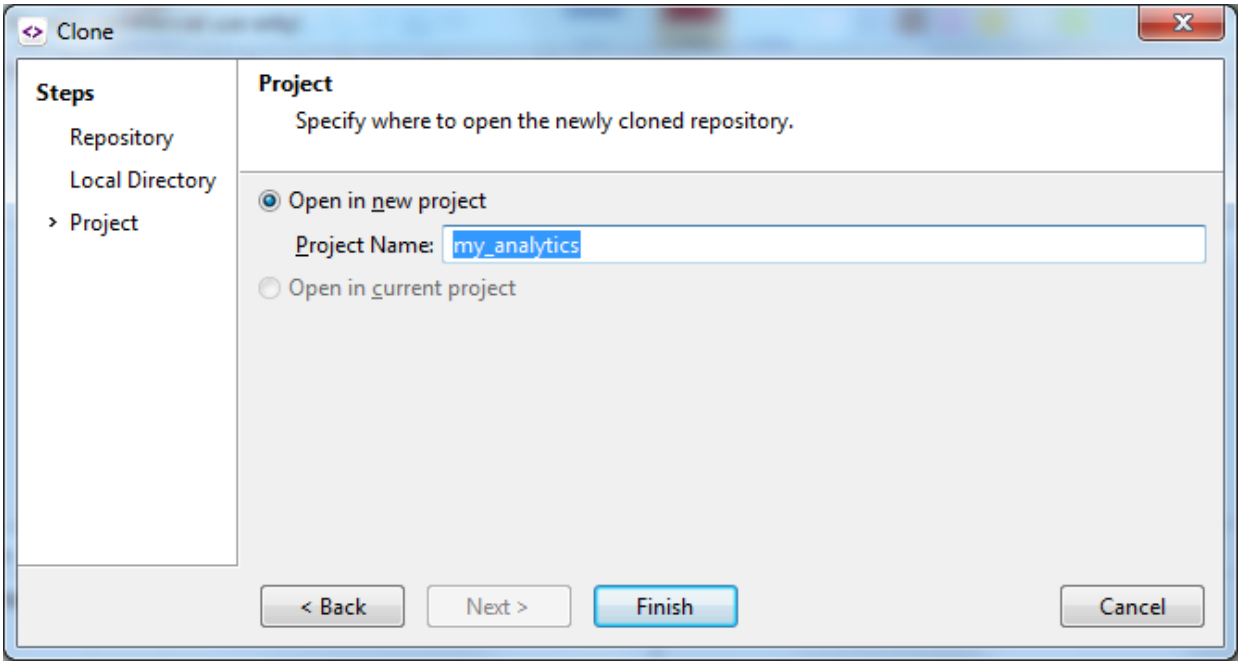

*Figure 17. The **Project** step of the **Clone** wizard*

After clicking the **Finish** button, a full copy of the repository will be created on your local machine at the location you specified.

4. Modify New Repository

Now that you have cloned the repository you can make modifications to the app, and continue on to the **commit and sync to the fork** step.

Note: As with any version control system, branching is supported by Git. For information on Git's very powerful branching capabilities, and ideas for implementing them in your project workflow, look here: http://progit.org/book/ch3-0.html

5. Committing and Syncing to the Fork

The modifications that you made to the app exist on your local machine, but in order for the changes to exist on the GitHub server you need to do two things: ***Commit*** your changes to the local repository, and ***sync*** the local repository to the remote repository on the GitHub server.

We'll begin performing these operations by launching SmartGit. Figure 18 shows the SmartGit main view containing an open project that represents a local repository, and any files that have been modified.

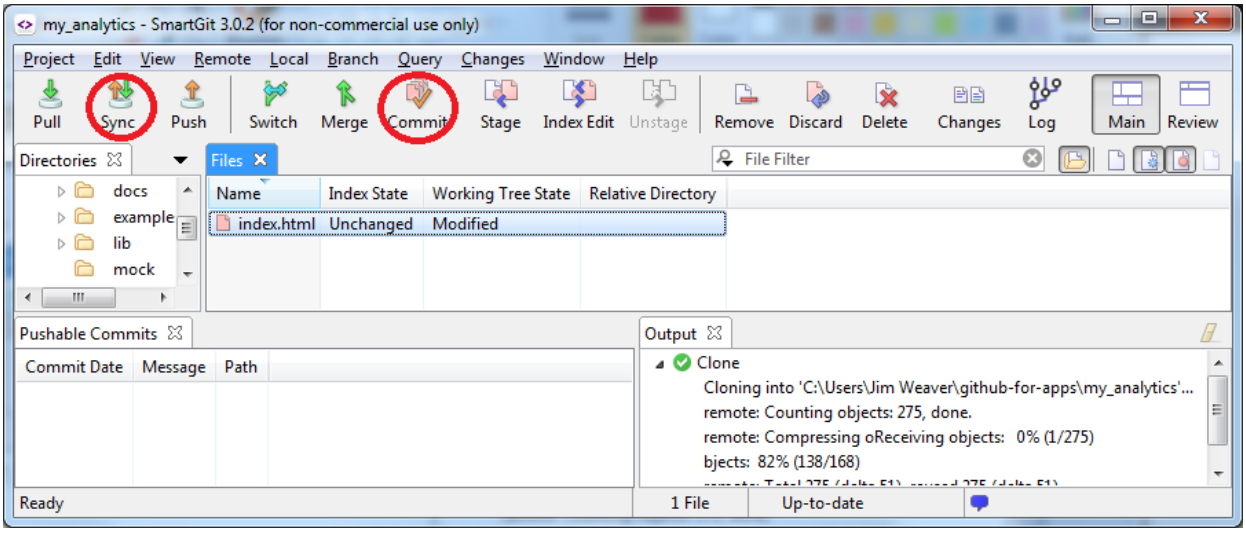

Figure 18. SmartGit main view

Selecting at least one of the modified files and clicking the **Commit** button causes the **Commit** dialog shown in Figure 19 to appear.

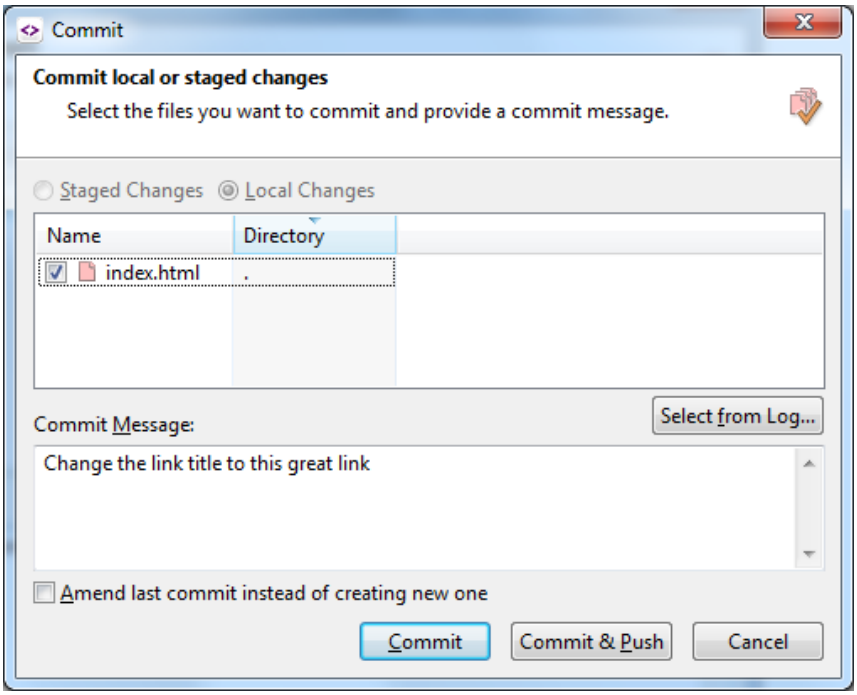

Figure 19. SmartGit Commit dialog

After typing in the **Commit Message** “Change the link title to this great link”, we’ll click the **Commit** button which causes the changes to be committed to the local repository.

Next we’ll click the **Sync** button previously shown in Figure 18 to cause the remote repository to be synchronized with the local repository.

6. Submitting a Request

If **creating a new app**, to submit your app for possible inclusion into the Rally Apps catalog, send an email to app-submission@rallydev.com. Be sure to include your contact information and a link to the repository for your App.

If **maintaining an existing app**, you'll need to send a pull request for your modifications to be applied to the original repository. To do this, we'll click the **Pull Request** button on the GitHub repository page shown in Figure 20.

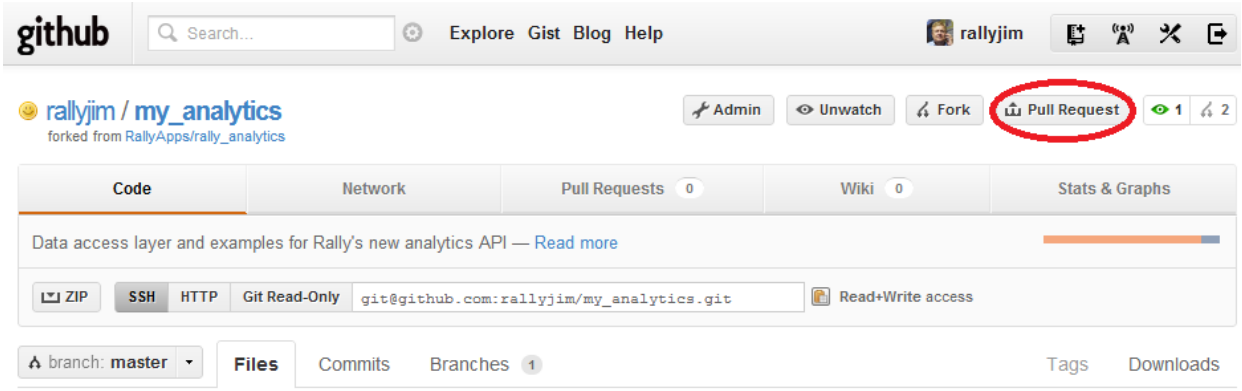

*Figure 20. The **Pull Request** button on the GitHub repository page*

As a result of clicking the **Pull Request** button, the page in Figure 21 appears in which you can view and modify information about the pull request. For example, we entered the message "This modification changes the link title to this great link". When ready to send the request, click the **Send pull request** button.

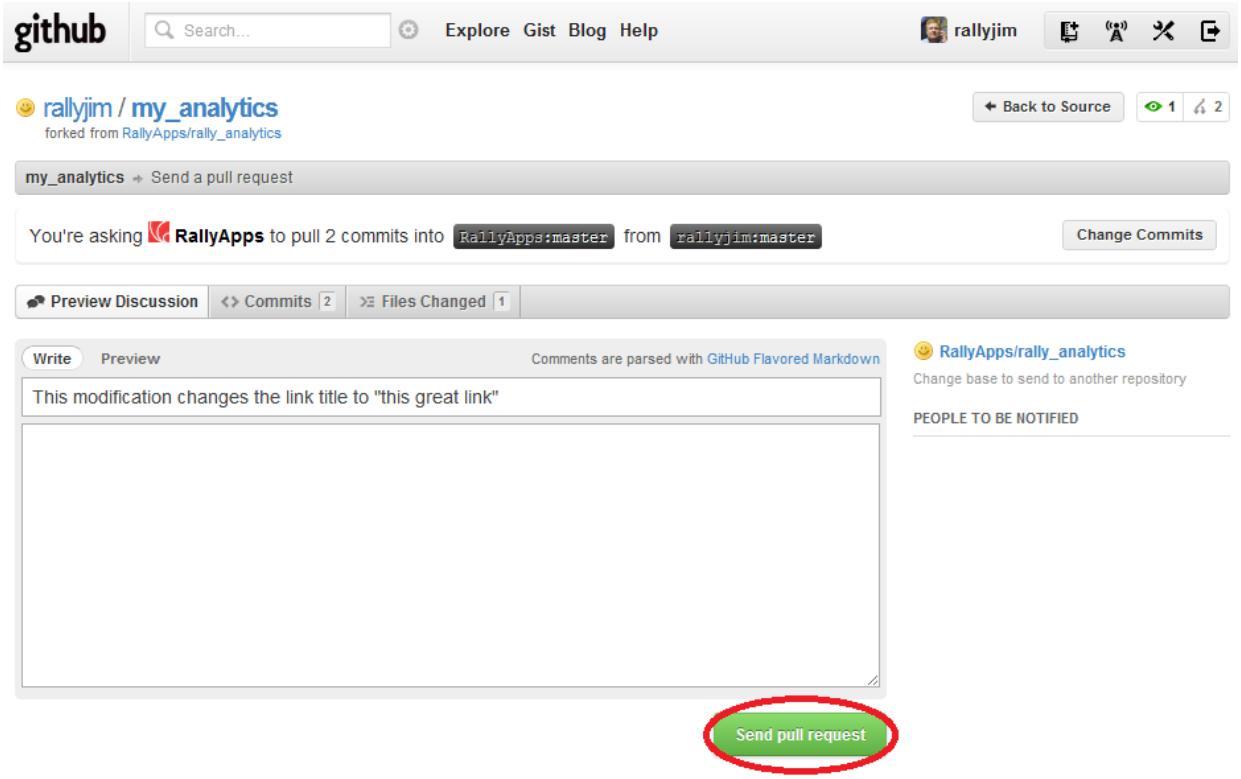

Figure 21. The GitHub pull request information page

As a result of sending the pull request, the repository maintainer will be prompted to review and merge the modifications into the original repository. For more information about pull requests, consult the corresponding GitHub Help page: http://help.github.com/send-pull-requests/

Summary

In this document, you learned how to participate in the GitHub Rally for Apps from a developer perspective. Specifically, you learned to:

- Create a **GitHub account**
- Install and configure **Git**
- Install **SmartGit for Windows**
- Use GitHub and the installed software to implement the **two development workflows**, which are: **creating a new app**, and **maintaining an existing app**.

We hope that you've found this document helpful, and encourage you to contact us at github-for-apps@rallydev.com if you have any questions or feedback.