Using GitHub for Rally Apps (Windows Version)

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

Introduction

Rally is conducting a pilot that leverages public repositories in GitHub to enable customer collaboration in creating and evolving apps.

The purpose of this document is to help you get set up to participate in this pilot, as well as to understand the relevant workflows. 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 in the pilot: **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. Repositories will be distinctly associated with either your personal or employer(s) work. Also, your single GitHub account can be given "owner team" status on a work account.

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

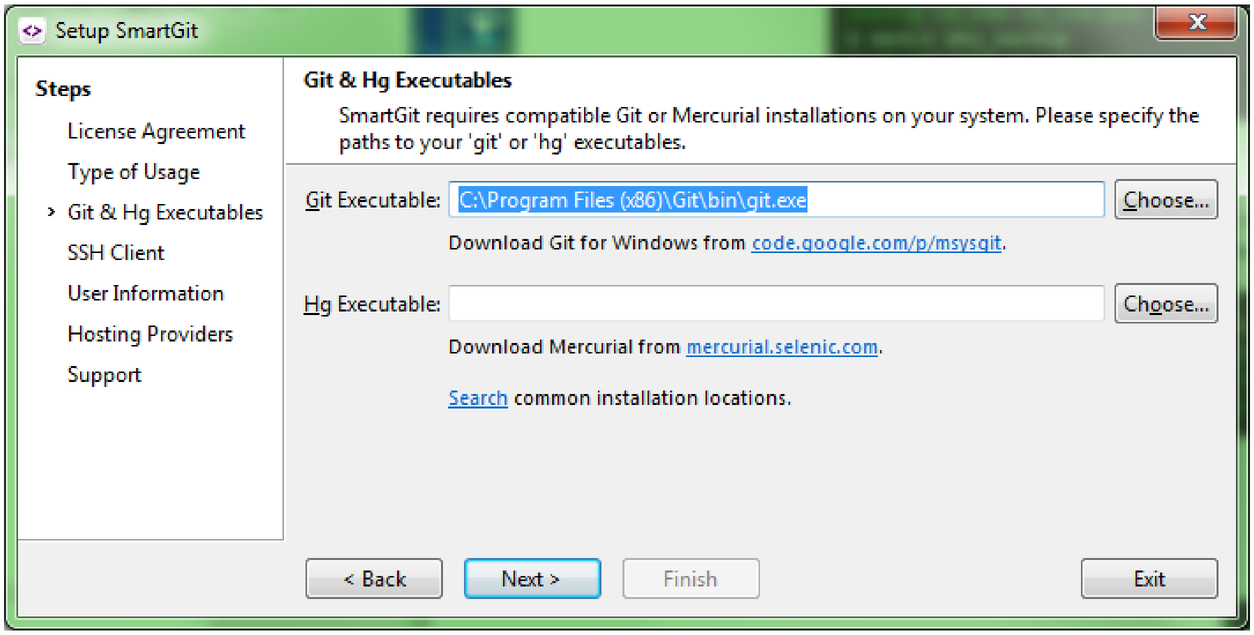
Now that you have a GitHub account, you'll need to install and configure an application named Git on your machine. Git is an open source version control system that runs on your local machine, and communicates with the GitHub repositories. To download and install Git, follow the instructions provided on the Set Up Git page on the GitHub site: 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, it has a command-line interface. To make the process of interacting with the repositories easier, you may want to download and install a graphical user interface. We've included here instructions for using an application named SmartGit. 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 0a, 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. [TODO: Renumber all of the Figures in this document after it has been reviewed and tested]


Setup SmartGit

Steps
- License Agreement
- Type of Usage
- › Git & Hg Executables
- SSH Client
- User Information
- Hosting Providers
- Support

Git & Hg Executables

SmartGit requires compatible Git or Mercurial installations on your system. Please specify the paths to your 'git' or 'hg' executables.

Git Executable: C:\Program Files (x86)\Git\bin\git.exe [Choose...]

Download Git for Windows from code.google.com/p/msysgit.

Hg Executable: [Choose...]

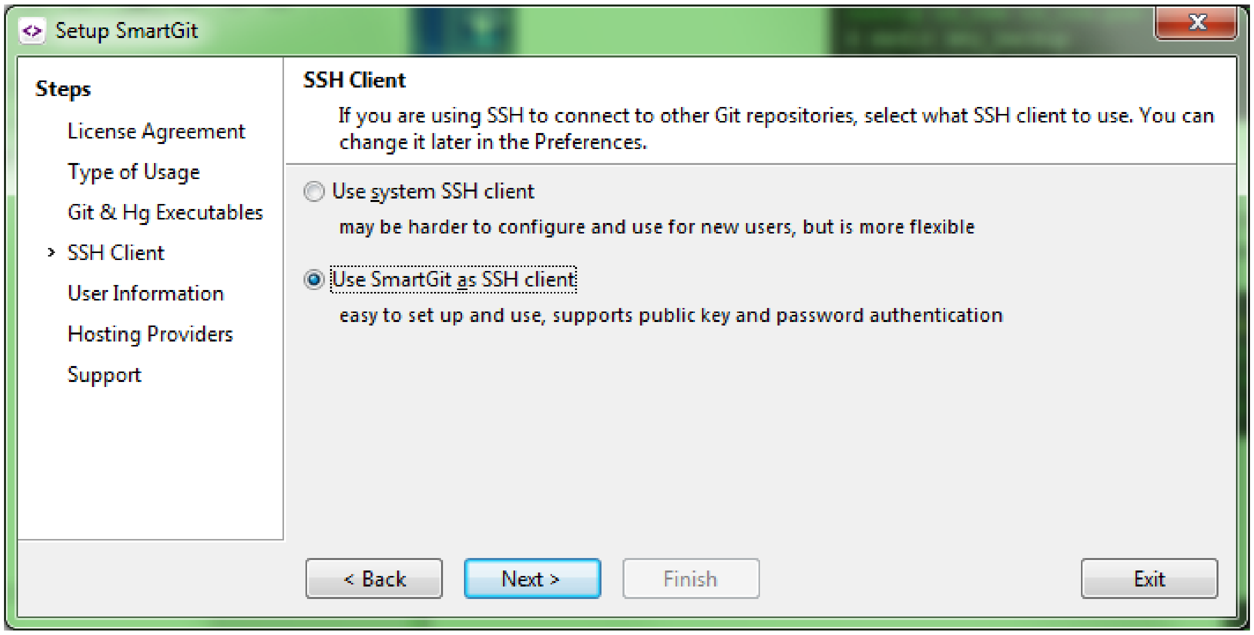
Download Mercurial from mercurial.selenic.com.

Search common installation locations.

< Back | Next > | Finish | Exit

Figure 0a. The ***Setup SmartGit*** *wizard* ***Git & Hg Executables*** *step*

In the **SSH Client** step shown in Figure 0b, choose the **Use SmartGit as SSH Client** radio button.


Setup SmartGit

Steps
- License Agreement
- Type of Usage
- Git & Hg Executables
- › SSH Client
- User Information
- Hosting Providers
- Support

SSH Client

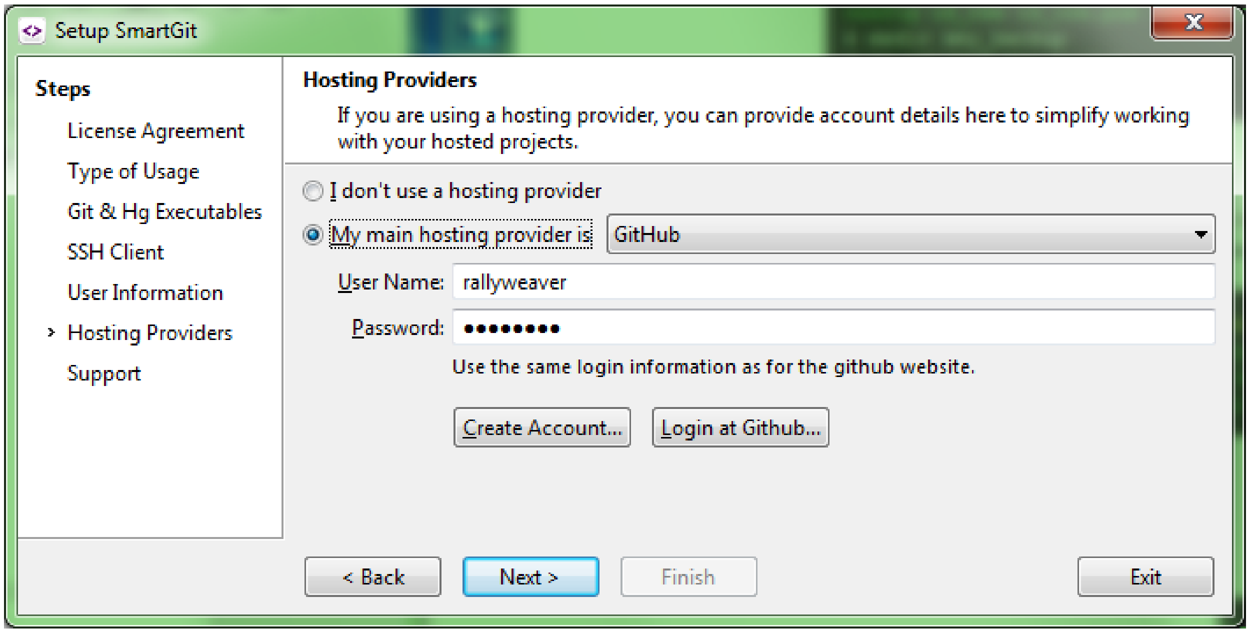
If you are using SSH to connect to other Git repositories, select what SSH client to use. You can change it later in the Preferences.

- Use system SSH client
  may be harder to configure and use for new users, but is more flexible
- [✓ Use SmartGit as SSH client]
  easy to set up and use, supports public key and password authentication

< Back | Next > | Finish | Exit

Figure 0b. The ***Setup SmartGit*** *wizard* ***SSH Client*** *step*

In the **Hosting Providers** step shown in Figure 0c, choose **GitHub** and enter your GitHub **User Name** and **Password**. To verify that you supplied the correct information, click the **Login at GitHub** button. [TODO: Change the screen shot after a Rally App has been added to the repository, perhaps supplying a different User Name]



Hosting Providers

If you are using a hosting provider, you can provide account details here to simplify working with your hosted projects.

I don't use a hosting provider

[✓ My main hosting provider is] GitHub

User Name: rallyweaver

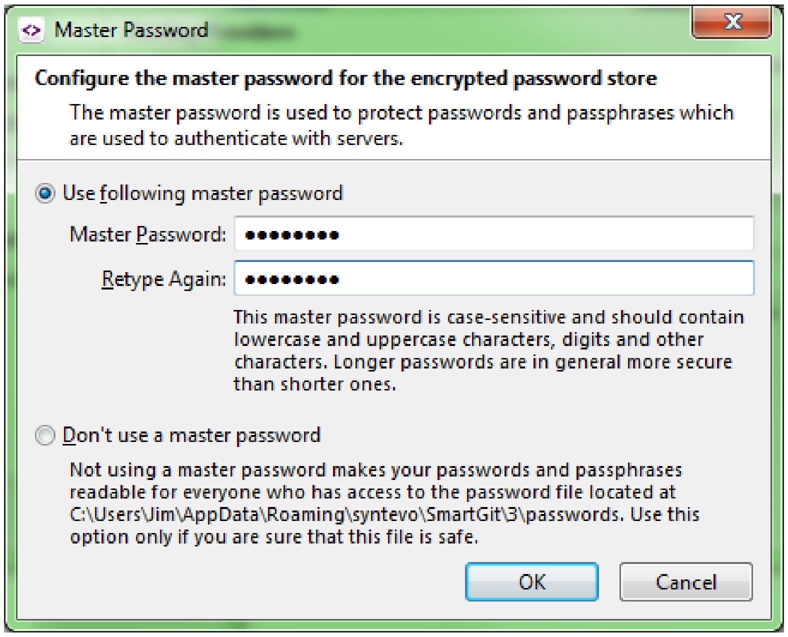
Password: ••••••••

Use the same login information as for the github website.

*Figure 0c. The **Setup SmartGit** wizard **Hosting Providers** step*

In the **Master Password** dialog shown in Figure 0d, 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.



Configure the master password for the encrypted password store

The master password is used to protect passwords and passphrases which are used to authenticate with servers.

[✓ Use following master password]

Master Password: •••••••••

Retype Again: •••••••••

This master password is case-sensitive and should contain lowercase and uppercase characters, digits and other characters. Longer passwords are in general more secure than shorter ones.

Don't use a master password

Not using a master password makes your passwords and passphrases readable for everyone who has access to the password file located at C:\Users\Jim\AppData\Roaming\syntevo\SmartGit\3\passwords. Use this option only if you are sure that this file is safe.

*Figure 0d. The **Master Password** dialog*

When you're finished with the Setup SmartGit wizard, the dialog shown in Figure 0e should appear. Go ahead and click the **Close** button, as we don't need to perform the functionality offered in this dialog just yet.

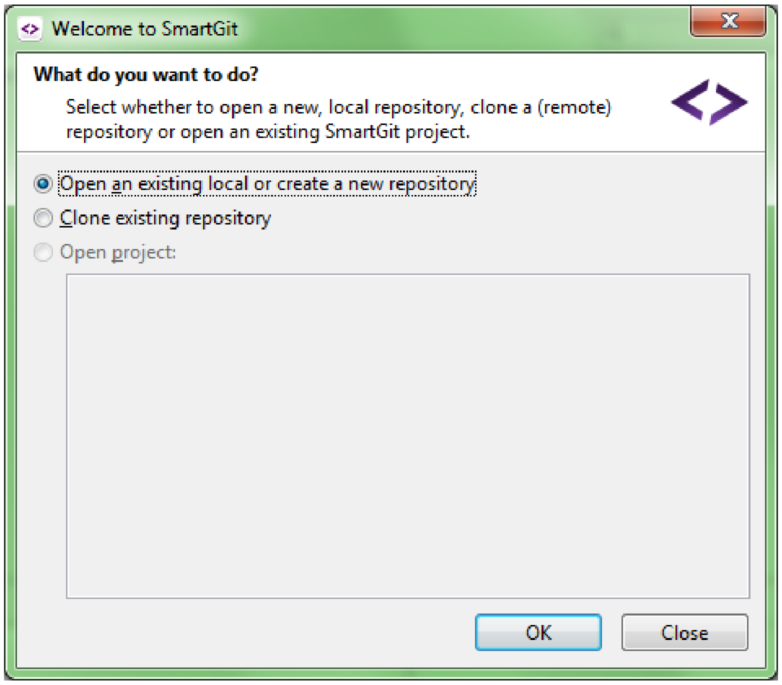

Welcome to SmartGit
What do you want to do?
Select whether to open a new, local repository, clone a (remote) repository or open an existing SmartGit project.
Open an existing local or create a new repository
Clone existing repository
Open project:
OK
Close

*Figure 0e. The **Welcome to SmartGit** dialog*

Note: You won't need it to work with Rally repositories but if you are curious, additional information on using Git and its command-line interface is located can be found at: http://progit.org/book/

Congratulations, you are now ready to participate in the development workflows of this pilot. 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 in the GitHub for Rally Apps pilot. These are:

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

Let's take a look at the **maintaining an existing app** workflow first:

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 copies of the main repository for the app, and then submit a request to the maintainer of the main repository for your modifications to be applied. Here are the steps that you'll follow:

1. Fork the repository that contains the app
2. Clone the new repository to your local machine
3. Modify the application
4. Commit and sync to your fork
5. Send a pull request for inclusion into the original repository

To make these steps easier to visualize, Figure 1 depicts them graphically:

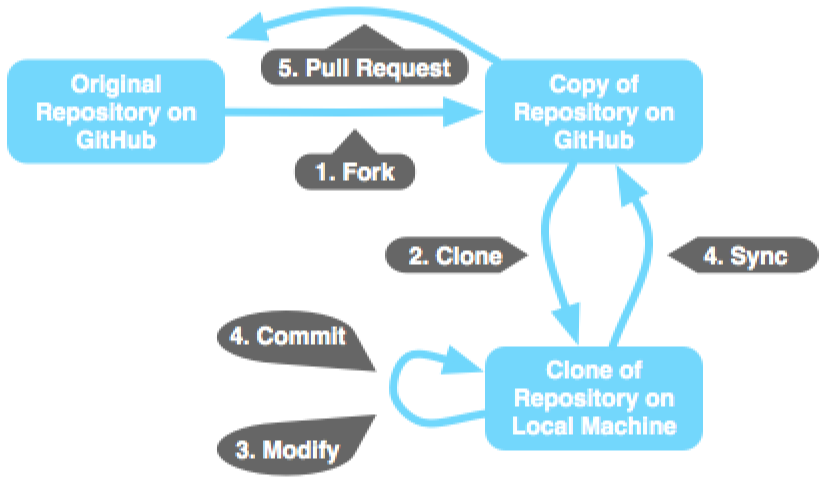

5. Pull Request
Original Repository on GitHub
Copy of Repository on GitHub
1. Fork
2. Clone
4. Sync
4. Commit
Clone of Repository on Local Machine
3. Modify

*Figure 1. The **maintain an existing app** workflow*

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

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 RallySoftware account on GitHub, which is located at the following URL: https://github.com/organizations/RallySoftware

When you are on the desired repository page, click the **Fork** button as shown in Figure 2 below to perform the fork operation. [TODO: Change the screen shot after a Rally App has been added to the repository, remembering to highlight the Fork button]

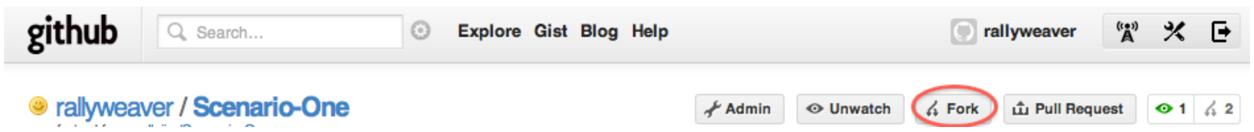


*Figure 2. The **Fork** button on the repository page*

If the dialog shown in Figure 3 appears, select your account as the one to which you want to fork the repository. [TODO: Change the screen shot after a Rally App has been added to the repository]

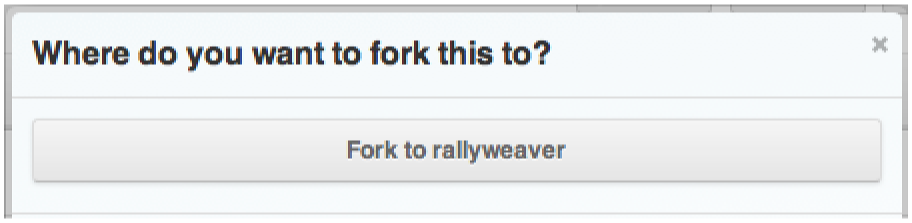


*Figure 3. The **Fork** dialog on the repository page*

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 4:

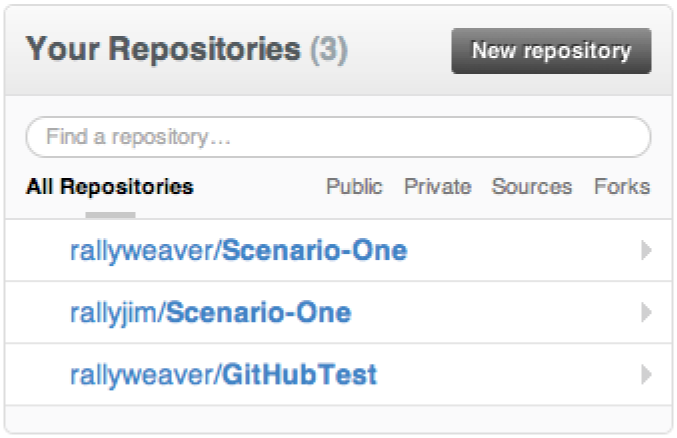


*Figure 4. Selecting a repository from the **Your Repositories** box*

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.

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 5. [TODO: Change the screen shot after a Rally App has been added to the repository, and draw a less pixelated oval]

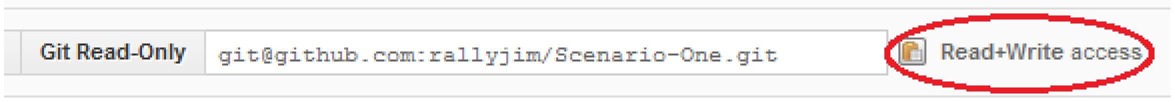
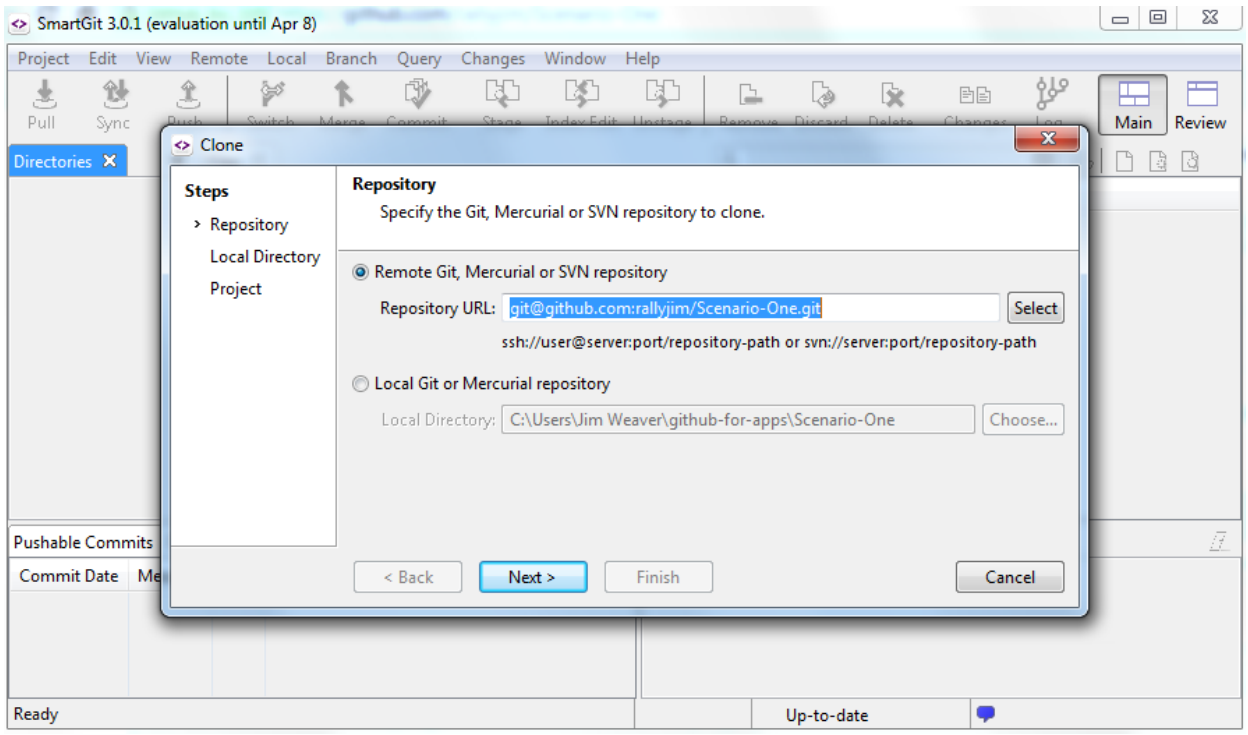


*Figure 5. The **Read+Write access** button on the repository page*

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 6a. [TODO: Change the screen shot after a Rally App has been added to the repository]



Repository
Specify the Git, Mercurial or SVN repository to clone.

(●) Remote Git, Mercurial or SVN repository
Repository URL: git@github.com:rallyjim/Scenario-One.git
ssh://user@server:port/repository-path or svn://server:port/repository-path

( ) Local Git or Mercurial repository
Local Directory: C:\Users\Jim Weaver\github-for-apps\Scenario-One

*Figure 6a. The **Repository** step of the **Clone** wizard*

Clicking the **Next** button may produce the dialog shown in Figure 6b, prompting you for the **master password**.

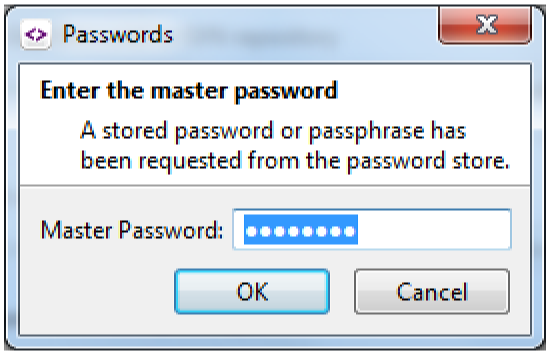
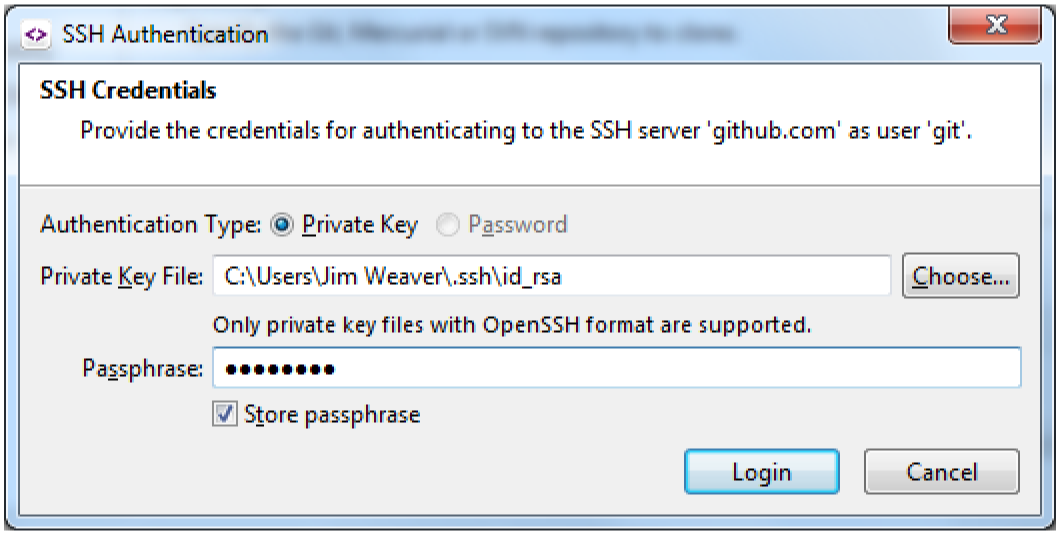

Passwords
Enter the master password
A stored password or passphrase has been requested from the password store.
Master Password: ●●●●●●●●
OK
Cancel

Figure 6b. 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 6c 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. [TODO: Change the screen shot after a Rally App has been added to the repository]


SSH Authentication

SSH Credentials

Provide the credentials for authenticating to the SSH server 'github.com' as user 'git'.

Authentication Type: [✓ Private Key] Password

Private Key File: C:\Users\Jim Weaver\.ssh\id_rsa

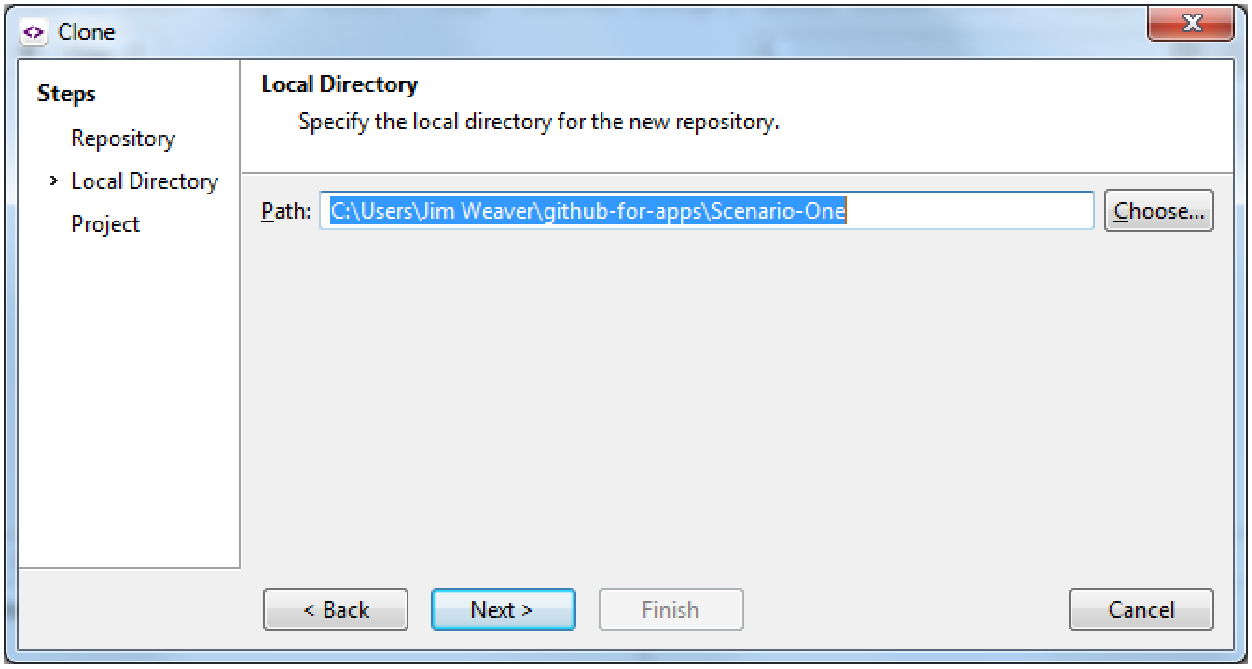
Only private key files with OpenSSH format are supported.

Passphrase: ●●●●●●●●●

[✓ Store passphrase]

Figure 6c. Entering SSH Credentials

In the **Local Directory** step of the **Clone** wizard shown in Figure 6d, specify the location in which you'd like the clone to be created. [TODO: Change the screen shot after a Rally App has been added to the repository]



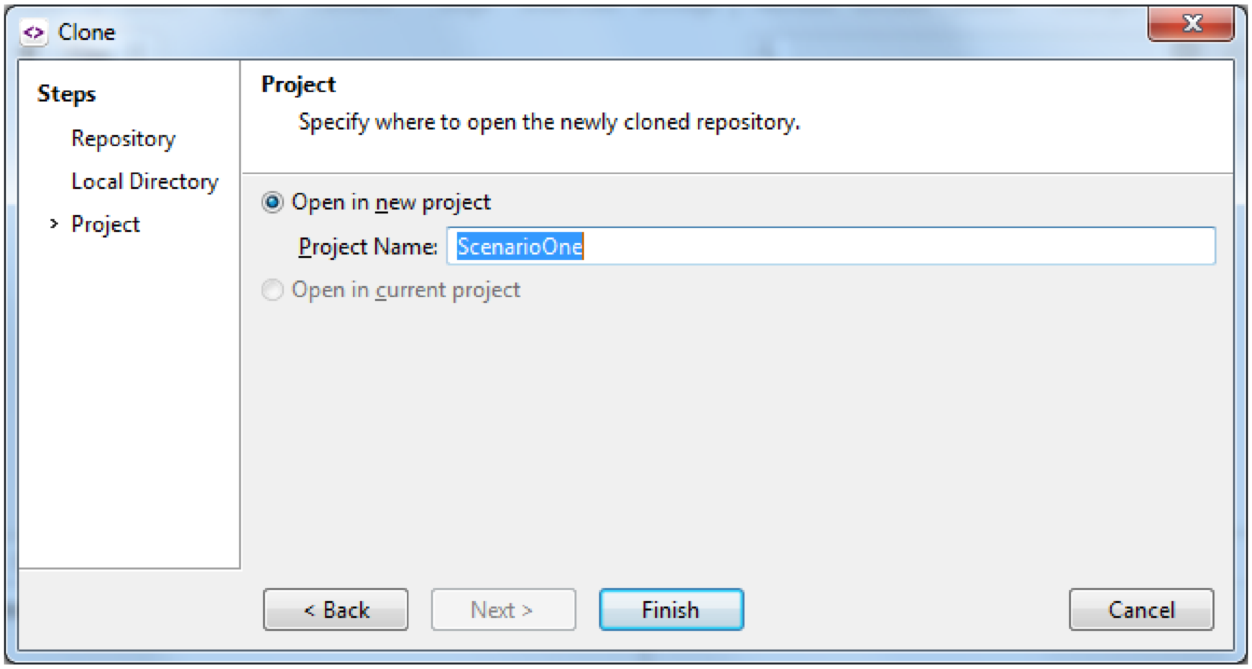
Local Directory

Specify the local directory for the new repository.

Path: C:\Users\Jim Weaver\github-for-apps\Scenario-One

*Figure 6d. The **Local Directory** step of the **Clone** wizard*

In the **Project** step of the **Clone** wizard shown in Figure 6e, specify a new SmartGit **Project Name** for the newly cloned repository. [TODO: Change the screen shot after a Rally App has been added to the repository]



Project

Specify where to open the newly cloned repository.

[✓ Open in new project]

Project Name: ScenarioOne

Open in current project

*Figure 6e. The **Project** step of the **Clone** wizard*

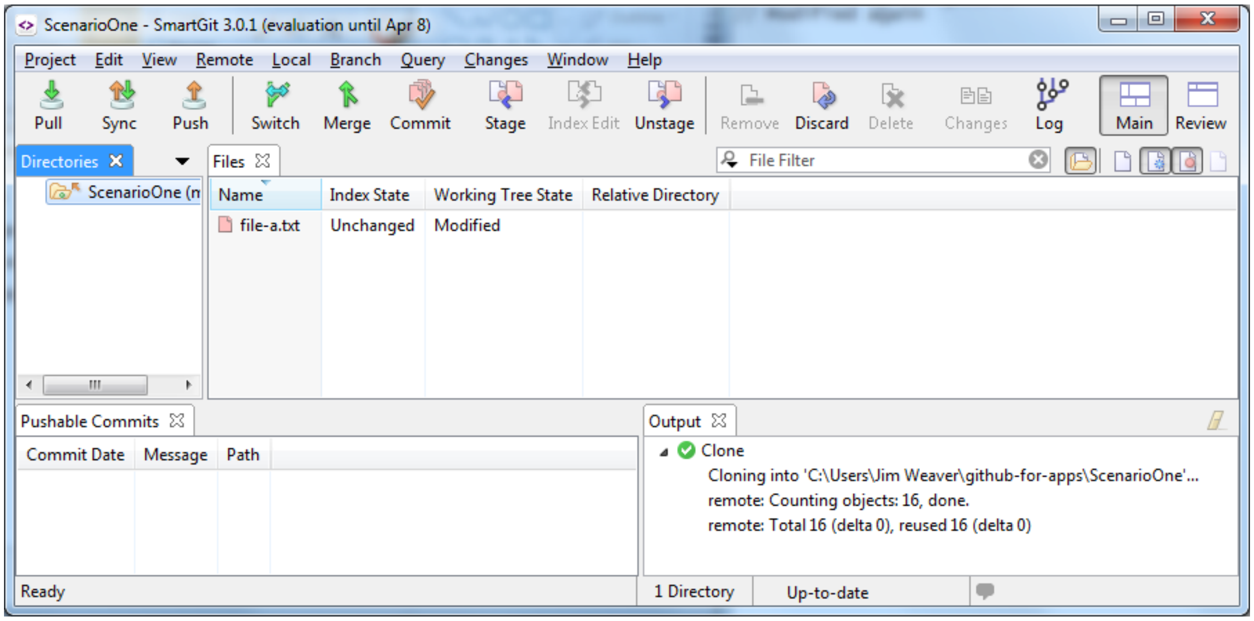
After clicking the **Finish** button, a full copy of the repository will be created on your local machine at the location you specified. Now that you have cloned the repository you can make modifications to the app, and continue on to the **commit and sync to the fork** step.

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

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 invoking SmartGit. Figure 7 shows the SmartGit main view containing an open project that represents a local repository, and any files that have been modified. [TODO: Change the screen shot after a Rally App has been added to the repository, and draw ovals around the Sync, and Commit, buttons]


ScenarioOne - SmartGit 3.0.1 (evaluation until Apr 8)
Project Edit View Remote Local Branch Query Changes Window Help
Pull Sync Push Switch Merge Commit Stage Index Edit Unstage Remove Discard Delete Changes Log Main Review
Directories
ScenarioOne (n
Files
File Filter
Pushable Commits
Output
Ready
1 Directory
Up-to-date

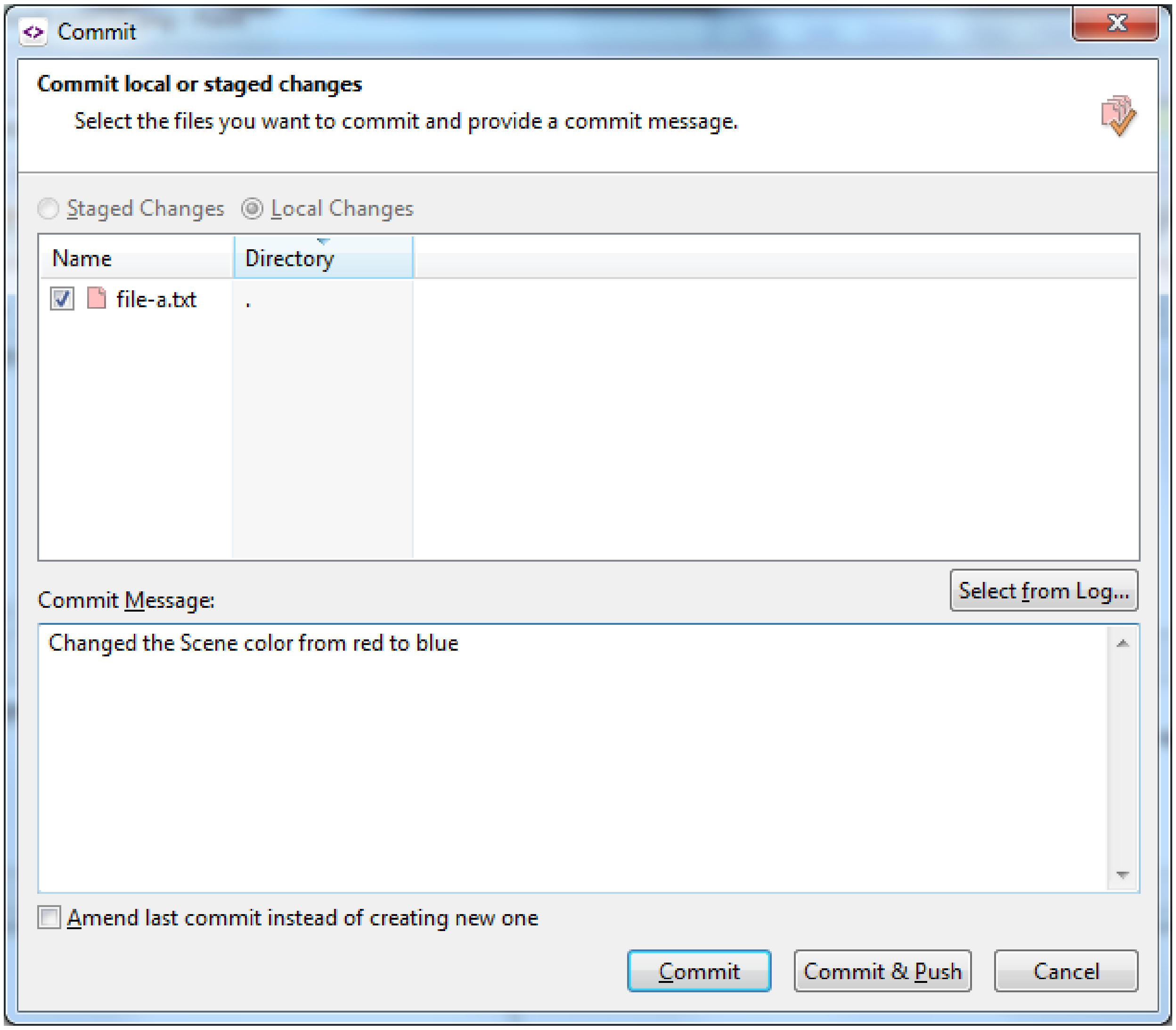
Name	Index State	Working Tree State	Relative Directory
file-a.txt	Unchanged	Modified	

Commit Date	Message	Path

```
Clone
    Cloning into 'C:\Users\Jim Weaver\github-for-apps\ScenarioOne'...
    remote: Counting objects: 16, done.
    remote: Total 16 (delta 0), reused 16 (delta 0)
```

Figure 7. SmartGit main view

Selecting at least one of the modified files and clicking the **Commit** button causes the **Commit** dialog shown in Figure 8 to appear. [TODO: Change the screen shot after a Rally App has been added to the repository]



Commit local or staged changes

Select the files you want to commit and provide a commit message.

Staged Changes [✓ Local Changes]

Name	Directory
✓ file-a.txt	.

Commit Message:

Changed the Scene color from red to blue

☐ Amend last commit instead of creating new one

Figure 8. SmartGit Commit dialog

After typing in the **Commit Message** “Change the Scene color from red to blue”, 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 7 to cause the remote repository to be synchronized with the local repository. Now that the remote repository has the modifications, we’ll need to request that the original repository gets updated as well by creating a ***pull request***.

Sending a pull request for inclusion into the original repository

To send a pull request, we’ll click the **Pull Request** button on the GitHub repository page shown in Figure 9. [TODO: Change the screen shot after a Rally App has been added to the repository, noting that when the screen shot is taken from a Windows machine the Clone in Mac button won’t be present as it incorrectly is in Figure 9]

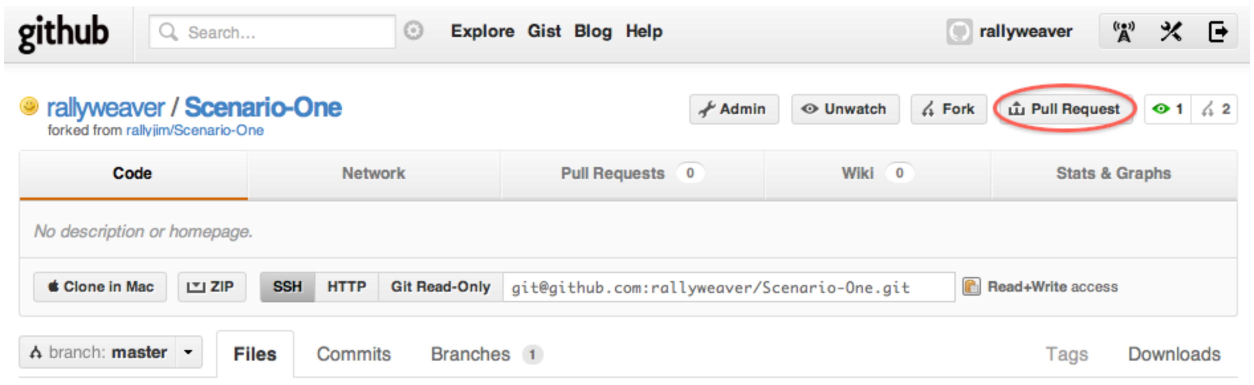
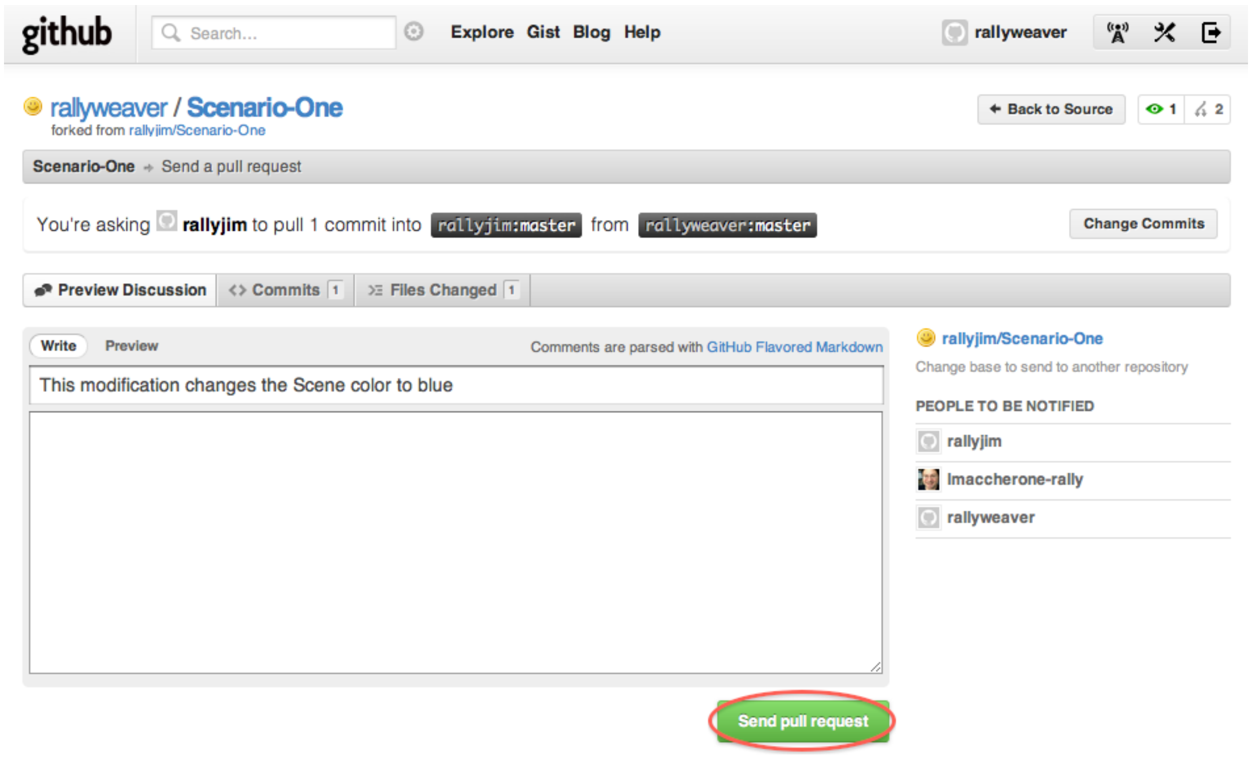


*Figure 9. The **Pull Request** button on the GitHub repository page*

As a result of clicking the **Pull Request** button, the page in Figure 10 appears in which you can view and modify information about the pull request. For example, we entered the message "This modification changes the Scene color to blue". When ready to send the request, click the **Send pull request** button. [TODO: Change the screen shot after a Rally App has been added to the repository]



You're asking rallyjim to pull 1 commit into `rallyjim:master` from `rallyweaver:master`

This modification changes the Scene color to blue

Figure 10. 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/

Now that we've examined the **maintaining an existing app** workflow, let's take a look at the **creating a new app** workflow:

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 original repository for your new app. Modifications made to the application may be committed and synchronized directly to this original repository, eliminating the need for the pull request discussed in the **maintaining an existing app** workflow discussed earlier. Here are the steps that you'll follow:

1. Fork the repository that contains the app or app template
2. Rename the project
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

To make these steps easier to visualize, Figure 11 depicts them graphically:

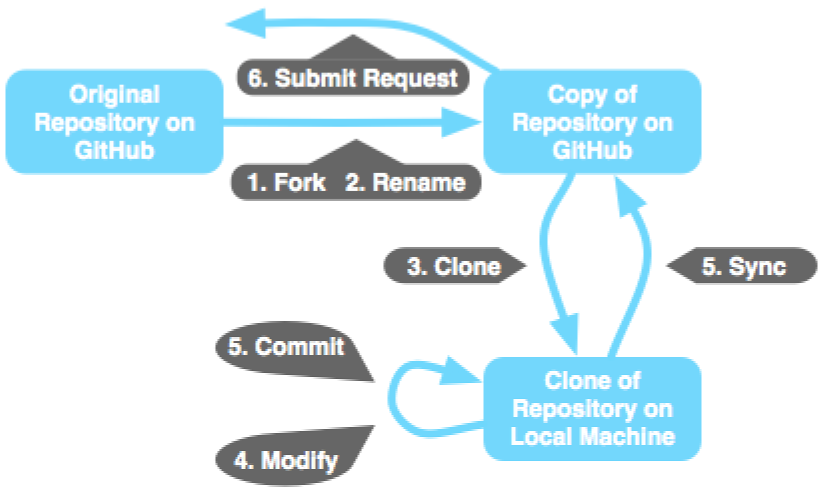

Original Repository on GitHub
6. Submit Request
Copy of Repository on GitHub
1. Fork 2. Rename
3. Clone
5. Sync
5. Commit
Clone of Repository on Local Machine
4. Modify

*Figure 11. The **create a new app** workflow*

Many of these steps are identical to the steps in the **maintaining an existing app** workflow, so some information previously covered will be repeated for your convenience.

Forking the repository that contains the app or app template

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 RallySoftware account on GitHub, which is located at: https://github.com/organizations/RallySoftware

When you are on the desired repository page, click the **Fork** button as shown in Figure 12 below to perform the fork operation. [TODO: Change the screen shot after a Rally App has been added to the repository, remembering to highlight the Fork button]

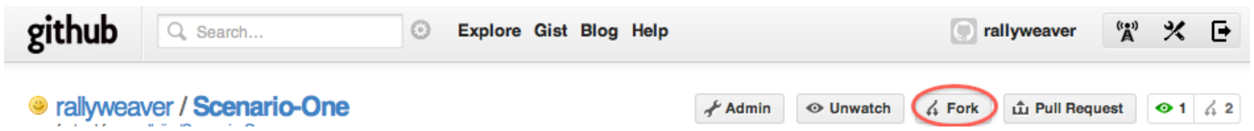


*Figure 12. The **Fork** button on the repository page*

If the dialog shown in Figure 13 appears, select your account as the one to which you want to fork the repository. [TODO: Change the screen shot after a Rally App has been added to the repository]

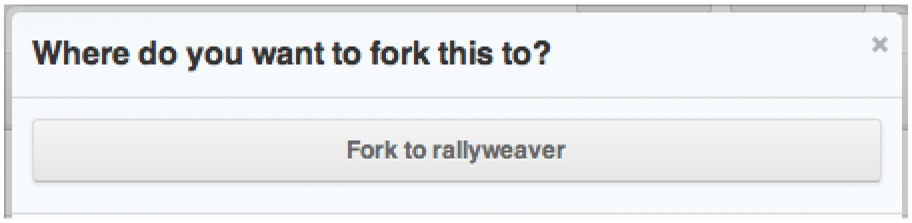


*Figure 13. The **Fork** dialog on the repository page*

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 14:

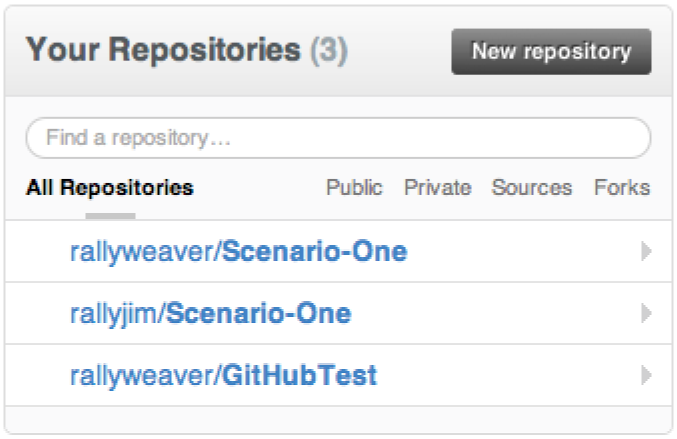


*Figure 14. Selecting a repository from the **Your Repositories** box*

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

Renaming the Project

Because you'll be creating a new app, you need to rename the new repository that you created in the previous step. To do this, click the **Admin** button on the GitHub repository page shown in Figure 15. [TODO: Change the screen shot after a Rally App has been added to the repository]

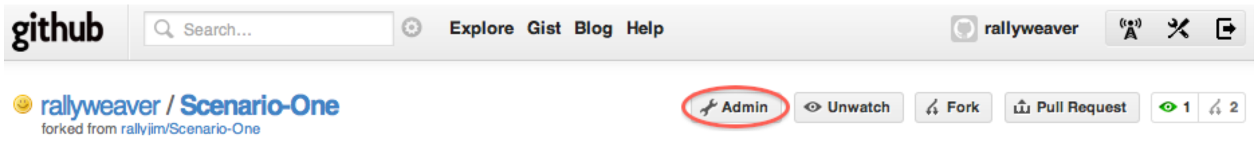
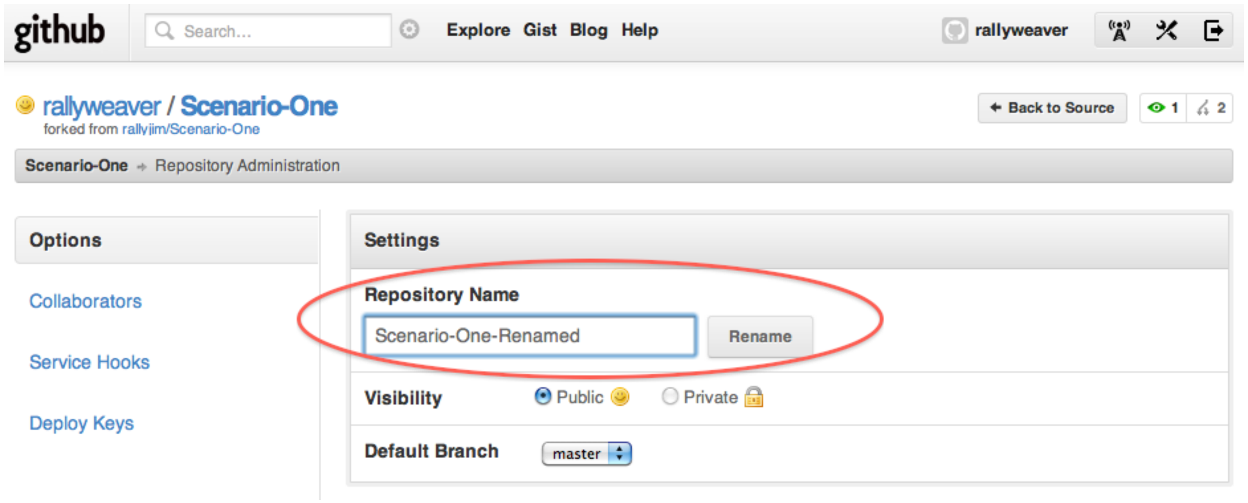


*Figure 15. The **Admin** button on the GitHub repository page*

Then, from the GitHub repository admin page shown in Figure 16, enter a new **Repository Name** and click the **Rename** button. [TODO: Change the screen shot after a Rally App has been added to the repository]



Repository Name: Scenario-One-Renamed

Visibility: [✓ Public] Private

Default Branch: master

Figure 16. The GitHub repository admin page

Cloning the New Repository

[TODO: Continue modifying the document to reflect SmartGit instructions, copying from the "Maintaining an Existing App" section after it has been reviewed and tested]

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 **Clone in Mac** button on the repository's page as shown in Figure 17.

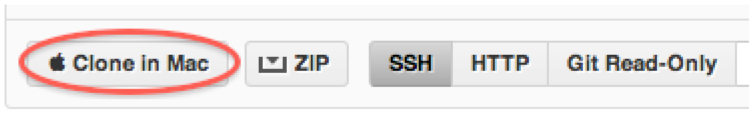


*Figure 17. The **Clone in Mac button** on the repository page*

After clicking the Clone in Mac button, GitHub for Mac should be invoked as shown in Figure 18, prompting you to specify the location in which you'd like the clone to be created. [TODO: Change the screen shot after a Rally App has been added to the repository]

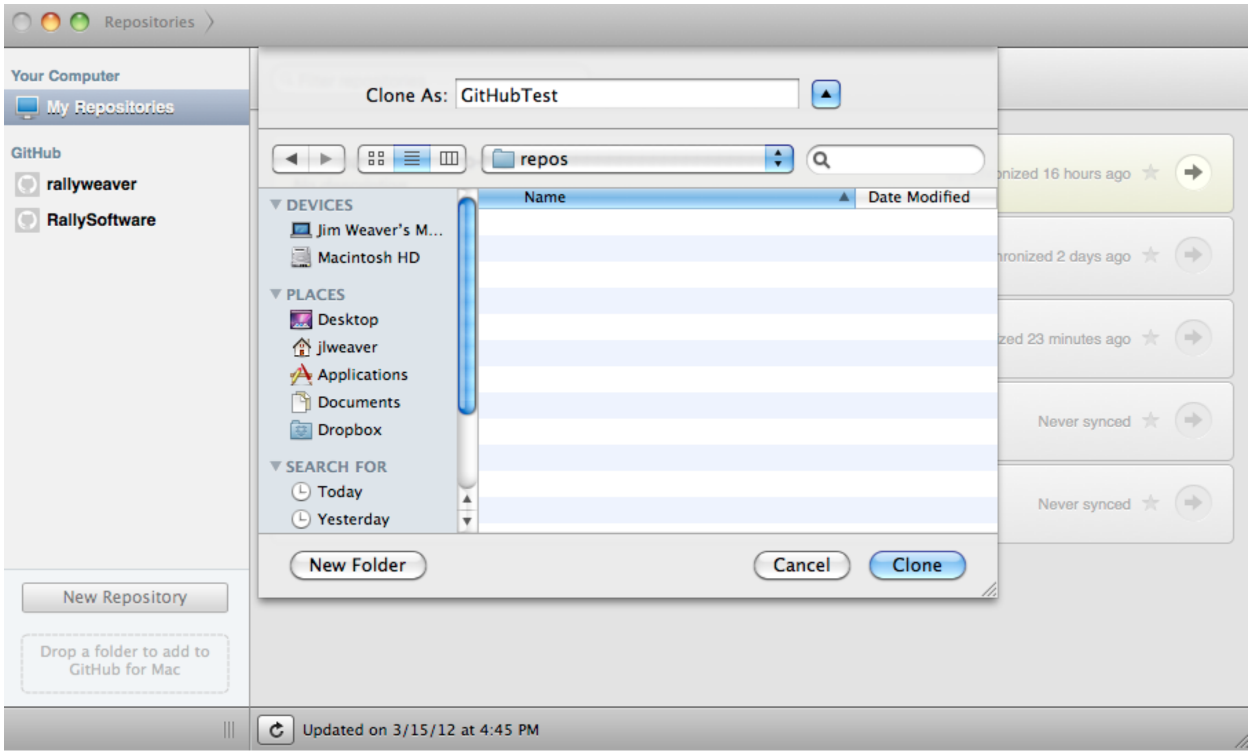


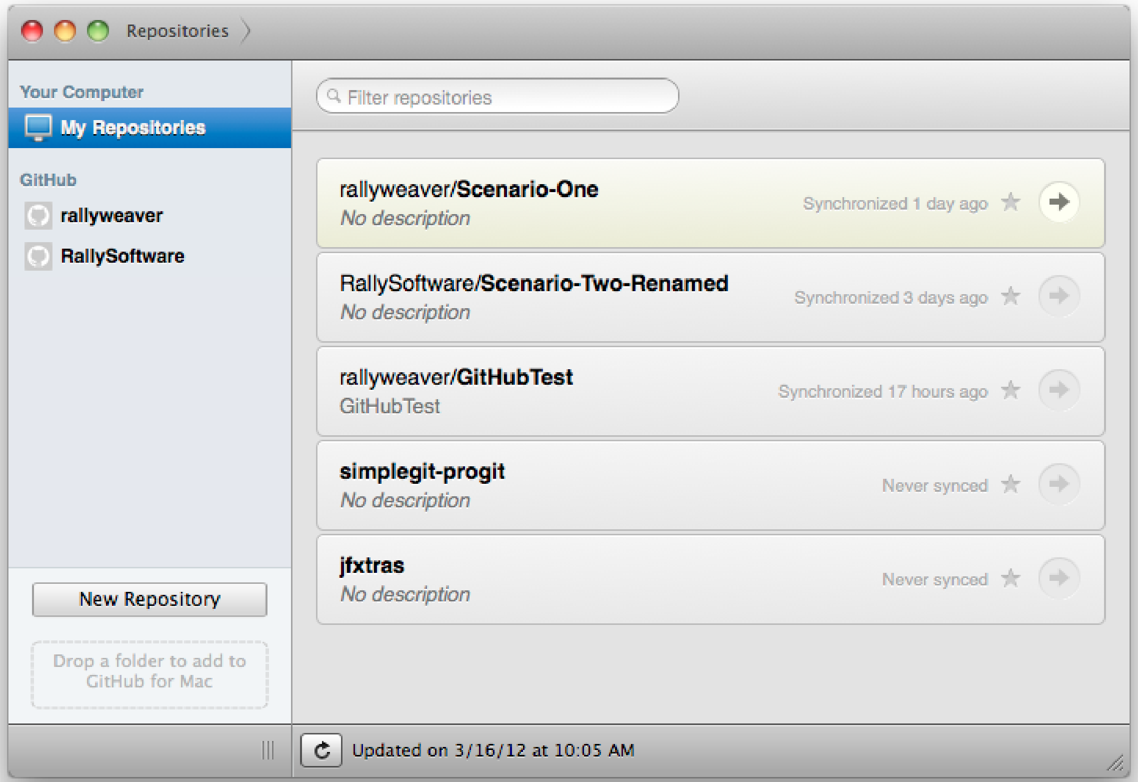
Figure 18. GitHub for Mac prompting for location to save the clone

A full copy of the repository will be created on your local machine at the location you specified. Now that you have cloned the repository you can make modifications to the app, and continue on to the **committing and syncing to the fork** step.

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. As you'll see shortly, both of these operations may be performed in one step using GitHub for Mac.

We'll begin performing these operations by invoking GitHub for Mac. Figure 19 shows the GitHub for Mac startup screen that contains a list of local repositories. [TODO: Change the screen shot after a Rally App has been added to the repository]



rallyweaver/**Scenario-One** — *No description* — Synchronized 1 day ago

RallySoftware/**Scenario-Two-Renamed** — *No description* — Synchronized 3 days ago

rallyweaver/**GitHubTest** — GitHubTest — Synchronized 17 hours ago

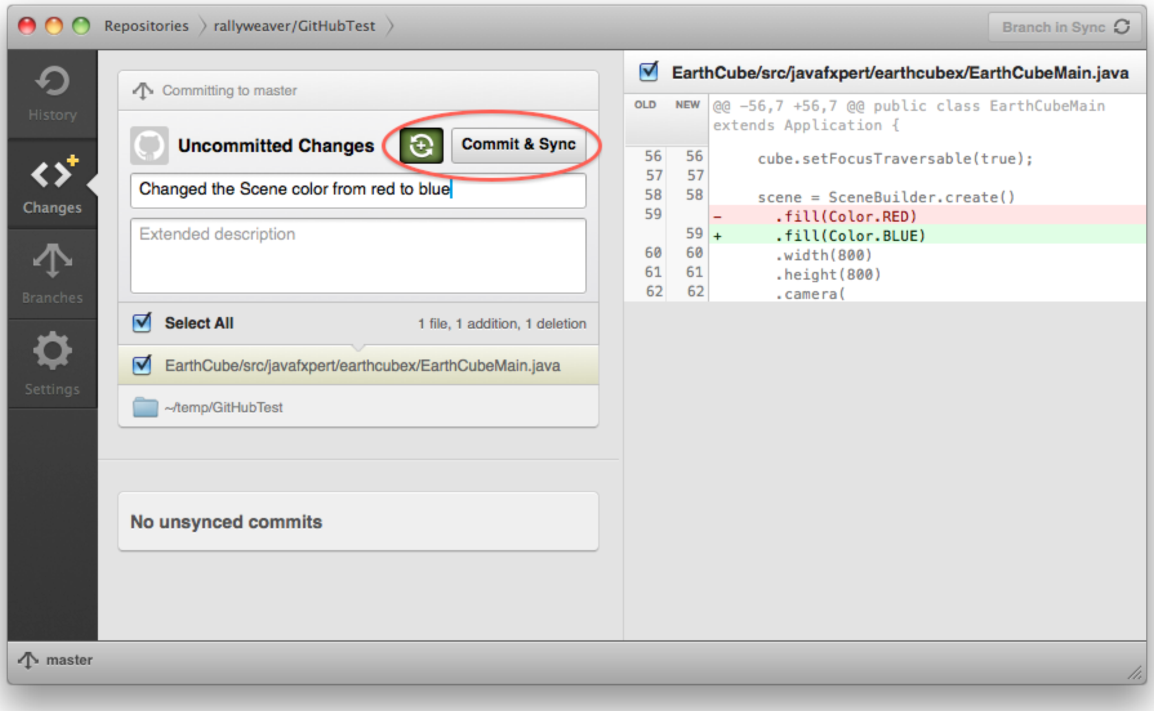
simplegit-progit — *No description* — Never synced

jfxtras — *No description* — Never synced

Updated on 3/16/12 at 10:05 AM

Figure 19. GitHub for Mac local repository list

We'll then double-click on the repository that we'd like to commit and sync, which as shown in Figure 20 navigates to the **Changes** tab of the repository detail view. [TODO: Change the screen shot after a Rally App has been added to the repository]



Changed the Scene color from red to blue

EarthCube/src/javafxpert/earthcubex/EarthCubeMain.java

```
@@ -56,7 +56,7 @@ public class EarthCubeMain
extends Application {
        cube.setFocusTraversable(true);

        scene = SceneBuilder.create()
-         .fill(Color.RED)
+         .fill(Color.BLUE)
          .width(800)
          .height(800)
          .camera(
```

Figure 20. GitHub for Mac repository detail view

Notice that in Figure 20 there is a button labeled **Commit & Sync**, and that clicking the button to its left causes this button's label to toggle between **Commit & Sync** and **Commit**. We're using the **Commit & Sync** setting so that both operations are performed with one button click.

After typing in the **Commit Summary** message "Change the Scene color from red to blue", we'll click the **Commit & Sync** button which causes the changes to be committed to the local repository, and for the remote repository to be synchronized with the local repository. Now that the remote repository has the modifications, we'll submit a request for the new app to be added to the Rally Apps library.

(optional) Submitting your App to be Added to the Rally Apps Catalog

To submit your App for possible inclusion in 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.

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 **GitHub for Mac**
- Use GitHub and the installed software to implement the **two development workflows** in the pilot, which are: **maintaining an existing app**, and **creating a new 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.