Getting Started Guide to Robot Lifecycle Management in Kofax RPA

[david.wright@kofax.com](mailto:david.wright@kofax.com) , 19 May 2020.

*This guide assumes that you have Kofax RPA 10.4 (or later) installed and that the Management Console is licensed and running. This guide is also compatible with the Trial Version of Kofax RPA.*

*This guide assumes no knowledge of Git, nor does not require usage of Git at the command line. It uses GitHub as a robot repository.*

# Introduction

Robot Lifecycle Management (RLM) helps you manage

* robot development by yourself or with a team of robot builders.
* different versions of robots.
* automatic deployment of robots and schedules to other systems, like test or production.
* sharing robots privately or publicly with others even across the internet.
* improving robot security by knowing who changed what, where and when.

Robot Lifecycle Management is built on **Git**, <https://git-scm.com/>, *a free and open-source distributed Source Control Management (SCM) system for tracking changes in source code during software development*, invented by Linus Torvald in 2005, who also invented Linux.

Git is best thought of as a tool for storing the history of a collection of files. It stores the history as a compressed collection of interrelated snapshots of the project’s contents in the Git **repository**. In Git each such version is called a **commit**.

RLM connects to a Git Repository which can be on

* a folder on your machine
* File shares on your intranet
* A Git server at your company or customer.
* **GitHub**, Web-based version control repository https://github.com, which Microsoft acquired in 2018 (the free plan has unlimited private repositories with up to 3 collaborators & unlimited public repositories)
* **BitBucket**, Web-based version control repository [https://bitbucket.com](https://bitbucket.com/), acquired by Atlassian in 2010 (the free plan has unlimited private repositories with up to 5 users)
* Or any other cloud-based Git Repository, like **CloudForge**, **Sourceforge**, **GitLab**, etc..

In this Guide we will use **GitHub**, as it is easy. Local folders and local Git Server are more advanced.

# Basic Architecture

**Robots** are built in **Projects** in the **Design Studio**, and the robot files are stored as XML files typically in the local folder **My Documents\My Robots.**

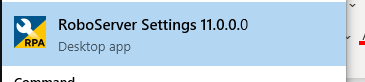
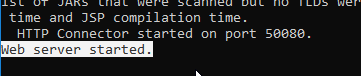
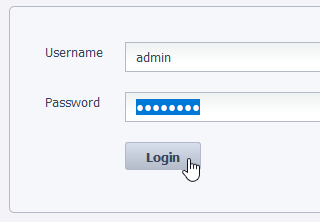
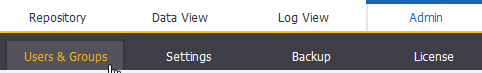
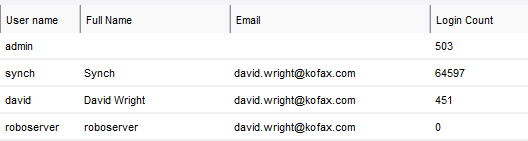
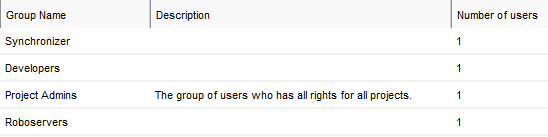
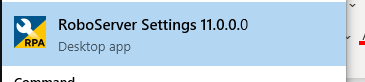
The **Design Studio** can upload/download/synchronize robot files in **projects** from the **Management Console**, which stores its robots in the Management Console database. Numerous Design Studios can be connected to the same Management Console – this is how colleagues can collaborate closely.[[1]](#footnote-1)

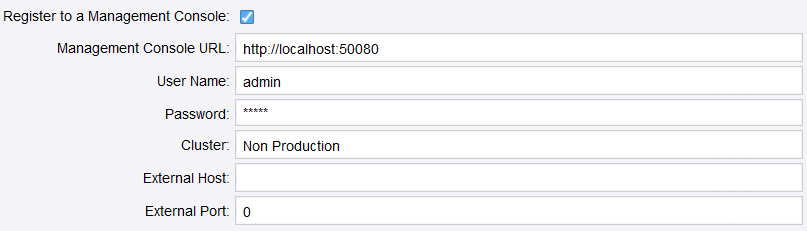
Each Management Console has its own **Synchronizer** which keeps **projects** synchronized between Management Console and a Git Repository every 10 seconds.[[2]](#footnote-2)

If a robot is changed in the Git Repository then within a few seconds the Design Studio user will see an icon next to the robot, showing if the robot requires downloading or that it is in conflict with the repository, because someone else changed it at the same time as them and committed it to the Repository.

# Enable User Management in Kofax RPA

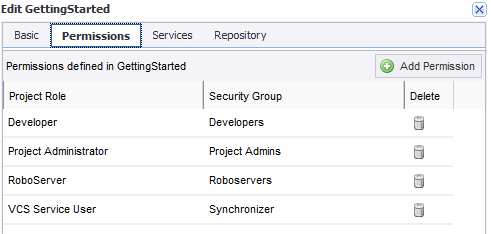
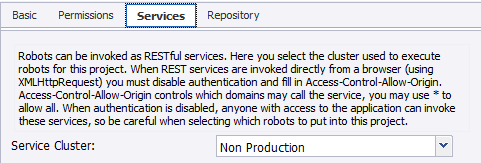
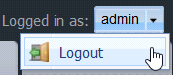
*It is recommended to enable User Management in Management Console. User Management is required to know who made a change to a robot. Without User Management Enabled, Management Console shows* ***anonymous*** *as the creator and modifier of robots.*

1. Shut down Management Console if it is running.
2. Open **Roboserver Settings** from Windows Start Menu/Kofax RPA.  
   
3. On the **Management Console** tab, select **Enable User Management**, and enter an Admin user-name: **admin** is recommended. Enter a password. This is a top-level Administrator account – all other users are managed from this account within the Management Console  
   
4. Click **OK** to close Roboserver Settings.
5. Start Management Console with **Start Management Console** from the Windows Menu  
   
6. Wait for **Web server started.** to appears. The Management Console is now ready.  
   
7. Open Management Console [http://localhost:50080](http://localhost:50080/) in a browser and log in as **admin**.  
   
8. Go to **Admin/Users&Groups** tab that is now visible.  
   
9. Add three users with passwords
   1. one for yourself (so you can build robots and use the Management Console)
   2. one called **synch**, that will synchronize Management Console to your Git repository.
   3. One called **Roboserver**, that will allow the roboserver to log in to the Management Console.  
      
10. Add 4 groups –**Developers** (robot builders), **Synchronizers, Roboservers** and **Project Admins** (can see & edit project settings in Management Console). *You need groups, because permissions are assigned to groups of users, not individual users.*
11. Add the users to the appropriate groups. Roboserver to **Roboservers**, Synch to **Synchronizers**, and yourself to **Developers** and **Project Admins**.
12. Go back to Roboserver Settings in the Windows Menu and give it the new credentials.  
    
13. On the General tab, select Register to a Management Console.

Enter [http://localhost:50080](http://localhost:50080/) for a single computer solution. Enter the roboserver username and password you chose. Enter your cluster name, typically **Non Production** for new users. ****

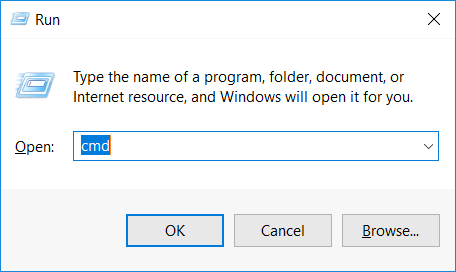
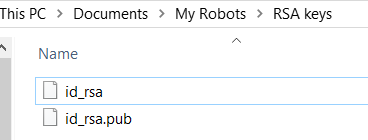
1. Press OK.

# Add Permissions to a new Project

1. Go to the **Admin/Projects** tab in Management Console and **create** a new project called **GettingStarted**, that will hold your robots, user permissions and Git configuration.
2. Add four roles to the project **Developer, Project Administrator, Roboserver** and **VCS Service User**. Assign your 3 user groups to them.  
   *VCS=Version Control System*. Git is a Version Control System.  
   
3. On the **Services** tabselect the Service Cluster **Non Production** so your robots have a place to run. *You should not be developing robots on a Production server anyway! You will use the Synchronizer to move robots into Production in section 14.*
4. **Save** the Project Settings without changing the Repository Settings. *You will create and configure the Git Repository in Section 9 and get the Synchronizer running in Section 6.*
5. **Logout** ofManagement Console and log in with your Project Admin account. Make sure you can log into the Design Studio and that you can also see your project settings in Management Console.  
    

# Creating a Private & Public key for the Synchronizer

You need to generate a private and public RSA key pair so that Synchronizer has permission to read & write to your Git repository (Git Server, GitHub, Bitbucket, etc)

1. In Windows Explorer create a folder called **RSA Keys** in the **My Robots folder**.   
   
2. Open a Command Prompt by press **Windows Key+R** and typing **cmd** and ENTER.  
   
3. At the Command Prompt type and press ENTER  
   Synchronizer -g "%userprofile%\Documents\My Robots\RSA keys"
4. Inside the folder in Windows Explorer you will see two files. **id\_rsa** is your **private key** – only you have it. Protect it well – share it with no one – a hacker who gets it could change your robots remotely and deploy them onto your machine. The **public key** **id\_rsa.pub** can be freely shared with anyone.  
   *The private key is used to authenticate with and the public key is used by others to verify that you are who you say you are.  
   In Section 8 we will give* ***the public key*** *to* ***GitHub*** *and give* ***the private key*** *to the* ***Synchronize****r – this is how GitHub will trust your Synchronizer.*  
   

# Configure and Start the Synchronizer

The Synchronizer synchronizes robots between Management Console and Git repositories both uploading and downloading.

It needs to be configured with **access** to the Management Console, and the private RSA key for access to Git repositories.

1. Copy the text below to the clipboard (Make sure you insert your synch account’s password).  
   Synchronizer.exe -c --mc-url <http://localhost:50080> --username synch --password \*\*\*\*\* --interval 10 --no-host-key true --private-key "%USERPROFILE%\Documents\My Robots\RSA keys\id\_rsa" -s
2. At the Command Prompt, paste it by right-clicking once.[[3]](#footnote-3)  
   *This configures the Synchronizer to check the Management Console every 10 seconds using the Kofax RPA account synch¸ with the password synch, and gives the Synchronizer the private key that it uses to access the Git Repository.*
3. Use the left arrow key to move cursor to \*\*\*\*\*\* and change it to the Synchronizer’s password you created in step 10 and press ENTER.
4. The Synchronizer encrypts the password, private key and other settings and stores them in  
   **%appdata%\..\Local\Kofax RPA\$Version\_Number$\Configuration\synchronizer.settings** .  
   This file will be used by the Synchronizer whenever it starts.
5. Start the Synchronizer from the Windows Menu[[4]](#footnote-4)  
   
6. It will show the following when successful  
   Unable to write to the configured log directory: C:\Program Files\Kofax RPA 10.4.0.0 183 x64\bin (Access is denied. (0x5))  
    The Wrapper may also have problems writing or rolling the log file.  
    Please make sure that the current user has read/write access.  
   --> Wrapper Started as Console  
   Launching a JVM...  
   WrapperManager: Initializing...
7. You can see the Synchronizer log files in **%appdata%\..\Local\Kofax RPA\$Version$\Logs\**
8. You will need to leave the Synchronizer running in the background as an application.

# Create a GitHub Account

This is identical to using a local Git server. You can also use <https://bitbucket.org>, or another other Git repository you prefer.

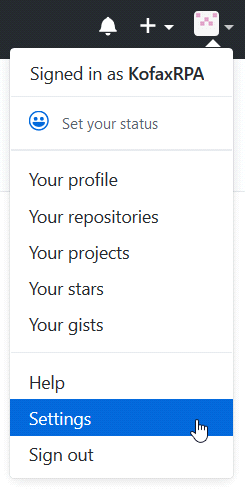
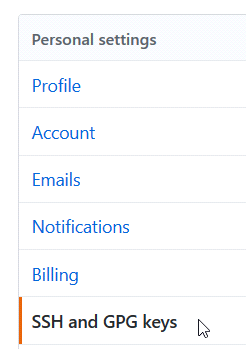
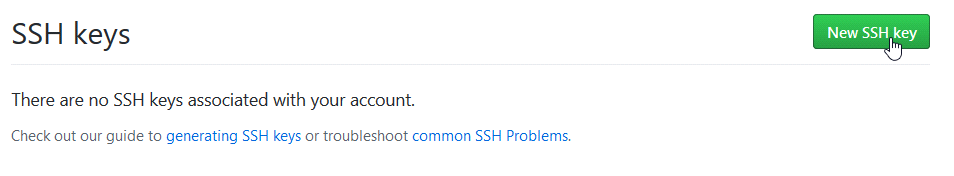
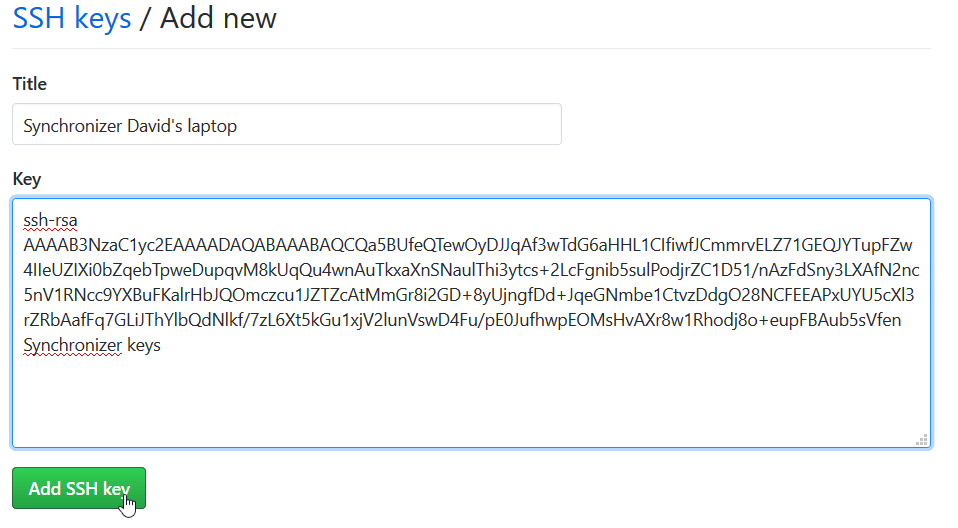
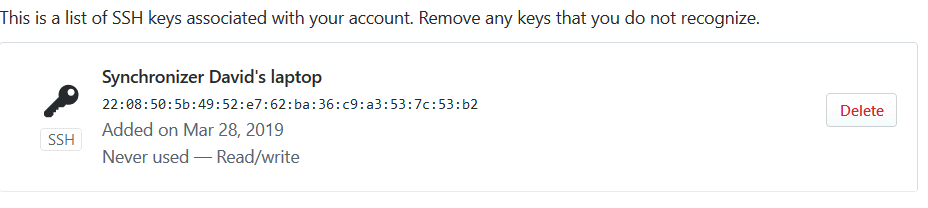
1. Sign up for a free account at GitHub at <https://github.com>. *Any robots you upload to GitHub are private – no-one else can see them. With the free version of GitHub you can add collaborators. You can make any repository public for anyone to be able to find your robots.*

*See* [*https://github.com/KofaxRPA*](https://github.com/KofaxRPA) *for many robot repositories from Kofax that you can use.*

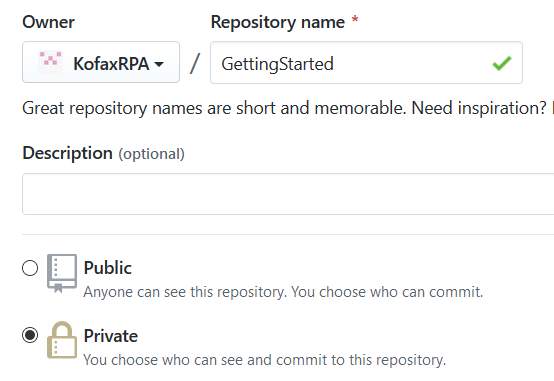
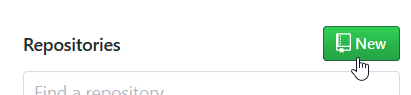
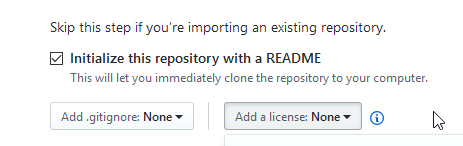
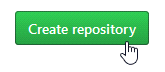
[*https://github.com/KofaxRPA/GettingStarted*](https://github.com/KofaxRPA/GettingStarted) *is the GitHub repository used for creating this guide.*

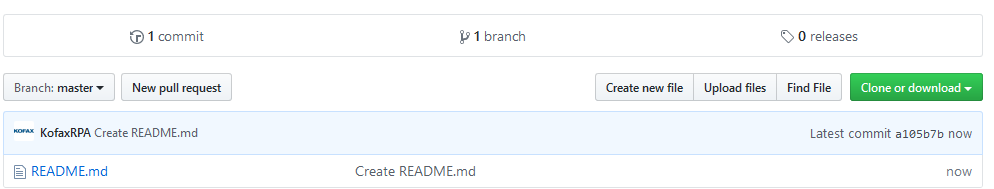
# Give GitHub your Synchronizer’s public RSA key

This is how GitHub will identify and trust your Synchronizer to change files in your repositories.

1. Go to [https://GitHub.com/settings/keys](https://github.com/settings/keys) or click on the  Icon in the top-right-corner next to the bell. Go to **Personal** **Settings** in GitHub (not the Repository Settings) and select **SSH and GPG keys  
    **
2. **Click New SSH Key  
   **
3. Open the file **"%UserProfile%\Documents\My Robots\RSA keys\id\_rsa.pub"** with Notepad and copy it to the clipboard.
4. Givethe SSH Key a **Name** and paste in your **public** key from **My Documents\My Robots\RSA keys\id\_rsa.pub**. Press **Add SSH Key***The key starts with* ***ssh-rsa*** *and you can type anything you like on the last line.* ****
5. Your GitHub Account will now trust your Synchronizer (and anyone who steals the private key **id\_rsa**!)  
   

# Create a Repository on GitHub

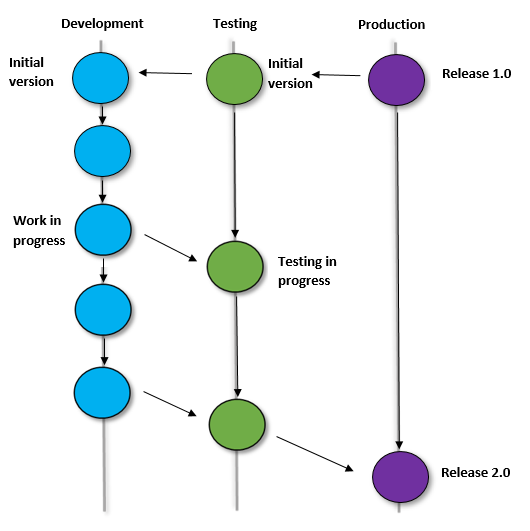
1. On [https://github.com](https://github.com/) create a **New Repository**. Give it a **Name** and set it to **Private**.  
   
2. Select **Initialize this repository with a README** and add a license if you intend it to be public later. ****
3. Click **Create repository.**  
   

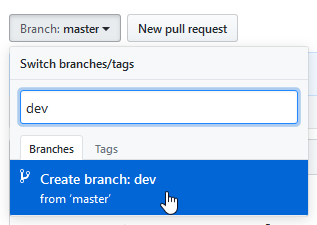
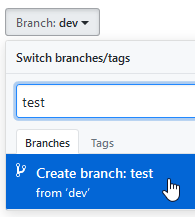
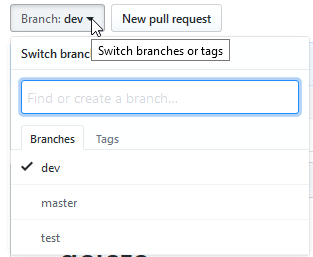
You now have a new Git repository! It has one branch called **master** and contains 1 file.  


# Create a Development Branch in your GitHub Repository

*In this guide we are going to take a robot from* ***development*** *to* ***testing*** *and then into* ***production.***

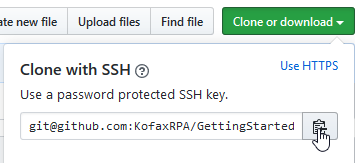
*We will be following this Git branching model[[5]](#footnote-5) with three branches, where the default* ***master*** *branch is used for Production.*

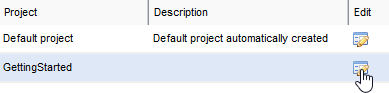
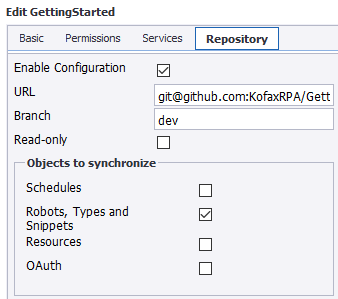


1. Open your Repository in GitHub.
2. Click on **Branch**  
   
3. Type “**dev**” and select **Create branch:** **dev** from master.  
   
4. Make another Branch called **test.**
5. Switch **GitHub** to the **dev** branch so that we can see the robot being developed. We will tell the Synchronizer to upload to the **dev** branch. The master branch will remain without a robot. The **test** branch will also remain empty until we **merge** the dev branch to the test branch in Section 18. After testing you can merge the **test** branch to the **master** branch.  
   

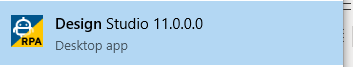
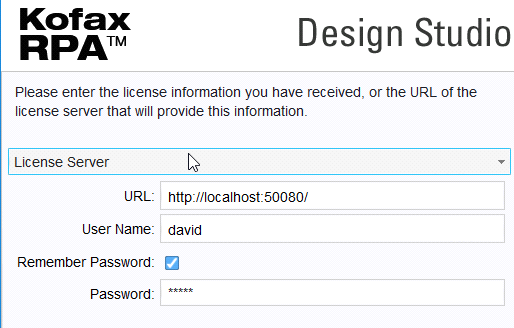
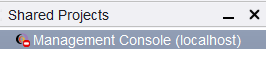
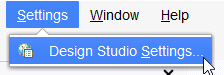
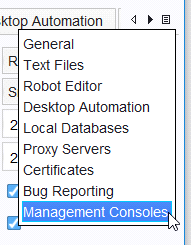
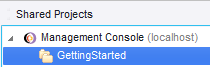
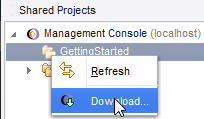
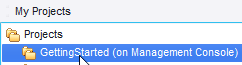
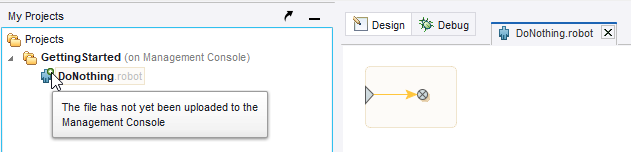
# Connect the RPA project to the Git Repository

1. Open your repository in Github

Click **Clone or download** and copy the SSH URL to the clipboard. *Make sure you use the SSH link and NOT the HTTPS link.*  


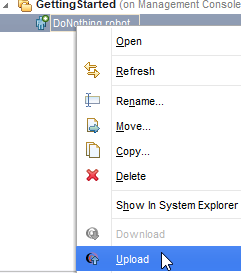
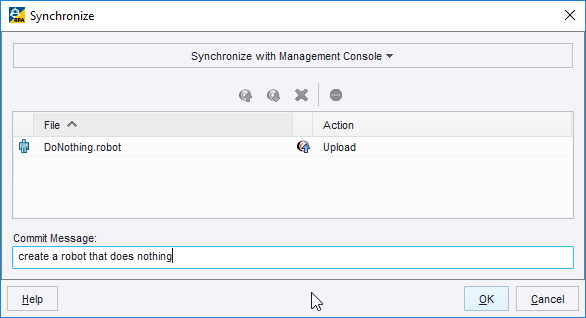
1. In Management Console Open the **Admin/Project** tab and Click Edit on your **GettingStarted** project.  
   
2. On the **Repository** tab enter the repository link in the URL field.  
   Type **dev** into the Branch and select “Robots, Types and Snippets”.  
   **Enable Configuration**  
   Do **not** select Read-only, because you want to write robots to the **dev** branch of the repository. *In your test environment you will select Read-only and the branch =* ***test.***Select **Robots, Types and Snippets**.  
   **WARNING!** Do **not** select Schedules or the others until you need them. If you select Schedules and someone steals your private key, then an attacker could remotely download schedules to your Management Console and execute robots of their choice.  
   
3. Press **Save**Within 10 seconds your Management Console will be connected to GitHub via the Synchronizer.  
   *Make sure that* ***Kofax RPA/Synchronizer*** *is running from the Windows Start Menu, see section 6.*

# Create a new Robot in your Project

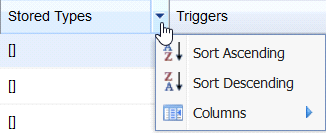
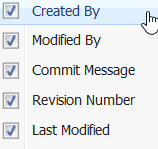
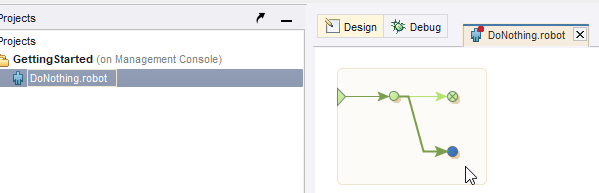
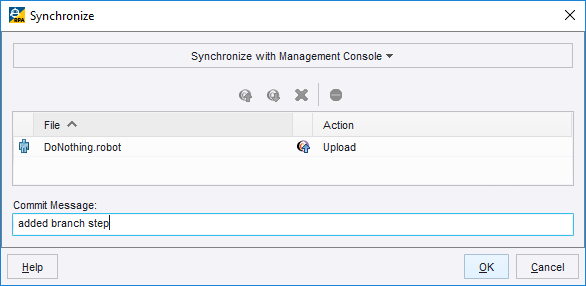
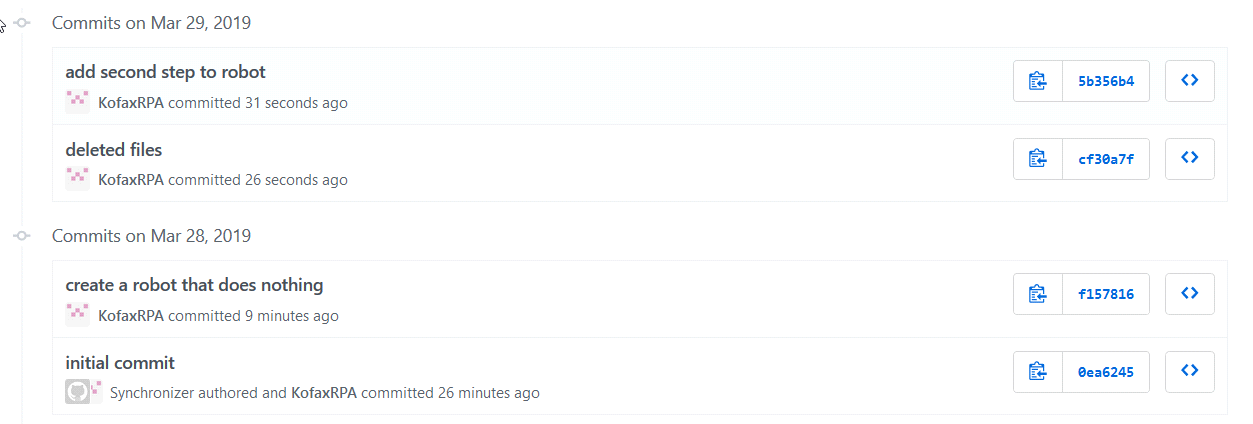
1. Start Design Studio from the Windows Menu  
   
2. Login with the **Developer** account that you created in section 3 to the License Server in the Management Console.  
   *You need to do this so that the Git Repository knows who you are when you make changes to robots.*  
   
3. *If you cannot see the Management Console in Design Studio, go to* ***Settings/DesignStudioSettings/ManagementConsoles*** *and add or correct the account you are logging in with. If your developer account is not attached to any project in Management Console, you won’t be able to log in****.*  
   ** 
4. You can now see the active Management Console and your empty project in the **Shared Projects** window.  
   ****
5. Right-Click on the project and **Download** the empty project to your Design Studio and select a local folder for it. This links your local folder to the Management Console, and the Synchronizer links the Management Console to the Git Repository.  
   ****
6. The project now appears in **My Projects** Window ****
7. Add a new Web Automation Robot to the project called **DoNothing.robot***You can see that* the robot has a **green plus icon** showing it has not been uploaded to the Management Console.  
   ****

# Commit a Robot to the Repository

*When you upload this robot to Management Console, the Synchronizer will push it within 10 seconds to Github.*

1. Right-click on the empty robot and select **Upload****
2. You will now be asked for a **commit** **message[[6]](#footnote-6)**.  
   *This will be stored in the Git repository – we can see all the commit messages in GitHub and revert to this version of the robot later. Enter a message that describes what you just did.*  
   **

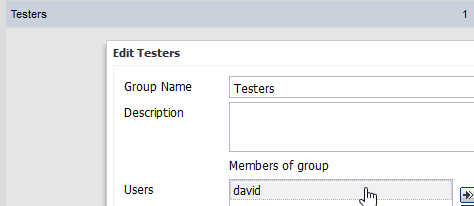
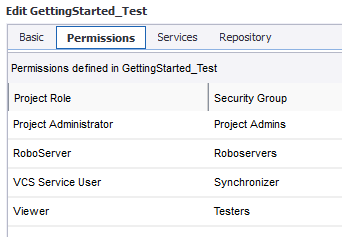
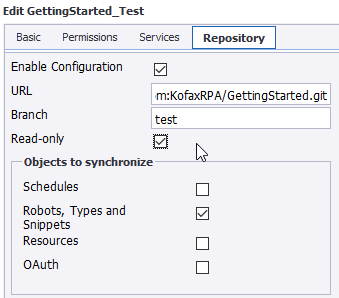
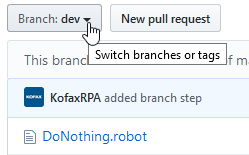
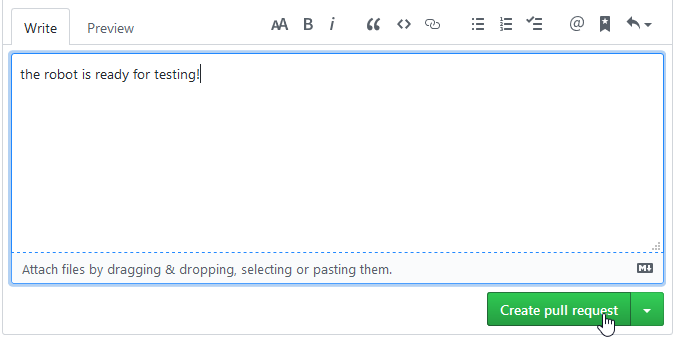
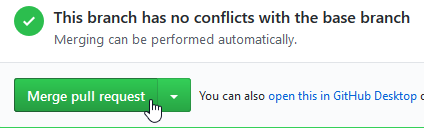
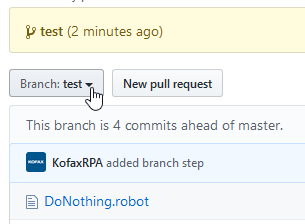
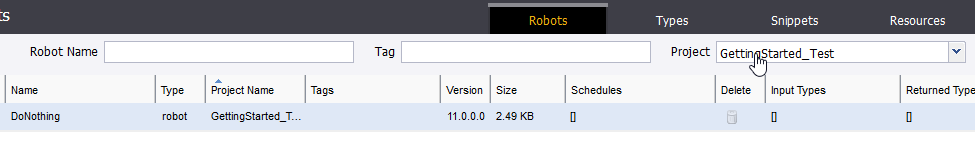
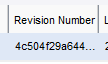
Each file that is in the Synchronize file list will be “committed”. This can be numerous robots, snippets, and types.  
The commit contains all files that are changed together. Unchanged files cannot be committed.

1. In the **Management Console** under **Repository**/**Robots** make sure these 5 columns are enabled by selecting the dropdown list at the right edge of any header cell: **Created By, Modified By,** **Commit Message**, **Revision Number** and **Last Modified**  
    
2. You can now see your most recent Commit Message along the commit details. The **Revision** **Number**, (Git commit Id) is a unique identifier for this commit
3. Go to GitHub, refresh the Repository page, click on **commits** and you will see commit details there, with the same commit id.  
   
4. *Let’s make a change to the robot and recommit*. Go back to Design Studio. Add a branch.  
   
5. Commit the robot again by right-clicking the robot and selecting **Upload** or **Synchronize. Add** a commit message   
   
6. Go to GitHub, wait 10 seconds, refresh the browser and you will see the growing history of commits.  
   

# Deploy a robot into Test or Production

You can use Git to deploy a robot to another project, or cluster or even another Management Console.

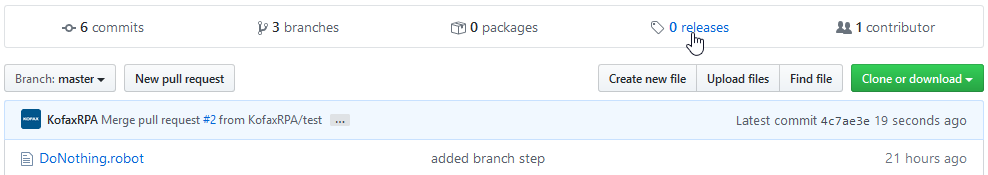
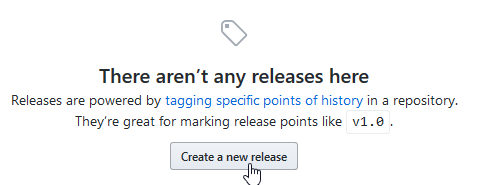
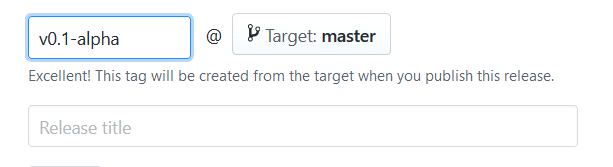
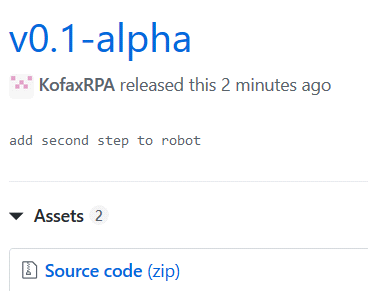
Here we will create a **read-only** project, still on the Non Production cluster and automatically deploy the robot there.

1. Create a User Group called **Testers** in ManagementConsole/Admin/Users&Groups/AddGroup  
   *You will need to log in as the admin user to create Groups.*
2. Add yourself to the **Testers** group.  
   
3. Create a new project called **GettingStarted\_Test** in Management Console/Admin/Projects/New.  
   *You will need to log in as an admin user. A member of Project Admins cannot create new projects.*
4. Add **Testers** with role **Viewer**, **Project Admins**, **Roboservers** and **Synchronizer** to the Project. **Developers** cannot upload robot changes to this project.  
   
5. Select **Non Production** for the Services.  
   
6. In the Repository settings add the SSH URL from GitHub, use the branch **test** and make sure Read-only is selected. *This project doesn’t allow anyone to change the robots, and it cannot upload to GitHub – this is exactly what you want in a test environment.*  
   
7. Go to your GitHub repository and select the **dev** branch so you see the **DoNothing.robot**.  
   
8. Click **New pull request** to pull the **dev** branch to the **test** branch.  
   
9. Select **test** as the destination and GitHub will quickly check if the merge is possible. (*The test branch has nothing newer than the dev branch, so there are no conflicts.*)  
   
10. Add a comment and select **Create pull request.**
11. Click **Merge pull request** and **Confirm merge**. *Don’t delete the* ***dev*** *branch, we can still use it.*
12. Goback to the Repository and select the **test** branch. You will see the robot in the test branch.  
      
    
13. In the Management Console, go to Repository/Robots and select the Project **GettingStarted\_Test.** There you will see the robot automatically deployed within 10 seconds.
14. Scroll to the right and compare the **Revision Number** with the **Commit ID** from GitHub, and you will see that they match – this is your guarantee of the robot version.  
     

*Your test environment is now ready for user testing. GitHub is tracking the correct version and the developers have no access to the test robots.*

# Create a Version Number for the Project

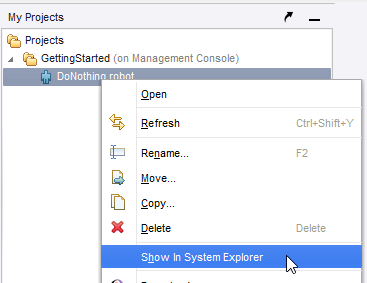
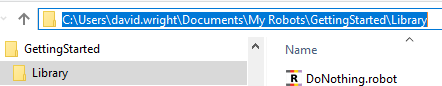
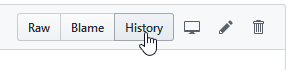
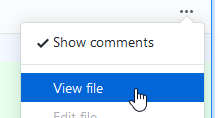
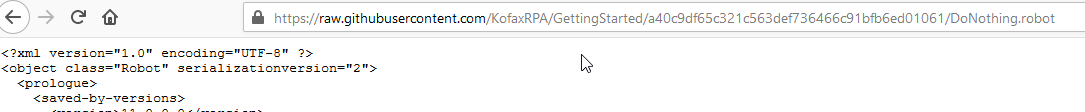
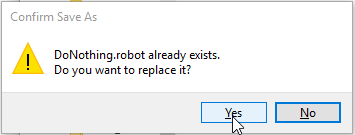
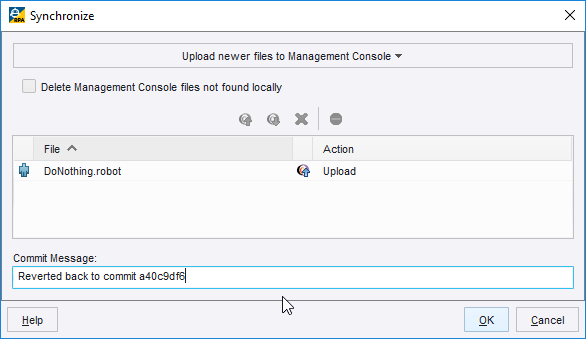
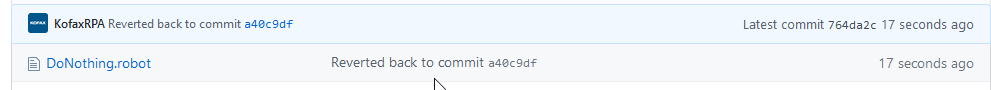
*GitHub uses* ***tags*** *to manage* ***releases****. After testing has been completed we want to merge the test robots with the master branch and tag the master with* ***a******version number***

1. In GitHub merge the **test** branch to the **master** branch with a pull request in the same way that you merged the **dev** branch to the **test** branch in Section 14.  
     
   
2. In GitHub make sure that you are looking at the master branch and click on **releases.**
3. Click **Create a new release.**
4. Give the release a name in the **Tag Version** field and click **Publish Release**  
   
5. You now have a clearly tagged release, and even a download package, that others can download and use. <https://github.com/KofaxRPA/GettingStarted/releases/tag/v0.1-alpha>   
   

# Revert to a previous version of a Robot

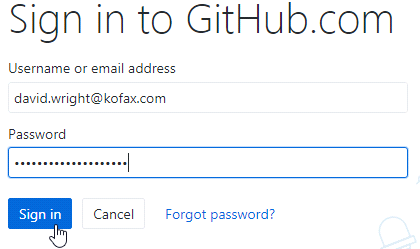
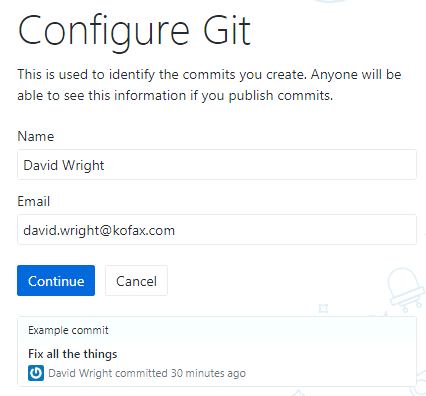
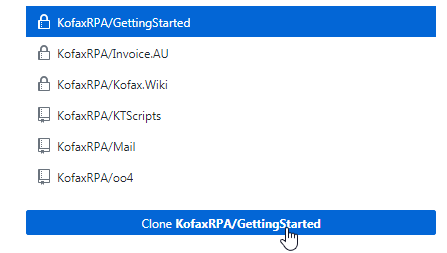
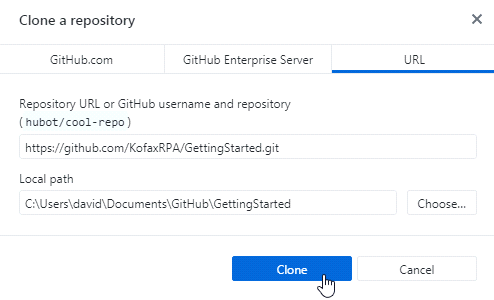
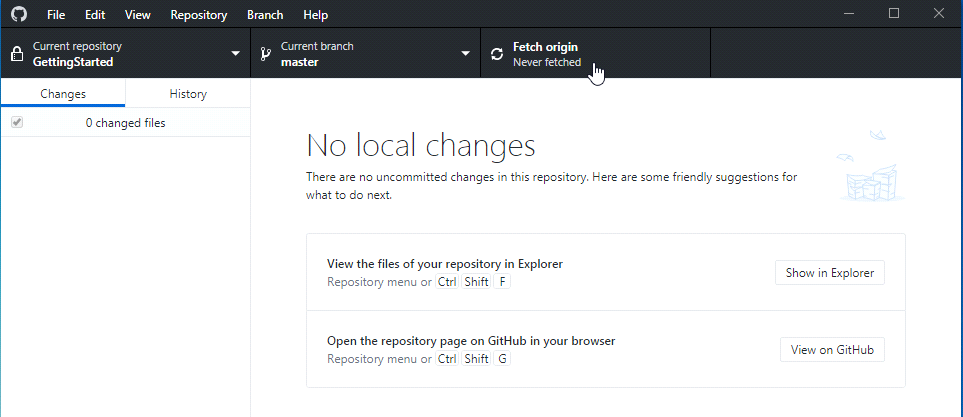
*You cannot revert to a previous version of a robot* ***directly*** *on GitHub’s website. The two main alternatives are to use the* ***command line*** *or to install* ***GitHub Desktop****. Neither are difficult, but are beyond the scope of this guide.*

*Here we will use a method to revert that does not use any other software. We will download an older version of the robot from GitHub to the Design Studio.*

1. Right-click on the robot in Design Studio and select **Show in System Explorer**  
   
2. Copy the Path to the clipboard. You will need this later when you download the older version of the robot from GitHub.  
   
3. Open your repository on GitHub and click on the robot.  
   
4. Select **History**.  
   
5. And pick a commit that want to revert to.  
   
6. In the top-right corner select **…/View file.**  
   
7. You are now looking at the commit of the initial version of the robot. Click on **Raw**.  
   
8. Here you see the initial version of the robot. Check the commit number in the URL – it starts with **a409c**, exactly as it did when it first appeared in GitHub.  
   
9. Download this robot from the Browser and replace the original robot  
   
10. Open the Robot in Design Studio and you are back where you started. You can now recommit this older version  
    
11. Go to GitHub and refresh the browser. It shows the older version of the commit with a new commit id. As GitHub Help states, *“When you revert to a previous commit, the revert is also a commit. The original commit also remains in the repository's history.”* [*https://help.github.com/en/desktop/contributing-to-projects/reverting-a-commit*](https://help.github.com/en/desktop/contributing-to-projects/reverting-a-commit)

*GitHub Desktop will create another copy of the repository on your harddisk and you will be able to revert that – and the changes will push all the way back to your robot. The diagrom below shows how the robot moves from* ***GitHub Desktop*** *to* ***RPA Design Studio***

**

1. Download, install and start **GitHub Desktop** from <https://desktop.github.com/>  
   
2. Start GitHub Desktop and click **Sign in to GitHub.com**  
   
3. Configure Git with your Name and Email address, so other users in GitHub know who you are!  
   
4. Click on your **GettingStarted** repository and **Clone** it.  
   *“cloning” a repository makes a local copy of the repository.*  
   
5. Select a location for your local clone  
   
6. Wait until it is finishing cloning and then click **fetch origin**.   
   ***Fetch*** *gets the files from GitHub and puts them into your local repository. Now your repository is identical to GitHub.*

1. A single Design Studio can connect to multiple Management Consoles, but each project in a Design Studio is linked to only one Management Console. [↑](#footnote-ref-1)
2. EachSynchronizer has a local bare Git repository for staging in the folder **%appdata%\..\local\Kofax RPA\$version$\Data\Synchronizer**. Do not edit any data here. The Synchronizer does not need to be run on the same machine as the Management Console. The Synchronizer [↑](#footnote-ref-2)
3. If you have multiple instances of Kofax RPA installed, the latest version of the Synchronizer should be found. When you run it, it will log the path to the Synchronizer, so you can check that it was the right version. You can also check that the config file was created 3 steps later. [↑](#footnote-ref-3)
4. The Synchronizer cannot be run as a Windows Service – it needs to be run as a normal application. If you are running the Synchronizer under Linux, you don’t have this problem. [↑](#footnote-ref-4)
5. See <https://nvie.com/posts/a-successful-git-branching-model/> for the original blog from 2010 where this model came from. [↑](#footnote-ref-5)
6. See <https://chris.beams.io/posts/git-commit/> for a good discussion on how to write commit messages [↑](#footnote-ref-6)