Kofax RPA
Best Practices Guide

Version: 11.4.0

Date: 2022-11-18

KOFAX

Table of Contents

Preface

This guide offers recommended methods and techniques to help you optimize performance and ensure success while using Kofax RPA.

Related Documentation

The documentation set for Kofax RPA is available here:[1]

https://docshield.kofax.com/Portal/Products/RPA/11.4.0-vcsft2fhaw/RPA.htm

In addition to this guide, the documentation set includes the following items:

Kofax RPA Release Notes
Contains late-breaking details and other information that is not available in your other Kofax RPA documentation.

Kofax RPA Technical Specifications
Contains information on supported operating systems and other system requirements.

Kofax RPA Installation Guide
Contains instructions on installing Kofax RPA and its components in a development environment.

Kofax RPA Upgrade Guide
Contains instructions on upgrading Kofax RPA and its components to a newer version.

Kofax RPA Administrator's Guide
Describes administrative and management tasks in Kofax RPA.

Kofax RPA Help
Describes how to use Kofax RPA. The Help is also available in PDF format and known as *Kofax RPA User's Guide*.

Kofax RPA Getting Started with Robot Building Guide
Provides a tutorial that walks you through the process of using Kofax RPA to build a robot.

---

[1] You must be connected to the Internet to access the full documentation set online. For access without an Internet connection, see the *Installation Guide*.

Kofax RPA Getting Started with Document Transformation Guide

Provides a tutorial that explains how to use Document Transformation functionality in a Kofax RPA environment, including OCR, extraction, field formatting, and validation.

Kofax RPA Desktop Automation Service Configuration Guide
Describes how to configure the Desktop Automation Service required to use Desktop Automation on a remote computer.

Kofax RPA Developer's Guide
Contains information on the API that is used to execute robots on RoboServer.

Kofax RPA Integration API documentation

Contains information about the Kofax RPA Java API and the Kofax RPA .NET API, which provide programmatic access to the Kofax RPA product. The Java API documentation is available from both the online and offline Kofax RPA documentation, while the .NET API documentation is available only offline.

> ℹ The Kofax RPA APIs include extensive references to RoboSuite, the original product name. The RoboSuite name is preserved in the APIs to ensure backward compatibility. In the context of the API documentation, the term RoboSuite has the same meaning as Kofax RPA.

Training

Kofax offers both classroom and computer-based training to help you make the most of your Kofax RPA solution. Visit the Kofax Education Portal at https://learn.kofax.com/ for details about the available training options and schedules.

Also, you can visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Getting help with Kofax products

The Kofax Knowledge Base repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base:

1. Go to the Kofax website home page and select **Support**.

2. When the Support page appears, select **Customer Support > Knowledge Base**.

ⓘ The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.

From the Knowledge Base home page, you can:

- Access the Kofax Community (for all customers).

  Click the **Community** link at the top of the page.
- Access the Kofax Customer Portal (for eligible customers).

  Click the **Support** link at the top of the page. When the Customer & Partner Portals Overview appears, click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).

  Click the **Support** link at the top of the page. When the Customer & Partner Portals Overview appears, click **Log in to the Partner Portal**.
- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Go to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Robot Lifecycle Management

The Robot Lifecycle Management feature enables you to control work objects of various types in a version control system such as Git. Using this functionality, you can compare and synchronize the state of objects between the Management Console and your repository with the help of the Kofax RPA Synchronizer included in your installation. You can synchronize the following objects: robots, types, snippets, resources, schedules, and OAuth.

These best practices are written with the assumption that you have working knowledge of the Git version control system.

Basic setup

Basic setup without Robot Lifecycle Management

The following diagram illustrates a basic recommended Kofax RPA setup without Robot Lifecycle Management. This setup consists of two Management Consoles: one for development and one for production.
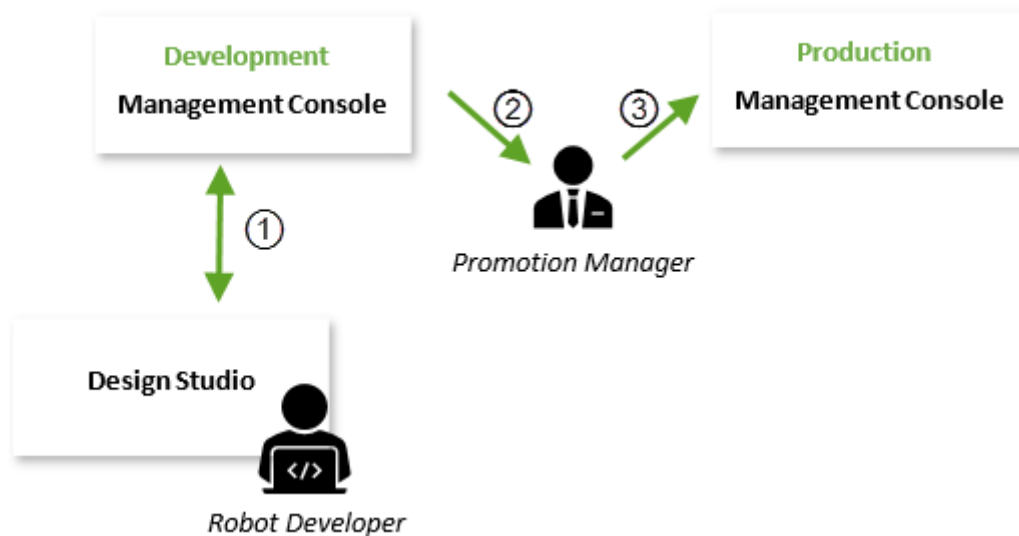
In this example setup, the Robot Developer establishes the synchronization **(1)** with a project stored in the development Management Console, creates new objects or makes changes to the existing objects, and then synchronizes the project again with that Management Console.

When the Promotion Manager considers a project for production, the manager makes a backup of that project **(2)** or particular objects from the development Management Console and then uploads the backup **(3)** to the production Management Console.

Without configuring Robot Lifecycle Management, this setup does not support the following features:

- Accurate version history for each object in a project.
- Ability to know which version is currently in production, to see the author of a change, the date an object was last changed, and a message explaining the changes.
- Ability to promote a specific version of an object from development to production.
- Ability to quickly revert to a previous object version in production.
- Ability to revert breaking changes in the development environment.

To include all of these features for use in your Kofax RPA environment, you can configure a basic setup with Robot Lifecycle Management as shown in the next topic.

Basic setup with Robot Lifecycle Management

The following diagram illustrates a basic recommended Kofax RPA setup with Robot Lifecycle Management. This setup shows two instances of Management Console that share a single repository: one instance for development and one instance for production.
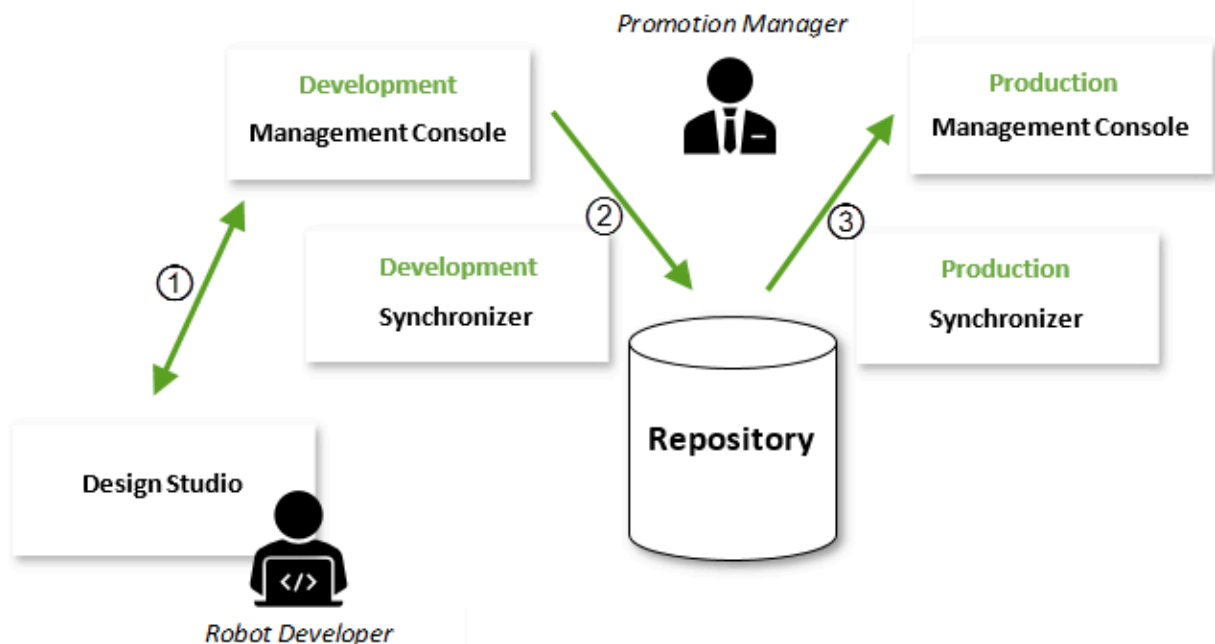
In this example, the Robot Developer establishes synchronization **(1)** with a project stored in the development Management Console.

Whenever a Robot Developer updates the project in the development Management Console, the changes are automatically synchronized **(2)** with the shared repository; they are pushed as a change commit on the specified branch. The responsibility of the Production Manager is to point to the branch to use in production.

At this point, two methods to promote the change are possible:
- Merge, re-base, cherry pick, or pull the changes to the branch specified in the production Management Console.
- Simply specify the production Management Console to the new version.

When the Promotion Manager approves the changes, the production Synchronizer takes the changes **(3)** from the repository and then pushes them to the production Management Console.

In the following topics, we will recreate this setup step-by-step.
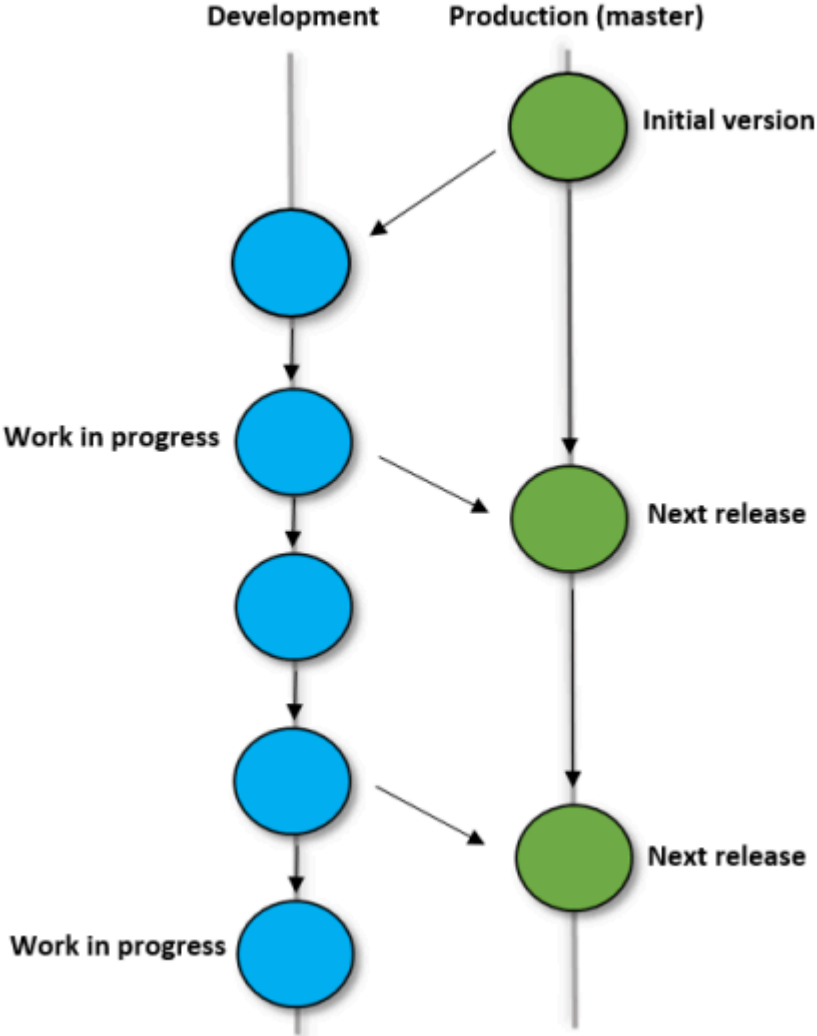
Basic setup with Robot Lifecycle Management
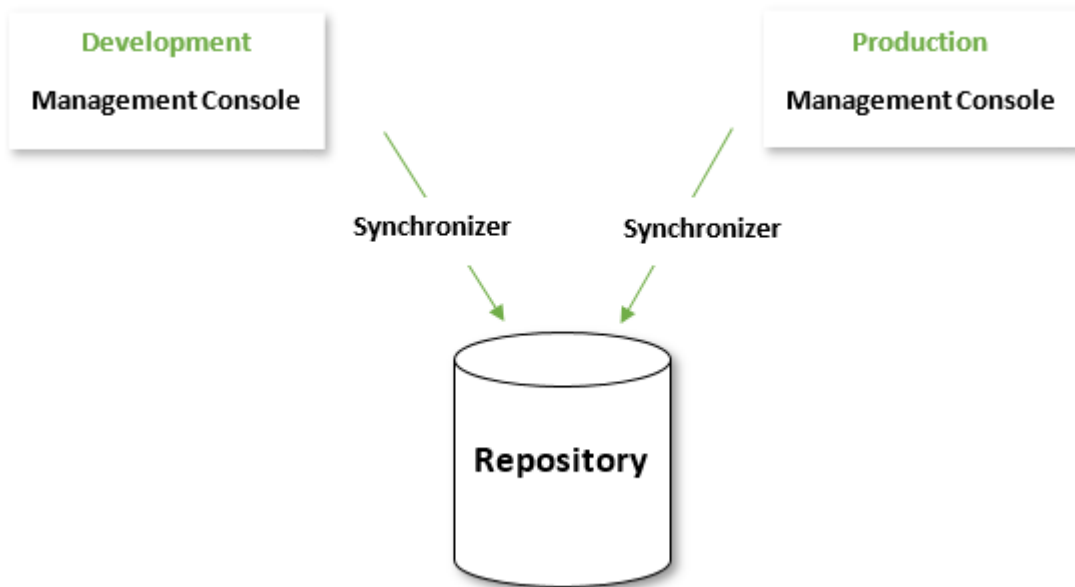
Select branching strategy

Many approaches are available to manage your work objects with a Git repository. We recommend that you follow the strategy and recommendations presented in the following topic.

Recommended branching strategy

According to this strategy, the head of the master branch always contains the current version running in production. You can add another branch to the master (production) branch, such as for development, and have the development Management Console synchronized with that branch.

The following figures demonstrate the minimal recommended branching strategy and setup. The blue and green circles are Git heads containing current versions in the development and production branches, respectively.

Development   Production (master)

Initial version

Work in progress

Next release

Next release

Work in progress

Recommendation 1: Always use the master branch for production.

Recommendation 2: When merging to the master branch, use the `--no-ff` flag, which prevents Git from executing in a "fast-forward" manner if it detects that your current head is an ancestor of the commit you are merging. It is useful to have the merge commits on your production branch to track the exact date and time of the merge.

ℹ️ To learn more about the strategy described above, see A successful Git branching model.

Other branching strategies

If you prefer a more complex setup, the minimal branching strategy shown in the previous topic can be easily expanded to cover a larger setup with three branches: development, testing, and production.
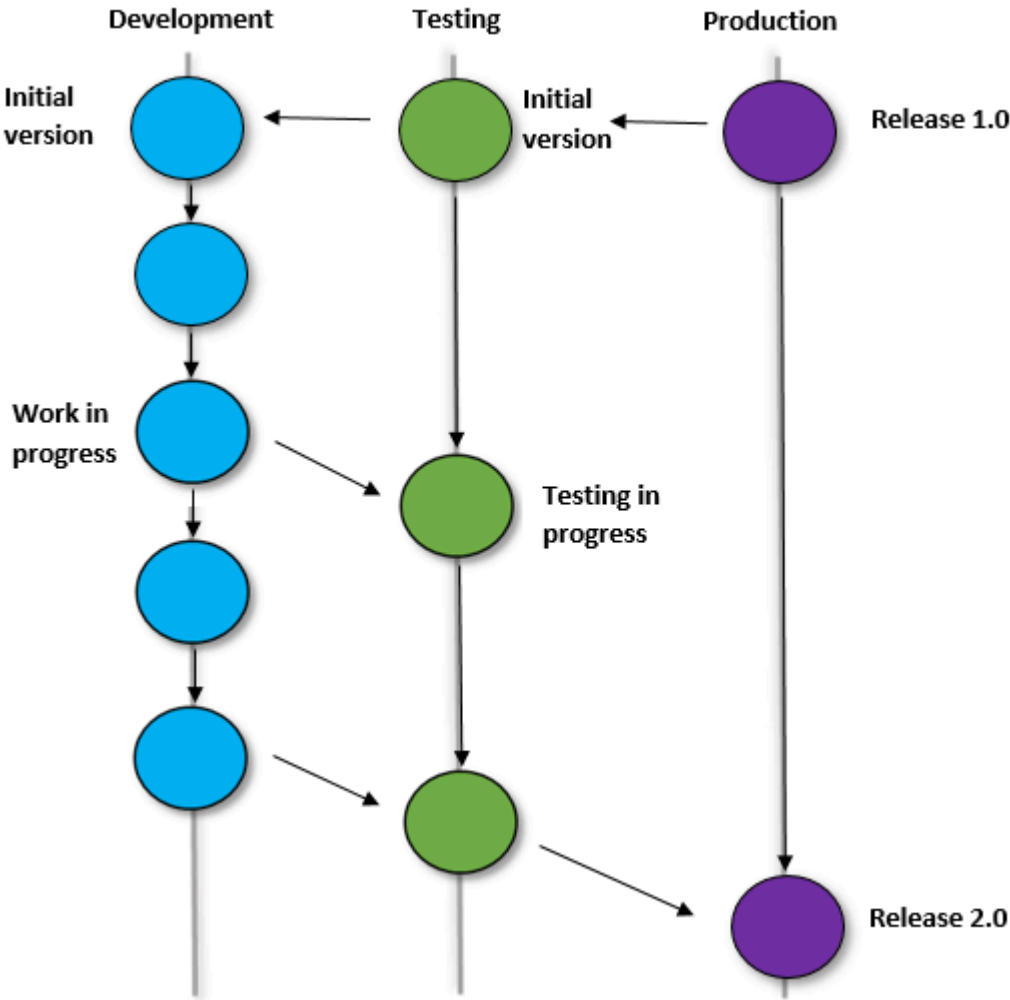
In this setup, you have three Management Consoles for development, testing, and production, synchronized with three branches in a Git repository through Synchronizers.
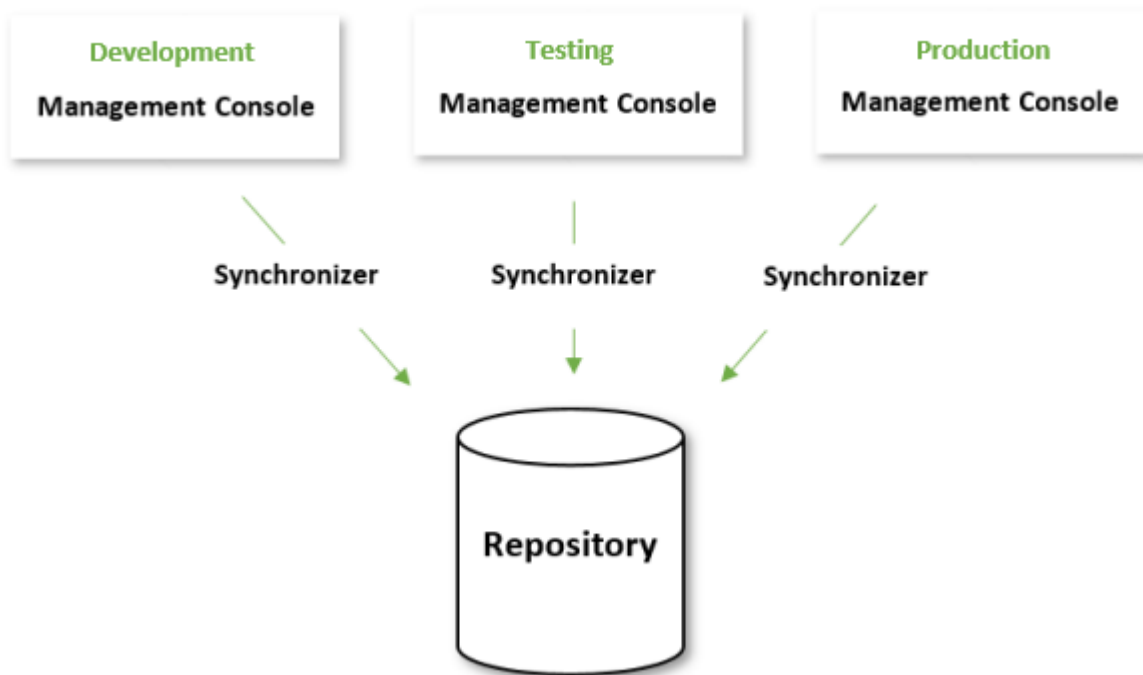
Create bare repository

Before starting the Synchronizers and setting up the Management Consoles for synchronization, you need to install Git, and open Git Bash, then initialize your Git repository. The method for creating a new bare repository depends on the third-party tool that you are using.

To create a bare repository and a development branch, you can execute the following commands.

```
# mkdir example.git
# cd example.git/
# git init --bare
Initialized empty Git repository in /gitrepos/example.git/
# cd ..
# git clone example.git
Cloning into 'example'...
warning: You appear to have cloned an empty repository.
done.
#cd example/
# git commit --allow-empty -m 'initial commit'
[master (root-commit) b96fdb8] initial commit
# git push origin
Everything up-to-date
# git checkout -b development
Switched to a new branch 'development'
# git push -u origin development:development
```

> ℹ️ Alternatively, you can use the Synchronizer to automatically create a bare repository. When you start the Synchronizer, it automatically creates a bare repository on the specified location if it was not already created, with an empty initial commit. However, the Synchronizer does not create a branch if it is not already present.

Set up Management Consoles

After creating a repository, you need to set up two Management Consoles: one for development and the other for production. They are used to synchronize with the development and production (master) branches in your file-based repository.

Set up development Management Console

1. Start the development Management Console.

2. On the menu, select **Admin** > **Projects**, click the ⋮ content menu for the project to synchronize with the repository, and then click **Edit**.

3. In the new dialog box, select the **Repository** tab.

   a. In the **URL** property, type the path to the repository that you created in the previous topic: **/gitrepos/example.git/**

   b. In the **Branch** property, type the branch to use: **development**.

   c. To enable the configuration specified above, select **Enable configuration**.

   d. Under **Objects to synchronize**, select the objects to include in the synchronization: **Robots, types, and snippets**.

   ☑ Enable configuration

   ┌ URL ──────────────────
   │ /gitrepos/example.git
   └───────────────────────

   ┌ Branch ───────────────
   │ development
   └───────────────────────

   ☐ Read-only

   Objects to synchronize

   ☐ Schedules
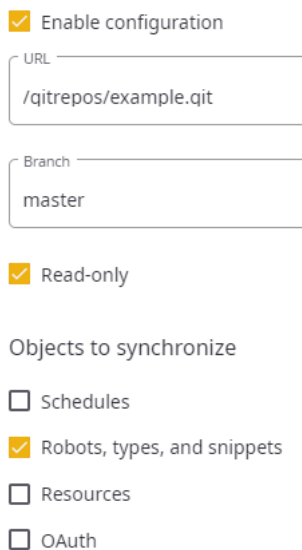   ☑ Robots, types, and snippets
   ☐ Resources
   ☐ OAuth

   e. Save the changes.

Set up production Management Console

1. Start the production Management Console.

2. On the menu, select **Admin** > **Projects**, click the ⋮ content menu for the project to synchronize with the repository, and then click **Edit**.

3. In the new dialog box, select the **Repository** tab.

    a. In the **URL** property, type the path to the repository that you created in in the previous topic: **/gitrepos/example.git/**

    b. In the **Branch** property, type the branch to use: **master**. We recommend that you always use the master branch for production.

    c. To make the repository the only source for object changes, select **Read-only**.

        We recommend that you select this option to avoid any changes to objects belonging to the synchronized project in the production Management Console.

    d. To enable the configuration specified above, select **Enable configuration**.

    e. Under **Objects to synchronize**, select the objects to include in the synchronization: **Robots, types, and snippets**.

    ☑ Enable configuration

    URL
    /gitrepos/example.git

    Branch
    master

    ☑ Read-only

    Objects to synchronize

    ☐ Schedules

   ☑ Robots, types, and snippets

    ☐ Resources

    ☐ OAuth

    f. Save the changes.

You have now set up your Management Consoles and are ready to start the synchronization.

Start synchronization

When using a file-based Git repository, both instances of the Synchronizer, one for the development Management Console and the other for the production Management Console, require direct access to the file system where the repository is located.

We recommend that you run both instances of the Synchronizer on the same computer where the repository is stored **(1)**. With this setup, the Promotion Manager can use one of the supported Git protocol standards **(2)** to access the repository and push changes to production.
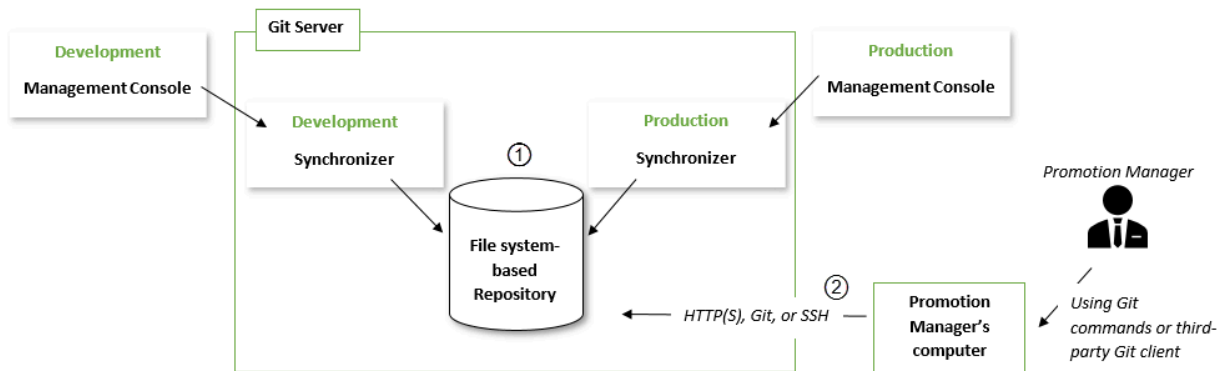
Recommendation 1: When synchronizing with a bare file-based repository, multiple Synchronizers can run on the same computer.

Recommendation 2: For convenience, the Promotion Manager can use a third-party Git client such as SourceTree instead of Git commands.

Recommendation 3: If Docker is used for deployment, the Synchronizer containers can share a volume.

Start the Synchronizers

Configure Synchronizer access to Management Console

To connect the two Management Consoles and the Git repository, follow this procedure.

1. In the **synchronizer.settings** file, point the development Synchronizer to the development Management Console and configure settings by using the commands listed in the following table.

    a. In the Command Prompt window, specify the `-c` property for the **synchronizer.exe** file to ignore the default `synchronizer.settings` file.

    b. Specify the other settings that you require.

    Example:

    ```
    Synchronizer.exe -c --mc-url http://127.0.0.1:8080/ManagementConsole --oauth-
    client-secret <insert-secret-string> --interval 10 --no-host-key false --private-
    key $USER_HOME \.ssh\id_rsa -s
    ```

Command	Description
`-c,--command-line`	Uses settings specified in the command-line and ignores the settings file.
`-e,--environment`	Uses settings from the environment and ignores the settings file.

Command	Description
`-g,--generate-ssh-keys <argument>`	Generates a key-pair for SSH authentication, and saves it to the specified folder. For example: `-g C:\Work\MyKeys`
`--mc_url <argument>`	*Required*. Use only when `-c` is specified. Specifies the URL to connect to the development or production Management Console containing the protocol and port number.
`--oauth-client-secret <argument>`	*Required*. Use only when `-c` is specified. Contains the plain text OAuth client secret copied from the OAuth server section of the Management Console to authenticate Synchronizer with the Management Console.
`--oauth-client-secret-file <argument>`	*Required*. Path to a file containing the OAuth client secret copied from the OAuth server section of the Management Console to authenticate Synchronizer with the Management Console.
`--interval <argument>`	*Required*. Use only when `-c` is specified. Sets the interval in seconds between synchronization runs. If set to a value equal to or less than `0` or a non-numeric value, the Synchronizer runs once and exits.
`--no-host-key <argument>`	*Optional*. Use only when `-c` is specified. Disables strict SSH host-key checking. Default is `false`.
`--private-key <argument>`	*Required*. Use only when `-c` is specified. Provides the path to the file containing a private SSH key to connect to a remote repository. When connecting to a local repository, this attribute is ignored, but a value must be specified.
`-r,--reset-hard`	Resets the version information and purges the entire local cache.
`-s,--save`	Saves the configuration settings in the `synchronizer.settings` file and exits.
`-v,--version`	Prints version information and exits.
`-h,--help`	Prints the description of properties and exits.

2. Repeat the previous steps for the production Synchronizer and the production Management Console, changing the properties as needed.

3. Save the configuration settings.

   The Management Consoles and the Git repository are now synchronized. Now you can promote objects to production and set up synchronization between the Management Console project and the GIT repository.

Set up synchronization between the Management Console project and github repository

To set up synchronization between the Management Console project and github repository, follow these steps.

1. Open the github repository website and create a new account.

2. In the Command Prompt window, run the `Synchronizer.exe` file with the `-g` parameter specified.
   Example

```
Synchronizer.exe -c --mc-url http://127.0.0.1:8080/ManagementConsole --oauth-
client-secret <insert-secret-string> -g $USER_HOME\.ssh\
```

This generates the RSA pair of public and private keys.

3. Log in to your github account. In the Settings menu, find the SSH and GPG keys section, and create a new SSH key by entering the public key from the generated key-pair.

4. Create a new private repository on the github website and copy the SSH address of your repository, such as `git@githubrepository.com/username/repository_name.git`.

5. Go to **Management Console** > **Admin** > **Projects** and select a project.

   Click **Edit** and in the **Edit project** window, select **Repository**.

   a. Click **Enable configuration** to activate the fields for configuring a repository.

   b. In the **URL** field, provide the SSH address you copied.

   c. In the **Branch** field, type the name of the branch to use.

   d. If **Read-only** is selected, the project files will be wiped out and replaced with the github repository files.

      If **Read-only** is not selected, the github repository files will be wiped out and replaced with the project files.

      In the read-only mode, all the objects that the Management Console contains, are taken from the repository. Before starting synchronization, make sure that the objects you want to synchronize are empty in the Management Console. Otherwise, an error message appears.

      To delete the object contents, select the required object from the **Objects to synchronize** list and click **Delete selected objects**. Use caution when deleting the objects so that you do not loose important data.

   e. For the **Objects to synchronize**, select the objects to include in the synchronization.

   f. Click **OK** to save the changes.

6. In the Command Prompt window, run the `Synchronizer.exe` file with the `--private-key` parameter specified.

   Example

```
Synchronizer.exe -c --mc-url http://127.0.0.1:8080/ManagementConsole --oauth-
client-secret <insert-secret-string> --interval 10 --no-host-key true --private-
key $USER_HOME\.ssh\id_rsa
```

You have now synchronized your github repository with your Management Console project.

Promote and revert changes

Robots, types and snippets can now be synchronized with the development Management Console from Design Studio using the upload function or from the Management Console interface using the Robots tab. At this point, you can make changes to the production Management Console only by promoting an object version from development to production.

Use the following command to merge from the development branch into the production (master) branch.

```
# cd example
```

```
# git checkout development
# git pull
# git checkout master
# git merge --no-ff development
# git push
```

If you need to revert changes, do so on the development branch by reverting commits to a previous good state and then merging the reverted commits to the production (master) branch.

Check synchronization result

To check the synchronization result, see the respective log file, which by default resides in: `home \AppData\Local\Kofax RPA\version\Logs`. The log file location can be configured with log4j2_synchronizer.properties, which resides in: `home\AppData\Local\Kofax RPA\version \Configuration`.

If synchronization is successful, the log contains the name of the synchronized project and a commit ID of the synchronization. The synchronization is run at the specified interval, and object changes are automatically synchronized between the Management Console and the repository once the changes appear.

If conflicting changes occur, the repository object version has a priority over changes in the Management Console. In this case, the conflicting Management Console changes are ignored.

> ⓘ Currently, it is not supported to rename and change the location of synchronized files from the repository because it references to these files in the Management Console become incorrect.

A successful synchronization includes:
- commit ID. Denotes a commit of the repository object changes synchronized with the Management Console.
- timestamp. All objects modified in the Management Console before this point in time are synchronized with the repository.

On the Management Console interface, to see the author of a newly added object, commit message, object revision, and date of the last modification, ensure that the following columns are added to the respective view: **Modified by**, **Commit message**, **Revision number**, and **Last modified**, respectively.

Access rights and prerequisites

By default, only users with the **admin**, **Administrator**, **Project Administrator**, or **VCS Service User** role can use Robot Lifecycle Management.

While all of these roles include the right to start the Synchronizer, **VCS Service User** is designed specifically for this purpose. The user with this role cannot make changes to a Management Console such as setting synchronization settings for a project. Therefore, you may assign this role to a user whose responsibilities do not include project management and primarily relate to synchronization with a version control system.

To allow a new user access to Robot Lifecycle Management, in the Management Console, the user must be added to a group that has one of these roles. For more information on user roles, see "Manage Users and Groups" in *Kofax RPA Help*.

If the Synchronizer operates on a file system that has specific non-standard limitations on characters and, therefore, may apply some restrictions on object names, it may cause issues during the synchronization of those objects, even if the file system itself is supported. To ensure successful synchronization, check that your file system allows characters used in the synchronized object names.

Chapter 2

Desktop Automation Service setup

The Desktop Automation Service is required if the Robot 🤖 needs to automate Windows desktop applications on a remote computer. See the *Kofax RPA Desktop Automation Service Configuration Guide* for more information.

You can have several Windows desktops available to run the applications. These desktops report to Management Console and thus are being administered. The Robot interacts with Management Console to get an available and suitable DAS, because the Robot requires specific conditions or features for its execution. To match the Robot with the right DAS, use labels.

The sections below describe several best practices for using DAS:
- Use labels for multiple DAS setup
- Automate Desktop Automation Service with configuration file

Use labels for multiple DAS setup

In your RPA setup, you may have multiple instances of the Desktop Automation Service (DAS) installed on different computers.

To connect multiple DAS instances to Design Studio simultaneously, you need to create separate devices for them. After that, ensure they are listed under **Required Devices** on the Action tab of the Call Robot step in the Basic Engine Robot 🤖.

When the DAS is installed on a desktop or a virtual machine, such a computer does not usually have an extensive set of software applications installed. Adding labels helps identify which software is installed on each computer and can be used by the Robot 🤖.

Later, when the Robot 🤖 needs to automate a particular business application installed on the computer, the system finds an available DAS that provides all labels required by the Robot 🤖.
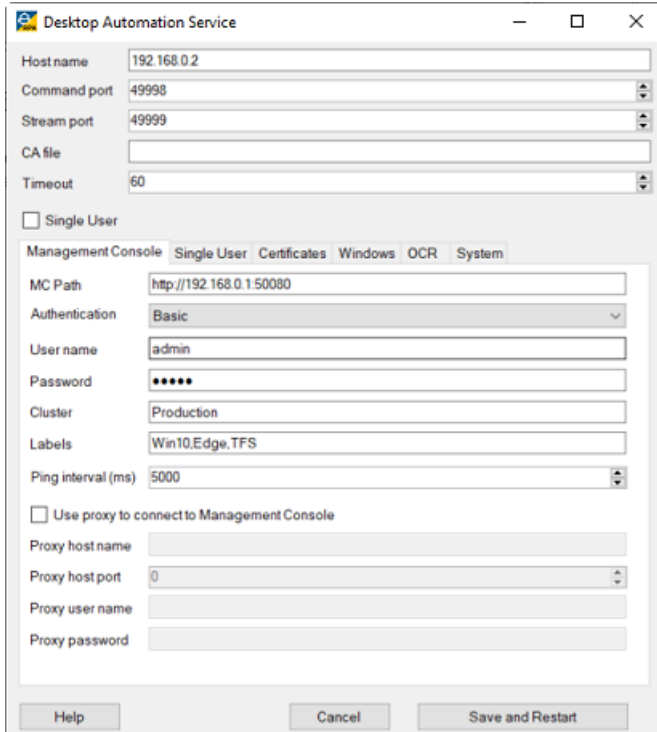
Possible scenarios include:
- Multiple setups are used with different DAS versions, while the business application to be automated can only work with a specific DAS version.
- Different desktops may be set up with different platform versions (Microsoft Windows), which may impact the Robot 🤖 configuration.

Follow the procedure below to assign labels.

1. Start the Desktop Automation Service from the Start menu. Once the service starts, you can see its status by looking at the icon in the notification area.

2. Right-click the Desktop Automation Service ⊜ icon in the notification area and click **Configure**. This action opens the Desktop Automation Service window.

3. On the **Management Console** tab, in the **Labels** field, type the labels. In the example below, the DAS is installed on a computer that has Microsoft Windows 10, the Microsoft Edge browser, and the Team Foundation Server (TFS) application installed. The labels must be separated by commas and must not contain any spaces between the comma and the next label.

   Within your organization, you can create any labels, without any limitations.

4. Click **Save and Restart**.

5. Check that the device is registered in the Management Console under **Admin** > **Devices**.

6. Launch Design Studio.

7. Right-click the required project in the **Projects** list and select **New** > **Automation Device Mapping**.
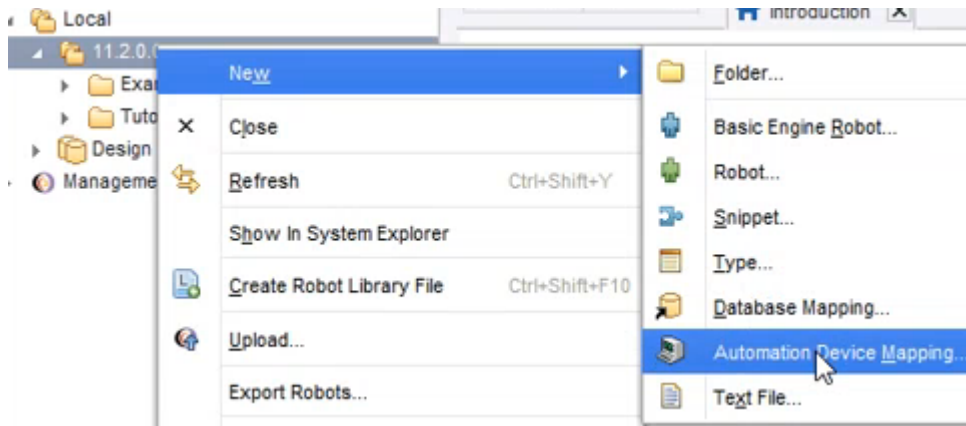
8. In the **New Device Mapping** screen, name the new mapping and click **Next**.

9. Ensure that **Management Console Based Device Mapping** is selected and type the required labels. In the example below, the Robot 🤖 is going to automate the TFS application on a computer with Microsoft Windows 10 installed.
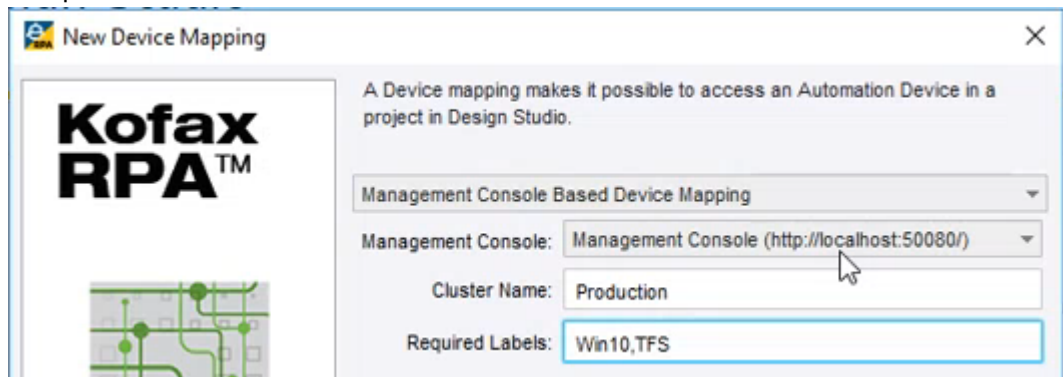
10. Finish the configuration and map the Robot 🤖 to this device.

ⓘ To execute the Robot on two computers with the DAS, create two device mappings with different labels in **Management Console**.

ⓘ We recommend to use labels for platform names and languages for the cases when the Robot needs to interact directly with the Microsoft Windows user interface or localized applications. It allows the Robot to select a properly configured DAS instance and facilitates a managed migration to other platforms.

Automate Desktop Automation Service with configuration file

If you are going to automate the deployment of the DAS, use the original configuration file that comes with the DAS installation. Do not use files from other product versions.

Open the `server.conf` file on your automation desktop. The file is located in the **C:** > **Users** > **UserName** > **AppData** > **Local** > **Kofax RPA 11.4.0** folder, where **UserName** is the name of the user the service is running under.

The following parameters must be changed in the configuration file to match the DAS setup:
- Host name
- MC Path
- User name
- Password
- (Optional) Labels

Additional configuration may be necessary to configure network security or other features.