exalate

JIRA GITHUB INTEGRATION

STEP-BY-STEP GUIDE

Businesses use different work management systems and issue trackers to keep track of their tasks. If you are using different platforms to manage your projects or you need to get synced with another team, integrating them can bring many benefits. Let's say you're working in Jira and need to integrate with another company using GitHub, then a Jira GitHub integration is exactly what you need.

Jira and GitHub are both excellent tools for managing projects, but they have different feature sets and their own strengths and weaknesses. Connecting them will enable teams to work in harmony and optimize their workflows without tampering with each other's autonomy or security.

By integrating these platforms, you can share items and have different teams using a common set of data to solve the problems they specialize in.

To make your integration as effective as possible, you need to know what the advantages of setting one up are. You also need to know about the potential issues involved. After that, you should choose the right integration solution, and then implement your integration.

In this guide, we'll take you through the process of working out what you need from your integration, and then show you how to connect the platforms in a step-by-step guide.

TABLE OF CONTENTS

CHAPTER 1

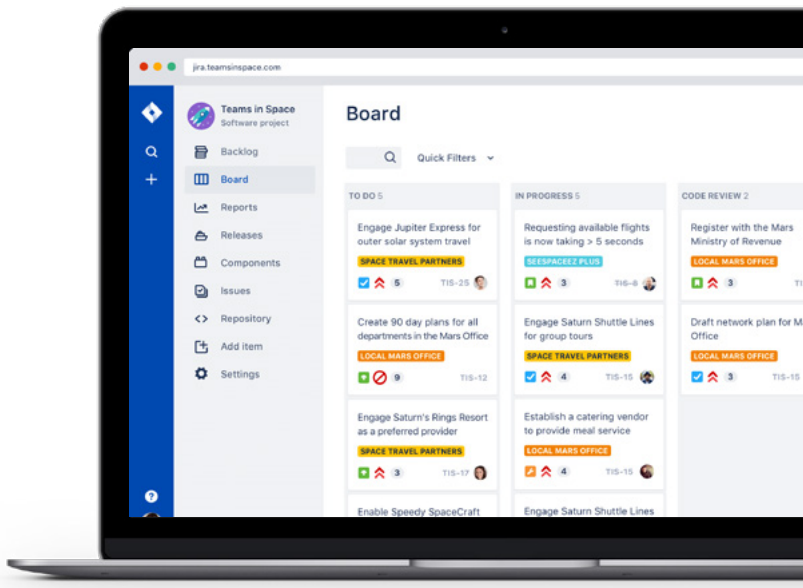
WHAT ARE THE BENEFITS OF A JIRA GITHUB INTEGRATION?

Connecting your platforms lets you share information and saves your teams from doing duplicate work. An integration that filters and sorts the shared data can present it to each team in a form most useful to them. Data they need is shared, but fields they don't need can be removed from their copy. That means they can function autonomously and more effectively.

Teams using GitHub and Jira will likely make use of whichever features are the best fit for their needs. Using them both together means your teams will be able to take advantage of all the positive features, while mitigating the drawbacks either platform may have

They are both widely used in software development, so those of you working in that field will likely be familiar with them.

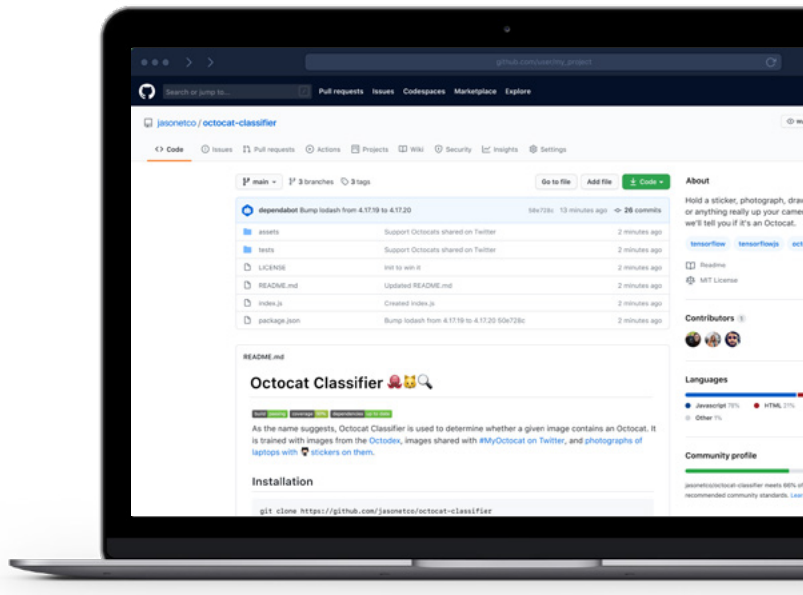
What is Jira?



Jira is an issue tracking system that also includes several other features. It is particularly suited for agile development and is frequently used by developers to manage their projects. It is available in several versions. There's a cloud-hosted version, but you can also self-host if you prefer.

Jira can be heavily customized and has a range of add-ons to give it extra functionality. You'll take advantage of this later when you set up your integration.

What is GitHub



GitHub is mostly used for storing code. Developers can use it to share a codebase among team members while tracking changes, allowing individuals to create their own branches to work on projects safely. It also tracks issues and comments, making it useful for managing projects.

Many development tools and project management platforms are designed to work with GitHub. It is hugely popular with developers throughout the world.

CHAPTER 2

CHOOSING THE RIGHT TECHNOLOGY FOR A JIRA GITHUB INTEGRATION

To get the most out of your integration you need to consider the potential issues that could arise with it. Problems are always easier to handle if you think about them in advance. With that in mind, choosing a platform that can handle these issues effectively will make life easier for your teams and allow them to focus on what they do best.

1. Decentralized Integration

The items you share in Jira and GitHub consist of many details. There are multiple fields containing different types of data. If you look at incoming items from the other team, you will find that some are useful, but others aren't. You need a system that can give you fine control over what is shared and allow you to pick exactly what fields are stored.

You don't necessarily want to have a meeting every time you need to make a change in the system. Being able to change things independently without having to involve the other team and having control over what is shared and what is not will make things faster and easier. Decentralized integration for each end of the system is a clear goal.

2. Flexibility

You might also want to customize fields and give them specific values as well as assign specific kinds of items to specific people. Being able to identify and filter incoming data will help you here.

Each team may want to make changes to the way they use information as their needs evolve and their understanding of how they can use the synchronization deepens. You may want to add or remove data provided to the other team.

In addition to controlling what is shared, you also need to control the conditions that trigger information exchange. Being able to create and combine rules to do this can give you the flexibility you need to make your integration effective.

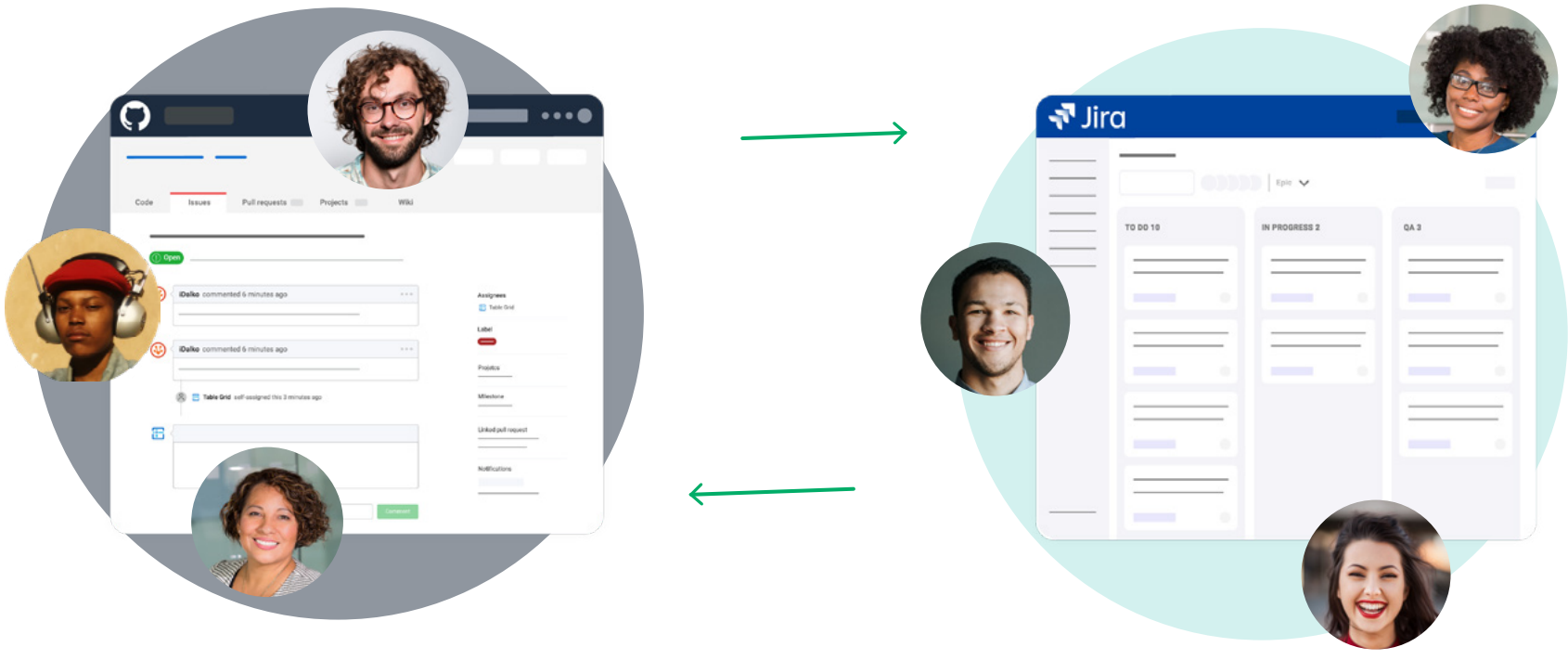
3. Reliability

In an ideal world, your integration would work perfectly when set up without any further intervention from you. In reality, systems usually suffer downtime. Connections are dropped, servers need maintenance and software needs updates.

When this happens on one end of your integration, you need to be confident the whole system won't break down. It needs to be able to handle outages to either side and recover gracefully when things are working again.

Your system should be able to handle whatever the world throws at it, and get up, brush itself down, and continue to work as before.

I've chosen Exalate to set up the Jira GitHub integration as it is designed with these three specific features in mind. It promises to deliver autonomy, flexibility, and reliability and is simple to set up and configure.



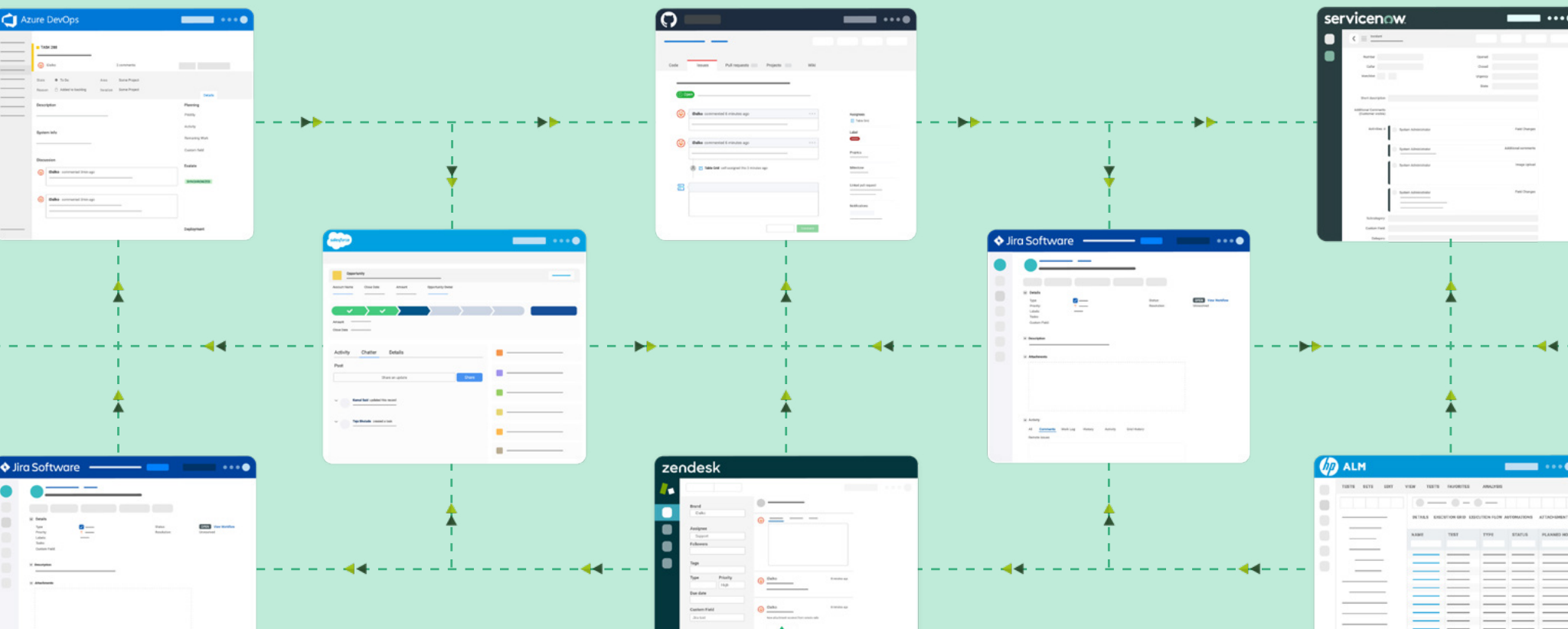
exalate

Bi-directionally connect your work across multiple work management systems in real-time

- A frictionless collaboration across **internal teams** and **outside company borders**
- Complete **autonomous control** over both outgoing and incoming information
- **Maximum security** due to complete control over the data being shared with the other systems
- **Limitless customizability** to fit your unique and complex integration needs
- Set up a synchronization between multiple work management systems. Available for Jira, Salesforce, Azure DevOps, ServiceNow, Zendesk, GitHub, HP ALM & more

BOOK A DEMO

TRY IT FOR FREE



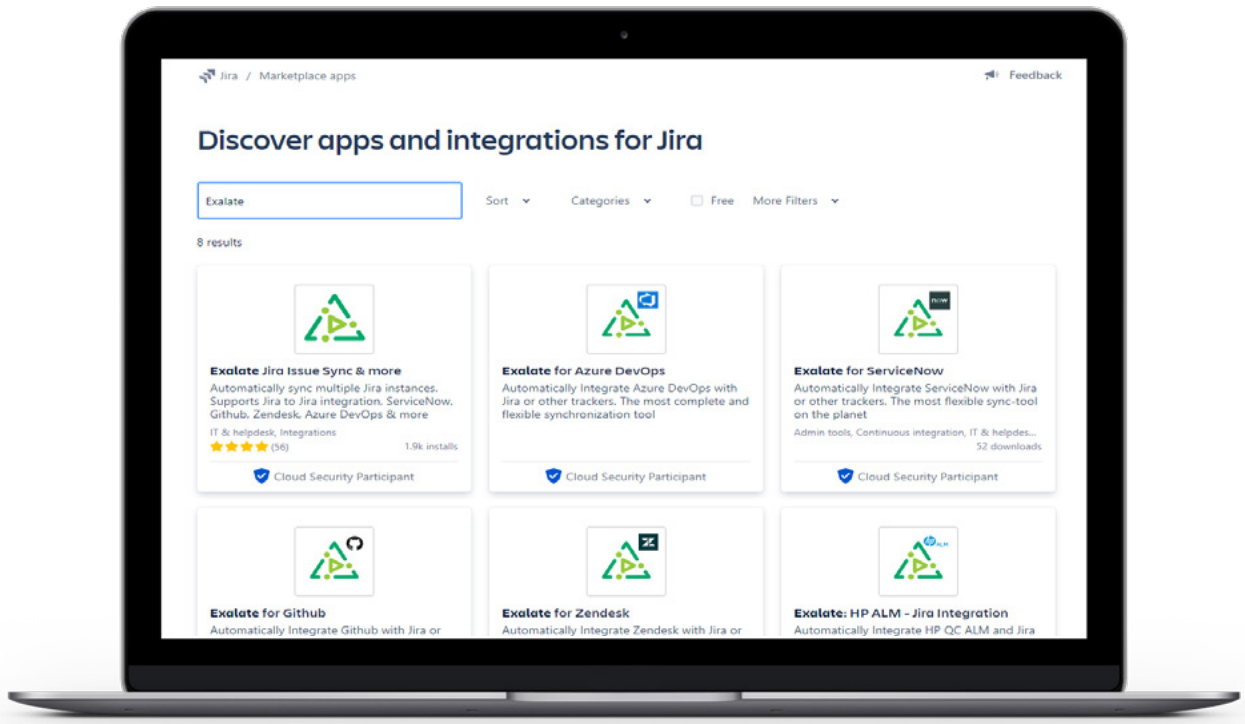
CHAPTER 3

HOW TO SET UP A JIRA GITHUB INTEGRATION IN SIX STEPS

To set up your Jira GitHub integration, you need to install Exalate on both platforms, then create a connection between them. After that you can configure the connection to meet your integration needs, controlling what is exchanged, and setting the conditions under which exchange takes place.

We also have this and other tutorials in a video format

CHECK OUR INTEGRATION VIDEOS →



1. Install Exalate on Jira

Log in to your Jira instance and find the Exalate app in the Atlassian marketplace. To do that, click the cog in the top right, and select “Apps” from the menu.

If you aren’t taken straight to the marketplace, click “Find new apps” in the left-side menu, and type “Exalate” into the search field. You should see a list of Exalate add-ons appear with “Exalate Jira Issue Sync & more” at the top.

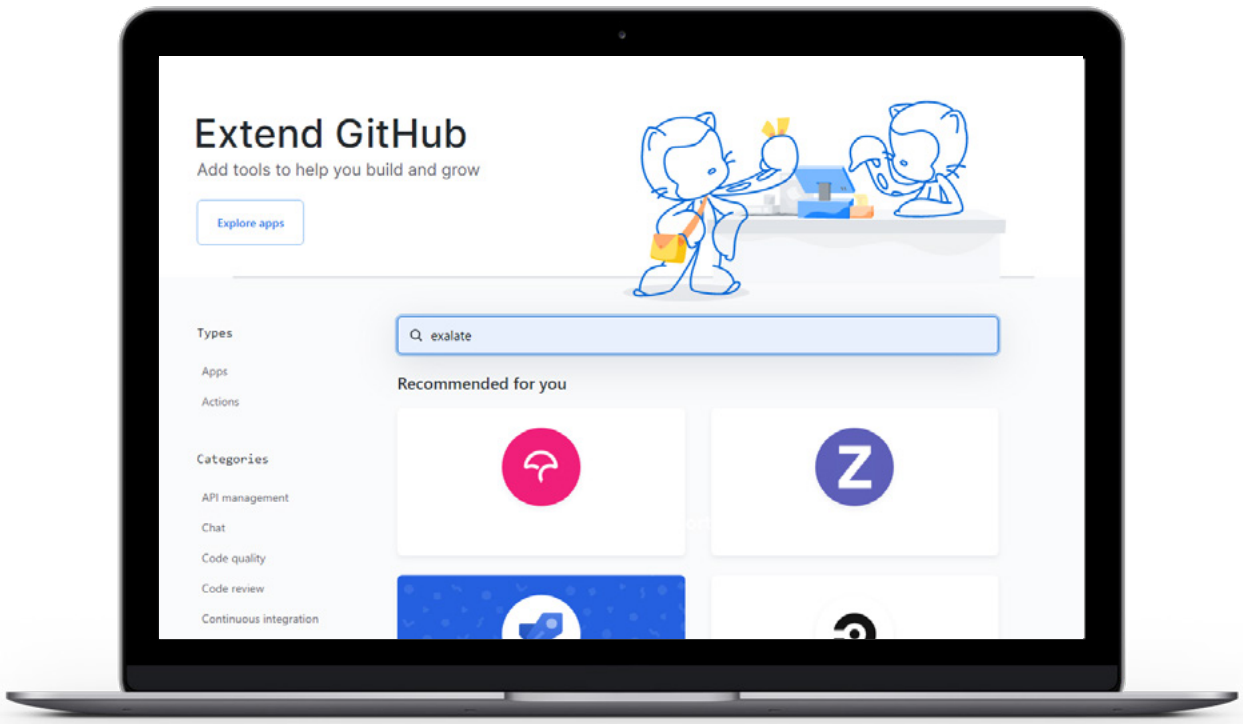
Click that, and choose “Try it free”. A confirmation screen will appear. Click the “Start free trial” button.

After a brief pause, you’ll see a popup appear saying Exalate has been installed successfully. Click “Get Started”. You’ll see the initiate connection button highlighted, but we’re going to leave Jira for the time being. We’ll return later on.

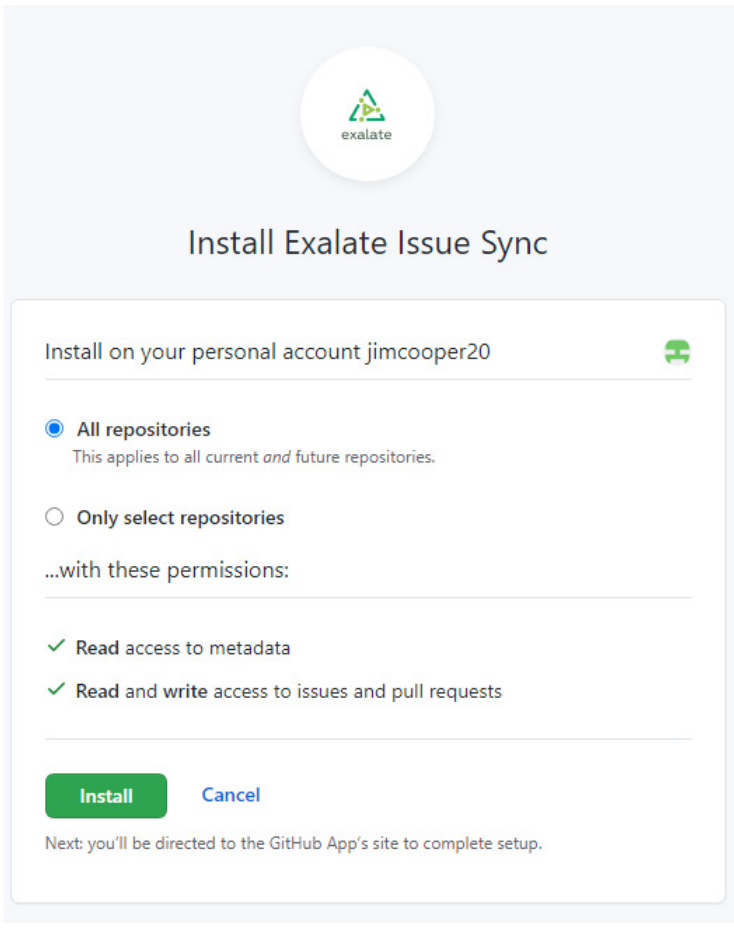
This tutorial uses Jira Cloud, which you can read more about here. You can also check out the Exalate documentation for Jira Server.

2. Install Exalate on GitHub

In GitHub, click “Marketplace”, then type “Exalate” into the search field that appears.



“Exalate Issue Sync” should appear, so click it to continue. Click on the “Set up a plan” button. Choose “Install it for free”, and Exalate will be available on your GitHub account.



You need to choose which of your repositories to grant Exalate access to. You can allow it to access all of them, or pick specific ones if you prefer. When you install Exalate, it gets access to the data it needs to exchange over your connection. Code access isn’t necessary, but Exalate needs access to metadata, issues, and pull requests.

Give it access to these and then click the “Install” button. You’ll need to either set up an OAuth token or allow Exalate to use your username and password to access GitHub.

Finally, you need to get an evaluation license for GitHub. Exalate is a paid app but is free to try, so there's no need to pay for now. Click "License Details" in Exalate's left-side menu. Then click the area at the top that says "30-Day Trial".

Enter your email in the popup that appears. When the license email arrives in your inbox, copy the evaluation key to your clipboard

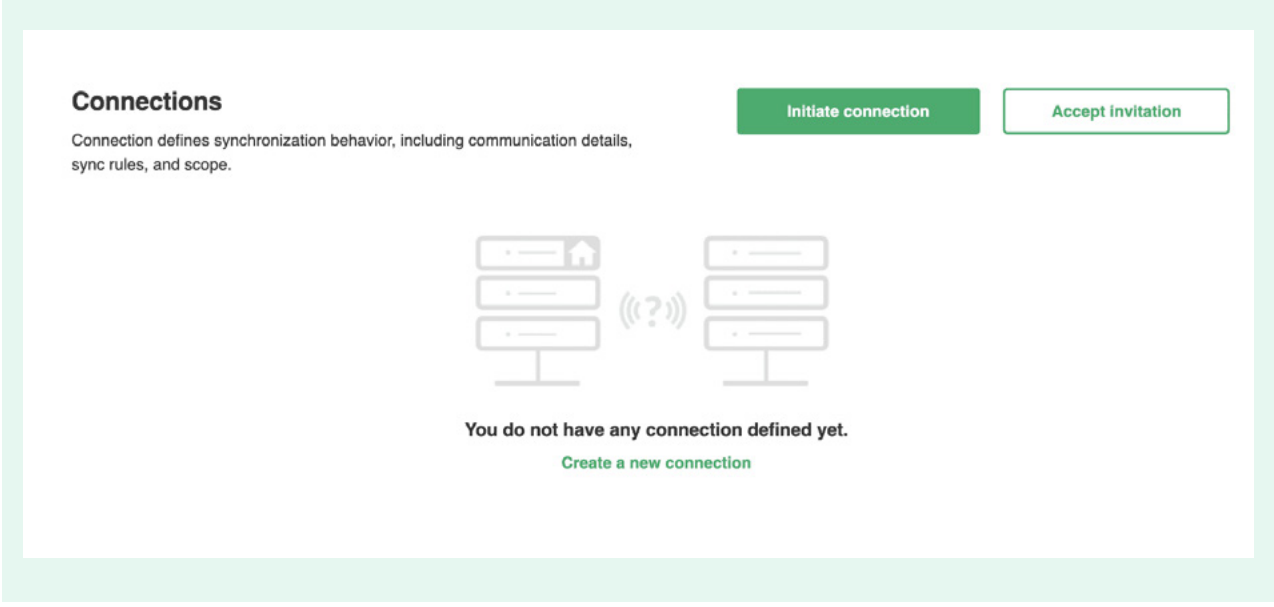
Back in GitHub, click the green "License Key" button at the bottom left of the "License Details" screen. Paste your license key from the email into the field that appears. Then click the "Update" button. Your license will now be installed and Exalate is ready to be used on GitHub.

3. Connect Your Jira and GitHub Instances

Exalate is now installed on both Jira and GitHub, but your instances are not yet connected. To create a connection, you need to “Initiate connection” from one side and “Accept invitation” on the other.

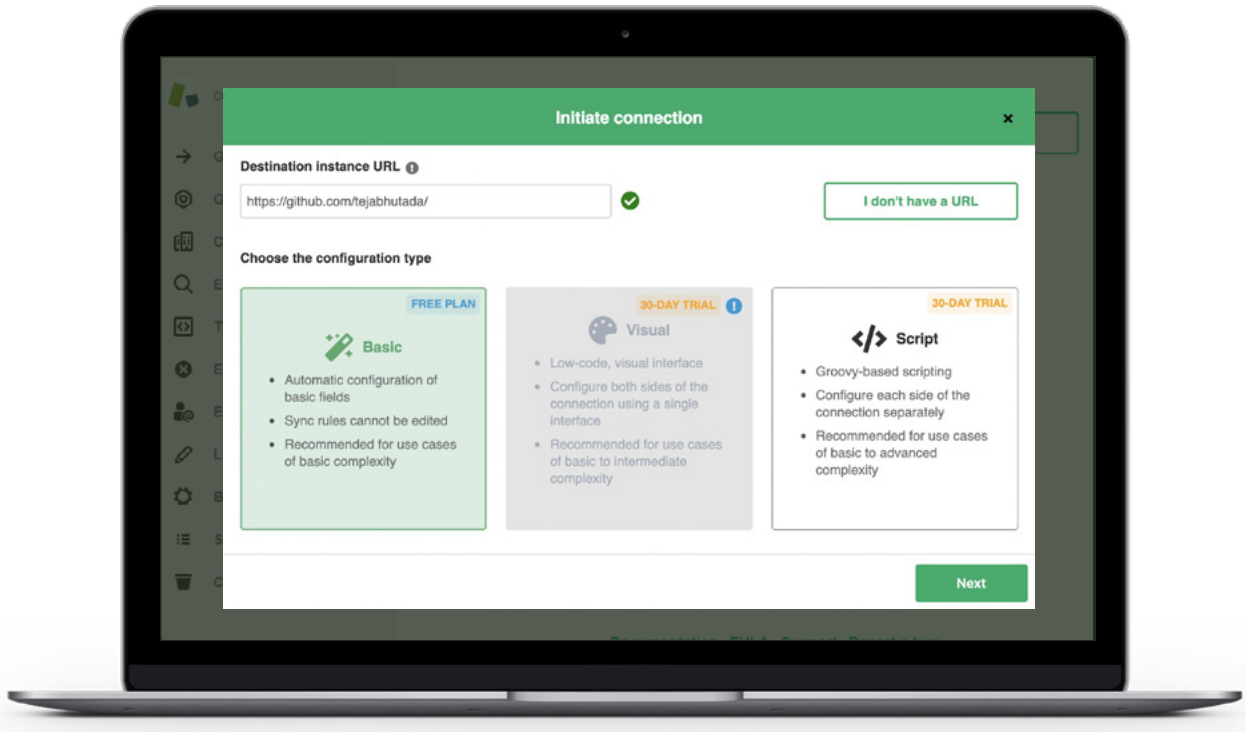
You can start from either side. Exalate’s interface is consistent across platforms, so once you’ve set it up, you can repeat the process easily. I’ll start from the Jira side.

On Jira, click “Apps” and then look for “Exalate” in the left-hand menu. Click “Connections”. Connections you create will be added here. Click on the “Initiate Connection” button.



You’ll be taken through several screens that let you configure your connection.

On the first screen after clicking “Initiate Connection”, you need to let Exalate know where your other instance is. Since we’re in Jira, enter your GitHub instance URL in the field provided. If you were in GitHub, you’d enter your Jira address instead.



The Visual Mode is a low-code interface for setting up connections between issue trackers. But it is not available yet for Jira-Github integration.

After Exalate finds your GitHub instance, it asks you to choose the configuration type. It supports 2 modes: the Basic Mode and the Script Mode.

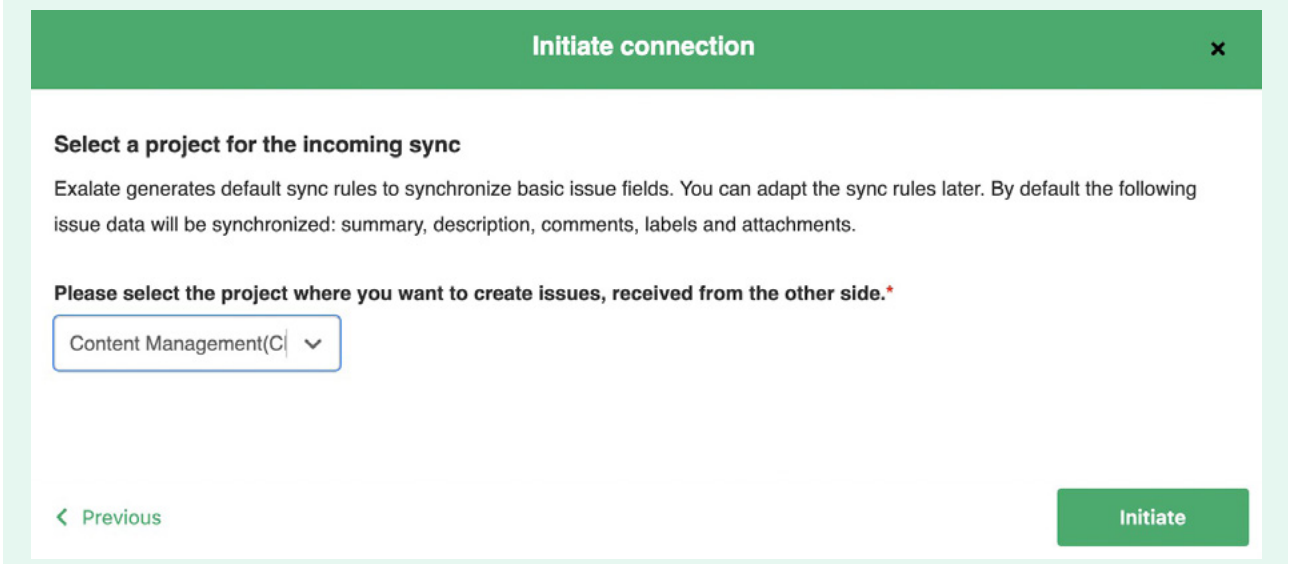
Basic Mode: enables you to set up a connection for a limited set of issue fields like summary, description, comments, attachments, and issue types. The sync rules are generated automatically by Exalate and cannot be modified. These connections are recommended for use cases with basic synchronization needs. This is a free plan.

Script Mode: is a bit trickier, but lets you use more advanced programming logic. The scripts are groovy-based, so you'll have ultimate flexibility to define your sync however you want. This plan is paid, but we offer a 30-day free trial. Install it here.

Continue with the Basic Mode

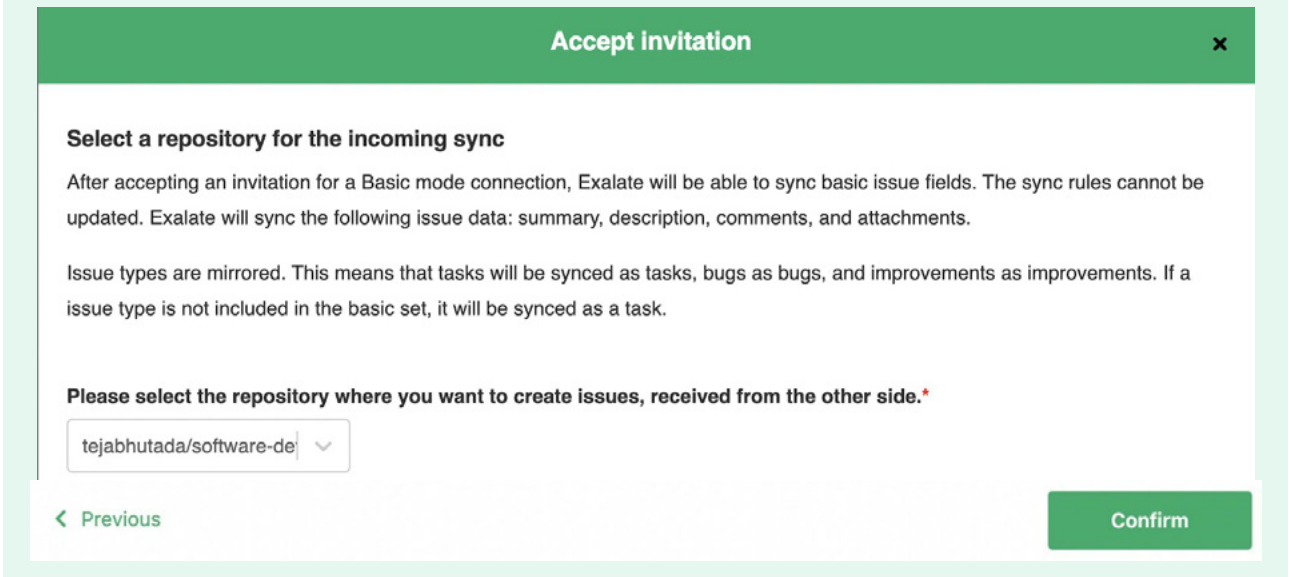
For getting started with the Basic Mode, click “Next” after selecting it.

This will redirect you to selecting a project on the Jira side. It’s the project you want to synchronize issues from Jira in. Choose the appropriate one from a dropdown list.



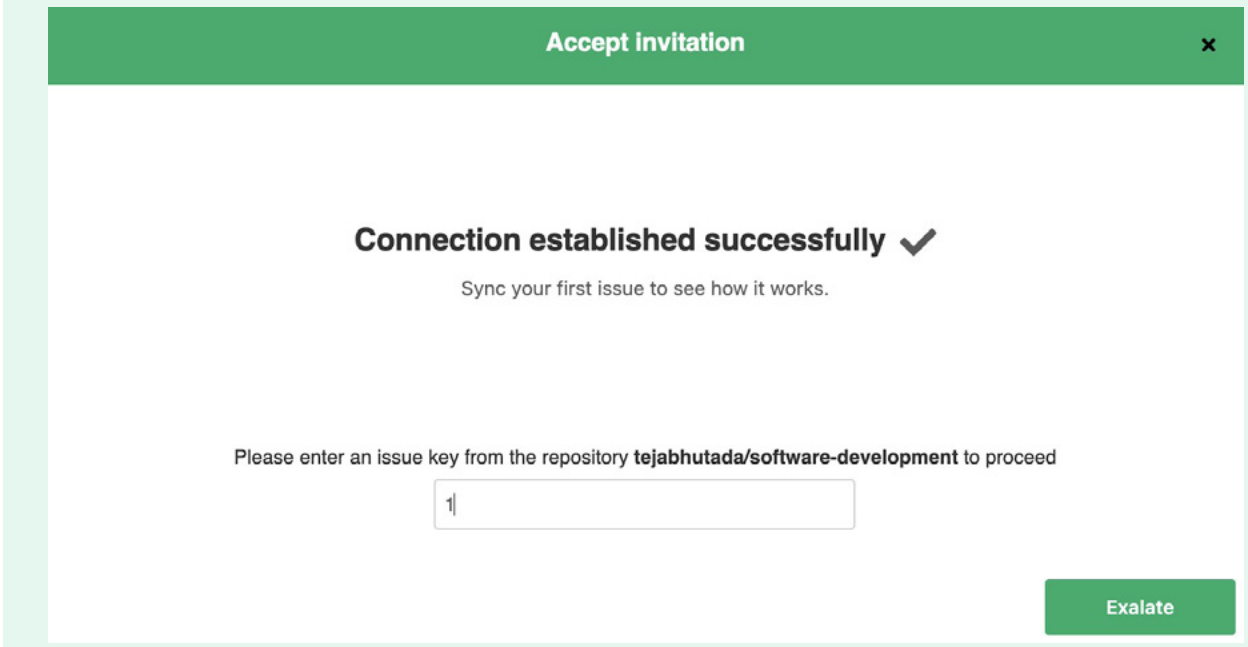
After clicking “Next”, you need to verify if you have admin access to the Github side. If you don’t have admin access, you will be redirected to the Github side to paste an invitation code.

For now, we click on “Yes, I have admin access” and then “Initiate” since we already have access to Github.

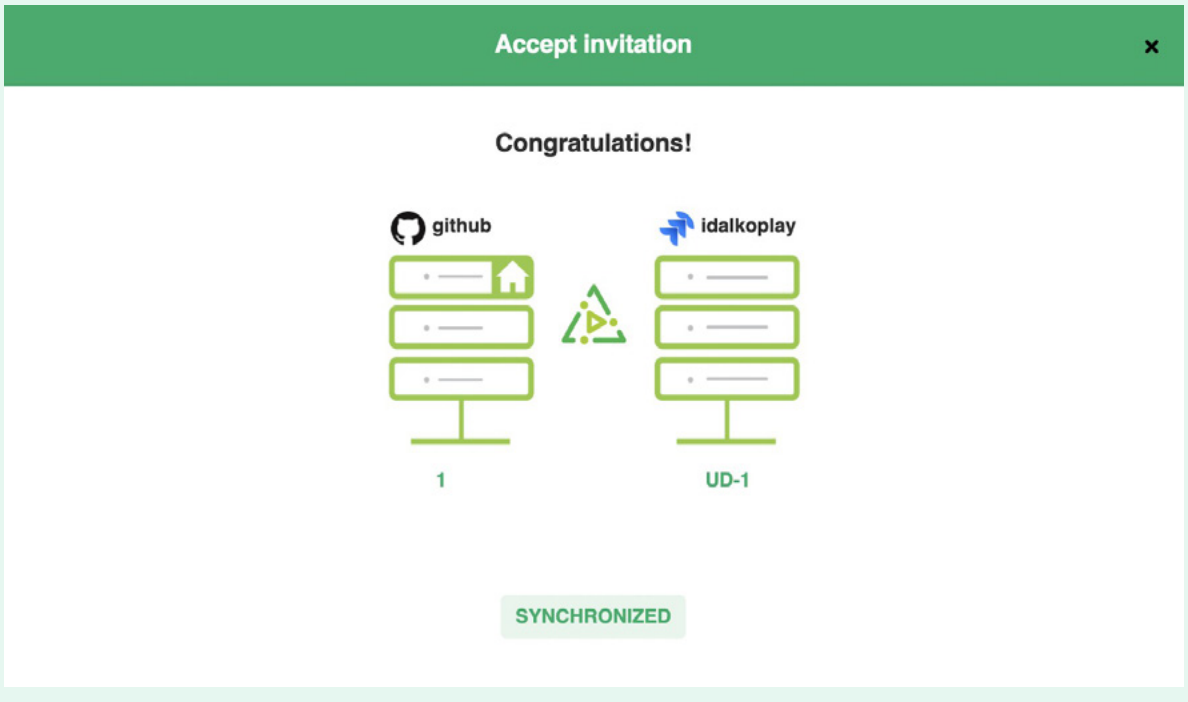


Select the repository you want to synchronize on the Github side and click “Confirm”.

Once the connection has been successfully established, you move ahead to syncing your first issue. The best thing about the Basic mode is that you can start directly with syncing specific issues, or you can always create triggers or sync issues in bulk.



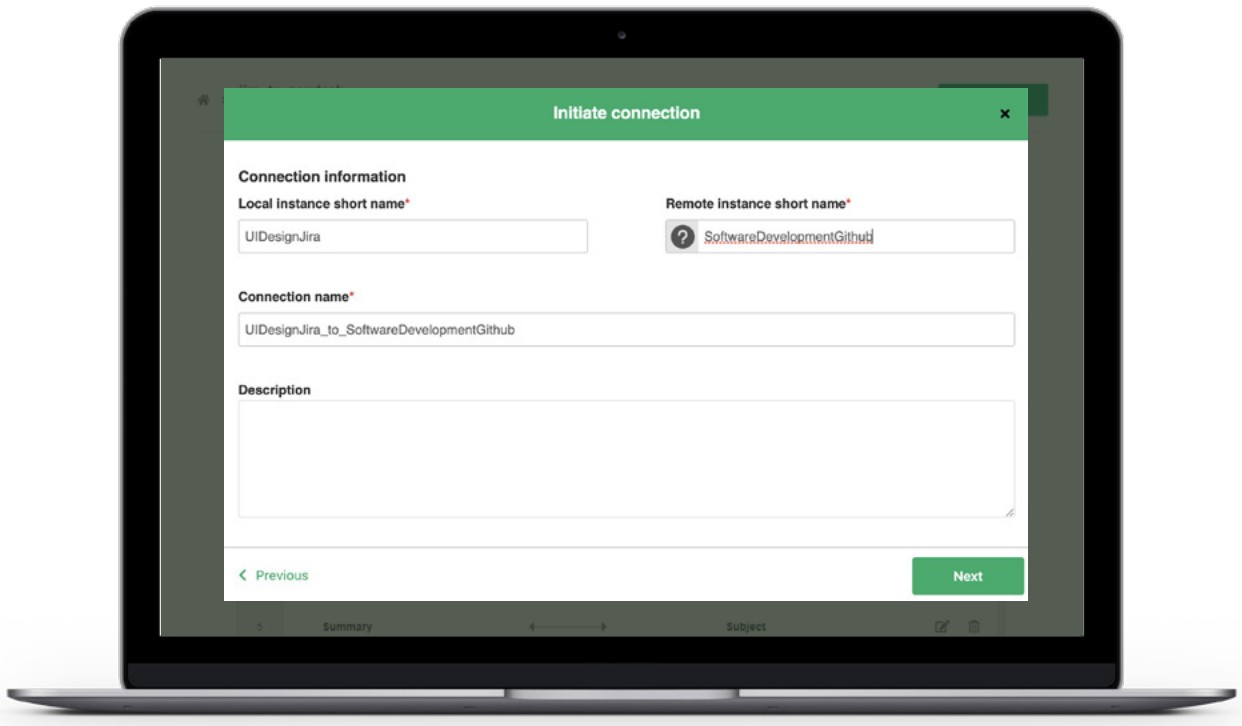
Enter the issue key from the repository and press “Exalate”. Wait for some time before the issue is successfully synchronized.



Continue with the Script mode

If you wish to use the Script mode, then select it and click “Next”.

Some more fields will appear allowing you to name each side of your connection. You might just want to call them Jira and GitHub, but alternatively, you could give them a name that reflects their role, or the team using them.



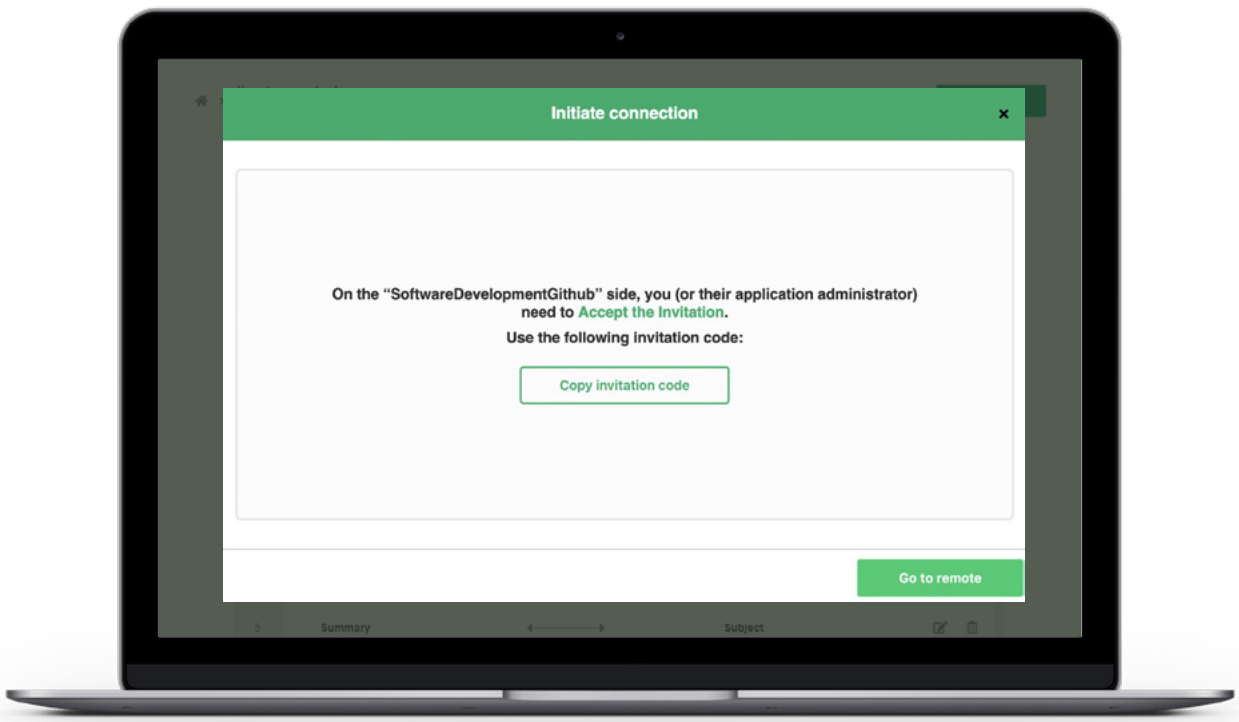
For instance here, we have made connections between the software development team using Github and the UI Design team using Jira.

Exalate will create a connection name using the instance names you used for each side. You can change this too if you like.

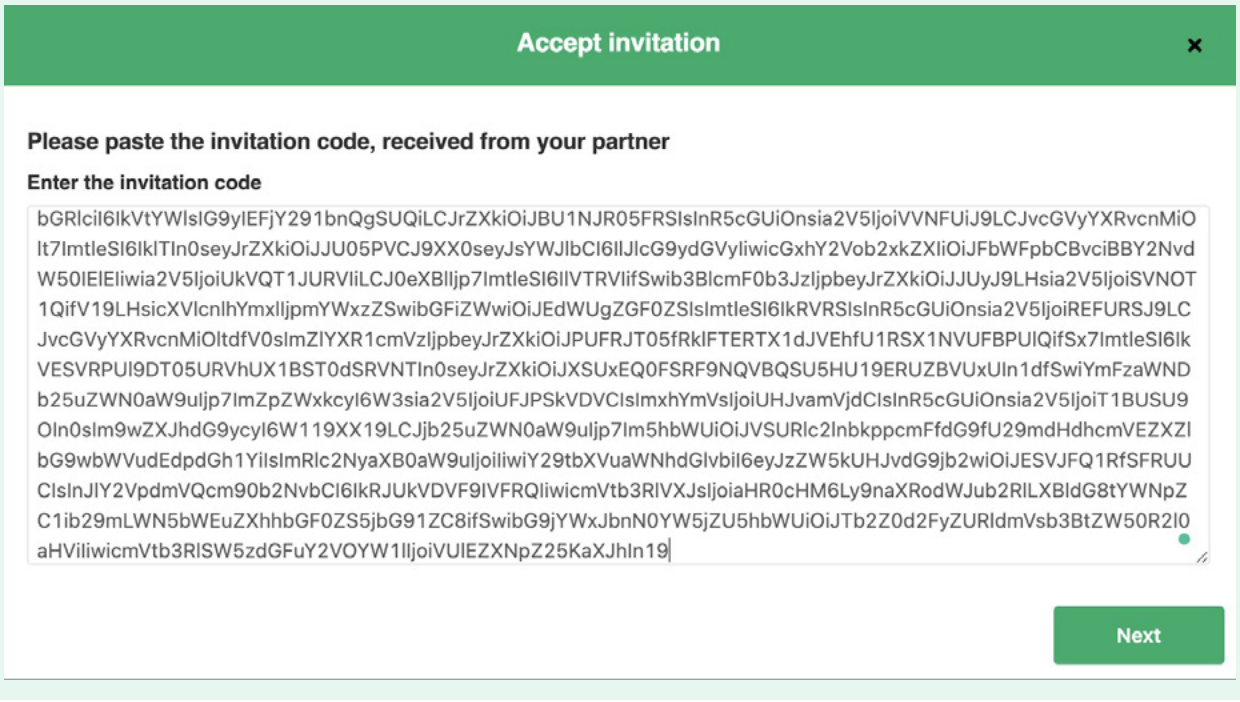
There’s also an optional description field which is best to be filled in, in case you forget what the connection is for later, or if someone else needs to check it. If you end up with lots of connections, this will also help you figure out what each one does.

Click “Next” when you’re happy with the description.

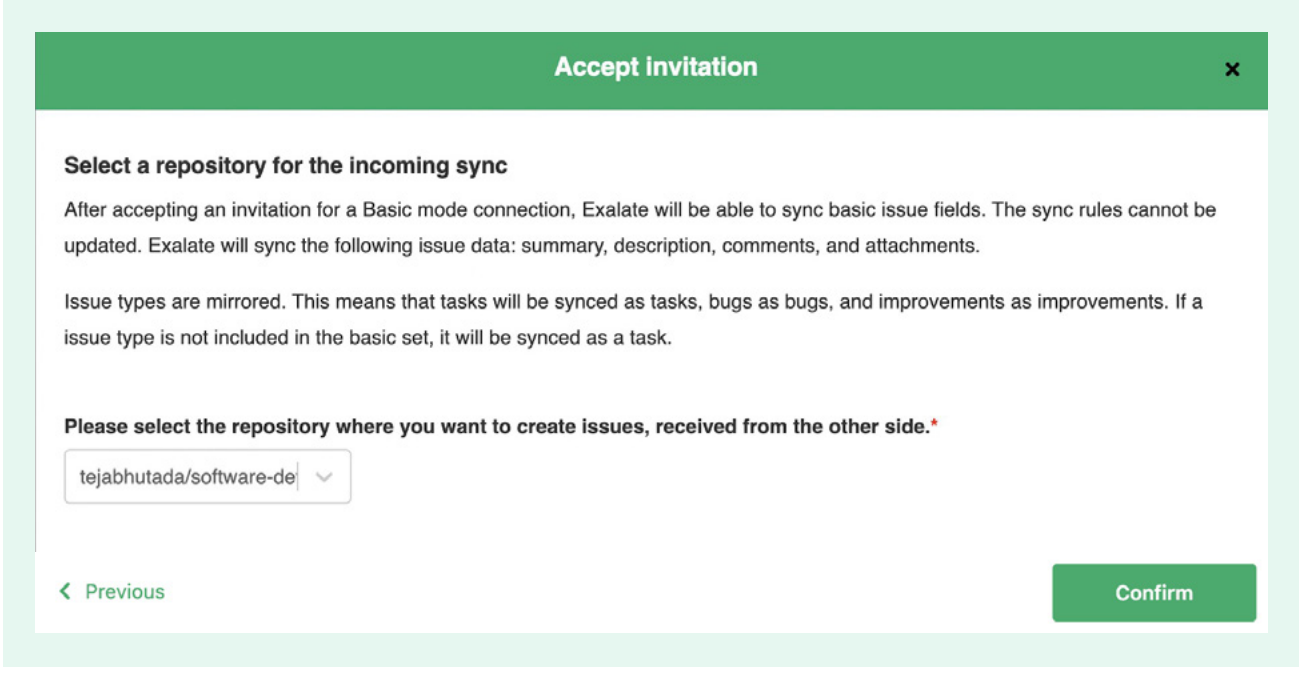
Now select the project on the Jira side and click “Initiate”. Exalate will now generate an invitation code for you. Click the green button to copy it to your clipboard. You should paste it somewhere safe.



Now we need to go to GitHub (you can do this by clicking “Go to remote”) and use our code to activate the connection. In the left-hand Exalate menu, click “Connections”. Then click the “Accept Invitation” button. Paste the invitation code into the field that appears and click “Next”.

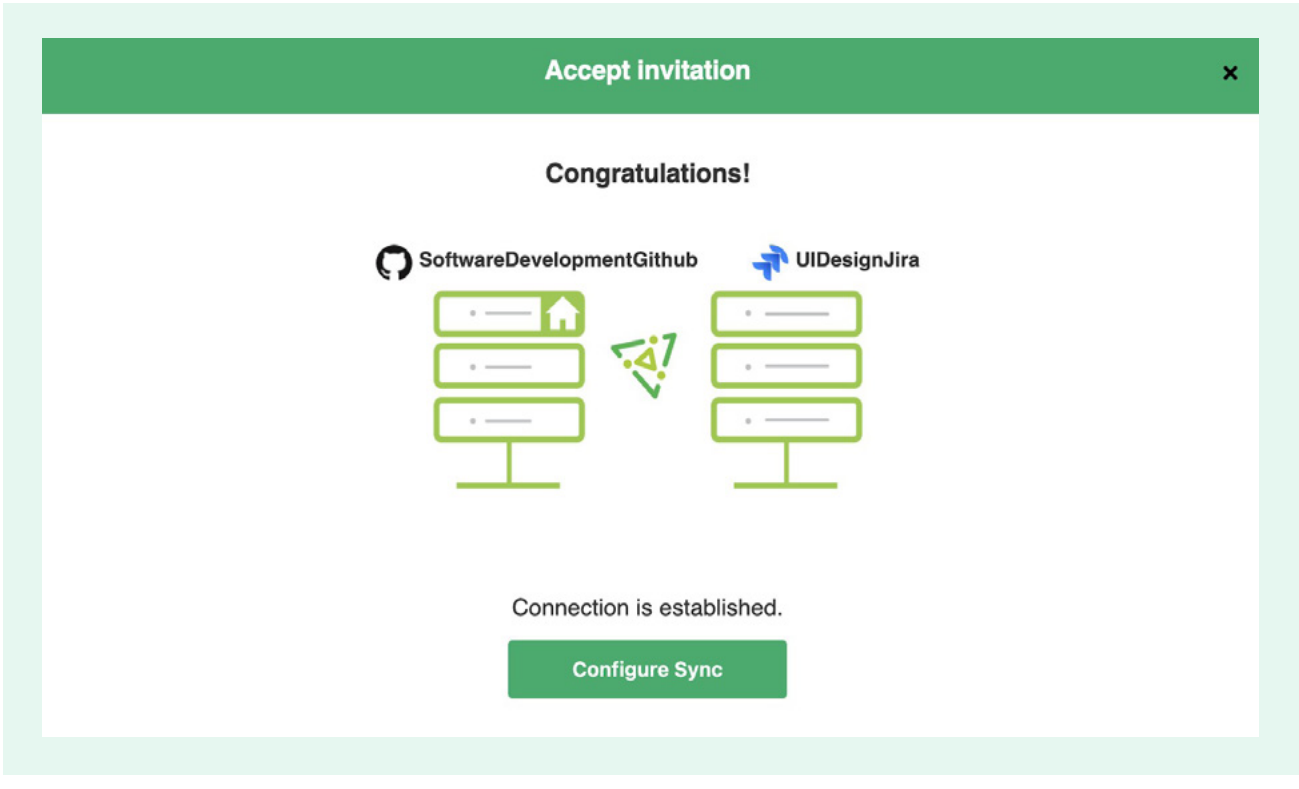


You'll be taken through a similar process to when you created the invitation in Jira. As before, choose the repository name from the drop-down list.



Click the green "Confirm" button.

After that's done, your connection is ready and you can move on to configuring it.



PRO TIP

Want someone to help you get started with Exalate?

Our partners are located all over the world in more than 50 countries. They will help you with implementation, support, and license management for Exalate. They can give a demo, prepare a PoC, and handle almost any complex use-cases.

FIND A PARTNER

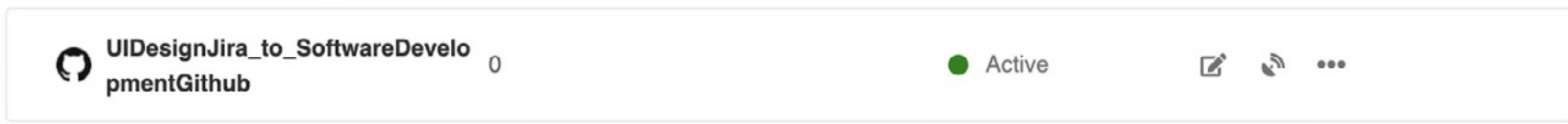
4. Configure Your Connection to Determine What Gets Shared

In this step, you'll take control over what your connection shares, and learn how data is mapped from your Jira issues to GitHub issues and vice versa. You can do this step in either Jira or GitHub, the process is the same for each one.

The configuration can be done after pressing the "Configure Sync" button in the above screen or by the "Edit Connection" under the "Connections" tab. The difference between them is that the first approach allows for configuration to happen for the first time when the connection is being created while the latter allows editing the connection after it's active.

In this guide, I'll use Jira, so take a look at the Jira connections screen and move your mouse over the connection you just created.

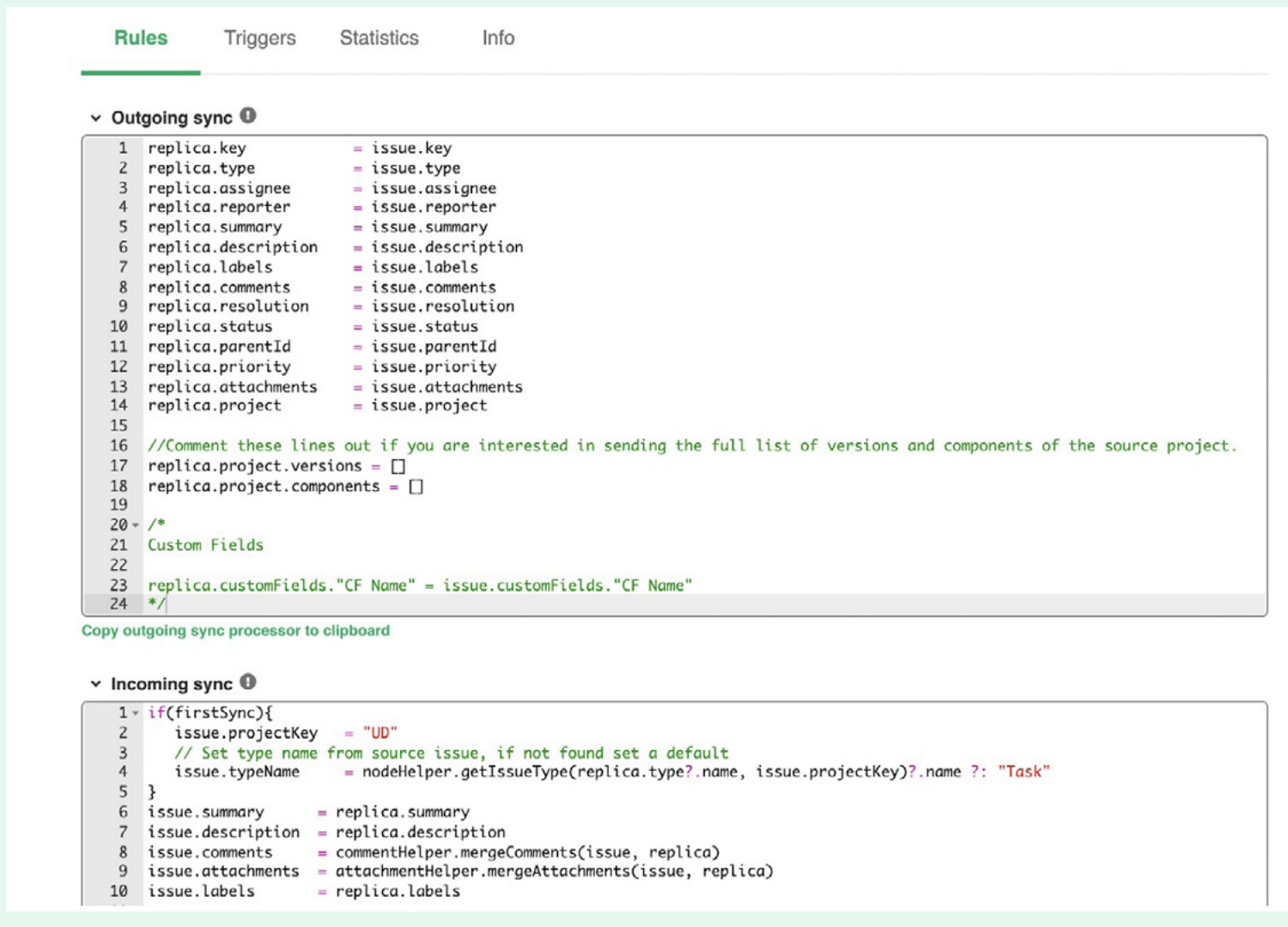
Several icons will appear. There's a square looking icon that lets you edit your connection, a radio signal icon that takes you to the other side of your connection, and three dots that let you activate or delete your connection.



Click the “Edit connection” icon. You’ll see a screen with four tabs. In the next step, you’ll learn about the “Triggers” tab, and in a second, I’ll tell you about the “Rules” tab, but it’s worth knowing about the other two.

The “Statistics” tab gives you information on how many issues are being shared, and lets you know when items were last synced. On the “Info” tab, you can see the URL for the other end of the connection and some other data.

For now, click on the “Rules” tab. Here you can see two sets of rules. At the top, the outgoing sync rules control how data in Jira issues are sent out to GitHub. Below that, the incoming sync rules show how data Exalate fetches from GitHub is interpreted by Jira.

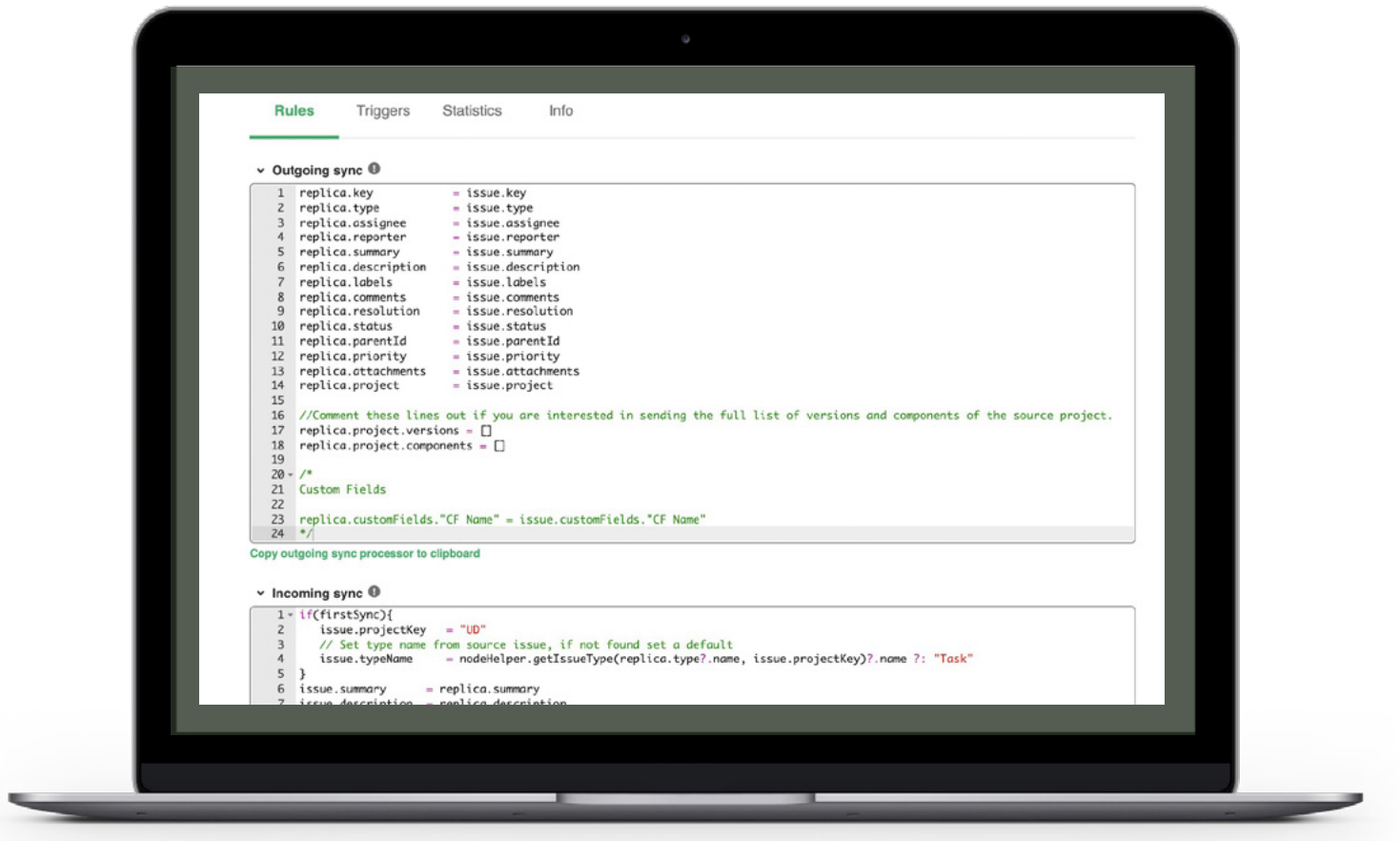


Rules are written in the “Groovy” scripting language. Each line handles a specific field. You can also see comments, which start with “//” for single line comments or are surrounded by “/*” and “*/” for multiple line comments.

Comments are just there to tell you how things work, but you can also use them to temporarily disable anything you don’t want to sync. To do that just put “//” at the start of the line. You can remove it again to reactivate it.

If there is information you don’t want to share, then comment that line out in the outgoing sync rules and Exalate won’t send it from Jira to GitHub.

You can also change the value Exalate copies to a particular field. To change it to a specific value, just replace the incoming value with the value you want to use, in quotes.



For example in the incoming sync rules, you can see the line issue. ***summary = replica.summary***. You could change that to ***issue.summary = "from GitHub"***. Doing that would mean all synced issues would have "from GitHub" as their summary value. You could also add a line, for instance, ***issue.assignee = "Kevin"***, to assign incoming issues to a specific person.

If you're comfortable with core programming concepts, you can use conditional statements to give you even more control over what happens. An "If" statement can let you sync items that meet the conditions you specify.

You could write if ***(replica.assignee == "Sally") { issue.priority = "Critical" }***, to ensure issues from Sally are given the urgency they deserve.

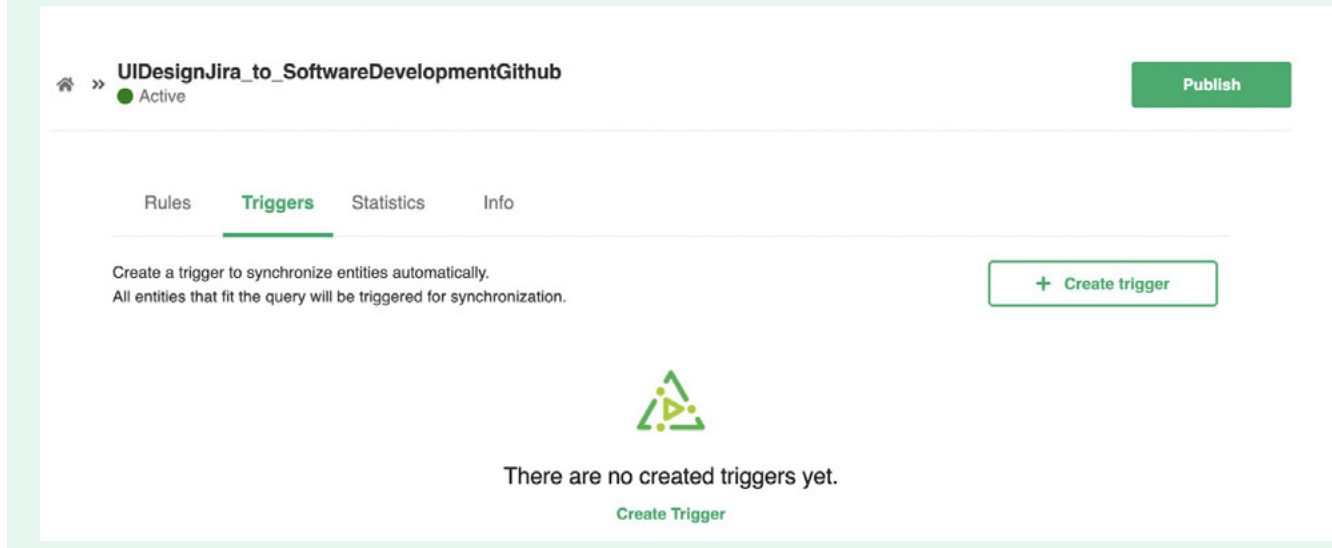
You can no doubt think of other ways to route the information. As you use the synchronization more, you will learn which information is useful to you. You can then adjust it to meet your requirements.

To learn more about sync rules, take a look at the documentation.

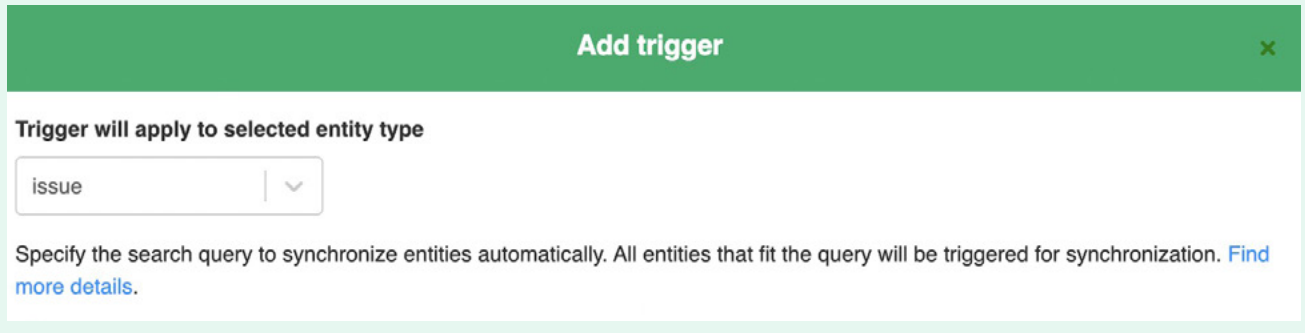
5. Set Up Automated Synchronization Triggers

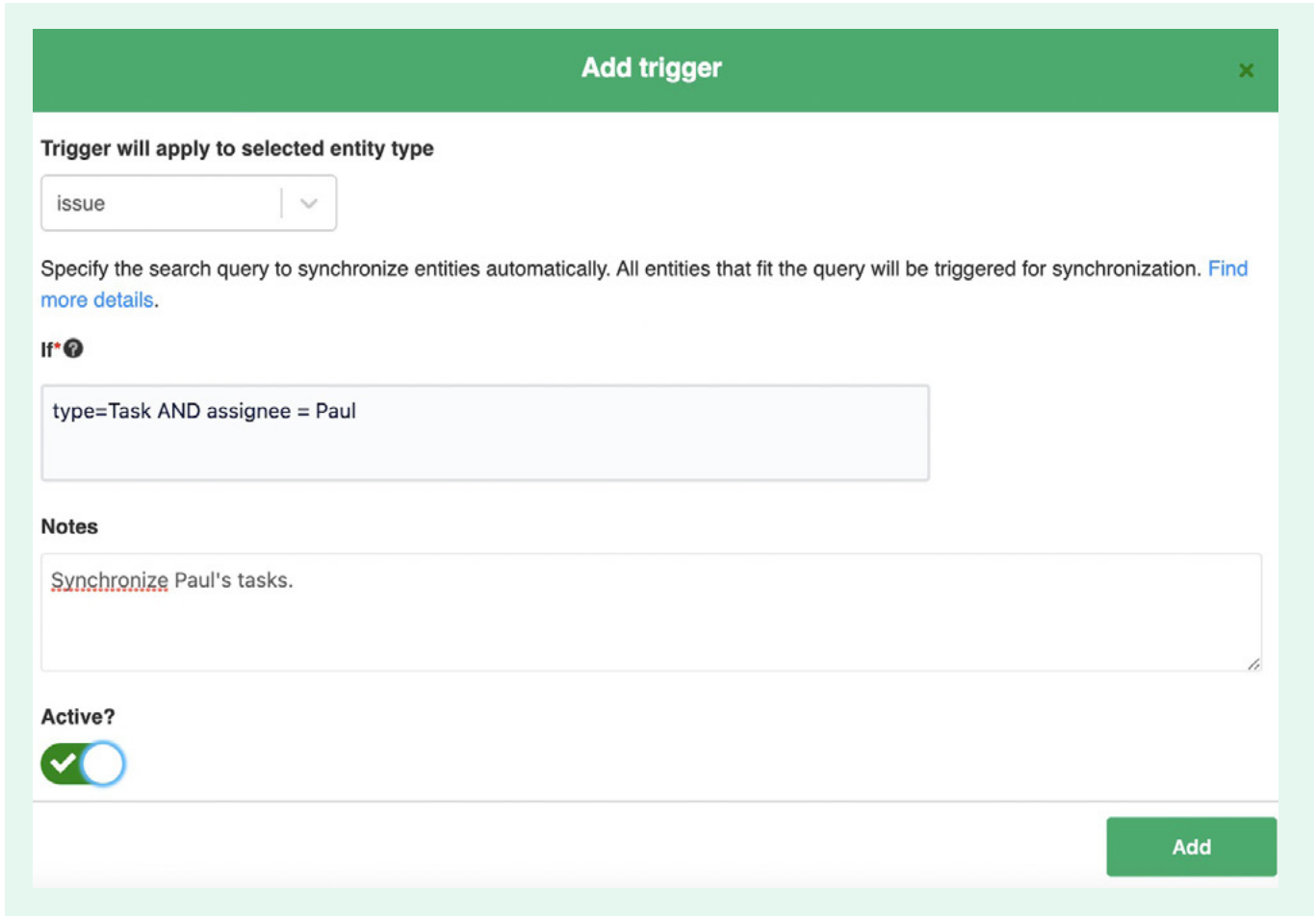
In the previous step, you defined what information Jira and GitHub share, and how they map it onto items. Now you'll define the conditions that trigger information exchange. On the edit connection screen (see the previous step), click the "Triggers" tab.

There's also a "Triggers" item in the left-hand menu which works similarly. If you use that, you'll see a list of triggers with an entry showing what connection they are for, and can pick one to edit. If you create a new trigger from this screen the only difference is you'll need to choose which connection it applies to.



To create your first trigger, click the white "Create trigger" button. The "Add trigger" pop-up will appear. The first field, at the top, is a drop-down menu letting you pick the kind of item that the trigger applies to. In the screenshot, I've selected "issue", but you can pick whatever you like.





Below that is a conditional field. This is where you set the rules that apply to the type of entity specified in the drop-down menu.

Jira's triggers use the JQL query language. Using it, you can create simple conditional statements to select specific entities, and also combine them using logical operators like "AND" and "OR".

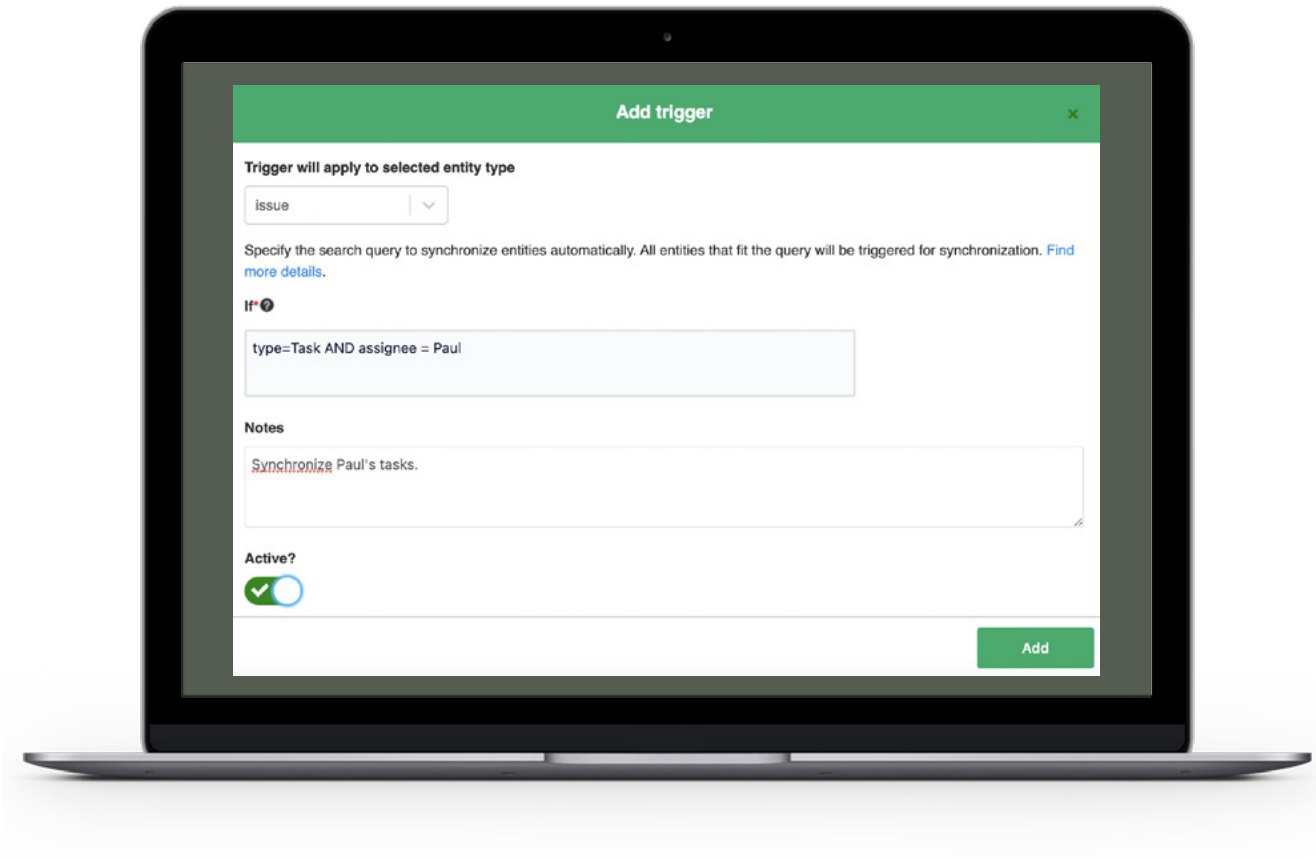
If you wanted to synchronize items that are assigned to Paul, you could add ***assignee = Paul*** to the field. If you wanted to pick out items of the "Tasks" type, you could add ***type = Task.*** To synchronize items that meet both these conditions simultaneously, you could use ***type = Task AND assignee = Paul***. That's what I've done in the screenshot above.

There is also a notes field, where you can describe what you've done. The more detail you add here, the easier it will be to work with later. This is especially useful if you have multiple triggers, or other people are going to be using the synchronization.

Read more about JQL triggers in the documentation.

Under the notes field is a switch that lets you activate the trigger. It won't work if you don't activate it! Triggers can be turned on and off easily, so instead of deleting them when you don't need them, you can disable them, and quickly reactivate them later if you need to.

When you're ready, click the "Add" button and the trigger will appear in your list. If there are tickets that match it, Exalate will start syncing them.



6. Start Synchronizing Tasks

Now that Jira and GitHub are connected you can start sharing tasks between them. Exalate can take a few minutes to synchronize everything, so don't panic if they don't get shared immediately.

If you wait a few minutes and check your connection statistics, you should see that items have been shared. If not, take a look at your rules and make sure there are some items they apply to. It might be worth creating an item that meets your conditions to test if the connection is working.

CHAPTER 4

COMMON USE CASES

There are teams using Jira and GitHub in many different scenarios and there are all sorts of cases where you would want to benefit from a Jira GitHub integration. To give you a few ideas on what you can do with integration, here are some example situations.

Software Project with Code on a Public Git Repository

If your developers have publicly available code, they may want to take advantage of GitHub's project management and issue reporting features and bring data into their own private system. With an integration that is easy to do.

You can have items pulled into Jira from GitHub letting you track the public discussion while keeping your own team's comments private. Issues on the public codebase can be assigned to someone specific, who can decide if the team needs to take further action on them.

You can then have fields that you want to share synced back to GitHub from Jira. That allows you to work with the open-source community, while keeping internal discussion private and gives you the best of both worlds.

"The most beneficial impact we've got from Exalate is the ability to be in full control of our GitHub issues that users raise. Because if you have over 50 active repositories, you cannot help with missing issues over time and they'd easily go stale. But with Exalate, we're now able to manage them all in one place. That is a real lifesaver."

GERWIN KLEIN | SEL4

READ FULL STORY →

Customer Service Team and Engineering Team

Your customer service team has to deal with problems reported by your customers, and they record them as issues in Jira. They can handle many of those themselves, but sometimes they will need to pass them on to engineers.

Instead of doing this manually, you can set up a Jira GitHub integration that automatically copies relevant fields from flagged tickets into GitHub. The Engineering team can then deal with these how they like and, when resolved, the integration passed specific details back into Jira.

That way any solution or suggestions from engineering are sent back to customer support, who can pass them on to customers.

Product Development and Quality Control

Your quality control team needs to ensure your product meets legal requirements in every market where it is available. They need a system for tracking legal issues and figuring out how to handle them. They also need to monitor the product for potential issues.

Your development team needs to be aware of these requirements and respond to them. A Jira GitHub integration can automatically send issues that require changes over to the development issue tracking system in GitHub. The developers can then work on these as part of their normal workflow.

When issues are resolved, quality control is informed automatically, as the integration copies that information to their system.

CHAPTER 5

CONCLUSION

exalate

Integrating your systems can give your teams a crucial edge over the competition. As you've seen, the technology is out there to connect platforms like Jira and GitHub seamlessly and help you solve the challenges involved.

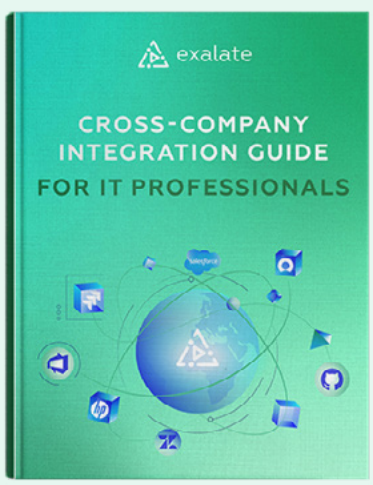
A flexible autonomous Jira GitHub integration can help teams on both sides of the connection continue working in the familiarity of their own environment without worrying about the security or duplication of their data.

LEARN MORE

DOWNLOAD THE EBOOK

DOWNLOAD THE EBOOK

DOWNLOAD THE EBOOK

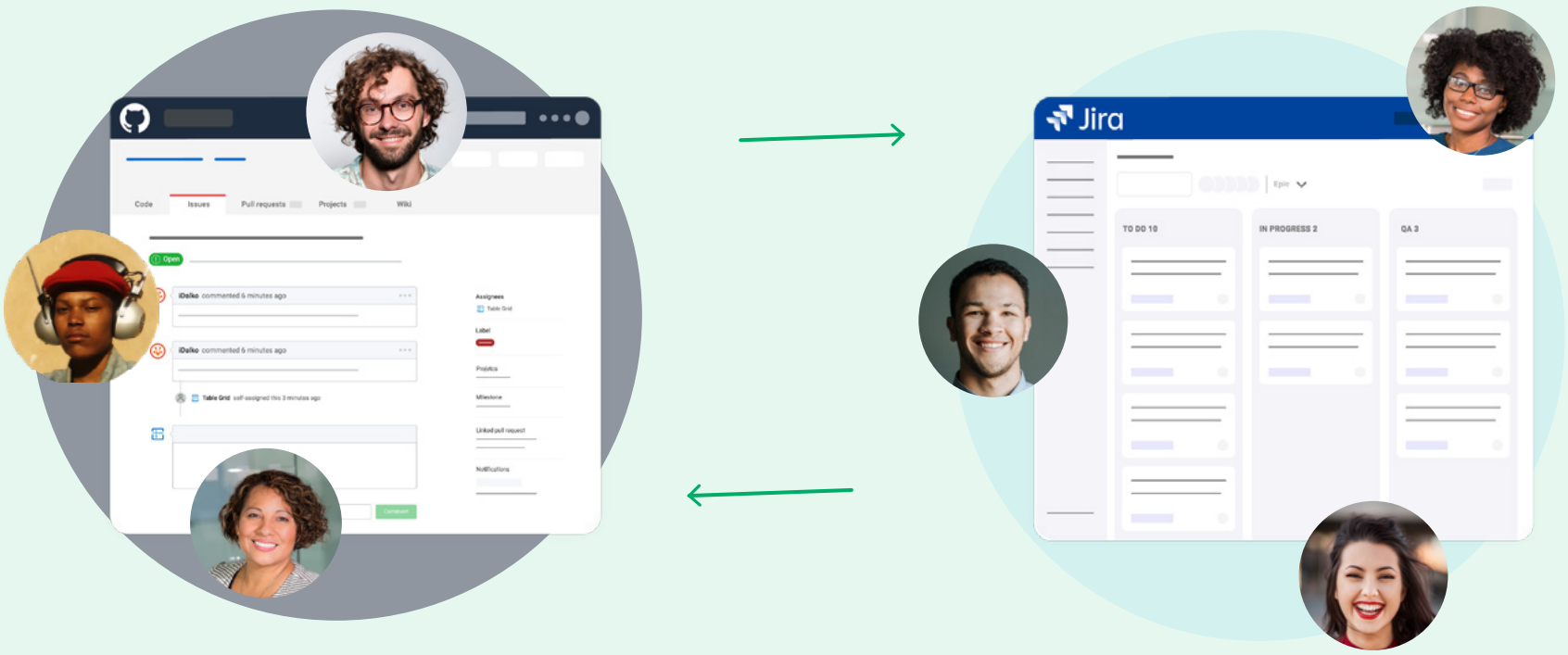


DOWNLOAD THE EBOOK

exalate

The Leading Cross-Company Integration Solution

Exalate is the leading and the only solution specifically built for cross-company integration scenarios with the most complex synchronization use cases.



The first thing that hits you when using this app is how easy it is to get a synchronization up and running. By default, everything works so well.

With the amazing support team and the flexible capability of Exalate, we were able to see it work nicely and become an important part of our work life cycle.

Exalate is marketed as the most flexible synchronization tool for issue trackers and that short definitions is pretty accurate from what we have observed.

BOOK A DEMO

TRY IT FOR FREE