GitHub Salesforce Integration: How to Set up a Sync in 6 Steps

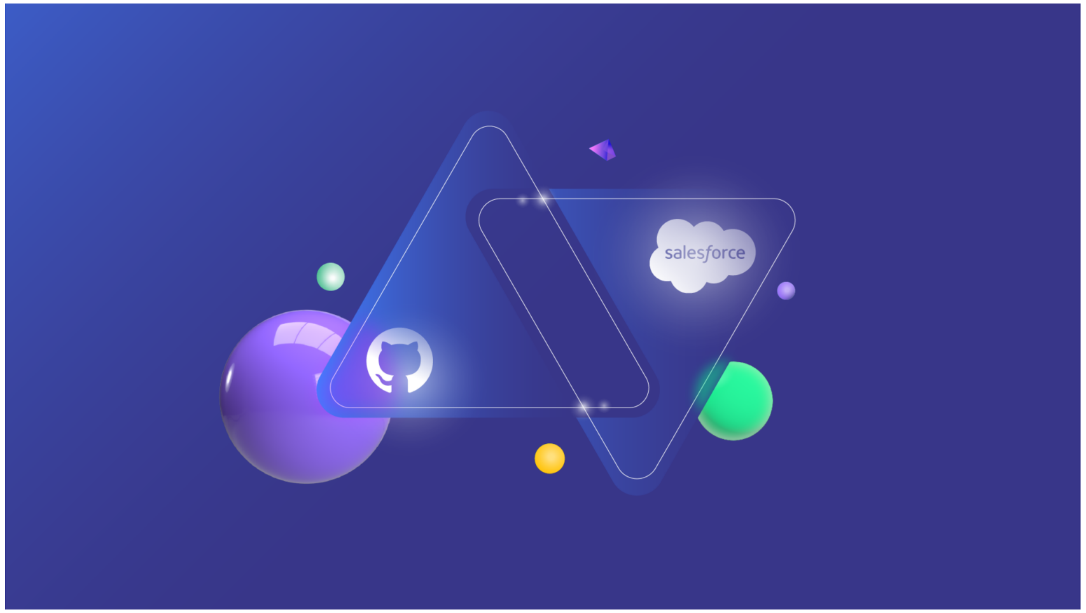


Table of contents

GitHub and Salesforce are among the most popular applications used by development and sales teams. These applications can help teams manage their work and workflows in a better manner. But these teams will benefit from the information being exchanged automatically between them, through a GitHub Salesforce integration, to have access to consistent and up-to-date business data.

In this guide, we will see how these teams can benefit from a GitHub Salesforce integration. We will also have a look at what factors to consider while choosing an integration technology.

We then move ahead and study a step-by-step approach for implementing this integration. And lastly, we have a look at the most common use cases of a Salesforce GitHub integration.

An overview of what is covered in this blog post:

***Note**: For this guide, we are using an integration tool called Exalate. We will learn more about it as we proceed.*

Get the GitHub Salesforce Integration Guide

Learn how to integrate GitHub and Salesforce, step-by-step.

GET THE EBOOK

Why Set up a GitHub Salesforce Integration in the First Place

GitHub

GitHub is an online source code management and versioning platform. It helps developers consolidate and collaborate their codes to maintain a consistent version that can be accessed and tracked by all the team members.

It is popular among the open-source community. It also supports issues that are worked on until resolved so developers can monitor and keep track of their progress.

Salesforce

Salesforce is a CRM tool that supports the entire sales workflow, right from lead generation to post-sales support. It is widely used by sales teams to have a complete overview of customers.

Accounts, **Products**, **Opportunities**, **Tasks**, and **Cases** are popular Salesforce entities that cater to different customer-centric tasks through a single Salesforce interface.

Why Integrate GitHub and Salesforce?

It's a known fact that enhanced customer support and better resolution of their queries are major factors to keep existing customers happier and maintain a low churn. So, to achieve this common goal, teams in an organization strive hard to search for the information they need in different applications or ask other team members for it.

But this way of doing things is manual. Teams either ask for information through phone calls or emails or get in endless meetings to gather the relevant details they need or spend time copy-pasting information by toggling between different applications.

All this leads to manual data entry mistakes, redundant, misplaced, or altered information, and above all wastes a lot of productive time.

Automated information exchange with the help of a cross-company integration tool can help teams maintain a single, consistent view of business-critical information without having to leave their current platform of choice.

So if your teams use Salesforce and GitHub, then an automatic, real-time data exchange through a GitHub Salesforce integration can help them deliver a better customer experience and allow them to collaborate harmoniously with each other.

Choosing the Right Technology for a GitHub Salesforce Integration

However, implementing a GitHub Salesforce integration is not as easy as it sounds.

It requires researching different tools and technologies that offer such integration and choosing the right one based on your business needs.

Here are a few drivers to help you get started.

Decentralized Integration

A tool having a decentralized integration must be your top choice. This is because it will allow you the autonomy to work in your environment independently without having to consult the other side. This lets you control what information you send and how you want to receive it. So, each side has independent information exchange rules.

Security

Data is of paramount importance for any business. So when you want to exchange it, care must be taken that it's done with the correct security measures in place.

Encrypted file transfers, access control mechanisms, and secure transfer protocols like HTTPS, etc should be inherently supported by the tool.

Flexibility

Integrations mature and change with time. So do your requirements for sending and receiving information. Sometimes you want to share something extra, or you want to stop sharing some data.

Here, how the tool adapts to these changing requirements must be acknowledged beforehand. Accommodating unique and advanced business integration use cases with minimal configurations must be an inherent feature of the integration tool you choose.

Reliability

Computer systems are prone to downtimes and failures. In such situations, your tool must be able to withstand the downtime and be able to apply the changes queued for synchronization in the same manner as they had originated. All this should happen automatically so the integration parties don't feel that there was anything wrong in the first place.

Number of Integrations Supported

We are discussing a GitHub Salesforce integration here, but it's just one single possibility amongst many others. You might have partners, suppliers, customers, or vendors who use different applications like Jira, Azure DevOps, ServiceNow, Zendesk, and the like. Adopting a tool that already supports these integrations is an added advantage.

There are a lot of integration tools in the market, but for this article, I have chosen Exalate.

This is because it has an inbuilt decentralized integration architecture, ensured with the help of sync rules on both sides. Plus, it uses security measures like encrypted file exchange and role-based access controls. Check this security whitepaper for further details.

Exalate also comes with a scripting mode that uses Groovy scripting, in addition to a low-code UI mode, suitable for both business and technical users. This helps you adapt it to your specific integration needs easily.

It has a transactional synchronization engine that queues all the changes that need to be applied automatically and breaks them down into atomic steps, to be retried in case of a failure, even when a system is being upgraded to a new version and/or a firewall is being reconfigured.

And last but not the least, it supports integrations for a variety of applications like Jira (cloud and on-premise), ServiceNow, Azure DevOps, Zendesk, and HP QC/ALM.

So let's get to implementing a GitHub Salesforce integration.

How to Set up a GitHub Salesforce Integration: the Step-by-Step Process

To start setting up a GitHub Salesforce integration, you must first install Exalate on both GitHub and Salesforce. This ensures you have autonomy since the Exalate app on both sides allows you to configure each side separately and helps you control what can be sent and received.

After that, you set up a connection between them, configure it, create automatic synchronization triggers, and experience 2-way sync to effectively handle your synchronization needs for you.

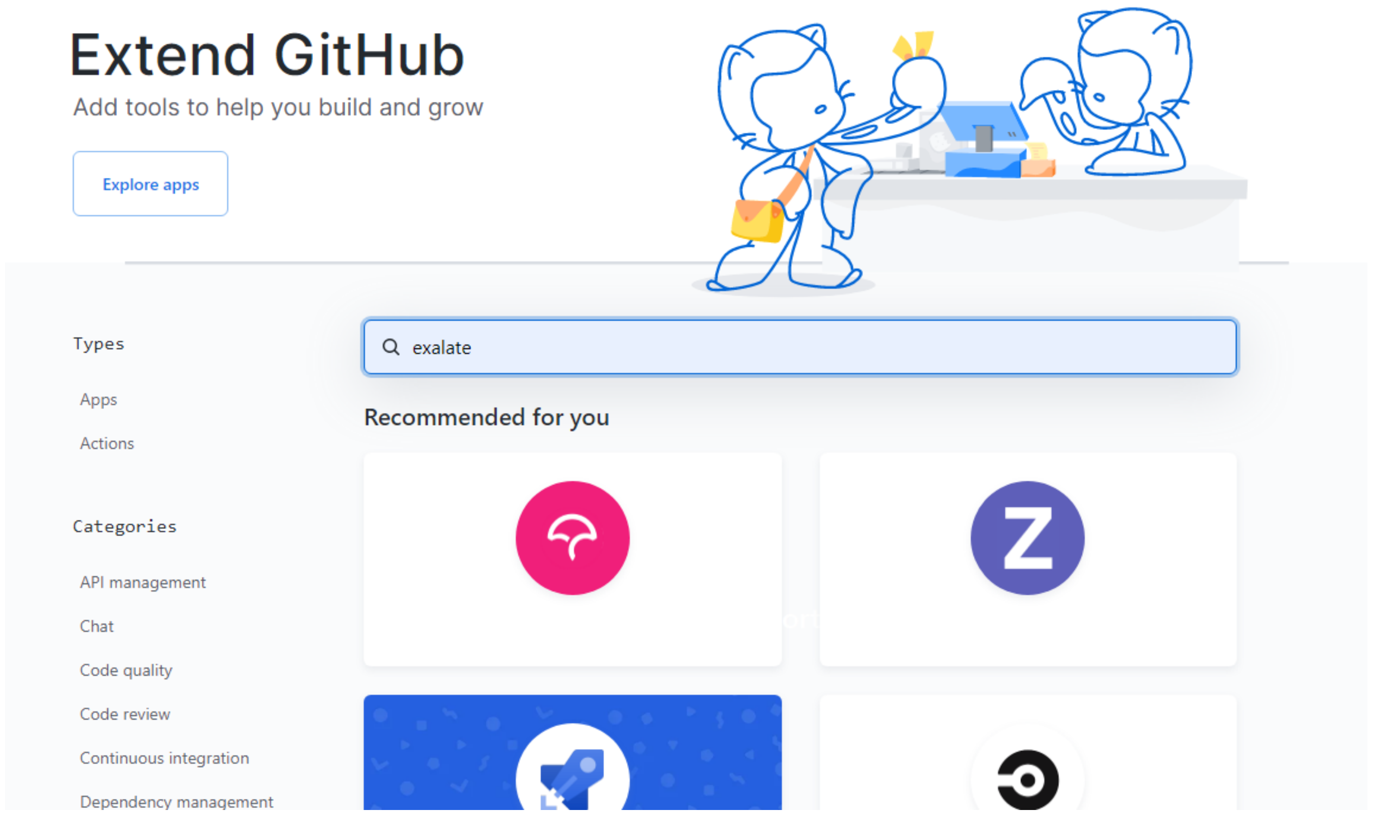
So let's get started!

Step 1: Install Exalate on GitHub

First, log in to your GitHub account.

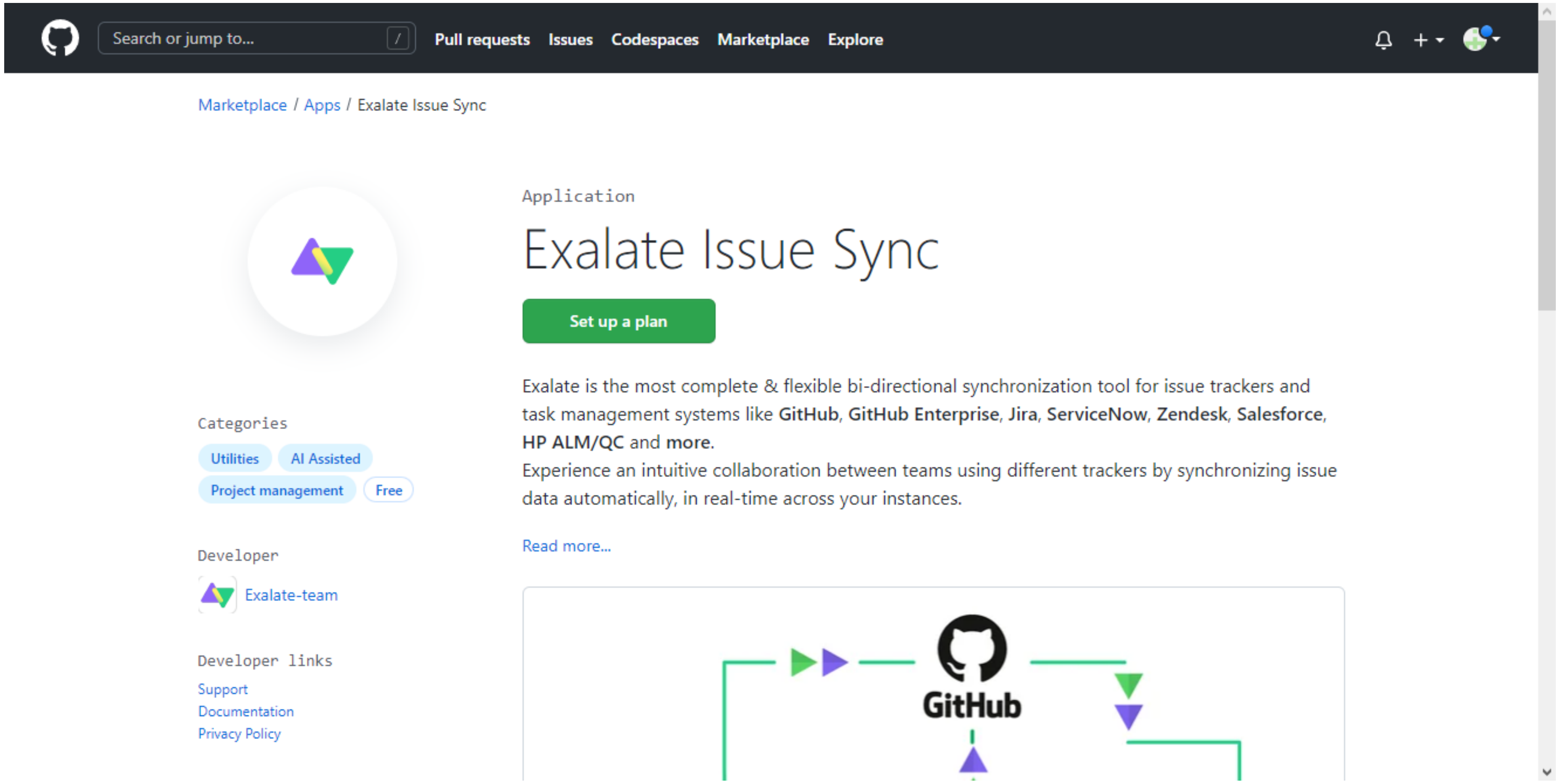
Click on “Marketplace”. Type “Exalate”. Or save some time, by clicking on this link instead.

Note*: Exalate is also available for GitHub Enterprise. For installing Exalate on GitHub Enterprise Cloud, check this article.*



You can use Exalate on GitHub with the 30-day free trial.

To proceed with it, click on the green “Set up a plan” button. Then click the “Install it for free” box.

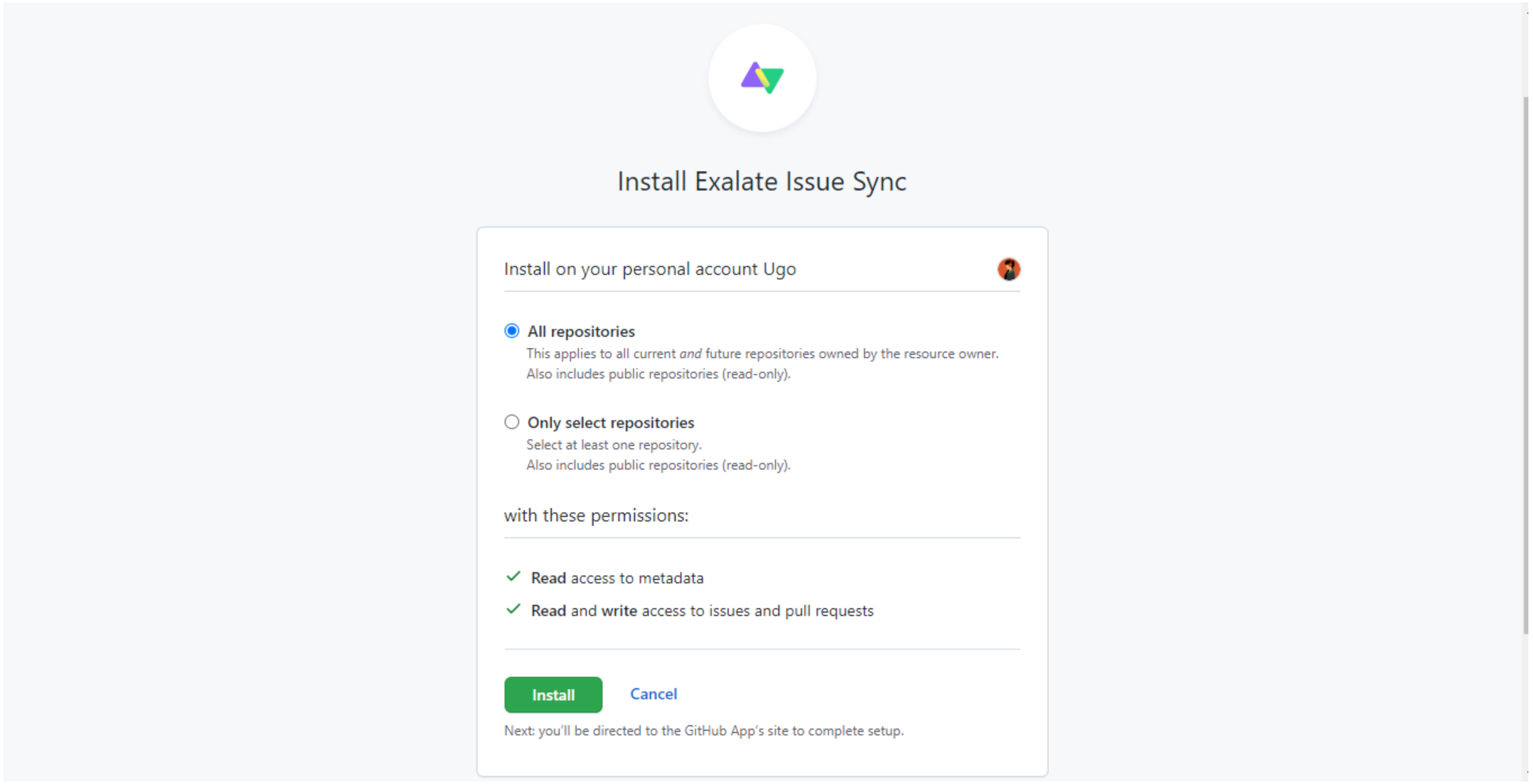


Exalate Issue Sync

Exalate is the most complete & flexible bi-directional synchronization tool for issue trackers and task management systems like **GitHub**, **GitHub Enterprise**, **Jira**, **ServiceNow**, **Zendesk**, **Salesforce**, **HP ALM/QC** and **more**.
Experience an intuitive collaboration between teams using different trackers by synchronizing issue data automatically, in real-time across your instances.

Then you will be asked to give Exalate permission to access your GitHub repositories. You can choose to give access to all of them or select specific ones only.

It is important to note here, that Exalate does not access the GitHub code, but only has access to metadata, issues, and pull requests.


Install Exalate Issue Sync

Install on your personal account Ugo

(●) **All repositories**
This applies to all current *and* future repositories owned by the resource owner.
Also includes public repositories (read-only).

( ) **Only select repositories**
Select at least one repository.
Also includes public repositories (read-only).

with these permissions:

✓ **Read** access to metadata
✓ **Read** and **write** access to issues and pull requests

Next: you'll be directed to the GitHub App's site to complete setup.

Select the permissions you want to give and click “Install”.

You will then be redirected to the Exalate site to accept the user agreement to continue.

Next, you need to select how Exalate accesses your GitHub account, either by an *OAuth* token or by entering your GitHub account username and password. In case you want to regenerate your OAuth token, follow these steps.

You will then need to request an evaluation license for using Exalate on GitHub. The steps for this are pretty straightforward.

Note: *You will not be able to synchronize your issues without requesting the evaluation license.*

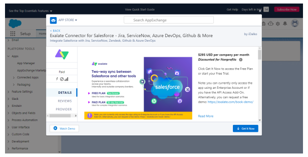
Step 2: Install Exalate on Salesforce

The Exalate app is available on Salesforce AppExchange. To install it you can either type "AppExchange" in the search bar of your Salesforce instance, or you can simply head over to its website directly.

Note*: Exalate accesses Salesforce through APIs. Salesforce has its own guidelines for API Access add-ons. For instance, API access is provided by default in Enterprise accounts, but it's not the case with other accounts like Professional. Visit this documentation page to learn about the different Salesforce editions Exalate supports.*

Once there, type "Exalate" in the search bar. In case you are not logged in, it will ask you to log in to the AppExchange console.

Once the app details are displayed, click on the green "Get It Now" button.



Exalate Connector for Salesforce - Jira, ServiceNow, Azure DevOps, Github & More

Integrate Salesforce with Jira, ServiceNow, Zendesk, Github & Azure DevOps

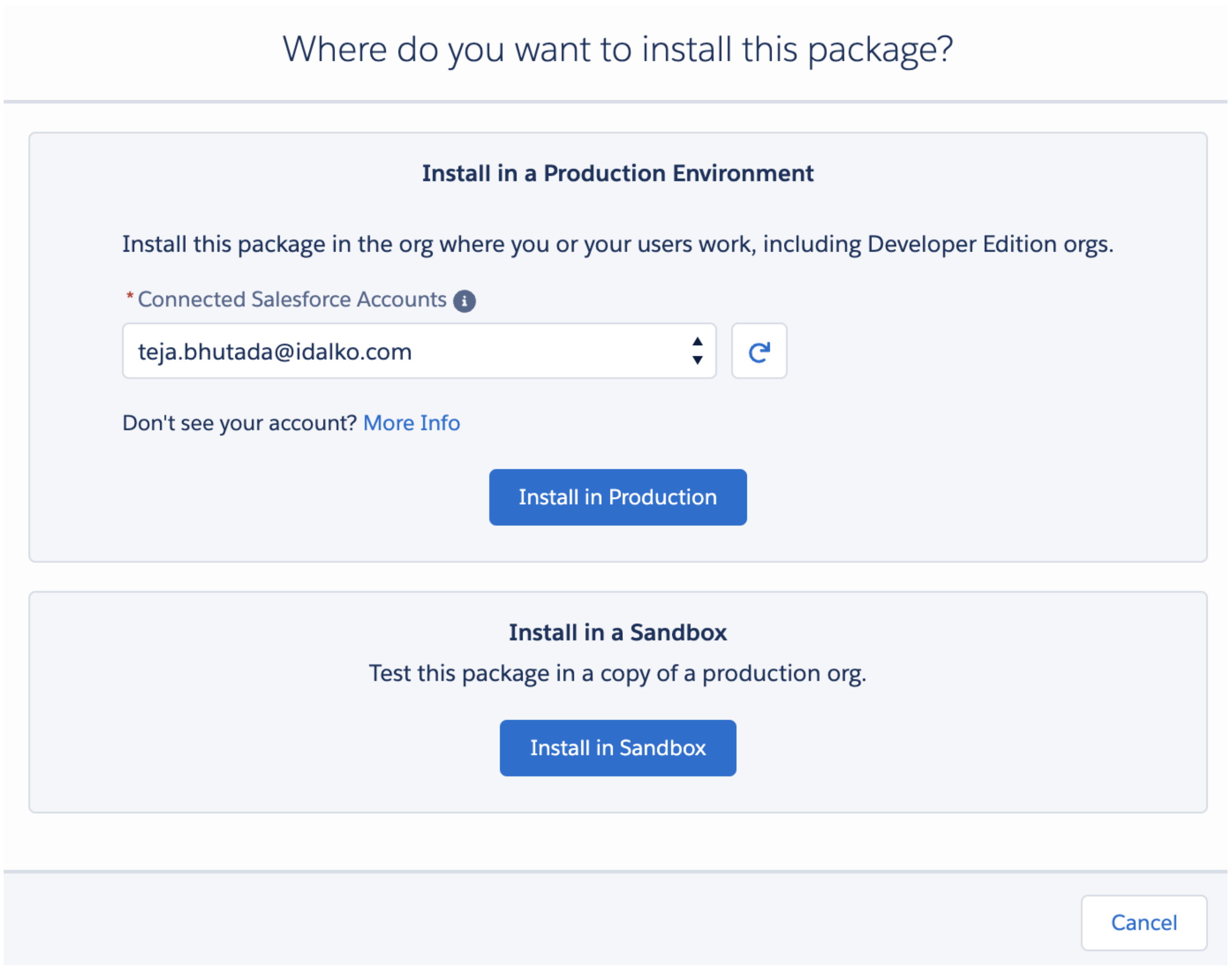
$295 USD per company per month
Discounted for Nonprofits

Click Get It Now to access the Free Plan or start your Free Trial.

Note: you can currently only access the app using an Enterprise Account or if you have the API Access Add-On. Alternatively, you can request a free demo: https://exalate.com/book-demo/

BOOK DEMO

Now choose where you want to install the package: Production or Sandbox. I have chosen "Install in Production" here.



Where do you want to install this package?

Install in a Production Environment

Install this package in the org where you or your users work, including Developer Edition orgs.

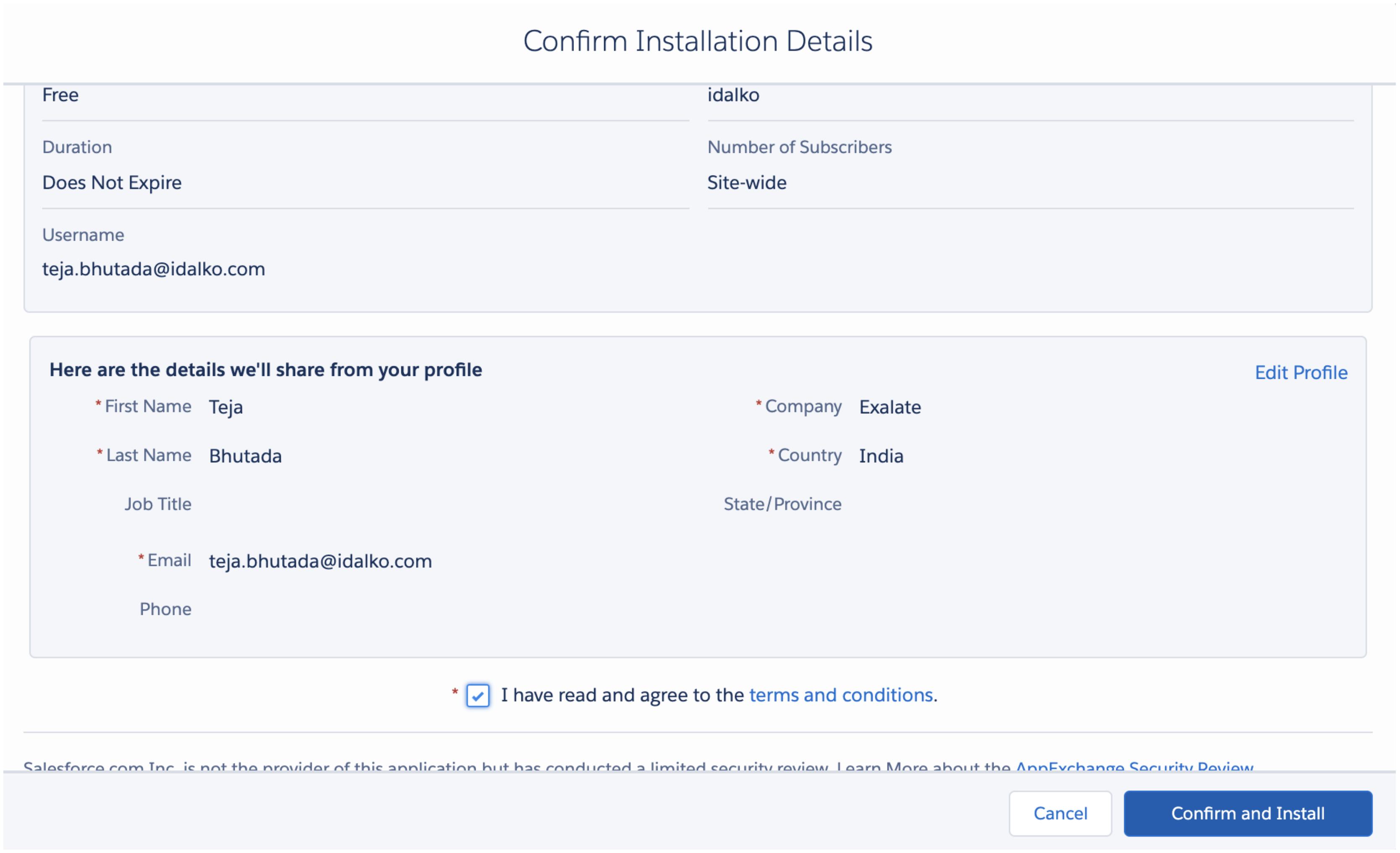
* Connected Salesforce Accounts: teja.bhutada@idalko.com

Don't see your account? More Info

Install in a Sandbox

Test this package in a copy of a production org.

A screen showing you the installation details will be displayed. Read on and agree to the "Terms and Conditions" and the bottom of the page and hit "Confirm and Install".



Free	idalko
Duration	Number of Subscribers
Does Not Expire	Site-wide
Username	
teja.bhutada@idalko.com	

Here are the details we'll share from your profile

* First Name	Teja	* Company	Exalate
* Last Name	Bhutada	* Country	India
Job Title		State/Province	
* Email	teja.bhutada@idalko.com		
Phone			

* ✓ I have read and agree to the terms and conditions.

Salesforce.com Inc. is not the provider of this application but has conducted a limited security review. Learn More about the AppExchange Security Review

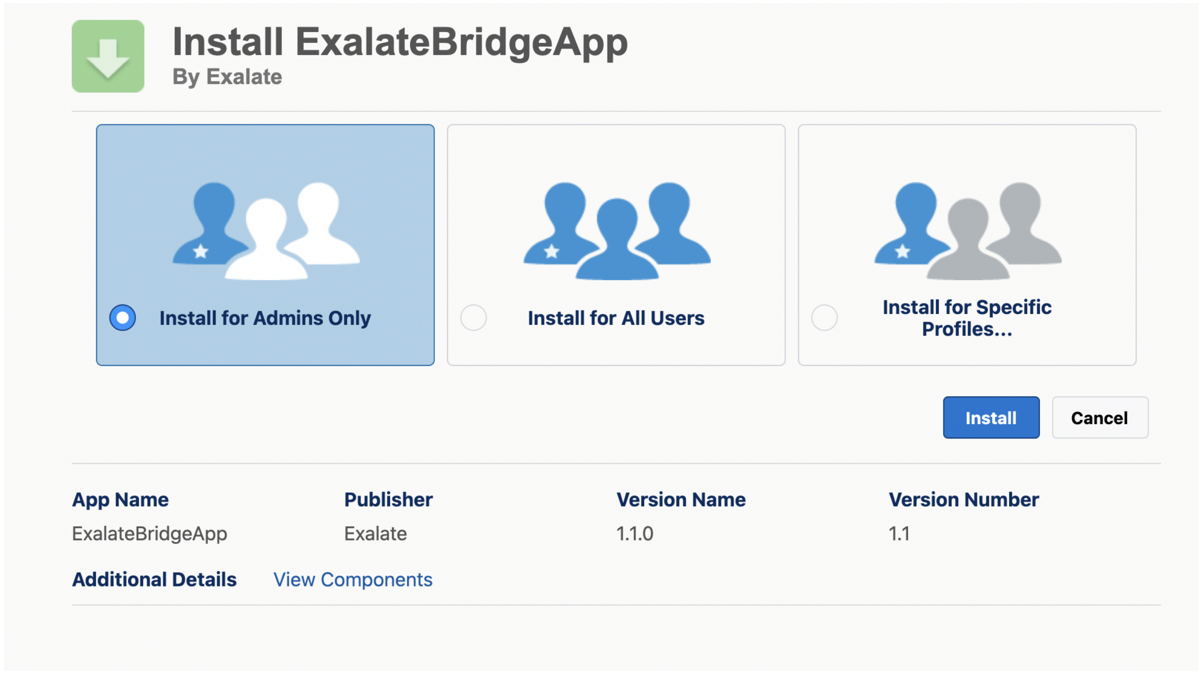
Next, you will be redirected to the Salesforce login screen, so "Log In" is there.

Then you need to select the users that will have permission to use the "Exalate" app. It can either be for admins, for all users, or only for specific profiles. Choose one option depending on your organizational needs.

If you choose to install it for specific profiles, then a list will appear with the available profiles and their permissions that you can set before you proceed.

If you are not sure about the permissions right now, you can change them later too.

Once you have made your decision, click "Install".



Install ExalateBridgeApp
By Exalate

[✓ Install for Admins Only] Install for All Users Install for Specific Profiles...

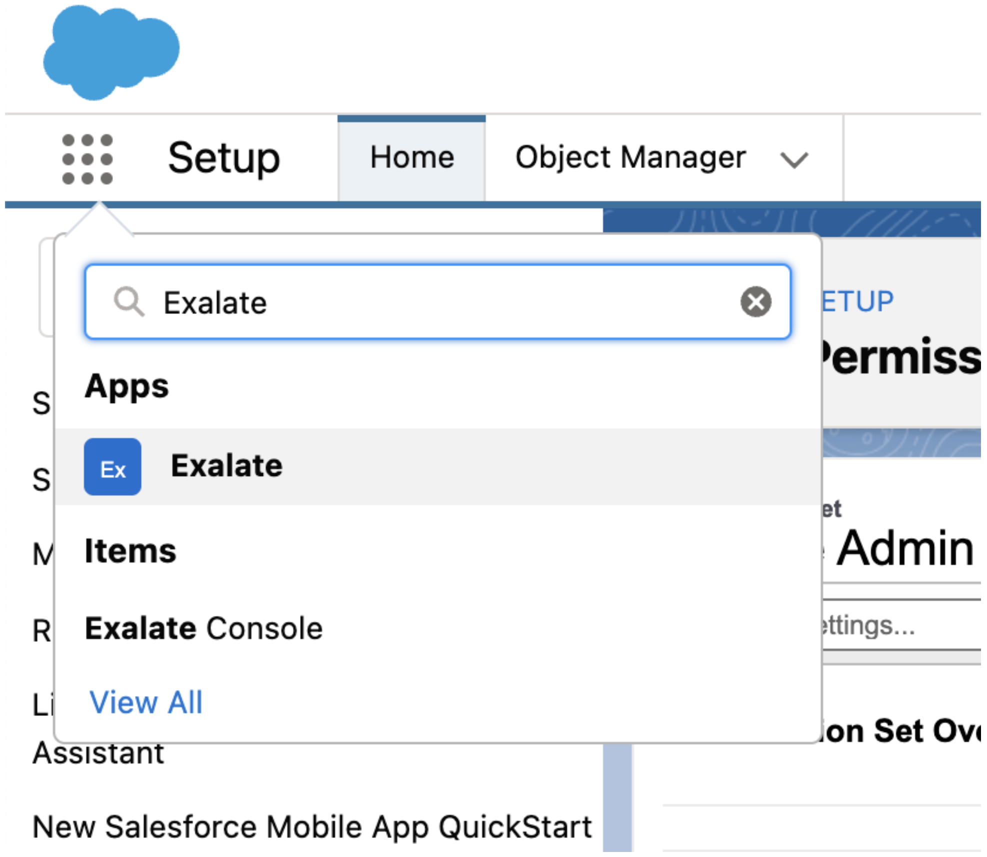
App Name	Publisher	Version Name	Version Number
ExalateBridgeApp	Exalate	1.1.0	1.1

Additional Details View Components

Then approve access to the required 3rd party websites and then click "Continue".

After a message of "Installation Complete!", click "Done" to go back to your Salesforce instance.

Now is the time you request access to the Exalate node. In your Salesforce instance, click on "Apps" at the top-left corner. Type "Exalate" there.



Then, click on "Request Node" and give necessary permissions to the app by clicking "Allow".

On the next form, fill in your contact details. Once you fill in the details click "Agree and Submit".

After which you will receive an email from Exalate that will verify your instance and activate an evaluation license.

So head over to your inbox. Open the email you may have received from the Exalate License Manager and click "Verify Exalate instance". Congratulations, you have successfully installed the app. You can start by initiating your first connection.

Step 3: Establish a Salesforce GitHub Connection

Now it’s time to set up your connection!

With Exalate, one side initiates the connection and the other side accepts it. Once the connection is established, you can proceed with configuring it to start sending information back and forth.

This is an important first step if you want to start synchronizing GitHub issues and Salesforce entities.

Say you want Cases to fire up new issues in GitHub. Or trigger dev issues/product updates directly from Salesforce to GitHub. Whatever the case is, we will see how to do that in the next few steps.

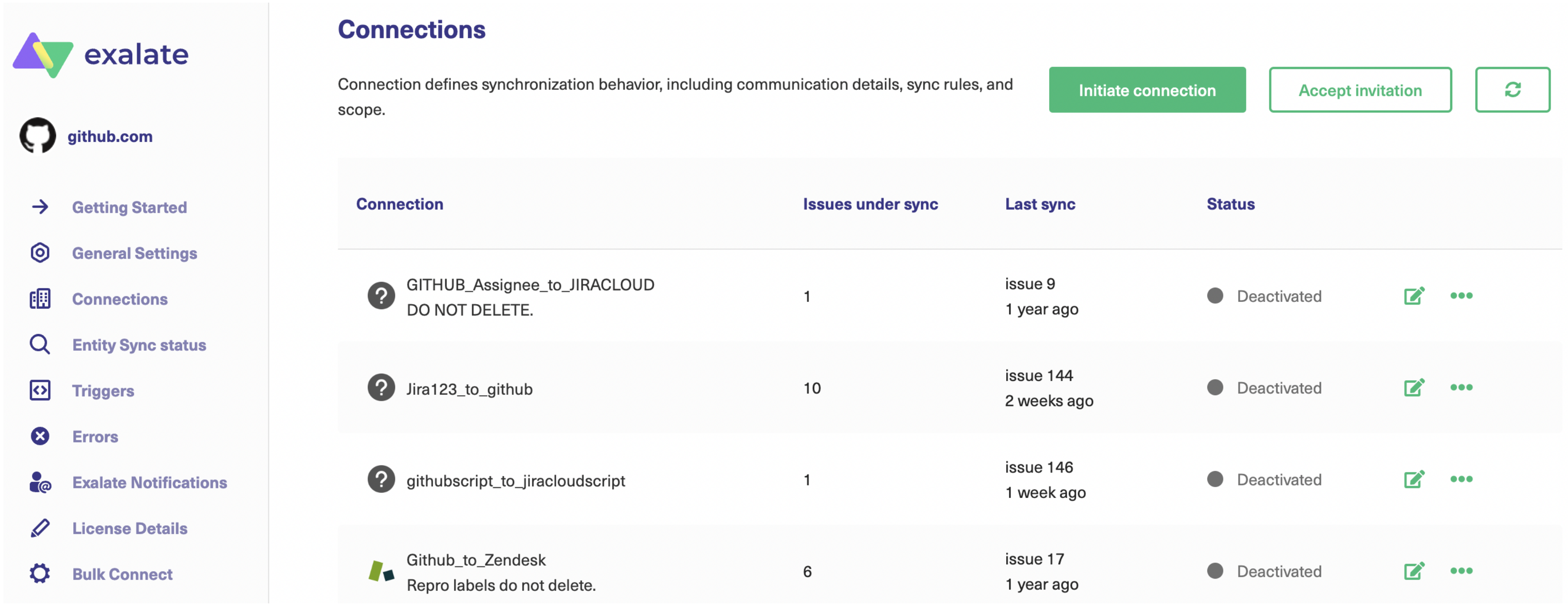
After installing the Exalate app, you should be redirected to a welcome screen.

From there, you navigate to the “Connections” tab in the left-hand menu. Connections in Exalate define how your synchronization behaves. It includes what information you want to send and receive and includes other configuration details.

With Exalate, you can initiate the connection from either the GitHub side or the Salesforce side. The UI for all the Exalate screens is the same, so it doesn't matter which side you start with.

For the purpose of our guide, we start with the GitHub side.

If this is your first time setting up a connection, the screen below should be blank, otherwise, it has all your connections listed. Click the green “Initiate Connection” button to start setting up the connection process.



Connections

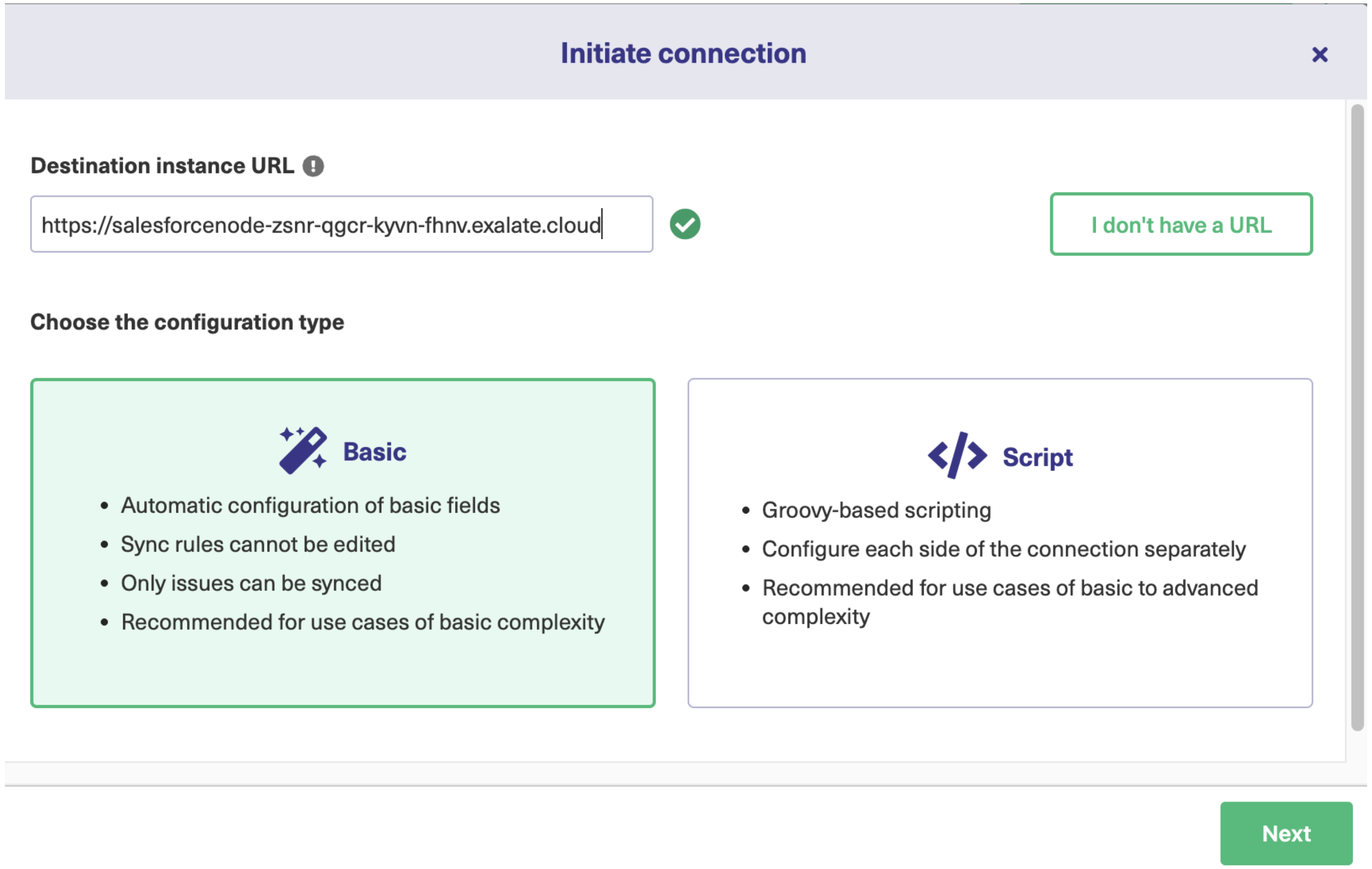
Connection defines synchronization behavior, including communication details, sync rules, and scope.

Connection	Issues under sync	Last sync	Status
GITHUB_Assignee_to_JIRACLOUD DO NOT DELETE.	1	issue 9 1 year ago	Deactivated
Jira123_to_github	10	issue 144 2 weeks ago	Deactivated
githubscript_to_jiracloudscript	1	issue 146 1 week ago	Deactivated
Github_to_Zendesk Repro labels do not delete.	6	issue 17 1 year ago	Deactivated

Then you will need to enter the destination URL. Here, it means your Salesforce URL. You can copy the URL, by going to the “Getting Started” on the left-hand Exalate menu.

After this, Exalate performs a quick check to see if it is installed on the destination instance or not. It will send appropriate messages while it does this.

After some time, more fields will appear asking you to choose the configuration mode you want for the connection.



Initiate connection

Destination instance URL

https://salesforcenode-zsnr-qgcr-kyvn-fhnv.exalate.cloud

Choose the configuration type

Basic
- Automatic configuration of basic fields
- Sync rules cannot be edited
- Only issues can be synced
- Recommended for use cases of basic complexity

Script
- Groovy-based scripting
- Configure each side of the connection separately
- Recommended for use cases of basic to advanced complexity

Exalate provides 2 such modes: the **Basic** mode and the **Script** mode.

The Basic mode has default mappings of different Salesforce entities and Github issues. These cannot be changed. It has an intuitive UI and you can start your synchronization immediately with this mode. It is suitable for use cases of basic complexity.

Note: *With the Free Plan, you can set up a connection in the Basic mode allowing you up to 1000 free syncs per month. You can get started here.*

The Script mode has advanced configurations, customizable mappings, and features. With this mode, you can change the existing mappings and configure new ones and control what and how you want the information to be passed to the other side. It allows you to sync almost any Salesforce entity with GitHub issues, the way you want to. This kind of customization allows you to adapt Exalate to use cases of advanced complexity.

We recommend you try this mode out since it allows you to handle your synchronization the way you want and use the full functionality of Exalate.

You can choose to upgrade the Basic connection in Salesforce or in Github and move to the script mode anytime you want.

We will have a look at both modes one by one.

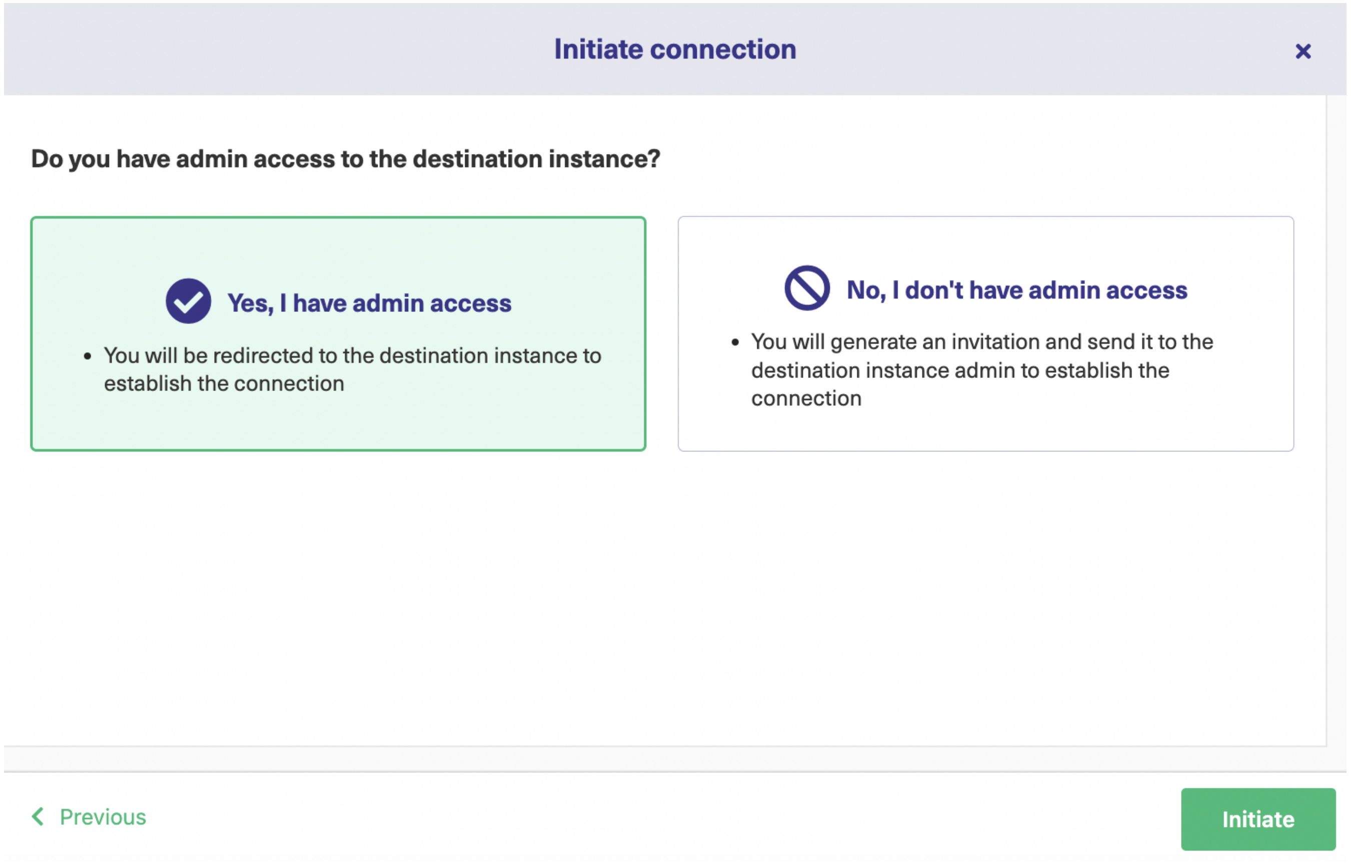
Continue with the Basic mode

Once you select “Basic” on the screen above, click “Next”.

You need to select the repository whose issues you want to sync on the Salesforce side. Select one from the dropdown list. Click “Next” when you are ready.

You will then be asked to verify admin access to the destination instance. In our case that’s Salesforce. Click “Yes, I have admin access” if you have it, otherwise click on “No, I don't have admin access”. The latter approach will generate an invitation code that you will need to copy and paste on the other instance.

Since I have access, I click on yes and hit “Initiate”.


Initiate connection

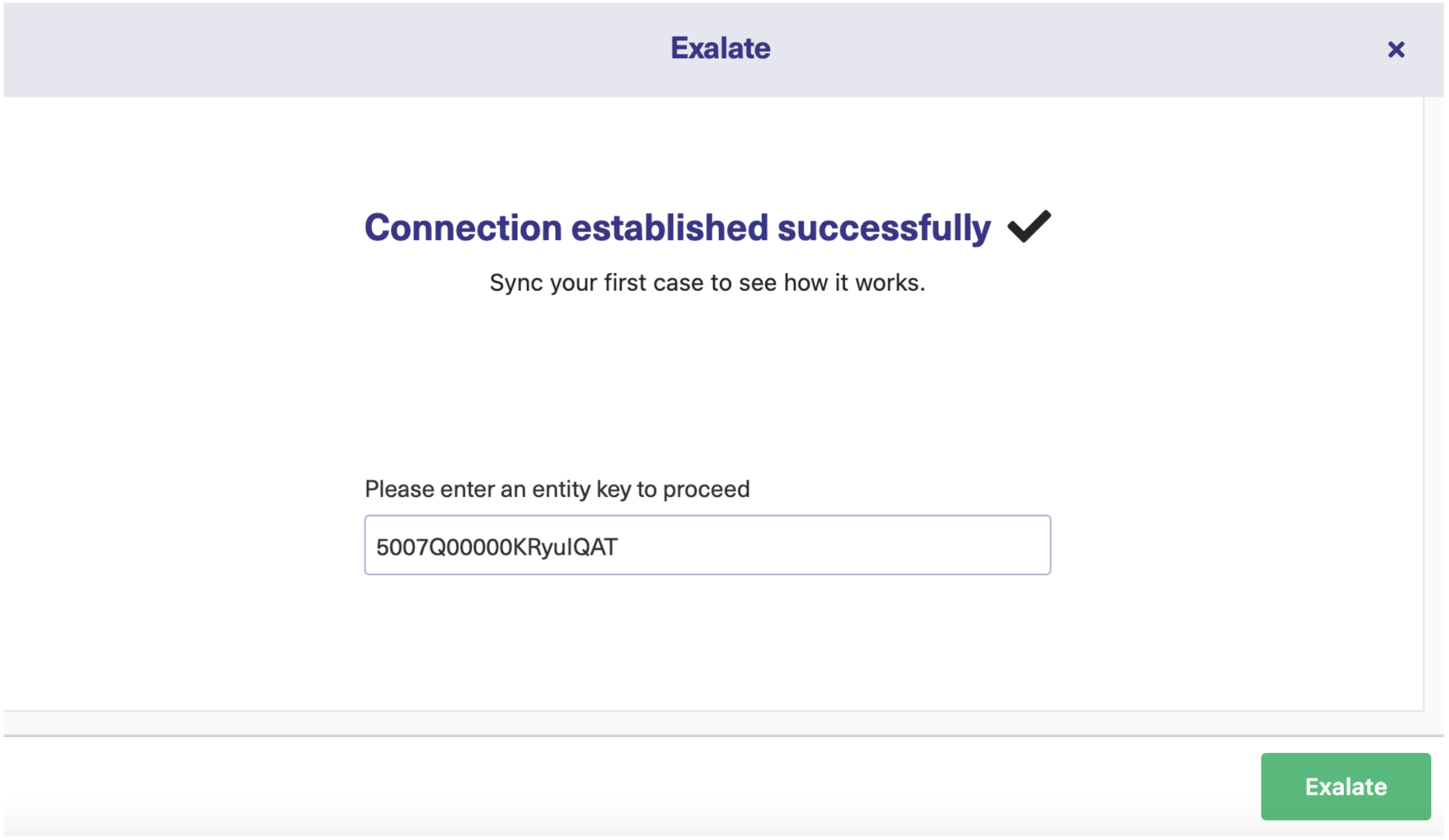
Do you have admin access to the destination instance?

[✓ **Yes, I have admin access**]
- You will be redirected to the destination instance to establish the connection

No, I don't have admin access
- You will generate an invitation and send it to the destination instance admin to establish the connection

After a quick verification, you will be redirected to the destination side (Salesforce).

The Basic connection allows you to start synchronization immediately. You can enter the Case key you want to sync, hit "Exalate", and you are good to go.



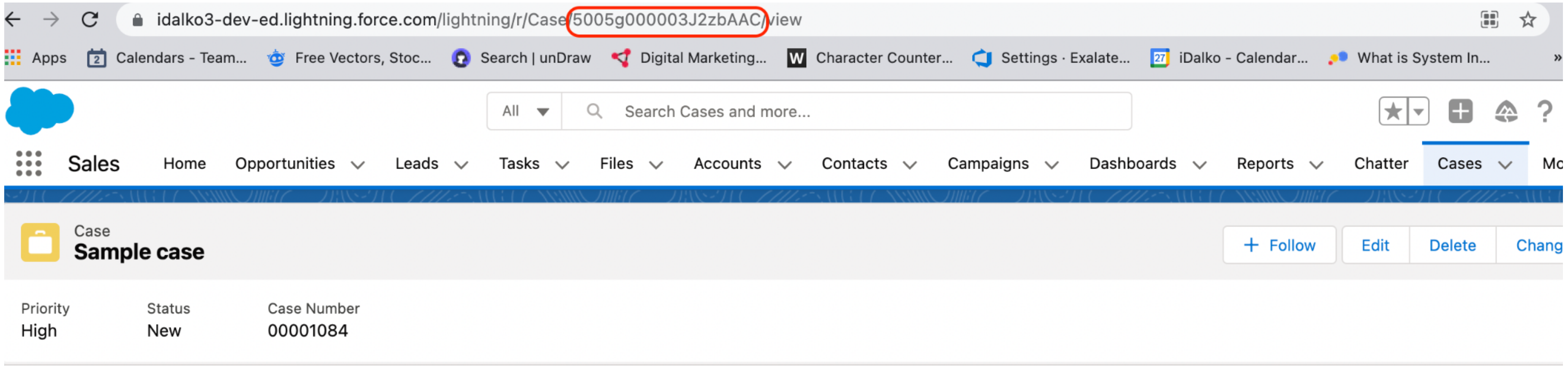
Connection established successfully ✓

Sync your first case to see how it works.

Please enter an entity key to proceed

5007Q00000KRyuIQAT

Note: *The Case number entered above is fetched from the URL of the Case you want to sync as shown in the image below.*


idalko3-dev-ed.lightning.force.com/lightning/r/Case/5005g000003J2zbAAC/view

Case
Sample case

Priority	Status	Case Number
High	New	00001084

On the GitHub side, the same Exalate screen will prompt you to enter the issue key.

Note: *On the GitHub side, the same screens will be displayed, but will be specific to issues.*

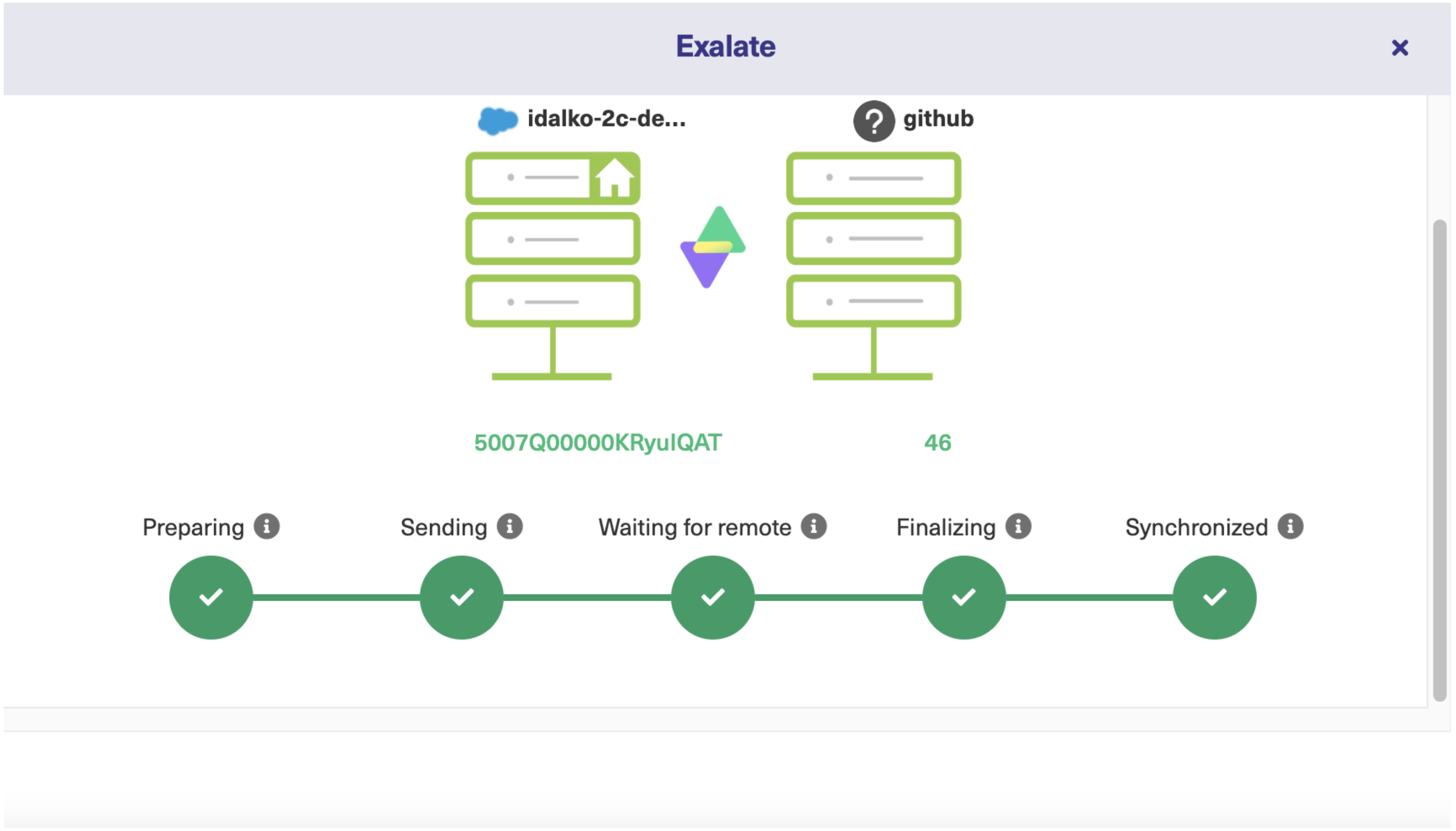
In case you don't want to "Exalate" your Case immediately, you can choose to press the close button on the screen above and use the trigger option to create automatic syncs to the other side or use the "Bulk Connect" option to sync Cases in bulk.

We will cover how to create triggers in step 5. Meanwhile, you can learn more about "Bulk Connect".

Once you Exalate the Case, it takes a while for it to be synced.

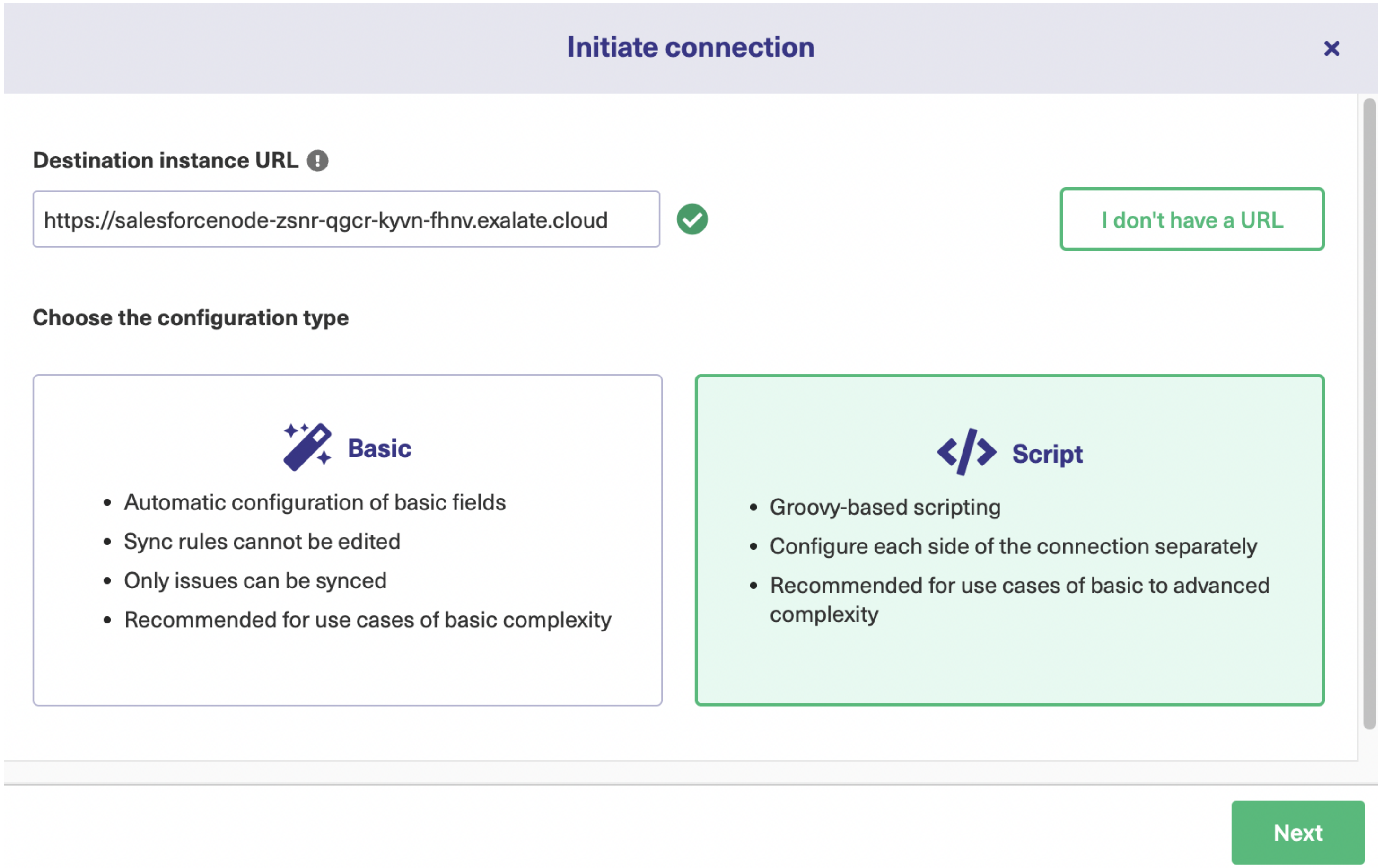
And finally! Your first Case has been synced. Subject, comments, status, and description for this Case can automatically be synced.

To go to the remote issue created on the GitHub side, click on the link generated for the issue (8 in our case). To refer to the Salesforce Case, click on the link below it.



Continue with the Script mode

Moving to the Script mode, press “Next” on the screen after you select “Script”.



Initiate connection

Destination instance URL

https://salesforcenode-zsnr-qgcr-kyvn-fhnv.exalate.cloud

I don't have a URL

Choose the configuration type

Basic

- Automatic configuration of basic fields
- Sync rules cannot be edited
- Only issues can be synced
- Recommended for use cases of basic complexity

Script

- Groovy-based scripting
- Configure each side of the connection separately
- Recommended for use cases of basic to advanced complexity

Next

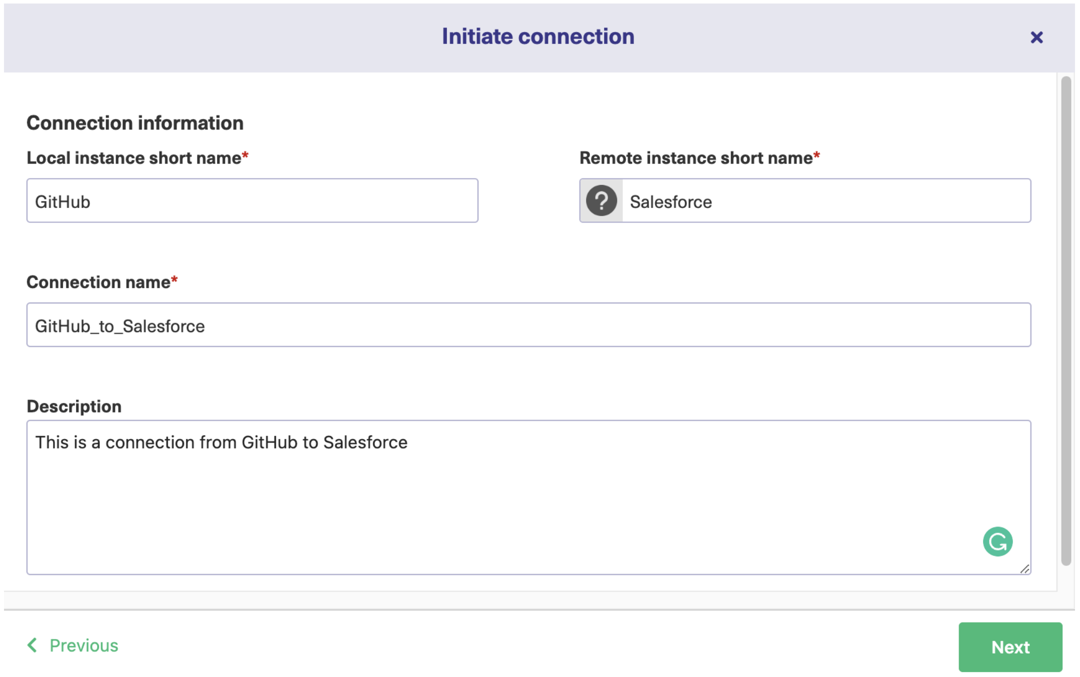
You now need to name the connection.

For this, enter the local instance name (GitHub). This is the side you initiate the connection from. Then enter the remote instance name (Salesforce). This is the destination side name.

Exalate then generates an automatic connection name for you. But you can edit it if you want.

Give your connection a description as well. This will be helpful when you have a list of them.

After you are done finalizing all this, click “Next”.



Connection information

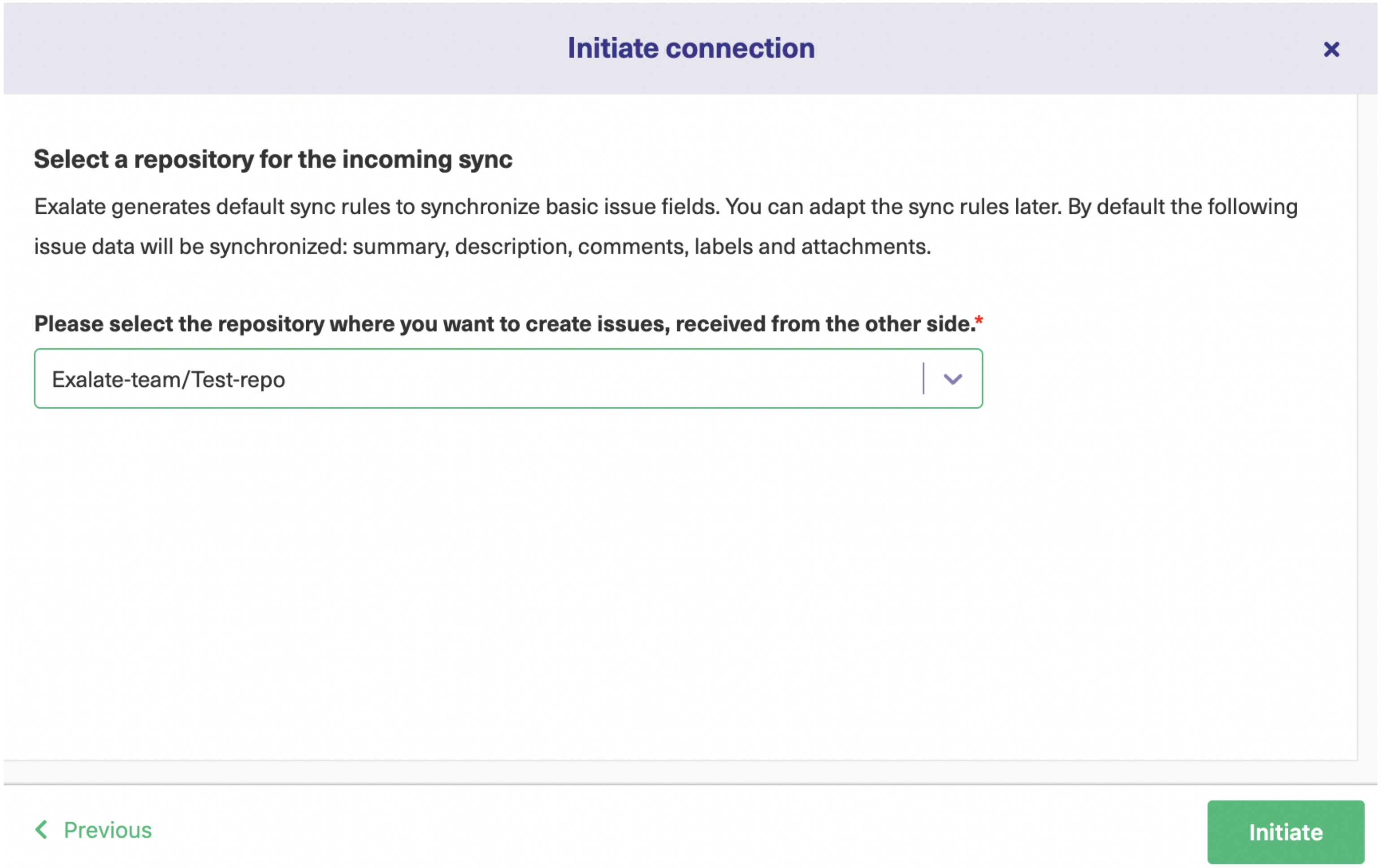
Local instance short name*: GitHub

Remote instance short name*: Salesforce

Connection name*: GitHub_to_Salesforce

Description: This is a connection from GitHub to Salesforce

Choose the repository you need to sync issues from on the GitHub instance. Select the correct one from the drop-down list. Click "Initiate".



Select a repository for the incoming sync

Exalate generates default sync rules to synchronize basic issue fields. You can adapt the sync rules later. By default the following issue data will be synchronized: summary, description, comments, labels and attachments.

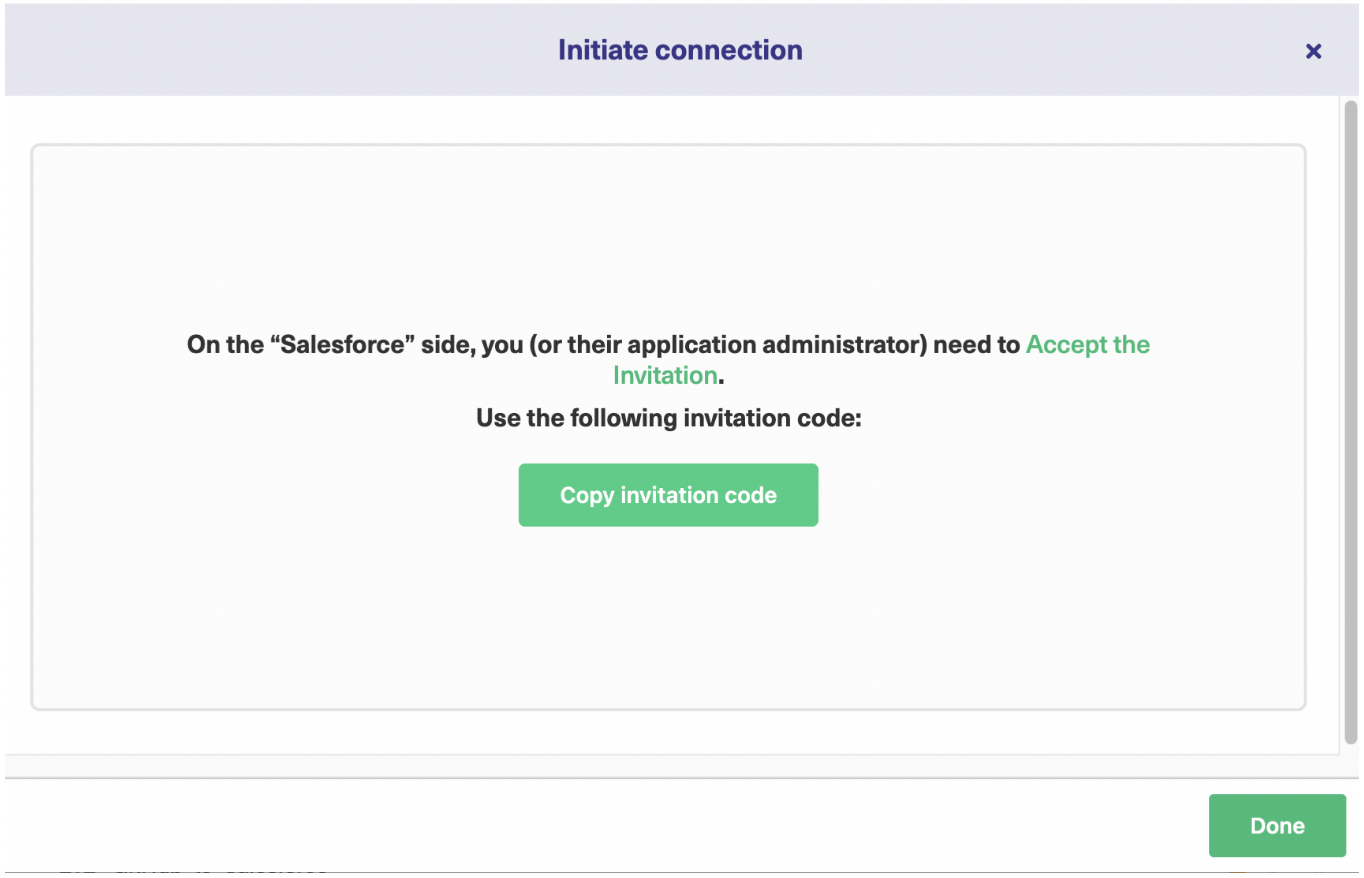
Please select the repository where you want to create issues, received from the other side.*

Exalate-team/Test-repo

Now an invitation code is generated.

This code is a unique secret that allows you to form a connection with the other side. You simply need to copy and paste it.

So click on “Copy invitation code” and save the copied code somewhere handy.

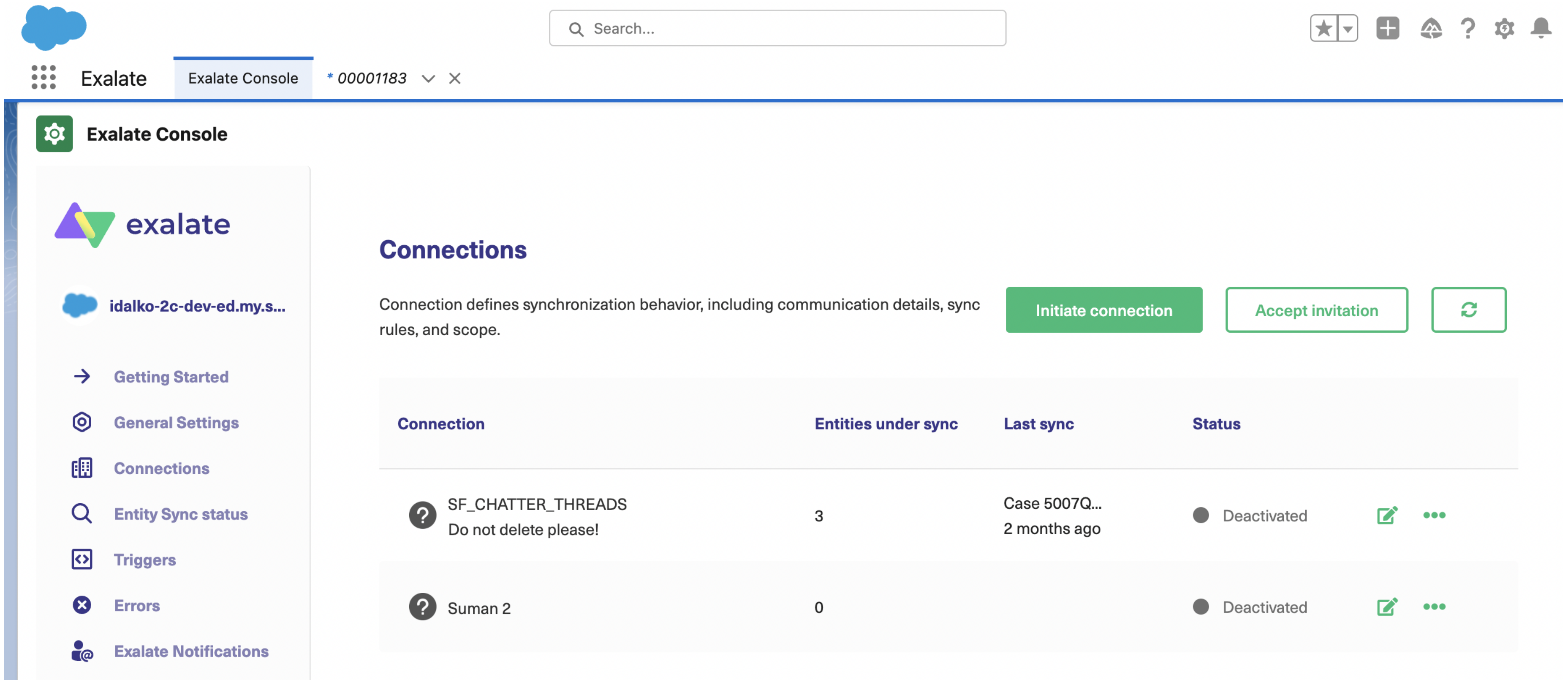

Initiate connection

On the “Salesforce” side, you (or their application administrator) need to Accept the Invitation.

Use the following invitation code:

Your work on the GitHub side is over for now. Then open up your Exalate console on Salesforce.

And navigate to the “Connections” tab on the left side and click on the “Accept Invitation” button.



Connections

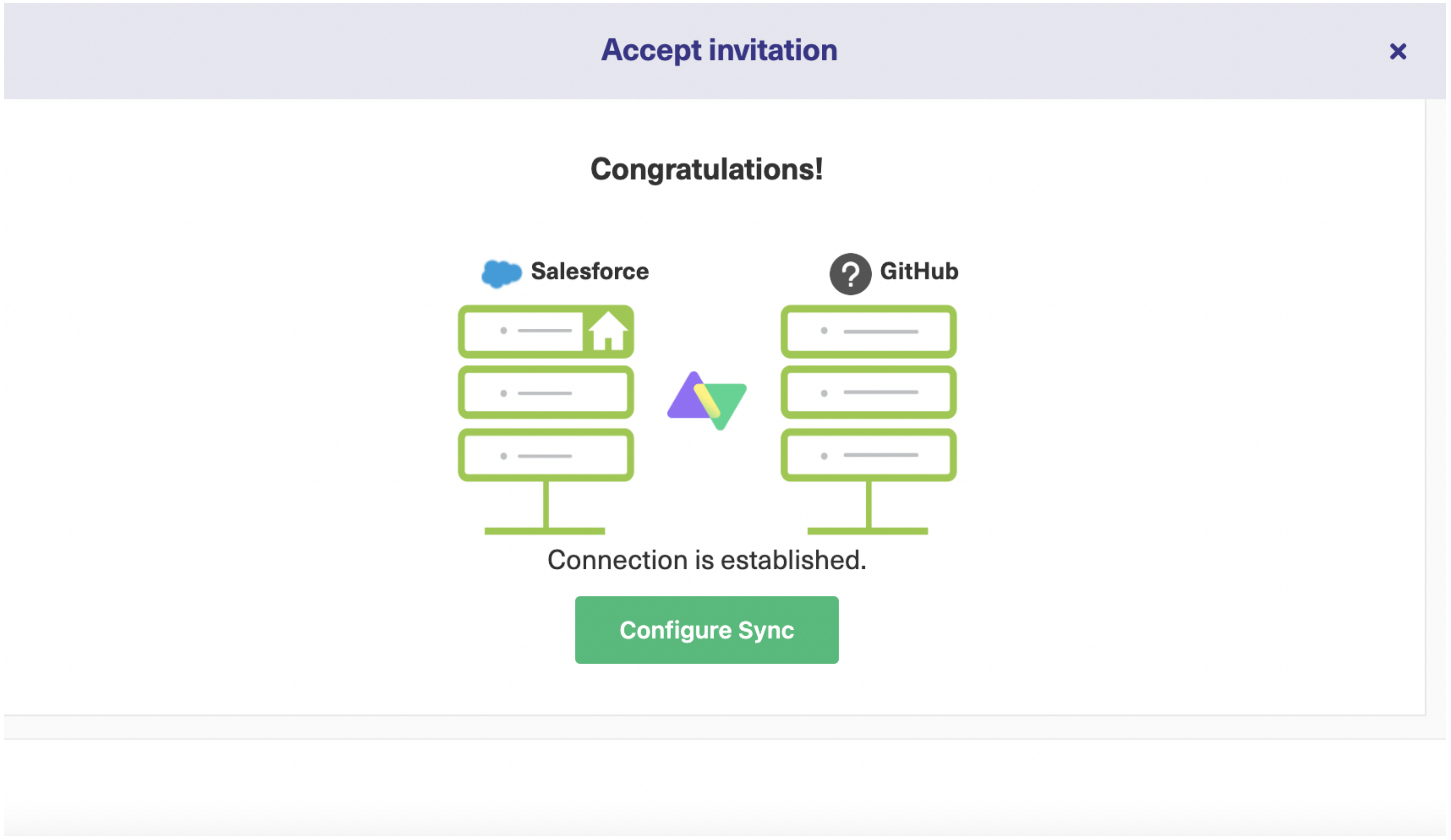
Connection defines synchronization behavior, including communication details, sync rules, and scope.

Connection	Entities under sync	Last sync	Status
SF_CHATTER_THREADS Do not delete please!	3	Case 5007Q... 2 months ago	Deactivated
Suman 2	0		Deactivated

The code you had copied, now needs to go in the text box. After pasting it, click “Next”.

Now your GitHub Salesforce connection has been successfully established. You can continue to configure your connection next.

This can be done in 2 ways: either by clicking the “Configure Sync” button shown on the screen below or by editing the connection to configure it. We will see how this is done in detail in step 4.


Accept invitation

Congratulations!

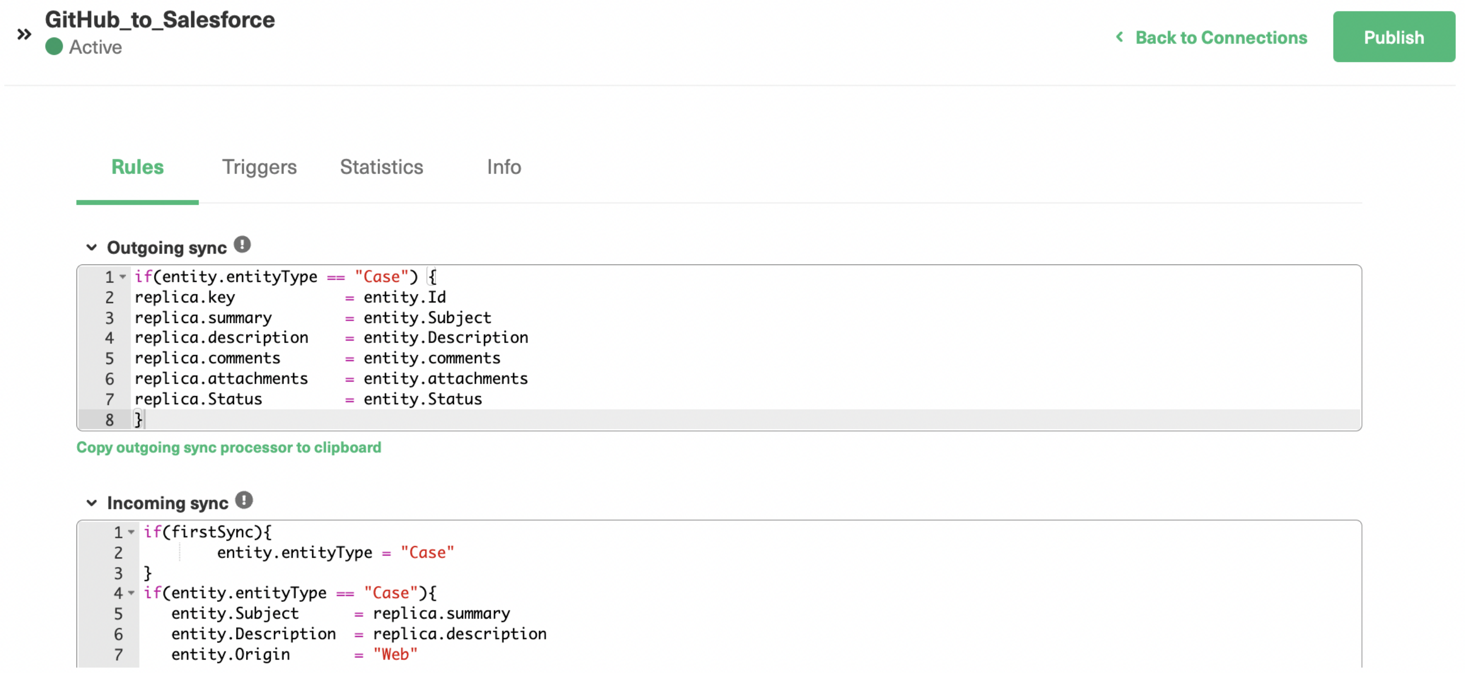
Connection is established.

Step 4: Configure the Connection to Decide What Information is Shared

If you have followed the steps sequentially, you will now be redirected to the following screen.

This screen allows you to configure the connection to control what information is shared with the other side and how you want to interpret the incoming information. Maybe you want to map the issue summary in GitHub as an internal comment under a Case in Salesforce or you may want to sync all open issues on GitHub to Cases in Salesforce.

All of this can be done by configuring the connection. We will get to a detailed explanation of this screen shortly.



```
if(entity.entityType == "Case") {
replica.key             = entity.Id
replica.summary         = entity.Subject
replica.description     = entity.Description
replica.comments        = entity.comments
replica.attachments     = entity.attachments
replica.Status          = entity.Status
}
```

```
if(firstSync){
        entity.entityType = "Case"
}
if(entity.entityType == "Case"){
    entity.Subject      = replica.summary
    entity.Description  = replica.description
    entity.Origin       = "Web"
```

BOOK DEMO

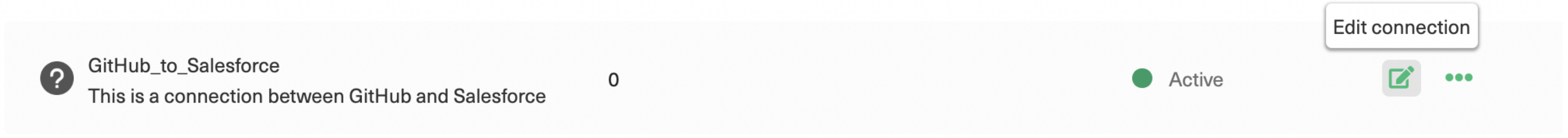
Meanwhile, as I mentioned earlier, you can configure the connection by clicking on the edit connection button as well. The connection you created earlier can be viewed under the “Connections” tab.

This tab specifies all the connections you have created as a list, with different options for each connection given next to their names. Every connection has a status that shows the connection being “Active”, “Pending”, “Error” or “Deactivated”.

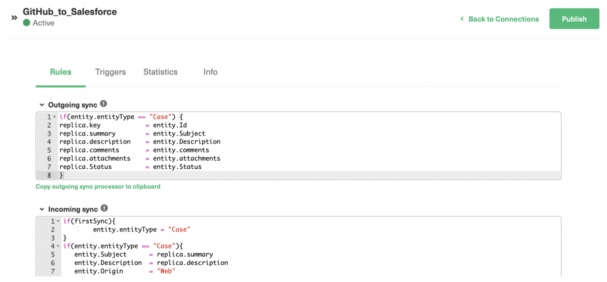
You can also see the number of issues under sync and the last synced time here.

The edit connection button shown in the image below allows you to configure the connection

There is a remote antenna button that takes you to the other side. Clicking the 3 dots allows you to activate, deactivate or delete the connection. You can even “Exalate” a single issue or a Case here.



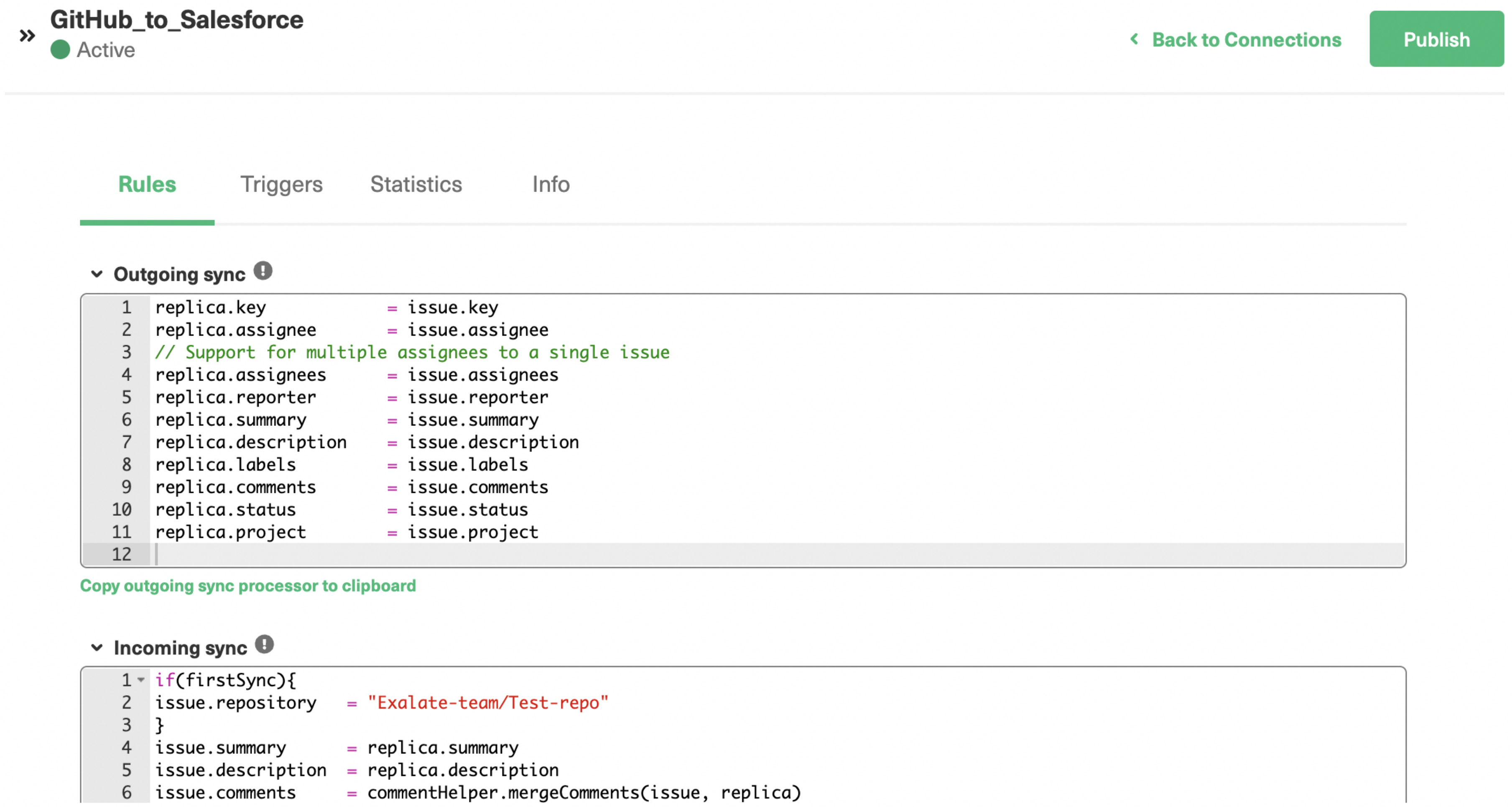
Let's move back to our configuration screen. Just so you are not lost, this is how it looks on the Salesforce side.



```
if(entity.entityType == "Case") {
replica.key            = entity.Id
replica.summary        = entity.Subject
replica.description    = entity.Description
replica.comments       = entity.comments
replica.attachments    = entity.attachments
replica.Status         = entity.Status
}
```

```
if(firstSync){
        entity.entityType = "Case"
}
if(entity.entityType == "Case"){
   entity.Subject       = replica.summary
   entity.Description   = replica.description
   entity.Origin        = "Web"
```

And the Github side



```
replica.key             = issue.key
replica.assignee        = issue.assignee
// Support for multiple assignees to a single issue
replica.assignees       = issue.assignees
replica.reporter        = issue.reporter
replica.summary         = issue.summary
replica.description     = issue.description
replica.labels          = issue.labels
replica.comments        = issue.comments
replica.status          = issue.status
replica.project         = issue.project

```

```
if(firstSync){
issue.repository    = "Exalate-team/Test-repo"
}
issue.summary       = replica.summary
issue.description   = replica.description
issue.comments      = commentHelper.mergeComments(issue, replica)
```

As seen above, the UI is the same, just the scripts have changed. We will see what the scripts are in a minute.

The screen has 4 tabs: “Rules”, “Triggers”, “Statistics” and “Info”.

We will cover “Triggers” in step 5.

The “Statistics” tab gives an overview of the synchronization information about the number of issues under synchronization, the last sync date, etc.

The “Info” tab provides a general description of the connection, stating its name, destination URL, description, and type.

The “Rules” tab has scripts written in Groovy Scripting language. These are sync rules that specify what information should be sent to the other side in the “Outgoing sync” rules, and map the information coming from the other side in the “Incoming sync” rules.

Both the outgoing and incoming sync rules are present on the Salesforce and GitHub sides as shown above.

You can alter the sync rules using scripts anytime you want. Alteration means editing the

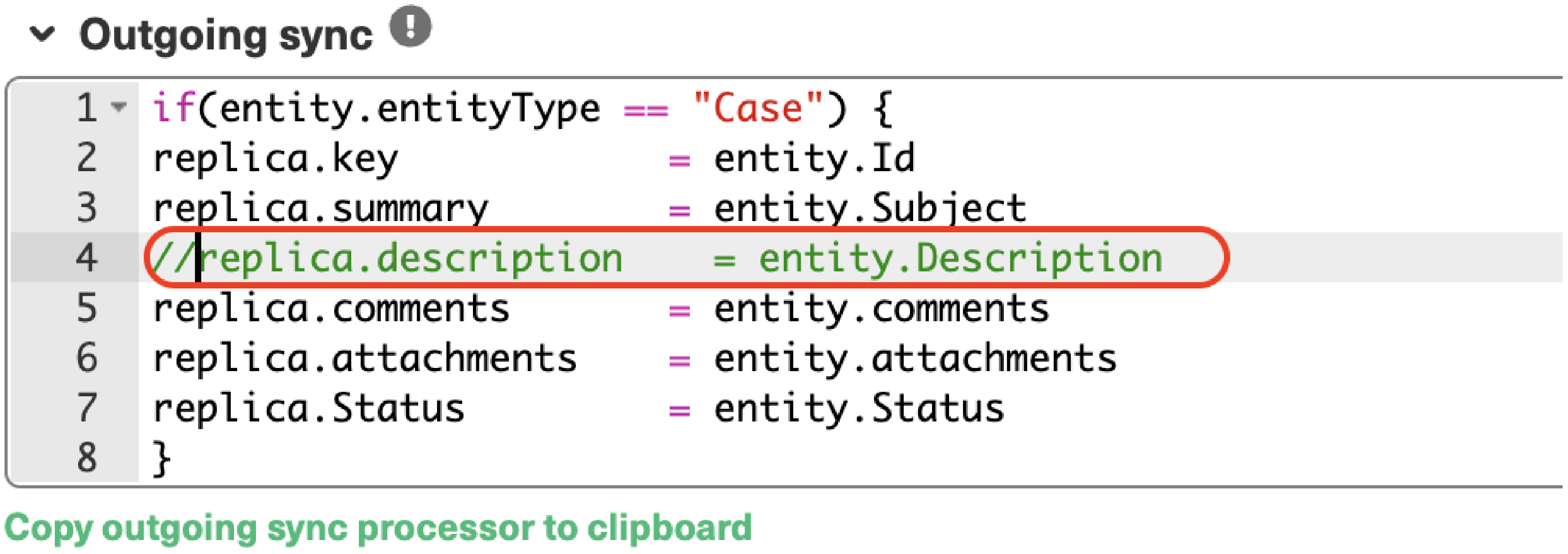
1. outgoing sync rules by
    1. not sending existing information anymore by deleting the rule
    2. sending new information by adding scripts

2. incoming sync rules by
    1. not receiving some information anymore by deleting the rule
    2. receiving new information by adding scripts

You can decide to not send or receive certain information by deleting or commenting on a particular line (read the rule) from the text box shown above.

Commenting means the particular line will be ignored at the time of synchronization. To comment, simply add a "//" at the start of the line. To remove multiple rules, add "/*" at the start of the block of lines and then "*/" whenever you want to end.

For instance, if you don't want to send a description of the Case from Salesforce to Github, you add "//" as follows:
//replica.description = entity.Description.



```
if(entity.entityType == "Case") {
replica.key                  = entity.Id
replica.summary              = entity.Subject
//replica.description        = entity.Description
replica.comments             = entity.comments
replica.attachments          = entity.attachments
replica.Status               = entity.Status
}
```

Sometimes you want to send or receive additional information. This is when you add new scripts in the outgoing and incoming syncs.

For instance, if you want to sync "Task" in Salesforce in addition to "Case", you add the following script in the outgoing sync box.

```
if(entity.entityType == "Task") {
    replica.key        = entity.Id
    replica.summary    = entity.Subject
    replica.description = entity.Description
}
```

There are countless possibilities in the way you can edit these script rules to suit your specific use case. So experiment as much as you want!

You can always refer to Exalate docs for GitHub and Salesforce to learn more about how to work with them.

Move on to creating triggers.

Step 5: Set up Automatic Synchronization Triggers to Share Information

Rules define what you want to send to or receive from the other side, but triggers define how you want to start the synchronization process.

They allow you to specify under what conditions you want the synchronization to happen.

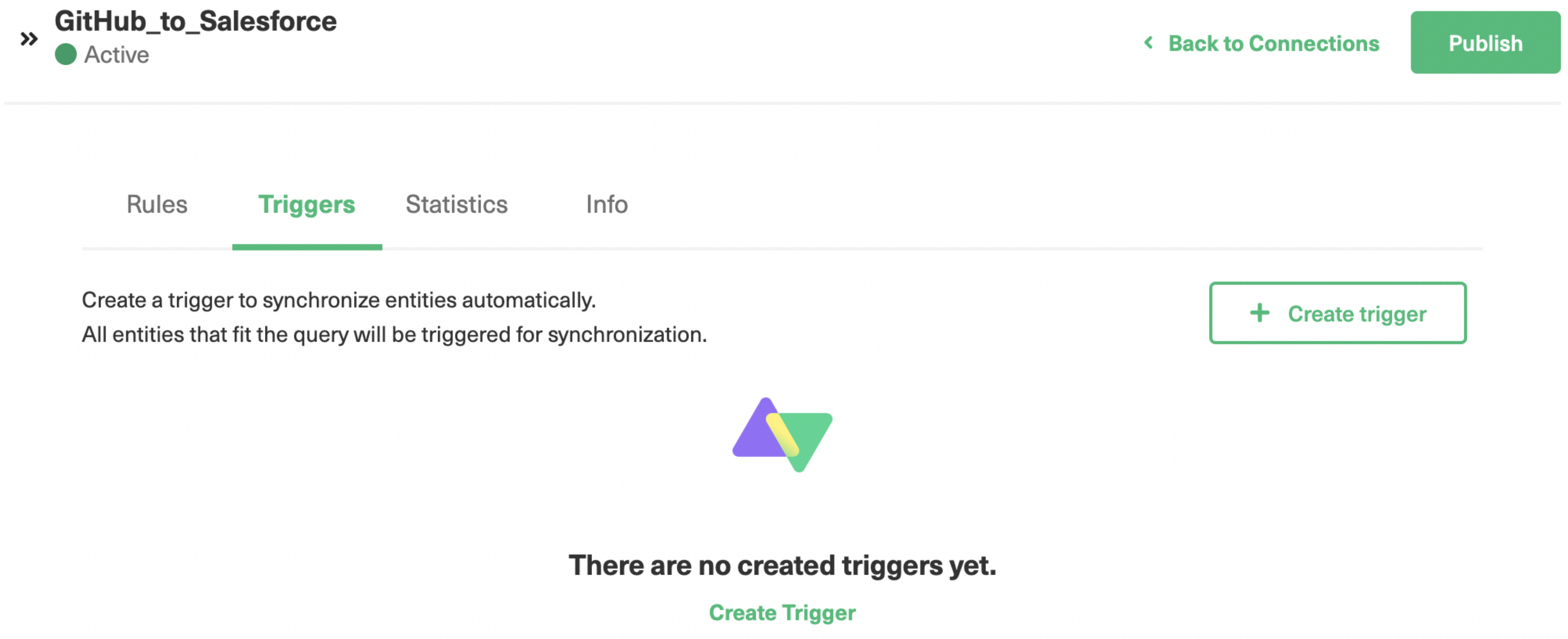
For instance, when the sales team logs a feature request raised by an important customer under a Case in Salesforce, an issue is automatically created in GitHub.

Such and many other conditions can be set under the search query, and if the condition is met then the trigger automatically starts synchronization based on the outgoing and incoming sync rules you have set.

For this, click the “Triggers” tab.

This tab shows a list of all the triggers you have created.

If this is your first time, the list is empty. Click “Create trigger”.



Create a trigger to synchronize entities automatically.
All entities that fit the query will be triggered for synchronization.

There are no created triggers yet.

An “Add trigger” screen pops up.

***Note:** You can also create triggers by clicking on the “Triggers” tab in the left-hand Exalate menu. When you choose this option, you need to additionally choose the connection name you want to execute the trigger for.*

The “Add trigger” screen first asks you to select an entity type. I have selected an “issue” type here since the screen is for GitHub. We will see how the Salesforce screen looks, shortly.

Under the “If” section, enter the search query.

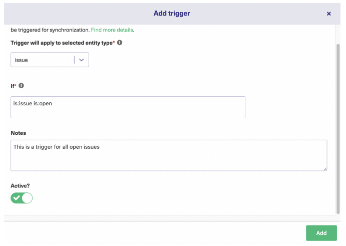
These search queries are platform-specific. Here, we are syncing all the issues with the status as open.

To know more about GitHub search queries visit this page. For Salesforce, Exalate uses the SOQL (Salesforce Object Query Language) syntax. Learn more about it here.

Add appropriate notes to help you understand why you have created this trigger.

Toggle the “Active” switch to activate or deactivate the trigger. This saves your time when you don’t want to create a trigger from scratch but don’t need to use it right now.

Click “Add” once you are done.



be triggered for synchronization. Find more details.

Trigger will apply to selected entity type*: issue

If*: is:issue is:open

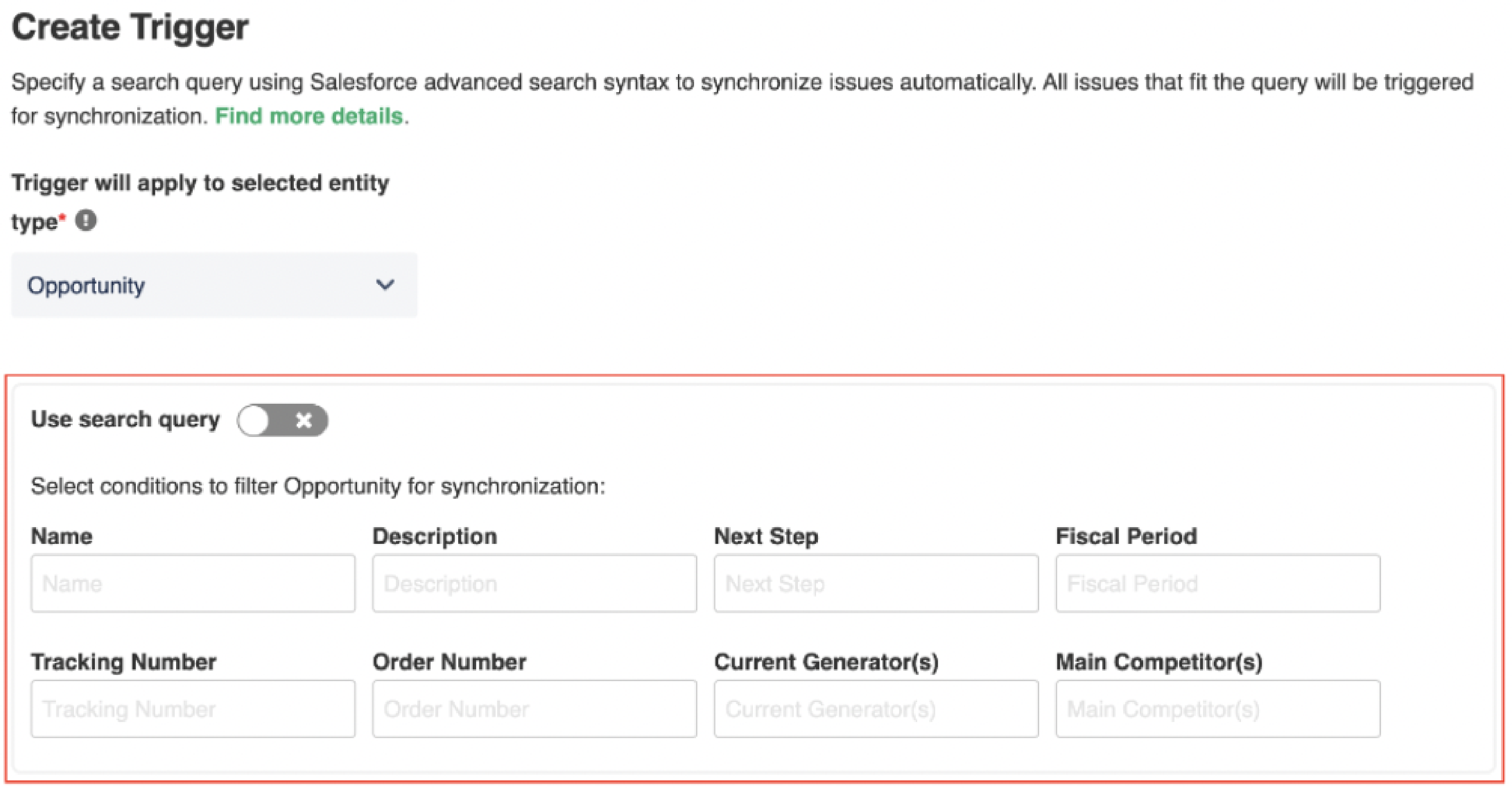
Notes: This is a trigger for all open issues

Active? ✓

On the Salesforce side, let's have a glimpse of how the add triggers screen looks like.

There are many entity types supported by Exalate. But the most popular ones are Opportunity, Product, Account, Case, and Task.

After you select Opportunity as the Salesforce entity type, you select conditions to filter entities for setting the trigger.



Create Trigger

Specify a search query using Salesforce advanced search syntax to synchronize issues automatically. All issues that fit the query will be triggered for synchronization. Find more details.

Trigger will apply to selected entity type*

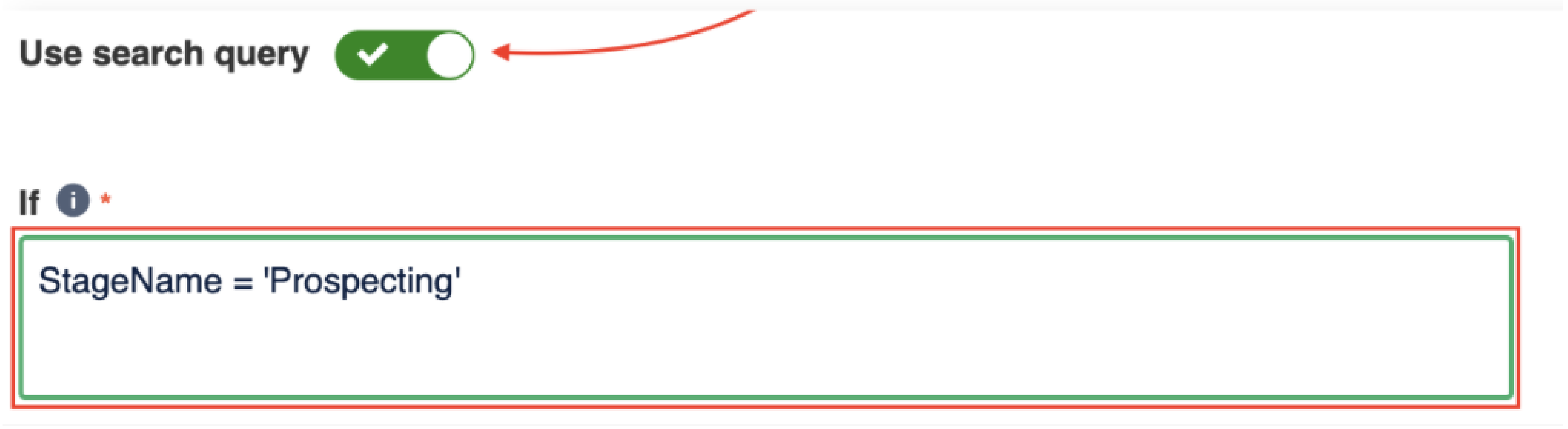
Opportunity

Use search query

Select conditions to filter Opportunity for synchronization:

Name	**Description**	**Next Step**	**Fiscal Period**
Name	Description	Next Step	Fiscal Period
Tracking Number	**Order Number**	**Current Generator(s)**	**Main Competitor(s)**
Tracking Number	Order Number	Current Generator(s)	Main Competitor(s)

You can also enter the search query with the toggle button as shown below.

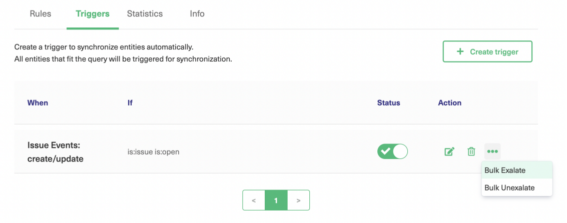


StageName = 'Prospecting'

You can learn more about how this is done on the Salesforce and the GitHub side if you want.

You can see the created trigger listed as follows. You can choose to edit or delete the trigger from here.

By clicking the 3 dots next to the trigger you can choose to “Bulk Exalate” all the current issues or entities fitting the search criteria.



Create a trigger to synchronize entities automatically.
All entities that fit the query will be triggered for synchronization.

When	If	Status	Action
Issue Events: create/update	is:issue is:open	✓	

Don’t forget to click “Publish” when you have finished making the changes.

Common Use Cases

The most plausible use case that can be supported by a GitHub Salesforce integration is enabling seamless collaboration between sales and development teams.

Sales teams get a lot of customer queries, feedback, and new feature requests. An integration solution like Exalate can allow them to fire these customer concerns into GitHub right from within Salesforce. This will create an issue in GitHub. The issue progress and the appropriate status updates can then be viewed by the sales team from the Salesforce UI. And they are in a better position to update their customers on the status of the issue.

Important product updates or feature request issues undertaken by the development teams in GitHub can be automated and synchronized with Salesforce entities. So the sales team always has the most recent and up-to-date product information ready to be taken up with their customers who have been waiting for exactly that update or feature.

Conclusion

We saw how a GitHub Salesforce integration can definitely help sales and development teams collaborate and share business-critical information with each other to help them enhance customer experience.

We also discussed how choosing the right technology for your integration can manifold the benefits such integration can bring along. Every organization has its own drivers for choosing an integration tool, but we chose Exalate since it checks most of our criteria.

And finally, we covered a step-by-step approach to implementing a GitHub Salesforce integration to ensure that the integration is handled systematically.

Recommended Reads:

- How to set up a Jira Salesforce Integration
- Salesforce to Salesforce Integration: Sync Multiple Salesforce Instances Bidirectionally
- How to Set up a Jira GitHub Integration
- Salesforce Zendesk Integration
- How to Set up a ServiceNow GitHub Integration
- How to Set Up a Salesforce ServiceNow Integration
- How to Set up an Azure DevOps GitHub Integration
- How to Set up a Zendesk GitHub Integration