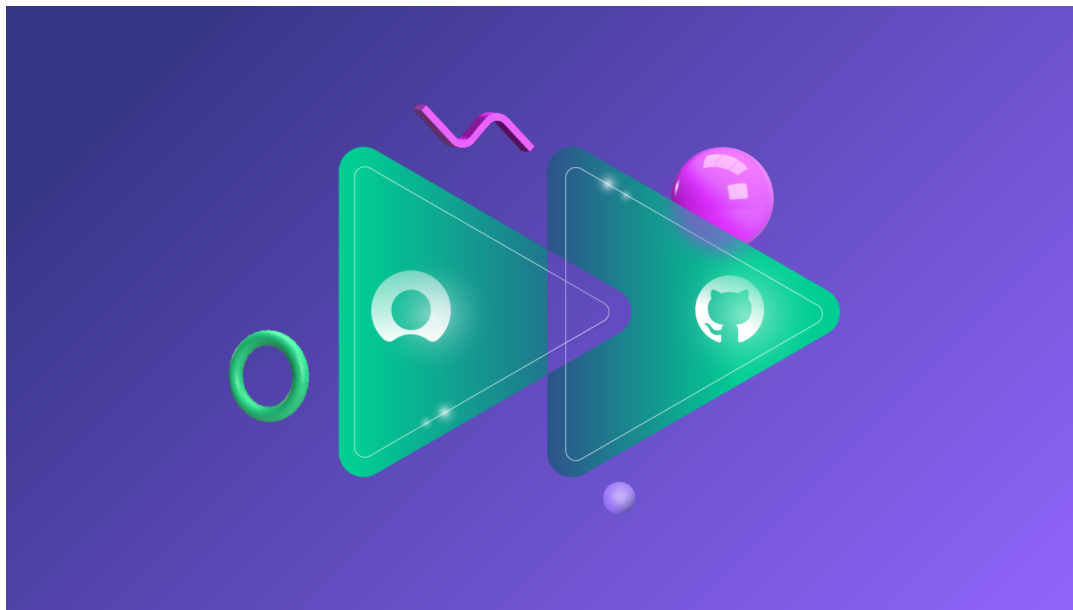




# How to Set up a ServiceNow GitHub Integration: The Comprehensive 2023 Guide



## Table of contents

### Why Integrate ServiceNow and GitHub

- Choosing the Right Technology For a ServiceNow GitHub Integration

### How to Set up a ServiceNow GitHub Integration in Six Steps

- Step 1 - Install Exalate on ServiceNow
- Step 2 - Install Exalate on GitHub
- Step 3 - Connect Your ServiceNow and GitHub Instances
- Step 4 - Configure Your Connection to Determine What Information Gets Shared
- Step 5 - Set Up Automated Synchronization Triggers
- Step 6 - Start Synchronizing Tasks

### Common Use Cases

- Developers and Support Team
- Product Development and Quality Control

### Conclusion



If you use platforms like ServiceNow and GitHub to organize your teams, you may want to consider a ServiceNow GitHub integration at some point along the way. Connecting these two platforms will simply mean easier and more accurate data sharing, as with integrations you can do that automatically and save everyone a lot of work.

Automatically filtering and sorting the many issues and tickets you use sounds like a difficult challenge. The right integration solution though can make it much easier. So in a few simple steps, you'll see how to set up a ServiceNow GitHub integration with the least fuss possible.

So let's get started!

Here's what we're going to cover in this blog post:

- [Why Integrate ServiceNow and GitHub](#)
- [How to set up a ServiceNow GitHub Integration in Six Steps](#)
- [Common Use Cases](#)

## Why Integrate ServiceNow and GitHub

ServiceNow is a workflow focussed platform able to handle everything from service management to help desk support. It offers a wide selection of web-based tools and these are complemented by many extensions.

**servicenow**<sup>™</sup>

With a business model focused on large teams and organizations, it is good for keeping track of large amounts of information and handling complex business relationships.

On the other hand, GitHub is a code storage platform enabling developers to easily handle version control and code distribution. It allows you to create and discuss issues, making it easier to manage projects.



It is aimed at open-source projects and is a great way to build a community of coders working together to help improve products.

These platforms contain huge amounts of information and allow teams to work together efficiently. So integrating them properly can help teams share data and manage workflows collectively while keeping the benefits of the individual platforms without jeopardizing either side's autonomy or security.

## Choosing the Right Technology For a ServiceNow GitHub Integration

Your teams need a solution that is **reliable** enough to handle outages and errors, **flexible** enough to adapt to their changing needs, and allows them to choose how they want to share information with the other side.

The right solution should also guarantee your team's **integration is decentralized** i.e each side has independent control over what data needs to be shared and with whom, so they can keep working in the comfort of their own environment.

The tool we've chosen for this guide is called [Exalate](#). It was designed with these criteria in mind. Let's see how to use it in practice.

## How to Set up a ServiceNow GitHub Integration in Six Steps

## Step 1 - Install Exalate on ServiceNow

First of all, you need to install Exalate. This can be done on your ServiceNow instance with docker, or you can ask Exalate to set up a separate node for you. I'll use the second option in this tutorial.

To learn more about both options, take a look at [this documentation](#).



To get a node, go to [Exalate's integrations page](#), and click ServiceNow.

### Reserve an Exalate for ServiceNow evaluation instance ✕

The Exalate for ServiceNow instance will allow you to setup a fully functional integration between your ServiceNow environment and any other supported tracking platform.

**Enter Organisation\***

**Enter First name**

**Enter Last name\***

**Enter Phone number\***

**Enter Email address\***

A form will appear. Fill this in to request an evaluation instance. Enter your details and click “Submit”.

Hi,

Your Exalate for ServiceNow node is ready for configuration.

You will have to follow a couple of steps to start connecting your instance to other instances on your Exalate network

These are documented in the [installation steps](#)

### Your data

Your node can be accessed on

<https://snownode-francis-fake1.exalate.cloud>

Your evaluation key is

```
eyJsaWNlbnNlVmVyc2lvbiI6IjIuMCIsIm9yZ2FuaXNhdGlvbiI6IklEQUxLTyIsInN0YXJ0RGF0ZSI6Ikd5vdiA3LCAyMDE5IDewOjQ0OjI2IEFNiIiwib3Bwb3J0dW5pdHkiOiJFQVNFLTMzNzQiLCJjb25uZWNoaW9ucyI6MCwidXNlcGxhbiI6MCwic2lnbmF0dXJlIjoiJDJhJDEyJE9wNlFRUzRndDJFYs90MX1vL2xUQy5oaGQ2dkNHURaZnJ0dElmZlA1cVB3YWcwaXVebVFLIiwiaXNFdmFsdWV0aW9uIjp0cnVlfQ==
```

Having troubles, just let us know by sending a mail to

[support@exalate.com](mailto:support@exalate.com)

Enjoy,

The Exalate team

You'll get a message that your node is ready. You also need to create a ServiceNow account with the [appropriate permissions](#).

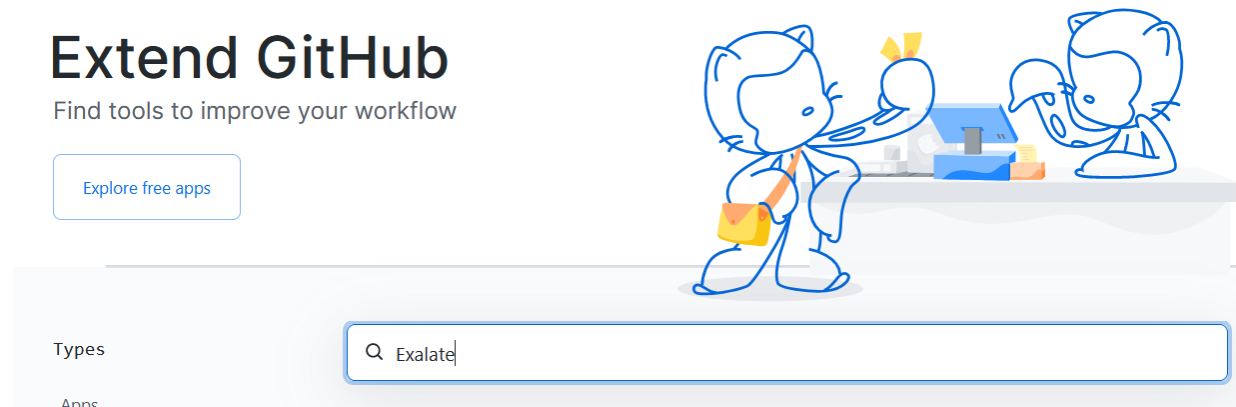
Once you've created the right user accounts, go to your Exalate node and click "general settings". Put in the URL of your ServiceNow instance, along with the username and password of your proxy user.

Now you should be able to access the Exalate console. You'll need to use an account with administrator permissions to do so.

## Step 2 - Install Exalate on GitHub

Next, you need to set Exalate up on GitHub. To do that, log in to your account, then click "Marketplace" on the top menu. Type "Exalate" into the search bar and press return. You'll see "Exalate Issue Sync" appear.

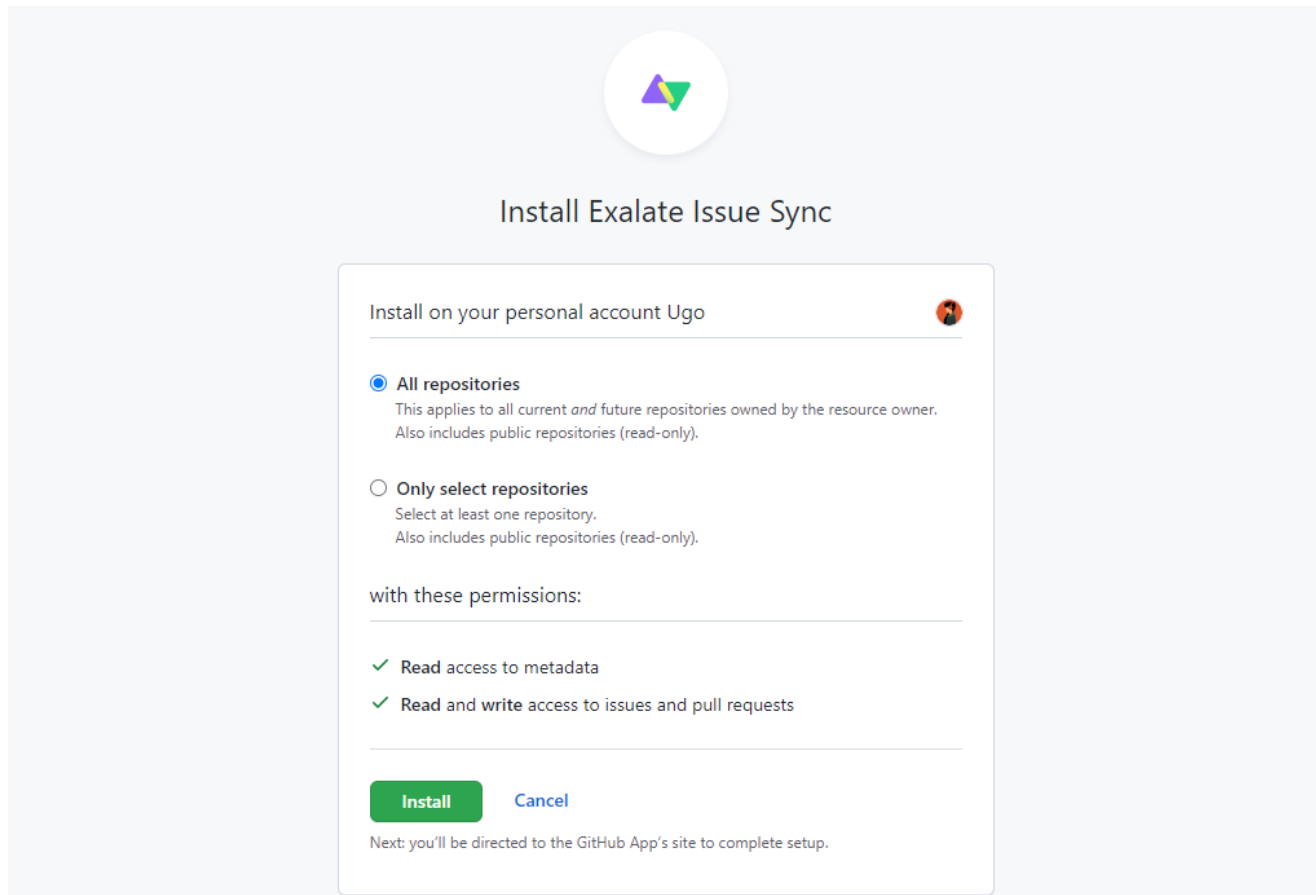
**Note:** Exalate is also available for GitHub Enterprise. For installing Exalate on GitHub Enterprise Cloud, check [this article](#).





Click on the app listing, then on “Set up a plan”, followed by “Install it for free”. Click the green button to complete your order and begin the installation.

**Note:** *Exalate offers a free trial, but it is not a free app.*



Exalate needs access to your projects and you can choose whether to give it access to specific repositories, or to all of them. It can read the metadata and needs both read and write access to issues and pull requests. If you are happy with that, click the green “Install” button.

Now Exalate will automatically create a node for you. You’ll be redirected to it, and will have to enter your details. After that, click the “Agree and submit” button.

### Email Confirmation



#### Almost there...

We sent a verification email to **jmolouy@...ko.com**

Please open the email and click "Verify Exalate instance" to verify your Exalate

If you didn't receive the email within 10 minutes, please make sure your email is correct and click Resend email. If you want to update the email address, click Change email.

Resend email

Change email

[Documentation](#) [EULA](#) [Support](#) [Report a bug](#)

You'll get a confirmation email in your intray. Click the button in the email to verify your account. Then, click continue.

- Account settings
- Profile**
- Account
- Appearance New
- Account security
- Billing & plans
- Security log
- Security & analysis
- Emails
- Notifications
- SSH and GPG keys
- Repositories
- Organizations
- Saved replies
- Applications

**Developer settings**

**Moderation settings**

- Blocked users

### Public profile

**Name**

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

**Public email**

You have set your email address to private. To toggle email privacy, go to [email settings](#) and uncheck "Keep my email address private."

**Bio**

You can @mention other users and organizations to link to them.

**URL**


**Twitter username**

**Company**

You can @mention your company's GitHub organization to link it.

**Location**

**Profile picture**



Next, you need to create an access token for use in Exalate. To do that, go back to GitHub. Open the top-right menu and click “settings”. On the next screen, look at the menu on the left and click “Developer settings”. Then on the next screen, click “Personal access tokens”.

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

### Personal access tokens

[Generate new token](#)

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the [GitHub API](#).

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Click the “Generate new token” button toward the top right. A detailed set of options will appear. You can optionally add a note to help remind you what your token is for later.

GitHub Apps

OAuth Apps

Personal access tokens

### New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

for Exalate

What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- repo** Full control of private repositories
  - repo:status** Access commit status
  - repo:deployment** Access deployment status
  - public\_repo** Access public repositories
  - repo:invite** Access repository invitations
  - security\_events** Read and write security events
- workflows** Update github action workflows
- writepackages** Upload packages to github package registry
- readpackages** Download packages from github package registry
- deletepackages** Delete packages from github package registry
- admin:org** Full control of orgs and teams, read and write org projects
  - write:org** Read and write org and team membership, read and write org projects
  - read:org** Read org and team membership, read org projects
- admin:public\_key** Full control of user public keys
  - write:public\_key** Write user public keys

The only option you need to select is “repo”, which should also automatically select all the other tick boxes in its group. After that, click the green button at the bottom to generate the token. Be careful on the next screen, because it’s the only chance you get to actually copy the token. If you forget, you’ll have to do it all over again!

Highlight the token (the long number with the light green background), and copy it somewhere safe. You have to use it each time you log in, it isn’t just a one-off activation, so don’t delete it after logging in for the first time.

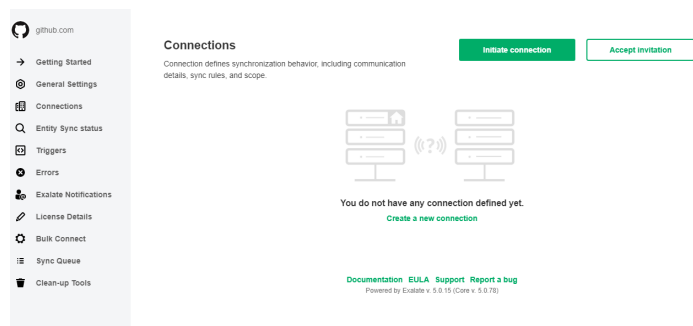
Next, go back to the Exalate node page and paste it into the “Token” box. Click the login button and, if all goes well, you should be taken to the main Exalate menu.

Congratulations! Exalate is now installed on both sides, so let’s set up the connection!

### Step 3 - Connect Your ServiceNow and GitHub Instances

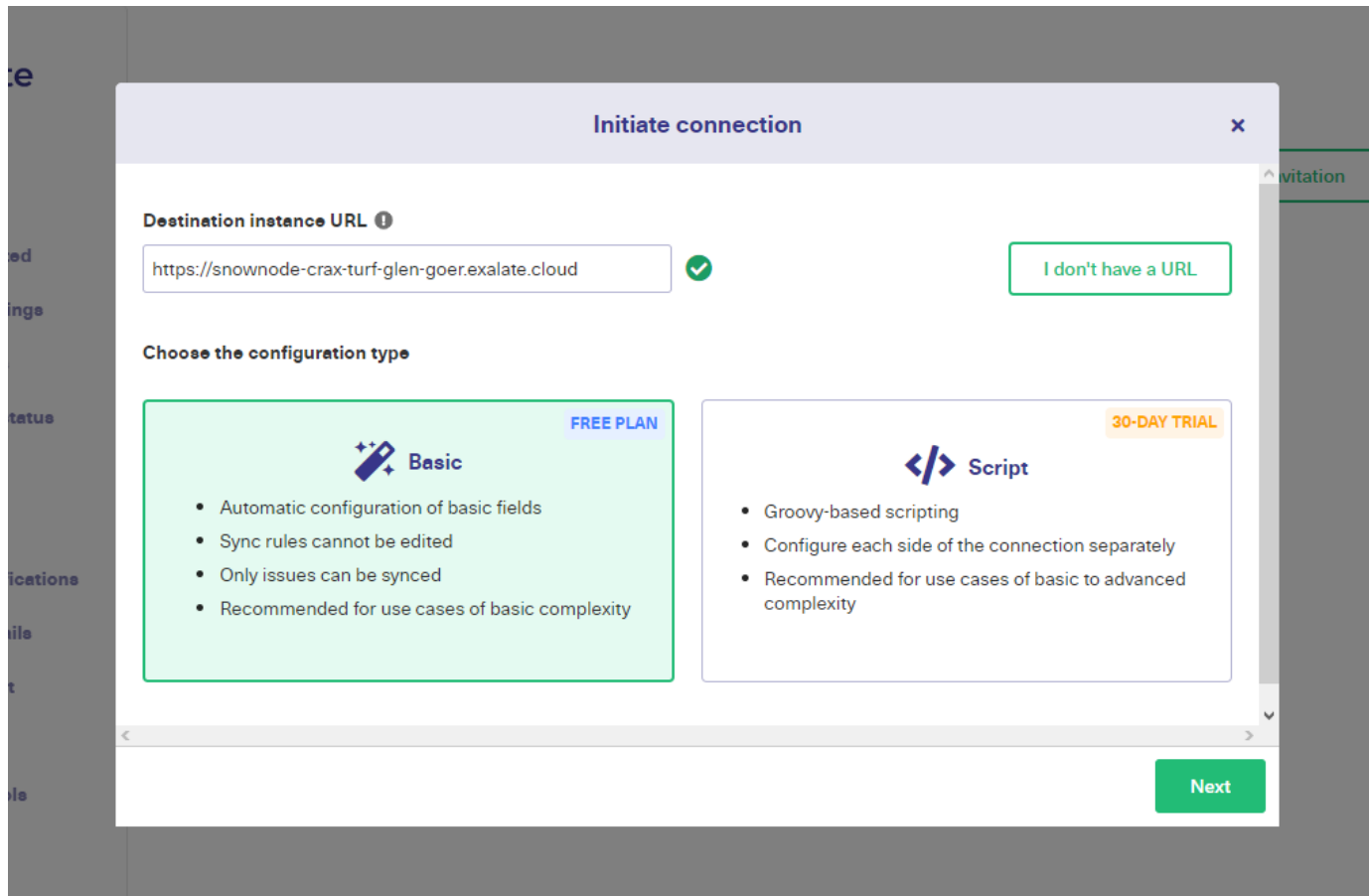
Now that you have Exalate running on both platforms, you can go ahead and connect them to one another. You do this by initiating the connection from one side and accepting it on the other side.

It doesn’t matter whether you initiate the connection in ServiceNow or GitHub. For this guide, I’ll use GitHub to initiate the connection, then accept it in ServiceNow. Exalate uses a common interface across platforms, so the process is much the same either way.



In your GitHub node, go to the Exalate menu and click “Connections” if you aren’t there already. This will show you any existing connections, but since this is the first time you’re here, there won’t be any.

Click “Initiate connection”.



Next, enter the destination URL i.e the ServiceNow URL. After checking if Exalate is installed on the destination side, you will be asked to choose the configuration type. Exalate comes in 2 configuration modes: the Basic mode and the Script mode.

The Basic mode already has GitHub and ServiceNow entities mapped with each other. Synchronization between them happens according to the mappings. These cannot be changed. So if you have simple synchronization needs then you can go ahead and use this mode.

Exalate also supports a [Free Plan](#) that comes with the Basic mode and it has up to 1000 free syncs per month.

The Script mode, on the other hand, has advanced configurations. With this mode, you can control what information is sent and received with the help of sync rules. It is best suited for complex and advanced integration use cases.

Let us see how they both work.

## The Basic mode

To continue using the Basic mode, click "Basic" on the screen shown above and then hit "Next".

Initiate connection ✕

**Select a repository for the incoming sync**

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

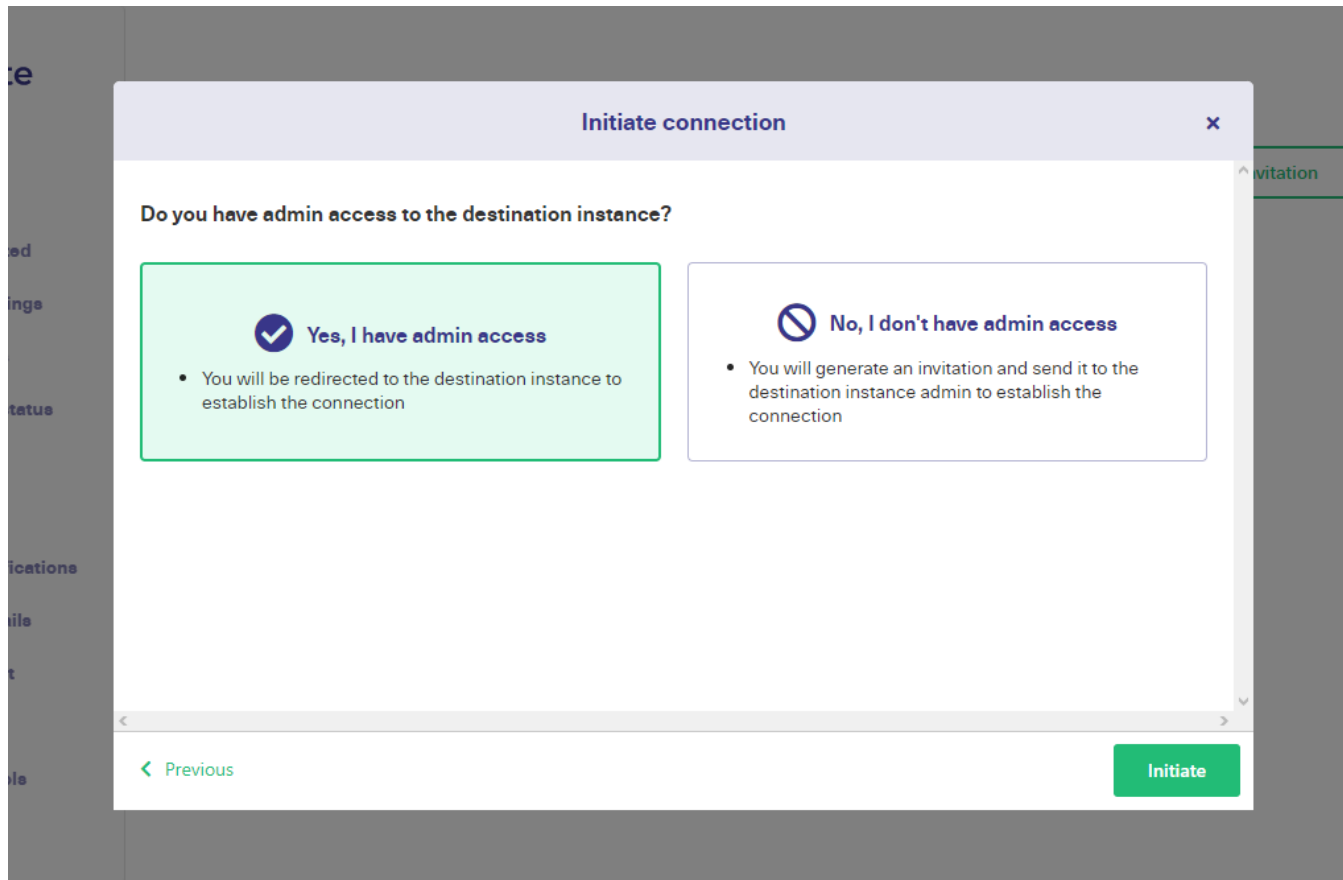
Please select the repository where you want to create issues, received from the other side.\*

---

[< Previous](#) [Next](#)

Proceed to select the repository on the GitHub side. This will be the repository in which Incidents from ServiceNow will be synced. Select the one you want from the drop-down list and click "Next".

**Note:** By default, the Basic Mode supports syncing the summary, description, comments, attachments, and issue types.





You then need to verify if you have admin access to the destination instance, i.e. ServiceNow in our case. Proceed with the verification by clicking "Yes, I have admin access". Then click "Initiate".

In case you don't have access, you will be required to copy an invitation code on the GitHub side and manually paste it on the ServiceNow side. We will see how to do this in detail in the Script mode.

Accept invitation



**Connection established successfully** ✓

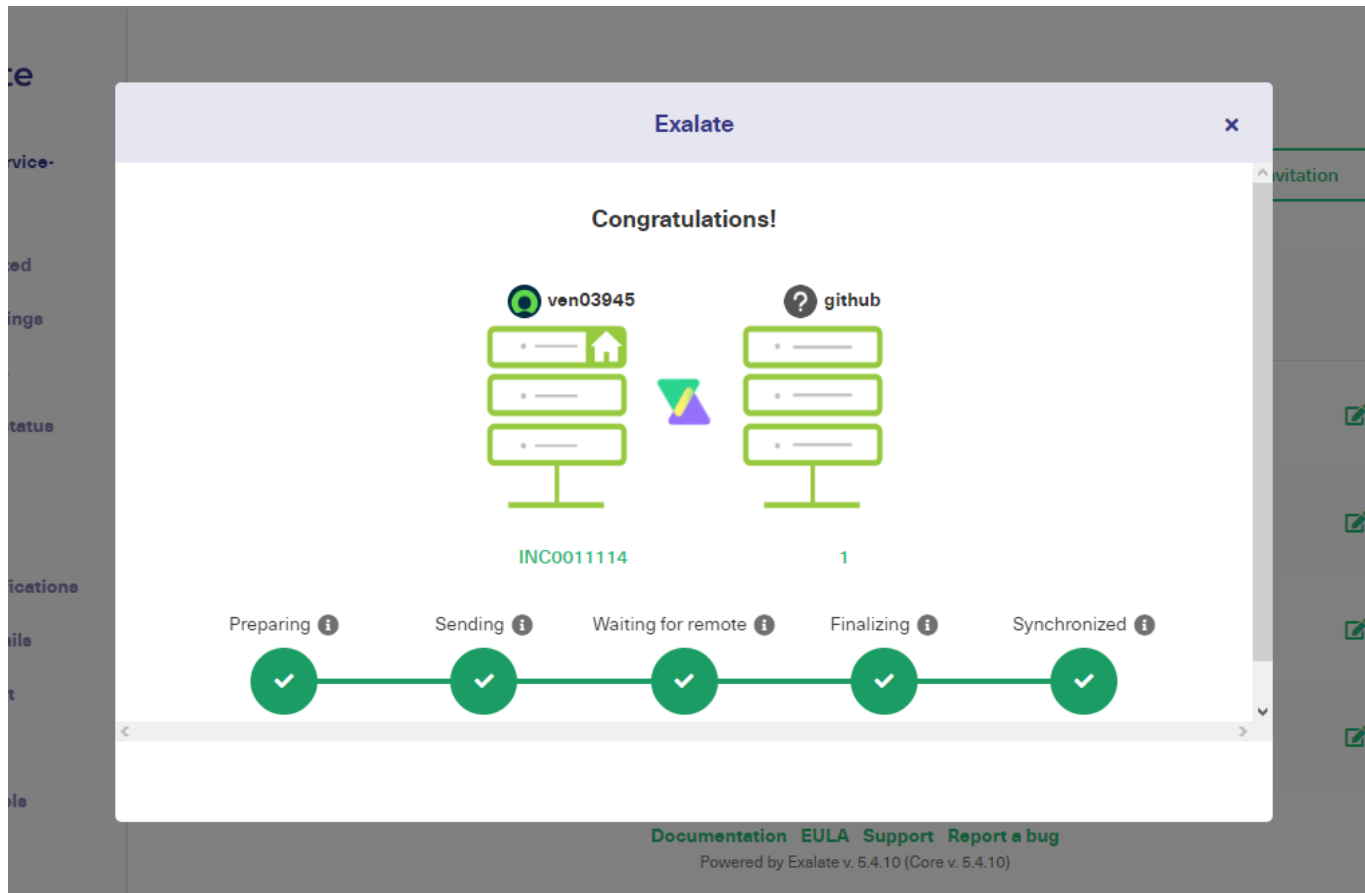
Sync your first incident to see how it works.

Please enter an incident number to proceed

INC0010111

Exalate

After verification, you will be redirected to the ServiceNow instance. Here, you need to enter the incident number you want to synchronize on the GitHub side. If you click "Exalate" a new issue will be created in GitHub. Information between the incident and the issue will be synced bi-directionally thereon. A similar screen on the GitHub side will require you to give an issue key. A reverse procedure will be followed then.



Issues and incidents can be synchronized, as explained in this section. In addition to this, you can create automatic synchronization [triggers](#) for syncing issues and incidents, or you can sync existing entities using the "[Bulk Connect](#)" option.

### The Script mode

To continue with this mode, click "Next" when you select "Script".

**Initiate connection** ✕

**Connection information**

<p><b>Local instance short name*</b></p> <input type="text" value="Github"/>	<p><b>Remote instance short name*</b></p> <input type="text" value="ServiceNow"/>
--	---

**Connection name\***

**Description**

This is a connection between Github and ServiceNow

---

[< Previous](#)Next

You then need to enter a name for your local instance (GitHub in this case), and the destination instance (ServiceNow). The names you use will be combined to make a connection name. You can change this if you like.

There's also an optional field for a description. This is especially useful if you have multiple connections.

When you're ready, click "Next".

### Initiate connection



#### 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.\*

[← Previous](#)

[Initiate](#)

On the next screen, you need to pick a GitHub repository from those available. Items in this repository will be synchronized with those in ServiceNow. Make a choice, and click “Initiate”.

**Initiate connection** ✕

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

Use the following invitation code:

[Copy invitation code](#)

**Go to remote**

Exalate will now generate a code that needs to be pasted into ServiceNow to complete the connection. Click the “Copy invitation code” button, and paste the code somewhere safe, like a text editor.

Then open ServiceNow. You can do this by clicking “Go to remote”, or you can navigate there directly. If you’re using your own instance, you can find the Exalate console by typing “Exalate” into the search field above the left-hand menu.

**Connections**

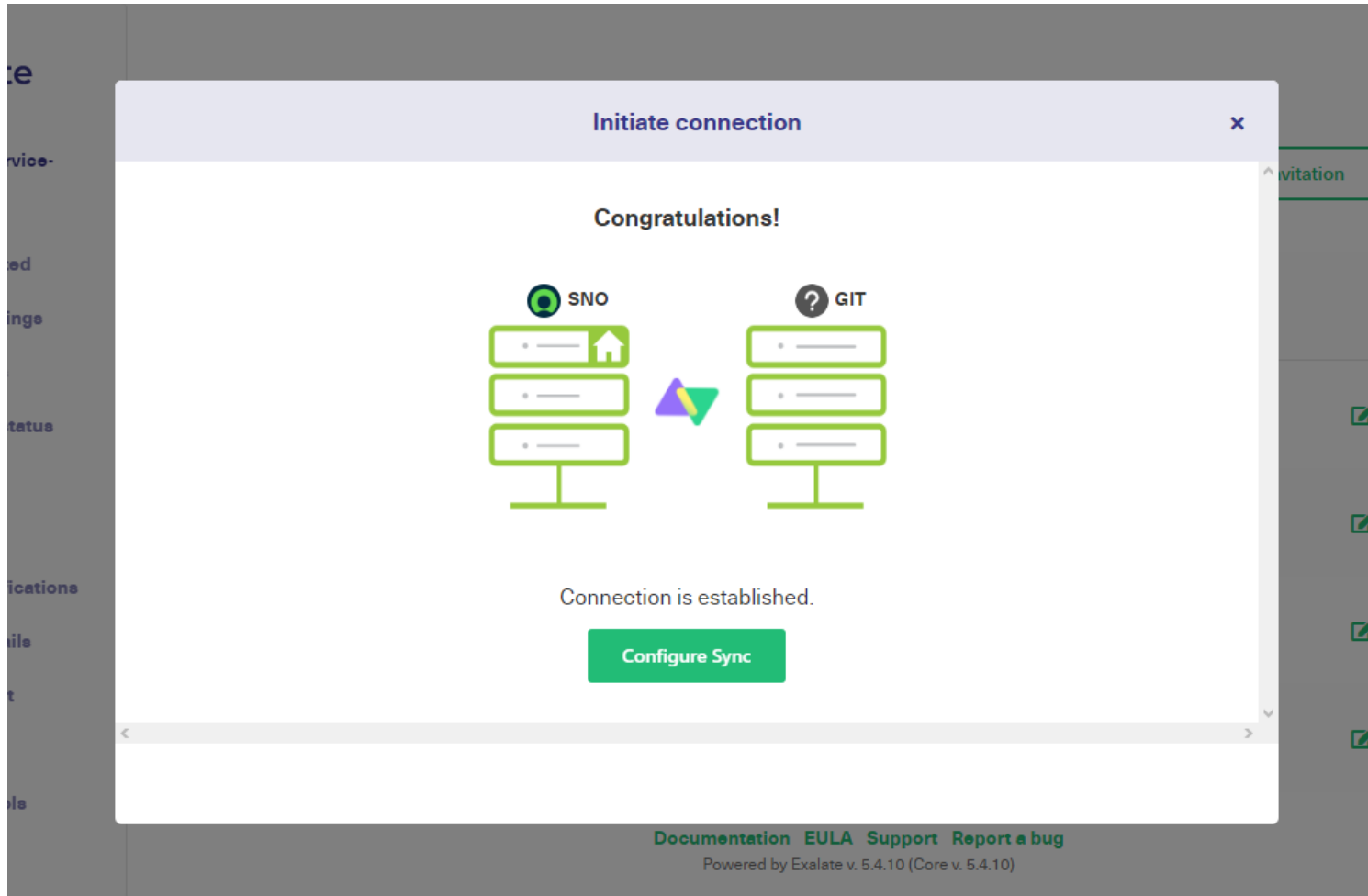
Connection defines synchronization behavior, including communication details, sync rules, and scope.

[Initiate connection](#) [Accept invitation](#)

Connection	Entities under sync	Last sync	Status
<b>Azure DevOps</b> ? <b>_to_ServiceNow</b> <small>This is a connection between Azure DevOps and S...</small>	0		● Active
<b>idalkomarketing_to_dev10268</b> ? <b>8</b>	1	Entity INC00... 7 hours ago	● Active
<b>Jira_to_ServiceNow</b> <small>This is a connection between Jira and ServiceNow</small>	0		● Active

In the Exalate console, click “Connections”. Then click the “Accept invitation” button.

You'll see a large text field. Paste the invitation code you got from GitHub here and click "Next". Click through the next confirmation screen and the connection will be established.



Now you've got your connection set up. If you click the "Configure Sync" button, you'll be taken straight to the edit connection screen.

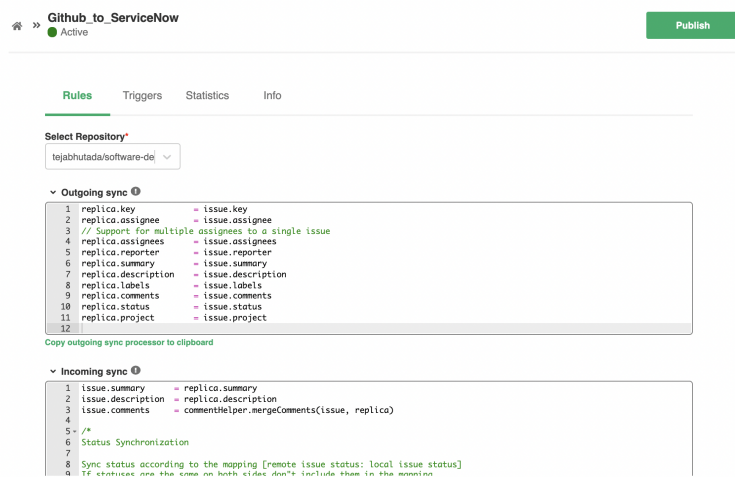
In the next steps, you can configure the connection to do exactly what you want.

## Step 4 - Configure Your Connection to Determine What Information Gets Shared

In this step, you can tune the connection to share the precise data you need. As well as specifying which items are synchronized, you can choose which fields are used and how they map to each other. You can also use your own values for specific fields.

To edit a connection, find its listing in the connections screen. You'll see four icons on the right. Click the furthest left to edit your connection.

You'll now see a screen with four tabbed areas. These are "Rules", "Triggers", "Statistics" and "Info". You'll look at triggers in step five. For now, make sure "Rules" is selected and have a look at the boxes below.



The screenshot shows the configuration page for a connection named "Github\_to\_ServiceNow". The "Rules" tab is active, displaying two sections for synchronization rules:

**Select Repository:** tejabhutada/software-dej

**Outgoing sync**

```
1 replica.key = issue.key
2 replica.assignee = issue.assignee
3 // Support for multiple assignees to a single issue
4 replica.assignees = issue.assignees
5 replica.reporter = issue.reporter
6 replica.summary = issue.summary
7 replica.description = issue.description
8 replica.labels = issue.labels
9 replica.comments = issue.comments
10 replica.status = issue.status
11 replica.project = issue.project
12
```

**Incoming sync**

```
1 issue.summary = replica.summary
2 issue.description = replica.description
3 issue.comments = commentHelper.mergeComments(issue, replica)
4
5 /*
6 Status Synchronization
7
8 Sync status according to the mapping [remote issue status: local issue status]
9 If statuses are the same on both sides don't include them in the mapping
```



There are two large areas containing outgoing sync rules and incoming sync rules. These define how synchronized items are mapped onto one another. Above those is a selection allowing you to change the repository the rules apply to.

Take a closer look at the outgoing sync section. The rules here copy items from the issue to the replica that will be sent to the other side and copied into ServiceNow. If you don't want all the information sent over, you can comment or delete any line you like.

Commenting by adding **two slashes** (//) at the start of the line is also a good way to quickly enable and disable items. That's useful if you want to restrict the information you send out, then add it back later.

By default, the mappings use the same suffix for every item, so `replica.assignee` is mapped to `issue.assignee`. You can change that easily. You could make the GitHub assignee map to a label in ServiceNow by removing the assignee line and changing the label line to read: `replica.label = issue.assignee`.

You can also use specific values. For example, changing that line to `replica.label = "from GitHub"` would add that label to synced ServiceNow items. That way, your team can see which items came from the synchronization.

When you've got everything set up how you want, click "Publish" at the top right, and your changes will be saved. Don't forget to check with both of your teams to make sure items are being exchanged correctly. It is easy to make a mistake, but also easy to fix it.

There's much more you can do with sync rules. Read [this documentation](#) to find out.

## Step 5 - Set Up Automated Synchronization Triggers

Synchronization Triggers define the conditions for information exchange. They control when items are synchronized.

There are two places to edit them. From the edit connection screen used in the previous step you can click "Triggers". That's what I'll do here. You can also access a separate triggers screen from the left-hand menu. It works almost the same way, except the triggers

listed there have a separate field to show what connection they apply to.

The screenshot displays the Exalate user interface. On the left is a sidebar with the Exalate logo and a navigation menu including 'github.com', 'Getting Started', 'General Settings', 'Connections', 'Entity Sync status', 'Triggers', 'Errors', 'Exalate Notifications', 'License Details', 'Bulk Connect', 'Sync Queue', 'Clean-up Tools', and 'Logout'. The main area shows the configuration for a connection named 'SNO1\_to\_GIT2', which is active. There are tabs for 'Rules', 'Triggers', 'Statistics', and 'Info', with 'Triggers' selected. A message states: 'Create a trigger to synchronize entities automatically. All entities that fit the query will be triggered for synchronization.' A '+ Create trigger' button is visible. At the bottom, there are links for 'Documentation', 'EULA', 'Support', and 'Report a bug', along with version information: 'Powered by Exalate v. 5.4.10 (Core v. 5.4.10)'.

The triggers screen shows your existing triggers but is blank the first time you look at them. To get started, click the “Create trigger” button on the right.

### Add trigger ✕

**Trigger will apply to selected entity type**

Select... ▾

Specify a search query using GitHub advanced search syntax to synchronize issues automatically. All issues that fit the query will be triggered for synchronization. [Find more details.](#)

**If** ?

is:issue is:open label:bug

**Notes**

Active?

**Add**

There are several controls on the add trigger screen. At the top is a dropdown box to pick what type of entity the trigger applies to.

Next, in the “If” section, you write a query that picks out the issues you want to synchronize. You can read more about the [query syntax here](#).

There’s a space to leave notes. As with the connection screen, this can be a lifesaver if you have lots of triggers, or multiple team members trying to figure out what each other is doing.

Last but not least is the “Active?” switch. Don’t forget to turn this on, or nothing will happen. You can also use it to disable your triggers temporarily.

When you’ve finished, click the “Add” button.

## Step 6 - Start Synchronizing Tasks

Now that you’ve set everything up, all you have to do is wait for Exalate to synchronize your items. If it isn’t done straight away, don’t panic! For performance reasons, Exalate doesn’t check continually, but if you go away and make a coffee, it should be done when you get back.

The good thing about Exalate is that you can now leave it to work automatically without having to do any more work. Having seen the benefits though, you may want to make adjustments or even set up a sync with another platform.

## Common Use Cases

There are many scenarios where you might want to integrate ServiceNow and GitHub. Here are some examples.

## Developers and Support Team

A common use for integration is to connect your support team and your developers. The support team deals with large numbers of issues from customers. These issues may be duplicated and contain details of customer interaction, along with the technical information the developers are interested in.

The right ServiceNow GitHub integration can automatically filter issues from the support team and copy them into the developers' issue tracking system on GitHub. When the developers resolve the issue, the integration passes the updates back to ServiceNow, alerting the support team to inform the customer that the issue has been resolved.



## Product Development and Quality Control

A product development team might use issues in GitHub to direct engineers in building the product they want. The quality control system analyzes requirements and works with the product development team to convert those requirements into feature requests that the engineers implement.

If the quality control team is tracking their work in ServiceNow, they can benefit hugely by directly monitoring how the issues they raise are implemented in GitHub. Synchronizing systems allow them to see information that is relevant to them, and provide further feedback if needed.

## Conclusion

Integrating software platforms is easy with the right solution. In just a few steps, you can get your teams (and even companies) to share information and work together seamlessly. Once the integration is ready, it can exchange items between your teams without creating extra work for anyone.

Making changes and evolving your integration is just as easy to do, so you control when and how data is shared without worrying that everything will come crashing down.

With a flexible ServiceNow GitHub integration, your teams can focus on their own tasks in their own familiar environment while enjoying easy and secure collaboration.

### *Recommended Reads:*

- [How to Set up a Jira GitHub Integration](#)
- [ServiceNow to ServiceNow Integration: Set up a Two-Way Sync](#)
- [Jira ServiceNow Integration: How to Set up an Integration in 6 Steps](#)
- [How to Set up an Azure DevOps GitHub Integration](#)
- [How to Set up a Zendesk GitHub Integration](#)
- [GitHub Salesforce Integration: How to Set up a Sync in 6 Steps](#)
- [How to Set up a Salesforce ServiceNow Integration](#)
- [ServiceNow Integrations: Integrate ServiceNow and Other Systems Bidirectionally](#)