

Build a custom connector with basic authentication to create a GitHub repository



In this example, you will create a **custom connector** that uses **basic authentication** to connect to GitHub, create a new repository and add a new issue as a task to be completed within the repository.

The complete [OpenAPI Specification](#) and icon for this example are available [here](#).

Summary

Basic authentication in the OpenAPI Specification

Create a GitHub account

Define basic authentication

Step 1: Create the basic OpenAPI Specification

Step 2: Create the security definitions object

Step 3: Create the basic authentication definition

Step 4: Add a security array to the HTTP method objects

Step 5: Add the security object reference to the array

The OpenAPI Specification

Create the authentication challenge workflow

Step 1: Add your custom connector

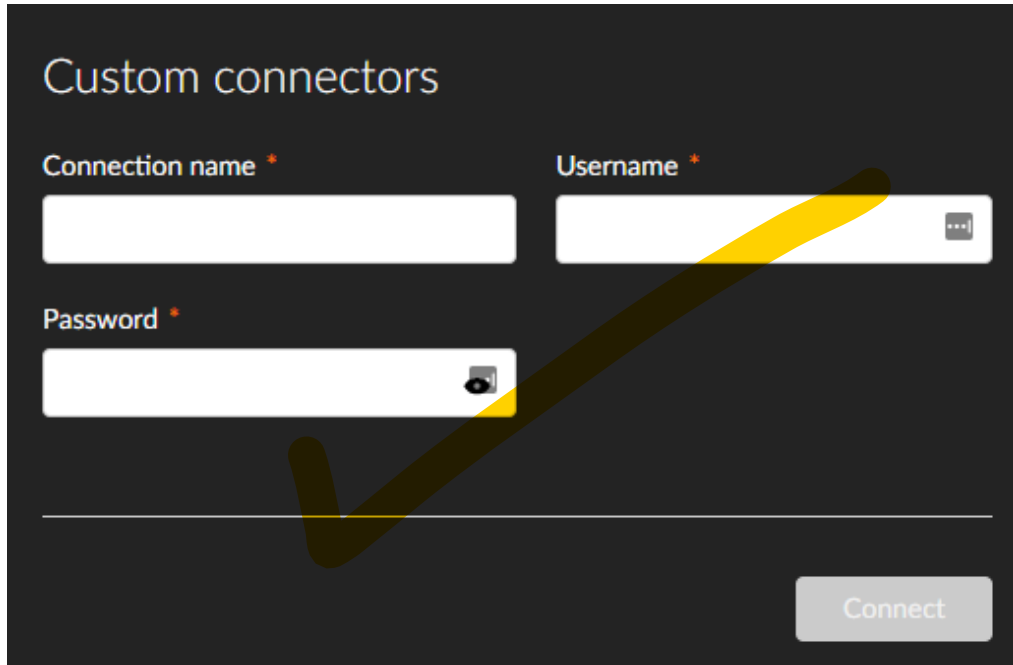
Step 2: Create the workflow

Step 3: Test the workflow

Summary

Basic authentication requires Nintex Workflow Cloud to provide a username and password in the [header](#) when making the [request](#) of the [API](#). The username and password must be accepted by the API for the API to process the request.

When using custom connectors with basic authentication, you must create a **connection** that stores the username and password so it can be passed to the API with the requests.



Custom connectors

Connection name *

Username *

Password *

Connect

Basic authentication in the OpenAPI Specification

To add basic authentication to an OpenAPI Specification, you:

- Define the basic authentication object inside the `securityDefinitions` object.
- Reference to this basic authentication object inside the HTTP method objects that require authentication.

Create a GitHub account

If you do not already have a GitHub account, sign up to create your first GitHub account at <https://github.com/join>.

You will use this username and password to create a connection to your custom connector.

Define basic authentication

Step 1: Create the basic OpenAPI Specification

Create an OpenAPI Specification that:

- Uses the **host** `api.github.com`.
- Accepts **HTTPS** and produces **application/json**.
- Uses a **post** method on `/user/repos` that accepts two string and two boolean parameters in the **body** for the:
 - Repository name.
 - Repository description.
 - Whether the repository can have issues.
 - Whether the repository should be initialized with a README file.
- Uses a **post** method on `/repos/{owner}/{repo}/issues` that accepts:
 - Two strings in the **path** for the GitHub username and repository name.
 - Two strings in the body for the issue title and description.

For more information on creating OpenAPI specifications with multiple operations, see [Define the operations](#).

Extend this custom connector

[Copy](#)

```
{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "GitHub"
  },
  "host": "api.github.com",
  "schemes": [
    "https"
  ],
  "produces": [
    "application/json"
  ],
  "consumes": [
    "application/json"
  ],
  "paths": {
    "/user/repos" : {
      "post" : {
        "summary": "Create a repository",
        "description": "Create a new repository",
        "operationId": "createRepo",
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "schema": {
              "required": ["name"],
              "properties": {
                "name": {
                  "type": "string",
                  "x-ntx-summary": "Repository name"
                },
                "description": {
                  "type": "string",
                  "x-ntx-summary": "Description of repository"
                },
                "has_issues": {
                  "type": "boolean",
                  "x-ntx-summary": "Can this repository have issues?"
                }
              }
            }
          }
        ]
      }
    }
  }
}
```

```

        "auto_init": {
          "type": "boolean",
          "x-ntx-summary": "Create an empty README?"
        }
      },
    ],
    "responses": {
      "200": {
        "description": "OK"
      }
    }
  },
  "/repos/{owner}/{repo}/issues": {
    "post": {
      "summary": "Create issue",
      "description": "Creates an issue in the given repository",
      "operationId": "createIssue",
      "parameters": [
        {
          "name": "owner",
          "type": "string",
          "in": "path",
          "required": true,
          "x-ntx-summary": "Username"
        },
        {
          "name": "repo",
          "type": "string",
          "in": "path",
          "required": true,
          "x-ntx-summary": "Repository name"
        },
        {
          "name": "body",
          "in": "body",
          "schema": {
            "required": ["title"],
            "properties": {
              "title": {
                "type": "string",
                "x-ntx-summary": "Title of issue"
              },
              "body": {
                "type": "string",
                "x-ntx-summary": "Description of issue"
              }
            }
          }
        }
      ]
    },
    "responses": {
      "200": {
        "description": "OK"
      }
    }
  }
}

```

Step 2: Create the security definitions object

Create the **securityDefinitions** object at the end of the OpenAPI Specification, before the final closing brace. Make sure you add a comma to the closing brace at the end of the **paths** object.

Copy

```
"securityDefinitions": {
```

```
}
}
```

Step 3: Create the basic authentication definition

Inside the **securityDefinitions** object, create an object to define the basic authentication. You can name the object any name that is unique within the OpenAPI Specification. In this example, we have named it **basicAuth**.

Add a **type** with a value of **basic** to the object to define it as basic authentication.

[Copy](#)

```
"securityDefinitions": {
  "basicAuth": {
    "type": "basic"
  }
}
```

Step 4: Add a security array to the HTTP method objects

Inside each HTTP method object, create a **security** array.

[Copy](#)

```
"paths": {
  "/repos/{owner}/{repo}/issues" : {
    "post" :{
      "summary": "Create issue",
      "description": "Creates an issue in the given repository",
      "operationId": "createIssue",
      "security" : [
        {
        }
      ]
    }
  },
}
```

Step 5: Add the security object reference to the array

Inside the **security** array in each HTTP method object, reference the security object you defined earlier by using its name as the key, and opening and closing brackets as the value.

[Copy](#)

```
"paths": {
  "/repos/{owner}/{repo}/issues" : {
    "post" :{
      "summary": "Create issue",
      "description": "Creates an issue in the given repository",
      "operationId": "createIssue",
      "security" : [
        {
          "basicAuth": []
        }
      ]
    }
  },
}
```

The OpenAPI Specification

This is the complete OpenAPI Specification that uses basic authentication to create a GitHub repository and issue.

[Copy](#)

```
{
  "swagger": "2.0",
  "info": {
    "version": "1.0.0",
    "title": "GitHub"
  },
  "host": "api.github.com",
  "schemes": [
    "https"
  ],
  "produces": [
    "application/json"
  ],
  "consumes": [
    "application/json"
  ],
  "paths": {
    "/user/repos" : {
      "post" : {
        "summary": "Create a repository",
        "description": "Create a new repository",
        "operationId": "createRepo",
        "security" : [
          {
            "basicAuth": []
          }
        ],
        "parameters": [
          {
            "name": "body",
            "in": "body",
            "schema": {
              "required": ["name"],
              "properties": {
                "name": {
                  "type": "string",
                  "x-ntx-summary": "Repository name"
                },
                "description": {
                  "type": "string",
                  "x-ntx-summary": "Description of repository"
                },
                "has_issues": {
                  "type": "boolean",
                  "x-ntx-summary": "Can this repository have issues?"
                },
                "auto_init": {
                  "type": "boolean",
                  "x-ntx-summary": "Create an empty README?"
                }
              }
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      },
      "post": {
        "summary": "Create issue",

```



```


    "description": "Creates an issue in the given repository",
    "operationId": "createIssue",
    "security" : [
      {
        "basicAuth": []
      }
    ],
    "parameters": [
      {
        "name": "owner",
        "type": "string",
        "in": "path",
        "required": true,
        "x-ntx-summary": "Username"
      },
      {
        "name": "repo",
        "type": "string",
        "in": "path",
        "required": true,
        "x-ntx-summary": "Repository name"
      },
      {
        "name": "body",
        "in": "body",
        "schema": {
          "required": ["title"],
          "properties": {
            "title": {
              "type": "string",
              "x-ntx-summary": "Title of issue"
            },
            "body": {
              "type": "string",
              "x-ntx-summary": "Description of issue"
            }
          }
        }
      }
    ],
    "responses" : {
      "200": {
        "description": "OK"
      }
    }
  },
  "securityDefinitions" : {
    "basicAuth" : {
      "type" : "basic"
    }
  }
}

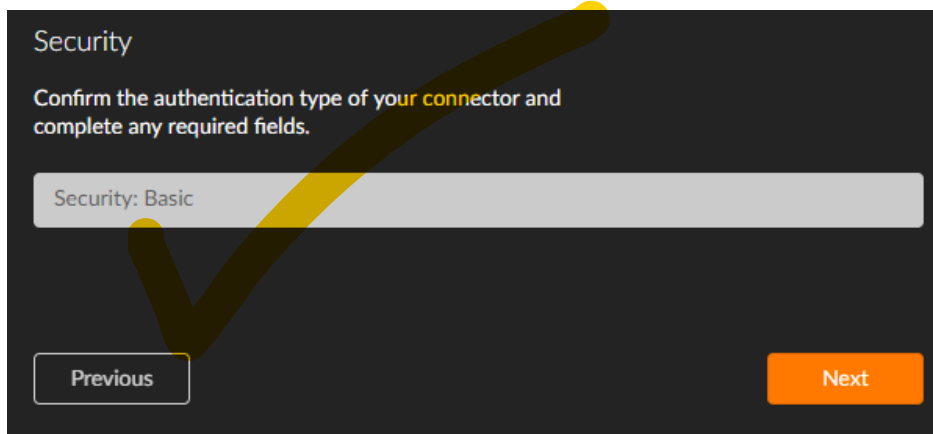
```

Create the authentication challenge workflow

Step 1: Add your custom connector

Import the OpenAPI Specification you created into Nintex Workflow Cloud:

1. Open your Nintex Workflow Cloud tenancy.
2. Click **Xtensions** in the dashboard to open the Xtensions page.
3. Click  in the Custom connector list.
4. Click **Choose a file**.
5. Navigate to the OpenAPI Specification on your computer.
6. Wait for Nintex Workflow Cloud to validate the file.
7. Click **Next**.
8. Nintex Workflow Cloud detects the basic authentication security template.



9. Click **Next**.
10. Edit the **Name** of the connector, which becomes the name of the action group in the Workflow designer.
11. Edit the **Description** of the connector. This appears in the Custom connector list in the Xtensions page.
12. Select or upload an icon for the connector. This is displayed for each action or event in the Workflow designer
13. Click **Publish**.

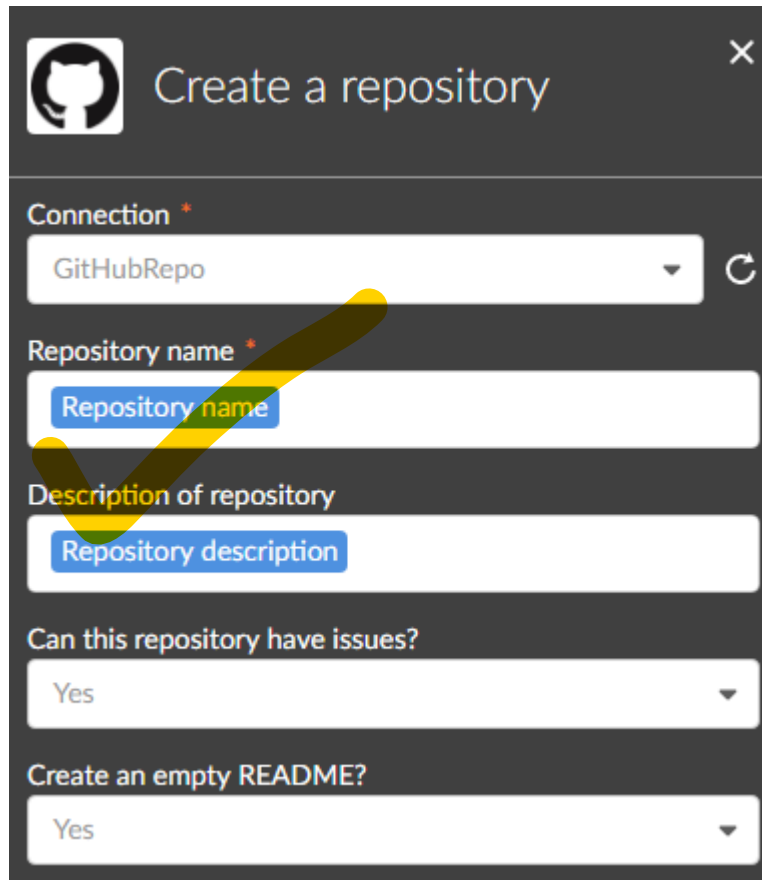
Step 2: Create the workflow

Create a workflow that prompts the user for the name and description of their new repository, creates a repository and adds a boilerplate issue to complete a project plan for the repository.

To create the workflow:

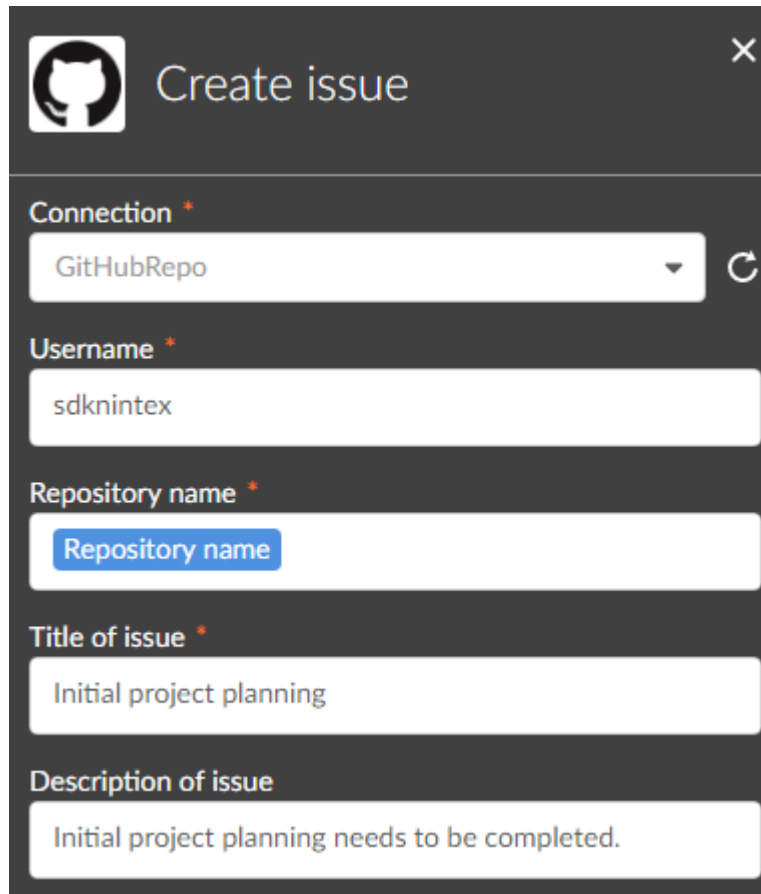
1. Click **Create workflow** in your Nintex Workflow Cloud tenancy.
2. Configure the **Start event** to be a **Public web form with two text variables** for the **repository name and description**.
For more information on designing forms in Nintex Workflow Cloud, see [Design a form](#).
3. **Drag a Create a repository action** after the Start event.
4. Configure the Create a repository action:
 - a. Use the start variables for the repository name and description.

- b. Select Yes for Can this repository have issues.

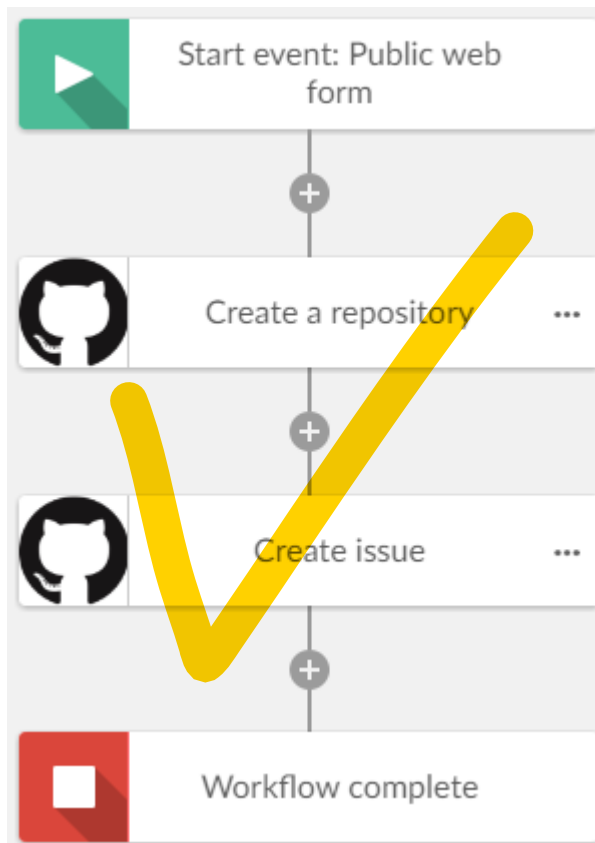


5. Drag a **Create issue** action after the **Create a repository** action.
6. **Configure the Create issue action:**
 - a. Use your GitHub username in the **Username** field.
 - b. Use the start variable for the **Repository name**.

c. Configure the title and description of the new issue.



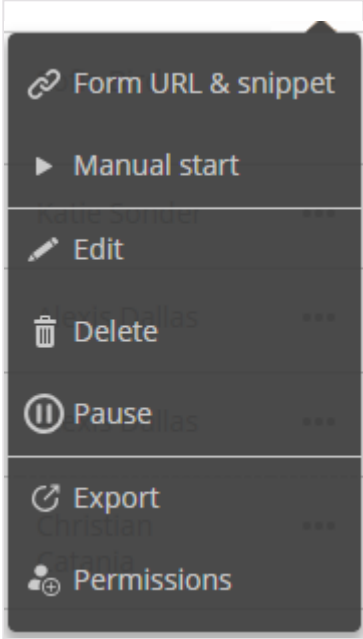
7. Publish your workflow.



Tip: If you want to troubleshoot a custom connector, select **Development** as the Assigned Use when you publish the workflow. Development workflows display more detailed error messages in their instance details. Republish your workflow as Production when you're ready to use it.

Step 3: Test the workflow

- 1. Click **Workflow** in your Nintex Workflow Cloud tenancy.
- 2. Click ... on the workflow you created.



- 3. Click **Manual start**.
- 4. Type a name and description for the repository.
The repository name must be unique in your GitHub account.
- 5. Click **Start**.
Your new repository is created, and an issue is added.

Resources

- [Search Resources](#)
- [SDKs](#)
- [Learning Center](#)
- [Product Help](#)
- [Status](#)

© 2018 Nintex Global Ltd. | [Legal](#) | [Security](#) | [Privacy](#)

Connect

- [Community](#)
- [Partner Central](#)
- [Nintex Xchange](#)
- [Blog](#)

