davinci

Connector Development Quick Start

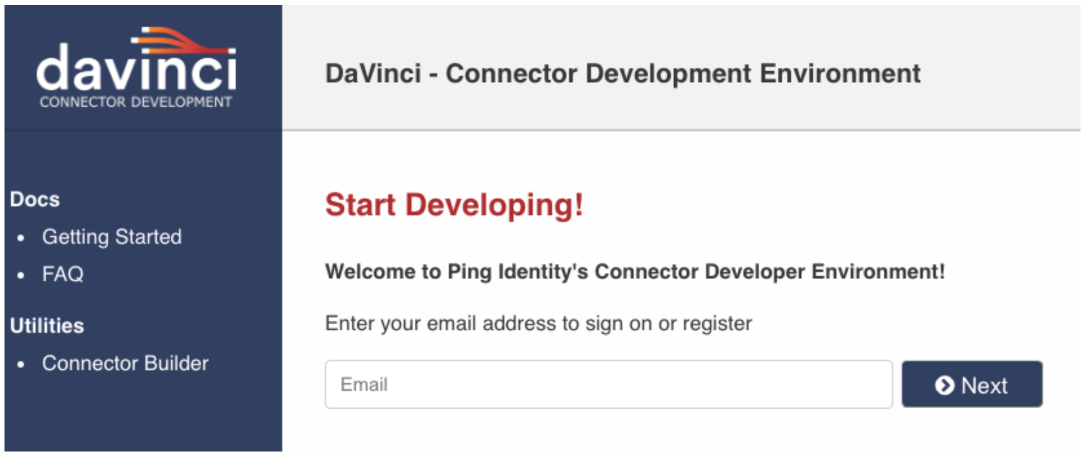
WELCOME / OVERVIEW

To accelerate connector development, Ping has created a hosted development experience. Gone are the days of installing software, standing up services, then the inevitable troubleshooting of wiring the two together. Simply log in to the portal, create a new connector project and start building.

Within the development environment, you'll have a fully-featured isolated DaVinci portal and a web-based instance of Visual Studio Code that comes pre-wired to the underlying DaVinci services. This enables you to quickly iterate on your connectors' capabilities.

CREATING A DEVELOPER ACCOUNT

To create a developer account, navigate to https://davinci.pingidentity.cloud/. Enter your email address then click the **>>** button to proceed.



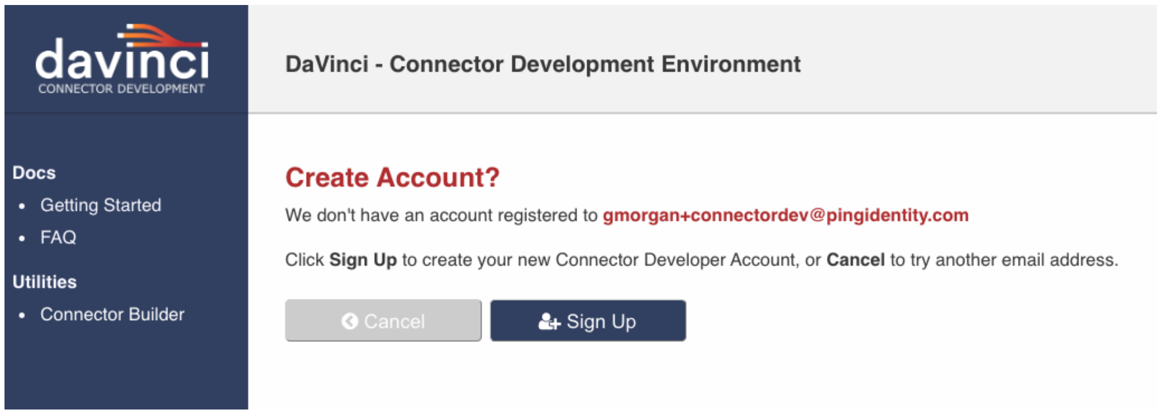
DaVinci - Connector Development Environment

Start Developing!

Welcome to Ping Identity's Connector Developer Environment!

Enter your email address to sign on or register

If this is your first time in the Developer Environment, you'll be prompted to register.



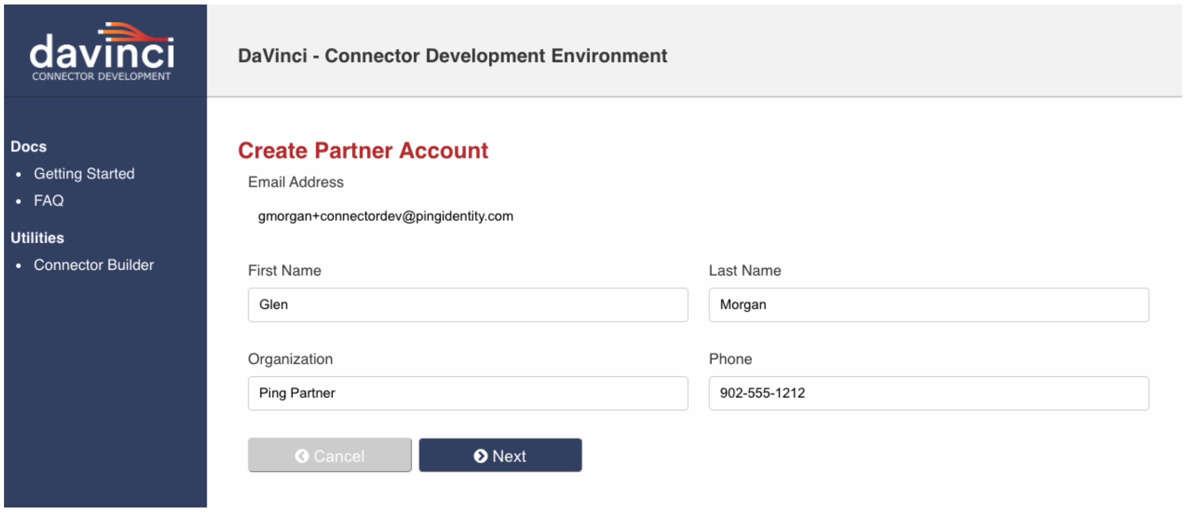
DaVinci - Connector Development Environment

Create Account?

We don't have an account registered to **gmorgan+connectordev@pingidentity.com**

Click **Sign Up** to create your new Connector Developer Account, or **Cancel** to try another email address.

Fill out your account information then click the next button to proceed to the terms and conditions acceptance screen.

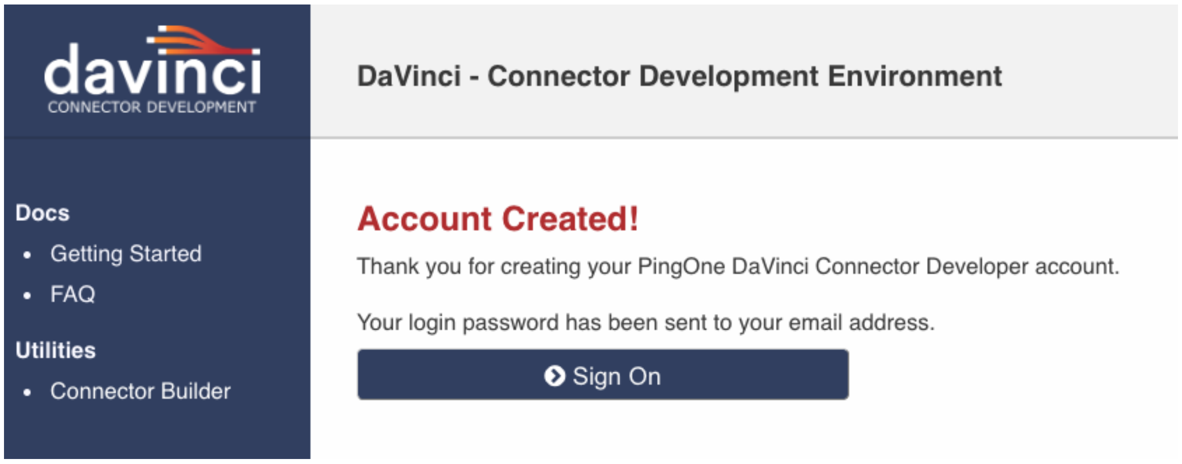


DaVinci - Connector Development Environment

Create Partner Account

Email Address	gmorgan+connectordev@pingidentity.com
First Name	Glen
Last Name	Morgan
Organization	Ping Partner
Phone	902-555-1212

Once the terms and conditions have been accepted, you'll receive a confirmation. Your credentials are emailed to the address you provided.



DaVinci - Connector Development Environment

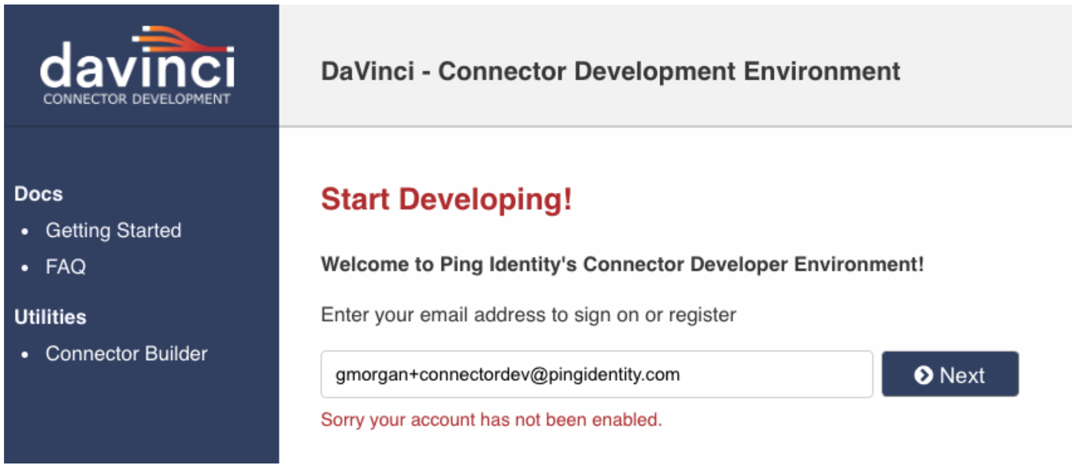
Account Created!

Thank you for creating your PingOne DaVinci Connector Developer account.

Your login password has been sent to your email address.

Your account registration is then sent to the DaVinci Connector Team for review. Once it is approved, you'll receive an email notifying you that you're eligible to proceed. Note: Depending on when you register, it may take up to 2 business days for approval.

If you try to log in before approval, you'll see the following error message



DaVinci - Connector Development Environment

Start Developing!

Welcome to Ping Identity's Connector Developer Environment!

Enter your email address to sign on or register

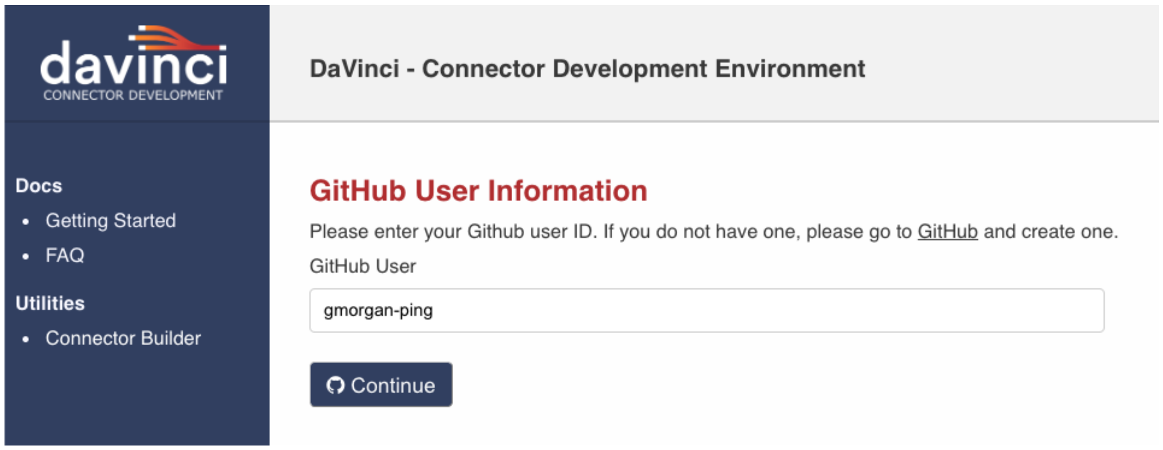
gmorgan+connectordev@pingidentity.com

Sorry your account has not been enabled.

LOGIN TO THE DEVELOPER ENVIRONMENT

Once your developer account has been approved, navigate to the login page at https://davinci.pingidentity.cloud/

Upon your first login, you'll be asked to provide your GitHub User Id as each DaVinci Connector project is saved to its own private GitHub repository. If you do not have a GitHub account, you'll need to create one before proceeding.


DaVinci - Connector Development Environment

Docs
- Getting Started
- FAQ

Utilities
- Connector Builder

GitHub User Information

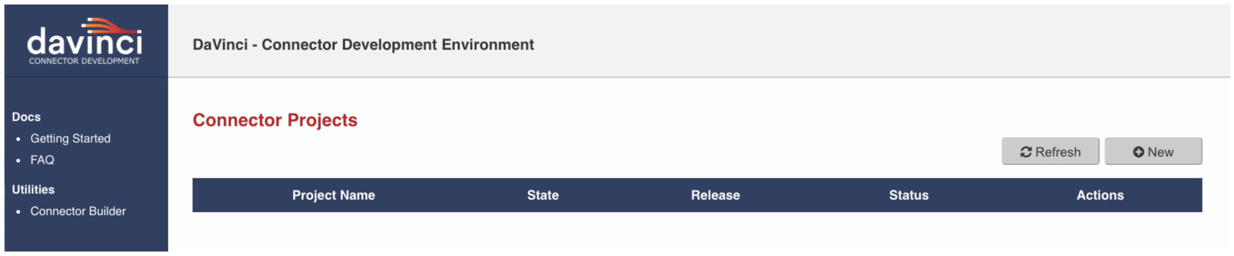
Please enter your Github user ID. If you do not have one, please go to GitHub and create one.

GitHub User: gmorgan-ping

Continue

CONNECTOR PROJECT DASHBOARD

The project dashboard lists all your DaVinci Connector projects. From here, you can manage project settings and services. To create a new project, click the **+ New** button.


DaVinci - Connector Development Environment

Docs
- Getting Started
- FAQ

Utilities
- Connector Builder

Connector Projects

Refresh New

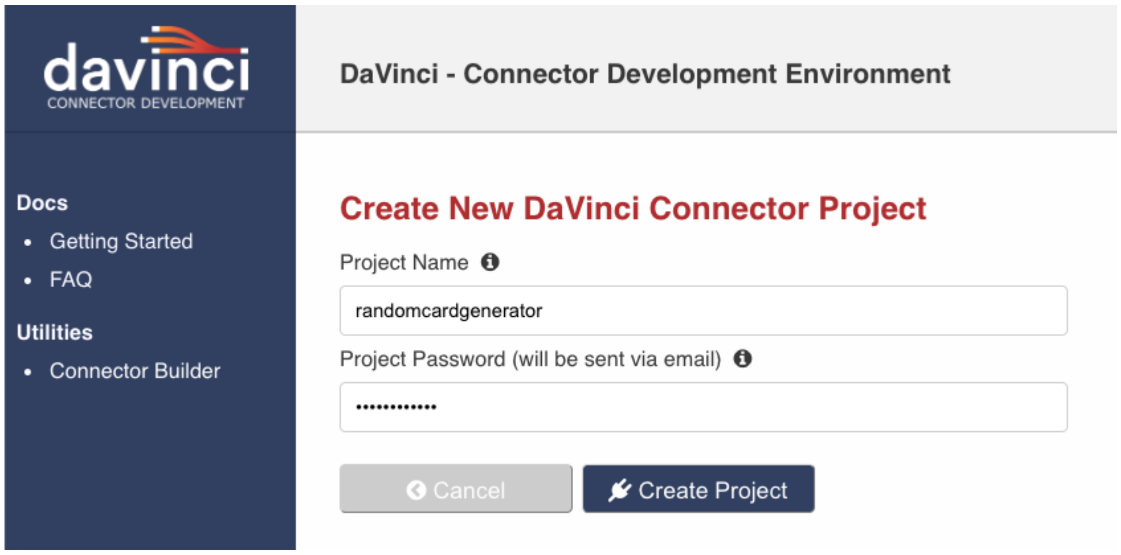
Project Name	State	Release	Status	Actions

CREATING YOUR FIRST CONNECTOR PROJECT

From the New Connector Project screen, enter a project name and set a project password. This password is separate from your developer user login and is used to authenticate into your project's DaVinci Admin Portal and VS Code instance.

Note: The project name must only include lower-case alphanumeric characters and hyphens.

Click **Deploy Project**

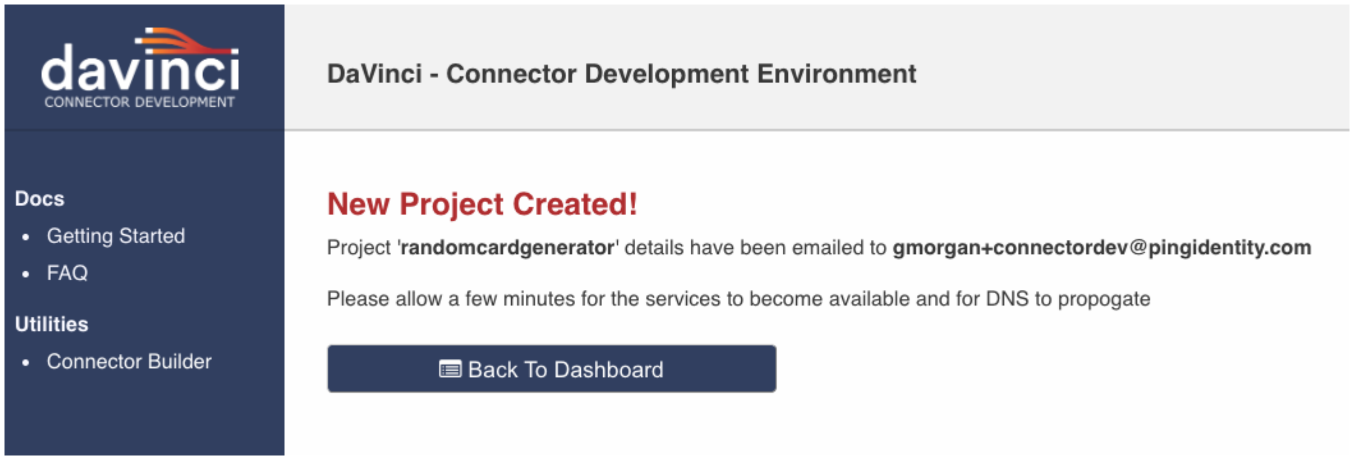


DaVinci - Connector Development Environment

Create New DaVinci Connector Project

Project Name: randomcardgenerator

Project Password (will be sent via email): ••••••••••••



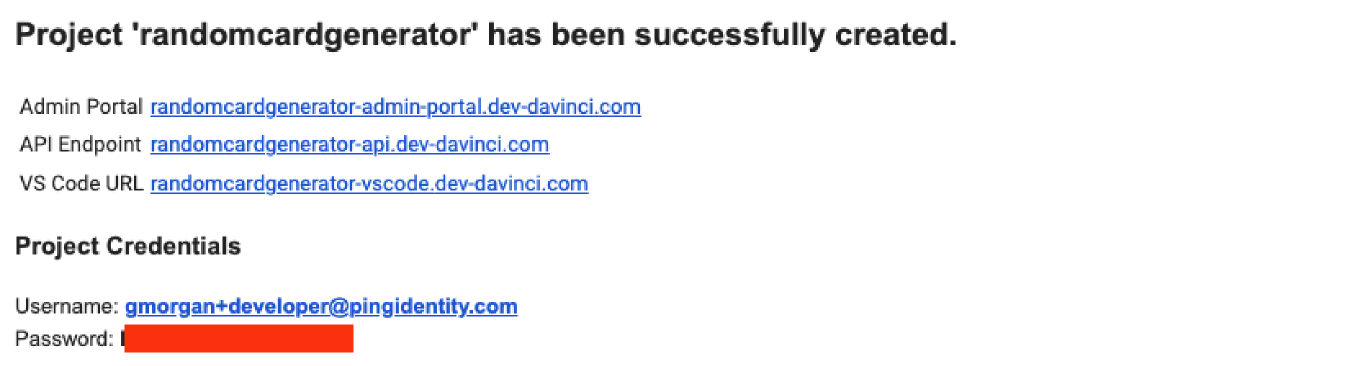
DaVinci - Connector Development Environment

New Project Created!

Project '**randomcardgenerator**' details have been emailed to **gmorgan+connectordev@pingidentity.com**

Please allow a few minutes for the services to become available and for DNS to propogate

A confirmation message will be displayed and 2 emails will be sent. The first email will contain your project name, service endpoints, and login credentials.



Project 'randomcardgenerator' has been successfully created.

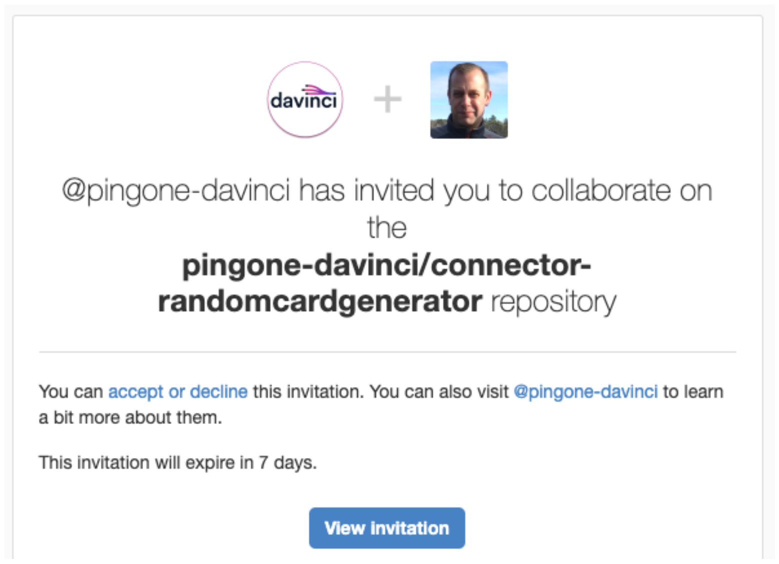
Admin Portal	randomcardgenerator-admin-portal.dev-davinci.com
API Endpoint	randomcardgenerator-api.dev-davinci.com
VS Code URL	randomcardgenerator-vscode.dev-davinci.com

Project Credentials

Username: **gmorgan+developer@pingidentity.com**
Password: [redacted]

The second email will be a Github repository invitation for your newly created project. Before accepting the invitation, please ensure you are logged into GitHub, otherwise you'll be presented with a 404 error.

Please note that the source repository is private. Only you and Ping Identity will have access to it.



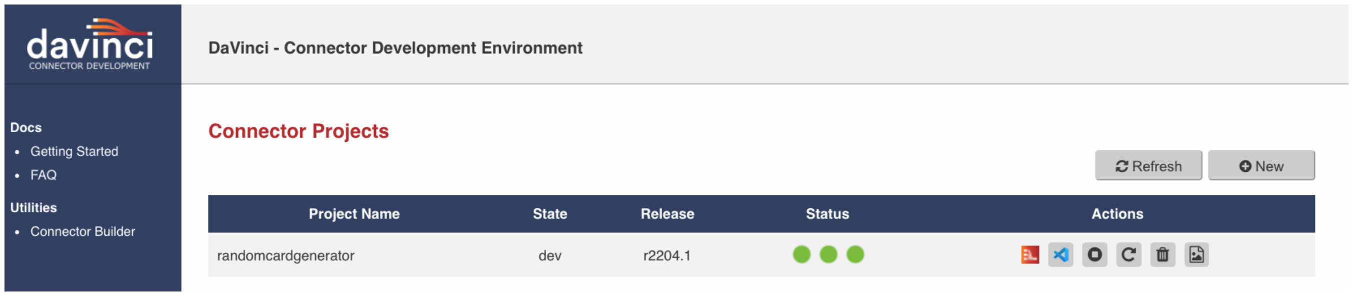
@pingone-davinci has invited you to collaborate on the **pingone-davinci/connector-randomcardgenerator** repository

You can accept or decline this invitation. You can also visit @pingone-davinci to learn a bit more about them.

This invitation will expire in 7 days.

PROJECT DASHBOARD

Back at the project dashboard, your connector projects and their respective statuses are displayed. Once all services are running, you'll see 3 green circles that represent the status of the DaVinci Admin Portal, DaVinci API service, and VS Code.



Project Name	State	Release	Status	Actions
randomcardgenerator	dev	r2204.1	● ● ●	

To launch either the Admin Portal or VS Code for your project, click on their links.

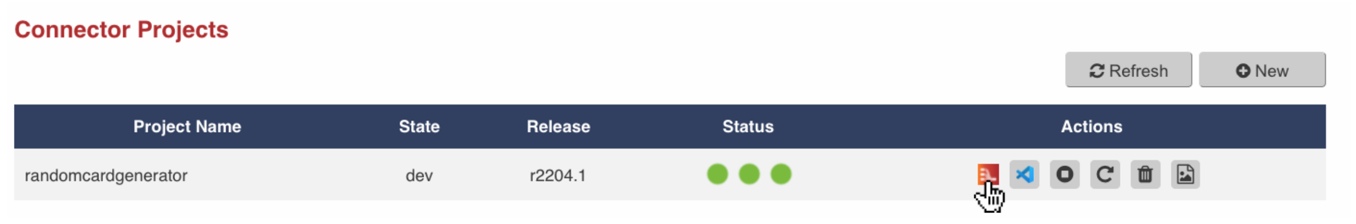
From the Actions list, you can **stop**, **restart** or **delete** the project and upload an **icon** for your Connector.

The dashboard also provides the connector's **release state** (dev, demo, marketplace, prod) and project **release template version**.

DAVINCI ADMIN PORTAL

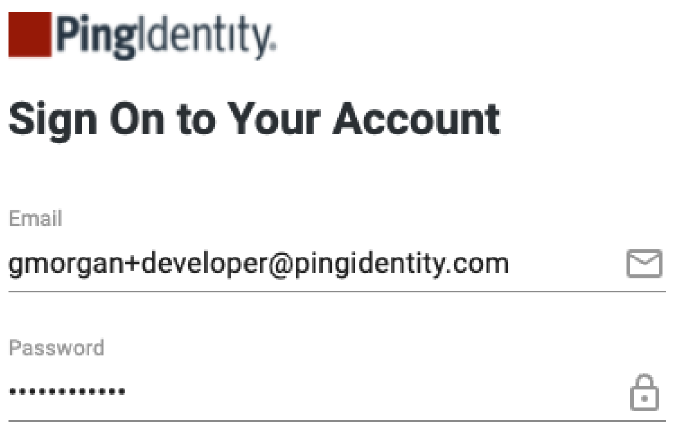
While your project is in the **dev** environment, it will run within its own isolated deployment space without access to other DaVinci tenants or projects.

To launch your project's DaVinci Admin Portal, click the portal link.



Project Name	State	Release	Status	Actions
randomcardgenerator	dev	r2204.1	● ● ●	

To login to the portal, enter your **developer account** email address and the **project password** (emailed to you)



gmorgan+developer@pingidentity.com

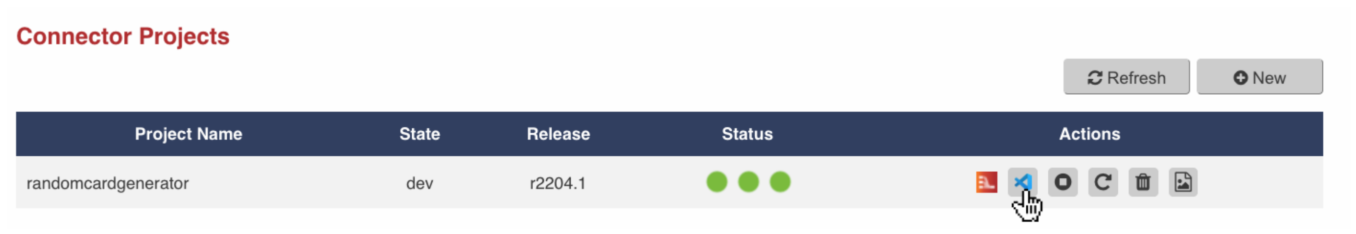
••••••••••••

DEVELOPING YOUR CONNECTOR

All coding for the connector project is performed in Visual Studio Code. To streamline connector development, the hosted instance of VS Code is preconfigured to publish your project to your DaVinci Admin portal.

From VS Code, you can develop, deploy, debug and execute your project's connector.

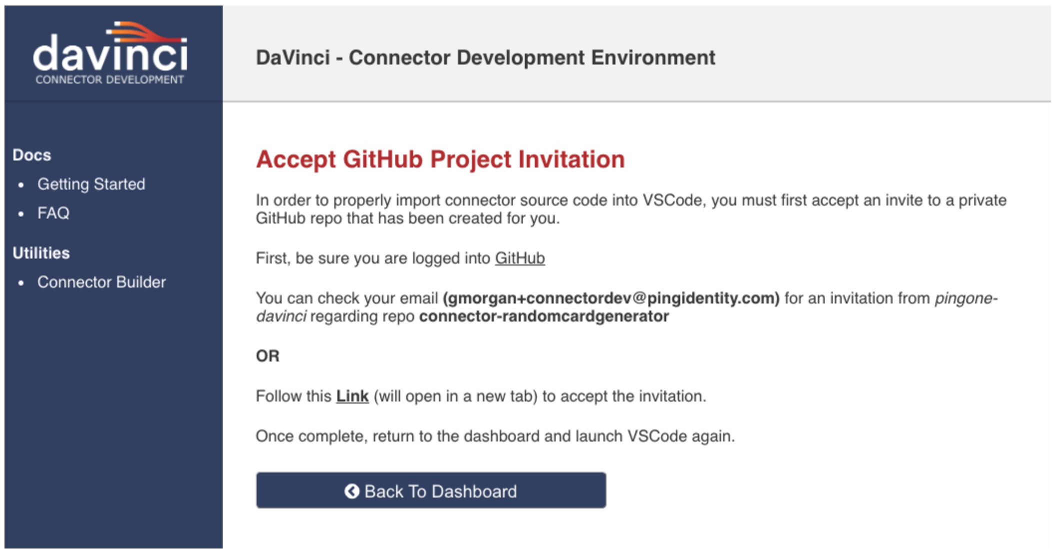
To launch, click on the icon from the project dashboard.



Connector Projects

Project Name	State	Release	Status	Actions
randomcardgenerator	dev	r2204.1	● ● ●	

The first time to launch VS Code for your project, a dialog will be presented ensuring that you have accepted the Github repo invitation. Accepting the invitation is required to continue so that the base connector project can be cloned into your workspace.



DaVinci - Connector Development Environment

Accept GitHub Project Invitation

In order to properly import connector source code into VSCode, you must first accept an invite to a private GitHub repo that has been created for you.

First, be sure you are logged into GitHub

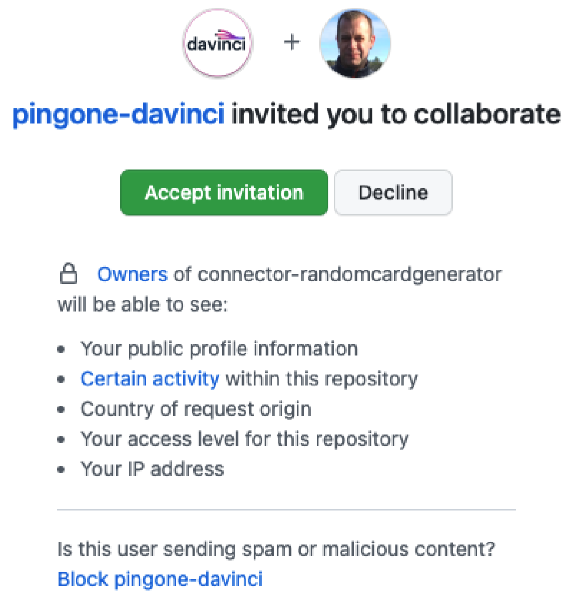
You can check your email **(gmorgan+connectordev@pingidentity.com)** for an invitation from *pingone-davinci* regarding repo **connector-randomcardgenerator**

OR

Follow this **Link** (will open in a new tab) to accept the invitation.

Once complete, return to the dashboard and launch VSCode again.

Ensure you are currently logged into the Github account that the invitation has been sent to otherwise you'll receive a 404 error.



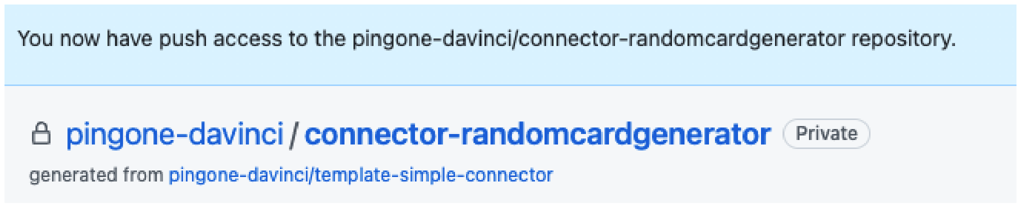
pingone-davinci invited you to collaborate

Owners of connector-randomcardgenerator will be able to see:

- Your public profile information
- Certain activity within this repository
- Country of request origin
- Your access level for this repository
- Your IP address

Is this user sending spam or malicious content?
Block pingone-davinci

When the invite is accepted in Github, you'll be presented with the repo. You may close this window for now or bookmark it for future reference.

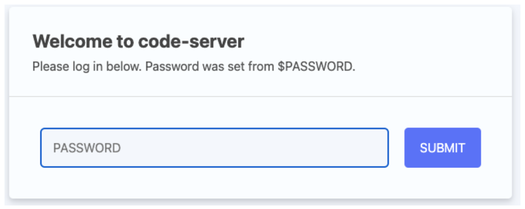


You now have push access to the pingone-davinci/connector-randomcardgenerator repository.

pingone-davinci / connector-randomcardgenerator Private
generated from pingone-davinci/template-simple-connector

Now, you can return to the dashboard then relaunch VS Code. As the invitation has been accepted you'll be signed into the private instance.

Note: If you bookmark VS Code's URL, you'll need to enter your project's password to successfully log in.

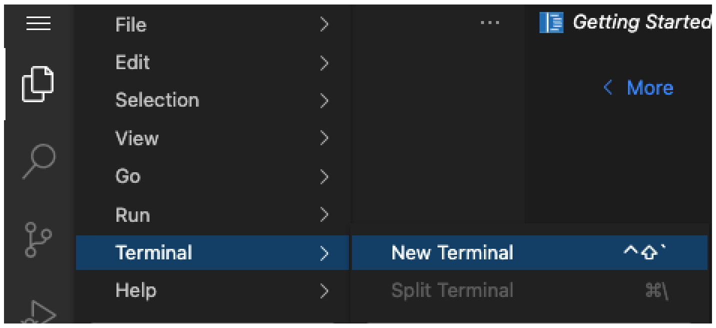


Welcome to code-server
Please log in below. Password was set from $PASSWORD.

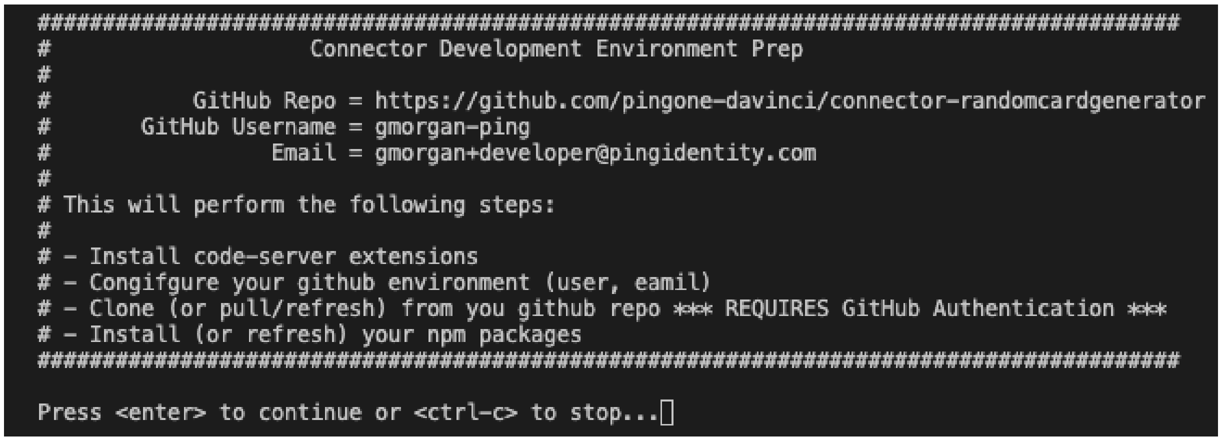
INITIALIZING THE CONNECTOR PROJECT

After launching VS Code for your newly created project, you'll need to initialize the project with your repositories' code base.

This is done by clicking the top-left menu (☰) then selecting Terminal -> New Terminal



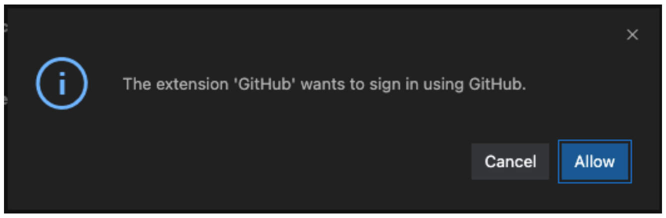
Once the terminal is opened, you'll be prompted to press Enter to kick off the Environment Prep script.



```
###########################################################################################
#                    Connector Development Environment Prep
#
#          GitHub Repo = https://github.com/pingone-davinci/connector-randomcardgenerator
#      GitHub Username = gmorgan-ping
#                Email = gmorgan+developer@pingidentity.com
#
# This will perform the following steps:
#
# - Install code-server extensions
# - Congifgure your github environment (user, eamil)
# - Clone (or pull/refresh) from you github repo *** REQUIRES GitHub Authentication ***
# - Install (or refresh) your npm packages
###########################################################################################

Press <enter> to continue or <ctrl-c> to stop...
```

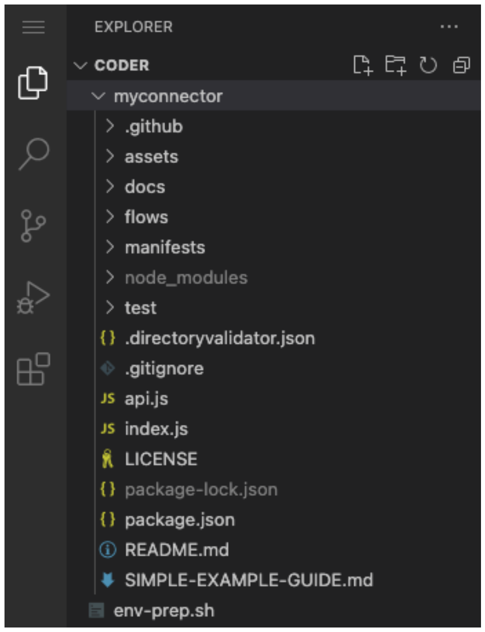
During the initial cloning of the repository, you'll receive a Github authorization prompt, click **Allow**.



The extension 'GitHub' wants to sign in using GitHub.

Once the env-prep.sh script completes, your environment is ready for connector development!

You'll notice several directories and files have been added.



Assets Directory
This is the location of your connector icon. The icon can be uploaded on the main project dashboard.

Docs Directory
As a part of the connector certification process, you'll be required to provide user documentation in markdown format before the connector is accepted by Ping Identity. See the *Contributed.md* file for more information. The *ReadMe.md* file will be autogenerated from comments in the project's manifest.js file.

Flows Directory
To ensure customers understand how to use your connector, we require you to provide reference flow(s) demonstrating its capabilities.

Manifests Directory
A connector will have at least one manifest.js file that describes the capabilities/attributes/schemas/metadata of your connector.

Test Directory
Place your connector's unit tests in this directory. Note: To ensure the integrity and functionality of your connector, Ping requires unit tests before certification.

CONNECTOR BUILDER OVERVIEW

To quickly get started with building your connector's capabilities, Ping has created a Connector Builder application that enables you to define Connector metadata, attributes, capabilities and input and output schemas.

We recommend that developers new to Connector building, use this tool to ensure that your manifest.js and index.js are well-formed.

You can launch the Connector Builder from the Project Dashboard or at https://davinci-connector.pingidentity.cloud/

GENERAL INFORMATION

Click the Update button within the General Information section to set your Connector's top-level details.

Connector ID
The connector id must be globally unique. Only upper and lowercase letters and numbers are allowed

Connector Name
This is a simple string to name your connector. If the service your connector is feature-rich to the extent that you might implement multiple connectors to avoid crowding a single connector with diverse capabilities, try naming your connector specifically with what you are planning to do.

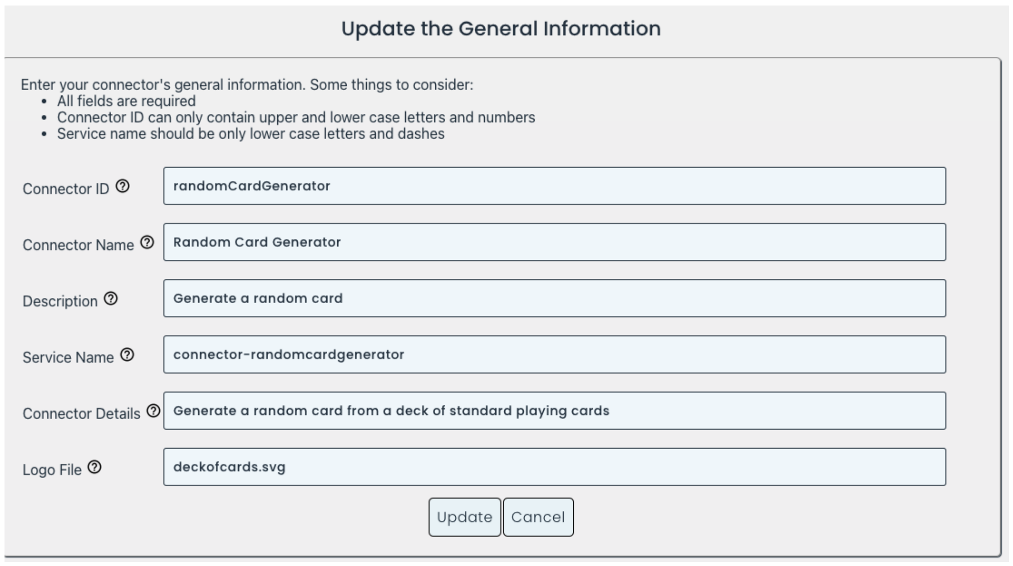
Description
A simple description to briefly explain what the connector does. Keep it brief.

Service Name
This is the ID of the Redis service to and from which events will be exchanged with the orchestration engine. This must be unique to your connector to avoid messaging issues. It is recommended to you the format *connector*-[name]. Note only lowercase letters and dashes are permitted.

Connector Details
This should be a longer description of your connector and its capabilities. It is displayed in the Admin Portal's *Connectors -> New Connector* as a description text field and has ample area to go into more details about the specifics of your connector.

Logo File

The logo file name must match what is being provided in the assets directory in the source repository.



Update the General Information

Enter your connector's general information. Some things to consider:

- All fields are required
- Connector ID can only contain upper and lower case letters and numbers
- Service name should be only lower case letters and dashes

Connector ID	randomCardGenerator
Connector Name	Random Card Generator
Description	Generate a random card
Service Name	connector-randomcardgenerator
Connector Details	Generate a random card from a deck of standard playing cards
Logo File	deckofcards.svg

Click **Update** to apply your settings

PROPERTIES

Properties are values that your connector will accept such as API keys, URLs, or input parameters.

Name

The name of the property. Only upper and lowercase letters and numbers are allowed

Display Name

The user-friendly name of the property is shown in the configuration UI

Info

This field is displayed next to the input to describe what the value should be

Preferred Control Type

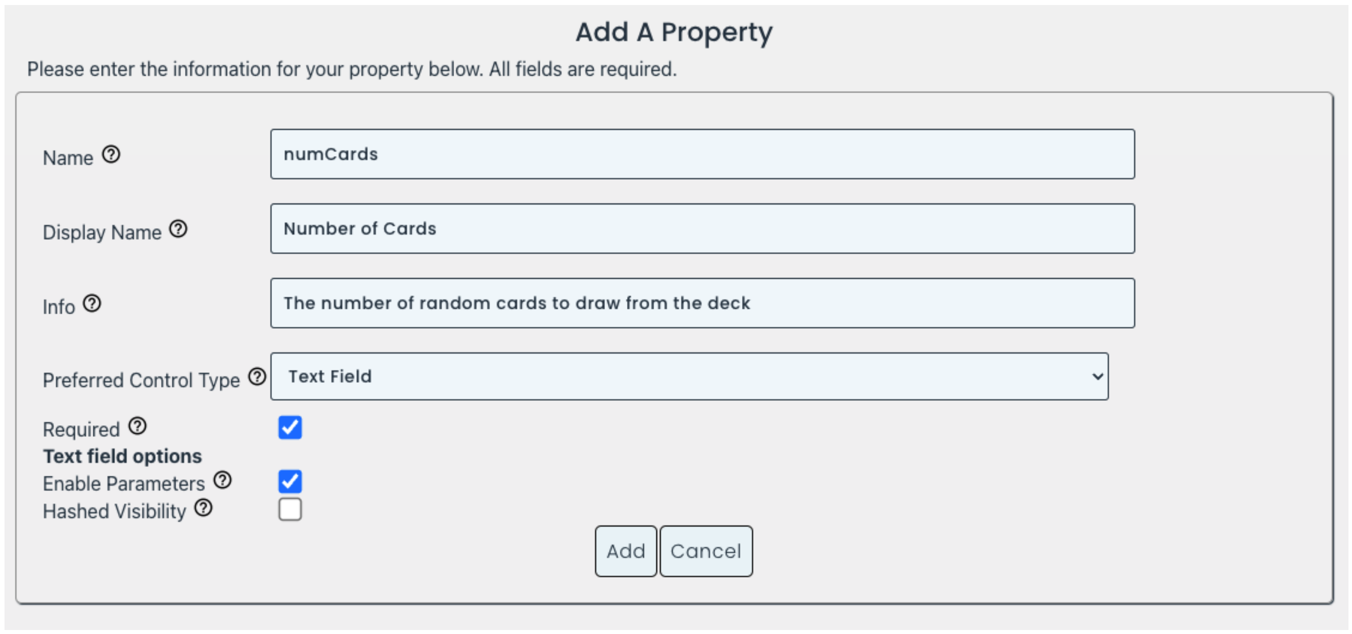
Governs the type of field that is rendered on the UI panel to assign a value to the property passed into the connector capability when it is invoked

Required

If required, the property value must be configured

Enable Parameters
The enable parameters displays the variable picker tool in the UI field. This button is represented with two curly braces **{}** and allows to pick global variables, flow variables, or output payload from connectors executing earlier in a flow

Hashed Visibility
Hide the values typed into the configuration i.e. sensitive data


Add A Property

Please enter the information for your property below. All fields are required.

Name	numCards
Display Name	Number of Cards
Info	The number of random cards to draw from the deck
Preferred Control Type	Text Field
Required	✓
Text field options	
Enable Parameters	✓
Hashed Visibility	☐

Click Add to apply your property. You may have as many properties as needed.

CAPABILITIES

The functionality that your connector will make available. Typically one capability per API call (Eg. updateUser, getRiskScore, setMFADevice)

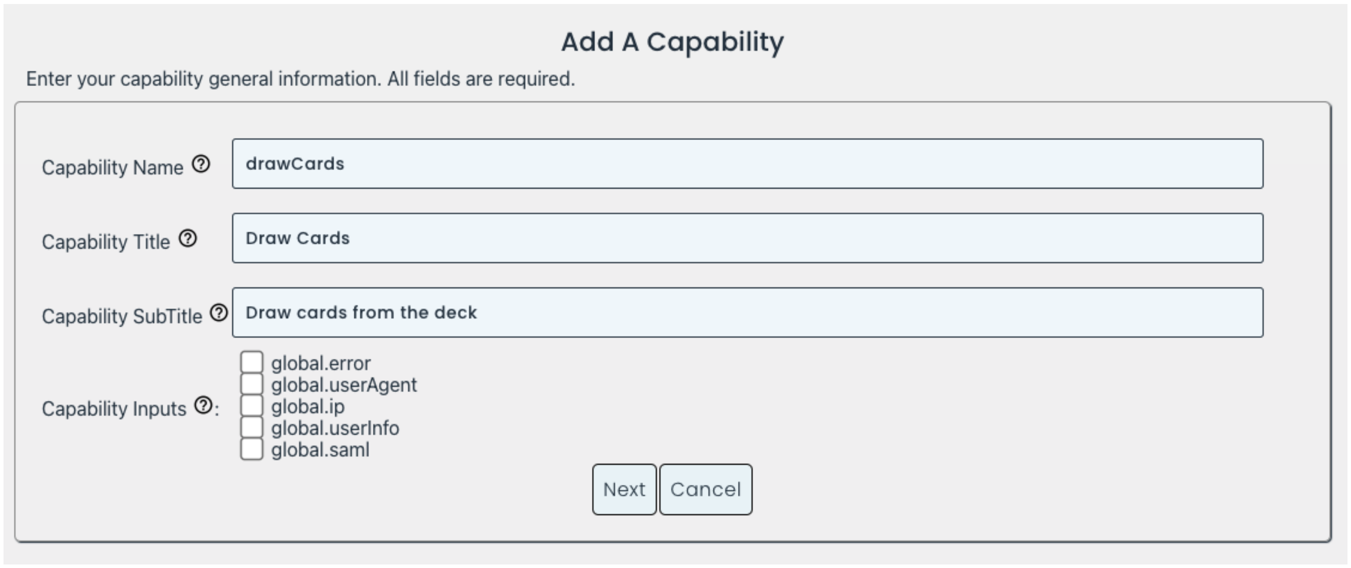
Click Add to create a Capability

Capability Name
The name of the capability. Only upper and lowercase letters and numbers are allowed

Capability Title
The user-friendly name of the capability as displayed in the connector UI

Capability Subtitle
The subtext for the title of the capability displayed in the connector UI

Capability Inputs
These are optional. They are global values that can be passed to the capability. For example, if your capability needed the user's IP address, you would select global.ip


Add A Capability

Enter your capability general information. All fields are required.

Capability Name: drawCards

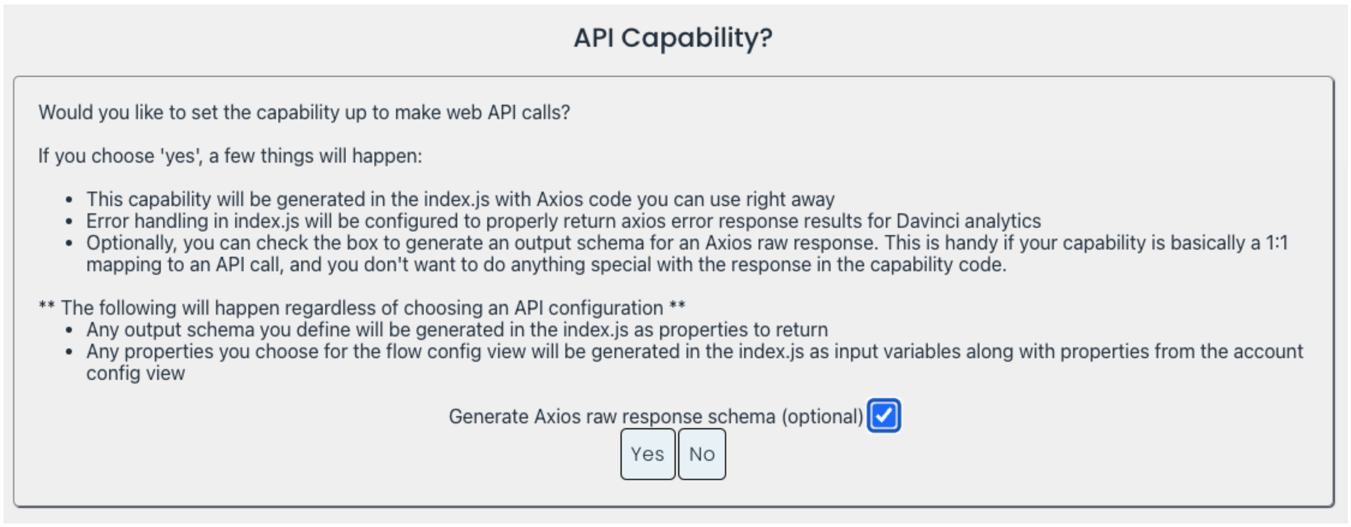
Capability Title: Draw Cards

Capability SubTitle: Draw cards from the deck

Capability Inputs:
- ☐ global.error
- ☐ global.userAgent
- ☐ global.ip
- ☐ global.userInfo
- ☐ global.saml

Click Next

Next, you'll be asked if this capability will make an API call. If it will, select "Generate Axios raw response schema' and click Yes. If your capability doesn't make API calls, select No.


API Capability?

Would you like to set the capability up to make web API calls?

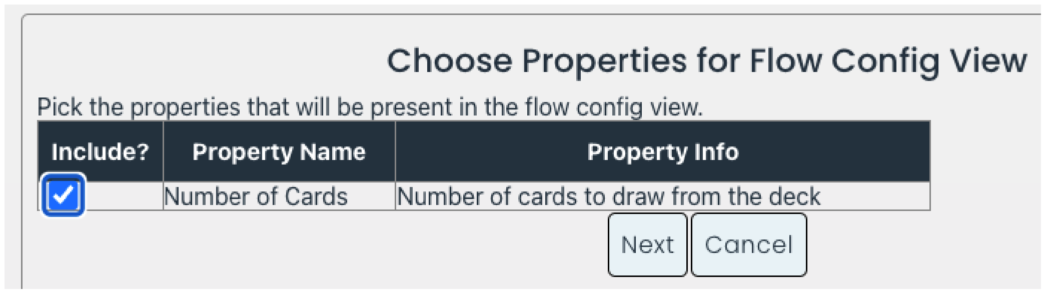
If you choose 'yes', a few things will happen:

- This capability will be generated in the index.js with Axios code you can use right away
- Error handling in index.js will be configured to properly return axios error response results for Davinci analytics
- Optionally, you can check the box to generate an output schema for an Axios raw response. This is handy if your capability is basically a 1:1 mapping to an API call, and you don't want to do anything special with the response in the capability code.

** The following will happen regardless of choosing an API configuration **
- Any output schema you define will be generated in the index.js as properties to return
- Any properties you choose for the flow config view will be generated in the index.js as input variables along with properties from the account config view

Generate Axios raw response schema (optional) ✓

Next, you may select which properties your capability will require.

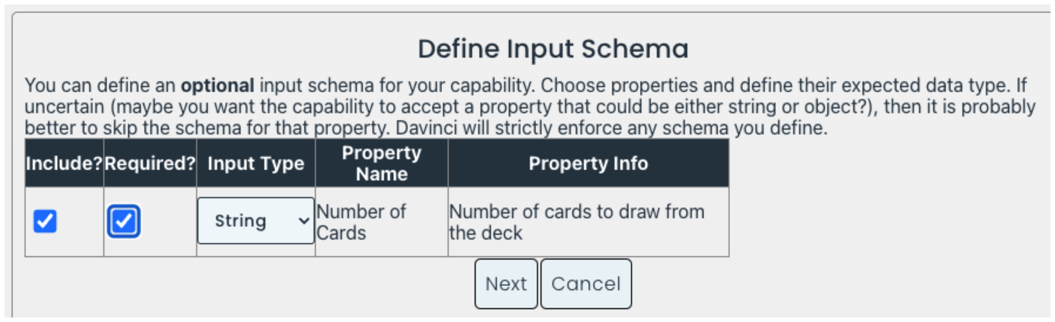


Choose Properties for Flow Config View

Pick the properties that will be present in the flow config view.

Include?	Property Name	Property Info
✓	Number of Cards	Number of cards to draw from the deck

Next, specify the optional Input Schema

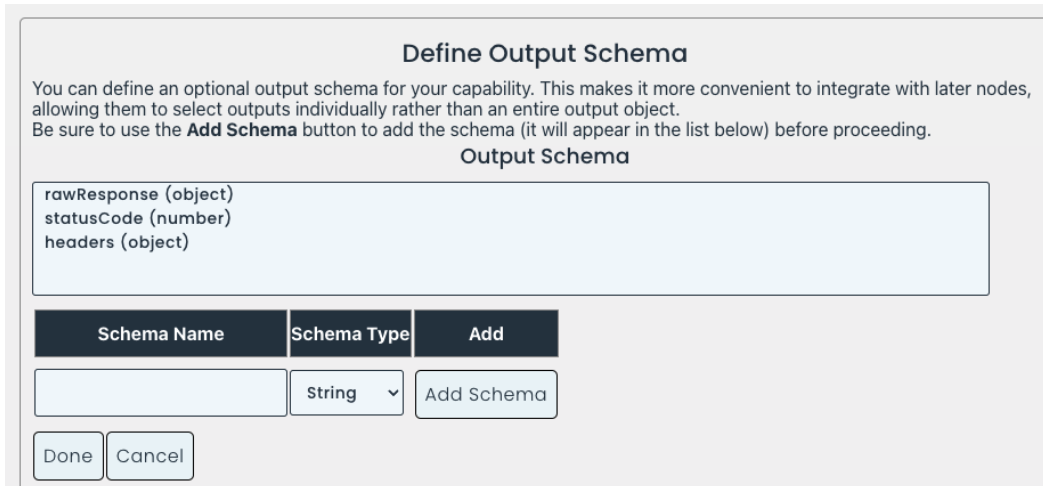


Define Input Schema

You can define an **optional** input schema for your capability. Choose properties and define their expected data type. If uncertain (maybe you want the capability to accept a property that could be either string or object?), then it is probably better to skip the schema for that property. Davinci will strictly enforce any schema you define.

Include?	Required?	Input Type	Property Name	Property Info
✓	✓	String	Number of Cards	Number of cards to draw from the deck

Finally, you can define your output schema. If you are uncertain what JSON payload your capability will return, you may leave this as the default and click done. All configuration changes can also be defined afterward in the manifest.js



Define Output Schema

You can define an optional output schema for your capability. This makes it more convenient to integrate with later nodes, allowing them to select outputs individually rather than an entire output object.
Be sure to use the **Add Schema** button to add the schema (it will appear in the list below) before proceeding.

Output Schema

```
rawResponse (object)
statusCode (number)
headers (object)
```

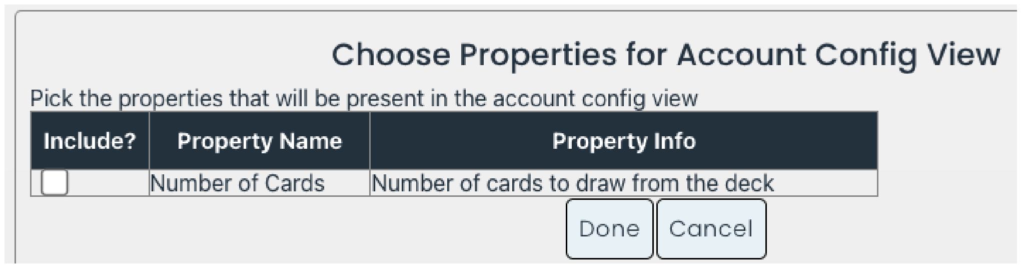
Schema Name	Schema Type	Add
	String	Add Schema

Click Done, and repeat this process if your connector has more than one capability.

ACCOUNT CONFIGURATION VIEW

If your connector has global properties you required defining such as API Base URL, Secrets, Client IDs, they can be specified here.

These properties will be displayed in the connector settings tab.



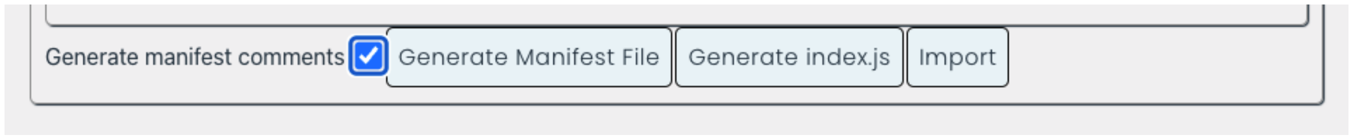
Choose Properties for Account Config View

Pick the properties that will be present in the account config view

Include?	Property Name	Property Info
☐	Number of Cards	Number of cards to draw from the deck

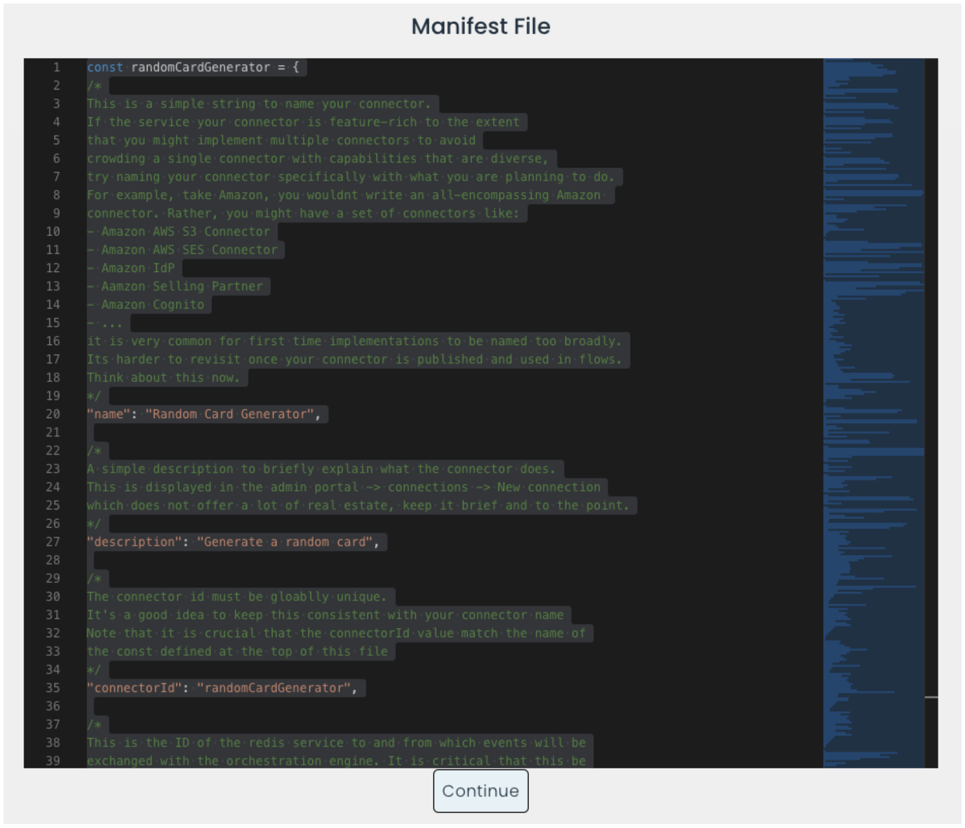
GENERATING MANIFEST AND INDEX JAVASCRIPT FILES

Once the settings have been defined within the Builder application, you can generate your manifest and index javascript files



Generate manifest comments ✓

Click the Generate Manifest File button, then copy all of the displayed text. You can replace the content of your existing manifest file with the generated one. Repeat this process for creating the index.js file



```
const randomCardGenerator = {
/*
This is a simple string to name your connector.
If the service your connector is feature-rich to the extent
that you might implement multiple connectors to avoid
crowding a single connector with capabilities that are diverse,
try naming your connector specifically with what you are planning to do.
For example, take Amazon, you wouldnt write an all-encompassing Amazon
connector. Rather, you might have a set of connectors like:
- Amazon AWS S3 Connector
- Amazon AWS SES Connector
- Amazon IdP
- Aamzon Selling Partner
- Amazon Cognito
- ...
it is very common for first time implementations to be named too broadly.
Its harder to revisit once your connector is published and used in flows.
Think about this now.
*/
"name": "Random Card Generator",

/*
A simple description to briefly explain what the connector does.
This is displayed in the admin portal -> connections -> New connection
which does not offer a lot of real estate, keep it brief and to the point.
*/
"description": "Generate a random card",

/*
The connector id must be gloablly unique.
It's a good idea to keep this consistent with your connector name
Note that it is crucial that the connectorId value match the name of
the const defined at the top of this file
*/
"connectorId": "randomCardGenerator",

/*
This is the ID of the redis service to and from which events will be
exchanged with the orchestration engine. It is critical that this be
```

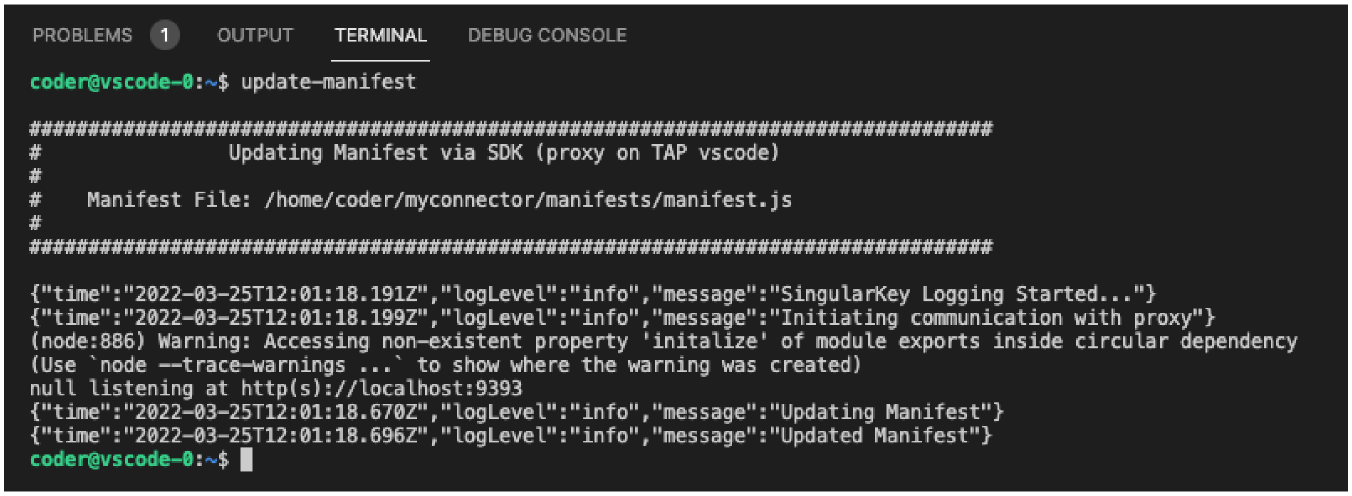
IMPORTING EXISTING MANIFEST

It is possible to import an existing manifest file and modify its properties and capabilities. At present, this feature is experimental and may not produce accurate results.

RUNNING AND DEBUGGING

Once you have implemented some or all the connector's capabilities you can launch the connector service from VS Code.

Whenever changes are made to the manifest.js file, you will need to run the script **update-manifest.sh** from the terminal. The update-manifest script updates the database that DaVinci uses to populate the connector list in the Admin Portal.

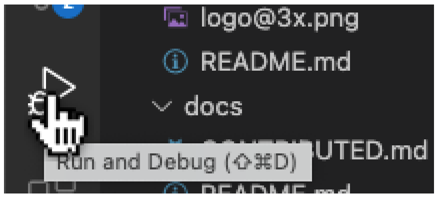


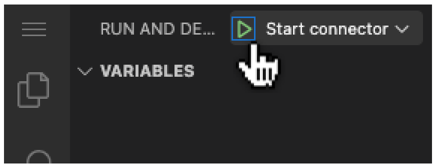
```
coder@vscode-0:~$ update-manifest

##################################################################################
#                 Updating Manifest via SDK (proxy on TAP vscode)
#
#    Manifest File: /home/coder/myconnector/manifests/manifest.js
#
##################################################################################

{"time":"2022-03-25T12:01:18.191Z","logLevel":"info","message":"SingularKey Logging Started..."}
{"time":"2022-03-25T12:01:18.199Z","logLevel":"info","message":"Initiating communication with proxy"}
(node:886) Warning: Accessing non-existent property 'initalize' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
null listening at http(s)://localhost:9393
{"time":"2022-03-25T12:01:18.670Z","logLevel":"info","message":"Updating Manifest"}
{"time":"2022-03-25T12:01:18.696Z","logLevel":"info","message":"Updated Manifest"}
coder@vscode-0:~$
```

Next, you'll need to start the connector microservice by clicking Run and Debug from the left menu, then **Start Connector**. If you have any breakpoints set, the connector when launched within a flow will break at those breakpoints allowing you to inspect your code.



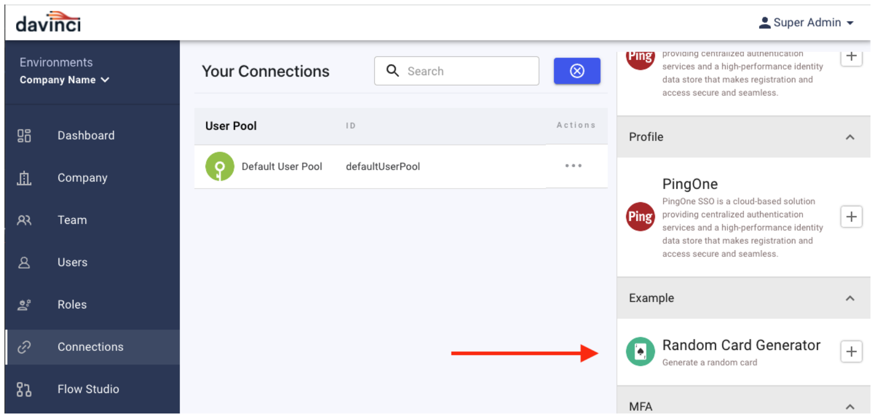


USING YOUR CONNECTOR

Once the update-manifest script has been executed and your connector is running, you will be able to use it within DaVinci.

Navigate to the project's instance of the Admin Portal then to Connections.

From the Add Connections list, your connector should appear. Note the category your connector appears in is specified in the manifest.js file.



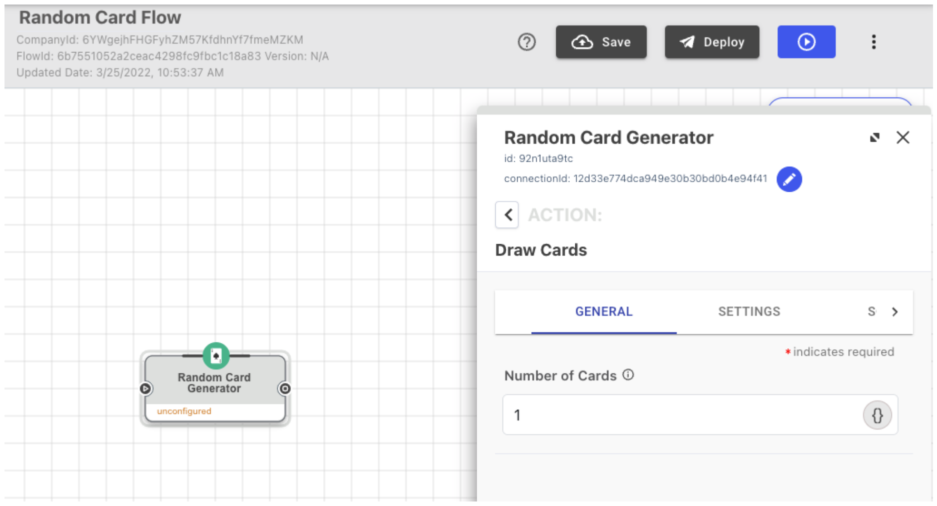
providing centralized authentication services and a high-performance identity data store that makes registration and access secure and seamless.

PingOne SSO is a cloud-based solution providing centralized authentication services and a high-performance identity data store that makes registration and access secure and seamless.

Click the **+** button to add. Give the instance a name and click **Create**

In the Flow Studio, create a new flow and add your connector.

Select the capability and enter the properties.



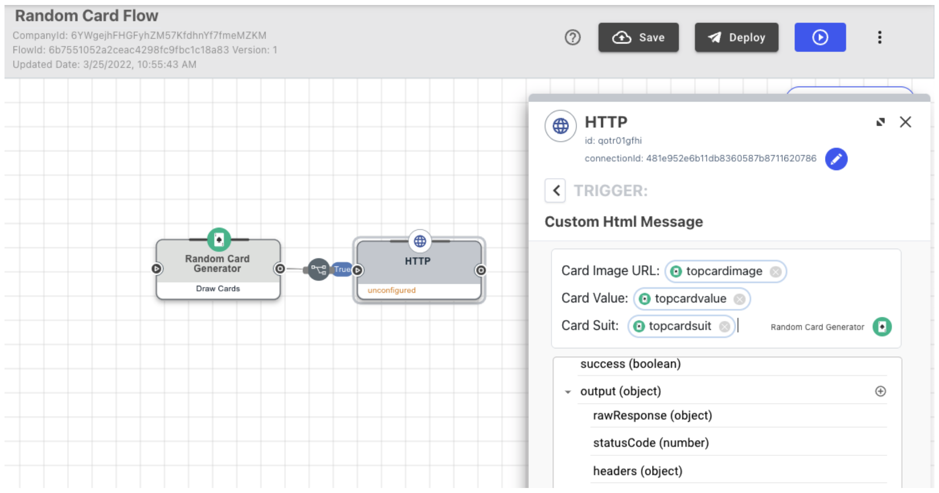
CompanyId: 6YWgejhFHGFyhZM57KfdhnYf7fmeMZKM
FlowId: 6b7551052a2ceac4298fc9fbc1c18a83 Version: N/A
Updated Date: 3/25/2022, 10:53:37 AM

id: 92n1uta9tc
connectionId: 12d33e774dca949e30b30bd0b4e94f41

*indicates required

Number of Cards: 1

Next, you may wish to display the results of your connector. In this example, an HTTP connector is used



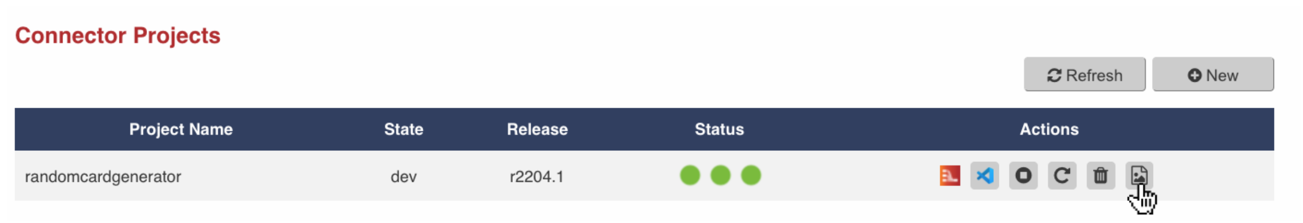
Card Image URL: topcardimage
Card Value: topcardvalue
Card Suit: topcardsuit

Save, deploy and run your flow.

If you have breakpoints, the flow will pause.

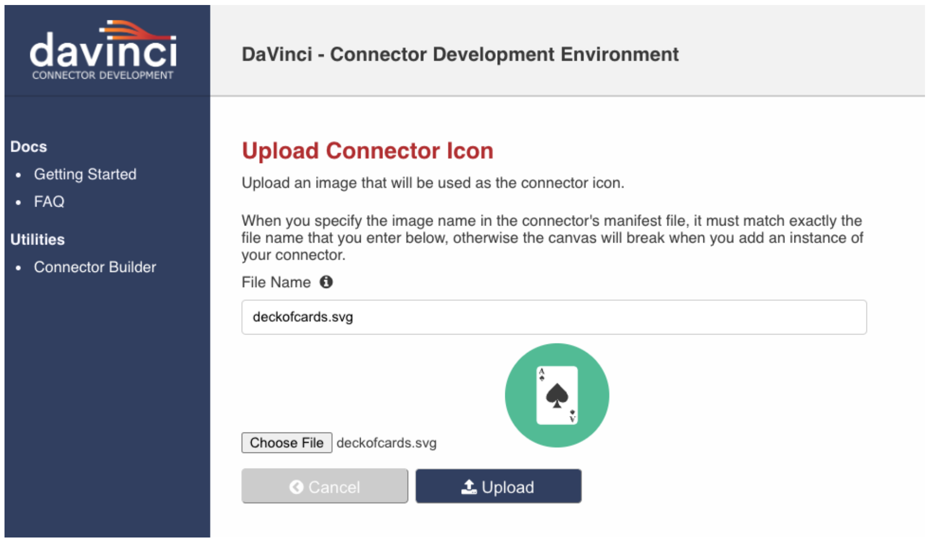
ADDING CONNECTOR ICON

Adding your connector's icon is done on the project dashboard. Click the icon button for your project



Project Name	State	Release	Status	Actions
randomcardgenerator	dev	r2204.1	● ● ●	

The file name and uploaded file name must match what is declared in your manifest.js file



Upload Connector Icon

Upload an image that will be used as the connector icon.

When you specify the image name in the connector's manifest file, it must match exactly the file name that you enter below, otherwise the canvas will break when you add an instance of your connector.

File Name: deckofcards.svg

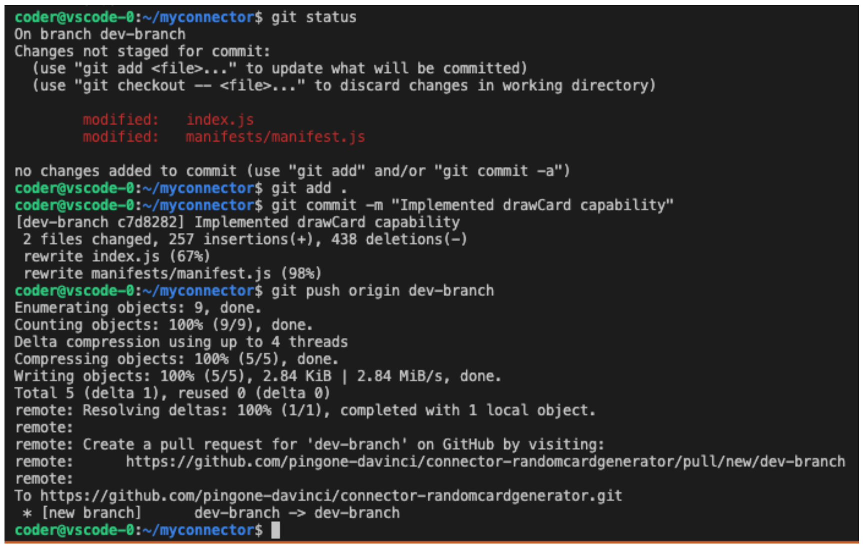
Choose File: deckofcards.svg

SAVING CHANGES BACK TO GITHUB

While the VS Code instances have mounted volumes, you'll want to routinely commit your changes back into Github.

Initially, your branch is Main, however, you are not permitted to directly update Main without performing a pull request.

For routine development, it is recommended that you create a topic branch while it is in active development, then once your connector is ready for submission perform a pull request.



```
coder@vscode-0:~/myconnector$ git status
On branch dev-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.js
        modified:   manifests/manifest.js

no changes added to commit (use "git add" and/or "git commit -a")
coder@vscode-0:~/myconnector$ git add .
coder@vscode-0:~/myconnector$ git commit -m "Implemented drawCard capability"
[dev-branch c7d8282] Implemented drawCard capability
 2 files changed, 257 insertions(+), 438 deletions(-)
 rewrite index.js (67%)
 rewrite manifests/manifest.js (98%)
coder@vscode-0:~/myconnector$ git push origin dev-branch
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.84 KiB | 2.84 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'dev-branch' on GitHub by visiting:
remote:      https://github.com/pingone-davinci/connector-randomcardgenerator/pull/new/dev-branch
remote:
To https://github.com/pingone-davinci/connector-randomcardgenerator.git
 * [new branch]      dev-branch -> dev-branch
coder@vscode-0:~/myconnector$ 
```