

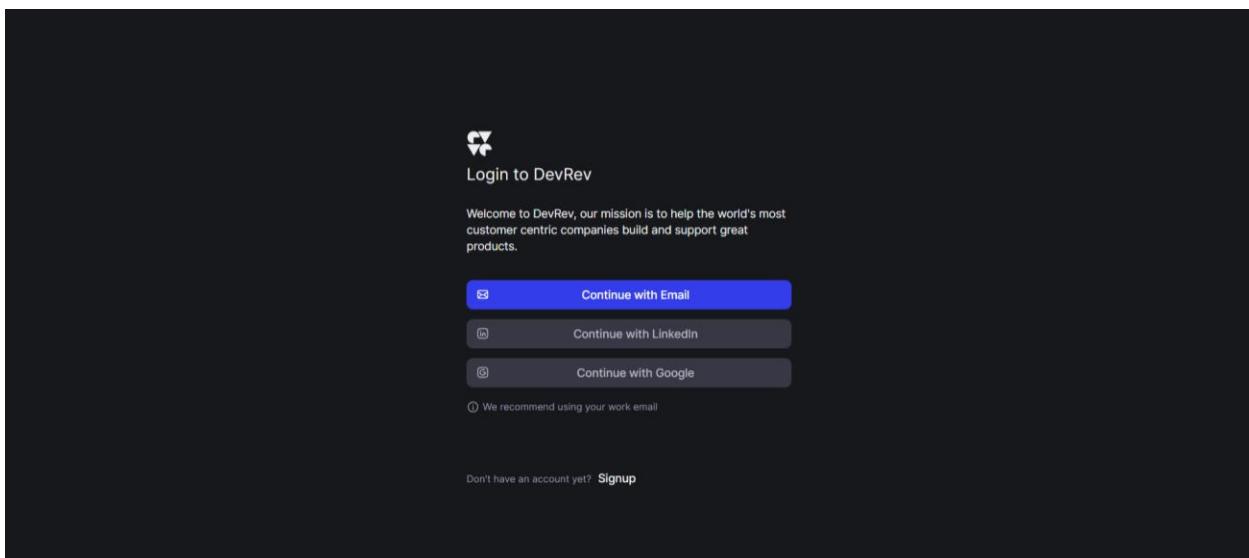
Assignment: Snap-in Development Challenge Submission

Step 1 : Utilizing the DevRev AP

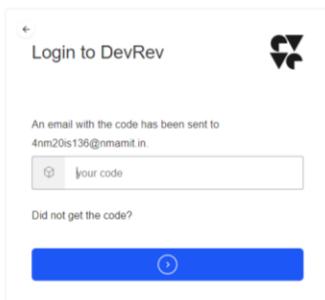
First task is to utilize the DevRev API to create a work item. Python was used as the programming language, which is compatible with the DevRev API. The Code snippet demonstrates the creation of a work item using the DevRev API.

Initial Steps to be executed:

1. Creation of DevRev account:
 - Click on the sign in option
 - Continue with Gmail



- Enter Work email: here college email id and enter the verification code sent on your email id entered.

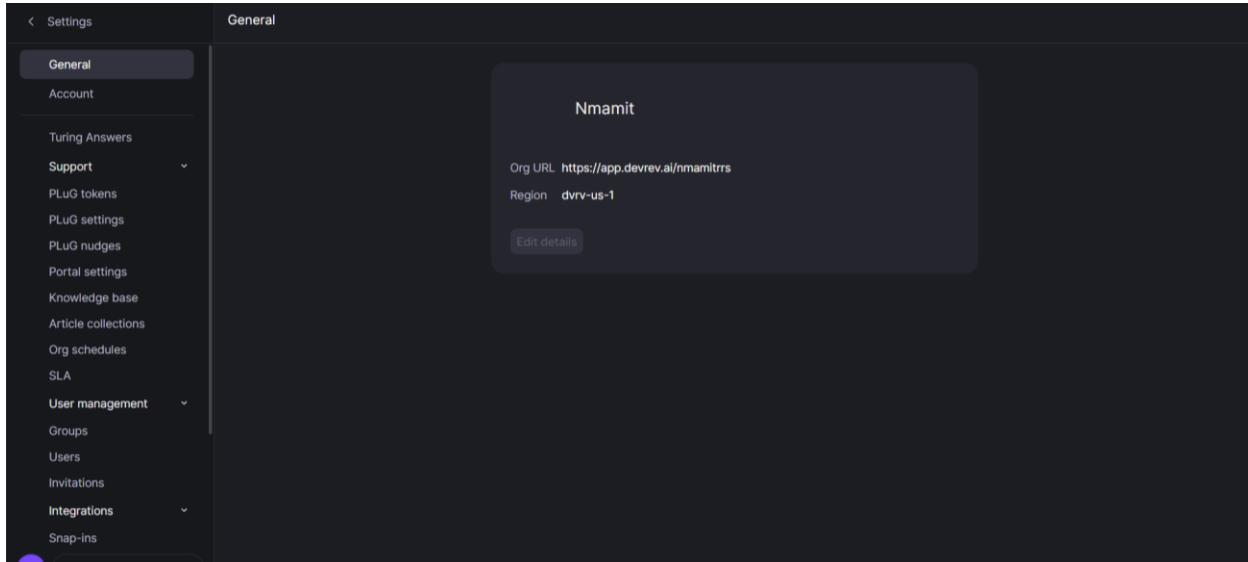


- Create a workspace or if already created, enter your workspace

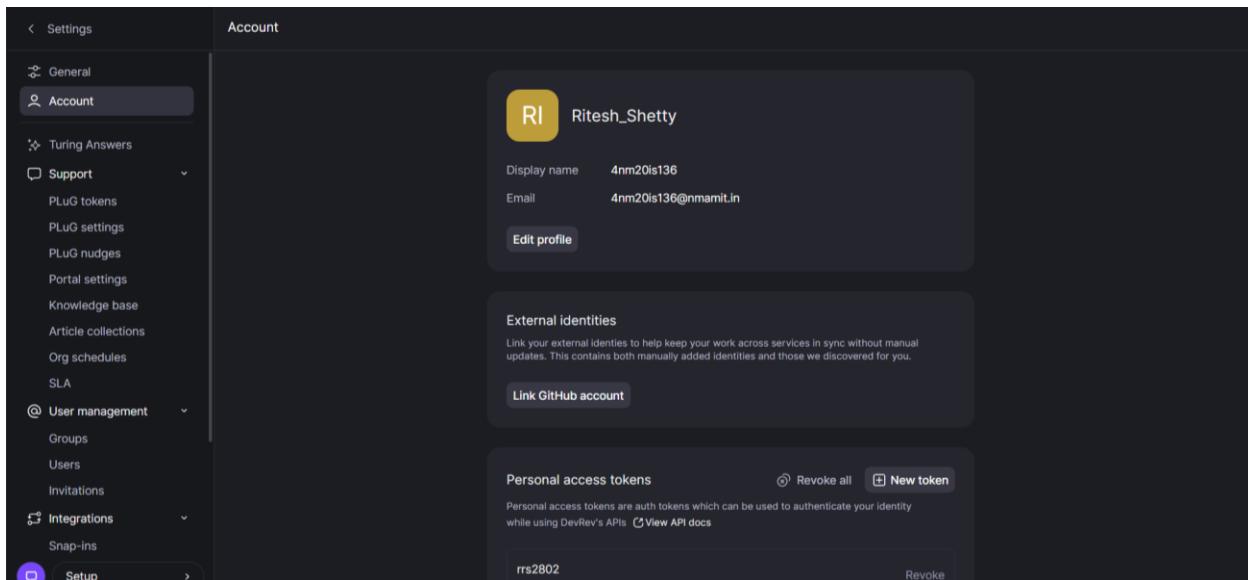
A composite screenshot of the DevRev platform. On the left, a sidebar titled "Create or join DevRev org" lists "Your dev orgs" (1 item: "Nmamit" with a "Join" button) and "Available to join" (10 items: "Nmamit", "Anikith", and "Anagha_Nmamit", each with a "Join" button). On the right, a main window titled "DevRev OneCRM Overview" displays a video player for a video titled "DevRev". The video player includes controls like play/pause, volume, and a progress bar showing 02:30.

2. Generating your PAT(Personal access token):

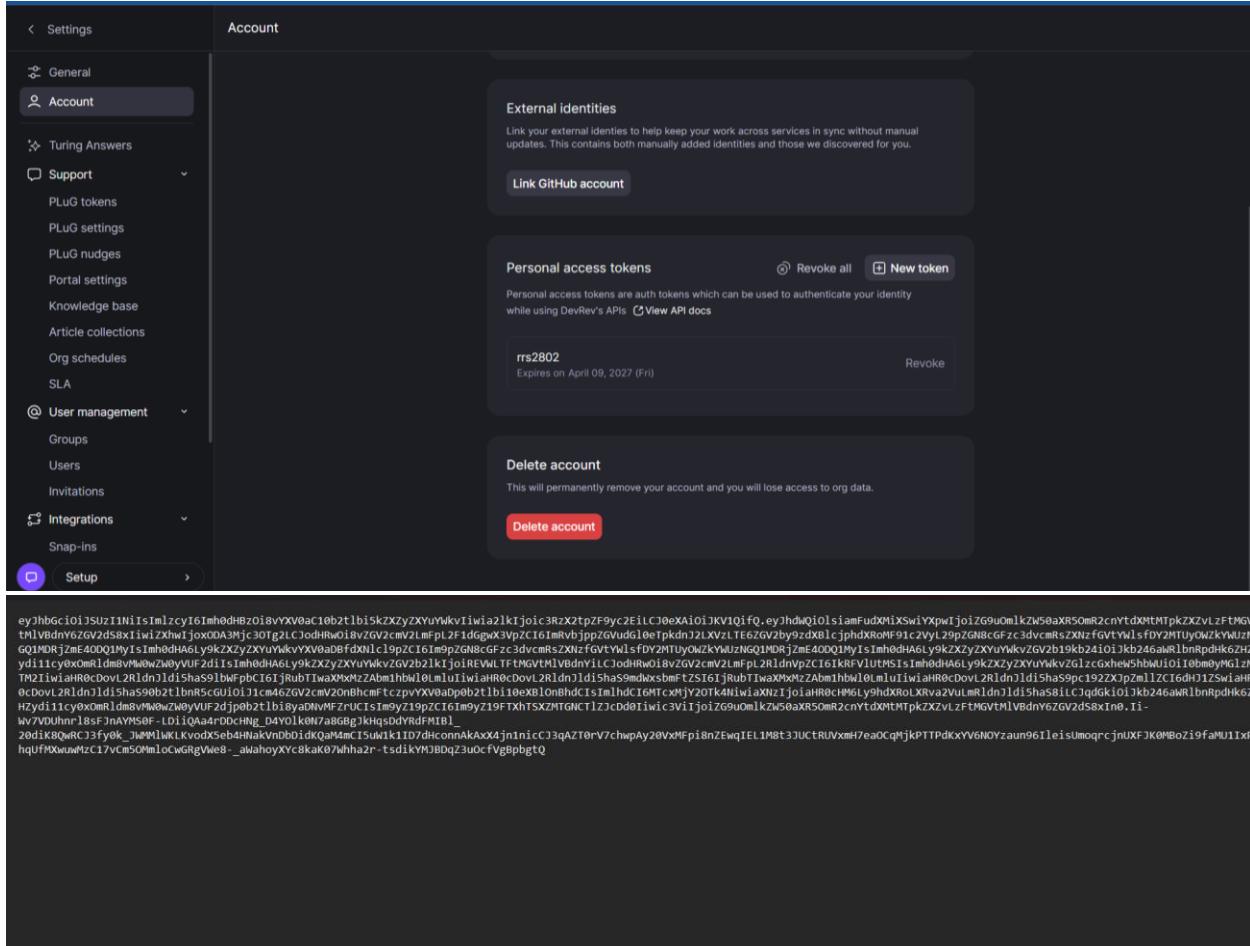
- Go to the relevant dev org in the DevRev app and navigate to Settings > Account > Personal Access Token to create a PAT.



- Use descriptive names for PATs to differentiate between them, as the token value can't be retrieved later.



- After creating the PAT, securely store it, as it can't be retrieved once you leave the page.



Brief explanation of what is PAT:

A personal access token (PAT) is used for authenticating to DevRev APIs and uniquely identifying a dev user within a dev org. It allows external third-party applications to access DevRev APIs on behalf of the corresponding dev user, with privileges matching those of the token owner. The validity duration of a PAT can be set, but it cannot be renewed; instead, a new PAT must be created and code updated to use it. For example, a VS Code plugin may rely on a user's PAT to authenticate and access DevRev APIs for specific tasks.

Explanation on what are Work Items:

A work item within DevRev is any artifact that necessitates action from a human or machine and is associated with owners and effort. Work items can be linked together, forming parent/child relationships, and can lead to other work of the same or different type. Tickets, issues, enhancements, and tasks are specific types of work items, each with distinct creation contexts and purposes.

- Tickets are created by customers or consumers and are part of the Support app.
- Issues are created by builders or maintainers and are part of the Build app.
- Enhancements serve as parent items for multiple issues, leading to desired product changes.
- Tasks are used to break down larger work into smaller pieces.

- DevRev's "Similar work" model helps prevent duplicate work during creation by identifying potential duplicates and allowing for linkage to related work items.

Code snippet and Explanation to utilize the DevRev API to create a work item:

Explanation for api1.py:

- The provided code makes use of the "requests" library to perform an HTTP GET request to a specified API endpoint which is to retrieve the user details using the unique API key.
 - The "dev_user" function is defined to make an HTTP GET request to the specified API endpoint using the provided API key for authorization. It handles potential errors and returns the response in JSON format if the status code is 200.
 - The "get_dev_user_info" function calls the "dev_user" function with the API URL and API key, and returns the response if it exists, otherwise it returns a failure message.
 - The "main" function sets the API URL and API key, then calls "get_dev_user_info" to retrieve and print the Dev User information.
 - The script uses the API URL "https://api.devrev.ai/dev-users.self" and a lengthy API key for authentication.
 - The script then executes the "main" function if it's the main module, triggering the retrieval and display of the User information.



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "OPEN EDITORS". The "DEVREV-FINAL-API" folder contains "api" and "api5.py". Inside "api", there are "api1.py" and "api7.py".
- Code Editor:** The "api1.py" file is open and visible. It contains Python code for interacting with an API, including functions like `dev_user` and `get_dev_user_info`, and a `main` function.
- Top Bar:** A tooltip for "Click here to ask Blackbox to help you code faster" is displayed above the code editor.
- Side Bar:** Various icons for file operations like copy, paste, and search are visible on the left side.

```
api > api1.py > ...
Click here to ask Blackbox to help you code faster
import requests

def dev_user(api_url, api_key):
    headers = {"Authorization": api_key}

    try:
        response = requests.get(api_url, headers=headers)
        if response.status_code == 200:
            return response.json()
        else:
            print(f"Error: {response.status_code} - {response.text}")
    except requests.exceptions.RequestException as e:
        print("Connection error:", e)
    return None

def get_dev_user_info(api_url, api_key):
    response = dev_user(api_url, api_key)
    if response:
        return response
    else:
        return "Failed to get Dev User info"

def main():
    api_url = "https://api.devrev.ai/dev-users.self"
    api_key = "eyJhbGciOiJSUzI1NiIsImIzcyI6Imh0dHBzO18vYXV0aC10b2tlb1skZXZyZXJuYWkvLiua2lkiJoic3RzX2tpZF9yc2EiLCJ0eXAiOiJKV1QiFQ.eyJhdWQiOiI1siamFuDX

    dev_user_info = get_dev_user_info(api_url, api_key)
    print("Dev User Info:", dev_user_info)

if __name__ == "__main__":
    main()
```

Output:

```
PS C:\DEVREV-FINAL-API\api> python api1.py
Dev User Info: {"dev_user": {"created_by": {"type": "dev_user", "display_id": "DEW-1", "display_name": "4nm20is136@manmit.in", "full_name": "Ritesh Shetty", "id": "don:identity:dvr-v-us-1:devo/1m0em2uAvv:devu/1", "state": "active"}, "created_date": "2024-04-09T11:45:11.536Z", "display_id": "DEW-1", "display_name": "4nm20is136", "email": "4nm20is136@manmit.in", "full_name": "Ritesh Shetty", "id": "don:identity:dvr-v-us-1:devo/1m0em2uAvv:devu/1", "modified_by": {"type": "dev_user", "display_id": "DEW-1", "display_name": "4nm20is136", "email": "4nm20is136@manmit.in", "full_name": "Ritesh Shetty", "id": "don:identity:dvr-v-us-1:devo/1m0em2uAvv:devu/1", "state": "active"}, "modified_date": "2024-04-09T11:55:00.843Z", "phone_numbers": ["+91852263369"]}, "state": "active"}
```

Explanation for api5.py:

- Importing the requests module: The code begins by importing the requests module, which allows the Python program to send HTTP requests.
- Defining the create_work function: This function is responsible for creating a work item using the DevRev API. It takes an API key and work data as input, constructs the API URL, sets the required headers, and then makes a POST request to create the work item.
- Main function: The main function is defined, which serves as the entry point of the program. It initializes the API key and then proceeds to create an issue and a ticket using the create_work function.
- API Key: The API key is a long string used for authentication and authorization to access the DevRev API which is unique to every user.
- Creating an issue: A sample issue is created with a specific title, type, applies_to_part, and owner details. The issue payload is then passed to the create_work function to create the issue via the DevRev API.
- Handling issue creation result: The result of the issue creation attempt is checked, and a success or failure message is printed based on the outcome.
- Creating a ticket: Similar to creating an issue, a sample ticket is created with specific details, and the ticket payload is then passed to the create_work function to create the ticket via the DevRev API.
- Handling ticket creation result: The result of the ticket creation attempt is checked, and a success or failure message is printed based on the outcome.
- Executing the main function: The program then checks if the script is being run as the main program, and if so, it executes the main function.
- Error handling: The code includes error handling using try-except blocks to catch any potential connection errors when making requests to the DevRev API.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows the project structure under "DEVREV-FINAL-API". The "api" folder contains "api1.py", "api2.py", and "api5.py".
- Open Editors:** Two files are open: "api5.py" and "api1.py".
- Code Editor:** The "api5.py" file is currently active, displaying Python code for creating work and managing issues using the requests library.
- Terminal:** The title bar indicates the terminal is connected to "DEVREV-FINAL-API".
- Side Bar:** Includes icons for file operations like Open, Save, Find, and Replace, as well as a "B" icon.
- Bottom Status Bar:** Shows "OUTLINE" and "TIMELINE" tabs.

```
File Edit Selection View Go Run Terminal Help < > DEVREV-FINAL-API
EXPLORER OPEN EDITORS DEVREV-FINAL-API
api > api5.py > main
    Click here to ask Blackbox to help you code faster
    1 import requests
    2
    3 def create_work(api_key, work_data):
    4     api_url = "https://api.devrev.ai/works.create"
    5     headers = {
    6         "Authorization": api_key,
    7         "Content-Type": "application/json"
    8     }
    9
   10    try:
   11        response = requests.post(api_url, headers=headers, json=work_data)
   12        if response.status_code == 201:
   13            return response.json()
   14        else:
   15            print(f"Error: {response.status_code} - {response.text}")
   16            return None
   17    except requests.exceptions.RequestException as e:
   18        print("Connection error:", e)
   19        return None
   20
   21 def main():
   22     api_key = "eyJhbGciOiJSUzI1NiImlzcyI6Imh0dHBzOi8vYXV0aC1ob2tlbi5kZXZyZXvYwkvIiwa2lkIjoic3RzX2tpF9yc2EiLCJ0eXAiOiJKV1QiQiFq.yejhdQoIolsiamFudx
   23
   24     # Creating an issue
   25     issue_title = "Ritesh Shetty CREATED a New Issue"
   26     issue_payload = {
   27         "type": "issue",
   28         "applies_to_part": "CAPL-1",
   29         "owned_by": ["don:identity:dvr-v-us-1:devo/1m0em2UAvv:devu/1"],
   30         "title": issue_title
   31     }
   32     issue_result = create_work(api_key, issue_payload)
   33     if issue_result:
   34         print("Issue created successfully.")
   35         print("Issue Details:", issue_result)
   36     else:
```



The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows the project structure with files `api1.py`, `api5.py`, and `api1.py` in the `DEVREV-FINAL-API/api` folder.
- Code Editor:** Displays Python code for creating issues and tickets using an API key. The code uses the `create_work` function to interact with the API.
- Terminal:** Shows the command `DEVREV-FINAL-API`.
- Status Bar:** Shows the file path `DEVREV-FINAL-API` and other system information.

```
File Edit Selection View Go Run Terminal Help < > DEVREV-FINAL-API

EXPLORER OPEN EDITORS api5.py 1 x api1.py 1
api > api5.py > main
21 def main():
22     api_key = "eyJhbGciOiJSUzI1NiJ9.eyJraWQiOiJyZXBza2kljoic3RzX2tpZ9yc2e1LCJ0eXAiOiJKV1QiLCJhdWQiOlsiamFudx
23
24     # Creating an issue
25     issue_title = "Ritesh Shetty CREATED a New Issue"
26     issue_payload = {
27         "type": "issue",
28         "applies_to_part": "CAPL-1",
29         "owned_by": ["don:identity:dvrus-us-1:devo/1m0em2UAvv:devu/1"],
30         "title": issue_title
31     }
32     issue_result = create_work(api_key, issue_payload)
33     if issue_result:
34         print("Issue created successfully.")
35         print("Issue Details:", issue_result)
36     else:
37         print("Failed to create issue")
38
39     # Creating a ticket
40     ticket_title = "Ritesh Shetty CREATED a New Ticket"
41     ticket_payload = {
42         "type": "ticket",
43         "applies_to_part": "PROD-1",
44         "owned_by": ["don:identity:dvrus-us-1:devo/1m0em2UAvv:devu/1"],
45         "title": ticket_title
46     }
47     ticket_result = create_work(api_key, ticket_payload)
48     if ticket_result:
49         print("Ticket created successfully.")
50         print("Ticket Details:", ticket_result)
51     else:
52         print("Failed to create ticket")
53
54 if __name__ == "__main__":
55     main()
```

Output:

```

PS C:\DEVREV-FINAL-API\api> python api1.py
Dev User Info: {'dev_user': {'created_by': {'type': 'dev_user', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'created_date': '2024-04-09T11:45:11.536Z', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'modified_date': '2024-04-09T11:45:11.536Z', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'modified_date': '2024-04-09T13:55:00.043Z', 'phone_numbers': ['+918652263369'], 'state': 'active'}}
Issue created: {'work': {'type': 'issue', 'applies_to_part': {'type': 'capability', 'display_id': 'CPL-1', 'id': 'don:core:dvr-us-1:devo/1m0em2UAvv:capability/1', 'name': 'Default Capability 1'}, 'created_by': {'type': 'dev_user', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'created_date': '2024-04-10T07:12:59.416Z', 'custom_fields': None, 'display_id': 'ISS-36', 'id': 'don:core:dvr-us-1:devo/1m0em2UAvv:issue/36', 'modified_by': {'type': 'dev_user', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'modified_date': '2024-04-10T07:12:59.416Z', 'owned_by': [{"type": "dev_user", "display_id": "DEU-1", "display_name": "4nm20is136", "email": "4nm20is136@nmamit.in", "full_name": "Ritesh Shetty", "id": "don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1", "state": "active"}], 'priority': 'P2', 'stage': {'name': 'triage'}, 'stock_schema_fragment': 'don:core:dvr-us-1:stock_sf/110623', 'title': 'Ritesh Shetty CREATED a New Issue'}}}
Ticket created successfully.
Ticket Details: {'work': {'type': 'ticket', 'applies_to_part': {'type': 'product', 'display_id': 'PROD-1', 'id': 'don:core:dvr-us-1:devo/1m0em2UAvv:product/1', 'name': 'Default Product 1'}, 'created_by': {'type': 'dev_user', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'created_date': '2024-04-10T07:13:00.155Z', 'custom_fields': None, 'display_id': 'TKT-7', 'id': 'don:core:dvr-us-1:devo/1m0em2UAvv:ticket/7', 'modified_by': {'type': 'dev_user', 'display_id': 'DEU-1', 'display_name': '4nm20is136', 'email': '4nm20is136@nmamit.in', 'full_name': 'Ritesh Shetty', 'id': 'don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1', 'state': 'active'}, 'modified_date': '2024-04-10T07:13:00.155Z', 'owned_by': [{"type": "dev_user", "display_id": "DEU-1", "display_name": "4nm20is136", "email": "4nm20is136@nmamit.in", "full_name": "Ritesh Shetty", "id": "don:identity:dvr-us-1:devo/1m0em2UAvv:devu/1", "state": "active"}]}, 'severity': 'medium', 'stage': {'name': 'queued'}, 'stock_schema_fragment': 'don:core:dvr-us-1:stock_sf/115436', 'title': 'Ritesh Shetty CREATED a New Ticket'}}
PS C:\DEVREV-FINAL-API\api>

```

Overall Output On the creation of Issues and Tickets on DevRev platform:

| Work Type | Stage | Priority | Owner |
|-----------|----------------------------|----------|-------|
| Ticket | Awaiting Customer Response | P2 | Namit |
| Ticket | Awaiting Customer Response | P2 | Namit |
| Ticket | Awaiting Customer Response | P2 | Namit |
| Ticket | In Development | P2 | Namit |

| Work Type | Stage | Priority | Owner |
|-----------|---------|----------|-------|
| Issue | Backlog | P2 | Namit |
| Issue | Backlog | P2 | Namit |
| Issue | Backlog | P2 | Namit |
| Issue | Backlog | P2 | Namit |

Step 2 : Creating and Installing a Snap-in

Second task is to create and install a snap-in. Submission should include:

1. Code for the "hello world" snap-in.
2. Instructions on how to install and use the snap-in.
3. A screenshot or demonstration of the snap-in.

Problem Statements that were executed in Snap-In Challenge are:

1. Schedule a reminder for someone by tagging them on a work item after MM HH DD time. Where MM is the minute, HH is the hour and DD is the time.
2. Command to clone a work item (issue/ticket).

3. Auto-Reply to a message if it is outside office hours. The start and end time for office hours can be specified in the input. The auto-reply message can also be taken as an input.

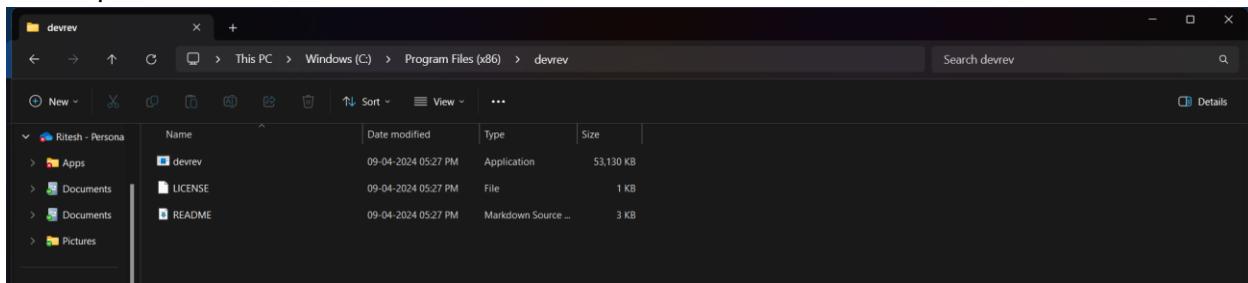
Brief explanation on what are Snap-in:

Snap-ins are collections of objects that extend DevRev's core platform value, allowing developers to develop at "arms-length" without making changes to the core platform, by interacting with DevRev objects through APIs, receiving updates through webhooks, and subscribing to external events via registered event sources such as GitHub or Slack.

Before creation of Snap-in a few prerequisites needs to be installed into the user system:

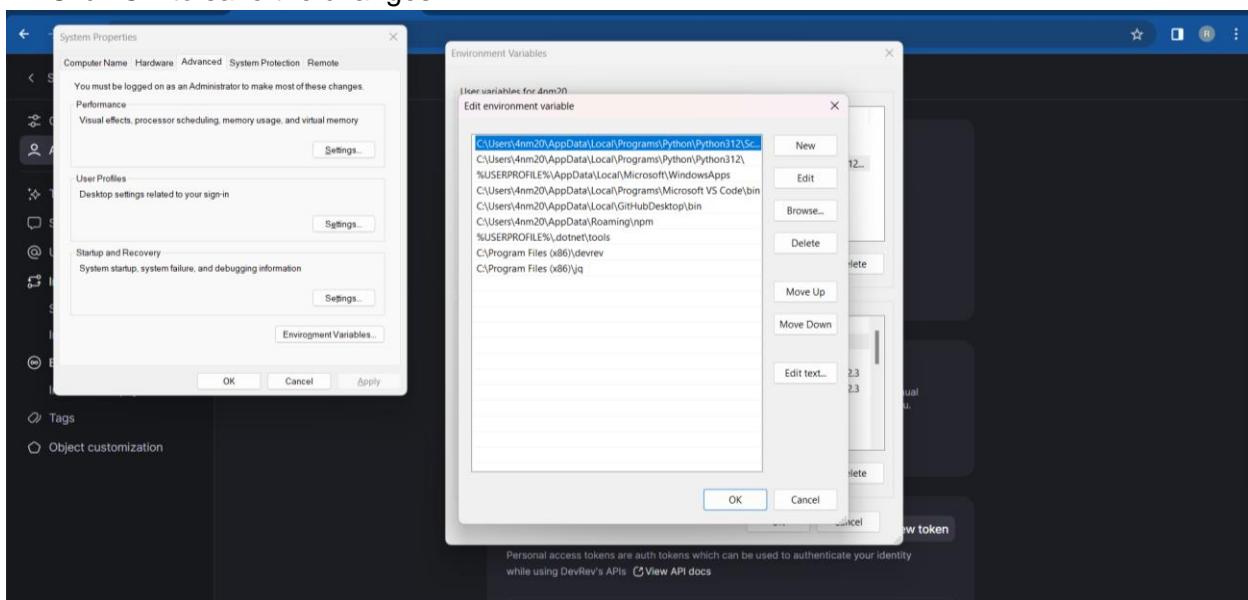
1. Install DevRev CLI:

1. Download the Windows executable for the supported architecture (Windows amd64).
2. Unzip the downloaded file.



3. Add the path to the environment variable:

- Open System Properties > This PC > Properties.
- In the System window, click Advanced system settings > Environment Variables > Path > Edit.
- In the Edit environment variable window, click New.
- Enter the path to the directory you want to add, for example: C:\Program Files\devrev\
- Click OK to save the changes.



4. Close all remaining windows by clicking OK.

```
C:\Windows\system32\cmd.exe x + v
C:\Users\4nm20>devrev --version
devrev version v0.4.6
gateway version 013f5f5dfb8c36c24366c2e80150c97eaf15e724

C:\Users\4nm20>devrev --help
Use '$ devrev profiles authenticate --org {org_slug} --usr {user_email}' to log in with an access profile
and then use CRUDL operations on all objects of the given devorg. The last used values of --org and --usr are stored in config.

Example login command:
'$ devrev profiles authenticate -o sandbox -u alice.devrev@devrev.ai'

For subsequent logins with same config as last one, use following:
'$ devrev profiles authenticate'

Usage:
  devrev [command]

Available Commands:
  artifacts      Manage artifacts
  completion    Generate the autocompletion script for the specified shell
  help          Help about any command
  marketplace  DevRev Marketplace interactions
  marketplace_category  DevRev Marketplace category interactions
  marketplace_items   Manage marketplace_items
  marketplace_submissions  Manage marketplace_submissions
  profiles       Manage locally stored access profiles
  snap_in        DevRev Snap-ins interactions
  snap_in_context  DevRev SnapInContext interaction
  snap_in_package  DevRev Snap-in package interactions
  snap_in_version  DevRev Snap-in version interactions

Flags:
  --debug        enables detailed report of all api calls
  -h, --help      help for devrev
  -v, --info      enables human readable info on stderr
  -q, --no-progress  prevents a progressbar on stderr
  -o, --org string  dev-org slug to use
  --summary      enables json summary of what was done on stderr
  --trace        enables stack traces
  -u, --usr string  dev-user email to log in with

28°C
Mostly clear
Q Search
ENG IN 10:27 PM 09-04-2024

C:\Windows\system32\cmd.exe x + v
Usage:
  devrev [command]

Available Commands:
  artifacts      Manage artifacts
  completion    Generate the autocompletion script for the specified shell
  help          Help about any command
  marketplace  DevRev Marketplace interactions
  marketplace_category  DevRev Marketplace category interactions
  marketplace_items   Manage marketplace_items
  marketplace_submissions  Manage marketplace_submissions
  profiles       Manage locally stored access profiles
  snap_in        DevRev Snap-ins interactions
  snap_in_context  DevRev SnapInContext interaction
  snap_in_package  DevRev Snap-in package interactions
  snap_in_version  DevRev Snap-in version interactions

Flags:
  --debug        enables detailed report of all api calls
  -h, --help      help for devrev
  -v, --info      enables human readable info on stderr
  -q, --no-progress  prevents a progressbar on stderr
  -o, --org string  dev-org slug to use
  --summary      enables json summary of what was done on stderr
  --trace        enables stack traces
  -u, --usr string  dev-user email to log in with
  --version      version for devrev

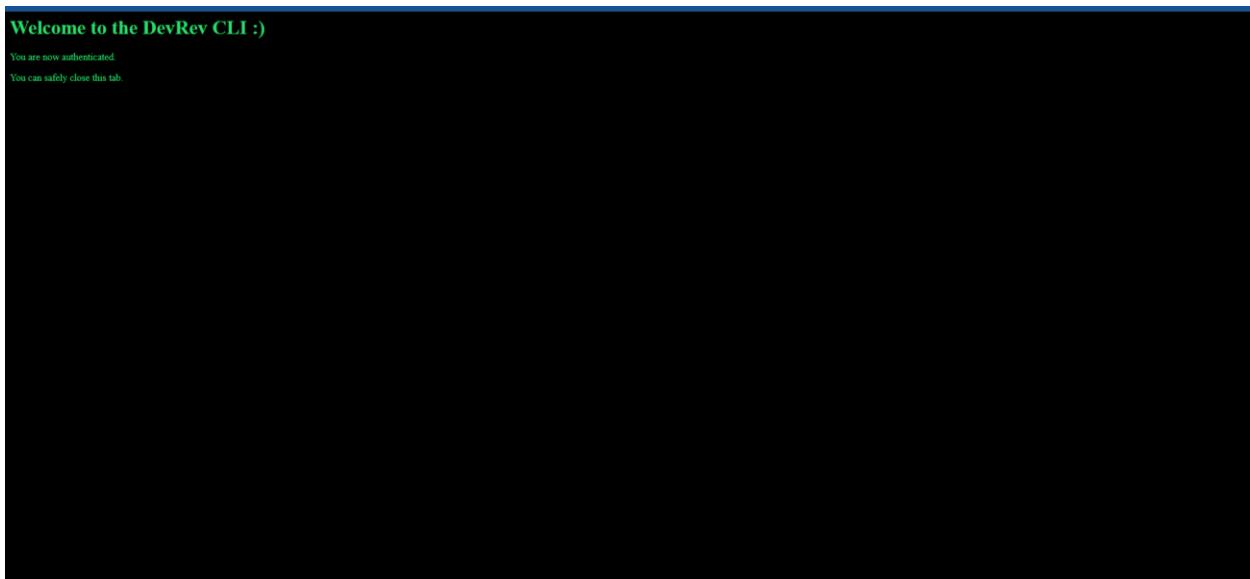
Use "devrev [command] --help" for more information about a command.

C:\Users\4nm20>$ devrev profiles authenticate --org Nmamitrrs --usr 4nm20is136@nmamit.in
'$' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\4nm20>devrev profiles authenticate --org Nmamitrrs --usr 4nm20is136@nmamit.in
https://auth.devrev.ai/authorize?access_type=offline&audience=janus&client_id=o404fpnvdAeisRnmkxOrZZ2Wt5wvzyGB&organization=org_EMxSIVLlCBNVIP7t&redirect_ur
i=http%3A%2Flocalhost%3A3000&response_type=code&scope=openid+profile+email+offline_access&state=k6NGesfMzCCRdXGJuTX4mcagpiquQC4C-YFqr5zxFwWBQOM5L6hHABYW2
U4L2vP3hdClobY-akf0ljZKGlrctg%3D%3D

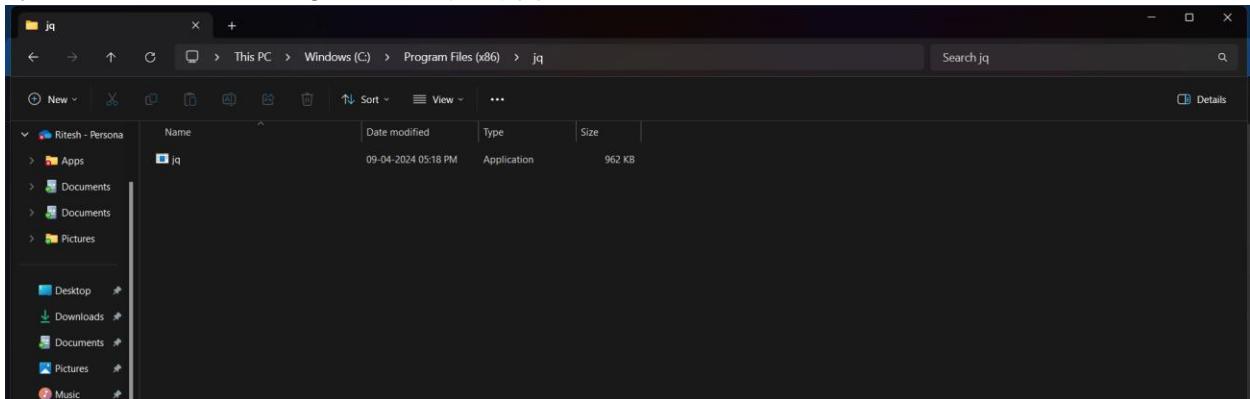
C:\Users\4nm20>
```

Output:

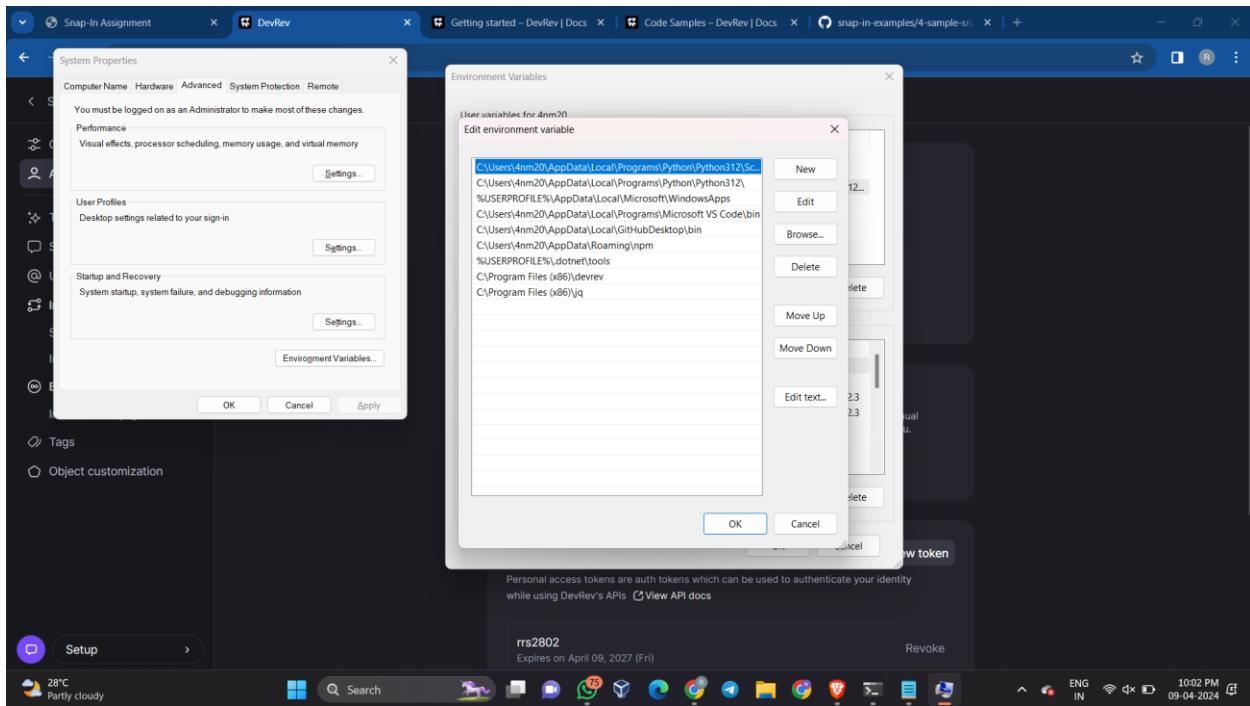


2. Install jq:

1. Open web browser and go to the official jq website at <https://stedolan.github.io/jq/download/>.
2. Scroll down to the Windows section and find the link to download the precompiled binary. It will be a .zip file.
3. Once the download is complete, extract the contents of the .zip file to a location on your local system, such as C:\Program Files(x86)\jq.



4. Next, add the directory where you extracted the jq binary to your system's PATH environment variable. This will allow you to run jq from any location in the command prompt.



5. Finally, open a new command prompt window to ensure that the changes to the PATH environment variable take effect, and then type "jq --version" to verify that jq has been successfully installed. If the installation was successful, you should see the version of jq that you downloaded.

```

Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\4nm20>jq --version
jq-1.7.1

C:\Users\4nm20>-

```

3. Install DevRev SDK:

1. Set up an environment variable named DEVREV_TOKEN. This token will be used as an authentication token by default. You can also pass in the URL and token in client.setup() as a parameter. This token is also required to run tests.
2. Open your terminal or command prompt.
3. Run the following command to install the @devrev/typescript-sdk package using npm:

```
npm install @devrev/typescript-sdk
```

4. Verify that the installation was successful by checking the package.json file in your project. Make sure that your project's package.json contains the "type":"module" setting, as the SDK is currently in beta.

The screenshot shows the GitHub repository page for `@devrev/typescript-sdk`. The top navigation bar includes tabs for `Readme`, `Code` (Beta), `4 Dependencies`, `0 Dependents`, and `23 Versions`. The `Readme` tab is highlighted. On the left, there are sections for **Authorization** and **Installation**. The **Authorization** section contains a note about setting an environment variable `DEVREV_TOKEN`. The **Installation** section shows the command `npm install @devrev/typescript-sdk`. To the right, there is an **Install** button with the command `> npm i @devrev/typescript-sdk`, a **Repository** link to `github.com/devrev/typescript-sdk`, a **Homepage** link to `github.com/devrev/typescript-sdk#read...`, a **Weekly Downloads** chart showing 258 downloads, a **Version** of 1.1.23, a **License** of MIT, an **Unpacked Size** of 703 kB, and a **Total Files** count of 25.

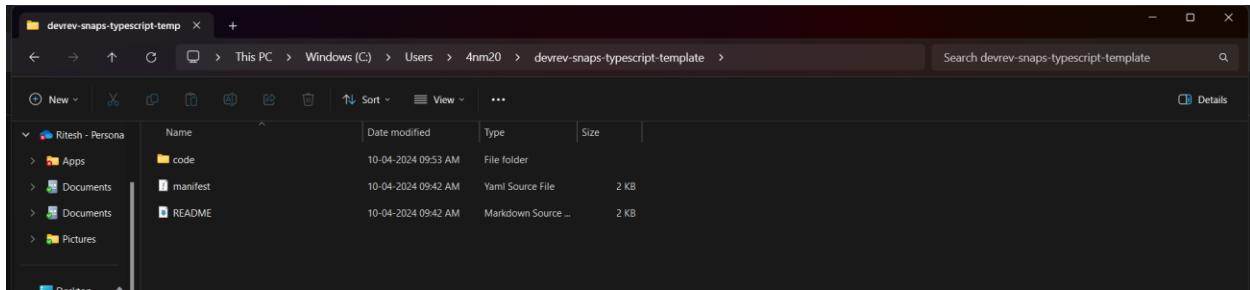
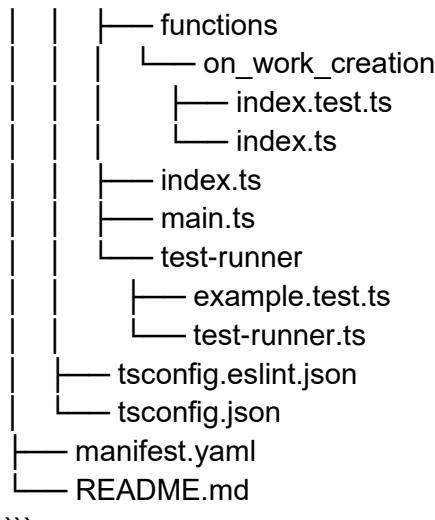
To create a snap-in using the provided content, you can follow these steps:

1. Open your terminal or command prompt.
2. Run the following command to initialize a snap-in template:

```
```
devrev snap_in_version init
````
```

3. After running the command, a new folder named `devrev-snaps-typescript-template` will be created in the filesystem. This folder will contain a `manifest.yaml` file and a `code` folder with the following structure:

```
```
devrev-snaps-typescript-template/
├── code
│ ├── babel.config.js
│ ├── jest.config.js
│ ├── nodemon.json
│ └── package.json
└── src
 ├── fixtures
 │ └── on_work_created_event.json
 └── function-factory.ts
````
```



4. The manifest.yaml file defines the resources to be created on the DevRev platform.
5. The code folder contains sample starter code for snap-ins. For detailed information on how to get started, you can refer to the starter example repo.

Creating a snap-in package:

1. Run the following command to create a snap-in package, replacing "helloworld" with a globally unique slug:

```
devrev snap_in_package create-one --slug my-first-snap-in | jq .
```

- Note: If the provided slug is already taken, a conflict error will occur. In this case, provide a different unique slug.

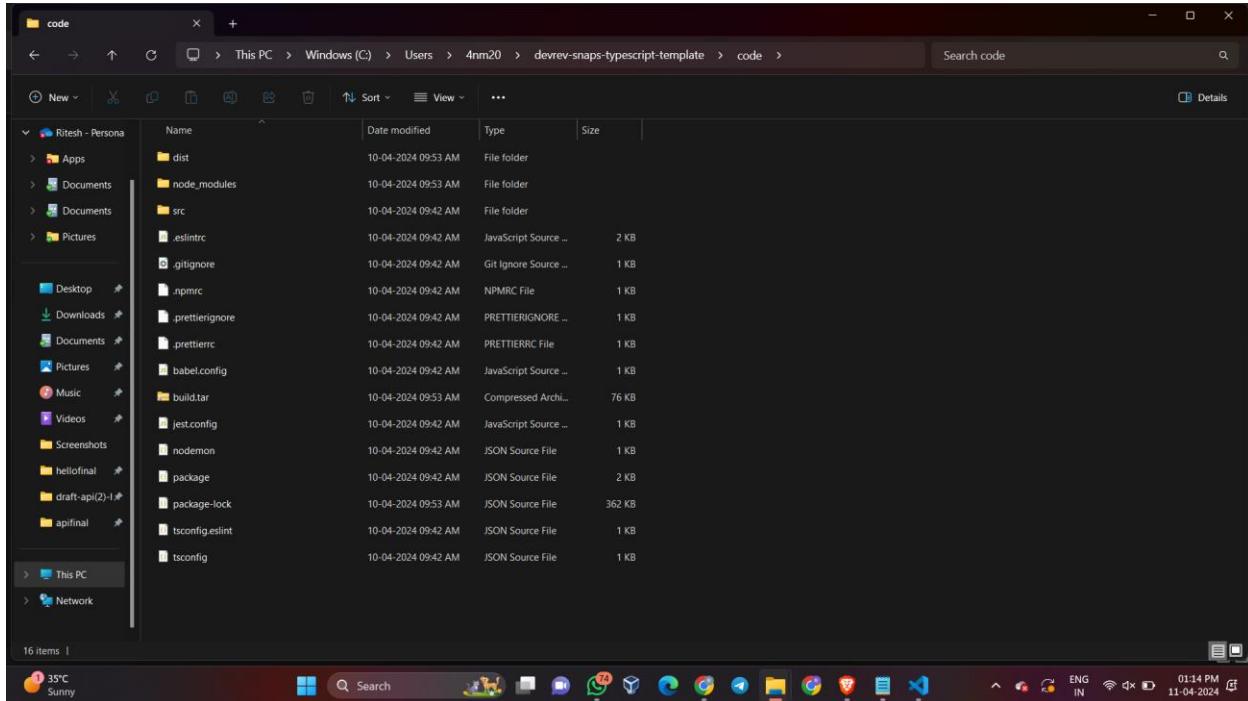
2. Upon successful creation, the CLI automatically stores the package ID in its context corresponding to the slug.

Creating a snap-in version:

1. Run the following command to create a snap-in version, specifying the path to the devrev-snaps-typescript-template folder:

```
devrev snap_in_version create-one --path ./devrev-snaps-typescript-template | jq .
```

2. The output will include the version ID and other relevant information. The CLI automatically stores the version ID in its context.



Explanation of code in Snap-in manifest file in context to Snap-in Development:

1. Version and Information:

- The "version" field specifies the version of the snap-in manifest.
- "name" and "description" provide a name and description for the snap-in.

2. Service Account:

- The "service_account" section specifies the display name of the service account that will take actions using the token of this service account.

3. Event Sources:

- The "event_sources" section defines the event sources that the snap-in will listen to.
- In this case, it references an organization event source named "devrev-webhook" that supports the "work_created" event type.

4. Inputs:

- The "inputs" section defines the organization level inputs for the snap-in.
- It includes input fields such as text, boolean, and array types, along with their descriptions, default values, and UI display names.

5. Functions:

- The "functions" section lists the functions available in the snap-in, along with their descriptions.
- In this case, it includes "function_1" and "function_2" with their respective descriptions.

6. Automations:

- The "automations" section specifies the automations that the snap-in will perform.
 - It includes an automation named "convergence_automation_devrev" that is triggered by the "work_created" event from the "devrev-webhook" event source and calls "function_1".

7. Commands:

- The "commands" section defines commands that can be used to trigger specific functions within the snap-in.
 - In this case, it includes a command named "helloworld" that triggers "function_2" and provides usage hints and supported object types.

The screenshot shows the Microsoft DevRev extension integrated into the Visual Studio Code interface. The left sidebar displays the project structure under 'Helloworld' with files like .eslintrc.js, .gitignore, and manifest.yaml selected. The main editor area shows the manifest.yaml configuration file with code for functions, automations, commands, and surfaces. The status bar at the bottom provides system information and the date.

```
manifest.yaml
code > manifest.yaml > {} event_sources > [ ] organization > {} > {} config > [ ] event_types > 0
48
49
50 # Functions reference: https://docs.devrev.ai/snap-ins/references/functions.
functions:
51   - name: function_1
52     description: Function to create a timeline entry comment on a DevRev work item created.
53   - name: function_2
54     description: Function to create a timeline entry comment on a DevRev work item on which comment is added.
55
56 # Automations reference: https://docs.devrev.ai/snap-ins/concepts#automation.
automations:
57   - name: convergence_automation_devrev
58     source: devrev-webhook
59     event_types:
60       - work_created
61     function: function_1
62
63 commands:
64   - name: helloworld
65     namespace: devrev
66     description: Command to trigger function to add comment to this work item.
67     surfaces:
68       - surface: discussions
69         object_types:
70           - issue
71           - ticket
72     usage_hint: "Command to add comment to this work item."
73     function: function_2
74
75
```

Explanation of code in Snap-in Functions and Logic defined for Snap-in Development:

1. Importing the Required Module:

- The code begins with importing the necessary module, in this case, "@devrev/typescript-sdk".

2. Defining the handleEvent Function:

- The "handleEvent" function is defined as an asynchronous function that takes an "event" parameter.

3. Accessing Event Context and Setup:

- Within the "handleEvent" function, the necessary event context values such as "service_account_token" and "devrev_endpoint" are accessed for authentication and API endpoint setup.

4. Setting up the DevRev SDK Client:

- The "client.setup" method is used to set up the DevRev SDK client with the specified API endpoint and authentication token.

5. Handling Event Data:

- Relevant event data is extracted based on the specific event payload and input data, and any necessary processing is performed.

6. Creating and Sending a Request:

- A request body is constructed based on the event data and a specific action, such as creating a timeline comment.
- The constructed request is sent using the DevRev SDK client, and the response is awaited.

7. Defining the run Function:

- The "run" function is defined as an asynchronous function that takes an array of "events" as a parameter.

8. Iterating Through Events:

- Within the "run" function, a loop iterates through the array of events, calling the "handleEvent" function for each event.

9. Logging and Exporting:

- Relevant information, such as event details and response data, is logged or processed as needed.
- The "run" function is exported as the default export, making it available for use in other modules.

Steps and Commands involved to Run Snap-in:

1. Packaging the Code:

- Run the following commands to package the code:
 - `npm install`
 - `npm run build`
 - `npm run package`
- Ensure that the process succeeds, and a "build.tar.gz" file is created, which can be provided when creating the snap_in_version.

```

PS C:\DEVREV-FINAL-HELLO\helloworld> cd code
PS C:\DEVREV-FINAL-HELLO\helloworld> npm install
> npm run build
> npm run package

PS C:\DEVREV-FINAL-HELLO\helloworld> devrev-snaps-typescript-template@1.0.0 build
> rimraf ./dist && tsc

> devrev-snaps-typescript-template@1.0.0 package
> tar -cvzf build.tar.gz dist package.json package-lock.json .npmrc

a dist
a dist/function-factory.dts
a dist/function-factory.js
a dist/functions
a dist/index.dts
a dist/index.js
a dist/main.dts
a dist/main.js
a dist/test-runner
a dist/test-runner/example.test.d.ts
a dist/test-runner/example.test.js
a dist/test-runner/test-runner.d.ts
a dist/test-runner/test-runner.js
a dist/functions/function_1
a dist/functions/function_2
a dist/functions/function_2/index.d.ts
a dist/functions/function_2/index.js
a dist/functions/function_2/index.test.d.ts
a dist/functions/function_2/index.test.js
a dist/functions/function_1/index.d.ts
a dist/functions/function_1/index.js
a dist/functions/function_1/index.test.d.ts
a dist/functions/function_1/index.test.js
a package.json
a package-lock.json
a .npmrc
PS C:\DEVREV-FINAL-HELLO\helloworld>

```

2. Authentication:

- To authenticate, run the command:

```

**devrev profiles authenticate --org <DevOrg-slug-name> --usr <your-email@example.com>**

```

- Replace `<DevOrg-slug-name>` with the unique slug name of your DevOrg, and `<your-email@example.com>` with your registered user email for the profile.

```

PS C:\DEVREV-FINAL-HELLO\code> devrev profiles authenticate --org 4nm20is136@nammit.in
PS C:\DEVREV-FINAL-HELLO\code> PS C:\DEVREV-FINAL-HELLO\code> devrev profiles authenticate --org 4nm20is136@nammit.in
https://auth.devrev.ai/authorize?access_type=offline&audience=janus&client_id=o4o4fpnvdAisRmnik0rZZ2kf5wzvayGB&organization=org_EMxSIVLLcBNVlp7t&redirect_uri=http%3A%2F%2Flocalhost%3A3000&response_type=code&scope=openid+profile+email+offline_access&state=yUmURjhgR7X19futv_a_bCF5zLwqJxF9H0Ed3M977tmM0eAo6zxMWh_jgK00uHA8p5_CpajVdDF1extuqcfg3D%3D

```

Welcome to the DevRev CLI :)

You are now authenticated.

You can safely close this tab

3. Snap-in Package:

- Create a snap-in package by running the command:

```
```
```

**devrev snap\_in\_package create-one --slug <slug name>| jq .**

```
PS C:\DEVREV-FINAL-HELLO\helloworld\code> devrev snap_in_package create-one --slug helloworld3 | jq .
{
 "snap_in_package": {
 "created_by": {
 "type": "dev_user",
 "display_handle": "4nm20is136",
 "display_id": "DEVO-1",
 "display_name": "4nm20is136",
 "email": "4nm20is136@mailto.in",
 "full_name": "Ritesh Shetty",
 "id": "don:identity:devr-us-1:devo/1m0em2uavv:devu/1",
 "id_vx": "don:DEV-1m0em2uavv:dev_user:DEVO-1",
 "state": "active",
 "thumbnail": "https://api.devrev.ai/internal/display-picture/Ritesh_Shetty.svg"
 },
 "created_date": "2024-04-10T05:22:22.876Z",
 "display_id": "snap_in_package-26227783-b25f-497c-8c75-e6b8839f726a",
 "id": "don:integration:devr-us-1:devo/1m0em2uavv:snap_in_package/26227783-b25f-497c-8c75-e6b8839f726a",
 "modified_by": {
 "type": "dev_user",
 "display_handle": "4nm20is136",
 "display_id": "DEVO-1",
 "display_name": "4nm20is136",
 "email": "4nm20is136@mailto.in",
 "full_name": "Ritesh Shetty",
 "id": "don:identity:devr-us-1:devo/1m0em2uavv:devu/1",
 "id_vx": "don:DEV-1m0em2uavv:dev_user:DEVO-1",
 "state": "active",
 "thumbnail": "https://api.devrev.ai/internal/display-picture/Ritesh_Shetty.svg"
 },
 "modified_date": "2024-04-10T05:22:22.876Z"
 }
}
```

PS C:\DEVREV-FINAL-HELLO\helloworld\code> Share Code Link Explain Code Comment Code Find Bugs Code Chat Blackbox Search Error

Ln 3, Col 1 Spaces: 4 UTF-8 CRLF ⓘ TypeScript ⓘ Go Live Blackbox ⓘ

31°C Sunny

### 4. Snap-in Version:

- Create a snap-in version by running the command:

```
```
```

devrev snap_in_version create-one --manifest manifest.yaml --archive build.tar.gz | jq .

```
PS C:\DEVREV-FINAL-HELLO\helloworld\code> devrev snap_in_version create-one --manifest manifest.yaml --archive build.tar.gz | jq .
{
  "snap_in_version": {
    "automations": [
      {
        "event_source": "devrev-webhook",
        "event_types": [
          "work_created"
        ],
        "function": "function_1",
        "name": "convergence_automation_devrev"
      }
    ],
    "commands": [
      {
        "description": "Command to trigger function to add comment to this work item.",
        "function": "function_2",
        "name": "helloworld",
        "namespace": "devrev",
        "surfaces": [
          {
            "object_types": [
              "issue",
              "ticket"
            ],
            "surface": "discussions"
          }
        ]
      }
    ]
  }
}
```

PS C:\DEVREV-FINAL-HELLO\helloworld\code> Share Code Link Explain Code Comment Code Find Bugs Code Chat Blackbox Search Error

Ln 3, Col 1 Spaces: 4 UTF-8 CRLF ⓘ TypeScript ⓘ Go Live Blackbox ⓘ

31°C Sunny

5. Show Snap-in Details:

- To show the snap-in details, run the command:

```

**devrev snap\_in show [id] | jq .**

```

- `'[id]'` is optional. If not provided, it uses the ID stored in the local profile.

```
PS C:\DEVREV-FINAL-HELLO\helloworld\code> devrev snap_in_package show | jq .
{
  "snap_in_package": {
    "created_by": {
      "type": "dev_user",
      "display_handle": "4nm20is136",
      "display_id": "DEVO-1",
      "display_name": "4nm20is136",
      "email": "4nm20is13@mainit.in",
      "full_name": "Ritesh Shetty",
      "id": "don:identity:devr-us-1:devo/1m0em2UAvv:devu/1",
      "id_v1": "don:DEV-1m0em2UAvv:dev_user:DEVO-1",
      "state": "active",
      "thumbnail": "https://api.devrev.ai/internal/display-picture/Ritesh_Shetty.svg"
    },
    "created_date": "2024-04-10T05:22:22.876Z",
    "display_id": "snap_in_package-26227783-b25f-497c-8c75-e6b8839f726a",
    "id": "don:integration:devr-us-1:devo/1m0em2UAvv:snap_in_package/26227783-b25f-497c-8c75-e6b8839f726a",
    "modified_by": {
      "type": "dev_user",
      "display_handle": "4nm20is136",
      "display_id": "DEVO-1",
      "display_name": "4nm20is136",
      "email": "4nm20is13@mainit.in",
      "full_name": "Ritesh Shetty",
      "id": "don:identity:devr-us-1:devo/1m0em2UAvv:devu/1",
      "id_v1": "don:DEV-1m0em2UAvv:dev_user:DEVO-1",
      "state": "active",
      "thumbnail": "https://api.devrev.ai/internal/display-picture/Ritesh_Shetty.svg"
    },
    "modified_date": "2024-04-10T05:22:22.876Z",
    "slug": "helloworld3"
  }
}
```

6. Create Snap-in Draft:

- To create a snap-in draft instance, run the command:

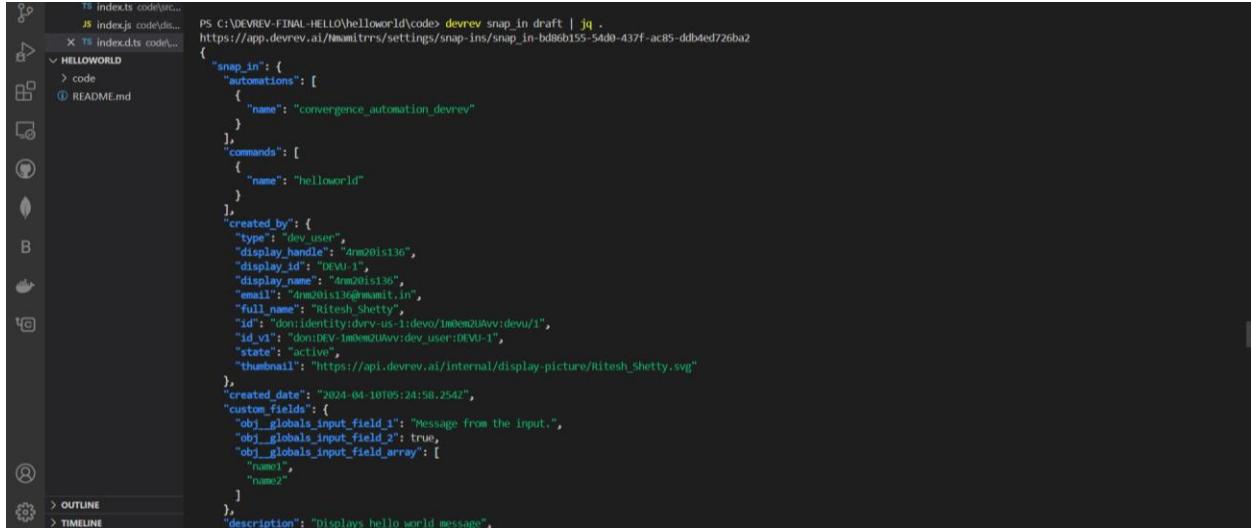
```

**devrev snap\_in draft --snap\_in\_version [id]**

```

- For example: **devrev snap_in draft | jq .**

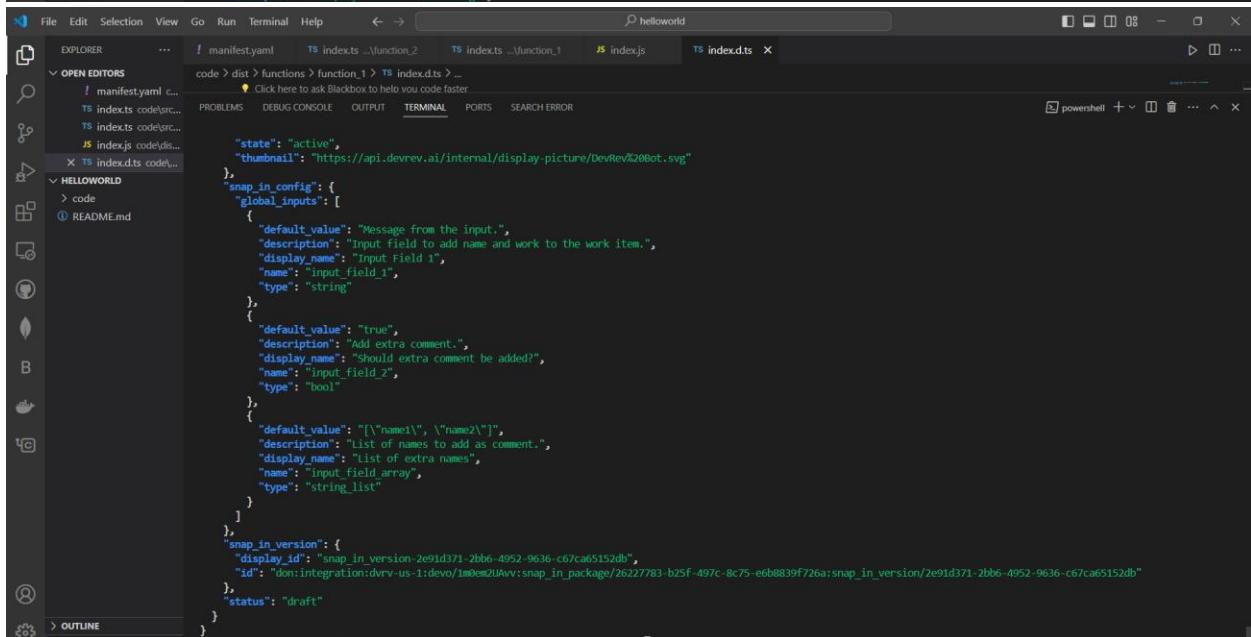
- The `--snap_in_version [id]` flag is optional. If not provided, it uses the snap-in version ID from the snap-in context.



```

PS C:\DEVREV-FINAL-HELLO\helloworld\code> devrev snap_in draft | jq .
{
  "snap_in": {
    "automations": [
      {
        "name": "convergence_automation_devrev"
      }
    ],
    "commands": [
      {
        "name": "helloworld"
      }
    ],
    "created_by": {
      "type": "dev_user",
      "display_handle": "4mz0is136",
      "display_id": "DEVU-1",
      "display_name": "4mz0is136",
      "email": "4mz0is136@munit.in",
      "full_name": "Ritesh Shetty",
      "id": "dom:identity:drv-us-1:devo/im0em2uuvv:devu/1",
      "id_v": "dom:DEV-im0em2uuvv:dev_user:DEVU-1",
      "state": "active",
      "thumbnail": "https://api.devrev.ai/internal/display-picture/Ritesh_Shetty.svg"
    },
    "created_date": "2024-04-10T05:24:58.254Z",
    "custom_fields": {
      "obj_globals_input_field_1": "Message from the input.",
      "obj_globals_input_field_2": true,
      "obj_globals_input_field_array": [
        "name1",
        "name2"
      ]
    },
    "description": "displays hello world message."
  }
}

```

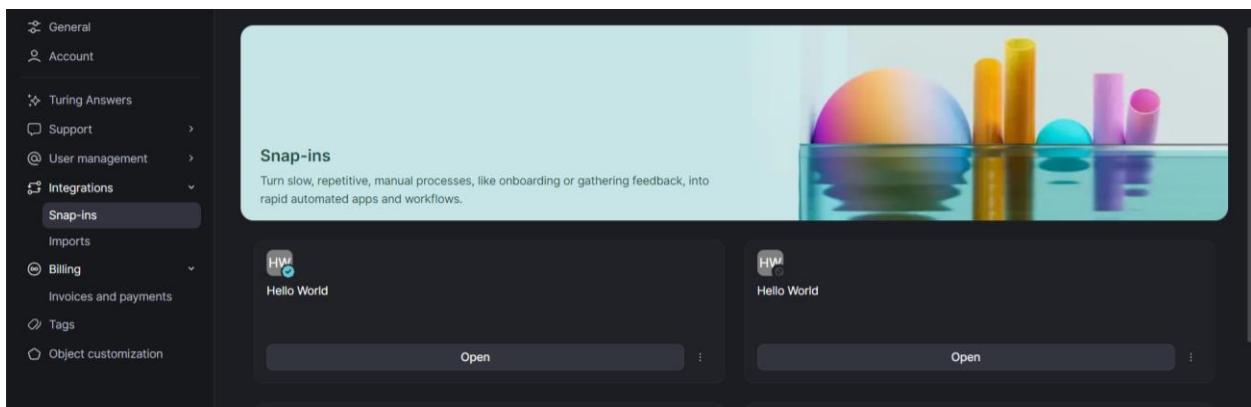


```

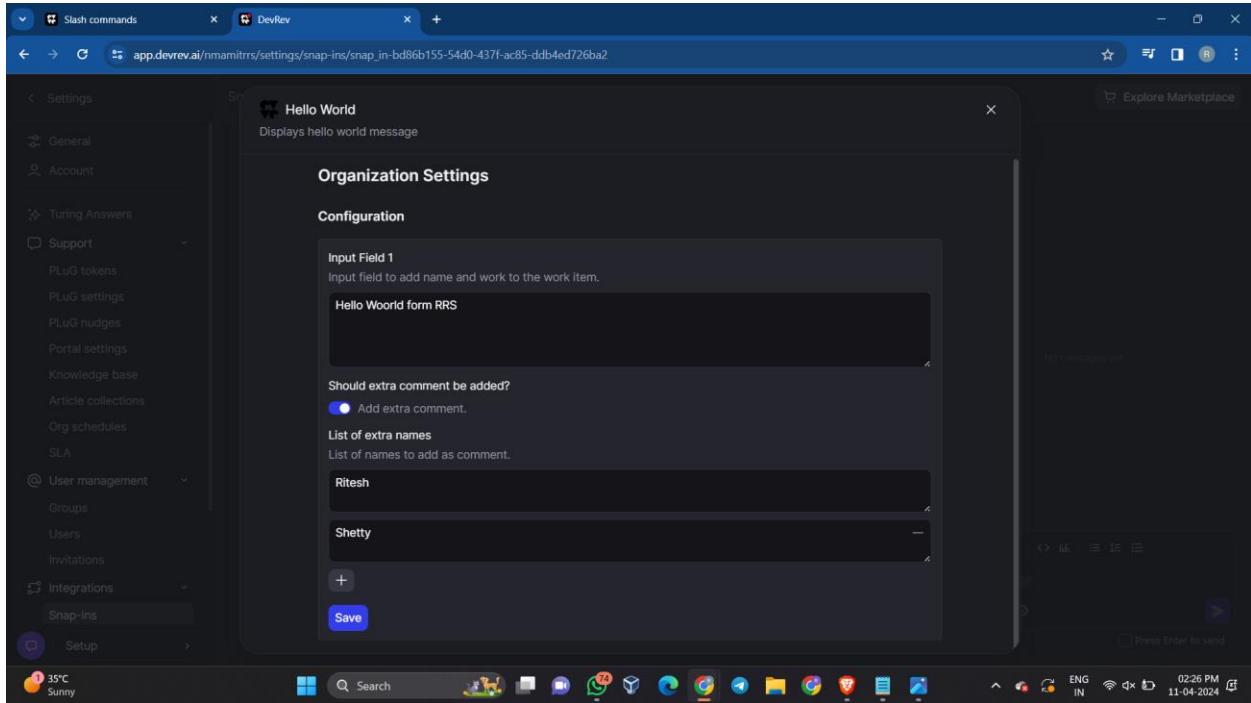
export interface HelloWorld {
  state: "active";
  thumbnail: string;
  global_inputs: {
    default_value: string;
    description: string;
    display_name: string;
    name: string;
    type: "string";
  };
  snap_in_config: {
    global_inputs: {
      default_value: string;
      description: string;
      display_name: string;
      name: string;
      type: "bool";
    };
    string_list: {
      default_value: string[];
      description: string;
      display_name: string;
      name: string;
      type: "string_list";
    };
  };
  snap_in_version: {
    display_id: string;
    id: string;
    status: "draft";
  };
}

```

Output For Hello-World Snap-in:



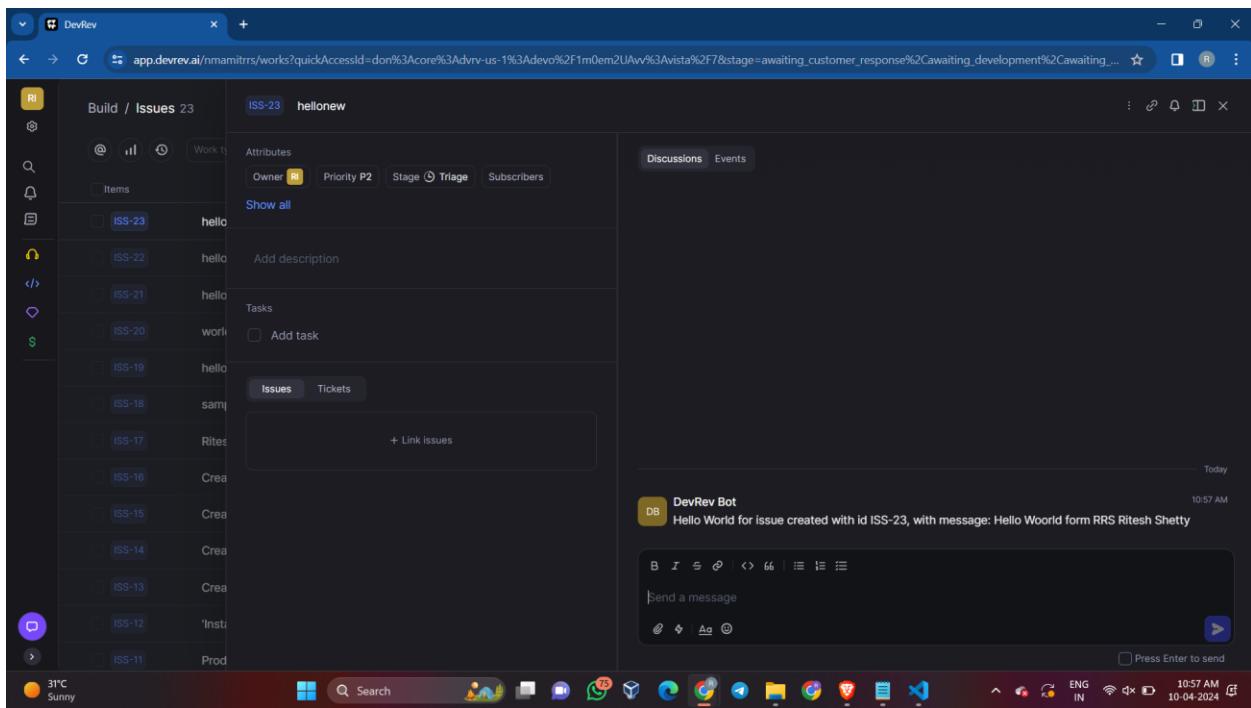
Configure and Deploy:



Go to Issues or Tickets or anything else as per the compatibility and create a new item:

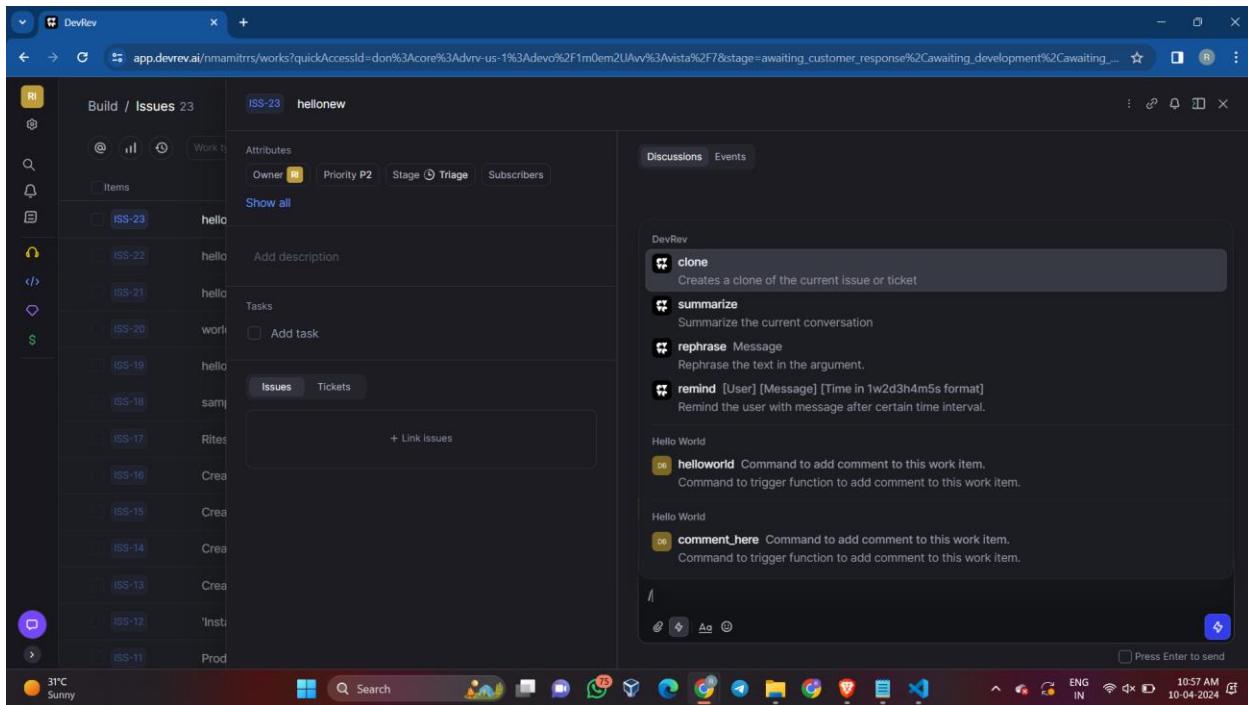
Hello world gets executed, this is a automated reply provided by the snap-in (Method-1)

For Issues:

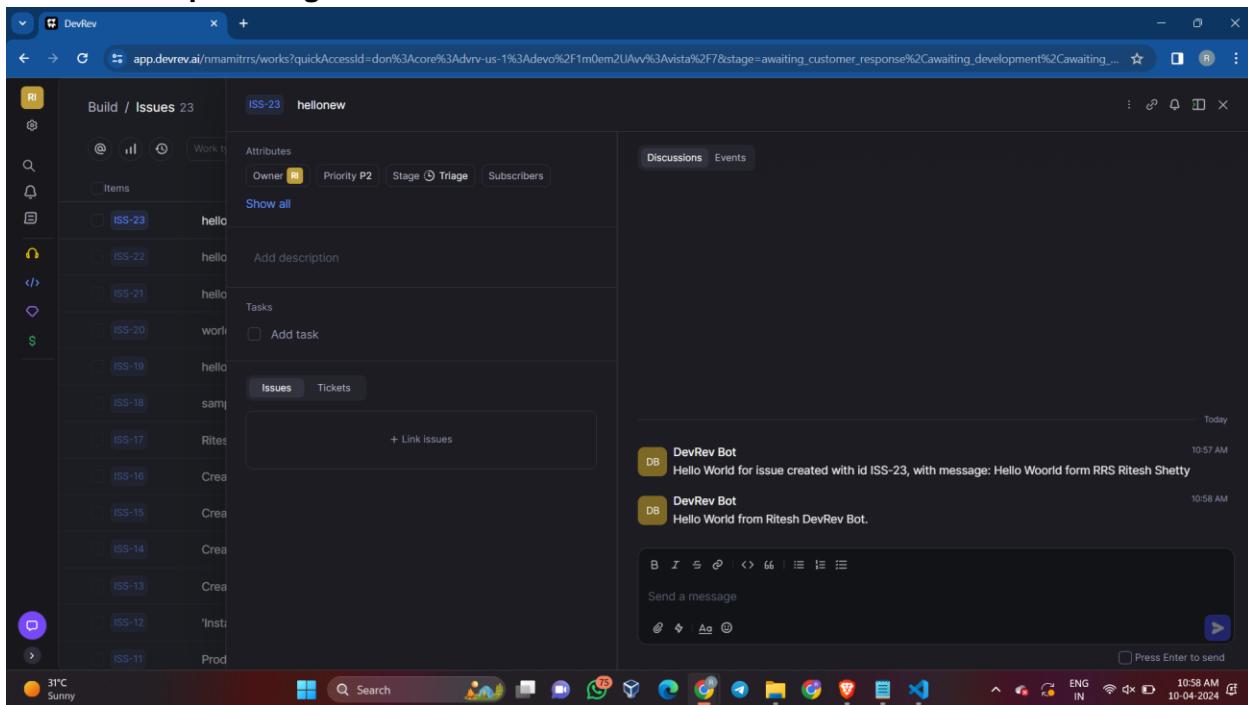


For method 2 , using slash command, click on the input box provided and type '/', list of all the slash command will be visible

Select helloworld slash command and execute



Method 2 output using slash command



For Tokens:

The image consists of two vertically stacked screenshots of a Microsoft Teams application window titled "DevRev".

Screenshot 1: Issues View

- Title:** Build / Issues 23
- Issue Details:** ISS-23, hellonew
- Attributes:** Owner (Ritesh Shetty), Priority P2, Stage Triage
- Tasks:** Add task
- Message Panel:** DevRev Bot says: "Hello World for issue created with id ISS-23, with message: Hello Woorld form RRS Ritesh Shetty" at 10:57 AM.

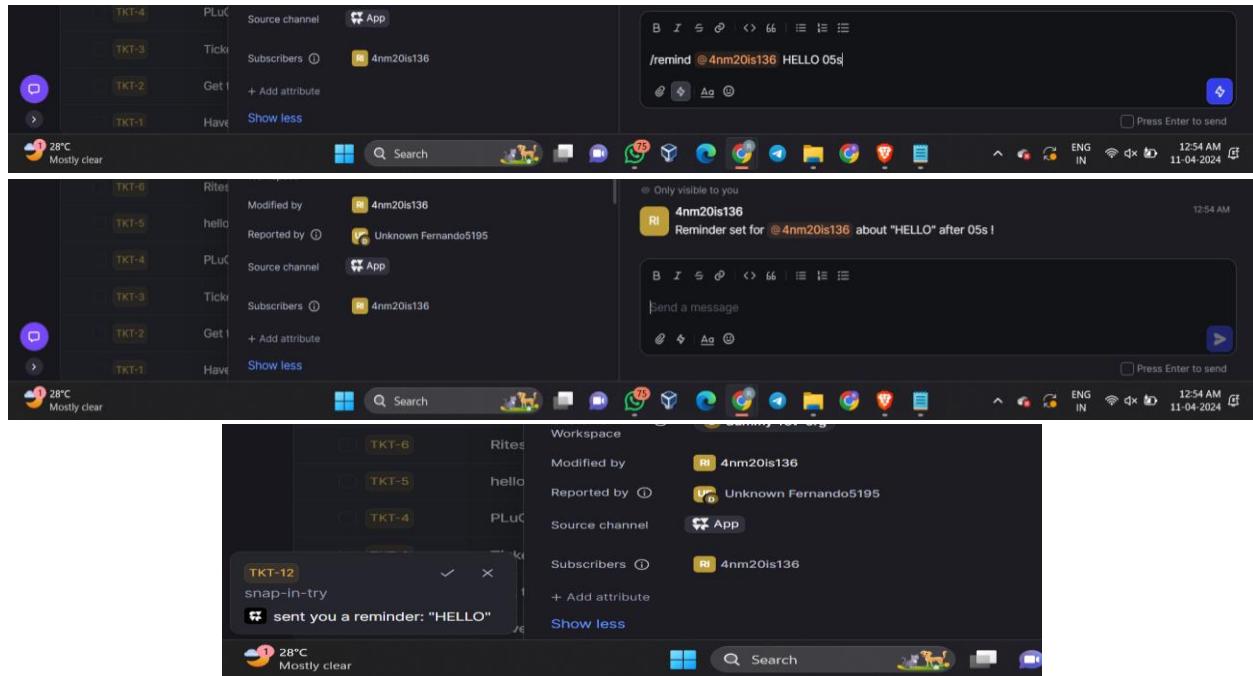
Screenshot 2: Tickets View

- Title:** Support / Tickets 5
- Ticket Details:** TKT-5, helloworld
- Attributes:** Owner (Ritesh Shetty), Severity Medium, Stage Queued
- Tasks:** Add task
- Message Panel:**
 - DevRev Bot says: "Hello World for issue created with id TKT-5, with message: Hello Woorld form RRS Ritesh Shetty" at 10:56 AM.
 - DevRev Bot says: "Hello World from Ritesh DevRev Bot." at 10:57 AM.

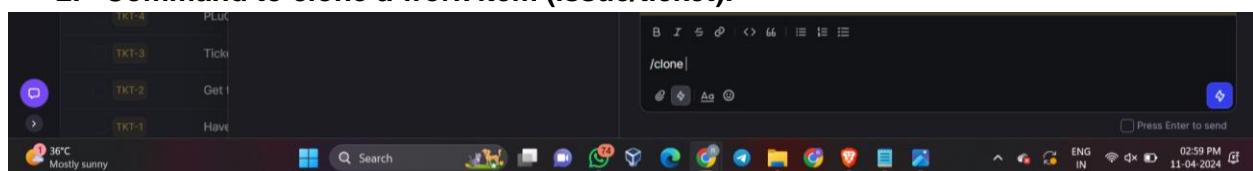
Output for Snap-in:

- Configure the snap-in in setting
- Deploy the Snap-in
- Come back to the Main page
- Either create a new Issue or Ticket
- Within it execute the slash command or function defined

1. Schedule a reminder for someone by tagging them on a work item after MM HH DD time. Where MM is the minute, HH is the hour and DD is the time.



2. Command to clone a work item (issue/ticket).



Ritesh Shetty - 4NM20IS136

The screenshot shows a ticket creation interface for a support system. The ticket being created is TKT-12, with the subject "snap-in-try". The form fields include:

- Owner: 4nm20is136
- Group: Platform Users
- Severity: Medium
- Stage: Queued
- Part: Default Product 1
- Created by: 4nm20is136
- Created date: 12:52 AM, Today
- Modified date: 12:53 AM, Today
- Tags: owner_assigned
- Target Close Date: Add Target Close Date
- Customer Workspace: dummy-rev-org
- Modified by: 4nm20is136
- Reported by: Unknown Fernando5195
- Source channel: App
- Subscribers: 4nm20is136

In the internal discussions tab, there are three messages:

- DevRev Bot: Hello World for issue created with id TKT-12, with message: Hello Woorld form RRS Ritesh Shetty (12:52 AM)
- DevRev Bot: Hello World from Ritesh DevRev Bot. (12:53 AM)
- 4nm20is136: This ticket was cloned and # TKT-13 was created. (12:53 AM)

A modal window at the bottom right shows the cloned ticket TKT-13 with the subject "[CLONE] snap-in-try" and status "Queued Medium".

- 3. Auto-Reply to a message if it is outside office hours. The start and end time for office hours can be specified in the input. The auto-reply message can also be taken as an input.**

Configure the Snap-in

Auto customer reply

Replies to new conversations with an automated response.

Business start hours *
Start time of business days in 24 hour format (HHMM).
900

Business end hours *
End time of business days in 24 hour format (HHMM).
1800

UTC offset *
Hours offset from UTC in 24 hour format (HHMM) (GMT+5:30 → 530).
530

Business days *
Select your organization's business days.
Monday, Tuesday, Friday, Saturday, Sunday

Enable a custom button
 Enable this option to attach the custom message button with the automated reply

Text displayed on custom button *
Enter a text to appear on the custom button.
ML Project

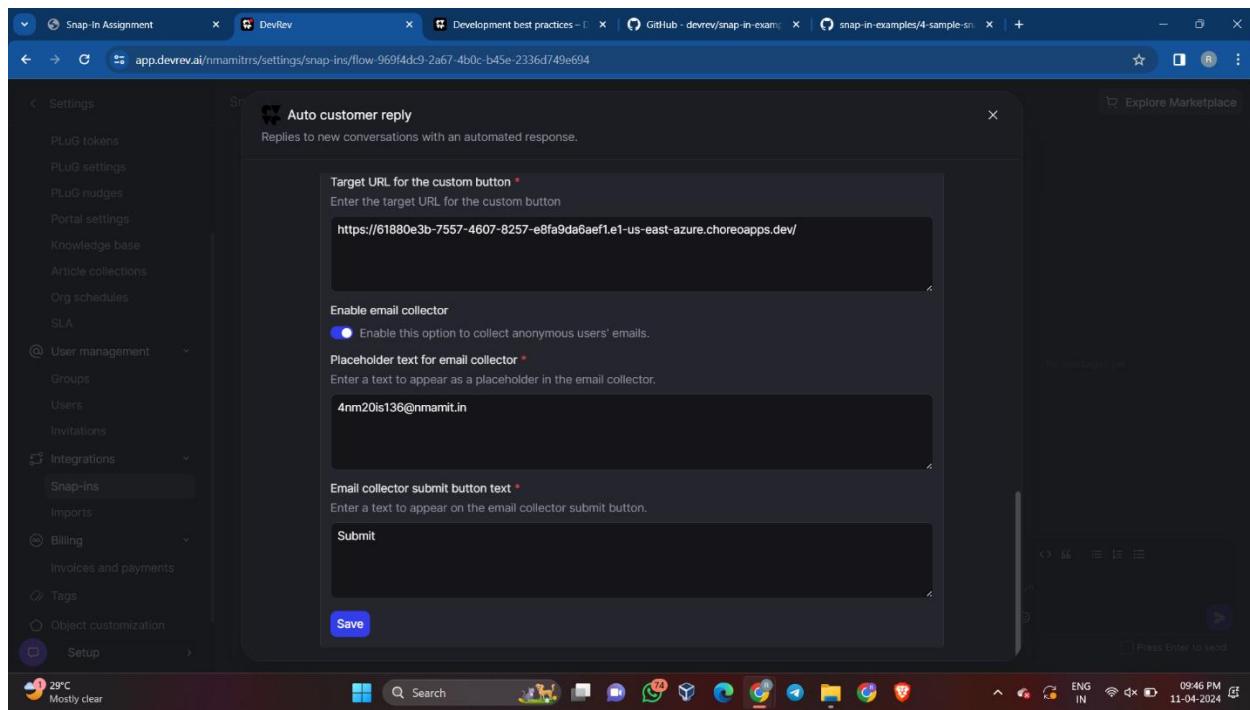
Text displayed on custom button *
Enter a text to appear on the custom button.
ML Project

Target URL for the custom button *
Enter the target URL for the custom button
https://61880e3b-7557-4607-8257-e0fa9da6aeff1ef1-us-east-azure.choreoapps.dev/

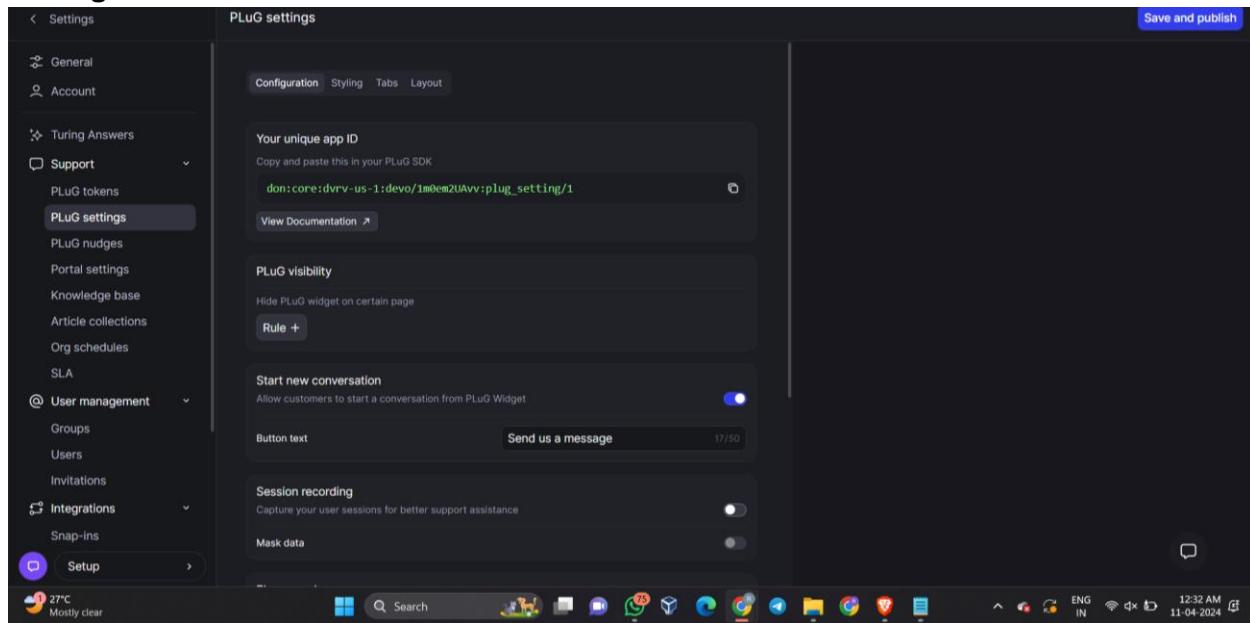
Enable email collector
 Enable this option to collect anonymous users' emails.

Placeholder text for email collector *
Enter a text to appear as a placeholder in the email collector.
4nm20is136@nmamit.in

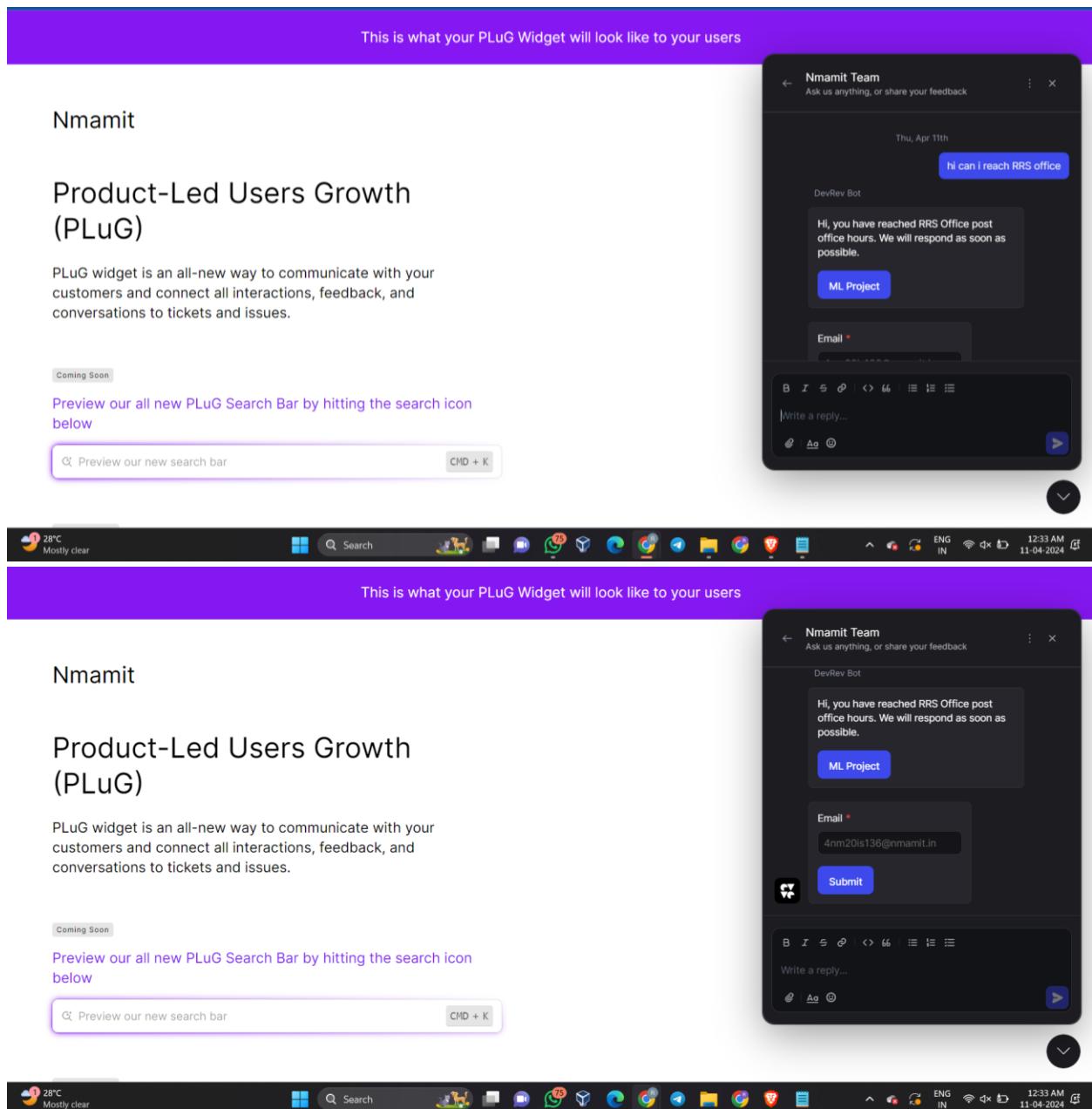
Email collector submit button text *
Enter a text to appear on the email collector submit button.



Setting a PLuG in DevRev



With PLuG:



Without using PLoG:
Create a New Customer Conversation under Support > Inbox on the Main Page

The image consists of two vertically stacked screenshots of a customer support inbox interface, likely from a software like Zendesk or similar. Both screenshots show a dark-themed interface with a sidebar on the left containing various icons and a weather widget at the bottom.

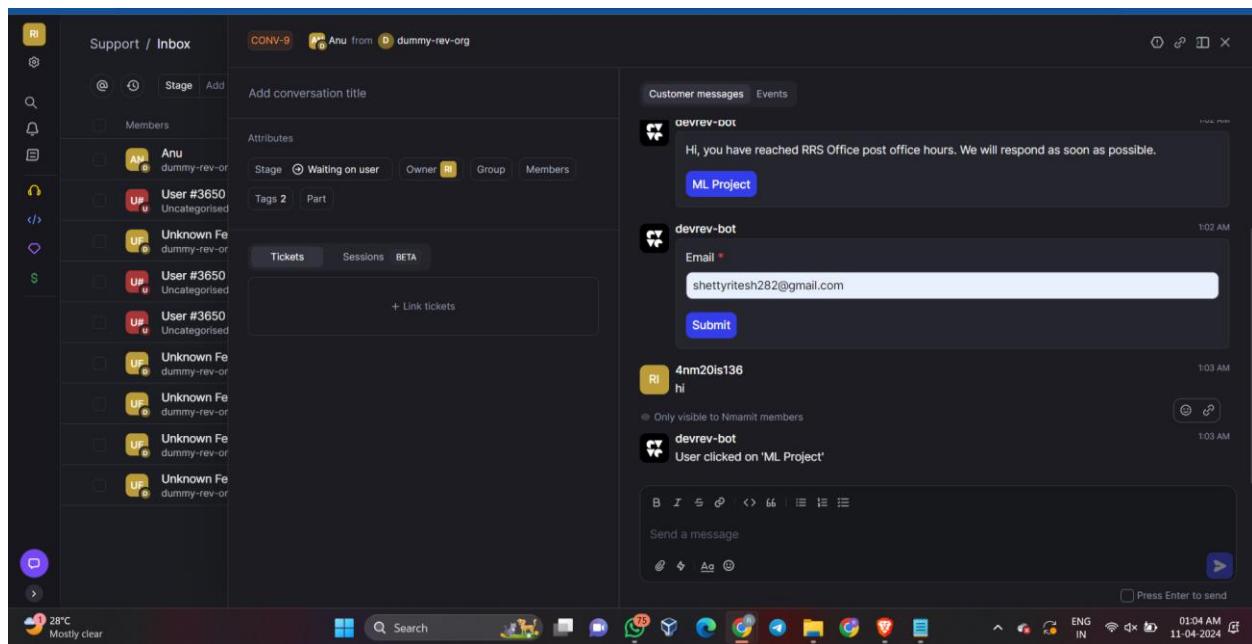
Screenshot 1 (Top):

- Title:** CONV-7 Unknown Fernando5195 from dummy-rev-org
- Attributes:** Stage (New), Owner (Unassigned), Group, Members
- Message 1:** Unknown Fernando5195 (12:35 AM) I am having trouble adding new users to my account
- Message 2:** devrev-bot (12:35 AM) Hi, you have reached RRS Office post office hours. We will respond as soon as possible.
- Message 3:** devrev-bot (12:35 AM) Email: 4nm20is136@nmamit.in
- Action:** Submit
- Text Input:** Send a message

Screenshot 2 (Bottom):

- Title:** CONV-8 User #3650 from Nmamit / Uncategorized Contacts
- Attributes:** Stage (New), Owner (Unassigned), Group, Members
- Message 1:** User #3650 (12:46 AM) hi
- Message 2:** devrev-bot (12:47 AM) Hi, you have reached RRS Office post office hours. We will respond as soon as possible.
- Message 3:** devrev-bot (12:47 AM) Email: 4nm20is136@nmamit.in
- Action:** Submit
- Text Input:** Send a message

Both screenshots include a standard Windows taskbar at the bottom with various application icons and system status indicators.



Thank You