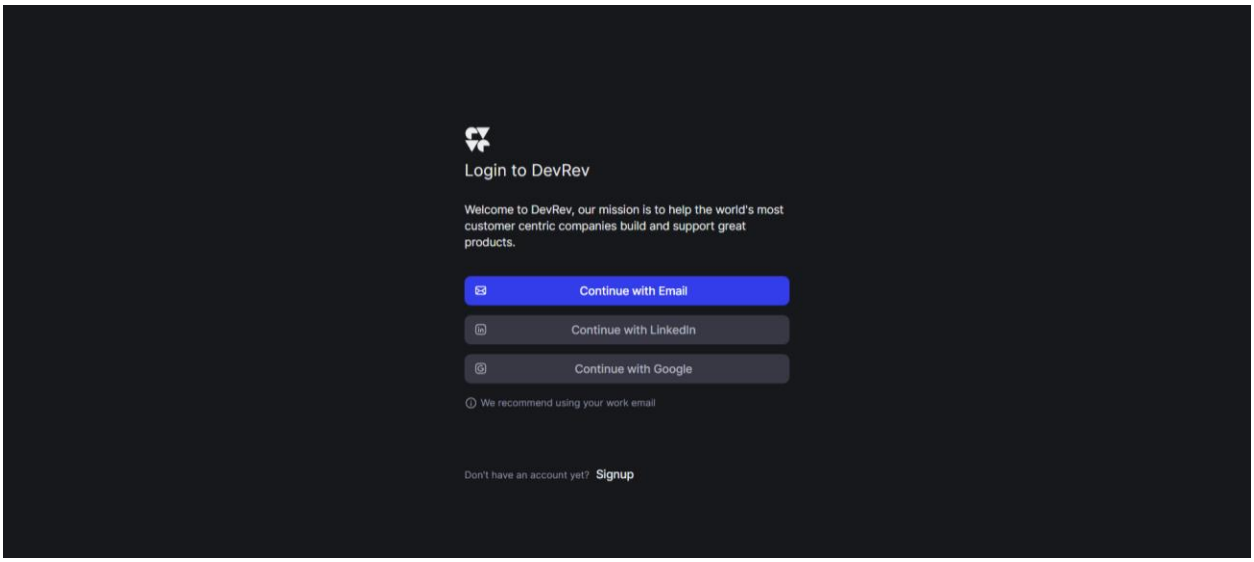
Assignment: Snap-in Development Challenge Submission

Step 1 : Utilizing the DevRev AP

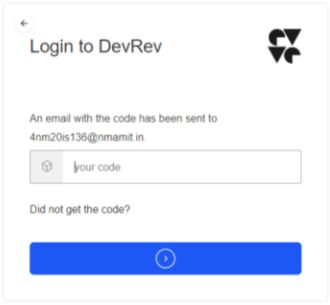
First task is to utilize the DevRev API to create a work item. Python was used as the programming language, which is compatible with the DevRev API. The Code snippet demonstrates the creation of a work item using the DevRev API.

Initial Steps to be executed:

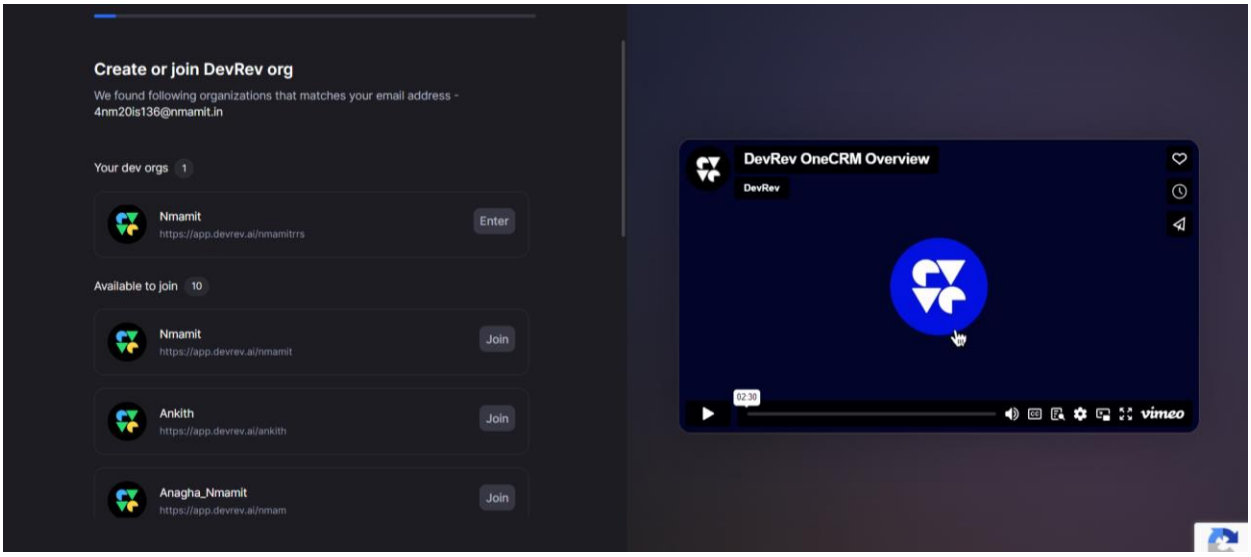
1. Creation of DevRev account:
    - Click on the sign in option
    - Continue with Gmail



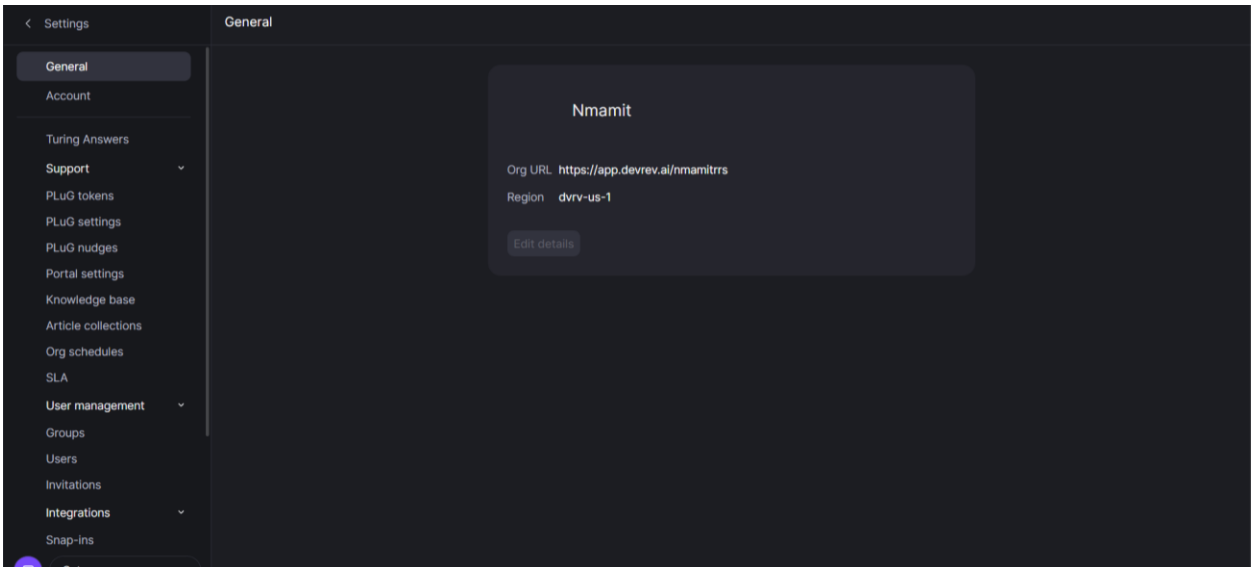
- Enter Work email: here college email id and enter the verification code sent on your email id entered.



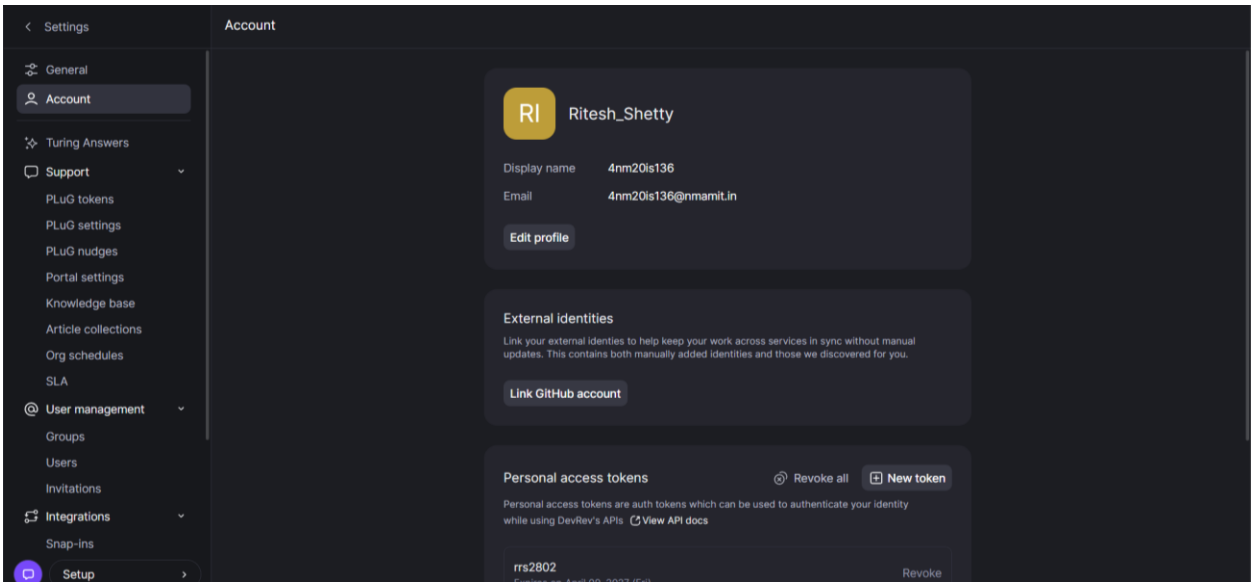
- Create a workspace or if already created, enter your workspace



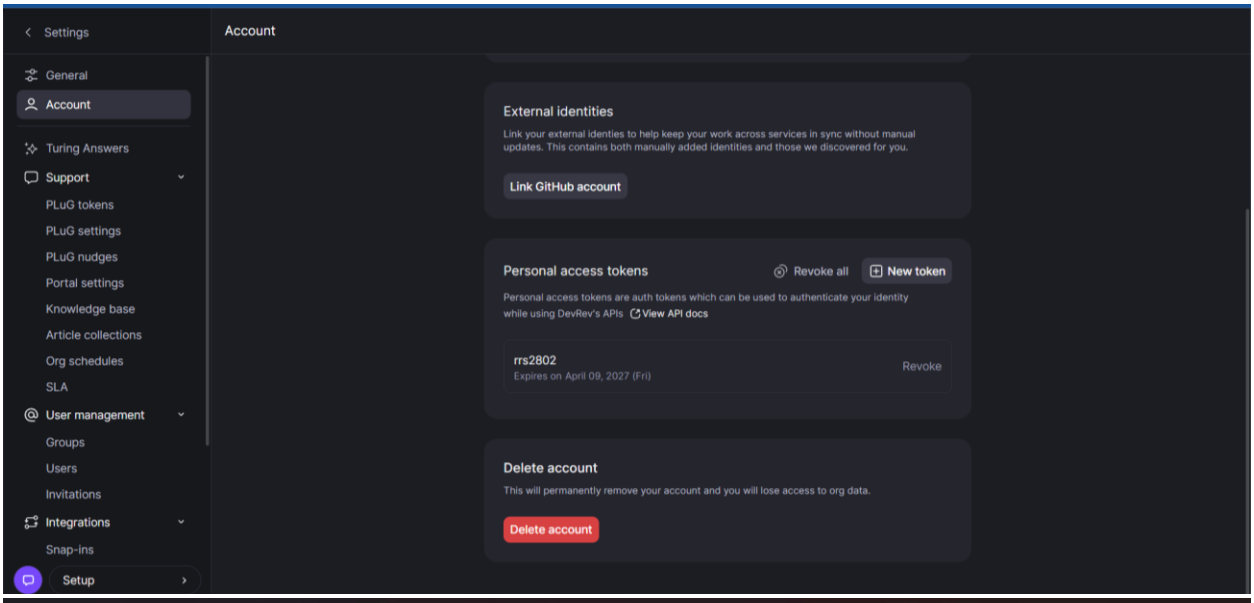
2. Generating your PAT(Personal access token):
    - Go to the relevant dev org in the DevRev app and navigate to Settings > Account > Personal Access Token to create a PAT.



    - 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.

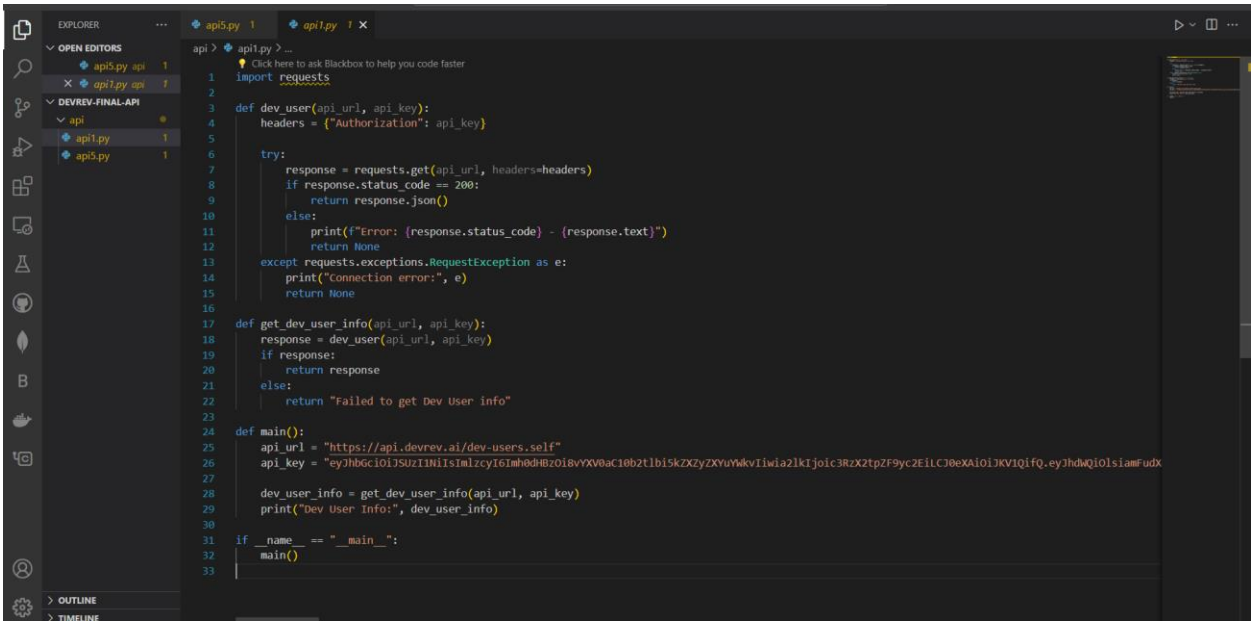
- Tickets are created by customers or consumers and are part of the Support app.
- Issues are created by builders or maintainers and are part of the Build app.
- Enhancements serve as parent items for multiple issues, leading to desired product changes.
- Tasks are used to break down larger work into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

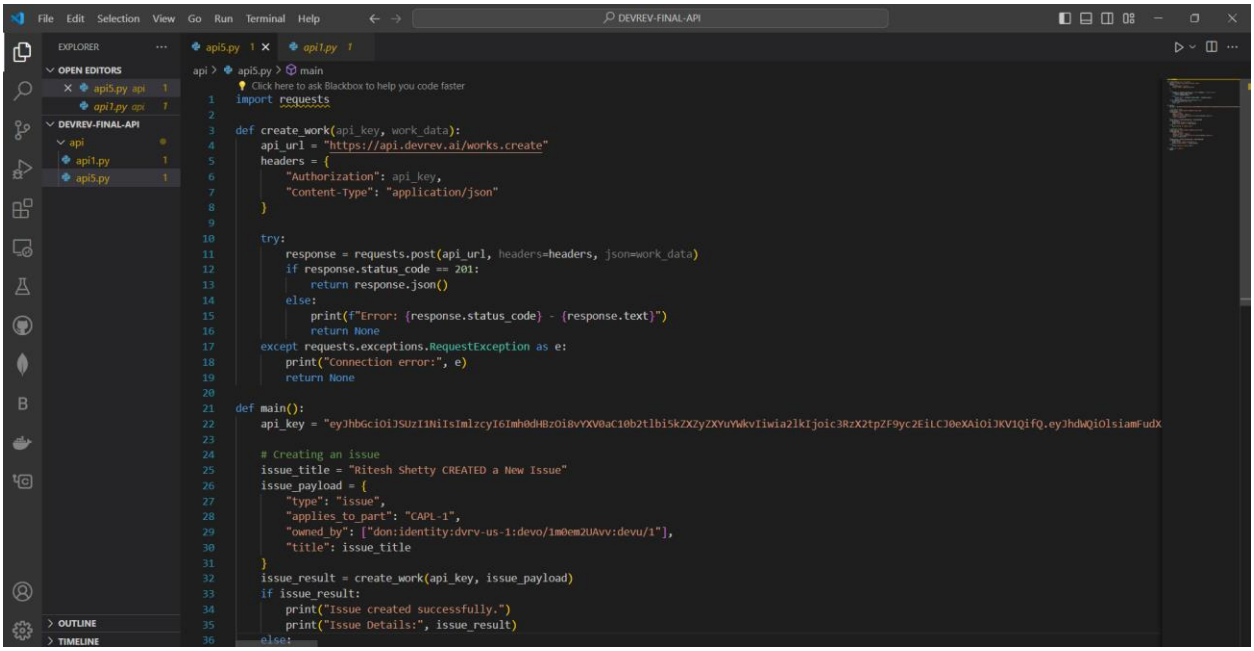
- The provided code makes use of the "requests" library to perform an HTTP GET request to a specified API endpoint which is to retrieve the user details using the unique API key.
- The "dev_user" function is defined to make an HTTP GET request to the specified API endpoint using the provided API key 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.



Output:

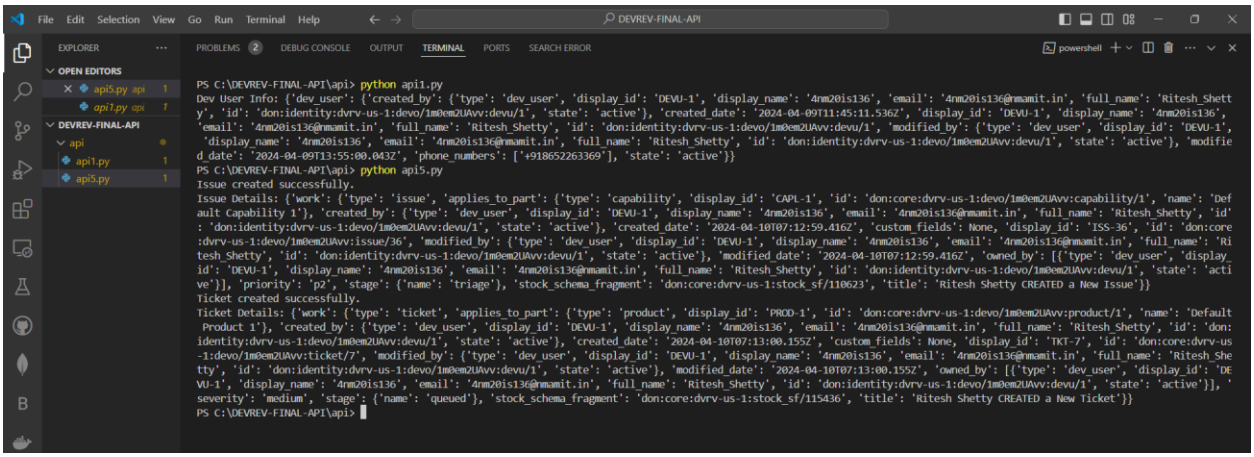
Explanation for api5.py:

- 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.

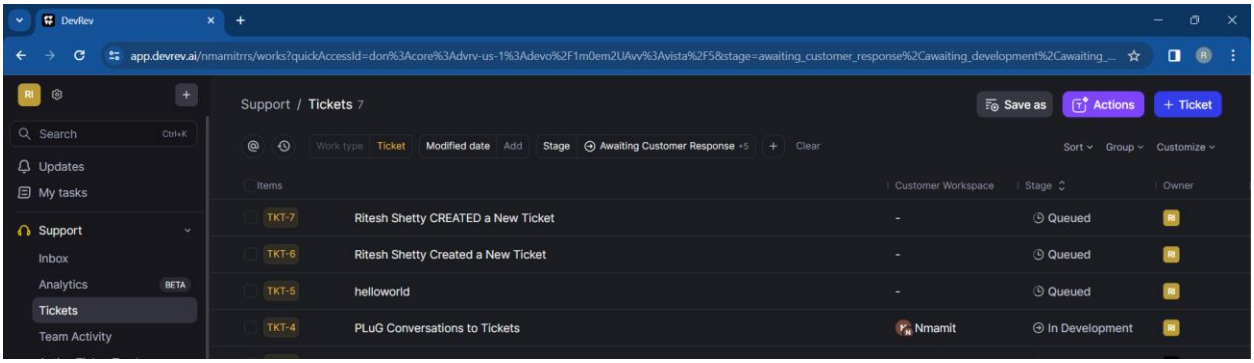


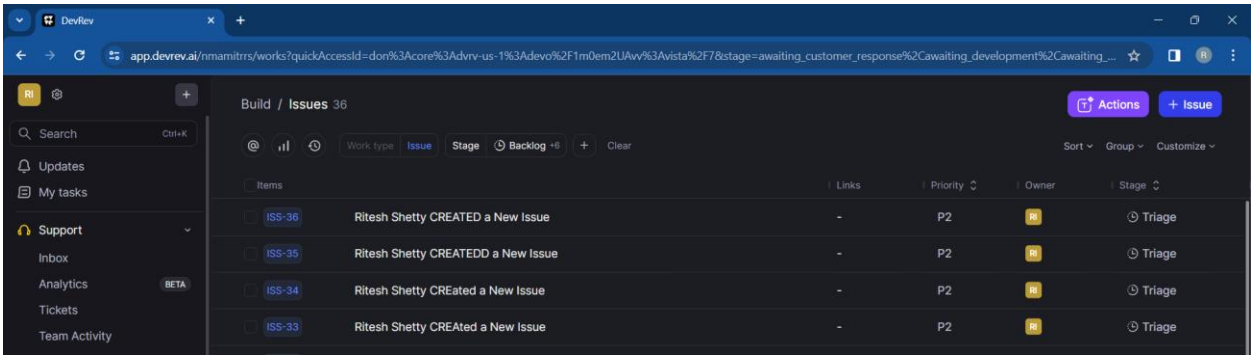
```python
import requests

def create_work(api_key, work_data):
    api_url = "https://api.devrev.ai/works.create"
    headers = {
        "Authorization": api_key,
        "Content-Type": "application/json"
    }

    try:
        response = requests.post(api_url, headers=headers, json=work_data)
        if response.status_code == 201:
            return response.json()
        else:
            print(f"Error: {response.status_code} - {response.text}")
            return None
    except requests.exceptions.RequestException as e:
        print("Connection error:", e)
        return None

def main():
    api_key = "eyJhbGciOiJSUzI1NiIsImlzcyI6Imh0dHBzOi8vYXV0aC10b2tlbi5kZXZyZXVuYWkvIiwia2lkIjoic3RzX2tpZF9yc2EiLCJ0eXAiOiJKV1QifQ.eyJhdWQiOlsiamFudX

    # Creating an issue
    issue_title = "Ritesh Shetty CREATED a New Issue"
    issue_payload = {
        "type": "issue",
        "applies_to_part": "CAPL-1",
        "owned_by": ["don:identity:dvrv-us-1:devo/1m0em2UAvv:devu/1"],
        "title": issue_title
    }
    issue_result = create_work(api_key, issue_payload)
    if issue_result:
        print("Issue created successfully.")
        print("Issue Details:", issue_result)
    else:
        print("Failed to create issue")

    # Creating a ticket
    ticket_title = "Ritesh Shetty CREATED a New Ticket"
    ticket_payload = {
        "type": "ticket",
        "applies_to_part": "PROD-1",
        "owned_by": ["don:identity:dvrv-us-1:devo/1m0em2UAvv:devu/1"],
        "title": ticket_title
    }
    ticket_result = create_work(api_key, ticket_payload)
    if ticket_result:
        print("Ticket created successfully.")
        print("Ticket Details:", ticket_result)
    else:
        print("Failed to create ticket")

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

Output:



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





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.

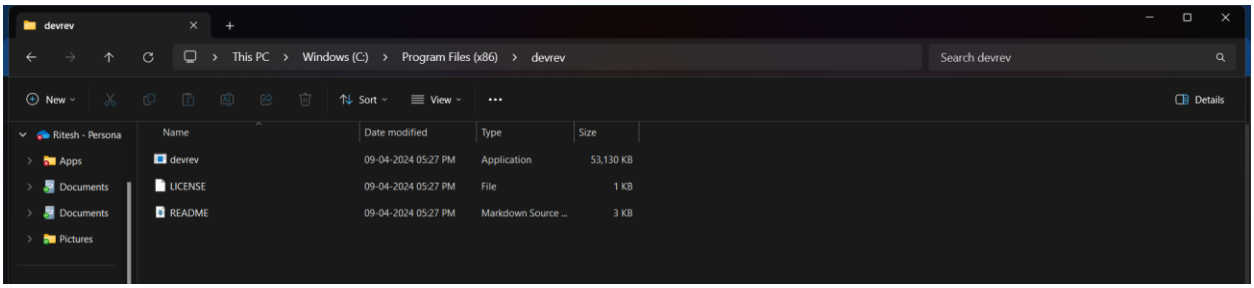
Brief explanation on what are Snap-in:

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.

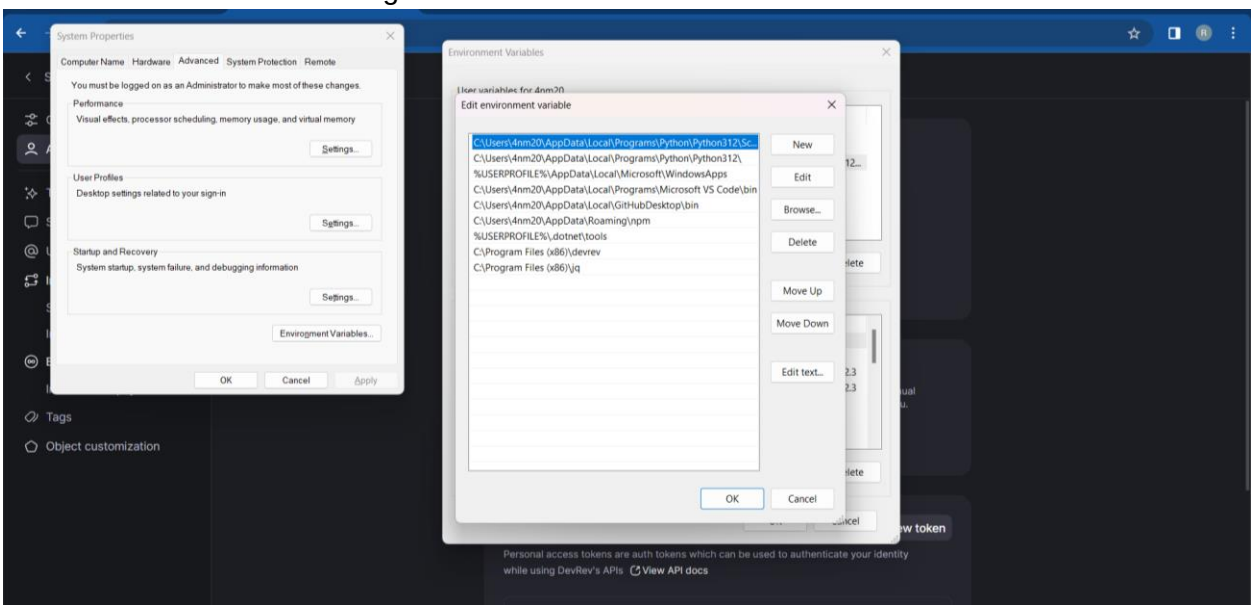
Before creation of Snap-in a few prerequisites needs to be installed into the user system:

1. Install DevRev CLI:

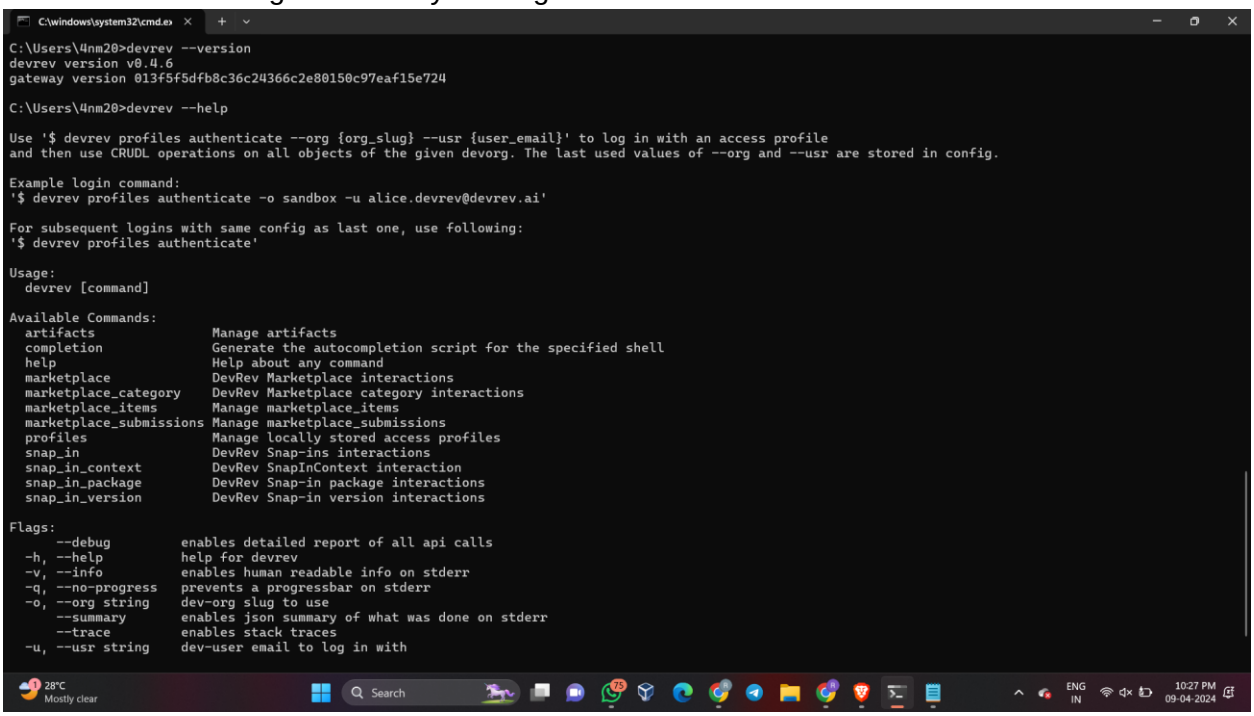
1. Download the Windows executable for the supported architecture (Windows amd64).
2. Unzip the downloaded file.

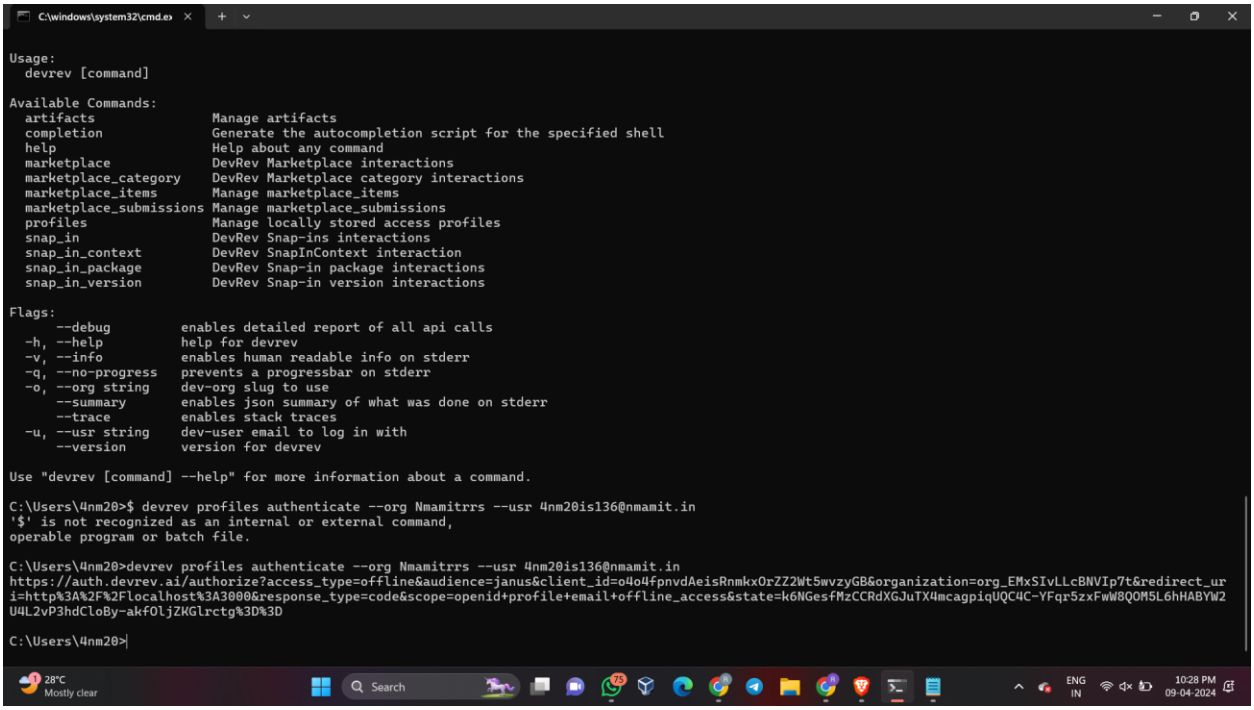


3. Add the path to the environment variable:
   - Open System Properties > This PC > Properties.
   - In the System window, click Advanced system settings > Environment Variables > Path > Edit.
   - In the Edit environment variable window, click New.
   - Enter the path to the directory you want to add,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

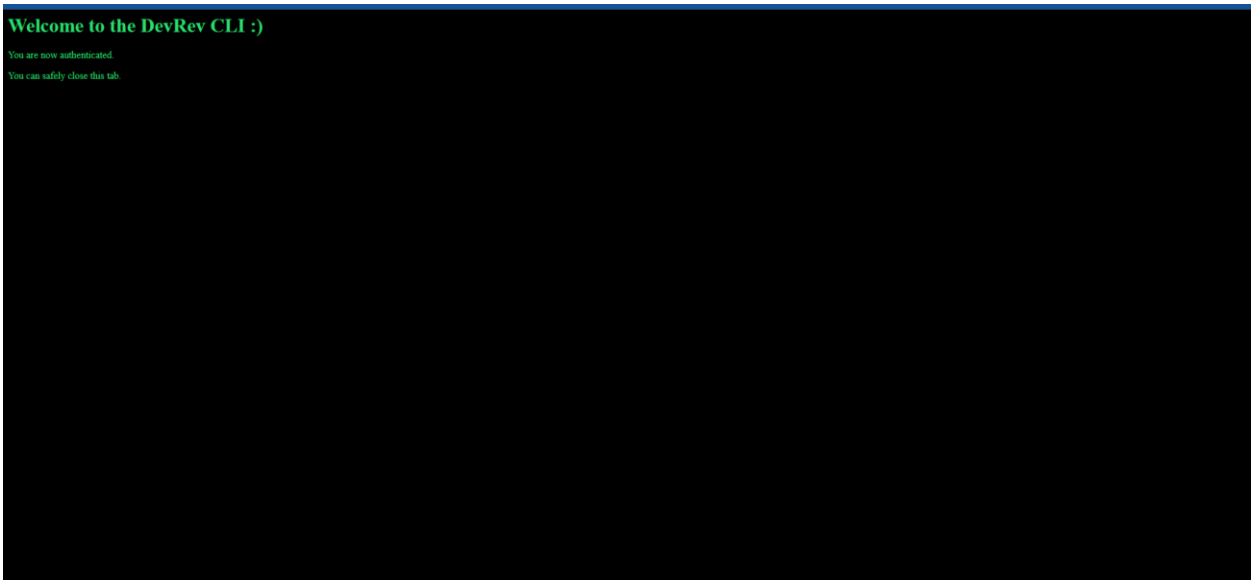


4. Close all remaining windows by clicking OK.



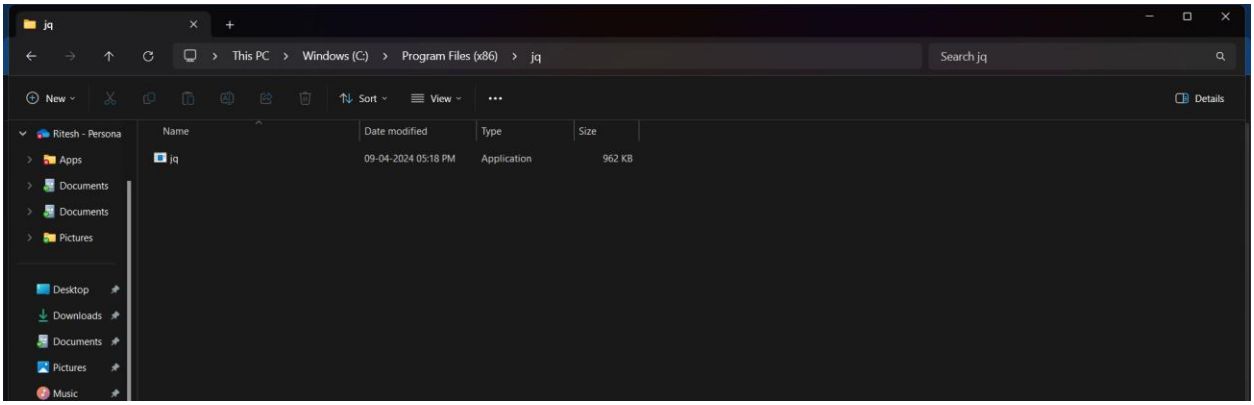


Output:

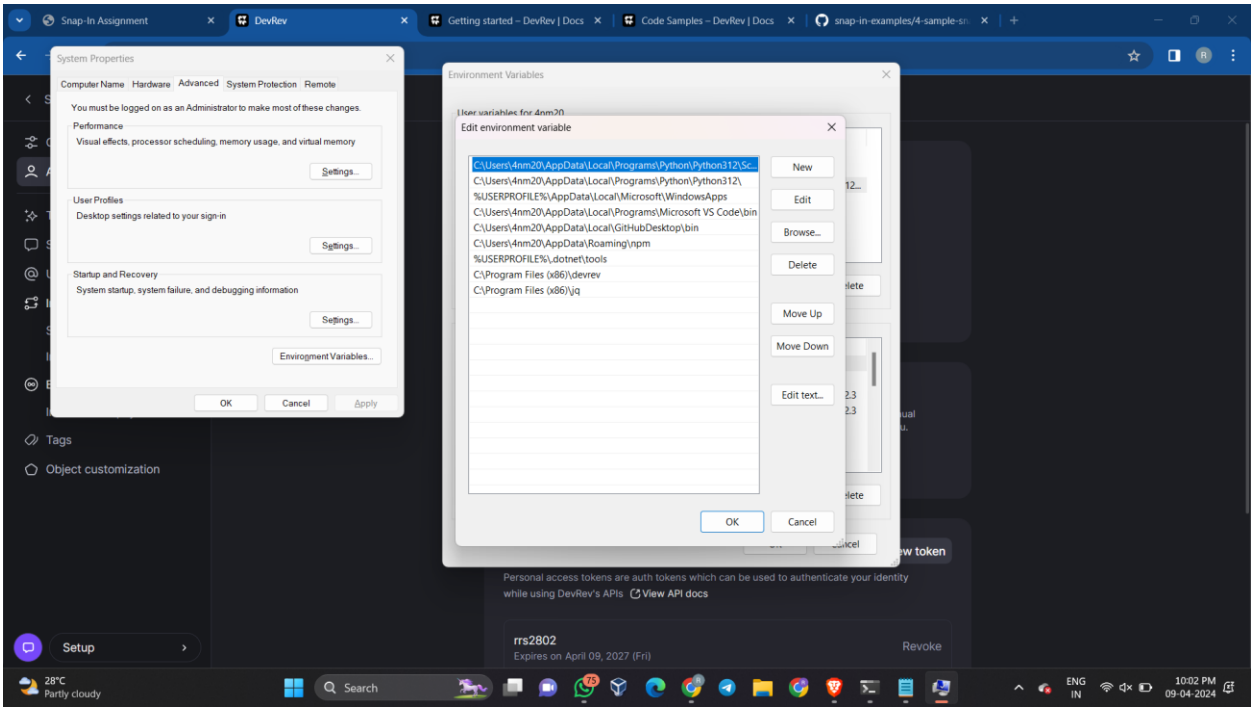


2. Install jq:

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.

3. Install DevRev SDK:

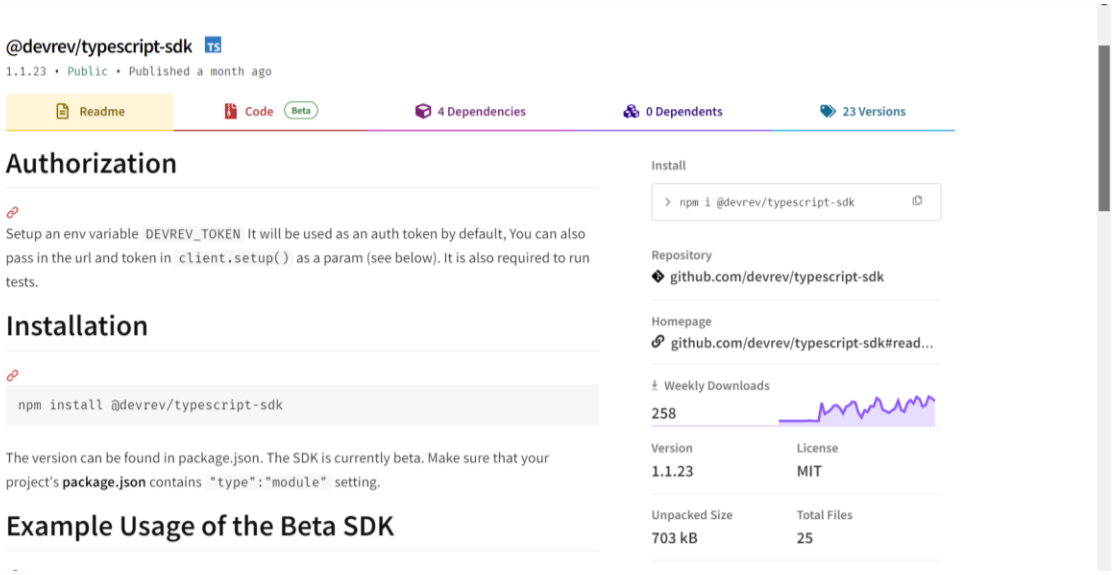
1. Set up an environment variable named DEVREV_TOKEN. This token will be used as an authentication token by default. You can also pass in the URL and token in client.setup() as a parameter. This token is also required to run tests.

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.



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

1. Open your terminal or command prompt.

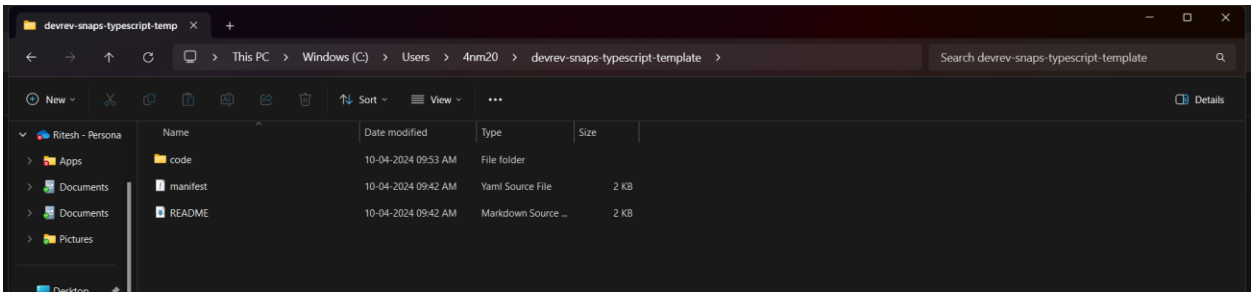
2. Run the following command to initialize a snap-in template:

```
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
```

```
│   │   ├── functions
│   │   │   └── on_work_creation
│   │   │       ├── index.test.ts
│   │   │       └── index.ts
│   │   ├── index.ts
│   │   ├── main.ts
│   │   └── test-runner
│   │       ├── example.test.ts
│   │       └── test-runner.ts
│   ├── tsconfig.eslint.json
│   └── tsconfig.json
├── manifest.yaml
└── README.md
```



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.

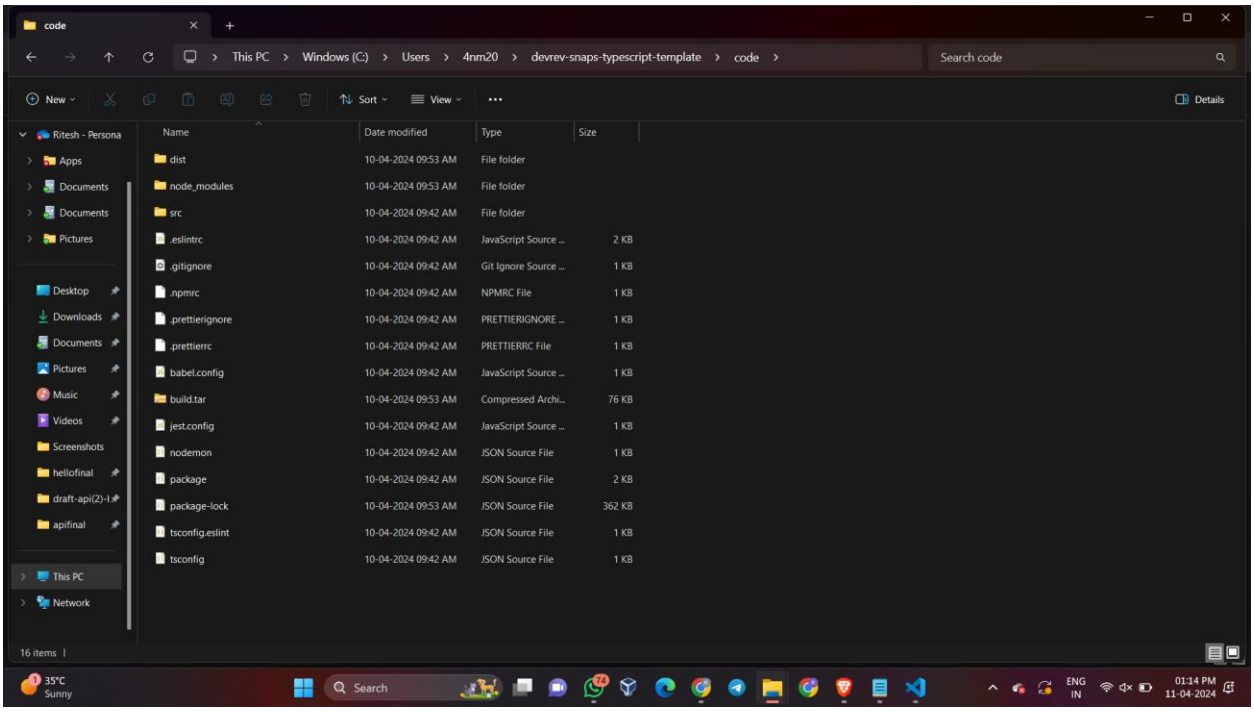
2. Upon successful creation, the CLI automatically stores the package ID in its context corresponding to the slug.

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:

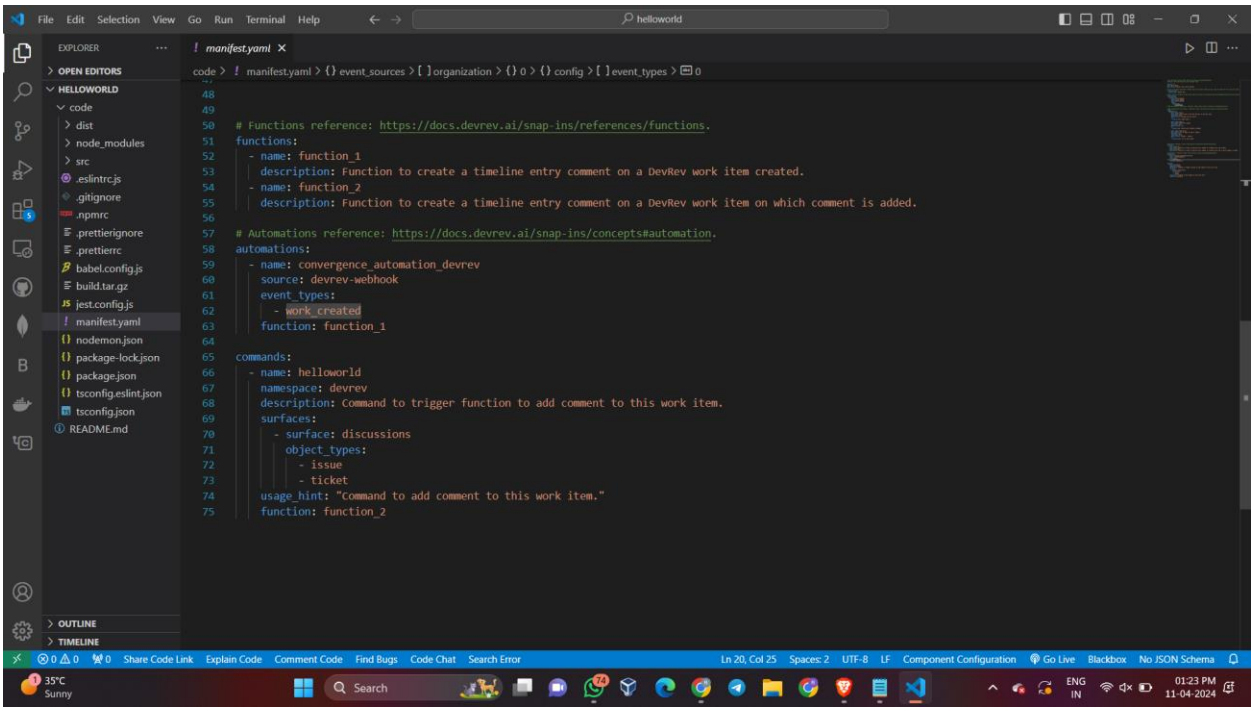
- The "functions" section lists the functions available in the snap-in, along with their descriptions.
- In this case, it includes "function_1" and "function_2" with their respective descriptions.

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.



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.

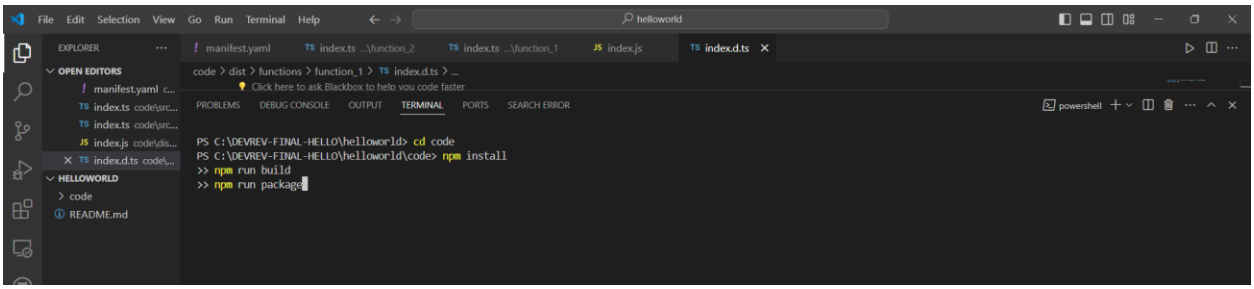
9. Logging and Exporting:

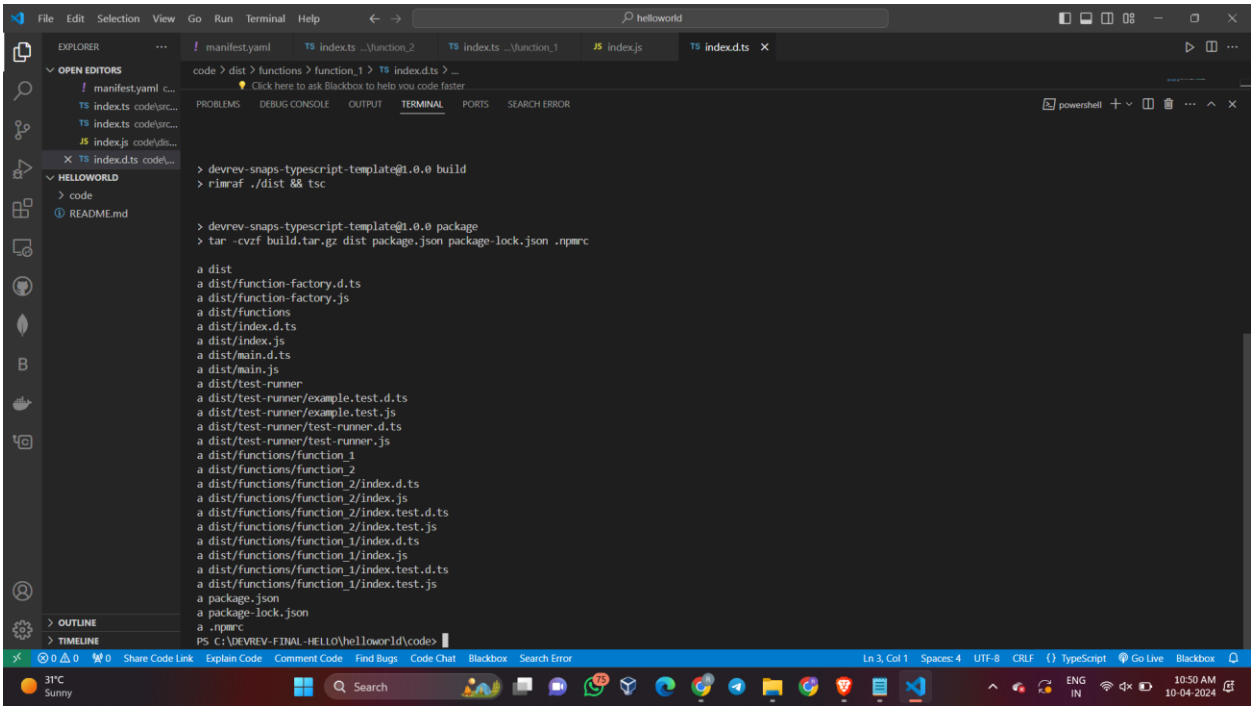
- Relevant information, such as event details and response data, is logged or processed as needed.
- The "run" function is exported as the default export, making it available for use in other modules.

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.



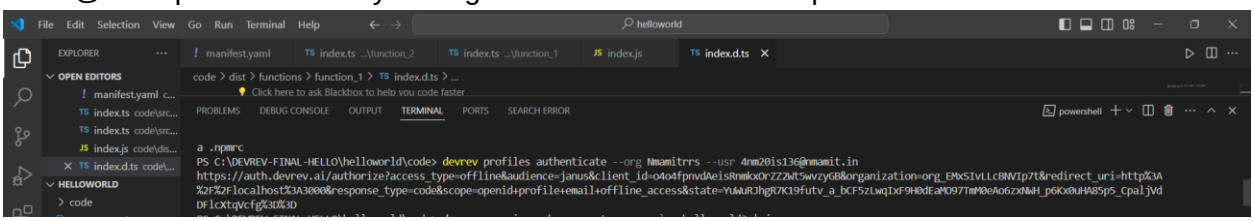


2. Authentication:

- To authenticate, run the command:

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

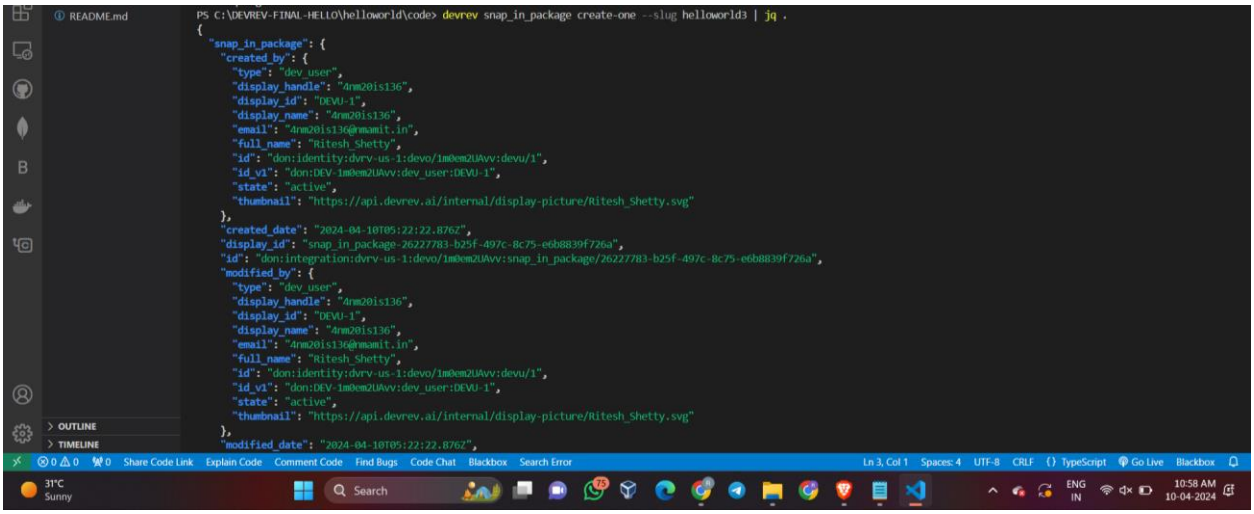
- Replace `<DevOrg-slug-name>` with the unique slug name of your DevOrg, and `<your-email@example.com>` with your registered user email for the profile.



3. Snap-in Package:

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

```
devrev snap_in_package create-one --slug <slug name>| jq .
```

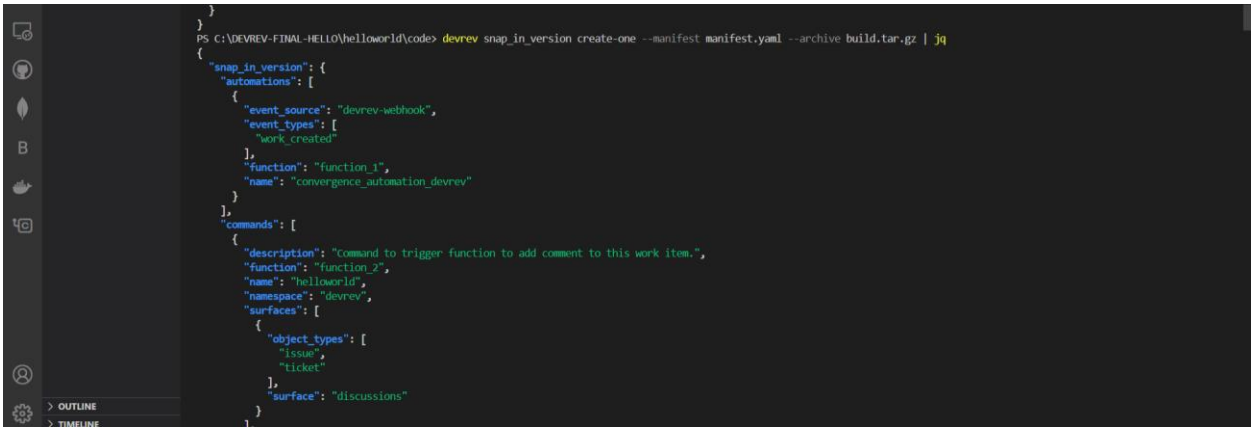


4. Snap-in Version:

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

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

- For example: `devrev snap_in_version create-one --path ./code`
- Once the snap_in_version is created, its ID is stored in the snap-in context.

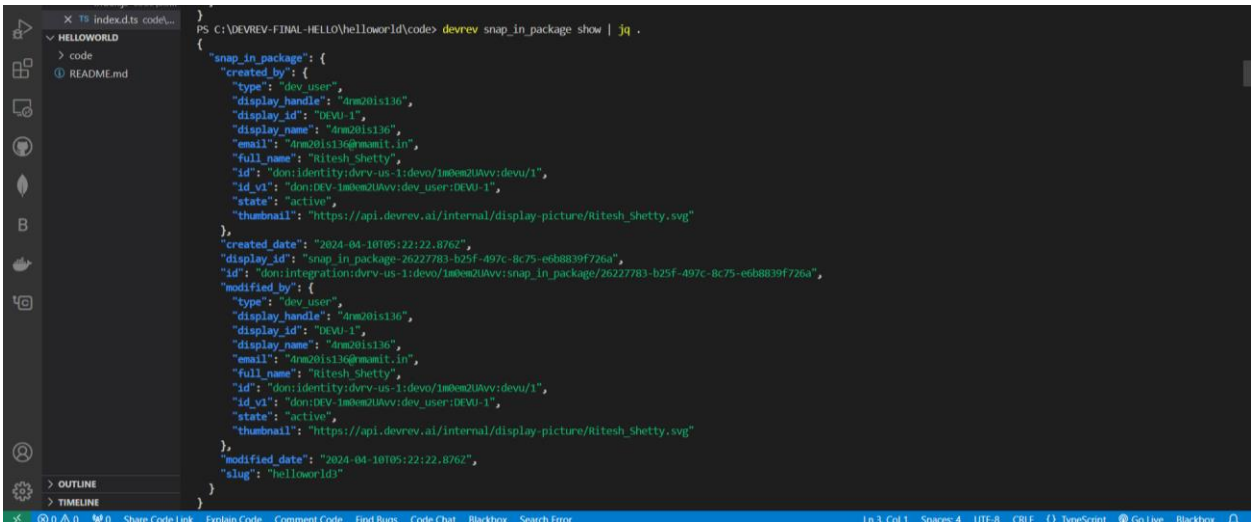


5. Show Snap-in Details:

- 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.

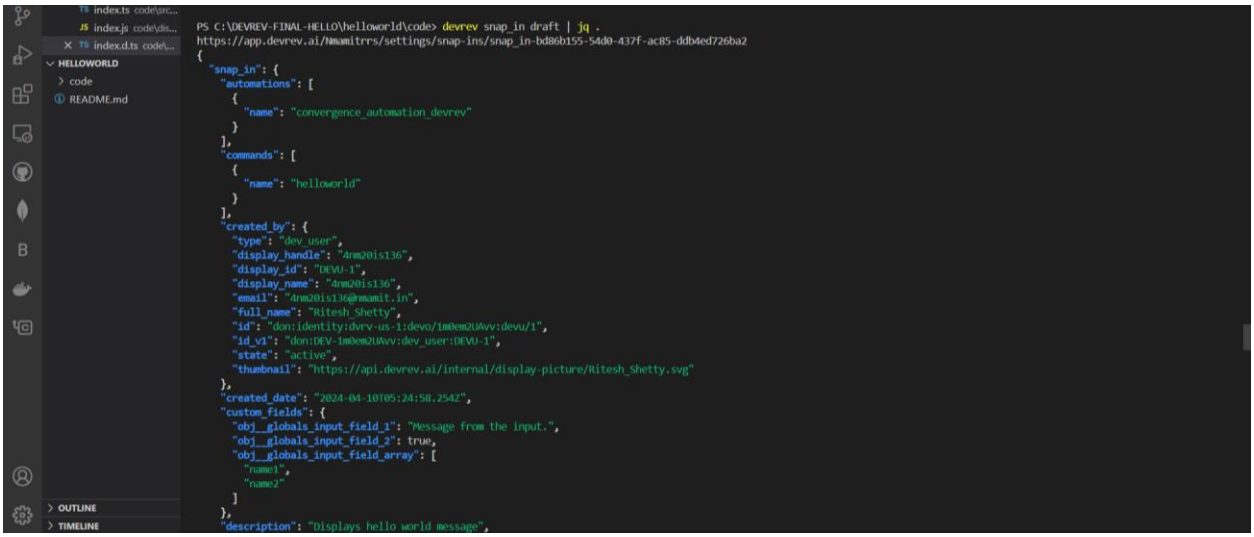


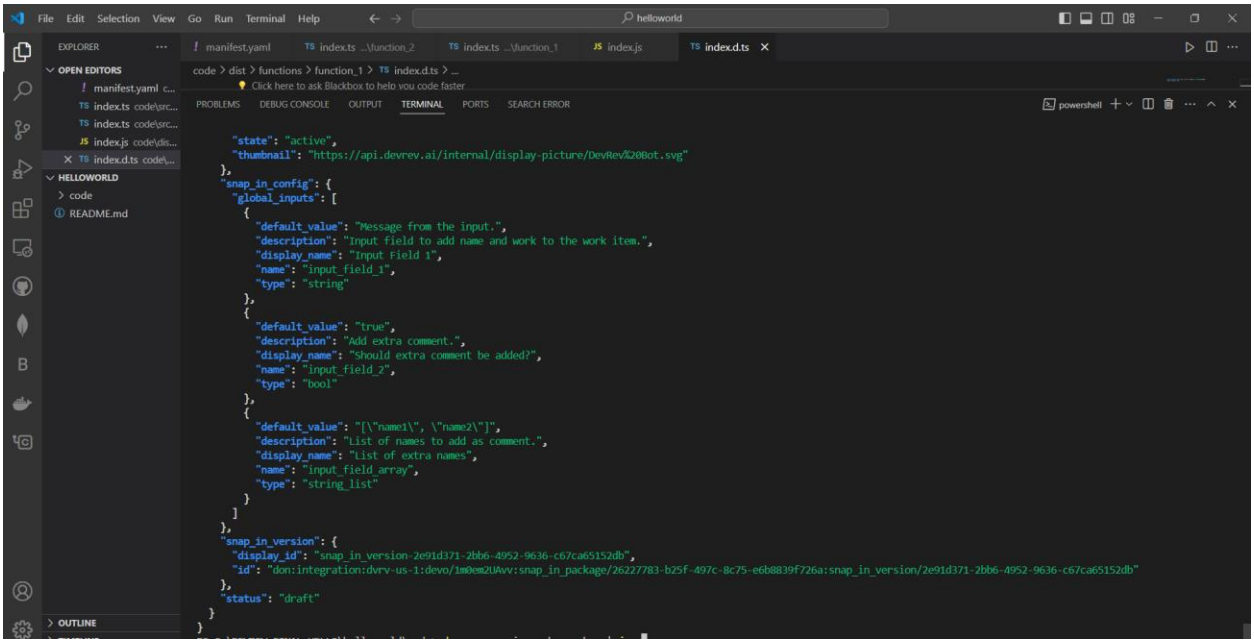
6. Create Snap-in Draft:

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

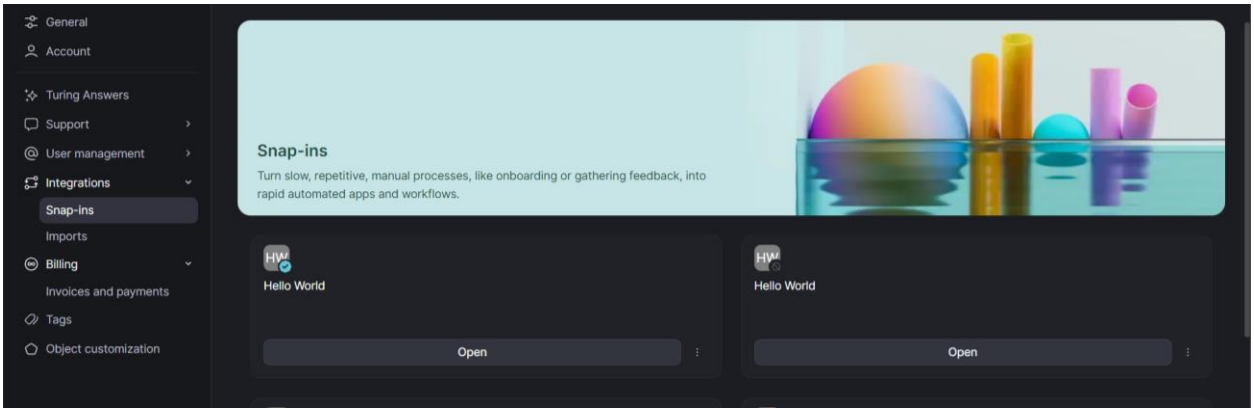
```
devrev snap_in draft --snap_in_version [id]
```

- 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.

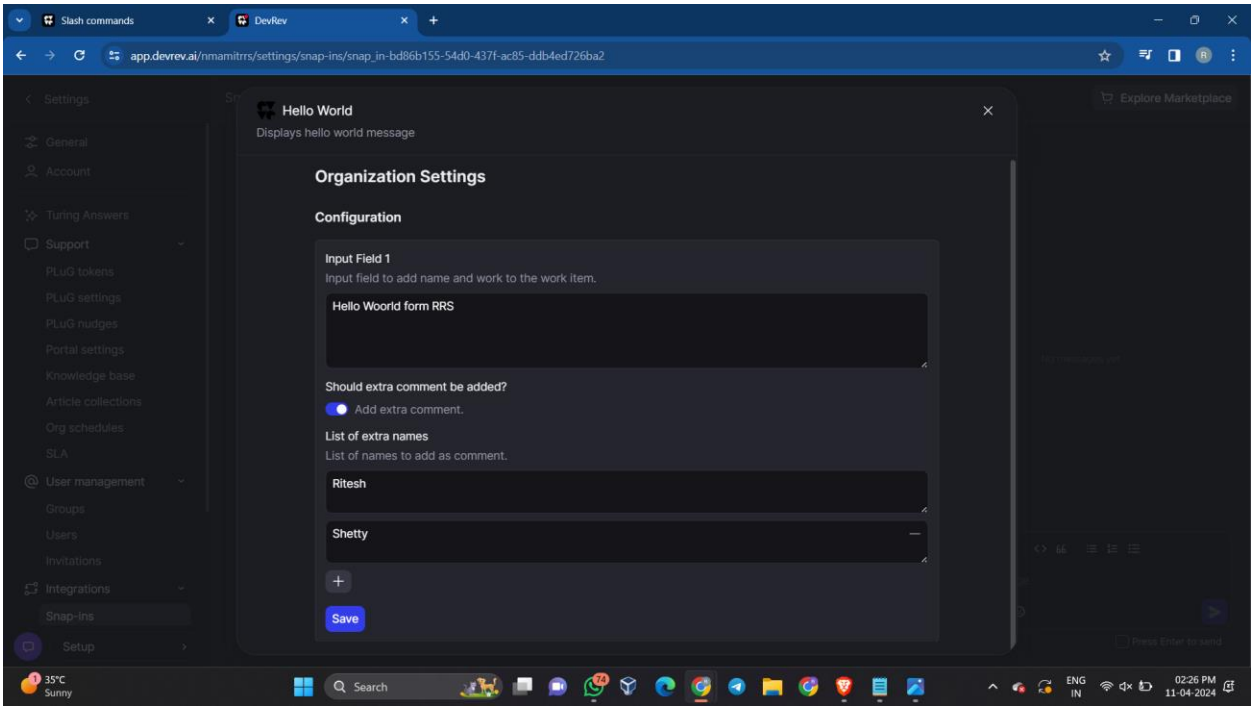




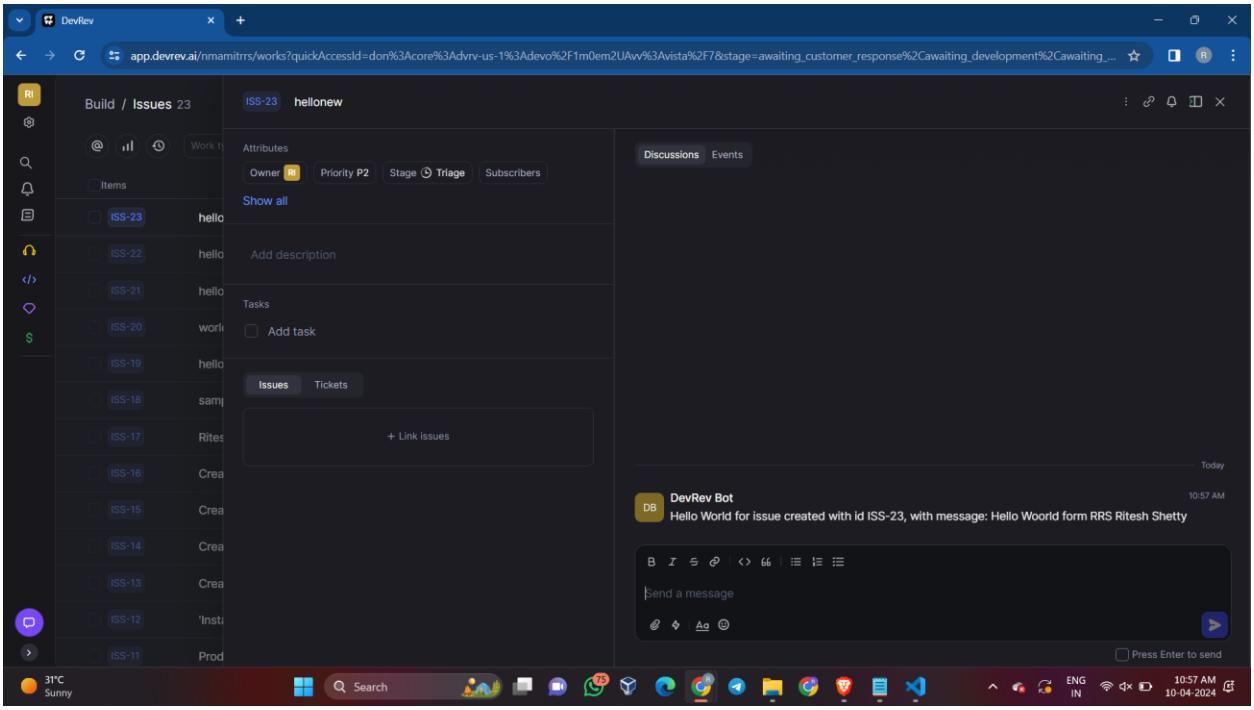
Output For Hello-World Snap-in:



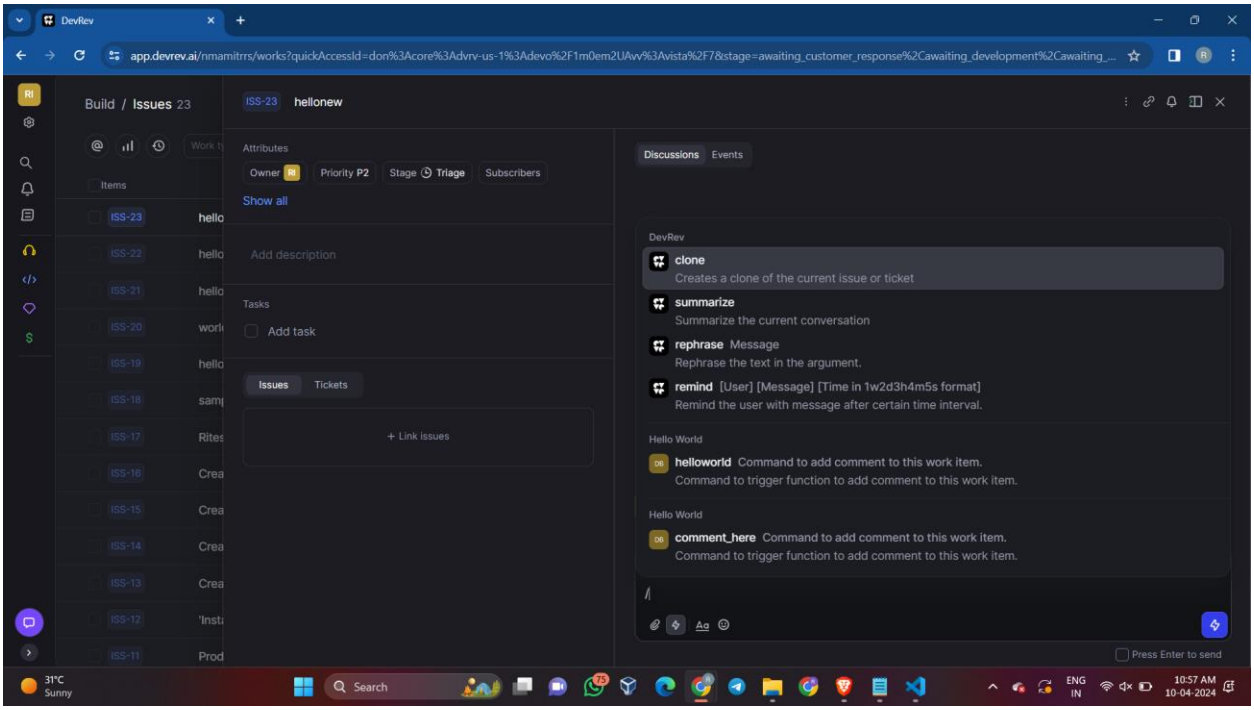
Configure and Deploy:



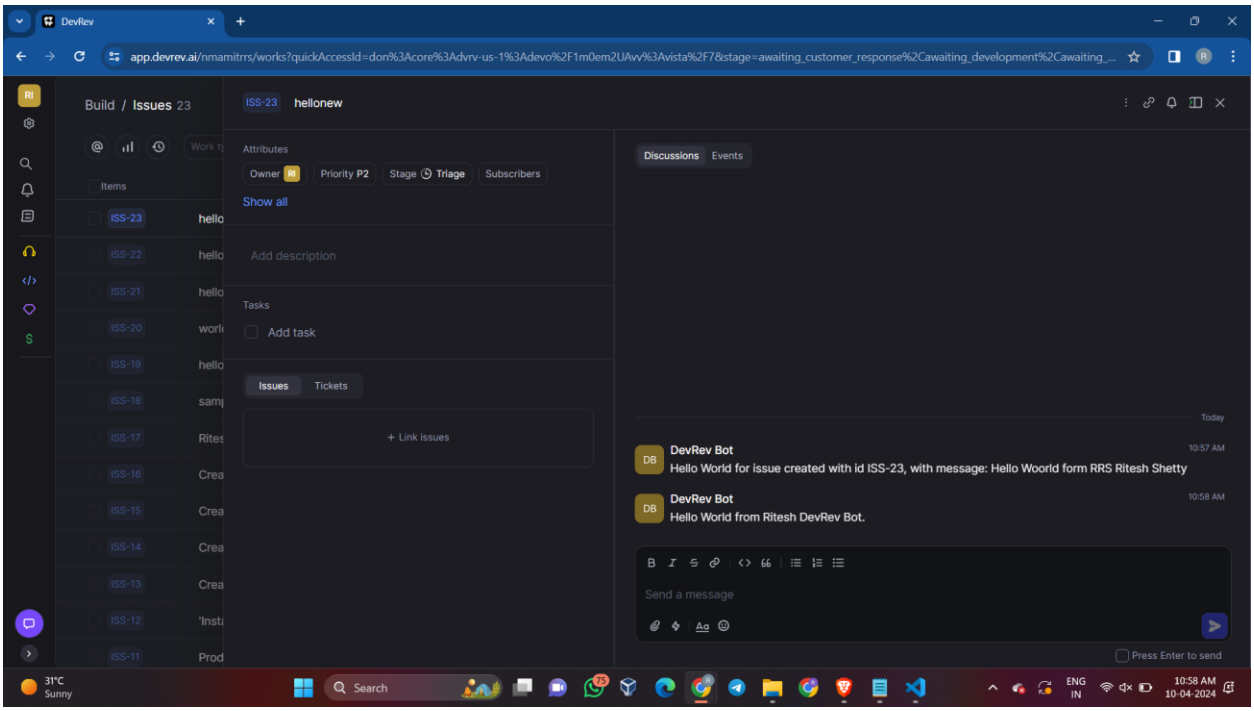
Go to Issues or Tickets or anything else as per the compatablity and create a new item:
Hello world gets executed, this is a automated reply provided by the snap-in (Method-1)
For Issues:



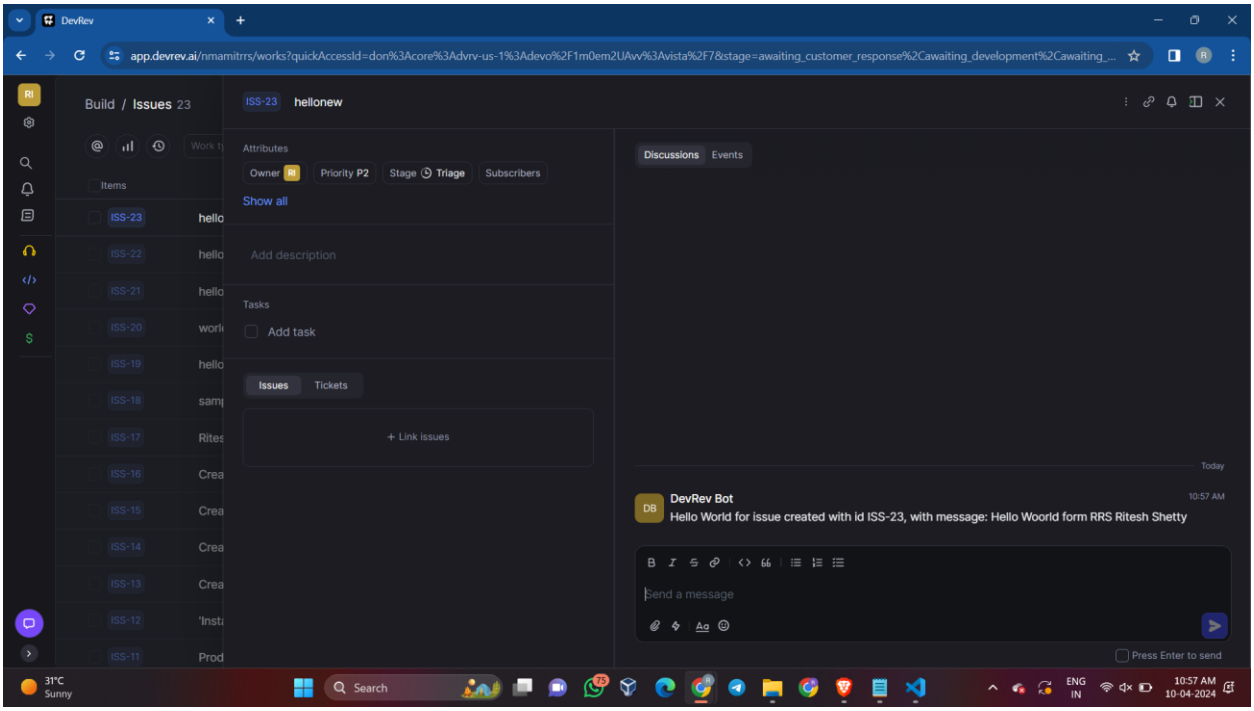
For method 2 , using slash command, click on the input box provided and type ’/’, list of all the slash command will be visible
Select helloworld slash command and execute

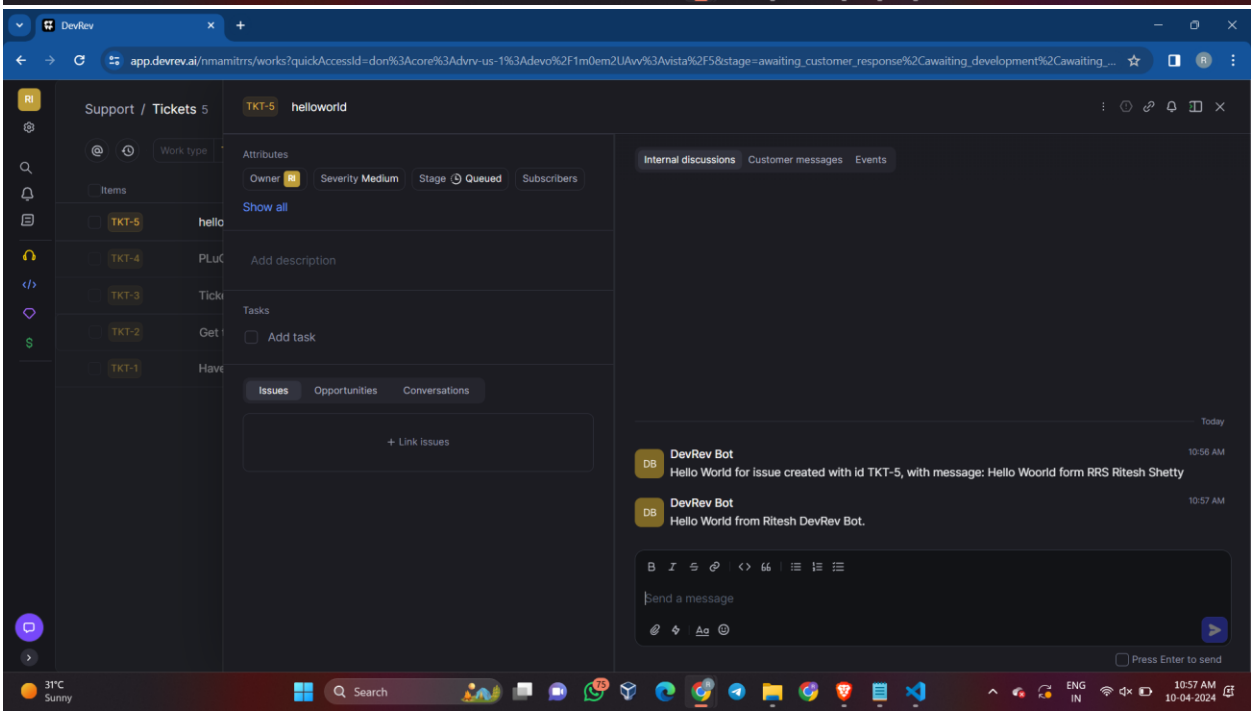


Method 2 output using slash command



For Tokens:

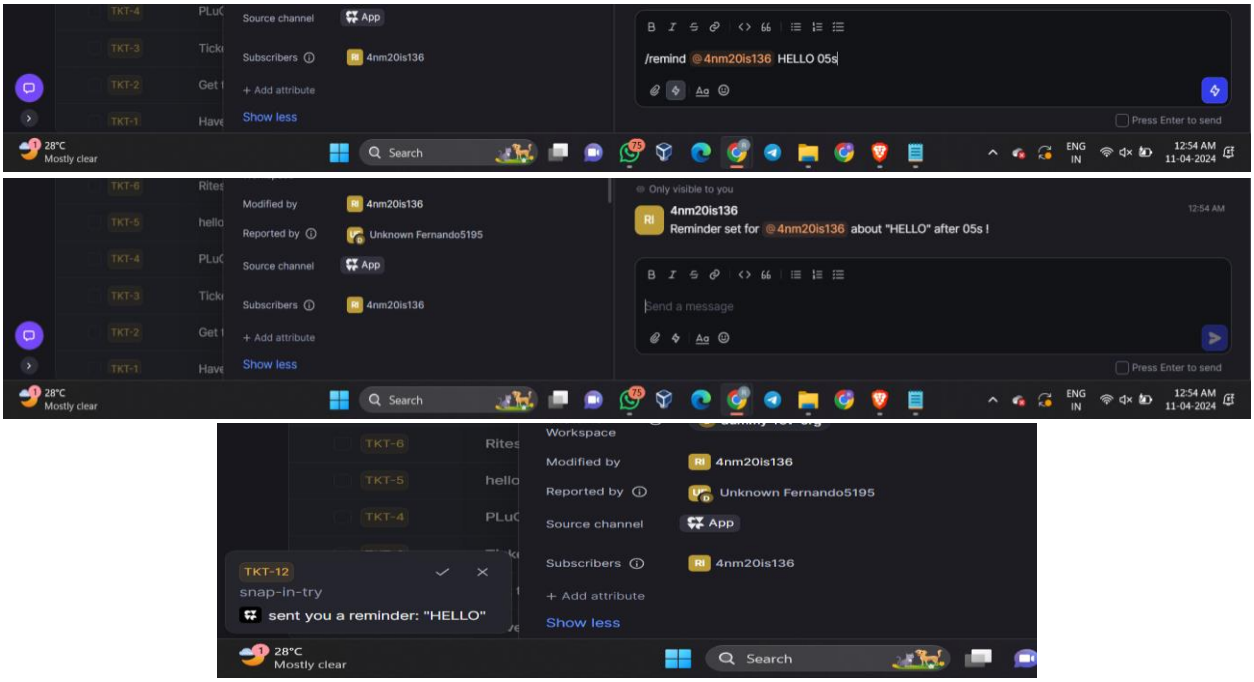




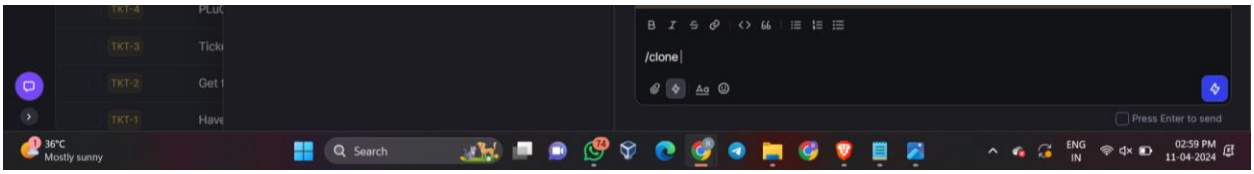
Output for Snap-in:

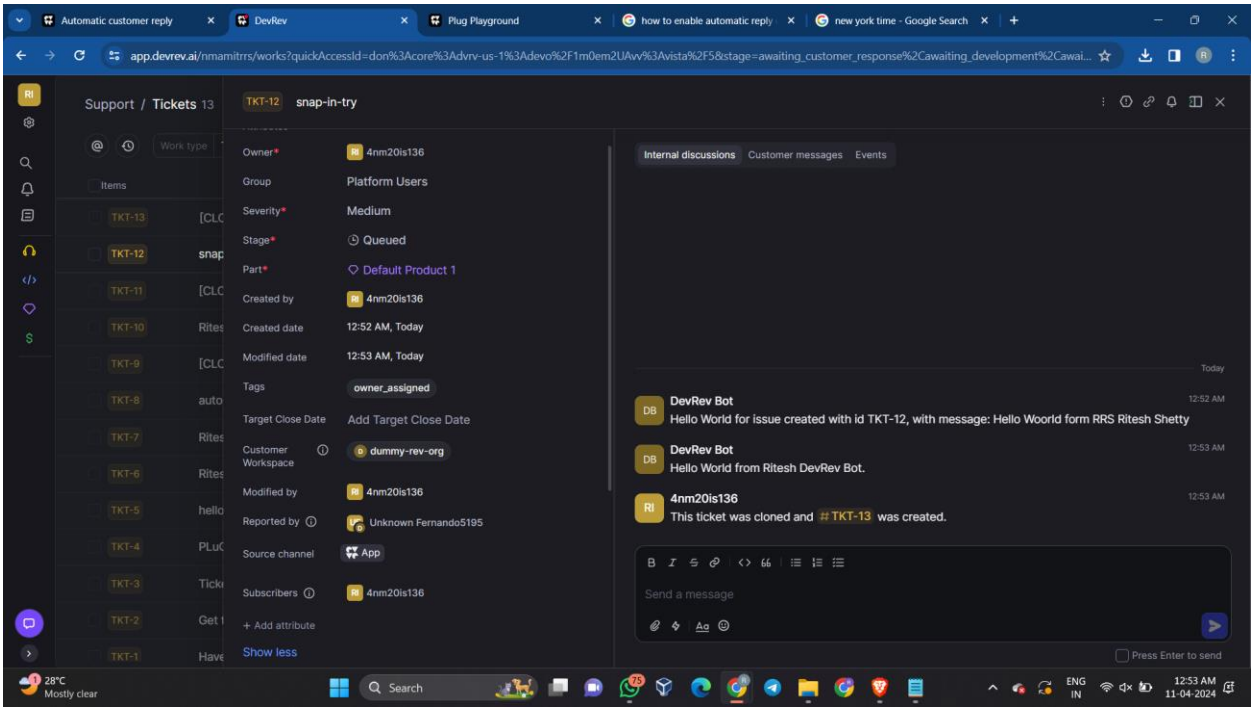
- Configure the snap-in in setting
- Deploy the Snap-in
- Come back to the Main page
- Either create a new Issue or Ticket
- Within it execute the slash command or function defined

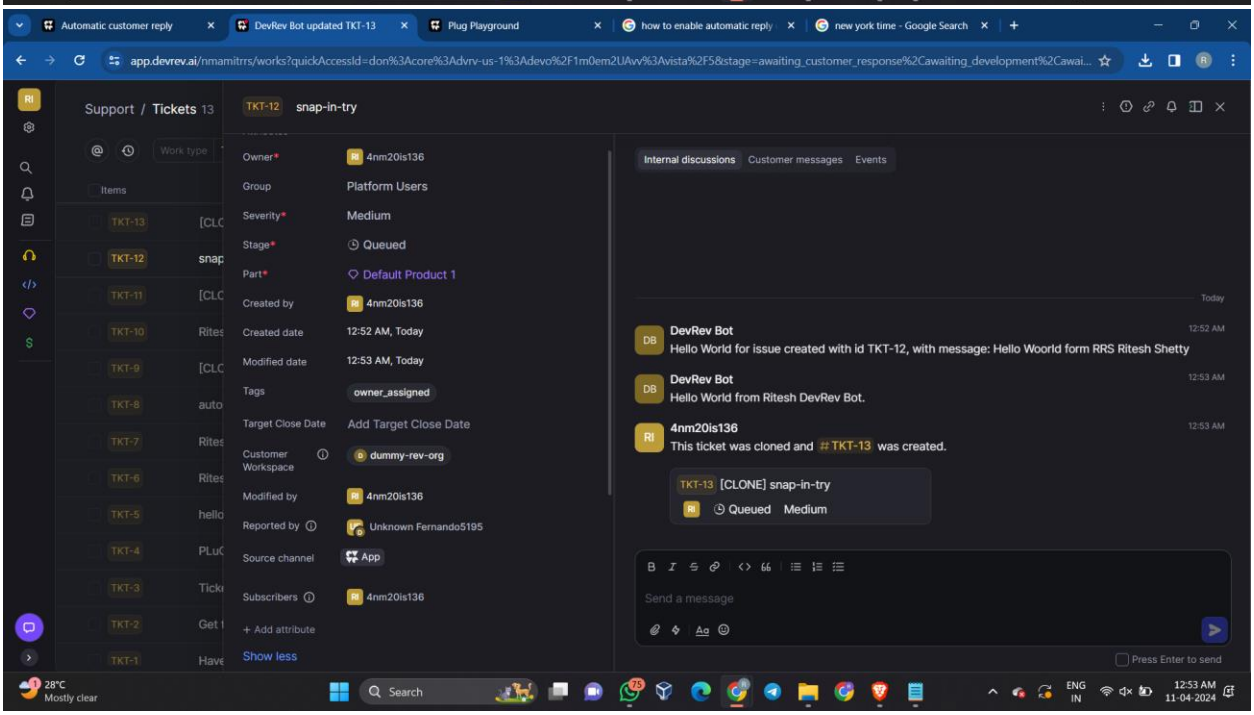
1. **Schedule a reminder for someone by tagging them on a work item after MM HH DD time. Where MM is the minute, HH is the hour and DD is the time.**

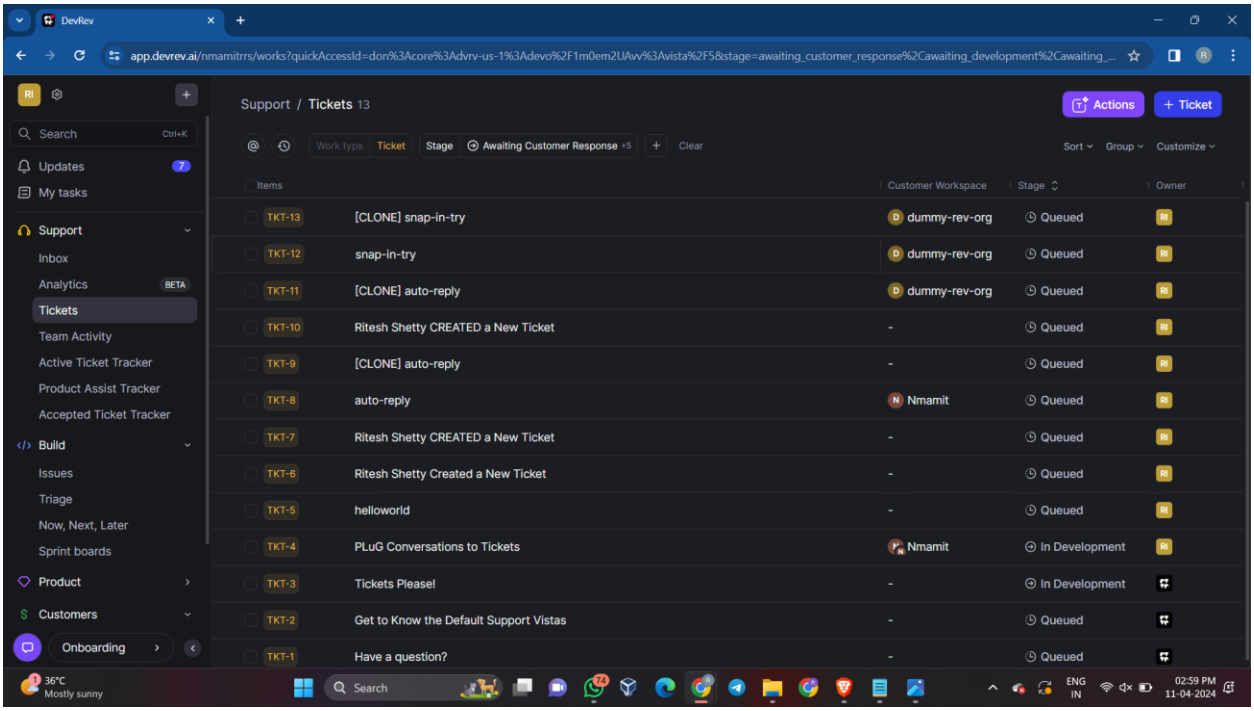


2. **Command to clone a work item (issue/ticket).**



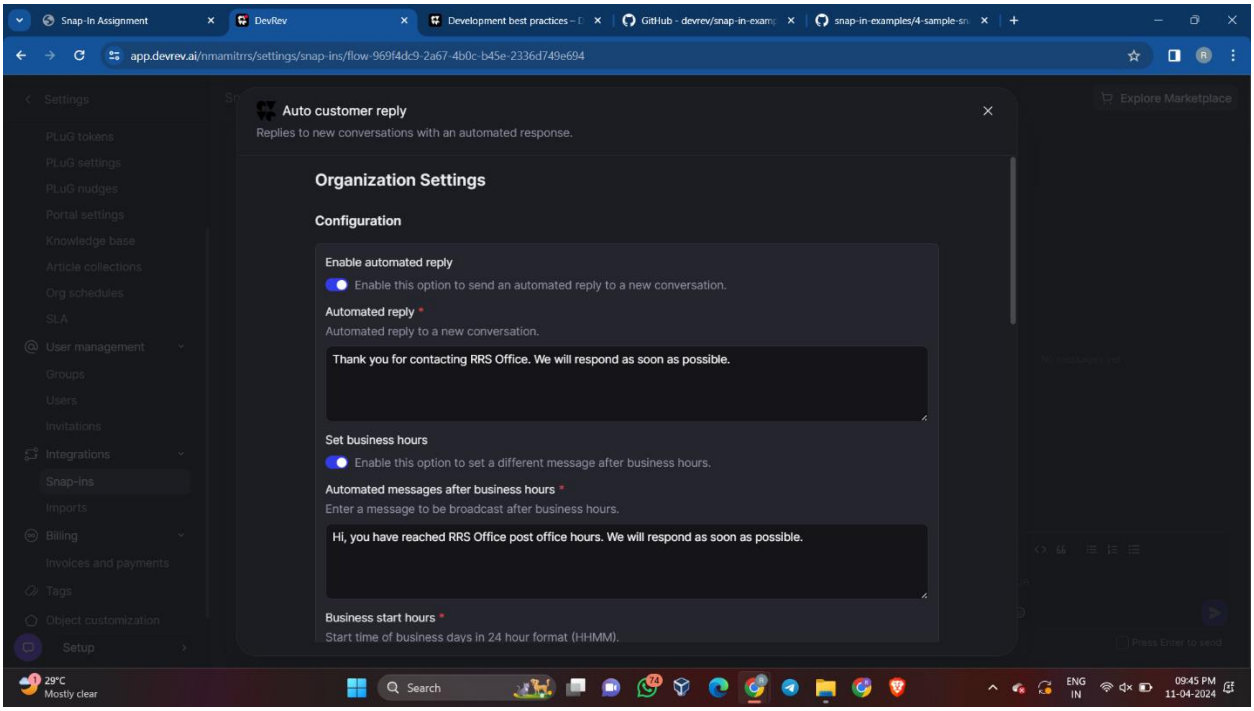

Automatic customer reply
DevRev
Plug Playground
how to enable automatic reply
new york time - Google Search
app.devrev.ai/nmamitrrs/works?quickAccessId=don%3Acore%3Advrv-us-1%3Adevo%2F1m0em2UAvv%3Avista%2F5&stage=awaiting_customer_response%2Cawaiting_development%2Cawai...
Support / Tickets 13
TKT-12 snap-in-try
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
+ Add attribute
Show less
Internal discussions
Customer messages
Events
Today
DevRev Bot 12:52 AM
Hello World for issue created with id TKT-12, with message: Hello Woorld form RRS Ritesh Shetty
DevRev Bot 12:53 AM
Hello World from Ritesh DevRev Bot.
4nm20is136 12:53 AM
This ticket was cloned and #TKT-13 was created.
Send a message
Press Enter to send
28°C Mostly clear
Search
ENG IN
12:53 AM 11-04-2024

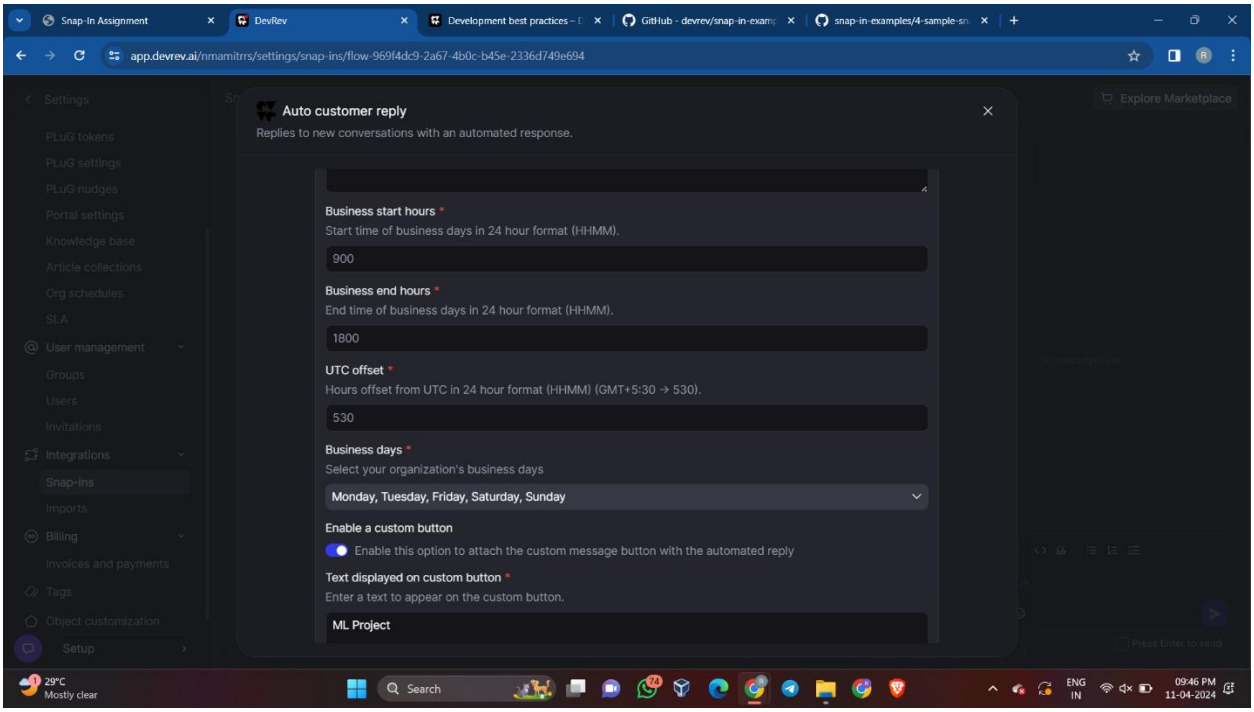

Automatic customer reply
DevRev Bot updated TKT-13
Plug Playground
how to enable automatic reply
new york time - Google Search
app.devrev.ai/nmamitrrs/works?quickAccessId=don%3Acore%3Advrv-us-1%3Adevo%2F1m0em2UAvv%3Avista%2F5&stage=awaiting_customer_response%2Cawaiting_development%2Cawai...
Support / Tickets 13
TKT-12 snap-in-try
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
+ Add attribute
Show less
Internal discussions
Customer messages
Events
Today
DevRev Bot 12:52 AM
Hello World for issue created with id TKT-12, with message: Hello Woorld form RRS Ritesh Shetty
DevRev Bot 12:53 AM
Hello World from Ritesh DevRev Bot.
4nm20is136 12:53 AM
This ticket was cloned and #TKT-13 was created.
TKT-13 [CLONE] snap-in-try
Queued Medium
Send a message
Press Enter to send
28°C Mostly clear
Search
ENG IN
12:53 AM 11-04-2024

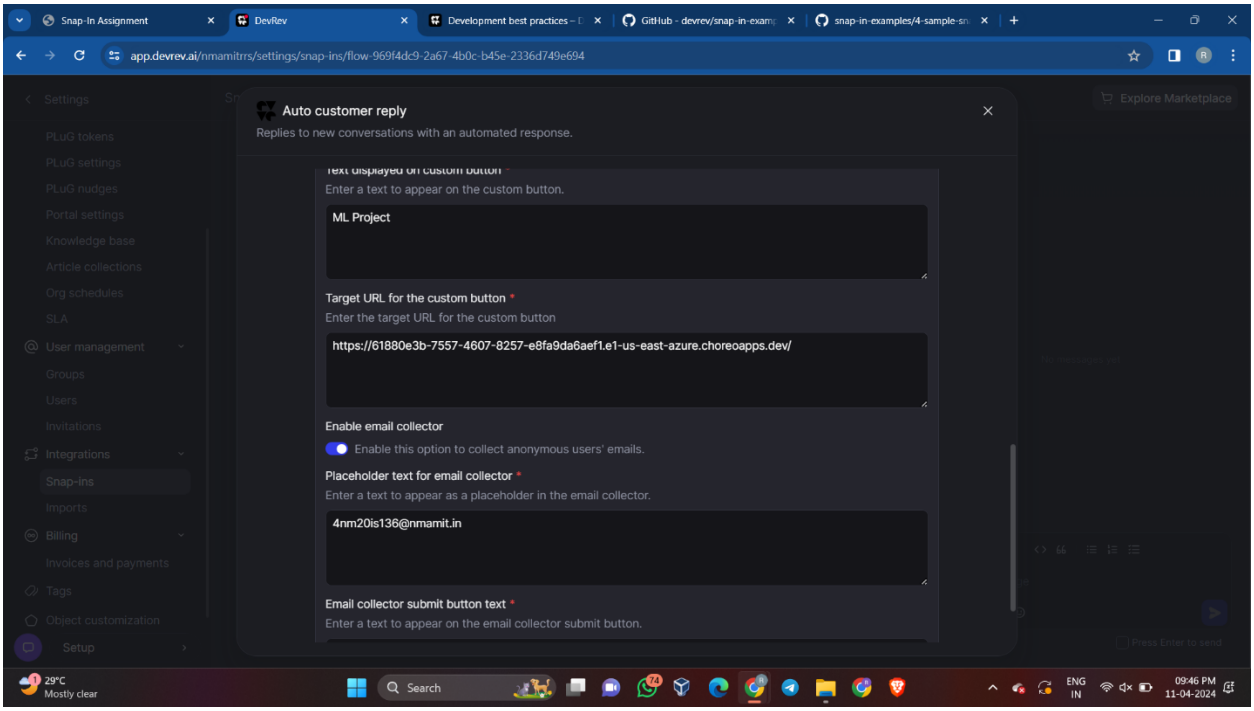


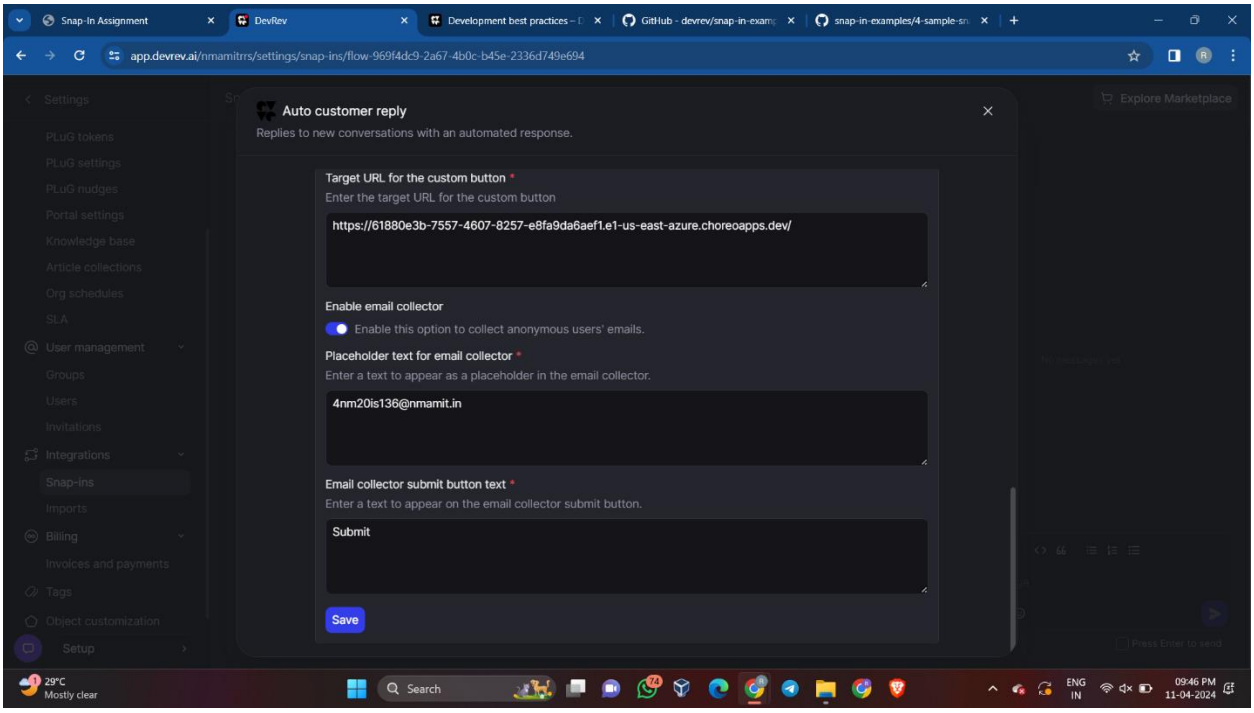
3. **Auto-Reply to a message if it is outside office hours. The start and end time for office hours can be specified in the input. The auto-reply message can also be taken as an input.**

Configure the Snap-in

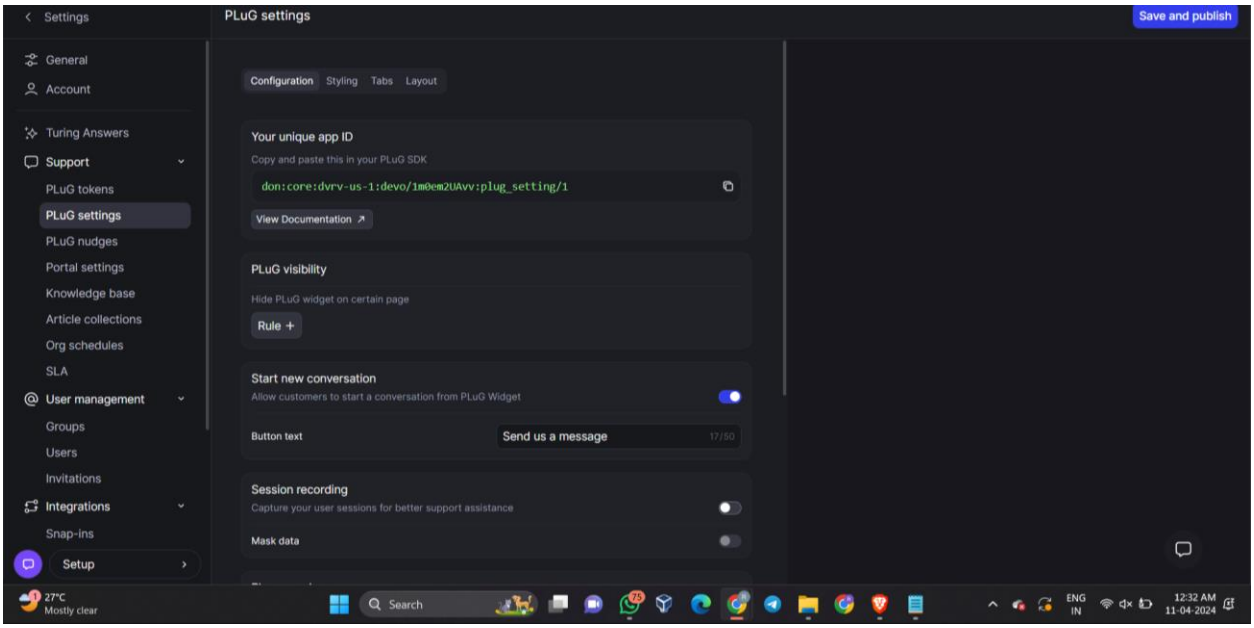



Snap-In Assignment
DevRev
app.devrev.ai/nmamitrrs/settings/snap-ins/flow-969f4dc9-2a67-4b0c-b45e-2336d749e694
Auto customer reply
Replies to new conversations with an automated response.
Business start hours *
Start time of business days in 24 hour format (HHMM).
900
Business end hours *
End time of 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

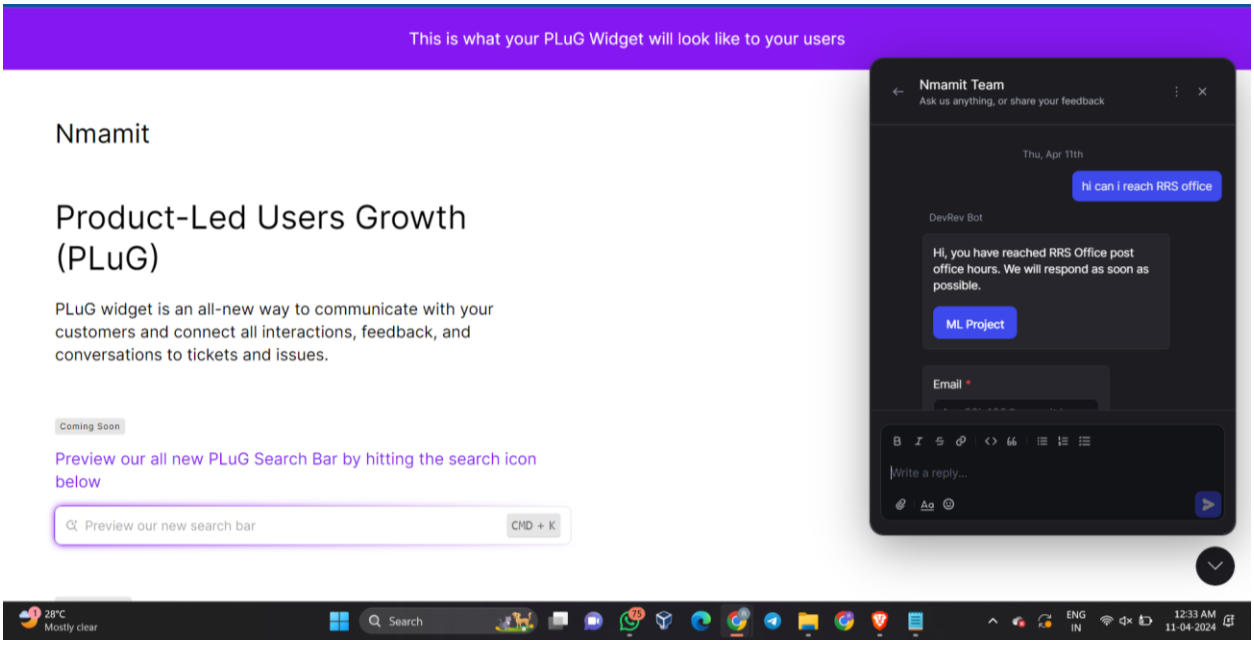

Snap-In Assignment
DevRev
app.devrev.ai/nmamitrrs/settings/snap-ins/flow-969f4dc9-2a67-4b0c-b45e-2336d749e694
Auto customer reply
Replies to new conversations with an automated response.
Enter a text to appear on the custom button.
ML Project
Target URL for the custom button *
Enter the target URL for the custom button
https://61880e3b-7557-4607-8257-e8fa9da6aef1.e1-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.

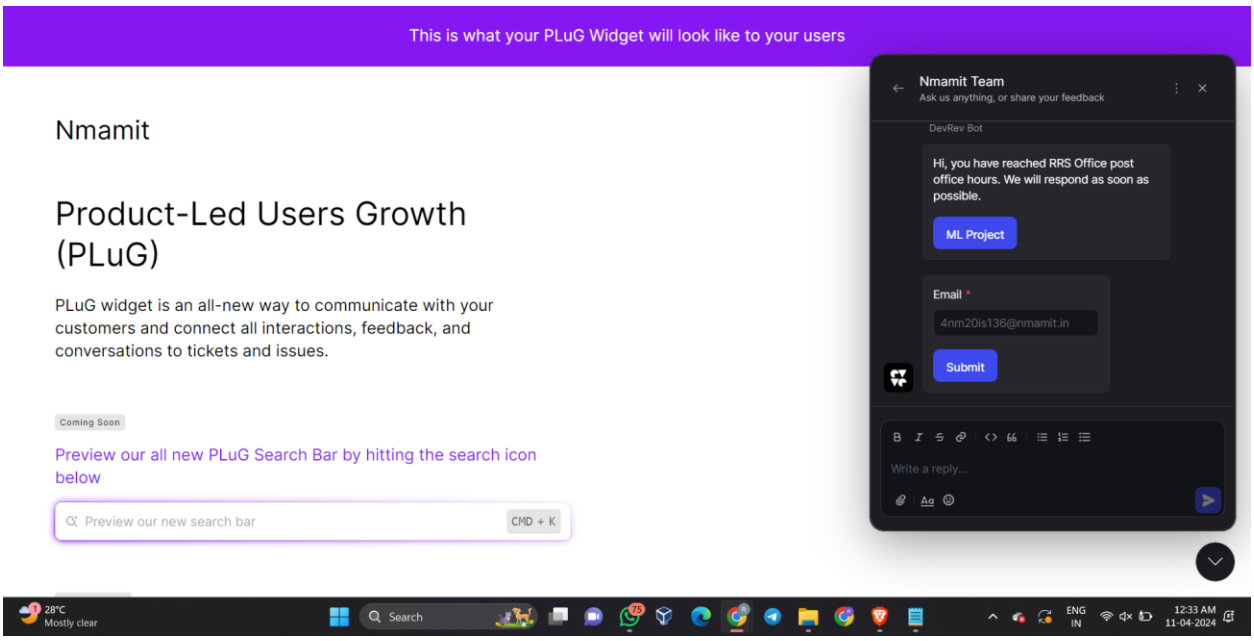


Setting a PLuG in DevRev

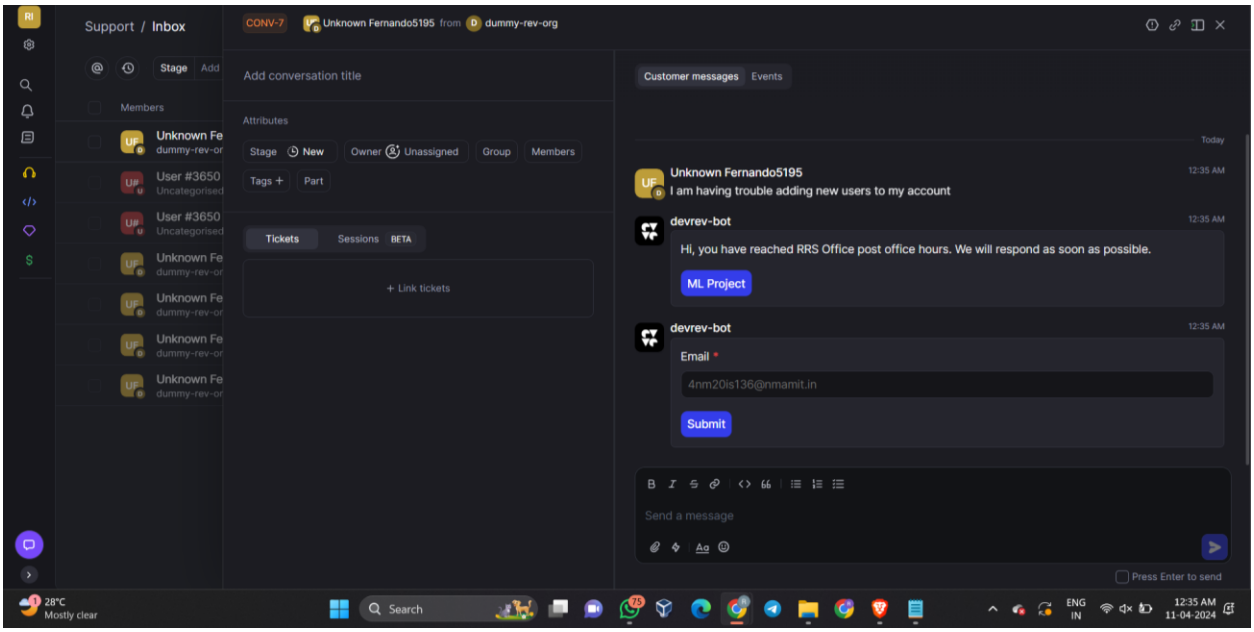


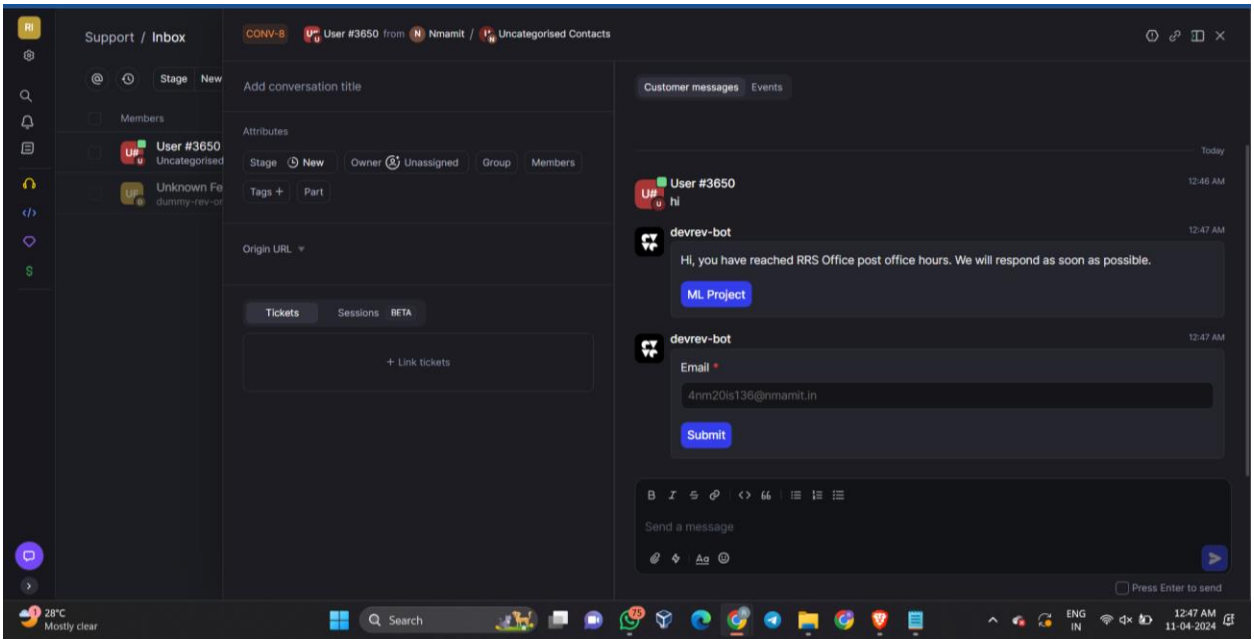
With PLuG:

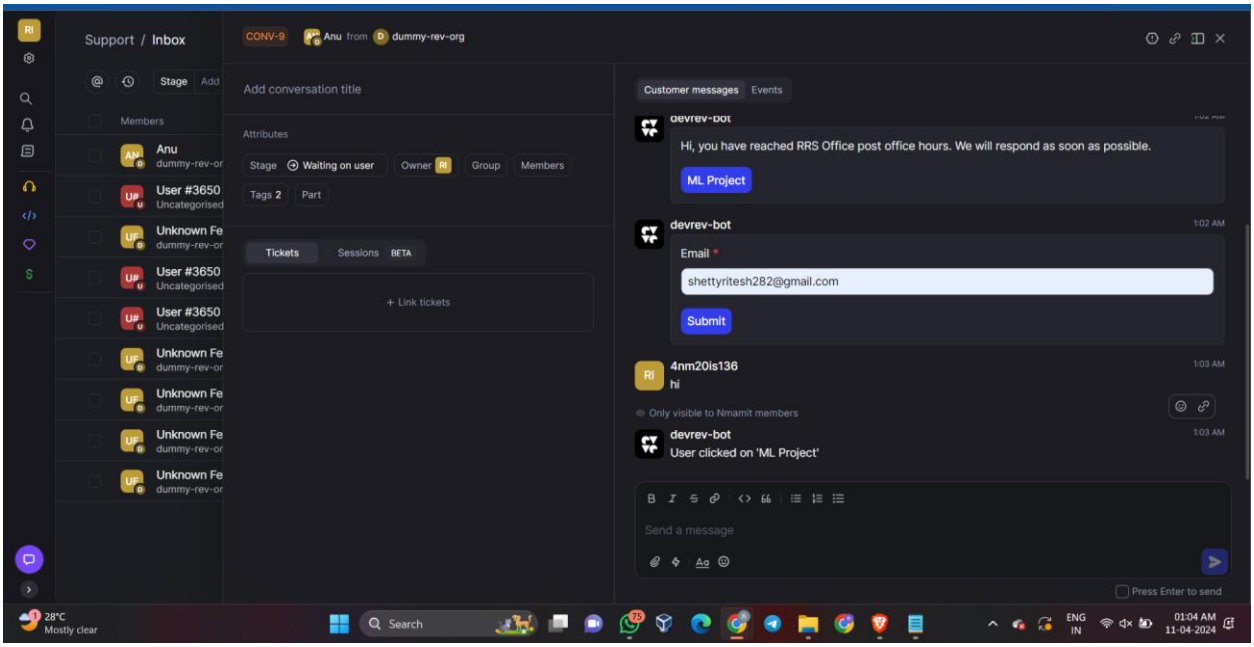




Without using PLuG:
Create a New Customer Conversation under Support > Inbox on the Main Page


Support / Inbox
CONV-7
Unknown Fernando5195 from dummy-rev-org
Add conversation title
Attributes
Stage New
Owner Unassigned
Group
Members
Tags +
Part
Tickets
Sessions BETA
+ Link tickets
Customer messages
Events
Today
Unknown Fernando5195
I am having trouble adding new users to my account
12:35 AM
devrev-bot
Hi, you have reached RRS Office post office hours. We will respond as soon as possible.
ML Project
Email *
4nm20is136@nmamit.in
Submit
Send a message
Press Enter to send
28°C
Mostly clear
Search
ENG
IN
12:35 AM
11-04-2024


Support / Inbox
CONV-8
User #3650 from Nmamit / Uncategorised Contacts
Add conversation title
Attributes
Stage New
Owner Unassigned
Group
Members
Tags +
Part
Origin URL
Tickets
Sessions BETA
+ Link tickets
Customer messages
Events
Today
User #3650
hi
12:46 AM
devrev-bot
Hi, you have reached RRS Office post office hours. We will respond as soon as possible.
ML Project
Email *
4nm20is136@nmamit.in
Submit
Send a message
Press Enter to send
28°C
Mostly clear
Search
ENG
IN
12:47 AM
11-04-2024


Support / Inbox
CONV-9
Anu from dummy-rev-org
Add conversation title
Attributes
Stage
Waiting on user
Owner
Group
Members
Tags 2
Part
Tickets
Sessions BETA
+ Link tickets
Customer messages
Events
devrev-bot
Hi, you have reached RRS Office post office hours. We will respond as soon as possible.
ML Project
devrev-bot
1:02 AM
Email *
shettyritesh282@gmail.com
Submit
4nm20is136
1:03 AM
hi
Only visible to Nmamit members
devrev-bot
1:03 AM
User clicked on 'ML Project'
Send a message
Press Enter to send
28°C
Mostly clear
Search
ENG
IN
01:04 AM
11-04-2024

****

Thank You