IBM watsonx Gen AI Challenge

Lab 3

IBM watsonx Assistant

Authors: Joël Hartmann, Valentin Meyer

Last edited: 29 April 2025

Zurich, CH

Content

Acknowledgments

We would like to thank Dorothee Reinhard for helping us to create and refine this lab.

Contact

Dorothee Reinhard	dorothee.reinhard@ch.ibm.com
Karim Ibrahim	karim.ibrahim@ibm.com
Dean Heinzmann	dean.heizmann@ibm.com
Valentin Meyer	valentin.meyer@ibm.com
Joël Hartmann	joel.hartmann@ibm.com

Notice

Every Group has access to:

1 x watsonx assistant **Plus Plan**

1x watsonx assistant **Trial Plan**

You should use the Trial Plan for completing this lab as well as to play around, test things and to understand how watsonx assistant works while you must use the Plus Plan for implementing your final solution for your specific use-case. In case of any questions do hesistate to contact us directly or via slack.

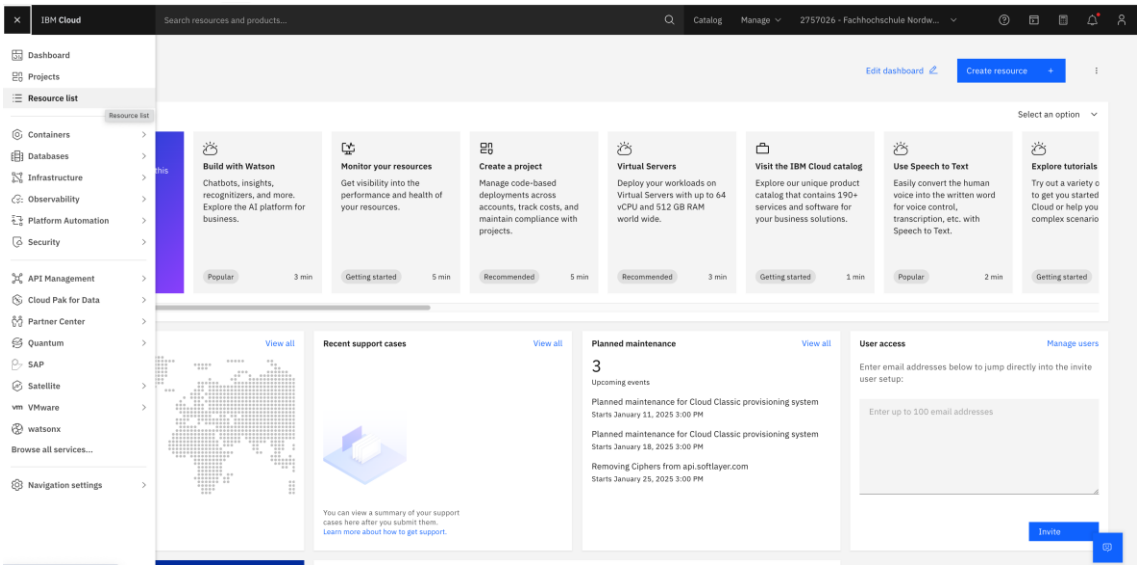
Note: Every Group Member can create its own assistant in the Trial Plan but not in the Plus Plan. In the Plus Plan, every group has exactly one assistant.

Important!

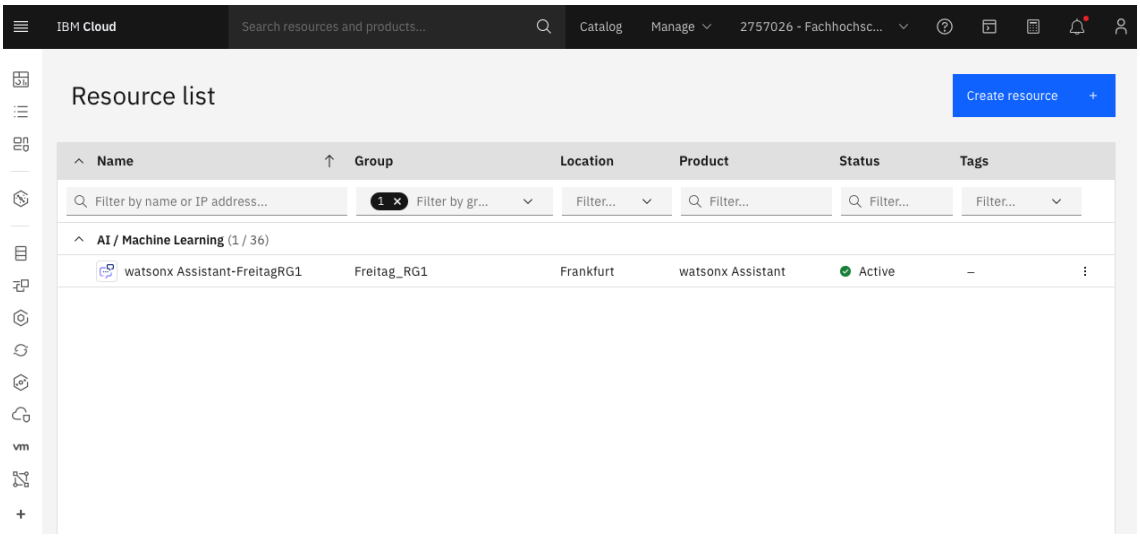
You may **skip the first task**, as the watsonx assistant instance has already been set up. Each group has access to one watsonx assistant, which can be found in your resource list under AI / Machine Learning. **Your assigned watsonx assistant corresponds to your group number.** Therefore, you can proceed directly to **Task 2**. Enjoy!

Task 1: Select the watsonx Assistant Trial Instance in the IBM Cloud

1. In a web browser, navigate to the following URL https://cloud.ibm.com/
2. Log in with your IBM Cloud credentials. This should be **your IBMid.**
3. You should view the dashboard which shows a list of your applications and services.
4. Open the hamburger menu located in the top left corner and select "**Resource List**"



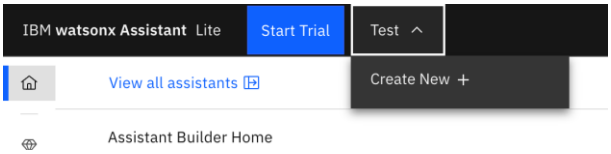
5. You should now see both **project specific watsonx assistant instances (Plus & Trial Plans).** (Note: The Plus Plan is under the Freitag Resource Group, don't worry about that.)



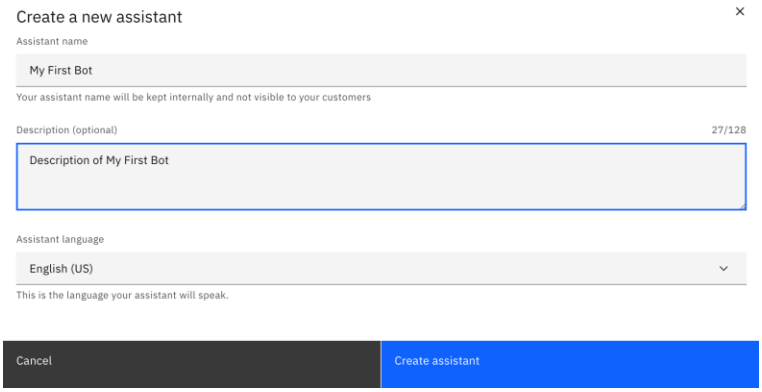
6. Click on the watsonx assistant instance with the **TRIAL** suffix.

7. In the *Credentials* section click **Show Credentials**. You should see the *API Key* for your service. Feel free to copy them to your clipboard, to a text file, or just return to this section of the IBM Cloud web interface when the credentials are needed.

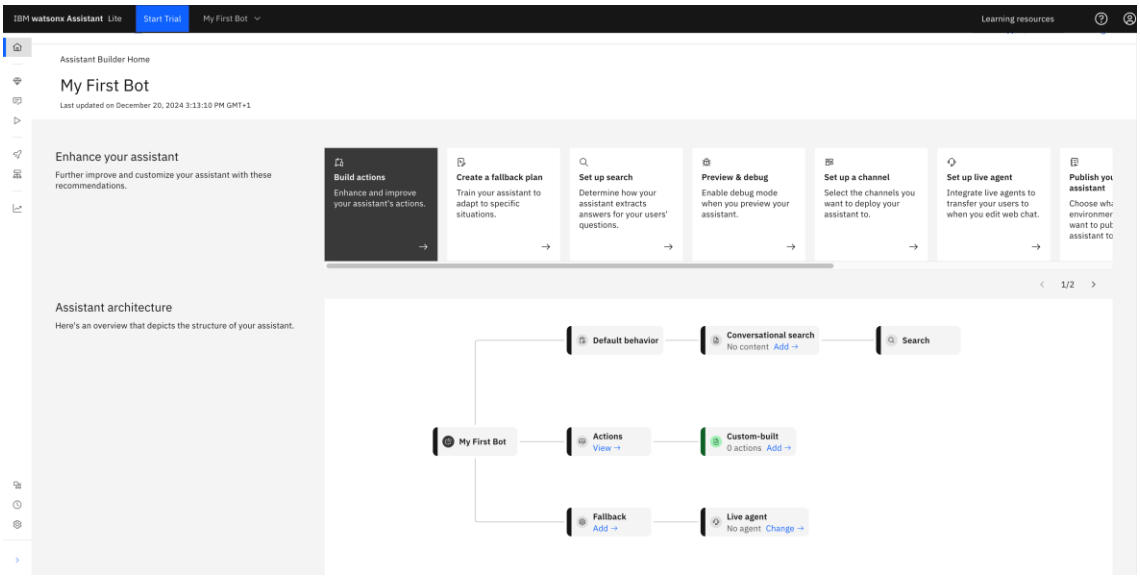
8. **Launch the watsonX** assistant by clicking the **Launch Watson Assistant** Button.
9. Create a new assistant → select **Create New +**



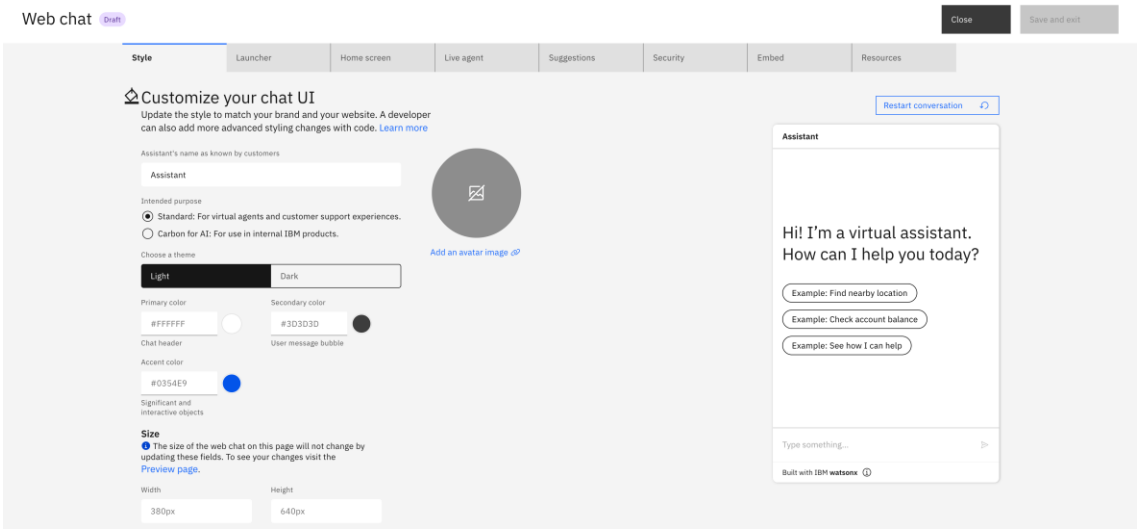
10. **Enter a name** for your assistant, a **description** and choose its **language**.
(Each Group Member can create their own assistant in the TRIAL Plan but not in the PLUS Plan.)



11. **Congratulations you have created your first watsonx Assistant!**
    You will be greeted with the following menu:

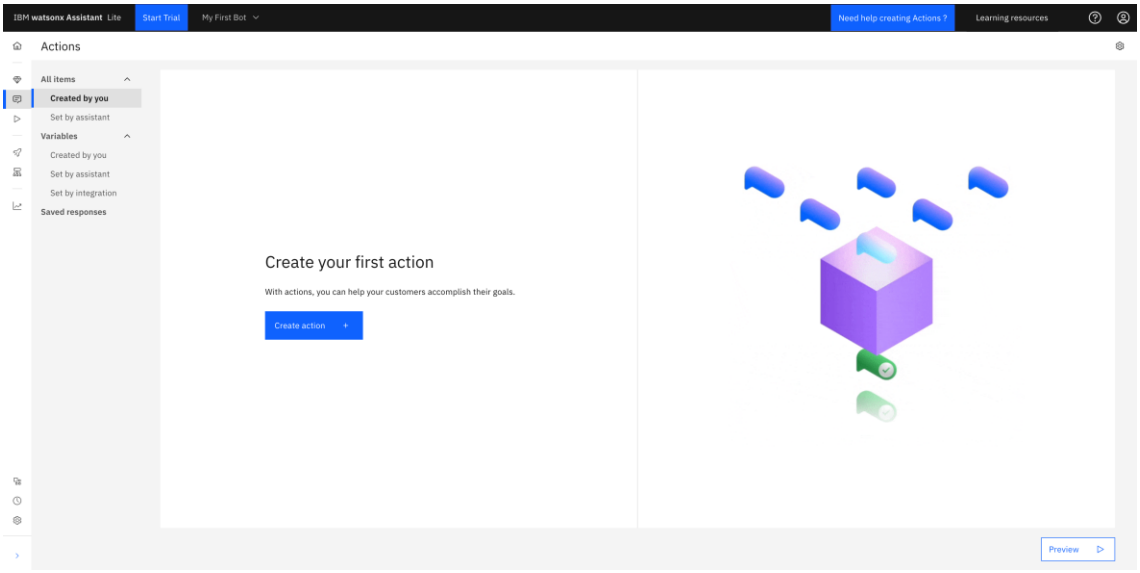


12. Now you can see its **architecture** and **enhance** your assistant (e.g., you can select **Customize web chat** and design your chat UI and see a preview of your assistant):

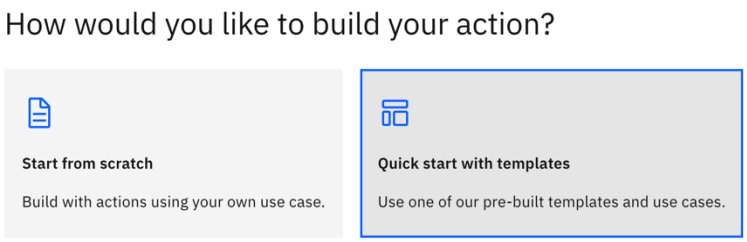


Task 2: Create an Action from Template

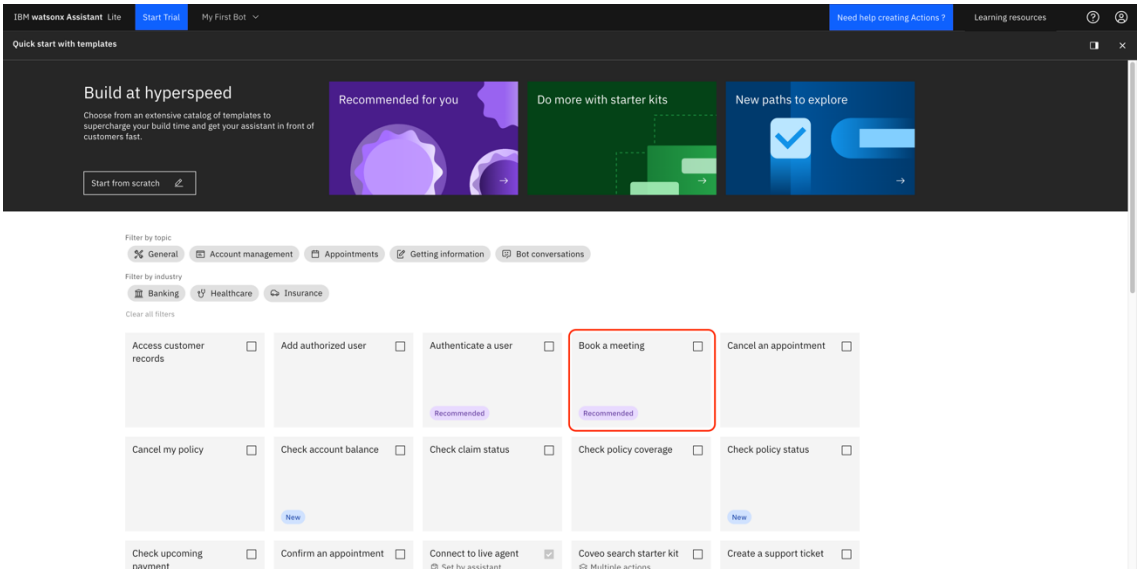
1. In the **Assistant Builder Home** view (see Task 1, Step 11) select **Actions**.
2. Create a new **Action** by clicking the **Create action** button.



3. Actions can be created from scratch or from templates. Choose **Quick start with templates.**

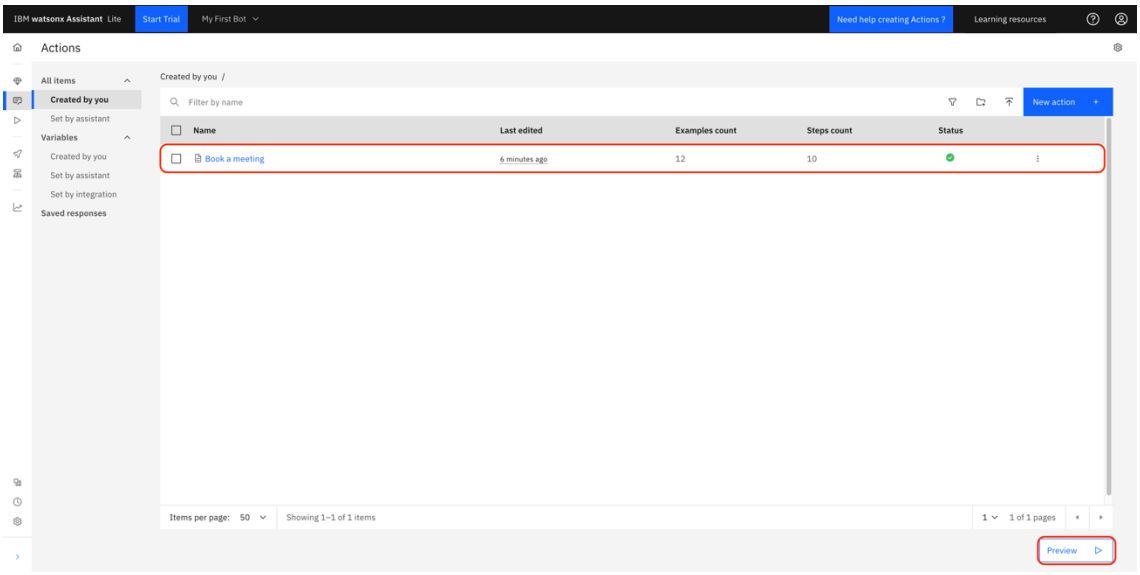


4. Select the **“Book a Meeting”** Template



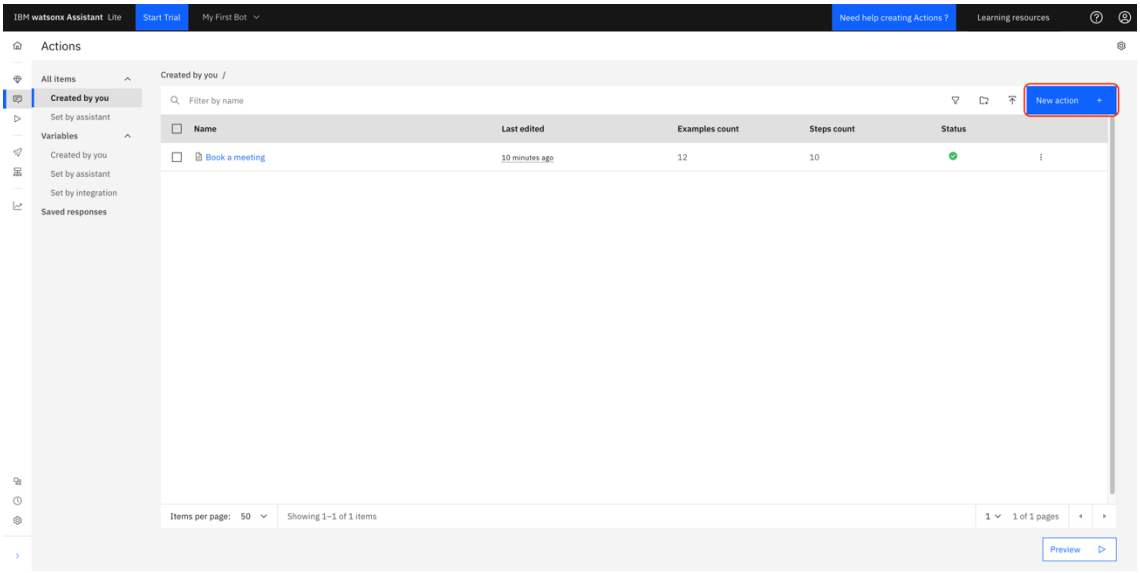
5. A sidebar will open with all the selected templates. Click **Add templates**.

6. **Congratulations!** You have added a new *Action from Template*. Select your newly created template **Book a meeting** under **Created by you** to customize it further. You can also interact with your assistant by clicking on **Preview**:

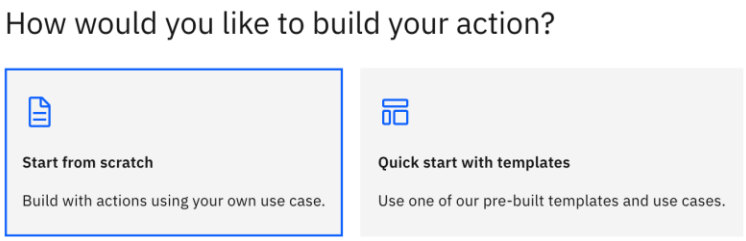


Task 3: Chitchat Action (Request/Response)

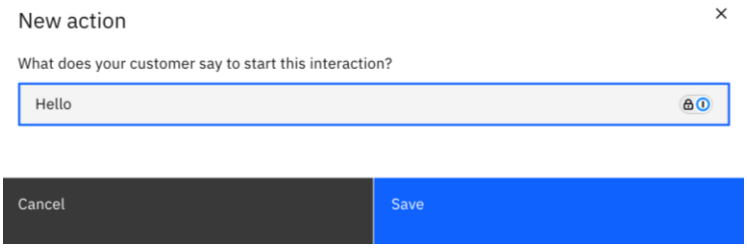
1. Create a new **action** by clicking the **New action +** button on the top right.



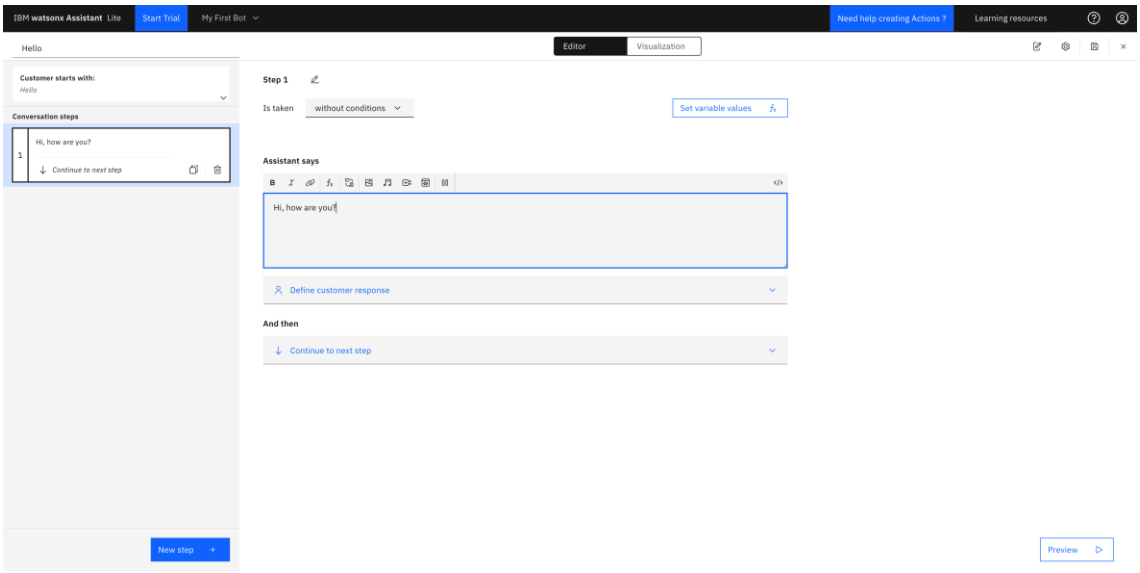
2. Actions can be created from scratch or from templates. Choose **Start from scratch.**



3. Enter in the textfield: Hello. This is what your customer would say to initiate a conversation. Click on **save**.



4. This time we will define a static response returned by the assistant. Enter in the **Assistant says** textfield: *Hi, how are you?*

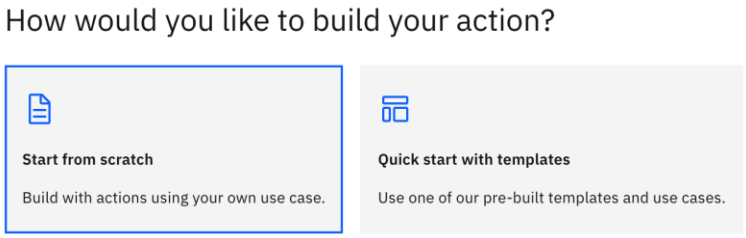


5. You can add additional steps (**New step +**) and define a more sophisticated flow of the conversation. Under **Define customer response** you can pre-define response options for the customer.
6. **Congratulations!** You have defined your first **Action from scratch**. Your assistant updates automatically, to see your new action select the **Preview** button in the bottom right corner.

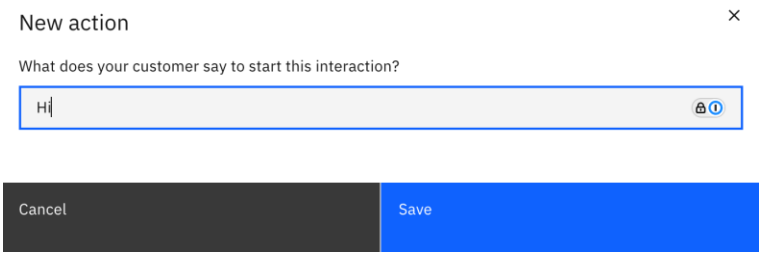
Task 4: Information Extraction & Session Variables

In this Task we will extract information provided by the user via chat and store it in a context variable. Learn more on: https://cloud.ibm.com/docs/watson-assistant?topic=watson-assistant-manage-info#store-session-variable

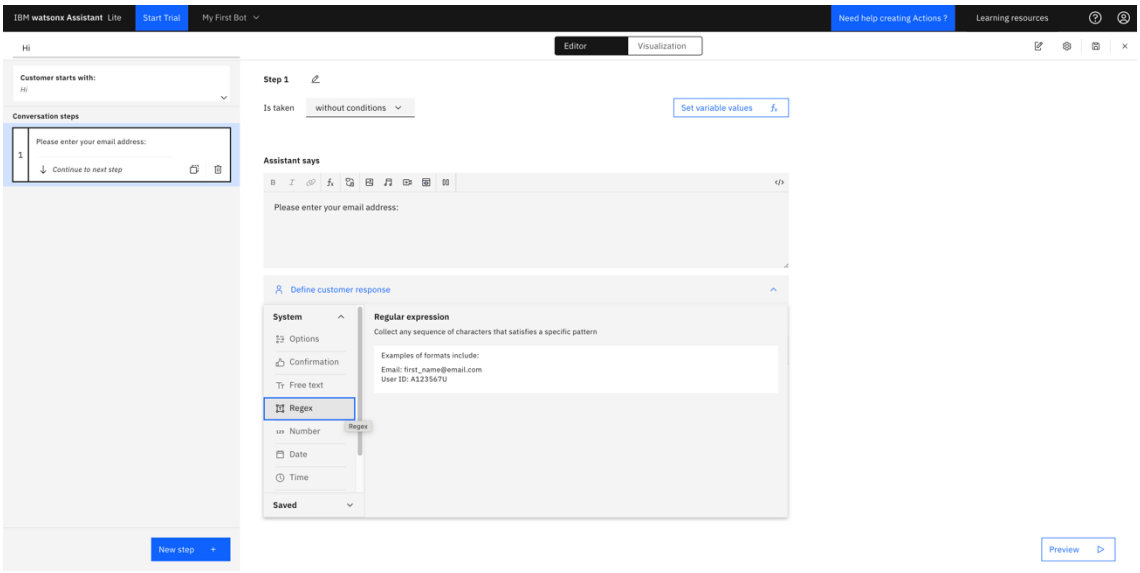
1. Create a new **action** (see Step 1, Task 3).
2. Choose **Start from scratch**:



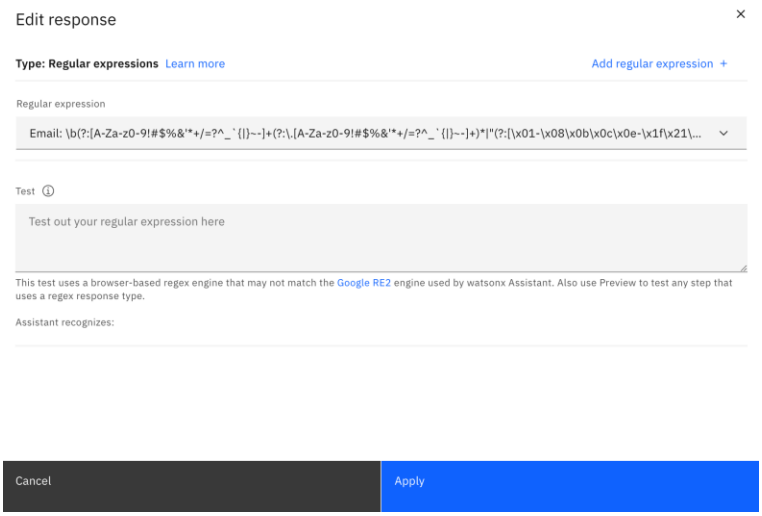
3. **Define** an example Phrase (e.g., Hi) that will start the conversation:
(Note: it should be a different on from before to not trigger the same action)



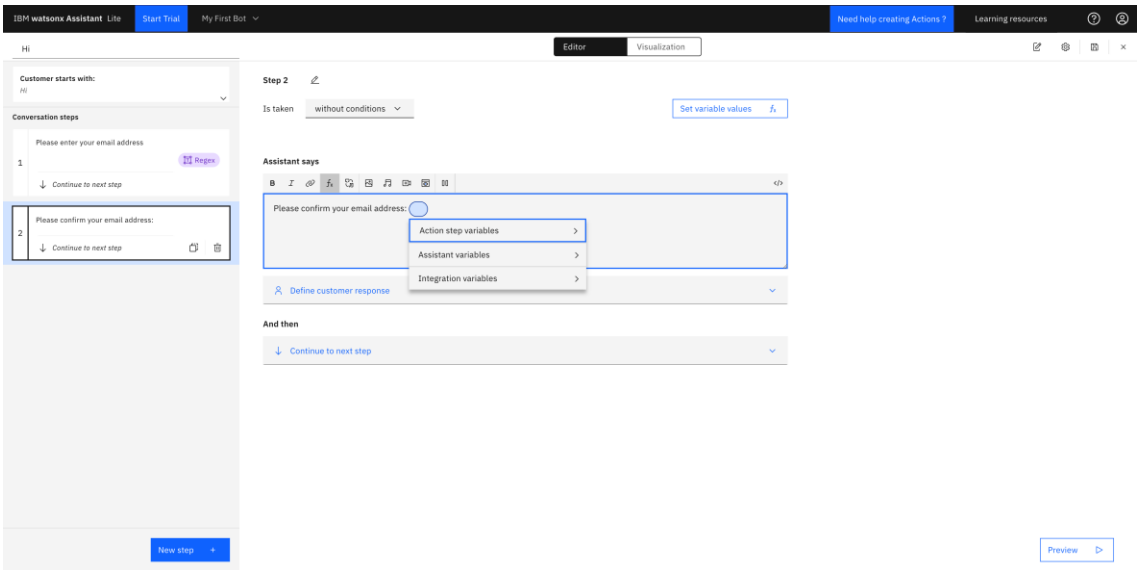
4. **Define** what the assistant says (e.g., *Please enter your email address*). Then select **Define customer response** and select **Regex** and select the **Email Regex** in the Dialog.



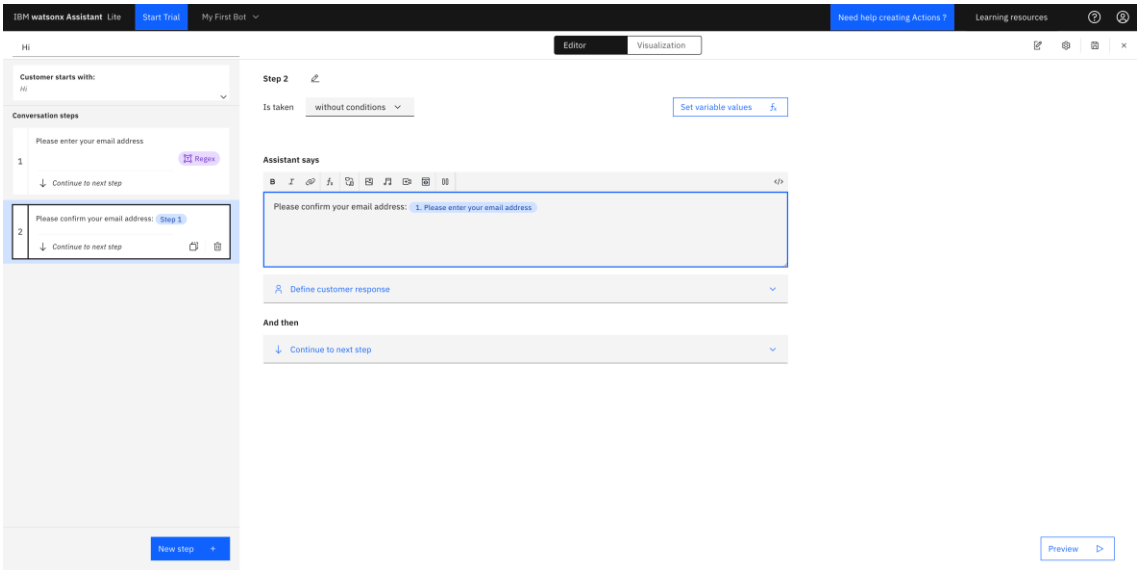
Select **Email** as the type of regular expression and click **apply**:



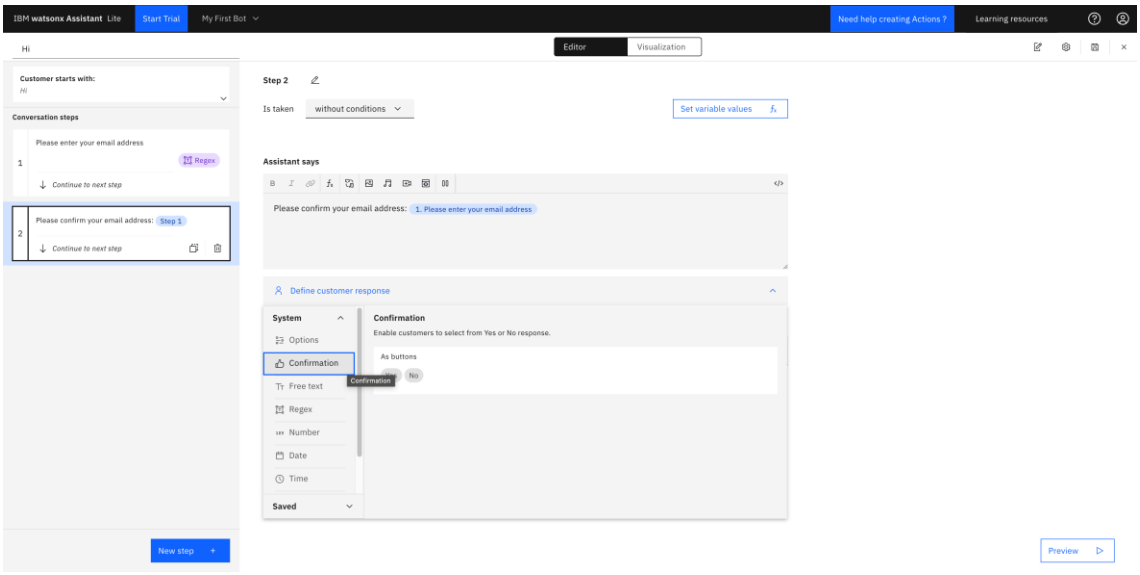
5. Create a new step by clicking the **New Step +** button in the bottom left corner. **Define** what the assistant says to confirm the email (e.g., *Please confirm your email address:*) and enter a **“$”** sign to open the variables context menu.



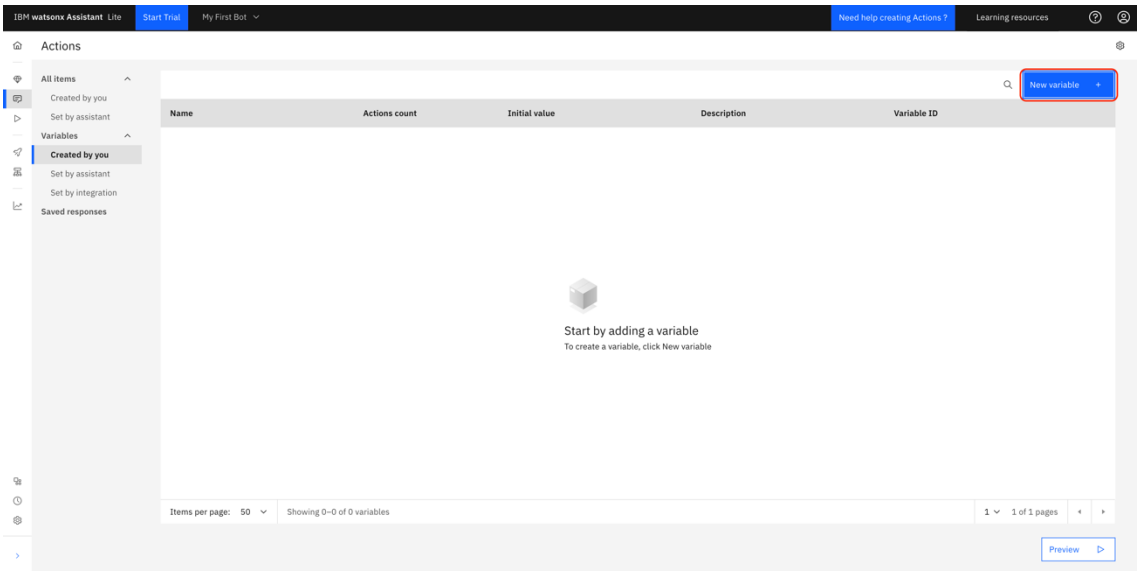
Select the **Action Step Variable** “1. Please enter your email address” from the previous conversation step. Through this configuration the assistant is now able to access the email address provided by the user in the previous step and display it in its response.



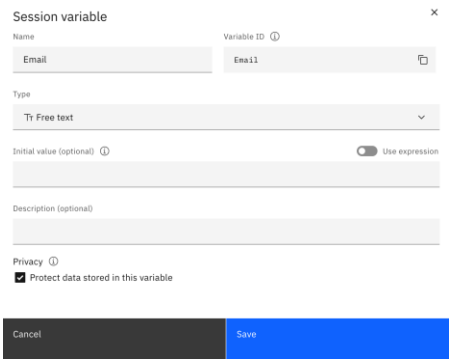
6. **Define** a customer response of type **Confirmation** to present yes and no buttons to the user.



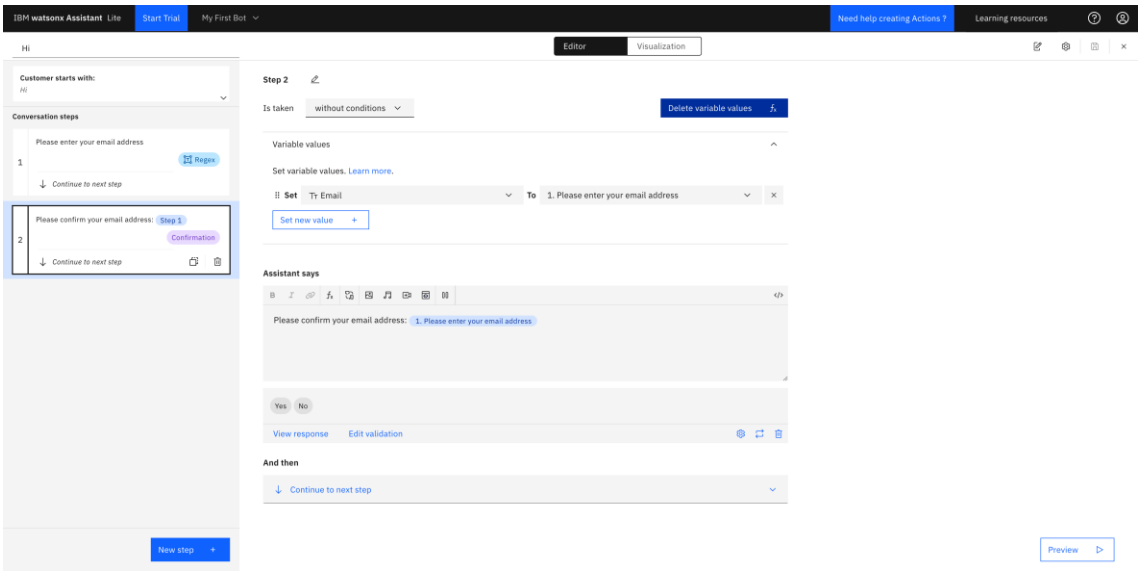
7. **To create a new Session Variable** navigate to Actions → Variables → Created by you and click on **New Variable +**.



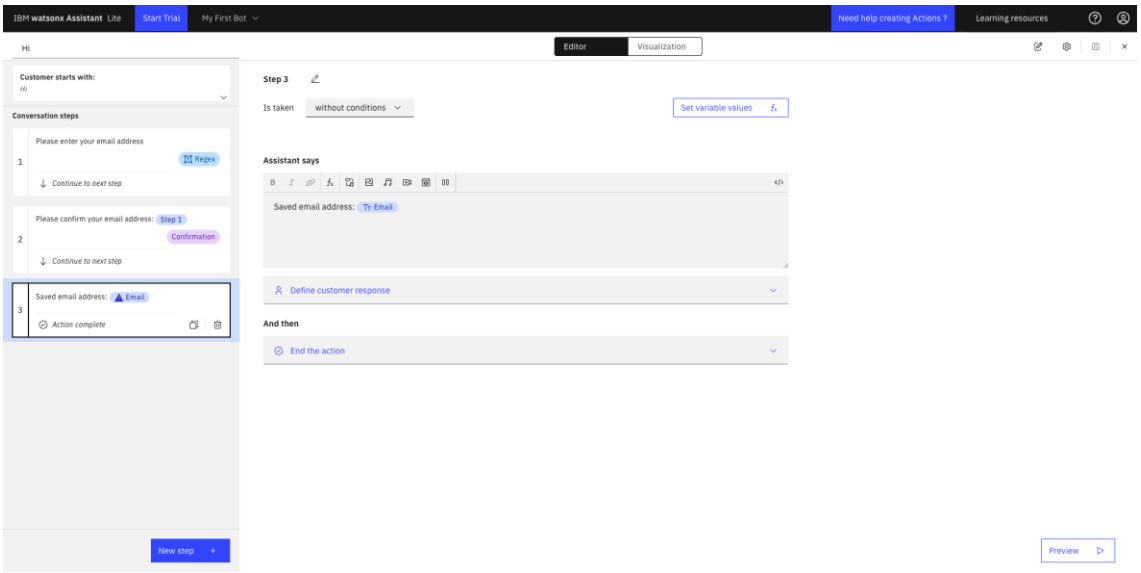
Fill in the following values and save the variable:



8. Head back to **the second step** of your action for this task. And click on the **Set variable values** button in the top right corner. Then click on **Set new value → Session variable → Email** and **Action Step Variables → 1. Please enter your email address**.



9. **Create a new step** via the "**New Step +**" button in the bottom left corner. Enter in the textfield below "Assistant says": Saved Email address ${Email}. This gives the step access to the session variable "**Email**". Session variables can be set in one action but can be used by all other actions, remember that it has to be set before it can be accessed. Be sure to end the action by setting **And then** to **End the action**.

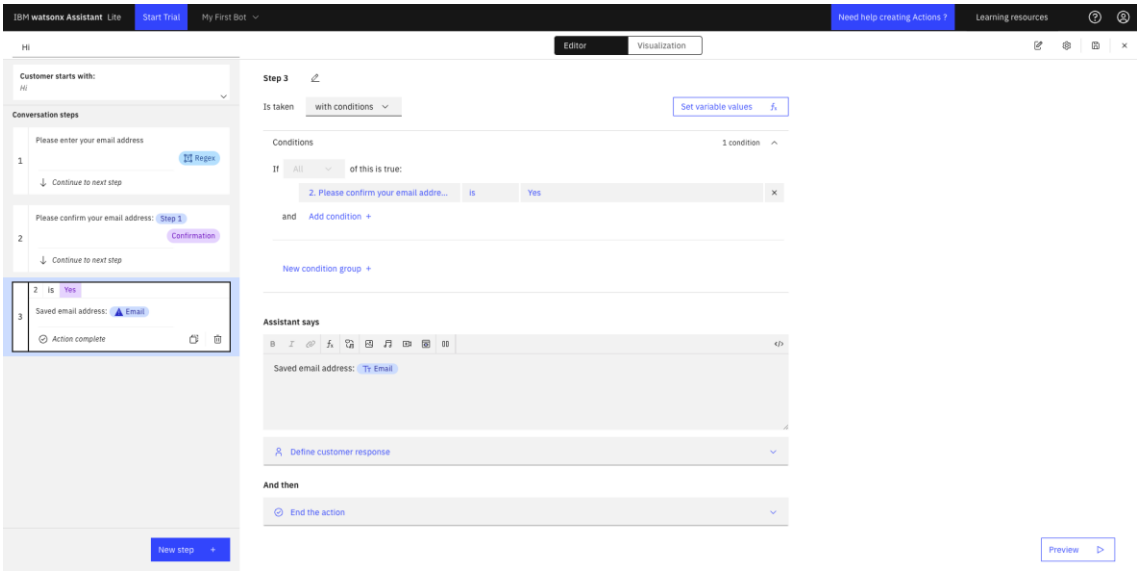


10. **Congratulations!** You have now explored the concept of Action Variables, that are usable in the current action, and Session Variables, that are usable in all actions. Preview your assistant and play around with it.

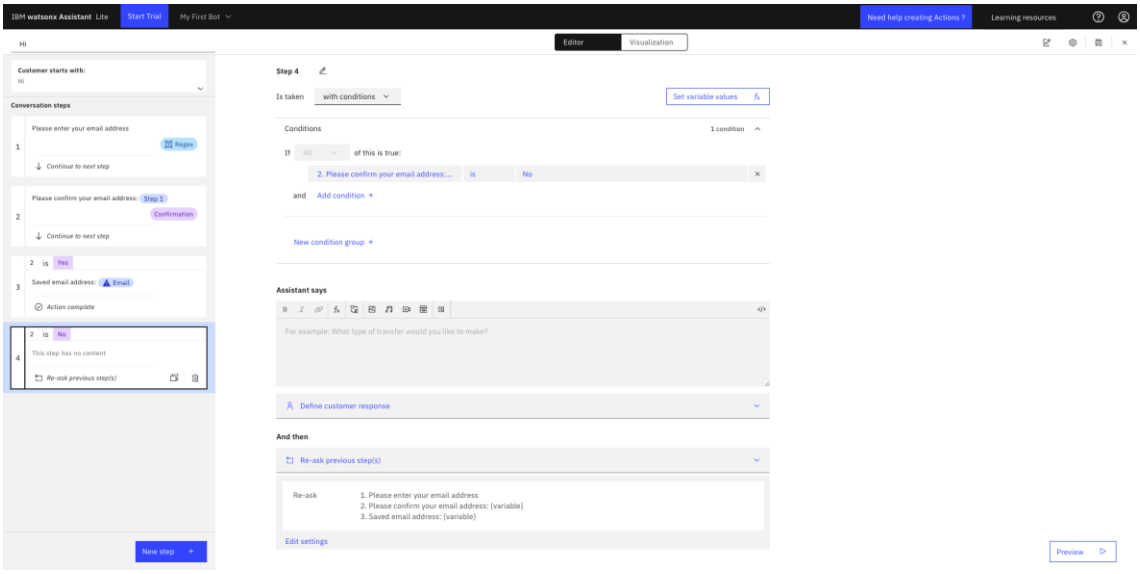
OPTIONAL

(As you may have noticed, we don't treat the user's responses (yes or no) separately. In this sub-task, we will therefore explore conditions for handling both responses appropriately).

11. The third step of the action is the end of the action, which should only be reached if the user agrees with the saved email. For this we set the **Is taken** variable to **with conditions**. Then we set the condition to **2. Please confirm your email address is Yes**:

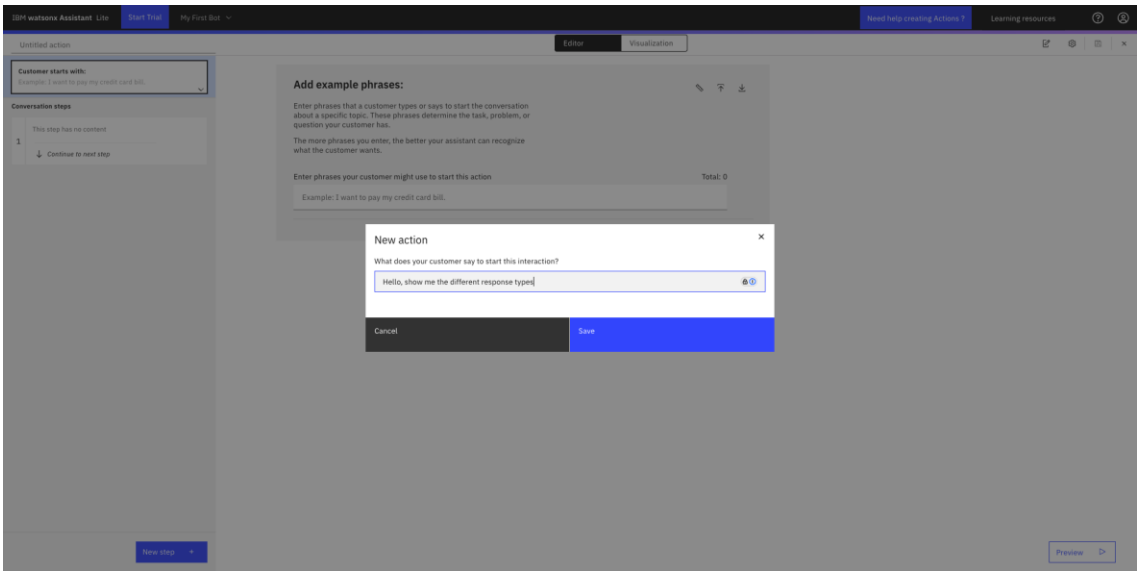


12. Next is the case where the user answers **No**, in which case we simply loop back and repeat the action from step 1. We add a new step and set the **Is taken** variable to **with conditions**. This time: **2. Please confirm your email address is No**. Then we set the **And then** variable to **Re-ask previous step(s)** and select all steps:

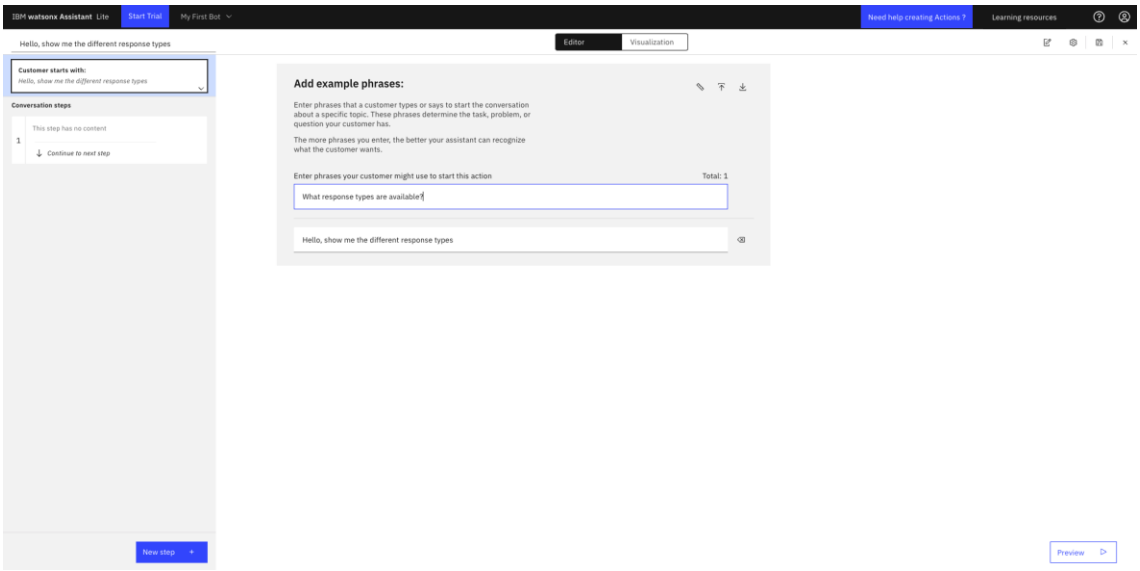


Task 5: Response Types

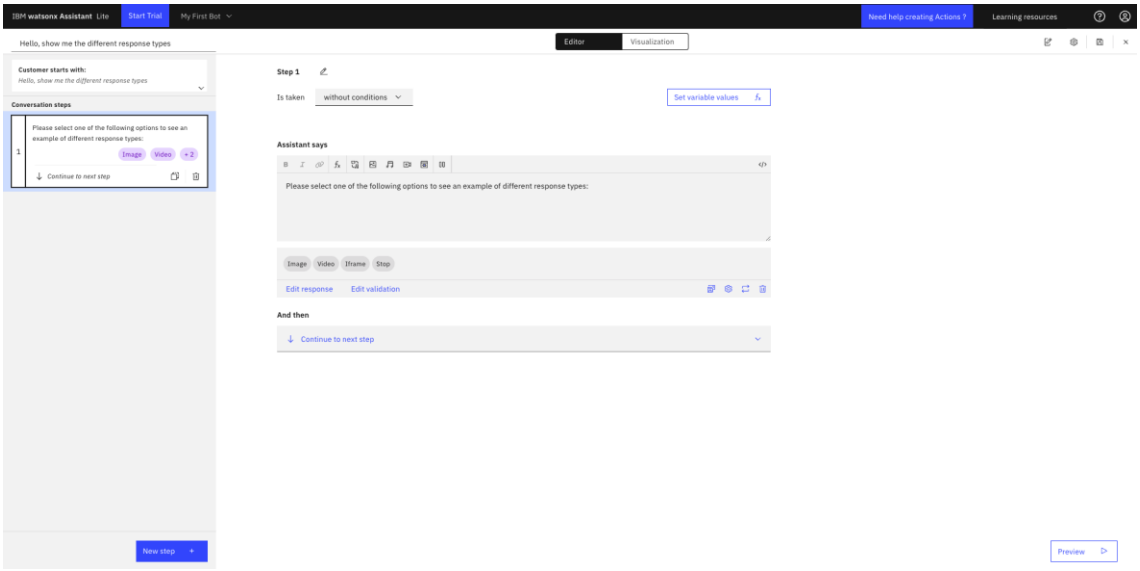
1. Create a new **action** (see Step 1, Task 3).
2. Choose **Start from scratch** (see Step 2, Task 4).
3. **Enter an example** of what your customer says to start this action (i.e *Hello, show me the different response types)*



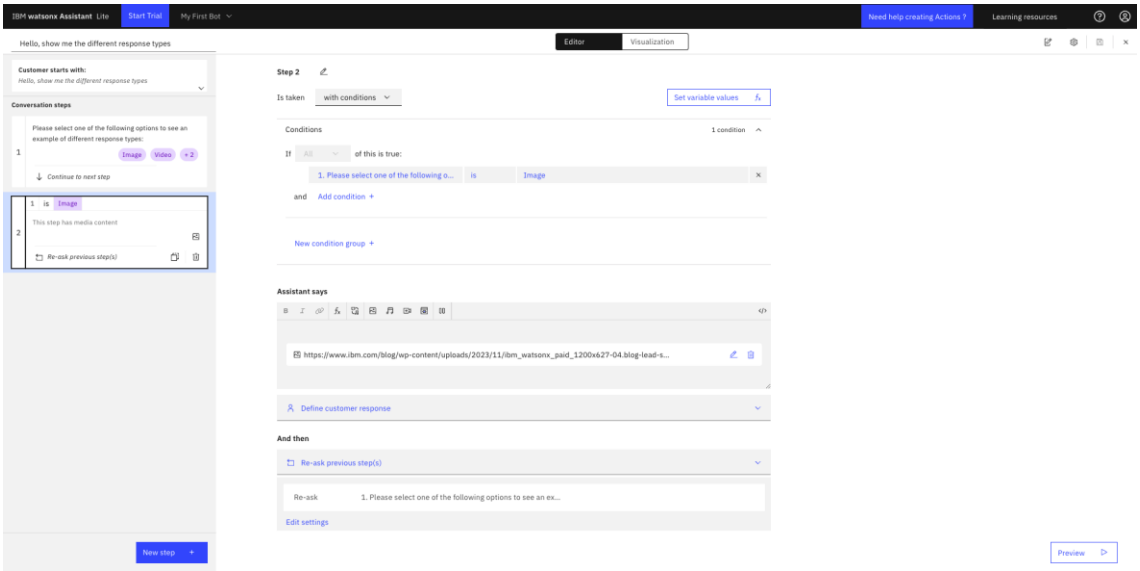
4. **Add more examples** (e.g., *What response types are available?*) by clicking on the element above Conversation steps and typing in the text box. This will help your watsonx assistant to better understand the customer's needs:



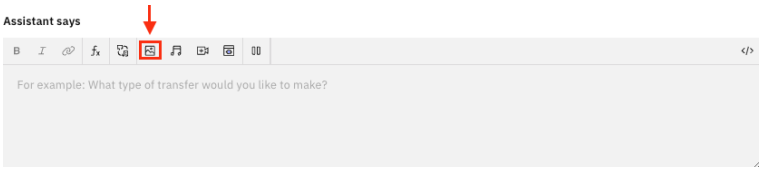
5. Now that watsonx assistant knows how to start this action, we need to define the steps that define the interaction between watsonx assistant and the customer. Navigate to the **first step**. Write your desired response in the textfield “Assistant says” (e.g., *Please select one of the following options to see an example of different response types:*). Then select **Define a customer response** of type **Options**, enter the following 4 options: **Image**, **Video**, **Iframe**, and **Stop**



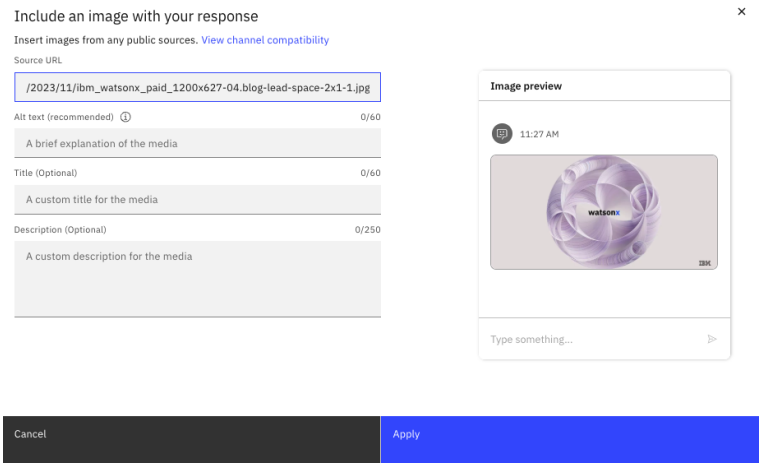
6. **Add a new step** and define a new condition by setting the **Is taken** variable to **with conditions**. Then set the condition to **1. Please select from the following options is Image**. This ensures that this step is only considered if the user has selected the **Image** option in the previous step.



Then we define the assistant's response. **Click on the Image button** in the menu on top of the textfield below Assistant says.



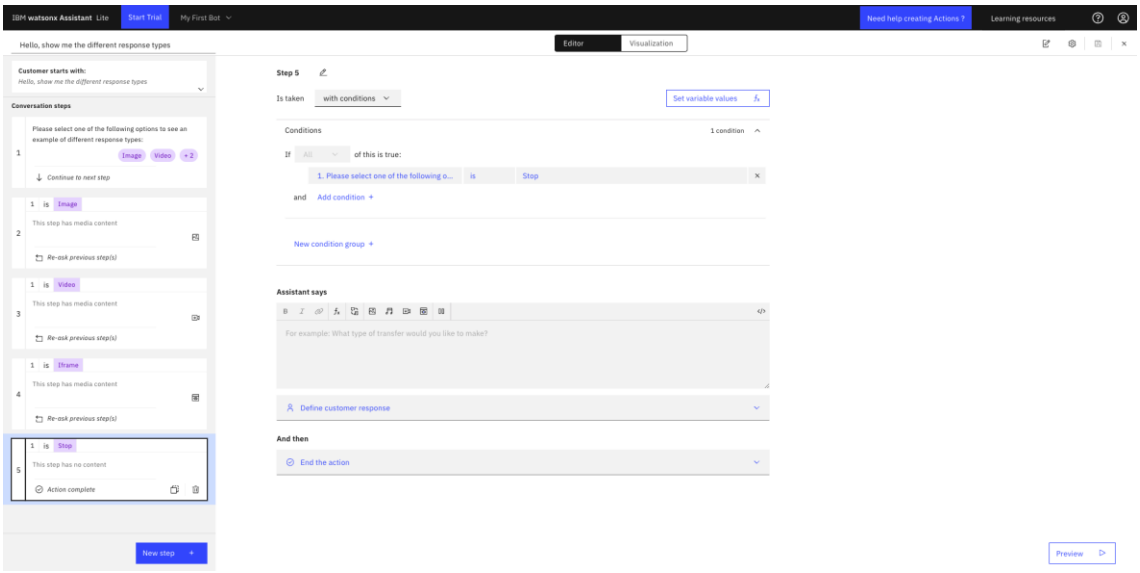
Search for an Image online to display and enter its URL into the Source URL field. After filling out all the fields click **Apply**:



To create a loop, we set the **And then** variable to **Re-ask previous step(s)** and select the first step (see Step 12, Task 4).

7. **Repeat Step 6** (or duplicate Step 2 and adjust it accordingly) for the response types: **Video** and **Iframe**. Remember to adjust the condition.
   (Note: use embedded links instead of just the URL from the browser so that your watsonx assistant can scale the frame correctly.)

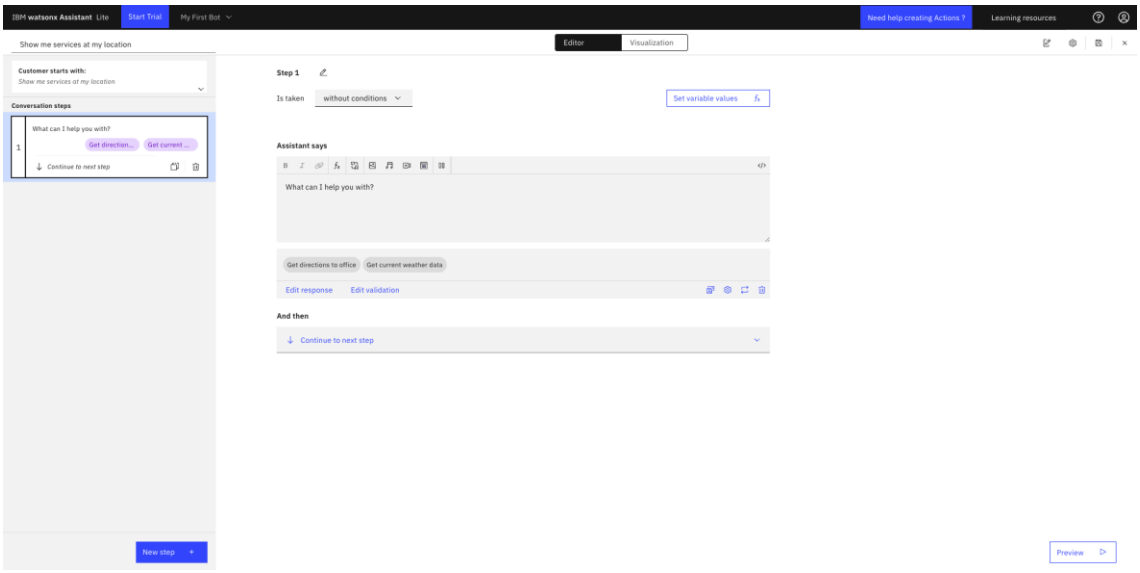
8. We have now created an infinite loop, so what's missing is the break condition. The action should stop as soon as the users chooses the option **Stop** in Step 1. For that, we create a **New step +** and we set the **And then** variable to **End the action**.



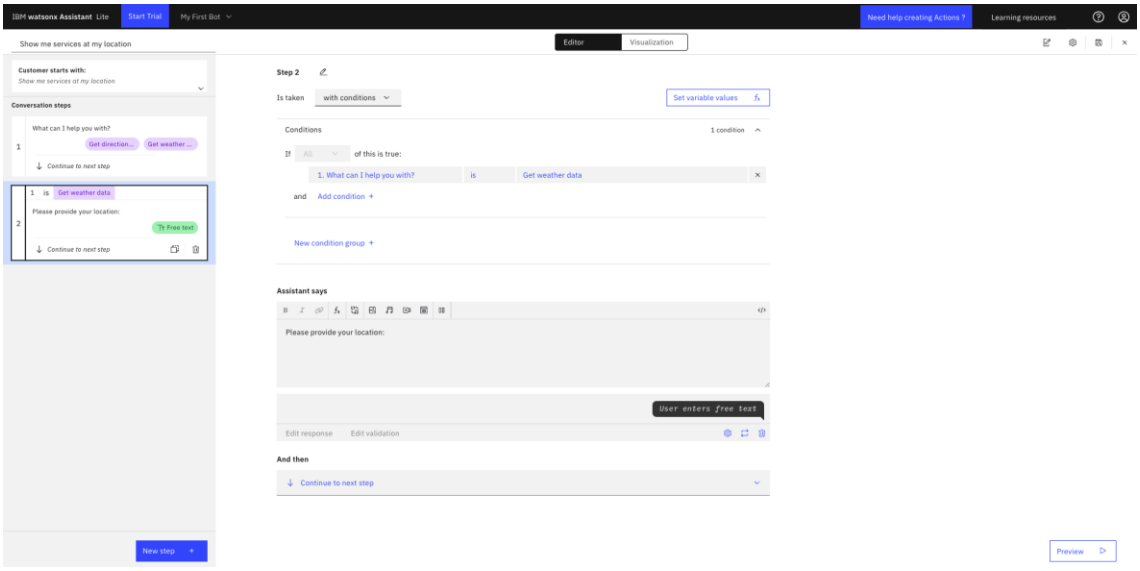
9. **Congratulations!** You have created different response types. Click **Preview** to see your assistant in action.

Task 6: Custom Extension / Dynamic Options

1. Create a new **action** (see Step 1, Task 3).
2. Choose **Start from scratch** (see Step 2, Task 4).
3. **Add an example phrase** (e.g., *Show me services at my location*), see Step 4, Task 5.
4. **Create a new Step** that asks the user *What can I help you with?* and provides two answer options to choose from (*Get directions to office*, *Get current weather data*):

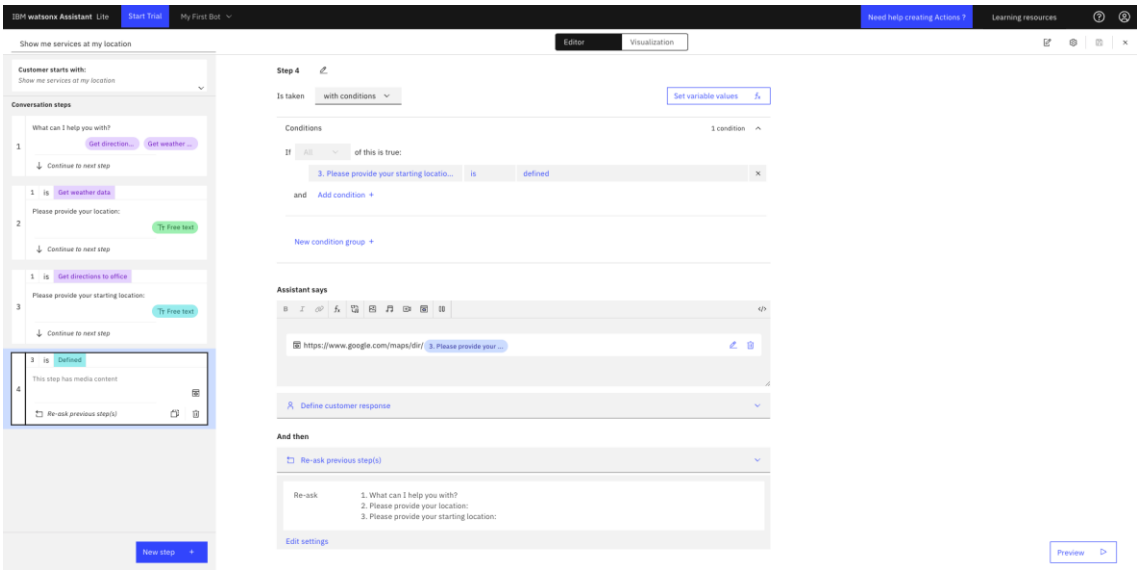


5. **Create a new Step** that asks the user *Please provide your location*: and requires the user to enter some free text. The step should only be triggered if in the previous step the option *Get weather data* has been selected. To do this select **with conditions** and set the condition to **1. What can I help you with is Get weather data**:

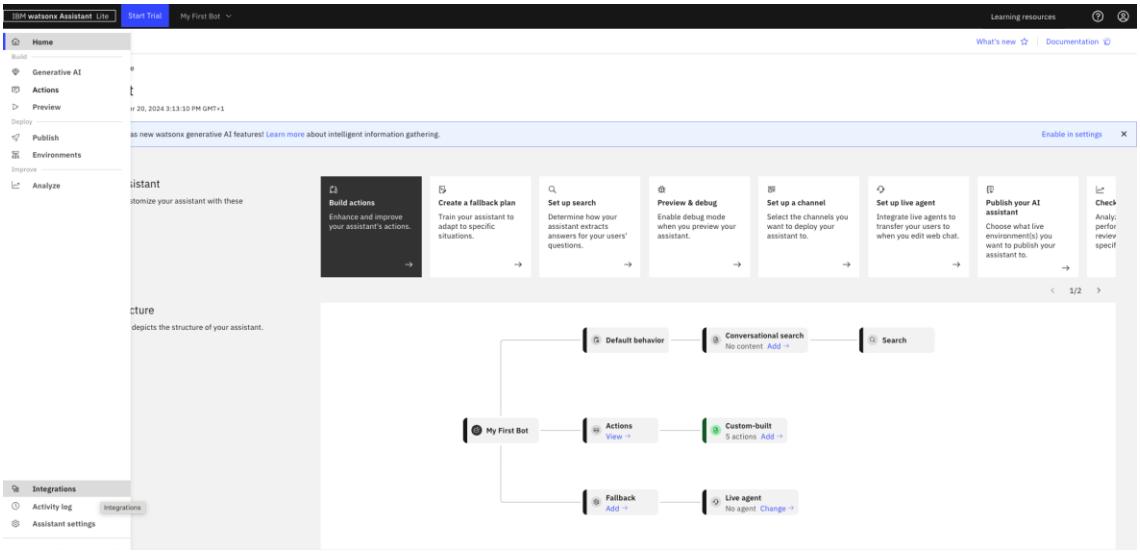


6. Repeat the previous step for the second option **Get way to office** (for that you can also duplicate Step 2 and adjust it accordingly).

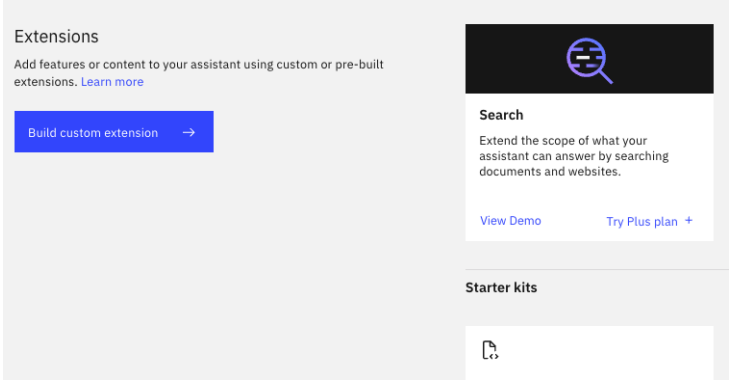
7. Create a **New step +** where you set a condition that checks if a starting location has been provided. As a response define a link to your favorite navigation service where you put in the variable set in Step 5 to be part of the URL:



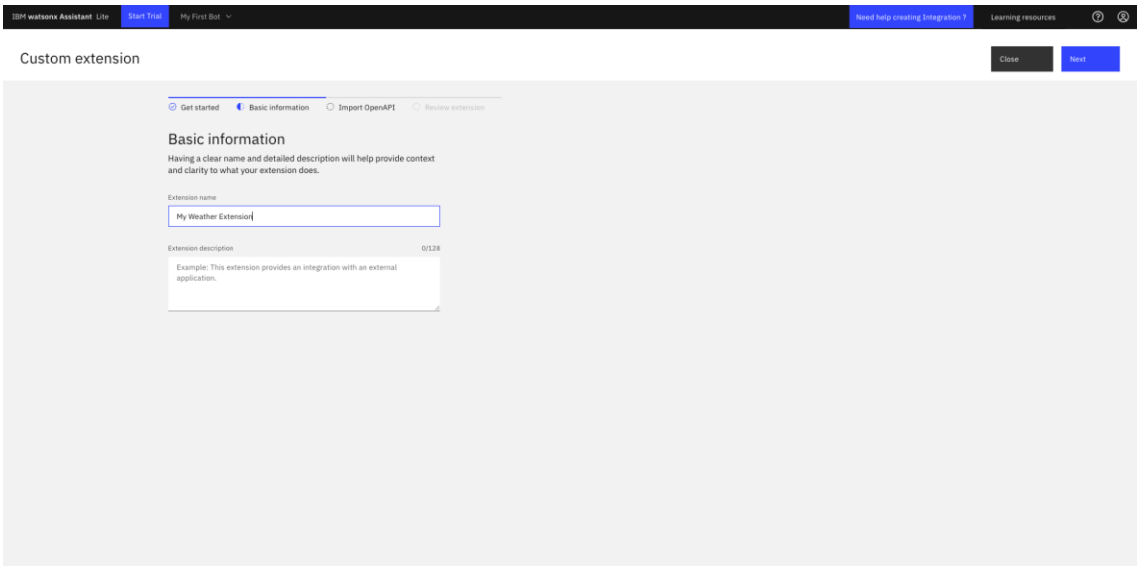
8. Head over to the Integrations Section

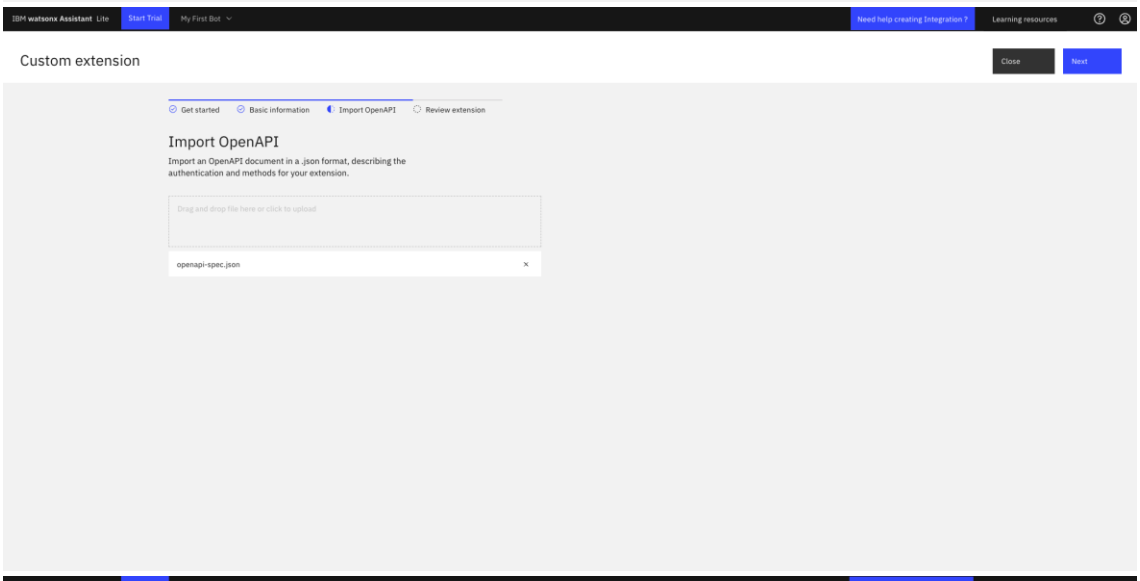


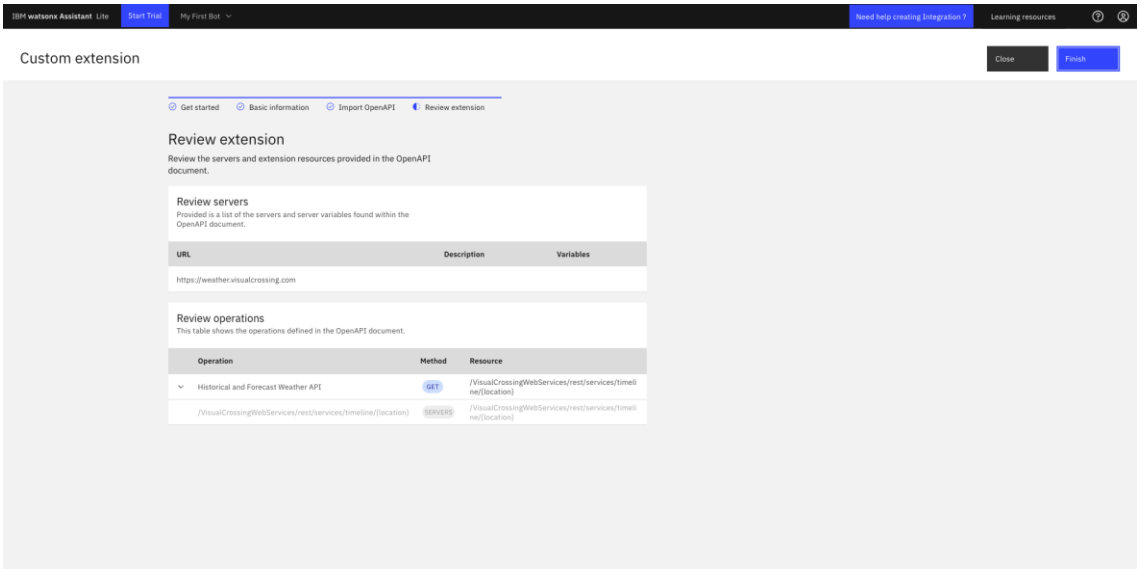
9. Scroll down to **Extensions** and select **Build custom extension**.



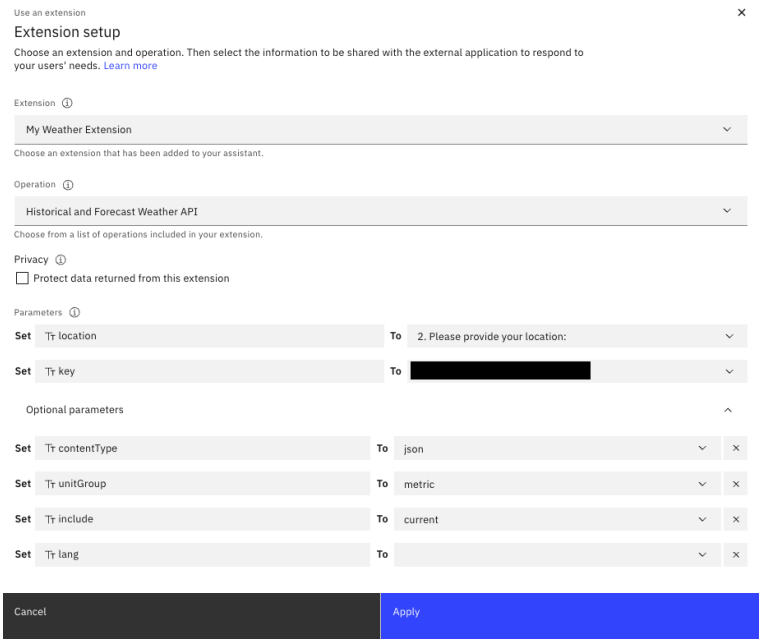
10. **Work through** the wizard and import the OpenAPI Spec from the **Appendix** as a .json file. The file is also available in the Box folder where you found this lab.


IBM watsonx Assistant Lite
Start Trial
My First Bot
Need help creating Integration ?
Learning resources
Custom extension
Close
Next
Get started
Basic information
Import OpenAPI
Review extension
Basic information
Having a clear name and detailed description will help provide context and clarity to what your extension does.
Extension name
My Weather Extension
Extension description
0/128
Example: This extension provides an integration with an external application.

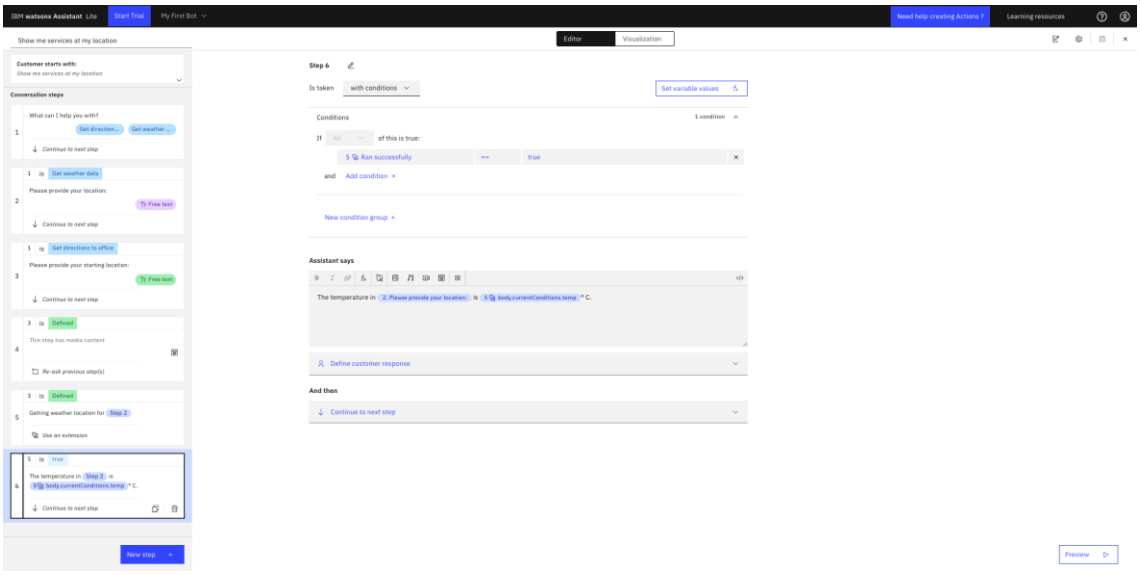

IBM watsonx Assistant Lite
Start Trial
My First Bot
Need help creating Integration ?
Learning resources
Custom extension
Close
Next
Get started
Basic information
Import OpenAPI
Review extension
Import OpenAPI
Import an OpenAPI document in a .json format, describing the authentication and methods for your extension.
Drag and drop file here or click to upload
openapi-spec.json


IBM watsonx Assistant Lite
Start Trial
My First Bot
Need help creating Integration ?
Learning resources
Custom extension
Close
Finish
Get started
Basic information
Import OpenAPI
Review extension
Review extension
Review the servers and extension resources provided in the OpenAPI document.
Review servers
Provided is a list of the servers and server variables found within the OpenAPI document.
URL
Description
Variables
https://weather.visualcrossing.com
Review operations
This table shows the operations defined in the OpenAPI document.
Operation
Method
Resource
Historical and Forecast Weather API
GET
/VisualCrossingWebServices/rest/services/timeline/{location}
/VisualCrossingWebServices/rest/services/timeline/{location}
SERVERS
/VisualCrossingWebServices/rest/services/timeline/{location}

11. **Navigate back** to your Action
12. **Create a new Step** that will call the previously defined custom extension. Define the extension as follows. If the API does not work, create a new Account on https://www.visualcrossing.com and create a new API Key under the profile settings



13. **Create a new step** that will reflect the answer gotten by the custom extension as follows:



14. **Congratulations** ! You have created a custom extension that speaks to a third party webservice. To see what this means preview your assistant and play around with it.

Task 7: Visualize an Action

1. Open the Action defined in Task 2 or in Task 6
2. In the top center section, click on **Visualization** to visualize your action.
3. **Congratulations**! You have now visualized your flow. Use this option to understand the big picture of the conversation and to discuss with your stakeholders.

Appendix

OpenAPI Spec:


```
{
  "openapi": "3.0.1",
  "info": {
    "description": "Weather Forecast and Historical Weather Data via
RESTful API.",
    "version": "4.6",
    "title": "Visual Crossing Weather API",
    "termsOfService": "https://www.visualcrossing.com/weather-services-
terms",
    "contact": {
      "email": "info@visualcrossing.com",
      "url": "https://www.visualcrossing.com/weather-api",
      "name": "Visual Crossing Corporation"
    },
    "license": {
      "url": "https://www.visualcrossing.com/weather-api",
      "name": "Visual Crossing Weather API"
    }
  },
  "servers": [
    {
      "url": "https://weather.visualcrossing.com"
    }
  ],
  "paths": {
    "/VisualCrossingWebServices/rest/services/timeline/{location}": {
      "get": {
        "description": "Seamless access to daily and hourly historical and
forecast weather data plus weather alerts, events and current
conditions.",
        "parameters": [
          {
            "name": "location",
            "in": "path",
            "schema": {
              "type": "string"
            },
            "required": true,
            "example": "London,UK",
            "description": "Locaton of interest as an address, partial
address or decimal latitude,longtude value"
          },
          {
```

```
            "name": "contentType",
            "in": "query",
            "schema": {
              "type": "string"
            },
            "example": "json",
            "description": "data format of the output either json or CSV"
          },
          {
            "name": "unitGroup",
            "in": "query",
            "schema": {
              "type": "string"
            },
            "example": "us"
          },
          {
            "name": "include",
            "in": "query",
            "schema": {
              "type": "string"
            },
            "example": "days",
            "description": "data to include in the output (required for
CSV format - days,hours,alerts,current,events )"
          },
          {
            "name": "lang",
            "in": "query",
            "schema": {
              "type": "string"
            },
            "example": "us",
            "description": "Language to use for weather descriptions"
          },
          {
            "name": "key",
            "in": "query",
            "schema": {
              "type": "string"
            },
            "required": true,
            "example": "INSERT_YOUR_KEY"
          }
        ],
        "responses": {
          "200": {
```

```
            "description": "Auto generated using Swagger Inspector",
            "content": {
              "application/json":{
                "schema": {
                  "type": "object",
                  "properties": {
                    "currentConditions": {
                      "type": "object",
                      "properties": {
                        "temp": {
                          "type": "number"
                        }
                      }
                    }
                  }
                }
              }
            }
          },
          "servers": [
            {
              "url": "https://weather.visualcrossing.com"
            }
          ],
          "summary": "Historical and Forecast Weather API",
          "tags": [
            "Timeline Weather API (15-day forecast request)"
          ]
        },
        "servers": [
          {
            "url": "https://weather.visualcrossing.com"
          }
        ]
      }
    },
    "externalDocs": {
      "url":
"https://www.visualcrossing.com/resources/documentation/weather-
api/timeline-weather-api/",
      "description": "Full Timeline Weather API Documentation"
    }
}
```