Lab 4: Adding Discovery to the Chatbot: Part-2

Objective:

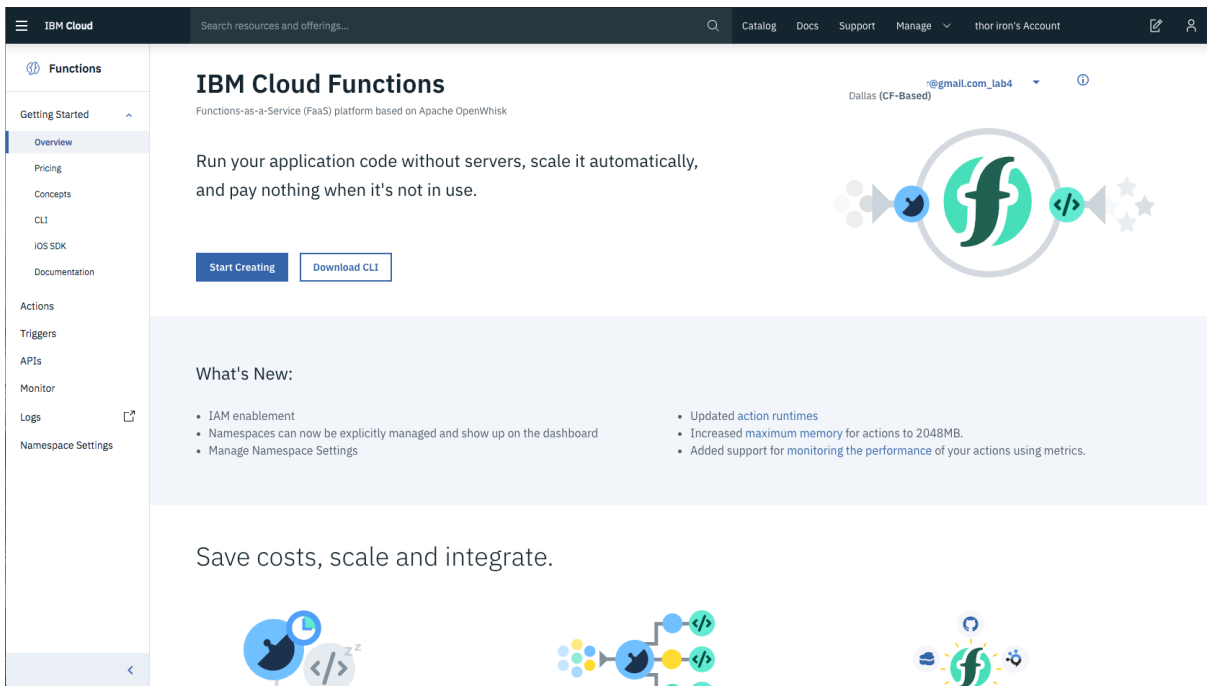
- How to use IBM Cloud Functions to make programmatic calls from dialog node.
- Learn about IBM functions.

Pre-requisite:

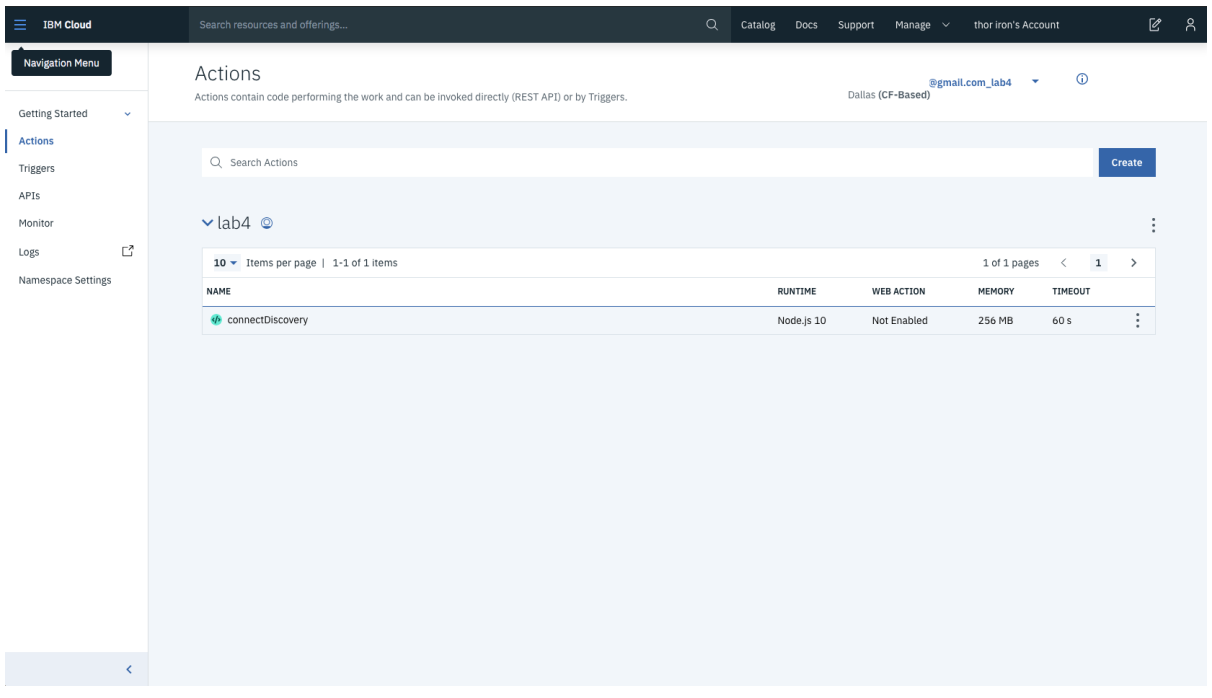
You should have completed the previous lab-**Adding Discovery to the Chatbot:Part-1** succsesfully. If not, then please consider doing the previous lab first before moving further with this lab.

In the previous lab you defined an action. At the end of this lab, your chatbot will be able to call a cloud Function to receive a response from Watson Discovery and present course recommendations to the user.

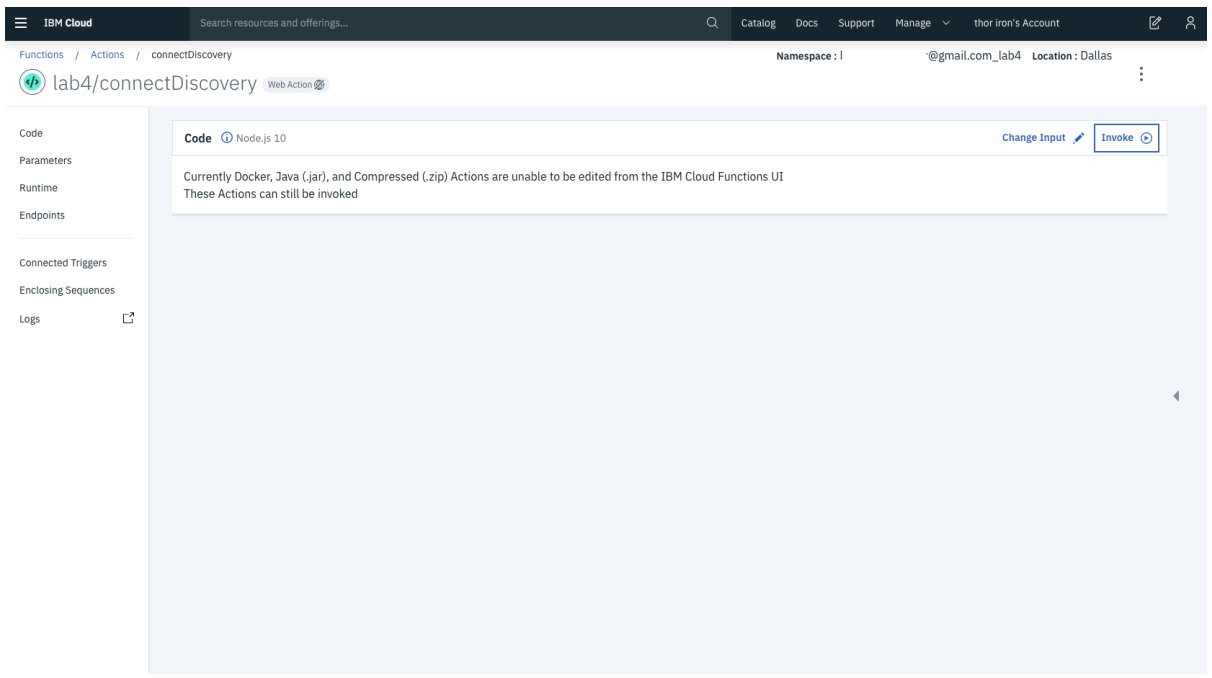
1. Log in to IBM Cloud and then search for **Functions** in the search bar. Select Functions. After the page is loaded, **make sure the top right hand side shows the name of your organization_lab4**. If your organization is abcd@gmail.com, then you should see the following, <abcd@gmail.com_lab4>. Click **Actions**.



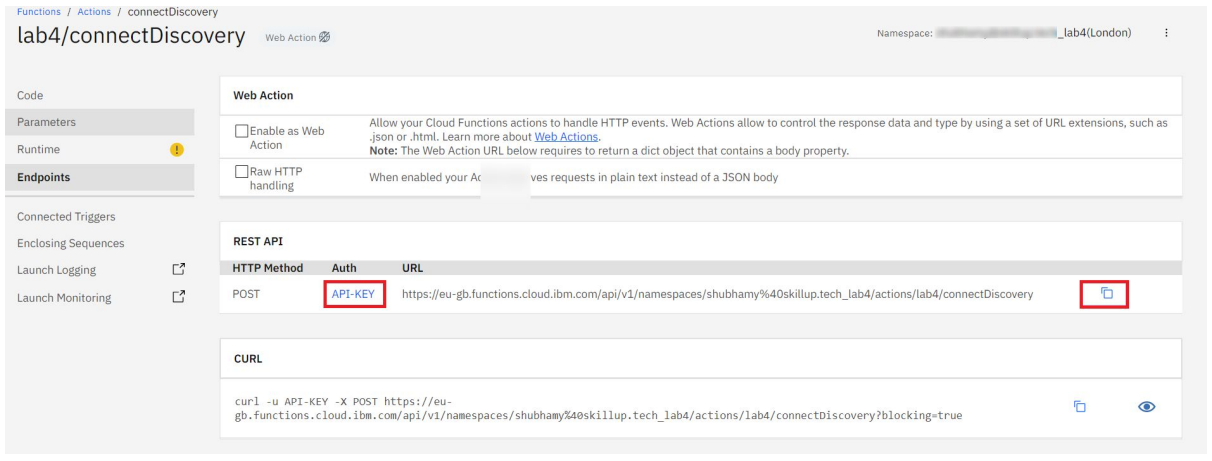
2. Click connectDiscovery action under lab4 package.



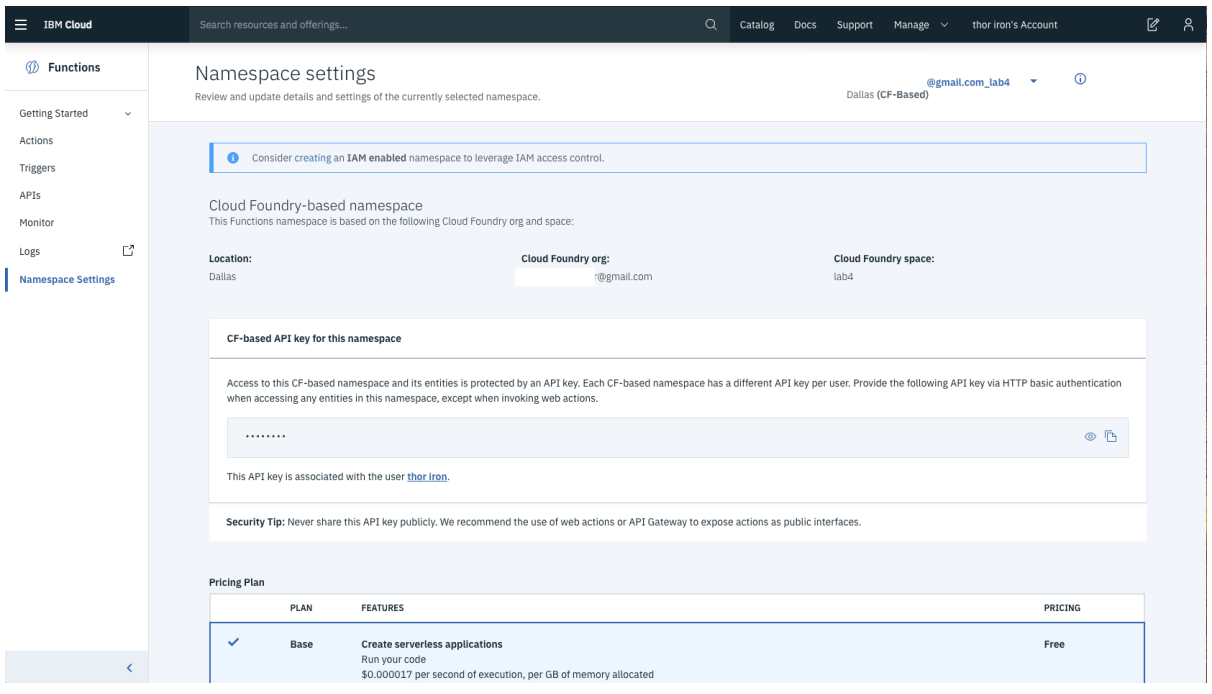
3. Click **Endpoints** on the left sidebar.



4. Click **copy Icon** to copy the URL for the action and make a copy of the REST API URL. Next, click **API-KEY** under REST API.



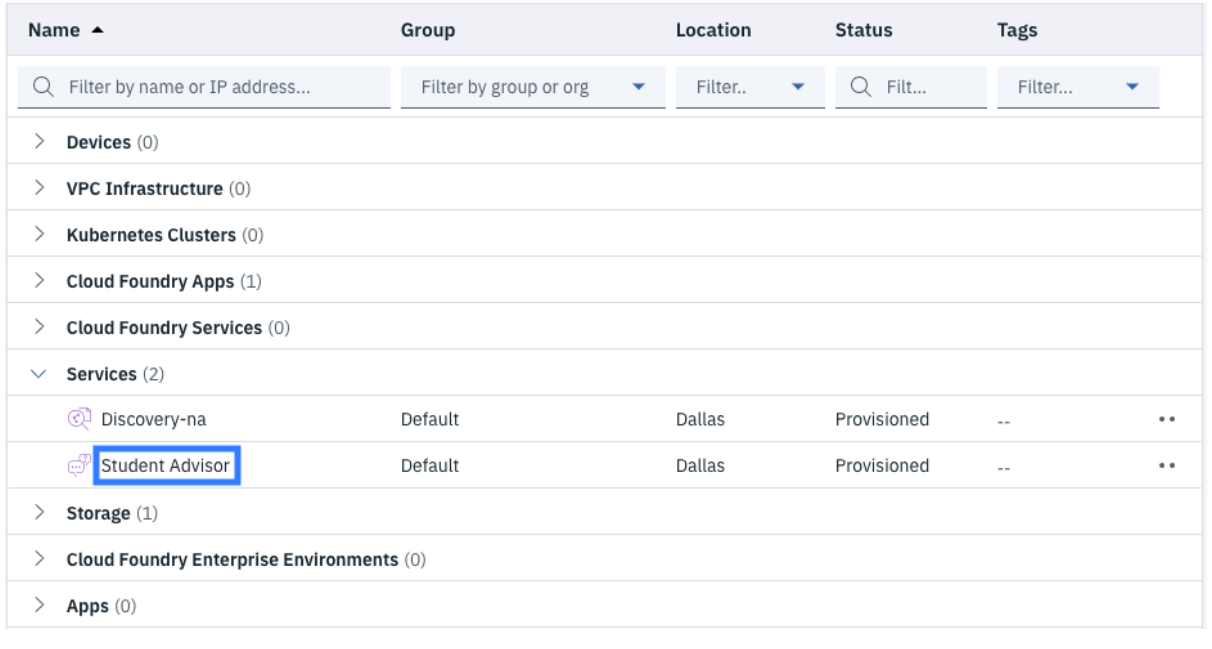
5. **Click on the eye icon** to obtain your **CF-based API key** and make a copy of the credentials.



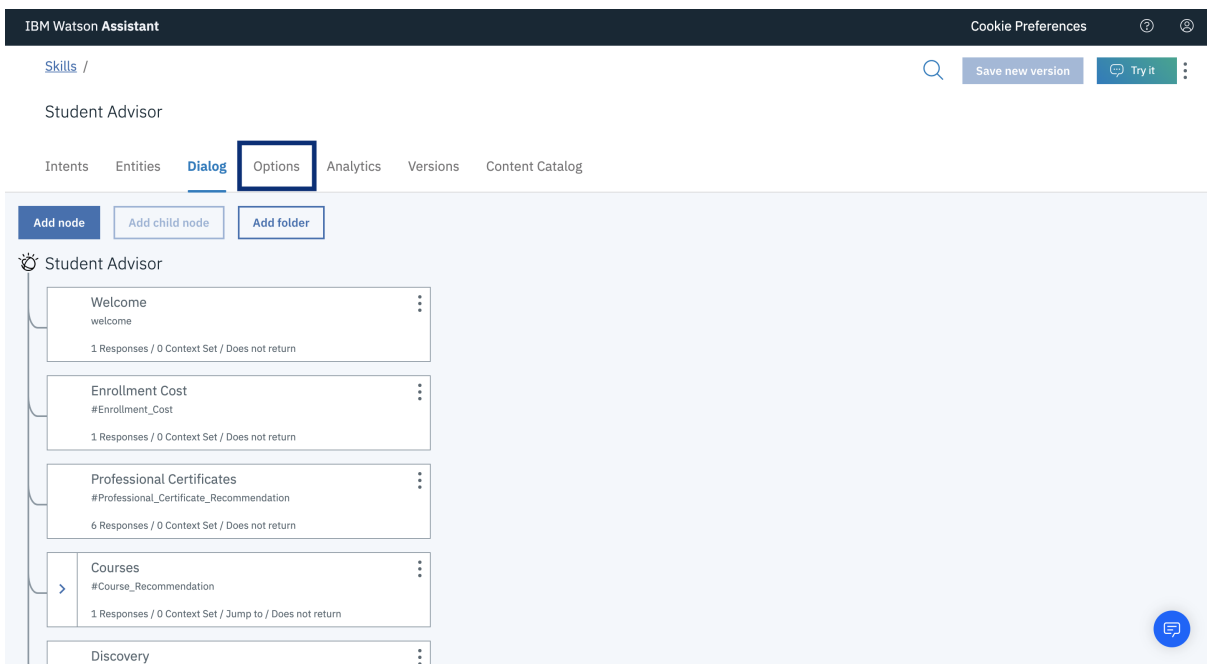
You're now all set to begin integrating this function with Watson Assistant.

Integrate Discovery with our Chatbot

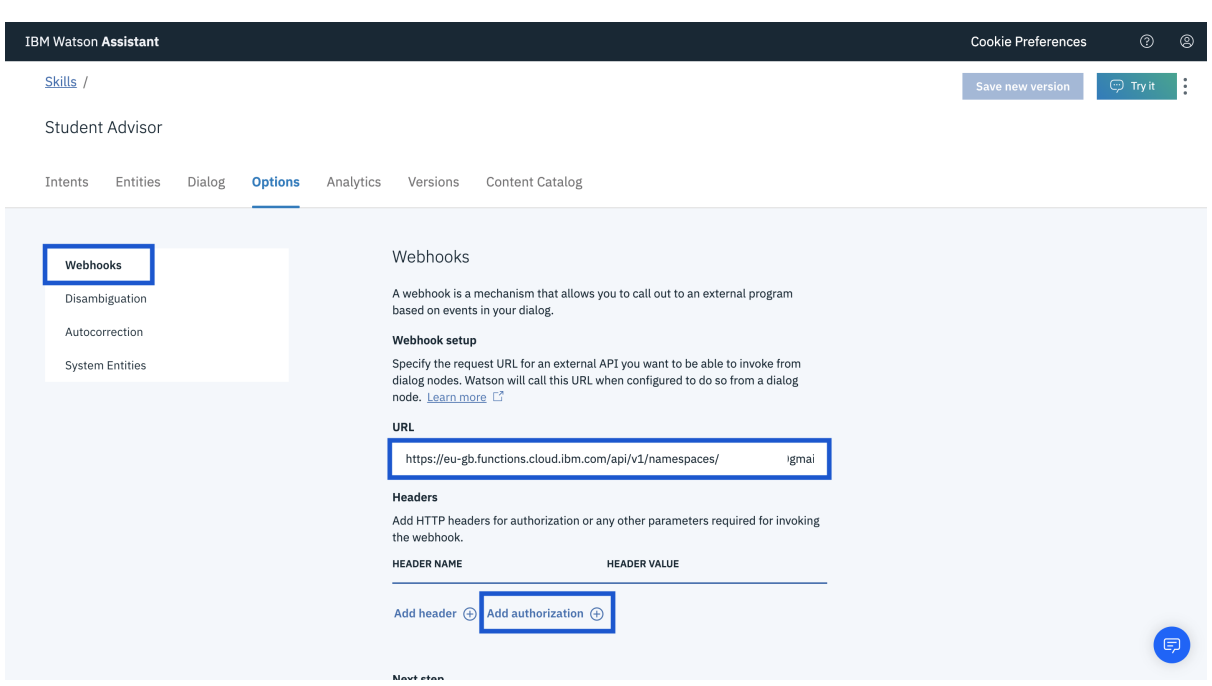
1. From your Dashboard, click on the **Watson Assistant service** you created in Lab 3. (Click on the name, not the icon next to it.) and then click **Launch Watson Assistant**.



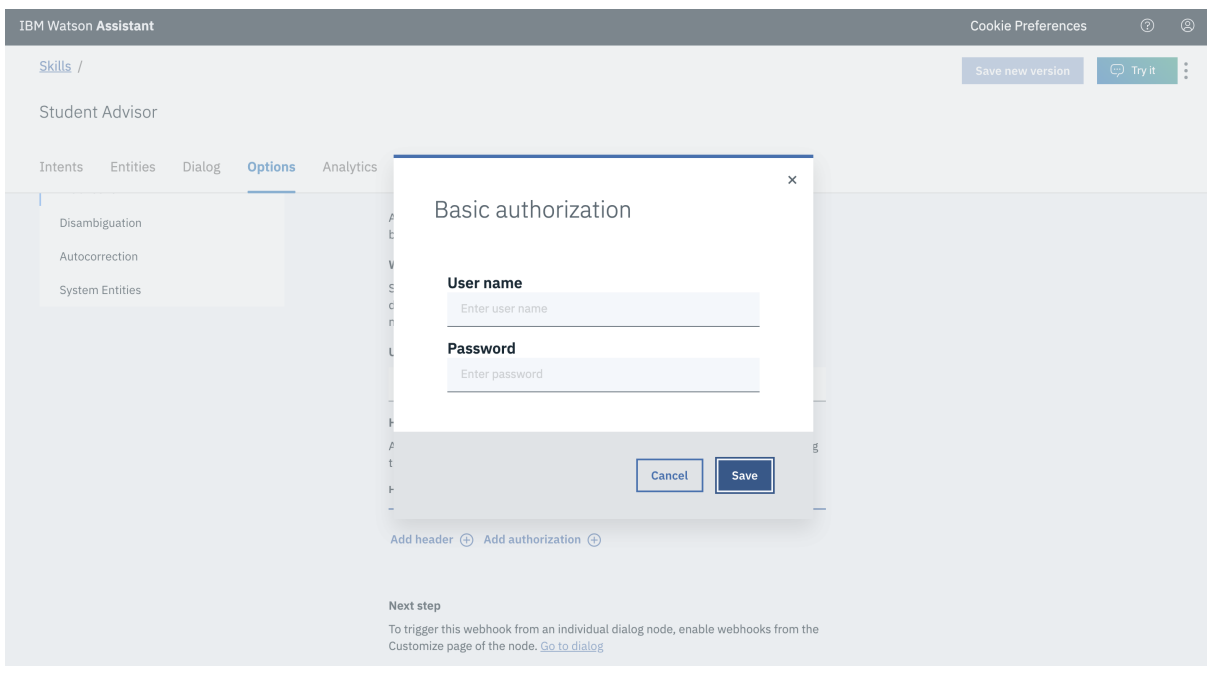
2. Under **Skills** tab, click **Student Advisor** skill. Then, click the **Options** tab.



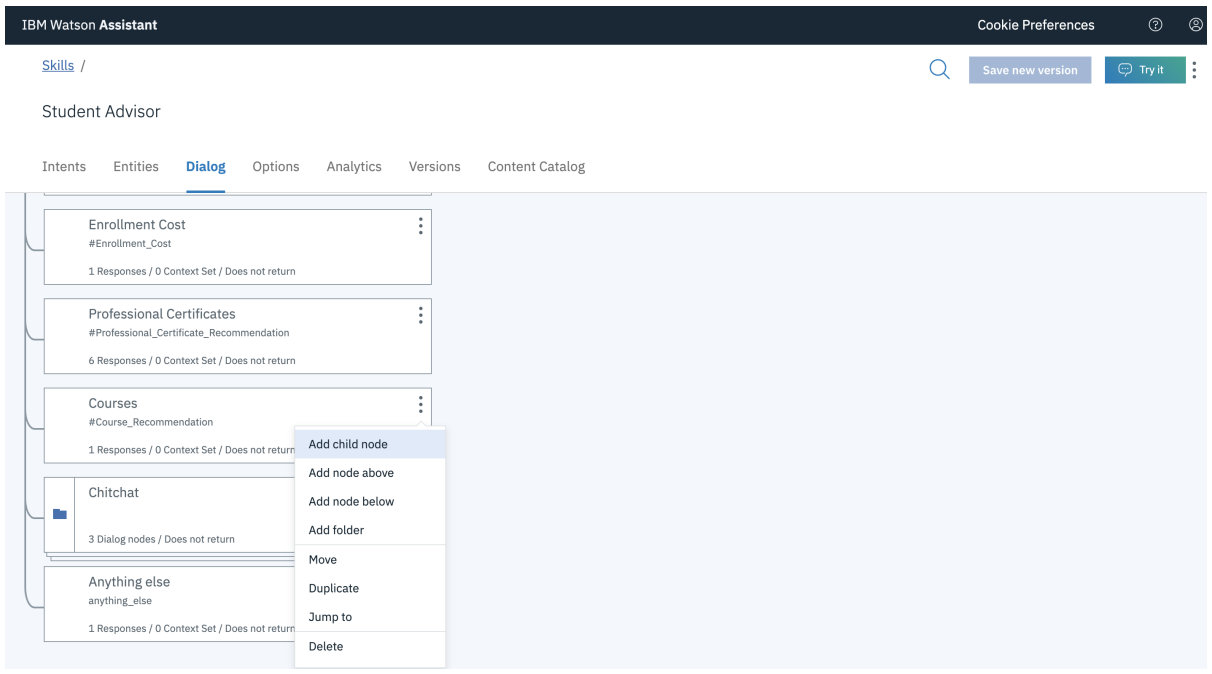
3. Click **Webhooks**. In the **URL** field, specify the **REST API URL** for the action. Append a **?blocking=true** parameter to the action URL to force a synchronous call to be made (e.g. https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/YOUR-ORG_YOUR-SPACE/actions/lab4/connectDiscovery?blocking=true). Finally, click **Add authorization**.



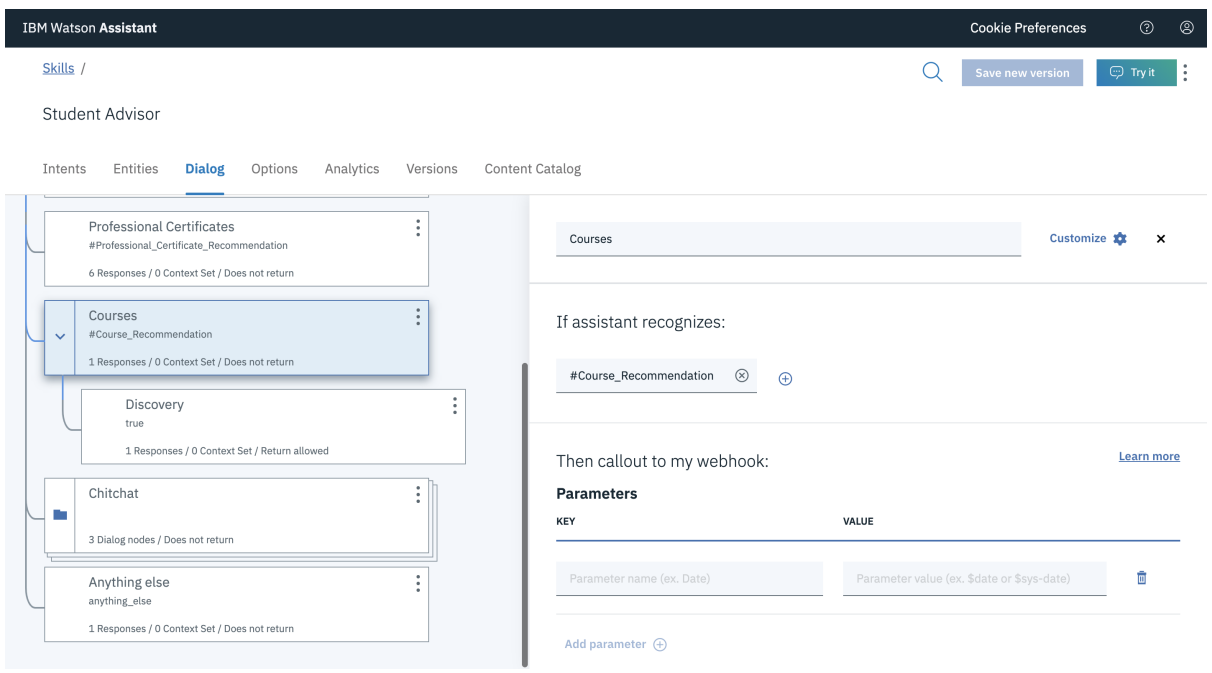
4. We'll use the **CF-based API key** you obtained from action endpoints. The part before the **colon(:)** is your **User name**. Copy this to the User name field. Similarly, the part after the colon(:) is your **password**. (e.g. Your action API key = User name:password). Finally **save**.



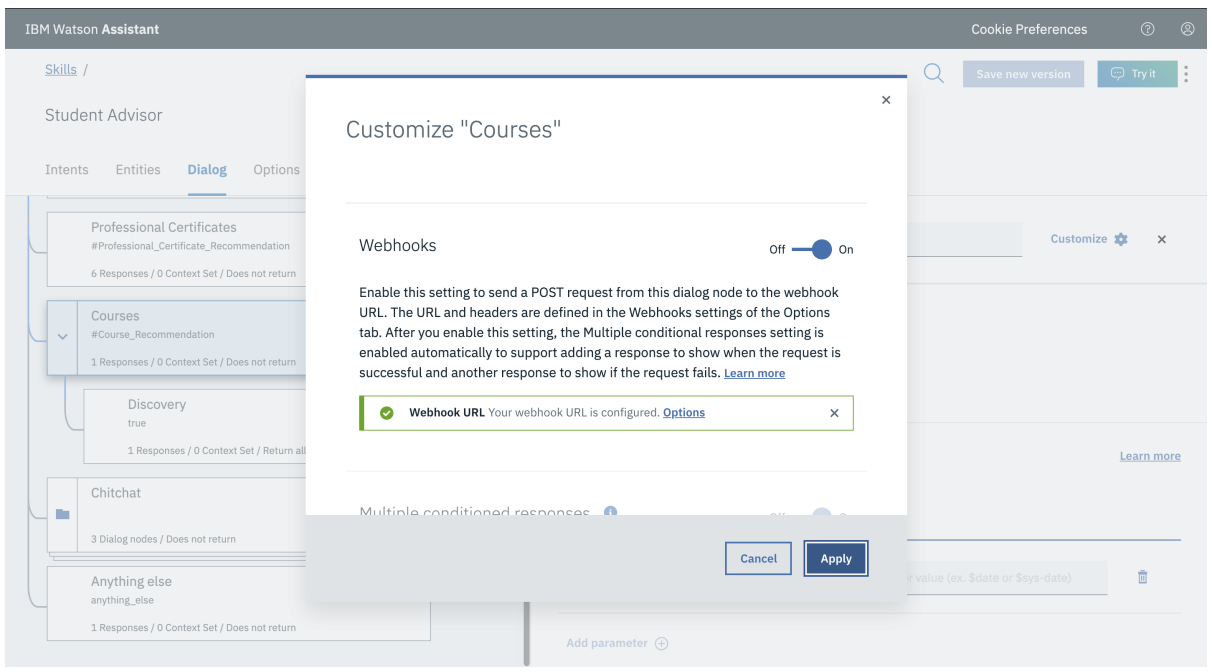
5. Find **Courses** node under the **Dialog** tab. Click the **3 dots icon** in the Courses node and click **Add child node**. Then, name the new node (e.g., Discovery).



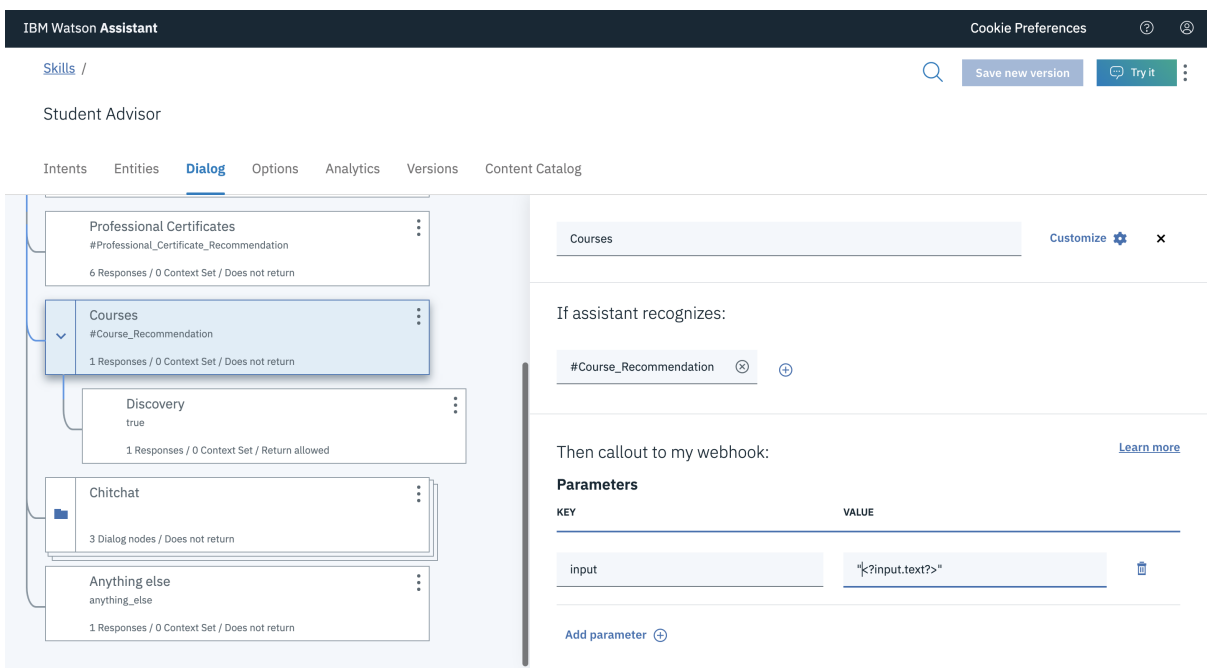
6. Click the **Courses node** and then click **Customize** to customize the node.



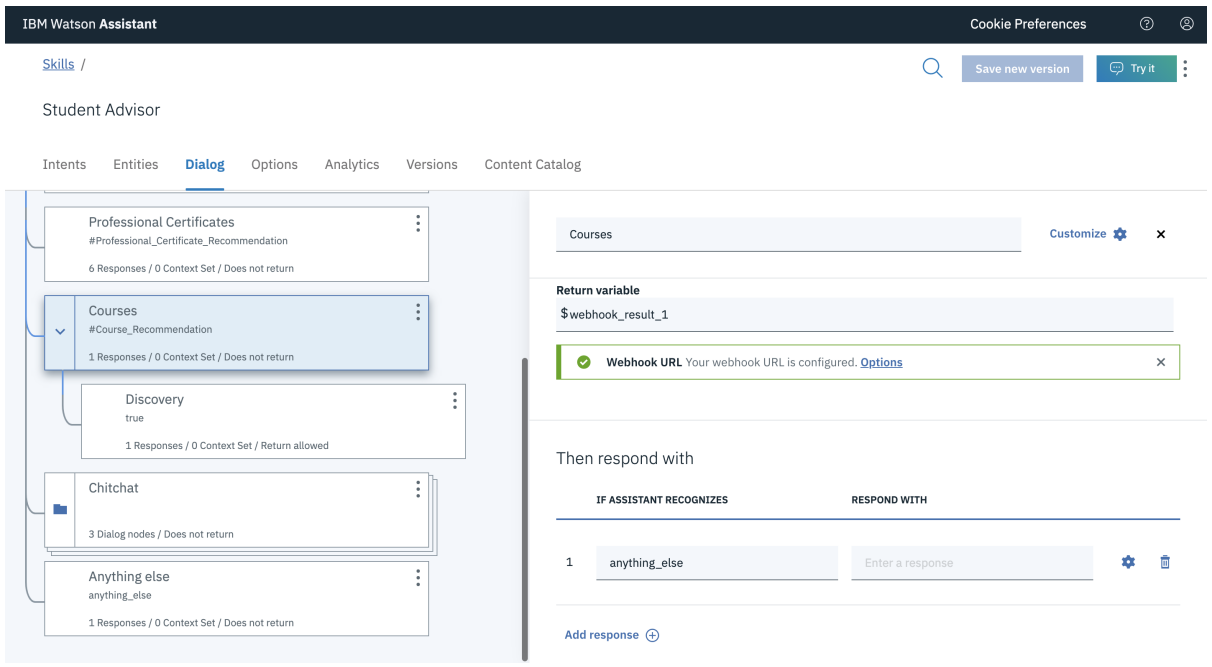
7. Scroll down to the ***Webhooks*** section, and switch the toggle to **On**, and then click **Apply**.



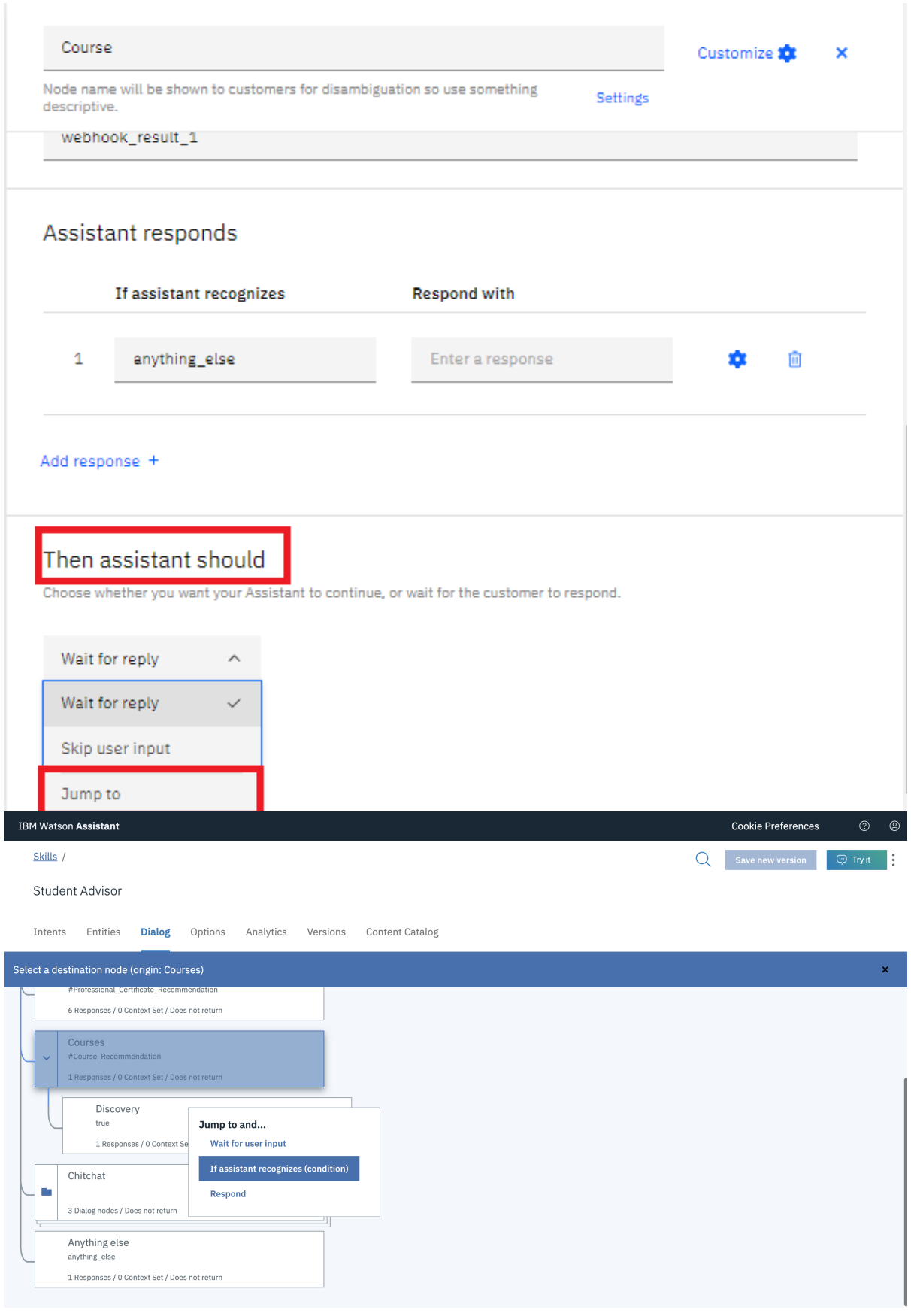
8. Now, we will add a parameter, input as a key and `"<?input.text?>"` as a value.



9. Scroll down and set the Return variable to **webhook_result_1**.

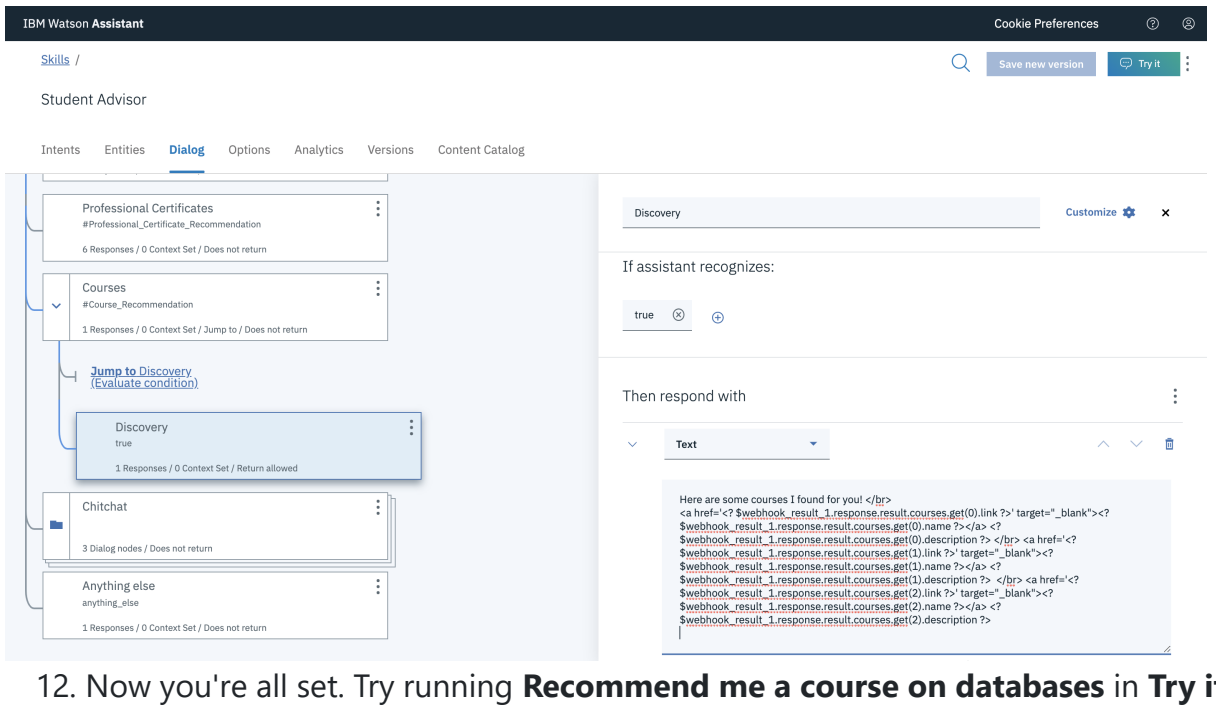


10. Scroll down and under And **Then assistant should**, select **Jump to option** and select the child node you created and then select If **assistant recognizes**(condition).



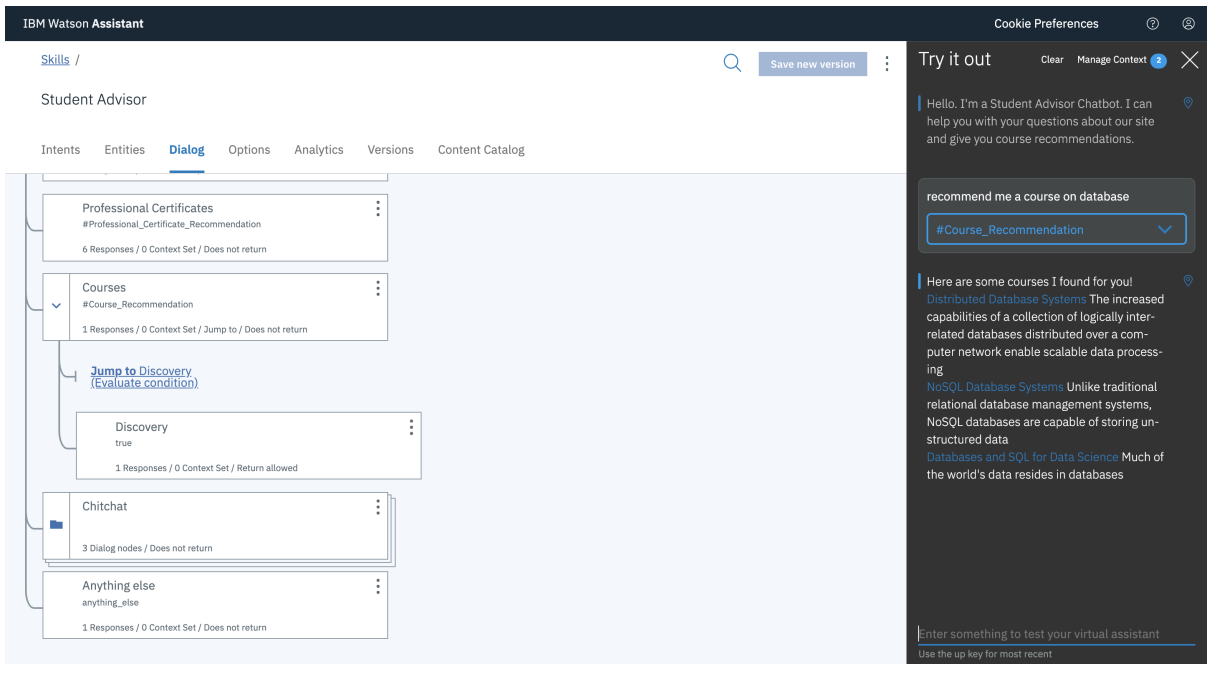
11. Put **true** as a condition for the node, as we always want to execute it when giving course recommendations. Add the following sample response, which retrieves a list of relevant courses from Discovery via the Function we defined earlier on.

```
Here are some courses I found for you! </br>
<a href='<? $webhook_result_1.response.result.courses.get(0).link ?>' target="_blank"><?
$webhook_result_1.response.result.courses.get(0).name ?></a> <? $webhook_result_1.response.result.courses.get(0).description ?>
</br> <a href='<? $webhook_result_1.response.result.courses.get(1).link ?>' target="_blank"><?
$webhook_result_1.response.result.courses.get(1).name ?></a> <? $webhook_result_1.response.result.courses.get(1).description ?>
</br> <a href='<? $webhook_result_1.response.result.courses.get(2).link ?>' target="_blank"><?
$webhook_result_1.response.result.courses.get(2).name ?></a> <? $webhook_result_1.response.result.courses.get(2).description ?>
```



12. Now you're all set. Try running **Recommend me a course on databases** in **Try it out panel** to confirm that the node works as expected. If you see a result similar to the image below, you are all set for this lab.

To recap, we created a Cloud Function that connects to our Discovery collection and retrieves documents relevant to the user query. We then invoke that function from the relevant node, when people express the intent of receiving a course recommendation. The result is then displayed to the user in our response.



In short, we managed to dynamically invoke results rather than hardcoding the responses. This is a very powerful and flexible approach. You'll use it again when working on the capstone project.

Author(s)

Antonio Cangiano

Changelog

Date	Version	Changed by	Change Description
2020-09-16	2.0	Shubham	Migrated Lab to Markdown and added to course repo in GitLab
2020-09-16	2.1	Anamika	Updated Instructions