Lab 3: Creating a Chatbot (1h)

Objective for Exercise:

- How to create a Simple Student Advisor Chatbot.
- How to add various intents and entities.

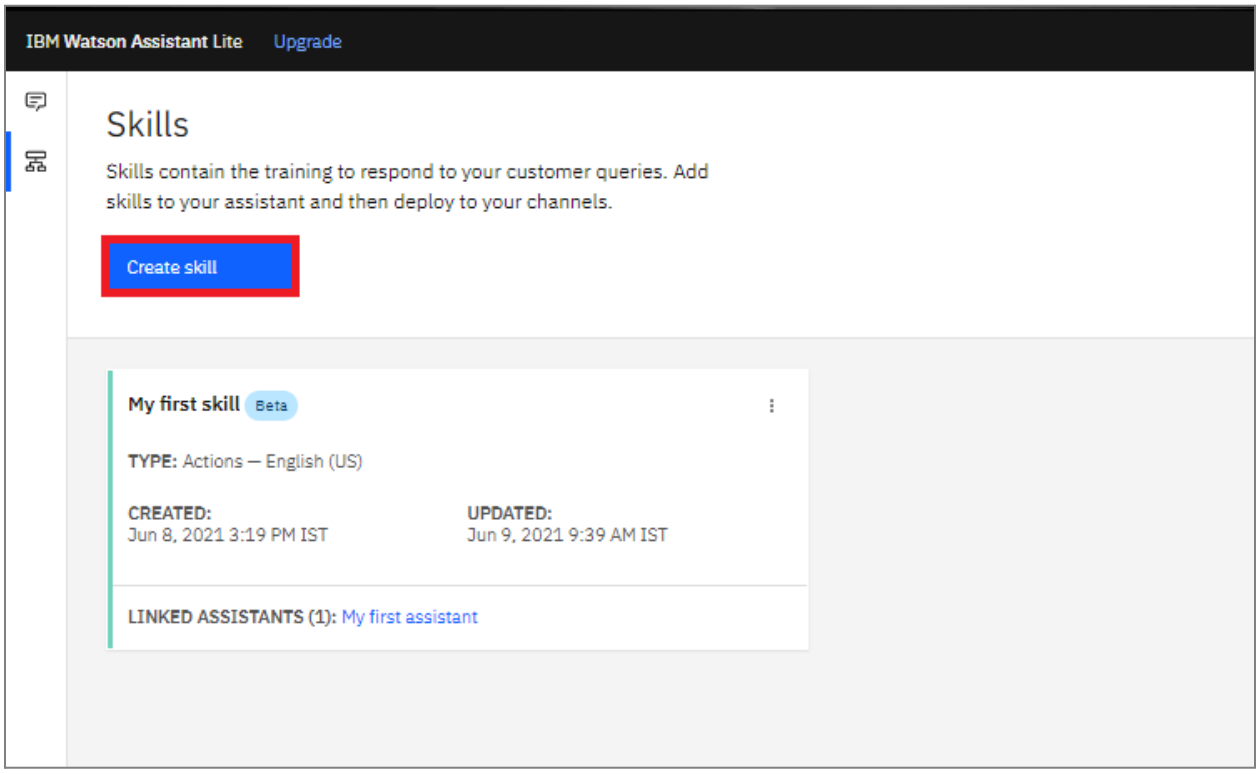
Pre-requisite:

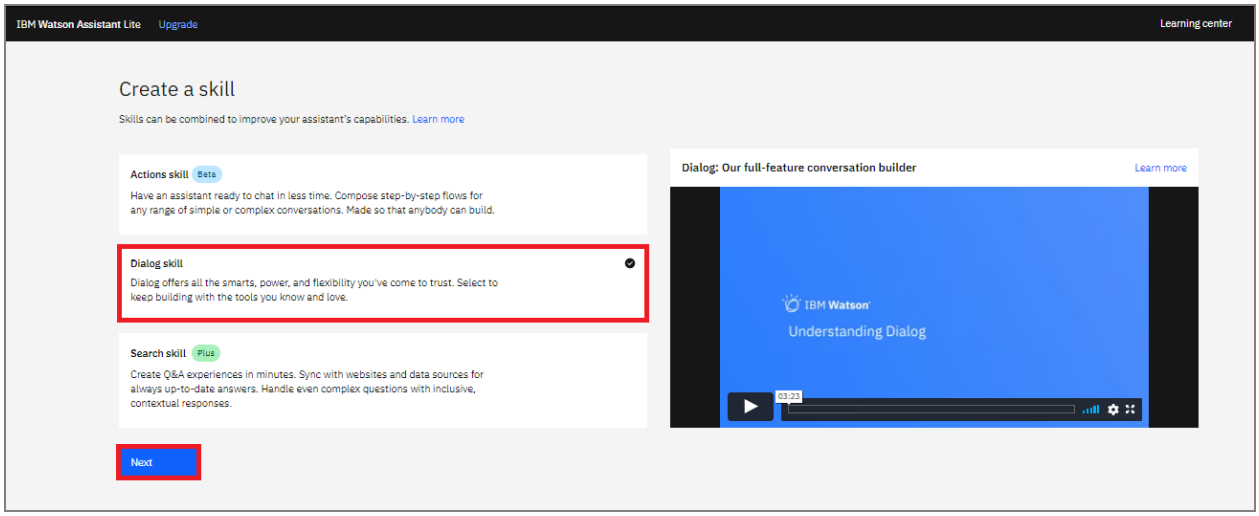
We assume that you are already familiar with Watson Assistant. If you're not, then you can refer back to the previous reading documentation for more information or can consider taking Antonio's course on building chatbots without programming first.

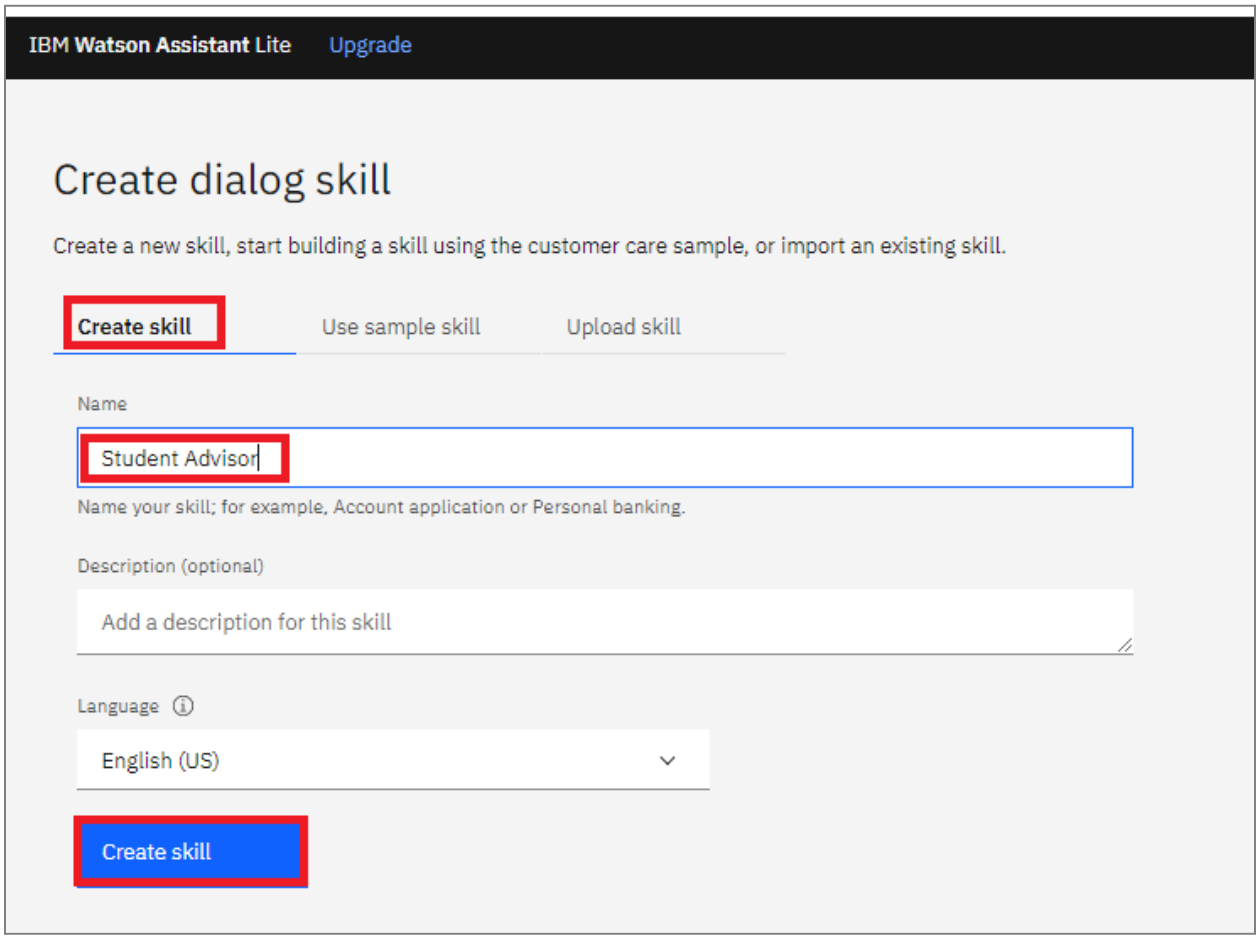
In the next lab, we'll integrate it with the Watson Discovery collection we just defined. The chatbot, as mentioned before, will be rather barebone... just enough of a shell for you to build upon and show you how to integrate it with other services.

Create a chatbot, its intents and entities

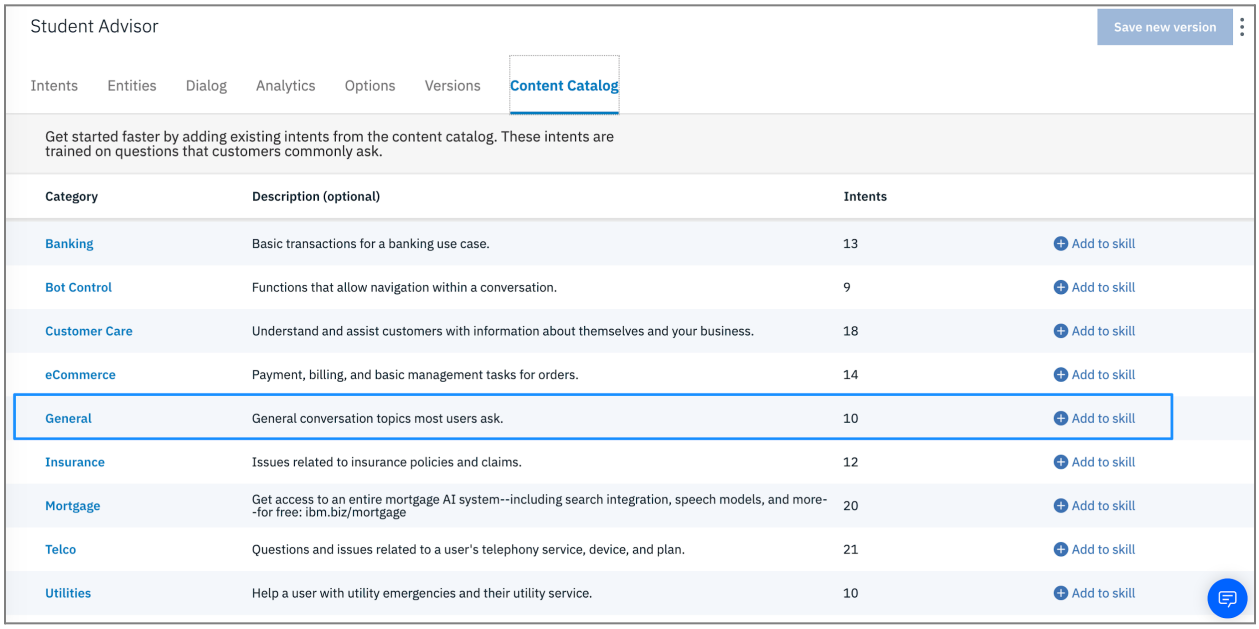
1. Launch your Watson Assistance instance and create an assistant called Student Advisor Chatbot.
2. Add a dialog skill to the assistant (call it Student Advisor or something similar).




IBM Watson Assistant Lite
Upgrade
Learning center
Create a skill
Skills can be combined to improve your assistant's capabilities. Learn more
Actions skill
Beta
Have an assistant ready to chat in less time. Compose step-by-step flows for any range of simple or complex conversations. Made so that anybody can build.
Dialog skill
Dialog offers all the smarts, power, and flexibility you've come to trust. Select to keep building with the tools you know and love.
Search skill
Plus
Create Q&A experiences in minutes. Sync with websites and data sources for always up-to-date answers. Handle even complex questions with inclusive, contextual responses.
Next
Dialog: Our full-feature conversation builder
Learn more
IBM Watson
Understanding Dialog
03:23


IBM Watson Assistant Lite
Upgrade
Create dialog skill
Create a new skill, start building a skill using the customer care sample, or import an existing skill.
Create skill
Use sample skill
Upload skill
Name
Student Advisor
Name your skill; for example, Account application or Personal banking.
Description (optional)
Add a description for this skill
Language
English (US)
Create skill

3. From within that skill, add the *General* collection of intents in the *Content Catalog* by clicking on *Add to skill*.



This will add some generally useful intents, including chitchat to the chatbot. In this lab we'll only leverage a few of them, but feel free to flesh them out as you build a more robust chatbot in the final project within module 6.

4. Now that we have some chitchat intents, let's add some domain-specific intents to our dialog skill. Start by creating an *#Enrollment_Cost intent* , since this is a very common question for students. For the examples used to train Watson, feel free to add the following phrases:

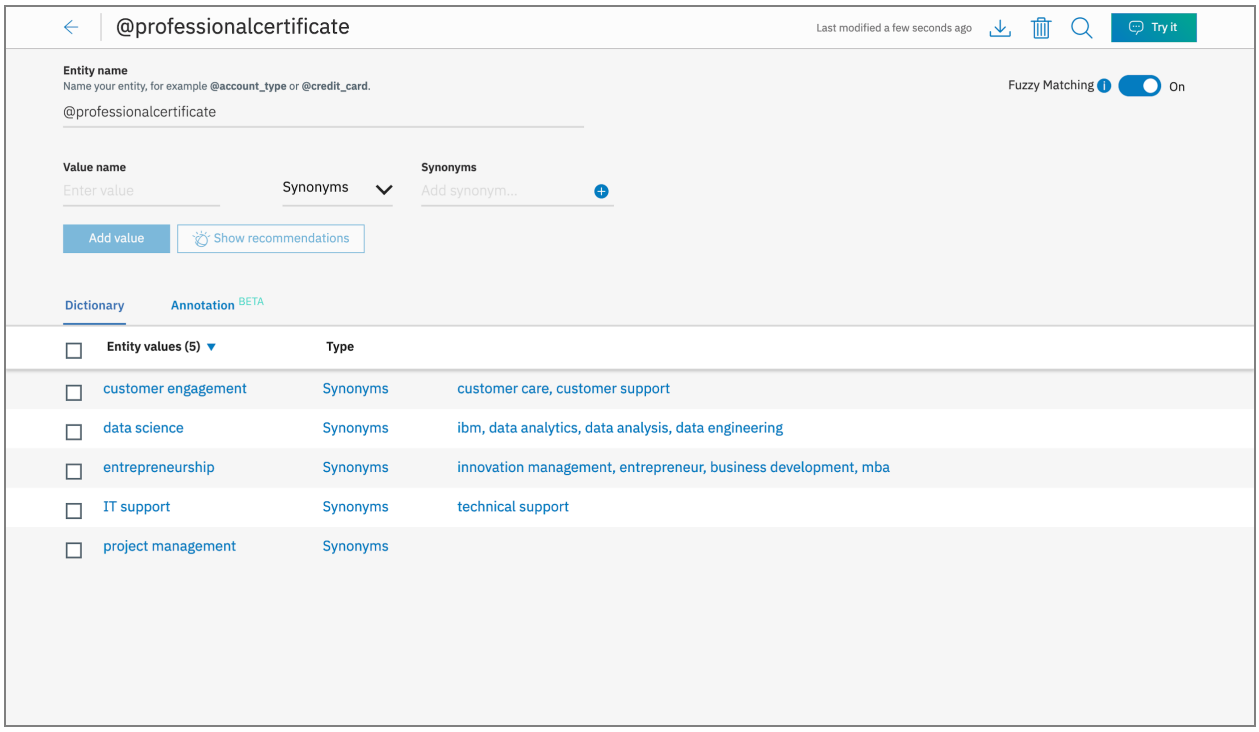
- Are your courses free?
- Can I audit your courses for free?
- Do I have to pay for every course in a specialization?
- How much do your courses cost?
- How much is a subscription?

5. Create a *#Professional_Certificate_Recommendation intent* using the following examples:

- Can you recommend a professional certificate to me?
- Do you offer a deep learning certificate?
- Which computer science certificate?
- Which data science certificate?
- Which data science certificate do you recommend?

Since there are only a few professional certificates being offered at the moment, we can hardcode our responses in a node of our dialog. But we want to provide specific answers, so that when people ask about the Deep Learning certificate, we provide them with a link to the right certificate. So we'll need an entity to identify that information.

6. Go ahead and create a *@professionalcertificate* entity with the following values and synonyms.



7. Finally, create a #Course_Recommendation intent with the following examples:

- Do you have any courses on genetics?
- I want to learn Python
- Recommend me a course on nutrition
- What courses do you recommend?
- What's the best course on Machine Learning?
- Which data science course should I take?

We'll use this intent in the next module when dynamically retrieving responses from the Watson Discovery collection.

Add nodes to the Dialog

With intents and entities defined, it is now time to create the dialog.

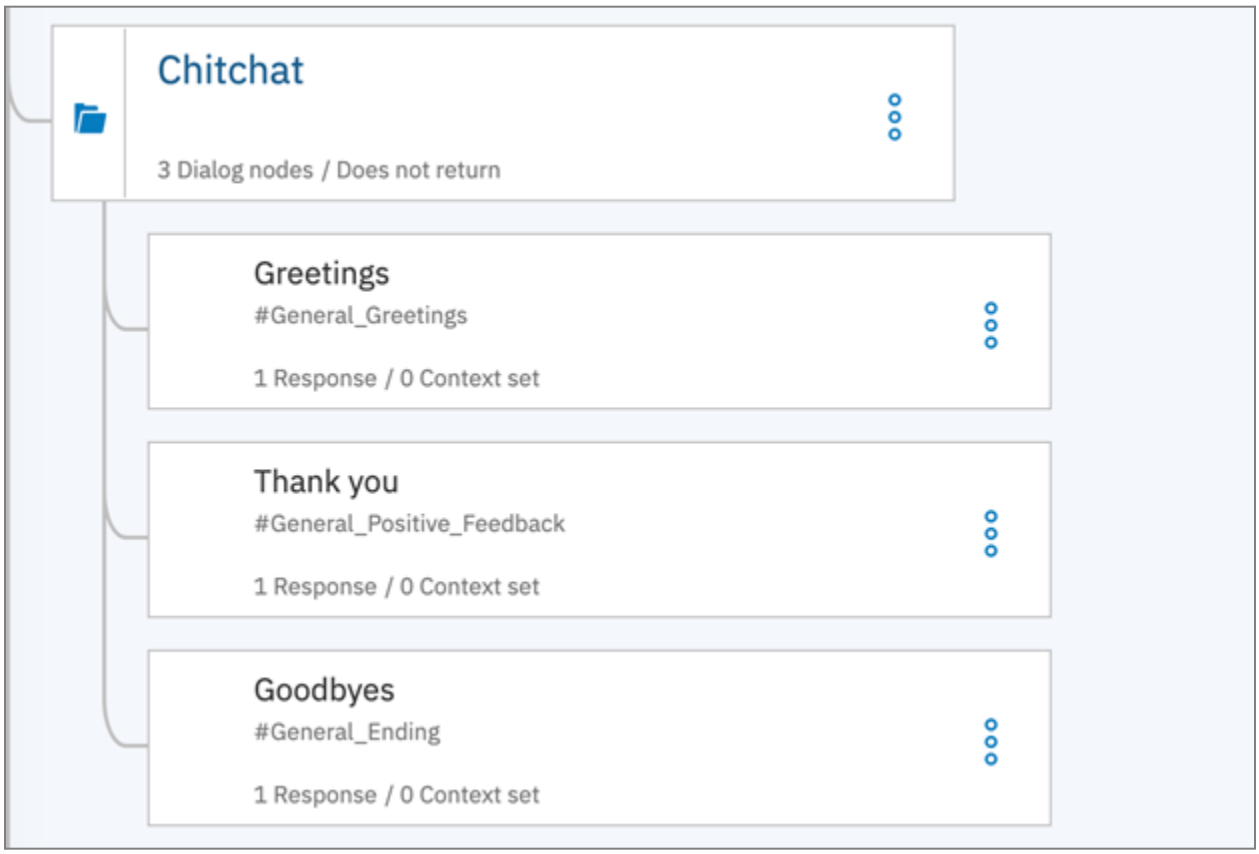
1. From within your skill, click on the *Dialog* tab and create the dialog.

2. Customize the *Welcome node* prompt with:

Hello. I'm a Student Advisor Chatbot. I can help you with your questions about our site and give you course recommendations.

Or something similar.

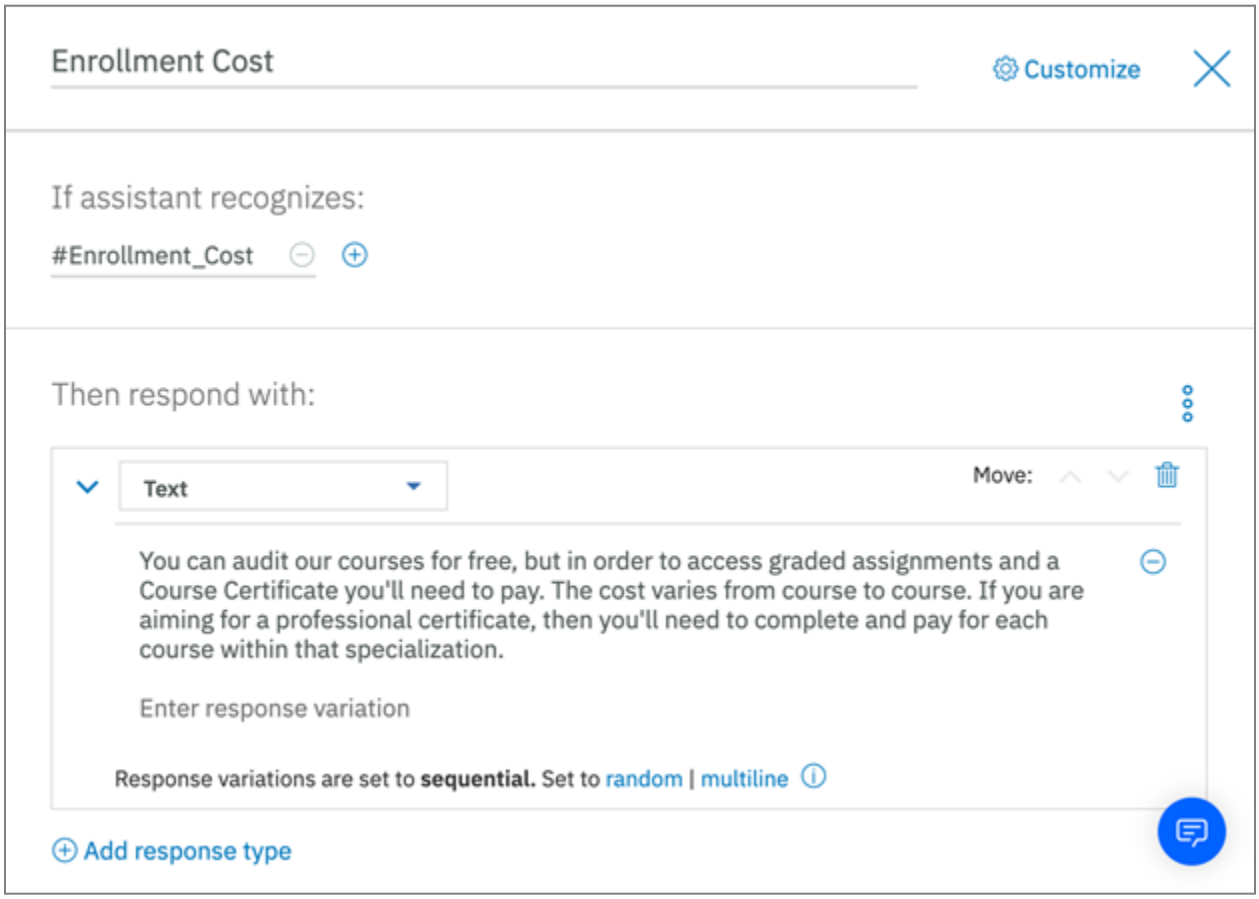
3. If you wish, customize the *Anything else* node responses.
4. Create a Chitchat folder, and add three nodes within it to handle greetings, thank yous, and goodbyes.You can leverage the General intents we added in the first part of this lab (i.e., *#General_Greetings*, *#General_Positive_Feedback*, and *#General_Ending* respectively). Define an appropriate response for each (e.g., *Hi there. How can I help you?* for *Greetings*.)



In module 6, for the project assignment, you'll want to flesh out this chitchat folder further to make the chatbot even more flexible and user friendly.

5. Above the Chitchat folder, we'll need to create three domain-specific nodes. Let's start with the first one. Add an *Enrollment Cost* node with the *#Enrollment_Cost* intent as its condition. For the response use following or something similar:

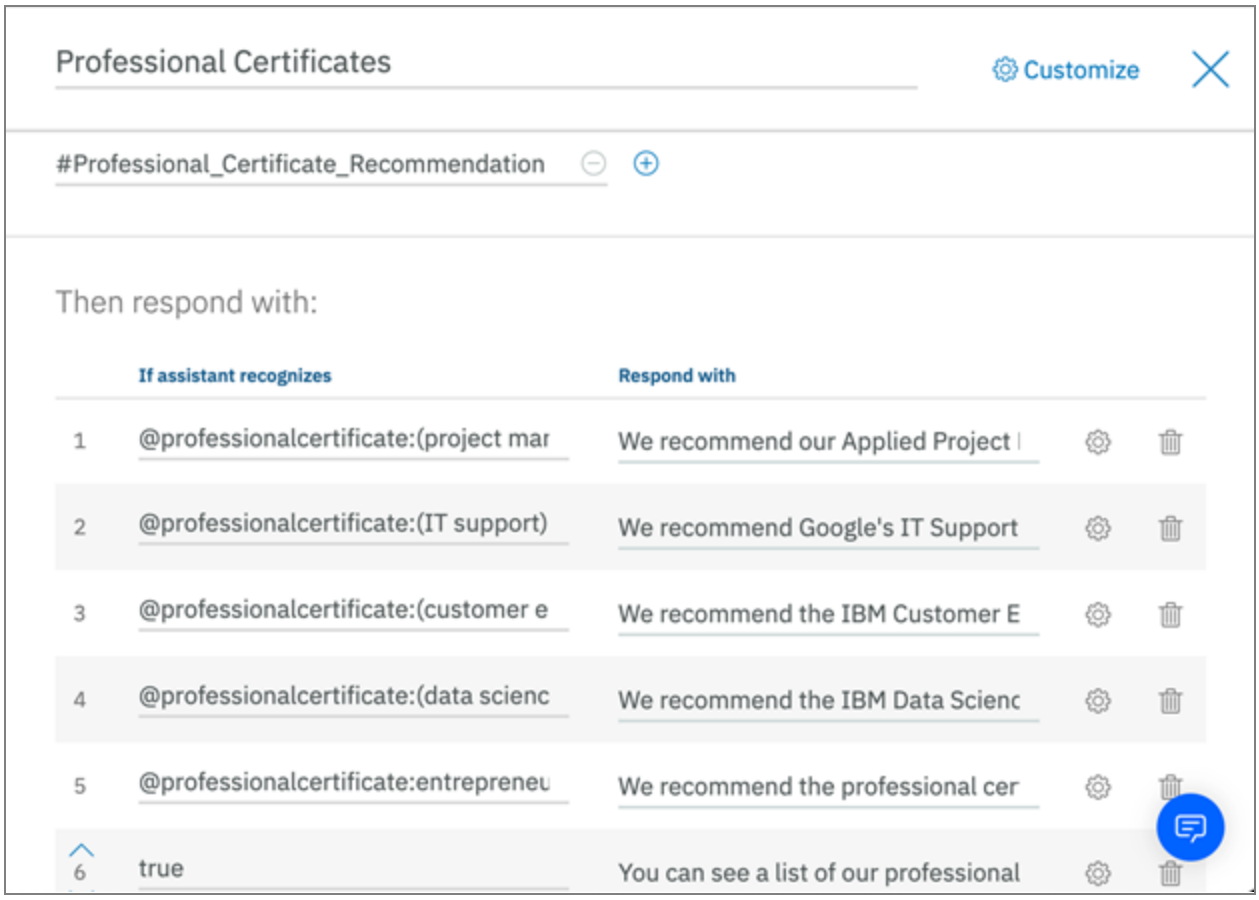
You can audit our courses for free, but in order to access graded assignments and a Course Certificate you'll need to pay. The cost varies from course to course. If you are aiming for a professional certificate, then you'll need to complete and pay for each course within that specialization.



6. Test the chatbot in the *Try it out* panel to ensure everything is working as expected so far.
7. Add a *Professional Certificate* node just below Enrollment Cost. For its condition, use *#Professional_Certificate_Recommendation*.
8. Enable multiple conditional responses from the *Customize link* within the node. Then add a condition and response for each entity value available, and the *true* fallback case, as shown in the table (and image) below.

Condition	Response

@professionalcertificate:(project management)	We recommend our Applied Project Management Certificate by University of California Irvine.
@professionalcertificate:(IT support)	We recommend Google's IT Support certificate.
@professionalcertificate:(customer engagement)	We recommend the IBM Customer Engagement Specialist professional certificate.
@professionalcertificate:(data science)	We recommend the IBM Data Science professional certificate.
@professionalcertificate:entrepreneurship	We recommend the professional certificate in Innovation Management and Entrepreneurship by HEC Paris.
true	You can see a list of our professional certificates on our site.



9. Finally, add a *Courses* node with the condition *#Course_Recommendation*. Ignore the response section for now.

At this point you have a basic, but functional chatbot to assist online learners. In the next lab, we'll see how to connect it to our Watson Discovery collection to have our newly defined Courses node dynamically retrieve and issue responses.

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

Objective:

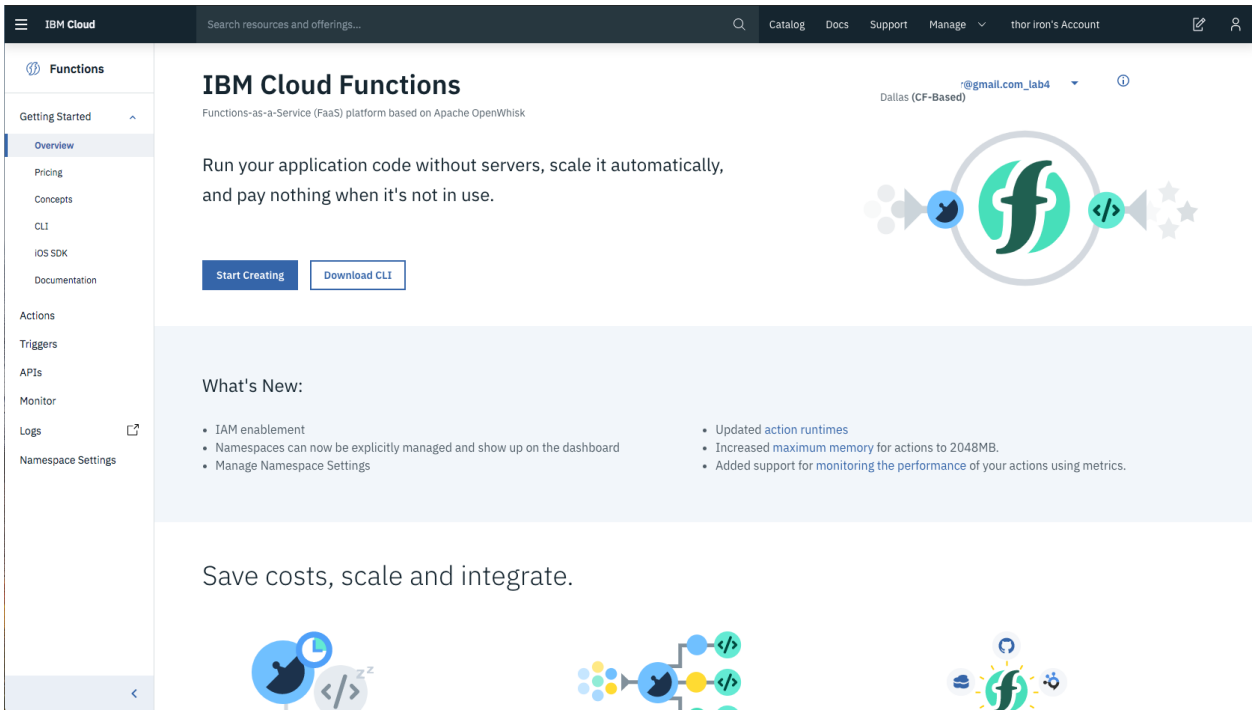
- How to use IBM Cloud Functions to make programmatic calls from dialog node.
- Learn about IBM functions.

Pre-requisite:

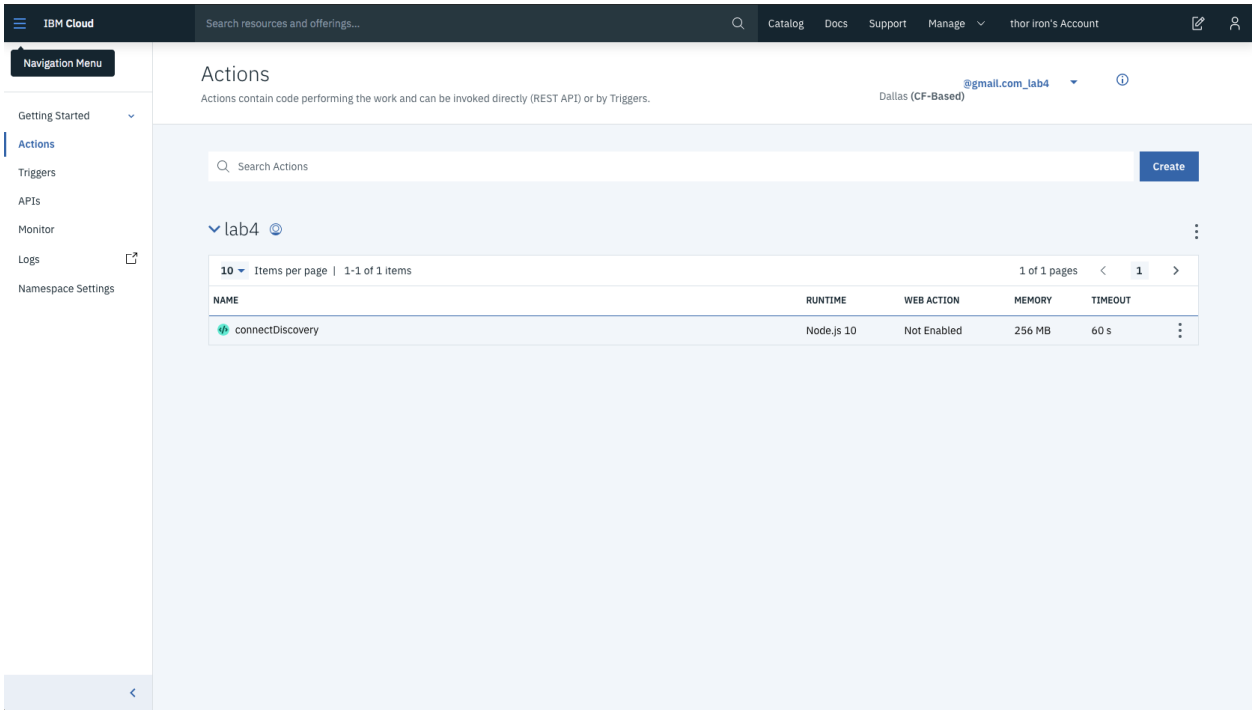
You should have completed the previous lab-Adding Discovery to the Chatbot:Part-1 succsesfully. If not, then please consider doing the previous lab first before moving further with this lab.

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.

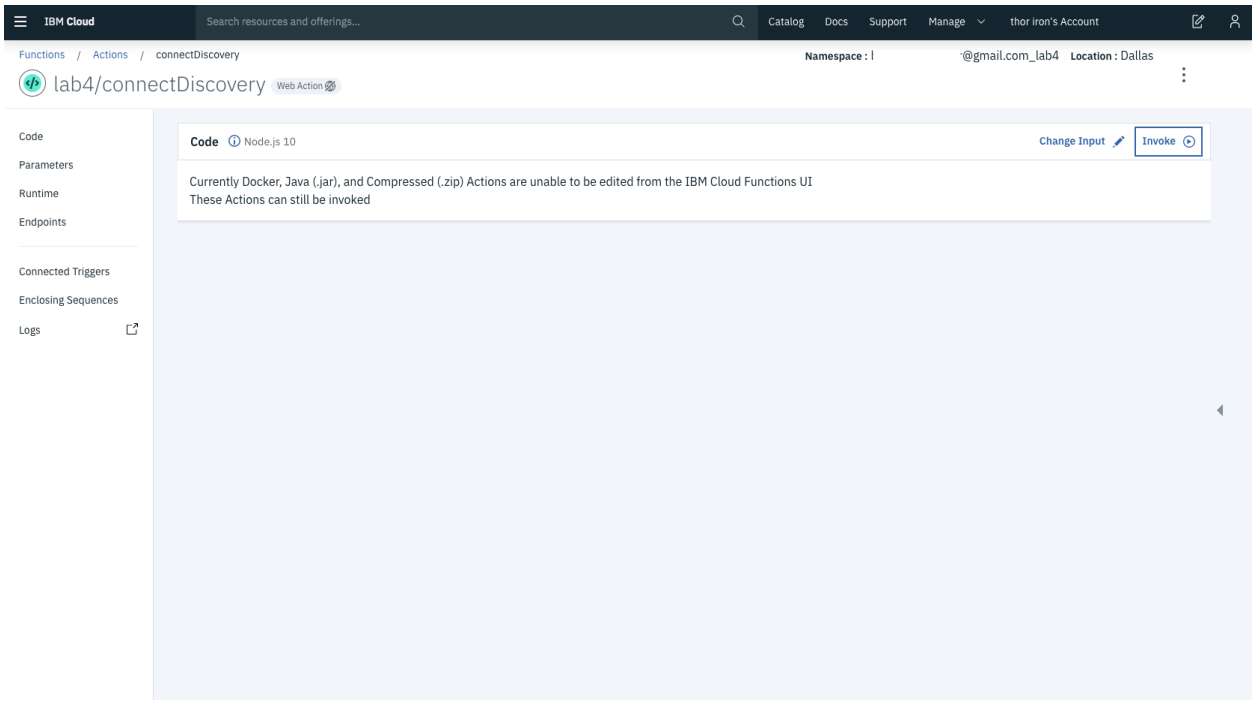
1. Log in to IBM Cloud and then search for Functions in the search bar. Select Functions. After the page is loaded, make sure the top right hand side shows the name of your organization_lab4. If your organization is abcd@gmail.com, then you should see the following, <abcd@gmail.com_lab4>. Click Actions.



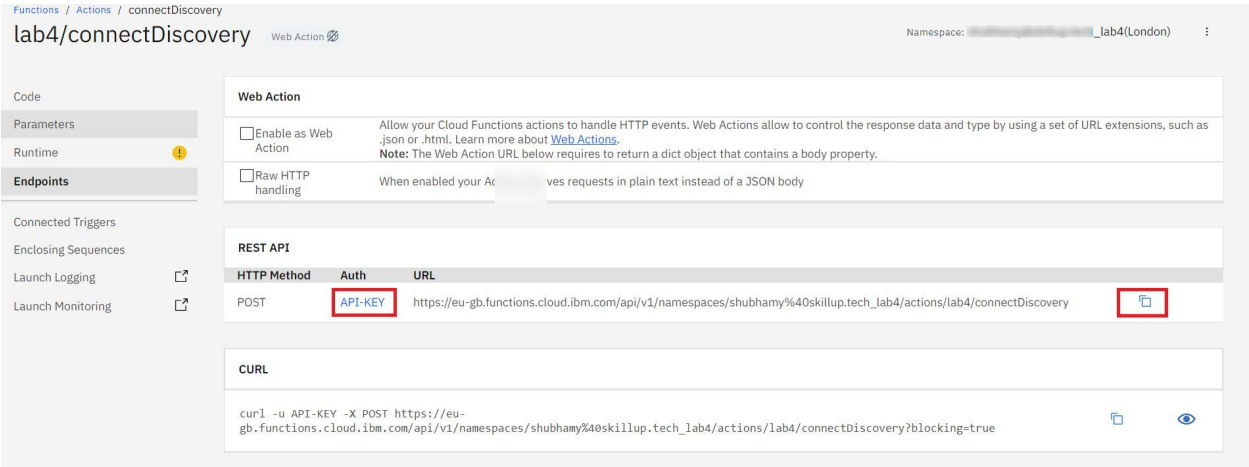
2. Click connectDiscovery action under lab4 package.



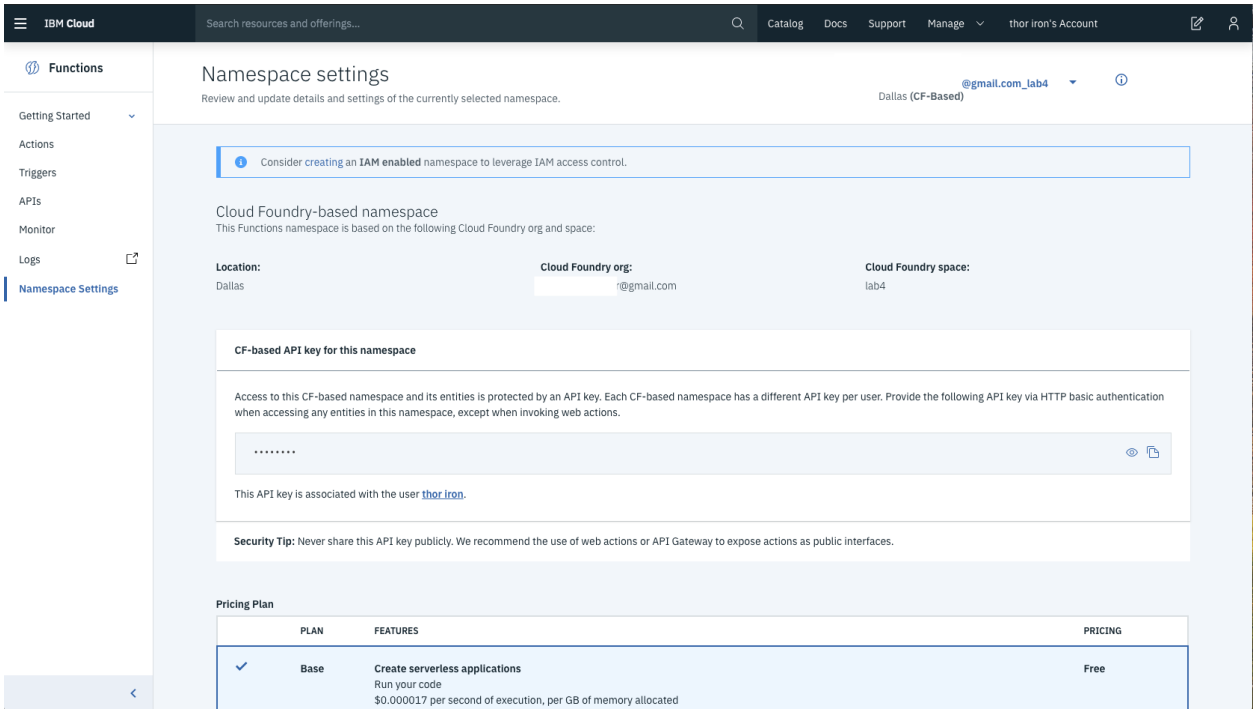
3. Click Endpoints on the left sidebar.



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.



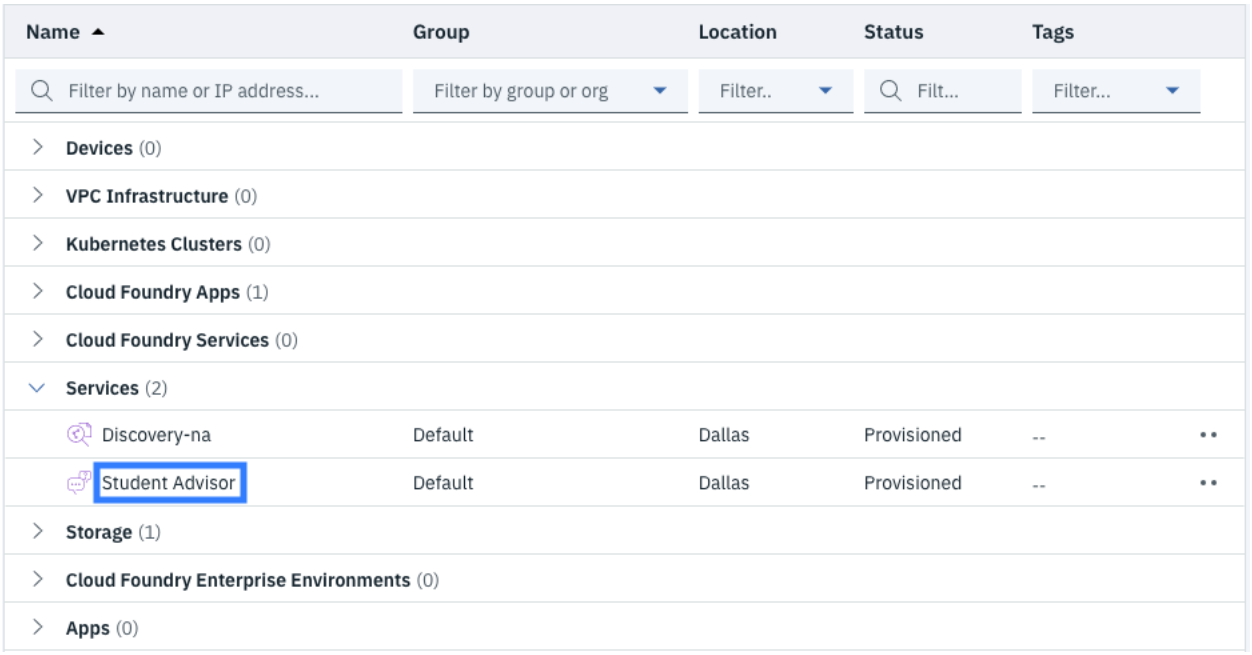
5. Click on the eye icon to obtain your CF-based API key and make a copy of the credentials.



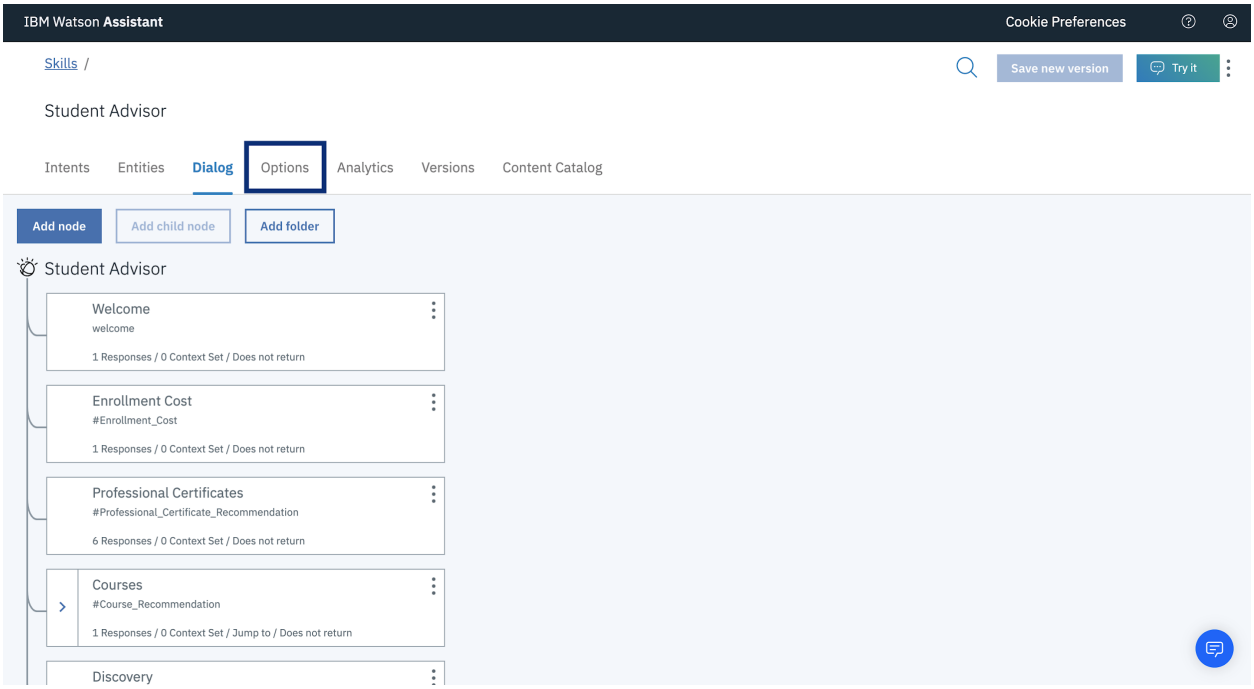
You're now all set to begin integrating this function with Watson Assistant.

Integrate Discovery with our Chatbot

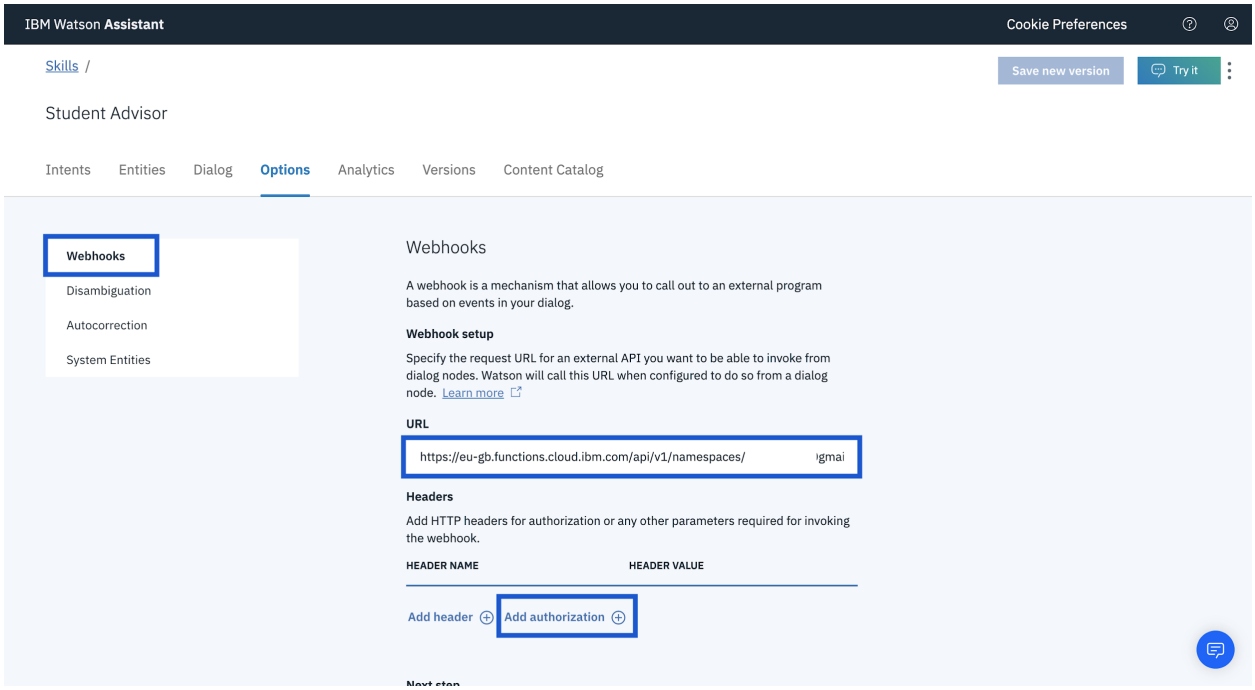
1. From your Dashboard, click on the Watson Assistant service you created in Lab 3. (Click on the name, not the icon next to it.) and then click Launch Watson Assistant.



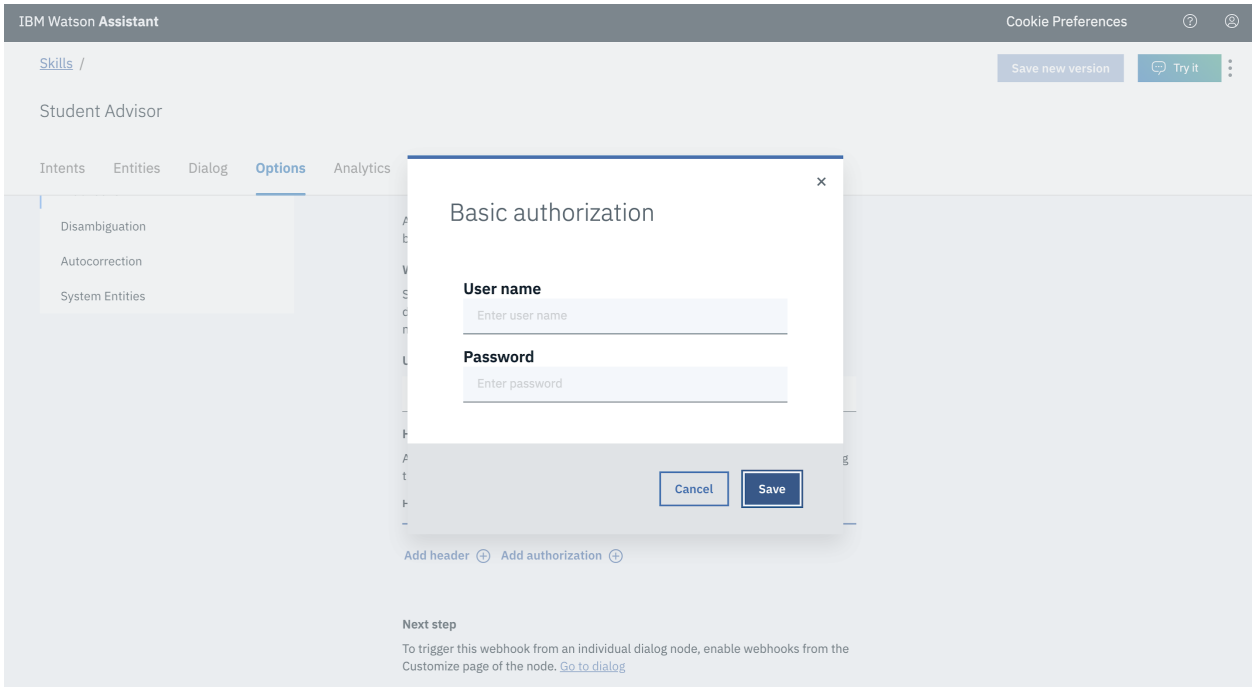
2. Under Skills tab, click Student Advisor skill. Then, click the Options tab.



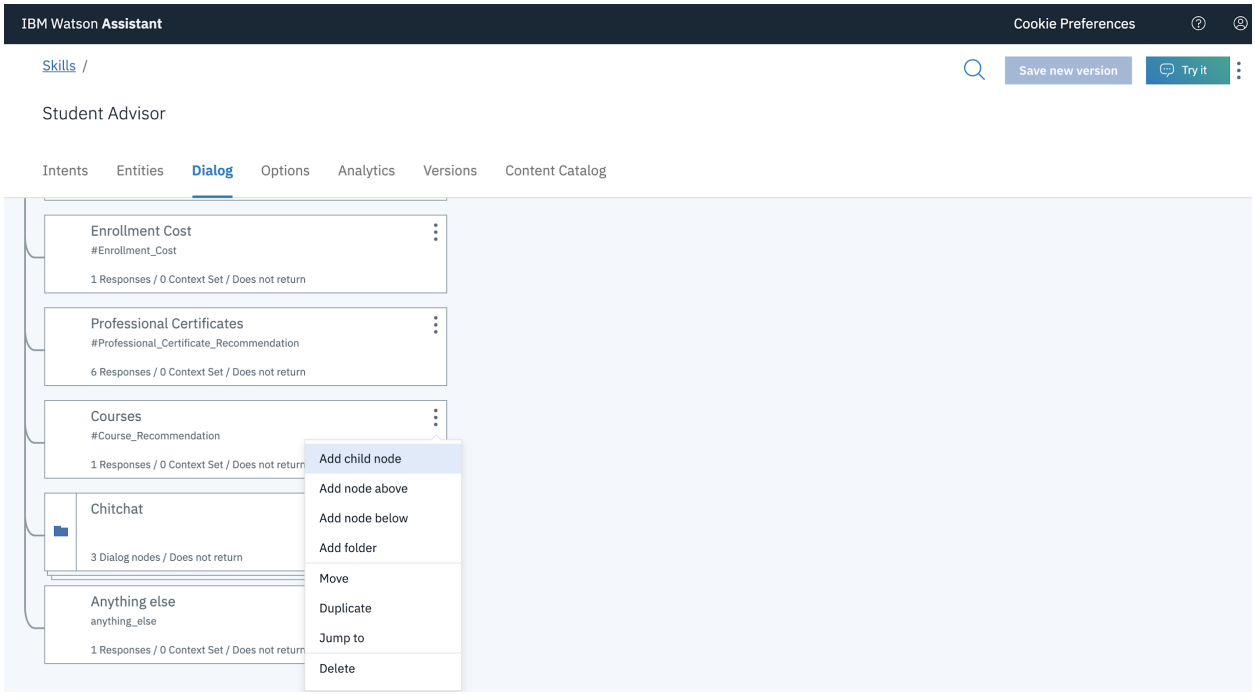
3. Click Webhooks. In the URL field, specify the REST API URL for the action. Append a ?blocking=true parameter to the action URL to force a synchronous call to be made (e.g. https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/YOUR-ORG_YOUR-SPACE/actions/lab4/connectDiscovery?blocking=true). Finally,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



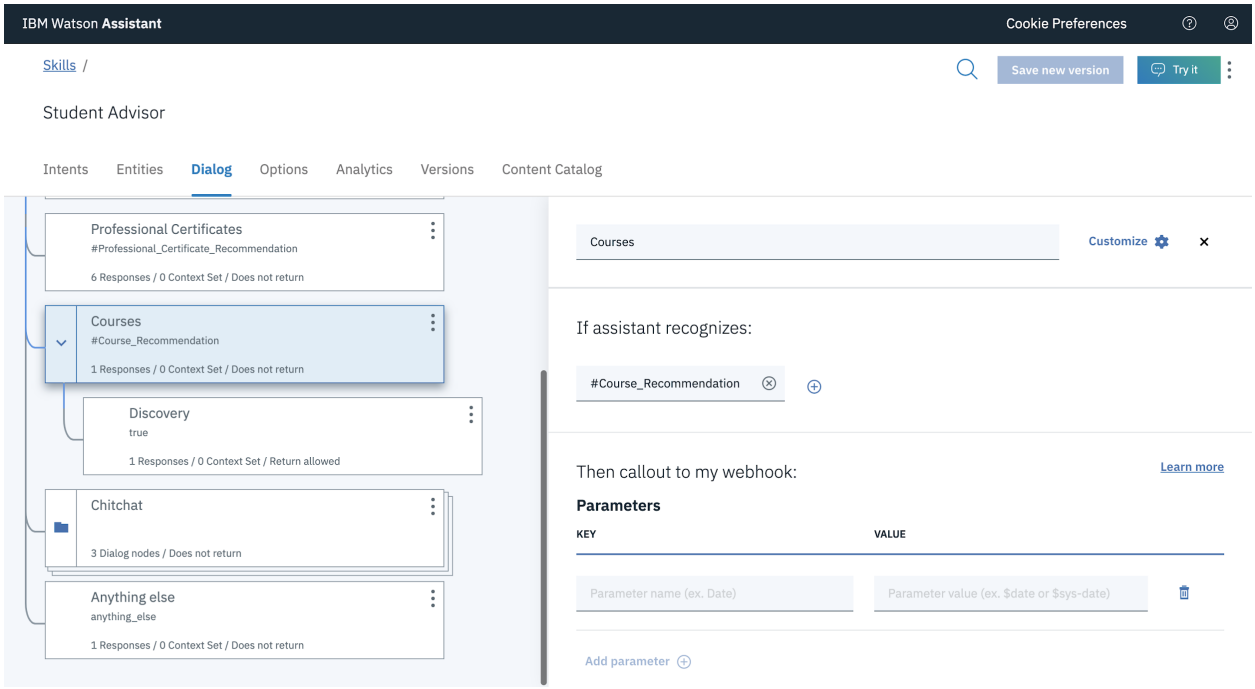
4. We'll use the CF-based API key you obtained from action endpoints. The part before the colon(:) is your User name. Copy this to the User name field. Similarly, the part after the colon(:) is your password. (e.g. Your action API key = User name:password). Finally save.



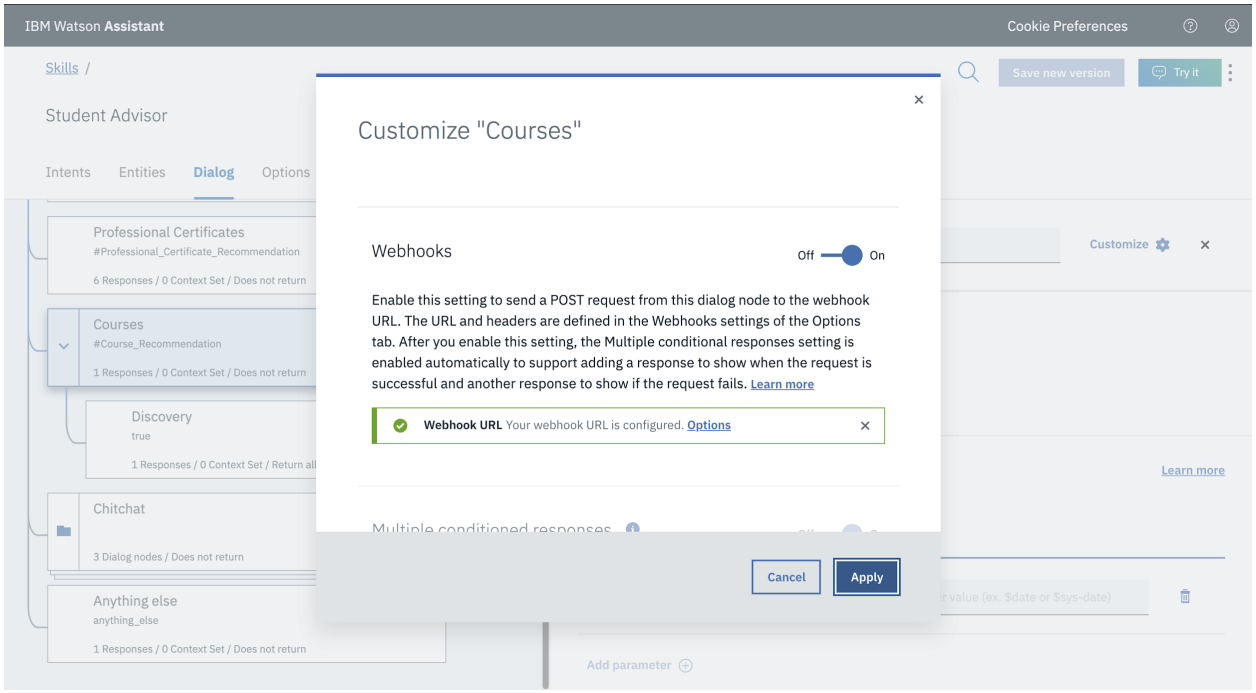
5. Find Courses node under the Dialog tab. Click the 3 dots icon in the Courses node and click Add child node. Then, name the new node (e.g., Discovery).



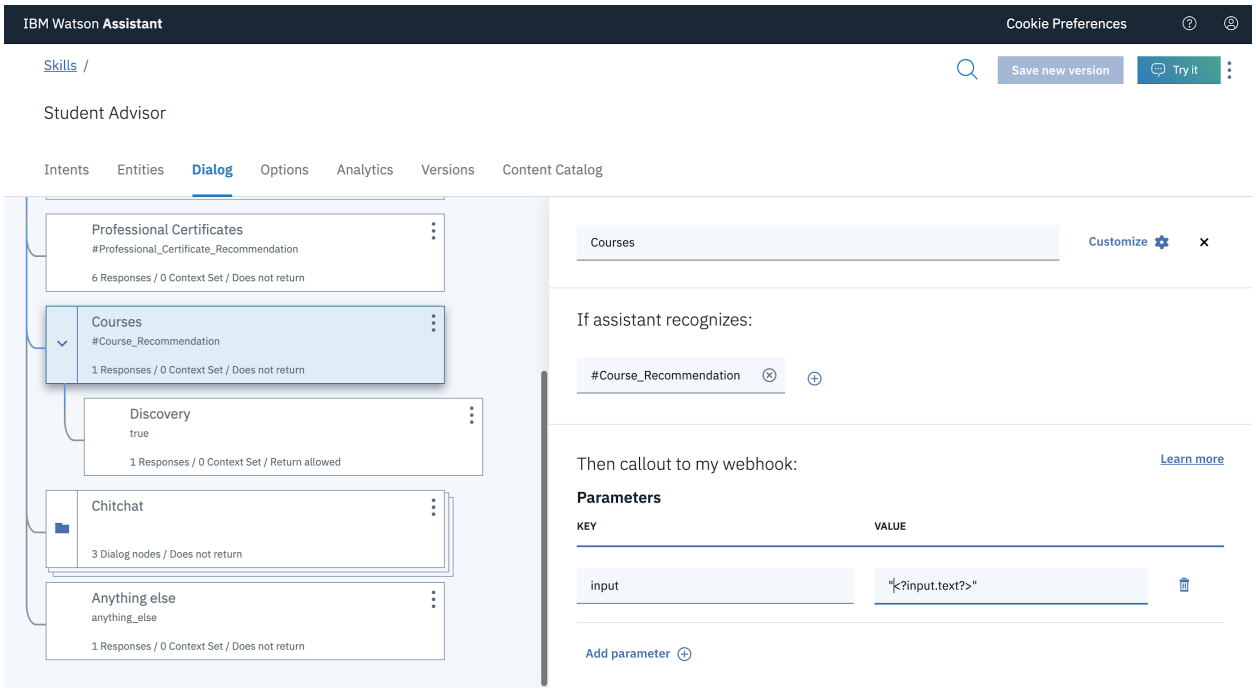
6. Click the Courses node and then click Customize to customize the node.



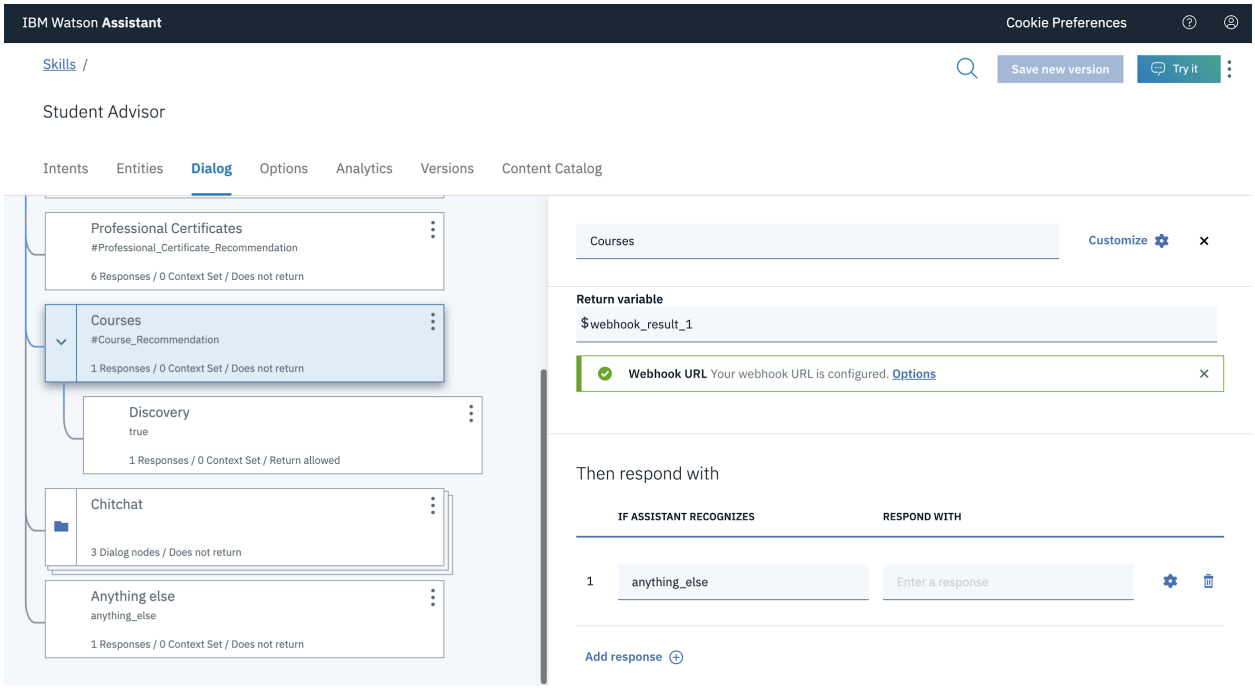
7. Scroll down to the *Webhooks* section, and switch the toggle to On, and then click Apply.



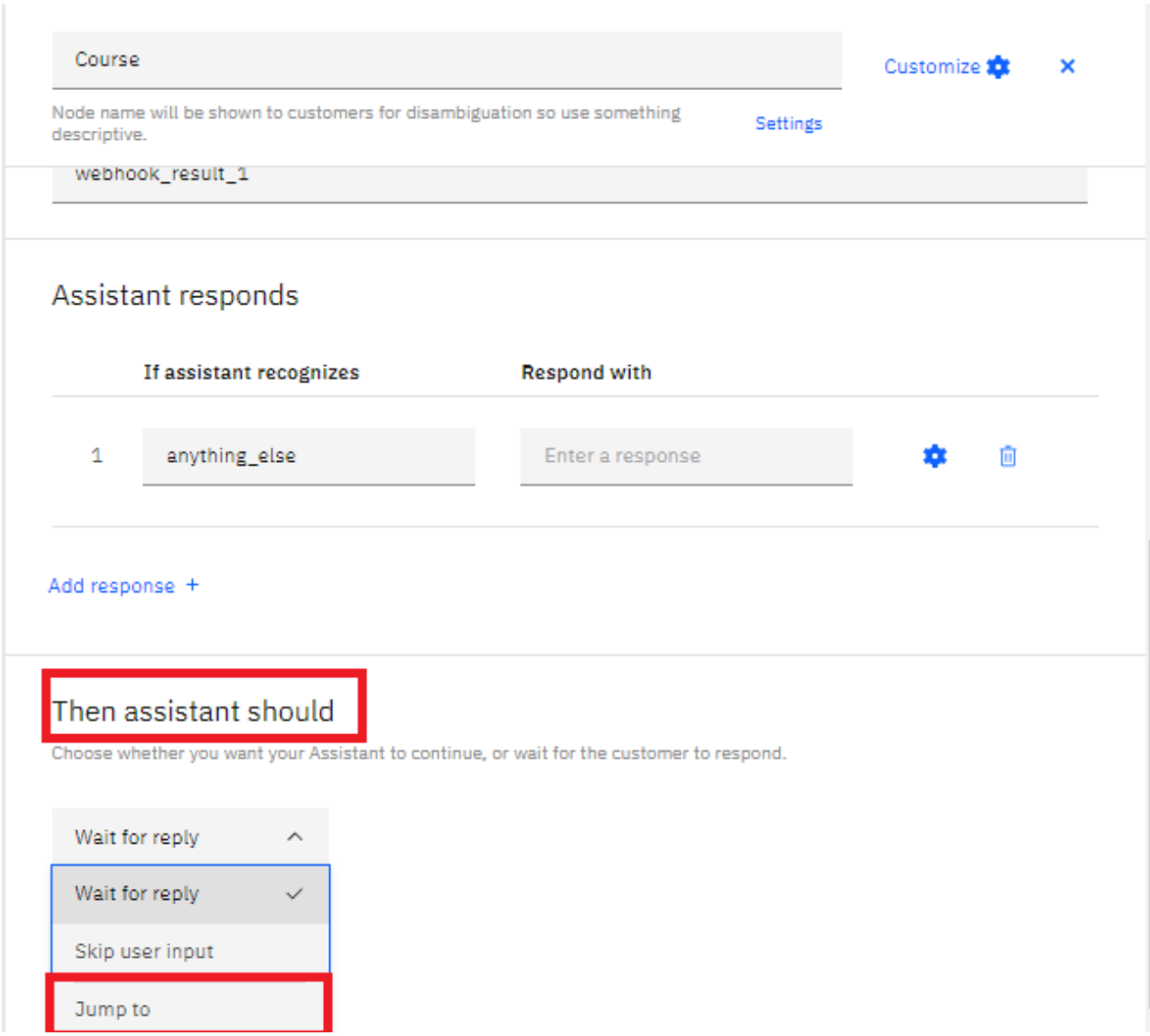
8. Now, we will add a parameter, input as a key and `"<?input.text?>"` as a value.

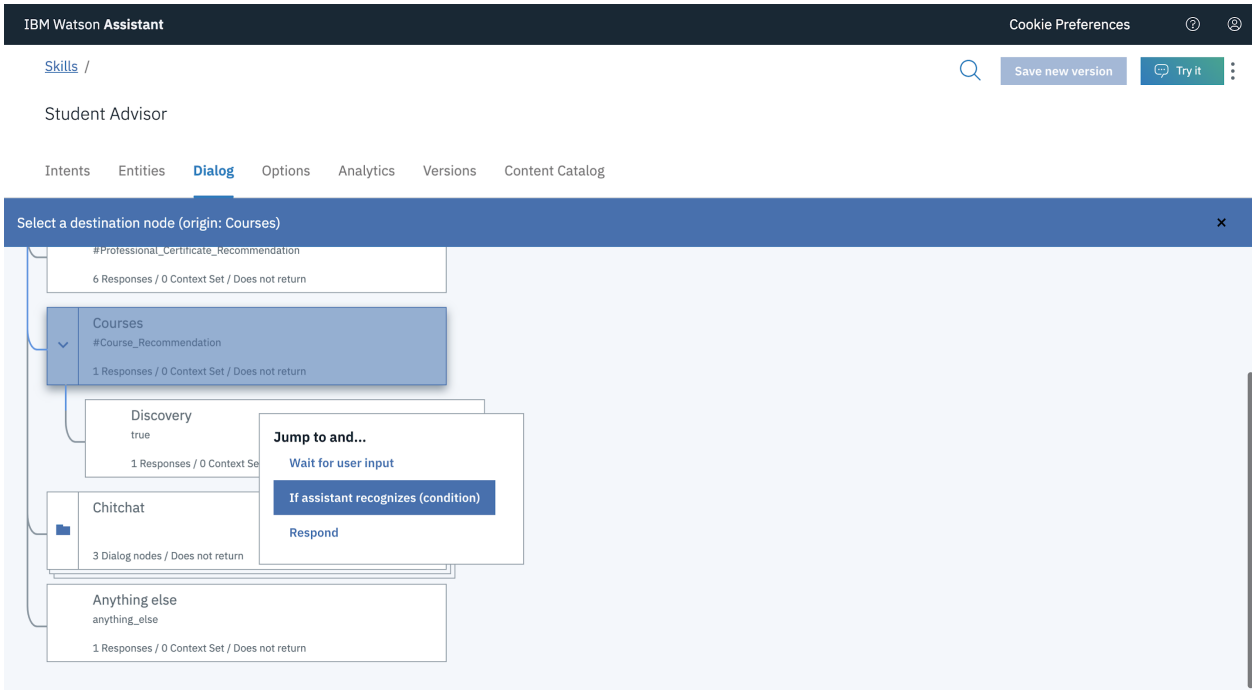


9. Scroll down and set the Return variable to webhook_result_1.



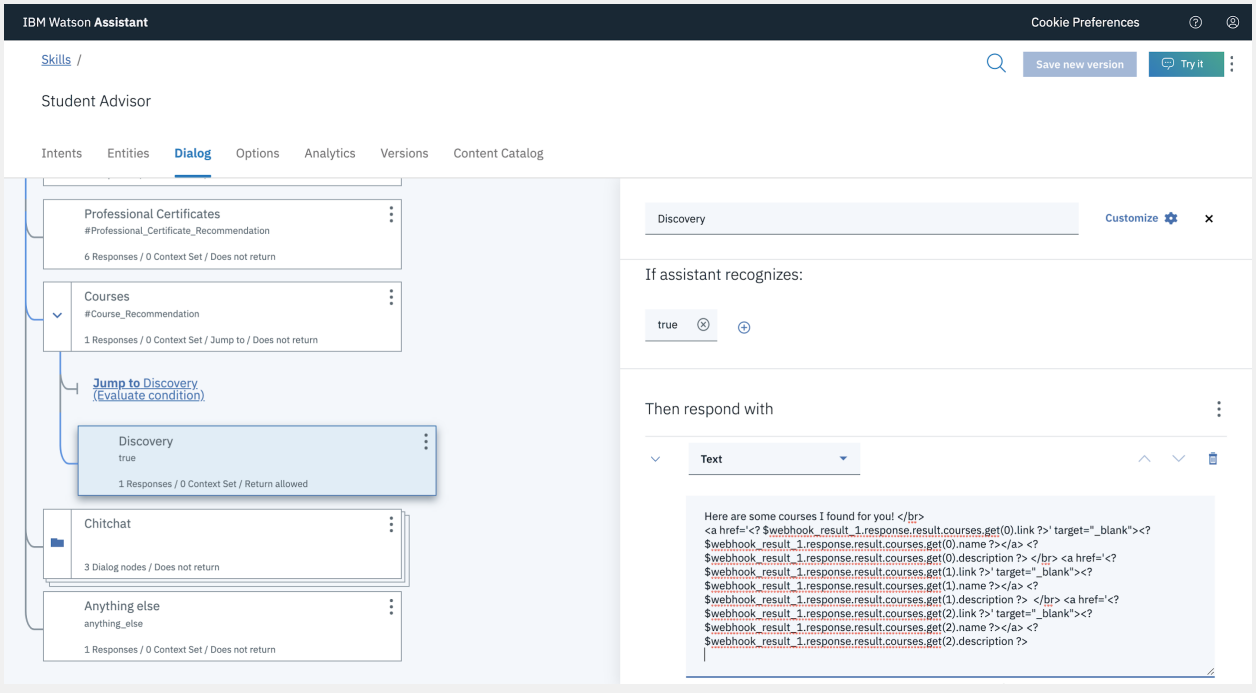
10. Scroll down and under And Then assistant should, select Jump to option and select the child node you created and then select If assistant recognizes(condition).


Course
Customize
Node name will be shown to customers for disambiguation so use something descriptive.
Settings
webhook_result_1
Assistant responds
If assistant recognizes
Respond with
1
anything_else
Enter a response
Add response +
Then assistant should
Choose whether you want your Assistant to continue, or wait for the customer to respond.
Wait for reply
Wait for reply
Skip user input
Jump to



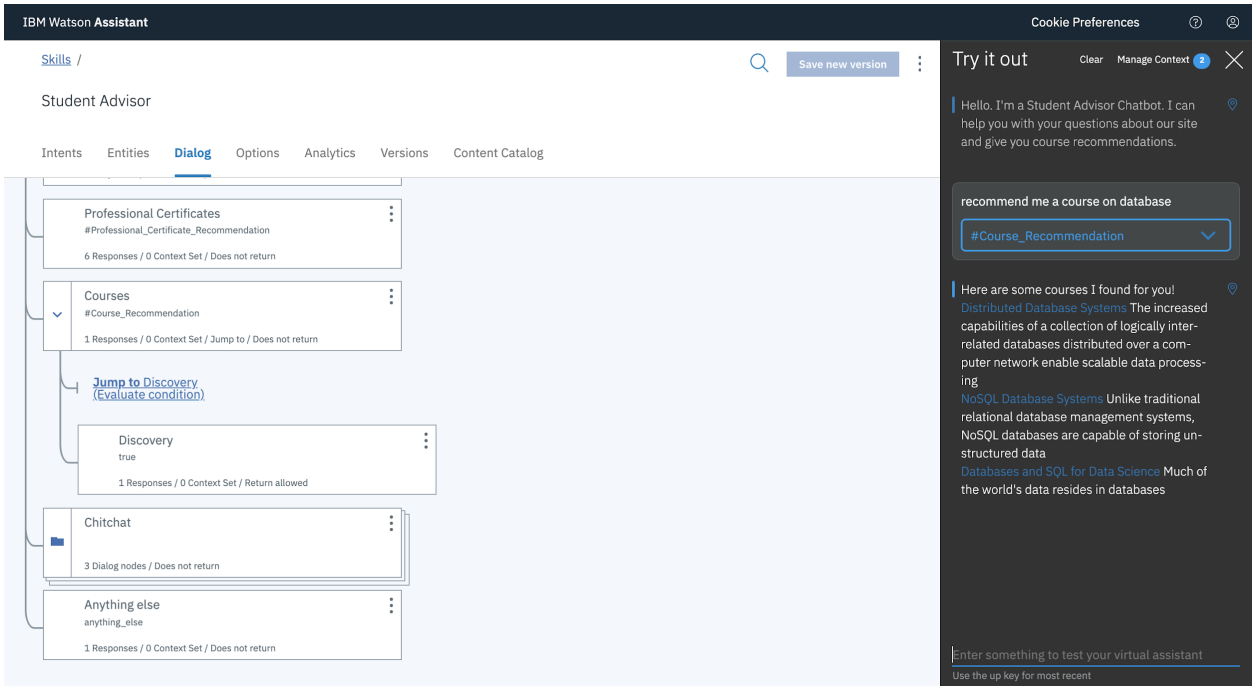
11. Put true as a condition for the node, as we always want to execute it when giving course recommendations. Add the following sample response, which retrieves a list of relevant courses from Discovery via the Function we defined earlier on.

```
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.

Lab 7: Deploying to Slack (1h)

Objective for Exercise:

- How to integrate the basic chatbot with Slack.

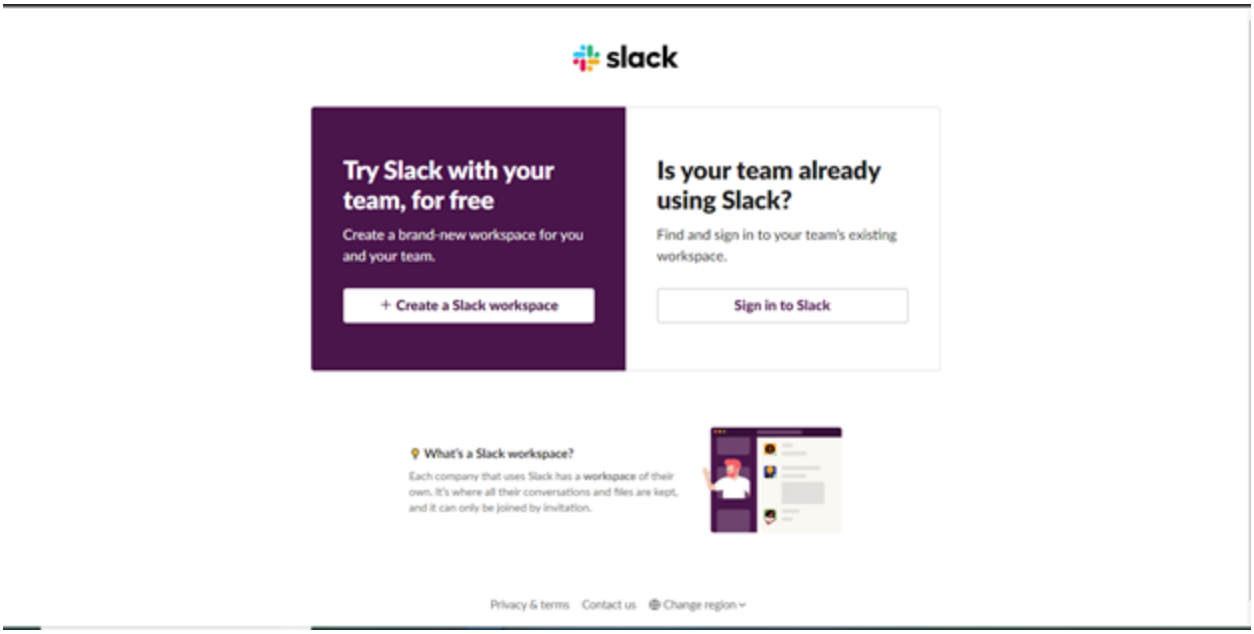
Deploying Your Chatbot On Slack

Slack is a communication tool, typically used by people working in a team. It can also be a channel for you to connect with friends, family or anyone that you want to share information with. You can improve the workflow in your team or set reminders of important events with your friends and family by deploying chatbots on Slack.

For this lab you will integrate our chatbot with Slack.

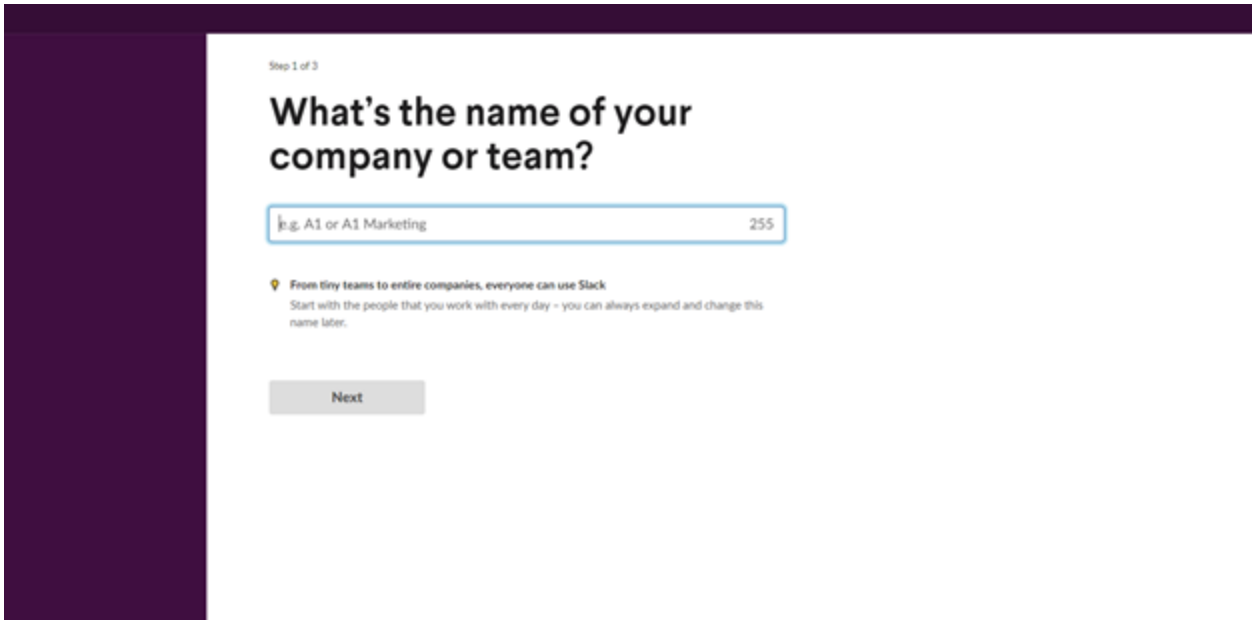
CREATE A SLACK APP

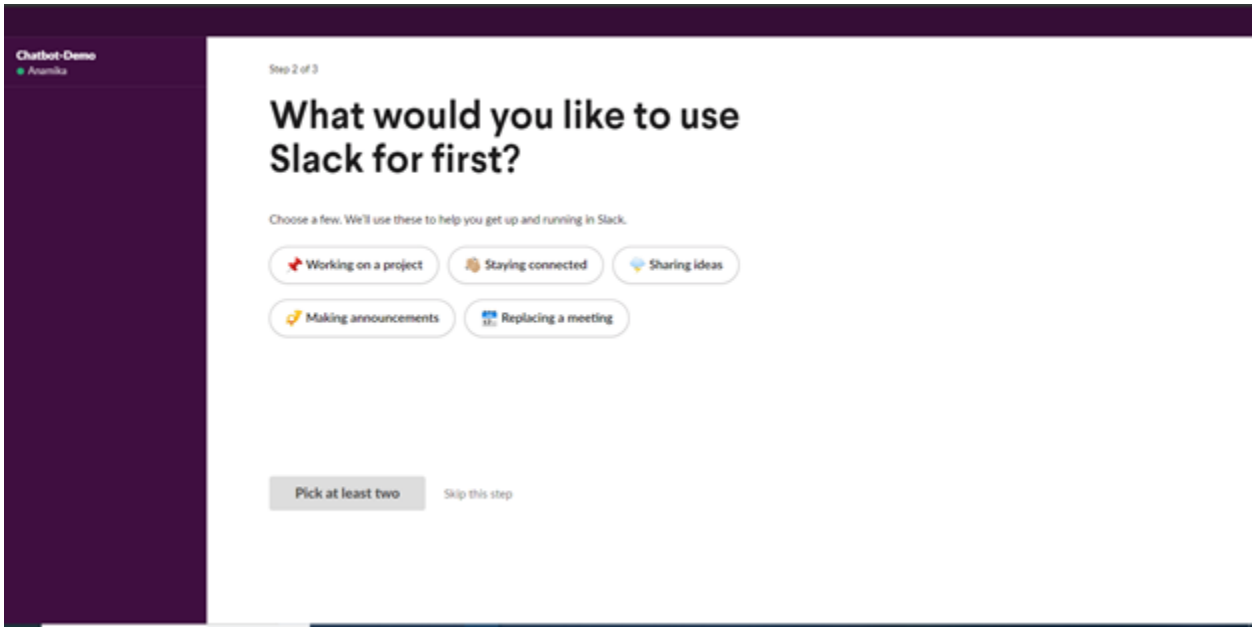
1. Head over to Slack.com, click on the *Try Slack* button, then click on Create a Slack Workspace.

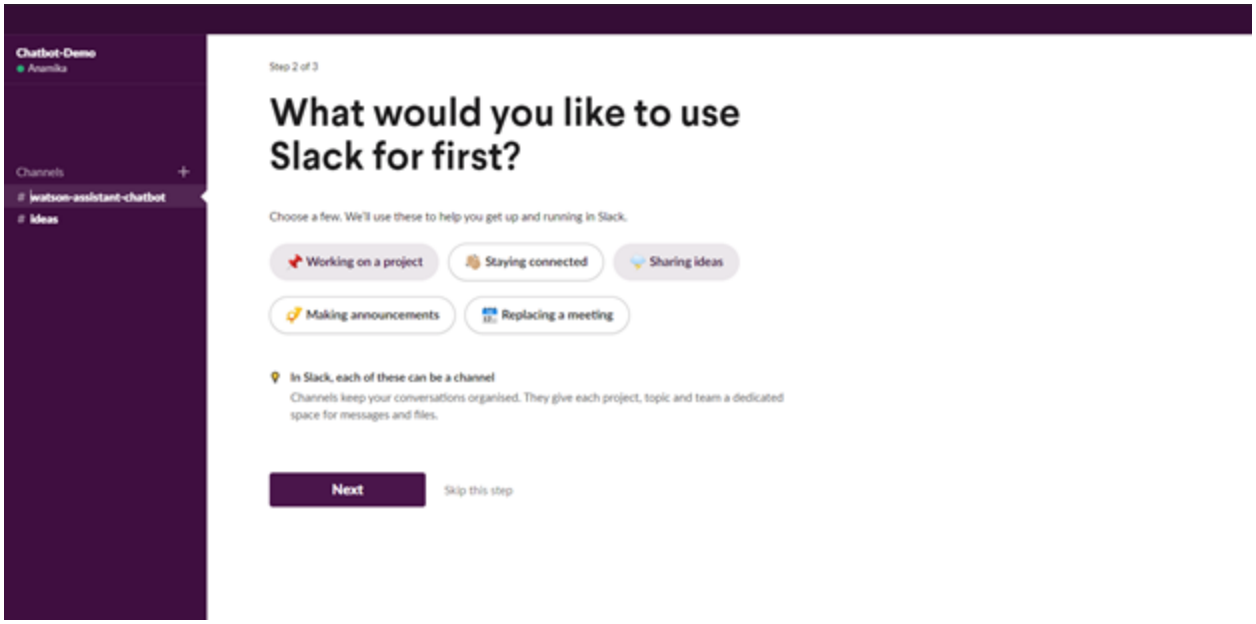


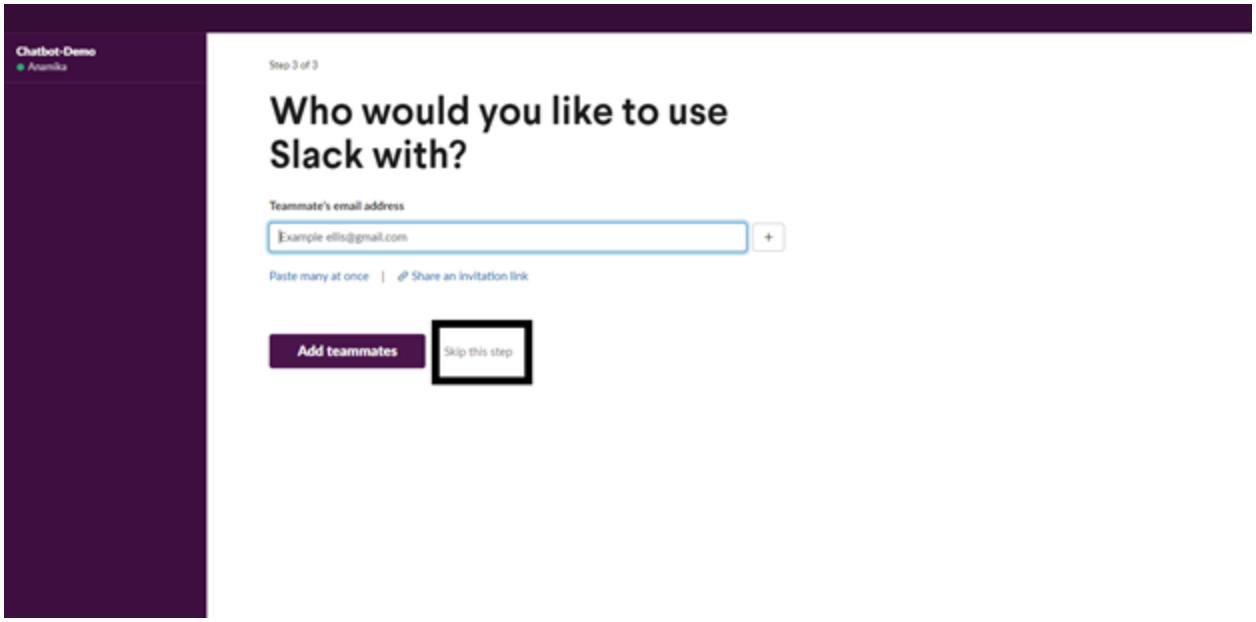
2. Next, enter your email address and click on the *Confirm* button. You might be asked to enter a confirmation code that will be sent to the email address you specified.

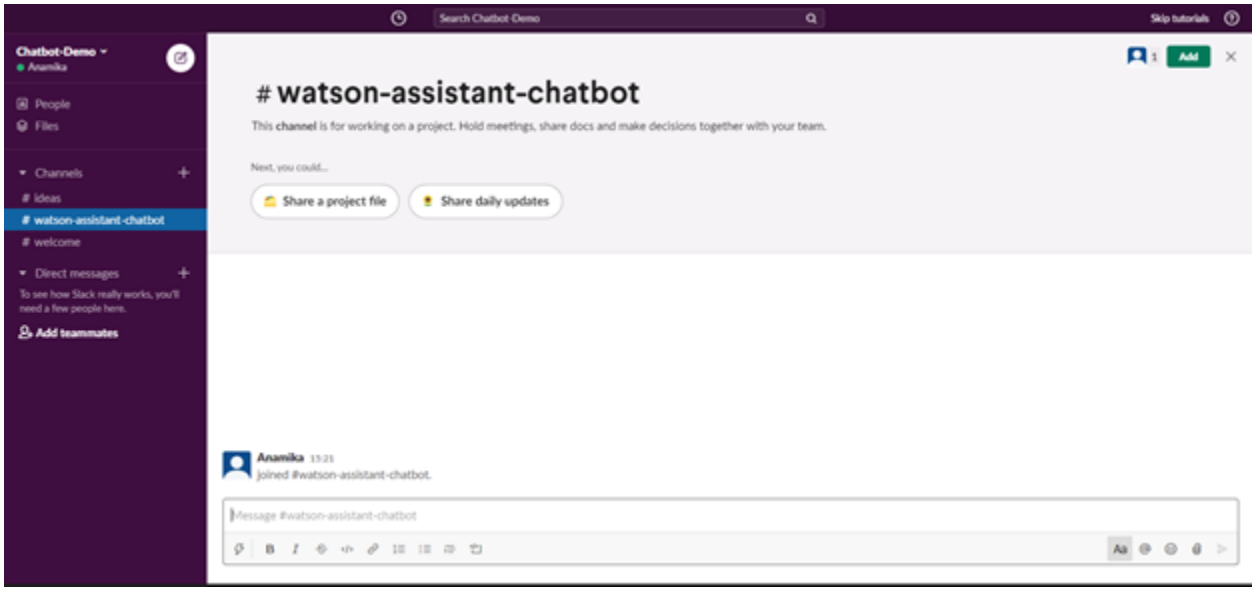
3. Specify a team and project name of choice. Feel free to skip inviting teammates.


Step 1 of 3
What's the name of your company or team?
e.g. A1 or A1 Marketing
255
From tiny teams to entire companies, everyone can use Slack
Start with the people that you work with every day – you can always expand and change this name later.
Next


Chatbot-Demo
Anamika
Step 2 of 3
What would you like to use Slack for first?
Choose a few. We'll use these to help you get up and running in Slack.
Working on a project
Staying connected
Sharing ideas
Making announcements
Replacing a meeting
Pick at least two
Skip this step

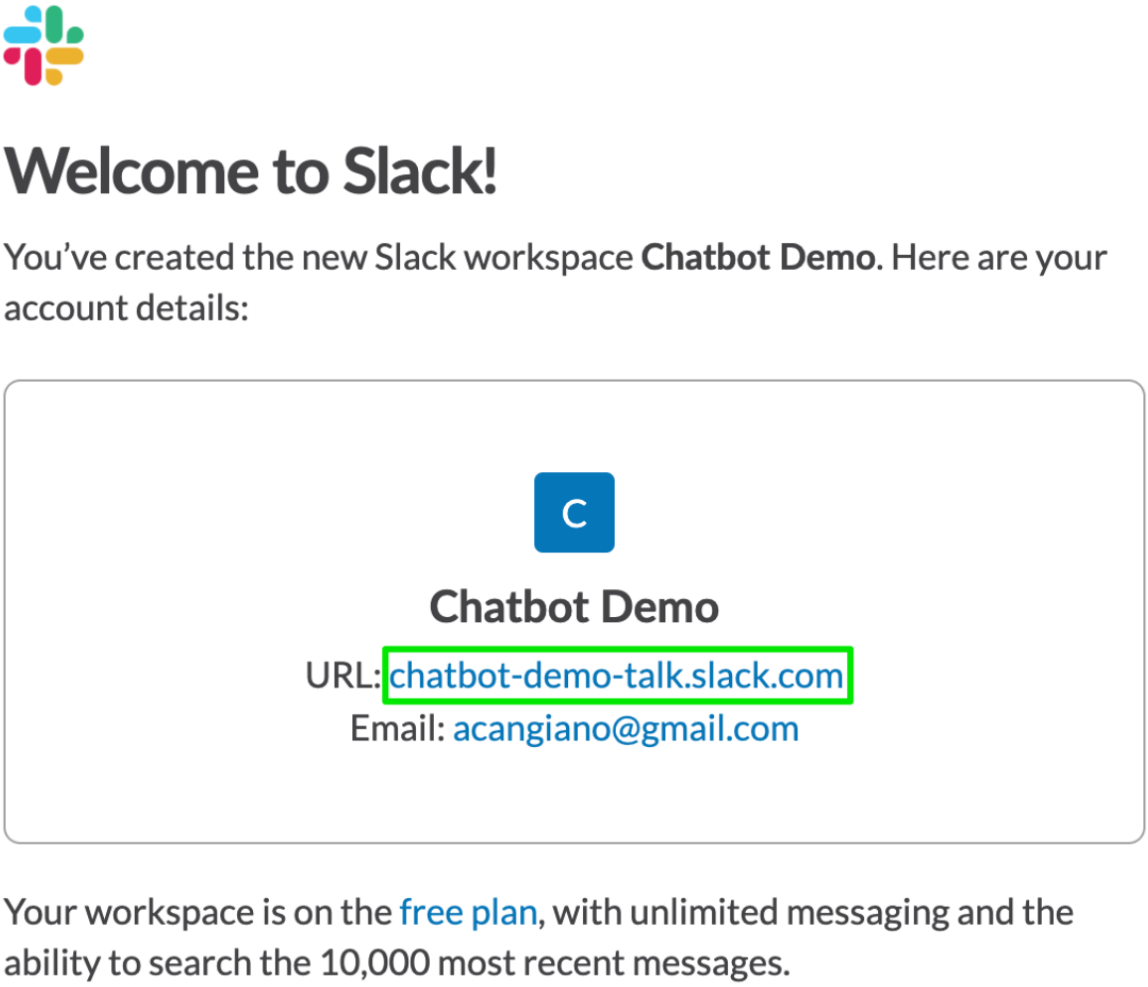

Chatbot-Demo
Anamika
Channels
watson-assistant-chatbot
ideas
Step 2 of 3
What would you like to use Slack for first?
Choose a few. We'll use these to help you get up and running in Slack.
Working on a project
Staying connected
Sharing ideas
Making announcements
Replacing a meeting
In Slack, each of these can be a channel
Channels keep your conversations organised. They give each project, topic and team a dedicated space for messages and files.
Next
Skip this step



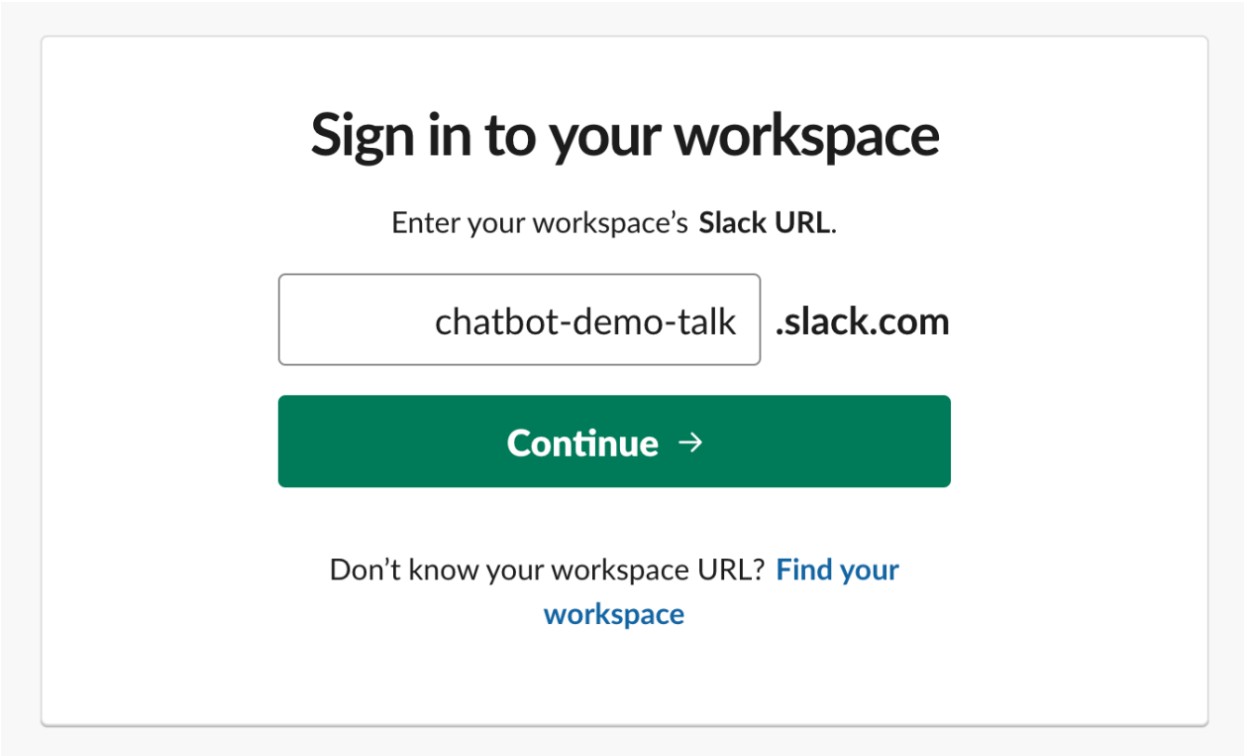


Now that we have a Slack workspace and channel, we need to proceed to create a Slack App to use to communicate with our existing chatbot.

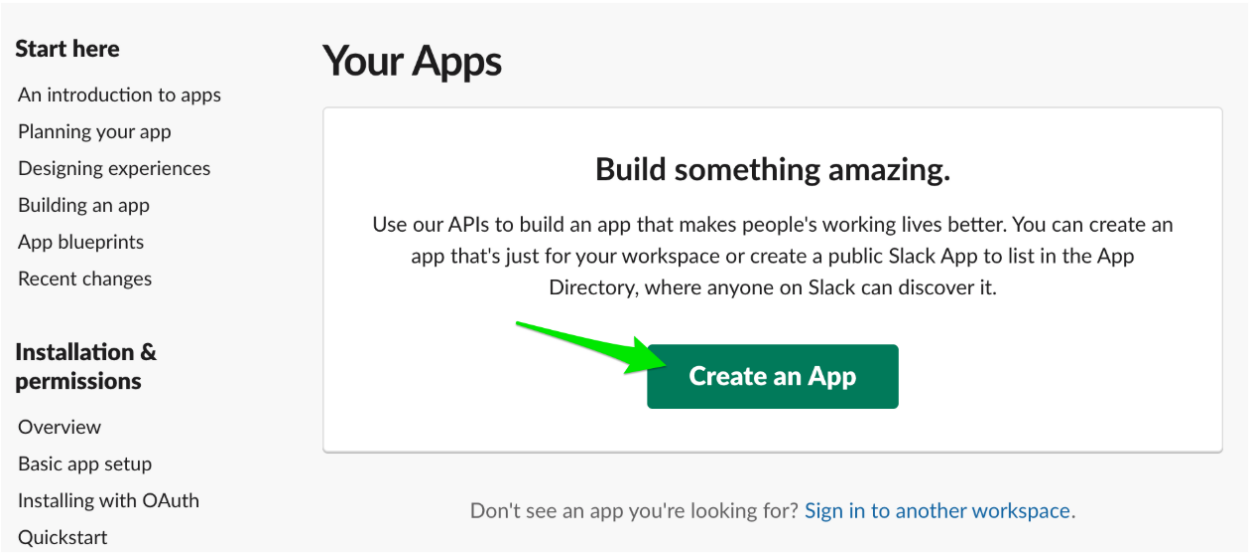
4. Check your email to find the URL of your slack as shown in the figure below.



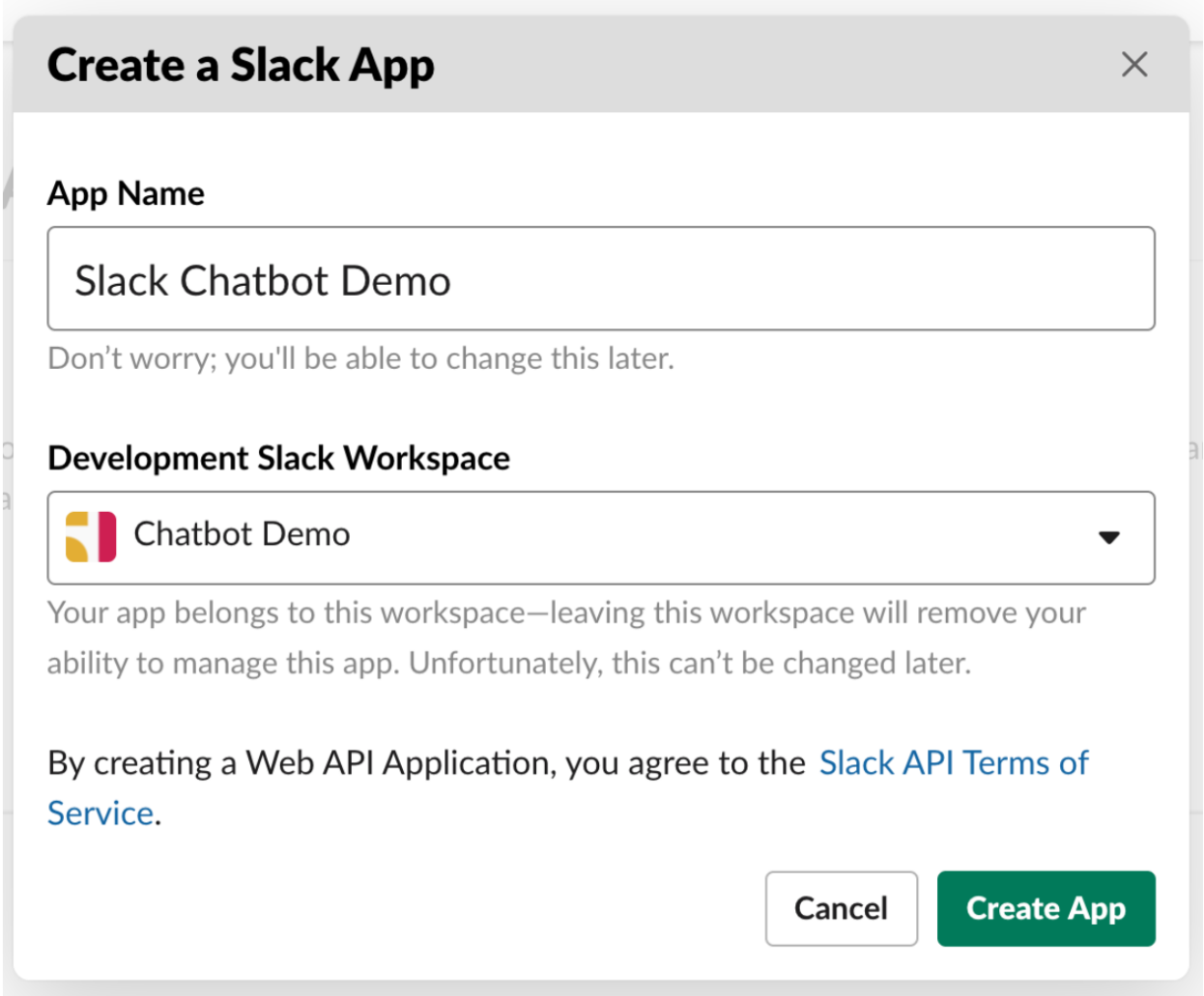
5. Sign into your slack workspace with the following link: https://slack.com/signin?redir=/api/apps. Enter your URL (minus the .slack.com part) and click *Continue*.



6. Once you are signed in, click on *Create an App*.



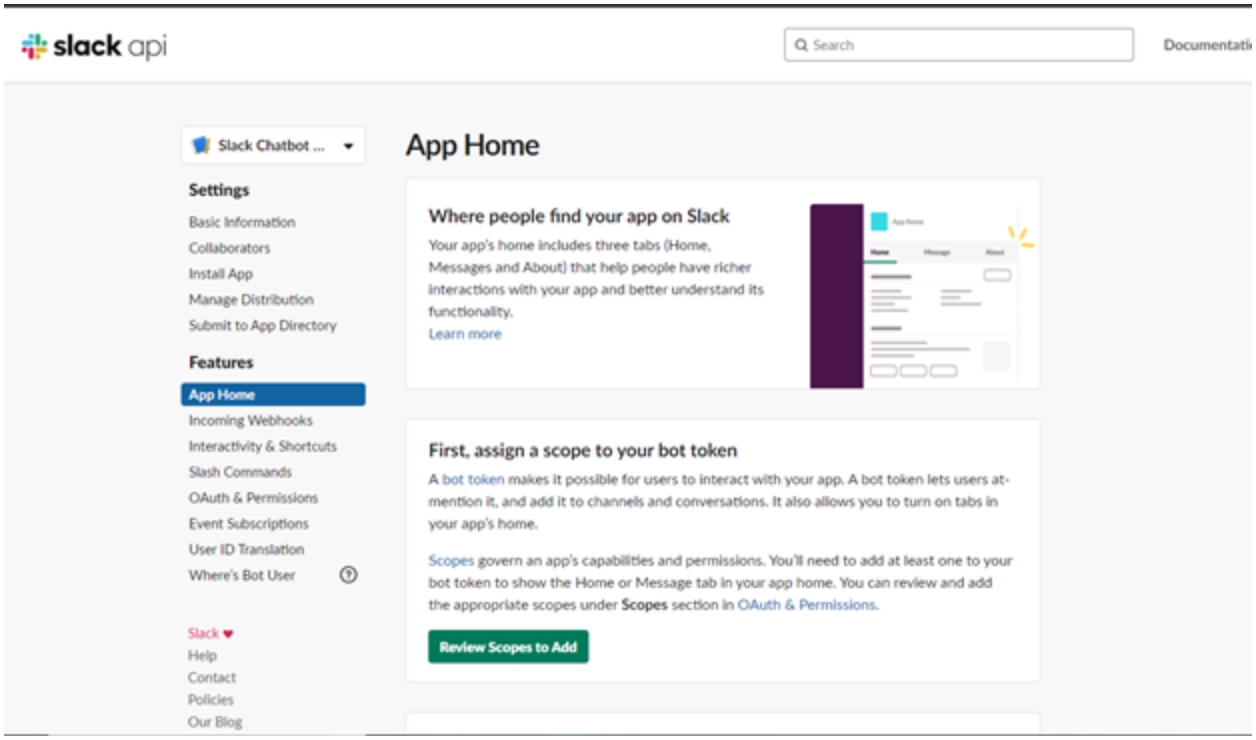
7. Give your app a name and specify the workspace that you want to deploy this app on. Then click on *Create* App.



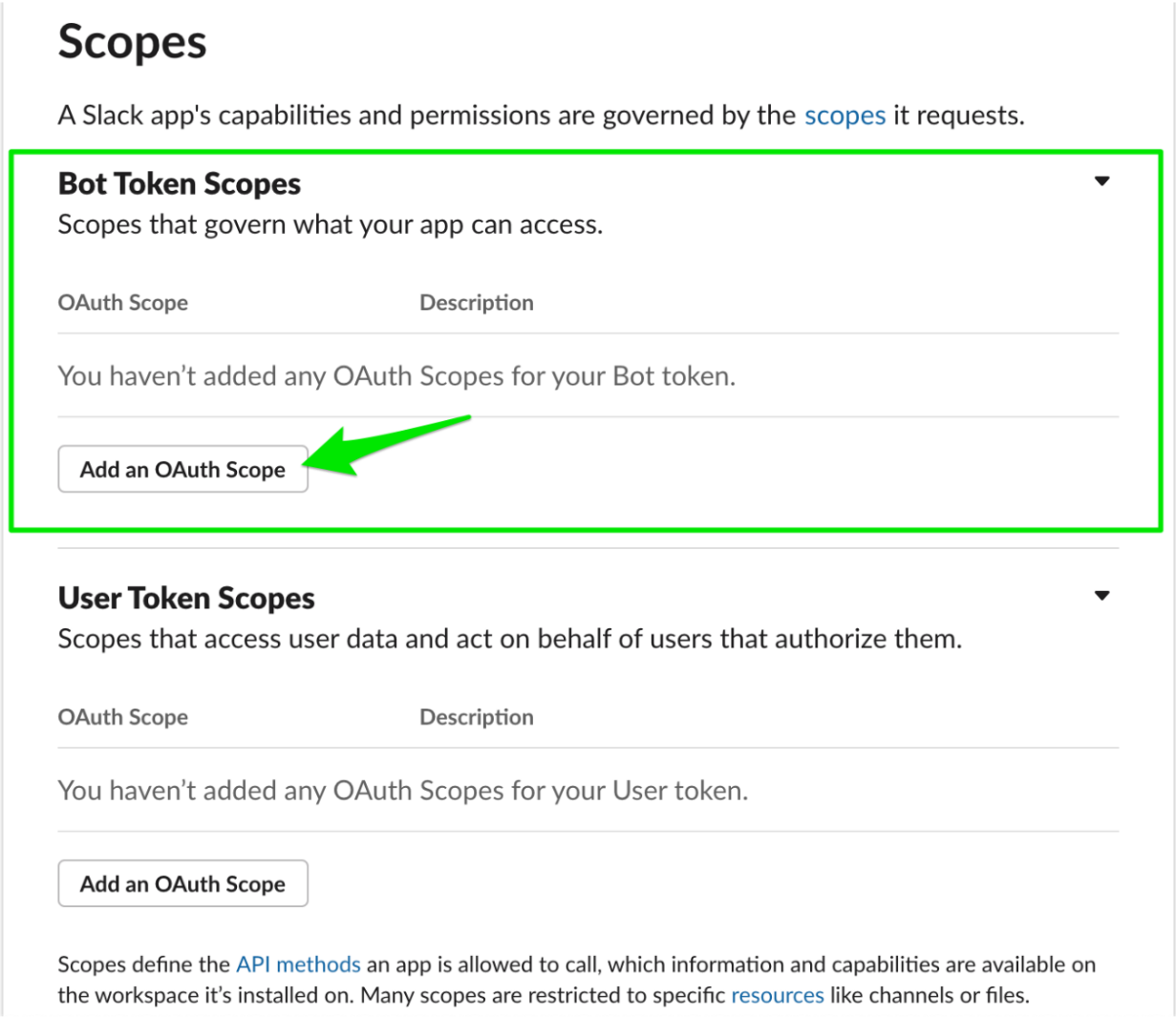
ADD PERMISSIONS

Now that we have a Slack App (e.g., *Slack Chatbot Demo*), we need to ensure that the chatbot will have the right permission.

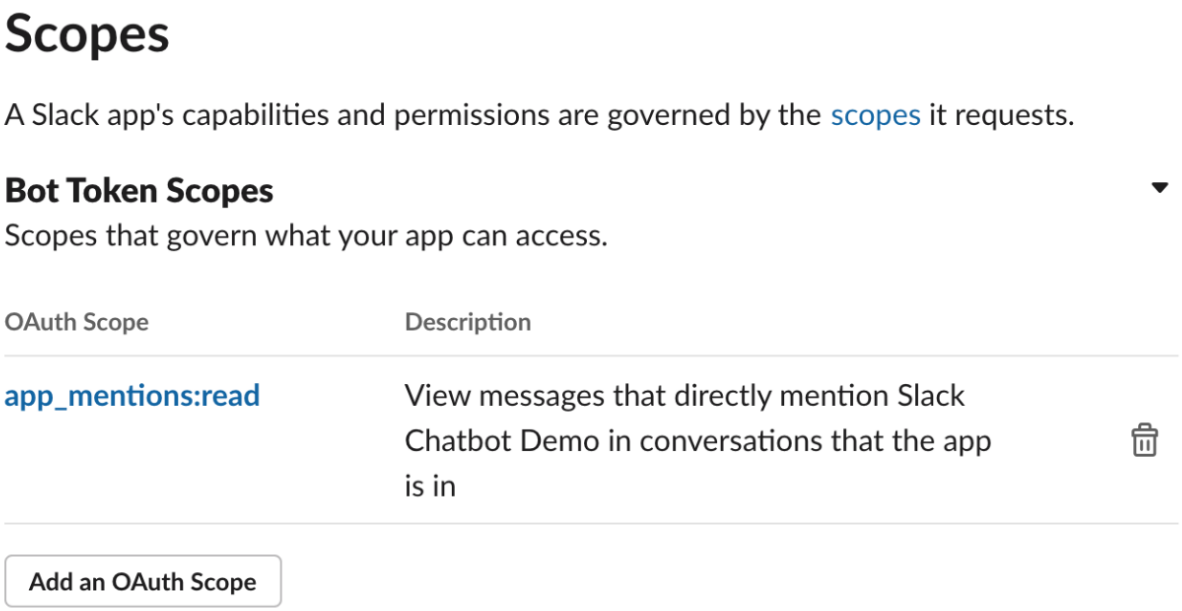
1. Click on *App Home* in the sidebar and then scroll to the *Scopes* section. Click on Review Scopes to Add.



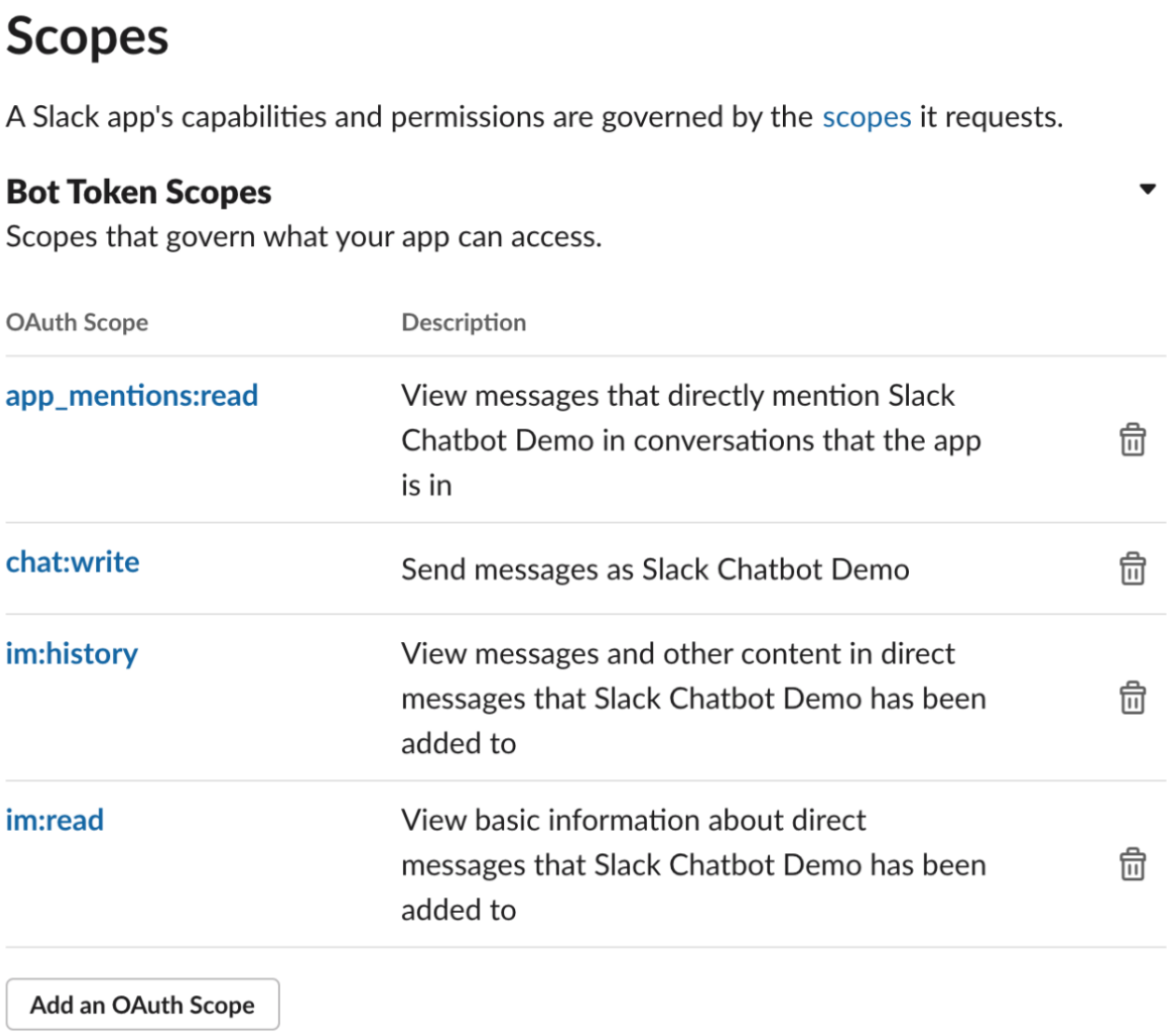
2. Scroll to scopes section. You'll want to click on *Add OAuth Scope* button under *Bot Token Scopes* as shown in figure.



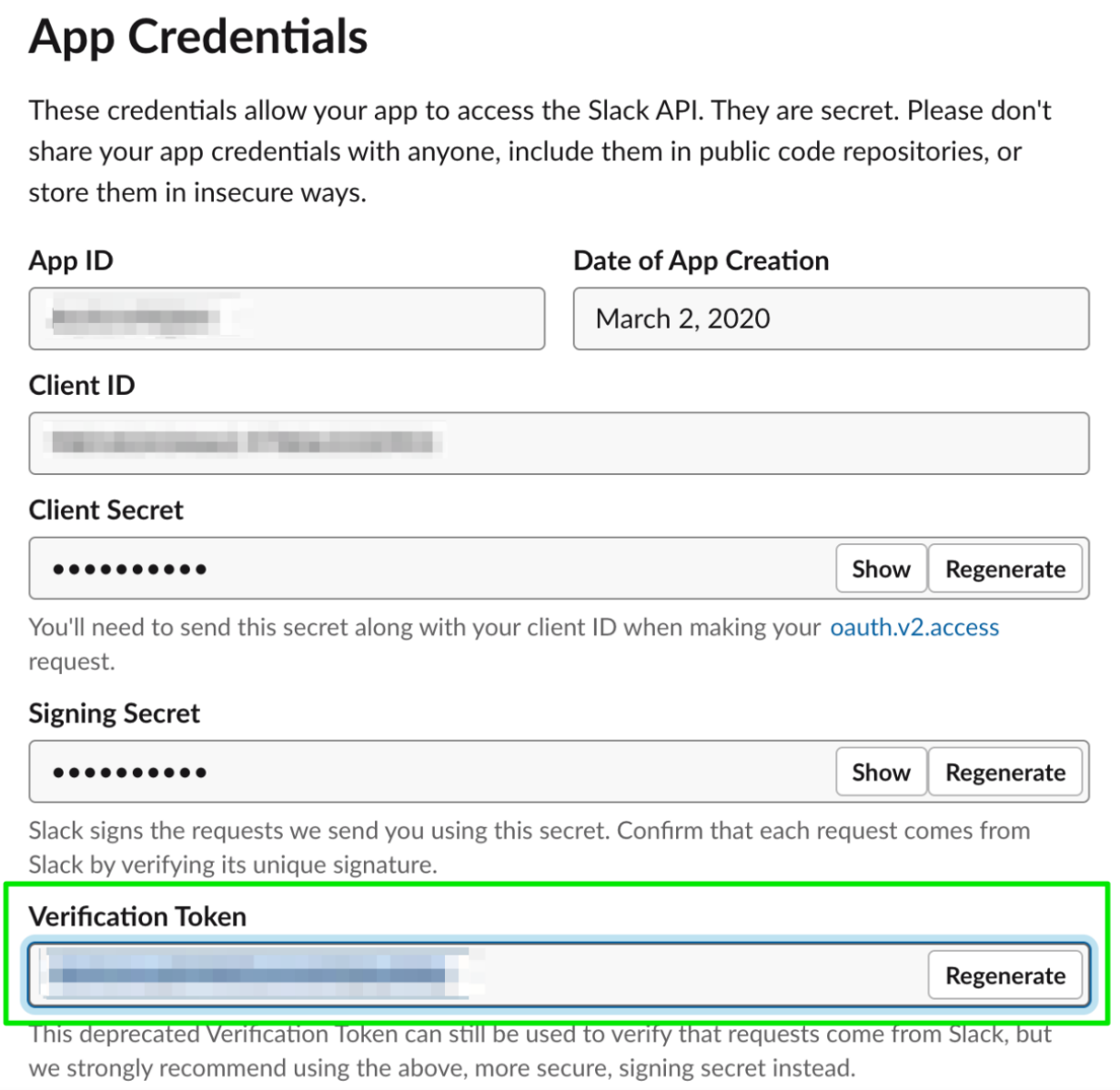
3. Select *app_mentions:read*. The new scope will be added as shown in the figure below. This permission enables the bot to detect mentions from Slack users.



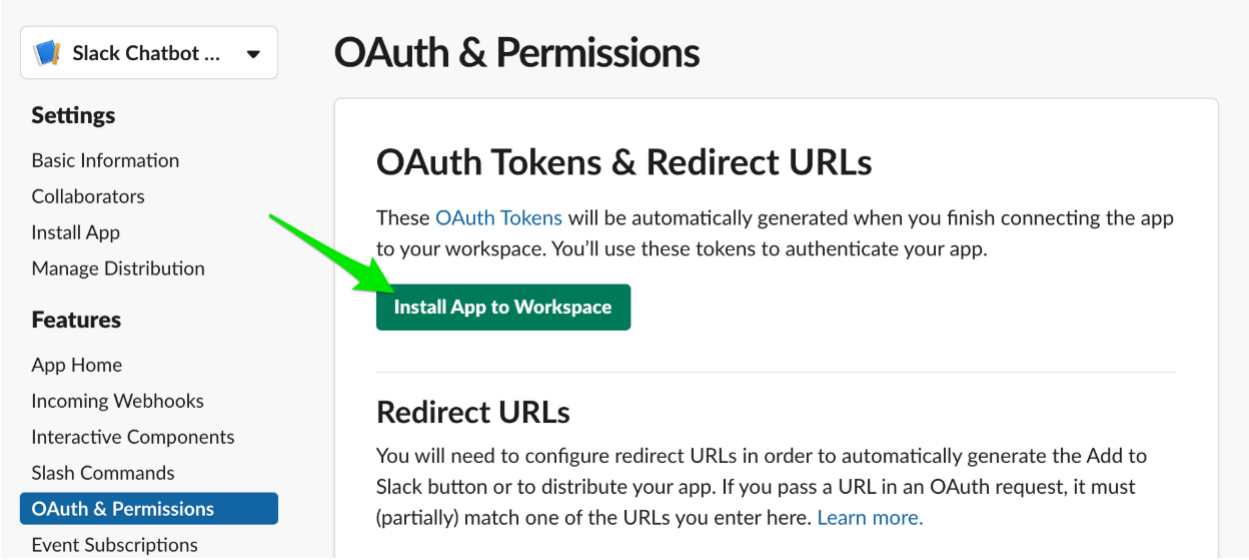
4. Repeat the process of clicking on *Add an OAuth Scope* under *Bot Token Scopes*, to add the following permissions: *chat:write*, *im:history*, and *im:read*. Once you are done, the list will be shown as seen in the image below.



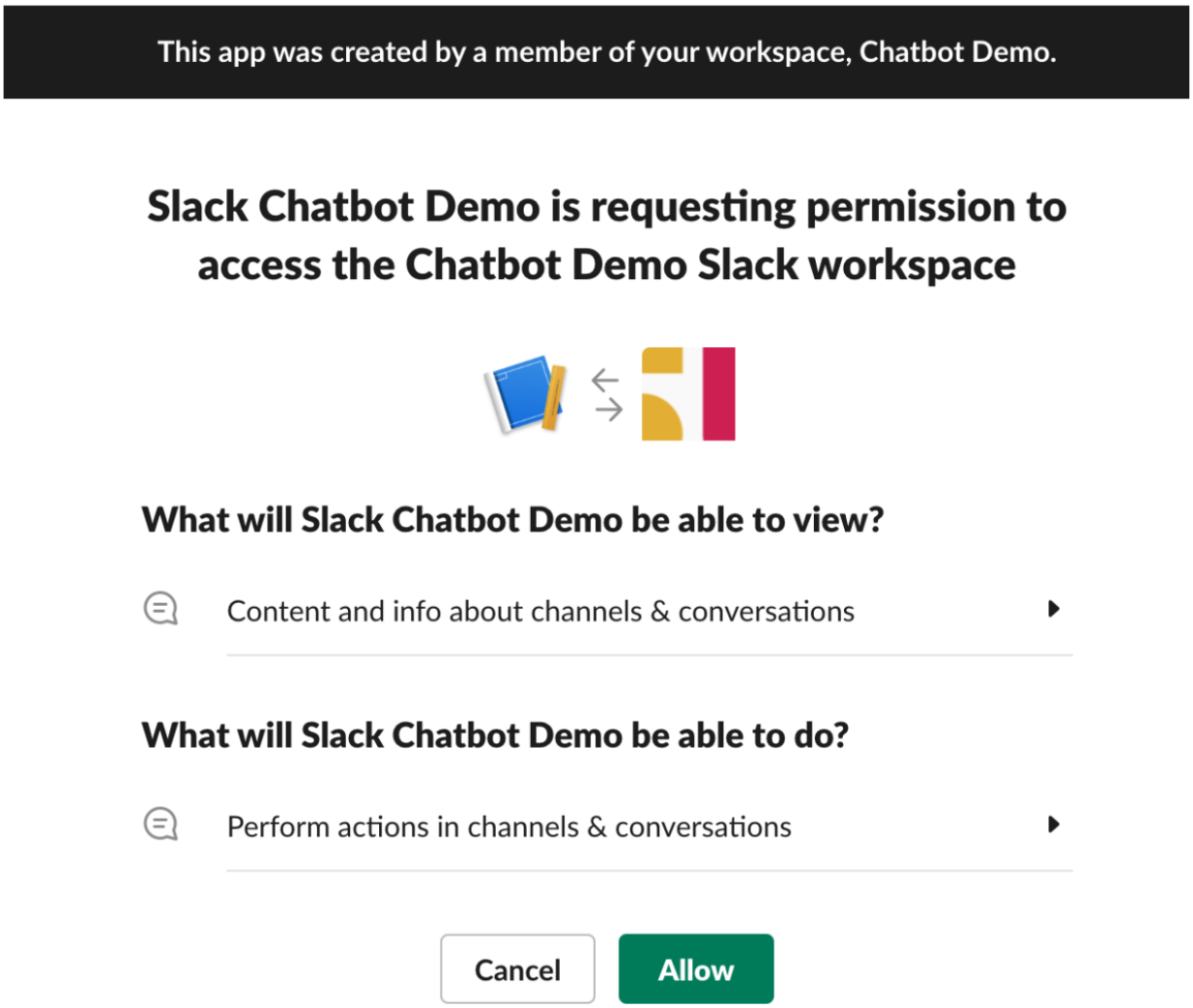
5. Click on *Basic Information* in the left sidebar. Then copy and make note of your *Verification Token*. We'll need this information when integrating Slack within Watson Assistant.



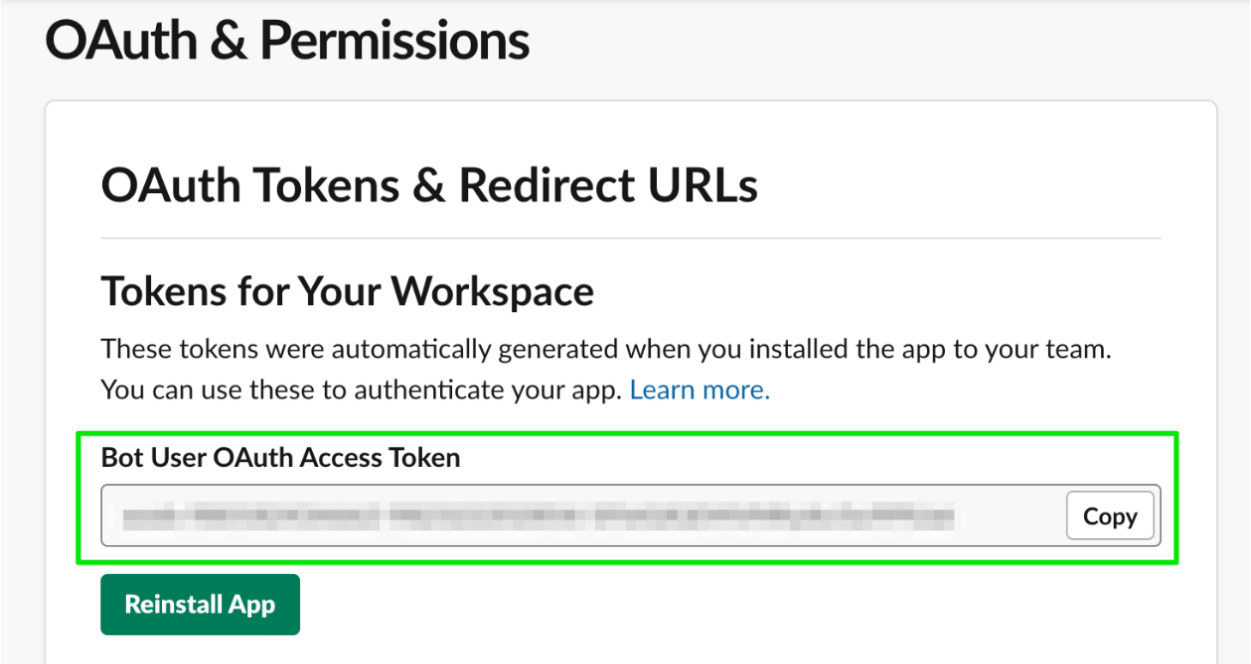
6. Now we need to install our Slack app within our workspace. Click on *OAuth & Permissions* in the left sidebar, then click on *Install App to Workspace*.



7. *Click on Allow* when asked to confirm your authorization.



8. Copy and make note of the *Bot User OAuth Access Token* that will be provided to you.



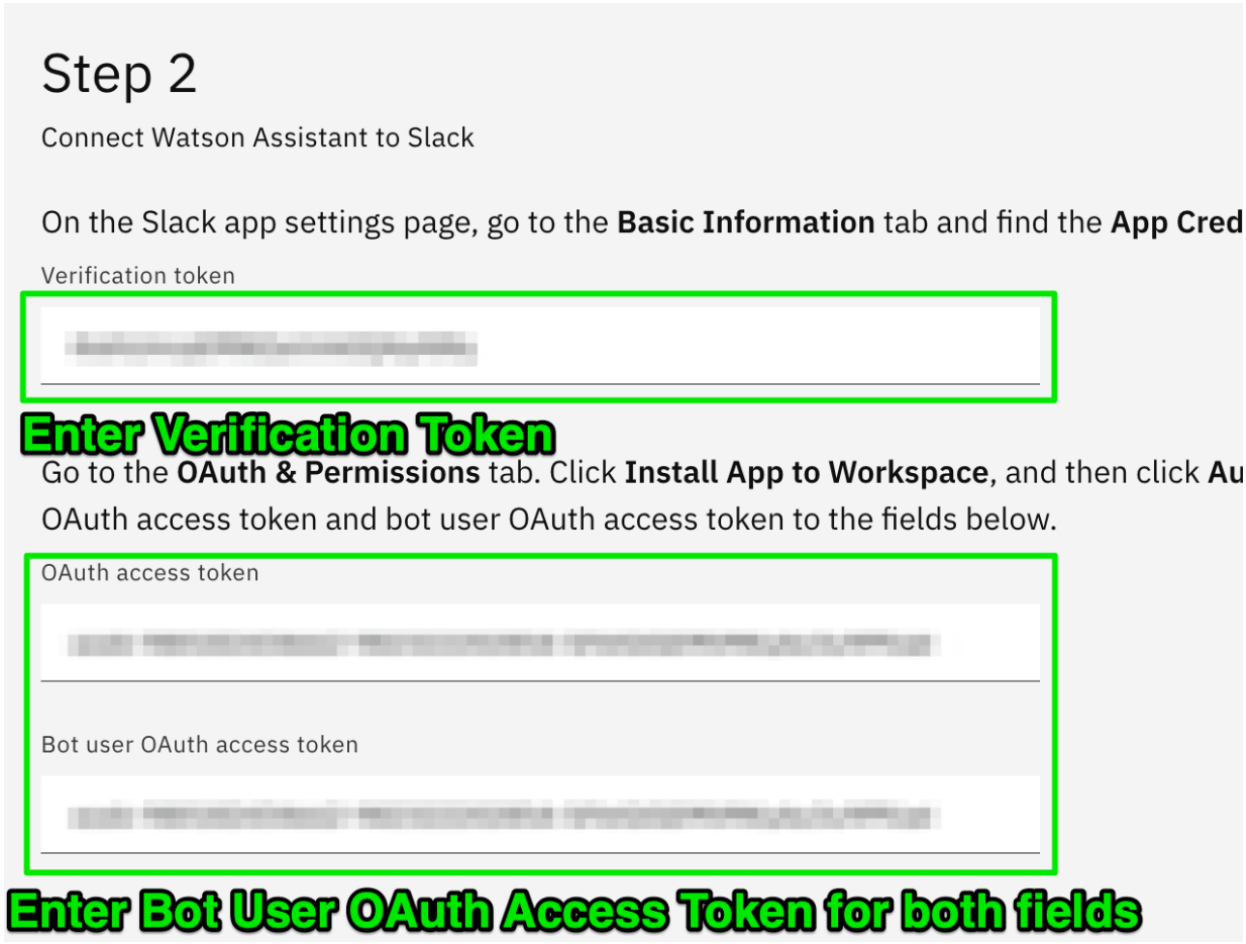
INTEGRATE SLACK WITHIN WATSON ASSISTANT

Now that we have the *Verification Token*, and *Bot User OAuth Access Token* for our Slack app, we can proceed to add a Watson Assistant integration for Slack.

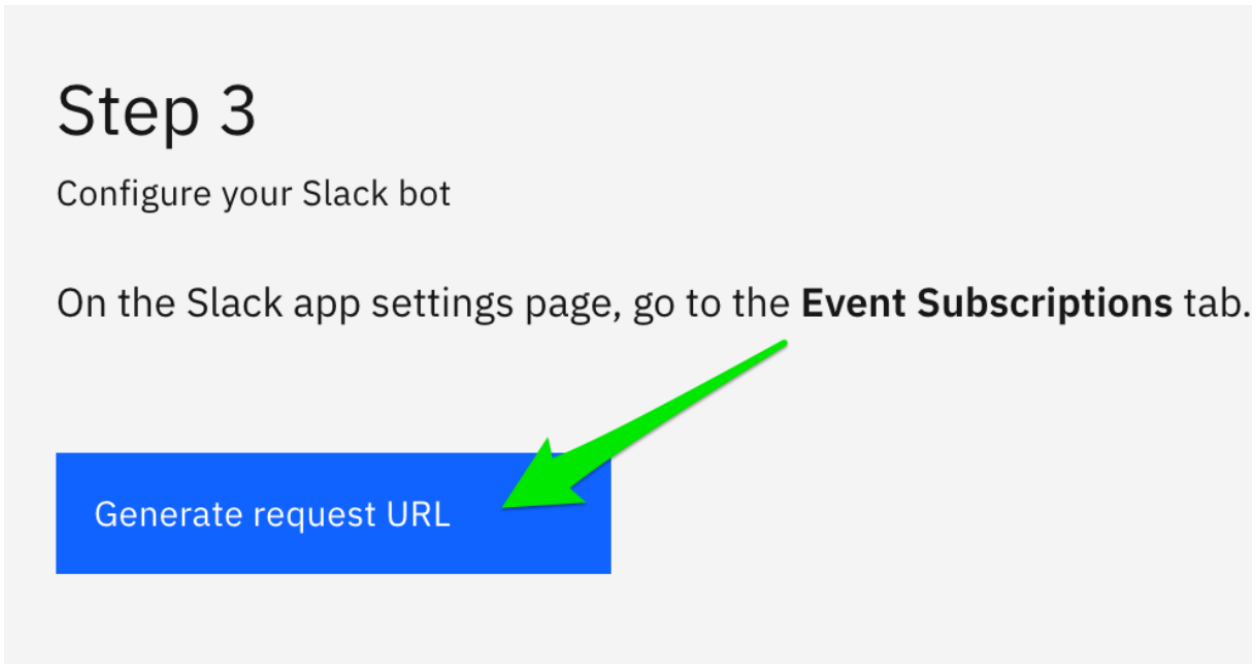
1. From the Student Advisor Chatbot assistant within Watson Assistant, click on *Add integration* - much like we did in the previous lab - but this time select *Slack*.

2. Scroll to Step 2 on the page, and paste your *Verification Token* in the *Verification token* input field.

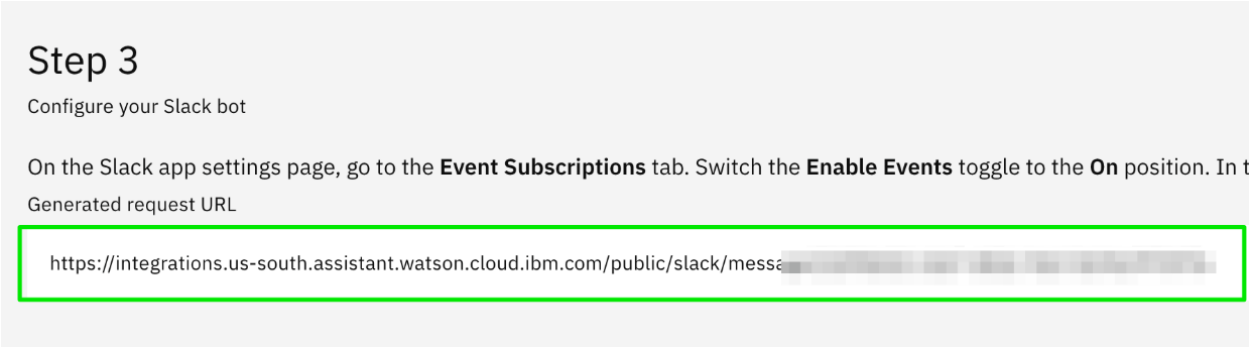
3. Paste your *Bot User OAuth Access Token* (that you made note of, earlier) in both the *OAuth access token* and *Bot user OAuth access token* input fields.



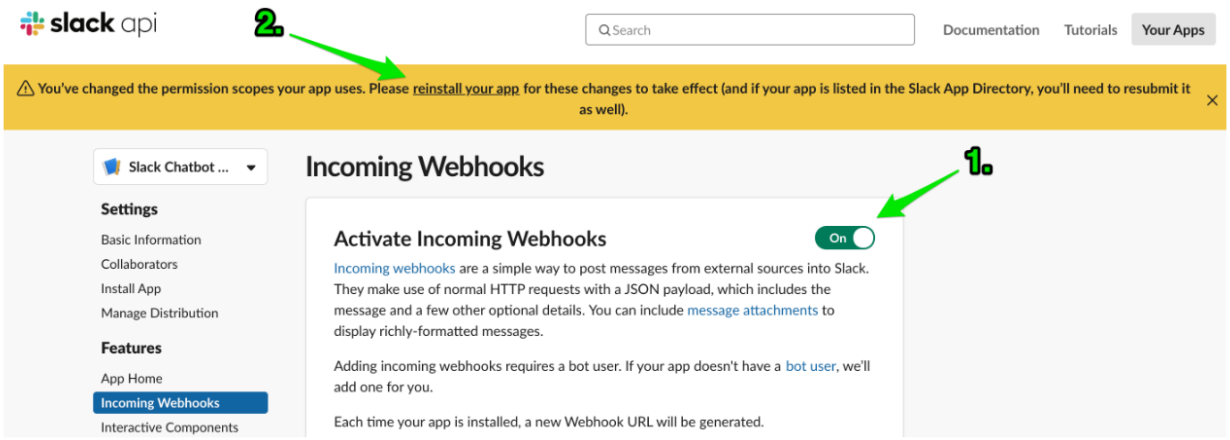
4. Scroll to Step 3 on the page, and click on the *Generate request URL* button.



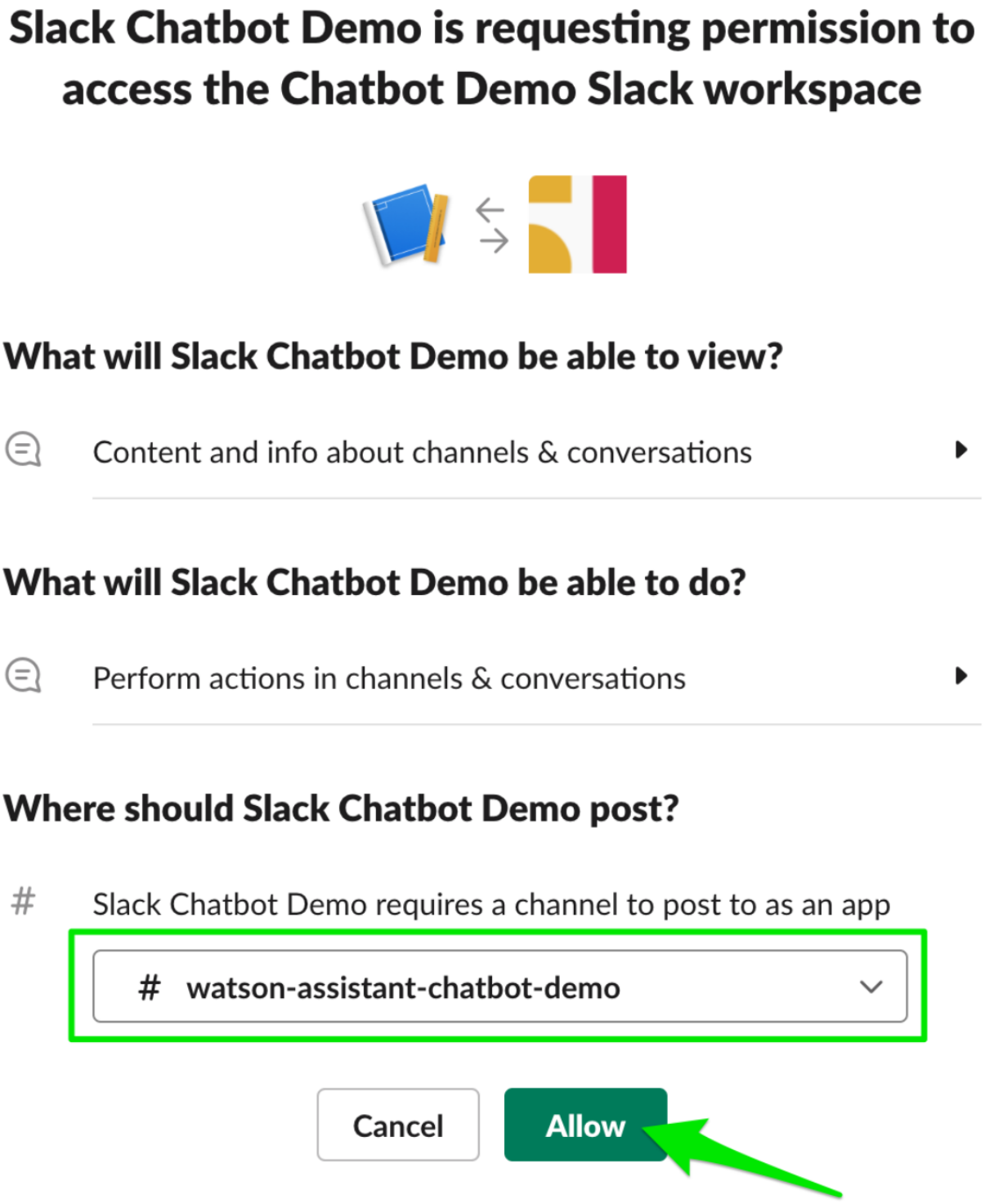
5. This will generate a URL that we'll need to provide to Slack. (You may need to scroll back to Step 3 if the page refreshed.) Copy and make note of the *Generated request URL*.



6. Go back to the page for your Slack App. If you closed it, you can find it at the following URL https://api.slack.com/apps. In the left sidebar, under the *Features* section click on *Incoming Webhooks*. Turn *Activate Incoming Webhooks* on. In the warning message that appears, click on *reinstall your app*.

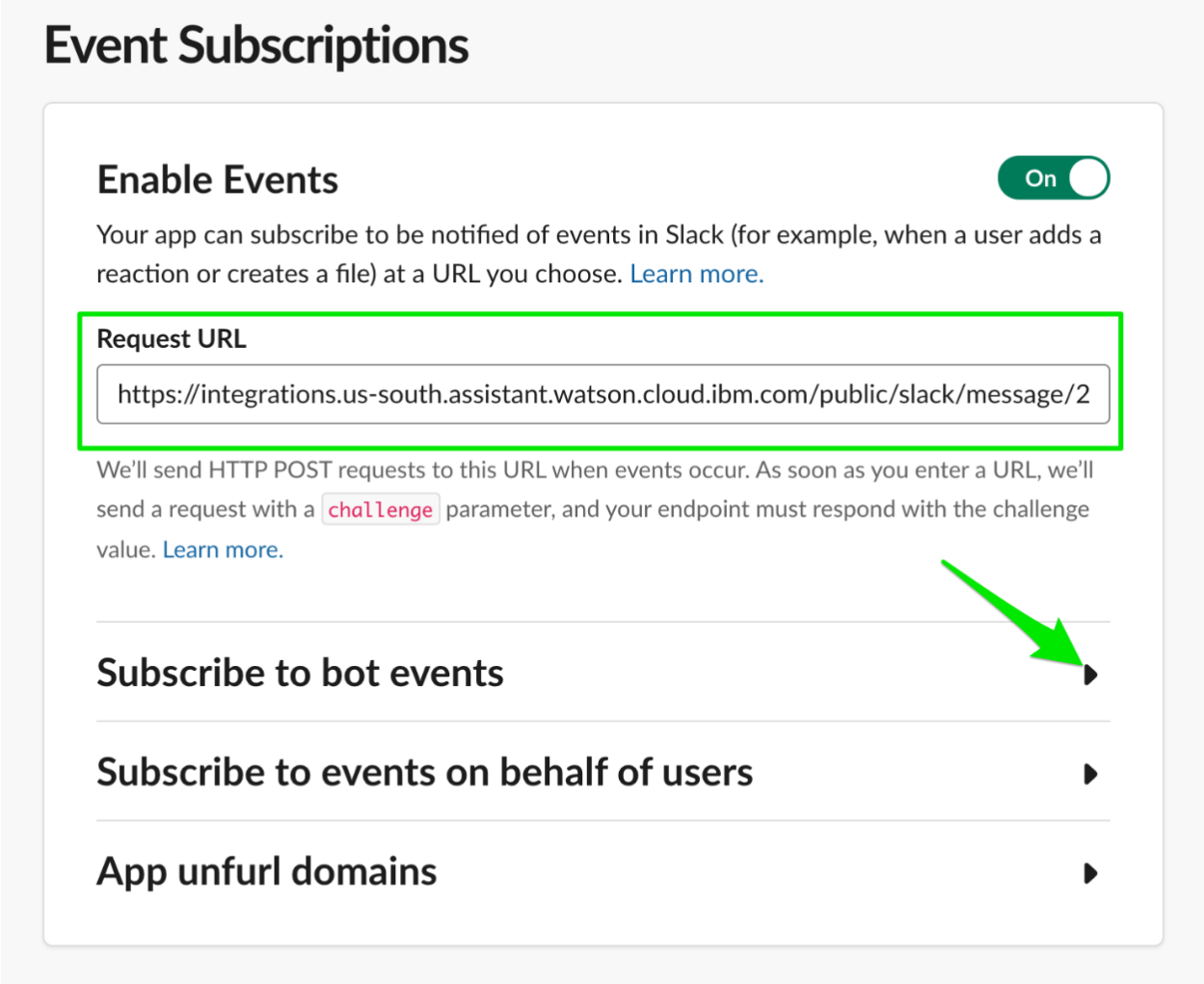


7. You'll be asked to specify a channel (e.g., *#watson-assistant-chatbot-demo*). Select the one that was generated for you earlier on when you specified the project name.



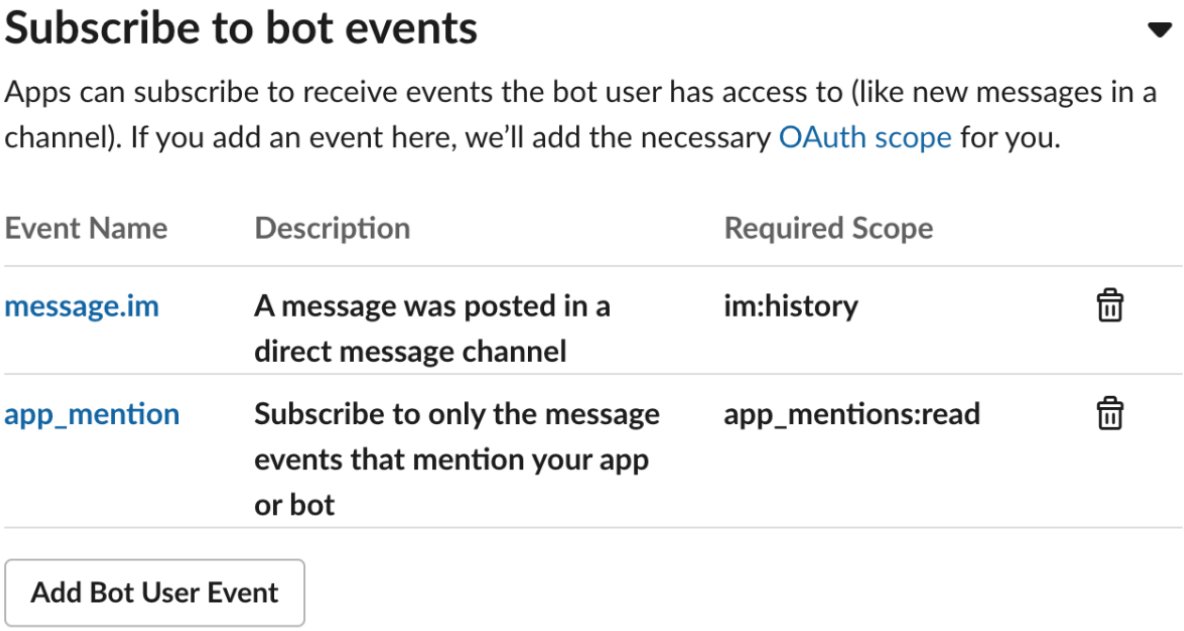
8. Now click on *Event Subscriptions* in the sidebar and switch on *Enable Events*. Then paste the request URL you saved from Watson Assistant.

9. Next, click on *Subscribe to bot events*, as shown by the arrow in the figure below.



This will open the section that will allow you to subscribe to specific events.

10. Click on Add Bot User Event and select the event type message.im. Then repeat the process to add app_mention. This will enable both direct messages and mentions to trigger a response from the chatbot.



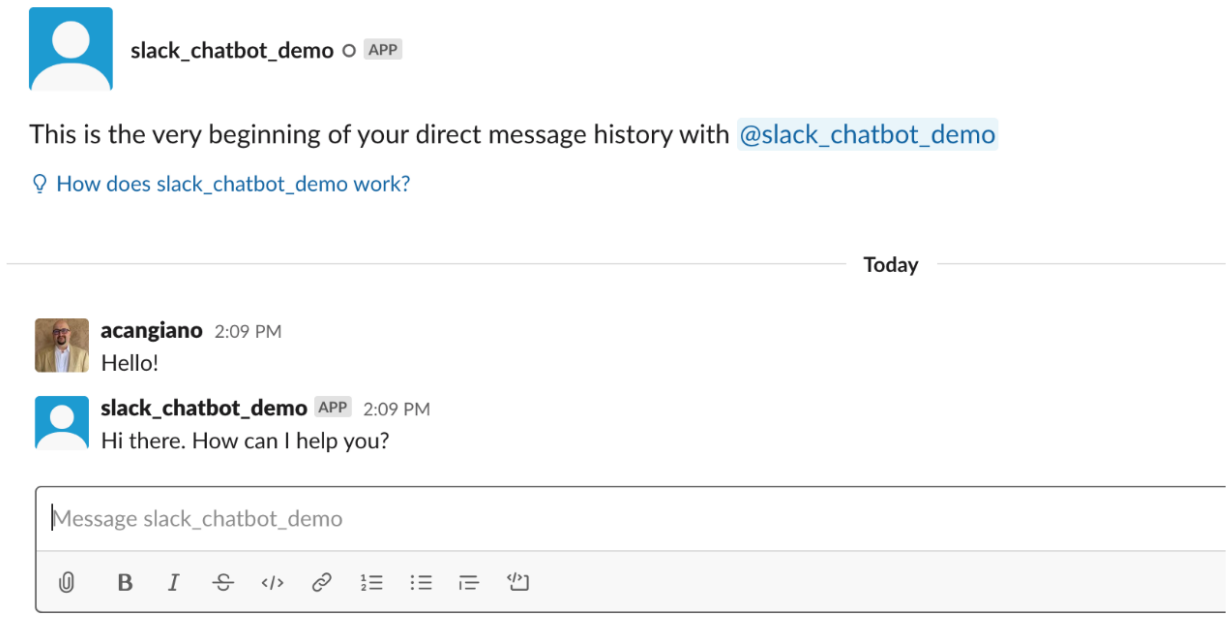
Then click on Save Changes.

(If the Save Changes is disabled/grey, please paste the Request URL again so that Slack can validate the URL and turn the Save Changes button to enabled/green.)

You're done! Your bot will now be available within your Slack workspace. Any direct message or mention will be sent in input to your Watson Assistant chatbot, and its response provided back to the user in Slack.

(Optional) Test your Slack App

If you are curious to see it in action, simply login into your Slack workspace, click on *Direct Messages*, and select your Slack app (e.g., *slack_chatbot_demo*). Try to say hi and you should see a response from the chatbot.



The chatbot will also respond to mentions in channels, provided you invite it to the given channel, after mentioning it as shown in the image below.

acangiano 12:58 PM

added an integration to this channel: Slack Chatbot Demo

acangiano 2:05 PM

@slack_chatbot_demo Hello!

Only visible to you

Slackbot 2:05 PM

OK! I've invited @slack_chatbot_demo to this channel.

slack_chatbot_demo APP 2:05 PM

was added to #watson-assistant-chatbot-demo by acangiano.

acangiano 2:12 PM

@slack_chatbot_demo Hey there!

slack_chatbot_demo APP 2:12 PM

Hi there. How can I help you?

Message #watson-assistant-chatbot-demo