

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).

The screenshot shows the IBM Watson Assistant Lite interface. At the top, there is a navigation bar with the text "IBM Watson Assistant Lite" and "Upgrade". Below the navigation bar, there are two main sections: "Skills" and "Intents". The "Skills" section is currently active, indicated by a blue vertical bar on the left. The "Skills" heading is bolded. Below the heading, there is a descriptive text: "Skills contain the training to respond to your customer queries. Add skills to your assistant and then deploy to your channels." A large blue button labeled "Create skill" is centered below this text. In the main content area, there is a card for a skill named "My first skill" (Beta). The card displays the following details:

- TYPE:** Actions – English (US)
- CREATED:** Jun 8, 2021 3:19 PM IST
- UPDATED:** Jun 9, 2021 9:39 AM IST
- LINKED ASSISTANTS (1):** My first assistant

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](#)



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

Name your skill; for example, Account application or Personal banking.

Description (optional)

Language ⓘ

[Create skill](#)

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

The screenshot shows the 'Student Advisor' interface with the 'Content Catalog' tab selected. A message at the top says: 'Get started faster by adding existing intents from the content catalog. These intents are trained on questions that customers commonly ask.' Below is a table of categories:

Category	Description (optional)	Intents	Action
Banking	Basic transactions for a banking use case.	13	Add to skill
Bot Control	Functions that allow navigation within a conversation.	9	Add to skill
Customer Care	Understand and assist customers with information about themselves and your business.	18	Add to skill
eCommerce	Payment, billing, and basic management tasks for orders.	14	Add to skill
General	General conversation topics most users ask.	10	Add to skill
Insurance	Issues related to insurance policies and claims.	12	Add to skill
Mortgage	Get access to an entire mortgage AI system--including search integration, speech models, and more--for free: ibm.biz/mortgage	20	Add to skill
Telco	Questions and issues related to a user's telephony service, device, and plan.	21	Add to skill
Utilities	Help a user with utility emergencies and their utility service.	10	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.

- 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?
- 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.

The screenshot shows the Entity view in Watson Assistant. The entity name is `@professionalcertificate`. The interface includes fields for 'Value name' (with placeholder 'Enter value'), 'Synonyms' (with a dropdown menu), and a button to 'Add synonym...'. A 'Fuzzy Matching' toggle switch is turned on. Below these, there are tabs for 'Dictionary' (selected) and 'Annotation BETA'. The main table lists five entity values with their types and synonyms:

Entity values (5)	Type	Synonyms
<input type="checkbox"/> customer engagement	Synonyms	customer care, customer support
<input type="checkbox"/> data science	Synonyms	ibm, data analytics, data analysis, data engineering
<input type="checkbox"/> entrepreneurship	Synonyms	innovation management, entrepreneur, business development, mba
<input type="checkbox"/> IT support	Synonyms	technical support
<input type="checkbox"/> project management	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.

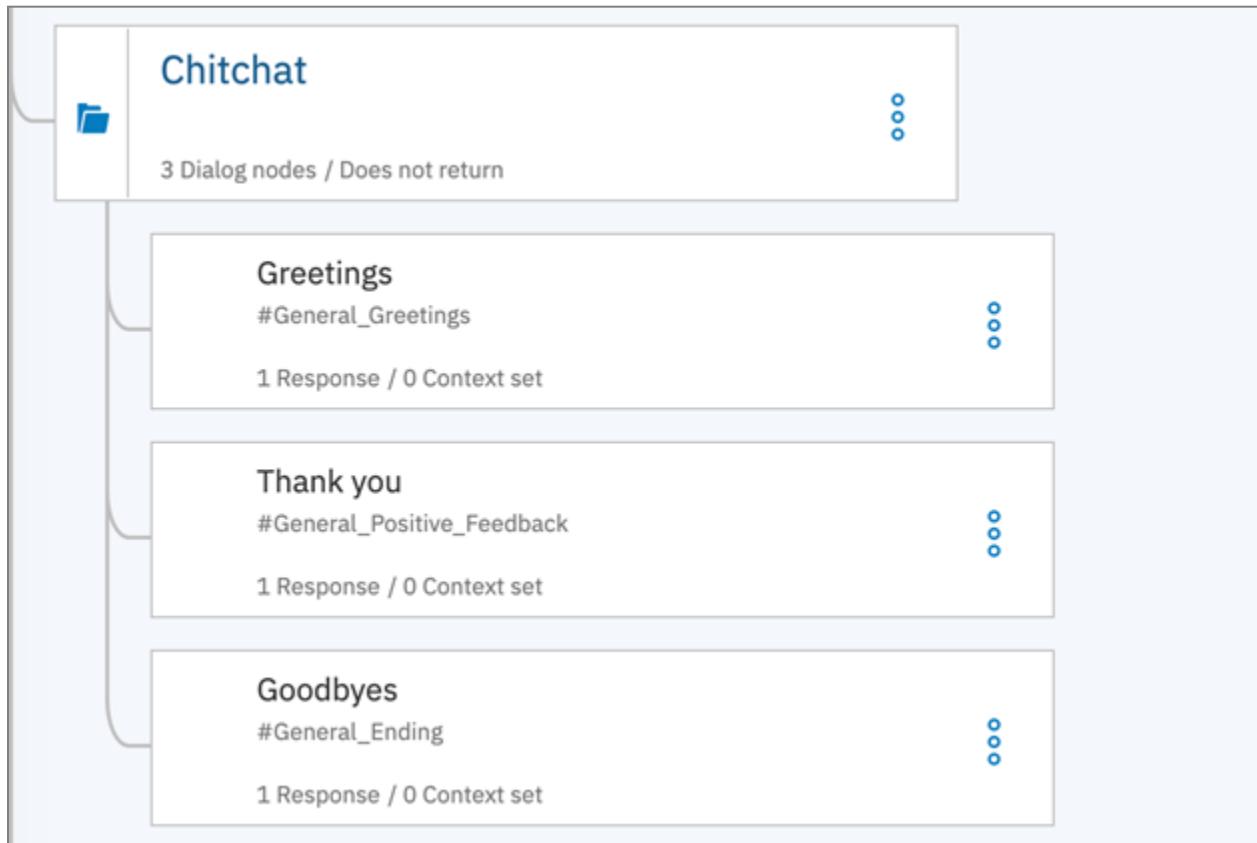
1. From within your skill, click on the *Dialog* tab and create the dialog.

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.

The screenshot shows a chatbot configuration interface. At the top, there is a header "Enrollment Cost" with a "Customize" link and a blue "X" icon. Below this, a section titled "If assistant recognizes:" contains the condition "#Enrollment_Cost" followed by a minus sign (-) and a plus sign (+). A "Text" response type is selected, indicated by a dropdown menu. The response text is: "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." Below the response text, there is a placeholder "Enter response variation". A note at the bottom states: "Response variations are set to sequential. Set to random | multiline ⓘ". At the bottom left is a button "+ Add response type", and at the bottom right is a blue circular icon with a white speech bubble symbol.

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.

Professional Certificates

#Professional_Certificate_Recommendation - +

Then respond with:

If assistant recognizes	Respond with
1 @professionalcertificate:(project management)	We recommend our Applied Project Management Course
2 @professionalcertificate:(IT support)	We recommend Google's IT Support Course
3 @professionalcertificate:(customer experience)	We recommend the IBM Customer Experience Course
4 @professionalcertificate:(data science)	We recommend the IBM Data Science Course
5 @professionalcertificate:entrepreneurship	We recommend the professional certificate in entrepreneurship
6 true	You can see a list of our professional certificates

- 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 successfully. 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.

The screenshot shows the IBM Cloud Functions dashboard. The left sidebar is titled 'Functions' and includes sections for Getting Started, Overview, Pricing, Concepts, CLI, iOS SDK, Documentation, Actions, Triggers, APIs, Monitor, Logs, and Namespace Settings. The 'Overview' section is currently selected. The main content area is titled 'IBM Cloud Functions' and describes it as a 'Functions-as-a-Service (FaaS) platform based on Apache OpenWhisk'. It features a large 'Start Creating' button and a 'Download CLI' button. A 'What's New:' section lists recent changes, including IAM enablement, updated action runtimes, increased maximum memory, and support for monitoring performance. Below this, there are three icons representing different integration points: a bell icon for triggers, a network icon for actions, and a gear icon for monitors. The top right corner of the dashboard shows the organization name 'abcd@gmail.com_lab4' and the location 'Dallas (CF-Based)'.

2. Click connectDiscovery action under lab4 package.

The screenshot shows the IBM Cloud Actions interface. In the top navigation bar, there are links for Catalog, Docs, Support, Manage, and the user's account (@gmail.com_lab4, Dallas (CF-Based)). On the left sidebar, under the 'Actions' section, the 'connectDiscovery' action is listed. The main content area displays the 'Actions' list with one item:

NAME	RUNTIME	WEB ACTION	MEMORY	TIMEOUT
connectDiscovery	Node.js 10	Not Enabled	256 MB	60 s

3. Click Endpoints on the left sidebar.

The screenshot shows the detailed view of the 'connectDiscovery' action. The left sidebar has a 'Endpoints' section selected. The main content area shows the action configuration:

Code (Node.js 10) **Change Input** **Invoke**

Currently Docker, Java (.jar), and Compressed (.zip) Actions are unable to be edited from the IBM Cloud Functions UI
These Actions can still be invoked

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.

- 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

- 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.

Name ▲	Group	Location	Status	Tags
Filter by name or IP address...		Filter by group or org	Filter...	Filter...
> Devices (0)				
> VPC Infrastructure (0)				
> Kubernetes Clusters (0)				
> Cloud Foundry Apps (1)				
> Cloud Foundry Services (0)				
Services (2)				
Discovery-na	Default	Dallas	Provisioned	--
Student Advisor	Default	Dallas	Provisioned	--
> Storage (1)				
> Cloud Foundry Enterprise Environments (0)				
> Apps (0)				

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

The screenshot shows the IBM Watson Assistant interface. At the top, there's a navigation bar with 'IBM Watson Assistant' on the left and 'Cookie Preferences' on the right. Below the navigation bar, the URL 'Skills /' is visible, followed by a search icon, a 'Save new version' button, a 'Try it' button, and a more options menu. The main content area is titled 'Student Advisor'. Below the title, there are tabs: 'Intents', 'Entities', 'Dialog' (which is highlighted with a blue box), 'Options', 'Analytics', 'Versions', and 'Content Catalog'. Under the 'Dialog' tab, there are three buttons: 'Add node' (highlighted with a blue box), 'Add child node', and 'Add folder'. The main pane displays a tree structure of dialog nodes for the 'Student Advisor' skill. The nodes are: 'Welcome' (with sub-node 'welcome'), 'Enrollment Cost' (with sub-node '#Enrollment_Cost'), 'Professional Certificates' (with sub-node '#Professional_Certificate_Recommendation'), 'Courses' (with sub-node '#Course_Recommendation'), and 'Discovery'. Each node has a 'More options' menu icon (three dots) to its right. A blue message icon is located in the bottom right corner of the main pane.

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.

IBM Watson Assistant

Cookie Preferences ? ☰

Skills /

Student Advisor

Intents Entities Dialog Options Analytics Versions Content Catalog

Webhooks

Disambiguation Autocorrection System Entities

Webhooks

A webhook is a mechanism that allows you to call out to an external program based on events in your dialog.

Webhook setup

Specify the request URL for an external API you want to be able to invoke from dialog nodes. Watson will call this URL when configured to do so from a dialog node. [Learn more](#)

URL

`https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/` igmai

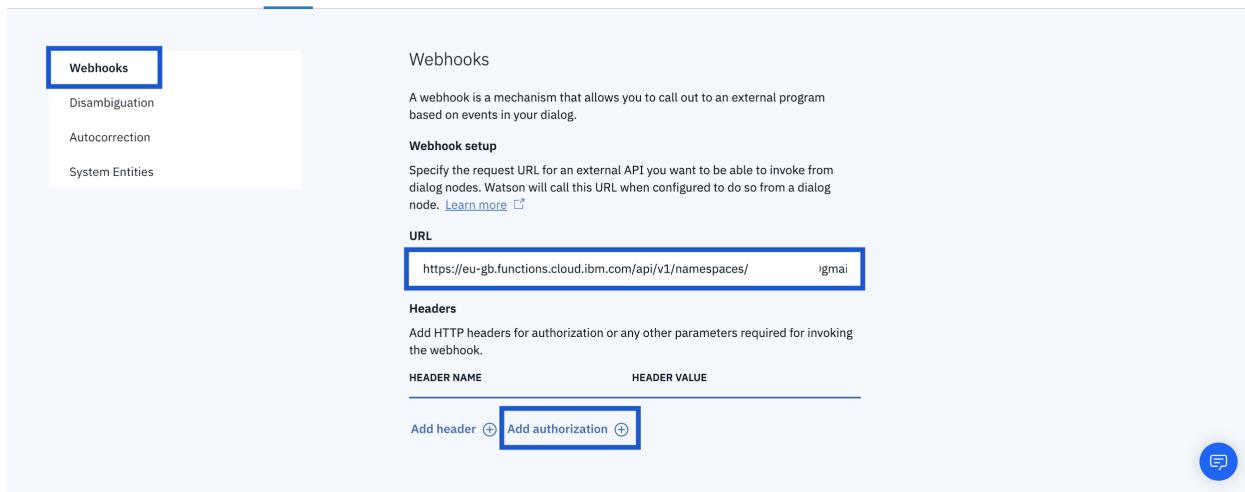
Headers

Add HTTP headers for authorization or any other parameters required for invoking the webhook.

HEADER NAME	HEADER VALUE
Add header +	Add authorization +

Next step

Save new version Try it



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.

IBM Watson Assistant

Cookie Preferences ? ☰

Skills /

Student Advisor

Intents Entities Dialog Options Analytics

Disambiguation Autocorrection System Entities

Basic authorization

User name
Enter user name

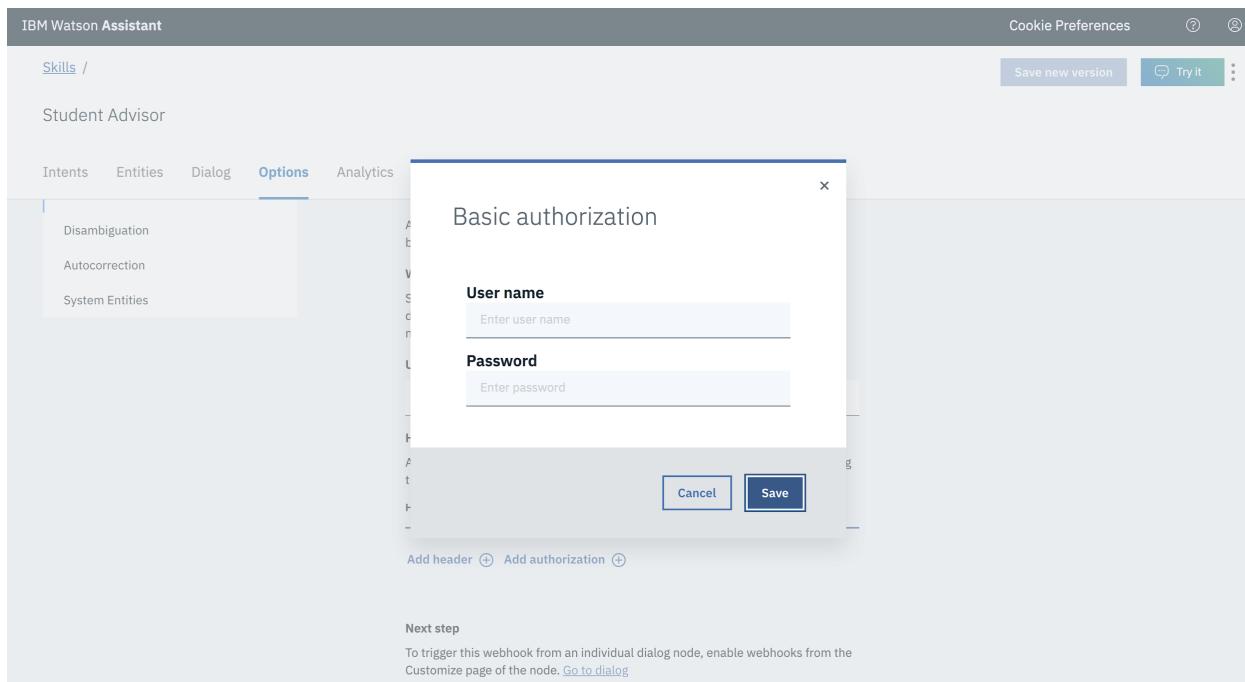
Password
Enter password

Cancel Save

Add header + Add authorization +

Next step

To trigger this webhook from an individual dialog node, enable webhooks from the Customize page of the node. [Go to dialog](#)



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).

IBM Watson Assistant

Cookie Preferences ? @

Skills /

Student Advisor

Intents Entities Dialog Options Analytics Versions Content Catalog

Enrollment Cost
#Enrollment_Cost
1 Responses / 0 Context Set / Does not return

Professional Certificates
#Professional_Certificate_Recommendation
6 Responses / 0 Context Set / Does not return

Courses
#Course_Recommendation
1 Responses / 0 Context Set / Does not return

Chitchat
3 Dialog nodes / Does not return

Anything else
anything_else
1 Responses / 0 Context Set / Does not return

Add child node
Add node above
Add node below
Add folder
Move
Duplicate
Jump to
Delete

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

IBM Watson Assistant

Cookie Preferences ? @

Skills /

Student Advisor

Intents Entities Dialog Options Analytics Versions Content Catalog

Professional Certificates
#Professional_Certificate_Recommendation
6 Responses / 0 Context Set / Does not return

Courses
#Course_Recommendation
1 Responses / 0 Context Set / Does not return

Discovery
true
1 Responses / 0 Context Set / Return allowed

Chitchat
3 Dialog nodes / Does not return

Anything else
anything_else
1 Responses / 0 Context Set / Does not return

Courses

Customize ⚙️ ×

If assistant recognizes:

#Course_Recommendation ✖️ +

Then callout to my webhook: [Learn more](#)

Parameters

KEY	VALUE
Parameter name (ex. Date)	Parameter value (ex. \$date or \$sys-date)

Add parameter +

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`.

IBM Watson Assistant

Cookie Preferences ⚙️

Skills /

Student Advisor

Intents Entities Dialog Options Analytics Versions Content Catalog

Professional Certificates
#Professional_Certificate_Recommendation
6 Responses / 0 Context Set / Does not return

Courses
#Course_Recommendation
1 Responses / 0 Context Set / Does not return

Discovery
true
1 Responses / 0 Context Set / Return allowed

Chitchat
3 Dialog nodes / Does not return

Anything else
anything_else
1 Responses / 0 Context Set / Does not return

Courses

Customize ⚙️ X

Return variable
\$webhook_result_1

Webhook URL Your webhook URL is configured. [Options](#) X

Then respond with

IF ASSISTANT RECOGNIZES RESPOND WITH

1 anything_else Enter a response

Add response +

The screenshot shows the IBM Watson Assistant interface. On the left, a tree view displays several dialog nodes: 'Professional Certificates', 'Courses', 'Discovery', 'Chitchat', and 'Anything else'. The 'Courses' node is currently selected. On the right, a configuration panel for the 'Courses' node is open, showing settings for 'Return variable' (\$webhook_result_1) and 'Webhook URL' (which is configured). Below this, under 'Then respond with', there is a section for 'IF ASSISTANT RECOGNIZES' with a condition 'anything_else' and a placeholder 'Enter a response'.

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

Skills /

Student Advisor

Intents Entities Dialog Options Analytics Versions Content Catalog

Select a destination node (origin: Courses)

#Professional_Certificate_Recommendation
6 Responses / 0 Context Set / Does not return

Courses
#Course_Recommendation
1 Responses / 0 Context Set / Does not return

Discovery
true
1 Responses / 0 Context Set

Chitchat
3 Dialog nodes / Does not return

Anything else
anything_else
1 Responses / 0 Context Set / Does not return

Jump to and...
Wait for user input
If assistant recognizes (condition)
Respond

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 ?>
```

The screenshot shows the IBM Watson Assistant interface. The top navigation bar includes 'Skills / Student Advisor', 'Cookie Preferences', and buttons for 'Save new version', 'Try it', and more. The main area is titled 'Dialog' and shows a tree of dialog nodes:

- Professional Certificates**: #Professional_Certificate_Recommendation, 6 Responses / 0 Context Set / Does not return
- Courses**: #Course_Recommendation, 1 Responses / 0 Context Set / Jump to / Does not return
 - Jump to Discovery (Evaluate condition)**
 - Discovery**: true, 1 Responses / 0 Context Set / Return allowed
- Chitchat**: 3 Dialog nodes / Does not return
- Anything else**: anything_else, 1 Responses / 0 Context Set / Does not return

The 'Discovery' node is selected, and its configuration panel is open on the right. It includes sections for 'If assistant recognizes:' (with a 'true' condition) and 'Then respond with' (a 'Text' field containing an HTML template for course recommendations).

```

<Here are some courses I found for you!>
<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.

The screenshot shows the IBM Watson Assistant interface. On the left, under the 'Dialog' tab, there is a tree view of a dialog flow. The root node is 'Professional Certificates' with condition '#Professional_Certificate_Recommendation'. It has 6 responses and does not return. Below it is a node for 'Courses' with condition '#Course_Recommendation', which has 1 response and does not return. A 'Jump to Discovery (Evaluate condition)' node follows. This leads to a 'Discovery' node with condition 'true', which has 1 response and allows context return. The flow continues through 'Chitchat' (3 dialog nodes, does not return) and 'Anything else' (anything_else, does not return). On the right, a 'Try it out' panel shows a conversation: 'Hello. I'm a Student Advisor Chatbot. I can help you with your questions about our site and give you course recommendations.' followed by a message input field containing 'recommend me a course on database' and a dropdown menu showing '#Course_Recommendation'. Below this, a response is shown: 'Here are some courses I found for you! Distributed Database Systems The increased capabilities of a collection of logically inter-related databases distributed over a computer network enable scalable data processing. NoSQL Database Systems Unlike traditional relational database management systems, NoSQL databases are capable of storing unstructured data. Databases and SQL for Data Science Much of the world's data resides in databases'. At the bottom of the panel is a text input field with placeholder 'Enter something to test your virtual assistant' and a note 'Use the up key for most recent'.

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.

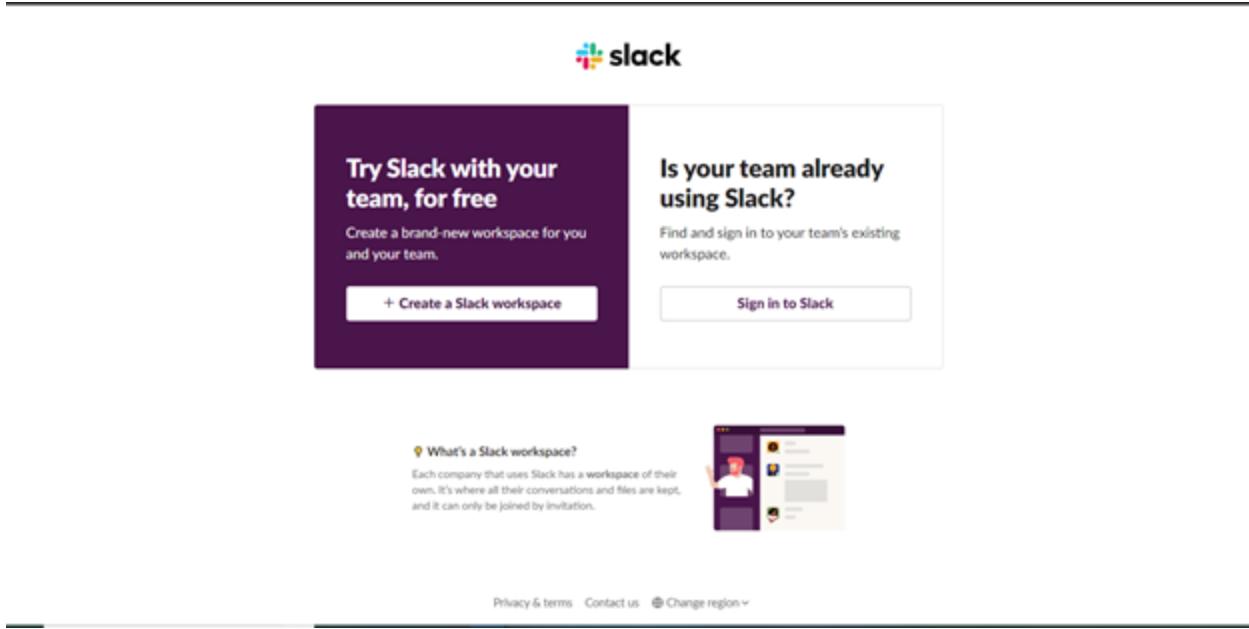
Deploying Your Chatbot On Slack

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](https://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?

I.e. 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

Click at least two Skip this step

The image shows a screenshot of the Slack onboarding process. At the top left, there's a dark sidebar with the title "Chatbot-Demo" and a user icon for "Anamika". Below this, under "Channels", are two listed: "# Watson-assistant-chatbot" and "# Ideas". On the right side, the main content area has a header "Step 2 of 3" and a large title "What would you like to use Slack for first?". Below the title, a sub-instruction reads: "Choose a few. We'll use these to help you get up and running in Slack." There are five circular buttons for selecting purposes: "Working on a project" (red), "Staying connected" (blue), "Sharing ideas" (green), "Making announcements" (orange), and "Replacing a meeting" (purple). A note below the buttons says: "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." At the bottom, there are two buttons: a dark blue "Next" button on the left and a smaller "Skip this step" link on the right.

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

The image consists of two screenshots of the Slack application interface.

The top screenshot shows the "Add team members" step of a workspace setup. It features a dark purple sidebar with the workspace name "Chatbot-Demo" and one member, "Aramika". The main area is titled "Who would you like to use Slack with?" and contains a text input field with "Example ellis@gmail.com" and a "+" button. Below the input field are links for "Paste many at once" and "Share an invitation link". At the bottom are two buttons: "Add teammates" and "Skip this step".

The bottom screenshot shows the Slack workspace interface. The sidebar lists channels: "# ideas", "# watson-assistant-chatbot" (which is selected and highlighted in blue), and "# welcome". It also shows "People" and "Files" sections. The main area displays the "# watson-assistant-chatbot" channel. The channel header says "# watson-assistant-chatbot" and describes it as "This channel is for working on a project. Hold meetings, share docs and make decisions together with your team." Below the header are buttons for "Share a project file" and "Share daily updates". A message from user "Aramika" is shown: "joined #watson-assistant-chatbot". The message input field at the bottom has placeholder text "Message #watson-assistant-chatbot" and various message formatting icons. The top right corner of the interface includes a search bar, a user icon, and a "Skip tutorials" link.

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.



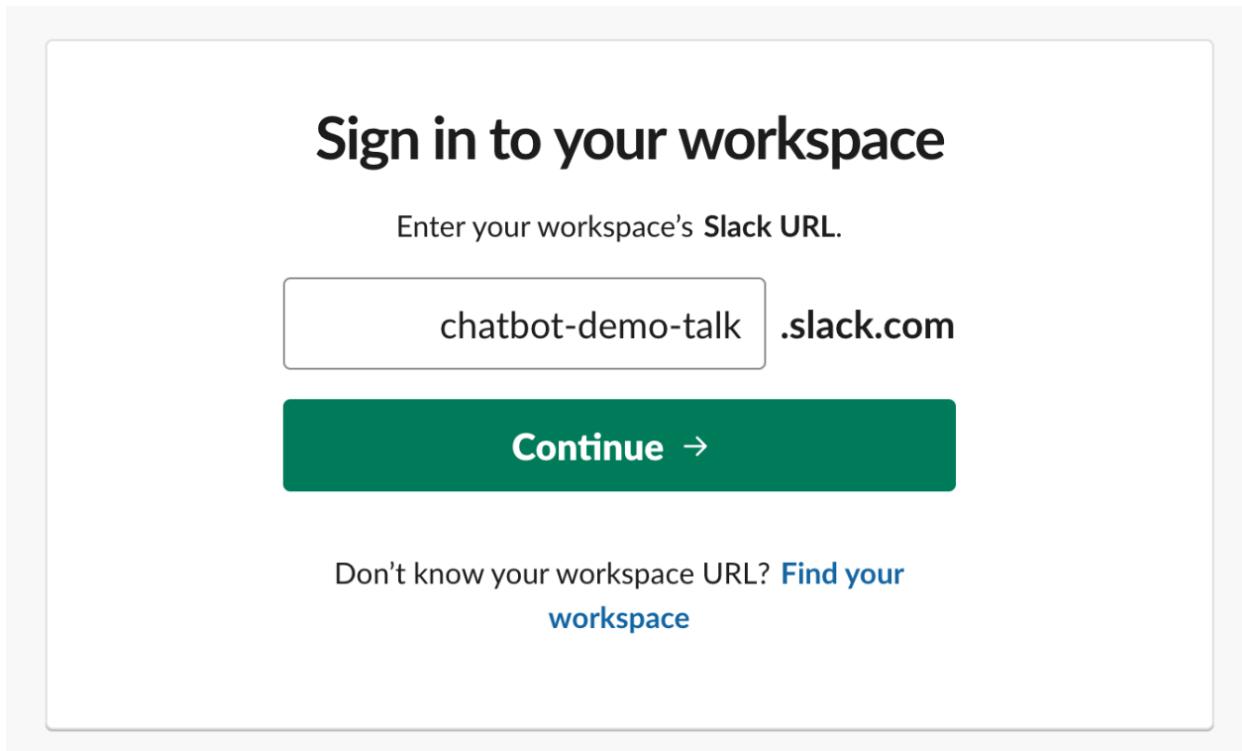
Welcome to Slack!

You've created the new Slack workspace **Chatbot Demo**. Here are your account details:

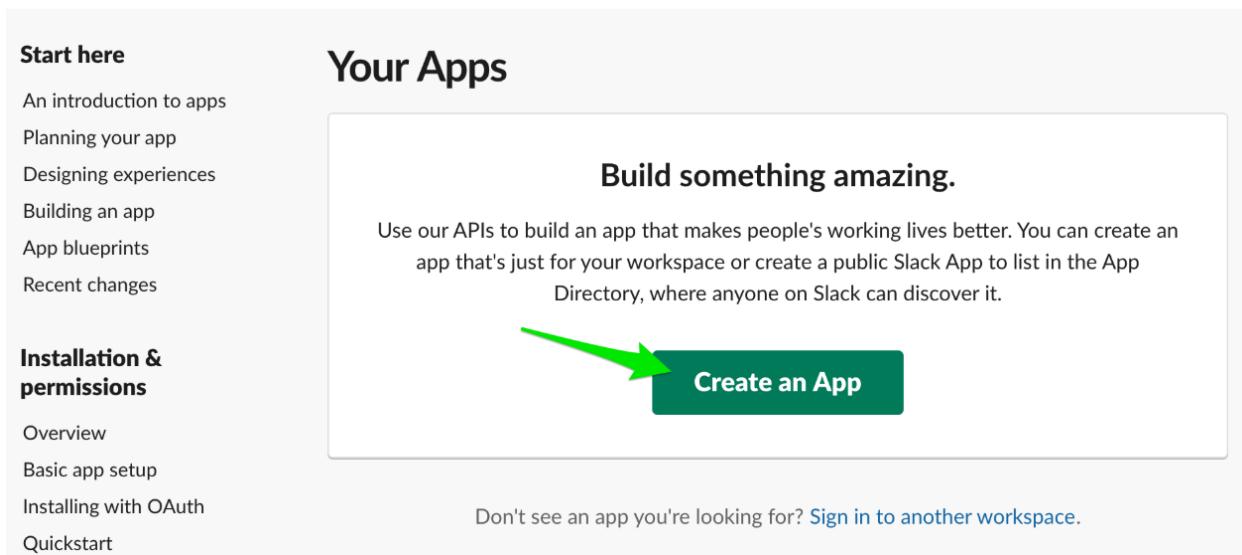

Chatbot Demo
URL: chatbot-demo-talk.slack.com
Email: acangiano@gmail.com

Your workspace is on the **free plan**, with unlimited messaging and the ability to search the 10,000 most recent messages.

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*.



The screenshot shows the "Your Apps" section of the Slack developer portal. On the left, there's a sidebar with "Start here" and "Installation & permissions" sections. The main area has a heading "Build something amazing." with a descriptive paragraph about using APIs to build apps. A prominent green "Create an App" button is at the bottom right of this section. A green arrow points from the top of the image towards this button.

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

Create a Slack App

X

App Name

Slack Chatbot Demo

Don't worry; you'll be able to change this later.

Development Slack Workspace

 Chatbot Demo ▾

Your app belongs to this workspace—leaving this workspace will remove your ability to manage this app. Unfortunately, this can't be changed later.

By creating a Web API Application, you agree to the [Slack API Terms of Service](#).

Cancel

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*.

slack api

Search Documentation

Slack Chatbot ... ▾

App Home

Settings

- Basic Information
- Collaborators
- Install App
- Manage Distribution
- Submit to App Directory

Features

- App Home**
- Incoming Webhooks
- Interactivity & Shortcuts
- Slash Commands
- OAuth & Permissions
- Event Subscriptions
- User ID Translation
- Where's Bot User

Slack ❤️

Help

Contact

Policies

Our Blog

Where people find your app on Slack

Your app's home includes three tabs (Home, Messages and About) that help people have richer interactions with your app and better understand its functionality.

Learn more

App Home

Home Messages About

First, assign a scope to your bot token

A bot token makes it possible for users to interact with your app. A bot token lets users @-mention it, and add it to channels and conversations. It also allows you to turn on tabs in your app's home.

Scopes govern an app's capabilities and permissions. You'll need to add at least one to your bot token to show the Home or Message tab in your app home. You can review and add the appropriate scopes under Scopes section in OAuth & Permissions.

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.

Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description
You haven't added any OAuth Scopes for your Bot token.	
Add an OAuth Scope	

User Token Scopes

Scopes that access user data and act on behalf of users that authorize them.

OAuth Scope	Description
You haven't added any OAuth Scopes for your User token.	
Add an OAuth Scope	

Scopes define the [API methods](#) an app is allowed to call, which information and capabilities are available on the workspace it's installed on. Many scopes are restricted to specific [resources](#) like channels or files.

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.

Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description	▼
app_mentions:read	View messages that directly mention Slack Chatbot Demo in conversations that the app is in	
Add an OAuth Scope		

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.

Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes

Scopes that govern what your app can access.

OAuth Scope	Description	
app_mentions:read	View messages that directly mention Slack Chatbot Demo in conversations that the app is in	
chat:write	Send messages as Slack Chatbot Demo	
im:history	View messages and other content in direct messages that Slack Chatbot Demo has been added to	
im:read	View basic information about direct messages that Slack Chatbot Demo has been added to	

[Add an OAuth Scope](#)

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.

App Credentials

These credentials allow your app to access the Slack API. They are secret. Please don't share your app credentials with anyone, include them in public code repositories, or store them in insecure ways.

App ID

Date of App Creation

March 2, 2020

Client ID

Client Secret

••••••••••

Show **Regenerate**

You'll need to send this secret along with your client ID when making your [oauth.v2.access](#) request.

Signing Secret

••••••••••

Show **Regenerate**

Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.

Verification Token

XXXXXXXXXXXXXX

Regenerate

This deprecated Verification Token can still be used to verify that requests come from Slack, but we strongly recommend using the above, more secure, signing secret instead.

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*.

The screenshot shows the 'OAuth & Permissions' section of the Slack developer console. On the left, there's a sidebar with 'Settings' and 'Features' sections, and a blue bar at the bottom labeled 'OAuth & Permissions'. The main area has a heading 'OAuth Tokens & Redirect URLs'. Below it, a green button labeled 'Install App to Workspace' is highlighted with a large green arrow pointing to it. A descriptive text explains that OAuth tokens will be generated when connecting the app to a workspace.

Slack Chatbot ... ▾

OAuth & Permissions

Settings

- Basic Information
- Collaborators
- Install App
- Manage Distribution

Features

- App Home
- Incoming Webhooks
- Interactive Components
- Slash Commands

OAuth & Permissions

Event Subscriptions

OAuth Tokens & Redirect URLs

These [OAuth Tokens](#) will be automatically generated when you finish connecting the app to your workspace. You'll use these tokens to authenticate your app.

[Install App to Workspace](#)

Redirect URLs

You will need to configure redirect URLs in order to automatically generate the Add to Slack button or to distribute your app. If you pass a URL in an OAuth request, it must (partially) match one of the URLs you enter here. [Learn more](#).

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

This app was created by a member of your workspace, Chatbot Demo.

Slack Chatbot Demo is requesting permission to access the Chatbot Demo Slack workspace



What will Slack Chatbot Demo be able to view?

- Content and info about channels & conversations

What will Slack Chatbot Demo be able to do?

- Perform actions in channels & conversations

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

OAuth & Permissions

OAuth Tokens & Redirect URLs

Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. [Learn more.](#)

Bot User OAuth Access Token

[Copy](#)

[Reinstall App](#)

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.

Step 2

Connect Watson Assistant to Slack

On the Slack app settings page, go to the **Basic Information** tab and find the **App Cred**

Verification token

[REDACTED]

Enter Verification Token

Go to the **OAuth & Permissions** tab. Click **Install App to Workspace**, and then click **Authz**. Enter the verification token and bot user OAuth access token to the fields below.

OAuth access token

[REDACTED]

Bot user OAuth access token

[REDACTED]

Enter Bot User OAuth Access Token for both fields

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

Step 3

Configure your Slack bot

On the Slack app settings page, go to the **Event Subscriptions** tab.

Generate request URL



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*.

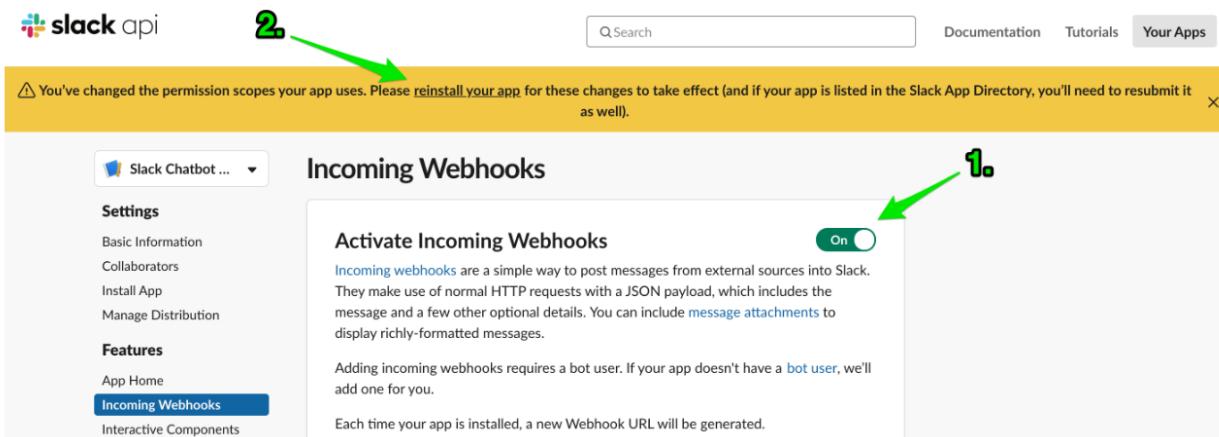
Step 3

Configure your Slack bot

On the Slack app settings page, go to the **Event Subscriptions** tab. Switch the **Enable Events** toggle to the **On** position. In the **Generated request URL**

`https://integrations.us-south.assistant.watson.cloud.ibm.com/public/slack/messages/incoming`

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.

Slack Chatbot Demo is requesting permission to access the Chatbot Demo Slack workspace



What will Slack Chatbot Demo be able to view?

- Content and info about channels & conversations ▶
-

What will Slack Chatbot Demo be able to do?

- Perform actions in channels & conversations ▶
-

Where should Slack Chatbot Demo post?

- # Slack Chatbot Demo requires a channel to post to as an app

watson-assistant-chatbot-demo ▾

Cancel

Allow



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.

Event Subscriptions

Enable Events

On

Your app can subscribe to be notified of events in Slack (for example, when a user adds a reaction or creates a file) at a URL you choose. [Learn more.](#)

Request URL

https://integrations.us-south.assistant.watson.cloud.ibm.com/public/slack/message/2

We'll send HTTP POST requests to this URL when events occur. As soon as you enter a URL, we'll send a request with a `challenge` parameter, and your endpoint must respond with the challenge value. [Learn more.](#)



Subscribe to bot events

Subscribe to events on behalf of users

App unfurl domains

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.

Subscribe to bot events



Apps can subscribe to receive events the bot user has access to (like new messages in a channel). If you add an event here, we'll add the necessary [OAuth scope](#) for you.

Event Name	Description	Required Scope	
message.im	A message was posted in a direct message channel	im:history	
app_mention	Subscribe to only the message events that mention your app or bot	app_mentions:read	

[Add Bot User Event](#)

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.



This is the very beginning of your direct message history with [@slack_chatbot_demo](#)

💡 How does slack_chatbot_demo work?

Today



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

