IBM Watson Assistant

with Webhook Integration (Cloud Function/Node.js)

Cognitive Solutions Application Development

Updated: May 6, 2020
Klaus-Peter Schlotter
kps@de.ibm.com

Version 5.0 (Watson library V5.x)

Uses the Assistant API V2

Table of Contents

Overview

The IBM Watson Developer Cloud (WDC) offers a variety of services for developing cognitive applications. Each Watson service provides a Representational State Transfer (REST) Application Programming Interface (API) for interacting with the service. Software Development Kits (SDKs) are also available and provide high-level wrappers for the underlying REST API. Using these SDKs will allow you to speed up and simplify the process of integrating cognitive services into your applications.

The Watson Assistant (formerly Conversation) service combines a number of cognitive techniques to help you build and train a bot - defining intents and entities and crafting a dialog to simulate conversation. The system can then be further refined with supplementary technologies to make the system more human-like or to give it a higher chance of returning the right answer. Watson Assistant allows you to deploy a range of bots via many channels, from simple, narrowly focused bots to much more sophisticated, full-blown virtual agents across mobile devices, messaging platforms like Slack, or even through a physical robot.

Examples of where Watson Assistant could be used include:

- Add a chat bot to your website that automatically responds to customers' questions
- Build messaging platform chat bots that interact instantly with channel users
- Allow customers to control your mobile app using natural language virtual agents
- And more!

Objectives

- Learn how to provision a Watson Assistant service and utilize the web tool interface
- Learn how to train your chat bot to answer common questions
- Learn how to integrate other systems via Webhooks (Cloud Function/Node.js).

Prerequisites

Before you start the exercises in this guide, you will need to complete the following prerequisite tasks:

- Create a IBM Cloud account

Section 1: Create a Conversation Dialog in IBM Cloud

Create a Conversation Service in IBM Cloud

IBM Cloud offers services, or cloud extensions, that provide additional functionality that is ready to use by your application's running code.

You have two options for working with applications and services in IBM Cloud. You can use the IBM Cloud web user interface or the Cloud Foundry command-line interface. (See Lab 01 on how to use the Cloud Foundry CLI).

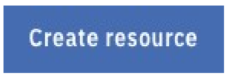
Note: In this lab we use the IBM Cloud web UI.

Step 1 In a web browser, navigate to the following URL

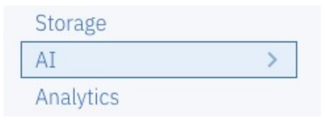
https://cloud.ibm.com.

Step 2 Log in with your IBM Cloud credentials. This should be your IBMid.

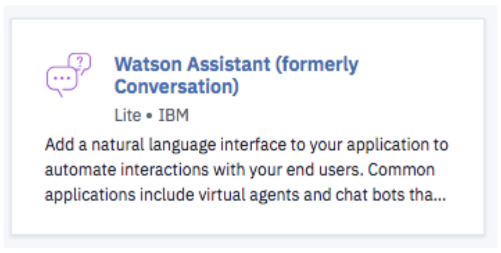
Step 3 You should start on your dashboard which shows a list of your applications and services. Scroll down to the All Services section and click **Create Resource**.



Step 4 On the left, under Services, **click** on *AI* to filter the list and only show the cognitive services.



Step 5 Click on the Watson Assistant service.


Watson Assistant (formerly Conversation)

Lite • IBM

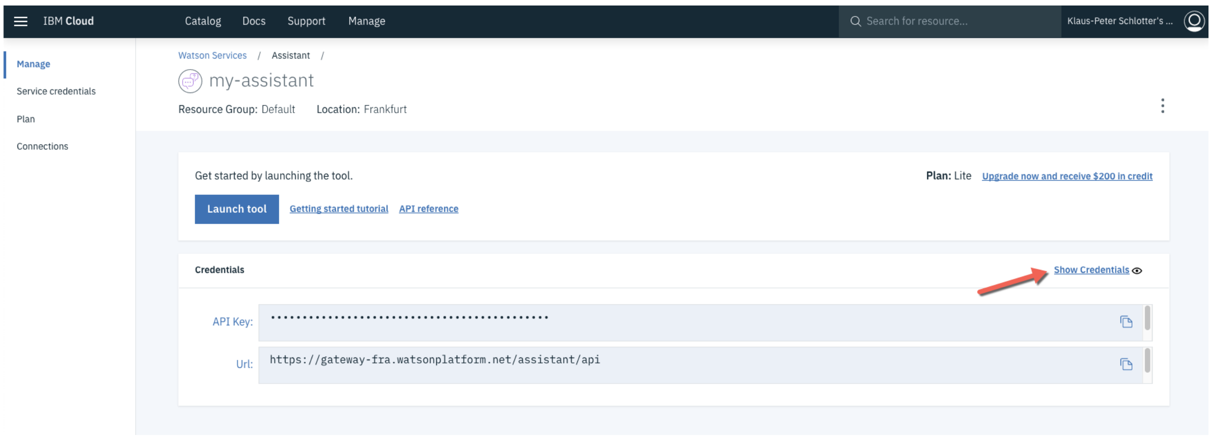
Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots tha...

Step 6 Review the details for this service. At the top, there will be a description of the service. At the bottom, you can review the pricing plans. The Lite plan for this service provides no cost monthly allowances for workspaces, intents, and API calls. Enjoy your demo!

Step 7 Enter the information for your service, then **click** Create.

Field	Color property
Service name	my-assistant
Selected Plan	Lite
Chose a region/location to deploy	<yourRegion>
Select a resource group	Default

Step 8 IBM Cloud has created a new service instance.



IBM Cloud Catalog Docs Support Manage Search for resource... Klaus-Peter Schlotter's ...

Manage
Service credentials
Plan
Connections

Watson Services / Assistant /

my-assistant

Resource Group: Default Location: Frankfurt

Get started by launching the tool.

Plan: Lite Upgrade now and receive $200 in credit

Launch tool Getting started tutorial API reference

Credentials Show Credentials

API Key: ··

Url: `https://gateway-fra.watsonplatform.net/assistant/api`

Step 9 In the *Credentials* section **click** Show Credentials . You should see the *API Key* for your service. Later in this exercise, you will enter this value into a JSON configuration file for your Node.js application. Feel free to copy them to your clipboard, to a text file, or just return to this section of the IBM Cloud web interface when the credentials are needed.

Create a Watson Assistant Skill

Before using the Assistant instance, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. You will create these items in a Skill in the following steps. A Skill is a container for the artifacts that define the behavior of your service instance.

The resulting Skill is also available as JSON backup file on GitHub. In a Terminal you can download it with the following command:

```
wget https://github.com/iic-dach/csadConversation/blob/master/Lab03a_skill-CSAD-with-CloudFunction.json
```

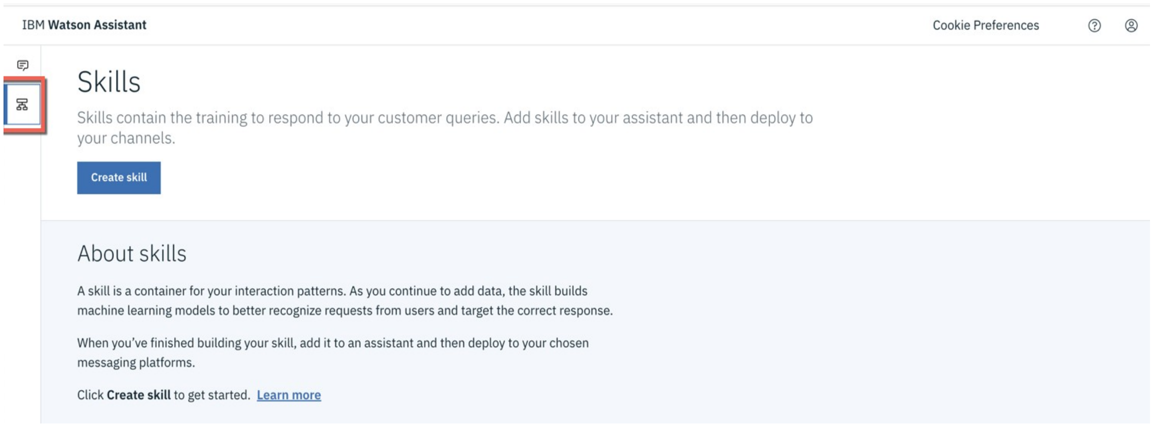
On the *Add Dialog Skill* page, on the *Import skill* tab you can import this JSON file..

Step 10 Click the **Launch tool** button.

Launch Watson Assistant

Step 11 The IBM Watson Assistant opens on the *Assistants* tab.

Step 12 Open the *Skills* tab. Here you can create new Skills.


IBM Watson Assistant

Cookie Preferences

Skills

Skills contain the training to respond to your customer queries. Add skills to your assistant and then deploy to your channels.

Create skill

About skills

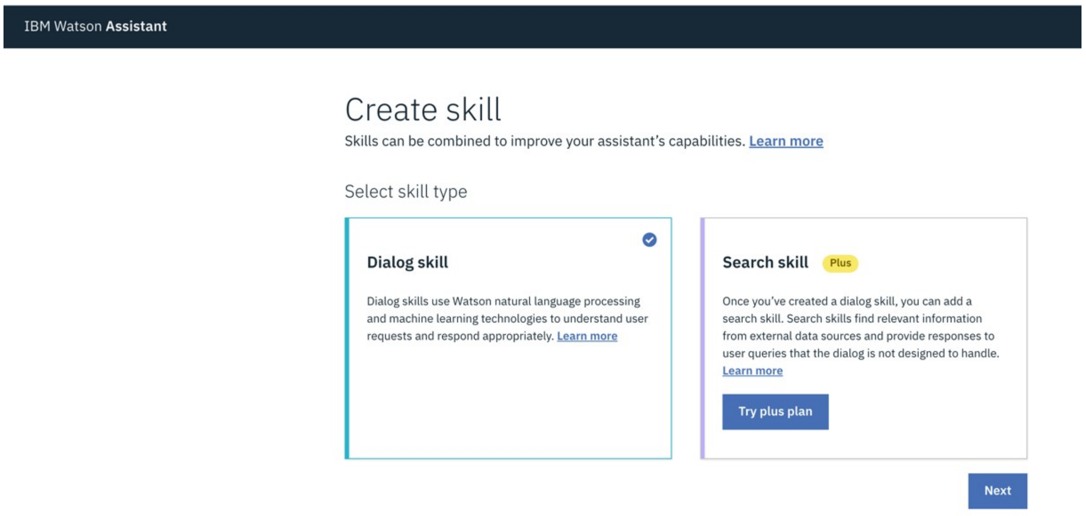
A skill is a container for your interaction patterns. As you continue to add data, the skill builds machine learning models to better recognize requests from users and target the correct response.

When you've finished building your skill, add it to an assistant and then deploy to your chosen messaging platforms.

Click **Create skill** to get started. Learn more

Step 13 First, you will need to create a new workspace. Workspaces enable you to maintain separate intents, user examples, entities, and dialogs for each application. **Click** Create skill

Step 14 On the *Create skill* page **select** *Dialog skill* and **click** Next


IBM Watson Assistant

Create skill

Skills can be combined to improve your assistant's capabilities. Learn more

Select skill type

Dialog skill

Dialog skills use Watson natural language processing and machine learning technologies to understand user requests and respond appropriately. Learn more

Search skill Plus

Once you've created a dialog skill, you can add a search skill. Search skills find relevant information from external data sources and provide responses to user queries that the dialog is not designed to handle. Learn more

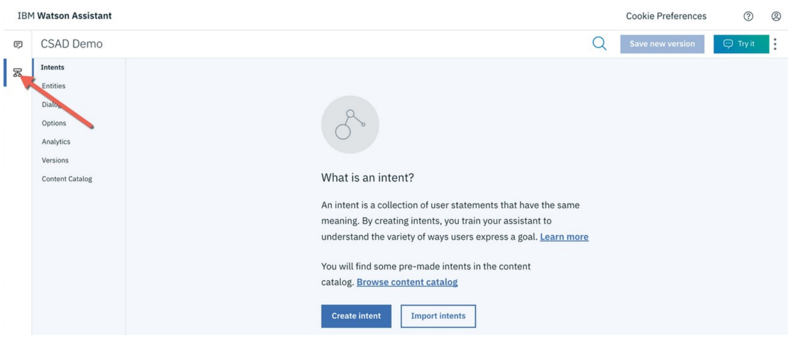
Try plus plan

Next

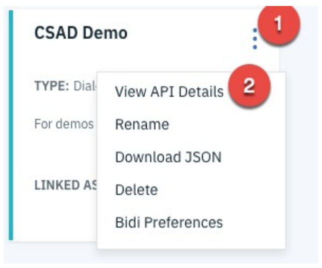
Step 15 On the *Create Dialog Skill* page, on the *Create skill* tab, **enter** the following values and click Create dialog skill.

Field	Value
Name	CSAD Demo
Description	For demos only
Language	We use **English(U.S.)** for this demo

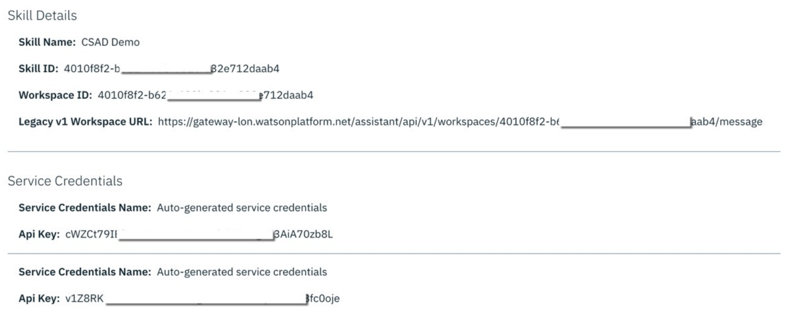
Step 16 Once the Skill has been created, you will be redirected to it. However, before proceeding, you will need to know how to identify the Skill so that it can be referenced by future applications. At the left, **click** Skills.


IBM Watson Assistant
Cookie Preferences
CSAD Demo
Save new version
Try it
Intents
Entities
Dialog
Options
Analytics
Versions
Content Catalog
What is an intent?
An intent is a collection of user statements that have the same meaning. By creating intents, you train your assistant to understand the variety of ways users express a goal. Learn more
You will find some pre-made intents in the content catalog. Browse content catalog
Create intent
Import intents

Step 17 On the tile for your new Skill, click **Options** → **View API Details**.



Step 18 Locate the Workspace ID. You will need this value in future steps when creating a JSON configuration file for your demo application. Feel free to copy the value or just return to this section of the Conversation tooling web interface when the ID is needed.


Skill Details

Skill Name: CSAD Demo

Skill ID: 4010f8f2-b[redacted]82e712daab4

Workspace ID: 4010f8f2-b62[redacted]712daab4

Legacy v1 Workspace URL: https://gateway-lon.watsonplatform.net/assistant/api/v1/workspaces/4010f8f2-b6[redacted]aab4/message

Service Credentials

Service Credentials Name: Auto-generated service credentials

Api Key: cWZCt79II[redacted]3AiA70zb8L

Service Credentials Name: Auto-generated service credentials

Api Key: v1Z8RK[redacted]fc0oje

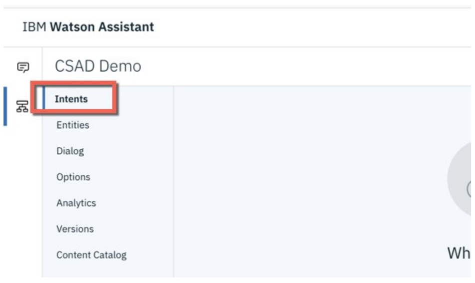
Step 19 In the upper right corner of this page, **click** ✕ .

Step 20 **Click** on the Skill tile to be taken back to the new Skill.

Create Intents

Before using the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. An intent is the purpose or goal of a user's input.

Step 21 First, you will need to define some intents to help control the flow of your dialog. An intent is the purpose or goal of a user's input. In other words, Watson will use natural language processing to recognize the intent of the user's question/statement to help it select the corresponding dialog branch for your automated conversation. If not already there, **click** the *Intents* tab at the left of your workspace.



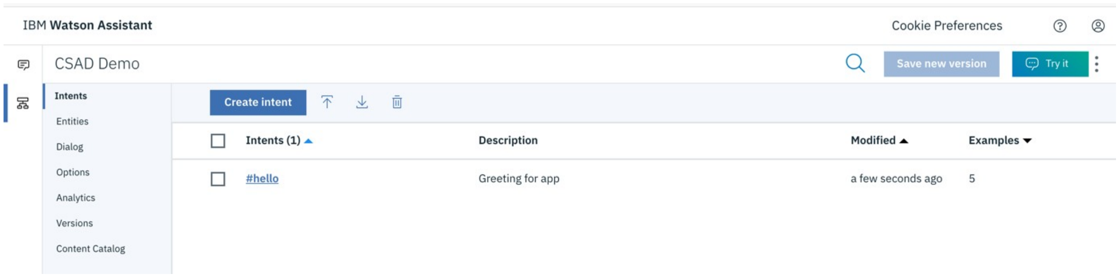
Step 22 **Click** Create intent enter the following values and **click** Create intent

Field	Value
Intent name	hello
Description	Greeting for app

Step 23 The user examples are phrases that will help Watson recognize the new intent. (Enter multiple examples by **pressing** "Enter" or by **clicking** the *Add example*). When finished, **click** ← at the top of the page.

Field	Value
User example	Good morning Greetings Hello Hi Howdy

Step 24 You should see your new intent listed on the Intents page. The number you see will indicate the total number of user examples that exist for that intent.



Intents (1)	Description	Modified	Examples
#hello	Greeting for app	a few seconds ago	5

Step 25 Repeat the previous steps to create a new *intent* that helps the user end the conversation.

Field	Value
Intent name	goodbye
Description	Goodbye
User example	Bye Goodbye Farewell I am leaving See you later

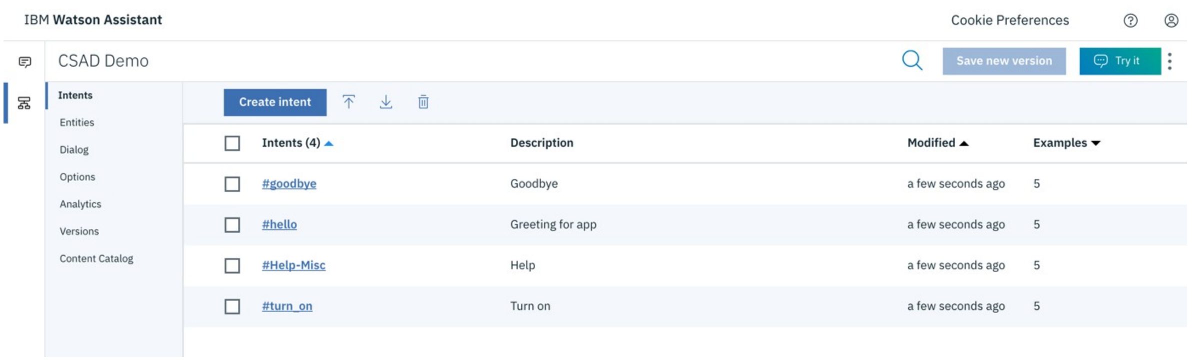
Step 26 Repeat the previous steps to create a new *intent* that helps the user ask for help. In the programming section of this guide, you will learn how to identify the user’s intent (in a third-party application) so that you can perform the re-quested action.

Field	Value
Intent name	Help-Misc
Description	Help
User example	I have a request I would like some assistance I need information I have a problem I need help

Step 27 Repeat the previous steps to create a new *intent* that helps the user issue commands to turn on a device. In this example, you will assume the user is interacting with a home/business automation system. The purpose of this intent is to show you (in the next section) how to associate *entities* with an *intent* when building your dialog tree. Additionally, this intent demonstrates that the Conversation service can be used for more than just chat bots. It can be used to provide a natural language interface to any type of system!

Field	Value
Intent name	turn_on
Description	Turn on
User example	Arm the security system Lock the doors I need lights Turn on the lights Start recording

Step 28 At this point, you have defined some intents for the application along with the example utterances that will help train Watson to recognize and control the conversation flow.



Intents (4)	Description	Modified	Examples
#goodbye	Goodbye	a few seconds ago	5
#hello	Greeting for app	a few seconds ago	5
#Help-Misc	Help	a few seconds ago	5
#turn_on	Turn on	a few seconds ago	5

Create Entities

Before using the new Assistant, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. An entity is the portion of the user's input that you can use to provide a different response or action to an intent.

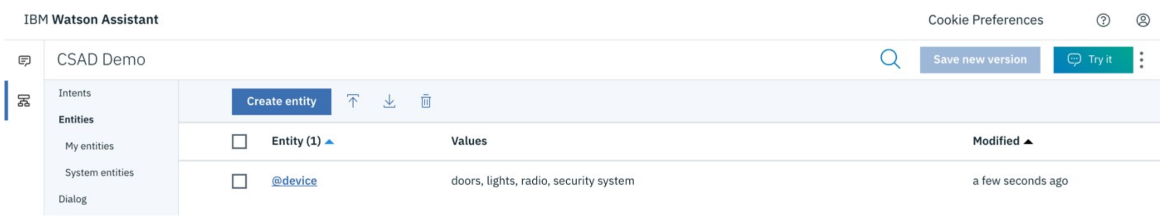
Step 29 Next, you will need to create some *entities*. An *entity* is the portion of the user's input that you can use to provide a different response or action to an *intent*. These entities can be used to help clarify a user's questions/phrases. You should only create *entities* for things that matter and might alter the way a bot responds to an *intent*. If not already there, **click** the *Entities* tab at the left of your Skill.

Step 30 On the *Entities* tab, click Create entity. In the dialog window, enter the following information. In this example, the **#turn_on** *intent* will indicate that the user wants to turn on a car device. So, you will need to create a new *entity* representing a device. **Enter** *device* and **click** Create entity. You will then provide *values* (and possibly synonyms) for the various types of devices that can be turned on. (Enter multiple *examples* by **pressing** "Enter" or by **clicking** the plus sign at the end of the line.)

Click Add value. When finished, **click** ← at the top of the page.

Field	Value	Synonym
Entity name	device	
Value	security system	alarm
	lights	bulb, lamp
	doors	locks, gates
	radio	car radio

Step 31 You should see your new *entity* listed on the *Entities* page.


IBM Watson Assistant | Cookie Preferences

CSAD Demo | Save new version | Try it

Intents
Entities
My entities
System entities
Dialog

Create entity

	Entity (1)	Values	Modified
☐	@device	doors, lights, radio, security system	a few seconds ago

Step 32 Repeat the previous steps to create the following new *entity* for **lights** controlled by the system.

Field	Value	Synonym
Entity name	lights	
Value	fog lamp	fog light
	high beam	full beam, main beam, brights
	low beam	headlights, passing lights, dim light
	rear fog lamp	rear fog light

Step 33 Repeat the previous steps to create the following new *entity* to request the current time.

Note: You can get the time by using one of the predefined system entities. In this exercise, go ahead and enter it manually so it can be used to demonstrate future concepts.

Field	Value	Synonym
Entity name	help	
Value	time	clock, hour, minute, second

Step 34 At this point, you have defined *intents* and the associated *entities* to help Watson determine the appropriate response to a user's natural language input. Your application will be able to:

- Respond to a request to turn on specific devices
- If a user turns on lights, provide additional choices for light locations

Create Dialogs

Before using the new conversation, you will need to train it with the intents, entities, and/or dialog nodes relevant to your application's use case. A dialog uses the intent and entity that have been identified, plus context from the application, to interact with the user and provide a response.

Step 35 Next, you will need to create some *dialogs*. A dialog is a conversational set of nodes that are contained in a workspace. Together, each set of nodes creates the overall dialog tree. Every branch on this tree is a different part of the conversation with a user. If not already there, **click** the *Dialog* tab at the left of your Skill.

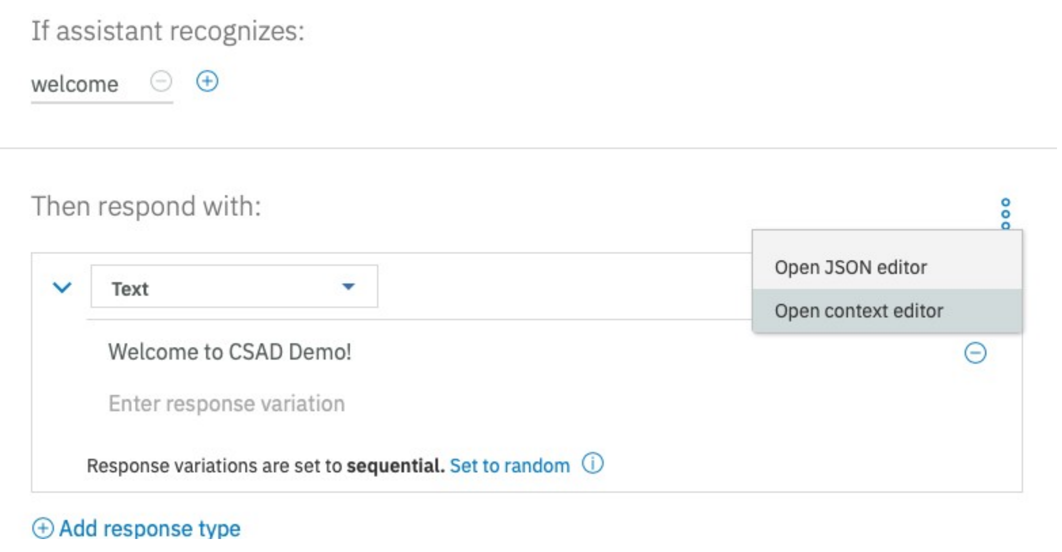
Step 36 Review the documentation for creating dialog nodes and for defining conditions.

Step 37 On the Dialog tab, two default nodes are created for you named *Welcome* and *Anything* else. The *Welcome* node is the starting point for this application's conversation. That is, if an API query is made without a context defined, this node will be returned. The *Anything else* node will be used if the user input does not match any of the defined nodes.

Step 38 **Click** on the *Welcome* node to update the following properties

Field	Value
Name	Welcome *(should be the default)*
If bot recognizes:	welcome *(should be the default)*
Then respond with:	Welcome to the CSAD Demo!

Step 39 **Click** ⋮ then **click** *Open context editor*.


If assistant recognizes:

welcome

Then respond with:

Text

Welcome to CSAD Demo!

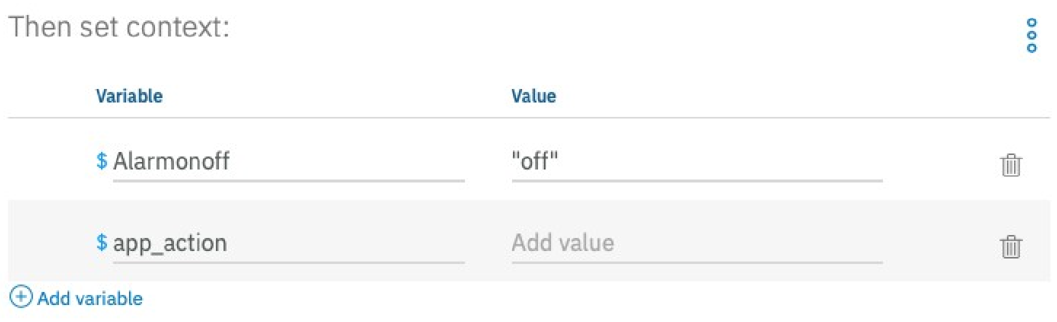
Enter response variation

Response variations are set to **sequential.** Set to random

Add response type

Step 40 Add two variables and **click** Customize ⚙ ✕ to close the properties view of the node.

Variable	Value
Alarmonoff	off
app_action	


Then set context:

Variable	Value
$ Alarmonoff	"off"
$ app_action	Add value

Add variable

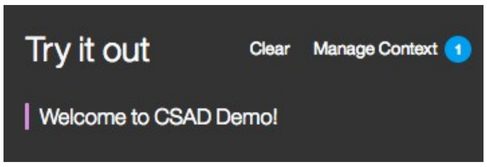
Step 41 Expand the *Anything else* node and review it's default values. **Close** the view by **clicking** × .

Field	Value
Name	Anything else (should be the default)
If bot recognizes:	anything_else (should be the default)
Then respond with:	1. I didn't understand. You can try rephrasing 2. Can you reword your statement? I'm not understanding. 3. I didn't get your meaning. (all should be default)
Response variations are sequential.* (Set to random)	

* Sequential means 1st response presented at first hit of anything_else node, and so on. Random means any of the responses is presented randomly.

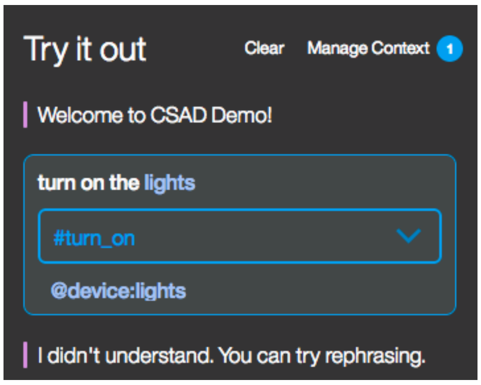
Step 42 Now it's time to test the conversation.
In the upper right corner, **click** Try it

Step 43 A test user interface will immediately launch and, based on the *Welcome* node, provides a greeting to the end user. (You may see a message that Watson is being trained.)



Welcome to CSAD Demo!

Step 44 Since you have not yet defined any other dialog nodes (associated with your *intents* and *entities*), everything typed by the user will get routed to the *Anything else* node. F.e **type** "*turn on the lights*".



Welcome to CSAD Demo!
turn on the lights
#turn_on
@device:lights
I didn't understand. You can try rephrasing.

Although we have defined this phrase in intents and entities, the system does not recognize them because we have not yet defined a node to catch them the bot does not yet understand (*Anything else* node).

Step 45 Did you notice the drop-down menu that appeared for your invalid input? You can optionally assign this phrase to an existing intent (or verify the correct intent was used). You can use this functionality in the future to keep Watson trained on new user inputs and to ensure the correct response is returned. Cool! For now, just proceed to the next step.

Step 46 **Click** in the top right corner to close the chat pane. Proceed to the next section.

Build your CSAD Assistant dialog

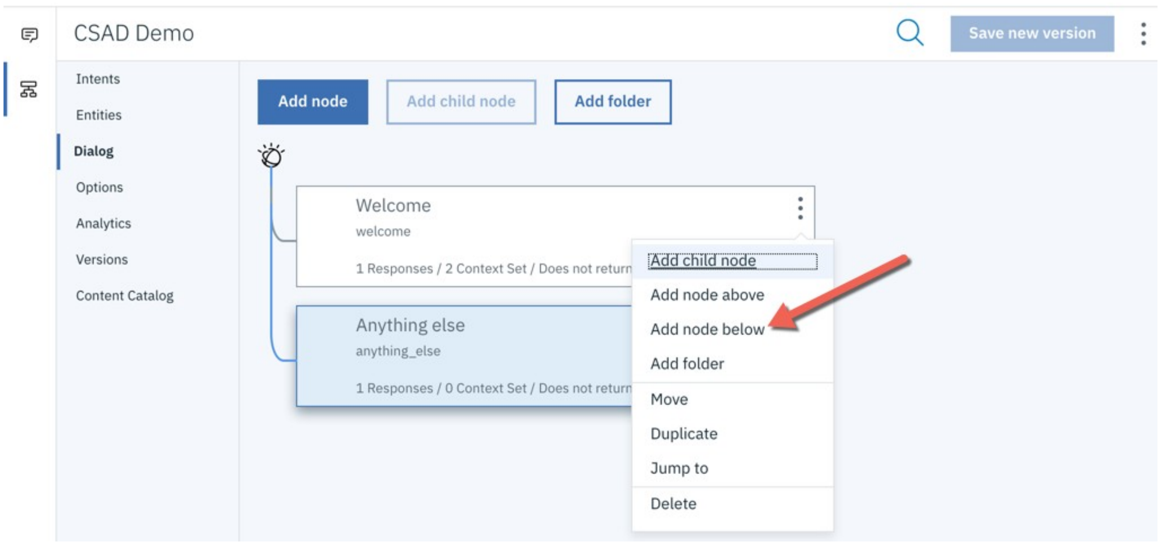
In this section, you will continue building your demo bot utilizing the intents, entities, and dialog nodes that you created in previous steps. You will do this entirely with the web interface (no programming or text/XML file hacking required!)

Step 47 You should create a dialog branch for each of the intents you identified as well as the start and end of the conversation. Determining the most efficient order in which to check conditions is an important skill in building dialog trees. If you find a branch is becoming very complex, check the conditions to see whether you can simplify your dialog by reordering them.

Note: It's often best to process the most specific conditions first.

Step 48 **Click** the menu on the *Welcome* node and then **click** *Add node below*.

In this step you are creating a new branch in your dialog tree that represents an alternative conversation.



Step 49 In this new node, enter the following values. By setting the condition to an *intent*, you are indicating that this node will be triggered by any input that matches the specified *intent*. Then **click** × to close the dialog.

Field	Value
Name this node…	Hello
If bot recognizes:	#hello
Then respond with:	Hi! What can I do for you?

Step 50 **Click** the menu ⋮ on the *Hello* node and then **click** *Add node below*, with the following values. Then **click** × to close the dialog.

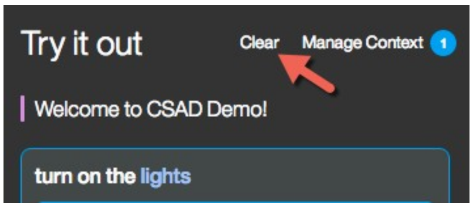
Field	Value
Name this node…	Goodbye
If bot recognizes:	#goodbye
Then respond with:	Until our next meeting.

Step 51 Using the same steps (Step 40 ff) as before, test the conversation by typing the following chat line(s):

```
Hello

Farewell
```

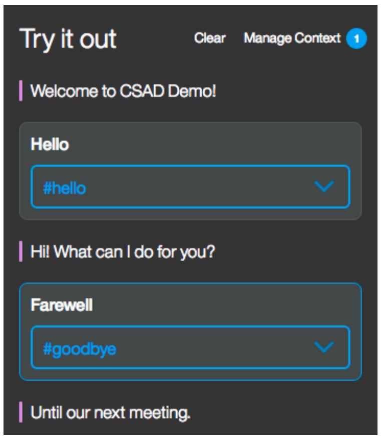
You can clear previous tests by clicking **Clear** at the top of the dialog.



Welcome to CSAD Demo!

turn on the lights

You should see the appropriate result:



Welcome to CSAD Demo!

Hello

#hello

Hi! What can I do for you?

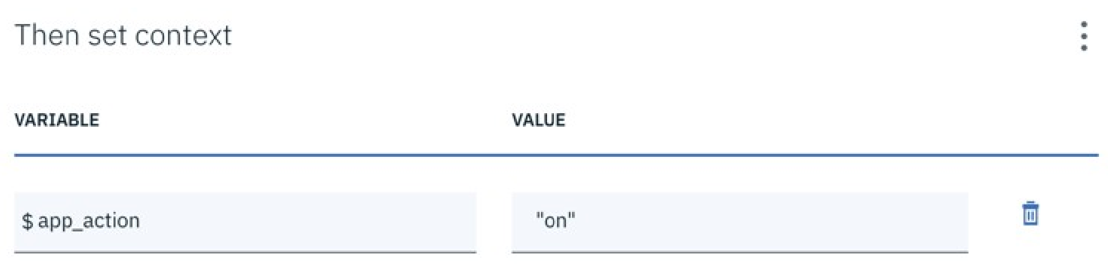
Farewell

#goodbye

Until our next meeting.

Step 52 Next, you should create a new node below **Hello** (*Add node below*) for the **#turn_on** *intent*. As you'll recall, you have multiple devices that you might want to turn on. In earlier steps, you documented these devices using a new **@device** *entity*. This dialog branch will require multiple nodes to represent each of those devices. In this new node, enter the following values. In this example, the dialog branch will need additional information to determine which device needs to be turned on. So, leave the "Responses" field blank. **Click** × to close.

Field	Value
Name this node…	Turn on
If bot recognizes:	#turn_on
Then respond with:	
In the Context Editor (click ⋮)	
app_action	on


Then set context

VARIABLE	VALUE
$ app_action	"on"

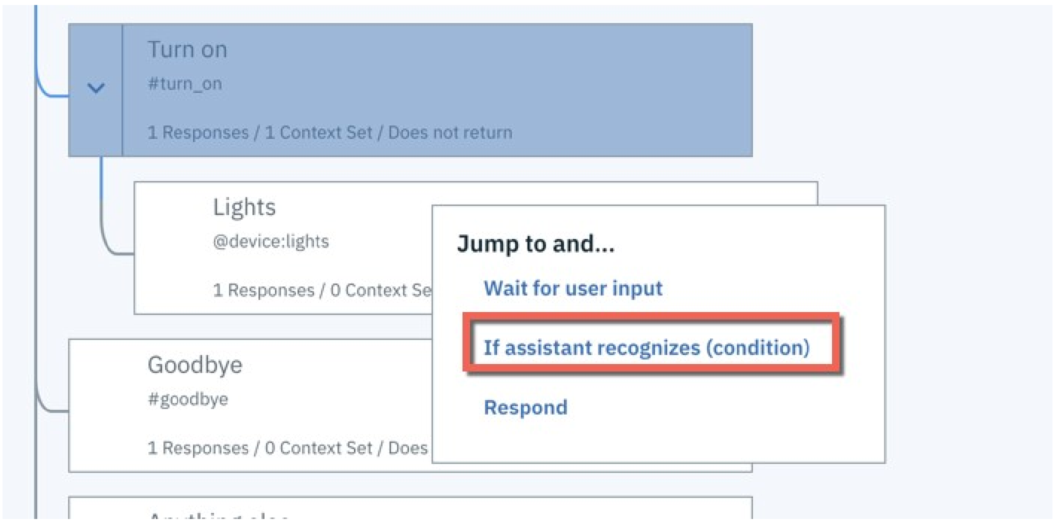
Step 53 **Click** the menu ⋮ on the *Turn On* node and **click** *Add child node*.

Step 54 In this new node, enter the following values. In this example, the only way this node will be reached is if the user specifies the **@device** *entity* "lights" (or one of its synonyms). Then **click** × to close the dialog.

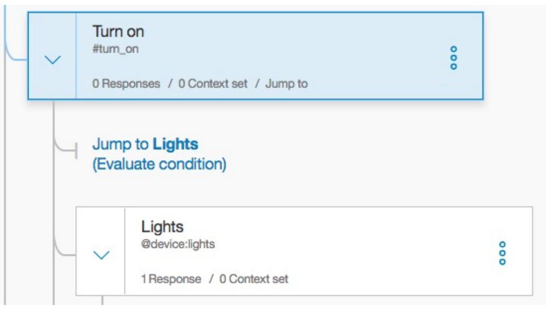
Field	Value
Name this node…	Lights
If bot recognizes:	@device:lights
Then respond with:	OK! Which light would you like to turn on?

Step 55 In this scenario, you want to automatically move from the *Turn On* node to the *Lights* node without waiting for additional user input.

a) **Click** ⋮ on the *Turn on* node and **select** *Jump to.* Now the Turn on node is selected. **Click** the *Lights* node and **select** I*f assistant recognizes (condition)*.


Turn on
#turn_on
1 Responses / 1 Context Set / Does not return
Lights
@device:lights
1 Responses / 0 Context Se
Goodbye
#goodbye
1 Responses / 0 Context Set / Does
Jump to and...
Wait for user input
If assistant recognizes (condition)
Respond

b) If bot recognizes (condition) is **lights**.



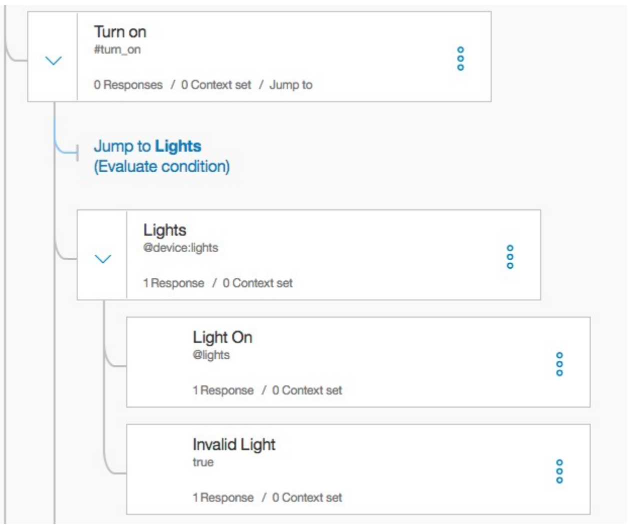
Step 56 At this point, Watson will ask you which lights to turn on. As you will recall, in earlier steps you created a @lights entity to define the available lights in the system. **Click** the menu ⋮ on the *Lights* node, **click** *Add child node*. In this new node, enter the following values. By setting the condition to an entity, you are indicating that this node will be triggered by any input that matches the specified entity.

Field	Value
Name this node…	Light On
If bot recognizes:	@lights
Then respond with:	OK! Turning on @lights light.

Step 57 Next, you will want Watson to respond if it does not recognize a light, or entity, provided by the user. **Click** the menu on the **Light On** node and **click** *Add node below* to create a new peer node. In this new node, enter the following values.

Note: The *true* keyword in *If bot recognizes* always executes this node when reached. That means, the conditions of the siblings above (Light On) are not recognized!

Now your tree for the lights should look like the following:



Turn on
#turn_on
0 Responses / 0 Context set / Jump to

Jump to **Lights**
(Evaluate condition)

Lights
@device:lights
1 Response / 0 Context set

Light On
@lights
1 Response / 0 Context set

Invalid Light
true
1 Response / 0 Context set

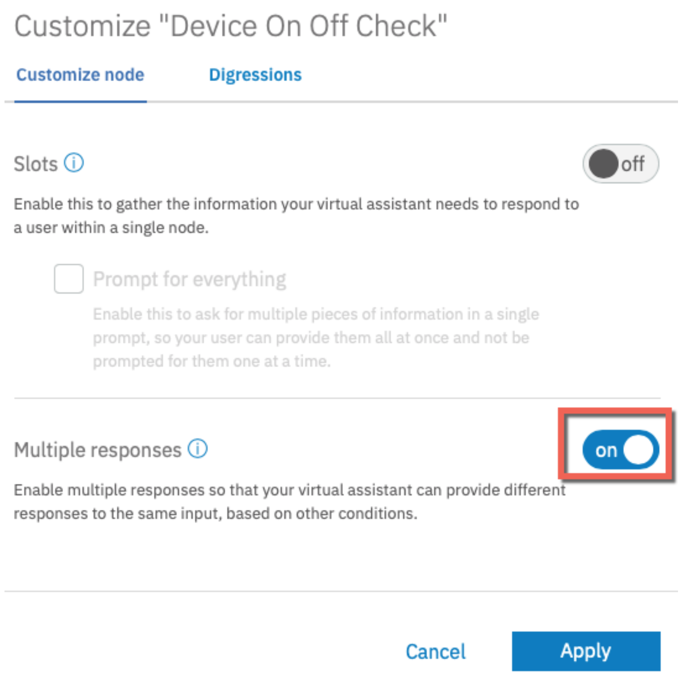
Step 58 You will need to add nodes to deal with the other devices that can be turned on. As you'll recall, you defined those devices with the **@device** *entity*. The **Turn On** node automatically jumps to the Lights node. So, **click** the menu on the Lights node and **click** *Add node below* to create a new peer node. **Enter** the following values.

Field	Value
Name this node…	Device
If bot recognizes:	@device
Then respond with:	

Step 59 On the *Device* node **click** and **click** *Add a child node*. Add the following values

Field	Value
Name this node…	Device On Off Check
If bot recognizes:	true

Step 60 **Click** Customize and **enable** *Multiple responses*. The click Apply.



Customize "Device On Off Check"

Slots — off

Enable this to gather the information your virtual assistant needs to respond to a user within a single node.

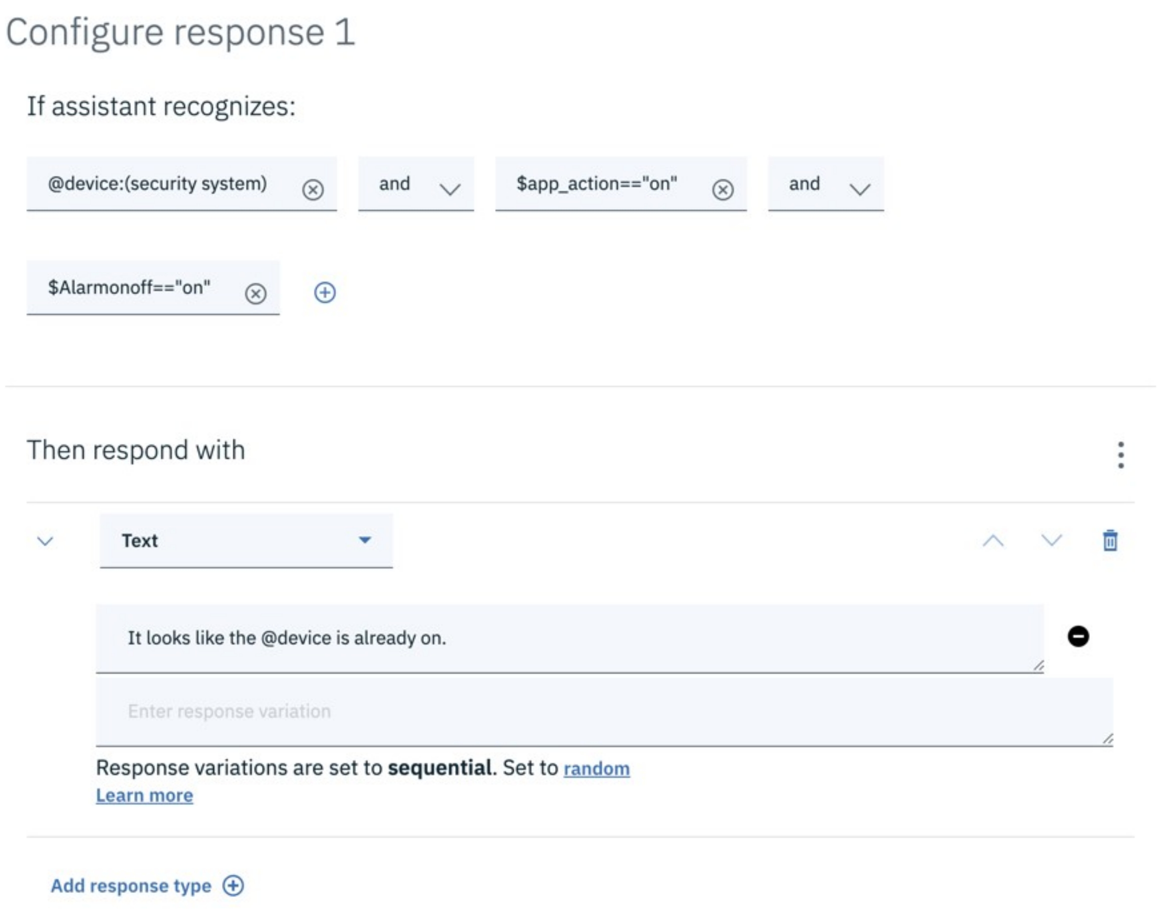
☐ Prompt for everything

Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.

Multiple responses — on

Enable multiple responses so that your virtual assistant can provide different responses to the same input, based on other conditions.

Step 61 Now add the following three answers in *Then respond with*: Use the ⚙ to customize and enter the values interactively and **click** Save. See the screenshots of each response:


Configure response 1

If assistant recognizes:

@device:(security system) and $app_action=="on" and $Alarmonoff=="on"

Then respond with

Text

It looks like the @device is already on.

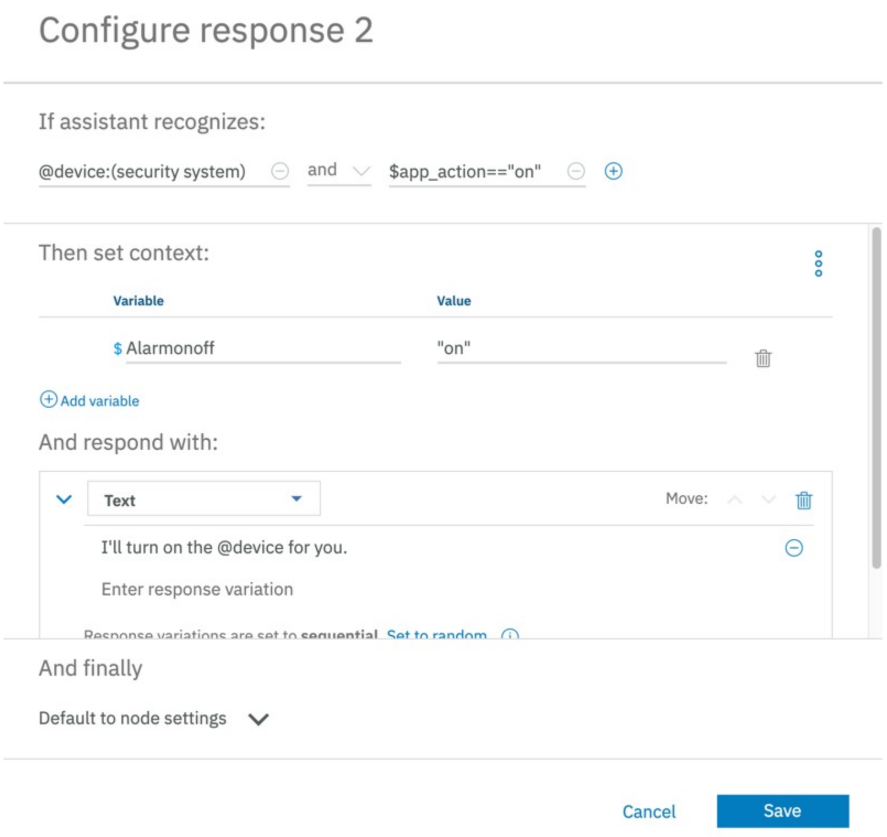
Enter response variation

Response variations are set to **sequential**. Set to random

Learn more

Add response type

If assistant recognizes:
@Device:(security system) AND $app_action==”on” AND $Alarmonoff==”on”
Then respond with:
It looks like the @device is already on.


Configure response 2

If assistant recognizes:

@device:(security system) and $app_action=="on"

Then set context:

Variable	Value
$ Alarmonoff	"on"

Add variable

And respond with:

Text

I'll turn on the @device for you.

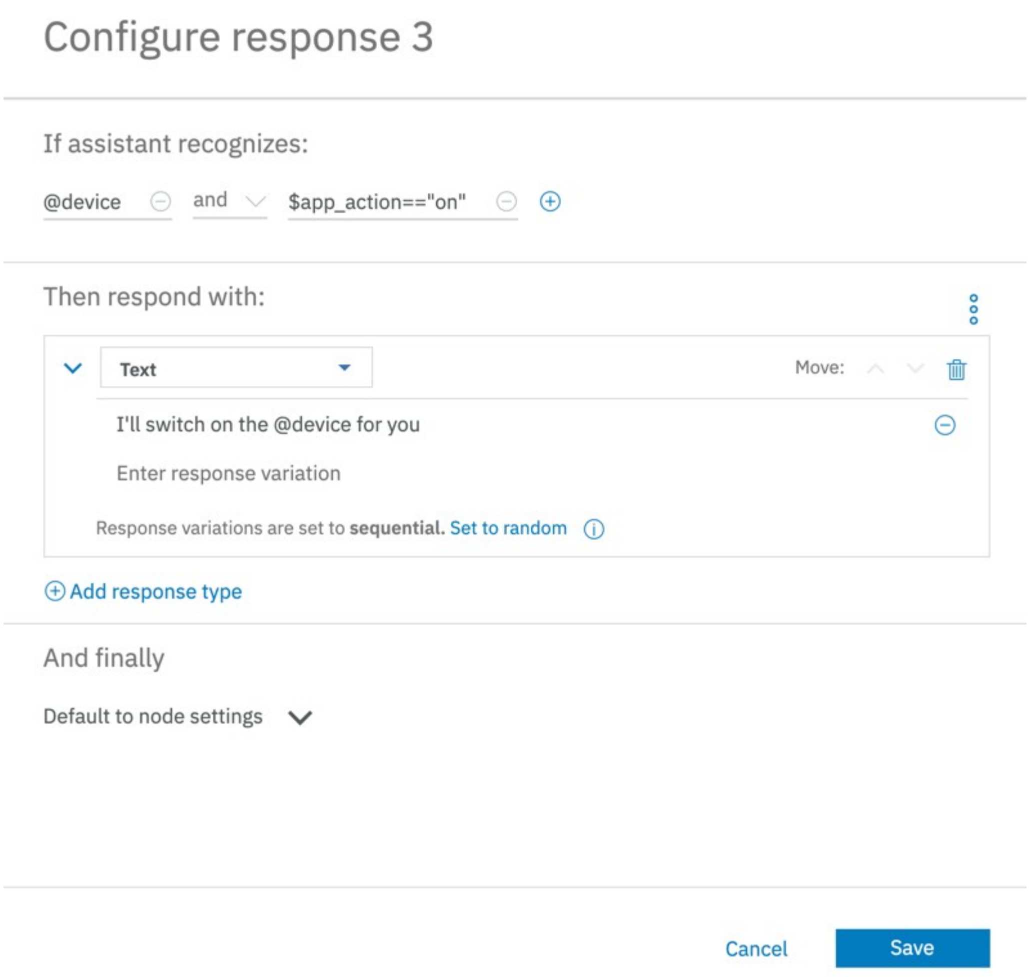
Enter response variation

Response variations are set to **sequential**. Set to random

And finally

Default to node settings

If assistant recognizes:
@Device:(security system) AND $app_action=="on"
Then set context (open the context editor)
$Alarmonoff "on"
And respond with:
I'll turn on the @device for you.


Configure response 3

If assistant recognizes:

@device and $app_action=="on"

Then respond with:

Text

I'll switch on the @device for you

Enter response variation

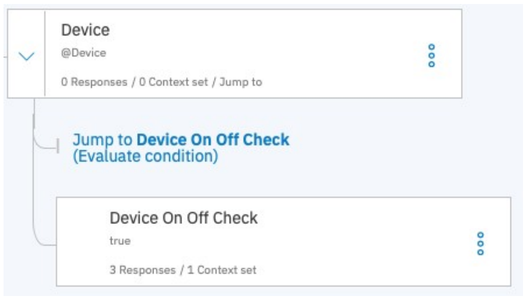
Response variations are set to **sequential.** Set to random

And finally

Default to node settings

If assistant recognizes:
@Device AND $app_action==”on”
Then respond with:
I’ll switch on the @device for you.

Step 62 Now **click** ⋮ on the *Device* node and **click** *Jump to*, then **select** the *Device On Off Check (If recognizes condition)*.



Device
@Device
0 Responses / 0 Context set / Jump to

Jump to **Device On Off Check** (Evaluate condition)

Device On Off Check
true
3 Responses / 1 Context set

Step 63 Next, you will want Watson to respond if it does not recognize a device, or entity, provided by the user. **Click** the menu ⋮ of the *Device* node and **click** *Add node below* to create another peer node with the following values:

Field	Value
Name this node…	Invalid Device
If bot recognizes:	true
Then respond with:	I'm sorry, I don't know how to do that. I can turn on lights, radio, or security systems.

Step 64 Using the same steps as before, **click** [Try it] to test the conversation by **typing** the following chat line(s):

```
Hello

Arm the security system

Arm the security system
```
→ Should be already on

```
Turn on the lights

The headlights

Turn on the lights

fog lamp
```

Note: You could add a context variable for each device/light and control the on/off behavior as for the *security system*.

Step 65 Finally, you will want to enable the Conversation service to identify your **#Help-Misc** *intent*. You will make this a part of the overall “Help” system for the bot. This intent will be used in the upcoming programming exercises to show you how to perform specific application actions based on a detected *intent* and *entity*. **Click** the menu ⋮ on the *Turn On* node and **click** *Add node below* to create a new peer node. Enter the following values.

Field	Value
Name this node…	Help
If bot recognizes:	#Help-Misc
Then respond with:	How can I help you?

Step 66 **Click** *Add child node* on the menu of the Help node.. In this new node, enter the following values. In this example, the only way this node will be reached is if the user specifies the @help:time *entity* (or one of its synonyms).

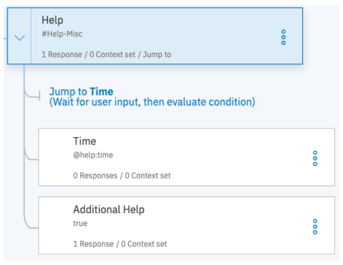
Note: You will provide your own custom response to the query in the programming exercises.

Field	Value
Name this node…	Time
If bot recognizes:	@help:time
The respond with:	The time will be provided by the webhook created later in this tutorial.

Step 67 **Click** *Add node below* on the menu of the *Time* node. In this new node, enter the following values. By setting the condition to an entity, you are indicating that this node will be triggered by any input that matches the specified entity.

Field	Value
Name this node…	Additional Help
If bot recognizes:	true
Then respond with:	I’m sorry, I can’t help with that. For additional help, please call 555-2368.

Step 68 **Click** ⋮ on the *Help* node and **select** *Jump to*. **Click** the *Time* node and **select** *Jump to and… Wait for user input*.



Help
#Help-Misc
1 Response / 0 Context set / Jump to

Jump to **Time**
(Wait for user input, then evaluate condition)

Time
@help:time
0 Responses / 0 Context set

Additional Help
true
1 Response / 0 Context set

Step 69 This is already an epic conversation, but feel free to experiment with more functionality:

- Define entities for additional devices and lights
- Add more synonyms for entities
- Add new intents, such as #turn_off to turn off devices

Step 70 **(Optional)** Add a slots dialog to book a table in a restaurant

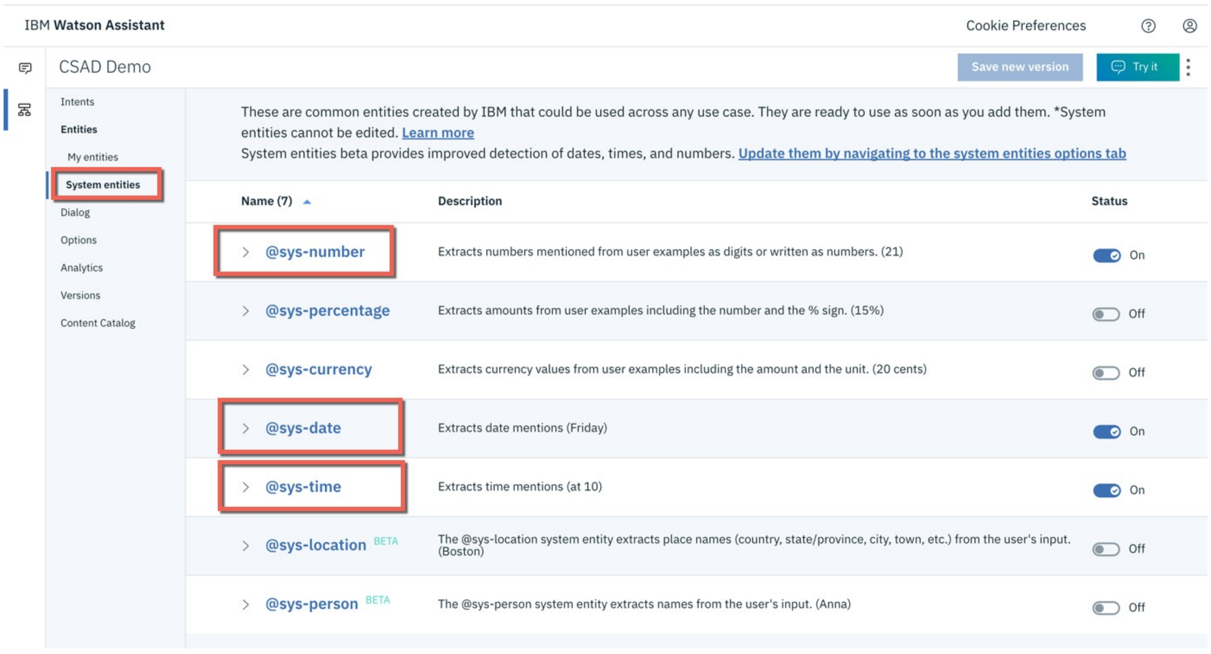
a) Add an Intent with the following values (see Step 21 ff):

Field	Value
Intent name	book_table
Description	Book a table in one of the restaurants
User example	I'd like to make a reservation I want to reserve a table for dinner Can 3 of us get a table for lunch? Do you have openings for next Wednesday at 7? Is there availability for 4 on Tuesday night? I'd like to come in for brunch tomorrow Can I reserve a table?

b) Add an Entity for Locations (See Step 29 ff)

Field	Value	Synonym
Entity name	locations	
Value	First Street	first, 1st
Value	Main Street	Main

c) Enable the System entities @sys-date, @sys-number, @sys-time



These are common entities created by IBM that could be used across any use case. They are ready to use as soon as you add them. *System entities cannot be edited. Learn more
System entities beta provides improved detection of dates, times, and numbers. Update them by navigating to the system entities options tab

Name (7)	Description	Status
@sys-number	Extracts numbers mentioned from user examples as digits or written as numbers. (21)	On
@sys-percentage	Extracts amounts from user examples including the number and the % sign. (15%)	Off
@sys-currency	Extracts currency values from user examples including the amount and the unit. (20 cents)	Off
@sys-date	Extracts date mentions (Friday)	On
@sys-time	Extracts time mentions (at 10)	On
@sys-location BETA	The @sys-location system entity extracts place names (country, state/province, city, town, etc.) from the user's input. (Boston)	Off
@sys-person BETA	The @sys-person system entity extracts names from the user's input. (Anna)	Off

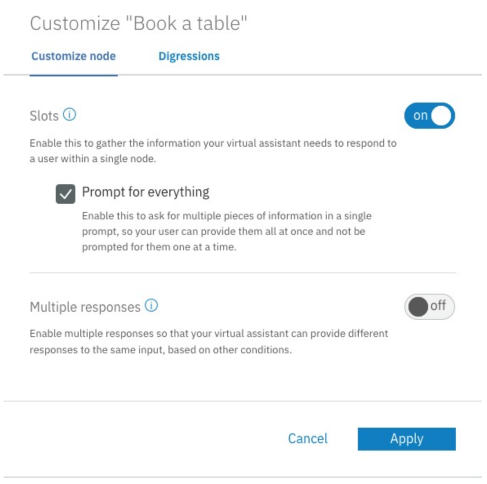
d) Add a dialog node for #book_table (Step 48 ff)

Click the menu ⋮ on the *Turn On* node and
click *Add node below* to create a new peer node. Enter the following values.

Field	Value
Name this node…	Book a Table
If bot recognizes:	#book_table

e) Click ⚙ Customize at the top of dialog node definition panel.

f) **Enable** *Slots* and **select** *Prompt for everything*. The click Apply



Customize "Book a table"

Slots ⓘ — on

Enable this to gather the information your virtual assistant needs to respond to a user within a single node.

☑ Prompt for everything

Enable this to ask for multiple pieces of information in a single prompt, so your user can provide them all at once and not be prompted for them one at a time.

Multiple responses ⓘ — off

Enable multiple responses so that your virtual assistant can provide different responses to the same input, based on other conditions.

g) Now you can enter the slots (Then check for:) ⊕ Add slot for more lines.

Check for	Save it as	If not present ask
@locations	$locations	Which store you want to got to? First or Main?
@sys-date	$date	What day you want to come in?
@sys-time	$time	What time did you want to arrive?
@sys-number	$number	How many people in your party?

h) In the field *If no slots are prefilled, ask this first*: **enter**

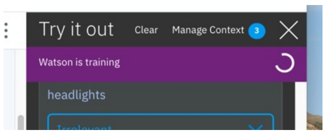
```
I need some more information to continue. I will need
the location, date, time, and number of people
```

i) In the field *Then respond with*: enter

```
Great, I have a table of $number people, on $date at
$time at our $locations store.
```

j) You can try (see Step 40 ff) this with a view sample inputs. Always Clear

If you see indication for "Watson is training", wait until completion:

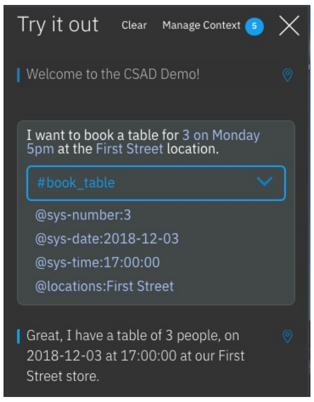


```
I want to book a table for 3 on Monday 5pm at the First Street location.

I want to book a table *clear previous dialog

I want a table for 3 please  *clear previous dialog
```

The result should look something like this:



Welcome to the CSAD Demo!

I want to book a table for 3 on Monday 5pm at the First Street location.

@sys-number:3
@sys-date:2018-12-03
@sys-time:17:00:00
@locations:First Street

Great, I have a table of 3 people, on 2018-12-03 at 17:00:00 at our First Street store.

Section 2: Backend Integration with Cloud Functions

(When Optional Step 70 has been done!)

In section 4 we will build a node.js server task to use the Watson Assistant skill built in Section 1. Watson Assistant also allows to call Cloud Functions (serverless computing) from within the context of a node definition.

To show such an integration we store the restaurant table booking dialog result (Step 71) in a Cloudant database using a Cloud Function. This can be achieved without programming.

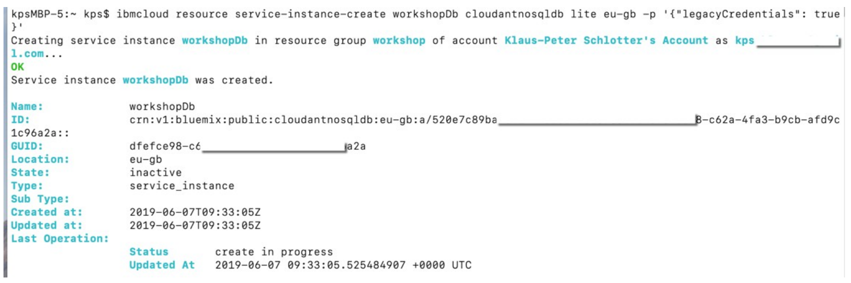
Step 71 Make sure you are logged in to your with the ibmcloud cli api to your account an respective region.

Step 72 In a terminal create a Cloudant service with the following command:

```
ibmcloud resource service-instance-create workshopDb cloudantnosqldb lite <your
region> -p '{"legacyCredentials": true}'
```

```
valid regions: eu-gb, eu-de, us-south, etc
```

Note: Legacy credentials provide a password attribute for default behaviour. If you have an existing Cloudant service without legacy credentials see Step 85

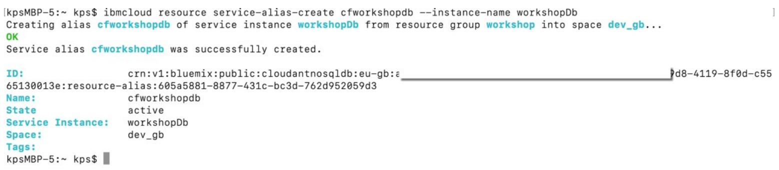


```
kpsMBP-5:~ kps$ ibmcloud resource service-instance-create workshopDb cloudantnosqldb lite eu-gb -p '{"legacyCredentials": true
}'
Creating service instance workshopDb in resource group workshop of account Klaus-Peter Schlotter's Account as kps[redacted]
l.com...
OK
Service instance workshopDb was created.

Name:                workshopDb
ID:                  crn:v1:bluemix:public:cloudantnosqldb:eu-gb:a/520e7c89ba[redacted]8-c62a-4fa3-b9cb-afd9c
1c96a2a::
GUID:                dfefce98-c6[redacted]a2a
Location:            eu-gb
State:               inactive
Type:                service_instance
Sub Type:
Created at:          2019-06-07T09:33:05Z
Updated at:          2019-06-07T09:33:05Z
Last Operation:
                     Status       create in progress
                     Updated At   2019-06-07 09:33:05.525484907 +0000 UTC
```

Step 73 Create a Cloud Foundry alias of your Cloudant services

```
ibmcloud resource service-alias-create cfworkshopdb --instance-name workshopDb
```

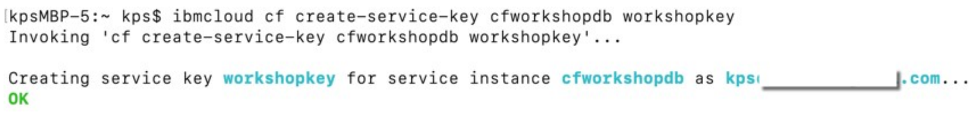


```
kpsMBP-5:~ kps$ ibmcloud resource service-alias-create cfworkshopdb --instance-name workshopDb
Creating alias cfworkshopdb of service instance workshopDb from resource group workshop into space dev_gb...
OK
Service alias cfworkshopdb was successfully created.

ID:                  crn:v1:bluemix:public:cloudantnosqldb:eu-gb:a[redacted]d8-4119-8f0d-c55
65130013e:resource-alias:605a5881-8877-431c-bc3d-762d952059d3
Name:                cfworkshopdb
State                active
Service Instance:    workshopDb
Space:               dev_gb
Tags:
kpsMBP-5:~ kps$
```

Step 74 Create Credentials for this service with the following command

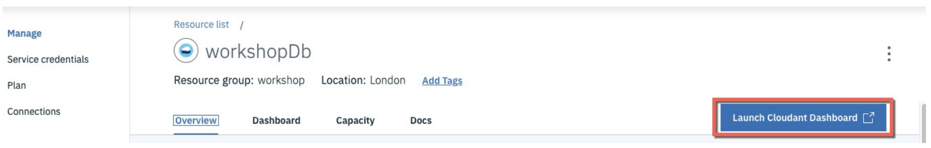
```
ibmcloud cf create-service-key cfworkshopdb workshopkey
```



```
kpsMBP-5:~ kps$ ibmcloud cf create-service-key cfworkshopdb workshopkey
Invoking 'cf create-service-key cfworkshopdb workshopkey'...

Creating service key workshopkey for service instance cfworkshopdb as kps[redacted].com...
OK
```

Step 75 In the cloud console open the Dashboard of your Cloudant service



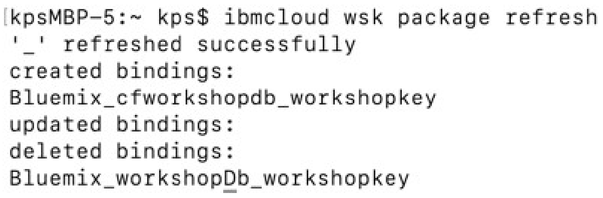
Step 76 At to top **click** Create Database . **Enter** a name f.e. *reservations* and **click** Create

Step 77 In a Terminal install the IBM Cloud Functions CLI plugin

```
ibmcloud plugin install Cloud-Functions -r bluemix
```

Step 78 With the following command you can automatically create Cloud Functions from certain IBM Services such as Cloudant, Weather Service, Watson Services to name a view.

```
ibmcloud wsk package refresh
```

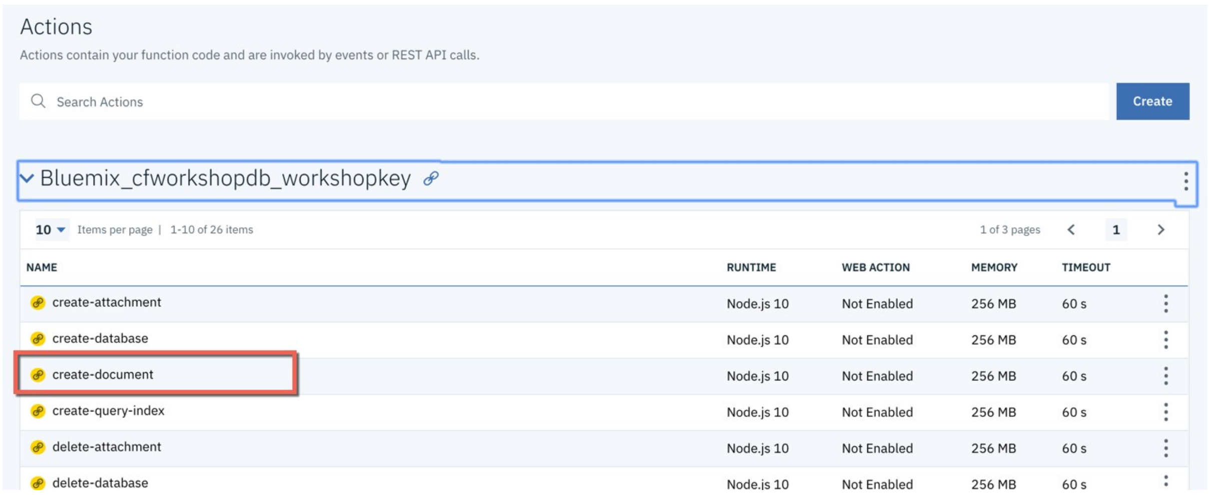


```
kpsMBP-5:~ kps$ ibmcloud wsk package refresh
'_' refreshed successfully
created bindings:
Bluemix_cfworkshopdb_workshopkey
updated bindings:
deleted bindings:
Bluemix_workshopDb_workshopkey
```

Step 79 In your cloud console goto Functions . Then **click** *Actions* in the left menu. You should see the functions that the previous command has created from some deployed services.

Note: Make sure you have selected your ORG and SPACE at the top that you have a connection to from the Command Line Interface.

Step 80 We are interested in the *create-document* function of the Cloudant service. Click ⋮ → Manage Action



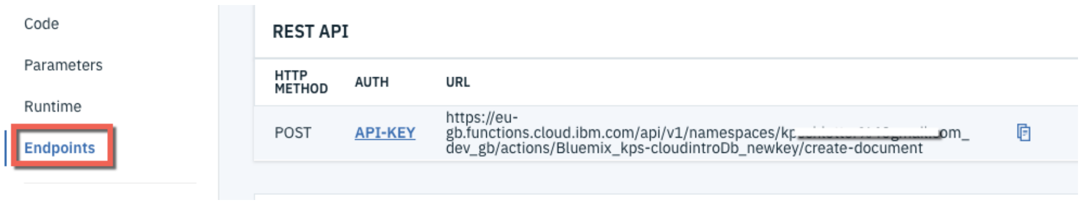
Actions

Actions contain your function code and are invoked by events or REST API calls.

Search Actions
Create
Bluemix_cfworkshopdb_workshopkey
10 Items per page | 1-10 of 26 items
1 of 3 pages
1

NAME	RUNTIME	WEB ACTION	MEMORY	TIMEOUT
create-attachment	Node.js 10	Not Enabled	256 MB	60 s
create-database	Node.js 10	Not Enabled	256 MB	60 s
create-document	Node.js 10	Not Enabled	256 MB	60 s
create-query-index	Node.js 10	Not Enabled	256 MB	60 s
delete-attachment	Node.js 10	Not Enabled	256 MB	60 s
delete-database	Node.js 10	Not Enabled	256 MB	60 s

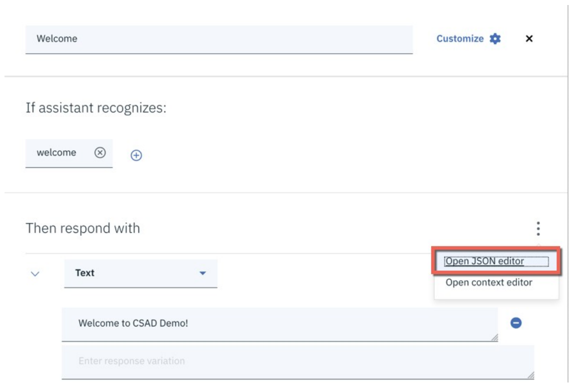
Step 81 **Select** *Endpoints* and **copy** the *URL* and the *API-KEY* for later use in our Assistant Skill. (click API-KEY to get the value).



REST API

HTTP METHOD	AUTH	URL
POST	API-KEY	https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/kp[redacted]om_dev_gb/actions/Bluemix_kps-cloudintroDb_newkey/create-document

Step 82 In your Assistant Skill **open** the *Welcome* node and the *Open JSON editor*. Replace the content with the following:

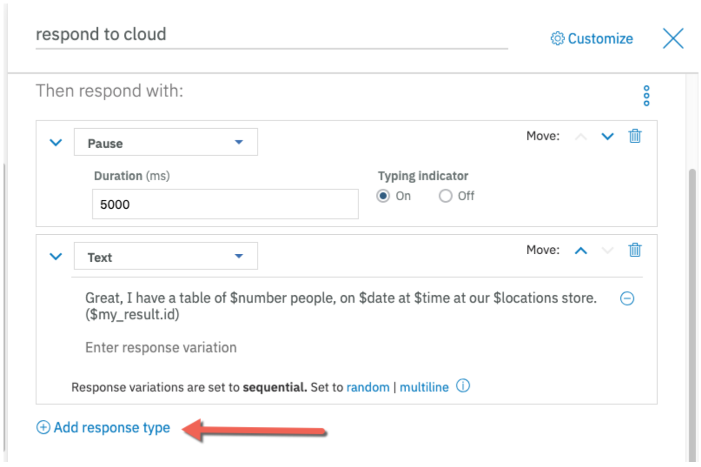


There is a lab03_codesnippets.txt file that contains all the code snippets of this lab section.

```
{
  "output": {
    "generic": [
      {
        "values": [
          {
            "text": "Welcome to CSAD Demo!"
          }
        ],
        "response_type": "text",
        "selection_policy": "sequential"
      }
    ]
  },
  "context": {
    "private": {
      "my_credentials": {
        "api_key": "<yourCloudFunctionsAPIKey>"
      }
    },
    "Alarmonoff": "off",
    "app_action": ""
  }
}
```

Step 83 **Click** ⋮ on the *Book a table* node and **select** *Add child node*.

Field	Value
Name this node…	Respond to cloud
If bot recognizes:	true
Then respond with (Pause)	5000
Then respond with (Text) (click + to add an response type)	Great, I have a table of $number people, on $date at $time at our $locations store. ($my_result.id)
And finally	Wait for user input


respond to cloud

Then respond with:

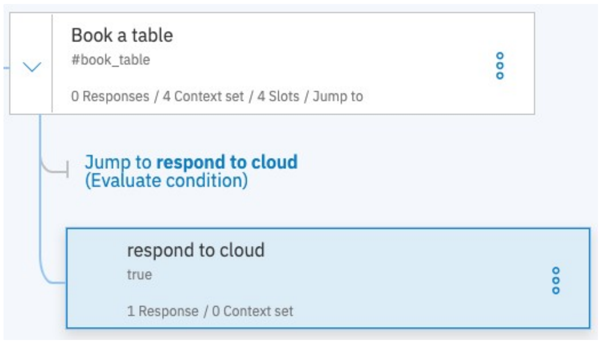
Pause — Duration (ms): 5000 — Typing indicator: [✓ On] Off

Text — Great, I have a table of $number people, on $date at $time at our $locations store. ($my_result.id)

Enter response variation

Response variations are set to **sequential.** Set to random | multiline

Step 84 **Click** ⋮ on the *Book a table* node and **select** *Jump to (recognizes condition)*. **Select** the *Respond to cloud node*.


Book a table
#book_table
0 Responses / 4 Context set / 4 Slots / Jump to

Jump to **respond to cloud** (Evaluate condition)

respond to cloud
true
1 Response / 0 Context set

Step 85 **Open** the *Book a table* node and on ⋮ *Open JSON editor* and **replace** the content with the following:

```
{
  "output": {
    "text": {
      "values": [],
      "selection_policy": "sequential"
    }
  },
  "actions": [
    {
        "name": "/kpschxxxx.com_dev_gb/actions/Bluemix_kps-cloudin-
troDb_newkey/create-document",
      "type": "server",
      "parameters": {
        "doc": {
          "date": "$date",
          "time": "$time",
          "number": "$number",
          "locations": "$locations"
        },
        "dbname": "reservations"
      },
      "credentials": "$private.my_credentials",
      "result_variable": "$my_result"
    }
  ]
}
```

Actions name is the part of the URL from Step 80 after .../namespaces.

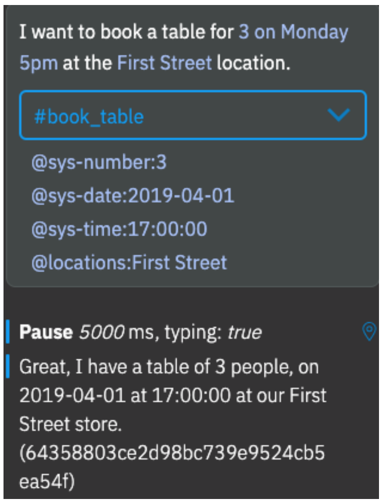
Note: In case you get an Error Code 400 you do not have legacy credentials, containing *password* and *port*, for your Cloudant service. Add the following two paramters from your Cloudant credentials to the code above:

```
"dbname": "reservations",
"iamApiKey": "<apikey from your Cloudant credentials>",
"url": "<url from your Cloudant credentials>"
```

Step 86 Try the dialog with the following sentence:

```
I want to book a table for 3 on Monday 5pm at the First Street location.
```

An you should see a result like this:



I want to book a table for 3 on Monday 5pm at the First Street location.

#book_table

@sys-number:3

@sys-date:2019-04-01

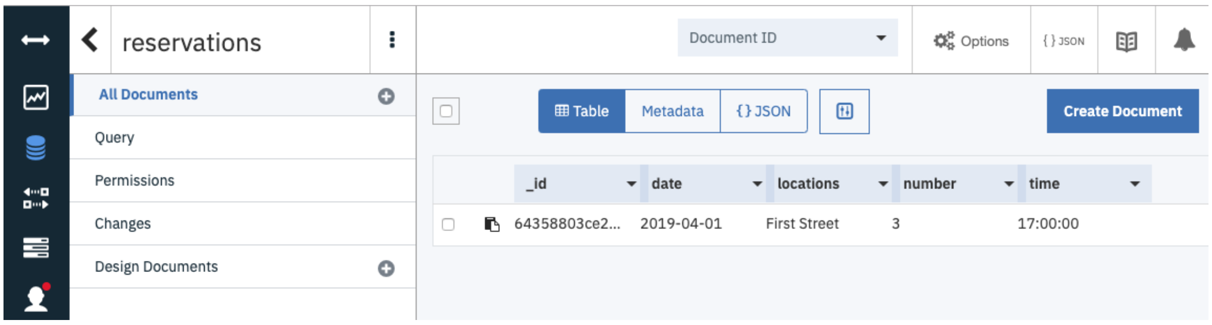
@sys-time:17:00:00

@locations:First Street

Pause *5000* ms, typing: *true*

Great, I have a table of 3 people, on 2019-04-01 at 17:00:00 at our First Street store. (64358803ce2d98bc739e9524cb5ea54f)

Step 87 Go back to your Cloudant dashboard. There you should see the document that was created.



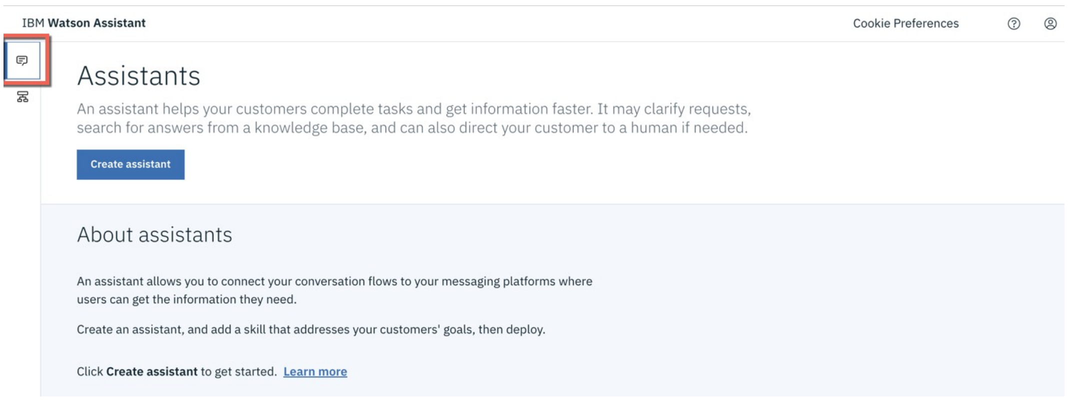
_id	date	locations	number	time
64358803ce2...	2019-04-01	First Street	3	17:00:00

Section 3: Create an Assistant from our Skill

IBM Watson Assistant has a V2 API that provides a session context and therefore the context has not to be passed with any client request.

This is now also the API version used in the following Node.js application.

Step 88 Go to the Assistants section.



IBM Watson Assistant

Cookie Preferences

Assistants

An assistant helps your customers complete tasks and get information faster. It may clarify requests, search for answers from a knowledge base, and can also direct your customer to a human if needed.

Create assistant

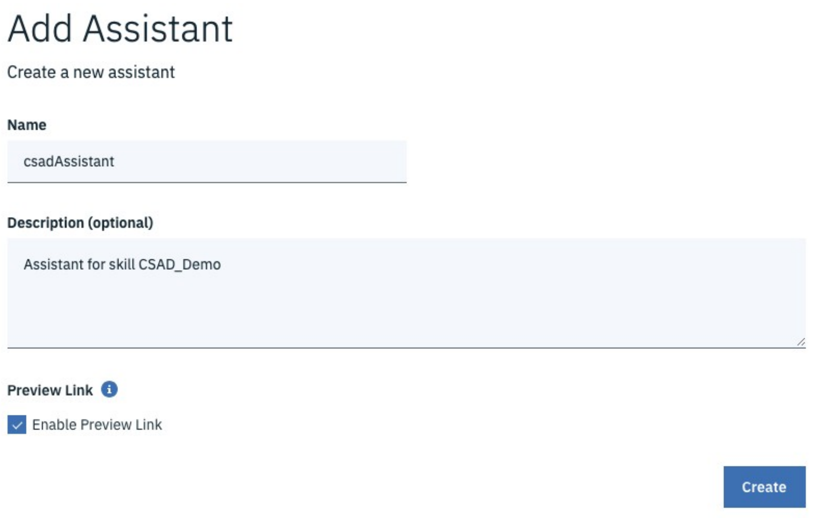
About assistants

An assistant allows you to connect your conversation flows to your messaging platforms where users can get the information they need.

Create an assistant, and add a skill that addresses your customers' goals, then deploy.

Click **Create assistant** to get started. Learn more

Step 89 **Click** Create assistant to create a new Assistant, **enter** name and description then **click** Create assistant.



Add Assistant

Create a new assistant

Name

csadAssistant

Description (optional)

Assistant for skill CSAD_Demo

Preview Link

☑ Enable Preview Link

Create

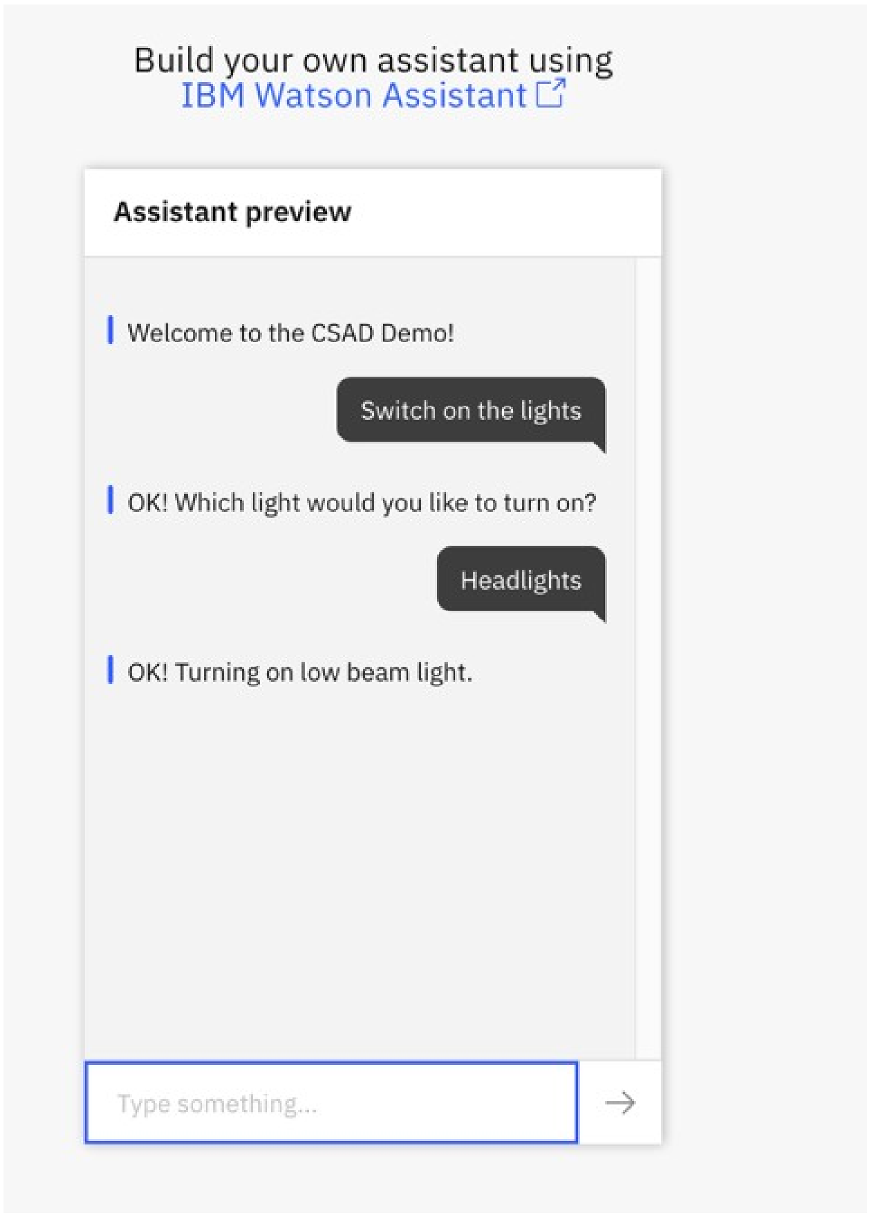
Step 90 Click Add dialog skill. and **select** our *CSAD Demo* Skill. With the Integrations function you can now directly integrate this Assistant into Facebook, Slack and other applications.

This is out of scopt of this introductory demo!!

Step 91 **Click** Preview Link to make this Assistant available on the internet. **Enter** a name and click Save Changes.

Step 92 **Copy** the link and **open** it in a browser.

Step 93 It should look like the following:



Build your own assistant using IBM Watson Assistant

Assistant preview

Welcome to the CSAD Demo!

Switch on the lights

OK! Which light would you like to turn on?

Headlights

OK! Turning on low beam light.

Step 94 Try some of your dialogs.

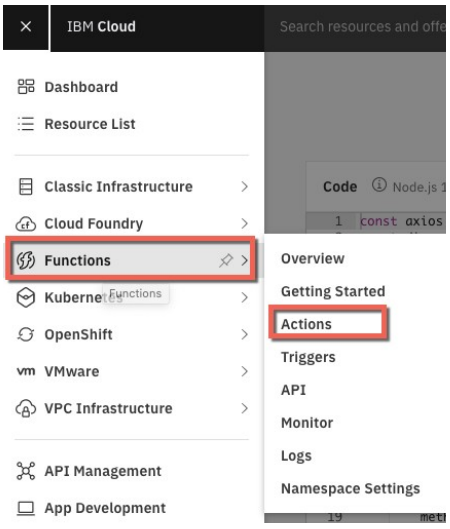
Section 4: Create a Cloud Function used as Webhook

In Lab 3 the integration with external systems was done with a Node.js server in between the web client and the Watson Assistant service where the server interprets the intents and entities returned from the agent and acts accordingly. F.e. the **@help:time** entity is caught by the server and the message for the client is created with the system time.

Watson Assistant recently was enhanced with a Webhook capability where you can call the Webhook directly from an Assistant's dialog node. The Webhook has to have an URL that is publicly available.

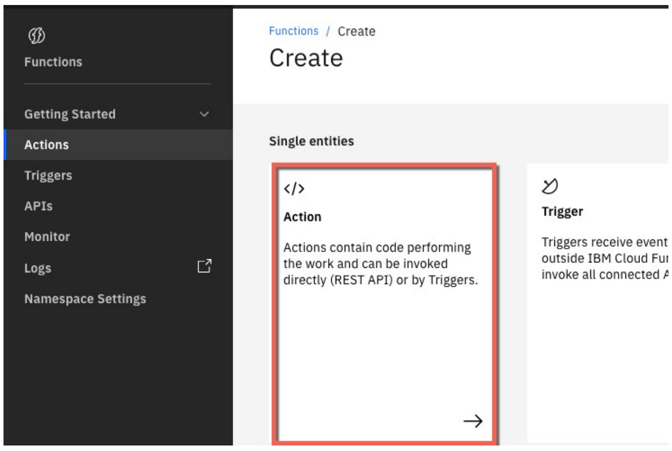
For simplicity we will realize the webhook for this Lab and the Lab 5a by implementing a Cloud Function in the IBM Cloud.

Step 95 In the Cloud console navigate to Functions → Actions



Step 96 **Click** the Create button

Step 97 **Click** on *Action*



Actions contain code performing the work and can be invoked directly (REST API) or by Triggers.

Triggers receive event outside IBM Cloud Fu invoke all connected

Step 98 **Name** it f.e. *csadbot*, make sure *Runtime* is Node.js 10 or later.
Click Create

Step 99 The Code section of the Action opens. **Paste** the following code into it by replacing the default. (You can also **Paste** from here). Click Save

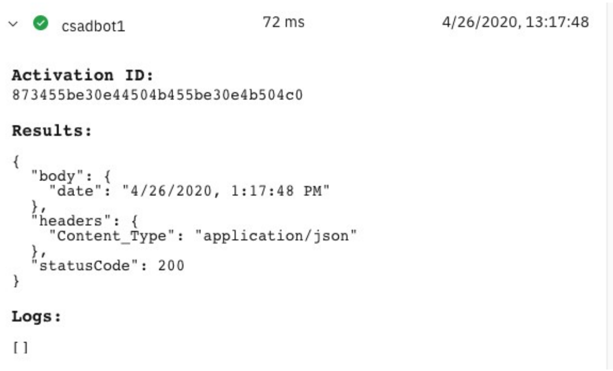
```
function main(params) {
  if (params.action === "gettime") {
     const deTime = new Date().toLocaleString("de-DE", {timeZone:
"Europe/Berlin"});
    return {
      statusCode: 200,
      headers: {
        Content_Type: "application/json"
      },
      body: { message: deTime }
    };
  } else {
    return {
      statusCode: 404,
      headers: { "Content-Type": "application/json" },
      body: {message: `action ${params.action} not defined` }
    };
  }
}
```

Step 100 **Click** Invoke with parameters to customize the parameters .

Step 101 **Type** the following between the curly braces then **click** Apply

```
"action": "gettime"
```

Step 102 Click Invoke and the following should be the result:



```
csadbot1                72 ms                4/26/2020, 13:17:48

Activation ID:
873455be30e44504b455be30e4b504c0

Results:
{
  "body": {
    "date": "4/26/2020, 1:17:48 PM"
  },
  "headers": {
    "Content_Type": "application/json"
  },
  "statusCode": 200
}

Logs:
[]
```

Step 103 At the top of the page besides the action's name **click** Web Action

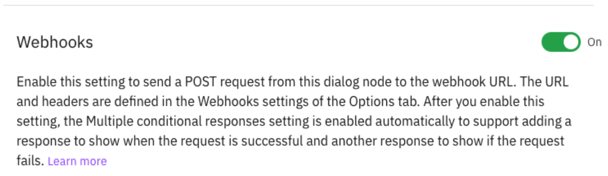
Step 104 **Select** *Enable as Web Action* and then click Save

Step 105 Now the HTTP Method *ANY* gets active (Public). **Copy** the URL to the clipboard

Step 106 **Click** the Time dialog node in your Assistant created in Step 67.

Step 107 Click Customize at the top of the definition page.

Step 108 **Enable** Webhooks then **click** Apply


Webhooks — On

Enable this setting to send a POST request from this dialog node to the webhook URL. The URL and headers are defined in the Webhooks settings of the Options tab. After you enable this setting, the Multiple conditional responses setting is enabled automatically to support adding a response to show when the request is successful and another response to show if the request fails. Learn more

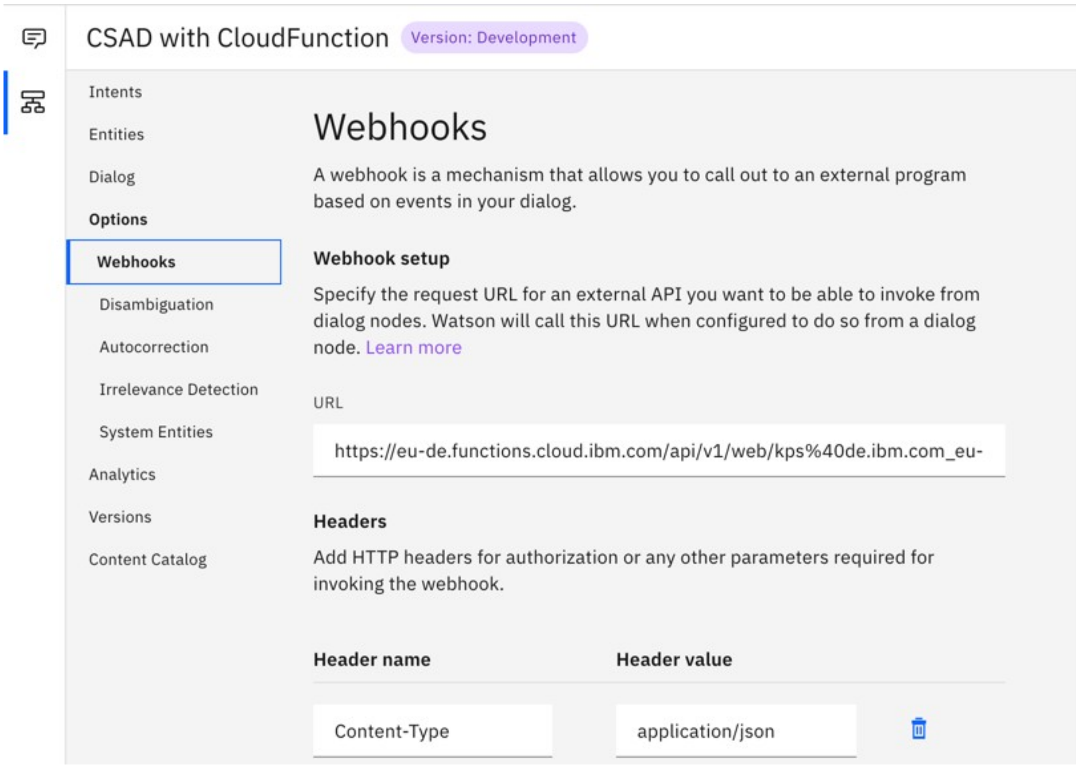
Step 109 As *callout parameter* specify the following:

Key	Value
action	gettime

Step 110 In *Assistant responds* enter the following responses

If assistant recognizes	Respond with
$webhook_result_1	The time is <? $webhook_result_1.message ?>
anything_else	Something went wrong.

Step 111 In the *Options → Webhook* section **paste** the *URL* from **Step 105**.


CSAD with CloudFunction Version: Development

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-de.functions.cloud.ibm.com/api/v1/web/kps%40de.ibm.com_eu-

Headers

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

Header name	Header value
Content-Type	application/json

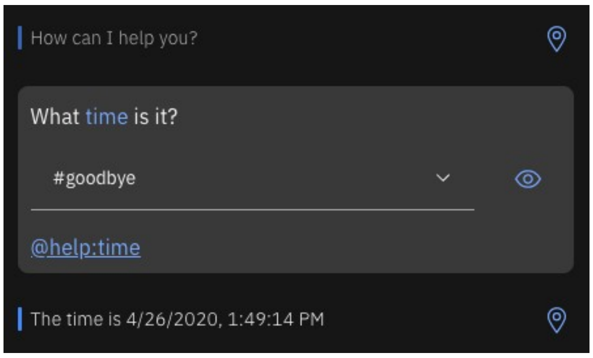
Step 112 Add the following header:

Header name	Header value
Content-Type	application/json

Step 113 Test your agent again with the *Try it out* feature. Enter the following:

```
I need help
What time is it?
```

You should see the following returned from the Webhook.



How can I help you?

What time is it?

#goodbye

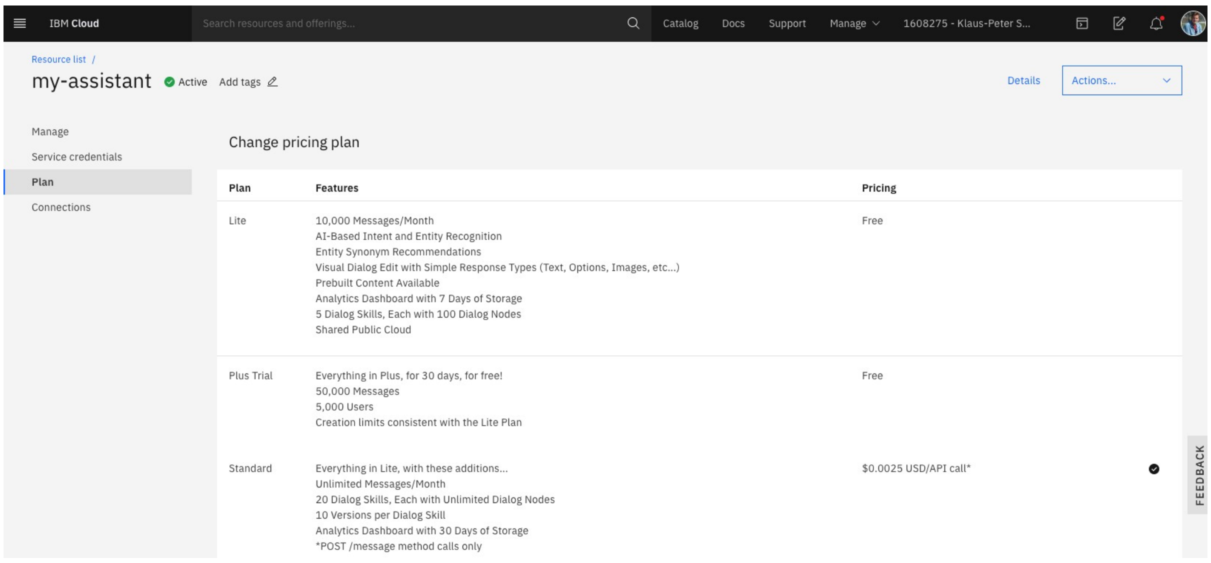
@help:time

The time is 4/26/2020, 1:49:14 PM

Section 5: Integrate the Chatbot in your Web Site using Webchat

The Webchat feature of Watson Assistant provides similar capabilities as the Assistant's Preview Link does (See Step 94). It provides a Chatbot without a specific application such as created in Lab 3 using Node.js.

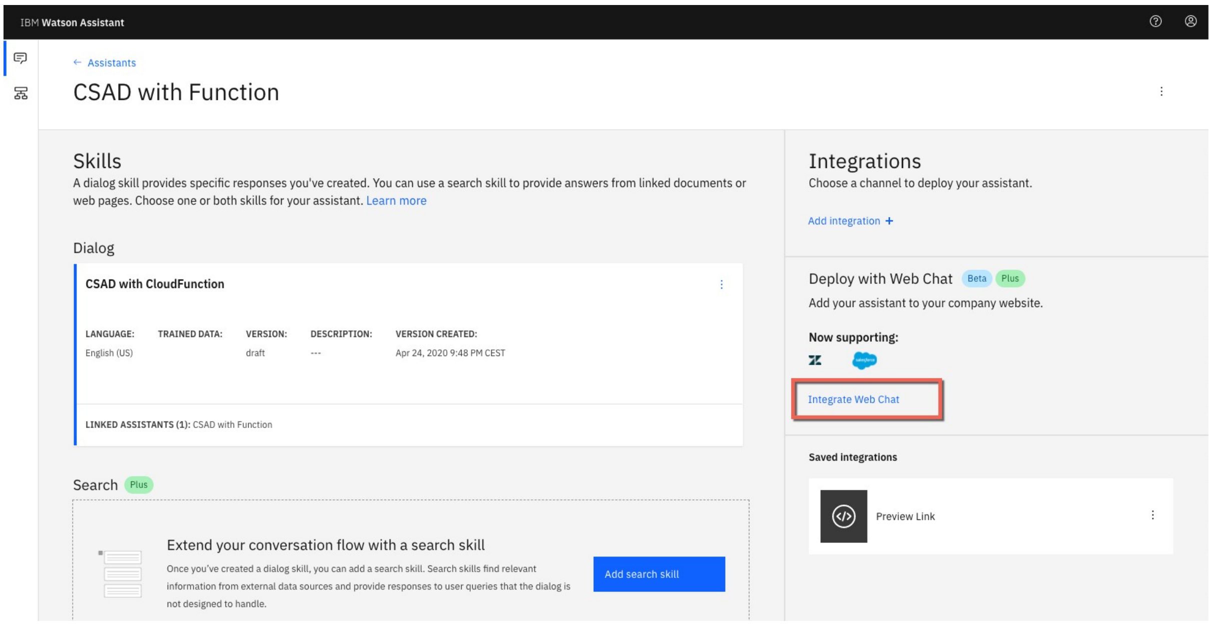
To use this feature you have to upgrade your Watson Assistant to the **Plus Trial** plan.



Change pricing plan

Plan	Features	Pricing
Lite	10,000 Messages/Month AI-Based Intent and Entity Recognition Entity Synonym Recommendations Visual Dialog Edit with Simple Response Types (Text, Options, Images, etc...) Prebuilt Content Available Analytics Dashboard with 7 Days of Storage 5 Dialog Skills, Each with 100 Dialog Nodes Shared Public Cloud	Free
Plus Trial	Everything in Plus, for 30 days, for free! 50,000 Messages 5,000 Users Creation limits consistent with the Lite Plan	Free
Standard	Everything in Lite, with these additions... Unlimited Messages/Month 20 Dialog Skills, Each with Unlimited Dialog Nodes 10 Versions per Dialog Skill Analytics Dashboard with 30 Days of Storage *POST /message method calls only	$0.0025 USD/API call*

Step 114 On your *Assistant* page **click** Integrate Web Chat.



A dialog skill provides specific responses you've created. You can use a search skill to provide answers from linked documents or web pages. Choose one or both skills for your assistant. Learn more

CSAD with CloudFunction

LANGUAGE:	TRAINED DATA:	VERSION:	DESCRIPTION:	VERSION CREATED:
English (US)		draft	---	Apr 24, 2020 9:48 PM CEST

LINKED ASSISTANTS (1): CSAD with Function

Extend your conversation flow with a search skill

Once you've created a dialog skill, you can add a search skill. Search skills find relevant information from external data sources and provide responses to user queries that the dialog is not designed to handle.

Choose a channel to deploy your assistant.

Add your assistant to your company website.

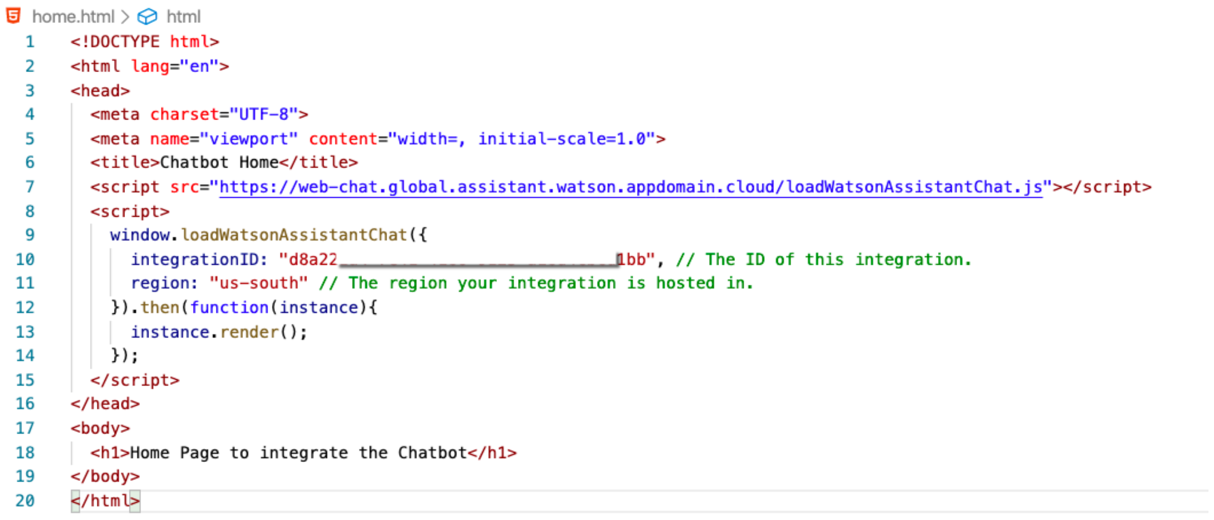
Step 115 On the *Web Chat integration* page Click Create.

Step 116 Now you have certain options available. We want to *Add the chat UI to your web page*. **Copy** the script to the clipboard.

Step 117 Create a simple HTML file with a text editor. F.e. the following.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=, initial-scale=1.0">
    <title>Chatbot Home</title>
  </head>
  <body>
    <h1>Home Page to integrate the Chatbot</h1>
  </body>
</html>
```

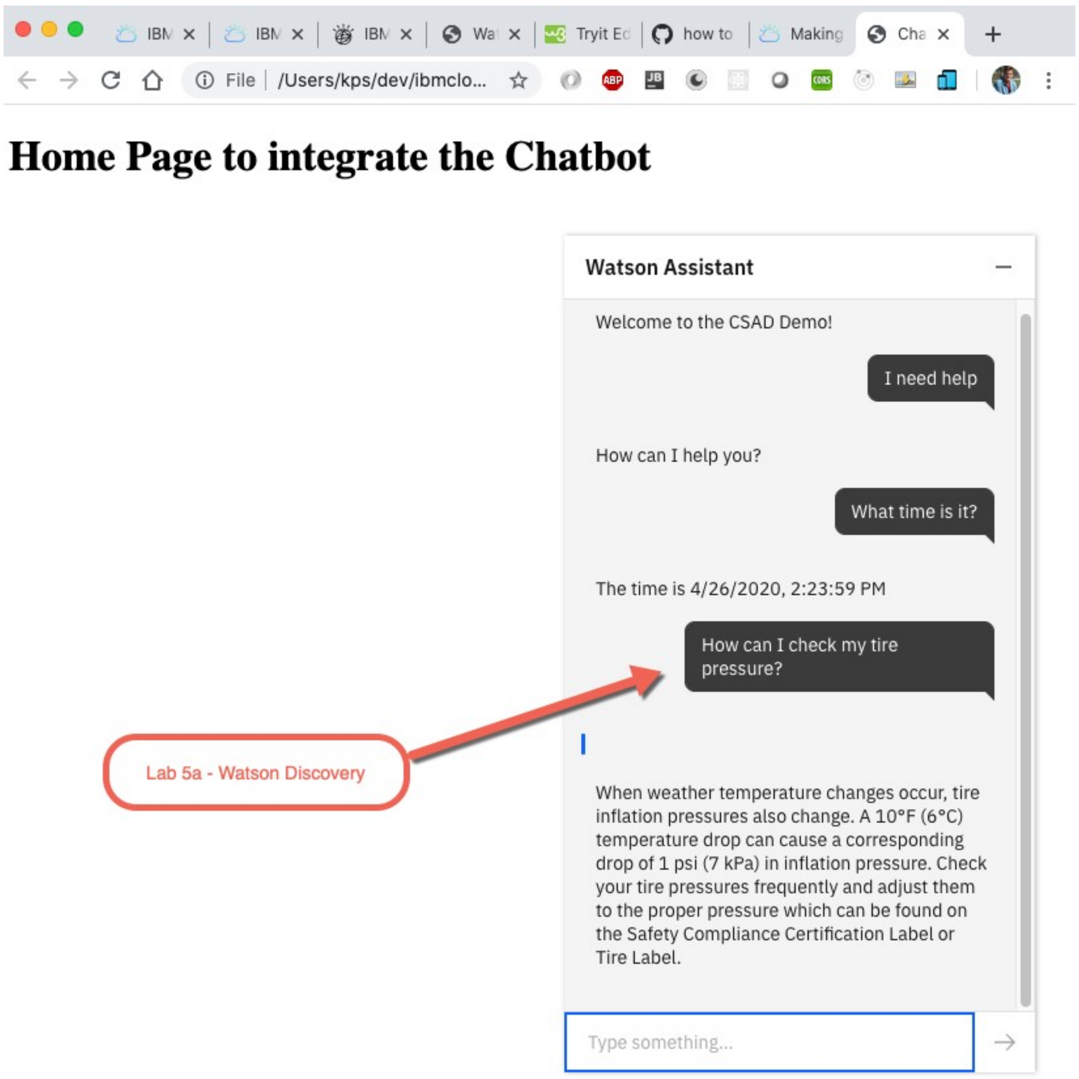
Step 118 Now paste the JavaScript from your Assistant before the </html> tag.



home.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=, initial-scale=1.0">
  <title>Chatbot Home</title>
  <script src="https://web-chat.global.assistant.watson.appdomain.cloud/loadWatsonAssistantChat.js"></script>
  <script>
    window.loadWatsonAssistantChat({
      integrationID: "d8a22[redacted]1bb", // The ID of this integration.
      region: "us-south" // The region your integration is hosted in.
    }).then(function(instance){
      instance.render();
    });
  </script>
</head>
<body>
  <h1>Home Page to integrate the Chatbot</h1>
</body>
</html>
```

Step 119 Now open the page in Google Chrome.



Home Page to integrate the Chatbot

Watson Assistant

Welcome to the CSAD Demo!

I need help

How can I help you?

What time is it?

The time is 4/26/2020, 2:23:59 PM

How can I check my tire pressure?

When weather temperature changes occur, tire inflation pressures also change. A 10°F (6°C) temperature drop can cause a corresponding drop of 1 psi (7 kPa) in inflation pressure. Check your tire pressures frequently and adjust them to the proper pressure which can be found on the Safety Compliance Certification Label or Tire Label.

You have successfully completed this lab.