
# Playbook

**Christopher Crane**
IBM Data and AI Client Success
ccrane@us.ibm.com

**Morgan Langlais**
IBM Data and AI Client Success
morganlanglais@ibm.com

## Welcome to a Watson Assistant Multi-skill capability

- Target numerous skills from a single skill.
- No custom orchestration code.
- Have numerous dev teams work independently with discreet development & release cadences.

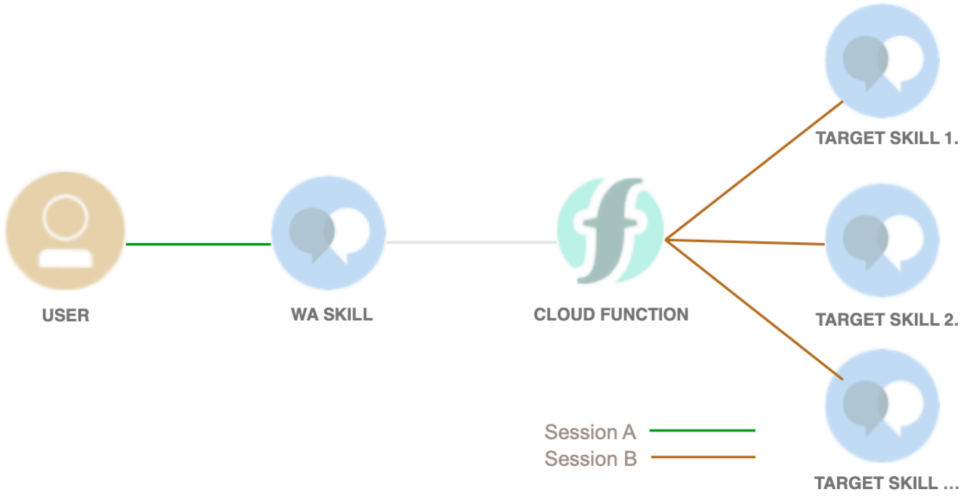
## Introduction and Overview

Multiskill Bot enables target skills to be accessed from a single “router” skill.
This run book will walk through:

1. Gathering Requirements
2. Create Cloud Function
3. Create a New Watson Assistant Skill
4. Get the Cloud Functions endpoint
5. Add the webhook to your skill
6. Test
7. Make it your own

The intended user experience for Multiskill enabled Assistants should be seamless. The user enters an utterance, which triggers a response from the pertinent target skill without the user knowing or caring where the response originated. Behind the scenes numerous skills may be involved in what appears to the user to be a single conversation.

The simple context diagram below shows the components and context of this solution.




**A user connects to a skill, which can either serve the user, or pass the conversation off to any number of target skills**

This flow is made possible by linking Watson Assistant to a custom application that sits in Cloud Functions. The Cloud Function is invoked by a webhook from Watson Assistant. Once invoked, the Cloud Function leverages the target Assistant to correctly respond to the user.

**Terminology**
The skill that the user interacts directly with (Session A above) can be referred to as:

- Parent,
- Vessel, or
- Router skill.

The skill accessed via the cloud function can be referred to as the:

- Child,
- Nested or
- Target skill.

## 1. Gathering Requirements

In your IBM Cloud account you should have the following

- Watson Assistant instance.
- Cloud Functions instance

On your local computer you should have the following

- IBM Cloud CLI
- Download the files in the repository here which includes the following:
    - multiskill_cloudfunction.zip (Cloud Function code and dependencies)
    - skill-polyglot-bot.json (the Watson assistant skill)

## 2. Create Cloud Function

Cloud Functions is a serverless programming platform for developing code that scalably executes on demand. We will be leveraging this tool because it integrates easily with Watson Assistant via a Webhook and carries a low cost.

2.1. Login to your IBM Cloud from the CLI using the following command:
- ***ibmcloud login –sso***

2.2. Choose the target space:
- ***ibmcloud* target --cf**

2.3. Place your multiskill_cloudfunction.zip file from step 1 in a folder where you can find it.

2.4. Using IBM Cloud CLI Locate the folder containing your zip file.

2.5. From within the folder, create your cloud function with the following command:
- ***ibmcloud* fn action create [your cloud function name here] multiskill_cloudfunction.zip --kind nodejs:10**

2.6. You can always update this file later with the following command:
- ***ibmcloud* fn action update [your cloud function name here] multiskill_cloudfunction.zip --kind nodejs:10**

| Action | CLI Command |
|---|---|
| **Login to IBM Cloud via CLI** | ibmcloud login –sso |
| **Choose the target space** | ibmcloud target --cf |
| **Create CF Action** | ibmcloud fn action create [your cloud function name here] multiskill_cloudfunction.zip --kind nodejs:10 |
| **Update CF Action** | ibmcloud fn action update [your cloud function name here] multiskill_cloudfunction.zip --kind nodejs:10 |

**CLI command summary**

## 3. Create a new Watson Assistant skill

This Watson assistant skill will be responsible for connecting your target skill choices to the user as well as setting and passing pertinent variables to the cloud function.

3.1. Create a Watson Assistant instance if you do not have one

3.2. If you haven't, download the dialog skill "multiskill.json " from step 1.

3.3. Launch your Watson Assistant Instance

3.4. Navigate to the Skills page (vertical menu on left, bottom icon) & Click "Create Skill"

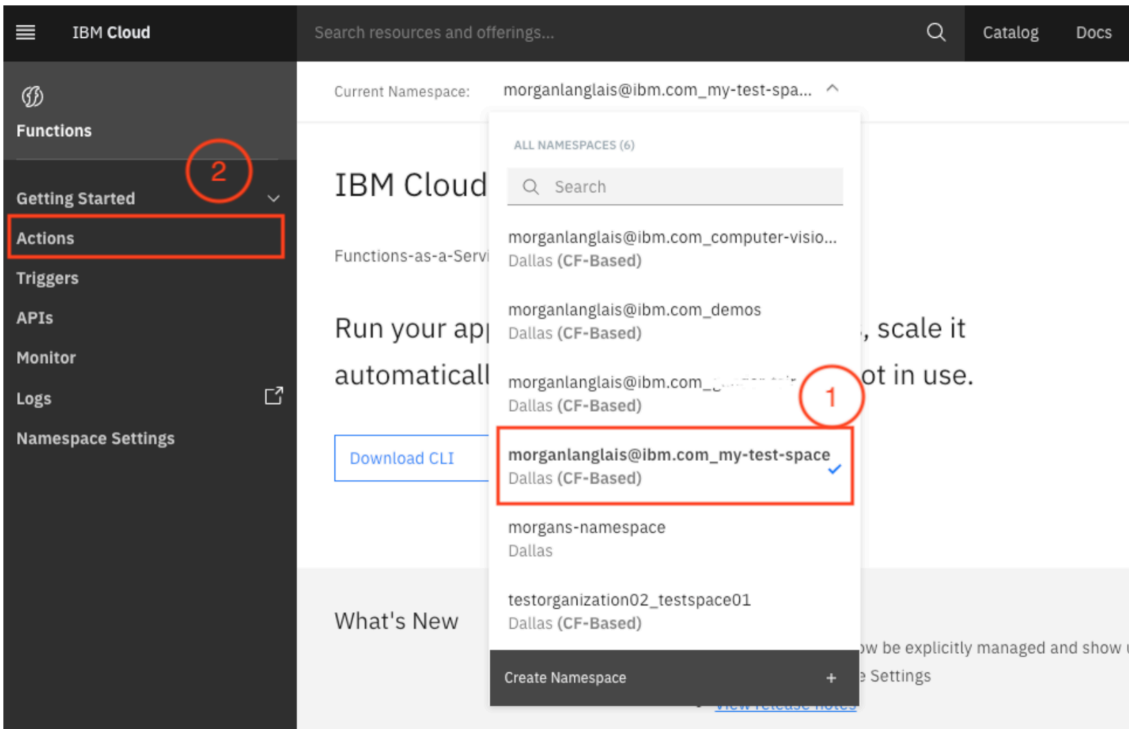
3.5. Select skill type, "Dialog Skill" and Click "Next"

3.6. Choose "Import Skill" and select: "Choose JSON File"

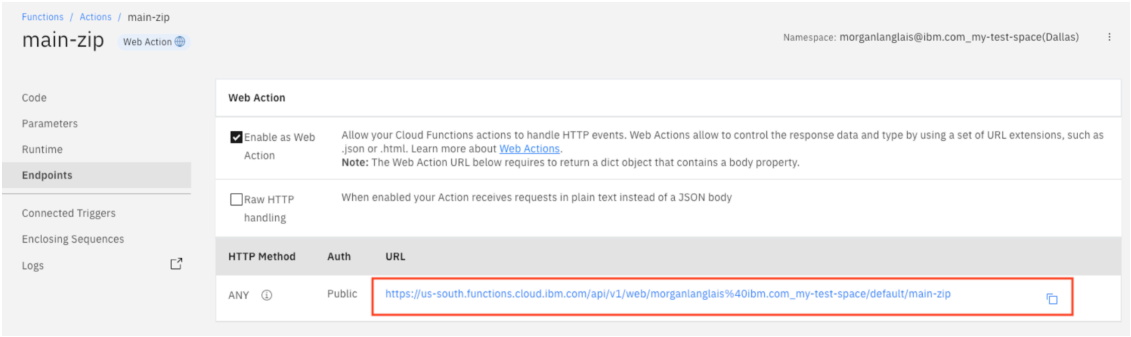
3.7. Navigate to the “multiskill.json file that you downloaded and click "Import"
3.8. Attach the skill to an Assistant. To create an assistant go to the Assistants page (Vertical menu on left, top icon). Once created, link it to the skill.

## 4. Get the Cloud Functions endpoint for your webhook

4.1. Open your Cloud Functions dashboard here: https://cloud.ibm.com/functions/
4.2. Select the namespace you wish to deploy the function in from the drop-down (1), then click "Actions" on the left nav bar (2)

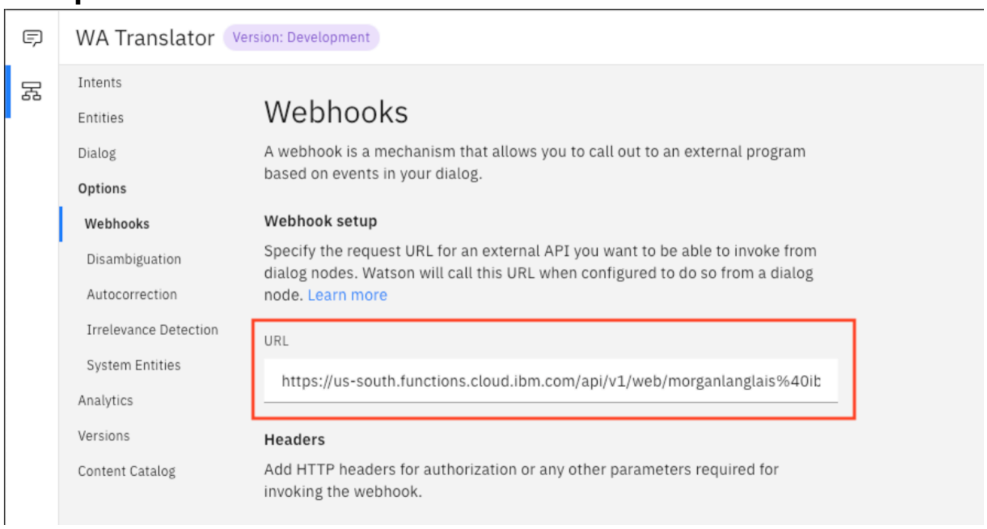



4.3. Click my-action or the name that you previously gave your action in step 2
4.4. Select Endpoints then check "Enable as web action" and click "Save"
4.5. Copy the URL in the "Web Action" section and save for later



## 5. Add the webhook to your skill

5.1. Open the skill you imported in Watson Assistant earlier
5.2. Navigate to **Options -> Webhooks**



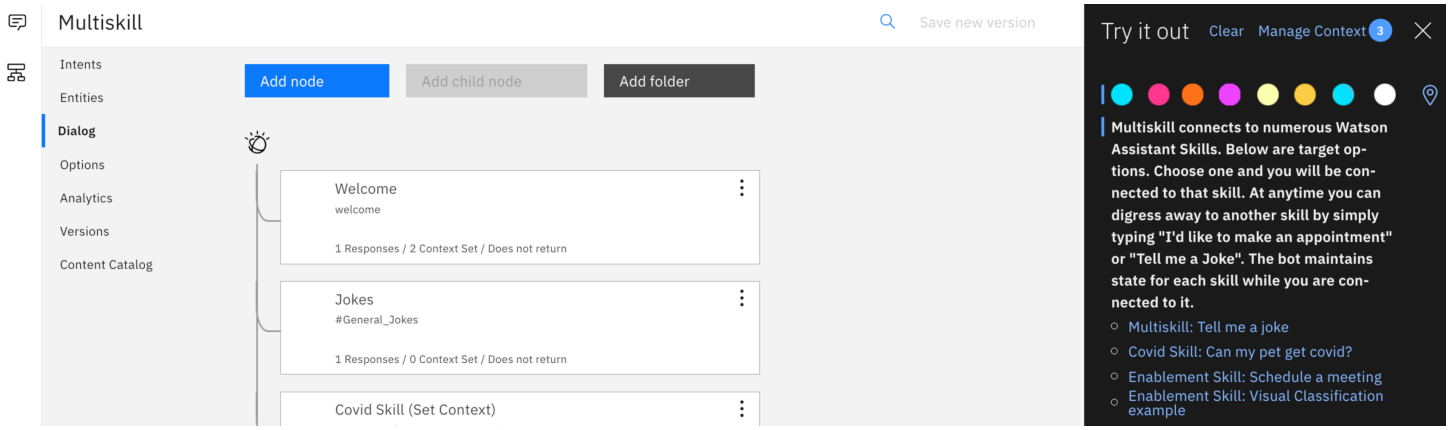

5.3. Replace this URL with your webhook URL that you copied previously
**Important**: Make sure you append ".json" to the end of this URL in order for Watson Assistant to process the data correctly.

**Done**! You are now ready to configure your skill.

## 6. Test

The components have been loaded with test configurations. Let's see if everything is working.
6.1. Navigate to the dialog tab of the skill. On the top right side you should see a "Try it" button: 

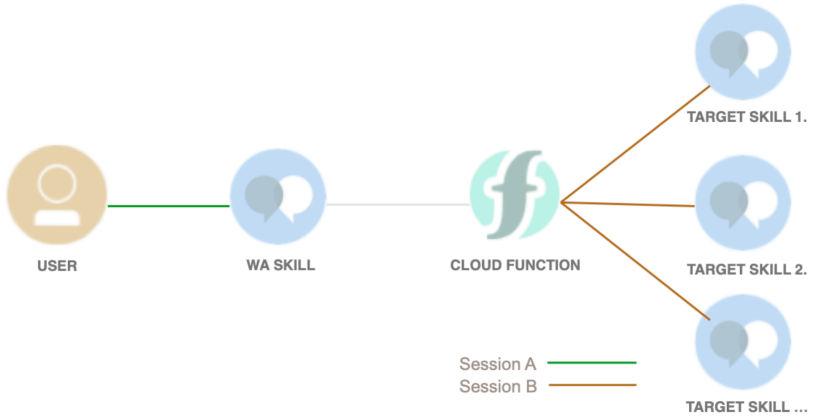
6.2. Press "Try it" and the following options should present:



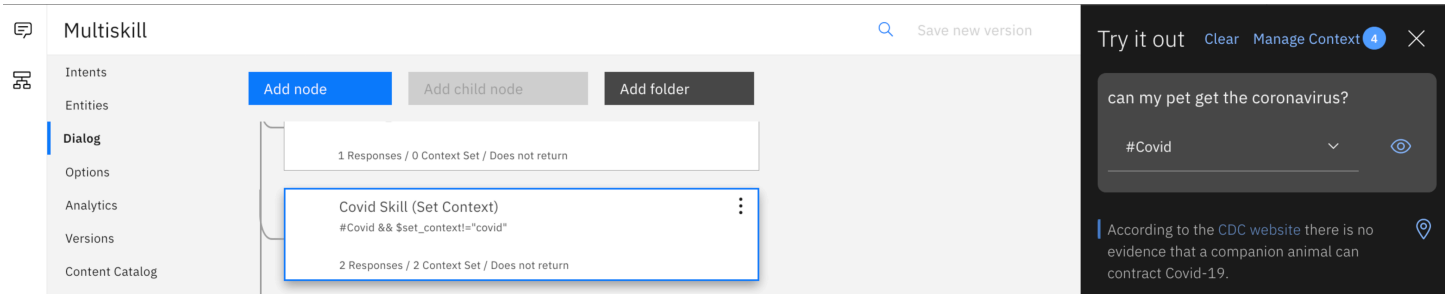

6.3. Try several of the options presented.
Responses will indicate proper setup. Congratulations! In the next section we will discuss how to make it your own.

## 7. Make it your own

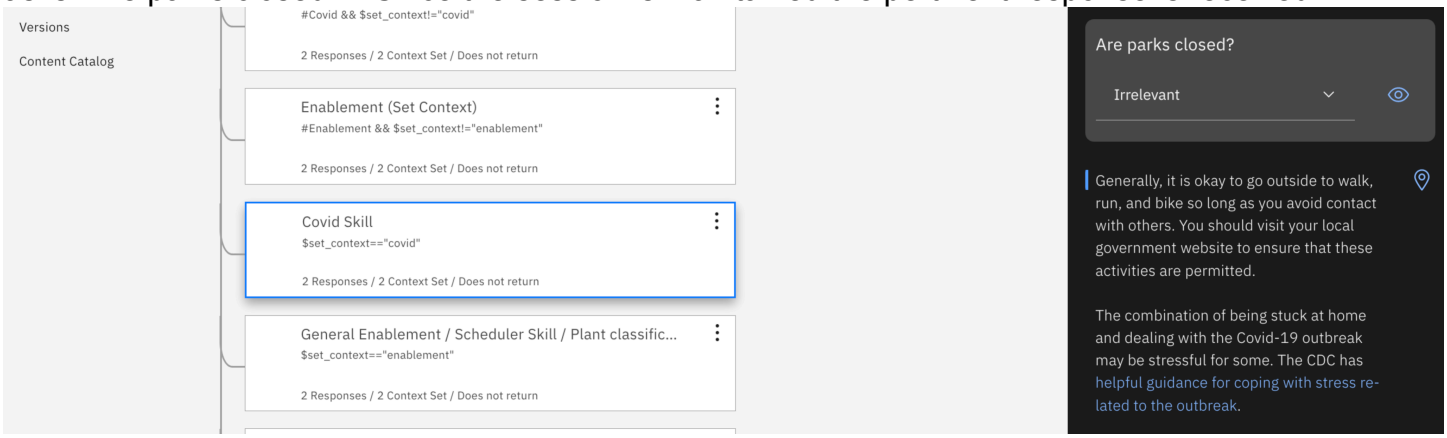
First a bit of background on how Multiskill works. As mentioned before, WA is leveraging Cloud Functions to access multiple skills. In order for the Cloud Function to understand which skill to target, Watson Assistant must pass it a number of parameters.




Watson Assistant is using dialog nodes to hold the pertinent values and the correct dialog nodes are triggered based on intents and variables that get set when a user gives input. In the following example, the user has asked “can my pet get the coronavirus?” The “Covid Skill (Set Context)” node gets triggered sending the request to a Covid specific skill, eliciting the pertinent response.




Once this connection has been made the session is maintained, allowing the user to ask pertinent follow on questions that on their own would not appear to be Covid related. Below the user then asks “Are parks closed?”. Since the session is maintained the pertinent response is received.

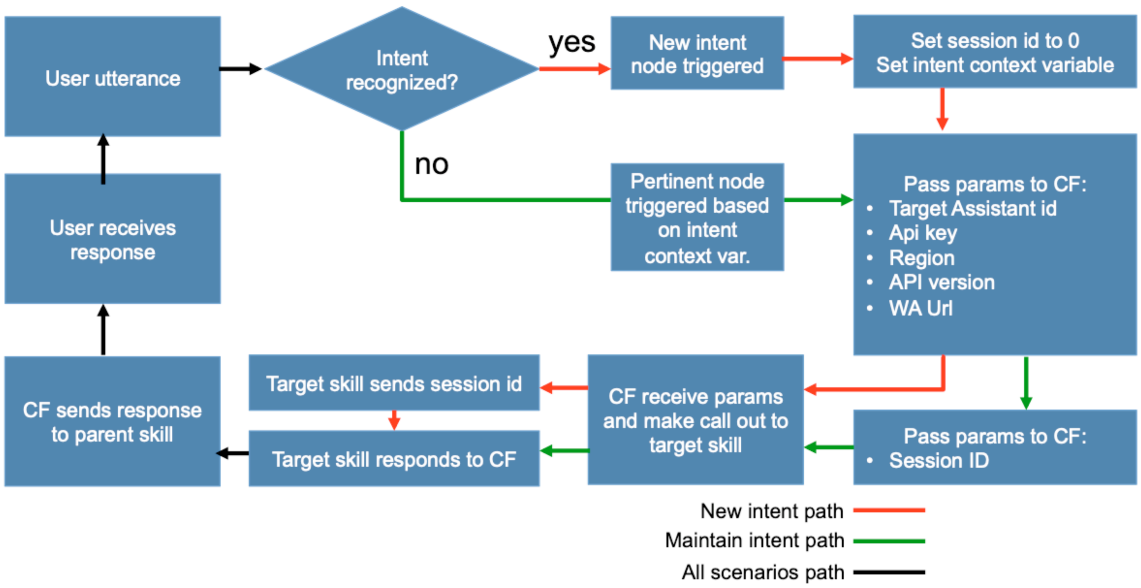



If the user asks a series of Covid related questions then asks if they can get an appointment (a

function served out of a different skill) the session is broken with the Covid skill and will be connected with the enablement/scheduling skill.

*This behavior is possible due to 2 components: Intents and dialog conditions. When the user triggers an intent the dialog sends the conversation to the pertinent skill and creates a session and maintains the state. Those variables remain until a new intent is triggered.*

- *For each target skill an intent needs to be created.*
- *For each target skill 2 (Two) dialog nodes need to be configured.*
  - *The first dialog node sets the intent specific context variable and resets the session id.*
  - *The second dialog node carries all subsequent conversation until a new intent is identified.*




Scenario flow diagram differs by intent recognition from within WA

**This image shows the first node triggered by a Covid question: *"Covid Skill (Set Context)".***

**Below that is the node triggered by all subsequent Covid questions: *"Covid Skill".***

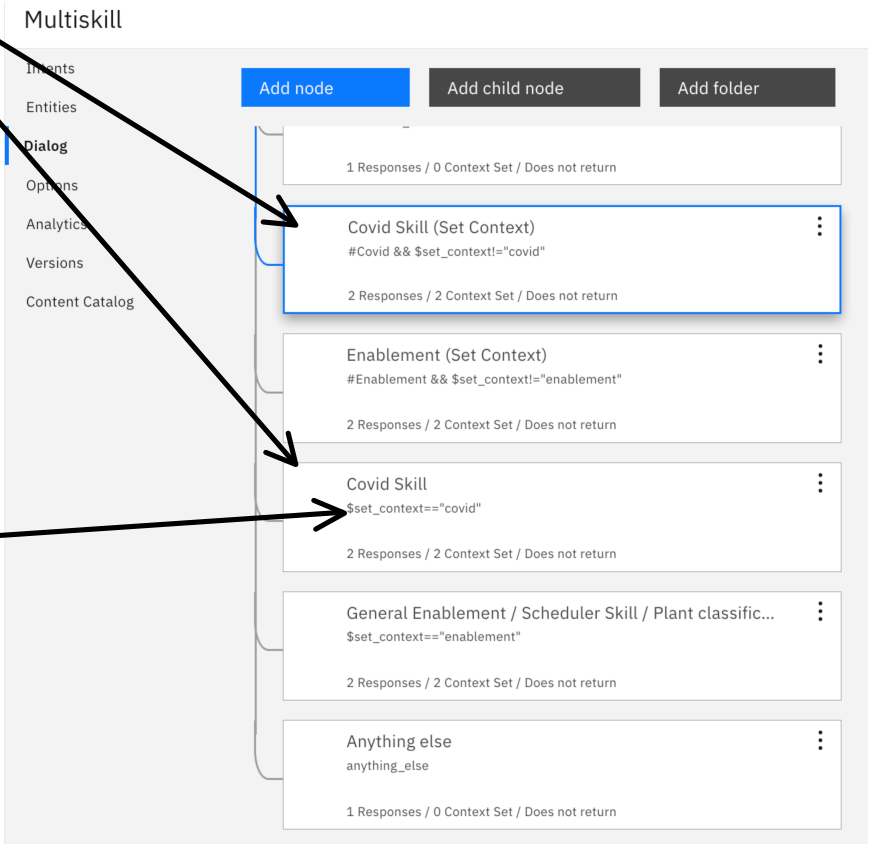
**The node conditions are different for each.**

**When the Covid intent is initially triggered, this first Covid node:**

1. **resets the session id,**
2. **processes the utterance and then**
3. **sets the $set_context to "Covid".**

**You can see here that the condition for the node only triggers upon the #Covid intent AND the absence of the $set_context value of "covid".**

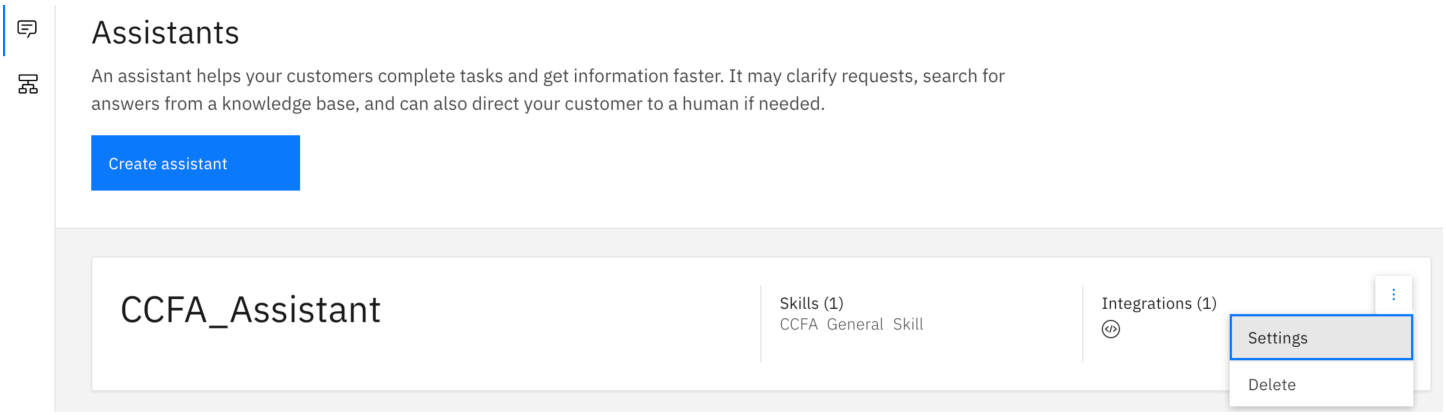
**As a result, the conversation will no longer return to this node. Instead they will go to the second Covid node which gets triggered by the $set_context value. This node is similar to the first but doesn't reset the session i.d. thus allowing session to be maintained.**

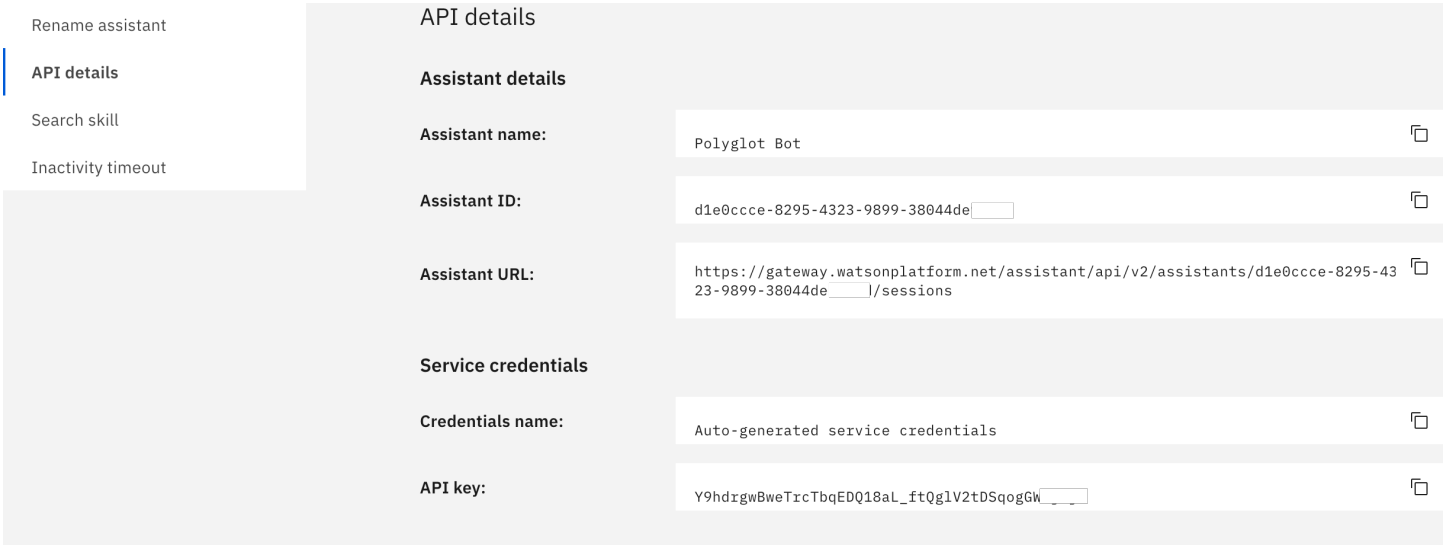


Lets start configuring. All the configuration values are loaded into the nodes allowing flexibility and ease of implementation.

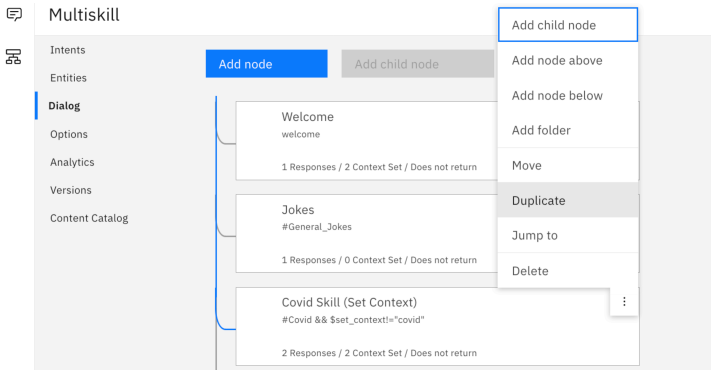
**Assistant Id**

7.1. First, navigate to the target assistant where we will obtain the assistant. Find it here in the settings for your Assistant. Also note the Assistant Url and API Key

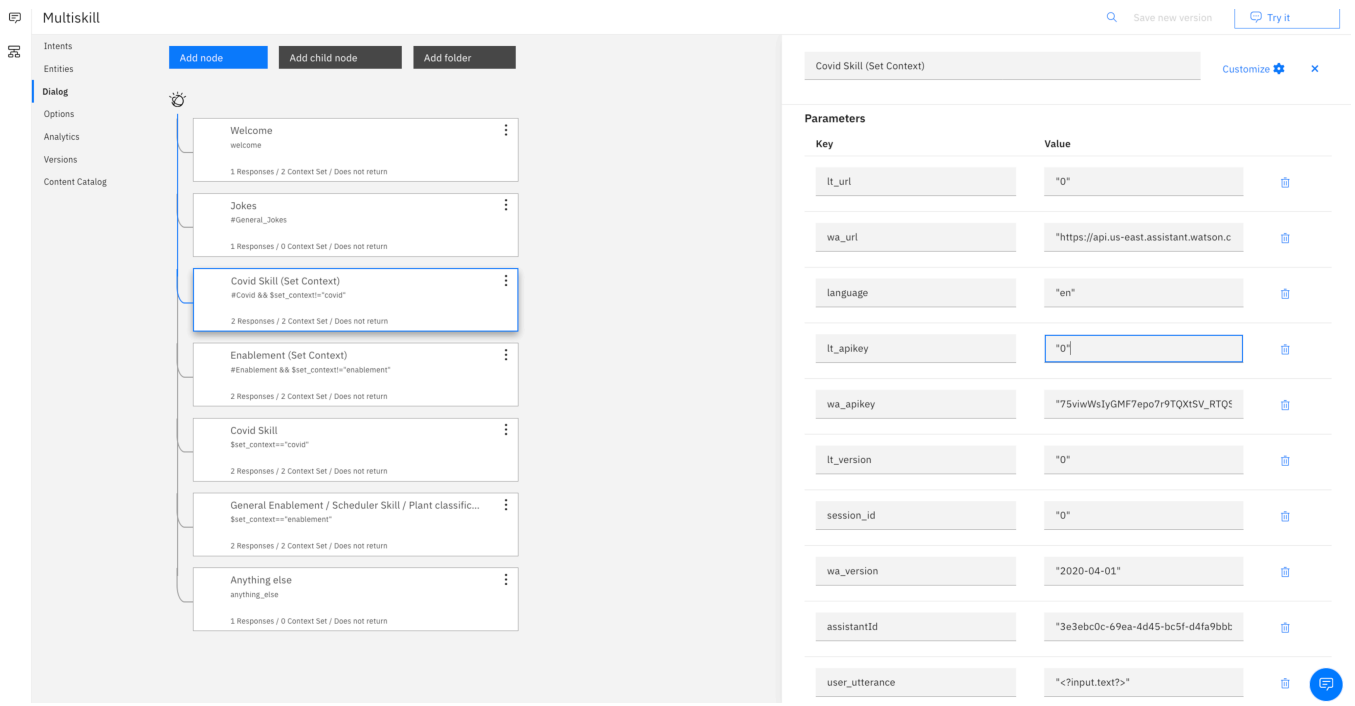



7.2. Copy the Assistant ID and then open the multiskill skill. Navigate to the “Covid Skill (Set Context)” node. Since we don’t have any intents trained yet we are going to skip configuring the node conditions for now. Open the node, (or copy and then open it as shown below) and paste the value for the assistant id over the existing one.

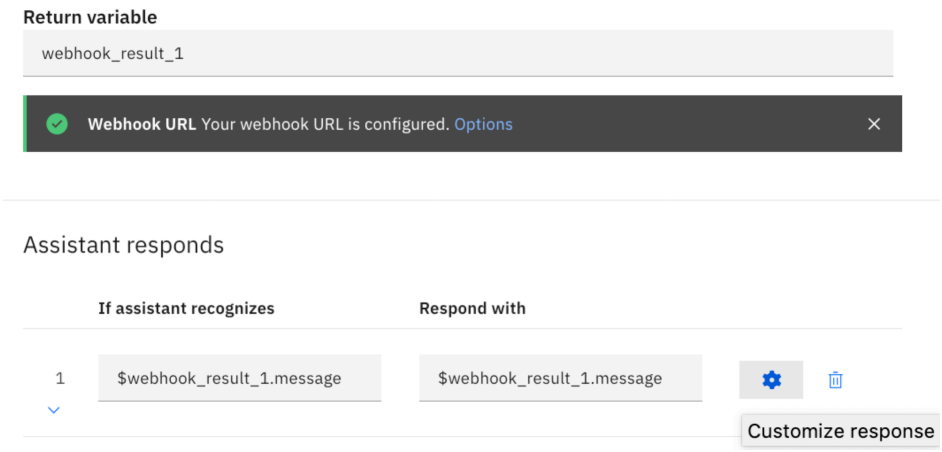


Do the same for the url and the apikey (version most likely will not be different). For the url you only need the core domain including everything up to but ending with .com or .net.

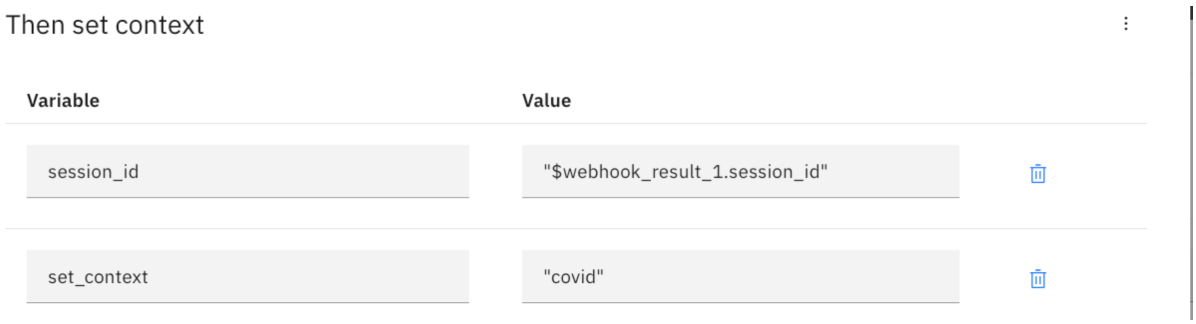



**Fill in the variables as shown above with your target Assistant information**

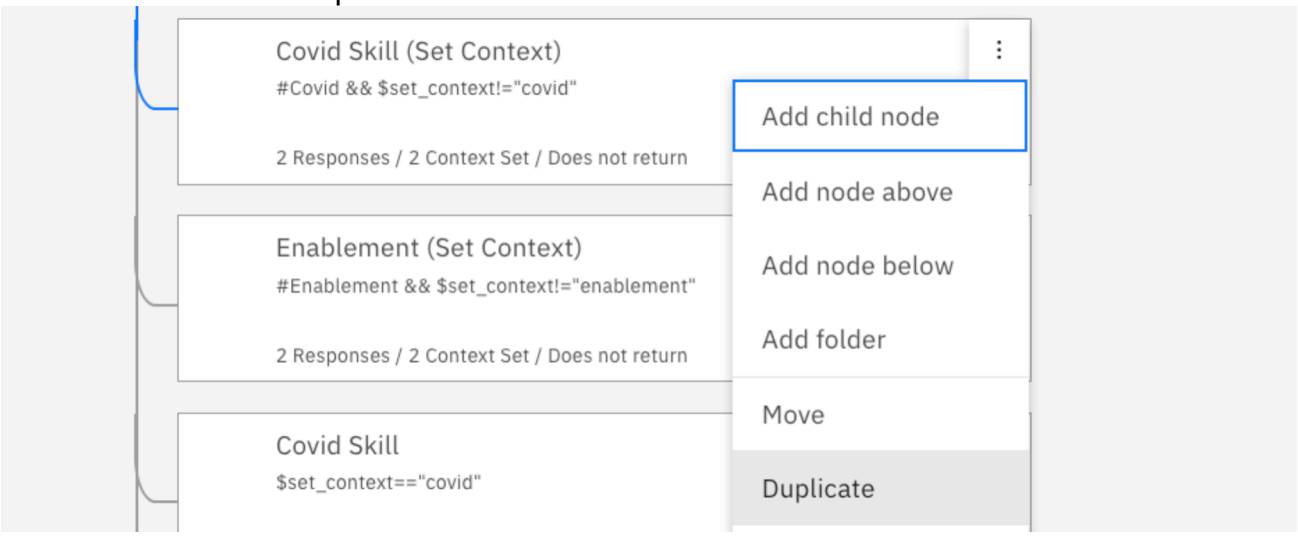
7.3. Once these values have been entered scroll down the node to the webhook response section and open the response cog as shown in blue below.




7.4. We will need to tailor the set_context variable to match your target skill's intended use. Rename it something meaningful and unique like "scheduling" or "Covid" or "credit_card_returns" etc. Be sure to press save and continue.

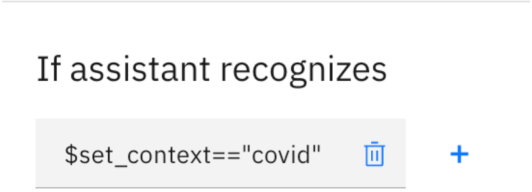


7.5. To save time duplicate the node as shown below to create the second node:



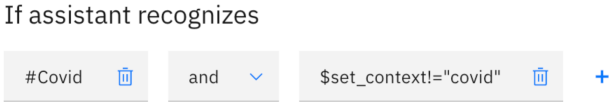

This duplicated node needs to be configured to catch all the requests after the Intent has been first been identified. Because dialog is evaluated top down it is helpful if this second node is below the first. This way when you are following your dialog you know that the top node get hits first when a new intent is triggered/classified.

In the copied node Change the name and the node conditions to match the format below. Instead of “Covid” yours should match the exact value you entered for the “set_context” variable in the section 7.4.




7.6. Now go create an intent that matches your target skill. The intent does not need to include all of the target skill’s training data. ***Only enough to trigger connecting to the skill.*** For example: Even though our Covid target skill is trained on ‘how to get tested’, ‘how to reduce the spread’, ‘where to get tested’ and information on closures, all the training data in the Multiskill needs to identify is if they are asking about Covid. Once the utterance is passed to the pertinent skill, that skill can handle more detailed intent and entity classifications.

7.7. Once your intent is trained return to the first (or top) node pertinent to your target skill. Change the node conditions to match the following format: #(the newly created intent) and $set_context!=”[your associated set context value]”.




**How your configured node should look but with your intent and context value**

**Congratulations, your skill should now be configured to reach out to your target skill.**

**Be sure to modify the welcome options to match your use case and delete all the vestige / example nodes that are not needed.**

**Keep in mind that although this skill specializes in reaching out to other Assistants you can also build native capabilities in this skill too. See the joke node as an example.**