

Building Cognitive Applications with IBM Watson Services: Conversation

In partnership with
IBM Skills Academy Program

International Technical Support Organization

**Building Cognitive Applications with IBM Watson Services:
Conversation**

May 2017



Cognitive Calculator chatbot

This chapter guides you through building the Cognitive Calculator chatbot sample application. The app demonstrates the use of Watson Conversation service in creating a calculator chatbot. The chatbot chats with the user in natural language, the Conversation service determines the user request and the application performs simple calculations to respond to the user.

The sample application demonstrates the integration of the Conversation service with a Node.js application.

The following topics are covered in this chapter:

[Getting started](#)

[Architecture](#)

[Two ways to deploy the application: Step-by-step and quick deploy](#)

[Step-by-step implementation](#)

[Quick deployment of application](#)

[References](#)

1.1 Getting started

To start, read through the objectives, prerequisites, and expected results of this use case.

1.1.1 Objectives

By the end of this chapter, you should be able to accomplish these objectives:

- 1 Create a Conversation service instance in IBM Bluemix.
- 2 Create a Conversation workspace, add intents, entities, system entities, and a dialog for the Cognitive Calculator chatbot application.
- 3 Integrate the Watson Conversation service in a Node.js application to perform the calculation functionality.

Prerequisites

To complete the steps in this chapter, be sure these prerequisites are met:

- 1 Review [“Basics of Conversation service”](#), and [“Conversation service workspace”](#)
- 2 Access to a web browser (Chrome, Firefox, or Internet Explorer)
- 3 Basic JavaScript skills
- 4 Understand Bluemix DevOps basics
- 5 Understand Git basics
- 6 Have a Bluemix account
- 7 Have an account on GitHub

Expected results

[Figure 1-1](#) shows the simple Cognitive Calculator chatbot application:

The user starts the conversation with the addition operation.

The user tries to add two numbers but specifies only one number without specifying the other number.

The chatbot application prompts the user to specify two numbers to be able to perform the addition operation.

The user specifies the two numbers to add.

The application adds the two numbers and returns the result to the user.

The user then wants to multiply two numbers.

The chatbot prompts the user to enter the numbers to multiply.

The user requests a subtraction operation which the chatbot application does not understand.

Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?

let's start with the addition

Kindly help me with the numbers that you want to add.

i'd like to add three

Oops. I found only 1 operand(number) and required at least 2 operand(number) to perform addition operation.
Kindly provide other number/s.

and 4

The result is : 7

thanks

You are welcome.

multiply 5, 6 and 7

The result is : 210

5 + 5+ 15

The result is : 25

thanks

Glad !! I could help you today.

what is your name?

I am Cognitive Calculator, I could help you with addition or multiplication of numbers.

how is weather

Well sometimes i do not have answer to your query as i am not trained on it.

divide 5 and 6

Sorry!! At this time i dont know how to perform divide operation. Currently i can help you with only addition and multiplication of numbers.

bye

Bye. See you again.

Type something

Figure 1-1 Cognitive Calculator chatbot

1.2 Architecture

[Figure 1-2](#) shows the components and runtime flow of the application.

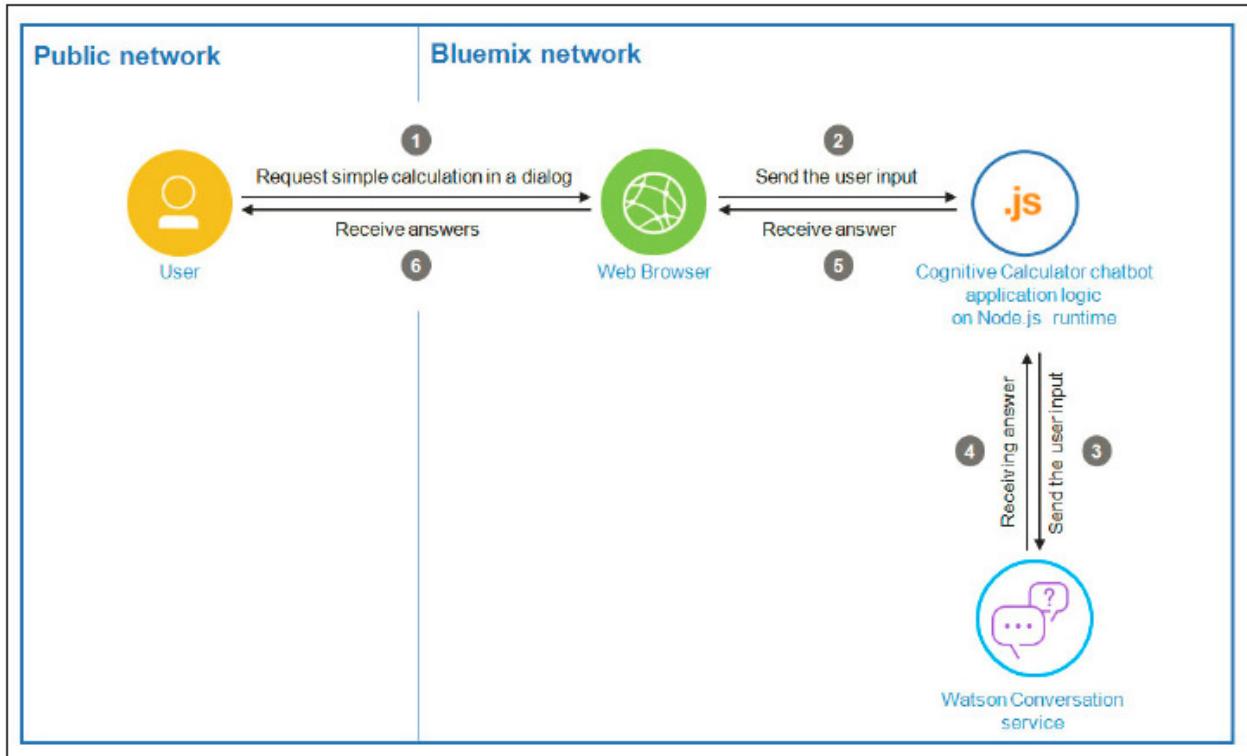


Figure 1-2 Architecture

The figure describes these steps:

In a web browser, the user engages in a conversation with the Cognitive Calculator chatbot application, requesting a simple calculation operation, such as I'd like to calculate the addition of 3 and 5.

The request is passed from the web browser to the chatbot application that runs on Node.js.

The application passes the request to the Conversation service.

The Conversation service understands the intent and entities passed by the application. For the user request I'd like to calculate the addition of 3 and 5, the *intent* is addition and the *entities* are 3 and 5. Then, it returns a response to the application based on the dialog built in the workspace associated with Conversation service instance. It returns the calculated result back to the user. Actual calculation is happens at orchestration layer. AI layer will pass context parameter “operation” : “add” ad OL layer can particular action on that event.

The chatbot Node.js application adds the two entities returned from the Conversation service, calculate the result and set the “result” context parameter and send back to conversation service. Conversation service will send the final message “The result is : 8” to the node.js application.

1.3 Two ways to deploy the application: Step-by-step and quick deploy

Two Git repositories are provided for this use case:

- 1 Step-by-step deployment (incomplete) version of the application

This repository contains an incomplete version of the application and is used in all sections of 1.4, “Step-by-step implementation”. This version takes you through the key steps to integrate the IBM Watson APIs with the application logic.

- 2 Quick deployment (complete) version of the application

This repository contains the final version of the application. If you want to bypass the implementation steps and instead run the application as a demonstration, download this full version. Downloading and running this full version demonstration is explained in 1.5, “Quick deployment of application”.

1.4 Step-by-step implementation

Implementing this use case involves the following steps:

- 1 [Downloading the project from the Git repository.](#)
- 2 [Configuring the Conversation workspace for the Cognitive Calculator chatbot.](#)
- 3 [Developing the Cognitive Calculator chatbot application.](#)
- 4 [Testing the application.](#)

1.4.1 Downloading the project from the Git repository

The version of the repository that you use in these steps includes the *incomplete* version of the application code. You will follow the steps to complete the code.

Download the code from GitHub:

<https://github.com/dakspate/calculator-bot>

1.4.2 Configuring the Conversation workspace for the Cognitive Calculator chatbot

This section guides you through creating the Calculator Conversation workspace for the Cognitive Calculator chatbot, and developing the relevant intents, entities, and dialog that are specific to the application. It also shows you how to test the conversation flow.

Complete these steps:

Log in to Bluemix.

On the Bluemix Dashboard, click the **Conversation** service instance that you created in 1.1.1, “[Creating a Watson Conversation service instance](#)”, which is listed under Services ([Figure 1-3](#)).

The screenshot shows the IBM Bluemix Apps interface. At the top, there is a navigation bar with a menu icon, the IBM Bluemix Apps logo, and links for Catalog, Support, and Account. Below the navigation bar, the heading "All Services (6)" is displayed, along with a "Create Service" button. A sub-section titled "Services 32/40 Used" is shown, containing a table with three columns: NAME, SERVICE OFFERING, and ACTIONS. The table has one row with the entry "Conversation" under both columns, and a vertical ellipsis icon in the ACTIONS column.

NAME	SERVICE OFFERING	ACTIONS
Conversation	Conversation	:

Figure 1-3 Conversation service instance

3. Click **Launch tool** (Figure 1-4) to open the Conversation tool.

The screenshot shows the IBM Bluemix Watson interface. At the top, there is a navigation bar with a menu icon, the IBM Bluemix Watson logo, and links for Catalog, Support, and Account. Below the navigation bar, the heading "Watson" is displayed, followed by the service name "Conversation". A navigation bar below the heading includes tabs for Manage, Service Credentials, and Connections, with the "Manage" tab currently selected. On the left side, there is a purple circular icon with a speech bubble symbol. The main area features the word "Conversation" in large blue text, followed by the subtext "Add a natural language". To the right of this text is a green "Launch tool" button with a white arrow icon. A vertical ellipsis icon is located in the top right corner of the main content area.

Figure 1-4 Launch Conversation tool

4. On the Watson Conversation dashboard, click **Create** to create a workspace ([Figure 1-5](#)).

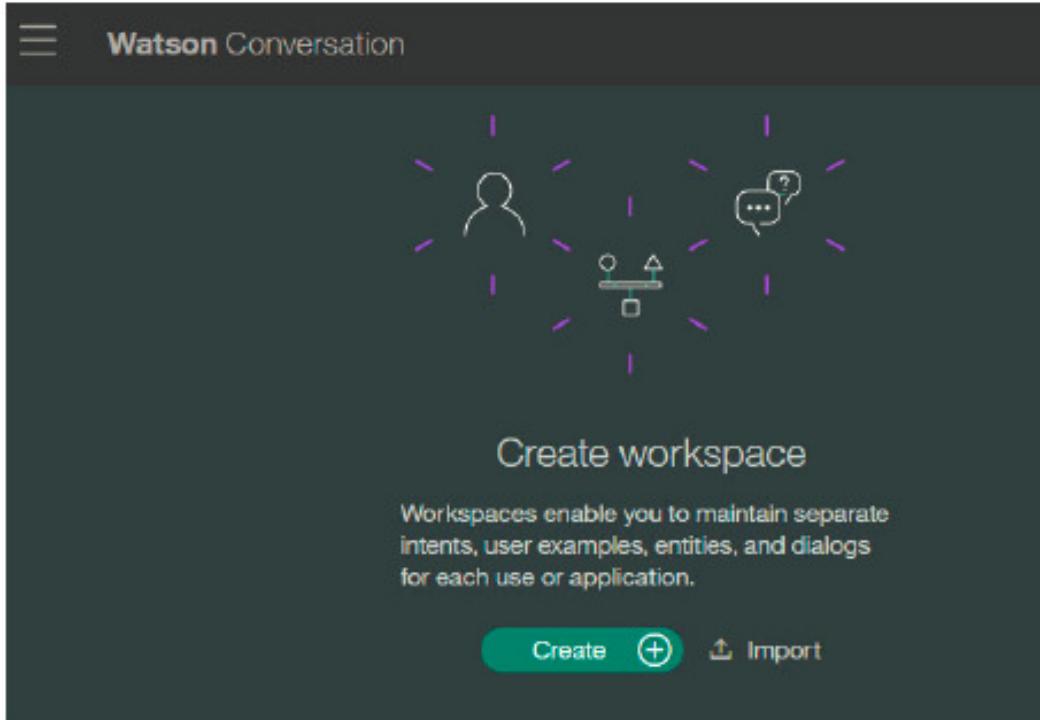


Figure 1-5 Watson Conversation Dashboard

In the Create a workspace window ([Figure 1-6](#)), enter the following information and then click **Create**:

- Name: Calculator
- Description: Calculator Conversation workspace that allows addition and multiplication operations using Natural Language.
- Language: English (U.S.)

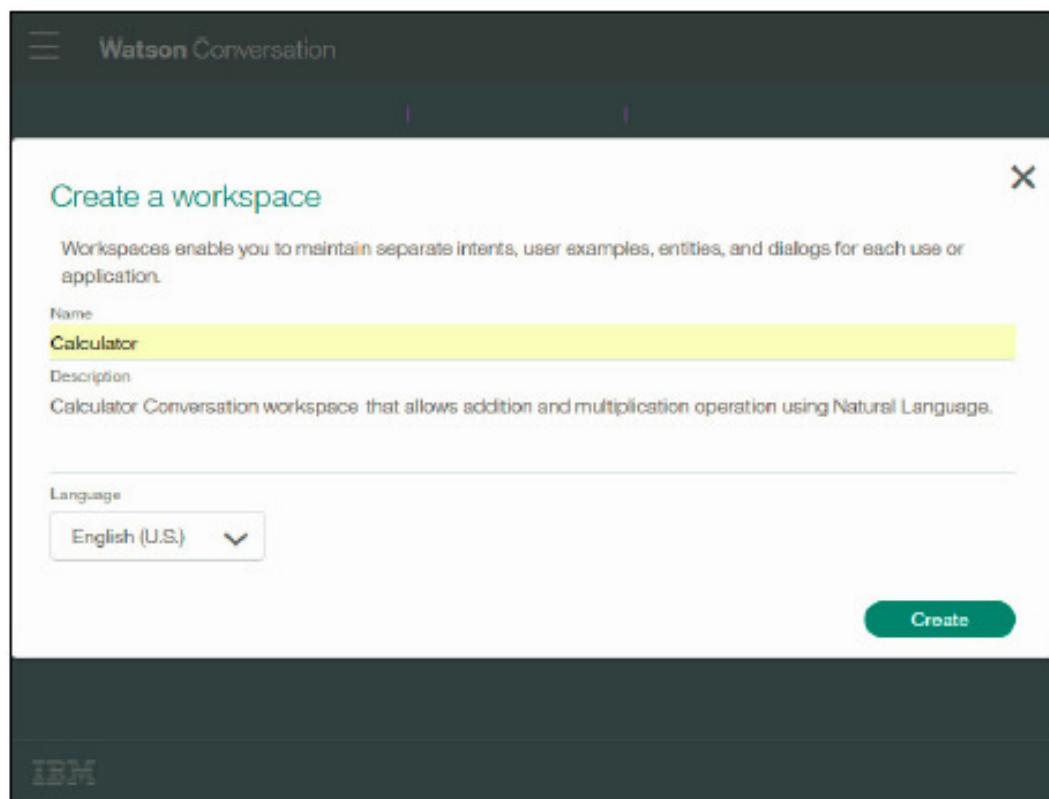


Figure 1-6 Create the Calculator workspace

Get the Workspace ID so that you can configure your application to point to this workspace in step 1:

- Click the **three horizontal bars** at the top-left corner ([Figure 1-7](#)).

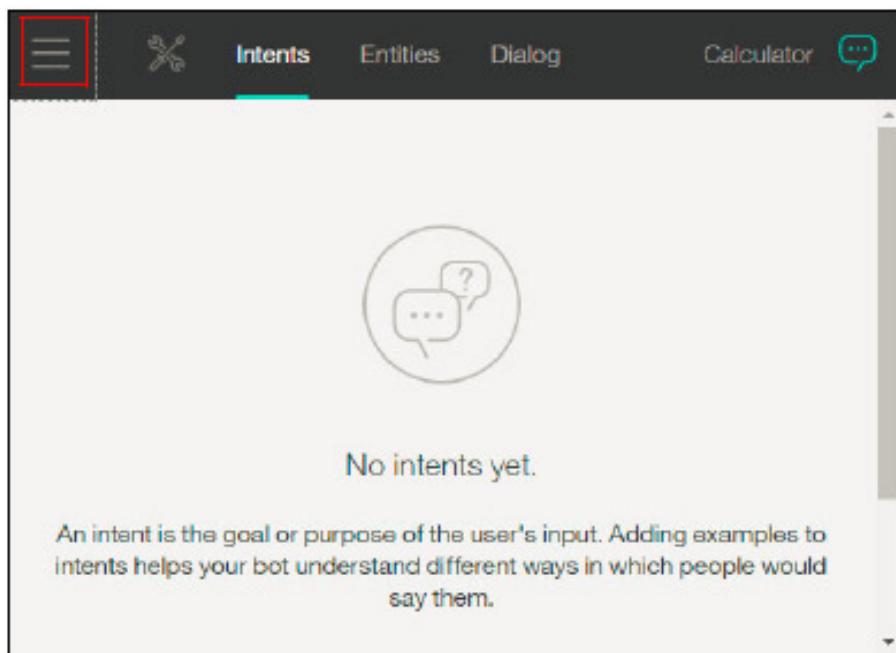


Figure 1-7 Calculator workspace

b. Click **Back to workspaces** (Figure 1-8).

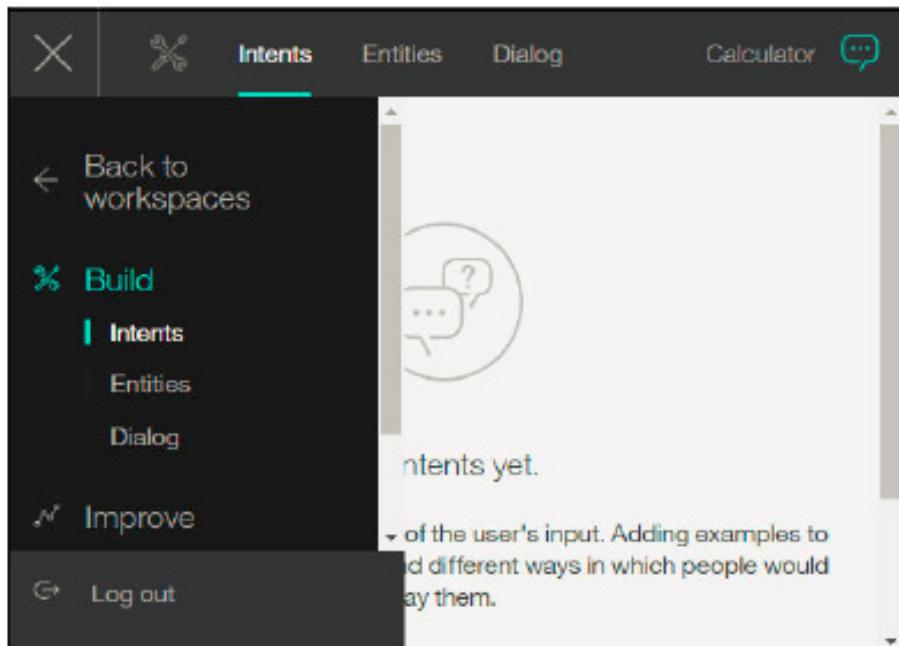


Figure 1-8 Calculator Conversation workspace

Click the **three vertical dots** at the top right of the Calculator box and then select **View details** (Figure 1-9).

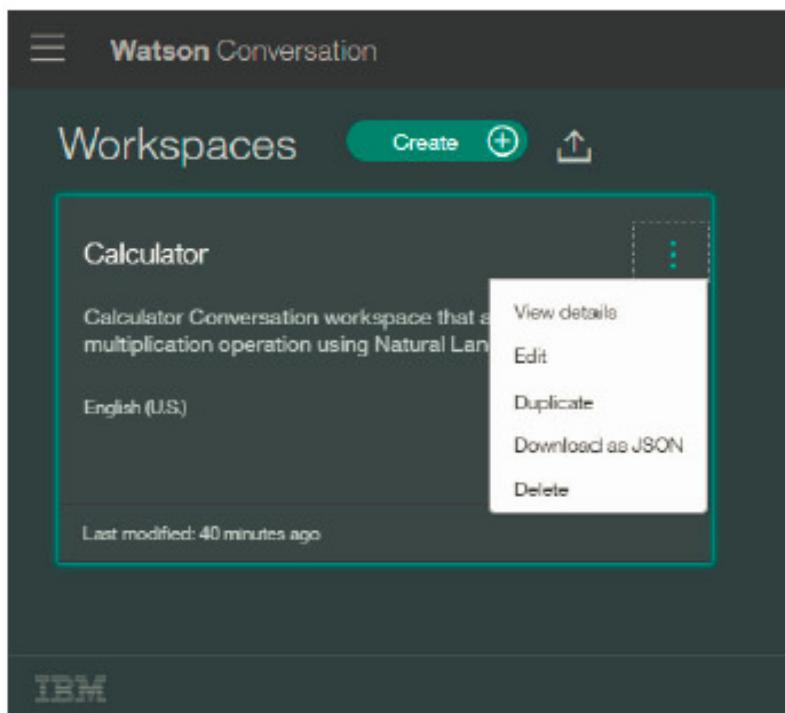


Figure 1-9 Calculator workspace menu

Copy the Workspace ID value and save it in a local text file ([Figure 1-10](#)). You will use the value of the Workspace ID in step [1](#).

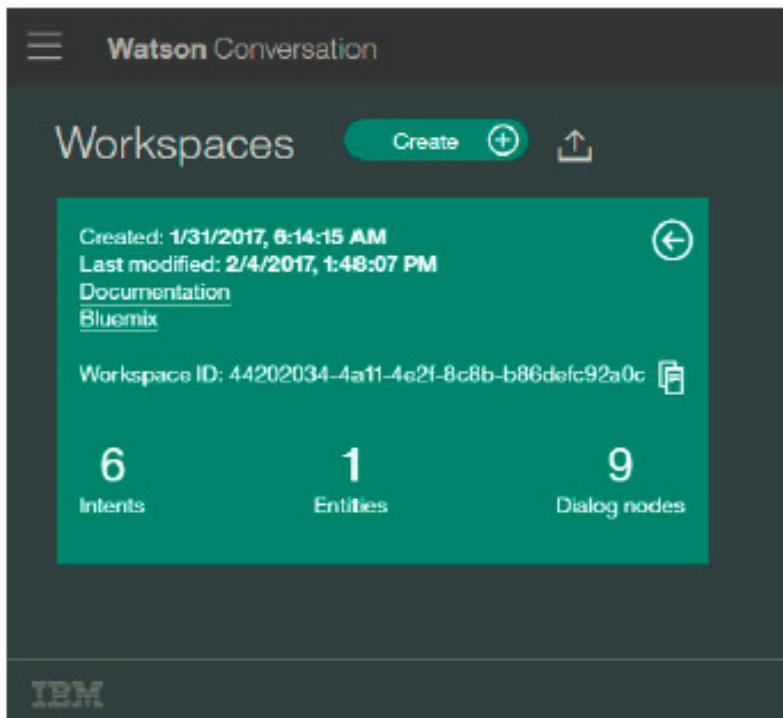


Figure 1-10 Workspace ID

Add intents

For the Conversation service to be able to understand the goal or purpose of the user's input in natural language, you must train the workspace with some examples for each intent. You will create an intent for the *calculation* functionality and another intents like *greetings*, *what is your name*, *how you can help me*, *thanks* and *bye* functionality. Although you are required to train the workspace by providing a minimum of five examples of user input for each intent, to improve the accuracy, you should provide more than five examples.

The steps in this section describe how you create the intents that are listed in [Table 1-1](#).

Table 1-1 Intents to be created for the Calculator chatbot use case

Intent	Description
#calculation	Identifies that the user wants to perform calculation of numbers User examples: ! Calculate numbers ! kindly help me by calculating ! compute numbers ! evaluate result ! i want to compute numbers
#greetings	Identifies that the user is greeting to chat bot application. User examples: ! Hi ! GM ! Hello ! Good Morning ! Hye

Intent	Description
#what_is_your_name	<p>Identifies that the user wants to know chat bot name. User examples: ! What is your name ! Do you have a preferred name ! Do you have a nickname ! What do you call yourself ! What should I call you ! What name do you go by ! What do people call you ! Do you have a name ! What are you called ! by which name people call you</p>
#how_you_can_help_me	<p>Identifies that the user wants to know capabilities of chat bot. User examples ! list all your capabilities ! Tell me your capabilities ! which operations you can perform ! how you can help me</p>
#how_is_weather	<p>Identifies that the user wants to know weather details User examples: ! How is the weather ! How is the weather today ! How is the weather where you are ! What is the weather today ! What is the weather like</p>
#thank_you	<p>Identifies that the user is happy with the answers and thanking to the chat bot. User examples: ! thanks ! You made my day ! great ! Thank you ! Superb</p>
#bye	<p>Identifies that the user wants to terminate/exit from the chat bot User Examples: ! Bye ! See you ! cu ! Bye bye ! See you soon</p>

[Figure 1-11](#) shows a conversation between the user and the Calculator chatbot application, and shows how the Conversation service maps the user input in natural language to the corresponding intent configured in the Conversation workspace.

Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?

calculate

let's start with the addition

Kindly help me with the numbers that you want to add.

i'd like to add three

Oops. I found only 1 operand(number) and required at least 2 operand(number) to perform addition operation.
Kindly provide other number/s.

and 4

The result is : 7

thank_you

thanks

You are welcome.

multiply 5, 6 and 7

The result is : 210

5 + 5 + 15

The result is : 25

thank_you

thanks

Glad !! I could help you today.

what_is_your_name

what is your name?

I am Cognitive Calculator, I could help you with addition or multiplication of numbers.

how_is_weather

how is weather

Well sometimes i do not have answer to your query as i am not trained on it.

divide 5 and 6

Sorry!! At this time i dont know how to perform divide operation. Currently i can help you with only addition and multiplication of numbers.

bye

bye

Bye. See you again.

Type something

Figure 1-11 Cognitive Calculator chatbot showing intents extracted from the user input

To add the intents that are listed in [Table 1-1](#) to the Calculator workspace, follow these steps:

1. Open the Calculator workspace ([Figure 1-12](#)).

The screenshot shows the 'Watson Conversation' interface with the title 'Workspaces'. A green 'Create' button with a plus sign is visible. Below it is a card for the 'Calculator' workspace, which includes a description: 'Calculator Conversation workspace that allows addition and multiplication operation using Natural Language.', the language 'English (U.S.)', a 'Get started' button, and a note 'Last modified: 23 hours ago'.

Figure 1-12 Conversation Workspaces

At the start of the conversation, the user specifies the mathematical operation to be performed, addition or multiplication, ([Figure 1-13](#)).

The screenshot shows a chatbot conversation. The user message is: 'Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?'. To the right, a JSON object representing the bot's response is shown:

```
Watson understands
1 {
2   "intents": [],
3   "entities": [],
4   "input": {},
5   "output": {
6     "text": [

```

Figure 1-13 Calculator chatbot

Create the intents that will enable the Conversation service to interpret the user input:

Create an intent for the addition operation capability:

- i. Click **Create new** to create new intent ([Figure 1-14](#)).

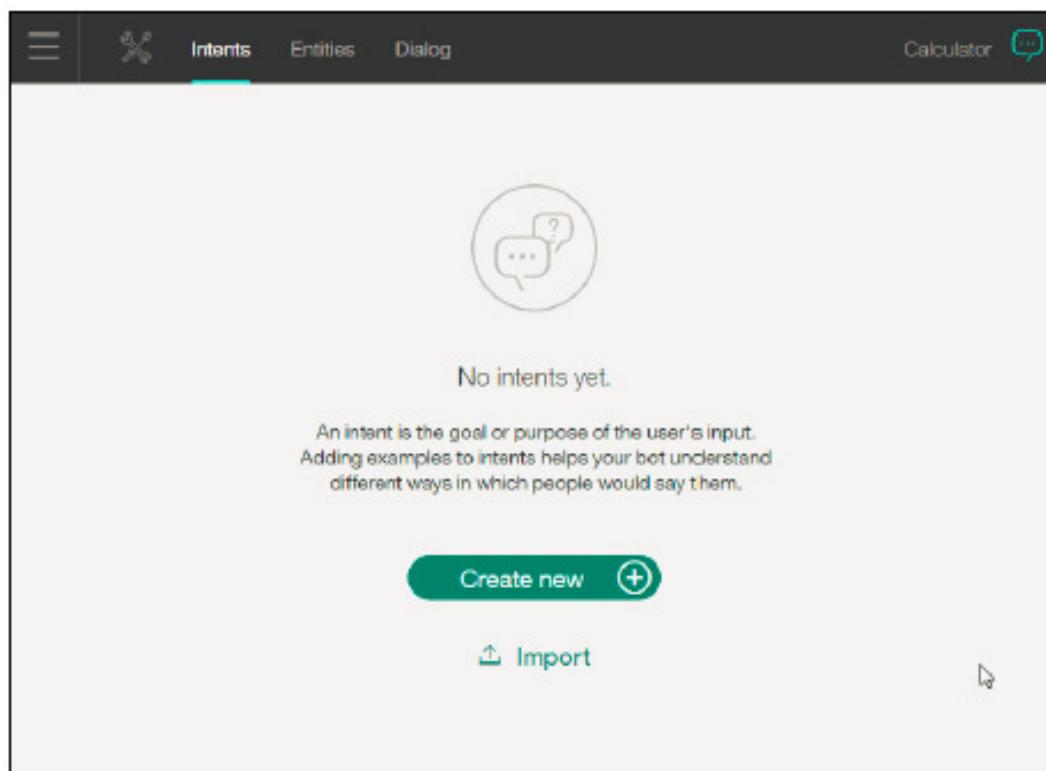


Figure 1-14 Create #calculation intent (1 of 3)

- i. Type calculation in the Intent name field ([Figure 1-15](#)).

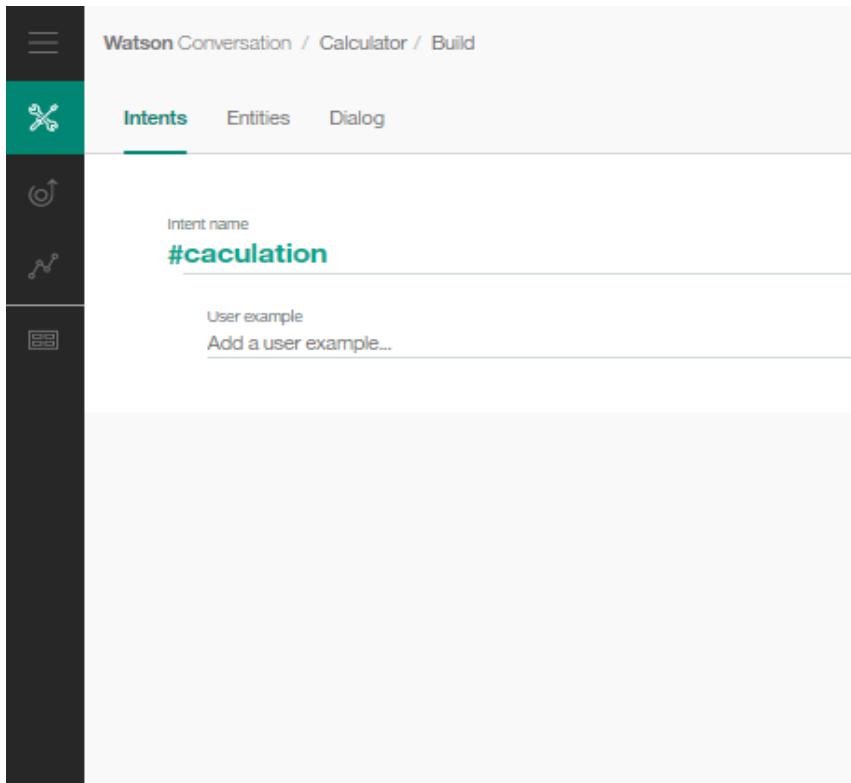


Figure 1-15 Create #calculation intent (2 of 3)

- ii. Add a minimum of five user examples for this intent ([Figure 1-16](#)), then click **Create**.

#calculation

Add a new user example...

- calculate numbers
- kindly help me by calculating
- compute numbers
- evaluate result
- assess
- i want to compute numbers
- help me in assessing

Figure 1-16 Create #calculation intent: user examples (3 of 3)

Create a `what_is_your_name` intent to know the chat bot name capability and provide user examples ([Figure 1-17](#)).

The screenshot shows a user interface for creating an intent named `#what_is_your_name`. At the top, there is a dropdown arrow icon. Below it, the intent name is displayed in a teal-colored font. To the right of the name is a horizontal line. Underneath the name is a button labeled **Add a new user example...** with a plus sign icon. Below this button is a list of ten user examples, each preceded by a small square checkbox:

- Do you have a preferred name
- Do you have a nickname
- What do you call yourself
- What should I call you
- What name do you go by
- What do people call you
- Do you have a name
- What are you called
- What is your name

Figure 3-17 Create `#what_is_your_name` intent with user examples

After the user requests the operation to be performed, the user specifies the actual addition or multiplication calculation. The service must also be able to identify whether the user provides only one operand and respond accordingly. The service must be configured for other normal chitchat questions/intents like how is weather, thanks, bye, how you can help me the way normal user talks.

The Conversation service can understand user inputs, greetings, thank_you, how_is_weather, how_you_can_help_me intents:

- Create the greetings intent with the user examples ([Figure 1-18](#)).

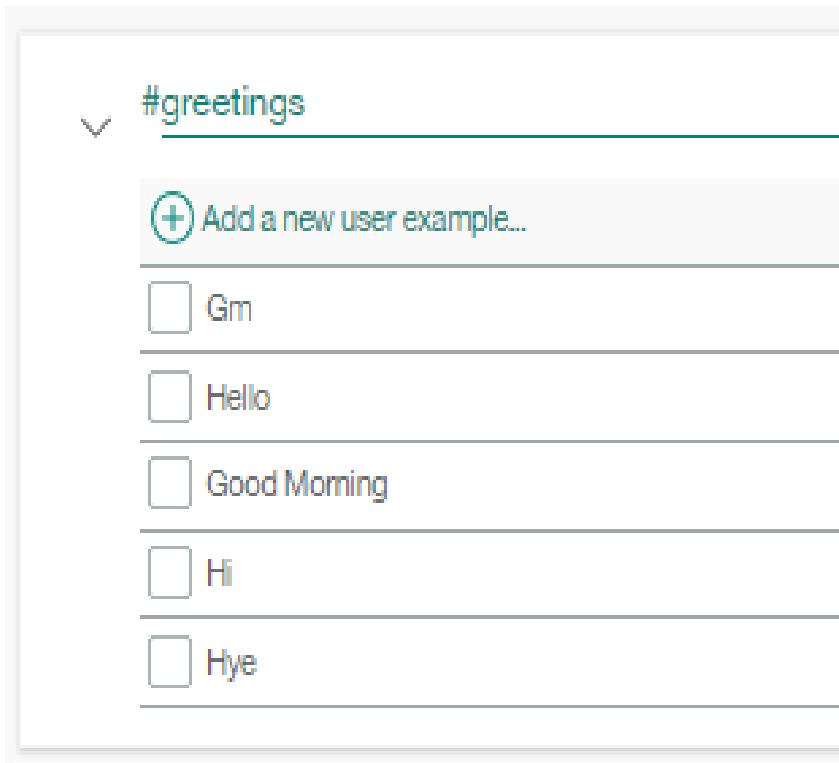


Figure 1-18 Create #greetings intent and user examples

b. Create thank_you intent with the user examples ([Figure 1-19](#)).

#thank_you

+ Add a new user example...

thanks

you made my day

you are fantastic

superb

thank you

great

gr8

Figure 1-19 Create #thank_you intent with user examples

c. Create the bye intent with the user examples ([Figure 1-20](#)).

#bye

X Add a new user example...

cu

bye

see you

bye bye

see you soon

Figure 1-20 Create #bye intent with user examples

d. Create the how_is_weather intent with the user examples ([Figure 1-21](#)).

The screenshot shows a list of user examples for the intent '#how_is_weather'. At the top is a button labeled '+ Add a new user example...'. Below it is a vertical list of ten examples, each preceded by an empty checkbox:

- What is the weather like where you are
- What is the weather like today
- What is the weather like
- What is the weather where you are
- What is the weather today
- What is the weather
- How is the weather where you are
- How is the weather today
- How is the weather

Figure 1-21 Create #how_is_weather intent with user examples

e. Create the how_you_can_help_me intent with the user examples ([Figure 1-22](#)).

The screenshot shows a list of user examples for the intent '#how_you_can_help_me'. At the top is a button labeled '+ Add a new user example...'. Below it is a vertical list of five examples, each preceded by an empty checkbox:

- list all your capabilities
- Tell me your capabilities
- which operations you can perform
- how you can help me
- what you can do for me

Figure 1-22 Create #how_is_weather intent with user examples

Now, you have all the intents needed for the Cognitive Calculator chatbot (Figure 1-23).

The screenshot shows the Microsoft Bot Framework Emulator's workspace interface. On the left is a vertical toolbar with icons for scissors, a circular arrow, a gear, and a grid. The main area has tabs for 'Intents' (which is selected and highlighted in green), 'Entities', and 'Dialog'. Below the tabs is a toolbar with buttons for 'Create new' (with a plus sign), 'Import' (with an upward arrow), 'Export' (with a downward arrow), and 'Delete' (with a trash can icon). The main content area lists seven intents:

- > #greetings
Gm
- > #thank_you
thanks
- > #bye
cu
- > #how_is_weather
What is the weather like where you are
- > #how_you_can_help_me
list all your capabilities
- > #what_is_your_name
Do you have a preferred name
- > #calculation
calculate numbers

Figure 1-23 Calculator workspace intents

Add entities

1) User-defined Entities

You want the service to identify the various arithmetic operation like add, subtract, multiply and divide along with the symbols like +, -, *, and /. operands of the addition and multiplication operations.

1. Click **Entities** on the top toolbar ([Figure 1-24](#)).

The screenshot shows the Watson Conversation interface with the 'Calculator' workspace selected. The top navigation bar includes 'Watson Conversation / Calculator / Build'. Below the navigation is a toolbar with tabs: 'Intents' (selected), 'Entities' (highlighted with a red box), and 'Dialog'. The toolbar also features 'Create new' (with a plus sign), 'Import' (with an upward arrow), 'Export' (with a downward arrow), and 'Delete' (with a trash can icon). The main content area displays a list of user-defined entities:

- > #greetings
Gm
- > #thank_you
thanks
- > #bye
cu
- > #how_is_weather
What is the weather like where you are
- > #how_you_can_help_me
list all your capabilities
- > #what_is_your_name
Do you have a preferred name

Figure 1-24 Calculator workspace: Adding entities

2. Click **My entities** (Figure 1-25).

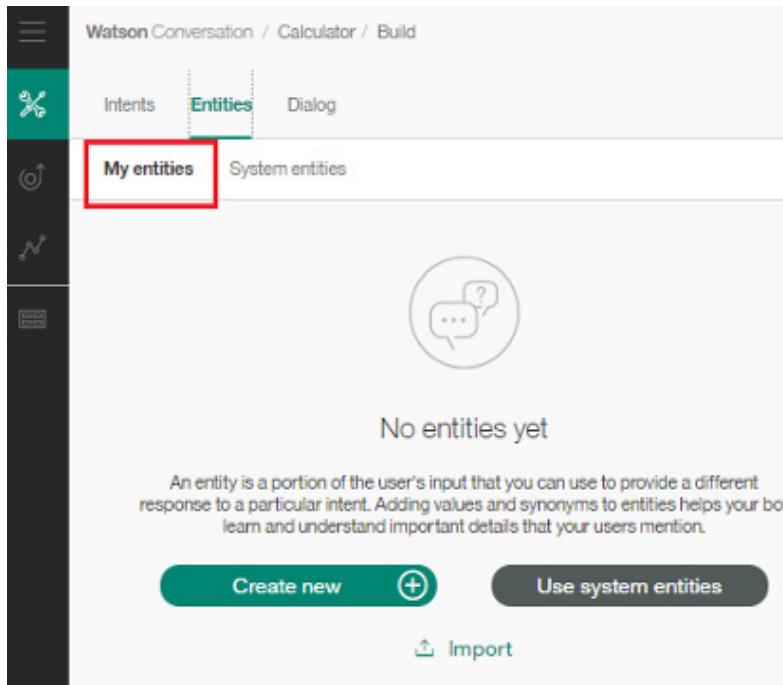


Figure 1-25 My entities

3. Click create new button and add entity named operations. It has values like add, subtract, multiply, and divide along with synonyms for them. So when ever user refer any of the synonyms in the ask, service will map it with associated entity. **My entities** along with synonyms.(Figure 1-26).

The screenshot shows the 'My entities' page. The top navigation bar has tabs for 'Intents', 'Entities' (selected), and 'Dialog'. Below the tabs, there are buttons for 'Create new' (with a plus sign), 'Import', 'Export', and 'Delete'. It shows '1 entity' and 'Sort by: Newest'. A dropdown menu is open for an entity named '@operations'. The list includes:

Value	Synonyms
add	total, aggregate, +, summation, sum of, sum, addition (7 Synonyms)
subtract	subtraction, - (3 Synonyms)
divide	division, / (2 Synonyms)
multiply	X, * (3 Synonyms)

At the top right, there's a 'Fuzzy Matching BETA' toggle switch set to 'off' and a trash can icon.

Figure 1-26 My entities

4. Enable Fuzzy Matching^(BETA) feature on entities to help service to understand relevant entities even if user misspelled it. Example given if user asks for “devide number 4 by 2”, here spelling of divide is wrong but as we turned on fuzzy matching feature it should able to map devide with divide entity value. Enable Fuzzy Matching^(BETA) shown in [\(Figure 1-27\)](#).

The screenshot shows the Entities interface with the 'Entities' tab selected. Under the 'My entities' tab, there is a list of operations: add, subtract, divide, and multiply. Each operation has multiple synonyms listed next to it. In the top right corner, there is a 'Fuzzy Matching BETA' toggle switch, which is currently set to 'on'. This toggle is highlighted with a red rectangle.

Operation	Synonyms
add	total, aggregate, +, summation, sum of, sum, addition (7 Synonyms)
subtract	subtraction, - (3 Synonyms)
divide	division, / (2 Synonyms)
multiply	X, * (3 Synonyms)

Figure 1-27 Enabling Fuzzy Matching^(BETA)

2) System Entities

You want the service to identify the operands of the addition and multiplication operations. The operands are numbers written as either digits (3, 64, 873, and so on) or text (one, two, eighty-seven, and so on). Use an available system entity that identifies the numbers:

1. Click **System entities** tab besides My entities tab.[\(Figure 1-28\)](#).

The screenshot shows the Entities interface with the 'Entities' tab selected. Under the 'System entities' tab, there is a list of operations: add, subtract, divide, and multiply. Each operation has multiple synonyms listed next to it. In the top right corner, there is a 'Fuzzy Matching BETA' toggle switch, which is currently set to 'on'. This toggle is highlighted with a red rectangle. The 'System entities' tab is also highlighted with a red rectangle.

Operation	Synonyms
add	total, aggregate, +, summation, sum of, sum, addition (7 Synonyms)
subtract	subtraction, - (3 Synonyms)
divide	division, / (2 Synonyms)
multiply	X, * (3 Synonyms)

Figure 1-28 System Entities

2) Switch the **off** toggle to the **on** position beside @sys-number to enable this system entity ([Figure 1-29](#)).

The screenshot shows the 'Entities' tab selected in the top navigation bar. Under the 'System entities' section, the @sys-number entity is highlighted with a red border. Its description states: 'Extracts numbers mentioned from user examples as digits or written as numbers. (21)' and its toggle switch is set to 'on'.

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

Figure 1-29 System entities: Enable @sys-number

Create the dialog

Follow these steps:

1. Click **Dialog** in the top toolbar and click **Create** to create the dialog ([Figure 1-30](#)).

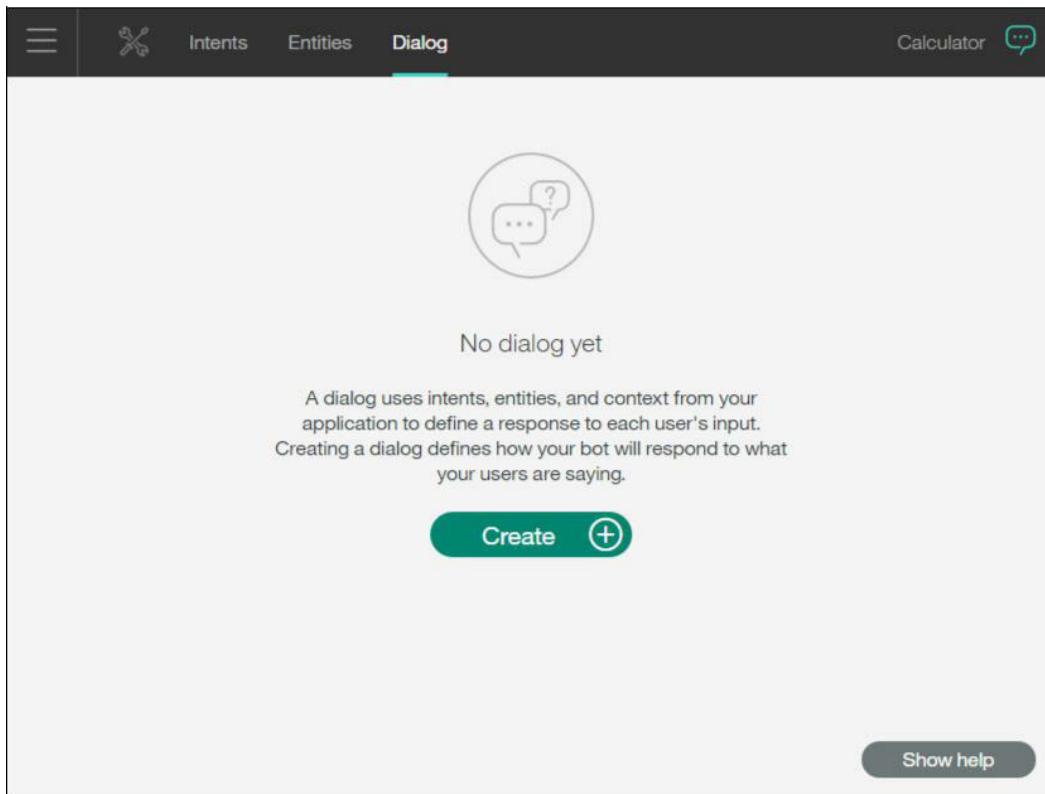


Figure 1-30 Dialog

When you creating the dialog for the first time it will create two default node named “welcome” and “anything_else” node. Welcome node can be used to display welcome message to user when he/she start conversation with chat bot. Anything_else node can be used to display specific message when none of the node condition matched. Two default node is created ([Figure 1-31](#)).

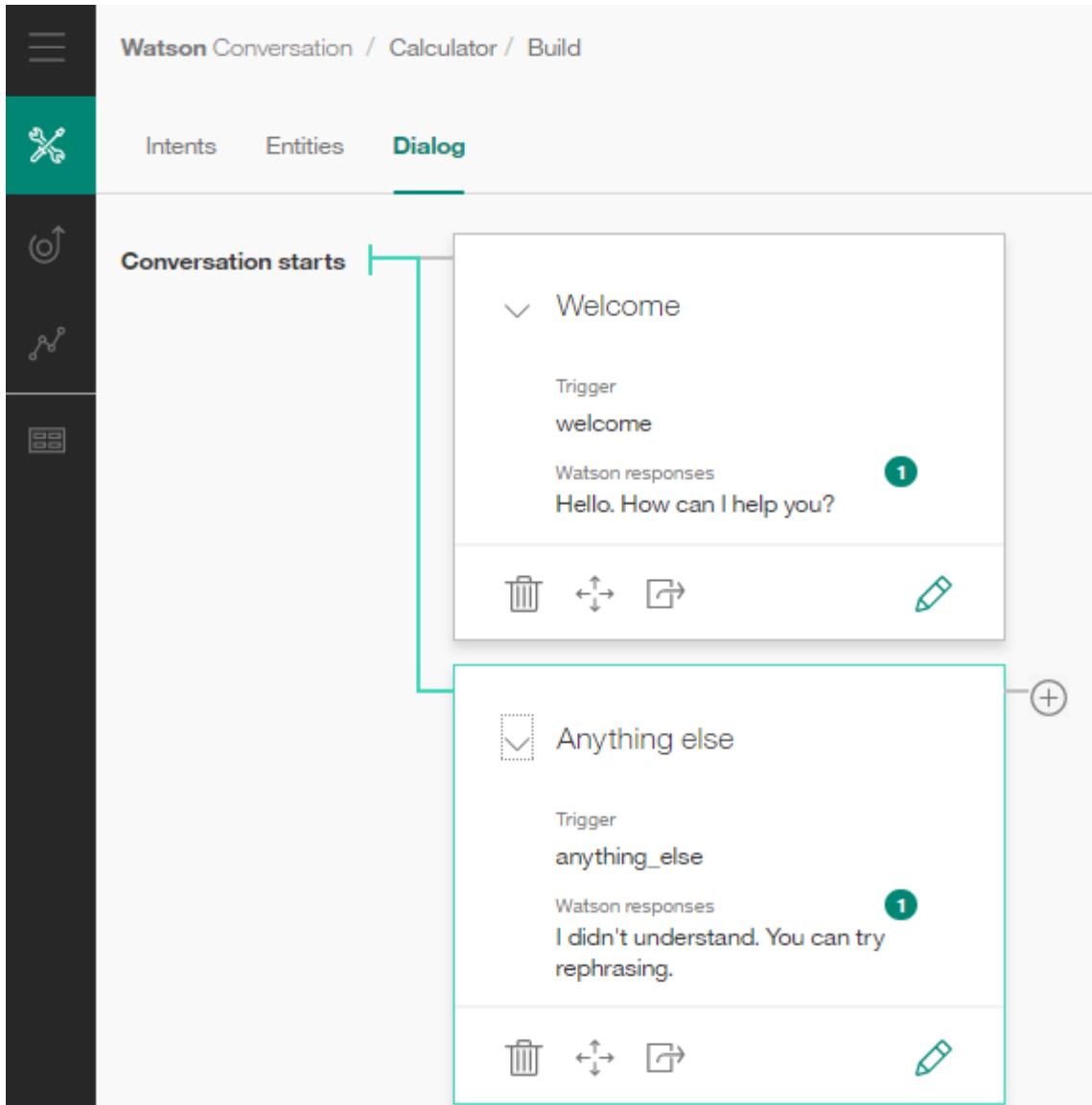


Figure 1-31 Dialog default base node

2. Write the response that you want the chatbot to provide and then press **Enter** (Figure 1-32). In this case, you might want the chatbot to respond with this greeting:

Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?

Watson Conversation / Calculator / Build

Intents Entities Dialog

Conversation starts

Welcome

Trigger welcome

Watson responds

Hi, I am VCal - a calculator you can chat with. I ...

Anything else

Trigger anything_else

Watson responds

I didn't understand. You can try rephrasing.

Welcome

If bot recognizes:

welcome

Then respond with:

Add response condition

1. Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as additi

Add a variation to this response

Add another response

Figure 1-32 response text in welcome node

3. Click the plus sign (+) to create a new node(Figure 1-33)

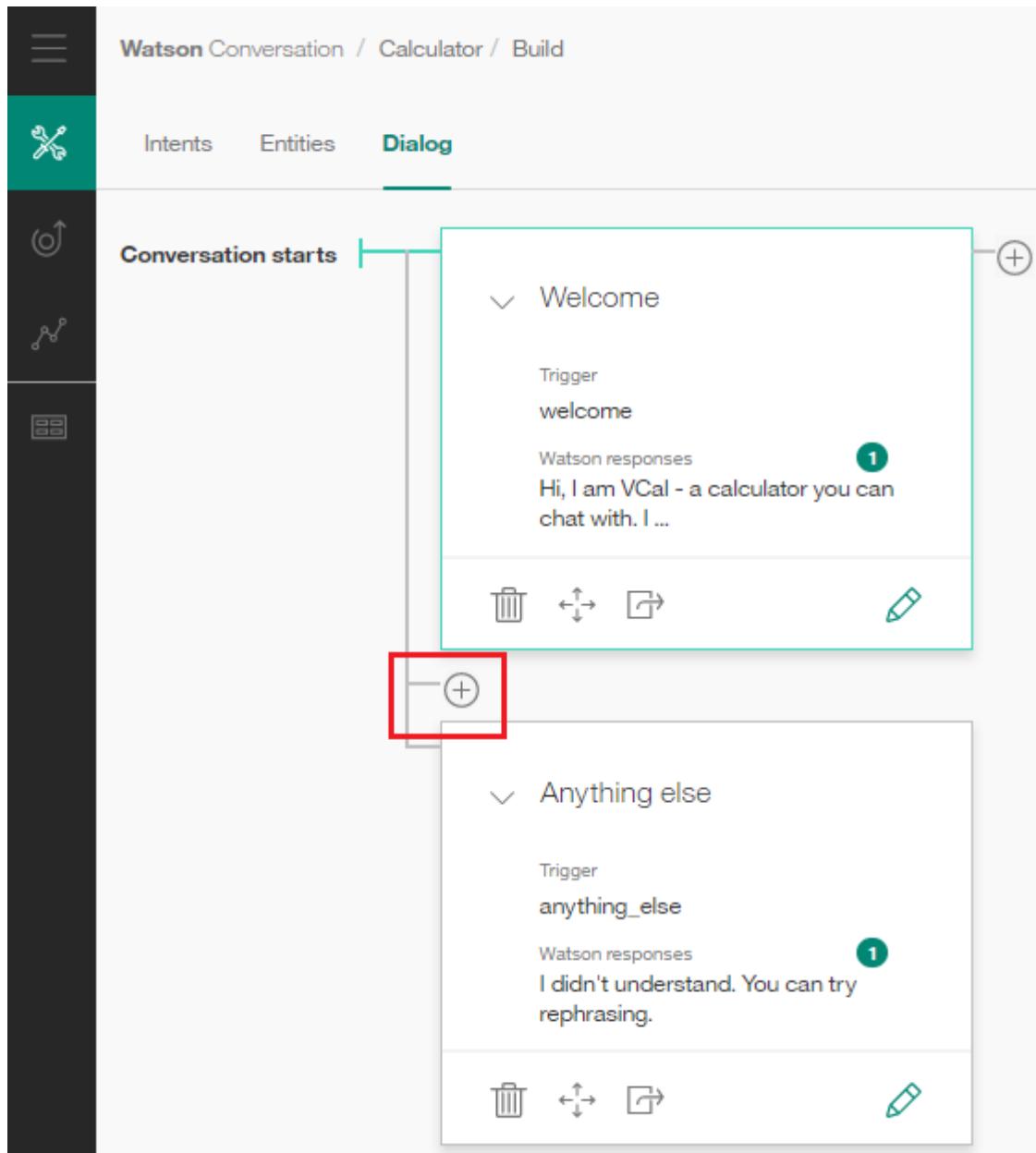


Figure 1-33 Create a new node

4 This node will be triggered when user input is recognized as the #greetings intent.(Figure 1-34)
Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #greetings and then select it from the autocomplete text box
- 2 Click the (+) sign besides #greetings to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition type “Hi !! How may I help you today?”

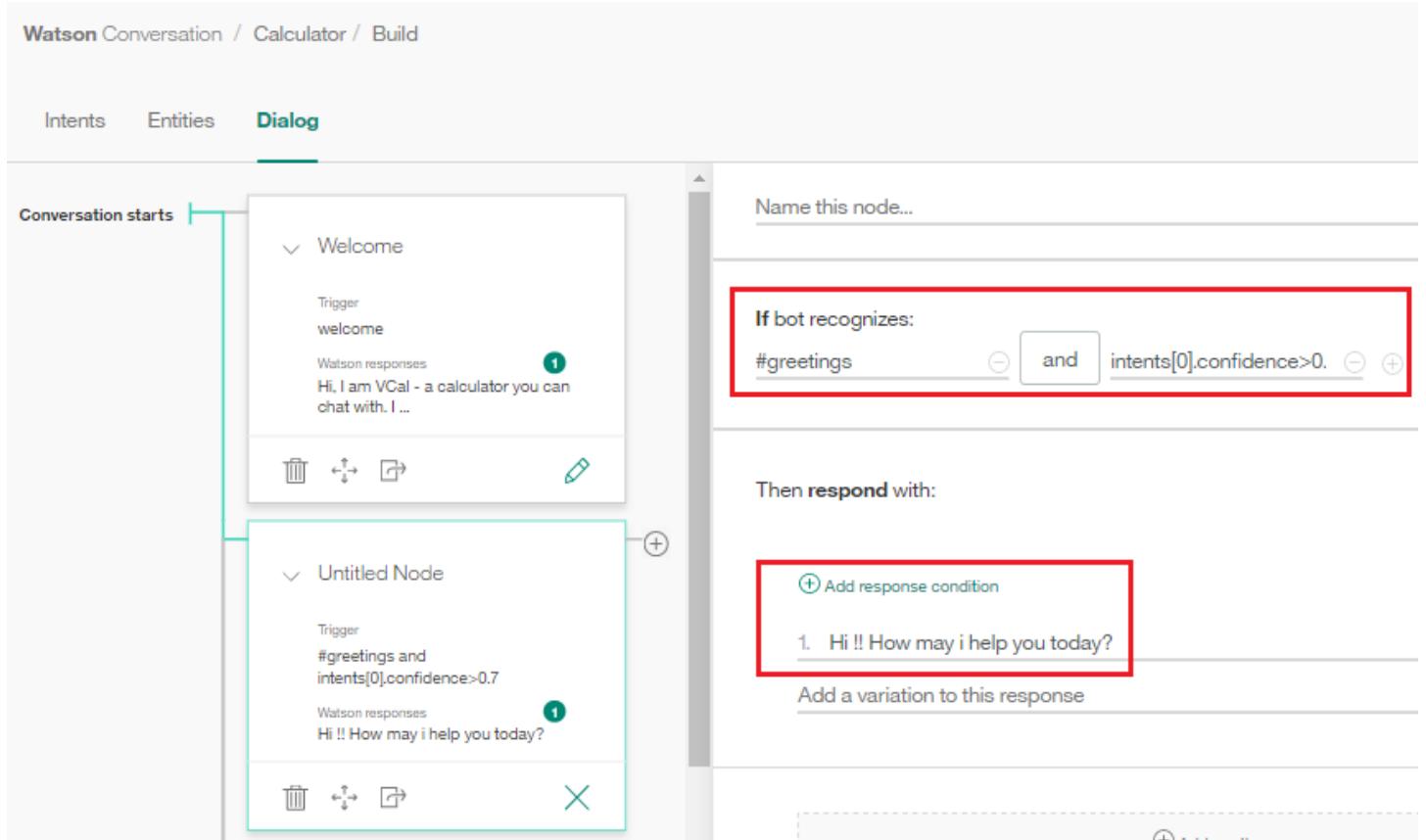


Figure 1-34 #greetings node creation

5 Click (+) symbol below #greeting node to add new node. This node will be triggered when user input is recognized as the #what_is_your_name intent.(Figure 1-35)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #what_is_your_name and then select it from the autocomplete text box
- 2 Click the (+) sign besides #what_is_your_name to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition type “I am Cognitive Calculator, I could help you with addition or multiplication of numbers.”

Watson Conversation / Calculator / Build

Intents Entities Dialog

Conversation starts

> Welcome

#greetings and intents[0].confidence>0.7

Untitled Node

Trigger
#what_is_your_name and intents[0].confidence>0.7

Watson responses
I am Cognitive Calculator, I could help you with a...

Name this node...

If bot recognizes:
#what_is_your_name and intents[0].confidence>0.7

Then respond with:

Add response condition

I am Cognitive Calculator, I could help you with addition or multiplication of numbers.

Add a variation to this response

The screenshot shows the Watson Conversation interface in the 'Build' tab. On the left, a tree view shows a 'Conversation starts' node with a 'Welcome' child node, and an 'Untitled Node' node which is expanded to show its trigger ('#what_is_your_name and intents[0].confidence>0.7') and Watson response ('I am Cognitive Calculator, I could help you with a...'). On the right, a configuration panel is open for the 'Untitled Node'. It has a red box around the 'If bot recognizes:' section, which contains the trigger '#what_is_your_name and intents[0].confidence>0.7'. Another red box surrounds the 'Then respond with:' section, which contains the response 'I am Cognitive Calculator, I could help you with addition or multiplication of numbers.' Below this, there is a link to 'Add a variation to this response'.

Figure 1-35 #what_is_your_name node creation

5 Click (+) symbol below #what_is_your_name node to add new node. This node will be triggered when user input is recognized as the #how_is_weather intent.(Figure 1-36)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #how_is_weather and then select it from the autocomplete text box
- 2 Click the (+) sign besides #how_is_weather to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition type “Well sometimes i do not have answer to your query as i am not trained on it.”

Watson Conversation / Calculator / Build

Intents Entities Dialog

The screenshot shows the Watson Conversation interface in the 'Build' tab. On the left, a vertical tree view displays the conversation flow: 'Conversation starts' leads to 'Welcome', then '#greetings and intents[0].confidence>0.7', then '#what_is_your_name and intents[0].confidence>0.7'. A new node, 'Untitled Node', is currently selected and expanded. Its trigger is set to '#how_is_weather and intents[0].confidence>0.7'. Its Watson response is 'Well sometimes i do not have answer to your query ...'. On the right, the configuration panel shows the 'If bot recognizes:' section with the trigger '#how_is_weather and intents[0].confidence>0.7' highlighted with a red box. Below it, the 'Then respond with:' section shows the response 'Well sometimes i do not have answer to your query as i am not trained on it.' also highlighted with a red box. The entire configuration area is enclosed in a large red box.

Name this node...

If bot recognizes:
#how_is_weather and intents[0].confidence>0.7

Then respond with:

Add response condition
1. Well sometimes i do not have answer to your query as i am not trained on it.

Add a variation to this response

Add another response

And then Wait for user input

Figure 1-36 #how_is_weather node creation

6 Click (+) symbol below #how_is_weather node to add new node. This node will be triggered when user input is recognized as the #how_you_can_help_me intent.(Figure 1-37)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #how_you_can_help_me and then select it from the autocomplete text box
- 2 Click the (+) sign besides #how_you_can_help_me to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition type “I can help you in calculating numbers. I can perform addition and multiplication of numbers for you.”

Watson Conversation / Calculator / Build

Intents Entities Dialog

The screenshot shows the Watson Conversation interface in the 'Build' tab. On the left, there's a tree view of nodes:

- #what_is_your_name and intents[0].confidence>0.7
- #how_is_weather and intents[0].confidence>0.7
- Untitled Node (selected, highlighted in green)
 - Trigger: #how_you_can_help_me and intents[0].confidence>0.7
 - Watson responses:
I can help you in calculating numbers.
I can perfo...
 - Buttons: trash, move, copy, X

On the right, the configuration pane for the selected node:

- Name this node... (empty)
- If bot recognizes:
#how_you_can_help_m (minus) and intents[0].confidence>0. (minus) (+) (highlighted with a red box)
- Then respond with:
 - Add response condition (highlighted with a red box)
 - 1. I can help you in calculating numbers. I can perform addition and multiplication of nu... (highlighted with a red box)
 - Add a variation to this response
 - Add another response (button)

Figure 1-37 #how_you_can_help_me node creation

7 Click (+) symbol below #how_you_can_help_me node to add new node. This node will be triggered when user input is recognized as the #thank_you intent.(Figure 1-38)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #thank_you and then select it from the autocomplete text box
- 2 Click the (+) sign besides #thank_you to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition add multiple responses and set response variation.
 - 5.1 “Glad !! I could help you today.”
 - 5.2 “You are welcome.”
 - 5.3 “My pleasure.”
 - 5.4 Set the response variation to random.

Watson Conversation / Calculator / Build

Intents Entities Dialog

The screenshot shows the Watson Conversation interface in 'Build' mode. On the left, there's a tree view of nodes:

- #what_is_your_name and intents[0].confidence>0.7
- #how_is_weather and intents[0].confidence>0.7
- #how_you_can_help_me and intents[0].confidence>0.7
- Untitled Node (selected):
 - Trigger: #thank_you and intents[0].confidence>0.7
 - Watson responses:
Glad !! I could help you today.

On the right, the 'Dialog' tab is active, showing the configuration for the Untitled Node:

- Name this node...**: (empty)
- If bot recognizes:** #thank_you and intents[0].confidence>0.7
- Then respond with:**
 - Add response condition: Glad !! I could help you today.
 - You are welcome.
 - My pleasure.
- Add a variation to this response**: Response variations are set to **random**. Set to sequential

Figure 1-38 #thank_you node creation

8 Click (+) symbol below #thank_you node to add new node. This node will be triggered when user input is recognized as the #bye intent.(Figure 1-39)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #bye and then select it from the autocomplete text box
- 2 Click the (+) sign besides #bye to add one more condition
- 3 Make sure condition type should be "and"
- 4 Add the “intents[0].confidence>0.7” as the next condition
- 5 Under Add the response condition add multiple responses and set response variation.
 - 5.1 “Bye. See you again.”
 - 5.2 “Have a nice day.”
 - 5.3 Set the response variation to sequential(default).

Watson Conversation / Calculator / Build

Intents Entities Dialog

The screenshot shows the Watson Conversation interface in the 'Build' tab. On the left, there's a tree view of nodes:

- #how_is_weather and intents[0].confidence>0.7
- #how_you_can_help_me and intents[0].confidence>0.7
- #thank_you and intents[0].confidence>0.7
- Untitled Node (selected):
 - Trigger: #bye and intents[0].confidence>0.7
 - Watson responses:
 - Bye. See you again.

On the right, the configuration pane for the selected 'Untitled Node' is shown:

- Name this node...**: (empty)
- If bot recognizes:** (highlighted with a red box)
 - #bye
 - and
 - intents[0].confidence>0.7
- Then respond with:**
 - Add response condition**: (highlighted with a red box)
 - Bye. See you again.
 - Have a nice day.
 - Add a variation to this response**: (highlighted with a red box)

Response variations are **sequential**. Set to random ⓘ
- Add another response**: (button at the bottom right)

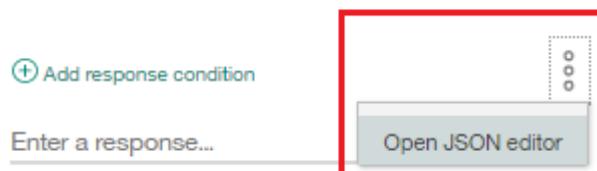
Figure 1-39 #bye node creation

8 Click (+) symbol below #bye node to add new node. This node will be triggered when user input is recognized as the #calculation intent.(Figure 1-40)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing #calculation and then select it from the autocomplete text box
- 2 Click the (+) sign besides #calculation to add one more condition
- 3 Make sure condition type should be “or”
- 4 Add the “@operations” as the next condition
- 5 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:



- 6 Add below json string as shown in image 1-40.
{ "context": { "calculationflag": true }, "output": {} }

A screenshot of the Watson Conversation interface under the 'Dialog' tab. On the left, there is a list of nodes: 'Untitled Node' (selected), 'Hello World', 'Goodbye', and 'Calculator'. The 'Untitled Node' card shows its configuration: Trigger is '#calculation or @operations', Watson responses show 'No response yet', and the JSON response is: { "output": {}, "context": { "calculationflag": true } }. On the right, the 'If bot recognizes:' section is configured with '#calculation' and '@operations' separated by an 'or' operator. The 'Then respond with:' section contains the JSON string: { "context": { "calculationflag": true }, "output": {} }. Red boxes highlight the 'If bot recognizes:' section and the 'Then respond with:' section.

Figure 1-40 #calculation node creation

9 Click (+) symbol right to the #calculation node to add new node. This node will be triggered if to check if user provided text has “@operations:add” entity.(Figure 1-41)

Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing “@operations:add” and then select it from the autocomplete text box
- 2 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:



- 3 Add below json string as shown in image 1-41.

```
{ "context": { "loop": -1, "number_arr": [], "additionflag": true }, "output": {} }
```

Watson Conversation / Calculator / Build

A screenshot of the Watson Conversation interface in "Build" mode. The interface is divided into several sections: "Intents", "Entities", "Dialog" (which is selected), "Nodes", and "Responses".
In the "Nodes" section, there are two nodes:

- Untitled Node**: Triggered by "#calculation or @operations". Its Watson responses section contains a JSON editor with the following code:

```
{ "output": {}, "context": { "calculationFlag": true } }
```
- add_operation**: Triggered by "@operations:add". Its Watson responses section contains a JSON editor with the following code:

```
{ "output": {}, "context": { "loop": -1, "number_arr": [], "additionflag": true } }
```


On the right side, under "Responses", there is a section titled "add_operation" with the following configuration:

- If bot recognizes: **@operations:add** (highlighted with a red box)
- Then respond with: (highlighted with a red box)

```
1 {  
2   "context": {  
3     "loop": -1,  
4     "number_arr": [],  
5     "additionflag": true  
6   },  
7   "output": {}  
8 }
```

Figure 1-41 add_operation node creation

10 Click jump to symbol on the #calculation node to jump to next "@operations:add" node condition. (Figure 1-42)

- 1 Click  symbol on "#calculation" node
- 2 select "@operations:add" node.
- 3 Click on "Go to condition"

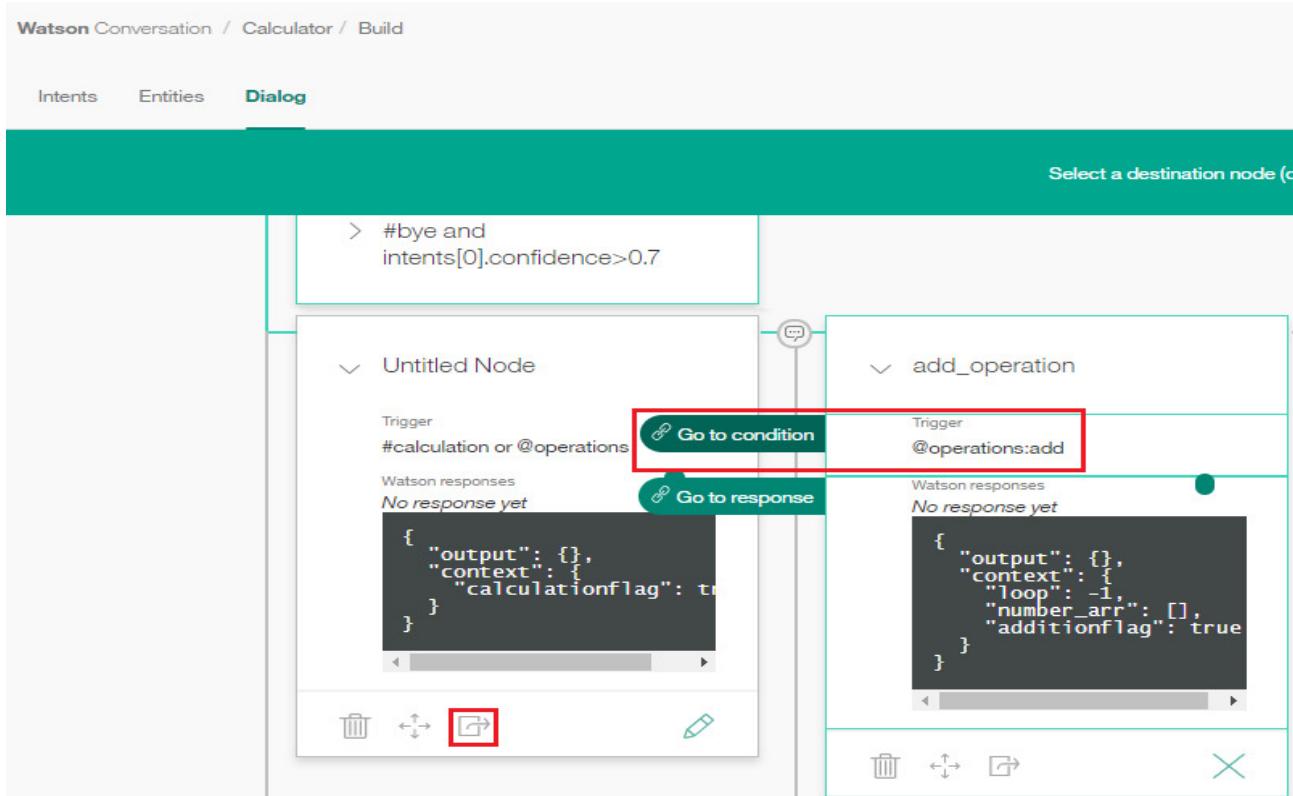
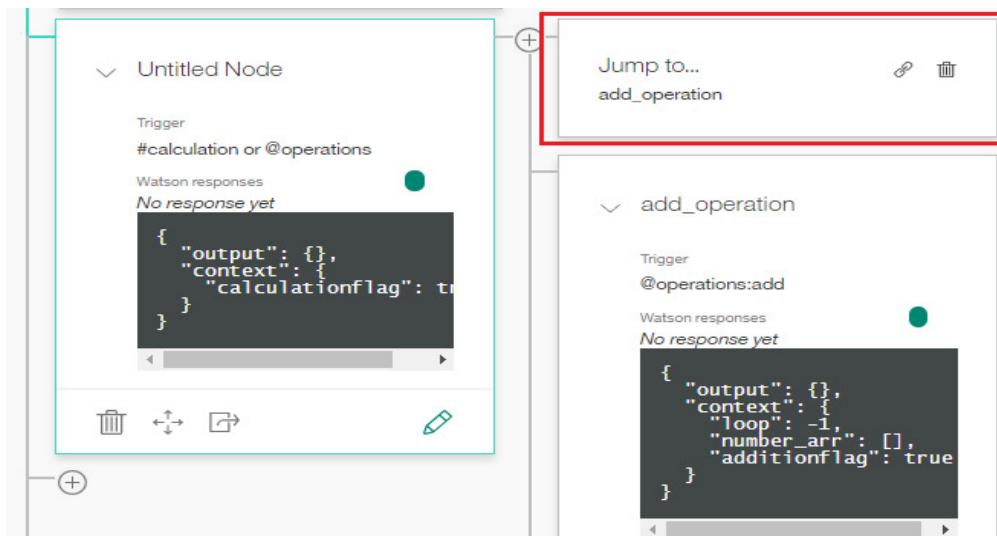


Figure 1-42 jump to from #calculation node

After adding the jump to node it should look like below figure.



11 Click (+) symbol right to the @operations:add node to add new node. This node will be executed automatically as the condition is true.(Figure 1-43)
Specify this information and then press Enter:

- 1 Under If “bot recognizes”, enter condition “true”
- 2 Leave Add response condition blank.

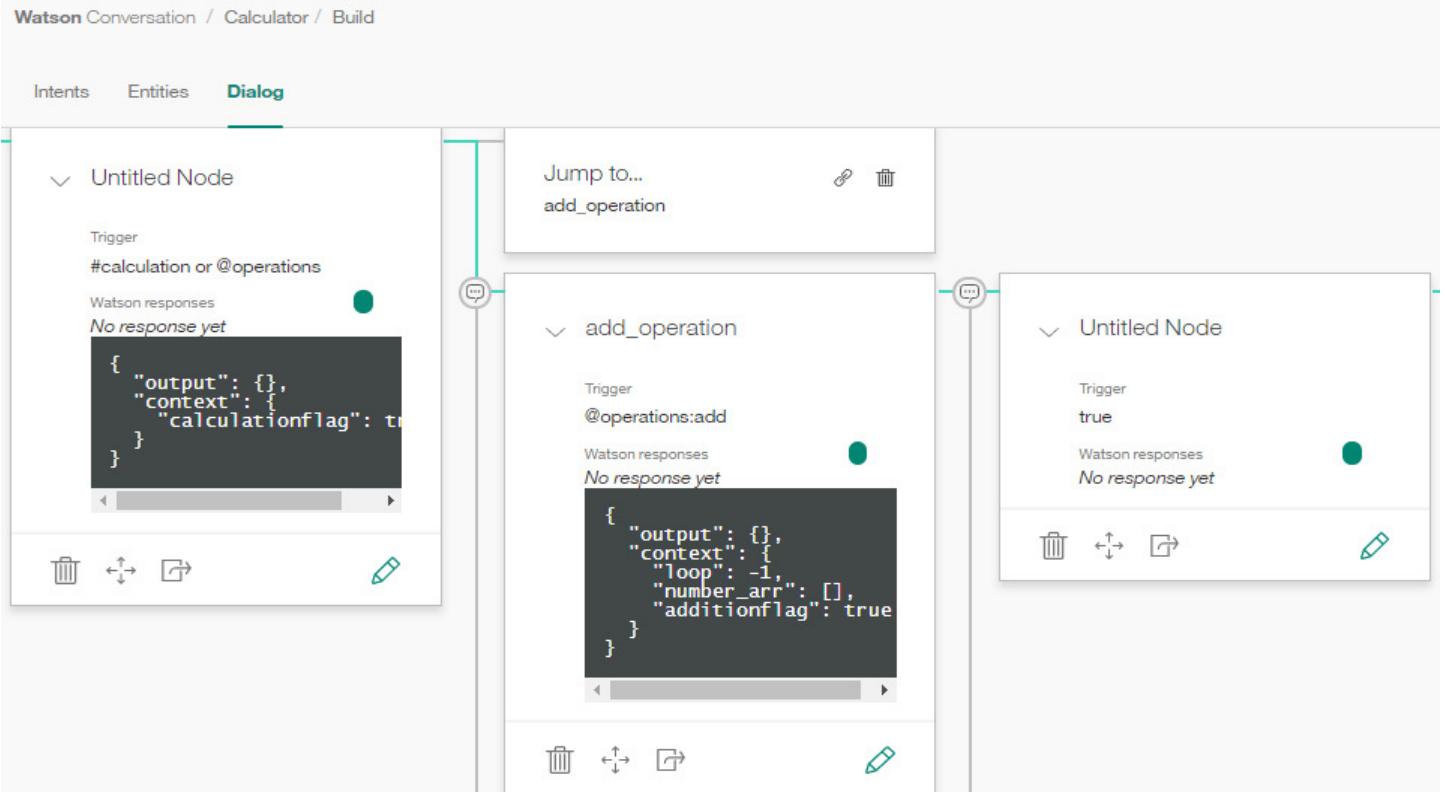


Figure 1-43 true condition node creation

12 Click jump to symbol on the @operations:add node to jump to next “true” node condition. (Figure 1-44)

- 1 Click  symbol and select “true” node.
- 2 Click on “Go to condition”

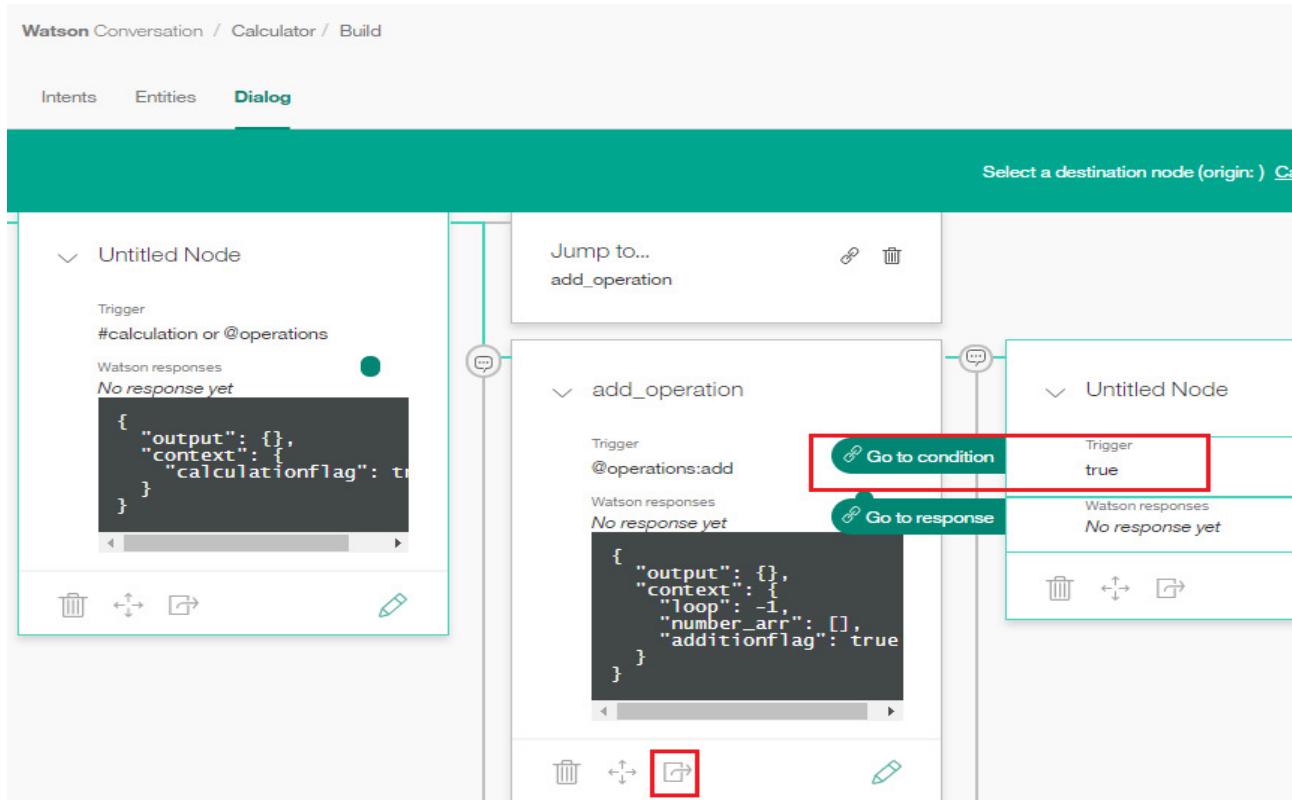
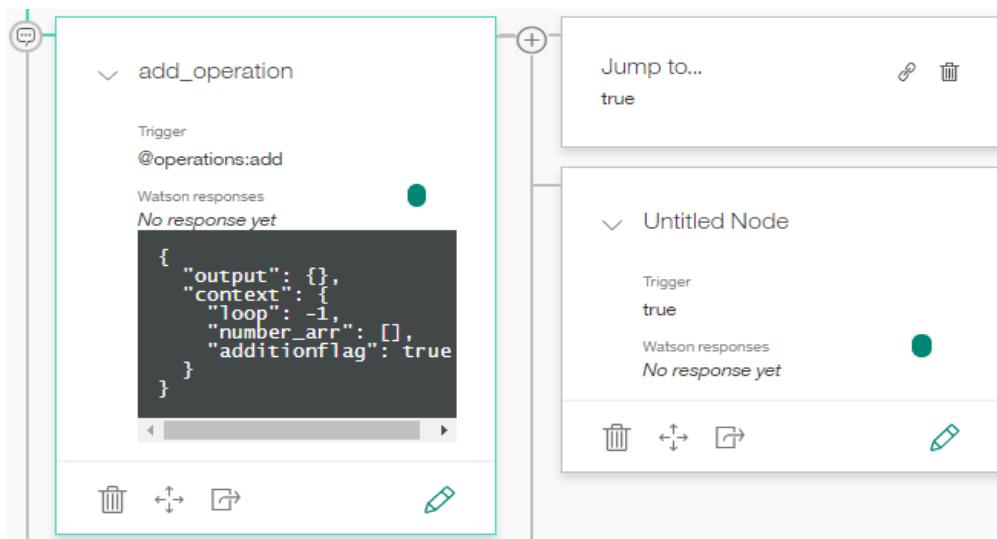


Figure 1-44 jump to true node

After adding the jump to node it should look like below figure.



13 Click (+) symbol right to the “true” node to add new node. This node will be executed to check if any system-number found in user text. If found any numerical data then store it in number_arr.(Figure 1-45)
Specify this information and then press Enter:

- 1 Under If “bot recognizes”, start typing “entities['sys-number']!=null”
- 2 Add one more condition by clicking (+) symbol besides first condition
- 3 Selected operation must be “and”
- 4 Add new condition “\$loop<entities['sys-number'].size()-1”
- 5 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:



- 6 Add below json string as shown in image 1-41.
- ```
{
 "context": {
 "loop": "<?context.loop + 1 ?>",
 "number": "<? entities['sys-number'].get($loop).value ?>",
 "number_arr": "<? context.number_arr.append($number) ?> ",
 "output": {}
 }
}
```

The screenshot shows the Watson Conversation interface in the "Dialog" tab. A red box highlights the "If bot recognizes:" condition in the "Untitled Node" configuration. Below it, a red box highlights the JSON response code. The JSON code defines a context object with variables for loop, number, number\_arr, and output.

**Watson Conversation / Calculator / Build**

**Intents Entities Dialog**

**Untitled Node**

Trigger: true  
Watson responses: No response yet

**Untitled Node**

Trigger:  
entities['sys-number']!=null and  
\$loop<entities['sys-number'].size()-1

Watson responses: No response yet

```
{
 "output": {},
 "context": {
 "loop": "<?context.loop + 1 ?>",
 "number": "<? entities['sys-number'].get($loop).value ?>",
 "number_arr": "<? context.number_arr.append($number) ?> ",
 "output": {}
 }
}
```

Name this node...  
If bot recognizes:  
entities['sys-number']!=null and \$loop<entities['sys-number'].size()-1  
Then respond with:  
1 {  
2 "context": {  
3 "loop": "<?context.loop + 1 ?>",  
4 "number": "<? entities['sys-number'].get(\$loop).value ?>",  
5 "number\_arr": "<? context.number\_arr.append(\$number) ?>"  
6 },  
7 "output": {}  
8 }

Figure 1-45 sys-number entity check node creation

14 Click jump to symbol on the @operations:add node to jump to next “true” node condition. (Figure 1-46)

- 1 Click  symbol and select “entities['sys-number'].....” node.
- 2 Click on “Go to condition”

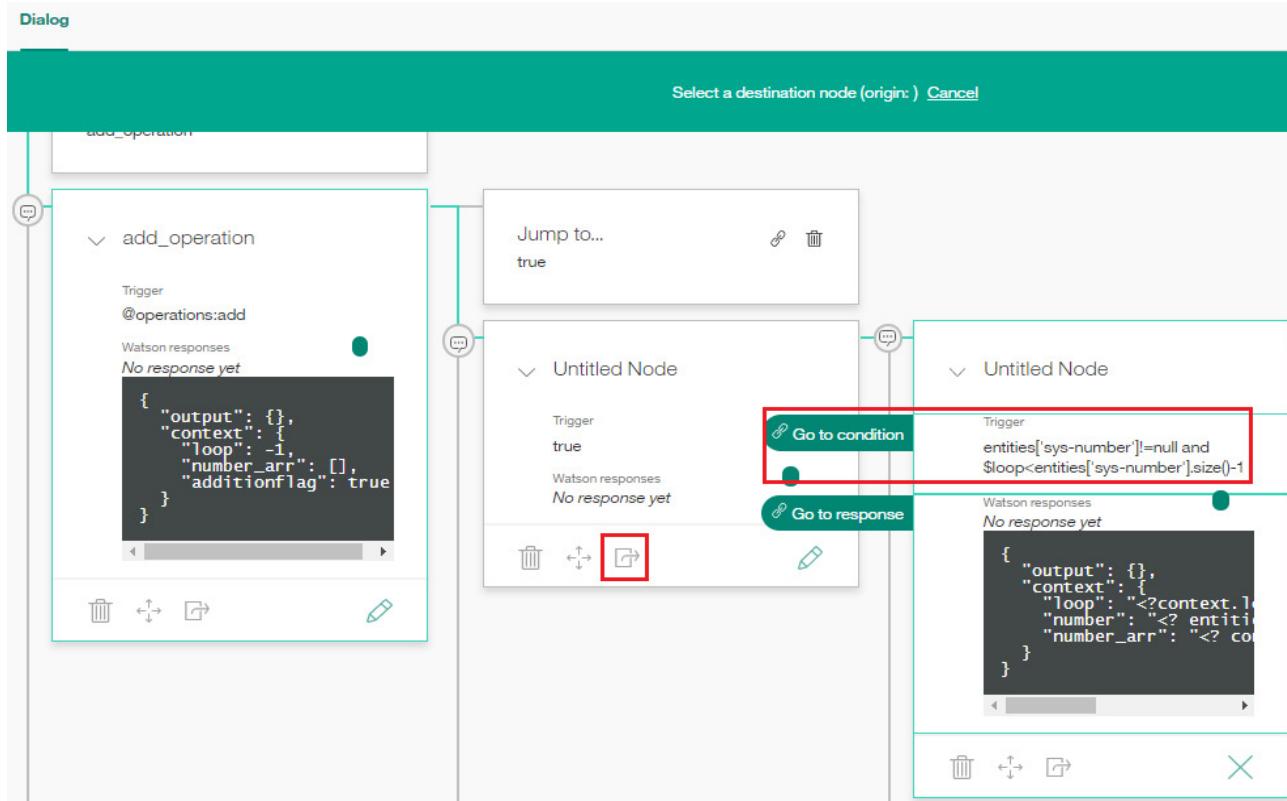
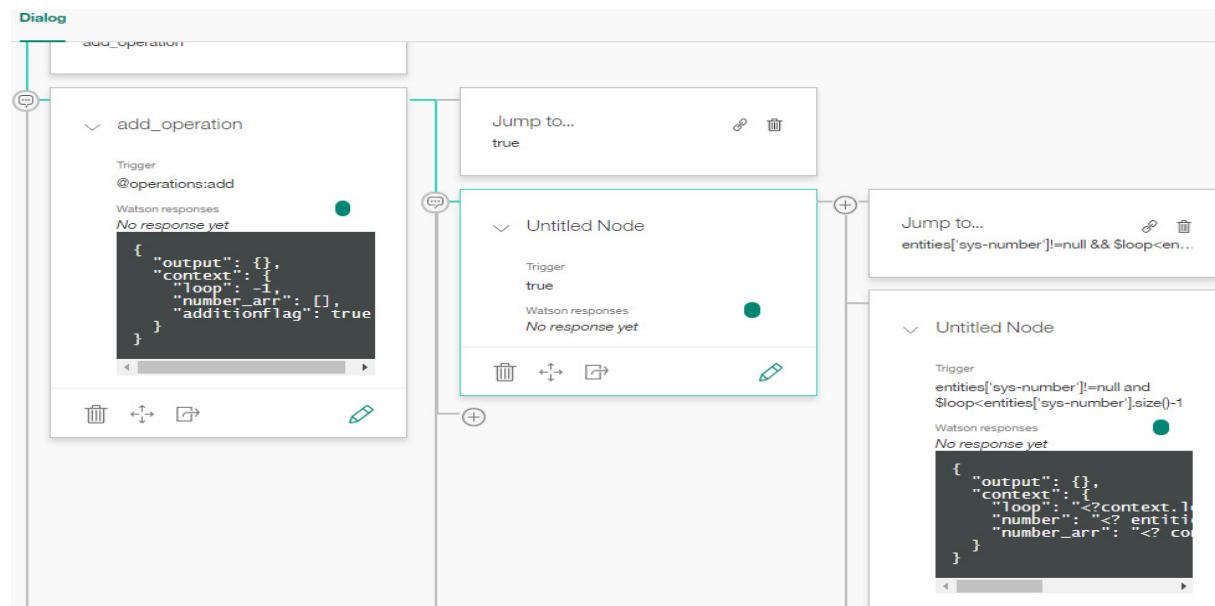


Figure 1-46 jump to true node

After adding the jump to node it should look like below figure.



15 Click jump to symbol on the @operations:add node to jump to next “true” node condition. (Figure 1-47)

- 1 Click  symbol and select “true” node.
- 2 Click on “Go to condition”

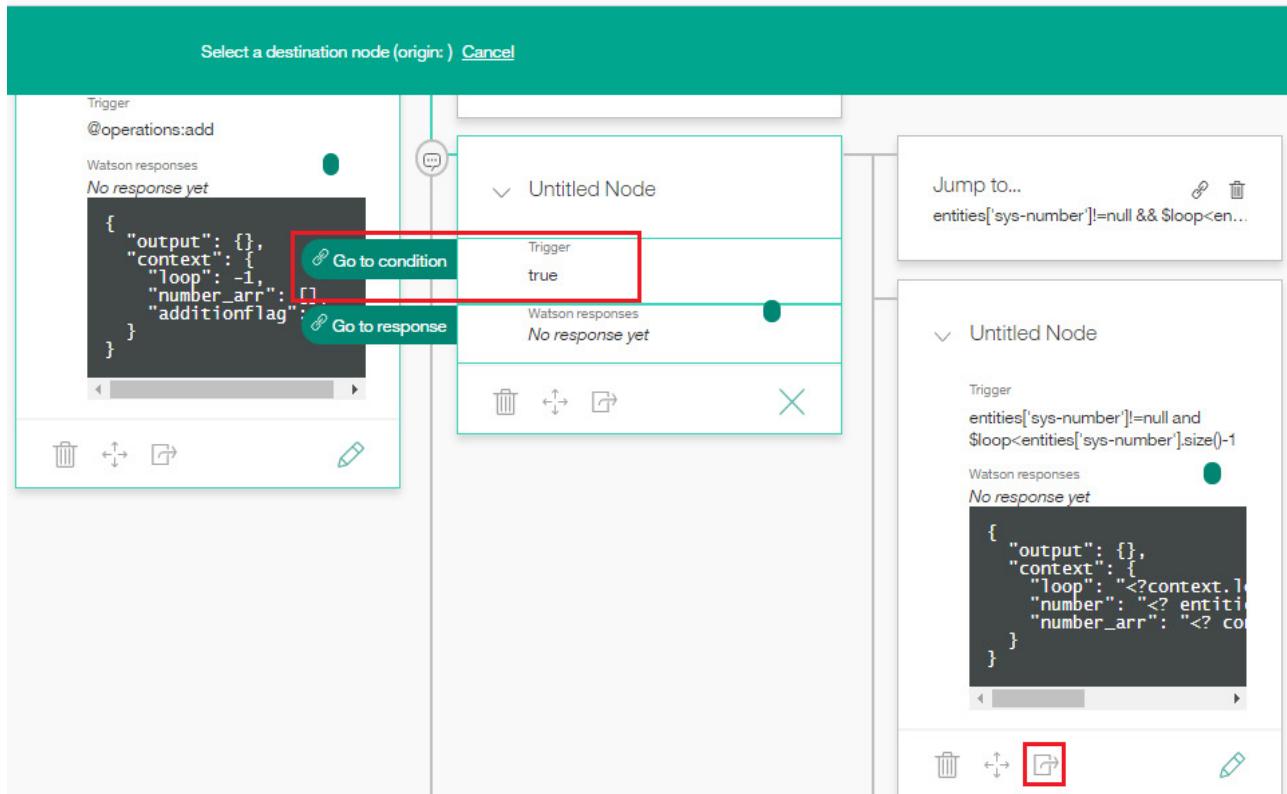
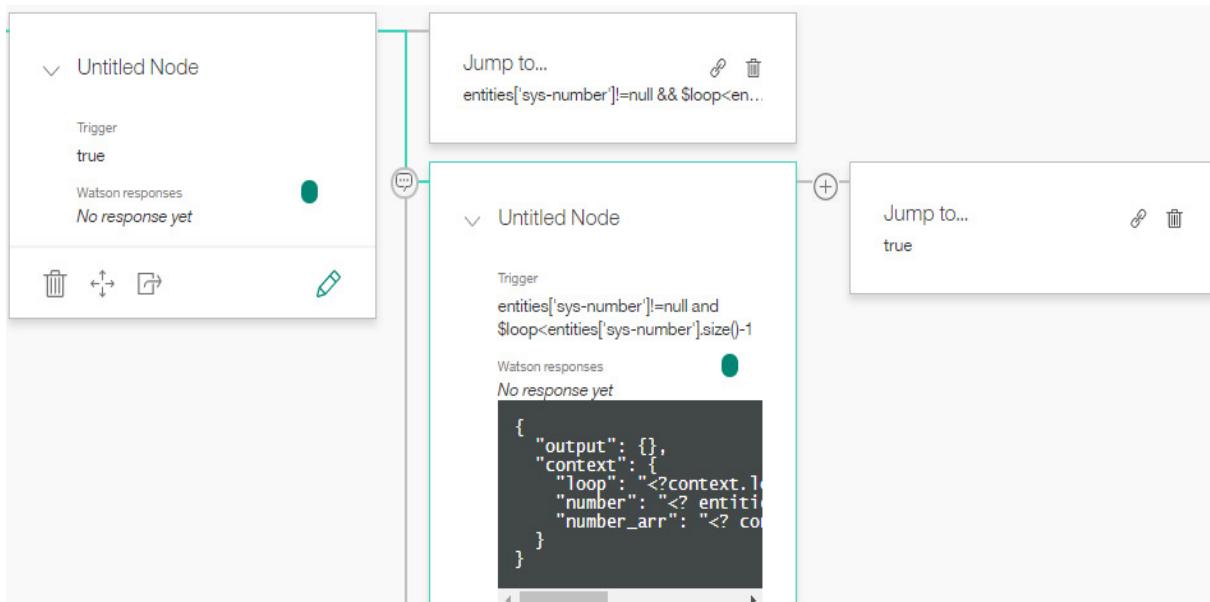


Figure 1-47 jump to true node

After adding the jump to node it should look like below figure.



16 Click (+) symbol below “true” node to add one more parallel node. (Figure 1-48)

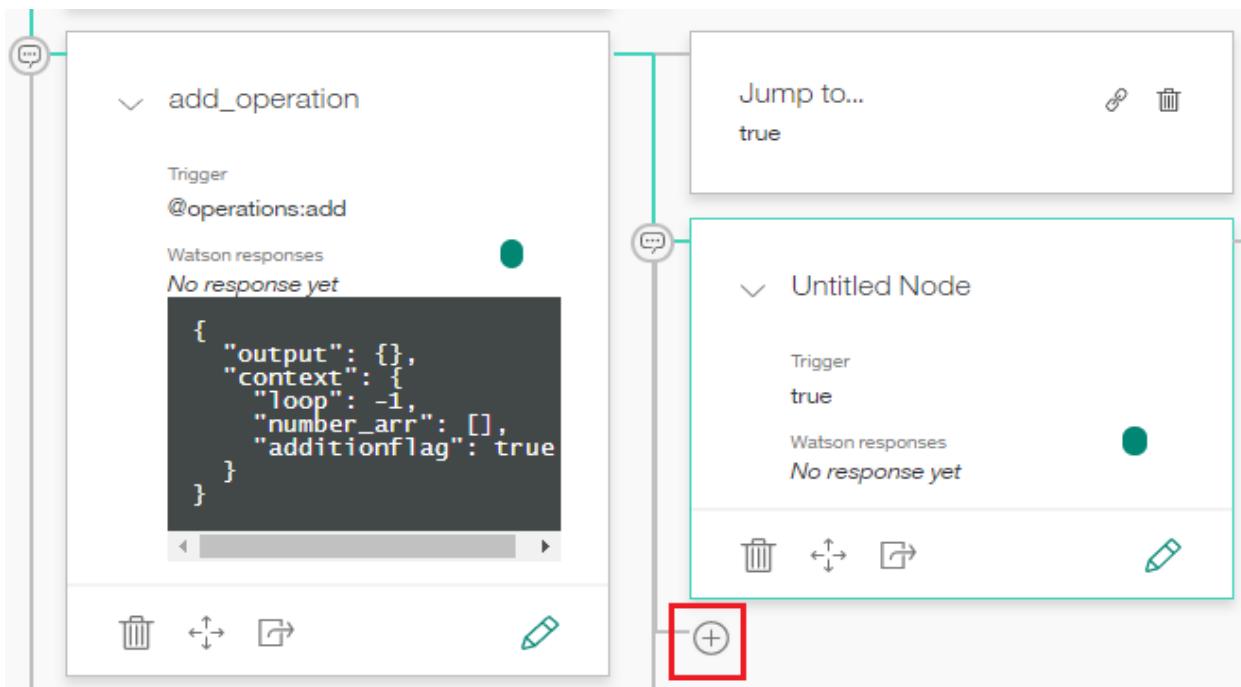


Figure 1-48 true node creation

17 Specify this information and then press Enter ( Figure 1-49):

- 1 Under If “bot recognizes”, enter condition “\$number\_arr.length>=2”
- 2 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:



- 3 Add below json string as shown in image 1-49.

```
{ "context": { "action": "add" }, "output": {} }
```

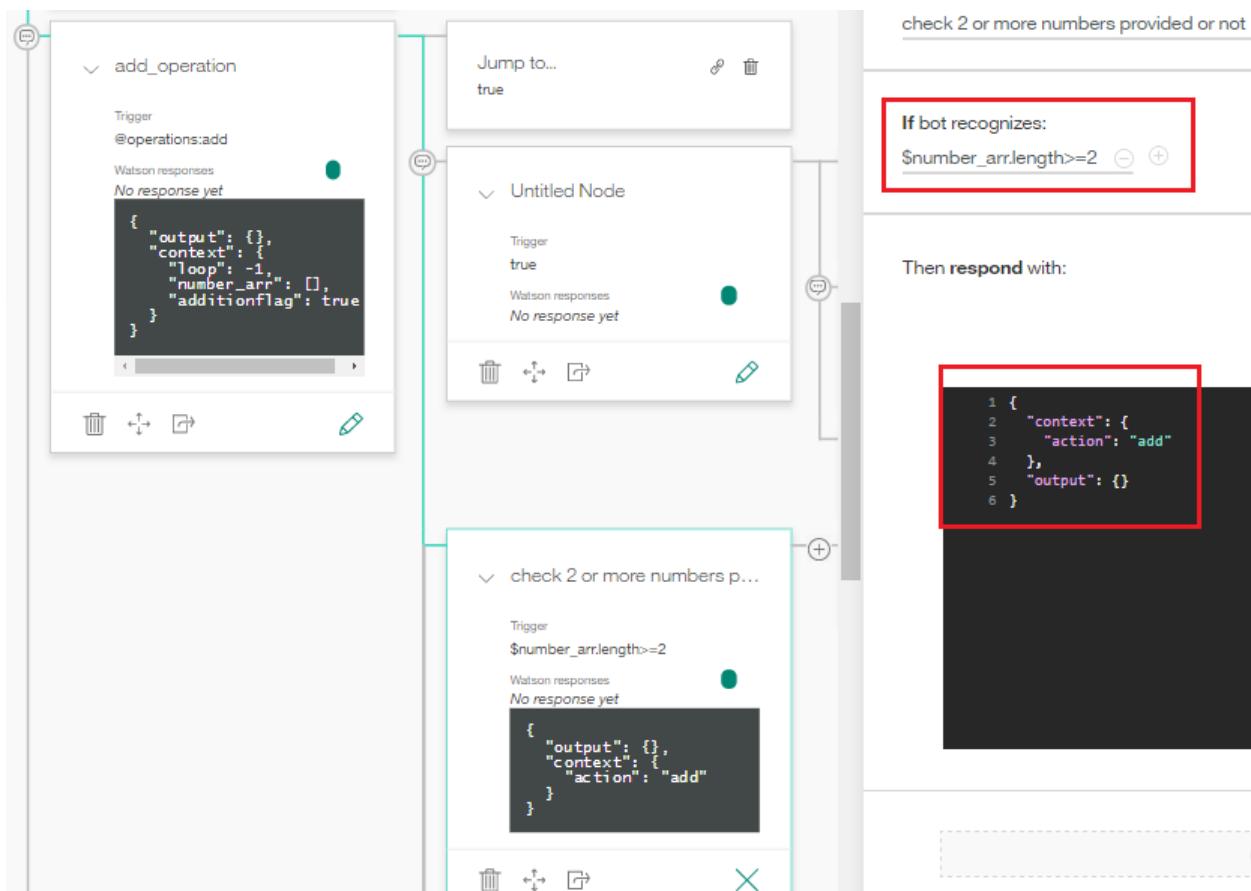


Figure 1-49 context variable number\_arr length check node creation

18 Click (+) symbol on the right side of “\$number\_arr.length>=2” node. Specify below information and then press Enter ( Figure 1-50):

- 1 Under If “bot recognizes”, enter condition “\$result!=null”
- 2 Add response condition text “The result is : \$result”

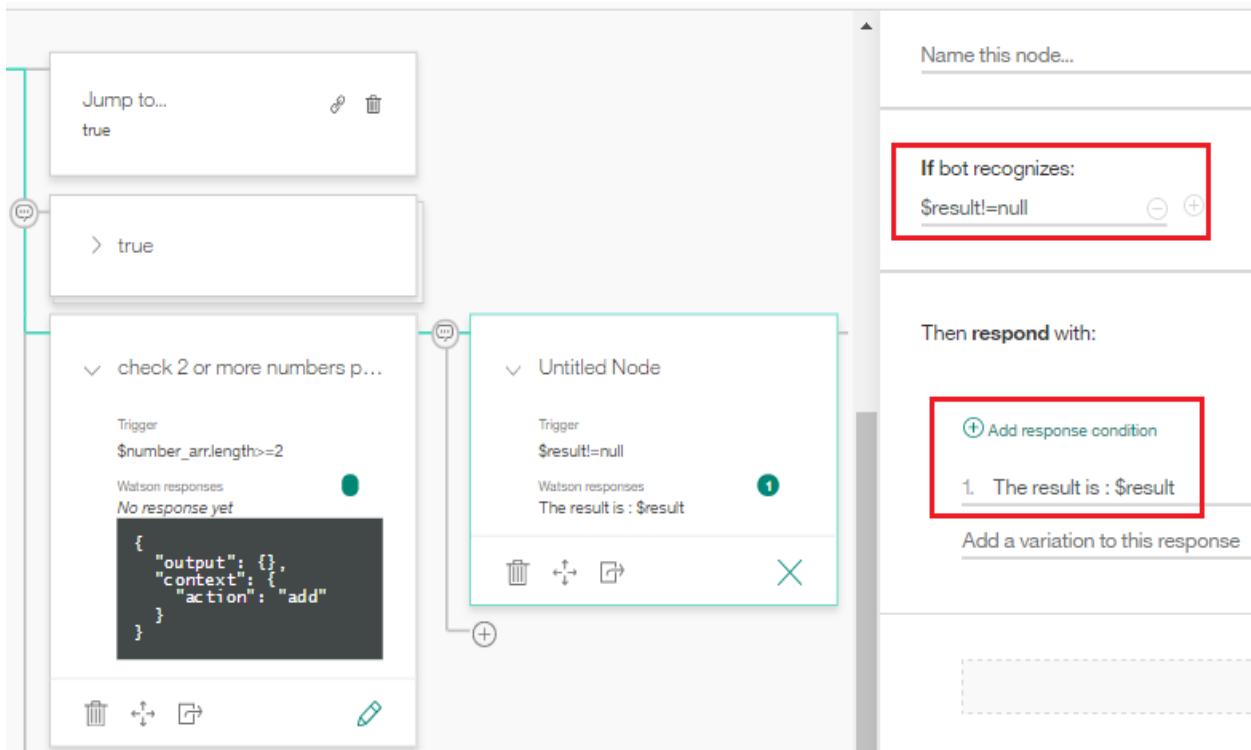


Figure 1-50 result node creation

19 Click (+) below the “entities['sys-number']!=null and \$loop<entities['sys-number'].size()-1 ” node. ( Figure 1-51):

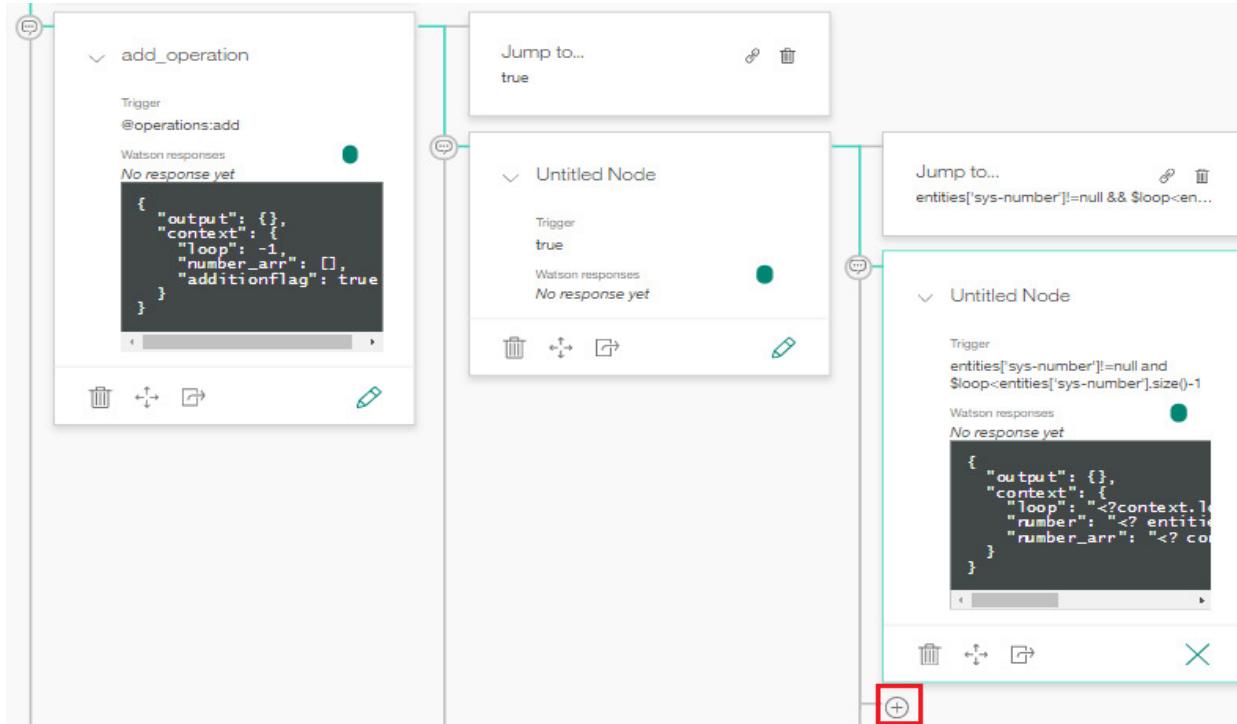


Figure 1-51 true node creation

20 Specify below information and then press Enter ( Figure 1-52):

- 1 Under If “bot recognizes”, enter condition “true”
- 2 Leave Add response condition blank.

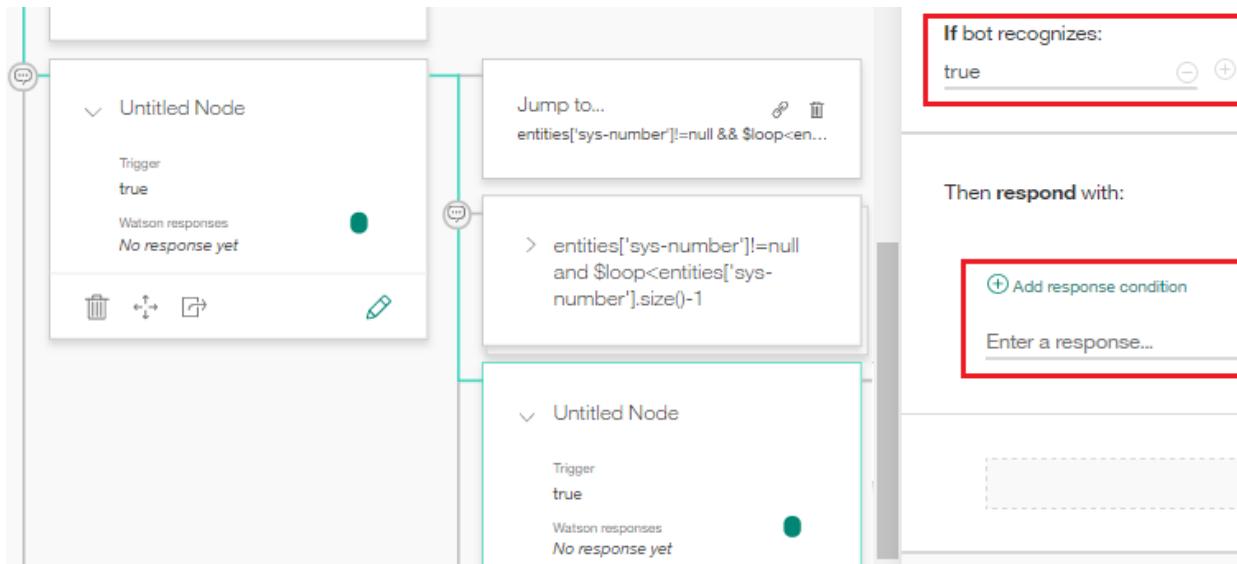


Figure 1-52 true node creation

21 Click jump to symbol on the “true” node(created in previous step no 20) to jump to next “\$number\_arr.length>=2” node(created in previous step no 17) . (Figure 1-53)

- 1 Click  symbol and select “true” node.
- 2 Click on “Go to condition” of “\$number\_arr.length>=2” node

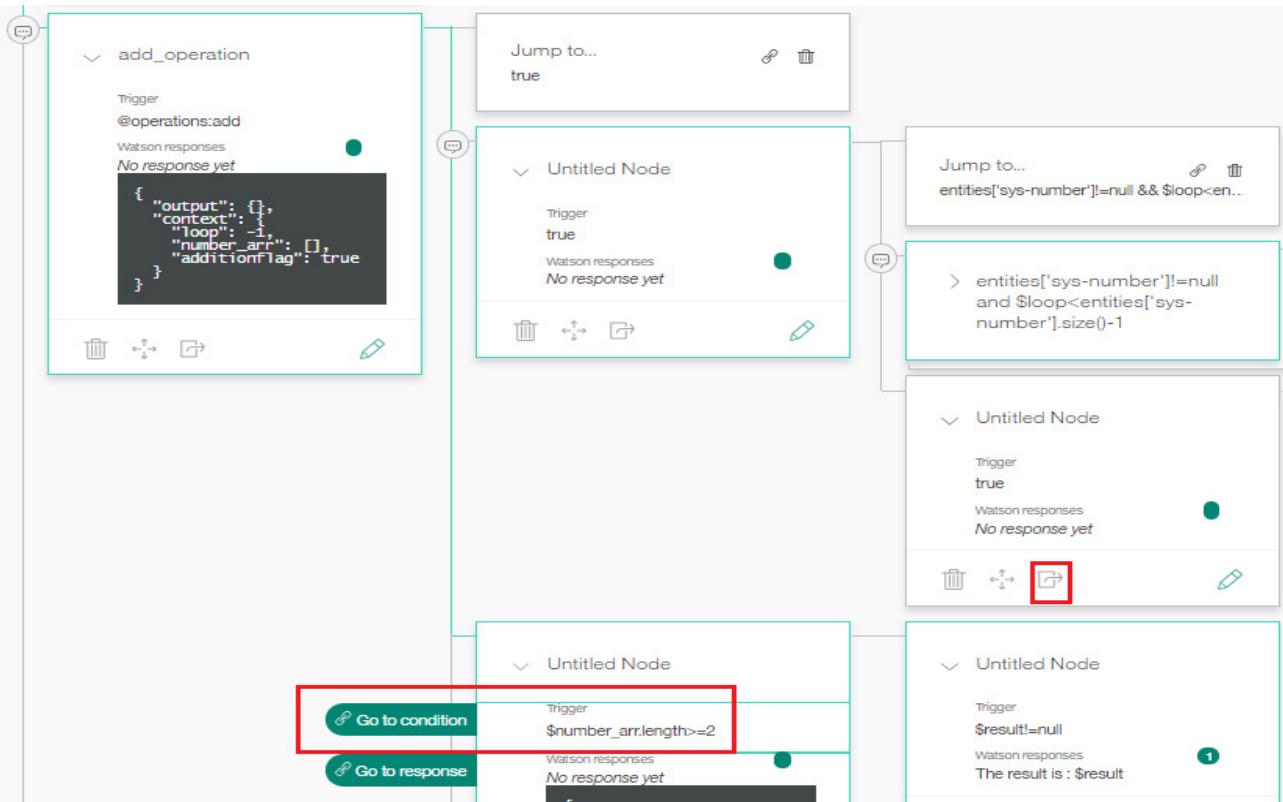
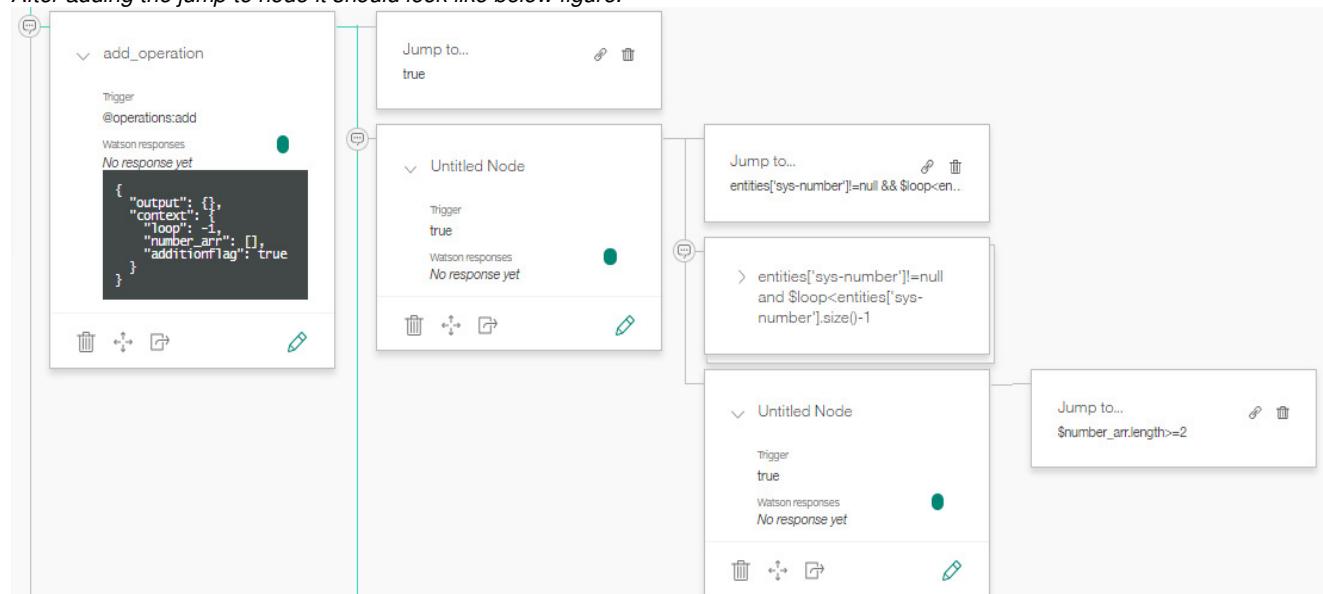


Figure 1-53 jump to node

After adding the jump to node it should look like below figure.



22 Click (+) below the “\$number\_arr.length>=2” node. ( Figure 1-54):

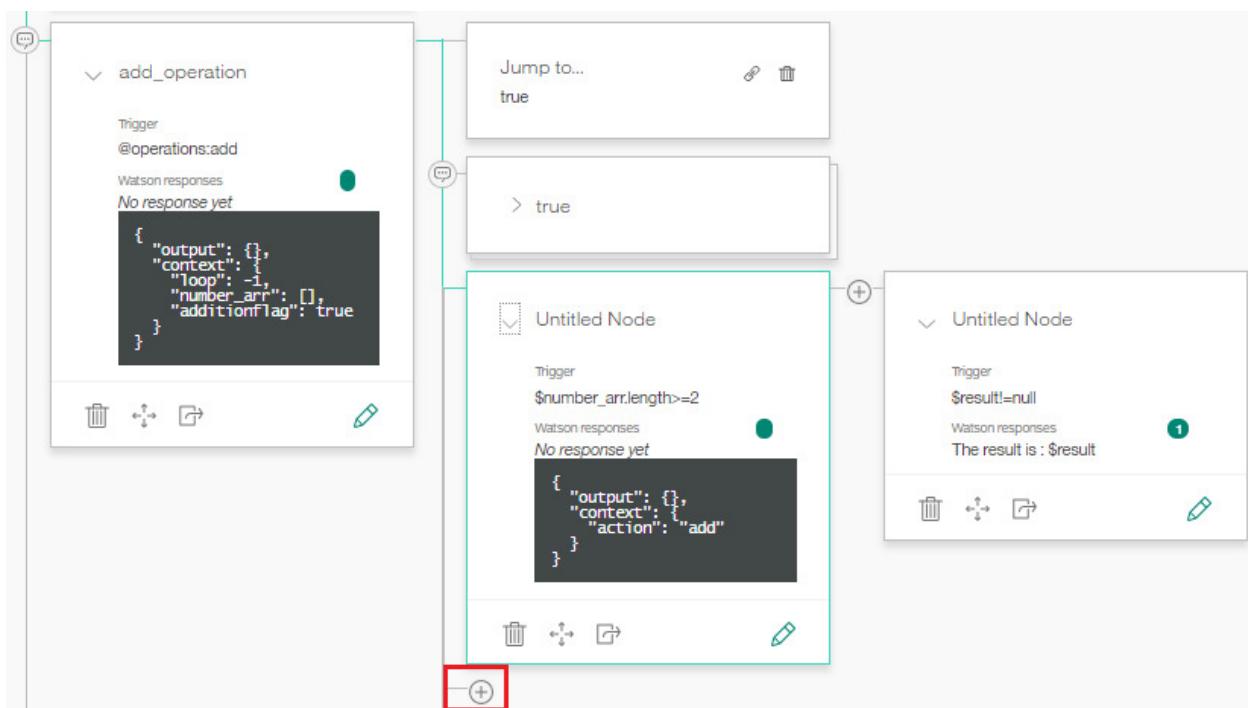


Figure 1-54 sys-number entities check node

23 Specify below things and press enter (figure 1-55)

- 1 Under If "bot recognizes", enter condition "entities['sys-number']!=null"
- 2 Click on (+) symbol besides first condition to add one more condition
- 3 Select operation "and"
- 4 Enter next condition "entities['sys-number'].size()==1"
- 5 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:



- 6 Add below json string as shown in image 1-55.

```
{
 "context": {
 "loop": -1
 },
 "output": {
 "text": {
 "values": [
 "Oops. I found only 1 operand(number) and required at least 2 operand
 (number) to perform addition operation.
Kindly provide other number/s."
],
 "selection_policy": "sequential"
 }
 }
}
```

The screenshot displays the Watson Assistant interface with several panels. On the left, there's a panel titled 'add\_operation' containing a JSON configuration. In the center, there's a 'Jump to...' section with three options: 'true', '> true', and '> \$number\_arr.length>=2'. To the right, there's a 'If bot recognizes:' section with a condition 'entities['sys-number']!=null and entities['sys-number'].size()==1'. Below this is a 'Then respond with:' section containing the JSON string from step 6. A large red box highlights the 'Then respond with:' section and its content. At the bottom right, there's a button labeled 'Add another response'.

Figure 1-55

24 Click jump to symbol on the “entities['sys-number']!=null and entities['sys-number'].size()==1” node to jump to next “User Input” node as shown in Figure 1-56.

1 Click  symbol on “entities['sys-number']!=null and entities['sys-number'].size()==1” node.

2 Click on “User Input” node as shown in figure 1-56

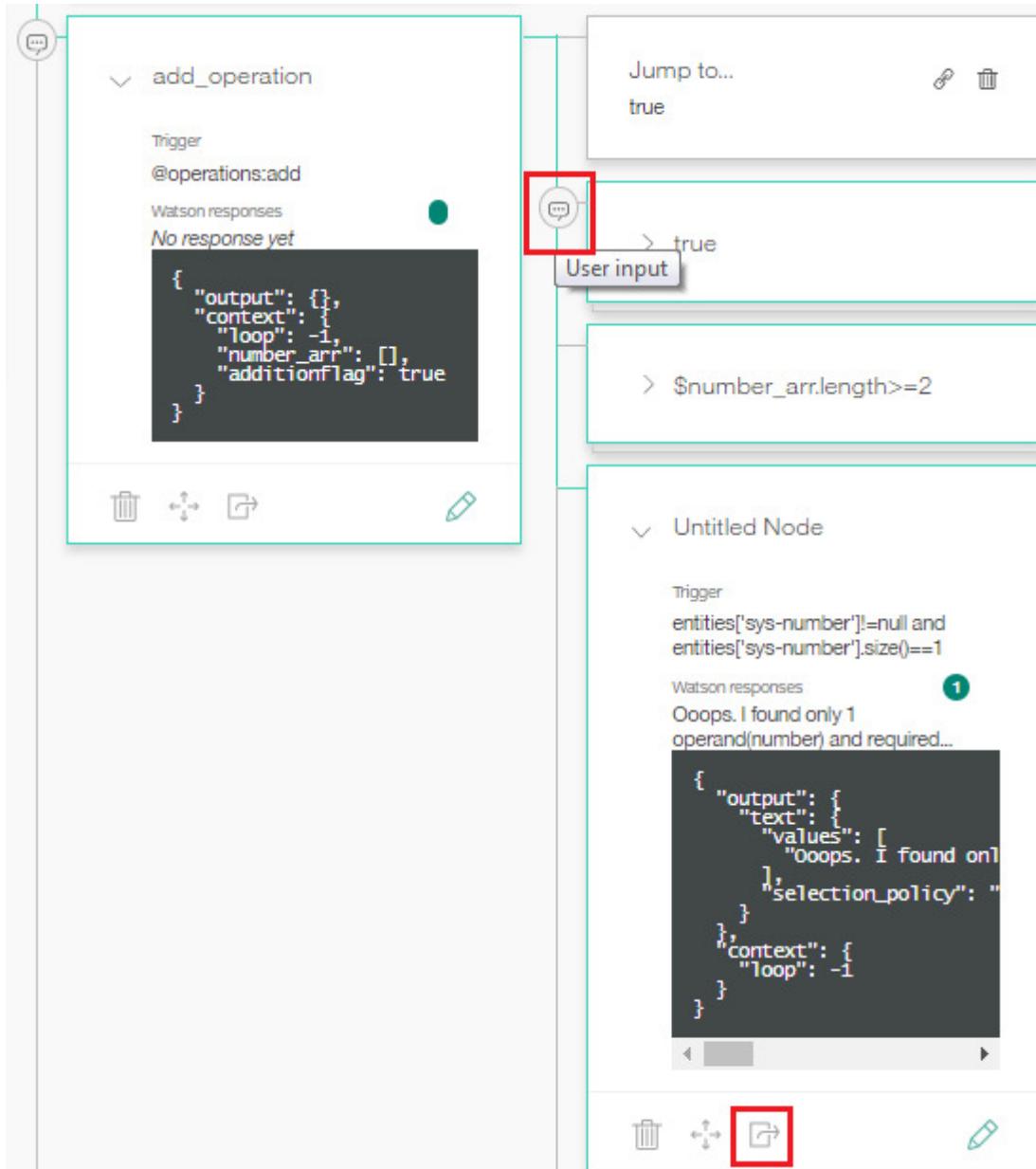
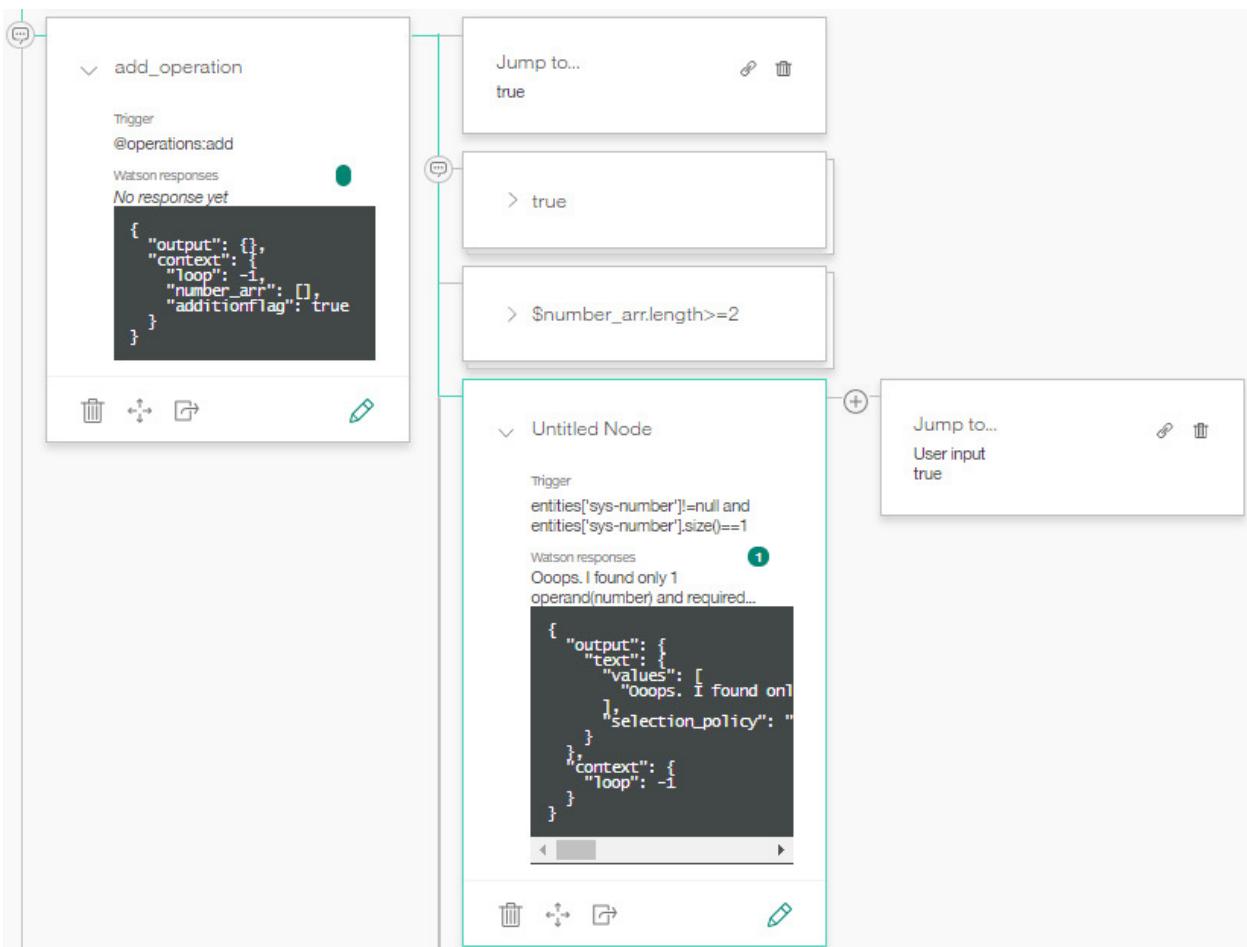


Figure 1-56

After adding jump to node will looks like below



25 Click (+) below the “entities['sys-number']!=null and entities['sys-number'].size()==1 ” node.  
( Figure 1-57):

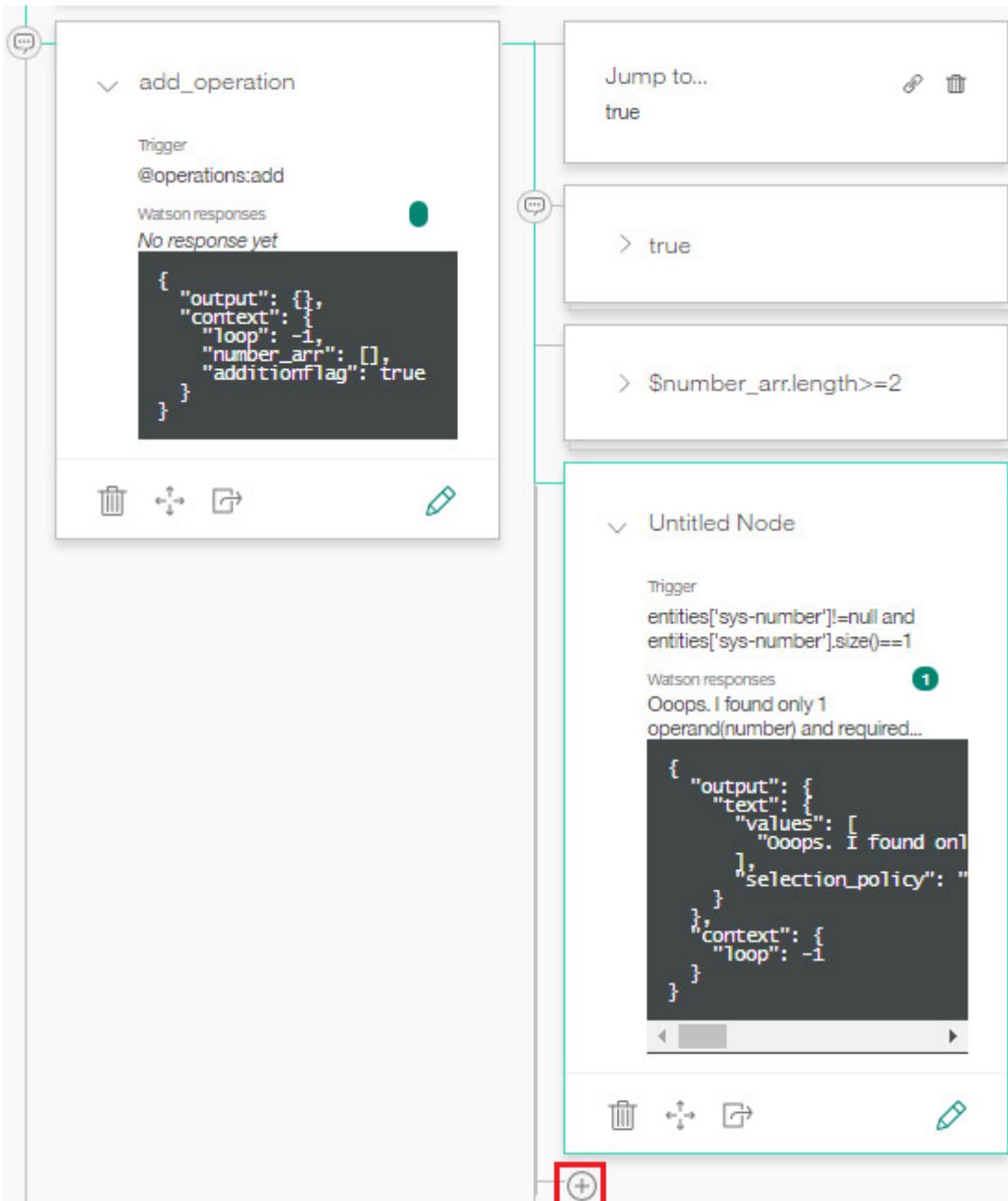


Figure 1-57

26 Specify below things and press enter (figure 1-58)

- 1 Under If "bot recognizes", enter condition "\$additionflag==true"
- 2 Click on the three ellipses to open JSON editor to add the response condition.

Then respond with:

The screenshot shows the IBM Watson Assistant interface with a node configuration window open. The node is titled "Untitled Node".

**Trigger:** \$additionflag==true

**Watson responses:** Kindly help me with the numbers that you want to add.

**JSON Editor Content:**

```
1 {
2 "context": {
3 "additionflag": false
4 },
5 "output": {
6 "text": {
7 "values": [
8 "Kindly help me with the numbers that you want to add."
9],
10 "selection_policy": "sequential"
11 }
12 }
13 }
```

**Condition:** If bot recognizes: \$additionflag==true

**Then respond with:**

```
1 {
2 "context": {
3 "additionflag": false
4 },
5 "output": {
6 "text": {
7 "values": [
8 "Kindly help me with the numbers that you want to add."
9],
10 "selection_policy": "sequential"
11 }
12 }
13 }
```

Figure 1-58

27 Click jump to symbol on the “\$additionflag==true” node to jump to next “User Input” node as shown in Figure 1-59.

1 Click  symbol on “\$additionflag==true” node.

2 Click on “User Input” node as shown in figure 1-59

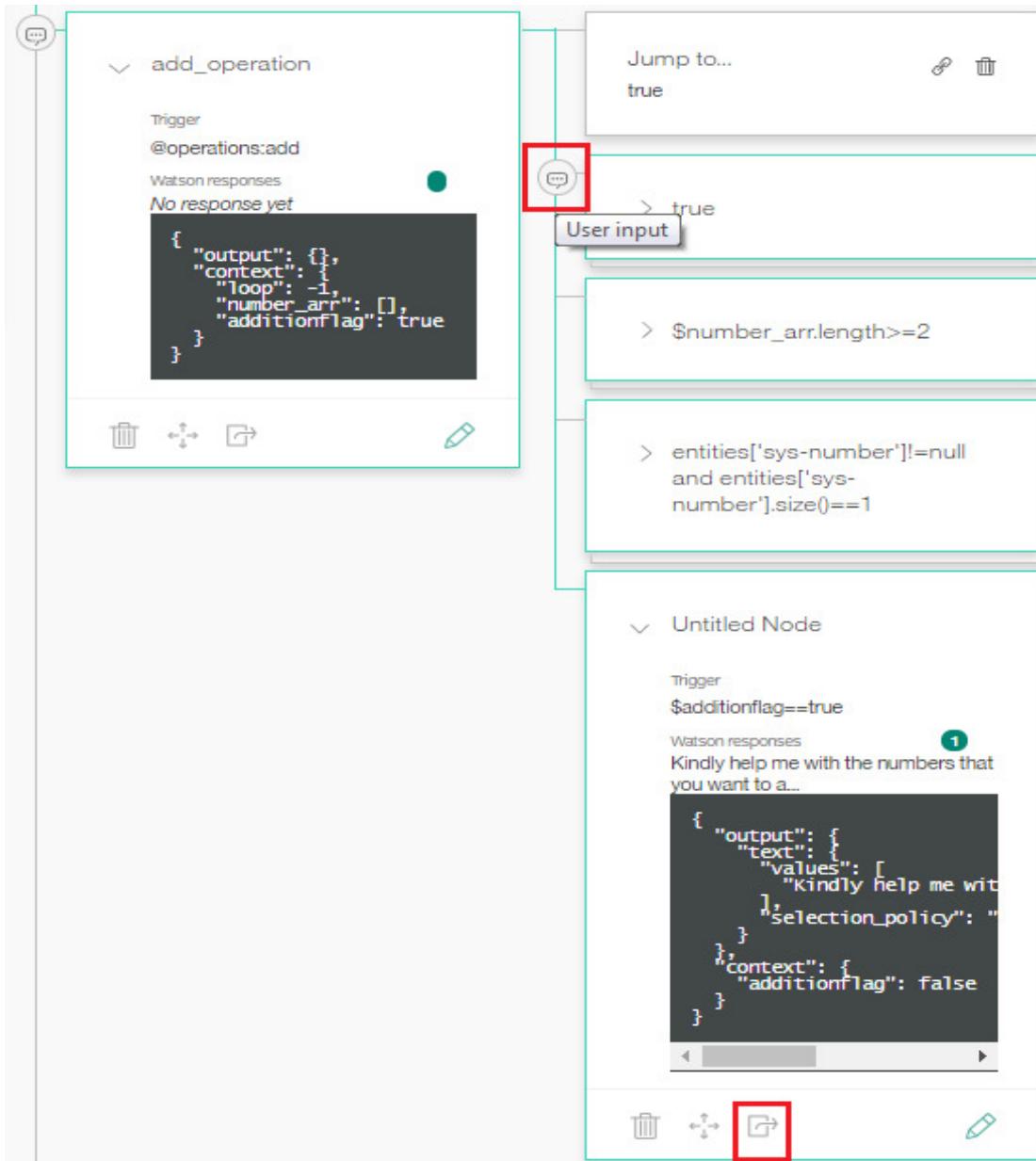
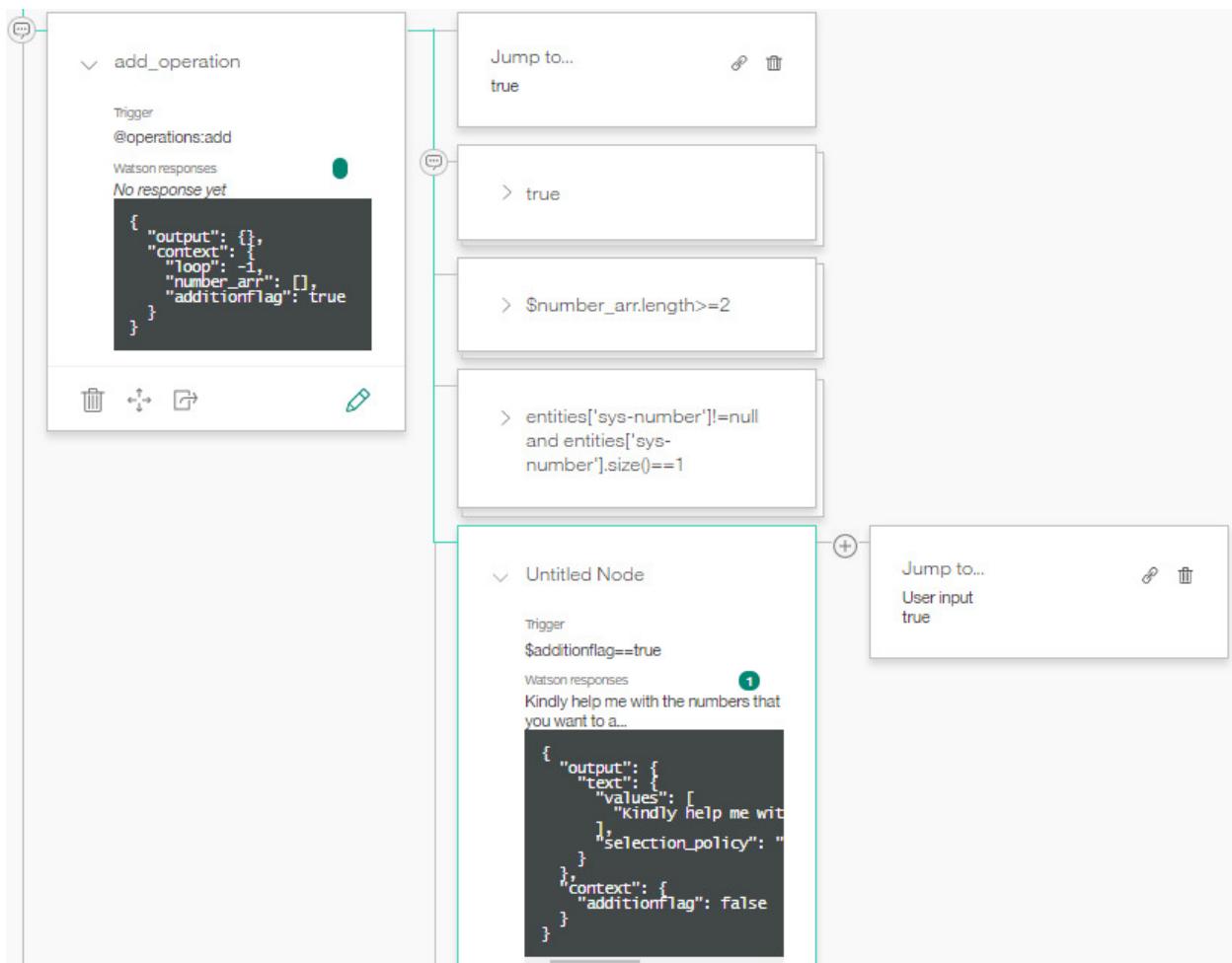


Figure 1-59

After adding jump to node it will looks like below



28 Click on “anything\_else” node.(Figure 1-60)

Specify below things and press enter:

- 1 Add below message in “Add Response condition”

“Sorry!! I am unable to answer your query at this time as i am not trained on it. Kindly connect with help center. I can help you in performing addition and multiplication operations.”

The screenshot shows the Dialog builder interface with the 'Dialog' tab selected. On the left, there's a tree view of entities:

- #thank\_you and intents[0].confidence>0.7
- #bye and intents[0].confidence>0.7
- #calculation or @operations
- anything\_else (selected, indicated by a red border)

On the right, under the selected 'anything\_else' node, the configuration pane shows:

- If bot recognizes:** anything\_else (with a minus sign and plus sign button)
- Then respond with:**
- Add response condition** (button with a plus sign)
- 1.** Sorry!! I am unable to answer your query at this time as i am no (highlighted with a red box)
- Add a variation to this response** (button)
- Add another response** (button)

Figure 1-60 anything\_else node

29 Similarly create the flow for Multiplication, Subtraction and Division operations.

## Test the conversation flow

Follow these steps:

1. Click the **Ask Watson** icon at the top right ([Figure 1-61](#)).

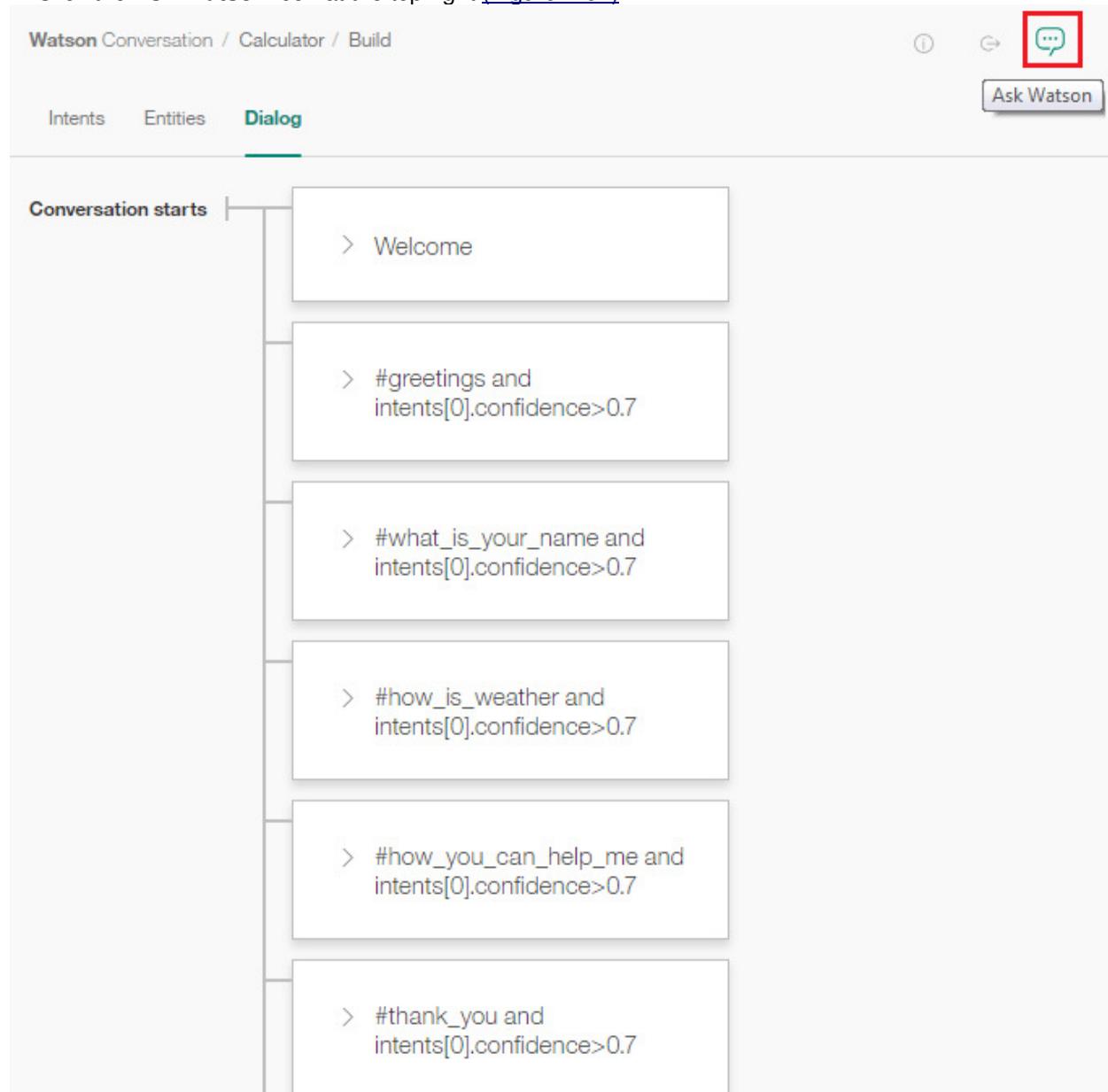


Figure 1-61 Calculator Conversation workspace

Test the dialog. For each user input, the Conversation service analyzes intents and entities and responds according to the conversation flow in the dialog ([Figure 1-62](#)).

The screenshot shows the Watson Assistant interface. On the left, there is a tree view of dialog nodes:

- Node 1:** Trigger: #what\_is\_your\_name and intents[0].confidence>0.7. Watson response: I am Cognitive Calculator, I could help you with a... (with a note '1').
- Node 2:** Trigger: #how\_is\_weather and intents[0].confidence>0.7.
- Node 3:** Untitled Node. Trigger: #how\_you\_can\_help\_me and intents[0].confidence>0.7. Watson response: I can help you in calculating numbers. I can perfo... (with a note '1').
- Node 4:** Untitled Node. Trigger: #thank\_you and intents[0].confidence>0.7. Watson response: Glad !! I could help you today. (with a note '1').
- Node 5:** Untitled Node. Trigger: #bye and intents[0].confidence>0.7.

On the right, the "Try it out" window shows the conversation flow:

- User message: hi. Response: #greetings.
- User message: Hi !! How may i help you today?
- User message: how you can help me. Response: #how\_you\_can\_help\_me.
- User message: I can help you in calculating numbers. I can perform addition and multiplication of numbers for you.
- User message: i want to perform addition of numbers. Response: #calculation @operations:add.
- User message: Kindly help me with the numbers that you want to add.
- User message: 5 and 7. Response: Irrelevant @sys-number:5 @sys-number:7.
- User message: thanks. Response: #thank\_you.
- User message: My pleasure.
- User message: bye. Response: #bye.
- User message: Bye. See you again.

*Figure 1-62 Testing the dialog*

## 1.3 Developing the Cognitive Calculator chatbot application

This section shows how to develop the Cognitive Calculator chatbot application that integrates with the Conversation service in Node.js.

### Create a Node.js application on Bluemix

Follow these steps:

From the Bluemix dashboard, click **Create App**.

From the Cloud Foundry Apps section, click **SDK for Node.js**.

In the Create a Cloud Foundry App window ([Figure 1-63](#)) enter the following information, and then click **Create**:

- App name: conv-201-xxx-calc
- Host name: conv-201-xxx-calc

Replace xxx with a random value; the host name of the application must be unique.

The screenshot shows the 'Create a Cloud Foundry App' interface. On the left, there's a brief description of the 'SDK for Node.js™' template, which allows for developing, deploying, and scaling server-side JavaScript® apps with ease. It provides enhanced performance, security, and serviceability. To the right, there are input fields for 'App name' (set to 'conv-201-xxx-calc'), 'Host name' (set to 'conv-201-xxx-calc'), and 'Domain' (set to 'mybluemix.net'). Below these fields, there's a 'Pricing Plans' section with a note that monthly prices are shown for Egypt. At the bottom, there are links for 'Need Help?' (Contact Bluemix Sales) and 'Estimate Monthly Cost' (Cost Calculator), followed by a large blue 'Create' button.

*Figure 1-63 Create Node.js application*

**Stop:** Wait until the application is started to proceed. The application status should indicate Running, as shown in [Figure 1-64](#)

## Configure the application

Follow these steps:

Configure the application environment variables. Add the WORKSPACE\_ID environment variable with the Workspace ID of your Calculator Conversation workspace ([Figure 1-64](#)):

- 1 Click **Runtime** on the left navigation bar.
- 2 Click the **Environment variables** tab.
- 3 Click **Add**.
- 4 For the name, specify WORKSPACE\_ID.
- 5 For the value, specify the Workspace ID value that you created earlier
- 6 Click **Save**.

The screenshot shows the IBM Bluemix Cloud Foundry Apps interface. On the left, a sidebar has 'Runtime' selected. In the main area, an application named 'conv-201-xxx-calc' is shown as 'Running'. The 'Environment variables' tab is active. Under 'User defined', there is one entry:

| NAME         | VALUE                    | ACTION |
|--------------|--------------------------|--------|
| WORKSPACE_ID | 44202034-4a11-4e2f-8c8b- |        |

At the bottom of the environment variables section are buttons for 'Add', 'Save', 'Reset', and 'Export'.

*Figure 1-64 Adding WORKSPACE\_ID as environment variable*

**Stop:** Wait until the application is restaged before you continue.

Bind the Conversation service to your application ([Figure 1-65](#)):

- 1 Click **Connections** from the left toolbar.
- 2 Click **Connect existing**.
- 3 Click **Conversation**.
- 4 Click **Connect**.

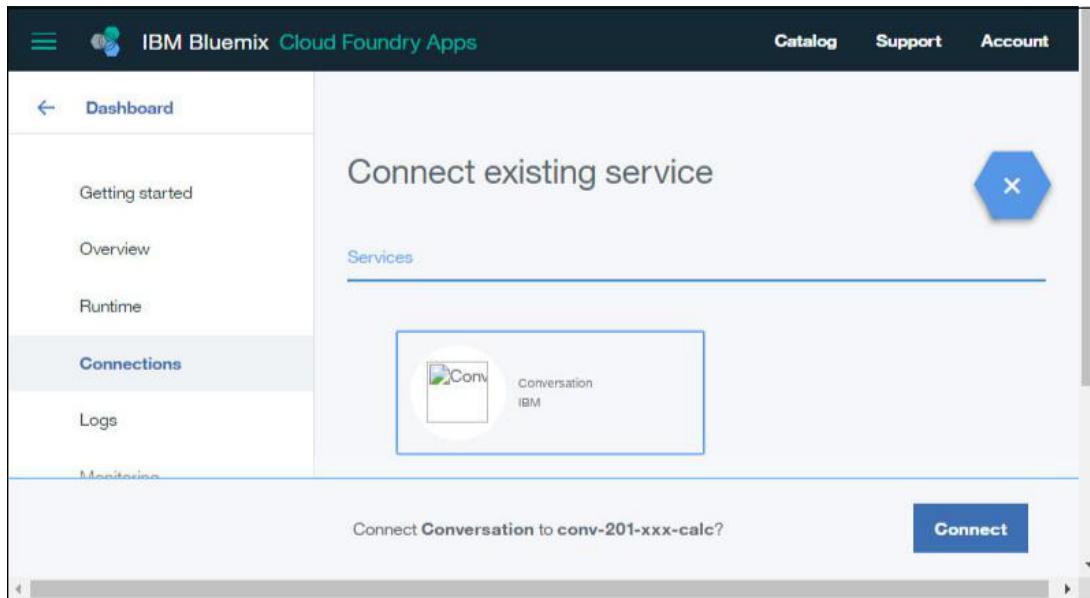


Figure 1-65 Connect existing service

e. Click **Restage** to make the service available for use by the application ([Figure 1-66](#)).

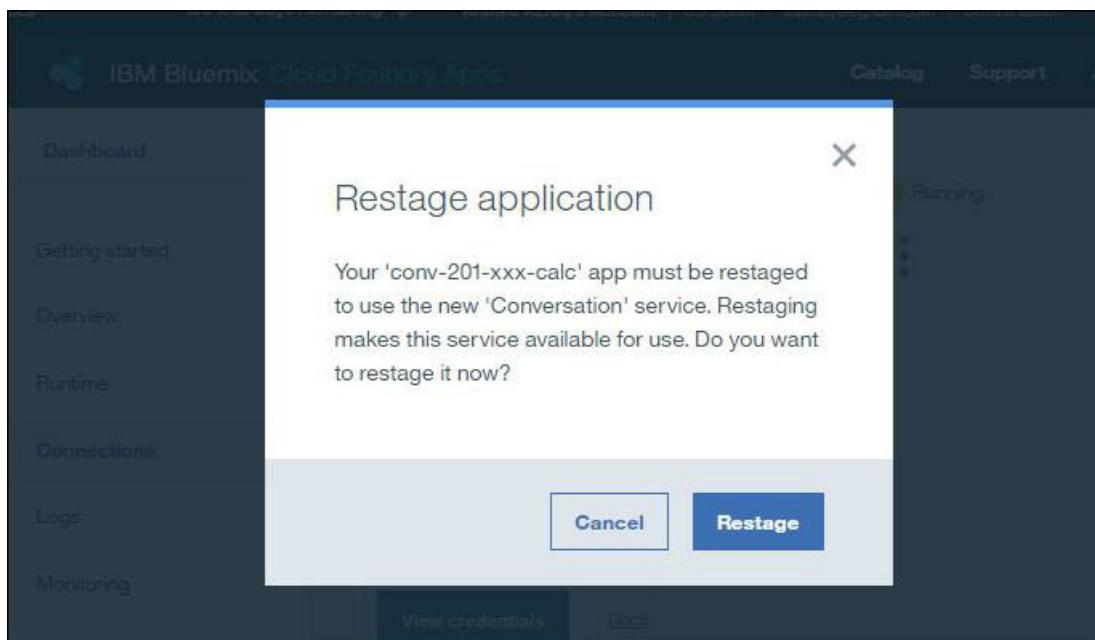


Figure 1-66 Restage application

**Stop:** Wait until the restaging is completed and the application is in a running state before you continue.

## Clone the Conversation sample application

In the next steps, you clone a sample Node.js application, which is a simple chatbot, to your Bluemix workspace.

Click **Overview** in the left navigation toolbar (Figure 1-67).

Scroll to the Continuous delivery panel, on the right, and click **Enable**.

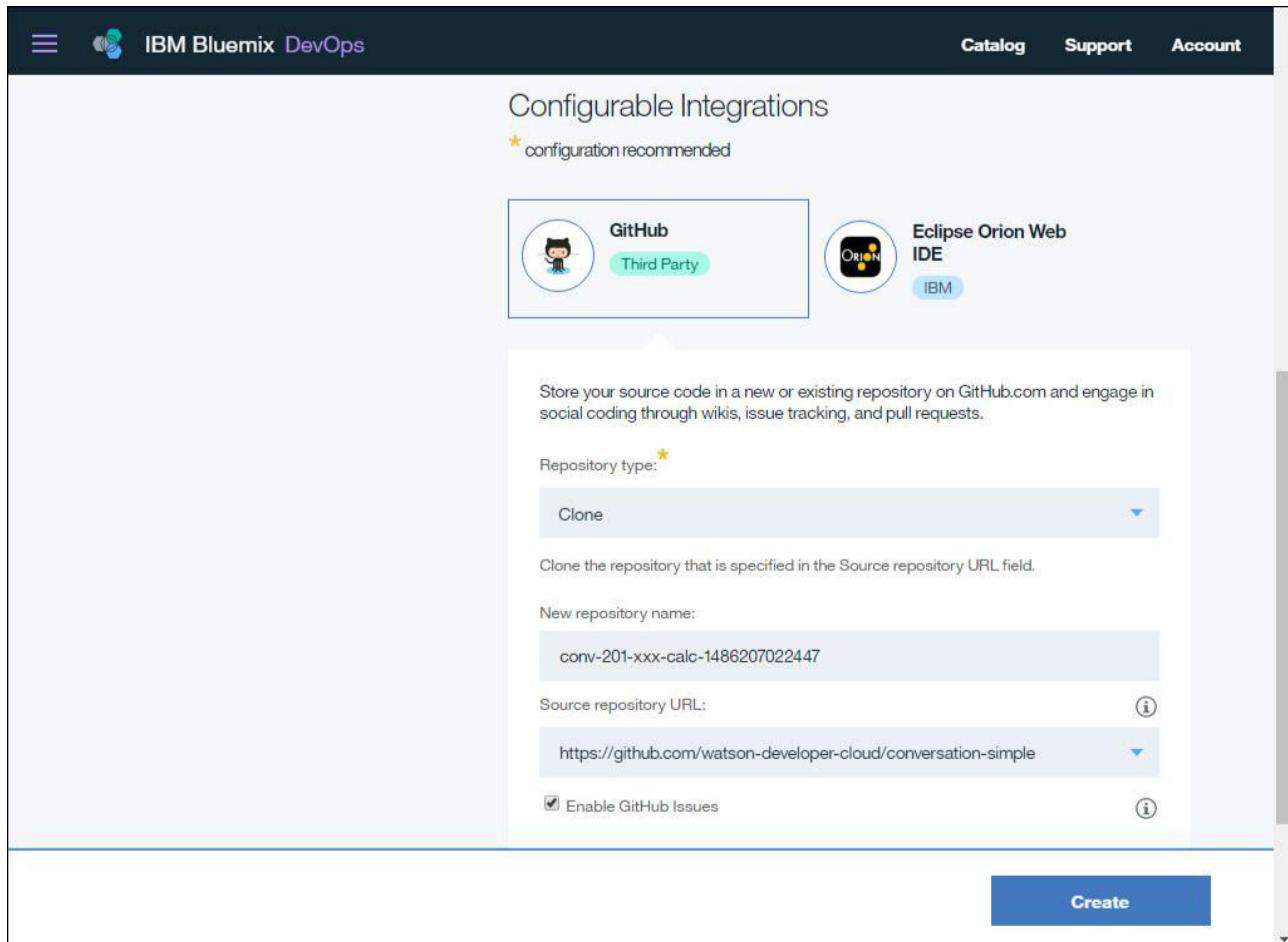
This enables the continuous delivery toolchain. With it, you can automate builds, tests, and deployments through the Delivery Pipeline, GitHub, and more.

The screenshot shows the IBM Bluemix Cloud Foundry Apps interface. The top navigation bar includes 'Catalog', 'Support', and 'Account'. The left sidebar has a 'Dashboard' link and sections for 'Getting started', 'Overview', 'Runtime', 'Connections', 'Logs', and 'Monitoring'. The main area displays an application named 'conv-201-xxx-calc' with a status of 'Running' (green circle). Below the app name are buttons for 'Rollback to DEA', 'View app', and other actions. To the right, there's an 'Activity feed' section showing log entries: 'started conv-201-xxx-calc app' (Feb 4, 2017), 'started conv-201-xxx-calc app' (Feb 4, 2017), 'updated conv-201-xxx-calc app' (changed routes, Feb 4, 2017), and 'created conv-201-xxx-calc app' (Feb 4, 2017). Further right is a 'Continuous delivery' panel with a message: 'Continuous delivery is not enabled for this app. Enable continuous delivery to automate builds, tests, and deployments through the Delivery Pipeline, GitHub, and more.' It features a prominent blue 'Enable' button and a note: 'Looking for the ADD GIT button to set up a project at [hub.jazz.net](#)? Click [here](#)'.

Figure 1-67 Application overview

A new tab opens ([Figure 1-68](#)):

- 1 Scroll to **Configurable Integrations** and click **GitHub**.
- 2 Keep the repository type as Clone.
- 3 Keep the default new repository name.
- 4 For the Source repository URL, specify this GitHub repository URL:  
<https://github.com/dakspate/calculator-bot>
- 5 Click **Create**.



The screenshot shows the IBM Bluemix DevOps interface. At the top, there is a navigation bar with icons for three, IBM Bluemix DevOps, Catalog, Support, and Account. Below the navigation bar, the main content area has a title "Configurable Integrations" and a note "configuration recommended". There are two sections: "GitHub" (selected) and "Eclipse Orion Web IDE". The GitHub section contains fields for "Repository type:" (set to "Clone"), "New repository name:" (set to "conv-201-xxx-calc-1486207022447"), "Source repository URL:" (set to "https://github.com/watson-developer-cloud/conversation-simple"), and a checked checkbox for "Enable GitHub Issues". A large blue "Create" button is at the bottom right.

Figure 1-68 GitHub configurations

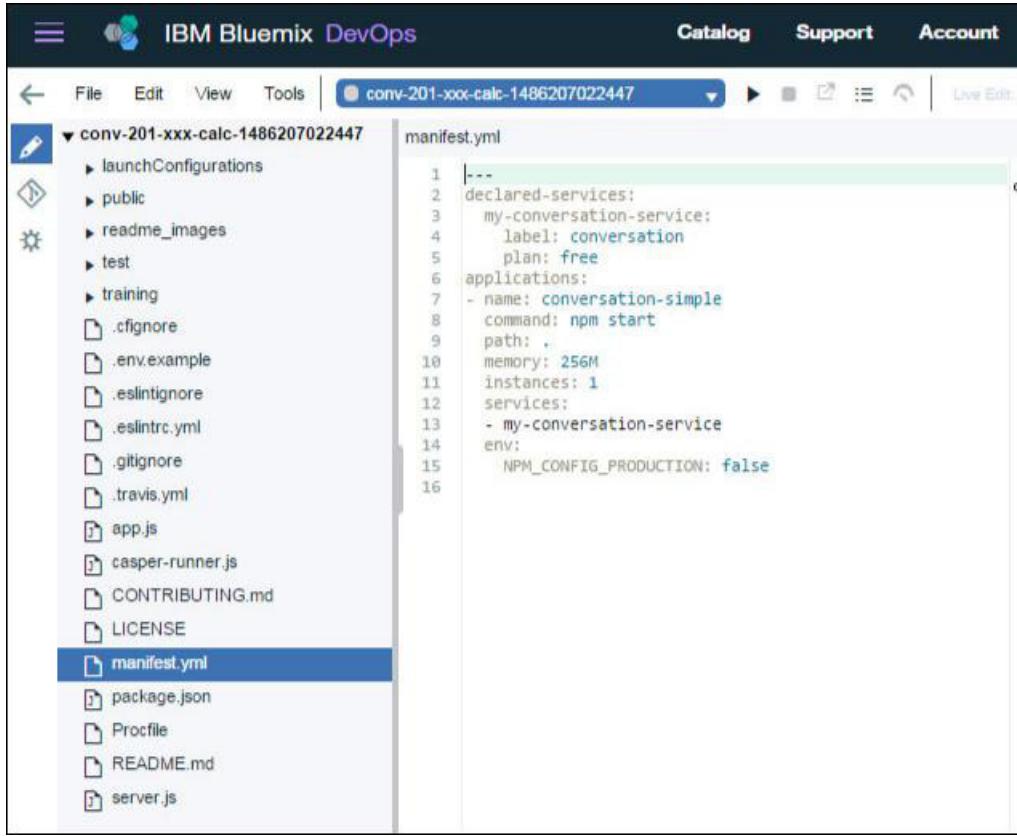
## Edit the application code

In this section, you edit the code to implement the calculation functionality: 1. In the Toolchains window, click **Eclipse Orion Web IDE** ([Figure 1-69](#)).

The screenshot shows the IBM Bluemix DevOps interface. At the top, there's a navigation bar with 'IBM Bluemix DevOps' and links for 'Catalog', 'Support', and 'Account'. Below the navigation bar, the title 'conv-201-xxx-calc-148620...' is displayed along with a 'View app' button and a more options menu. On the left, a sidebar titled 'Toolchains' has sections for 'Overview', 'Connections', and 'Manage'. The main area is titled 'Toolchains' and contains three columns: 'THINK', 'CODE', and 'DELIVER'. Each column has a card for 'Issues' (configured), 'GitHub' (configured), and 'Delivery Pipeline' (configured). Below these is a card for 'Eclipse Orion Web IDE' (configured). A blue button 'Add a Tool' with a plus sign is located in the 'CODE' column. In the bottom right corner, there's a 'Feedback' button.

Figure 1-69 Toolchains window: Click Eclipse Orion Web IDE

Update the manifest.yml file with the host name and service name (Figure 1-70): a. In the list of files on the left, click the manifest.yml file.



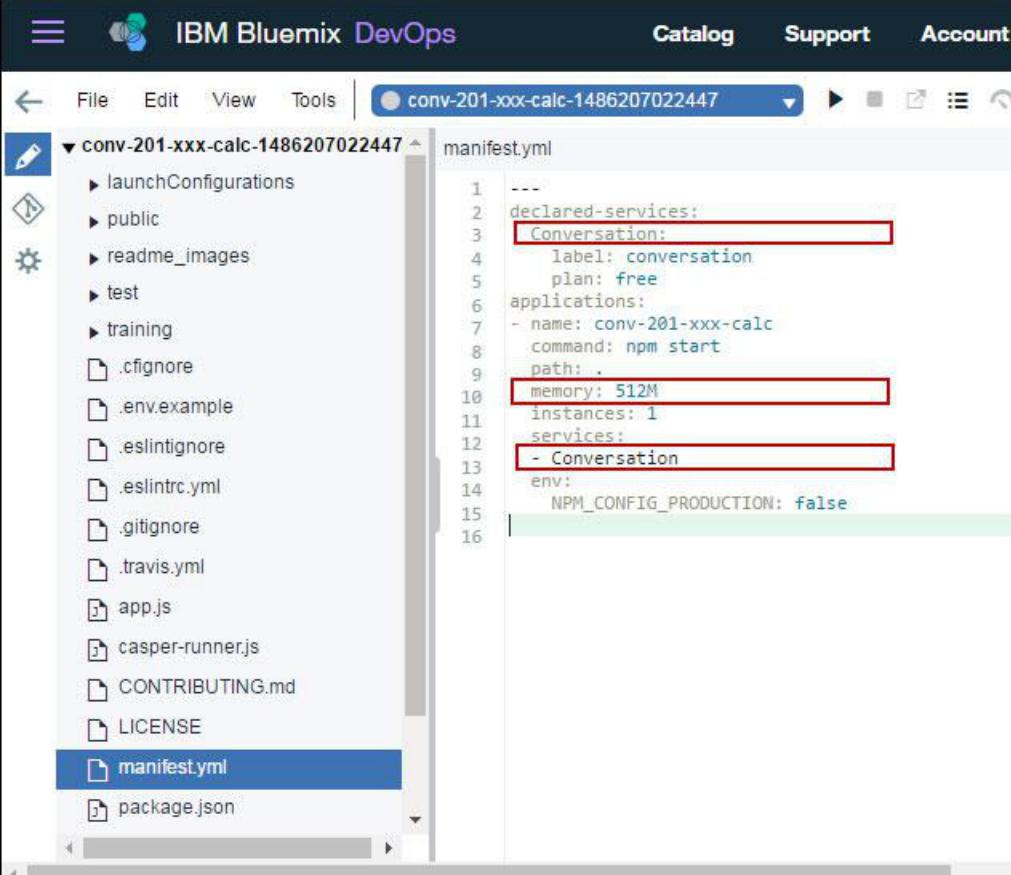
The screenshot shows the IBM Bluemix DevOps interface. At the top, there are navigation links: Catalog, Support, and Account. Below the header, a breadcrumb trail indicates the project path: conv-201-xxx-calc-1486207022447. On the left, a sidebar lists various files and folders, including launchConfigurations, public, readme\_images, test, training, .cignore, .env.example, .eslintignore, .eslintrc.yml, .gitignore, .travis.yml, app.js, casper-runner.js, CONTRIBUTING.md, LICENSE, manifest.yml (which is highlighted in blue), package.json, Procfile, README.md, and server.js. To the right of the sidebar, the content of the manifest.yml file is displayed in a code editor:

```
1 ---
2 declared-services:
3 my-conversation-service:
4 label: conversation
5 plan: free
6 applications:
7 - name: conversation-simple
8 command: npm start
9 path: .
10 memory: 256M
11 instances: 1
12 services:
13 - my-conversation-service
14 env:
15 NPM_CONFIG_PRODUCTION: false
16
```

Figure 1-70 The manifest.yml file before update

In the manifest.yml file shown in [Figure 1-71](#), update this information:

- 1 Update the Conversation service to match the name of the Conversation service instance created in earlier step.
- 2 Update the application name to match the name of your application. To do this, update line 7 to conv-201-xxx-calc (where xxx is the value that you used to make your application and host names unique in earlier step).
- 3 Increase the memory to 512M, by updating line 10.



The screenshot shows the IBM Bluemix DevOps interface. The top navigation bar includes 'Catalog', 'Support', and 'Account'. The left sidebar shows a project structure for 'conv-201-xxx-calc-1486207022447' containing files like 'manifest.yml', 'package.json', and various configuration and test files. The 'manifest.yml' file is selected and open in the main editor area. The code content is as follows:

```
1 ---
2 declared-services:
3 Conversation:
4 label: conversation
5 plan: free
6 applications:
7 - name: conv-201-xxx-calc
8 command: npm start
9 path: .
10 memory: 512M
11 instances: 1
12 services:
13 - Conversation
14 env:
15 NPM_CONFIG_PRODUCTION: false
16
```

Lines 3, 10, and 13 are highlighted with red boxes, indicating the specific fields to be updated according to the instructions in the text above.

Figure 1-71 The manifest.yml file after the update

## Push the changes to Git

Follow these steps:

1. Click the **Git** icon on the left toolbar ([Figure 1-72](#)).

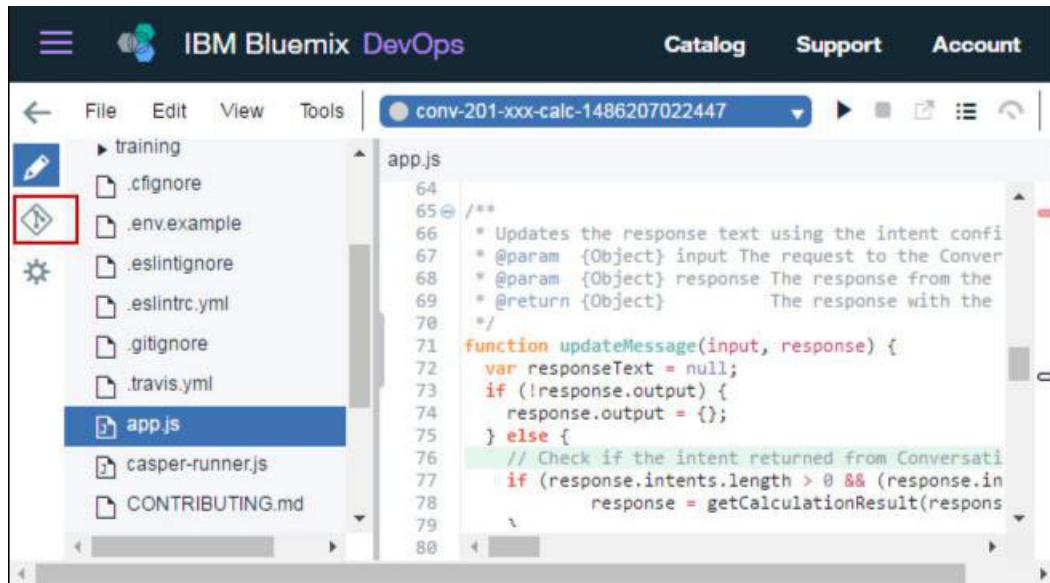


Figure 1-72 Click the Git icon in the IBM Bluemix DevOps page

2. Enter any descriptive commit message (such as Edit the application logic to perform the calculation functionality), and click **Commit** ([Figure 1-73](#)).

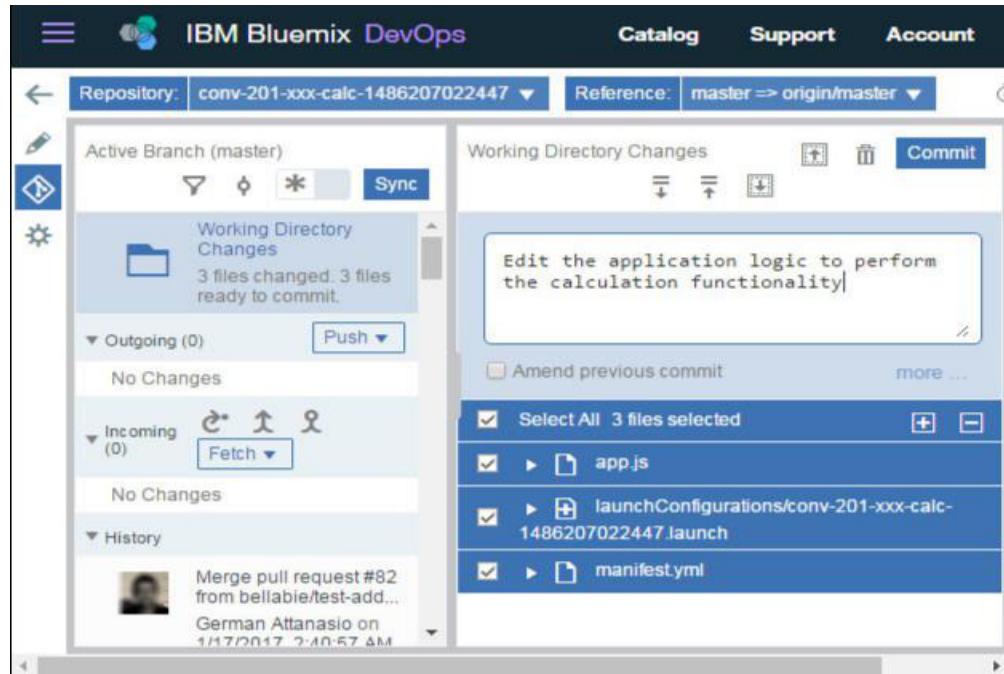


Figure 11-73 DevOps Git: Commit changes

3. Click **Push** to push your committed changes to the remote branch ([Figure 1-74](#)).

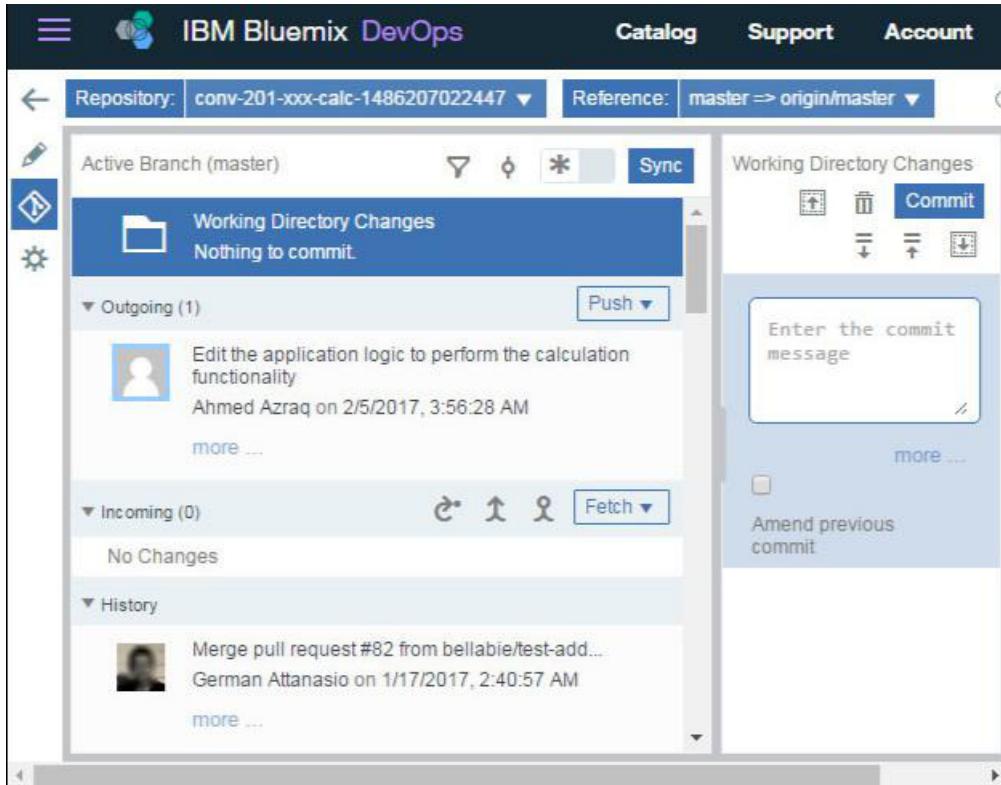


Figure 1-74 DevOps Git: Push changes to remote branch

4. Return to the **Toolchains** tab, and click on **Delivery Pipeline** (Figure 1-75).

The screenshot shows the IBM Bluemix DevOps interface. At the top, there is a navigation bar with the IBM Bluemix DevOps logo, Catalog, Support, and Account links. Below the navigation bar, the title "conv-201-xxx-calc-1486207..." is displayed, along with a "View app" button and a more options menu. On the left, a sidebar titled "Toolchains" has sections for Overview, Connections, and Manage. The "Overview" section is currently selected. In the main area, there are three columns: THINK, CODE, and DELIVER. Each column contains a card for a tool, all of which are marked as "Configured". The THINK column has a card for "Issues" (conv-201-xxx-calc-148...). The CODE column has a card for "GitHub" (conv-201-xxx-calc-148...). The DELIVER column has a card for "Delivery Pipeline" (conv-201-xxx-calc-148...). Below these cards is another card for "Eclipse Orion Web IDE". At the bottom right of the main area, there is a "Feedback" button.

Figure 1-75 Toolchains: Select Delivery Pipeline

Wait until the build and deploy stages are completed ([Figure 1-76](#)). When they are completed, and with no errors, your application is ready to be tested.

The screenshot shows the IBM Bluemix DevOps interface for a delivery pipeline named "conv-201-xxx-calc-1486207022447 | Delivery Pipeline". The pipeline is divided into two main stages: "Build Stage" and "Deploy Stage".

**Build Stage:**

- STAGE PASSED:** Indicated by a green bar.
- LAST INPUT:** Last commit by Ahmed Azraq, 8m ago. Includes a link to "Edit the application logic to perform the ca...".
- JOB:** Build Passed 4m ago. Includes a link to "View logs and history".
- LAST EXECUTION RESULT:** Build 1. Includes a link to "View logs and history".

**Deploy Stage:**

- STAGE PASSED:** Indicated by a green bar.
- LAST INPUT:** Stage: Build Stage / Job: Build.
- JOB:** Build 1. Includes a link to "View logs and history".
- LAST EXECUTION RESULT:** conv-201-xxx-calc. Includes a link to "conv-201-xxx-calc.mybluemix.net". A green circular icon indicates the result is successful.

Figure 1-76 Delivery Pipeline: Build Stage and Deploy Stage

**Stop:** Wait until the build and deploy stages are completed before testing.

## 1.4 Testing the application

Follow these steps:

Open your application route (the URL to access your application) in a web browser with the following address, where xxx is the value that you added earlier to make your application name unique : <http://conv-201-xxx-calc.mybluemix.net/>

The screenshot shows a chatbot application interface. On the left, a purple sidebar contains a message from the bot: "Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?". On the right, the main window has a title bar "Watson understands" and a purple circular button with three dots (...). The main area displays the bot's internal state as a JSON object:

```
1 {
2 "intents": [],
3 "entities": [],
4 "input": {},
5 "output": {
6 "text": [
7 "Hi, I am VCal - a calculator you can chat with. I can help you"
8],
9 "nodes_visited": [
10 "Welcome"
11],
12 "log_messages": []
13 },
14 "context": {
15 "conversation_id": "b6da54ad-ca3c-4614-b5a7-83ed18818947",
16 "system": {
17 "dialog_stack": [
18 {
19 "dialog_node": "root"
20 }
21],
22 "dialog_turn_counter": 1,
23 "dialog_use_case": "calculator"
24 }
25 }
26 }
```

Figure 3-56 Calculator chatbot application

Test the addition and multiplication functionalities by chatting with the application

Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?

kindly add numbers

Kindly help me with the numbers that you want to add.

5 and 45

The result is : 50

multiply 5 and 6

The result is : 30

5 + 10 + 15

The result is : 30

6 \* 3

The result is : 18

User input

```
1 {
2 "input": {
3 "text": "6 * 3"
4 },
5 "context": {
6 "conversation_id": "b6da54ad-ca3c-4614-b5a7-83ed18818947",
7 "system": {
8 "dialog_stack": [
9 {
10 "dialog_node": "root"
11 }
12],
13 "dialog_turn_counter": 8,
14 "dialog_request_counter": 8,
15 "_node_output_map": {
16 "Welcome": [
17 0
18],
19 "node_3_1497615462744": [
20 0
21],
22 "node_2_1497615350325": [
23 0
24],
25 "node_4_1497763865025": [
26 0
27]
28 },
29 }
30 }
31 }
```

Try various scenarios and identify those for which the application fails to respond appropriately. Failing to respond correctly means that more training is needed. Training is performed by adding more user examples to the intents in the Calculator Conversation workspace.

Hi, I am VCal - a calculator you can chat with. I can help you with simple arithmetic such as addition and multiplication. What mathematical operation would you like to perform?

hi

Hi !! How may i help you today?

calculate numbers

I understand that you want to calculate something. I can help you by calculating addition and multiplication of numbers at this time. Which operation you want to perform?

add 2 and

Ooops. I found only 1 operand(number) and required at least 2 operand(number) to perform addition operation. Kindly provide other number/s.

3

The result is : 5

thanks

You are welcome.

what is your name

I am Cognitive Calculator, I could help you with addition or multiplication of numbers.

how to play football

Sorry!! I am unable to answer your query at this time as i am not trained on it. Kindly connect with help center. I can help you in performing addition and multiplication operations.

User input

```
1 {
2 "input": {
3 "text": "how to play football"
4 },
5 "context": {
6 "conversation_id": "d8f15d25-ebb1-40f9-aa7c-ec2a3432803f",
7 "system": {
8 "dialog_stack": [
9 {
10 "dialog_node": "root"
11 }
12],
13 "dialog_turn_counter": 8,
14 "dialog_request_counter": 8,
15 "_node_output_map": {
16 "Welcome": [
17 0
18],
19 "node_15_1497768845200": [
20 0
21],
22 "node_9_1497764933969": [
23 0
24],
25 "node_1_1497614922579": [
26 0
27],
28 "node_2_1497615350325": [
29 0
30],
31 "node_18_1497769245471": [
32 0,
33 1,
34 2,
35 0
36],
37 "node_16_1497768940321": [
38 0
39 1
40]
41 }
42 }
43 }
44 }
```

## 1.5 Quick deployment of application

Use the following steps to quickly deploy the full application:

- 1 Copy conversation service model file named “calculator\_model.json” from the Repository URL :  
[https://github.com/dakspate/calculator-bot/blob/master/resources/calculator\\_model.json](https://github.com/dakspate/calculator-bot/blob/master/resources/calculator_model.json)
- 2 Import the above file in your Calculator workspace into your newly created Conversation service.
- 3 Clone node.js application from below url :  
<https://github.com/dakspate/calculator-bot/>
- 4 Follow the steps in [“Configure the application” on page 63](#) to configure your application to point to the Calculator workspace.
- 5 Test the application as described in [“Testing the application” on page 74.](#)

## Lab Exercises

- 1) Add subtraction functionality

User can able to perform subtraction of numbers.

Changes needs to be done :

- 1.1) Add subtract entity value along with synonyms for @Operations entity
- 1.2) Add subtract node parallel to “@Operations:add” node
- 1.3) Check if user provided minimum 2 numbers to perform subtract operation
- 1.4) Set “operation” to “subtract” in context parameter
- 1.5) In node.js application calculate result
- 1.6) Set “result” context parameter with the value of calculated result

- 2) Ask for user confirmation to move ahead with other operation once result is displayed.

User : “add 3 and 5”.

Bot : “The result is : 8”.

Bot : “Is there any thing else I can help you with”.

**User :“yes”**

Bot : “I can help you by doing arithmetic operations on number like addition and multiplication.”

User : “add 3 and 5”.

Bot : “The result is : 8”.

Bot : “Is there any thing else I can help you with”.

**User :“no”**

Bot : “I hope I helped you today”

Changes needs to be done :

- 1.1) Add user consent entity @consent
- 1.2) Add 2 values for “@consent” entity
  - yes** with synonyms like ya,yup,y
  - no** with synonyms like nope,na,n
- 1.3) Add node after result display node
- 1.4) Check for user input and based on the user input “yes”/“no” display proper message