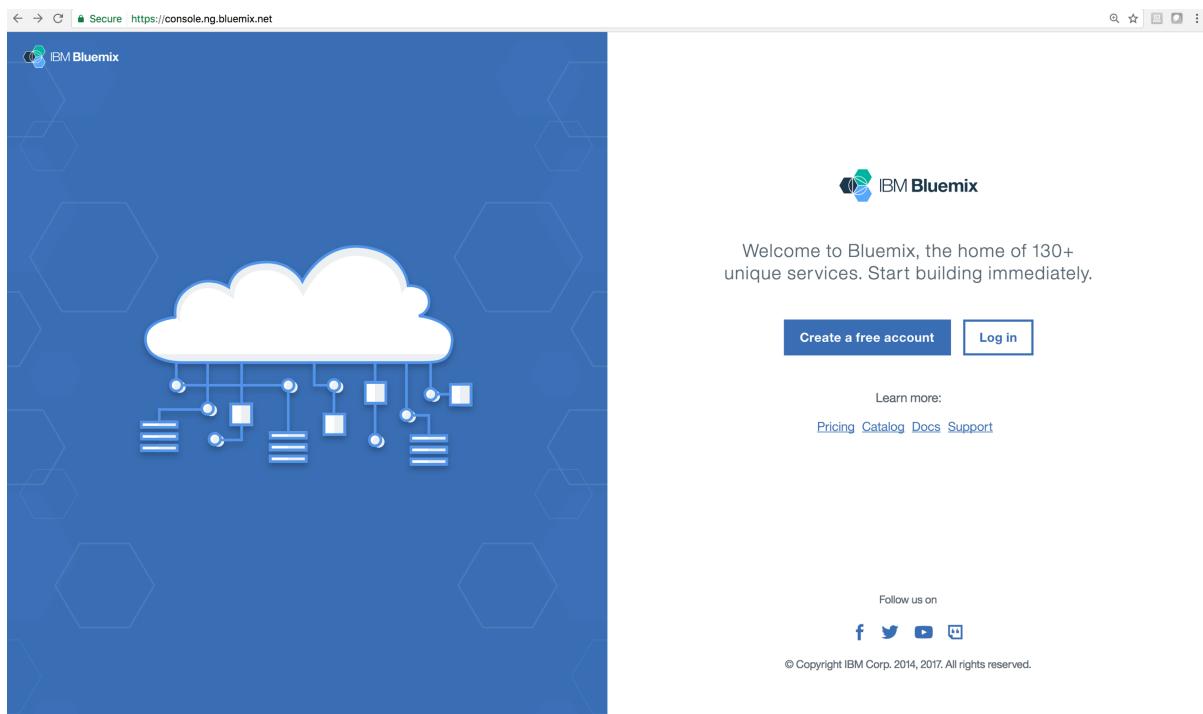


TUTORIAL: ChatBot with Watson Conversation

1) Bluemix Registration:

Go to <https://www.blueix.net> and click on “Create a free account”



Obs.: If you already have Bluemix registration jump to step 2.

A screenshot of the Bluemix sign-up form. On the left, there's promotional text: "Sign up for an IBMid and create your Bluemix account", "Try Bluemix free for 30 days", "Start building immediately.", "Production app? No problem.", and "We're here to help.". On the right, the sign-up form fields are shown with sample data: First Name (Sergio), Last Name (GAMA), Company (IBM), Country or Region (Brazil), Phone Number (11950526416), and Password (represented by a series of dots). Below the form are checkboxes for "Keep me informed of products, services, and offerings from IBM companies worldwide." and "By clicking Create Account, I accept the [Bluemix privacy policy](#) and [Bluemix terms](#)". At the bottom are "Create Account" and "Privacy - Terms" buttons.

Fill out the form, as the example above, and click on “Create Account”. A e-mail form confirmation will be send to you, check it out.

The screenshot shows a Gmail inbox with the following details:

- Header: Google, search bar, navigation icons, 1-2 of 2, settings.
- Toolbar: COMPOSE, inbox dropdown (Inbox (2)), primary, social, promotions.
- Inbox list:
 - The Bluemix Team - Action required: Confirm your Bluemix account
- Message preview: Action required: Confirm your Bluemix account - Hello Sergio, Thank you for signing up for IBM Bluemix! Your 30-day trial is fr 8:13 am

Open the e-mail sent, and click on the link to confirm as the below screenshot:

Action required: Confirm your Bluemix account

The Bluemix Team no-reply@bluemix.net via sendgrid.softlayer.com
to me 8:13 AM (1 minute ago)

Hello Sergio,

Thank you for signing up for IBM Bluemix!
Your **30-day trial is free**, with no credit card required.
You get access to 2 GB of runtime and container memory to run apps, unlimited IBM services and APIs, and complimentary support.

Confirm Account

By confirming your account, you accept the [Bluemix Terms of Use](#)
If you have problems logging in, go to [Bluemix Support](#) to contact us.
We look forward to seeing what you create!

—
Thank you,
IBM Bluemix
[bluemix.net](#)

Click on the “Confirm Account”.



Success!

You successfully signed up for a Bluemix account and it is now activated.

Click the link to log in.

[Log in](#)

Your account has been confirmed successfully. Click in the login and insert your credentials. In case you are a IBMer, the system will send you to IBM system to login with IBM Credentials.

A screenshot of the IBM Bluemix web interface. At the top, there's a dark header bar with the IBM logo, "US South" region indicator, and navigation links for "Catalog", "Support", and "Manage". Below the header, the main content area has a title "Create organization" with a progress bar showing steps 1, 2, and 3. A yellow warning message box says: "⚠ You don't have access to this organization yet. Try again later." To the right of the progress bar, there's a note: "Organization and space, click here to learn more." The main body of the page contains instructions: "Before you start using Bluemix, you need to set up your environment." and "To start, name your first organization. Think of an org as a project or team that shares resources, such as apps, databases, and other services. Orgs exist in geographic regions, so decide where you'd like to put your first one." Below these instructions, there are dropdown menus for "Region" (set to "US South") and "Organization Name" (set to "Bluemix Man"), and a prominent blue "Create" button. At the bottom of the page, there's a section titled "NEED SOME SUGGESTIONS? TRY THESE" with two suggestions: "brbluemixman" and "brbluemixman@gmail.com". At the very bottom, there are links for "LOG OUT | SUPPORT".

As soon as you login in the system you have to create an organization name, for instance your name, or the name of company, etc, and click "Create"

The screenshot shows the 'Create space' step in the IBM Bluemix Apps interface. At the top, there's a navigation bar with 'IBM Bluemix Apps'. Below it, a modal window titled 'Create space' is open, showing a progress bar with steps 1, 2, and 3. Step 2 is highlighted. The main content area says 'Now, let's get you set up with a space.' and 'Spaces help you manage access and permissions for a set of resources, and map nicely to development stages like dev, test, and prod. Name your first space now—you can add more spaces later.' A text input field shows 'Org name: Bluemix Man' and a 'Space name' dropdown with 'dev' selected. A 'Create' button is visible. Below the input fields, there's a section titled 'NEED SOME SUGGESTIONS? TRY THESE' with buttons for 'dev', 'test', and 'prod'. At the bottom of the modal are 'LOG OUT | SUPPORT' links.

The second step before you can start your first application or consume a service on Bluemix you have to create a space, a space is one way to organize your projects, and configure team work according to area or environment, such as “Dev”, “Homologation”, “Production”, with different teams with different roles. For now let’s create a space, for example “Dev”, and click “Create”.

The screenshot shows the 'Summary' step in the IBM Bluemix Apps interface. At the top, there's a navigation bar with 'IBM Bluemix Apps'. Below it, a modal window titled 'Summary' is open, showing a progress bar with steps 1, 2, and 3. Step 3 is highlighted. The main content area says 'Good to Go!' and 'You're up and running with your first org and space. Are you ready to get started with Bluemix?'. It displays the 'Org name: Bluemix Man' and 'Space name: dev'. A large blue 'I'm Ready' button is prominent at the bottom. At the bottom of the modal are 'LOG OUT | SUPPORT' links.

That's it! You're ready to go! Click on “I'm ready” and let's start our first app.

IBM Bluemix Apps

29 Trial Days Remaining ▾

IBM | US South : Bluemix Man : dev

Catalog Support Manage

Docs

Apps

You don't have any apps yet. Get started with one of the options that follow, or go to the catalog to create an app.

Create app

Create a Cloud Foundry app

Order a monthly Bare Metal Server

Create and run event-driven apps

That's your first screen, it's your bluemix dashboard. For now, click on "Catalog".

2) Create a Watson Conversation service and your bot

All Categories

Infrastructure

Compute

Storage

Network

Security

Apps

Bollerplates

Cloud Foundry Apps

Containers

OpenWhisk

Mobile

Services

Data & Analytics

Watson

Internet of Things

APIs

Network

Storage

Security

DevOps

Application Services

Search

Filter

Conversation

Discovery

Document Conversion

Language Translator

Natural Language Classifier

Natural Language Understanding

Personality Insights

Retrieve and Rank

Speech to Text

Text to Speech

Tone Analyzer

Visual Recognition

Go to Watson group (left side), and click on Conversation.

IBM | US South : Bluemix Man : dev 

Catalog Support Manage

[View all](#)

Conversation

Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots that can integrate and communicate on any channel or device. Train Watson Conversation service through an easy-to-use web application, designed so you can quickly build natural conversation flows between your apps and users, and deploy scalable, cost effective solutions.

Service name: Conversation-aboutme

Credential name: Credentials-1

Images

Click an image to enlarge and view screen captures, slides, or videos. Screen caps show the user interface for the service after it has been provisioned.

Connect to: Leave unbound

[View Docs](#)

Need Help? [Contact Bluemix Sales](#)

Estimate Monthly Cost [Cost Calculator](#)

[Create](#)



As most of all service on Bluemix in this screen you have a description about the service, business model and a link to detailed documentation. Change the last part of the name of service to “-aboutme”. It’s not necessary, but it will help us find our service easily.

29 Trial Days Remaining 

IBM | US South : Bluemix Man : dev 

Catalog Support Manage

Watson / Conversation-aboutme

Conversation-aboutme



Conversation

Add a natural language interface to your application to automate interactions with your end users. Common applications include virtual agents and chat bots that can integrate and communicate on any channel or device.

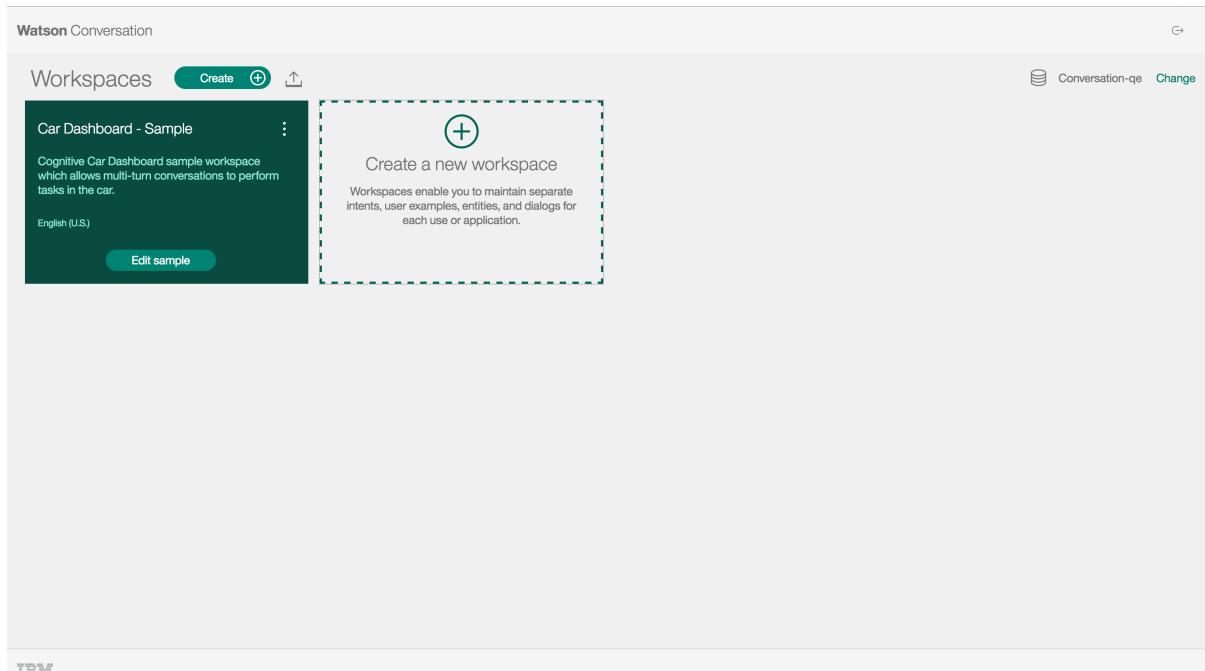
[Launch tool](#) 

Developer resources:

- Documentation
- Demo

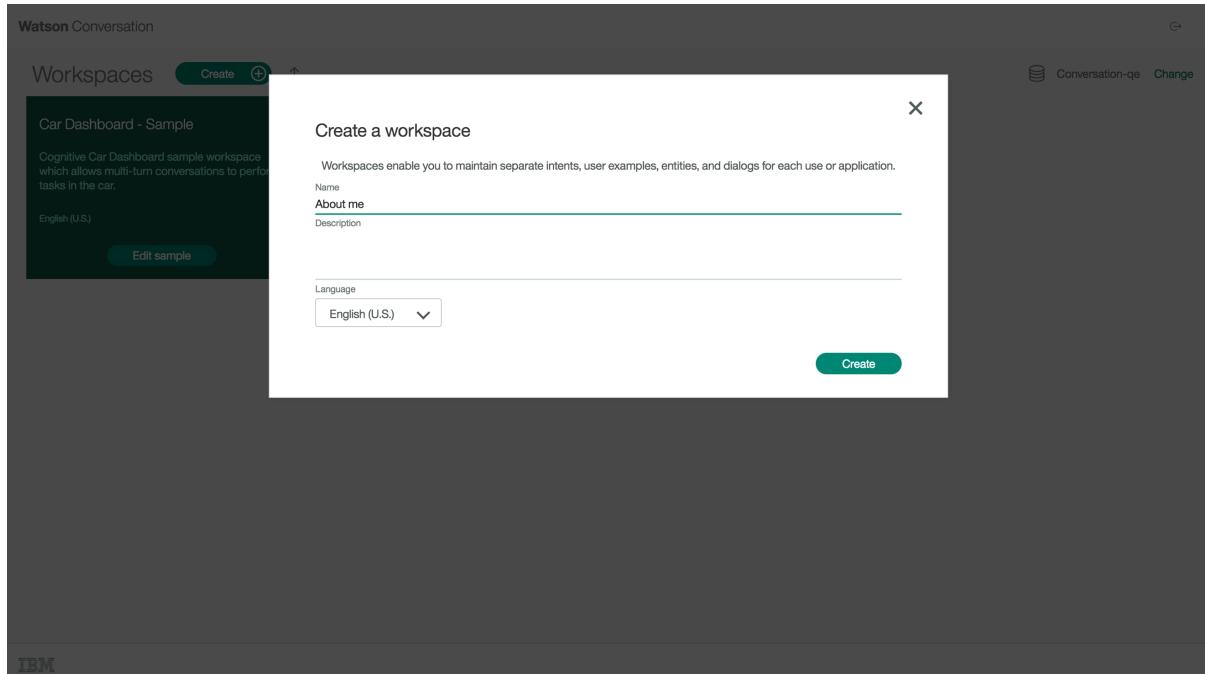
Getting started

Here you go!! Now you have a Watson Conversation for you. This is a service panel, where there is the service credentials and a button to open a tool. So, let's open our tool and teach Watson about our bout. Click on “Launch Tool” button.



IBM

Now you need to create a workspace, which is like a knowledge domain, or the space for a specific area which is intended to develop a Bot, for instance “Support area” or “Customer Attendance Service”, etc. In this case as the proposal is to develop a bot to answer about ourselves. Click on “Create a Workspace. By the way, the service brings a bot example, “Car Dashboard”.



IBM

Name it “About me”, or as you prefer, and choose the language, in this case “English”. Click on “Create”.

No intents yet.

An intent is the goal or purpose of the user's input. Adding examples to intents helps your bot understand different ways in which people would say them.

Create new

Import

Inside your workspace, you need to create your intents. Intents identify according to the intention of the user target, for instance, if we were creating a "disk Pizza", one intention could be "Do a order" or "Ask a pizza", and for instance we have given at least 5 phrases to Watson learned the standard, so an intention has a group of possibilities of user interact with your bot, which will have a response for this group. Let's click on "Create New".

Intent name
#whoami

User example
Add a user example...

What is your name?

Who are you?

Tell me who you are

Tell me your name

I'd like to know your name

An intent is the goal or purpose of the user's input. List several ways that the question or statement might be entered, to help your bot recognize the intent.

Learn more about intents

Questions? Visit forum

In the column "Intent name", insert "whoami", the "#" the system includes automatically, again this name is a classification for a group of possibilities that users normally use to interact when they have this intention, ok? As mentioned previously, include at least 5 possibilities, could be more, but the recommendation is about 10. Include the examples above, or as you prefer. Have in mind that it is not necessary to include all variations, as Watson works with taxonomy and understands the standard. Click on the first line, type the first

example, and type enter to jump the next, do the same until you reach the last one. Then click “done” button to conclude this intention.

Watson Conversation / About me / Build

Intents Entities Dialog

Create new + ↑

1 intents Sort by: Newest ▼

> #whoami
I'd like to know your name 5

Click on “Create new” to create the next intention as the following:

Watson Conversation / About me / Build

Intents Entities Dialog

Intent name #greetings

Done -

User example
Add a user example... +

Hi -
Hello -
Whatsup! -
Hey -
Hey there -

An intent is the goal or purpose of the user's input. List several ways that the question or statement might be entered, to help your bot recognize the intent.

[Learn more about intents](#)

Questions? [Visit forum](#)

Don't forget to click on “Done” when you finished. Go to the next one.

Watson Conversation / About me / Build

Intents Entities Dialog

Create new + ↑

2 intents Sort by: Newest ▼

> #greetings
Hello 5

> #whoami
What is your name? 5

Click on “Create new” and create the next one as the following:

Watson Conversation / About me / Build

Intents Entities Dialog

Intent name
#goodbye

User example
Add a user example...

Bye (-)

Good bye (-)

See ya (-)

Bye bye (-)

See you later (-)

I have to go (-)

Done (+) (trash)

Click “done” when you finished.

Watson Conversation / About me / Build

Intents Entities Dialog

Create new (+) (up)

3 intents Sort by: Newest (down)

> #goodbye 6
Bye

> #greetings 5
Hi

> #whoami 5
What is your name?

Click on “Create new” to create an intention to respond about what you do.

Watson Conversation / About me / Build

Intents Entities Dialog

Intent name
#whatdoyoudo

User example
Add a user example...

What do you do? (-)

What's your occupation? (-)

What do you work on? (-)

What do you work with? (-)

What type of work you do? (-)

Done (+) (trash)

Click on “Done” and let’s go to create the last one to respond about your hobby, so what do you like to do in your free time.

Watson Conversation / About me / Build

Intents Entities Dialog

Intent name
#hobby

User example
Add a user example... +

What do you like to do? -

What is your hobby? -

What do you like to do in your free time? -

What do you do in your free time? -

Do you like to play soccer? -

Do you play tennis? -

Done - -

Click “Done” and let’s go to entities. Click on “Entities” as showed in the screen above.

Watson Conversation / About me / Build

Intents Entities Dialog

My entities System entities

No entities yet

An entity is a portion of the user's input that you can use to provide a different response to a particular intent. Adding values and synonyms to entities helps your bot learn and understand important details that your users mention.

Create new + Use system entities

Import

Now you are going to create entities. An *entity* represents a class of object or a data type that is relevant to a user's purpose. By recognizing the entities that are mentioned in the user's input, the Conversation service can choose the specific actions to take to fulfill an intent. Click on “Create new” and follow the example below.

The screenshot shows the Watson Conversation interface in the 'Build' section, specifically the 'Entities' tab. A new entity named '@sport' has been created. The 'Fuzzy Matching' feature is turned off. Under the 'Value' section, there are four listed items: Soccer, Bycicle, Football, and Tennis. Under the 'Synonyms' section, there is one item: Bike. A 'Done' button is visible at the top right.

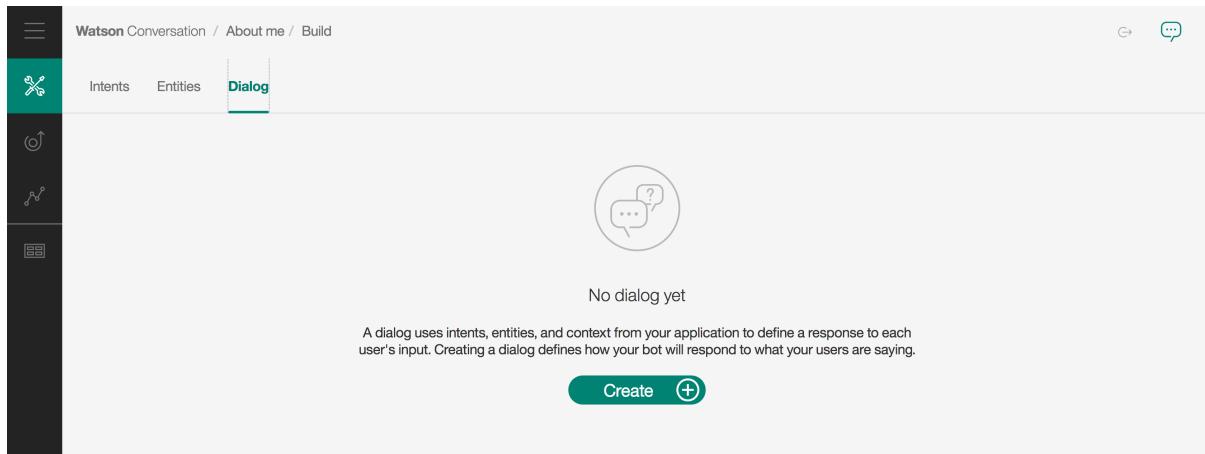
In this example you are creating a entity to be specific for responses about what you like to do in your free time, for instance: “Do you like play soccer? “ on this case you intention and entities, but in case the user ask: “What do you like to do?”, it’s more generic, so your answer must be more generic as well. Attention: For this lab keep Fuzzy Matching “off”.

Entity has a name and a group of possibilities, and for each possibility you can add synonyms, once Watson read the correct word to captured entity.

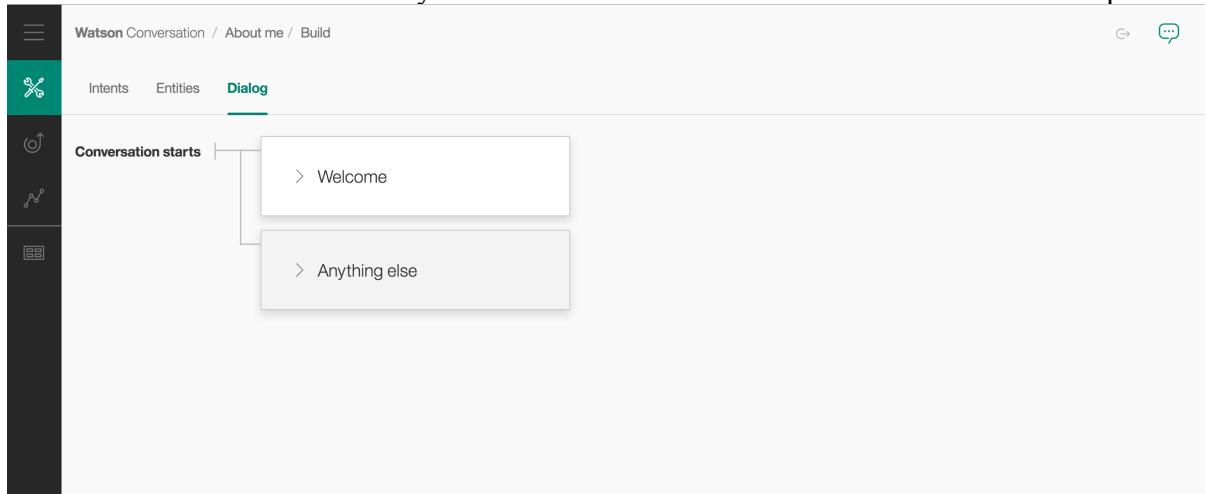
Type “sport” on field Entity (@ is included automatically by the system), then include in the list above the values above, as the listed above, and click “Done”.

The screenshot shows the 'My entities' section of the Watson Conversation interface. An arrow points from the previous screenshot's 'Done' button here. The entity '@sports' is listed, showing its synonyms: Soccer, Bycicle, Tennis, and Football. A 'Create new' button is visible at the bottom left.

That is it for entities. Now let's create our dialog. Click on “Dialog” as showed above.

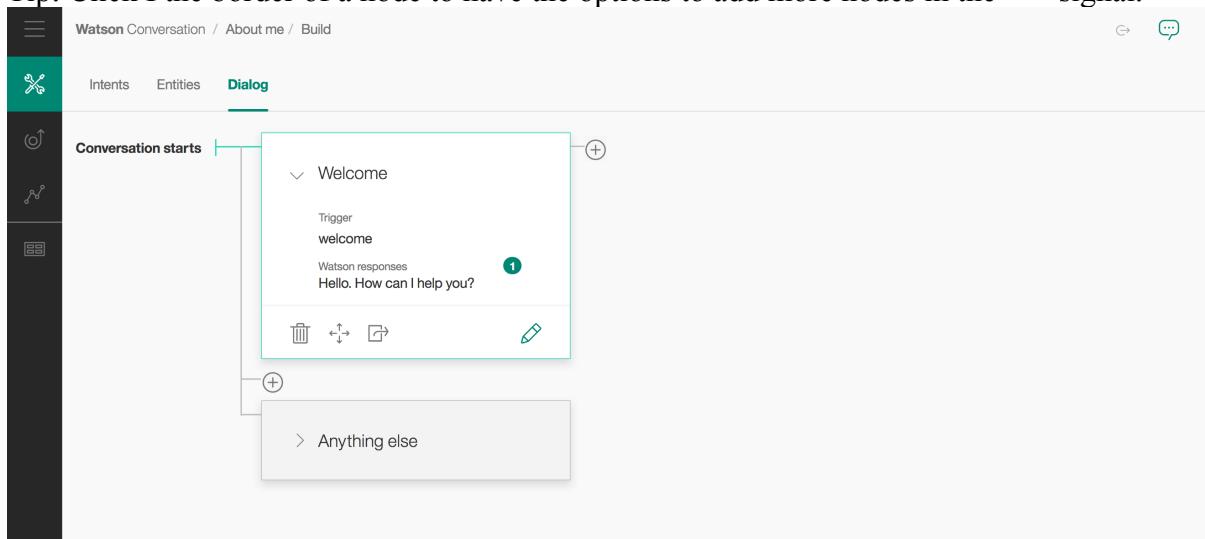


At this point you are prepared to complete your bot, creating a dialog flow, which will be based on intentions and entities you created. Click on “Create” and follow the next steps.



The system generates automatically 2 nodes, “Welcome” that is the first node executed by the service in the first requisition, such as “Hello, How can I help you?”, and “Anything else”, which is the last node verified condition, that means no one the flow dialog no one of the node of this flow has attended the conditions. Let’s include a node to if the intention of the user is greetings, in this case you will create a specific response for that.

Tip: Click on the border of a node to have the options to add more nodes in the “+” signal.



Click in the “+” signal below “Welcome” node.

Include in the first field the name for this node (it's not mandatory). In the field if, where you can validate a condition based on Intention, or Entity, or context variable, or a combination of them. In this case type “#”, wait the system shows the list of intentions, so click on “greetings”, Go to Responses and add the following response.

Click in the “x” signal on upside right to close this node, and click in “+” below of greetings node.

Watson Conversation / About me / Build

Dialog

Hello. How can I help you?

- Greetings
 - Trigger: #greetings

Watson responds: Hi! What do you like to know about me?
- Goodbye
 - Trigger: No condition yet

Watson responds: No response yet
- Anything else

Goodbye

Trigger

if #

- #goodbye
- #greetings
- #hobby
- #whatdoyoudo
- #whoami
- # (create new condition)

Enter a response...

Add another response

After response Watson will Wait for user input

Name it as Goodbye, and include #goodbye in the condition, so go to the response and include the response, as the following.

Watson Conversation / About me / Build

Dialog

Hello. How can I help you?

- Greetings
 - Trigger: #greetings

Watson responds: Hi! What do you like to know about me?
- Goodbye
 - Trigger: #goodbye

Watson responds: Thanks for camel! You're always very welcome!
- Anything else

Goodbye

Trigger

if #goodbye

Responses

1. Thanks for camel! You're always very welcome!

Add a variation to this response

Add another response

After response Watson will Wait for user input

Once again click on “+” above Goodbye node.

Include the node to respond to the `#whatdoyoudo` intention, as showing above. And create another one for `#whoami` as the screenshot below:

Create another node for `#hobby` intention as the following, but this time keep the response empty, because you will use another technique to be specific, once you created entities for this intention.

Watson Conversation / About me / Build

Intents Entities Dialog

Hobby

Trigger `#hobby`
Watson responds `No response yet`

Responses

`+ Add response condition`
Enter a response...

After response Watson will `Wait for user input`

Click on the signal “+” in the tight side of node “Hobby”.

Watson Conversation / About me / Build

Intents Entities Dialog

Hobby

Trigger `#hobby`
Watson responds `No response yet`

Soccer

Trigger `@sports:Soccer`
Watson responds `I like soccer, but in fact I prefer watch it than play.`

Responses

`+ Add response condition`
1. `I like soccer, but in fact I prefer watch it than play.`
Add a variation to this response

After response Watson will `Wait for user input`

Name this node as Soccer, and add `@sports:Soccer` in the condition, and the response, which is specific for soccer. Feel free to change the response according to you. Now we go include other node below soccer (click on “+” signal below soccer and so on), to treat other entities sports as the following:

For Bike:

Watson Conversation / About me / Build

Dialog

```

graph TD
    Hobby[Hobby] --> Soccer[Soccer]
    Hobby --> Bike[Bike]
    Hobby --> Anything[Anything else]
    Soccer --> Response1[Response 1]
    Bike --> Response2[Response 2]
  
```

A dialog tree structure:

- Hobby** intent (Trigger: #hobby) branches into three nodes:
 - Soccer** node (Trigger: @sports:Soccer):
 - Watson responses: I like soccer, but in fact I prefer watch it than ...
 - Bike** node (Trigger: @sports:Bycicle):
 - Watson responses: I have the habit to ride my bike 3 times a week. I...
 - Anything else** node

Learn more about dialog

<https://www.ibmwatconversation.com/us-south/9d92cca3-1366-48c9-90...1ec2/worksapces/755148b0-f3ec-45f9-9975-d8b6add77522/build/dialog#>

Questions? [Visit forum](#)

For Tennis:

Watson Conversation / About me / Build

Dialog

```

graph TD
    Bike[Bike] --> Tennis[Tennis]
    Anything[Anything else] --> Tennis
  
```

A dialog tree structure:

- Bike** intent (Trigger: @sports:Bycicle) branches into two nodes:
 - Tennis** node (Trigger: @sports:Tennis):
 - Responses: I do not know play tennis, but I'd love to learn.
 - Anything else** node

<https://www.ibmwatconversation.com/us-south/9d92cca3-1366-48c9-9048-ac6ec631ec2/worksapces/755148b0-f3ec-45f9-9975-d8b6add77522/build/dialog#>

For football:

The screenshot shows the Watson Conversation builder interface. On the left, there are tabs for Intents, Entities, and Dialog. The Dialog tab is selected. In the main area, there is a tree view of nodes. At the top level, there is a node labeled '#hobby' with the Watson response 'No response yet'. Three arrows point down to child nodes: 'Soccer', 'Bike', and 'Tennis'. Below 'Tennis', there is a new node labeled 'Untitled Node' with a green border. Inside this node, the trigger is '@sport:Football' and the Watson response is 'No way! I think football is very aggressive. It's ...'. Another 'Untitled Node' is shown below it with the trigger 'anything_else'. On the right side, there are sections for naming the node, setting triggers (with one entry 'if @sport:Football'), and defining responses. A response is listed: '1. No way! I think football is very aggressive. It's not for me.' There is also a section for adding more responses.

In case the user do not mentioned no one of the entities above, we need to have a generic response for that, so you have to include a “anything_else” method, as the following:

This screenshot shows the same Watson Conversation builder interface as the previous one, but with a different configuration. The root node '#hobby' still has branches for Soccer, Bike, and Tennis. However, the node under Tennis now has a different response: 'I do not know play tennis, b to learn.'. Below this, there is another 'Untitled Node' with the trigger 'anything_else' and the Watson response 'When is possible, normally when I'm not travelling ...'. The rest of the interface, including the trigger configuration and response sections, remains similar to the first screenshot.

Ok! You already have all the nodes you need, but you have to inform to Watson to entry in the entities condition when the #hobby condition ne true. Go back to the #hobby node and click on icon “Jump to” (third one in the bottom of node), as showed below:

A dialog uses intents, entities, and context from your application to return a response to each user's input. Create a dialog branch for each intent that you define. In each node (box), enter a condition based on the input, such as the name of an intent. Then enter the response that your bot should make when that condition is true.

[Learn more about dialog](#)

https://www.ibmwatsonconversation.com/us-south/9d92cca3-1366-48c9-9048-ac6ac631ece2/workspaces/755148b0-f3ec-45f9-9975-d8b6add77522/build/dialog#

Questions? [Visit forum](#)

The system will show the following information in a green bar, that means it is waiting you indicate for where it have to jump. Click on “Soccer” node, and the system will show 2 options as the following:

Select a destination node (origin:) [Cancel](#)

[Go to condition](#)

[Go to response](#)

click on “Go to condition” button. The screen will looks like as the following:

A dialog uses intents, entities, and context from your application to return a response to each user's input. Create a dialog branch for each intent that you define. In each node (box), enter a condition based on the input, such as the name of an intent. Then enter the response that your bot should make when that condition is true.

[Learn more about dialog](#)

Questions? [Visit forum](#)

That's it! Now let's test it and learn more and more. Click on "..." button in the corner in the right side above. Talk to it!

Try it out

Hello. How can I help you?
hey there!!
#greetings

Hil. What do you like to know about me?
who are u?
#whoami

Thanks for ask. I'm Sergio Gamal
hum.. What do u like to do?
#hobby

When is possible, normally when I'm not travelling I go to the gym and ride my bike 3 times a week.
do u like football?
#hobby
@sport:Football

No way! I think football is very aggressive. It's not for me.

Enter something to test your bot
Use the up key for most recent

Congratulations!! Now we can plug this service in any channel as you want, such as Facebook messenger, Telegram, slack, etc.

3) Plug your bot to a web page.

3.1 Go to GitHub page, case you do not have a account create one.

<https://www.github.com>

3.2 Create a app on Bluemix and clone a github source code.

Back to Bluemix catalog and create “Continuous Delivery” service, as the following:

The screenshot shows the IBM Bluemix Catalog interface. On the left, there's a sidebar with categories like All Categories, Infrastructure, Apps, Services, and DevOps (which is currently selected). In the main area, there's a search bar and a filter button. Below the search bar, there are several service cards. One card, "Continuous Delivery", is highlighted with a red border. Other visible services include Active Deploy, Auto-Scaling, Availability Monitoring, DevOps Insights, Globalization Pipeline, IBM Alert Notification, Monitoring and Analytics, BlazeMeter, HipTest, jKool, and Load Impact. Each service card has a small icon, a name, a brief description, and a status indicator (e.g., IBM, Deprecated, Beta, Third Party).

Click on “Continuous Delivery service”and go the next page.

This screenshot shows the "Continuous Delivery" service creation page. At the top, it says "Continuous Delivery". Below that, there's a "Service name:" field containing "Continuous Delivery-DBG", which is circled in red. Underneath, there's a "Features" section with several bullet points: "Create an integrated devops toolchain.", "Edit your code from anywhere.", "Git Repos and Issue Tracking.", "Deliver continuously using an automated pipeline.", and "Improve quality through Insights.". At the bottom of the page, there are two buttons: "Need Help? Contact Bluemix Sales" and "Estimate Monthly Cost Cost Calculator". A large blue "Create" button is at the very bottom right, also circled in red.

Change the name as showed above and click on “Create” button.

IBM Bluemix DevOps

Continuous Delivery-DBG

Get started with Continuous Delivery

Start here

Start with a pipeline

You can easily start deploying your app with an automated build and delivery pipeline. Later, you can quickly add more tools to your toolchain.

Start from a toolchain template

Create a toolchain that integrates tools for planning, developing, testing, deploying, and managing your apps. You can always add or remove tools from your toolchain.

Already have toolchains? [View your toolchains](#).

Click on “Start here” over “Start from toolchain template”

Toolchains

Create a Toolchain

A toolchain is a set of integrated tools for development, deployment, monitoring, and more. Select a toolchain template. After you create a toolchain, you can add more tool integrations to it as needed.

[Create a toolchain from an application](#)

[Learn more about toolchains](#)

Continuous Delivery Templates

- Garage Method Cloud-native Tutorial toolchain
- Microservices toolchain with DevOps Insights
- Microservices toolchain with DevOps Insights (v2)
- Secure container toolchain
- Simple Cloud Foundry toolchain (v2) 
- Simple Cloud Foundry toolchain with DevOps Insights
- Simple container toolchain

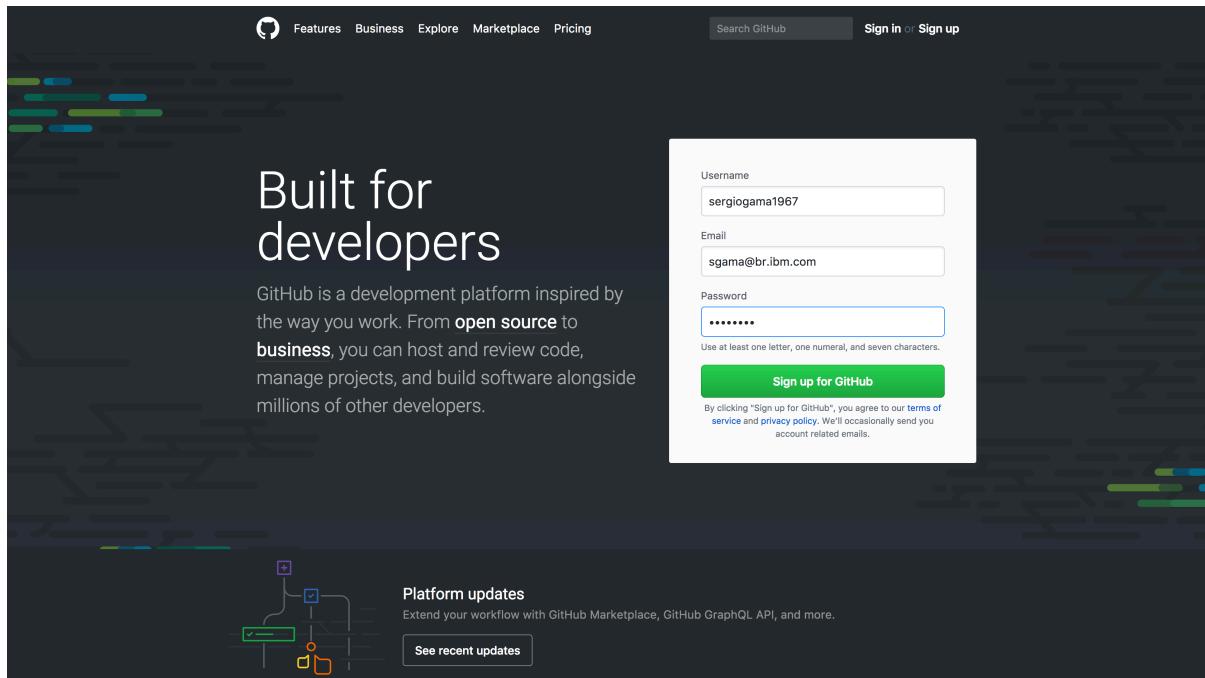
DevOps Insights Templates

- Delivery Insights with IBM UrbanCode Deploy
- Deployment Risk Analytics with GitHub and Jenkins
- Developer Insights and Team Dynamics with GitHub and JIRA

Other Templates

Ask us a Question

Click on “Simple Cloud Foundry toolchain” service. Open a new browser tab and open <https://www.github.com>, and create an account if you do not have.



Fill out the fields and click on “Sign up for GitHub” and follow the steps to create your account. As soon as you have created your account go to the URL:

<https://www.github.com/sergiogama/ibm-dbg-chat>

Click on “Fork” up right in the screen as the following:

This screenshot shows a GitHub repository page. At the top, it displays the repository name "sergiogama / ibm-dbg-chat". Below the name are buttons for "Watch" (with 1 follower), "Star" (0 stars), and "Fork" (0 forks, circled in red). The "Fork" button is highlighted with a red circle. The page then lists repository statistics: 12 commits, 1 branch, 0 releases, 1 contributor, and Apache-2.0 license. It shows a commit history from "sergiogama" with a timestamp of "7 days ago". The commit details include files like config, conversation, public, routes, views, LICENSE, README.md, app.js, manifest.yml, and package.json. Below the commit history is a section titled "Chatbot with Watson Conversation in Node.js" which contains instructions for connecting to a Watson Conversation service.

Now you have a copy of the web front end for your chatbot, as the following:

Created for toolchain: <https://console.bluemix.net/devops/toolchains/1605830c-f7a3-4259-b355-7e46c18bd16b>

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with sergiogama:master. Latest commit 99a@3be 7 days ago

sergiogama committed on GitHub Add files via upload

File	Action	Time
config	Update bot.js	7 days ago
conversation	Add files via upload	7 days ago
public	Add files via upload	7 days ago
routes	Add files via upload	7 days ago
views	Update chat.html	7 days ago
LICENSE	Add files via upload	7 days ago
README.md	Update README.md	7 days ago
app.js	Add files via upload	7 days ago
manifest.yml	Update manifest.yml	7 days ago
package.json	Add files via upload	7 days ago

README.md

Chatbot with Watson Conversation in Node.js

Copy URL and Back tp bluemix browser tab.

23 Trial Days Remaining | IBM | US South : Bluemix Man : dev | Catalog | Support | Manage

Organization: Bluemix Man | Toolchain Name: simple-toolchain-20170607143012290

Tool Integrations:

- GitHub
- Eclipse Orion Web IDE
- Delivery Pipeline

Store your source code in a new or existing repository on GitHub.com and engage in social coding through wikis, issue tracking, and pull requests.

Repository type: Clone

Clone the repository that is specified in the Source repository URL field.

New repository name: simple-toolchain-20170607143012290

Source repository URL: <https://github.com/sergiogama1967/ibm-dbg-chat>

Enable GitHub Issues

Track deployment of code changes

Create

Change the name of “Touchain Name” and paste the URL copied previously on “Source repository URL” field, and click on “Create” Button.

Click on “Eclipse Orion Web IDE”. Open the folder tree in left side and open bot.js file, as the following:

```

10  * @requires app.js
11  *
12  */
13 var watson = require('watson-developer-cloud');
14 var CONVERSATION_NAME = "Conversation-Demo"; // conversation name goes here.
15 var fs = require('fs');
16 // load local VCAP configuration
17 var appEnv = null;
18 var conversationWorkspace, conversation;
19
20 // =====
21 // CREATE THE SERVICE WRAPPER =====
22 // =====
23 // Create the service wrapper
24 conversation = watson.conversation({
25   url: "https://gateway.watsonplatform.net/conversation/api"
26   , username: "<username>" // Replace <username>, including "<" and ">"
27   , password: "<password>" // Replace <password>, including "<" and ">"
28   , version_date: '2017-04-10'
29   , version: 'v1'
30 });
31 // check if the workspace ID is specified in the environment
32 conversationWorkspace = "<workspace_id>"; // Replace <workspace_id>, including "<" and ">"
33 // if not, look it up by name or create one
34 // Allow clients to interact
35
36 var chatbot = {
37   sendMessage: function (req, callback) {
38     // var owner = req.user.username;
39     buildContextToObject(req, function (err, params) {
40       if (err) {
41         console.log("Error in building the parameters object: ", err);
42         return callback(err);
43       }
44       if (params.message) {

```

Now let's change the code to insert your Watson Conversation credentials, in lines 26, 27 and 32, in the right side. Back to browser tab of your conversation, and click on menu icon in up left side, as the following:

The screenshot shows the Watson Conversation interface. On the left, a sidebar has options for Build, Deploy, Improve, and Back to workspaces. The main area is titled 'Dialog' and contains two dialog branches:

- Welcome**: Triggered by 'welcome'. Watson responds with 'Hello. How can I help you?'. A note indicates it's a branch for the 'trigger welcome' intent.
- Greetings**: Triggered by '#greetings'. Watson responds with 'Hi! What do you like to know about me?'. A note indicates it's a branch for the 'trigger #greetings' intent.

On the right, a 'Try it out' panel shows a conversation with the bot:

- Hi
- #greetings
- Hi! What do you like to know about me?
- who are you?
- #whoami
- Thanks for ask. I'm Sergio Gama!
- What do you do?
- #whatdoyoudo
- I'm Developer Advocate at IBM, Leader in Latin America. My job is disseminate IBM Cloud Technology, including Watson and IoT.
- Do you like soccer?
- #hobby
- @sport:Soccer

A note at the bottom left says: 'A dialog uses intents, entities, and context from your application to return a response to each user's input. Create a dialog branch for each intent that you define. In each node (box), enter a condition based on the input, such as the name of an intent. Then enter the response that your bot should make when that condition is true.'

[Learn more about dialog](#)

Questions? [Visit forum](#)

<https://www.ibmwatsonconversation.com/us-south/b9d92cca3-1366-48c9-9048-je6ac631ece2/workspaces/755148b0-f3ec-45f9-9975-d8b6add77522/deploy>

Click on “Deploy” option. In the next screen you have the information need as the following:

The screenshot shows the 'Watson Conversation / About me / Deploy' screen. It includes the following sections:

- Workspace Details**: Shows 'Workspace name' as 'About me', 'Workspace ID' as '755148b0-f3ec-45f9-9975-d8b6add77522', and 'Workspace URL' as 'https://gateway.watsonplatform.net/conversation/api/workspaces/755148b0-f3ec-45f9-9975-d8b6add77522/message/'.
- Service Credentials**: Shows 'Service name' as 'Conversation-qe' and 'Credential name' as 'Credentials-1'. It displays 'Username' as 'f48d380f-7d77-49c6-9080-20c2df293650' and 'Password' as '6gAvZSNQupyx'.
- Deployment Options**: Three options are listed:
 - Publish with Botkit**: Using the Botkit framework, you can publish your work to Slack, Facebook, Messenger, or Twilio. [Learn how](#)
 - Build with a sample app**: We provide a number of sample applications for different use cases. Find one that matches your goal and use it as a starting point for building your own application. [Learn how](#)
 - Connect to your app**: If you are building an application that will use the Conversation service, learn how to integrate it. [Learn how](#)

Copy the credentials and workspace_id and back to DevOps browser tab. Change the code as the following:

```

10  * @requires app.js
11  *
12  */
13 var watson = require('watson-developer-cloud');
14 var CONVERSATION_NAME = "Conversation-Demo"; // conversation name goes here.
15 var fs = require('fs');
16 // load local VCAP configuration
17 var appEnv = null;
18 var conversationWorkspace, conversation;
19
20 // =====
21 // CREATE THE SERVICE WRAPPER =====
22 // =====
23 // Create the service wrapper
24 conversation = watson.conversation({
25   url: "https://gateway.watsonplatform.net/conversation/api"
26   , username: "48d380f-7d77-49c6-9080-20c2df293650" // Replace <username>, including "<" and ">"
27   , password: "6gAvZSNQupyxx" // Replace <password>, including "<" and ">"
28   , version_date: '2017-04-10'
29   , version: 'v1'
30 });
31 // check if the workspace ID is specified in the environment
32 conversationWorkspace = "755148b0-f3ec-45f9-9975-d8b6add77522"; // Replace <workspace_id>, including "<" and
33 // if not, look it up by name or create one
34 // Allow clients to interact
35
36 var chatbot = {
37   sendMessage: function (req, callback) {
38     // var owner = req.user.username;
39     buildContextObject(req, function (err, params) {
40       if (err) {
41         console.log("Error in building the parameters object: ", err);
42         return callback(err);
43     }
44     if (params.message) {

```

Now, open manifest.yml file in the left side pane, and change the host name as the following:

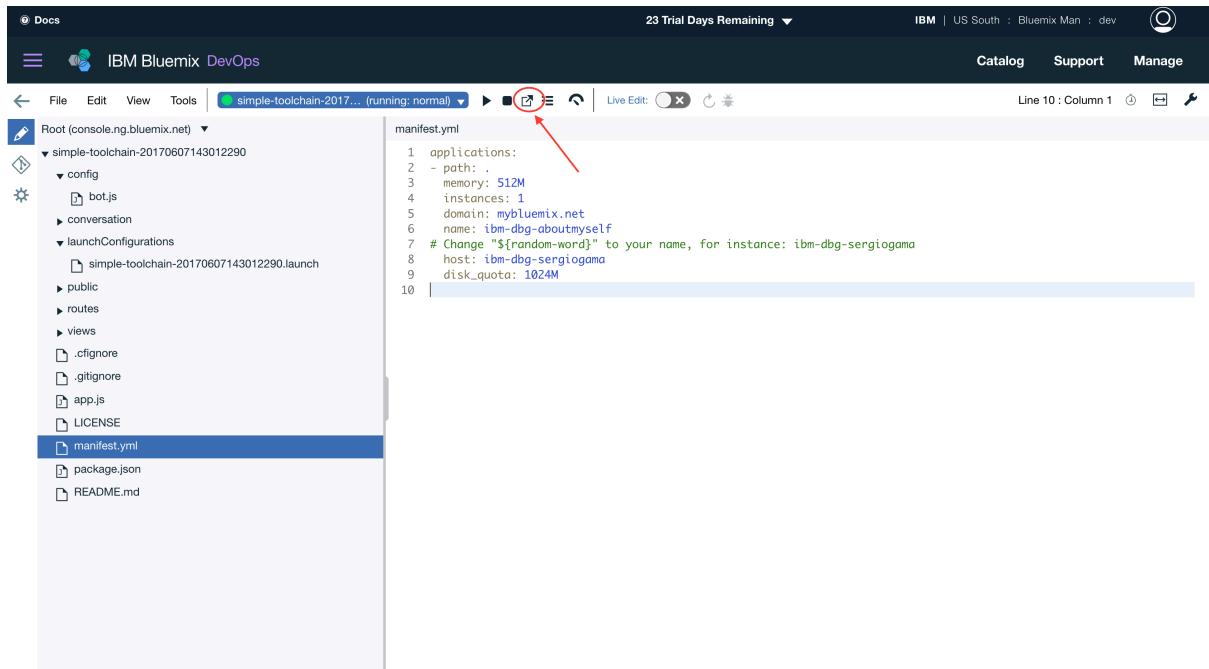
Tip: change to your name, but pay attention as this host name will be part of a public domain. In the example below will be <https://ibm-dbg-sergiogama.mybluemix.net>

```

1  applications:
2  - path: .
3  memory: 512M
4  instances: 1
5  domain: mybluemix.net
6  name: ibm-dbg-aboutmyself
7  # Change "${random-word}" to your name, for instance: ibm-dbg-sergiogama
8  host: ibm-dbg-sergiogama
9  disk_quota: 1024M
10

```

Deploy your chatbot, just click on “Run” icon as showed above. Wait 1 to 3 minutes, until the bullet comes green again, as the following:



The screenshot shows the IBM Bluemix DevOps interface. The top navigation bar includes 'Docs', 'IBM Bluemix DevOps' (with a gear icon), 'Catalog', 'Support', 'Manage', and a user profile. The main area has a dark header with 'simple-toolchain-20170607143012290 (running: normal)'. The left sidebar shows a file tree with files like 'Root (console.ng.bluemix.net)', 'simple-toolchain-20170607143012290', 'config', 'conversation', 'launchConfigurations', 'public', 'routes', 'views', '.cignore', '.gitignore', 'app.js', 'LICENSE', and 'manifest.yml'. The right panel is an editor for 'manifest.yml' with the following content:

```
1 applications:
2 - path: .
3 memory: 512M
4 instances: 1
5 domain: mybluemix.net
6 name: ibm-dbg-aboutmyself
7 # Change "${random-word}" to your name, for instance: ibm-dbg-sergiogama
8 host: ibm-dbg-sergiogama
9 disk_quota: 1024M
```

As developer click on the icon as showed above, cross the finger, and...