

Conector eva – infobip

Manual de despliegue y configuración

Tabla de contenido

[HISTÓRICO DEL DOCUMENTO 2](#_Toc52962360)

[OBJETIVO 3](#_Toc52962361)

[1. INTRODUCCIÓN 4](#_Toc52962362)

[**1.1.** **Requisitos previos** 4](#_Toc52962363)

[**1.2.** **Diagrama de componentes básico** 4](#_Toc52962364)

[2. CONFIGURACIÓN DE CUENTA DE AZURE 5](#_Toc52962365)

[**2.1.** **Creación de recursos necesarios** 5](#_Toc52962366)

[3. CONECTOR BOT FRAMEWORK/TEAMS 7](#_Toc52962367)

[**3.1.** **Descargar el conector de OIL** 7](#_Toc52962368)

[**3.2.** **Configurar propiedades del conector.** 7](#_Toc52962369)

[**3.3.** **Ejecutar el proyecto localmente** 8](#_Toc52962370)

[**3.4.** **Bot Framework Emulator** 9](#_Toc52962371)

[4. DESPLIEGUE DE LA APLICACIÓN EN AZURE 10](#_Toc52962372)

[**4.1.** **Modificar el pom.xml** 10](#_Toc52962373)

[**4.2.** **Despliegue del código en Azure** 10](#_Toc52962374)

[5. Creación de la aplicación de MS Teams 11](#_Toc52962375)

[**5.1.** **Edición del manifiesto** 11](#_Toc52962376)

[**5.2.** **Creación de la aplicación en Teams** 12](#_Toc52962377)

# HISTÓRICO DEL DOCUMENTO

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Empresa** |
| 1.0 | 06/04/2021 | Víctor Rubén Torres Criado | everis |

# OBJETIVO

Este documento tiene como finalidad recopilar y organizar el conjunto de aquellas buenas prácticas a la hora de crear una solución de asistentes virtuales o chatbots con el conector infobip.

# INTRODUCCIÓN

En este documento se exponen los pasos necesarios para el despliegue del conector infobip y su configuración necesaria.

Para la utilización del canal WhatsApp, se usa un conector desarrollado especialmente para eva, el cual implementa la API de infobip, lo que permite conexión de eva y WhatsApp mediante peticiones REST.

## **Requisitos previos**

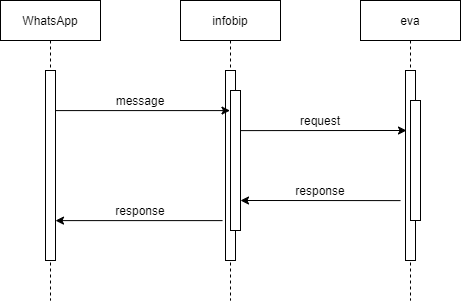
Para el despliegue del conector se necesitan algunos pre-requisitos con el fin de poder realizar la instalación y despliegue en un ambiente en la nube del conector. En este manual se explica como publicar el conector usando Azure Kubernetes Service (AKS), como un microservicio dentro de la infraestructura de eva. Para esto hay que solicitar apoyo y tener los permisos necesarios del eva Cloud de everis.

Los requisitos previos necesarios son los siguientes:

|  |
| --- |
| *Java 1.8* |
| *Maven* |
| *Cuenta de Microsoft Azure* |
| *docker* |
| *kubectl* |

## **Diagrama de componentes básico**

El siguiente diagrama muestra el esquema de interconexión básica de los diferentes componentes de la solución.



Para esta implementación se debe crear una aplicación personalizada de Teams, la cual, una vez instalada, se comunica usando servicios API Rest

El conector del presente documento desarrollado con infobip, recibe las peticiones desde el canal WhatsApp y las reenvía a eva para realizar la identificación de la intención y obtener la respuesta adecuada.

# DESPLIEGUE DE LA APLICACIÓN EN AZURE

En esta sección se indicaran los pasos para desplegar el código fuente del conector en la plataforma Microsoft Azure.

## **Crear imagen Docker**

Se trata de una imagen que contiene el código fuente del conector, para ello necesitamos estar logados en el contenedor de registro de la infraestructura eva.

* docker login [nombre del contenedor].azurecr.io –u [user] –p [password]

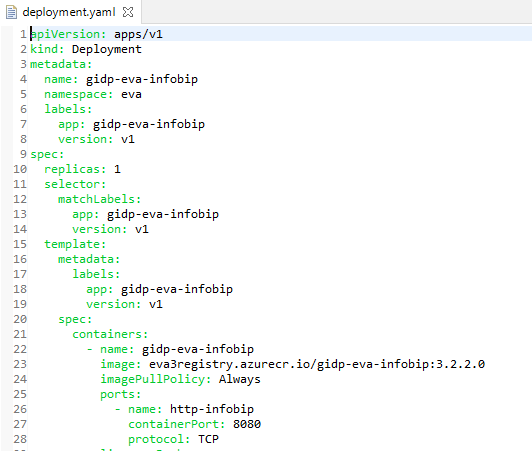
Una vez logados, desde la carpeta raíz de nuestro conector, ejecutamos los siguientes comandos:

1. mvn clean package
2. docker build -t [nombre del contenedor].azurecr.io/[imagen]:[tag] .
3. docker push [nombre del contenedor].azurecr.io/[imagen]:[tag]

## **Despliegue del código en Azure**

Para el despliegue se utiliza kubernetes y los ficheros de despliegue; para ello y previo logeo en nuestro clúster de nodos de Azure, en la carpeta deployment ubicada en la raíz del proyecto, modificamos el fichero deployment.yaml que contiene y se abre un terminal de comandos desde la misma carpeta.

En este fichero debemos modificar la etiqueta image con el valor que se le ha dado en el paso 2.1 cuando se creó la imagen docker.



Una vez modificado el fichero, desde la consola de comandos ejecutamos las siguientes instrucciones en el siguiente orden:

1. kubectl apply –f deployment.yaml
2. kubectl apply –f service.yaml
3. kubectl apply –f virtual-service.yaml
4. kubectl apply –f destination-rule.yaml
5. kubectl apply –f horizontal-pod-autoescaler.yaml

Al cabo de un tiempo si ejecutamos el comando “kubectl get pods –n eva” veremos que nuestro nuevo pod estará levantado y con nuestro código fuente del conector funcionando.

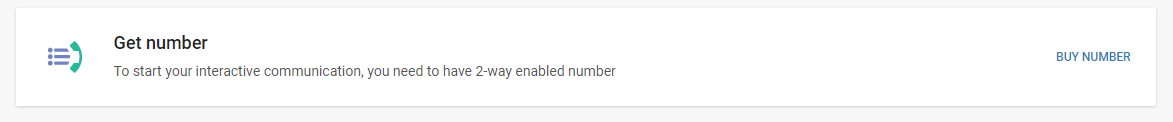
NOTA: Se puede modificar el nombre del pod que se generará, para ello sustituiremos el valor de “gidp-eva-infobip” por cualquier otro en todos los ficheros.

# Infobip

En esta sección se indicaran los pasos para la conexión entre el canal WhatsApp y el conector desplegado.

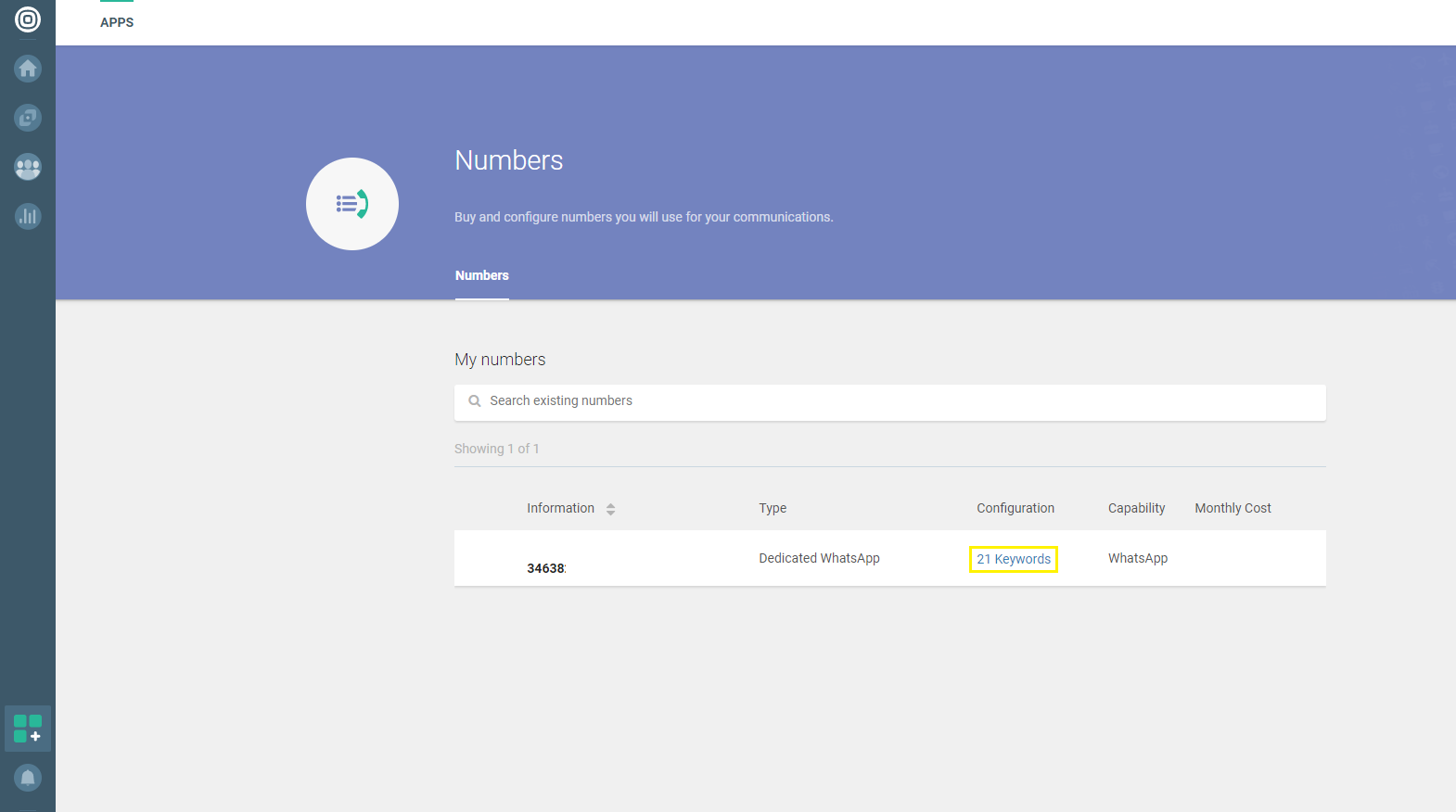
## **Obtener número de teléfono y keywords**

Lo primero que se debe obtener es un número que se obtendrá mediante la compra desde el portal de infobip.

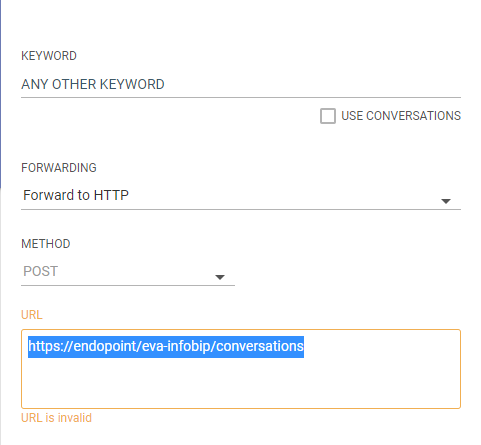


Cuando se haya obtenido, procedemos a la vinculación de este número con el endpoint de nuestro conector, para que todo mensaje entrante sea tratado por el conector desarrollado.

Para ello, nos dirigimos a Apps > Numbers > [nuestro teléfono] y pinchamos en el apartado de keywords.



En el menú lateral, seleccionaremos Any Other Keyword para editarla, añadiremos en la URL el endpoint de nuestro conector junto con la terminación /conversations y guardamos.



Ahora desde nuestro cockpit de eva, iremos a configuraciones y en parámetros añadiremos los valores correspondientes:

- **whatsapp.infobip.info** -> {"omniUrl":"<>","user":"<>","password":"<>","whatsappNumber":"<>","channel":"WHATSAPP","keyword":"<keyword to reference eva bot>"}

- **whatsapp.broker.info** -> {"baseUrl":"http://eva-broker:8080","apiKey":"eva API KEY","project":"eva project name","channel":"WhatsApp","os":"Infobip connector","locale":"project language"}

\*Example:

- **whatsapp.infobip.info** -> {"omniUrl":"https://api.infobip.com/omni/1/","user":"victor.ruben.torres.criado@everis.com","password":"123456789","whatsappNumber":"34638202449","channel":"WHATSAPP","keyword":"GIDP"}

- **whatsapp.broker.info** -> {"baseUrl":"http://eva-broker:8080","apiKey":"12345678","project":"GIDP WA","channel":"WhatsApp","os":"Infobip connector","locale":"es-ES"}

Con todo configurado, podemos probar la conexión, para ello, agregamos el número a nuestros contactos y desde WhatsApp, enviamos un sms.

## **3.2. Funcionalidades**

Con este conector, tenemos la posibilidad de enviar mensajes de texto, imágenes, documentos, localización y audios.

Para el tratamiento de los datos recibidos por parte del usuario en un motor cognitivo se definieron las siguientes reglas:

* Texto: Se envía el mismo texto como input
* Documentos: se envía en el siguiente formato: URL # nombredocumento
* Resto: se envía la URL

Si lo que se quiere es enviar desde eva mediante el cockpit al cliente:

* Texto: Respuesta normal de texto
* Localización: Se tiene que enviar con la siguiente estructura
  + LOCATION-{

“longitude”: Double

“latitude”: Double

“locationName”: String

“address”: String

}

* Resto: Para documentos o audios o videos, se envía la URL de su almacenamiento en un storage, con su correspondiente extensión.
  + Ejemplo: https://storage.com/midocumento.pdf