

## AI CHATBOT, DIGITAL WORKPLACE

# How to create a Botpress workplace chatbot using business data

Botpress is an unusual offering in the landscape of AI chatbots; it's free, and it's open-source, which immediately puts it at the top of the list for many people. Not only that, but it can be hosted on your own equipment - a big plus for organizations that want to keep tight control over where their chatbot requests go.

But if you're planning to use Botpress for an internal office chatbot, will the open-source nature of the platform be a hindrance or an advantage? I've created my own AI chatbot using Botpress and documented the process, so you can follow along to build your own chatbot - or [skip ahead](#) to see my bottom line evaluation of Botpress as a chatbot for your business.

## 1. Determine a use case

Obviously, if you are wanting to create a workplace AI chatbot you need to have a use case for your project. We've already written a comprehensive guide on how to [find the best use cases for enterprise chatbots](#), which I recommend you check out - it's great for helping to define a fixed, achievable scope for your project.

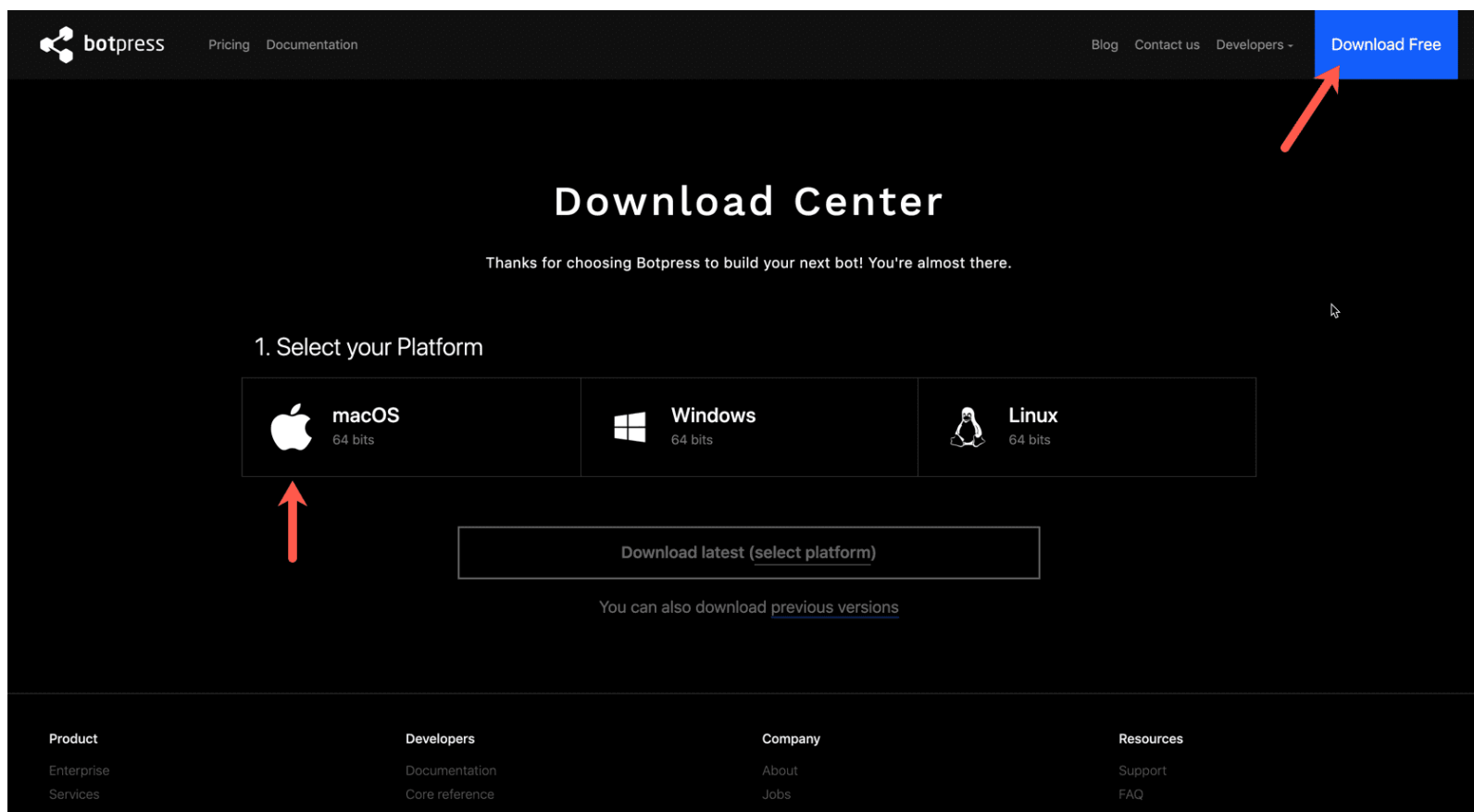
For my team's purposes, I thought it would be cool to make a chatbot that **looks up our newsletter subscriber count from Mailchimp**. From reviewing the [Mailchimp API docs](#), it seems that functionality should be readily accessible to our chatbot. When considering the use case for your chatbot, it's worth checking the API documentation of the apps or services your chatbot will connect to.

### Editor's tip:

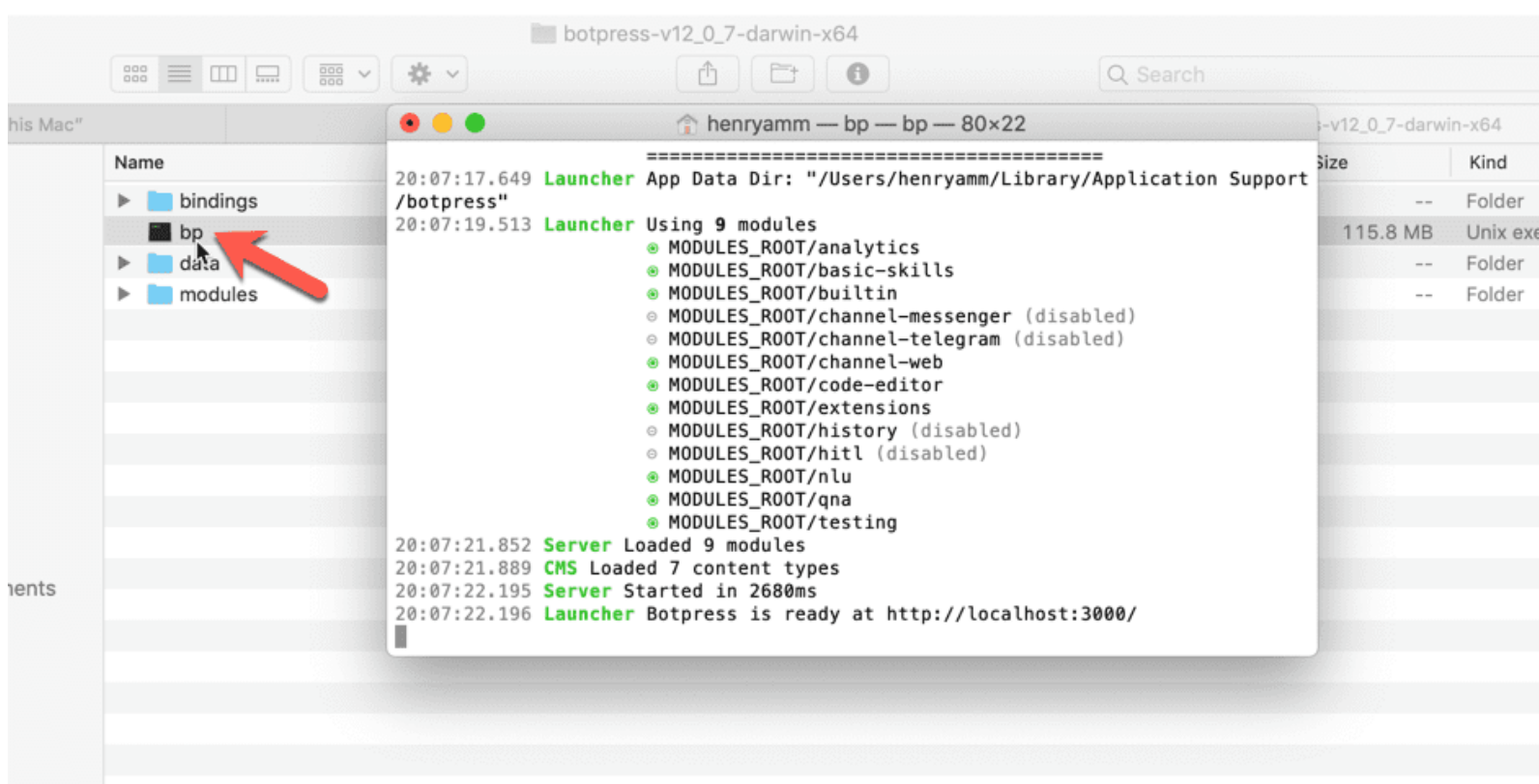
Communicating with different application APIs can be a complicated process, **especially if your intended chatbot use case involves connecting to multiple services**. If you've got grand designs for your chatbot but want to avoid an API nightmare, [Digital Assistant](#) comes with **over 50 connectors out-of-the box** for many popular services.

## 2. Set up Botpress

To install Botpress, head over to their website, click on '[Download Free](#)', and select your platform. I'm going to install Botpress on a Mac, but if you have another platform you can check the [installation instructions](#).



After you download and extract the zip file, double-click the **bp** file within to start a local Botpress development server.

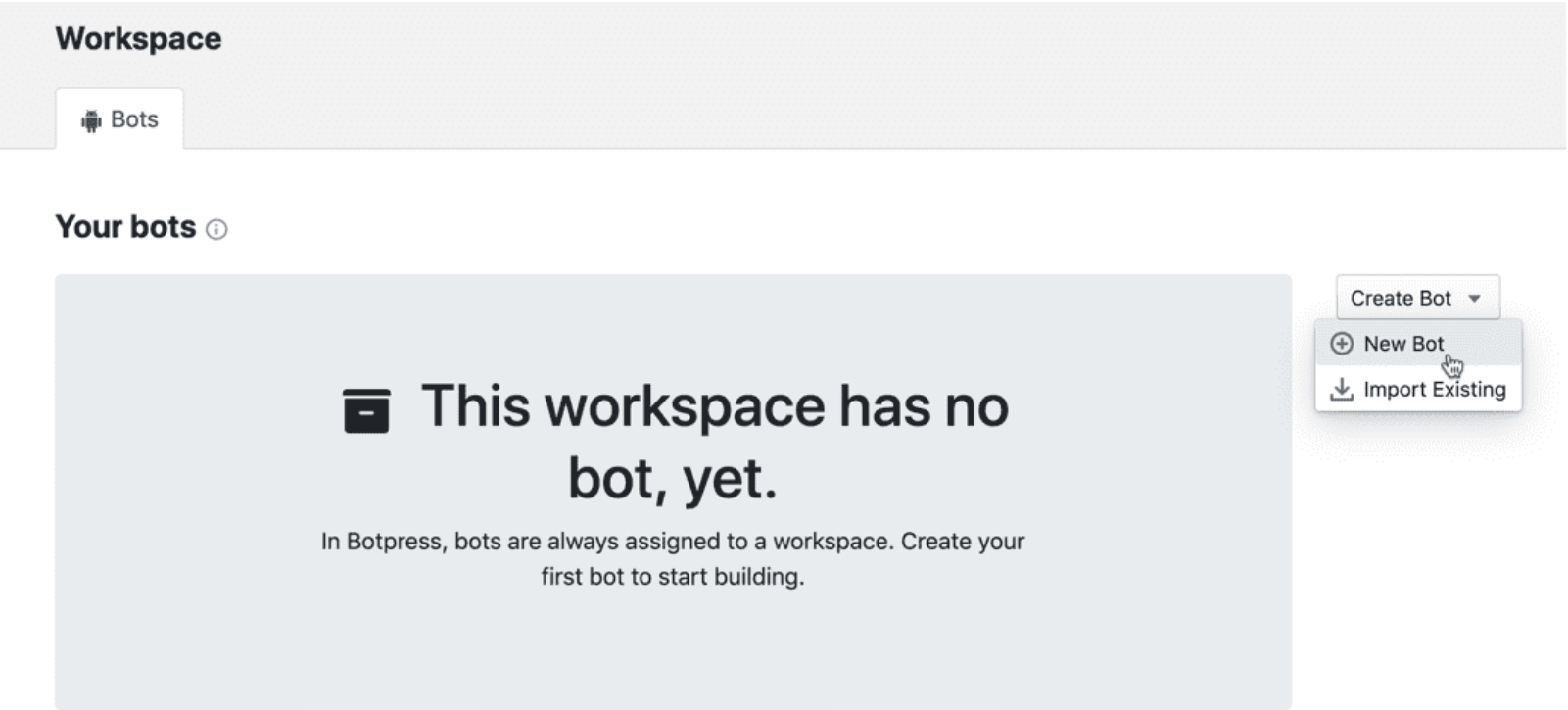


In the case of a Mac, this will open **Terminal** and run the installation which at the end should let you know that

```
1 Botpress is ready at http://localhost:3000/
```

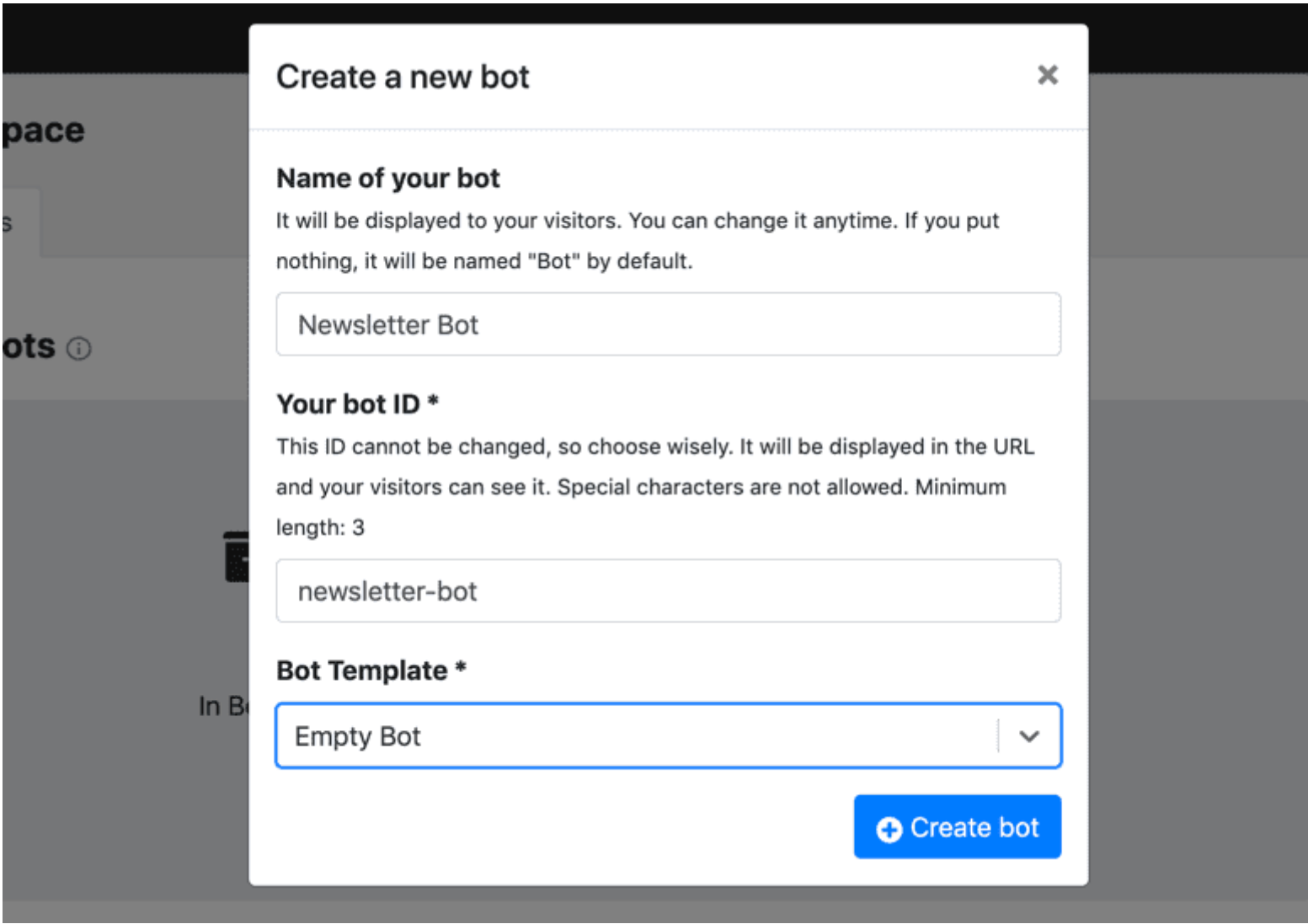
Navigate to that page and then create your admin account.

## 3. Create a new bot

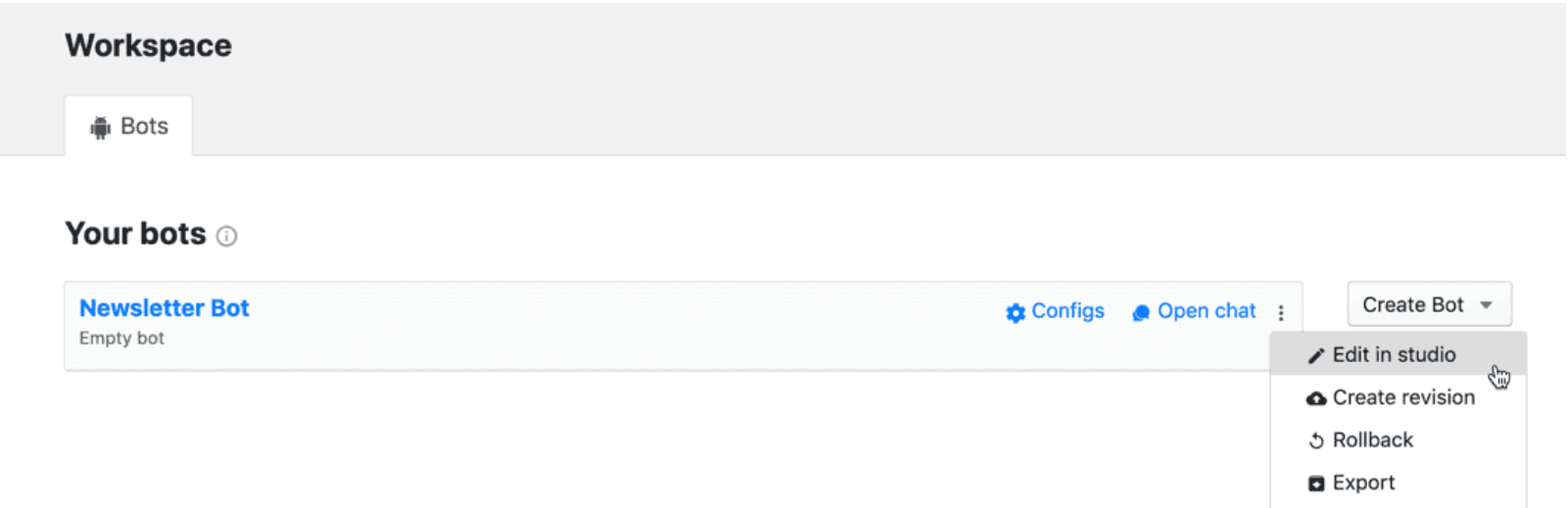


Next, we want to actually create our first bot by click on **Create Bot** and then selecting **New Bot**.

I'm going to call our bot the **Newsletter Bot**, and then as a **Bot Template** I will select **Empty Bot** before clicking on **Create bot**.



Now we want to go straight to **Edit in studio** which opens the otherwise empty bot in the development environment.

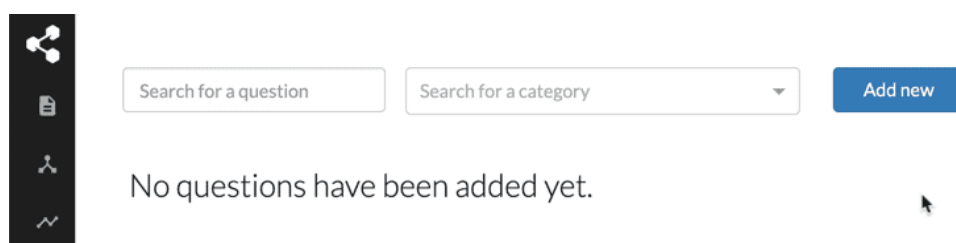


In the Studio you have a number of menus along the left:

- **Content:** this is where you can create different bot responses. You can also create and edit responses here, or you do that from within the Flow editor directly
- **Flows:** here you design the chat flows for your bot. It will look familiar to you if you've ever used any graphical workflow software. Each 'decision point' is shown as a **Node** where actions can be taken. The AI will automatically determine which path it takes based on the user's language
- **NLU:** short for Natural Language Understanding. You can enter training phrases here for each 'intent' (action you want the bot to perform or process) so the it learns which phrases correlate to your defined actions
- **Q&A:** basic text responses. This is is useful for responses questions like 'Hello', 'Help', or 'What can you do?', as well as simple knowledge base questions.

## 4. Add smalltalk

We'll stick with the **Q&A** menu for a moment. We'll use this menu to cover some of the basic smalltalk like 'Hi'.



Click on **Add New** and enter a list of questions you can think of that should trigger a given response from the bot.

Let's say we add in a pretty common 'What can you do?' style question. Q&A questions can be asked anytime, and they can either start a flow or directly skip to a specific node. Or, of course, they can just be on their own which is what we will do.

**Create a new Q&A**

**Category**

global x ▼

◆ Add 5 more questions to make your Q&A more resilient.

**Questions**

Type/Paste your questions here separated with a new line

What can you do?  
Help  
What is your purpose?  
How do you work?  
Show me what I can do...

**Answers**

☒ Bot will say:

I am the Newsletter bot. You can ask me things like 'How many subscribers do we have?' and I will get that information for you in real-time.

and / or

☐ Redirect to flow

Select... ▼

Node

Select... ▼

Cancel Save

You can add as much smalltalk as you like to give your chatbot some personality, and as a way to provide simple help or direction to your users.

## 5. Creating our first Flow

My bot has a very specific use case: it should be able to tell the user how many subscribers are in the newsletter list. However, if the user asks anything that does **not** match this intent, we should prepare our chatbot to gracefully handle their question.

So navigate to the **Flows** menu and then click on **Insert New Node**.


Then double-click the new node and rename it to something like "**No\_match\_found**".

You also have three tabs in this popup: **On Enter**, **On Receive** and **Transitions** that you can add **Actions** to. These actions will be triggered when the conversation flows into this node and they could either be a text response or something more complicated like a script.

While we will add a complicated **Action** further down, for now we just want a simple text response that says:

Sorry, I didn't quite understand that. Can I help you with anything else?

So in the **On Enter** tab click on the  icon to add a new action.

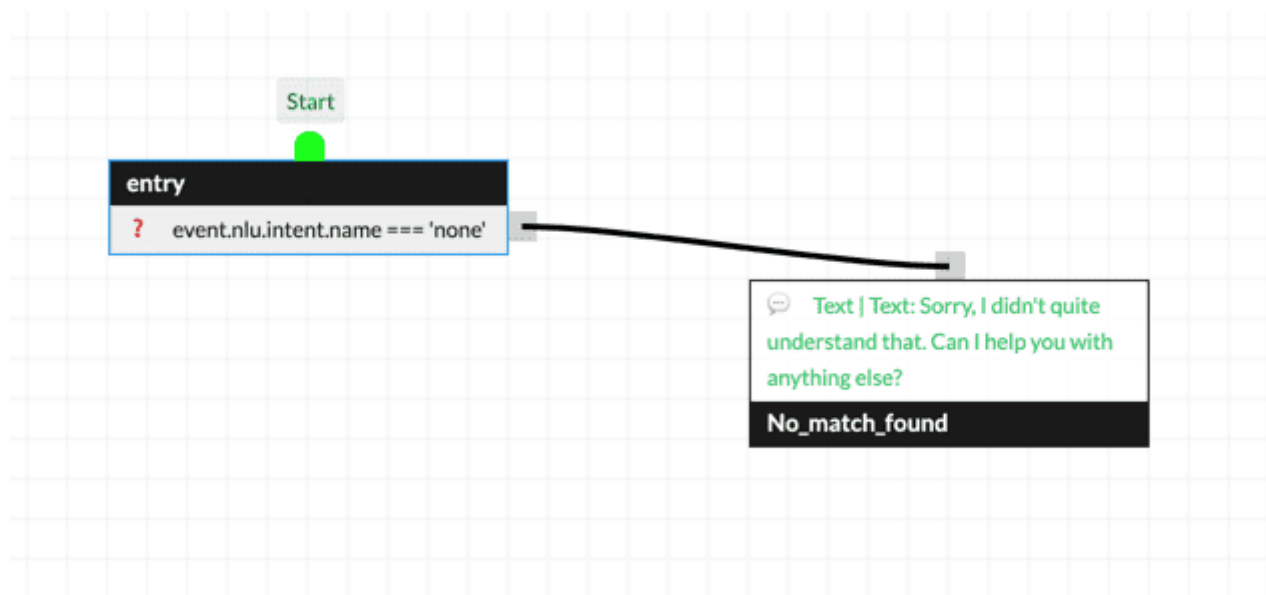
You will now see that the **Message** section is still blanked out and that's because messages have to be created in the **Content** system first. You can do that in the same dialogue though by clicking on the  blue folder icon.

## 6. Make a transition

**Transitions** are links between nodes. So in our example we want to link the start node to our new **No\_match\_found** node. To do that, double-click the start node and add a **Transition** to it.



If you select 'Intent is' as **none** and 'Transition to node' as **No\_match\_found** and click **Update** you should see an automatic line being drawn from the first node to the second.





#### Editor's tip:

At this point you could try to see if the bot picks up on this already by opening the

 Emulator and typing in anything. However if you do that, be sure to hit **Save**  first

because Botpress won't do it for you. 😊

## 7. Add a real Action

By now we've covered all the basics, so we can now move onto our actual use case: **Getting the subscriber count from Mailchimp into our chatbot.**

Like I said above, **Actions** allow the execution of code so we can use that to pull data dynamically from the Mailchimp API.

#### Editor's tip:

Writing Action functions can be **technically challenging!** I've included my function for the Mailchimp API below, but your function will depend heavily on the API you're getting information from. (If you don't want to go through the hassle of writing functions, you can [get a Digital Assistant for free](#) - it includes connectors to **over 50 applications out of the box**, with zero code needed.)

To create a new **Action**, navigate to the **Code Editor** tab and add a new `.js` function as shown.

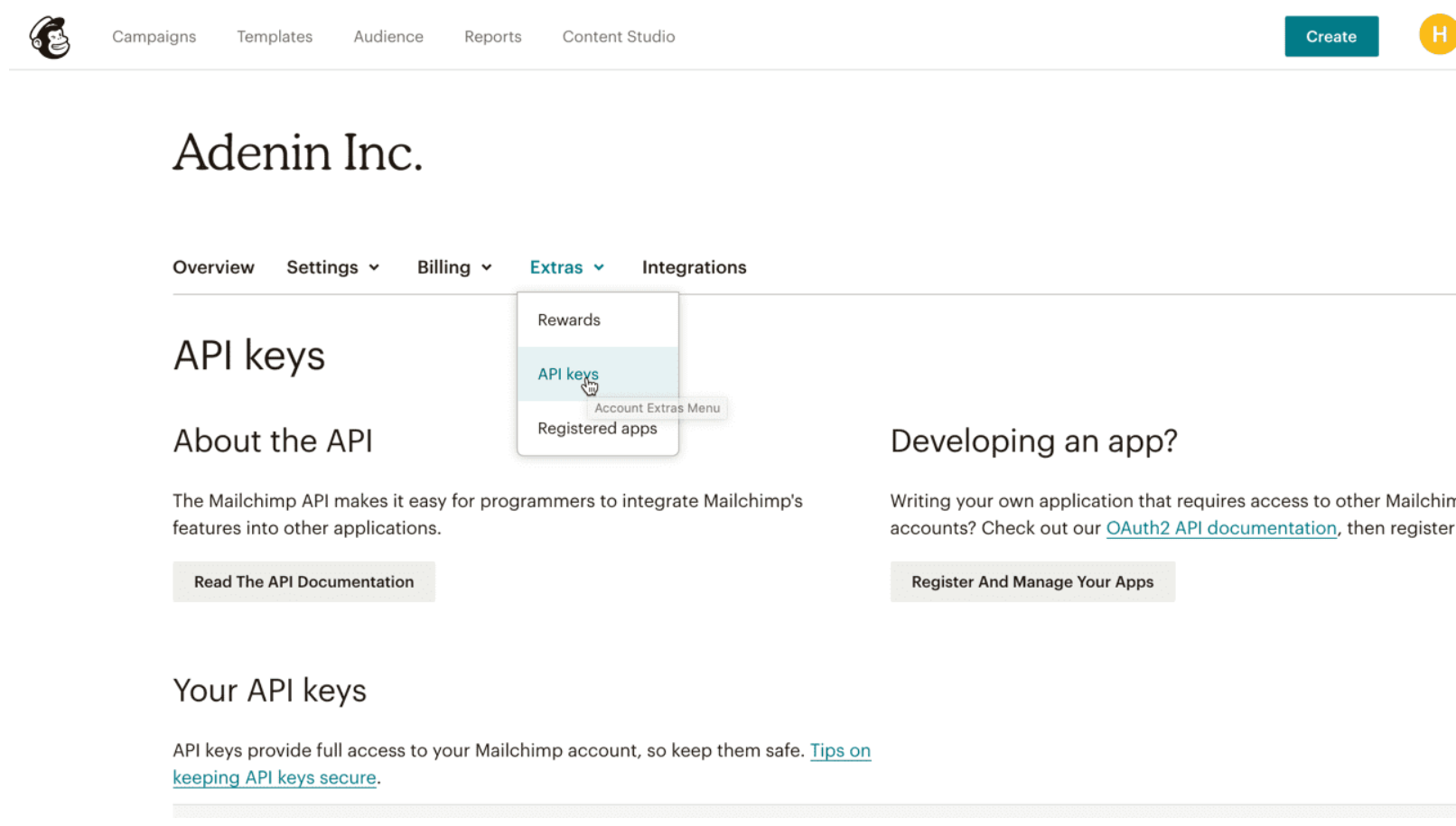
Obviously this is where it gets a little tricky as you have to write your own function. Below is the function I used for Mailchimp which I've annotated as I go along. However if you don't happen to use Mailchimp yourself, this code will look vastly different.

```
1  /** Your code starts below */
2
3  const axios = require('axios');
```

Axios is a Botpress dependency that lets us make HTTP requests.

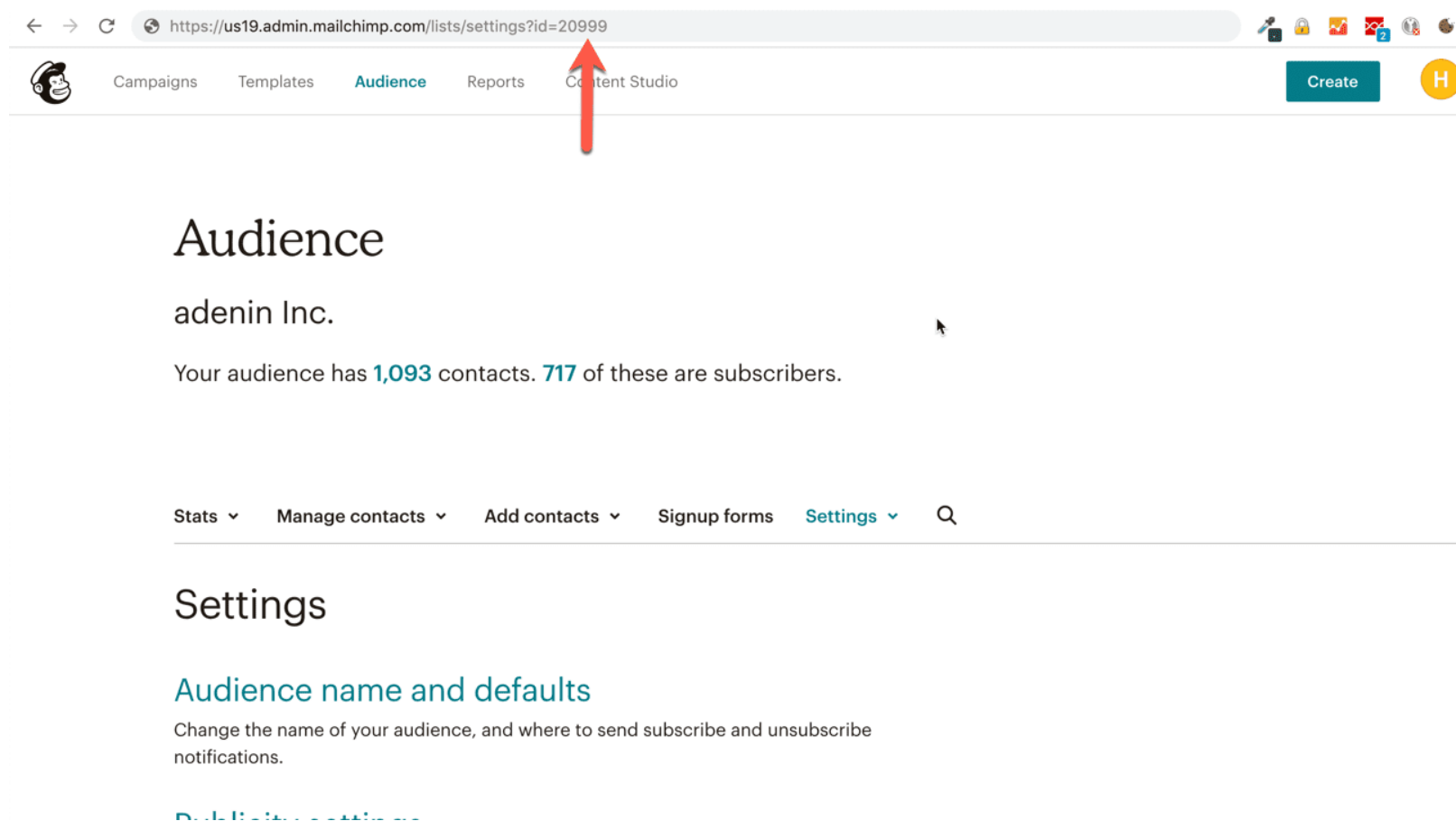
```
1  const listID = 'xxxxxxxxx';
2  const API_KEY = "xxxxxxxxxxxxxxxxxxxxxxxx-us10";
```

Define the Mailchimp list ID that you want to get information about, as well as your Mailchimp API key. You can create your API key from your Mailchimp account settings and the list ID can be obtained from the list settings themselves.



The screenshot shows the Mailchimp account settings page for 'Adenin Inc.'. The navigation bar at the top includes 'Campaigns', 'Templates', 'Audience', 'Reports', and 'Content Studio', along with a 'Create' button and a user profile icon. The 'Extras' menu is open, showing options for 'Rewards', 'API keys', 'Account Extras Menu', and 'Registered apps'. The 'API keys' section is highlighted. Below this, there is a section titled 'About the API' with a description and a link to 'Read The API Documentation'. To the right, there is a section titled 'Developing an app?' with a description and a link to 'Register And Manage Your Apps'. Below these sections, there is a section titled 'Your API keys' with a description and a link to 'Tips on keeping API keys secure'.

Create an API key from account settings under Extras → API keys



The screenshot shows the Mailchimp 'Audience' settings page for 'adenin Inc.'. The navigation bar at the top includes 'Campaigns', 'Templates', 'Audience', 'Reports', and 'Content Studio', along with a 'Create' button and a user profile icon. The 'Audience' section is highlighted. Below this, there is a section titled 'Settings' with a sub-section 'Audience name and defaults'. The 'Audience name and defaults' section has a description: 'Change the name of your audience, and where to send subscribe and unsubscribe notifications.' Below this, there is a section titled 'Subscription settings'.

```
1  const configAxios = {
2    headers: {
3      'authorization': "Basic " + Buffer.from('randomstring:' +
API_KEY).toString('base64'),
4      'Content-Type': 'application/json',
5    },
6  };
```

This is a configuration object for our request. We need to authenticate with the Mailchimp API before it can send us any data – this is where your API key comes in.



```

1  const getData = async () => {
2      console.log("Starting");
3      try {
4          const { data } = await
axios.get(`https://us19.api.mailchimp.com/3.0/lists/${listID}?fields=stats`,
configAxios);
5          temp.count = data.stats.member_count;
6      } catch (err) {console.log(err)}
7  }

```

We make a **network request to the Mailchimp API lists endpoint**, in this case requesting specifically the 'stats' field. You can see documentation of the full API response [here](#). Once we've got the data, we **store the part we need in a memory variable** – see below for more info about this. The name of the variable is important as we'll need it in a moment, so don't forget it!

```

1  return getData();
2  /** Your code ends here */
3  ...

```

We return the function so the bot can call it when the correct conditions have been met. Make sure to save your code at this point!

## 8. Commit to Memory

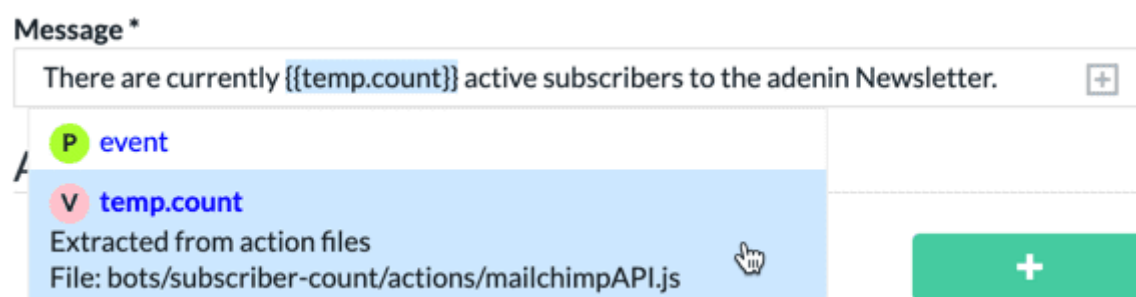
**Memory** is temporary storage in the bot that can store data returned by an Action until it is parsed into a text response. We are only going to save our variable until the end of the Flow, but there's a full explanation of [how Memory works](#) in case you want to do more with it.

In our code this line saves the return from the API as temporary memory with the name `temp.count`:

```

1  temp.count = data.stats.member_count;

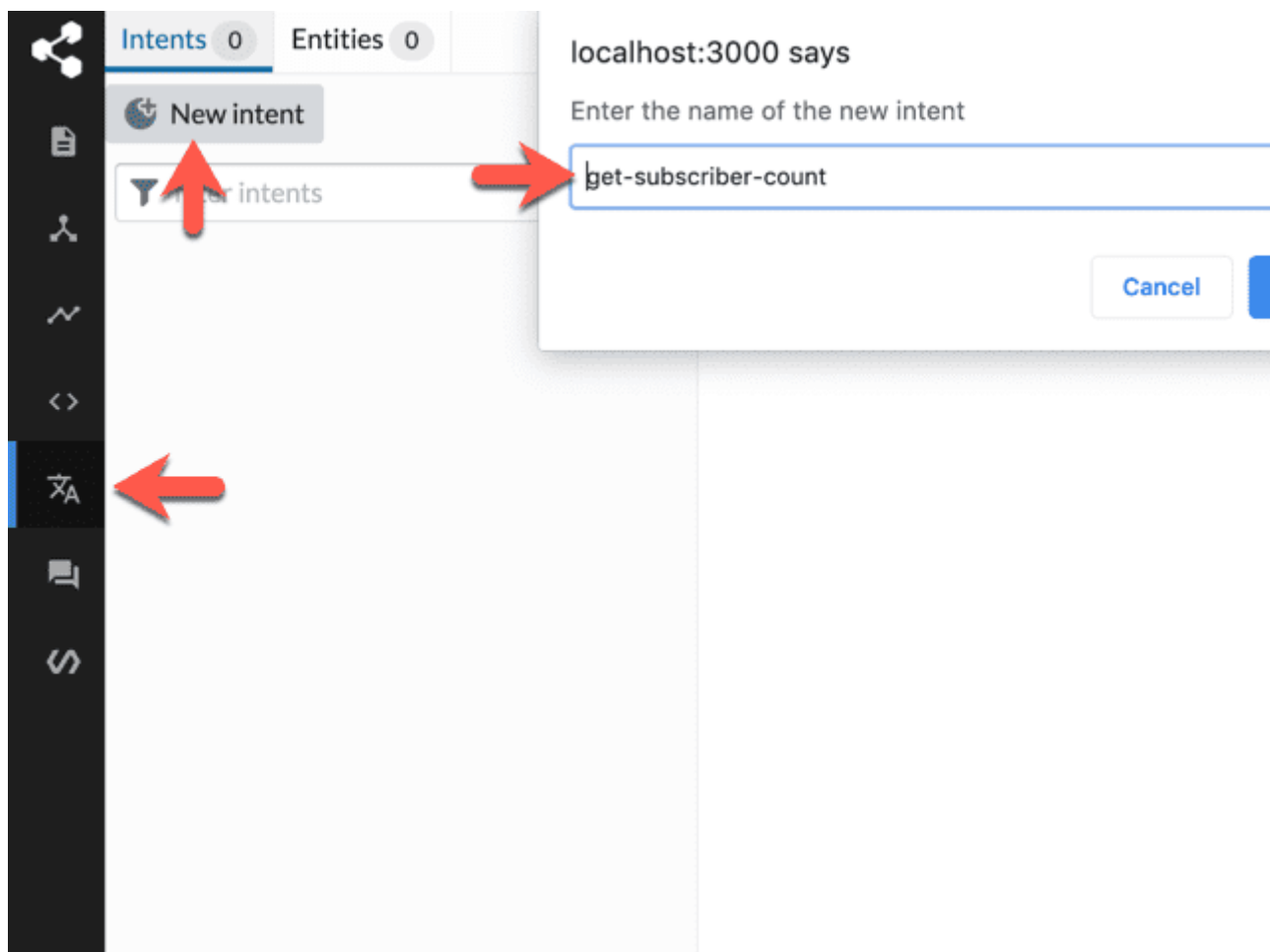
```



In a little bit we will create an Action and define what the bot should actually say. Botpress will automatically suggest we use the value stored from our script so there is nothing else to do.

## 9. Train the intent

For the bot to understand the user input we need to train it with sample utterances. To do that, switch to the **NLU** menu and click on **New Intent**. We'll name the intent **get-subscriber-count**.



Now let's add some possible questions or commands that the user might say when they want to trigger our Action. You should add at least 5 utterances here so the bot can understand a variety of ways users might ask for the intent.

intents/get-subscriber-count

Current contexts i

global x

Type to create a new utterance

How many newsletter subscriptions

Number of newsletter subscriptions

Number of active subscribers

How many subscribers

Subscriber count

How many people are subscribed to the company newsletter?

How many people are subscribed to my newsletter?


How many subscribers do I have?

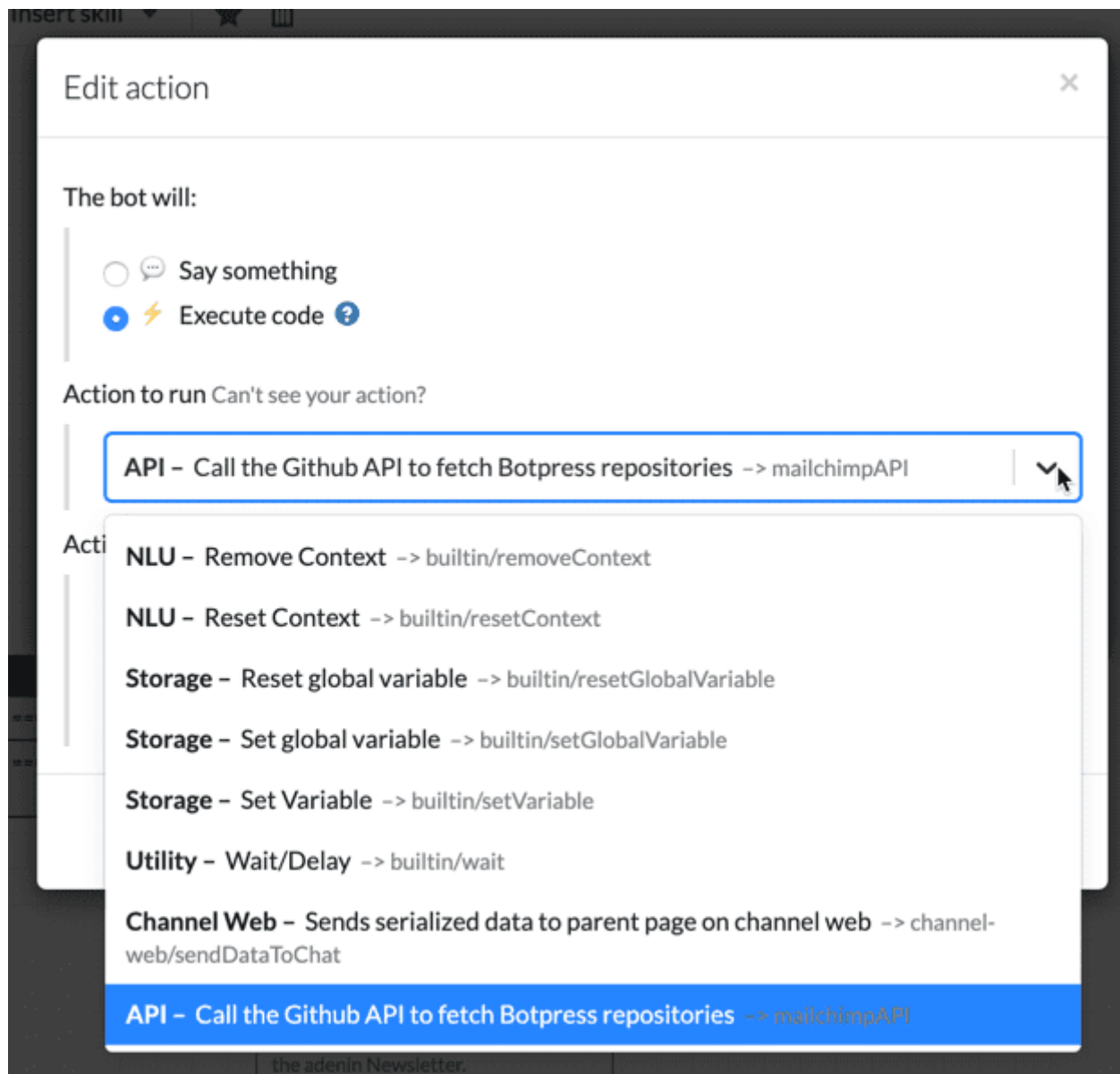
How many subscribers do we have?


Get subscriber count

## 10. Add the Action

Finally, we're ready to add the **Action** into our **Flow**. So let's switch back to the Flow menu and add a new **Node**, then double click that to edit it.


In the **On Enter** tab click the  to add a new action. In the next window we will select **Execute code** and select the action we made earlier from the dropdown.



So now this node will receive the data but there's one more thing we need to do to be able to answer the user: **Parse the data into a sentence**. So let's click the  again and this time we **Say something** and enter the following message:

There are currently `{{temp.count}}` active subscribers to the adenin Newsletter.

Note how this sentence uses `{{temp.count}}` as the placeholder from the **Memory** we created to dynamically insert our answer into the sentence.

Now we need to connect our start node to this new node. Double click on the start node again and go to the **Transitions** tab and click the  again. This time we'll select the **get-subscriber-count** intent.

New condition for transition

Condition

☐ Always

☒ Always

☐ Member Property

☐ New Registration (advanced)

When condition is met, do

☐ End Flow

☐ Return to previous flow

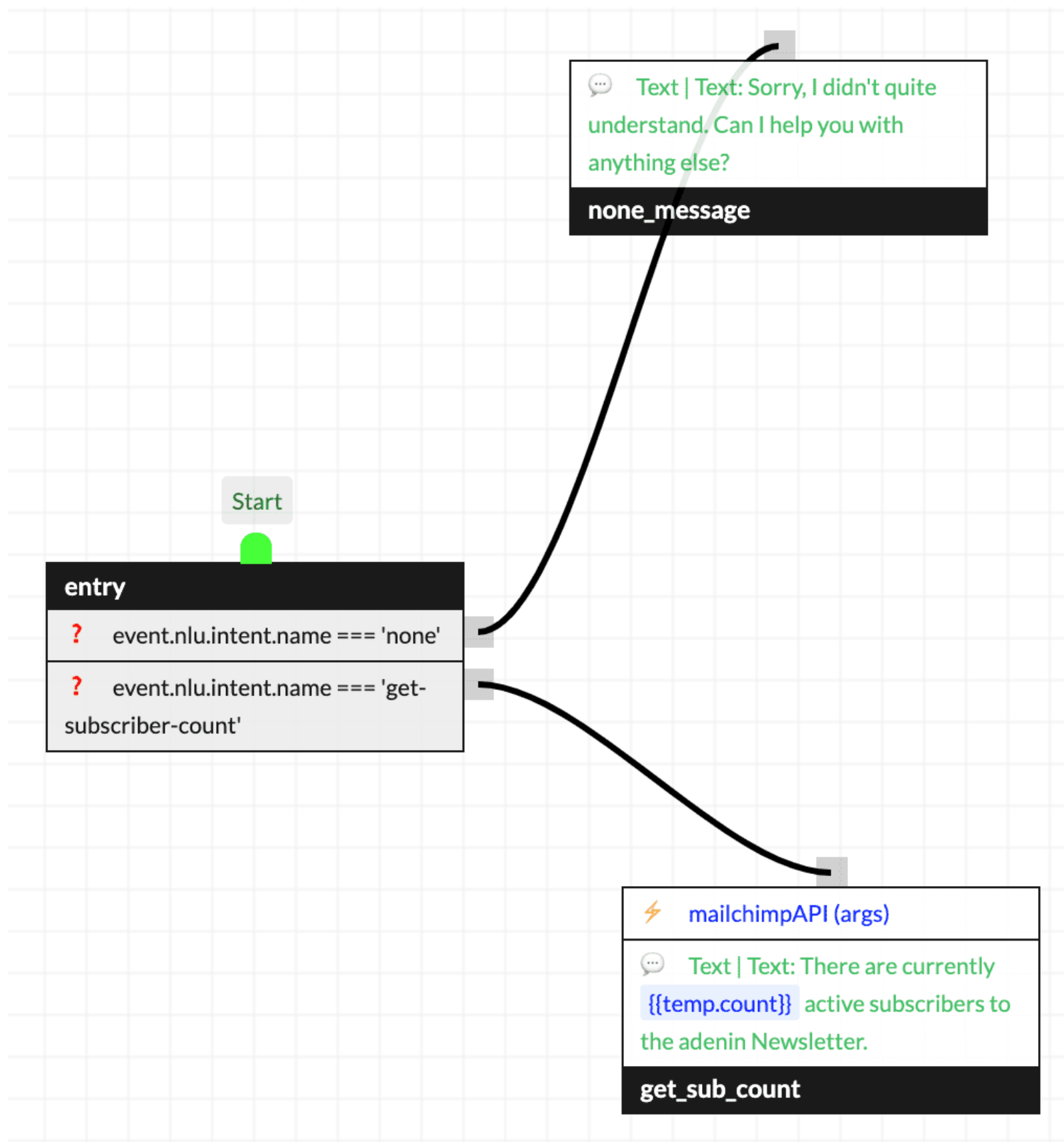
☒ Transition to next

☐ Transition to another

Cancel

Save

Once that's done, we're done. The finished flow should look something like this:



## 11. Debugging

Next we give the bot a **final test in the Emulator** by asking one of the utterances from our training data.



Whatever the reply you'll get, a debugging console to the left of the chat gives you a clue about which steps your **Flow** took and with what percent of confidence the bot picked an **Action**.

**Editor's tip:**

Things can (and likely will) go wrong! The debugger is a very useful tool for deciphering error messages and understanding why your chatbot isn't responding how you expect it to.

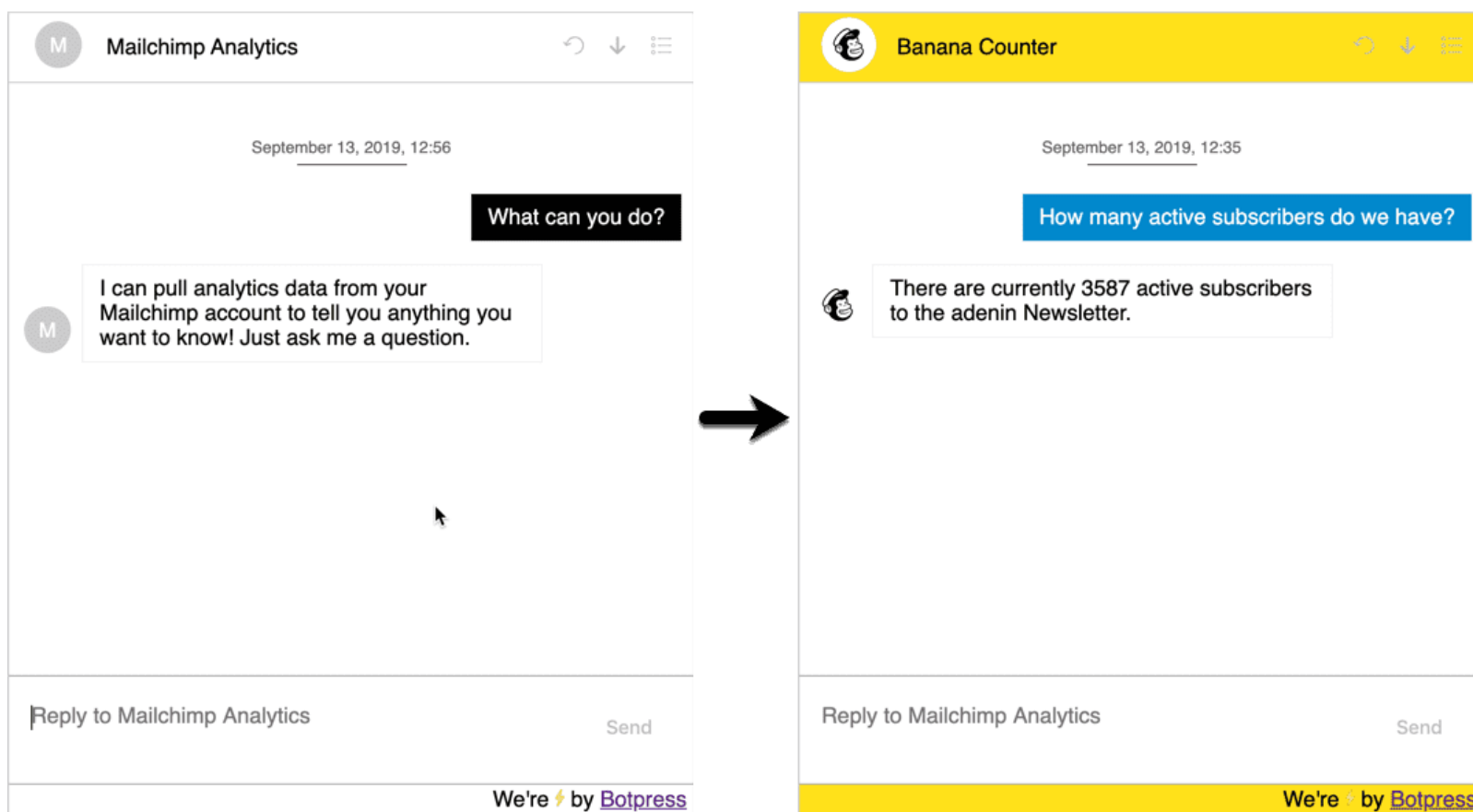
## 12. Deploy

**Deploying your Botpress chatbot** is probably the easiest part, as it comes with a handy webchat component. Just go back to the Botpress main admin panel and click on **Open chat**. This loads your bot into a webchat module which you can directly link to.

Alternatively, it's you can [embed the webchat directly into your website](#). And since you host the bot yourself, it's entirely secure and everything stays behind your firewall.

Embedding the Botpress bot also allows you to customize the look and feel to match your Intranet's look and feel. If you're using a static site, you can even get a [module for webchat](#) from npm.





Since version 12 you can also deploy your bots to external **Channels**, among them [Microsoft Teams](#) and [Slack](#). That's extremely helpful and makes Botpress an even better tool for internally-facing chatbots, as most coworkers will want to use their chatbots on either one of those two.

However, the setup is a little long-winded as you can't simply go into a 'store' to enable these for your collaboration app. Instead, you need to manually configure these as a custom App (for **Slack**) or a custom Bot (on your own **Azure** instance).

#### Editor's tip:

If you're thinking of building a chatbot to embed in your intranet or to power your digital workplace, [Digital Assistant](#) could save you a lot of time and effort. The chatbot **already works with your existing business apps**, and it can be added to collaboration apps like Slack or Microsoft Teams in seconds - **sparing you the extra legwork of programming API requests and configuring custom solutions**.

## Bottom line

As always when we make a chatbot here's our key takeaways for Botpress.

- + Everything you need to run an on-premise, secure chatbot
- + Great documentation, community, debugging tools
- + The 'studio' while not sporting the slickest UI, is **very robust** and easy enough to understand
- + Webchat module easy to embed and customize
- + **Free**

- **Only for developers** as there are no plug-and-play integrations with other applications
- You need to **host it yourself** (on-premise or with a hosting provider)
- **No Single Sign On** for business applications built-in, so showing personalized lists, tasks, tickets, etc. will need even more scripting
- **Setting up external channels** is not as easy as with other platforms

Overall using Botpress is quite pleasant, there are no blindspots in their functionality and the platform is quite lively, if a little geared towards developers. 👍

However, as you can see, for the specific purpose of building an **internal** chatbot, there are a few features we're missing: Just imagine you had a user that wanted to ask for their leave allowance through Slack, and Botpress would fall flat on its face. 🙄

### Trying to decide which chatbot is right for your team or organization?

We've created a series of guides and teardowns of different chatbot-building platforms.

Learn how to scope out [workplace-related use cases](#), or [view all our guides on the adenin blog](#).



Article by: **Henry Amm**



I'm the Senior Director for the Digital Assistant Platform. Previously gained 6 years of experience as an Intranet consultant. Fluent in German.