

# AI & CHATBOT

Aula 06 – Node-RED e Integração de  
Serviços em Nuvem II

Prof. Henrique Ferreira

Prof. Miguel Bozer

Prof. Guilherme Aldeia

Prof. Michel Fornaciali

FIAP  
GRADUAÇÃO

# Telegram

Conectando o bot a um serviço de mensagens

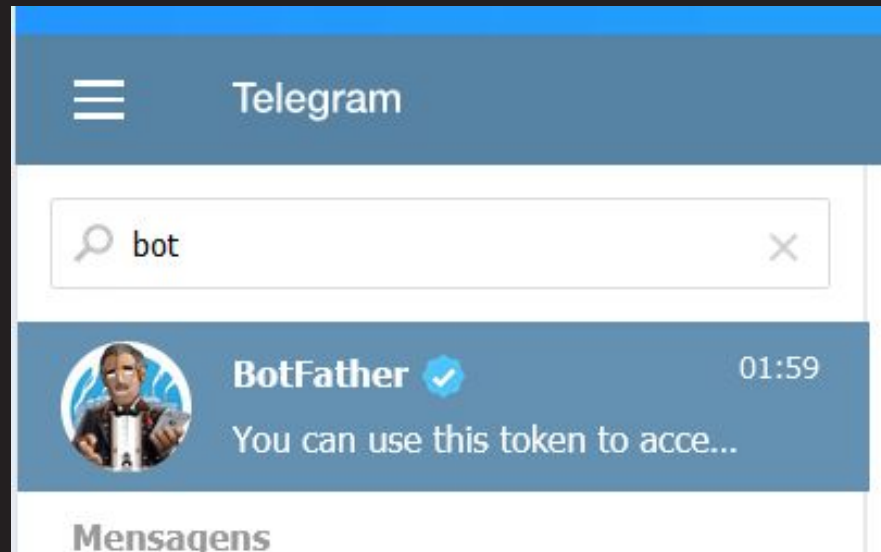
# Telegram



- Nesta primeira etapa vamos apenas habilitar o serviço de criação de bots do Telegram;
- Escolhemos o Telegram pois não é necessária uma conta empresa na plataforma para poder criar bots;
- Primeiro passo: instale o aplicativo do Telegram no seu celular;
- Segundo passo: acesso o Telegram Web no seu navegador;

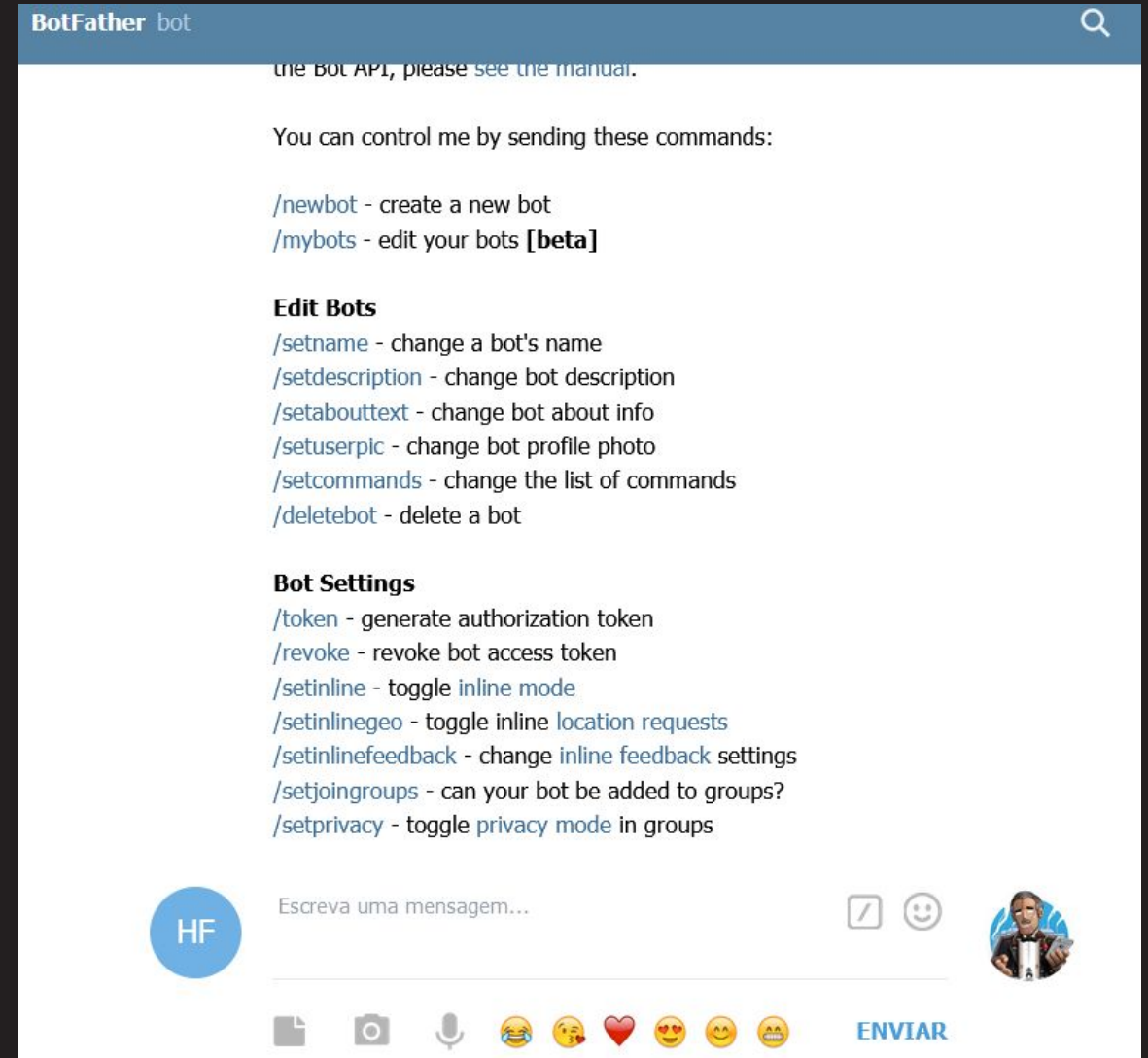
# Telegram - Botfather

- Na barra de busca, digite BotFather.
- Selecione o bonequinho maneiro do BotFather imitando o bom Don Corleone em O Poderoso Chefão.



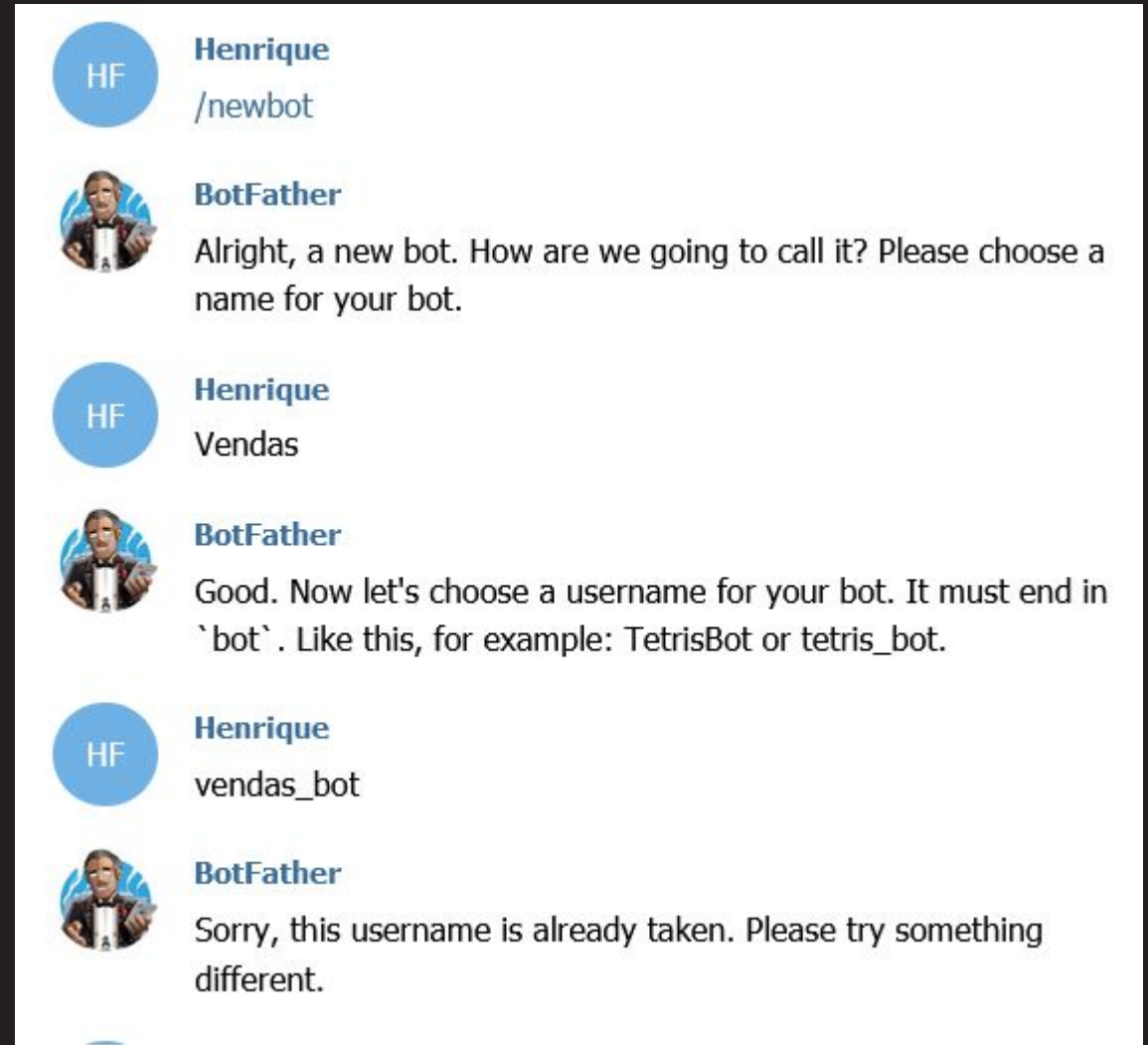
# Telegram - Botfather

- O BotFather irá soltar uma mensagem com uma série de comandos para se criar e editar um bot



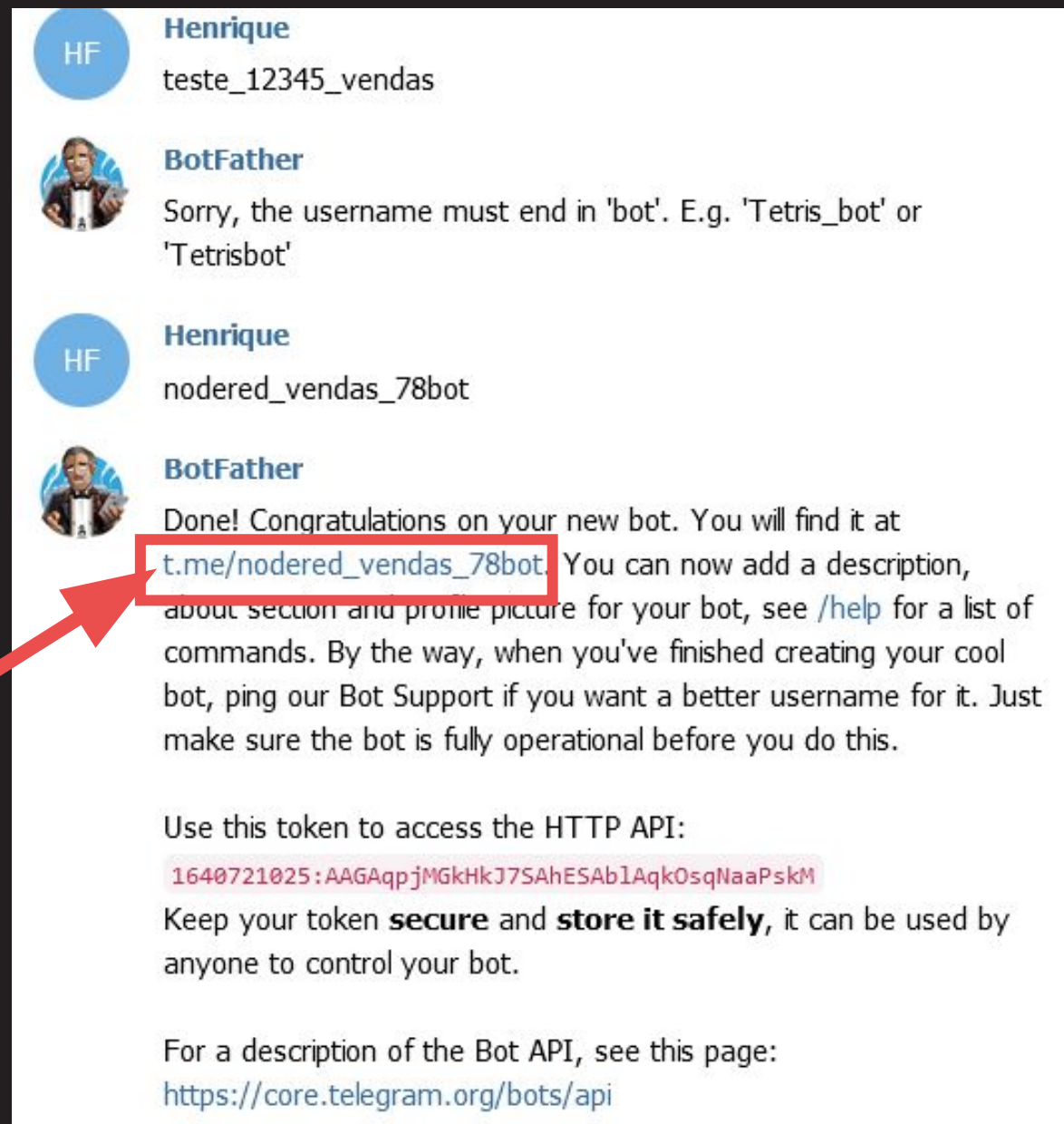
# Telegram - Botfather

- Digite **/newbot** para criar um bot;
- Dê um nome para seu bot
- Agora escolha um username para ele. Atenção, usernames são públicos, então você precisa criar um username único.



# Telegram - Botfather

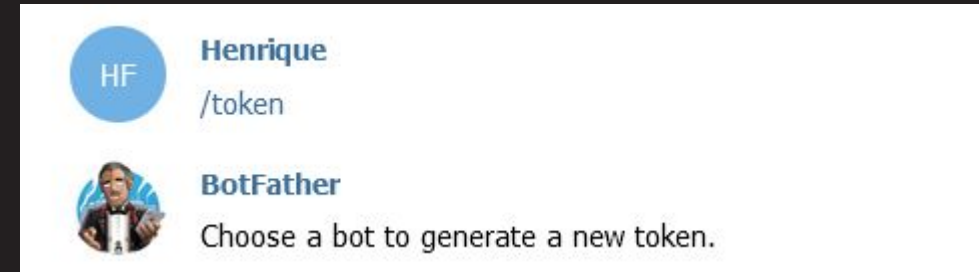
- Não esqueça de escrever bot na parte final do username.
- O BotFather irá soltar uma mensagem de criação bem sucedida.
- Este é o link para iniciar uma conversa com o seu bot;
- Anote o número do HTTP API para acessar o seu bot.





# Telegram - Botfather

- Digite **/token** para verificar o token do seu bot no telegrama;
- Em seguida coloque **@username** do seu bot para verificar o HTTP API sempre que necessário;
- Esse número será usado no Node-RED para criar uma conexão entre o Telegram e o Watson Assistant;





# Testando o bot com Node-RED

Fluxo de demonstração para testar a conexão com o bot e o Telegram

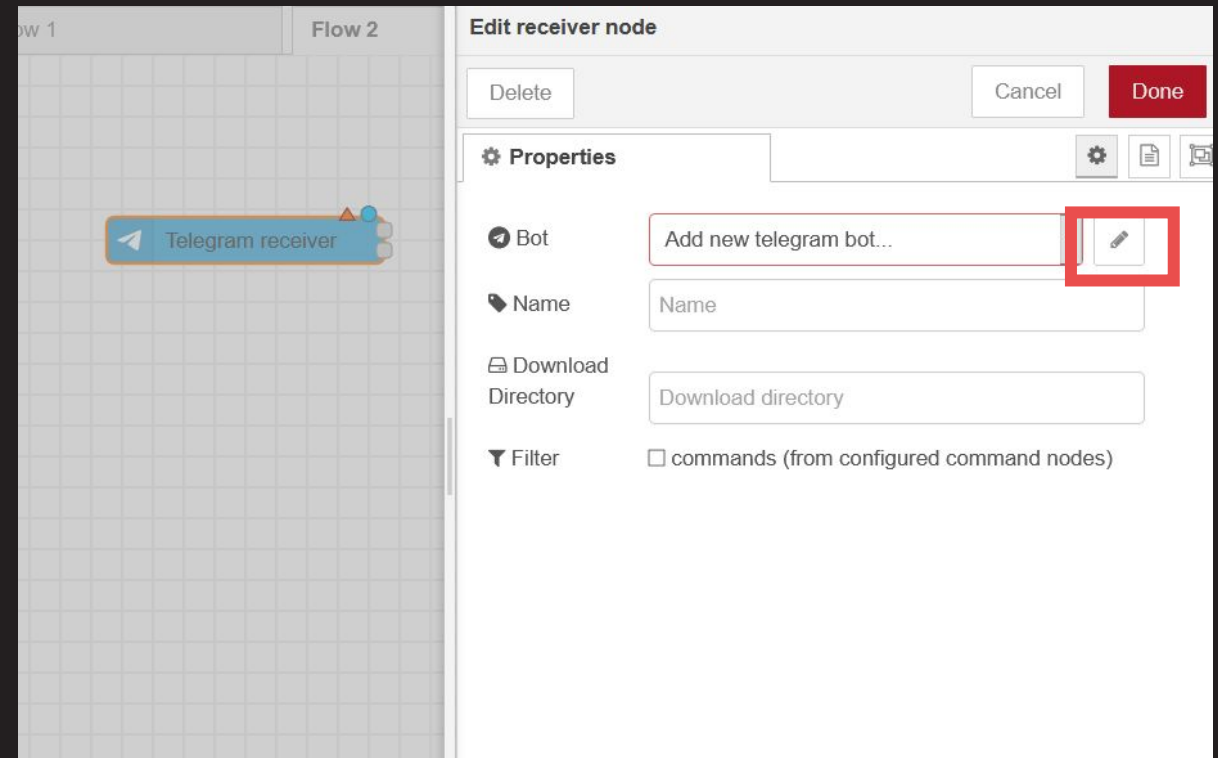
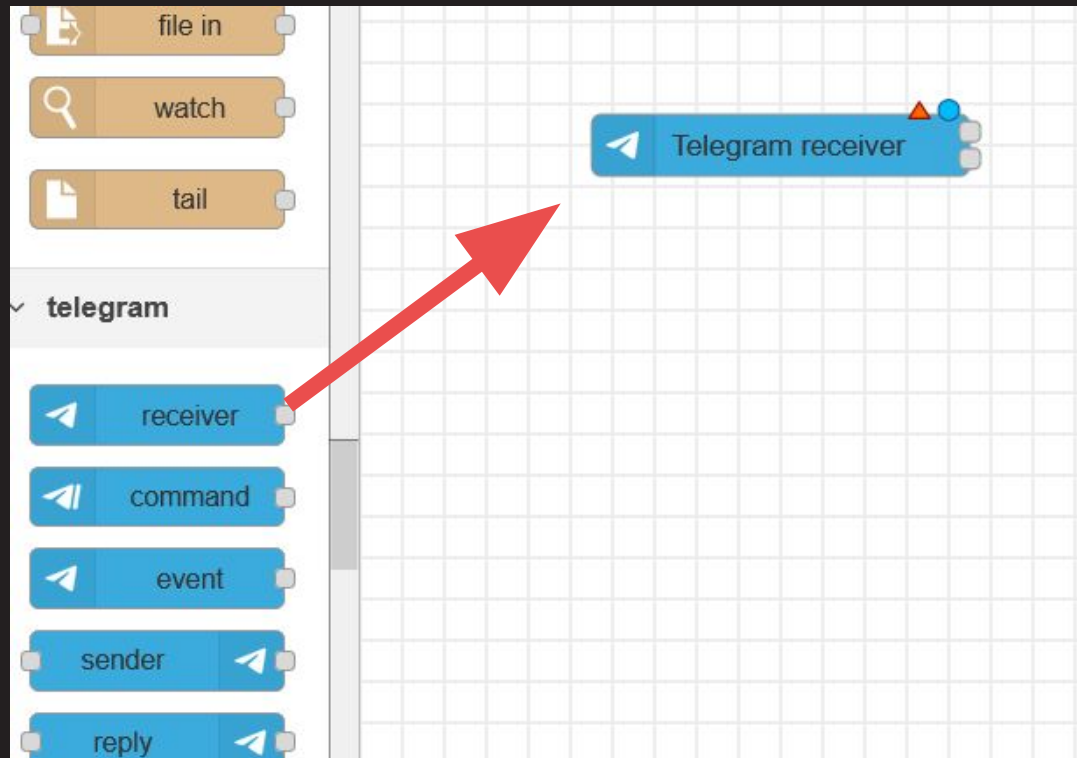
# Abrindo o Node-RED

- Abra o terminal e suba o servidor Node-RED na sua máquina;
- Para o ambiente de desenvolvimento,, coloque o IP local e porta no seu navegador;

```
D:\>node-red
4 Apr 12:32:37 - [info]
Welcome to Node-RED
=====
4 Apr 12:32:37 - [info] Node-RED version: v1.2.9
4 Apr 12:32:37 - [info] Node.js version: v12.13.0
4 Apr 12:32:37 - [info] Windows_NT 10.0.19042 x64 LE
4 Apr 12:32:39 - [info] Loading palette nodes
4 Apr 12:32:43 - [info] Settings file : C:\Users\Ferreira\.node-red\settings.js
4 Apr 12:32:43 - [info] Context store : 'default' [module=memory]
4 Apr 12:32:43 - [info] User directory : C:\Users\Ferreira\.node-red
4 Apr 12:32:43 - [warn] Projects disabled : editorTheme.projects.enabled=false
4 Apr 12:32:43 - [info] Flows file : C:\Users\Ferreira\.node-red\flows_DESKTOP-E3VFKJ5.json
4 Apr 12:32:43 - [info] Server now running at http://127.0.0.1:1880/
4 Apr 12:32:43 - [warn]
```

# Telegram + Node-RED

Vamos adicionar um nó de recebimento do telegram (**telegram receiver**); Duplo clique no nó para alterar as propriedades, e então, clique no ícone do lápis;



# Telegram + Node-RED

Precisamos colocar o nome do Bot e o **Token** para se conectar com o bot que criamos através do BotFather. Uma vez preenchido, clique em **Update**.

Edit receiver node > Edit telegram bot node

Delete Cancel Update

**Properties**

**Bot-Name** (Name of bot to connect to)

**Token** (Enter the bot token from botfather here)

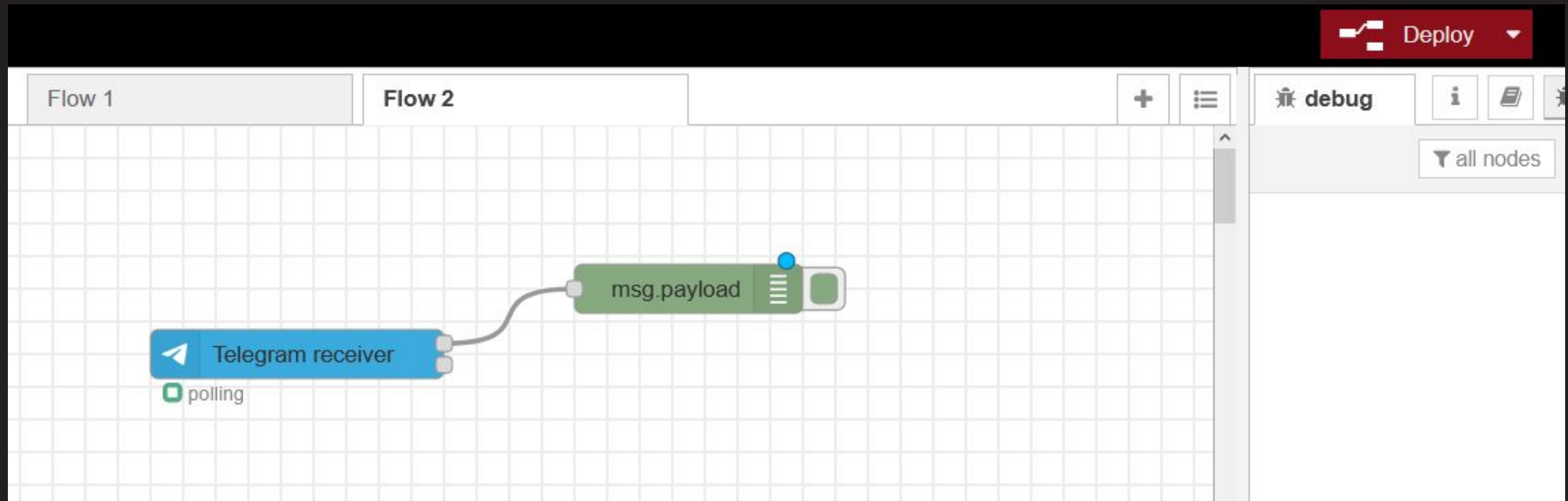
**Tip:** If you don't have a token yet, you can create a new one here: [@BotFather](#).

**Users** (Optional list of authorized user names e.g.: hugo,sepp,egon)

**ChatIds** (Optional list of authorized chat-ids e.g.: -1234567,2345678,-3456789)

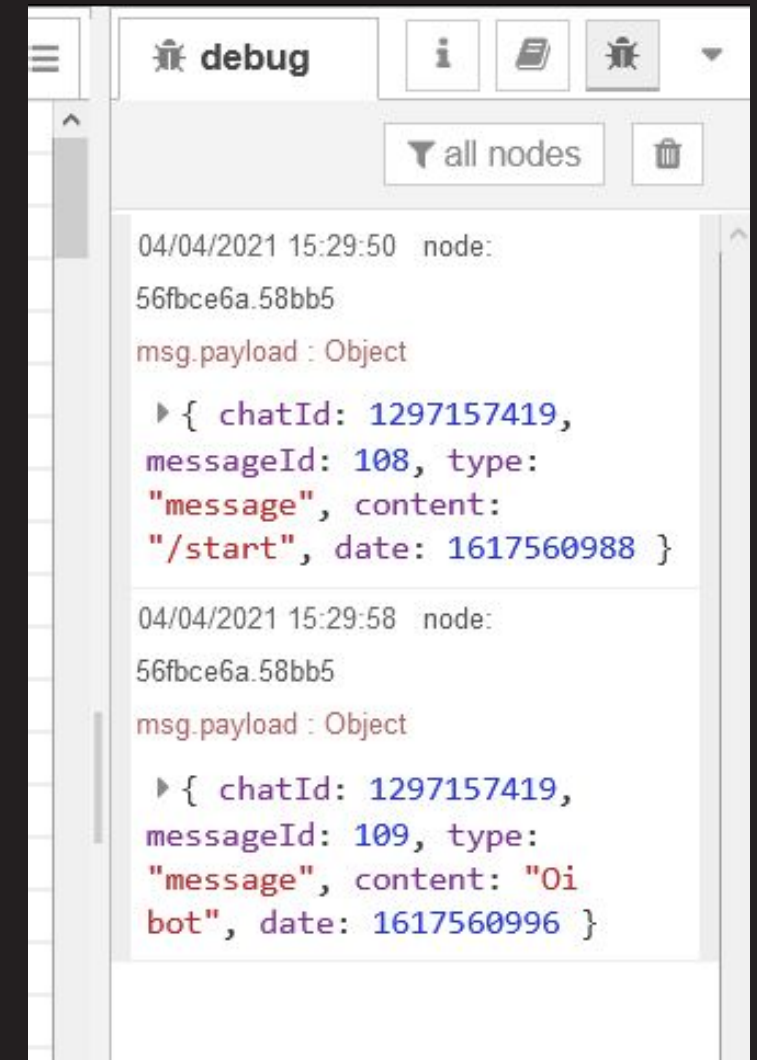
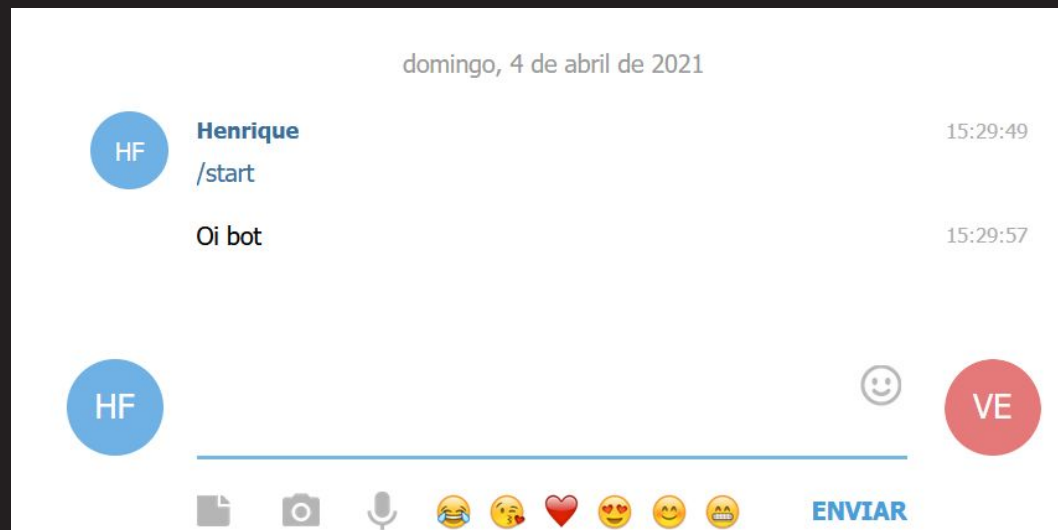
# Telegram + Node-RED

Vamos testar se estamos recebendo as mensagens do Telegram no Node-RED. Adicionamos um nó de Debug e damos deploy



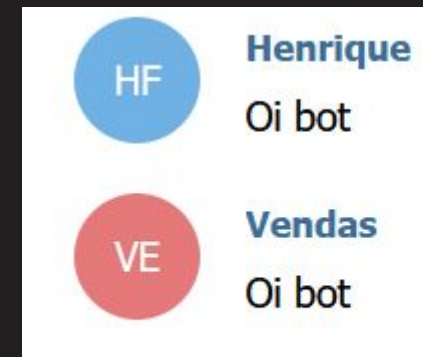
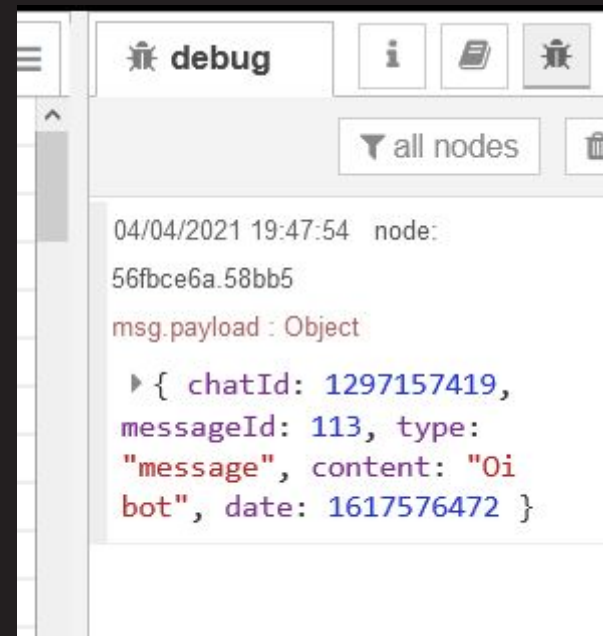
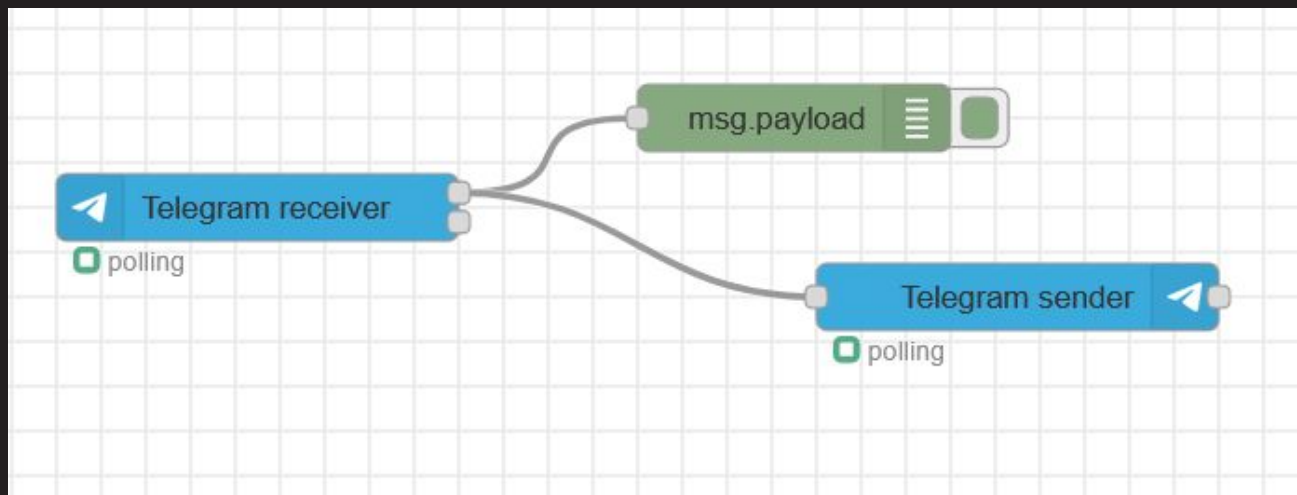
# Telegram + Node-RED

No Telegram Web (ou no seu celular), digite alguma mensagem para o bot. Não esqueça de conversar com o seu bot e não com o BotFather, clicando no link fornecido. Observe o menu de debug no Node-RED.



# Telegram + Node-RED

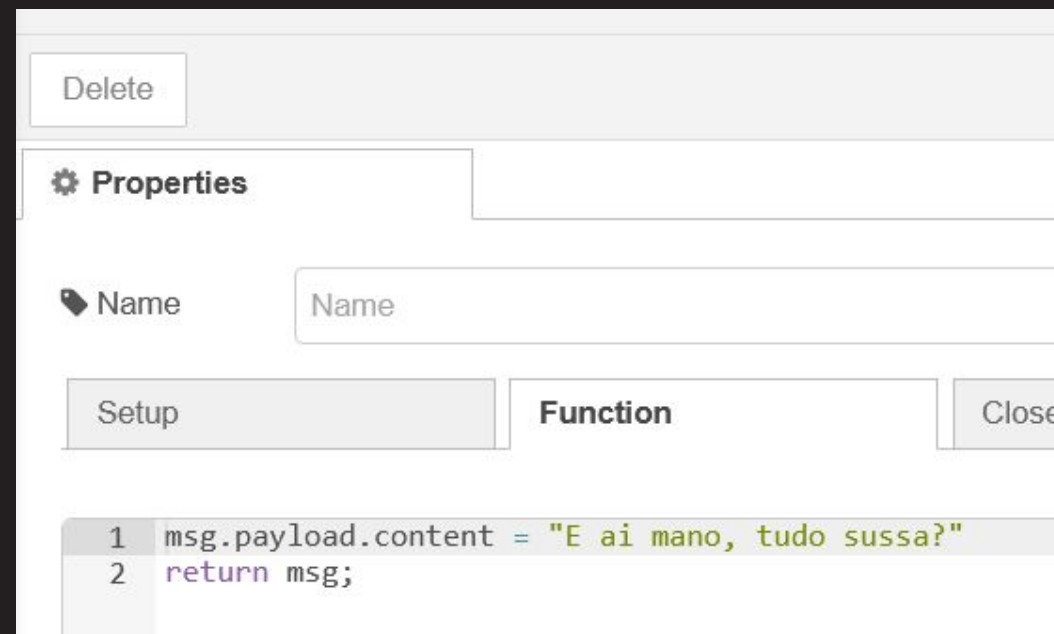
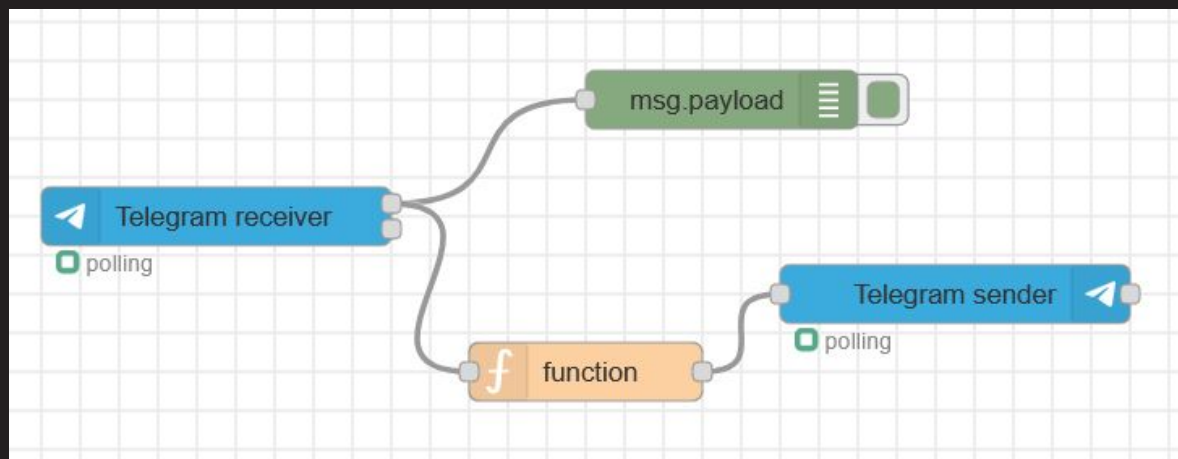
Vamos fazer o bot do Telegram repetir para nós o que dizemos para ele. Para isso, basta usar o **Telegram Sender** na frente do **Telegram Receiver** (não esqueça de configurar o **Token** no nó sender):





# Telegram + Node-RED

Podemos processar a mensagem recebida com um nó de function e depois enviá-la para o sender:



Resultado:

HF

Henrique  
oi bot

VE

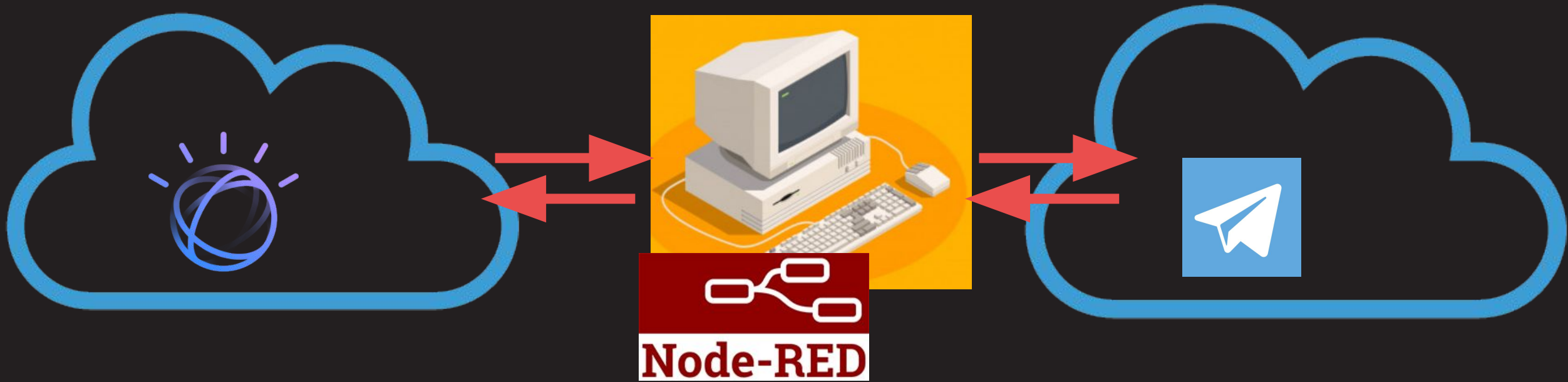
Vendas  
E aí mano, tudo sussa?

**Pergunta:** o que o bot responde para outra pergunta? Como podemos deixá-lo mais inteligente?

# Integrando nosso bot

Conectando o IBM Watson Assistant ao Telegram

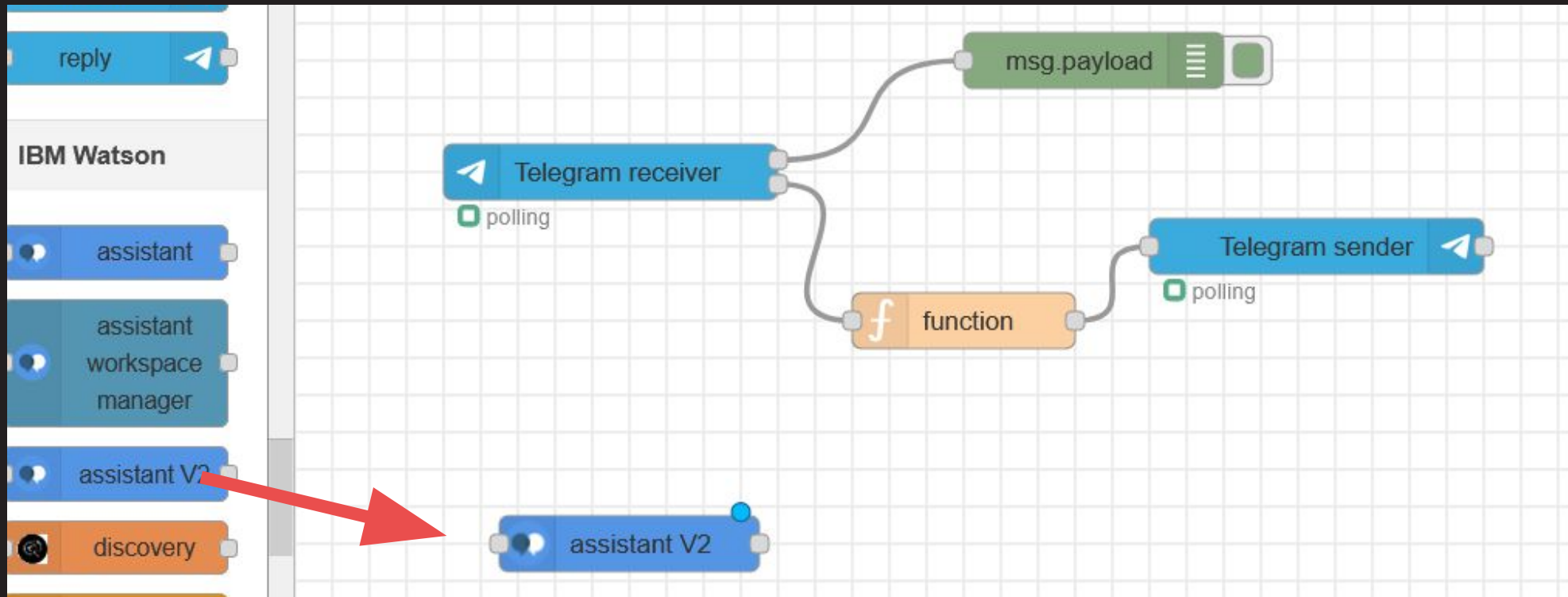
# Vamos usar o Watson Assistant para resolver o processamento inteligente da mensagem



- Vamos começar programando nossa integração localmente em nossas máquinas;
- O servidor Node-RED no nosso computador servirá de orquestrador;

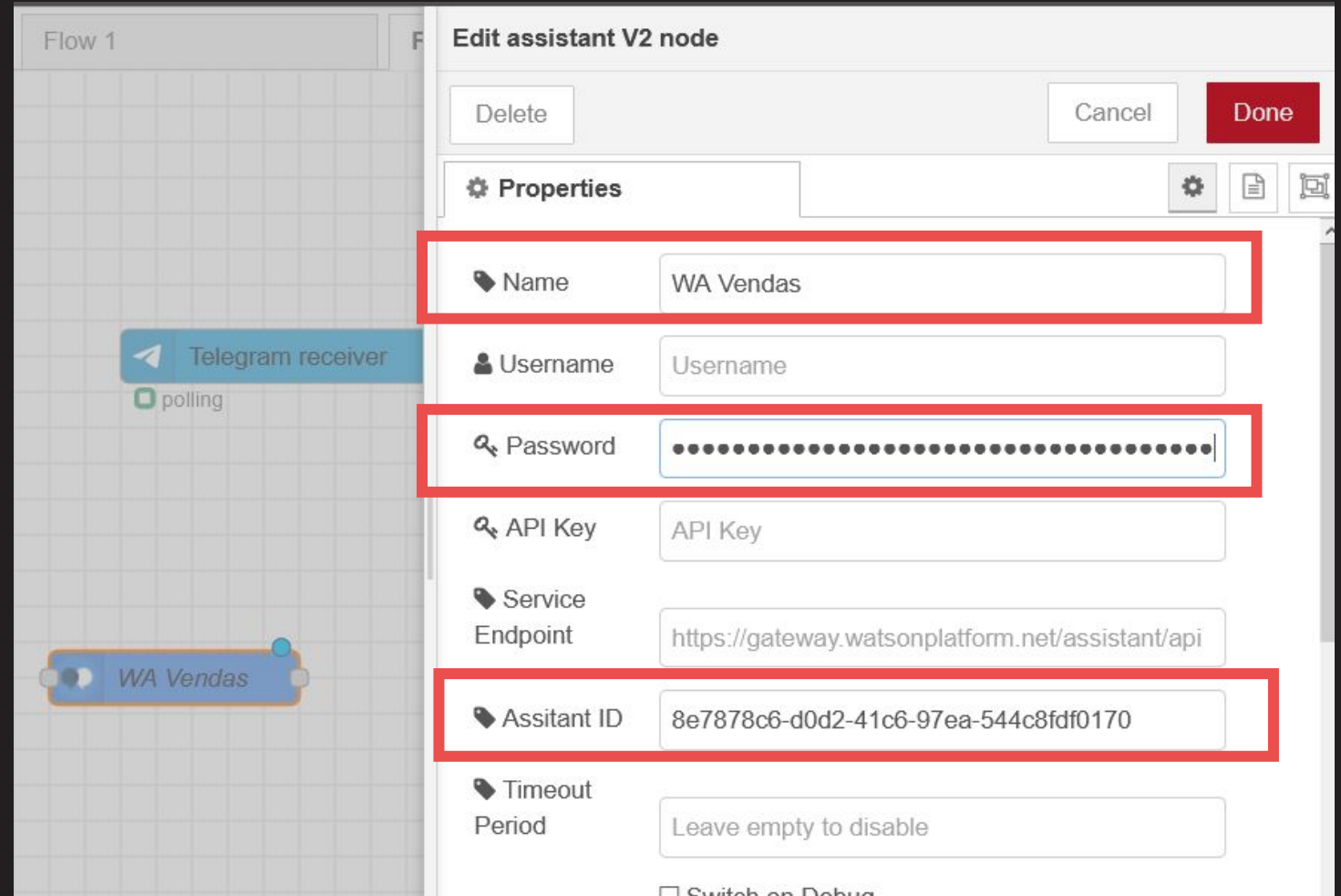
# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Vamos trocar o nosso nó de **function** pelo nó do **assistant V2**:



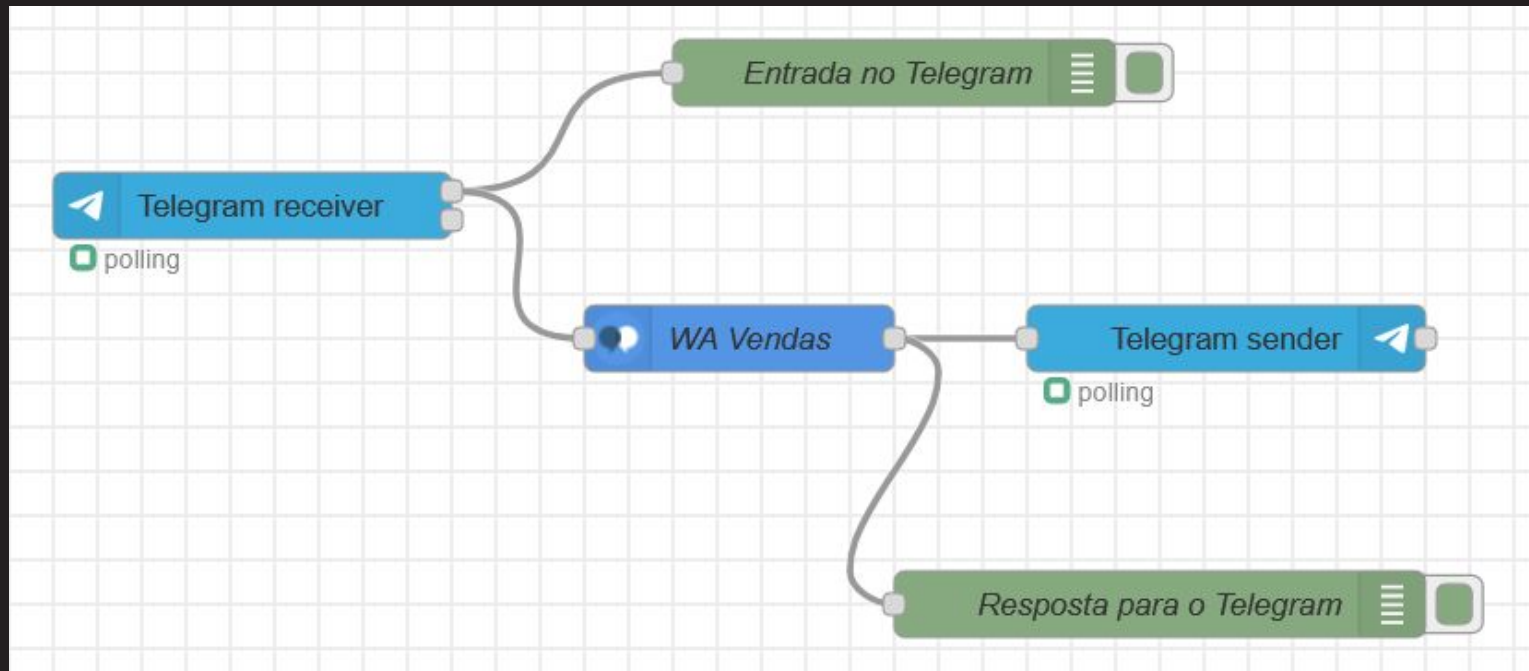
# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Como na aula anterior, precisamos colocar as credências do nosso serviço na nuvem da IBM dentro do nó;



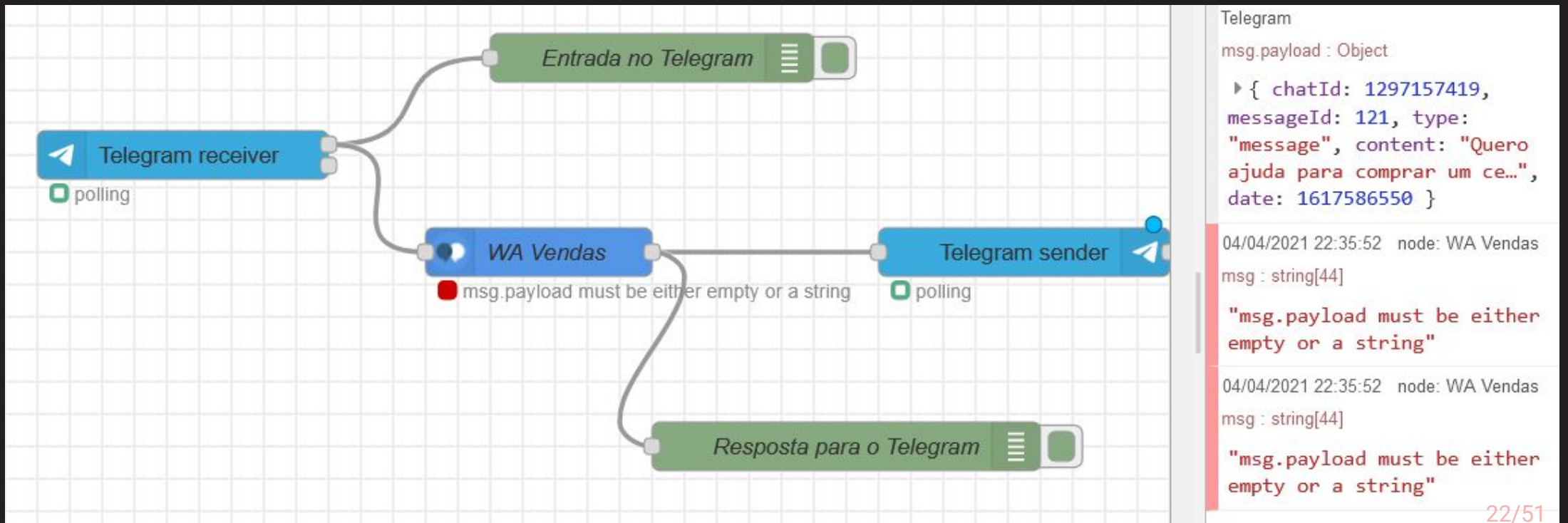
# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Adicionamos um nó de debug na saída do Watson Assistant;
- Teste o bot enviando uma mensagem pelo Telegram; O que aconteceu?



# Fazendo o Watson Assistant para processar as mensagens do Telegram

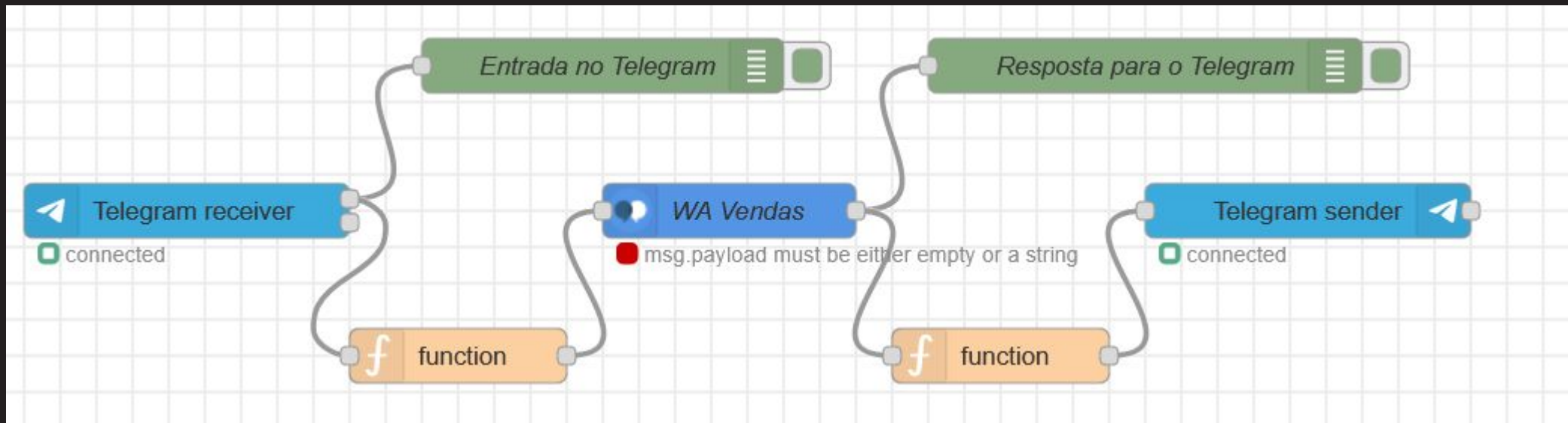
- A mensagem não está corretamente preparada; Devemos sempre nos atentar sobre o **padrão de mensagem que cada serviço/aplicativo gera e consome**;





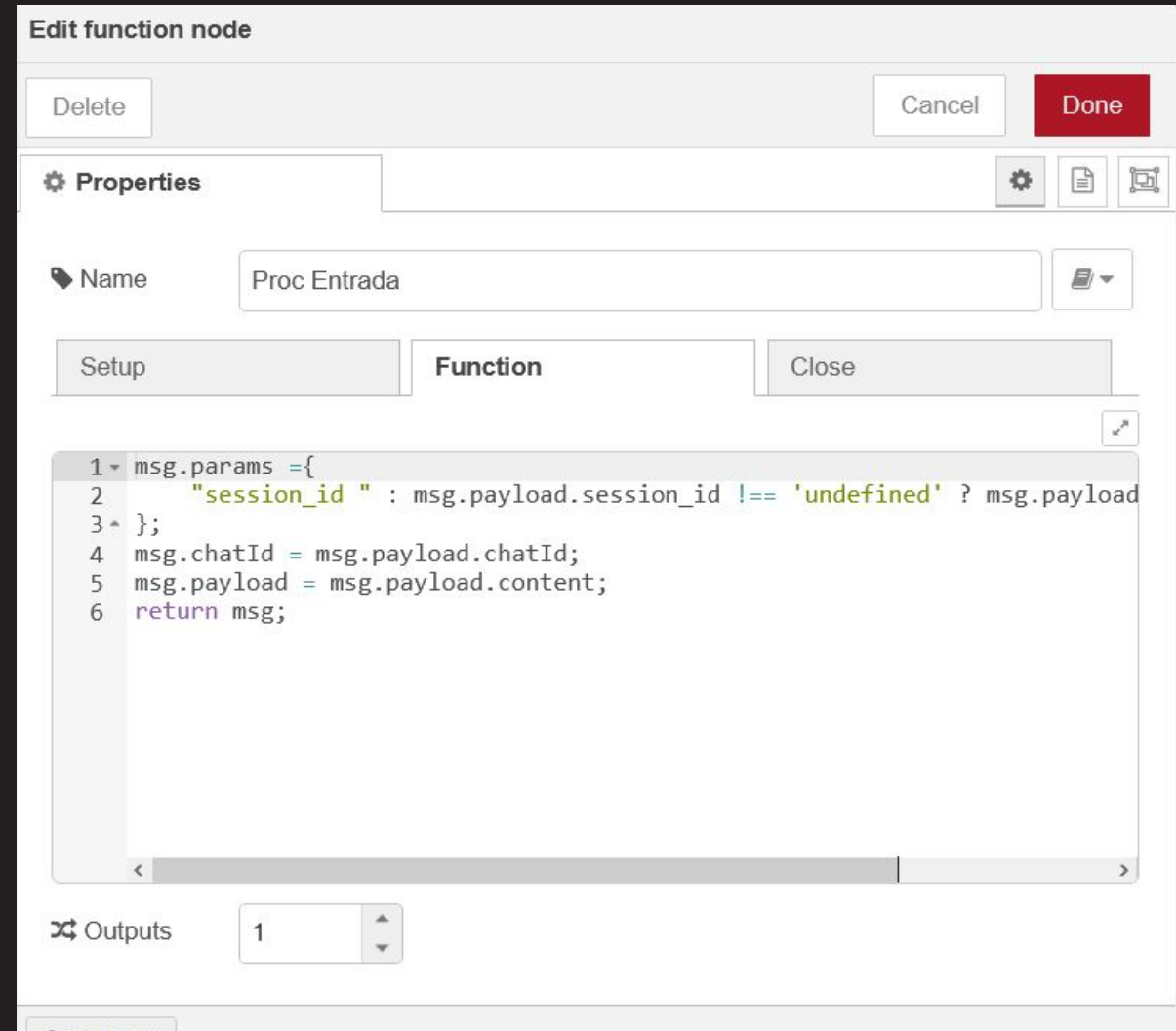
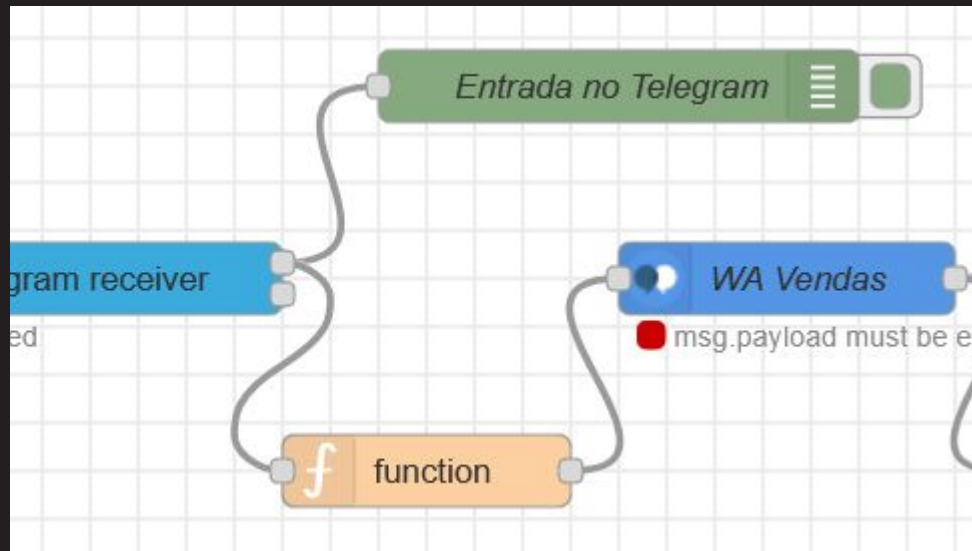
# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Vamos fazer como fizemos na primeira aula, adicionando dois nós de function, antes e depois do nó do WA:



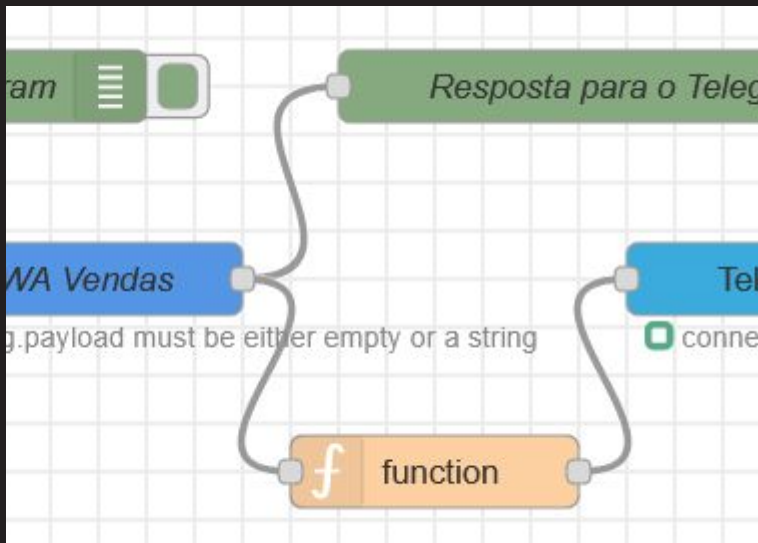
# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Agora vamos inserir o código de alteração da mensagem em cada nó de function; Primeiro o de Entrada



# Fazendo o Watson Assistant para processar as mensagens do Telegram

- Agora vamos inserir o código de alteração da mensagem em cada nó de function; Agora o de Saída (retorno)



Edit function node

Delete Cancel Done


Properties


Name Proc Retorno

Setup Function Close

```
1 msg.payload = {  
2   chatId : msg.chatId,  
3   type : "message",  
4   content : msg.payload.output.generic[0].text  
5 };  
6 return msg;
```

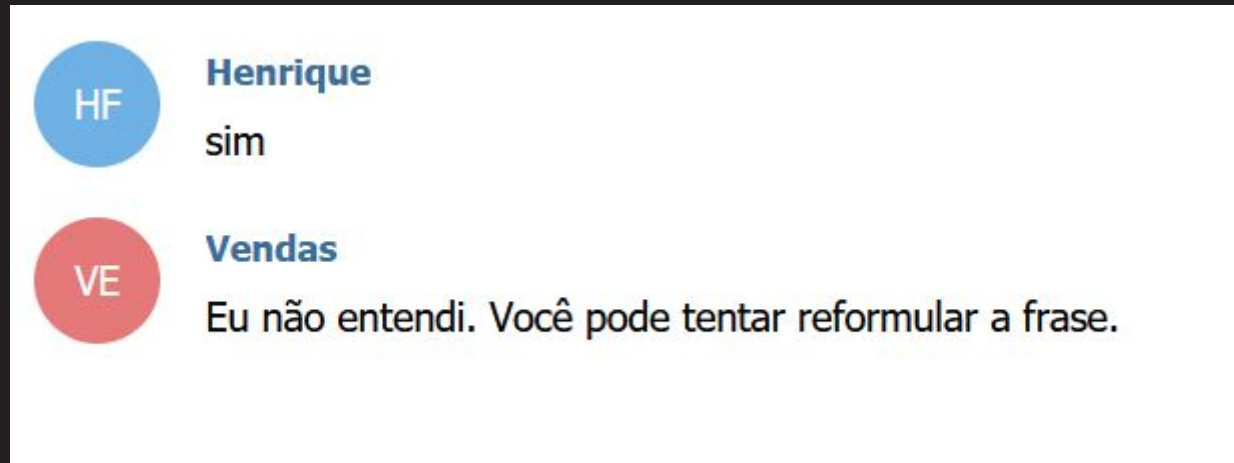
# Resultado

**Henrique**  
Quero ajuda para comprar um celular

**Vendas**  
Legal vou te ajudar. Irei fazer algumas perguntas para selecionar o melhor produto para você, ok?

O que acontece se for respondido positivamente a pergunta do bot?

# Resultado



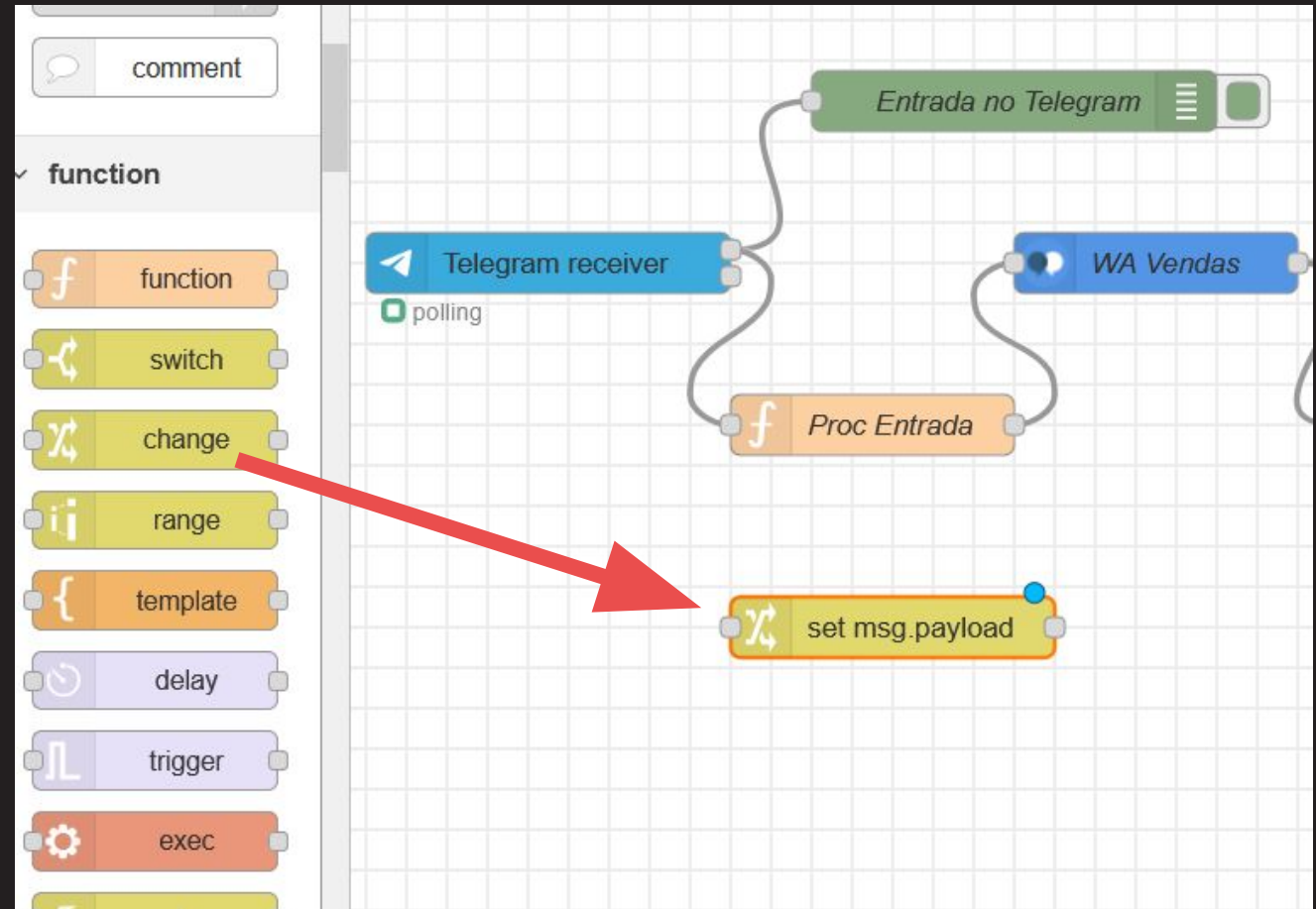
Isso acontece pois o **bot não sabe que está falando com a mesma pessoa**. Como nosso bot não está configurado para identificar um simples “sim” no início da árvore de diálogo, então ele irá cair no nó de **Em outros casos** (condição de `anything_else`)

# Integrando nosso bot 2

Mantendo o contexto de identificação de usuário

# Fluxo de integração: mantendo o ID do usuário

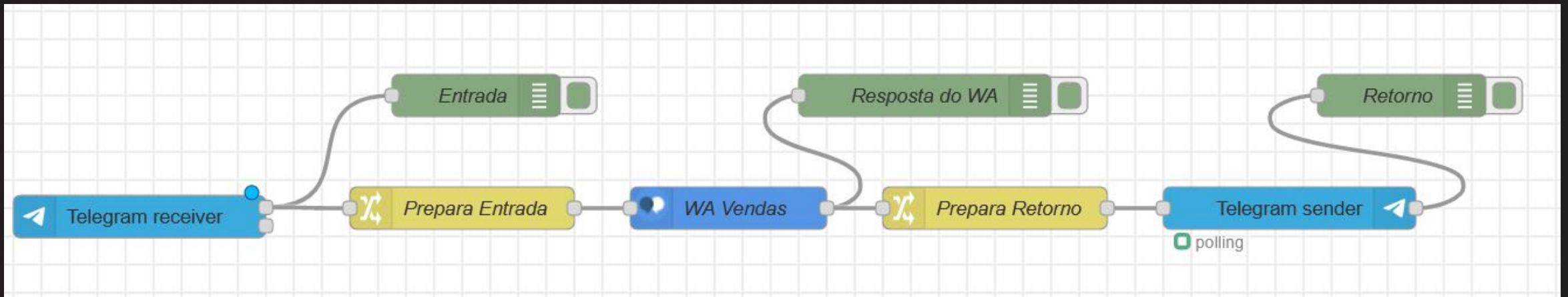
- Vamos seguir a seguinte abordagem: usar o ID do Telegram como ID do Watson Assistant. Para isso vamos usar um nó de function especial, o **change**;
- Vamos trocar os dois nós de function por novos dois nós de change;





# Fluxo de integração: mantendo o ID do usuário

- O fluxo ficar assim:



# Fluxo de integração: mantendo o ID do usuário

- Duple clique no primeiro nó;
- Vamos adicionar as seguintes regras de configuração:

Edit change node

Delete Cancel Done

⚙ Properties

📌 Name Prepara Entrada

📋 Rules

Set	▼ msg. params.session_id	×
to	▼ msg. payload.chatId	
Set	▼ msg. chatId	×
to	▼ msg. payload.chatId	
Set	▼ msg. payload	×
to	▼ msg. payload.content	

+ add

# Fluxo de integração: mantendo o ID do usuário

- Duple clique no segundo nó;
- Vamos adicionar as seguintes regras de configuração:

Edit change node

Delete Cancel Done

⚙ Properties

📁 Name Prepara Retorno

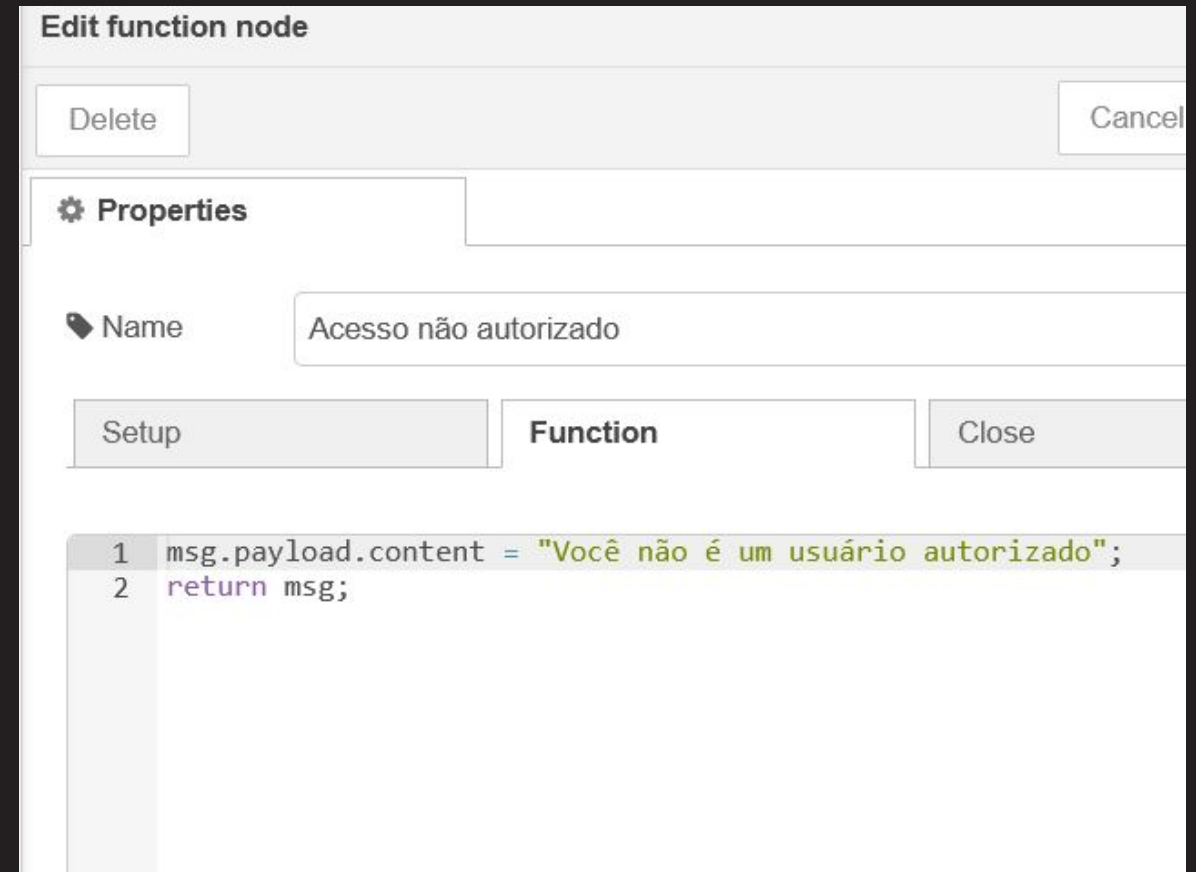
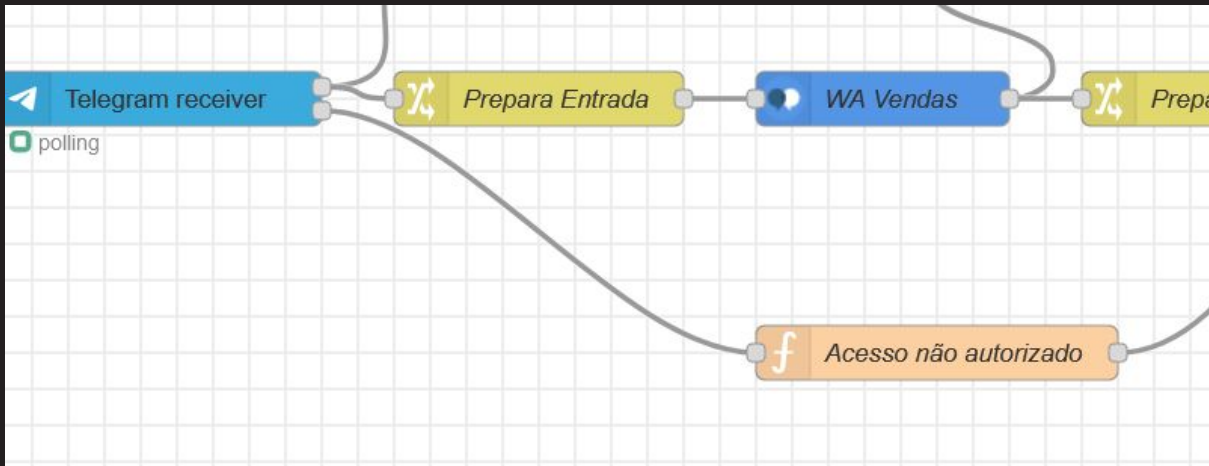
☰ Rules

≡	Set	▼ msg. payload.chatId	×
	to	▼ msg. chatId	
≡	Set	▼ msg. payload.type	×
	to	▼ a_z message	
≡	Set	▼ msg. payload.content	×
	to	▼ msg. payload.output.generic[0].text	

+ add

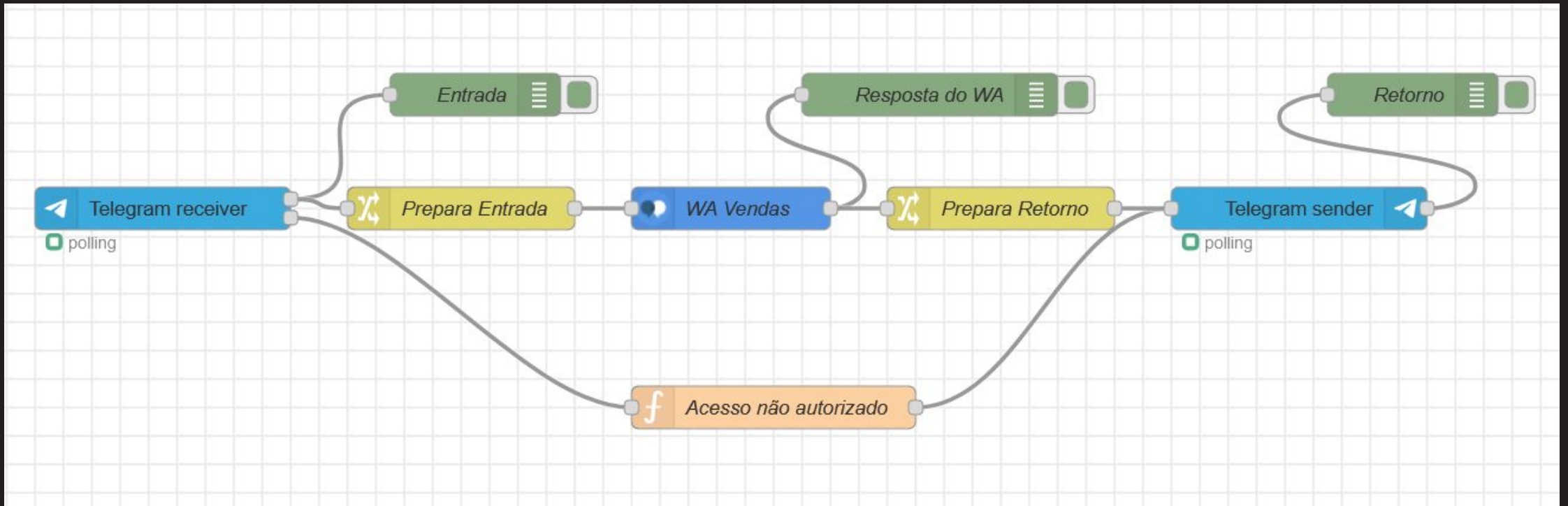
# Fluxo de integração: mantendo o ID do usuário

- Vamos adicionar um nó para negar acesso não autorizado:



# Fluxo de integração: mantendo o ID do usuário

- O fluxo final será assim:



# Testando...

- Salve o fluxo final!
- Observação: se você estiver tendo problemas, certifique-se que há apenas um fluxo com nós do Telegram;

The screenshot shows a Telegram chat interface with a white background. On the left, there are circular avatars: blue for Henrique (HF) and red for Vendas (VE). The messages are as follows:

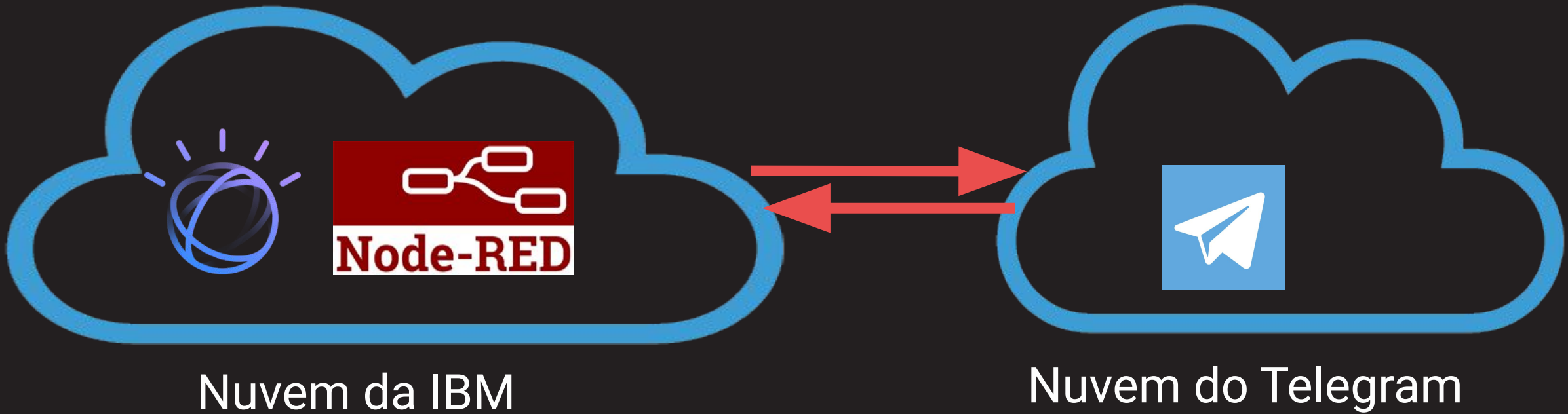
- Henrique (HF):** /start
- Vendas (VE):** Quero ajuda para comprar um celular
- Vendas (VE):** Legal vou te ajudar. Irei fazer algumas perguntas para selecionar o melhor produto para você, ok?
- Henrique (HF):** sim
- Vendas (VE):** Você tem preferência por alguma marca de celular?
- Henrique (HF):** da samsung
- Vendas (VE):** E você tem algum valor máximo para comprar esse produto?

# Integrando tudo na Nuvem

Subindo a integração local para a Nuvem da IBM














# Vamos colocar o orquestrador na nuvem



# Node-RED na Nuvem

- Existem algumas formas de acessar o serviço;
- Uma delas é:

Lista de recursos

Nome	Grupo	Localização	Produto	Status
<input type="text" value="Filtrar por nome ou endereço IP..."/>	<input type="text" value="Filtrar por grupo ou org ..."/>	<input type="text" value="Filtrar ..."/>	<input type="text" value="Filtrar ..."/>	<input type="text" value="Filtrar ..."/>
Dispositivos (0)				
VPC infrastructure (0)				
Clusters (0)				
Satellite (0)				
Apps do Cloud Foundry (1)				
 NodeRED-FIAP-1TDS	profhenrique.santos@fiap.com.br / dev	Dallas	SDK for Node.js™	 Iniciado
Serviços do Cloud Foundry (1)				
Serviços (4)				
 Continuous Delivery	Default	Dallas	Continuous Delivery	 Ativo
 Speech to Text-56	Default	Dallas	Speech to Text	 Ativo
 Watson Assistant-fiap	Default	Dallas	Watson Assistant	 Ativo
 nodered-fiap-1tds-cloudant-1617496646088	Default	Dallas	Cloudant	 Ativo
Armazenamento (0)				
Rede (0)				
Namespaces do Functions (0)				
Apps (1)				
 NodeRED-FIAP-1TDS	Default	Global	Cloud Application	—
Developer tools (1)				

# Node-RED na Nuvem

Lista de recursos /

## NodeRED-FIAP-1TDS

Em execução [Visite a URL do aplicativo](#) [Incluir tags](#)

[Detalhes](#) [Ações...](#)

- Introdução
- Visão geral**
- Tempo de execução
- Conexões
- Logs
- Gerenciamento de API
- Ajuste automático de escala

### Instâncias

Editar

Funcionamento

100%

1/1 instâncias estão em execução

Instâncias

1

MB de memória por instância

0 256 256

### Tempo de execução

SDK for Node.js™

256

Total de alocação de MB

0 MB ainda disponível

Usado Grátis

### Custo do tempo de execução

Os custos atual e estimado excluem os serviços conectados.

US\$ 0,00 0.00

Encargos atuais para o período de faturamento

US\$ 0,00 0,00

Total estimado para o período de faturamento  
1 de abr. de 2021 - 30 de abr. de 2021

### Conexões (1)

[nodered-fiap-1tds-cloudant-1617496646088-61095](#)

# Node-RED na Nuvem

- Não esqueça a senha se não terá que criar de novo!

Welcome to your new Node-RED instance on IBM Cloud

We know you're eager to start wiring up your flows, but first there are a couple of tasks you should do:

- Secure your Node-RED editor
- Learn how to install additional nodes

Progress indicator: 1 of 4 steps completed (Step 1 is active).

Navigation: Previous, **Next**



Secure your Node-RED editor

☒ Secure your editor so only authorised users can access it

Username: 1tds

Password: ..... (Strength: good)

☐ Allow anyone to view the editor, but not make any changes

☐ Not recommended: Allow anyone to access the editor and make changes

Progress indicator: 2 of 4 steps completed (Step 2 is active).

Navigation: Previous, **Next**

# Node-RED na Nuvem

## Learn how to install additional nodes

Node-RED provides a **huge catalog of extra nodes** you can install into the editor.

Many of these nodes can be installed directly from the editor's palette manager feature. However that can cause issues due to the limited memory of the default Node-RED starter application.

The *recommended approach* is to edit your application's `package.json` file to include the additional node modules and then redeploy the application. This can be done using the Continuous Delivery feature on the application's IBM Cloud dashboard.

For more information, follow [this tutorial on IBM Developer](#).



Previous

Next

## Finish the install

You have made the following selections:

- Secure your editor so only authorised users can access it

You can change these settings at any time by setting the following environment variables via the IBM Cloud console:

- `NODE_RED_USERNAME` - the username
- `NODE_RED_PASSWORD` - the password
- `NODE_RED_GUEST_ACCESS` - if set to 'true', allows anyone read-only access to the editor



Previous

Finish

# Node-RED na Nuvem

Node-RED on IBM Cloud

## Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform.

More information about Node-RED, including documentation, can be found at [nodered.org](https://nodered.org).

[Go to your Node-RED flow editor](#)

[Learn how to customise Node-RED](#)

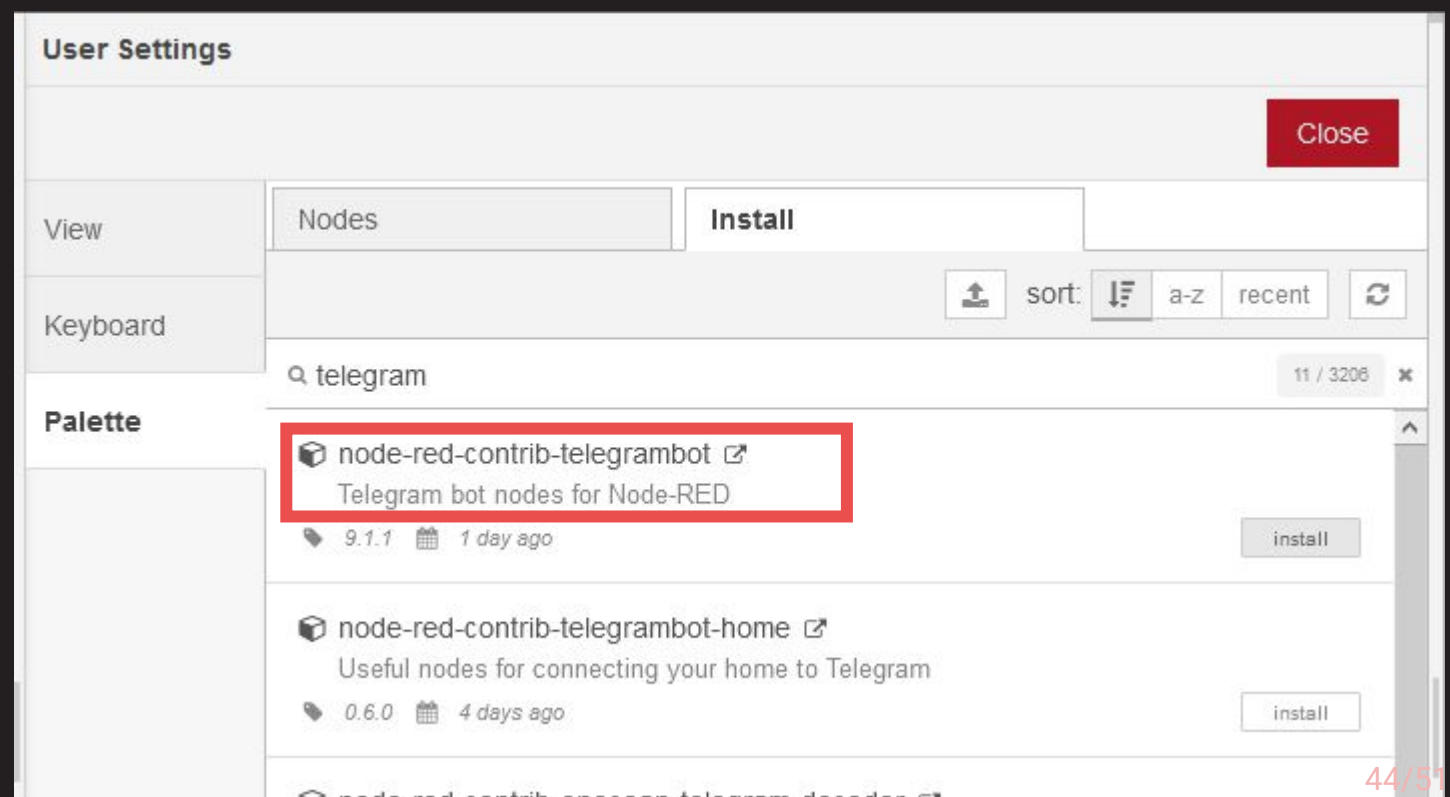
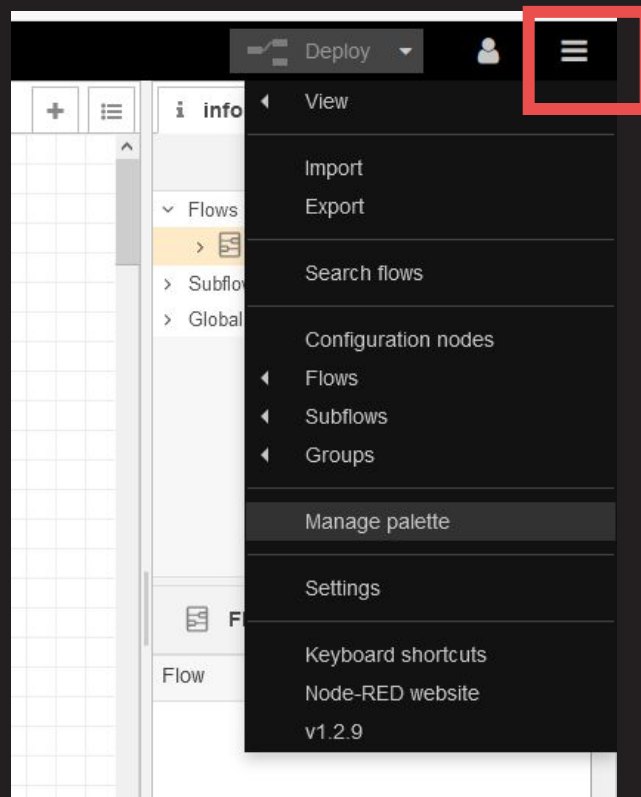
# Node-RED na Nuvem

- Irá abrir o ambiente de desenvolvimento. Perceba que não estamos mais no IP local.

The screenshot shows a web browser window with the Node-RED interface. The address bar is highlighted with a red rectangle, showing the URL: `https://nodered-fiap-1tds-manhahenrique.mybluemix.net/red/#flow/5cf19dfc.0dbc5c`. The Node-RED interface includes a left sidebar with a 'common' node palette containing nodes like 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link out', and 'comment'. The main workspace, titled 'Flow 1', contains a flow with an 'inject' node connected to a 'msg.payload' node. The right sidebar shows the 'info' tab with a search bar and a list of flows, including 'Flow 1'. At the bottom right, a status bar displays the flow ID '5cf19dfc.0dbc5c' and the page number '43/51'.

# Node-RED na Nuvem

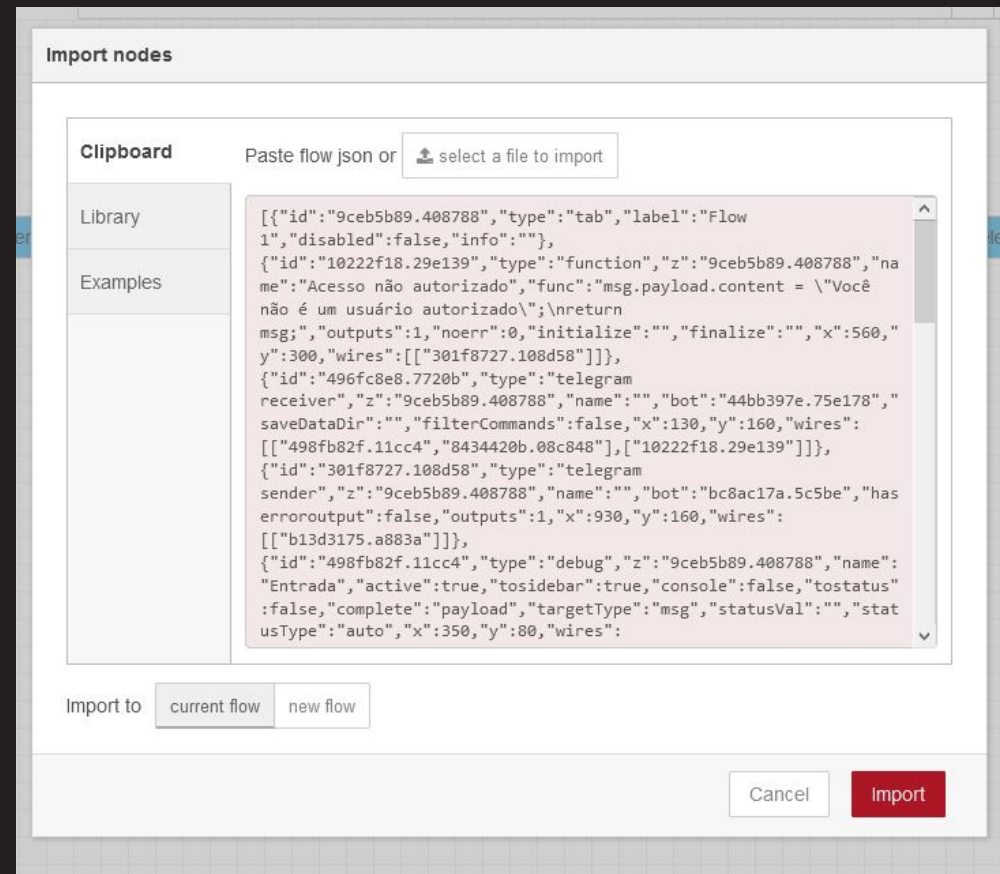
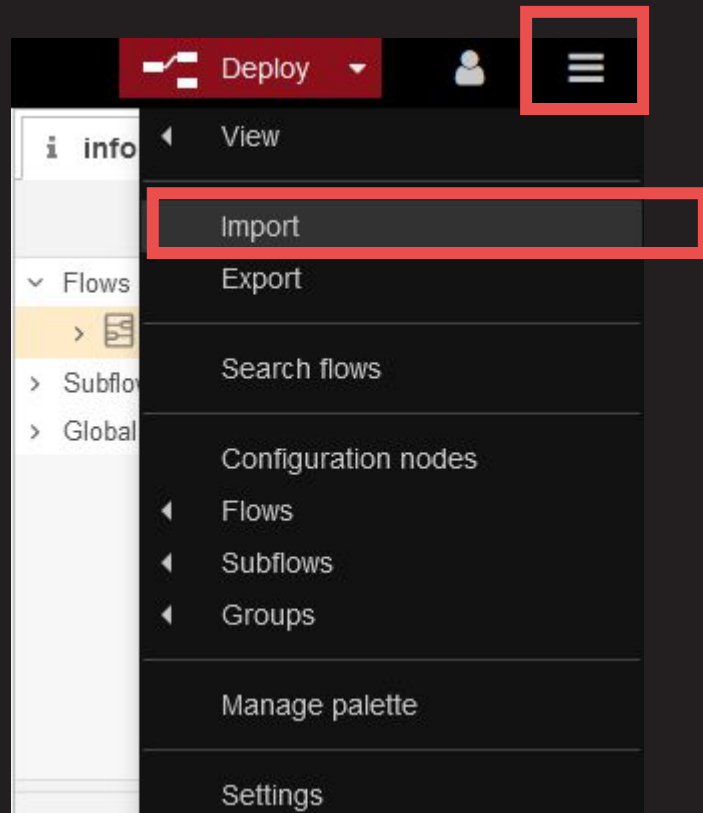
- Instalar a biblioteca do Telegram no nosso ambiente online (pode levar alguns minutos); Não precisamos instalar os nós do Watson Assistant pois eles já vem instalados por padrão.





# Node-RED na Nuvem

- Importando o fluxo que fizemos localmente:



# Node-RED na Nuvem

- Na sequência adicione as credências em cada nó de serviço (os dois do Telegram e do Watson Assistant);
- Clique em Deploy para rodar;
- Teste seu bot;

# Descanso

Do professor =D

# Exercícios

1. Mande um emoticon para seu bot. Observe a mensagem pelos nós de Debug para entender o que acontece com ela durante o fluxo de processamento. E se você mandar uma imagem? São criados novos atributos (parâmetros) dentro da mensagem do Telegram? Como o Watson Assistant lida com isso?
2. Explore as propriedades das mensagens para entender como elas são passadas de um nó para o outro no Node-RED. Tente fazer o rastreamento do ID de usuário ao longo do fluxo.

# Próximos Passos

O que veremos na próxima aula

# Na próxima aula...

- Introdução ao Reconhecimento de Fala;
- Ensinando o bot a ouvir;

**Copyright © 2022**

**Slides do Prof. Henrique Ferreira, com adaptações dos  
slides dos Prof. Marcelo Grave - FIAP**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).