

# AI & CHATBOT

Aula 05 – Introdução ao Node-RED e à  
Integração de Serviços

Prof. Érick

Slides Adaptados do Prof.  
Henrique Ferreira.

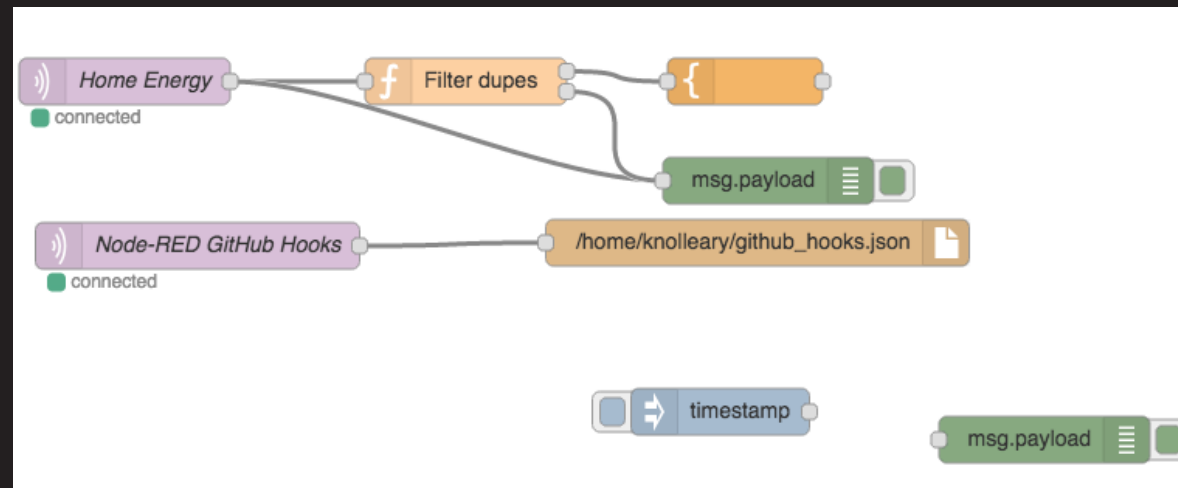
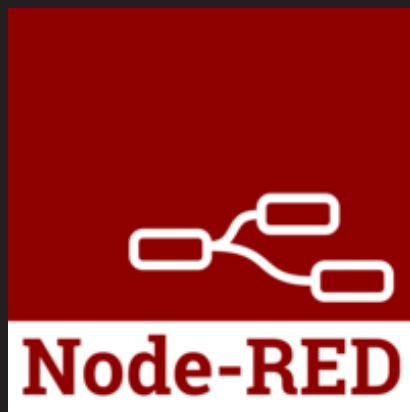
FIAP  
GRADUAÇÃO

# Node-RED 101

O que é o Node-RED e qual a sua aplicação?

# O que é o Node-RED?

- O Node-RED é uma ferramenta de **programação visual** com foco em aplicações de **integração de APIs para serviços online** e para Internet das Coisas (IoT).
- Ele é baseado em JavaScript e roda em ambiente Node.js
- Os “programas” feitos em Node-RED são chamados de fluxos (flows). Eles são salvos em formato JSON.



# Onde eu programo em Node-RED?

- Você pode programar em Node-RED em um ambiente em nuvem ou remoto (rodando no servidor de uma empresa ou de outra pessoa) ou no ambiente instalado na sua própria máquina local.
- Em ambos os casos, a interface do ambiente de programação funciona como um servidor local, que é acessado pelo navegador (browser).

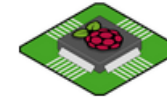
# Onde eu programo em Node-RED?

- Nas aulas nós vamos aprender tanto o desenvolvimento em nuvem quanto o desenvolvimento local.



## Running locally

Installing Node-RED on your local computer



## Raspberry Pi

Get started using our all-in-one install script for the mighty Raspberry Pi



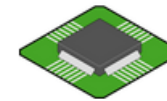
## Docker

Running Node-RED using Docker



## Install from git

Building Node-RED from source. Get the very latest development code and start contributing.



## BeagleBone Boards

Running Node-RED on BeagleBone boards



## Android

A bit experimental, but you can run on Android devices using Termux



## IBM Cloud

Deploying Node-RED from the IBM Cloud catalog in a couple of clicks



## AWS

Get started running on Elastic Beanstalk or EC2



## Microsoft Azure

Running on an Azure Virtual Machine instance

# Como eu programo em Node-RED?

Aba de nós

Aba de desenvolvimento do fluxo

Aba de informações

The screenshot displays the Node-RED web interface. On the left, the 'common' nodes panel is visible, containing various nodes like 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link out', 'comment', 'function', 'switch', 'change', 'range', and 'template'. A red arrow points from the text 'Aba de nós' to this panel. The central workspace, labeled 'Flow 1', is a large grid area for building flows. A red arrow points from the text 'Aba de desenvolvimento do fluxo' to this workspace. On the right, the 'Info' sidebar is open, showing a search bar and a list of flows, with 'Flow 1' selected. Below this, the 'Flow 1' details are shown, including the flow ID 'fa16a4d1.cd2648'. A red arrow points from the text 'Aba de informações' to the 'Info' sidebar. At the bottom of the sidebar, there is a message: 'You can confirm your changes in the node edit tray with `ctrl-enter` or cancel them with `ctrl-escape`'.

# Como eu programo em Node-RED?

**Tipos de nós:** existem basicamente três tipos de nós, os de entrada, de processamento e de saída.



**Mensagens:** no Node-RED chamamos um programa de fluxo (**Flow**). Este fluxo é criado através de nós que criam, recebem e processam, mensagens. Cada mensagem tem uma carga útil (**Payload**) que pode assumir diferentes valores e tipos.

# Como eu programo em Node-RED?

The image displays six vertical palettes of Node-RED nodes, each with a category header and a list of nodes. Each node is a rounded rectangle with an icon on the left and a label on the right. Some nodes have small circular ports on their ends.

- common**
  - inject
  - debug
  - complete
  - catch
  - status
  - link in
  - link out
  - comment
- function**
  - function
  - switch
  - change
  - range
  - template
  - delay
  - trigger
  - exec
  - rbe
- network**
  - mqtt in
  - mqtt out
  - http in
  - http response
  - http request
  - websocket in
  - websocket out
  - tcp in
  - tcp out
  - tcp request
  - udp in
  - udp out
- sequence**
  - split
  - join
  - sort
  - batch
- parser**
  - csv
  - html
  - json
  - xml
  - yaml
- storage**
  - file
  - file in
  - watch
  - tail



# Node-RED no meu computador

Instalando o Node-RED localmente




# Instalando o NodeJS

<https://nodejs.org/en/download/>

## Downloads

Latest LTS Version: **14.16.0** (includes npm 6.14.11)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 <b>Windows Installer</b> <small>node-v14.16.0-x64.msi</small>	 <b>macOS Installer</b> <small>node-v14.16.0.pkg</small>	 <b>Source Code</b> <small>node-v14.16.0.tar.gz</small>

### Windows Installer (.msi)

32-bit

64-bit

### Windows Binary (.zip)

32-bit

64-bit

### macOS Installer (.pkg)

64-bit

### macOS Binary (.tar.gz)

64-bit

### Linux Binaries (x64)

64-bit

### Linux Binaries (ARM)

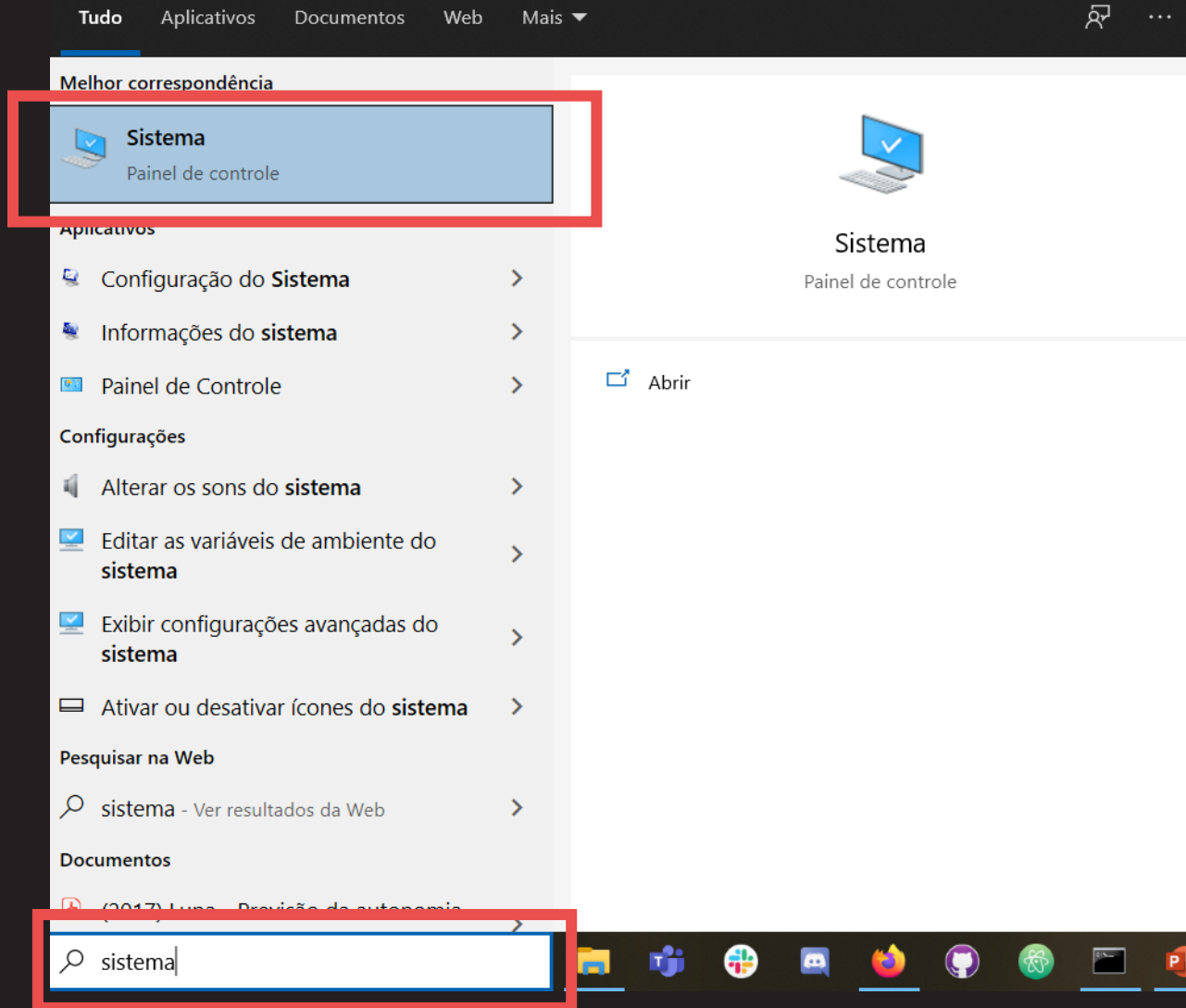
ARMv7

ARMv8

### Source Code

node-v14.16.0.tar.gz

# 32-bit ou 64-bit?



## Sobre

O computador está monitorado e protegido.

[Veja detalhes em Segurança do Windows](#)

## Especificações do dispositivo

### Inspiron 5584

Nome do dispositivo	DESKTOP-E3VFKJ5
Processador	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
RAM instalada	8,00 GB (utilizável: 7,87 GB)
ID do dispositivo	38C82385-F481-4101-AD05-F5A3AB9450BA
ID do Produto	00342-41396-69232-AAOEM

**Tipo de sistema** Sistema operacional de 64 bits, processador baseado em x64

**Caneta e toque** Nenhuma entrada à caneta ou por toque disponível para este vídeo

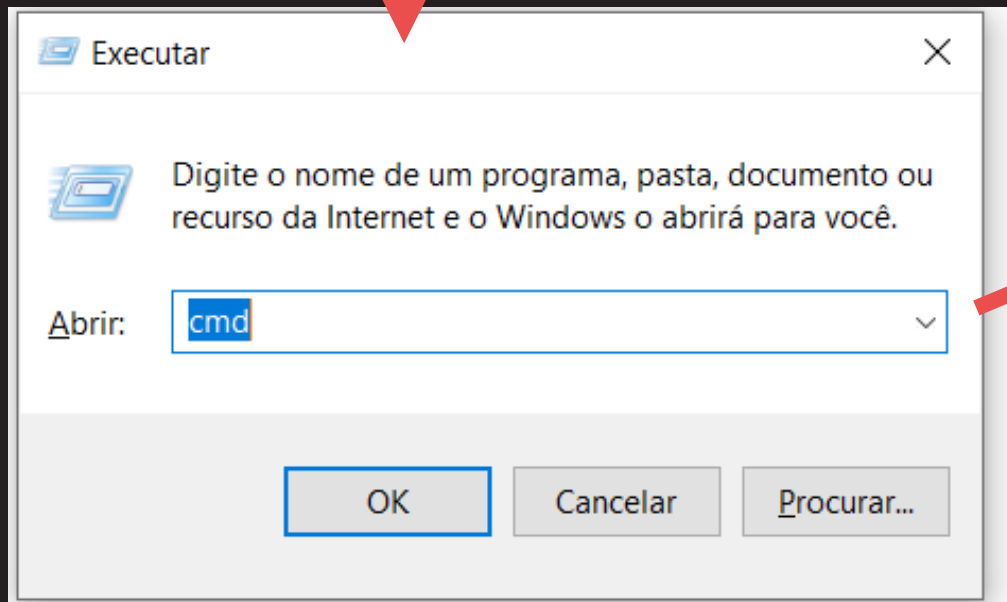
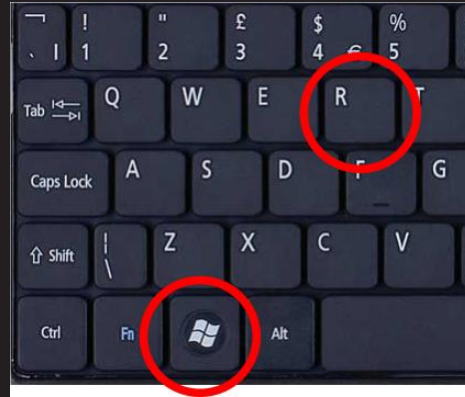
Copiar

Renomear este computador

## Especificações do Windows

# Verificando a instalação do Node JS

Tecla Windows+R



# Instalando o Node-RED

Instalando:

```
D:\node-red
```

```
npm install -g --unsafe-perm node-red
```

Executando:

```
D:\node-red
```

```
node-red
```

```
4 Aug 17:53:46 - [info]
```

```
Welcome to Node-RED
```

```
=====
```

```
4 Aug 17:53:46 - [info] Node-RED version: v1.1.2
```

```
4 Aug 17:53:46 - [info] Node.js version: v10.16.3
```

```
4 Aug 17:53:46 - [info] Windows_NT 10.0.18362 x64 LE
```

```
4 Aug 17:53:46 - [info] Loading palette nodes
```

```
4 Aug 17:53:48 - [info] Settings file : C:\Users\andre\.node-red\se
```

# Instalando o Node-RED

Uma vez executando, o Node-RED roda um **servidor web localmente** (na sua máquina). Como acessar esse servidor web? Usando o navegador, digite o **IP local** e a **porta** onde está rodando o Node-RED

```
node-red
28 Mar 20:38:19 - [warn]
-----
Your flow credentials file is encrypted using a system-generated key.
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
28 Mar 20:38:19 - [info] Starting flows
28 Mar 20:38:19 - [info] Started flows
28 Mar 20:38:19 - [info] Server now running at http://127.0.0.1:1880/
```

Copie e cole no seu navegador o número que aparece no seu terminal:

http://127.0.0.1:1880/

↑  
IP local

↑  
porta

# Abrindo o Node-RED

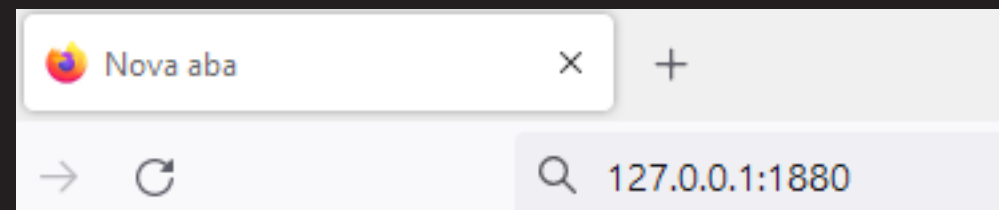
- Abra o terminal e suba o servidor Node-RED na sua máquina (não feche o terminal – ele precisa estar rodando para o servidor ficar online);
- Abra o seu navegador e coloque o endereço de IP local e porta;

Terminal



```
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]
```

Navegador



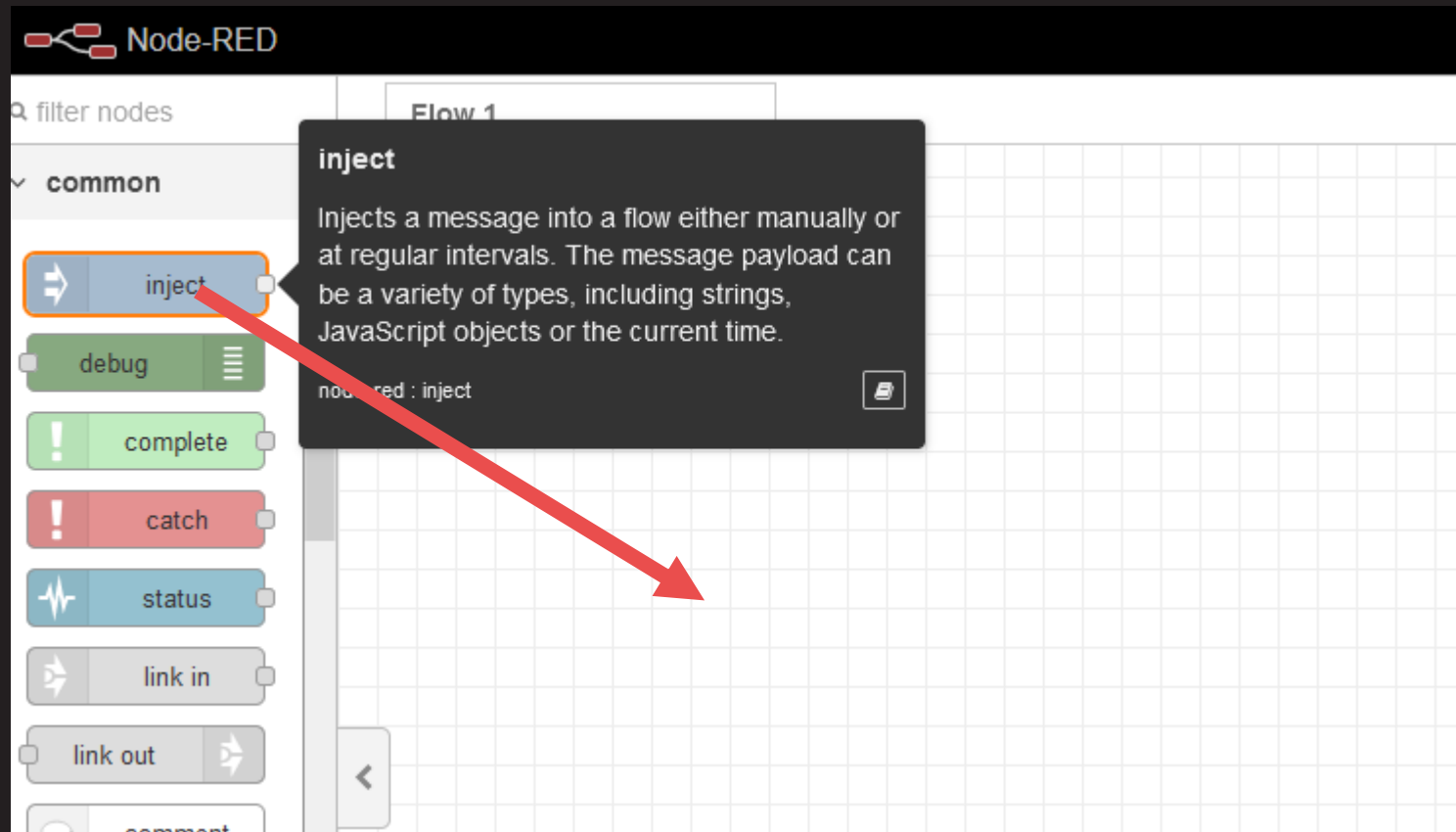
# Hello World!

O primeiro fluxo com Node-RED



# Primeiro fluxo com Node-RED

Segure e arraste o Nó de Inject da aba de nós para a área de desenvolvimento



# Primeiro fluxo com Node-RED

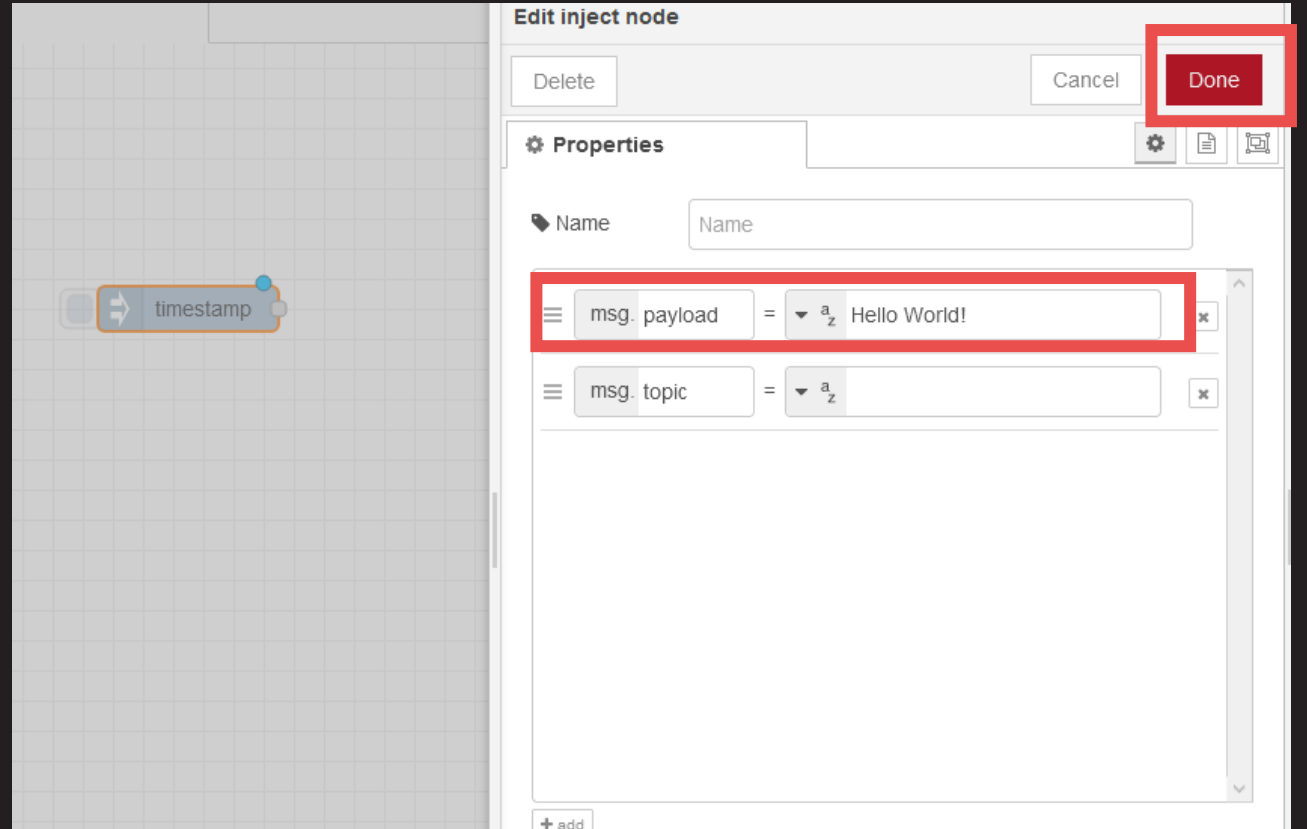
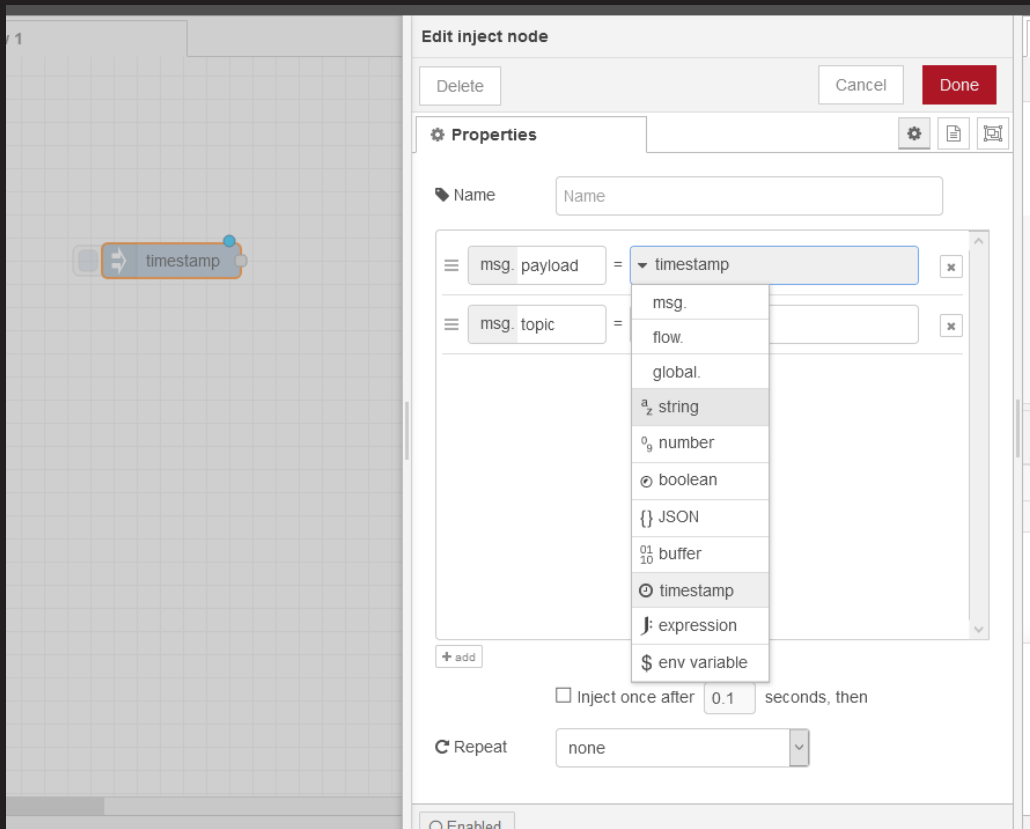
Duplo clique no nó:

Selecione string

The screenshot shows the Node-RED web interface. On the left, the 'common' node palette includes 'inject', 'debug', 'complete', 'catch', 'status', 'link in', 'link out', and 'comment'. The 'function' palette includes 'function', 'switch', 'change', 'range', and 'template'. The main workspace shows a 'Flow 1' canvas with a 'timestamp' node. The 'Edit inject node' dialog is open, displaying the 'Properties' tab. A red box highlights the 'msg.payload' field, which is set to 'timestamp'. A red arrow points from the text 'Selecione string' to this field. Below the 'msg.payload' field, the 'msg.topic' field is set to 'a\_z'. At the bottom of the dialog, there are options for 'Inject once after 0.1 seconds, then' and 'Repeat none'. On the right, the 'info' sidebar shows the 'timestamp' node details, including its ID '7fb78cd5.7750b4' and type 'inject'. A note at the bottom of the info sidebar states: 'Pressing enter will edit the first node in the current selection'.

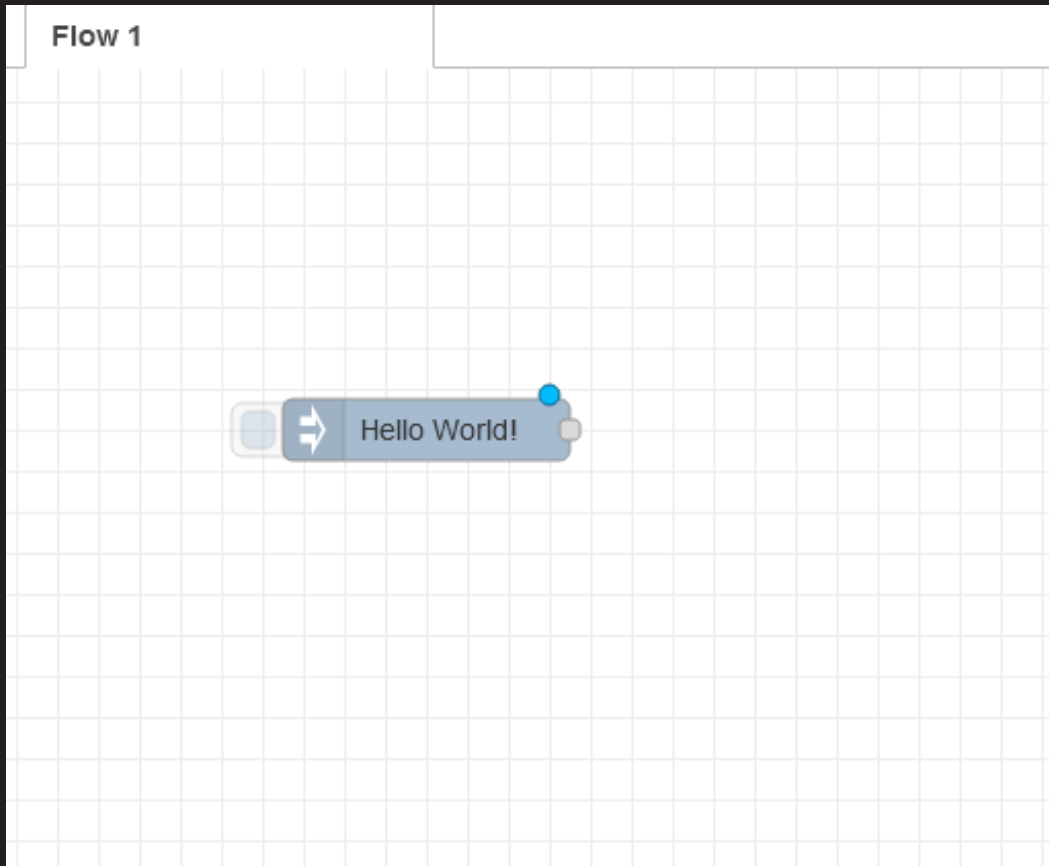
# Primeiro fluxo com Node-RED

Após selecionar string, digite a mensagem em payload, e clique em Done:



# Primeiro fluxo com Node-RED

O resultado será:



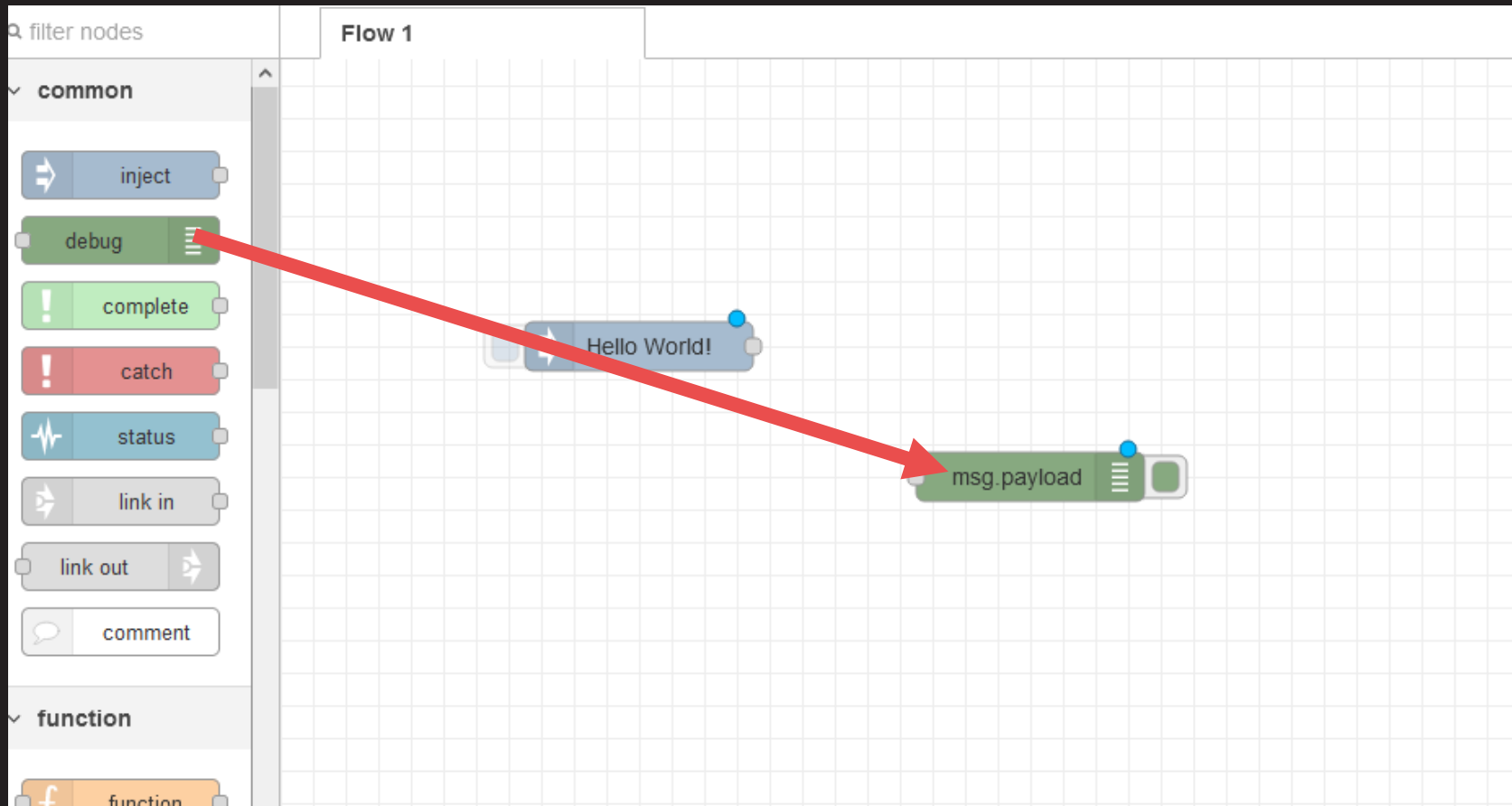
Este nós está “injetando” uma mensagem, do tipo string, com o valor “Hello World”;

String é um tipo de variável; Existem variáveis numéricas inteiras (int), variáveis numéricas decimais (float), vetores/arrays entre outros objetos. **Uma string é uma variável que armazena dentro dela valores de texto.**

No Node-RED os nós criam e processam mensagens que contêm um conteúdo (payload). Neste caso, nossa payload é uma string, ou seja, um pedaço de texto.

# Primeiro fluxo com Node-RED

Precisamos de um destino para a mensagem de Hello World. Vamos usar um nó de debug. Arraste e solte:



**Debug** é um termo usado sempre que você quiser testar um código para ver se ele funciona como você esperava que ele funcionasse. “Debugar” é ver e tirar bugs do seu código.

# Primeiro fluxo com Node-RED

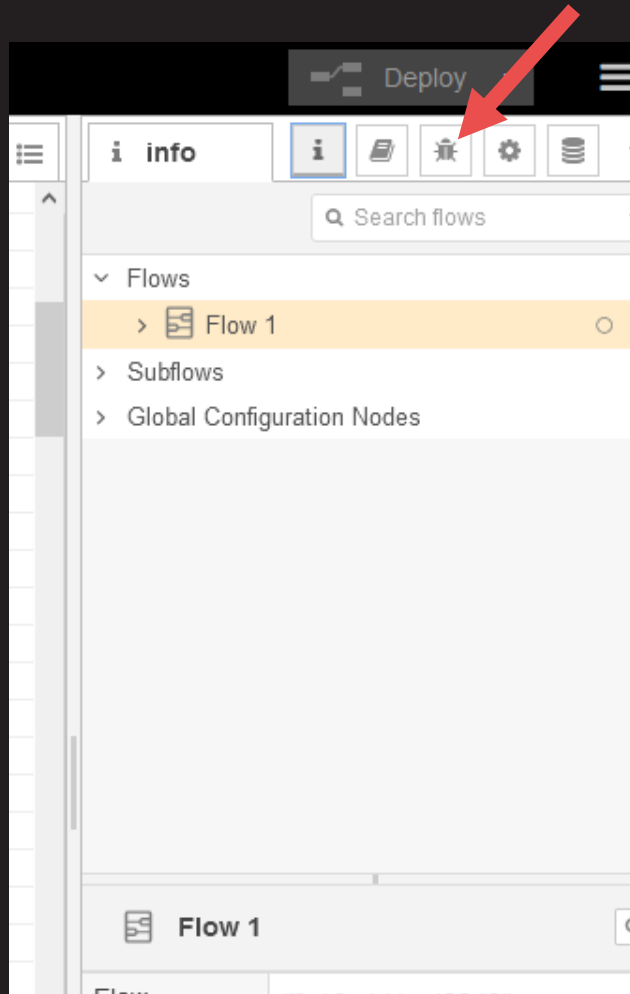
Agora conecte os dós nós.

E clique em deploy.

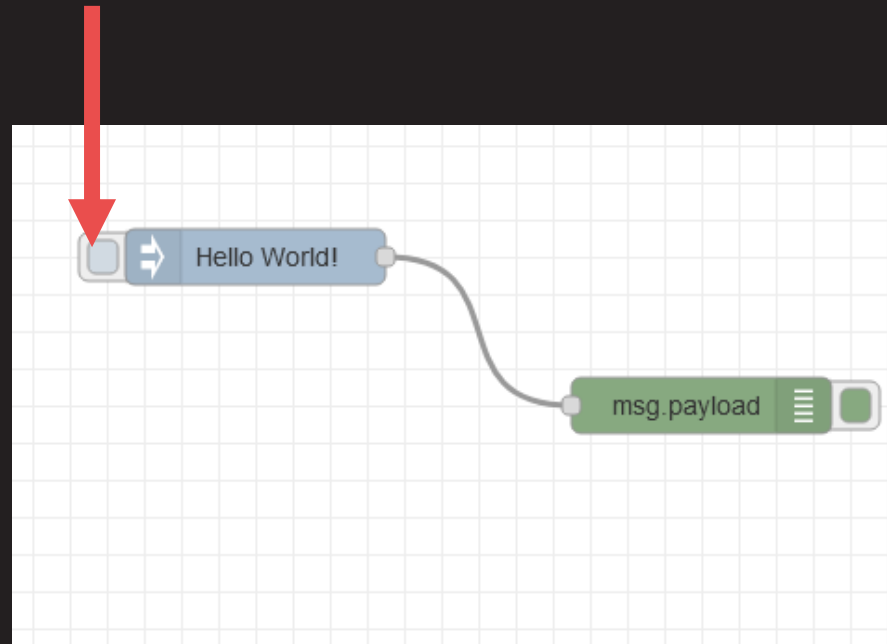
The screenshot displays the Node-RED web interface. The main workspace, titled 'Flow 1', features a grid background. A blue 'Hello World!' node is connected to a green 'msg.payload' node via a curved line. A red arrow points from the text 'Agora conecte os dós nós.' to the connection point between these two nodes. In the top right corner, a red 'Deploy' button is visible, with another red arrow pointing to it from the text 'E clique em deploy.'. The right sidebar contains an 'info' panel with a search bar and a list of flows, including 'Flow 1'.

# Primeiro fluxo com Node-RED

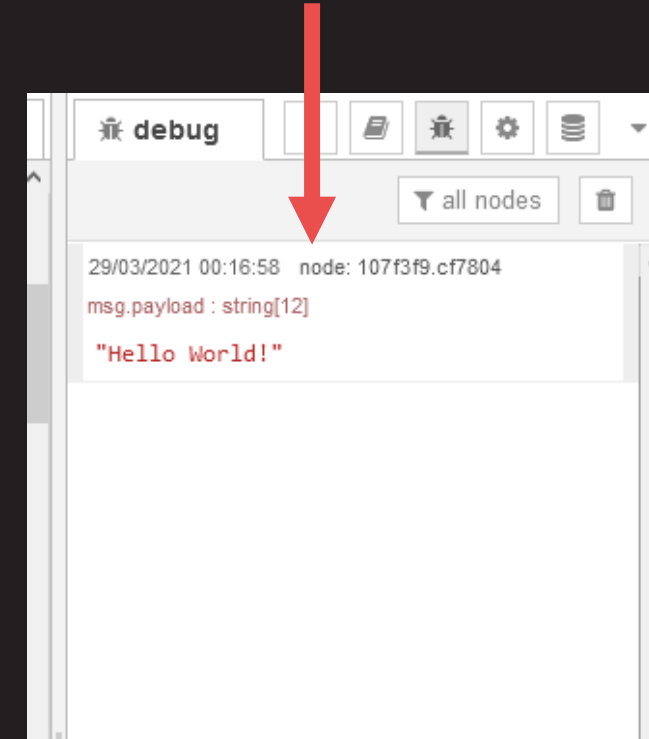
Abra o menu de debug



E injete e mensagem clicando no botão esquerdo do nó de Inject

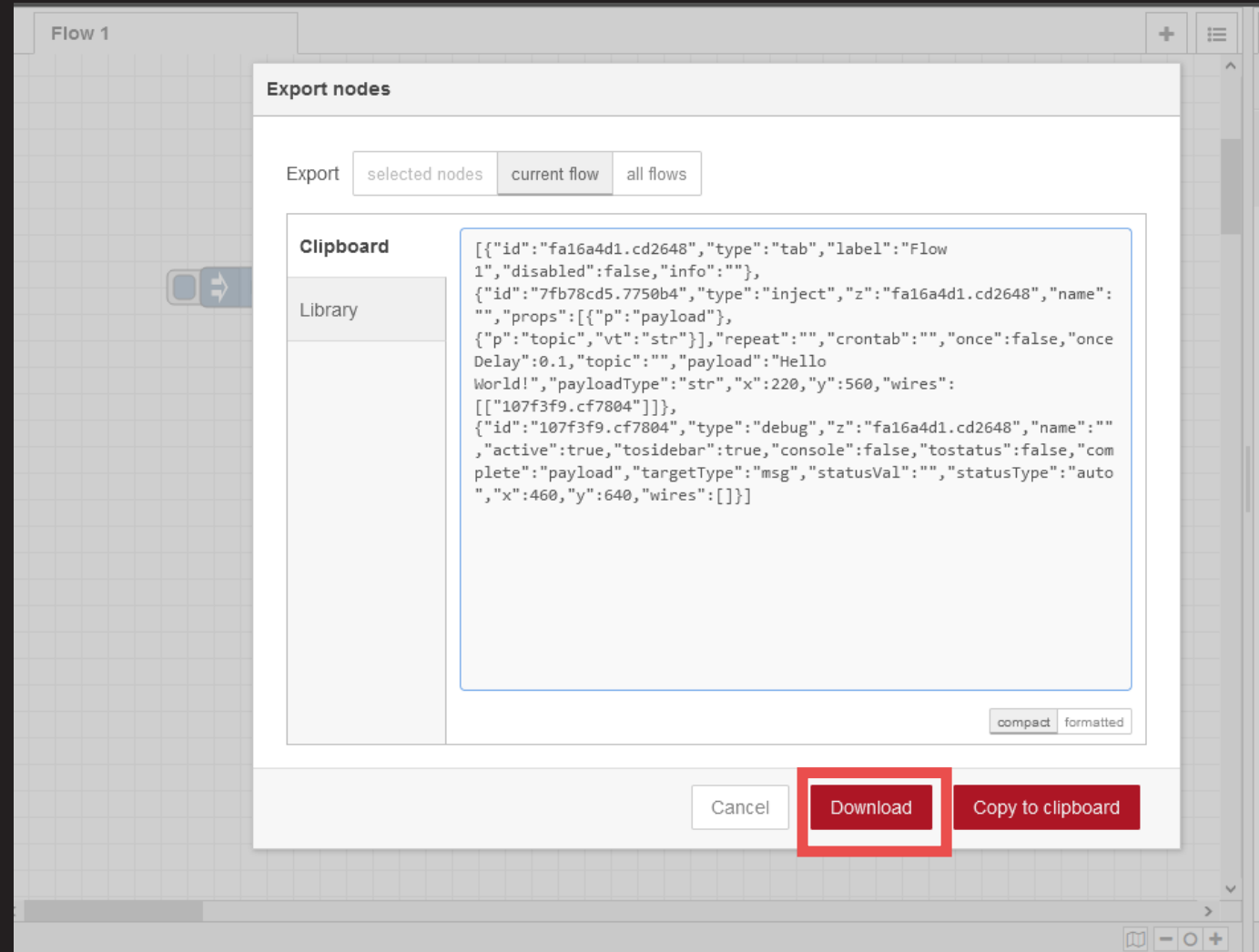
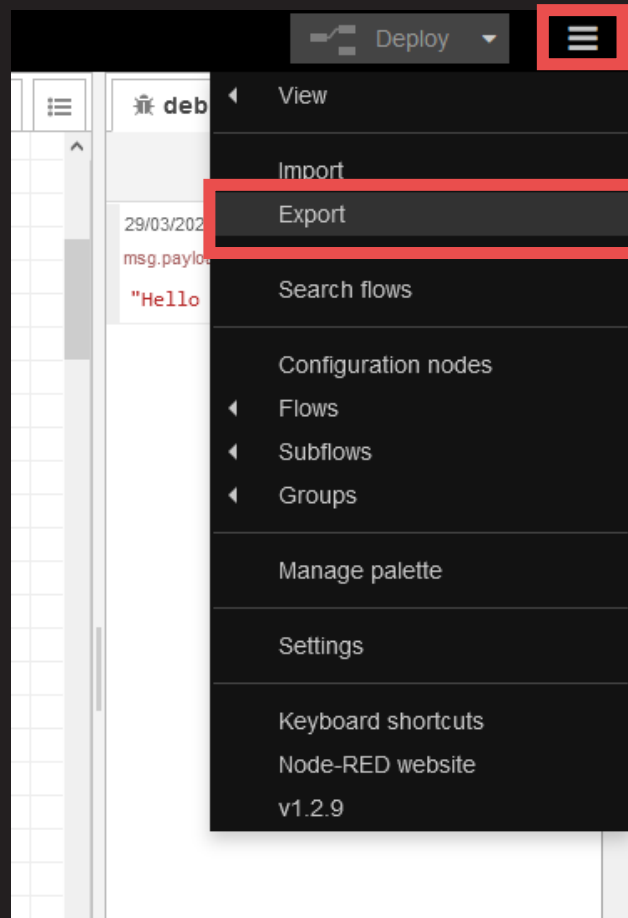


Resultado



# Primeiro fluxo com Node-RED

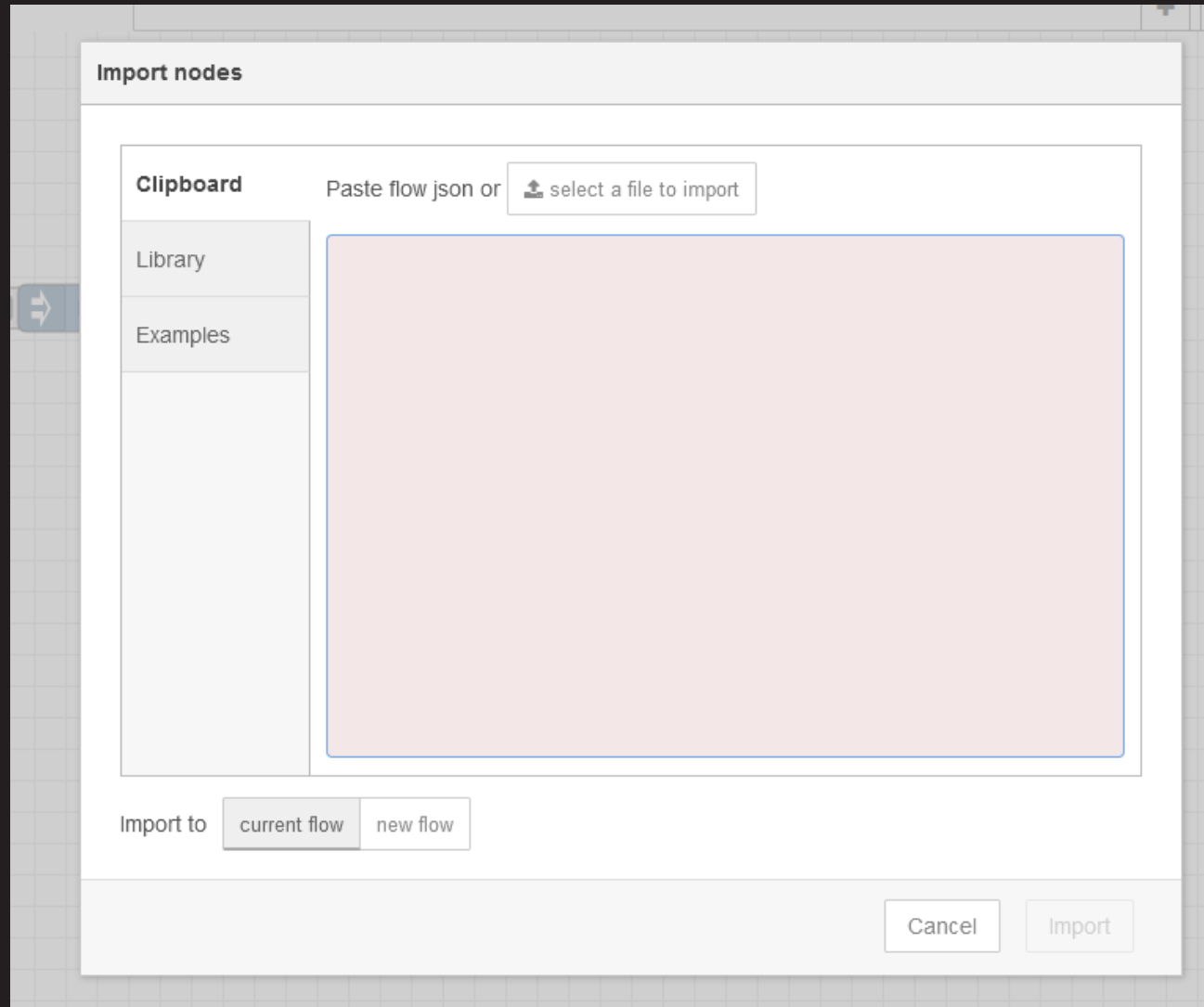
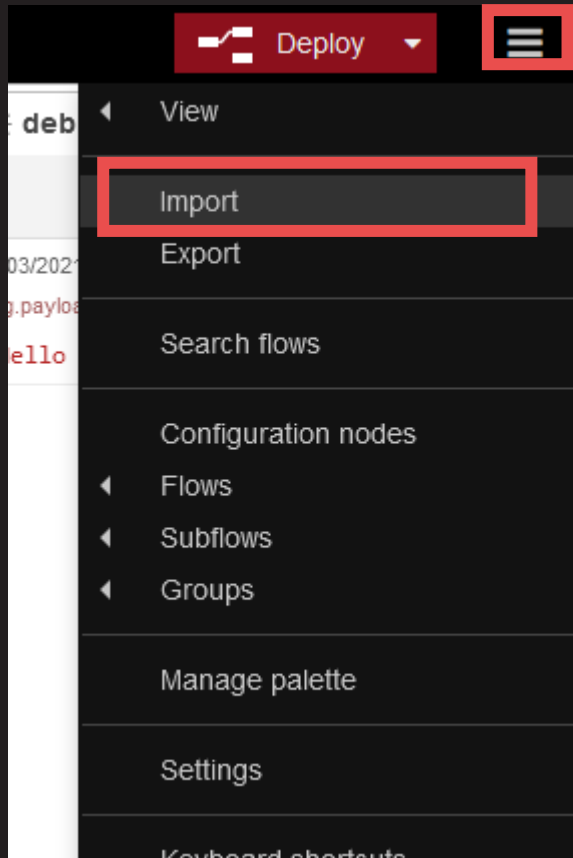
Salvando um fluxo:





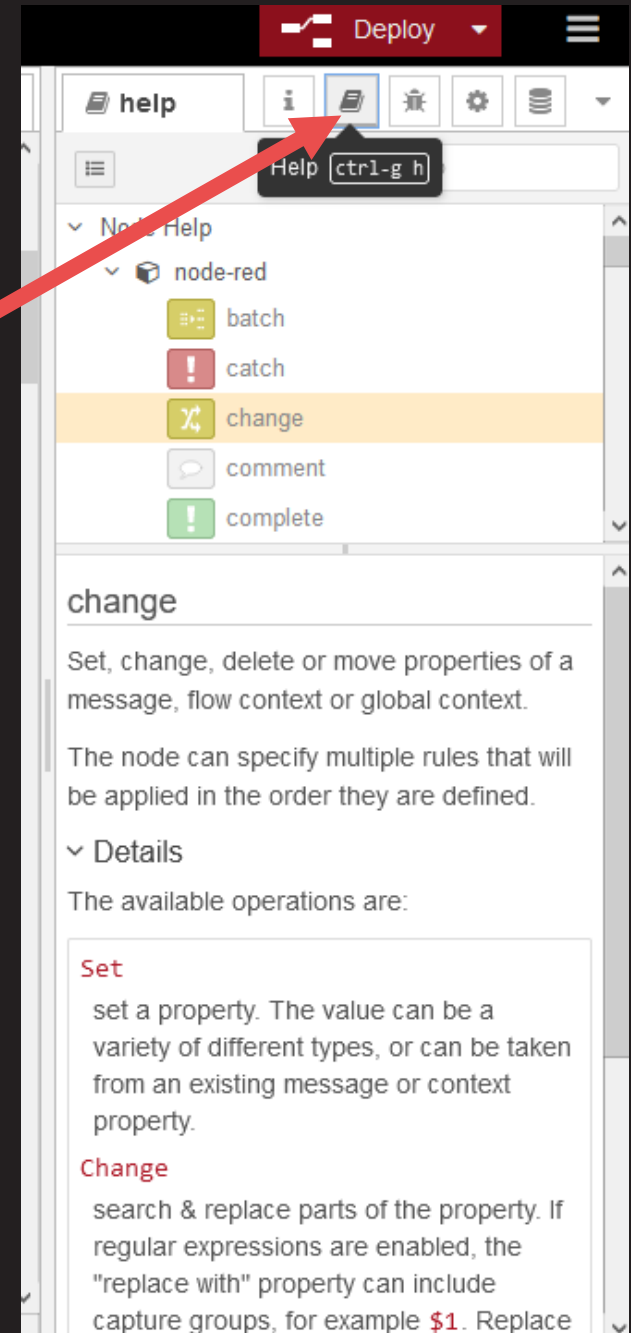
# Primeiro fluxo com Node-RED

Carregando um fluxo:



# Primeiro fluxo com Node-RED

Menu ajuda (help) vocês encontram as especificações do que cada tipo de nó faz. Explore para conhecer mais!

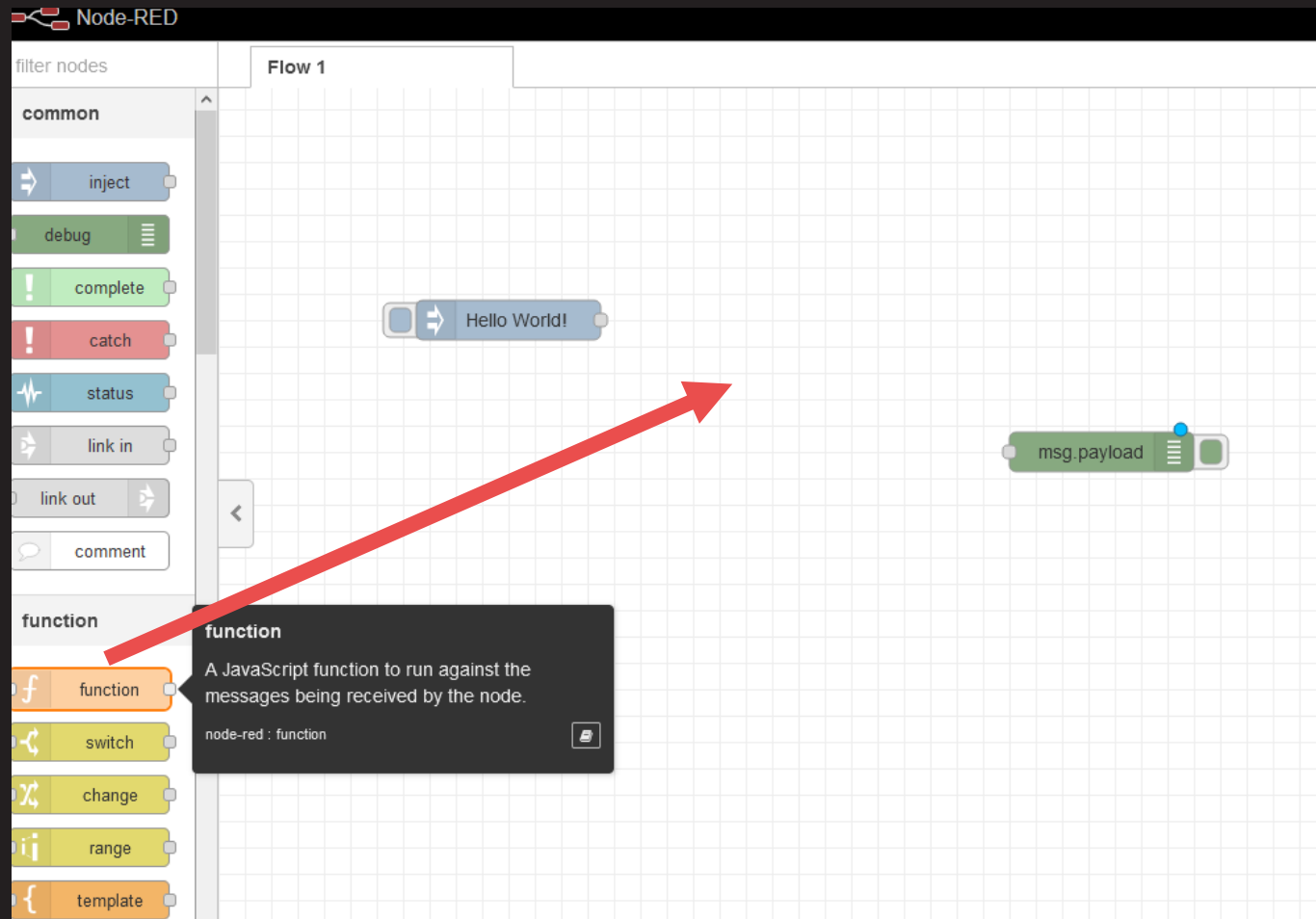


# Hello World 2.0

Manipulando a mensagem de Hello World

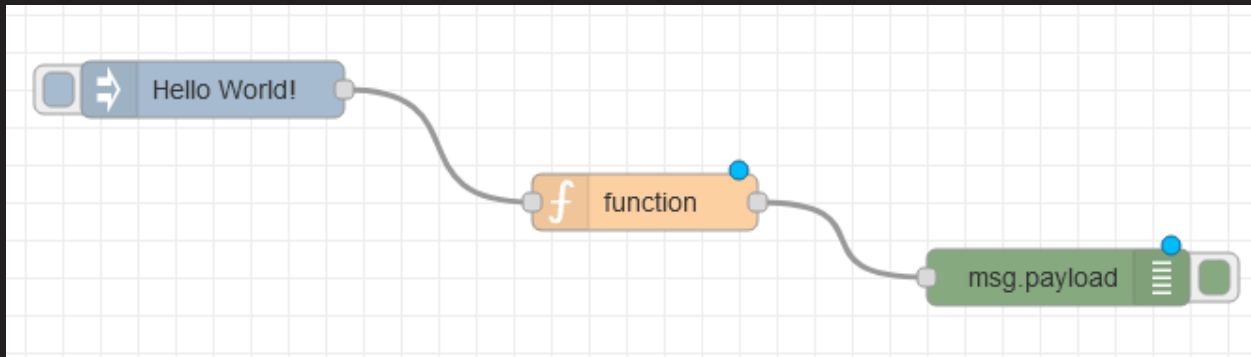
# Hello World 2.0

Arraste e solte o nó de fuction para o meio do fluxo 1:

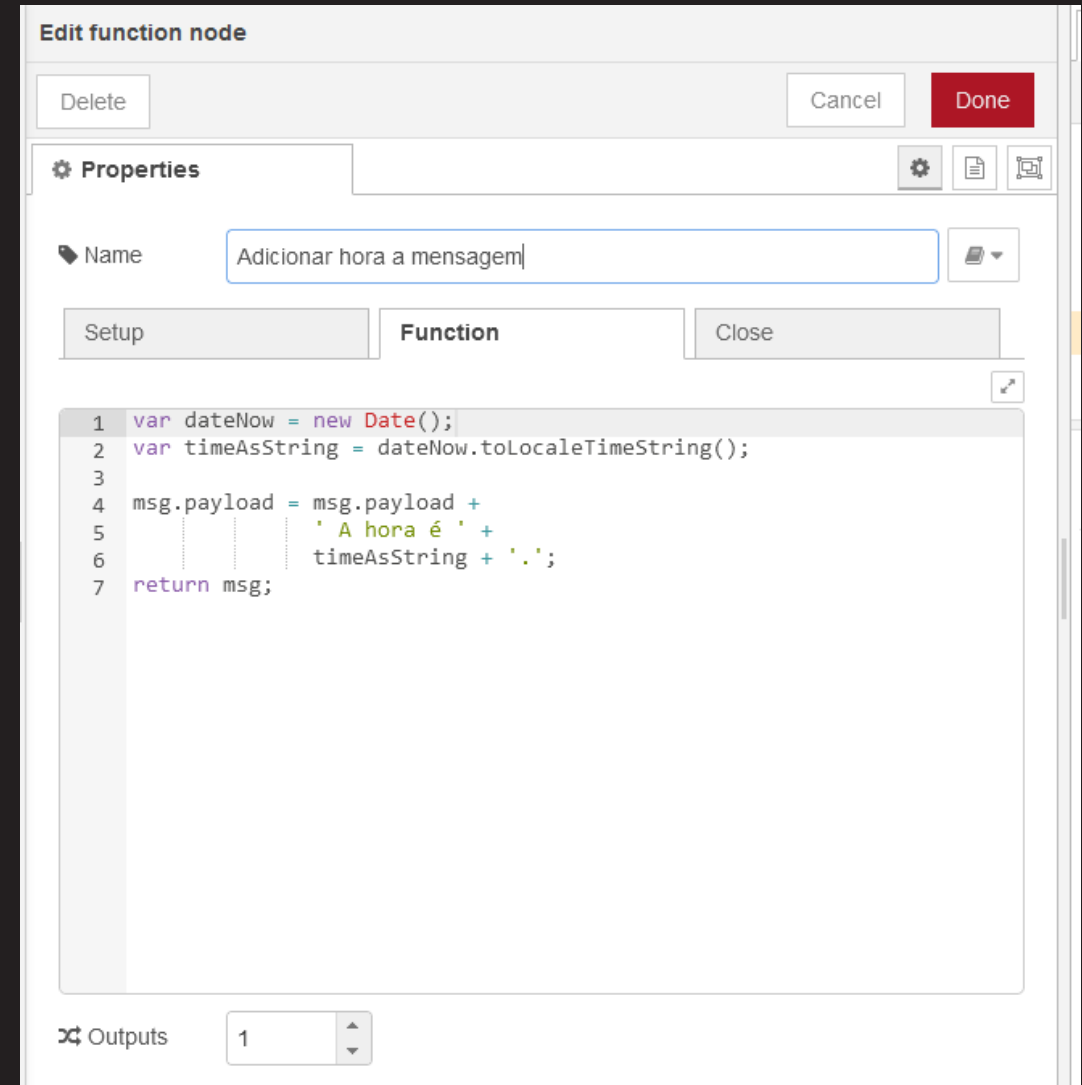


# Hello World 2.0

Arraste e solte o nó de function para o meio do fluxo 1:



Duplo clique no nó function. Vamos inserir código **JavaScript** para ele pegar a hora. O nome do nó será “Adicionar hora a Mensagem”. Ao terminar, clique em Done.



# Hello World 2.0

Clique em Deploy. Em seguida, com o menu de Debug aberto, aperte no botão de injeção ao lado do nó Hello World. Pronto!

The screenshot displays the Node-RED web interface. The main workspace shows a flow named 'Flow 1' with three nodes connected in sequence: a blue 'Hello World!' node, an orange 'Adicionar hora a mensagem' (Add time to message) function node, and a green 'msg.payload' output node. The right-hand sidebar contains a 'debug' panel with a search bar set to 'all nodes'. It lists two debug messages:

- 29/03/2021 00:16:58 node: 107f3f9.cf7804  
msg.payload : string[12]  
"Hello World!"
- 29/03/2021 00:38:04 node: 107f3f9.cf7804  
msg.payload : string[30]  
"Hello World! A hora é 0:38:03."

# Hello World 2.0

Vamos fazer ele repetir a injeção de mensagem , selecionando interval no campo Repeat, e configurando para 2 segundos.

O que acontece agora?

The screenshot shows the 'Edit inject node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Below is a 'Properties' tab with a 'Name' field. The main area contains two message entries: 'msg. payload' with value 'Hello World!' and 'msg. topic' with value 'a\_z'. At the bottom, there is a checkbox for 'Inject once after 0.1 seconds, then' and a 'Repeat' section. The 'Repeat' section has a dropdown set to 'interval', and 'every 2 seconds'.

Edit inject node

Delete Cancel Done

Properties

Name

msg. payload = a\_z Hello World!

msg. topic = a\_z

+ add

☐ Inject once after 0.1 seconds, then

Repeat interval

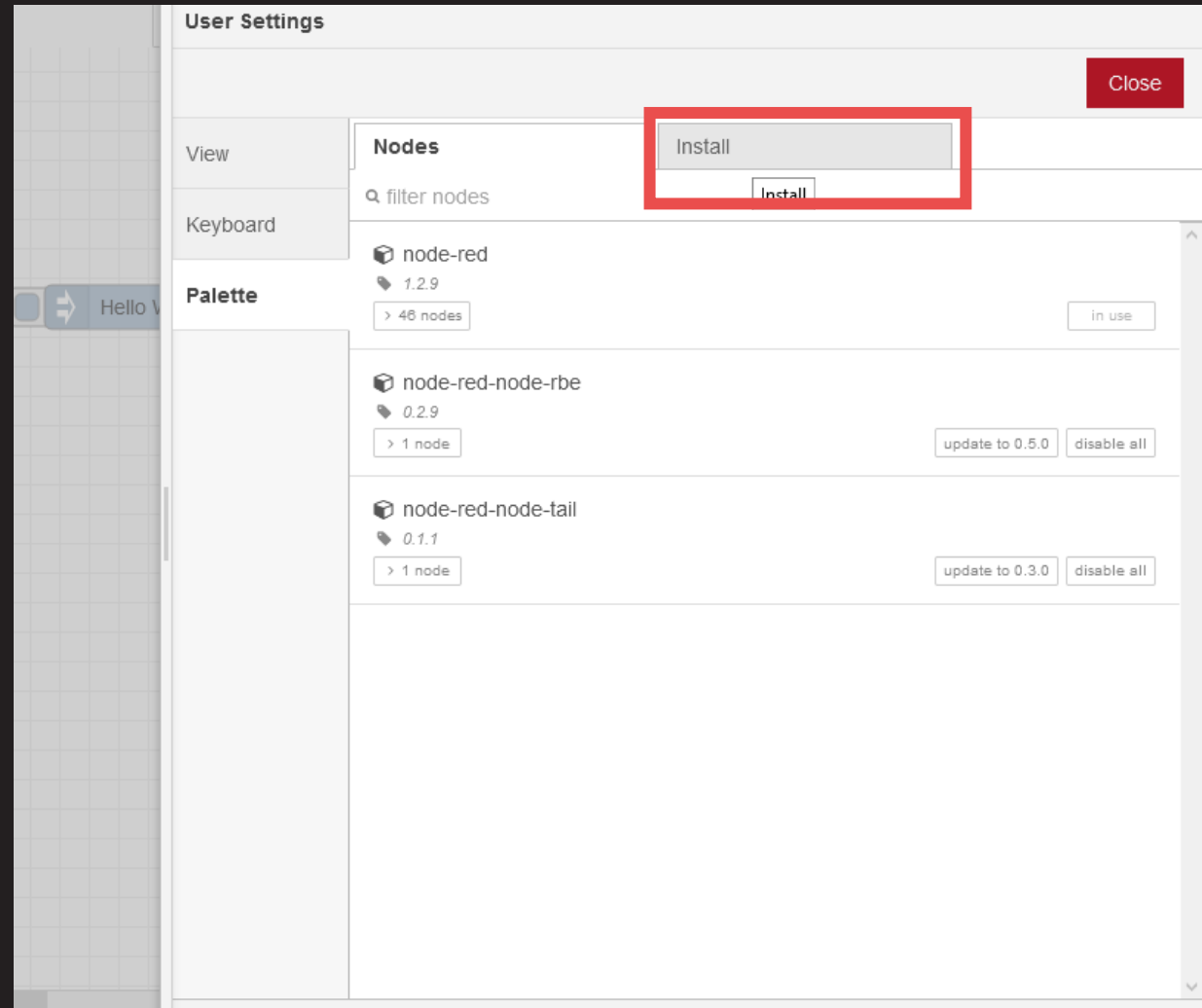
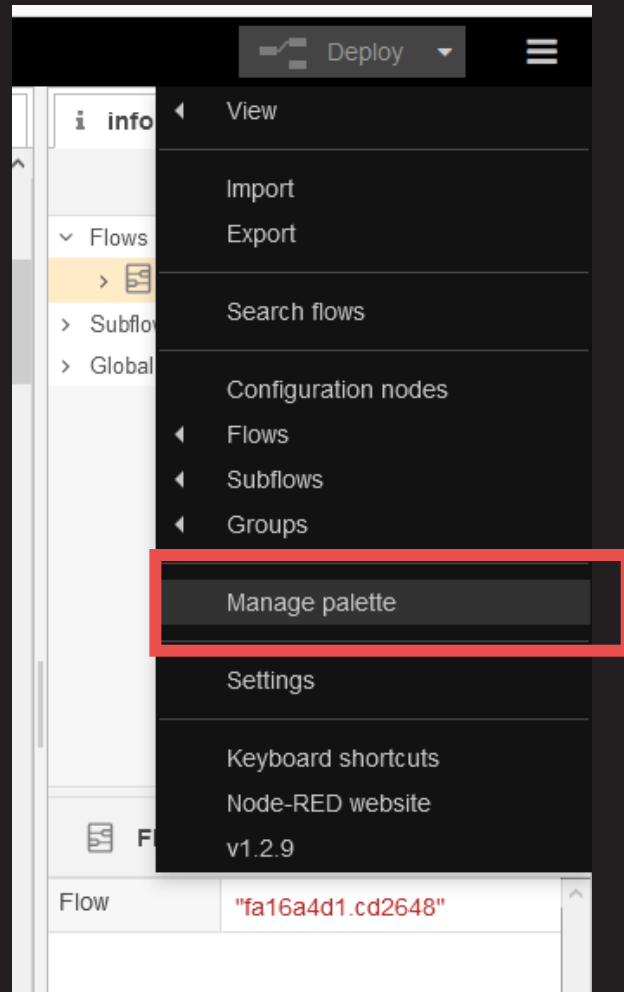
every 2 seconds

# Nós especiais e bibliotecas

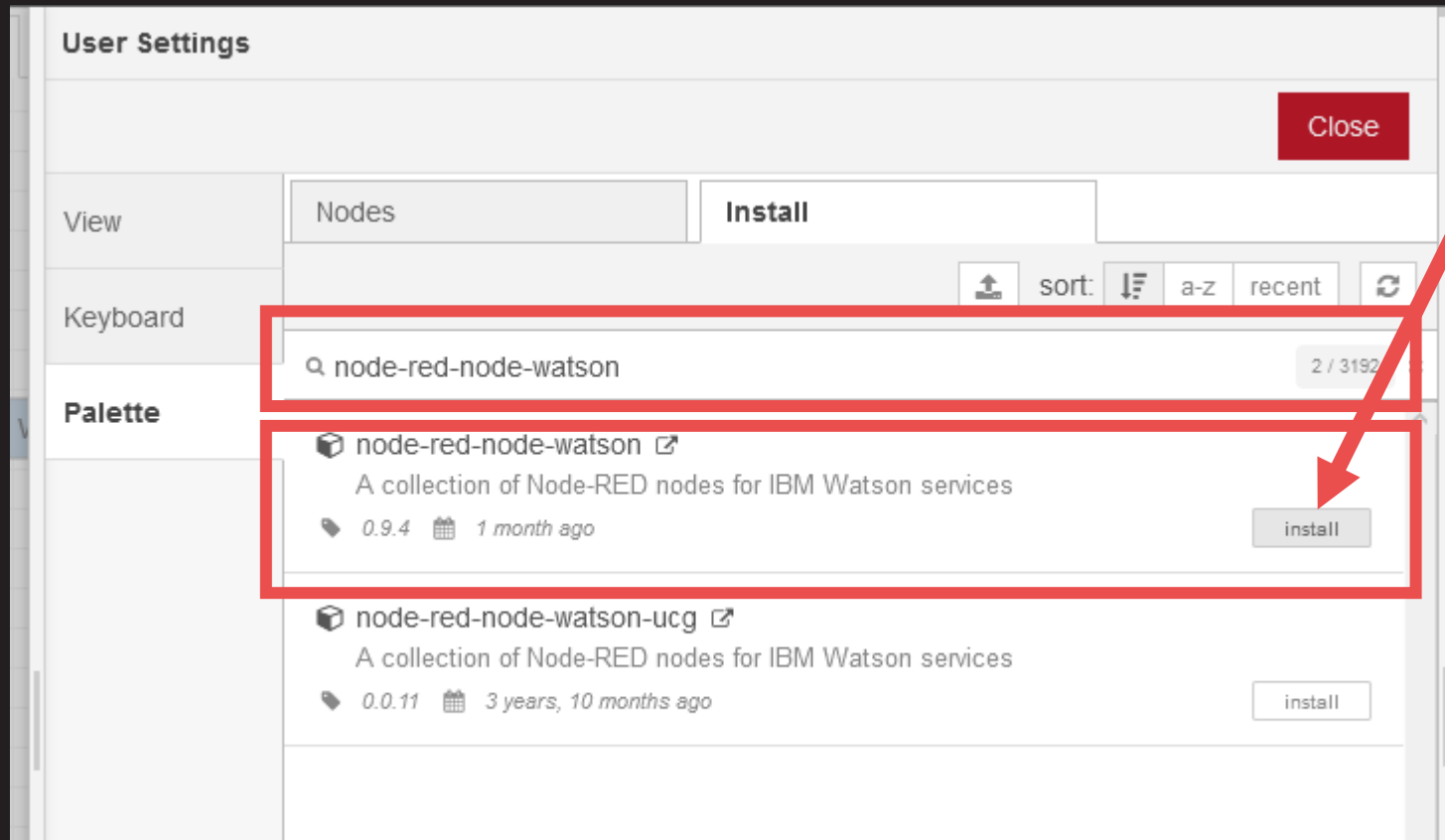
Instalando a biblioteca de desenvolvimento da IBM Cloud e do Telegram



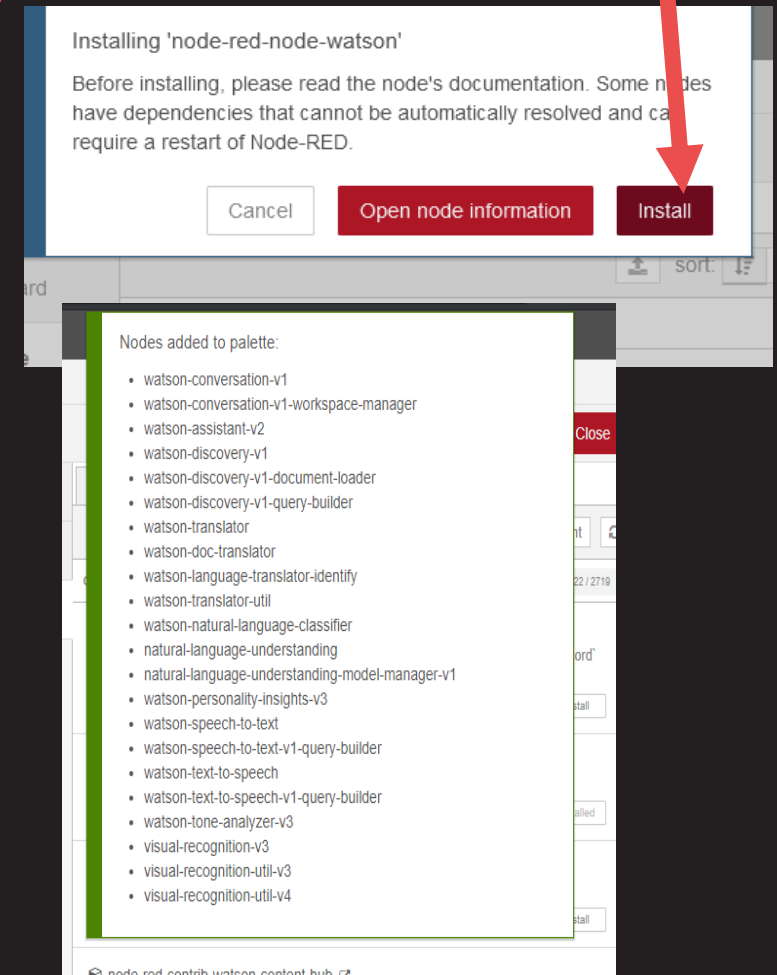
# Instalando a bibliotecas



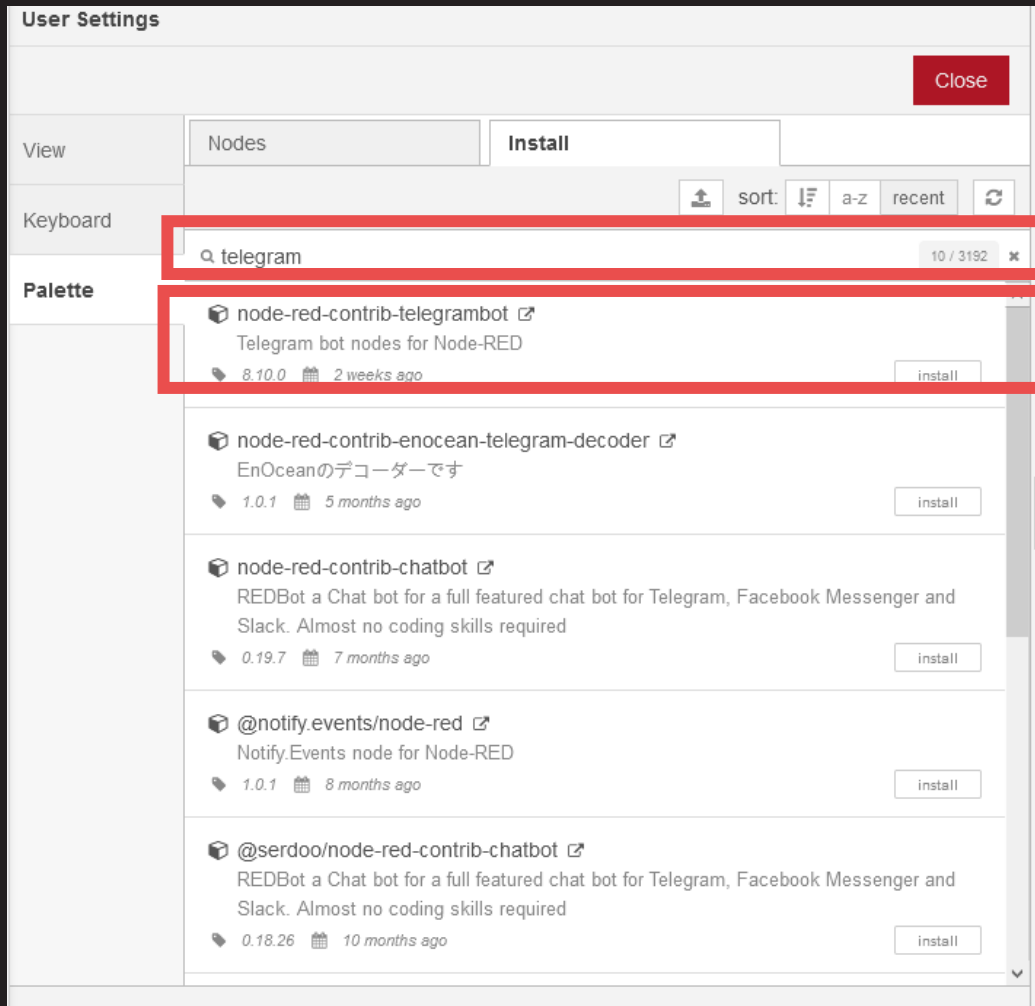
# Instalando a biblioteca do IBM Watson



## Instale o pacote



# Instalando a biblioteca do Telegram



## Nodes added to palette:

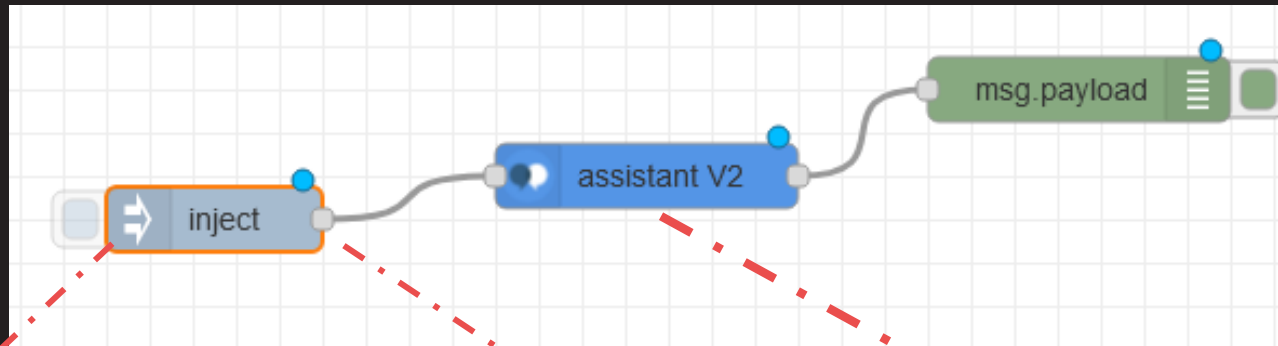
- telegram bot
- telegram receiver
- telegram command
- telegram event
- telegram sender
- telegram reply

# Watson Assistant

Testando o chatbot WA no Node-RED

# Configurando conexão com o Assistant de Vendas

Construa esse fluxo:



**Edit inject node**

Delete Cancel Done

**Properties**

Name

msg.payload = Qual o prazo de entrega?

**Edit assistant V2 node**

Delete Cancel Done

**Properties**

Name

Username

Password

API Key

Service Endpoint

Assistant ID

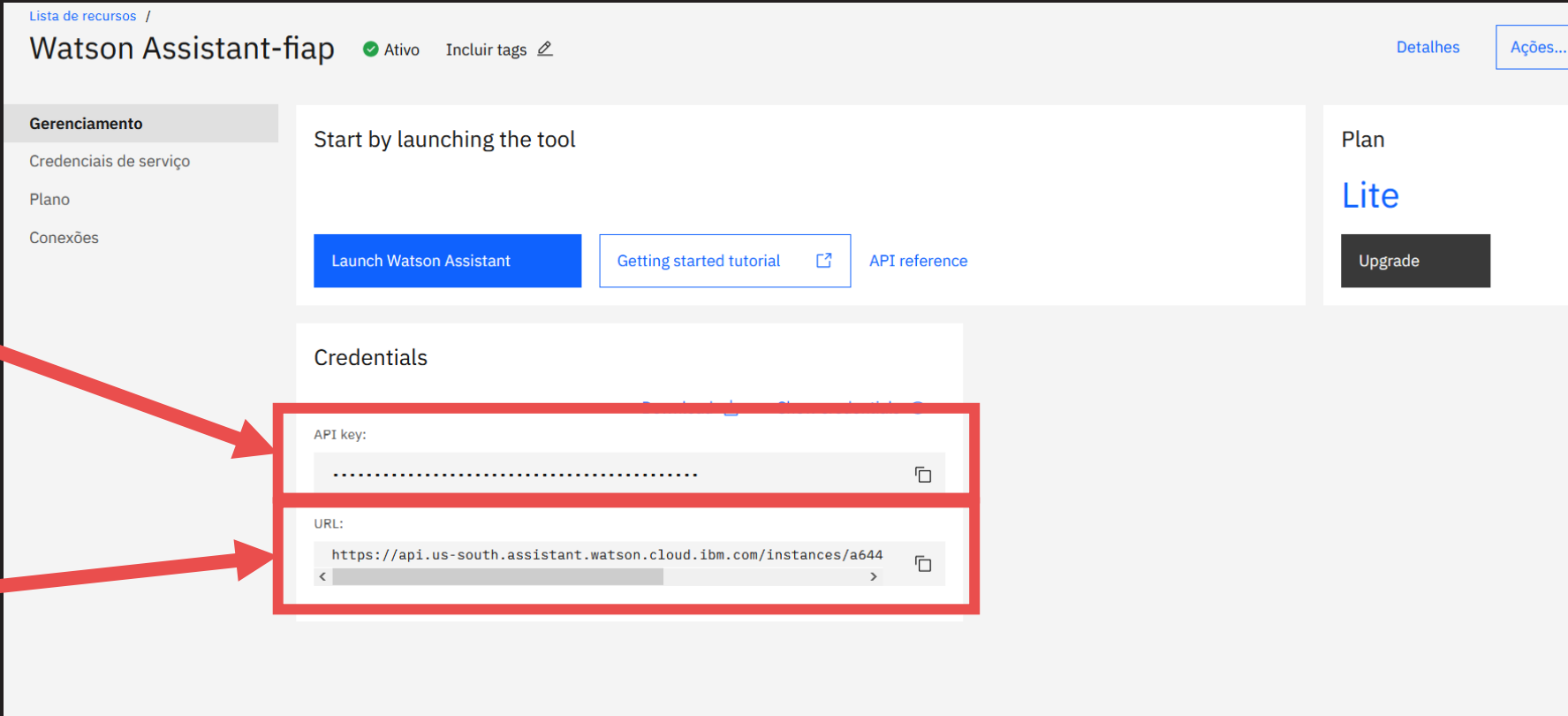
Timeout Period

Leave empty to disable

☐ Switch on Debug

# Configurando conexão com o Assistant de Vendas

A API Key fica na página de serviço do Watson Assistant. Este número é a senha para acessar todos os seus bots. Guarde-o bem!



The screenshot shows the Watson Assistant-fiap service page. The left sidebar contains a menu with 'Gerenciamento', 'Credenciais de serviço', 'Plano', and 'Conexões'. The main content area has a header with 'Watson Assistant-fiap', a status 'Ativo', and a link to 'Incluir tags'. Below this is a 'Start by launching the tool' section with buttons for 'Launch Watson Assistant', 'Getting started tutorial', and a link to 'API reference'. To the right is a 'Plan' section showing 'Lite' and an 'Upgrade' button. The 'Credentials' section is highlighted with a red box, containing an 'API key' field (masked with dots) and a 'URL' field (containing 'https://api.us-south.assistant.watson.cloud.ibm.com/instances/a644'). Two red arrows point from external text labels to these fields: 'API Key' points to the API key field, and 'Service Endpoint' points to the URL field.

API Key

Service Endpoint

# Configurando conexão com o Assistant de Vendas

Uma vez dentro do WA, clique nos três pontinhos no Assistant e selecione Settings:

## Assistants

An assistant helps your customers complete tasks and get information faster. It may clarify requests, search for answers from a knowledge base, and can also direct your customer to a human if needed.

Create assistant

### Vendas

Ajuda os clientes com eventuais dúvidas e indicando os melho...

#### Skills (1)

Ajuda

#### Integrations (1)



Rename

Settings

Delete

# Configurando conexão com o Assistant de Vendas

O ID do Assistant pode ser copiado agora:

## Assistant settings

Vendas

API details

Webhooks

Inactivity timeout

### API details

#### Assistant details

Assistant name:

Vendas

Assistant ID:

8e7878c6-d0d2-41c6-97ea-544c8fdf0170

Assistant URL:

https://api.us-south.assistant.watson.cloud.ibm.com/instances/a6440d84-f598-4

< >

#### Service credentials

Credentials name:

Auto-generated service credentials



# Testando a configuração

- Dê deploy e injete a mensagem.
- O que está aparecendo no nó de debug?
- Você é capaz de encontrar a resposta do chabot?

# Telegram

Conectando o bot a um serviço de mensagens

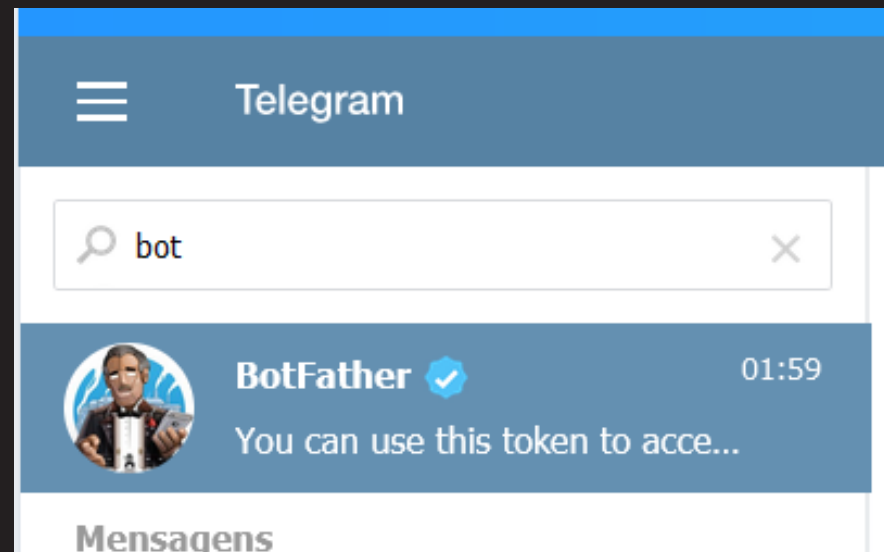
# Telegram



- 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;

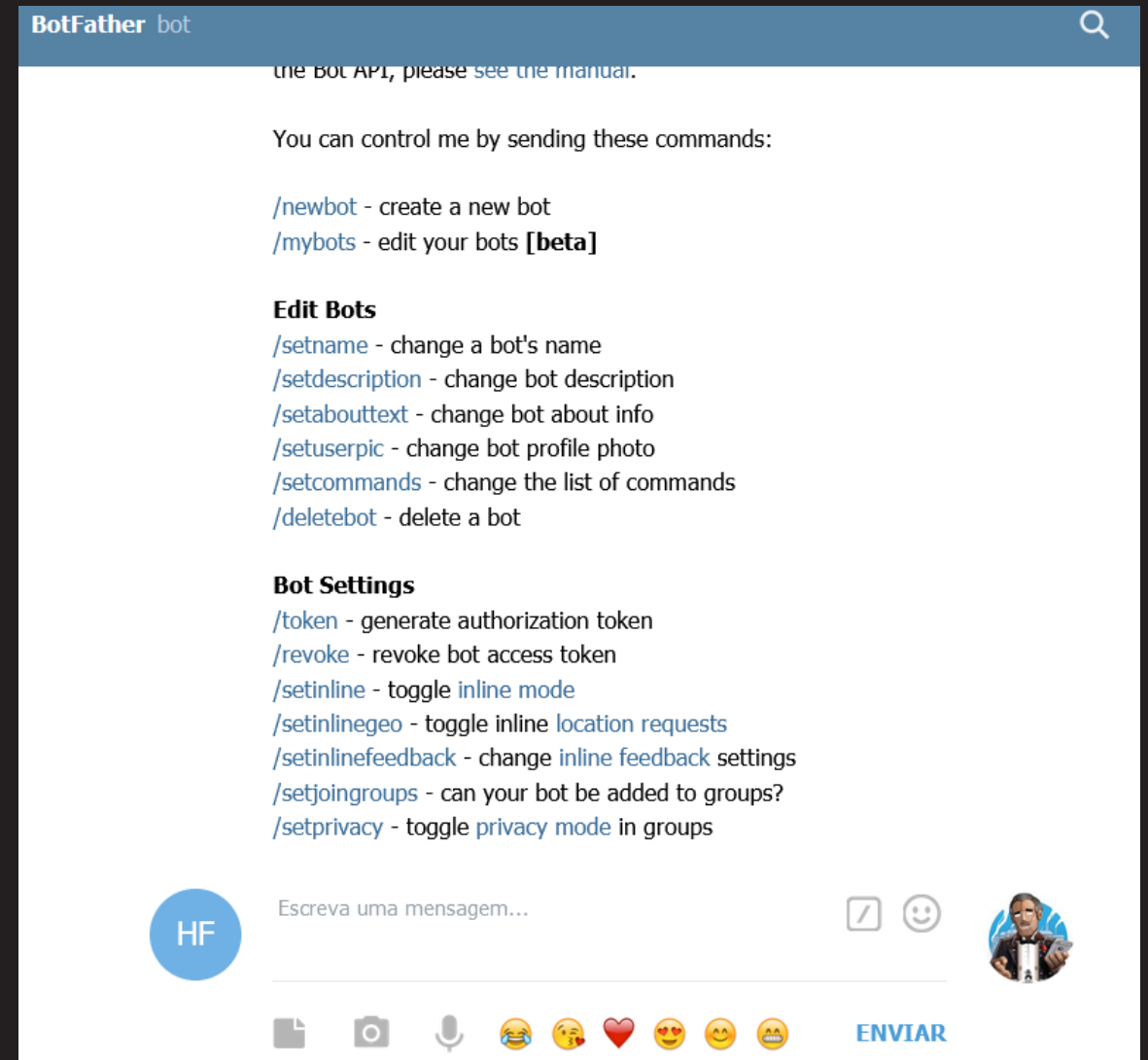
# Telegram - Botfather

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



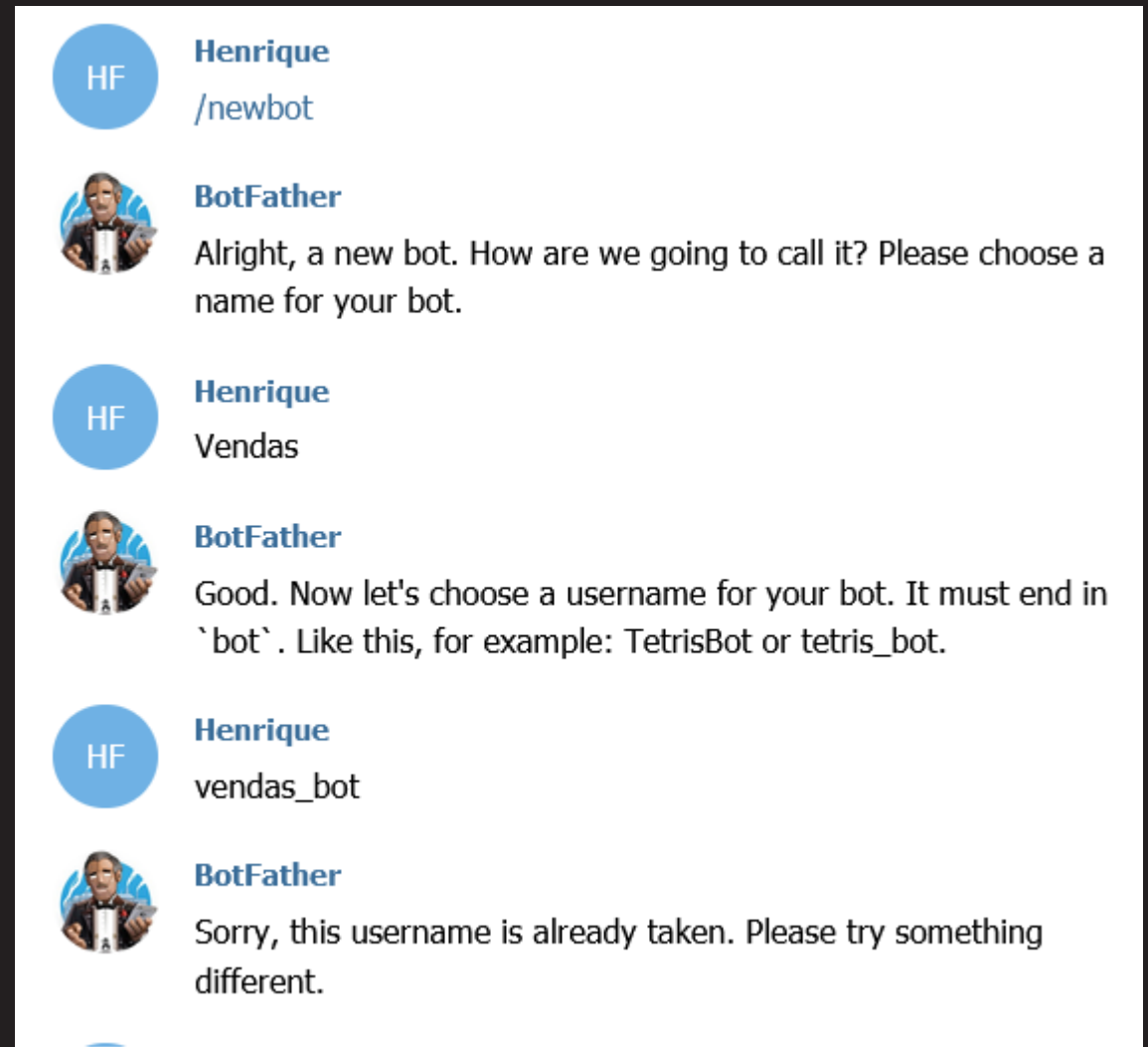
# Telegram - Botfather

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



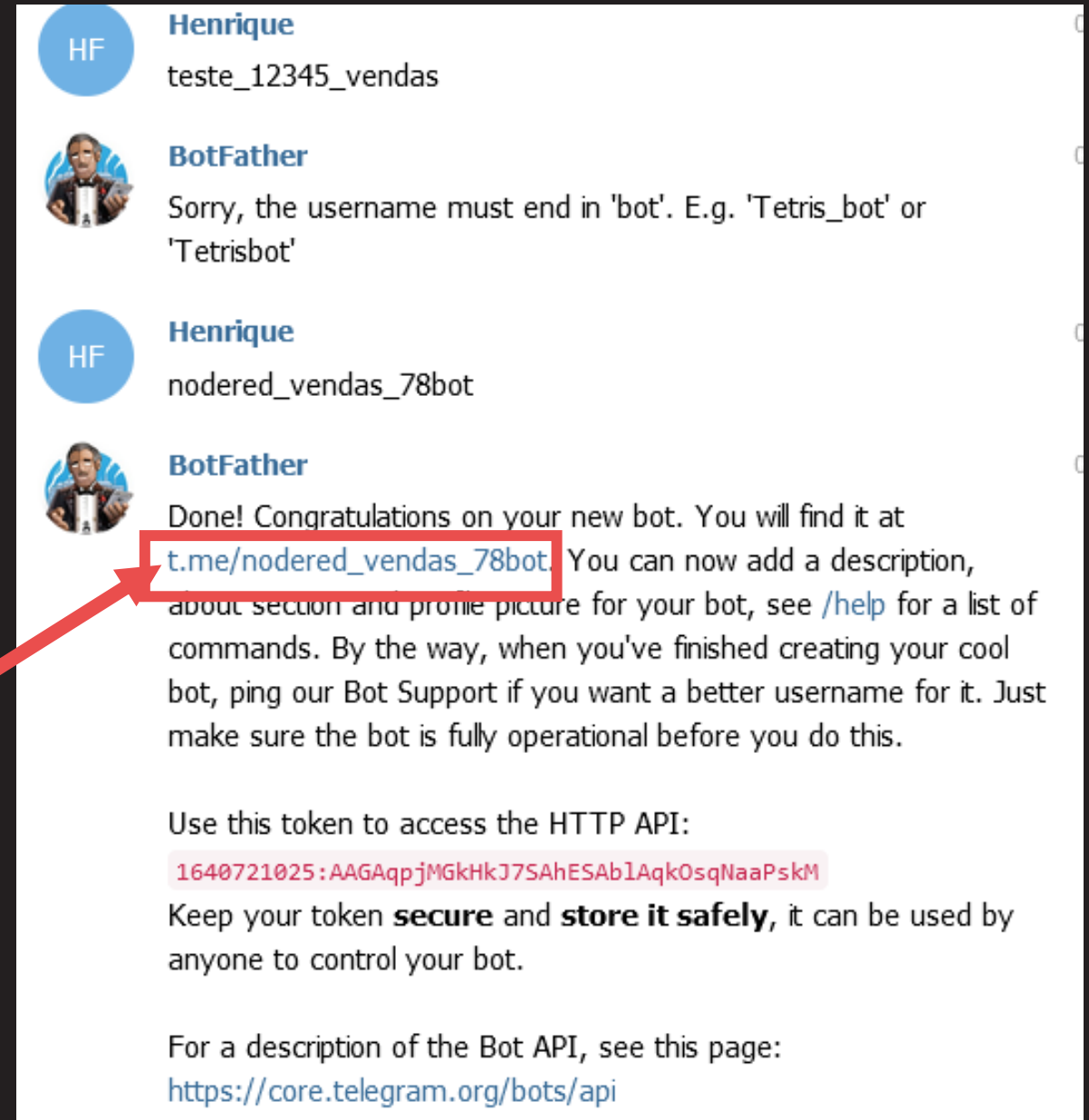
# Telegram - Botfather

- 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.



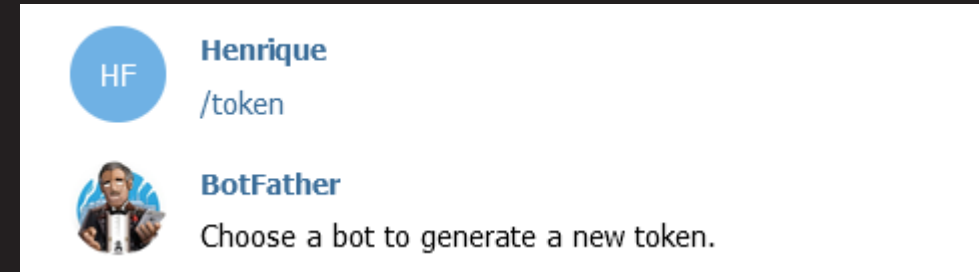
# Telegram - Botfather

- 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;



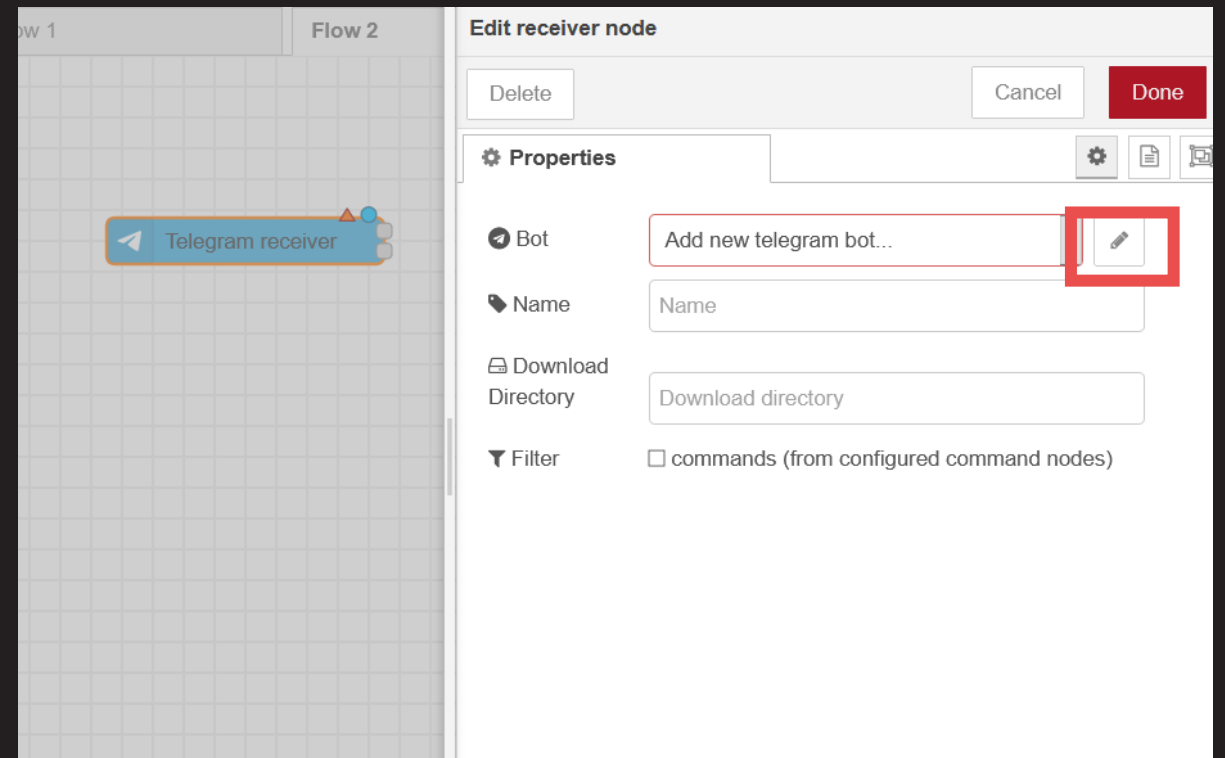
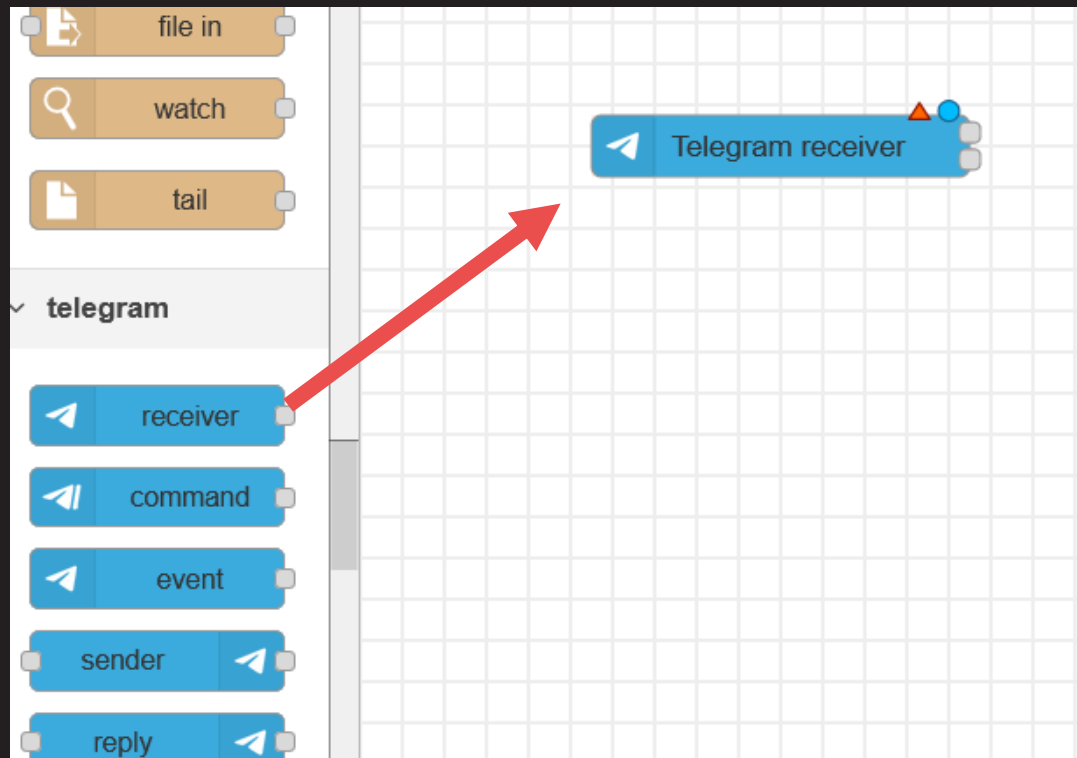


# Conectando-se ao Telegram

Fluxo de demonstração para testar a conexão com o Telegram

# 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)

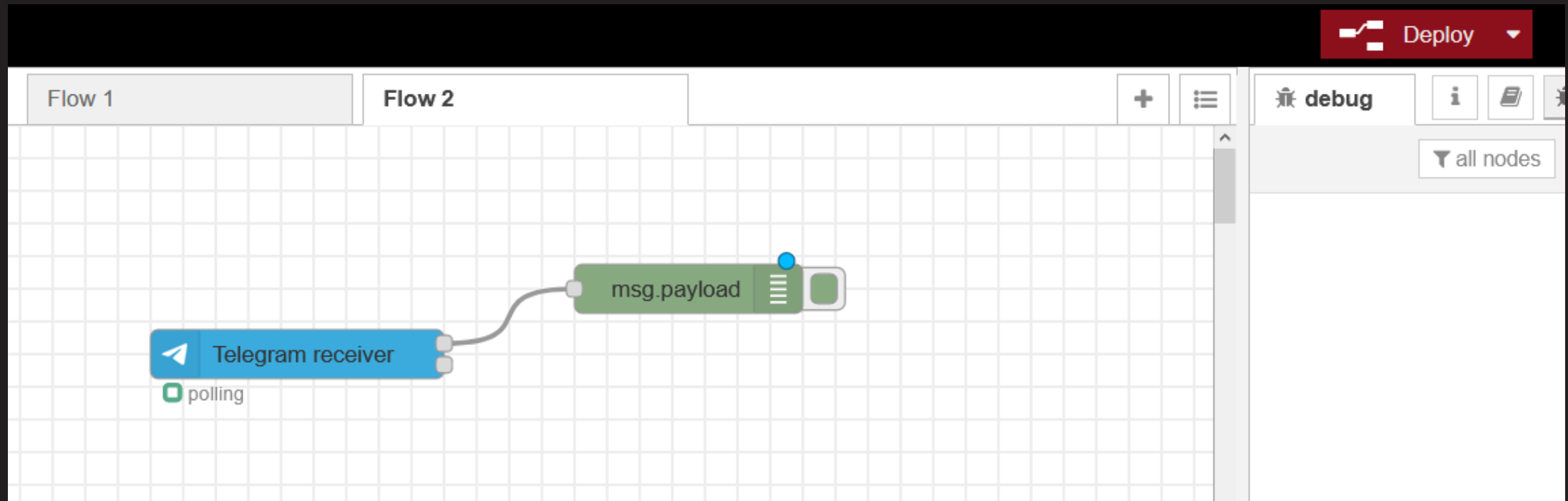
**Tip:** If you don't have a token yet, you can create a new one here: [@BotFather](#).

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

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

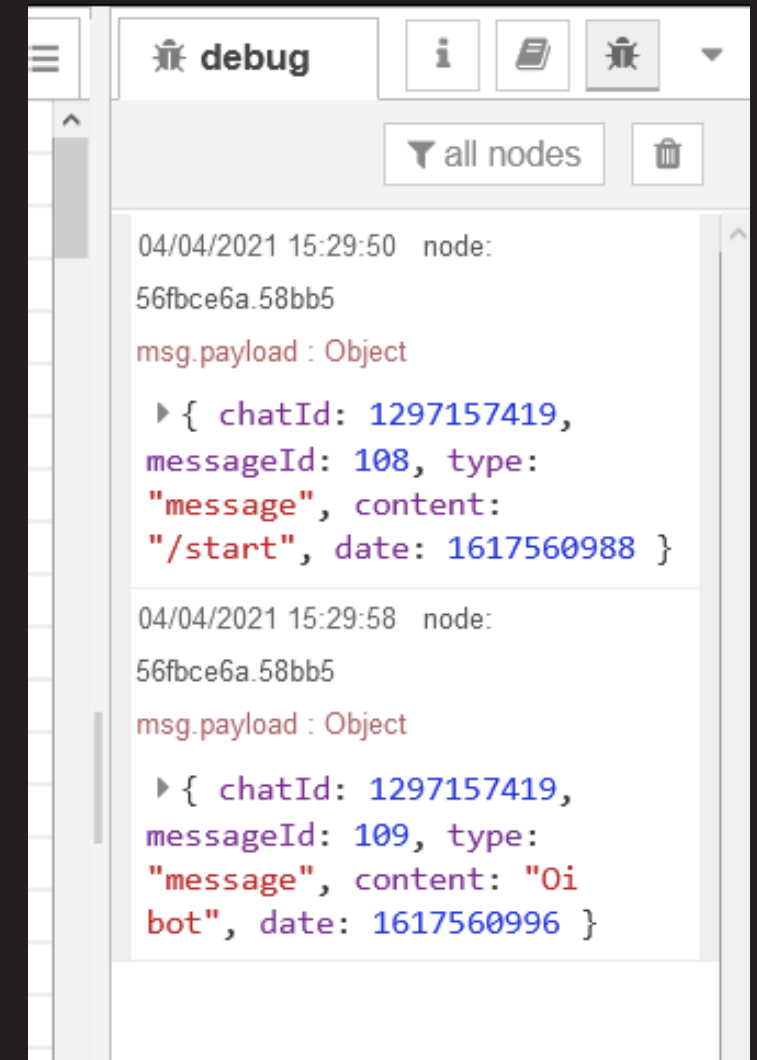
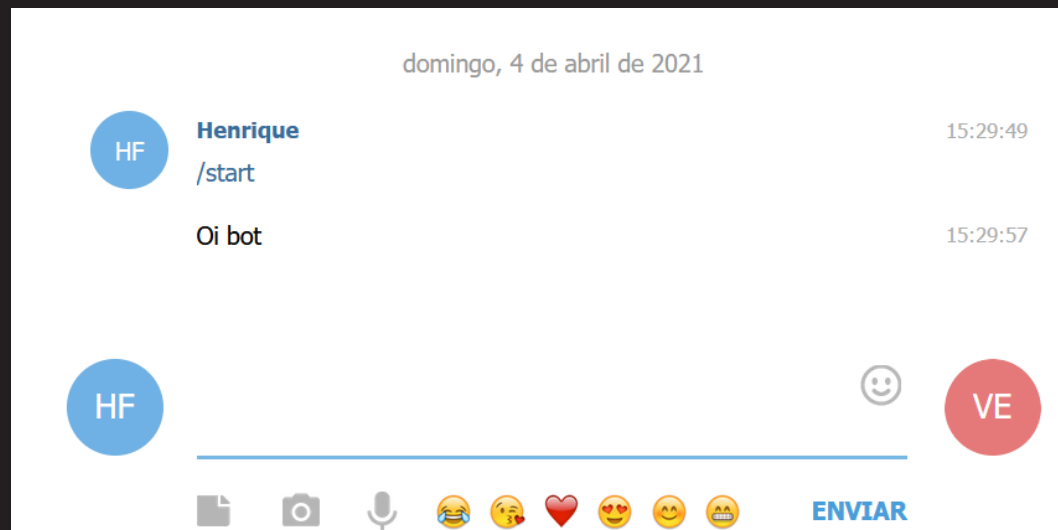
# Telegram + Node-RED

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



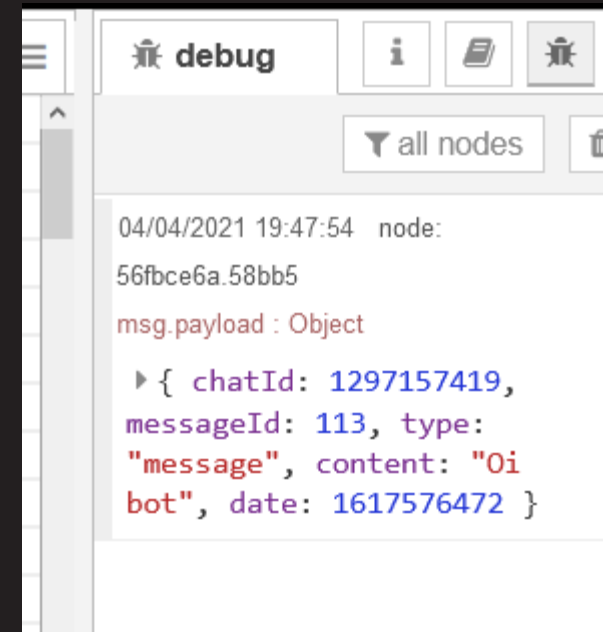
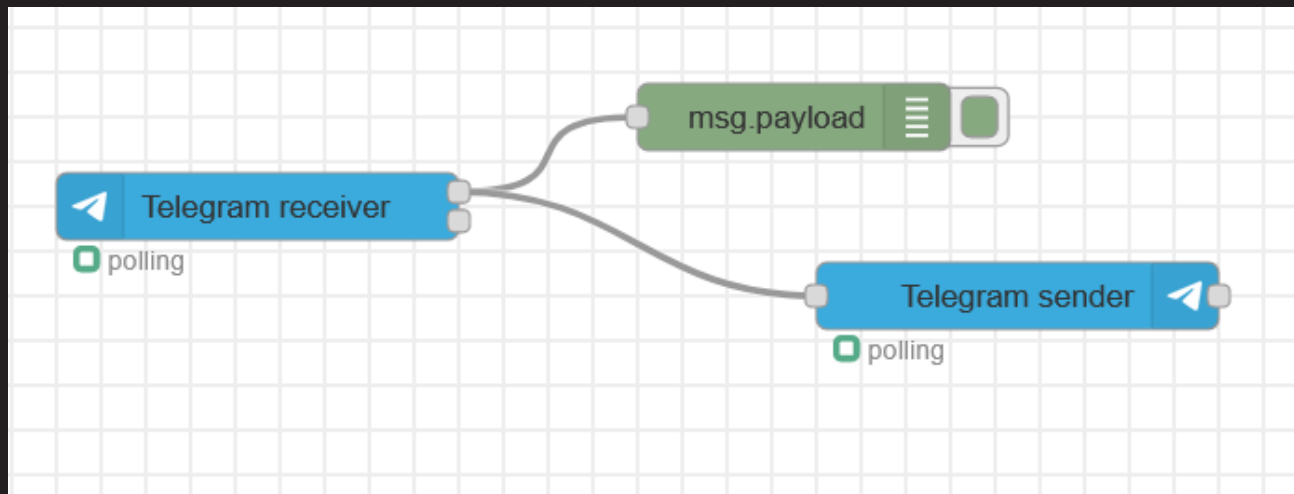
# Telegram + Node-RED

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.



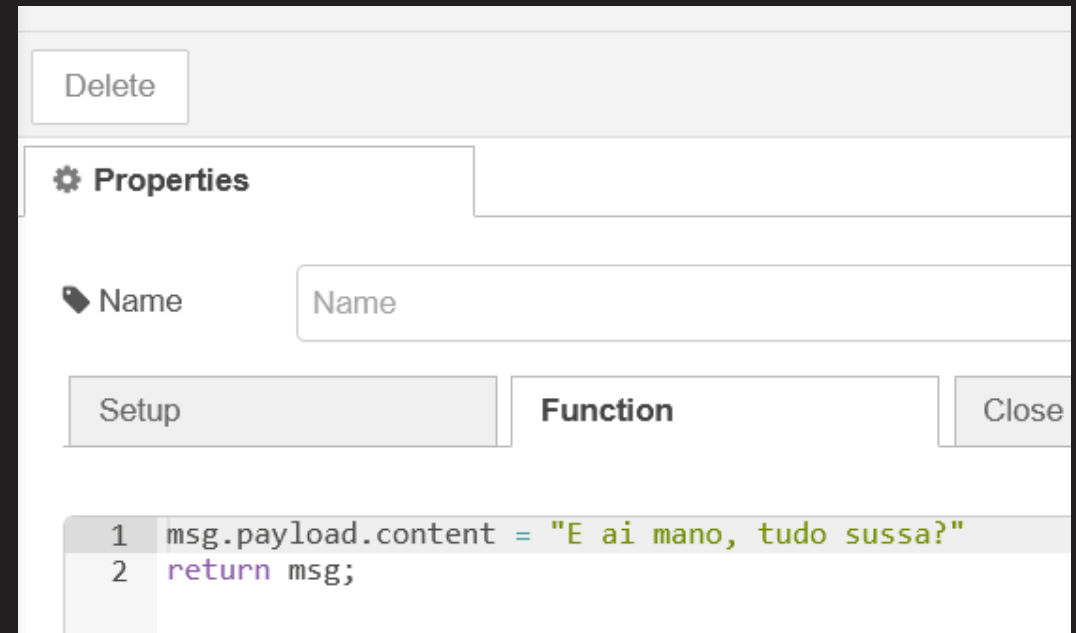
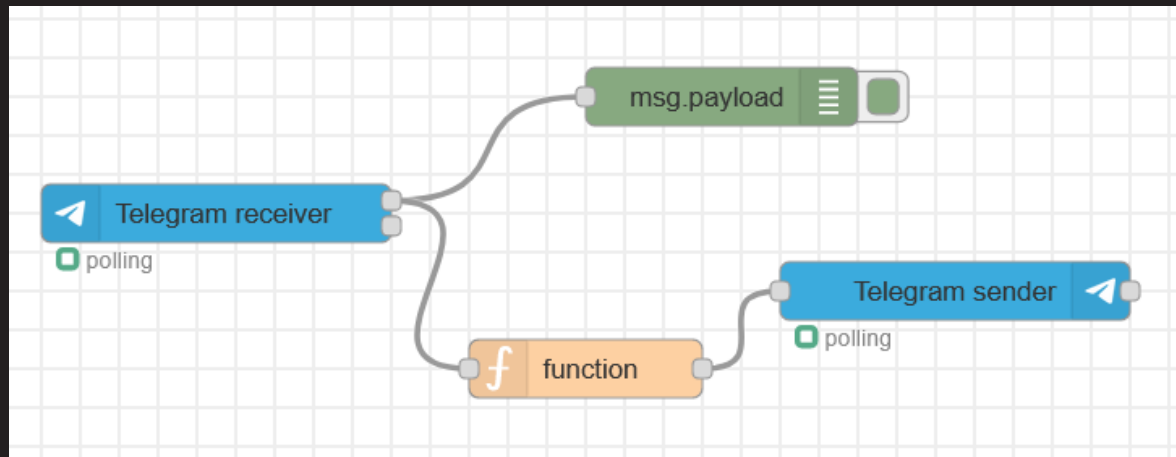
# Telegram + Node-RED

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 ai mano, tudo sussa?

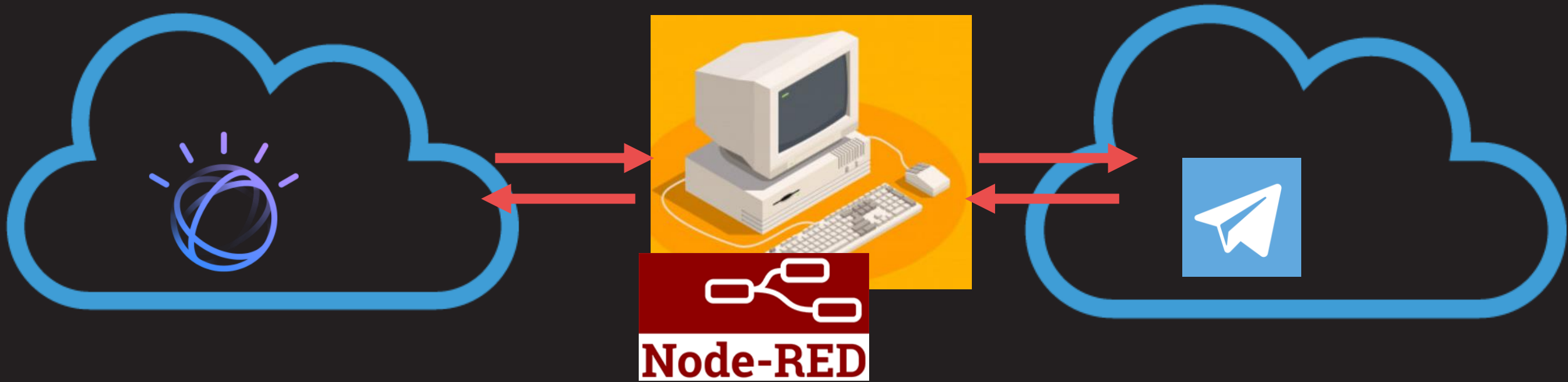
**Pergunta:** o que o bot responde para outra pergunta? Como podemos deixá-lo mais inteligente?

# Integrando nosso bot

Conectando o IBM Watson Assistant ao Telegram



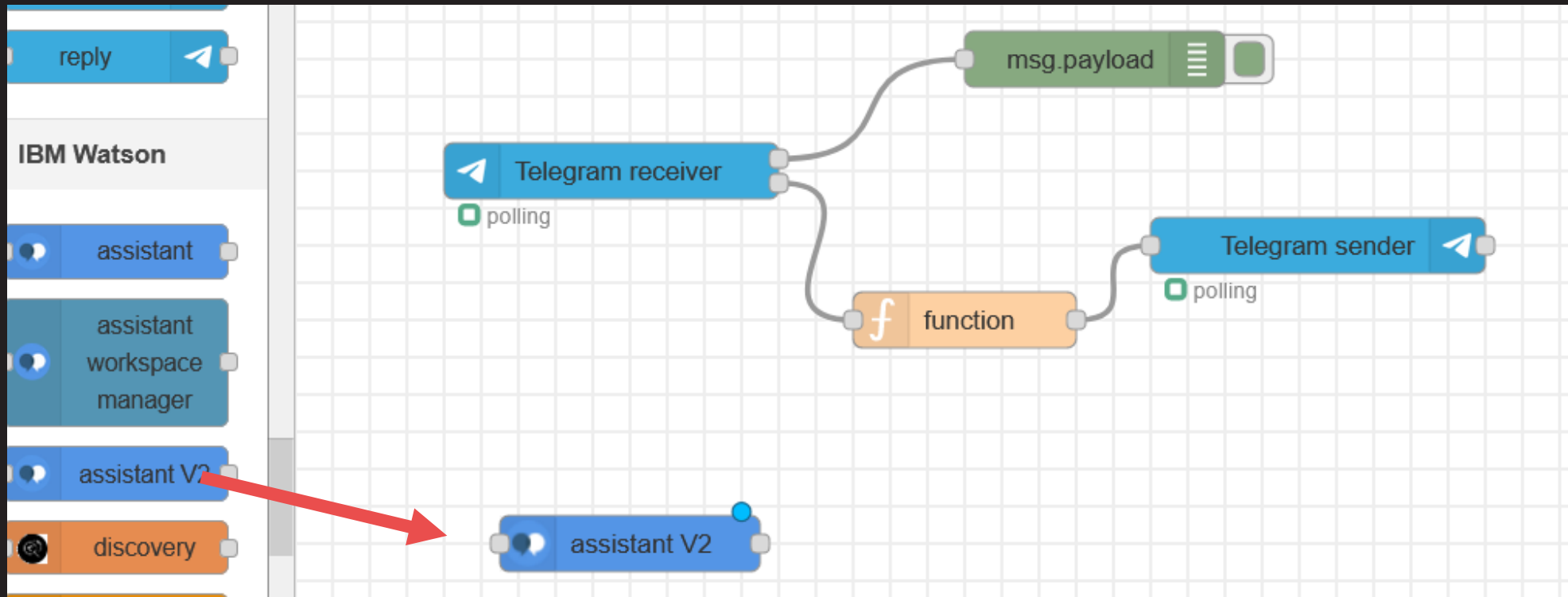
# Vamos usar o Watson Assistant para resolver o processamento inteligente da mensagem



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

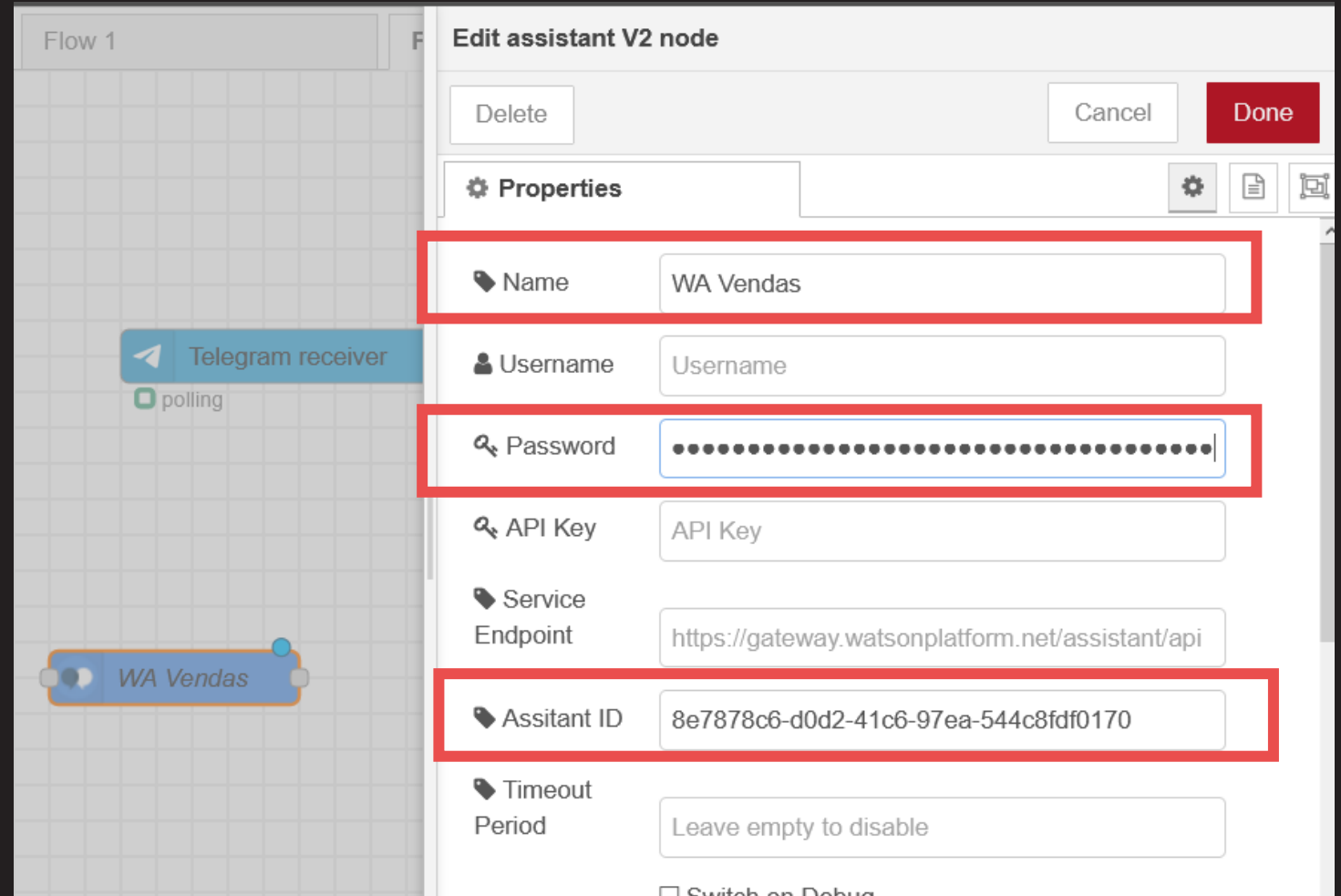
# Fazendo o Watson Assistant processar as mensagens do Telegram

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



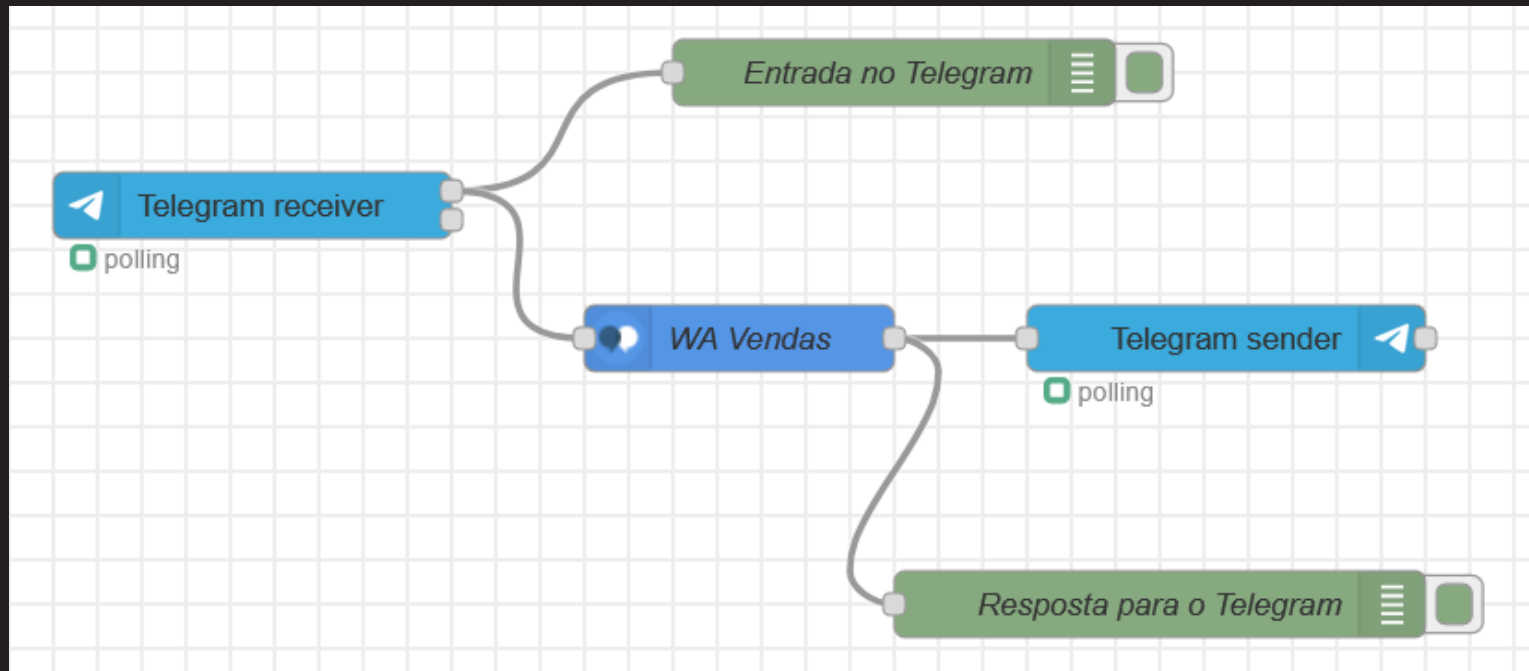
# Fazendo o Watson Assistant processar as mensagens do Telegram

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



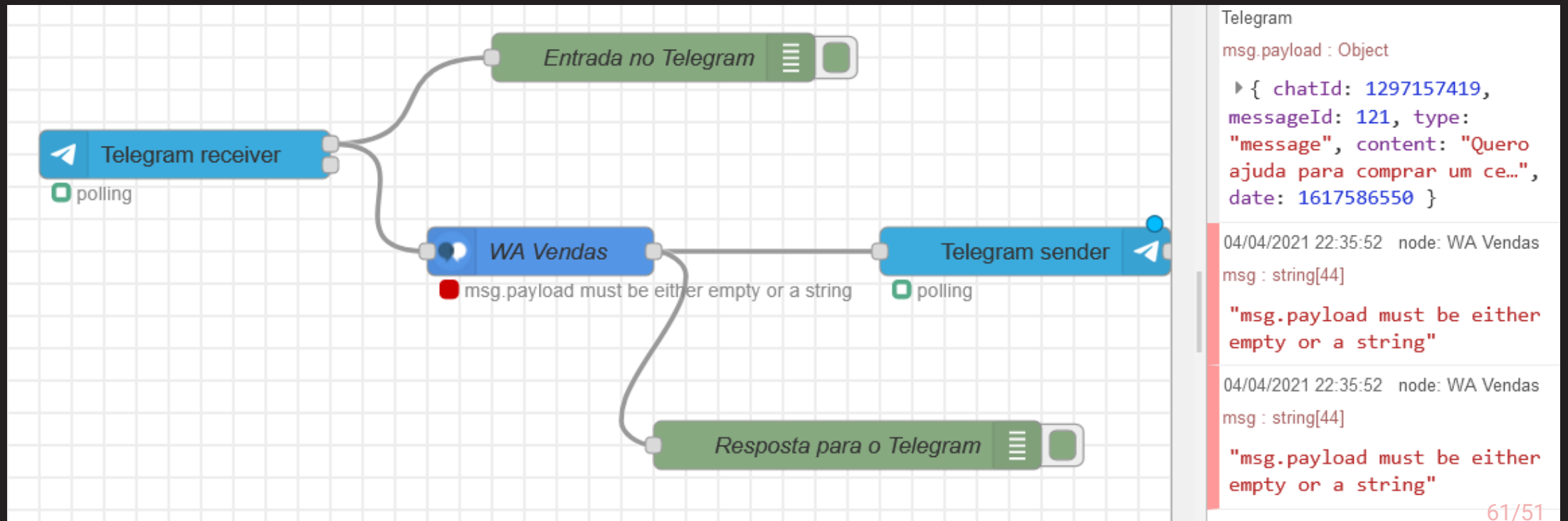
# Fazendo o Watson Assistant processar as mensagens do Telegram

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



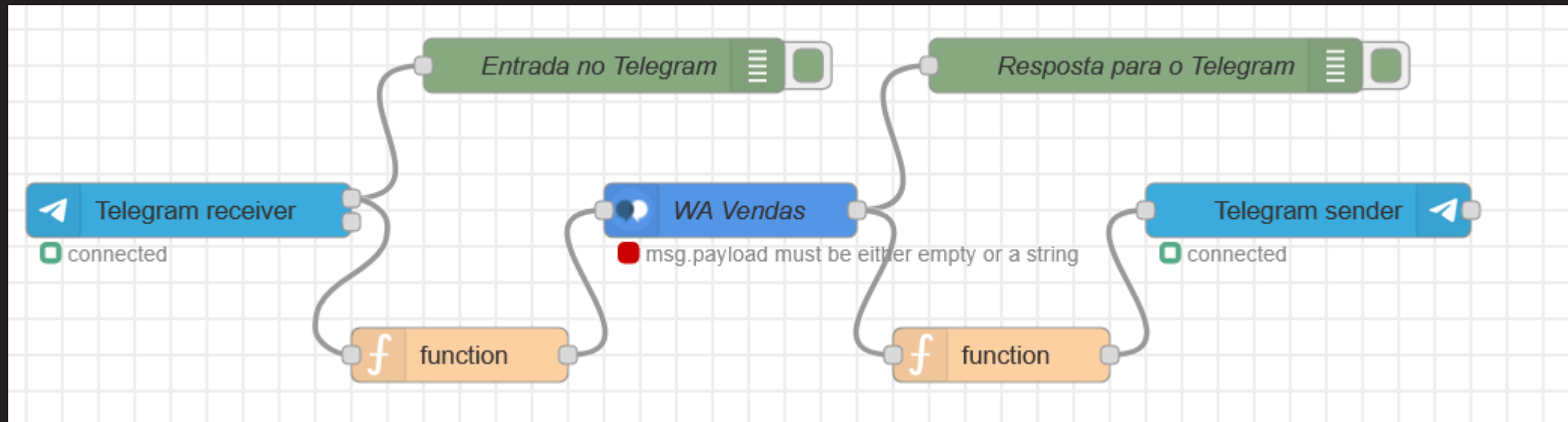
# Fazendo o Watson Assistant processar as mensagens do Telegram

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



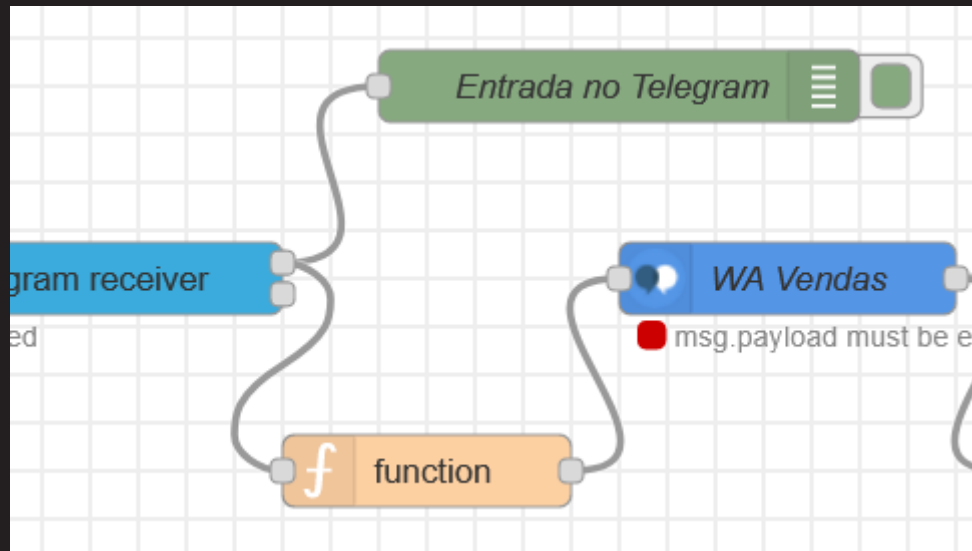
# Fazendo o Watson Assistant processar as mensagens do Telegram

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



# Fazendo o Watson Assistant processar as mensagens do Telegram

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



Edit function node

Delete Cancel Done

Properties

Name Proc Entrada

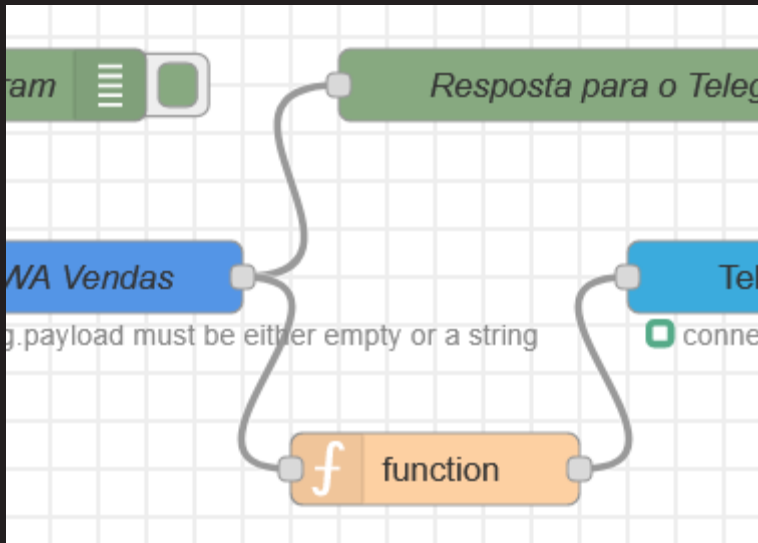
Setup Function Close

```
1 msg.params = {
2   "session_id" : msg.payload.session_id !== 'undefined' ? msg.payload
3 };
4 msg.chatId = msg.payload.chatId;
5 msg.payload = msg.payload.content;
6 return msg;
```

Outputs 1

# Fazendo o Watson Assistant 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

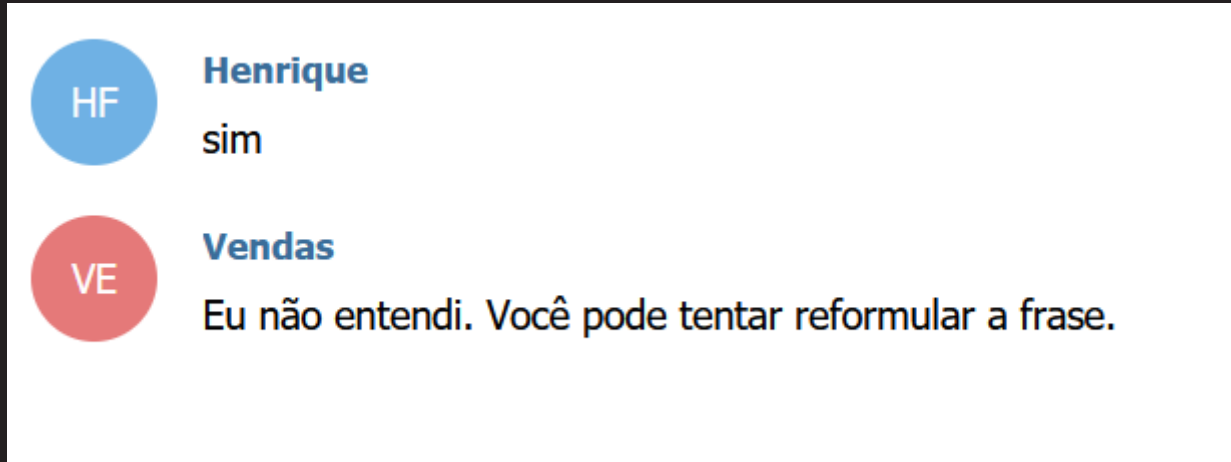


**Vendas**

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



HF Henrique  
sim

VE Vendas  
Eu não entendi. Você pode tentar reformular a frase.

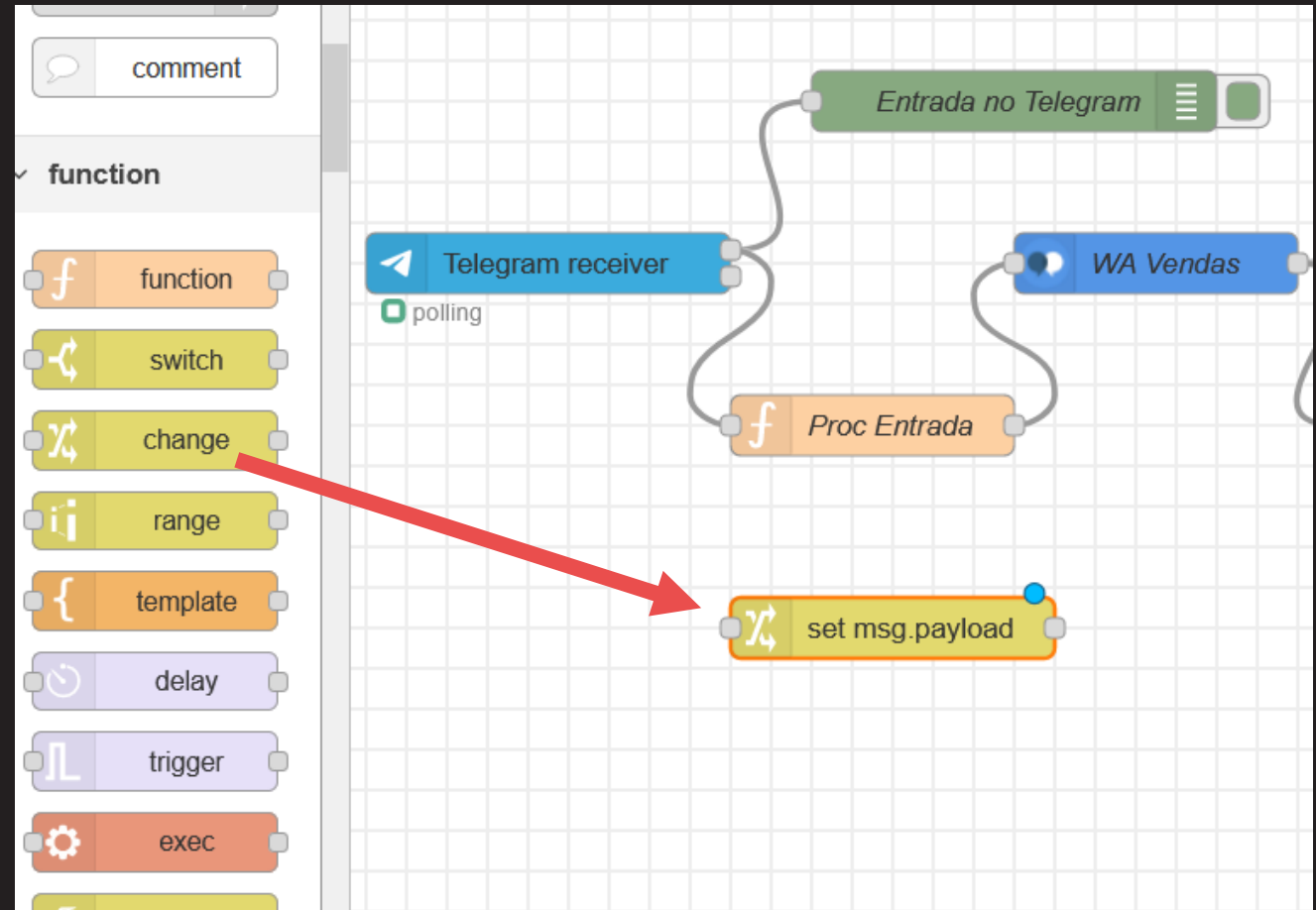
Isso acontece pois o **bot não sabe que está falando com a mesma pessoa**. Como nosso bot não está configurado para identificar um simples “sim” no início da árvore de diálogo, então ele irá cair no nó de **Em outros casos** (condição de `anything_else`)

# Integrando nosso bot 2

Mantendo o contexto de identificação de usuário

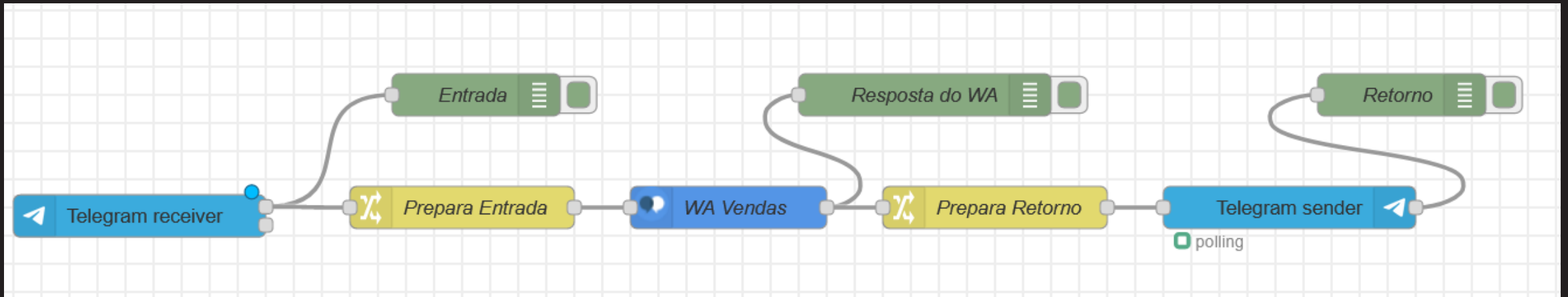
# Fluxo de integração: mantendo o ID do usuário

- 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

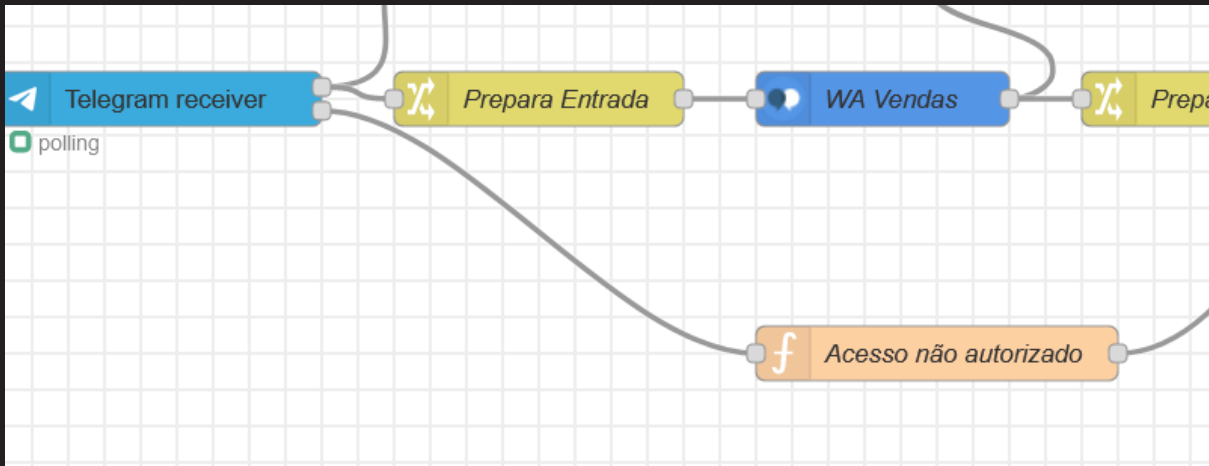
☰ Rules

≡	Set	▼ msg. payload.chatId	×
	to	▼ msg. chatId	
≡	Set	▼ msg. payload.type	×
	to	▼ a <sub>z</sub> 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:



Edit function node

Delete Cancel

⚙ Properties

🏷 Name Acesso não autorizado

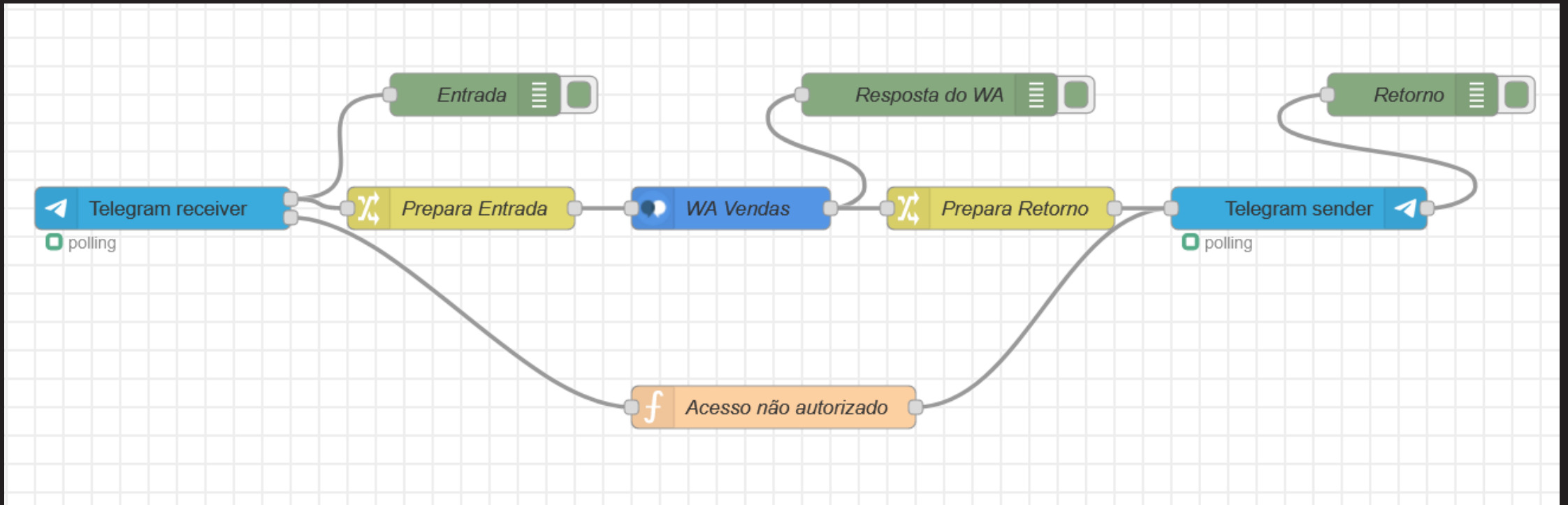
Setup Function Close

```
1 msg.payload.content = "Você não é um usuário autorizado";
2 return msg;
```



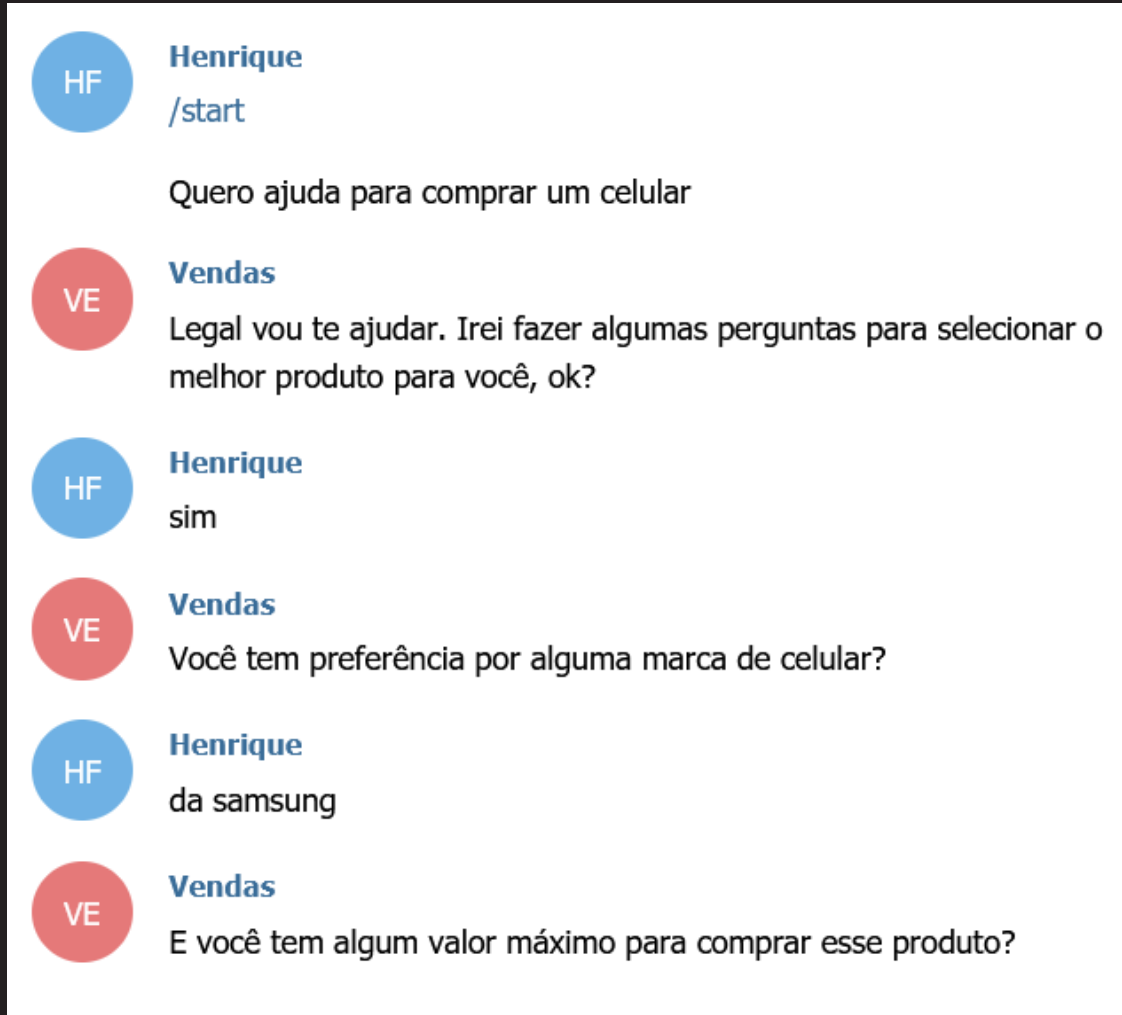
# Fluxo de integração: mantendo o ID do usuário

- 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. The chat history includes the following messages:

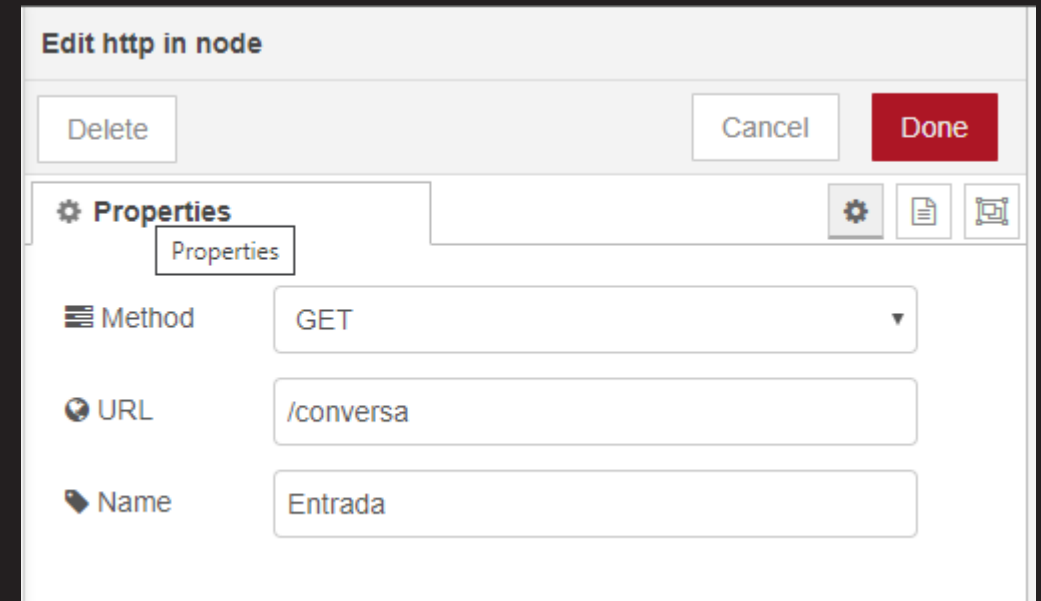
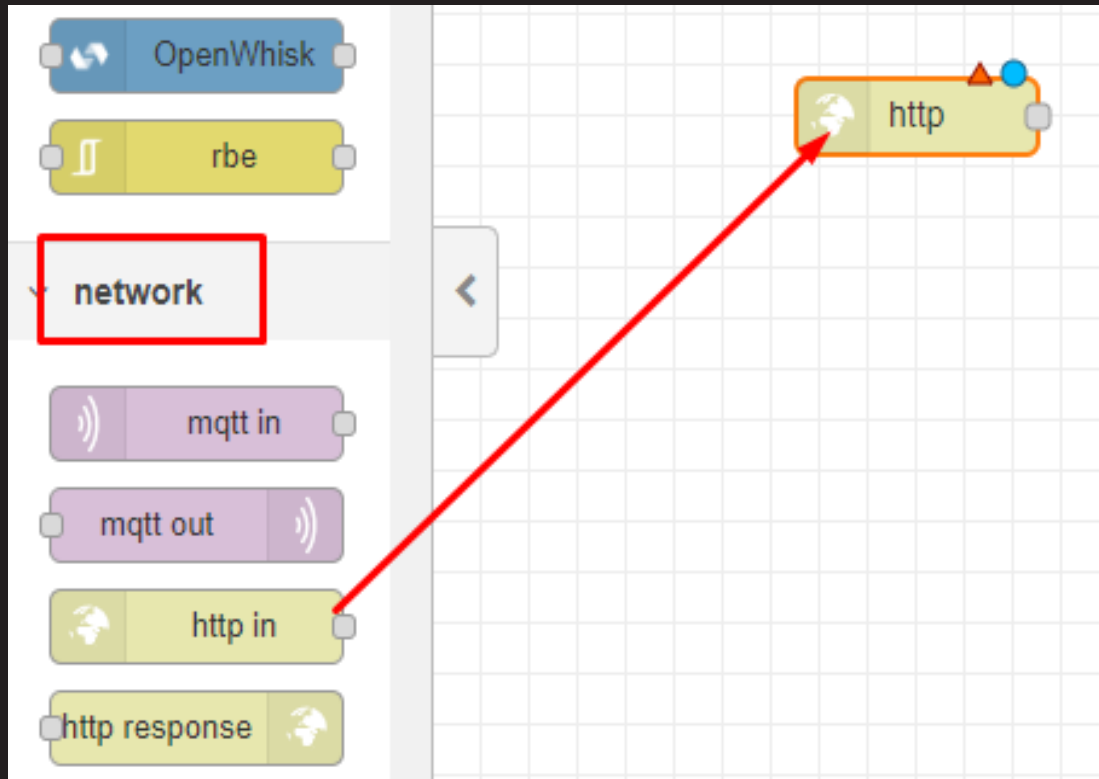
- Henrique (HF):** /start
- Vendas (VE):** Quero ajuda para comprar um celular
- Vendas (VE):** Legal vou te ajudar. Irei fazer algumas perguntas para selecionar o melhor produto para você, ok?
- Henrique (HF):** sim
- Vendas (VE):** Você tem preferência por alguma marca de celular?
- Henrique (HF):**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

# Conectando o Bot via HTTP

Fazendo um fluxo para enviar e receber mensagens do Watson Assistant

# Configurando conexão com o Assistant de Vendas

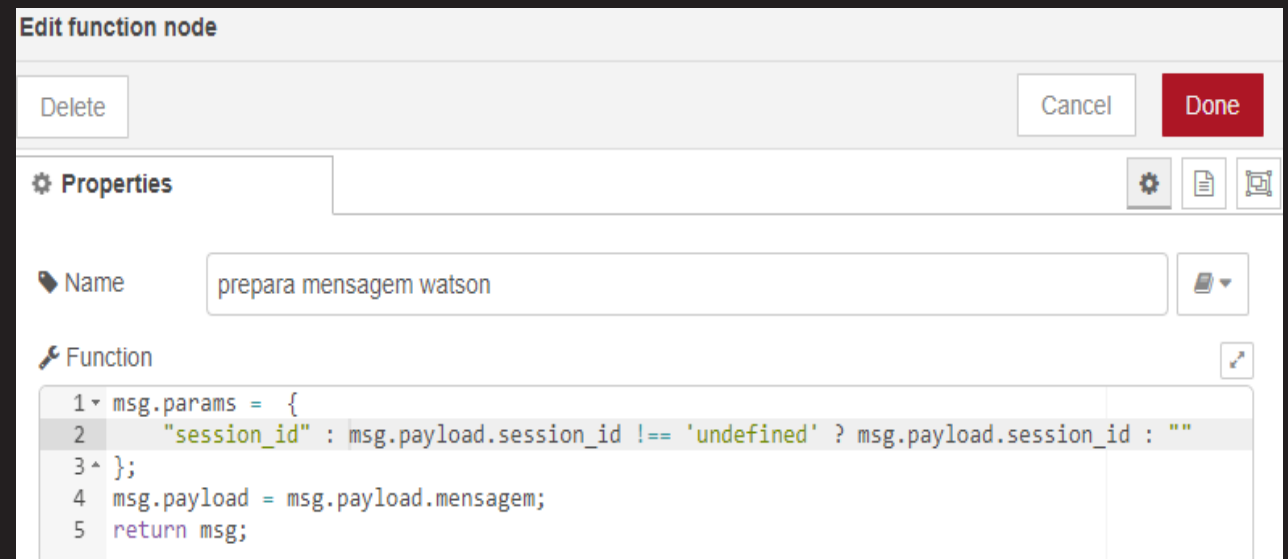
Adicione um nó de **http in**. Depois dê um duplo clique e preencha as propriedades do nó como abaixo:



# Configurando conexão com o Assistant de Vendas

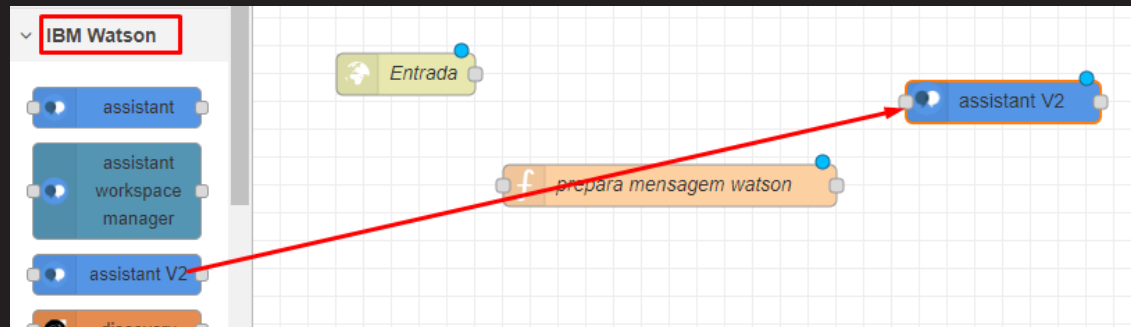
Adicione um nó de **function** com nome de Prepara Mensagem para o Watson. Nas propriedades, digite o seguinte código:

```
msg.params = {  
    "session_id" : msg.payload.session_id !== 'undefined' ? msg.payload.session_id : ""  
};  
msg.payload = msg.payload.mensagem;  
return msg;
```



# Configurando conexão com o Assistant de Vendas

Arraste e solte um nó de Assistant v2.



Precisamos preencher o campo **API Key** e **Assistant ID**. Estes valores são do seu serviço. Você precisará abrir a IBM Cloud para pegá-los.

Edit assistant V2 node

Delete Cancel Done

**Properties**

Name

Username

Password

API Key

Service Endpoint

Assistant ID

Timeout Period

☐ Switch on Debug

☐ Restart Dialog

☒ Return Context

☐ Return Alternate Intents

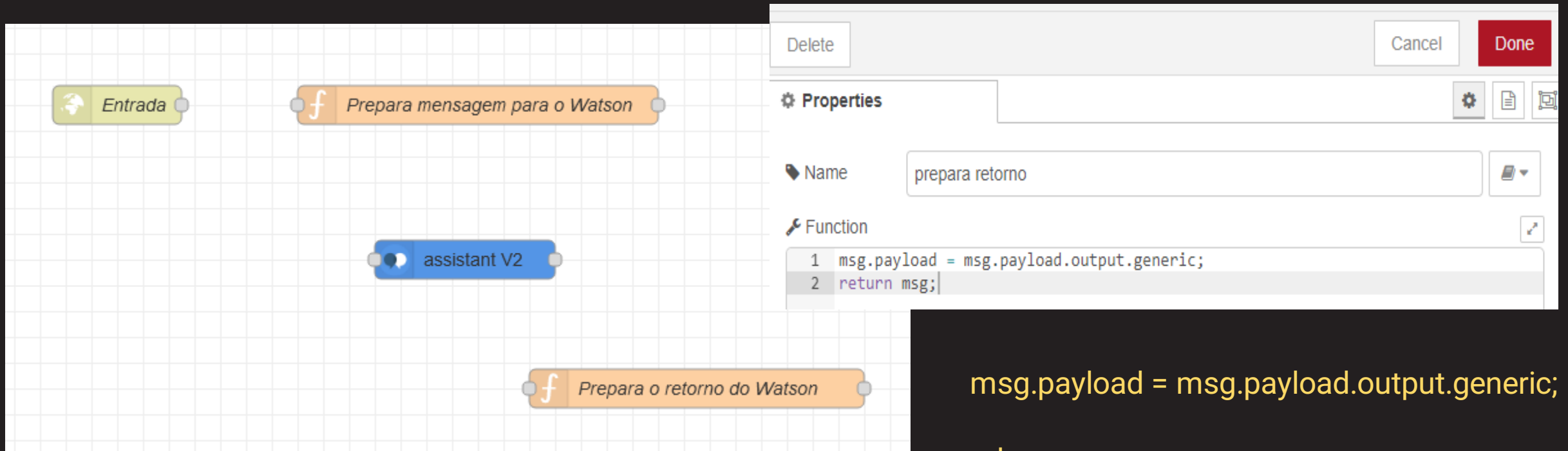
☒ Multiple Sessions

☐ Opt Out Request Logging

**Note:** When using multiple sessions, and `msg.params.session_id` is not set then a new session id is generated. See info box for details.

# Configurando conexão com o Assistant de Vendas

Adicione mais um nó de **function** para tratar os dados de retorno.



The screenshot displays the IBM Cloud Functions interface. On the left, a workflow is visible on a grid background, consisting of four nodes: a yellow 'Entrada' node, an orange 'Prepara mensagem para o Watson' node, a blue 'assistant V2' node, and another orange 'Prepara o retorno do Watson' node. On the right, a configuration panel for a function named 'prepara retorno' is shown. The 'Function' tab is active, displaying the following code:

```
1 msg.payload = msg.payload.output.generic;  
2 return msg;
```

Below the code editor, the same two lines of code are repeated in a larger font for emphasis:

```
msg.payload = msg.payload.output.generic;  
return msg;
```

# Configurando conexão com o Assistant de Vendas

Adicione um nó de http response e configure-o como na imagem:

The image shows a Node-RED workspace with a palette on the left containing various nodes. A red arrow points from the 'http response' node in the palette to an 'http' node in the workspace. The 'http' node is connected to a 'Vendas' node and two function nodes labeled 'prepara mensagem watson' and 'prepara retorno'. A properties dialog is open for the 'http' node, showing the following configuration:

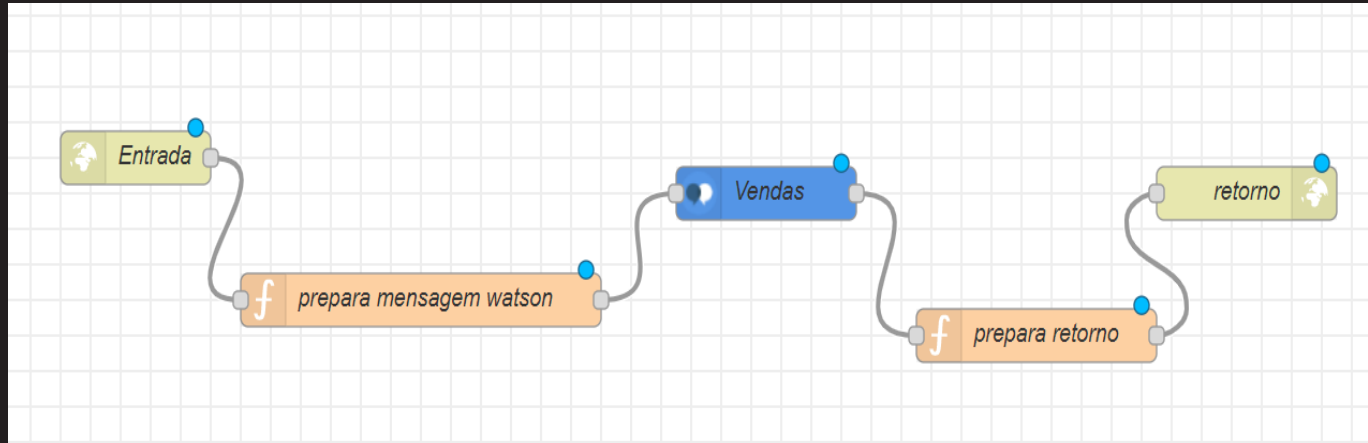
- Name:** retorno
- Status code:** msg.statusCode
- Headers:** Access-Control-Allow-Origin (value: \*)

The 'Access-Control-Allow-Origin' header and its value '\*' are highlighted with red boxes in the original image.



# Configurando conexão com o Assistant de Vendas

Temos um integração pronta!



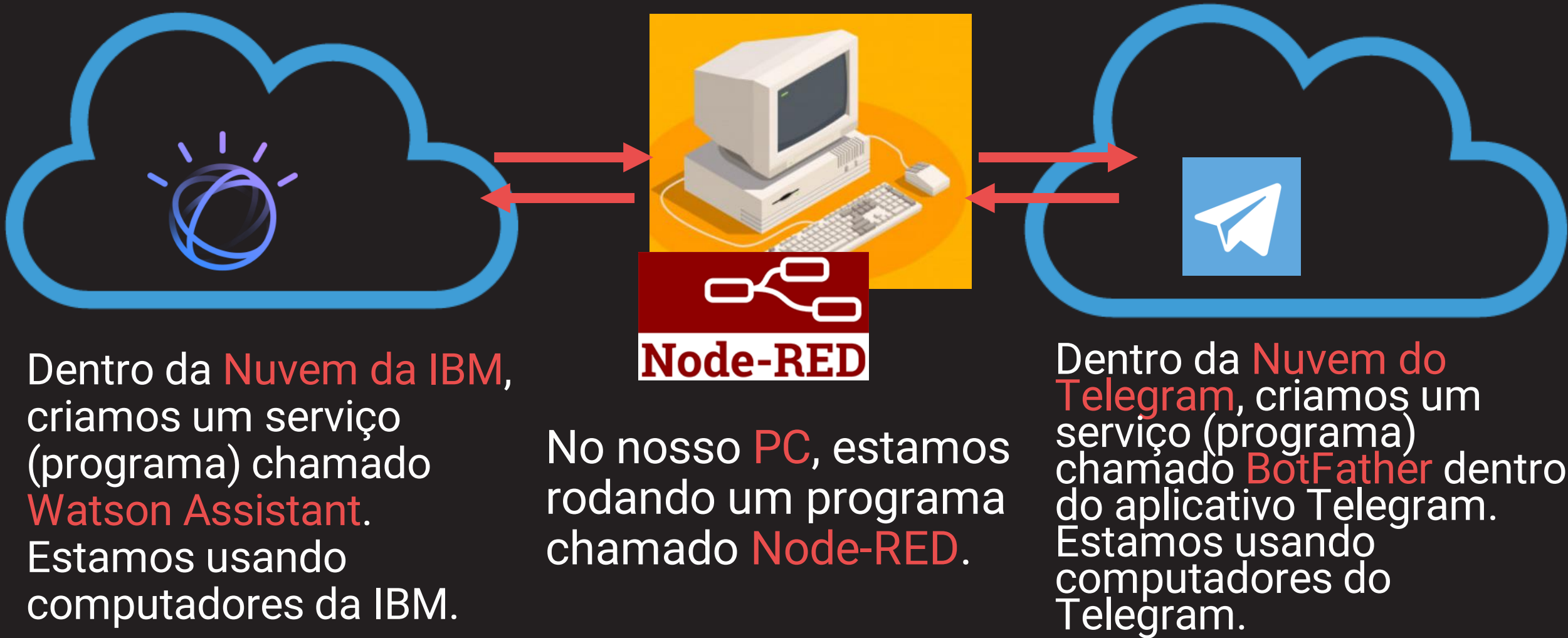
Para testar localmente, digite no navegador o IP local, seguido da porta do Node-RED / o endereço do nó de entrada.



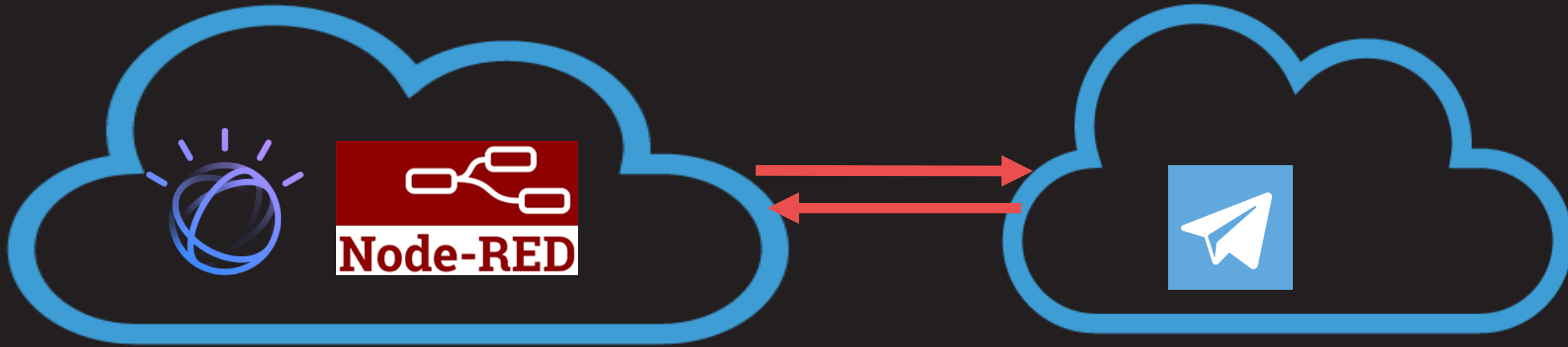
# Integração de Serviços

Entendendo como as coisas se conectam na internet

# Integração de Serviços – Computação Distribuída



# Integração de Serviços – Computação Distribuída

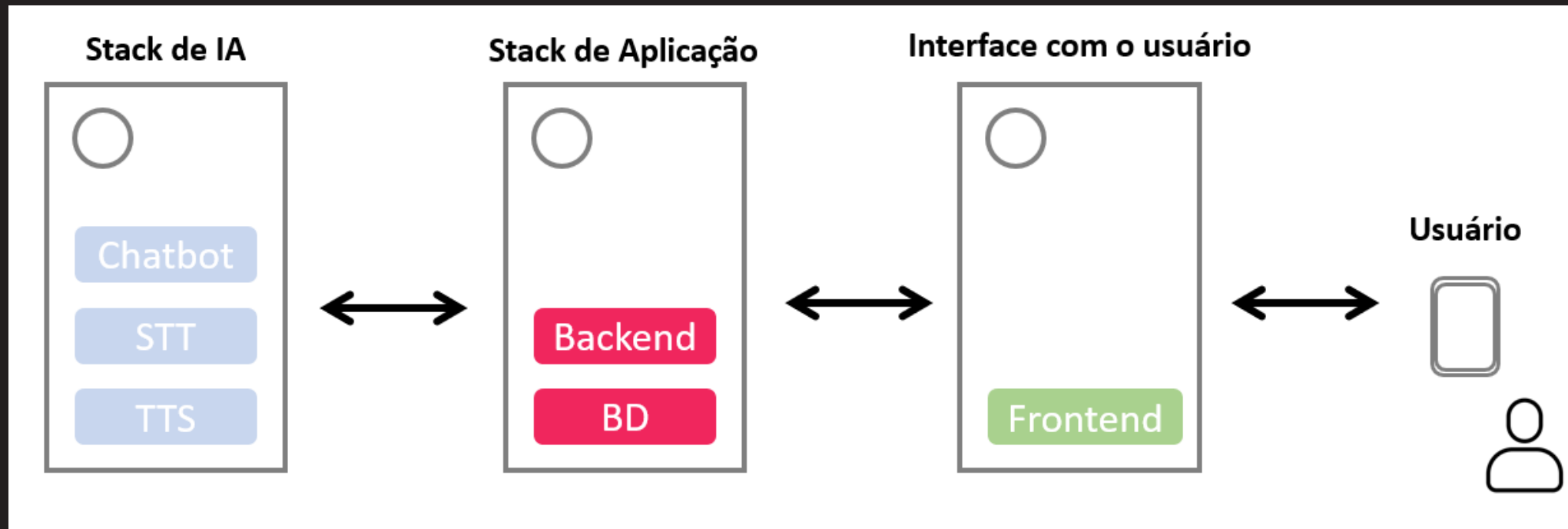


Dentro da **Nuvem da IBM**, podemos criar mais de um serviço: o **Watson Assistant** e o **Node-Red**. Estamos usando computadores da IBM.

Dentro da **Nuvem do Telegram**, criamos um serviço (programa) chamado **BotFather** dentro do aplicativo Telegram. Estamos usando computadores do Telegram.

# Integração de Serviços – Computação Distribuída

No contexto de computação distribuída (computação em nuvem) e integração de serviços, podemos ter vários servidores (sistemas computacionais) conectados. Por exemplo, uma das arquitetura possíveis é:



# 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 © 2023**

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

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