

# Redes de Computadores

Mininet e OpenDaylight

Prof. Rodrigo de Souza Couto

# Atenção

- Esta aula de laboratório ainda está na versão Beta, algumas atividades podem não funcionar
- Além disso, ainda é necessário um trabalho de formatação destes slides, que foram feitos a partir da colagem de vários slides de minha autoria

# PREPARAÇÃO DO AMBIENTE

# Preparação da VM utilizando uma appliance

- Instale o Virtualbox
  - <https://www.virtualbox.org/>
- Baixe a appliance disponibilizada pelo professor
  - Link no email ou transferência física
- No virtualbox, importe a appliance e siga as instruções
  - Arquivo->Importar Appliance Virtual

# Conectividade da VM com o hospedeiro

- No virtualbox, vá em Arquivo->Preferências->Rede
  - Na aba "Rede Exclusivas do Hospedeiro" verifique se já existe uma rede criada
    - Se existir, edite a rede para ter a seguinte configuração
      - IP: 192.168.241.1
      - Máscara: 255.255.255.0
      - DHCP habilitado
    - Se não existir, crie uma nova e coloque as configs acima
  - Na versão mais recente o procedimento é diferente:
    - Arquivo->Host Network manager
- Veja exemplo no slide 12
  - Dependendo da versão do Virtualbox, interface pode ser diferente. Veja outro exemplo de interface no slide 13

- Geral
- Entrada
- Atualizar
- Idioma
- Tela
- Rede**
- Extensões
- Proxy

**Rede**

Redes NAT Redes Exclusivas de Hospedeiro

VirtualBox Host-Only Ethernet Adapter #2

Detalhes de Rede Exclusiva de Hospedeiro (host-only)

Placa Servidor DHCP

Endereço IPv4: 192.168.56.1

Máscara de Rede IPv4: 255.255.255.0

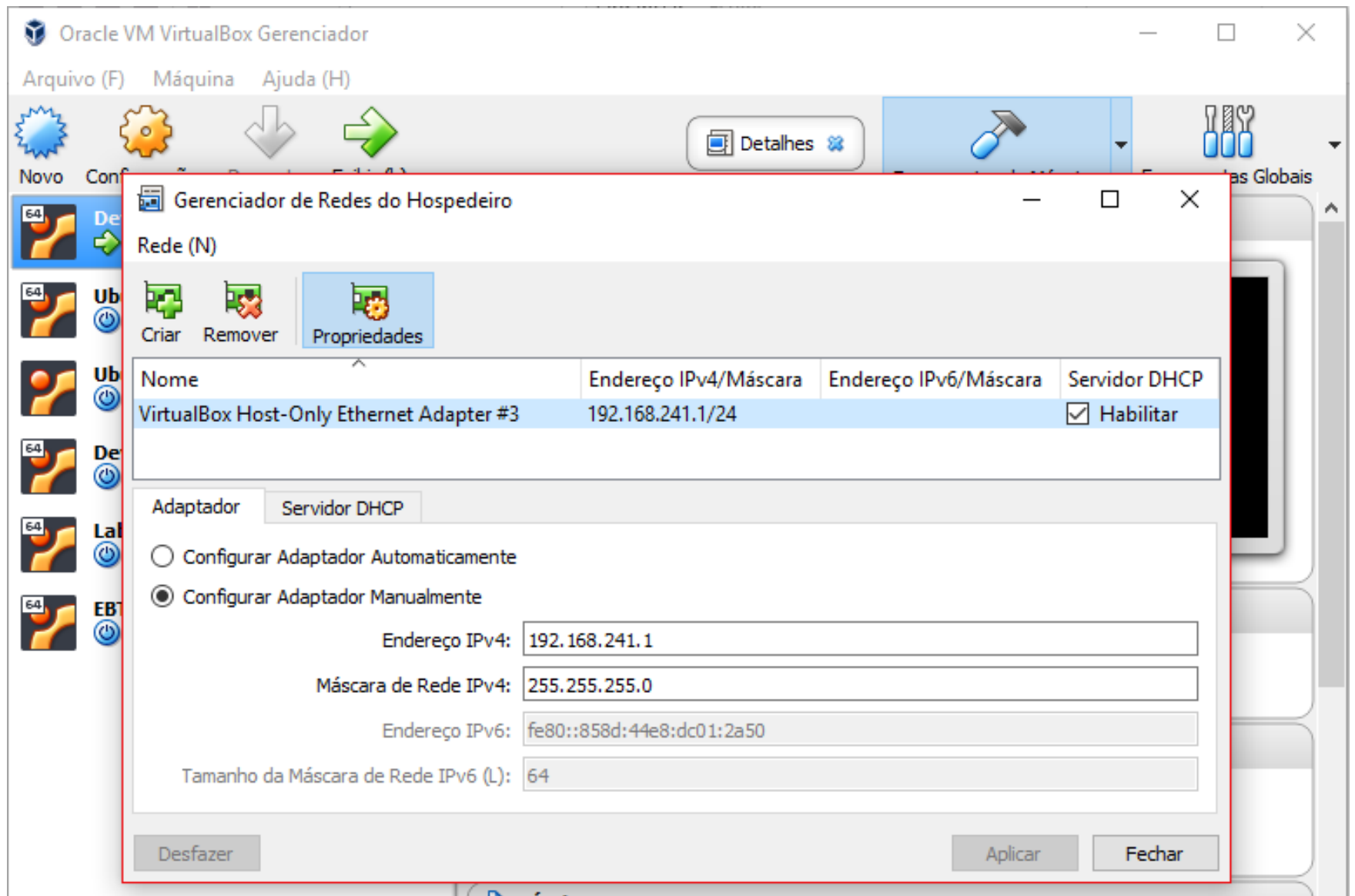
Endereço IPv6:

Tamanho da Máscara de Rede IPv6 (L):

OK

Cancel

Driver do Hospedeiro : Windows DirectSound  
Controladora: ICH AC97



# Conectividade da VM com o hospedeiro

- Com a VM desligada, clique em seu nome com o botão esquerdo e vá em Configurações->Rede->Adaptador 2
  - Marque a caixa "Habilitar Placa de Rede"
  - Na opção "conectado a", escolha "Placa de rede exclusiva de hospedeiro"
    - Após isso, escolha em "Nome" a rede configurada na etapa anterior
- Veja o exemplo no slide seguinte



Oracle VM VirtualBox Gerenciador

Arquivo (F) Máquina Ajuda (H)

Novo Configurações Descartar Exibir (h)

64 DevStack - Ubuntu Server Executando

Detalhes Snapshots

Geral Pré-Visualização

Nome: DevStack - Ubuntu

DevStack - Ubuntu Server - Configurações

Geral Sistema Monitor Armazenamento Áudio Rede Portas Seriais USB Pastas Compartilhadas Interface do Usuário

### Rede

Adaptador 1 Adaptador 2 Adaptador 3 Adaptador 4

Habilitar Placa de Rede

Conectado a: Placa de rede exclusiva de hospedeiro (host-only)

Nome: VirtualBox Host-Only Ethernet Adapter #2

▶ Avançado (D)

OK Cancel

# IP de comunicação da VM com o hospedeiro

- Inicie a VM e faça login
  - Usuário: curso
  - Senha: cursosdn
- Execute o comando `ifconfig` pra descobrir o IP que foi atribuído à VM. IP estará na faixa 192.168.241.X
  - `sudo ifconfig`
- O IP acima será utilizado para acesso SSH à VM e para acesso à interface web do ODL

# Acessando a VM no Linux

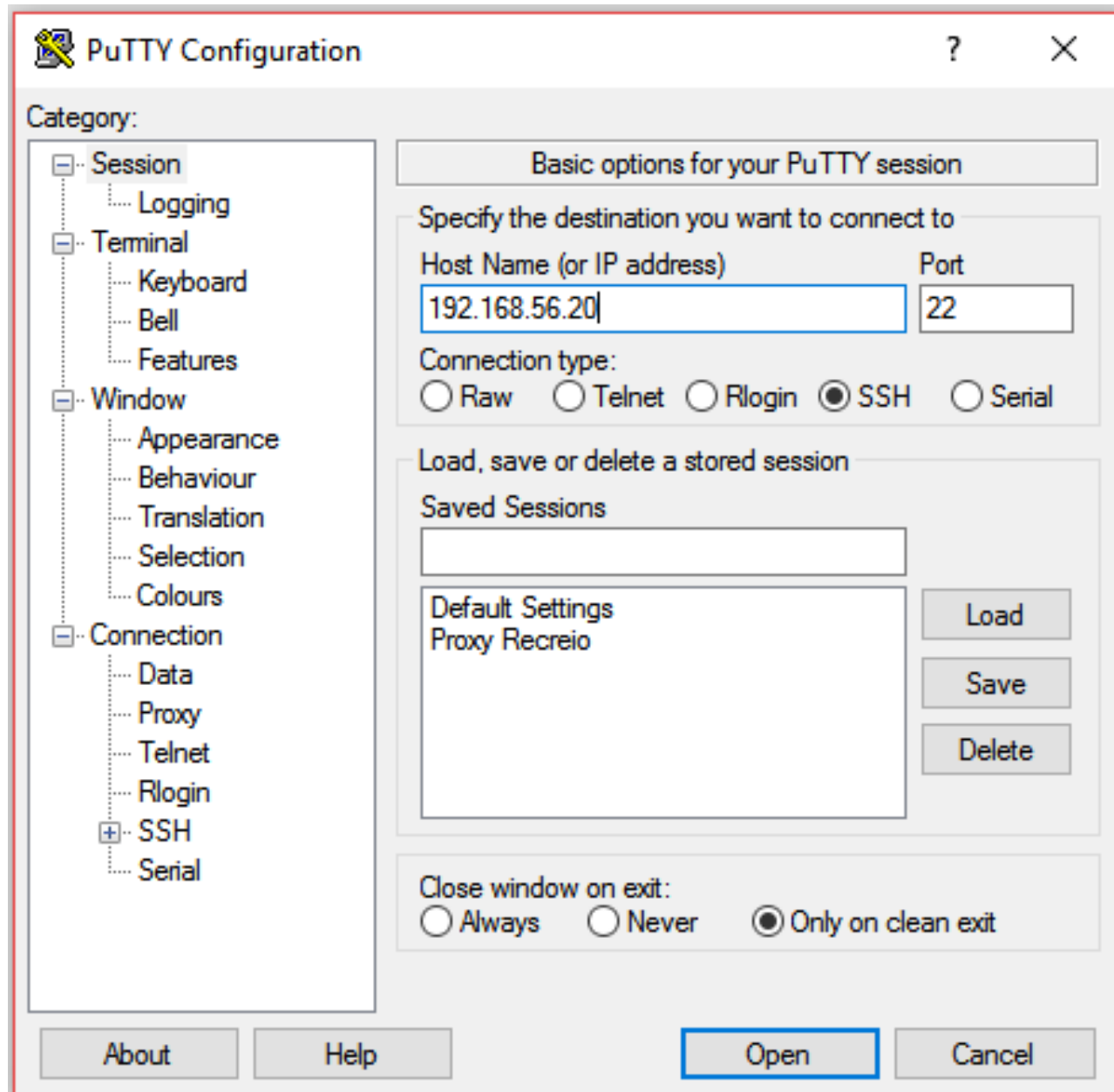
- Basta apenas usar o comando `ssh`

# Acessando a VM no Windows

- No hospedeiro, baixe o putty para realizar o ssh
  - <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
- Utilizando o putty, acesse a VM no endereço obtido anteriormente
  - Utilize seu login e sua senha quando for solicitado
  - Veja tela a seguir
- Para envio de arquivos do Windows à VM, use o WINSOCP
  - <https://winscp.net/eng/download.php>

A partir de agora, caso não seja dito o contrário, todos os passos consideram acesso à VM e por SSH

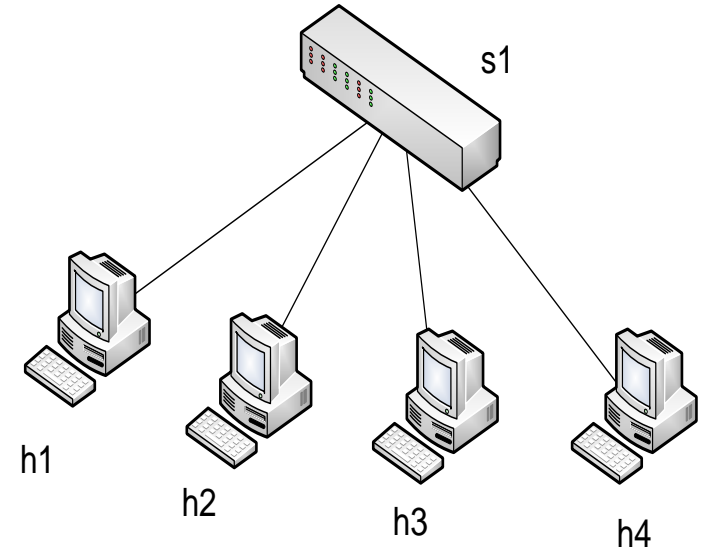
**ATENÇÃO!!!** Caso vc use esta opção, substitua todas as instruções relacionadas a “localhost” pelo IP de sua VM



UTILIZANDO O MININET

# Primeiro Contato

- Verifiquem ao arquivo `aula01_a.py`
  - Vejam os comentários
  - Execute o arquivo como
    - `sudo python aula01_a.py`



- Dicas
  - Se aparecer algum erro de que o controlador já está executando ou algo do tipo, limpe o mininet
    - `sudo mn -c`
  - Ou procure o processo do pox e mate-o
    - `"ps aux | grep pox"`, veja o nr do processo e mate com `"kill -9 nrProcesso"`

# Atividade 1

## Características dos enlaces

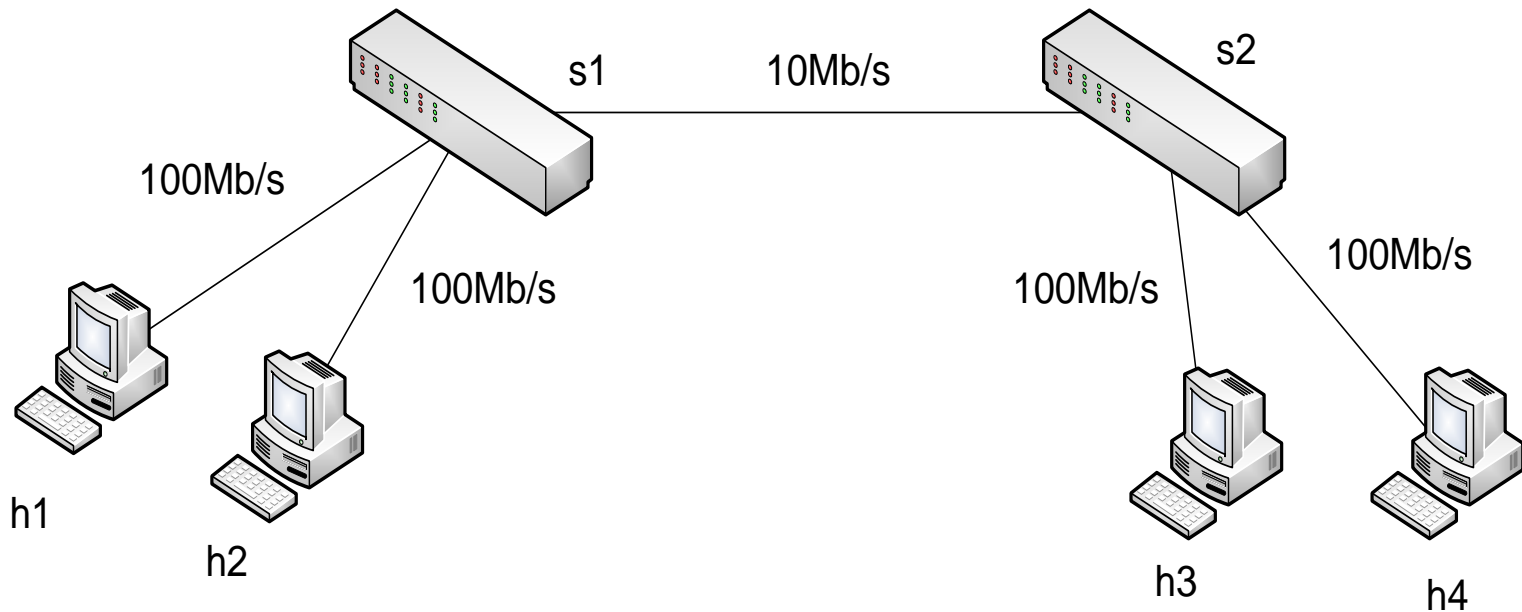
- Mude a banda dos enlaces e realize testes com iperf
- Mude a latência do links e realize testes com o ping



# Atividade 2

## Criação de topologia

- Modifique o arquivo para criar a seguinte rede



# Atividade 3

## Utilização do iperf

- Na rede da atividade anterior, teste a banda entre os diversos hosts

# Atividade 4

## Utilização de comandos reais

- Os hosts no mininet podem ser vistos como computadores reais. Assim, é possível executar comandos reais neles
- Veja o arquivo "aula1\_b.py", que consiste em uma modificação do "aula1\_a.py", só que com o iperf executando de forma "real"

# Atividade 5

## Iperf em paralelo

- Na rede da Atividade 2 coloque TODOS os links em 10Mb/s
- Após isso, realize dois teste de iperf em paralelo de h1 para h2 e h3 para h4
  - Anote os valores obtidos

# Atividade 6

## Aplicação L2 learning

- Substitua a aplicação hub pela l2 learning e repita o teste anterior
  - "forwarding.hub" -> "forwarding.l2\_learning"
- Após isso, realize dois testes de iperf em paralelo de h1 para h2 e h3 para h4
  - Compare os valores obtidos com a Atividade 5

# Atividade 7

## Parâmetros do Iperf

- Na linha de comando do Linux, faça "man iperf" para ver as possíveis opções e as utilize no programa

# Atividade 8

## Comando ping

- Substitua um dos pings do arquivo por um ping "real"
  - P.ex. 10 pings de h1 para h2
    - `"h1.cmd('ping -c 10 ' + h2_ip + ' > /tmp/resultping ')"`
- Explore as opções do ping com "man ping" e teste diversas opções

**COMEÇANDO A USAR O ODL**



# Inicialização do ODL

- Pare os serviços do openvswitch
  - `sudo service openvswitch-controller stop`
  - `sudo service openvswitch-switch stop`
- Entre no diretório e execute o Karaf
  - `cd ~/distribution-karaf-0.6.1-Carbon/`
  - `./bin/karaf`
- Já no karaf, instale a aplicação L2 switch
  - `feature:install odl-l2switch-switch-ui`
- Instale também a aplicação DLUX (interface gráfica de gerenciamento)
  - `feature:install odl-dluxapps-applications`

# Carregando módulos para RESTCONF e YANG

- Utilizados para realizar requisições e configuração de fluxos
  - Detalhes mais adiante
- No karaf, faça:
  - `feature:install odl-restconf-all`
  - `feature:install odl-dluxapps-yangui`
  - `feature:install odl-dluxapps-nodes`

# Testando com o Mininet

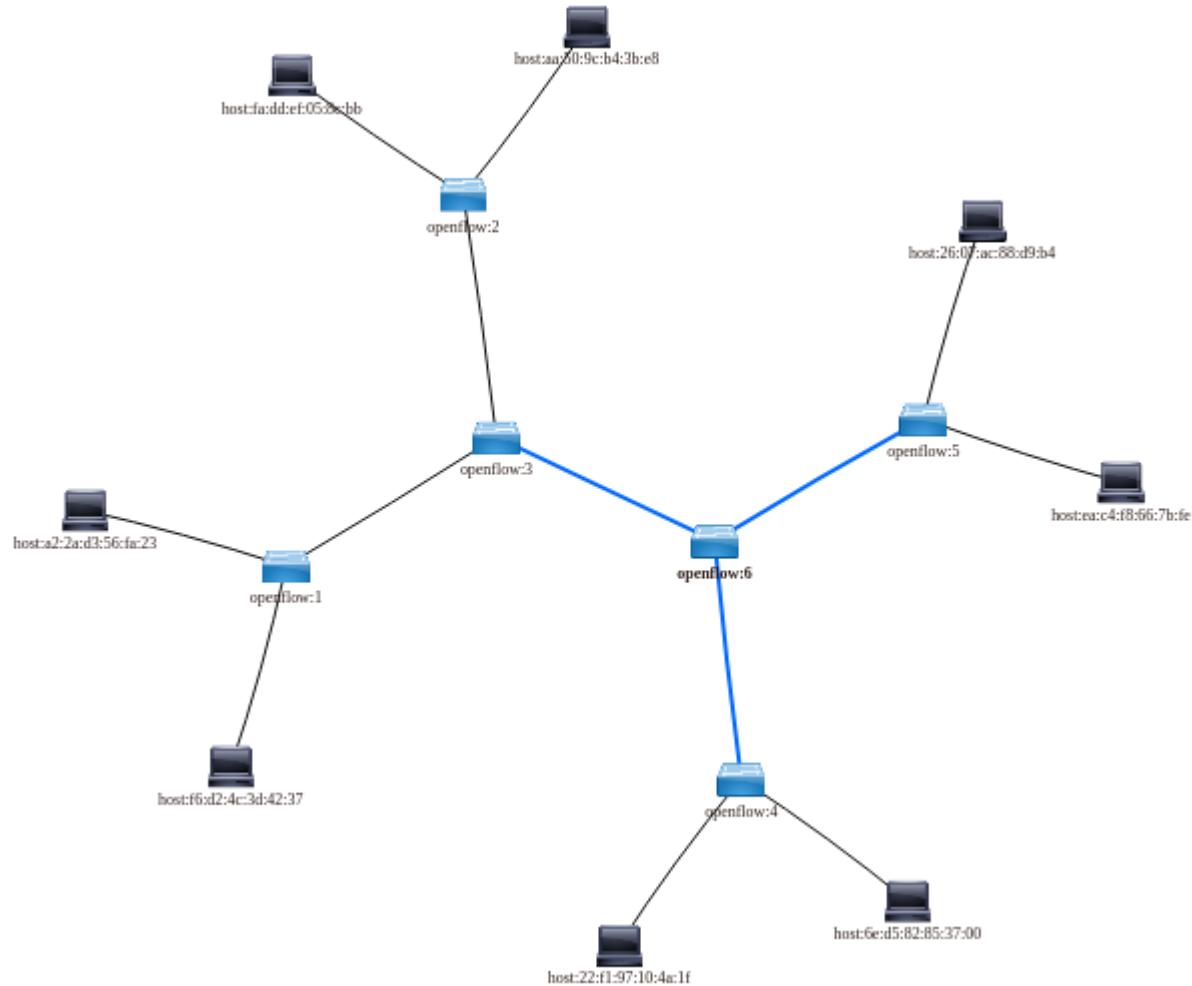
- Abra o arquivo `aula01_c.py` para verificar como chamar um controlador externo
- Abra um outro terminal e inicie o `openvswitch`
  - `sudo service openvswitch-switch start`
- Execute o arquivo `aula01_c.py`. Se os hosts pingarem, significa que o ODL foi bem instalado, bem como sua aplicação L2-learning. Apenas os primeiros pings serão perdidos.
  - `sudo python aula01_c.py`
- A aplicação acima não irá terminar. O programa entrará na linha de comando do Mininet. Isso será útil para deixar a rede ligada para executarmos a próxima etapa.

# Testando interface gráfica do ODL

- <http://localhost:8181/index.html>
  - Usuário: admin
  - Senha: admin
- Entre na aba de Topology
- Verifique se a topologia é a mesma que a especificada em aula01\_c.py

**LEMBRE-SE!!! Caso vc use VM c/ Ubuntu Server, substitua todas as instruções relacionadas a "localhost" pelo IP de sua VM**

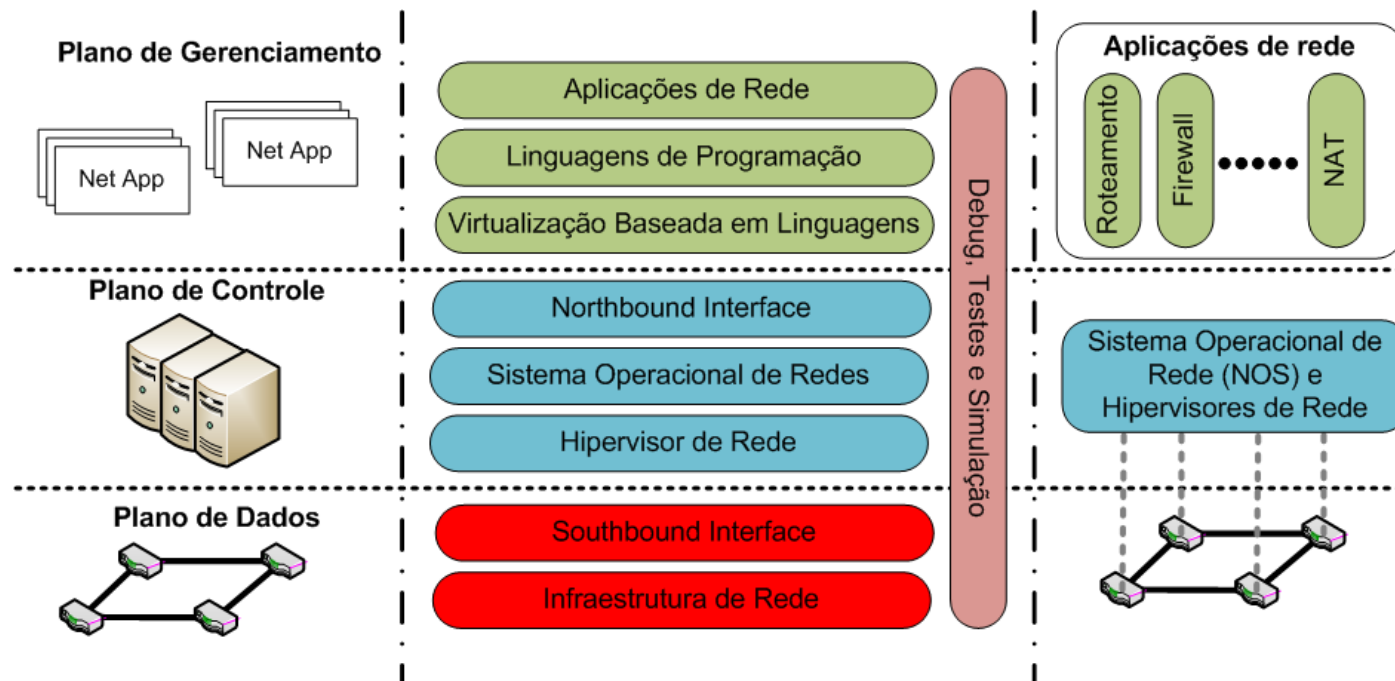
# Topologia do arquivo aula01\_c.py



# DESENVOLVIMENTO DE APLICAÇÕES NO OPENDAYLIGHT

# Relembrando a arquitetura SDN

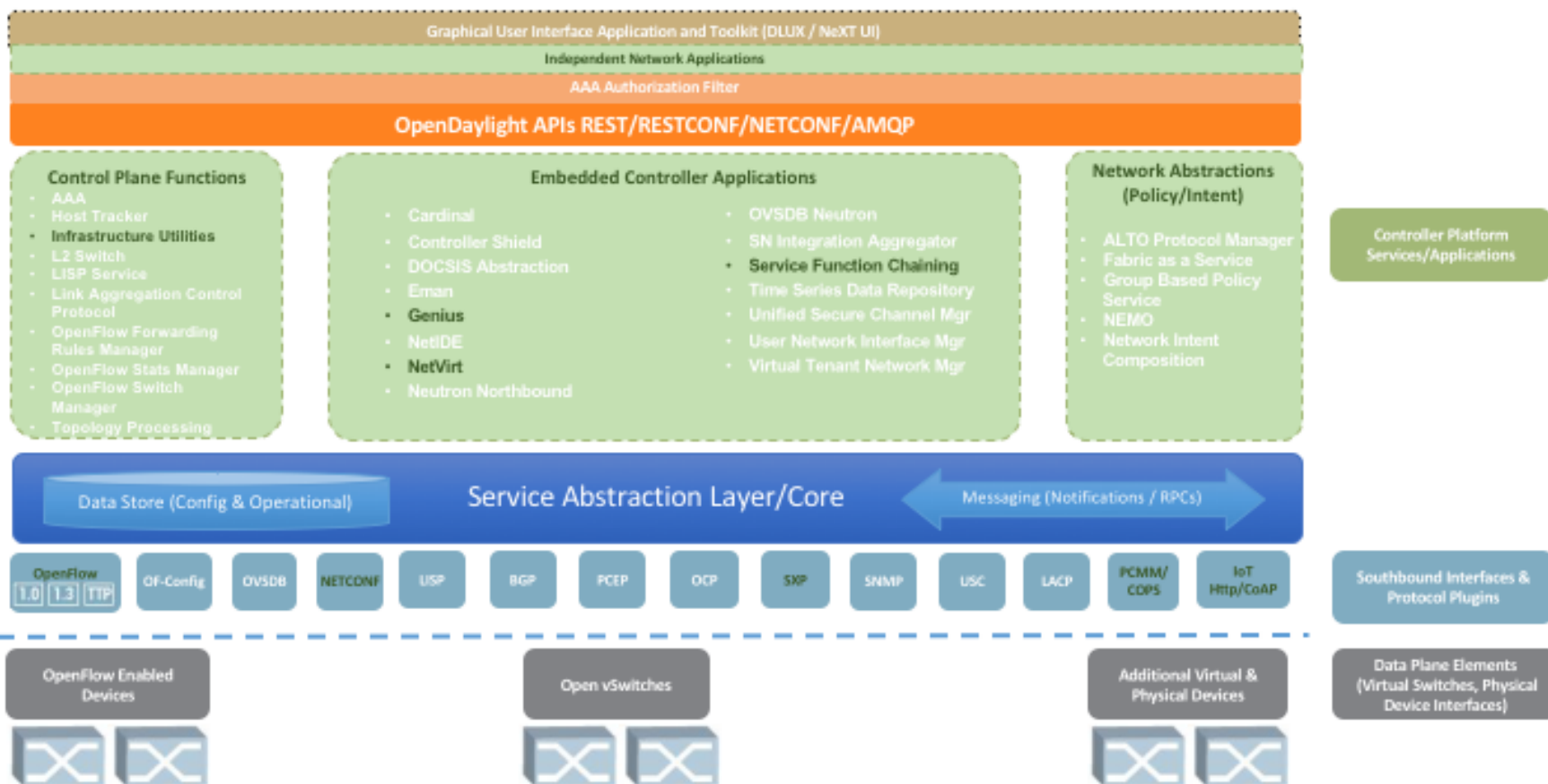
- Controlador deve, pelo menos, oferecer uma Northbound Interface
- Comandos da Northbound serão traduzidas para mensagens OpenFlow



# Arquitetura Opendaylight



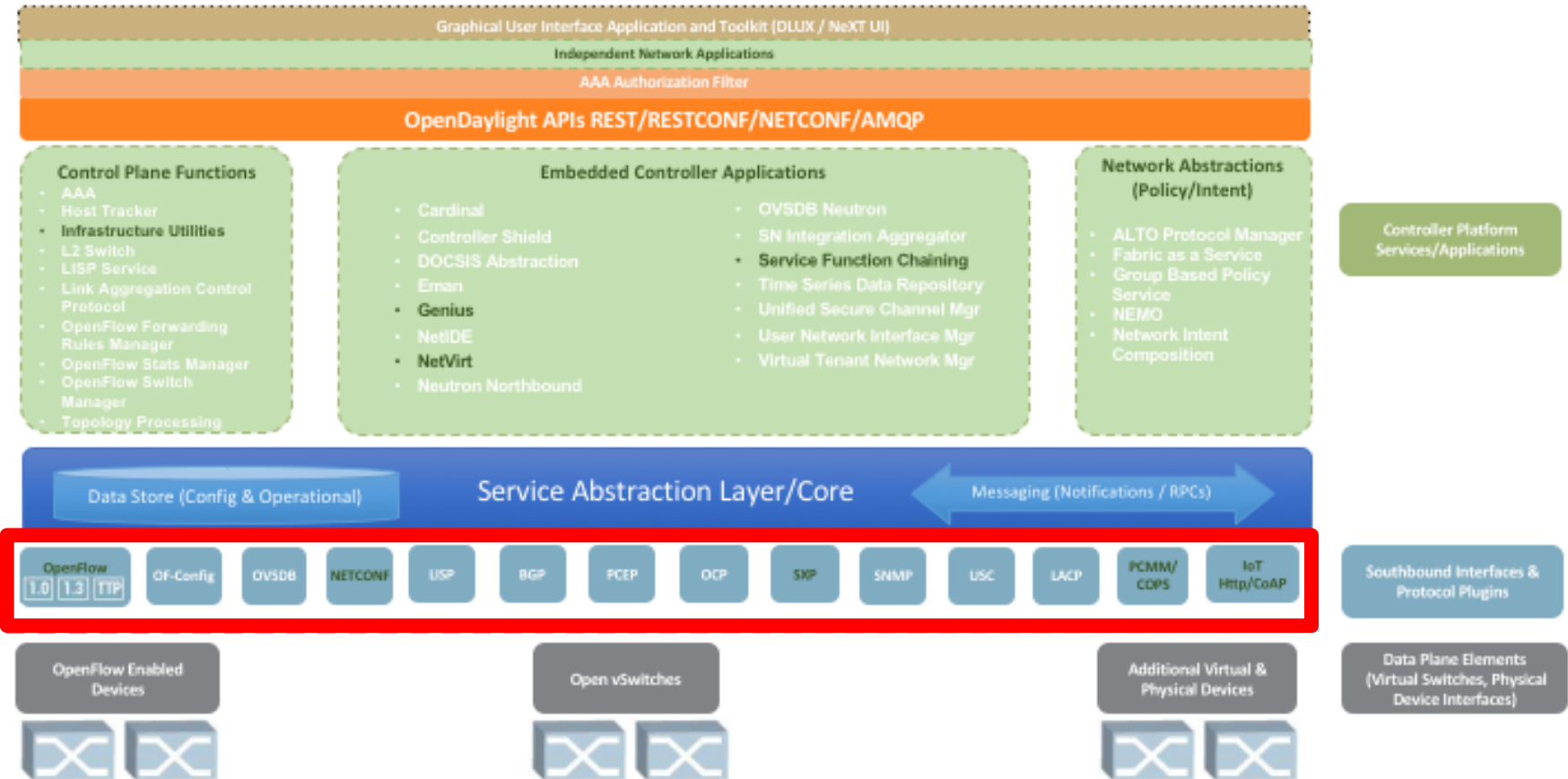
## Carbon: Proliferating Use Cases





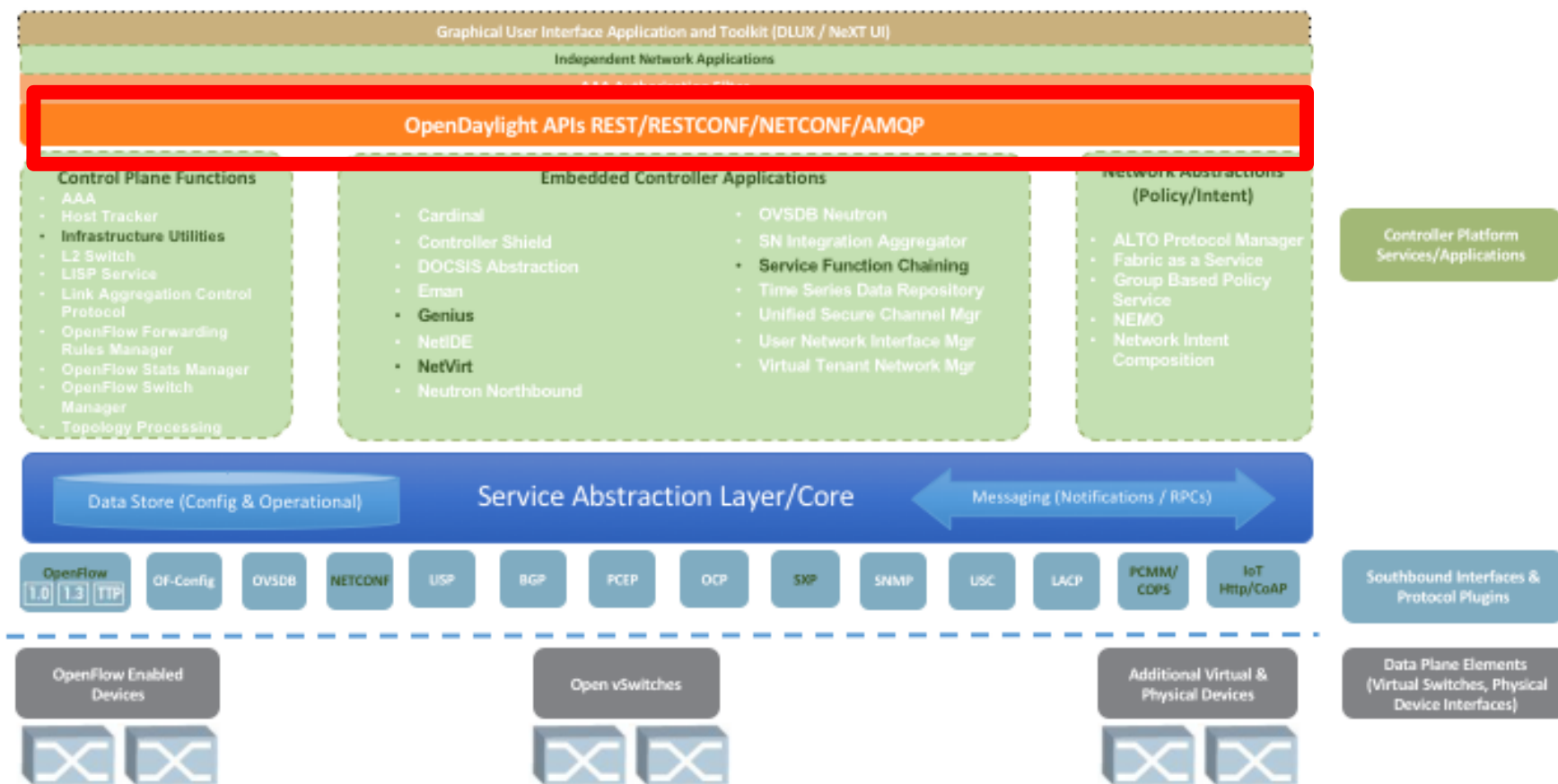
# Arquitetura Opendaylight

- Suporte a diferentes Southbound APIs
  - OpenFlow, BGP, SNMP, NETCONF, etc.

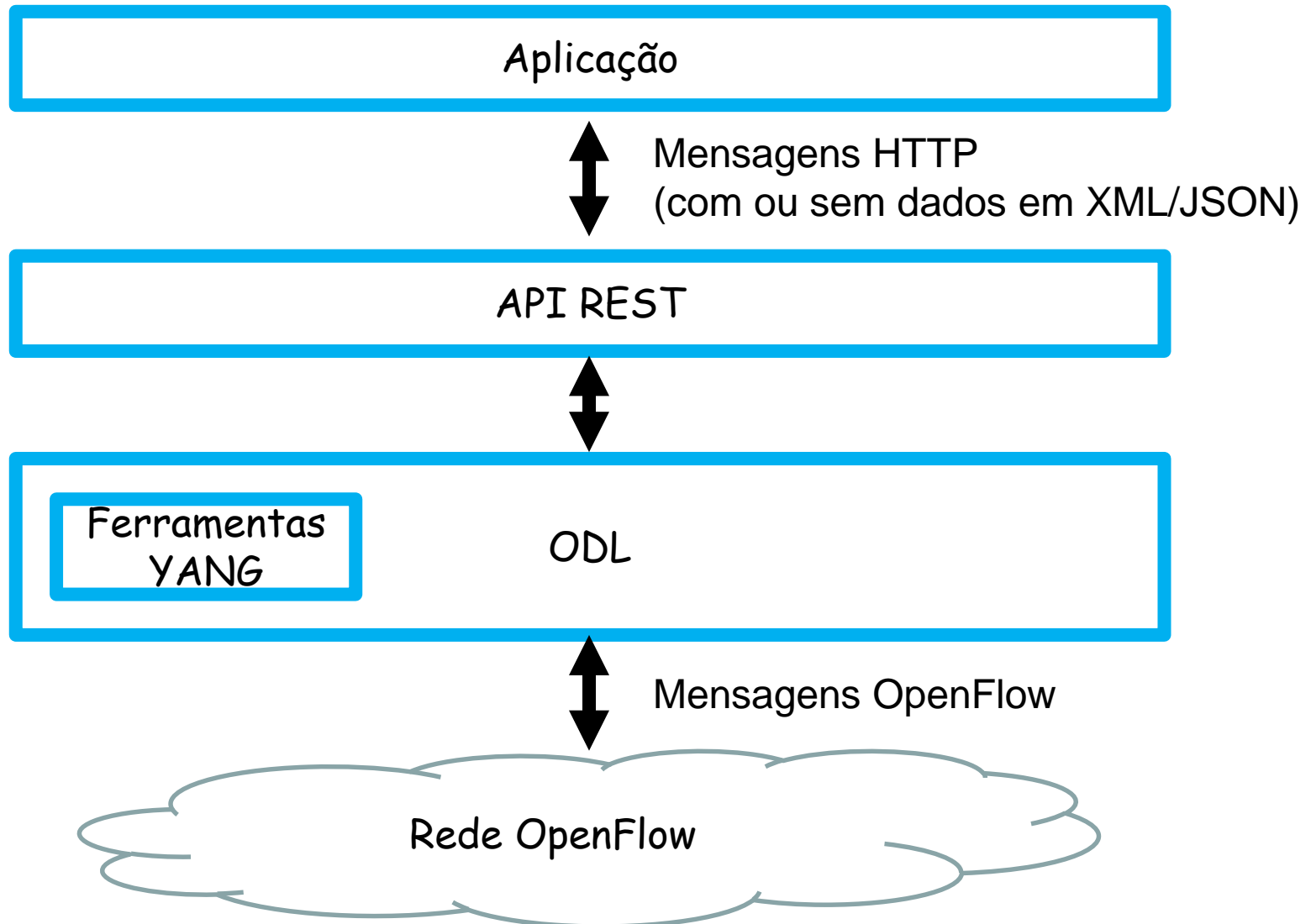


# Arquitetura Opendaylight

- Utilização de APIs para desenvolvimento de aplicações
  - Ex: API REST é acessada por requisições HTTP



# Visão Geral de uma aplicação utilizando a API REST



# Modelo YANG

- Linguagem para modelar interações com dispositivos de rede
  - P.ex. configuração remota de uma interface de rede
- Modelo definido pelo IETF para configuração do protocolo NETCONF
- Mensagens na linguagem YANG podem ser traduzidas para o formato JSON e XML
  - Envio por requisições HTTP
- Para gerenciar dispositivos de rede no OpenDaylight é possível enviar ao controlador mensagens XML no formato YANG
- Mais infos: <https://tools.ietf.org/html/rfc6020>

# O que é XML (eXtensible Markup Language)?

- Linguagem utilizada para descrever diferentes tipos de dados
- Exemplo: Receita de pão em XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

# Exemplo de mensagem

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <strict>false</strict>
  <instructions>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>0</order>
          <dec-nw-ttl/>
        </action>
      </apply-actions>
    </instruction>
  </instructions>
  <table_id>2</table_id>
  <id>124</id>
  <cookie_mask>255</cookie_mask>
  <installHw>false</installHw>
```

# Exemplo de mensagem

```
<installHw>>false</installHw>
  <match>
    <ethernet-match>
      <ethernet-type>
        <type>2048</type>
      </ethernet-type>
    </ethernet-match>
    <ipv4-destination>10.0.1.1/24</ipv4-destination>
  </match>
  <hard-timeout>12</hard-timeout>
  <cookie>1</cookie>
  <idle-timeout>34</idle-timeout>
  <flow-name>FooXf1</flow-name>
  <priority>2</priority>
  <barrier>>false</barrier>
</flow>
```

# Enviando requisições por URL

- Requisições acessam módulos do opendaylight
- Exemplos abaixo usam o módulo **opendaylight-inventory**
- Verifique fluxos de um switch
  - `http://<controller-ip>:8181/restconf/operational/opendaylight-inventory:nodes/node/{node-id}/table/{table-id}/flow/{flow-id}`
    - **Ex:** <http://localhost:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1/table/0/>
- Verifique entradas de um fluxo
  - `http://<controller-ip>:8181/restconf/operational/opendaylight-inventory:nodes/node/{node-id}/table/{table-id}/`
    - **Ex:** <http://localhost:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:1/table/0/flow/L2switch-71>

**Mensagens deste exemplo apenas recebem XML**



# Aplicações para configuração pela interface web

- Utilizadas para configuração manual dos fluxos
  - Úteis para realizar configurações pontuais na rede, sem necessidade de desenvolver uma aplicação
- Listam todos os módulos disponíveis
- YangUI
  - Facilita a criação de mensagem YANG
  - Abstrai do usuário o procedimento de envio de mensagens HTTP
  - Mais infos em <https://www.youtube.com/watch?v=X85SBwJIcFM>
- YangMan
  - Semelhante à YangUI, mas abstração é menor
  - Mesma ideia do POSTMAN, para requisições HTTP genéricas
  - Mensagens em JSON, mas a ideia é a mesma do XML

# Utilização da YangUI

- Facilita o envio e recebimento de comandos
- Verifique como realizar os comandos anteriores pela YangUI
  - <http://127.0.0.1:8181/index.html#/yangui/index>
- Teste comandos do tipo operational (isto é, apenas consulta)



# Entendendo o Fluxo:

## Definição de Fluxo

- Lembrando que cada fluxo é composto por
  - Correspondência ("match")
    - Define o que o switch tem q olhar no pacote para identificá-lo como pertencente a um fluxo
      - Ex: IP Destino: 10.0.0.X
  - Ações ("actions")
    - Define uma ou mais ações que deve ser realizadas com o pacote
      - Ex: Enviar pela porta física 1
  - Estatísticas
    - Contadores que mostram o uso de um fluxo
      - Ex: Quando pacotes deram "match"

# Entendendo o Fluxo: Aplicação L2 Switch

- A aplicação L2 switch instalada envia um pacote recebido para todas as interfaces do switch
  - Exceto a interface que chegou o pacote
  - Também envia para o Controlador
  - Útil para controle e quando não tem ainda nenhuma definição de fluxo na rede
    - **P.ex: Utilizada para construir a topologia**
- O exemplo a seguir analisa o fluxo de um switch diretamente conectado aos hosts

# Entendendo o Fluxo

Request sent successfully

GET /operational/opendaylight-inventory:nodes/node/ openflow:1 /flow-node-inventory:table/ 0 /flow/ L2switch-71

flow list **flow <id:L2switch-71>**

id **match** L2switch-71

in-port **openflow:1:2**

instructions

instruction list **instruction <order:0>**

order 0

instruction apply-actions-case

apply-actions

action list **action <order:0>** action <order:1> action <order:2>

order 0

action output-action-case

output-action

output-node-connector 1

max-length 65535

Trata como este fluxo se o pacote entrar pela interface 2

# Entendendo o Fluxo

GET /operational/.opendaylight-inventory:nodes/node/ openflow:1 /flow-node-inventory:table/ 0 /flow/ L2switch-71

Request sent successfully

flow list **A** flow <id:L2switch-71>

id 🔍 L2switch-71

match **A**

in-port openflow:1:2

instructions

instruction list instruction <order:0>

order 🔍 0

instruction apply-actions-case

apply-actions

**action list** action <order:0> action <order:1> action <order:2>

order 🔍 0

action output-action-case

output-action

output-node-connector 1

max-length 65535

**Lista de ações**

# Entendendo o Fluxo

GET /operational/.opendaylight-inventory:nodes/node/ openflow:1 /flow-node-inventory:table/ 0 /flow/ L2switch-71

Request sent successfully

flow list **A** flow <id:L2switch-71>

id L2switch-71

match **A**

in-port openflow:1:2

instructions

instruction list instruction <order:0>

order 0

instruction apply-actions-case

apply-actions

action list action <order:0> action <order:1> action <order:2>

order 0

action output-action-case

output-action

output-node-connector 1

max-length 65535

Cada ação diz para enviar para uma das portas (duas portas físicas e 1 Controlador)

# Entendendo o Fluxo

GET /operational/.opendaylight-inventory:nodes/node/ openflow:1 /flow-node-inventory:table/ 0 /flow/ L2switch-71

Request sent successfully

flow list **A** flow <id:L2switch-71>

id **A** L2switch-71

match **A**

in-port openflow:1:2

instructions

instruction list instruction <order:0>

order 0

instruction apply-actions-case

apply-actions

action list action <order:0> action <order:1> action <order:2>

order 0

action output-action-case

output-action

output-node-connector 1

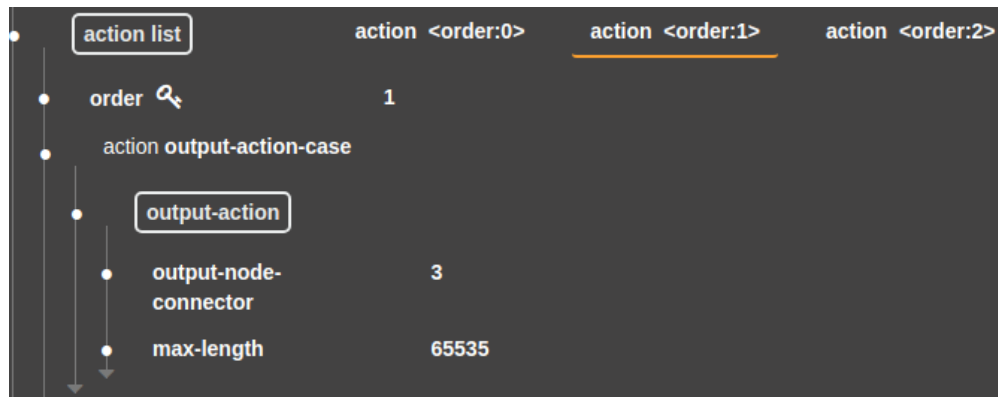
max-length 65535

Cada ação diz para enviar para uma das portas (duas portas físicas e 1 Controlador)



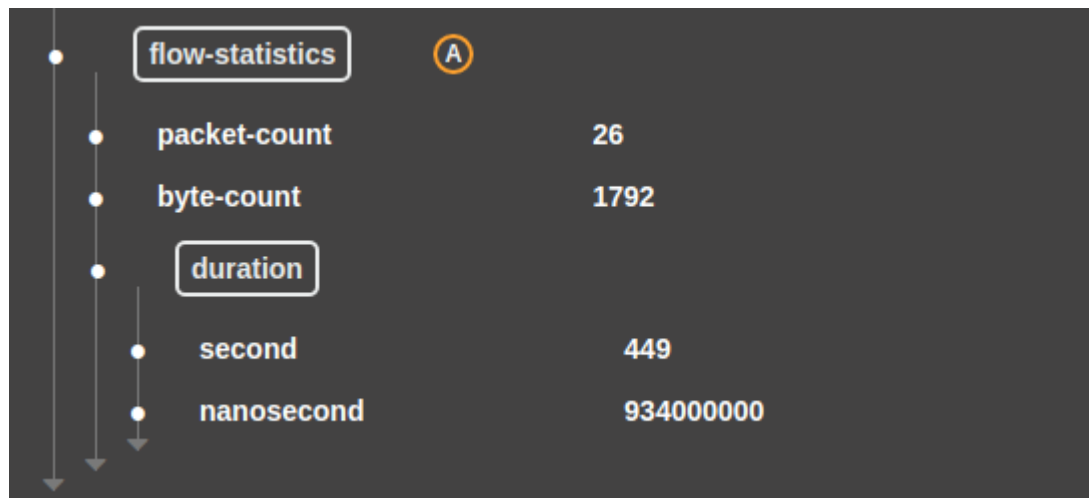
# Entendendo o Fluxo

- Outras ações deste fluxo



# Entendendo o Fluxo

- Estatísticas do Fluxo



The image shows a screenshot of a network flow statistics tool. On the left, there are three vertical lines with dots at the top and arrows pointing downwards, representing flow paths. The main content is a table with the following data:

flow-statistics	
packet-count	26
byte-count	1792
duration	
second	449
nanosecond	934000000

# Atividade

- Explore o módulo opendaylight-inventory
  - Com YANGUI ou YANGMAN
  - Realize comandos do tipo "operational"
    - Consulte infos de um switch
    - Consulte infos de um fluxo
- Veja exemplos de mensagens XML enviadas
  - [https://wiki.opendaylight.org/view/Editing\\_OpenDaylight\\_OpenFlow\\_Plugin:End\\_to\\_End\\_Flows:Example\\_Flows](https://wiki.opendaylight.org/view/Editing_OpenDaylight_OpenFlow_Plugin:End_to_End_Flows:Example_Flows)
  - Outros exemplos  
<http://www.brocade.com/content/html/en/user-guide/SDN-Controller-2.1.0-User-Guide/GUID-2C40188E-9379-4951-8345-D38FA8F9F2A0.html>

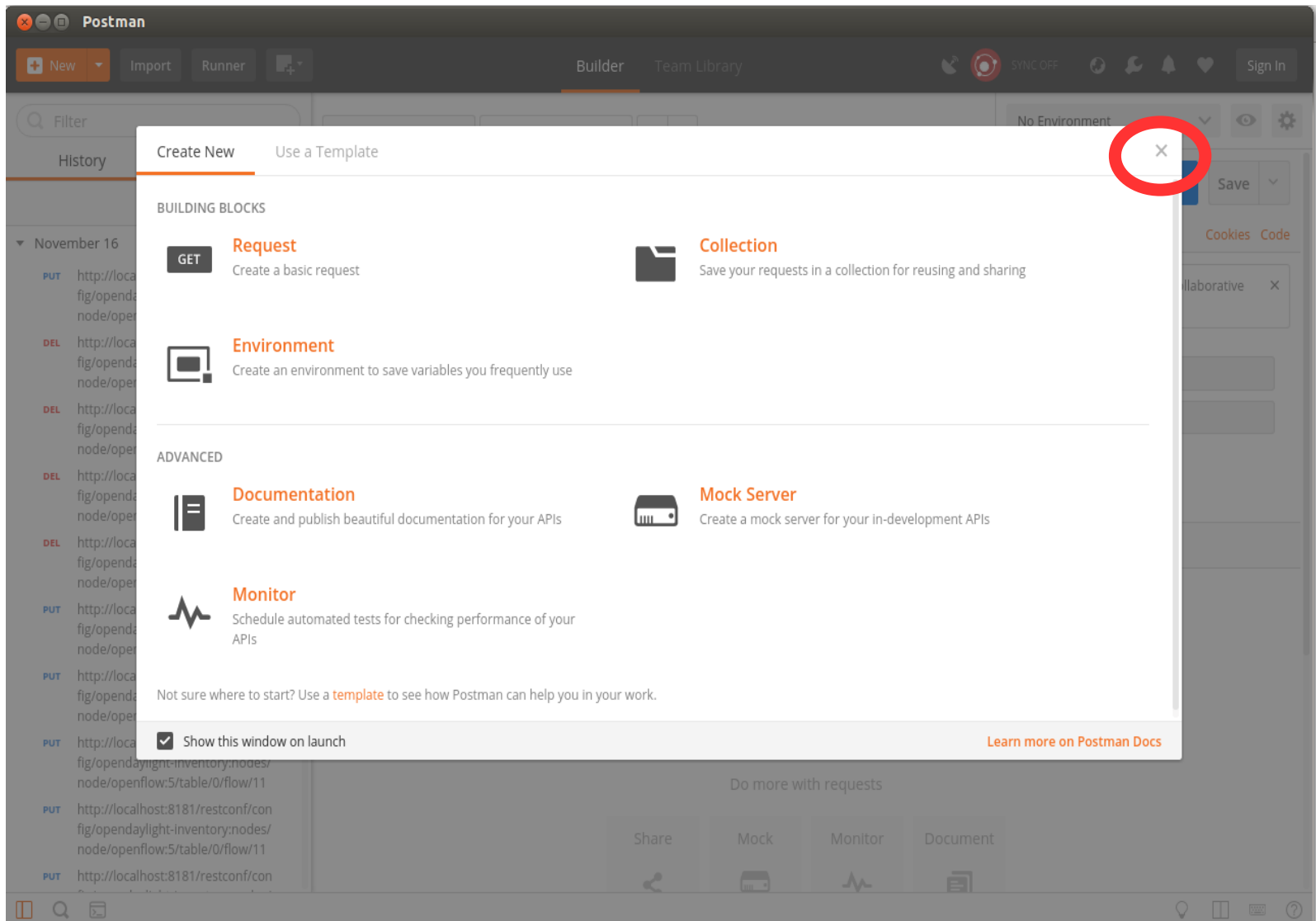
# ENVIO DE REQUISIÇÕES COM O POSTMAN

# Instalação

Baixe o postman em

<https://www.getpostman.com/>

# Ao abrir o Postman, ignore a janela que aparece, clicando no X



# Selecione uma requisição do tipo PUT

The screenshot displays the Postman application interface. On the left sidebar, the 'History' tab is active, showing a list of requests. The most recent request, a PUT method to the URL `http://localhost:8181/restconf/config/operdaylight-inventory:nodes/node/openflow:5/table/0/flow/10`, is highlighted. A red circle is drawn around the 'PUT' method label in the top bar of the request editor. The main panel shows the configuration for this request, including the 'Authorization' tab set to 'Basic Auth' with 'admin' as the username and a masked password. A 'Send' button is visible in the top right of the editor. At the bottom of the interface, there are buttons for 'Share', 'Mock', 'Monitor', and 'Document'.

# Coloque a URL desejada. Veja slide seguinte para mais detalhes

The image shows the Postman application interface. The main window is titled "Postman" and has a dark theme. At the top, there are tabs for "New", "Import", "Runner", and "Builder". The "Builder" tab is active. Below the tabs, there are several buttons: "New", "Import", "Runner", and a plus sign. To the right, there are icons for "SYNC OFF", a globe, a refresh icon, a bell, a heart, and a "Sign In" button.

The main area is divided into several sections. On the left, there is a "History" and "Collections" sidebar. The "History" section shows a list of requests, including a "PUT" request to `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`. The "Collections" section is empty. Below the sidebar, there is a "Filter" input field and a "Clear all" button.

The main workspace shows a "PUT" request configuration. The URL is `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`, which is highlighted with a red box. Below the URL, there are tabs for "Authorization", "Headers (1)", "Body", "Pre-request Script", and "Tests". The "Authorization" tab is active, showing a "Basic Auth" configuration. The "Username" field is set to "admin" and the "Password" field is masked with "....". There is a "Show Password" checkbox. A warning message says: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables".

At the bottom of the main workspace, there is a "Response" section. It contains a message: "Hit the Send button to get a response." and a "Send" button. Below this, there are buttons for "Share", "Mock", "Monitor", and "Document".



# Conteúdo da URL

- Exemplo:

<http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11>

- Na URL anterior tem-se

- localhost: nome ou IP da máquina na qual está o controlador
- config: mostra que se trata de uma requisição de configuração. Para apenas coletar dados usa-se “operational”
- .opendaylight-inventory:nodes: Aplicação utilizada. Na nossa disciplina a gente só usou essa
- openflow:5 : Nome do comutador a ser configurado. Na interface web do OpendayLight é possível ver os nomes
- 0 (antes do “table/”): Tabela 0 do comutador. Nas aplicações mais comuns usaremos só essa tabela
- 11 (antes de “flow/”): Id escolhido para o fluxo. Tome cuidado para não usar um id que já exista.

# Clique em “Authorization” para configurar a senha para acesso ao Controlador

The screenshot displays the Postman application interface. The main window shows a PUT request to the URL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The "Authorization" tab is selected and highlighted with a red box. The "Authorization" configuration is set to "Basic Auth". The "Username" field is filled with "admin" and the "Password" field is masked with ".....". A "Show Password" checkbox is present and unchecked. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)". The "Response" section is empty, displaying the message "Hit the Send button to get a response." and a blue "Send" button. At the bottom, there are buttons for "Share", "Mock", "Monitor", and "Document".

# Coloque o usuário e senha do seu Controlador

The image shows the Postman application interface. The main window displays a PUT request configuration for the URL `http://localhost:8181/restconf/config/operdaylight-inventory:nodes/node/openflow:5/table/0/...`. The request type is set to PUT. The Authorization tab is selected, and the authentication method is Basic Auth. The Username field is filled with 'admin' and the Password field is masked with dots. A red rectangular box highlights the Username and Password input fields. A warning message above the fields states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables". The interface also shows a list of request history on the left and a 'Send' button at the bottom right.

# Clique em “Preview Request” para inserir a senha no cabeçalho da requisição

The screenshot displays the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', 'Builder', and 'Team Library'. The main workspace shows a PUT request to the URL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The 'Authorization' tab is active, showing 'Basic Auth' selected. A red box highlights the 'Preview Request' button. The 'Response' section is empty, displaying the message 'Hit the Send button to get a response.' and a 'Send' button. The bottom of the interface features a 'Do more with requests' section with buttons for 'Share', 'Mock', 'Monitor', and 'Document'.

Postman

New Import Runner

Builder Team Library

Filter

History Collections

Clear all

Today

- PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`

November 16

- PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- DEL `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`

PUT `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`

Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Preview Request

Response

Hit the Send button to get a response.

Do more with requests

Share Mock Monitor Document

# Você verá que o contador do cabeçalho incrementou após a ação anterior

The screenshot displays the Postman interface for configuring a REST client request. The 'Headers' tab is selected and highlighted with a red box, indicating that the header count has incremented to (1). The request is a PUT method to the endpoint `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The 'Authorization' section is configured with 'Basic Auth' and fields for 'Username' (admin) and 'Password' (masked). A warning message is visible: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables". The 'Response' section is empty, displaying the message "Hit the Send button to get a response." and a "Send" button. The bottom of the interface shows utility buttons: "Share", "Mock", "Monitor", and "Document".

# Clique em “Body” para configurar o corpo da requisição (ou seja o XML enviado)

The screenshot displays the Postman application interface. At the top, the 'Builder' tab is active, showing a REST client request configuration. The URL is `http://localhost:8181/restconf/config/opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request method is `PUT`. The 'Body' tab is selected and highlighted with a red box. The 'Authorization' tab is also visible, showing 'Basic Auth' configuration with 'Username' set to 'admin' and 'Password' masked with dots. A warning message states: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The 'Response' section is empty, displaying the message: 'Hit the Send button to get a response.' At the bottom, there are buttons for 'Share', 'Mock', 'Monitor', and 'Document'.

# Selecione o tipo de dado “raw”

The image shows the Postman application interface. On the left, there is a sidebar with a search filter and a history list. The main area is divided into a top bar with navigation buttons (New, Import, Runner) and a central workspace. The workspace shows a REST client configuration for a PUT request to `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The 'Body' tab is selected, and the 'raw' radio button is highlighted with a red box. Other options include 'form-data', 'x-www-form-urlencoded', and 'binary'. The 'XML (application/xml)' dropdown is also visible. Below the body configuration, there is a 'Response' section with a message: 'Hit the Send button to get a response.' and a blue 'Send' button.

Postman

New Import Runner

Builder Team Library

Filter

History Collections

Clear all

Today

- PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`

November 16

- PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`
- DEL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- DEL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`
- PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11`

PUT `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...` Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

form-data x-www-form-urlencoded raw binary XML (application/xml)

Response

Hit the Send button to get a response.

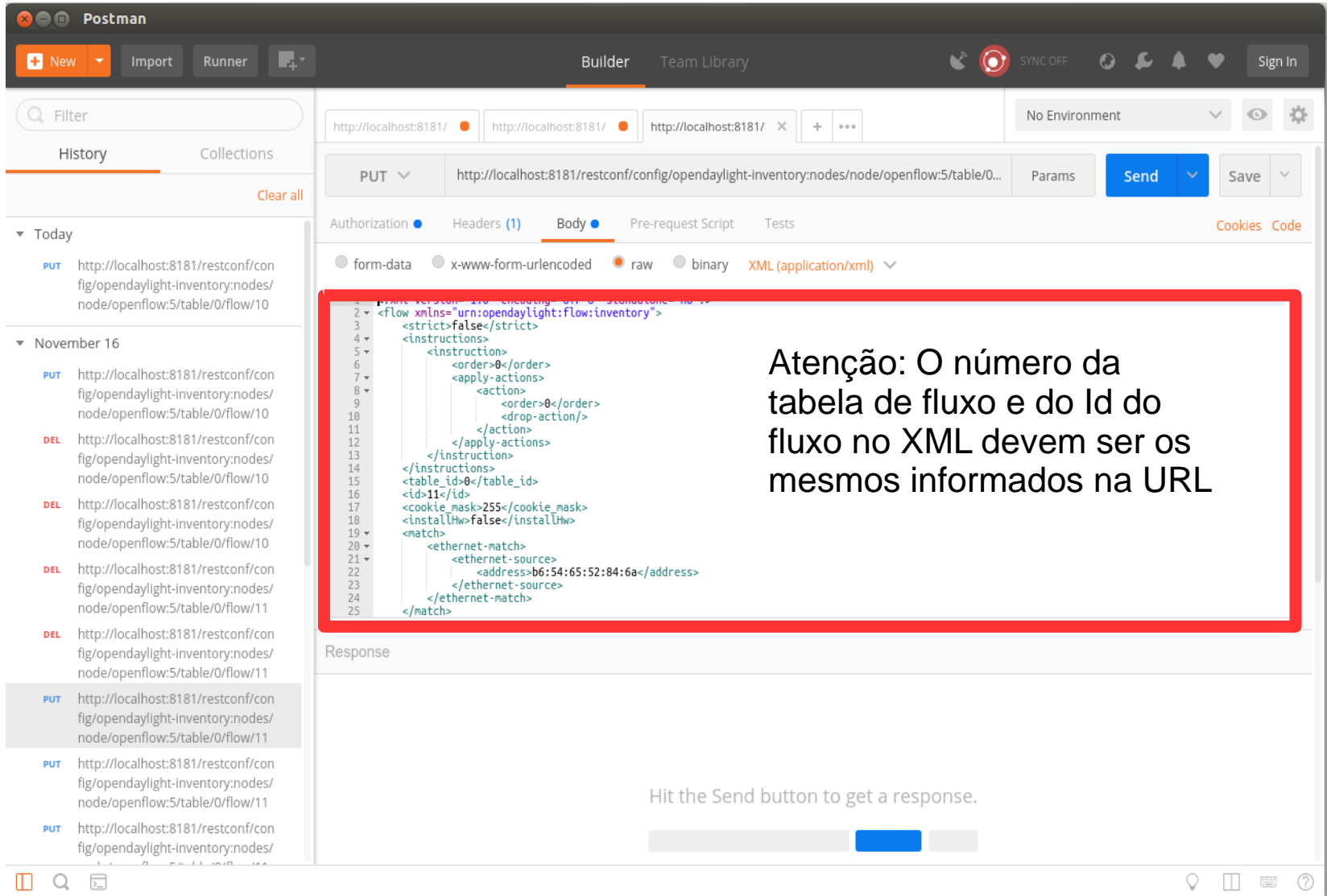
Send

# Em seguida, selecione o tipo XML (application/xml)

The screenshot displays the Postman application interface. On the left, a sidebar shows a history of requests, with the most recent one highlighted. The main workspace is configured for a PUT request to the URL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/flow/10`. The 'Body' tab is selected, and the content type is set to 'XML (application/xml)'. The 'Send' button is visible, and the response area at the bottom contains the text 'Hit the Send button to get a response.'



# Insira seu XML. Mais detalhes no slides seguintes



The screenshot shows the Postman interface with a RESTCONF PUT request. The URL is `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is XML, and the `id` attribute is set to `11`. A red box highlights the XML body, and a text overlay reads: "Atenção: O número da tabela de fluxo e do Id do fluxo no XML devem ser os mesmos informados na URL".

```
2 <flow xmlns="urn:opendaylight:flow:inventory">
3   <strict>false</strict>
4   <instructions>
5     <instruction>
6       <order>0</order>
7       <apply-actions>
8         <action>
9           <order>0</order>
10          <drop-action/>
11        </action>
12      </apply-actions>
13    </instruction>
14  </instructions>
15  <table_id>0</table_id>
16  <id>11</id>
17  <cookie_mask>255</cookie_mask>
18  <install_h>false</install_h>
19  <match>
20    <ethernet-match>
21      <ethernet-source>
22        <address>b6:54:65:52:84:6a</address>
23      </ethernet-source>
24    </ethernet-match>
25  </match>
```

Atenção: O número da tabela de fluxo e do Id do fluxo no XML devem ser os mesmos informados na URL

# Como construir um XML

É possível encontrar exemplos de XML para o OpenDaylight no seguinte link:

[https://wiki.opendaylight.org/view/Editing\\_OpenDaylight\\_OpenFlow\\_Plugin:End\\_to\\_End\\_Flows:Example\\_Flows](https://wiki.opendaylight.org/view/Editing_OpenDaylight_OpenFlow_Plugin:End_to_End_Flows:Example_Flows)

Caso o link esteja inacessível, use o pdf comandos.pdf que está no arquivo do laboratório em questão

**CUIDADO 1:** É preciso enviar um XML com prioridade mais alta do que o da aplicação switch l2 executando (que, geralmente é 2)

**CUIDADO 2:** O idle timeout e o hard timeout dos exemplos é muito baixo. Coloque timeout maiores na sua requisição.

# Exemplos

Veja os exemplos no arquivo do laboratório dropIP.xml e dropMac.xml que configuram fluxos para descartar quadros pelo IP e pelo MAC, respectivamente

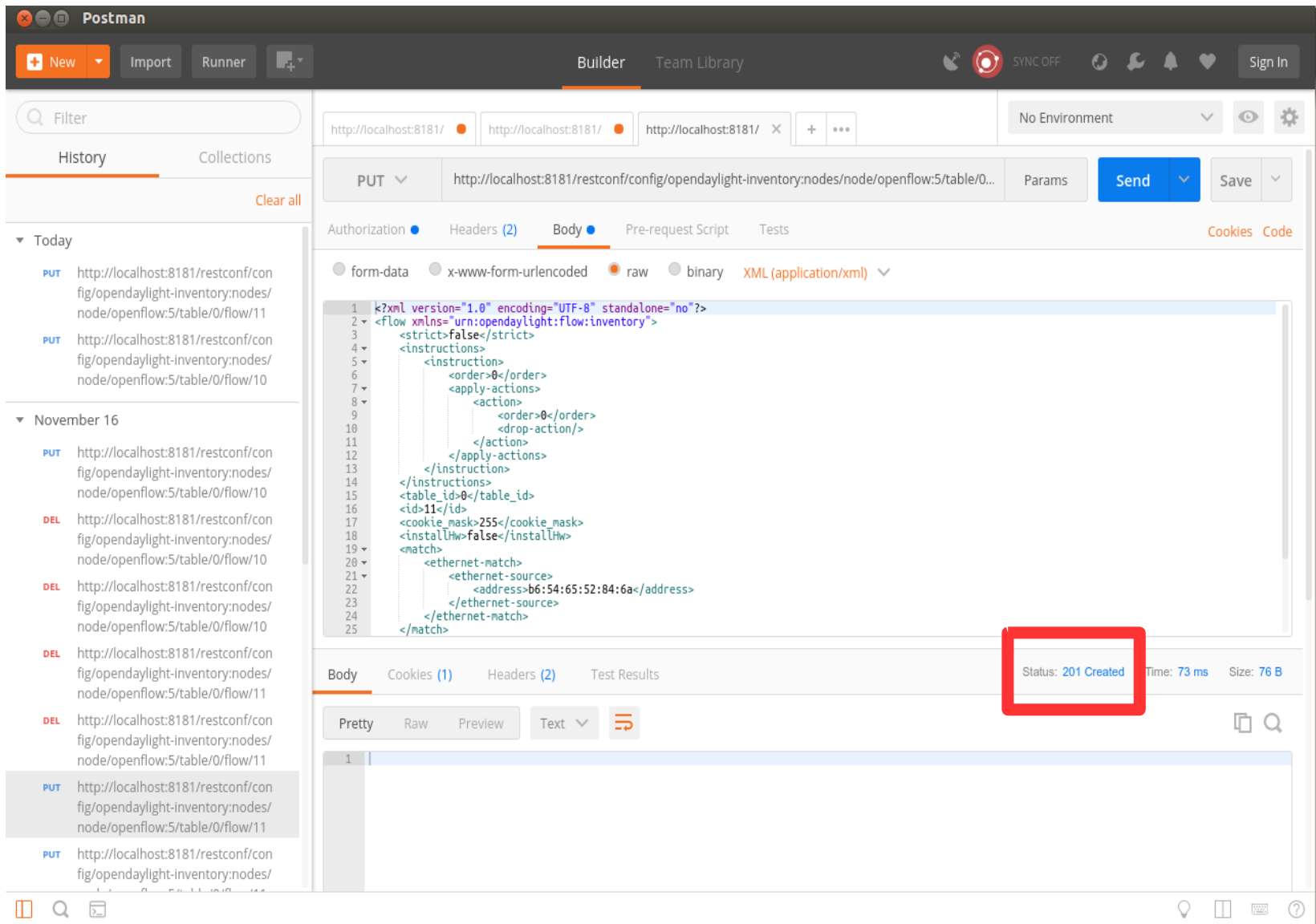
# Envie a requisição

The screenshot shows the Postman application interface. The main area displays a REST client request configuration for a PUT method to the URL `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is in raw XML format, containing an XML document with the following structure:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <flow xmlns="urn:opendaylight:flow:inventory">
3   <strict>false</strict>
4   <instructions>
5     <instruction>
6       <order>0</order>
7       <apply-actions>
8         <action>
9           <order>0</order>
10          <drop-action/>
11        </action>
12      </apply-actions>
13    </instruction>
14  </instructions>
15  <table_id>0</table_id>
16  <id>11</id>
17  <cookie_mask>255</cookie_mask>
18  <install_h>false</install_h>
19  <match>
20    <ethernet-match>
21      <ethernet-source>
22        <address>b6:54:65:52:84:6a</address>
23      </ethernet-source>
24    </ethernet-match>
25  </match>
26 </flow>
```

The 'Send' button is highlighted with a red box. Below the request body, the 'Response' section is empty, with the text 'Hit the Send button to get a response.' and a blue 'Send' button.

# Caso o fluxo não exista e a requisição tenha sucesso, você receberá uma resposta com código 201



The screenshot displays the Postman interface for a REST client. The main window shows a PUT request to the endpoint `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is an XML document defining a flow configuration. The response status is `201 Created`, which is highlighted with a red box. The response time is 73 ms and the size is 76 B.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:.opendaylight:flow:inventory">
  <strict>false</strict>
  <instructions>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>0</order>
          <drop-action/>
        </action>
      </apply-actions>
    </instruction>
  </instructions>
  <table_id>0</table_id>
  <id>11</id>
  <cookie_mask>255</cookie_mask>
  <install_hw>false</install_hw>
  <match>
    <ethernet-match>
      <ethernet-source>
        <address>b6:54:65:52:84:6a</address>
      </ethernet-source>
    </ethernet-match>
  </match>
</flow>
```

Status: 201 Created Time: 73 ms Size: 76 B

# Caso o fluxo exista, ele será alterado. Assim, você receberá uma resposta com código 200

The screenshot shows the Postman interface with a PUT request to the endpoint `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is XML, and the response status is 200 OK, highlighted with a red box.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <flow xmlns="urn:opendaylight:flow:inventory">
3   <strict>false</strict>
4   <instructions>
5     <instruction>
6       <order>0</order>
7       <apply-actions>
8         <action>
9           <order>0</order>
10          <drop-action/>
11        </action>
12      </apply-actions>
13    </instruction>
14  </instructions>
15  <table_id>0</table_id>
16  <id>11</id>
17  <cookie_mask>255</cookie_mask>
18  <install_hw>false</install_hw>
19  <match>
20    <ethernet-match>
21      <ethernet-source>
22        <address>b6:54:65:52:84:6a</address>
23      </ethernet-source>
24    </ethernet-match>
25  </match>
```

Status: 200 OK Time: 46 ms Size: 71 B

# Caso tenha alguma erro na sua URL ou no seu XML, você receberá um erro 400

The screenshot shows the Postman interface with a PUT request to `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is XML. The error message in the response body is:

```
{
  "errors": {
    "error": [
      {
        "error-type": "protocol",
        "error-tag": "invalid-value",
        "error-message": "The value '11' for key 'id' specified in the URI doesn't match the value '12' specified in the message body."
      }
    ]
  }
}
```

The status bar at the bottom right of the Postman window shows "Status: 400 Bad Request" (highlighted with a red box), "Time: 42 ms", and "Size: 325 B".

# O Motivo do erro pode ser visto no corpo da resposta

The screenshot shows the Postman interface with a PUT request to `http://localhost:8181/restconf/config/.opendaylight-inventory:nodes/node/openflow:5/table/0/...`. The request body is XML, and the response body is JSON. A red box highlights the error message in the response body.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <strict>false</strict>
  <instructions>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>0</order>
          <drop-action/>
        </action>
      </apply-actions>
    </instruction>
  </instructions>
  <table_id>0</table_id>
  <id>12</id>
  <cookie_mask>255</cookie_mask>
  <installHw>false</installHw>
  <match>
    <ethernet-match>
      <ethernet-source>
        <address>b6:54:65:52:84:6a</address>
      </ethernet-source>
    </ethernet-match>
  </match>
</flow>
```

```
{
  "errors": {
    "error": [
      {
        "error-type": "protocol",
        "error-tag": "invalid-value",
        "error-message": "The value '11' for key 'id' specified in the URI doesn't match the value '12' specified in the message body."
      }
    ]
  }
}
```



# Verificação do Fluxo

- Realize a mesma requisição, mas escrita como “operational” em um browser ou no POSTMAN com a opção GET, ao invés de PUT:

<http://localhost:8181/restconf/operational/opendaylight-inventory:nodes/node/openflow:5/table/0/flow/11>

- Se o controlador retornar o XML do fluxo correto, isso confirma que o fluxo foi instalado com sucesso