

# Netkit - Sistema para Emulação de Redes de Computadores

Prof. Thiago Guimarães Tavares - [thiagogmta@gmail.com](mailto:thiagogmta@gmail.com) - IFTO Campus Palmas

## Objetivos do capítulo

Ao final deste capítulo você deverá ser capaz de:

- Entender o que é e para que serve o Netkit
- Aprender a realizar a instalação da ferramenta
- Criar uma rede ponto a ponto e um laboratório com 4 hosts e um hub

## Apresentação

### O que é?

Netkit é um sistema para emulação de redes de computadores que possibilita a criação de instâncias virtuais de vários dispositivos de uma rede. Cada dispositivo virtual possui:

- Um console (janela do xterminal).
- 32mb de consumo de memória ram. Esse valor pode ser alterado, mas é setado por padrão.
- Um sistema de arquivos armazenado em um único arquivo no disco com extensão .disk
- Uma ou várias interfaces de rede. Cada interface de rede pode ser conectada a um *domínio de colisão* (hub/switch).

### Para que serve?

Redes de computadores são ambientes tipicamente complexos e são compostos por vários hosts, serviços, interfaces e protocolos em funcionamento. Por conta da diversidade de equipamentos e serviços o estudo prático da disciplina se torna dificultoso. Recriar um ambiente de redes real envolve alto custo. Existem alternativas para a simulação desses ambientes que possibilitam o estudo de várias ferramentas e protocolos. O Netkit é uma dessas ferramentas e serve justamente para criarmos um ambiente de redes de teste que necessita, para seu funcionamento, apenas de um computador com sistema operacional linux.

### Como funciona?

A idéia básica do Netkit é que se possa criar várias máquinas virtuais de baixo consumo que possam ser conectadas a 'domínios de colisão virtuais' ou seja ligadas a hubs ou switches. Podemos manipular cada host através de um terminal do xterminal onde teremos permissão de administrador para realização de qualquer configuração, inclusive a configuração de servidores, roteadores, firewall etc.

# Exemplificação

## NOTE

O Netkit funciona em ambientes linux. Para os nossos teste foi utilizado a distribuição Ubuntu.

## Instalação

Inicialmente deve-se realizar o download dos três arquivos necessários para o funcionamento do Netkit. sendo eles:

- Netkit - Pacote com os aruivos para funcionamento do Netkit
- Netkit-filesystem - Pacote que contem uma imagem minima do sistema de arquivos da distribuição Debian incluindo vários pacotes instalados.
- Netkit-kernel - Módulo do kernel que permitirá emular o kernel do host real em uma istância do xtermianl.

Os arquivos se encontram em: <http://www.netkit.org/> e para os testes deste material foi utilizado a ultima versão estável a 2.8.

## Baixando os pacotes

```
$ cd ~
$ wget http://wiki.netkit.org/download/netkit/netkit-2.8.tar.bz2
$ wget http://wiki.netkit.org/download/netkit-filesystem/netkit-filesystem-i386-F5.2.tar.bz2
$ wget http://wiki.netkit.org/download/netkit-kernel/netkit-kernel-i386-K2.8.tar.bz2
```

## Descompactando

```
$ tar -xjSf netkit-2.8.tar.bz2
$ tar -xjSf netkit-filesystem-i386-F5.2.tar.bz2
$ tar -xjSf netkit-kernel-i386-K2.8.tar.bz2
```

## Exportando Variáveis de Ambiente

As variáveis de ambiente são responsáveis por armazenar os caminhos dos arquivos de configuração do netkit

```
export NETKIT_HOME=~/.netkit
export PATH=$PATH:$NETKIT_HOME/bin
export MANPATH=$NETKIT_HOME/man
```

## Salvando em um arquivo

Podemos salvar as instruções que exportam as variáveis de ambiente em um arquivo para não

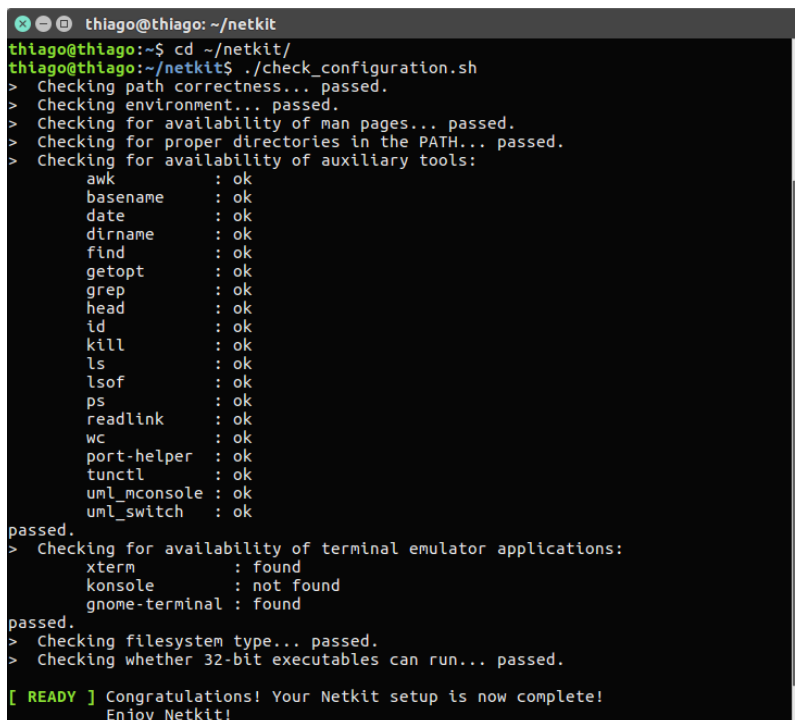
precisar executar esses três comandos todas as vezes que o computador for ligado.

```
$ echo export NETKIT_HOME=~/.netkit >> ~/.bashrc
$ echo export PATH=$PATH:$NETKIT_HOME/bin >> ~/.bashrc
$ echo export MANPATH=$NETKIT_HOME/man >> ~/.bashrc
```

## Verificando Tudo

```
$ cd ~/.netkit
$ ./check_configuration.sh
```

Ao verificar a configuração realizada no seu terminal deverá apresentar as seguintes informações



```
thiago@thiago: ~/.netkit
thiago@thiago:~$ cd ~/.netkit/
thiago@thiago:~/.netkit$ ./check_configuration.sh
> Checking path correctness... passed.
> Checking environment... passed.
> Checking for availability of man pages... passed.
> Checking for proper directories in the PATH... passed.
> Checking for availability of auxiliary tools:
    awk      : ok
    basename : ok
    date     : ok
    dirname  : ok
    find     : ok
    getopt   : ok
    grep     : ok
    head     : ok
    id       : ok
    kill     : ok
    ls       : ok
    lsof     : ok
    ps       : ok
    readlink : ok
    wc       : ok
    port-helper : ok
    tuncctl  : ok
    uml_mconsole : ok
    uml_switch : ok
passed.
> Checking for availability of terminal emulator applications:
    xterm      : found
    konsole    : not found
    gnome-terminal : found
passed.
> Checking filesystem type... passed.
> Checking whether 32-bit executables can run... passed.
[ READY ] Congratulations! Your Netkit setup is now complete!
Enjoy Netkit!
```

Figure 1. Parabéns! Sua instalação do Netkit está completa!

Caso apresente algum erro, poderá ser necessário a instalação de algumas dependência.

## Instalação de Dependências

```
$ sudo apt-get update
$ sudo apt-get install ia32-libs
$ sudo apt-get install libc6-i386
```

## Utilizando o Ambiente

A seguir será exibido como inicializar um único host ou um laboratório com vários dispositivos bem como a abordagem dos comandos necessários.

## Comandos

O Netkit possui dois grupos de comandos. Esses comandos podem ser executados no seu terminal linux sendo comandos com prefixo **v** e os comandos com prefixo **l**.

- v<comando>
- l<comando>

Os comandos com prefixo **v** são utilizados para configuração e inicialização de um único host virtual. Já os comandos com prefixo **l** são utilizados para manipulação de laboratórios com vários hosts.

### V<comandos>

**vstart** Inicia uma nova maquina virtual cada novo host virtual instanciado pelo mesmo usuário deve possuir um nome diferente. Caso parâmetros adicionais não sejam especificados o host virtual será inicializado conforme parâmetros especificados nos arquivos de configuração do netkit. Exemplo de funcionamento:

```
$ vstart pc1 ①
```

① Inicia um host virtual com nome: pc1.

Pode ser observado que ao iniciar um host virtual será criado um arquivo pc1.disk e esse arquivo será o sistema de arquivos do seu host virtual. A medida que configurações sejam feitas, estas serão armazenadas dentro desse disco virtual. Caso o disco virtual seja excluído as configurações serão perdidas. O host virtual poderá ser finalizado e inicializado novamente posteriormente através do mesmo comando.

Alguns parâmetros adicionais:

- --mem=xx - Define quantidade de memória ram a ser consumida pelo host virtual onde xx deve ser substituído pelo valor em mb.
- --ethx=DOMAIN - Definie uma interface de rede para o host virtual onde o x deve ser substituido pelo número da interface. O DOMAIN simboliza o domínio de colisão de pacotes (hub/switch) e deve ser substituído pelo nome do equipamento caso pertinente.
- --exec=COMMAND - Especifica comando a ser executado assim que o host virtual for iniciado.

Para mais parâmetros verifique o manual do netkit em <http://www.netkit.org/>.

Exemplos de utilização:

```
$ vstart pc1 --eth0=A --eth1=B --mem=100 ①
```

① Inicia um host virtual com nome: pc1 e duas interfaces de rede (eth0 e eth1) cada interface está conectada a um domínio de colisão sendo respectivamente A e B. O host em questão irá consumir 100mb de memória ram.

## NOTE

Você poderá iniciar um novo host a partir de qualquer diretório desde que este diretório não contenha espaços em seu nome. Caso isso ocorra um aviso de erro poderá ser exibido em relação ao caminho das variáveis de ambiente.

**vlist** Lista máquinas virtuais em atividade exibindo informações a seu respeito.

**vconfig** Habilita novas interfaces de rede para máquinas virtuais que já estejam em funcionamento.

```
$ vconfig --eth2=A pc1 ①
```

① Adiciona nova interface de rede (eth2) conectada ao domínio de colisão A ao pc1

**vcrash** Finaliza um ou vários hosts virtuais. Os hosts virtuais também podem ser desligados através do comando halt dentro de seu próprio terminal.

```
$ vcrash pc1 ①
```

① finaliza a execução do host pc1

## L<comandos>

A seguir serão apresentando os comandos do grupo l para que possamos trabalhar com laboratórios. Entretanto, os comandos apenas surtirão efeito após a criação da estrutura de diretórios e arquivos para o laboratório.

**lstart** Inicia todos os hosts do laboratório

**lcrash** Finaliza todos os hosts do laboratório

**lclean** Após a finalização do laboratório o comando lclean pode ser utilizado para limpar eventuais arquivos temporários.

**linfo** Apresenta informações a respeito do laboratório sem que este ainda tenha sido inicializado.

## Criando uma Rede Ponto-a-Ponto

Nessa sessão criaremos uma rede ponto-a-ponto através de dois hosts virtuais conectados a um domínio de colisão. Execute os comandos a seguir no terminal:

```
$ vstart pc1 --eth0=A ①  
$ vstart pc2 --eth0=A ②
```

① Inicializa um host virtual (pc1) com uma interface de rede (eth0) conectada ao domínio A

② Inicializa um segundo host virtual (pc2) com uma interface de rede (eth0) conectada ao mesmo domínio A

Para melhor compreensão criamos dois hosts ligados a uma espécie de hub virtual. Nesse ambiente

de testes os dois hosts estão fisicamente conectados. Para que possam trocar informação basta configurarmos os endereços em suas respectivas interfaces de rede. Podemos fazer isso através do comando **ifconfig**. Logo abaixo temos dois comandos que devem ser executados cada um em uma janela de terminal dos hosts virtuais.

```
pc1$ ifconfig eth0 192.168.10.1 ①
pc2$ ifconfig eth0 192.168.10.2 ②
```

- ① Atribui o endereço 192.168.10.1 ao pc1
- ② Atribui o endereço 192.168.10.2 ao pc2

Para testarmos a comunicação podemos enviar um pacote de teste do pc1 para o pc2. Utilizaremos dois comandos um para enviar os pacotes através do pc1 o comando **ping** e outro para monitorar o recebimento dos pacotes no pc2 o comando **tcpdump**.

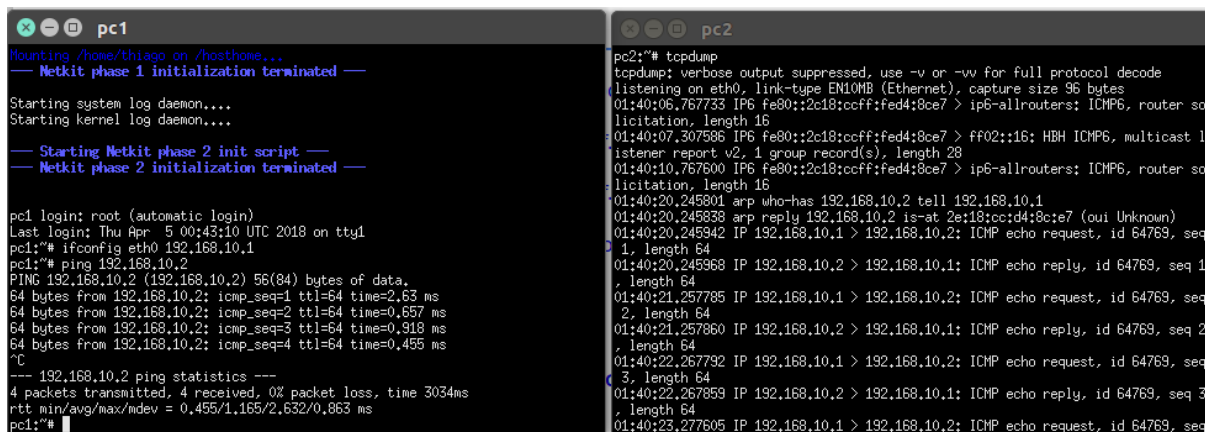
No terminal do pc2 execute

```
$ tcpdump
```

No terminal do pc1 execute

```
$ ping 192.168.10.2
```

A imagem a seguir ilustra o funcionamento do teste onde o pc1 envia pacotes ao pc2 enquanto este monitora o recebimento.



```
pc1
Mounting /home/thiago on /hosthome...
--- Netkit phase 1 initialization terminated ---
Starting system log daemon....
Starting kernel log daemon....

--- Starting Netkit phase 2 init script ---
--- Netkit phase 2 initialization terminated ---

pc1 login: root (automatic login)
Last login: Thu Apr  5 00:43:10 UTC 2018 on tty1
pc1:~# ifconfig eth0 192.168.10.1
pc1:~# ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data:
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=2.63 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=64 time=0.657 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=64 time=0.918 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=64 time=0.455 ms
^C
--- 192.168.10.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3034ms
rtt min/avg/max/mdev = 0.455/1.165/2.632/0.863 ms
pc1:~#
```

```
pc2:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
01:40:06.767733 IP6 fe80::2c18:ccff:fed4:8ce7 > ip6-allrouters: ICMP6, router so
licitation, length 16
01:40:07.307586 IP6 fe80::2c18:ccff:fed4:8ce7 > ff02::16: HBH ICMP6, multicast l
istener report v2, 1 group record(s), length 28
01:40:10.767600 IP6 fe80::2c18:ccff:fed4:8ce7 > ip6-allrouters: ICMP6, router so
licitation, length 16
01:40:20.245801 arp who-has 192.168.10.2 tell 192.168.10.1
01:40:20.245838 arp reply 192.168.10.2 is-at 2e:18:cc:d4:8c:e7 (oui Unknown)
01:40:20.245942 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 64769, seq
1, length 64
01:40:20.245968 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 64769, seq 1
, length 64
01:40:21.257785 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 64769, seq
2, length 64
01:40:21.257860 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 64769, seq 2
, length 64
01:40:22.267792 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 64769, seq
3, length 64
01:40:22.267859 IP 192.168.10.2 > 192.168.10.1: ICMP echo reply, id 64769, seq 3
, length 64
01:40:23.277605 IP 192.168.10.1 > 192.168.10.2: ICMP echo request, id 64769, seq
```

Figure 2. Teste rede ponto-a-ponto

## Criando um Laboratório

Um laboratório no netkit é um set de configurações para hosts virtuais que podem ser iniciados e finalizados simultaneamente. Para tanto deve ser criada uma estrutura de arquivos e diretórios. Sendo:

- Lab.conf - Arquivo de configuração para descrever a estrutura e topologia do laboratório

- **Diretórios** - Um diretório para cada host virtual que poderá armazenar qualquer arquivo a ser utilizado por aquele host específico. Esse diretório funcionará como o diretório raiz (/) para o host. Dessa forma, caso seja criado um arquivo ou outros diretórios ali dentro, estes poderão ser visualizados dentro do terminal do host virtual.
- **.startup e .shutdown** - Cada host virtual poderá ter o seu arquivo .startup ou .shutdown e esses arquivos servem para respectivamente armazenarem instruções que serão processadas assim que o host for iniciado ou desligado.

Para nosso exemplo utilizaremos como cenário 3 hosts e dois hubs. O pc1 e 3 terão apenas uma interface de rede e estarão conectados respectivamente ao hub 01 e 02. Já o pc2 terá duas interfaces de rede e será conectado aos dois hubs. Segue diagrama do nosso cenário.

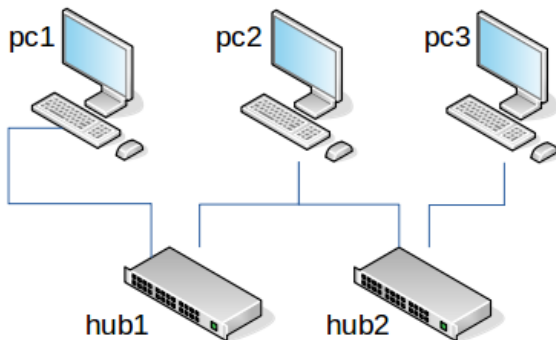


Figure 3. Laboratório 01

## 1 - Criando arquivos e diretórios

Crie um diretório para armazenar o laboratório. Dentro desse diretório serão criados um diretório para cada host, um arquivo .startup para cada host e um arquivo .conf para o laboratório.

**NOTE** O nome do diretório que irá armazenar o laboratório não deve conter espaços.

```

$ cd ~
$ mkdir lab01
$ cd lab01
$ touch lab.conf pc1.startup pc2.startup pc3.startup
$ mkdir pc1 pc2 pc3
  
```

## 2 - Realizando configurações

Apos criados os arquivos e diretórios iremos inserir as configurações.

```

#lab.conf
pc1[0]=hub1
pc2[0]=hub1
pc2[1]=hub2
pc3[0]=hub2

pc2[mem]=64
  
```

A lógica do lab.conf é a seguinte: O pc1 está conectado ao hub1 através da interface 0. O pc2 está conectado ao hub1 via interface 0 e ao hub2 via interface 1. Já o pc3 conecta-se ao hub2 via interface 0. Caso seja necessário alterar o tamanho de memória a ser consumido pelo host pode ser feito conforme a última linha do arquivo. A quantidade de memória padrão (32mb) já é suficiente para o funcionamento do host virtual.

```
#pc1.startup
ifconfig eth0 192.168.10.1
```

```
#pc2.startup
ifconfig eth0 192.168.10.2
ifconfig eth1 10.0.0.1
```

```
#pc3.startup
ifconfig eth0 10.0.0.2
```

Os arquivos .startup inserem configurações assim que o host for iniciado. Pode ser utilizado qualquer instrução linux para criarmos diversas configurações. Para os três arquivos apresentados foi utilizado o comando ifconfig para configurar as interfaces com os endereços escolhidos.

### 3 - Subindo o Laboratório

Dentro do diretório onde se encontram os arquivos do laboratório execute o seguinte comando:

```
$ lstart
```

Seu laboratório será iniciado e três janelas do xterminal serão abertas. Uma para cada host virtual. Realize alguns testes de comunicação entre os hosts. Em teoria o pc1 poderá comunicar-se com o pc2 e este poderá enviar pacotes ao pc1 e ao pc3.

### Troca de arquivos entre host real e host virtual

Dentro de cada host virtual existe um diretório chamado /hosthome que pode ser acessado via interface xterminal. Esse diretório aponta diretamente para o diretório /home do host real. Dessa forma é possível acessar, através do host virtual, os arquivos que estão armazenados no diretório /home do host principal.

#### WARNING

O diretório /hosthome é montado com permissão de leitura e escrita, tenha cuidado com as atividades a serem realizadas dentro deste diretório.

### Acesso a Internet através do Host virtual

É possível a criação de um túnel (tap interface) entre o host real e o host virtual de forma que o host virtual acesse redes externas através do host real.



## Acesso a rede externa com vstart

```
vstart pc1 --eth0=tap,10.0.0.1,10.0.0.2 --eth1=A
```

O comando a cima inicia uma nova máquina virtual de nome pc1. Essa máquina irá ter duas interfaces de rede eth0 e eth1. A interface eth0 será configurada automaticamente com o endereço 10.0.0.2 quando a máquina for iniciada. O endereço ip 10.0.0.1 será utilizado para rotear o tráfego entre o host real e o host virtual. Para utilização do tap ambos os endereços devem estar na mesma faixa. Ao executar este comando a senha de root deve ser solicitada.

## Acesso a rede externa em um laboratório

Para configurar um acesso a rede externa dentro de um laboratório. Basta fazer acesso ao arquivo lab.conf e inserir a instrução correspondente ao tap ao pc desejado. Como mostra a instrução a baixo.

```
#lab.conf  
pc1[0]=tap,192.168.10.1,192.168.10.2
```

# Criação do ambiente e Informações Iniciais

O Apache é um serviço que oferece suporte a hospedagem de um ou vários sites no servidor, interpretando páginas estáticas e scripts. Para que seja possível a hospedagem de páginas dinâmicas com integração com PHP e MySQL por exemplo, faz-se necessário a implementação do LAMP. O apache pode ser dividido em dois grupos de versões mais utilizadas sendo a versão 1.3 e a versão 2 qual será utilizada neste guia. O motivo da versão 1.3 ainda ser utilizada é sua compatibilidade com módulos antigos não disponíveis para versão 2.

## Criação do Ambiente

Leve em consideração que nosso ambiente tem a seguinte topologia implementada através do netkit:

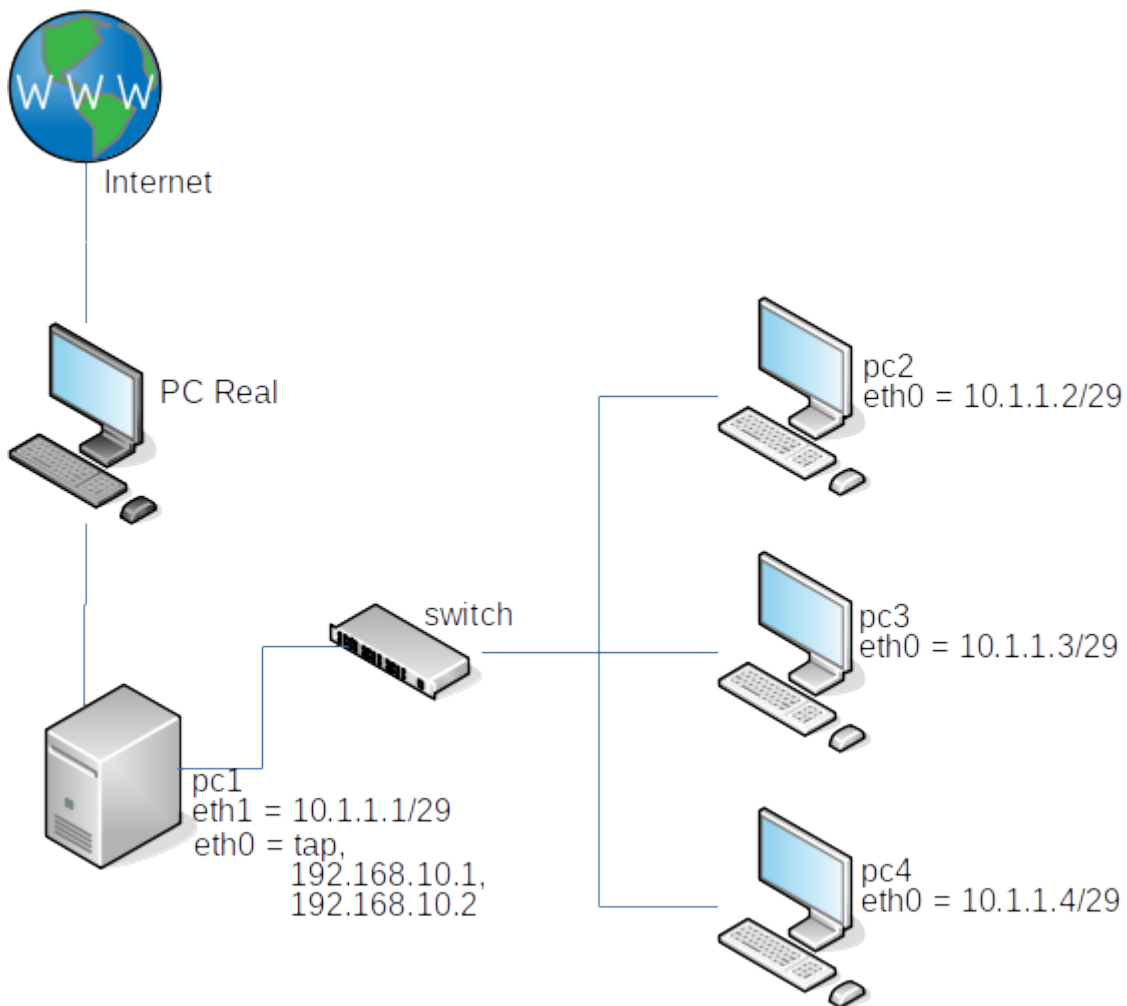


Figure 4. Laboratório 01