



**INSTITUTO FEDERAL**

Norte de Minas Gerais

Campus Januária

# Admin. Serviços de Redes

*- Kathará -*



# Porque Linux...

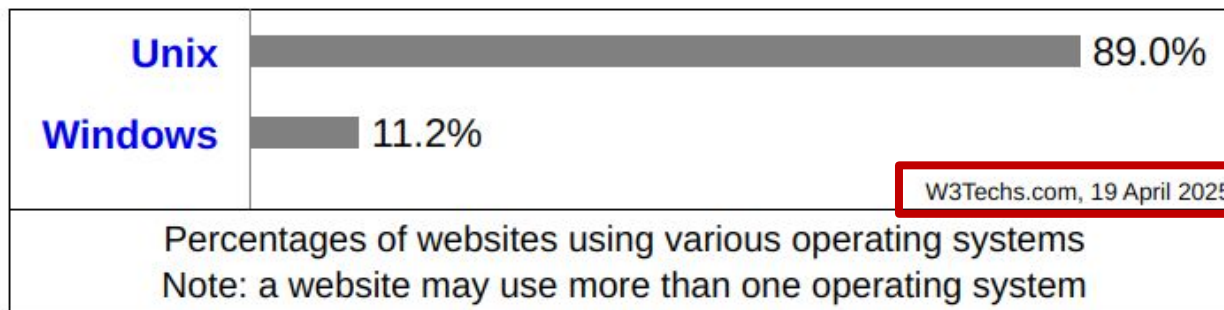
Technologies > Operating Systems

## Usage statistics and market shares of operating systems for websites

This diagram shows the percentages of websites using various operating systems. See [technologies overview](#) for explanations on the methodologies used in the surveys. Our reports are updated daily.

How to read the diagram:

Unix is used by 89% of all the websites whose operating system we know.





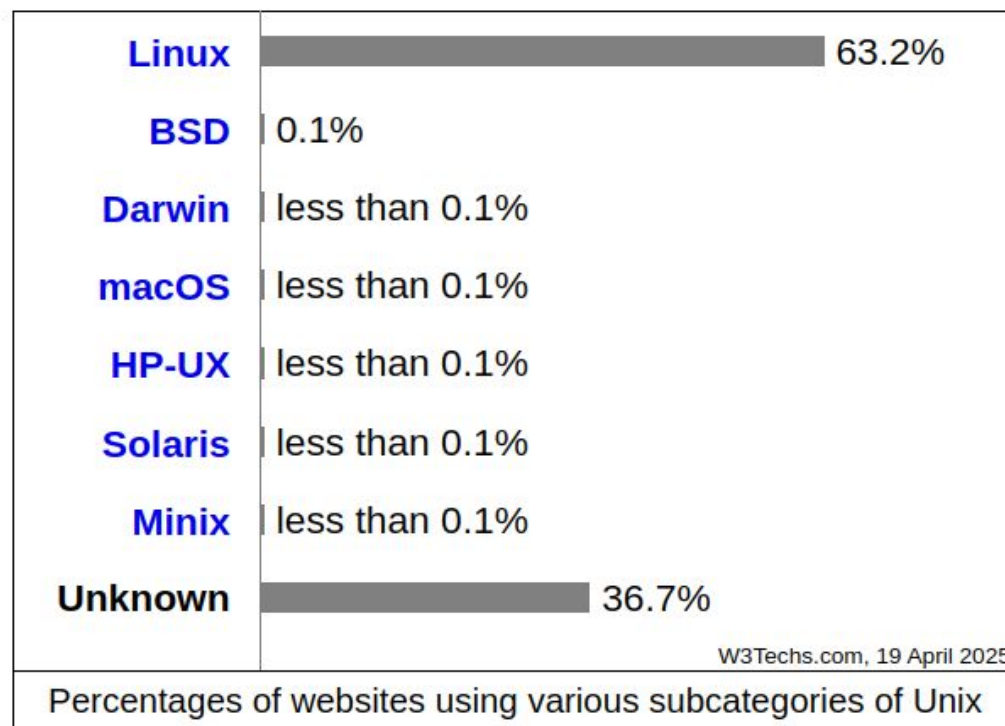
# Porque Linux...

## Subcategories of Unix

This diagram shows the percentages of websites using various subcategories of Unix.

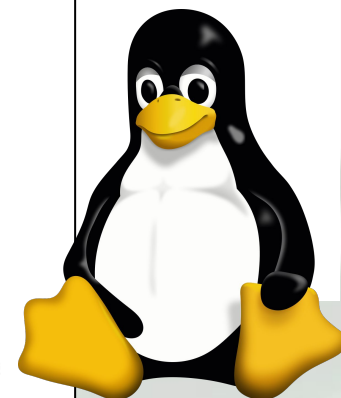
How to read the diagram:

Linux is used by 63.2% of all the websites who use Unix



**websites**

technologies  
used daily.

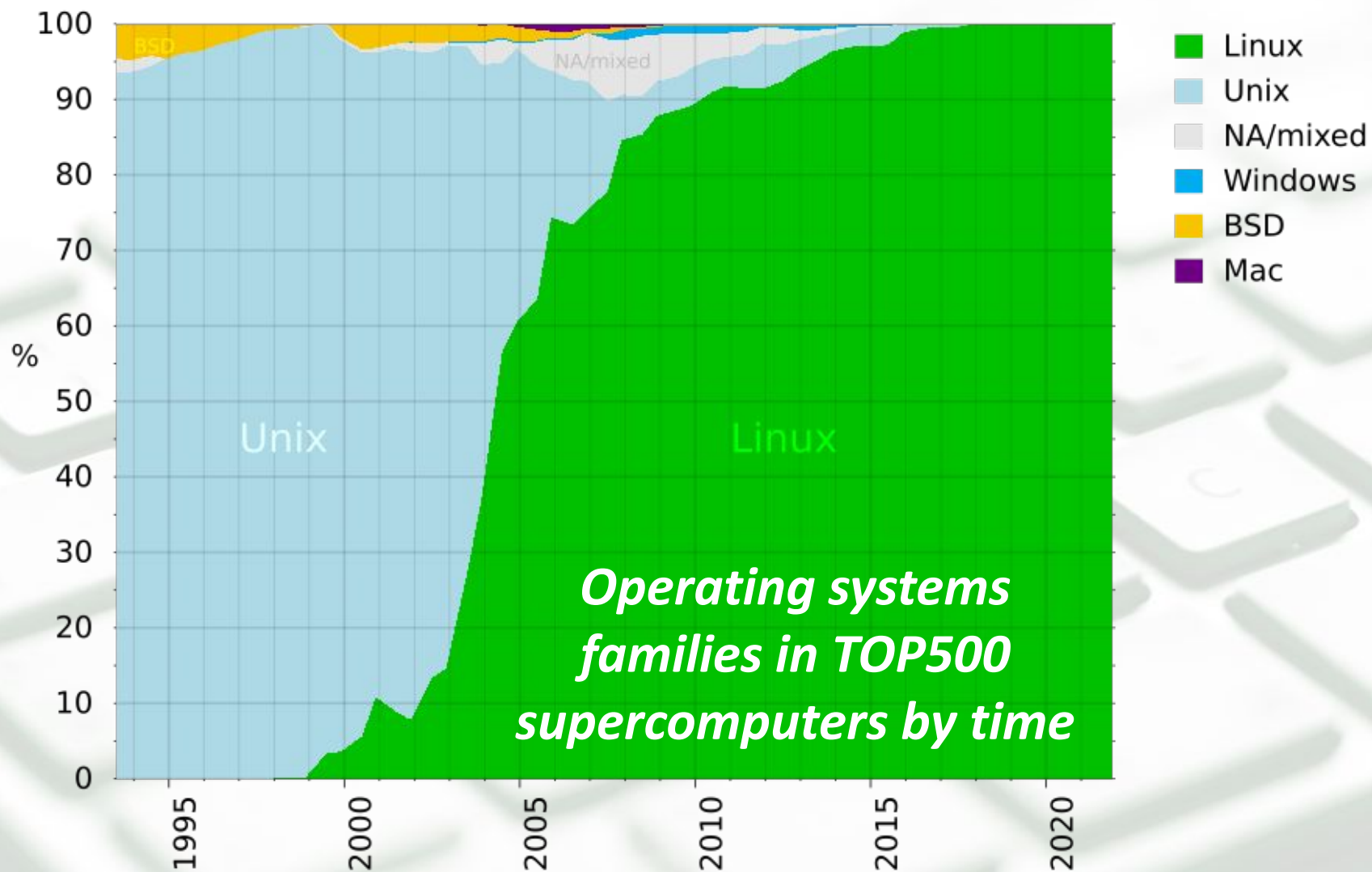






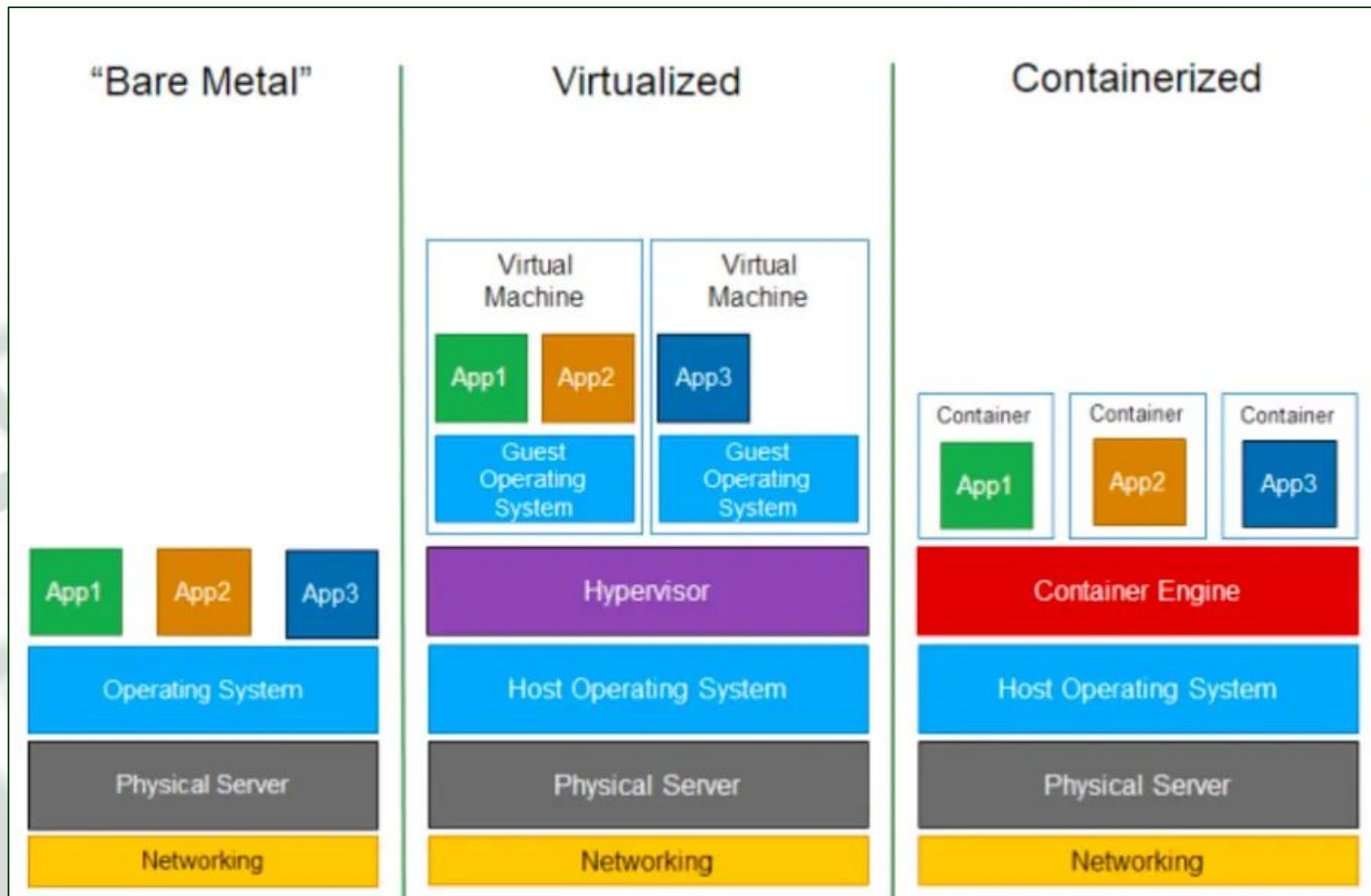
**INSTITUTO FEDERAL**  
Norte de Minas Gerais  
Campus Januária

# Porque Linux...



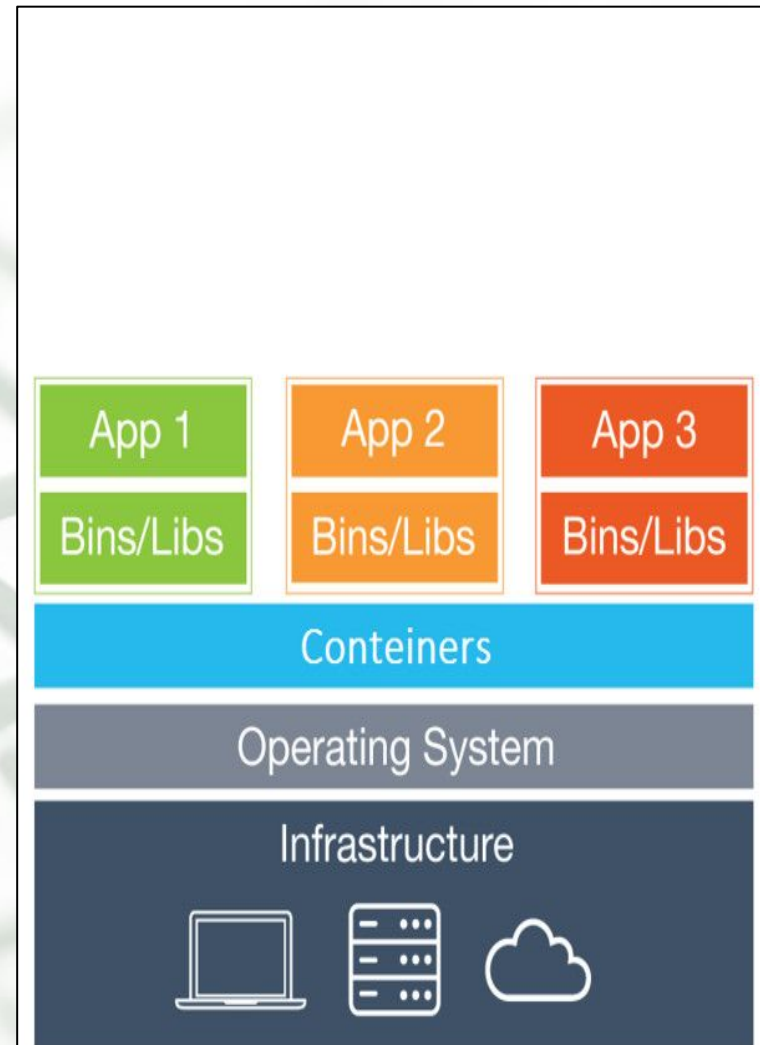
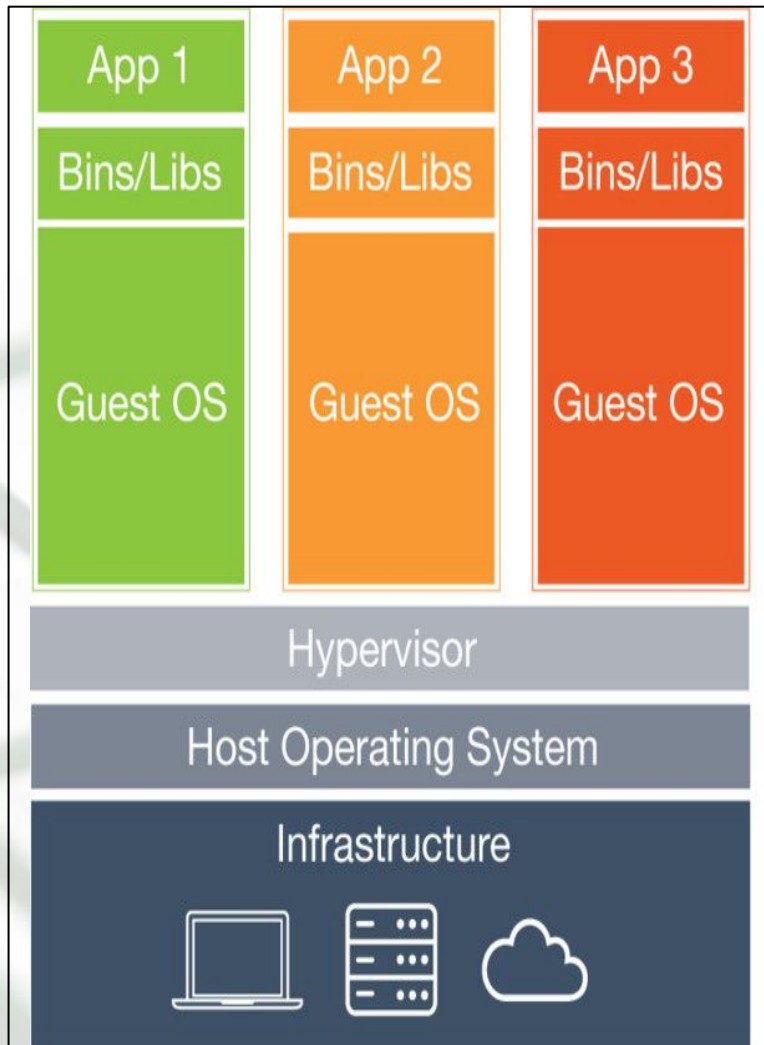


# InfraEstrutura de Servidores





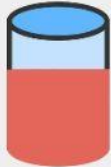
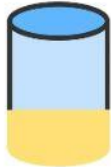


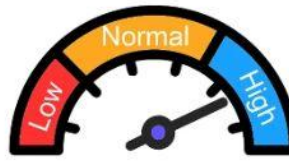


# InfraEstrutura de Servidores

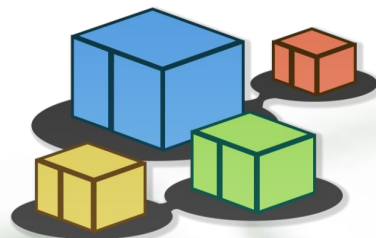




# InfraEstrutura de Servidores

	Virtualization	Containerization
Startup time	 minutes	 seconds
Disk space		
Portability	Less Portable	
Efficiency		
Operating system/kernel	Dedicated	Shared





# Kathará

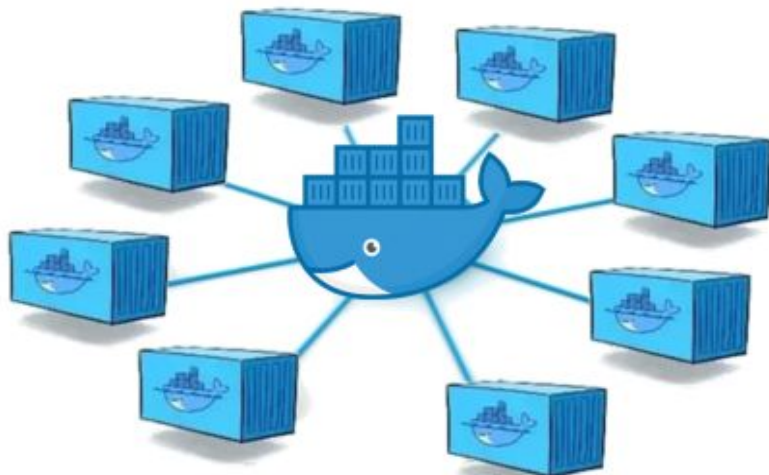
- **Kathará** é um ambiente *open-source* para **emulação** de redes de computadores baseado na tecnologia de virtualização por *containers* (**DOCKER**).
  - *Kathará é uma evolução do projeto Netkit.*





**INSTITUTO FEDERAL**  
Norte de Minas Gerais  
Campus Januária

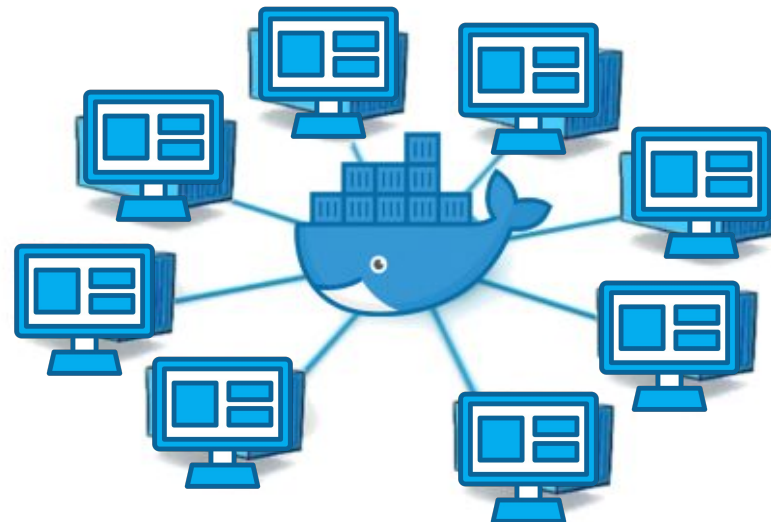
# Kathará



Containers

Operating System

Infrastructure

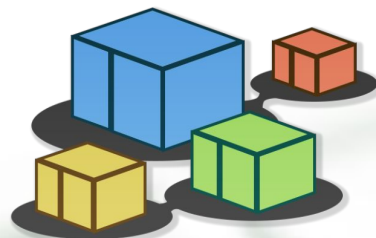


Containers

Operating System

Infrastructure





# Kathará

- A ferramenta permite a **criação, configuração e gerenciamento de redes e serviços**, desde as mais simples às mais complexas.

- **Site**

- <https://www.kathara.org/>





# Instalação & Configuração

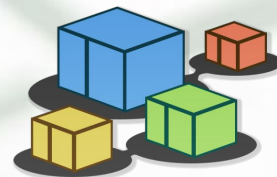
## ■ Orientações para Instalação e Configuração:

### 1º Passo - Install Docker:

```
$ sudo apt-get update  
$ sudo apt-get install docker.io  
$ sudo usermod -aG docker $(whoami)  
$ sudo docker run hello-world
```



docker



Kathará



# Instalação & Configuração

■ Or

1º Pa

\$ sudo

\$ sudo

\$ sudo

\$ sudo

```
adriano@adriano: ~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
[sudo] senha para adriano:  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
adriano@adriano:~$
```



rá





# Instalação & Configuração

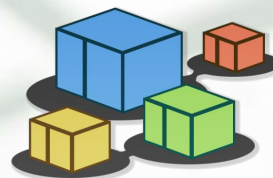
## ■ Orientações para Instalação e Configuração:

### 2º Passo - Install Kathará:

```
$ sudo add-apt-repository ppa:katharaframework/kathara  
$ sudo apt-get update  
$ sudo apt-get install kathara  
$ kathara check
```



docker





Kathará



# Instalação & Configuração

## ■ Orientações para Instalação e Configuração:

```
adriano@adriano-pc:~  
Arquivo Editar Ver Pesquisar Terminal Ajuda  
$> kathara check  
  
System Check  
  
Current Manager is: Docker (Kathara)  
Manager version is: 26.1.3  
Python version is: 3.11.11 (main, Dec 4 2024, 08:55:07) [GCC  
11.4.0]  
Kathara version is: 3.7.9  
Operating System version is: Linux-5.15.0-136-generic-x86_64  
[Deploying devices] _____ 1/1  
[Deleting devices] _____ 1/1  
✓ Container run successfully.
```





# Instalação & Configuração

## ■ Orientações para Instalação e Configuração:

### 3º Passo - Customização e Outras Definições:

```
$ sudo docker pull adrianoantunesp/kathara
$ kathara settings
$ [Choose default image]: adrianoantunesp/kathara
$ [Choose terminal emulator]: /usr/bin/gnome-terminal
$ [Automatically mount /hosthome on startup]: yes
$ [Automatically mount /shared on startup]: yes
```



# Criando nosso *Hello World!*

- Criando *Host* contendo uma Interface de Rede

```
$ kathara vstart -n pc0 --eth 0:A
```





# Criando nosso *Hello World!*

## ■ Criar

```
root@pc0: /
Arquivo  Editar  Ver    Pesquisar  Terminal  Ajuda
root@pc0:/#
```

\$ katha



# Criando nosso *Hello World*!

- Criando *Host* contendo uma Interface de Rede

```
$ kathara vstart -n pc0 --eth 0:A
```

**Cria e executa um novo contêiner que representa um host...**



# Criando nosso *Hello World*!

- Criando *Host* contendo uma Interface de Rede

```
$ kathara vstart -n pc0 --eth 0:A
```

...nomeado como  
“pc0”



# Criando nosso *Hello World*!

- Criando *Host* contendo uma Interface de Rede

```
$ kathara vstart -n pc0 --eth 0:A
```

...e que possui 1 interface de rede padrão Ethernet “eth0” conectada ao domínio de colisão (enlace) “A”

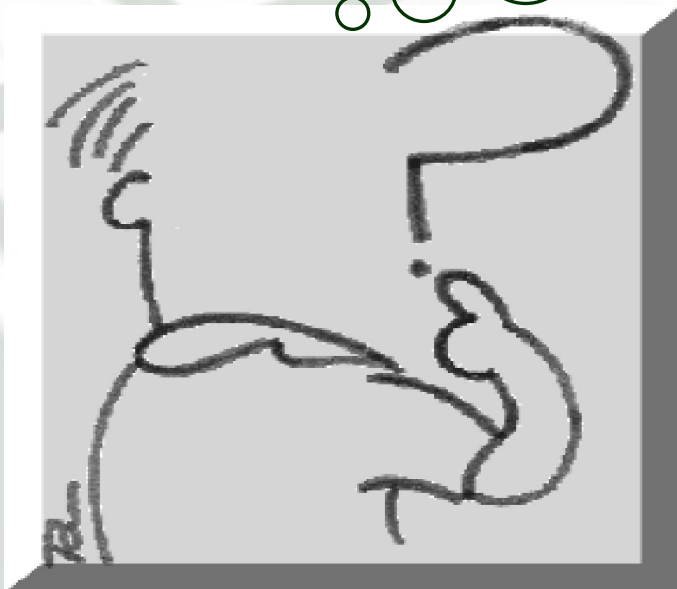




# Explicando...

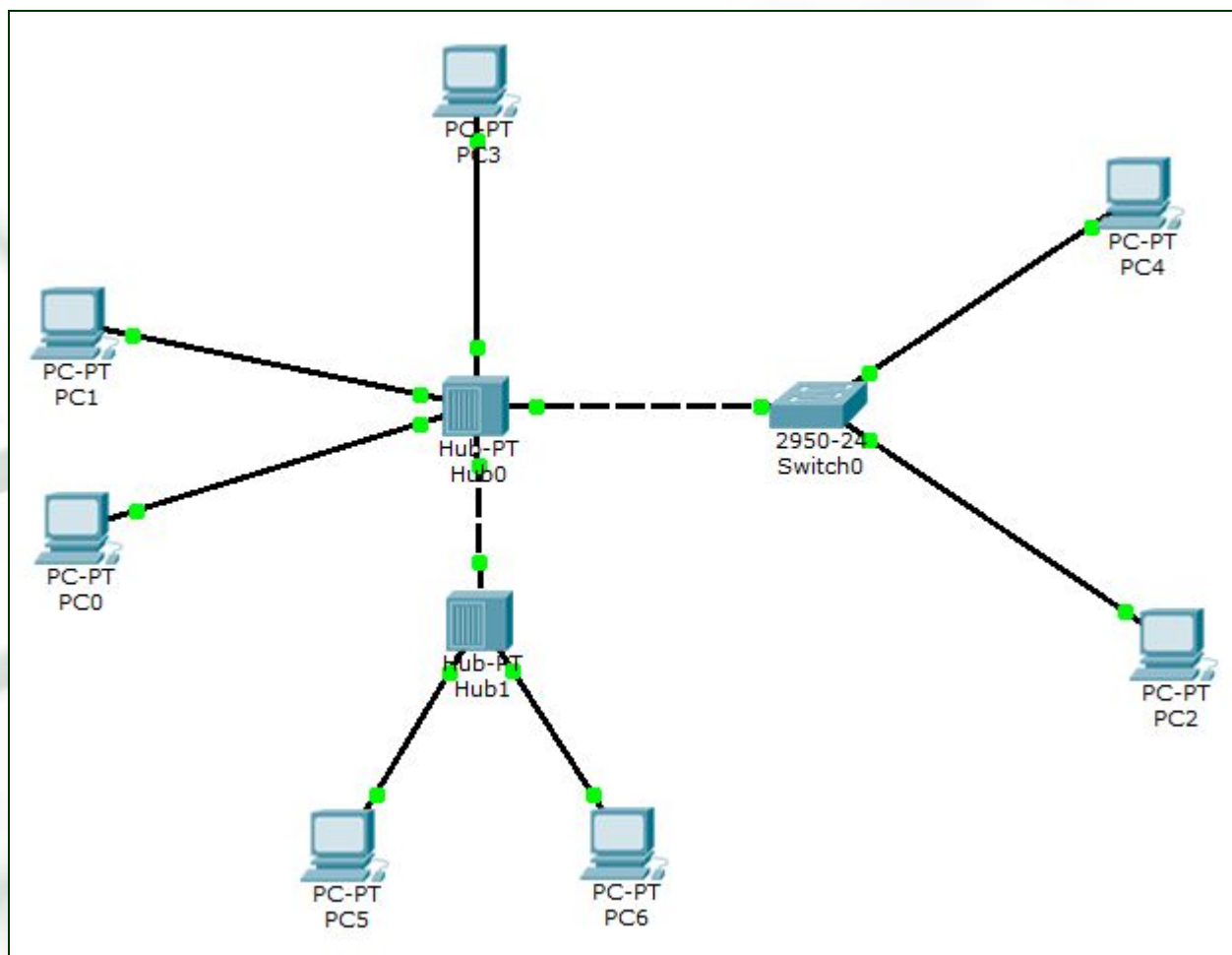
**Domínio de Colisão???**

Já ouvi falar disso...



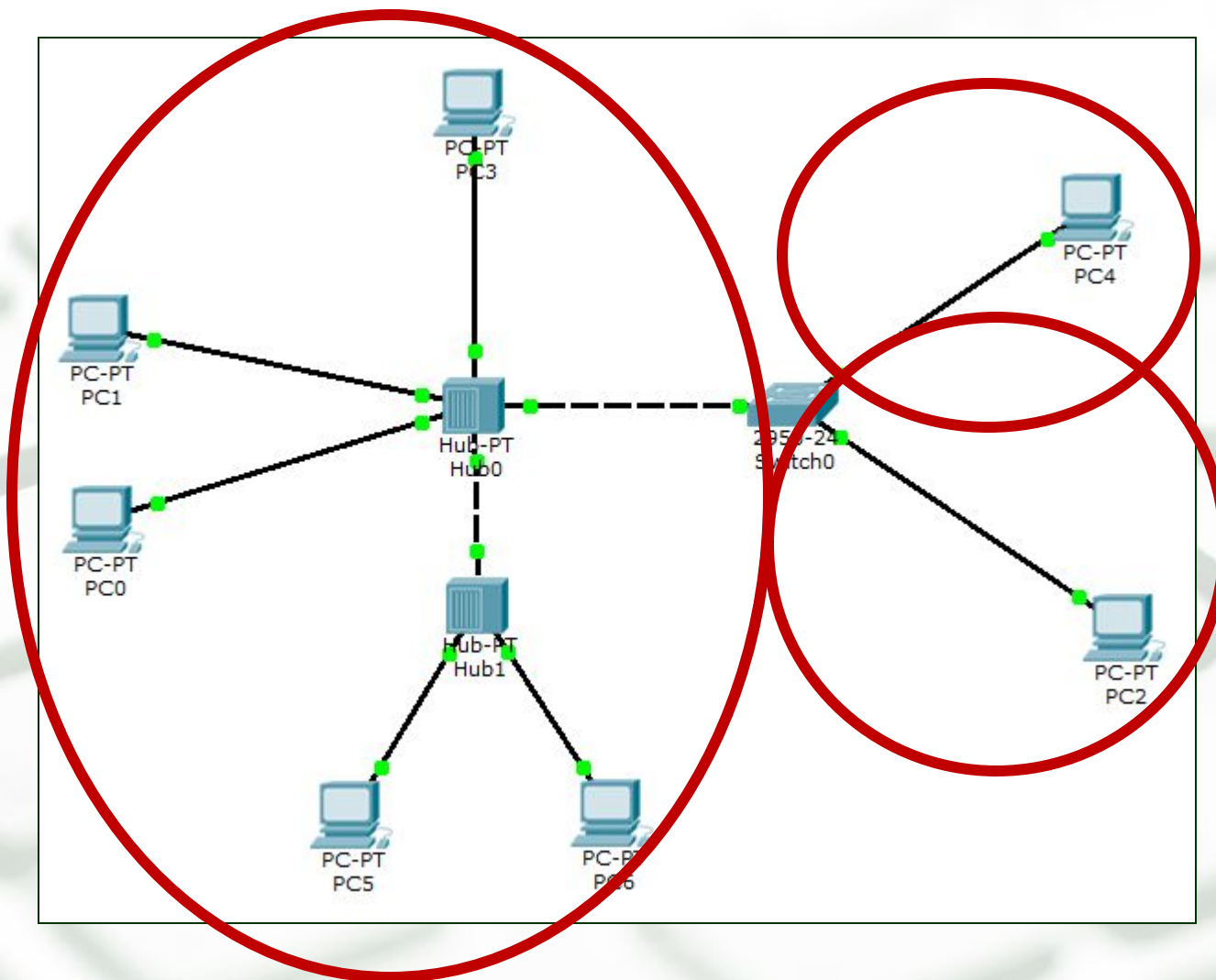


# Quantos Domínios de Colisão?



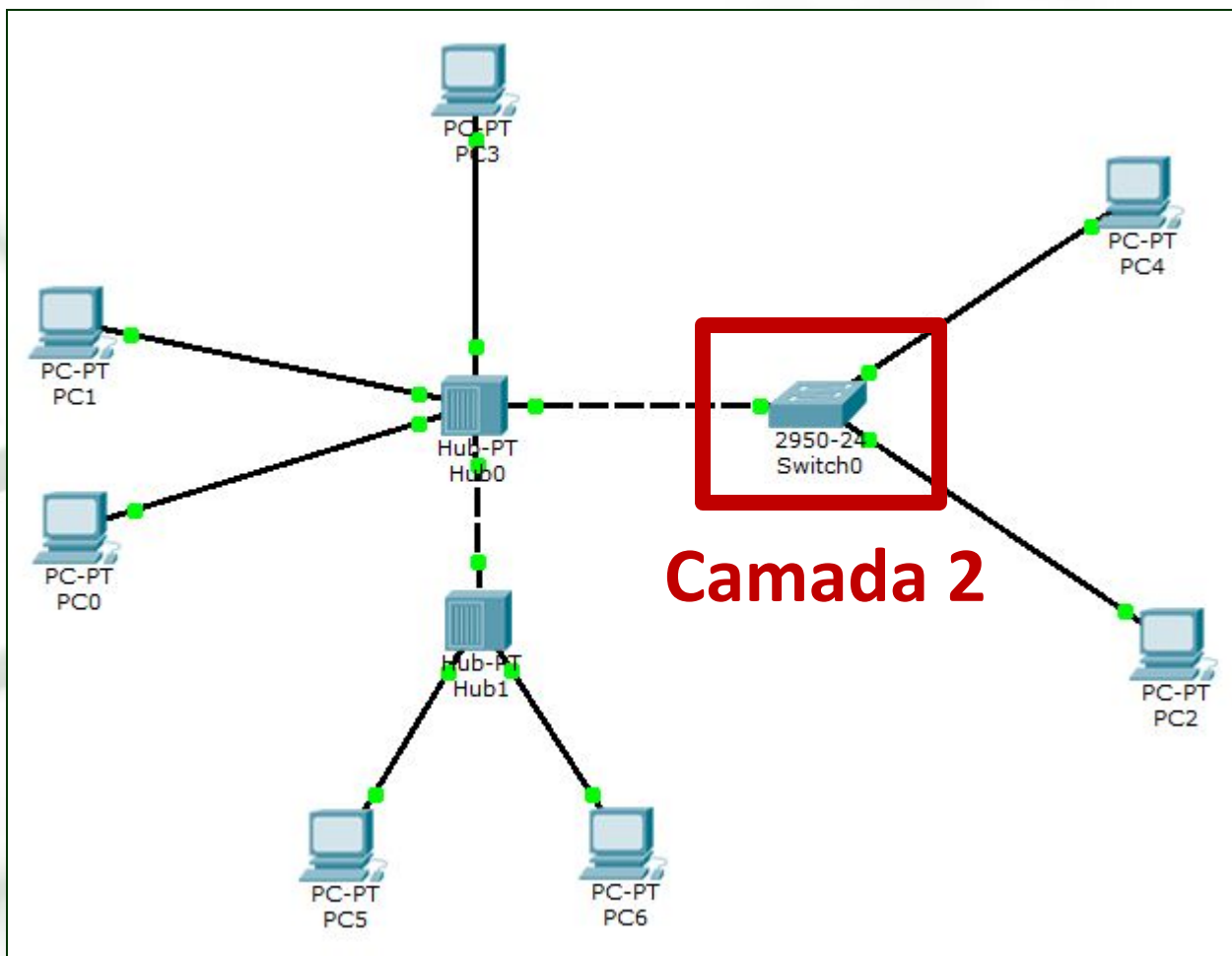


# Quantos Domínios de Colisão?





# Quantos Domínios de Colisão?







# Começando a Prática...

- Crie dois hosts utilizando o **Kathará**...
- Ambos contém uma interface de rede *Ethernet* e estão no mesmo domínio de colisão.
- Os hosts já conseguem se comunicar?

***SIM OU NÃO?***



# Interfaces de Rede

- **ifconfig** é a tradicional ferramenta para visualização e configuração das interfaces de rede em plataformas Linux.

```
$ ifconfig
```

- Outro utilitário (**mais recente**) que também permite a visualização e configuração de interfaces é o “**ip**”.

```
$ ip a
```



# Atividade

- Execute o comando **ifconfig** na **máquina hospedeira** (ou seja, o host que executa o docker/kathará).
- Identifique...
  - **Quantas** interfaces de rede existem.
  - A **diferença** entre as interfaces existentes.
  - O endereço **IP** de cada interface.
  - O endereço **MAC** de cada interface.
  - O endereço de **broadcast** da rede.
  - A **máscara** da rede.
  - A quantidade de pacotes **transmitidos**.
  - A quantidade de pacotes **recebidos**.




# Atividade

- Agora, execute o **ifconfig** nos hosts do kathará...



# Atividade

- Agora, ex



```
root@pc0: /
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
root@pc0:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 0a:71:36:d1:72:dc  txqueuelen 1000  (Ethernet)
        RX packets 58  bytes 8606 (8.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    loop txqueuelen 1000  (Local Loopback)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@pc0:/#
```

**O que falta?**





# Configurando uma Interface

```
$ ifconfig eth0 x.y.z.w/z
```

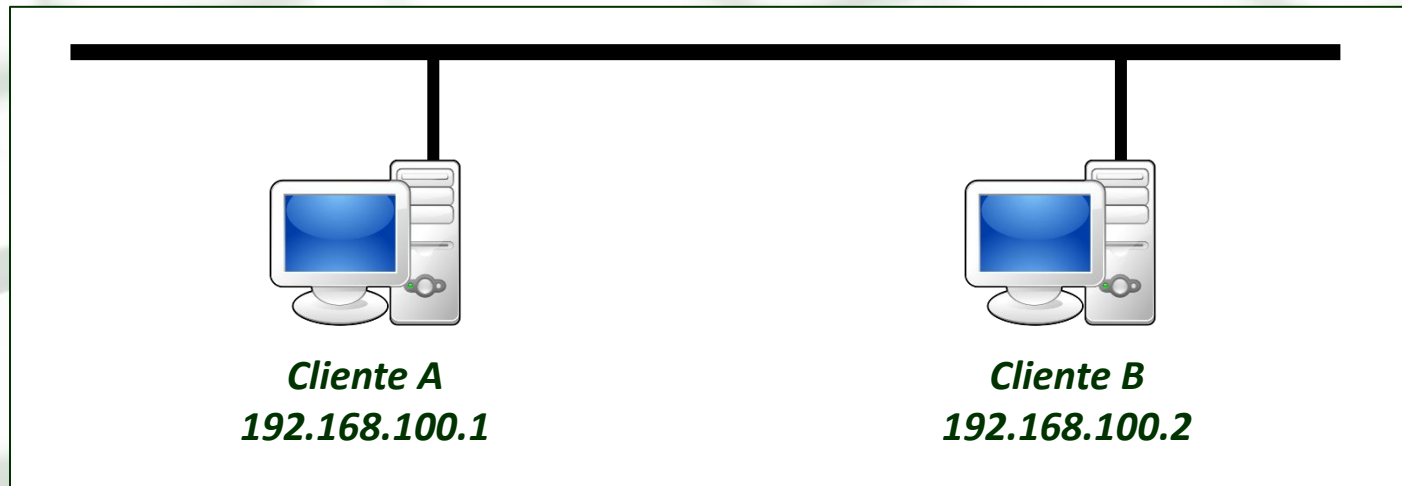
- O comando acima realiza a configuração **temporária** da interface.
  - Atribui o endereço **x.y.z.w** à interface eth0.
  - Atribui máscara de rede correspondente à **/z**.

***Obs: Essa configuração é VOLÁTIL.***



# Laboratório 01-1

- Crie o seguinte laboratório no Kathará:



- Como testar a **conectividade** entre as máquinas?



# PING

- O utilitário **PING** é famoso por testar a conectividade entre dois terminais em rede.
- PING **não** é um protocolo!
- PING é uma **aplicação** baseada em um protocolo de camada 3, chamado **ICMP** (*Internet Control Message Protocol*).

```
$ ping <endereço IP alvo>
```

```
$ ping 192.168.100.2
```

## Observação

No Linux, o comando PING é infinito.

Para encerrar o processo, tecele CTRL + C



# PING

- O
- CO
- PING
- PING
- ca
- Pro

```

root@pc0: /
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda

root@pc0:/# ping 192.168.100.2
PING 192.168.100.2 (192.168.100.2) 56(84) bytes of data.
64 bytes from 192.168.100.2: icmp_seq=1 ttl=64 time=0.105 ms
64 bytes from 192.168.100.2: icmp_seq=2 ttl=64 time=0.113 ms
64 bytes from 192.168.100.2: icmp_seq=3 ttl=64 time=0.113 ms
64 bytes from 192.168.100.2: icmp_seq=4 ttl=64 time=0.114 ms
64 bytes from 192.168.100.2: icmp_seq=5 ttl=64 time=0.095 ms
64 bytes from 192.168.100.2: icmp_seq=6 ttl=64 time=0.115 ms
64 bytes from 192.168.100.2: icmp_seq=7 ttl=64 time=0.113 ms
64 bytes from 192.168.100.2: icmp_seq=8 ttl=64 time=0.080 ms
^C64 bytes from 192.168.100.2: icmp_seq=9 ttl=64 time=0.112 ms
64 bytes from 192.168.100.2: icmp_seq=10 ttl=64 time=0.119 ms
^C
--- 192.168.100.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 231ms
rtt min/avg/max/mdev = 0.080/0.107/0.119/0.017 ms
root@pc0:/#

```

\$ ping

\$ ping

PING

cesso,





# Finalizando os Hosts

- Ao fechar o terminal do Host, ele ainda estará em execução (segundo plano) e pode ser conectado novamente pelo comando abaixo (exemplo).

```
$ kathara connect -v pc0
```

- A lista de hosts em execução é obtida pelo comando

```
$ kathara list
```





# Finalizando os Hosts

- Para finalizar a execução do laboratório virtual (finalizar todos os contêineres), execute:

```
$ kathara wipe
```

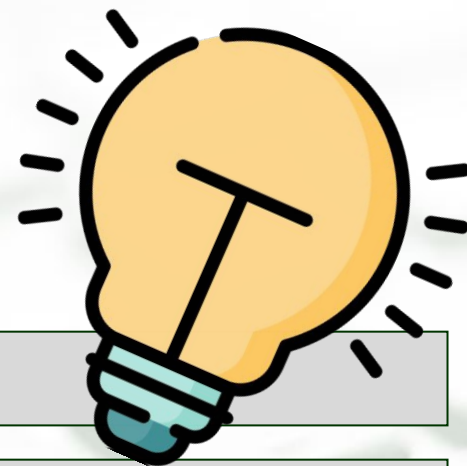
- Para finalizar a execução apenas de um host específico, execute:

```
$ kathara vclean -n pc1
```



# Definindo Comandos Rápidos

Utilize o recurso de “alias” do Linux para criar comandos rápidos...



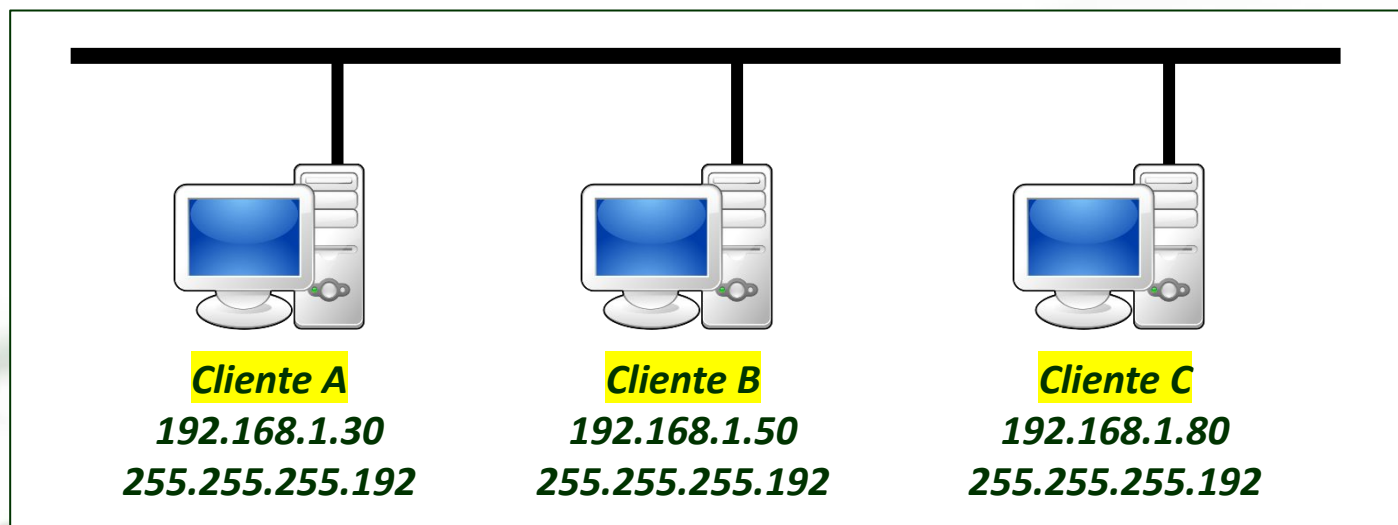
```
$ sudo nano .bashrc
```

```
(...)
```

```
alias ks="kathara vstart -n"  
alias kw="kathara wipe -f"  
alias kc="kathara connect -v"  
alias kx="kathara vclean -n"  
alias kl="kathara lstart"
```



# Laboratório 01-2



## ■ Teste a conectividade...

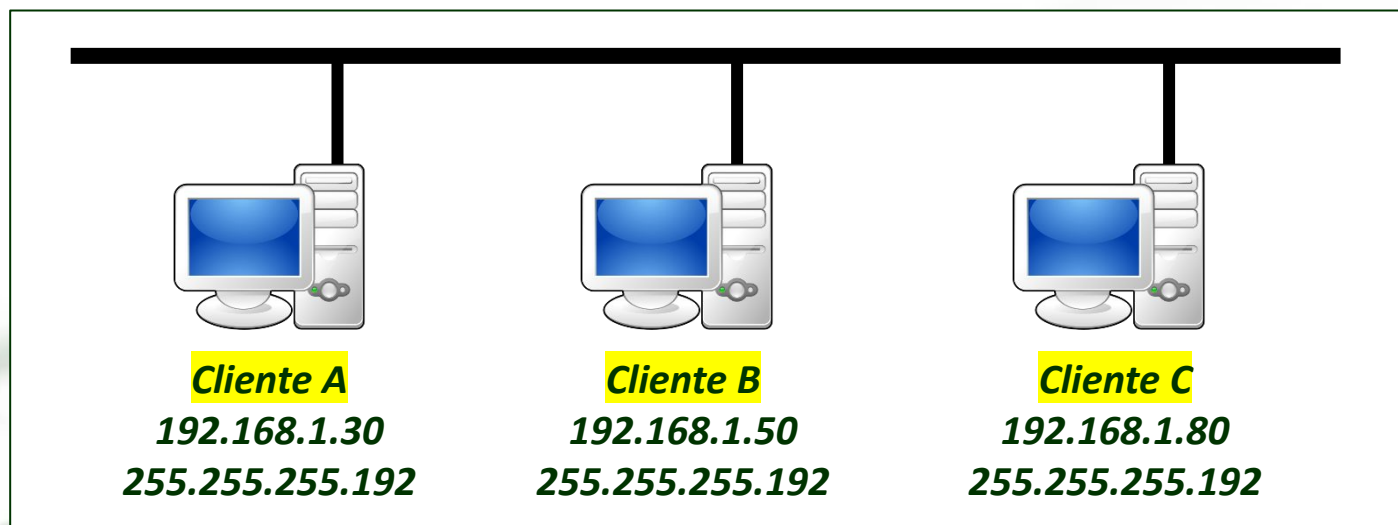
- PC1 -> PC2
- PC2 -> PC1
- PC1 -> PC3
- PC3 -> PC2

### Lembre-se

No Linux, o comando PING é infinito.  
Para encerrar o processo, tecle CTRL + C



# Laboratório 01-2



## ■ Teste a conectividade...

- PC1 -> PC2
- PC2 -> PC1
- ~~PC1 -> PC3~~
- ~~PC3 -> PC2~~

**Motivo???**





# Monitoramento de Rede

- Às vezes é tarefa bastante complexa identificar o ponto focal de um problema de comunicação em rede...
  - O problema é na origem ou no destino?
  - Os pacotes estão saindo pela interface?
  - Os pacotes estão chegando no destino?
  - Os pacotes estão se perdendo no meio do caminho?
  - etc...
- Para auxiliar nessa tarefa, utilizamos **ferramentas de monitoramento de rede**, que permitem a captura e análise de pacotes.



# TCPDUMP

**TCPDUMP**  
é uma  
ferramenta  
nativa de  
monitoramento  
nos sistemas  
Linux.

```
root@pc2: /
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
root@pc2:/# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:41:07.113067 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 28
16:41:07.113114 ARP, Reply 192.168.1.2 is-at 52:8b:35:89:16:b7 (oui Unknown), length 28
16:41:07.113158 IP 192.168.1.1 > 192.168.1.2: ICMP echo request, id 37, seq 1, length 64
16:41:07.113190 IP 192.168.1.2 > 192.168.1.1: ICMP echo reply, id 37, seq 1, length 64
16:41:08.131090 IP 192.168.1.1 > 192.168.1.2: ICMP echo request, id 37, seq 2, length 64
16:41:08.131139 IP 192.168.1.2 > 192.168.1.1: ICMP echo reply, id 37, seq 2, length 64
16:41:09.155052 IP 192.168.1.1 > 192.168.1.2: ICMP echo request, id 37, seq 3, length 64
16:41:09.155087 IP 192.168.1.2 > 192.168.1.1: ICMP echo reply, id 37, seq 3, length 64
16:41:12.258914 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 28
16:41:12.258946 ARP, Reply 192.168.1.1 is-at ea:18:99:65:ed:17 (oui Unknown), length 28
^C
10 packets captured
10 packets received by filter
```



# TCPDUMP

## ■ Principais parâmetros de filtragem do TCPDUMP

```
$ tcpdump -i eth0
```

```
$ tcpdump icmp
```

```
$ tcpdump icmp or tcp
```

```
$ tcpdump port 80
```

```
$ tcpdump src port 1025
```

```
$ tcpdump portrange 21-23
```

```
$ tcpdump host ifnmg.edu.br and port https
```

```
$ tcpdump -nX
```

```
$ tcpdump -w arquivoCaptura.pcap
```

[Veja Mais...](#)



# Laboratório 01-3

- Construa um novo cenário...



- Todos os hosts devem se comunicar entre si.





# Laboratório 01-3

- Um detalhe ainda não encaixa...

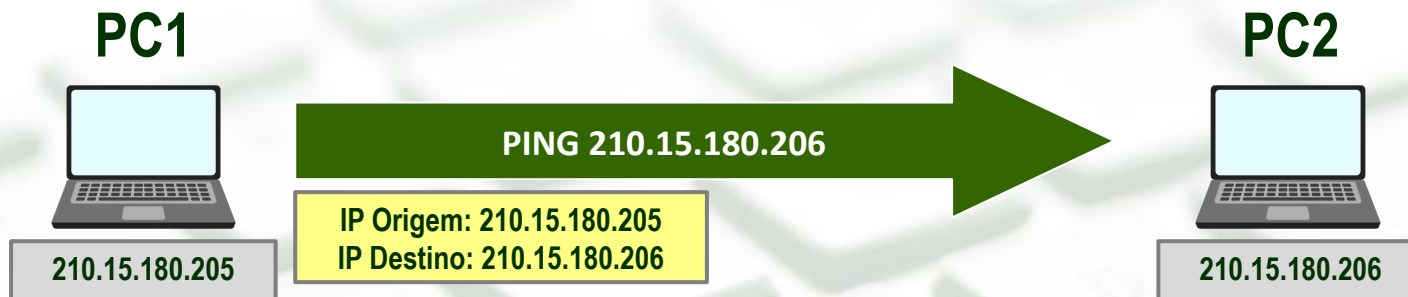






# Laboratório 01-3

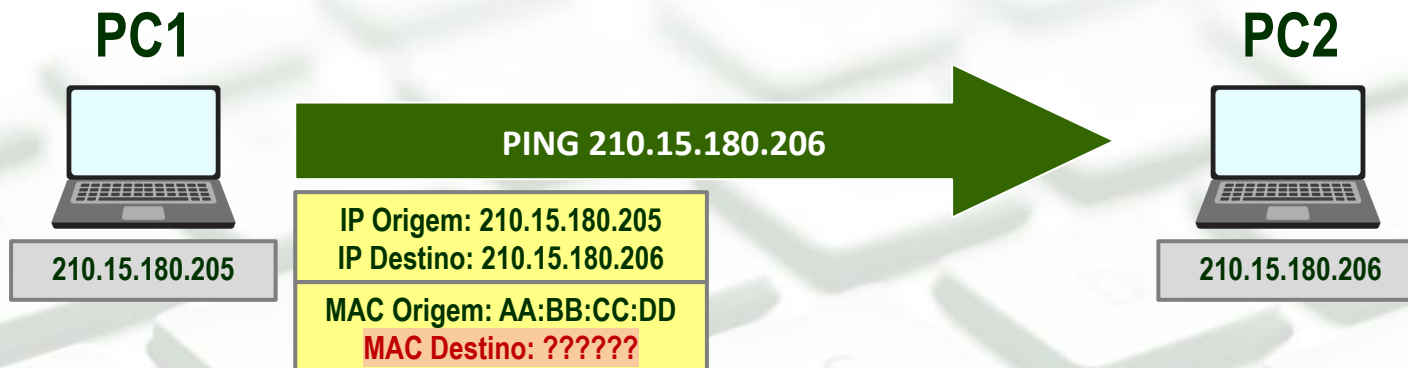
- Um detalhe ainda não encaixa...





# Laboratório 01-3

- Um detalhe ainda não encaixa...



**O endereço IP do alvo é informado na execução do comando ping, OK!**

**Mas como o PC1 descobre o endereço físico (MAC Address) do PC2?**



# Protocolo ARP

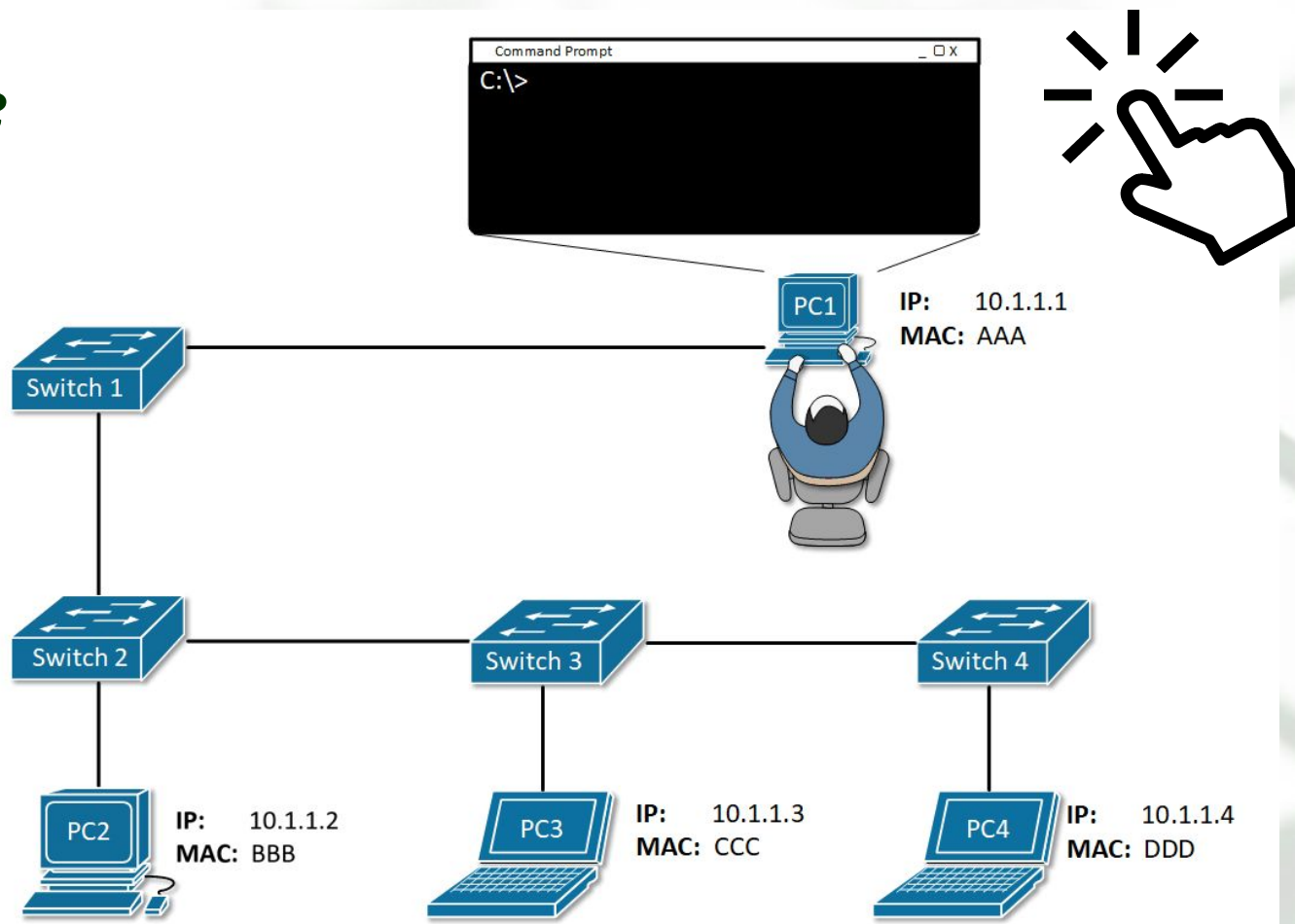
- **Protocolo ARP - *Address Resolution Protocol***
- Protocolo de Resolução de Endereços
  - IP -> MAC
- R-ARP (*Reverse ARP*)
  - MAC -> IP

ARP  
Address  
Resolution  
Protocol



# Protocolo ARP

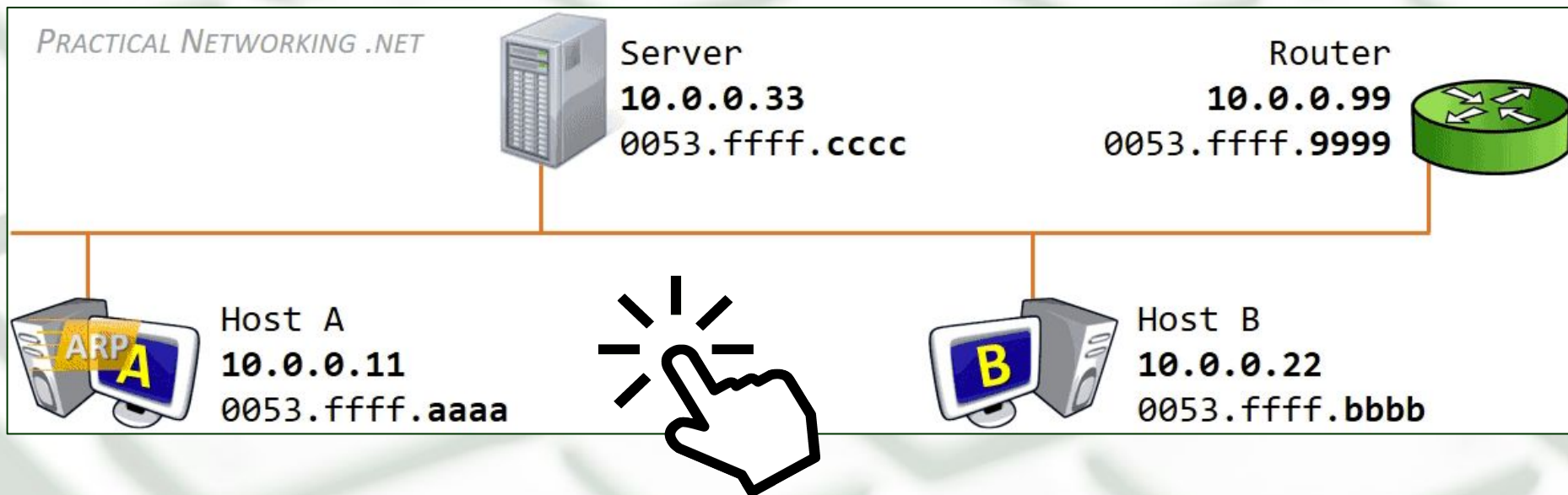
- *ARP Request*
- *ARP Response*





# Protocolo ARP

- **ARP Request**
- **ARP Response**







# Cache ARP

- Quando um PC resolve um endereço MAC através do protocolo ARP, essa informação é mantida em um cache para agilizar consultas futuras.

- Vamos verificar...

PC1 => PING => PC2

*Obs. Execute CTRL + C para interromper o PING.*

- No **PC1** e no **PC2** digite o comando para ver o cache ARP.

```
$ arp -v
```



# Exercício Prático

- Elimine os registros do cache ARP em alguns PC's

```
$ arp -d <endereço_IP>
```

Agora, utilize o TCPDUMP para rastrear e identificar os pacotes **ARP REQUEST** e **ARP RESPONSE**



# Arquivo de Configuração

- É possível criar um arquivo de texto para **configurar automaticamente** todo um Laboratório no Kathará.
- O arquivo deve ser nomeado como **lab.conf**
- É interessante também criar um diretório próprio para cada laboratório (p.ex: **~/kathara/lab01-4/lab.conf** )

```
pc1[0]="A"  
pc2[0]="A"  
pc3[0]="A"  
pc4[0]="A"
```

```
$ kathara lstart
```



# Arquivo de Configuração

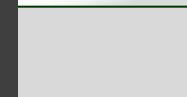
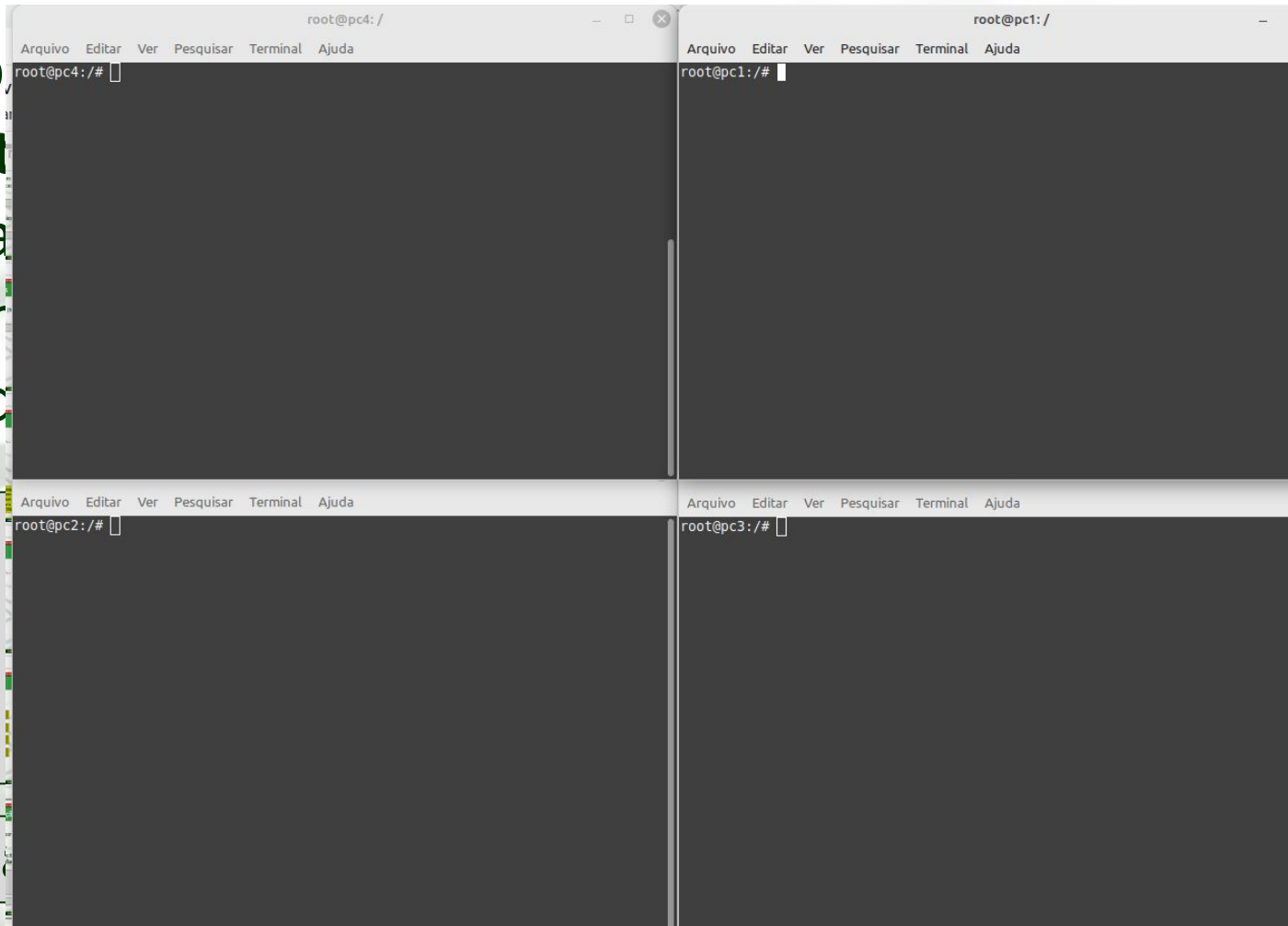
- É p...
- aut...
- O a...
- É ir...
- cac...

ar  
á.

para  
(nf )

pc1  
pc2  
pc3  
pc4

\$ k





# Comandos de Execução

- Também é possível definir **comandos de inicialização** para cada instância no ato da sua criação...

```
pc1[0]="A"  
pc2[0]="A"  
pc3[0]="A"  
pc4[0]="A"
```

```
pc1[exec]="ifconfig eth0 192.168.1.1/24"  
pc2[exec]="ifconfig eth0 192.168.1.2/24"  
pc3[exec]="ifconfig eth0 192.168.1.3/24"  
pc4[exec]="ifconfig eth0 192.168.1.4/24"
```





# Laboratório 01-4

- Usando arquivo de configuração, construa um novo cenário de laboratório...

**PC1**

Domínio de Colisão: A

**192.168.100.1/24**

**PC2**

Domínio de Colisão: B

**192.168.100.2/24**

**PC3**

Domínio de Colisão: B

**192.168.100.3/24**

**PC4**

Domínio de Colisão: A

**192.168.100.4/24**

- *Faça os teste e analise o comportamento do laboratório...*
- *Como seria possível resolver o problema?*



# Seminário Individual

## ■ Tema de Seminário Individual

### Utilitário brctl (Bridge-utils)

- *Para que serve.*
- *Principais comandos.*
- *Mostrar sua utilidade em cenário do Kathará.*

**LINK 1**

**LINK 2**