Manual do Usuário do WAVE

Leandro Cavalcanti de Almeida
(leandro.almeida@ifpb.edu.br)
Jefferson Lucas Ferreira da Silva
(ferreira.jefferson@academico.ifpb.edu.br)
Ricardo Pereira Lins
(ricardo.lins@academico.ifpb.edu.br)
Paulo Ditarso Maciel Jr.
(paulo.maciel@ifpb.edu.br)
Rafael Pasquini
(rafael.pasquini@ufu.br)
Fábio Luciano Verdi
(verdi@ufscar.br)

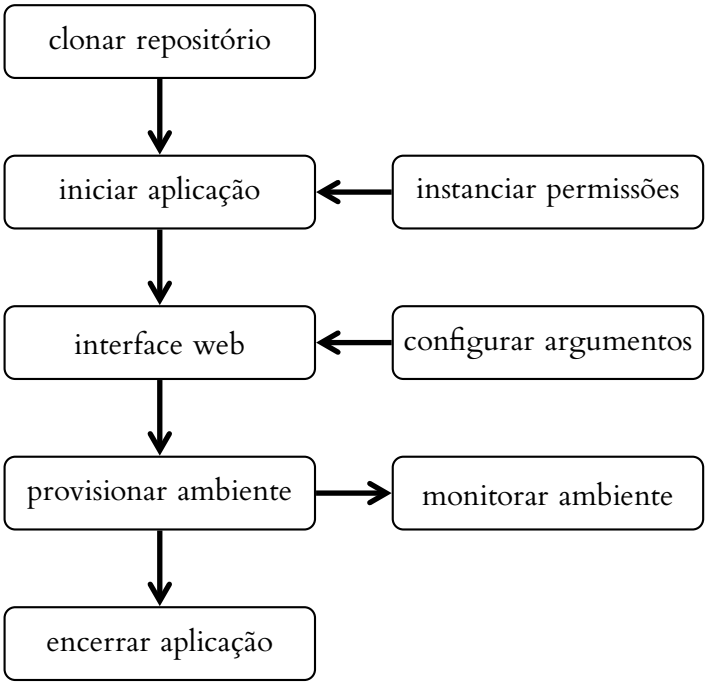
https://github.com/ifpb/wave/
versão 1.0, 23 de março de 2023

Resumo

Este manual apresenta a ferramenta WAVE, um gerador de carga de trabalho para experimentos verificáveis. Tendo como característica ser agnóstico à aplicação, o WAVE foi amplamente executado em testes de carga para aplicações de vídeo sob demanda e armazenamento de chave-valor. Na versão atual, o WAVE é capaz de gerar cargas para dois padrões distintos: sinusoidal e *flashcrowd*.

Sumário

Guia Rápido do Fluxo de Trabalho



1 Verificando os Requisitos Necessários

1.1 Verificando a se o **`Python3`** [7] está instalado e qual versão:

```
usuario@wave: ~ 70x5
usuario@wave:~$ python3 --version
Python 3.10.6
usuario@wave:~$ # caso contrário
usuario@wave:~$ sudo apt update && sudo apt install -y python3
```

1.2 Adicionalmente, é necessário o ambiente virtual **`VirtualEnv`**:

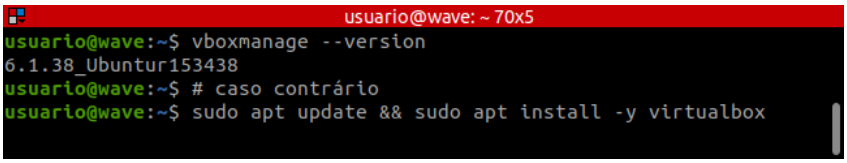
```
usuario@wave: ~ 70x5
usuario@wave:~$ pip3 show virtualenv | grep Version
Version: 20.21.0
usuario@wave:~$ # caso contrário
usuario@wave:~$ python3 -m pip install virtualenv
```

1.3 Verificando os componentes **`Docker`** [3] e **`docker-compose`**:

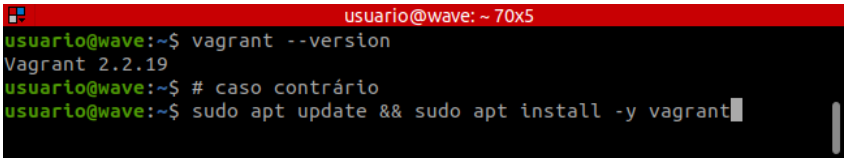
```
usuario@wave: ~ 70x5
usuario@wave:~$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
usuario@wave:~$ # caso contrário
usuario@wave:~$ sudo apt update && sudo apt install -y docker
```

```
usuario@wave: ~ 70x5
usuario@wave:~$ docker-compose --version
docker-compose version 1.29.2, build unknown
usuario@wave:~$ # caso contrário
usuario@wave:~$ sudo apt update && sudo apt install -y docker-compose
```

1.4 Verificando qual versão do **`VirtualBox`** [10] está instalada:



1.5 Verificando qual versão do **Vagrant** [9] está instalada:



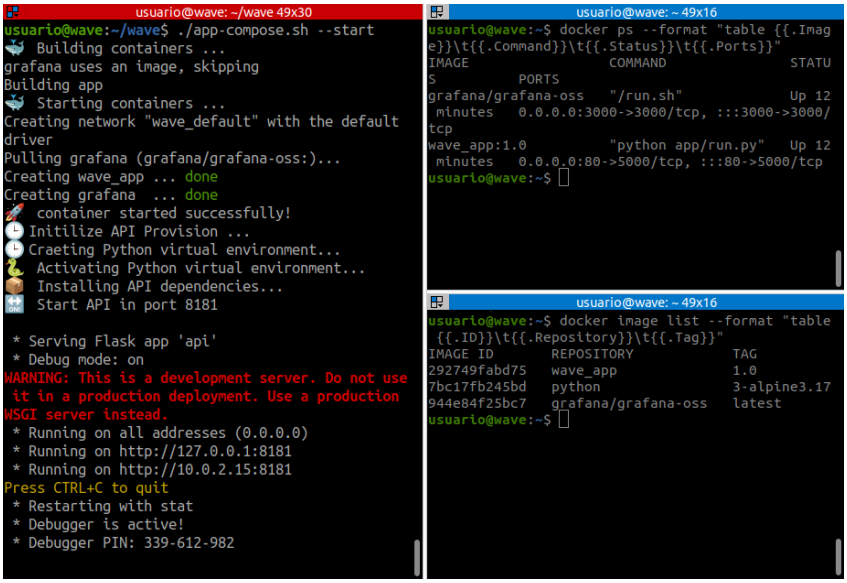
As versões apresentadas nas figuras dos passos 1.1–1.5 foram aquelas testadas no momento de elaboração deste manual.

2 Baixando o Código e Iniciando o Ambiente

2.1 Clonando o repositório oficial e iniciando o sistema:

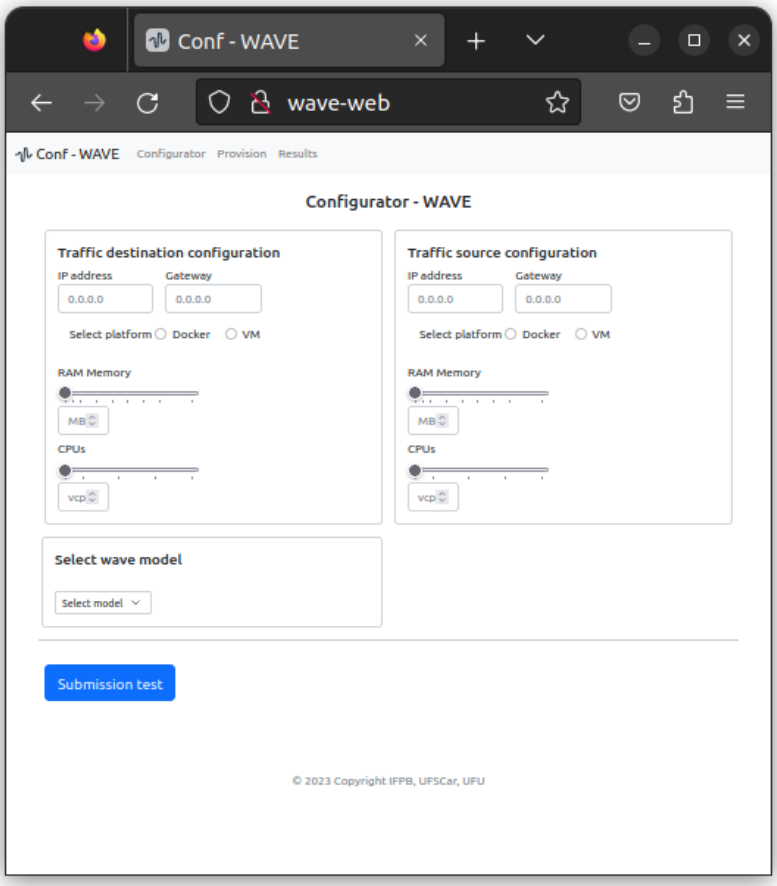
```
1  $ git clone https://github.com/ifpb/wave.git
2  $ cd wave
3  $ ./app-compose.sh --start
```

2.2 Verificando a execução em ambiente **Docker**:



Como pode ser observado na figura acima, o módulo de Inicialização do WAVE utiliza dois contêineres para sua execução: `wave_app` e `grafana-oss`. Ao lado esquerdo da figura temos a saída do comando de inicialização do WAVE, ou seja, saída da linha 3 do código exibido na seção 2.1.

2.3 O módulo WAVE Web pode ser acessado via navegador:



O formulário contém campos para inserir dados de rede tanto da origem da carga de tráfego quanto do destino. Além do endereço IP e *gateway* (caso origem e destino estiverem em redes separadas), é possível selecionar o provisionamento do ambiente através de máquina virtual com configuração de tamanho de memória e quantidade de CPUs virtuais. A opção de provisionamento por contêiner e a comunicação via *gateway* são funcionalidades futuras, ainda não implementadas. Por fim, o usuário pode escolher qual modelo de carga de trabalho ele deseja aplicar, seja *sinusoid* ou *flashcrowd*.

2.4 Ambiente de virtualização iniciado pelo módulo de Provisionamento:

As máquinas virtuais utilizadas como origem e destino da carga de tráfego estão configuradas a partir do diretório `app/provision`. Para tanto, a ferramenta *Vagrant* e o hipervisor *VirtualBox* são instanciados para execução do ambiente. Neste diretório encontra-se um arquivo `Vagrantfile` previamente editado, cujas configurações são alimentadas por um outro arquivo denominado `config.yaml`. Este último será gerado após o preenchimento do formulário no módulo WAVE WEB, contendo os argumentos que foram atribuídos pelo usuário. Além disso, foram pré-configuradas duas `vagrant boxes` (`wave-server` e `wave-client`) com os softwares necessários para ge-

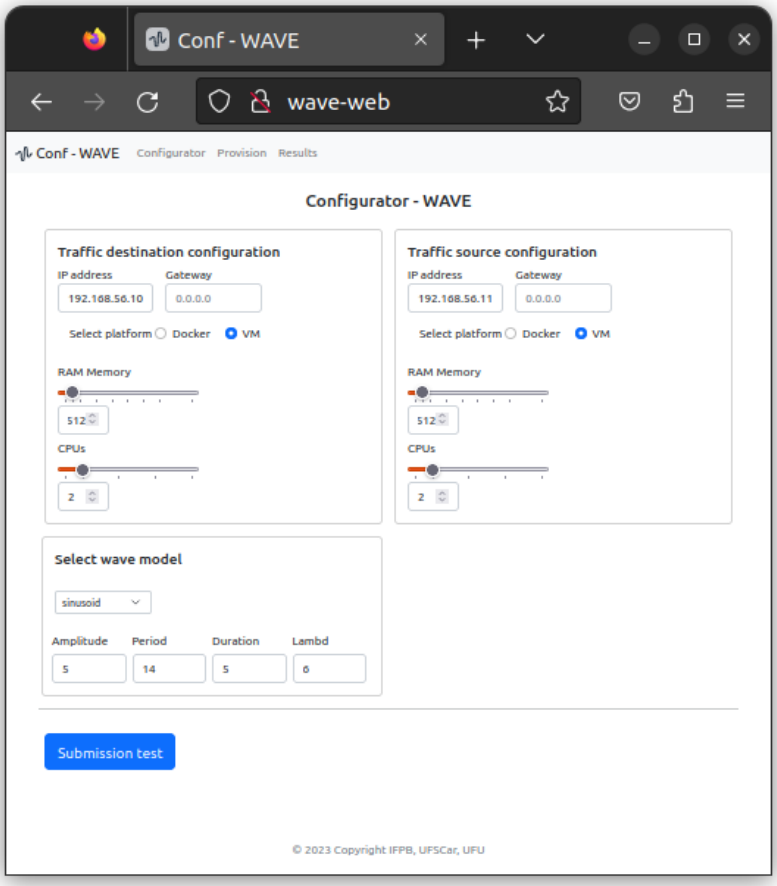
ração de carga entre a origem e o destino (`IPerf` [5] e modelos de carga de tráfego), respectivamente denominadas VM Cliente e VM Servidor. Caso as `boxes` ainda não estejam instaladas no computador local, estas serão baixadas na primeira execução do sistema. Segue abaixo o código do `Vagrantfile` responsável por provisionar o ambiente.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.require_version '>= 1.6.0' # Minimum Vagrant version
VAGRANTFILE_API_VERSION = '2' # Vagrant API version

require 'yaml' # Require 'yaml' module

# Edit config.yml to change VM configuration details
machines = YAML.load_file(File.join(File.dirname(__FILE__),
↪  'config.yaml'))

$SCRIPT = <<-EOF
server_ip=$(grep 'ip:' /vagrant/config.yaml | cut -d: -f2 |
↪  head -n1 | sed -e 's/\"//g')
echo -e "$server_ip server" | sudo tee -a /etc/hosts
client_ip=$(grep 'ip:' /vagrant/config.yaml | cut -d: -f2 |
↪  tail -n1 | sed -e 's/\"//g')
echo -e "$client_ip client" | sudo tee -a /etc/hosts
EOF

# Create boxes
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

# Iterate through entries in YAML file to create VMs
machines.each do |machines|

  # Configure the VMs per details in config.yml
  config.vm.define machines['traffic'] do |set|

    # Specify the hostname of the VM
    set.vm.hostname = machines['traffic']

    if machines['traffic'] == "server"
      set.vm.box = "ifpb/wave-server" # VM Server box
      set.vm.hostname = "wave-server" # VM Server hostname
    else
      set.vm.box = "ifpb/wave-client" # VM Client box
      set.vm.hostname = "wave-client" # VM Client hostname
```

```
37      end
38
39      # Iterate through networks as per settings in machines
40      set.vm.network "private_network", ip: machines['ip']
41
42      set.vm.provider 'virtualbox' do |vb|
43        vb.memory = machines['ram'] # Configure RAM
44        vb.cpus = machines['vcpu'] # Configure CPU
45      end # set.vm.provider 'virtualbox'
46
47      set.vm.provision "shell", inline: $SCRIPT
48
49    end # config.vm.define
50  end # machines.each
51  end # Vagrant.configure
```

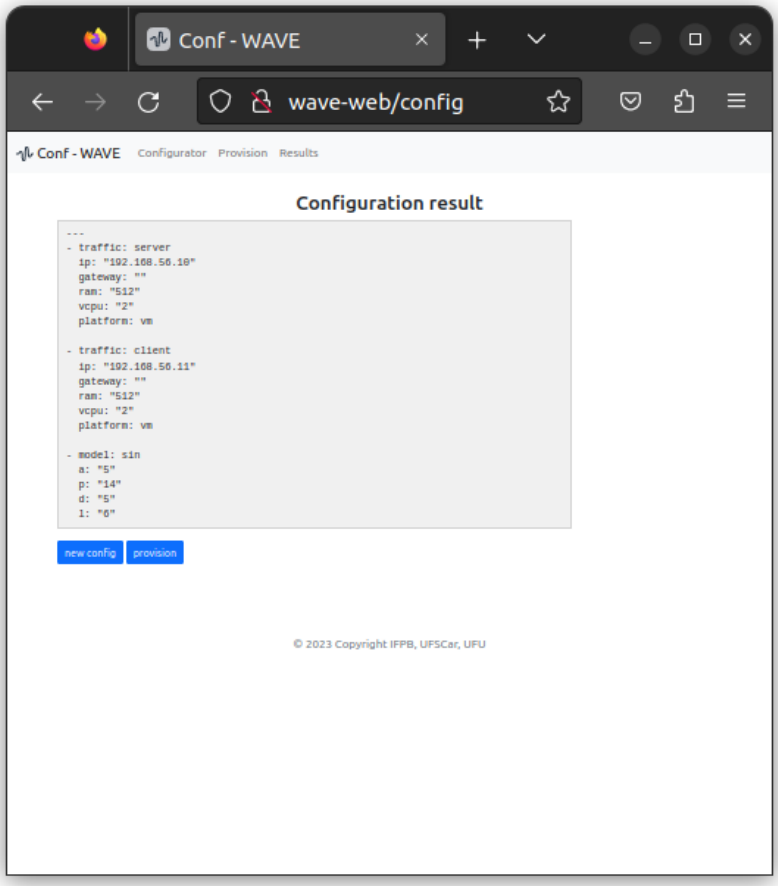
3 Interagindo com o WAVE WEB

3.1 Um vez inicializado o sistema, o usuário deverá inserir os argumentos:



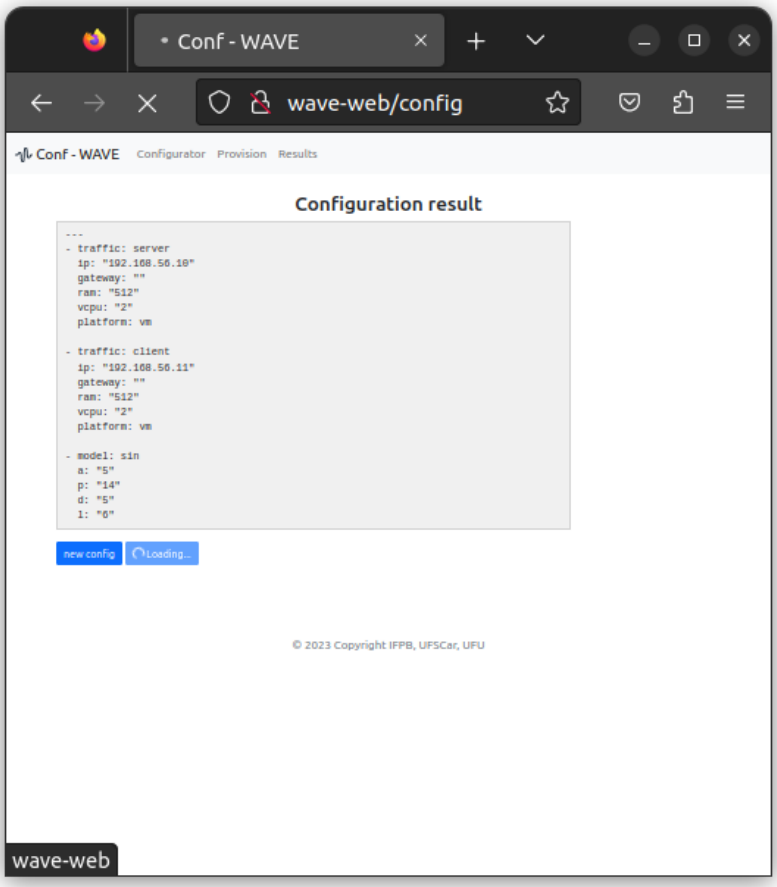
A figura anterior exemplifica o preenchimento do formulário Web com os parâmentros necessários para execução de uma carga de trabalho segundo um modelo *sinusoid*. Primeiramente, são configurados os endereços IPs da VM Servidor e VM Cliente. Atenção para um passo importante na configuração das VMs. Como o ambiente é configurado para utilizar interfaces de rede virtuais no modo *private* (linha 40 do `Vagrantfile` acima) ou *host-only* no `VirtualBox`, o `Vagrant` exige que tais interfaces sejam configuradas na faixa de endereços `192.168.56.0/24`. Para utilização de interfaces no modo *bridge*, basta alterar o termo *private* para *public* no `Vagrantfile`. Além do endereçamento, foram atribuídos valores para a quantidade de memória virtual e CPU virtuais em cada VM. Um modelo de carga *sinusoid* é instanciado na parte inferior do formulário, através dos seus respectivos argumentos. Para uma melhor compreensão dos modelos de carga suportados, recomenda-se a leitura em [1, 2, 8].

3.2 Consolidação dos argumentos inseridos:



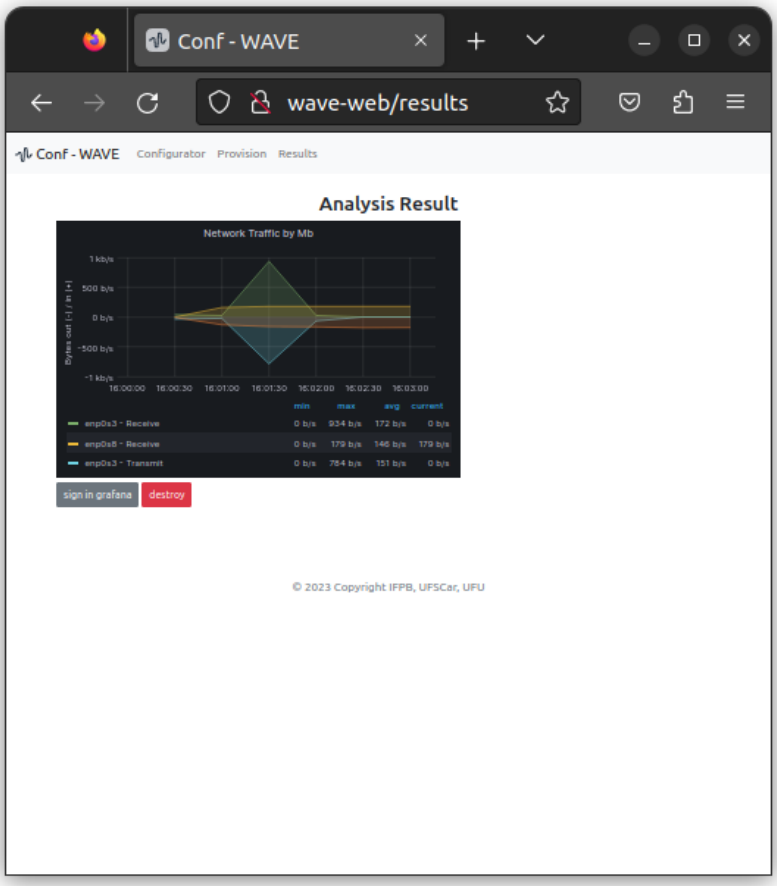
Após inserção dos argumentos no formulário da página inicial, o WAVE WEB apresenta uma página de conferência dos dados digitados. Tais argumentos são editados em um arquivo no formato YAML, denominado `config.yaml`. Caso o usuário perceba algum erro nos argumentos inseridos, poderá retornar à página inicial a partir do

botão de nova configuração, ou mesmo utilizando o menu na barra superior da página Web. Com os dados devidamente conferidos, o usuário pode então iniciar o provisionamento do ambiente.



Vale salientar que o módulo de Provisionamento, responsável por este processo de execução do ambiente, automaticamente configura e inicia as duas VMs envolvidas na geração e recepção da carga de tráfego. Mais especificamente, a VM Servidor é levantada já com o `IPerf` em execução no modo servidor, pronto para receber tráfego na sua porta padrão (5201). A VM Cliente, por sua vez, já é instanciada com o `IPerf` também instalado e com o código `Python` responsável pelos modelos de carga *sinusoid* e *flascrowd*. Também de forma automática, um dos modelos de carga é instanciado a partir dos parâmetros inseridos pelo usuário e a carga de trabalho (tráfego) específica é enviada à VM Servidor. De acordo com o modelo utilizado, instâncias do `IPef` no modo cliente são iniciadas e encerradas com o intuito de emular o comportamento desejado para o referido modelo de carga. Na VM Cliente é possível verificar o número de instâncias em execução a partir dos arquivos de *log* registrados. Adicionalmente, o módulo de Monitoramento permite verificar diversas métricas de execução da VM Servidor, como detalhado a seguir.

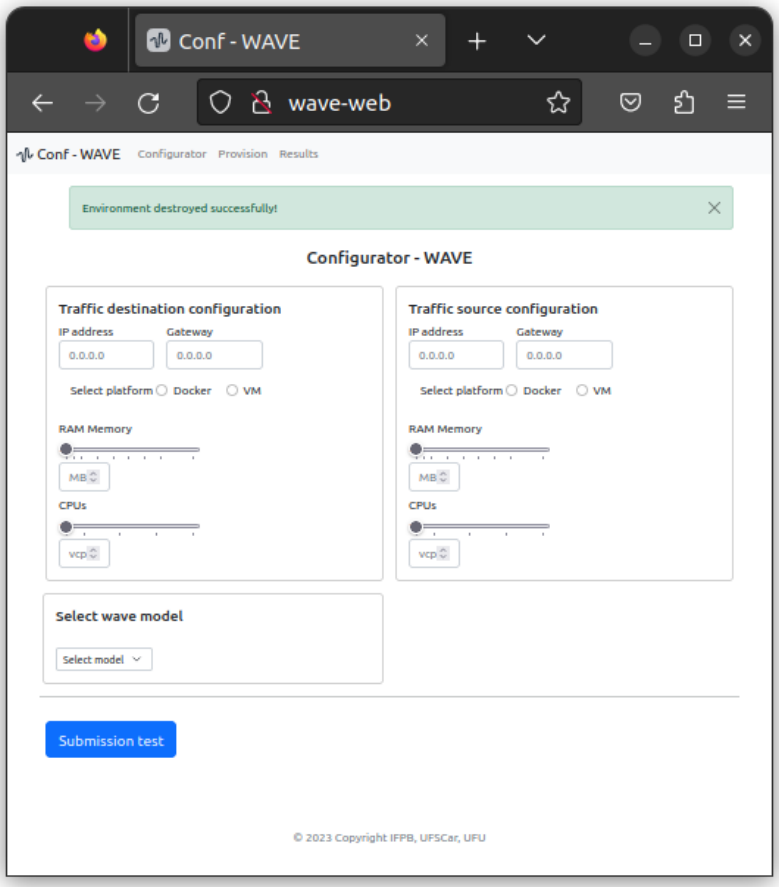
3.3 Uma vez provisionado o ambiente, segue uma página de verificação:



O módulo de Monitoramento é responsável por coletar e exibir métricas de execução da VM Servidor. Para tanto, são previamente configuradas na `vagrant box` do servidor as ferramentas `Prometheus` [6] e `node_exporter`. Utilizando o `Prometheus` como fonte de dados, o contêiner `Grafana` consegue exibir todas as métricas coletadas pelo `node_exporter`, dentre as quais estão várias relacionadas ao consumo de CPU, memória, disco e envio/recebimento de tráfego de rede. Após o provisionamento das VMs, a página acima exibe a taxa de bytes recebidos e enviados pelas interfaces de rede da VM Servidor. Com isso é possível, por exemplo, monitorar a execução do experimento, que consistem em enviar tráfego entre a origem e o destino de acordo com um modelo de carga específico. Adicionalmente, é possível abrir uma nova página Web com a interface do `Grafana`[1] [4] e consultar outras métricas de interesse, ou mesmo configurar novos painéis personalizados. Uma funcionalidade importante do módulo de Monitoramento, que está em fase de implementação, é gráfico adicional na página acima com o número de instâncias da aplicação executadas em função do tempo.

[1] Para o primeiro acesso, usuário e senha padrão é `admin`.

3.4 A qualquer momento é possível encerrar as máquinas virtuais:



Ao acionar o botão `destroy` na página de verificação do tráfego, o usuário é remetido à página inicial, onde poderá reiniciar o experimento caso seja de seu interesse.

4 Encerrando a Execução do WAVE

4.1 Finalizando e removendo o ambiente de contêineres:

```
1   $ ./app-compose.sh --destroy
```

Ao executar o comando acima, o usuário finaliza o módulo WAVE WEB e remove os contêineres responsáveis pelos demais módulos iniciados. Para reiniciar todo o sistema basta executar o mesmo comando, porém, substituindo o argumento `-destroy` por `-start`, como indicado na linha 3 do código na seção 2.1.

Referências

[1] Leandro Almeida, Fábio Verdi, and Rafael Pasquini. Estimando métricas de serviço através de in-band network telemetry. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 252–265, Porto Alegre, RS, Brasil, 2021. SBC.

[2] Ismail Ari, Bo Hong, Ethan Miller, Scott Brandt, and Darrell Long. Managing flash crowds on the internet. In *MASCOTS 2003*, pages 246– 249, 11 2003.

[3] Docker: Accelerated, Containerized Application Development. `https://www.docker.com/`. Acessado em 20/03/2023.

[4] Grafana: The open observability platform | Grafana Labs. `https://grafana.com/`. Acessado em 20/03/2023.

[5] IPerf - the ultimate speed test tool for TCP, UDP and SCTP- TEST the limits of your network + internet neutrality test. `https://iperf.fr/`. Acessado em 20/03/2023.

[6] Prometheus - Monitoring system & time series database. `https://prometheus.io/`. Acessado em 20/03/2023.

[7] Welcome to Python.org. `https://www.python.org/`. Acessado em 20/03/2023.

[8] Rolf Stadler, Rafael Pasquini, and Viktoria Fodor. Learning from network device statistics. *J. Netw. Syst. Manag.*, 25(4):672–698, 2017.

[9] Vagrant by HashiCorp. `https://www.vagrantup.com/`. Acessado em 20/03/2023.

[10] Oracle VM VirtualBox. `https://www.virtualbox.org/`. Acessado em 20/03/2023.