Deploy Zabbix Server, Frontend e Grafana no Docker

Primeiro passo corrigir o horário do servidor

Verificar o timezone atual

```
timedatectl status
```

Output:

```
               Local time: Wed 2020-06-03 05:03:25 -03
           Universal time: Wed 2020-06-03 08:03:25 UTC
                 RTC time: Wed 2020-06-03 08:03:24
                Time zone: America/New_York (-03, -0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Listar timezone disponiveis

```
timedatectl list-timezones | grep Sao_Paulo
```

Output:

```
America/Sao_Paulo
```

Definir timezone

```
timedatectl set-timezone America/Sao_Paulo
```

Verifique status do NTP

```
timedatectl status
```

```
               Local time: Wed 2020-06-03 05:03:25 -03
           Universal time: Wed 2020-06-03 08:03:25 UTC
                 RTC time: Wed 2020-06-03 08:03:24
                Time zone: America/Sao_Paulo (-03, -0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Ajustes de S.O após a instalação

Verificar se é necessário atualização do S.O

```
dnf clean all
dnf check-update
```

Instalar utilitários

```
dnf install -y net-tools vim nano epel-release wget curl tcpdump
```

Instalando o Docker

Instalando repositório do docker

```
dnf config-manager --add-
repo=https://download.docker.com/linux/centos/docker-ce.repo
```

Removendo repositórios antigos

```
dnf clean all
```

Listando versões disponiveis do docker

```
dnf list docker-ce --showduplicates | sort -r
```

Instalando device mapper

```
dnf install -y device-mapper-persistent-data
```

Instalando a ultima versão do Docker

Se precisar da ultima versão do docker, é necessário instalar manualmente a ultima versão do containerd.io

```
dnf install
https://download.docker.com/linux/centos/7/x86_64/stable/Packages/containe
```

```
rd.io-1.2.6-3.3.el7.x86_64.rpm
dnf install -y docker-ce
```

Se não, utilize o parâmetro --nobest na hora de instalação

```
dnf install -y docker-ce --nobest
```

Inicializando o daemon do docker

```
systemctl enable --now docker
systemctl status docker
```

Verificando versão do docker

```
docker version
```

Habilitar roteamento nos containers

```
firewall-cmd --zone=public --add-masquerade --permanent
firewall-cmd --reload
```

Ativando modo swarm no host

```
docker swarm init --advertise-addr 10.0.0.51
```

Inspect de todas as redes do docker

```
for net in `docker network ls |grep -v NAME | awk '{print $2}'`;do
ipam=`docker network inspect $net --format {{.IPAM}}` && echo $net -
$ipam; done
```

Validando conflito de rede

```
ifconfig ens18
```

Output:

```
ens18: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.34  netmask 255.255.255.0  broadcast 10.0.0.255
        inet6 fe80::d440:d1d9:33cf:3800  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::b5c3:70cc:d7a0:5d57  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::5b2:4fc5:e2cb:dbab  prefixlen 64  scopeid 0x20<link>
        ether ee:2d:73:2e:4b:9f  txqueuelen 1000  (Ethernet)
        RX packets 69824  bytes 177640772 (169.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 32110  bytes 2666365 (2.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Criando uma rede ingress com range diferente

Listando os nodes

```
docker node ls
```

Alterando disponibilidade do node no swarm

NESTE MOMENTO ELE NÃO IRA INICIAR NOVOS SERVIÇOS

<NODE_NAME> É O NOME DO HOSTNAME QUE RETORNA NO COMANDO docker node ls

```
docker node update --availability drain <NODE_NAME>
```

Removendo rede ingress atual

```
$ docker network rm ingress
```

Output:

```
WARNING! Before removing the routing-mesh network, make sure all the nodes
in your swarm run the same docker engine version. Otherwise, removal may
not
be effective and functionality of newly created ingress networks will be
impaired.
Are you sure you want to continue? [y/N]
```

Criando nova rede ingress

```
docker network create \
--driver overlay \
--ingress \
--subnet=192.168.102.0/28 \
--gateway=192.168.102.2 \
--opt com.docker.network.driver.mtu=1200 \
ingress
```

Ativando a disponibilidade do node novamente

<NODE_NAME> É O NOME DO HOSTNAME QUE RETORNA NO COMANDO docker node ls

```
docker node update --availability active <NODE_NAME>
```

Criando rede para o nosso ambiente

```
docker network create --driver overlay monitoring-network
```

Inspect de todas as redes do docker novamente

```
for net in `docker network ls |grep -v NAME | awk '{print $2}'`;do
ipam=`docker network inspect $net --format {{.IPAM}}` && echo $net -
$ipam; done
```

Adicionando outros nodes manager no cluster

Pegando Token para manager

```
docker swarm join-token manager
```

Criando regra de firewall em todos os nodes

```
firewall-cmd --parmanent --add-port=2377/tcp
firewall-cmd --permanent --add-port=7946/tcp
firewall-cmd --permanent --add-port=7946/udp
firewall-cmd --permanent --add-port=4789/udp
firewall-cmd --reload
```

Deploy stack Zabbix

Instalar o GIT

```
dnf install -y git
```

Clonando depositório

```
cd ~/
git clone <URL DO SEU GIT>
```

git@github.com:robertsilvatech/deploy-zabbix-docker-advanced.git

Inicianlizando a stack

```
cd <NOME_DA_PASTA_CLONADA>
sh grafana.sh
docker stack deploy -c docker-compose.yaml maratonazabbix
```

Listando stacks disponiveis

```
docker stack ls
```

Listando status dos serviços

```
docker service ls
```

Verificar logs

```
docker service logs -f maratonazabbix_zabbix-server
docker service logs -f maratonazabbix_zabbix-frontend
docker service logs -f maratonazabbix_grafana
```

Removendo stack

```
docker stack rm maratonazabbix
```