3.4. Операционные системы, лекция 2

1. На лекции мы познакомились с node\_exporter. В демонстрации его исполняемый файл запускался в background. Этого достаточно для демо, но не для настоящей production-системы, где процессы должны находиться под внешним управлением. Используя знания из лекции по systemd, создайте самостоятельно простой unit-файл для node\_exporter:
2. поместите его в автозагрузку,
3. предусмотрите возможность добавления опций к запускаемому процессу через внешний файл (посмотрите, например, на systemctl cat cron),
4. удостоверьтесь, что с помощью systemctl процесс корректно стартует, завершается, а после перезагрузки автоматически поднимается.

Мой Unit файл:

GNU nano 4.8 /etc/systemd/system/node\_exporter.service [root@vagrant:/usr/local# cat /etc/systemd/system/node\_exporter.service

[Unit]

Description=node\_exporter

Documentation=https://github.com/prometheus/node\_exporter/tree/master/docs

Wants=network.target

[Service]

ExecStart=/usr/local/bin/node\_exporter $OPTIONS

ExecReload=/usr/local/bin/node\_exporter -HUP $MAINPID

[Install]

WantedBy=multi-user.target

root@vagrant:/usr/local#

Попробовал засунуть его в автозагрузку, после рестарта вм “программа” заработала.

Так же попробовал перезагрузить “программу” заработала.

root@vagrant:/home/vagrant# systemctl status node\_exporter

● node\_exporter.service - node\_exporter

Loaded: loaded (/etc/systemd/system/node\_exporter.service; enabled; vendor preset: enabled)

Active: active (running) since Sat 2021-05-01 13:32:56 UTC; 22h ago

Docs: https://github.com/prometheus/node\_exporter/tree/master/docs

Main PID: 609 (node\_exporter)

Tasks: 3 (limit: 2281)

Memory: 10.2M

CGroup: /system.slice/node\_exporter.service

└─609 /usr/local/bin/node\_exporter

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - sockstat" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - stat" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - textfile" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - time" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - timex" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - uname" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - vmstat" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - xfs" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg=" - zfs" source="node\_exporter.go:97"

May 01 13:32:57 vagrant node\_exporter[609]: time="2021-05-01T13:32:57Z" level=info msg="Listening on :9100" source="node\_exporter.go:111"

2. Ознакомьтесь с опциями node\_exporter и выводом /metrics по-умолчанию. Приведите несколько опций, которые вы бы выбрали для базового мониторинга хоста по CPU, памяти, диску и сети.

curl -s<http://localhost:9100/metrics>

Метрики по HDD

Node\_filesystem\_avail\_bytes

Метрики по CPU

# HELP process\_cpu\_seconds\_total Total user and system CPU time spent in seconds.

# TYPE process\_cpu\_seconds\_total counter

process\_cpu\_seconds\_total 0.63

# HELP node\_memory\_Percpu\_bytes Memory information field Percpu\_bytes.

# TYPE node\_memory\_Percpu\_bytes gauge

node\_memory\_Percpu\_bytes 565248

node\_scrape\_collector\_duration\_seconds{collector="cpu"} 0.000456333

node\_scrape\_collector\_success{collector="cpu"} 1

Метрики по памяти

# HELP node\_memory\_MemFree\_bytes Memory information field MemFree\_bytes.

# TYPE node\_memory\_MemFree\_bytes gauge

node\_memory\_MemFree\_bytes 1.375215616e+09

# HELP node\_memory\_MemTotal\_bytes Memory information field MemTotal\_bytes.

# TYPE node\_memory\_MemTotal\_bytes gauge

node\_memory\_MemTotal\_bytes 2.084384768e+09

Метрики по сети

# TYPE node\_network\_transmit\_packets\_total counter

node\_network\_transmit\_packets\_total{device="eth0"} 10648

node\_network\_transmit\_packets\_total{device="lo"} 819

# TYPE node\_network\_transmit\_errs\_total counter

node\_network\_transmit\_errs\_total{device="eth0"} 0

node\_network\_transmit\_errs\_total{device="lo"} 0

# TYPE node\_network\_transmit\_drop\_total counter

node\_network\_transmit\_drop\_total{device="eth0"} 0

node\_network\_transmit\_drop\_total{device="lo"} 0

3. Установите в свою виртуальную машину Netdata. Воспользуйтесь готовыми пакетами для установки (sudo apt install -y netdata). После успешной установки:

в конфигурационном файле /etc/netdata/netdata.conf в секции [web] замените значение с localhost на bind to = 0.0.0.0,

добавьте в Vagrantfile проброс порта Netdata на свой локальный компьютер и сделайте vagrant reload:

config.vm.network "forwarded\_port", guest: 19999, host: 19999

После успешной перезагрузки в браузере на своем ПК (не в виртуальной машине) вы должны суметь зайти на localhost:19999. Ознакомьтесь с метриками, которые по умолчанию собираются Netdata и с комментариями, которые даны к этим метрикам.



4. Можно ли по выводу dmesg понять, осознает ли ОС, что загружена не на настоящем оборудовании, а на системе виртуализации?

По dmesg вроде понимает что работает в виртуализации.

root@vagrant:/var/log# dmesg | grep -i virtual

[ 0.000000] DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006

[ 0.006001] CPU MTRRs all blank - virtualized system.

[ 0.183596] Booting paravirtualized kernel on KVM

[ 3.803492] systemd[1]: Detected virtualization oracle.

5. Как настроен sysctl fs.nr\_open на системе по-умолчанию? Узнайте, что означает этот параметр. Какой другой существующий лимит не позволит достичь такого числа (ulimit --help)?

Состояние по умолчанию:

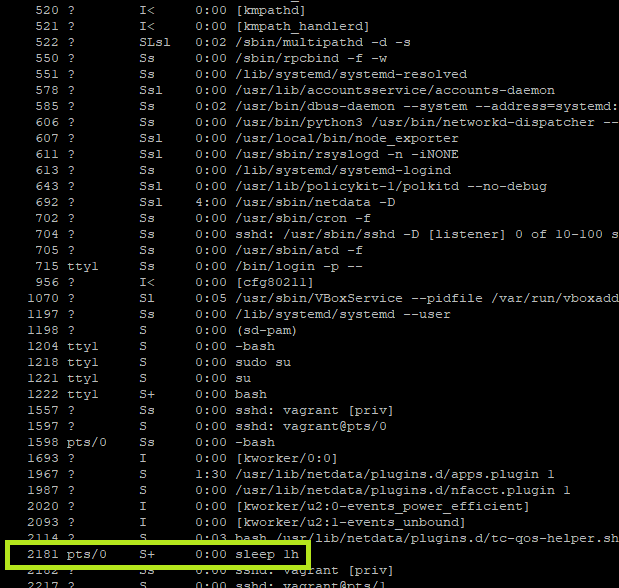
root@vagrant:/var/log# sysctl -n fs.nr\_open

1048576

Параметр означает количество открытых файлов.

Командой ulimit -n можно увеличить количество открытых файлов.

6. Запустите любой долгоживущий процесс (не ls, который отработает мгновенно, а, например, sleep 1h) в отдельном неймспейсе процессов; покажите, что ваш процесс работает под PID 1 через nsenter. Для простоты работайте в данном задании под root (sudo -i). Под обычным пользователем требуются дополнительные опции (--map-root-user) и т.д.



Найдите информацию о том, что такое :(){ :|:& };:. Запустите эту команду в своей виртуальной машине Vagrant с Ubuntu 20.04 (это важно, поведение в других ОС не проверялось). Некоторое время все будет "плохо", после чего (минуты) – ОС должна стабилизироваться. Вызов dmesg расскажет, какой механизм помог автоматической стабилизации. Как настроен этот механизм по-умолчанию, и как изменить число процессов, которое можно создать в сессии?

:(){ :|:& };: - fork bomb ( заполняет всю память и переводить виртуальную машину в swap)

К сожалению не нашел в dmesg конкретного ответа какой механизм помог автоматически стабилизировать систему:

Предположительно это

[ 3.768056] systemd[1]: Inserted module 'autofs4'

[ 3.803438] systemd[1]: systemd 245.4-4ubuntu3.3 running in system mode. (+PAM +AUDIT +SELINUX +IMA +APPARMOR +SMACK +SYSVINIT +UTMP +LIBCRYPTSETUP +GCRYPT +GNUTLS +ACL +XZ +LZ4 +SECCOM>

Значение по умолчанию:

root@vagrant:/# ulimit -u

7606

-u the maximum number of user processes

Изменить число процессов которое можно запустить в сессию можно командой

ulimit и конфигурационного файла /etc/security/limits.conf