1. На лекции мы познакомились с [node\_exporter](https://github.com/prometheus/node_exporter/releases). 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](https://www.freedesktop.org/software/systemd/man/systemd.service.html) для node\_exporter:
   * поместите его в автозагрузку,
   * предусмотрите возможность добавления опций к запускаемому процессу через внешний файл (посмотрите, например, на systemctl cat cron),
   * удостоверьтесь, что с помощью systemctl процесс корректно стартует, завершается, а после перезагрузки автоматически поднимается.

Устанавливаем

wget https://github.com/prometheus/node\_exporter/releases/download/v1.1.2/node\_exporter-1.1.2.linux-amd64.tar.gz

tar xvfz node\_exporter-1.1.2.linux-amd64.tar.gz

cd node\_exporter-1.1.2.linux-amd64

./node\_exporter

Демонизируем

nano /etc/systemd/system/node\_exporter.service

[Unit]

Description=node\_exporter

[Service]

EnvironmentFile=/etc/sysconfig/node\_exporter

ExecStart=/home/vagrant/node\_exporter-1.1.2.linux-amd64/node\_exporter $OPTIONS

Restart=on-failure

[Install]

WantedBy=multi-user.target

Включаем автозагрузку

sudo systemctl enable node\_exporter.service

Пишем конфиг

sudo nano /etc/sysconfig/node\_exporter

OPTIONS="--collector.textfile.directory /var/lib/node\_exporter/textfile\_collector"

sudo systemctl daemon-reload

После этого из данной папки будут забираться файлы коллекторов

после перезагрузки видим, что сервис автоматически поднимается

sudo systemctl status node\_exporter.service

1. 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.

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

# TYPE process\_cpu\_seconds\_total counter

process\_cpu\_seconds\_total 0.44 – позволит увидеть загрузку CPU

# HELP node\_memory\_MemAvailable\_bytes Memory information field MemAvailable\_bytes.

# TYPE node\_memory\_MemAvailable\_bytes gauge

node\_memory\_MemAvailable\_bytes 7.7428736e+08

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

# TYPE node\_memory\_MemFree\_bytes gauge

node\_memory\_MemFree\_bytes 7.13125888e+08

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

# TYPE node\_memory\_MemTotal\_bytes gauge

node\_memory\_MemTotal\_bytes 1.028923392e+09

# HELP node\_memory\_Mlocked\_bytes Memory information field Mlocked\_bytes.

# TYPE node\_memory\_Mlocked\_bytes gauge

node\_memory\_Mlocked\_bytes 1.9079168e+07

Использование памяти

# HELP node\_network\_transmit\_bytes\_total Network device statistic transmit\_bytes.

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

node\_network\_transmit\_bytes\_total{device="eth0"} 1.041439e+06

node\_network\_transmit\_bytes\_total{device="lo"} 1.029256e+06

Мониторинг трафика к примеру

# HELP node\_disk\_read\_bytes\_total The total number of bytes read successfully.

# TYPE node\_disk\_read\_bytes\_total counter

node\_disk\_read\_bytes\_total{device="dm-0"} 1.83677952e+08

node\_disk\_read\_bytes\_total{device="dm-1"} 3.342336e+06

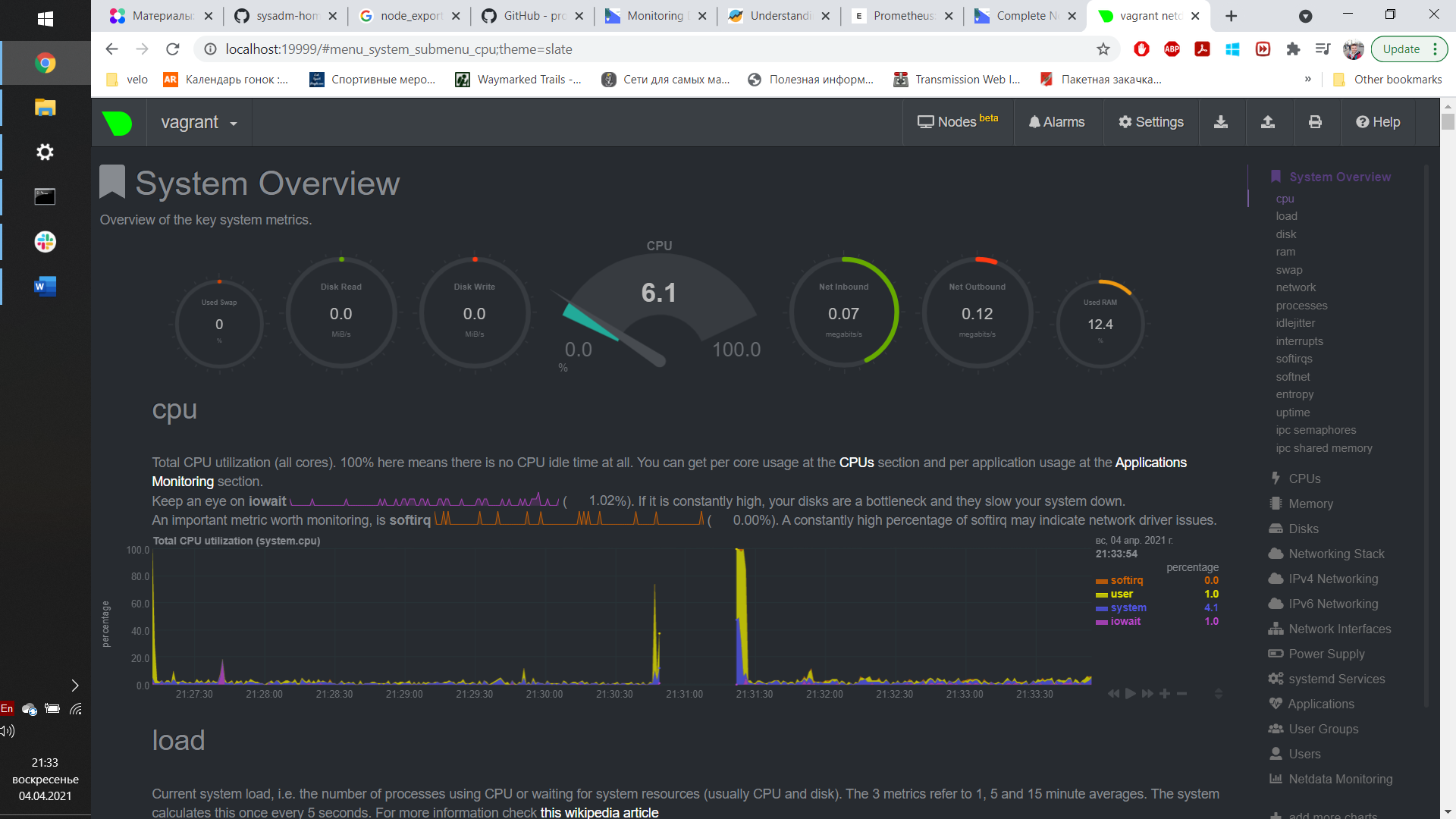
node\_disk\_read\_bytes\_total{device="sda"} 1.97436416e+08

Чтение с дисков

1. Установите в свою виртуальную машину [Netdata](https://github.com/netdata/netdata). Воспользуйтесь [готовыми пакетами](https://packagecloud.io/netdata/netdata/install) 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.



Это задание – самое простое, что было в рамках этого обучения вообще )

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

vagrant@vagrant:~$ sudo dmesg | grep virtual

[ 0.004593] CPU MTRRs all blank - virtualized system.

[ 0.136818] Booting paravirtualized kernel on KVM

[ 3.019676] systemd[1]: Detected virtualization oracle.

1. 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)?

fs.nr\_open  по умолчанию имеет значение 1048576 и означает максимальное количество открытых файлов, однако для каждой сессии по умолчанию устанавливает параметр ulimit -n равный 1024, изменив его, удастся достичь увеличения количества открытых файлов.

1. 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.

root@vagrant:/home/vagrant# sudo unshare -u bash

root@vagrant:/home/vagrant# ls -l /proc/$$/ns

total 0

lrwxrwxrwx 1 root root 0 Apr 4 19:19 cgroup -> 'cgroup:[4026531835]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 ipc -> 'ipc:[4026531839]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 mnt -> 'mnt:[4026531840]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 net -> 'net:[4026531992]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 pid -> 'pid:[4026531836]'

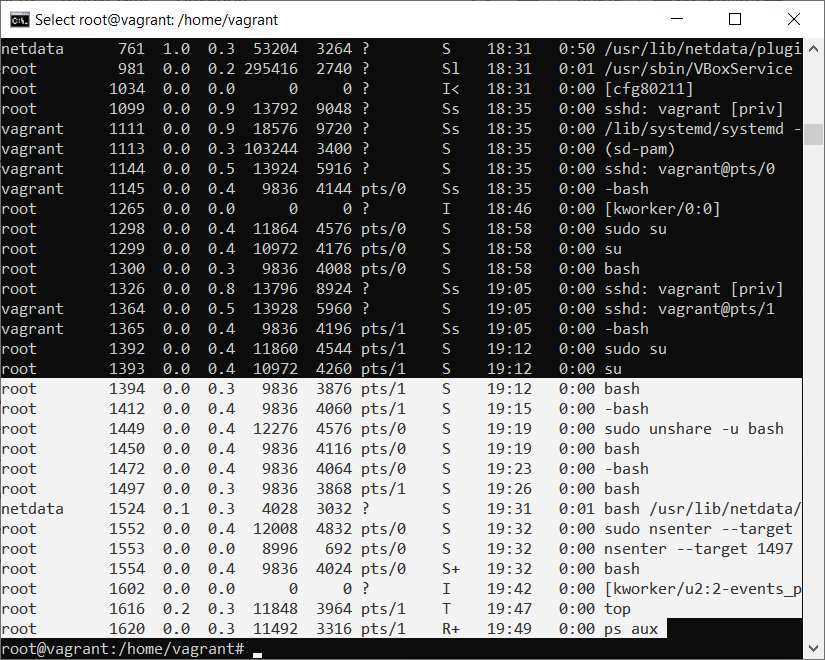
lrwxrwxrwx 1 root root 0 Apr 4 19:19 pid\_for\_children -> 'pid:[4026531836]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 user -> 'user:[4026531837]'

lrwxrwxrwx 1 root root 0 Apr 4 19:19 uts -> 'uts:[4026532194]'

Позапускал кучу всяких башей, но не вижу, где и как найти, что они работают под pid1

*Задание 6  
Примерный алгоритм выполнения:  
\* перемонтировать /proc;  
\* запустить процесс в неймспейсе PID;  
\* запустить процесс в неймспейсе MNT;  
\* вызвать ps через nsenter*



1. 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?

:(){ :|:& };:

..определяет функцию с именем : , которая порождает саму себя (дважды, одна труба в другую) и создает фон.

С разрывами строк:

:()

{

:|:&

};

:

Переименование функции : в forkbomb :

forkbomb()

{

forkbomb | forkbomb &

};

forkbomb

Вы можете предотвратить такие атаки, используя ulimit для ограничения количества processes-per-user:

$ ulimit -u 50

$ :(){ :|:& };:

-bash: fork: Resource temporarily unavailable

$

Более постоянно вы можете использовать /etc/security/limits.conf (по крайней мере, на Debian и других), например:

\* hard nproc 50

Конечно, это означает, что вы можете запустить только 50 процессов, вы можете увеличить это количество в зависимости от того, что делает машина!

[ 4755.885122] cgroup: fork rejected by pids controller in /user.slice/user-1000.slice/session-1.scope

[ 6057.702202] cgroup: fork rejected by pids controller in /user.slice/user-1000.slice/session-3.scope

