admin_doc

Документация для администратора

[1] Настраиваем доступ к серверу SSH по ключу

Если используется не YandexCloud, то нужно настроить SSH доступ к серверу по ключу (на который будет выполнятся установка)
Копируем публичный ключ со своей машины на сервер

```
ssh-copy-id -i ~/.ssh/id_rsa.pub admin@REMOTE_SERVER_IP
```

Закрываем доступ по паролю и использование root логина
В файле /etc/ssh/sshd_config выставляем такие значения директивам

```
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
PermitRootLogin No
PasswordAuthentication No
PermitEmptyPasswords No
```

Перезагружаем сервер sshd

```
sudo systemctl restart sshd
```

[2] Устанавливаем пакет gpg

```
sudo apt update && sudo apt install gpg
```

После установки запускаем генерацию пары ключей (приватный и публичный)

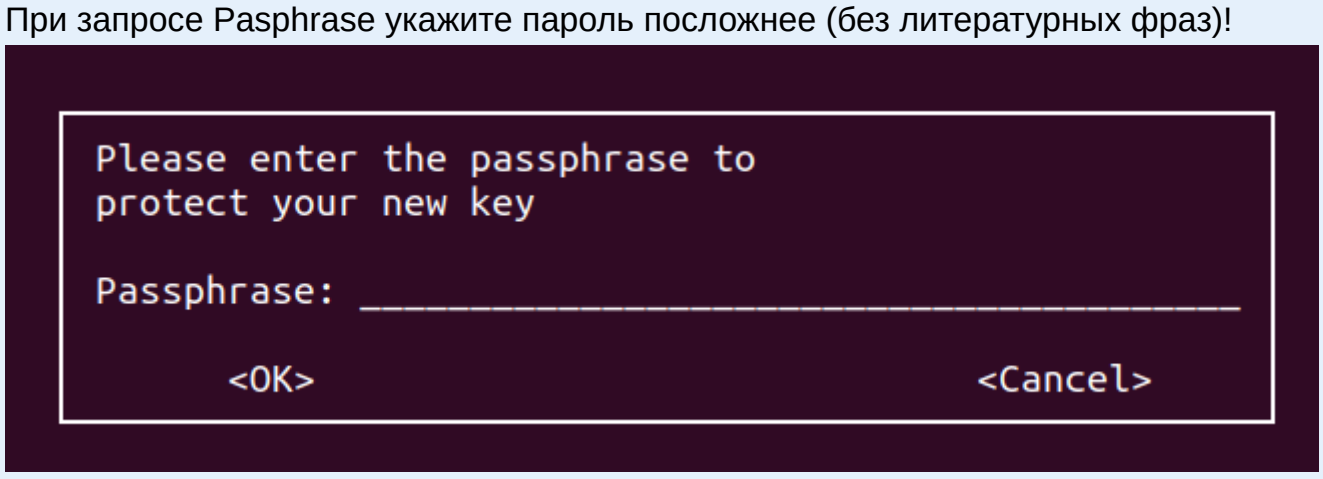
```
gpg --gen-key
```

будут запрошены данные для генерации пары GPG ключей (закрытый и открытый). Этими ключами можно будет шифровать-дешифровать файлы, для безопасной передачи между вами и сервером PKI.

Программа запросит данные: Имя, Email и длину ключа (от 1024 до 4096, использовать ключи размера меньше 2048 бит небезопасно), срок действия ключа, passphrase и др (укажите реальное Имя и свой email адрес, с которого вы будите отправлять файлы системному администратору PKI). После чего, ключи сохранятся во внутреннюю память системы (файлы созданы не будут!).

```
toorr2p@pki-ovpn-001:~/clients/client-1$ gpg --gen-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: pki-ovpn-001
Email address: admin@pki-ovpn-001.com
You selected this USER-ID:
    "pki-ovpn-001 <admin@pki-ovpn-001.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key 63C94CBE240B4D3C marked as ultimately trusted
gpg: directory '/home/toorr2p/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/toorr2p/.gnupg/openpgp-revocs.d/6
public and secret key created and signed.

pub   rsa3072 2023-12-18 [SC] [expires: 2025-12-17]
      66D8B394C1DECAD854CE768D63C94CBE240B4D3C
uid                      pki-ovpn-001 <admin@pki-ovpn-001.com>
sub   rsa3072 2023-12-18 [E] [expires: 2025-12-17]
```

При запросе Pasphrase укажите пароль посложнее (без литературных фраз)!



В итоге будут создана пара приватный-публичный ключи

```
public and secret key created and signed.

pub   rsa3072 2023-12-18 [SC] [expires: 2025-12-17]
      63206BF64DF59ED2E839D18F0D29877BCA9EF39E
uid                      Client1 <client1@gmail.com>
sub   rsa3072 2023-12-18 [E] [expires: 2025-12-17]
```

где:

- server-001.ru - то что было указано в запросе "Name" (укажите реальное ФИО латиницей!)
- admin@server-001.ru - должен быть указан ваш реальный email

Не забудьте создать сертификаты отзыва для ключей!

подробнее см. статью

[3] Устанавливаем пакеты для PKI на сервера

Скачиваем .deb пакеты из репозитария

- pki_0.1-1_all.deb
- ovpn_0.1-1_all.deb
- bac_0.1-1_all.deb

Запускаем установку пакетов

```
sudo apt install ./pki_0.1-1_all.deb ./ovpn_0.1-1_all.deb
```

ВАЖНО: меняем владельца и группу всех файлов и директорий для easy-rsa на пользователя администратора сервера и его группу (напр. если логин admin и его группа admin)

```
sudo bash -c 'chown -R admin:admin /opt/easy-rsa'
```

Запускаем сервер OpenVPN

```
sudo systemctl enable openvpn-server@server.service
```

```
sudo systemctl start openvpn-server@server.service
```

```
sudo systemctl status openvpn-server@server.service
```

Чек лист тестирования после установки

[1] Проверяем статус сервера openvpn

```
sudo systemctl status openvpn-server@server.service
```

Если статус не active то смотрим логи сервера

```
sudo cat /var/log/openvpn/openvpn.log
```

Также ошибки можно смотреть в системном журнале

```
journalctl -f
```

[2] Проверяем что сервер работает на 1194 порту

```
netstat -nlup
```

[3] Убеждаемся что сервер PKI доступен извне по UDP на 1194 порту

```
sudo nmap -sU -p U:1194 <SERVER_IP>
```

[4] Резервное копирование

Резервное копирование состоит из трех блоков

- **gpg** - пакет для шифрования файла резервной копии
- **s3cmd** - пакет для отправки резервной копии в облако SELECTEL
- **bac_0.1-1_all.deb** - пакет добавляющий сервис резервного копирования в систему

Далее сценарий по установке требуемых пакетов и их конфигурации:

[1] gpg

Ставим пакет если ранее он не был установлен

```
sudo apt update && sudo apt install gpg
```

Создаем ключи на сервере (если не были созданы ранее)

```
gpg --gen-key
```

Импортируем ключ GPG, ключ со своей локальной машины
для того что бы в случае взлома или краха системы на сервере PKI, иметь возможность расшифровки файлов на локальной машине и далее уже восстанавливать конфигурацию. Запоминаем email (вы его должны знать т.к он же с вашей локальной системы), этот Email является идентификатором ключа, его нужно будет указать при установке пакета bac_0.1-1_all.deb

Чтобы просмотреть ID ключа на своей локальной машине используйте команду

```
gpg --list-key
```

Что бы записать свой ключ в файл используйте команду

```
gpg --export -a "toorrp4@gmail.com" > ~/.ssh/gpg.key
```

Затем передаем файл с GPG ключом на сервер PKI

```
scp /tmp/gpg.key LOGIN@PKI_SERVER_IP:/tmp
```

Переходим на сервер и там импортируем ключ из этого файла

```
gpg --import /tmp/gpg.key
```

Просмотреть все ключи в системе (доступные текущему пользователю)

```
gpg --list-key
```

Подписываем импортированный ключ

```
gpg --sign-key toorrp4@gmail.com
```

[2] s3cmd

В качестве хранилища предусмотрен провайдер SELECTEL
Следует предварительно настроить облако (создать контейнер, настроить правила доступа к контейнеру, создать сервисного пользователя и S3 ключи для этого пользователя) подробнее описано в файле "Работа с SELECTEL хранилищем.pdf"

Устанавливаем пакет

```
sudo apt update && sudo apt install s3cmd
```

Конфигурация s3cmd - доступ к хранилищу

```
s3cmd --configure
```

Указываем значения при запросах (основные, для остальных можно просто нажать Enter):

```
Access Key YOUR_S3_KEY
Secret Key YOUR_S3_SECRET_KEY
Default Region [US]: ru-1
```

```
S3 Endpoint [s3.amazonaws.com]: s3.ru-1.storage.selcloud.ru
DNS-style bucket+hostname:port template for accessing a bucket [%(bucket)s.s3.amazonaws.com]: s3.ru-
1.storage.selcloud.ru
```

При запросе "Test access with supplied credentials? [Y/n]" указываем "Y" т.к требуется проверить соединение с облаком!

```
Access key and Secret key are your identifiers for Amazon S3. Leave them empty for using the env variables.
Access Key: 83e39[redacted]bed9da943c02c325
Secret Key: bcee1[redacted]3883f705db89
Default Region [US]: ru-1

Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.
S3 Endpoint [s3.amazonaws.com]: s3.ru-1.storage.selcloud.ru

Use "%(bucket)s.s3.amazonaws.com" to the target Amazon S3. "%(bucket)s" and "%(location)s" vars can be used
if the target S3 system supports dns based buckets.
DNS-style bucket+hostname:port template for accessing a bucket [%(bucket)s.s3.amazonaws.com]: s3.ru-1.storage.selcloud.ru

Encryption password is used to protect your files from reading
by unauthorized persons while in transfer to S3
Encryption password:
Path to GPG program [/usr/bin/gpg]:

When using secure HTTPS protocol all communication with Amazon S3
servers is protected from 3rd party eavesdropping. This method is
slower than plain HTTP, and can only be proxied with Python 2.7 or newer
Use HTTPS protocol [Yes]:

On some networks all internet access must go through a HTTP proxy.
Try setting it here if you can't connect to S3 directly
HTTP Proxy server name:

New settings:
  Access Key: 83e3[redacted]4ffdbed[redacted]2c325
  Secret Key: bcee1b[redacted]d838[redacted]5db89
  Default Region: ru-1
  S3 Endpoint: s3.ru-1.storage.selcloud.ru
  DNS-style bucket+hostname:port template for accessing a bucket: s3.ru-1.storage.selcloud.ru
  Encryption password:
  Path to GPG program: /usr/bin/gpg
  Use HTTPS protocol: True
  HTTP Proxy server name:
  HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] Y
```

Если тест соединения с облаком прошел успешно, обязательно указываем "y" для сохранения конфигурации

```
Test access with supplied credentials? [Y/n] Y
Please wait, attempting to list all buckets...
Success. Your access key and secret key worked fine :-)

Now verifying that encryption works...
Not configured. Never mind.

Save settings? [y/N] y
```

После сохранения настроек, будет выведен файл куда эти настройки сохранены, а также запрошено имя контейнера в SELECTEL, указываем его и жмем Enter

```
Save settings? [y/N] y
Configuration saved to '/root/.s3cfg'
Enter cloud storeage container name: test
Progress: [ 33%] [##########################################.
 main*  0  0  0
```

[3] bac_0.1-1_all.deb

Это тот самый пакет который добавит сервис резервного копирования в систему, который в свою очередь будет запускать bash скрипт выполняющий всю работу по добавлению нужных файлов и директорий в архив резервной копии, шифрование этого архива и отправку его в сетевое хранилище. При установке пакета будут запрошены такие данные:

- **"Enter your GPG key (email) which can decrypt the backup archive:"** - ID GPG ключа, это тот email нашего ключа с локальной машины, который был импортирован ранее
- **"Enter cloud storeage container name:"** - имя контейнера в SELECTEL
- **"Enter user under which to run the service:"** - имя пользователя и группу под которым запускать сервис. Здесь укажите то что выводить команда id

Скачиваем пакет из репозитария bac_0.1-1_all.deb

Устанавливаем пакет командой

```
sudo apt install ./bac_0.1-1_all.deb
```

После установки будет выведен перечень команд для запуска сервиса

```
sudo systemctl enable bac.timer
sudo systemctl start bac.timer
sudo systemctl enable bac.service
sudo systemctl start bac.service
sudo systemctl status bac.timer
```

Конфиги и файлы сервиса >

- **/etc/bac.conf** - конфиг с перечнем файлов и директорий попадающих в резервную копию
- **/usr/bin/bac.sh** - скрипт выполняющий резервирование, шифрование и отправку в облако
- **/etc/systemd/system/bac.timer** - сервис который запускается 1 раз в день
- **/etc/systemd/system/bac.service** - сервис запускающий скрипт bac.sh

ВАЖНО: для быстрого тестирования запуск сервиса в Unit файле bac.timer настроен на 1 раз в минуту, после того как вы убедились что файл резервной копии появился в контейнере (после запуска сервисов), следует изменить это значение на один раз в 24ч (или на любое другое).
Требуется отредактировать файл Unit

```
sudo vim /etc/systemd/system/bac.timer
```

Указав там такое значение (например 24h)

```
[Unit]
Description=Runs backup bac.service

[Timer]
OnUnitActiveSec=24h
Unit=bac.service

[Install]
WantedBy=multi-user.target
```

И перезагрузить системные демоны, командой

```
sudo systemctl daemon-reload
```

Ошибки сервиса >

Если видим что в облаке нет архва на нужную дату, при том что сервис активет, нужно смотреть журнал системы с фильтром по сервису

```
sudo journalctl -f -u bac.service
```

Там может быть ошибка шифрования архива "gpg: signing failed: Inappropriate ioctl for device", которая возникает если GPG ключ получателя, не подписан (например подпись слетает при перезагрузке VM). Для исправления нужно переподписать ключ, который передается на вход баш скрипту в файле Unit сервиса:

```
/etc/systemd/system/bac.service
```

```
Dec 26 20:50:54 pki-ovpn-002 bac.sh[31454]: gpg: signing failed: Inappropria
te ioctl for device
Dec 26 20:50:54 pki-ovpn-002 bac.sh[31454]: gpg: /tmp/bac_26122023.tar.gz: s
ign+encrypt failed: Inappropriate ioctl for device
Dec 26 20:50:54 pki-ovpn-002 bac.sh[31458]: gpg: Note: '--list-key >> '/var/
log/bac.log';' is not considered an option
Dec 26 20:50:54 pki-ovpn-002 bac.sh[31458]: gpg: WARNING: no command supplie
d.  Trying to guess what you mean ...
Dec 26 20:50:54 pki-ovpn-002 bac.sh[31458]: usage: gpg [options] [filename]
Dec 26 20:50:54 pki-ovpn-002 sudo[31459]:  toorr2p : TTY=unknown ; PWD=/tmp
; USER=root ; COMMAND=/usr/bin/bash -c echo \'[26.12.2023_20-50-53] Error GP
G encrypt file array fail\' >> \'/var/log/bac.log\';
Dec 26 20:50:54 pki-ovpn-002 sudo[31459]: pam_unix(sudo:session): session op
ened for user root by (uid=0)
Dec 26 20:50:54 pki-ovpn-002 sudo[31459]: pam_unix(sudo:session): session cl
osed for user root
Dec 26 20:50:54 pki-ovpn-002 systemd[1]: bac.service: Main process exited, c
ode=exited, status=1/FAILURE
Dec 26 20:50:54 pki-ovpn-002 systemd[1]: bac.service: Failed with result 'ex
it-code'.
Dec 26 20:50:54 pki-ovpn-002 systemd[1]: Failed to start Backup service.
```

Если вы забыли подписать импортированный в систему GPG ключ админа (который используется для шифрования архива резервной копии сервисом), то гарантированно возникнет ошибка

```
admin@server-001:/tmp$ sudo journalctl -f -u bac.service
-- Logs begin at Tue 2024-01-02 14:05:14 UTC. --
Jan 02 14:28:17 server-001 sudo[4880]: pam_unix(sudo:session): session closed for user admin
Jan 02 14:28:17 server-001 sudo[4882]:    admin : TTY=unknown ; PWD=/tmp ; USER=root ; COMMAND=/usr/bin/bash -c echo \'/home/admin/.gnupg
--------------#012pub   rsa3072 2024-01-02 [SC] [expires: 2026-01-01]#012      A2F9A9C58B6CA123724BFBD40E1EFCE4A819475A#012uid
erver-001.ru>#012sub   rsa3072 2024-01-02 [E] [expires: 2026-01-01]#012#012pub   rsa3072 2023-09-05 [SC] [expires: 2025-09-04]#012      5
0EEE62C0#012uid           [ unknown] Dokukin Vyacheslav Olegovich <toorrp4@gmail.com>#012sub   rsa3072 2023-09-05 [E] [expires: 2025-09-0
Jan 02 14:28:17 server-001 sudo[4882]: pam_unix(sudo:session): session opened for user root by (uid=0)
Jan 02 14:28:17 server-001 sudo[4882]: pam_unix(sudo:session): session closed for user root
Jan 02 14:28:17 server-001 sudo[4884]:    admin : TTY=unknown ; PWD=/tmp ; USER=root ; COMMAND=/usr/bin/bash -c echo \'[02.01.2024 14:28:
y fail\' >> \'/var/log/bac.log\';
Jan 02 14:28:17 server-001 sudo[4884]: pam_unix(sudo:session): session opened for user root by (uid=0)
Jan 02 14:28:17 server-001 sudo[4884]: pam_unix(sudo:session): session closed for user root
Jan 02 14:28:18 server-001 systemd[1]: bac.service: Main process exited, code=exited, status=1/FAILURE
Jan 02 14:28:18 server-001 systemd[1]: bac.service: Failed with result 'exit-code'.
Jan 02 14:28:18 server-001 systemd[1]: Failed to start Backup service.
```

Для ее исправления, нужно просто подписать этот ключ:

```
gpg --sign-key toorrp4@gmail.com
```

[4] Мониторинг

Система мониторинга организована на с помощью стека GAP (Grafana, Alertmanager, Prometheus). Доступ к Prometheus и Grafana через веб интерфейс открыть по адресам (и доступен только для конкретных IP, через блокировку в iptables). Grafana запущена в docker контейнере. Логин и пароль предоставляет администратор сервера по запросу на email toorrp4@gmail.com

- http://toorrp4.fvds.ru:9090/
- http://toorrp4.fvds.ru:3000/

Alertmanager

Alertmanager сконфигурирован так, что уведомления о срабатывании алертов приходят на email, а правила для них разнесены по файлам с логическими названиями соответствующими названиям экспортеров.

```
toorr2p@toorrp4:~$ lsa /etc/prometheus/
total 64
drwxr-xr-x   5 root root  4096 Jan  5 15:42 .
drwxr-xr-x 122 root root 12288 Jan  4 06:13 ..
drwxr-xr-x   2 root root  4096 Sep 24 01:36 alertmanager_templates
-rw-r--r--   1 root root  3702 Oct 13 23:14 alertmanager.yml
-rw-r--r--   1 root root  1561 Oct 10 19:01 alerts_nginx_rules.yml
-rw-r--r--   1 root root   475 Oct 21 19:34 alerts_node_cpu_rules.yml
-rw-r--r--   1 root root  1845 Oct 10 20:08 alerts_node_disk_rules.yml
-rw-r--r--   1 root root  1276 Dec 30 23:56 alerts_node_memory_rules.yml
-rw-r--r--   1 root root  1053 Oct 10 20:06 alerts_node_network_rules.yml
-rw-r--r--   1 root root   327 Oct 13 22:46 alerts_node_rules.yml
drwxr-xr-x   2 root root  4096 Sep 24 01:36 console_libraries
drwxr-xr-x   2 root root  4096 Sep 24 01:36 consoles
-rw-r--r--   1 root root   574 Oct 25 23:16 prometheus-nginxlog-exporter.hcl
-rw-r--r--   1 root root  2647 Jan  5 15:42 prometheus.yml
```

В нашем случае это файл

- **/etc/prometheus/alerts_node_rules.yml** - содержатся правила для алертов "prometheus-node-exporter" (который устанавливается на сервер PKI при установке пакета ovpn)
  Здесь основными правилами критических алертов являются следующие:

OpenVpnServiceStateFailed (severity: critical) ›

```
node_systemd_unit_state{name="openvpn-server@server.service",state="failed",type="notify"}
  == 1
```

OpenVpnServiceStateFailed (0 active)

```
alert: OpenVpnServiceStateFailed
expr: node_systemd_unit_state{name="openvpn-server@server.service",state="failed",type="notify"}
  == 1
for: 1m
labels:
  severity: critical
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for
    more than 1 minutes.'
  summary: OpenVPN service state FAILED (instance {{ $labels.instance }}) !!!
```

Критический алерт, при его срабатывании следует выяснить причины из-за чего сервис openvpn-server@server.service "упал", смотреть можно с системном журнале командой, попробовать перезапустить сервис

```
sudo journalctl -f
```

OpenVpnServiceStateInactive (severity: critical) ›

```
node_systemd_unit_state{name="openvpn-server@server.service",state="inactive",type="notify"}
  == 1
```

OpenVpnServiceStateInactive (0 active)

```
alert: OpenVpnServiceStateInactive
expr: node_systemd_unit_state{name="openvpn-server@server.service",state="inactive",type="notify"}
  == 1
for: 1m
labels:
  severity: critical
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for
    more than 1 minutes.'
  summary: OpenVPN service state INACTIVE (instance {{ $labels.instance }}) !!!
```

Критический алерт, при его срабатывании следует выяснить причины из-за чего сервис openvpn-server@server.service "упал", смотреть можно с системном журнале командой, попробовать перезапустить сервис

```
sudo journalctl -f
```

InstanceDown (severity: critical) ›

```
up == 0
```

InstanceDown (0 active)

```
alert: InstanceDown
expr: up == 0
for: 1m
labels:
  severity: critical
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for
    more than 1 minutes.'
  summary: Host down (instance {{ $labels.instance }}) !!!
```

Критический алерт, при его срабатывании следует выяснить причины из-за чего сервер упал (логи), и перезагрузить сервер

HostHighCpuLoad (severity: critical) ›

```
(sum by (instance) (avg by (mode, instance) (rate(node_cpu_seconds_total{mode!="idle"}[2m]))) > 0.8) *
on(instance) group_left (nodename) node_uname_info{nodename=~".+"}
```

Критический алерт, при его срабатывании следует выяснить какое ПО утилизирует CPU

HostOutOfMemory (severity: critical) ›

```
(node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10) * on(instance) group_left
(nodename) node_uname_info{nodename=~".+"}
```

Критический алерт, следует выяснить какой процесс утилизирует больше всего памяти, с помощью утилит atop и top или htop, и причины из-за чего процесс "вышел из под контроля"

Остальные алерты не являются критическими но позволяют предотвратить возникновение критических:

HostSwapIsFillingUp (severity: warning) ›

```
((1 - (node_memory_SwapFree_bytes / node_memory_SwapTotal_bytes)) * 100 > 90) * on(instance)
group_left (nodename) node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при утилизации SWAP более чем на 90%, нужно проверить какой процесс больше всех потребляет RAM и SWAP и увеличить размер файла подкачики

HostOomKillDetected (severity: warning) ›

```
(increase(node_vmstat_oom_kill[1m]) > 0) * on(instance) group_left (nodename)
node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при обнаружении запуска системного механизма, который убивает самые прожорливые процессы в порядке приоритета процесса по возрастанию.

HostOutOfDiskSpace (severity: warning) ›

```
((node_filesystem_avail_bytes * 100) / node_filesystem_size_bytes < 10 and ON (instance, device,
mountpoint) node_filesystem_readonly == 0) * on(instance) group_left (nodename)
node_uname_info{nodename=~".+"}
```

Алерт срабатывающий, при заполнении системного раздела диска более чем на 90% (если осталось менее 10%). Следует выяснить командой du -hsx самые тяжеловесные элементы, выгрузить их в облако через s3cmd и удалить если они не нужны.

HostUnusualDiskReadLatency (severity: warning) ›

```
(rate(node_disk_read_time_seconds_total[1m]) / rate(node_disk_reads_completed_total[1m]) > 0.1 and
rate(node_disk_reads_completed_total[1m]) > 0) * on(instance) group_left (nodename)
node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при задержке чтения с диска, следует выяснить причины (у техподдержки облачного VPS например)

HostUnusualDiskWriteLatency (severity: warning) ›

```
(rate(node_disk_write_time_seconds_total[1m]) / rate(node_disk_writes_completed_total[1m]) > 0.1 and
rate(node_disk_writes_completed_total[1m]) > 0) * on(instance) group_left (nodename)
node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при задержке записи на диск, следует выяснить причины (у техподдержки облачного VPS например)

HostUnusualNetworkThroughputOut (severity: warning) ›

```
(sum by (instance) (rate(node_network_transmit_bytes_total[2m])) / 1024 / 1024 > 100) * on(instance)
group_left (nodename) node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при слишком большом объеме данных проходящих через сетевой интерфейс сервере, возможно сервер скомпрометирован и используется для DDos атак!

HostUnusualNetworkThroughputIn (severity: warning) ›

```
(sum by (instance) (rate(node_network_receive_bytes_total[2m])) / 1024 / 1024 > 100) * on(instance)
group_left (nodename) node_uname_info{nodename=~".+"}
```

Алерт срабатывающий при слишком большом объеме данных проходящих на сетевой интерфейс сервера. Возможно используется атака DDos на сервер!

Основной файл конфигурации Alertmanager это файл

```
/etc/prometheus/alertmanager.yml
```

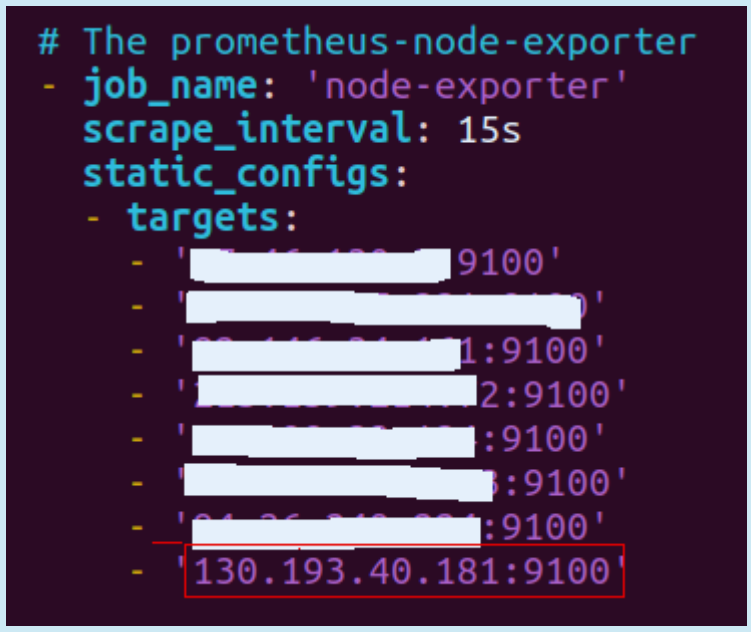
В нем содержатся конфиги, в частности email на который будут приходить уведомления об алертах.

Для мониторинга сервера PKI с OpenVPN, используется 2 экспортера

prometheus-node-exporter

необходимо добавить IP сервера в файл конфигурации Prometheus

```
sudo vim /etc/prometheus/prometheus.yml
```



prometheus-openvpn-exporter

ВАЖНО: перед запуском этого экспортера в контейнере, сохраните правила iptables, т.к они могут быть затерты правилами для docker !!!

Prometheus OpenVPN exporter
Этот экспортер не добавляется при установке DEB пакетов pki, ovpn, bac. Его нужно запустить в Docker контейнере, после завершения всех прочих настроек на сервере PKI
Для начала нужно установить пакет Docker по инструкции Установка Docker на Ubuntu 20.04

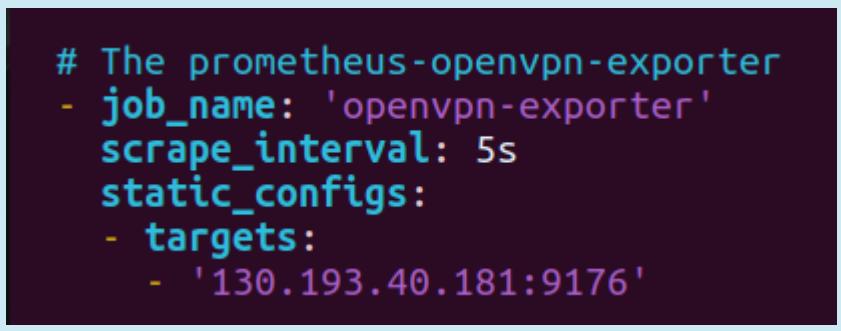
Затем нужно запустить контейнер с экспортером командой

```
sudo docker run -d -p 9176:9176 -v /var/log/openvpn:/etc/openvpn_exporter/server.status kumina/openvpn-exporter -openvpn.status_paths /etc/openvpn_exporter/server.status
```

И внести в конфигурационный файл prometheus новую "джабу"

```
sudo vim /etc/prometheus/prometheus.yml
```

в таком виде



ВАЖНО: проверить что алерт срабатывает, для этого нужно остановить сервис

```
sudo systemctl stop openvpn-server@server.service
```

и дождаться что бы алерт "зажегся", и на мейл пришло уведомление

/etc/prometheus/alerts_node_rules.yml > node-exporter [all]

OpenVpnServiceStateInactive (1 active)

InstanceDown (0 active)

OpenVpnServiceStateFailed (0 active)

/etc/prometheus/alerts_node_rules.yml > node-exporter [all]

OpenVpnServiceStateInactive (1 active)

InstanceDown (0 active)

OpenVpnServiceStateFailed (0 active)

1 alert for alertname=OpenVpnServiceStateInactive

View In AlertManager

[1] Firing

Labels
alertname = OpenVpnServiceStateInactive
instance = 130.193.40.181:9100
job = node-exporter
monitor = @toorr2p
name = openvpn-server@server.service
severity = critical
state = inactive
type = notify
Annotations
description = 130.193.40.181:9100 of job node-exporter has been down for more than 1 minutes.
summary = OpenVPN service state INACTIVE (instance 130.193.40.181:9100) !!!
Source

И не забыть снова включить сервис!

```
sudo systemctl start openvpn-server@server.service
```

Метрики которые позволят построить графики и получить статистику

```
process_resident_memory_bytes{instance="130.193.40.181:9176",job="openvpn-exporter"}
```

```
process_virtual_memory_bytes{instance="130.193.40.181:9176",job="openvpn-exporter"}
```

```
process_cpu_seconds_total{instance="130.193.40.181:9176",job="openvpn-exporter"}
```

После внесения изменений в конфигурацию Prometheus или Alertmanager, выполните перезагрузку соотв. сервиса prometheus

```
sudo systemctl restart prometheus
```

alertmanager

```
sudo systemctl restart prometheus-alertmanager.service
```

Подробные инструкции по развертыванию GAP

- Документация по установке Prometheus и Alertmanager описаны в файле prometeus.pdf
- Документация по установке Grafana описаны в файле grafana.pdf
- Описание алертов находится в файле описания инфраструктуры infrastructure_doc.pdf

Ошибки при работе с Prometheus и экспортерами

При добавлении новой "джабы" в конфиг prometheus

```
/etc/prometheus/prometheus.yml
```

Могут возникнуть ошибки, которые не видны при проверке статуса сервиса
Что бы получить детальный вывод отладки сервиса, сначала стопим его (если он запущен как сервис)

```
sudo systemctl stop prometheus
```

Смотрим путь к исполняемому файлу prometheus

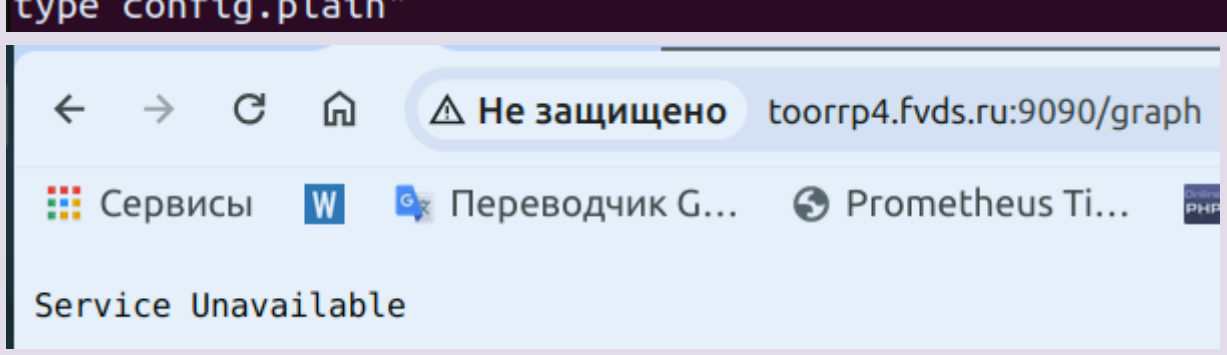
```
which prometheus
```

А затем запускаем его с аргументом уровня логирования debug

```
sudo /usr/bin/prometheus --log.level=debug
```

Где можно увидеть ошибку например такого плана (при этом systemctl status prometheus выводит active (running) якобы все ок, но при этом вебморда prometheus не доступна!)

```
level=error ts=2024-01-04T16:00:42.553Z caller=main.go:725 err="error loading co
nfig from \"/etc/prometheus/prometheus.yml\": couldn't load configuration (--con
fig.file=\"/etc/prometheus/prometheus.yml\"): parsing YAML file /etc/prometheus/
prometheus.yml: yaml: unmarshal errors:\n  line 35: field job-name not found in
type config.plain"
```



Проверить корректность конфига

```
promtool check config /etc/prometheus/prometheus.yml
```

Взаимодействие с клиентами VPN сервера

[1] Выпуск сертификата клиента

Получаем файл запроса на подпись публичного ключа клиента (например client-1.req - запрос на выпуск сертификата), а также публичный GPG ключ клиента (например архивом по email)

Для удобства разделения файлов клиентов, создаем директорию (если ее нет), например по имени файла клиентского запроса, которое можно посмотреть в полученном архиве у файла с расширением .req ($ tar -tf client-1_req.tar.gz) в примере будет использоваться "client-1.req"

```
mkdir -p ~/clients/client-1
```

Распаковываем архив в созданную клиентскую директорию

```
tar -zxvf /tmp/client-1_req.tar.gz -C ~/clients/client-1
```

Копируем запрос на подпись в директорию easy-rsa

```
cp ~/clients/client-1/client-1.req /opt/easy-rsa/pki/reqs/
```

Переходим в директорию

```
cd /opt/easy-rsa
```

Подписываем запрос (исп. тип запроса "client" и имя нашего клиента "client-1")

```
./easyrsa sign-req client client-1
```

Будет запрошена Passphrase к приватному ключу сервера PKI, вводим ее.
После чего запрос подпишется приватным ключом сервера и будет создан сертификат клиента. О чем будет сообщено программой, и будет выведен путь к файлу сертификата

```
Certificate created at: /opt/easy-rsa/pki/issued/client-1.crt
```

```
toorr2p@pki-ovpn-001:~$ cd /opt/easy-rsa/
toorr2p@pki-ovpn-001:/opt/easy-rsa$ ./easyrsa sign-req client client-1

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1f  31 Mar 2020


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a client certificate for 1080 days:

subject=
    commonName                = client-1


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /opt/easy-rsa/pki/safessl-easyrsa.cnf
Enter pass phrase for /opt/easy-rsa/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'client-1'
Certificate is to be certified until Dec  2 21:36:57 2026 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /opt/easy-rsa/pki/issued/client-1.crt
```

Этот сертификат нужно передать клиенту, с помощь него, он сможет сгенерировать скриптом client_gen_ovpn.sh, конфигурационный файл .ovpn для своего openvpn клиента и подключится к серверу PKI(VPN)

Также клиенту нужно передать сертификата клиента (client-1.crt), публичный ключ сервера OpenVPN (ta.key) и корневой сертификат удостоверяющего центра (ca.crt), копируем их в клиентскую директорию

```
cp /opt/easy-rsa/pki/issued/client-1.crt /opt/easy-rsa/pki/ta.key /opt/easy-rsa/pki/ca.crt
~/clients/client-1/
```

В итоге в клиентской директории вы должны получить такой минимальный набор файлов

- **client-1.crt** - ваш сертификат
- **ta.key** - открытый публичный ключ сервера OpenVPN
- **ca.crt** - корневой сертификат сервера PKI (удостоверяющего цена)

Обязательно проверьте это командой

```
ls -la ~/clients/client-1/
```

```
toorr2p@pki-ovpn-001:~/clients/client-1$ lsa
total 28
drwxrwxr-x 2 toorr2p toorr2p 4096 Dec 18 21:39 .
drwxrwxr-x 3 toorr2p toorr2p 4096 Dec 18 21:32 ..
-rw------- 1 toorr2p toorr2p  749 Dec 18 21:39 ca.crt
-rw------- 1 toorr2p toorr2p 2791 Dec 18 21:37 client-1.crt
-rw------- 1 toorr2p toorr2p  440 Dec 18 21:12 client-1.req
-rw-rw-r-- 1 toorr2p toorr2p 2448 Dec 18 21:12 gpgpub.key
-rw------- 1 toorr2p toorr2p  636 Dec 18 21:39 ta.key
```

ⓘ [2] Импорт публичного GPG ключа клиента ›

Находясь в клиентской директории, нужно импортировать публичный ключ клиента, из файла с расширением .key, командой

```
gpg --import ~/clients/client-1/gpgpub.key
```

Просмотреть и удостоверится о том что данный ключ принадлежит клиенту можно командой (которая выведет отпечаток ключа)

```
gpg --fingerprint client1@gmail.com
```

пример

```
toorr2p@pki-ovpn-001:~/clients/client-1$ gpg --import gpgpub.key
gpg: directory '/home/toorr2p/.gnupg' created
gpg: keybox '/home/toorr2p/.gnupg/pubring.kbx' created
gpg: /home/toorr2p/.gnupg/trustdb.gpg: trustdb created
gpg: key 0D29877BCA9EF39E: public key "Client1 <client1@gmail.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1
toorr2p@pki-ovpn-001:~/clients/client-1$ gpg --fingerprint client1@gmail.com
pub   rsa3072 2023-12-18 [SC] [expires: 2025-12-17]
      6320 6BF6 4DF5 9ED2 E839  D18F 0D29 877B CA9E F39E
uid           [ unknown] Client1 <client1@gmail.com>
sub   rsa3072 2023-12-18 [E] [expires: 2025-12-17]
```

как видно из команд email client1@gmail.com является указателем на ключ в системе после импорта

ⓘ [3] Сборка архива и его шифрование GPG ›

После того как в клиентской директории содержится все необходимые файлы, нужно запаковать их в архив

Переходим в клиентскую директорию

```
cd ~/clients/client-1
```

Пакуем файлы в архив

```
tar -zcvf client-1_sig.tar.gz ./
```

```
toorr2p@pki-ovpn-001:~/clients/client-1$ tar -zcvf client-1_sig.tar.gz ./
./
./gpgpub.key
./ta.key
./ca.crt
./client-1.crt
./client-1.req
tar: .: file changed as we read it
toorr2p@pki-ovpn-001:~/clients/client-1$ lsa
total 36
drwxrwxr-x 2 toorr2p toorr2p 4096 Dec 18 21:52 .
drwxrwxr-x 3 toorr2p toorr2p 4096 Dec 18 21:32 ..
-rw------- 1 toorr2p toorr2p  749 Dec 18 21:39 ca.crt
-rw------- 1 toorr2p toorr2p 2791 Dec 18 21:37 client-1.crt
-rw------- 1 toorr2p toorr2p  440 Dec 18 21:12 client-1.req
-rw-rw-r-- 1 toorr2p toorr2p 4869 Dec 18 21:52 client-1_sig.tar.gz
-rw-rw-r-- 1 toorr2p toorr2p 2448 Dec 18 21:12 gpgpub.key
-rw------- 1 toorr2p toorr2p  636 Dec 18 21:39 ta.key
```

После чего этот архив нужно зашифровать используя команду

```
gpg --encrypt --sign --armor -r client1@gmail.com client-1_sig.tar.gz
```

Где:

- **-r** - параметр указывающий на получателя (зашифрованного сообщения, файла)
- client1@gmail.com - указатель на ключ клиента, который сможет прочесть-расшифровать файл или сообщение
- client-1_sig.tar.gz - файл который шифруем

Шифруем файл

```
toorr2p@pki-ovpn-001:~/clients/client-1$ lsa
total 36
drwxrwxr-x 2 toorr2p toorr2p 4096 Dec 18 21:52 .
drwxrwxr-x 3 toorr2p toorr2p 4096 Dec 18 21:32 ..
-rw------- 1 toorr2p toorr2p  749 Dec 18 21:39 ca.crt
-rw------- 1 toorr2p toorr2p 2791 Dec 18 21:37 client-1.crt
-rw------- 1 toorr2p toorr2p  440 Dec 18 21:12 client-1.req
-rw-rw-r-- 1 toorr2p toorr2p 4869 Dec 18 21:52 client-1_sig.tar.gz
-rw-rw-r-- 1 toorr2p toorr2p 2448 Dec 18 21:12 gpgpub.key
-rw------- 1 toorr2p toorr2p  636 Dec 18 21:39 ta.key
toorr2p@pki-ovpn-001:~/clients/client-1$ gpg --encrypt --sign --armor -r client1@gmail.com client-1_sig.tar.gz
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2025-12-17
gpg: D51FC8EB90E9BB6A: There is no assurance this key belongs to the named user

sub  rsa3072/D51FC8EB90E9BB6A 2023-12-18 Client1 <client1@gmail.com>
 Primary key fingerprint: 6320 6BF6 4DF5 9ED2 E839  D18F 0D29 877B CA9E F39E
      Subkey fingerprint: 2264 3637 9536 BD18 7982  6B76 D51F C8EB 90E9 BB6A

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
```

И на выходе получаем его зашифрованную копию client-1_sig.tar.gz.asc которую можно отправить клиенту

```
toorr2p@pki-ovpn-001:~/clients/client-1$ lsa
total 44
drwxrwxr-x 2 toorr2p toorr2p 4096 Dec 18 21:59 .
drwxrwxr-x 3 toorr2p toorr2p 4096 Dec 18 21:32 ..
-rw------- 1 toorr2p toorr2p  749 Dec 18 21:39 ca.crt
-rw------- 1 toorr2p toorr2p 2791 Dec 18 21:37 client-1.crt
-rw------- 1 toorr2p toorr2p  440 Dec 18 21:12 client-1.req
-rw-rw-r-- 1 toorr2p toorr2p 4869 Dec 18 21:52 client-1_sig.tar.gz
-rw-rw-r-- 1 toorr2p toorr2p 7926 Dec 18 21:59 client-1_sig.tar.gz.asc
-rw-rw-r-- 1 toorr2p toorr2p 2448 Dec 18 21:12 gpgpub.key
-rw------- 1 toorr2p toorr2p  636 Dec 18 21:39 ta.key
```

не шифрованный архив лучше сразу удалить

```
rm -f client-1_sig.tar.gz
```

Полученный файл client-1_sig.tar.gz.asc, клиент сможет расшифровать своим приватным ключом, и выполнить все остальные действия по генерации конфигурационного файла .ovpn с помощью клиентской инструкции client_doc.pdf