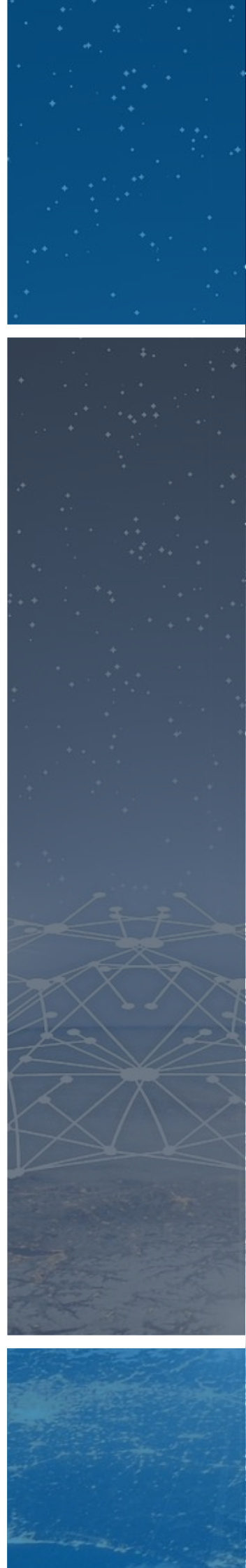


# РАЗВЕРТЫВАНИЕ СТЕНДА НБТ БЕЗ ВИРТУАЛИЗАЦИИ

ООО «НОРБИТ»  
ИНН/КПП: 7702314674/771301001



Готовим операционную систему для удобного использования. Данный раздел повторяем для каждой машины, входящей в стенд:

Проверяем состояние apparmor

**apparmor\_status**

Если видим такой вывод:

```
root@nbt-ub-consul-01:/var/log# apparmor_status
apparmor module is loaded.
28 profiles are loaded.
28 profiles are in enforce mode.
```

То отключаем apparmor:

```
sudo systemctl stop apparmor
sudo systemctl disable apparmor
```

Доставляем необходимые пакеты:

```
sudo apt install net-tools mc wget -y
```

Выставляем временную зону:

```
sudo timedatectl set-timezone Europe/Moscow
```

Обновляем систему:

```
sudo apt upgrade -y
```

## CONSUL

Бинарник консула скачивается вручную через vpn. Распаковывается и помещается на сервер по пути PATH. В нашем примере это:

```
/usr/bin
```

Задаются права на исполнение для пользователя, группы и остальных:

```
sudo chmod ugo+x /usr/bin/consul
```

**Проверяем запуская в консоли consul. Вывод должен быть вот таким:**

```
pasha@nbt-ub-consul-01:/usr/bin$ consul
Usage: consul [--version] [--help] <command> [<args>]

Available commands are:
  acl           Interact with Consul's ACLs
  agent        Runs a Consul agent
  catalog      Interact with the catalog
  config       Interact with Consul's Centralized Configurations
  connect      Interact with Consul Connect
  debug        Records a debugging archive for operators
  event        Fire a new event
  exec         Executes a command on Consul nodes
  force-leave  Forces a member of the cluster to enter the "left" state
  info         Provides debugging information for operators.
  intention    Interact with Connect service intentions
  join         Tell Consul agent to join cluster
  keygen       Generates a new encryption key
  keyring      Manages gossip layer encryption keys
  kv           Interact with the key-value store
  leave        Gracefully leaves the Consul cluster and shuts down
  lock         Execute a command holding a lock
  login        Login to Consul using an auth method
  logout       Destroy a Consul token created with login
  maint        Controls node or service maintenance mode
  members      Lists the members of a Consul cluster
  monitor      Stream logs from a Consul agent
  operator     Provides cluster-level tools for Consul operators
  reload       Triggers the agent to reload configuration files
  rtt         Estimates network round trip time between nodes
  services     Interact with services
  snapshot     Saves, restores and inspects snapshots of Consul server state
  tls          Builtin helpers for creating CAs and certificates
  validate     Validate config files/directories
  version      Prints the Consul version
  watch        Watch for changes in Consul
```

Добавляем consul как службу в systemd.

```
cd /usr/lib/systemd/system
```

Создаём файл consul.service с описанием службы:

```
sudo mcedit consul.service
```

Наполняем его следующим содержимым:

```
/usr/lib/systemd/system/consul.service  [-M--]  0 L:[ 1+2
[Unit]
Description="HashiCorp Consul - A service mesh solution"
Documentation=https://www.consul.io/
Requires=network-online.target
After=network-online.target
ConditionFileNotEmpty=/etc/consul.d/consul.hcl

[Service]
#EnvironmentFile=/etc/consul.d/consul.env
User=root
Group=root
ExecStart=/usr/bin/consul agent -config-dir=/etc/consul.d/
ExecReload=/bin/kill --signal HUP $MAINPID
KillMode=process
KillSignal=SIGTERM
Restart=on-failure
LimitNOFILE=65536

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

[Unit]

Description="HashiCorp Consul - A service mesh solution"

Documentation=https://www.consul.io/

Requires=network-online.target

After=network-online.target

ConditionFileNotEmpty=/etc/consul.d/consul.hcl

```
[Service]
#EnvironmentFile=/etc/consul.d/consul.env
User=root
Group=root
ExecStart=/usr/bin/consul agent -config-dir=/etc/consul.d/
ExecReload=/bin/kill --signal HUP $MAINPID
KillMode=process
KillSignal=SIGTERM
Restart=on-failure
LimitNOFILE=65536
```

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

```
cd /etc/systemd/system/multi-user.target.wants
```

**Создаём симлинк для этого файла:**

```
sudo ln -s /usr/lib/systemd/system/consul.service /etc/systemd/system/multi-user.target.wants/consul.service
```

**Создаём каталоги и конфигурационные файлы для запуска consul:**

```
sudo mkdir /etc/consul.d /opt/consul
sudo mcedit /etc/consul.d/consul.hcl
```

Наполняем его содержимым, подставляя свои значения:

```
log_level = "INFO"
datacenter = "yourDataCenterName"
data_dir = "/opt/consul"
client_addr = "0.0.0.0"
bind_addr = "0.0.0.0"
ui = true
server = true
```

```
bootstrap = true
telemetry {
  prometheus_retention_time="20s"
}
node_name = "yourNodeName"
advertise_addr = "yourIP"
http_config {
  response_headers {
    Access-Control-Allow-Origin = "*"
  }
}

addresses {
  dns="0.0.0.0"
  http="0.0.0.0"
}
ports {
  dns=8600
}
recursors = ["DNS1ip","DNS2ip"]
start_join = ["yourIP"]
```

В нашем примере будет выглядеть так:

```
/etc/consul.d/consul.hcl
log_level = "INFO"
datacenter = "dpr"

data_dir = "/opt/consul"
client_addr = "0.0.0.0"
bind_addr = "0.0.0.0"

ui = true

server = true
bootstrap = true

telemetry {
  prometheus_retention_time = "20s"
}

node_name = "consulseparated"
advertise_addr = "172.30.62.12"

http_config {
  response_headers {
    Access-Control-Allow-Origin = "*"
  }
}

addresses {
  dns = "0.0.0.0"
  http = "0.0.0.0"
}

ports {
  dns = 8600
}

recursors = ["172.30.2.2", "172.30.5.2"]
start_join = ["172.30.62.12"]
```

**Перезапускаем демона systemd для того, чтобы он перечитал созданную конфигурацию:**

```
sudo systemctl daemon-reload
```

**Отключаем службу systemd-resolved (её заменит consul):**

```
sudo systemctl stop systemd-resolved  
sudo systemctl disable systemd-resolved
```

**Запускаем службу consul и добавляем в автозапуск:**

```
sudo systemctl start consul  
sudo systemctl enable consul
```

**Для регистрации сервиса в Consul ВРУЧНУЮ в папке /etc/consul.d создаём json-файл с именем сервиса и следующим содержимым, подставив необходимые значения:**

```
{ "service": { "name": "MyService", "tags": [ "MyServiceTags" ], "address":  
"ServiceIP", "port": ServicePort }}
```

**В нашем примере создадим службу PostgreSQL. Содержимое файла postgres.json в этом случае:**

```
{ "service": { "name": "postgres", "tags": [ "sep" ], "address": "172.30.62.10",  
"port": 5432 }}
```

**Для применения новой конфигурации используем:**

```
consul reload
```

**Проверить, что сервис появился, можно зайдя на Web-интерфейс сервера Consul:**

**Добавляем корневой сертификат в доверенные:**

```
sudo mkdir /usr/local/share/ca-certificates/nbt  
sudo cp /tmp/root.crt /usr/local/share/ca-certificates/nbt/root.crt  
sudo update-ca-certificates
```



The screenshot shows a web browser window with the title "Services - Consul". The address bar displays "172.30.62.11:8500/ui/dpr/services". The browser's address bar also shows "Не защищено" (Not secure) and "172.30.62.11:8500/ui/dpr/services". Below the address bar, there are icons for Gmail, YouTube, Перевести (Translate), and Work.

The Consul UI header is dark with a menu icon, the Consul logo, and the text "dpr". A sidebar on the left lists navigation options: Services, Nodes, Key/Value, Intentions, ACCESS CONTROLS (with a red dot), Tokens, and Policies.

The main content area is titled "Services 2 total". It features a search bar with the text "Search" and a "Search Across" dropdown menu. Below the search bar, there are two service entries:

- consul**: 1 instance
- postgres**: 1 instance, with a tag icon and the label "sep"

## POSTGRESQL

### Устанавливаем PostgrSQL:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -  
sudo apt-get update
```

Перед выполнением следующей команды необходимо определиться с нужной версией Postgre. По умолчанию будет установлена последняя (14-я на текущий момент):

```
sudo apt-get -y install postgresql
```

### Если необходима иная версия, то в команде указываем в явном виде, например:

```
sudo apt-get -y install postgresql-12
```

### Устанавливаем разрешения на доступ к нашему серверу в файле pg\_hba.conf:

В нашем примере этот файл находится в /etc/postgresql/14/main

### Добавляем в конце файла:

```
host all postgres all md5
```

### Далее, редактируем конфигурационный файл postgresql.conf:

Находим параметр listen\_addresses, раскомментируем и присваиваем ему значение '\*'. Это даст возможность обращаться к кластеру извне сервера:

```
#-----  
# CONNECTIONS AND AUTHENTICATION  
#-----  
  
# - Connection Settings -  
  
listen_addresses = '*'<<-----># what IP address(es) to listen on;  
<-----><-----><-----><-----><-----># comma-separated list of addresses;  
<-----><-----><-----><-----><-----># defaults to 'localhost'; use '*' for all  
<-----><-----><-----><-----><-----># (change requires restart)
```

### Перезапускаем службу для применения изменений:

```
systemctl stop postgresql@14-main
```

```
systemctl start postgresql@14-main
```

**Меняем пароль пользователя postgres:**

```
sudo -u postgres psql
```

```
ALTER USER postgres PASSWORD 'MyPass';
```

Где MyPass – пароль, который мы задаём.

**Проверяем доступность, подсоединившись извне, например, через dbeaver.**

**Регистрируем сервер СУБД в DNS (создаём запись А-типа).**

## CAMUNDA

**Устанавливаем пререквизиты:**

### Java

Качаем с <https://jdk.java.net/archive/> нужную версию. В нашем примере будет 15-я. Текущие поддерживаемые версии Java можно посмотреть на официальном сайте Camunda

[https://download.java.net/java/GA/jdk15.0.2/0d1cfde4252546c6931946de8db48ee2/7/GPL/openjdk-15.0.2\\_linux-x64\\_bin.tar.gz](https://download.java.net/java/GA/jdk15.0.2/0d1cfde4252546c6931946de8db48ee2/7/GPL/openjdk-15.0.2_linux-x64_bin.tar.gz)

**Копируем скачанный тарбол в папку /tmp на сервере.**

**Создаём каталог для хранения развёрнутых версий Java и переходим в него:**

```
sudo mkdir -p /usr/java/openjdk && cd /usr/java/openjdk
```

**Копируем тарбол в текущую папку и распаковываем:**

```
sudo cp /tmp/openjdk-15.0.2_linux-x64_bin.tar.gz openjdk-15.0.2_linux-x64_bin.tar.gz
```

```
sudo tar -xzf openjdk-15.0.2_linux-x64_bin.tar.gz
```

**Задаём переменные окружения:**

```
sudo vi /etc/profile
```

**Добавляем в конец файла:**

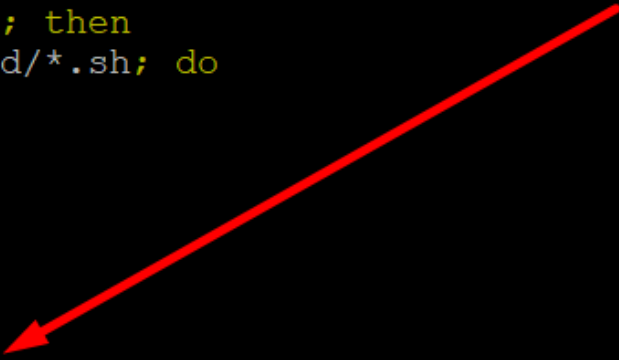
```
# OpenJDK 15
JAVA_HOME=/usr/java/openjdk/jdk-15.0.2
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
export JAVA_HOME
export PATH
```

```
# /etc/profile: system-wide .profile file for the Bourne
# and Bourne compatible shells (bash(1), ksh(1), ash(1),

if [ "${PS1-}" ]; then
  if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

# OpenJDK 15
JAVA_HOME=/usr/java/openjdk/jdk-15
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
export JAVA_HOME
export PATH
~
```



Сохраняем изменения, выходим.

Регистрируем текущую версию Java в операционной системе.

```
sudo update-alternatives --install "/usr/bin/java" "java" "/usr/java/openjdk/jdk-15.0.2/bin/java" 1
```

```
sudo update-alternatives --install "/usr/bin/javac" "javac" "/usr/java/openjdk/jdk-15.0.2/bin/javac" 1
```

### Проверяем текущую версию Java:

```
java -version
```

Если видим вот такой вывод:

```
pasha@nbt-ub-cames-01:/usr/java/openjdk$ java -version
openjdk version "15.0.2" 2021-01-19
OpenJDK Runtime Environment (build 15.0.2+7-27)
OpenJDK 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

То настроили Java правильно.

## Сама Camunda.

Скачиваем тарбол с сайта и копируем на сервер:

<https://downloads.camunda.cloud/release/camunda-bpm/run/>

Создаём каталог, где будет размещено наше приложение и переходим в него:

```
sudo mkdir -p /apps/camunda && cd /apps/camunda
```

Копируем тарбол в текущую папку и распаковываем:

```
sudo cp /tmp/camunda-bpm-run-7.17.0.tar.gz camunda-bpm-run-7.17.0.tar.gz && sudo tar -xzf camunda-bpm-run-7.17.0.tar.gz
```

Перед запуском Camunda необходимо донстроить:

Поскольку у нас используется PostgreSQL в качестве СУБД – доставляем ява-драйвер:

Скачиваем с сайта <https://jdbc.postgresql.org/download.html>, копируем на сервер.

В нашем примере используется последняя на текущий момент версия 42.3.3

Размещаем в папке распакованной Камунды /configuration/userlib:

```
sudo cp /tmp/postgresql-42.3.3.jar /apps/camunda/configuration/userlib/postgresql-42.3.3.jar
```

Создаём на сервере СУБД базу данных под Camunda и прописываем в конфигурационном файле (/configuration/default.yml) обращение к этой БД. В нашем случае это будет база process-engine.

**Рабочий конфиг в этом случае выглядит так:**

```
camunda.bpm:
```

```
  admin-user:
```

```
    id: demo
```

```
    password: demo
```

```
  run:
```

```
    cors:
```

```
      enabled: true
```

```
      allowed-origins: "*"
```

```
# datasource configuration is required
```

```
spring.datasource:
```

```
  url: jdbc:postgresql://nbt-ub-postgr-01.dpr.norbit.ru:5432/process-engine
```

```
  driver-class-name: org.postgresql.Driver
```

```
  username: postgres
```

```
  password: 12345678
```

```
server.port: 8113
```

```
/apps/camunda/configuration/default.yml
camunda.bpm:
  admin-user:
    id: demo
    password: demo
  run:
    cors:
      enabled: true
      allowed-origins: "*"

# datasource configuration is required
spring.datasource:
  url: jdbc:postgresql://nbt-ub-postgr-01.dpr.norbit.ru:5432/process-engine
  driver-class-name: org.postgresql.Driver
  username: postgres
  password: 12345678
server.port: 8113
```

Где

jdbc:postgresql – тип используемого соединения с БД (postgresql)

nbt-ub-postgr-01.dpr.norbit.ru:5432/process-engine – адрес сервера СУБД, порт и имя используемой базы

org.postgresql.Driver – прямое указание Java-машине, какой драйвер использовать для данного соединения

postgres – имя роли в БД с необходимыми правами доступа

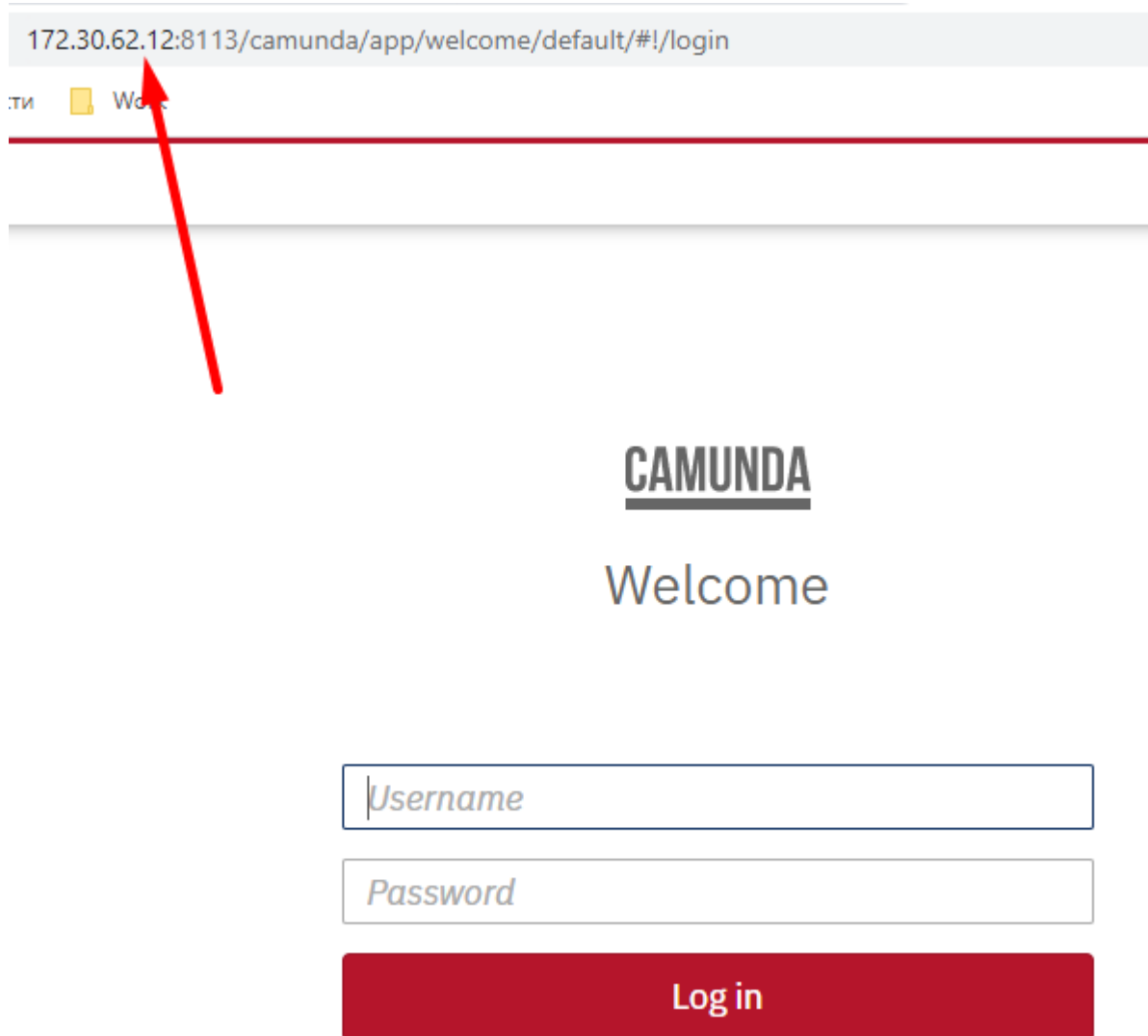
12345678 – пароль роли в БД с необходимыми правами доступа

server.port: 8113 – порт, на котором начнёт вещать Camunda после запуска (должен быть свободен для успешного запуска).

Camunda запускается скриптом start.sh и останавливается скриптом shutdown.sh

Запускаем start.sh и проверяем браузером, что Camunda начала вещать на заданном порту (8113):





172.30.62.12:8113/camunda/app/welcome/default/#!/login

ти Wo

# CAMUNDA

## Welcome

Добавляю запуск Camunda как linux-сервис:

## ELASTICSEARCH

Для стенда важна версия пакета 7.9.2. В текущих реалиях скачиваем руками через VPN и копируем на сервер.

**Переходим в папку со скачанным приложением и устанавливаем его:**

```
cd /tmp && sudo dpkg -i elasticsearch-7.9.2-amd64.deb
```

**Редактируем файл настроек ElasticSearch:**

```
sudo mcedit /etc/elasticsearch/elasticsearch.yml
```

**Стираем всё содержимое и наполняем своим. В нашем примере:**

```
bootstrap.memory_lock: false
cluster.name: dpr-sep-elasticsearch
http.port: 9200
http.cors.allow-origin: '*'
http.cors.enabled: true
http.cors.allow-headers : X-Requested-With,X-Auth-Token,Content-Type,Content-
Length,Authorization
http.cors.allow-credentials: true
network.host: 172.30.62.12
node.data: true
node.ingest: true
node.master: true
node.max_local_storage_nodes: 1
#node.name: ax.elastic.service.consul
path.data: /apps/elasticsearch/data
path.logs: /apps/elasticsearch/log
transport.tcp.port: 9300
xpack.license.self_generated.type: basic
xpack.security.enabled: false
path.repo: ["/backup/elasticsearch"]
discovery.type: single-node
```

```
cluster.max_shards_per_node: 2100
```

```
/etc/elasticsearch/elasticsearch.yml 681
bootstrap.memory_lock: false
cluster.name: dpr-sep-elasticsearch
http.port: 9200
http.cors.allow-origin: '*'
http.cors.enabled: true
http.cors.allow-headers : X-Requested-With,X-Auth-Token,Content-Type,Content-Length,Authorization
http.cors.allow-credentials: true
network.host: 172.30.62.12
node.data: true
node.ingest: true
node.master: true
node.max_local_storage_nodes: 1
#node.name: ax.elastic.service.consul
path.data: /apps/elasticsearch/data
path.logs: /apps/elasticsearch/log
transport.tcp.port: 9300
xpack.license.self_generated.type: basic
xpack.security.enabled: false
path.repo: ["/backup/elasticsearch"]
discovery.type: single-node
cluster.max_shards_per_node: 2100
```

**Создаём папки приложения, которые указали в конфиге и добавляем разрешения для пользователя, под которым запускаем службу Elasticsearch:**

```
mkdir -p /backup/elasticsearch /apps/elasticsearch/data /apps/elasticsearch/log
```

```
sudo chmod -R 777 /apps/elasticsearch
```

**Запускаем службу elasticsearch:**

```
sudo systemctl start elasticsearch
```

**Проверяем, что она запустилась успешно, и что она добавлена в автозагрузку:**

```
sudo systemctl status elasticsearch
```

```
pasha@nbt-ub-comes-01:/tmp$ systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-04-07 13:19:25 UTC; 36min ago
     Docs: https://www.elastic.co
   Main PID: 22392 (java)
    Tasks: 50 (limit: 9373)
   Memory: 2.3G
   CGroup: /system.slice/elasticsearch.service
           └─22392 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60
             └─22612 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller
```

**Если видим статус disabled – добавляем в автозагрузку командой:**

```
sudo systemctl enable elasticsearch
```

**Проверяем доступность ресурса с других машин стенда:**

```
curl ub-cames-01.nbt.dpr.norbit.ru:9200/_cluster/health?pretty
```

## СЕРТИФИКАТЫ

Генерация корневого ключа:

**ВНИМАНИЕ!** Данный ключ используется для подписывания запросов на сертификаты. Хранить в безопасном сухом и тёмном месте!

```
openssl genrsa -des3 -out rootCA.key 4096
```

**Далее создаём и самоподписываем корневой сертификат:**

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.crt
```

**Где:**

rootCA.key – ключ, который мы создали на предыдущем шаге

rootCA.crt – имя корневого сертификата, который мы генерируем

**/\*Создание сертификата (для каждого сервера):**

**Создаём ключ сертификата:**

```
openssl genrsa -out mydomain.com.key 2048
```

**Создаём запрос на подписание (csr):**

```
openssl req -new -key mydomain.com.key -out mydomain.com.csr
```

**Проверяем содержимое csr-файла:**

```
openssl req -in mydomain.com.csr -noout -text
```

\*/

CSR+KEY

```
openssl req -out dpr.norbit.ru.csr -newkey rsa:2048 -nodes -keyout dpr.norbit.ru.key -config san.cnf
```

CSR

```
openssl x509 -req -in dpr.norbit.ru.csr -CA DPRrootCA.crt -CAkey DPRrootCA.key -CAcreateserial -out dpr.norbit.ru.crt -days 500 -sha256 -extfile san.cnf -extensions req_ext
```

```
openssl req -out standsip.csr -newkey rsa:2048 -nodes -keyout standsip.key -subj  
"/C=RU/ST=Moscow/L=Moscow/O=Norbit/OU=Norbit-DPR/CN=consul.service.consul" -  
config san.cnf -extensions req_ext
```

SiteCert from CSR

```
openssl x509 -req -in standsip.csr -CA DPRrootCA.crt -CAkey DPRrootCA.key -  
CAcreateserial -out standsip.crt -days 500 -sha256 -extfile san.cnf -extensions req_ext
```

PFX from CRT and KEY

```
openssl pkcs12 -export -out standsip.pfx -inkey standsip.key -in standsip.crt
```

Содержимое SAN.CNF

```
[ req ]
```

```
default_bits = 2048
```

```
distinguished_name = req_distinguished_name
```

```
req_extensions = req_ext
```

```
[ req_distinguished_name ]
```

```
countryName = Country Name (2 letter code)
```

```
stateOrProvinceName = State or Province Name (full name)
```

```
localityName = Locality Name (eg, city)
```

```
organizationName = Organization Name (eg, company)
```

```
commonName = Common Name (e.g. server FQDN or YOUR name)
```

```
[ req_ext ]
```

```
subjectAltName = @alt_names
```

```
[alt_names]
```

```
DNS.1 = dpr.norbit.ru
```

```
DNS.2 = *.dpr.norbit.ru
```

```
DNS.3 = ub-front-01.nbt.dpr.norbit.ru
```

```
DNS.4 = ub-back-01.nbt.dpr.norbit.ru
```

```
DNS.5 = ub-cames-01.nbt.dpr.norbit.ru
```

```
IP.1 = 172.30.62.10
```

```
IP.2 = 172.30.62.11
```

```
IP.3 = 172.30.62.12
```

```
IP.4 = 172.30.62.13
```

IP.5 = 172.30.62.14

**Добавляем сертификат УЦ в доверенные на каждой машине стенда:**

```
sudo update-ca-certificates
```

**Бэк**

Добавляем ключ подписывания пакета Майкрософт в список доверенных ключей и добавить репозиторий пакетов:

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
```

```
sudo dpkg -i packages-microsoft-prod.deb
```

```
rm packages-microsoft-prod.deb
```

```
sudo apt update
```

**Устанавливаем среду выполнения Dotnet:**

```
sudo apt install -y aspnetcore-runtime-5.0 dotnet-sdk-5.0
```

**Далее распаковываем все артефакты бэка каждый в отдельный каталог в папку /apps.**

Каждое из приложений запускаем как сервис. Для этого создаём файл сервиса и помещаем в каталог /etc/systemd/system. В качестве примера используем приложение audit, распакованное в /apps/audit.

```
sudo vi /etc/systemd/system/audit.service
```

**Образец содержимого файла сервиса audit.service:**

```
[Unit]
```

```
Description=Audit dotnet application
```

[Service]

WorkingDirectory=/apps/audit

# systemd will run this executable to start the service

# if /usr/bin/dotnet doesn't work, use `which dotnet` to find correct dotnet executable path

ExecStart=/bin/bash -c "cd /apps/audit && \$(cat run.cmd)"

# to query logs using journalctl, set a logical name here

SyslogIdentifier=AuditBack

# Use your username to keep things simple.

# If you pick a different user, make sure dotnet and all permissions are set correctly to run the app

# To update permissions, use 'chown yourusername -R /srv/HelloWorld' to take ownership of the folder and files,

# Use 'chmod +x /srv/HelloWorld/HelloWorld' to allow execution of the executable file

User=pasha

# This environment variable is necessary when dotnet isn't loaded for the specified user.

# To figure out this value, run 'env | grep DOTNET\_ROOT' when dotnet has been loaded into your shell.

Environment=DOTNET\_ROOT=/usr/share/dotnet

[Install]

WantedBy=multi-user.target

### Где:

Description – описание нашего сервиса (показывается при запуске)

ExecStart – команда, запускающая приложение

WorkingDirectory – рабочий каталог приложения

SyslogIdentifier – Идентификатор сообщений приложения в syslog

User – пользователь, от которого запускается сервис

Environment – переменная окружения DOTNET\_ROOT, необходимая для корректного запуска dotnet-приложений. Должна указывать на исполняемый файл dotnet.



**Сохраняем, закрываем.**

**Перечитываем конфигурацию:**

```
sudo systemctl daemon-reload
```

**Запускаем наш новый сервис и добавляем в автозагрузку (используем имя созданного файла):**

```
sudo systemctl start audit.service && systemctl enable audit.service
```

**Проверяем, что он запустился успешно:**

```
sudo systemctl status audit.service
```

## **СИДИНГ** (заполнение базы необходимыми настройками)

Выполняется единожды при создании стенда или при обновлении бэка.

**Создать пользователя admin и роли. Для этого в папке с IdentityServer выполнить:**

```
dotnet ./NbtPlatform.Core.IdentityServer.Seed.dll
```

**В папке с компонентом clitools выполняем:**

```
dotnet ./NbtFramework.Cli.SeedData.dll
```

## **TODO** (Описание конфигов приложения)

### **Порты БЭКА**

### BACK Services PORTS

Audit	- 5046
Backgroundworker	- 5084
Boinstance	- 5005
Vpmengine	- 5035
Calendar	- 5071

Clitools	- 5009
Documenttemplate	- 5068
ExternalSystem	- 5061
Generatingtestdata	- 5009
Integrationlog	- 5042
Log	- 5049
Manageapiresource	- 5024
Managecms	- 5087
Managefile	- 5031
Manageuser	- 5006
Notice	- 5052
Platformconfigurator-	5006 (changed to 5007)
Scheduler	- 5086
Temporaryfilestorage-	5037
Uiformsettings	- 5018
Warehouse	- 5083
Webdefaultapigateway-	5001
Widget	- 5073

## ФРОНТ

Знакомим Identity с фронтом (сидинг фронта). Для этого выполняем в папке с IdentityServer:

```
dotnet ./NbtPlatform.Core.IdentityServer.Seed.dll /addwebclient <имя собранного фронта>  
https://<адрес веб-интерфейса фронта>
```