?????????? ? ???????????

- Гайд по установке Kaspersky EDR Expert (on-premise)
- Гайд по развертыванию Cross-DC Kaspersky EDR Expert (on-premise)

???? ?? ????????? Kaspersky EDR Expert (on-premise)

> Инструкция составлена на основе официальной документации KEDR 8.3 и опыта эксплуатации.
> Дата исправления: 10.11.2026

????? 1. ???? ?????????

Параметр	Стандартная конфигурация (несколько узлов)	Демонстрационная конфигурация (один узел)
СУБД PostgreSQL	Устанавливается **вне кластера Kubernetes** на отдельном сервере	Устанавливается на одном хосте с **кластером Kubernetes**
Минимальное количество узлов	1 первичный + 3 рабочих узла + сервер СУБД + 1 узел администратора (опционально)	1 узел (все компоненты на одном устройстве) + 1 узел администратора (опционально)
Назначение	Промышленная эксплуатация	Тестирование, демонстрация, обучение

???? ?????:

- **Узел администратора** - устройство с утилитой KDT (Kaspersky Deployment Toolkit) для развертывания и управления компонентами OSMP.
- **Primary/master/controller/первичный рабочий узел** - узел контроллера, осуществляющий управление кластером k0s.
- **Worker/рабочий узел** - узел кластера k0s с полезной нагрузкой.
- **DB/СУБД** - сервер с СУБД для кластера OSMP.
- **KUMA services/устройство с сервисами KUMA** - устройства с установленными сервисами KUMA: коллектор, коррелятор, хранилище (в случае KEDR входят в состав Kubernetes кластера).
- **Целевые устройства** - устройства, на которых устанавливается OSMP (все вышеперечисленные узлы)

????? 2. ??????????? ?????????? ?????????????

2.1. ???? PostgreSQL

Параметр	Требование

Расположение	Рекомендуется вне кластера Kubernetes (стандартная конфигурация)
Привилегированная учётная запись	Требуется учётная запись с правами суперпользователя для создания баз данных во время развертывания
Поддержка кластеров	Поддерживается синхронная репликация (минимум 3 узла, максимум 15).
Дисковая подсистема	SSD/NVMe рекомендуется

2.2. ??????? ??????????

Требование	Описание
Широковещательный домен	Все целевые устройства кластера Kubernetes **должны находиться в одном широковещательном домене** (одна L2-сеть)
Статические IPv4-адреса	Все узлы кластера и шлюз Kubernetes должны иметь статические IPv4-адреса в одной подсети
Синхронизация времени	Разница во времени между узлами не должна превышать 5 секунд (рекомендуется использовать NTP)
DNS	Должна быть настроена зона для домена smp_domain (например, smp.local) с записями для всех сервисов

2.3. ?????????? ?????????? (???????????)

Компонент	Процессор 3 ГГц	ОЗУ	Дисковая подсистема
Все компоненты на одном узле (демонстрационная конфигурация)	8 ядер	56 Гб	2530 Гб

Для корректного развертывания решения убедитесь, что процессор целевого устройства (компонентов KEDR) поддерживает набор инструкций BMI, AVX и SSE 4.2.

2.4. ??????????? ??????????

Требования к программному обеспечению и поддерживаемым системам и платформам

Операционная система	**OSMP с компонентами KUMA** Поддерживаются следующие 64-разрядные версии операционных систем: • Ubuntu Server 24.04 LTS. • Ubuntu Server 26.04 LTS. • Astra Linux Special Edition РУСБ.10015-01 (очередное обновление 1.8.5).

<table>
<tr>
<td>Платформы виртуализации</td>
<td>
<table>
<tr><th>Название платформы виртуализации</th><th>OSMP</th><th>Sandbox (опционально)</th></tr>
<tr><td>VMware ESXi 7.0.3</td><td>Есть.</td><td>Есть.</td></tr>
<tr><td>Hyper-V для Windows Server 2022</td><td>Есть.</td><td>Нет.</td></tr>
<tr><td>ПК СВ "Брест" 3.3</td><td>Есть.</td><td>Есть.</td></tr>
<tr><td>"РЕД Виртуализация" 7.3</td><td>Есть.</td><td>Есть.</td></tr>
<tr><td>zVirt Node 4.2</td><td>Нет.</td><td>Есть.</td></tr>
<tr><td>zVirt Node 4.5</td><td>Есть.</td><td>Нет.</td></tr>
<tr><td>Nutanix 7.3</td><td>Есть.</td><td>Нет.</td></tr>
<tr><td>Proxmox Virtual Environment 9.1.1</td><td>Есть.</td><td>Нет.</td></tr>
</table>
</td>
</tr>
<tr>
<td>Система управления базами данных (СУБД)</td>
<td>PostgreSQL 17.9 64-разрядная.
PostgreSQL 18.4 64-разрядная.
Postgres Pro 18 Enterprise 64-разрядная (кластер Built-in High Availability).
Corosync/Pacemaker поддерживается для всех перечисленных версий PostgreSQL.</td>
</tr>
</table>

????? 3. ?????????? ?????????

> Отключайте файл подкачки (swap) в продуктовых средах: **sudo swapoff -a**

?????????? ??????????????:

Системные требования предоставлены в справке.

Установка обязательных пакетов на устройстве администратора:

```
sudo apt update
sudo apt install -y python3
```

Установите пакет для Docker версии 25 или выше. Пример:

```
sudo apt install docker.io

sudo usermod -aG docker $USER
newgrp docker
```

Генерация SSH-ключа (без парольной фразы):

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""
```

Скопируйте ключ на все целевые устройства:

```
ssh-copy-id username@<IP_целевого_устройства>

# Проверка подключения без пароля
ssh username@<IP_целевого_устройства> "sudo whoami"
```

> При установке от учётной записи **root** убедитесь, что ключ на целевых устройствах располагается по пути **/root/.ssh/authorized_keys**

??????? ?????????? (OSMP):

Проверка cgroup v2:

```
mount | grep cgroup
# Должно быть: cgroup2 on /sys/fs/cgroup type cgroup2
```

Отключите SELinux (если установлен):

```
# Проверка статуса
sudo getenforce

# Отключение (требуется перезагрузка)
sudo setenforce 0 sudo sed -i 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
```

Отключите МКЦ и ЗПС (Astra Linux):

```
#Отключение МКЦ (мандатного контроля целостности)
sudo astra-mic-control disable

#Отключение ЗПС (замкнутой программной среды)
sudo astra-digsig-control disable
```

```
sudo reboot
```

Настройте proxy (если требуется):

```
# Отредактируйте /etc/environment
sudo nano /etc/environment

# Добавьте:
HTTP_PROXY="http://proxy.example.com:8080"
HTTPS_PROXY="http://proxy.example.com:8080"
NO_PROXY="localhost,127.0.0.1,<IP_адреса_узлов_кластера>"
```

Настройте firewall (если используется) согласно статье.

Для отключения firewall'а выполните:

```
sudo systemctl stop ufw
sudo systemctl disable ufw
```

Установка обязательных пакетов:

```
# Общие пакеты для всех узлов:
sudo apt update
sudo apt install -y sudo nfs-common cron tar open-iscsi python3-cffi-backend python3-
cryptography python3-netaddr python-apt-common systemd cryptsetup-bin cryptsetup netcat-
traditional containerd

# Убедитесь, что на каждом целевом узле установлены пакеты python3-apt и libnfs и curl
```

Настройка беспарольного sudo:

```
# Для пользователя, который будет использоваться KDT:
echo "username ALL=(ALL) NOPASSWD: ALL" | sudo tee -a /etc/sudoers

#Это позволит учетной записи иметь возможность повышать привилегии (sudo) без ввода пароля
```

Настройка синхронизации времени:

```
sudo timedatectl set-ntp true
```

Настройка IP-переадресации (только для первичного узла с UFW):

```
# В файле /etc/default/ufw установите:
DEFAULT_FORWARD_POLICY="ACCEPT"

# Примените изменения:
sudo ufw reload
```

Проверить наличие службы systemd-resolved и хотя бы одного DNS сервера:

```
sudo systemctl status systemd-resolved


sudo resolvectl status
```

???? PostgreSQL:

Установка обязательных пакетов:

```
sudo apt update
sudo apt install -y postgresql
```

Настройка параметров в **postgresql.conf:**

```
sudo nano /etc/postgresql/<ВЕРСИЯ>/main/postgresql.conf

# Переопределите следующие параметры:
listen_addresses = '*'
port = 5432
max_connections = 512
shared_buffers = 8GB                  # 25% от ОЗУ, минимум 3 ГБ
temp_buffers = 24MB
work_mem = 64MB
maintenance_work_mem = 1GB
max_stack_depth = 7MB                 # Для Linux: ulimit -s минус 1 МБ
effective_io_concurrency = 200        # 200 для SSD или 2 для HDD
max_parallel_workers_per_gather = 0
wal_buffers = 64MB
max_wal_size = 4GB
min_wal_size = 1GB
random_page_cost = 1.1                # 1.1 для SSD или 4.0 для HDD
effective_cache_size = 24GB           # 75% от ОЗУ
log_hostname = 1
standard_conforming_strings = on      # Обязательно должно быть 'on'
```

Для упрощения воспользуйтесь одной из команд:

```
#для SSD
VER=$(ls -d /etc/postgresql/[0-9]* 2>/dev/null | sort -V | tail -n 1 | grep -oE '[0-9]+');
RAM_MB=$(free -m | awk '/^Mem:/{print $2}'); SB=$((RAM_MB/4 < 3072 ? 3072 : RAM_MB/4));
EC=$((RAM_MB*3/4)); CONF=/etc/postgresql/$VER/main/postgresql.conf; sudo sed -i -e
"s|^\s*#\?\s*listen_addresses\s*=.*|listen_addresses = '*'|" -e "s|^\s*#\?\s*port\s*=.*|port =
5432|" -e "s|^\s*#\?\s*max_connections\s*=.*|max_connections = 512|" -e
"s|^\s*#\?\s*shared_buffers\s*=.*|shared_buffers = ${SB}MB|" -e
"s|^\s*#\?\s*temp_buffers\s*=.*|temp_buffers = 24MB|" -e "s|^\s*#\?\s*work_mem\s*=.*|work_mem
= 64MB|" -e "s|^\s*#\?\s*maintenance_work_mem\s*=.*|maintenance_work_mem = 1GB|" -e
"s|^\s*#\?\s*max_stack_depth\s*=.*|max_stack_depth = 7MB|" -e
"s|^\s*#\?\s*effective_io_concurrency\s*=.*|effective_io_concurrency = 200|" -e
"s|^\s*#\?\s*max_parallel_workers_per_gather\s*=.*|max_parallel_workers_per_gather = 0|" -e
"s|^\s*#\?\s*wal_buffers\s*=.*|wal_buffers = 64MB|" -e
"s|^\s*#\?\s*max_wal_size\s*=.*|max_wal_size = 4GB|" -e
"s|^\s*#\?\s*min_wal_size\s*=.*|min_wal_size = 1GB|" -e
"s|^\s*#\?\s*random_page_cost\s*=.*|random_page_cost = 1.1|" -e
"s|^\s*#\?\s*effective_cache_size\s*=.*|effective_cache_size = ${EC}MB|" -e
"s|^\s*#\?\s*log_hostname\s*=.*|log_hostname = 1|" -e
"s|^\s*#\?\s*standard_conforming_strings\s*=.*|standard_conforming_strings = on|" $CONF && for
P in "listen_addresses='*'" "port=5432" "max_connections=512" "shared_buffers=${SB}MB"
"temp_buffers=24MB" "work_mem=64MB" "maintenance_work_mem=1GB" "max_stack_depth=7MB"
"effective_io_concurrency=200" "max_parallel_workers_per_gather=0" "wal_buffers=64MB"
"max_wal_size=4GB" "min_wal_size=1GB" "random_page_cost=1.1" "effective_cache_size=${EC}MB"
"log_hostname=1" "standard_conforming_strings=on"; do N=${P%=*}; V=${P#*=}; grep -q
"^\s*#\?\s*$N\s*=" $CONF || echo "$N = $V" | sudo tee -a $CONF > /dev/null; done && sudo
systemctl restart postgresql

# для HDD
VER=$(ls -d /etc/postgresql/[0-9]* 2>/dev/null | sort -V | tail -n 1 | grep -oE '[0-9]+');
RAM_MB=$(free -m | awk '/^Mem:/{print $2}'); SB=$((RAM_MB/4 < 3072 ? 3072 : RAM_MB/4));
EC=$((RAM_MB*3/4)); CONF=/etc/postgresql/$VER/main/postgresql.conf; sudo sed -i -e
"s|^\s*#\?\s*listen_addresses\s*=.*|listen_addresses = '*'|" -e "s|^\s*#\?\s*port\s*=.*|port =
5432|" -e "s|^\s*#\?\s*max_connections\s*=.*|max_connections = 512|" -e
"s|^\s*#\?\s*shared_buffers\s*=.*|shared_buffers = ${SB}MB|" -e
"s|^\s*#\?\s*temp_buffers\s*=.*|temp_buffers = 24MB|" -e "s|^\s*#\?\s*work_mem\s*=.*|work_mem
= 64MB|" -e "s|^\s*#\?\s*maintenance_work_mem\s*=.*|maintenance_work_mem = 1GB|" -e
"s|^\s*#\?\s*max_stack_depth\s*=.*|max_stack_depth = 7MB|" -e
"s|^\s*#\?\s*effective_io_concurrency\s*=.*|effective_io_concurrency = 2|" -e
```

```
"s|^\s*#\?\s*max_parallel_workers_per_gather\s*=.*|max_parallel_workers_per_gather = 0|" -e
"s|^\s*#\?\s*wal_buffers\s*=.*|wal_buffers = 64MB|" -e
"s|^\s*#\?\s*max_wal_size\s*=.*|max_wal_size = 4GB|" -e
"s|^\s*#\?\s*min_wal_size\s*=.*|min_wal_size = 1GB|" -e
"s|^\s*#\?\s*random_page_cost\s*=.*|random_page_cost = 4.0|" -e
"s|^\s*#\?\s*effective_cache_size\s*=.*|effective_cache_size = ${EC}MB|" -e
"s|^\s*#\?\s*log_hostname\s*=.*|log_hostname = 1|" -e
"s|^\s*#\?\s*standard_conforming_strings\s*=.*|standard_conforming_strings = on|" $CONF && for
P in "listen_addresses='*'" "port=5432" "max_connections=512" "shared_buffers=${SB}MB"
"temp_buffers=24MB" "work_mem=64MB" "maintenance_work_mem=1GB" "max_stack_depth=7MB"
"effective_io_concurrency=200" "max_parallel_workers_per_gather=0" "wal_buffers=64MB"
"max_wal_size=4GB" "min_wal_size=1GB" "random_page_cost=1.1" "effective_cache_size=${EC}MB"
"log_hostname=1" "standard_conforming_strings=on"; do N=${P%=*}; V=${P#*=}; grep -q
"^\s*#\?\s*$N\s*=" $CONF || echo "$N = $V" | sudo tee -a $CONF > /dev/null; done && sudo
systemctl restart postgresql
```

Разрешение удаленного подключения к СУБД:

```
sudo nano /etc/postgresql/<ВЕРСИЯ>/main/pg_hba.conf

# В секции 'IPv4 local connections' переопределите значение:
host   all   all   0.0.0.0/0    scram-sha-256
```

Перезапуск службы:

```
sudo systemctl restart postgresql
```

Создание привилегированной учётной записи:

```
# Подключитесь к PostgreSQL:
sudo -u postgres psql

# Создайте учётную запись с правами суперпользователя и базу, например:
CREATE USER <kaspersky_admin> WITH PASSWORD '<StrongPassword123!>' SUPERUSER;
CREATE DATABASE <kaspersky_admin> OWNER <kaspersky_admin>;
```

?????????? ??????????????:

1. Зарезервируйте IP-адрес из той же подсети, что и у серверов Primary/Worker. Адрес должен быть свободен и будет назначен в процессе установки (указывается в файле **param.yaml** в поле **ingress_ip**).

2. Добавьте в DNS-зону вашей организации (предпочтительно создать поддомен) следующие записи:

```
console.<smp_domain> <ingress_ip>
admsrv.<smp_domain> <ingress_ip>
api.<smp_domain> <ingress_ip>
monitoring.<smp_domain> <ingress_ip>
updater.<smp_domain> <ingress_ip>
agentserver.<smp_domain> <ingress_ip>
kuma.<smp_domain> <ingress_ip>
*.kuma.<smp_domain> <ingress_ip>
```

где, **<smp_domain>** - домен или поддомен вашей организации (данный домен должен быть указан в файле параметров (param.yaml) в одноименном поле **smp_domain**), а **<ingress_ip>** - зарезервированный IP из предыдущего пункта.

> Обратите внимание, что для имени **kuma** необходимо создать wildcard (*) на DNS-сервере

????? 4. ????? ?????????

?????????? ????? param.yaml:

> Для корректной работы KDT с конфигурационным файлом добавьте пустую строку в конце файла

Формат файла для стандартной конфигурации с сервисами KUMA внутри кластера

```
schemaType: ParameterSet
schemaVersion: 1.0.1
namespace: ""
name: bootstrap
project: xdr
nodes:
  - desc: cdt-primary1
    type: primary
    host: "<IPv4_первичного_узла>"
    access:
```

```
      ssh:
        user: "<имя_пользователя>" # По умолчанию root
        key: "<путь_к_открытому_ключу>" # По умолчанию /root/.ssh/id_rsa
  - desc: cdt-w1
    type: worker
    host: "<IPv4_рабочего_узла_1>"
    access:
      ssh:
        user: "<имя_пользователя>"
        key: "<путь_к_открытому_ключу>"
  - desc: cdt-w2
    type: worker
    host: "<IPv4_рабочего_узла_2>"
    access:
      ssh:
        user: "<имя_пользователя>"
        key: "<путь_к_открытому_ключу>"
    kuma_roles: ["storage"]  # Хранилище закреплено за этим узлом
  - desc: cdt-w3
    type: worker
    host: "<IPv4_рабочего_узла_3>"
    access:
      ssh:
        user: "<имя_пользователя>"
        key: "<путь_к_открытому_ключу>"
parameters:
  - name: psql_dsn
    source:
      value: "postgres://<dbms_username>:<password>@<fqdn_СУБД>:<порт_СУБД>" # Если пароль
содержит спецсимволы, их необходимо привести к URI кодировке
  - name: ingress_ip
    source:
      value: "<IPv4_шлюза_кластера_Kubernetes>" # Ранее зарезервированный ingress_ip
  - name: ssh_pk
    source:
      path: "<путь_к_закрытому_ключу_на_устройстве_администратора>"
  - name: admin_password
    source:
      value: "<пароль_учётной_записи_admin>"
```

```
  - name: core_disk_request
    source:
      value: 100Gi
  - name: metrics_disk_request
    source:
        value: 80Gi
  - name: inventory
    source:
      value: "/dev/null"  # Все сервисы KUMA будут внутри кластера Kubernetes
  - name: license
    source:
      value: "<путь_к_лицензионному_ключу>" # Рекомендуется файл переименовать в
license.key
  - name: smp_domain
    source:
      value: "<доменное_имя>"
  - name: pki_host_list
    source:
      value: "admsrv api console kuma monitoring agentserver updater"
  - name: grafana_admin_user
    source:
      value: <имя_пользователя>
  - name: grafana_admin_password
    source:
      value: "ваш_пароль"
```

Формат файла для демонстрационной конфигурации с сервисами KUMA внутри кластера

```
schemaType: ParameterSet
schemaVersion: 1.0.1
namespace: ""
name: bootstrap
project: xdr
nodes:
  - desc: cdt-1
    type: primary-worker
    host: "<IPv4_единственного_узла>"
```

```
    access:
      ssh:
        user: "<имя_пользователя>" # По умолчанию root
        key: "<путь_к_ключу>" # По умолчанию /root/.ssh/id_rsa
parameters:
  - name: psql_dsn
    source:
      value: "postgres://<dbms_username>:<password>@<fqdn_СУБД_в_кластере>:<порт>"  # Если пароль содержит спецсимволы, их необходимо привести к URI кодировке
  - name: ingress_ip
    source:
      value: "<IPv4_шлюза_кластера_Kubernetes>" # Ранее зарезервированный ingress_ip
  - name: ssh_pk
    source:
      path: "<путь_к_ключу>"
  - name: admin_password
    source:
      value: "<пароль_учётной_записи_admin>"
  - name: inventory
    source:
      value: "/dev/null" # Все сервисы KUMA будут внутри кластера Kubernetes
  - name: license
    source:
      value: "<путь_к_лицензионному_ключу>" # Рекомендуется файл переименовать в license.key
  - name: smp_domain
    source:
      value: "<доменное_имя>"
  - name: pki_host_list
    source:
      value: "admsrv api console kuma monitoring agentserver updater"
  - name: low_resources
    source:
      value: "true"
  - name: openbao_ha_mode
    source:
      value: "false"
  - name: default_class_replica_count
    source:
```

```
      value: "1"
  - name: grafana_admin_user
    source:
      value: "<имя_пользователя>"
  - name: grafana_admin_password
    source:
      value: "<пароль_пользователя>"
```

?????????

1. Загрузите на устройство администратора следующие файлы:

- `bin.tar.gz` - архив с утилитой KDT, используется для развертывания и изменения параметров инсталляции
- `xdr-<version>.tar.gz` - транспортный архив с компонентами SMP
- `license.key` - ключ лицензии

2. Распакуйте архив `bin.tar.gz`

```
tar -xzf bin.tar.gz
```

3. Дайте файлу kdt права на выполнение:

```
chmod +x ./kdt
```

4. Запустите установку командой:

```
./kdt apply -k <полный путь к транспортному архиву> -i <полный путь к файлу конфигурации param.yaml> --accept-eula

#При установке от УЗ root добавить ключ --allow-root
```

????? 5. ??????? ???????? ????? ?????????

Первый вход в веб-интерфейс:

1. Откройте в браузере **https://console.<smp_domain>**
2. Войдите под учётной записью **admin** с паролем из параметра **admin_password** и завершите настройку 2FA.

Активация лицензии в части KSC:

1. В главном окне приложения нажмите на имя Сервера администрирования. Откроется окно свойств Сервера администрирования
2. На вкладке **Общие** выберите раздел **Лицензионные ключи**
3. В разделе **Действующая лицензия** нажмите на кнопку **Выбрать**
4. В открывшемся окне выберите лицензионный ключ, который вы хотите использовать для активации Kaspersky EDR Expert (on-premise) 8.0. Если лицензионного ключа нет в списке, нажмите на кнопку **Добавить лицензионный ключ** и укажите новый лицензионный ключ
5. Нажмите на кнопку **Сохранить**

Активация лицензии в части KUMA:

1. В главном окне приложения перейдите в **Параметры**, выберите раздел **KUMA** и нажмите **Открыть параметры KUMA**
2. Перейдите в раздел **Лицензия** и убедитесь, что лицензия добавлена
3. Если нет, то нажмите на группу лицензии и в редакторе укажите код активации или файл ключа KEDR

Предоставление доступа к функционалу Kaspersky EDR:

1. В главном окне приложения перейдите в **Параметры**, выберите раздел **Тенанты** и нажмите на **Root tenant**
2. На вкладке **Права доступа** нажмите на кнопку **Добавить пользователя**
3. Выберите пользователя **admin** и предоставьте ему роль **Главный администратор**
4. Нажмите кнопку **Сохранить** и далее **Сохранить и закрыть**

Установка плагинов управления:

1. Загрузите плагин управления конечным устройством для OSMP с расширением .tar на узел администратора
2. Выполните установку командой:

```
./kdt apply -k <полный путь к tar-архиву с плагином управления>


#При установке от УЗ root добавить ключ --allow-root
```

Обновление баз Kaspersky EDR:

1. В главном окне приложения перейдите в **Параметры**, выберите раздел **Тенанты** и нажмите на **Root tenant**
2. На вкладке **Параметры** перейдите к **Параметры сканирования** и на вкладку **Обновить базы**
3. Запустите обновление и дождитесь успешного статуса

Подключение Sandbox:

1. В главном окне приложения перейдите в **Параметры**, выберите раздел **Тенанты** и нажмите на **Root tenant**
2. На вкладке **Параметры** перейдите к **Серверы Sandbox** и нажмите кнопку **Добавить**
3. Укажите IP-адрес Sandbox 8 версии, получите отпечаток, задайте произвольное имя, нажмите кнопку **Добавить**
4. В браузере перейдите и авторизуйтесь в веб-консоли Sandbox и примите запрос на подключение в разделе **Авторизация**
5. В консоли OSMP дождитесь, что статус авторизации изменился на **Одобрено** и отобразились наборы виртуальных машин

Настройка политик для подключения телеметрии с конечных устройств:

1. В главном окне приложения перейдите в **Управление активами - Политики**
2. Создайте политику или откройте её свойства и перейдите в **Параметры приложения**
3. Раскройте ветку **Встроенные агенты** и перейдите в параметры для **Endpoint Detection and Response Expert (on-premise)**
4. Установите переключатель в статус **ВКЛЮЧЕН.** Установите переключатель замка, чтобы изменение применялось на конечном узле
5. Выберите **Endpoint Detection and Response Expert (версия 8.0 и выше)**
6. Для раздела **Подключение к серверам сбора телеметрии** нажмите кнопку **Добавить** и нажмите **Выбрать сервер**
7. В открывшемся окне отметьте единственный сервер коллектора и нажмите кнопку **Добавить**
8. Ниже повторите действия для указания сервера реагирования
9. При необходимости настройте другие пункты политики и сохраните изменения

Настройка политик для подключения Sandbox к конечным устройствам:

1. В главном окне приложения перейдите в **Параметры**, выберите раздел **Тенанты** и нажмите на **Root tenant**
2. На вкладке **Параметры** перейдите к **Сертификаты** и нажмите кнопку **Экспортировать** сертификат сервера и ниже сертификат Endpoint Agent, указав пароль
3. В главном окне приложения перейдите в **Управление активами - Политики**
4. Создайте политику или откройте её свойства и перейдите в **Параметры приложения**
5. Раскройте ветку **Встроенные агенты** и перейдите в параметры для **Sandbox**
6. Установите переключатель в статус **ВКЛЮЧЕН.** Установите переключатель замка, чтобы изменение применялось на конечном узле
7. Выберите режим интеграции **KATA Sandbox**
8. Для раздела **Подключение к серверам Sandbox** нажмите кнопку **Добавить** и укажите адрес сервера **agentserver.<smp_domain>**
9. Нажмите **Настройки подключения** и загрузите экспортированный сертификат сервера. В этом же поле обязательно включите использование двусторонней

аутентификации и подставьте сертификат Endpoint Agent с паролем, указанным при его экспорте

10. При необходимости настройте другие пункты политики и сохраните изменения

Базовая настройка завершена. Подключите совместимые приложения KES для Windows 12.11 и выше, KES для Linux 12.4, KES для Mac 12.2.1 и приступайте к работе с Kaspersky EDR Expert (on-premise).

???? ?? ????????????? Cross-DC Kaspersky EDR Expert (on-premise)

> Инструкция составлена на основе официальной документации KEDR 8.3 и опыта эксплуатации.
> Дата исправления: 12.10.2026

1. ??????????? ? ?????????? ? ????????

1.1. ????? ??????

- **[3DC] 3 полноценных ЦОДа:** Симметричная архитектура. Storage: 1 шард × 3 реплики. K8s Workers: 3+3+3. HAProxy: 3 узла.
- **[2.5DC] Модель «2,5 ЦОДа»:** Асимметричная. ЦОД 3 - только арбитраж. Storage: 1 шард × 4 реплики (2+2+0). K8s Workers: 3+3+1. HAProxy: 2 узла.

1.2. ??????? ??????????

- **Ingress IP** (шлюз кластера) должен быть в **той же подсети**, что и узлы Primary/Worker в данном ЦОДе.
- Все узлы Kubernetes внутри одного ЦОДа должны находиться в **одном широковещательном домене (L2)**.
- Задержка между ЦОДами: **≤ 20 мс**.

1.3. ????????? ???????????

Для корректной работы кластера Kubernetes и балансировки трафика (MetalLB) на уровне гипервизора или сетевого коммутатора для всех целевых хостов **необходимо отключить защиту от ARP/MAC-спуфинга**:

- **VMware ESXi:** В настройках vSwitch или Port Group включить **Forged Transmits** и **MAC Address Changes** (Accept).
- **Hyper-V:** Включить **MAC Address Spoofing** для виртуальных сетевых адаптеров.
- **Физические коммутаторы:** Отключить Dynamic ARP Inspection (DAI) на портах или настроить их как доверенные (trusted).

---

2. ?????????? ???????????? ??????? (??? ????)

Выполнить под пользователем **root** или **<USER_NAME>** с правами **sudo NOPASSWD**.

```
# 1. Обновление и базовые пакеты
apt update && apt upgrade -y
apt install -y python3 python3-apt curl systemd-timesyncd net-tools jq nfs-common tar
wireguard wireguard-tools libnfs12 open-iscsi cryptsetup cryptsetup-bin netcat-traditional
containerd python3-cryptography python3-cffi-backend python3-netaddr

# 2. Проверка cgroup v2 (Обязательно)
mount | grep cgroup # Ожидаемый вывод: cgroup2 on /sys/fs/cgroup type cgroup2

# 3. Отключение swap
swapoff -a
sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

# 4. Отключение SELinux (если установлен)
setenforce 0
sed -i 's/^SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config

# 5. Синхронизация времени (Разброс ≤ 5 секунд)
apt install -y chrony
systemctl enable chrony && systemctl start chrony
timedatectl set-ntp true
timedatectl set-timezone Europe/Moscow
timedatectl set-local-rtc 0

# 6. Создание пользователя для KDT
useradd -m -s /bin/bash <USER_NAME>
echo "<USER_NAME> ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/<USER_NAME>
chmod 440 /etc/sudoers.d/<USER_NAME>

# 7. Настройка UFW (если используется)
ufw allow 22/tcp
ufw allow 6443/tcp       # Kubernetes API
ufw allow 2379:2380/tcp  # etcd
ufw allow 10250/tcp      # Kubelet
ufw allow 5432/tcp       # PostgreSQL
ufw allow 7210/tcp       # KUMA Core API
ufw allow 7220:7240/tcp  # KUMA сервисы
ufw allow 8123/tcp       # ClickHouse
```

```
ufw allow 9443/tcp
ufw allow 13000/tcp      # Агентский трафик
ufw allow 80/tcp         # HTTP redirect
ufw allow 514/tcp        # Syslog (опционально, но рекомендуется)
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/' /etc/sysctl.conf
sysctl -p
# Для Primary узлов K8s разрешить IP-переадресацию:
sed -i 's/DEFAULT_FORWARD_POLICY="DROP"/DEFAULT_FORWARD_POLICY="ACCEPT"/' /etc/default/ufw
ufw reload
```

---

3. ??????? PostgreSQL

3.1. ????????? (?? ???? 3 ????? ??)

```
apt update
apt install -y postgresql-16 patroni etcd etcd-client
systemctl stop postgresql
systemctl disable postgresql
```

3.2. ???????????? Patroni (**/etc/patroni/patroni.yml**)

```
scope: xdr_cluster
namespace: /service/
name: <ИМЯ_УЗЛА> # db-dc1 / db-dc2 / db-dc3

restapi:
  listen: <IP_ЭТОГО_УЗЛА>:8008
  connect_address: <IP_ЭТОГО_УЗЛА>:8008

etcd:
  hosts: <DB_DC1_IP>:2379,<DB_DC2_IP>:2379,<DB_DC3_IP>:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    synchronous_mode: true
    synchronous_mode_strict: true
```

```
    postgresql:
      use_pg_rewind: true
      parameters:
        listen_addresses: '*'
        port: 5432
        max_connections: 512
        shared_buffers: 8192MB
        temp_buffers: 24MB
        work_mem: 16MB
        maintenance_work_mem: 128MB
        max_stack_depth: 7MB
        max_parallel_workers_per_gather: 0
        standard_conforming_strings: on
        log_hostname: 1
        synchronous_commit: "on"

postgresql:
  listen: <IP_ЭТОГО_УЗЛА>:5432
  connect_address: <IP_ЭТОГО_УЗЛА>:5432
  data_dir: /var/lib/postgresql/16/main
  bin_dir: /usr/lib/postgresql/16/bin
  pgpass: /tmp/pgpass
  authentication:
    replication:
      username: replicator
      password: <REPLICATOR_PASSWORD>
    superuser:
      username: postgres
      password: <POSTGRES_PASSWORD>
```

Специфика для [2.5DC] (Узел в ЦОД 3 — Witness): В секцию **tags** добавить:

```
tags:
  nofailover: true
  noloadbalance: true
  nosync: true
```

*(Для [3DC] оставить **tags** пустыми или со значениями **false**).*

3.3. ?????? ? ????????? ????

```
systemctl enable patroni && systemctl start patroni
patronictl -c /etc/patroni/patroni.yml list

# Создание пользователя (выполнить на Leader-узле через psql)
sudo -u postgres psql -h <DB_VIP_IP> -p 5432 <<EOF
CREATE USER <kaspersky_admin> WITH PASSWORD '<StrongPassword123!>' SUPERUSER;
CREATE DATABASE <kaspersky_admin> OWNER <kaspersky_admin>;
GRANT pg_read_all_stats TO "<kaspersky_admin>";
EOF
```

3.4. pg_hba.conf (?? ???? ????? ??)

```
echo "host all all 0.0.0.0/0 scram-sha-256" | tee -a /etc/postgresql/16/main/pg_hba.conf
echo "host replication replicator 0.0.0.0/0 scram-sha-256" | tee -a
/etc/postgresql/16/main/pg_hba.conf
sudo -u postgres psql -c "SELECT pg_reload_conf();"
```

---

4. ????????????? (HAProxy + Keepalived)

4.1. ????????? ? Keepalived

```
apt update && apt install -y haproxy keepalived
```

4.2. Keepalived (**/etc/keepalived/keepalived.conf**)

[3DC] Три узла с приоритетами 101 / 100 / 99.
[2.5DC] Два узла с приоритетами 101 (MASTER, ЦОД 1) и 100 (BACKUP, ЦОД 2).

```
vrrp_script chk_haproxy {
    script "killall -0 haproxy"
    interval 2
    weight 2
}

vrrp_instance VI_1 {
    state MASTER        # BACKUP на резервных узлах
    interface <ИМЯ_ИНТЕРФЕЙСА>
    virtual_router_id 51
    priority 101        # 100 / 99 на остальных
    advert_int 1
    authentication {
```

```
        auth_type PASS
        auth_pass <KEEPALIVED_AUTH_PASSWORD>
    }
    virtual_ipaddress {
        <HAPROXY_VIP_IP>
    }
    track_script {
        chk_haproxy
    }
}
```

```
systemctl enable keepalived && systemctl start keepalived
# Проверка: ip a | grep <HAPROXY_VIP_IP>
```

4.3. HAProxy (**/etc/haproxy/haproxy.cfg**)

```
global
    log /dev/log local0
    log /dev/log local1 notice
    chroot /var/lib/haproxy
    stats socket /run/haproxy/admin.sock mode 660 level admin
    stats timeout 30s
    user haproxy
    group haproxy
    daemon

defaults
    log global
    mode tcp
    option tcplog
    option dontlognull
    timeout connect 5s
    timeout client 50s
    timeout server 50s

frontend http_front
    bind *:80
    default_backend http_back
frontend https_front
    bind *:443
```

```
    default_backend https_back
frontend https9_front
    bind *:9443
    default_backend https9_back
frontend https4_front
    bind *:4443
    default_backend https4_back
frontend agent_front
    bind *:13000
    default_backend agent_back

backend http_back
    option tcp-check
    timeout check 2s
    server dc1_ingress <INGRESS_IP_DC1>:80 check inter 5s
    server dc2_ingress <INGRESS_IP_DC2>:80 check inter 5s
    server dc3_ingress <INGRESS_IP_DC3>:80 check inter 5s

backend https_back
    option tcp-check
    timeout check 2s
    server dc1_ingress <INGRESS_IP_DC1>:443 check inter 5s
    server dc2_ingress <INGRESS_IP_DC2>:443 check inter 5s
    server dc3_ingress <INGRESS_IP_DC3>:443 check inter 5s

backend https9_back
    option tcp-check
    timeout check 2s
    server dc1_ingress <INGRESS_IP_DC1>:9443 check inter 5s
    server dc2_ingress <INGRESS_IP_DC2>:9443 check inter 5s
    server dc3_ingress <INGRESS_IP_DC3>:9443 check inter 5s

backend https4_back
    option tcp-check
    timeout check 2s
    server dc1_ingress <INGRESS_IP_DC1>:4443 check inter 5s
    server dc2_ingress <INGRESS_IP_DC2>:4443 check inter 5s
    server dc3_ingress <INGRESS_IP_DC3>:4443 check inter 5s

backend agent_back
```

```
    option tcp-check
    timeout check 2s
    server dc1_ingress <INGRESS_IP_DC1>:13000 check inter 5s
    server dc2_ingress <INGRESS_IP_DC2>:13000 check inter 5s
    server dc3_ingress <INGRESS_IP_DC3>:13000 check inter 5s
```

```
systemctl enable haproxy && systemctl restart haproxy
```

4.4. DNS

На корпоративном DNS-сервере создаются A-записи, указывающие на **VIP балансировщика** (**<HAPROXY_VIP_IP>**):

```
console.<smp_domain>
admsrv.<smp_domain>
api.<smp_domain>
monitoring.<smp_domain>
updater.<smp_domain>
agentserver.<smp_domain>
kuma.<smp_domain>
*.kuma.<smp_domain>   ← wildcard обязателен
```

---

5. ?????????? ?????????? ?????????????

```
# 1. Установка Docker
apt update && apt install -y ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
apt update && apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin
usermod -aG docker <USER_NAME>
newgrp docker

# 2. Генерация SSH-ключей
su - <USER_NAME>
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa -N ""
# Скопировать на ВСЕ узлы (K8s, DB, KUMA, HAProxy):
```

```
ssh-copy-id <USER_NAME>@<TARGET_IP>
```

6. ?????????????? ????? ??? KDT

6.1. ???? ????????? KUMA (**kuma_distributed_inventory.yml**)

Создать на Admin Node.

[3DC]

```
all:
  vars:
    generate_etc_hosts: false
    deploy_example_services: false
    ansible_connection: local
    ansible_user: nonroot

kuma:
  vars:
    ansible_connection: ssh
    ansible_user: "<USER_NAME>"
    ansible_become: true
    ansible_become_method: sudo
  children:
    kuma_utils:
      hosts:
        kuma-utils.local:
          ansible_host: <ADMIN_NODE_IP>

    kuma_collector:
      hosts:
        kuma-coll-dc1.local:
          ansible_host: <KUMA_COLL_DC1_IP>
        kuma-coll-dc2.local:
          ansible_host: <KUMA_COLL_DC2_IP>
        kuma-coll-dc3.local:
          ansible_host: <KUMA_COLL_DC3_IP>

    kuma_correlator:
      hosts:
```

```
    kuma-corr-dc1.local:
      ansible_host: <KUMA_CORR_DC1_IP>
    kuma-corr-dc2.local:
      ansible_host: <KUMA_CORR_DC2_IP>
    kuma-corr-dc3.local:
      ansible_host: <KUMA_CORR_DC3_IP>

kuma_storage:
  hosts:
    kuma-stor-dc1.local:
      ansible_host: <KUMA_ST_DC1_IP>
      shard: 1
      replica: 1
      keeper: 1
    kuma-stor-dc2.local:
      ansible_host: <KUMA_ST_DC2_IP>
      shard: 1
      replica: 2
      keeper: 2
    kuma-stor-dc3.local:
      ansible_host: <KUMA_ST_DC3_IP>
      shard: 1
      replica: 3
      keeper: 3

kuma_keeper:
  hosts:
    kuma-keep-dc1.local:
      ansible_host: <KUMA_KP_DC1_IP>
      keeper_id: 1
    kuma-keep-dc2.local:
      ansible_host: <KUMA_KP_DC2_IP>
      keeper_id: 2
    kuma-keep-dc3.local:
      ansible_host: <KUMA_KP_DC3_IP>
      keeper_id: 3
```

*(Для **[2.5DC]** использовать конфигурацию 1 шард × 4 реплики: 2 узла Storage в ЦОД 1, 2 узла в ЦОД 2, 0 в ЦОД 3. Collector/Correlator только в ЦОД 1 и 2).*

6.2. ???????? ???? ?????????? KDT (**params.yaml**)

```
schemaType: ParameterSet
schemaVersion: 1.0.1
namespace: ""
name: bootstrap
project: xdr

regions:
  dc1_active:
    description: "Data Center 1"
  dc2_standby:
    description: "Data Center 2"
  dc3_arbiter:
    description: "Data Center 3"

zones:
  zone1:
    region: dc1_active
    ingress_ip: <INGRESS_IP_DC1>
  zone2:
    region: dc2_standby
    ingress_ip: <INGRESS_IP_DC2>
  zone3:
    region: dc3_arbiter
    ingress_ip: <INGRESS_IP_DC3>

nodes:
  # --- ЦОД 1 ---
  - desc: k8s-pri-dc1
    type: primary
    zone: zone1
    host: <K8S_PRI_DC1_IP>
    access:
      ssh:
        user: "<USER_NAME>"
        key: "/home/<USER_NAME>/.ssh/id_rsa"
  - desc: k8s-w-dc1-1
    type: worker
    zone: zone1
    host: <K8S_W_DC1_1_IP>
    access:
```

```
    ssh:
      user: "<USER_NAME>"
      key: "/home/<USER_NAME>/.ssh/id_rsa"
- desc: k8s-w-dc1-2
  type: worker
  zone: zone1
  host: <K8S_W_DC1_2_IP>
  access:
    ssh:
      user: "<USER_NAME>"
      key: "/home/<USER_NAME>/.ssh/id_rsa"
- desc: k8s-w-dc1-3
  type: worker
  zone: zone1
  host: <K8S_W_DC1_3_IP>
  access:
    ssh:
      user: "<USER_NAME>"
      key: "/home/<USER_NAME>/.ssh/id_rsa"

# --- ЦОД 2 ---
- desc: k8s-pri-dc2
  type: primary
  zone: zone2
  host: <K8S_PRI_DC2_IP>
  access:
    ssh:
      user: "<USER_NAME>"
      key: "/home/<USER_NAME>/.ssh/id_rsa"
- desc: k8s-w-dc2-1
  type: worker
  zone: zone2
  host: <K8S_W_DC2_1_IP>
  access:
    ssh:
      user: "<USER_NAME>"
      key: "/home/<USER_NAME>/.ssh/id_rsa"
- desc: k8s-w-dc2-2
  type: worker
  zone: zone2
```

```
    host: <K8S_W_DC2_2_IP>
    access:
      ssh:
        user: "<USER_NAME>"
        key: "/home/<USER_NAME>/.ssh/id_rsa"
  - desc: k8s-w-dc2-3
    type: worker
    zone: zone2
    host: <K8S_W_DC2_3_IP>
    access:
      ssh:
        user: "<USER_NAME>"
        key: "/home/<USER_NAME>/.ssh/id_rsa"

  # --- ЦОД 3 ---
  # [3DC]: добавьте сюда еще 3 worker узла (k8s-w-dc3-2, k8s-w-dc3-3) по аналогии.
  # [2.5DC]: оставьте только 1 primary и 1 worker узел, как указано ниже.
  - desc: k8s-pri-dc3
    type: primary
    zone: zone3
    host: <K8S_PRI_DC3_IP>
    access:
      ssh:
        user: "<USER_NAME>"
        key: "/home/<USER_NAME>/.ssh/id_rsa"
  - desc: k8s-w-dc3-1
    type: worker
    zone: zone3
    host: <K8S_W_DC3_1_IP>
    access:
      ssh:
        user: "<USER_NAME>"
        key: "/home/<USER_NAME>/.ssh/id_rsa"

parameters:
  - name: psql_dsn
    source:
      value:
"postgres://<kaspersky_admin>:<StrongPassword123!>@<DB_DC1_IP>:5432,<DB_DC2_IP>:5432,<DB_DC3_I
P>:5432"
```

```
  - name: ssh_pk
    source:
      path: "/home/<USER_NAME>/.ssh/id_rsa"
  - name: admin_password
    source:
      value: "<STRONG_ADMIN_PASSWORD>"
  - name: inventory
    source:
      value: "/home/<USER_NAME>/kuma_distributed_inventory.yml"
  - name: license
    source:
      value: "/home/<USER_NAME>/license.key"
  - name: smp_domain
    source:
      value: "<smp_domain>"
  - name: pki_host_list
    source:
      value: "admsrv api console kuma monitoring agentserver updater"
  - name: grafana_admin_user
    source:
      value: "xdr_grafana"
  - name: grafana_admin_password
    source:
      value: "<STRONG_GRAFANA_PASSWORD>"

# ВАЖНО: После последней строки обязательно должен быть один пустой перенос строки (нажмите Enter)!
```

7. ????????????? ???? ??????? (KDT)

На **Admin Node**:

```
cd ~
mkdir kdt && tar -xzf bin.tar.gz -C kdt
chmod +x ./kdt/kdt

./kdt/kdt apply -k /home/<USER_NAME>/xdr-<version>.tar.gz -i /home/<USER_NAME>/params.yaml --accept-eula
```

8. ????????? ??????? ???????? KUMA

8.1. ?????????? ?????? KUMA ????? KDT

Если хосты не были подготовлены во время установки SMP, выполните на Admin Node:

```
./kdt/kdt invoke kuma --action addHosts --param
host_inventory='/home/<USER_NAME>/kuma_distributed_inventory.yml'
```

8.2. ????????? ???????? ????? CLI

1. **Storage:** В консоли OSMP: *Мониторинг → Ресурсы и сервисы → Хранилища → Создать* . Указать FQDN, shard, replica, keeper_id. Скопировать **<service_id>**. На хосте выполнить:

   ```
   sudo /opt/kaspersky/kuma/kuma storage --core https://kuma.<smp_domain>:7210 --id
   <service_id> --api.port 7230 --install
   ```

2. **Точка назначения Storage:** *Мониторинг → Ресурсы и сервисы → Точки назначения → Создать* (Тип: storage).
3. **Correlator:** В консоли: *Корреляторы → Создать*. Привязать правила, в маршрутизации выбрать точку назначения Storage. Скопировать команду. На хосте выполнить:

   ```
   sudo /opt/kaspersky/kuma/kuma correlator --core https://kuma.<smp_domain>:7210 --id
   <service_id> --api.port 7220 --install
   ```

4. **Точка назначения Correlator:** Создать аналогично (Тип: correlator).
5. **Collector:** В консоли: *Коллекторы → Создать*. Указать транспорт, парсинг. В маршрутизации добавить **обе** точки назначения (Correlator и Storage). На хосте выполнить:

   ```
   sudo /opt/kaspersky/kuma/kuma collector --core https://kuma.<smp_domain>:7210 --id
   <service_id> --api.port 7221 --install
   ```

   (Примечание: если несколько сервисов на одном хосте, используйте уникальные порты, например 7222, 7223).

---

9. ?????????? ??? Failover

При физическом отказе целого ЦОДа (любого из активных):

1. **Потеря детектов Sandbox:** Файлы, отправленные в упавший ЦОД и не успевшие обработаться, теряются. Окно потери: **15 минут** (при пиковой нагрузке - до 1 часа).
2. **Потеря задач IRP:** Задачи "Get File and Scan" и "IoA Scan", не завершённые к моменту сбоя, теряются безвозвратно.

3. **Задержки KSB:** Автоматические задачи проверки Kaspersky Sandbox не перезапускаются автоматически в течение **24 часов**.
4. **Перегрузка очередей:** При переключении всей нагрузки на один ЦОД возможна перегрузка Sandbox и «дроп» части заданий.
5. **Неконсистентность:** Возможна локальная неконсистентность данных в окне **10 минут** до и после отказа ЦОДа.