

???? ? ???? 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-переедресацию:
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 А-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 BCE 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:
  # --- Ц0Д 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"

# --- Ц0Д 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.
-

Revision #8

Created 12 October 2026 07:59:39 by Кирилл

Updated 12 October 2026 09:05:15 by Кирилл