Установка кассового сервера

Кассовый сервер УКМ 5 представляет собой средство управления настройками касс и функционально описан **здесь**.

Перед установкой кассового сервера, необходимо развернуть сервер на ESXI 6.5 , соответствующий рекомендуемым системным требованиям тестового сервера:
- 6 ядер CPU;
- 22gb ОЗУ (доступную через команду **htop**);
- 100gb жесткий диск;
- 2 сетевых интерфейса:
- eth0 – интерфейс выхода в локальную сеть,
- eth1 – должен быть присвоен адрес **172.17.*.*** (виртуальная сеть, без выхода в локальную сеть), если сеть настроена правильно - адрес присвоится автоматически;
- доступ в Интернет на момент установки.

Скорость дисков при проверке командой **hdparm -tT** должна быть не ниже **Timing buffered disk reads 380 MB/sec** (без нагрузки).

Сервер должен находиться в одной локальной сети с кассами.

Серверная часть кассовой программы, а также программное обеспечение кассовых терминалов полноценно функционируют на ОС Linux, включая x64/x86 debian-based.

Необходимо предоставить дуплексный доступ кассового сервера УКМ 5 к серверу лицензирования через интернет по протоколу HTTPS с пропускной способностью не менее 24 кбайт/сек (приблизительно) на порт **15002**.

После разворачивания виртуальной машины, необходимо подключиться к ней по **ssh** с помощью утилиты **putty** или аналогичной.

Логин и пароль нужно запросить у представителя технической поддержки.

Далее, с помощью проводника (**mc**) или командной строки создаем следующие каталоги:

Создаем требуемые папки:

```
mkdir /opt/ukm5-server
mkdir /home/support/1.37
```

Закачиваем архив **docker-1.37.2.zip** в **/home/support/1.37/** и архив **kubernetes-1.37.2.tgz** в **/home/support/1.37/** с нашего **ftp**

Проверка сетевых адаптеров и настройка DNS

Интерфейс **eth0** – внутренняя сеть клиента, статический.

> ⚠ Не должен начинаться на 172.17, 172.18, 10.0 – на эти адреса завязана работа внутри кубов.

Интерфейс **eth1** (172.17.8.101) должен быть подключен в изолированную сеть (DMZ).

Интерфейс **docker0** – автоматический.

Настраиваем DNS в докере, чтобы сервер нормально общался в сети клиента со службами внутри себя и вне себя:

Пример правильной настроенной сети:

```
sudo mcedit /etc/systemd/system/docker.service.d/docker-dns.conf
```

Если у клиента нет своих DNS-серверов, можно оставить общедоступные:

Содержимое docker-dns.conf

```
[Service]
Environment="DOCKER_DNS_OPTIONS=\
--dns 10.100.0.3 --dns 8.8.8.8 --dns 8.8.4.4 \
--dns-search default.svc.cluster.local --dns-search svc.cluster.local \
--dns-opt ndots:2 --dns-opt timeout:2 --dns-opt attempts:2 \
"
```

10.100.0.3 – системный кубовый DNS, этот параметр нельзя менять.

Вывод ifconfig

```
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
inet 172.18.0.1 netmask 255.255.0.0 broadcast 172.18.255.255
ether 02:42:34:f3:65:60 txqueuelen 0 (Ethernet)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet <ip-адрес_клиента> netmask 255.255.255.0 broadcast <ip-шлюза_клиента>
inet6 fe80::20c:29ff:fec4:dfdd prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:c4:df:dd txqueuelen 1000 (Ethernet)
RX packets 13226845 bytes 6830028149 (6.3 GiB)
RX errors 0 dropped 4 overruns 0 frame 0
TX packets 2100608 bytes 157899252 (150.5 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.17.8.101 netmask 255.255.255.0 broadcast 172.17.8.255
inet6 fe80::20c:29ff:fec4:dfe7 prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:c4:df:e7 txqueuelen 1000 (Ethernet)
RX packets 9069806 bytes 697074353 (664.7 MiB)
RX errors 0 dropped 36 overruns 0 frame 0
TX packets 480 bytes 20480 (20.0 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 54962423 bytes 9159436515 (8.5 GiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 54962423 bytes 9159436515 (8.5 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Если на данном этапе не удается получить указанный вывод сетевых интерфейсов, необходимо обратиться к представителю технической поддержки!

Настройка второго диска

Пропускаем этот пункт, если диск один.

Диск (следующие пункты нужны, если система будет на втором диске – "боевой" сервер).
Для тестового сервера пункты по работе с дисками можно пропустить.

ls /sys/class/scsi_host/
echo "- - -" > /sys/class/scsi_host/host0/scan
echo "- - -" > /sys/class/scsi_host/host1/scan
echo "- - -" > /sys/class/scsi_host/host2/scan
fdisk -l

Форматирование диска:

fdisk /dev/sdb
[n]
[p] (primary) – **важно**
<Enter>
<Enter>
<Enter>
[t]
<Enter>
[8e] Linux LVM – **важно**
[p]
[w]

Формат раздела (создастся раздел LVM на все свободное пространство примонтированного диска).

---

partprobe
pvcreate /dev/sdb1
[y]
pvdisplay
vgcreate disk2 /dev/sdb1
vgdisplay
lvcreate -n data -l 100%FREE disk2
[y]
lvdisplay
mkfs.xfs /dev/disk2/data

Необходимо примонтировать диск и перенести данные (оригинальный раздел дата перейдет на новый диск).

sudo mv /data /data.orig
sudo mkdir -p /data
sudo mount /dev/disk2/data /data
sudo echo '/dev/disk2/data /data xfs defaults 0 0' >> /etc/fstab
sudo cp -rp /data.orig/* /data

Перезагрузка сервера

Проверка запуска MariaDB и установка HAproxy

```
sudo systemctl restart mariadb
sudo systemctl enable mariadb
sudo systemctl status mariadb
```

1. Устанавливаем HAproxy (чтобы видеть IP-адреса касс):

```
cd /opt/compose
sudo yum install haproxy
```

2. Копируем конфиг текущего HAproxy в свежеустановленный (перезаписать файл, при необходимости):

```
sudo cp /opt/compose/cfg/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg
```

3. Останавливаем Docker-Compose:

```
sudo docker-compose down
```

4. В текущем файле **/etc/haproxy/haproxy.cfg** комментируем кусок про **frontend stats** (с **216** строки до конца):

```
sudo mcedit /etc/haproxy/haproxy.cfg
```

5. Проверяем текущий конфиг HAproxy на валидность:

```
haproxy -c -f /etc/haproxy/haproxy.cfg
```

Ответ – **Configuration file is valid**.
Если есть ошибки – смотрим, чем вызваны, правим, проверяем снова.

6. Запускаем HAproxy:

```
sudo systemctl start haproxy
sudo systemctl status haproxy
```

7. Удаляем Docker-Compose из автозагрузки:

```
sudo mcedit /opt/compose/docker-compose.yaml
```

Меняем параметр **restart** на **no**:

> restart: 'no'

Сохраняем файл. При следующей загрузке Docker-Compose не стартует контейнер с HAproxy.

8. Добавляем HAproxy в загрузку:

```
sudo systemctl enable haproxy
```

Разворачивание экосистемы

```
cd /opt/ecosystem
sh _create-ecosystem.sh
```

Ожидаем установки и смотрим вывод статуса служб:

> kubectl get pods -A
> kubectl get pods -A -o wide

Prometheus не нужен, его можно удалить.

Настройка внешнего доступа к kafka

Редактируем файл **kube-configmap.yaml**:

```
sudo mcedit /opt/ecosystem/kafka/kube-configmap.yaml
```

Нужно вписать актуальный **IP/FQDN** клиентского сервера УКМ 5 **eth0** в соответствующих строчках:

Содержимое файла kube-configmap.yaml

data:
  KAFKA_CFG_ADVERTISED_HOST_NAME: "*ip-сервера или dns-hostname*"
  KAFKA_CFG_ADVERTISED_LISTENERS: "INTERNAL://:9093,CLIENT://:9092,EXTERNAL://*ip-сервера или dns-hostname*:39094"

```
cd /opt/ecosystem/kafka/
sh _restart-kafka-zookeeper.sh
```

Загружаем докеры:

Распаковываем архив **docker-1.37.2.zip** в каталог **/home/support/1.37** (пароль на архив: **12345**):

```
cd /home/support/1.37 ENTER unzip docker-1.37.2.zip
```

```
sudo docker load < admin-impl-1.37.2
sudo docker load < converters_service-1.37.2
sudo docker load < db-migration-srvdata-1.37.2
sudo docker load < db-migration-srvsales-1.37.2
sudo docker load < flush-queue-1.37.2
sudo docker load < frontend-1.37.2
sudo docker load < import-export-api-1.37.2
sudo docker load < jsreport-1.37.2
sudo docker load < licensing-impl-1.37.2
sudo docker load < login-impl-1.37.2
sudo docker load < logs_cleaner-1.37.2
sudo docker load < monitoring-impl-1.37.2
sudo docker load < ntp_timezones-1.37.2
sudo docker load < pos_agent-1.37.2
sudo docker load < public-api-1.37.2
sudo docker load < receipts_cutter-1.37.2
sudo docker load < tradedata_processing-1.37.2
sudo docker load < transport_service-1.37.2
sudo docker load < transport_service_nginx-1.37.2
```

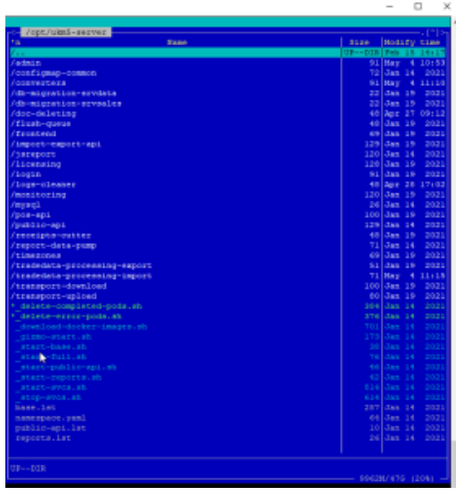
Проверить загруженные контейнеры можно с помощью команды:

sudo docker images

mc [support@ukm5-server-sn]:~/1.28

```
[support@ukm5-server-sn 1.28]$ sudo docker images
REPOSITORY                                          TAG                 IMAGE ID            CREATED             SIZE
docker-repo.servplus.ru:5000/admin-impl             1.37.2              20667821b317        12 days ago         199MB
docker-repo.servplus.ru:5000/jsreport               1.37.1              26405ab35223        2 weeks ago         882MB
docker-repo.servplus.ru:5000/report-data-pump       1.37.1              30cf4ddb774c        2 weeks ago         174MB
docker-repo.servplus.ru:5000/clickhouse             1.37.1              3cad13850753        2 weeks ago         665MB
docker-repo.servplus.ru:5000/public-api             1.37.1              5c69925a416b        2 weeks ago         158MB
docker-repo.servplus.ru:5000/frontend               1.37.1              b6f38747ffaf        2 weeks ago         84.5MB
docker-repo.servplus.ru:5000/transport_service_nginx 1.37.1             39980add21c9        2 weeks ago         23.2MB
docker-repo.servplus.ru:5000/logs_cleaner           1.37.1              7427556d68ba        2 weeks ago         7.67MB
docker-repo.servplus.ru:5000/receipt-search-cleaner 1.37.1              681able697ba        2 weeks ago         7.67MB
docker-repo.servplus.ru:5000/receipt-search-appender 1.37.1             3114c5fca7dc        2 weeks ago         176MB
docker-repo.servplus.ru:5000/receipts_cutter        1.37.1              49152fe9b417        2 weeks ago         344MB
docker-repo.servplus.ru:5000/ntp_timezones          1.37.1              8bda6064f029        2 weeks ago         23.6MB
docker-repo.servplus.ru:5000/import-export-api      1.37.1              c03ca4876e2e        2 weeks ago         193MB
docker-repo.servplus.ru:5000/transport_service      1.37.1              56babd43e15b        2 weeks ago         137MB
docker-repo.servplus.ru:5000/tradedata_processing   1.37.1              fa421cbceccd        2 weeks ago         174MB
docker-repo.servplus.ru:5000/monitoring-impl        1.37.1              a3d0df525bb4        2 weeks ago         181MB
docker-repo.servplus.ru:5000/pos_agent              1.37.1              48166df68f81        2 weeks ago         148MB
docker-repo.servplus.ru:5000/converters_service     1.37.1              dddb3671a89e        2 weeks ago         161MB
docker-repo.servplus.ru:5000/admin-impl             1.37.1              c236b212c39e        2 weeks ago         199MB
docker-repo.servplus.ru:5000/login-impl             1.37.1              50c5f6b0082c        2 weeks ago         157MB
docker-repo.servplus.ru:5000/licensing-impl         1.37.1              7409eef8e7fd        2 weeks ago         169MB
docker-repo.servplus.ru:5000/db-migration-srvsales  1.37.1              2449a9b4f207        2 weeks ago         468MB
docker-repo.servplus.ru:5000/db-migration-srvdata   1.37.1              a4bcf5ad90a0        2 weeks ago         468MB
```

Далее распаковываем архив **kubernetes-1.37.2.tgz** в **/opt/ukm5-server/**:



Вносим в конфигурационные файлы следующие изменения:

Необходимо отредактировать параметр **DOWNLOAD_URL_PREFIX**, заменить **ukm5-qa-server** на **ukm5-server (или ваше FQDN имя сервера)**:

```
sudo mcedit /opt/ukm5-server/configmap-common/configmap-common.yaml
```

Изменяем параметр: **PRODUCE_ITEMS_DOCUMENT** значение с **false** на **true**:

```
sudo mcedit /opt/ukm5-server/tradedata-processing-import/configmap.yaml
```

Далее запускаем установку базовых служб сервера скриптом **_start-base.sh**:

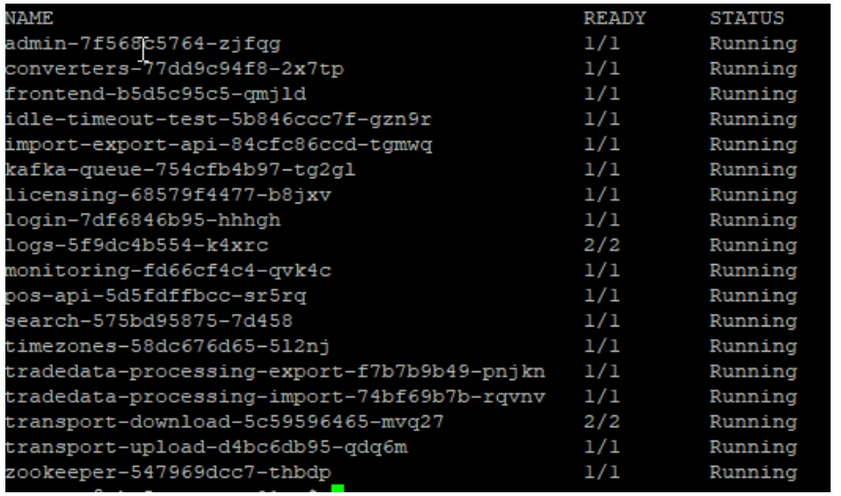
```
sudo su

cd /opt/ukm5-server/
./_start-base.sh
```

Запускаем и ждем ~10-50 минут.

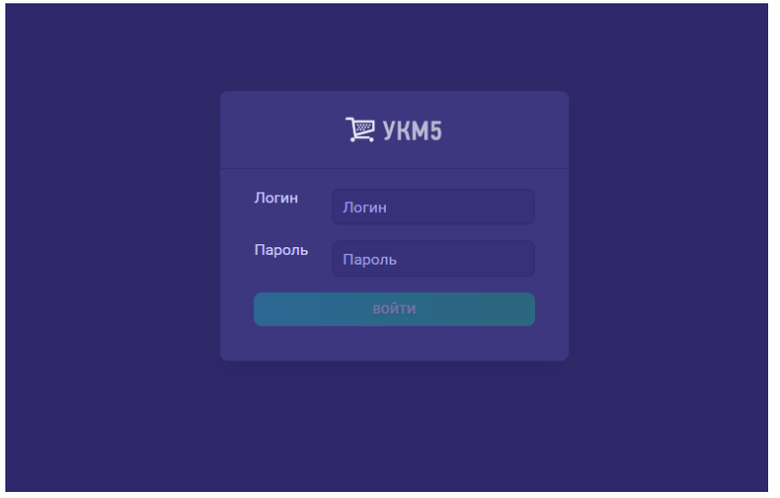
Следить за установкой сервера можно из параллельного окна **putty** следующей командой:

kubectl get pods -A

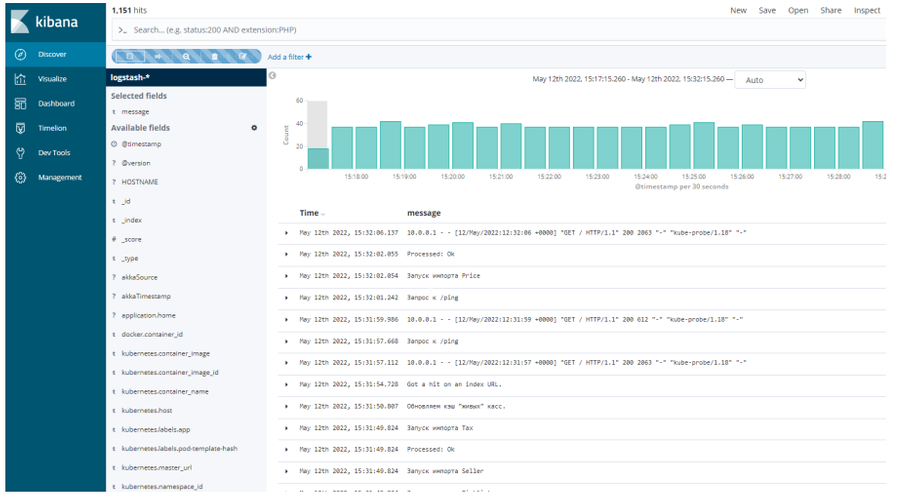
Службы должны запускаться и быть в состоянии **Running 1/1**:



После успешного завершения работы скрипта **_start-base.sh** станет доступен интерфейс администратора сервера по адресу: **http://<ip_адрес_вашего_сервера>/**:



Также будет доступна Kibana для просмотра логов сервера по адресу **http://<ip_адрес_вашего_сервера>:5601**:



Если на данном этапе какая-либо из этих страниц не открывается, необходимо обратиться к представителю технической поддержки!