Программное обеспечение «Базис.Virtual Security». Руководство по установке. Часть 2. Программный модуль «Базис.DynamiX».

RU.НРФЛ.00002-03 93 01 Ч2

Москва

22/03/2024

Содержание

1 Аннотация

В настоящем документе приводятся сведения о действиях по установке и настройке программного модуля «Базис.DynamiX». Настоящий документ предназначен для инженерно-технических специалистов, осуществляющих эксплуатацию, поддержку и сопровождение программного модуля и представляет собой руководство по установке платформы динамической инфраструктуры с собственной системой оркестрации «Базис.DynamiX».

2 Перечень эксплуатационных документов

- RU.НРФЛ.00002-03 93 01 Программное обеспечение «Базис.Virtual Security». Руководство по установке;
- RU.НРФЛ.00002-03 93 01 Ч1 «Программное обеспечение «Базис.Virtual Security». Руководство по установке. Часть 1. Программный модуль «Базис.Virtual Security»;
- RU.НРФЛ.00002-03 93 01 Ч3 «Программное обеспечение «Базис.Virtual Security». Руководство по установке. Часть 3. Программный модуль «Базис.vCore».

3 Общие сведения.

3.1 Назначение

Программный модуль «Базис.DynamiX» представляет собой платформу динамической инфраструктуры с собственной системой оркестрации. Система оркестрации платформы позволяет автоматизировать весь жизненный цикл приложений и использовать механизм self-healing и управлять любыми элементами системы такими, как виртуальные машины, контейнеры, ресурсы хранения, PaaS-блоки и т. д.

Реализуемые задачи:

- тестирование работы программного модуля на определенных ОС;
- разработка "приложения в облаке" (с базами данных) и тестирование во время разработки;
- предоставление виртуальных серверов (вычислительных мощностей) с настроенными службами;
- автоматизация тестирования при помощи контейнеров Kubernetes;
- сервис-провайдер.

Использование технического решения «Базис.DynamiX» предоставляет следующие преимущества:

- быстрый запуск (deployment) инфраструктуры;
- готовые инструменты для управления виртуальным дата-центром и сетевыми функциями;
- блочное хранение с настраиваемой производительностью;
- авторизация и аутентификация через отдельный модуль (security broker);
- полнофункциональный REST API;
- техническая поддержка и возможность индивидуально адаптировать решения для конкретного партнера.

Программный модуль «Базис.DynamiX» состоит из следующих компонентов:

- подсистемы управления, которая реализовывает основные функции централизованного управления виртуальными машинами, виртуализированным оборудованием и другими сущностями среды виртуализации с использованием различных интерфейсов администрирования;
- подсистемы взаимодействия с базой данных, которая предоставляет унифицированные механизмы взаимодействия с системой управления базами данных для обеспечения постоянного хранения данных подсистемы управления;
- подсистемы взаимодействия с агентами, которая реализовывает программные интерфейсы взаимодействия для передачи служебной информации между подсистемой управления и различными сервис-агентами, функционирующими на гипервизорах 1 типа vCore или сертифицированных хостовых операционных системах, которые могут выступать средствами виртуализации;
- вспомогательной системы, которая отвечает за отслеживание состояния вычислительных узлов и мониторинг основных служб, необходимых для корректного функционирования средства виртуализации;
- сервис-агента (JSagent), который обеспечивает информационный обмен между внутренними подсистемами средства виртуализации на базе сертифицированной операционной системы и подсистемой управления.

3.2 Системные требования

К аппаратному и программному обеспечению, которые используются для функционирования программного модуля «Базис.DynamiX», предъявляются требования, изложенные в таблице 1.

Таблица 1. Минимальные требования к программному и аппаратному обеспечению

Элемент	Параметр
Узлы вычисления среды виртуализации	
Операционная система	• Astra Linux Special Edition сертификат № 2557 (выдан ФСТЭК России 30.01.2012, действителен до 27.01.2026); • Альт 8 СП (релиз 10) сертификат № 3866 (выдан ФСТЭК России 10.08.2018, действителен до 10.08.2028).
Гипервизоры 1 типа	• Базис.vCore (для исполнения И1)

Библиотеки	• Libvirt 9.5; • qemu 7.2.
Процессор	• Процессор с тактовой частотой от 2.2 ГГц; • 8 ядер (для тестового и демонстрационного использования); • 10 ядер и больше (промышленного использования); • Поддерживаются процессоры двух типов: AMD64 и Intel64. Процессоры должны иметь поддержку аппаратной виртуализации AMD-V или Intel VT. Также необходимо наличие атрибута NX bit; • Поддерживаемые модели процессоров: • AMD: Семейство процессоров EPYC с микроархитектурой Zen и выше; • Intel: Семейство процессоров Xeon (5 и 6-го поколения) и выше.
Оперативная память	• 32 Гб (ECC) DDR4 2133 МГц (тестового и демонстрационного использования); • 256 Гб (ECC) DDR4 2133 МГц (для промышленного использования).
Жесткий диск	• Не менее двух SSD-дисков по 256 Гб.
Сетевой адаптер	• Не менее двух Ethernet-адаптеров 10 Гбит/сек (для тестового использования); • Для промышленного использования пропускная способность сети выбирается исходя из профиля сетевой нагрузки.
Контроллер управления	
Операционная система	• Astra Linux Special Edition сертификат № 2557 (выдан ФСТЭК России 30.01.2012, действителен до 27.01.2026); • Альт 8 СП (релиз 10) сертификат № 3866 (выдан ФСТЭК России 10.08.2018, действителен до 10.08.2028).

Процессор	• Процессор 8 ядер серверного класса и тактовой частотой от 2.2 ГГц;
	• Требования по архитектуре процессора: x86_64;
	• Поддерживаемые модели:
	  • AMD: с микроархитектурой Zen и выше;
	  • Intel: 5 и 6-го поколения.

Оперативная память	• 32 Гб (ECC) DDR4 2133 МГц (рекомендовано 64 ГБ для промышленной эксплуатации).
Жесткий диск	• Не менее двух SSD 960 Гб.
Сетевой адаптер	• Ethernet 10 Гбит/сек * 2 (для тестового использования); • Для промышленного использования пропускная способность сети выбирается исходя из профиля сетевой нагрузки.
Клиент	
Браузер	• Из состава сертифицированной ОС.
Сетевой адаптер	• Ethernet 100 Мбит/сек.

Монитор	• Диагональ от 17"; • Разрешение от 1280x1024 (4:3), от 1440x900 (16:9).
Периферийное оборудование	• Клавиатура; • Манипулятор типа мышь.

4 Инсталляция программного модуля на базе Astra Linux 1.7.3.

4.1 Установка ОС – Control узлы, CPU узлы.

1. Подключиться к IPMI-интерфейсу узла.
2. Смонтировать образ <1.7.3-03.11.2022_15.53.iso>.
3. Начать графическую установку ОС.
4. Настройки клавиатуры оставить по умолчанию.
5. Имя хоста ввести в соответствии с планом инсталляции, например, "astra-ctrl-01".
6. Задать имя пользователя и пароль: d-energy / d_energy_base.
7. Указать часовой пояс - **"Москва"**.
8. Разбивку дисков привести к виду:
   - 2 GB на boot раздел, остальное для ОС;
   - для Control узлов - *Swap создавать не нужно;*
   - для CPU узлов – Swap 8 GB.
9. При выборе дополнительных пакетов снять весь выбор и поставить выбор только у ssh сервера.
10. При выборе дополнительных настроек ОС выставить уровень защищённости **"Воронеж"**.
11. Выставленные параметры оставить без изменений, так же добавить **"запрет автоматической настройки сети"**.
12. В случае аппаратного рейд-массива или установки без рейд-массива, выбрать **"установить загрузчик в основную загрузочную запись - да"**, ввести пароль d_energy_base для grub и завершить установку.
13. Перезагрузить ОС.
14. Выдать максимальный уровень для пользователя root:
    - авторизоваться под учетной записью d-energy;
    - Inegrity level – указать 63;
    - ввести sudo pdpl-user – i 63 root.
15. В настройках системы включить английскую локаль:
    - ввести dpkg-reconfigure locales;
    - выбрать локали 158 и 388 (английская и русская);
    - в качестве локали по умолчанию поставить английскую.
16. Настроить сеть:
    - установить пакеты для OVS switch из DEB пакетов <Pre_Libs-OVS-LLDP.iso>;
    - настроить сеть см. раздел "Настройка сети".
17. Настроить DNS:
    - создать файл /etc/resolv.conf (даже если не используется и не предоставлен заказчиком DNS сервер – **файл создавать обязательно**);
    - при необходимости указать DNS запись в /etc/resolv.conf (например: *nameserver 8.8.8.8*).

4.2 Настройка локального источника репозиториев пакетов для ОС Астра.

Действия выполняются на «первом» Control узле.

1. Смонтировать ISO с ОС Астра <1.7.3-03.11.2022_15.53.iso>:
   - подключить ISO через IPMI;
   - смонтрировать ISO:

   ```
   mount /dev/cdrom
   ```

2. Установить web-сервер:

   ```
   sudo apt install apache2
   ```

3. Создать символьную ссылку на репозиторий:

   ```
   mkdir /srv/repo/
   sudo ln -s /srv/repo /var/www/html/
   ```

4. Настроить Apache2:

- в /etc/apache2/sites-enabled/000-default.conf после строки DocumentRoot /var/www/html добавить строки:

```
<Directory /var/www/html/repo>
    Options Indexes MultiViews FollowSymLinks
    AllowOverride None
    Order Deny,Allow
    Require all granted
</Directory>
```

- в файле **/etc/apache2/sites-enabled/000-default.conf** изменить строку *<VirtualHost *:80> на строку <VirtualHost *:2080>;*
- в файле **/etc/apache2/apache2.conf** указать параметр *AstraMode off;*
- в файле **/etc/apache2/ports.conf** заменить *Listen 80* на *Listen 2080;*
- перезапустить web-сервер:

```
systemctl restart apache2
```

5. Создать каталоги:

```
mkdir /srv/repo/alse/
mkdir /srv/repo/alse/astra
mkdir /srv/repo/alse/main
mkdir /srv/repo/alse/extended
mkdir /srv/repo/alse/update-base
mkdir /srv/repo/alse/update-main
```

6. Скопировать файлы "Репозитории пакетов Астры" <base-1.7.3-03.11.2022_15.53.tar, extended-1.7.3.ext1.3-23.11.2022_19.59.tar, repository-update.iso> в каталог /tmp/.
7. Распаковать архив <repo-mirror-astra-testing.tar.gz> в каталог /tmp/.
8. Копирование репозиториев:
   - extended:

```
cd /srv/repo/alse/extended
tar -xvf /tmp/extended-1.7.3.ext1.3--23.11.2022_19.59.tar
```

   - update-base:

```
cd /srv/repo/alse/update-base
tar -xvf /tmp/base-1.7.3-03.11.2022_15.53.tar
```

   - update-main:

```
mkdir /mnt/iso
mount -o loop /tmp/repository-update.iso /mnt/iso
cp -r /mnt/iso/* /srv/repo/alse/update-main/
umount /mnt/iso
```

- astra(testing des)

```
cp -r /tmp/repo-mirror/astra/* /srv/repo/alse/astra/
cp /tmp/repo-mirror/repo.asc /srv/repo/alse/astra/
```

9. Создать файл приоритетов для репозитория astra testing /etc/apt/preferences.d/des и добавить в него строки:

```
Package: *
Pin: release a=des-dev
Pin-Priority: 1000
```

10. Отредактировать файл /etc/apt/sources.list:
    - закомментировать все не закомментированные строки в файле;
    - добавить строки, где 10.10.1.1 mgmt адрес первого контроллера:

```
deb http://10.10.1.1:2080/repo/alse/astra/1.7_x86-64/ 1.7_x86-64 testing
deb http://10.10.1.1:2080/repo/alse/update-main/ stable main contrib non-free
deb http://10.10.1.1:2080/repo/alse/update-base/ stable main contrib non-free
deb http://10.10.1.1:2080/repo/alse/extended/ 1.7_x86-64 main contrib non-free
```

11. Раскопировать конфигурационные файлы (sources.list, des, repo.asc) по всем Control и Cpu узлам:

```
for i in 10.10.1.{1..3} 10.10.1.{11..13}; do echo $i; scp /etc/apt/sources.list $i:/etc/apt/sources.list ;done
for i in 10.10.1.{1..3} 10.10.1.{11..13}; do echo $i; scp /etc/apt/preferences.d/des $i:/etc/apt/preferences.d/des ;done
for i in 10.10.1.{1..3} 10.10.1.{11..13}; do echo $i; scp /tmp/repo-mirror/repo.asc $i:/tmp/repo.asc ;done
```

12. На всех Control и Cpu узлах добавить ключ к репозиторию Astra и обновить список репозиториев:

```
for i in 10.10.1.{1..3} 10.10.1.{11..13}; do echo $i; ssh $i 'apt-key add /tmp/repo.asc' ;done
for i in 10.10.1.{1..3} 10.10.1.{11..13}; do echo $i; ssh $i 'apt update' ;done
```

4.3 Настройка сети.

Сети, настраиваемые на CTRL узлах:

- backplane1;
- public;

- mgmt.

Сети, настраиваемые на CPU узлах:

- backplane1;
- mgmt;
- storage*.

> *Если внешнее СХД подключено по iscsi, то необходимо настроить сеть storage.
> Имя интерфейсов должно отличаться от простого "storage", например, можно использовать номер VLAN в названии - "storage101".

Сети, настраиваемые на DES узлах (опционально):

- backplane1;
- mgmt;
- storage.

> Имя интерфейса "storage" использовать только для инсталляций с DES.

Типовые конфигурационных файлов:

- Backplane1;
- MGMT;
- Public;
- Storage.

4.3.1 Backplane1 (сеть для взаимодействия компонентов платформы)

```
vim /etc/network/interfaces.d/Backplane1.conf
```

Привести к виду конфигурационный файл (заменяя на нужные значения):

```
### backplane1
auto backplane1
iface backplane1 inet static
address 172.20.211.12/24
dns-nameservers 172.20.212.254
mtu 9000
pre-up ip l set ens2f0 up
pre-up ip l set ens2f0 mtu 9000
pre-up ip link set dev ens2f0 txqueuelen 10000
pre-up sysctl -w net.ipv6.conf.ens2f0.disable_ipv6=1
post-up tc qdisc replace dev ens2f0 root mq
pre-up ip l set ens4f0 up
pre-up ip l set ens4f0 mtu 9000
pre-up ip link set dev ens4f0 txqueuelen 10000
pre-up sysctl -w net.ipv6.conf.ens4f0.disable_ipv6=1
post-up tc qdisc replace dev ens4f0 root mq
pre-up ovs-vsctl --may-exist add-br backplane1
pre-up ovs-vsctl --may-exist add-bond backplane1 bond-backplane1 ens2f0 ens4f0
bond_mode=balance-tcp lacp=active
pre-up ovs-vsctl set port backplane1 tag=511
post-up tc qdisc replace dev backplane1 root fq
```

```
ifup backplane1
ip -br a
```

4.3.2 MGMT (сеть для внутреннего управления компонентами платформы)

```
vim /etc/network/interfaces.d/mgmt.conf
```

Привести к виду (заменяя на нужные значения… gateway ..254):

```
cat <<EOF > /etc/network/interfaces.d/mgmt.conf
auto mgmt
iface mgmt inet static
address 172.20.212.12/24
netmask 255.255.255.0
gateway 172.20.212.254
post-up tc qdisc replace dev mgmt root fq
pre-up ovs-vsctl --may-exist add-br backplane1
pre-up ovs-vsctl --may-exist add-port backplane1 mgmt -- set Interface mgmt
type=interna
l
pre-up ip l set mgmt up
pre-up ovs-vsctl set port mgmt tag=512
EOF
```

Настройка MTU и сети

Значения MTU:

- backplane1 и интерфейсы бонда MTU = 9000;
- storage = 9000;
- mgmt = 1500.

1. Задать MTU на интерфейсах:

```
ip l set enp22s0f1 mtu 9000
```

2. Задать MTU в ovs:

```
ovs-vsctl set int backplane1 mtu_request=9000
```

Проверить перемещение пакетов между узлами размером 9000 байт.

3. Выполнить команды:

```
для 1500
ping -M do -s 1472 example.com
для 9000
ping -M do -s 8972 example.com
```

Настройка NTP

1. Произвести настройку сервера времени на всех серверах:

```
vim /etc/systemd/timesyncd.conf
```

2. Прописать адрес(а) NTP:

```
[Time]
NTP= <ntp_ip_here>
```

3. Перезапустить службы:

```
systemctl daemon-reload
systemctl restart systemd-timesyncd.service
```

4. Установить часовой пояс:

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

5. Проверить:

```
systemctl status systemd-timesyncd.service
timedatectl
```

Настройка DNS

Произвести настройку DNS (на всех вычислительных узлах).

1. Остановить и отключить службу systemd-resolved:

```
systemctl stop systemd-resolved
systemctl mask systemd-resolved
```

2. Удалить ссылку:

```
rm /etc/resolv.conf
```

3. Создать конфигурационный файл resolv.conf:

```
vim /etc/resolv.conf
```

4.3.3 Проверка работоспособности

Пройти по всем пунктам инструкции и проверить: вычислительные узлы, сети, связность, работу SSH между вычислительными узлами, hostname, timesyncd. Убедиться в корректности настроек.

4.4 Настройка ОС – Control узлы.

1. Разместить SSH ключи для коммуникации Серверов между собой.
2. Настроить IpTables:
   - установить пакет iptables:

     ```
     apt install iptables
     ```

   - оставить только legacy iptables:

     ```
     update-alternatives --set iptables /usr/sbin/iptables-legacy
     update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
     ```

   - создать новый файл **vim /etc/iptables.rules** и вставить строки:

     ```
     *nat
     -A POSTROUTING -o public -j MASQUERADE
     COMMIT
     *filter
     -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
     -A OUTPUT -m conntrack --ctstate ESTABLISHED -j ACCEPT
     -A INPUT -i public -p tcp -m multiport --dports 3080,3023,7022,80,443,22 -j ACCEPT
     -A INPUT -i public -j REJECT --reject-with icmp-port-unreachable
     -A FORWARD -i mgmt -o public -m state --state RELATED,ESTABLISHED -j ACCEPT
     -A FORWARD -i public -o mgmt -j ACCEPT
     -A FORWARD -j ACCEPT
     COMMIT
     ```

   - настроить сохранение и восстановление правил (информация с официального источника "Операционная Система Astra Linux Special Edition РУСБ.10015-01 очередное обновление 1.6"):
     - создать файл **vim /etc/network/if-post-down.d/iptables** со следующим содержанием:

       ```
       #!/bin/sh
       touch /etc/iptables.rules
       chmod 640 /etc/iptables.rules
       ```

```
iptables-save > /etc/iptables.rules
exit 0
```

- создать файл **vim /etc/network/if-pre-up.d/iptables** со следующим содержанием:

```
#!/bin/sh
iptables-restore < /etc/iptables.rules
exit 0
```

- сделать созданные файлы исполняемыми, выполнив в терминале команду:

```
sudo chmod +x /etc/network/if-post-down.d/iptables
sudo chmod +x /etc/network/if-pre-up.d/iptables
```

- запустить созданные файлы в заданном порядке:

```
/etc/network/if-pre-up.d/iptables

/etc/network/if-post-down.d/iptables
```

- убедиться, что в файле **/etc/iptables.rules** присутствуют записи созданные ранее:

```
cat /etc/iptables.rules
```

3. Установить пакеты:

```
apt install ca-certificates
apt update
apt install docker.io telnet net-tools inetutils-telnet rsync wget
openvswitch-switch apt-transport-https curl haproxy libssl1.1 bridge-utils
ethtool python2.7 bash-completion ipmitool
```

4. Дополнительно поставить пакет "libssl1.0" <Post_Libs.tar>.
5. Настройка Docker для работы с registry:
   - создать файл "**/etc/docker/daemon.json**" и добавить строку:

```
{ "insecure-registries":["registry-1:5000"] }
```

   - перезапустить Docker:

```
systemctl restart docker
```

6. Добавить настройку в конфигурацию systemd:

```
echo "DefaultTasksMax=infinity" >> /etc/systemd/system.conf
```

7. Настроить NTP:
   - в файл **/etc/systemd/timesyncd.conf** прописать строки:

```
[Time]
NTP=<ntp_ip_here>
```

- перезапустить службы:

```
systemctl daemon-reload
systemctl enable systemd-timesyncd.service
systemctl restart systemd-timesyncd.service
```

- проверить работу NTP:

```
systemctl status systemd-timesyncd.service
timedatectl
```

4.5 Установка кластера управления.

1. Подготовить и заполнить System-config.yaml:
   - инструкция по заполнению System-config см. в разделе "System-config";
   - проверить в system-config название БД Mongo и порт - "mongo:27017";
   - в system-config параметр version для всех узлов задать - **astra1.7.**
2. Загрузить сборку "Базис.DynamiX" <638_decort.tar.gz> на первый контроллер в "/root/update".
3. Перейти в папку:

```
cd /root/update
```

4. Разархивировать содержимое архива:

```
tar -xvf *_decort.tar.gz
```

5. Перейти в папку:

```
cd /root/update/decort/de
```

6. Загрузить образ management:

```
docker load -i management\:3*.tar
```

7. Запустить management:

```
docker run -d --name management registry-1:5000/de/management:3* init
```

8. Скопировать system-config.yaml в management:

```
docker cp /root/system-config.yaml management:/tmp/
```

9. Зайти в консоль management:

```
docker exec -it management bash
```

10. Установить kubernetes:

```
installer --config /tmp/system-config.yaml kubernetes install
```

11. Проверить установку на других контроллерах:

```
kubectl get nodes
```

12. Установить Registry в консоли management:

```
installer --config /tmp/system-config.yaml registry install
```

13. Проверить хосты в консоли контроллера:

```
cat /etc/hosts
```

Должна появиться запись **"ip-адрес registry-1"**.
В случае, если запись "ip-адрес registry-1" не появилась выполните следующее:

- выполнить команду:

```
kubectl get svc
```

- прописать "ip-адрес registry-1" вручную на всех контроллерах в /etc/hosts:

```
echo 'ip-адрес registry-1' >> /etc/hosts
```

14. Установить mongo:

```
installer --config /tmp/system-config.yaml mongo install
```

15. Установить cluster:

```
installer --config /tmp/system-config.yaml cluster install
```

16. Дождаться окончания процесса установки в консоли и отследить состояние подов:

```
watch kubectl get pods
```

17. Убедиться, что все поды находятся в рабочем состоянии (в состоянии running).
18. Установить keepalive:
    - проверить, что в конфигурационном файле system-config заполнена секция VIP;
    - выполнить:

```
installer --config /tmp/system-config.yaml keepalive install --ipaddress <VIP>
```

19. Проверить работу портала:

```
curl https://имя_площадки:8443
```

20. Зайти на портал и инициировать диалог: “Change settings” (cloudbroker → locations → GID → Actions → Change settings). Добавить:

```
allowedports:
```

```
- 80
- 443
- 22
- 10250
- 7022
- 3080
- 3023
```

21. Чтобы зайти на портал в hosts системы, внести запись:

```
<vip> domain_name sso-domain_name defense-domain_name des-domain_name novnc-domain_name
```

4.5.1 Действия при необходимости переустановки

Действия переустановки, если ранее описанные шаги необходимо повторить заново.

1. Создать скрипт:

```
vim /root/clear_all.sh
```

2. В файл добавить строки:

```
#!/bin/bash
export DECORT_CTRL=`eval echo 10.232.2.{1..3}`
for i in ${DECORT_CTRL}; do ssh $i kubeadm reset; done
for i in ${DECORT_CTRL}; do ssh $i rm -rf /etc/kubernetes/; done
for i in ${DECORT_CTRL}; do ssh $i rm -rf /opt/decort/; done
for i in ${DECORT_CTRL}; do ssh $i rm -rf /var/decort/; done
for i in ${DECORT_CTRL}; do ssh $i systemctl stop etcd; done
for i in ${DECORT_CTRL}; do ssh $i rm -rf /var/lib/etcd/; done
for i in ${DECORT_CTRL}; do ssh $i rm -rf /etc/systemd/system/kubelet.service.d/; done
for i in ${DECORT_CTRL}; do ssh $i rm /etc/systemd/system/etcd.service; done
for i in ${DECORT_CTRL}; do ssh $i systemctl daemon-reload; done
docker stop management
docker rm management
```

3. Запустить файл:

```
bash /root/clear_all.sh
```

4. Повторить установку начиная с 3-го пункта инструкции по установке кластера управления.

4.6 Настройка ОС – CPU узлы.

1. Разместить SSH ключи для коммуникации серверов между собой.
2. Добавить настройку в конфигурацию systemd:

```
echo "DefaultTasksMax=infinity" >> /etc/systemd/system.conf
```

3. Установить пакеты:

```
apt install ca-certificates
apt update
apt install telnet net-tools inetutils-telnet rsync wget openvswitch-switch apt-transport-https curl libssl1.1 python2.7 bridge-utils ethtool bash-completion ipmitool qemu-system-x86 qemu-utils qemu-kvm qemu-user-binfmt libvirt-clients libvirt-daemon libvirt-daemon-system virtinst python-pexpect ovmf openvswitch-common genisoimage runc open-iscsi multipath-tools lvm2-lockd sanlock redis-server libmhash2
```

4. Установить пакет python-libvirt:
   - если не подключен репозиторий MAIN, то при добавлении узла будет ошибка из-за того, что не найден пакет python-libvirt. Обходное решение: вручную установить пакет python-libvirt <Post_Libs.tar>:

     ```
     dpkg -i python-libvirt_5.0.0-1_amd64.deb
     ```

   - или, если репозиторий MAIN подключен - установить пакет через apt:

     ```
     apt install  python-libvirt
     ```

5. Дополнительно поставить пакет "libssl1.0" <Post_Libs.tar>.
6. Настроить NTP (Проверка синхронизации времени будет успешна только после установки DX):
   - в файл **/etc/systemd/timesyncd.conf** прописать строки:

     ```
     [Time]
     NTP=<ntp_ip_here>
     ```

   - перезапустить службы:

     ```
     systemctl daemon-reload
     systemctl enable systemd-timesyncd.service
     systemctl restart systemd-timesyncd.service
     ```

   - проверить работу NTP:

     ```
     systemctl status systemd-timesyncd.service
     timedatectl
     ```

4.7 Подключение узлов к платформе.

1. Зайти в pod management'a:

```
kubectl exec -it $(kubectl get pod | grep management | awk '{print $1}') bash
```

2. Выполнить подключение Control узлов к платформе hostname можно указывать несколько штук через запятую и без пробелов. Более 5ти за раз не рекомендуется:

```
installer node service install --name <hostname>
```

3. Проверить наличие записи "DefaultTasksMax=infinity" в конфигурации systemd:

```
cat /etc/systemd/system.conf | grep DefaultTasksMax=infinity
```

   - если записи нет, то нужно добавить настройку в конфигурацию systemd и перезапустить конфигурацию systemd manager:

```
echo "DefaultTasksMax=infinity" >> /etc/systemd/system.conf
systemctl daemon-reload
```

4.7.1 Подключение вычислительных узлов (CPU) к «Базис.DynamiX»

Внести изменения в system-config (см. п. 4 раздела "Установка кластера управления").

1. Зайти в pod management'a:

```
kubectl exec -it $(kubectl get pod | grep management | awk '{print $1}') bash
```

2. Выполнить подключение CPU узлов к платформе hostname, можно указывать несколько штук через запятую и без пробелов. Более 5-ти за раз не рекомендуется:

```
installer node service install --name <hostname>
```

3. Удалить поддержку SVM:
   - на CPU хосте редактируем файл /usr/share/libvirt/cpu_map/x86_qemu64.xml:

```
vim /usr/share/libvirt/cpu_map/x86_qemu64.xml
```

   - удалить строку 36 содержащую текст <feature name='svm'/>;
   - перезапустить libvirtd:

```
systemctl restart libvirtd
```

4. Проверяем наличие записи "DefaultTasksMax=infinity" в конфигурации systemd:

```
cat /etc/systemd/system.conf | grep DefaultTasksMax=infinity
```

   - если записи нет, то нужно добавить настройку в конфигурацию systemd и перезапускаем конфигурацию systemd manager:

```
echo "DefaultTasksMax=infinity" >> /etc/systemd/system.conf
```

```
systemctl daemon-reload
```

4.8 Подключение СХД.

4.8.1 Shared - ISCSI.

Настройка ОС на CPU узлах

1. Отредактировать файл /etc/iscsi/iscsid.conf:
   - закомментировать 44 строку и раскомментировать 41 строку;
   - сделать тоже самое через sed:

```
sed -i 's|node.startup = manual|node.startup = automatic|' /etc/iscsi/iscsid.conf
```

2. Установить из пакетов Sanlock версии 3.8.5 <Post_Libs.tar> - **действие опциональное**:

```
dpkg -i <sanlock 3.8.5 deb packages>
```

3. Отредактировать файл /lib/systemd/systemd-wdmd:

```
vim /lib/systemd/systemd-wdmd
```

   - найти строку: . /etc/rc.d/init.d/functions и заменить её на . /lib/lsb/init-functions:

```
. /lib/lsb/init-functions
```

   - сделать тоже самое через sed:

```
sed -i 's|/etc/rc.d/init.d/functions|/lib/lsb/init-functions|' /lib/systemd/systemd-wdmd
```

2. Перезапустить сервисы:

```
systemctl daemon-reload
systemctl enable sanlock lvmlockd wdmd
systemctl restart sanlock lvmlockd wdmd
```

3. Добавить конфигурационный файл для multipath для СХД (если имеется/необходим) - **действие опциональное**:
   - добавьте конфигурационный файл;
   - перезапустите сервис:

```
systemctl restart multipath-tools multipathd
```

4. Multipathd: включить автоматическое обнаружение multipath лунов:
   - открыть файл /etc/multipath.conf на редактирование:

```
vim /etc/multipath.conf
```

   - добавить строки:

```
defaults {
find_multipaths yes
}
```

- перезапустить сервис:

```
systemctl restart multipathd
```

5. Подключение к СХД:
   - подключиться к СХД:

```
iscsiadm -m discovery -t st -p <ip контроллера схд>
iscsiadm -m node -L automatic
```

   - проверить работу multipath

```
multipath -ll
```

Добавление SEP в платформу

1. Подготовить конфигурационный файл для SEP (формат JSON):
   - параметры для заполнения:

`disk_max_size - максимальный размер диска ВМ в ГБ`

`edgeuser_name и edgeuser_password - параметры в данный момент не активны`

`format - всегда qcow2`

`name_prefix - ограничения: только "_" , латинские буквы, в середине "_de_"`

`multipathNum - кол-во путей до дисков (чтобы платформа проверяла потерянные пути)`

`pools - во что будут объединяться луны`

`accessAccountIds и accessResGroupIds - не заполняется`

`allocation_type - block - пустое пространство, без разделов; file - с файловой системой, где будет создаваться qcow2 файл`

`description - заполняется в свободной форме, можно указать имена лунов`

`name - название вашего пула в платформе, ограничения - только "_" , латинские буквы`

`types - диски каких типов могут создаваться в этом пуле, D - data диски, B - boot диски`

`usage_limit - какой общий объем дисков для ВМ платформа может создать в этом пуле в ГБ (обычно указывается суммарный объем всех лунов)`

`stripes - возможность разворачивать создаваемые диски по лунам (не обязательное поле)`

```
wwns - тут просто указываем wwn-ы

protocol - параметр в данный момент не активен, scsi(для FC) или iscsi
```

- пример конфигурационного файла - системным луном для global lock (Этот лун запросить отдельно, размер не менее 2 Гб. На него будет повешен global lock. Диски машин в нём создаваться не будут), обязательно указывать его первым в списке пулов:

```
{
  "disk_max_size": 2000,
  "edgeuser_name": "user_name",
  "edgeuser_password": "password",
  "format": "raw",
  "multipathNum": 2,
  "housekeeping_settings": {
    "disk_del_queue": {
      "purge_attempts_threshold": 20
    }
  },
  "name_prefix": "test_de_shared_",
  "pools": [
    {
      "accessAccountIds": [],
      "accessResGroupIds": [],
      "allocation_type": "block",
      "description": "Description",
      "name": "de_global_lock",
      "system": "true",
      "types": [
        "D",
        "B"
      ],
      "usage_limit": 99999999,
      "wwns": [
        "60060e8012d25f005040d25f0000120c"
      ]
    },
    {
      "accessAccountIds": [],
      "accessResGroupIds": [],
      "allocation_type": "block",
      "description": "Description",
      "name": "first_pool",
      "types": [
        "D",
        "B"
      ],
      "usage_limit": 99999999,
```

```
      "wwns": [
        "60060e8012d25f005040d25f00000a16"
      ]
    },
    {
      "accessAccountIds": [],
      "accessResGroupIds": [],
      "allocation_type": "block",
      "description": "shared_lun",
      "name": "pool_de_wwn",
      "stripes": 4,
      "types": [
        "D",
        "B"
      ],
      "usage_limit": 2000,
      "wwns": [
        "60060e8012d25f005040d25f00001202",
        "60060e8012d25f005040d25f00001203",
        "60060e8012d25f005040d25f00001204",
        "60060e8012d25f005040d25f00001205"
      ]
    },
    {
      "accessAccountIds": [],
      "accessResGroupIds": [],
      "allocation_type": "file",
      "description": "shared_lun",
      "name": "pool_de_add",
      "types": [
        "D",
        "B"
      ],
      "usage_limit": 2000,
      "wwns": [
        "60060e8012d25f005040d25f0000120a",
        "60060e8012d25f005040d25f00001213"
      ]
    },
    {
      "accessAccountIds": [],
      "accessResGroupIds": [],
      "allocation_type": "block",
      "description": "shared_lun",
      "name": "pool_de_test",
      "types": [
```

```
        "D",
        "B"
      ],
      "usage_limit": 30,
      "wwns": [
        "60060e8012d25f005040d25f00001217"
      ]
    }
  ],
  "protocol": "scsi"
}
```

2. Завести на платформу shared_sep (использование API не обязательно, можно добавлять через web-интерфейс):
   - перейти в метод api cloudbroker__sep - Create SEP object - Try it out - заполнить форму:
     - gid - grid платформы;
     - name - SharedSEP (может быть любым);
     - sep_type - SHARED - в случае "shared_sep" (так же может быть DORADO, TATLIN, HITACHI и прочие в случае, если поддерживаются драйвера с СХД);
     - description - описание (опционально);
     - config - конфигурационный файл (вставляет одной строкой в поле);
     - provider_nids и consumer_nids - менять не нужно;
     - enable - false, (менять не нужно, включается после удачного заведения);
   - далее:
     - в методе api cloudbroker__sep - Enable SEP - Try it out - указать id SharedSEP - Execute;
     - в методе api cloudbroker__sep - Add consumer nodes to SEP parameters - заполнить sepID и node ID - Execute (в первый раз добавляется только одну node).

4.8.2 Driver TATLIN - ISCSI.

Настройка ОС на CPU Узлах

1. Настроить Storage сеть и убедиться, что IP адрес(а) СХД для передачи данных доступны из этой сети.
2. Отредактировать файл /etc/iscsi/iscsid.conf:
   - закомментировать 44 строку и раскомментировать 41;
   - альтернативно сделать тоже через sed:

```
sed -i 's|node.startup = manual|node.startup = automatic|' /etc/iscsi/iscsid.conf
```

3. Отредактировать файл /lib/systemd/systemd-wdmd:

```
vim /lib/systemd/systemd-wdmd
```

   - найти строку: . /etc/rc.d/init.d/functions и поменять её на . /lib/lsb/init-functions:

```
. /lib/lsb/init-functions
```

   - альтернативно сделать через sed:

```
sed -i 's|/etc/rc.d/init.d/functions|/lib/lsb/init-functions|' /lib/systemd/systemd-wdmd
```

4. Старт/перезапуск сервисов:

```
systemctl daemon-reload
systemctl enable wdmd
systemctl restart wdmd
```

5. Добавить конфигурационный файл для multipath для СХД (если имеется/необходим) - действие опциональное.
   a. добавить конфигурационный файл и перезапустить сервис:

```
systemctl restart multipath-tools multipathd
```

6. Multipathd: Включить автоматическое обнаружение multipath лунов:
   - открыть файл /etc/multipath.conf на редактирование:

```
vim /etc/multipath.conf
```

   - добавить строки:

```
defaults {
find_multipaths yes
}
```

   - перезапуск сервиса:

```
systemctl restart multipathd
```

Необходимые пререквизиты для создания SEP

- IP адрес менеджмента СХД;
- логин/пароль с полным доступом к СХД;
- доступ из сети Public до менеджмента СХД;
- доступ к менеджмента СХД по https с точки вашего подключения к инфраструктуре (для того чтобы иметь возможность подключиться к СХД и посмотреть/настроить необходимые параметры);
- логин/пароль для iSCSI Discovery;
- имя созданного заказчиком StoragePool на СХД - его имя можно посмотреть, подключившись к менеджменту через https
- добавление SEP в платформу.

Добавление SEP в платформу

1. Подготовить конфигурационный файл для SEP (формат JSON):
   - параметры для заполнения

```
API_URLs - адрес подключения к управлению СХД (должен быть доступ из сети Public)
disk_max_size - максимальный размер диска ВМ в ГБ
edgeuser_name - логин для ISCSI Discovery. Задается в СХД Settings -> System Settings -> iSCSI Discovery (Authentication: OneWay)
edgeuser_password - пароль для ISCSI Discovery
format - всегда raw
```

`hostGroupName` - Имя хост группы которая создастся сама на СХД когда будете создавать SEP в платформе

`mgmt_name` - Логин подключения к управлению СХД

`mgmt_password` - Пароль подключения к управлению СХД

`name_prefix` - префикс имен для создаваемых лунов на СХД. ограничения: только "_" , латинские буквы

`name` - имя Storage Poll (Заказчик создает заранее на СХД)

`types` - диски каких типов могут создаваться в этом пуле, D - data диски, B - boot диски

`usage_limit` - какой общий объем storage pool в GB

`ports` - имена портов. Посмотреть можно в СХД: Settings -> Ports Configuration (указать все с Role: Data I/O)

- пример конфигурационного файла:

```
{
  "API_URLs": [
    "ssh://10.40.17.79"
  ],
  "disk_max_size": 40000,
  "edgeuser_name": "pd40-user",
  "edgeuser_password": "pd40pass",
  "format": "raw",
  "hostGroupName": "pd40under-dx",
  "hosts": {},
  "housekeeping_settings": {
    "disk_del_queue": {
      "purge_attempts_threshold": 20
    }
  },
  "mgmt_password": "basis2023",
  "mgmt_user": "basis",
  "model": "Tatlin",
  "multipathNum": 16,
  "name_prefix": "pd40under-dx-",
  "pools": [
    {
      "name": "undercloud",
      "types": [
        "B",
        "D",
        "ANY"
      ],
```

```
      "usage_limit": 74520
    }
  ],
  "ports": [
    "p30",
    "p31",
    "p40",
    "p41",
    "p50",
    "p51",
    "p60",
    "p61"
  ],
  "protocol": "iscsi",
  "simple_clone": false
}
```

- завести на платформу SEP: SYSTEM → Storage EndPoints → +(Create a new SEP):
  - Grid - выбрать из списка;
  - Tame - к примеру TAT01 (может быть любым, согласовывается с заказчиком);
  - SEP Type - TATLIN;
  - description - поле опциональное;
  - Config - конфигурационный файл для SE3 подготовленный ранее, вставить в одну строку в поле;
  - provider_nids и consumer_nids - не менять;
  - enable - false, (не менять, включается после удачного заведения).
- включить созданный SEP;
- добавить или подключить CPU хосты к SEP: Add Consumer Node.

Редактирование уже созданного SEP

После того как создается SEP, конфиг SEP хранимый в платформе изменяется и дополняется системой. Поэтому если выгрузить его через API, он будет отличается от того, что использовали при создании SEP.

Правильный путь редактирования настроек SEP, если это потребовалось:

- выгрузить конфигурационный файл SEP через API /cloudbroker/sep/getConfig;
- отредактировать полученный конфигурационный файл;
- загрузить обновленный конфигурационный файл в SEP через API /cloudbroker/sep/configInsert.

4.9 System-config.

Цель данного раздела - правильное заполнение "system-config.yaml".

Если нужно заново вложить system-config в k8s

```
#сделать бэкап kubectl get -o yaml cm system-config > system-config.yaml

#сделать правки онлайн kubectl edit cm system-config

#или исправить файл, потом вложить в k8s kubectl apply -f system-config.yaml
```

Если system-config поломался при редактировании из kubectl, его можно починить следующим способом:

```
kubectl get -o yaml cm system-config | sed -E 's/[[:space:|]]+\\n/\\n/g' |
kubectl apply -f -
```

Скопировать корневой сертификат. Как сгенерировать самоподписанный сертификат см. в разделе "Создание самоподписанного сертификата для Dynamix":

```
"certificates":
"ca": |
```

Задать имя домена на который выписан сертификат, например domain.ru и вставить сертификат домена:

```
"domain.ru":
"crt": |
```

Вставляем ключ сертификата домена:

```
"key": |
```

В блоке environment, меняем следующие строки(например портал площадки - site.domain.ru):

> "grid": - это уникальный id площадки, необходимо его согласовать в чате инсталляции.

> В описании блока ниже показаны только те параметры, которые требуют корректировки, остальные оставить без изменений.
> На этапе инсталляции нельзя указывать количество реплик "portal_replicas" и "osis_replicas" больше 1.

блок environment

```
"environment":
  "basedomain": |-      - изменить, например для адреса портала 'site.domain.ru', это домен 'domain.ru'
    domain.ru
  "customer_dns":      - изменить, перечисляем dns сервера
    "ns1": |-
      10.0.16.36
    "ns2": |-
      10.0.16.38
  "dns":               - изменить, перечисляем dns сервера, тоже что и предыдущее
    "ns1": |-
      10.0.16.36
    "ns2": |-
      10.0.16.38
  "grid":              - изменить, уникальный id площадки
    "id": !!int |-
      199
  "osis_mem_limit": |-
    3
  "osis_replicas": |-
    1
  "prometheus_mem_limit": |-
    3
  "portal_replicas": |-
    1
  "ssl":
    "decs3o": |-       - изменить, например для адреса портала 'site.domain.ru', это домен 'domain.ru'.
      domain.ru
    "defense": |-
      domain.ru
    "des": |-
      domain.ru
    "novnc": |-
      domain.ru
```

```
    "ovs": |-
      domain.ru
    "root": |-
      domain.ru
  "subdomain": |-      - изменить субдомен на свой, например для адреса портала 'site.domain.ru' - это 'site'
    site
```

В блоке network, прописать сети и VIP согласно ip плану.

В блоке nodes, описать сети для каждой конкретного node, описание начинается с - "backplane":
Описание контроллер node с пояснениями:

```
- "backplane":          - backplane адрес node
    "ipaddress": |-
      10.199.32.1/24
  "fallback":           - public адрес node
    "gateway": |-
      10.0.12.73
    "ipaddress": |-
      10.0.12.74
  "gateway-management": - gw_mgmt адрес node
    "ipaddress": |-
      10.199.0.1/19
   "ip-lsb": !!int |-    - последний октет ip адреса в сетях backplane, mgmt, gw_mgmt, vxbackend
    1
  "ipmi":               - ipmi адрес node
    "ipaddress": |-
      10.0.15.61/24
    "macaddress": |-    - mac интерфейса ipmi
      a0:c5:f2:11:1c:ce
    "password": |-      - пароль ipmi
      password
    "username": |-      - пользователь ipmi
      admin
  "management":         - mgmt адрес node
    "ipaddress": |-
      10.199.33.1/24
    "macaddress": |-
      1c:34:da:5d:ed:a0 - mac интерфейса mgmt
  "name": |-            - hostname node. Указывать без доменного суффикса
    oep-ctrl-01
  "roles":              - роль node (controller)
  - |-
    controller
  "vxbackend":          - vxbackend адрес node
    "ipaddress": |-
```

```
      10.240.0.1/16
  "osversion":          - версия ОС node ubuntu 16 - 16|ubuntu 18 - 18)
    "18"
```

Описание cpu node с пояснениями:

```
- "backplane":          - backplane адрес node
    "ipaddress": |-
      10.199.32.11/24
  "gateway-management": - gw_mgmt адрес node
    "ipaddress": |-
      10.199.0.11/19
   "ip-lsb": !!int |-    - последний октет ip адреса в сетях backplane, mgmt, gw_mgmt, vxbackend
    11
  "ipmi":               - ipmi адрес node
    "ipaddress": |-
      10.0.15.64/23
    "macaddress": |-
      a0:c5:f2:11:1b:0e - mac интерфейса ipmi
    "password": |-      - пароль ipmi
      password
    "username": |-      - пользователь ipmi
      admin
  "management":         - gw_mgmt адрес node
    "ipaddress": |-
      10.199.33.11
    "macaddress": |-
      B8:CE:F6:47:53:B6 - mac интерфейса mgmt
  "name": |-            - hostname node. Указывать без доменного суффикса
    oep-cpu-01
   "roles":              - роль узла , если схд != ovs то роль CPU узла всегда cpu_des (cpu|cpu_des|storage)
  - |-
    cpu_des
  "vxbackend":          - vxbackend адрес node
    "ipaddress": |-
      10.240.0.11/16
  "osversion":          - версия ОС node (ubuntu 16 - 16|ubuntu 18 - 18)
    "18"
```

В конце system-config есть ssh ключ его нужно положить на все 3 контроллера.

4.10 Создание самоподписанного сертификата для Dynamix.

Уровень: L1-2

```
#Зайти на хост
ssh -p 17223 root@<IP-адрес площадки>

#Правка конфигурационного файла
cd /etc/ssl/DECS/new/vpn/
vim openssl.cnf

#В самом низу будет форма перечислений
#Прописать туда все пять поддоменов платформы с нужным базовым доменом

[alt_names]
DNS.1 = iaas.<базовый домен>
DNS.2 = novnc-iaas.<базовый домен>
DNS.3 = sso-iaas.<базовый домен>
DNS.4 = defense-iaas.<базовый домен>
DNS.5 = des-iaas.<базовый домен>

#Запустится процесс
openssl req -new -sha256 -nodes -newkey rsa:2048 -keyout iaas.gkh.local.key
-out iaas.gkh.local.csr -config openssl.cnf

#В процессе будут выходить вопросы, можно оставлять предложенные дефолты, там где FQDN -
выставить www.ENV_NAME.BASEDOMAIN, например выше было www.iaas.gkh.local

#Подписать:
openssl ca -out ИМЯ_ПЛОЩАДКИ.crt -config openssl.cnf -in iaas.gkh.local.csr

#На выходе получите два файла, содержащие ключ и сертификат
iaas.<базовый домен>.key
iaas.<базовый домен>.crt

#Скопировать текстом или по scp

#Может понадобится RootCA cert, он есть вот тут:
/etc/ssl/DECS/new/vpn/demoCA/rootCA.crt
```

5 Обновление программного модуля.

Обновление ПО производится специалистами Службы Технической поддержки по следующему алгоритму:

1. Оповещение Клиента о начале обновления:

> Начало обновления платформы с <current_version> до версии <update_version> на площадке <имя площадки>.

2. Создание сессии:

```
screen -S update
```

В случае проблем со связью переподключение к сессии:

```
screen -r update
```

5.1 Подготовка

В зависимости от версии ОС этой nodes.

1. Проверка свободного места в корневом разделе (нужно не менее 70GB):

```
df -h /
```

2. Проверка состояния k8s:

```
kubectl get pods
```

3. Состояние подов фиксируется, при проблемах восстанавливается работоспособность:

```
kubectl get nodes
```

4. Проверка состояния узлов (все узлы должны быть в Enabled или Restrikted).

5. Загрузить ***build*** на контроллер (один из способов ***wget***)и распаковать:

6. Скачать последний актуальный build из репозитория.

7. Создаем директорию , если она ранее не была создана и удалить старые сборки:

```
cd /root/update/
wget https://colba.decs.online/index.php/s/44YApcw2Q4mBBjJmk -o
<release>.tar.gz
tar -xvf <release>.tar.gz
```

6. Docker. Загрузить в registry:

```
docker load -i decort/de/management\:3.8.X.tar

docker push registry-1:5000/de/management:3.8.X
```

7. Проверить:

```
docker images
```

8. Изменить image tag в deployment management:

```
kubectl edit deploy management
image: registry-1:5000/de/management:x.x.x - новая версия
watch kubectl get pods - дождаться перезапуска пода mgmt
```

5.2 Установка обновления

1. Зайти в под mgmt:

```
kubectl get pods
kubectl exec -it management-xxx bash
```

2. Прописать переменную `DECORT_MNT_PATH`

3. Запустить обновление кластера:

```
installer cluster update
```

4. Установить патчи. Действие актуально если возникли какие то проблемы и принято решение отложить обновление nodes.

5. Запустить обновление узлов (список узлов лучше уточнить зайдя на портале в разделе "*Phisical nodes*"):

```
installer node service update --name <имя узла/узлов, как указано в system-config>
```

5.3 Установка патчей

1. Проверка наличия патчей для новой версии:

```
ls patches/
```

2. Выйти из mgmt.

3. При наличии патчей для роли master выполнить их из пода agentcontroller:

```
kubectl get pods
kubectl exec -it agentcontroller-xxx bash
```

4. Проверить директорию для нашей версии:

```
/opt/jumpscale7/apps/agentcontroller/jumpscripts/cloudbroker/upgrades/x.x.x
```

5. Применить патчи для роли master:

```
jspython path/to/1_patch_file.py
```

6. При наличии других ролей (cpu, storagenode) подключиться к таким nodes и выполнить:

```
js
acl = j.clients.agentcontroller.get()
acl.executeJumpscript("digitalenergy", "ИМЯ_СКРИПТА", gid="GID", role="ROLE",
all=True, wait=True)
```

7. Заменить *ИМЯ_СКРИПТА* на правильное имя скрипта патча, *GID* - grid id номер площадки, *Role* - роль node (проверить в пути к патчу).

8. Сообщить Клиенту о завершении обновления:

> Обновление платформы до *<update_version>* на площадке *<имя площадки>* успешно завершено.

5.4 Очистка "регистри" от неиспользуемых образов

После обновления площадки и всех проверок можно почистить локальный регистри.

1. Войти в под management:

```
kubectl exec -it management-xxx bash
```

2. Запустить garbage collector:

```
installer registry cleanup
```

В результате произойдет очистка /var/decort/registry от всех неиспользуемых образов.