

VM	Network Device	Network	VLAN	IP	Gateway
rtr-cod	net1	178.207.179.0/29	-	178.207.179.4	178.207.179.1
	net2	172.16.1.0/30		172.16.1.1	
fw-cod	net0	172.16.1.0/30		172.16.1.2	
	net1	192.168.10.0/24	100	192.168.10.254	172.16.1.1
		192.168.20.0/24	200	192.168.20.254	
		192.168.30.0/24	300	192.168.30.254	
		192.168.40.0/24	400	192.168.40.254	
		192.168.50.0/24	500	192.168.50.254	
sw1-cod	SVI	192.168.30.0/24	300	192.168.30.1	192.168.30.254
sw2-cod	SVI	192.168.30.0/24	300	192.168.30.2	192.168.30.254
cli-cod	net0	192.168.40.0/24	400	192.168.40.40	192.168.40.254
srv1-cod	net0	192.168.10.0/24	100	192.168.10.1	192.168.10.254
	net1	192.168.20.0/24	200	192.168.20.1	
srv2-cod	net0	192.168.10.0/24	100	192.168.10.2	192.168.10.254
	net1	192.168.20.0/24	200	192.168.20.2	
sip-cod	net0	192.168.50.0/24	500	192.168.50.50	192.168.50.254
admin-cod	net0	192.168.30.0/24	300	192.168.30.30	192.168.30.254
rtr-a	net1	178.207.179.24/29	-	178.207.179.28	178.207.179.25
	net2	172.20.10.0/24	100	172.20.10.254	
		172.20.20.0/24	200	172.20.20.254	
		172.20.30.0/24	300	172.20.30.254	
sw1-a	SVI	172.20.30.0/24	300	172.20.30.1	172.20.30.254
sw2-a	SVI	172.20.30.0/24	300	172.20.30.2	172.20.30.254
dc-a	net0	172.20.10.0/24	100	172.20.10.10	172.20.10.254
cli1-a	net0	172.20.20.0/24	200	172.20.20.1	172.20.20.254
cli2-a	net0	172.20.20.0/24	200	172.20.20.2	172.20.20.254

```
ecorouter>enable
ecorouter#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ecorouter(config)#hostname rtr-cod
rtr-cod(config)#ip domain-name cod.ssa2026.region
rtr-cod(config)#write memory
Building configuration...
```

```
rtr-cod(config)#
```

- Создадим интерфейс с именем isp и назначим на него IP-адрес 178.207.179.4/29, также зададим для данного интерфейса описание (description - опциональный, необязательный параметр):

```
rtr-cod(config)#interface isp
rtr-cod(config-if)#ip address 178.207.179.4/29
rtr-cod(config-if)#description "Connecting to an ISP provider"
rtr-cod(config-if)#exit
rtr-cod(config)#
```

- Создадим интерфейс с именем fw-cod и назначим на него IP-адрес 172.16.1.1/30, также зададим для данного интерфейса описание (description - опциональный, необязательный параметр):

```
rtr-cod(config)#interface fw-cod
rtr-cod(config-if)#ip address 172.16.1.1/30
rtr-cod(config-if)#description "Connecting to fw-cod"
```

```
rtr-cod(config-if)#exit
```

```
rtr-cod(config)#
```

- В режиме конфигурирования порта te0 необходимо создать service-instance с произвольным именем, например te0/isp:
 - также необходимо указать (инкапсулировать) что будет обрабатываться не тегированный трафик (untagged);
 - и указать в какой интерфейс (ранее созданный с именем isp) нужно отправлять обработанные кадры.

```
rtr-cod(config)#port te0
```

```
rtr-cod(config-port)#service-instance te0/isp
```

```
rtr-cod(config-service-instance)#encapsulation untagged
```

```
rtr-cod(config-service-instance)#connect ip interface isp
```

```
rtr-cod(config-service-instance)#exit
```

```
rtr-cod(config-port)#exit
```

```
rtr-cod(config)#
```

- В режиме конфигурирования порта te1 необходимо создать service-instance с произвольным именем, например te1/fw-cod:
 - также необходимо указать (инкапсулировать) что будет обрабатываться не тегированный трафик (untagged);
 - и указать в какой интерфейс (ранее созданный с именем fw-cod) нужно отправлять обработанные кадры.

```
rtr-cod(config-port)#service-instance te1/fw-cod
```

```
rtr-cod(config-service-instance)#encapsulation untagged
```

```
rtr-cod(config-service-instance)#connect ip interface fw-cod
```

```
rtr-cod(config-service-instance)#exit
```

```
rtr-cod(config-port)#exit
```

```
rtr-cod(config)# write memory
```

```
Building configuration...
```

```
rtr-cod(config)#
```

- В отличие от rtr-cod, на rtr-a по заданию нет никаких требований по настройке BGP, а значит для доступа в сеть Интернет, маршрут по умолчанию можно задать вручную:
 - переходим в режим конфигурации (configure terminal)
 - с помощью команды `ip route <IP_NETWORK/PREFIX> <NEXT_HOP_IP_ADDRESS>` задаём маршрут по умолчанию (шлюз)

```
rtr-a(config)#ip route 0.0.0.0/0 178.207.179.25
```

- Реализуем создание под-интерфейсов для дальнейшего маршрутизации между VLAN-ами:
 - Создаём интерфейсы с произвольными именами для каждого VLAN-а и назначаем на них IP-адреса:

```
rtr-a(config)#interface vl100
rtr-a(config-if)#ip address 172.20.10.254/24
rtr-a(config-if)#description "VLAN - SRV"
rtr-a(config-if)#exit
rtr-a(config)#
rtr-a(config)#interface vl200
rtr-a(config-if)#ip address 172.20.20.254/24
rtr-a(config-if)#description "VLAN - CLI"
rtr-a(config-if)#exit
rtr-a(config)#
rtr-a(config)#interface vl300
rtr-a(config-if)#ip address 172.20.30.254/24
rtr-a(config-if)#description "VLAN - MGMT"
rtr-a(config-if)#exit
rtr-a(config)#
```

НАСТРОЙКА ВЛАНОВ НА RTR-A

```
rtr-a(config)#port tel
rtr-a(config-port)#service-instance tel/vl100
rtr-a(config-service-instance)#encapsulation dot1q 100
rtr-a(config-service-instance)#rewrite pop 1
rtr-a(config-service-instance)#connect in
```

```

rtr-a(config-service-instance)#connect ip interface vl100
rtr-a(config-service-instance)#exit
rtr-a(config-port)#
rtr-a(config-port)#service-instance tel/vl200
rtr-a(config-service-instance)#encapsulation dot1q 200
rtr-a(config-service-instance)#rewrite pop 1
rtr-a(config-service-instance)#connect ip interface vl200
rtr-a(config-service-instance)#exit
rtr-a(config-port)#
rtr-a(config-port)#service-instance tel/vl300
rtr-a(config-service-instance)#encapsulation dot1q 300
rtr-a(config-service-instance)#rewrite pop 1
rtr-a(config-service-instance)#connect ip interface vl300
rtr-a(config-service-instance)#exit
rtr-a(config-port)#exit

```

Базовая настройка BGP:

- Запустите протокол BGP, указав нужную автономную систему, командой:
router bgp <№>:

```

rtr-cod(config)#router bgp 64500
rtr-cod(config-router)#

```

- Указать уникальный идентификатор маршрутизатора в протоколе BGP, командой: bgp router-id <IP>:

```

rtr-cod(config-router)#bgp router-id 178.207.179.4
rtr-cod(config-router)#

```

- Сконфигурируйте BGP соседство с Интернет провайдером ISP, указав адрес соседа и номер локальной AS, используя команду: neighbor <NEIGHBOR_IP> remote-as <\$>:

```

rtr-cod(config-router)#neighbor 178.207.179.1 remote-as 31133

```

Настройка туннеля GRE между устройствами:

- Создайте интерфейс туннеля с именем tunnel.0:

```
rtr-cod(config)#interface tunnel.0
```

```
rtr-cod(config-if-tunnel)#
```

- Назначьте IP адрес в соответствии с требованиями задания:

```
rtr-cod(config-if-tunnel)#ip address 10.10.10.1/30
```

```
rtr-cod(config-if-tunnel)#
```

- Задайте режим работы туннеля GRE и адресов начала (источника - rtr-cod) и конца туннеля (назначения - rtr-a):

```
rtr-cod(config-if-tunnel)#ip tunnel 178.207.179.4 178.207.179.28
```

```
mode gre
```

```
rtr-a#show interface tunnel.0
Interface tunnel.0 is up
Snmp index: 9
Ethernet address: (port not configured)
MTU: 1476
Tunnel source: 178.207.179.28
Tunnel destination: 178.207.179.4
Tunnel mode: GRE
Tunnel keepalive: disabled
NAT: no
ARP Proxy: disable
ICMP redirects on, unreachable on
IP URPF is disabled
Label switching is disabled
<UP, BROADCAST, RUNNING, NOARP, MULTICAST>
inet 10.10.10.2/30 broadcast 10.10.10.3/30
total input packets 0, bytes 0
total output packets 0, bytes 0
rtr-a#
```

Настройка dynamic PAT:

- Определяемся с внутренними (inside) и внешними (outside) интерфейсами с точки зрения технологии NAT:
 - интерфейс isp - внешний (outside);
 - интерфейс fw-cod - внутренний (inside)
- Назначаем интерфейс isp как ip nat outside:

```
rtr-cod(config)#interface isp
rtr-cod(config-if)#ip nat outside
rtr-cod(config-if)#exit
rtr-cod(config)#
```

- Назначаем интерфейс fw-cod как ip nat inside:

```
rtr-cod(config)#interface fw-cod
rtr-cod(config-if)#ip nat inside
rtr-cod(config-if)#exit
rtr-cod(config)#
```

- Создаём nat pool, чтобы указать диапазоны IP-адресов, который в дальнейшем будут попадать под правила трансляции:
 - необходимо не только указать диапазон IP-адресов из сети между rtr-cod и fw-cod
 - но и диапазоны IP-адресов из сетей:
 - 192.168.10.0/24 - vlan100
 - 192.168.30.0/24 - vlan300
 - 192.168.40.0/24 - vlan400
 - 192.168.50.0/24 - vlan500
 - т.к. в дальнейшем будет реализована маршрутизация между rtr-cod и fw-cod
 - диапазон 192.168.20.0/24 - vlan200 указывать не надо, т.к. по условиям задания
 - *"Для «cod» трафик VLAN - DATA не должен маршрутизироваться"*

```
rtr-cod(config)#ip nat pool fw-cod 172.16.1.1-172.16.1.2
rtr-cod(config)#ip nat pool vlan100 192.168.10.1-192.168.10.254
rtr-cod(config)#ip nat pool vlan300 192.168.30.1-192.168.30.254
rtr-cod(config)#ip nat pool vlan400 192.168.40.1-192.168.40.254
rtr-cod(config)#ip nat pool vlan500 192.168.50.1-192.168.50.254
rtr-cod(config)#
```

- Создать правило трансляции адресов для каждого созданного nat pool, через интерфейс, который с точки зрения NAT outside:

```
rtr-cod(config)#ip nat source dynamic inside pool fw-cod overload
interface isp
rtr-cod(config)#ip nat source dynamic inside pool vlan100 overload
interface isp
rtr-cod(config)#ip nat source dynamic inside pool vlan300 overload
interface isp
```

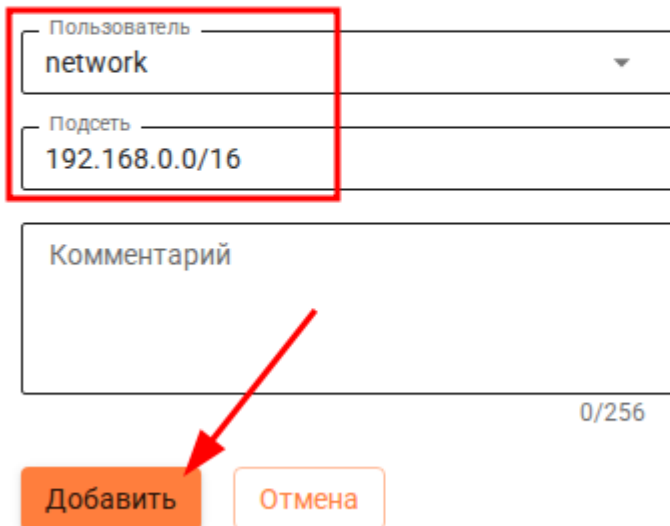
```
rtr-cod(config)#ip nat source dynamic inside pool vlan400 overload
interface isp
rtr-cod(config)#ip nat source dynamic inside pool vlan500 overload
interface isp
rtr-cod(config)#write memory
Building configuration...
```

HA FW-COD

Авторизация

НАСТРОЙКИ IP И MAC ПО ПОДСЕТЯМ

Добавление правила авторизации



Пользователь

Подсеть

Комментарий

0/256

```
ip route add 0.0.0.0/0 via 172.16.1.1
```

- Так же на данном этапе стоит добавить статические маршруты в локальные сети COD-а:
 - динамическую маршрутизацию, например OSPF между rtr-cod и fw-cod по условиям задания реализовать нельзя, т.к.
 - на rtr-cod в дальнейшем для обеспечения динамической маршрутизации между офисом «а» и «cod»
 - "Все интерфейсы, кроме туннельных, должны быть переведены в пассивный режим"
 - так же не стоит добавлять маршрут в сеть 192.168.20.0/24 (vlan200), т.к.

- "Для «cod» трафик VLAN - DATA не должен маршрутизироваться"

```
rtr-cod(config)#ip route 192.168.10.0/24 172.16.1.2
rtr-cod(config)#ip route 192.168.30.0/24 172.16.1.2
rtr-cod(config)#ip route 192.168.40.0/24 172.16.1.2
rtr-cod(config)#ip route 192.168.50.0/24 172.16.1.2
rtr-cod(config)#
```

HA RTR-A

```
rtr-a#show interface isp | grep NAT
NAT: outside
rtr-a#show interface vl100 | grep NAT
NAT: inside
rtr-a#show interface vl200 | grep NAT
NAT: inside
rtr-a#show interface vl300 | grep NAT
NAT: inside
rtr-a#
```

```
rtr-a#show run | grep pool
ip nat pool vlan100 172.20.10.1-172.20.10.254
ip nat pool vlan200 172.20.20.1-172.20.20.254
ip nat pool vlan300 172.20.30.1-172.20.30.254
rtr-a#
```

```
rtr-a#show run | grep overload
ip nat source dynamic inside-to-outside pool vlan100 overload interface isp
ip nat source dynamic inside-to-outside pool vlan200 overload interface isp
ip nat source dynamic inside-to-outside pool vlan300 overload interface isp
rtr-a#
```

```
rtr-cod(config)#router ospf 1
rtr-cod(config-router)#
```

- Сконфигурируйте OSPF идентификатор маршрутизатора (необязательный этап)
 - Используйте команду `ospf router-id <значение>`, в качестве значения укажем туннельный IP-адрес маршрутизатора:

```
rtr-cod(config-router)#router-id 10.10.10.1
rtr-cod(config-router)#
```

- Переводим все интерфейсы в пассивный режим:

```
rtr-cod(config-router)#passive-interface default
```

```
rtr-cod(config-router)#
```

- Исключаем интерфейс tunnel.0 из пассивного режима для установления соседства и дальнейшего обмена маршрутной информацией:

```
rtr-cod(config-router)#no passive-interface tunnel.0
```

```
rtr-cod(config-router)#
```

- Объявляем сети, которые будут задействованы в процессе маршрутизации:
 - импортировав (redistribute) ранее указанные статические маршруты (до локальных сетей COD-a) в процесс OSPF

```
rtr-cod(config-router)#redistribute static
```

```
rtr-cod(config-router)#network 10.10.10.0/30 area 0
```

```
rtr-cod(config-router)#network 172.16.1.0/30 area 0
```

```
rtr-cod(config-router)#exit
```

```
rtr-cod(config)#
```

- Обеспечиваем защиту протокола маршрутизации посредством парольной защиты:

```
rtr-cod(config)#interface tunnel.0
```

```
rtr-cod(config-if-tunnel)#ip ospf authentication message-digest
```

```
rtr-cod(config-if-tunnel)#ip ospf message-digest-key 1 md5
```

```
P@ssw0rd
```

```
rtr-cod(config-if-tunnel)#exit
```

```
rtr-cod(config)#write memory
```

```
Building configuration...
```

```

rtr-a#show running-config ospf
!
!
router ospf 1
  ospf router-id 10.10.10.2
  passive-interface default
  no passive-interface tunnel.0
  network 10.10.10.0/30 area 0.0.0.0
  network 172.20.10.0/24 area 0.0.0.0
  network 172.20.20.0/24 area 0.0.0.0
  network 172.20.30.0/24 area 0.0.0.0
!
interface tunnel.0
  ip ospf authentication message-digest
  ip ospf message-digest-key 1 md5 0xeb5442f9b26607db
!
rtr-a#

```

```

rtr-a#show ip ospf neighbor
Total number of full neighbors: 1
OSPF process 1 VRF(default):
Neighbor ID    Pri  State           Dead Time   Address      Interface    Instance ID
10.10.10.1     1    Full/Backup     00:00:34    10.10.10.1   tunnel.0     0
rtr-a#

```

```

rtr-a#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default

IP Route Table for VRF "default"
Gateway of last resort is 178.207.179.25 to network 0.0.0.0

S*      0.0.0.0/0 [1/0] via 178.207.179.25, isp
C       10.10.10.0/30 is directly connected, tunnel.0
O       172.16.1.0/30 [110/2] via 10.10.10.1, tunnel.0, 00:00:13
C       172.20.10.0/24 is directly connected, vl100
C       172.20.20.0/24 is directly connected, vl200
C       172.20.30.0/24 is directly connected, vl300
C       178.207.179.24/29 is directly connected, isp
O E2    192.168.10.0/24 [110/20] via 10.10.10.1, tunnel.0, 00:00:13
O E2    192.168.30.0/24 [110/20] via 10.10.10.1, tunnel.0, 00:00:13
O E2    192.168.40.0/24 [110/20] via 10.10.10.1, tunnel.0, 00:00:13
O E2    192.168.50.0/24 [110/20] via 10.10.10.1, tunnel.0, 00:00:13
rtr-a#

```

```

rtr-cod#show ip ospf neighbor

Total number of full neighbors: 1
OSPF process 1 VRF(default):
Neighbor ID      Pri   State           Dead Time   Address        Interface       Instance ID
10.10.10.2       1    Full/DR         00:00:37    10.10.10.2     tunnel.0        0
rtr-cod#show ip route

Codes: C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default

IP Route Table for VRF "default"
Gateway of last resort is 178.207.179.1 to network 0.0.0.0

B*    0.0.0.0/0 [20/0] via 178.207.179.1, isp, 00:04:57
C     10.10.10.0/30 is directly connected, tunnel.0
C     172.16.1.0/30 is directly connected, fw-cod
O     172.20.10.0/24 [110/2] via 10.10.10.2, tunnel.0, 00:00:43
O     172.20.20.0/24 [110/2] via 10.10.10.2, tunnel.0, 00:03:22
O     172.20.30.0/24 [110/2] via 10.10.10.2, tunnel.0, 00:03:22
C     178.207.179.0/29 is directly connected, isp
S     192.168.10.0/24 [1/0] via 172.16.1.2, fw-cod
S     192.168.30.0/24 [1/0] via 172.16.1.2, fw-cod
S     192.168.40.0/24 [1/0] via 172.16.1.2, fw-cod
S     192.168.50.0/24 [1/0] via 172.16.1.2, fw-cod
rtr-cod#

```

```

[root@localhost ~]# ping -c 3 172.16.1.2
PING 172.16.1.2 (172.16.1.2) 56(84) bytes of data:
64 bytes from 172.16.1.2: icmp_seq=1 ttl=62 time=130 ms
64 bytes from 172.16.1.2: icmp_seq=2 ttl=62 time=130 ms
64 bytes from 172.16.1.2: icmp_seq=3 ttl=62 time=130 ms

--- 172.16.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 129.508/129.920/130.158/0.292 ms
[root@localhost ~]# tracepath -n 172.16.1.2
1?: [LOCALHOST] pmtu 1500
1: 172.20.30.254 72.228ms
1: 172.20.30.254 21.991ms
2: 172.20.30.254 64.922ms pmtu 1476
2: 10.10.10.1 97.804ms
3: 172.16.1.2 99.717ms reached
Resume: pmtu 1476 hops 3 back 3
[root@localhost ~]#

```

НА ВИРТУАЛКАХ

```
vim /etc/net/ifaces/default/options
```

```
ovs-vsctl add-br sw1-a
```

- Проверить создание коммутатора можно с помощью команды `ovs-vsctl show`:
- Добавим интерфейс, направленный в сторону dc-a (ens20) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 100:

```
ovs-vsctl add-port sw1-a ens20 tag=100
```

- Интерфейс в сторону rtr-a (ens19) добавляем в созданный коммутатор, но настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200 и 300)

```
ovs-vsctl add-port sw1-a ens19 trunk=100,200,300
```

- Интерфейсы в стороны sw2-a (ens21, ens22) добавляем в созданный коммутатор, но настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200 и 300)

```
ovs-vsctl add-port sw1-a ens21 trunk=100,200,300
```

```
ovs-vsctl add-port sw1-a ens22 trunk=100,200,300
```

```
modprobe 8021q
```

```
echo "8021q" | tee -a /etc/modules
```

- Запускаем процесс STP на коммутаторе:
 - в режиме 802.1w (rstp)

```
ovs-vsctl set bridge sw1-a rstp_enable=true
```

```
ovs-vsctl set bridge sw1-a other_config:stp-protocol=rstp
```

- Задаём данному коммутатору наименьший приоритет:

```
ovs-vsctl set bridge sw1-a other_config:rstp-priority=0
```

```
mkdir /etc/net/ifaces/mgmt
```

- Описываем файл options для создания management интерфейса с именем mgmt:

```
vim /etc/net/ifaces/mgmt/options
```

- где:
 - TYPE - тип интерфейса (internal);
 - BOOTPROTO - определяет как будут назначаться сетевые параметры (статически);
 - CONFIG_IPV4 - определяет использовать конфигурацию протокола IPv4 или нет;

- BRIDGE - определяет к какому мосту необходимо добавить данный интерфейс;
- VID - определяет принадлежность интерфейса к VLAN;
-

```
TYPE=ovsport
BOOTPROTO=static
CONFIG_IPV4=yes
BRIDGE=sw1-a
VID=300
```

- Помимо того, что интерфейс mgmt является портом доступа (access) необходимо использовать NativeVLAN:

```
ovs-vsctl set port mgmt vlan_mode=native-untagged
```

SW2-A

```
ovs-vsctl add-br sw2-a
```

- Проверить создание коммутатора можно с помощью команды ovs-vsctl show:



- Добавим интерфейс, направленный в сторону cli1-a (ens22) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 200:

```
ovs-vsctl add-port sw2-a ens22 tag=200
```

- Добавим интерфейс, направленный в сторону cli2-a (ens21) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 200:

```
ovs-vsctl add-port sw2-a ens21 tag=200
```

- Интерфейсы в стороны sw1-a (ens19, ens20) добавляем в созданный коммутатор, но настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200 и 300)

```
ovs-vsctl add-port sw1-a ens19 trunk=100,200,300
```

```
ovs-vsctl add-port sw1-a ens20 trunk=100,200,300
```

- Проверить добавление портов в коммутатор можно с помощью команды `ovs-vsctl show`:

- Включаем модуль ядра отвечающий за тегированный трафик (802.1Q):

```
modprobe 8021q
```

```
echo "8021q" | tee -a /etc/modules
```

- Запускаем процесс STP на коммутаторе:
 - в режиме 802.1w (rstp)

```
ovs-vsctl set bridge sw2-a rstp_enable=true
```

```
ovs-vsctl set bridge sw2-a other_config:stp-protocol=rstp
```

```
mkdir /etc/net/ifaces/mgmt
```

- Описываем файл options для создания management интерфейса с именем `mgmt`:

```
vim /etc/net/ifaces/mgmt/options
```

- где:

- TYPE - тип интерфейса (internal);
- BOOTPROTO - определяет как будут назначаться сетевые параметры (статически);
- CONFIG_IPV4 - определяет использовать конфигурацию протокола IPv4 или нет;
- BRIDGE - определяет к какому мосту необходимо добавить данный интерфейс;
- VID - определяет принадлежность интерфейса к VLAN;

```
TYPE=ovsport
BOOTPROTO=static
CONFIG_IPV4=yes
BRIDGE=sw2-a
VLAN=300
```

```
ovs-vsctl set port mgmt vlan_mode=native-untagged
```

sw1-cod (alt-server):

```
ovs-vsctl add-br sw1-cod
```

- Проверить создание коммутатора можно с помощью команды `ovs-vsctl show`:



- Добавим интерфейс, направленный в сторону admin-cod (enp2s29) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 300:

```
ovs-vsctl add-port sw1-cod enp2s29 tag=300
```

- Добавим интерфейс, направленный в сторону srv2-cod (enp2s1) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 100:

```
ovs-vsctl add-port sw1-cod enp2s1 tag=100
```

- Добавим интерфейс, направленный в сторону srv2-cod (enp3s12) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 200:

```
ovs-vsctl add-port sw1-cod enp3s12 tag=200
```

- Интерфейс в сторону fw-cod (ens19) добавляем в созданный коммутатор, но настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200,300,400 и 500)

```
ovs-vsctl add-port sw1-cod ens19 trunk=100,200,300,400,500
```

```
modprobe 8021q
```

```
echo "8021q" | tee -a /etc/modules
```

- На базе интерфейсов ens21 и ens22, направленных в сторону sw2-cod, создадим bond-интерфейс в режиме active-backup:

```
ovs-vsctl add-bond sw1-cod bond0 ens21 ens22
```

```
bond_mode=active-backup
```

- Интерфейс типа bond в сторону sw2-cod (bond0) настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200,300,400 и 500)

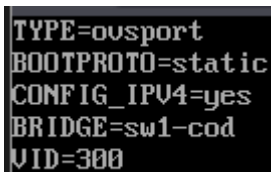
```
ovs-vsctl set port bond0 trunk=100,200,300,400,500
```

```
mkdir /etc/net/ifaces/mgmt-cod
```

- Описываем файл options для создания management интерфейса с именем mgm-codt:

```
vim /etc/net/ifaces/mgmt-cod/options
```

- где:
 - TYPE - тип интерфейса (internal);
 - BOOTPROTO - определяет как будут назначаться сетевые параметры (статически);
 - CONFIG_IPV4 - определяет использовать конфигурацию протокола IPv4 или нет;
 - BRIDGE - определяет к какому мосту необходимо добавить данный интерфейс;
 - VID - определяет принадлежность интерфейса к VLAN;



```
TYPE=ovsport
BOOTPROTO=static
CONFIG_IPV4=yes
BRIDGE=sw1-cod
VID=300
```

```
ovs-vsctl set port mgmt-cod vlan_mode=native-untagged
```

sw2-cod (alt-server):

```
ovs-vsctl add-br sw2-cod
```

- Проверить создание коммутатора можно с помощью команды ovs-vsctl show:

- Добавим интерфейс, направленный в сторону srv1-cod (ens21) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 100:

```
ovs-vsctl add-port sw2-cod ens21 tag=100
```

- Добавим интерфейс, направленный в сторону srv1-cod (ens22) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 200:

```
ovs-vsctl add-port sw2-cod ens22 tag=200
```

- Добавим интерфейс, направленный в сторону sip-cod (enp2s29) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 500:

```
ovs-vsctl add-port sw2-cod enp2s29 tag=500
```

- Добавим интерфейс, направленный в сторону cli-cod (enp3s12) в созданный коммутатор и назначим его в качестве порта доступа (access), указав принадлежность к VLAN 400:

```
ovs-vsctl add-port sw2-cod enp3s12 tag=400
```

- Проверить добавление портов в коммутатор можно с помощью команды ovs-vsctl show:

- Включаем модуль ядра отвечающий за тегированный трафик (802.1Q):

```
modprobe 8021q
```

```
echo "8021q" | tee -a /etc/modules
```

- На базе интерфейсов ens19 и ens20, направленных в сторону sw1-cod, создадим bond-интерфейс в режиме active-backup:

```
ovs-vsctl add-bond sw2-cod bond0 ens19 ens20
```

```
bond_mode=active-backup
```

- Интерфейс типа bond в сторону sw2-cod (bond0) настраиваем как магистральный (trunk) порт:
 - также разрешаем пропуск только требуемых VLAN (100,200,300,400 и 500)

```
ovs-vsctl set port bond0 trunk=100,200,300,400,500
```

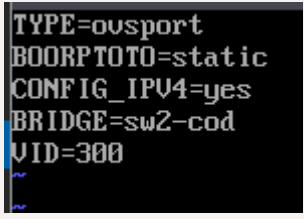
- Проверить можно с помощью команды ovs-vsctl show:

- Режим работы bond интерфейса проверить можно с помощью команды `ovs-appctl bond/show`:
- Сетевая подсистема `etcnet` будет взаимодействовать с `openvswitch`, для того чтобы корректно можно было назначить IP-адрес на интерфейс управления
 - создаём каталог для management интерфейса с именем `mgmt-cod`:

```
mkdir /etc/net/ifaces/mgmt-cod
```

- Описываем файл `options` для создания management интерфейса с именем `mgm-codt`:

```
vim /etc/net/ifaces/mgmt-cod/options
```



```
TYPE=ovsport
BOORPTOTO=static
CONFIG_IPV4=yes
BRIDGE=sw2-cod
VID=300
```

```
ovs-vsctl set port mgmt-cod vlan_mode=native-untagged
```

```
echo "nameserver 77.88.8.8" > /etc/resolv.conf
```

- Устанавливаем пакеты **freeradius** и **freeradius-utils**:

```
apt-get update && apt-get install -y freeradius freeradius-utils
```

- Включаем службу **radiusd** в автозагрузку и запускаем ее:

```
systemctl enable --now radiusd
```

- Создаем клиентов добавляя в файле **/etc/raddb/clients.conf** следующее содержимое:

- пример конфигурации файла **/etc/raddb/clients.conf** для любого клиента

```
client ALL {
    ipaddr = 0.0.0.0
    netmask = 0
    secret = P@ssw0rd
}
```

- Редактируем файл с пользователями **/etc/raddb/users** добавляя в самый конец следующее содержимое:

```
#####
# You should add test accounts to the TOP of this file! #
# See the example user "bob" above. #
#####
#user ClearText-Password := "P@ssw0rd"
#user Service-Type = Administrative-User,
#user Cisco-AVPair = "shell:roles=admin"
```

- Перезапускаем службу **radiusd**:

```
systemctl restart radiusd
```

rtr-cod (ecorouter):

- Выполняем подключение к RADIUS-серверу:

```
rtr-cod(config)#security none
rtr-cod(config)#aaa radius-server 192.168.10.1 port 1812 secret P@ssw0rd auth
rtr-cod(config)#
```

- Задаём приоритет:

```
rtr-cod(config)#aaa precedence local radius
rtr-cod(config)#write memory
Building configuration...

rtr-cod(config)#
```

- Проверить возможность входа из-под пользователя **netuser** с паролем **P@ssw0rd**:

```
rtr-cod login: netuser
Password:

User Access Verification

EcoRouterOS version Jasmine 26/12/2024 23:46:47
rtr-cod>enable
rtr-cod#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
rtr-cod(config)#
```

- Проверить вход под локальной учетной записью даже при доступности RADIUS-сервера:

```
rtr-cod login: admin
Password:

User Access Verification

EcoRouterOS version Jasmine 26/12/2024 23:46:47
rtr-cod>
```

- Проверить доступ по SSH из-под пользователя **netuser** с паролем **P@ssw0rd**:
 - например с **srv1-cod**

```
[root@srv1-cod ~]# ssh netuser@172.16.1.1
The authenticity of host '172.16.1.1 (172.16.1.1)' can't be established.
ED25519 key fingerprint is SHA256:k5nMnJ+uMSuU10B/QUzwU/CJFUsu/i46KD3nYxIOJjl.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.16.1.1' (ED25519) to the list of known hosts.
netuser@172.16.1.1's password:

User Access Verification

EcoRouterOS version Jasmine 26/12/2024 23:46:47
rtr-cod>enable
rtr-cod#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
rtr-cod(config)#
```

sw1-cod и sw2-cod (alt-server):

- Временно для возможности установки необходимых пакетов зададим публичный DNS-сервер:

```
echo "nameserver 77.88.8.8" > /etc/resolv.conf
```

```
apt-get update && apt-get install -y pam_radius
```

- Редактируем конфигурационный файл `/etc/pam_radius_auth.conf`:

```
# server[:port] shared secret timeout (s) source_ip urf
192.168.10.1 P@ssw0rd 3
#127.0.0.1 secret 3
#other-server other-secret 5 192.168.1.10 urf-blue
#12001:0db8:85a3::41:1812 other6-secret 3 12001:0db8:85a3::31 urf-red
#other-other-server other-other-secret 5 0 urf-blue
#
# having localhost in your radius configuration is a Good Thing.
#
:wq!
```

- Редактируем конфигурационный файл `/etc/pam.d/ssh`:

```
##PAM-1.0
auth required pam_userpass.so
auth sufficient pam_radius_auth.so
auth include common-login-use_first_pass
account include common-login
password include common-login
session include common-login
```

- Редактируем конфигурационный файл `/etc/pam.d/system-auth-local`:

```
##PAM-1.0
auth sufficient pam_radius_auth.so
auth include system-auth-local-only
auth include system-auth-common
account include system-auth-local-only
account include system-auth-common
password include system-auth-local-only
password include system-auth-common
session include system-auth-local-only
session include system-auth-common
```

- Также в случае с Linux, данного пользователя необходимо создать локально:

```
useradd netuser
```

- Проверить возможность входа из-под пользователя `netuser` с паролем `P@ssw0rd`:

```
Welcome to ALT Server 11.0 (Mendeleevium)!
Hostname: sw2-cod.cod.ssa2026.region
IP: 192.168.30.2
sw2-cod login: netuser
Password:
[netuser@sw2-cod ~]$
```

- Проверить доступ по SSH из-под пользователя `netuser` с паролем `P@ssw0rd`:
 - например с `srv1-cod`

```
Last login: Mon Nov 17 13:00:16 MSK 2025 on tty1
[root@srv1-cod ~]# ssh netuser@192.168.30.2
The authenticity of host '192.168.30.2 (192.168.30.2)' can't be established.
ED25519 key fingerprint is SHA256:A8TkadFGEbXdLvLNzB5Bgm2Jtf6gOUdSOsYusrCg05w.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.30.2' (ED25519) to the list of known hosts.
netuser@192.168.30.2's password:
Last login: Mon Nov 17 13:06:04 2025
[netuser@sw2-cod ~]$
```

apt-get install tzdata

rtr-cod и rtr-a (ecorouter):

- Устанавливаем корректный часовой пояс:

```
rtr-cod(config)#ntp timezone utc+3
rtr-cod(config)#
```

- Устанавливаем синхронизацию времени с 100.100.100.100:

```
rtr-cod(config)#ntp server 100.100.100.100
rtr-cod(config)#write memory
Building configuration...

rtr-cod(config)#
```

- Проверить текущий часовой пояс можно командой `show ntp timezone` из режима администрирования (**enable**):

```
rtr-cod#show ntp timezone
System Time zone: Europe/Moscow
rtr-cod#
```

- Проверить адреса ntp-серверов для синхронизации можно командой `show ntp status` из режима администрирования (**enable**):

```
rtr-cod#show ntp status
Status Description
* best
+ sync
- failed
? unknown

-----
Status | VR name | Server | Stratum | Delay | Version | Offset | Last |
-----
* | default | 100.100.100.100 | 5 | 0.2069 | 4 | 0.8753 | 1:17 |
rtr-cod#
```

sw1-cod, sw2-cod, cli-cod, srv1-cod, srv2-cod, admin-cod, sw1-a, sw2-a, dc-a, cli1-a и cli2-a (alt-server, alt-workstation):

- Устанавливаем корректный часовой пояс:

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

- Редактируем конфигурационный файл `/etc/chrony.conf`:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (https://www.pool.ntp.org/join.html)
#pool pool.ntp.org iburst
pool 100.100.100.100 iburst
# Record the rate at which the system clock gains/losses time.
```

- Перезапускаем службу `chronyd`:

```
systemctl restart chronyd
```

- Проверяем с каким сервером синхронизировалось время с помощью команды `chronyc tracking`:

srv1-cod (alt-server):

- Установим пакет bind и bind-utils:

```
apt-get install -y bind bind-utils
```

- Отредактируем конфигурационный файл /etc/bind/options.conf:

```
options {
    version "unknown";
    directory "/etc/bind/zone";
    dump-file "/var/run/named/named_dump.db";
    statistics-file "/var/run/named/named.stats";
    recursing-file "/var/run/named/named.recursing";
    secroots-file "/var/run/named/named.secroots";

    // disables the use of a PID file
    pid-file none;

    /*
     * Oftenly used directives are listed below.
     */

    listen-on { any; };
    listen-on-v6 { none; };

    /*
     * If the forward directive is set to "only", the server will only
     * query the forwarders.
     */
    forward first;
    forwarders { 100.100.100.100; };

    /*
     * Specifies which hosts are allowed to ask ordinary questions.
     */
    allow-query { any; };

    /*
     * This lets "allow-query" be used to specify the default zone access
     * level rather than having to have every zone override the global
     * value. "allow-query-cache" can be set at both the options and view
     * levels. If "allow-query-cache" is not set then "allow-recursion" is
     * used if set, otherwise "allow-query" is used if set unless
     * "recursion no;" is set in which case "none;" is used, otherwise the
     * default (localhost; localnets;) is used.
     */
    allow-query-cache { any; };

    /*
     * Specifies which hosts are allowed to make recursive queries
     * through this server. If not specified, the default is to allow
     * recursive queries from all hosts. Note that disallowing recursive
     * queries for a host does not prevent the host from retrieving data
     * that is already in the server's cache.
     */
    allow-recursion { any; };
```

- Добавить в конфигурационный файл /etc/bind/rfc1912.local информацию о файлах зон прямого и обратного просмотра:

- Добавить в конфигурационный файл `/etc/bind/rfc1912.local` информацию о файлах зон прямого и обратного просмотра:

```
zone "cod.ssa2026.region" {
    type master;
    file "cod.ssa2026.region";
    allow-transfer { 172.20.10.10; };
};

zone "168.192.in-addr.arpa" {
    type master;
    file "168.192.in-addr.arpa";
    allow-transfer { 172.20.10.10; };
};

zone "office.ssa2026.region" {
    type forward;
    forward only;
    forwarders { 172.20.10.10; };
};

zone "20.172.in-addr.arpa" {
    type forward;
    forward only;
    forwarders { 172.20.10.10; };
};
```

- Скопировать файл шаблона для зоны прямого просмотра:

```
cp /etc/bind/zone/localhost /etc/bind/zone/cod.ssa2026.region
```

- Выдать права на файл зоны прямого просмотра:

```
chown root:named /etc/bind/zone/cod.ssa2026.region
```

- Привести файл `/etc/bind/zone/cod.ssa2026.region` зоны прямого просмотра к следующему виду:

```
$TTL      1D
@IN      SOA  cod.ssa2026.region. root.cod.ssa2026.region. (
                2025110500      ; serial
                12H              ; refresh
                1H               ; retry
                1W               ; expire
                1H               ; ncache
        )
IN      NS   cod.ssa2026.region.
IN      A    192.168.10.1
rtr-cod IN      A    172.16.1.1
fw-cod  IN      A    192.168.10.254
sw1-cod IN      A    192.168.30.1
sw2-cod IN      A    192.168.30.2
cli-cod IN      A    192.168.40.40
srv1-cod IN     A    192.168.10.1
srv2-cod IN     A    192.168.10.2
sip-cod IN      A    192.168.50.50
admin-cod IN     A    192.168.30.30
monitoring IN    CNAME srv1-cod.cod.ssa2026.region.
```

- Скопировать файл шаблона для зоны обратного просмотра:

```
cp /etc/bind/zone/localhost /etc/bind/zone/168.192.in-addr.arpa
```

- Выдать права на файл зоны прямого просмотра:

```
chown root:named /etc/bind/zone/168.192.in-addr.arpa
```

- Привести файл `/etc/bind/zone/168.192.in-addr.arpa` зоны обратного просмотра к следующему виду:

```
$TTL      1D
@         IN      SOA     cod.ssa2026.region. root.cod.ssa2026.region. (
                                2025110500      : serial
                                12H               : refresh
                                1H               : retry
                                1W               : expire
                                1H               : mcache
                                )
254.10    IN      NS      cod.ssa2026.region.
1.30      IN      PTR     fu-cod.cod.ssa2026.region.
2.30      IN      PTR     su1-cod.cod.ssa2026.region.
40.40     IN      PTR     su2-cod.cod.ssa2026.region.
1.10      IN      PTR     cli-cod.cod.ssa2026.region.
2.10      IN      PTR     srv1-cod.cod.ssa2026.region.
50.50     IN      PTR     srv2-cod.cod.ssa2026.region.
30.30     IN      PTR     sip-cod.cod.ssa2026.region.
~         IN      PTR     admin-cod.cod.ssa2026.region.
```

- Включить и добавить в автозагрузку службу `bind`:

```
systemctl enable --now bind
```

- Задаём DNS-сервер и домен поиска:

```
cat <<EOF > /etc/net/iface/ens19/resolv.conf
search cod.ssa2026.region
nameserver 127.0.0.1
EOF
```

- Перезагружаем сервер:

```
reboot
```

- Проверяем записи типа `A`:

```
(root@srv1-cod ~)# host fu-cod
fu-cod.cod.ssa2026.region has address 192.168.10.254
(root@srv1-cod ~)# host su1-cod
su1-cod.cod.ssa2026.region has address 192.168.30.1
(root@srv1-cod ~)# host su2-cod
su2-cod.cod.ssa2026.region has address 192.168.30.2
(root@srv1-cod ~)# host cli-cod
cli-cod.cod.ssa2026.region has address 192.168.40.40
(root@srv1-cod ~)# host srv1-cod
srv1-cod.cod.ssa2026.region has address 192.168.10.1
(root@srv1-cod ~)# host srv2-cod
srv2-cod.cod.ssa2026.region has address 192.168.10.2
(root@srv1-cod ~)# host sip-cod
sip-cod.cod.ssa2026.region has address 192.168.50.50
(root@srv1-cod ~)# host admin-cod
admin-cod.cod.ssa2026.region has address 192.168.30.30
(root@srv1-cod ~)# _
```

sw1-cod и sw2-cod (alt-server):

- Задаём в качестве DNS-сервера **srv1-cod**:

```
cat <<EOF > /etc/net/ifaces/mgmt-cod/resolv.conf
search cod.ssa2026.region
nameserver 192.168.10.1
EOF
```

- Перезагружаем службу **network**:

```
systemctl restart network
```

- Проверить:

```
[root@sw1-cod ~]# cat /etc/resolv.conf
# Generated by resolvconf
# Do not edit manually, use
# /etc/net/ifaces/<interface>/resolv.conf instead.
search cod.ssa2026.region
nameserver 192.168.10.1
[root@sw1-cod ~]# ping -c3 cod.ssa2026.region
PING cod.ssa2026.region (192.168.10.1) 56(84) bytes of data:
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=1 ttl=63 time=2.28 ms
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=2 ttl=63 time=2.97 ms
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=3 ttl=63 time=2.93 ms

--- cod.ssa2026.region ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/ndev = 2.281/2.724/2.966/0.313 ms
[root@sw1-cod ~]# ping -c3 ya.ru
PING ya.ru (77.88.55.242) 56(84) bytes of data:
64 bytes from ya.ru (77.88.55.242): icmp_seq=1 ttl=49 time=90.4 ms
64 bytes from ya.ru (77.88.55.242): icmp_seq=2 ttl=49 time=95.9 ms
64 bytes from ya.ru (77.88.55.242): icmp_seq=3 ttl=49 time=94.9 ms

--- ya.ru ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/ndev = 94.059/96.375/98.362/1.460 ms
[root@sw1-cod ~]#
```

srv2-cod (alt-server):

- Задаём в качестве DNS-сервера **srv1-cod**:

```
cat <<EOF > /etc/net/ifaces/ens20/resolv.conf
search cod.ssa2026.region
nameserver 192.168.10.1
EOF
```

- Перезагружаем службу **network**:

```
systemctl restart network
```

- Проверить:

```
[root@srv2-cod ~]# cat /etc/resolv.conf
# Generated by resolvconf
# Do not edit manually, use
# /etc/net/ifaces/<interface>/resolv.conf instead.
search cod.ssa2026.region
nameserver 192.168.10.1
[root@srv2-cod ~]# ping -c3 cod.ssa2026.region
PING cod.ssa2026.region (192.168.10.1) 56(84) bytes of data:
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=1 ttl=64 time=1.41 ms
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=2 ttl=64 time=2.49 ms
64 bytes from srv1-cod.cod.ssa2026.region (192.168.10.1): icmp_seq=3 ttl=64 time=2.10 ms

--- cod.ssa2026.region ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/ndev = 1.413/2.001/2.493/0.446 ms
[root@srv2-cod ~]# ping -c3 ya.ru
PING ya.ru (77.88.44.242) 56(84) bytes of data:
64 bytes from ya.ru (77.88.44.242): icmp_seq=1 ttl=47 time=44.4 ms
64 bytes from ya.ru (77.88.44.242): icmp_seq=2 ttl=47 time=43.4 ms
64 bytes from ya.ru (77.88.44.242): icmp_seq=3 ttl=47 time=41.5 ms

- ya.ru ping statistics -
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/ndev = 41.497/43.898/44.352/1.191 ms
[root@srv2-cod ~]#
```

вариант реализации:

dc-a (alt-server):

- Временно для возможности установки необходимых пакетов зададим публичный DNS:

```
echo "nameserver 77.88.8.8" > /etc/resolv.conf
```

- Устанавливаем пакеты **task-samba-dc**, **bind** и **bind-utils**:

```
apt-get update && apt-get install -y task-samba-dc bind bind-utils
```

- Если при установке системы в настройках сети было указано полное имя домена (напр
- Для решения проблемы необходимо закомментировать все строки в файле **/etc/bind/lc**

```
//include "/etc/bind/rfc1912.conf";  
// Consider adding the 1918 zones here,  
// if they are not used in your organization.  
//     include "/etc/bind/rfc1918.conf";  
// Add other zones here
```

- Отключите **chroot**:

```
control bind-chroot disabled
```

- Отключите **KRB5RCACHETYPE**:

```
echo 'KRB5RCACHETYPE="none"' >> /etc/sysconfig/bind
```

- Подключите плагин **BIND_DLZ**:

```
echo 'include "/var/lib/samba/bind-dns/named.conf";' >> /etc/bind/named.conf
```

```

dump-file "/var/run/named/named_dump.db";
statistics-file "/var/run/named/named.stats";
recursing-file "/var/run/named/named.recursing";
secroots-file "/var/run/named/named.secroots";

tkey-gssapi-keytab "/var/lib/samba/bind-dns/dns.keytab";
minimal-responses yes;

// disables the use of a PID file
pid-file none;

/*
 * Oftenly used directives are listed below.
 */

listen-on { any; };
listen-on-v6 { none; };

/*
 * If the forward directive is set to "only", the server will only
 * query the forwarders.
 */
forward first;
forwarders { 100.100.100.100; };

/*
 * Specifies which hosts are allowed to ask ordinary questions.
 */
allow-query { any; };

/*
 * This lets "allow-query" be used to specify the default zone access
 * level rather than having to have every zone override the global
 * value. "allow-query-cache" can be set at both the options and view
 * levels. If "allow-query-cache" is not set then "allow-recursion" is
 * used if set, otherwise "allow-query" is used if set unless
 * "recursion no;" is set in which case "none;" is used, otherwise the
 * default (localhost; localnets;) is used.
 */
allow-query-cache { any; };

/*
 * Specifies which hosts are allowed to make recursive queries
 * through this server. If not specified, the default is to allow
 * recursive queries from all hosts. Note that disallowing recursive
 * queries for a host does not prevent the host from retrieving data
 * that is already in the server's cache.
 */
allow-recursion { any; };

```

- в раздел **logging** добавьте строку:
 - **logging** — подавление предупреждений о «lame servers»

```
logging {
    category lame-servers { null; };
    // The default debug channel has the special property that it only
    // produces output when the server's debug level is non-zero. It
    // normally writes to a file called named.run in the server's working
    // directory.
```

- Восстановить к начальному состоянию Samba:

```
rm -f /etc/samba/smb.conf
```

```
rm -f /etc/samba/smb.conf
```

```
rm -rf /var/cache/samba
```

```
mkdir -p /var/lib/samba/sysvol
```

- Интерактивное создание домена:

```
samba-tool domain provision
```

- в качестве **DNS backend** указать **BIND9_DLZ**

```
[root@dc-a ~]# samba-tool domain provision
Realm (OFFICE.SSA2026.REGION1):
Domain (OFFICE1):
Server Role (dc, member, standalone) [dc]:
DNS backend (SAMBA_INTERNAL, BIND9_FLATFILE, BIND9_DLZ, NONE) [SAMBA_INTERNAL]: BIND9_DLZ
Administrator password:
Retype password: _
```

- результат успешного создания домена в интерактивном режиме:

```
INFO 2025-11-19 09:22:10.503 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #2414: ghd/gua root key added with guid 322d45c0-0b6b-432b-c0
91-9bec195a63ef
INFO 2025-11-19 09:22:10.506 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #2425: A Kerberos configuration suitable for Samba AD has been
generated at /usr/lib/samba/private/krb5.conf
INFO 2025-11-19 09:22:10.507 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #2427: Merge the contents of this file with your system krb5.co
nf or replace it with this one. Do not create a symlink!
INFO 2025-11-19 09:22:10.744 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #492: Once the above files are installed, your Samba AD server
will be ready to use.
INFO 2025-11-19 09:22:10.746 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #499:
INFO 2025-11-19 09:22:10.746 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #498:
INFO 2025-11-19 09:22:10.748 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #499:
INFO 2025-11-19 09:22:10.749 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #500:
INFO 2025-11-19 09:22:10.749 pid:1376 /usr/lib64/samba-dc/python3.12/samba/provision/_init_.py #501:
[root@dc-a ~]#
```

Server Role:	active directory domain controller
Hostname:	dc-a
NetBIOS Domain:	OFFICE
DNS Domain:	office.ssa2026.region
DOMAIN SID:	S-1-5-21-2856879486-8942876818-1752411318

- Включить в автозагрузку службы **samba** и **bind**, также запустить их:

- Включить в автозагрузку службы **samba** и **bind**, также запустить их:

```
systemctl enable --now samba
```

```
systemctl enable --now bind
```

- При создании домена Samba автоматически генерирует корректный файл **krb5.conf** для домена в каталоге **/etc/**:
 - Можно просто заменить этим файлом файл, находящийся в каталоге **/etc/**:

```
cp /var/lib/samba/private/krb5.conf /etc/krb5.conf
```

- Перезагрузить службу **samba**:

```
systemctl restart samba
```

- Просмотр общей информации о домене можно выполнить с помощью команды **samba-tool domain info**:

```
[root@dc-a ~]# samba-tool domain info 127.0.0.1
Forest           : office.ssa2026.region
Domain           : office.ssa2026.region
Netbios domain   : OFFICE
DC name          : dc-a.office.ssa2026.region
DC netbios name  : DC-A
Server site      : Default-First-Site-Name
Client site      : Default-First-Site-Name
[root@dc-a ~]#
```

- Просмотр предоставляемых служб можно выполнить с помощью команды **smbclient -L localhost -Uadmi**:
 - **netlogon** и **sysvol** создаются автоматически и необходимы для работы контроллера домена

```
[root@dc-a ~]# smbclient -L localhost -U administrator
Password for [OFFICE\administrator]:

      Sharename      Type      Comment
      -----
      sysvol         Disk
      netlogon       Disk
      IPC$           IPC       IPC Service (Samba 4.21.9-alt1)
SMB1 disabled -- no workgroup available
[root@dc-a ~]#
```

- Проверка конфигурации DNS:
 - проверка наличия **nameserver 127.0.0.1** в **/etc/resolv.conf**:

```
[root@dc-a ~]# cat /etc/resolv.conf
# Generated by resolvconf
# Do not edit manually, use
# /etc/net/ifaces/<interface>/resolv.conf instead.
search office.ssa2026.region
nameserver 127.0.0.1
[root@dc-a ~]#
```

- проверка имён хостов **_kerberos._udp.**:

```
[root@dc-a ~]# host -t SRV _kerberos._udp.office.ssa2026.region.
_kerberos._udp.office.ssa2026.region has SRV record 0 100 88 dc-a.office.ssa2026.region.
[root@dc-a ~]#
```

```
root@dc-a ~]# host -t SRV _ldap._tcp.office.ssa2026.region.  
_ldap._tcp.office.ssa2026.region has SRV record 0 100 389 dc-a.office.ssa2026.region.  
root@dc-a ~]#
```

- проверка имён хостов "адрес хоста":

```
root@dc-a ~]# host -t A dc-a.office.ssa2026.region.  
dc-a.office.ssa2026.region has address 172.20.10.10  
root@dc-a ~]#
```

- Проверка Kerberos-аутентификации (имя домена должно быть в верхнем регистре):

```
root@dc-a ~]# kinit administrator@OFFICE.SSA2026.REGION  
Password for administrator@OFFICE.SSA2026.REGION:  
Warning: Your password will expire in 41 days on Wed Dec 31 09:22:12 2025  
root@dc-a ~]#
```

- Просмотр полученного билета:

```
root@dc-a ~]# klist  
Ticket cache: FILE:/tmp/krb5cc_0  
Default principal: administrator@OFFICE.SSA2026.REGION  
  
Valid starting Expires Service principal  
11/19/25 09:49:33 11/19/25 19:49:33 krbtgt/OFFICE.SSA2026.REGION@OFFICE.SSA2026.REGION  
renew until 11/20/25 09:45:24  
root@dc-a ~]#
```

cli1-a (alt-workstation):

- Установить пакет **admc**:

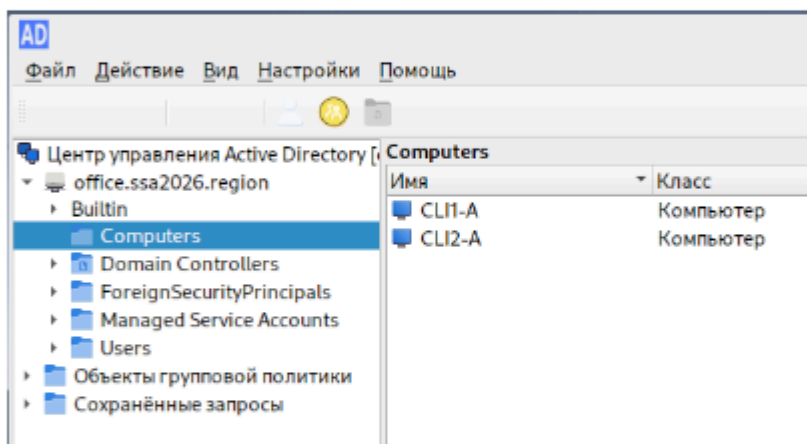
```
apt-get install -y admc
```

- Запуск **ADMC** осуществляется из меню запуска приложений: пункт **Системные** → **ADMC** или из ко
- Для запуска ADCM необходимо предварительно получить ключ Kerberos для администратора до

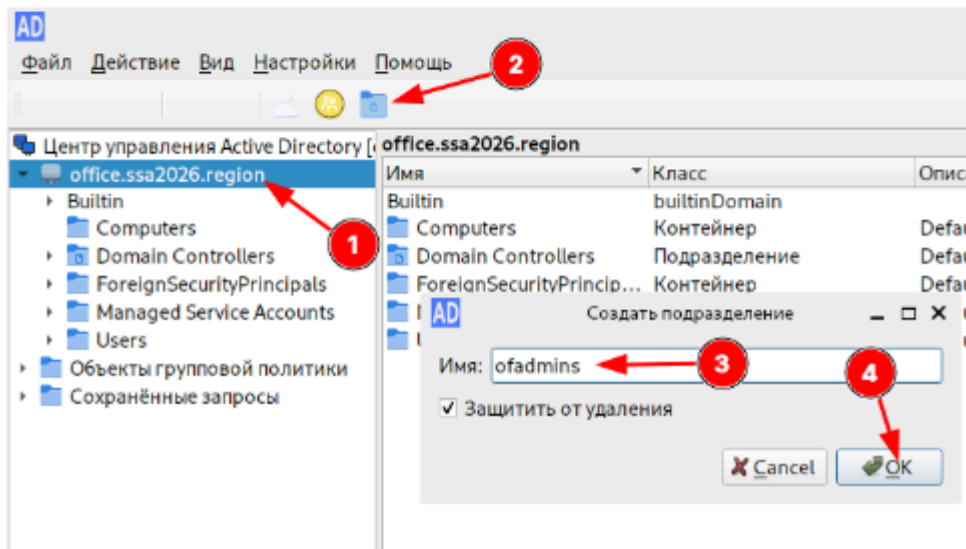
```
user@cli1-a: /home/user

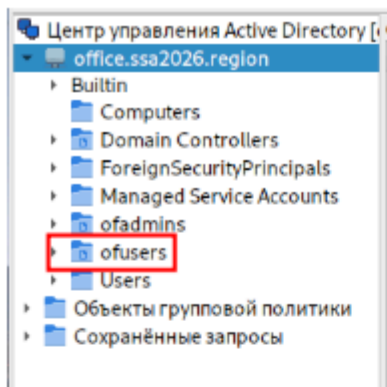
[user@cli1-a ~]$ kinit administrator@OFFICE.SSA2026.REGION
Password for administrator@OFFICE.SSA2026.REGION:
Warning: Your password will expire in 41 days on Cp 31 дек 2025 09:22:12
[user@cli1-a ~]$
```

- Результат успешного запуска **admc**:

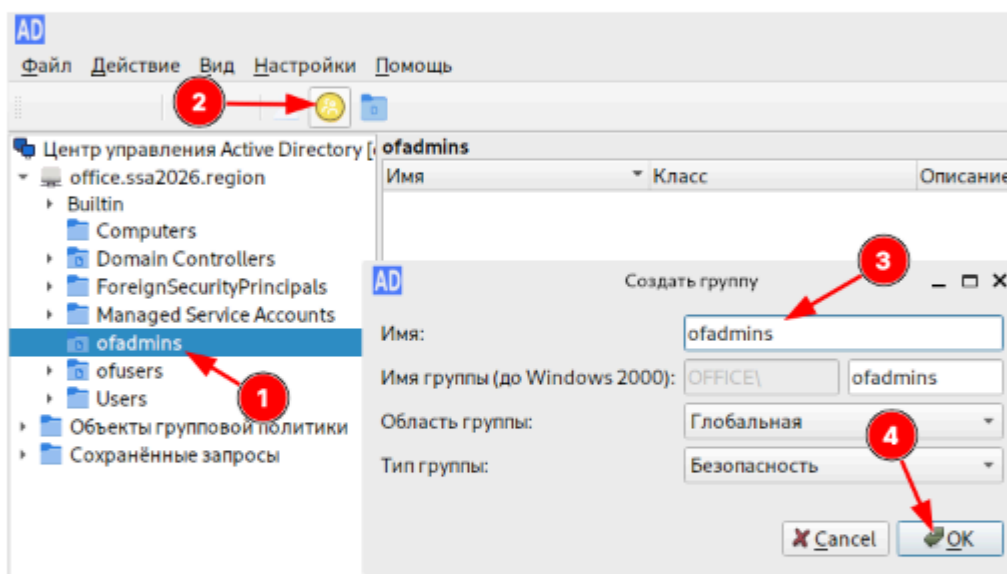


- Создать **ofadmins**:

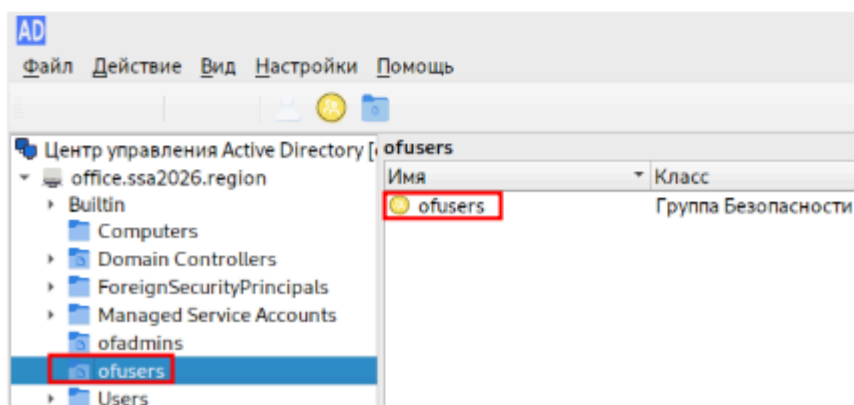




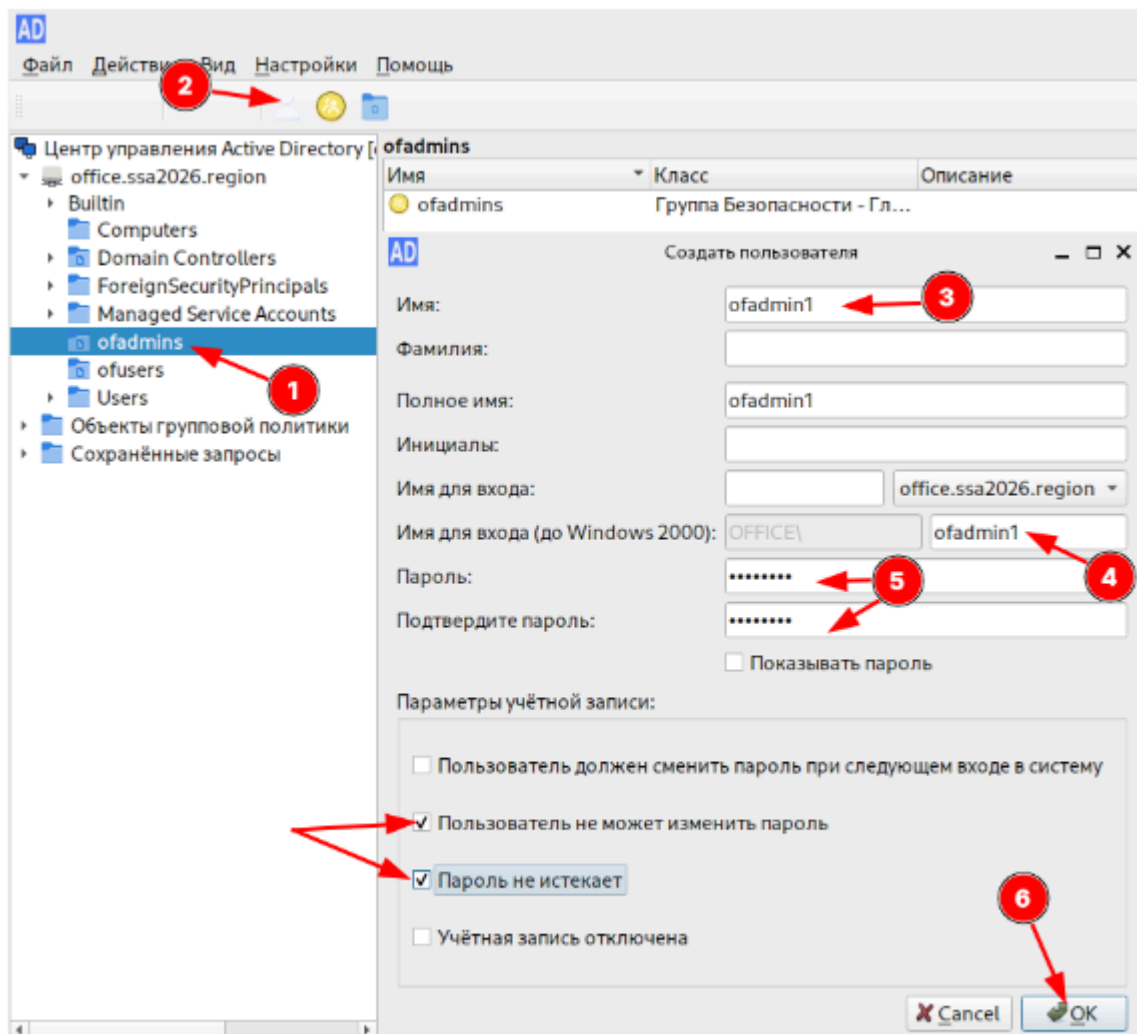
- Создать группу **ofadmins** в подразделение **ofadmins**:



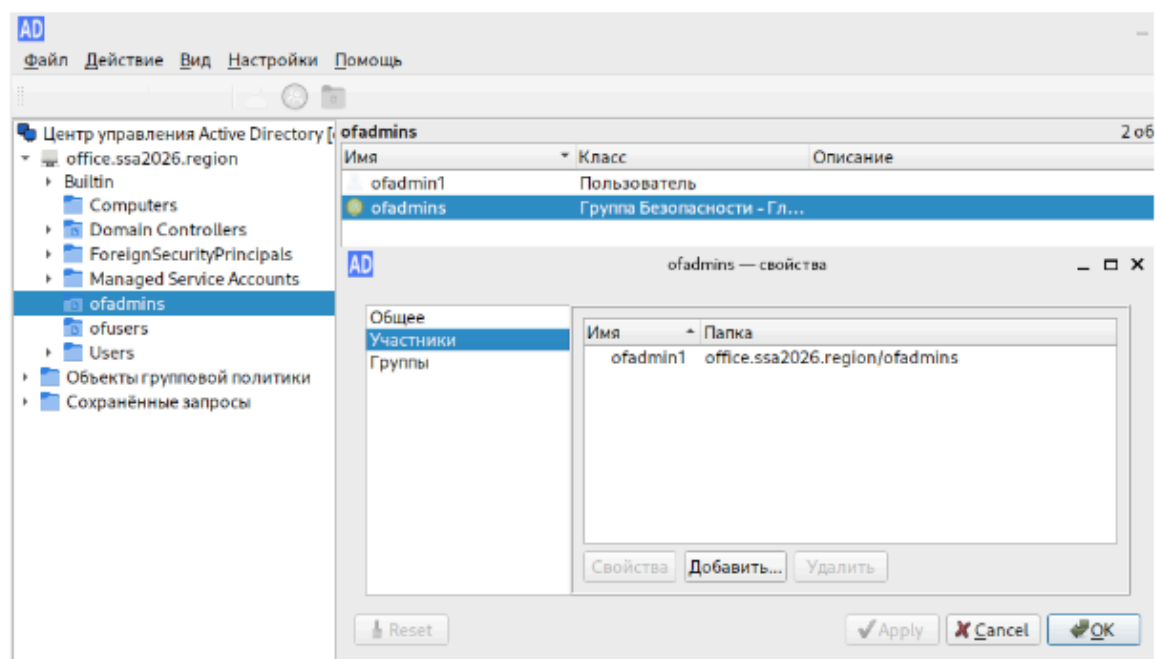
- Создать группу **ofusers** в подразделение **ofusers**:



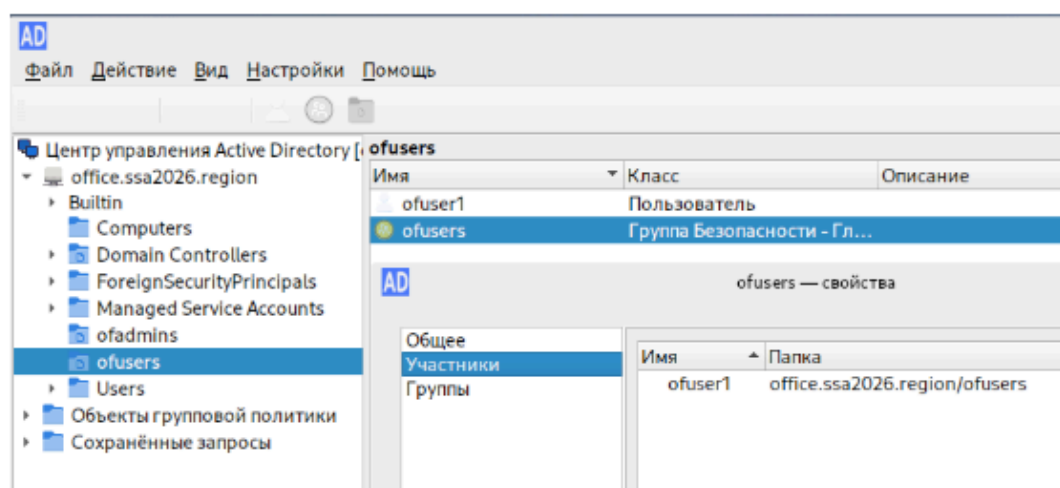
- Создать пользователя **ofadmin1** в подразделение **ofadmins**:



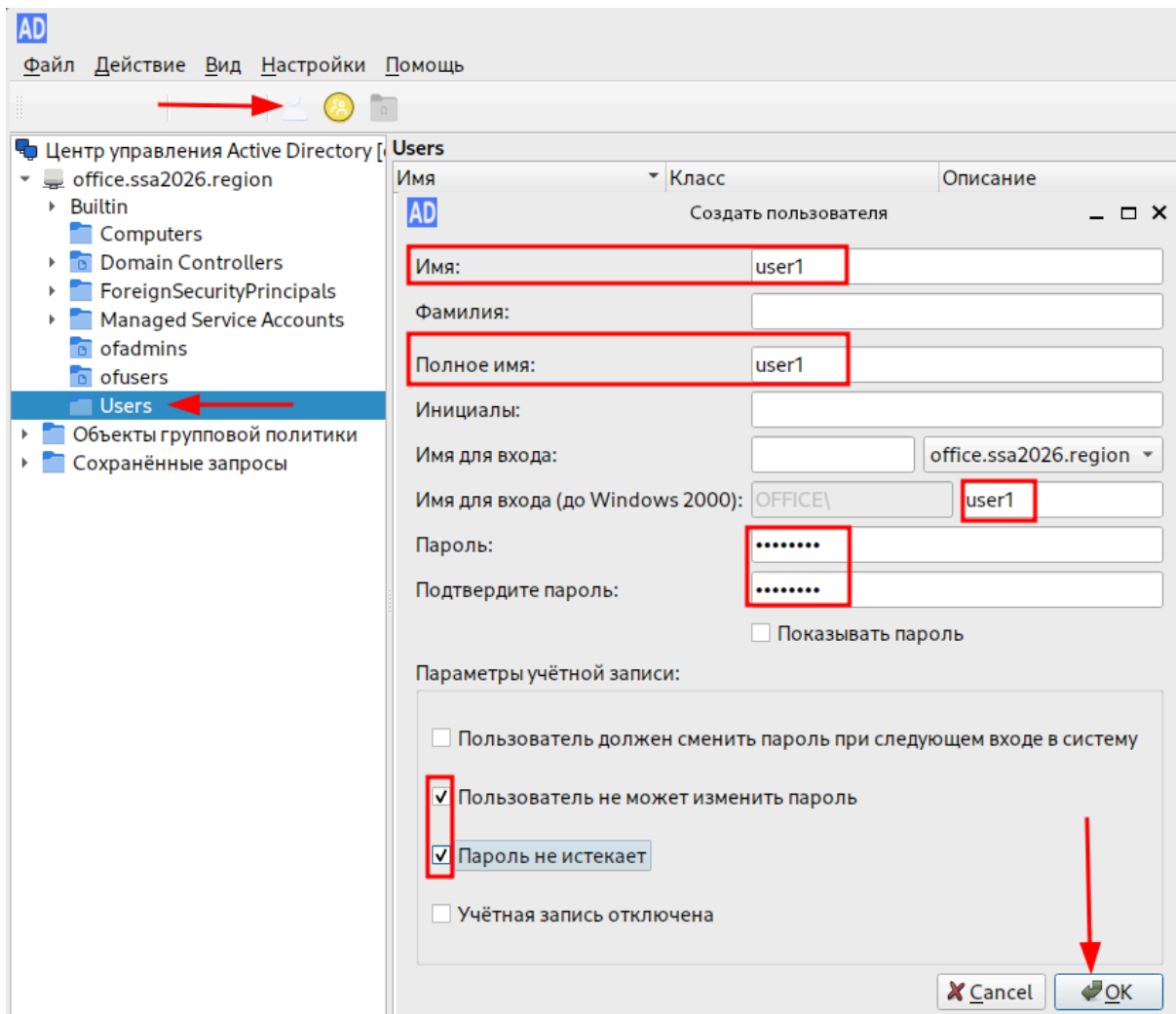
- Добавить созданного пользователя в группу ofadmins:



- Создать пользователя **ofuser1** в подразделение **ofusers**
 - Добавить созданного пользователя в группу **ofusers**



- Создать пользователя **user1**:



cli1-a и cli2-a (alt-workstation):

- Необходимо установить пакет **gpupdate**:

```
apt-get install -y gpupdate
```

- Включить модуль групповых политик:

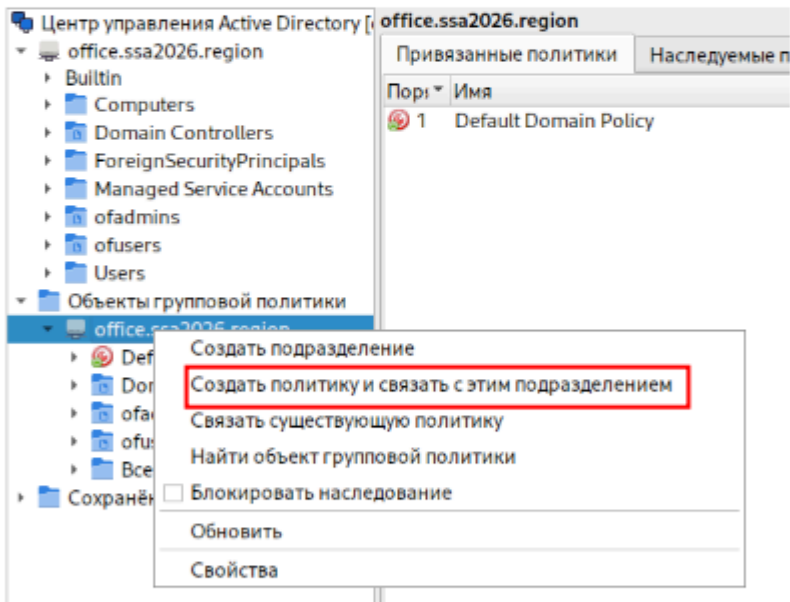
```
gpupdate-setup enable
```

cli1-a (alt-workstation):

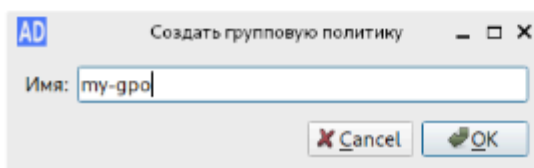
- Установим пакет **gpiui**, для редактирования настроек клиентской конфигурации:

```
apt-get install -y gpiui
```

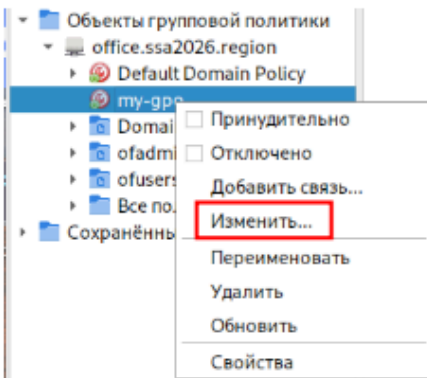
- В оснастке ADMC - переходим в раздел **Объекты групповой политики** -> ПКМ по **office.ssa2026.region**



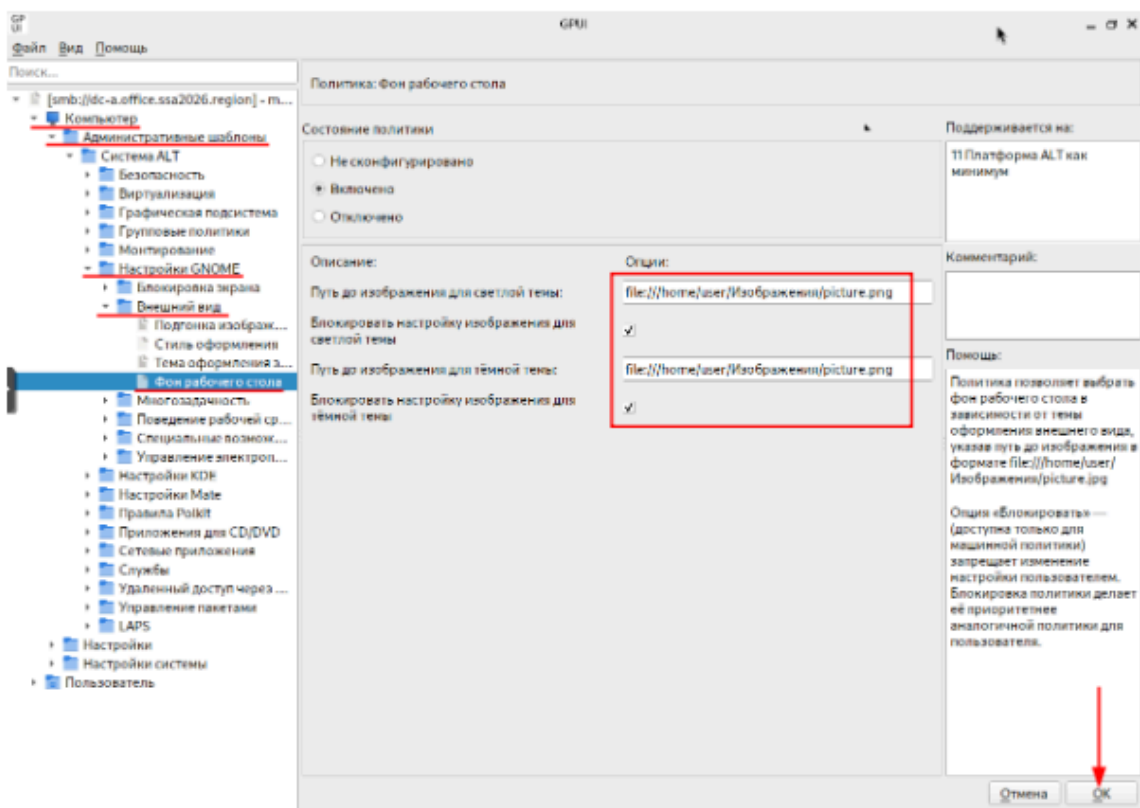
- Задаём имя и нажимаем **ОК**:

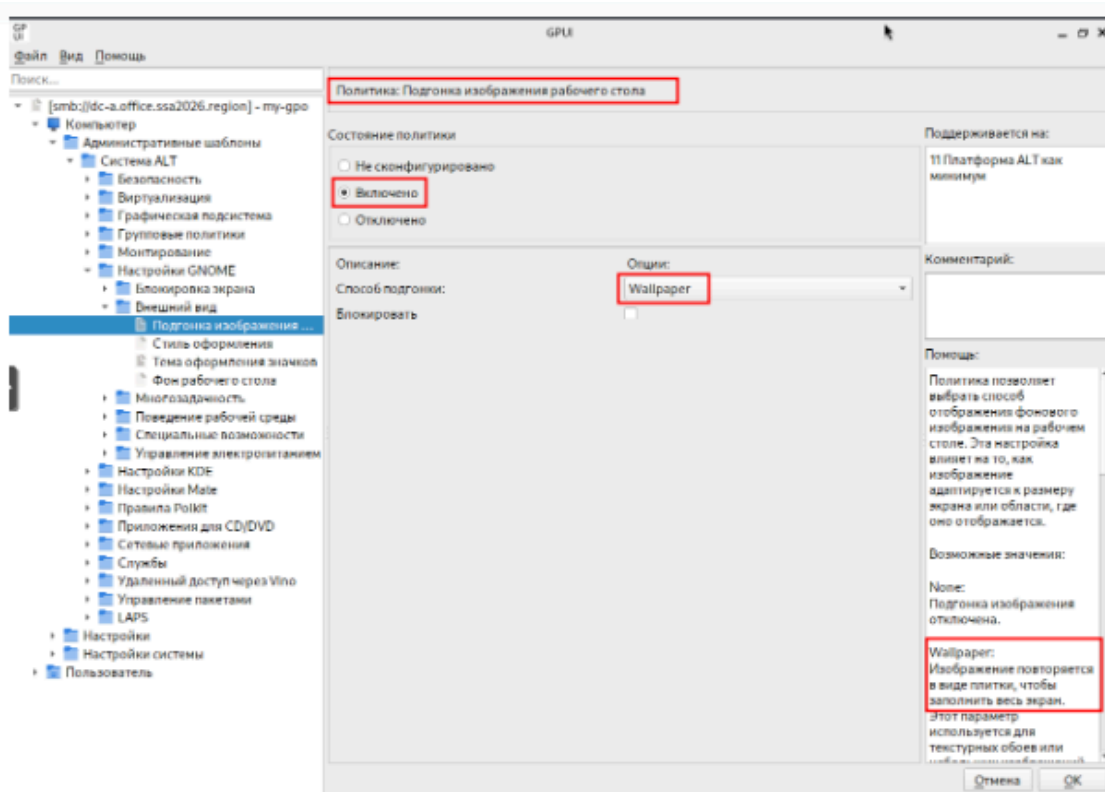


- Выбираем созданную групповую политику **ПКМ** и нажимаем **Изменить**:



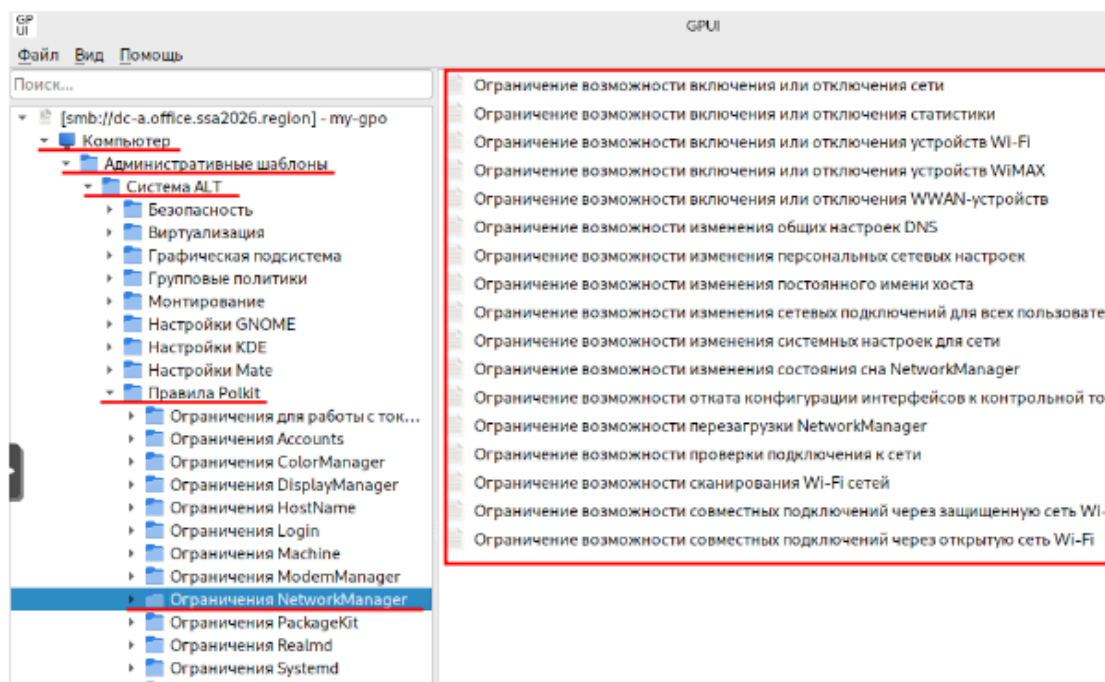
- В соответствии с требованиями задания - реализуем необходимый функционал:
 - Задаём картинку компании для рабочего стола и запрещаем её менять
 - но, чтобы на cli2-а (или же из под любого другого пользователя) картинка также была установлена, её необходи





- Запрещаем изменение сетевых настроек:

- пройтись по всему списку и выставить **Включено**, с вариантом ограничений **№** (выставить чек-бокс **Блокировать**):



dc-a (alt-server):

- Средствами **samba-tool** можно посмотреть список зон DNS:

```
(root@dc-a ~)# samba-tool dns zonelist 127.0.0.1
Password for [administrator@OFFICE.SSA2026.REGION]:
2 zone(s) found

pszZoneName      : office.ssa2026.region
Flags            : DNS_RPC_ZONE_DSINTEGRATED DNS_RPC_ZONE_UPDATE_SECURE
ZoneType         : DNS_ZONE_TYPE_PRIMARY
Version          : 50
dwDpFlags        : DNS_DP_AUTOCREATED DNS_DP_DOMAIN_DEFAULT DNS_DP_ENLISTED
pszDpFqdn        : DomainDnsZones.office.ssa2026.region

pszZoneName      : _msdcs.office.ssa2026.region
Flags            : DNS_RPC_ZONE_DSINTEGRATED DNS_RPC_ZONE_UPDATE_SECURE
ZoneType         : DNS_ZONE_TYPE_PRIMARY
Version          : 50
dwDpFlags        : DNS_DP_AUTOCREATED DNS_DP_FOREST_DEFAULT DNS_DP_ENLISTED
pszDpFqdn        : ForestDnsZones.office.ssa2026.region
(root@dc-a ~)#
```

- Средствами **samba-tool** можно посмотреть список DNS-записей в определённой зоне:

```
(root@dc-a ~)# samba-tool dns query 127.0.0.1 office.ssa2026.region @ A
Password for [administrator@OFFICE.SSA2026.REGION]:
Name=, Records=1, Children=0
  A: 172.20.10.10 (flags=600000f0, serial=38, ttl=900)
Name=_msdcs, Records=0, Children=0
Name=_sites, Records=0, Children=1
Name=_tcp, Records=0, Children=4
Name=_udp, Records=0, Children=2
Name=c111-a, Records=1, Children=0
  A: 172.20.20.1 (flags=f0, serial=39, ttl=3600)
Name=c112-a, Records=1, Children=0
  A: 172.20.20.2 (flags=f0, serial=39, ttl=3600)
Name=dc-a, Records=1, Children=0
  A: 172.20.10.10 (flags=f0, serial=1, ttl=900)
Name=DomainDnsZones, Records=0, Children=2
Name=ForestDnsZones, Records=0, Children=2
(root@dc-a ~)#
```

- Средствами **samba-tool** создадим недостающие записи типа **A** в зоне прямого просмотра:

```
samba-tool dns add 127.0.0.1 office.ssa2026.region rtr-a A 172.20.10.254 -U administrator
samba-tool dns add 127.0.0.1 office.ssa2026.region rtr-a A 172.20.20.254 -U administrator
samba-tool dns add 127.0.0.1 office.ssa2026.region rtr-a A 172.20.30.254 -U administrator
samba-tool dns add 127.0.0.1 office.ssa2026.region sw1-a A 172.20.30.1 -U administrator
samba-tool dns add 127.0.0.1 office.ssa2026.region sw2-a A 172.20.30.2 -U administrator
```

- Проверить наличие записей:

- Добавить в конфигурационный файл `/etc/bind/local.conf` информацию о файле зоны обратного просмотра:

```
//include "/etc/bind/rfc1912.conf";
// Consider adding the 1910 zones here,
// if they are not used in your organization.
// include "/etc/bind/rfc1918.conf";
// Add other zones here
zone "20.172.in-addr.arpa" {
    type master;
    file "20.172.in-addr.arpa";
    allow-transfer { 192.168.10.1; };
};

zone "cod.ssa2026.region" {
    type forward;
    forward only;
    forwarders { 192.168.10.1; };
};

zone "168.192.in-addr.arpa" {
    type forward;
    forward only;
    forwarders { 192.168.10.1; };
};
```

- Скопировать файл шаблона для зоны обратного просмотра:

```
cp /etc/bind/zone/localhost /etc/bind/zone/20.172.in-addr.arpa
```

- Выдать права на файл зоны обратного просмотра:

```
chown root:named /etc/bind/zone/20.172.in-addr.arpa
```

- Привести файл `/etc/bind/zone/20.172.in-addr.arpa` зоны обратного просмотра к следующему виду:

```
$TTL 1D
IN SOA office.ssa2026.region. root.office.ssa2026.region. (
    2025110500 ; serial
    12H ; refresh
    1H ; retry
    1W ; expire
    1H ; ncache
)
IN NS office.ssa2026.region.
254.10 IN PTR rtr-a.office.ssa2026.region.
254.20 IN PTR rtr-a.office.ssa2026.region.
254.30 IN PTR rtr-a.office.ssa2026.region.
1.30 IN PTR sw1-a.office.ssa2026.region.
2.30 IN PTR sw2-a.office.ssa2026.region.
10.10 IN PTR dc-a.office.ssa2026.region.
1.20 IN PTR cli1-a.office.ssa2026.region.
2.20 IN PTR cli2-a.office.ssa2026.region.
```

- Перезапустить службу `bind`:

```
systemctl restart bind
```

- Проверить записи типа PTR:

```
(root@dc-a ~) # host 172.20.10.254
254.10.20.172.in-addr.arpa domain name pointer rtr-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.20.254
254.20.20.172.in-addr.arpa domain name pointer rtr-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.30.254
254.30.20.172.in-addr.arpa domain name pointer rtr-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.30.1
1.30.20.172.in-addr.arpa domain name pointer sw1-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.30.2
2.30.20.172.in-addr.arpa domain name pointer sw2-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.20.1
1.20.20.172.in-addr.arpa domain name pointer cli1-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.20.2
2.20.20.172.in-addr.arpa domain name pointer cli2-a.office.ssa2026.region.
(root@dc-a ~) # host 172.20.10.10
10.10.20.172.in-addr.arpa domain name pointer dc-a.office.ssa2026.region.
(root@dc-a ~) # _
```

- Проверить несколько записей для forward зон:

```
(root@dc-a ~) # host 192.168.10.1
1.10.168.192.in-addr.arpa domain name pointer srv1-cod.cod.ssa2026.region.
(root@dc-a ~) # host 192.168.10.2
2.10.168.192.in-addr.arpa domain name pointer srv2-cod.cod.ssa2026.region.
(root@dc-a ~) # _
```

sw1-a и sw2-a (alt-server):

- Задаём в качестве DNS-сервера dc-a:

```
cat <<EOF > /etc/net/iface/mgmt/resolv.conf
search office.ssa2026.region
nameserver 172.20.10.10
EOF
```

- Перезагружаем службу network:

```
systemctl restart network
```

- Проверить:

```
(root@sw1-a ~) # cat /etc/resolv.conf
# Generated by resolvconf
# Do not edit manually, use
# /etc/net/iface/<interface>/resolv.conf instead.
search office.ssa2026.region
nameserver 172.20.10.10
(root@sw1-a ~) # ping -c3 office.ssa2026.region
PING office.ssa2026.region (172.20.10.10) 56(84) bytes of data.
64 bytes from dc-a.office.ssa2026.region (172.20.10.10): icmp_seq=1 ttl=63 time=77.2 ms
64 bytes from dc-a.office.ssa2026.region (172.20.10.10): icmp_seq=2 ttl=63 time=70.0 ms
64 bytes from dc-a.office.ssa2026.region (172.20.10.10): icmp_seq=3 ttl=63 time=73.3 ms
-- office.ssa2026.region ping statistics --
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 73.257/76.143/77.907/2.066 ms
(root@sw1-a ~) # ping -c3 ya.ru
PING ya.ru (77.88.55.242) 56(84) bytes of data.
64 bytes from ya.ru (77.88.55.242): icmp_seq=1 ttl=48 time=97.3 ms
64 bytes from ya.ru (77.88.55.242): icmp_seq=2 ttl=48 time=95.0 ms
64 bytes from ya.ru (77.88.55.242): icmp_seq=3 ttl=48 time=92.7 ms
-- ya.ru ping statistics --
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 92.650/95.265/97.336/1.947 ms
(root@sw1-a ~) # _
```

srv1-cod (alt-server):

- Создадим директорию `/var/ca`:

```
mkdir /var/ca
```

- Отредактируем конфигурационный файл `/etc/openssl/openssl.cnf`:

```
#####
[ CA_default ]
dir               = /var/ca                # Where everything is kept
certs             = $dir/certs             # Where the issued certs are kept
crl_dir           = $dir/crl               # Where the issued crl are kept
database          = $dir/index.txt         # database index file.
#unique_subject   = no                     # Set to 'no' to allow creation of
                                           # several certs with same subject.
new_certs_dir     = $dir/newcerts          # default place for new certs.

certificate       = $dir/cacert.pem        # The CA certificate
serial            = $dir/serial            # The current serial number
crlnumber         = $dir/crlnumber         # the current crl number
                                           # must be commented out to leave a V1 CRL
crl               = $dir/crl.pem           # The current CRL
private_key       = $dir/private/cakey.pem # The private key

x509_extensions   = usr_cert              # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt          = ca_default             # Subject Name options
cert_opt          = ca_default             # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions  = crl_ext

default_days      = 365                    # how long to certify for
default_crl_days  = 30                     # how long before next CRL
default_md        = sha256                 # which md to use.
preserve          = no                     # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-))
policy            = policy_anything
```

- Перейдём в директорию `/var/ca` и создадим необходимую структуру:

```
cd /var/ca
```

```
mkdir certs
```

```
mkdir crl
```

```
mkdir newcerts
```

```
mkdir private
```

```
touch index.txt
```

```
touch serial
```

```
touch crlnumber
```

```
echo 01 > serial
```

```
echo 01 > crlnumber
```

- Должна получиться следующая структура:

```
root@srv1-cod ca# pwd
/var/ca
root@srv1-cod ca# ls
certs  crl  crlnumber  index.txt  newcerts  private  serial
root@srv1-cod ca# _
```

- Создадим ключ:

```
openssl genrsa -out private/cakey.pem
```

- Создадим запрос:

```
openssl req -new -key private/cakey.pem -out cacert.csr
```

- Указав имя организации и центра сертификации:

```
root@srv1-cod ca# openssl req -new -key private/cakey.pem -out cacert.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []: IRPO
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: ssa2026
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
root@srv1-cod ca#
```

- Подписываем запрос и выпускаем сертификат сроком на 5 лет:

```
root@srv1-cod ca# openssl x509 -req -days 1825 -in cacert.csr -signkey private/cakey.pem -out cacert.pem
Certificate request self-signature ok
subject=C=RU, O=IRPO, CN=ssa2026
root@srv1-cod ca# _
```

- Разрешим доступ до srv1-cod по SSH для пользователя root

cli-cod, admin-cod, cli1-a и cli2-a (alt-workstation):

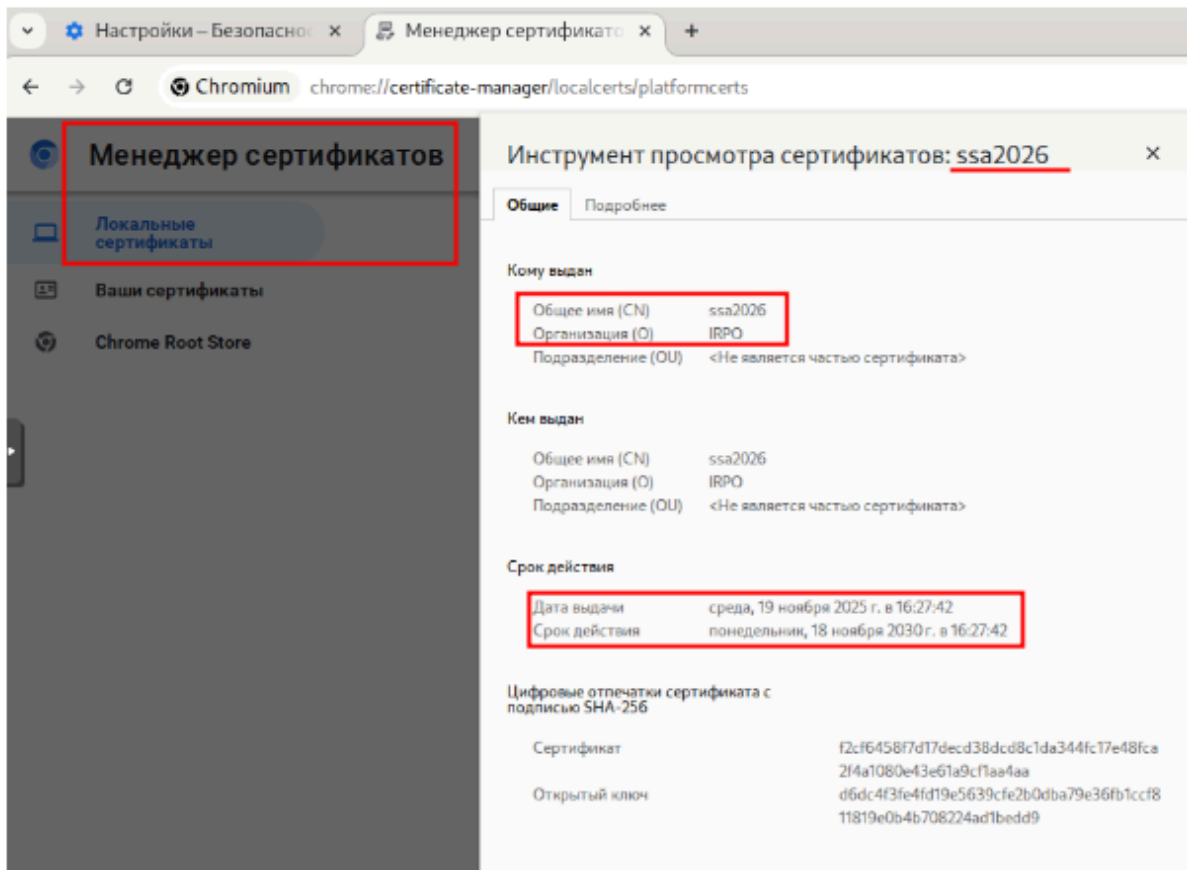
- Забираем через `scp` корневой сертификат и помещаем его в локальное хранилище сертификатов:

```
scp root@srv1-cod.cod.ssa2026.region:/var/ca/cacert.pem /etc/pki/ca-trust/source/anchors/ca.crt
```

- Обновляем:

```
update-ca-trust
```

- Проверить в браузере наличие корневого УЦ:



srv1-cod (alt-server):

- Перейдём в директорию `/var/ca`:

```
cd /var/ca
```

- Создаём ключ для сертификата (для системы мониторинга):

```
openssl genrsa -out necerts/monitoring.key
```

```
openssl req -new -key newcerts/monitoring.key -out newcerts/monitoring.csr
```

- Результат:

```
(root@srv1-cod:~) openssl req -new -key newcerts/monitoring.key -out newcerts/monitoring.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:IRPD
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:srv1-cod.ssa2026.region
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
optional company name []:
root@srv1-cod:~#
```

- Создадим файл с расширениями:

```
cat <<EOF > monitoring.ext
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
subjectAltName=@alt_names
[alt_names]
DNS.1=srv1-cod.ssa2026.region
DNS.2=monitoring.cod.ssa2026.region
IP.1=192.168.10.1
EOF
```

- Подписываем запрос выпуская сертификат (для системы мониторинга):

```
openssl x509 -req -CA cacert.pem -CAkey private/cakey.pem -in
newcerts/monitoring.csr -out newcerts/monitoring.crt -days 365
-CACreateserial -extfile monitoring.ext
```

srv2-cod (alt-server):

- Устанавливаем PostgreSQL сервер:

```
apt-get update && apt-get install -y postgresql17-server
```

- Создаём системные базы данных:

```
/etc/init.d/postgresql initdb
```

- Результат:

```
[root@srv2-cod ~]# /etc/init.d/postgresql initdb
Creating default database:
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "ru_RU.UTF-8".
Кодировка БД по умолчанию, выбранная в соответствии с настройками: "UTF8".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных включён.

Введите новый пароль суперпользователя:
Повторите его:

исправление прав для существующего каталога /var/lib/pgsql/data... ok
создание подкаталогов... ok
выбирается реализация динамической разделяемой памяти... posix
выбирается значение "max_connections" по умолчанию... 100
выбирается значение "shared_buffers" по умолчанию... 128MB
выбирается часовой пояс по умолчанию... Europe/Moscow
создание конфигурационных файлов... ok
▶ выполняется подготовительный скрипт... ok
▶ выполняется заключительная инициализация... ok
сохранение данных на диске... ok

Success. You can now start the database server using:

    service postgresql start
or
    pg_ctl -D /var/lib/pgsql/data -l файл_журнала start

[root@srv2-cod ~]# _
```

- Включаем и добавляем в автозагрузку PostgreSQL:

```
systemctl enable --now postgresql
```

- По умолчанию доступ из сети к PostgreSQL отключён:

```
[root@srv2-cod ~]# ss -tlnp | grep "5432"
LISTEN 0      200          127.0.0.1:5432      0.0.0.0:*    users:((("postgres",pid=1577,fd=7))
[root@srv2-cod ~]#
```

```

#-----
# CONNECTIONS AND AUTHENTICATION
#-----
# - Connection Settings -
listen_addresses = '*'

#port = 5432
max_connections = 100

```

- Настраиваем парольную аутентификацию для удалённого доступа отредактировав конфигурационный файл `/var/lib/pgsql/data/pg_hba.conf`:
 - добавляем следующую запись:

```

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all scram-sha-256
# IPv4 local connections:
host all all 0.0.0.0/0 md5
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all scram-sha-256
host replication all 127.0.0.1/32 scram-sha-256

```

- Перезапускаем PostgreSQL:

```
systemctl restart postgresql
```

- Проверяем доступ по сети:

```

[root@sru2-cod ~]# ss -tlnp | grep "5432"
LISTEN 0      200      0.0.0.0:5432      0.0.0.0:*      users:((("postgres",pid=1679,fid=7))
LISTEN 0      200      [::]:5432        [::]:*         users:((("postgres",pid=1679,fid=8))
[root@sru2-cod ~]#

```

- Создадим суперпользователя `superadmin`:

- задав пароль `P@ssw0rdSQL`

```
createuser -U postgres --superuser --encrypted --pwprompt superadmin
```

- Создадим базу данных `superadmin`:

```
su - postgres -s /bin/sh -c 'createdb -O superadmin superadmin'
```

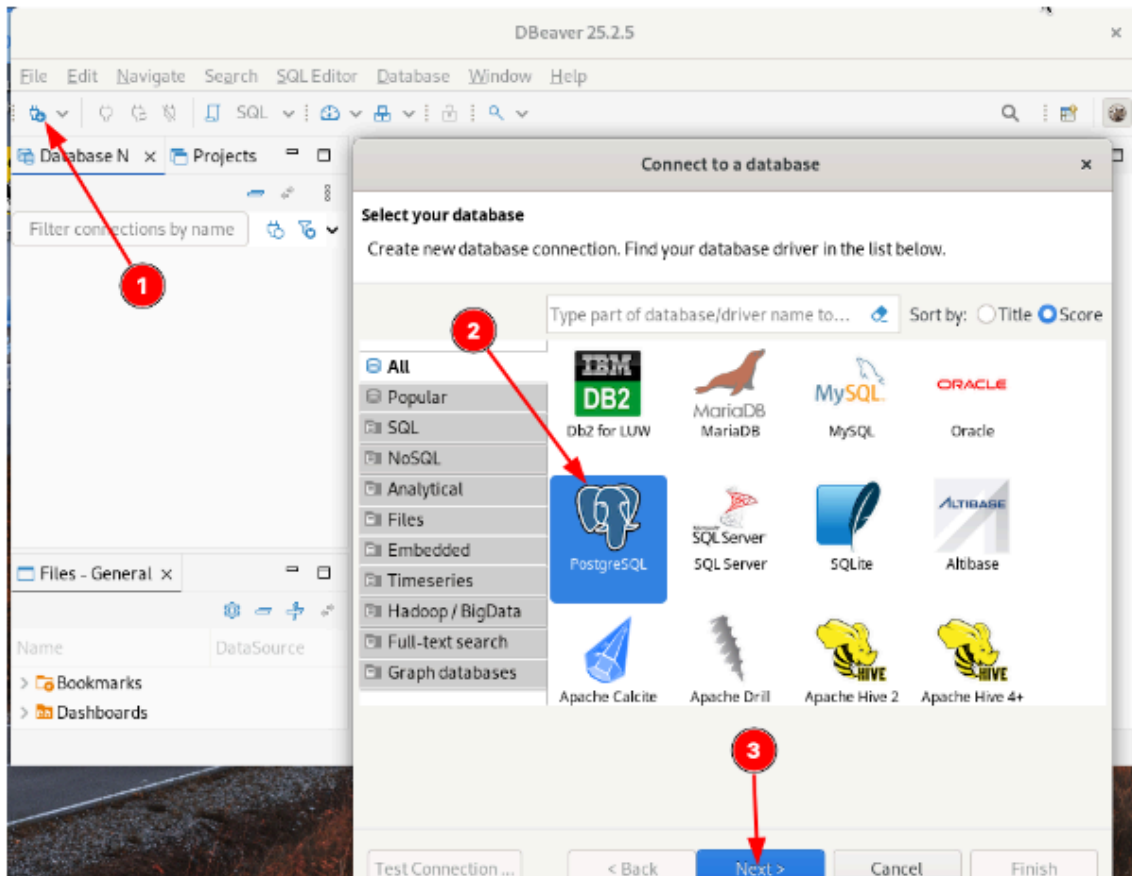
admin-cod (alt-server):

- Устанавливаем **DBeaver**:

```
apt-get update && apt-get install -y eepr
```

```
epm play -y dbeaver
```

- Открываем **DBeaver** и создаём подключение к БД:



Connect to a database

Connection Settings

PostgreSQL connection settings

 PostgreSQL

Main

Advanced

Driver properties

+ SSH, SSL, ...

No profile

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:postgresql://srv2-cod.cod.ssa2026.region:5432/superadmin

Host: srv2-cod.cod.ssa2026.region

Port: 5432

Database: superadmin ☐ Show all databases

Authentication

Authentication: Database Native

Username: superadmin

Password:

••••••••

☒ Save password

[Connection variables information](#)

[PostgreSQL](#)

Connection details (name, type, ...)

Driver name: PostgreSQL

Driver Settings

Driver license

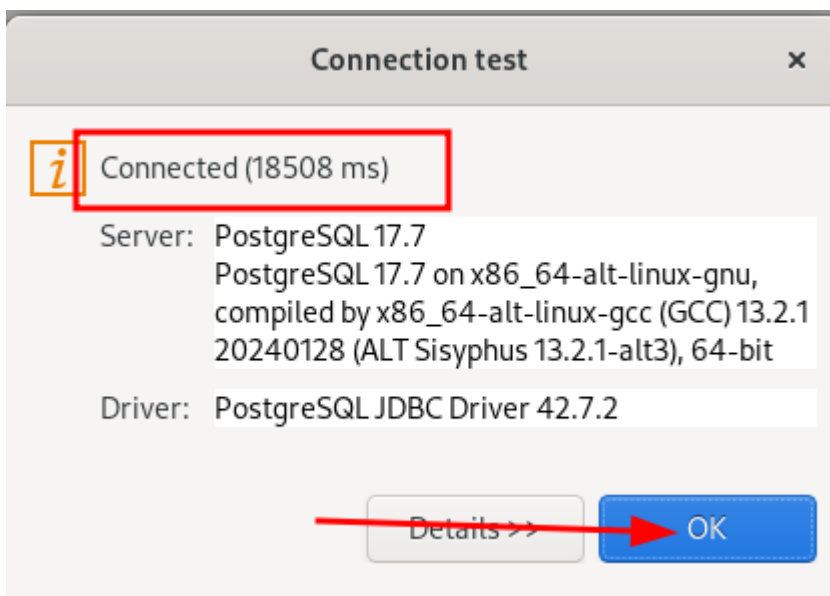
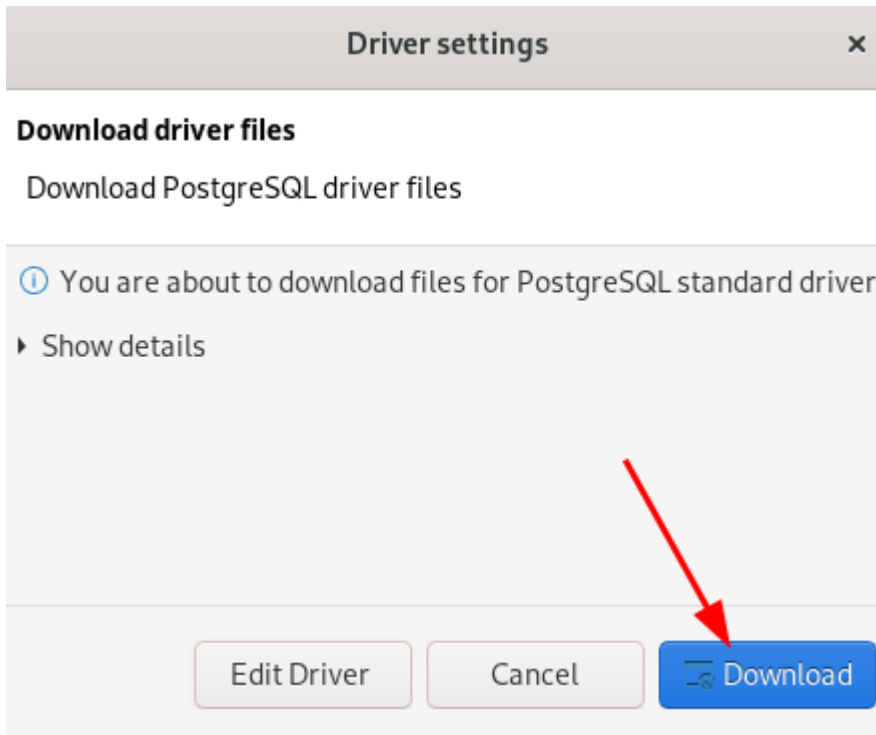
Test Connection ...

< Back

Next >

Cancel

Finish



Connect to a database

Connection Settings

PostgreSQL connection settings

 PostgreSQL

MainAdvancedDriver properties

+ SSH, SSL, ...No profile

Server

Connect by: ☒ Host ☐ URL

URL:

jdbc:postgresql://srv2-cod.cod.ssa2026.region:5432/superadmin

Host:

srv2-cod.cod.ssa2026.region

Port:

5432

Database:

superadmin

☐ Show all databases

Authentication

Authentication: Database Native

Username:

superadmin

Password:

●●●●●●●●

☒ Save password

Connection variables information

PostgreSQL

Connection details (name, type, ...)

Driver name: PostgreSQL

Driver Settings

Driver license

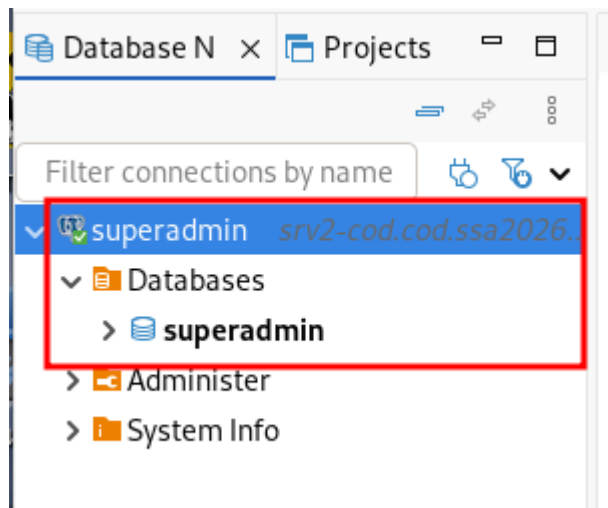
Test Connection ...

< Back

Next >

Cancel

Finish



srv2-cod (alt-server):

- Установить пакет `scsitarget-utils`:

```
apt-get update && apt-get install -y scsitarget-utils
```

- Включить и добавить в автозагрузку службу `tgt`:

```
systemctl enable --now tgt
```

- С помощью утилиты `lsblk` посмотреть список блочных устройств и определиться с диском, который будет использоваться в данном примере это `sda`

```
root@srv2-cod ~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda          8:0    0   5G  0 disk
sdb          8:16   0   2G  0 disk
└─sdb1       8:17   0    2G  0 part [SWAP]
└─sdb2       8:18   0   10G  0 part /
sr0         11:0    1 1024M  0 rom
```

- Настроить отдачу нашего диска по iSCSI отредактировав конфигурационный файл `/etc/tgt/targets.conf`:
 - добавив в конец файла следующее содержимое

```
# This is a sample config file for tgt-admin.
#
# The "#" symbol disables the processing of a line.
# Set the driver. If not specified, defaults to "iscsi".
default-driver iscsi

# Set iSNS parameters, if needed
#iSNSServerIP 192.168.111.222
#iSNSServerPort 3205
#iSNSAccessControl On
#iSNS On

# Continue if tgtadm exits with non-zero code (equivalent of
# --ignore-errors command line option)
#ignore-errors yes

<target iqn.2026-11.region.ssa2026.cod:data.target>
    direct-store /dev/sda
</target>
```

- Перезапустить службу `tgt`:

```
systemctl restart tgt
```

- Проверить можно с помощью команды `tgtadm --lld iscsi --op show --mode target`:

```

[root@sru2-cod ~]# tgtadm --lld iscsi --op show --mode target
Target 1: iqn.2026-11.region.ssa2026.cod:data.target
System information:
  Driver: iscsi
  State: ready
I_T nexus information:
LUN information:
  LUN: 0
    Type: controller
    SCSI ID: IET      00010000
    SCSI SN: beaf10
    Size: 0 MB, Block size: 1
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    SWP: No
    Thin-provisioning: No
    Backing store type: null
    Backing store path: None
    Backing store flags:
  LUN: 1
    Type: disk
    SCSI ID: IET      00010001
    SCSI SN: beaf11
    Size: 5369 MB, Block size: 512
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    SWP: No
    Thin-provisioning: No
    Backing store type: rdwr
    Backing store path: /dev/sda
    Backing store flags:
Account information:
ACL information:
  ALL
[root@sru2-cod ~]#

```

Для

корректной работы необходимо указать, чтобы LVM не сканировал наши iSCSI-диски (sd*) в конфигурационном файле /etc/lvm/lvm.conf:

- в блоке devices

```

#
# This configuration option has an automatic default value.
# filter = [ "a|*" ]
filter = [ "r|/dev/sd*" ]
# Configuration option devices/global-filter

```

srv1-cod (alt-server):

- Установим пакет **open-iscsi**:

```
apt-get update && apt-get install -y open-iscsi
```

- Включаем и добавляем в автозагрузку службу **iscsid**:

```
systemctl enable --now iscsid
```

- Посмотреть доступные для подключения target-ы можно с помощью команды:

```
iscsiadm -m discovery -t sendtargets -p 192.168.20.2
```

- Подключить target-ы:

```
iscsiadm -m node --login
```

- В файле **/etc/iscsi/iscsid.conf** внести изменения:
 - закомментировать **node.startup = manual**
 - раскомментировать **node.startup = automatic**

```
#####
# Startup settings
#####

# To request that the iscsi service scripts startup a session, use "automatic":
node.startup = automatic
#
# To manually startup the session, use "manual". The default is manual.
#node.startup = manual

# For "automatic" startup nodes, setting this to "Yes" will try logins on each
# available iface until one succeeds, and then stop. The default "No" will try
# logins on all available ifaces simultaneously.
```

- В файле **/var/lib/iscsi/send_targets/<TargetServer>,<Port>/st_config** внести изменения:
 - параметр **discovery.sendtargets.use_discoveryd = No** поменять на **discovery.sendtargets.use_discoveryd = Yes**

```
■ BEGIN RECORD 2.1.10
discovery.startup = manual
discovery.type = sendtargets
discovery.sendtargets.address = 192.168.10.2
discovery.sendtargets.port = 3260
discovery.sendtargets.auth.authmethod = None
node.session.auth.chap_algs = MD5
discovery.sendtargets.timeo.login_timeout = 15
discovery.sendtargets.use_discoveryd = Yes
discovery.sendtargets.discoveryd_poll_interval = 30
discovery.sendtargets.reopen_max = 5
discovery.sendtargets.timeo.auth_timeout = 45
discovery.sendtargets.timeo.active_timeout = 30
discovery.sendtargets.iscsi.MaxRecvDataSegmentLength = 32768
■ END RECORD
```

- Выполнить перезагрузку устройства:

```

Welcome to ALT Server 11.0 (Mendeleevium)!

Hostname: srv1-cod.cod.ssa2026.region
IP: 192.168.10.1
srv1-cod login: root
Password:
Last login: Thu Nov 20 16:12:22 MSK 2025 on tty1
[root@srv1-cod ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
├─sda         8:0    0   20G  0 disk
│ └─sda1       8:1    0    2G  0 part [SWAP]
│ └─sda2       8:2    0   18G  0 part /
└─sdb         8:16   0    5G  0 disk
sr0          11:0    1 1024M  0 rom
[root@srv1-cod ~]#

```

srv1-cod (alt-server):

- Реализуем LVM-том:
 - при помощи утилиты **lsblk** необходимо определить диск который будет использоваться;
 - в данном случае будет использоваться диск **sdb** (ранее подключённый по iSCSI)

```

[root@srv1-cod ~]# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
├─sda         8:0    0   20G  0 disk
│ └─sda1       8:1    0    2G  0 part [SWAP]
│ └─sda2       8:2    0   18G  0 part /
└─sdb         8:16   0    5G  0 disk
sr0          11:0    1 1024M  0 rom
[root@srv1-cod ~]#

```

- Помечаем диски, что данный диск будет использоваться для LVM:

```
pvccreate /dev/sdb
```

- результат:

```

[root@srv1-cod ~]# pvccreate /dev/sdb
Physical volume "/dev/sdb" successfully created.
[root@srv1-cod ~]# _

```

- Создаём группу логических томов с именем VG:

```
vgcreate VG /dev/sdb
```

- результат:

```

[root@srv1-cod ~]# vgcreate VG /dev/sdb
Volume group "VG" successfully created
[root@srv1-cod ~]#

```

- Создадим логический том с именем **DATA**, используя **100%** дискового пространства:

```
lvcreate -l 100%FREE -n DATA VG
```

- результат:

```

[root@srv1-cod ~]# lvcreate -l 100%FREE -n DATA VG
Logical volume "DATA" created.
[root@srv1-cod ~]# _

```

- Проверить можно с помощью команды `lvdisplay`:

```
[root@srv1-cod ~]# lvdisplay
--- Logical volume ---
LV Path                /dev/VG/DATA
LV Name                 DATA
VG Name                 VG
LV UUID                FzuK7J-asKb-hIhP-gMif-nYNw-bzFB-nb9DQ5
LV Write Access         read/write
LV Creation host, time srv1-cod.cod.ssa2026.region, 2025-11-21 08:30:15 +0300
LV Status                available
# open                  0
LV Size                 <5.00 GiB
Current LE              1279
Segments                1
Allocation              inherit
Read ahead sectors      auto
- currently set to     256
Block device            253:0

[root@srv1-cod ~]# _
```

- в качестве файловой системы используем `xfs`:

```
mkfs.xfs /dev/VG/DATA
```

- результат:

```
[root@srv1-cod ~]# mkfs.xfs /dev/VG/DATA
meta-data=/dev/VG/DATA             isize=512    agcount=4, agsize=327424 blks
=                                   sectsz=512   attr=2, projid32bit=1
=                                   crc=1      finobt=1, sparse=1, rmapbt=1
=                                   reflink=1  bigtime=1 inobtcount=1 nrext64=1
=                                   exchange=0
data     =                          bsize=4096   blocks=1309696, inapct=25
=                                   sunit=0     swidth=0 blks
naming   =version 2                 bsize=4096   ascii-ci=0, ftype=1, parent=0
log      =internal log             bsize=4096   blocks=16384, version=2
=                                   sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                     extsz=4096   blocks=0, rtextents=0

[root@srv1-cod ~]# _
```

- Настраиваем автоматическое монтирование логического тома в `/opt/data`:

- создадим директорию для монтирования:

```
mkdir /opt/data
```

- Добавляем в файл `/etc/fstab` следующую строку для автоматического монтирования логического тома `DATA`:

- *P.S. или на основе `UUID`*

```
proc                /proc                proc                nosuid,noexec,gid=proc                0 0
devpts              /dev/pts            devpts              nosuid,noexec,gid=tty,node=620,ptmxnode=0666 0 0
tmpfs               /tmp                tmpfs               nosuid                0 0
UUID=5f52230a-0885-4f1d-ad03-a424e1077d2c /                   ext4               relative            1 1
UUID=6bb42088-62a6-4df6-8814-e89862322443 swap                swap               defaults            0 0
/dev/VG/DATA        /opt/data           xfs                 defaults,_,netdev   0 0
```

- Выполняем монтирование:

```
mount -av
```

srv1-cod (alt-server):

- Устанавливаем пакеты для NFS сервера:

```
apt-get install -y nfs-server nfs-utils
```

- Назначаем права на созданную ранее директорию для общего доступа (полный доступ):

```
chmod 777 /opt/data
```

- Редактируем файл `/etc/exports`:

- Добавляем туда следующую информацию

```
#/srv/share -rw,insecure,fsid=0,sec=krb5 *  
/opt/data 192.168.30.0/24(rw,no_root_squash)
```

- Экспортируем файловую систему, указанную выше в `/etc/exports`:

```
exportfs -arv
```

- Результат:

```
(root@srv1-cod ~) exportfs -arv  
exporting 192.168.30.0/24:/opt/data  
(root@srv1-cod ~)
```

- Запускаем и добавляем в автозагрузку NFS - сервер:

```
systemctl enable --now nfs-server
```

admin-cod (alt-workstation):

- Выполняем установку пакетов для NFS - клиента:

```
apt-get update && apt-get install -y nfs-utils nfs-clients
```

- Создадим директорию для монтирования общего ресурса:

```
mkdir /mnt/nfs
```

- Задаём права на созданную директорию:

```
chmod 777 /mnt/nfs
```

- Настраиваем автоматическое монтирование общего ресурса через `fstab`:

- Добавляем следующую информацию:

```
proc          /proc          proc          nosuid,noexec,gid=proc          0 0  
devpts        /dev/pts       devpts        nosuid,noexec,gid=tty,mode=620,ptmxmode=0666 0 0  
tmpfs         /tmp           tmpfs         nosuid          0 0  
UUID=eb35d215-ef34-4519-b707-8daa5f637662 /             btrfs        relatime,subvol=@              0 2  
UUID=eb35d215-ef34-4519-b707-8daa5f637662 /home          btrfs        nosuid,relatime,subvol=@home   0 0  
UUID=477c4adc-ab05-4d7c-a66d-afae7c94e593 swap           swap          defaults          0 0  
/dev/sr0      /media/ALTLinux udf,iso9660    ro,noauto,user=utf8,nofail,comment=x-gvfs-show 0 0  
192.168.10.1:/opt/data /mnt/nfs       nfs           defaults,_netdev 0 0
```

```
mount -av
```

• Результат:

```
[root@admin-cod ~]# mount -av
/proc                : already mounted
/dev/pts              : already mounted
/tmp                  : already mounted
/                     : ignored
/home                 : already mounted
swap                  : ignored
/media/ALTLinux        : ignored
mount.nfs: timeout set for Fri Nov 21 09:03:07 2025
mount.nfs: trying text-based options 'vers=4.2,addr=192.168.10.1,clientaddr=192.168.30.30'
/mnt/nfs               : successfully mounted
[root@admin-cod ~]#
```

• Проверяем:

```
df -h
```

srv2-cod (alt-server):

• Устанавливаем пакет **zabbix-server-pgsql**:

```
apt-get update && apt-get install -y zabbix-server-pgsql
```

• Создаём пользователя **zabbix_user** в базе данных PostgreSQL:

```
su - postgres -s /bin/sh -c 'createuser --no-superuser --no-createdb --no-createrole --encrypted --pwprompt zabbix_user'
```

• После запуска данной команды - задаём в качестве пароля для пользователя **zabbix_user** - пароль **P@ssw0rdZabbix** и подтверждаем его:

```
[root@srv2-cod ~]# su - postgres -s /bin/sh -c 'createuser --no-superuser --no-createdb --no-createrole --encrypted --pwprompt zabbix_user'
Enter password for new role:
Enter it again:
Password:
[root@srv2-cod ~]#
```

• Создаём базу данных с именем **zabbix**:

```
su - postgres -s /bin/sh -c 'createdb -O zabbix_user zabbix'
```

• Выполняем перезагрузку службы **postgresql**:

```
systemctl restart postgresql
```

• Добавляем в базу данные для веб-интерфейса:

```
su - postgres -s /bin/sh -c 'psql -U zabbix_user -f /usr/share/doc/zabbix-common-database-pgsql-*/schema.sql zabbix'
```

```
su - postgres -s /bin/sh -c 'psql -U zabbix_user -f /usr/share/doc/zabbix-common-database-pgsql-*/images.sql zabbix'
```

```
su - postgres -s /bin/sh -c 'psql -U zabbix_user -f /usr/share/doc/zabbix-common-database-pgsql-*/data.sql zabbix'
```

srv1-cod (alt-server):

- Устанавливаем пакеты для веб-сервера **apache2**:

```
apt-get install -y apache2 apache2-mod_{php8.2,ssl}
```

- Включаем и добавляем в автозагрузку службу отвечающую за веб-сервер **apache2**:

```
systemctl enable --now httpd2
```

- Установим **PHP** и необходимые модули для корректной работы:

```
apt-get install -y php8.2 php8.2-{mbstring,sockets,gd,xmlreader,pgsql,ldap,openssl}
```

- Меняем некоторые опции **php** в файле `/etc/php/8.2/apache2-mod_php/php.ini`:
 - Находим следующие параметры и приводим их к следующему виду:

```
;;;;;;;;;;  
; Maximum execution time of each script in seconds  
; http://php.net/max-execution-time  
; Note: This directive is hardcoded to 0 for the CLI SAPI  
max_execution_time = 600  
  
; Maximum amount of time each script may spend parsing request  
; idea to limit this time on production servers in order to avoid  
; long running scripts.  
; Note: This directive is hardcoded to -1 for the CLI SAPI  
; Default Value: -1 (Unlimited)  
; Development Value: 60 (60 seconds)  
; Production Value: 60 (60 seconds)  
; http://php.net/max-input-time  
max_input_time = 600  
  
; Maximum input variable nesting level  
; http://php.net/max-input-nesting-level  
; max_input_nesting_level = 64  
  
; How many GET/POST/COOKIE input variables may be accepted  
; max_input_vars = 1000  
  
; Maximum amount of memory a script may consume (128MB)  
; http://php.net/memory-limit  
memory_limit = 256M
```

```
; Maximum size of POST data that PHP will accept.  
; Its value may be 0 to disable the limit. It is ignored if  
; is disabled through enable_post_data_reading.  
; http://php.net/post-max-size  
post_max_size = 32M
```

```
[Date]  
; Defines the default timezone used by the date functions  
; http://php.net/date.timezone  
date.timezone = Europe/Moscow
```

```
; Always populate the $HTTP_RAW_POST_DATA variable. PHP's de  
; to disable this feature and it will be removed in a future  
; If post reading is disabled through enable_post_data_readi  
; $HTTP_RAW_POST_DATA is *NOT* populated.  
; http://php.net/always-populate-raw-post-data  
always_populate_raw_post_data = -1
```

- Перезапускаем службу отвечающую за веб-сервер `apache2`:

```
systemctl restart httpd2
```

- Устанавливаем `zabbix-server-pgsql`:

```
apt-get install -y zabbix-server-pgsql
```

- Вносим изменения в конфигурационный файл `/etc/zabbix/zabbix_server.conf`:
 - Добавляем следующие изменения:

```
# Mandatory: no
# Default:
DBHost=192.168.10.2

### Option: DBName
# Database name.
# If the Net Service Name connection method is used
# the tnsnames.ora file or set to empty string; also
# empty string.
#
# Mandatory: yes
# Default:
# DBName=
DBName=zabbix

# Option: DBSchema
# Schema name. Used for PostgreSQL.
#
# Mandatory: no
# Default:
# DBSchema=

### Option: DBUser
# Database user.
#
# Mandatory: no
# Default:
# DBUser=
DBUser=zabbix_user

### Option: DBPassword
# Database password.
# Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=P@ssw0rdZabbix

### Option: DBSocket
# Path to MySQL socket.
```

- Добавим **Zabbix-сервер** в автозапуск и запустить его:

```
systemctl enable --now zabbix_pgsql
```

- Установка веб-интерфейса Zabbix:

```
apt-get install zabbix-phpfrontend-{apache2,php8.2} -y
```

- Создаём файл конфигурации Virtualhost для Zabbix `/etc/httpd/conf/sites-available/zabbix.conf`:
 - Содержимое:

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    DocumentRoot /var/www/webapps/zabbix/ui
    ServerName srv1-cod.cod.ssa2026.region
    ServerAlias monitoring.cod.ssa2026.region 192.168.10.1
    <Directory /var/www/webapps/zabbix/ui/>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
    SSLCertificateFile /var/lib/ssl/certs/monitoring.crt
    SSLCertificateKeyFile /var/lib/ssl/private/monitoring.key
</VirtualHost>

<VirtualHost *:80>
    ServerName srv1-cod.cod.ssa2026.region
    ServerAlias monitoring.cod.ssa2026.region 192.168.10.1
    Redirect permanent / https://srv1-cod.cod.ssa2026.region/
</VirtualHost>
```

- Копируем ранее созданные сертификаты:

```
cp /var/ca/newcerts/monitoring.crt /var/lib/ssl/certs/
```

```
cp /var/ca/newcerts/monitoring.key /var/lib/ssl/private/
```

- Добавляем символическую осылку на файл `/etc/httpd/conf/sites-available/zabbix.conf`:

```
ln -s /etc/httpd/conf/sites-available/zabbix.conf /etc/httpd/conf/sites-enabled/
```

- Включаем модуль **ssl** и **rewrite**, а также порт **https**:

```
a2enmod ssl; a2enmod rewrite; a2enport https
```

- Проверяем синтаксис конфигурационного файла `/etc/httpd/conf/sites-available/zabbix.conf`:

```
apachectl -t
```

- Результат:

```
[root@srv1-cod ~]# apachectl -t
Syntax OK
[root@srv1-cod ~]#
```

- Назначаем владельца и группу владельцев для файлов проекта **zabbix**:

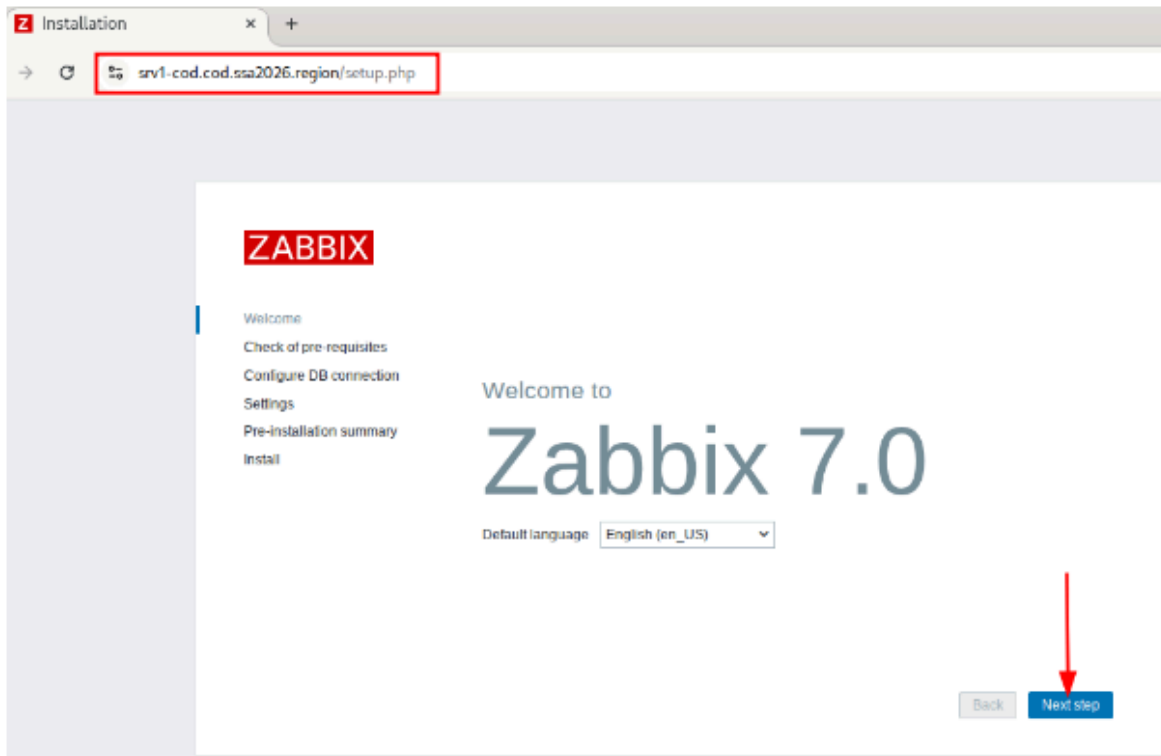
```
chown -R apache2:apache2 /var/www/webapps/zabbix
```

- Перезагружаем службу **httpd2**:

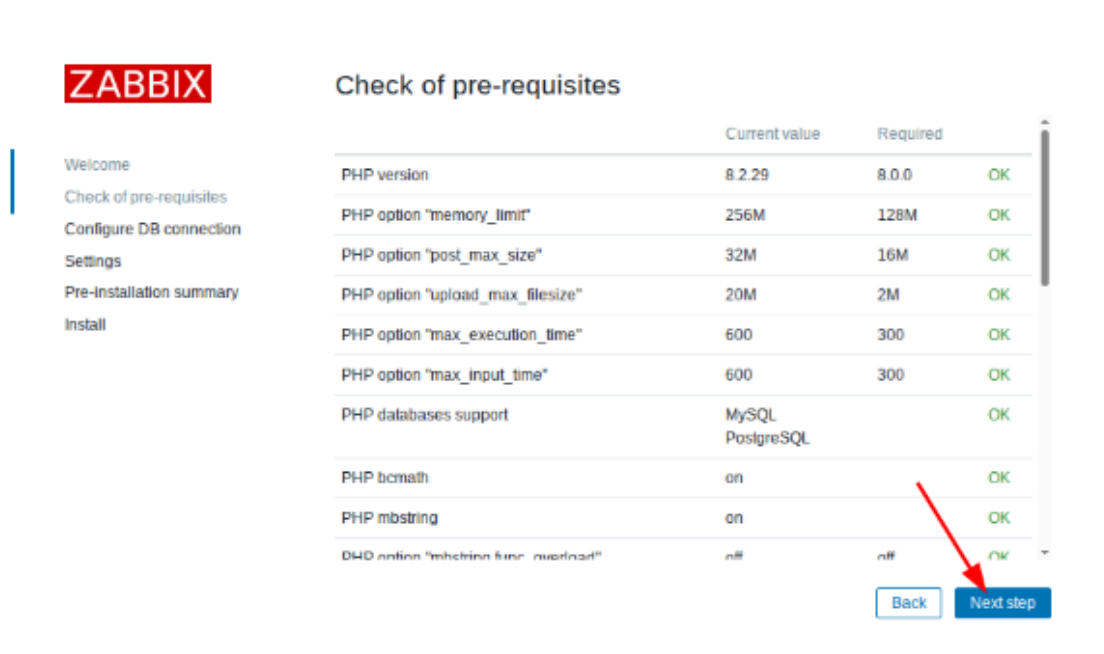
```
systemctl restart httpd2
```

admin-cod (alt-workstation):

- Продолжаем установку **zabbix** - через веб-интерфейс:
 - переходим на <https://srv1-cod.cod.ssa2026.region/setup.php> и нажимаем **Next step**:



- нажимаем **Next step**:



- В качестве базы данных выбираем - **PostgreSQL**;
- В качестве сервера базы данных указываем IP - адрес или имя **srv2-cod.cod.ssa2026.region**;
- Указываем имя созданной базы данных **zabbix**;
- Указываем имя созданного пользователя **zabbix_user**;
- Указываем пароль для пользователя zabbix **P@ssw0rdZabbix**;
- Нажимаем **Next step**

ZABBIX

Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Welcome

Check of pre-requisites

Configure DB connection

Settings

Pre-installation summary

Install

Database type PostgreSQL

Database host srv2-cod.cod.ssa2026.re

Database port 0 0 - use default port

Database name zabbix

Database schema

Store credentials in Plain text HashiCorp Vault CyberArk Vault

User zabbix_user

Password *****

Database TLS encryption ☒

Back

Next step

- при необходимости задаём имя серверу и нажимаем **Next step**

ZABBIX

Settings

Zabbix server name monitoring.cod.ssa2026.r

Default time zone System: (UTC+03:00) Europe/Moscow

Default theme Blue

Welcome

Check of pre-requisites

Configure DB connection

Settings

Pre-installation summary

Install

Back

Next step

- проверяем введенные ранее параметры и нажимаем **Next step**



Welcome
Check of pre-requisites
Configure DB connection
Settings
Pre-installation summary
Install

Pre-installation summary

Please check configuration parameters. If all is correct, press "Next step" button, or "Back" button to change configuration parameters.

Database type	PostgreSQL
Database server	srv2-cod.cod.ssa2026.region
Database port	default
Database name	zabbix
Database user	zabbix_user
Database password	*****
Database schema	
Database TLS encryption	true
Zabbix server name	monitoring.cod.ssa2026.region

[Back](#)[Next step](#)

- нажимаем **Finish**



Welcome
Check of pre-requisites
Configure DB connection
Settings
Pre-installation summary
Install

Install

Congratulations! You have successfully installed Zabbix frontend.

Configuration file "conf/zabbix.conf.php" created.

[Back](#)[Finish](#)

- выполняем вход из под пользователя по умолчанию: **Admin** с паролем: **zabbix**

srv1-cod.cod.ssa2026.region/index.php

ZABBIX

Username

Admin

Password

☒ Remember me for 30 days

Sign in

- Результат:

monitoring.cod.ssa2026

srv1-cod.cod.ssa2026.region/zabbix.php?action=dashboard.view&dashboardid=1

ZABBIX

Global view

All dashboards / Global view

Top hosts by CPU utilization

Host name	Utilization	1m avg	5m avg	15m avg	Processes
Zabbix server					

1.10 ↓
Zabbix server
Values per sec...

System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Zabbix server version	7.0.20	
Zabbix frontend version	7.0.20	
Number of hosts (enabled/disabled)	1	1 / 0
Number of templates	350	
Number of items (enabled/disabled/not supported)	121	110 / 0 / 11

Host availability

Available	Not ava...	Mixed	Unknown	Tot
0	1	0	0	1

Problems by severity

Disaster	High	Average	Warning	Information	Not class...
0	0	1	0	0	0

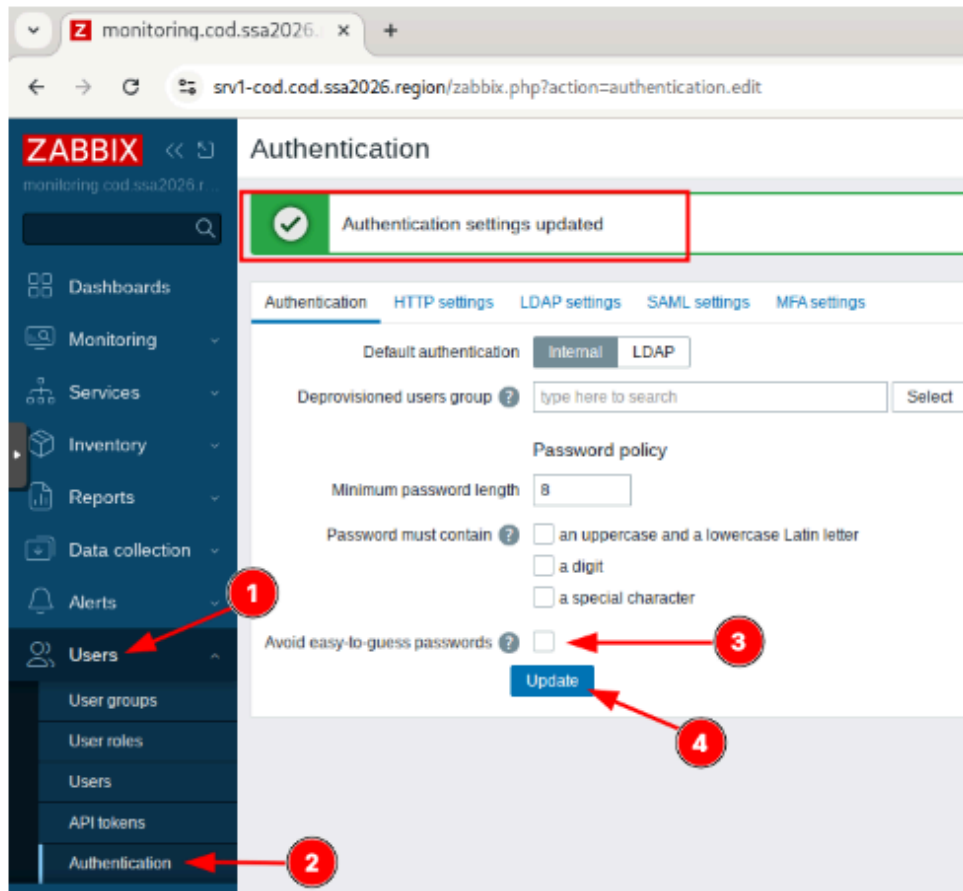
Geomap

Current problems

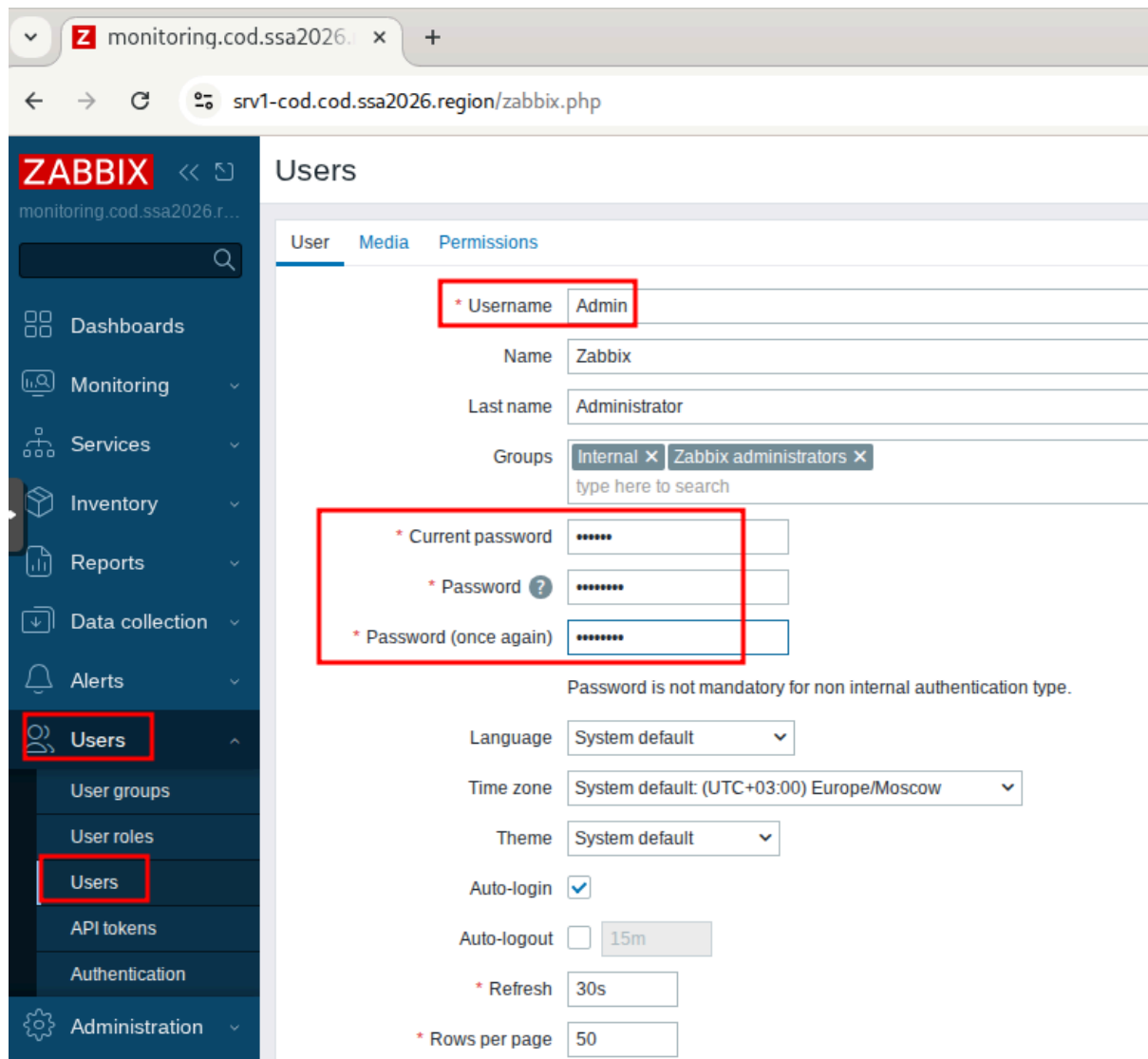
Time	Info	Host	Problem + Severity	Duration	Update	Actions	Tags
10:37:26 AM		Zabbix server	Linux: Zabbix agent is not available (for 3m)	59m 47s	Update	class:os component:system scope:availability	

- В качестве пароля для пользователя **Admin** - необходимо установить **P@ssw0rd**:
 - переходим в настройки аутентификации и снимаем галочку, которая запрещает использование слабых паролей

- В качестве пароля для пользователя **Admin** - необходимо установить **P@ssw0rd**:
 - переходим в настройки аутентификации и снимаем галочку, которая запрещает использование слабых паролей



- Задаём новый пароль **P@ssw0rd** - для пользователя **Admin**



srv1-cod, srv2-cod, dc-a, sw1-cod, sw2-cod, sw1-a, sw2-a (alt-server):

- Устанавливаем необходимый пакет **zabbix-agent**:

```
apt-get install zabbix-agent -y
```

- Редактируем конфигурационный файл **/etc/zabbix/zabbix_agentd.conf**:
 - вносим следующие изменения:

```
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=
Server=192.168.10.1
### Option: ListenPort
# Agent will listen on this port for connections from the server
```

- в поле **Hostname** необходимо указывать полное доменное имя соответствующего хоста

```
ServerActive=192.168.10.1
# Option: Hostname
# List of comma delimited unique, case sensitive hostnames.
# Required for active checks and must match hostnames as configured on the server.
# Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=
Hostname=srv1-cod.cod.ssa2026.region
```

- Включаем и добавляем в автозагрузку службу **zabbix_agentd**:

```
systemctl enable --now zabbix_agentd.service
```

admin-cod (alt-workstation):

- Переходим в веб-интерфейс управления fw-cod:

The screenshot shows the web interface of Idec NGFW Novum. The browser address bar displays the URL `https://fw-cod.cod.ssa2026.region:8443/#/monitor/zabbix`. The left sidebar menu has 'Мониторинг' (Monitoring) selected. The main configuration area for 'Zabbix-агент' (Zabbix agent) is shown. The status is 'Остановлен' (Stopped). The configuration includes checkboxes for 'Zabbix-агент', 'Отправка данных к Zabbix (активный режим)', and 'Приём запросов от Zabbix (пассивный режим)'. There are input fields for 'Название сервера Idec NGFW Novum', 'Адрес сервера' (containing '192.168.10.1'), and 'Порт для подключения' (with radio buttons for 10050 and 10051). A 'Сохранить' (Save) button is at the bottom.

- Результат:

The screenshot shows the Zabbix agent status page. The status is 'Работает' (Working). A green message box at the bottom says 'Модуль "zabbix-agent" работает' (The 'zabbix-agent' module is working).

rtr-cod и rtr-a (ecorouter):

- Запуск сервиса SNMP:

```
rtr-cod(config)#snmp-server enable snmp
rtr-cod(config)#
```

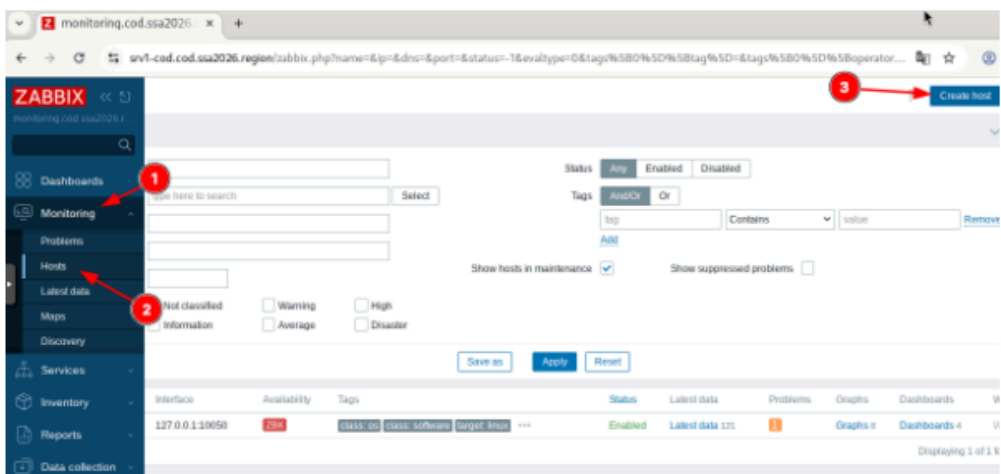
- Настроить SNMP community:

```
rtr-cod(config)#snmp-server community public ro
rtr-cod(config)#write memory
Building configuration...

rtr-cod(config)#
```

admin-cod (alt-workstation):

- Переходим на <https://srv1-cod.cod.ssa2026.region>
- Каждый хост необходимо зарегистрировать на сервере Zabbix, сделать это можно, используя веб-интерфейс
 - переходим Monitoring -> Hosts -> Create host



- Добавляем хосты в соответствии с требованиями задания:

Host

Host IPMI Tags Macros Inventory ● Encryption Value mapping

* Host name

Visible name

Templates
type here to search

* Host groups
type here to search

Interfaces	Type	IP address	DNS name	Connect to	Port
Agent		192.168.10.1	srv1-cod.cod.ssa2026.region	<input checked="" type="button" value="IP"/> DNS	10050

[Add](#)

Description

- Аналогично все остальные хосты:

Name ▲	Interface	Availability	Tags	Status	Latest data
dc-a.office.ssa2026.region	172.20.10.10:10050	ZBX	class: os target: linux	Enabled	Latest data 77
rtr-a.office.ssa2026.region	10.10.10.2:161	SNMP	class: network target: cisco target: cisco-ios	Enabled	Latest data 23
rtr-cod.cod.ssa2026.region	172.16.1.1:161	SNMP	class: network target: cisco target: cisco-ios	Enabled	Latest data 23
srv1-cod.cod.ssa2026.region	192.168.10.1:10050	ZBX	class: os target: linux	Enabled	Latest data 180
srv2-cod.cod.ssa2026.region	192.168.10.2:10050	ZBX	class: os target: linux	Enabled	Latest data 95
sw1-a.office.ssa2026.region	172.20.30.1:10050	ZBX	class: os target: linux	Enabled	Latest data 131
sw1-cod.cod.ssa2026.region	192.168.30.1:10050	ZBX	class: os target: linux	Enabled	Latest data 149
sw2-a.office.ssa2026.region	172.20.30.2:10050	ZBX	class: os target: linux	Enabled	Latest data 131
sw2-cod.cod.ssa2026.region	192.168.30.2:10050	ZBX	class: os target: linux	Enabled	Latest data 149

- Логинимся из-под суперпользователя **root** с паролем указанным на этапе установки ОС
- Редактируем конфигурационный файл **/etc/sysconfig/network-scripts/ifcfg-eth0**:

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
BOOTPROTO="static"
IPADDR="192.168.50.50"
NETMASK="255.255.255.0"
GATEWAY="192.168.50.254"
DNS1="192.168.10.1"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6_INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="eth0"
UUID="6080706e-7d6a-4baf-ae4a-3d6860361673"
DEVICE="eth0"
ONBOOT="yes"
```

- Для реализации внутренних звонков необходимо использовать **CHAN_SIP**:
 - переходим в **Settings -> Advanced Settings**:
 - ищем параметр **SIP Channel Driver** и выбираем **chan_sip** и нажимаем **Submit**

SIP Channel Driver ? ↻

chan_sip

- переходим в **Settings -> Asterisk SIP Settings**:
- ищем параметр **Bind Port** и меняем с 5160 на 5060 и нажимаем **Submit**

Bind Port ?

5060

- Нажимаем **Apply Config**, после чего перезагружаем **sip-cod** целиком:

Dashboard
Reports
Settings
UCP

Apply Config

Yes No

Yes No

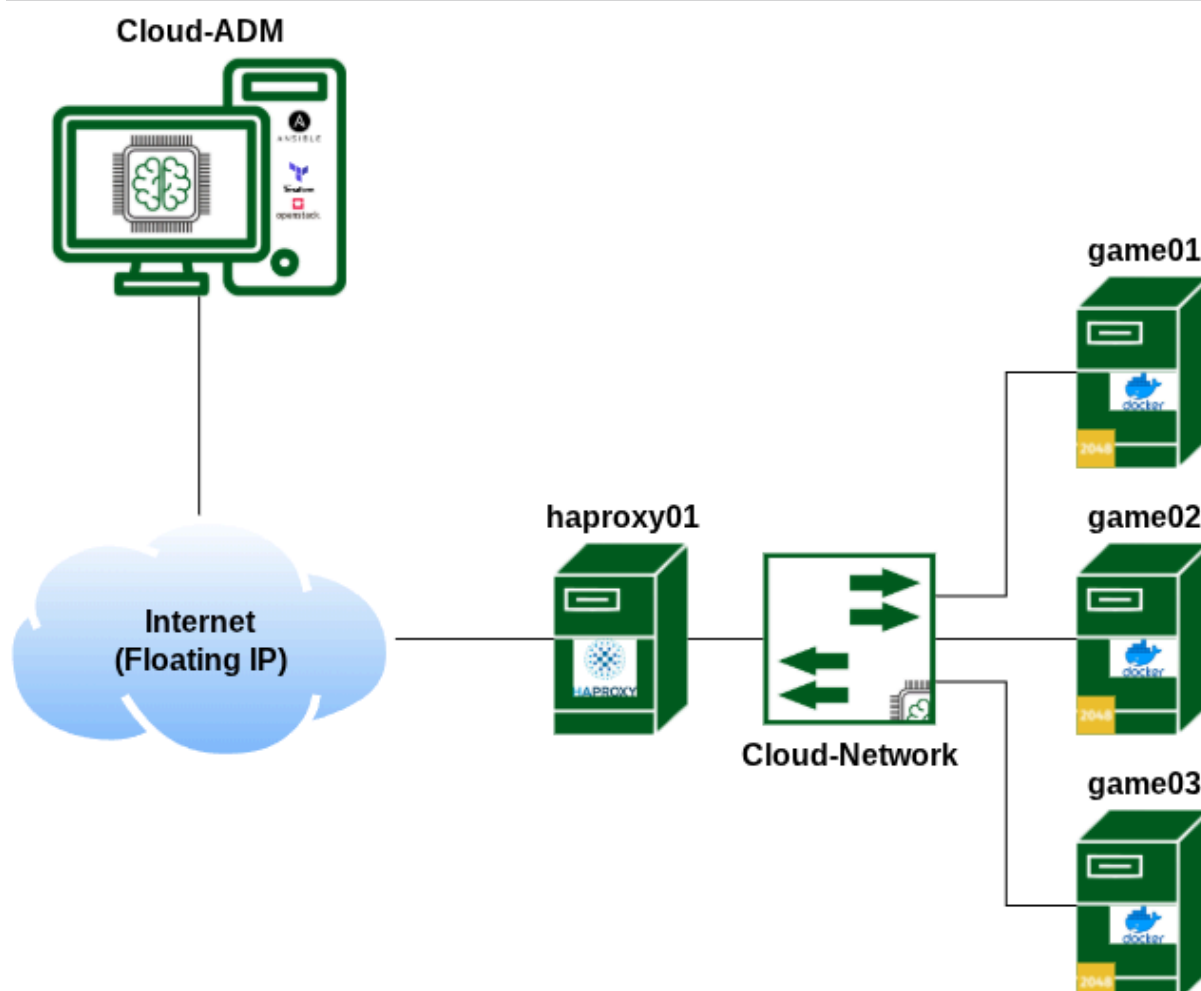
Yes No

5060

Reloading...

МОДУЛЬ Г

- https://disk.yandex.ru/d/uhpN6U6UYRK_Nw (у приложения есть README от программиста-разработчика)



Задание:

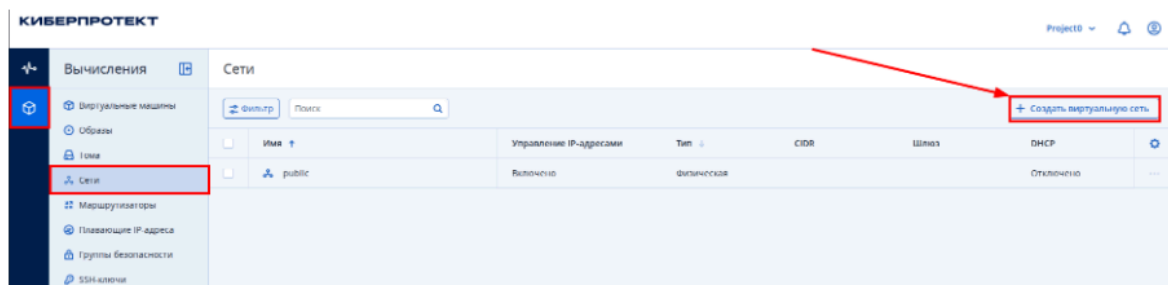
- Создайте виртуальный инстанс с именем Cloud-ADM, подключите его к необходимым сетям, ассоциируйте с ним плавающий IP-адрес.
- Установите следующие параметры для создаваемого инстанса:
 - Тип виртуальной машины: 2 vCPU, 4 ГБ RAM;
 - Размер диска: 30 ГБ.
- В качестве операционной системы используйте Альт Рабочая станция, шаблон alt-workstation-11.0-p11-cloud-x86_64.
- Настройте внешнее подключение по SSH сохранив ключевую пару для доступа с вашего локального ПК на рабочем столе с расширением .pem:
 - Создайте в PuTTY профиль с именем Cloud-ADM;
 - Реализуйте возможность установления соединения с инстансом Cloud-ADM с локального ПК через PuTTY, без необходимости ввода дополнительных параметров.
- Настройте внешнее подключение по RDP сохранив на рабочем столе профиль с именем Cloud-ADM:
 - Для подключения используйте имя пользователя altlinux и пароль P@ssw0rd.

Вариант реализации:

- Создаём ключевую пару для доступа к Cloud-ADM на рабочем столе с расширением '.pem':

```
puttygen -t rsa -b 2048 -o Рабочий\стол/Cloud-ADM.pem
```

- Создаём виртуальную сеть с произвольным именем, например cloud-net:
 - Перейдите в **ВЫЧИСЛЕНИЯ**;
 - Выберите раздел **Сети**;
 - Нажмите **Создать виртуальную сеть**:



- На этапе **Конфигурация сети** необходимо задать **Имя** и нажать **Далее**:
 - Управление IP-адресами** - при включенном управлении IP-адресами система сможет управлять назначением IP-адресов.

Создать виртуальную сеть

The screenshot shows the 'Создать виртуальную сеть' (Create virtual network) dialog box. On the left, there are three tabs: 'Конфигурация сети' (Network configuration), 'Управление IP-адресами' (IP address management), and 'Сводка' (Summary). The 'Конфигурация сети' tab is selected. In the main area, there is a toggle switch for 'Управление IP-адресами' (IP address management) which is turned on. Below it, there is a text input field for 'Имя' (Name) containing the text 'cloud-net'. A red box highlights this field. At the bottom right, there are two buttons: 'Отмена' (Cancel) and 'Далее' (Next). A red arrow points to the 'Далее' button.

- На этапе **Управление IP-адресами** необходимо добавить подсеть нажав **Добавить** -> **Подсеть IPv4**:

Создать виртуальную сеть

• Конфигурация сети

• Управление IP-адресами

• Сводка

Настройте параметры сети для управления адресами IPv4 и IPv6.

Подсети

Добавить

Назад

Далее

- При добавлении подсети IPv4 необходимо:
 1. Указать **CIDR** (например, 192.168.1.0/24);
 2. Указать **Шлюз**, чтобы у виртуальных машин в данной виртуальной подсети была возможность выходить в сеть Интернет (после создания виртуального маршрутизатора);
 3. **Встроенный сервер DHCP** - виртуальный сервис DHCP будет работать только в данной сети и не будет доступен для других сетей;
 4. При необходимости задать **Пулы IP-адресов** (в данном случае у всех создаваемых виртуальных машин будет статические IPv4 адреса из данной подсети)
 5. Указать **Серверы DNS** и нажать **Добавить**.

Добавить подсеть IPv4

CIDR

192.168.1.0/24

Шлюз (необязательно)

192.168.1.1

☒ Встроенный сервер DHCP

Пулы IP-адресов

Добавить

Серверы DNS

Добавить

77.88.8.8

Отмена

Добавить

- После создание Подсети необходимо нажать **Далее**:

Создать виртуальную сеть

• Конфигурация сети

• Управление IP-адресами

• Сводка

Настройте параметры сети для управления адресами IPv4 и IPv6.

Подсети

Добавить

192.168.1.0/24

Назад

Далее

- На этапе **Сводка** необходимо проверить конфигурацию виртуальной сети и нажать **Создать виртуальную сеть**:

Создать виртуальную сеть

• Конфигурация сети

• Управление IP-адресами

• Сводка

Проверьте конфигурацию виртуальной сети. При необходимости измените ее, вернувшись на предыдущие шаги.

Тип	Виртуальная (на основе VXLAN)
Имя	cloud-net
Подсеть IPv4	
Версия IP подсети	IPv4
CIDR	192.168.1.0/24
Встроенный сервер DHCP	Включено
Шлюз	192.168.1.1
Серверы DNS	77.88.8.8

Назад

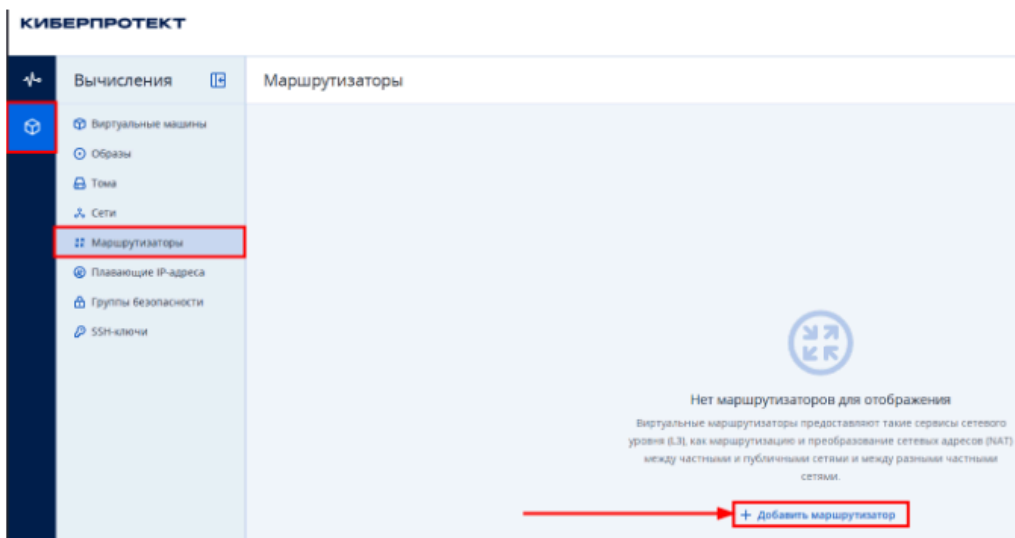
Создать виртуальную сеть

- Результат успешного создания виртуальной сети с именем **cloud-net** и подсетью IPv4 **192.168.1.0/24**:

Сети

	Поиск						
☐	Имя ↑	Управление IP-адресами	Тип ↓	CIDR	Шлюз	DHCP	
☐	public	Включено	Физическая			Отключено	...
☐	cloud-net	Включено	Виртуальная	192.168.1.0/24	192.168.1.1	Включено	...

- Создаём виртуальный маршрутизатор с произвольным именем, например **cloud-rtr**:
 - Перейдите в **ВЫЧИСЛЕНИЯ**;
 - Выберите раздел **Маршрутизаторы**;
 - Нажмите **Добавить маршрутизатор**:



- Во время добавления виртуального маршрутизатора необходимо заполнить следующие поля:
 1. Указать **Имя**;
 2. Выбрать сеть, через которую будут предоставляться доступ к публичным сетям (в данном случае сеть с именем **public**);
 3. **SNAT** - при включении SNAT маршрутизатор позволит обмениваться трафиком с публичными сетями и Интернетом;
 4. В поле **Добавить внутренние интерфейсы** необходимо выбрать ранее созданную виртуальную сеть с именем **cloud-net**;
 5. Нажать **Создать**.

Добавить виртуальный маршрутизатор... ✕

Имя
cloud-rtr

Укажите сеть, через которую будет предоставляться доступ к публичным сетям.

Сеть
public

☒ SNAT ?

Добавить внутренние интерфейсы

[+ Добавить](#)

cloud-net: 192.168.1.0/24

[Отмена](#)

[Создать](#)

- Результат успешного создания виртуального маршрутизатора с именем **cloud-rtr**:

Маршрутизаторы

Поиск				
☐	Имя ↑	Статус ↓	Внешняя сеть	SNAT
☐	 cloud-rtr	● Запущена	—	Включено

- Создаём виртуальный инстанс с именем **Cloud-ADM**:
 - Перейдите в **ВЫЧИСЛЕНИЯ**;
 - Выберите раздел **Виртуальные машины**;
 - Нажмите **Создать виртуальную машину**.

Вычисления  Виртуальные машины

 Виртуальные машины

- Образы
- Томы
- Сети
- Маршрутизаторы
- Плавающие IP-адреса
- Группы безопасности
- SSH-ключи

Нет виртуальных машин для отображения
Созданные вами виртуальные машины будут отображены здесь.

[+ Создать виртуальную машину](#)

- В окне создания виртуальной машины необходимо указать **Имя: Cloud-ADM**, в соответствии с требованиями задания, и в разделе **Образ** нажать **Укажите**:

Создать виртуальную машину ✕

Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Имя

Развернуть из: ☒ Образ ☐ Том

Образ	Укажите	
Томы	Укажите	
Тип VM	Укажите	
Сетевые интерфейсы	Укажите	

[Расширенные настройки >](#)

[Отмена](#) [Развернуть](#)

- На этапе **Образы** необходимо выбрать соответствующий требованиям задания образ для инстанса **Cloud-ADM** и нажать **Готово**:

Образы



Имя	Тип	Мин. раз...	Тип ОС	Размер
 alt-p11-cloud-x86_64.qcow2	Шаблон	2 ГиБ	Generic Linux	386 МиБ
 alt-workstation-11.0-p11-cloud-x86_64.qcow2	Шаблон	13 ГиБ	Generic Linux	5 ГиБ
 cirros	Шаблон	1 ГиБ	Generic Linux	20 МиБ

[Отмена](#)[Готово](#)

- По требованиям задания диск у **Cloud-ADM** должен быть размером в **30 ГиБ**, выбрать редактирование данного **Тома**:

Создать виртуальную машину



Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Cloud-ADM

Развернуть из:



Образ



Том

Образ

alt-workstation-11.0-p11-cloud-x86_64.qcow2



Тома

Загрузочный том 13 ГиБ, default

Загрузочный



Тип ВМ

Укажите



Сетевые интерфейсы

Укажите



SSH-ключ (необязательно)

Укажите



Скрипт настройки (необязательно)

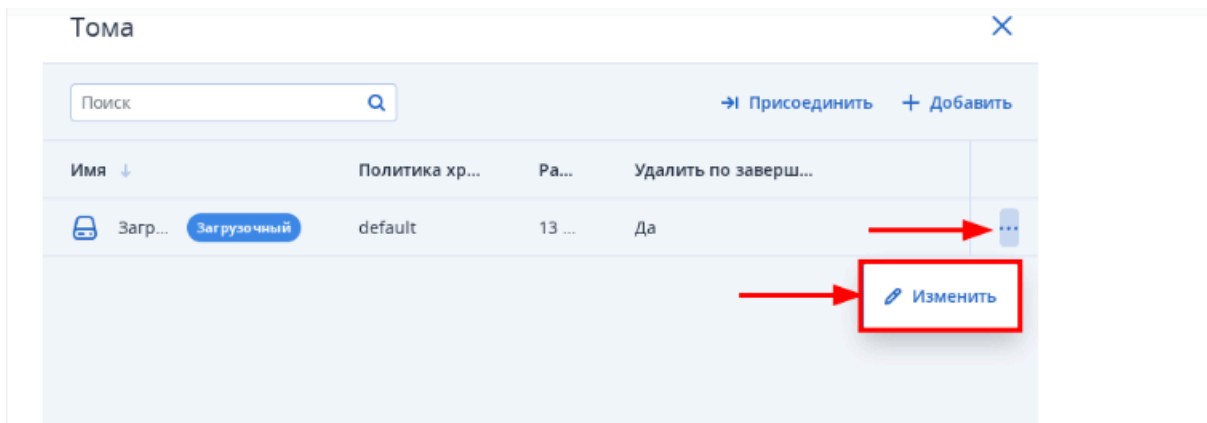
Укажите



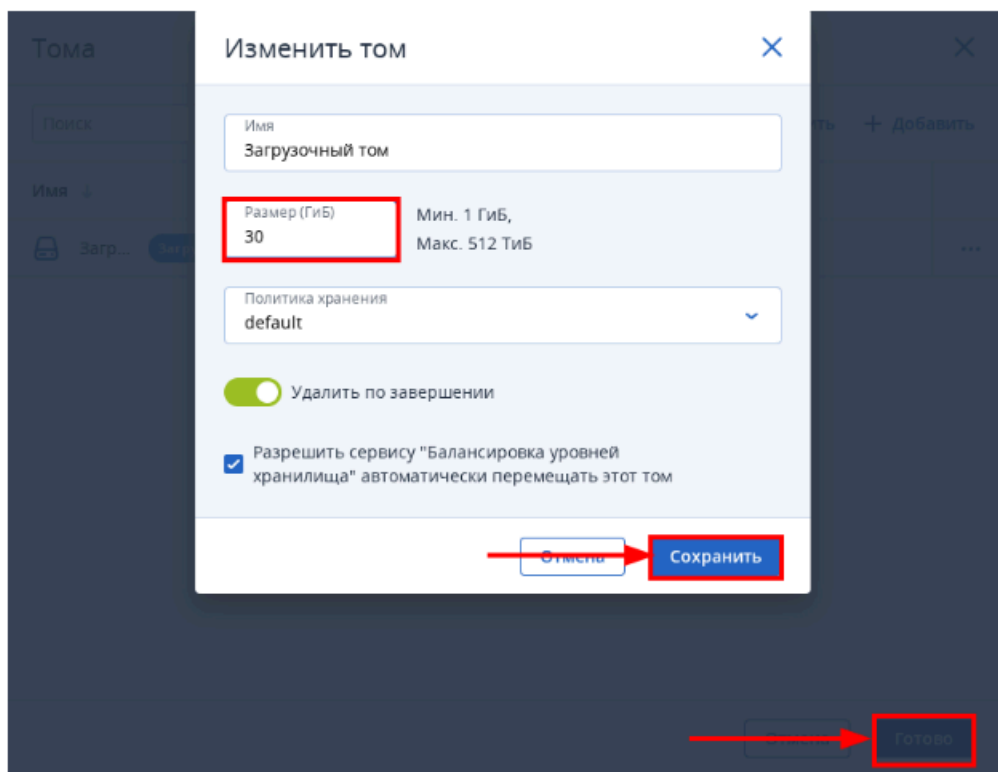
Расширенные настройки >

[Отмена](#)[Развернуть](#)

- Нажать **Изменить**:



- Задать **Размер** в 30 ГиБ, затем выбрать **Сохранить** и нажать **Готово**:



- В окне создания виртуальной машины необходимо указать **Тип ВМ**, в соответствии с требованиями задания, нажать **Укажите**:

Создать виртуальную машину



Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Имя
Cloud-ADM|

Развернуть из: ☒ Образ ☐ Том

Образ	alt-workstation-11.0-p11-cloud-x86_64.qcow2	
Томы	Загрузочный том — 30 ГиБ, default Загрузочный	
Тип VM	Укажите	
Сетевые интерфейсы	Укажите	
SSH-ключ (необязательно)	Укажите	
Скрипт настройки (необязательно)	Укажите	

Расширенные настройки >

Отмена

Развернуть

- Выбрать **Тип VM**, который будет соответствовать **2 вЦП и 4 ГиБ Память** и нажать **Готово**:

Тип VM



Поиск

Имя ↓	вЦП ↓	Память	Размещение
mini	1	1 ГиБ	—
tiny	1	512 МиБ	—
small	1	2 ГиБ	—
medium	2	4 ГиБ	—
large	4	8 ГиБ	—
xlarge	8	16 ГиБ	—

Отмена

Готово

- В окне создания виртуальной машины необходимо указать **Сетевые интерфейсы** нажать **Укажите**:

Создать виртуальную машину

Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Имя
Cloud-ADM

Развернуть из: ☒ Образ ☐ Том

Образ	alt-workstation-11.0-p11-cloud-x86_64.qcow2	
Томы	Загрузочный том — 30 ГиБ, default Загрузочный	
Тип VM	medium — 2 вЦП, 4 ГиБ ОЗУ	
Сетевые интерфейсы	Укажите	
SSH-ключ (необязательно)	Укажите	
Скрипт настройки (необязательно)	Укажите	

Расширенные настройки >

Отмена

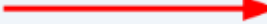
Развернуть

- Для добавления сетевого интерфейса необходимо нажать **Добавить**:

Сетевые интерфейсы



Нет подключенных сетевых интерфейсов



+ Добавить

Отмена

Готово

- В форме добавления сетевого интерфейса необходимо заполнить следующие поля:
 - Выбрать ранее созданную виртуальную сеть с именем **cloud-net**;
 - Задать **Основной IP-адрес** (статический) из ранее созданной виртуальной подсети **192.168.1.0/24**;
 - Нажать **Добавить**:

Добавить сетевой интерфейс



Сеть

cloud-net: 192.168.1.0/24



MAC-адрес

Автоматически



Назначить автоматически

Основной IP-адрес ⓘ



Добавить

IPv4:

192.168.1.10



Назначить автом...



Дополнительные IP-адреса ⓘ

IPv4-адреса



Добавить

Группы безопасности

default



Защита от спуфинга

Невозможно настроить защиту от спуфинга, если выбрана хотя бы одна группа безопасности.

Отмена

Добавить

- Нажать Готово:

Сетевые интерфейсы

Поиск

+ Добавить

Имя	Группы без...	IP-адреса	MAC-адрес	Защита от ...	
cloud-...	1	192.168.1.10	Автоматически	Да	...

Отмена

Готово

- В окне создания виртуальной машины необходимо указать **SSH-ключ**, но в данном конкретном примере это будет реализовано через **Скрипт Настройки** (cloud-init) нажать **Укажите**:

Создать виртуальную машину

Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Имя
Cloud-ADM

Развернуть из: ☒ Образ ☐ Том

Образ	alt-workstation-11.0-p11-cloud-x86_64.qcow2	
Тома	Загрузочный том — 30 ГиБ, default	Загрузочный
Тип VM	medium — 2 вЦП, 4 ГиБ ОЗУ	
Сетевые интерфейсы	cloud-net — Автоматически Основной IP-адрес: 192.168.1.10 Группы безопасности: 1	
SSH-ключ (необязательно)	Укажите	
Скрипт настройки (необязательно)	Укажите	

Расширенные настройки >

Отмена

Развернуть

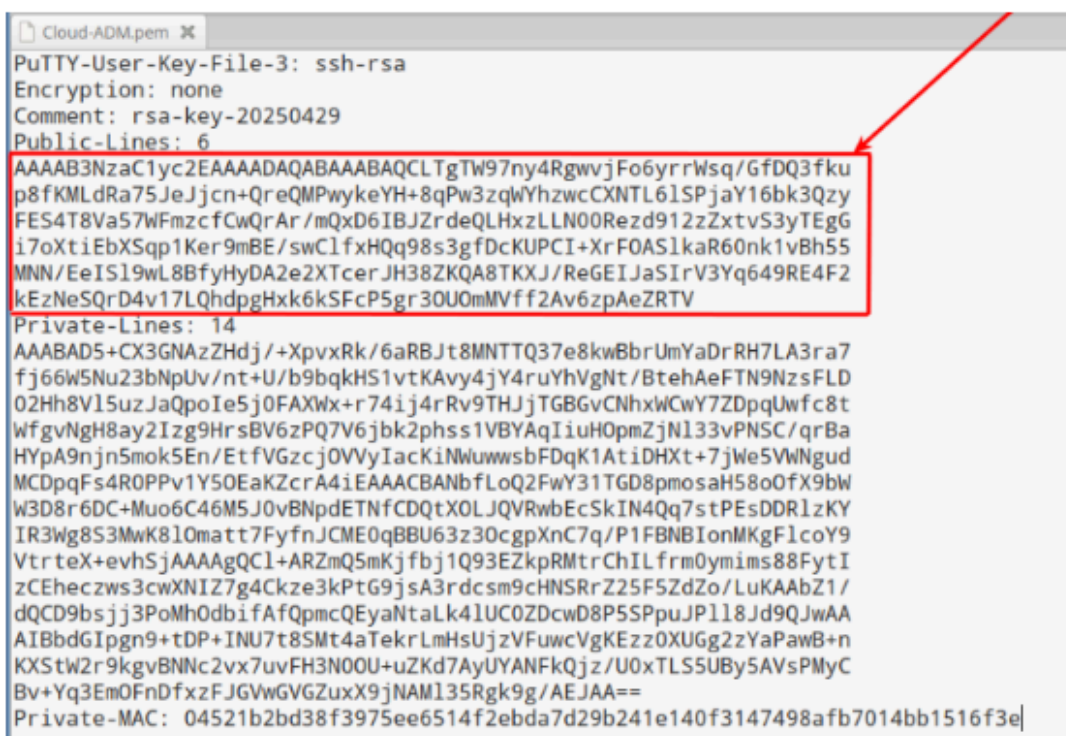
- Поместить в скрипт настройки следующее содержимое:
 - Создание пользователя altlinux с указанием для него пароля P@ssw0rd (требуется по заданию, для дальнейшего подключения по RDP, а также подойдет для доступа через локальную консоль);
 - Разрешение использования sudo без пароля для пользователя altlinux;
 - Передача публичной части SSH-ключа сгенерированного через puttygen

```
#cloud-config
chpasswd:
  expire: false
  users:
    - {name: altlinux, password: P@ssw0rd, type: text}
    - {name: root, password: toor, type: text}
ssh_pwauth: false
```

```
users:
  - name: altlinux
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: wheel
    shell: /bin/bash
    ssh_authorized_keys: ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAADAQABAAQCLTgTW97ny4RgwvjFo6yrrWsq/GfDQ3fku
p8fKMLdRa75JeJjcn+QreQMPwykeYH+8qPw3zqWYhzwcCXNTL6lSPjaY16bk3Qzy
FES4T8Va57WFmzcfCwQrAr/mQxD6IBJZrdeQLHxzLLN00Rezd912zZxtvS3yTEgG
i7oXtiEbXSqp1Ker9mBE/swClfxHQq98s3gfDcKUPCI+XrFOASlkaR60nk1vBh55
MNN/EeISl9wL8BfyHyDA2e2XTcerJH38ZKQA8TKXJ/ReGEIJasIrV3Yq649RE4F2
kEzNeSQRd4v17LQhdpgHxk6kSFcP5gr3OUOmMVff2Av6zpAeZRTV
```

- Публичная часть SSH-ключа берётся из файла **Cloud-ADM.pem** расположенного на Рабочем столе:



```
Cloud-ADM.pem
PuTTY-User-Key-File-3: ssh-rsa
Encryption: none
Comment: rsa-key-20250429
Public-Lines: 6
AAAAB3NzaC1yc2EAAAADAQABAAQCLTgTW97ny4RgwvjFo6yrrWsq/GfDQ3fku
p8fKMLdRa75JeJjcn+QreQMPwykeYH+8qPw3zqWYhzwcCXNTL6lSPjaY16bk3Qzy
FES4T8Va57WFmzcfCwQrAr/mQxD6IBJZrdeQLHxzLLN00Rezd912zZxtvS3yTEgG
i7oXtiEbXSqp1Ker9mBE/swClfxHQq98s3gfDcKUPCI+XrFOASlkaR60nk1vBh55
MNN/EeISl9wL8BfyHyDA2e2XTcerJH38ZKQA8TKXJ/ReGEIJasIrV3Yq649RE4F2
kEzNeSQRd4v17LQhdpgHxk6kSFcP5gr3OUOmMVff2Av6zpAeZRTV
Private-Lines: 14
AAABAD5+CX3GNAzZHdj/+XpvxRk/6aRBjt8MNTTQ37e8kwBbrUmYaDrRH7LA3ra7
fj66W5Nu23bNpUv/nt+U/b9bqkHS1vtKAvy4jY4ruYhVgNt/BtehAeFTN9NzsFLD
02Hh8Vl5uzJaQpoIe5j0FAXWx+r74ij4rRv9THjTGBGvCNhxwCwY7ZDpqUwfc8t
WfgvNgH8ay2Izg9HrsBV6zPQ7V6jBk2phss1VBAYqIiuH0pmZjN133vPN5C/qRba
HYpA9nJn5mok5En/EtfVGzCjOVVyIacKiNwuwbsbFDqK1AtiDHxt+7jWe5VWNgud
MCDpqFs4R0PPv1Y50EaKZcrA4iEAAACBANbfLoQ2FwY31TGD8pmosaH58oOfX9bw
W3D8r6DC+Muo6C46M5J0vBNpdETNfCDQtXOLJQVRwbEcSkIN4Qq7stPEsDDRlZKY
IR3Wg8S3MwK8lOmatt7FyfnJCME0qBBU63z30cgpXnC7q/P1FBNBionMKgFlcoY9
VtrteX+evhSjAAAGQCL+ARZmQ5mKjfbj1Q93EZkPRMtrChILfrm0ymims88FytI
zCEhecZws3cwXNIZ7g4Ckze3kPtG9jsA3rdcsm9cHNSRrZ25F5ZdZo/LuKAAbZ1/
dQCD9bsjj3PoMhOdbifAfQpmcQEyaNtaLk4lUC0ZDcwD8P5SPpuJP1l8Jd9QJwAA
AIBbdGIpgn9+tDP+INU7t8SMt4aTekrLmHsUjzVFuwcVgKEzz0XUGg2zYaPawB+n
KXStw2r9kgvBNNc2vx7uvFH3N0OU+uZKd7AyUYANfkQjz/U0xTLSSUBy5AVsPMYC
Bv+Yq3EmQFndfxzFJGVwGVGZuxX9jNAML35Rgk9g/AEJAA==
Private-MAC: 04521b2bd38f3975ee6514f2ebda7d29b241e140f3147498afb7014bb1516f3e|
```

- В окне создания виртуальной машины необходимо проверить заданные параметры и нажать **Развернуть**:

Создать виртуальную машину



Проверьте конфигурацию виртуальной машины. При необходимости измените ее, вернувшись на предыдущие шаги.

Имя Cloud-ADM	Развернуть из: ☒ Образ ☐ Том
Образ	alt-workstation-11.0-p11-cloud-x86_64.qcow2
Том	Загрузочный том — 30 ГиБ, default Загрузочный
Тип ВМ	medium — 2 вЦП, 4 ГиБ ОЗУ
Сетевые интерфейсы	cloud-net — Автоматически Основной IP-адрес: 192.168.1.10 Группы безопасности: 1
SSH-ключ (необязательно)	Укажите
Скрипт настройки (необязательно)	Задан

Расширенные настройки >

Отмена

Развернуть

- Результат успешного развёртывания инстанса **Cloud-ADM**:
- Результат успешного развёртывания инстанса **Cloud-ADM**:

Виртуальные машины

Фильтр		Поиск							+ Создать виртуальную машину	
☐	Имя ↑	Статус ↓	IP-адрес	вЦП ↓	ОЗУ ↓	Хранилище ↓	Томы		⚙	
☐	Cloud-ADM	Запущена	192.168.1.10	2	4 ГиБ	0 байт	1		...	

- При нажатие на созданный инстанс **Cloud-ADM** можно проверить параметры в соответствии с требованиями задания:

Свойства		
Имя	Cloud-ADM	
Образ	alt-workstation-11.0-p11-cloud-x86_64.qcow2	
Режим эмуляции ЦП	Использовать настройки кластера: host-model	
Тома	Том 1 (30 ГиБ, default) Загрузочный	
Тип VM	medium (2 ЦП, 4 ГиБ ОЗУ)	
Горячее подключение ЦП и ОЗУ	Отключено	
Автоматические миграции DRS	Включено	

- Создаём Плавающий IP-адрес для возможности удалённого подключения к инстансу **Cloud-ADM** по **SSH**:
 1. Перейдите в **ВЫЧИСЛЕНИЯ**;
 2. Выберите раздел **Плавающие IP-адреса**;
 3. Нажмите **Добавить плавающий IP-адрес**:

КИБЕРПРОТЕКТ

 Вычисления
  Плавающие IP-адреса

 Виртуальные машины
  Образы
  Тома
  Сети
  Маршрутизаторы
  **Плавающие IP-адреса**
 Группы безопасности
  SSH-ключи



Нет плавающих IP-адресов

Плавающие IP-адреса, назначенные виртуальным машинам или балансировщикам нагрузки, будут отображены здесь. Любой виртуальной машине или балансировщику нагрузки, имеющим фиксированный частный IP-адрес, можно назначить плавающий публичный IP-адрес для выхода во внешние сети, например, Интернет.

 **Добавить плавающий IP-адрес**

- В форме добавления плавающего IP-адреса необходимо заполнить следующие поля:
 1. Указать **Сеть** откуда будет взят плавающий IP-адрес;
 2. Выбрать **инстанс**, который необходимо связать с плавающим IP-адресом;
 3. Выбрать **IP-адрес** назначенный на выбранный инстанс;
 4. Нажать **Добавить**:

Добавить плавающий IP-адрес

Выберите сеть, откуда будет взят плавающий IP-адрес.

Сеть

public

Выберите приватный IP-адрес виртуальной машины или балансировщика нагрузки, который необходимо связать с плавающим IP-адресом.

Cloud-ADM

IP-адрес

(Основной) 192.168.1.10

Отмена

Добавить

- Результат успешного добавление **Плавающего IP-адреса**:

Плавающие IP-адреса

Поиск					
☐	IP-адрес	Статус	Сеть	Назначен	IP-адрес VM
☐	 172.20.102.183	 Запущен	public	Cloud-ADM	192.168.1.10

- Проверить связность в инстансом через **Плавающий IP-адрес**:

```
PING 172.20.102.183 (172.20.102.183) 56(84) bytes of data:
64 bytes from 172.20.102.183: icmp_seq=1 ttl=61 time=2.45 ms
64 bytes from 172.20.102.183: icmp_seq=2 ttl=61 time=1.93 ms
64 bytes from 172.20.102.183: icmp_seq=3 ttl=61 time=1.14 ms

--- 172.20.102.183 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.138/1.840/2.453/0.540 ms
```

- Средствами **Скрипта настройки** был создан пользователь **altlinux** с паролем **P@ssw0rd**
- Можно воспользоваться встроенной консолью для проверки доступа к инстансу:

Виртуальные машины

Cloud-ADM

Поиск

Имя ↑

Статус ↓

Cloud-ADM

Запущена

Консоль

Перезагрузить

Выключить

Обзор

Сведения

Статус

Время создания

SSH-ключ

Идентификатор виртуальной машины

Запущена

24 нояб

Не указа

035f7a25

- Выполнить вход из-под пользователя **altlinux** с паролем указанным через **Скрипт настройки** на этапе создания инстанса:

Cloud-ADM / Консоль

Отправить комбинацию клавиш

Выберите действие

Nov 24 11:05 AM

Type to search

Nov 24 11:05 AM

Nov 24 11:05 AM

- Также, если посмотреть сетевые параметры, то можно будет увидеть следующее:

```
altlinux@cloud-adm: /home/altlinux
[altlinux@cloud-adm ~]$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:66:10:a6 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 192.168.1.10/24 brd 192.168.1.255 scope global dynamic noprefixroute
        valid_lft 8674sec preferred_lft 8674sec
    inet6 fe80::d89:afab:c9b:ac3/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[altlinux@cloud-adm ~]$
```

- Если по каким-то причинам необходимо будет перезагрузить данную виртуальную машину, то после перезагрузки мы получим:

```
altlinux@cloud-adm: /home/altlinux
[altlinux@cloud-adm ~]$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN group default qlen 1000
    link/ether fa:16:3e:66:10:a6 brd ff:ff:ff:ff:ff:ff
[altlinux@cloud-adm ~]$
```

- За настройку сети в Альт Рабочей станции оптимизированной под использование в облачных провайдерах отвечает **netplan**, его конфигурационный файл выглядит следующим образом:

```
altlinux@cloud-adm: /home/altlinux

network:
  version: 2
  ethernets:
    eth0:
      match:
        macaddress: "fa:16:3e:66:10:a6"
      dhcp4: true
      set-name: "eth0"
      mtu: 1450

"/etc/netplan/50-cloud-init.yaml" 9L, 155B
```

- netplan в качестве бэкенда может использовать и другие сетевые подсистемы, например NetworkManager, для этого необходимо явно указать это в данном конфигурационном файле:

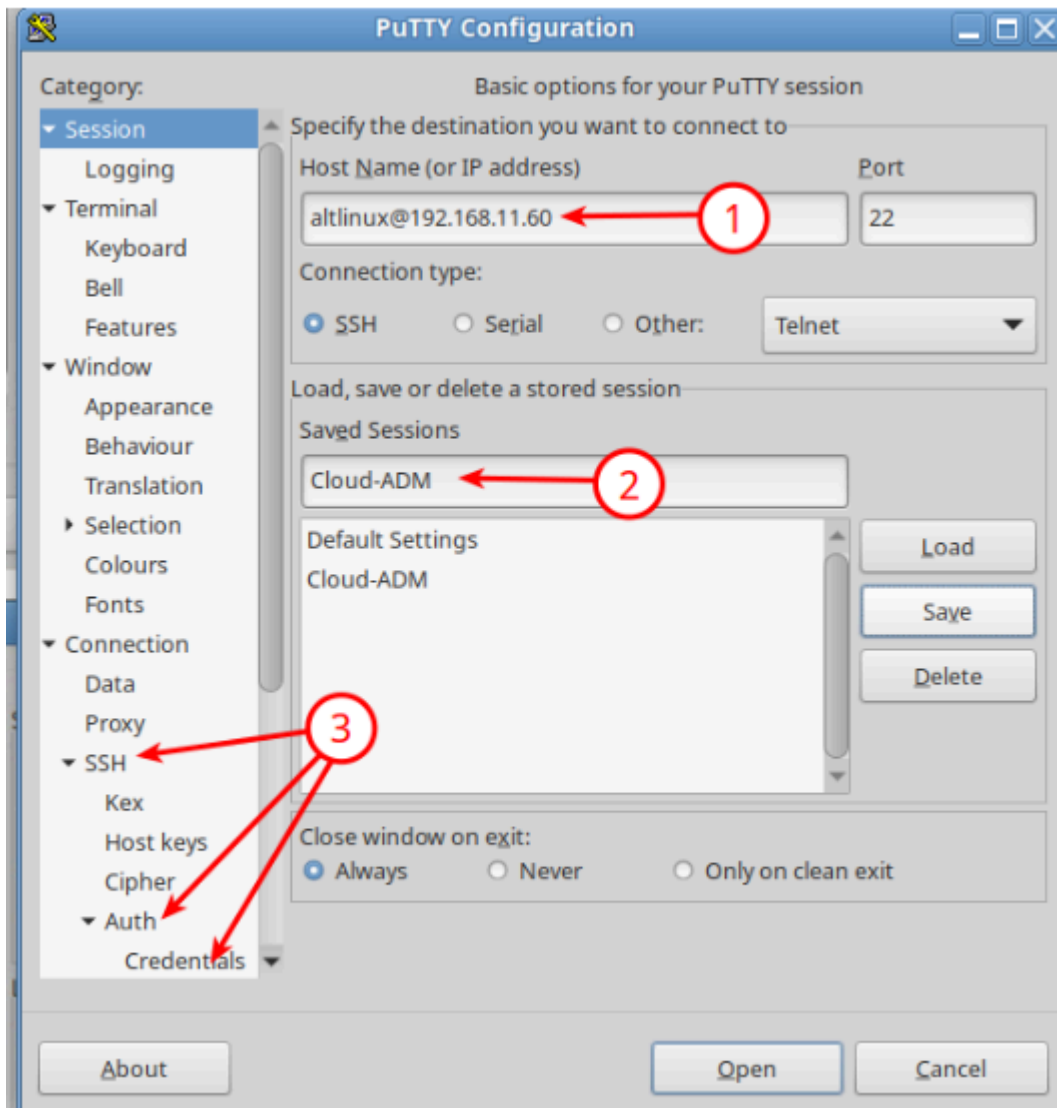
-

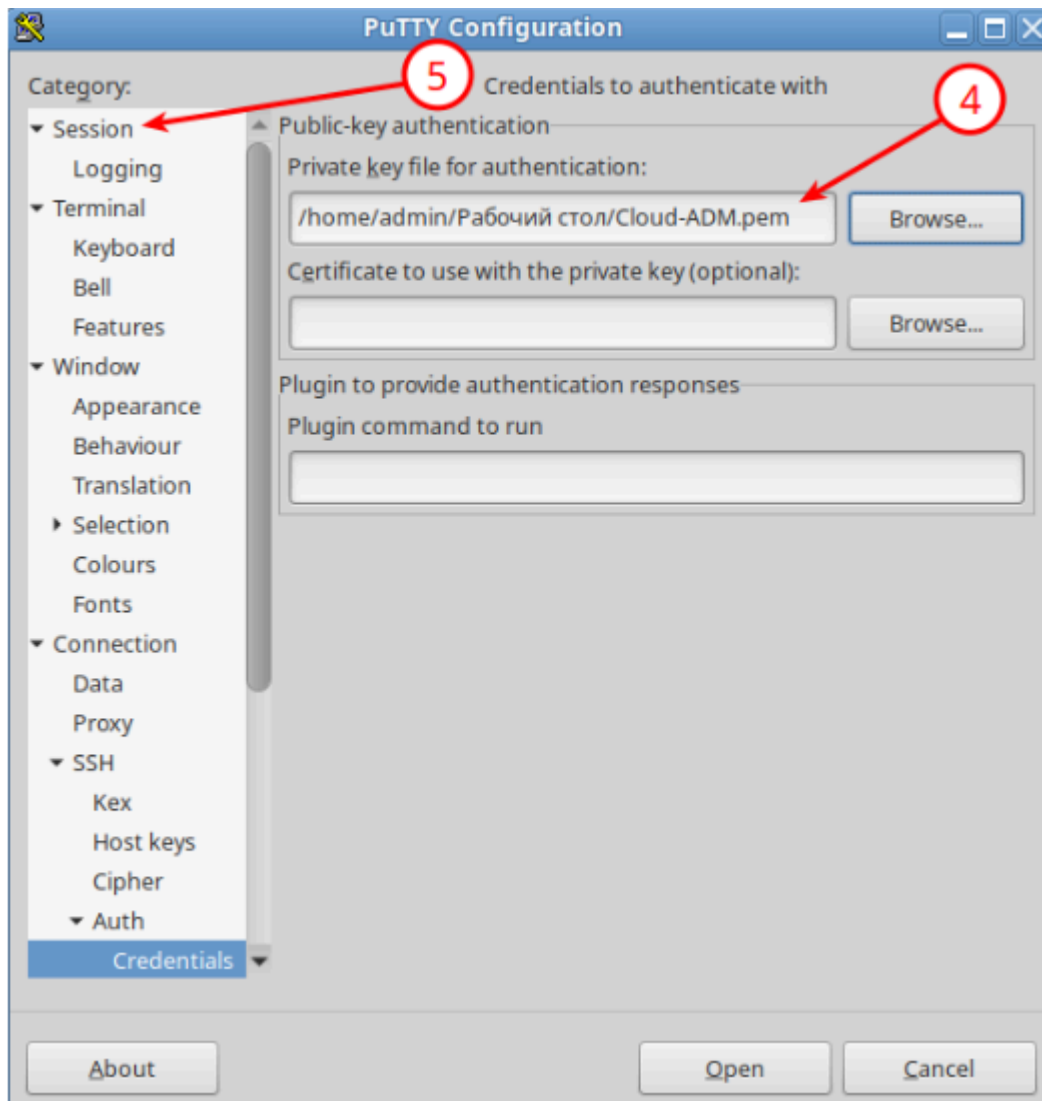
```
network:
  version: 2
  render: NetworkManager
  ethernets:
    eth0:
      match:
        macaddress: "fa:16:3e:66:10:a6"
      dhcp4: true
      set-name: "eth0"
      mtu: 1450
```

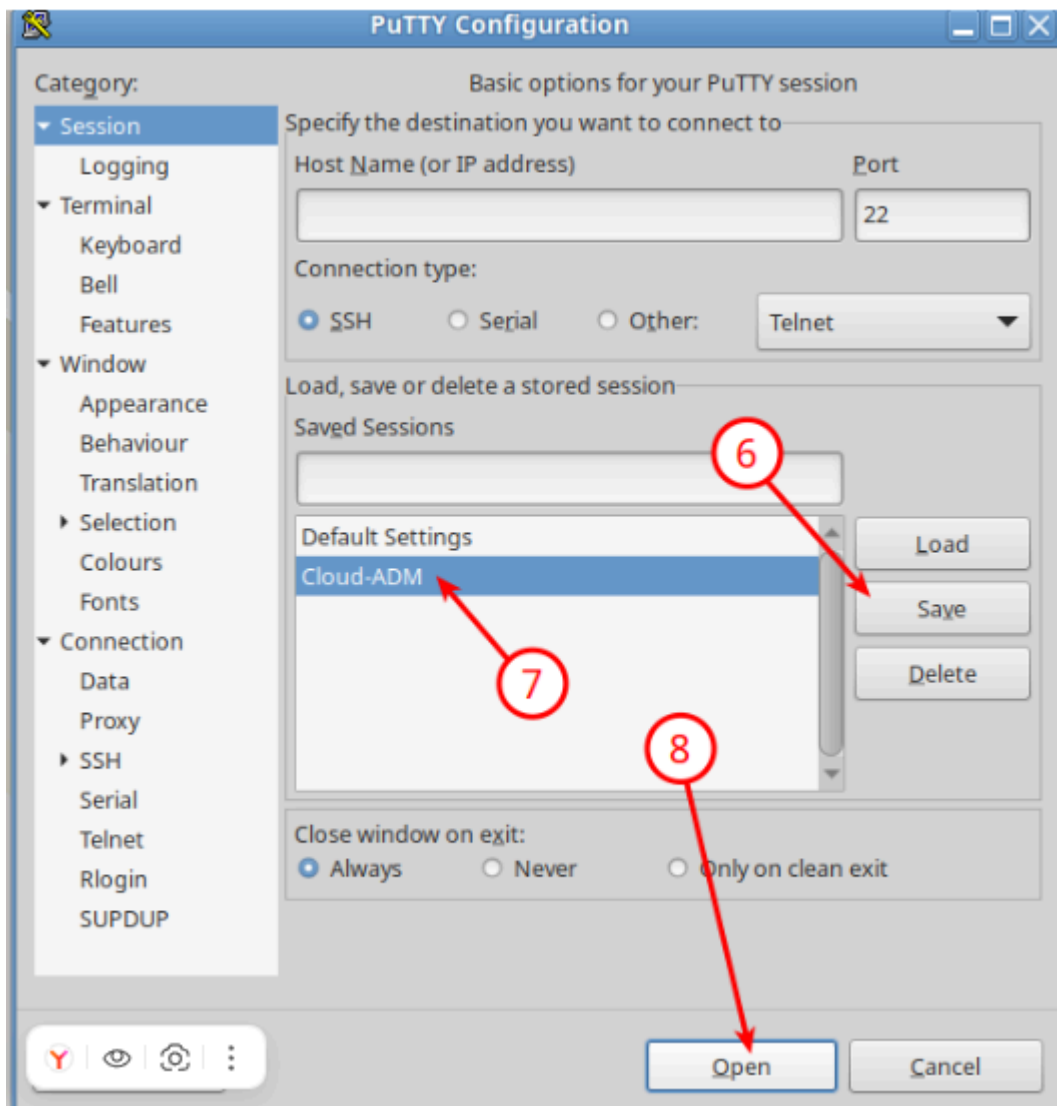
- В результате, при последующих перезагрузках данной виртуальной машины IP-адрес следовать не будет:

```
altlinux@cloud-adm: /home/altlinux
[altlinux@cloud-adm ~]$ ip -c a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc fq_codel state UP group default qlen 1000
    link/ether fa:16:3e:66:10:a6 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname ens3
    inet 192.168.1.10/24 brd 192.168.1.255 scope global dynamic noprefixroute eth0
        valid_lft 84794sec preferred_lft 84794sec
    inet6 fe80::5d99:afa5:670b:ec78/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[altlinux@cloud-adm ~]$
```

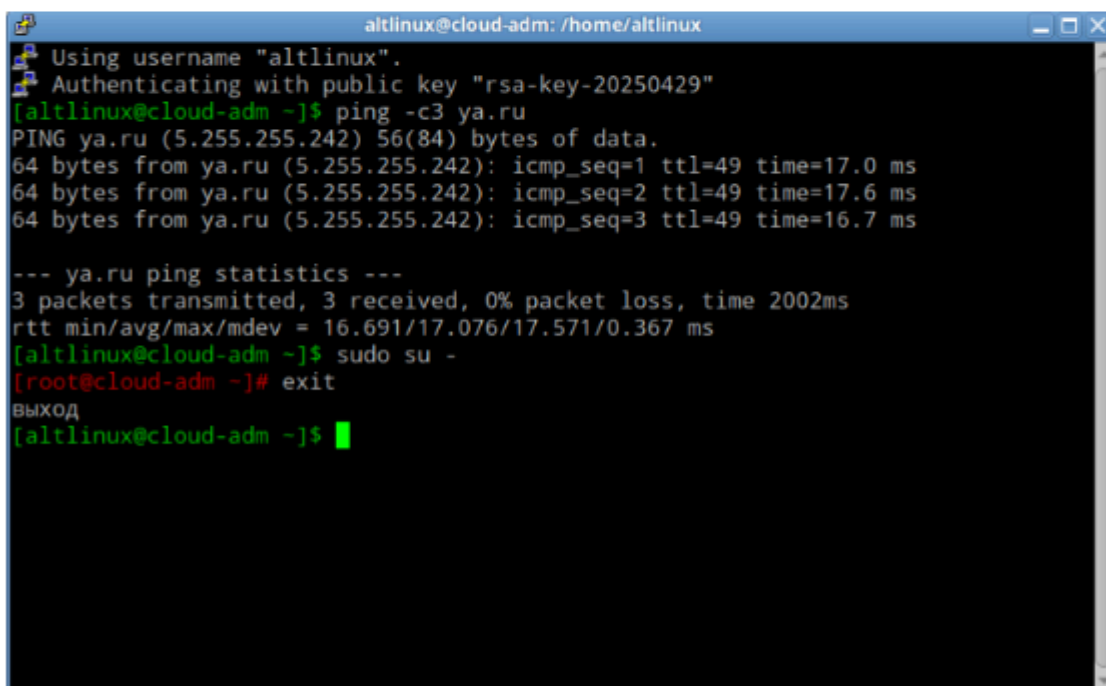
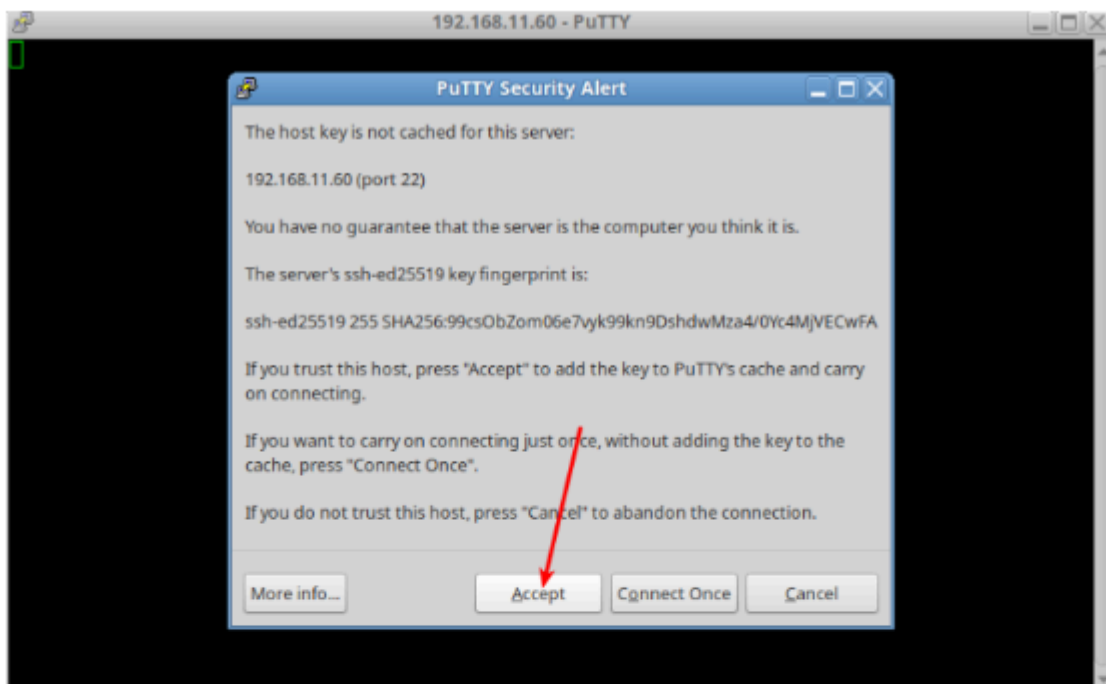
- Для подключения к Cloud-ADM необходимо создать профиль в putty с одноимённым именем (по требованию задания):
 - Используя для подключения Плавающий IP адрес и ранее сгенерированную ключевую пару на Рабочем столе







- Результат подключения к инстансу **Cloud-ADM** через **putty**:



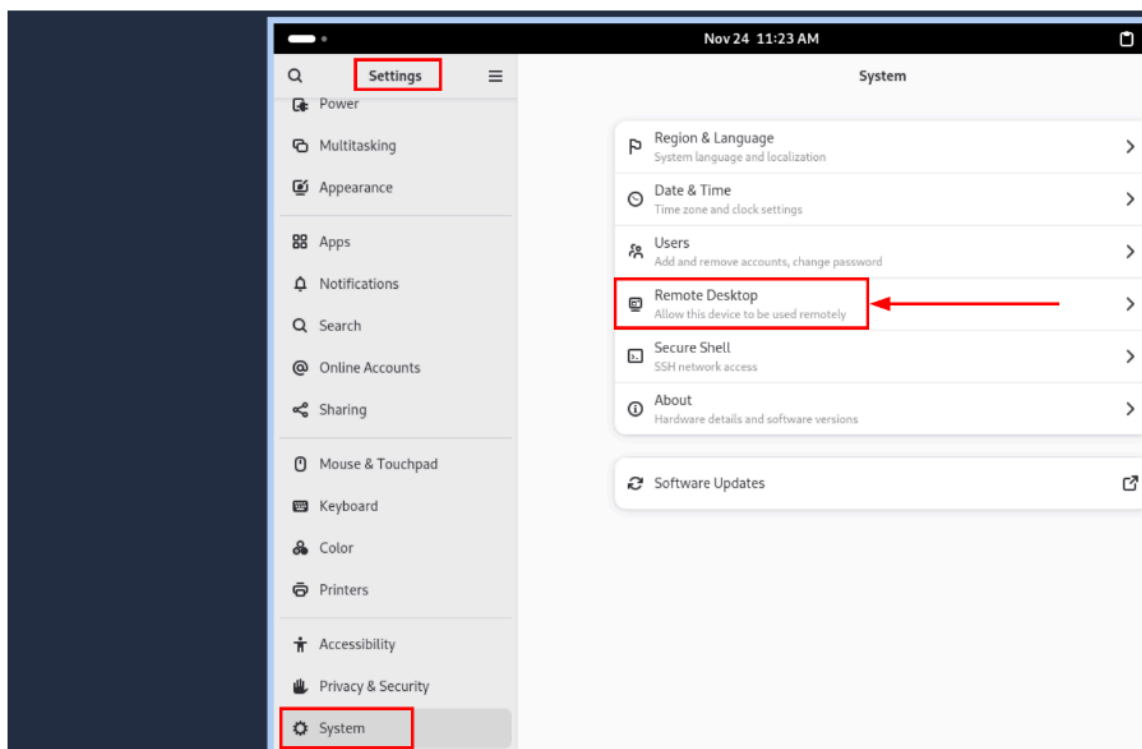
- Для возможности удаленного подключения (RDP) необходимо разрешить общий доступ к **fuse**:

```
sudo control fusermount public
```

- Используя встроенную консоль, необходимо перейти на инстансе Cloud-ADM в Settings -> System -> Remote Desktop:

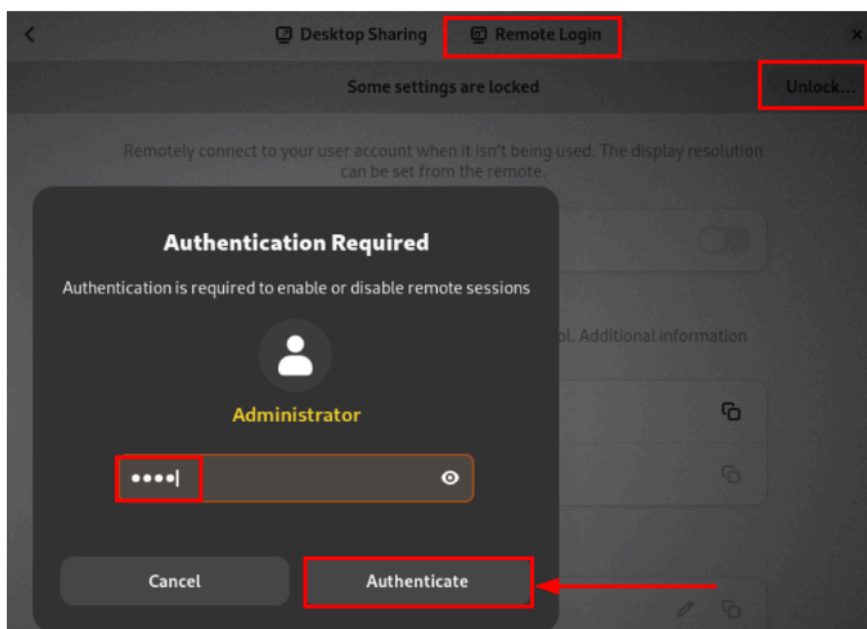
Cloud-ADM / Консоль

Отправить



- Перейти в Remote Login и нажать Unlock, ввести пароль от суперпользователя root (пароль для пользователя root, так же задавался средствами Скрипта настройки:

- Перейти в Remote Login и нажать Unlock, ввести пароль от суперпользователя root (пароль для пользователя root, так же задавался средствами Скрипта настройки:



- Включить Remote Login и указать Login Details:

 Desktop Sharing

 Remote Login

Remotely connect to your user account when it isn't being used. The display resolution can be set from the remote.

Remote Login



How to Connect

Use a remote desktop app to connect using the RDP protocol. Additional information about the device's network location may also be required.

Hostname
cloud-adm

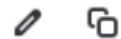


Port
3389



Login Details

Username
altlinux



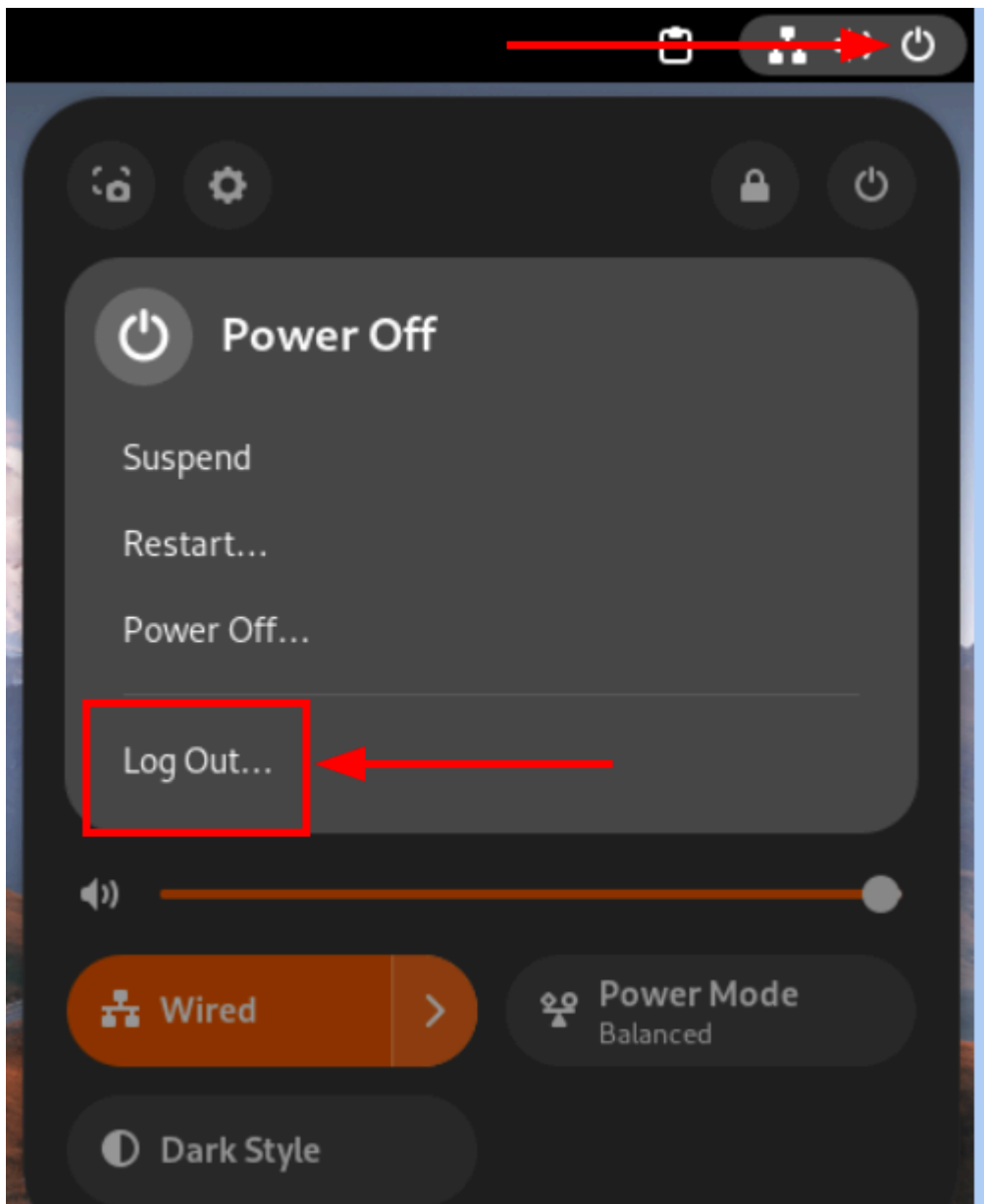
Password
P@ssw0rd



Generate New Password

Verify Encryption

- Разлогиниться из встроенной консоли:
-



- Для применения изменений необходимо перезапустить службу `gnome-remote-desktop`:

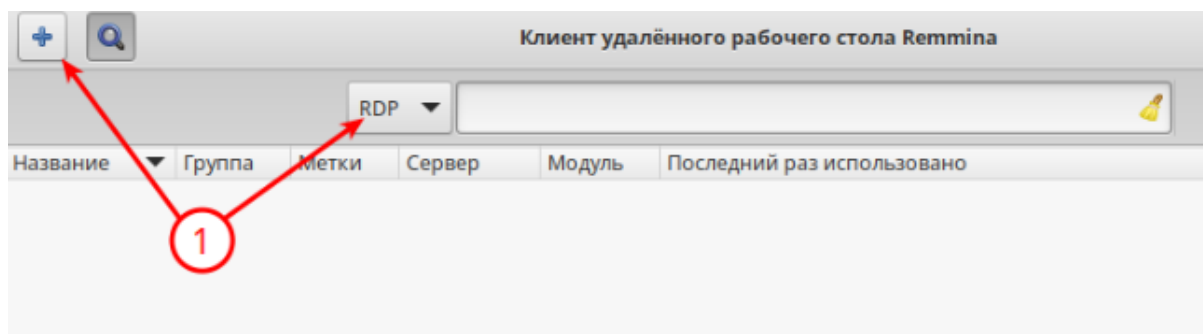
```
sudo systemctl daemon-reload
```

```
sudo systemctl restart gnome-remote-desktop
```

- Проверить удалённый доступ по RDP, например с использованием Remmina:



- Создаём профиль Cloud-ADM.rdp для подключения к удалённому рабочему столу инстанса Cloud-ADM:
 - Для этого используем remmina и нажимаем Добавить новый профиль соединения:



- Заполняем создаваемый профиль, указав в качестве Сервера - Плавающий-IP:

Профиль удаленного соединения

Название: Cloud-ADM (2)

Группа:

Метки:

Протокол: RDP - Протокол удалённого рабочего стола (3)

Основные | Дополнительные | Поведение | SSH туннель | Примечания

Сервер: 192.168.11.60 (4)

Имя пользователя: altlinux (5)

Пароль: P@ssw0rd (6)

Домен:

Общая папка:

☐ Ограниченный режим администратора

Хеш пароля:

☐ Поддержка леворучной мыши ☐ Отключить плавную прокрутку

☐ Разрешить несколько мониторов ☐ Растянуть экран на несколько мониторов

Отмена | Сохранить "по умолчанию" | Сохранить (7) | Подключить

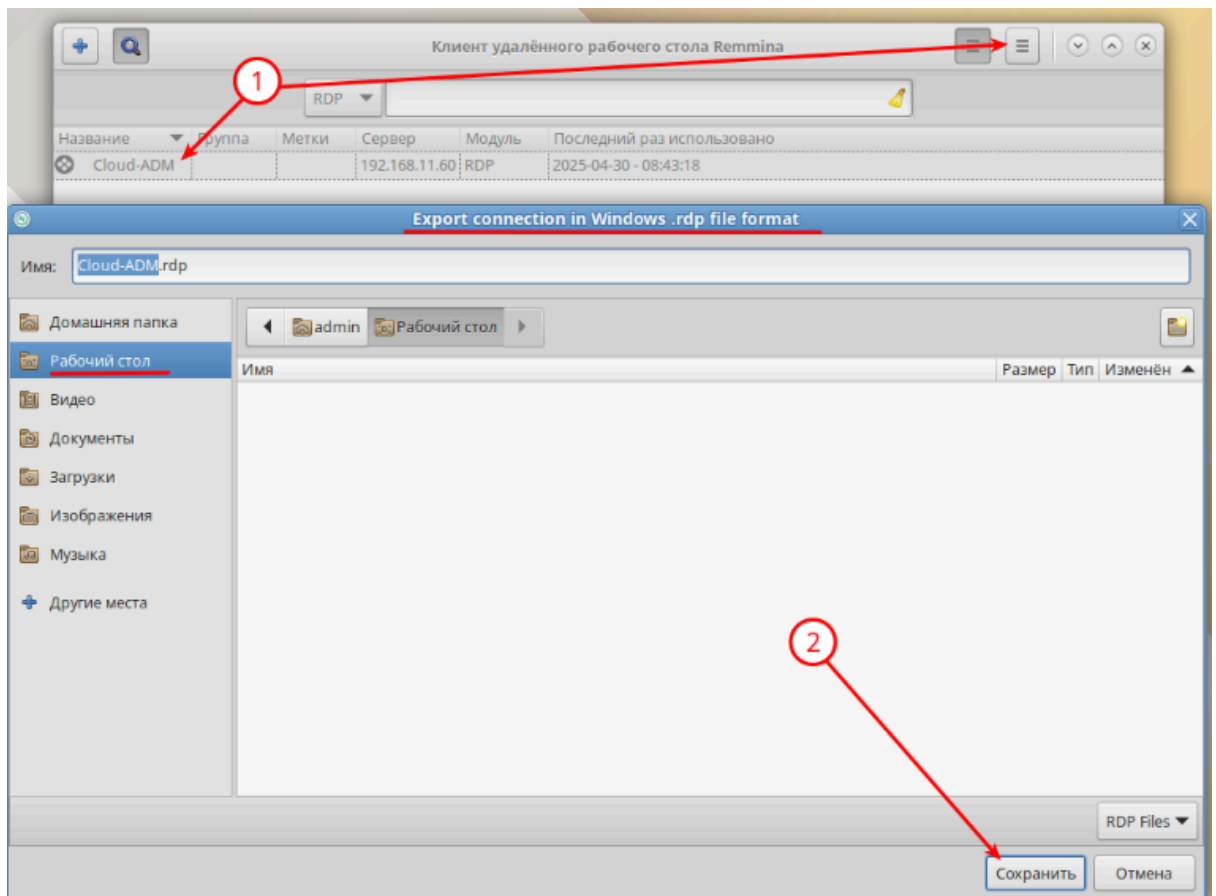
- Результат:

Клиент удалённого рабочего стола Remmina

RDP

Название	Группа	Метки	Сервер	Модуль	Последний раз использовано
Cloud-ADM			192.168.11.60	RDP	

- Далее экспортируем созданный профиль в файл формата .rdp с сохранением на рабочий стол:



Cloud-ADM:

- Устанавливаем **wget** и **unzip**:

```
sudo apt-get update && sudo apt-get install -y wget unzip
```

- Скачиваем архив (последней стабильной версии) с **Terraform** с одного из доступных зеркал:
 - с зеркала **Яндекса**:

```
wget https://hashicorp-releases.yandexcloud.net/terraform/1.14.0/terraform_1.14.0_linux_amd64.zip
```

- или с зеркала **VK Cloud**:

```
wget https://hashicorp-releases.mcs.mail.ru/terraform/1.14.0/terraform_1.14.0_linux_amd64.zip
```

- Распаковываем его в каталог **/usr/local/bin**:

```
sudo unzip terraform_1.14.0_linux_amd64.zip -d /usr/local/bin/
```

- Проверяем:

```
[altlinux@cloud-adm ~]$ terraform --version
Terraform v1.14.0
on linux_amd64
[altlinux@cloud-adm ~]$
```

Cloud-ADM:

- Установим пакеты `python3-module-openstackclient` и `python3-module-pip`:

```
sudo apt-get install -y python3-module-openstackclient python3-module-pip
```

- Проверяем:

```
[altlinux@cloud-adm ~]$ openstack --version  
openstack 6.2.0  
[altlinux@cloud-adm ~]$
```

Cloud-ADM:

- Установка ansible средствами pip:

```
pip3 install ansible
```

- Добавляем путь в переменную окружения `PATH`:

```
export PATH=/home/altlinux/.local/bin:$PATH
```

- Проверяем:

```
[altlinux@cloud-adm ~]$ ansible --version  
ansible [core 2.20.0]  
config file = None  
configured module search path = ['/home/altlinux/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']  
ansible python module location = /home/altlinux/.local/lib/python3/site-packages/ansible  
ansible collection location = /home/altlinux/.ansible/collections:/usr/share/ansible/collections  
executable location = /home/altlinux/.local/bin/ansible  
python version = 3.12.7 (main, Oct 2 2024, 04:23:59) [GCC 13.2.1 20240128 (ALT Sisypheus 13.2.1-alt3)] (/usr/bin/python3)  
jinja version = 3.1.4  
pyyaml version = 6.0.1 (with libyaml v0.2.5)  
[altlinux@cloud-adm ~]$
```

- На виртуальной машине Cloud-ADM создайте скрипт «`deploy_project_01.sh`».
- В качестве домашней директории используйте путь `/home/altlinux/Projects/Project_01`.
- Скрипт должен реализовывать следующий функционал:
 - Автоматическое развёртывание виртуальных машин в соответствие с топологией (см. Топология `Project_01`).
 - Характеристики виртуальных машин: 1vCPU, 1 ГБ ОЗУ, 10 ГБ размер диска.
 - Образ операционной системы: `alt-p11-cloud-x86_64.qcow2`.
 - Для виртуальной машины `haproxy01` должен создаваться и ассоциироваться Плавающий-IP.
- На виртуальной машине Cloud-ADM создайте скрипт «`configure_project_01.sh`».
- В качестве домашней директории используйте путь `/home/altlinux/Projects/Project_01`.

- f. Скрипт должен реализовывать следующий функционал:
1. Установку необходимых компонентов для запуска приложения (см. Приложение Project_01) на виртуальных машинах: game01, game02, game03 в виде Docker-контейнеров.
 2. Образ приложения нужно сделать легковесным через мультитейджинг.
 3. На виртуальной машине haproxy01 настройку распределения входящих запросов через haproxy.
 4. Настройку доступа к просмотру собранной статистики в веб-интерфейсе с виртуальной машины Cloud-ADM по haproxy01.dev.au.team /haproxy?stats
 5. Приложение должно быть доступно с виртуальной машины Cloud-ADM из веб-браузера по именам: game01.dev.au.team, game02.dev.au.team и game03.dev.au.team на 80 порту.
 6. Приложение должно быть доступно из вне по Плавающему-IP виртуальной машины haproxy01 на порту 443.
 7. С виртуальной машины Cloud-ADM доступ к приложению должен быть по <https://game.au.team>.
 8. Проблем с сертификатом возникать не должно.
- g. На виртуальной машине Cloud-ADM создайте скрипт «destroy_project_01.sh».
- h. В качестве домашней директории используйте путь /home/altlinux/Projects/Project_01.
- i. Скрипт должен реализовывать следующий функционал:
1. Удалять все автоматически созданные ресурсы через скрипт deploy_project_01.sh.
 2. Если скрипт destroy_project_01.sh не реализован, участник не получит баллов за данный пункт задания, а эксперты вручную выполнят удаление ресурсов.

Вариант реализации:

Cloud-ADM:

Все последующие действия выполняются из-под пользователя altlinux (не root)

- Создаём директорию /home/altlinux/Projects/Project_01:

```
mkdir -p /home/altlinux/Projects/Project_01
```

- Для скрипта **deploy_project_01.sh** будет использоваться **Terraform**
- Создадим директорию **terraform** в ранее созданной директории для **Project_01**:

```
mkdir /home/altlinux/Projects/Project_01/terraform
```

- Перейдём в директорию /home/altlinux/Projects/:

```
cd /home/altlinux/Projects/
```

- Создадим файл с именем **cloudinit.conf** (имя произвольное) и поместим в него следующее содержимое:
 - указав переменные окружения для работы **OpenStack CLI** с API облака
 - в данном конкретном примере подразумевается что у участника:
 1. Внешний доступ к панели самообслуживания: <https://cyberinfra.ssa2026.region:8800>
 2. Домен: **Competence_SiSA**
 3. Проект: **Project0**
 4. Учётная запись: **User0**
 5. Пароль от уч.записи: **P@ssw0rd**

```
export OS_AUTH_URL=https://cyberinfra.ssa2026.region:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_TYPE=password
export OS_PROJECT_DOMAIN_NAME=Competence_SiSA
export OS_USER_DOMAIN_NAME=Competence_SiSA
export OS_PROJECT_NAME=Project0
export OS_USERNAME=User0
export OS_PASSWORD=P@ssw0rd
```

- Применяем переменные окружения указанные в файле

```
source cloudinit.conf
```

- Проверяем возможность взаимодействовать чере openstack-cli с облаком:
 - например, выведем список инстансов командой **openstack --insecure server list**:

```
[altlinux@cloud-adm Projects]$ openstack --insecure server list
```

ID	Name	Status	Networks	Image	Flavor
035f7a29-ce73-4989-a180-0b6e0499eae2	Cloud-ADM	ACTIVE	cloud-net=172.20.102.183, 192.168.1.10	N/A (booted from volume)	medium

- или сетей командой **openstack --insecure network list**:
-

ID	Name	Subnets
61845892-f9cc-4fde-962c-34b59425a74d	cloud-net	13592ca4-8782-410b-9bcc-90810ccab6fe
edd83757-8b79-4afe-b1ce-edaf4b1830b6	public	780c2cb6-d0d2-4566-bce7-f802375045fa, b77cee34-9aaf-4d78-9b86-7585df597c2d

[altlinux@cloud-adm Projects]\$

- Для корректной работы с Terraform, необходимо создать файл конфигурации зеркала
- Файл должен иметь имя `.terraformrc` и быть расположен в домашнем каталоге пользователя
- Файл `~/.terraformrc` должен содержать в себе следующее:

```
provider_installation {
  network_mirror {
    url = "https://terraform-mirror.mcs.mail.ru"
    include = ["registry.terraform.io/*/*"]
  }
  direct {
    exclude = ["registry.terraform.io/*/*"]
  }
}
```

- **Перейдём в директорию `Project_01/terraform/`:**

```
cd Project_01/terraform/
```

- Создадим файл **provider.tf** и опишем параметры для подключения к провайдеру **openstack** для работы с облаком:
 - в данном конкретном примере подразумевается что у участника:
 1. Внешний доступ к панели самообслуживания: <https://cyberinfra.ssa2026.region:8800>
 2. Домен: **Competence_SiSA**
 3. Проект: **Project0**
 4. Учётная запись: **User0**
 5. Пароль от уч.записи: **P@ssw0rd**

```

terraform {
  required_providers {
    openstack = {
      source = "terraform-provider-openstack/openstack"
      version = "2.1.0"
    }
  }
}

provider "openstack" {
  auth_url      = "https://cyberinfra.ssa2026.region:5000/v3"
  tenant_name   = "Project0"
  user_name     = "User0"
  password      = "P@ssw0rd"
  insecure      = true
}

```

- Инициализируем текущий каталог для работы с **terraform** и провайдером **openstack**:

```
terraform init
```

- Результат успешной инициализации **Terraform** провайдера:

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

```
[altlinux@cloud-adm terraform]$
```

- Чтобы с инстанса **Cloud-ADM** был доступ по SSH (для Ansible) до всех создаваемых инстансов (средствами Terraform), будет использоваться файл **cloud-init.yml**
- Создадим ключевую пару:

```
ssh-keygen -t rsa
```

- Поместим содержимое публичного ключа (**id_rsa.pub**) в файл **cloud-init.yml**:

```
cat ~/.ssh/id_rsa.pub > cloud-init.yml
```

- Откроем файл **cloud-init.yml** на редактирование и добавим в самое начало (до содержимого ключа ssh):
 - таким образом, на создаваемых в дальнейшем инстансах будут:
 - создан пользователь **altlinux** с паролем **P@ssw0rd**;
 - задан для суперпользователя **root** пароль **toor**;
 - передан публичный SSH-ключ **Cloud-ADM**.

```
#cloud-config
chpasswd:
  expire: false
  users:
    - {name: altlinux, password: P@ssw0rd, type: text}
    - {name: root, password: toor, type: text}
  ssh_pwauth: false

users:
  - name: altlinux
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: wheel
    shell: /bin/bash
    ssh_authorized_keys:
      - <СОДЕРЖИМОЕ_ПУБЛИЧНОГО_КЛЮЧА_SSH>
```

- Создадим файл vm-game.tf и опишем конфигурацию для создаваемых инстансов game01, game02 и game03:
 - для минимального написания строк кода (время на выполнения задания не резиновое, хардкод приветствуется, данный модуль не проверяет применение лучших практик для автоматизации):
 1. используем цикл по счётчику count;
 2. избавляемся от параметризации с использованием переменных (хардкодим все значения в данный файл).
 3. значение для параметра flavor_id можно узнать используя команду openstack --insecure flavor list скопировав ID Типа ВМ с 1024 RAM и 1 VCPUs;
 4. значение для параметра uuid можно узнать используя команду openstack --insecure image list скопировав ID образа alt-p11-cloud-x86_64.qcow2.

```

resource "openstack_compute_instance_v2" "game" {
  count      = "3"
  name       = "game0${count.index + 1}"
  flavor_id  = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data  = file("cloud-init.yml")

  block_device {
    uuid          = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type    = "image"
    volume_size    = "10"
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_game[count.index].id
  }
}

```

- Создадим файл `vm-haproxy01.tf` и опишем конфигурацию для создаваемого инстанса `haproxy01`:
 - значение для параметра `flavor_id` можно узнать используя команду `openstack --insecure flavor list` скопировав ID Типа VM с 1024 RAM и 1 VCPUs;
 - значение для параметра `uuid` можно узнать используя команду `openstack --insecure image list` скопировав ID образа `alt-p11-cloud-x86_64.qcow2`.

```

resource "openstack_compute_instance_v2" "haproxy" {
  name       = "haproxy01"
  flavor_id  = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data  = file("cloud-init.yml")

  block_device {
    uuid          = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type    = "image"
    volume_size    = "10"
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.haproxy.id
  }
}

```

- Создадим файл `network.tf` и опишем конфигурацию сети в соответствии с топологией для `Project_01`:
 - значение для параметра `network_id` можно узнать используя команду `openstack --insecure network list` скопировав ID виртуальной сети с именем `cloud-net`;
 - значение для параметра `subnet_id` можно узнать используя команду `openstack --insecure subnet list` скопировав ID виртуальной подсети `192.168.1.0/24`;

```
resource "openstack_networking_port_v2" "port_vm_game" {
  count      = "3"
  name       = "port_vm_game0${count.index + 1}"
  network_id = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.10${count.index + 1}"
  }
}

resource "openstack_networking_port_v2" "haproxy" {
  name       = "port_vm_haproxy"
  network_id = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.100"
  }
}
```

- Создадим файл `floatingip.tf` и опишем конфигурацию Плавающего IP-адреса в соответствии с топологией для `Project_01`:

```
resource "openstack_networking_floatingip_v2" "floatingip_haproxy" {
  pool = "public"
}

resource "openstack_networking_floatingip_associate_v2" "association_haproxy" {
  port_id      = openstack_networking_port_v2.haproxy.id
  floating_ip = openstack_networking_floatingip_v2.floatingip_haproxy.address
}
```

- Выполняем проверку синтаксиса и структуры файлов конфигурации **Terraform**:

```
terraform validate
```

```
[altlinux@cloud-adm terraform]$ terraform validate
Success! The configuration is valid.

[altlinux@cloud-adm terraform]$
```

- Перейдём в директорию `/home/altlinux/Projects/Project_01`:

```
cd /home/altlinux/Projects/Project_01
```

- Создадим файл `deploy_project_01.sh` и укажем инструкции необходимые для запуска **Terraform**:

```
#!/bin/bash

cd /home/$USER/Projects
source cloudinit.conf
cd /home/$USER/Projects/Project_01/terraform
terraform init
terraform apply -auto-approve

sleep 60
echo "done"
```

- Выдать права на исполнение для файла `deploy_project_01.sh`:

```
chmod +x deploy_project_01.sh
```

- Запустить скрипт на исполнение:

```
./deploy_project_01.sh
```

- Запустить скрипт на исполнение:

```
./deploy_project_01.sh
```

- Результат:

```
Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
done
[altlinux@cloud-adm Project_01]$
```

- Проверить автоматически созданные инстансы в облаке:

Виртуальные машины

☐	Имя ↑	Статус ↓	IP-адрес	вЦП ↓	ОЗУ ↓	Хранилище
☐	Cloud-ADM	▶ Запущена	192.168.1.10	2	4 ГиБ	30 ГиБ
☐	game01	▶ Запущена	192.168.1.101	1	1 ГиБ	10 ГиБ
☐	game02	▶ Запущена	192.168.1.102	1	1 ГиБ	10 ГиБ
☐	game03	▶ Запущена	192.168.1.103	1	1 ГиБ	10 ГиБ
☐	haproxy01	▶ Запущена	192.168.1.100	1	1 ГиБ	10 ГиБ

- Проверить Плавающий IP-адрес для инстанса **haproxy01**:

Плавающие IP-адреса

☐	IP-адрес ↓	Статус	Сеть	Назначен	IP-адрес ВМ
☐	172.20.102.159	✔ Запущен	public	haproxy01	192.168.1.100
☐	172.20.102.183	✔ Запущен	public	Cloud-ADM	192.168.1.10

- Поскольку IP-адресация задаётся статическая, то для удобства работы редактируем конфигурационный файл **/etc/hosts**:

```
127.0.0.1    localhost.localdomain localhost
::1         localhost6.localdomain6 localhost6
192.168.1.100 haproxy01.dev.au.team haproxy01
192.168.1.101 game01.dev.au.team game01
192.168.1.102 game02.dev.au.team game02
192.168.1.103 game03.dev.au.team game03
```

- Создадим директорию **ansible** в ранее созданной директории для **Project_01**:

```
mkdir /home/altlinux/Projects/Project_01/ansible
```

- Перейдём в директорию **/home/altlinux/Projects/Project_01/ansible**:

```
cd /home/altlinux/Projects/Project_01/ansible
```

- Создадим файл конфигурации **ansible.cfg** поместим в него следующее содержимое:

```
[defaults]
inventory = ./inventory.yml
host_key_checking = False
callback_enabled = profile_tasks
callback_whitelist = profile_tasks
```

- Создадим файл **inventory.yml** и опишем файл инвентаря для **Ansible**:
 - описав 2 группы хостов с именами **proxys** и **games**:
 - группа **proxys** содержит в себе инстанс **haproxy01**;
 - группа **games** содержит в себе инстансы **game01**, **game02** и **game03**;

```
all:
  children:
    proxys:
      hosts:
        haproxy01:

    games:
      hosts:
        game01:
        game02:
        game03:
```

- Создадим директорию **group_vars** где будем создавать файлы с переменными для групп хостов:

```
mkdir group_vars
```

- Создадим файл **'all.yml'** в директории **group_vars**:

```
---
ansible_python_interpreter: /usr/bin/python3
ansible_ssh_user: altlinux
ansible_ssh_private_key_file: ~/.ssh/id_rsa
```

- Проверяем возможность **ansible** подключиться к инстансам описанным в инвентарном файле:

```
ansible -m ping all
```

- Результат:

```
[altlinux@cloud-adm ansible]$ ansible -m ping all
haproxy01 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
game02 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
game01 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
game03 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[altlinux@cloud-adm ansible]$
```

- Скачиваем любым способом файлы
https://disk.yandex.ru/d/uhpN6U6UYRK_Nw (у приложения есть README от программиста-разработчика)
- Распаковываем архив в директорию
/home/altlinux/Projects/Project_01/2048-game/:

```
unzip ~/Project_01.zip -d /home/altlinux/Projects/Project_01/2048-game/
```

- В директории `/home/altlinux/Projects/Project_01/ansible` создаём файл **Dockerfile**:
 - образ приложения нужно сделать легковесным через мультитейджинг

```
FROM node:16-alpine AS builder

WORKDIR /2048-game

COPY package*.json ./
RUN npm install --include=dev

COPY . .
RUN npm run build

EXPOSE 8080

FROM nginx:stable-alpine3.19
COPY --from=builder /2048-game/dist /usr/share/nginx/html
```

- В директории `/home/altlinux/Projects/Project_01/ansible` создаём файл **games_playbook.yml**:
 - таким образом, на группе хостов **games** (на инстансах: **game01**, **game02** и **game03**) будет настроено:
 1. Установлены пакеты: **docker-engine** и **docker-buildx**;
 2. Запущена и добавлена в автозагрузку служба **docker**;
 3. Скопированы все файлы с **Cloud-ADM** на удалённые хосты, необходимые для приложения;
 4. Скопирован с **Cloud-ADM** на удалённые хосты файл **Dockerfile**;
 5. На удалённых хостах выполнена локальная сборка образа для приложения;
 6. На удалённых хостах выполнен запуск контейнеров с веб-приложением.

```

---
- hosts: games
  become: true

  tasks:
    - name: Install docker
      community.general.apt_rpm:
        name:
          - docker-engine
          - docker-buildx
        state: present
        update_cache: true

    - name: Started and enabled docker
      ansible.builtin.systemd:
        name: docker
        state: started
        enabled: true

    - name: Copying the project files
      ansible.builtin.copy:
        src: ../2048-game/
        dest: "/home/{{ ansible_ssh_user }}/2048-game/"

    - name: Copying the Dockerfile
      ansible.builtin.copy:
        src: ./Dockerfile
        dest: "/home/{{ ansible_ssh_user }}/2048-game/"

    - name: Build docker image
      community.docker.docker_image_build:
        name: "2048-game"
        tag: latest
        path: "/home/{{ ansible_ssh_user }}/2048-game/"
        dockerfile: Dockerfile

    - name: Start docker container
      community.docker.docker_container:
        name: "2048-game"
        image: "2048-game"
        ports: "80:80"
        state: started
        restart: true

```

- На текущем этапе можно выполнить тестовый запуск созданного playbook-a **games_playbook.yml**:

```
ansible-playbook games_playbook.yml
```

- Результат:

```

PLAY [games] *****

TASK [Gathering Facts] *****
ok: [game01]
ok: [game02]
ok: [game03]

TASK [Install docker] *****
changed: [game03]
changed: [game02]
changed: [game01]

TASK [Started and enabled docker] *****
changed: [game03]
changed: [game02]
changed: [game01]

TASK [Copying the project files] *****
changed: [game03]
changed: [game02]
changed: [game01]

TASK [Copying the Dockerfile] *****
changed: [game01]
changed: [game03]
changed: [game02]

TASK [Build docker image] *****
changed: [game03]
changed: [game02]
changed: [game01]

TASK [Start docker container] *****
changed: [game03]
changed: [game01]
changed: [game02]

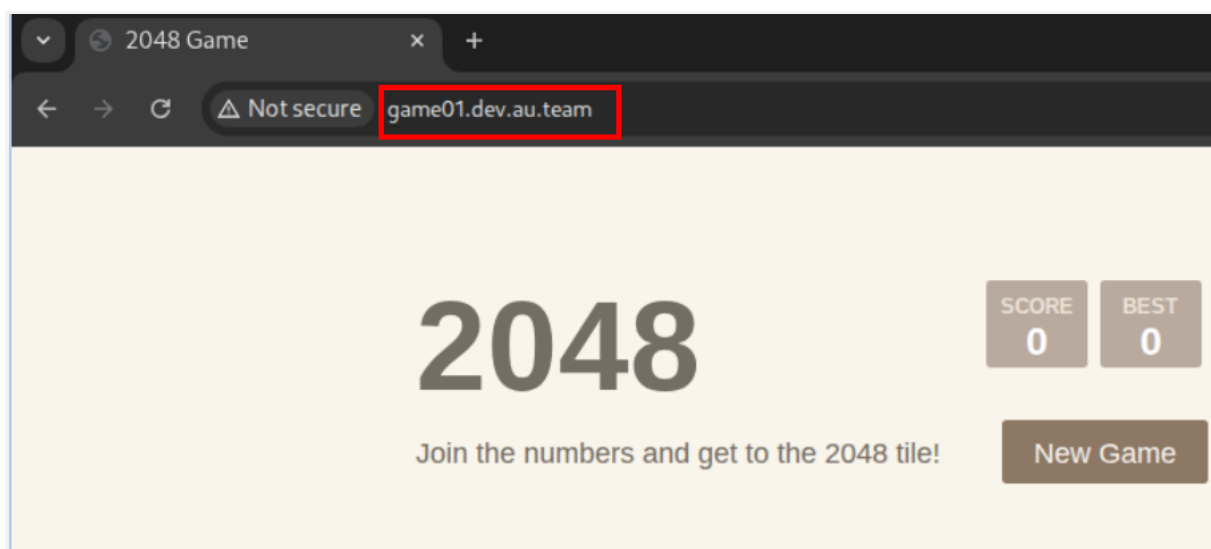
PLAY RECAP *****
game01      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
game02      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
game03      : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

[altlinux@cloud-adm ansible]$

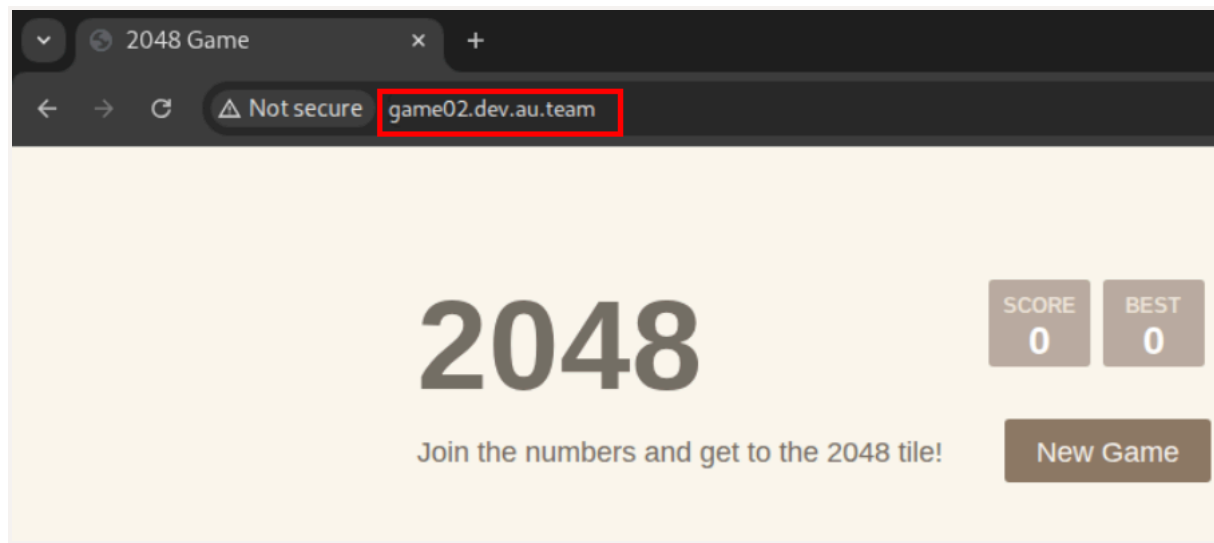
```

- Таким образом при обращении в браузере с Cloud-ADM по описанным ниже url должно открываться веб-приложение:

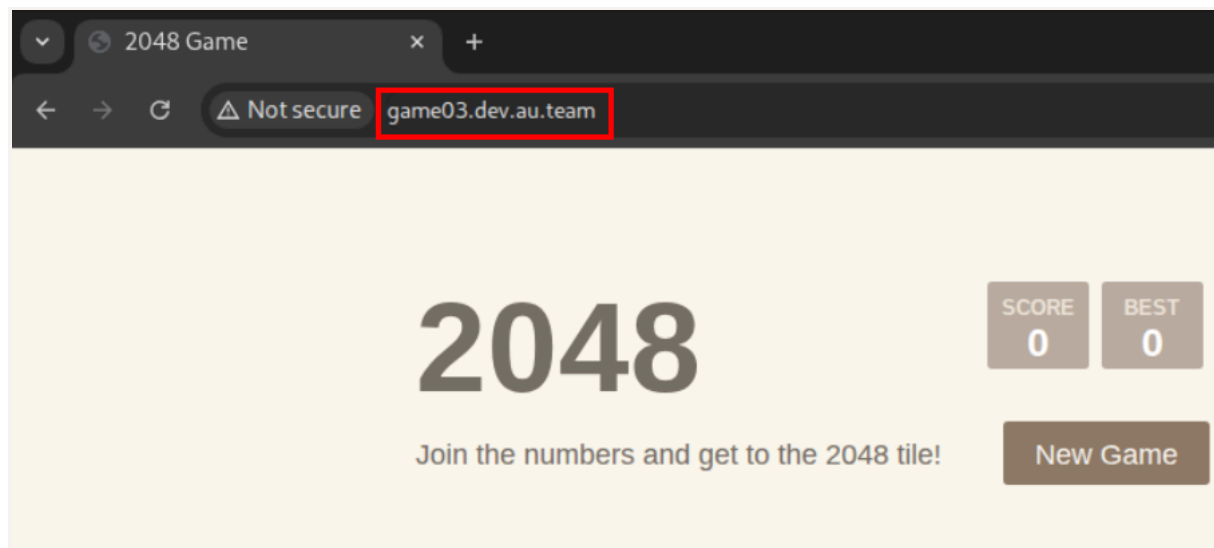
game01.dev.au.team:



game02.dev.au.team:



game03.dev.au.team:



- В директории `/home/altlinux/Projects/Project_01/ansible` создаём директорию `files`:

```
mkdir files
```

- В директории `/home/altlinux/Projects/Project_01/ansible/files` создаём конфигурационный файл `haproxy.cfg`
 - в данном файле стандартное содержимое `haproxy.cfg` за исключением блока `frontend` и `backend` для реализации необходимого функционала
 - для удобства можно найти конфигурационный файл в сети Интернет;
 - или же выполнить установку пакета `haproxy` и забрать его из директории `/etc/haproxy/haproxy.cfg` (т.к. инстанс `haproxy01` в любом случае будет удалён).

```
global
    log /dev/log daemon
    chroot      /var/lib/haproxy
    pidfile     /run/haproxy.pid
    maxconn     4000
    user        _haproxy
    group       _haproxy
    daemon
    stats socket /var/lib/haproxy/stats

defaults
    mode                http
    log                 global
    option               httplog
    option               dontlognull
    option http-server-close
    option forwardfor    except 127.0.0.0/8
    option               redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn              3000

frontend http_front
    bind *:80
    bind *:443 ssl crt /var/lib/ssl/game.pem
    default_backend http_back

backend http_back
    balance roundrobin
    server game01 192.168.1.101:80 check
    server game02 192.168.1.102:80 check
    server game03 192.168.1.103:80 check

listen stats
    bind :80
    bind *:443 ssl crt /var/lib/ssl/game.pem
    stats enable
    stats uri /
    stats refresh 5s
    stats realm Haproxy\ Stats
```

- Средствами утилиты **openssl** разворачиваем свой Удостоверяющий Центр сертификатов:

```
openssl req -x509 -sha256 -days 3653 -newkey rsa:2048 -keyout ca.key -out ca.crt
```

- Результат:

```
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:au.team
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:ca.au.team
Email Address []:
[altlinux@cloud-adm ansible]$
```

- Добавляем корневой сертификат в хранилище на **Cloud-ADM**:

```
sudo cp ca.crt /etc/pki/ca-trust/source/anchors/ && sudo update-ca-trust
```

- Перемещаем файлы сертификатов в директорию **files**:

```
mv ca.* files/
```

- Генерируем ключи, запросы и сертификаты для веб:
 - Ключ:

```
openssl genrsa -out files/game.key 2048
```

- Запрос:

```
openssl req -key files/game.key -new -out files/game.csr
```

- Результат:

```
[altlinux@cloud-adm ansible]$ openssl req -key files/game.key -new -out files/game.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:au.team
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:game.au.team
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[altlinux@cloud-adm ansible]$
```

- Файл с расширениями для сертификата:

```
cat <<EOF > files/game.ext
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
subjectAltName=@alt_names
[alt_names]
DNS.1=game.au.team
IP.1=192.168.1.100
EOF
```

- Выпускаем сертификат:

```
openssl x509 -req -CA files/ca.crt -CAkey files/ca.key -in
files/game.csr -out files/game.crt -days 365 -CAcreateserial
-extfile files/game.ext
```

- Создаём цепочку для haproxy:

```
cat files/game.key files/game.crt > files/game.pem
```

- В директории `/home/altlinux/Projects/Project_01/ansible` создаём файл `proxys_playbook.yml`:
 - таким образом, на группе хостов **games** (на инстансах: **game01**, **game02** и **game03**) будет настроено:
 1. Установлен пакет **haproxy**;
 2. Скопирован с **Cloud-ADM** на удалённый хост файл **haproxy.cfg**;
 3. Скопирован с **CloudpADM** на удалённый хост файл **game.pem**;
 4. Включена и добавлена в автозагрузку служба **haproxy**.

```

---
- hosts: proxys
  become: true

  tasks:
    - name: Install HAProxy
      community.general.apt_rpm:
        name: haproxy
        state: present
        update_cache: true

    - name: Copy file 'haproxy.cfg'
      ansible.builtin.copy:
        src: files/haproxy.cfg
        dest: /etc/haproxy/haproxy.cfg
      notify:
        - Restarted HAProxy

    - name: Copy certificate for HAProxy
      ansible.builtin.copy:
        src: files/game.pem
        dest: /var/lib/ssl/game.pem
      notify:
        - Restarted HAProxy

    - name: Started and enabled HAProxy
      ansible.builtin.systemd:
        name: haproxy
        state: restarted
        enabled: true

  handlers:
    - name: Restarted HAProxy
      ansible.builtin.systemd:
        name: haproxy
        state: restarted

```

- На текущем этапе можно выполнить тестовый запуск созданного playbook-а **proxys_playbook.yml**:

```
ansible-playbook proxys_playbook.yml
```

- Результат:

```
[altlinux@cloud-adm ansible]$ ansible-playbook proxys_playbook.yml

PLAY [proxys] *****

TASK [Gathering Facts] *****
ok: [haproxy01]

TASK [Install HAProxy] *****
changed: [haproxy01]

TASK [Copy file 'haproxy.cfg'] *****
changed: [haproxy01]

TASK [Copy certificate for HAProxy] *****
changed: [haproxy01]

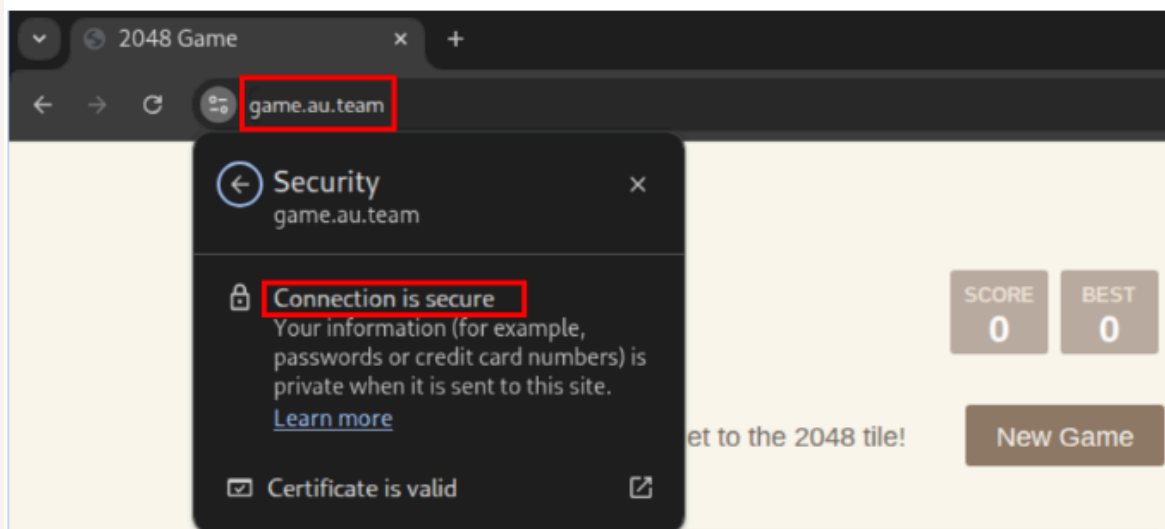
TASK [Started and enabled HAProxy] *****
changed: [haproxy01]

RUNNING HANDLER [Restarted HAProxy] *****
changed: [haproxy01]

PLAY RECAP *****
haproxy01          : ok=6   changed=5   unreachable=0   failed=0   skipped=0   rescued=0   ignored=0

[altlinux@cloud-adm ansible]$
```

- Таким образом при обращении в браузере с **Cloud-ADM** по описанным ниже url должно открываться веб-приложение:
 - <https://game.au.team> (в файле /etc/hosts должна быть добавлена запись):



o haproxy01.dev.au.team /haproxy?stats

Statistics Report for HAP

Not secure haproxy01.dev.au.team/haproxy?stat

HAProxy version 3.0.12-alt1, released 2025/11/17

Statistics Report for pid 1654

> General process information

pid = 1654 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h04m50s; warnings = 0
system limits: memmax = unlimited; ulimit-n = 8054
maxsock = 8054; maxconn = 4000; reached = 0; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
Running tasks: 0/19 (0 niced); idle = 100 %

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

backup UP

backup UP, going down

backup DOWN, going up

not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

Scope :

[Hide DOWN servers](#)

[Disable refresh](#)

[Refresh now](#)

[CSV export](#)

[JSON export \(schema\)](#)

External resources:

[Primary site](#)

[Updates \(v3.0\)](#)

[Online manual](#)

http_front																															
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend	0	0	1	0	1	0	1	3	000	2			3	785	74	161	0	0	0			OPEN									

http_back																														
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
game01	0	0	-	0	1	0	1	-	3	3	14s	1	519	1	805		0	0	0	0	0	4m50s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-
game02	0	0	-	0	1	0	1	-	3	3	14s	1	282	70	986		0	0	0	0	0	4m50s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-
game03	0	0	-	0	1	0	1	-	2	2	15s	994	1	370		0	0	0	0	0	4m50s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-	
Backend	0	0		0	3		0	1	300	2	8	14s	3	785	74	161	0	0	0	0	0	4m50s UP		3/3	3	0		0	0s	

stats																															
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend	1	1	-	1	1	0	1	1	3	000	1		0	0	0	0	0					OPEN									
Backend	0	0		0	0		0	0	300	0	0	0s	0	0	0	0	0	0	0	0	0	4m50s UP		0/0	0	0		0			

o Плавающий IP-адрес:

Киберпротект

Плавающие IP-адреса

Вычисления

Плавающие IP-адреса

Виртуальные машины

Образы

Топы

Сети

Маршрутизаторы

Плавающие IP-адреса

Группы безопасности

IP-адрес	Статус	Сеть	Назначен	IP-адрес ВМ
172.20.102.127	Запущен	public	haproxy01	192.168.1.100
172.20.102.183	Запущен	public	Cloud-ADM	192.168.1.10

2048 Game

2048 Game

2048

Join the numbers and get to the 2048 tile!

SCORE 0 BEST 0

New Game

- Перейдём в директорию `/home/altlinux/Projects/Project_01`:

```
cd /home/altlinux/Projects/Project_01
```

- Создадим файл `configure_project_01.sh` и укажем инструкции необходимые для запуска **Ansible**:

```
#!/bin/bash

export PATH=/home/altlinux/.local/bin:$PATH
cd /home/$USER/Projects/Project_01/ansible
sleep 10
ansible-playbook games_playbook.yml
sleep 5
ansible-playbook proxys_playbook.yml
```

- Выдать права на исполнение для файла `configure_project_01.sh`:

```
chmod +x configure_project_01.sh
```

- Запустить скрипт на исполнение:

```
./configure_project_01.sh
```

- Создадим файл `destroy_project_01.sh` и укажем инструкции необходимые для запуска **Ansible**:

```
#!/bin/bash

cd /home/$USER/Projects
source cloudinit.conf
cd /home/$USER/Projects/Project_01/terraform
terraform destroy -auto-approve
rm -f ~/.ssh/known_hosts

echo "done"
```

- Выдать права на исполнение для файла `destroy_project_01.sh`:

```
chmod +x destroy_project_01.sh
```

- Запустить скрипт на исполнение:

```
./destroy_project_01.sh
```

- Результат:

```
Destroy complete! Resources: 10 destroyed.
done
[altlinux@cloud-adm Project_01]$
```

- Таким образом, структура для **Project01** получилась следующая:

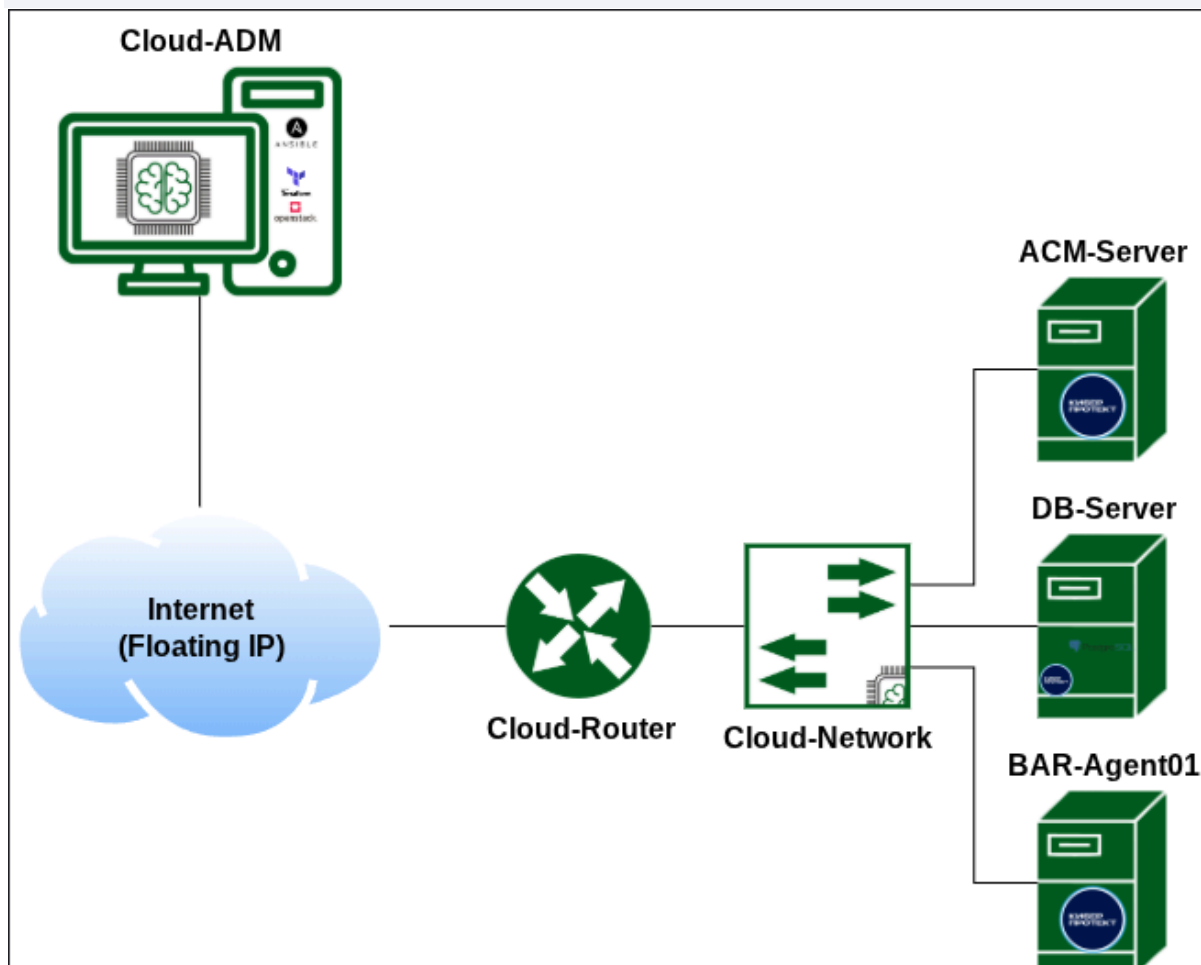
```
[altlinux@cloud-adm Project_01]$ tree
```

```
.
├── 2048-game
│   ├── assets
│   │   └── images
│   │       └── game-2048-welcome-page.png
│   ├── package.json
│   ├── package-lock.json
│   ├── README.md
│   ├── src
│   │   └── index.js
│   └── webpack.config.js
├── ansible
│   ├── ansible.cfg
│   ├── Dockerfile
│   ├── files
│   │   ├── ca.crt
│   │   ├── ca.key
│   │   ├── ca.srl
│   │   ├── game.crt
│   │   ├── game.csr
│   │   ├── game.ext
│   │   ├── game.key
│   │   ├── game.pem
│   │   └── haproxy.cfg
│   ├── games_playbook.yml
│   ├── group_vars
│   │   └── all.yml
│   ├── inventory.yml
│   └── proxys_playbook.yml
├── configure_project_01.sh
├── deploy_project_01.sh
├── destroy_project_01.sh
└── terraform
    ├── cloud-init.yml
    ├── floatingip.tf
    ├── network.tf
    ├── provider.tf
    ├── terraform.tfstate
    ├── terraform.tfstate.backup
    ├── vm-game.tf
    └── vm-haproxy01.tf
```

```
9 directories, 32 files
```

```
[altlinux@cloud-adm Project_01]$
```

Project_02



Задание:

а. Любым способом разверните недостающие виртуальные машины в облаке в соответствие с топологией (см. Топология Project_02).

1. Характеристики виртуальных машин:

1. ACM-Server: 1vCPU, 2 ГБ ОЗУ, 20 ГБ размер диска.
2. DB-Server: 1vCPU, 1 ГБ ОЗУ, 20 ГБ размер диска.
3. BAR-Agent01: 1vCPU, 1 ГБ ОЗУ, 10 ГБ размер

2. Образ операционной системы: alt-p11-cloud-x86_64.qcow2.

б. На развёрнутых виртуальных машинах реализуйте следующий функционал:

1. Установку сервера управления Кибер Бэкап (версии не ниже 18) на виртуальной машине ACM-Server.

1. Сервер управления должен быть доступен с Cloud-ADM по имени `cb.au.team` на порту `9877`.
 2. Доступ в веб-интерфейс сервера управления должен быть из-под пользователя `root` с паролем `toor`.
 3. Для установки сервера управления в качестве СУБД необходимо использовать PostgreSQL.
2. Установку агента для Linux на виртуальной машине BAR-Agent01 с последующей регистрацией на сервере управления.
 3. Установку агента для PostgreSQL на виртуальной машине DB-Server с последующей регистрацией на сервере управления.

Вариант реализации:

Cloud-ADM:

Развёртывание недостающих виртуальных машин

- Допустимы любые способы, в том числе и ручное создание VM
- Модернизирует ранее написанные файлы для **Terraform** чтобы ускорить процесс создания VM
- Создадим директорию для **Project02** и перейдём в неё:

```
cd ~/Projects/
```

```
mkdir -p Project_02/terraform
```

```
cd Project_02/terraform
```

- Скопируем файл **provider.tf** из **Project01**:

```
cp ~/Projects/Project_01/terraform/provider.tf ./
```

- Менять в данном файле ничего не следует, инициализируем текущий каталог для работы с **Terraform**:

```
terraform init
```

- Результат:

```
Terraform has been successfully initialized!
```

- Скопируем файл **network.tf** из **Project01**:

```
cp ~/Projects/Project_01/terraform/network.tf ./
```

- модернизируем файл **network.tf** в текущей директории для **Project02**:
 - избавляемся от цикла через счётчик **count**
 - копируем данный блок, создавая порты с фиксированными IP-адресами для каждой VM

```
resource "openstack_networking_port_v2" "port_vm_acm-server" {
  name          = "port_acm-server"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.104"
  }
}

resource "openstack_networking_port_v2" "port_vm_db-server" {
  name          = "port_db-server"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.105"
  }
}

resource "openstack_networking_port_v2" "port_vm_bar-agent01" {
  name          = "port_bar-agent01"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.106"
  }
}
```

- Скопируем файл **network.tf** из **Project01**:

```
cp ~/Projects/Project_01/terraform/vm-game.tf ./vm.tf
```

- модернизируем файл **vm.tf** в текущей директории для **Project02**:
 - избавляемся от цикла через счётчик **count**
 - копируем данный блок, создавая VM с соответствующими параметрами (по требованию задания)

```

resource "openstack_compute_instance_v2" "acm-server" {
  name      = "ACM-Server"
  flavor_id = "101"
  user_data = file("cloud-init.yml")

  block_device {
    uuid              = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type       = "image"
    volume_size       = "20"
    boot_index        = 0
    destination_type  = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_acm-server.id
  }
}

resource "openstack_compute_instance_v2" "db-server" {
  name      = "DB-Server"
  flavor_id = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data = file("cloud-init.yml")

  block_device {
    uuid              = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type       = "image"
    volume_size       = "20"
    boot_index        = 0
    destination_type  = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_db-server.id
  }
}

resource "openstack_compute_instance_v2" "bar-agent01" {
  name      = "BAR-Agent01"
  flavor_id = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data = file("cloud-init.yml")

  block_device {
    uuid              = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type       = "image"
    volume_size       = "10"
    boot_index        = 0
    destination_type  = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_bar-agent01.id
  }
}

```

- Скопируем файл **network.tf** из **Project01**:

```
cp ~/Projects/Project_01/terraform/cloud-init.yml ./
```

- Запускаем автоматическое развёртывания VM для **Project02** через **Terraform**:

```
terraform apply
```

- Результат:

```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
```

- Результат в веб-интерфейсе облака:

Виртуальные машины

Фильтр		Поиск				
☐	Имя ↓	Статус ↓	IP-адрес	вЦП ↓	ОЗУ ↓	Хранилище ↓
☐	 Cloud-ADM	 Запущена	192.168.1.10	2	4 ГиБ	30 ГиБ
☐	 ACM-Server	 Запущена	192.168.1.104	1	2 ГиБ	20 ГиБ
☐	 DB-Server	 Запущена	192.168.1.105	1	1 ГиБ	20 ГиБ
☐	 BAR-Agent01	 Запущена	192.168.1.106	1	1 ГиБ	10 ГиБ

- Для удобства приводим конфигурационный файл **/etc/hosts** к следующему виду:

```
127.0.0.1    localhost.localdomain localhost
::1         localhost6.localdomain6 localhost6

192.168.1.100 haproxy01.dev.au.team haproxy01
192.168.1.100 game.au.team
192.168.1.101 game01.dev.au.team game01
192.168.1.102 game02.dev.au.team game02
192.168.1.103 game03.dev.au.team game03
192.168.1.104 acm-server.au.team acm-server
192.168.1.105 db-server.au.team db-server
192.168.1.106 bar-agent01.au.team bar-agent01
[attilinx@cloud-adm terraform]$
```

- Проверяем доступ до созданных VM:

```
[altlinux@cloud-adm terraform]$ ssh acm-server
The authenticity of host 'acm-server (192.168.1.104)' can't be established.
ED25519 key fingerprint is SHA256:Vdp2R7EUHKWtRNYoJCqKrWlQeFNaUUx/Fx9X/flElGg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'acm-server' (ED25519) to the list of known hosts.
[altlinux@acm-server ~]$ exit
выход
Connection to acm-server closed.
[altlinux@cloud-adm terraform]$ ssh db-server
The authenticity of host 'db-server (192.168.1.105)' can't be established.
ED25519 key fingerprint is SHA256:HQ4d67+FVFruEJwGdrddYp8QFMdylWwCnvp/qnxtJKw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'db-server' (ED25519) to the list of known hosts.
[altlinux@db-server ~]$ exit
выход
Connection to db-server closed.
[altlinux@cloud-adm terraform]$ ssh bar-agent01
The authenticity of host 'bar-agent01 (192.168.1.106)' can't be established.
ED25519 key fingerprint is SHA256:le0NMNMAaoTnBnWPC2Uw5QZmNYbuJ4MFQIZtXtL2T6s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'bar-agent01' (ED25519) to the list of known hosts.
[altlinux@bar-agent01 ~]$ exit
выход
Connection to bar-agent01 closed.
[altlinux@cloud-adm terraform]$
```

- Передаём файл для установки Кибер Бекап на ВМ (как файл попал на Cloud-ADM не рассматривается):

```
scp CyberBackup_18_64-bit.x86_64 acm-server:~/
```

```
scp CyberBackup_18_64-bit.x86_64 db-server:~/
```

```
scp CyberBackup_18_64-bit.x86_64 bar-agent01:~/
```

DB-Server:

- Установим пакет `postgresql17-server`:

```
apt-get update && apt-get install -y postgresql17-server
```

- Создаём системные базы данных:

```
/etc/init.d/postgresql initdb
```

- Включаем и добавляем в автозагрузку PostgreSQL:

```
systemctl enable --now postgresql
```

- По умолчанию доступ из сети к PostgreSQL отключён:

```
[root@db-server ~]# ss -tulpn | grep "5432"
tcp LISTEN 0      200      127.0.0.1:5432      0.0.0.0:*      users:((("postgres",pid=1624,fd=7))
[root@db-server ~]#
```

- Разрешаем доступ к PostgreSQL из сети отредактировав конфигурационный файл `/var/lib/pgsql/data/postgresql.conf`:
 - в конфигурационном файле находим строку `'listen_addresses = 'localhost''` и приводим её к следующему виду:

```
# - Connection Settings -
listen_addresses = '*'
```

- Настраиваем парольную аутентификацию для удалённого доступа отредактировав конфигурационный файл `/var/lib/pgsql/data/pg_hba.conf`:
 - добавляем следующую запись:

```
# TYPE      DATABASE    USER        ADDRESS          METHOD
# "local" is for Unix domain socket connections only
local      all         all                                     scram-sha-256
# IPv4 local connections:
host       all         all          0.0.0.0/0        md5
host       all         all          127.0.0.1/32     scram-sha-256
# IPv6 local connections:
host       all         all          ::1/128          scram-sha-256
# Allow replication connections from localhost, by a user with the
```

- Перезапускаем PostgreSQL:

```
systemctl restart postgresql
```

- Проверяем доступ по сети:

```
[root@db-server ~]# ss -tln | grep "5432"
LISTEN 0      200      0.0.0.0:5432      0.0.0.0:*      users:((("postgres",pid=1968,fd=7))
LISTEN 0      200      [::]:5432        [::]:*        users:((("postgres",pid=1968,fd=8))
[root@db-server ~]#
```

- Создадим пользователя **cyberbackup** с паролем **cyberbackup** с ролью **superadmin**:

```
createuser -U postgres --superuser --encrypted --pwprompt cyberbackup
```

ACM-Server:

- Обновляем систему до актуального состояния и перезагружаем устройство:

```
apt-get update && apt-get dist-upgrade -y && update-kernel -y && apt-get clean && reboot
```

- Должны быть установлены следующие пакеты:
 - где **<x.x>** – версия ядра (**uname -r**)

```
apt-get install kernel-source-<x.x>
```

- например:

```
[root@acm-server ~]# uname -r
6.12.57-6.12-alt1
[root@acm-server ~]# apt-cache search kernel-source | grep "6.12"
kernel-source-6.12 - Linux kernel 6.12 sources
[root@acm-server ~]#
```

```
apt-get install -y kernel-source-6.12
```

```
apt-get install -y kernel-headers-modules-6.12 gcc make kmod-sign
```

- Задаём разрешение на исполнение установочному файлу с дистрибутивом Кибер Бэкап:
 - для этого дистрибутив должен быть заранее скачен и помещён на виртуальную машину

```
cd /home/altlinux/
```

```
chmod +x CyberBackup_18_64-bit.x86_64
```

- Из под суперпользователя запускаем файл установки:

```
./CyberBackup_18_64-bit.x86_64
```

- Результат:

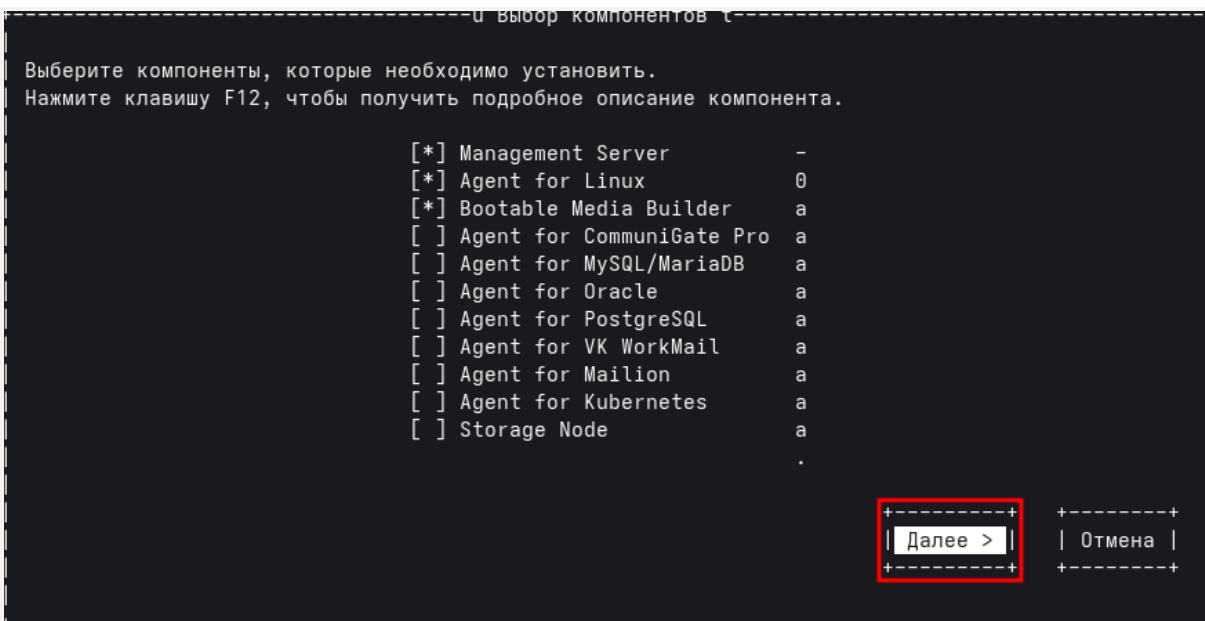
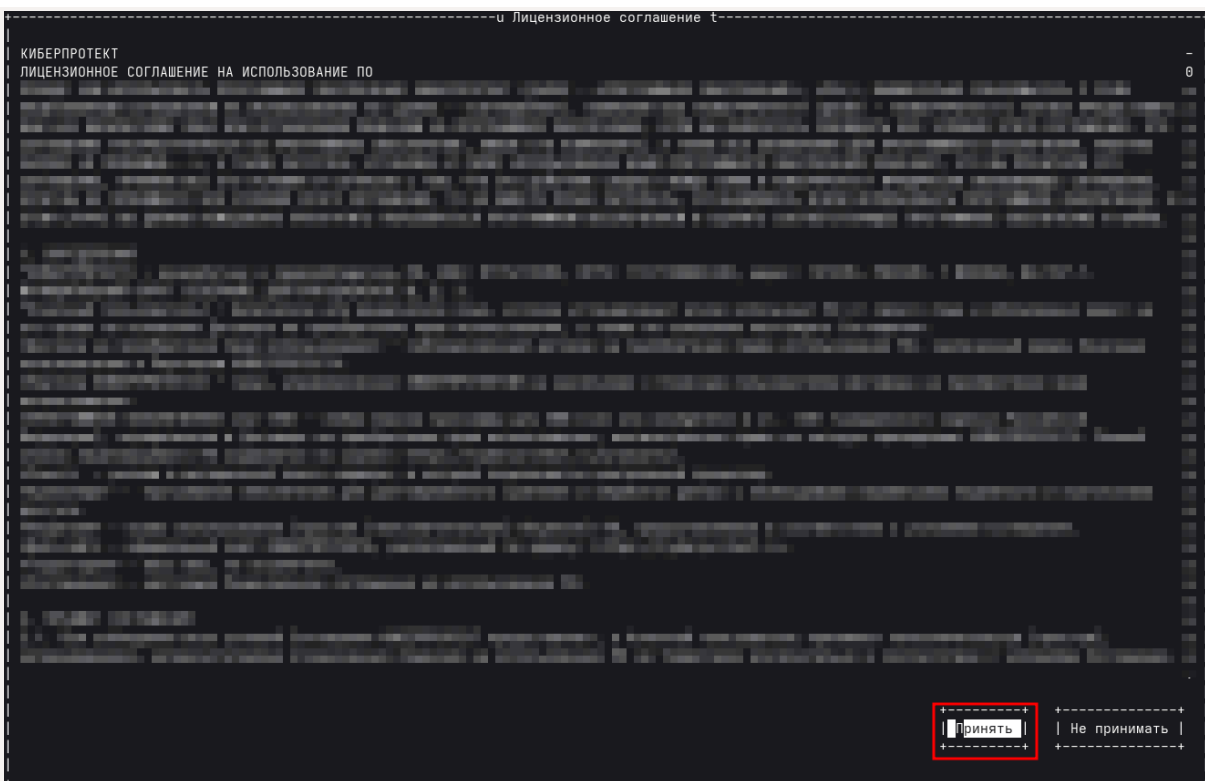
Инициализация...

```
-----и Добро пожаловать t-----
Установка Кибер Бэкап.

Процедура установки описана в руководстве по установке Кибер Бэкап. Прежде чем продолжить, можно ознакомиться с руководством.

Нажмите кнопку "Далее >", чтобы начать обновление системы с помощью пакетов Кибер Бэкап. Чтобы прервать процесс, нажмите кнопку "Отмена".
```

Далее > | Отмена |



+-----и Выбор СУБД t-----+

Выберите СУБД для использования Сервером управления

() Использовать встроенную SQLite

(*) Использовать PostgreSQL/Patroni (Должна быть заранее установлена)

+-----+ +-----+

| Далее > | | Отмена |

+-----+ +-----+

+-----и Сведения о подключении PostgreSQL/Patroni t-----+

Укажите данные для подключения к PostgreSQL/Patroni и аутентификации в ней

Хост: 192.168.1.105

Порт: 5432

Пользователь: cyberbackup

Пароль: *****

+-----+ +-----+

| Далее > | | Отмена |

+-----+ +-----+

+-----и Укажите порт для Cyberprotect Web Server t-----+

9877

+-----+ +-----+

| Далее > | | Отмена |

+-----+ +-----+

+и Укажите TCP-порт для обмена данными между компонентами продукта t+

7780

Далее >

Отмена

```
+-----и Установка пакетов Кибер Бэкап t-----+
```

Имя: MonitoringServer-18.0.38188-1

Размер: 695745764

Описание: Acronis Monitoring Service

Прогресс пакета

61%

61%

Общий прогресс

11%

11%

Запуск служб Киберпротект...

Cloud-ADM:

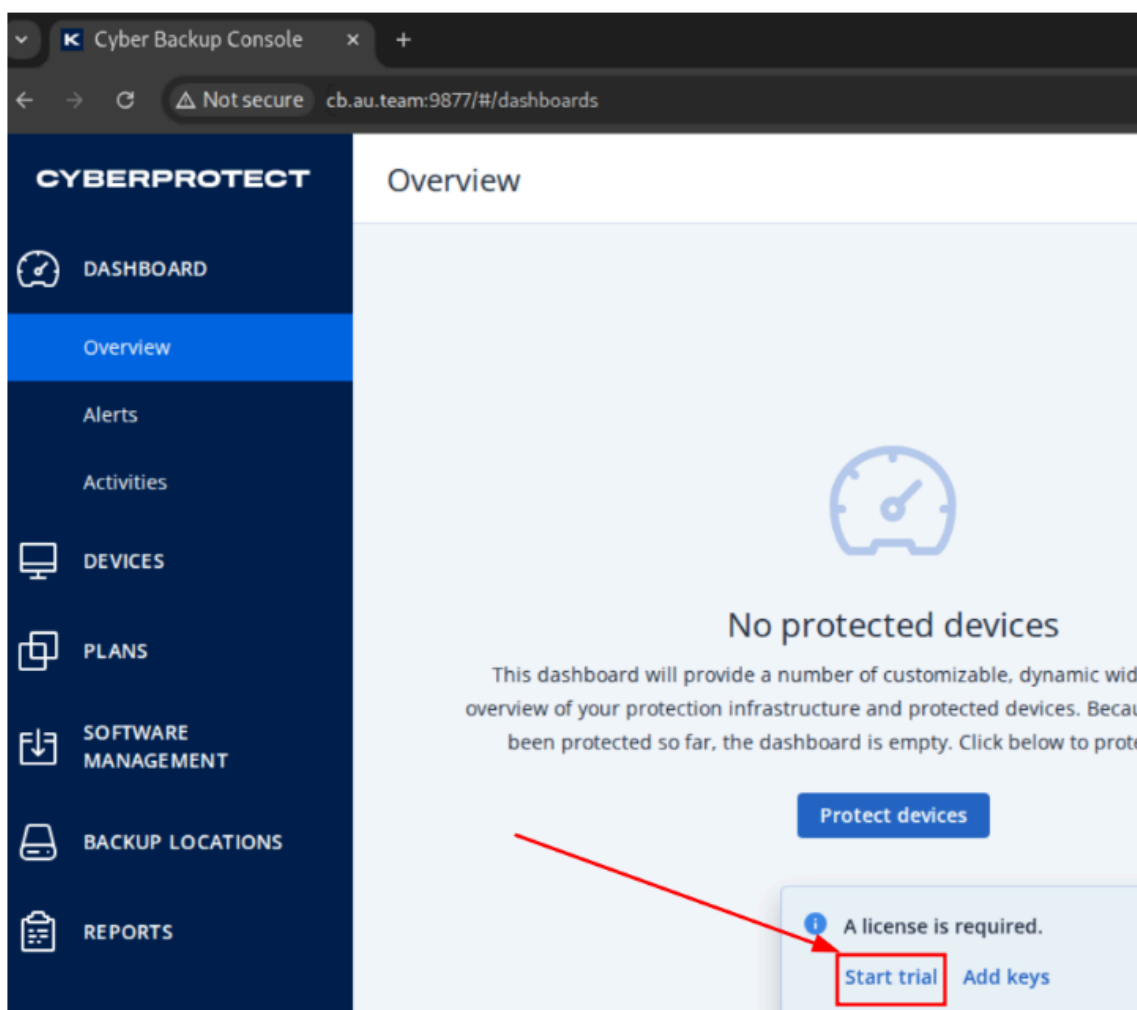
- Добавляем в конфигурационный файл `/etc/hosts` следующую запись:

```
127.0.0.1      localhost.localdomain localhost
::1           localhost6.localdomain6 localhost6
192.168.1.100  haproxy01.dev.au.team  haproxy01
192.168.1.100  game.au.team
192.168.1.101  game01.dev.au.team     game01
192.168.1.102  game02.dev.au.team     game02
192.168.1.103  game03.dev.au.team     game03
192.168.1.104  acm-server.au.team     acm-server
192.168.1.105  db-server.au.team      db-server
192.168.1.106  bar-agent01.au.team    bar-agent01
192.168.1.104  cb.au.team
```

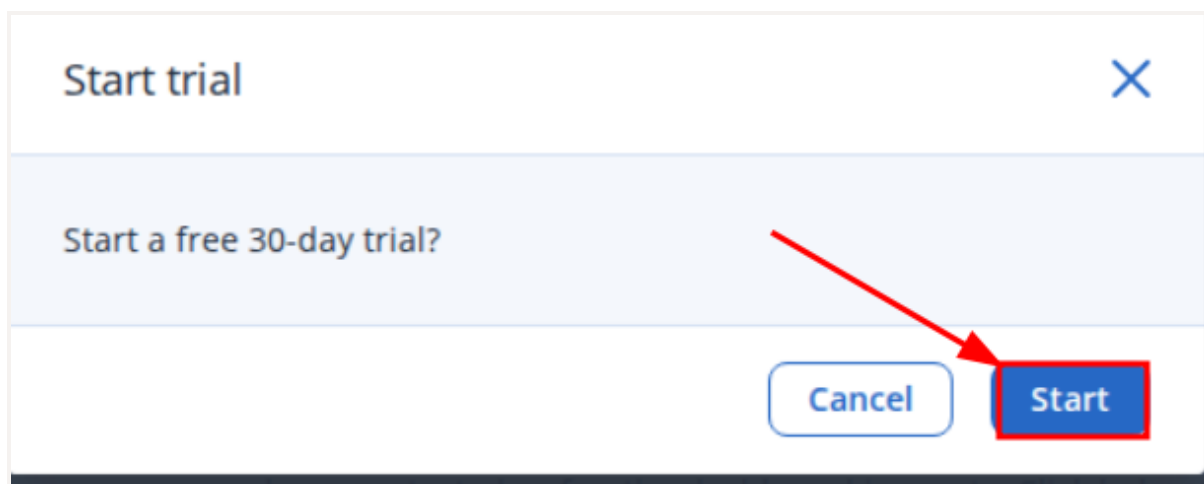
- Открываем веб-браузер и переходим в веб-интерфейс управления <http://cb.au.team:9877>:

The screenshot shows a web browser window with the address bar displaying `cb.au.team:9877/` and the URL `http://cb.au.team:9877/idp/authorize/local?req=su3z6elyaz4aatugmfc5icg5be`. The page has a dark blue background with the text "КИБЕР Бэкап" (Cyber Backup) in white. In the center, there is a white sign-in form with the title "Sign in". The form contains two input fields: "USER NAME" with a hint "What account do I use?" and "PASSWORD". Below the fields is a blue "SIGN IN" button.

- Выполняем вход из-под пользователя **root** с паролем **toor**:



- Активируем стартовую лицензию на 30-дней



DB-Server:

- Обновляем систему до актуального состояния и перезагружаем устройство:

```
apt-get update && apt-get dist-upgrade -y && update-kernel -y && apt-get clean && reboot
```

- Должны быть установлены следующие пакеты:

```
apt-get install -y kernel-source-6.12
```

```
apt-get install -y kernel-headers-modules-6.12 gcc make kmod-sign
```

- Задаём разрешение на исполнение установочному файлу с дистрибутивом Кибер Бэкап:
 - для этого дистрибутив должен быть заранее скачен и помещён на виртуальную машину

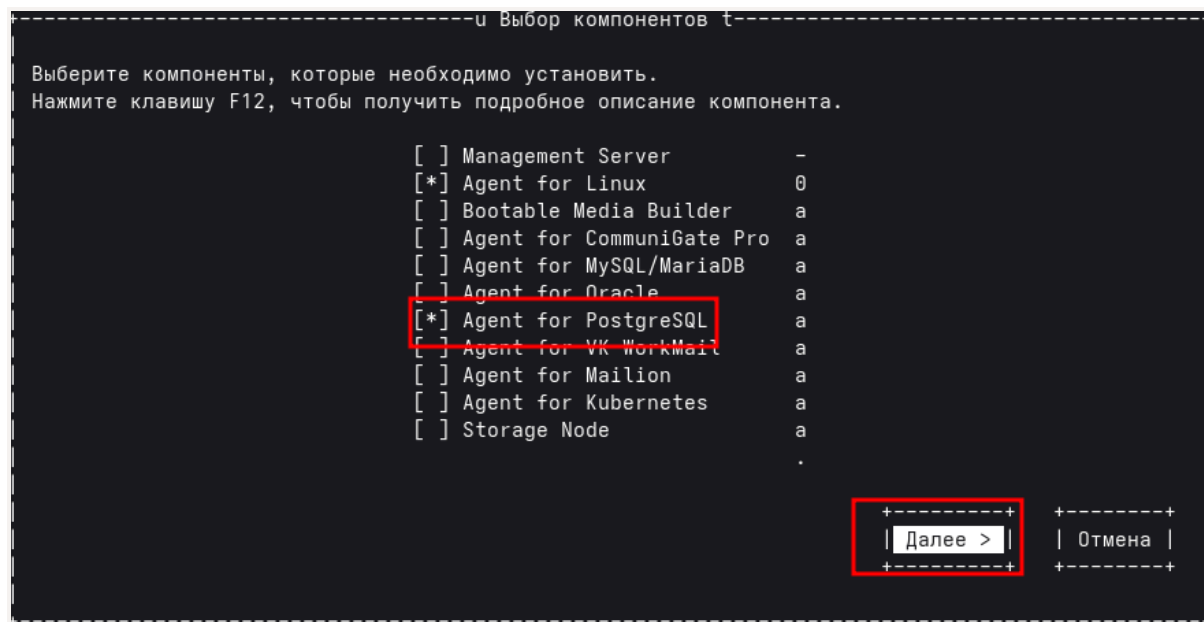
```
cd /home/altlinux/
```

```
chmod +x CyberBackup_18_64-bit.x86_64
```

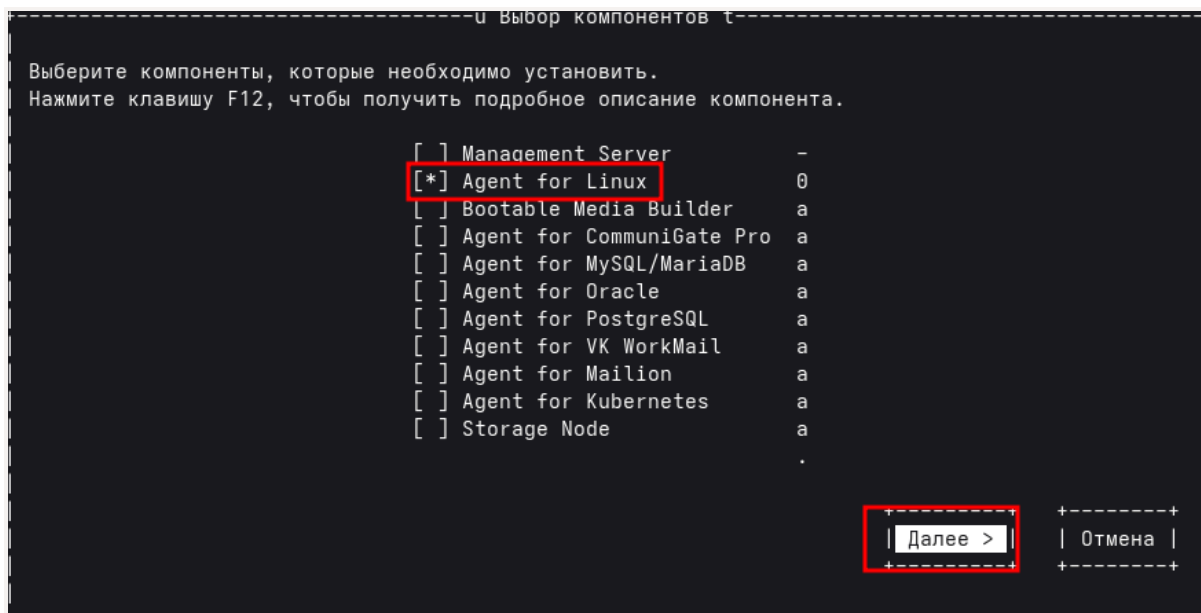
- Из под суперпользователя запускаем файл установки:

```
./CyberBackup_18_64-bit.x86_64
```

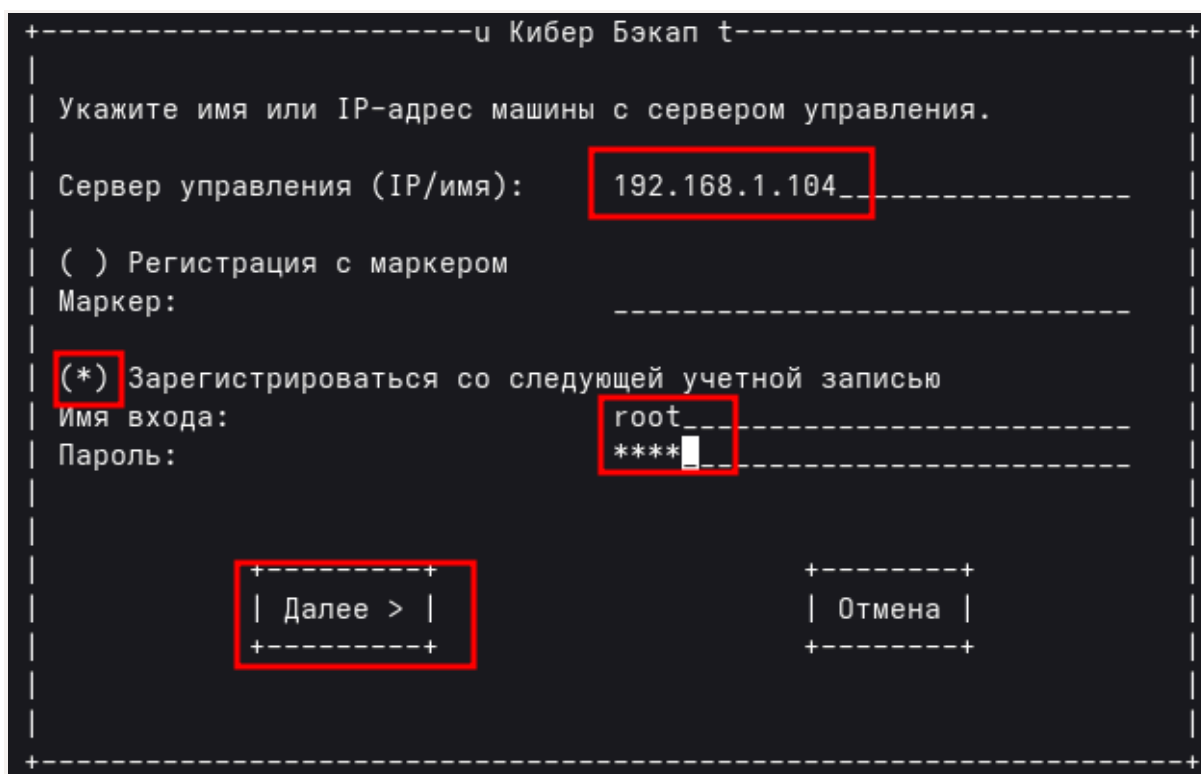
- Результат:
 - на этапе выбора компонентов выбираем:



на этапе подключения к серверу управления:



- на этапе подключения к серверу управления:



Cloud-ADM:

- Проверить наличие агентов в веб-интерфейсе управления:

The screenshot displays the Cyberprotect Cloud-ADM web interface. The browser address bar shows the URL: `cb.au.team:9877/#/backup-console/resources/F656DB8D-3B82-40BA-B06A-BFFC9B0`. The interface has a dark blue sidebar on the left with the following menu items: DASHBOARD, DEVICES (highlighted with a red box), Physical and virtual machines (highlighted with a red box), Machines with agents, Unmanaged machines, PLANS, SOFTWARE MANAGEMENT, BACKUP LOCATIONS, and REPORTS. The main content area is titled "Physical and virtual machines" and lists three machines. The first machine is not shown. The second machine, "bar-agent01", and the third machine, "db-server", are both marked as "Not protected" and have "ENABLE PROTECTION" and "RECOVER" buttons. A red box highlights the details for "bar-agent01" and "db-server".

Machine Name	Status	Last backup	Actions
bar-agent01	Not protected	—	ENABLE PROTECTION, RECOVER
db-server	Not protected	—	ENABLE PROTECTION, RECOVER

- PostgreSQL агент:

db-server



ONLINE



Comment:
No comment yet.



IP addresses:
192.168.1.105



Total size of backups in cloud storage:
Loading...



Unit:
Organization



Agent version: 18.0.38188



Installed agents:
Agent for PostgreSQL
Agent for Linux

- Linux agent:

bar-agent01



ONLINE

 Comment:
No comment yet.

IP addresses:
192.168.1.106

Total size of backups in cloud storage:
Loading...

Unit:
Organization

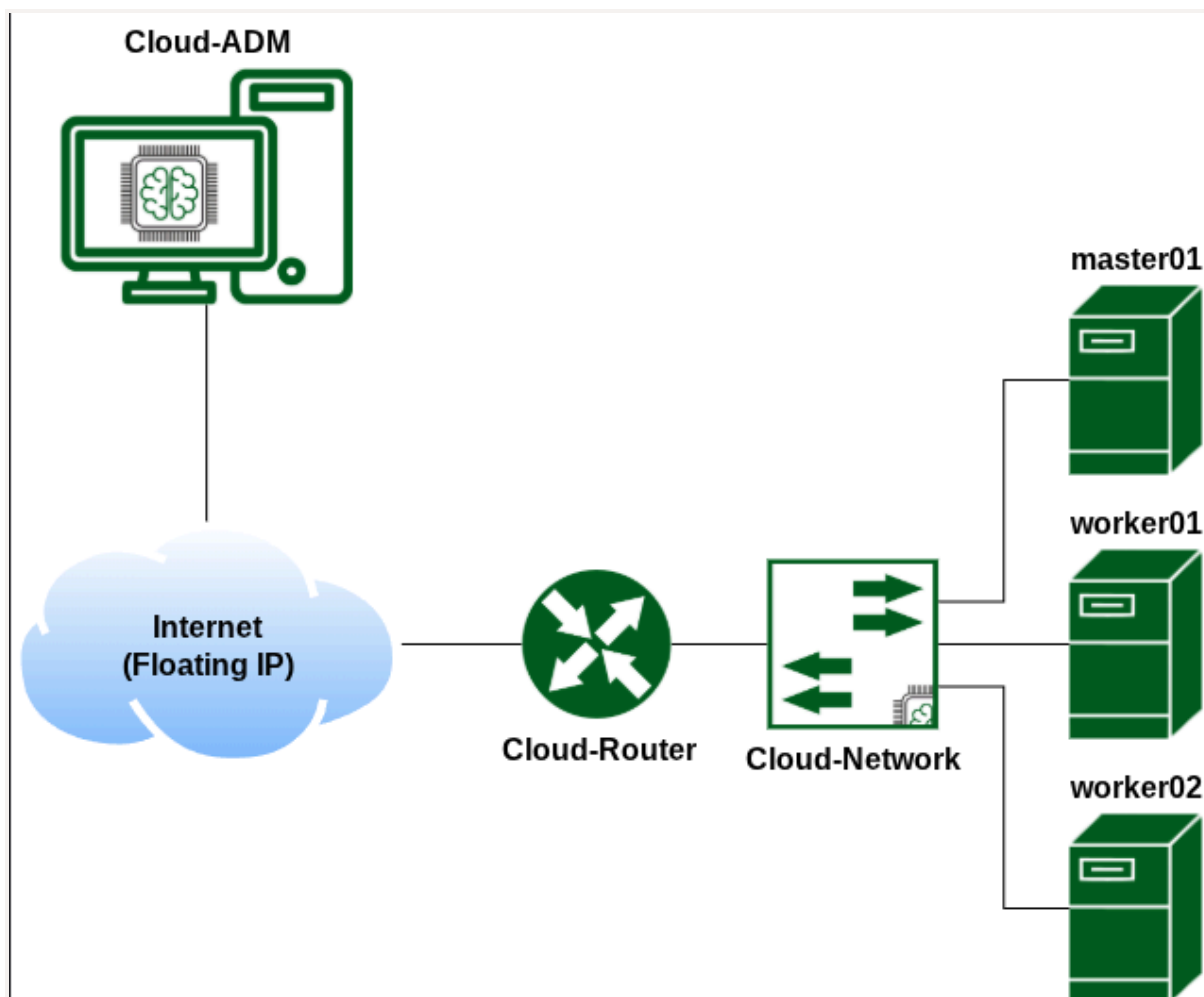
 Agent version: 18.0.38188

Installed agents:
Agent for Linux

Project_03

Приложение Project_03:

- <https://disk.yandex.ru/d/ltDZh8DTdK5kAA>



Задание:

а. Любым способом разверните недостающие виртуальные машины в облаке в соответствие с топологией (см. Топология Project_03).

1. Характеристики виртуальных машин:

1. master01: 1vCPU, 1 ГБ ОЗУ, 10 ГБ размер диска.
2. worker01: 1vCPU, 1 ГБ ОЗУ, 10 ГБ размер диска.
3. worker02: 1vCPU, 1 ГБ ОЗУ, 10 ГБ размер

2. Образ операционной системы: alt-p11-cloud-x86_64.qcow2.

б. На развёрнутых виртуальных машинах реализуйте следующий функционал:

1. Сформируйте кластер Docker Swarm.

1. Используйте VM master01 в качестве менеджера кластера, worker01 и worker02 - в качестве исполнителей.

2. Запускать контейнеры на менеджере запрещено.
2. Разместите в кластере Docker Swarm service для организации собственного хранилища образов контейнеров:
 1. В качестве образа используйте registry:3
 2. Локальное хранилище образов должно быть доступно со всех узлов кластера на порту 5000
3. Разместите в кластере Docker Swarm stack для организации работы приложения (см. Приложение Project_03)
 1. В качестве имени для stack используйте school-site-project
 2. Приложение должно быть доступно с Cloud-ADM по <http://school-site.au.team> и <https://school-site.au.team>
 3. В случае доступа по https, проблем с сертификатом возникать не должно.
4. На данный момент это просто статический сайт, но стоит предусмотреть возможность развёртывания таких сервисов в стеке как:
 1. redis - для решения задач, требующих быстрой обработки данных
 2. СУБД: PostgreSQL – для хранения данных
 3. Веб-приложение пока не умеет работать с данными сервисами, но сервисы должны быть развёрнуты в рамках стека school-site-project для будущей разработки

Вариант реализации:

Cloud-ADM:

Развёртывание недостающих виртуальных машин

- Допустимы любые способы, в том числе и ручное создание VM
- Модернизирует ранее написанные файлы для **Terraform** чтобы ускорить процесс создания VM

```
cd ~/Projects/
```

```
mkdir -p Project_03/terraform
```

```
cd Project_03/terraform
```

- За основу возьмём файлы из **Project02**:

```
cp ~/Projects/Project_02/terraform/provider.tf ./
```

```
cp ~/Projects/Project_02/terraform/cloud-init.yml ./
```

```
cp ~/Projects/Project_02/terraform/network.tf ./
```

```
cp ~/Projects/Project_02/terraform/vm.tf ./
```

- Инициализируем текущий каталог для работы с **Terraform**:

```
terraform init
```

- Модернизируем файл **network.tf** в текущей директории для **Project03**:

```
resource "openstack_networking_port_v2" "port_vm_master01" {
  name          = "port_master01"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.107"
  }
}

resource "openstack_networking_port_v2" "port_vm_worker01" {
  name          = "port_worker01"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.108"
  }
}

resource "openstack_networking_port_v2" "port_vm_worker02" {
  name          = "port_worker02"
  network_id    = "61845892-f9cc-4fde-962c-34b59425a74d"
  admin_state_up = true

  fixed_ip {
    subnet_id = "13592ca4-8782-410b-9bcc-90810ccab6fe"
    ip_address = "192.168.1.109"
  }
}
```

- Модернизируем файл **vm.tf** в текущей директории для **Project03**:

```

resource "openstack_compute_instance_v2" "master01" {
  name      = "master01"
  flavor_id = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data = file("cloud-init.yml")

  block_device {
    uuid           = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type    = "image"
    volume_size    = "10"
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_master01.id
  }
}

resource "openstack_compute_instance_v2" "worker01" {
  name      = "worker01"
  flavor_id = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data = file("cloud-init.yml")

  block_device {
    uuid           = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type    = "image"
    volume_size    = "10"
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_worker01.id
  }
}

resource "openstack_compute_instance_v2" "worker02" {
  name      = "worker02"
  flavor_id = "03bf1b85-2f5f-4ada-a07b-8b994b6dcb57"
  user_data = file("cloud-init.yml")

  block_device {
    uuid           = "827e08fa-fd3c-41cd-92ca-845bb5018478"
    source_type    = "image"
    volume_size    = "10"
    boot_index     = 0
    destination_type = "volume"
    delete_on_termination = true
  }

  network {
    port = openstack_networking_port_v2.port_vm_worker02.id
  }
}

```

- Запускаем автоматическое развёртывания VM для Project03 через Terraform:

```
terraform apply
```

- Результат:

```
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
```

- Результат в веб-интерфейсе облака:

Виртуальные машины

☐	Имя ↓	Статус ↓	IP-адрес	вЦП ↓	ОЗУ ↓	Хранилище
☐	 Cloud-ADM	 Запущена	192.168.1.10	2	4 ГиБ	30 ГиБ
☐	 ACM-Server	 Запущена	192.168.1.104	1	2 ГиБ	20 ГиБ
☐	 worker02	 Запущена	192.168.1.109	1	1 ГиБ	10 ГиБ
☐	 worker01	 Запущена	192.168.1.108	1	1 ГиБ	10 ГиБ
☐	 master01	 Запущена	192.168.1.107	1	1 ГиБ	10 ГиБ
☐	 DB-Server	 Запущена	192.168.1.105	1	1 ГиБ	20 ГиБ
☐	 BAR-Agent01	 Запущена	192.168.1.106	1	1 ГиБ	10 ГиБ

- Для удобства приводим конфигурационный файл `/etc/hosts` к следующему виду:

```
127.0.0.1    localhost.localdomain localhost
::1         localhost6.localdomain6 localhost6

192.168.1.100 haproxy01.dev.au.team haproxy01
192.168.1.100 game.au.team
192.168.1.101 game01.dev.au.team game01
192.168.1.102 game02.dev.au.team game02
192.168.1.103 game03.dev.au.team game03
192.168.1.104 acm-server.au.team acm-server
192.168.1.105 db-server.au.team db-server
192.168.1.106 bar-agent01.au.team bar-agent01
192.168.1.104 cb.au.team
192.168.1.107 master01.au.team master01
192.168.1.108 worker01.au.team worker01
192.168.1.109 worker02.au.team worker02
```

- Проверяем доступ до созданных VM:

```
[altlinux@cloud-adm terraform]$ ssh master01
The authenticity of host 'master01 (192.168.1.107)' can't be established.
ED25519 key fingerprint is SHA256:/qxbLF+rEU+WLVES1jpD01oHKdZ8PfYUTrbYoyabTQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'master01' (ED25519) to the list of known hosts.
[altlinux@master01 ~]$ exit
выход
Connection to master01 closed.
[altlinux@cloud-adm terraform]$ ssh worker01
The authenticity of host 'worker01 (192.168.1.108)' can't be established.
ED25519 key fingerprint is SHA256:l01E81WBZ/iDYIFmFHDvfWh4APPBcPeHt694N7CvJCU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'worker01' (ED25519) to the list of known hosts.
[altlinux@worker01 ~]$ exit
выход
Connection to worker01 closed.
[altlinux@cloud-adm terraform]$ ssh worker02
The authenticity of host 'worker02 (192.168.1.109)' can't be established.
ED25519 key fingerprint is SHA256:tl+SCihMDD028VIj2FULg1UoORqOwa14wWr4tqZ/104.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'worker02' (ED25519) to the list of known hosts.
[altlinux@worker02 ~]$ exit
выход
Connection to worker02 closed.
[altlinux@cloud-adm terraform]$
```

master01, worker01 и worker02:

- Установим пакет **docker-engine**:

```
apt-get update && apt-get install -y docker-engine
```

- Включить и добавить в автозагрузку службу **docker**:

```
systemctl enable --now docker
```

- Внести изменения в конфигурационный файл **/etc/docker/daemon.json**:
 - привести параметр **live-restore** к следующему виду:

```
{
  "init-path": "/usr/bin/tini",
  "userland-proxy-path": "/usr/bin/docker-proxy",
  "default-runtime": "docker-runc",
  "live-restore": false,
  "log-driver": "journald",
  "runtimes": {
    "docker-runc": {
      "path": "/usr/bin/runc"
    }
  },
  "default-ulimits": {
    "nofile": {
      "Name": "nofile",
      "Hard": 64000,
      "Soft": 64000
    }
  },
  "storage-driver": "overlay2"
}
```

- Перезагрузить службу **docker** для применения изменений:

```
systemctl restart docker
```

master01:

- Создаём кластер Docker Swarm:

```
docker swarm init
```

- Результат:
 - команда генерирует два случайных токена, рабочий токен и токен менеджера
 - когда добавляется новый узел к swarm, узел присоединяется как рабочий или управляющий узел на основе токена

```
[root@master01 ~]# docker swarm init
Swarm initialized: current node (g7ndsy8h08caticjokutm7jci) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-1v28dvd0cwkm5jqt4pipj52y5y7x6a6g06tbo7n60fz31hnrn-269v37bfivrr1peo5zhvbkv2 192.168.1.107:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

```
[root@master01 ~]#
```

worker01 и worker02:

- Добавляем узлы в кластер Docker Swarm:
 - worker01:

```
[root@worker01 ~]# docker swarm join --token SWMTKN-1-1v28dvd0cwksjqwt4pipj52y5y7x6a6g06tbo7n60fz31hnrn-269v37bfivrr1peo5zhvbkv2 192.168.1.107:2377
This node joined a swarm as a worker.
[root@worker01 ~]#
```

- worker02:

```
[root@worker02 ~]# docker swarm join --token SWMTKN-1-1v28dvd0cwksjqwt4pipj52y5y7x6a6g06tbo7n60fz31hnrn-269v37bfivrr1peo5zhvbkv2 192.168.1.107:2377
This node joined a swarm as a worker.
[root@worker02 ~]#
```

master01:

- Проверить наличие нод в кластере:

```
docker node ls
```

- Результат:

```
[root@master01 ~]# docker node ls
ID                                HOSTNAME    STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
g7ndsy8h00caticjokutm7jci *    master01    Ready    Active           Leader             28.3.3
j29m0cu58zidqiss8n141ckus        worker01    Ready    Active           -                  28.3.3
r78iq9yo5g5qxa4rleo3x3g89        worker02    Ready    Active           -                  28.3.3
[root@master01 ~]#
```

- Запускаем сервис для локального хранилища на базе образа **registry:3**:

```
docker service create \
  --name registry \
  --publish published=5000,target=5000 \
  --constraint node.role==worker \
  registry:3
```

- Проверьте статус запущенного сервиса можно с помощью **docker service ls**:

```
[root@master01 ~]# docker service ls
ID                NAME      MODE      REPLICAS    IMAGE      PORTS
f4a3nzaqbh0w     registry  replicated 1/1          registry:3 *:5000->5000/tcp
[root@master01 ~]#
```

- Проверить что задача развернулась именно на нод с ролью **worker** можно с помощью команды **docker service ps registry**:

```
[root@master01 ~]# docker service ps registry
ID                NAME      IMAGE      NODE      DESIRED STATE    CURRENT STATE    ERROR    PORTS
jziqmsnsk03y     registry.1 registry:3  worker01  Running           Running about a minute ago
[root@master01 ~]#
```

Cloud-ADM:

- Любым возможным способом скачиваем файлы приложения **Project03** и передаём их на **master01**:

```
scp Project03.zip master01:~/
```

master01:

- Устанавливаем пакет **unzip** и **docker-compose-v2**:
 - Docker Compose будет использоваться для тестирования сборки и запуска приложения с последующим добавлением его в ранее развёрнутое локальное хранилище

```
apt-get install -y unzip docker-compose-v2
```

- Распаковываем архив с приложением и переходим в директорию:

```
unzip /home/altlinux/Project03.zip -d ./
```

```
cd School-site-project-main/
```

- Создаём максимально минимальный **Dockerfile** для сборки образа приложения:

```
FROM nginx:1.29.3-alpine
COPY . /usr/share/nginx/html
```

- Создадим файл **compose.yaml** и укажем в нём следующее содержимое:
 - для сборки образа и тестирования приложения
 - образ называем именно так, чтобы в дальнейшем можно было его добавить в ранее развёрнутое локальное хранилище образов
 - можно использовать **127.0.0.1**, всё остальное за вас сделает маршрутизация в рамках кластера Docker Swarm

```
services:
  app:
    image: 127.0.0.1:5000/app
    build: .
    ports:
      - "80:80"
```

- Выполним сборку и запуск контейнера для проверки работоспособности:

```
docker compose up -d
```

- Проверить сборку образа можно с помощью команды **docker image ls**:

```
[root@master01 School-site-project-main]# docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
127.0.0.1:5000/app  latest     0df2346836e3  58 seconds ago  53MB
[root@master01 School-site-project-main]#
```

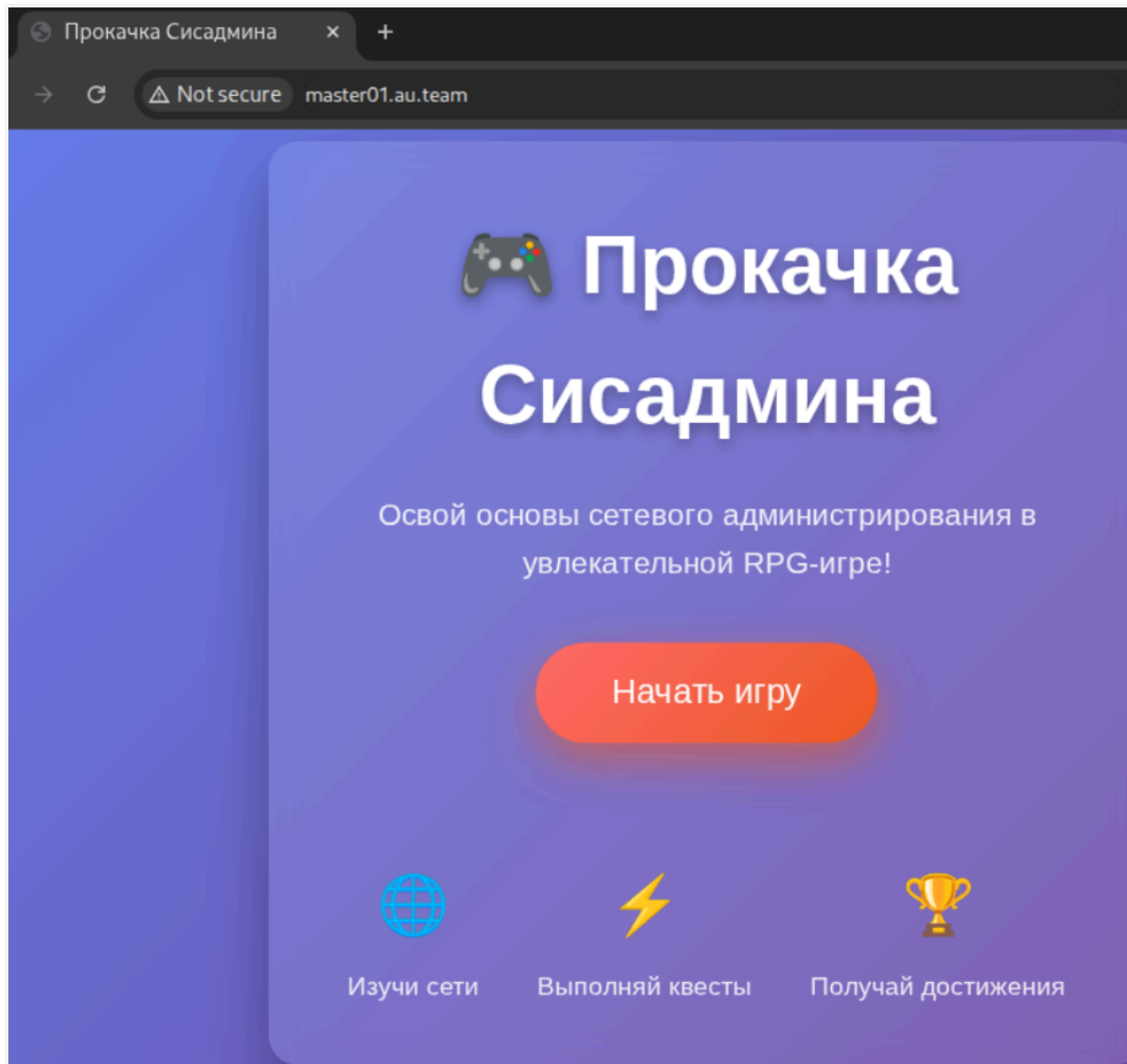
- Проверить сборку образа можно с помощью команды **docker image ls**:

```
[root@master01 School-site-project-main]# docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
127.0.0.1:5000/app  latest     0df2346836e3  58 seconds ago  53MB
[root@master01 School-site-project-main]#
```

- Проверить запущенный контейнер можно с помощью команды **docker compose ps**:

```
[root@master01 School-site-project-main]# docker compose ps
NAME                                IMAGE                                COMMAND                                SERVICE  CREATED
STATUS    PORTS
school-site-project-main-app-1     127.0.0.1:5000/app                  "/docker-entrypoint..."            app      2 minutes ago
Up 2 minutes    0.0.0.0:80->80/tcp, [::]:80->80/tcp
[root@master01 School-site-project-main]#
```

- Проверить работоспособность веб-приложения с **Cloud-ADM** в браузере:
 - обращаясь в веб-браузере на IP-адрес любой из нод:



- Удаляем всё что было использовано для тестового запуска:

```
docker compose down --volumes
```

```
docker image rm 127.0.0.1:5000/app
```

Cloud-ADM:

- Сгенерируем необходимые файлы сертификатов, через ранее развёрнутый ЦА:
 - создаём ключ:

```
openssl genrsa -out site.key 2048
```

- создаём запрос:

```
openssl req -key site.key -new -out site.csr
```

- Результат:

```
[altlinux@cloud-adm ~]$ openssl req -key site.key -new -out site.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [RU]:
State or Province Name (full name) []:
Locality Name (eg, city) []:
Organization Name (eg, company) []:au.team
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:school-site.au.team
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[altlinux@cloud-adm ~]$
```

- Файл с расширениями для сертификата **site.ext**:

```
cat <<EOF > site.ext
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
subjectAltName=@alt_names
[alt_names]
DNS.1=school-site.au.team
IP.1=192.168.1.107
EOF
```

- Выпускаем сертификат:

```
openssl x509 -req -CA Projects/Project_01/ansible/files/ca.crt -CAkey Projects/Project_01/ansible/files/ca.key -in site.csr -out site.crt -days 365 -CAcreateserial -extfile site.ext
```

```
openssl x509 -req -CA Projects/Project_01/ansible/files/ca.crt  
-CAkey Projects/Project_01/ansible/files/ca.key -in site.csr -out  
site.crt -days 365 -CAcreateserial -extfile site.ext
```

- Передаём ключ и сертификат на **master01**:

```
scp site.key master01:~/
```

```
scp site.crt master01:~/
```

master01:

- Копируем ключ и сертификат в директорию с файлами приложения:

```
cp /home/altlinux/site.crt ./
```

```
cp /home/altlinux/site.key ./
```

- Создаём свой конфигурационный файл для веб-сервера **nginx**, например **custom.conf**:

```
server {  
    listen      80;  
    server_name school-site.au.team;  
  
    location / {  
        root    /usr/share/nginx/html;  
        index   index.html index.htm;  
    }  
}  
  
server {  
    listen      443 ssl;  
    server_name school-site.au.team;  
  
    ssl_certificate /etc/nginx/site.crt;  
    ssl_certificate_key /etc/nginx/site.key;  
  
    location / {  
        root    /usr/share/nginx/html;  
        index   index.html index.htm;  
    }  
}
```

- Модернизируем ранее написанный **Dockerfile** для сборки образа с приложением:

```
FROM nginx:1.29.3-alpine
COPY site.crt site.key /etc/nginx/
COPY custom.conf /etc/nginx/conf.d/default.conf
COPY . /usr/share/nginx/html
```

- Модернизируем ранее написанный **compose.yaml**:

```
services:
  app:
    image: 127.0.0.1:5000/app
    build: .
    ports:
      - "80:80"
      - "443:443"
```

- Выполним сборку и запуск контейнера для проверки работоспособности:

```
docker compose up -d
```

- Проверить доступ к приложению с **Cloud-ADM** как по **http** так и по **https**
- Удаляем всё что было использовано для тестового запуска:

```
docker compose down --volumes
```

- Отправляем средствами Docker Compose образ нашего приложения в ранее развёрнутое хранилище образов:

```
docker compose push
```

- Результат:

```
[root@master01 School-site-project-main]# docker compose push
[+] Pushing 11/11
✓ Pushing 127.0.0.1:5000/app: 3631a80df5bd Pushed
✓ Pushing 127.0.0.1:5000/app: f37e191b8fbe Pushed
✓ Pushing 127.0.0.1:5000/app: fc249937edf3 Pushed
✓ Pushing 127.0.0.1:5000/app: 25906c27b84d Pushed
✓ Pushing 127.0.0.1:5000/app: 99ea4bde418d Pushed
✓ Pushing 127.0.0.1:5000/app: 3297b9628ff3 Pushed
✓ Pushing 127.0.0.1:5000/app: b74d92be8225 Pushed
✓ Pushing 127.0.0.1:5000/app: 2c79d5d895bb Pushed
✓ Pushing 127.0.0.1:5000/app: 2660a7d4b906 Pushed
✓ Pushing 127.0.0.1:5000/app: 50b58ca2a3f5 Pushed
✓ Pushing 127.0.0.1:5000/app: 256f393e029f Pushed
[root@master01 School-site-project-main]#
```

- Удаляем образ собранный локально:

```
docker image rm 127.0.0.1:5000/app
```

- Модернизируем ранее написанный **compose.yaml**:

```
services:
  app:
    image: 127.0.0.1:5000/app
    ports:
      - "80:80"
      - "443:443"
    deploy:
      placement:
        constraints:
          - "node.role==worker"

  redis:
    image: redis:8.4.0-alpine
    deploy:
      placement:
        constraints:
          - "node.role==worker"

  db:
    image: postgres:18.1-alpine3.22
    environment:
      POSTGRES_PASSWORD: "P@ssw0rd"
    deploy:
      placement:
        constraints:
          - "node.role==worker"
```

- Запускаем развёртывание стека с именем **school-site-project**:

```
docker stack deploy --compose-file compose.yaml school-site-project
```

- Проверить созданный стек в кластере можно с помощью команды **docker stack ls**:

```
[root@master01 School-site-project-main]# docker stack ls
NAME                SERVICES
school-site-project 3
[root@master01 School-site-project-main]#
```

- Проверить созданные сервисы в рамках стека можно с помощью команды **docker stack services school-site-project**:

```
[root@master01 School-site-project-main]# docker stack services school-site-project
ID                NAME                MODE                REPLICAS            IMAGE                PORTS
zid1pgjphj7v     school-site-project_app replicated          1/1                127.0.0.1:5000/app:latest *:80->80/tcp, *:443->443/tcp
vemqeafnlap8     school-site-project_db replicated          1/1                postgres:18.1-alpine3.22
zajc5vix6431     school-site-project_redis replicated          1/1                redis:8.4.0-alpine
```

- Проверить на каких нодах запустились задачи сервисов в рамках стека можно с помощью команды **docker stack ps school-site-project**:

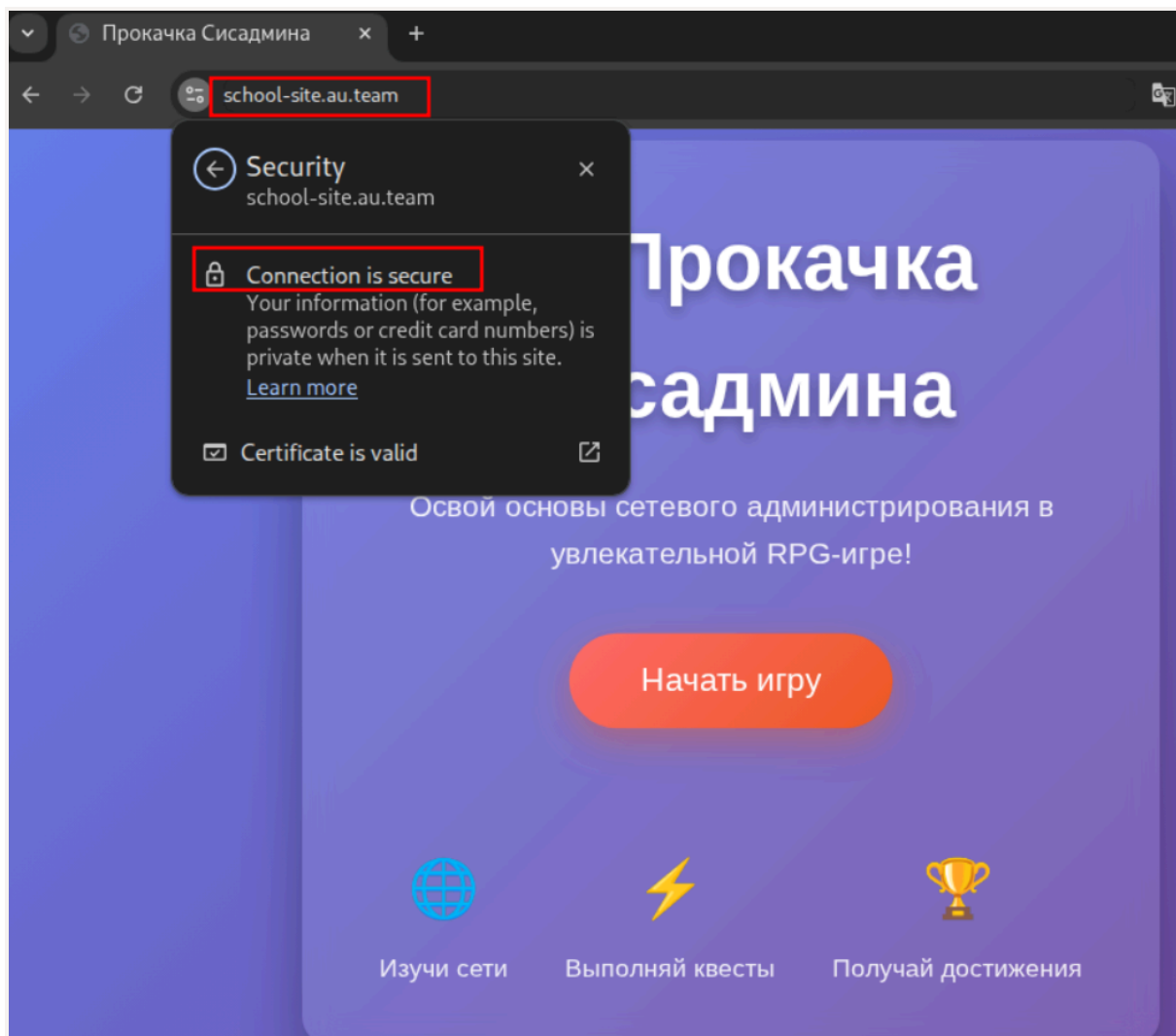
```
[root@master01 School-site-project-main]# docker stack ps school-site-project
ID                NAME                IMAGE                NODE                DESIRED STATE       CURRENT STATE
nusnkfdatt0o0     school-site-project_app.1 127.0.0.1:5000/app:latest worker01            Running              Running about a minute ago
vda64xmb6rf6     school-site-project_db.1  postgres:18.1-alpine3.22 worker02            Running              Running about a minute ago
7m47fw1j024i     school-site-project_redis.1 redis:8.4.0-alpine    worker02            Running              Running about a minute ago
```

Cloud-ADM:

- Добавляем в файл **/etc/hosts** следующую запись:

```
127.0.0.1      localhost.localdomain localhost
::1           localhost6.localdomain6 localhost6
192.168.1.100  haproxy01.dev.au.team  haproxy01
192.168.1.100  game.au.team
192.168.1.101  game01.dev.au.team     game01
192.168.1.102  game02.dev.au.team     game02
192.168.1.103  game03.dev.au.team     game03
192.168.1.104  acm-server.au.team     acm-server
192.168.1.105  db-server.au.team      db-server
192.168.1.106  bar-agent01.au.team    bar-agent01
192.168.1.104  cb.au.team
192.168.1.107  master01.au.team       master01
192.168.1.108  worker01.au.team       worker01
192.168.1.109  worker02.au.team       worker02
192.168.1.107  school-site.au.team
```

- Проверяем доступ к приложению:
 - <http://school-site.au.team>:



ЗАПАС 1.ds

project_01

Часть 1. Создание управляющей машины Cloud-ADM в Proxmox

Мы создадим виртуальную машину, которая будет выступать в роли "джемп-хоста" (бастиона) и центра управления проектом.

1. Загрузка образа ОС:

- Скачайте образ `alt-workstation-10.4-p10-cloud` (скорее всего в формате QCOW2) на ваш Proxmox-сервер.
- Обычно образы загружают в хранилище, например, в `local`.
Выполните в SSH вашего Proxmox узла:
- `bash`

```
cd /var/lib/vz/template/iso/
```

- `wget <ссылка_на_образ_alt-workstation-10.4-p10-cloud.qcow2>`

2. Создание виртуальной машины Cloud-ADM:

Выполните следующие команды в консоли Proxmox узла. Давайте назначим ей ID, например, `900`.

3. `bash`

1. Создаем виртуальную машину без диска

```
qm create 900 --name Cloud-ADM --memory 4096 --cores 2 --net0 virtio,bridge=vmbro
```

2. Импортируем скачанный диск в хранилище (например, local-lvm)

Убедитесь, что имя файла совпадает с тем, что вы скачали.

```
qm importdisk 900 /var/lib/vz/template/iso/alt-workstation-10.4-p10-cloud.qcow2 local-lvm
```

3. Прикрепляем импортированный диск как неиспользуемый диск 0

```
qm set 900 --scsihw virtio-scsi-pci --scsi0 local-lvm:vm-900-disk-0
```

4. Устанавливаем загрузку с диска

```
qm set 900 --boot c --bootdisk scsi0
```

5. Добавляем серийный консольный вывод (полезно для облачных образов)

```
qm set 900 --serial0 socket --vga serial0
```

6. Добавляем ключ SSH для доступа (опционально, но рекомендуется)

Замените "ваш-публичный-ключ" на содержимое вашего id_rsa.pub

```
qm set 900 --sshkey ~/.ssh/id_rsa.pub
```

7. Настраиваем размер диска (30 ГБ)

```
qm resize 900 scsi0 30G
```

8. Включаем агент QEMU (для корректного выключения и информации об IP)

4. `qm set 900 --agent 1`

5. Запуск и настройка сети:

- Запустите машину в веб-интерфейсе Proxmox.
 - Войдите в неё через консоль в веб-интерфейсе (логин: `altlinux`, пароль уточните у образа, обычно задается через `cloud-init`, но в простых случаях может быть пустым или `altlinux`).
 - Настройте статический IP или убедитесь, что DHCP выдал адрес.
 - Критически важно для связи с будущими VM: Создайте внутреннюю сеть. В Proxmox это делается созданием моста (Linux Bridge). Добавьте новый мост, например, `vmbbr1` без физического интерфейса (Bridge ports оставьте пустым). Подключите к этому мосту сетевую карту вашей VM Cloud-ADM.
6. Настройка доступа с локального ПК:
- Putty (SSH): Узнайте IP-адрес вашей машины Cloud-ADM в сети, доступной с вашего ПК. Создайте в Putty сессию с этим IP, сохраните её как "Cloud-ADM".
 - RDP: Установите на Cloud-ADM пакет `xrdp`: `sudo apt-get update && sudo apt-get install xrdp`. Настройте пароль для пользователя `altlinux` (или используйте `P@ssw0rd`, если он уже установлен). На локальном ПК создайте RDP-подключение к IP-адресу Cloud-ADM и сохраните профиль.
-

Часть 2. Скрипты развертывания проекта в Cloud-ADM

Зайдите на машину Cloud-ADM по SSH. Все скрипты будут выполняться на ней.

Подготовка:

```
bash
mkdir -p /home/altlinux/Projects/Project_01
cd /home/altlinux/Projects/Project_01
```

2.1 Скрипт `deploy_project_01.sh`

Этот скрипт будет использовать инструменты командной строки Proxmox (`pvesh`, `qm`) для создания VM. Для удаленного управления с Cloud-ADM, вам нужно либо запускать их по SSH на Proxmox-хосте (настроив ключи), либо настроить API-доступ.

Упрощенный вариант с запуском команд на хосте через SSH:

```
bash
```

```
#!/bin/bash
```

```
# deploy_project_01.sh
```

```
# Параметры подключения к Proxmox хосту (ЗАМЕНИТЕ НА СВОИ!)
```

```
PM_HOST="192.168.1.10" # IP вашего Proxmox сервера
```

```
PM_USER="root"
```

```
SSH_KEY="/home/altlinux/.ssh/id_rsa" # Сгенерируйте ключ и добавьте на Proxmox
```

```
# Параметры сети Proxmox (ЗАМЕНИТЕ НА СВОИ!)
```

```
TARGET_STORAGE="local-lvm" # Хранилище для дисков
```

```
NET_BRIDGE="vmb1" # Мост для внутренней сети
```

```
GATEWAY="192.168.100.1" # Ваш внутренний шлюз
```

```
NETMASK="24"
```

```
DNS="8.8.8.8"
```

```
echo "[*] Начинаю развертывание VM в Proxmox..."
```

```
# ID для новых VM (убедитесь, что они свободны)
```

```
HAPROXY_VMID=101
```

```
GAME1_VMID=102
```

```
GAME2_VMID=103
```

```
GAME3_VMID=104
```

```
# Функция создания VM
```

```
create_vm() {
```

```
    local vmid=$1
```

```
    local name=$2
```

```
    local ip=$3
```

```
    echo "[*] Создание $name (ID: $vmid) с IP $ip"
```

```
    ssh -i $SSH_KEY $PM_USER@$PM_HOST "
```

```
        qm create $vmid --memory 1024 --cores 1 --name $name --net0
```

```
virtio,bridge=$NET_BRIDGE
```

```
        qm set $vmid --scsihw virtio-scsi-pci
```

```
        qm set $vmid --ide2 $TARGET_STORAGE:cloudinit
```

```
        qm set $vmid --boot c --bootdisk scsi0
```

```
        qm set $vmid --serial0 socket --vga serial0
```

```
        qm set $vmid --agent 1
```

```
    # Настройка сети через cloud-init
```

```
    qm set $vmid --ipconfig0 ip=$ip/$NETMASK,gw=$GATEWAY
```

```
    qm set $vmid --nameserver $DNS
```

```
qm set $vmid --ciuser altlinux
qm set $vmid --cipassword P@ssw0rd
qm set $vmid --sshkey /root/.ssh/authorized_keys # Подставьте свой
ключ
```

```
# Скачивание и импорт образа OS (если еще не скачан)
if [ ! -f /var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 ];
then
    wget -O /var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2
<URL_образа_alt-p11>
fi
qm importdisk $vmid
/var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 $TARGET_STORAGE
qm set $vmid --scsi0 $TARGET_STORAGE:vm-$vmid-disk-0
qm resize $vmid scsi0 10G
qm start $vmid
"
echo "[+] $name создана и запущена."
}
```

Создаем VM. IP-адреса должны быть в одной подсети с мостом vmbr1.

```
create_vm $HAPROXY_VMID "haproxy01" "192.168.100.10"
create_vm $GAME1_VMID "game01" "192.168.100.11"
create_vm $GAME2_VMID "game02" "192.168.100.12"
create_vm $GAME3_VMID "game03" "192.168.100.13"
```

```
echo "[+] Все VM созданы. Ожидание запуска..."
sleep 30
```

```
echo "[!] ВАЖНО: Для 'Плавающего IP' вручную настройте проброс портов на
вашем роутере или используйте iptables на хосте Прохтох, перенаправляя порт
443 на IP 192.168.100.10."
```

```
echo "[!] Либо создайте отдельный мост с выходом в интернет и назначьте
haproxy01 реальный IP из вашей локальной сети."
```

2.2 Скрипт `configure_project_01.sh`

Этот скрипт настраивает ПО внутри созданных машин. Он должен уметь подключаться к ним по SSH.

```
bash
```

```
#!/bin/bash
# configure_project_01.sh
```

```

SSH_USER="altlinux"
SSH_PASS="P@ssw0rd" # Для sshpass или настройки ключей

# Цвета для вывода
RED='\033[0;31m'
GREEN='\033[0;32m'
NC='\033[0m'

echo -e "${GREEN}[*] Начинаем настройку серверов...${NC}"

# Функция установки Docker на ALT Linux
install_docker_alt() {
    local host=$1
    echo "[*] Установка Docker на $host"
    sshpass -p "$SSH_PASS" ssh -o StrictHostKeyChecking=no $SSH_USER@$host "
        sudo apt-get update
        sudo apt-get install -y docker-engine docker-compose
        sudo systemctl enable --now docker
        sudo usermod -aG docker $SSH_USER
    "
}

# Функция деплоя приложения (мультистейджинг билд)
deploy_app() {
    local host=$1
    local app_name=$2
    echo "[*] Деплой приложения $app_name на $host"

    # Создаем Dockerfile с мультистейджинг билдом
    sshpass -p "$SSH_PASS" ssh $SSH_USER@$host "
        mkdir -p ~/app
        cat > ~/app/Dockerfile << 'EOF'
# 1-stage: build
FROM golang:1.20-alpine AS builder
WORKDIR /app
RUN echo 'package main; import (\n"fmt\n"; \n"net/http\n"); func main() {
http.HandleFunc(\n"/\n", func(w http.ResponseWriter, r *http.Request) {
fmt.Fprintf(w, \n"Hello from $app_name\n") }); http.ListenAndServe(\n":80\n",
nil) }' > main.go
RUN go build -o myapp main.go

# 2-stage: run
FROM alpine:latest
WORKDIR /root/

```

```

COPY --from=builder /app/myapp .
EXPOSE 80
CMD [\"./myapp\"]
EOF

    cd ~/app
    docker build -t $app_name:latest .
    docker run -d --name $app_name --restart unless-stopped -p 80:80
$app_name:latest
"

    echo -e "${GREEN}[+] Приложение $app_name запущено на $host${NC}"
}

# Установка Docker на игровые сервера
for server in game01 game02 game03; do
    install_docker_alt $server
done

# Деплой приложений
deploy_app "game01" "game01"
deploy_app "game02" "game02"
deploy_app "game03" "game03"

# Настройка HAProxy на haproxy01
echo "[*] Настройка HAProxy и SSL на haproxy01..."
sshpas -p "$SSH_PASS" ssh $SSH_USER@haproxy01 "
    sudo apt-get update
    sudo apt-get install -y haproxy certbot

    # Генерация самоподписанного сертификата (чтобы не было проблем)
    sudo mkdir -p /etc/ssl/private
    sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \\\
        -keyout /etc/ssl/private/haproxy.key \\\
        -out /etc/ssl/certs/haproxy.crt \\\
        -subj "/C=RU/ST=State/L=City/O=Org/CN=game.au.team"
    # Объединяем для HAProxy
    cat /etc/ssl/certs/haproxy.crt /etc/ssl/private/haproxy.key | sudo tee
/etc/ssl/private/haproxy.pem

    # Конфигурация HAProxy
    sudo cat > /etc/haproxy/haproxy.cfg << 'EOF'
global
    log /dev/log local0
    maxconn 4096
    user haproxy
    group haproxy
    daemon

```

```

defaults
    log global
    mode http
    option httplog
    option dontlognull
    retries 3
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http-in
    bind *:80
    bind *:443 ssl crt /etc/ssl/private/haproxy.pem
    http-request redirect scheme https unless { ssl_fc } # Принудительный
HTTPS

    # ACL для доменных имен
    acl host_game01 hdr(host) -i game01.dev.au.team
    acl host_game02 hdr(host) -i game02.dev.au.team
    acl host_game03 hdr(host) -i game03.dev.au.team
    acl host_stats    hdr(host) -i haproxy01.dev.au.team

    # Маршрутизация
    use_backend game01-cluster if host_game01
    use_backend game02-cluster if host_game02
    use_backend game03-cluster if host_game03
    use_backend stats          if host_stats

    # По умолчанию на game01
    default_backend game01-cluster

backend game01-cluster
    server game01 192.168.100.11:80 check

backend game02-cluster
    server game02 192.168.100.12:80 check

backend game03-cluster
    server game03 192.168.100.13:80 check

backend stats
    stats enable
    stats uri /haproxy?stats
    stats refresh 10s
    stats auth admin:password # Задайте свой пароль

```

EOF

```
sudo systemctl restart haproxy
sudo systemctl enable haproxy
"

# Настройка файла hosts на Cloud-ADM
echo "[*] Настройка /etc/hosts на Cloud-ADM..."
sudo tee -a /etc/hosts << EOF
192.168.100.10 haproxy01 haproxy01.dev.au.team game.au.team
192.168.100.11 game01.dev.au.team
192.168.100.12 game02.dev.au.team
192.168.100.13 game03.dev.au.team
EOF

echo -e "${GREEN}[+] Настройка завершена!${NC}"
echo "Статистика HAProxy: http://haproxy01.dev.au.team/haproxy?stats"

echo "Приложение: https://game.au.team (будет перенаправлять на game01,
game02, game03 в зависимости от имени)"
```

2.3 Скрипт `destroy_project_01.sh`

```
bash

#!/bin/bash
# destroy_project_01.sh

PM_HOST="192.168.1.10"
PM_USER="root"
SSH_KEY="/home/altlinux/.ssh/id_rsa"
VM_IDS=(101 102 103 104)

echo "[*] Уничтожение созданных ресурсов..."

for vmid in "${VM_IDS[@]}; do
    echo "[*] Остановка и удаление ВМ с ID: $vmid"
    ssh -i $SSH_KEY $PM_USER@$PM_HOST "
        qm stop $vmid --skiplock
        sleep 5
        qm destroy $vmid --purge
    "
done

# Очистка hosts файла (опционально, аккуратно)
```

```
echo "[*] Очистка /etc/hosts..."
sudo sed -i '/dev.au.team/d' /etc/hosts
sudo sed -i '/game.au.team/d' /etc/hosts
```

```
echo "[+] Все ресурсы удалены."
```

Заключение

1. Сделайте скрипты исполняемыми: `chmod +x *.sh`.
2. Запускайте последовательно:
 - `./deploy_project_01.sh`
 - `./configure_project_01.sh`
 - (После проверки) `./destroy_project_01.sh`

Важные замечания для Proxmox:

- Сеть: Вам нужно создать мост (`vmbbr1`) без физического порта, чтобы изолировать VM проекта. Cloud-ADM должна быть подключена к этому мосту, чтобы видеть остальные сервера.
- Плавающий IP: В Proxmox нет нативного аналога Floating IP. Вам придется настроить NAT или проброс портов на уровне вашего роутера/хоста Proxmox, чтобы трафик из интернета на порт 443 попадал на внутренний IP `haproxy01` (192.168.100.10).

PROJECT_02

```
# Создаем директорию проекта
```

```
mkdir -p /home/altlinux/Projects/Project_02
```

```
cd /home/altlinux/Projects/Project_02
```

```
# Создаем директорию для ключей и конфигов
```

```
mkdir -p configs scripts
```

Скрипт 1: deploy_project_02.sh

Этот скрипт создает все необходимые виртуальные машины в Proxmox.

```
bash
```

```
cat > /home/altlinux/Projects/Project_02/deploy_project_02.sh <<
'EOF'

#!/bin/bash

# deploy_project_02.sh

# Скрипт для развертывания инфраструктуры Project_02 в Proxmox

# Конфигурация Proxmox

PM_HOST="192.168.1.10" # ЗАМЕНИТЕ НА IP ВАШЕГО PROXMOX ХОСТА

PM_USER="root"

SSH_KEY="/home/altlinux/.ssh/id_rsa" # Сгенерируйте ключ и добавьте
на Proxmox

# Параметры хранилища и сети

STORAGE="local-lvm"

BRIDGE_WAN="vmbr0"      # Мост для доступа в интернет

BRIDGE_LAN="vmbr2"      # Мост для внутренней сети Project_02

# Проверка наличия SSH ключа

if [ ! -f "$SSH_KEY" ]; then
```

```

    echo "[-] SSH ключ не найден. Генерируем новый..."

    ssh-keygen -t rsa -b 2048 -f /home/altlinux/.ssh/id_rsa -N ""

    echo "[!] Скопируйте публичный ключ на Proxmox хост:"

    echo "    ssh-copy-id root@$PM_HOST"

    exit 1

fi

# Функция создания VM в Proxmox

create_vm_proxmox() {

    local vmid=$1

    local name=$2

    local memory=$3

    local disk=$4

    local ip=$5

    echo "[*] Создание $name (ID: $vmid, IP: $ip) на Proxmox..."

    # Проверяем, не занят ли VMID

    if ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm status $vmid"
2>/dev/null; then

        echo "[-] VM с ID $vmid уже существует. Пропускаем..."

        return 1

```

```
fi
```

```
# Создаем VM
```

```
ssh -i $SSH_KEY $PM_USER@$PM_HOST "
```

```
# Создаем базовую конфигурацию
```

```
qm create $vmid --name $name --memory $memory --cores 1 \
```

```
    --net0 virtio,bridge=$BRIDGE_LAN \
```

```
    --scsihw virtio-scsi-pci \
```

```
    --ide2 $STORAGE:cloudinit \
```

```
    --boot c --bootdisk scsi0 \
```

```
    --serial0 socket --vga serial0 \
```

```
    --agent 1
```

```
# Настройка cloud-init
```

```
qm set $vmid --ipconfig0 ip=$ip/24,gw=192.168.200.1
```

```
qm set $vmid --nameserver 8.8.8.8
```

```
qm set $vmid --ciuser altlinux
```

```
qm set $vmid --cipassword P@ssw0rd
```

```
# Импорт образа ОС
```

```
if [ ! -f /var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 ]; then
```

```
        echo '[-] Образ не найден. Скачайте  
alt-p11-cloud-x86_64.qcow2 в /var/lib/vz/template/iso/'
```

```
        exit 1
```

```
    fi
```

```
        qm importdisk $vmid  
/var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 $STORAGE
```

```
        qm set $vmid --scsi0 $STORAGE:vm-$vmid-disk-0
```

```
        qm resize $vmid scsi0 ${disk}G
```

```
        # Запускаем VM
```

```
        qm start $vmid
```

```
    "
```

```
    if [ $? -eq 0 ]; then
```

```
        echo "[+] $name успешно создана и запущена"
```

```
    else
```

```
        echo "[-] Ошибка при создании $name"
```

```
    fi
```

```
}
```

```
# Создаем Cloud-Router (ID: 201)
```

```
echo "[*] Создание Cloud-Router..."
```

```
create_vm_proxmox 201 "Cloud-Router" 512 5 "192.168.200.1"

# Создаем ACM-Server (ID: 202)

echo "[*] Создание ACM-Server..."

create_vm_proxmox 202 "ACM-Server" 2048 20 "192.168.200.10"


# Создаем DB-Server (ID: 203)

echo "[*] Создание DB-Server..."

create_vm_proxmox 203 "DB-Server" 1024 20 "192.168.200.11"


# Создаем BAR-Agent01 (ID: 204)

echo "[*] Создание BAR-Agent01..."

create_vm_proxmox 204 "BAR-Agent01" 1024 10 "192.168.200.12"


echo "[*] Ожидание полной загрузки VM (60 секунд)..."

sleep 60


# Обновляем /etc/hosts на Cloud-ADM

echo "[*] Обновление /etc/hosts на Cloud-ADM..."

sudo tee -a /etc/hosts << HOSTS

192.168.200.1 cloud-router

192.168.200.10 acm-server cb.au.team
```

```
192.168.200.11 db-server
```

```
192.168.200.12 bar-agent01
```

```
HOSTS
```

```
echo "[+] Деплой Project_02 завершен!"
```

```
echo "    Cloud-Router: 192.168.200.1"
```

```
echo "    ACM-Server:   192.168.200.10"
```

```
echo "    DB-Server:    192.168.200.11"
```

```
echo "    BAR-Agent01:  192.168.200.12"
```

```
EOF
```

```
chmod +x /home/altlinux/Projects/Project_02/deploy_project_02.sh
```

Скрипт 2: configure_router.sh

```
bash
```

```
cat > /home/altlinux/Projects/Project_02/configure_router.sh << 'EOF'
```

```
#!/bin/bash
```

```
# configure_router.sh
```

```
# Настройка Cloud-Router для NAT и маршрутизации
```

```
ROUTER_IP="192.168.200.1"
```

```
PASSWORD="P@ssw0rd"
```

```
echo "[*] Настройка Cloud-Router ($ROUTER_IP)..."
```

```
# Функция для выполнения команд на роутере
```

```
exec_on_router() {
```

```
    sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no  
    altlinux@$ROUTER_IP "sudo -i $1"
```

```
}
```

```
# Проверка доступности роутера
```

```
echo "[*] Проверка доступности роутера..."
```

```
if ! ping -c 3 $ROUTER_IP &>/dev/null; then
```

```
    echo "[-] Роутер недоступен. Проверьте сеть."
```

```
    exit 1
```

```
fi
```

```
# Настройка роутера
```

```
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no  
altlinux@$ROUTER_IP << 'SSH_EOF'
```

```
sudo -i
```

```
# Обновление системы
```

```
echo "[*] Обновление системы..."
```

```
apt-get update
```

```
apt-get install -y iptables dnsmasq
```

```
# Включаем IP forwarding
```

```
echo "[*] Включение IP forwarding..."
```

```
echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
```

```
sysctl -p
```

```
# Настройка NAT
```

```
echo "[*] Настройка NAT..."
```

```
# Очищаем старые правила
```

```
iptables -t nat -F
```

```
iptables -F
```

```
# Добавляем новые правила
```

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
iptables -A FORWARD -i eth0 -o eth1 -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

```
# Сохраняем правила
```

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

```
# Создаем сервис для восстановления правил при загрузке
```

```
cat > /etc/systemd/system/iptables-restore.service << 'SERVICE'
```

```
[Unit]
```

```
Description=Restore iptables rules
```

```
Before=network.target
```

```
[Service]
```

```
Type=oneshot
```

```
ExecStart=/sbin/iptables-restore /etc/iptables.rules
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
SERVICE
```

```
systemctl enable iptables-restore.service
```

```
# Настройка dnsmasq для DNS и DHCP
```

```
echo "[*] Настройка dnsmasq..."
```

```
cat > /etc/dnsmasq.conf << 'DNS'
```

```
interface=eth1
```

```
dhcp-range=192.168.200.50,192.168.200.150,12h
```

```
dhcp-option=3,192.168.200.1
```

```
dhcp-option=6,8.8.8.8
```

```
# Локальные DNS записи
```

```
address=/acm-server.local/192.168.200.10
```

```
address=/db-server.local/192.168.200.11
```

```
address=/bar-agent01.local/192.168.200.12
```

```
address=/cb.au.team/192.168.200.10
```

```
DNS
```

```
systemctl enable --now dnsmasq
```

```
# Проверка настроек
```

```
echo "[*] Проверка NAT и маршрутизации..."
```

```
iptables -t nat -L -n -v
```

```
echo "[+] Cloud-Router настроен успешно!"
```

```
echo "      Внутренние машины теперь имеют доступ в интернет"
```

```
SSH_EOF
```

```
# Проверка доступа в интернет через роутер

echo "[*] Проверка доступа в интернет через роутер..."

sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$ROUTER_IP "ping -c 2 8.8.8.8" &>/dev/null


if [ $? -eq 0 ]; then

    echo "[+] Роутер имеет доступ в интернет"

else

    echo "[-] Роутер НЕ имеет доступ в интернет. Проверьте
подключение eth0."

fi


echo "[+] Настройка роутера завершена!"

EOF
```

```
chmod +x /home/altlinux/Projects/Project_02/configure_router.sh
```

Скрипт 3: install_cb.sh (основной скрипт установки Кибер Бэкап)

```
bash
```

```
cat > /home/altlinux/Projects/Project_02/install_cb.sh << 'EOF'
```

```
#!/bin/bash
```

```
# install_cb.sh
```

```
# Установка Кибер Бэкап и агентов
```

```
PASSWORD="P@ssw0rd"
```

```
ACM_SERVER="192.168.200.10"
```

```
DB_SERVER="192.168.200.11"
```

```
BAR_AGENT="192.168.200.12"
```

```
# Цвета для вывода
```

```
RED='\033[0;31m'
```

```
GREEN='\033[0;32m'
```

```
YELLOW='\033[1;33m'
```

```
NC='\033[0m'
```

```
echo -e "${GREEN}[*] Начинаем установку Кибер Бэкап...${NC}"
```

```
# Функция проверки доступности сервера
```

```
wait_for_server() {
```

```
    local server=$1
```

```
    local max_attempts=30
```

```
    local attempt=1
```

```

echo -n "[*] Ожидание загрузки $server"

while ! ping -c 1 $server &>/dev/null; do

    if [ $attempt -ge $max_attempts ]; then

        echo -e "\n${RED}[-] Сервер $server недоступен${NC}"

        return 1

    fi

    echo -n "."

    sleep 5

    ((attempt++))

done

echo -e "\n${GREEN}[+] Сервер $server доступен${NC}"

return 0

}

# Проверка доступности всех серверов

echo "[*] Проверка доступности серверов..."

wait_for_server $ACM_SERVER || exit 1

wait_for_server $DB_SERVER || exit 1

wait_for_server $BAR_AGENT || exit 1

# 1. Установка PostgreSQL и Кибер Бэкап на ACM-Server

echo -e "${GREEN}[*] Установка на ACM-Server ($ACM_SERVER)...${NC}"

```

```
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no  
altlinux@$ACM_SERVER << 'SSH_ACM'
```

```
sudo -i
```

```
# Установка PostgreSQL
```

```
echo "[*] Установка PostgreSQL..."
```

```
apt-get update
```

```
apt-get install -y postgresql15-server postgresql15-contrib
```

```
# Инициализация PostgreSQL
```

```
echo "[*] Инициализация PostgreSQL..."
```

```
/etc/init.d/postgresql initdb
```

```
# Настройка PostgreSQL для удаленного доступа
```

```
sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'/"  
/var/lib/pgsql/data/postgresql.conf
```

```
echo "host all all 0.0.0.0/0 md5" >> /var/lib/pgsql/data/pg_hba.conf
```

```
# Запуск PostgreSQL
```

```
systemctl enable postgresql
```

```
systemctl start postgresql
```

```
# Создание базы данных для Кибер Бэкап
```

```
echo "[*] Создание базы данных..."
```

```
sudo -u postgres psql << PSQL
```

```
CREATE USER backup_user WITH PASSWORD 'backup_pass';
```

```
CREATE DATABASE backup_db OWNER backup_user;
```

```
GRANT ALL PRIVILEGES ON DATABASE backup_db TO backup_user;
```

```
\c backup_db
```

```
GRANT ALL ON SCHEMA public TO backup_user;
```

```
PSQL
```

```
# Скачивание и установка Кибер Бэкап
```

```
echo "[*] Установка Кибер Бэкап сервера..."
```

```
cd /tmp
```

```
# Замените URL на актуальные для Кибер Бэкап версии 18
```

```
wget https://example.com/cyberbackup-server-18.0.x86_64.rpm || {
```

```
    echo "[-] Не удалось скачать Кибер Бэкап. Создаем заглушку для  
    демонстрации..."
```

```
# Создаем заглушку для демонстрации
```

```
mkdir -p /opt/cyberbackup/bin
```

```
cat > /opt/cyberbackup/bin/backup-cli << 'CLI'
```

```
#!/bin/bash
```

```
echo "Кибер Бэкап CLI (демо-режим)"
```

```
CLI
```

```
chmod +x /opt/cyberbackup/bin/backup-cli
```

```
# Создаем systemd сервис
```

```
cat > /etc/systemd/system/cyberbackup-server.service << 'SERVICE'
```

```
[Unit]
```

```
Description=Cyber Backup Server
```

```
After=network.target postgresql.service
```

```
[Service]
```

```
Type=simple
```

```
ExecStart=/bin/bash -c 'while true; do echo "Кибер Бэкап сервер  
работает..." >> /var/log/cyberbackup.log; sleep 60; done'
```

```
User=root
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
SERVICE
```

```
systemctl daemon-reload
```

```
systemctl enable cyberbackup-server

systemctl start cyberbackup-server

}

# Настройка конфигурации сервера

mkdir -p /etc/cyberbackup

cat > /etc/cyberbackup/backup-server.conf << CONF

DB_TYPE=postgresql

DB_HOST=localhost

DB_PORT=5432

DB_NAME=backup_db

DB_USER=backup_user

DB_PASSWORD=backup_pass


SERVER_PORT=9877

LOG_LEVEL=info


# Настройки доступа

ADMIN_USER=root

ADMIN_PASSWORD=toor

CONF
```

```
# Настройка пароля root для веб-интерфейса
```

```
echo "root:toor" | chpasswd
```

```
# Открываем порт в firewall
```

```
iptables -A INPUT -p tcp --dport 9877 -j ACCEPT
```

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

```
echo "[+] ACM-Server настроен. Кибер Бэкап сервер установлен."
```

```
echo "    Веб-интерфейс будет доступен на порту 9877"
```

```
SSH_ACM
```

```
# 2. Установка агента на BAR-Agent01
```

```
echo -e "${GREEN}[*] Установка на BAR-Agent01 ($BAR_AGENT)...${NC}"
```

```
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no  
altlinux@$BAR_AGENT << 'SSH_BAR'
```

```
sudo -i
```

```
# Обновление системы
```

```
apt-get update
```

```
# Установка агента Кибер Бэкап
```

```
echo "[*] Установка агента Кибер Бэкап..."
```

```
cd /tmp
```

```
# Замените URL на актуальный
```

```
wget https://example.com/cyberbackup-agent-18.0.x86_64.rpm || {
```

```
    echo "[-] Не удалось скачать агент. Создаем заглушку..."
```

```
    # Создаем заглушку агента
```

```
    mkdir -p /opt/cyberbackup-agent/bin
```

```
    cat > /opt/cyberbackup-agent/bin/agent << 'AGENT'
```

```
#!/bin/bash
```

```
echo "Cyber Backup Agent (демо-режим)"
```

```
while true; do
```

```
    echo "Агент работает..." >> /var/log/cyberbackup-agent.log
```

```
    sleep 60
```

```
done
```

```
AGENT
```

```
chmod +x /opt/cyberbackup-agent/bin/agent
```

```
# Создаем systemd сервис
```

```
cat > /etc/systemd/system/cyberbackup-agent.service << 'SERVICE'
```

```
[Unit]
```

```
Description=Cyber Backup Agent
```

```
After=network.target
```

```
[Service]
```

```
Type=simple
```

```
ExecStart=/opt/cyberbackup-agent/bin/agent
```

```
User=root
```

```
Restart=always
```

```
[Install]
```

```
WantedBy=multi-user.target
```

```
SERVICE
```

```
}
```

```
# Настройка конфигурации агента
```

```
mkdir -p /etc/cyberbackup
```

```
cat > /etc/cyberbackup/backup-agent.conf << CONF
```

```
SERVER_HOST=192.168.200.10
```

```
SERVER_PORT=9877
```

```
CLIENT_NAME=bar-agent01
```

```
CONF
```

```
# Запуск агента
```

```
systemctl daemon-reload
```

```
systemctl enable cyberbackup-agent
```

```
systemctl start cyberbackup-agent
```

```
# Регистрация агента на сервере
```

```
echo "[*] Регистрация агента на сервере управления..."
```

```
/opt/cyberbackup-agent/bin/agent --register --server 192.168.200.10  
--name bar-agent01 || {
```

```
    echo "[-] Не удалось зарегистрировать агент, но сервис запущен"
```

```
}
```

```
echo "[+] BAR-Agent01 настроен"
```

```
SSH_BAR
```

```
# 3. Установка PostgreSQL и агента на DB-Server
```

```
echo -e "${GREEN}[*] Установка на DB-Server ($DB_SERVER)...${NC}"
```

```
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no  
altlinux@$DB_SERVER << 'SSH_DB'
```

```
sudo -i
```

```
# Установка PostgreSQL
```

```
echo "[*] Установка PostgreSQL..."
```

```
apt-get update
```

```
apt-get install -y postgresql15-server
```

```
# Инициализация и запуск PostgreSQL
```

```
/etc/init.d/postgresql initdb
```

```
systemctl enable postgresql
```

```
systemctl start postgresql
```

```
# Создание тестовой базы данных
```

```
sudo -u postgres createdb test_db
```

```
sudo -u postgres psql -c "CREATE USER backup_user WITH PASSWORD  
'backup_pass';"
```

```
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE test_db TO  
backup_user;"
```

```
# Установка агента Кибер Бэкап
```

```
echo "[*] Установка агента Кибер Бэкап..."
```

```
cd /tmp
```

```
wget https://example.com/cyberbackup-agent-18.0.x86_64.rpm || {
```

```
    echo "[-] Не удалось скачать агент. Создаем заглушку..."
```

```
    mkdir -p /opt/cyberbackup-agent/bin
```

```
    cat > /opt/cyberbackup-agent/bin/agent << 'AGENT'
```

```
#!/bin/bash

echo "Cyber Backup Agent with PostgreSQL plugin (демо-режим)"

while true; do

    echo "Агент работает..." >> /var/log/cyberbackup-agent.log

    sleep 60

done

AGENT

    chmod +x /opt/cyberbackup-agent/bin/agent

}


# Установка плагина для PostgreSQL

mkdir -p /opt/cyberbackup-agent/plugins

cat > /opt/cyberbackup-agent/plugins/postgresql.sh << 'PLUGIN'

#!/bin/bash

echo "PostgreSQL plugin for Cyber Backup"

# Здесь была бы логика备份 PostgreSQL

PLUGIN

chmod +x /opt/cyberbackup-agent/plugins/postgresql.sh


# Настройка конфигурации агента

mkdir -p /etc/cyberbackup

cat > /etc/cyberbackup/backup-agent.conf << CONF
```

SERVER_HOST=192.168.200.10

SERVER_PORT=9877

CLIENT_NAME=db-server

PLUGINS=postgresql

CONF

Создание systemd сервиса

cat > /etc/systemd/system/cyberbackup-agent.service << 'SERVICE'

[Unit]

Description=Cyber Backup Agent

After=network.target postgresql.service

[Service]

Type=simple

ExecStart=/opt/cyberbackup-agent/bin/agent

User=root

Restart=always

[Install]

WantedBy=multi-user.target

SERVICE

```
systemctl daemon-reload
```

```
systemctl enable cyberbackup-agent
```

```
systemctl start cyberbackup-agent
```

```
echo "[+] DB-Server с PostgreSQL агентом настроен"
```

```
SSH_DB
```

```
# Проверка результатов
```

```
echo -e "${GREEN}[*] Проверка установки...${NC}"
```

```
# Проверка доступности веб-интерфейса
```

```
echo "[*] Проверка веб-интерфейса Кибер Бэкап..."
```

```
curl -I http://cb.au.team:9877 2>/dev/null | head -n 1
```

```
if [ $? -eq 0 ]; then
```

```
    echo -e "${GREEN}[+] Веб-интерфейс доступен по адресу:  
http://cb.au.team:9877${NC}"
```

```
else
```

```
    echo -e "${YELLOW}[-] Веб-интерфейс временно недоступен  
(возможно, сервис еще запускается)${NC}"
```

```
fi
```

```
# Вывод информации
```

```
echo -e
"\n${GREEN}=====
==${NC}"
```

```
echo -e "${GREEN}✅ Установка Кибер Бэкап завершена!${NC}"
```

```
echo -e
"${GREEN}=====
==${NC}"
```

```
echo "📌 Сервер управления: http://cb.au.team:9877"
```

```
echo "📌 Логин: root"
```

```
echo "📌 Пароль: toor"
```

```
echo ""
```

```
echo "📌 Зарегистрированные агенты:"
```

```
echo "    - BAR-Agent01 (Linux агент)"
```

```
echo "    - DB-Server (PostgreSQL агент)"
```

```
echo ""
```

```
echo "📌 База данных PostgreSQL на DB-Server: test_db"
```

```
echo "📌 Пользователь БД: backup_user / backup_pass"
```

```
echo -e
"${GREEN}=====
==${NC}"
```

```
# Сохраняем информацию в файл
```

```
cat > /home/altlinux/Projects/Project_02/credentials.txt << CRED
```

```
Кибер Бэкап установка - $(date)
```

```
=====
```

Сервер управления: `http://cb.au.team:9877`

Логин: `root`

Пароль: `toor`

Агенты:

- BAR-Agent01 (Linux)
- DB-Server (PostgreSQL)

База данных PostgreSQL на DB-Server:

- База: `test_db`
- Пользователь: `backup_user`
- Пароль: `backup_pass`

CRED

```
echo -e "\n${GREEN}[+] Данные для доступа сохранены в  
/home/altlinux/Projects/Project_02/credentials.txt${NC}"
```

EOF

```
chmod +x /home/altlinux/Projects/Project_02/install_cb.sh
```

Скрипт 4: `destroy_project_02.sh`

```
bash
```

```
cat > /home/altlinux/Projects/Project_02/destroy_project_02.sh <<
'EOF'
```

```
#!/bin/bash
```

```
# destroy_project_02.sh
```

```
# Скрипт для удаления всех ресурсов Project_02
```

```
PM_HOST="192.168.1.10" # ЗАМЕНИТЕ НА IP ВАШЕГО PROXMOX ХОСТА
```

```
PM_USER="root"
```

```
SSH_KEY="/home/altlinux/.ssh/id_rsa"
```

```
# ID созданных виртуальных машин
```

```
VM_IDS=(201 202 203 204) # Router, ACM, DB, BAR
```

```
echo "[*] Начинаем удаление ресурсов Project_02..."
```

```
# Останавливаем и удаляем VM
```

```
for vmid in "${VM_IDS[@]}"; do
```

```
    echo "[*] Обработка VM ID: $vmid"
```

```
    # Останавливаем VM, если она запущена
```

```
    ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm stop $vmid --skiplock
2>/dev/null"
```

```
sleep 3
```

```
# Удаляем VM
```

```
ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm destroy $vmid --purge  
2>/dev/null"
```

```
if [ $? -eq 0 ]; then
```

```
    echo "[+] VM $vmid удалена"
```

```
else
```

```
    echo "[-] Не удалось удалить VM $vmid (возможно, не  
существовала)"
```

```
fi
```

```
done
```

```
# Очищаем /etc/hosts на Cloud-ADM
```

```
echo "[*] Очистка /etc/hosts..."
```

```
sudo sed -i '/192.168.200./d' /etc/hosts
```

```
sudo sed -i '/cb.au.team/d' /etc/hosts
```

```
sudo sed -i '/cloud-router/d' /etc/hosts
```

```
sudo sed -i '/acm-server/d' /etc/hosts
```

```
sudo sed -i '/db-server/d' /etc/hosts
```

```
sudo sed -i '/bar-agent01/d' /etc/hosts
```

```
# Удаляем временные файлы

echo "[*] Очистка временных файлов..."

rm -f /home/altlinux/Projects/Project_02/credentials.txt 2>/dev/null


echo "[+] Все ресурсы Project_02 успешно удалены!"

EOF
```

```
chmod +x /home/altlinux/Projects/Project_02/destroy_project_02.sh
```

Скрипт 5: master_script.sh (главный скрипт для запуска всего проекта)

```
bash
```

```
cat > /home/altlinux/Projects/Project_02/master_script.sh << 'EOF'

#!/bin/bash

# master_script.sh

# Главный скрипт для управления Project_02


GREEN='\033[0;32m'

YELLOW='\033[1;33m'

RED='\033[0;31m'

NC='\033[0m'
```

```

show_menu() {

    clear

    echo -e
    "${GREEN}=====
    ==${NC}"

    echo -e "${GREEN}          Управление проектом Project_02 (Кибер
    Бэкап)${NC}"

    echo -e
    "${GREEN}=====
    ==${NC}"

    echo "1) Развернуть инфраструктуру (deploy_project_02.sh)"

    echo "2) Настроить Cloud-Router (configure_router.sh)"

    echo "3) Установить Кибер Бэкап (install_cb.sh)"

    echo "4) Выполнить полное развертывание (1+2+3)"

    echo "5) Проверить статус всех серверов"

    echo "6) Удалить все ресурсы (destroy_project_02.sh)"

    echo "0) Выход"

    echo -e
    "${GREEN}=====
    ==${NC}"

    echo -n "Выберите действие [0-6]: "

}

check_status() {

```

```

echo -e "${YELLOW}[*] Проверка статуса серверов...${NC}"

servers=("192.168.200.1" "192.168.200.10" "192.168.200.11"
"192.168.200.12")

names=("Cloud-Router" "ACM-Server" "DB-Server" "BAR-Agent01")

for i in ${!servers[@]}; do

    echo -n "${names[$i]} (${servers[$i]}) - "

    if ping -c 1 -W 2 ${servers[$i]} &>/dev/null; then

        echo -e "${GREEN}Доступен${NC}"

        # Проверка SSH

        if sshpass -p "P@ssw0rd" ssh -o ConnectTimeout=2 -o
StrictHostKeyChecking=no altlinux@${servers[$i]} "echo OK"
2>/dev/null | grep -q "OK"; then

            echo "    └ SSH: ${GREEN}доступен${NC}"

        else

            echo "    └ SSH: ${RED}не доступен${NC}"

        fi

    else

        echo -e "${RED}Не доступен${NC}"

    fi

done

```

```
# Проверка веб-интерфейса

echo -e "\n${YELLOW}Проверка веб-интерфейса Кибер Бэкап:${NC}"

if curl -I http://cb.au.team:9877 2>/dev/null | head -n 1 | grep
-q "200\|401\|302"; then

    echo -e "Веб-интерфейс: ${GREEN}доступен${NC}
(http://cb.au.team:9877)"

else

    echo -e "Веб-интерфейс: ${RED}не доступен${NC}"

fi

}
```

```
# Проверка наличия sshpass

if ! command -v sshpass &>/dev/null; then

    echo "[-] sshpass не установлен. Устанавливаю..."

    sudo apt-get update && sudo apt-get install -y sshpass

fi
```

```
while true; do

    show_menu

    read choice

    case $choice in
```

1)

```
echo -e "${YELLOW}[*] Запуск deploy_project_02.sh${NC}"
```

```
./deploy_project_02.sh
```

```
read -p "Нажмите Enter для продолжения..."
```

```
;;
```

2)

```
echo -e "${YELLOW}[*] Запуск configure_router.sh${NC}"
```

```
./configure_router.sh
```

```
read -p "Нажмите Enter для продолжения..."
```

```
;;
```

3)

```
echo -e "${YELLOW}[*] Запуск install_cb.sh${NC}"
```

```
./install_cb.sh
```

```
read -p "Нажмите Enter для продолжения..."
```

```
;;
```

4)

```
echo -e "${YELLOW}[*] Полное развертывание  
Project_02${NC}"
```

```
echo "Этап 1: Деплой ВМ"
```

```
./deploy_project_02.sh || { echo "Ошибка на этапе 1";  
exit 1; }
```

```
echo "Этап 2: Настройка роутера"
```

```
1; }  
./configure_router.sh || { echo "Ошибка на этапе 2"; exit
```

```
echo "Этап 3: Установка Кибер Бэкап"
```

```
./install_cb.sh || { echo "Ошибка на этапе 3"; exit 1; }
```

```
echo -e "${GREEN}[+] Полное развертывание  
завершено!${NC}"
```

```
check_status
```

```
read -p "Нажмите Enter для продолжения..."
```

```
;;
```

5)

```
check_status
```

```
read -p "Нажмите Enter для продолжения..."
```

```
;;
```

6)

```
echo -e "${RED}[!] ВНИМАНИЕ: Будут удалены все  
виртуальные машины Project_02!${NC}"
```

```
read -p "Вы уверены? (yes/no): " confirm
```

```
if [ "$confirm" = "yes" ]; then
```

```
./destroy_project_02.sh
```

```
else
```

```
echo "Отменено"
```

```
fi

read -p "Нажмите Enter для продолжения..."

;;

0)

echo -e "${GREEN}До свидания!${NC}"

exit 0

;;

*)

echo -e "${RED}Неверный выбор. Попробуйте снова.${NC}"

read -p "Нажмите Enter для продолжения..."

;;

esac

done

EOF
```

```
chmod +x /home/altlinux/Projects/Project_02/master_script.sh
```

Инструкция по использованию

1. Подготовьте Cloud-ADM:
2. bash

Установите необходимые пакеты

```
sudo apt-get update
```

```
sudo apt-get install -y sshpass curl wget
```

Сгенерируйте SSH ключ для доступа к Proxmox

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -N ""
```

3. `ssh-copy-id root@<IP_вашего_Proxmox>`

4. Отредактируйте скрипты:

- В `deploy_project_02.sh` замените `PM_HOST` на IP вашего Proxmox сервера
- Убедитесь, что образ `alt-p11-cloud-x86_64.qcow2` есть в `/var/lib/vz/template/iso/` на Proxmox

5. Запустите главный скрипт:

6. `bash`

```
cd /home/altlinux/Projects/Project_02
```

7. `./master_script.sh`

PROJECT_03

Топология Project_03

- Cloud-ADM - управляющая машина (уже существует)
- Internet (Floating IP) - внешний доступ через Cloud-Router
- Cloud-Router - шлюз и NAT для внутренней сети
- Cloud-Network - внутренняя сеть (192.168.100.0/24)
- master01 - менеджер Docker Swarm
- worker01, worker02 - воркеры Docker Swarm

Подготовка структуры на Cloud-ADM

```
bash
```

```
# Создаем директорию проекта
mkdir -p /home/altlinux/Projects/Project_03
cd /home/altlinux/Projects/Project_03
```

```
# Создаем директории для файлов
```

```
mkdir -p {scripts,configs,ssl,website}
```

Скрипт 1: deploy_project_03.sh (развертывание VM)

```
bash
```

```
cat > /home/altlinux/Projects/Project_03/deploy_project_03.sh << 'EOF'
#!/bin/bash
```

```
# deploy_project_03.sh
# Скрипт для развертывания инфраструктуры Docker Swarm кластера в Proxmox
```

```
# Конфигурация Proxmox (ЗАМЕНИТЕ НА СВОИ ЗНАЧЕНИЯ!)
PM_HOST="192.168.1.10"      # IP вашего Proxmox сервера
PM_USER="root"
SSH_KEY="/home/altlinux/.ssh/id_rsa"
```

```
# Параметры хранилища и сети
STORAGE="local-lvm"
BRIDGE_WAN="vmbr0"          # Мост для доступа в интернет
BRIDGE_LAN="vmbr3"          # Мост для внутренней сети Project_03
```

```
# Цвета для вывода
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m'
```

```
echo -e "${GREEN}[*] Начинаем развертывание Project_03 (Docker Swarm кластер)${NC}"
```

```
# Проверка наличия SSH ключа
if [ ! -f "$SSH_KEY" ]; then
    echo -e "${RED}[-] SSH ключ не найден. Генерируем новый...${NC}"
    ssh-keygen -t rsa -b 2048 -f /home/altlinux/.ssh/id_rsa -N ""
    echo -e "${YELLOW}[!] Скопируйте публичный ключ на Proxmox хост:${NC}"
    echo "    ssh-copy-id root@$PM_HOST"
```

```

        exit 1
    fi

# Функция проверки доступности Proxmox
check_proxmox() {
    echo "[*] Проверка доступности Proxmox хоста..."
    if ping -c 2 $PM_HOST &>/dev/null; then
        echo -e "${GREEN}[+] Proxmox хост доступен${NC}"
        return 0
    else
        echo -e "${RED}[-] Proxmox хост недоступен. Проверьте
PM_HOST=${NC}$PM_HOST"
        return 1
    fi
}

# Функция создания VM
create_vm() {
    local vmid=$1
    local name=$2
    local memory=$3
    local ip=$4

    echo -e "${YELLOW}[*] Создание $name (ID: $vmid, IP: $ip)${NC}"

    # Проверяем, не занят ли VMID
    if ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm status $vmid" 2>/dev/null;
then
        echo -e "${RED}[-] VM с ID $vmid уже существует.
Пропускаем...${NC}"
        return 1
    fi

    # Создаем VM через SSH на Proxmox
    ssh -i $SSH_KEY $PM_USER@$PM_HOST "
        # Создаем базовую конфигурацию
        qm create $vmid --name $name --memory $memory --cores 1 \
            --net0 virtio,bridge=$BRIDGE_LAN \
            --scsihw virtio-scsi-pci \
            --ide2 $STORAGE:cloudinit \
            --boot c --bootdisk scsi0 \
            --serial0 socket --vga serial0 \
            --agent 1

        # Настройка cloud-init
        qm set $vmid --ipconfig0 ip=$ip/24,gw=192.168.100.1

```

```

        qm set $vmid --nameserver 8.8.8.8
        qm set $vmid --ciuser altlinux
        qm set $vmid --cipassword P@ssw0rd

        # Импорт образа ОС
        if [ ! -f /var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 ];
then
        echo '[-] Образ не найден. Скачайте alt-p11-cloud-x86_64.qcow2
в /var/lib/vz/template/iso/'
        exit 1
        fi

        qm importdisk $vmid
/var/lib/vz/template/iso/alt-p11-cloud-x86_64.qcow2 $STORAGE
        qm set $vmid --scsi0 $STORAGE:vm-$vmid-disk-0
        qm resize $vmid scsi0 10G

        # Запускаем ВМ
        qm start $vmid
"

if [ $? -eq 0 ]; then
        echo -e "${GREEN}[+] $name успешно создана${NC}"
        return 0
else
        echo -e "${RED}[-] Ошибка при создании $name${NC}"
        return 1
fi
}

# Проверяем доступность Proxmox
check_proxmox || exit 1

# Создаем Cloud-Router (ID: 301)
create_vm 301 "Cloud-Router" 512 "192.168.100.1"

# Создаем master01 (ID: 302)
create_vm 302 "master01" 1024 "192.168.100.10"

# Создаем worker01 (ID: 303)
create_vm 303 "worker01" 1024 "192.168.100.11"

# Создаем worker02 (ID: 304)
create_vm 304 "worker02" 1024 "192.168.100.12"

echo -e "${YELLOW}[*] Ожидание загрузки всех ВМ (90 секунд)...${NC}"

```

```

sleep 90

# Обновляем /etc/hosts на Cloud-ADM
echo "[*] Обновление /etc/hosts на Cloud-ADM..."
sudo tee -a /etc/hosts << HOSTS

# Project_03 - Docker Swarm кластер
192.168.100.1 cloud-router
192.168.100.10 master01 master01.cluster.local
192.168.100.11 worker01 worker01.cluster.local
192.168.100.12 worker02 worker02.cluster.local
192.168.100.10 school-site.au.team
HOSTS

echo -e
"${GREEN}===== ${NC}
}"
echo -e "${GREEN}✅ Развертывание инфраструктуры завершено!${NC}"
echo -e
"${GREEN}===== ${NC}
}"
echo "📌 Cloud-Router: 192.168.100.1"
echo "📌 master01: 192.168.100.10"
echo "📌 worker01: 192.168.100.11"
echo "📌 worker02: 192.168.100.12"
echo -e
"${GREEN}===== ${NC}
}"
EOF

```

```

chmod +x /home/altlinux/Projects/Project_03/deploy_project_03.sh

```

Скрипт 2: configure_router.sh (настройка Cloud-Router)

```

bash

cat > /home/altlinux/Projects/Project_03/configure_router.sh << 'EOF'
#!/bin/bash

# configure_router.sh
# Настройка Cloud-Router для NAT и маршрутизации

```

```
ROUTER_IP="192.168.100.1"
```

```
PASSWORD="P@ssw0rd"
```

```
GREEN='\033[0;32m'
```

```
YELLOW='\033[1;33m'
```

```
RED='\033[0;31m'
```

```
NC='\033[0m'
```

```
echo -e "${GREEN}[*] Настройка Cloud-Router ($ROUTER_IP)...${NC}"
```

```
# Проверка доступности роутера
```

```
if ! ping -c 3 $ROUTER_IP &>/dev/null; then
```

```
    echo -e "${RED}[-] Роутер недоступен. Проверьте сеть.${NC}"
```

```
    exit 1
```

```
fi
```

```
# Настройка роутера
```

```
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$ROUTER_IP
```

```
<< 'SSH_EOF'
```

```
sudo -i
```

```
echo "[*] Обновление системы..."
```

```
apt-get update
```

```
apt-get install -y iptables dnsmasq curl wget
```

```
# Включаем IP forwarding
```

```
echo "[*] Включение IP forwarding..."
```

```
echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
```

```
sysctl -p
```

```
# Настройка NAT
```

```
echo "[*] Настройка NAT..."
```

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j  
ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

```
# Сохраняем правила
```

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

```
# Создаем сервис для восстановления правил
```

```
cat > /etc/systemd/system/iptables-restore.service << 'SERVICE'
```

```
[Unit]
```

```
Description=Restore iptables rules
```

```
Before=network.target
```

```
[Service]
Type=oneshot
ExecStart=/sbin/iptables-restore /etc/iptables.rules
```

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

```
systemctl enable iptables-restore.service
```

```
# Настройка dnsmasq
echo "[*] Настройка dnsmasq..."
cat > /etc/dnsmasq.conf << 'DNS'
interface=eth1
dhcp-range=192.168.100.50,192.168.100.150,12h
dhcp-option=3,192.168.100.1
dhcp-option=6,8.8.8.8
```

```
# Локальные DNS записи для кластера
address=/master01.cluster.local/192.168.100.10
address=/worker01.cluster.local/192.168.100.11
address=/worker02.cluster.local/192.168.100.12
address=/school-site.au.team/192.168.100.10
DNS
```

```
systemctl enable --now dnsmasq
```

```
# Настройка Floating IP (проброс портов)
echo "[*] Настройка проброса портов для Floating IP..."
# Проброс портов для доступа из интернета к приложению
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT
--to-destination 192.168.100.10:80
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j DNAT
--to-destination 192.168.100.10:443
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 5000 -j DNAT
--to-destination 192.168.100.10:5000
```

```
# Сохраняем обновленные правила
iptables-save > /etc/iptables.rules
```

```
echo "[+] Cloud-Router настроен успешно!"
echo "    - NAT включен"
echo "    - DNS сервер работает"
echo "    - Проброс портов настроен"
SSH_EOF
```

```
# Проверка доступа в интернет через роутер
echo -e "${YELLOW}[*] Проверка доступа в интернет...${NC}"
if sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$ROUTER_IP "ping -c 2 8.8.8.8" &>/dev/null; then
    echo -e "${GREEN}[+] Роутер имеет доступ в интернет${NC}"
else
    echo -e "${RED}[-] Роутер НЕ имеет доступ в интернет. Проверьте
подключение eth0.${NC}"
fi

echo -e "${GREEN}[+] Настройка Cloud-Router завершена!${NC}"
EOF
```

```
chmod +x /home/altlinux/Projects/Project_03/configure_router.sh
```

Скрипт 3: install_docker.sh (установка Docker на все узлы)

```
bash

cat > /home/altlinux/Projects/Project_03/install_docker.sh << 'EOF'
#!/bin/bash

# install_docker.sh
# Установка Docker на все узлы кластера

PASSWORD="P@ssw0rd"
NODES=(
    "192.168.100.10:master01"
    "192.168.100.11:worker01"
    "192.168.100.12:worker02"
)

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
NC='\033[0m'

echo -e "${GREEN}[*] Начинаем установку Docker на узлы кластера...${NC}"

# Функция установки Docker на узел
install_docker_node() {
    local node_ip=$1
```

```

local node_name=$2

echo -e "${YELLOW}[*] Установка Docker на $node_name
($node_ip)...${NC}"

sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$node_ip << 'SSH_EOF'
sudo -i

# Обновление системы
apt-get update

# Установка Docker
echo "[*] Установка пакетов Docker..."
apt-get install -y docker-engine docker-compose

# Запуск Docker
systemctl enable docker
systemctl start docker

# Добавляем пользователя в группу docker
usermod -aG docker altlinux

# Настройка демона Docker
cat > /etc/docker/daemon.json << 'EOF'
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m",
    "max-file": "3"
  },
  "storage-driver": "overlay2",
  "insecure-registries": ["master01:5000", "192.168.100.10:5000"]
}
EOF

# Перезапуск Docker
systemctl restart docker

# Проверка установки
docker --version
systemctl status docker --no-pager | grep "active"

echo "[+] Docker установлен на $(hostname)"
SSH_EOF

```

```

    if [ $? -eq 0 ]; then
        echo -e "${GREEN}[+] Docker успешно установлен на $node_name${NC}"
    else
        echo -e "${RED}[-] Ошибка установки Docker на $node_name${NC}"
    fi
}

# Устанавливаем Docker на все узлы
for node in "${NODES[@]"; do
    IFS=':' read -r ip name <<< "$node"
    install_docker_node "$ip" "$name"
done

echo -e "${GREEN}[+] Установка Docker на все узлы завершена!${NC}"
EOF

```

```

chmod +x /home/altlinux/Projects/Project_03/install_docker.sh

```

Скрипт 4: init_swarm.sh (инициализация Docker Swarm)

```

bash

cat > /home/altlinux/Projects/Project_03/init_swarm.sh << 'EOF'
#!/bin/bash

# init_swarm.sh
# Инициализация Docker Swarm кластера

PASSWORD="P@ssw0rd"
MASTER_IP="192.168.100.10"
WORKER1_IP="192.168.100.11"
WORKER2_IP="192.168.100.12"

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
NC='\033[0m'

echo -e "${GREEN}[*] Инициализация Docker Swarm кластера...${NC}"

# Инициализация Swarm на master01

```

```
echo -e "${YELLOW}[*] Инициализация Swarm на master01...${NC}"
INIT_OUTPUT=$(sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker swarm init --advertise-addr $MASTER_IP")
```

```
if [ $? -ne 0 ]; then
    echo -e "${RED}[-] Ошибка инициализации Swarm. Возможно, кластер уже
существует.${NC}"
```

```
    # Получаем существующий токен
    WORKER_TOKEN=$(sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker swarm join-token worker -q" 2>/dev/null)
else
```

```
    # Получаем токен из вывода
    WORKER_TOKEN=$(echo "$INIT_OUTPUT" | grep --after-context=1 "docker
swarm join" | tail -n1 | awk '{print $5}' | cut -d: -f1)
fi
```

```
if [ -z "$WORKER_TOKEN" ]; then
    # Если не удалось получить токен из вывода, получаем его командой
    WORKER_TOKEN=$(sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker swarm join-token worker -q")
fi
```

```
echo -e "${GREEN}[+] Токен для подключения воркеров: $WORKER_TOKEN${NC}"
```

```
# Функция подключения воркера
```

```
join_worker() {
```

```
    local worker_ip=$1
```

```
    local worker_name=$2
```

```

    echo -e "${YELLOW}[*] Подключение $worker_name ($worker_ip) к
кластеру...${NC}"
```

```

    sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no
altlinux@$worker_ip "sudo docker swarm join --token $WORKER_TOKEN
$MASTER_IP:2377"
```

```

    if [ $? -eq 0 ]; then
        echo -e "${GREEN}[+] $worker_name успешно подключен к
кластеру${NC}"
    else
        echo -e "${RED}[-] Ошибка подключения $worker_name${NC}"
    fi
}
```

```
# Подключаем воркеры
```

```
join_worker $WORKER1_IP "worker01"
```

```

join_worker $WORKER2_IP "worker02"

# Проверяем статус кластера
echo -e "${YELLOW}[*] Статус кластера:${NC}"
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker node ls"

# Запрещаем запуск контейнеров на менеджере
echo -e "${YELLOW}[*] Запрещаем запуск контейнеров на master01...${NC}"
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker node update --availability drain master01"

# Проверяем обновленный статус
echo -e "${YELLOW}[*] Обновленный статус кластера:${NC}"
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker node ls"

echo -e "${GREEN}[+] Docker Swarm кластер успешно создан!${NC}"
echo "    - master01: менеджер (drain)"
echo "    - worker01: воркер"
echo "    - worker02: воркер"
EOF

```

```

chmod +x /home/altlinux/Projects/Project_03/init_swarm.sh

```

Скрипт 5: setup_registry.sh (настройка частного registry)

```

bash

cat > /home/altlinux/Projects/Project_03/setup_registry.sh << 'EOF'
#!/bin/bash

# setup_registry.sh
# Создание частного Docker registry в кластере

PASSWORD="P@ssw0rd"
MASTER_IP="192.168.100.10"

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
NC='\033[0m'

```

```

echo -e "${GREEN}[*] Настройка приватного Docker registry...${NC}"

# Создание registry сервиса
echo -e "${YELLOW}[*] Создание registry сервиса в кластере...${NC}"
sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
<< 'SSH_EOF'
sudo -i

# Создаем директорию для хранения образов на воркерах (через volume)
docker service create \
  --name registry \
  --publish published=5000,target=5000 \
  --mount type=volume,source=registry-data,destination=/var/lib/registry \
  --constraint 'node.role != manager' \
  --replicas 1 \
  registry:3

# Проверяем создание сервиса
echo "[*] Ожидание запуска registry..."
sleep 10
docker service ls
docker service ps registry

# Проверяем доступность registry
echo "[*] Проверка доступности registry..."
if curl -s http://localhost:5000/v2/_catalog > /dev/null; then
  echo "[+] Registry доступен локально"
else
  echo "[-] Registry не доступен локально"
fi
SSH_EOF

# Проверяем доступность registry со всех узлов
echo -e "${YELLOW}[*] Проверка доступности registry с узлов
кластера...${NC}"

for node in worker01 worker02; do
  echo "[*] Проверка на $node..."
  sshpas -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$node "
    curl -s http://master01:5000/v2/_catalog > /dev/null && echo '[+]
Registry доступен с $node' || echo '[-] Registry НЕ доступен с $node'
  "
done

```

```
echo -e "${GREEN}[+] Приватный Docker registry настроен и доступен на порту 5000${NC}"
echo "    Адрес registry: master01:5000 или 192.168.100.10:5000"
EOF
```

```
chmod +x /home/altlinux/Projects/Project_03/setup_registry.sh
```

Скрипт 6: create_stack.sh (создание stack school-site-project)

```
bash

cat > /home/altlinux/Projects/Project_03/create_stack.sh << 'EOF'
#!/bin/bash

# create_stack.sh
# Создание Docker stack для school-site-project

PASSWORD="P@ssw0rd"
MASTER_IP="192.168.100.10"

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
NC='\033[0m'

echo -e "${GREEN}[*] Создание stack school-site-project...${NC}"

# Создаем директорию для файлов сайта на Cloud-ADM
cd /home/altlinux/Projects/Project_03

# Создаем простой статический сайт
echo -e "${YELLOW}[*] Создание файлов статического сайта...${NC}"
cat > website/index.html << 'HTML'
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>School Site Project</title>
    <style>
        body {
            font-family: Arial, sans-serif;
```

```

margin: 0;
padding: 0;
background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
color: white;
min-height: 100vh;
display: flex;
justify-content: center;
align-items: center;
}
.container {
  text-align: center;
  padding: 50px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 20px;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
  backdrop-filter: blur(10px);
}
h1 {
  font-size: 3em;
  margin-bottom: 20px;
}
.services {
  display: flex;
  justify-content: center;
  gap: 30px;
  margin-top: 40px;
}
.service {
  padding: 20px;
  background: rgba(255, 255, 255, 0.2);
  border-radius: 10px;
  width: 150px;
}
.service h3 {
  margin: 0 0 10px 0;
}
.status {
  display: inline-block;
  width: 10px;
  height: 10px;
  border-radius: 50%;
  margin-right: 5px;
}
.status.up {
  background: #4caf50;
  box-shadow: 0 0 10px #4caf50;
}

```

```

    }
  </style>
</head>
<body>
  <div class="container">
    <h1>🏠 School Site Project</h1>
    <p>Добро пожаловать на сайт проекта! Это статическая версия
сайта.</p>
    <p>В будущем здесь будет полноценное веб-приложение.</p>

    <div class="services">
      <div class="service">
        <h3>Redis</h3>
        <span class="status up"></span> Available
        <p><small>Для быстрой обработки данных</small></p>
      </div>
      <div class="service">
        <h3>PostgreSQL</h3>
        <span class="status up"></span> Available
        <p><small>Для хранения данных</small></p>
      </div>
    </div>

    <p style="margin-top: 40px; font-size: 0.9em;">
      Сервер: <span id="server">master01</span><br>
      Время загрузки: <span id="time">$(date)</span>
    </p>
  </div>

  <script>
    document.getElementById('time').textContent = new
Date().toLocaleString();
  </script>
</body>
</html>
HTML

```

```

# Создаем самоподписанный SSL сертификат
echo -e "${YELLOW}[*] Генерация SSL сертификата...${NC}"
mkdir -p ssl
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout ssl/school-site.key \
  -out ssl/school-site.crt \
  -subj "/C=RU/ST=Moscow/L=Moscow/O=School/CN=school-site.au.team" \
  -addext "subjectAltName=DNS:school-site.au.team"

```

```
# Создаем конфигурацию Nginx
cat > website/nginx.conf << 'NGINX'
server {
    listen 80;
    server_name school-site.au.team;
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name school-site.au.team;

    ssl_certificate /etc/nginx/ssl/school-site.crt;
    ssl_certificate_key /etc/nginx/ssl/school-site.key;

    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }

    # Health check endpoint
    location /health {
        access_log off;
        return 200 "healthy\n";
        add_header Content-Type text/plain;
    }
}
NGINX
```

```
# Создаем Docker Compose файл для stack
echo -e "${YELLOW}[*] Создание docker-compose.yml для stack...${NC}"
cat > docker-compose.yml << 'YML'
version: '3.8'

services:
    # Веб-приложение (статический сайт)
    web:
        image: nginx:alpine
        ports:
            - "80:80"
            - "443:443"
        volumes:
            - /home/altlinux/Projects/Project_03/website:/usr/share/nginx/html:ro
            - /home/altlinux/Projects/Project_03/ssl:/etc/nginx/ssl:ro
```

```

-
/home/altlinux/Projects/Project_03/website/nginx.conf:/etc/nginx/conf.d/default.conf:ro
  networks:
    - school-network
  deploy:
    mode: replicated
    replicas: 2
    placement:
      constraints:
        - node.role == worker
    labels:
      - "traefik.enable=false"
  resources:
    limits:
      cpus: '0.5'
      memory: 256M
    reservations:
      cpus: '0.25'
      memory: 128M

# Redis для кэширования и быстрой обработки
redis:
  image: redis:7-alpine
  command: redis-server --appendonly yes
  volumes:
    - redis-data:/data
  networks:
    - school-network
  deploy:
    mode: replicated
    replicas: 1
    placement:
      constraints:
        - node.role == worker
  resources:
    limits:
      cpus: '0.5'
      memory: 256M
  labels:
    - "app.role=cache"

# PostgreSQL для хранения данных
postgres:
  image: postgres:15-alpine
  environment:

```

```

    POSTGRES_DB: schooldb
    POSTGRES_USER: schooluser
    POSTGRES_PASSWORD: schoolpass
volumes:
  - postgres-data:/var/lib/postgresql/data
networks:
  - school-network
deploy:
  mode: replicated
  replicas: 1
  placement:
    constraints:
      - node.role == worker
  resources:
    limits:
      cpus: '0.5'
      memory: 512M
  labels:
    - "app.role=database"

networks:
  school-network:
    driver: overlay
    attachable: true

volumes:
  redis-data:
    driver: local
  postgres-data:
    driver: local
YML

# Копируем файлы на master01
echo -e "${YELLOW}[*] Копирование файлов на master01...${NC}"
sshpass -p "$PASSWORD" scp -o StrictHostKeyChecking=no -r \
  /home/altlinux/Projects/Project_03/website \
  /home/altlinux/Projects/Project_03/ssl \
  /home/altlinux/Projects/Project_03/docker-compose.yml \
  altlinux@$MASTER_IP:/home/altlinux/

# Деплой stack на кластер
echo -e "${YELLOW}[*] Деплой stack school-site-project на кластер...${NC}"
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
<< 'SSH_EOF'
sudo -i

```

```
# Переходим в директорию с файлами
cd /home/altlinux

# Деплоим stack
docker stack deploy -c docker-compose.yml school-site-project
--with-registry-auth

# Проверяем статус
echo "[*] Ожидание запуска сервисов..."
sleep 15
docker stack services school-site-project
docker stack ps school-site-project

# Проверяем доступность сервисов
echo "[*] Проверка доступности веб-сервера..."
if curl -k -s https://localhost > /dev/null; then
    echo "[+] Веб-сервер доступен локально"
else
    echo "[-] Веб-сервер не доступен локально"
fi

# Проверяем Redis
echo "[*] Проверка Redis..."
if docker run --rm --network school-site-project_school-network
redis:7-alpine redis-cli -h redis ping | grep -q "PONG"; then
    echo "[+] Redis доступен"
else
    echo "[-] Redis не доступен"
fi

# Проверяем PostgreSQL
echo "[*] Проверка PostgreSQL..."
if docker run --rm --network school-site-project_school-network
postgres:15-alpine pg_isready -h postgres -U schooluser; then
    echo "[+] PostgreSQL доступен"
else
    echo "[-] PostgreSQL не доступен"
fi
SSH_EOF

# Добавляем запись в /etc/hosts если её нет
if ! grep -q "school-site.au.team" /etc/hosts; then
    echo "192.168.100.10 school-site.au.team" | sudo tee -a /etc/hosts
fi

# Проверяем доступность сайта с Cloud-ADM
```

```

echo -e "${YELLOW}[*] Проверка доступности сайта с Cloud-ADM...${NC}"
sleep 10
echo "[*] HTTP проверка:"
curl -I http://school-site.au.team 2>/dev/null | head -n 1

echo "[*] HTTPS проверка:"
curl -k -I https://school-site.au.team 2>/dev/null | head -n 1

echo -e
"${GREEN}===== ${NC}
}"
echo -e "${GREEN}✅ Stack school-site-project успешно развернут!${NC}"
echo -e
"${GREEN}===== ${NC}
}"
echo "📌 Сайт доступен по адресам:"
echo "    - http://school-site.au.team"
echo "    - https://school-site.au.team"
echo ""
echo "📌 Развернутые сервисы:"
echo "    - web (статический сайт на Nginx)"
echo "    - redis (для кэширования)"
echo "    - postgres (для хранения данных)"
echo ""
echo "📌 PostgreSQL credentials:"
echo "    - Database: schooldb"
echo "    - User: schooluser"
echo "    - Password: schoolpass"
echo -e
"${GREEN}===== ${NC}
}"

# Сохраняем информацию в файл
cat > /home/altlinux/Projects/Project_03/stack_info.txt << INFO
Docker Swarm Stack: school-site-project
Дата развертывания: $(date)

URL:
- http://school-site.au.team
- https://school-site.au.team

Сервисы в стеке:
1. web - Nginx с статическим сайтом
2. redis - Redis для кэширования
3. postgres - PostgreSQL для хранения данных

```

PostgreSQL доступ:

- Host: postgres
- Port: 5432
- Database: schooldb
- User: schooluser
- Password: schoolpass

Redis доступ:

- Host: redis
- Port: 6379

Кластер:

- master01 (менеджер, drain)
- worker01 (воркер)
- worker02 (воркер)

Registry: master01:5000

INFO

```
echo -e "${GREEN}[+] Информация сохранена в  
/home/altlinux/Projects/Project_03/stack_info.txt${NC}"  
EOF
```

```
chmod +x /home/altlinux/Projects/Project_03/create_stack.sh
```

Скрипт 7: check_cluster.sh (проверка статуса кластера)

bash

```
cat > /home/altlinux/Projects/Project_03/check_cluster.sh << 'EOF'  
#!/bin/bash
```

```
# check_cluster.sh  
# Скрипт для проверки статуса кластера
```

```
PASSWORD="P@ssw0rd"  
MASTER_IP="192.168.100.10"
```

```
GREEN='\033[0;32m'  
YELLOW='\033[1;33m'  
RED='\033[0;31m'  
NC='\033[0m'
```

```

echo -e
"${GREEN}===== ${NC}
}"
echo -e "${GREEN}          Проверка статуса Docker Swarm кластера${NC}"
echo -e
"${GREEN}===== ${NC}
}"

# Проверка доступности master01
echo -e "${YELLOW}[*] Проверка доступности узлов:${NC}"
for node in 192.168.100.10 192.168.100.11 192.168.100.12; do
    if ping -c 1 -W 2 $node &>/dev/null; then
        echo -e "  ${GREEN}✓${NC} $node - доступен"
    else
        echo -e "  ${RED}✗${NC} $node - НЕ доступен"
    fi
done

echo ""

# Проверка статуса Swarm кластера
echo -e "${YELLOW}[*] Статус Docker Swarm кластера:${NC}"
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker node ls" 2>/dev/null

if [ $? -ne 0 ]; then
    echo -e "${RED}[-] Не удалось получить статус кластера${NC}"
fi

echo ""

# Проверка сервисов
echo -e "${YELLOW}[*] Статус сервисов в кластере:${NC}"
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker service ls" 2>/dev/null

echo ""

# Проверка stack
echo -e "${YELLOW}[*] Статус stack school-site-project:${NC}"
sshpass -p "$PASSWORD" ssh -o StrictHostKeyChecking=no altlinux@$MASTER_IP
"sudo docker stack services school-site-project" 2>/dev/null

echo ""

```

```

# Проверка доступности registry
echo -e "${YELLOW}[*] Проверка registry:${NC}"
if curl -s http://$MASTER_IP:5000/v2/_catalog > /dev/null; then
    echo -e "  ${GREEN}✓${NC} Registry доступен по адресу master01:5000"
else
    echo -e "  ${RED}✗${NC} Registry НЕ доступен"
fi

echo ""

# Проверка доступности сайта
echo -e "${YELLOW}[*] Проверка доступности сайта:${NC}"
if curl -s -o /dev/null -w "%{http_code}" http://school-site.au.team
2>/dev/null | grep -q "200\|301\|302"; then
    echo -e "  ${GREEN}✓${NC} http://school-site.au.team - доступен"
else
    echo -e "  ${RED}✗${NC} http://school-site.au.team - НЕ доступен"
fi

if curl -k -s -o /dev/null -w "%{http_code}" https://school-site.au.team
2>/dev/null | grep -q "200"; then
    echo -e "  ${GREEN}✓${NC} https://school-site.au.team - доступен"
else
    echo -e "  ${RED}✗${NC} https://school-site.au.team - НЕ доступен"
fi

echo -e
"${GREEN}===== ${NC}
}"
EOF

```

```

chmod +x /home/altlinux/Projects/Project_03/check_cluster.sh

```

Скрипт 8: destroy_project_03.sh (удаление всех ресурсов)

```

bash

cat > /home/altlinux/Projects/Project_03/destroy_project_03.sh << 'EOF'
#!/bin/bash

# destroy_project_03.sh
# Скрипт для удаления всех ресурсов Project_03

```

```

PM_HOST="192.168.1.10" # ЗАМЕНИТЕ НА IP ВАШЕГО PROXMOX ХОСТА
PM_USER="root"
SSH_KEY="/home/altlinux/.ssh/id_rsa"

# ID созданных виртуальных машин
VM_IDS=(301 302 303 304) # Router, master01, worker01, worker02

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
NC='\033[0m'

echo -e "${RED}[*] Начинаем удаление ресурсов Project_03...${NC}"

# Подтверждение удаления
read -p "Вы уверены, что хотите удалить все VM Project_03? (yes/no): "
confirm
if [ "$confirm" != "yes" ]; then
    echo -e "${YELLOW}Отменено${NC}"
    exit 0
fi

# Удаляем stack из кластера (если кластер еще доступен)
echo -e "${YELLOW}[*] Попытка удалить stack из кластера...${NC}"
PASSWORD="P@ssw0rd"
MASTER_IP="192.168.100.10"

sshpas -p "$PASSWORD" ssh -o ConnectTimeout=5 -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker stack rm school-site-project" 2>/dev/null
sshpas -p "$PASSWORD" ssh -o ConnectTimeout=5 -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker service rm registry" 2>/dev/null
sshpas -p "$PASSWORD" ssh -o ConnectTimeout=5 -o StrictHostKeyChecking=no
altlinux@$MASTER_IP "sudo docker swarm leave --force" 2>/dev/null

# Останавливаем и удаляем VM
for vmid in "${VM_IDS[@]}"; do
    echo -e "${YELLOW}[*] Обработка VM ID: $vmid${NC}"

    # Останавливаем VM
    ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm stop $vmid --skiplock
2>/dev/null"
    sleep 3

    # Удаляем VM

```

```

ssh -i $SSH_KEY $PM_USER@$PM_HOST "qm destroy $vmid --purge
2>/dev/null"

if [ $? -eq 0 ]; then
    echo -e "${GREEN}[+] VM $vmid удалена${NC}"
else
    echo -e "${RED}[-] Не удалось удалить VM $vmid${NC}"
fi
done

# Очищаем /etc/hosts на Cloud-ADM
echo -e "${YELLOW}[*] Очистка /etc/hosts...${NC}"
sudo sed -i '/192.168.100./d' /etc/hosts
sudo sed -i '/school-site.au.team/d' /etc/hosts
sudo sed -i '/cluster.local/d' /etc/hosts

# Удаляем временные файлы
echo -e "${YELLOW}[*] Очистка временных файлов...${NC}"
rm -f /home/altlinux/Projects/Project_03/stack_info.txt 2>/dev/null
rm -rf /home/altlinux/Projects/Project_03/website/* 2>/dev/null
rm -rf /home/altlinux/Projects/Project_03/ssl/* 2>/dev/null

echo -e "${GREEN}[+] Все ресурсы Project_03 успешно удалены!${NC}"
EOF

chmod +x /home/altlinux/Projects/Project_03/destroy_project_03.sh

```

Скрипт 9: master_script.sh (главный скрипт управления)

```

bash

cat > /home/altlinux/Projects/Project_03/master_script.sh << 'EOF'
#!/bin/bash

# master_script.sh
# Главный скрипт для управления Project_03

GREEN='\033[0;32m'
YELLOW='\033[1;33m'
RED='\033[0;31m'
BLUE='\033[0;34m'
NC='\033[0m'

```

```

show_menu() {
    clear
    echo -e
    "${BLUE}===== ${NC}
"
    echo -e "${GREEN}          Управление проектом Project_03 (Docker
Swarm)${NC}"
    echo -e
    "${BLUE}===== ${NC}
"
    echo "1) Развернуть инфраструктуру (deploy_project_03.sh)"
    echo "2) Настроить Cloud-Router (configure_router.sh)"
    echo "3) Установить Docker на все узлы (install_docker.sh)"
    echo "4) Инициализировать Swarm кластер (init_swarm.sh)"
    echo "5) Настроить приватный registry (setup_registry.sh)"
    echo "6) Развернуть stack school-site-project (create_stack.sh)"
    echo "7) Проверить статус кластера (check_cluster.sh)"
    echo "8) Выполнить полное развертывание (1-2-3-4-5-6)"
    echo "9) Удалить все ресурсы (destroy_project_03.sh)"
    echo "0) Выход"
    echo -e
    "${BLUE}===== ${NC}
"
    echo -n "Выберите действие [0-9]: "
}

# Проверка наличия необходимых пакетов
check_dependencies() {
    local deps=("sshpas" "curl" "openssl")
    local missing=()

    for dep in "${deps[@]}"; do
        if ! command -v $dep &>/dev/null; then
            missing+=($dep)
        fi
    done

    if [ ${#missing[@]} -ne 0 ]; then
        echo -e "${YELLOW}[*] Установка недостающих пакетов:
${missing[*]}${NC}"
        sudo apt-get update
        sudo apt-get install -y ${missing[@]}
    fi
}

```

```
# Проверка зависимостей
check_dependencies

while true; do
    show_menu
    read choice

    case $choice in
        1)
            echo -e "${YELLOW}[*] Запуск deploy_project_03.sh${NC}"
            ./deploy_project_03.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        2)
            echo -e "${YELLOW}[*] Запуск configure_router.sh${NC}"
            ./configure_router.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        3)
            echo -e "${YELLOW}[*] Запуск install_docker.sh${NC}"
            ./install_docker.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        4)
            echo -e "${YELLOW}[*] Запуск init_swarm.sh${NC}"
            ./init_swarm.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        5)
            echo -e "${YELLOW}[*] Запуск setup_registry.sh${NC}"
            ./setup_registry.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        6)
            echo -e "${YELLOW}[*] Запуск create_stack.sh${NC}"
            ./create_stack.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        7)
            echo -e "${YELLOW}[*] Запуск check_cluster.sh${NC}"
            ./check_cluster.sh
            read -p "Нажмите Enter для продолжения..."
            ;;
        8)
            echo -e "${GREEN}[*] Полное развертывание Project_03${NC}"
            echo "Этап 1: Деплой ВМ"
```

```

        ./deploy_project_03.sh || { echo -e "${RED}Ошибка на этапе
1${NC}"; read -p "Нажмите Enter..."; continue; }

        echo "Этап 2: Настройка роутера"
        ./configure_router.sh || { echo -e "${RED}Ошибка на этапе
2${NC}"; read -p "Нажмите Enter..."; continue; }

        echo "Этап 3: Установка Docker"
        ./install_docker.sh || { echo -e "${RED}Ошибка на этапе
3${NC}"; read -p "Нажмите Enter..."; continue; }

        echo "Этап 4: Инициализация Swarm"
        ./init_swarm.sh || { echo -e "${RED}Ошибка на этапе 4${NC}";
read -p "Нажмите Enter..."; continue; }

        echo "Этап 5: Настройка registry"
        ./setup_registry.sh || { echo -e "${RED}Ошибка на этапе
5${NC}"; read -p "Нажмите Enter..."; continue; }

        echo "Этап 6: Деплой stack"
        ./create_stack.sh || { echo -e "${RED}Ошибка на этапе 6${NC}";
read -p "Нажмите Enter..."; continue; }

        echo -e "${GREEN}[+] Полное развертывание завершено!${NC}"
        ./check_cluster.sh
        read -p "Нажмите Enter для продолжения..."
        ;;
    9)
        echo -e "${RED}[!] ВНИМАНИЕ: Будут удалены все виртуальные
машины Project_03!${NC}"
        ./destroy_project_03.sh
        read -p "Нажмите Enter для продолжения..."
        ;;
    0)
        echo -e "${GREEN}До свидания!${NC}"
        exit 0
        ;;
    *)
        echo -e "${RED}Неверный выбор. Попробуйте снова.${NC}"
        read -p "Нажмите Enter для продолжения..."
        ;;
esac
done
EOF

```

```
chmod +x /home/altlinux/Projects/Project_03/master_script.sh
```

README.md для проекта

```
bash
```

```
cat > /home/altlinux/Projects/Project_03/README.md << 'EOF'
```

```
# Project_03 - Docker Swarm кластер
```

```
## Топология
```

- **Cloud-ADM** - управляющая машина
- **Cloud-Router** - шлюз, NAT, DNS (192.168.100.1)
- **master01** - менеджер Swarm (192.168.100.10)
- **worker01** - воркер (192.168.100.11)
- **worker02** - воркер (192.168.100.12)

```
## Функционал
```

1. Docker Swarm кластер
2. Приватный registry (порт 5000)
3. Stack school-site-project с сервисами:
 - Web (статический сайт) - порты 80, 443
 - Redis (кэширование)
 - PostgreSQL (хранение данных)

```
## Использование
```

```
### Предварительные требования
```

1. Установить sshpass: `sudo apt-get install sshpass`
2. Настроить SSH ключ для доступа к Прокмох: `ssh-copy-id root@<Proxmox_IP>`
3. Скачать образ alt-p11-cloud-x86_64.qcow2 на Прокмох в /var/lib/vz/template/iso/

```
### Запуск
```

```
```bash
```

```
cd /home/altlinux/Projects/Project_03
```

```
./master_script.sh
```

## Проверка

- Сайт: <http://school-site.au.team> или <https://school-site.au.team>

- Registry: [http://master01:5000/v2/\\_catalog](http://master01:5000/v2/_catalog)
- Статус кластера: `./check_cluster.sh`

## Структура скриптов

- `deploy_project_03.sh` - создание ВМ в Proxmox
- `configure_router.sh` - настройка Cloud-Router
- `install_docker.sh` - установка Docker на все узлы
- `init_swarm.sh` - инициализация Swarm кластера
- `setup_registry.sh` - создание приватного registry
- `create_stack.sh` - деплой stack с приложением
- `check_cluster.sh` - проверка статуса
- `destroy_project_03.sh` - удаление всех ресурсов
- `master_script.sh` - главное меню

## Важно!

Перед запуском отредактируйте `deploy_project_03.sh` и `destroy_project_03.sh`:

- Укажите правильный IP вашего Proxmox сервера в переменной `PM_HOST`  
EOF

```
text
```

```

```

```
Инструкция по использованию
```

### 1. \*\*Подготовка Cloud-ADM:\*\*

```
```bash
```

```
# Установите необходимые пакеты
```

```
sudo apt-get update
```

```
sudo apt-get install -y sshpass curl openssl
```

```
# Сгенерируйте SSH ключ для доступа к Proxmox
```

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/id_rsa -N ""
```

```
ssh-copy-id root@<IP_вашего_Proxmox>
```

2. Настройте скрипты:

- В `deploy_project_03.sh` и `destroy_project_03.sh` замените `PM_HOST` на IP вашего Proxmox сервера
3. Запустите главный скрипт:
 4. `bash`

```
cd /home/altlinux/Projects/Project_03
```

5. `./master_script.sh`

ЗАПАС [2.CG](#)

PROJECT_01

Отлично, делаем **Модуль В полностью под Proxmox VE**, строго по твоей топологии:

Cloud-ADM → Internet (Floating IP) → haproxy01 → Cloud-Network → game01/game02/game03

Ниже даю **пошаговую архитектуру + готовые рабочие скрипты**.

Архитектура в Proxmox

◆ Сети

Создаём 2 bridge:

1 vmbr0 — External (Internet)

- Привязан к физическому интерфейсу
- Для Floating IP

2 vmbr1 — Cloud-Network (внутренняя)

- Linux bridge
- Без физического интерфейса
- Для связи:

- haproxy01
 - game01
 - game02
 - game03
-

1 Cloud-ADM (ручное создание)

Параметры:

- 2 vCPU
 - 4GB RAM
 - 30GB Disk
 - ISO: alt-workstation-10.4-p10-cloud
 - Сети:
 - net0 → vmbr0
 - net1 → vmbr1
-

SSH настройка

На локальном ПК:

```
ssh-keygen -t rsa -b 4096 -f Cloud-ADM.pem
```

Загрузить публичный ключ в Cloud-ADM:

```
mkdir -p ~/.ssh  
nano ~/.ssh/authorized_keys
```

Вставить содержимое Cloud-ADM.pem.pub

Права:

```
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/authorized_keys
```

◆ PuTTY профиль

1. Host Name: Floating IP Cloud-ADM
2. Connection → SSH → Auth → Private key → Cloud-ADM.pem
3. Session → Save → Cloud-ADM

Теперь подключение без параметров.

RDP

Установить xrdp:

```
sudo apt install xrdp -y  
sudo systemctl enable xrdp  
sudo systemctl start xrdp
```

Логин:

```
user: altlinux  
pass: P@ssw0rd
```

Сохранить .rdp файл как Cloud-ADM.rdp

2 Project_01

Создаём директорию:

```
mkdir -p /home/altlinux/Projects/Project_01  
cd /home/altlinux/Projects/Project_01
```



deploy_project_01.sh

```
#!/bin/bash
```

```
TEMPLATE="alt-p11-cloud-x86_64.qcow2"
```

```
STORAGE="local-lvm"
```

```
BRIDGE_INT="vmbr1"
```

```
BRIDGE_EXT="vmbr0"
```

```
create_vm () {
```

```
VMID=$1
```

```
NAME=$2
```

```
BRIDGE=$3
```

```
qm create $VMID --name $NAME --memory 1024 --cores 1 \
```

```
--net0 virtio,bridge=$BRIDGE \
```

```
--scsihw virtio-scsi-pci \
```

```
--scsi0 $STORAGE:10 \
```

```
--ostype l26
```

```
qm importdisk $VMID $TEMPLATE $STORAGE
```

```
qm set $VMID --boot c --bootdisk scsi0
```

```
qm start $VMID
```

```
}
```

```
# game servers
```

```
create_vm 201 game01 $BRIDGE_INT
```

```
create_vm 202 game02 $BRIDGE_INT
```

```
create_vm 203 game03 $BRIDGE_INT
```

```
# haproxy
```

```
qm create 204 --name haproxy01 --memory 1024 --cores 1 \
```

```
--net0 virtio,bridge=$BRIDGE_EXT \
```

```
--net1 virtio,bridge=$BRIDGE_INT \
```

```
--scsihw virtio-scsi-pci \
```

```
--scsi0 $STORAGE:10 \
```

```
--ostype l26
```

```
qm importdisk 204 $TEMPLATE $STORAGE
qm set 204 --boot c --bootdisk scsi0
qm start 204
```

```
echo "Project_01 deployed."
```

```
chmod +x deploy_project_01.sh
```



configure_project_01.sh

```
#!/bin/bash
```

```
GAME_IPS=("192.168.100.11" "192.168.100.12" "192.168.100.13")
HAPROXY_IP="192.168.100.10"
```

```
# ① Установка Docker
for IP in "${GAME_IPS[@]}"
do
ssh root@$IP "
apt update &&
apt install docker.io -y &&
systemctl enable docker &&
systemctl start docker
"
done
```

```
# ② Легковесный multi-stage Dockerfile
cat <<EOF > Dockerfile
FROM node:18-alpine AS builder
WORKDIR /app
COPY . .
RUN npm install && npm run build

FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
EOF
```

```
# ③ HAProxy настройка
ssh root@$HAPROXY_IP "
```

```
apt install haproxy -y
cat <<CFG > /etc/haproxy/haproxy.cfg
global
    daemon
    maxconn 256

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend http_front
    bind *:80
    bind *:443 ssl crt /etc/ssl/private/haproxy.pem
    default_backend game_servers

backend game_servers
    balance roundrobin
    server game01 192.168.100.11:80 check
    server game02 192.168.100.12:80 check
    server game03 192.168.100.13:80 check

listen stats
    bind *:8404
    stats enable
    stats uri /haproxy?stats
CFG

systemctl restart haproxy
"

echo "Project configured."

chmod +x configure_project_01.sh
```



HTTPS без ошибки сертификата

Устанавливаем Let's Encrypt:

```
apt install certbot -y
certbot certonly --standalone -d game.au.team
```

Объединяем сертификат:

```
cat fullchain.pem privkey.pem > /etc/ssl/private/haproxy.pem
```



destroy_project_01.sh

```
#!/bin/bash
```

```
for VMID in 201 202 203 204
do
qm stop $VMID
qm destroy $VMID --purge
done
```

```
echo "All resources destroyed."
```

```
chmod +x destroy_project_01.sh
```



DNS (важно)

На Cloud-ADM:

```
192.168.100.10 game01.dev.au.team
192.168.100.10 game02.dev.au.team
192.168.100.10 game03.dev.au.team
192.168.100.10 haproxy01.dev.au.team
```

PROJECT_02

Отлично. Ниже даю **готовое решение под Proxmox**, строго по топологии:

CLOUD_ADM

↓

Internet (Floating IP)



Cloud-Router (NAT)



Cloud-Network (vmbr1)



ACM-Server

DB-Server

BAR-Agent01

Скрипты выполняются на **CLOUD_ADM**.



Предположения по сети

Узел	IP
Cloud-Router (LAN)	192.168.200.1
ACM-Server	192.168.200.1 0
DB-Server	192.168.200.2 0
BAR-Agent01	192.168.200.3 0

Cloud-Network: 192.168.200.0/24

Bridge: vmbr1

Образ: alt-p11-cloud-x86_64.qcow2

Storage: local-lvm



Создаём рабочую директорию

```
mkdir -p /home/altlinux/Projects/Project_02
```

```
cd /home/altlinux/Projects/Project_02
```

1 deploy_project_02.sh

Создаёт VM строго по требованиям.

```
#!/bin/bash

TEMPLATE="/var/lib/vz/template/qcow2/alt-p11-cloud-x86_64.qcow2"
STORAGE="local-lvm"
BRIDGE="vbr1"

create_vm () {
VMID=$1
NAME=$2
MEM=$3
DISK=$4
IP=$5

qm create $VMID --name $NAME --memory $MEM --cores 1 \
--net0 virtio,bridge=$BRIDGE \
--scsihw virtio-scsi-pci \
--ostype l26

qm importdisk $VMID $TEMPLATE $STORAGE
qm set $VMID --scsi0 $STORAGE:vm-$VMID-disk-0
qm resize $VMID scsi0 ${DISK}G
qm set $VMID --boot order=scsi0

qm set $VMID --ipconfig0 ip=${IP}/24,gw=192.168.200.1

qm start $VMID
}

create_vm 401 ACM-Server 2048 20 192.168.200.10
create_vm 402 DB-Server 1024 20 192.168.200.20
create_vm 403 BAR-Agent01 1024 10 192.168.200.30

echo "VM deployment completed."

chmod +x deploy_project_02.sh
```

2 configure_project_02.sh

Реализует:

- PostgreSQL
- Кибер Бэкап >= 18
- root/toor
- агенты
- регистрацию
- доступ cb.au.team:9877

```
#!/bin/bash
```

```
ACM_IP="192.168.200.10"
```

```
DB_IP="192.168.200.20"
```

```
AGENT_IP="192.168.200.30"
```

```
# Добавляем DNS имя на Cloud-ADM
```

```
grep -q cb.au.team /etc/hosts || \  
echo "$ACM_IP cb.au.team" >> /etc/hosts
```

```
echo "Installing PostgreSQL on ACM-Server..."
```

```
ssh root@$ACM_IP "  
apt update &&  
apt install postgresql postgresql-contrib -y &&  
sudo -u postgres psql -c \"CREATE DATABASE cyberbackup;\" &&  
sudo -u postgres psql -c \"CREATE USER cbuser WITH PASSWORD  
'cbpass';\" &&  
sudo -u postgres psql -c \"GRANT ALL PRIVILEGES ON DATABASE  
cyberbackup TO cbuser;\"  
"
```

```
echo "Installing Cyber Backup Management Server..."
```

```
ssh root@$ACM_IP "
```

```
wget -O cb.deb https://download.cyberbackup.ru/CB18_linux_x86_64.deb
&&
apt install ./cb.deb -y
"

echo "Setting root password..."

ssh root@$ACM_IP "echo 'root:toor' | chpasswd"

echo "Installing Linux Agent on BAR-Agent01..."

ssh root@$AGENT_IP "
wget -O agent.deb
https://download.cyberbackup.ru/agent_linux_x86_64.deb &&
apt install ./agent.deb -y
"

echo "Registering Linux Agent..."

ssh root@$AGENT_IP "
cbregister --server $ACM_IP --user root --password toor
"

echo "Installing PostgreSQL Agent on DB-Server..."

ssh root@$DB_IP "
apt install postgresql -y &&
wget -O pgagent.deb
https://download.cyberbackup.ru/agent_postgresql.deb &&
apt install ./pgagent.deb -y
"

echo "Registering PostgreSQL Agent..."

ssh root@$DB_IP "
cbregister --server $ACM_IP --user root --password toor
"

echo "Configuration complete."

chmod +x configure_project_02.sh
```

3 destroy_project_02.sh

```
#!/bin/bash

for VMID in 401 402 403
do
qm stop $VMID
qm destroy $VMID --purge
done

echo "All resources removed."

chmod +x destroy_project_02.sh
```

Cloud-Router (обязательно)

На Cloud-Router:

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sysctl -p

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
apt install iptables-persistent -y
netfilter-persistent save
```

Порядок выполнения

```
./deploy_project_02.sh
# дождаться старта VM

./configure_project_02.sh
```

PROJECT_03



1. Подготовка директории

На CLOUD-ADM:

```
mkdir -p /home/altlinux/Projects/Project_02  
cd /home/altlinux/Projects/Project_02
```



2. Скрипт развертывания VM

deploy_project_02.sh

```
#!/bin/bash
```

```
TEMPLATE="/var/lib/vz/template/qcow2/alt-p11-cloud-x86_64.qcow2"  
STORAGE="local-lvm"  
BRIDGE="vbr1"  
GATEWAY="192.168.200.1"
```

```
create_vm () {  
  VMID=$1  
  NAME=$2  
  RAM=$3  
  DISK=$4  
  IP=$5
```

```
  qm create $VMID --name $NAME --memory $RAM --cores 1 \  
  --net0 virtio,bridge=$BRIDGE \  
  --scsihw virtio-scsi-pci --ostype l26
```

```
  qm importdisk $VMID $TEMPLATE $STORAGE  
  qm set $VMID --scsi0 $STORAGE:vm-$VMID-disk-0  
  qm resize $VMID scsi0 ${DISK}G  
  qm set $VMID --boot order=scsi0
```

```
  qm set $VMID --ipconfig0 ip=${IP}/24,gw=${GATEWAY}
```

```
  qm start $VMID  
}
```

```
create_vm 501 master1 2048 20 192.168.200.10
create_vm 502 worker1 1024 20 192.168.200.20
create_vm 503 worker2 1024 10 192.168.200.30
```

```
echo "VM deployment complete."
```

```
chmod +x deploy_project_02.sh
```

3. Скрипт настройки инфраструктуры

configure_project_02.sh

Этот скрипт:

- Устанавливает PostgreSQL на master1
- Устанавливает Кибер Бэкап ≥18
- Настраивает root/toor
- Устанавливает агенты
- Регистрирует их
- Делает доступ cb.au.team:9877

```
#!/bin/bash
```

```
MASTER="192.168.200.10"
```

```
DB="192.168.200.20"
```

```
AGENT="192.168.200.30"
```

```
echo "Настройка DNS имени на Cloud-ADM..."
```

```
grep -q cb.au.team /etc/hosts || \
```

```
echo "$MASTER cb.au.team" >> /etc/hosts
```

```
#####
```

```
echo "Установка PostgreSQL на master1..."
```

```
#####
```

```
ssh root@$MASTER <<EOF
```

```
apt update
```

```
apt install postgresql postgresql-contrib -y
```

```
sudo -u postgres psql -c "CREATE DATABASE cyberbackup;"
```

```
sudo -u postgres psql -c "CREATE USER cbuser WITH PASSWORD  
'cbpass';"
```

```
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE  
cyberbackup TO cbuser;"
```

```
echo "root:toor" | chpasswd
```

```
EOF
```

```
#####
```

```
echo "Установка Cyber Backup Server >=18..."
```

```
#####
```

```
ssh root@$MASTER <<EOF
```

```
wget -O cb.deb https://download.cyberbackup.ru/CB18_linux_x86_64.deb
```

```
apt install ./cb.deb -y
```

```
EOF
```

```
#####
```

```
echo "Установка Linux Agent на worker2..."
```

```
#####
```

```
ssh root@$AGENT <<EOF
```

```
wget -O agent.deb
```

```
https://download.cyberbackup.ru/agent_linux_x86_64.deb
```

```
apt install ./agent.deb -y
```

```
cbregister --server $MASTER --user root --password toor
```

```
EOF
```

```
#####
```

```
echo "Установка PostgreSQL + Agent на worker1..."
```

```
#####
```

```
ssh root@$DB <<EOF
apt install postgresql -y
wget -O pgagent.deb
https://download.cyberbackup.ru/agent_postgresql.deb
apt install ./pgagent.deb -y

cbregister --server $MASTER --user root --password toor
EOF

echo "Configuration completed."

chmod +x configure_project_02.sh
```

4. Скрипт удаления

destroy_project_02.sh

```
#!/bin/bash

for VMID in 501 502 503
do
qm stop $VMID
qm destroy $VMID --purge
done

echo "All VMs destroyed."

chmod +x destroy_project_02.sh
```

5. Cloud-Router (обязательно!)

На Cloud-Router:

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
sysctl -p

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
apt install iptables-persistent -y  
netfilter-persistent save
```



Порядок выполнения

```
./deploy_project_02.sh  
# дождаться старта VM
```

```
./configure_project_02.sh
```

Проверка:

```
http://cb.au.team:9877
```

Логин:

```
root  
toor
```