**СЕТЬ ЦОД**

**1. Настройка сетевых интерфейсов:**

Отключаем NetworkManager, если включен:

*systemctl status NetworkManager*

*systemctl stop NetworkManager*

*systemctl disable NetworkManager*

Через networking:

*nano /etc/network/interfaces*

*auto eth0*

*iface eth0 inet static*

*address x.x.x.x*

*netmask x.x.x.x*

*gateway x.x.x.x*

*dns-nameservers x.x.x.x x.x.x.x*

*systemctl restart networking*

DNS:

*nano /etc/resolv.conf*

*nameserver x.x.x.x*

**2. SSH:**

На всех тачках создаем пользователя cod\_admin:

*sudo adduser cod\_admin*

*sudo usermod -aG sudo cod\_admin*

Делаем доступ к sudo без пароля:

*visudo*, в конец файла добавляем:

*cod\_admin ALL=(ALL) NOPASSWD:ALL*

На DC-STORAGE генерим SSH-ключи:

*mkdir -p /ssh\_keys*

*ssh-keygen -t rsa -b 4096 -f /ssh\_keys/id\_rsa -N ""*

Копируем публичный ключ на целевой хост:

*ssh-copy-id -i /ssh\_keys/id\_rsa.pub cod\_admin@10.15.10.100*

Настройка SSH на DC-MAILSERVER:

*nano /etc/ssh/sshd\_config*

*PasswordAuthentication no*

*ChallengeResponseAuthentication no*

*UsePAM no*

*AllowUsers cod\_admin*

Добавляем iptables:

*iptables -A INPUT -p tcp -s 10.15.10.0/24 --dport 22 -j ACCEPT*

*iptables -A INPUT -p tcp --dport 22 -j REJECT*

Сохранение правил:

*netfilter-persistent save*

Настройка SSH на DC-STORAGE:

*nano /etc/ssh/sshd\_config*

*PasswordAuthentication yes*

*UsePAM yes*

*AllowUsers cod\_admin*

Добавляем iptables:

*iptables -A INPUT -p tcp -s VPN --dport 22 -j ACCEPT*

*iptables -A INPUT -p tcp --dport 22 -j REJECT*

**3. Настройка GRE:**

Пример для DC-RTR-1:

*nano /etc/network/interfaces*

*auto gre-msk*

*iface gre-msk inet static*

*address 10.6.6.1*

*netmask 255.255.255.252*

*pre-up ip tunnel add gre-msk mode gre remote 188.121.90.2 local 200.100.100.20 ttl 255*

*post-up ip route add 192.168.1.0/24 via 10.6.6.2*

*pre-down ip tunnel del gre-msk mode gre remote 188.121.90.2 local 200.100.100.20 ttl 255*

*auto gre-yekt*

*iface gre-yekt inet static*

*address 10.7.7.1*

*netmask 255.255.255.252*

*pre-up ip tunnel add gre-yekt mode gre remote 88.8.8.27 local 200.100.100.20 ttl 255*

*post-up ip route add 192.168.2.0/24 via 10.7.7.2*

*pre-down ip tunnel del gre-yekt mode gre remote 88.8.8.27 local 200.100.100.20 ttl 255*

На MSK-RTR и YEKT-RTR маршруты в 10.15.10.0/24 и в 192.168.\*.0/24 добавить только для основных роутеров-туннелей.

**Настройка IPSec:**

*apt install strongswan*

*nano /etc/ipsec.conf*

*config setup*

*charondebug="all"*

*uniqueids=no*

*strictcrlpolicy=no*

*conn gre-msk*

*authby=secret*

*left=200.100.100.20*

*right=188.121.90.2*

*leftprotoport=gre*

*rightprotoport=gre*

*type=tunnel*

*esp=aes256-sha1*

*ike=aes256-sha1-modp1024*

*auto=start*

*conn gre-yekt*

*authby=secret*

*left=200.100.100.20*

*right=88.8.8.27*

*leftprotoport=gre*

*rightprotoport=gre*

*type=tunnel*

*esp=aes256-sha1*

*ike=aes256-sha1-modp1024*

*auto=start*

*nano /etc/ipsec.secrets*

*200.100.100.20 188.121.90.2 : PSK "At0mSk1lls"*

*200.100.100.20 88.8.8.27 : PSK "At0mSk1lls"*

*systemctl restart strongswan-starter* ИЛИ *ipsec start*, проверить подключения – *ipsec status*

**4. Настройка OSPF:**

Пример для DC-RTR-1:

Включить ospfd в /etc/frr/daemons

*vtysh*

*conf t*

*router ospf*

*network 10.6.6.0/30 area 10*

*network 10.7.7.0/30 area 10*

*exit*

*interface gre-msk*

*ip ospf authentication message-digest*

*ip ospf message-digest-key 1 md5 C00lCompanY*

*exit*

*interface gre-yekt*

*ip ospf authentication message-digest*

*ip ospf message-digest-key 1 md5 C00lCompanY*

*exit*

*do write*

**5. Настройка PAT, пример для DC-RTR-1:**

*iptabels -t nat -A POSTROUTING -s 10.15.10.0/24 -o ens3 -j MASQUERADE*

Если не заработает, пишем статический NAT

*iptables -t nat -A POSTROUTING -s 10.15.10.0/24 -o ens3 -j SNAT --to-source 100.200.100.20*

Также разрешаем IP форвардинг:

*nano /etc/sysctl.conf*

*net.ipv4.ip\_forward=1*

*sysctl -p*

Если правильно поняла, то PAT также нужен на туннели в MSK-RTR и YEKT-RTR, чтобы была доступность из локали в ЦОД?

**6. Настройка VRRP через keepalived:**

*apt install keepalived*

*nano /etc/keepalived/keepalived.conf*

Конфигурация для DC-RTR-1:

*vrrp\_instance VI\_MSK {*

*state BACKUP*

*interface ens4*

*virtual\_router\_id 51*

*priority 100*

*advert\_int 1*

*virtual\_ipaddress {*

*10.15.10.1/24*

*}*

*}*

*vrrp\_instance VI\_YEKT {*

*state MASTER*

*interface ens4*

*virtual\_router\_id 52*

*priority 150*

*advert\_int 1*

*virtual\_ipaddress {*

*10.15.10.1/24*

*}*

*}*

Конфигурация для DC-RTR-2:

*vrrp\_instance VI\_MSK {*

*state MASTER*

*interface ens4*

*virtual\_router\_id 51*

*priority 150*

*advert\_int 1*

*virtual\_ipaddress {*

*10.15.10.1/24*

*}*

*}*

*vrrp\_instance VI\_YEKT {*

*state BACKUP*

*interface ens4*

*virtual\_router\_id 52*

*priority 100*

*advert\_int 1*

*virtual\_ipaddress {*

*10.15.10.1/24*

*}*

*}*

*systemctl restart keepalived*

**7. Настройка приоритетов на MSK и YEKT:**

MSK-RTR:

Создаем скрипт:

*nano /root/mail\_check.sh:*

*#!/bin/bash*

*mail\_ip=”10.15.10.100”*

*main\_dc=”10.5.5.1”*

*backup\_dc=”10.7.7.1”*

*ping\_test() {*

*ping -c 1 -W 1 $1 > /dev/null 2>&1*

*return $?*

*}*

*ping\_test $mail\_ip*

*mail\_is\_up=$?*

*ping\_test $main\_dc*

*main\_is\_up=$?*

*if [ $mail\_is\_up -eq 0 ]; then*

*if [ $main\_is\_up -eq 0 ]; then*

*ip route change 10.15.10.0/24 via $main\_dc*

*ip route change 192.168.2.0/24 via $main\_dc*

*fi*

*else*

*ip route change 10.15.10.0/24 via $backup\_dc*

*ip route change 192.168.2.0/24 via $backup\_dc*

*fi*

YEKT-RTR:

Создаем скрипт:

*nano /root/mail\_check.sh:*

*#!/bin/bash*

*mail\_ip=”10.15.10.100”*

*main\_dc=”10.6.6.1”*

*backup\_dc=”10.8.8.1”*

*ping\_test() {*

*ping -c 1 -W 1 $1 > /dev/null 2>&1*

*return $?*

*}*

*ping\_test $mail\_ip*

*mail\_is\_up=$?*

*ping\_test $main\_dc*

*main\_is\_up=$?*

*if [ $mail\_is\_up -eq 0 ]; then*

*if [ $main\_is\_up -eq 0 ]; then*

*ip route change 10.15.10.0/24 via $main\_dc*

*ip route change 192.168.1.0/24 via $main\_dc*

*fi*

*else*

*ip route change 10.15.10.0/24 via $backup\_dc*

*ip route change 192.168.1.0/24 via $backup\_dc*

*fi*

Добавляем задачу в cron на MSK-RTR и YEKT-RTR:

*crontab -e*

*\* \* \* \* \* /root/mail\_check.sh*

**8. Настройка OpenConnect**

На CLOUD-VM качаем ocserv:

*apt install ocserv*

Создаем директорию, куда поместим сертификат:

*mkdir -p /etc/ocserv/certs*

Копируем туда сертификат с DC, выдаем права:

*chmod 600 server-key.pem*

Редактируем конфиг:

*nano /etc/ocserv/ocserv.conf:*

*auth = "plain[/etc/ocserv/passwd]"*

*tcp-port = 443*

*udp-port = 443*

*run-as-user = ocserv*

*run-as-group = ocserv*

*# === Сертификаты ===*

*server-cert = /etc/ocserv/certs/server-cert.pem*

*server-key = /etc/ocserv/certs/server-key.pem*

*# === Сетевые настройки ===*

*ipv4-network = 10.10.10.0 #IP VPN сети*

*ipv4-netmask = 255.255.255.0*

*dns = 8.8.8.8*

*# === Доступ пользователей ===*

*default-domain = vpn.atomskills.ru*

*max-clients = 16*

*max-same-clients = 2*

*keepalive = 300*

*# === Пути до сетей DC, MSK, YEKT ===*

*config\_per\_user = /etc/ocserv/config-per-user/*

Создаем файл с пользователями:

*touch /etc/ocserv/passwd*

*chmod 600 /etc/ocserv/passwd*

Создаем пользователей:

*ocpasswd -c /etc/ocserv/passwd cod\_admin*

Вводим пароль, повторяем. Создаем двух других пользователей.

Создаем конфиги для юзеров, куда прописываем маршруты:

*mkdir -p /etc/ocserv/config-per-user*

*nano /etc/ocserv/config-per-use/cod\_admin:*

*route = ip\_сети*

*chmod 600 /etc/ocserv/config-per-user/cod\_admin*

Перезапускаем ocserv:

*systemctl restart ocserv*

*systemctl status ocserv*

Проверяем подключение на REMOTE-TERMINAL:

*apt install openconnect*

*openconnect --user=cod\_admin vpn.atomskills.ru --passwd-on-stdin <<< “P@ssw0rd1234”*

**9. Настройка NFS**

На DC-STORAGE:

*apt install nfs-kernel-server*

*sudo mkdir -p /storage/it*

*sudo mkdir -p /storage/office*

*sudo groupadd IT*

*sudo groupadd office*

*sudo chown root:IT /storage/it*

*sudo chown root:office /storage/office*

*sudo chmod 1770 /storage/it*

*sudo chmod 1770 /storage/office*

Настраиваем NFS:

*nano /etc/exports:*

*/storage/it 192.168.1.0/24(rw,sync,no\_subtree\_check,root\_squash)*

*/storage/office 192.168.1.0/24(rw,sync,no\_subtree\_check,root\_squash)*

*exportfs -ra*

*systemctl restart nfs-server*

На клиентах:

*apt install nfs-common libpam-mount*

*groupadd IT*

*groupadd office*

*useradd -m -g IT testuser1*

*useradd -m -g office testuser2*

*passwd testuser1*

*passwd testuser2*

Редактируем PAM-mount:

*nano /etc/security/pam\_mount.conf.xml:*

В секции <pam\_mount> пишем:

*<volume user="\*" fstype="nfs" server="10.15.10.150" path="/storage/it" mountpoint="~/Desktop/IT\_Folder" options="vers=4" sgrp="IT"/>*

*<volume user="\*" fstype="nfs" server="10.15.10.150" path="/storage/office" mountpoint="~/Desktop/Office\_Folder" options="vers=4" sgrp="office"/>*

Активируем PAM-mount:

*nano /etc/pam.d/common-session:*

*session optional pam\_mount.so*

**10. Настройка LVM и бэкапов файлов пользователей группы office**

На DC-STORAGE:

*apt install lvm2 cryptsetup*

Создаем директорию, где будут лежать виртуальные диски:

*mkdir -p /var/lvm-disks*

Создаем три файла для виртуальных дисков:

*sudo dd if=/dev/zero of=/var/lvm-disks/disk1.img bs=1M count=1024*

*sudo dd if=/dev/zero of=/var/lvm-disks/disk2.img bs=1M count=1024*

*sudo dd if=/dev/zero of=/var/lvm-disks/disk3.img bs=1M count=1024*

Подключение файлов-образов как loop-устройства:

*sudo losetup /dev/loop0 /var/lvm-disks/disk1.img*

*sudo losetup /dev/loop1 /var/lvm-disks/disk2.img*

*sudo losetup /dev/loop2 /var/lvm-disks/disk3.img*

Добавляем скрипт в автозагрузку, чтобы вирутальные диски подключались:

*sudo nano /etc/systemd/system/lvm-loop.service:*

*[Unit]*

*Description=Attach loop devices for LVM*

*DefaultDependencies=no*

*After=systemd-modules-load.service*

*[Service]*

*Type=oneshot*

*ExecStart=/bin/bash -c 'losetup /dev/loop0 /var/lvm-disks/disk1.img; losetup /dev/loop1 /var/lvm-disks/disk2.img; losetup /dev/loop2 /var/lvm-disks/disk3.img;'*

*RemainAfterExit=true*

*[Install]*

*WantedBy=multi-user.target*

*systemctl daemon-reload*

*systemctl enable lvm-loop*

Создаем физические тома:

*sudo pvcreate /dev/loop0 /dev/loop1 /dev/loop2*

Создаем группу томов:

*sudo vgcreate vg\_crypto /dev/loop0 /dev/loop1 /dev/loop2*

Создаем логический том:

*sudo lvcreate -L 2.8G -n lv\_secure vg\_crypto*

Также нужно автоматически активировать логические тома:

*sudo nano /etc/systemd/system/lvm-activation.service:*

*[Unit]*

*Description=Activate LVM volumes*

*DefaultDependencies=no*

*After=lvm-loop.service*

*[Service]*

*Type=oneshot*

*ExecStart=/sbin/vgchange -ay*

*RemainAfterExit=true*

*[Install]*

*WantedBy=multi-user.target*

*systemctl daemon-reload*

*systemctl enable lvm-activation*

Шифруем:

*sudo cryptsetup luksFormat /dev/vg\_crypt/lv\_secure*

Пароль: *P@ssw0rd*

Открываем зашифрованный том:

*sudo cryptsetup luksOpen /dev/vg\_crypto/lv\_secure crypto\_lv*

Форматируем:

*mkfs.ext4 /dev/mapper/crypto\_lv*

Добавляем ключевой файл:

*dd if=/dev/random of=/root/luks\_key bs=1024 count=4*

*chmod 400 /root/luks\_key*

Добавляем ключ в шифрованный раздел:

*cryptsetup luksAddKey /dev/vg\_crypto/lv\_secure /root/luks\_key*

Настройка автоматического монтирования:

*sudo nano /etc/crypttab*

*crypto\_lv /dev/vg\_crypto/lv\_secure /root/luks\_key luks*

*sudo nano /etc/fstab*

*/dev/mapper/crypto\_lv /crypto-folder ext4 defaults 0 2*

*systemctl daemon-reload*

**Резервное копирование:**

*apt install inotify-tools rsync*

*nano /usr/local/bin/backup\_save.sh:*

*#!/bin/bash*

*SOURCE\_DIR="/storage/office"*

*DEST\_DIR="/crypto-folder"*

*backup\_file() {*

*FILE=$1*

*DEST\_PATH="${DEST\_DIR}/$(basename "$FILE")"*

*if [[ "$(basename "$FILE" | tr '[:upper:]' '[:lower:]')" =~ save ]]; then*

*# Копирование файла*

*rsync -a --ignore-existing "$FILE" "$DEST\_PATH"*

*fi*

*}*

*inotifywait -m -r -e create -e moved\_to -e modify --format "%w%f" "$SOURCE\_DIR" | while read FILE*

*do*

*backup\_file "$FILE"*

*done*

*chmod +x /usr/local/bin/backup\_save.sh*

Добавляем юнит на запуск скрипта при старте системы:

*nano /etc/systemd/system/backup\_save.service:*

*[Unit]*

*Description=Backup Save Files Service*

*After=network.target*

*[Service]*

*ExecStart=/usr/local/bin/backup\_save.sh*

*Restart=always*

*User=root*

*Group=root*

*[Install]*

*WantedBy=multi-user.target*

*sudo systemctl daemon-reload*

*sudo systemctl enable backup\_save.service*

*sudo systemctl start backup\_save.service*

**СЕТЬ МСК**

**1. DHCP-сервер на MSK-RTR:**

*sudo apt install isc-dhcp-server*

*nano /etc/default/isc-dhcp-server:*

*DHCPDv4\_CONF=/etc/dhcp/dhcpd.conf*

*INTERFACESv4="ens4" #*Внутренний интерфейс

*nano /etc/dhcp/dhcpd.conf:*

*subnet 192.168.1.0 netmask 255.255.255.0 {*

*range 192.168.1.50 192.168.1.100;*

*option routers 192.168.1.1;*

*option domain-name-servers 192.168.1.2, 77.88.8.1;*

*option domain-name "company.cool";*

*default-lease-time 600;*

*max-lease-time 7200;*

*}*

*sudo systemctl restart isc-dhcp-server*

**СЕТЬ ЕКТ**

**1. Перевод Zabbix на HTTPS:**

Копируем сертификат и ключ в директорию /etc/ssl/zabbix (или в ту директорию, которая по заданию указана, тогда указываем путь до нее)

Редактируем конфигурацию apache2:

*nano /etc/apache2/sites-available/000-default.conf:*

Добавляем новую секцию, аналогично существующей, 80 порт потом комментируем, примерно должно быть так:

*<VirtualHost \*:443>*

*ServerAdmin webmaster@localhost*

*DocumentRoot /var/www/html*

*SSLEngine on*

*SSLCertificateFile /etc/ssl/zabbix/zabbix.crt*

*SSLCertificateKeyFile /etc/ssl/zabbix/zabbix.key*

*ErrorLog ${APACHE\_LOG\_DIR}/error.log*

*CustomLog ${APACHE\_LOG\_DIR}/access.log combined*

*</VirtualHost>*

Активируем SSL и перезапускаем Apache:

*a2enmod ssl*

*systemctl restart apache2*

**Настройка отправки сообщений на почту:**

В веб-интерфейсе Zabbix переходим в Администрирование – Способ оповещений, нажимаем на Email, в параметрах указываем:

*SMTP сервер – mail.company.cool*

*SMTP helo – company.cool*

*SMTP email –* [*zabbix@company.cool*](mailto:zabbix@company.cool) (проверить, чтобы такой пользователь был на почтовом сервере, если нет – создать)

Можно проверить работу: «Тест» на [admin@company.cool](mailto:admin@company.cool).

**Если не будет пользователя, создаем:**

Администрирование – Пользователи – Создать пользователя:

Группа – Zabbix administrator

Оповещения – Добавить:

Тип – Email

Отправлять на – [admin@company.cool](mailto:admin@company.cool)

Права доступа – Роль – Super admin role (на другие роли почему-то не отправляет письма).

Поменять язык – Настройка пользователя – Профиль – Язык – Русский

**Настройка метрики для отправки сообщений:**

Переходим в Настройки – Узлы сети – Zabbix server, выбираем Триггеры, создаем новый:

Указываем имя, важность, выражение:

Для ЦПУ – last(/Zabbix server/system.cpu.util)>80 (Добавить – Элемент данных – Linux: CPU utilization)

Для диска – last(/Zabbix server/vfs.fs.size[/, pused])>90 (/: Space utilization)

**Настройка отправки почты по триггеру:**

Настройка – Действие – Действие для триггера – Создать действие:

Пишем имя, выбираем Условие:

Тип – триггер

Оператор – равно

Триггер – наши созданные

На вкладке «Операции», добавляем операцию, выбираем «Отправка пользователям» - admin.

Отправка только через: EMAIL

Пользовательское сообщение:

Тема: Alert: {TRIGGER.NAME}

Сообщение: Event: {EVENT.NAME} on {HOST.NAME} at {DATE}.

**Добавление YEKT-RTR и YEKT-DB в мониторинг:**

*apt install zabbix-agent*

*nano /etc/zabbix/zabbix\_agentd.conf:*

*Server=192.168.2.150*

*ServerActive=192.168.2.150*

*Hostname=yetk-rtr*

*sudo systemctl restart zabbix-agent*

*sudo systemctl enable zabbix-agent*

В Zabbix:

Настройки – Узлы сети – Создать узел сети

Имя – yekt-rtr

Шаблон – Linux filesystems by Zabbix agent

Группа – Linux servers

Интерфейс – 192.168.2.1

Добавить

Настройка триггера:

Узлы сети – yetk-rtr – Триггеры – Создать новый триггер:

Для ЦПУ – last(/yekt-rtr/system.cpu.util)>80 (Добавить – Элемент данных – Linux: CPU utilization)

Для диска – last(/yekt-rtr/vfs.fs.size[/, pused])>90 (/: Space utilization)

**2. Сбор логов**

На YEKT-WORKER создаем пользователя:

*adduser yekt\_admin*

Пароль: *P@ssw0rd*

На YEKT-DB и YEKT-BILLING:

*sudo apt install rsyslog*

*nano /etc/rsyslog.conf:*

*module(load="imfile")*

*module(load="omfwd")*

*input(type="imfile"*

*File="/var/log/apache2/access.log"*

*Tag="apache-access"*

*Severity="info"*

*Facility="local1")*

*input(type="imfile"*

*File="/var/log/apache2/error.log"*

*Tag="apache-error"*

*Severity="error"*

*Facility="local1")*

*\*.\* @192.168.2.200:514*

Перезапускаем:

*sudo systemctl restart rsyslog*

На YEKT-WORKER:

*sudo apt install rsyslog*

*nano /etc/rsyslog.conf:*

*module(load="imudp")*

*input(type="imudp" port="514")*

*$template ApacheLogsDB,"/home/username/Desktop/Logs/YEKT-DB/%PROGRAMNAME%.log"*

*if ($fromhost-ip == '192.168.2.150') and ($programname == 'apache2') then -?ApacheLogsDB*

*$template ApacheLogsBilling,"/home/username/Desktop/Logs/YEKT-BILLING/%PROGRAMNAME%.log"*

*if ($fromhost-ip == '192.168.2.100') and ($programname == 'apache2') then -?ApacheLogsBilling*

**3. Веб-сайт на Flask:**

Ставим пакеты:

*sudo apt install python3 python3-pip*

Создаем файлы:

*mkdir /flask\_ site*

*cd /flask\_ site*

*touch app.py*

*mkdir templates*

*cd templates*

*touch login.html admin.html worker.html index.html*

Сертификат и ключ положить в /etc/ssl/flask\_auth – flask.crt и flask.key

Создаем виртуальное окружение для Flask в /flask\_site:

*python3 -m venv venv*

*source venv/bin/ac tivation*

*pip3 install flask*

**app.py:**

from flask import Flask, render\_template, request, redirect, url\_for, session

app = Flask(\_\_name\_\_)

app.secret\_key = ‘P@ssw0rd’

pin\_codes = {

'ssl\_admin': '159753486',

'ssl\_worker': '951753426'

}

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/login/<role>', methods=['GET', 'POST'])

def login(role):

error\_message = None

if request.method == “POST”:

pin = request.form.get('pin')

user = 'ssl\_admin' if role == 'admin' else 'ssl\_worker'

if pin\_codes.get(user) == pin:

session['role'] = role

return redirect(url\_for(role))

else:

error\_message = "Authorization failed!"

return render\_template('login.html', role=role, error\_message=error\_message)

@app.route('/admin')

def admin():

if session.get('role') == 'admin':

return render\_template('admin.html')

return redirect(url\_for('index'))

@app.route('/worker')

def worker():

if session.get('role') == 'worker':

return render\_template('worker.html')

return redirect(url\_for('index'))

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=443, ssl\_context=('/etc/ssl/flask\_auth/flask.crt', '/etc/ssl/flask\_auth/flask.key'))

**index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Choosing a role</title>

</head>

<body>

<h2>Choose a role:</h2>

<ul>

<li><a href="{{ url\_for('login', role='admin') }}">Click here for admin</a></li>

<li><a href="{{ url\_for('login', role='worker') }}">Click here for worker</a></li>

</ul>

</body>

</html>

**login.htnl:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Authorization of the {{ role }}</title>

</head>

<body>

<h2>Authorization {{ role }}</h2>

<form method="post">

<label for="pin">Enter the PIN code:</label>

<input type="password" name="pin" required>

<button type="submit">Enter</button>

</form>

{% if error\_message %}

<p style="color:red;">{{ error\_message }}</p>

{% endif %}

</body>

</html>

**admin.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Administrator</title>

</head>

<body>

<h2>Hello Admins!</h2>

</body>

</html>

**worker.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Worker</title>

</head>

<body>

<h2>Hello Workers!</h2>

</body>

</html>

Запускаем приложение:

*python3 app.py*

Добавляем в автозагрузку:

*nano /etc/systemd/system/flask-app.service:*

*[Unit]*

*Description=Flask Application*

*After=network.target*

*[Service]*

*User=root*

*WorkingDirectory=/flask\_site*

*ExecStart=/flask\_site/venv/bin/python3 /flask\_site/app.py*

*Restart=always*

*[Install]*

*WantedBy=multi-user.target*

*systemctl enable flask-app*

*systemctl start flask-app*

**ЕСЛИ АВТОРИЗАЦИЯ ПО ЛОГИНУ И ПИН КОДУ:**

**Меняем app.py:**

*from flask import Flask, render\_template, request, redirect, url\_for, session*

*app = Flask(\_\_name\_\_)*

*app.secret\_key = ‘P@ssw0rd’*

*pin\_codes = {*

*'ssl\_admin': '159753486',*

*'ssl\_worker': '951753426'*

*}*

*@app.route('/')*

*def index():*

*return render\_template('index.html')*

*@app.route('/login/<role>', methods=['GET', 'POST'])*

*def login(role):*

*error\_message = None*

*if request.method == “POST”:*

*username = request.form.get('username')*

*pin = request.form.get('pin')*

*user = None*

*if role == 'admin' and username == 'ssl\_admin':*

*user = 'ssl\_admin'*

*elif role == 'worker' and username == 'ssl\_worker':*

*user = 'ssl\_worker'*

*if user and pin\_codes.get(user) == pin:*

*session['role'] = role*

*return redirect(url\_for(role))*

*else:*

*error\_message = "Authorization failed!"*

*return render\_template('login.html', role=role, error\_message=error\_message)*

*@app.route('/admin')*

*def admin():*

*if session.get('role') == 'admin':*

*return render\_template('admin.html')*

*return redirect(url\_for('index'))*

*@app.route('/worker')*

*def worker():*

*if session.get('role') == 'worker':*

*return render\_template('worker.html')*

*return redirect(url\_for('index'))*

*if \_\_name\_\_ == '\_\_main\_\_':*

*app.run(host='0.0.0.0', port=443, ssl\_context=('/etc/ssl/flask\_auth/flask.crt', '/etc/ssl/flask\_auth/flask.key'))*

**Меняем login.html:**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<title>Authorization of the {{ role }}</title>*

*</head>*

*<body>*

*<h2>Authorization {{ role }}</h2>*

*<form method="post">*

*<label for="username">Enter the user's name:</label>*

*<input type="text" name="username" required><br><br>*

*<label for="pin">Enter the PIN code:</label>*

*<input type="password" name="pin" required><br><br>*

*<button type="submit">Enter</button>*

*</form>*

*{% if error\_message %}*

*<p style="color:red;">{{ error\_message }}</p>*

*{% endif %}*

*</body>*

*</html>*

**Apache2 + PHP:**

*apt install php libapache2-mod-php apache2 apache2-utils*

Сертификат и ключ положить в /etc/apache2/ssl – atom.key и atom.crt

*mkdir -p /var/www/atom*

*cd /var/www/secure\_site*

**nano index.html:**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<title>Choosing a role</title>*

*</head>*

*<body>*

*<h2>Choose a role:</h2>*

*<a href="login.html?role=admin">Click here for admin</a><br>*

*<a href="login.html?role=worker">Click here for worker</a>*

*</body>*

*</html>*

*nano login.html:*

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<title>Authorization</title>*

*</head>*

*<body>*

*<h2>Enter your username and PIN code</h2>*

*<form action="verify.php" method="post">*

*<input type="hidden" name="role" value="<?php echo htmlspecialchars($\_GET['role']); ?>">*

*Login: <input type="text" name="username" required><br><br>*

*PIN code: <input type="password" name="password" required><br><br>*

*<input type="submit" value="Enter">*

*</form>*

*</body>*

*</html>*

**nano admin\_success.html:**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<title>Admin</title>*

*</head>*

*<body>*

*<h2>Hello Admins!</h2>*

*</body>*

*</html>*

**nano worker\_success.html:**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<title>Worker</title>*

*</head>*

*<body>*

*<h2>Hello Workers!</h2>*

*</body>*

*</html>*

**nano verify.php:**

*<?php*

*session\_start();*

*$correct\_logins = [*

*'ssl\_admin' => '159753486',*

*'ssl\_worker' => '951753426'*

*];*

*$username = $\_POST['username'] ?? '';*

*$password = $\_POST['password'] ?? '';*

*if (isset($correct\_logins[$username]) && $correct\_logins[$username] === $password) {*

*if ($username === 'ssl\_admin') {*

*header('Location: admin\_success.html');*

*} else {*

*header('Location: worker\_success.html');*

*}*

*exit;*

*} else {*

*echo "Authorization failed";*

*}*

**nano /etc/apache2/sites-available/atom.conf:**

*<VirtualHost \*:443>*

*ServerAdmin webmaster@localhost*

*DocumentRoot /var/www/atom*

*ServerName atom.local*

*SSLEngine on*

*SSLCertificateFile /etc/apache2/ssl/atom.crt*

*SSLCertificateKeyFile /etc/apache2/ssl/atom.key*

*<Directory /var/www/atom>*

*Options Indexes FollowSymLinks*

*AllowOverride None*

*Require all granted*

*</Directory>*

*ErrorLog ${APACHE\_LOG\_DIR}/error.log*

*CustomLog ${APACHE\_LOG\_DIR}/access.log combined*

*</VirtualHost>*

Все подключаем и запускаем:

*a2enmod ssl*

*a2enmod php*

*a2ensite atom.conf*

*systemctl restart apache2*