Оглавление

[**1.** **Сервер 192.168.59.182 – METIS** Основное по работе SQL Server Management Studio (SSMS) 4](#_Toc123194913)

[ **Добавление пользователей для входа** 4](#_Toc123194914)

[ **Изменение порта и добавление правил в брандмауэра** 5](#_Toc123194915)

[ **Создание базы данных** 5](#_Toc123194916)

[ **Настройка SQL Сервера** 6](#_Toc123194917)

[ **Перенос базы данных** 8](#_Toc123194918)

[ **Бэкап базы и восстановление** 9](#_Toc123194919)

[ **Пример настройки Бэкапирования всех баз** 13](#_Toc123194920)

[ **Удаление устаревших файлов по расписанию** 15](#_Toc123194921)

[ **Добавление базы в кластер** 15](#_Toc123194922)

[ **Создание базы и добавление в кластер средствами 1С** 16](#_Toc123194923)

[ **Кластеры и консоли кластеров серверов 1С** 18](#_Toc123194924)

[ **Основное** 18](#_Toc123194925)

[ **Установка и подключение консоли MMC** 19](#_Toc123194926)

[ **Возможные ошибки , проблемы: Этот хост неизветен** 20](#_Toc123194927)

[ **Настройка выгрузки Бэкапов при помощи Effector server 3 на сетевой диск Gertruda** 20](#_Toc123194928)

[ **Отчистка пользовательского кэша 1С на РДП серверах** 21](#_Toc123194929)

[**2.** **Сервер 192.168.1.131 – 1CR** 21](#_Toc123194930)

[ **Основное** 21](#_Toc123194931)

[ **Настройка планировщика Cron** 22](#_Toc123194932)

[ **Настройка Ядра ОС Linux** 22](#_Toc123194933)

[**3.** **Сервер 192.168.1.115 – PGseк** 28](#_Toc123194934)

[ **Основное** 28](#_Toc123194935)

[ **Настройка планировщика Cron** 29](#_Toc123194936)

[ **Работа с Postgres** 29](#_Toc123194937)

[ **Настройка Ядра ОС Linux** 31](#_Toc123194938)

[**4.** **Сервер 192.168. 2.16 – 1C-Br** 37](#_Toc123194939)

[**** **Основное** 37](#_Toc123194940)

[**** **Настройка планировщика Cron** 38](#_Toc123194941)

[**** **Публикация базы** 38](#_Toc123194942)

[**** **Настройка Ядра ОС Linux** 39](#_Toc123194943)

[**5.** **Сервер 192.168. 1.109 – 1C** 45](#_Toc123194944)

[ **Основное** 45](#_Toc123194945)

[ **Настройка Ядра ОС Linux** 45](#_Toc123194946)

[**6.** **Linux , решение типовых задач** 50](#_Toc123194947)

[ **Изменение настроек Ядра ОС Linux. Сделать так, чтобы они не изменялись при перезапуске сервера.** 51](#_Toc123194948)

[ **Добавление репозитария** 51](#_Toc123194949)

[ **Настройка сети** 52](#_Toc123194950)

[ **Sudo доступ без ввода пароля** 53](#_Toc123194951)

[ **Монтирование сетевого диска** 53](#_Toc123194952)

[**7.** **Установка и настройка** 53](#_Toc123194953)

[1. **Установка и настройка Postgres12 под 1C на Astra Linux** 53](#_Toc123194954)

[2. **Установка и настройка 1C-сервера на Astra Linux** 54](#_Toc123194955)

[3. **Установка и настройка 1C-сервера версии 8.3.21 и выше на Astra Linux** 55](#_Toc123194956)

[4. **Установка и настройка Zabbix-сервера + Postgres 14.5 + Apache на Ubuntu Linux 18.04** 57](#_Toc123194957)

[ **Установка Postgres 14.5 на Ubuntu 18.4** 57](#_Toc123194958)

[ **Установка Апач** 58](#_Toc123194959)

[ **Установка Zabbix сервер** 60](#_Toc123194960)

[ **Настройка оповещения в Telegram чат** 60](#_Toc123194961)

[ **Пример установки Zabbix agent 6.2 на хосте Ubuntu 20.04** 62](#_Toc123194962)

[5. **Установка и настройка Ansible на Ubuntu Linux 18.04** 63](#_Toc123194963)

[ **Установка Ansible** 63](#_Toc123194964)

[ **Настройка подключения к хосту по shh key** 64](#_Toc123194965)

[ **Администрирование Postgres-12-1C на Astra Linux, особенности:** 67](#_Toc123194966)

[ **Пример работы с Postgres через Ansible используя плейбуки** 68](#_Toc123194967)

[ **Пример написания плебука для установки 1С Сервер версии 8.3.21+** 69](#_Toc123194968)

[6. **Установка Git и подключение репозитория с данным гайдом** 72](#_Toc123194969)

[7. **Установка и настройка VPN сервера (OpenVPN) ОС Ubuntu** 73](#_Toc123194970)

[8. **Настройка подключения к удаленному рабочему столу с Windows на Linux Ubuntu** 75](#_Toc123194971)

[9. **Монтирование сетевого диска , настройка бэкапов на Postgres и планировщика Cron** 77](#_Toc123194972)

[ **Подготовка** 77](#_Toc123194973)

[ **Монтируем сетевую папку** 78](#_Toc123194974)

[ **Настройка Бэкапов баз на Субд сервере Postgres** 78](#_Toc123194975)

[ **Настройка планировщика Cron** 80](#_Toc123194976)

[10. **Настройка route . Задача : чтобы интернет работал , только при подключении по ВПН** 80](#_Toc123194977)

[11. **Установка Docker и Docker-compose на Astra Linux** 83](#_Toc123194978)

[ **Простая установка и удаление** 83](#_Toc123194979)

[ **Установка в изолированном сегменте** 84](#_Toc123194980)

[ **Установка актуальных версий** 84](#_Toc123194981)

[ **Установка Docker Compose v.1.29** 85](#_Toc123194982)

[12. **Установка 1С Server + Postgres + Web Server (Apache) в Docker-Compose** 85](#_Toc123194983)

[ **Предварительная настройка** 86](#_Toc123194984)

[ **Описание для будущего билда docker-compose.yml** 87](#_Toc123194985)

[ **Загрузка снимком (images) и запуск проекта** 89](#_Toc123194986)

[13. **Сервер управления и мониторинга. Ubuntu 20.04 Ansible + Zabbix в Docker-Compose** 90](#_Toc123194987)

[ **Предварительные работы** 90](#_Toc123194988)

[ **Установка Ansible** 91](#_Toc123194989)

[ **Установка Git** 91](#_Toc123194990)

[ **Установка Docker** 92](#_Toc123194991)

[ **Установка Docker-compose** 92](#_Toc123194992)

[ **Установка Zabbix в Docker-compose** 93](#_Toc123194993)

[ **Настройка systrmd сервиса для автозапуска контенеров** 93](#_Toc123194994)

[ **Настройка Агента zabbix** 94](#_Toc123194995)

[14. **Установка GLPI в Docker-compose** 96](#_Toc123194996)

[**mariadb.env** 96](#_Toc123194997)

[**docker-compose .yml** 96](#_Toc123194998)

[**7.** **Настройка серверов** 97](#_Toc123194999)

[ **Настройка AD DNS Сервера** 97](#_Toc123195000)

[ **Если сервер находится в корпоративной сети со своим DNS** 97](#_Toc123195001)

[**8.** **Гипервизор VMware vSphere ESXi** 97](#_Toc123195002)

[ **Установка и настройка** 97](#_Toc123195003)

[ **Установка и настройка vCenter** 97](#_Toc123195004)

[ **VMware vCenter Converter Standalone для копирования виртуальных машин** 98](#_Toc123195005)

1. **Сервер 192.168.59.182 – METIS** Основное по работе SQL Server Management Studio (SSMS)
   * **Добавление пользователей для входа**



Безопасность(Security)-Имена для входа(Logins)-Создать имя для входа(New Logins)

Search – Advanced Найти нужного пользователя в домене , затем задать роль (Server Roles)

Список ролей :



* + **Изменение порта и добавление правил в брандмауэра**

Поиск и запуск SQL Server Configuration Manager : Screenshot_1

Здесь отображаются запущенные службы 

А так же протоколы :

Shared Memory для работы через оперативную память локально , дает прирост производительности .

TCP/IP для работы по сети , тут можно посмотреть и задать порт



\*Для данного порта требуется создать правило в Брандмауэр Защитника Windows

\*Поиск и запуск Windows Firewall и добавить правило для данного порта на подключение

* + **Создание базы данных**



Initial Size (Начальный размер ) , желательно сразу задать примерный размер базы , чтобы база не была сильно фрагментирована , так же можно задать шаг прироста базы и лимиты



* + **Настройка SQL Сервера**



В разделе Memory (память ) нужно обязательно ограничивать максимальную память сервера иначе он заберет всю доступную

****

Во вкладке Connections(соединения) можно настроить количество одновременных соединений и доступ удаленного соединения с сервером



Во вкладке Database Settings можно настроить место хранения данных , журнала и бэкапов . Для повышения производительности рекомендуют использовать разные диски



В данном случае все хранится на одном :

C:\Program Files\Microsoft SQL Server\MSSQL11.SQL\_SERVER\MSSQL\DATA\

C:\Program Files\Microsoft SQL Server\MSSQL11.SQL\_SERVER\MSSQL\Backup

* + **Перенос базы данных**

Все действующие базы будут заняты службой SQL сервера и для переноса требуется сначала ее отсоеденить



Tasks(Задачи)-Detach(Отсоединение)



После этого можно переносить фалы в любую другую директорию или даже на другой сервер .

Для того ,чтобы присоединить базу выбираем Attach(Присоеденить)



Выбираем нужный каталог и файл mdf



Автоматически подтягивает и файл журнала транзакций



* + **Бэкап базы и восстановление**

Рассмотрим на примере задачи : Требуется сделать бэкап базы NPOL и загрузить его в пустую базу TikhonovTestDB созданную ранее .



Tasks– Back Up



Проверяем ,что база выбрана верно – NPOL , а тип бэкапа – полный Full . Так же смотрим директорию или выбираем ,куда будет выгружен сам бэкап .



Выбираем базу ,в которую будет загружен бэкап (TikhonovTestBD) Tasks-Restore-Database



Во вкладке General выбираем Divace ( расположение бэкапа ) то ,откуда будет загружаться данные в базу . Проверяем ,что база в Destination выбрана верно и переходим во вкладку Files



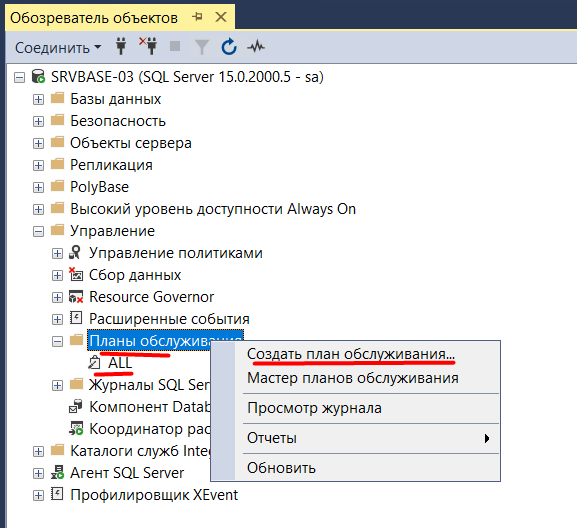
Здесь внимательно указываем mdf b log файлы базы ,в корую будут выгружаться данные из бэкапа ( TikhonovTestDB.mdf и TikhonovTestDB.log )



Переходим в Options и выставляем пункты, как на скриншоте, затем перепроверяем все и можно приступать к процессу загрузки



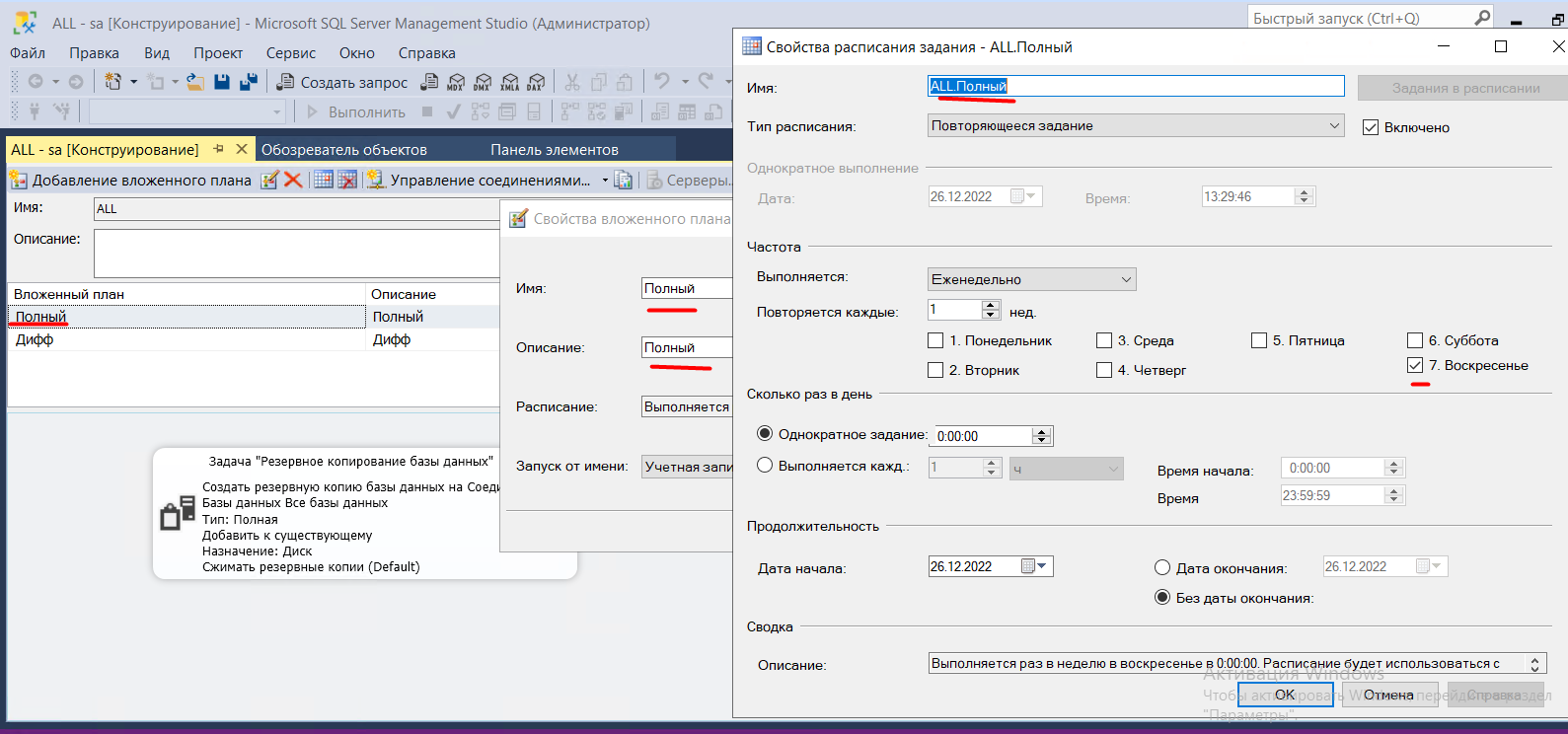
* + **Пример настройки Бэкапирования всех баз**



**Планы обслуживания->Создать план обслуживания**

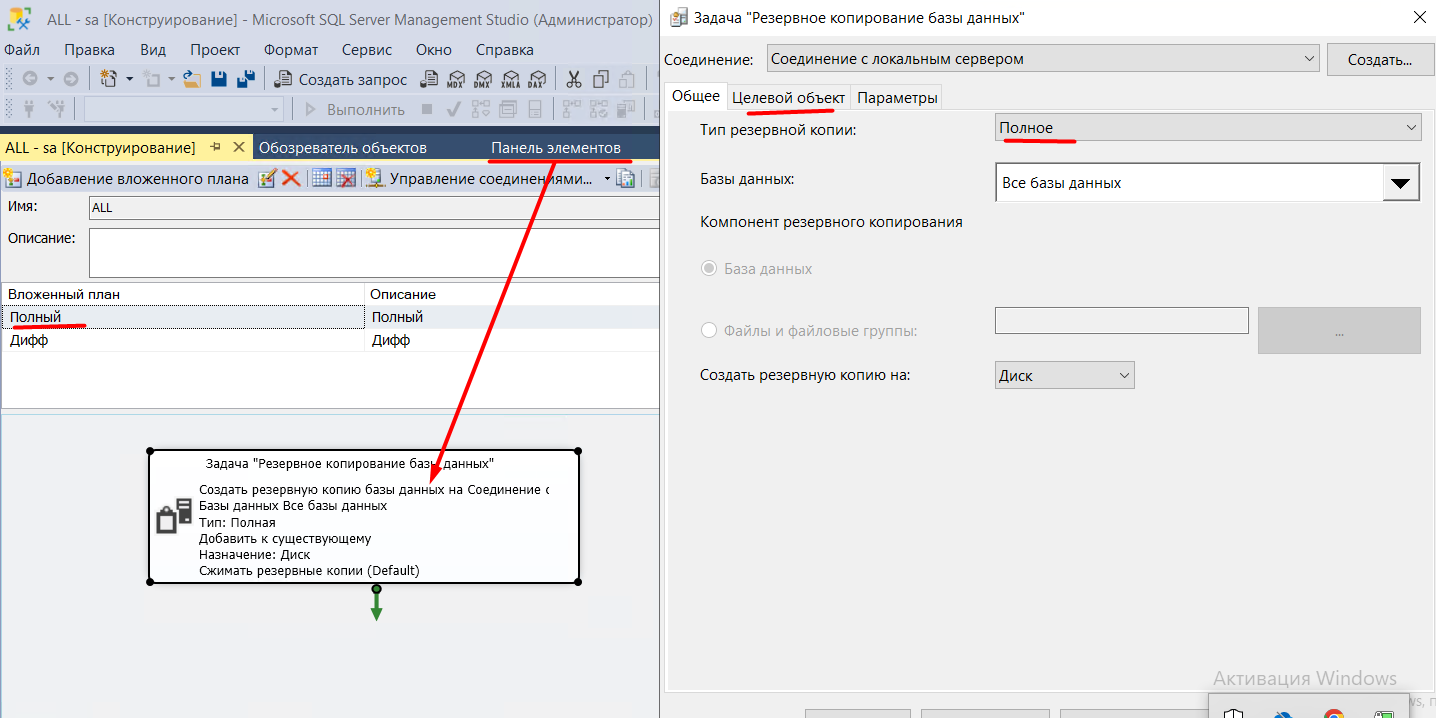
В примере создали план с названием ALL

Заходим в созданный план и добавляем вложенные планы : для полного бэкапа всех баз раз в неделю в Воскресенье (Полный) и для диффирициорованного (только изменения – Дифф) выполняемого во все дни , кроме Воскресенья

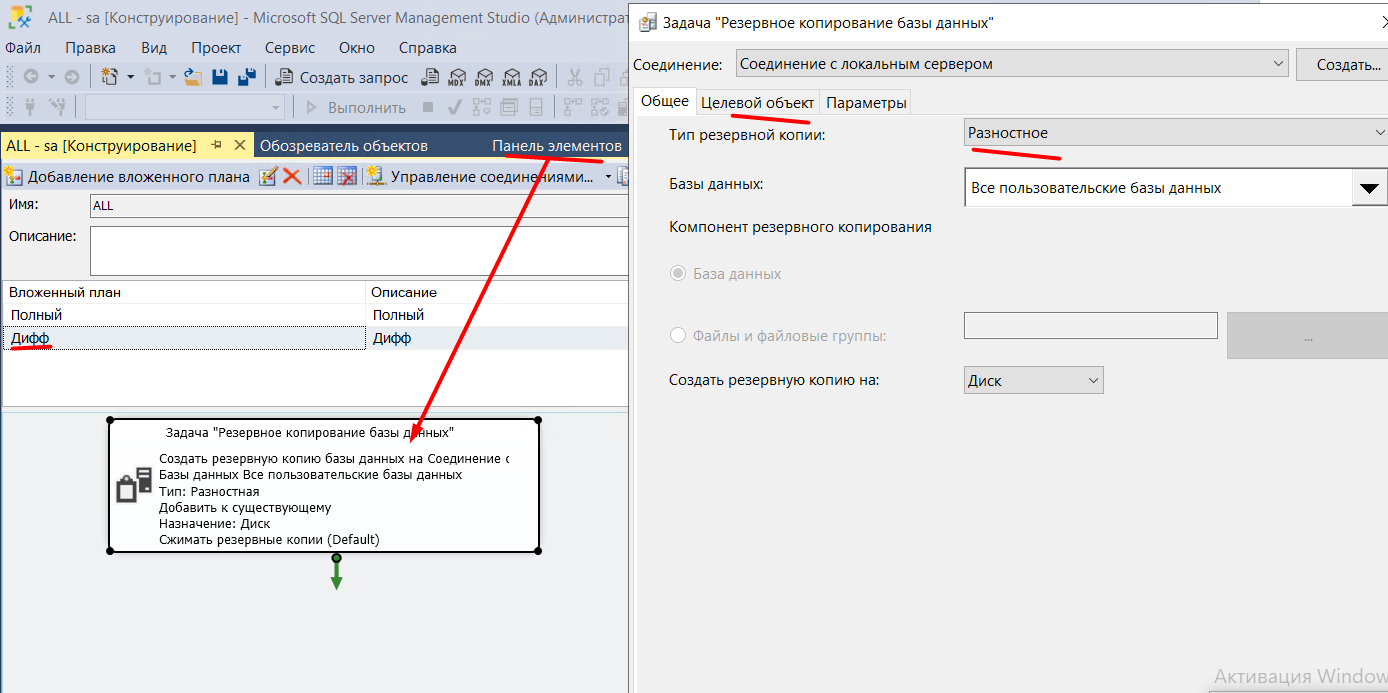


Для каждого из планов вытаскиваем из **Панель элементов**  задачу **Резервное копирование баз данных**

Затем производим настройку данной задачи , для плана – **Полный** тип будет **Полное** и **Все базы данных** , во вкладке **Целевой объект** указываем формат **bak** и путь к папке , где будут храниться бэкапы



Затем производим настройку данной задачи , для плана – **Дифф** тип будет **Разностное** и **Все пользовательские базы данных** , во вкладке **Целевой объект** указываем формат **diff** и путь к папке , где будут храниться бэкапы



* + **Удаление устаревших файлов по расписанию**

Что нужно сделать?

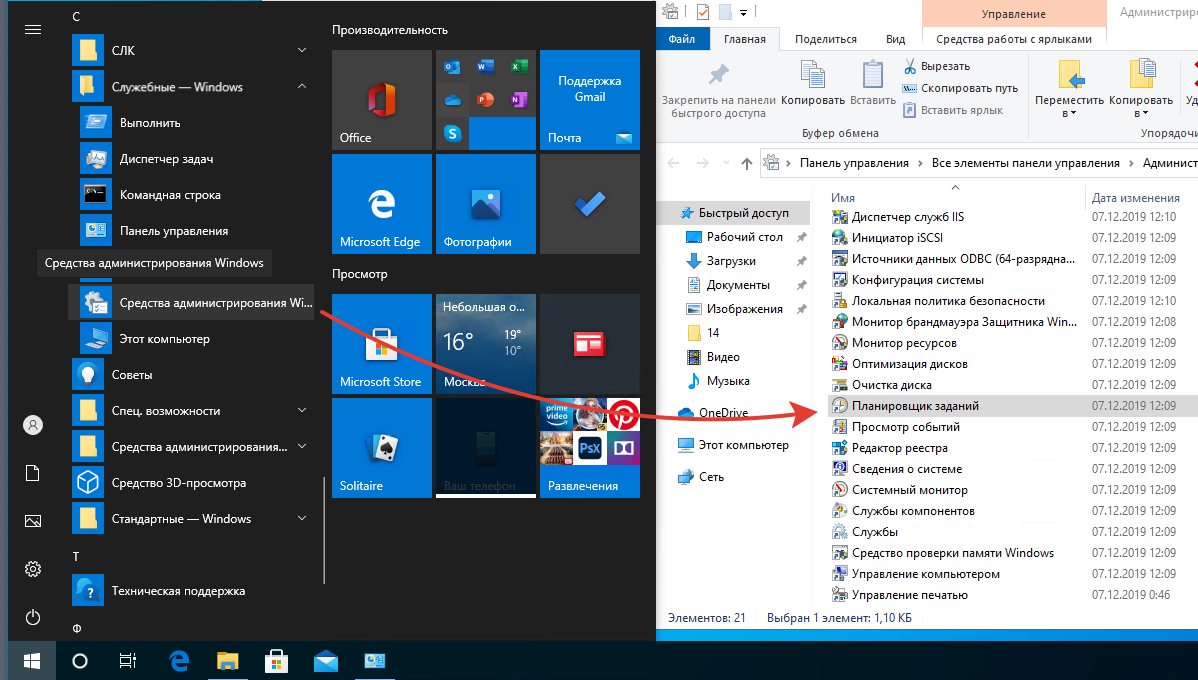
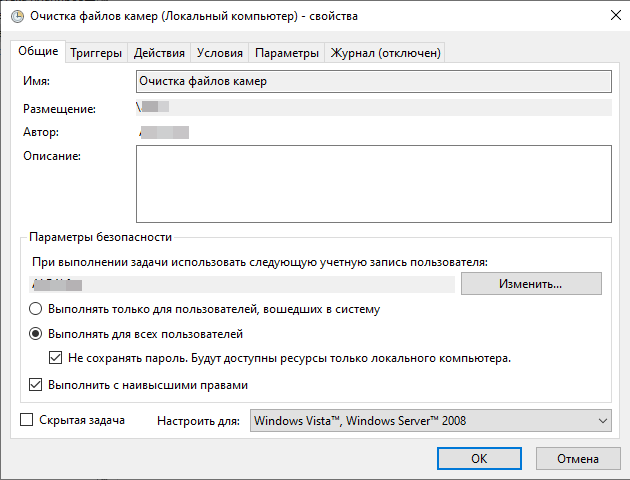
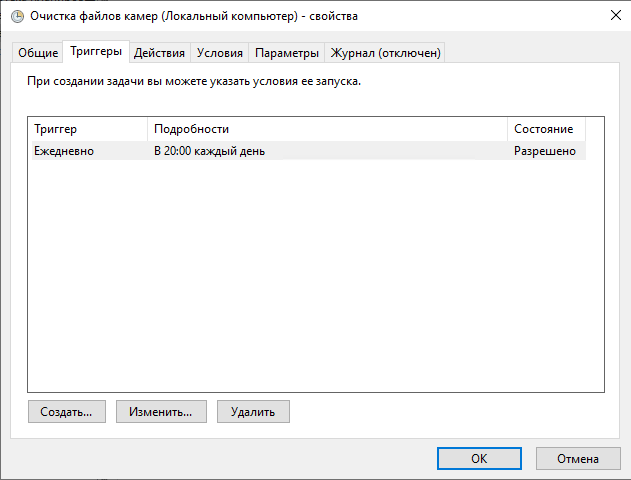
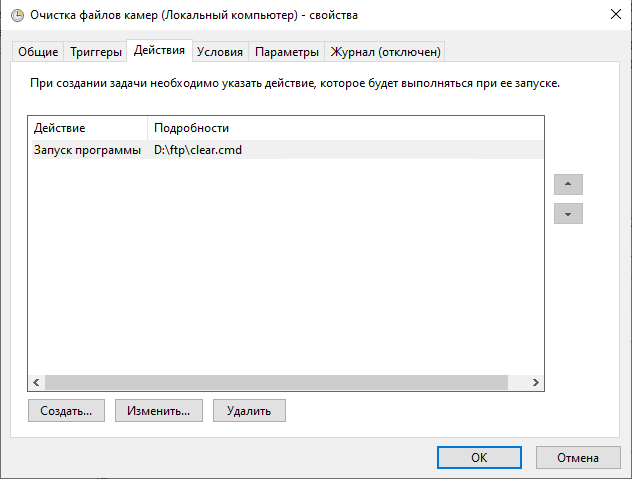
1. Создадим файл **clear.cmd**, рядом с папкой которую нам необходимо удалить (не внутри, а рядом) в каталог **D:\ftp**2. В файле **clear.cmd** добавим следующий текст:

**@echo off**

**SET Path2Del=D:\ftp\CAMERA**

**forfiles -p "%Path2Del%" -s -m \*.\* -d -15 -c "cmd /c del /q @path"**

**for /f %%D in ('DIR cd "%Path2Del%" /AD/B/S ^| sort /r') do RD "%%D"**

3. Там где Path2Del замените на свой путь, параметр *-d -15* означает, что необходимо удалить файлы старше *15*дней. Если хотите увеличить или уменьшить этот срок замените *15* на нужное число.  
4. Откройте меню Пуск > Служебные Windows > Средства администрирование Windows > Планировщик заданий  
  
5. Создадим в планировщике заданий новое задание:  
  
  
  
6. После этого каждый день, в 20:00 будет произведено удаление файлов старше 15 дней.

* + **Добавление базы в кластер**

В консоли кластера добавляем новую базу



Имя желательно должно совпадать с именем базы на сервере , пользователь БД ,должен иметь права администратора



Проверяем , база должна появиться в списке на кластере



* + **Создание базы и добавление в кластер средствами 1С**

Через 1С есть возможность создать новую базу данных и сразу добавить ее в кластер серверов 1С и создать новую базу в СУБД



При необходимости можно создать базу по шаблону с определенными конфигурациями.

Необходимые конфигурации можно загрузить с Портала 1С: ИТС https://releases.1c.ru/total



Указываем кластер серверов 1С , имя информационной базы в кластере ( будет отображаться в консоли кластера серверов 1С , рекомендуется ,чтобы совпадало с именем информационной базы) , Сервер баз данных и тип СУБД , пользователь бд должен обладать необходимыми правами ,для создания базы данных на сервере СУБД.



* + **Кластеры и консоли кластеров серверов 1С**
    - * **Основное**

Используемые порты:

1560-1591 - для рабочего процесса

1541 - для менеджера кластера

1540 - для агента сервера

Кластеры:

Metis – 192.168.59.182 (консоль)

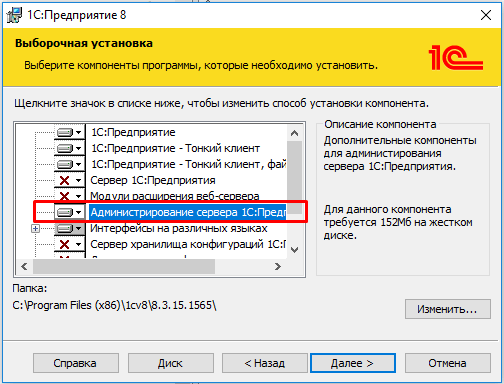
192.168.1.109 – 192.168.1.116 (консоль)

192.168.1.131 – 192.168.1.114 (консоль)

01URPAPPDEV – 10.1.6.5 (консоль)

* + - * **Установка и подключение консоли MMC**

При установке платформы 1С выбираем компонент: Администрирование сервера 1С



Расположение по итогу установки: C:\Program Files (x86)\1cv8\common

**1CV8 Servers.msc**

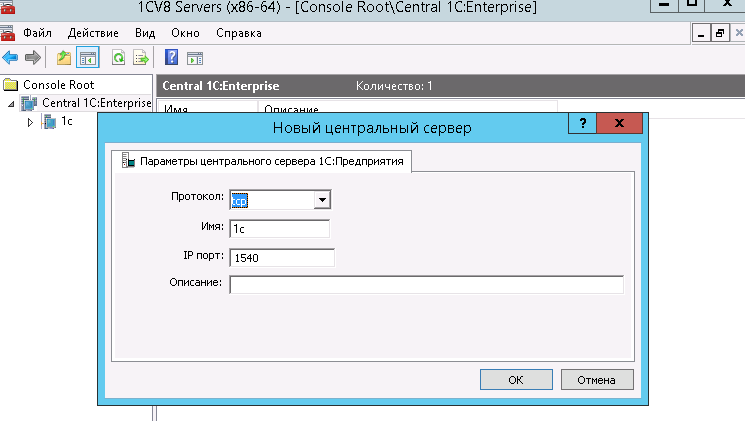
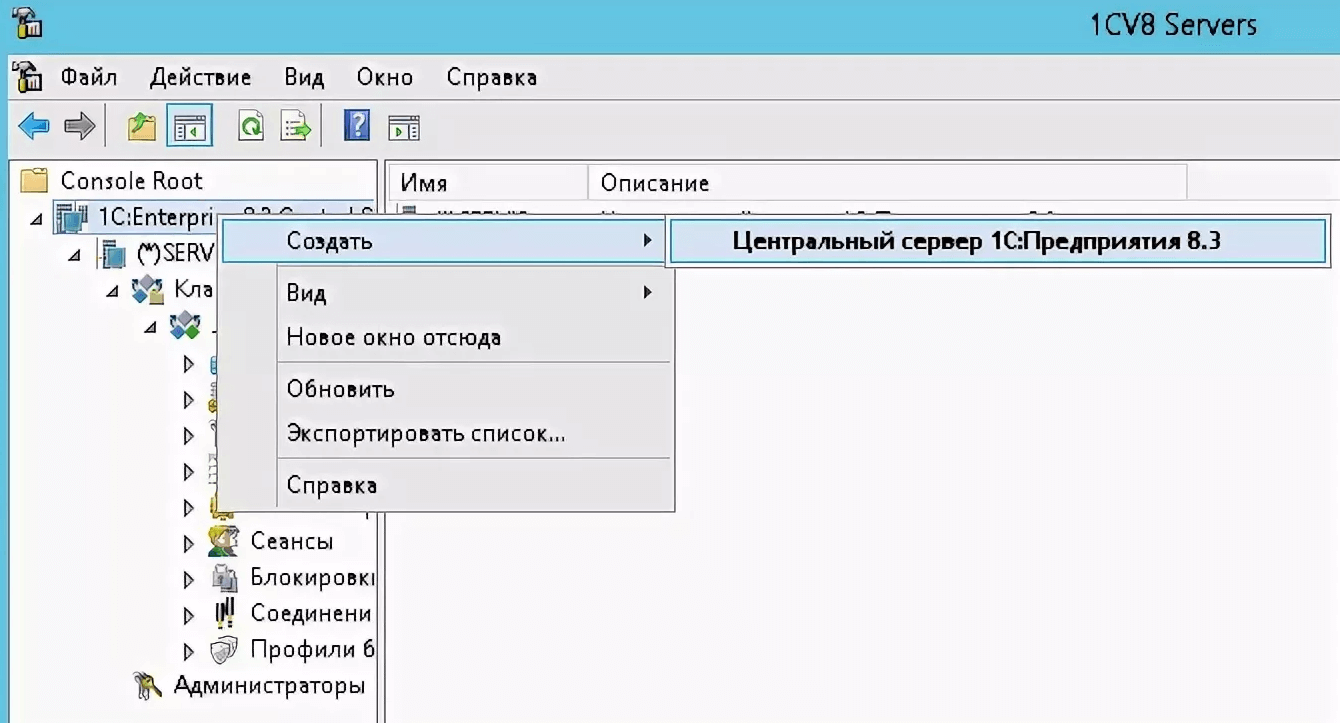
Перед запуском требуется регистрация утилиты.

В поиске: **Регистрация утилиты администрирования серверов**

Примерное место расположения:

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\1C Предприятие 8\Дополнительно\8.3.\*\*.\*\*\*\*

Запускаем с правами администратора, затем можно запускать **1CV8 Servers.msc**

****

Подробнее в материале:

<https://www.koderline.ru/expert/instruktsii/article-konsol-klasterov-v-1s/#jakor1>

<https://infostart.ru/1c/articles/307973/>

* + - * **Возможные ошибки , проблемы: Этот хост неизветен**

Изображение выглядит как текст

Автоматически созданное описание

При попытки подключиться к базе определенного кластера ( в данном случае кластера 192.168.1.109 -1С ) выдает подобное окно с ошибкой.

Следует проверить в C:\Windows\System32\drivers\etc файл hosts , прописан ли там сервер:

192.168.1.109 1C

Подробнее о данной проблеме и решении можно почитать тут : <https://buhexpert8.ru/obuchenie-1s/oshibki-v-1s/etot-host-neizvesten-1s-kak-ispravit.html>

* + **Настройка выгрузки Бэкапов при помощи Effector server 3 на сетевой диск Gertruda**

В **Effector server 3** на Метис настроено 6 ежедневных задач:

1. 1CRstor – забирает содержимое \1CRStor\_backup на сервере 192.168.1.116 и отправляет на сетевой диск Gertruda - 1cbkp\GZ\Repo1cr , архивы хранятся за последние 7 дней. Предварительно на сервере 192.168.1.131 через планировщик cron скрипт делает бэкап конфигураций в \backup (смонтировано в \\192.168.1.116\1CRStor\_backup)
2. , а затем зачищает содержимое \backup ( после выгрузки на Гертруду) . Данные настройки подробнее будут рассматриваться в разделе по серверу 192.168.1.131 \*
3. CopyBCPGZ - забирает содержимое x:\backupPG на сервере 192.168.1.116 и отправляет на сетевой диск Gertruda - 1cbkp\GZ\Daily , архивы хранятся за последние 7 дней. Предварительно на сервере 192.168.1.115 через планировщик cron скрипт делает бэкап конфигураций в \mnt\backupPG (смонтированный сет.диск 192.168.1.116 ого сервера) , а затем зачищает содержимое \mnt\backupPG ( после выгрузки на Гертруду) . Данные настройки подробнее будут рассматриваться в разделе по серверу 192.168.1.115
4. CopyStorageGZ - забирает содержимое x:\1CStorage на сервере 192.168.1.116 и отправляет на сетевой диск Gertruda - 1cbkp\GZ\Repo , архивы хранятся за последние 7 дней.
5. File\_1C\_Backup – Запущено две задачи по бэкапу баз на Метисе : File\_1C\_Backup\_Typolev\_Mainbase и File\_1C\_Backup\_ ITIL\_PROFF1 , бэкапы хранятся 7 дней на сетевом диске Gertruda - 1cbkp\File\_1C\ Typolev\_Mainbase и 1cbkp\File\_1C\ITIL\_PROFF1 .
6. Repository\_Backup – Бэкап репозитария с Метис раз в неделю ,хранятся за 7 последних недель на сетевом диске Gertruda - 1cbkp\Repositories



* + **Отчистка пользовательского кэша 1С на РДП серверах**

Создаем текстовый документ .txt с содержанием :

@FOR /D %%i in ("C:\Users\\*") do (

@FOR /D %%j in ("%%i\Local settings\Application data\1C\1Cv8\????????-????-????-????-????????????") do rd /s /q "%%j"

@FOR /D %%j in ("%%i\AppData\Roaming\1C\1Cv8\????????-????-????-????-????????????") do rd /s /q "%%j"

)

Сохраняем в формате .bat

Запускать от имени администратора

1. **Сервер 192.168.1.131 – 1CR**
   * **Основное**

|  |  |
| --- | --- |
| **Основные директории сервера** | |
| /etc/init.d/ | Хранятся скрипты |
| /backup | Хранятся бэкапы за последние сутки |
| /stor1c | Хранилище конфигураций |
| /var/log/cronlog/ | Логи по отработанным скриптам планировщика cron |
| /var/spool/cron/crontabs | Все настройки cron хранятся тут |

Бэкапы конфигураций раз в день , в 6 50 полностью удаляются, занимают около 81 Гб на диске в директории /backup



Раздел в раздел /backup смотнирован сетевую папку 192.168.1.116 ого сервера , сетевая папка: **1CRStor\_backup**

|  |  |
| --- | --- |
| **Основные скрипты** | |
| backup.sh | Бэкапы из /store1c в /backup |



* + **Настройка планировщика Cron**



1. В 15:2 каждую субботу отрабатывает команда reboot
2. В 15:0 каждый день отрабатывает скрипт на перезапуск сервера 1С
3. В 25:0 каждый день отрабатывает скрипт (backup.sh) на бэкап конфигураций из /store1c в /backup

*Effector sever 3 на сервере Metis забирает бэкапы в другое сетевое хранилище см р 1.11*

1. Каждый день в 5:50 из директории /backup и удаляет все файлы

Логи по ошибкам находятся в директории /var/log/cronlog/ и называются в соответствии с отработанным скриптом или действием :

reboot.log – ребут

restart1c.log – проверка скрипта на рестарт сервера 1с

backup.log – бэкапы конфигураций раз в день

backuprm.log – удаление всех бэкапов конфигураций

* + **Настройка Ядра ОС Linux**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Наименование | Значение | Откуда | Описание | | Количество процессоров | 12 | cat /proc/cpuinfo | grep processor | wc -l | Количество процессоров, показанное в /proc/cpuinfo, может не соответствовать реальному количеству ядер процессора. Например, процессор с 2 ядрами и гиперпоточностью будет показан как процессор с 4 ядрами. | | Идентификатор ядра | 0,1,2,3,4,5 - (всего 6) | cat /proc/cpuinfo | grep 'core id' | Есть 6 разных идентификатора ядра. Это указывает на то, что существует 4 реальных ядра. | | Данную информацию можно также посомтреть при помощи команды - lscpu. | | | | | Диапазоны портов |  | /proc/ioports | В этом файле содержаться все зарегистрированные диапазоны портов, а также устройства, которые их используют | | MemTotal : доступный объем оперативной памяти | 69066052 kB = 69 Gb | /proc/meminfo | физическая память за вычетом нескольких зарезервированных битов и бинарного кода ядра | | Команда для вывода информации об использовании оперативной памяти - free | | | | | Модуль ядра, или его часто называют драйвером, — это фрагмент кода, расширяющий функциональные возможности ядра.Запустите lsmod из командной строки, чтобы узнать, какие модули ядра загружены в данный момент. | | | | | Точки монтирования |  | /proc/mounts | В этом файле перечислены все точки монтирования и все подключенные файловые системы | | Точную версию ядра | Linux version 4.15.3-1-generic (root@build) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)) #astra21 SMP Thu Aug 22 12:16:21 UTC 201$ | /proc/version | Точная версию ядра, компилятора, и в некоторых случаях, даже дистрибутива | | /proc/pid/ - Файловая система proc состоит не только из файлов, но здесь есть и папки. Больше всего здесь папок с номерами вместо имен. Каждый этот номер означает PID процесса, а эта папка содержит информацию о каждом запущенном в системе процессе. Когда процесс заканчивается, его каталог исчезает из системы | | Основные файлы: cmdline - содержит команду с помощью которой был запущен процесс, а также переданные ей параметры cwd - символическая ссылка на текущую рабочую директорию процесса exe - ссылка на исполняемый файл root - ссылка на папку суперпользователя environ - переменные окружения, доступные для процесса fd - содержит файловые дескрипторы, файлы и устройства, которые использует процесс maps, statm, и mem - информация о памяти процесса stat, status - состояние процесса | С помощью этих файлов вы можете составлять различные скрипты. Например если вы хотите уничтожить все зомби процессы, то вы можете сканировать все директории на наличие Z в файле status. Так же само можно проверить запущена ли нужная вам программа просмотрев все cmdline. | | /proc/sys/ -Эта папка в proc linux не только предоставляет информацию о системе, но и позволяет изменять различные параметры ядра на лету, а также включать дополнительные функции. | | Основные подкаталоги в этой папке: debug - содержит отладочную информацию, она будет вам полезна если вы разработчик ядра dev - параметры различных устройств, подключенных к системе fs - вся информация о файловой системе kernel - позволяет напрямую настраивать ядро net - настройка разных параметров сети vm - взаимодействие с подсистемой vm | Чтобы посмотреть можно ли записывать в файлы используйте команду:   ls -ld /proc/sys/  Если у файла есть флаг W, значит в него можно записывать данные. | | Настройки Ядра | Значение | Путь | Команда | | Если установлено 1, разрешает выполнение 32 битных программ в 64 битной системе | 1 (по умолчанию) | /proc/sys/abi/vsyscall32 | sysctl -w abi.vsyscall32=1 | | При возникновении ошибки в ядре выводить значения регистров процессора и стек вызовов процедур | 1 (по умолчанию) | /proc/sys/debug/exception-trace | sysctl -w debug.exception-trace=1 | | Максимальная частота генерации прерываний от системного таймера High Precision Event Timer (HPET), который пришел на замену таймеру реального времени RTC. | 64 (по умолчанию) | /proc/sys/dev/hpet/max-user-freq | sysctl -w dev.hpet.max-user-freq=64 | | Количество асинхронных операций ввода и вывода в вашей файловой системе | 0 | /proc/sys/fs/aio-nr | sysctl -w fs.aio-nr=0 | | Максимальное количество дескрипторов файлов, которые может создать и обрабатывать ядро. Если вы часто получаете сообщения об ошибке из-за невозможности создать дескриптор файла увеличьте этот лимит. По умолчанию установлено значение 10 % от вашей оперативной памяти. | 6887700 = 6 Gb | /proc/sys/fs/file-max | sysctl -w fs.file-max=6887700 | | Подсистема ядра inotify позволяет следить за изменениями в файловой системе. Этот параметр устанавливает максимальное количество событий, которые могут находиться в очереди, перед тем как их обработает программа. | 16384 | /proc/sys/fs/inotify/max\_queued\_events | sysctl -w fs.inotify.max\_queued\_events=16384 | | Позволяет установить доменное имя NIS (Network Internet Services) и YP (Yellow Pages). Но не путайте это доменное имя с DNS, это совсем разные вещи. | (none) | /proc/sys/kernel/domainname | - | | Имя вашего компьютера. Это самый простой способ изменить имя компьютера прямо сейчас, без перезагрузки. | 1CR | /proc/sys/kernel/hostname | sysctl -w kernel.hostname=Pgser | | Позволяет отключить загрузку модулей ядра. | 0 | /proc/sys/kernel/modules\_disabled | sysctl -w kernel.modules\_disabled=0 | | Рандомизация адресного пространства это функция увеличивающая безопасность вашей системы. Она защищает от атак на переполнение буфера. По умолчанию включена. | 2 - полностью автоматизировано | /proc/sys/kernel/randomize\_va\_space | sysctl -w kernel.randomize\_va\_space=2 | | Позволяет включить или отключить управление ядром с помощью SysRQ | 0 - отключить все функции | /proc/sys/kernel/sysrq | sysctl -w kernel.sysrq=0 | | Максимальное количество запущенных потоков для процессов. | 539279 | /proc/sys/kernel/threads-max | sysctl -w kernel.threads-max=539279 | | Если включено, ядро будет игнорировать все icmp запросы. Рекомендуется для защиты от DDOS атак. | 0 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_all | sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0 | | Так же, как и в предыдущем варианте, только игнорироваться будут только широковещательные icmp запросы | 1 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts | sysctl -w net.ipv4.icmp\_echo\_ignore\_broadcasts=1 | | Максимальное количество узлов, через которые может пройти отправленный пакет перед тем, как достигнет цели. | 64 | /proc/sys/net/ipv4/ip\_default\_ttl | sysctl -w net.ipv4.ip\_default\_ttl=64 | | Разрешить проходящие пакеты через этот компьютер. Обычно такая настройка параметров ядра Linux нужна для роутеров | 0 | /proc/sys/net/ipv4/ip\_forward | sysctl -w net.ipv4.ip\_forward=0 | | Диапазон локальных портов, которые могут быть использованы вашими программами | 32768 60999 (по умолчанию) | /proc/sys/net/ipv4/ip\_local\_port\_range | sysctl -w net.ipv4.ip\_local\_port\_ranged= "32768 60999" | | Установите 1 чтобы защитить компьютер от атаки TCP TimeWait. | 0 | /proc/sys/net/ipv4/tcp\_rfc1337 | sysctl -w net.ipv4.tcp\_rfc1337=0 | | Таймаут ожидания завершения соединения после отправки пакета FIN. Рекомендовано 15. | 60 | /proc/sys/net/ipv4/tcp\_fin\_timeout | sysctl -w net.ipv4.tcp\_fin\_timeout=60 | | Поддерживать соединение активным определенное время, например, 300 секунд. По истечении этого времени TCP соединение будет разорвано. | 7200 | /proc/sys/net/ipv4/tcp\_keepalive\_time | sysctl -w net.ipv4.tcp\_keepalive\_time=7200 | | Указывает размер по умолчанию буфера для сокета получения данных по сети | 212992 | /proc/sys/net/core/rmem\_default | sysctl -w net.**core.rmem\_default**e=212992 | | Максимальный размер буфера сокета для получения данных по сети | 212992 | /proc/sys/net/core/rmem\_max | sysctl -w net.core.rmem\_max=212992 | | Размер сокета по умолчанию для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_default | sysctl -w net.core.wmem\_defaulte=212992 | | Максимальный размер буфера сокета для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_max | sysctl -w net.core.wmem\_max=212992 | | Количество памяти, доступной для работы TCP. | 4096 87380 6291456 (минимум, по умолчанию, максимум) | /proc/sys/net/ipv4/tcp\_rmem |  | | Указывает процент от общей системной памяти, когда фоновый демон pdflush записи данных начнет переписывать кешированные данные на диск | 10 (Рекомендуется 3) | /proc/sys/vm/dirty\_background\_ratio | sysctl -w vm.dirty\_background\_ratio =10 | | Указывает сколько общей оперативной памяти должно быть занято, чтобы процесс, который ведет запись данных на диск инициировал запись кэшированных данных непосредственно на жесткий диск | 20 (Рекомендуется 10) | /proc/sys/vm/dirty\_ratio | sysctl -w vm.dirty \_ratio =20 | | Устанавливает процент свободной памяти, по достижении которого данные начинают переноситься на swap раздел, для систем с большим количеством памяти рекомендовано значение 10. | 60 (Рекомендуется 10) | /proc/sys/vm/swappiness | sysctl -w vm. swappiness =60 | | Определяет условия разрешения и отказа запросов больших объемов памяти. Возможные значения: 0 (по умолчанию) — ядро использует эвристический алгоритм для расчета перерасхода памяти, принимая во внимание объем доступной памяти и число неверных запросов.  1 — ядро не обрабатывает перерасход памяти. При этом вероятность превышения нагрузки на память возрастает, но в то же время увеличивается производительность задач, активно использующих память. 2 — отказ обработки запросов, запрашивающих память, размер которой превышает суммарный размер памяти пространства подкачки и ОЗУ в соответствии с overcommit\_ratio (По умолчанию равен 50). | 0 (Рекомендуется 2) | /proc/sys/vm/overcommit\_memory | sysctl -w vm. overcommit\_memory =2 | | **Так же для СУБД сервера репомендуется в /etc/systemd/logind.conf указать RemoveIPC=no (RemoveIPC в systemd: отключить высвобождение памяти ОС для УЗ) на данный момент RemoveIPC=yes** | | | | |

1. **Сервер 192.168.1.115 – PGseк**

* **Основное**

|  |  |
| --- | --- |
| **Основные директории сервера** | |
| /usr/local/bin/ | Хранятся скрипты |
| /mnt/backupPG/ | Хранятся бэкапы за последние сутки |
| /var/log/cronlog/  /var/log/cronlog/errolog/ | Логи по отработанным скриптам планировщика cron (Ход выполнения/ошибки) |
| /var/spool/cron/crontabs | Все настройки cron хранятся тут |

Состояние мест хранения бэкапов

Бэкапы баз раз в день , хранятся за последние сутки , занимают 164Гб на сетевом диске размером 200 Гб смонтированным в директорию /mnt/backupPG/



|  |  |
| --- | --- |
| **Основные скрипты** | |
| backup-sql\_new.sh | Бэкапы баз в /mnt/backupPG/ |
| vakuupdb.sh | Ваккум баз и реиндексация всех баз |
| vacuumin131new.sh | Ваккум баз и реиндексация баз кластера 1.131 |

* **Настройка планировщика Cron**



1. В 0:20 каждый день отрабатывает скрипт на бэкап всех баз (backup-sql\_new.sh) в директорию /mnt/backupPG/
2. В 2:15 каждое воскресенье отрабатывает скрипт на ваккум и переиндексацию баз (vakuupdb.sh)
3. В 2:15 каждый четверг отрабатывает скрипт на ваккум и переиндексацию баз кластера 131 (vacuumin131new.sh)
4. Каждый день в 0:10 проверяет директорию /mnt/packupPG/ и удаляет все файлы

Логи по ошибкам находятся в директории /var/log/cronlog/errolog/ и называются в соответствии с отработанным скриптом или действием :

vacuumin131new.log – проверка скрипта на ваккум и реиндексацию наличие ошибок

vakuupdb.log – проверка скрипта на ваккум и реиндексацию наличие ошибок

backup-sql\_new.log – бэкапы раз в день

Логи по ходу выполнения скриптов находятся в директории /var/log/cronlog/

vacuumin131new.log – проверка выполнения скрипта реиндексации и ваккума

* **Работа с Postgres**

Чтобы узнать, какая версия PostgreSQL работает в вашей системе, вызовите команду postgres с параметром —version или  -V:

**postgres --version**

Версию клиентской утилиты psql в PostgreSQL  можно найти с помощью следующей команды:

**psql --version**

**Сервис Postgres: postgrespro-ent-12.service**

* Вход в интерактивный режим на 1.115 (Указание местоположения сокета)

**sudo -u postgres psql**

Если выдает подобную ошибку:



Требуется указать путь к Unix-сокету:

**sudo -u postgres psql --host=/tmp**

Для исправления данной ошибки ,была создана ссылка в /var/run/postgresql/ в /tmp/ ,где лежит наш сокет:

**ln -s /tmp/.s.PGSQL.5432 /var/run/postgresql/.s.PGSQL.5432**

Исправление ошибки в несовпадении версии сервера и psql **:**

**find / -name psql -type f 2>/dev/null**

**ln -s /opt/pgpro/ent-12/bin/psql /usr/bin/psql**

Пересоздание ссылки в /usr/bin/

* Создал скрипт в /home/itihonov/

skbackup.sh

Состав:

#!/bin/sh

# Устанавливаем дату

DATA=`date +"%Y-%m-%d\_%H-%M"`

# Бэкапим базу данных

/opt/pgpro/ent-12/bin/pg\_dump -U postgres GZ\_base67 > /mnt/backupPG/$DATA-GZ\_base67.dump

* Даю права на исполнение :

chmod 744 skbackup.sh

* Исполнение: /home/itihonov/skbackup.sh
* Заливаем бэкап новую базу :

Sudo su postgres psql --host=/tmp gz\_base677 < /mnt/backupPG/2022-05-19\_13-36-GZ\_base67.dump

* Проверка размера базы , размер копии базы может не совпадать с оригинальной , чаще всего он меньше .

SELECT pg\_size\_pretty( pg\_database\_size( 'gz\_base677' ) );

* **Настройка Ядра ОС Linux**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Наименование | Значение | Откуда | Описание | | Количество процессоров | 16 | cat /proc/cpuinfo | grep processor | wc -l | Количество процессоров, показанное в /proc/cpuinfo, может не соответствовать реальному количеству ядер процессора. Например, процессор с 2 ядрами и гиперпоточностью будет показан как процессор с 4 ядрами. | | Идентификатор ядра | 0,1,2,3 - (всего 4) | cat /proc/cpuinfo | grep 'core id' | Есть 4 разных идентификатора ядра. Это указывает на то, что существует 4 реальных ядра. | | Данную информацию можно также посомтреть при помощи команды - lscpu. | | | | | Диапазоны портов |  | /proc/ioports | В этом файле содержаться все зарегистрированные диапазоны портов, а также устройства, которые их используют | | MemTotal : доступный объем оперативной памяти | 72161300 kB = 72 Gb | /proc/meminfo | физическая память за вычетом нескольких зарезервированных битов и бинарного кода ядра | | Команда для вывода информации об использовании оперативной памяти - free | | | | | Модуль ядра, или его часто называют драйвером, — это фрагмент кода, расширяющий функциональные возможности ядра.Запустите lsmod из командной строки, чтобы узнать, какие модули ядра загружены в данный момент. | | | | | Точки монтирования |  | /proc/mounts | В этом файле перечислены все точки монтирования и все подключенные файловые системы | | Точную версию ядра | Linux version 4.15.3-3-generic (builder@build) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)) #astra25+ci7 SMP Wed Oct 28 03:38:36 | /proc/version | Точная версию ядра, компилятора, и в некоторых случаях, даже дистрибутива | | /proc/pid/ - Файловая система proc состоит не только из файлов, но здесь есть и папки. Больше всего здесь папок с номерами вместо имен. Каждый этот номер означает PID процесса, а эта папка содержит информацию о каждом запущенном в системе процессе. Когда процесс заканчивается, его каталог исчезает из системы | | Основные файлы: cmdline - содержит команду с помощью которой был запущен процесс, а также переданные ей параметры cwd - символическая ссылка на текущую рабочую директорию процесса exe - ссылка на исполняемый файл root - ссылка на папку суперпользователя environ - переменные окружения, доступные для процесса fd - содержит файловые дескрипторы, файлы и устройства, которые использует процесс maps, statm, и mem - информация о памяти процесса stat, status - состояние процесса | С помощью этих файлов вы можете составлять различные скрипты. Например если вы хотите уничтожить все зомби процессы, то вы можете сканировать все директории на наличие Z в файле status. Так же само можно проверить запущена ли нужная вам программа просмотрев все cmdline. | | /proc/sys/ -Эта папка в proc linux не только предоставляет информацию о системе, но и позволяет изменять различные параметры ядра на лету, а также включать дополнительные функции. | | Основные подкаталоги в этой папке: debug - содержит отладочную информацию, она будет вам полезна если вы разработчик ядра dev - параметры различных устройств, подключенных к системе fs - вся информация о файловой системе kernel - позволяет напрямую настраивать ядро net - настройка разных параметров сети vm - взаимодействие с подсистемой vm | Чтобы посмотреть можно ли записывать в файлы используйте команду:   ls -ld /proc/sys/  Если у файла есть флаг W, значит в него можно записывать данные. | | Настройки Ядра | Значение | Путь | Команда | | Если установлено 1, разрешает выполнение 32 битных программ в 64 битной системе | 1 (по умолчанию) | /proc/sys/abi/vsyscall32 | sysctl -w abi.vsyscall32=1 | | При возникновении ошибки в ядре выводить значения регистров процессора и стек вызовов процедур | 1 (по умолчанию) | /proc/sys/debug/exception-trace | sysctl -w debug.exception-trace=1 | | Максимальная частота генерации прерываний от системного таймера High Precision Event Timer (HPET), который пришел на замену таймеру реального времени RTC. | 64 (по умолчанию) | /proc/sys/dev/hpet/max-user-freq | sysctl -w dev.hpet.max-user-freq=64 | | Количество асинхронных операций ввода и вывода в вашей файловой системе | 0 | /proc/sys/fs/aio-nr | sysctl -w fs.aio-nr=0 | | Максимальное количество дескрипторов файлов, которые может создать и обрабатывать ядро. Если вы часто получаете сообщения об ошибке из-за невозможности создать дескриптор файла увеличьте этот лимит. По умолчанию установлено значение 10 % от вашей оперативной памяти. | 7197103 = 7Gb | /proc/sys/fs/file-max | sysctl -w fs.file-max=7197103 | | Подсистема ядра inotify позволяет следить за изменениями в файловой системе. Этот параметр устанавливает максимальное количество событий, которые могут находиться в очереди, перед тем как их обработает программа. | 16384 | /proc/sys/fs/inotify/max\_queued\_events | sysctl -w fs.inotify.max\_queued\_events=16384 | | Позволяет установить доменное имя NIS (Network Internet Services) и YP (Yellow Pages). Но не путайте это доменное имя с DNS, это совсем разные вещи. | (none) | /proc/sys/kernel/domainname | - | | Имя вашего компьютера. Это самый простой способ изменить имя компьютера прямо сейчас, без перезагрузки. | PGser | /proc/sys/kernel/hostname | sysctl -w kernel.hostname=Pgser | | Позволяет отключить загрузку модулей ядра. | 0 | /proc/sys/kernel/modules\_disabled | sysctl -w kernel.modules\_disabled=0 | | Рандомизация адресного пространства это функция увеличивающая безопасность вашей системы. Она защищает от атак на переполнение буфера. По умолчанию включена. | 2 - полностью автоматизировано | /proc/sys/kernel/randomize\_va\_space | sysctl -w kernel.randomize\_va\_space=2 | | Позволяет включить или отключить управление ядром с помощью SysRQ | 0 - отключить все функции | /proc/sys/kernel/sysrq | sysctl -w kernel.sysrq=0 | | Максимальное количество запущенных потоков для процессов. | 563454 | /proc/sys/kernel/threads-max | sysctl -w kernel.threads-max=563454 | | Если включено, ядро будет игнорировать все icmp запросы. Рекомендуется для защиты от DDOS атак. | 0 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_all | sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0 | | Так же, как и в предыдущем варианте, только игнорироваться будут только широковещательные icmp запросы | 1 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts | sysctl -w net.ipv4.icmp\_echo\_ignore\_broadcasts=1 | | Максимальное количество узлов, через которые может пройти отправленный пакет перед тем, как достигнет цели. | 64 | /proc/sys/net/ipv4/ip\_default\_ttl | sysctl -w net.ipv4.ip\_default\_ttl=64 | | Разрешить проходящие пакеты через этот компьютер. Обычно такая настройка параметров ядра Linux нужна для роутеров | 0 | /proc/sys/net/ipv4/ip\_forward | sysctl -w net.ipv4.ip\_forward=0 | | Диапазон локальных портов, которые могут быть использованы вашими программами | 32768 60999 (по умолчанию) | /proc/sys/net/ipv4/ip\_local\_port\_range | sysctl -w net.ipv4.ip\_local\_port\_ranged= "32768 60999" | | Установите 1 чтобы защитить компьютер от атаки TCP TimeWait. | 0 | /proc/sys/net/ipv4/tcp\_rfc1337 | sysctl -w net.ipv4.tcp\_rfc1337=0 | | Таймаут ожидания завершения соединения после отправки пакета FIN. Рекомендовано 15. | 60 | /proc/sys/net/ipv4/tcp\_fin\_timeout | sysctl -w net.ipv4.tcp\_fin\_timeout=60 | | Поддерживать соединение активным определенное время, например, 300 секунд. По истечении этого времени TCP соединение будет разорвано. | 7200 | /proc/sys/net/ipv4/tcp\_keepalive\_time | sysctl -w net.ipv4.tcp\_keepalive\_time=7200 | | Указывает размер по умолчанию буфера для сокета получения данных по сети | 212992 | /proc/sys/net/core/rmem\_default | sysctl -w net.**core.rmem\_default**e=212992 | | Максимальный размер буфера сокета для получения данных по сети | 212992 | /proc/sys/net/core/rmem\_max | sysctl -w net.core.rmem\_max=212992 | | Размер сокета по умолчанию для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_default | sysctl -w net.core.wmem\_defaulte=212992 | | Максимальный размер буфера сокета для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_max | sysctl -w net.core.wmem\_max=212992 | | Количество памяти, доступной для работы TCP. | 4096 131072 6291456 (минимум, по умолчанию, максимум) | /proc/sys/net/ipv4/tcp\_rmem |  | | Указывает процент от общей системной памяти, когда фоновый демон pdflush записи данных начнет переписывать кешированные данные на диск | 10 (Рекомендуется 3) | /proc/sys/vm/dirty\_background\_ratio | sysctl -w vm.dirty\_background\_ratio =10 | | Указывает сколько общей оперативной памяти должно быть занято, чтобы процесс, который ведет запись данных на диск инициировал запись кэшированных данных непосредственно на жесткий диск | 20 (Рекомендуется 10) | /proc/sys/vm/dirty\_ratio | sysctl -w vm.dirty \_ratio =20 | | Устанавливает процент свободной памяти, по достижении которого данные начинают переноситься на swap раздел, для систем с большим количеством памяти рекомендовано значение 10. | 60 (Рекомендуется 10) | /proc/sys/vm/swappiness | sysctl -w vm. swappiness =60 | | Определяет условия разрешения и отказа запросов больших объемов памяти. Возможные значения: 0 (по умолчанию) — ядро использует эвристический алгоритм для расчета перерасхода памяти, принимая во внимание объем доступной памяти и число неверных запросов.  1 — ядро не обрабатывает перерасход памяти. При этом вероятность превышения нагрузки на память возрастает, но в то же время увеличивается производительность задач, активно использующих память. 2 — отказ обработки запросов, запрашивающих память, размер которой превышает суммарный размер памяти пространства подкачки и ОЗУ в соответствии с overcommit\_ratio (По умолчанию равен 50). | 0 (Рекомендуется 2) | /proc/sys/vm/overcommit\_memory | sysctl -w vm. overcommit\_memory =2 | | **Так же для СУБД сервера репомендуется в /etc/systemd/logind.conf указать RemoveIPC=no (RemoveIPC в systemd: отключить высвобождение памяти ОС для УЗ) на данный момент RemoveIPC=yes** | | | | |

1. **Сервер 192.168. 2.16 – 1C-Br**

* **Основное**

|  |  |
| --- | --- |
| **Основные директории сервера** | |
| /usr/local/bin/ | Хранятся скрипты |
| /mnt/backupsql/monthly/ | Хранятся бэкапы сроком до 183 дней |
| /nfs/backup/ | Хранятся бэкапы сроком до 7 дней |
| /var/log/cronlog/ | Логи по отработанным скриптам планировщика cron |
| /var/spool/cron/crontabs | Все настройки cron хранятся тут |

Состояние мест хранения бэкапов на 08.06.22

Бэкапы баз 2 раза в месяц , 17ого и 28ого числа , хранятся не старше 184 дней ,занимаю 75 Гб из 148 Гб (54%) , диск смонтирован в директорию /mnt/backupsql/



Бэкапы баз раз в день , хранятся не старше 7 дней , занимают 197 Гб на сетевом диске размером 5,5 Тб смонтированным в директорию /nfs/backup/

|  |  |
| --- | --- |
| **Основные скрипты** | |
| backup-sql.sh | Бэкапы баз в /nfs/backup/ |
| backup-monthly-sql.sh | Бэкапы баз в /mnt/backupsql/monthly/ |
| maintenance-sql.sh | Ваккум баз и реиндексация |

* **Настройка планировщика Cron**

****

1. В 22:15 каждый день отрабатывает скрипт на бэкап всех баз (backup-sql.sh) в директорию /nfs/backup/
2. В 0:15 каждое воскресенье отрабатывает скрипт на ваккум и переиндексацию баз (maintenance-sql.sh)
3. В 1:20 происходит рестарт сервера 1с
4. В 3:30 , 17ого числа каждого месяца отрабатывает скрипт (backup-monthly-sql.sh) на бэкап баз в директорию /mnt/backupsql/monthly/
5. В 3:30 , 28ого числа каждого месяца отрабатывает скрипт (backup-monthly-sql.sh) на бэкап баз в директорию /mnt/backupsql/monthly/
6. В 4:20 проверка файлов в директории /nfs/backup/ и удаление всех старше 7 дней
7. В 2:20 проверка файлов в директории /mnt/backupsql/monthly/ и удаление всех старше 183 дней

Логи по ошибкам находятся в директории /var/log/cronlog/ и называются в соответствии с отработанным скриптом или действием :

restartsrv1c.log - рестарт сервера 1с

delmonbackup.log – удаление бэкапов старше 183 дней

delbackup.log – удаление бэкапов старше 7 дней

backup-sql.log – бэкапы раз в день

backup-monthly-sql1.log – бэкапы 17ого числа

backup-monthly-sql2.log – бэкапы 28ого числа

* **Публикация базы**

Гайд - <https://its.1c.ru/db/metod8dev/content/5979/hdoc>

Пример на сервере 1С-Br 192.168.2.16

* Зайти в /opt/1cv8/x86\_64/8.3.18.1363/ - cd /opt/1cv8/x86\_64/8.3.18.1363/
* Публикуем базу

./webinst -publish -apache24 -wsdir ZUP\_31\_AFS\_test1 -dir /var/www/html/1c/ZUP\_31\_AFS\_test1/ -connstr "Srvr=192.168.2.16;Ref=ZUP\_31\_AFS\_test1;" -confPath /etc/apache2/apache2.conf

Если vrd файл существует то :

./webinst -publish -apache24 -wsdir ZUP\_31\_AFS\_test1 -descriptor /var/www/html/1c/ZUP\_31\_AFS\_test1/default.vrd -dir /var/www/html/1c/ZUP\_31\_AFS\_test1/ -connstr "Srvr=192.168.2.16;Ref=ZUP\_31\_AFS\_test1;" -confPath /etc/apache2/apache2.conf

* Перезагрузка апача и проверка

Sudo service apache2 restart

service apache2 status

Дополнительно :

/var/www/html/1c/ -здесь vrd от опубликованных баз

/etc/apache2/apache2.conf –проверка конфигураций апача ,все опубликованные базы тут прописаны

* **Настройка Ядра ОС Linux**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Наименование | Значение | Откуда | Описание | | Количество процессоров | 4 | cat /proc/cpuinfo | grep processor | wc -l | Количество процессоров, показанное в /proc/cpuinfo, может не соответствовать реальному количеству ядер процессора. Например, процессор с 2 ядрами и гиперпоточностью будет показан как процессор с 4 ядрами. | | Идентификатор ядра | 0,1,2,3 - (всего 4) | cat /proc/cpuinfo | grep 'core id' | Есть 4 разных идентификатора ядра. Это указывает на то, что существует 4 реальных ядра. | | Данную информацию можно также посомтреть при помощи команды - lscpu. | | | | | Диапазоны портов |  | /proc/ioports | В этом файле содержаться все зарегистрированные диапазоны портов, а также устройства, которые их используют | | MemTotal : доступный объем оперативной памяти | 12264748 kB = 12 Gb | /proc/meminfo | физическая память за вычетом нескольких зарезервированных битов и бинарного кода ядра | | Команда для вывода информации об использовании оперативной памяти - free | | | | | Модуль ядра, или его часто называют драйвером, — это фрагмент кода, расширяющий функциональные возможности ядра.Запустите lsmod из командной строки, чтобы узнать, какие модули ядра загружены в данный момент. | | | | | Точки монтирования |  | /proc/mounts | В этом файле перечислены все точки монтирования и все подключенные файловые системы | | Точную версию ядра | Linux version 5.4.0-96-generic (buildd@lgw01-amd64-051) (gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)) #109-Ubuntu SMP Wed Jan 12 16:4> | /proc/version | Точная версию ядра, компилятора, и в некоторых случаях, даже дистрибутива | | /proc/pid/ - Файловая система proc состоит не только из файлов, но здесь есть и папки. Больше всего здесь папок с номерами вместо имен. Каждый этот номер означает PID процесса, а эта папка содержит информацию о каждом запущенном в системе процессе. Когда процесс заканчивается, его каталог исчезает из системы | | Основные файлы: cmdline - содержит команду с помощью которой был запущен процесс, а также переданные ей параметры cwd - символическая ссылка на текущую рабочую директорию процесса exe - ссылка на исполняемый файл root - ссылка на папку суперпользователя environ - переменные окружения, доступные для процесса fd - содержит файловые дескрипторы, файлы и устройства, которые использует процесс maps, statm, и mem - информация о памяти процесса stat, status - состояние процесса | С помощью этих файлов вы можете составлять различные скрипты. Например если вы хотите уничтожить все зомби процессы, то вы можете сканировать все директории на наличие Z в файле status. Так же само можно проверить запущена ли нужная вам программа просмотрев все cmdline. | | /proc/sys/ -Эта папка в proc linux не только предоставляет информацию о системе, но и позволяет изменять различные параметры ядра на лету, а также включать дополнительные функции. | | Основные подкаталоги в этой папке: debug - содержит отладочную информацию, она будет вам полезна если вы разработчик ядра dev - параметры различных устройств, подключенных к системе fs - вся информация о файловой системе kernel - позволяет напрямую настраивать ядро net - настройка разных параметров сети vm - взаимодействие с подсистемой vm | Чтобы посмотреть можно ли записывать в файлы используйте команду:   ls -ld /proc/sys/  Если у файла есть флаг W, значит в него можно записывать данные. | | Настройки Ядра | Значение | Путь | Команда | | Если установлено 1, разрешает выполнение 32 битных программ в 64 битной системе | 1 (по умолчанию) | /proc/sys/abi/vsyscall32 | sysctl -w abi.vsyscall32=1 | | При возникновении ошибки в ядре выводить значения регистров процессора и стек вызовов процедур | 1 (по умолчанию) | /proc/sys/debug/exception-trace | sysctl -w debug.exception-trace=1 | | Максимальная частота генерации прерываний от системного таймера High Precision Event Timer (HPET), который пришел на замену таймеру реального времени RTC. | 64 (по умолчанию) | /proc/sys/dev/hpet/max-user-freq | sysctl -w dev.hpet.max-user-freq=64 | | Количество асинхронных операций ввода и вывода в вашей файловой системе | 0 | /proc/sys/fs/aio-nr | sysctl -w fs.aio-nr=0 | | Максимальное количество дескрипторов файлов, которые может создать и обрабатывать ядро. Если вы часто получаете сообщения об ошибке из-за невозможности создать дескриптор файла увеличьте этот лимит. По умолчанию установлено значение 10 % от вашей оперативной памяти. | 9223372036854775807 (по умолчанию в данной ОС) | /proc/sys/fs/file-max | sysctl -w fs.file-max=9223372036854775807 | | Подсистема ядра inotify позволяет следить за изменениями в файловой системе. Этот параметр устанавливает максимальное количество событий, которые могут находиться в очереди, перед тем как их обработает программа. | 16384 | /proc/sys/fs/inotify/max\_queued\_events | sysctl -w fs.inotify.max\_queued\_events=16384 | | Позволяет установить доменное имя NIS (Network Internet Services) и YP (Yellow Pages). Но не путайте это доменное имя с DNS, это совсем разные вещи. | (none) | /proc/sys/kernel/domainname | - | | Имя вашего компьютера. Это самый простой способ изменить имя компьютера прямо сейчас, без перезагрузки. | br-1c | /proc/sys/kernel/hostname | sysctl -w kernel.hostname=br-1c | | Позволяет отключить загрузку модулей ядра. | 0 | /proc/sys/kernel/modules\_disabled | sysctl -w kernel.modules\_disabled=0 | | Рандомизация адресного пространства это функция увеличивающая безопасность вашей системы. Она защищает от атак на переполнение буфера. По умолчанию включена. | 2 - полностью автоматизировано | /proc/sys/kernel/randomize\_va\_space | sysctl -w kernel.randomize\_va\_space=2 | | Позволяет включить или отключить управление ядром с помощью SysRQ | 176 (Оно получается из суммы 128 (что позволяет перезагружать и отключать питание) + 32 (возможность перемонтировать файловые системы в режиме только для чтения) + 16, что включает команду синхронизации. ) | /proc/sys/kernel/sysrq | sysctl -w kernel.sysrq=176 | | Максимальное количество запущенных потоков для процессов. | 95118 | /proc/sys/kernel/threads-max | sysctl -w kernel.threads-max=95118 | | Если включено, ядро будет игнорировать все icmp запросы. Рекомендуется для защиты от DDOS атак. | 0 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_all | sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0 | | Так же, как и в предыдущем варианте, только игнорироваться будут только широковещательные icmp запросы | 1 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts | sysctl -w net.ipv4.icmp\_echo\_ignore\_broadcasts=1 | | Максимальное количество узлов, через которые может пройти отправленный пакет перед тем, как достигнет цели. | 64 | /proc/sys/net/ipv4/ip\_default\_ttl | sysctl -w net.ipv4.ip\_default\_ttl=64 | | Разрешить проходящие пакеты через этот компьютер. Обычно такая настройка параметров ядра Linux нужна для роутеров | 0 | /proc/sys/net/ipv4/ip\_forward | sysctl -w net.ipv4.ip\_forward=0 | | Диапазон локальных портов, которые могут быть использованы вашими программами | 32768 60999 (по умолчанию) | /proc/sys/net/ipv4/ip\_local\_port\_range | sysctl -w net.ipv4.ip\_local\_port\_ranged= "32768 60999" | | Установите 1 чтобы защитить компьютер от атаки TCP TimeWait. | 0 | /proc/sys/net/ipv4/tcp\_rfc1337 | sysctl -w net.ipv4.tcp\_rfc1337=0 | | Таймаут ожидания завершения соединения после отправки пакета FIN. Рекомендовано 15. | 60 | /proc/sys/net/ipv4/tcp\_fin\_timeout | sysctl -w net.ipv4.tcp\_fin\_timeout=60 | | Поддерживать соединение активным определенное время, например, 300 секунд. По истечении этого времени TCP соединение будет разорвано. | 7200 | /proc/sys/net/ipv4/tcp\_keepalive\_time | sysctl -w net.ipv4.tcp\_keepalive\_time=7200 | | Указывает размер по умолчанию буфера для сокета получения данных по сети | 212992 | /proc/sys/net/core/rmem\_default | sysctl -w net.**core.rmem\_default**e=212992 | | Максимальный размер буфера сокета для получения данных по сети | 212992 | /proc/sys/net/core/rmem\_max | sysctl -w net.core.rmem\_max=212992 | | Размер сокета по умолчанию для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_default | sysctl -w net.core.wmem\_defaulte=212992 | | Максимальный размер буфера сокета для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_max | sysctl -w net.core.wmem\_max=212992 | | Количество памяти, доступной для работы TCP. | 4096 131072 6291456 (минимум, по умолчанию, максимум) | /proc/sys/net/ipv4/tcp\_rmem |  | | Указывает процент от общей системной памяти, когда фоновый демон pdflush записи данных начнет переписывать кешированные данные на диск | 10 (Рекомендуется 3) | /proc/sys/vm/dirty\_background\_ratio | sysctl -w vm.dirty\_background\_ratio =10 | | Указывает сколько общей оперативной памяти должно быть занято, чтобы процесс, который ведет запись данных на диск инициировал запись кэшированных данных непосредственно на жесткий диск | 20 (Рекомендуется 10) | /proc/sys/vm/dirty\_ratio | sysctl -w vm.dirty \_ratio =20 | | Устанавливает процент свободной памяти, по достижении которого данные начинают переноситься на swap раздел, для систем с большим количеством памяти рекомендовано значение 10. | 60 (Рекомендуется 10) | /proc/sys/vm/swappiness | sysctl -w vm. swappiness =60 | | Определяет условия разрешения и отказа запросов больших объемов памяти. Возможные значения: 0 (по умолчанию) — ядро использует эвристический алгоритм для расчета перерасхода памяти, принимая во внимание объем доступной памяти и число неверных запросов.  1 — ядро не обрабатывает перерасход памяти. При этом вероятность превышения нагрузки на память возрастает, но в то же время увеличивается производительность задач, активно использующих память. 2 — отказ обработки запросов, запрашивающих память, размер которой превышает суммарный размер памяти пространства подкачки и ОЗУ в соответствии с overcommit\_ratio (По умолчанию равен 50). | 0 (Рекомендуется 2) | /proc/sys/vm/overcommit\_memory | sysctl -w vm. overcommit\_memory =2 | | **Так же для СУБД сервера репомендуется в /etc/systemd/logind.conf указать RemoveIPC=no (RemoveIPC в systemd: отключить высвобождение памяти ОС для УЗ) на данный момент RemoveIPC=yes** | | | | |

1. **Сервер 192.168. 1.109 – 1C**

* **Основное**

|  |  |
| --- | --- |
| **Основные директории сервера** | |
|  |  |

|  |  |
| --- | --- |
| **Основные скрипты** | |
|  |  |
|  |  |
|  |  |

* **Настройка Ядра ОС Linux**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Наименование | Значение | Откуда | Описание | | Количество процессоров | 16 | cat /proc/cpuinfo | grep processor | wc -l | Количество процессоров, показанное в /proc/cpuinfo, может не соответствовать реальному количеству ядер процессора. Например, процессор с 2 ядрами и гиперпоточностью будет показан как процессор с 4 ядрами. | | Идентификатор ядра | 0,1,2,3 - (всего 4) | cat /proc/cpuinfo | grep 'core id' | Есть 4 разных идентификатора ядра. Это указывает на то, что существует 4 реальных ядра. | | Данную информацию можно также посомтреть при помощи команды - lscpu. | | | | | Диапазоны портов |  | /proc/ioports | В этом файле содержаться все зарегистрированные диапазоны портов, а также устройства, которые их используют | | MemTotal : доступный объем оперативной памяти | 81451220 kB = 81 Gb | /proc/meminfo | физическая память за вычетом нескольких зарезервированных битов и бинарного кода ядра | | Команда для вывода информации об использовании оперативной памяти - free | | | | | Модуль ядра, или его часто называют драйвером, — это фрагмент кода, расширяющий функциональные возможности ядра.Запустите lsmod из командной строки, чтобы узнать, какие модули ядра загружены в данный момент. | | | | | Точки монтирования |  | /proc/mounts | В этом файле перечислены все точки монтирования и все подключенные файловые системы | | Точную версию ядра | Linux version 4.15.3-1-generic (root@build) (gcc version 6.3.0 20170516 (Debian 6.3.0-18+deb9u1)) #astra21 SMP Thu Aug 22 12:16:21 UTC 2019 | /proc/version | Точная версию ядра, компилятора, и в некоторых случаях, даже дистрибутива | | /proc/pid/ - Файловая система proc состоит не только из файлов, но здесь есть и папки. Больше всего здесь папок с номерами вместо имен. Каждый этот номер означает PID процесса, а эта папка содержит информацию о каждом запущенном в системе процессе. Когда процесс заканчивается, его каталог исчезает из системы | | Основные файлы: cmdline - содержит команду с помощью которой был запущен процесс, а также переданные ей параметры cwd - символическая ссылка на текущую рабочую директорию процесса exe - ссылка на исполняемый файл root - ссылка на папку суперпользователя environ - переменные окружения, доступные для процесса fd - содержит файловые дескрипторы, файлы и устройства, которые использует процесс maps, statm, и mem - информация о памяти процесса stat, status - состояние процесса | С помощью этих файлов вы можете составлять различные скрипты. Например если вы хотите уничтожить все зомби процессы, то вы можете сканировать все директории на наличие Z в файле status. Так же само можно проверить запущена ли нужная вам программа просмотрев все cmdline. | | /proc/sys/ -Эта папка в proc linux не только предоставляет информацию о системе, но и позволяет изменять различные параметры ядра на лету, а также включать дополнительные функции. | | Основные подкаталоги в этой папке: debug - содержит отладочную информацию, она будет вам полезна если вы разработчик ядра dev - параметры различных устройств, подключенных к системе fs - вся информация о файловой системе kernel - позволяет напрямую настраивать ядро net - настройка разных параметров сети vm - взаимодействие с подсистемой vm | Чтобы посмотреть можно ли записывать в файлы используйте команду:   ls -ld /proc/sys/  Если у файла есть флаг W, значит в него можно записывать данные. | | Настройки Ядра | Значение | Путь | Команда | | Если установлено 1, разрешает выполнение 32 битных программ в 64 битной системе | 1 (по умолчанию) | /proc/sys/abi/vsyscall32 | sysctl -w abi.vsyscall32=1 | | При возникновении ошибки в ядре выводить значения регистров процессора и стек вызовов процедур | 1 (по умолчанию) | /proc/sys/debug/exception-trace | sysctl -w debug.exception-trace=1 | | Максимальная частота генерации прерываний от системного таймера High Precision Event Timer (HPET), который пришел на замену таймеру реального времени RTC. | 64 (по умолчанию) | /proc/sys/dev/hpet/max-user-freq | sysctl -w dev.hpet.max-user-freq=64 | | Количество асинхронных операций ввода и вывода в вашей файловой системе | 0 | /proc/sys/fs/aio-nr | sysctl -w fs.aio-nr=0 | | Максимальное количество дескрипторов файлов, которые может создать и обрабатывать ядро. Если вы часто получаете сообщения об ошибке из-за невозможности создать дескриптор файла увеличьте этот лимит. По умолчанию установлено значение 10 % от вашей оперативной памяти. | 8126176 = 8Gb | /proc/sys/fs/file-max | sysctl -w fs.file-max=8126176 | | Подсистема ядра inotify позволяет следить за изменениями в файловой системе. Этот параметр устанавливает максимальное количество событий, которые могут находиться в очереди, перед тем как их обработает программа. | 16384 | /proc/sys/fs/inotify/max\_queued\_events | sysctl -w fs.inotify.max\_queued\_events=16384 | | Позволяет установить доменное имя NIS (Network Internet Services) и YP (Yellow Pages). Но не путайте это доменное имя с DNS, это совсем разные вещи. | (none) | /proc/sys/kernel/domainname | - | | Имя вашего компьютера. Это самый простой способ изменить имя компьютера прямо сейчас, без перезагрузки. | 1C | /proc/sys/kernel/hostname | sysctl -w kernel.hostname=1C | | Позволяет отключить загрузку модулей ядра. | 0 | /proc/sys/kernel/modules\_disabled | sysctl -w kernel.modules\_disabled=0 | | Рандомизация адресного пространства это функция увеличивающая безопасность вашей системы. Она защищает от атак на переполнение буфера. По умолчанию включена. | 2 - полностью автоматизировано | /proc/sys/kernel/randomize\_va\_space | sysctl -w kernel.randomize\_va\_space=2 | | Позволяет включить или отключить управление ядром с помощью SysRQ | 0 - отключить все функции | /proc/sys/kernel/sysrq | sysctl -w kernel.sysrq=0 | | Максимальное количество запущенных потоков для процессов. | 563454 | /proc/sys/kernel/threads-max | sysctl -w kernel.threads-max=563454 | | Если включено, ядро будет игнорировать все icmp запросы. Рекомендуется для защиты от DDOS атак. | 0 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_all | sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0 | | Так же, как и в предыдущем варианте, только игнорироваться будут только широковещательные icmp запросы | 1 | /proc/sys/net/ipv4/icmp\_echo\_ignore\_broadcasts | sysctl -w net.ipv4.icmp\_echo\_ignore\_broadcasts=1 | | Максимальное количество узлов, через которые может пройти отправленный пакет перед тем, как достигнет цели. | 64 | /proc/sys/net/ipv4/ip\_default\_ttl | sysctl -w net.ipv4.ip\_default\_ttl=64 | | Разрешить проходящие пакеты через этот компьютер. Обычно такая настройка параметров ядра Linux нужна для роутеров | 0 | /proc/sys/net/ipv4/ip\_forward | sysctl -w net.ipv4.ip\_forward=0 | | Диапазон локальных портов, которые могут быть использованы вашими программами | 32768 60999 (по умолчанию) | /proc/sys/net/ipv4/ip\_local\_port\_range | sysctl -w net.ipv4.ip\_local\_port\_ranged= "32768 60999" | | Установите 1 чтобы защитить компьютер от атаки TCP TimeWait. | 0 | /proc/sys/net/ipv4/tcp\_rfc1337 | sysctl -w net.ipv4.tcp\_rfc1337=0 | | Таймаут ожидания завершения соединения после отправки пакета FIN. Рекомендовано 15. | 60 | /proc/sys/net/ipv4/tcp\_fin\_timeout | sysctl -w net.ipv4.tcp\_fin\_timeout=60 | | Поддерживать соединение активным определенное время, например, 300 секунд. По истечении этого времени TCP соединение будет разорвано. | 7200 | /proc/sys/net/ipv4/tcp\_keepalive\_time | sysctl -w net.ipv4.tcp\_keepalive\_time=7200 | | Указывает размер по умолчанию буфера для сокета получения данных по сети | 212992 | /proc/sys/net/core/rmem\_default | sysctl -w net.**core.rmem\_default**e=212992 | | Максимальный размер буфера сокета для получения данных по сети | 212992 | /proc/sys/net/core/rmem\_max | sysctl -w net.core.rmem\_max=212992 | | Размер сокета по умолчанию для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_default | sysctl -w net.core.wmem\_defaulte=212992 | | Максимальный размер буфера сокета для отправки данных по сети. | 212992 | /proc/sys/net/core/wmem\_max | sysctl -w net.core.wmem\_max=212992 | | Количество памяти, доступной для работы TCP. | 4096 87380 6291456 (минимум, по умолчанию, максимум) | /proc/sys/net/ipv4/tcp\_rmem |  | | Указывает процент от общей системной памяти, когда фоновый демон pdflush записи данных начнет переписывать кешированные данные на диск | 10 (Рекомендуется 3) | /proc/sys/vm/dirty\_background\_ratio | sysctl -w vm.dirty\_background\_ratio =10 | | Указывает сколько общей оперативной памяти должно быть занято, чтобы процесс, который ведет запись данных на диск инициировал запись кэшированных данных непосредственно на жесткий диск | 20 (Рекомендуется 10) | /proc/sys/vm/dirty\_ratio | sysctl -w vm.dirty \_ratio =20 | | Устанавливает процент свободной памяти, по достижении которого данные начинают переноситься на swap раздел, для систем с большим количеством памяти рекомендовано значение 10. | 60 (Рекомендуется 10) | /proc/sys/vm/swappiness | sysctl -w vm. swappiness =60 | | Определяет условия разрешения и отказа запросов больших объемов памяти. Возможные значения: 0 (по умолчанию) — ядро использует эвристический алгоритм для расчета перерасхода памяти, принимая во внимание объем доступной памяти и число неверных запросов.  1 — ядро не обрабатывает перерасход памяти. При этом вероятность превышения нагрузки на память возрастает, но в то же время увеличивается производительность задач, активно использующих память. 2 — отказ обработки запросов, запрашивающих память, размер которой превышает суммарный размер памяти пространства подкачки и ОЗУ в соответствии с overcommit\_ratio (По умолчанию равен 50). | 0 (Рекомендуется 2) | /proc/sys/vm/overcommit\_memory | sysctl -w vm. overcommit\_memory =2 | | **Так же для СУБД сервера репомендуется в /etc/systemd/logind.conf указать RemoveIPC=no (RemoveIPC в systemd: отключить высвобождение памяти ОС для УЗ) на данный момент RemoveIPC=yes** | | | | |
|  |

1. **Linux , решение типовых задач**

* **Изменение настроек Ядра ОС Linux. Сделать так, чтобы они не изменялись при перезапуске сервера.**

**Задача:** чтобы параметры ядра vm.dirty\_background\_ratio =3, vm.dirty \_ratio =10, vm. swappiness =10,  vm. overcommit\_memory =2; даже после перезапуска сервера.

создаем свой файл конфигурации в каталоге /etc/sysctl.d/.

Например, если вы меняете параметры ядра из-за требований к базе данных, создайте файл:

/etc/sysctl.d/60-mysql.conf

и добавьте свои собственные параметры.

Содержание nano 60-mysql.conf :

vm.dirty\_background\_ratio =3

vm.dirty \_ratio =10

vm. swappiness =10

  vm. overcommit\_memory =2

Чтобы выполнить это:

**service** procps restart

Просто отметить:

procps - это системный файл, и он **НИКОГДА не** должен редактироваться.

/etc/sysctl.conf также не следует редактировать, поскольку его можно изменить при обновлении системы / ядра, и если оно отличается от ожидаемого, обновление будет остановлено с вопросом, может ли более новая версия заменить текущую.

* **Добавление репозитария**

Для использования сетевых репозиториев, работающих по протоколу HTTPS убедитесь, что в системе установлены пакеты **apt-transport-https** и **ca-certificates**, обеспечивающие возможность загрузки пакетов с использованием этого протокола  . Если нет - то установите их:

**sudo apt install apt-transport-https ca-certificates**

Для установки можно использовать также протокол HTTP.

В файле **/etc/apt/sources.list** указать путь к сетевому репозиторию (тег https:// или http:// или ftp://)

**nano /etc/apt/sources.list - открываем и дописываем**

# Основной репозиторий

deb https://dl.astralinux.ru/astra/stable/1.7\_x86-64/repository-main/ 1.7\_x86-64 main contrib non-free

# Оперативные обновления основного репозитория

deb https://dl.astralinux.ru/astra/stable/1.7\_x86-64/repository-update/ 1.7\_x86-64 main contrib non-free

# Базовый репозиторий

deb https://dl.astralinux.ru/astra/stable/1.7\_x86-64/repository-base/ 1.7\_x86-64 main contrib non-free

# Расширенный репозиторий

deb https://dl.astralinux.ru/astra/stable/1.7\_x86-64/repository-extended/ 1.7\_x86-64 main contrib non-free

# Расширенный репозиторий (компонент astra-ce)

deb https://dl.astralinux.ru/astra/stable/1.7\_x86-64/repository-extended/ 1.7\_x86-64 astra-ce

Обновить список пакетов:

**sudo apt update**

* **Настройка сети**

**Полезные команды:**

ip a

ifconfig - требует установка пакета : apt install net-tools

Настройка осуществляется путем правки файла /etc/network/interfaces. Каждый сетевой интерфейс (сетевая карта, хотя это не совсем точное название) настраивается отдельно. Настройки для сервера выглядят так:

nano /etc/network/interfaces

auto lo eth0 eth1

iface lo inet loopback

iface eth0 inet static

address 192.168.0.1

netmask 255.255.255.0

gateway 192.168.0.1 # В качестве шлюза - наш сервер с IP=1

network 192.168.0.0 # Указываем сеть, это обязательно для работы в составе ALD

broadcast 192.168.0.255 # Сервер ALD начиная с Astra 1.5 выводит ошибку, если не видит

dns-nameservers 192.168.0.1 # Тут через пробел можно перечислить ВСЕ DNS-серверы сети

dns-search DOMAIN.NET # Имя домена ALD

#iface eth1 inet dhcp

service networking restart

/etc/netplan/00-installer-config.yaml настройки сети на Ubuntu и более новых ОС Linux

* **Sudo доступ без ввода пароля**

Вводим в терминал **visudo** и дописываем:

# Allow members of group sudo to execute any command

%sudo ALL=(ALL:ALL) ALL

**ivan ALL=(ALL:ALL) NOPASSWD:ALL**

Готово, пользователю **ivan** не надо будет вводить дополнительно пароль при использовании sudo

* **Монтирование сетевого диска**

mount -t cifs //192.168.1.116/backupPG /mnt/backupPG -o user=логин,password=парол

1. **Установка и настройка**
2. **Установка и настройка Postgres12 под 1C на Astra Linux**

**wget** [**https://repo.postgrespro.ru/pg1c-12/keys/pgpro-repo-add.sh**](https://repo.postgrespro.ru/pg1c-12/keys/pgpro-repo-add.sh)

**sh pgpro-repo-add.sh**

Если наш продукт единственный Postgres на вашей машине и вы хотите

сразу получить готовую к употреблению базу:

**apt-get install postgrespro-1c-12**

\_\_\_\_

Если у вас уже установлен другой Postgres и вы хотите чтобы он

продолжал работать параллельно (в том числе и для апгрейда с более

старой major-версии):

apt-get install postgrespro-1c-12-contrib

/opt/pgpro/1c-12/bin/pg-setup initdb

/opt/pgpro/1c-12/bin/pg-setup service enable

/opt/pgpro/1c-12/bin/pg-setup service start

-------

На заметку : сервис данной Postgres будет называться нестандартно: postgrespro-1c-12

**service postgrespro-1c-12 status**

Конфигурации:

/var/lib/pgpro/1c-12/data/pg\_hba.conf

Задать пароль пользователю postgres:

**sudo su postgres  
cd ~  
psql -c "ALTER ROLE postgres WITH PASSWORD ' 12345678';"  
exit**

1. **Установка и настройка 1C-сервера на Astra Linux**

* **Ошибки:**

В Astra Linux 1C автоматически не запускается при загрузке сервера, лечится следующим действием:  
  
nano /etc/systemd/logind.conf  
  
[Login]  
KillUserProcesses=no  
  
далее  
  
systemctl restart systemd-logind

* **Установка 1С сервера на Astra Linux:**
* Загрузить архив дистрибутива на сервер:

/home/ivan/Linux\_Plat\_8-3-18-1363-20220905T125840Z-001.zip

* Распаковать архив:

unzip Linux\_Plat\_8-3-18-1363-20220905T125840Z-001.zip

* Переходим в новую директорию с пакетами:

cd /home/ivan/Linux\_Plat\_8-3-18-1363

* Устанавливаем пакеты в следующей последовательности:

dpkg –i 1c-enterprise-8.3.18.1363-common\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-common-nls\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-server\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-server-nls\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-ws\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-ws-nls\_8.3.18-1363\_amd64.deb

dpkg –i 1c-enterprise-8.3.18.1363-crs\_8.3.18-1363\_amd64.deb

* Создать символьные ссылки на файлы службы:

sudo ln -s /opt/1cv8/x86\_64/8.3.20.1789/srv1cv83 /etc/init.d/srv1cv83  
sudo ln -s /opt/1cv8/x86\_64/8.3.20.1789/srv1cv83.conf /etc/default/srv1cv83

* Обновить списки системных служб:

sudo systemctl daemon-reload

* Разрешить автоматический запуск службы при перезагрузке ОС:

sudo systemctl enable srv1cv83

* Запустить службу:

sudo systemctl start srv1cv83

Материал:

<https://entnet.ru/special/1s-predpriyatie/server-1s-i-pg.html>

https://wiki.astralinux.ru/pages/viewpage.action?pageId=41191288

1. **Установка и настройка 1C-сервера версии 8.3.21 и выше на Astra Linux**

В этой версии 1С поменяли технологию и команды инсталляции.

скачиваем дистриб серв платформы с портала 1С.

Имя файла будет server64\_8\_3\_21\_1622.tar.gz. Его нужно передать на Astra сервер.

Я обычно winscp для этого использую. # home/admin

Распаковываем архив в отдельную директорию,

выполняем по пунктам:

# apt update

# cd /home/admin/

# home/admin# mkdir 1c-serv ( mkdir 1c-serv создать каталог)

# mv server64\_8\_3\_21\_1622.tar.gz 1c-server ( переместить)

# cd 1c-server (войти)

# tar xzvf server64\_8\_3\_21\_1622.tar.gz (разархивировать)

# rm server64\_8\_3\_21\_1150.tar.gz (удаление)

# chmod +x setup-full-8.3.21.1150-x86\_64.run (изменение прав допуска/запуска)

# ./setup-full-8.3.21.1150-x86\_64.run --mode unattended --enable-components server,ws

(ИНСТАЛ сервера +веб сервера)

В данном случае я установил сам кластер серверов 1С и модуль расширения веб сервера. Не забудьте изменить версию платформы в имени файла на свою.

systemctl link /opt/1cv8/x86\_64/8.3.21.1622/srv1cv8-8.3.21.1622@.service

# sudo dpkg -l | grep 1c (смотрим установленные пакеты) проверим все-ли службы запустились

# sudo systemctl start [srv1cv8-8.3.21.1622@default.service](mailto:srv1cv8-8.3.21.1622@default.service) старт сервера

# sudo systemctl status [srv1cv8-8.3.21.1622@default.service](mailto:srv1cv8-8.3.21.1622@default.service) статус сервера проверяем

# sudo systemctl is-enabled [srv1cv8-8.3.21.1622@default.service](mailto:srv1cv8-8.3.21.1622@default.service)

End

Настроим сервер 1С. С этой версии 1С стал использовать подсистему systemd. В состав дистрибутива включен systemd-сценарий запуска кластера серверов и сервера администрирования (ras) и их конфигурационные файлы. Он находится в папке с платформой - /opt/1cv8/x86\_64/номер платформы.  
Установить его можно командой:  
systemctl link /opt/1cv8/x86\_64/8.3.21.1302/srv1cv8-8.3.21.1302@.service.   
  
Затем выполняем стандартные процедуры:  
Запускаем сервер 1С  
sudo systemctl start [srv1cv8-8.3.21.1302@default.service](mailto:srv1cv8-8.3.21.1302@default.service)  
Проверяем работу сервера 1С   
sudo systemctl status [srv1cv8-8.3.21.1302@default.service](mailto:srv1cv8-8.3.21.1302@default.service)  
Добавляем демона в автозапуск   
sudo systemctl enable [srv1cv8-8.3.21.1302@default.service](mailto:srv1cv8-8.3.21.1302@default.service)

Сами данные кластера хранятся в отдельном каталоге cd /home/usr1cv82/.1cv82/1C/1Cv82. Мы можем его скопировать куда нам удобно cp -r /home/usr1cv82/.1cv82/1C/1Cv82 /var/1C82. Рекомендуется хранить эти данные на быстром диске, поскольку в данных кластера находятся и журнал регистрации 1С и индексы полнотекстового поиска.

 Редактирование параметров экземпляра кластера серверов

Для того, чтобы отредактировать параметры какого-либо экземпляра кластера серверов, необходимо использовать следующую команду:

systemctl edit srv1cv8-A.B.C.D@instanceName

Затем в открывшемся файле необходимо создать секцию [Service] (если такая секция еще не задана) и в ней указать изменяемые параметры в следующей нотации: Environment=SRV1CV8\_REGPORT=1541.

1. **Установка и настройка Zabbix-сервера + Postgres 14.5 + Apache на Ubuntu Linux 18.04**

* **Установка Postgres 14.5 на Ubuntu 18.4**

Создание файла конф.репозитория

sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb\_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

Импортируем ключ

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add –

apt-get update

apt-get -y install postgresql

sudo -u postgres psql

\conninfo

* **Установка Апач**

**Установка Apache**

Как и в случае с многим другим популярным ПО, Apache можно установить прямо из стандартных репозиториев Debian.

Сначала обновите локальный индекс пакетов:

$ sudo apt update

А затем выполните установку:

$ sudo apt install apache2

Подтвердите действие, и после этого утилита apt установит Apache и все необходимые зависимости.

**Настройка файрвола**

Теперь необходимо внести некоторые изменения в настройки файрвола.

Выведите все доступные профили приложений на своем сервере:

$ sudo ufw app list

Вывод будет, например, таким:

Available applications:

AIM

Bonjour

CIFS

. . .

WWW

WWW Cache

WWW Full

WWW Secure

. . .

Профили Apache начинаются с WWW:

* WWW: этот профиль открывает 80 порт (обычный, незашифрованный веб-трафик);
* WWW Cache: профиль открывает только 8080 порт (иногда используется для кэширования и веб-прокси);
* WWW Secure: профиль открывает только 443 порт (TLS/SSL зашифрованный трафик);
* WWW Full: профиль открывает оба порта – 80 и 443.

Из этих профилей желательно выбрать тот, который разрешает только те порты, что вам нужны.

Т.к. SSL на сервере еще не настроен, вам нужно открыть порт 80:

$ sudo ufw allow 'WWW'

Проверьте, вступили ли изменения в силу:

$ sudo ufw status

В выводе вы увидите, что трафик HTTP теперь разрешен:

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

WWW ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

WWW (v6) ALLOW Anywhere (v6)

Значит, все нормально, и порт успешно открыт.

**Проверка веб-сервера**

Но на всякий случай это можно проверить командой:

$ sudo systemctl status apache2

* **Установка Zabbix сервер**

wget <https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.0-3%2Bubuntu18.04_all.deb>

dpkg -i zabbix-release\_6.0-3+ubuntu18.04\_all.deb

apt update

apt install zabbix-server-pgsql zabbix-frontend-php php7.2-pgsql zabbix-apache-conf zabbix-sql-scripts zabbix-agent

sudo -u postgres createuser --pwprompt zabbix  
 sudo -u postgres createdb -O zabbix zabbix

zcat /usr/share/doc/zabbix-sql-scripts/postgresql/server.sql.gz | sudo -u zabbix psql zabbix

Редактируем nano /etc/zabbix/zabbix\_server.conf

DBPassword=password

systemctl restart zabbix-server zabbix-agent apache2  
systemctl enable zabbix-server zabbix-agent apache2

* **Настройка оповещения в Telegram чат**

**Шаг 1: Создать группу в Telegram и добавить туда бота для получения ID**

****Создаем группу и добавляем бота **@getmyid\_bot** , он выдаст ID личный и ID группы ,это второй(**-832378868**) .Он нам еще пригодится.

Изображение выглядит как текст

Автоматически созданное описание

**Шаг 2: Создаем бота у @BotFather**

Пишем в Telegram боту (как и любому другому пользователю) @BotFather команду [/newbot](tg://bot_command?command=newbot)

Задаем **name** и **username**(должен быть уникальным) . Получаем **Токен** бота **5492590427:AAEKGyp4iIvdCcUcKZOffHEZeyHnbBWcBHk**

**Шаг 3: Настройка в Вэб интерфейсе Zabbix**

Администрирование -> Способы оповещения -> Telegram (тип Webhook)

****



Вводим в поле Token – токен который выдал **@BotFather** : 5492590427:AAEKGyp4iIvdCcUcKZOffHEZeyHnbBWcBHk

Заходим : Администрирование -> Пользователи ->Выбиваем пользователя: Admin -> Оповещение

Добавляем тип оповещения тип : Telegram и в поле Отправить на дополняем ID чата от бота **@getmyid\_bot** : **-832378868** 

Готово!

* **Пример установки Zabbix agent 6.2 на хосте Ubuntu 20.04**

# wget https://repo.zabbix.com/zabbix/6.2/ubuntu/pool/main/z/zabbix-release/zabbix-release\_6.2-4%2Bubuntu20.04\_all.deb  
# dpkg -i zabbix-release\_6.2-4+ubuntu20.04\_all.deb  
# apt update

apt zabbix-agent

Редактируем конфиг , пишем IP Zabbix сервера и имя сервера , такое же как будет указанно в узле сети на сервере.

nano /etc/zabbix/zabbix\_agentd.conf

# systemctl restart zabbix-agent   
# systemctl enable zabbix-agent

1. **Установка и настройка Ansible на Ubuntu Linux 18.04**

* **Установка Ansible**

Чтобы добавить архив PPA (архив персональных пакетов) официального проекта в список источников вашей системы, запустите на узле управления следующую команду:

sudo apt-add-repository ppa:ansible/ansible

sudo apt update

После этого обновления вы можете установить программное обеспечение Ansible следующим образом:

sudo apt install ansible

Файл конфигураций Ansible находится в : /etc/ansible/ansible.cfg

Файл с хостами в: /etc/ansible/hosts

Переходим в директорию /root и создаем там директории :

mkdir ansible

cd /root/ansible

mkdir group\_vars -создание директории для групповых переменных (облегчает чтение файлa hosts ,хорошая практика)

Расположение логов : /var/log/ansible.log

Подключение хостов : nano /etc/ansible/hosts по умолчанию

[servers]

server1 ansible\_host=203.0.113.111

server2 ansible\_host=203.0.113.112

server3 ansible\_host=203.0.113.113

Проверка

ansible-inventory --list –y

* **Настройка подключения к хосту по shh key**

**На Ansible сервере:**

**ssh-keygen -t rsa**

Задаем пароль и запоминаем , допустим (12345), в результате :

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

Далее переименовываем для удобства id\_rsa в имя хоста – server1

mv id\_rsa server1

Открываем и копируем содержимое :

cat id\_rsa.pub

mv id\_rsa.pub server1.pub – заодно переименуем ,для удобства

**На сервере-хосте:**

Открываем nano /root/.ssh/authorized\_keys

И вставляем туда содежимое cat id\_rsa.pub с Ansible-сервера

!**Проблема** AstraLinux – отсутствие директории .ssh вместе с authorized\_keys

**Решение**:

Генерируем ключи :ssh-keygen , в результате создается директория .ssh в /root

Заходим cd /root/.shh и создаем здесь файл nano authorized\_keys ,в который и вписываем содержимое cat id\_rsa.pub с Ansible-сервера

Далее на Ansible сервере редактируем nano /root/ansible/hosts

a

**Проверка**

Тест: ansible all -m ping

Ансибл , all-все хосты , -m – значит модуль , ping – пинг)

Вводим заданный пароль(12345) и в результате должны получить :

server1 | SUCCESS => {

"changed": false,

"ping": "pong"

}

**Создание групповых переменных**

nano /etc/ansible/hosts на данный момент выглядит так :

[group1]

serverone ansible\_host=192.168.56.106 ansible\_user=root ansible\_ssh\_private\_key\_file=/root/.ssh/all

serverthree ansible\_host=192.168.56.109 ansible\_user=root ansible\_ssh\_private\_key\_file=/root/.ssh/all

[group2]

servertwo ansible\_host=192.168.56.114 ansible\_user=root ansible\_ssh\_private\_key\_file=/root/.ssh/all

[group3]

serverfour ansible\_host=192.168.56.113 ansible\_user=root ansible\_ssh\_private\_key\_file=/root/.ssh/all

[gr12:children]

group1

group2

[gr23:children]

group2

group3

[gr13:children]

group1

group3

[allgr:children]

group1

group2

group3

Хосты поделены на группы , к каждому свой ssh ключ , но можно создать в /root/ancible/ group\_vars файл для всех групп allgr , куда вынести переменные ansible\_user=root ansible\_ssh\_private\_key\_file=/root/.ssh/all ,при этом нужно на всех серверах прокинуть один ssh ключ all.pub .по аналогии ,как делали для отдельного сервера

**На сервере:**

ssh-keygen -создаем

mv id\_rsa.pub all.pub -переименовываем

mv id\_rsa all

nano all.pub -копируем

**На хостах:**

nano /root/.ssh/authorized\_keys

добавляем скопированный ключ из all.pub ,сохраняем

**На сервере:**

Переходим в cd /root/ansible/group\_vars

nano allgr –создаем и редактируем (название файла по названию группы ,для которой прописываются переменные)

ansible\_user : root

ansible\_ssh\_private\_key\_file : /root/.ssh/all

ansible\_python\_interpreter : /usr/bin/python3

На хостах установить пакет **apt install python3-psycopg2**

nano /etc/ansible/hosts –заходим и редактируем

[group1]

serverone ansible\_host=192.168.56.106

serverthree ansible\_host=192.168.56.109

[group2]

servertwo ansible\_host=192.168.56.114

[group3]

serverfour ansible\_host=192.168.56.113

[gr12:children]

group1

group2

[gr23:children]

group2

group3

[gr13:children]

group1

group3

[allgr:children]

group1

group2

group3

Проверка :

ansible-inventory --list –y

ansible all -m ping

**Создание плейбуков:**

В директории /root/ansible создаем файл touch ping.yml и заходим в него для редактирования vi ping.yml

(редактор vi – нажать I для выхода из командного режима , ESC чтобы ввойти в командный режим , q! –выйти без сохранения , wq –сохранить и выйти)

- name: Ping Servers

hosts: all

become : yes

tasks:

- name: Task ping

ping:

* **Администрирование Postgres-12-1C на Astra Linux, особенности:**

**Требуются пакеты на хосте:**

**apt install acl**

**apt install python3-psycopg2**

**Первая ошибка**: An exception occurred during task execution. To see the full traceback, use -vvv. The error was: соединения через Unix-сокет "/var/run/postgresql/.s.PGSQL.5432"?

Postgres установленный под 1С создает сокет в директори: **/tmp/** , пока модуль будет искать его в стандартом месте : **/var/run/postgresql/**

**Решение:**

ln -s /tmp/.s.PGSQL.5432 /var/run/postgresql/.s.PGSQL.5432

Создаем символьную ссылку ,при необходимости создаем директорию postgresql ,если она отсутсвует.

**Вторая ошибка:** An exception occurred during task execution. To see the full traceback, use -vvv. The error was: FileNotFoundError: [Errno 2] Нет такого файла или каталога: '/var/lib/postgresql/data/pg\_hba.conf'

**Решение:**

ln -s /var/lib/pgpro/1c-12/data/pg\_hba.conf /var/lib/postgresql/data/pg\_hba.conf

(для postgrespro на 115ом сервере)

ln -s /var/lib/pgpro/ent-12/data/pg\_hba.conf /var/lib/postgresql/data/pg\_hba.conf

Аналогично создаем символьную ссылку , при необходимости ,создаем недостающие директории

На заметку : сервис данной Postgres будет называться нестандартно: postgrespro-1c-12

* **Пример работы с Postgres через Ansible используя плейбуки**

- Подключаемся к 192.168.1.120

- Переходим в директорию ansible

**cd /root/ansible**

Список доступных плейбуков:

crdb.yml - создание новой базы

drop.yml – удаление базы

dump.yml – бэкап(дамп) базы

rsdb.yml – загрузка базы из дампа

**Задача:** Создать новую базу

- Открываем файл vars.yml

**nano vars.yml**

и добавляем название новой базы, например: test\_create\_ansdb, сохранили и вышли.

**create\_db\_name:** *test\_create\_ansdb*

- Запускаем плейбук:

**ansible-playbook crdb.yml**

- Готово, база создана.

**Задача:** Создать бэкап , дамп базы

- Открываем файл vars.yml

**nano vars.yml**

и добавляем название базы, например: test\_create\_ansdb, сохранили и вышли.

**dump\_db\_name:** *test\_create\_ansdb*

- Запускаем плейбук:

**ansible-playbook dump.yml**

- Готово, дамп базы создан (на 192.168.1.120): /home/backup/test\_create\_ansdb.dump.gz.

**Задача:** Загрузка базы из дампа

- Открываем файл vars.yml

**nano vars.yml**

и добавляем название базы, например: test\_create\_ansdb и имя дамп файла расположенного в директории /home/restore/ (на сервере 1.120), например: /home/restore/dumptestbase.dump.gz, сохранили и вышли.

#Restore DataBase

**restore\_db\_name:** *test\_create\_ansdb*

#Restore from dump file , dir: /home/restore/{dump\_file\_name}.dump.gz

**dump\_file\_name:** *dumptestbase*

- Запускаем плейбук:

**ansible-playbook rsdb.yml**

- Готово, база восстановлена из дампа.

**Задача:** Удаление базы (Drop Data Base)

- Открываем файл vars.yml

**nano vars.yml**

и добавляем название базы, например: test\_create\_ansdb, сохранили и вышли.

#Drop DataBase

**drop\_db\_name:** *test\_create\_ansdb*

- Запускаем плейбук:

**ansible-playbook drop.yml**

- Готово, базa test\_create\_ansdb удалена.

* **Пример написания плебука для установки 1С Сервер версии 8.3.21+**

Прежде всего нам нужен сервер с установленным Ansible. К нему нужно подключить сервер по ssh, на который будет устанавливаться 1С. Данный процесс описан выше.

Выбираем любую удобную для нас директорию откуда мы и будем в будущем запускать наш плейбук .

Основная директория Ansible : **/etc/ansible** , допустим мы создали здесь директорию **mkdir playbook** , зашли туда **cd /etc/ansible/playbook**

**nano srv1cinstall.yml**

В yml формате важны отступы!

Начинается каждый плебук с указания hosts – хосты на которых он будет выполнен: all – на всех.

Become: true – чтобы позволить исполнять плейбук от пользователя root.

- hosts: all  
 become: true

В vars: указываем переменные, что мы задаем для удобства, чтобы быстро редактировать плейбук на установку платформы 1С другой версии.

installer\_on\_ansible\_server – полное имя и директорию в которой лежит .run файл установки 1С на сервере Ansible.

Ver – версия платформы в формате 8.3.2x.xxxx

vars:  
 installer\_on\_ansible\_server: /home/ivan/setup-full-8.3.21.1622-x86\_64.run  
 ver: 8.3.21.1622

tasks – дальше идет перечисление задач ,которые будут выполняться последовательно.

name- имя задачи , можно не указывать , но тогда не будет понятно при исполнении плейбука , что за задача выполняется настоящий момент.

tasks:  
  
 - name: Update and Upgrade  
 apt:  
 update\_cache: yes  
 upgrade: yes

Соответственно данная задача с имеем **Update and Upgrade**

Использует **apt –** модуль, про обширные возможности которого можно почитать в документации к Ansible.

**update\_cache: yes  
upgrade: yes**  
Позволяют обновить индекс пакетов и списки пакетов и обновить пакеты программного обеспечения до последней версии

- name: Copy .run file  
 copy:  
 src: "{{installer\_on\_ansible\_server}}"  
 dest: /root/srv1c.run  
 mode: '0777'

Далее модуль **copy**, который копирует файл , (**src**)полный путь и имя которого мы указали в **var** (переменных) . **dest –**директория в которую будет скопирован .run файл с одновременным переименованием в **srv1c.run** и выдача полных прав 777 (**mode**)

- name: Start install  
 shell: /root/srv1c.run --mode unattended --enable-components server,ws,server\_admin,liberica\_jre

Далее используется модуль **shell**- он просто исполняет в терминале ту команду, что тут указана:

**/root/srv1c.run --mode unattended --enable-components server,ws,server\_admin,liberica\_jre,ru**

Запускает пакетную установку 1с.

Желательно использовать специализированные модули и не писать плейбук из одних модулей **shell** – ведь в ней отсутствуют дополнительны проверки.

- name:  
 shell: systemctl link /opt/1cv8/x86\_64/{{ver}}/srv1cv8-{{ver}}@.service

Создает линк на службу : **srv1cv8-{{ver}}@.service**

- name: Demon reload  
 systemd: daemon\_reload=yes

Перезапуск демонов

- name: Enabled for srv1cv83  
 systemd:  
 name: srv1cv8-{{ver}}@default.service  
 enabled: yes  
 masked: no

Тут аналогично команде **systemctl enabled srv1cv8-{{ver}}@.service** , но с дополнительными проверками заложенными в модуль **systemd**

- name: Started srv1cv83  
 shell: systemctl start [srv1cv8-{{ver}}@default.service](mailto:srv1cv8-%7b%7bver%7d%7d@default.service)

Стартует **srv1cv8-{{ver}}@default.service**

Запуск плейбука :

Находимся в директории **/etc/ansible/playbook** где лежит наш **srv1cinstall.yml** файл

**ansible-playbook** **srv1cinstall.yml**

1. **Установка Git и подключение репозитория с данным гайдом**

Материал: <https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0-Git>

Установка в Windows

Для установки Git в Windows также имеется несколько способов. Официальная сборка доступна для скачивания на официальном сайте Git. Просто перейдите на страницу <https://git-scm.com/download/win>, и загрузка запустится автоматически.

Зайти в Git Bush и перейти в директорию ,в которую скачаем репозиторий ,например: D:\Git

**cd /d/Git**

Или создать папку и запустить Git Bush сразу в ней : 

Git Bush Here

**git clone** <https://github.com/MakeUsWhole/admin.git>



Для обновления файлов в соотвествии с последними обновлениями в репозитории

**git pull**

1. **Установка и настройка VPN сервера (OpenVPN) ОС Ubuntu**

Шаг 1: Установка пакетов

**sudo apt install mc nano wget curl**

Шаг 2:Установка ,настройка OpenVpn сервера

**wget** [**https://git.io/vpn**](https://git.io/vpn) **-O** [**openvpn-install.sh**](https://openvpn-install.sh/) **&& bash** [**openvpn-install.sh**](https://openvpn-install.sh/)

Далее настраиваем ВПН сервер по шагам , название сервера = его статический IP , и в конце создаем первого клиента . В /root/ появится файл {имя\_клиента}.ovpn , его нужно отправить пользователю с установленным OpenVpn

Шаг 3: Настройка роутера в сети VPN сервера

В браузере вводим: **192.168.1.1** или **192.168.1.0** , лагин и пароль по умолчанию: **admin** и **admin** .

В примере : **Интернет-> Переадресация портов**   
   
Добавляем службу: **ovpn** с диапазоном портов: **1194** (на него придет запрос на подключение к впн от клиента) , локальный IP сервера VPN , в данном случае: **192.168.1.133** , локальный порт: **1194** . Протокол выбираем в зависимости от того, как настроили VPN сервер: **UDP** или **TCP**



Готово!

1. **Настройка подключения к удаленному рабочему столу с Windows на Linux Ubuntu**

*Шаг 1: Обновление пакетов*

**sudo apt update**

*Шаг 2: Открываем порт 3389*

**ufw allow 3389**

*Шаг 3: Настройка роутера в сети РДП сервера*

В браузере вводим: **192.168.1.1** или **192.168.1.0** , лагин и пароль по умолчанию: **admin** и **admin** .

В примере : **Интернет-> Переадресация портов**   
   
Добавляем службу: **rdp** с диапазоном портов: **3389** (на него придет запрос на подключение к впн от клиента) , локальный IP сервера RDP , в данном случае: **192.168.1.133** , локальный порт: **3389** . Протокол выбираем: **TCP**



*Шаг 4: Установка необходимых пакетов*

**apt install xrdp**

**apt install xorgxrdp**

*Шаг 5: Включение сервиса и проверка статуса*

**systemctl enable xrdp**

**systemctl status xrdp**

*Шаг 6: Установка необходимых пакетов*

**apt install xfce4**

*Шаг 7: Включение сервиса и проверка статуса*

**systemctl start xrdp**

**systemctl status xrdp**



*Шаг 8: Узнаем ip RDP сервера*

Используем сервис <https://2ip.ru/>



В данном примере ip (статический) : 78.107.237.245

*Шаг 9: Подключаемся с Windows клиента к RDP серверу Ubunty*

В сервисе **Подключение к удаленному рабочему столу** пишем IP сервера RDP и логин пользователя .



Вводим пароль и готово !

1. **Монтирование сетевого диска , настройка бэкапов на Postgres и планировщика Cron**

* **Подготовка**

Рассматриваться будет на примере двух серверов.   
192.168.1.116 - Сервер , где будут храниться Бэкапы (ОС Windows)   
192.168.1.137 – Сервер Субд Postgres

**Шаг 1 : Подготовка на сервере 192.168.1.116**

1) На сервере 116 создали папку и расшарили ееИзображение выглядит как текст

Автоматически созданное описание

2) Создаем пользователя для подключения  
Управление компьютером > Пользователи > Новый пользователь  
В примере Имя **test1137** Пароль **Qwerty123**

**Шаг 2. Подготовка на сервере 192.168.1.137**  
  
На сервере 137 создали директорю ,куда будут выгружаться бэкапы  
Например /mnt/backup137

**cd /mnt**

**mkdir backup137**

* **Монтируем сетевую папку**

**mount -t cifs //192.168.1.116/backup137 /mnt/backup137 -o user=test1137,password=Qwerty123**   
! Если выдает ошибку, то возможно потребуется установка пакета :  
**apt install cifs-utils**

* **Настройка Бэкапов баз на Субд сервере Postgres**

Перешли в папку

**cd /usr/local/bin**   
Запустили редактор (создаем скрипт)  
**sudo nano backup.sh**

Содержание:  
  
**#!/bin/sh**

**# Устанавливаем дату**

**DATA=`date +"%Y-%m-%d\_%H-%M"`**

**# Бэкапим базу данных base1c и сразу сжимаем**

**/opt/pgpro/ent-12/bin/pg\_dump -U postgres test111 | pigz > /mnt/backup137/$DATA-test111.sql.gz**

**/opt/pgpro/ent-12/bin/pg\_dump -U postgres test121 | pigz > /mnt/backup137/$DATA-test121.sql.gz**

Права на использование скрипта  
**chmod 755 backup.sh**

Для работы скрипта требуется установленный пакет

**apt install pigz**

Права на использование **pg\_dump**

**cd /opt/pgpro/ent-12/bin/**

**chmod 755 pg\_dump**

Если ругается на библиотеку : libpq.so.5 то:

Находим ее

**find / -name libpq.so.5**Нашло :  
**/opt/pgpro/ent-12/lib/libpq.so.5.12** -нужная нам

и

/opt/pgpro/1c-12/lib/libpq.so.5

Создаем символьную ссылку

**ln -s /opt/pgpro/ent-12/lib/libpq.so.5.12 /usr/lib/libpq.so.5**

Пробуем запускать скрипт

**/usr/local/bin/backup.sh**Если выдает :

pg\_dump: ошибка: не удалось подключиться к базе "test111": ВАЖНО: пользователь "postgres" не прошёл проверку подлинности (Peer)

Нужно отредактировать конфигурации постгрес сервера и перезагрузить его:

Находим файл конфигураций:

**find / -name pg\_hba.conf**Открываем и редактируем: **nano /var/lib/pgpro/1c-12/data/pg\_hba.conf** Редактируем содержимое

# "local" is for Unix domain socket connections only

local all all **trust**Требуется установить параметр **trust** для локальных соединений

Перезапустить службу постгрес

**service postgrespro-1c-12 status**

**service postgrespro-1c-12 restart**

**service postgrespro-1c-12 status**

* **Настройка планировщика Cron**

**crontab -e**

Редактируем и дописываем:

**20 0 \* \* \* /usr/local/bin/backup.sh**

**10 0 \* \* \* find /mnt/backup137 -type f -delete**

Каждый день в 0:10 удаляется все содержимое /mnt/backup137   
В 0:20 выполняется наш скрипт на бэкап баз

1. **Настройка route . Задача : чтобы интернет работал , только при подключении по ВПН**

**Отказываемся от DHCP.**

1. заходим в cmd от имени администратора. Нам нужно узнать основной сетевой адаптер, и какой адрес, шлюз и DNS нам дал DHCP:

ipconfig/all



Записываем все значения в блокнот.

2. Включаем наш ВПН и снова открываем cmd и пишем команду:

route print -4

Там будет высвечено 2 default маршрута для трафика. Начинаются они с 0.0.0.0.

Различить кто есть чей легко. У стандартного Адрес шлюза будет с адресом, который мы записали ранее. У ВПН - отличный от этого значения.

Нужно записать шлюз ВПНа тоже в блокнот.

3. Идем в - Панель управления\Все элементы панели управления\Сетевые подключения.

В разных виндах это по разному, главное добраться вот до этого окошка, в общем:



Находим наш основной адаптер, данные которого мы ранее записали, жмем ПКМ, выбираем "Свойства". Затем "IP версии 4", снова свойства. В появившемся окне убираем галочки с автоматических настроек и прописываем все, что записали ранее.

В итоге должно получится что-то типо такого:



Если инет работает, поздравляю, ты все сделал как надо . Этот мув поможет нам избежать в дальнейшем автоматически создаваемого маршрута всего трафика на шлюз интернета в роутер.

**Пробрасываем новые роуты**.

1. Для начала удалим дефолтный маршрут 0.0.0.0

route delete 0.0.0.0 mask 0.0.0.0

2. Затем добавим новый дефолтный маршрут на шлюз ВПН, который записали ранее:

route add -p 0.0.0.0 mask 0.0.0.0 10.8.0.1 metric 1 if 19

где:

-p - постоянный маршрут, который будет подрубаться всякий раз при перезагрузки адаптера(а ля компа)

0.0.0.0 - дефолтный айпишник любого трафика, если для адреса не задан какой-либо маршрут

10.8.0.1 - Шлюз ВПНа

metric 1 - приоритет, если для одного и того же адреса есть несколько маршрутов в таблице. Но у нас 1, но на всякий случай напишем наивысший приор

if 19 - номер сетевого интерфейса. Берется из команды route print -4

3. Теперь создаем маршрут, который будет посылать трафик только на айпи ВПН сервака в интернетный шлюз роутера:

route add -p 78.107.237.245 mask 255.255.255.255 192.168.88.1 metric 2 if 16

где:

192.168.88.1 - интернетный шлюз, который мы записали вначале гайда.

**Итог:**Можно создать 2 bat файла , для переключения.   
Первый: Служит для переключения потока интернета только через ВПН ( без ВПН интернета не будет ). Запускать от имени администратора и строго до подключения по ВПН.

route delete 0.0.0.0 mask 0.0.0.0

route add -p 0.0.0.0 mask 0.0.0.0 10.8.0.1 metric 1 if 15

Второй: Служит для того ,чтобы вернуть все обратно . Чтобы поток интернета шел через роутер. Сначала отключаемся от ВПН , затем запускаем от имени администратора.

route delete 0.0.0.0 mask 0.0.0.0

route add -p 0.0.0.0 mask 0.0.0.0 192.168.0.1 metric 1 if 15

1. **Установка Docker и Docker-compose на Astra Linux**

* **Простая установка и удаление**

В дистрибутив выходит достаточно старая версия:

CE 2.12.44 (orel): Docker version 18.09.7, build 2d0083d

Для ее установки необходимо обновить данные репозитория и запустить установку командой:

apt update

apt install docker.io

Я передумал, это слишком старый докер!

Если указанная версия Docker не подходит, удаление производится следующей командой:

apt-get remove docker docker-engine docker.io containerd runc

* **Установка в изолированном сегменте**

Следуя данному руководству вы сможете установить версии Docker и Dockeer-Compose созданные до начала СВО в изолированном сегменте.

**Версия Docker созданная до начала СВО: 19.03.9 (март 2019)**

Необходимо скачать дистрибутив. Данная операция может быть проведена на любой другой машине с доступом в интернет. После скачивания все файлы необходимо передать на целевую машину или сразу выполнять скачивание с нее.

wget https://download.docker.com/linux/debian/dists/stretch/pool/stable/amd64/docker-ce\_19.03.9~3-0~debian-stretch\_amd64.deb

docker-ce-cli

wget <https://download.docker.com/linux/debian/dists/stretch/pool/stable/amd64/docker-ce-cli_19.03.9~3-0~debian-stretch_amd64.deb>

wget https://download.docker.com/linux/debian/dists/stretch/pool/stable/amd64/containerd.io\_1.4.3-1\_amd64.deb

Для установки скаченных дистрибутивов необходимо выполнить команду:

dpkg -i docker-ce\_19.03.9~3-0~debian-stretch\_amd64.deb docker-ce-cli\_19.03.9~3-0~debian-stretch\_amd64.deb containerd.io\_1.4.3-1\_amd64.deb

Первоначальная настройка после установки выполняется следующим образом:

sudo groupadd docker

sudo usermod -aG docker $USER

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

**Docker Compose, созданный до СВО 1.27.4 (сентябрь 2020)**

Установка Docker Compose производится путем скачивания собранного образа с сайта Github. Скачать образ можно на любой машине с доступом в интернет и передать его на целевую систему.

wget <https://github.com/docker/compose/releases/download/1.27.4/docker-compose-Linux-x86_64>

mv ./docker-compose-Linux-x86\_64 /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

* **Установка актуальных версий**

Если заказчиком не предъявляются повышенные требования к свободному программному обеспечению можно установить самую актуальную версию, подключив репозиторий Docker в систему.

**Установка актуальной версии Docker**

Данная операция выполняется на целевой машине с доступом в интернет.

**sudo** apt **install** apt-transport-https ca-certificates curl gnupg2 software-properties-common

curl -fsSL https://download.docker.com/linux/debian/gpg | **sudo** **apt-key add** –

**sudo** printf "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable **\n**" > /etc/apt/sources.list.d/docker.list

**sudo** **apt-get update**

**sudo** **apt-get install** docker-ce docker-ce-cli containerd.io

Первоначальная настройка после установки, выполняется следующей командой:

Создаем группу пользователей docker (для взаимодействия с docker без ввода sudo) и добавляем в нее $USER - текущий пользователь.

Добавляем сервисы containerd.service и docker.service в автозагрузку

sudo groupadd docker

sudo usermod -aG docker $USER

sudo systemctl enable docker.service

sudo systemctl enable containerd.service

* **Установка Docker Compose v.1.29**

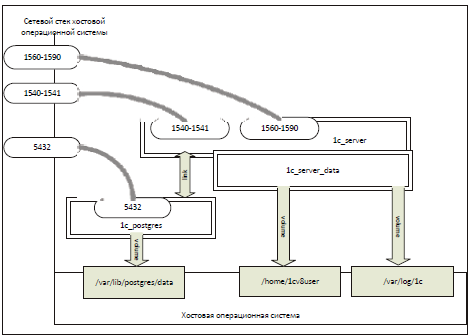
Данная команда позволяет установить актуальную на момент составления статью версию Docker Compose.

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

Самая последняя версия доступна для [скачивания на Github](https://github.com/docker/compose/releases), установка производится аналогичным образом.

1. **Установка 1С Server + Postgres + Web Server (Apache) в Docker-Compose**



* **Предварительная настройка**

Переходим в корень и создаем директории:

cd /

mkdir mydocker

Переходим в новую директорию проекта , и создаем директории:

**mkdir pg**

**cd pg**

**mkdir data**

**mkdir backup**

**cd ..**

**mkdir httpd**

**cd httpd**

**mkdir sites**

**mkdir log**

**touch 1c.conf**

**cd ..**

**mkdir 1c**

**cd 1c**

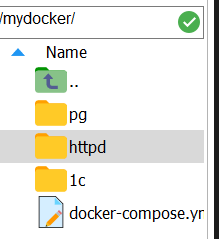
**mkdir tmp**

**mkdir licenses**

**mkdir cache**

**cd ..**

По итогу мы находимся в директории /mydocker



Создаем docker-compose.yml

**nano docker-compose.yml**

* **Описание для будущего билда docker-compose.yml**

Содержание, внимательно, это yml – все отступы очень важны:

version: "3.9"

services:

sv00apptest01:

image: bulanovas/server1c:8.3.20.1789df

volumes:

- "./1c/cache:/home/usr1cv8/.1cv8/1C/1cv8"

- "./1c/licenses:/var/1C/licenses"

- "./1c/tmp:/tmp"

networks:

- 1c

hostname: sv00apptest01

ports:

- "1540:1540"

- "1541:1541"

- "1560-1591:1560-1591"

tty: true

container\_name: server1c

apache\_1c:

image: bulanovas/apache1c:8.3.20.1789df

volumes:

- "./httpd/sites:/var/www/1c"

- "./httpd/1c.conf:/etc/apache2/sites-available/1c.conf"

- "./httpd/log:/var/log/apache2"

networks:

- 1c

hostname: sv00webtest01

ports:

- "80:80"

- "443:443"

container\_name: apache1c

pg\_1c:

image: bulanovas/postgres1c:12.5-3.1C

volumes:

- "./pg/data:/var/lib/postgresql/data"

- "./pg/backup:/backup"

environment:

- POSTGRES\_PASSWORD="12345"

networks:

- 1c

ports:

- "5432:5432"

container\_name: postgres1c

networks:

1c:

external: true

Сохранили и закрыли.

* **Загрузка снимком (images) и запуск проекта**

Загружаем необходимые images, для этого потребуется ввести три команды:

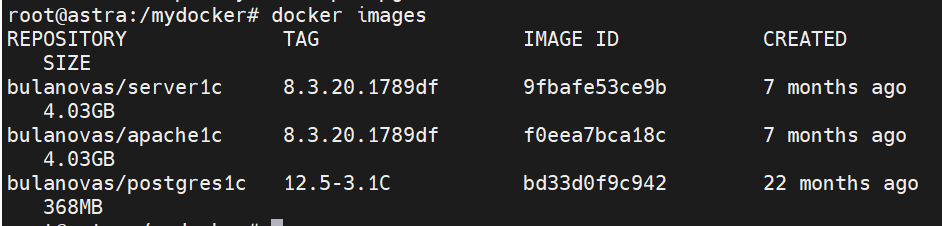
**docker pull bulanovas/apache1c:8.3.20.1789df**

**docker pull bulanovas/server1c:8.3.20.1789df**

**docker pull bulanovas/postgres1c**

После каждой , будет происходить загрузка. В результате мы должны увидеть при вводе команды:

**docker images**

****

Запускаем наш проект:

Требуется находиться в директории /mydocker (где и расположен наш docker-compose.yml) **cd /mydocker**

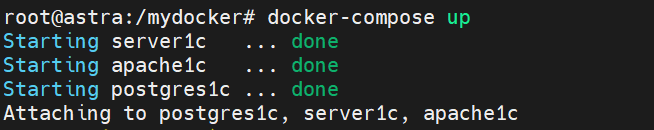
Последовательно вводим команды:

**docker-compose build**

**С**оздается билд проекта по описанию, что мы сделали в docker-compose.yml

**docker-compose up**

Запуск проекта, должны увидеть:



Для завершения работы серверов: Ctrl + C

Команды и все необходимое можно найти в : <https://github.com/MakeUsWhole/docker-master.git>

1. **Сервер управления и мониторинга. Ubuntu 20.04 Ansible + Zabbix в Docker-Compose**

* **Предварительные работы**

Для установки OpenSSH на сервер вы можете воспользоваться командой:

\* sudo apt install openssh-server

TCP порт 80

TCP порт 10050 - для работы Zabbix Agent.

TCP порт 10051 - для работы Zabbix Trapper.

Настройка sudo без пароля для пользователя ivan

**sudo visudo**

ivan ALL=(ALL:ALL) NOPASSWD:ALL

**apt install net-tools** --> для ifconfig

ip a

lo

enp0s3

enp0s8

**nano /etc/netplan/00-installer-config.yaml**

network:

ethernets:

enp0s3:

dhcp4: true

enp0s8:

dhcp4: true

version: 2

https://losst.pro/nastrojka-seti-netplan-v-ubuntu - материал

**netplan --debug generate**

**netplan apply**

* **Установка Ansible**

# apt-get install python-yaml python-jinja2 python-paramiko python-crypto

**apt-add-repository ppa:ansible/ansible**

**apt update**

**apt install ansible**

Файл конфигураций /etc/ansible/ansible.cfg

**nano /etc/ansible/hosts**

ssh-keygen -t rsa

* **Установка Git**

**add-apt-repository ppa:git-core/ppa**

**apt update**

**apt install git**

В директории /etc/ansible выполняем команду :

**git clone** [**https://github.com/MakeUsWhole/ansible\_playbook.git**](https://github.com/MakeUsWhole/ansible_playbook.git)

* **Установка Docker**

**sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common**

**curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -**

**sudo printf "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable \n" > /etc/apt/sources.list.d/docker.list**

**sudo apt-get update**

**sudo apt-get install docker-ce docker-ce-cli containerd.io**

Настройка финальная

**sudo usermod -aG docker ivan**

**sudo systemctl enable docker.service**

**sudo systemctl enable containerd.service**

* **Установка Docker-compose**

**curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose**

**chmod +x /usr/local/bin/docker-compose**

Ошибка ! **cgroup mountpoint does not exist: unknown** при запуске контейнера. Фикс:

**sudo mkdir /sys/fs/cgroup/systemd  
sudo mount -t cgroup -o none,name=systemd cgroup /sys/fs/cgroup/systemd**

* **Установка Zabbix в Docker-compose**

**cd /opt/**

последняя версия (6.2)

**git clone** [**https://github.com/zabbix/zabbix-docker**](https://github.com/zabbix/zabbix-docker)

если нужна стабильная 6 версия:

**git clone -b 6.0** [**https://github.com/zabbix/zabbix-docker**](https://github.com/zabbix/zabbix-docker)

переходим в директорию **cd /opt/zabbix-docker**/

Здесь много docker-compose файлов, выбираем подходящий, либо редактируем под себя. Моя версия далее будет называться **mydocker.yaml**

В котором используются:

zabbix-server:

image: zabbix/zabbix-server-pgsql:ubuntu-6.2-latest

zabbix-web-nginx-pgsql:

image: zabbix/zabbix-web-nginx-pgsql:ubuntu-6.2-latest

zabbix-agent:

image: zabbix/zabbix-agent:ubuntu-6.2-latest

postgres-server:

image: postgres:14-alpine

Полное содержание **mydocker.yaml** можно посмотреть и скачать тут: [**https://github.com/MakeUsWhole/docker-master.git**](https://github.com/MakeUsWhole/docker-master.git)

Если вам необходимо установить какие-то нестандартные значения параметров, то сделать это можно в env файлах, которые находятся в директории **./env\_vars/.**

* **Настройка systrmd сервиса для автозапуска контенеров**

Создаем файл /etc/systemd/system/zabbix-compose.service и вставляем в него следующее содержимое:

|  |
| --- |
| [Unit]  Description=Zabbix services with docker-compose  Requires=docker.service  After=docker.service    [Service]  WorkingDirectory=/opt/zabbix-docker/  User=root  Group=docker  Type=oneshot  RemainAfterExit=yes    ExecStartPre=/usr/local/bin/docker-compose -f ./mydocker.yaml down -v    # Compose up  ExecStart=/usr/local/bin/docker-compose -f ./ mydocker.yaml up -d    # Compose down, remove containers  ExecStop=/usr/local/bin/docker-compose -f ./ mydocker.yaml down    [Install]  WantedBy=multi-user.target |

Теперь выполним следующие команды:

|  |
| --- |
| **systemctl enable zabbix-compose.service**  **systemctl daemon-reload**  **systemctl start zabbix-compose.service** |

* **Настройка Агента zabbix**

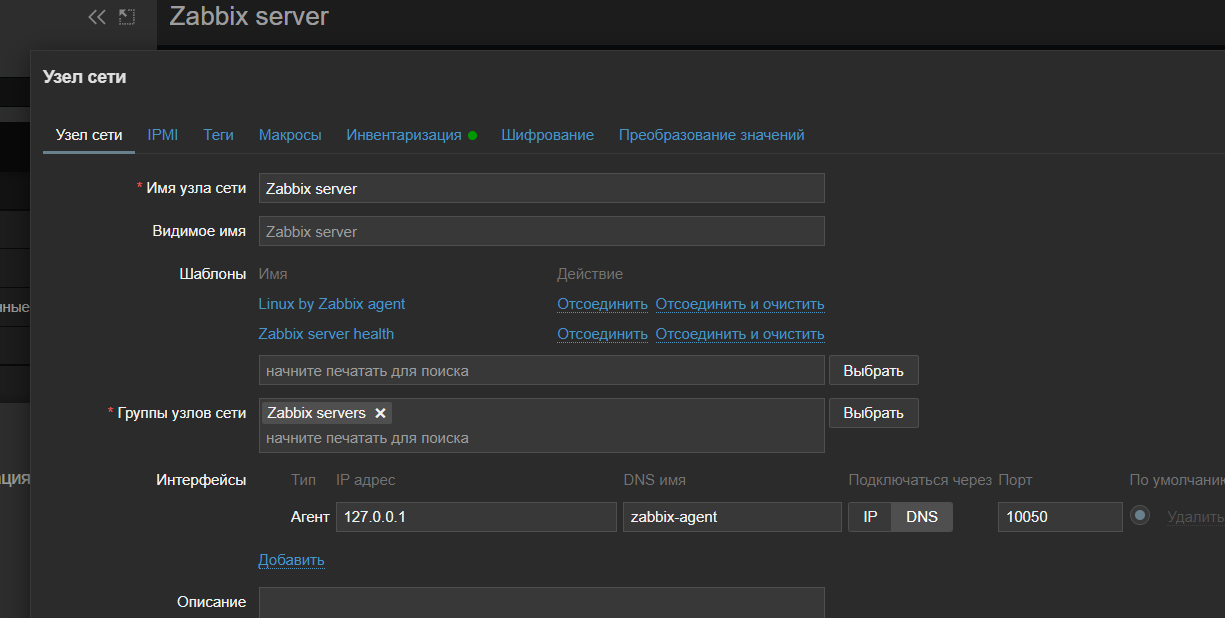
Столкнулся с проблемой zabbix-server и zabbix-agent локально не видят друг друга. Решил так:

Во-первых проверил что в .env\_agent задано ZBX\_SERVER\_HOST=zabbix-server

В docker-compose в zabbix-agent есть параметры

container\_name: zabbix-agent  
hostname: zabbix-agent  
links:  
- zabbix-server:zabbix-server  
depends\_on:  
- zabbix-server

В админке zabbix при добавлении нового узла сети в "Имя узла" стоит "zabbix-agent" ("Видимое имя" я поставил "zabbix-server"), в "Интерфейсы агента" указано zabbix-agent и поставлен переключатель IP/DNS в положение "DNS"



Если не заработало - подключаемся к контейнеру с zabbix-сервером и в нем пробуем пинговать zabbix-агент:

zabbix\_get -s zabbix-agent -k agent.ping

В ответ должно прийти "1" (есть коннект). Если этого не произошло - гуглим ошибку. Или не правильно настроили конфиг агента, или что-то не так с сетью. Или в имени узла указали не то, что прописано в /etc/hostname (это решается с помощью

hostname: zabbix-agent

из п.2)

**Источники:**

https://github.com/zabbix/zabbix-docker

https://techoverflow.net/2020/10/24/create-a-systemd-service-for-your-docker-compose-project-in-10-seconds/

https://github.com/docker/compose/issues/4266

<https://www.youtube.com/watch?v=ScKlF0ICVYA>

1. **Установка GLPI в Docker-compose**

На сервере уже должен быть предустановлен Docker и Docker-compose.

Для установки используем *docker-compose.yml* и *mariadb.env* файл с переменными. Переменные можно задавать редактируя *mariadb.env*  :

* MariaDB root password (пароль от root ) MARIADB\_ROOT\_PASSWORD=diouxx
* GLPI database (база данных) MARIADB\_DATABASE=glpidb
* GLPI user database (пользователь базы данных) MARIADB\_USER=glpi\_user
* GLPI user password (пароль пользователя базы данных) MARIADB\_PASSWORD=glpi

**mariadb.env**

MARIADB\_ROOT\_PASSWORD=diouxx

MARIADB\_DATABASE=glpidb

MARIADB\_USER=glpi\_user

MARIADB\_PASSWORD=glpi

**docker-compose .yml**

version: "3.2"

services:

#MariaDB Container

mariadb:

image: mariadb:10.7

container\_name: mariadb

hostname: mariadb

volumes:

- /var/lib/mysql:/var/lib/mysql

env\_file:

- ./mariadb.env

restart: always

#GLPI Container

glpi:

image: diouxx/glpi

container\_name : glpi

hostname: glpi

ports:

- "80:80"

volumes:

- /etc/timezone:/etc/timezone:ro

- /etc/localtime:/etc/localtime:ro

- /var/www/html/glpi/:/var/www/html/glpi

environment:

- TIMEZONE=Europe/Brussels

restart: always

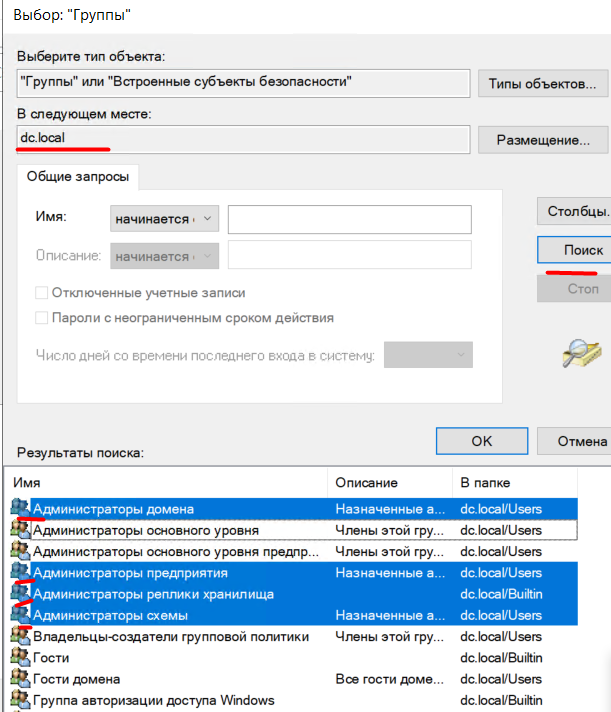
Для запуска выполнить команду:

**docker-compose up -d**

1. **Настройка серверов**

* **Настройка AD DNS Сервера**

Потребуются права для дальнейших настроек , потребуется перезагрузка сервера:



Предпочтительные DNS сервера

195.210.46.195

195.210.46.132

212.19.149.178

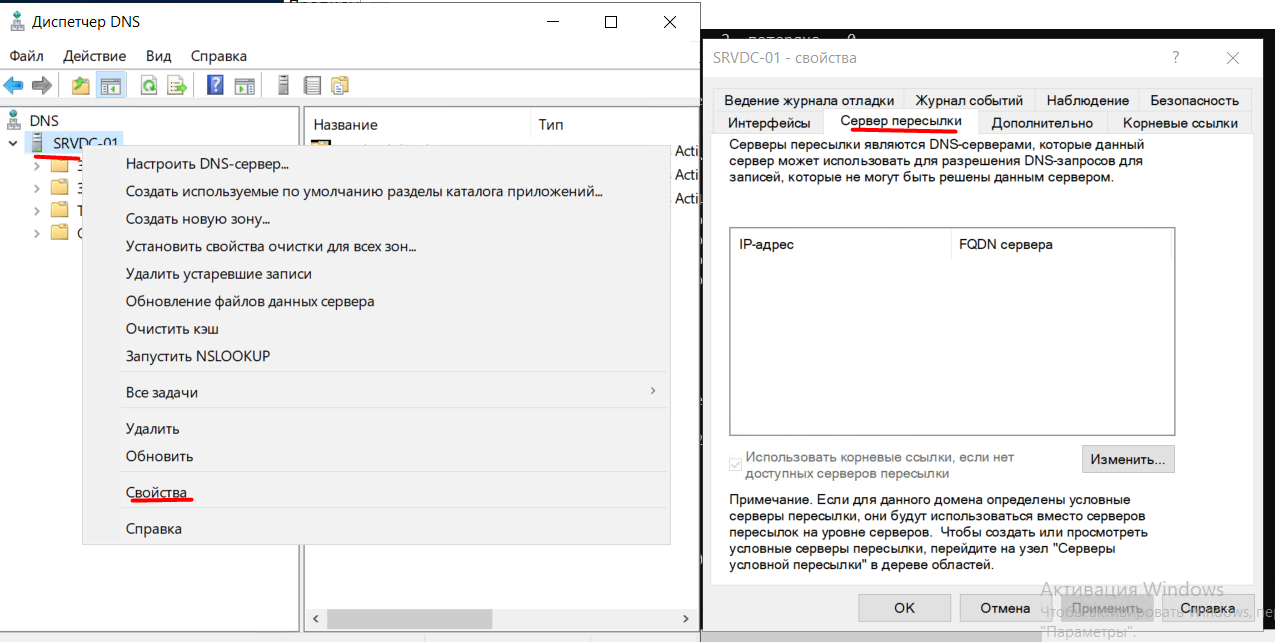
Прописываем роуты для них указав свой шлюз

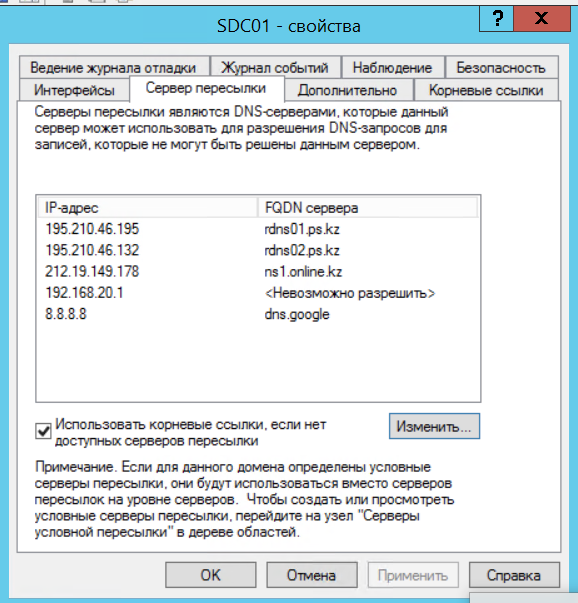
**route add 195.210.46.195 mask 255.255.255.255 172.16.10.1 -p**

**route add 195.210.46.132 mask 255.255.255.255 172.16.10.1 -p**

**route add 212.19.149.178 mask 255.255.255.255 172.16.10.1 -p**

И добавляем в Сервер пересылки:





Пингуем (ping ntp0.ntp-servers.net) NTP сервера ,чтобы узнать IP

ntp0.ntp-servers.net 88.147.254.227

ntp1.ntp-servers.net 109.195.19.73

0.pool.ntp.org 80.241.0.72

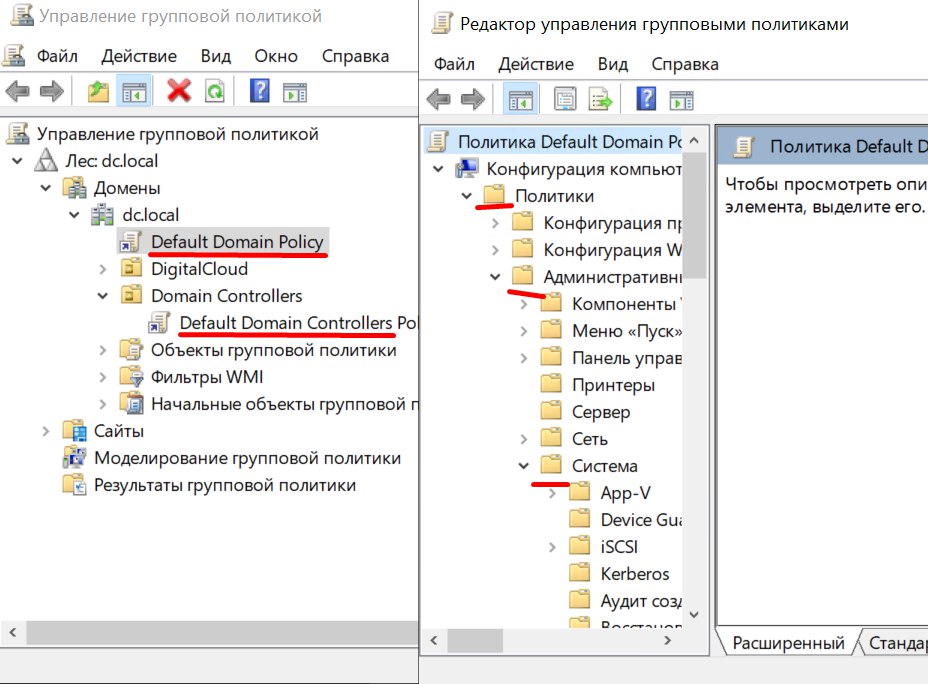
Прописываем роуты для NTP серверов

**route add 88.147.254.227 mask 255.255.255.255 172.16.10.1 -p**

**route add 109.195.19.73 mask 255.255.255.255 172.16.10.1 -p**

**route add 80.241.0.72 mask 255.255.255.255 172.16.10.1 -p**

**gpmc.msc**



Политика - Административные шаблоны - Система - Служба времени Windows - Поставщик времени

(нужно все отключить)

**gpedit.msc**

Политика Локальный компьютер - Административные шаблоны - Система - Служба времени Windows - Поставщик времени

(нужно все отключить)

**net stop w32time**

**w32tm.exe /unregister**

**w32tm /register**

**net start w32time**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**w32tm /config /update /manualpeerlist:"ntp0.ntp-servers.net,0x8 ntp1.ntp-servers.net,0x8 0.pool.ntp.org,0x8 3.pool.ntp.org,0x8" /syncfromflags:MANUAL /reliable:YES**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**w32tm /config /update**

**w32tm /query /peers**

**w32tm /query /status**

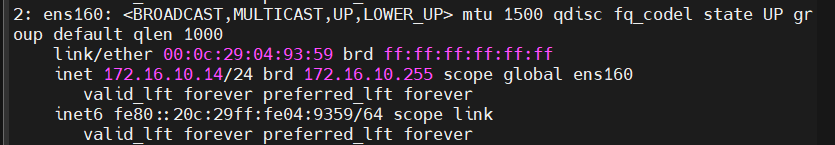
=========================

* **Если сервер находится в корпоративной сети со своим DNS**

Часто в корп. Сети есть свой DNS сервер и определенные ограничения. Все сервера должны ссылаться на него, что приводит к ряду ошибок при работе Docker контейнеров.

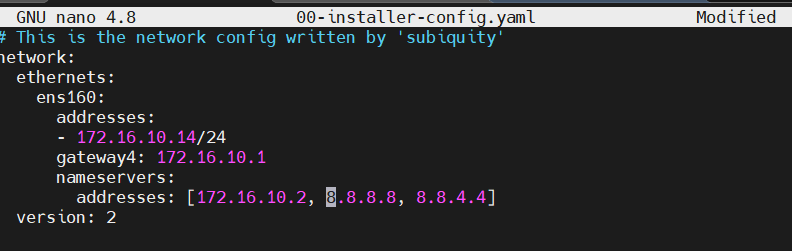
Чтобы исправить ситуация начнем с настройки сети на самом сервере, затем в докере.

Первым делом смотрим доступные сетевые интерфейсы **ip a** или **ifconfig**



В примере ens160 интерфейс настроен на статический ip 172.16.10.14

В Ubuntu 20 система скриптов ifup/ifdown заменена программой netplan, со своими конфигурационными файлами на языке YAML — /etc/netplan/имяфайла.yaml



В примере интерфейс настроен работать через шлюз 172.16.10.1 и использует 3 адреса DNS серверов.

После внесения изменения в .yaml файл делаем проверку и запись:

**netplan generate**

**и применение**

**netplan apply**

Следующим делом нужно проверить **cat /etc/resolv.conf**

nameserver 172.16.10.2

nameserver 8.8.8.8

Здесь так же должны быть прописаны DNS сервера , в том числе и корпоративный .

Осталось настроить Docker

Указаны DNS серверы в файле конфигурации Docker - /etc/docker/daemon.json

Создадим этот файл и добавим туда следующие строки: **nano /etc/docker/daemon.json**

**{**

**"dns": ["172.16.10.2", "8.8.8.8", "8.8.8.4"]**

**}**

Перезапустим докер

**service docker restart**

Запустим контейнер и проверим /etc/resolv.conf

root@ca005a605c62:/# **cat /etc/resolv.conf**

nameserver 172.16.10.2

nameserver 8.8.8.8

Так же есть настройки по умолчанию , чтобы данный способ работал , тут не стоит ничего менять

/etc/default/docker

Материал:

<https://www.tune-it.ru/web/adpashnin/blog/-/blogs/nastrojka-dns-v-docker>

<https://parallel.uran.ru/book/export/html/442>

<https://losst.pro/nastrojka-seti-netplan-v-ubuntu>

1. **Гипервизор VMware vSphere ESXi**

* **Установка и настройка**

Лучшей практикой будет установка на флешку установленную во внутренний порт сервера.

1. Понадобится установочная флешка с esxi , iso файл можно найти здесь:

<https://archive.org/details/VMware-VMvisor-Installer-6.7.0.update03-14320388.x86_64>

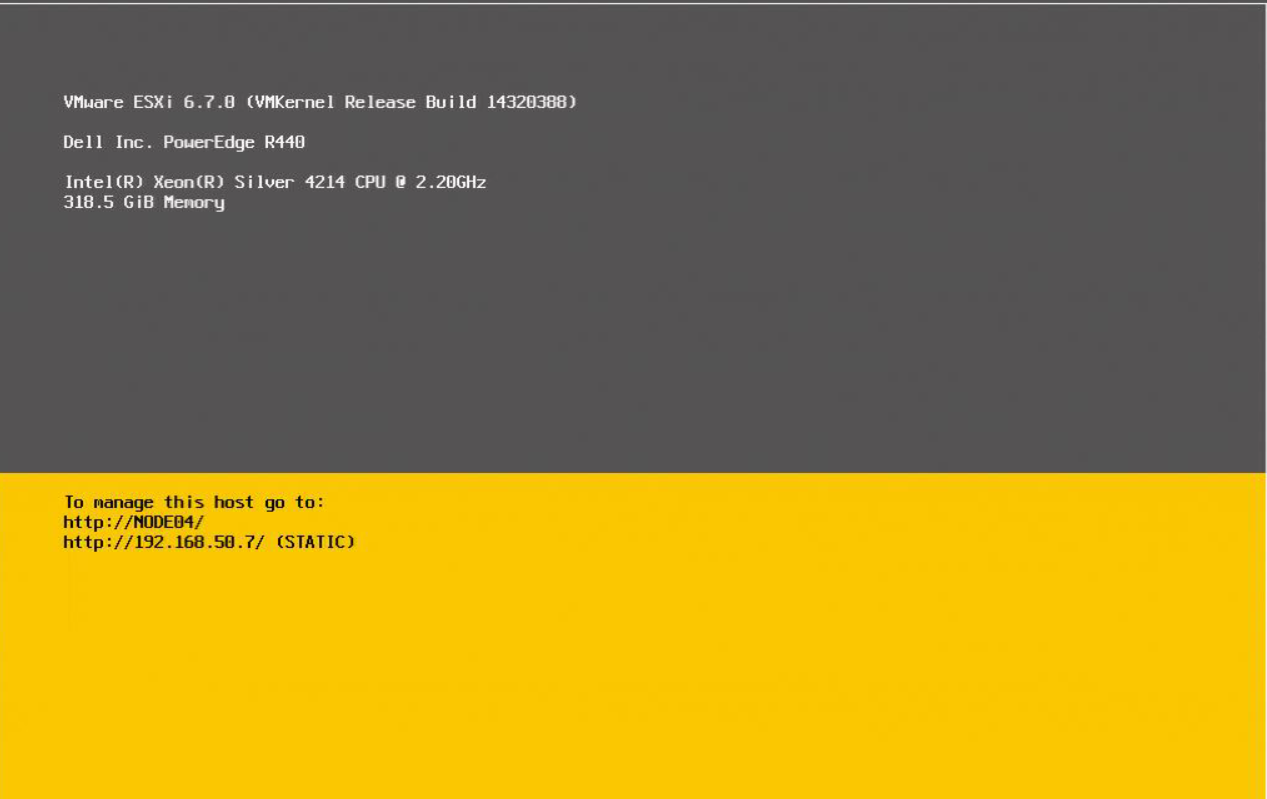
Для создания установочной флешки используем Rufus

<https://rufus.ie/ru/>

В процессе установки выбираем носителем нашу флешку , установленную во внутренний порт сервера. В остальном процесс хорошо описан здесь:

<https://wiki.merionet.ru/servernye-resheniya/57/ustanovka-vmware-vsphere-esxi-6-7/>

После установки по заданному IP будет доступен вэб интерфейс управления гипервизором



В примере это : <http://192.168.50.7> , внимательно ,ваш сервер должен находиться в той же подсети ,чтобы подключить вэб интерфейсу ( то есть в 192.168.50.0)

* **Установка и настройка vCenter**

Потребуется настройка DNS . Подробно процесс описан и показан в данных материалах.

Материал на тему:

<https://internet-lab.ru/vcenter_6_7_install>

<https://yandex.kz/video/preview/8895059860144744212>

* **VMware vCenter Converter Standalone для копирования виртуальных машин**

VMware vCenter Converter Standalone

Позволяет переносить виртуальные машины с одного сервера гипервизора VMware а другой .

Виртуальная машина должна быть отключена , после завершения задачи у нас появится точная копия машины на указанном VMware сервере.

