# Лабораторная работа №12

**Отчет**

Мосолов Александр Денисович

# Содержание

# Список иллюстраций

# 1 Цель работы

Получение практических навыков по управлению системным временем, часовыми поясами, а также настройке протокола сетевого времени (NTP) с использованием службы chrony в среде Linux.

# 2 Задание

1. Изучить и применить команды настройки времени (`timedatectl`, `date`, `hwclock`).
2. Настроить серверную виртуальную машину в качестве NTP-сервера для локальной сети.
3. Настроить клиентскую машину на синхронизацию времени с локальным сервером.
4. Автоматизировать процесс настройки через скрипты Provisioning в Vagrant.

# 3 Выполнение лабораторной работы

## 3.1 Настройка параметров времени

В начале работы я проверяю текущее состояние подсистемы времени на сервере. Команда `timedatectl` выводит детальную информацию: * **Local time**: Текущее системное время (в данном случае совпадает с UTC). * **Universal time**: Время UTC. * **RTC time**: Время в аппаратных часах (Real Time Clock). * **Time zone**: Текущий часовой пояс (UTC). * **System clock synchronized**: `yes` указывает, что служба NTP активна и синхронизирует время (рис. 3.1).

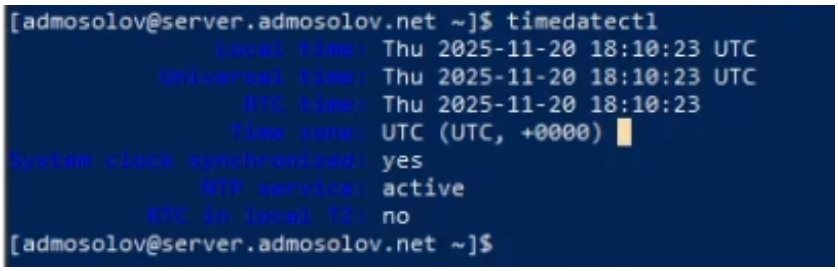


Рис. 3.1: Проверка статуса времени на сервере

Аналогичную проверку провожу на клиенте. Здесь уже установлен часовой пояс `Europe/Moscow` (+0300). Заметно, что локальное время отличается от UTC на +3 часа, что соответствует настройке зоны. Параметр `RTC in local TZ: no` говорит о том, что аппаратные часы хранят время в UTC, что является стандартом для Unix-систем (рис. 3.2).

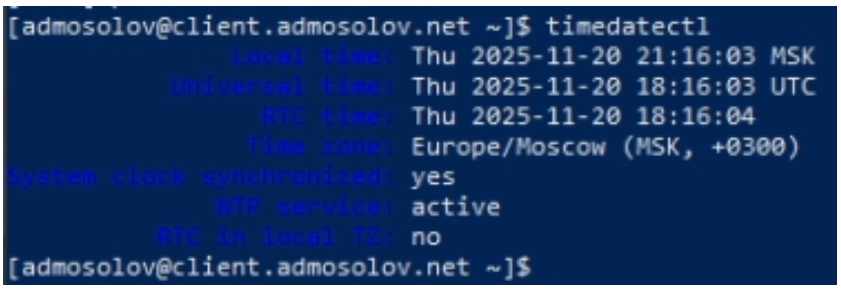


Рис. 3.2: Проверка статуса времени на клиенте

Для изменения часового пояса сначала выполняю поиск доступных зон с помощью `list-timezones` и фильтра `grep`. Найдя зону `Europe/London`, применяю её командой `set-timezone`. Это действие изменяет смещение локального времени относительно UTC, но не меняет само значение UTC (рис. 3.3).

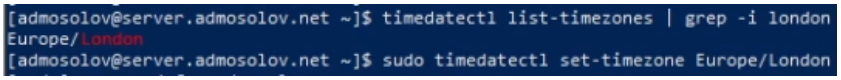


Рис. 3.3: Поиск и смена часового пояса на Лондон

Возвращаю корректный для работы часовой пояс `Europe/Moscow`. После выполнения команды вывод `timedatectl` подтверждает, что зона изменена на MSK (+0300), а локальное время пересчитано соответствующим образом (рис. 3.4).

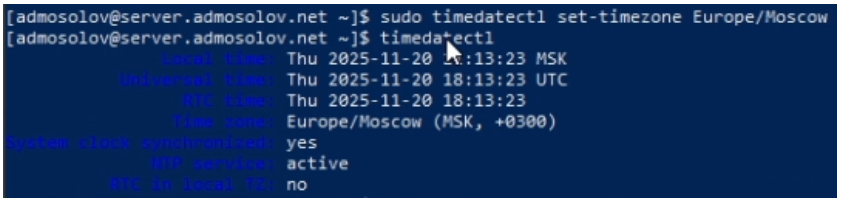


Рис. 3.4: Возврат часового пояса Europe/Moscow

Команда `date` без аргументов выводит текущее системное время с учетом установленной временной зоны. На сервере это выглядит следующим образом (рис. 3.5).

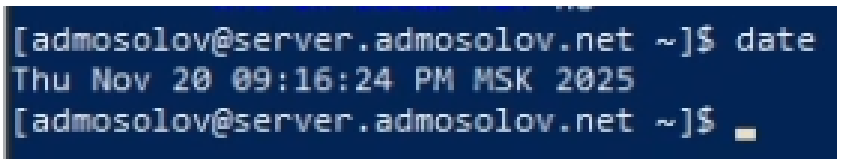


Рис. 3.5: Вывод команды date на сервере

На клиенте выполняю ту же команду. Сравнение показывает, что время на обеих машинах синхронизировано (с точностью до секунд, учитывая разницу во времени ввода команд) (рис. 3.6).

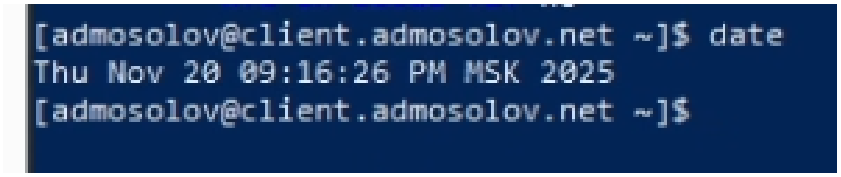


Рис. 3.6: Вывод команды date на клиенте

Команда `date` обладает широкими возможностями форматирования. Я протестировал следующие ключи форматирования на сервере: * `+%Y-%m-%d`: вывод только даты. * `+%H:%M:%S.%3N`: вывод времени с точностью до миллисекунд. * `-R`: вывод в формате RFC 2822 (используется в заголовках email). * `-u`: вывод времени UTC независимо от локальной настройки. * Переопределение переменной окружения `TZ` позволяет вывести время в другой зоне без изменения системных настроек (рис. 3.7).

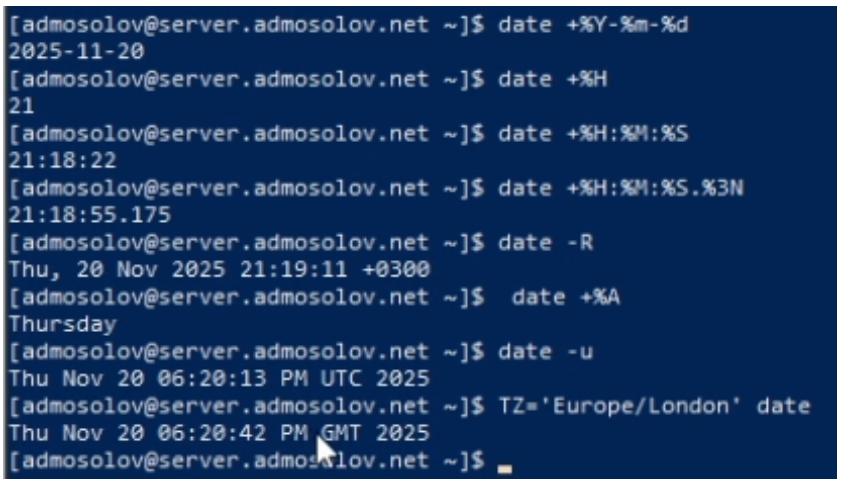


Рис. 3.7: Примеры форматирования date на сервере

Те же операции форматирования были проверены на клиентской машине для подтверждения идентичности функционала (рис. 3.8).

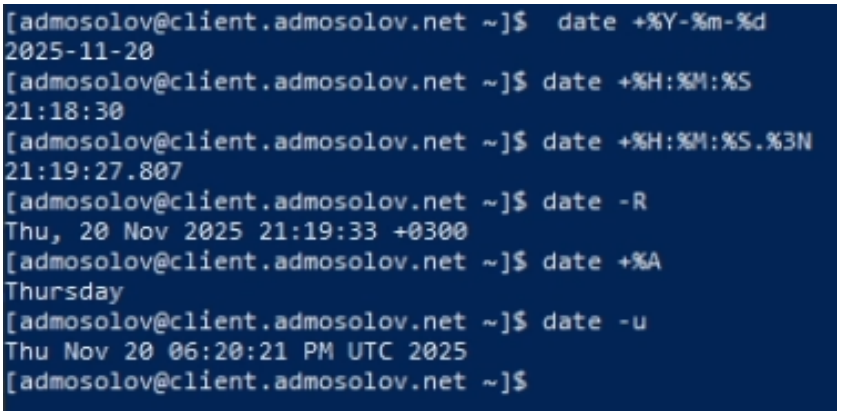


Рис. 3.8: Примеры форматирования date на клиенте

Команда `date` также позволяет выполнять арифметические операции с датами, что полезно при написании скриптов (например, для ротации логов). Я использовал флаг `--date` для вывода даты следующего дня, даты через неделю или неделю назад (рис. 3.9).

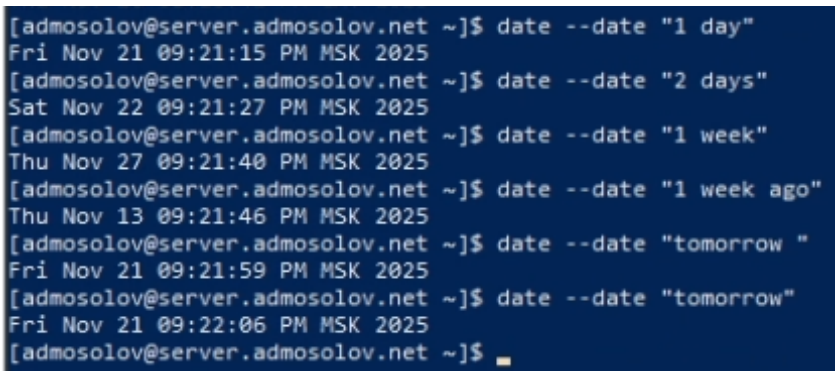


Рис. 3.9: Арифметика времени с командой date

Утилита `hwclock` управляет аппаратными часами (RTC), которые работают от батарейки на материнской плате. Команда показывает время, хранящееся в чипе. Вывод подтверждает, что аппаратные часы идут по UTC, а ядро Linux при загрузке считывает это время и применяет смещение часового пояса (рис. 3.10).

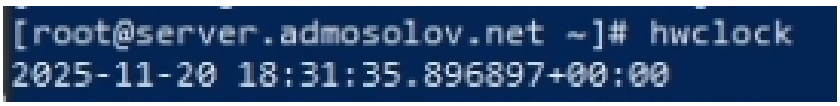


Рис. 3.10: Проверка аппаратных часов на сервере

Проверка `hwclock` на клиенте показывает аналогичный результат. Разница между системным временем (date) и аппаратным (hwclock) обусловлена программной коррекцией дрейфа часов ядром ОС (рис. 3.11).

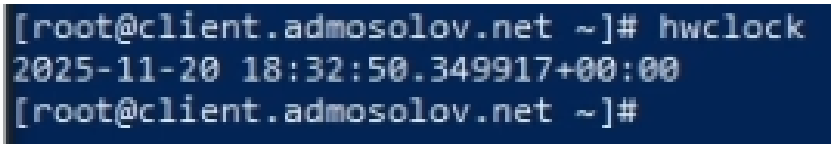


Рис. 3.11: Проверка аппаратных часов на клиенте

## 3.2 Управление синхронизацией времени

Для организации NTP-сервера использую пакет `chrony`. На сервере проверяю его наличие через пакетный менеджер `dnf`. Пакет уже установлен в системе (рис. 3.12).

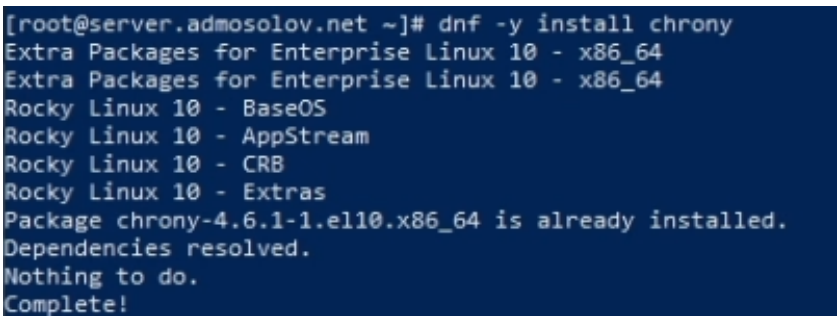


Рис. 3.12: Установка пакета chrony

До начала настройки я проверяю текущие источники времени командой `chronyc sources`. **Пояснение вывода:** * В выводе присутствуют внешние сервера (например, `dot.kkursor.ru`), но статус соединения показывает ошибку. * Сообщение `506 Cannot talk to daemon` внизу означает, что служба `chronyd` в данный момент остановлена или недоступна, поэтому клиент `chronyc` не может получить от нее данные (рис. 3.13).

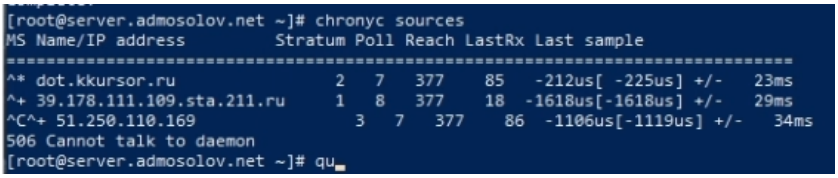


Рис. 3.13: Проверка источников (служба остановлена)

Для того чтобы сервер начал отдавать время клиентам локальной сети, редактирую конфигурационный файл `/etc/chrony.conf`. Директива `allow 192.168.0.0/16` разрешает входящие NTP-запросы от всех узлов подсети 192.168.x.x. Без этой строки сервер будет работать только как клиент внешних источников (рис. 3.14).

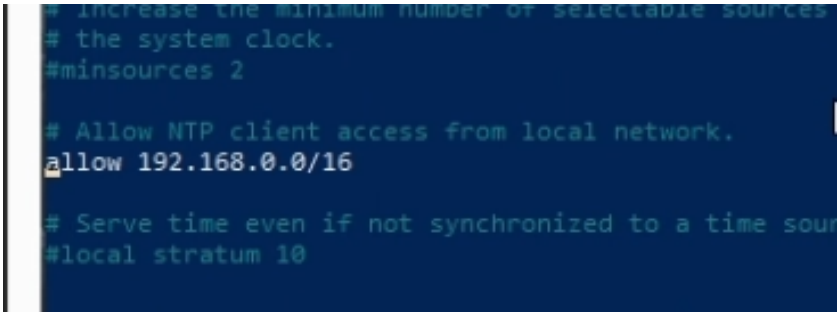


Рис. 3.14: Настройка доступа в chrony.conf

Чтобы применить настройки, перезапускаю службу `chronyd`. Также необходимо открыть порт 123 (UDP) в брандмауэре, так как протокол NTP использует именно его. Команда `--permanent` сохраняет правило после перезагрузки (рис. 3.15).

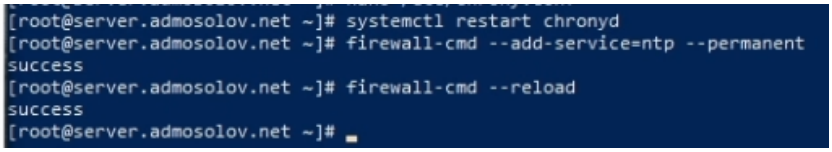


Рис. 3.15: Перезапуск службы и открытие порта firewall

Перехожу к настройке клиента. В файле `/etc/chrony.conf` указываю адрес моего сервера: `server server.admosolov.net iburst`. **Пояснение:** * Опция `iburst` (Initial Burst) критически важна: она заставляет клиент отправить пачку из 8 пакетов при запуске службы, что позволяет синхронизировать время за несколько секунд, вместо стандартного ожидания в несколько минут (рис. 3.16).

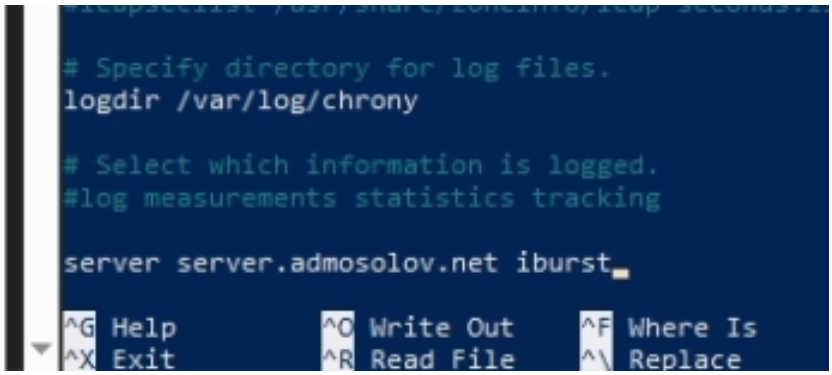


Рис. 3.16: Настройка клиента на локальный сервер

После перезапуска службы на клиенте снова выполняю `chronyc sources`. **Пояснение вывода:** * `M` (Mode): `^` означает сервер. * `S` (State): `*` означает, что данный источник (server.admosolov.net) выбран как текущий эталон времени. * `Stratum`: `3`. Это означает, что мой сервер имеет страту 2 (берет время от внешних источников страты 1), а клиент, соответственно, получает страту 3. * `Reach`: `37` (в восьмеричной системе) показывает успешность последних попыток опроса. * **Вывод:** Синхронизация с локальным сервером работает корректно (рис. 3.17).

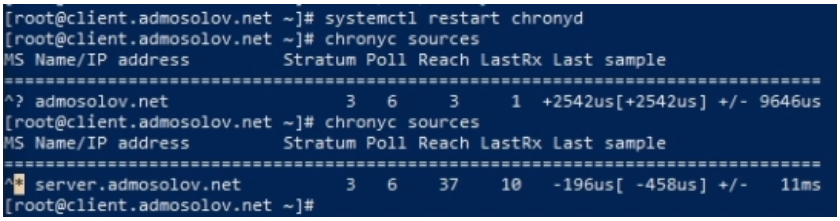


Рис. 3.17: Успешная синхронизация на клиенте

Для более глубокого анализа использую команду `chronyc tracking`. **Пояснение вывода:** * **Reference ID**: Идентификатор сервера (`server.admosolov.net`), с которым идет синхронизация. * **Stratum**: 4 (в данный момент, возможно сервер переключился на источник страты 3). Чем меньше число, тем ближе к эталонным атомным часам. * **System time**: Отклонение системных часов от NTP-времени (0.0011 секунды медленнее). * **Last offset**: Смещение, вычисленное при последнем опросе (-0.0009 секунды). * **Root delay/dispersion**: Накопленные задержки и погрешности от корня (stratum 1). Эти данные подтверждают, что часы клиента жестко привязаны к серверу с высокой точностью (рис. 3.18).

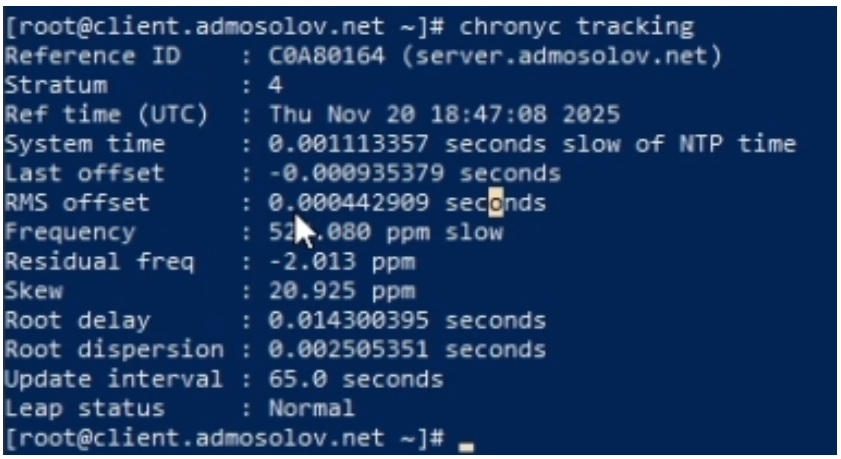


Рис. 3.18: Подробная статистика chronyc tracking

## 3.3 Автоматизация настройки (Provisioning)

Чтобы настройки не пропали при пересоздании виртуальных машин и для соблюдения практики “Инфраструктура как код”, я создаю скрипты настройки. На сервере подготавливаю структуру каталогов для хранения эталонного конфига (рис. 3.19).

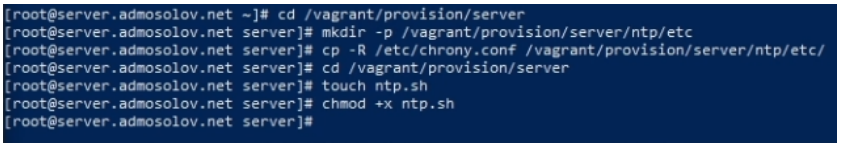


Рис. 3.19: Подготовка каталогов на сервере

Создаю скрипт `ntp.sh` для сервера. Он выполняет: установку chrony, копирование заранее подготовленного конфига в `/etc`, восстановление контекста SELinux (`restorecon`), настройку firewall и перезапуск службы (рис. 3.20).

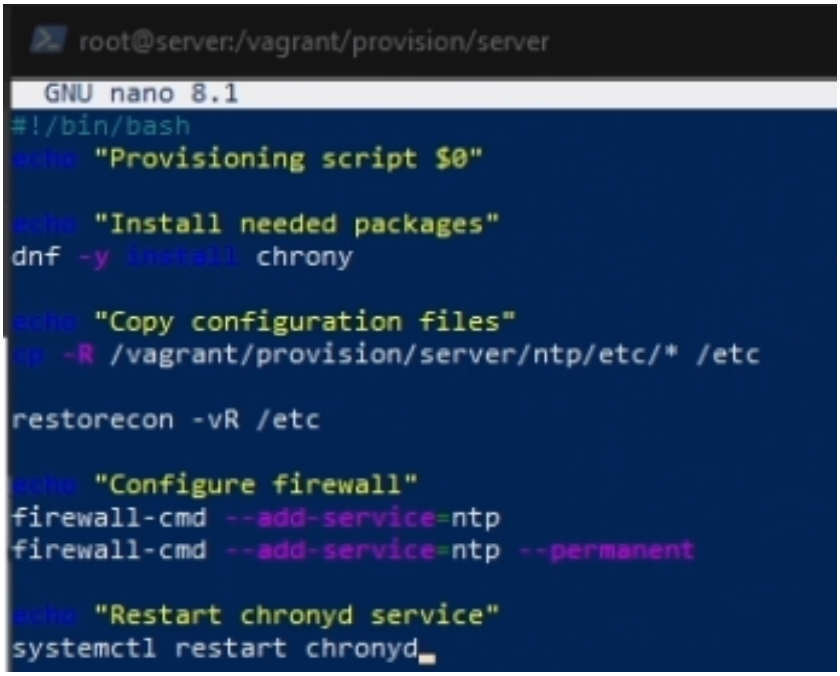


Рис. 3.20: Скрипт настройки сервера ntp.sh

Аналогичные действия провожу на клиенте. Создаю каталог для хранения клиентской версии `chrony.conf` (рис. 3.21).

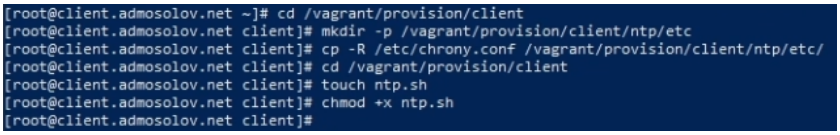


Рис. 3.21: Подготовка каталогов на клиенте

Скрипт клиента `ntp.sh` проще: он копирует конфигурационный файл (где прописан локальный сервер) и перезапускает службу (рис. 3.22).

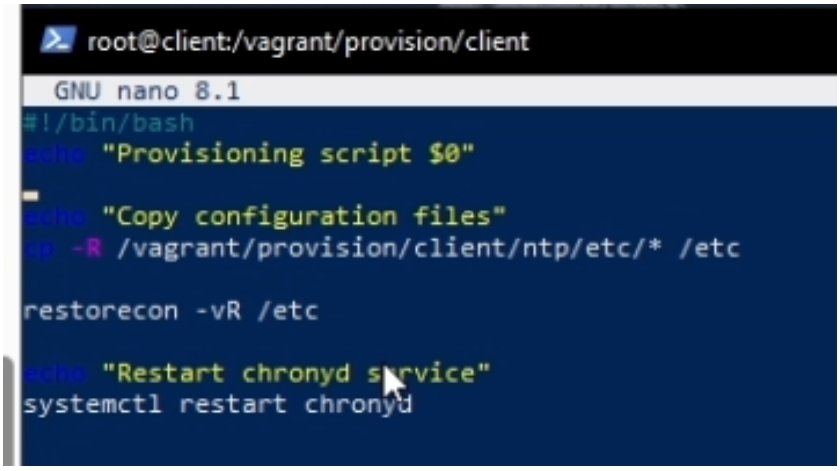


Рис. 3.22: Скрипт настройки клиента ntp.sh

В завершение, подключаю эти скрипты в `Vagrantfile`. Для сервера добавляю блок `vm.provision` типа `shell`, указывая путь к скрипту. Это гарантирует, что при команде `vagrant up` сервер будет настроен автоматически (рис. 3.23).

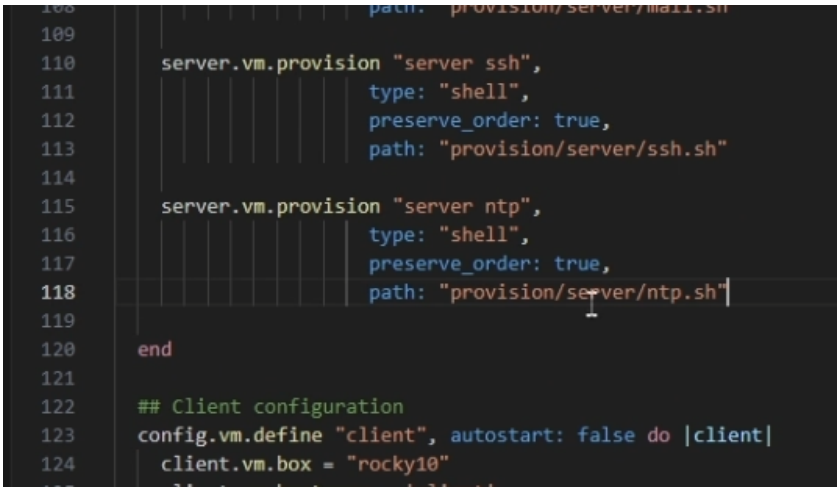


Рис. 3.23: Provisioning для сервера в Vagrantfile

Аналогичную запись добавляю для клиента. Теперь вся лабораторная работа полностью автоматизирована и воспроизводима (рис. 3.24).

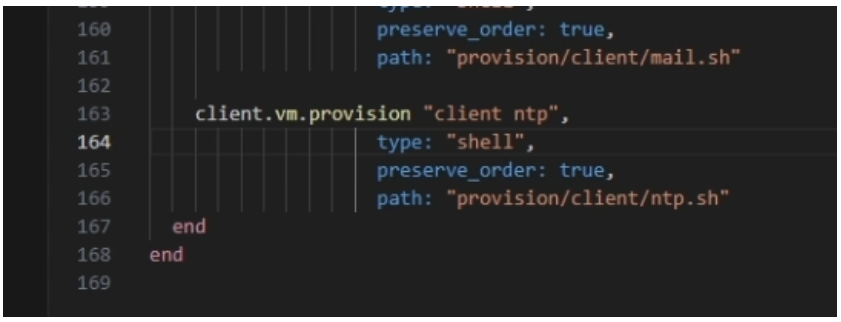


Рис. 3.24: Provisioning для клиента в Vagrantfile

# 4 Выводы

В ходе лабораторной работы я научился: 1. Диагностировать и изменять настройки системного времени и часовых поясов в Linux (`timedatectl`). 2. Использовать команду `date` для отображения, форматирования и вычисления дат. 3. Понимать разницу между системным временем и аппаратными часами (`hwclock`). 4. Настраивать службу `chronyd`: * Конфигурировать сервер для обслуживания локальной сети (директива `allow`). * Настраивать клиент для синхронизации с конкретным сервером и использования режима `iburst`. * Интерпретировать вывод команд диагностики `chronyc sources` и `chronyc tracking`. 5. Настраивать правила межсетевого экрана (`firewall-cmd`) для пропуска NTP-трафика. 6. Писать Bash-скрипты для автоматической настройки сервисов и интегрировать их в Vagrant.

# 5 Ответы на контрольные вопросы

1. **Почему важна точная синхронизация времени для служб баз данных?** Для обеспечения целостности транзакций (ACID), корректной работы механизмов репликации (определение последней версии данных), правильного восстановления из журналов (logs) и временных меток (timestamps) при записи данных. Рассинхронизация может привести к потере данных или нарушению логики работы распределенных БД.

2. **Почему служба проверки подлинности Kerberos сильно зависит от правильной синхронизации времени?** Протокол Kerberos использует метки времени в выдаваемых билетах (tickets) для предотвращения “атак повторного воспроизведения” (replay attacks). Если время на клиенте и KDC (Key Distribution Center) различается больше, чем на допустимое значение (обычно 5 минут), билет считается недействительным, и аутентификация отклоняется.

3. **Какая служба используется по умолчанию для синхронизации времени на RHEL 7?** Служба `chronyd` (демон Chrony). Она заменила устаревший `ntpd` в качестве решения по умолчанию.

4. **Какова страта по умолчанию для локальных часов?** Если сервер не синхронизирован с внешним источником, его локальным часам обычно присваивается высокая страта (например, 10), чтобы клиенты предпочитали другие, более точные источники, если они доступны. В конфиге это задается директивой `local stratum 10`.

5. **Какой порт брандмауэра должен быть открыт, если вы настраиваете свой сервер как одноранговый узел NTP?** Порт **123** протокола **UDP**. NTP использует этот порт как для отправки, так и для получения запросов времени.

6. **Какую строку вам нужно включить в конфигурационный файл chrony, если вы хотите быть сервером времени, даже если внешние серверы NTP недоступны?** `local stratum 10`. Эта команда предписывает Chrony продолжать обслуживать клиентов, используя локальные системные часы как эталон, даже если связь с вышестоящими NTP-серверами потеряна (“orphaned mode”).

7. **Какую страту имеет хост, если нет текущей синхронизации времени NTP?** Формально, если синхронизация отсутствует, страта не определена или считается равной 16 (бесконечность/недостижимость в терминологии NTP), что означает, что время данного хоста недостоверно.

8. **Какую команду вы бы использовали на сервере с chrony, чтобы узнать, с какими серверами он синхронизируется?** `chronyc sources` (для краткого списка) или `chronyc sources -v` (для списка с пояснениями заголовков столбцов).

9. **Как вы можете получить подробную статистику текущих настроек времени для процесса chrony вашего сервера?** `chronyc tracking` — эта команда выводит системную информацию о процессе синхронизации: идентификатор эталона, страту, текущее смещение, задержку, частоту коррекции и статус “Leap status”.