|  |  |
| --- | --- |
| 1) С какими дистрибутивами на базе ядра Linux приходилось работать? Назовите несколько принципиальных различий между ними. | Именно работать приходилось в основном с отечественными дистрибутивами: Astra Linux и ALT Linux.  В качестве домашних и образовательных дистрибутивов использовал чистый Debian, Mint, RedOS(в меньшей степени, пользовался больше из интереса).  Если брать принципиальные различия в общем понимании, то это целевое назначение этих систем – конечный пользователь. В одном случае это преимущественно отечественные предприятия, а в других - это свободно распространяемые модели, которые рассчитаны на более широкий круг пользователей.  Также есть принципиальные различия в построении файловой системы, синтаксиса команд и набора адаптированного дистрибутива(apps). |
| 2) Объясните разницу между абсолютными и относительными путями в linux. | Разница абсолютного и относительного путей заключается в расположении файла или каталога относительно текущего каталога.  То есть, простыми словами, абсолютный – это полный путь, от самого корневого каталога ( / ), а относительный это путь относительно текущего каталога. |
| 3) Объясните разницу в шебанг скрипта bash:  #! /bin/bash и  #! /usr/bin/env bash | Разница шебанга в способе обращения к запуску bash. В первом случае обращение идёт напрямую, а во втором происходит поиск bash через окружение системы. |
| 4) Есть три процесса в linux:  - №1, занимающий 500 МБ оперативной памяти со временем жизни 10 минут  - №2, занимающий 1 ГБ оперативной памяти со временем жизни 30 минут  - №3, занимающий 250 МБ оперативной памяти со временем жизни 60 минут  При недостатке ресурсов, какой процесс ООМ Killer остановит первым? | Скорее всего остановится процесс №2, поскольку занимает больше всего памяти, но при этом не самое большое время жизни. |
| 5) Есть ли опыт администрирования брокеров сообщений?  Если есть и ответ "Apache Kafka" - вопрос - в чем суть нового решения в kafka под названием kraft? | Этот блок я объединю и дам единый ответ.  С Kafka я не работал, но имею представление что это, зачем нужно и как работает.  Круглогодичные системы - не администрировал. Автоматизация задач, которая была необходима в работе – небольшие скрипты на баше или сценарии Ansible, которые просто автоматизируют обновление или установку того или иного дистрибутива, либо осуществляет подмену или создания того или иного файла конфига. Всё это затрагивало только машины простых пользователей, без автоматизации работы серверов.  Никаких глобальных скриптов автоматизации создавать в работе не приходилось.  Мониторинг в рабочих процессах осуществляется только посредством Zabbix. Доводилось создавать не слишком сложные дэшборды, создавать триггеры для для мониторинга маршрутизаторов, в частности – Eltex. В остальном же использовал по большей части либо готовые пользовательские пресеты.  Если подытожить, то с инструментами DevOps именно в рабочей деятельности мне сталкиваться не доводилось, поскольку работаю всё же в области более близкой к системному администрированию, за исключением Zabbix и Ansible.  Более обширные знания по этой тематике у меня есть только за счет самостоятельного обучения.  Так, например, доводилось разворачивать отказоустойчивую облачную инфраструктуру на Яндекс.Cloud, с использованием terraform, ansible, Prometheus, Grafana, ELK стека.  Не скажу, что в каждом из этих инструментов я глубоко разбираюсь, но разворачивать, настраивать базовые конфиги и взаимодействие между друг другом смогу. |
| 6) Был ли опыт администрирования продуктовых систем, работающими в режиме 24/7/365. В чем заключались работы? |
| 7) Как вы подходите к автоматизации задач администрирования серверов? |
| 8) Расскажите о своем опыте работы с системами мониторинга |
| 9) Вы столкнулись аварией продуктового сервиса, связанного с его недоступностью пользователям. Опишите порядок ваших действий. | Если корректно настроен мониторинг доступности системы, то с его помощью попытаюсь понять масштаб проблемы, оповестить всех ответственных лиц. Если будет очевидно, что проблема именно с ПО, а не техническая, то проверю последние логи сервисов, либо в целом системы, что поможет выяснить в чем конфликт, и далее уже пытаться его решить в рамках своей компетенции. После восстановления убедиться в корректной работоспособности, в том числе и корректной работы созависимых сервисов.  Если же проблема техническая, то здесь скорее всего стоит эскалировать её техническим специалистам. |

|  |
| --- |
| Имеем три сервера с адресами:  10.0.0.1  10.0.0.2  10.0.0.3  Сервера связаны между собой ssh-ключами авторизации.  На каждом сервере имеется конфигурационный файл /opt/core/eapconfig.json вида:  {  "server.url": "http://localhost:5000", // для каждого сервера свой порт  "SignalRConfiguration": {  "url": "http://localhost:6000"  },  "license": {  "key\_name": "rsa-private-1.b8",  "license\_service": "Trs.License.Services.ILicenseService, Trs.License.Domain"  },  "databases": [  {  "name": "Primary Relational Database",  "categories": [  "Relational",  "Blob"  ],  "connection": "Host=localhost;Database=dev;Username=myname;Password=mypass",  "accessMode": "ReadWrite",  "model": {  "name": "Primary, Trs.Eap.Data.Model.Primary | Packages/",  "provider": "Trs.Eap.Data.EntityFramework.Provider, Trs.Eap.Data.EntityFramework.DataProvider",  "builder": "Trs.Eap.Data.EntityFramework.EfModelBuilder, Trs.Eap.Data.EntityFramework.ModelBuilder",  "storage": "Trs.Eap.Data.EntityFramework.Storage.PosgreSql`1, Trs.Eap.Data.EntityFramework.Storage.PostgreSql"  },  "usePublicSchema": false  }  ]  }  Необходимо создать bash-скрипт в соответствии с требованиями:  1. bash-скрипт должен принимать при запусках два строковых аргумента.  2. bash-скрипт должен запускаться на одном из серверов и изменять конфигурационный файл /opt/core/eapconfig.json локально и удаленно на двух других серверах.  3. bash-скрипт должен обрабатывать любую допустимую структуру json-файла (например, отсутствие табуляции, все содержимое в одну строку)  4. bash-скрипт должен удовлетворять требованию повторяемости (возможности повторных запусков без необходимости изменения структуры скрипта).  5. Что следует менять:  - аргумент №1 - значение json-ключа: key\_name объекта license  - аргумент №2 - значение json-ключа: model.name массива databases |
| Да, можно написать bash-скрипт, который выполнит задачу без использования дополнительных утилит вроде `jq`. В таком случае, для изменения значений в JSON-файле будем использовать стандартные утилиты, такие как `sed` и `awk`. Однако нужно учесть, что этот подход требует более аккуратного подхода к структуре JSON и может быть менее гибким.  Вот пример скрипта:  ```bash  #!/bin/bash  # Проверка наличия двух аргументов  if [ "$#" -ne 2 ]; then  echo "Usage: $0 <new\_key\_name> <new\_model\_name>"  exit 1  fi  # Аргументы  NEW\_KEY\_NAME=$1  NEW\_MODEL\_NAME=$2  # Массив серверов  SERVERS=("10.0.0.1" "10.0.0.2" "10.0.0.3")  CONFIG\_PATH="/opt/core/eapconfig.json"  # Функция для обновления конфигурации на сервере  update\_config() {  local server=$1  echo "Updating configuration on server $server..."  ssh $server << EOF  # Резервное копирование текущего конфигурационного файла  cp $CONFIG\_PATH $CONFIG\_PATH.bak  # Обновление ключа key\_name в объекте license  sed -i.bak 's|\"key\_name\": \"[^\"]\*\"|\"key\_name\": \"$NEW\_KEY\_NAME\"|' $CONFIG\_PATH  # Обновление значения model.name в массиве databases  sed -i.bak 's|\"name\": \"Primary, [^\"]\*\"|\"name\": \"$NEW\_MODEL\_NAME\"|' $CONFIG\_PATH  EOF  # Проверка успешности обновления  if [ $? -eq 0 ]; then  echo "Configuration updated successfully on server $server."  else  echo "Failed to update configuration on server $server."  fi  }  # Обновление конфигурации на всех серверах  for server in "${SERVERS[@]}"; do  update\_config $server  done  ```  ### Объяснение скрипта:  1. \*\*Проверка аргументов\*\*: Скрипт проверяет, были ли переданы два аргумента. Если нет, выводится сообщение об ошибке, и скрипт завершает работу.  2. \*\*Аргументы\*\*: Значения аргументов сохраняются в переменные `NEW\_KEY\_NAME` и `NEW\_MODEL\_NAME`.  3. \*\*Список серверов\*\*: Массив `SERVERS` содержит IP-адреса серверов.  4. \*\*Функция `update\_config`\*\*: Эта функция выполняет основные действия:  - Подключается к серверу через SSH.  - Создает резервную копию текущего файла конфигурации.  - Использует `sed` для обновления значений ключей `key\_name` и `model.name` в JSON-файле:  - Для ключа `key\_name` используется первая команда `sed`, которая заменяет строку с ключом на новую строку с переданным значением.  - Для ключа `model.name` используется вторая команда `sed`, которая выполняет аналогичное действие.  - Проверяет успешность операции и выводит соответствующее сообщение.  5. \*\*Цикл обновления конфигурации\*\*: Скрипт запускает функцию `update\_config` для каждого сервера из списка.  ### Важные замечания:  - \*\*Меньшая гибкость\*\*: Использование `sed` требует строгого соответствия формату JSON-файла. Например, если JSON будет отформатирован по-другому, скрипт может не сработать как ожидалось.  - \*\*Резервное копирование\*\*: Создается резервная копия файла перед внесением изменений с суффиксом `.bak`, чтобы можно было восстановить файл в случае ошибки.  - \*\*Многократный запуск\*\*: Скрипт можно запускать повторно без необходимости изменения его структуры.  Теперь вы можете использовать этот скрипт аналогичным образом:  ```bash  ./update\_config.sh "new\_rsa\_private\_key" "New Model Name"  ``` |