УРОК 15. ПРОДОЛЖЕНИЕ РАБОТЫ СО СКРИПТАМИ

ПРАКТИЧЕСКАЯ РАБОТА СО СКРИПТАМИ

Скрипт — это небольшая программа, которая содержит последовательность действий, созданных для автоматического выполнения задачи.

Задача 1

Создать скрипт, который будет мониторить занятое дисковое пространство.
Если место заканчивается, а именно в корневом разделе файловой системы занято больше 70% , то то найдем самые большие директории и файлы.

Решение 1

```
#!/bin/bash

# Пороговое значение для занятого дискового пространства (в процентах)
threshold=70

# Получаем процент занятого дискового пространства корневого раздела
disk_usage=$(df / | awk '{print $5}' | sed 's/%//')

if [ "$disk_usage" -gt "$threshold" ]; then
    echo "Занято более $threshold% дискового пространства. Поиск самых больших директорий и файлов..."

    # Поиск самых больших директорий и файлов
    echo "Самые большие директории и файлы:"
    du -ah / | sort -rh | head -n 10
else
    echo "Занятое дисковое пространство на корневом разделе меньше или равно $threshold%."
fi
```

Задача 2

Создать скрипт, который будет готовить нам отчет о системе: версия операционной системы, дата и время, время работы, загруженность системы, занятое дисковое пространство, топ процессы по использованию памяти, количество процессов, количество пользователей.

Решение 2

```
#!/bin/bash

# Версия операционной системы
os_version=$(cat /etc/os-release | grep "PRETTY_NAME" | cut -d '"' -f 2)

# Дата и время
current_date=$(date "+%Y-%m-%d")
current_time=$(date "+%H:%M:%S")

# Время работы системы
uptime_info=$(uptime -p)

# Загруженность системы
system_load=$(uptime | awk -F'[a-z]:' '{ print $2 }')

# Занятое дисковое пространство
disk_usage=$(df / | awk '{print $5}' | sed 's/%//')

# Топ процессы по использованию памяти
top_processes=$(ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head -n 6)

# Количество процессов
process_count=$(ps -ef | wc -l)

# Количество пользователей
user_count=$(who | wc -l)

# Выводим отчет
echo "Отчет о системе"
echo "Версия операционной системы: $os_version"
```


```
echo "Дата: $current_date"
echo "Время: $current_time"
echo "Время работы системы: $uptime_info"
echo "Загруженность системы: $system_load"
echo "Занятое дисковое пространство: $disk_usage"
echo "Топ процессы по использованию памяти:"
echo "$top_processes"
echo "Количество процессов: $process_count"
echo "Количество пользователей: $user_count"
```

Задача 3

Создать задание, которое будет раз в неделю (каждую неделю в воскресенье в 2 часа ночи) будет бекапить /opt и /home/ec2-user директории, создавая архивы. Настроить скрипт так, чтобы он хранил не более 3 копий архивов с бекапами.

Решение 3

Создайте скрипт с именем, например, backup_script.sh:

```
#!/bin/bash

# Директории для резервного копирования
backup_dirs=("/opt" "/home/ec2-user")

# Директория для хранения архивов
backup_location="/backup"

# Максимальное количество хранимых архивов
max_backup_count=3

# Создаем каталог для хранения бекапов, если он не существует
mkdir -p "$backup_location"

# Проходимся по каждой директории и создаем архив
for dir in "${backup_dirs[@]}"; do
                        backup_file="$backup_location/$(basename       "$dir")_backup_$(date
+%Y-%m-%d).tar.gz"
   tar -czf "$backup_file" "$dir"
done
```

Удаляем старые архивы, если наше задание выполняется раз в неделю, то -mtime будет 21

find "$backup_location" -maxdepth 1 -type f -name "*.tar.gz" -mtime +21 -delete

fi

Настройка cron:
crontab -e

Добавьте следующую строку, чтобы задание выполнялось каждую неделю в воскресенье в 2 часа ночи и результаты работы перенаправлялись в log файл:

0 2 * * 0 /полный/путь/к/вашему/скрипту/backup_script.sh >> /полный/путь/к/вашему/файлу_журнала.log 2>&1

Задача 4

Создать скрипт , который будет проверять на ошибки в логах в директории /var/log и готовить отчет об их количестве и самых частых ошибках.
Добавить шаблон для поиска ошибок как переменную: искать error Error ERROR WARNING и прочие похожие сочетания.

Решение 4

#!/bin/bash

# Директория с логами
log_directory="/var/log"

# Файл для сохранения отчета
report_file="/tmp/error_report.txt"

# Шаблон для поиска ошибок
error_patterns="(error|Error|ERROR|warning|Warning|WARNING)"

Поиск ошибок в логах и подсчет их количества
grep -E -r "$error_patterns" "$log_directory" | awk -F ':' '{print $2}' | sort | uniq -c | sort -nr > "$report_file"


```
# Вывод отчета в консоль
echo "Отчет о ошибках в логах:"
cat "$report_file"

# Подготовка списка наиболее частых ошибок (первые 5)
echo -e "\nНаиболее частые ошибки (первые 5):\n"
head -n 5 "$report_file"
```