**DevOps Real-time interview Questions & Answers**

**Complete Shell Scripting**

***with***

**Detailed Use Cases**

**Overview:**

**1. Basics of Shell Scripting**

**1.1 Variables**

**1.2 Command Substitution**

**1.3 Input and Output**

**1.4 Passing Arguments**

**2. Intermediate Shell Scripting**

**2.1 Conditional Statements**

**2.2 Loops**

**2.3 Functions**

**3. Advanced Shell Scripting**

**3.1 File Handling**

**3.2 Error Handling**

**3.3 Scheduling Scripts with Cron**

**3.4 Debugging**

**4. Real-World Use Cases**

**4.1 Automating AWS Tasks**

**4.2 Log File Parsing**

**4.3 Deployment Automation**

**4.4 Disk Usage Monitoring**

**4.5 Network Monitoring**

**Explanations:**

**1. Basics of Shell Scripting**

**1.1 Variables**

**Why?**

Variables store values, making scripts dynamic and reusable.

**Use Case: Dynamically Storing User Input**

#!/bin/bash

username="Alice"

server="prod-server"

echo "Deploying application for user $username on $server"

**1.2 Command Substitution**

**Why?**

Store the output of commands for further processing.

**Use Case: Fetching System Uptime**

#!/bin/bash

uptime=$(uptime -p)

echo "The system has been running for: $uptime"

**1.3 Input and Output**

**Why?**

To interact with the user or external systems.

**Use Case: Save a User's Input to a File**

#!/bin/bash

read -p "Enter your name: " name

echo "$name" > username.txt

**1.4 Passing Arguments**

**Why?**

Makes scripts reusable for different inputs.

**Use Case: Dynamic Greeting Script**

#!/bin/bash

echo "Hello, $1! Welcome to $2!"

Run the script:

./script.sh Alice Linux

**2. Intermediate Shell Scripting**

**2.1 Conditional Statements**

**Why?**

To execute logic based on specific conditions.

**Use Case: Checking Disk Space**

#!/bin/bash

disk\_space=$(df / | grep / | awk '{ print $5 }' | sed 's/%//')

if [ $disk\_space -gt 80 ]; then

echo "Warning: Disk space usage is at $disk\_space%."

else

echo "Disk space usage is under control."

fi

**2.2 Loops**

**Why?**

Handle repetitive tasks efficiently.

**Use Case: Ping Multiple Servers**

#!/bin/bash

servers=("192.168.1.1" "192.168.1.2" "192.168.1.3")

for server in "${servers[@]}"; do

echo "Pinging $server..."

ping -c 1 $server > /dev/null && echo "$server is reachable" || echo "$server is down"

done

**2.3 Functions**

**Why?**

Modularize scripts for better readability and reuse.

**Use Case: Reusable Deployment Function**

#!/bin/bash

deploy\_app() {

echo "Deploying $1 on $2"

# Add deployment commands here

}

deploy\_app "frontend" "server1"

deploy\_app "backend" "server2"

**3. Advanced Shell Scripting**

**3.1 File Handling**

**Why?**

File handling automates tasks like reading logs or generating reports.

**Use Case: Backup and Compress Files**

#!/bin/bash

backup\_dir="/backup"

mkdir -p $backup\_dir

tar -czvf $backup\_dir/backup\_$(date +%F).tar.gz /var/www/html

echo "Backup created at $backup\_dir"

**3.2 Error Handling**

**Why?**

Prevents failures during execution by handling errors gracefully.

**Use Case: Verify and Install Missing Packages**

#!/bin/bash

set -e

if ! dpkg -l | grep "nginx" > /dev/null; then

echo "Installing Nginx..."

sudo apt update && sudo apt install -y nginx

else

echo "Nginx is already installed."

fi

**3.3 Scheduling Scripts with Cron**

**Why?**

Automates recurring tasks like backups, monitoring, etc.

**Use Case: Daily Cleanup**

#!/bin/bash

find /tmp -type f -mtime +7 -exec rm -f {} \;

echo "Old temporary files cleaned up."

Add to crontab:

0 3 \* \* \* /path/to/script.sh

**3.4 Debugging**

**Why?**

Helps identify and resolve issues in complex scripts.

**Use Case: Debugging a Deployment Script**

#!/bin/bash

set -x

echo "Deploying application..."

# Add deployment commands here

set +x

echo "Deployment complete."

**4. Real-World Use Cases**

**4.1 Automating AWS Tasks**

**Why?**

Reduces manual effort in managing AWS resources.

**Use Case: Start/Stop EC2 Instances**

#!/bin/bash

instances=("i-1234567890abcdef" "i-abcdef1234567890")

for instance in "${instances[@]}"; do

aws ec2 start-instances --instance-ids $instance

echo "Started instance $instance"

done

**4.2 Log File Parsing**

**Why?**

Automates extracting insights from log files.

**Use Case: Find Errors in Logs**

#!/bin/bash

grep "ERROR" /var/log/app.log > error\_report\_$(date +%F).log

echo "Error report generated."

**4.3 Deployment Automation**

**Why?**

Streamlines CI/CD processes by automating deployments.

**Use Case: Pull Code and Restart Services**

#!/bin/bash

git pull origin main

sudo systemctl restart my-service

echo "Deployment complete."

**4.4 Disk Usage Monitoring**

**Why?**

Prevents downtime by monitoring storage utilization.

**Use Case: Alert if Disk Usage Exceeds 90%**

#!/bin/bash

usage=$(df / | grep / | awk '{print $5}' | sed 's/%//')

if [ $usage -gt 90 ]; then

echo "Disk usage critical: $usage%"

# Add alerting mechanism here (e.g., email or Slack)

fi

**4.5 Network Monitoring**

**Why?**

Ensures availability of critical servers.

**Use Case: Alert if Server is Unreachable**

#!/bin/bash

servers=("google.com" "github.com")

for server in "${servers[@]}"; do

ping -c 2 $server > /dev/null

if [ $? -ne 0 ]; then

echo "Server $server is unreachable"

else

echo "Server $server is reachable"

fi

done

**5. Best Practices**

1. **Add Comments:** Improve readability for others.
2. **Error Handling:** Use set -e to stop on errors.
3. **Use ShellCheck:** Identify common errors.
4. **Test Scripts:** Always test in staging before production.
5. **Keep it Simple:** Avoid overcomplicating scripts.

This guide provides a complete understanding of shell scripting, practical use cases, and detailed examples for real-world automation tasks.