Full stack Lab Material

By
K.R.Uthayan

Shell Scripts

```
#!/bin/bash
echo "System Information:"
uname -a
echo "Disk Usage:"
df -h
echo "Memory Usage:"
free -m
```

- Outputs basic system information, including kernel version, disk usage, and memory usage.

Shell Scripts

```
#!/bin/bash
echo "Compiling Java files..."
javac -d bin src/*.java
if [ $? -eq 0 ]; then
  echo "Compilation successful."
else
  echo "Compilation failed."
  exit 1
fi
```

- Compiles Java files in a src directory and outputs the results to a bin directory.

Shell Scripts

- javac -d bin src/*.java: This command attempts to compile all Java files in the src directory, putting the compiled .class files in the bin directory.
- if [$? -eq 0]; then: Right after the javac command, this line checks if the compilation was successful by evaluating the exit status ($?).
- echo "Compilation successful.": If the compilation was successful, this message is printed.
- else: If the exit status is not 0 (meaning the compilation failed), the script executes the code under else.
- echo "Compilation failed.": This message is printed if the compilation fails.
- exit 1: The script exits with a status of 1, which generally indicates an error.

Shell Scripts

1. $?

- What It Is: This is a special variable in Unix-like systems (including Linux and macOS) that holds the exit status of the last command executed.
- What It Represents:
- If a command runs successfully, $? will have a value of 0.
- If the command fails, $? will have a non-zero value (usually 1, but it can be other values depending on the command and the nature of the error).

2. if [$? -eq 0]; then

- What It Is: This is a conditional statement in shell scripting that checks the exit status of the previous command.
- Explanation:
- if [$? -eq 0]: This checks if the value of $? is equal to 0, which means the previous command was successful.
- then: If the condition is true (i.e., if the previous command was successful), the commands following then will be executed.

Shell Scripts

```
#!/bin/bash
echo "Running tests..."
for test in tests/*.sh; do
  bash "$test"
  if [ $? -ne 0 ]; then
    echo "Test $test failed."
    exit 1
  fi
done
echo "All tests passed."
```

- Executes all shell scripts in the tests directory, treating them as test cases. If any test fails, the build fails.

Shell Scripts

```
#!/bin/bash
echo "Deploying application..."
scp -r ./app user@remote_server:/var/www/html
if [ $? -eq 0 ]; then
  echo "Deployment successful."
else
  echo "Deployment failed."
  exit 1
fi
```

- Deploys an application by copying files to a remote server via scp.

Shell Scripts

- scp: Secure Copy Protocol. It allows you to securely transfer files between hosts over a network.
- -r: Recursively copy entire directories. This is necessary when you want to copy a directory and all its contents.
- ./app: The source directory on your local machine that you want to copy. In this case, app is the directory containing your files.
- user@remote_server: The SSH credentials for the remote server where you want to copy the files.
- user: The username on the remote server.
- remote_server: The hostname or IP address of the remote server.
- /var/www/html: The destination directory on the remote server where the files will be copied. This directory is often used to serve web content, particularly on web servers like Apache or Nginx.

Shell Scripts

Copying the Directory:

- This command will copy the entire app directory from your local machine to the /var/www/html directory on the remote server.
- The -r flag ensures that all files and subdirectories inside app are also copied.
- Example Scenario:
- If you are deploying a web application, this command will place your application files into the web server's document root (/var/www/html) on the remote server.
- Permissions: Ensure that the user has the necessary permissions to write to the /var/www/html directory on the remote server. If not, you may need to use sudo or adjust permissions.
- SSH Access: Make sure you have SSH access to the remote server, and that you can log in with the credentials provided (i.e., user@remote_server).
- Port Specification (Optional): If the SSH server on the remote host listens on a non-default port (not 22), you can specify it with the -P flag
- scp -r ./app user@192.168.1.100:/var/www/html
- This would copy the app directory from your local machine to the /var/www/html directory on a remote server with the IP address 192.168.1.100, using the username user.

Shell Scripts

```
#!/bin/bash
echo "Checking environment variables..."
echo "Jenkins Home: $JENKINS_HOME"
echo "Workspace: $WORKSPACE“
```

- Displays the values of some key environment variables in the Jenkins job.

```
#!/bin/bash
echo "Starting backup..."
tar -czf /backup/$(date +%F).tar.gz /data
if [ $? -eq 0 ]; then
  echo "Backup successful."
else
  echo "Backup failed."
  exit 1
fi
```

- Creates a compressed backup of the /data directory and stores it in /backup with the current date as the filename.

Shell Scripts

```
#!/bin/bash
echo "Cleaning up old files..."
find /var/log -type f -mtime +30 -exec rm -f {} \;
if [ $? -eq 0 ]; then
  echo "Cleanup successful."
else
  echo "Cleanup failed."
  exit 1
fi
```

- Deletes files in /var/log that are older than 30 days.

Shell Scripts

```
#!/bin/bash
echo "Checking for updates..."
sudo apt-get update && sudo apt-get upgrade -y
if [ $? -eq 0 ]; then
  echo "System updated successfully."
else
  echo "Update failed."
  exit 1
fi
```

- Updates the system packages.

1. Understanding Build, Test, and Deploy

- Build: In Java, javac is the command that compiles the .java files into bytecode (.class files). This is the build step.
- Test: Running tests can involve executing the Java program with specific test cases, or using a testing framework like JUnit.
- Deploy: Deployment typically involves copying the compiled .class files to a directory where they can be executed, either on the same host or a remote host.

2. Setting Up the Jenkins Freestyle Project

Freestyle Project: In Jenkins, a Freestyle project allows you to define and execute a series of steps, including building, testing, and deploying your application.

3. **Sample Java Program**

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

4. **Shell Script for Jenkins Freestyle Project**

Step 1: Build

```
#!/bin/bash
echo "Building the Java program..."
javac HelloWorld.java
if [ $? -eq 0 ]; then
    echo "Build successful."
else
    echo "Build failed."
    exit 1
fi
```

4. **Shell Script for Jenkins Freestyle Project**

Step 2: Test

```
#!/bin/bash
echo "Running the Java program..."
java HelloWorld > output.txt
if grep -q "Hello, World!" output.txt; then
    echo "Test passed."
else
    echo "Test failed."
    exit 1
fi
```

4. **Shell Script for Jenkins Freestyle Project**

Step 3: Deploy [locally]

```
#!/bin/bash
echo "Deploying the application locally..."
cp HelloWorld.class /path/to/deploy/directory
if [ $? -eq 0 ]; then
    echo "Deployment successful."
else
    echo "Deployment failed."
    exit 1
fi
```

- Local Deployment: The .class file is copied to a specific directory on the local machine.
- Remote Deployment: The .class file is copied to a remote machine using scp.

4. **Shell Script for Jenkins Freestyle Project**

```
#!/bin/bash
echo "Deploying the application to a remote host..."
scp HelloWorld.class user@remote_host:/path/to/deploy/directory
if [ $? -eq 0 ]; then
    echo "Deployment to remote host successful."
else
    echo "Deployment to remote host failed."
    exit 1
fi
```

4. **Shell Script for Jenkins Freestyle Project**

Step 4: Open Chrome Browser (Optional)

If you want to open the Chrome browser as part of your test (for example, to simulate end-user interaction or to check a deployment):

```
#!/bin/bash
echo "Opening Chrome browser..."
google-chrome http://localhost:8080
if [ $? -eq 0 ]; then
    echo "Chrome browser opened successfully."
else
    echo "Failed to open Chrome browser."
    exit 1
fi
```