

# Day 15: Introduction to Shell Scripting

DevOps Beginner Lesson

Presented by Mohd Shahid

# What is Shell Scripting?

A Shell Script is a text file containing a sequence of commands that are executed by the shell.

Why is Shell Scripting important in DevOps?

- Automates repetitive tasks
- Enhances infrastructure provisioning
- Can integrate with DevOps tools
- Streamlines CI/CD pipelines

# Components of a Shell Script

- **Shebang (`#!/bin/bash`)**

This tells the system which interpreter to use when executing the script. In this case, it's Bash (`/bin/bash`). CommandsThe script contains

- **commands**

you'd run in the terminal. These could be system commands (like `ls`, `cat`, `echo`, etc.) or custom ones.

# Real-World Use Cases in DevOps

- Automating server setup
- CI/CD pipeline scripting
- File backups and restores
- Log management and monitoring

# Writing Your First Shell Script

A shell script is simply a series of commands written in a text file, which the shell executes.

## 1. Create a Script:

```
nano hello.sh
```

## 2. Add this code:

```
#!/bin/bash
```

```
echo 'Hello, DevOps Challenge!'
```

## 3. Make the script executable:

```
chmod +x hello.sh
```

## 4. Run the script:

```
./hello.sh
```

# Variables in Shell Scripting

Variables store data.

Define a variable:

```
name='DevOps'
```

Access the variable:

```
echo $name
```

# Taking User Input

1. Use the 'read' command:

```
echo 'Enter your name:'
```

```
read name
```

```
echo 'Hello, $name!'
```

# Conditionals (If-Else Statements)

Example of an if-else statement:

```
if [ $num -gt 10 ]; then  
    echo 'Greater than 10'  
else  
    echo 'Less than or equal to 10'  
fi
```



# Loops in Shell Scripting

For loop example:

```
for i in 1 2 3 4 5; do  
    echo $i  
done
```

While loop example:

```
count=1  
while [ $count -le 5 ]; do  
    echo $count  
    ((count++))  
done
```

# Functions in Shell Scripting

- Define a function:

- `greet() {`

- `echo 'Hello, $1!'`

- `}`

- Call the function:

- `greet 'DevOps'`

# Arrays in Shell Scripting

Define an array:

```
fruits=('Apple' 'Banana' 'Orange')
```

Access array elements:

```
echo ${fruits[0]}
```

# Debugging Shell Scripts

Use `set -x` to debug:

```
set -x
```

```
echo 'This is a test'
```

```
set +x
```

# Best Practices

1. Use shebang `#!/bin/bash`
2. Name variables and functions clearly
3. Comment your scripts
4. Test scripts regularly
5. Use `set -e` to exit on error

# Summary

- Write and execute shell scripts
- Use variables, loops, and conditionals
- Automate DevOps tasks