Bash Scripting for DevOps

Author: Zayan Ahmed | **Estimated Reading time:** 4 mins

Have you ever wondered how computers can do tasks automatically without someone sitting there to press buttons? Bash scripting is one of the tools that makes this possible. It's like giving your computer a list of instructions and asking it to follow them step by step. For people in DevOps, this is super helpful because they often need to manage servers, deploy code, and monitor systems, all without doing things manually every time.

Bash scripting is like writing a recipe for your computer. Instead of cooking food, though, it's running commands to get work done. Let's explore how this works with some simple examples that solve real problems in DevOps.

Scenario 1: Checking if a Server Is Running

Imagine you're managing a group of servers, and you need to check if one of them is running. Doing this manually for 10 or 20 servers would take forever! Instead, you can write a Bash script that checks the server's status for you.

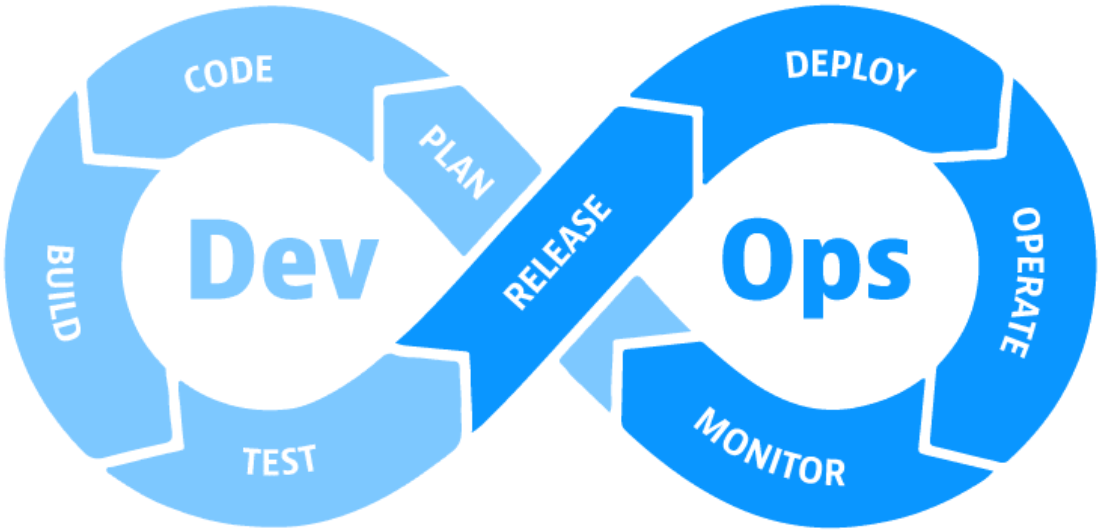
Here's a script for that:

```
#!/bin/bash

# Define the server's IP or hostname
SERVER="192.168.1.100"

# Ping the server to check if it's up
if ping -c 1 $SERVER &> /dev/null; then
```

```
 echo "Server $SERVER is running!"
else
 echo "Server $SERVER is not reachable."
fi
```

Explanation:

- The `SERVER` variable holds the server's IP address.
- The `ping` command sends a small message to the server to see if it responds.
- The `if` statement checks if the `ping` command works. If it does, the script says the server is running. If not, it says the server isn't reachable.
- You can run this script on your terminal to quickly check your server's status.



Scenario 2: Automating Backups

Backing up important files is a common DevOps task. Instead of copying files manually every day, you can use a Bash script to do it for you.

Here's a script to back up files to a folder named "backup":

```
#!/bin/bash

# Define the folder to back up and where to store the backup
SOURCE="/home/user/documents"
DESTINATION="/home/user/backup"

# Create the backup folder if it doesn't exist
```

```
mkdir -p $DESTINATION

# Copy the files
cp -r $SOURCE/* $DESTINATION

# Print a success message
echo "Backup completed successfully!"
```

Explanation:

- The `SOURCE` variable is the folder you want to back up.
- The `DESTINATION` variable is where the backup will be stored.
- The `mkdir -p` command makes sure the backup folder exists (it creates it if it doesn't).
- The `cp -r` command copies all files and subfolders from the source to the destination.
- Finally, the script tells you that the backup is done.

Scenario 3: Restarting a Service Automatically

Sometimes, services on a server stop working. With Bash scripting, you can check if a service is running and restart it if needed. Here's an example for a web server:

```
#!/bin/bash

# Define the service name
SERVICE="apache2"

# Check if the service is running
if systemctl is-active --quiet $SERVICE; then
 echo "$SERVICE is running."
else
 echo "$SERVICE is not running. Restarting..."
 systemctl restart $SERVICE
 echo "$SERVICE has been restarted."
fi
```

Explanation:

- The `SERVICE` variable is the name of the service you want to monitor (e.g., `apache2` for the Apache web server).
- The `systemctl is-active` command checks if the service is running.

- If the service isn't running, the script restarts it using `systemctl restart`.
- This script makes sure your important services stay online without you having to monitor them constantly.

Why DevOps Engineers Love Bash Scripting

Bash scripts are powerful tools for DevOps because they save time and reduce human error. Instead of doing the same tasks over and over again, you can write a script once and use it whenever you need. Scripts can also run automatically, like an alarm clock, so you don't have to remember to start them.

With Bash scripting, you can:

- Manage servers easily.
- Automate repetitive tasks like backups.
- Monitor and fix issues without manual effort.

Conclusion

Bash scripting might seem tricky at first, but it's just about writing instructions that your computer can follow. For DevOps engineers, it's like having a superpower that makes everyday tasks faster and simpler. Start with small scripts, like the ones here, and soon you'll be automating like a pro. Give it a try and see how much time you can save!

Want more? 🤔
Follow me on **LinkedIn** 😊