PW SKILLS | DevOps and Cloud Computing

# Introduction to Bash Scripting and Automation in the Cloud

## Objective

- Understand the basics of Bash scripting and its importance in automation.
- Learn to launch and connect to an EC2 instance using VSCode.
- Explore shell scripting fundamentals, including syntax and common commands.
- Identify tasks suitable for automation in cloud and development workflows.

PW SKILLS

## Explaining what Bash scripting is and why it's widely used

## Bash scripting

Bash scripting is a scripting language used to automate tasks in UNIX/Linux environments by writing sequences of commands in a text file (.sh extension) that the Bash shell executes.

- It starts with a shebang line (#!/bin/bash) to specify the interpreter and allows variables, loops, conditionals, and direct system command integration.

Key reasons for its widespread use:

1. Task automation
2. Deep OS integration
3. Portability
4. Efficiency
5. Ease of learning

## Pop Quiz

Q. What does the shebang line (#!) in a Bash script specify?

A

The interpreter to be used for executing the script

B

The version of Bash being used

## Pop Quiz

Q. What does the shebang line (#!) in a Bash script specify?

**A**

The interpreter to be used for executing the script

**B**

The version of Bash being used

PW SKILLS

## Common use cases: task automation, system management, and deployment scripts.

## Common Use Cases of Automation

Automation has become an essential tool across various domains, enhancing efficiency and accuracy in numerous processes. The three primary use cases of automation include task automation, system management, and deployment scripts.

### Task Automation

Task automation involves using software to perform repetitive tasks without human intervention. Common use cases include:

- Data Entry
- Email Notifications
- Report Generation Appointment Scheduling

## Common Use Cases of Automation

### System Management

System management automation focuses on maintaining and overseeing IT systems with minimal manual effort. Key use cases include:

- Configuration Management
- Monitoring and Alerts
- Resource Allocation

### Deployment Scripts

Deployment automation involves using scripts to manage the release of software updates seamlessly. This includes:

- Continuous Integration/Continuous Deployment (CI/CD)
- Automated Testing
- Version Control Integration

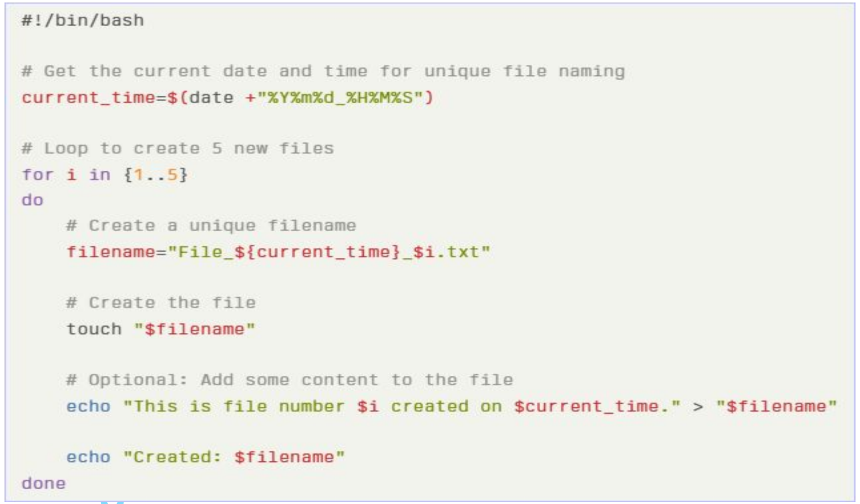
PW SKILLS

## Demonstrating writing a simple Bashscript to automate repetitive tasks (e.g., file creation).

## Let’s do it

### Simple Bash Script to Automate File Creation

Below is a simple Bash script that automates the creation of multiple text files. Each time you run the script, it will create five new files with unique names based on the current date and time.



```bash
#!/bin/bash

# Get the current date and time for unique file naming
current_time=$(date +"%Y%m%d_%H%M%S")

# Loop to create 5 new files
for i in {1..5}
do
    # Create a unique filename
    filename="File_${current_time}_$i.txt"

    # Create the file
    touch "$filename"

    # Optional: Add some content to the file
    echo "This is file number $i created on $current_time." > "$filename"

    echo "Created: $filename"
done
```

PW SKILLS

## Introducing AWS EC2 and its relevance for cloud computing.

## Introducing AWS EC2

Amazon Elastic Compute Cloud (EC2) is a core component of Amazon Web Services (AWS) that provides scalable computing capacity in the cloud. It enables users to launch and manage virtual servers, known as instances, on-demand, allowing for significant flexibility and cost savings in computing resources.

### Relevance in Cloud Computing

AWS EC2 plays a pivotal role in cloud computing by:

- Reducing Costs
- Enabling Rapid Deployment
- Supporting Diverse Workloads

PW SKILLS

## Demonstrating the steps to launch an EC2 instance

## Let’s do it

To launch an EC2 instance on AWS, follow these steps:

**Step 1:** Choose an Amazon Machine Image (AMI)

- Log in to the AWS Management Console and navigate to the EC2 Dashboard.
- Click on the Launch Instance button.
- Under Application and OS Images (Amazon Machine Image), select the Quick Start tab.
- Choose an AMI that suits your needs. For beginners, selecting Amazon Linux or any Free Tier eligible AMI is recommended.

## Let’s do it

**Step 2:** Configure Instance Details

- Instance Type
- Configure Instance Details

**Step 3:** Configure Security Groups

- In the Configure Security Group section, create a new security group or select an existing one.
- Define inbound rules to control traffic (e.g., allow SSH access on port 22).
- Ensure that you set appropriate rules for your use case.

## Let’s do it

**Step 4:** Key Pairs

- Under Key pair (login), choose an existing key pair or create a new one.
- If creating a new key pair, download the private key file (.pem) and store it securely, as it will be needed to access your instance.

Final Steps

- Review all settings to ensure everything is configured correctly.Click on the Launch button to start your EC2 instance.

## How to connect to the EC2 instance using VSCode and the Remote - SSH extension.

## Let’s do it

**Step 1: Install the Remote - SSH Extension**

- Open VSCode.
- Click on the Extensions icon in the sidebar or press Ctrl + Shift + X.
- Search for Remote - SSH by Microsoft and click Install.

**Step 2: Configure SSH Access**

- Open your terminal and set permissions for your SSH key:

```
chmod 400 /path/to/your-key.pem
```

- Test your SSH connection to ensure it works:

```
ssh -i /path/to/your-key.pem ec2-user@your-instance-public-ip
```

## Let’s do it

**Step 3: Add the EC2 Instance to SSH Configuration**

- In VSCode, open the Command Palette by pressing Ctrl + Shift + P.
- Type and select Remote-SSH: Add New SSH Host....
- Enter the SSH connection string:

```
ssh -i /path/to/your-key.pem ec2-user@your-instance-public-ip
```

- Choose to save this configuration in the default SSH config file (usually located at ~/.ssh/config).

**Step 4: Connect to Your EC2 Instance**

- Again, open the Command Palette and select Remote-SSH: Connect to Host....
- Choose your EC2 instance from the list of configured hosts.
- If prompted, enter your SSH key passphrase.

## Take A 5-Minute **Break!**

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

## The structure of a shell script (shebang line, commands, and comments).

## The structure of a shell script

A shell script is a text file containing a series of commands that the shell interprets and executes. The structure of a shell script typically includes the following components:

### Shebang Line

The shebang line is the first line of a shell script and begins with #!. It specifies the interpreter that should be used to execute the script. For example, a common shebang for Bash scripts is:

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

## The structure of a shell script

### Commands

The main body of the script consists of a series of commands that are executed sequentially. These can be built-in shell commands, external programs, or custom functions. For example:

```
echo "Hello, World!"
ls -l
```

## The structure of a shell script

### Comments

Comments are used to provide explanations or documentation within the script. They are ignored by the interpreter and are denoted by the # symbol. For example:

```
# This script displays a greeting message
echo "Hello, World!"  # Print greeting
```

- Comments help make scripts more understandable for others (or for yourself when revisiting your code later).

## Pop Quiz

Q. Which symbol is used to denote a comment in a shell script?

A

//

B

#

## Pop Quiz

Q. Which symbol is used to denote a comment in a shell script?

A

//

B

#

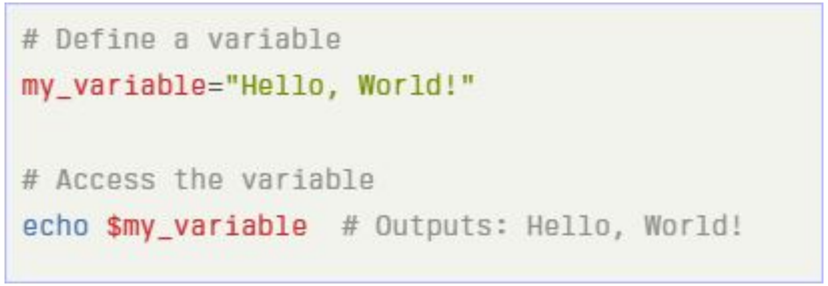
PW SKILLS

## Basic syntax: variables, loops (for, while), and conditionals (if, else).

## Basic syntax

### 1. Variables

In Bash, you can define variables without specifying a type. The syntax is straightforward:



```bash
# Define a variable
my_variable="Hello, World!"

# Access the variable
echo $my_variable  # Outputs: Hello, World!
```

Note: There should be no spaces around the ‘=‘ sign when assigning a value to a variable.

## Basic syntax

2. Loops

**For Loop**

A ‘for’ loop is used to iterate over a list of items or a range of numbers.

```
# For loop example
for i in {1..5}
do
    echo "Iteration $i"
done
```

This will output:

```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

## Basic syntax

### While Loop

A ‘while’ loop continues executing as long as a specified condition is true.

```
# While loop example
count=1
while [ $count -le 5 ]
do
    echo "Count is $count"
    ((count++))  # Increment count by 1
done
```

This will output

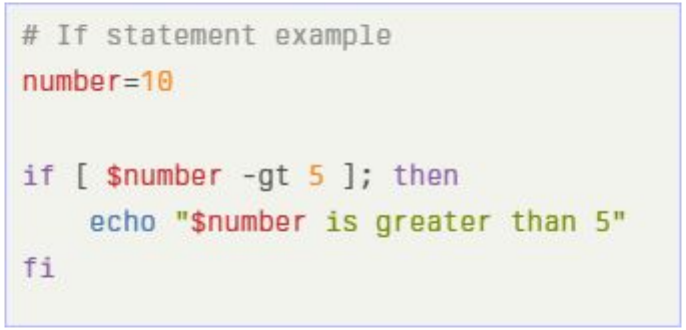
```
Count is 1
Count is 2
Count is 3
Count is 4
Count is 5
```

## Basic syntax

3. Conditionals

**If Statement**

An ‘if’ statement allows you to execute commands based on whether a condition is true.



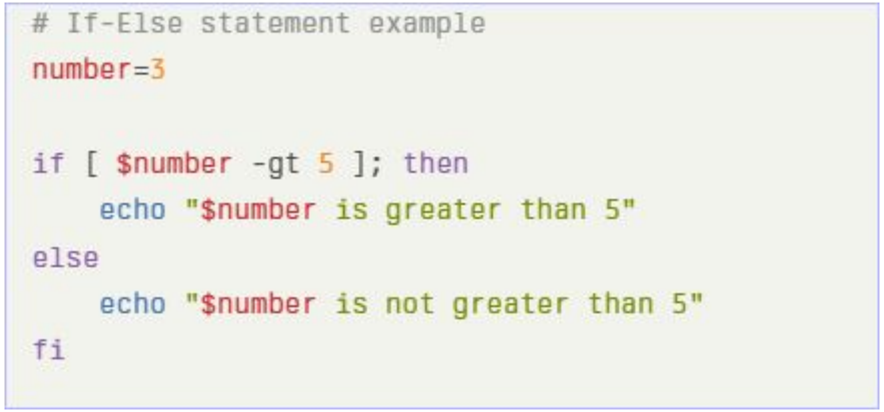
```
# If statement example
number=10

if [ $number -gt 5 ]; then
    echo "$number is greater than 5"
fi
```

## Basic syntax

**If-Else Statement**

You can extend the ‘if’ statement with an ‘else’ clause to handle alternative conditions.

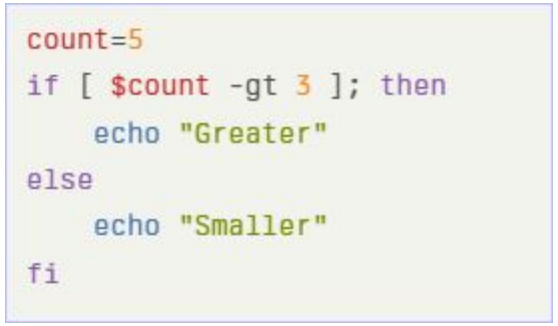


```
# If-Else statement example
number=3

if [ $number -gt 5 ]; then
    echo "$number is greater than 5"
else
    echo "$number is not greater than 5"
fi
```

## Pop Quiz

Q. What will be the output of this code snippet?



```
count=5
if [ $count -gt 3 ]; then
    echo "Greater"
else
    echo "Smaller"
fi
```

A

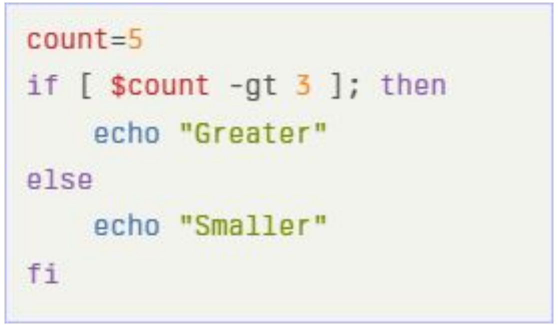
Greater

B

Smaller

## Pop Quiz

Q. What will be the output of this code snippet?



```
count=5
if [ $count -gt 3 ]; then
    echo "Greater"
else
    echo "Smaller"
fi
```

A

Greater

B

Smaller

PW SKILLS

## Demonstrating common commands

## Lets do it

**File Operations**

1. Creating Files: ‘touch’

   The touch command is used to create an empty file or update the timestamp of an existing file.

```
touch filename.txt
```

1. Creating Directories: ‘mkdir’

```
mkdir new_directory
```

To create a parent directory along with a child directory, use:

## Lets do it

3. Removing Files: 'rm'

```
rm filename.txt
```

To remove a directory and its contents recursively, use:

```
rm -r directory_name
```

### Process Management

1. Viewing Processes: 'ps'

```
ps aux   # Shows detailed information about all processes
```

## Lets do it

2. Killing Processes: ‘kill’

```
kill PID  # Replace PID with the actual process ID
```

- To forcefully kill a process, you can use:

```
kill -9 PID
```

**Networking Commands**

- Checking Connectivity: ‘ping’

```
ping example.com  # Replace with the desired hostname or IP address
```

- Fetching Data: ‘curl’

```
curl http://example.com  # Fetches the content of the specified URL
```

## Pop Quiz

Q. How do you remove a file named example.txt?

**A**

```
rm example.txt
```

**B**

```
rmdir example.txt
```

## Pop Quiz

Q. How do you remove a file named example.txt?

A

rm example.txt

B

rmdir example.txt

## Pop Quiz

Q. Which command is used to check network connectivity to a host?

A

curl

B

ping

## Pop Quiz

Q. Which command is used to check network connectivity to a host?

**A**

curl

**B**

ping

PW SKILLS

## Discussing automation scenarios

## Automation Scenarios

1. System Maintenance Tasks (Backups, Log Rotation):

- Automation Tools: Use cron or anacron to schedule regular backups and log rotations.
- For example, a shell script can be set to run daily at midnight to back up important directories and rotate logs, ensuring system performance and data integrity without manual intervention. This enhances reliability and reduces the risk of human error.

## Automation Scenarios

2. Cloud Resource Management (Starting/Stopping EC2 Instances):

- Automation Tools: Utilize AWS CLI or SDKs to automate the management of EC2 instances. Scripts can be scheduled to start or stop instances based on usage patterns or specific times, optimizing costs and resource allocation.
- For instance, instances can be automatically stopped during off-peak hours and started again during business hours.

## Automation Scenarios

3. DevOps Workflows (CI/CD Pipelines):

- Automation Tools: Implement CI/CD tools like Jenkins, GitLab CI, or CircleCI to automate the software development lifecycle. These tools can trigger builds, run tests, and deploy applications automatically upon code changes in the repository.
- This streamlines the development process, reduces time-to-market, and ensures consistent deployment practices.

## Time for case study!

## Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this session
- Review your lecture notes
- Note down questions and queries regarding this session and consult the teaching assistants

# Thanks!

PW SKILLS