Linux Shell Script Collection – Documentation

Repository: variable_shell_script

Overview:

This repository contains various bash scripts designed to demonstrate shell scripting concepts and automate system administration tasks such as environment setup, OS checks, Docker and AWS EC2 deployment, and Django application deployment.

How to Use This Repository:

To Create a File:

```
vim filename.sh
```

To Change File Permission to Executable:

```
chmod +x filename.sh
```

To Run the File:

```
./filename.sh
```

🔹 Script Descriptions:

accessing_var.sh

Description: Demonstrates how to declare, access, unset, and make variables read-only in shell script.

Code:

```
#!/bin/bash

firstname="Max"
lastname="Washington"
echo "$firstname $lastname"
unset lastname
echo "hello I am $firstname  $lastname!"
username=$firstname $lastname
userage="21"
user_blood_grp="O+"
readonly user_blood_grp
user_blood_grp="B-"
echo "hello I am $username , I am $userage years old and my blood group is $user_blood_grp"
```

dispaly_msg.sh

Description: Takes the user's age and displays an appropriate message using conditional statements (if-elif-else).

Code:

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

```
echo -n "enter your age : "
read Age
if [ $Age -lt 13 ]; then
message="you'r a child."
elif [ $Age -lt 18 ]; then
message="you'r a teenager"
elif [ $Age -lt 40 ]; then
message="you'r an adult."
elif [ $Age -lt 75 ]; then
message="you'r old."
else
message="you'r a very very old person, time to take off
from the earth. Thank you!"
fi
echo "$message"
```

find_length.sh

Description: Calculates the area of a circle based on radius input from the user.

Code:

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

```
echo "enter the radius of circle : "
read radius
area=$(echo "3.14 * $radius * $radius" | bc)
echo "The area of circle is : $area"
```

array_test.sh

Description: Demonstrates array creation, manipulation, and operations in bash scripting.

Code:

```
#!/bin/bash

arr=("jhon" "robart" "1" "jeny" "michel" "2")
echo "All elements: ${arr[@]}"
echo "Array length: ${#arr[@]}"
echo "Length of arr[1]: ${#arr[1]}"
echo "First element: ${arr[0]}"
echo "Element at index 3: ${arr[3]}"
echo "From index 2: ${arr[@]:2}"
echo "From index 3: ${arr[]:3}"
echo "From range 1 to 3: ${arr[@]:1:3}"
echo "From range 2 to 4: ${arr[]:2:4}"
search_ele=$(echo "${arr[@]}"|grep -c "michel")
```

```
echo "'michel' found: $search_ele"
replace_ele=("${arr[@]/michel/MICHEL}")
echo "After replacement: ${replace_ele[]}"
arr+=("apple")
echo "After append: ${arr[]}"
```

checkOSinfo.sh

Description: Displays operating system information using /etc/os-release file.

Code:

```
#!/bin/bash
checkOSinfo() {
if [ -f /etc/os-release ]; then
echo "OS configuration:"
cat /etc/os-release
else
echo "file not found"
fi
}
checkOSinfo
```

checkdocker.sh

Description: Checks whether Docker is installed and displays Docker version if available.

Code:

```
#!/bin/bash

checkdockerinstall() {
if ! command -v docker &>/dev/null; then
echo "docker is not installed."
return 1
else
echo "docker is already installed."
docker --version
return 0
fi
}
checkdockerinstall
```

create_ec2.sh

Description: Creates an AWS EC2 instance using the AWS CLI. Automatically installs the AWS CLI if not present.

Code:

```
#!/bin/bash

set -euo pipefail

check_awscli() {
```

```
if ! command -v aws &> /dev/null; then
return 1
fi
return 0
}

install_awscli() {
echo "Installing AWS CLI..."
curl                                                              -s
"https://awscli.amazonaws.com/awscli-exe-linux-x86_64.z
ip" -o "awscliv2.zip"
sudo apt-get install -y unzip &> /dev/null
unzip -q awscliv2.zip
sudo ./aws/install
aws --version
rm -rf awscliv2.zip ./aws
}

wait_for_instance() {
local instance_id="$1"
echo "Waiting for instance $instance_id..."
while true; do
state=$(aws   ec2   describe-instances   --instance-ids
"$instance_id"                                       --query
'Reservations[0].Instances[0].State.Name'      --output
```

```
text)
if [[ "$state" == "running" ]]; then
echo "Instance $instance_id is now running."
break
fi
sleep 10
done
}

create_ec2_instance() {
local ami_id="$1"
local instance_type="$2"
local key_name="$3"
local subnet_id="$4"
local security_group_ids="$5"
local instance_name="$6"
instance_id=$(aws ec2 run-instances
--image-id "$ami_id"
--instance-type "$instance_type"
--key-name "$key_name"
--subnet-id "$subnet_id"
--security-group-ids "$security_group_ids"
--tag-specifications
"ResourceType=instance,Tags=[{Key=Name,Value=$instance_
name}]"
```

```
--query 'Instances[0].InstanceId'
--output text)
if [[ -z "$instance_id" ]]; then
echo "Failed to create EC2 instance." >&2
exit 1
fi
echo "Instance $instance_id created."
wait_for_instance "$instance_id"
}

main() {
if ! check_awscli; then
echo "AWS CLI not found. Installing..."
install_awscli
fi
echo "Creating EC2 instance..."
AMI_ID="..." # Add your AMI
INSTANCE_TYPE="t2.micro"
KEY_NAME="..."
SUBNET_ID="..."
SECURITY_GROUP_IDS="..."
INSTANCE_NAME="..."
create_ec2_instance      "$AMI_ID"      "$INSTANCE_TYPE"
"$KEY_NAME"      "$SUBNET_ID"      "$SECURITY_GROUP_IDS"
"$INSTANCE_NAME"
```

```
}

main "$@"
```

deploy_django_app.sh

Description: Automates deployment of a Django app using Docker and Docker Compose.

Code:

```
#!/bin/bash

code_clone() {
echo "Cloning Django app..."
if [ -d "django-notes-app" ]; then
echo "Directory exists. Skipping."
else
git                                                          clone
https://github.com/Rakesh02Kumar/django-notes-app.git
|| {
echo "Clone failed."
return 1
}
fi
}

install_requirements() {
```

```
echo "Installing dependencies..."
sudo  apt-get  update  &&  sudo  apt-get  install  -y
docker.io nginx docker-compose || {
echo "Install failed."
return 1
}
}

required_restarts() {
echo "Performing restarts..."
sudo chown "$USER" /var/run/docker.sock || {
echo "Permission change failed."
return 1
}
}

deploy() {
echo "Deploying Django app..."
cd django-notes-app || return 1
docker build -t notes-app . && docker-compose up -d ||
{
echo "Deploy failed."
return 1
}
}
```

```
echo "********** DEPLOYMENT STARTED ************"
if ! code_clone; then
cd django-notes-app || exit 1
fi
if ! install_requirements; then
exit 1
fi
if ! required_restarts; then
exit 1
fi
if ! deploy; then
echo "Deployment failed. Mail the admin..."
exit 1
fi
echo "********** DEPLOYMENT DONE ************"
```