DevOps Shack

IPV4 & IPV6: Understanding IP Addresses

IPV4

1. IP Address Representation:

- Decimal Notation: 255.255.255.255
- Binary Notation: 11111111.11111111.11111111.11111111

2. Binary to Decimal Conversion:

- Binary: 11111111.11111111.11111111.11111111
- Decimal: 255.255.255.255

3. Subnetting:

- Subnet: 10.2.11.32
- Binary: 00001010.00000010.00001011.00100000
- Decimal: 10.2.11.32
- Subnet Mask: 255.255.255.224 (CIDR /27)
- Subnet Range: 10.2.11.0 to 10.2.11.31

IPV6

1. IPV6 Address:

- IPV6 addresses are 128 bits long, represented in hexadecimal separated by colons.

CIDR Notation:

1. CIDR for IPV4:

- CIDR Notation: 10.0.0.0/16
- Range: 10.0.0.0 to 10.0.255.255 (256*256 addresses)

2. CIDR for Subnetting:

- Subnet: 10.0.1.0/24
  - Range: 10.0.1.0 to 10.0.1.255
- Subnet: 10.0.2.0/24
  - Range: 10.0.2.0 to 10.0.2.255
- Subnet: 10.0.3.0/24
  - Range: 10.0.3.0 to 10.0.3.255

Networking Concepts:

Virtual Private Cloud (VPC):

A Virtual Private Cloud (VPC) is a virtual network dedicated to your AWS account. It provides logically isolated sections of the AWS Cloud where you can launch resources. Key aspects of VPC include:

Example:

Let's create a VPC with CIDR block 10.0.0.0/16:

```
VPC: 10.0.0.0/16
```

Subnetting:

Subnetting involves dividing a large network into smaller, manageable sub-networks or subnets. Subnets are created within a VPC and allow you to segment resources based on use case or security requirements.

Example:

Within the VPC, let's create two subnets:

```
Subnet 1: 10.0.1.0/24
Subnet 2: 10.0.2.0/24
```

Security Group:

A Security Group acts as a virtual firewall for your instance to control inbound and outbound traffic. It allows or denies traffic based on rules defined for the security group.

Example:

Create a security group for a web server allowing HTTP (port 80) and SSH (port 22) traffic:

```
Security Group:
  - Inbound Rule: Allow traffic on port 80 (HTTP)
  - Inbound Rule: Allow traffic on port 22 (SSH)
```

Route Table:

A Route Table contains a set of rules, called routes, that are used to determine where network traffic is directed. Each subnet in a VPC must be associated with a route table. A default route table is automatically created when you create a VPC.

Example:

Create a route table with a route to an internet gateway for public subnets:

```
Route Table:
  - Destination: 0.0.0.0/0
    Target: Internet Gateway
```

Ports:

Ports are logical constructs that represent specific communication endpoints in networking. They allow different services on the same device to utilize network resources without interference.

Example:

Suppose you have an EC2 instance in Subnet 1 running a web server. The Security Group associated with it allows traffic on port 80 (HTTP).

```
EC2 Instance (Subnet 1):
  - IP: 10.0.1.10
  - Security Group: Allow Inbound on port 80

Access the web server at http://10.0.1.10
```

Conclusion:

Understanding and configuring VPCs, subnets, security groups, route tables, and ports are essential for designing a secure and scalable network architecture in the cloud. The examples provided illustrate the basic configurations, but real-world scenarios may involve more complex setups and additional considerations based on specific use cases and security requirements.

Corprorate Shell Scripts

Real-time shell scripts are commonly used in various companies to automate tasks, manage system configurations, and perform routine operations. Below are ten examples of real-time shell scripts with detailed explanations:

1. **Backup Script:**
    - **Objective:** Automate the backup of critical data.
    - **Example Script:**

```
#!/bin/bash
backup_dir="/path/to/backup"
source_dir="/data/to/backup"
timestamp=$(date +"%Y%m%d_%H%M%S")
tar -czvf "$backup_dir/backup_$timestamp.tar.gz" "$source_dir"
```

2. **Log Rotation Script:**
    - **Objective:** Rotate and compress log files to save disk space.
    - **Example Script:**

```
#!/bin/bash
log_dir="/var/log"
find "$log_dir" -name "*.log" -exec gzip {} \;
find "$log_dir" -name "*.gz" -mtime +7 -delete
```

3. **Monitoring Script:**
    - **Objective:** Check the status of critical services and alert if they are down.
    - **Example Script:**

```
#!/bin/bash
services=("apache2" "mysql" "nginx")
for service in "${services[@]}"; do
  if ! pgrep -x "$service" > /dev/null; then
    echo "$service is not running. Restarting..."
    systemctl restart "$service"
  fi
done
```

4. **User Account Management Script:**
    - **Objective:** Automate user account creation and modification.
    - **Example Script:**

```
#!/bin/bash
username="newuser"
password="password123"
useradd -m -p $(openssl passwd -1 "$password") "$username"
```

5. **Disk Space Monitoring Script:**
   - **Objective:** Alert when disk space crosses a certain threshold.
   - **Example Script:**

```
#!/bin/bash
threshold=90
current_usage=$(df -h / | awk 'NR==2 {print $5}' | tr -d '%')
if [ "$current_usage" -gt "$threshold" ]; then
  echo "Disk space usage is above $threshold%. Please
investigate."
fi
```

6. **Database Backup Script:**
   - **Objective:** Backup a MySQL or PostgreSQL database.
   - **Example Script:**

```
#!/bin/bash
db_name="mydatabase"
username="dbuser"
password="dbpassword"
timestamp=$(date +"%Y%m%d_%H%M%S")
mysqldump -u "$username" -p"$password" "$db_name" >
"backup_$timestamp.sql"
```

7. **File Synchronization Script:**
   - **Objective:** Synchronize files between different servers.
   - **Example Script:**

```
#!/bin/bash
source_dir="/path/to/source"
dest_server="user@remote_server:/path/to/destination"
rsync -avz "$source_dir" "$dest_server"
```

8. **System Resource Usage Script:**
   - **Objective:** Monitor CPU and memory usage.
   - **Example Script:**

```
#!/bin/bash
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -
d. -f1)
mem_usage=$(free | awk '/Mem/ {print $3/$2 * 100.0}')
echo "CPU Usage: $cpu_usage%"
echo "Memory Usage: $mem_usage%"
```

9. **Automatic Software Updates Script:**
   - **Objective:** Update the system and installed packages.
   - **Example Script:**

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

```
apt update && apt upgrade -y
```

10. **Web Server Log Analysis Script:**
    - **Objective:** Analyze web server logs for important metrics.
    - **Example Script:**
    - `#!/bin/bash`
    - `log_file="/var/log/apache2/access.log"`
    - `total_requests=$(cat "$log_file" | wc -l)`
    - `unique_visitors=$(awk '{print $1}' "$log_file" | sort -u | wc -l)`
    - `echo "Total Requests: $total_requests"`
      `echo "Unique Visitors: $unique_visitors"`

These scripts serve as examples and may need customization based on specific requirements and environments. Always ensure proper testing before deploying scripts in production environments.

20 common DevOps errors that you might encounter in a Linux environment, along with their potential solutions:

1. **Permission Denied Error:**
    - **Error:** Permission denied while trying to execute a command or access a file.
    - **Solution:** Ensure that the user has the necessary permissions. Use `chmod` or `chown` to modify permissions or ownership.

2. **Out of Memory Error:**
    - **Error:** Processes being killed due to lack of memory.
    - **Solution:** Identify memory-hungry processes using tools like `top` or `htop`, optimize or scale resources accordingly.

3. **Connection Refused Error:**
    - **Error:** Unable to connect to a service or application.
    - **Solution:** Check if the service is running, the firewall settings, and if the correct port is open.

4. **Package Not Found Error:**
    - **Error:** Unable to install a package using the package manager.
    - **Solution:** Update package repositories (`apt-get update` or `yum update`) and verify package name.

5. **Kernel Panic:**
    - **Error:** The Linux kernel encounters a critical error.

- **Solution:** Analyze kernel logs, update the kernel, or troubleshoot hardware issues.

6. **Disk Space Full:**
   - **Error:** No space left on device.
   - **Solution:** Identify and remove unnecessary files, or resize partitions.
7. **Unable to Start Service:**
   - **Error:** A service fails to start.
   - **Solution:** Check service logs (`systemctl status <service>`), review configuration files, and address any dependencies.
8. **SELinux/AppArmor Denial:**
   - **Error:** Security policies prevent an operation.
   - **Solution:** Update policies or adjust security contexts using `setenforce` or `aa-complain`.
9. **DNS Resolution Issues:**
   - **Error:** Unable to resolve domain names.
   - **Solution:** Check DNS configuration (`/etc/resolv.conf`), try different DNS servers, and ensure network connectivity.
10. **Firewall Blocking Traffic:**
    - **Error:** Connections are blocked by the firewall.
    - **Solution:** Adjust firewall rules using tools like `iptables` or `firewalld`.
11. **SSL/TLS Handshake Failure:**
    - **Error:** Unable to establish a secure connection.
    - **Solution:** Check certificate validity, ensure the correct protocols are used, and troubleshoot network issues.
12. **SSH Connection Timeout:**
    - **Error:** SSH connections are timing out.
    - **Solution:** Check network connectivity, firewall settings, and SSH server logs.
13. **File Not Found:**
    - **Error:** File or directory not found.
    - **Solution:** Double-check the path and file/directory names, verify permissions, and ensure the file exists.
14. **Service Crashes on Startup:**

- **Error:** Service fails immediately upon starting.
- **Solution:** Examine logs for error messages, review configuration files, and address any missing dependencies.

15. **Incorrect Cron Job Execution:**
    - **Error:** Cron jobs not executing as expected.
    - **Solution:** Check cron syntax, ensure the correct user context, and review cron job logs.
16. **Network Unreachable:**
    - **Error:** Unable to reach a network.
    - **Solution:** Check network configuration (`ifconfig` or `ip`), ensure correct routing, and troubleshoot hardware.
17. **Slow Application Performance:**
    - **Error:** Application is responding slowly.
    - **Solution:** Identify bottlenecks using performance monitoring tools, optimize code, and scale resources.
18. **SSL Certificate Expiry:**
    - **Error:** SSL certificate has expired.
    - **Solution:** Renew the certificate, update the configuration, and restart the service.
19. **MySQL Connection Issues:**
    - **Error:** Unable to connect to the MySQL database.
    - **Solution:** Check MySQL server status, verify credentials, and review MySQL error logs.
20. **Dependency Version Conflict:**
    - **Error:** Conflicts between library or package versions.
    - **Solution:** Use virtual environments, containers, or package managers to isolate dependencies and their versions.

Always be cautious when applying solutions, and make sure they are appropriate for your specific situation. Additionally, keep backups before making significant changes to your system.