Table of Contents

1. Introduction

This documentation outlines the steps for creating and launching a Virtual Machine (VM) and setting up a development environment for a web application. The process includes deploying code, cloning a repository, configuring security, and connecting a database. These steps are crucial for web application development and deployment.

2. Prerequisites

- Amazon Web Services (AWS) account with appropriate permissions.
- Codebase for the web application, referred to as the "Rubric App."
- Access to the AWS Management Console.

3.1. Log in to your AWS Management Console

- Open a web browser and navigate to the [AWS Management Console](https://aws.amazon.com/).
- Sign in with your AWS account credentials.

3.2. Navigate to the EC2 Service

- In the AWS Management Console, find and click on the "Services" dropdown in the top-left corner.
- Select "EC2" under the "Compute" section.

3.3. Launch a New EC2 Instance

- In the EC2 Dashboard, click the "Instances" link on the left-hand menu.
- Click the "Launch Instances" button to start the instance creation wizard.

3.4. Configure Instance Details

- In the "Step 1: Choose an Amazon Machine Image (AMI)," select the AMI that matches your requirements. This choice can be a Linux distribution, Windows, or another operating system.

- Click the "Next: Configure Instance Details" button.

3.5. Configure Instance Type
- In "Step 2: Choose an Instance Type," select the instance type that suits your application's needs. Consider factors such as CPU, memory, and network performance.
- Click "Next: Add Storage" when ready.

3.6. Add Storage (Optional)
- In "Step 3: Add Storage," you can configure the amount of storage needed for your VM. Adjust the size and type of the root volume based on your application's storage requirements.
- Click "Next: Add Tags" when done.

3.7. Add Tags (Optional)
- In "Step 4: Add Tags," you can add metadata tags to your instance for easier identification and management.
- Click "Next: Configure Security Group" to proceed.

3.8. Configure Security Group
- In "Step 5: Configure Security Group," create a new security group or select an existing one. A security group acts as a virtual firewall to control inbound and outbound traffic to your instance.
- Define the inbound and outbound rules to allow traffic on specific ports. For instance, you might open port 80 for HTTP and port 22 for SSH access.
- Click "Review and Launch" when you've configured the security group.

3.9. Review and Launch
- In "Step 6: Review," review your instance configuration settings. Ensure everything is as desired.
- Click "Launch" to proceed.

3.10. Choose or Create a Key Pair
- In the "Select an existing key pair or create a new key pair" dialog, choose an existing key pair if you have one. If not, you can create a new key pair.
- This key pair is used to securely connect to your instance via SSH.
- After making your selection, click "Launch Instances."

3.11. Launch the VM
- AWS will now launch your instance. You will see a confirmation message.

3.12. Access the VM
- Once the VM is running, you can access it using SSH or other remote access methods, depending on your chosen operating system.

That's it! You've successfully created and launched a Virtual Machine (VM) on AWS. You can now proceed with the subsequent steps to set up your development environment, deploy code, and configure the VM for your application.

Certainly, let's provide more detailed steps for Section 4, "Deploy Code (Rubric App)."

4. Deploy Code (Rubric App)

4.1. Upload the Rubric App Code to the VM
- To deploy your "Rubric App" code to the VM, you'll typically use a file transfer method. The specific steps may vary depending on your preferred version control system or deployment method.

4.2. Install Necessary Dependencies and Libraries
- After uploading your code to the VM, you need to install any dependencies and libraries required for your application to run successfully. This typically involves using package managers or other installation methods specific to your programming language or framework.

Example for Node.js and npm:

```
# Navigate to the project directory
cd /path/to/your/project

# Install project dependencies
npm install
```

Example for Python and pip:

```
# Navigate to the project directory
cd /path/to/your/project
```

# Install project dependencies

```
pip install -r requirements.txt
```

4.3. Configuration Files
- Check and configure any application-specific configuration files. These files may include environment variables, database connection settings, and other runtime parameters.

4.4. Start Your Application
- Depending on your application, you may need to execute specific commands to start your application. For instance, if you're using a Node.js application, you might run:

```
node app.js
```

4.5 Configuring a Web Server (e.g., Apache or Nginx) to Serve Your Application

For Nginx, use commands like:

```
sudo apt-get update
sudo apt-get install nginx
```

Nginx uses the concept of server blocks or virtual hosts to manage multiple websites or applications on a single server. You need to create a configuration file for your application.

For Nginx, you can create a configuration file in the /etc/nginx/sites-available/ directory. Use a command like:

```
sudo nano /etc/nginx/sites-available/your-app
```

Configure the Server Block:

Inside the configuration file, you'll need to define how the web server should handle requests for your application.

<u>**Key settings include:**</u>
Server Name: Specify the domain or IP address that should be associated with your application.
Document Root: Set the directory where your application's code is located.
Proxy Configuration (if applicable): If your application is running on a different port or on another server, you might need to configure a reverse proxy.

Example Nginx configuration:

```
server {
    listen 80;
    server_name your-app-domain.com;
    root /path/to/your/app;
    # Additional configuration, if needed
}
```

Enable the Configuration:
In Nginx, after creating the configuration file, you need to enable it.

For Nginx, use:

```
sudo ln -s /etc/nginx/sites-available/your-app /etc/nginx/sites-enabled/
```

Test Configuration and Reload the Web Server:

For Nginx:

```
sudo nginx -t
```

If the test is successful, reload the web server to apply the changes:

For Nginx:

```
sudo systemctl reload nginx
```

5. Clone the Repository to the VM

5.1. Open a terminal on the VM.
A terminal is a command-line interface that allows you to interact with the operating system using text commands. It provides a way to navigate the file system, run programs, and perform various tasks. In the context of a virtual machine (VM), opening a terminal is similar to opening a command prompt or terminal window on a physical machine.

To open a terminal on the VM:

Linux/Unix-based Systems: You can usually find the terminal application in the system menu or use a keyboard shortcut like Ctrl + Alt + T.

Windows: If the VM is running Windows, you can open the Command Prompt or PowerShell by searching for it in the Start menu.

Mac: On a Mac VM, you can open the Terminal application, which is usually located in the "Utilities" folder within the "Applications" folder.

5.2. Clone the repository using a command like `git clone <repository_url>`.
Now that you have the terminal open, the next step is to clone the repository. Cloning means making a copy of the entire repository, including all its files and commit history, onto your local machine or, in this case, the VM.

Replace <repository_url> with the actual URL of the Git repository you want to clone. This URL can be obtained from the hosting service where the repository is stored, such as GitHub, GitLab, or Bitbucket.

For example, if the repository is hosted on GitHub, the command might look like this:

```
git clone https://github.com/example/repo.git
```

This command tells Git to clone the repository located at the specified URL. The repository will be copied to a new directory with the same name as the repository.

After executing this command, Git will download all the files from the repository to your VM, and you'll have a local copy that you can work with. This local copy includes the entire project history, so you can switch between different versions of the code and contribute to the development of the project.

6. Prepare the VM for SkillBuilder

6.1. Install any necessary development tools or environments required for SkillBuilder.

Before deploying SkillBuilder on the AWS VM, you need to ensure that all the required development tools and environments are installed. This step ensures that the VM has everything it needs to execute the application correctly. The specific tools and environments will depend on the technology stack used by SkillBuilder.

Example for a Node.js App:

```
# Install Node.js and npm (Node Package Manager)
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm

# Navigate to the SkillBuilder directory
cd /path/to/skillbuilder

# Install project dependencies
npm install
```

Example for a Python/Django App:

```
# Install Python and pip (Python Package Installer)
sudo apt-get update
sudo apt-get install python3
sudo apt-get install python3-pip

# Navigate to the SkillBuilder directory
cd /path/to/skillbuilder
```

```
# Install project dependencies
pip3 install -r notes.txt
```

Make sure to adapt the commands based on the requirements of the SkillBuilder application.

6.2. Configure environment variables or settings for SkillBuilder if needed.
SkillBuilder may require specific configuration settings or environment variables to run correctly. These settings can include database connection strings, API keys, and other environment-specific configurations. Setting these variables ensures that the application behaves as expected in the AWS environment.

Example: Setting Environment Variables for a Node.js App:

```
# Open the configuration file
nano /path/to/skillbuilder/.env

# Add the required environment variables
NODE_ENV=production
PORT=3000
DATABASE_URL=your_database_url
```

Example: Setting Environment Variables for a Python/Django App:

```
# Open the configuration file
nano /path/to/skillbuilder/.env

# Add the required environment variables
DEBUG=False
SECRET_KEY=your_secret_key
DATABASE_URL=your_database_url
```

Ensure that you replace placeholder values with actual configurations needed for SkillBuilder.

7. Run Front-End and Back-End

7.1. Start the front-end and back-end servers of the Rubric App using appropriate commands.

For a Node.js Front-End:

```
# Navigate to the front-end directory
cd /path/to/skillbuilder/frontend

# Install dependencies
npm install

# Start the front-end server
npm start
```

For a Python/Django Back-End:

```
# Navigate to the back-end directory
cd /path/to/skillbuilder/backend

# Apply database migrations
python manage.py migrate

# Start the back-end server
python manage.py runserver 0.0.0.0:8000
```

In this example, python manage.py runserver 0.0.0.0:8000 starts the Django development server and makes it accessible externally by binding to all network interfaces (0.0.0.0) on port 8000.

For a React Front-End with Create React App:

```
# Navigate to the front-end directory
cd /path/to/skillbuilder/frontend

# Install dependencies
npm install

# Start the front-end server
npm start
```

Make sure to run these commands in the appropriate directories, and pay attention to any error messages or logs that might indicate issues with the server startup process. Once both the front-end and back-end servers are running, you should be able to access SkillBuilder by navigating to the appropriate URL or IP address of your AWS server in a web browser.

8. Configure Security Zones and Routing on Amazon

8.1. Set up security groups, network ACLs, and route tables in your Amazon VPC to control traffic.

Security Groups:
Security groups act as virtual firewalls for your instances to control inbound and outbound traffic.

Steps:

- ➔ Go to the AWS Management Console and navigate to the EC2 dashboard.
- ➔ Under the "Network & Security" section, select "Security Groups."
- ➔ Create a new security group or modify an existing one.
- ➔ Define inbound and outbound rules based on your application's requirements. For example, if your application uses HTTP, open port 80 for inbound traffic.

Network ACLs:
Network ACLs are stateless, numbered lists that control traffic at the subnet level.

Steps:

- ➔ From the VPC dashboard, select "Network ACLs."
- ➔ Create a new network ACL or modify an existing one.
- ➔ Define inbound and outbound rules similarly to security groups, specifying the allowed IP ranges, protocols, and ports.

Route Tables:
Route tables determine where network traffic is directed. Each subnet in your VPC must be associated with a route table.

Steps:

- ➔ Under the VPC dashboard, select "Route Tables."
- ➔ Create a new route table or modify an existing one.
- ➔ Define routes to direct traffic to the desired destinations, such as the internet gateway or other VPC resources.

8.2. Ensure that the required ports are open for communication between your VM and other resources.

Define Inbound Rules:

- ● Open ports 80 (HTTP) and 443 (HTTPS) for incoming traffic.

Steps:

- ➔ For HTTP traffic, add a rule allowing incoming traffic on port 80.
- ➔ For HTTPS traffic, add a rule allowing incoming traffic on port 443

9. Install and Connect the Database
9.1. Install the database software on the VM.
9.2. Configure the database to allow connections from the VM.

9.3. Ensure the necessary ports are open for database connections.
9.4. Seed the database with data from the current app.
9.5 Accessing the database using SQLite

```
cd JSV
cd rubricapp
cd BackEndFlask
cd instance
sqlite3 account.db
```

SQLite Commands Can be Found at this Link:
https://sqlite.org/cli.html

10. Load Database with Current Data
10.1. Import the current data into the database, making sure the schema aligns with the application's requirements.

11. Test Login via Web Server
11.1. Open a web browser on the VM.
11.2. Point it to the web server and test the login functionality to ensure that the application is working correctly.

12. Limit Traffic to VM
1. Configure security groups or network policies to allow traffic only from specific IP addresses.

13. Open VM to the World
13.1. If necessary, modify the security groups to open the VM to the internet, allowing access from any IP address.

14. Conclusion
Following these steps, you will have successfully created a VM, deployed the Rubric App, configured security, connected the database, and tested your web application. Always exercise caution when opening your VM to the world, and ensure that security best practices are followed to protect your environment.

For additional guidance or specific details, refer to the video tutorial at https://www.youtube.com/watch?v=haB4uRs0ehc](https://www.youtube.com/watch?v=haB4uRs0ehc).