Packer & AMIs

Building Custom Application AMI using Packer

* Use **Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type** as your source image to create custom AMI using Packer.
  + **As of 02/24/22**, the current AMI ID is **ami-033b95fb8079dc481**.
* All AMI you build should be private.
  + Only you can deploy EC2 instances from it.
* All AMI builds should happen in your dev AWS account and shared with your demo account.
* AMI builds should be set up to run in your default VPC.
* The AMI should include everything needed to run your application and the application binary itself. For e.g., if you are using Tomcat to run your Java web application, your AMI must have Java & Tomcat installed. You should also make sure the Tomcat service will start up when an instance is launched. If you are using Python, make sure you have the right version of python and the libraries you need to be installed in the AMI.
* The packer template should be stored in the same repo as the web application.
* **For this assignment only, install MySQL or PostgreSQL locally in the AMI.**

Continuous Integration: Add New GitHub Actions Workflow for Web App

1. When a pull request is merged, a GitHub Actions workflow should be triggered to do the following:
   1. Run the unit test.
   2. Build the application artifact (war, jar, zip, etc.).
   3. Build the AMI with application dependencies and set up the application by copying the application artifacts and the configuration files.
   4. Configure the application to start automatically when VM is launched.

Infrastructure as Code w/ CloudFormation

In this assignment, you will update the CloudFormation template to add the following resources to the stack.

App Security Group

* Create an EC2 security group for your EC2 instances that will host web applications.
* Add ingress rule to allow TCP traffic on ports 22, 80, 443, and port on which your application runs from anywhere in the world.
* This security group will be referred to as the application security group.

EC2 Instance

Create an EC2 instance with the following specifications. For any parameter not provided in the table below, you may go with default values. The EC2 instance should belong to the VPC you have created.

* Application security group should be attached to this EC2 instance.
* Make sure the EBS volumes are terminated when EC2 instances are terminated.

| **Parameter** | **Value** |
| --- | --- |
| Amazon Machine Image (AMI) | Your custom AMI |
| Instance Type | t2.micro |
| Protect against accidental termination | No |
| Root Volume Size | 20 |
| Root Volume Type | General Purpose SSD (GP2) |

For Demo: Deploying Application by Launching the AMI

1. To demo a properly built AMI, launch EC2 instance with the custom AMI using the CloudFormation template to create the stack.
2. The application should work when the EC2 instance is in a "running" state.
3. No manual start/stop commands should be required and no SSHing.

Domain Name System (DNS) Setup

Register Domain Name

Namecheap Domains: Only **.me** TLD domains are free for students.

1. Register a domain name with a domain registrar such as [Namecheap](https://www.namecheap.com/domains/registration.aspx). Namecheap offers a free domain with the [Github Student Developer pack](https://education.github.com/pack).
2. Your domain name can be anything.

Configuring Amazon Route 53 For DNS Service

Your new domain will be set up to use the registrar’s name servers by default. You will need to make the following changes.

CREATE HOSTED ZONE FOR DOMAIN IN ROOT AWS ACCOUNT

**This setup is done manually from the AWS console.**

1. Create a [public hosted zone](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingHostedZone.html) in [Amazon Route 53](https://aws.amazon.com/route53/) for the domainyourdomainname.tld.
2. Configure [Namecheap to use custom name servers](https://www.namecheap.com/support/knowledgebase/article.aspx/767/10/how-can-i-change-the-nameservers-for-my-domain) provided by [Amazon Route 53](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/GetInfoAboutHostedZone.html) to use the Route53 name servers.
3. Create a type [TXT](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html#TXTFormat) record for your domain with [**TTL**](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-values-basic.html#rrsets-values-basic-ttl) of 1 minute. Type TXT record should contain the text value "csye6225-spring-2022".

CREATE SUBDOMAIN & HOSTED ZONE FOR DEV AWS ACCOUNT

**This setup is done manually from the AWS console.**

1. Create a [public hosted zone](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingHostedZone.html) in [Amazon Route 53](https://aws.amazon.com/route53/) for the subdomain dev.yourdomainname.tld.
2. Configure name servers for the subdomain in the root account. See [docs](https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingNewSubdomain.html).
3. Create a type [TXT](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html#TXTFormat) record for the subdomain with [**TTL**](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-values-basic.html#rrsets-values-basic-ttl) of 1 minute. Type TXT record should contain the text value "csye6225-spring-2022-dev".

CREATE SUBDOMAIN & HOSTED ZONE FOR DEMO AWS ACCOUNT

**This setup is done manually from the AWS console.**

1. Create a [public hosted zone](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingHostedZone.html) in [Amazon Route 53](https://aws.amazon.com/route53/) for the subdomain prod.yourdomainname.tld.
2. Configure name servers for the subdomain in the root account. See [docs](https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingNewSubdomain.html).
3. Create a type [TXT](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html#TXTFormat) record for the subdomain with [**TTL**](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-values-basic.html#rrsets-values-basic-ttl) of 1 minute. Type TXT record should contain the text value "csye6225-spring-2022-demo".

Submission

The assignment will be considered late if commits are made to the **main** branch after the due date.

1. All work for this assignment must be done on the feature branch in your fork and merged to main when you are dev complete.
2. The feature and main branches must be in sync.
3. Submit your code from all repositories in this assignment. **Read the instructions carefully to create your zip file.**
   1. Create a folder with the naming convention **firstname\_lastname\_neuid**
   2. In the **firstname\_lastname\_neuid**clone all of your GitHub (organization) repositories with the **git clone**command. It is important that you clone the repos so that your commit history and branches are preserved.
   3. Once you have cloned all of your repositories, you will create a zip of the **firstname\_lastname\_neuid\_a\_##** directory. The zip file should be **firstname\_lastname\_neuid\_a\_##.zip** where **##** is the assignment number.
   4. Now unzip the zip file in some other directory and confirm the content of the zip files.
   5. Upload the Zip to this assignment.
4. You are allowed to resubmit. If you think there may be an issue with the ZIP file, feel free to submit it again. Only the latest will be used grading.

Grading Guidelines

The following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

10% penalty will be imposed if the application crashes due to unhandled exceptions/errors.

Previous Assignment Objectives

* TAs must verify that students have resolved all open issues from the previous assignment(s).

Git Repository Content Check (10% Penalty)

* Check the repository for any IDE-specific files. IDE configuration files must not be in the repository.
  + Verify their .gitignore configuration.
* Check the repository for build artifacts such as .class, .jar, .war files and build, node\_modules directory. None of these should be checked into the repository.
* Check for dependencies. Dependencies from the Maven repository or npm should not be committed to the git repository.

Pre-requisite for Demo (5% Penalty)

* Delete all VPCs and AMIs you may have created in your demo AWS account.

Building Custom Application AMI using Packer (35%)

* Verify students are using Amazon Linux 2 as the source image for building AMI. The source AMI id may be different than the one listed on the assignment page.
  + The builder used is amazon-ebs.
* The shell provisioner(s) does the following:
  + Upgrade the OS packages.
  + Install all the application pre-requisites, middleware, and runtime.
  + Install MySQL or PostgreSQL.
  + Update permission on the copied application artifacts.
* The file provisioner(s) is running a shell script to do the following:
  + Copy the application artifact to the right location.
  + Copy the application configuration file to the right location.

CI/CD (20%)

* AMI template should be validated in the pull request status check. Ask the student to raise PR with the invalid template. PR status check should fail and block merge.
* The application artifact is built for copying to AMI.
* AMI is built when PR is merged.
* AMI is automatically shared with the DEMO account. The AWS account id is provided in the Packer template.

Domain Name Setup (10%)

* Check if students have configured their domain registrar to use Amazon Route 53 name servers. Check using <https://toolbox.googleapps.com/apps/dig/#NS/>.
* Query for TXT record for **student’s domain, dev subdomain, and prod subdomain**. Check the value returned using https://toolbox.googleapps.com/apps/dig/#TXT/ .

App Demo using AMI & Infrastructure as Code w/ CloudFormation (35%)

* Once the AMI is built and available in the DEMO AWS account, students will create the CloudFormation stack which will create the networking resources and launch EC2 instance in the newly created VPC. EC2 instance should not be in DEFAULT VPC.
* Validate CloudFormation template does not have hard-coded values. AMI id should be passed as a parameter to stack. Networking resources should be referenced within the CloudFormation stack template.
* Application endpoints from A1 & A2 should work once the instance is ready. Students cannot SSH into the instance to set up a database or start the application.
* Reboot the EC2 instance and validate the application is running automatically on reboot.