6225 – website Material

## Lecture 2 Agenda[¶](https://fall2019.csye6225.cloud/lectures/02/#agenda)

* [Team Formation](https://fall2019.csye6225.cloud/teams/)
* Introduction to Linux Command Line
* Editing with *vi*
* Introduction to Shell Scripting

Slides[¶](https://fall2019.csye6225.cloud/lectures/02/#slides)

* [Introduction to Linux Command Line](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EWYKksLeMLpMth7BdaAl0aUBYYEpwsvAWidyNiz-ja2CNA?e=l62d5q)
* [Editing with *vi*](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EblgoCvyWbtPvOefA6kb8FoBNVEY4w3DWkOl0WrosdDBmg?e=lifbpB)
* [Introduction to Shell Scripting](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EU4C9IgSFnNDj8ASNIoQ0CYBwHKNGNRJ1jcQDlHGfCFpPg?e=kg86ci)

Reading[¶](https://fall2019.csye6225.cloud/lectures/02/#reading)

Recommended Books[¶](https://fall2019.csye6225.cloud/lectures/02/#recommended-books)

* [Unix Shell Programming by Stephen G. Kochan](https://learning.oreilly.com/library/view/unix-shell-programming/0672324903/)
* [UNIX and Linux System Administration Handbook by Dan Mackin, Ben Whaley, Trent R. Hein, Garth Snyder, Evi Nemeth](https://learning.oreilly.com/library/view/unix-and-linux/9780134278308/)
* [Learning the vi and Vim Editors by Elbert Hannah, Linda Lamb, Arnold Robbins](https://learning.oreilly.com/library/view/learning-the-vi/9780596529833/)

*vi* editor[¶](https://fall2019.csye6225.cloud/lectures/02/#vi-editor)

* [Mastering the VI editor](http://web.eng.hawaii.edu/Tutor/vi.html)

Linux[¶](https://fall2019.csye6225.cloud/lectures/02/#linux)

* [The Linux Command Line](http://linuxcommand.org/tlcl.php)
* [Linux Documentation](https://linux.die.net/)
* [Linux Load Averages: Solving the Mystery](http://www.brendangregg.com/blog/2017-08-08/linux-load-averages.html)
* [Save a file (that requires root permissions) from within vim editor](https://www.cyberciti.biz/faq/vim-vi-text-editor-save-file-without-root-permission/)

Shell Scripting[¶](https://fall2019.csye6225.cloud/lectures/02/#shell-scripting)

* <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>
* <https://bash.cyberciti.biz/guide/Main_Page>
* <https://supportweb.cs.bham.ac.uk/docs/tutorials/docsystem/build/tutorials/unixscripting/unixscripting.html>

Cheatsheets[¶](https://fall2019.csye6225.cloud/lectures/02/" \l "cheatsheets" \o "Permanent link)

* [Vi Cheat Sheet](https://www.splunk.com/content/dam/splunk-blogs/images/2012/08/VIM-cheatsheet1.png)
* [UNIX/Linux Command Reference](https://files.fosswire.com/2007/08/fwunixref.pdf)
* [Bash Redirection Cheat Sheet](https://github.com/pkrumins/bash-redirections-cheat-sheet/blob/master/bash-redirections-cheat-sheet.pdf)

Assignment 2

### Setup GitHub Repository[¶](https://fall2019.csye6225.cloud/assignments/02/#setup-github-repository)

1. [Create a GitHub repository for assignments](https://help.github.com/articles/create-a-repo/) from the [template](https://github.com/tejasparikh/csye6225-fa2019-template) repository. Instructions on create a new repository from template repository can be found [here](https://help.github.com/en/articles/creating-a-repository-from-a-template). This must be a private repository that only your team, TAs and instructor can access. Make sure to create empty repository. Just like we learned in the lab, one member should create the repo and other’s should fork this repo.
2. GitHub repository name must be **ccwebapp**.
3. Add all TAs and me to your GitHub repository as collaborators. Our emails and GitHub IDs can be found on home page.
4. Update **README.md** in your repository. Your readme file must contain following:
   1. Team member information such as Name and Email address.
   2. Prerequisites for building and deploying your application locally.
   3. Build and Deploy instructions for web application.

### Web Application Development[¶](https://fall2019.csye6225.cloud/assignments/02/#web-application-development)

**Danger**

Instructor and teaching assistants will not be able to answer programming or framework related questions for the web applications you are building. We will gladly assist you with AWS & GCP SKDs.

Start implementing APIs for the Recipie Management System. See [Cloud Native Web Application Requirements](https://fall2019.csye6225.cloud/cna/) for technical requirements.

Features of the Recipie Management System application will be split among various applications. For this assignment, we will focus on the backend API (no UI) service. Additional features of the Recipie Management System will be implemented in future assignments. We will also build the infrastructure on cloud to host the application.

This assignment will focus on user management aspect of the application. You will implement RESTful APIs based on user stories you will find below.

#### RESTful API Endpoints To Be Implemented[¶](https://fall2019.csye6225.cloud/assignments/02/#restful-api-endpoints-to-be-implemented)

[API specifications](https://app.swaggerhub.com/apis-docs/csye6225/csye6225-fall2019/assignment-02)

#### User Stories[¶](https://fall2019.csye6225.cloud/assignments/02/#user-stories)

1. All API request/response payloads should be in JSON.
2. No UI should be implemented for the application.
3. As a user, I expect all APIs call to return with proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
4. As a user, I expect the code quality of the application is maintained to highest standards using unit and/or integration tests.
5. Your web application must only support [Token-Based authentication and not Session Authentication](https://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication).
6. As a user, I must provide [basic](https://en.wikipedia.org/wiki/Basic_access_authentication#Client_side) [authentication](https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication) token when making a API call to protected endpoint.
7. Create a new user
   1. As a user, I want to create an account by providing following information.
      1. Email Address
      2. Password
      3. First Name
      4. Last Name
   2. account\_created field for the user should be set to current time when user creation is successful.
   3. User should not be able to set values for account\_created and account\_updated. Any value provided for these fields must be ignored.
   4. Password field should never be returned in the response payload.
   5. As a user, I expect to use my email address as my username.
   6. As a user, I expect application to enforce strong password as recommended by [NIST](https://www.nist.gov/blogs/taking-measure/easy-ways-build-better-p5w0rd).
   7. Application must return 400 Bad Reqest HTTP response code when a user account with the email address already exists.
   8. As a user, I expect my password to be stored securely using [BCrypt password hashing scheme](https://docs.spring.io/spring-security/site/docs/current/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html) with [salt](https://en.wikipedia.org/wiki/Salt_(cryptography)).
8. Update user information
   1. As a user, I want to update my account information. I should only be allowed to update following fields.
      1. First Name
      2. Last Name
      3. Password
   2. Attempt to update any other field should return 400 Bad Reqest HTTP response code.
   3. account\_updated field for the user should be updated when user update is successful.
   4. A user can only update their own account information.
9. Get user information
   1. As a user, I want to get my account information. Response payload should return all fields for the user except for password.

## Documentation[¶](https://fall2019.csye6225.cloud/assignments/02/#documentation)

* <https://stackoverflow.com/questions/19332821/http-basic-authentication-over-ssl-for-rest-api>
* <https://en.wikipedia.org/wiki/Basic_access_authentication>
* [RESTful API Authentication Basics](https://dzone.com/articles/restful-api-authentication-basics-1)
* <https://security.stackexchange.com/questions/81756/session-authentication-vs-token-authentication>
* <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>

## Submission[¶](https://fall2019.csye6225.cloud/assignments/02/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment2** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/02/#grading-guidelines)

**Warning**

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

### Web Application (50%)[¶](https://fall2019.csye6225.cloud/assignments/02/#web-application-50)

* Students to demo the web application from their laptop. As student to download code from GitHub repository from the master branch. Students cannot use existing code on the system for demo as it could have been modified after the assignment deadline.
* Verify passwords are encrypted with BCrypt hashing and salt in the database.
* Verify that authentication is done via basic auth (token based) and not session based.
* APIs can be demoed using any Postman or Restlet or some other REST client but not via browser.
* Check for UI. Application cannot have UI.
* Check the response payload to make sure it meets the assignment objective. Password field should not be part of the response payload.
* Test for duplicate account in application. Application should NOT allow multiple accounts with same email address.
* Test updating fields such as account\_created and account\_updated. User should never be able to set values for them. These fields are on set by the application.
* Verify non-email username cannot be used for account creation.
* Verify that weak passwords cannot be used to create accounts.
  + Check for password length of 8 or shorter which should be rejected.
  + Check for simple all char passwords.
  + Check for complex passwords.
* Create multiple accounts and see if User A can get/update information for User B.

### Git (50%)[¶](https://fall2019.csye6225.cloud/assignments/02/#git-50)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

Lecture 3

Slides[¶](https://fall2019.csye6225.cloud/lectures/03/#slides)

* [Version Control with Git](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EY6tq8d2ZVNGphY18g7EGhoBA0YYA73S-jxHY05QNLUunA?e=nVXoEK)

Lab[¶](https://fall2019.csye6225.cloud/lectures/03/#lab)

* [Git Forking Workflow](https://fall2019.csye6225.cloud/lab/git/)

Reading[¶](https://fall2019.csye6225.cloud/lectures/03/#reading)

Recommended Books[¶](https://fall2019.csye6225.cloud/lectures/03/#recommended-books)

* [Git Pocket Guide](https://learning.oreilly.com/library/view/git-pocket-guide/9781449327507)

Git[¶](https://fall2019.csye6225.cloud/lectures/03/#git)

* [Pro Git](https://git-scm.com/book/en/v2)
* [Learn Git](https://www.visualstudio.com/learn-git/)
* [Understanding the GitHub Flow](https://guides.github.com/introduction/flow/)
* [Forking Workflow](https://www.atlassian.com/git/tutorials/comparing-workflows#forking-workflow)

Assignment 3

## Objectives[¶](https://fall2019.csye6225.cloud/assignments/03/#objectives)

### Setup AWS Organizations[¶](https://fall2019.csye6225.cloud/assignments/03/#setup-aws-organizations)

Enable AWS Organizations and create 2 child accounts. Call the first account dev which you will use for assignment development and second account prod which you will use to demo assignments for grading.

### Web Application Development[¶](https://fall2019.csye6225.cloud/assignments/03/#web-application-development)

**Danger**

Instructor and teaching assistants will not be able to answer programming or framework related questions for the web applications you are building. We will gladly assist you with AWS & GCP SKDs.

In this assignment you will implement APIs to allow users to create, update, delete and get recipes.

#### RESTful API Endpoints To Be Implemented[¶](https://fall2019.csye6225.cloud/assignments/03/#restful-api-endpoints-to-be-implemented)

[API specifications](https://app.swaggerhub.com/apis-docs/csye6225/fall2019/assignment03)

About the field data types in swagger docs:

1. If a field has readOnly set to true, the value in this field is populated by your application. Example field are timestamp and id. readOnly properties are included in responses but not in requests.
2. If a field has writeOnly set to true, the value in this field is provided by API caller in POST or PUT request but these fields are not part of GET request. Example is password field. writeOnly properties may be sent in requests but not in responses.
3. multipleOf keyword is used to specify that a number must be the multiple of another number.
4. The minimum and maximum keywords are used to specify the range of possible values.

#### User Stories[¶](https://fall2019.csye6225.cloud/assignments/03/#user-stories)

1. All API request/response payloads should be in JSON.
2. No UI should be implemented for the application.
3. As a user, I expect all APIs call to return with proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
4. As a user, I expect the code quality of the application is maintained to highest standards using unit and/or integration tests.
5. As a user, I want to get any recipe from the application.
6. As a user, I want to create a new recipe in the system.
   1. As a user, I expect the application to compute total time for recipe by adding up prep time and cooking time.
7. As a user, I want to update a recipe that I have created. If I tries to update a recipe owned (authored) by someone else, application should return appropriate error.
8. As a user, I want to delete a recipe that I have created. If I tries to delete a recipe owned (authored) by someone else, application should return appropriate error.

## Submission[¶](https://fall2019.csye6225.cloud/assignments/03/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment3** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/03/#grading-guidelines)

**Warning**

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

### Git (40%)[¶](https://fall2019.csye6225.cloud/assignments/03/#git-40)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

### AWS Organizations (10%)[¶](https://fall2019.csye6225.cloud/assignments/03/#aws-organizations-10)

* Validate that student has setup organization and created 2 member accounts and that they can login to them.
* Remind student to setup multi-factor authentication for member accounts as well.

### Web Application (50%)[¶](https://fall2019.csye6225.cloud/assignments/03/#web-application-50)

Note: Caller here refers to the user making the API call.

* Validate the create, update, delete, and get APIs work.
* Validate caller cannot update or delete other user’s recipe.
* Validate caller can get anyone’s recipe. Caller does not need to provide authentication token.
* Validate caller cannot set value of readOnly fields.
* Validate caller can set value of writeOnly fields and that these fields are not returned in GET request.

Lecture 4

Agenda[¶](https://fall2019.csye6225.cloud/lectures/04/#agenda)

* [Assignment #02](https://fall2019.csye6225.cloud/assignments/02/) review
* [How Assignment Grading Works](https://fall2019.csye6225.cloud/#assignment-grading)
* AWS Organization Setup
* Networking Fundamentals
* Fundamentals of Cloud Computing
* [Assignment #03](https://fall2019.csye6225.cloud/assignments/03/) discussion

Slides[¶](https://fall2019.csye6225.cloud/lectures/04/#slides)

* [Networking Fundamentals](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EQnUZMIR7F1DpLld7Ywu9d4B-06T_aPWHLPOcmqjZ9--og?e=XhFhGH)
* [Fundamentals of Cloud Computing](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EUgFvpr6h1tMk8CLj3uwPqkBp7KePFwQwJygtvMYFdS9xA?e=OOUAUn)

Reading[¶](https://fall2019.csye6225.cloud/lectures/04/#reading)

Cloud Computing[¶](https://fall2019.csye6225.cloud/lectures/04/#cloud-computing)

* [Cloud Computing: Concepts, Technology & Architecture (ISBN: 9780133387520)](https://www.pearsonhighered.com/program/Erl-Cloud-Computing-Concepts-Technology-Architecture/PGM239182.html)
* [Cloud Computing, Server Utilization, & the Environment](https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/)

Networking[¶](https://fall2019.csye6225.cloud/lectures/04/#networking)

* [Amazon Virtual Private Cloud Documentation](https://aws.amazon.com/documentation/vpc/)
* [Virtual Private Cloud (VPC) on Google Cloud Platform](https://cloud.google.com/vpc/)
* [IPv4](https://en.wikipedia.org/wiki/IPv4)
* [IPv6](https://en.wikipedia.org/wiki/IPv6)
* [Network Address Translation (NAT)](https://en.wikipedia.org/wiki/Network_address_translation)
* [Reserved IP addresses](https://en.wikipedia.org/wiki/Reserved_IP_addresses)
* [Private network](https://en.wikipedia.org/wiki/Private_network)
* [Multicast](https://en.wikipedia.org/wiki/Multicast)
* [Subnetwork](https://en.wikipedia.org/wiki/Subnetwork)
* [CIDR Notation](https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing#CIDR_notation)
* [OSI Model](https://en.wikipedia.org/wiki/OSI_model)
* [Internet Protocol Suite](https://en.wikipedia.org/wiki/Internet_protocol_suite)
* [VPC Networking Components such as Network, Interfaces, Route, Tables, Internet, Gateways, Egress-Only, Internet, Gateways, DNS, Elastic, IP, Addresses,VPC, Endpoints, NAT, VPC, Peering, ClassicLink](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Networking.html)

Assignment 4

## Objectives[¶](https://fall2019.csye6225.cloud/assignments/04/#objectives)

### AWS IAM Setup[¶](https://fall2019.csye6225.cloud/assignments/04/#aws-iam-setup)

**Danger**

No one except you should be able to modify resources in your environment.

Create [IAM](https://aws.amazon.com/iam/) users, and groups.

#### Create Groups[¶](https://fall2019.csye6225.cloud/assignments/04/#create-groups)

1. Create following groups
   1. csye6225-fa19-team
   2. csye6225-fa19-ta
2. These groups should have read-only access to AWS services and resources in your account. Search for policy pre-defined read-only policy **[arn:aws:iam::aws:policy/ReadOnlyAccess](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/policies/arn:aws:iam::aws:policy/ReadOnlyAccess$serviceLevelSummary)** and attach it to the groups.

#### Create Users[¶](https://fall2019.csye6225.cloud/assignments/04/#create-users)

1. Create user account for all of your team members and teaching assistants.
2. Use first name as the username. Do not configure credentials for the users. Leave the default setting Autogenerated password checked and copy the generated password. Autogenerated passwords are not emailed out by AWS. You need to manually send the email with password.

### Install and Configure AWS Command Line Interface[¶](https://fall2019.csye6225.cloud/assignments/04/#install-and-configure-aws-command-line-interface)

* Install and configure AWS Command Line Interface (CLI) on your development virtual machine. See [Install the AWS Command Line Interface on Linux](https://docs.aws.amazon.com/cli/latest/userguide/awscli-install-linux.html) for detailed instructions.
* Create dev profile for your dev AWS account and prod profile for your production AWS account. Default profile should not be setup.
* dev profile should be set to use us-east-1 region and prod profile should be setup to use us-east-2 region.

### Infrastructure as Code[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code)

We are going to start setting up our AWS infrastructure. This assignment will focus on setting up our networking resources such as Virtual Private Cloud (VPC), Internet Gateway, Route Table and Routes. We will use both AWS Command Line Interface, AWS CloudFormation, and Terraform for infrastructure setup and tear down.

**Naming Resources**

Wherever supported, you must provide unique names to the resources. You may be asked to create multiple networks in the same account during grading.

#### AWS Networking Setup[¶](https://fall2019.csye6225.cloud/assignments/04/#aws-networking-setup)

Here what you need to do for networking infrastructure setup:

1. Create [Virtual Private Cloud (VPC)](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc.html).
2. Create [subnets](https://docs.aws.amazon.com/vpc/latest/userguide/working-with-vpcs.html#AddaSubnet) in your VPC. You must create 3 subnets, each in different availability zone in the same region in the same VPC.
3. Create [Internet Gateway](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-internetgateway.html) resource. and [attach the Internet Gateway](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc-gateway-attachment.html) to the VPC.
4. Create a public [route table](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-route-table.html). Attach all subnets created above to the route table.
5. Create a public [route](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-route.html) in the public route table created above with destination CIDR block 0.0.0.0/0 and internet gateway created above as the target.

#### Expected Command Line Arguments[¶](https://fall2019.csye6225.cloud/assignments/04/#expected-command-line-arguments)

Variables should be declared for following properties. Values are provided as command line arguments at run time. For any property not listed below, you may set default values for them.

1. AWS region
2. VPC CIDR block
3. Subnet CIDR block
4. VPC name

#### Infrastructure as Code with AWS Command Line Interface[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-aws-command-line-interface)

For this objective, you must complete following tasks:

1. Install and setup AWS command line interface.
2. Create shell script csye6225-aws-networking-setup.sh to create and configure required networking resources using AWS CLI. Script should take all required values as parameter and should not contain hardcoded values.
3. Create shell script csye6225-aws-networking-teardown.sh to delete networking resources using AWS CLI. Script should take all required values as parameter and should not contain hardcoded values.
4. Should one of the resource creation command fail, your script should print out proper error message and exit gracefully. Your scripts are not required to rollback already created resources.
5. You must be able to use same script in the same AWS account and region to create multiple VPCs including all of it resources (listed in the “AWS Networking Setup” section) such as subnets, internet gateway, route table, etc.

#### Infrastructure as Code with AWS CloudFormation[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-aws-cloudformation)

For this objective, you must complete following tasks:

1. Install and setup AWS command line interface.
2. Create shell script csye6225-aws-cf-create-stack.sh to create and configure required networking resources using AWS CloudFormation. Script should wait until the stack is terminated and print message on script if it was success or failure.Your shell script may assume that the CloudFormation template is in the same directory as your script.
3. Create shell script csye6225-aws-cf-terminate-stack.sh to delete CloudFormation stack. Script should wait until the stack is terminated and print message on script if it was success or failure. Your shell script may assume that the CloudFormation template is in the same directory as your script.
4. Create CloudFormation template csye6225-cf-networking.json or csye6225-cf-networking.yaml that can be used to setup required networking resources. Both scripts must accept stack name among other things as parameter.
5. Do not hard code values in your shell script, or the CloudFormation template. Values must be provided on command line to the shell script as command line arguments.
6. You must be able to use same scripts and templates in the same AWS account and region to create multiple VPCs including all of it resources (listed in the “AWS Networking Setup” section) such as subnets, internet gateway, route table, etc.

#### Infrastructure as Code with Terraform[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-terraform)

For this objective, you must complete following tasks:

1. Install and setup AWS command line interface.
2. Create a terraform module for networking.
3. Networking module must setup all networking resources.
4. Values should not be hard coded in your terraform configuration files.
5. You must be able to use same terraform configuration files in the same AWS account and region to create multiple VPCs including all of it resources (listed in the “AWS Networking Setup” section) such as subnets, internet gateway, route table, etc.

#### Folder Structure for Infrastructure Code[¶](https://fall2019.csye6225.cloud/assignments/04/#folder-structure-for-infrastructure-code)

1. Create a folder structure similar to the one shown below. You can refer to [template repository](https://github.com/tejasparikh/csye6225-fa2019-template).
2. Shell scripts for Infrastructure as Code with AWS Command Line Interface must be stored in scripts directory.
3. CloudFormation templates and it’s shell scripts must be in cloudformation directory.
4. Terraform configuration files should be in the terraform directory.
5. README.md in each directory must document instructions on running the scripts contained in the folder.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | .  ├── infrastructure  │ ├── aws  │ │ ├── cloudformation  │ │ │ ├── README.md  │ │ │ ├── csye6225-aws-cf-create-stack.sh  │ │ │ ├── csye6225-aws-cf-terminate-stack.sh  │ │ │ ├── csye6225-cf-networking.json  │ │ ├── codedeploy  │ │ │ └── README.md  │ │ └── scripts  │ │ └── README.md  │ │ ├── terraform  │ │ │ └── README.md |

## Documentation[¶](https://fall2019.csye6225.cloud/assignments/04/#documentation)

### AWS Command Line Interface[¶](https://fall2019.csye6225.cloud/assignments/04/#aws-command-line-interface)

* [Install AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/installing.html)
* [Configure AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)

### AWS CloudFormation[¶](https://fall2019.csye6225.cloud/assignments/04/#aws-cloudformation)

* [AWS CloudFormation](https://aws.amazon.com/cloudformation/)
* [AWS CloudFormation Intrinsic Function Reference](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html)
* [Virtual Private Cloud](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc.html)
* [Internet Gateway](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-internetgateway.html)
* [Route Table](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-route-table.html)
* [Route](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-route.html)
* [Attaches a gateway to a VPC](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc-gateway-attachment.html)

### Terraform[¶](https://fall2019.csye6225.cloud/assignments/04/#terraform)

* [Terraform](https://www.terraform.io/)

## Submission[¶](https://fall2019.csye6225.cloud/assignments/04/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment4** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/04/#grading-guidelines)

**Warning**

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

### Previous Assignment Objectives[¶](https://fall2019.csye6225.cloud/assignments/04/#previous-assignment-objectives)

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

### Git (10%)[¶](https://fall2019.csye6225.cloud/assignments/04/#git-10)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

#### Git Repository Content Check[¶](https://fall2019.csye6225.cloud/assignments/04/#git-repository-content-check)

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

### IAM (10%)[¶](https://fall2019.csye6225.cloud/assignments/04/#iam-10)

* Check if all users (team and TAs) are created and assigned to correct group.
* Verify users have console access and not programmatic access. For this check the email students have sent to you. Did the email contain password or credentials for programmatic access or just username?
* Check the policy attached to the group is **[arn:aws:iam::aws:policy/ReadOnlyAccess](https://console.aws.amazon.com/iam/home?region=us-east-1" \l "/policies/arn:aws:iam::aws:policy/ReadOnlyAccess$serviceLevelSummary)**.
* Verify that you (TA) have received both username and password from student for IAM console. Login and verify that you can access their AWS account and your account can only READ. Try deleting their networking stack and it should not allow you to do that.
* Verify that student has MFA enabled on member accounts and the organization root account.

### Infrastructure as Code with AWS CloudFormation (20%)[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-aws-cloudformation-20)

**Info**

All demo’s must be performed by downloading code from student’s GitHub repository’s master branch through browser.

* Students should demo creation of CloudFormation stack using csye6225-aws-cf-create-stack.sh shell script.
* Students should demo termination of CloudFormation stack using csye6225-aws-cf-terminate-stack.sh shell script.
* Students should demo csye6225-aws-networking-setup.sh shell script.
* Students should demo csye6225-aws-networking-teardown.sh shell script.
* Verify that CloudFormation templates do not contain any hardcoded values.
* Try scripts with invalid parameters.
* **Create multiple VPCs using CloudFormation without any conflicts.**

### Infrastructure as Code with AWS Command Line Interface (30%)[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-aws-command-line-interface-30)

**Info**

All demo’s must be performed by downloading code from student’s GitHub repository’s master branch through browser.

* Verify AWS CLI is configured with IAM user credentials.
* Verify AWS CLI profiles are setup correctly for all member accounts and that default profile is not configured.
* Verify default regions for AWS member accounts are setup as per requirement.
* Students should demo csye6225-aws-networking-setup.sh shell script.
* Students should demo csye6225-aws-networking-teardown.sh shell script.
* Verify that shell scripts do not contain any hardcoded values.
* Try scripts with invalid parameters.
* **Create multiple VPCs using CLI without any conflicts.**

### Infrastructure as Code with Terraform (30%)[¶](https://fall2019.csye6225.cloud/assignments/04/#infrastructure-as-code-with-terraform-30)

**Info**

All demo’s must be performed by downloading code from student’s GitHub repository’s master branch through browser.

* Students should demo creation of networking resources using terraform apply command.
* Students should demo cleanup of networking resources using terraform destroy command.
* Verify that terraform configuration files do not contain any hardcoded values.
* Verify Terraform state file is not checked into the GitHub repository.
* **Create multiple VPCs using CLI and CloudFormation without any conflicts.**

Lecture 5

## Agenda[¶](https://fall2019.csye6225.cloud/lectures/05/#agenda)

* Review [Assignment #03](https://fall2019.csye6225.cloud/assignments/03/)
* Introduction to Amazon Web Services (AWS)
* Identity & Access Management (IAM)
* Infrastructure as Code
  + AWS SDK
  + AWS CLI
  + AWS CloudFormation
  + Terraform
* Discuss [Assignment #04](https://fall2019.csye6225.cloud/assignments/04/)

## Slides[¶](https://fall2019.csye6225.cloud/lectures/05/#slides)

* [Identity & Access Management](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EYin9PpFA7RKiqwXdtuUrpEBCGTEDtfR1DzFE6EyqRLMIA?e=c1TgWk)
* [Infrastructure as Code](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EYbvgg6fO05Fo2-J07WbIX0BVqEVPnf7A0A7PiyyDRNsiQ?e=2SbzFI)
* [AWS CloudFormation](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/ESkxAFRgqM5Lg1TK4uA722EBgJosuE_GqEjM3PYwYRVy8Q?e=N9eO2Z)
* [Terraform](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EduTHszoHSRHt-UvgyViXUkB1xDKOcKd-VtizNhMZ7NurA?e=8Jab0h)

## Reading[¶](https://fall2019.csye6225.cloud/lectures/05/#reading)

### IAM[¶](https://fall2019.csye6225.cloud/lectures/05/#iam)

* [Amazon Resource Names (ARNs)](https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html)
* [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/)
* [IAM Actions, Resources, and Condition Keys for AWS Services](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_actions-resources-contextkeys.html)
* [IAM JSON Policy Elements Reference](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html)

### AWS SDK[¶](https://fall2019.csye6225.cloud/lectures/05/#aws-sdk)

* [AWS Command Line Interface](https://aws.amazon.com/cli/)
  + [Install AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/installing.html)
  + [Configure AWS CLI](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)
* [Tools for Amazon Web Services](https://aws.amazon.com/tools/)

### CloudFormation[¶](https://fall2019.csye6225.cloud/lectures/05/#cloudformation)

* [AWS CloudFormation](https://aws.amazon.com/cloudformation/)
* [CloudFormation Intrinsic Function References](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html)

#### Sample CloudFormation Template[¶](https://fall2019.csye6225.cloud/lectures/05/#sample-cloudformation-template)

* [VPC](https://gist.github.com/tejasparikh/913a310591ab596e2f508b5bf9622b91)
* [EC2 Instance](https://gist.github.com/tejasparikh/8eb20a9f2b88a2e91b311057563713c7)

### Infrastructure as Code[¶](https://fall2019.csye6225.cloud/lectures/05/#infrastructure-as-code)

* <https://www.cio.com/article/3017722/what-is-infrastructure-as-code-and-why-should-you-embrace-it.html>
* <https://dzone.com/articles/how-to-leverage-infrastructure-as-code-with-devops>
* <https://www.thoughtworks.com/insights/blog/infrastructure-code-reason-smile>

### Terraform[¶](https://fall2019.csye6225.cloud/lectures/05/#terraform)

* [Terraform](https://www.terraform.io/)

Assignment 5

## Objectives[¶](https://fall2019.csye6225.cloud/assignments/05/#objectives)

### Domain Name System (DNS) Setup[¶](https://fall2019.csye6225.cloud/assignments/05/#domain-name-system-dns-setup)

#### Register Domain Name[¶](https://fall2019.csye6225.cloud/assignments/05/#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 [Github Student Developer pack](https://education.github.com/pack).
2. Your domain name can be anything.

#### Configuring Amazon Route 53 For DNS Service[¶](https://fall2019.csye6225.cloud/assignments/05/#configuring-amazon-route-53-for-dns-service)

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

##### CREATE HOSTED ZONE FOR DOMAIN IN ROOT AWS ACCOUNT[¶](https://fall2019.csye6225.cloud/assignments/05/#create-hosted-zone-for-domain-in-root-aws-account)

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/).
2. Configure [Namecheap to use custom nameservers](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 nameservers.
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-fa2019".

##### CREATE SUBDOMAIN & HOSTED ZONE FOR DEV AWS ACCOUNT[¶](https://fall2019.csye6225.cloud/assignments/05/#create-subdomain-hosted-zone-for-dev-aws-account)

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 nameservers 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-fa2019-dev".

##### CREATE SUBDOMAIN & HOSTED ZONE FOR PROD AWS ACCOUNT[¶](https://fall2019.csye6225.cloud/assignments/05/#create-subdomain-hosted-zone-for-prod-aws-account)

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 nameservers 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-fa2019-prod".

#### Subscribe to CentOS in AWS Marketplace[¶](https://fall2019.csye6225.cloud/assignments/05/#subscribe-to-centos-in-aws-marketplace)

Before you can use CentOS AMIs, you may have to manually subscribe for it at <https://aws.amazon.com/marketplace/pp/B00O7WM7QW>.

### Building Custom AMI using Packer[¶](https://fall2019.csye6225.cloud/assignments/05/#building-custom-ami-using-packer)

* Create a AMI using packer. Use **CentOS Linux 7 (AMI ID ami-9887c6e7)** as your source image.
* This AMI should be private so that only you can deploy EC2 instances from it.
* The AMI should include everything needed to run your application. 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 Tomcat service will start up when a instance is launched. If you are using Python, make sure you have the right version of python and libraries you need installed in the AMI.

A screenshot of a cell phone

Description automatically generated

## Documentation[¶](https://fall2019.csye6225.cloud/assignments/05/#documentation)

### Packer & AWS AMIs[¶](https://fall2019.csye6225.cloud/assignments/05/#packer-aws-amis)

* [Amazon Machine Images (AMI)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html)
* [HashiCorp Packer](https://www.packer.io/)

## Submission[¶](https://fall2019.csye6225.cloud/assignments/05/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment5** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/05/#grading-guidelines)

**Warning**

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

### Previous Assignment Objectives[¶](https://fall2019.csye6225.cloud/assignments/05/#previous-assignment-objectives)

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

### Git (10%)[¶](https://fall2019.csye6225.cloud/assignments/05/#git-10)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

#### Git Repository Content Check[¶](https://fall2019.csye6225.cloud/assignments/05/#git-repository-content-check)

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

### Domain Name Setup (30%)[¶](https://fall2019.csye6225.cloud/assignments/05/#domain-name-setup-30)

* 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/>.

### Building AMIs (60%)[¶](https://fall2019.csye6225.cloud/assignments/05/#building-amis-60)

Students must demo following:

1. Build AMI using packer.
2. Launch EC2 instance from the AMI
3. Deploy application on the EC2 instance. Students can do manual install of database, schema, and application configuration.
4. Verify application APIs can be accessed from the IP address of EC2 instance. Security group may be manually configured to permit access to the application.

Lecture 6

Agenda[¶](https://fall2019.csye6225.cloud/lectures/06/#agenda)

* Review [Assignment #04](https://fall2019.csye6225.cloud/assignments/04/)
* Virtualization
* Network Firewalls & Security Groups
* Domain Name System (DNS)
* AWS Route 53
* Amazon Machine Images (AMI)
* HashiCorp Packer
* Discuss [Assignment #05](https://fall2019.csye6225.cloud/assignments/05/)

Slides[¶](https://fall2019.csye6225.cloud/lectures/06/#slides)

* [Virtualization](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/Ec2ZVHzXTTZBieKEqZYJ-woBSMoeZ3imX_s-3bQeZesw3Q?e=4wQtxl)
* [Network Firewalls & Security Groups](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EYtQHvrOK71Lt2-m7NLFGTwBidbPXZwDW1jf52zdJNlv2A?e=O0lOGX)
* [Domain Name System (DNS)](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EeeJhByM3fVHpVWmSZWofEwBa6-oABVs40FZBPn4hBzb2w?e=sdQTM0)
* [Amazon Machine Images (AMI) & HashiCorp Packer](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EULF3Gz-zrRAtzCwGByAUR4B_W0ED6N83P1KDPZNwSfOWA?e=5EIkh0)

Amazon AMI & HashiCorp Packer[¶](https://fall2019.csye6225.cloud/lectures/06/#amazon-ami-hashicorp-packer)

Packer Template[¶](https://fall2019.csye6225.cloud/lectures/06/#packer-template)

* <https://github.com/tejasparikh/fall2019-ami>

Validate Packer Template[¶](https://fall2019.csye6225.cloud/lectures/06/#validate-packer-template)

|  |  |
| --- | --- |
| 1 | ./packer validate ubuntu-ami.json |

Build AMI[¶](https://fall2019.csye6225.cloud/lectures/06/#build-ami)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ./packer build \  -var 'aws\_access\_key=REDACTED' \  -var 'aws\_secret\_key=REDACTED' \  -var 'aws\_region=us-east-1' \  -var 'subnet\_id=REDACTED' \  ubuntu-ami.json |

Reading[¶](https://fall2019.csye6225.cloud/lectures/06/#reading)

Virtualization and AWS EC2[¶](https://fall2019.csye6225.cloud/lectures/06/#virtualization-and-aws-ec2)

* [Understanding virtualization](https://www.redhat.com/en/topics/virtualization)
* [Virtualization](https://en.wikipedia.org/wiki/Virtualization)
* [Amazon Elastic Compute Cloud (Amazon EC2)](https://aws.amazon.com/ec2/)

Network Firewalls & Security Groups[¶](https://fall2019.csye6225.cloud/lectures/06/#network-firewalls-security-groups)

* [Security Groups & Network Firewalls](http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html)
* [Amazon EC2 Security Groups for Linux Instances](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html)

DNS[¶](https://fall2019.csye6225.cloud/lectures/06/#dns)

* [Root name server](https://en.wikipedia.org/wiki/Root_name_server)
* [An Introduction to DNS Terminology, Components, and Concepts](https://www.digitalocean.com/community/tutorials/an-introduction-to-dns-terminology-components-and-concepts)
* [DNS Best Practices, Network Protections, and Attack Identification](https://www.cisco.com/c/en/us/about/security-center/dns-best-practices.html)
* [AWS Route 53](http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html)

Packer & AWS AMIs[¶](https://fall2019.csye6225.cloud/lectures/06/#packer-aws-amis)

* [Amazon Machine Images (AMI)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html)
* [HashiCorp Packer](https://www.packer.io/)

Assignment 6

## Objectives[¶](https://fall2019.csye6225.cloud/assignments/06/#objectives)

### Infrastructure as Code[¶](https://fall2019.csye6225.cloud/assignments/06/#infrastructure-as-code)

In this assignment, you will create application terraform module. Read the [docs](https://learn.hashicorp.com/terraform/getting-started/dependencies.html) to manage dependencies for different resources.

#### App Security Group[¶](https://fall2019.csye6225.cloud/assignments/06/#app-security-group)

* Create a EC2 security group for your EC2 instances that will host web application.
* 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 application security group.

#### DB Security Group[¶](https://fall2019.csye6225.cloud/assignments/06/#db-security-group)

* Create a EC2 security group for your RDS instances.
* Add ingress rule to allow TCP traffic on port 3306 for MySQL or 5432 for PostgreSQL.
* Source of the traffic should be the application security group.
* This security group will be referred to as database security group.

#### S3 Bucket[¶](https://fall2019.csye6225.cloud/assignments/06/#s3-bucket)

* Create a private S3 bucket with bucket name webapp.YOUR-DOMAIN-NAME.TLD where YOUR-DOMAIN-NAME.TLD is your domain name.
* Make sure Terraform can delete the bucket even if it is not empty. See [this](http://salewski.github.io/2017/04/30/terraform-howto-delete-a-non-empty-aws-s3-bucket.html) article.
* Enable [default encryption for S3 Buckets](https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html).
* Create a lifecycle policy for the bucket to transition objects from STANDARD storage class to STANDARD\_IA storage class after 30 days.

#### RDS Instance[¶](https://fall2019.csye6225.cloud/assignments/06/#rds-instance)

**Danger**

Setting Public accessibility to true will expose your instance to the internet.

Your RDS instance should be created with following configuration. You may use default value/settings for any property not mentioned below.

| Property | Value |
| --- | --- |
| Database Engine | MySQL/PostgreSQL/Aurora |
| DB Instance Class | db.t2.medium |
| Multi-AZ deployment | No |
| DB instance identifier | csye6225-fall2019 |
| Master username | dbuser |
| Master password | pick a strong password |
| Subnet group | Subnet for RDS instances |
| Public accessibility | Yes |
| Database name | csye6225 |

#### EC2 Instance[¶](https://fall2019.csye6225.cloud/assignments/06/#ec2-instance)

Create EC2 instance with follow specifications.

**Note**

For any parameter not provided in the table below, you may go with default values. For this assignment the EC2 instance can belong to the VPC you have created in the networking module.

* Application security group should be attached to this EC2 instance.
* Make sure the EBS volumes are terminated when EC2 instances are terminated.
* EC2 instance creation is dependent on creation of the RDS instance.

| 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) |

#### DynamoDB Table[¶](https://fall2019.csye6225.cloud/assignments/06/#dynamodb-table)

Create a DynamoDB table with provided configuration. You may use default value/settings for any property not mentioned below.

| Property | Value |
| --- | --- |
| Table Name | csye6225 |
| Primary Key | id |
| Primary Key data type | string |

### Web Application[¶](https://fall2019.csye6225.cloud/assignments/06/#web-application)

[API specifications](https://app.swaggerhub.com/apis-docs/csye6225/fall2019/assignment-06#/)

About the field data types in swagger docs:

1. If a field has readOnly set to true, the value in this field is populated by your application. Example field are timestamp and id. readOnly properties are included in responses but not in requests.
2. If a field has writeOnly set to true, the value in this field is provided by API caller in POST or PUT request but these fields are not part of GET request. Example is password field. writeOnly properties may be sent in requests but not in responses.
3. multipleOf keyword is used to specify that a number must be the multiple of another number.
4. The minimum and maximum keywords are used to specify the range of possible values.

#### User Stories[¶](https://fall2019.csye6225.cloud/assignments/06/#user-stories)

1. Fix the typo in the endpoint. Change recipie to recipe.
2. All API request/response payloads should be in JSON.
3. No UI should be implemented for the application.
4. As a user, I expect all APIs call to return with proper [HTTP status code](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes).
5. As a user, I expect the code quality of the application is maintained to highest standards using unit and/or integration tests.
6. As a user, I want to get, add, and delete pictures to the recipe. I should be authenticated & authorized to be able to perform these operations.
   1. As a user, I want to add a images to my recipe.
   2. As a user, I want to delete images attached to my recipe.
   3. Application must support popular image formats such as png, jpg, and jpeg.
7. As a user, I expect images to be stored in Amazon S3 bucket.
8. Metadata about images attached to my recipe should be stored in RDBMS such as MySQL. See AWS docs for available [metadata](https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingMetadata.html#object-metadata).
9. Updating existing image requires deleting it first and then uploading a new image.
10. Update your web application so that endpoints that use the S3 buckets and RDS instance can now work.

## Documentation[¶](https://fall2019.csye6225.cloud/assignments/06/#documentation)

* [Terraform Resource Dependencies](https://learn.hashicorp.com/terraform/getting-started/dependencies.html)
* [Random String in Terraform](https://www.terraform.io/docs/providers/random/r/string.html)
* [AWS::DynamoDB::Table](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-dynamodb-table.html)
* [AWS::RDS::DBInstance](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html)
* [AWS::RDS::DBSecurityGroup](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-security-group.html)
* [AWS::RDS::DBSecurityGroupIngress](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-rds-security-group-ingress.html)
* [AWS::EC2::SecurityGroup](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group.html)
* [AWS::EC2::SecurityGroupEgress](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-security-group-egress.html)
* [AWS::EC2::SecurityGroupIngress](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group-ingress.html)
* [Getting Started with Amazon RDS](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.html)

## Submission[¶](https://fall2019.csye6225.cloud/assignments/06/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment6** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/06/#grading-guidelines)

**Warning**

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

### Previous Assignment Objectives[¶](https://fall2019.csye6225.cloud/assignments/06/#previous-assignment-objectives)

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

### Git (20% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/06/#git-20-penalty)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

#### Git Repository Content Check[¶](https://fall2019.csye6225.cloud/assignments/06/#git-repository-content-check)

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

### Pre-requisite for Demo[¶](https://fall2019.csye6225.cloud/assignments/06/#pre-requisite-for-demo)

* Student must not have anything running in their demo (prod) account. If they have already setup something, ask them to delete it.

### Infrastructure as Code (60%)[¶](https://fall2019.csye6225.cloud/assignments/06/#infrastructure-as-code-60)

* Student’s prod account must not have any infrastructure setup prior to demo.
* Verify that 2 EC2 security groups are created. Once for EC2 instances and second one for RDS.
* For S3 bucket check following:
  + S3 bucket to store images is created using Terraform.
  + S3 bucket is private.
  + S3 bucket is encrypted.
  + S3 bucket has lifecycle policies set up to transition all objects from STANDARD storage class to standard\_IA storage class after 30 days.
* For RDS instance check following:
  + RDS instance is created in the database EC2 security group.
  + Security group rule only allows traffic from the application security group.
* For EC2 instance check following:
  + Instance(s) are created in the application security group.
  + Instance has a 20GB root volume.
  + EC2 instance is created after RDS instance has been created.
* Verify that a DynamoDB table is created in the Terraform application module. Ignore the actual table setup and configuration for this assignment.

### Web Application (40%)[¶](https://fall2019.csye6225.cloud/assignments/06/#web-application-40)

* Web application can be demoed from local environment pointing to RDS & S3 bucket in the cloud. Students can make manual change to config file for this during demos.
* Endpoints can still be recipie. There is no penalty for this.
* User is able to attach an image to the recipe after it is created. Check following:
  + Image uploaded must be png, jpg or jpeg. User should not be able to attach PDF or other formats.
  + Verify image metadata is stored in the database. Test by running SQL queries against the database. Look for metadata such as md5 hash, size, etc.
  + Through AWS console, verify uploaded images are present in S3 bucket.
  + Verify that user is able to create multiple recipes and attach same image to all of them. Each upload should be a different object in S3 bucket.
* User is able to delete images attached to recipe. Check following:
  + Deleting image attached to recipe will delete the object from S3 bucket.
  + Deleting image from one recipe should not delete image from any other recipe even if same image was uploaded to multiple recipe.
* Lecture 7  
    
  Review [Assignment #05](https://fall2019.csye6225.cloud/assignments/05/)
* Cloud Storage Solutions
  + Relational Databases
  + NoSQL Databases
  + Object Storage
  + Instance Store (Local Hard Drive)
  + Block Storage
* Discuss [Assignment #06](https://fall2019.csye6225.cloud/assignments/06/)

Slides[¶](https://fall2019.csye6225.cloud/lectures/07/#slides)

* [Cloud Storage Solutions](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/ESxBVCiOnSFGtnBIDCzYTPUBP3x2jXDofAH3A-lxnS-8gQ?e=y9xnsi)

Reading[¶](https://fall2019.csye6225.cloud/lectures/07/#reading)

AWS Cloud Storage Solutions[¶](https://fall2019.csye6225.cloud/lectures/07/#aws-cloud-storage-solutions)

* [AWS Storage Services Whitepaper](https://d1.awsstatic.com/whitepapers/Storage/AWS%20Storage%20Services%20Whitepaper-v9.pdf)
* [Cloud Storage with AWS](https://aws.amazon.com/products/storage/?nc2=h_l3_db)
* [Amazon Elastic Block Storage (Amazon EBS)](https://aws.amazon.com/products/storage/?nc2=h_l3_db#ebs)
* [Amazon Elastic File System (Amazon EFS)](https://aws.amazon.com/products/storage/?nc2=h_l3_db#efs)
* [Amazon Simple Storage Service (Amazon S3)](https://aws.amazon.com/products/storage/?nc2=h_l3_db#s3)
* [Cloud Databases with AWS](https://aws.amazon.com/products/databases/?nc2=h_l3_db)
* [Amazon RDS](https://aws.amazon.com/rds/)
* [Amazon DynamoDB](https://aws.amazon.com/dynamodb/)
* [Amazon EC2 Security Groups for Linux Instances](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html)
* [Amazon RDS Security Groups](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html)

## Assignment 7 Objectives[¶](https://fall2019.csye6225.cloud/assignments/07/#objectives)

Assignment objective is to setup continuous integration and deployment pipeline with Github, CircleCI & AWS CodeDeploy.

### Sign up for CircleCI[¶](https://fall2019.csye6225.cloud/assignments/07/" \l "sign-up-for-circleci" \o "Permanent link)

1. Sign up for [CircleCI](https://circleci.com/) using your GitHub account.
2. Free plan provides **1000 build minutes** per month. If you exceed the free tier usage, you will have to pay.
3. Enable all of your repositories for build in CircleCI.

### Create IAM User for CircleCI[¶](https://fall2019.csye6225.cloud/assignments/07/" \l "create-iam-user-for-circleci" \o "Permanent link)

**Note**

This one time setup can be done through AWS console.

* Create a new user circleci with programmatic access only.
* Note policies are attached to this user via Terraform.

### AMI Updates[¶](https://fall2019.csye6225.cloud/assignments/07/#ami-updates)

#### Install CodeDeploy Agent[¶](https://fall2019.csye6225.cloud/assignments/07/#install-codedeploy-agent)

* AMI should now have CodeDeploy agent [installed](https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html).
* The region for CodeDeploy agent install should be the same region where AMI is built.

#### Install CloudWatch Agent[¶](https://fall2019.csye6225.cloud/assignments/07/#install-cloudwatch-agent)

* Update your packer build template to [install](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-first-instance.html#download-CloudWatch-Agent-on-EC2-Instance-first) the [Unified CloudWatch Agent](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/UseCloudWatchUnifiedAgent.html) in your AMIs.
* You may also need to create a [systemd service](https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units) to start, stop, and restart the CloudWatch agent.
* The CloudWatch agent must be setup to start automatically when an EC2 instance is launched using your AMI.
* CloudWatch agent will not work until CodeDeploy deploys your application on the EC2 instance.

### Infrastructure as Code[¶](https://fall2019.csye6225.cloud/assignments/07/#infrastructure-as-code)

#### IAM Updates[¶](https://fall2019.csye6225.cloud/assignments/07/#iam-updates)

**Warning**

Assignment objective requires you to identify permissions needed and create appropriate roles, and policies.

In your Terraform application module, create all required roles, policies. Not all required roles, and policies are listed in this assignment.

##### IAM ROLES & POLICIES FOR AMI[¶](https://fall2019.csye6225.cloud/assignments/07/#iam-roles-policies-for-ami)

1. Create a **circleci-ec2-ami** IAM policy and attach it to the circleci user. IAM policy can be found [here](https://www.packer.io/docs/builders/amazon.html#iam-task-or-instance-role).
2. Create all the roles and policies discussed in the [lecture](https://fall2019.csye6225.cloud/lectures/08/#iam-setup).

##### EC2 IAM ROLE FOR CLOUDWATCH AGENT[¶](https://fall2019.csye6225.cloud/assignments/07/#ec2-iam-role-for-cloudwatch-agent)

* [Update the IAM role attached to EC2 instance for use with CloudWatch Agent](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/create-iam-roles-for-cloudwatch-agent.html).

#### Create S3 bucket for CodeDeploy[¶](https://fall2019.csye6225.cloud/assignments/07/" \l "create-s3-bucket-for-codedeploy" \o "Permanent link)

**Note**

This setup can be done through AWS console.

* Create a private S3 bucket in same region as your EC2 instance with bucket name codedeploy.YOUR-DOMAIN-NAME.TLD where YOUR-DOMAIN-NAME.TLD is your domain name.
* Make sure Terraform can delete the bucket even if it is not empty. See [this](http://salewski.github.io/2017/04/30/terraform-howto-delete-a-non-empty-aws-s3-bucket.html) article.
* Enable [default encryption for S3 Buckets](https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html).
* Create a lifecycle rule for the bucket to delete objects that are older than 60 days. See this [article](https://www.joe0.com/2017/05/24/amazon-s3-how-to-delete-files-older-than-x-days/) for details.

#### EC2 & Userdata[¶](https://fall2019.csye6225.cloud/assignments/07/" \l "ec2-userdata" \o "Permanent link)

Make following updates to the EC2 resource:

1. EC2 instance should be launched with [user data](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata).
2. EC2 instance should be tagged appropriately so that CodeDeploy can identify EC2 instances on which application is suppose to be deployed.
3. Database username, password, hostname should be passed to the web application using [user data](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata).
4. S3 bucket to store images must be passed to the application via EC2 user data.

#### CodeDeploy Application Resource[¶](https://fall2019.csye6225.cloud/assignments/07/#codedeploy-application-resource)

* See [lecture](https://fall2019.csye6225.cloud/lectures/08/#create-codedeploy-application) page for details.

#### CodeDeploy Deployment Group Resource[¶](https://fall2019.csye6225.cloud/assignments/07/#codedeploy-deployment-group-resource)

* See [lecture](https://fall2019.csye6225.cloud/lectures/08/#create-codedeploy-deployment-group) page for details.

### CI/CD[¶](https://fall2019.csye6225.cloud/assignments/07/#cicd)

#### CodeDeploy AppSpec[¶](https://fall2019.csye6225.cloud/assignments/07/" \l "codedeploy-appspec" \o "Permanent link)

Create AWS CodeDeploy appspec.yml to deploy your application on EC2 instances. The appspec.yml file should be in root of your repository.

#### Implement Continuous Deployment for Building AMIs[¶](https://fall2019.csye6225.cloud/assignments/07/#implement-continuous-deployment-for-building-amis)

1. Any changes pushed to your AMI GitHub repository should trigger a build with new private AMI being registered with AWS.
2. Any IAM roles and policies needed to support this workflow must be added to your Terraform application module.

#### Implement Continuous Deployment for Web Application[¶](https://fall2019.csye6225.cloud/assignments/07/#implement-continuous-deployment-for-web-application)

1. Any changes pushed to your application GitHub repository should trigger a build with artifact deployed on the EC2 instance.
2. Any IAM roles and policies needed to support this workflow must be added to your Terraform application module.

### Web Application Updates[¶](https://fall2019.csye6225.cloud/assignments/07/#web-application-updates)

Update your web application to connect to RDS instance and S3 bucket using AWS IAM roles.

1. Any IAM roles, policies created to support this requirement must be added to your Terraform application module.
2. Images must be stored in the S3 bucket when the application is running on cloud (EC2 instances).
3. Web app must connect to RDS instance when running on cloud (EC2 instances).

#### User Stories[¶](https://fall2019.csye6225.cloud/assignments/07/#user-stories)

1. As a user, I want all application log data to be available in CloudWatch.
2. As a user, I want metrics on API usage available in CloudWatch.
3. Create following custom metrics for every API we have implemented in the web application. The metrics data should be collected in CloudWatch.
   1. Count number of times each API is called.
   2. Using Timer metrics data type, time (in milliseconds) each API call so we can understand how long it takes for the application to process an API call.
   3. Using Timer metrics data type, time (in milliseconds) each database query executed by your application.
   4. Using Timer metrics data type, time (in milliseconds) each call made to AWS S3 service by your application.
4. You can retrieve custom metrics using either [StatsD](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-custom-metrics-statsd.html) or [collectd](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-custom-metrics-collectd.html).
5. CloudWatch agent configuration file must be copied over to the EC2 server when application is being deployed by CodeDeploy. You will also need to configure CloudWatch agent before starting your service.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/07/#grading-guidelines)

**Warning**

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

### Previous Assignment Objectives[¶](https://fall2019.csye6225.cloud/assignments/07/#previous-assignment-objectives)

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

### Git (20% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/07/#git-20-penalty)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

#### Git Repository Content Check[¶](https://fall2019.csye6225.cloud/assignments/07/#git-repository-content-check)

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

### SSH’ing to EC2 Instance (20% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/07/#sshing-to-ec2-instance-20-penalty)

* Infrastructure must be setup via Terraform.
* Manually setup must be done prior to demo.
* Application must be deployed thru CI/CD pipeline.
* SSH’ing into EC2 instance for any reason (other than being asked for TA) will result in 20% penalty.

### Pre-requisite for Demo (10% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/07/#pre-requisite-for-demo-10-penalty)

* Anything setup by Terraform should not exist in the account prior to demo.
* It is ok to have resources that are setup thru console such as circleci IAM user.
* Delete all existing custom AMIs from the account prior to the demo.
* Delete all existing web application zip files from the S3 bucket used to store build artifacts. Do NOT delete the bucket itself. TAs must verify the bucket is empty.
* Delete your application, networking, IAM, etc. resources that are managed by Terraform. You do not need to delete anything that is setup manually.

### CI/CD for AMIs (10%)[¶](https://fall2019.csye6225.cloud/assignments/07/#cicd-for-amis-10)

1. Using Terraform, create all AWS resources need to build AMIs. See [docs](https://www.hashicorp.com/blog/resource-targeting-in-terraform/) to target specific Terraform resources.
2. Build a new AMI using AMI CI/CD pipeline. **CircleCI build must be triggered using API. Do NOT commit to GitHub repo to trigger the build.**
3. Verify that AMI is built on top of CentOS 7 base image using the source image ami-9887c6e7 as specified in [assignment #05](https://fall2019.csye6225.cloud/assignments/05/#building-custom-ami-using-packer).

### Infrastructure as Code (30%)[¶](https://fall2019.csye6225.cloud/assignments/07/#infrastructure-as-code-30)

1. Using Terraform, create all AWS resources need to run your application. You must use the AMI you have built to launch EC2 instance.
2. TAs must verify that no resources exist in the default VPC.
3. All IAM roles and policies must be created using Terraform. Student cannot make any changes to created resources using AWS console. TAs do not have to look at specific role or policies. Check will be performed indirectly by CI/CD pipeline results.
4. During the demo, note any feature of the web application that does not work as it will be indication of IaC not implemented correctly.

### CI/CD for Web Application (30%)[¶](https://fall2019.csye6225.cloud/assignments/07/#cicd-for-web-application-30)

#### Test Web App without Modification[¶](https://fall2019.csye6225.cloud/assignments/07/#test-web-app-without-modification)

1. Students can configure CircleCI job manually before proceeding further. They are permitted to configure Environment Variables for the job.
2. Once the application stack is ready and EC2 instances are up, trigger a new CircleCI build for the web application using API. Do NOT commit to GitHub repo to trigger the build.
   1. Verify that build artifact in S3 bucket.
   2. Only one build artifact must exist in S3 bucket at this point.
3. Database schema must also be bootstrapped at this point. Student cannot run SQL scripts to setup DB schema. Verify RDS instance only has private IP address and security group only allows traffic from web application security group.
4. CodeDeploy will pull latest build artifact and deploy it.
   1. TAs to monitor progress via AWS console.
   2. Once CodeDeploy has successfully deployed the web application on all instances, TAs will start testing the web application itself.
5. TAs to verify usage of EC2 user data to pass configuration information to the EC2 instances. No AWS credentials should be passed to EC2 instances. IAM roles and policies must be used to access all AWS resources.
6. Demo that web application works without requiring any manual changes to web application or any AWS resources. Students cannot SSH to any of the EC2 instances to fix issues. Demo web application on EC2 instances. Make API call to verify application works correctly.
   1. Make an API call to create account.
   2. Make an API call using the newly created account to create a new recipe.
   3. Make an API call to attach an image to the recipe.
   4. Make API call to GET the recipe created.
   5. Check that recipe’s images is stored in the S3 bucket. Deleting the recipe should delete the image and the recipe.

#### Test Web App with Modification[¶](https://fall2019.csye6225.cloud/assignments/07/#test-web-app-with-modification)

1. Now create a new branch of master branch called a7-demo. Configure CircleCI through environment variables to build from this new branch.
2. Students will make a simple code change to the API endpoint for demo. The /v1/recipes endpoint will be moved to /v2/recipes endpoint. Commit this change directly to the a7-demo branch.
3. A build should be automatically triggered by CircleCI now.
   1. Check that a new build artifact exists in the S3 bucket.
   2. Monitor CodeDeploy status.
   3. Test changes with API calls.
   4. Make API call to GET a the recipe on /v2/recipes endpoint.
4. Changes from a7-demo git branch should not be merged into master branch.

### CloudWatch Logs (15%)[¶](https://fall2019.csye6225.cloud/assignments/07/#cloudwatch-logs-15)

* Students to demo application logs in CloudWatch. Monitor the logs as new version of the application is deployed. Look for logs generated in last 5 minutes.
* Look for log messages generated by the application from API calls made for demo.
* As there may be some lag before logs are available in CloudWatch, wait for about 1-2 minutes for data to be available.

### CloudWatch Metrics (15%)[¶](https://fall2019.csye6225.cloud/assignments/07/#cloudwatch-metrics-15)

* Students must demo application metrics are being pushed to CloudWatch.
* Verify metrics for counters and timers as you make API calls for demo.
* Note the value of these metrics before and after the demo.
* Count the number of API calls being made and see if the count displayed in dashboard add up.
* When looking at metrics in dashboard, look for data from last 5 or 15 minutes.
* As there may be some lag before metrics are available in CloudWatch, wait for about 1-2 minutes for data to be available.

Lecture 8

## Agenda[¶](https://fall2019.csye6225.cloud/lectures/08/#agenda)

* Review [Assignment #06](https://fall2019.csye6225.cloud/assignments/06/)
* Operational Visibility in Cloud
  + Logging
  + Metrics
  + Monitoring
  + Alerting
* Continuous Deployment (GitHub + CircleCI + Amazon CodeDeploy)
* Discuss [Assignment #07](https://fall2019.csye6225.cloud/assignments/07/)

## Slides[¶](https://fall2019.csye6225.cloud/lectures/08/#slides)

* [Logging, Metrics, Monitoring, & Alerting](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EUz7uPcgJc5CrUKhV-q3W1oBoDwCvYnydyirqRXe0lCrKQ?e=vMY7Ad)
* [Continuous Deployment (GitHub + CircleCI + Amazon CodeDeploy)](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/ERU4f_t-N5pEtFa1y5rlNwMB2ZltnT96Xt4Y-Bre_9g-uQ?e=ow7PIq)

## Continuous Integration & Deployment with Github, CircleCI and AWS CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "continuous-integration-deployment-with-github-circleci-and-aws-codedeploy" \o "Permanent link)

Continuous Deployment is a software development practice in which every code change goes through the entire pipeline and is put into production, automatically, resulting in many production deployments every day.

### Continuous Deployment Workflow[¶](https://fall2019.csye6225.cloud/lectures/08/#continuous-deployment-workflow)

#### AMI[¶](https://fall2019.csye6225.cloud/lectures/08/#ami)

1. Developer commits code changes to GitHub repository.
2. CircleCI will trigger a new build on commit notification.
3. CircleCI will run the build steps from CircleCI config file .circleci/config.yml from your repository. Build steps should do the following:
   1. Install awscli in your primary container by following the [AWS CLI documentation](http://docs.aws.amazon.com/cli/latest/userguide/installing.html).
   2. Download HasiCorp Packer binary and make sure it is executable.
   3. Validate packer template.
   4. Build AMI and register it with AWS.

#### Web Application[¶](https://fall2019.csye6225.cloud/lectures/08/#web-application)

1. Developer commits code changes to GitHub repository.
2. CircleCI will trigger a new build on commit notification.
3. CircleCI will run the build steps from CircleCI config file .circleci/config.yml from your repository. Build steps should do the following:
   1. Install awscli in your primary container by following the [AWS CLI documentation](http://docs.aws.amazon.com/cli/latest/userguide/installing.html).
   2. Run unit tests.
   3. Build your artifacts if all tests are successful.
   4. Zip your artifacts and upload it to AWS S3 bucket dedicated for code deploy.
   5. Call AWS CodeDeploy to deploy the latest revision of your application to the EC2 instances.

### IAM Setup[¶](https://fall2019.csye6225.cloud/lectures/08/#iam-setup)

**Note**

Replace \* with appropriate ARN name to create secure policies.

#### CodeDeploy-EC2-S3 Policy for the Server (EC2)[¶](https://fall2019.csye6225.cloud/lectures/08/#codedeploy-ec2-s3-policy-for-the-server-ec2)

CodeDeploy-EC2-S3 policy allows EC2 instances to read data from S3 buckets. This policy is required for EC2 instances to download latest application revision.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | {  "Version": "2012-10-17",  "Statement": [  {  "Action": [  "s3:Get\*",  "s3:List\*"  ],  "Effect": "Allow",  "Resource": "\*"  }  ]  } |

#### CircleCI-Upload-To-S3 Policy for CircleCI to Upload to AWS S3[¶](https://fall2019.csye6225.cloud/lectures/08/#circleci-upload-to-s3-policy-for-circleci-to-upload-to-aws-s3)

CircleCI-Upload-To-S3 policy allows CircleCI to upload artifacts from latest successful build to dedicated S3 bucket used by code deploy.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Action": [  "s3:PutObject"  ],  "Resource": [  "\*"  ]  }  ]  } |

#### CircleCI-Code-Deploy Policy for CircleCI to Call CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "circleci-code-deploy-policy-for-circleci-to-call-codedeploy" \o "Permanent link)

CircleCI-Code-Deploy policy allows CircleCI to call CodeDeploy APIs to initiate application deployment on EC2 instances.

**Note**

1. Replace AWS\_REGION with AWS region.
2. Replace AWS\_ACCOUNT\_ID with your account id.
3. Replace CODE\_DEPLOY\_APPLICATION\_NAME with your Code Deploy Application Name (Note: We haven’t created Code Deploy Application yet. So note down the application name for later steps.)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Action": [  "codedeploy:RegisterApplicationRevision",  "codedeploy:GetApplicationRevision"  ],  "Resource": [  "arn:aws:codedeploy:AWS\_REGION:AWS\_ACCOUNT\_ID:application:CODE\_DEPLOY\_APPLICATION\_NAME"  ]  },  {  "Effect": "Allow",  "Action": [  "codedeploy:CreateDeployment",  "codedeploy:GetDeployment"  ],  "Resource": [  "\*"  ]  },  {  "Effect": "Allow",  "Action": [  "codedeploy:GetDeploymentConfig"  ],  "Resource": [  "arn:aws:codedeploy:AWS\_REGION:AWS\_ACCOUNT\_ID:deploymentconfig:CodeDeployDefault.OneAtATime",  "arn:aws:codedeploy:AWS\_REGION:AWS\_ACCOUNT\_ID:deploymentconfig:CodeDeployDefault.HalfAtATime",  "arn:aws:codedeploy:AWS\_REGION:AWS\_ACCOUNT\_ID:deploymentconfig:CodeDeployDefault.AllAtOnce"  ]  }  ]  } |

#### Create New IAM User for CircleCI[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "create-new-iam-user-for-circleci" \o "Permanent link)

Create a new IAM user circleci with programmatic access only. Attach following IAM policies to this newly created user:

1. CircleCI-Upload-To-S3
2. CircleCI-Code-Deploy
3. [circleci-ec2-ami](https://www.packer.io/docs/builders/amazon.html#iam-task-or-instance-role)

#### Create CodeDeployEC2ServiceRole IAM Role for EC2 Instance(s)[¶](https://fall2019.csye6225.cloud/lectures/08/#create-codedeployec2servicerole-iam-role-for-ec2-instances)

Create a new role CodeDeployEC2ServiceRole for EC2 instances that will be used to host your web application.

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

#### Create CodeDeployServiceRole IAM Role for CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "create-codedeployservicerole-iam-role-for-codedeploy" \o "Permanent link)

Create a new role CodeDeployServiceRole for EC2 instances that will be used to host your web application.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

### Create S3 bucket for CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "create-s3-bucket-for-codedeploy" \o "Permanent link)

1. Create a S3 bucket in same region as your EC2 instance.
2. Bucket name should be codedeploy.yourdomain.tld where yourdomain.tld should be replaced with your domain name.

### Create EC2 Instance to Host Application[¶](https://fall2019.csye6225.cloud/lectures/08/#create-ec2-instance-to-host-application)

Create one or more EC2 instance to host your application. This EC2 instance must have the IAM Role **CodeDeployEC2ServiceRole**. Tag the instance with KEY and VALUE of your choice. You will need the KEY and VALUE later when creating CodeDeploy application.

A screenshot of a cell phone

Description automatically generated

### Install CodeDeploy Agent[¶](https://fall2019.csye6225.cloud/lectures/08/#install-codedeploy-agent)

[Install](https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html) Code Deploy Agent from the region your EC2 instance is running.

### CodeDeploy App Spec[¶](https://fall2019.csye6225.cloud/lectures/08/#codedeploy-app-spec)

* Create AWS CodeDeploy App Spec file. The AppSpec file is used to manage each deployment as a series of lifecycle events. The **appspec.yml** file should be in root of your repository. See App Spec documentation [here](http://docs.aws.amazon.com/codedeploy/latest/userguide/writing-app-spec.html).

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | version: 0.0  os: linux  files:  - source: ./build/libs/ROOT.war  destination: /var/lib/tomcat7/webapps  hooks:  AfterInstall:  - location: ./restartTomcat.sh  timeout: 180  runas: centos |

### Create CodeDeploy Application[¶](https://fall2019.csye6225.cloud/lectures/08/#create-codedeploy-application)

1. Application Name - **csye6225-webapp**
2. Compute Platform - **EC2/On-premises**

### Create CodeDeploy Deployment Group[¶](https://fall2019.csye6225.cloud/lectures/08/#create-codedeploy-deployment-group)

1. Deployment group name - **csye6225-webapp-deployment**
2. Service role - **CodeDeployServiceRole**
3. Deployment type - **In-place**
4. Environment Configuration - **Amazon EC2 Instances**
   1. Provide the tag group key and values.
5. Deployment settings - **CodeDeployDefault.AllAtOnce**
6. Load Balancer - **disabled**
7. Rollback - **Roll back when a deployment fails**
8. Everything else can be left to default values.

### Setting CircleCI Environment Variable[¶](https://fall2019.csye6225.cloud/lectures/08/#setting-circleci-environment-variable)

1. Add [AWS access keys](https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys) for CircleCI IAM user to CircleCI as either [project environment variables](https://circleci.com/docs/2.0/env-vars/#setting-an-environment-variable-in-a-project) or [context environment variables](https://circleci.com/docs/2.0/env-vars/#setting-an-environment-variable-in-a-context).
2. Store your Access Key ID in a variable called AWS\_ACCESS\_KEY\_ID and your Secret Access Key in a variable called AWS\_SECRET\_ACCESS\_KEY.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

### CircleCI Config Example[¶](https://fall2019.csye6225.cloud/lectures/08/#circleci-config-example)

Repo: <https://github.com/tejasparikh/csye6225-spring2019-ami>

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | version: 2  jobs:  build:  docker:  - image: circleci/python:2.7-jessie  steps:  - checkout  - run:  name: Install packages  command: sudo apt-get update && sudo apt-get install wget zip unzip -y  - run:  name: Install awscli  command: sudo pip install awscli  - run:  name: Download packer  command: |  wget -q https://releases.hashicorp.com/packer/1.3.4/packer\_1.3.4\_linux\_amd64.zip  unzip packer\*.zip  chmod +x packer  - run:  name: Validate Packer Template  command: ./packer validate ubuntu-ami-template.json  - run:  name: Build AMI  command: |  ./packer build \  -var "aws\_region=${AWS\_REGION}" \  -var "subnet\_id=${AWS\_SUBNET\_ID}" \  ubuntu-ami-template.json |

### Trigger CircleCI Build Without GitHub Commit[¶](https://fall2019.csye6225.cloud/lectures/08/#trigger-circleci-build-without-github-commit)

You can trigger CircleCI job via api from command line using curl. See <https://circleci.com/docs/2.0/api-job-trigger/>.

|  |  |
| --- | --- |
| 1  2  3 | curl -u ${CIRCLE\_API\_USER\_TOKEN} \  -d build\_parameters[CIRCLE\_JOB]=build \  https://circleci.com/api/v1.1/project/<vcs-type>/<org>/<repo>/tree/<branch> |

**Example API Call**

Here’s an example api call for triggering build for <https://github.com/tejasparikh/csye6225-spring2019-ami> repository

|  |  |
| --- | --- |
| 1  2  3 | curl -u 0a1d67cdO\_PERSONAL\_OR\_PROJECT\_TOKEN\_cdbc356b0f5 \  -d build\_parameters[CIRCLE\_JOB]=build \  https://circleci.com/api/v1.1/project/github/tejasparikh/csye6225-spring2019-ami/tree/master |

**Note**

You may be prompted Enter host password for user for password. Simply hit the enter key and it should proceed.

**Output of the API call**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | {  "compare" : null,  "previous\_successful\_build" : {  "build\_num" : 6,  "status" : "success",  "build\_time\_millis" : 323972  },  "build\_parameters" : {  "CIRCLE\_JOB" : "build"  },  "oss" : true,  "committer\_date" : null,  "body" : null,  "usage\_queued\_at" : "2019-03-10T21:23:16.702Z",  "fail\_reason" : null,  "retry\_of" : null,  "reponame" : "csye6225-spring2019-ami",  "ssh\_users" : [ ],  "build\_url" : "https://circleci.com/gh/tejasparikh/csye6225-spring2019-ami/7",  "parallel" : 1,  "failed" : null,  "branch" : "master",  "username" : "tejasparikh",  "author\_date" : null,  "why" : "api",  "user" : {  "is\_user" : true,  "login" : "tejasparikh",  "avatar\_url" : "https://avatars2.githubusercontent.com/u/25620460?v=4",  "name" : "Tejas Parikh",  "vcs\_type" : "github",  "id" : 25620460  },  "vcs\_revision" : "01aff86f5e5dad1c9658a65c1a554cdd609d30d2",  "vcs\_tag" : null,  "build\_num" : 7,  "infrastructure\_fail" : false,  "committer\_email" : null,  "previous" : {  "build\_num" : 6,  "status" : "success",  "build\_time\_millis" : 323972  },  "status" : "not\_running",  "committer\_name" : null,  "retries" : null,  "subject" : null,  "vcs\_type" : "github",  "timedout" : false,  "dont\_build" : null,  "lifecycle" : "not\_running",  "no\_dependency\_cache" : false,  "stop\_time" : null,  "ssh\_disabled" : true,  "build\_time\_millis" : null,  "picard" : null,  "circle\_yml" : {  "string" : "version: 2\njobs:\n build:\n docker:\n - image: circleci/python:2.7-jessie\n steps:\n - checkout\n - run:\n name: Install packages\n command: sudo apt-get update && sudo apt-get install wget zip unzip -y\n - run:\n name: Install awscli\n command: sudo pip install awscli\n - run:\n name: Download packer\n command: |\n wget -q https://releases.hashicorp.com/packer/1.3.4/packer\_1.3.4\_linux\_amd64.zip \n unzip packer\*.zip \n chmod +x packer\n - run:\n name: Validate Packer Template\n command: ./packer validate ubuntu-ami-template.json\n - run:\n name: Build AMI\n command: |\n ./packer build \\\n -var \"aws\_region=${AWS\_REGION}\" \\\n -var \"subnet\_id=${AWS\_SUBNET\_ID}\" \\\n ubuntu-ami-template.json\n\n"  },  "messages" : [ ],  "is\_first\_green\_build" : false,  "job\_name" : null,  "start\_time" : null,  "canceler" : null,  "platform" : "2.0",  "outcome" : null,  "vcs\_url" : "https://github.com/tejasparikh/csye6225-spring2019-ami",  "author\_name" : null,  "node" : null,  "canceled" : false,  "author\_email" : null  } |

### Troubleshooting EC2 User Data Script & CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "troubleshooting-ec2-user-data-script-codedeploy" \o "Permanent link)

1. Instance will be online before user data script execution completes. You can follow the log in /var/log/cloud-init.log to see what AWS is doing. This behavior is normal and expected.
2. Look for util.py[DEBUG]: Running command ['/var/lib/cloud/instance/scripts/part-001'] with allowed return codes [0] (shell=False, capture=False) in /var/log/cloud-init.log to see user data script execution result code.
3. Full path for shell script created from instance’s user data is /var/lib/cloud/instance/scripts/part-001. You can run this script manually on shell ./part-001 as root user to follow the execution and see the error.
4. CodeDeploy fails : HEALTH\_CONSTRAINTS The overall deployment failed because too many individual instances failed deployment, too few healthy instances are available for deployment, or some instances in your deployment group are experiencing problems." (HEALTH\_CONSTRAINTS) - This is a very generic error and you have to usually click on deployment to get down to exact issue. Also check logs in /var/log/aws/codedeploy-agent on your EC2 instance if code deploy made it that far. Usually students encounter errors due to incorrect path for files or deployment directory. Also make sure that IAM role is attached to the EC2 instance.
5. See <https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting-deployments.html> for debugging CodeDeploy issues.
6. Download Bundle failed - The specified key does not exist. - This error indicates that file does not exist in S3 bucket.
7. [Creating AWS IAM Role using cloudformation does not create RolePolicies.](https://stackoverflow.com/questions/43300573/creating-aws-iam-role-using-cloudformation-does-not-create-rolepolicies)

## Metrics, StatsD, CloudWatch, & Web Application[¶](https://fall2019.csye6225.cloud/lectures/08/#metrics-statsd-cloudwatch-web-application)

[Assignment #07](https://fall2019.csye6225.cloud/assignments/07/) requires you to instrument your application. You can use either StatsD or CollectD (or something else) for collecting metrics. Given how popular StatsD has become, I recommend using StatsD.

### What is StatsD?[¶](https://fall2019.csye6225.cloud/lectures/08/#what-is-statsd)

[StatsD](https://github.com/etsy/statsd) is a simple daemon developed and released by Etsy to aggregate and summarize application metrics. With StatsD, applications are instrumented by developers using language-specific [client libraries](https://github.com/etsy/statsd/wiki#client-implementations). These libraries will communicate with the StatsD daemon using its dead-simple protocol, and the daemon will then generate aggregate metrics and relay them to virtually any graphing or monitoring backend.

### How Does StatsD Work?[¶](https://fall2019.csye6225.cloud/lectures/08/#how-does-statsd-work)

1. You instrument your code with one of the many StatsD libraries corresponding to your application’s programming language.
2. StatsD allows you to capture different types of metrics such as Gauges, Counters, Sets, and Timing Summary Statistics.
3. The StatsD client library then sends each individual call to the StatsD server over a UDP datagram.
4. The StatsD daemon will then listen to the UDP traffic from all application libraries, aggregate data over time and “flush” it at the desired interval to the backend of your choice.
5. The monitoring backend will turn your metrics from a stream of numbers on the wire into usable charts and alert you when needed.

### What do you need to do?[¶](https://fall2019.csye6225.cloud/lectures/08/#what-do-you-need-to-do)

1. Pick a StatsD [client library](https://github.com/etsy/statsd/wiki#client-implementations) for your application based on the programming language. Go through the documentation to see how to connect to StatsD daemon.
2. Modify your application to add counters and publish the metrics using the StatsD client instance created in step 1. Your StatsD daemon is usually running on localhost on port 8125.
3. Update your CloudWatch configuration to run a StatsD daemon on your EC2 instances. Example configuration can be found [here](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-custom-metrics-statsd.html).
4. Update your code and CloudWatch configuration on the EC2 instance to see your new metrics published to CloudWatch.

### Sample CloudWatch Agent Configuration[¶](https://fall2019.csye6225.cloud/lectures/08/#sample-cloudwatch-agent-configuration)

Save following configuration to cloudwatch-config.json file

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | {  "agent": {  "metrics\_collection\_interval": 10,  "logfile": "/var/logs/amazon-cloudwatch-agent.log"  },  "logs": {  "logs\_collected": {  "files": {  "collect\_list": [  {  "file\_path": "/opt/tomcat/logs/csye6225.log",  "log\_group\_name": "csye6225\_fall2019",  "log\_stream\_name": "webapp",  }  ]  }  },  "log\_stream\_name": "cloudwatch\_log\_stream"  }  } |

### Configure CloudWatch Agent[¶](https://fall2019.csye6225.cloud/lectures/08/#configure-cloudwatch-agent)

|  |  |
| --- | --- |
| 1  2  3  4  5 | sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \  -a fetch-config \  -m ec2 \  -c file:/opt/cloudwatch-config.json \  -s |

### Screenshots[¶](https://fall2019.csye6225.cloud/lectures/08/#screenshots)

#### CWAgent[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "cwagent" \o "Permanent link)

A screenshot of a cell phone

Description automatically generated

#### Custom Metrics[¶](https://fall2019.csye6225.cloud/lectures/08/#custom-metrics)

A screenshot of a cell phone

Description automatically generated

#### Graph Showing Metrics[¶](https://fall2019.csye6225.cloud/lectures/08/#graph-showing-metrics)

A screenshot of a social media post

Description automatically generated

## Reading[¶](https://fall2019.csye6225.cloud/lectures/08/#reading)

### AWS CLI[¶](https://fall2019.csye6225.cloud/lectures/08/#aws-cli)

* [CodeDeploy](https://docs.aws.amazon.com/cli/latest/reference/deploy/index.html)
* [S3 API](https://docs.aws.amazon.com/cli/latest/reference/s3api/index.html)

### EC2[¶](https://fall2019.csye6225.cloud/lectures/08/#ec2)

* [Instance Metadata and User Data](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html)
* [EC2 UserData](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata)

### Spring[¶](https://fall2019.csye6225.cloud/lectures/08/#spring)

* [Set the active Spring profiles](https://docs.spring.io/spring-boot/docs/current/reference/html/howto-properties-and-configuration.html#howto-set-active-spring-profiles)

### IAM[¶](https://fall2019.csye6225.cloud/lectures/08/#iam)

* [AWS::IAM::Role](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-role.html)
* [AWS::IAM::Policy](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-policy.html)
* [AWS::S3::Bucket](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html)
* [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html)

### AWS CodeDeploy[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "aws-codedeploy" \o "Permanent link)

* [AWS CodeDeploy AppSpec File Reference](http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html)
* [AWS CodeDeploy](http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html)
* [CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html)
* [Overview of CodeDeploy Deployment Types](https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html#welcome-deployment-overview)
* [Troubleshooting CodeDeploy](https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting.html)

### CircleCI[¶](https://fall2019.csye6225.cloud/lectures/08/" \l "circleci" \o "Permanent link)

* [CircleCI](https://circleci.com/)
* [CircleCI Documentation](https://circleci.com/docs)
* [Using Environment Variables](https://circleci.com/docs/2.0/env-vars/)
* [Pre-Built CircleCI Docker Images](https://circleci.com/docs/2.0/circleci-images/)
* [Concepts](https://circleci.com/docs/2.0/concepts/#section=getting-started)

### CI/CD[¶](https://fall2019.csye6225.cloud/lectures/08/#cicd)

**Note**

Access ACM articles using [ACM Digital Library](https://library.northeastern.edu/research/resources/items/acm-digital-library)

* [Manual Work is a Bug](https://queue.acm.org/detail.cfm?id=3197520)
* [Continuous Delivery Sounds Great, but Will It Work Here?](https://queue.acm.org/detail.cfm?id=3190610)
* [AWS CodeDeploy AppSpec File Reference](http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html)
* [AWS CodeDeploy](http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html)
* [AWS CloudFormation - AWS::IAM::User Resource Type](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-iam-user.html)
* [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html)

### Logging & Metrics[¶](https://fall2019.csye6225.cloud/lectures/08/#logging-metrics)

* [What is Amazon CloudWatch Logs?](http://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html)
* [Amazon CloudWatch](https://aws.amazon.com/cloudwatch)
* [The Log: What every software engineer should know about real-time data’s unifying abstraction](https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying)
* [Apache logging services](http://logging.apache.org/)
* [Effective Logging](http://www.kdgregory.com/index.php?page=java.logging)
* [Java Logging Standards and Guidelines](https://wiki.base22.com/display/btg/Java+Logging+Standards+and+Guidelines)
* [Retrieve Custom Metrics with StatsD](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-custom-metrics-statsd.html)
* [StatsD Client Libraries](https://github.com/etsy/statsd/wiki#client-implementations)

Assignment 8

## Objectives[¶](https://fall2019.csye6225.cloud/assignments/08/#objectives)

### IAM Users, Roles & Policies[¶](https://fall2019.csye6225.cloud/assignments/08/#iam-users-roles-policies)

* Add IAM roles & policies needed to meet the assignment objectives to the Terraform templates.

### Amazon Simple Email Service (SES) Configuration[¶](https://fall2019.csye6225.cloud/assignments/08/#amazon-simple-email-service-ses-configuration)

**Info**

Following steps are done **manually** and only once for each domain. You will have to validate your domain (or sub domain) in each account.

* [Verify Domain in Amazon SES](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domains.html)
* [Authenticate Email with DKIM in Amazon SES](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/dkim.html)
* [Move Out of the Amazon SES Sandbox by Requesting Production Access](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/request-production-access.html)

**Warning**

* When requesting production access for email, you must submit a genuine case or AWS will deny your request.
* AWS support can take 3-5 days to respond to your request so you only have one attempt at getting your request approved to meet the assignment deadline.
* There is nothing I or the TAs can do in getting your AWS request approved.
* I do not provide template or ticket content. Writing up the use case and your needs is one of the objectives of this assignment.

### Amazon Simple Notification Service Configuration[¶](https://fall2019.csye6225.cloud/assignments/08/#amazon-simple-notification-service-configuration)

Add Amazon Simple Notification Service (Amazon SNS) topic creation to your Terraform templates. SNS CloudFormation documentation can be found [here](http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-sns-topic.html).

### Setup GitHub Repository for Lambda Functions[¶](https://fall2019.csye6225.cloud/assignments/08/#setup-github-repository-for-lambda-functions)

1. [Create a GitHub repository for assignments](https://help.github.com/articles/create-a-repo/). This must be a private repository that only your team and TAs can access. Make sure to create empty repository. Just like we learned in the lab, one member should create the repo and other’s should fork this repo.
2. GitHub repository name must be **csye6225-fa19-lambda**.
3. Add all TAs to your GitHub repository as collaborators. TAs emails and GitHub IDs can be found on home page.
4. Add a **README.md** in your repository with team member information.

### Email Service & AWS Lambda[¶](https://fall2019.csye6225.cloud/assignments/08/#email-service-aws-lambda)

**About DynamoDB TTL**

DynamoDB typically deletes expired items within 48 hours of expiration. The exact duration within which an item truly gets deleted after expiration is specific to the nature of the workload and the size of the table. Items that have expired and not been deleted will still show up in reads, queries, and scans.

<http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/howitworks-ttl.html>

#### Authentication and Access Control for AWS Lambda[¶](https://fall2019.csye6225.cloud/assignments/08/#authentication-and-access-control-for-aws-lambda)

* Add [Authentication and Access Control for AWS Lambda](https://docs.aws.amazon.com/lambda/latest/dg/lambda-auth-and-access-control.html) to your Terraform templates.

#### Implement API Endpoint To Get My Recipes (Web Application)[¶](https://fall2019.csye6225.cloud/assignments/08/#implement-api-endpoint-to-get-my-recipes-web-application)

Swagger: <https://app.swaggerhub.com/apis-docs/csye6225/fall2019/assignment-08#/authenticated/post_v1_myrecipes>

User should be able to request link to all of their recipe’s in the system via email.

1. As a user, I want to be able to request links to all the recipes I have created by calling /v1/myrecipes endpoint. This request must be authenticated.
2. As a user, I can only request the links once every 30 minutes. Additional requests made by the user in the 30 minute window should be ignored.
3. As a user, I expect the web application to send a message on email\_request SNS topic for the email service function to actually send email.

#### Implement Lambda Function[¶](https://fall2019.csye6225.cloud/assignments/08/#implement-lambda-function)

1. Lambda function will be invoked by the SNS notification. Lambda function is responsible for sending email to the user.
2. As a user, I should be able to only have 1 request token active in database (DynamoDB) at a time.
3. As a user, I expect the request token (value) and username (key) be stored in DynamoDB with TTL of 30 minutes.
4. As a user, I expect the request token to expire after 30 minutes if it is not used by then.
5. As a user, I expect the to receive links to all the recipes I have created via email.
6. As a user, if I make multiple requests when there is a active token in the database, I should only receive 1 email.

### Implement CI/CD Pipeline for Lambda Function[¶](https://fall2019.csye6225.cloud/assignments/08/#implement-cicd-pipeline-for-lambda-function)

Using the same principles as in assignment #7, implement continuous deployment for Lambda functions. Every commit to the csye6225-fa19-lambda repository should trigger a CircleCI build and deployment of your updates function to AWS Lambda.

### Security Group Updates[¶](https://fall2019.csye6225.cloud/assignments/08/#security-group-updates)

**Tip**

For testing & debugging, you may manually modify security groups to allow direct access to the instances.

* Modify web application security group ingress so that the source is load balancer for the web application port. Use the port you application is accessible on.
* Make sure that the database security group now only allows traffic from web application security group. Your database should not be accessible from anywhere else.

### The Auto Scaling Application Stack[¶](https://fall2019.csye6225.cloud/assignments/08/#the-auto-scaling-application-stack)

So far our web application has been accessible by IP address in plain text (HTTP). We will now disable direct access to our web application. The web application will now only be accessible from load balancer.

### Setup Autoscaling for EC2 Instances[¶](https://fall2019.csye6225.cloud/assignments/08/#setup-autoscaling-for-ec2-instances)

Instead of launching EC2 instances standalone, we are now going to launch them in auto-scaling group with minimum of **3** instances and maximum of **10**. Use default or value you prefer for properties not listed below.

#### Launch Configuration[¶](https://fall2019.csye6225.cloud/assignments/08/#launch-configuration)

| Key | Value |
| --- | --- |
| ImageId | Your custom AMI |
| Instance Type | t2.nano or t2.micro |
| KeyName | YOUR\_AWS\_KEYNAME |
| AssociatePublicIpAddress | True |
| UserData | SAME\_USER\_DATA\_AS\_CURRENT\_EC2\_INSTANCE |
| IAM Role | SAME\_AS\_CURRENT\_EC2\_INSTANCE |
| Resource Name | asg\_launch\_config |
| Security Groups | Updated web security group. |

#### Auto Scaling Group[¶](https://fall2019.csye6225.cloud/assignments/08/#auto-scaling-group)

**Note**

You need to [tag](https://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/ASTagging.html)([AutoScalingGroup TagProperty](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-tags.html)) EC2 instances in your Auto Scaling Group so that CodeDeploy will find them and update them when new code deployment is triggered.

| Parameter | Value |
| --- | --- |
| Cooldown | 60 |
| LaunchConfigurationName | asg\_launch\_config |
| MinSize | 3 |
| MaxSize | 10 |
| DesiredCapacity | 3 |

#### AutoScaling Policies[¶](https://fall2019.csye6225.cloud/assignments/08/#autoscaling-policies)

Create following policies:

1. Scale up policy when average CPU usage is above 5%. Increment by 1.
2. Scale down policy when average CPU usage is below 3%. Decrement by 1.

##### SAMPLE AUTOSCALING POLICY[¶](https://fall2019.csye6225.cloud/assignments/08/#sample-autoscaling-policy)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | "WebServerScaleUpPolicy": {  "Type": "AWS::AutoScaling::ScalingPolicy",  "Properties": {  "AdjustmentType": "ChangeInCapacity",  "AutoScalingGroupName": {  "Ref": "WebServerGroup"  },  "Cooldown": "60",  "ScalingAdjustment": "1"  }  },  "WebServerScaleDownPolicy": {  "Type": "AWS::AutoScaling::ScalingPolicy",  "Properties": {  "AdjustmentType": "ChangeInCapacity",  "AutoScalingGroupName": {  "Ref": "WebServerGroup"  },  "Cooldown": "60",  "ScalingAdjustment": "-1"  }  },  "CPUAlarmHigh": {  "Type": "AWS::CloudWatch::Alarm",  "Properties": {  "AlarmDescription": "Scale-up if CPU > 90% for 10 minutes",  "MetricName": "CPUUtilization",  "Namespace": "AWS/EC2",  "Statistic": "Average",  "Period": "300",  "EvaluationPeriods": "2",  "Threshold": "90",  "AlarmActions": [  {  "Ref": "WebServerScaleUpPolicy"  }  ],  "Dimensions": [  {  "Name": "AutoScalingGroupName",  "Value": {  "Ref": "WebServerGroup"  }  }  ],  "ComparisonOperator": "GreaterThanThreshold"  }  },  "CPUAlarmLow": {  "Type": "AWS::CloudWatch::Alarm",  "Properties": {  "AlarmDescription": "Scale-down if CPU < 70% for 10 minutes",  "MetricName": "CPUUtilization",  "Namespace": "AWS/EC2",  "Statistic": "Average",  "Period": "300",  "EvaluationPeriods": "2",  "Threshold": "70",  "AlarmActions": [  {  "Ref": "WebServerScaleDownPolicy"  }  ],  "Dimensions": [  {  "Name": "AutoScalingGroupName",  "Value": {  "Ref": "WebServerGroup"  }  }  ],  "ComparisonOperator": "LessThanThreshold"  }  } |

### Setup Application Load Balancer For Your Web Application[¶](https://fall2019.csye6225.cloud/assignments/08/#setup-application-load-balancer-for-your-web-application)

* EC2 instances launched in the auto-scaling group should now be load balanced.
* Add load balancing resource to your Terraform template.
* Route53 resource record for your domain name should now be an alias for your load balancer application.
* Update CodeDeploy so that code changes can be deployed to all instances.
* Setup Application load balancer to accept HTTPS traffic and forward it to your application instances.
* Load balancer should not respond to any HTTP requests.

### Securing Infrastructure with SSL Certificates[¶](https://fall2019.csye6225.cloud/assignments/08/#securing-infrastructure-with-ssl-certificates)

* Configure your load balancer to use SSL (Secure Sockets Layer) certificate to protect your web application. Update Terraform template as required.
* You can get SSL certificates using [AWS Certificate Manager](https://aws.amazon.com/certificate-manager/) which might be easiest way to get the certificates and set them up.
* [Namecheap](https://www.namecheap.com/security/ssl-certificates.aspx) offers one year SSL certificate for free with [Github Student Developer pack](https://education.github.com/pack).
* You can also get SSL certificates for free from [Let’s Encrypt](https://letsencrypt.org/).

### DNS Update[¶](https://fall2019.csye6225.cloud/assignments/08/#dns-update)

* Your Terraform template should configure Route53 so that your domain points to your load balancer and your web application is accessible thru [**https://your-domain-name.tld/**](https://your-domain-name.tld/).
* Your application must be accessible using root context i.e. [**https://your-domain-name.tld/**](https://your-domain-name.tld/) and not [**https://your-domain-name.tld/app-0.1/**](https://your-domain-name.tld/app-0.1/)

### Web Application Firewall[¶](https://fall2019.csye6225.cloud/assignments/08/#web-application-firewall)

**Note**

Covered in next lecture.

* Deploy [AWS WAF](https://aws.amazon.com/waf/) to the Application Load Balancer (ALB) that fronts your web servers running on EC2.
* Use AWS WAF to [Mitigate OWASP’s Top 10 Web Application Vulnerabilities](https://aws.amazon.com/about-aws/whats-new/2017/07/use-aws-waf-to-mitigate-owasps-top-10-web-application-vulnerabilities/).
* All WAF resources and web security rules should be added to your application cloudformation stack.

### Create JMeter Load Testing Script[¶](https://fall2019.csye6225.cloud/assignments/08/#create-jmeter-load-testing-script)

Using [Apache JMeter](http://jmeter.apache.org/) create tests that can be run against your application APIs. Your JMeter tests need to make 500 concurrent API calls to your application to create recipes. All recipes can be created under a single user account and the user account itself can be created via API call made outside of JMeter. Configure the test to run in a loop.

Your JMeter tests should be stored in your application GitHub repository.

## Documentation[¶](https://fall2019.csye6225.cloud/assignments/08/#documentation)

### JMeter[¶](https://fall2019.csye6225.cloud/assignments/08/#jmeter)

* [JMeter User Manual](https://jmeter.apache.org/usermanual/index.html)
* [How To Use Apache JMeter To Perform Load Testing on a Web Server](https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server)
* [How To Use Apache JMeter To Perform Load Testing on a Web Server](https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server)
* [Fill Forms and Submit With JMeter](https://dzone.com/articles/fill-forms-and-submit-with-jmeter-made-easy)
* [Using CSV DATA SET with JMeter](https://guide.blazemeter.com/hc/en-us/articles/206733689-Using-CSV-DATA-SET-CONFIG)
* <https://www.blazemeter.com/blog/how-use-http-basic-authentication-jmeter/>

### Amazon Simple Email Service[¶](https://fall2019.csye6225.cloud/assignments/08/#amazon-simple-email-service)

* [Getting Started Receiving Email with Amazon SES](https://docs.aws.amazon.com/ses/latest/DeveloperGuide/receiving-email-getting-started.html)
* [Set Up an Email-Receiving Pipeline](https://aws.amazon.com/getting-started/projects/setup-email-receiving-pipeline/)

### Amazon Simple Notification & AWS Lambda[¶](https://fall2019.csye6225.cloud/assignments/08/#amazon-simple-notification-aws-lambda)

* [Using AWS Lambda with Amazon SNS](https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html)
* [Invoking Lambda functions using Amazon SNS notifications](https://docs.aws.amazon.com/sns/latest/dg/sns-lambda.html)
* [Creating a .zip Deployment Package (Java)](https://docs.aws.amazon.com/lambda/latest/dg/create-deployment-pkg-zip-java.html)
* [Libraries, samples and tools to help Go developers develop AWS Lambda functions](https://github.com/aws/aws-lambda-go)
* [Lambda Function Examples (receiving-email-action-lambda-example-functions)](https://docs.aws.amazon.com/ses/latest/DeveloperGuide/receiving-email-action-lambda-example-functions.html)
* [Tutorial: Using AWS Lambda with Amazon Simple Notification Service](https://docs.aws.amazon.com/lambda/latest/dg/with-sns-example.html)

### AWS CodeDeploy & AWS Lambda[¶](https://fall2019.csye6225.cloud/assignments/08/#aws-codedeploy-aws-lambda)

* [CodeDeployLambdaServiceRole](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EUhhrggMlARFopk2c5pg1X8B7pViOK-OX_svjj9gfbTolw?e=4gOJIu)
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-steps.html#deployment-process-workflow-lambda>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-hooks.html#appspec-hooks-lambda>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments-create-console-lambda.html>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-example.html#appspec-file-example-lambda>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file-structure-resources.html>
* <https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting-deployments-lambda.html>

### Load Testing & Autoscaling[¶](https://fall2019.csye6225.cloud/assignments/08/#load-testing-autoscaling)

* [Sample Auto Scaling CloudFormation Template](https://s3-us-east-2.amazonaws.com/cloudformation-templates-us-east-2/AutoScalingMultiAZWithNotifications.template)
* [Auto Scaling AutoScalingGroup TagProperty](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-tags.html)
* [AWS::Route53::RecordSet CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-route53-recordset.html)
* [Load Testing Web Servers with Siege](https://linode.com/docs/tools-reference/tools/load-testing-with-siege/)
* <https://docs.aws.amazon.com/autoscaling/ec2/userguide/autoscaling-load-balancer.html>

## Penetration Testing Tools[¶](https://fall2019.csye6225.cloud/assignments/08/#penetration-testing-tools)

* [Kali Linux](https://www.kali.org/)
* [Kali Linux Penetration Testing Tools](http://tools.kali.org/)
* [Automatic SQL injection and database takeover tool](https://github.com/sqlmapproject/sqlmap)
* [XSSer](http://tools.kali.org/web-applications/xsser)
* [Grabber](http://rgaucher.info/beta/grabber/)
* [Wapiti](http://wapiti.sourceforge.net/)
* [W3af](http://w3af.org/)
* [Wireshark](https://www.wireshark.org/)

## Submission[¶](https://fall2019.csye6225.cloud/assignments/08/#submission)

**Danger**

Assignment will be considered late if commits are made to master and feature branch after due date.

1. All work for this assignment must be done on **assignment8** feature branch and merged to master when you are dev complete.
2. All team member’s feature and master branches must be in-sync.

## Grading Guidelines[¶](https://fall2019.csye6225.cloud/assignments/08/#grading-guidelines)

**Warning**

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

**Note**

Student must demo from prod AWS account. TAs should not spend time verifying setup in dev or root account.

### Previous Assignment Objectives[¶](https://fall2019.csye6225.cloud/assignments/08/#previous-assignment-objectives)

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

### Git (20% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/08/#git-20-penalty)

* All team members must use the Github forking workflow and their repositories (master branch which must include code for this assignment) must be in-sync. Check this by asking students to create pull request between their master branch and their scrum master’s master branch. There should be no changes. Verify that all assignment changes are in master branch.
* Students must show pull requests raised for their code changes contribution. A student who has not raised any pull request for the assignment gets 0 points for the whole assignment.
* Added TAs and instructor as collaborator to the GitHub repository.
* Verify that students have README.md file in their git repository and it contains instructions on how to build, test and deploy their application including any pre-requisites for programming language, frameworks and third-party libraries.
* Verify that dev environment is not setup in Downloads folder.
* Git repositories should be cloned locally using git/ssh protocol and not https. Verify this by running git remote -v command in the cloned repository in the VM.
* Ask students to perform git pull from scrum master’s repo and run the git status command. This must be done from terminal.

#### Git Repository Content Check[¶](https://fall2019.csye6225.cloud/assignments/08/#git-repository-content-check)

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

### SSH’ing to EC2 Instance (20% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/08/#sshing-to-ec2-instance-20-penalty)

* Infrastructure must be setup via Terraform.
* Manually setup must be done prior to demo.
* Application must be deployed thru CI/CD pipeline.
* SSH’ing into EC2 instance for any reason (other than being asked for TA) will result in 20% penalty.

### Pre-requisite for Demo (10% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/08/#pre-requisite-for-demo-10-penalty)

* Anything setup by Terraform should not exist in the account prior to demo.
* It is ok to have resources that are setup thru console such as circleci IAM user.
* Delete all existing custom AMIs from the account prior to the demo.
* Delete all existing web application zip files from the S3 bucket used to store build artifacts. Do NOT delete the bucket itself. TAs must verify the bucket is empty.
* Delete your application, networking, IAM, etc. resources that are managed by Terraform. You do not need to delete anything that is setup manually.

### Pre-requisite for Demo (10% Penalty)[¶](https://fall2019.csye6225.cloud/assignments/08/#pre-requisite-for-demo-10-penalty_1)

* Anything setup by Terraform should not exist in the prod account prior to demo.
* It is ok to have resources that are setup thru console such as circleci IAM user.
* Delete all existing custom AMIs from the account prior to the demo.
* Delete all existing zip files from the S3 bucket used to store build artifacts.
* Delete your application, networking, IAM, etc. resources that are managed by Terraform. You do not need to delete anything that is setup manually.

### Amazon SES Setup (5%)[¶](https://fall2019.csye6225.cloud/assignments/08/#amazon-ses-setup-5)

1. Verify student’s domain has production access for SES in prod account.
   1. Check Verified Emails for SES and make sure all entries are removed.
2. Verify DKIM is configured for the domain.

### Infrastructure as Code (25%)[¶](https://fall2019.csye6225.cloud/assignments/08/#infrastructure-as-code-25)

1. Using Terraform, create all AWS resources need to run your application. You must use the AMI you have built to launch EC2 instance.
2. TAs must verify that no resources exist in the default VPC.
3. All IAM roles and policies must be created using Terraform. Student cannot make any changes to created resources using AWS console. TAs do not have to look at specific role or policies. Check will be performed indirectly by CI/CD pipeline results.
4. During the demo, note any feature of the web application that does not work as it will be indication of IaC not implemented correctly.
5. Validate load balancer has WAF configured using the AWS console.

### CI/CD for Lambda (10%)[¶](https://fall2019.csye6225.cloud/assignments/08/#cicd-for-lambda-10)

1. Have student trigger CircleCI build for web application and AWS Lambda function. Build must run from master branch. This will ensure that code on the EC2 instance and in Lambda function is latest.
   1. Make sure student starts the build with CircleCI API and not by making commits to their GitHub repository.

### Web Application, SNS, SES, and Lambda Demo (30%)[¶](https://fall2019.csye6225.cloud/assignments/08/#web-application-sns-ses-and-lambda-demo-30)

1. Student to create a new account with TAs email address.
2. Make API call to create a new recipe.
3. Make an API call to /v1/myrecipes endpoint with the newly created account.
   1. Verify that you receive email.
   2. Verify email does not end up in SPAM folder.
   3. Check the entry in DynamoDB table.
4. Make 5 more API calls to /v1/myrecipes endpoint. You should have only received 1 email.
5. Validate the content of the email have valid links to the recipe user has created. Clicking on the link should take you to the web application and show you the JSON response.
6. Validate

### Autoscaling & Load Balancer (30%)[¶](https://fall2019.csye6225.cloud/assignments/08/#autoscaling-load-balancer-30)

1. Verify that application can only be accessed via load balancer.
2. Student’s domain name should be a alias or CNAME for the load balancer host name.
3. Verify that application is only accessible over https.
4. Verify that EC2 security group only allows traffic from load balancer.
5. Student should have the minimum of 3 instances running in the auto-scale group.
   1. Terminate all 3 instances and wait for the auto scaler to replace them with new instances.
   2. Once the instances are up and running, validate that web application is running and APIs work.
6. Now student must run the JMeter tests to put load on their web application and demonstrate at least 1 new instance creation is triggered by auto scaler.

Lecture 10

Announcements[¶](https://fall2019.csye6225.cloud/lectures/10/#announcements)

* Final exam for Boston students will be on 11/21/2019 at 6pm.
* Final exam for Online students will be on 11/23/2019. You will be able to take exam between Noon and 6pm EST.
* Final project details will be posted on 11/14/2019.
* Assignment #07 grading deadline has been extended to 11/19/2019 @ 6pm.
* No assignment grading will be available after 12/04/2019.
* **Last day to drop a full-semester fall class with a W grade is 12/08/2019**

Agenda[¶](https://fall2019.csye6225.cloud/lectures/10/#agenda)

* Review [Assignment #07](https://fall2019.csye6225.cloud/assignments/07/)
* Email Service
* Microservices Architecture
* Event-driven architecture (Publish/Subscribe)
* Serverless Computing
* Load Balancing
* Content Delivery Network (CDN)
* High Availability
  + Availability Zones
  + Auto Scaling
* Load Testing w/JMeter
* Discuss [Assignment #08](https://fall2019.csye6225.cloud/assignments/08/)

Slides[¶](https://fall2019.csye6225.cloud/lectures/10/#slides)

* [Load Balancing, Auto Scaling, Regions, Availability Zones, and CDN](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EaVWbVVQfjVOqVzm98MTz5cB7dV-rdphwh9L5iFQxJiErw?e=lUB5XR)
* [Microservices Architecture](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EXmJsjGs7mBPmdufiCE2i7UBM4UJyZ6L0QtX9lXWwo-TQQ?e=ESf0Pb)
* [Serverless Computing](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EUeKIDPr7QJAv9FWw5xYnQwB2j_xrdwQKtAHTG6JOlmC4Q?e=ys5Ruu)
* [Email Service](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EQ0XMsMn3P1Hht-vwkb2X14Br3lbRb6mhjA1mdr5mgiiVA?e=muy0Ae)
* [Event-driven architecture (Publish/Subscribe)](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EYKqn_39J75DgxfCaC0HmYYBXrx-DN8rbV1OziTFXjCMPw?e=ywfNOR)
* [Load Testing w/JMeter](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EXcPNiCXzjBDqzBM_virYFoBQjTK_VyNzMek3iEODUdiUw?e=eR6UTU)

Demos[¶](https://fall2019.csye6225.cloud/lectures/10/#demos)

* Apache JMeter
* AWS SNS & AWS Lambda

Reading[¶](https://fall2019.csye6225.cloud/lectures/10/#reading)

Emails[¶](https://fall2019.csye6225.cloud/lectures/10/#emails)

* [Amazon Simple Email Service](https://aws.amazon.com/ses/)

Serverless Computing[¶](https://fall2019.csye6225.cloud/lectures/10/#serverless-computing)

* [Function-as-a-Service 101: What is it?](https://blogs.cisco.com/cloud/function-as-a-service-101-what-is-it)
* [Serverless Architectures](https://martinfowler.com/articles/serverless.html)
* [What Is Serverless Computing, and Why Should I Care?](https://developer.ibm.com/openwhisk/what-is-serverless-computing/)
* [AWS re:Invent 2016: The State of Serverless Computing](https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-the-state-of-serverless-computing-svr311?from_action=save)
* [Best Practices for Working with AWS Lambda Functions](http://docs.aws.amazon.com/lambda/latest/dg/best-practices.html)
* [Environment Variables for Lambda functions](http://docs.aws.amazon.com/lambda/latest/dg/env_variables.html)

Event Driven Architecture[¶](https://fall2019.csye6225.cloud/lectures/10/#event-driven-architecture)

* [Amazon Simple Notification Service](http://docs.aws.amazon.com/sns/latest/dg/welcome.html)
* [Amazon Simple Queue Service (SQS)](https://aws.amazon.com/sqs)
* [Amazon Simple Notification Service (SNS)](https://aws.amazon.com/sns/)
* [What do you mean by “Event-Driven”?](https://martinfowler.com/articles/201701-event-driven.html)

JMeter[¶](https://fall2019.csye6225.cloud/lectures/10/#jmeter)

* [JMeter User Manual](https://jmeter.apache.org/usermanual/index.html)
* [How To Use Apache JMeter To Perform Load Testing on a Web Server](https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server)

Auto Scaling & Load Balancing[¶](https://fall2019.csye6225.cloud/lectures/10/#auto-scaling-load-balancing)

* [AWS AutoScaling](https://aws.amazon.com/autoscaling/)
* [Auto Scaling in the Amazon Cloud - Netflix](http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html)
* [AWS Elastic Load Balancing](https://aws.amazon.com/elasticloadbalancing/)
* [Classic Load Balancer](https://aws.amazon.com/elasticloadbalancing/classicloadbalancer/)
* [Application Load Balancer](https://aws.amazon.com/elasticloadbalancing/applicationloadbalancer/)
* [Netflix Software Based Load Balancer (Zuul)](https://github.com/Netflix/zuul)
* <https://f5.com/glossary/load-balancer>

Infrastructure at Scale[¶](https://fall2019.csye6225.cloud/lectures/10/#infrastructure-at-scale)

* [How Twitter Handles 3,000 Images Per Second](http://highscalability.com/blog/2016/4/20/how-twitter-handles-3000-images-per-second.html)
* [The Infrastructure Behind Twitter: Scale](https://blog.twitter.com/2017/the-infrastructure-behind-twitter-scale)
* [Caching at Reddit](https://redditblog.com/2017/1/17/caching-at-reddit/)
* [How Uber Manages A Million Writes Per Second Using Mesos And Cassandra Across Multiple Datacenters](http://highscalability.com/blog/2016/9/28/how-uber-manages-a-million-writes-per-second-using-mesos-and.html)
* [Instant Messaging at LinkedIn: Scaling to Hundreds of Thousands of Persistent Connections on One Machine](https://engineering.linkedin.com/blog/2016/10/instant-messaging-at-linkedin--scaling-to-hundreds-of-thousands-)
* [Big Data in Real-Time at Twitter](https://www.slideshare.net/nkallen/q-con-3770885)
* <https://redditblog.com/2017/04/13/how-we-built-rplace/>

High Availability & Fault Tolerance Testing[¶](https://fall2019.csye6225.cloud/lectures/10/#high-availability-fault-tolerance-testing)

* [Netflix Chaos Monkey](https://github.com/Netflix/chaosmonkey)
* [Netflix Simian Army](https://github.com/Netflix/SimianArmy/wiki)

SSL[¶](https://fall2019.csye6225.cloud/lectures/10/#ssl)

* [Let’s Encrypt](https://letsencrypt.org/)
* [An Introduction to Let’s Encrypt](https://www.digitalocean.com/community/tutorials/an-introduction-to-let-s-encrypt)
* [Why HTTPS for Everything?](https://https.cio.gov/everything/)
* [The HTTPS-Only Standard](https://https.cio.gov/)
* [Server Name Indication](https://en.wikipedia.org/wiki/Server_Name_Indication)

## Lecture 11 Announcements[¶](https://fall2019.csye6225.cloud/lectures/11/#announcements)

* **Last day to drop a full-semester fall class with a W grade is 12/08/2019**
* **Final exam for Boston students will be on 11/21/2019 at 6pm.**
* **Final exam for Online students will be on 11/23/2019. You will be able to take exam between Noon and 6pm EST.**
* **No assignments will be graded after 9pm on 12/04/2019.**

Agenda[¶](https://fall2019.csye6225.cloud/lectures/11/#agenda)

* Final Exam setup for students in online section.
* Final Exam & “Cheat Sheet” for Boston section
* Review [Assignment #08](https://fall2019.csye6225.cloud/assignments/08/)
* [Group Project Discussion](https://fall2019.csye6225.cloud/group_project/)
* Securing Cloud Infrastructure
  + Operating System
  + Network
  + AWS Account
  + Virtual Private Cloud
* Securing Your Application
  + SQL Injection
  + Cross Site Scripting (XSS)
  + Cross-Site Request Forgery (CSRF)
* Privacy
* Architecting for the Cloud: Best Practices

Slides[¶](https://fall2019.csye6225.cloud/lectures/11/#slides)

* [Cloud & Application (In)security](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/Ef3N4_xlu8dNtPm2o_5N6jwB_WP15_C58ilLep0B3FpRQg?e=Ibj35v)
* [Architecting for the Cloud](https://northeastern-my.sharepoint.com/:b:/g/personal/tejasparikh_northeastern_edu/EWKeqhGhqo5Fk7U2hJcAUmwBVfCB1U7q_47HOodIjUmCpw?e=qGWzoP)

Reading[¶](https://fall2019.csye6225.cloud/lectures/11/#reading)

High Availability & Fault Tolerance Testing[¶](https://fall2019.csye6225.cloud/lectures/11/#high-availability-fault-tolerance-testing)

* [Netflix Chaos Monkey](https://github.com/Netflix/chaosmonkey)
* [Netflix Simian Army](https://github.com/Netflix/SimianArmy/wiki)

SSL[¶](https://fall2019.csye6225.cloud/lectures/11/#ssl)

* [Let’s Encrypt](https://letsencrypt.org/)
* [An Introduction to Let’s Encrypt](https://www.digitalocean.com/community/tutorials/an-introduction-to-let-s-encrypt)
* [Why HTTPS for Everything?](https://https.cio.gov/everything/)
* [The HTTPS-Only Standard](https://https.cio.gov/)
* [Server Name Indication](https://en.wikipedia.org/wiki/Server_Name_Indication)

Security[¶](https://fall2019.csye6225.cloud/lectures/11/#security)

* [Netflix Security Monkey](https://github.com/Netflix/security_monkey)
* [97 Things Every Programmer Should Know: Collective Wisdom from the Experts](http://programmer.97things.oreilly.com/wiki/index.php/Contributions_Appearing_in_the_Book)
* [Open Web Application Security Project (OWASP)](https://www.owasp.org/index.php/Main_Page)
* [Common Vulnerabilities and Exposures](https://cve.mitre.org/)
* [The Ten Commandments of Ethical Hacking](http://www.dummies.com/programming/networking/obeying-the-ten-commandments-of-ethical-hacking/)
* <https://github.com/cloud-custodian/cloud-custodian>

Cloud Architecture[¶](https://fall2019.csye6225.cloud/lectures/11/#cloud-architecture)

* [Azure Architecture Center](https://docs.microsoft.com/en-us/azure/architecture/)
* [AWS Architecting for The Cloud](https://aws.amazon.com/whitepapers/architecting-for-the-aws-cloud-best-practices/)
* [AWS Whitepapers](https://aws.amazon.com/whitepapers/)
* [Building scalable and resilient applications](https://cloud.google.com/solutions/scalable-and-resilient-apps)

Infrastructure at Scale[¶](https://fall2019.csye6225.cloud/lectures/11/#infrastructure-at-scale)

* [How Twitter Handles 3,000 Images Per Second](http://highscalability.com/blog/2016/4/20/how-twitter-handles-3000-images-per-second.html)
* [The Infrastructure Behind Twitter: Scale](https://blog.twitter.com/2017/the-infrastructure-behind-twitter-scale)
* [Caching at Reddit](https://redditblog.com/2017/1/17/caching-at-reddit/)
* [How Uber Manages A Million Writes Per Second Using Mesos And Cassandra Across Multiple Datacenters](http://highscalability.com/blog/2016/9/28/how-uber-manages-a-million-writes-per-second-using-mesos-and.html)
* [Instant Messaging at LinkedIn: Scaling to Hundreds of Thousands of Persistent Connections on One Machine](https://engineering.linkedin.com/blog/2016/10/instant-messaging-at-linkedin--scaling-to-hundreds-of-thousands-)
* [Big Data in Real-Time at Twitter](https://www.slideshare.net/nkallen/q-con-3770885)
* <https://redditblog.com/2017/04/13/how-we-built-rplace/>