Audrey Long

JHU WWW Security Course Project

05/04/2021

Introduction Welcome to the Course Project!
The project is designed to merge many of the concepts we´ve learned about throughout the semester and give you the ability to implement them inside the Amazon AWS cloud. If you have not already, please register for the AWS Free Tier by creating an account here: https://aws.amazon.com/free/ The project can be completed entirely for free or at a very minimal cost (less than a textbook)!
Throughout the project you will create a DevOps pipeline that deploys a live, publicly available web application. Once completed, you will be responsible for securing the architecture. You will implement your live web page using two different methods: server based (EC2 VM) and serverless (Lambdas)!

A. Traditional Web Architecture with CI/CD
In class we talked about traditional web server architectures and how a commit to a Git repository can trigger an automatic build, test, and deployment of code. Complete this tutorial to get your application up and running on the web. You'll get to work with a number of AWS' CI/CD services in this exercise. As a guide, CodeCommit is your Git repository, EC2 is your Linux VM, CodeDeploy is the deployment tool for your application, and CodePipeline is your CI/CD tool that allows you to construct a workflow that automates the entire process.

Deliverables:
1. Provide a screenshot of each "major" step as you complete the tutorial. (unimportant steps do not need to be captured)
2. Complete a 1-2 page write-up describing each of the AWS services you used and the role each one plays in the CI/CD process.

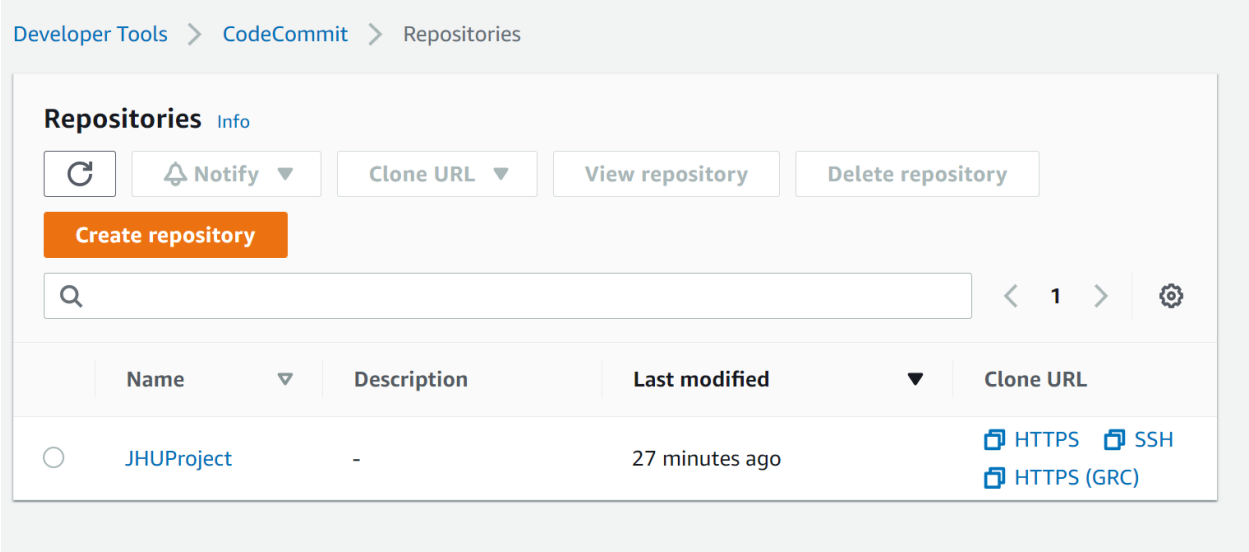

Figure 1: Create a CodeCommit repository

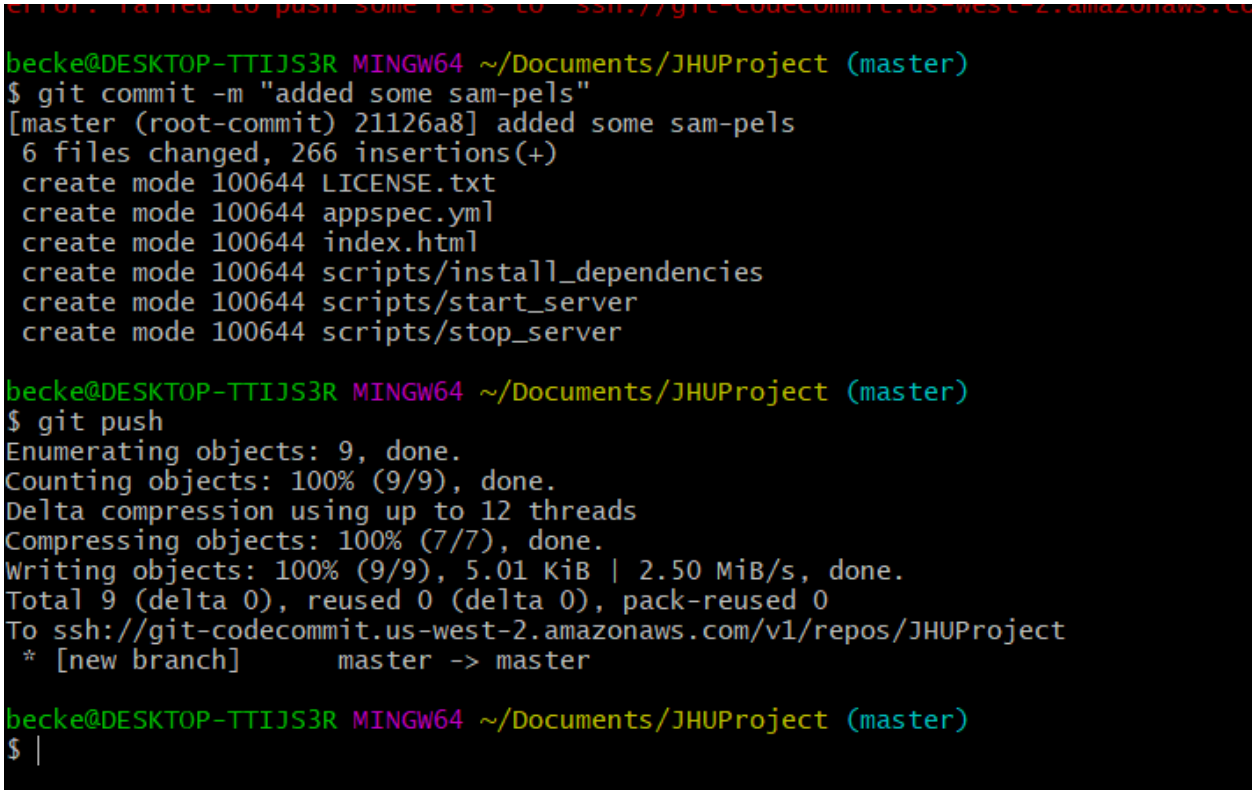

Figure 2: Add sample code to your CodeCommit repository

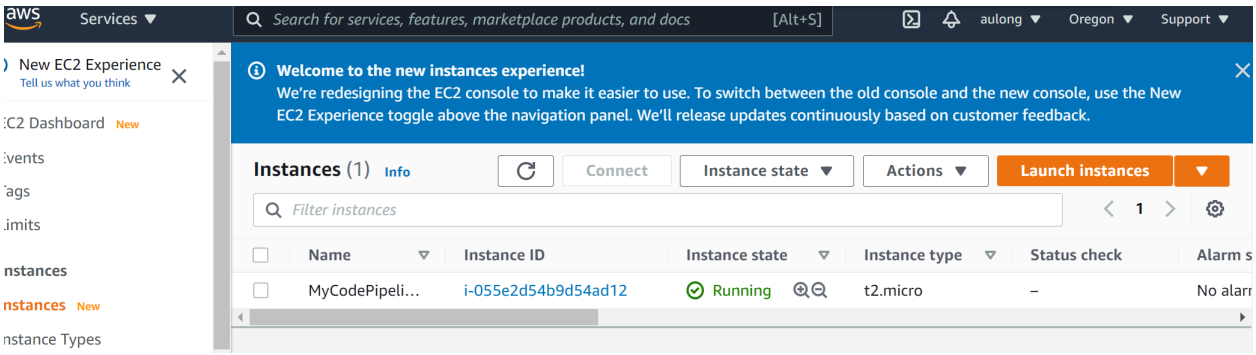

Figure 3: Create an EC2 Linux instance and install the CodeDeploy agent

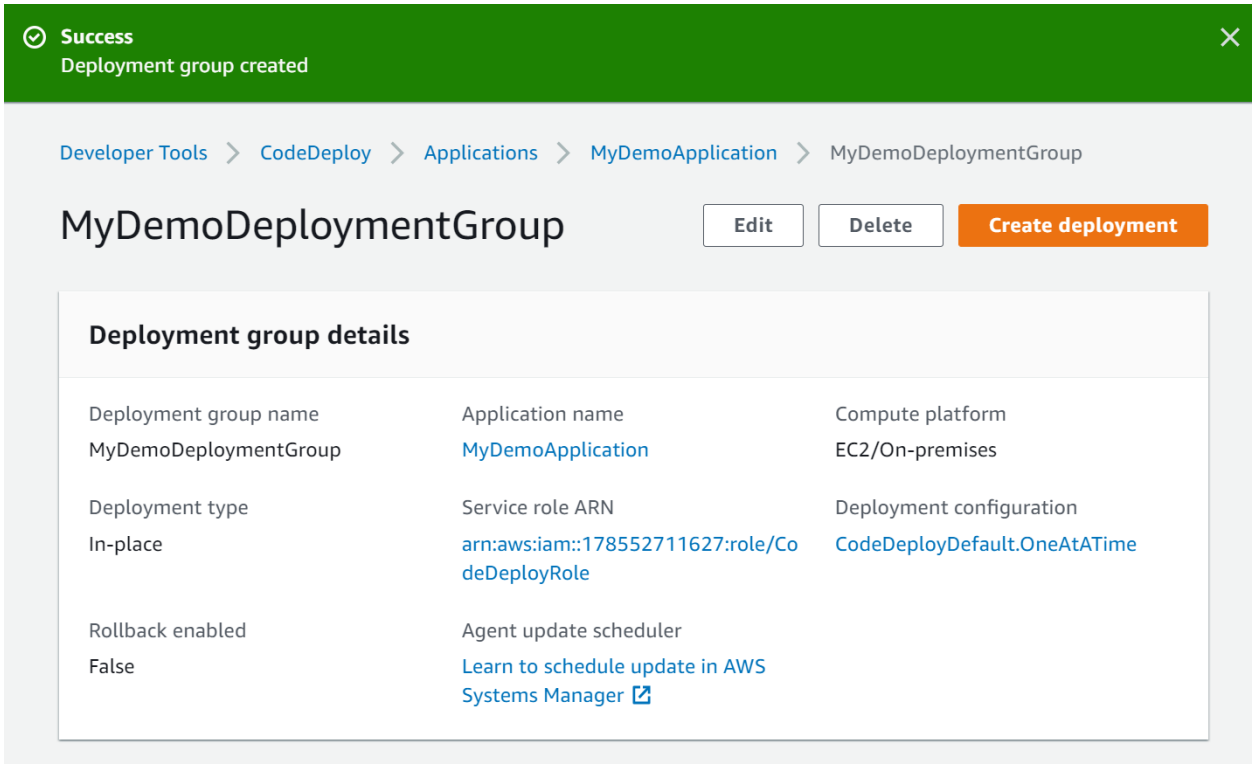

Figure 4: Create an application in CodeDeploy

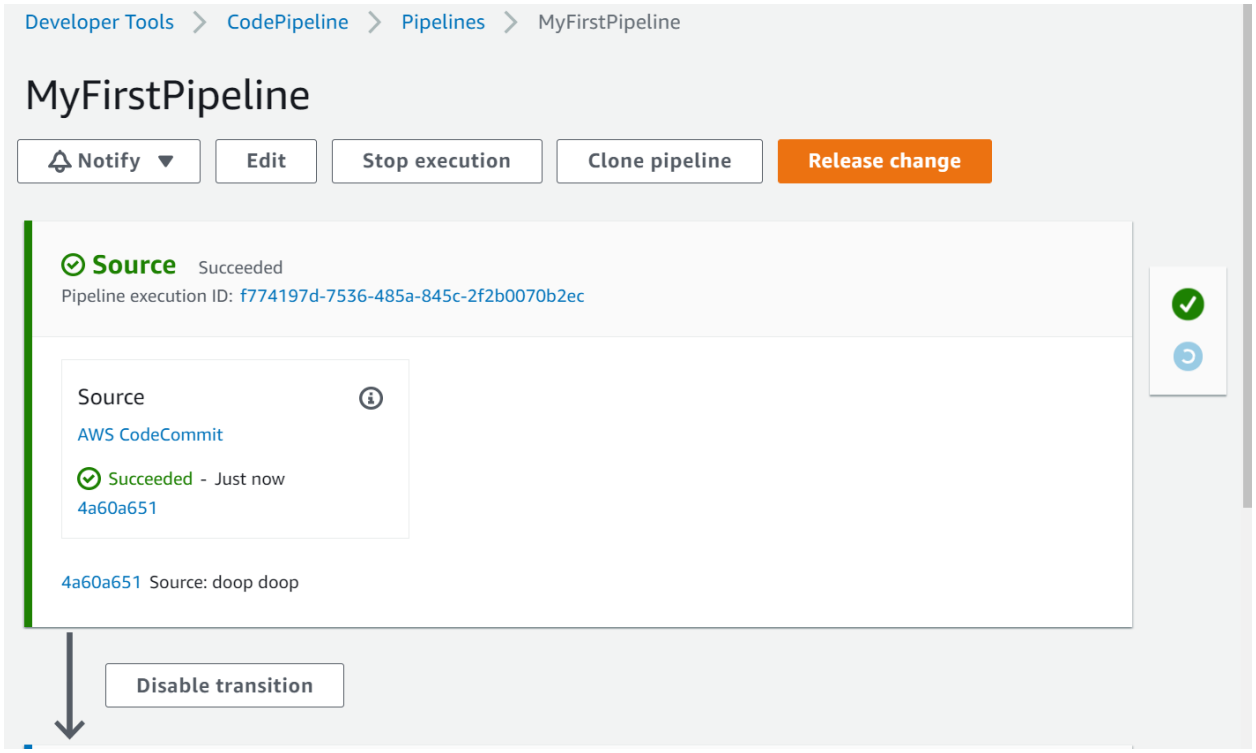

Figure 5: Create your first pipeline in CodePipeline

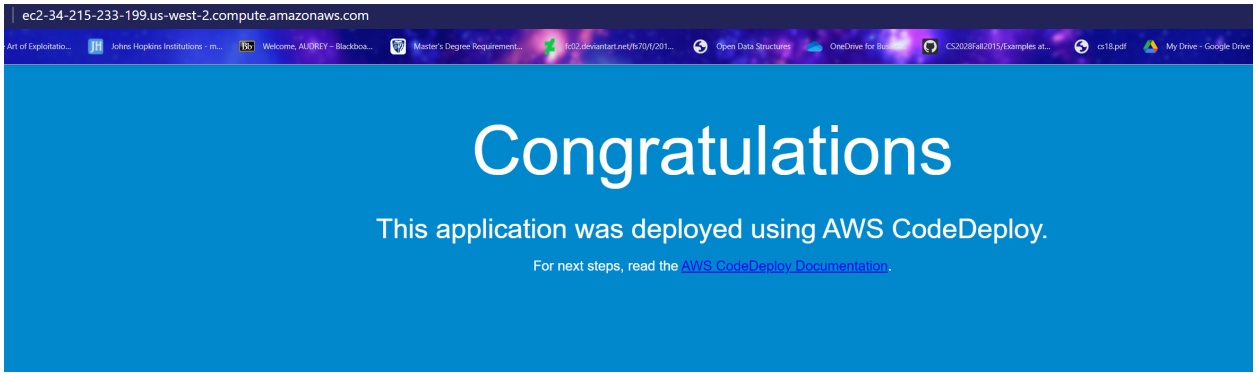

Figure 6: Deployed Application

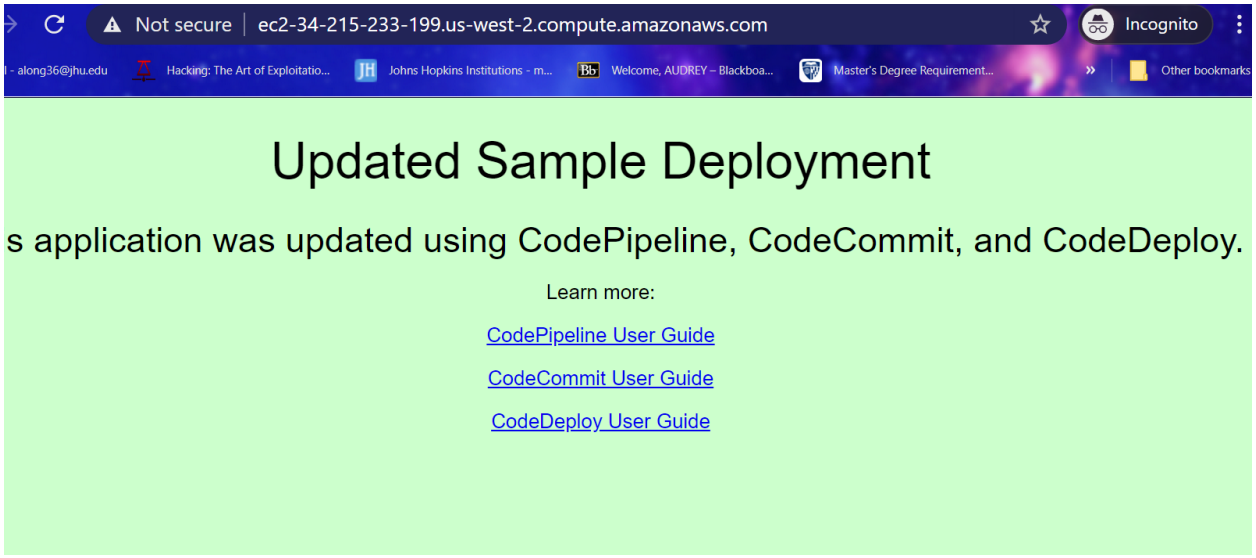

Figure 7: Modify code in your CodeCommit repository

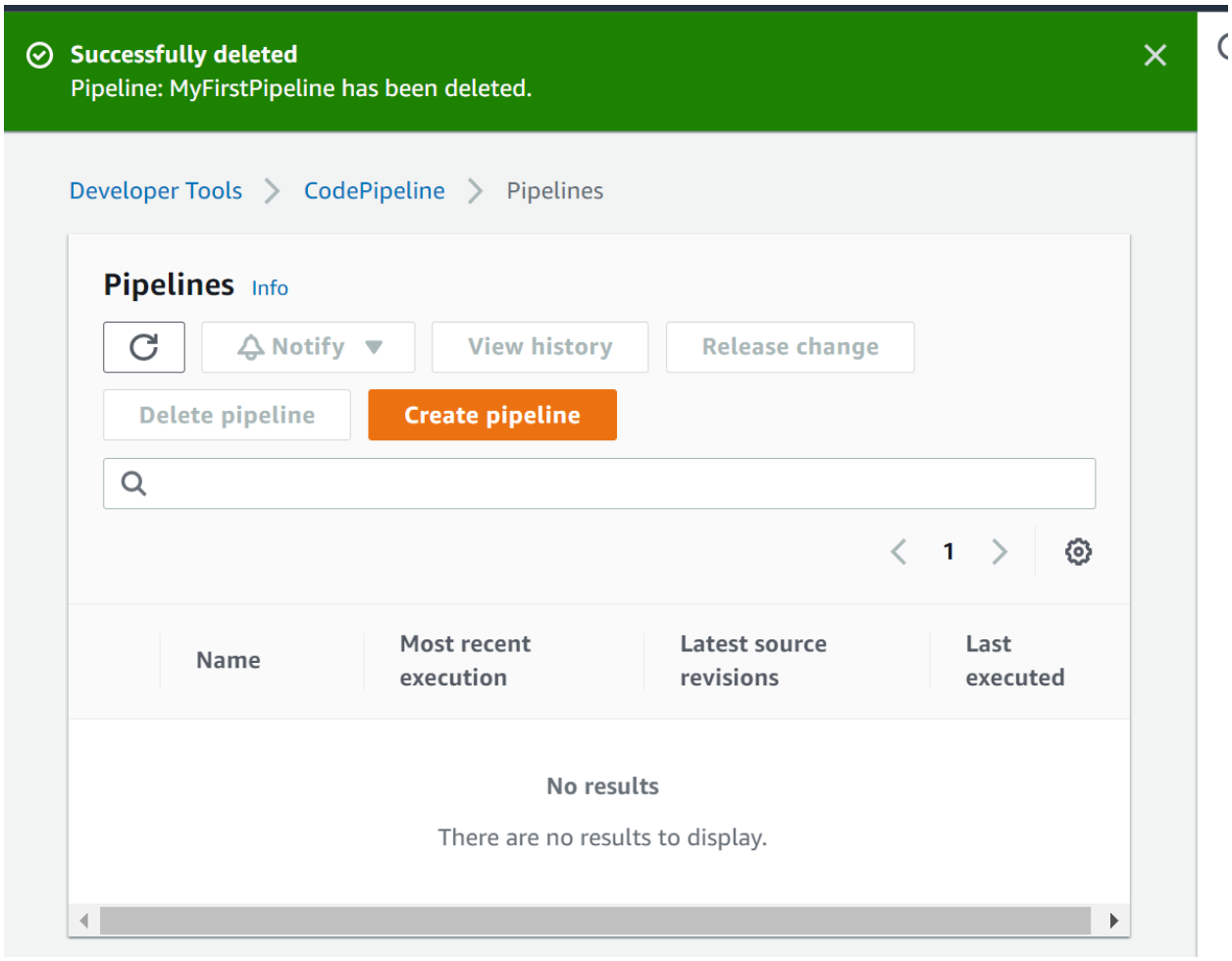

Figure 8: Clean up resources

The tutorial highlighted how to set up services and run a dev ops pipeline utilizing AWS services. The following services that were utilized during this tutorial include:

CodeCommit

This is the first service we got to play with when creating a fully functional deployment pipeline. This service lets you manage source control fully and host secure Git repositories. The role this service plays is grabbing the code that gets published after a developer pushes their code with a fully set up SSH instance connected to the code commit service. In order for this to function fully some integration needs to be done by the user to set up ssh keys, create an IaM user, and set up Git credentials in order to tie this service together.

IaM

As alluded to in the previous step, setting up an IaM user is a critical step to connect to the CodeCommit repository. This service allows the management of users and instances along with their permissions, policies, roles, groups, and key management associated with them. AWS IaM

enables managed access to all of AWS services. We need to create a user in order to access and utilize all of the services used within this pipeline.

EC2

After our code repository and user is all set up we are now able to connect to AWS services and push code into the repository, now we want to be able to utilize a deployment strategy with the newly committed code. EC2 is a web service that provides secure and resizable compute capacity in the cloud and is responsible for configuring resources to allow the application to run on a given compute environment.

CodeDeploy

This service is a software package and is responsible for deploying the code located in the codecommit repository and configuring it to the EC2 instance to deploy in a browser with little to no down time. A codedeploy agent needs to be configured first, then a codedeploy application needs to be made which is a resource containing the software application.

CodePipeline

This service is responsible for automatic event actions that get kicked once a pull request comes in. Once an event comes in the pipeline is responsible for chaining together all of the steps in order to then (in this case) deploy that code The pipeline will then kick code deploy to then deploy the ec2 instance and make the application live.

B. Serverless Web Architecture

We also briefly discussed “serverless” computing in class and saw an example of how an AWS Lambda function spins up a container to run a snippet of code then wipes out the container once the function completes. Complete the AWS tutorial for creating a serverless web architecture found here.

Figure 9: Host a static website

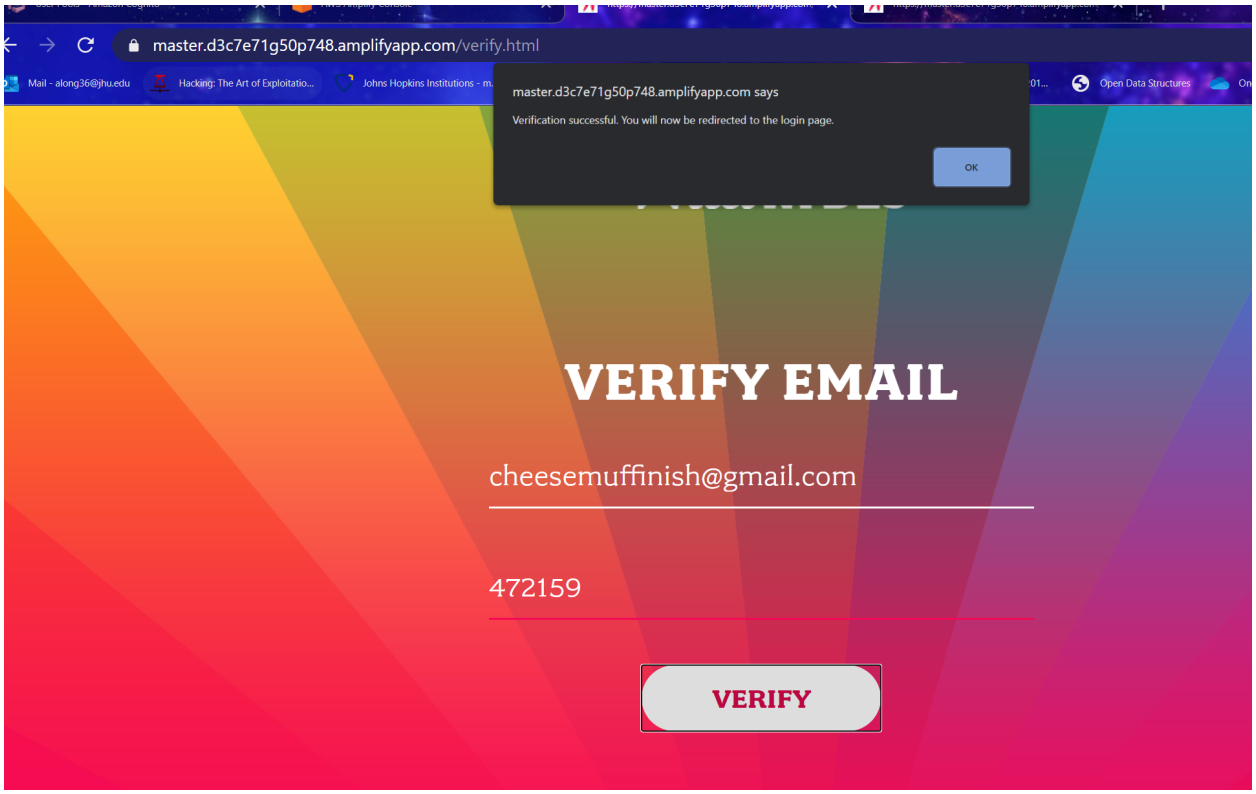

Figure 10: Manage Users

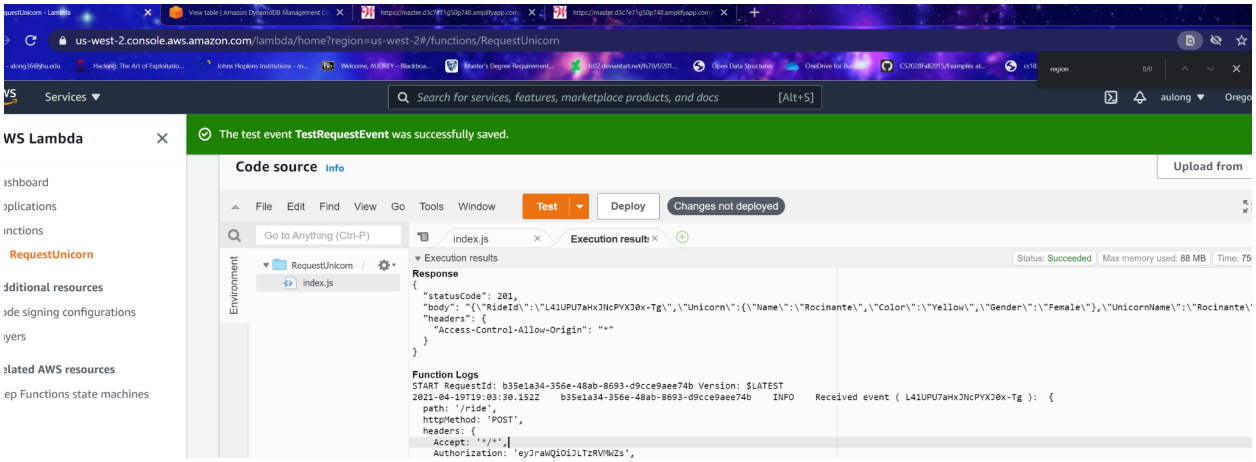

Figure 11: Build a Serverless Backend

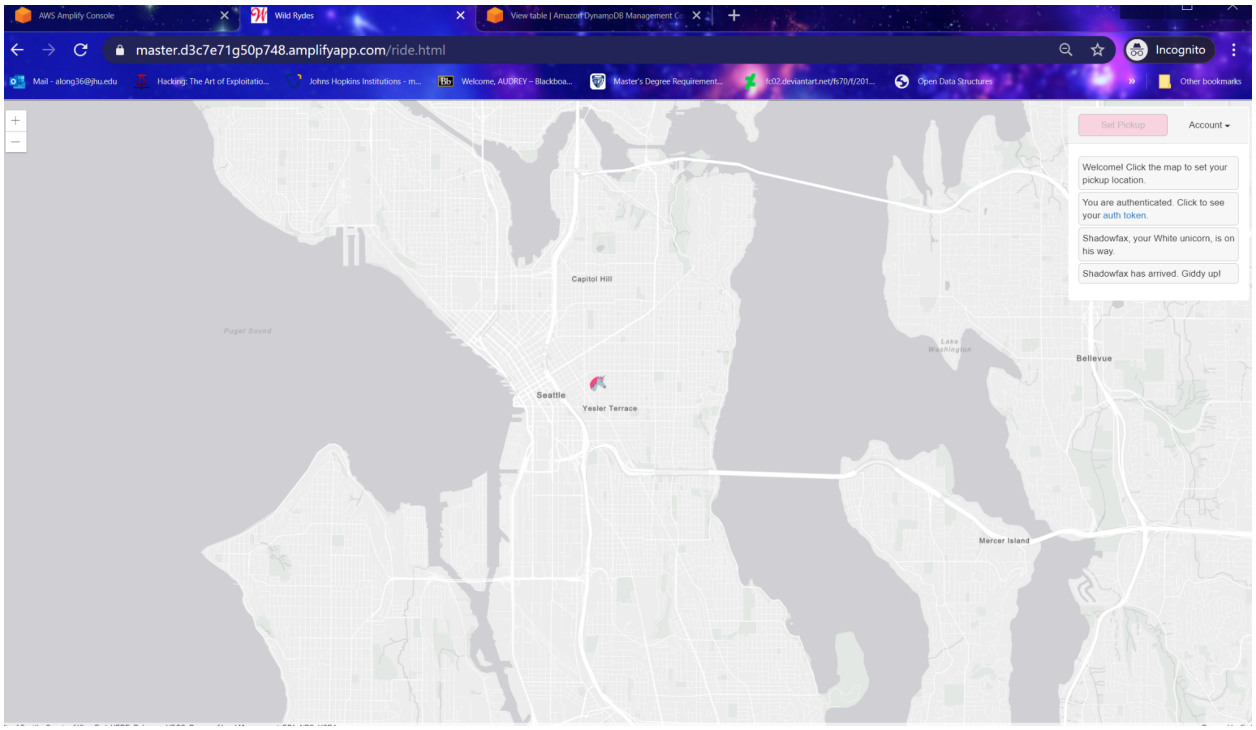

Figure 12: Deploy A Restful API

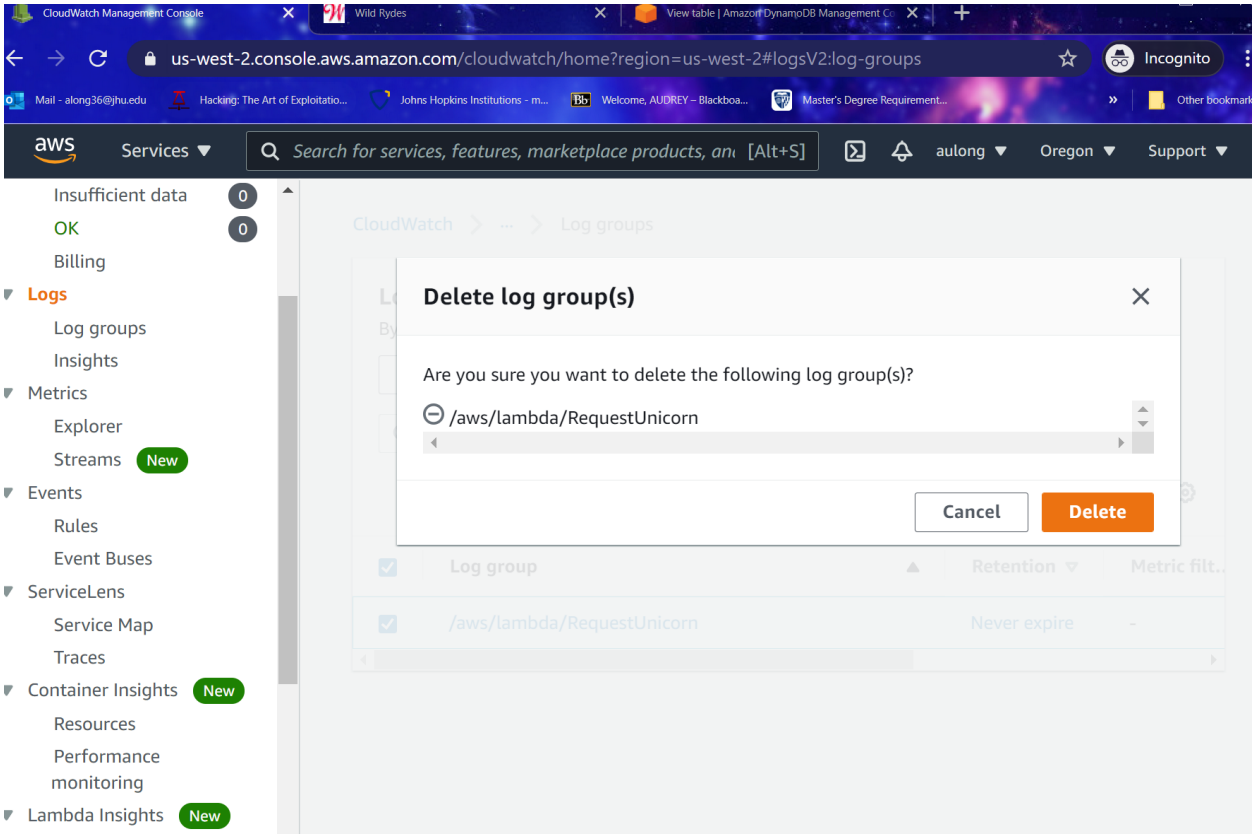

Figure 13: Terminate Resources

The tutorial highlighted how to set up services and run a serverless web application utilizing AWS services. The following services that were utilized during this tutorial include:

IaM

Setting up an IaM user is a critical step to connect to the CodeCommit repository, permission to write to the DynamoDB, and every Lambda Function has an IAM role associated with it. This service allows the management of users and instances along with their permissions, policies, roles, groups, and key management associated with them. AWS IaM enables managed access to all of AWS services. We need to create a user in order to access and utilize all of the services used within this pipeline. In this lab the IAM user grants the Lambda function permission to write logs to Amazon CloudWatch and writes to the DynamoDB table.

Amazon CloudWatch

Amazon CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources. With CloudWatch, you can collect and access all your performance and operational data in form of logs and metrics from a single platform. [9] This service is used for Lambda functions to write logs into this service.

Deliverables:
1. Provide screenshots of each “major” step as you complete the tutorial. (unimportant steps do not need to be captured)
2. Please provide a 1–2-page write-up detailing what each AWS service you used and what they provide to the CI/CD process.

tip: when you get to this part, there will be a link to download the requestUnicorn.js code from GitHub (Module 3, step 3, part i). their link is outdated. here is the new link: https://github.com/aws-samples/aws-serverless-webappworkshop/blob/main/workshop/content/ServerlessBackend/Lambda/requestUnicorn.js

C. Securing Your Web Architecture
Explain in detail with respect to your architecture, how you would achieve the following goals. For this portion of the project there is no technical implementation necessary. Your response to this section should be approximately 4-5 pages in length to cover the topics properly. Do not exceed 10 pages please. Try to cover as much as possible and be creative!

1. Include the CI4A (CIA, AAA Triads): Your web architecture is up and running, now it is time to think about how to secure it. Recall the main goals of information security: confidentiality, integrity, availability, authentication, authorization, and accounting.

Web security architecture has come a long way in regards to providing adequate configuration options to ensure that each web service component follows CI4A and web security best practice baselines. A lot of the services contain some very bare bones amounts of security, but each component allows the users to configure each web component to their technical and project needs. Looking at CIA holistically can be very ideal in regards to ensuring you are taking precautions with safeguarding user data; Confidentiality safeguards data by only allowing authorized users to gain entry to this data, Integrity ensures that the data payload has not been compromised throughout the transitionally process from source to destination, Availability refers to services and architecture maintaining health which ensure that users can always access their data.

Amazon uses the CIA triad to heart throughout all of their architecture and web services and ties them together in a centralized location. AWS provides IaaS, SaaS, and PaaS services and we used many of these services above when generating the serverless and server based pipelines. Some of the main components for each pipeline stems from similar architectural constructs security throughout the system including, but not limited to storage/database services, computing services, networking services, identity access management (IAM) services, and observability services.

All of the services in AWS include very robust security controls that the customer can enable if desired, for example IAM can be used as a centralized access services for all of aws services with role based access controls (RBAC), managed identities, policies, password management, and numerous identity solutions to cater to the wide array of authz and authn permutations.

Confidentiality is a mechanism which prevents adversaries from obtaining sensitive information from our system. The IAM service is very centralized and required to be configured for all of the services used within the pipeline tutorials to configure users, policies, and permission setup as a required step, this alone ensures confidentiality and availability using the least privilege model of access control since this concept is based around preventing adversaries from obtaining sensitive data such as credentials to obtains further system access.

Integrity ensures that the information within the system is not tampered with throughout the transitional and resting phase when it enters and leaves the system, as well as accounting for every hop within the system. Services that were used above such as Amazon S3 classifies the data being contained with a tagging system which adds granularity within Amazon services along with metadata containing modification timestamps, content length checking, and MD5 payloads which can guarantee that the payload will be untampered with throughout the data flow of the system.

Availability refers to the accessibility and usability of the data and services throughout the data system, if the system happens to go down or the data is not available whenever the user needs to access the data, it can be just as a security concern as having an adversary interfere with the payload itself; this construct in aws is being protect by DDoS protection mechanisms throughout the system to ensure that an adversary will have a hard time interfering with the architecture, flooding IP addresses, and preventing users from logging into their services. An example of availability within the AWS architecture that's been used above can be seen within the lambda functionality. By default lambda is built around aws regions and availability zones which are connected with low latency, high throughput, and redundant networking mechanics. The key to constant availability within the lambda architecture is the fact that the availability zones are fault tolerant, widely available,and have multiple instances to ensure that even if an outage occurred your access will be rerouted to a fully functioning instance.

2. Research Service Offerings: AWS provides a lot of functionality for achieving these goals, so be sure to research service offerings and leverage things like S3 encryption, Cognito user management, CloudTrail for logging, and many others.

Some service offerings that AWS provides a lot of security in their services they provide, most security features are turned on by default, but most need to be configured and implemented to get the full range of security in your project. Amazon adopts a shared responsibility model which is defined as “Security and Compliance is a shared responsibility between AWS and the customer. This shared model can help relieve the customer’s operational burden as AWS operates, manages and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates. The customer assumes responsibility and management of the guest operating system (including updates and security patches), other associated application software as well as the configuration of the AWS provided security group firewall. Customers should carefully consider the services they choose as their responsibilities vary depending on the services used, the integration of those services into their IT environment, and applicable laws and regulations. The nature of this shared responsibility also provides the flexibility and customer control that permits

the deployment. As shown in the chart below, this differentiation of responsibility is commonly referred to as Security “of” the Cloud versus Security “in” the Cloud.” [14]

Serverless web technologies come with many configurations and services that can be utilized to secure pipelines and architectures.Taking a look into some security services that i will be using include VPC private endpoint configuration with Amazon API gateway in order to lower my attack surface and lock down any endpoints to the lambdas functions. Everything can be accessed with Amazon IaM with the proper user policies, least privilege model adopted, and access controls set within each amazon service being used. Web application users can also be configured with aws Cognito to ensure the web application access is also locked down.

Logging and monitoring need to be implemented to ensure that critical errors and networking monitoring is enabled within the architecture and can be implemented with CloudWatch which can track and alert on any malicious or unsatisfactory event monitoring within the system. Cloudwatch can also be fed into a SIEM solution to get intrusion detection and packet monitoring along with a set of robust network rules and event triggering.

Ensure the latest security controls as well as updated protocols are being used in the system will ensure that adversaries cant exploit any underlying issues in the architecture is something that can be accomplished easily, enabling TLS 1.2 and the latest SSL can prevent data leaks and encryption in transit throughout the HTTP , as well as architectural hops.

FInally i would add the addition of a web application firewall (WAF) to my security solution. AWS WAF can protect my web application by only allowing certain APIS and endpoints from being hit, create security traffic rules, and run with customized and pre configured security rules to mitigate the OWASP top 10 security risks.

3. Describe Implementation: In addition to which services you would choose, please describe how you would implement them in a fair amount of detail.

I personally am interested in the serverless architecture and i believe that is where the future of cloud computing is heading, and with that in mind i will highlight some security features below that I would further implement into my secure serverless pipeline.

The heart of the serverless pipeline lies with AWS Lambda functionality and some things we can do to ensure security best practices would be the following: Ensure there exists one IAM role per lambda function, having the architecture as segmented as possible can prevent leaks and exploitation if an adversary happens to slip through the cracks. IAM can also play a pivotal role by deploying access control policies to the function along with generating endpoints and entry points. Lambda functions are built on an availability node network to ensure that availability is almost always ensured when using these services in your architecture. Code management within the lambda function is going to be the biggest security factor in this architecture. Secure coding practices will be crucial to ensure that input sanitization, well prepared algorithms, and good coding practices are utilized. This is the backbone of the pipeline and would need to be secure to act as the database which drives the code.

Amazon DynamoDB is a table that was used in our application to maintain a rideId. In DynamoDB, tables, items, and attributes are the core components that you work with. This is the underlying data structure that is utilized in serverless architecture and therefore needs to be locked down from adversaries. Encryption at rest should be enabled in this data state to ensure

if a breach occurred, that the data would be safeguarded. IaM policies and authorization can be set up so only trusted users can access this data in the database. Client side encryption can be utilized to encrypt sensitive data before its being sent to this service using the Amazon DynamoDB Encryption Client; which is an item encryptor that encrypts, signs, verifies, and decrypts table items. It takes in information about your table items and instructions about which items to encrypt and sign. It gets the encryption materials and instructions on how to use them from a cryptographic material provider that you select and configure. [18]

Amazon API Gateway is a service used to create a RESTful API that will expose an HTTP endpoint that can be invoked from customers' browsers and with that in mind needs to have very implicit security controls locked down to ensure only the traffic we are interested in comes into our system. This service can also be tied in with IaM for user access and resource policies, logging and monitoring. Cloud watch logs can be implemented to ensure we are gathering all of the critical logs and set up alerting if certain thresholds get crossed. API Gateway can also authorize lambda users using bearer token authentication to ensure the users invoking the requests are valid within the architecture. Another security feature that can be implemented is creating private endpoints which can only be accessed within a Virtual Private Cloud (VPC) which can also be conjured with resource policies to allow and deny access from selected VPCs. Depending on the scenario AWS direct connect can be utilized to establish on premises networks into VPCs to access the private endpoint.

AWS IaM is the identity solution used in Amazon services that are responsible for role based access, identity management solutions, policy controls, and is a centralized security solution for the whole suite of AWS architecture. IaM is a solution to achieve authorization, authentication, and confidentiality in our security model by creating adopting the strategy of least privilege and generating the roles and groups needed only for the purpose that user would need to utilize in the system along with policy setting for certificates and passwords, and safeguarding unauthorized users from entering the system. For the pipeline we already generated basic user policies, but ensuring that every service being used in the architecture belonging into user groups and set access policies is going to be crucial to ensuring all of our identity solutions will be locked down. For the user pool to be more secure I would enable multi factor authentication to be set for the users in order to protect their aws identities.

IaM is a wonder security solution to all services but we can also utilize AWS Cognito for the web application users. The user pools model is a great tool to ensure access control policies for users are set and enabled. AWS cognito will manage all of the users accessing the deployed web application and needs to be configured with very secure policies and user pools.

CloudWatch is the observability piece that is used to monitor and provide actionable insights into the system as a whole by collecting log data from all the services and presenting the data in one centralized location. With this service a user can customize the amount of log data and maintain specific endpoints for interesting data. Another service that can be used with this service is EventBridge which can manage cloudwatch events and provide even more granular security events and cloudwatch data itself can be fed into SIEM systems to utilize further security solutions. Cloudwatch can give me data and logs from my application deployment and integration pieces to let me know if a problem occurred or if anything suspicious is afoot in my architecture,

Aws code commit is the pipeline muscle which commits and allows users to push code from a repository such as GitHub and deploys those changes in the cloud. Clearly this is important because we don't want to give just anyone permission to push and deploy code into a live environment and want to preserve data integrity, along with the availability of the services and data to users. Since a lot of this traffic is web based I will ensure that the latest TLS and SSL protocols are up to date and utilized for the data encryption when it hops around the internet. Another way to ensure data privacy and security is to implement Amazon Macie which is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS. [17] The Macie service is an extra layers to ensure data classification and protection mechanisms are in place to ensure that data integrity and flow are safeguarded from adversaries.

Another thing i would implement is AWS WAF into my security solution, i would protect my deployed web application and configured only the private endpoints that were generated from locking down the lambda functions.the improved web traffic visibility from the waf will also integrate nicely with the cloudwatch service that was already mentioned above and generate rules to from my WAF into Cloudwatch, this will provide a safer environment for my web application and the data hosted on it. Web request filtering can also be enabled on the WAF and should be done to further lock down the service from adversaries.

4. Include Other Topics from the Course: Consider aspects from all areas we have touched upon this semester, that would apply in securing your application.

Static code analysis, credential scanning, and code coverage should at least have a bare minimum impact in being gating production quality code. Earlier in the semester we implemented and used SonarQube to demonstrate the power this tool brings to the table in regards to sniffing out vulnerabilities, potential bugs, unit testing code coverage quality, and showing potential exploits with the code smell functionality. I would implement SonarCloud into my serverless architecture pipeline to ensure that during the CI/CD process I am continuously pushing up production ready code into the main baseline and implementing a gate to halt the code if the quality threshold is not met.

Throughout this semester we learned alot about input validation and ensuring the adequate checking is happening in the functions we are creating so an adversary can not exploit the codebase any further from trying to send in malicious payloads without proper data sanitization techniques.

Data classification is another very important tool we can use in this scenario to properly mark the data entering and exiting the system along with the hops the data takes to ensure we are properly safeguarding the data within the systems and turning on additional security controls depending on the sensitivity and transitional phases. Data flow diagrams and threat models are tools which can help us factor in risk and data risk within the system, this will truly be a guideline to fully understand and implement the proper controls depending on the customer’s desires and security concerns with any cloud based architecture, the attack surface is large and must be contained.

HINTS: The following are just some tips and ideas to consider touching on in your report:

- Focus on the CI4A while progressing through the course project
- Take note of areas that provide for valuable monitoring/logging opportunities
- Would stricter access controls provide a more-cost effective way for implementation?
- We have discussed server-side vs. client-side encryption, which aspects/services apply best to this infrastructure?
- Access control strategy – how will you handle user management/access, database access, developer access to the pipeline, etc.?
- How can you leverage network/VPC security to help supplement encryption and access controls?
- How might you utilize your infrastructure setup and services to help protect against DoS attacks?
- How will you store your data, might there be any scenario for archiving?
- How will data storage play a role in the availability and cost of your application?

NOTE: Remember to tear down your infrastructure after you get your final grade, so you don't get charged. Remember to always list your references.

Due dates: You are encouraged to submit each section as you complete them. Please start the project early.

Suggested completion dates (soft due dates) will be posted on blackboard and SLACK Project channel. The hard due date is the date the final exam is due.

Progressive grading: If you submit the section of your project on or before the suggested completion date, we will review providing feedback. If there are any areas that you can improve in, you can make the adds/edits, then resubmit with the next section to credit.

Grading Rubric for this assignment

Assessment	Points Given
A. Traditional Web Architecture with CI/CD i. Tutorial completion documented major steps (5) ii. Write-up describing each AWS service used with focus on roles in CI/DC process. (15)	
B. Serverless Web Architecture i. Tutorial completion documented major steps (5) ii. Write-up describing each AWS service used with focus on roles in CI/DC process. (15)	
C. Securing Your Web Architecture Write-up describing each AWS service used with focus on roles in CI/DC process. (50)	
Student's submission was a professional presentation with spelling and grammar at master's level (10)	
TOTAL (100)	

References

https://aws.amazon.com/codecommit/ [1]

https://aws.amazon.com/iam/#:~:text=AWS%20Identity%20and%20Access%20Management%20(IAM)%20enables%20you%20to%20manage,offered%20at%20no%20additional%20charge[2]

https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc [3]

https://aws.amazon.com/cognito/ [4]

https://aws.amazon.com/amplify/ [5]

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html [6]

https://docs.aws.amazon.com/lambda/latest/dg/welcome.html [7]

https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html [8]

https://aws.amazon.com/cloudwatch/features/#:~:text=Amazon%20CloudWatch%20is%20a%20monitoring,metrics%20from%20a%20single%20platform. [9]

https://n2ws.com/blog/aws-cloud-security-compliance/aws-security-cloud-best-practices [10]

http://haleybush.cikeys.com/uncategorized/the-cia-triad-and-amazon-web-services/ [11]

https://docs.aws.amazon.com/whitepapers/latest/serverless-architectures-lambda/security-best-practices.html [12]

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/security.html [13]

https://aws.amazon.com/compliance/shared-responsibility-model/ [14]

https://docs.aws.amazon.com/apigateway/latest/developerguide/security.html [15]

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-private-apis.html [16]

https://aws.amazon.com/macie/ [17]

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/best-practices-security-preventative.html [18]