



CS 470 PROJECT TWO

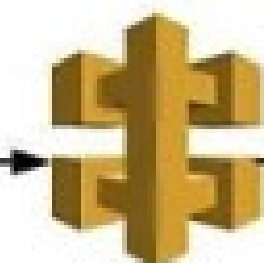
FULL STACK DEVELOPMENT II

CONFERENCE PRESENTATION: CLOUD DEVELOPMENT

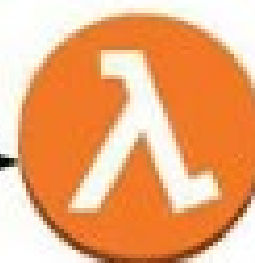
AWS



Amazon S3



Amazon
API



Lambda
function



Amazon
DynamoDB



neila Mawhir
December 9,



AGENDA

- **Containerization and Orchestration**
- **The Serverless Cloud**
- **Cloud-Based Development Principles**
- **Securing Your Cloud Application**





MIGRATION FRAMEWORK

Containers

Images

Volumes

Builds

Docker Scout

Extensions

Containers [Give feedback](#)

Container CPU usage ⓘ
0.53% / 800% (8 CPUs available)

Container memory usage ⓘ
713.7MB

☐ Only show running containers

<input type="checkbox"/>		Name	Container ID	Image	Port(s)
<input checked="" type="checkbox"/>	●	mongodb	1f258ce507c3	mongo	27017:27017 Show all ports
<input type="checkbox"/>	○	bold_sammet	f3e64f03f35e	node-lafs-w	4200:4200
<input type="checkbox"/>	●	gracious_fermi	0e282ecb4f9c	node-lafs-a	3000:3000





CONTAINERIZATION AND ORCHESTRATION

TOOLS FOR CONTAINERIZATION

Containers

Images

Volumes

Builds

Docker Scout

Extensions

< Images [Give feedback](#)

Local

Hub

2.24 GB / 855.24 MB in use 3 images

Search

	Name	Tag	Image ID
<input type="checkbox"/>	<input checked="" type="checkbox"/> mongo	latest	77c59b638412
<input type="checkbox"/>	<input checked="" type="checkbox"/> node-lafs-web	latest	0ca22c14a2e4
<input type="checkbox"/>	<input checked="" type="checkbox"/> node-lafs-api	latest	1a2096ae05d3



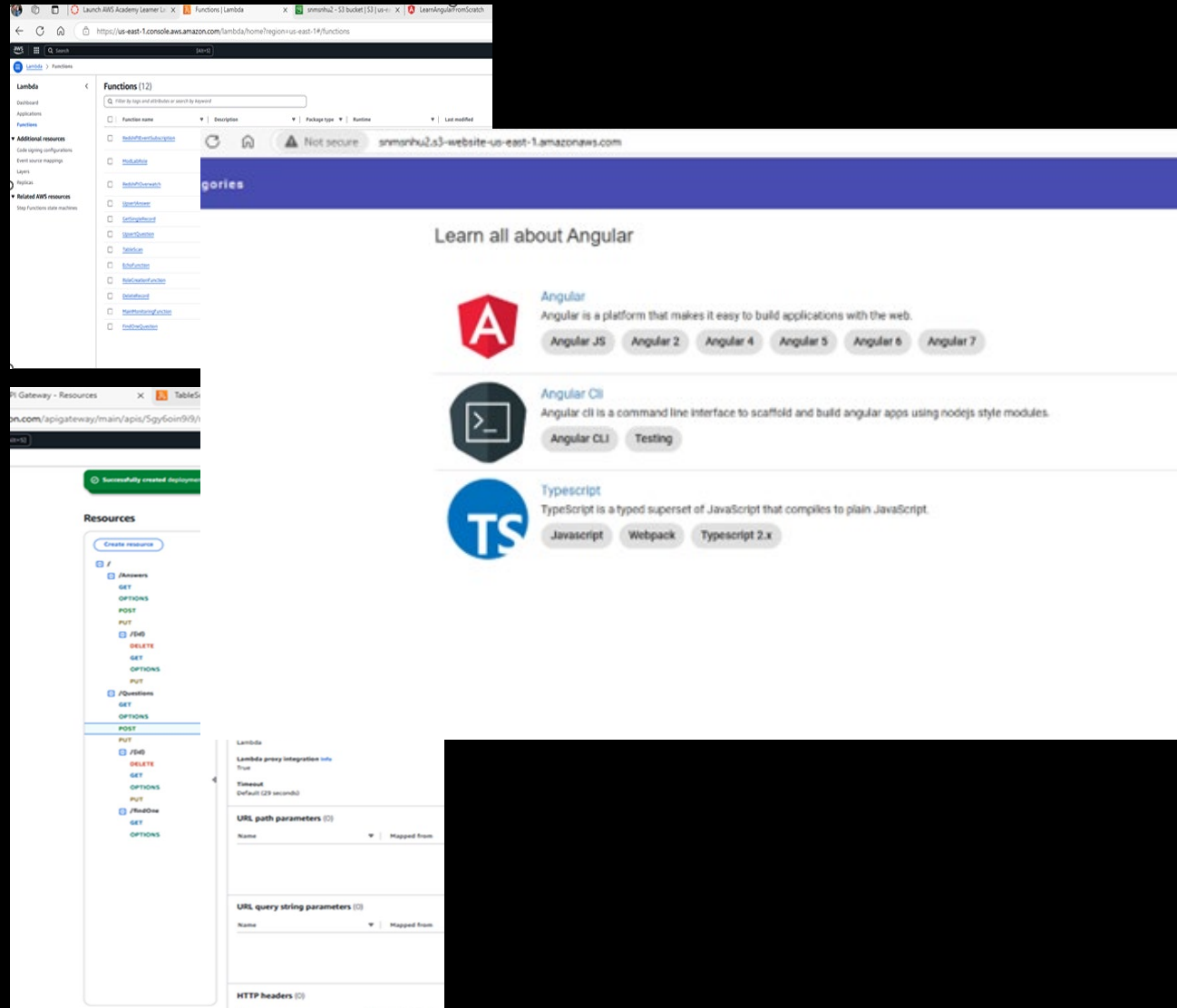


CONTAINERIZATION AND ORCHESTRATION

	C: > Users > cobra > lafs-api > docker-compose.yml
✓ LAFS-WEB	1 version: '3.7'
> .firebase	2 services:
> dist	3 # REST API running on Node JS container
> e2e	4 app:
> learn-angular-from-scratch-ste...	5 container_name: lafs-api
> node_modules	6 restart: always
> sdk	7 build: .
✓ src	8 ports:
> app	9 - '3000:3000'
> assets	10 # link this container to the Mongo DB container
✓ environments	11 links:
TS environment.prod.ts M	12 - mongo
TS environment.ts M	13 # pass in environment variables for database host and name
≡ browserslist	14 environment:
★ favicon.ico	15 - DB_HOST=mongo
<> index.html	16 - DB_NAME=lafs-db
K karma.conf.js	17 # Mongo DB storage container
TS main.ts M	18 mongo:
TS polyfills.ts	19 container_name: lafs-db
🔗 styles.scss	20 image: 'mongo:4'
TS test.ts	21 ports:
{ tsconfig.app.json	22 - '27017:27017'
{ tsconfig.spec.json	23 # Attach the external network to these containers
{ tslint.json	24 networks:
🔗 .dockerignore U	25 default:
⚙ .editorconfig	26 external:
🔥 firebase	27 name: lafs-net
	28



THE SERVERLESS CLOUD



Advantages:

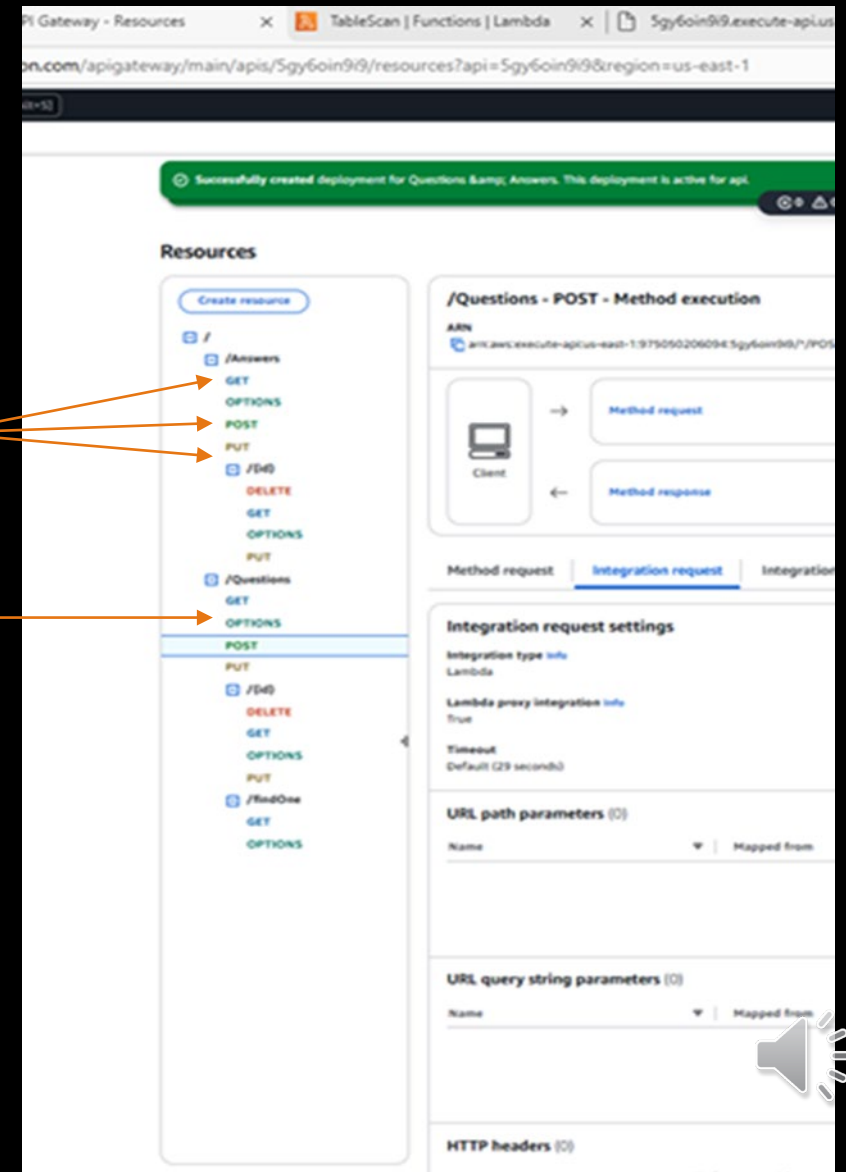
- ✓ Pay-as-you-go
- ✓ Cost Savings
- ✓ Scalability
- ✓ Flexibility
- ✓ Accuracy
- ✓ Speed
- ✓ Reduce overhead



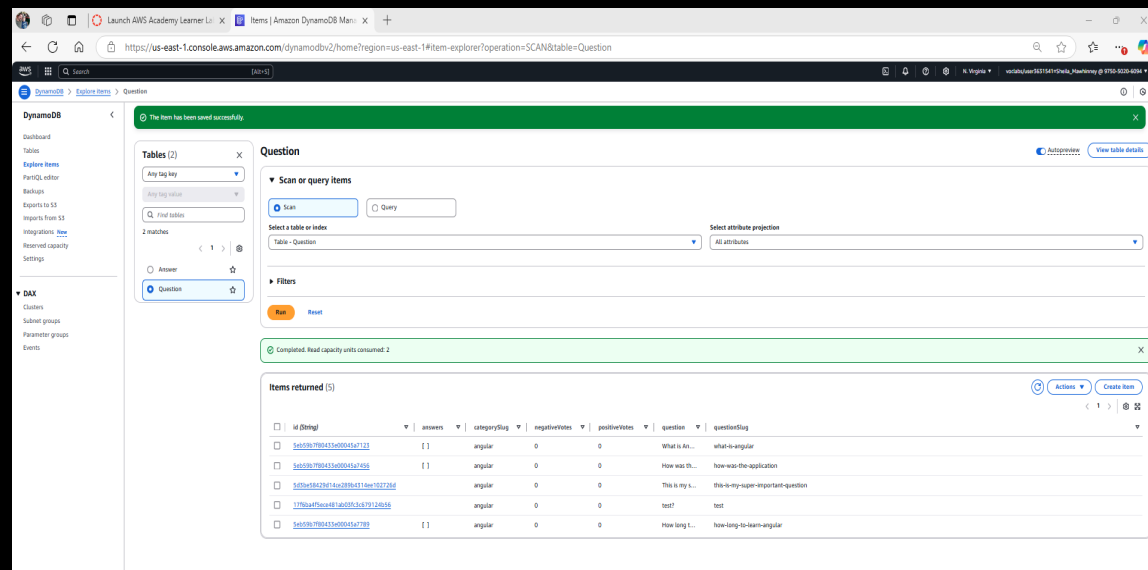
API GATEWAY LAMBDA

GET, POST, PUT

CORS - Options

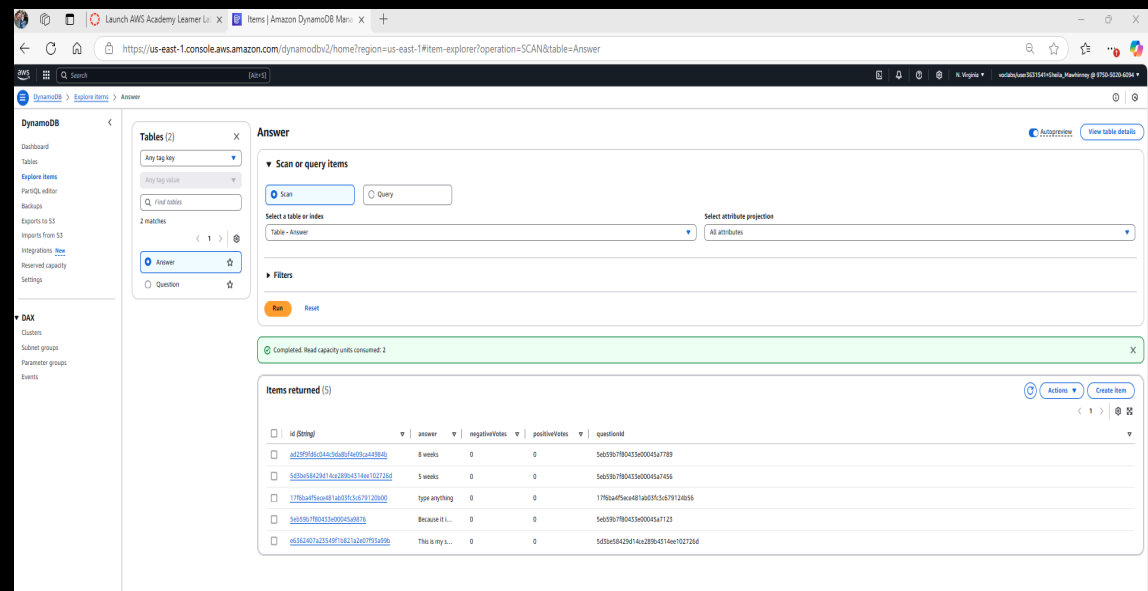


QUERIES AND SCRIPTS ARE PERFORMED ON DYNAMODB AND MONGODB



The screenshot shows the AWS DynamoDB console interface. The left sidebar contains navigation links for Dashboard, Tables, Explore Items, PartiQL editor, Backups, Exports to S3, Imports from S3, Integrations, and Settings. The main content area displays the 'Question' table. A 'Scan or query items' section is active, showing a successful scan operation. A green banner at the top indicates 'The item has been saved successfully'. Below the scan results, a table lists the items returned, including columns for id, answers, category, negative votes, positive votes, question, and questionId.

id (string)	answers	category (string)	negative votes	positive votes	question	questionId
5e659b78b435a00045a7123	[]	angular	0	0	What is An...	what-is-angular
5e659b78b435a00045a7456	[]	angular	0	0	How was th...	how-was-the-application
5e659b78b435a00045a7264	[]	angular	0	0	This is my ...	this-is-my-super-important-question
1770ba4f5eac481a00055a79124056	[]	angular	0	0	test?	test
5e659b78b435a00045a7789	[]	angular	0	0	How long L...	how-long-to-learn-angular



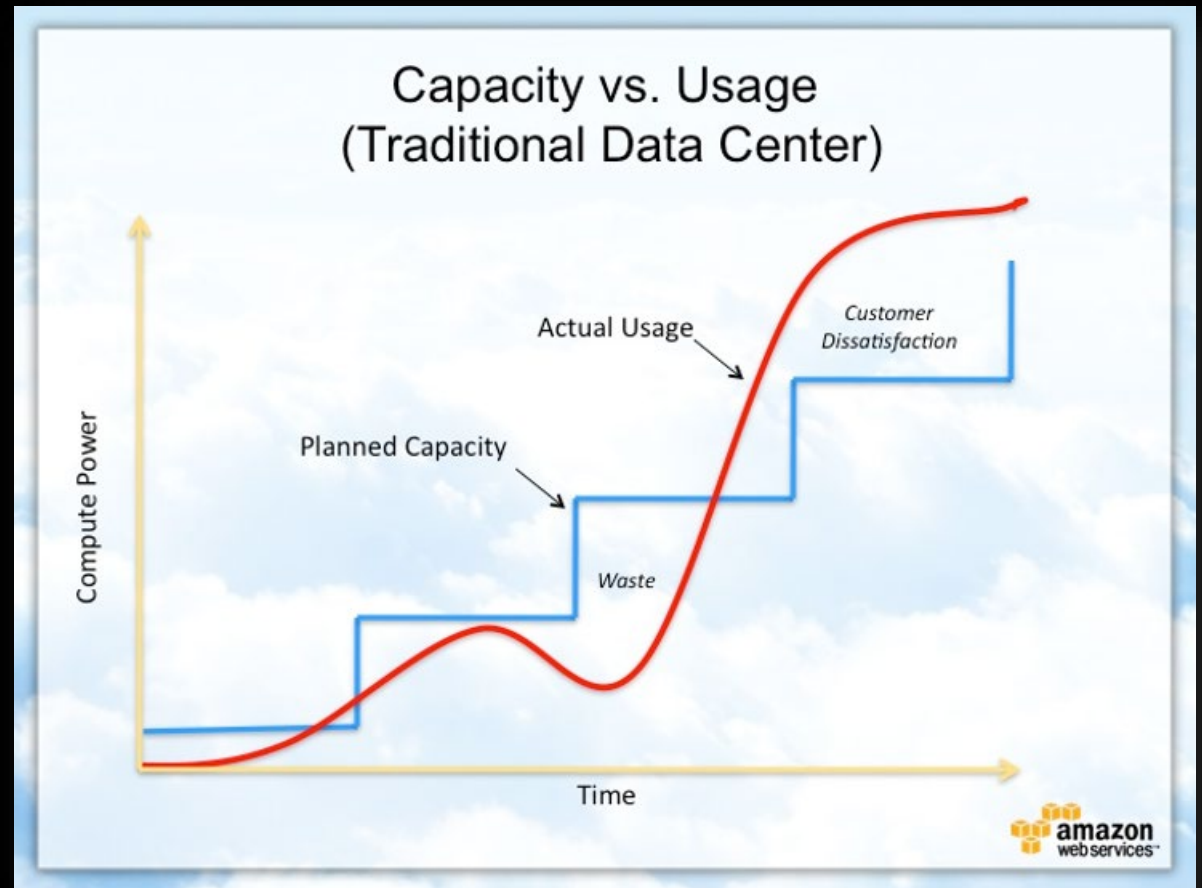
The screenshot shows the AWS DynamoDB console interface for the 'Answer' table. The left sidebar is the same as the previous screenshot. The main content area displays the 'Answer' table. A 'Scan or query items' section is active, showing a successful scan operation. A green banner at the top indicates 'Completed. Read capacity units consumed: 2'. Below the scan results, a table lists the items returned, including columns for id, answer, negative votes, positive votes, and questionId.

id (string)	answer	negative votes	positive votes	questionId
4a2c0f98b02442a0a0f6d79a449846	0 weeks	0	0	5e659b78b435a00045a7789
5e659b78b435a00045a7456	5 weeks	0	0	5e659b78b435a00045a7456
1770ba4f5eac481a00055a79124056	type anything	0	0	1770ba4f5eac481a00055a79124056
5e659b78b435a00045a7264	Because it L...	0	0	5e659b78b435a00045a7123
4e65a57a33434f13a813a2d1793a49b6	This is my L...	0	0	5e659b78b435a00045a7264



Elasticity:
adjust resources on demand

Pay-As-You-Go:
Actual usage
Save money on overhead



AWS SECURITY

Securing Your Cloud Application

Access: To secure a cloud using AWS, multiple layers of security are necessary. Key steps include implementing secure authentication, performing regular security audits, and keeping systems and software updated.

Policies: The connection between policies and roles is determined by the functions of the resources. Roles define the necessary functions for a resource, while policies are linked to these roles to outline the permissions.

API Security: To ensure the security of AWS Lambda and the API Gateway, it's beneficial to use an API Gateway authenticator and implement AWS Identity and Access Management (IAM). Securing AWS Lambda and the database in a cloud application involves multiple security layers, such as regular patching, using security groups, applying the principle of least privilege, and utilizing AWS IAM. When securing the S3 bucket and AWS Cloud program, it's important to regularly update and patch, use AWS IAM, access control lists, and bucket policies.



Conclusion

- **Scalability:**

Cloud platforms enable developers to easily scale their applications up or down based on real-time usage, eliminating the need for large upfront infrastructure investments and optimizing performance based on demand.

- **Pay-per-use model:**

The "pay-as-you-go" pricing structure allows developers to only pay for the resources they utilize, making cloud development cost-effective and flexible.

- **Accessibility and Collaboration:**

Cloud environments provide ubiquitous access to development tools and resources via the internet, enabling geographically dispersed teams to collaborate seamlessly on projects





THANK YOU!

Sheila Mawhinney

Sheila.Mawhinney@snhu.edu