

BÀI TẬP MÔN CÔNG NGHỆ MỚI TRONG PHÁT TRIỂN ỨNG DỤNG CNTT (Dành cho các lớp KTPM)

Mục lục

<i>Phân bổ thời gian thực hành.....</i>	3
<i>Tiêu chí đánh giá môn Công nghệ mới trong phát triển ứng dụng CNTT</i>	4
<i>BÀI TẬP TUẦN 01. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT.....</i>	5
<i>BÀI TẬP TUẦN 02. 03. 04. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT.....</i>	21
<i>BÀI TẬP TUẦN 05. 06. 07. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT.....</i>	60
<i>BÀI TẬP TUẦN 08. 09. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT.....</i>	95
<i>BÀI TẬP TUẦN 10. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT.....</i>	96

Phân bổ thời gian thực hành

Thời gian: 10 tuần, 3 tiết/tuần

Tài liệu được soạn thảo bởi cô Võ Thị Thanh Vân

- *Tuần 01.*
 - *Lập kế hoạch thực hiện đồ án.*
 - *Đăng ký tài khoản AWS.*
 - *Sử dụng 1 số dịch vụ của AWS*
 - *Phân tích nghiệp vụ để tài nhóm.*
- *Tuần 02. 03. 04.*
 - *DynamoDB và RDS*
 - *Cập nhật nhật ký thực hiện đồ án.*
 - *Lập kế hoạch test ứng dụng.*
- *Tuần 05. 06. 07.*
 - *Node.js*
 - *Cập nhật nhật ký thực hiện đồ án*
- *Tuần 08. 09.*
 - *Triển khai ứng dụng với các dịch vụ của AWS (Compute, Storage, Database, Security and Identity)*
 - *Cập nhật nhật ký thực hiện đồ án*
- *Tuần 10.*
 - *Thao tác với công cụ phát triển và quản trị ứng dụng điện toán đám mây.*
 - *Cập nhật nhật ký thực hiện đồ án*

Tiêu chí đánh giá môn Công nghệ mới trong phát triển ứng dụng CNTT

- CLO 1. Giải thích được lý do dẫn đến sự ra đời của điện toán đám mây.
- CLO 2. Hiểu và giải thích được các tính chất cơ bản của điện toán đám mây.
- CLO 3. Liệt kê được một số dịch vụ cơ bản của hai nhà cung cấp dịch vụ Cloud phổ biến nhất: Amazon Web Services (AWS) .
- CLO 4. Thực hiện được các thao tác cơ sở dữ liệu Big Data (DynamoDB)
- CLO 5. Xây dựng được ứng dụng Web (Node.js) kết nối Big Data (DynamoDB)
- CLO 6. Triển khai được ứng dụng Web trên nền điện toán đám mây (AWS)

BÀI TẬP TUẦN 01. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT

Mục tiêu:

- *Lập kế hoạch thực hiện đồ án.*
- *Đăng ký tài khoản AWS.*
- *Sử dụng 1 số dịch vụ của AWS*
- *Phân tích nghiệp vụ đề tài nhóm.*

Yêu cầu:

- *Đăng ký tài khoản AWS.*
- *Kế hoạch thực hiện có thể dùng Microsoft Word, Microsoft Project hoặc Excel, ...*

Phân 1. Lập kế hoạch thực hiện đồ án

Kế hoạch này dùng để tham khảo. Sinh viên tự tạo kế hoạch cho nhóm.

Thời gian	Nội dung
Tuần 01. 29.07.2019 04.08.2019	<ul style="list-style-type: none">▪ Chọn nhóm, đề tài.▪ Phân tích nghiệp vụ căn bản của đề tài.▪ Phân chia công việc chính.▪ Đăng ký email với AWS Educate (https://www.awseducate.com/).
Tuần 02. 05.08.2019 11.08.2019	<ul style="list-style-type: none">▪ Tìm hiểu các Cloud Computing Platform (Google App Engine, Azure, AWS).▪ So sánh các dịch vụ của các Cloud Computing Platform.▪ Đăng ký tài khoản AWS (với Credit Card cá nhân hoặc AWS Educate)▪ Tìm hiểu Scale-Web architecture.▪ Viết báo cáo phần nội dung lý thuyết nền tảng.▪ <i>Giải thích được lý do dẫn đến sự ra đời của điện toán đám mây.</i>▪ <i>Hiểu và giải thích được các tính chất cơ bản của điện toán đám mây.</i>
Tuần 03. 12.08.2019 18.08.2019	<ul style="list-style-type: none">▪ Lập kế hoạch test đề tài.▪ Đề xuất được 3 công nghệ phù hợp với bài toán cụ thể trong các công nghệ:<ul style="list-style-type: none">- Compute (EC2, Elastic Beanstalk)

	<ul style="list-style-type: none"> - Storage (S3, EFS) - Database (DynamoDB, RDS) - Networking and Content Delivery (Route53), - Security, Identity, & Compliance (IAM)
Tuần 04. 19.08.2019 25.08.2019	<ul style="list-style-type: none"> ▪ Thao tác với DynamoDB và RDS. ▪ Tìm hiểu các khái niệm liên quan đến Big Data.
Tuần 05. 26.08.2019 01.09.2019	<ul style="list-style-type: none"> ▪ Thao tác với DynamoDB và RDS. ▪ Tìm hiểu các phương pháp hiện thực và triển khai ứng dụng với Big Data.
Tuần 06. 02.09.2019 08.09.2019	<ul style="list-style-type: none"> ▪ Thao tác với DynamoDB và RDS. ▪ Tìm hiểu các phương pháp hiện thực và triển khai ứng dụng với Big Data.
Tuần 07. 09.09.2019 15.09.2019	<ul style="list-style-type: none"> ▪ Lập trình với Node.js ▪ Phân tích nghiệp vụ đề tài UML. ▪ Bổ sung tài liệu báo cáo.
Tuần 08. 16.09.2019 22.09.2019	<ul style="list-style-type: none"> ▪ Lập trình với Node.js ▪ Phân tích nghiệp vụ đề tài UML. ▪ Bổ sung tài liệu báo cáo.
Tuần 09. 23.09.2019 29.09.2019	<ul style="list-style-type: none"> ▪ Lập trình với Node.js ▪ Phân tích nghiệp vụ đề tài UML (tt). ▪ Bổ sung tài liệu báo cáo. ▪ <i>Hiện thực được Web application (Restful API) hoặc hiện thực hiện đầy đủ các giao tác với Big Data: thêm, xóa, sửa, tìm kiếm.</i>
Tuần 10. 30.09.2019 06.10.2019	<ul style="list-style-type: none"> ▪ Triển khai ứng dụng trên các dịch vụ đám mây <ul style="list-style-type: none"> - Compute (EC2, Elastic Beanstalk, Lamda) - Storage (S3, EFS)

	<ul style="list-style-type: none"> - Database (DynamoDB, RDS) - Networking and Content Delivery (Route53) - Security, Identity, & Compliance (IAM)
Tuần 11. 07.10.2019 13.10.2019	<ul style="list-style-type: none"> ▪ Triển khai ứng dụng trên các dịch vụ đám mây với AWS <ul style="list-style-type: none"> - Compute (EC2, Elastic Beanstalk, Lamda) - Storage (S3, EFS) - Database (DynamoDB, RDS) - Networking and Content Delivery (Route53) - Security, Identity, & Compliance (IAM)
Tuần 12. 07.10.2019 13.10.2019	<ul style="list-style-type: none"> ▪ Tìm hiểu các công cụ phát triển ứng dụng với điện toán đám mây <ul style="list-style-type: none"> - Visual Studio (.NET) - Eclipse (Java) - Visual Studio Code (Node.js) ▪ Hoàn tất tài liệu báo cáo của đồ án <ul style="list-style-type: none"> - Lý thuyết nền tảng - Phân tích, thiết kế đề tài (UML) - Hiện thực đề tài - Tài liệu tham khảo (liệt kê và trích dẫn tài liệu đúng theo quy định) ▪ Thực hiện kiểm thử ứng dụng (kiểm thử đề tài)
Tuần 13. 14.10.2019 20.10.2019	<ul style="list-style-type: none"> ▪ Tìm hiểu các công cụ hỗ trợ quản trị ứng dụng với điện toán đám mây <ul style="list-style-type: none"> - Database Backup - Management Tools ▪ Thực hiện kiểm thử ứng dụng (kiểm thử đề tài) ▪ Đóng gói ứng dụng, demo.
Tuần 14. 21.10.2019 28.10.2019	<ul style="list-style-type: none"> ▪ Báo cáo đề tài trước hội đồng (2 - 4 giảng viên). ▪ Đóng tài khoản AWS (nếu là tài khoản đăng ký với Credit Card cá nhân)

Phần 2. Đăng ký tài khoản

- Đăng ký tài khoản dùng Credit Card/Debit Card cá nhân (12 tháng Free Tier, hạn chế 1 số dịch vụ)
- Đăng ký tài khoản với AWS Educate. Sẽ nhận được các email hướng dẫn. Theo các bước đăng ký tài khoản.

Danh sách email chung DHKTPM12A (Sheet thứ 2):

<https://docs.google.com/spreadsheets/d/1AeutUIzTb-l3Y8lUg3qUAiZmmodBg5rDdEkiSzijoV/edit?usp=sharing>

Quy trình đăng ký:



B1. GV đăng ký lớp và gửi Email lớp cho AWS Educate.

B2. Nhận Email yêu cầu đăng ký. Tương tự Email sau:

Your AWS Educate Application Inbox x

AWS Educate Support <support@awseducate.com> Sun, Aug 4, 1:35 PM (2 days ago)

to me ...

Hi -

Your educator has invited you to join AWS Educate and access a "Classroom" for your course work. A "Classroom" is a hands-on learning environment for you to access AWS services and practice AWS. There are no costs or fees to access a Classroom.

Classrooms are managed by a third-party content and service provider, Vocareum ("Third-Party Content Provider"), and use of the Classroom feature is governed by the Third-Party Content Provider's terms and conditions (including its Privacy Policy) in addition to the AWS Educate Terms and Conditions.

If you accept the Classroom invitation, the Third-Party Content Provider may allow your educator to view your Classroom account and activity, including the AWS console in your Classroom account, the number of EC2 instances running and any Content running in the services, and your access activity.

Click [here](#) to complete the AWS Educate application process, accept your Classroom invitation and receive access to program benefits, including cloud career learning pathways and AWS Educate Promotional Credit (or a Starter Account), where applicable.

If you do not wish to proceed, ignore this email.

Thank you,

AWS Educate

B3. Nhấn vào đường link đăng ký.

Đăng ký: chọn trường Industrial University of Ho Chi Minh City. Country: Vietnam

Thông tin cá nhân.

Tháng năm sẽ tốt nghiệp 06/2020

B4. Sau khi đăng ký xong, chờ và sẽ nhận email

Email Verification - AWS Educate Application Inbox x Print Compose

AWS Educate Support <support@awseducate.com> to me ▼ 4:47 AM (11 hours ago) Star Reply More

Hello Van,

Thank you for submitting your AWS Educate application!

In order for your AWS Educate application to be processed, we need to verify your email address. Please use the verification URL below to confirm your email address and complete the application process.

<https://www.awseducate.com/ConfirmEmail?ref=93d9f70d94661eba151fe46b587883a6>

Thank you,
The AWS Educate Team

Reply Forward

B5. Sau khi đăng ký thông tin, nhận Email với các thao tác đổi password và login vào AWS Educate.

AWS Educate Application Approved [Inbox](#)

AWS Educate Support <support@awseducate.com>
to me ▾

7:37 AM (9 hours ago) [Star](#) [Reply](#) [Print](#)

Dear Van,

Congratulations!

Your AWS Educate application has been approved. As a member of the AWS Educate program, you will gain access to the benefits listed below:

[AWS Educate Student Portal](#)

The AWS Educate Student Portal is the hub for AWS Educate students around the world to find AWS content to help with classwork, connect to self-paced labs and training resources.

[Click here](#) to set your password and log in to the AWS Educate Student Portal. After logging in, click AWS Account at the top of the page to choose how you would like to access AWS services.

Bookmark the AWS Educate Student Portal for easy access, or [click here](#) to sign in directly.

You can access a video walk-through of the AWS Educate Student portal [here](#).

[Free AWS Essentials Training](#)

To access our foundational AWS Cloud Practitioner Essentials online learning class for free and find other self-paced labs, you must have either an AWS account or an Amazon ID.

- If you have an AWS account, sign in and [click here](#) to receive these benefits.
- If you do not have an AWS account, [click here](#) and follow the instructions to create an Amazon ID to access these benefits.

Once you access the Training and Certification portal, click "Learning Library" and search for "AWS Cloud Practitioner Essentials" to easily locate and enroll in AWS Cloud Practitioner Essentials on-line training. You can access AWS training any time after setting up your account by [clicking here](#).

Thank you again for participating in AWS Educate and we hope you enjoy the program!

Good luck with your continued studies,

The AWS Educate Team



aws educate

My Classrooms Portfolio Career Pathways Badges Jobs AWS Account Logout

 Van Vo

Consecutive Days: 1 Pathways Completed: 0 Badges Earned: 0

Preferred Language: English

Cloud technology is everywhere, creating over 18 million cloud jobs worldwide (source: Wanted Analytics). AWS Educate introduces you to lucrative cloud-enabled careers through more than 25 learning pathways, each with content from industry professionals, learning activities and labs, opportunities to earn AWS Educate Badges and Certificates of Completion, and access to the AWS Educate Job Board. Coupled with courses at your school or through online providers, AWS Educate puts you on the pathway to your dream job in the clouds.

Begin your journey today!



Your Journey to a Cloud Career with AWS Educate

CREATE YOUR PROFILE

GET CERTIFIED

FIND A JOB

LAND A JOB

YOUR JOURNEY TO CLOUD CAREER WITH AWS EDUCATE

ZEAL YOUR CAREER

Chon AWS Educate Starter Account



I'd like to use an AWS Educate Starter Account

Choose an AWS Educate Starter Account to receive access to an AWS account with a preset limit on your spend on AWS services. An AWS Educate Starter Account is run and managed by a third party (Vocareum, Inc.) and the Starter Account runs in the Vocareum's environment on AWS. Starter Accounts are subject to a separate agreement between you and Vocareum under separate terms and conditions.

The AWS Educate Starter Account provides access to most but not all AWS services. Students at AWS Educate member institution will receive up to \$75 (US) of AWS credit per year in their AWS Educate Starter Account, and students at non-member institution will receive up to \$30 (US) of AWS credit per year.

You don't need a credit card to use a Starter Account because AWS promotional credits are already available in the account. When your usage of AWS services exceed the balance on the account, the account is closed and any running services or other resources on the account are lost.

[Create Starter Account](#)

or choose another option



[aws !\[\]\(76135476e4c66624fcd079eb06e6e2d1_img.jpg\) educate](#)

My Classrooms Portfolio Career Pathways Badges Jobs AWS Account Logout

AWS Educate Starter Account

Your cloud journey has only just begun. Use your AWS Educate Starter Account to access the AWS Console and resources, and start building in the cloud!

[AWS Educate Starter Account](#)

Your account has an estimated **75** credits remaining and access will end on **Aug 5, 2020**.

Note: Clicking this button will take you to a third party site managed by Vocareum, Inc. ("Third Party Servicer"). In addition to the AWS Educate terms of service, your use of the AWS Educate Starter Account is governed by the Third Party Servicer's terms, including its Privacy Policy. AWS assumes no responsibility or liability and makes no representations or warranties regarding services provided by a Third Party Servicer.

Please read the terms and conditions shown below and click on the "I agree" button at the bottom of this page to continue.

Terms and Conditions

Welcome to the Vocareum, Inc. ("Vocareum") website located at www.vocareum.com (the "Site"). Please read these Terms of Service (the "Terms") and our Privacy Policy (<http://www.vocareum.com/privacy-policy/>) carefully because they govern your use of our Site and our web-based education and learning platform. To make these Terms easier to read, the Site and our platform are collectively called the "Services."

Using the functionality of our Services, teachers can create, customize and administer educational courses and invite students to participate in a class taught and supervised by the teacher using the online tools provided by Vocareum. Subject to your compliance with these Terms, Vocareum will make the Services available to you solely for the purpose of your internal, non-commercial use.

1. Agreement to Terms

By using our Services, you agree to be bound by these Terms. If you don't agree to these Terms, do not use the Services. If you are accessing and using the Services on behalf of an educational institution (such as your employer or the educational institution in which you are enrolled) or other legal entity, you represent and warrant that you have the authority to bind that educational institution or other legal entity to these Terms. In that case, "you" and "your" will refer to that educational institution or other legal entity

2. Changes to Terms or Services

Welcome to AWS Educate Starter Account

Use your Starter Account to access to a wide variety of AWS Services and start building! Click on the AWS Console button to sign in and get started.

- **What AWS services can I use in my Starter Account?**

You can use the following services in your Starter Account: apigateway, athena, cloud9, cloudformation, cloudfront, cloudtrail, cloudwatch, codecommit, codedeploy, codepipeline, cognito-identity, cognito-idp, cognito-sync, comprehend, deeplens, dynamodb, ec2, ecs, elasticache, elasticfilesystem, elasticloadbalancing, elasticmapreduce, events, execute-api, glue, iam, inspector, iot, kinesis, kinesisanalytics, firehose, kms, lambda, lex, logs, machinelearning, mobilehub, opsworks, polly, rds, rekognition, route53 (other than domain name purchasing), s3, sns, sqs, swf, sageMaker, translate, transcribe

- **What regions can I use with a Starter Account?**

- **Are Service Linked Roles supported?**

Your Starter Account Status

	Active
	full access (tv.fitse@gmail.com)
	\$75 credits (estimated)
	364d 23:57:59 remaining term
	0:60 session time

[Account Details](#) [AWS Console](#)

Welcome to AWS Educate Classroom Account

Use your AWS Educate Classroom Account to access to a wide variety of AWS Services and start building! Click on the AWS Console button to sign in and get started.

- What regions can I use with a Classroom Account?

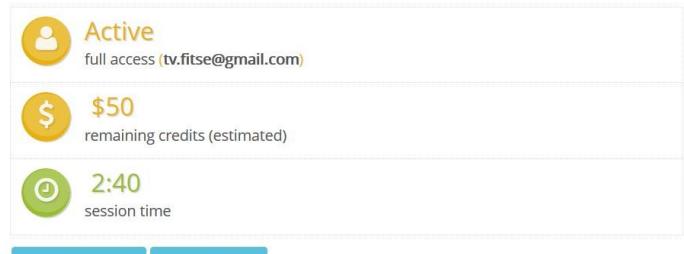
- Are Service Linked Roles supported?

- I can't start any resources. What happened?

- Can I create users within my Classroom Account for others to access?

- Can I create my own IAM policy within Starter Account or Classroom?

Your Classroom Account Status



[Account Details](#) [AWS Console](#)



Phần 3. Sử dụng 1 số dịch vụ của AWS

1. Dùng tài khoản AWS của Educate
2. AWS Console (<https://console.aws.amazon.com/>)

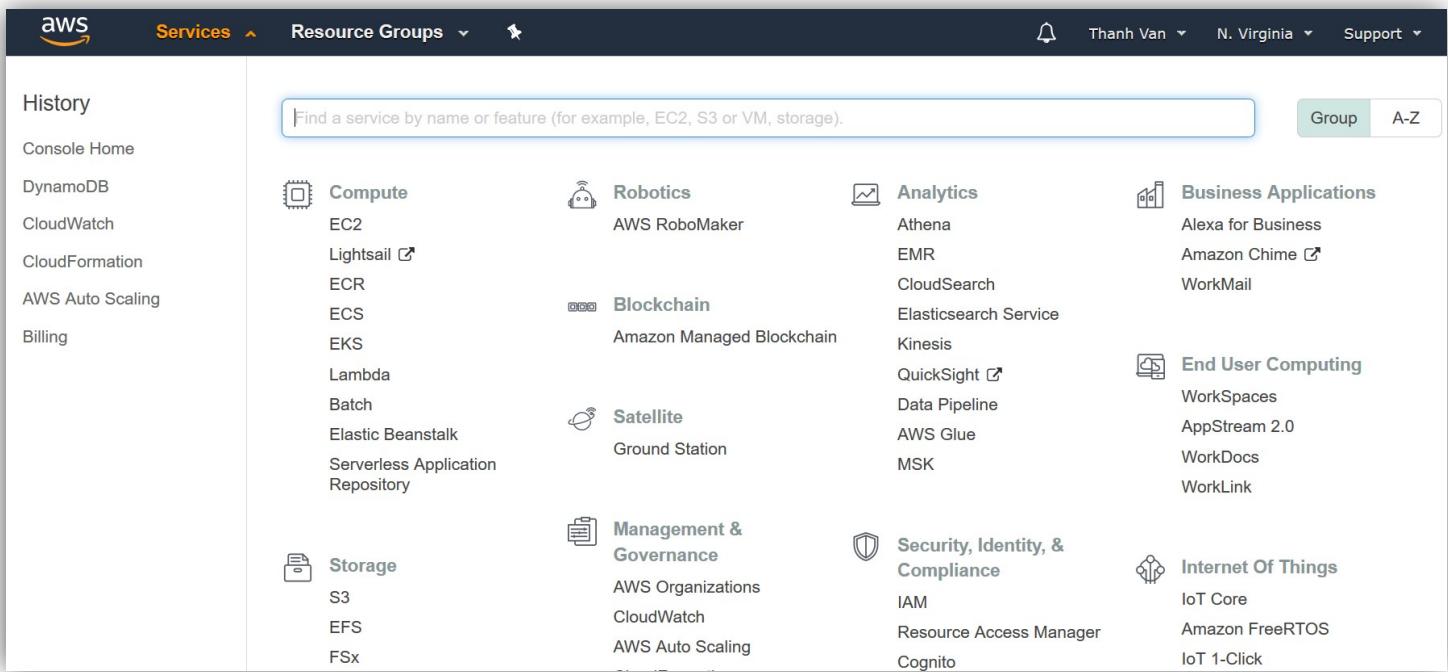
The screenshot shows the AWS Management Console homepage with the following sections:

- AWS services** (Left sidebar):
 - Find Services**: A search bar with placeholder text "Example: Relational Database Service, database, RDS".
 - Recently visited services**:
 - DynamoDB
 - CloudWatch
 - CloudFormation
 - All services**
- Build a solution** (Bottom left):

Get started with simple wizards and automated workflows.
- Access resources on the go** (Right sidebar):

Access the Management Console using the AWS Console Mobile App. [Learn more](#)
- Explore AWS** (Right sidebar):
 - Amazon RDS**: Set up, operate, and scale your relational database in the cloud. [Learn more](#)
 - Run Serverless Containers with AWS Fargate**: AWS Fargate runs and scales your containers without having to manage servers or clusters. [Learn more](#)
 - Amazon SageMaker**

Các Services của AWS



3. IAM (Identity and Access Management) giúp kiểm soát truy cập tới tài nguyên AWS.
 - Mặc định khi đăng ký AWS, tài khoản là root, tài khoản có quyền cao nhất với tất cả tài nguyên trong hệ thống AWS.
 - Nếu không sử dụng tất cả các dịch vụ của AWS thì đăng ký tài khoản IAM để giới hạn quyền truy cập tới tài nguyên cần thiết.
4. EC2

If you're using an older version of PuTTYgen, you may see a warning message about saving the key without a passphrase. You can ignore this message for now.

3. Choose Load. By default, PuTTYgen will automatically load the private key you just imported. If it doesn't, click the "Load" button and select the private key file you saved earlier.

PuTTYgen Notice

Successfully imported foreign key (OpenSSH SSH-2 private key (old PEM format)). To use this key with PuTTY, you need to use the "Save private key" command to save it in PuTTY's own format.

OK

5. To save the key in the format that PuTTY uses, click "Save private key". This will save the key without a passphrase.

Note

A passphrase on a private key can't be used without the password.

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCLpkUD/wxIMRN0QHa7ibLj9+gle/7GKfye7S50EQ/9xc0XxiSpI90m3NQsPMNM7pF11ZoxD7tLydASBM+nFVoGzeTPHE10mn8Abgm+JKjgtGiW6DAr0O3NcbpNwk3eLmiF3pI5INQVIvng6bJl5Kvl4Gxijm9USWmiaepYSJky
```

Key fingerprint: ssh-rsa 2048 59:bf:63:ec:74:5d:a5:1b:3d:20:f8:80:90:b7:f0:38

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

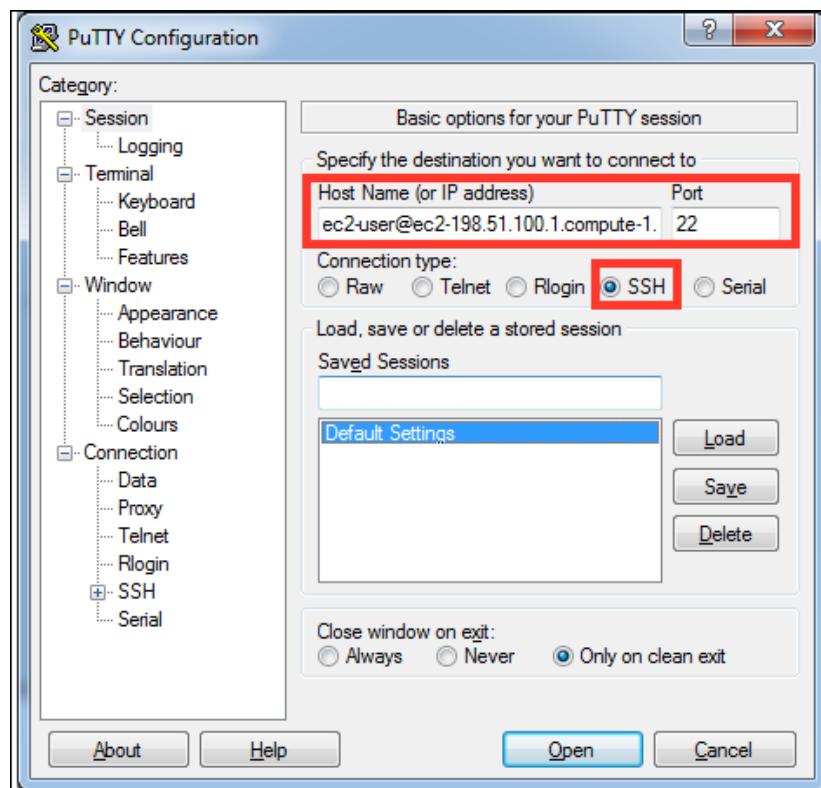
Load an existing private key file

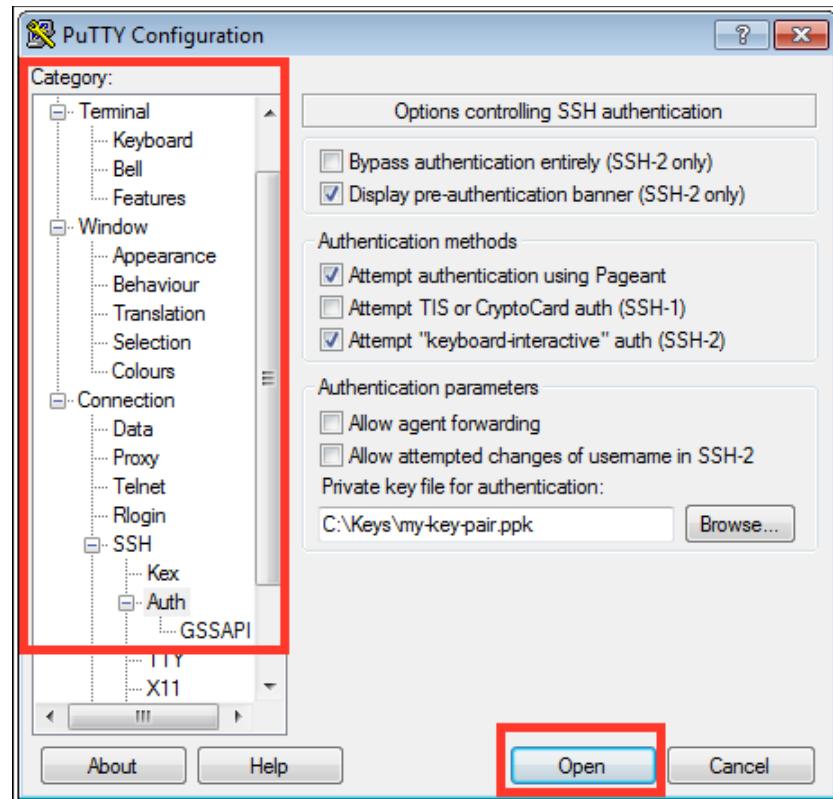
Save the generated key

Parameters

Type of key to generate: RSA DSA ECDSA Ed25519 SSH-1 (RSA)

Number of bits in a generated key: 2048

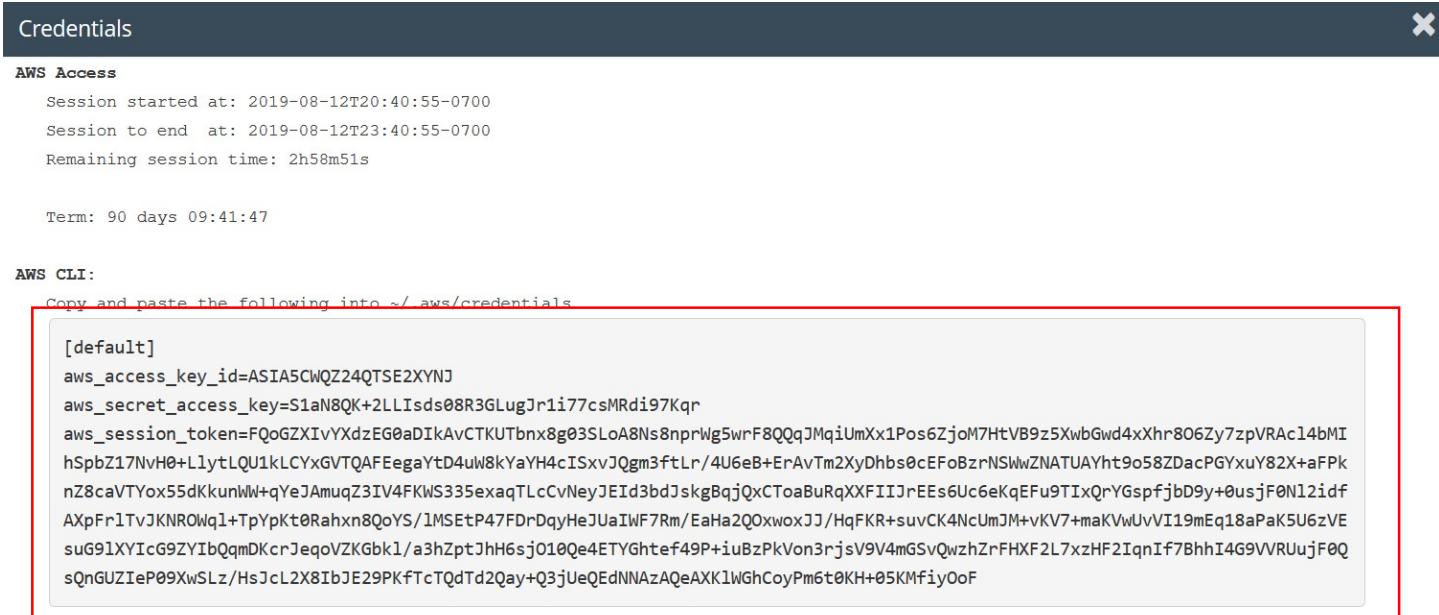




5. Elastic Beanstalk
6. S3
7. RDS

Phần 4. Thao tác với AWS Command Line Interface AWS CLI

- Với tài khoản AWS Educate, lấy trực tiếp trong phần chi tiết của tài khoản (Account Details).
- Lưu ý Token chỉ trong 1 khoản thời gian nhất định, nếu Token expired, lấy lại.



Credentials

AWS Access

```
Session started at: 2019-08-12T20:40:55-0700
Session to end at: 2019-08-12T23:40:55-0700
Remaining session time: 2h58m51s

Term: 90 days 09:41:47
```

AWS CLI:

```
Copy and paste the following into ~/.aws/credentials
```

```
[default]
aws_access_key_id=ASIA5CWQZ24QTSE2XYNj
aws_secret_access_key=S1aN8QK+2LLIsds08R3GLugJr1i77csMRdi97Kqr
aws_session_token=FPoGZXIVYXdzEG0aDIkAvCTKUTbnx8g03SLoA8Ns8nprWg5wrF8QQqJMqiUmXx1Pos6ZjoM7HtVB9z5XwbGwd4xXhr806Zy7zpVRAc14bMI
hSpbZ17NvH0+LlytLQu1kLCYxGVTQAFEEgaYtD4uW8kYaYH4cISxvJQgm3ftLr/4U6eB+ErAvTm2XyDhbs0cEFoBzrNSWzNATUAYht9o58ZDacPGYxuY82X+aFPk
nZ8caVTVox55dKkunlw+qYeJAmuqZ3IV4FKWS335exaqTLcCvNeyJEId3bdJskgBqjQxCToaBuRqXXFIIJrEEs6Uc6eKqEFu9TiXQrYGspfjbD9y+0usjF0Nl2idf
AXpFr1TvJKNR0Wql+TpYpKt0Rahxn8QoS/1MSEtP47FDtDqyHeJuIWF7Rm/EaHa2QOxwoxJJ/HqFKR+suvCK4NcUmJM+vKV7+maKVwUvVI19mEq18aPaK5U6zVE
suG91XYIcG9ZYIbQqmDKcrJeqoVZKgbk1/a3hZptJhH6sj010Qe4ETYGhtef49P+iuBzPkVon3rjsV9V4mGSvQwzhZrFHxF2L7xzHF2IqnIf7BhhI4G9VVRUuojF0Q
sQnGUZIeP09XwSLz/HsJcL2X8IbJE29PKfTcTQdT2Qay+Q3jUeQEdNNAzAqEAXK1WGHCoypM6t0KH+05KMfiyOoF
```

- Cài đặt AWS CLI, lấy file download 32/64 bits trên AWS documentation website.
- Cấu hình AWS CLI (trường hợp AWS Starter, không dùng bước này, lấy file lưu vào C:\Users\TenUser\.aws\credentials)

```
C:\Users\HP>aws configure
AWS Access Key ID [*****5VQ1]: ASIARALGJSGDZIRIB4OS
AWS Secret Access Key [*****ge71]: zZmL317KTfoKxe1nXOpE/SAlbqEDBkybr50cqxpQ
Default region name [None]: us-east-1
Default output format [None]: json
```

1. AWS CLI EC2 (<https://docs.aws.amazon.com/cli/latest/reference/ec2/>)

- aws ec2 describe-instances
- aws ec2 describe-instances --filter Name=tag:Name,Values=dev-server # filter
- aws ec2 start-instances --instance-ids i-00e0c79d05682c511
- aws ec2 stop-instances --instance-ids i-00e0c79d05682c511
- aws ec2 terminate-instances --instance-ids i-00e0c79d05682c511
- aws ec2 reboot-instances --instance-ids i-00e0c79d05682c511
- aws ec2 get-console-output --instance-id i-00e0c79d05682c511
- aws ec2 describe-key-pairs i-00e0c79d05682c511

2. AWS CLI cho S3 cú pháp chung: `aws s3 <Command> [<Arg> ...]`

2.1. Tạo bucket

```
aws s3 mb s3://mybucket
```

```
aws s3 mb s3://mybucket --region us-west-1
```

2.2. Liệt kê

```
aws s3 ls s3://mybucket --recursive
```

```
aws s3 ls
```

2.3. Di chuyển object/file (<https://docs.aws.amazon.com/cli/latest/reference/s3/mv.html>)

2.4. Xóa

- Bucket chỉ xóa được khi bucket rỗng (không objects và versioned objects)
- Muốn xóa bucket không rỗng dùng --force

```
aws s3 rb s3://mybucket
```

```
aws s3 rb s3://mybucket --force
```

2.5. Sync file(s) từ bucket xuống Local PC /Laptop – Sync file(s) sẽ download/upload các file(s) còn thiếu giữ local và cloud hoặc ngược lại.

```
aws s3 sync s3://mybucket C:\Users\S3
```

2.6. Các lệnh Copy trong S3 (<https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html>)

- aws s3 cp test.txt s3://mybucket/test2.txt
- **Copy tập tin Local lên S3 với expiration date**

```
aws s3 cp test.txt s3://mybucket/test2.txt --expires 2014-10-01T20:30:00Z
```

- aws s3 cp s3://mybucket/test.txt s3://mybucket/test2.txt

- **Copy đối tượng S3 về Local**

```
aws s3 cp s3://mybucket/test.txt test2.txt
```

- aws s3 cp s3://mybucket/test.txt s3://mybucket2/

- **Recursively copying S3 objects to a local directory**
`aws s3 cp s3://mybucket . --recursive`
- **Recursively copying local files to S3**
`aws s3 cp myDir s3://mybucket/ --recursive --exclude "*.jpg"`
- **Recursively copying S3 objects to another bucket**
`aws s3 cp s3://mybucket/ s3://mybucket2/ --recursive --exclude "another/*"`
`aws s3 cp s3://mybucket/logs/ s3://mybucket2/logs/ --recursive --exclude "*" --include "*.*log"`
- **Gán Access Control List (ACL) khi copy S3 object**
`aws s3 cp s3://mybucket/test.txt s3://mybucket/test2.txt --acl public-read-write`

3. CLI với DynamoDB

- Tạo bảng

```
aws dynamodb create-table
  --table-name MusicCollection
  --attribute-definitions
    AttributeName=Artist,
    AttributeType=S
    AttributeName=SongTitle,
    AttributeType=S
  --key-schema AttributeName=Artist,KeyType=HASH
  AttributeName=SongTitle,KeyType=RANGE
  --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1
```

- Tạo Item

```
aws dynamodb put-item
  --table-name MusicCollection
  --item '{
    "Artist": {"S": "No One You Know"},
    "SongTitle": {"S": "Call Me Today"} ,
    "AlbumTitle": {"S": "Somewhat Famous"} }'
```

--return-consumed-capacity TOTAL

```
aws dynamodb put-item  
  --table-name MusicCollection  
  --item '{  
    "Artist": {"S": "Acme Band"},  
    "SongTitle": {"S": "Happy Day"} ,  
    "AlbumTitle": {"S": "Songs About Life"} }'  
  --return-consumed-capacity TOTAL
```

- Truy vấn

```
aws dynamodb query --table-name MusicCollection  
  --key-condition-expression "Artist = :v1 AND SongTitle = :v2"  
  --expression-attribute-values file://expression-attributes.json
```

Phần 5. Phân tích nghiệp vụ đề tài nhóm.

- Phân tích các tình huống hoạt động của ứng dụng (tối thiểu 8 tình huống có kèm nghiệp vụ).
- Vẽ các Use case tổng quát, đặc tả các hoạt động.

BÀI TẬP TUẦN 02. 03. 04. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT

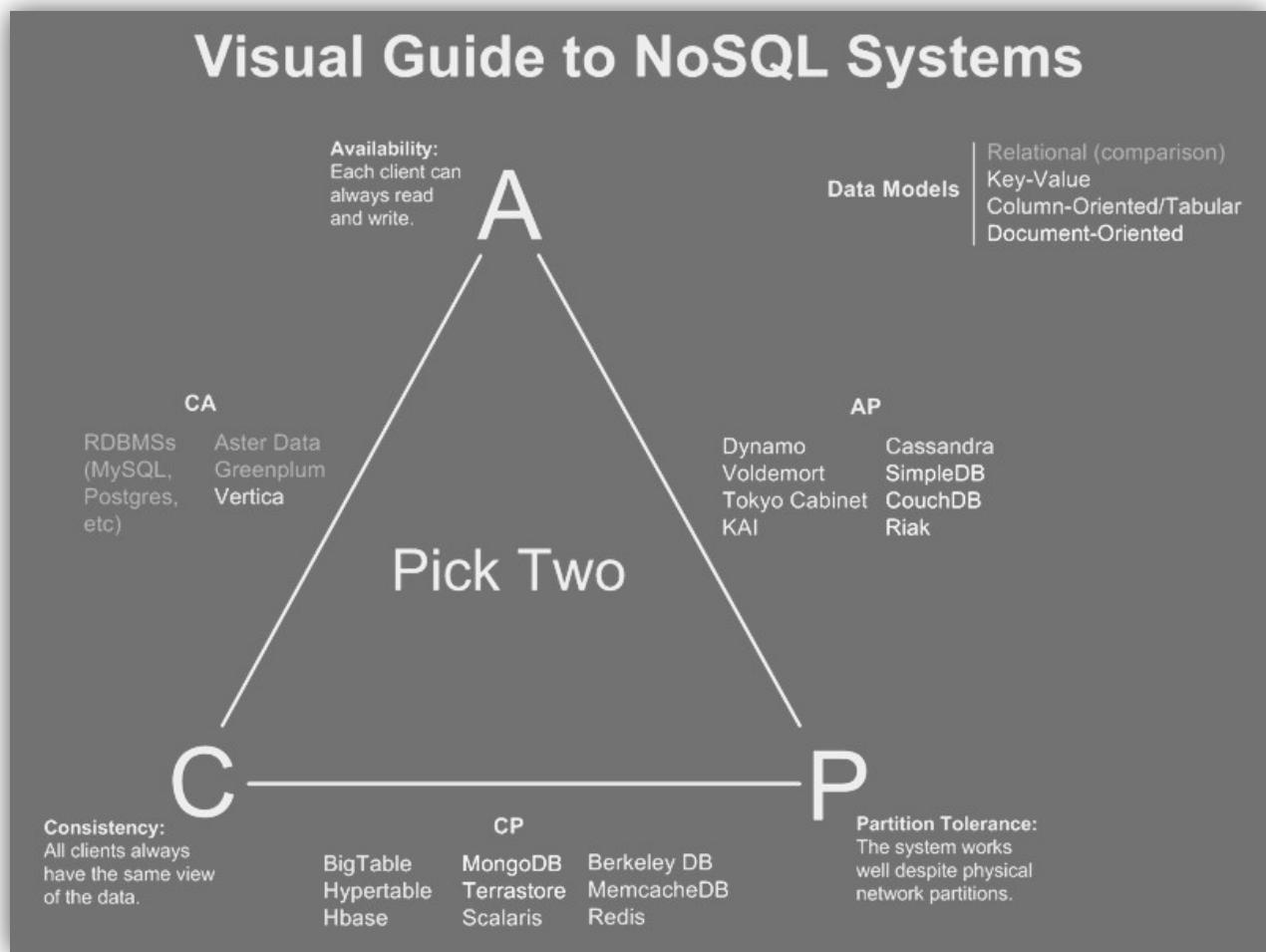
Mục tiêu:

- Hiểu và sử dụng được DynamoDB và RDS.
- Cài đặt AWS toolkit cho Eclipse và .NET.
- Cập nhật nhật ký thực hiện đồ án.
- Lập kế hoạch kiểm thử ứng dụng.

Yêu cầu:

- Đã có tài khoản AWS.
- Hoàn tất viết báo cáo phần nội dung lý thuyết nền tảng.
- Có thể dùng DynamoDB Local. Download phiên bản cài đặt.
- Kế hoạch kiểm thử có thể dùng Microsoft Word, hoặc Excel, cập nhật kế hoạch này trong file báo cáo.

Phần 1. *Big Data , NoSQL và DynamoDB*



Phân 2. Cài đặt offline DynamoDB

`java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar [options]`

Thay đổi port (**netstat -an**: Kiểm tra port đang chạy trong hệ thống)

`java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -port 8899`

Dùng JavaScript Shell thao tác với DynamoDB Local

The screenshot shows the DynamoDB JavaScript Shell interface. On the left, there is a code editor window with the following JavaScript code:

```
// This CreateTable request will create the Image table.  
// With DynamoDB Local, tables are created right away. If you  
// calling  
// a real DynamoDB endpoint, you will need to wait for the tab  
// become  
// ACTIVE before you can use it. See also dynamodb.waitFor().  
var params = {  
    TableName: 'Image',  
    KeySchema: [  
        {  
            AttributeName: 'Id',  
            KeyType: 'HASH'  
        }  
    ],  
    AttributeDefinitions: [  
        {  
            AttributeName: 'Id',  
            AttributeType: 'S'  
        }  
    ],  
    ProvisionedThroughput: {  
        ReadCapacityUnits: 1,  
        WriteCapacityUnits: 1  
    }  
};  
console.log("Creating the Image table");  
dynamodb.createTable(params, function(err, data) {  
    if (err) ppjson(err); // an error occurred  
    else ppjson(data); // successful response  
});
```

On the right, the results of the command execution are displayed in a large text area:

```
0: {  
    "AttributeName": "Id",  
    "KeyType": "HASH",  
    "TableStatus": "ACTIVE",  
    "CreationDateTime": "2019-07-12T06:46:41.307Z",  
    "ProvisionedThroughput": {  
        "LastIncreaseDateTime": "1970-01-01T00:00:00.000Z",  
        "LastDecreaseDateTime": "1970-01-01T00:00:00.000Z",  
        "NumberOfDecreasesToday": 0  
    },  
    "ReadCapacityUnits": 1,  
    "WriteCapacityUnits": 1,  
    "TableSizeBytes": 0,  
    "ItemCount": 0,  
    "TableArn": "arn:aws:dynamodb:ddblocal:000000000000:table/Image"
```

Dùng AWS CLI liệt kê các bảng trong DynamoDB Local

`aws dynamodb list-tables --endpoint-url http://localhost:8000`

Phân 3. Thao tác thêm, xóa, sửa, load table với DynamoDB (<https://aws.amazon.com/dynamodb/>).

Thành phần nền tảng của DynamoDB:

- **Tables** – Similar to other database systems, DynamoDB stores data in tables. A table is a collection of data.
- **Items** – Each table contains zero or more items. An item is a group of attributes that is uniquely identifiable among all of the other items. In DynamoDB, there is no limit to the number of items

you can store in a table. Some of the items have a nested attribute (People, Address). DynamoDB supports nested attributes up to 32 levels deep.

- **Attributes** – Each item is composed of one or more attributes. An attribute is a fundamental data element, something that does not need to be broken down any further. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.



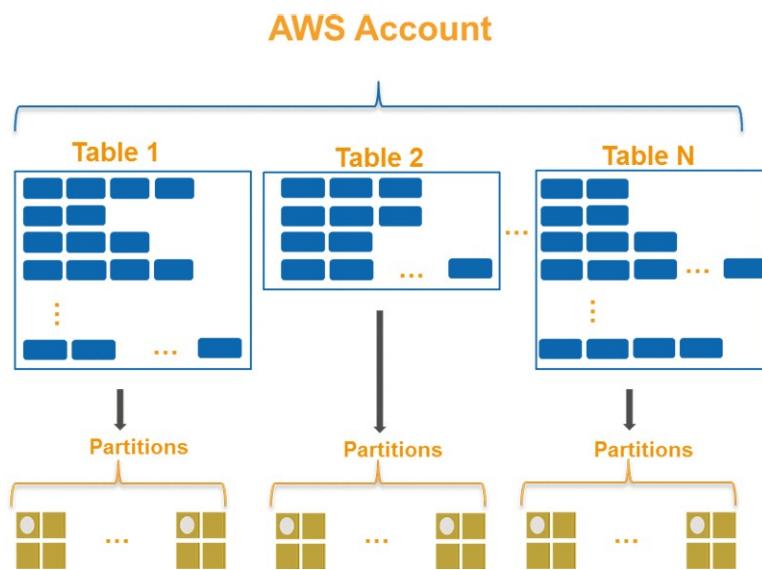
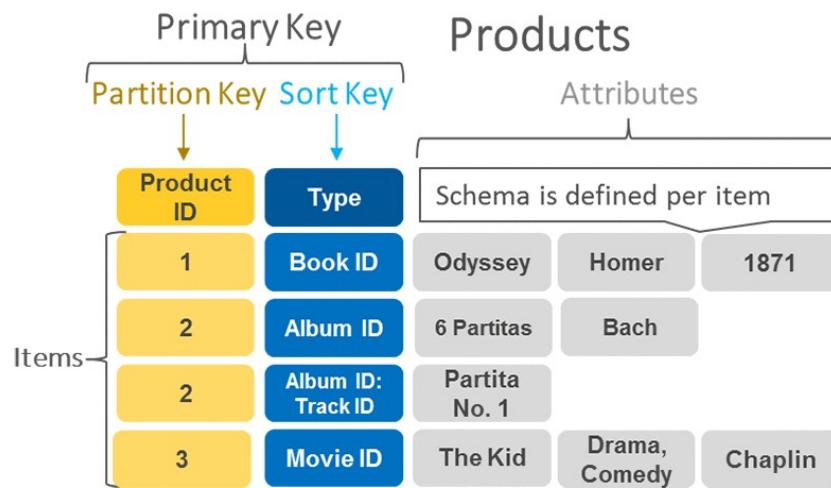
DynamoDB hỗ trợ 2 loại khóa chính trong bảng:

- **Partition key** – A simple primary key, composed of **one attribute** known as the partition key.

DynamoDB uses the partition key's value as input to an internal **hash function**. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored.

- **Partition key and sort key** – Referred to as a composite primary key, this type of key is composed of two attributes. The first attribute is the partition key, and the second attribute is the sort key.

DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored. All items with the same partition key value are stored together, in sorted order by sort key value.



Secondary Index – DynamoDB hỗ trợ 2 loại indexes (Mỗi bảng trong DynamoDB giới hạn 20 GSI và 5 LSI):

- **Global secondary index** — An index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions.

- **Local secondary index** — An index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.



SQL trong RDBMS

```
CREATE TABLE Music (
  Artist VARCHAR(20) NOT NULL,
  SongTitle VARCHAR(30) NOT NULL,
  AlbumTitle VARCHAR(25),
  Year INT,
  Price FLOAT,
  Genre VARCHAR(10),
  Tags TEXT,
  PRIMARY KEY(Artist, SongTitle)
);
```

DynamoDB

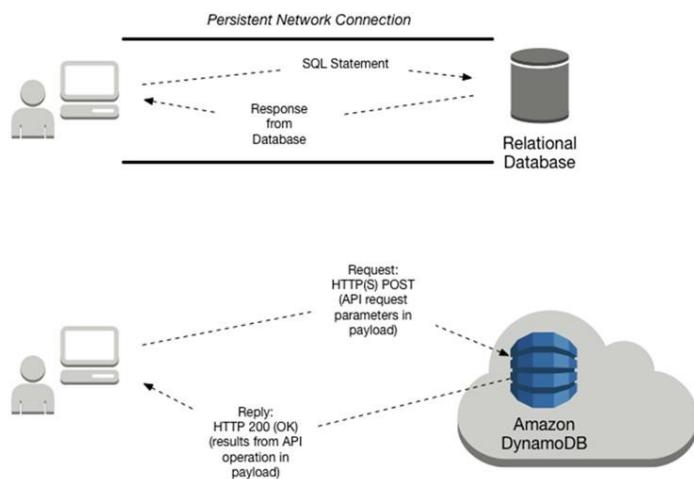
```
{
```

```

TableName : "Music",
KeySchema: [
    {
        AttributeName: "Artist",
        KeyType: "HASH", //Partition key
    },
    {
        AttributeName: "SongTitle",
        KeyType: "RANGE" //Sort key
    }
],
AttributeDefinitions: [
    {
        AttributeName: "Artist",
        AttributeType: "S"
    },
    {
        AttributeName: "SongTitle",
        AttributeType: "S"
    }
],
ProvisionedThroughput: {      // Only specified if using provisioned mode
    ReadCapacityUnits: 1,
    WriteCapacityUnits: 1
}
}

```

Thao tác từ Client đến RDBMS và DynamoDB



Kiểu dữ liệu trong DynamoDB

- **Scalar Types** – A scalar type can represent exactly one value. The scalar types are number, string, binary, Boolean, and null.
- **Document Types** – A document type can represent a complex structure with nested attributes (32 levels deep), such as you would find in a JSON document. The document types are list ([]) and map ({})(giới hạn cho đến 400 KB).
- **Set Types** – A set type can represent multiple scalar values. The set types are string set, number set, and binary set. (cùng loại)

Các thao tác với AWS Management Console

▪ Tạo bảng

Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Create table 1

Getting started guide

Create tables Create DynamoDB tables with a few clicks. Just specify the desired read and write throughput for your table, and DynamoDB handles the rest.
More about DynamoDB throughput

Add and query items Once you have created a DynamoDB table, use the AWS SDKs to write, read, modify, and query items in DynamoDB.
DynamoDB API reference

Monitor and manage tables Using the AWS Management Console, you can monitor performance and adjust the throughput of your tables, enabling you to scale seamlessly.
Monitoring tables

DynamoDB documentation & support

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* 2

Primary key* Partition key
 3

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- Encryption at Rest with DEFAULT encryption type **NEW!**

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Create 4 **Cancel**

Employee Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Triggers Access control Tags

Manage Stream

Your Table is Created.

Table details

Table name	Employee
Primary partition key	EmpID (String)
Primary sort key	-
Point-in-time recovery	DISABLED Enable
Encryption Type	DEFAULT
KMS Master Key ARN	Not Applicable
Time to live attribute	DISABLED Manage TTL
Table status	Active
Creation date	February 22, 2019 at 11:52:47 AM UTC+5:30
Last change to on-demand mode	Provisioned
Provisioned read capacity units	-
Provisioned write capacity units	5 (Auto Scaling Error)
Last decrease time	5 (Auto Scaling Error)
Last increase time	-
Storage size (in bytes)	-
Item count	0 bytes
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:xxxxxx:table/Employee

Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours.

■ Thêm item

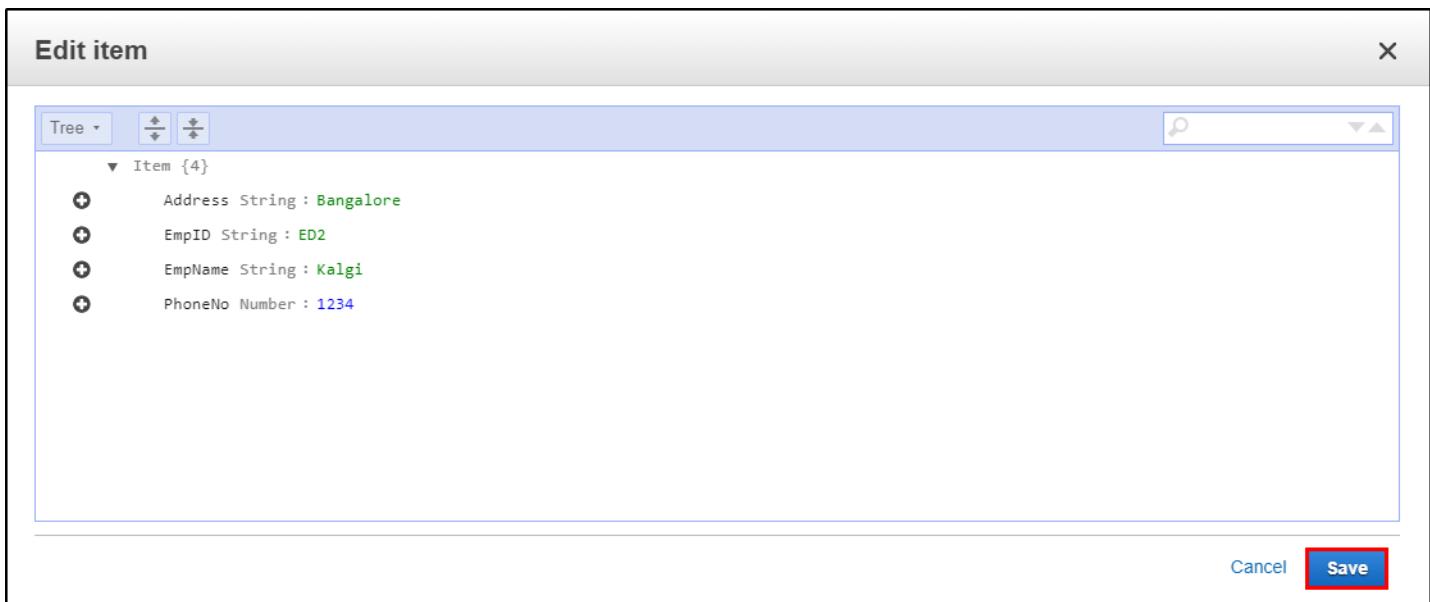
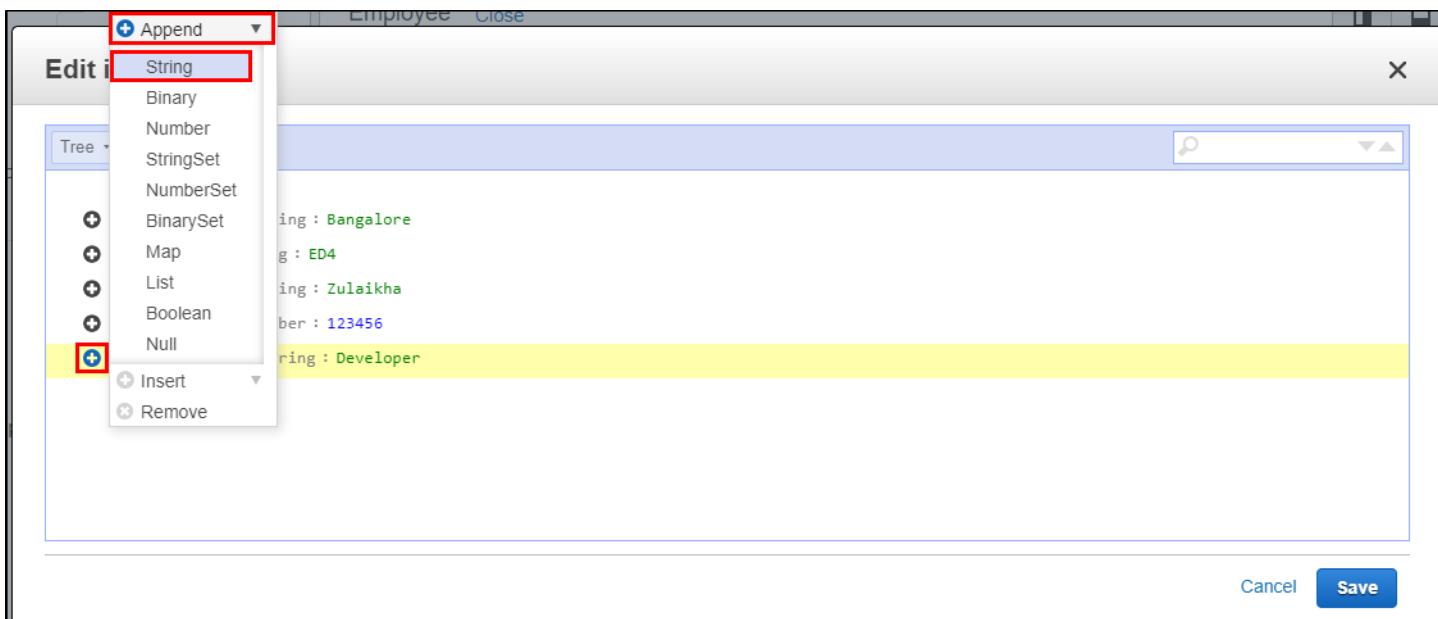
Employee Close

Overview **Items** Metrics Alarms Capacity Indexes Global Tables Backups Triggers Access control Tags

Create item Actions ▾

Scan: [Table] Employee: EmpID ▾ Viewing 1 to 2 items

Scan ▾ [Table] Employee: EmpID ▾ Add filter Start search



Employee [Close](#)

[Overview](#) **Items** [Metrics](#) [Alarms](#) [Capacity](#) [Indexes](#) [Global Tables](#) [Backups](#) [Triggers](#) [Access control](#) [Tags](#)

[Create item](#) [Actions](#)

Scan: [Table] Employee: EmpID ^ Viewing 1 to 4 items

	EmpID	Address	EmpName	PhoneNo	Position
<input type="checkbox"/>	ED1	Bangalore	Priyaj	123	
<input type="checkbox"/>	ED2	Bangalore	Kalgi	1234	
<input type="checkbox"/>	ED3	Bangalore	Omkar	12345	
<input type="checkbox"/>	ED4	Bangalore	Zulaikha	123456	Developer

Tìm kiếm, lọc

Employee [Close](#)

[Overview](#) **Items** [Metrics](#) [Alarms](#) [Capacity](#) [Indexes](#) [Global Tables](#) [Backups](#) [Triggers](#) [Access control](#) [Tags](#)

[Create item](#) [Actions](#)

Scan: [Table] Employee: EmpID ^ Viewing 1 to 3 items

	EmpID	Address	EmpName	PhoneNo	Position
<input type="checkbox"/>	ED2	Bangalore	Kalgi	1234	
<input type="checkbox"/>	ED3	Bangalore	Omkar	12345	
<input type="checkbox"/>	ED4	Bangalore	Zulaikha	123456	Developer

1 **Filter** PhoneNo Number >= 1234

2 Start search

Employee Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Triggers Access control Tags

Create item Actions ▾

Scan: [Table] Employee: EmpID ^ Viewing 1 to 1 items

Scan [Table] Employee: EmpID

Filter EmpID String = ED4

1 Add filter 2 Start search

	EmpID	Address	EmpName	PhoneNo	Position
<input checked="" type="checkbox"/>	ED4	Bangalore	Zulaikha	123456	Developer

Yêu cầu: Tạo các bảng

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html>

Bảng ProductCatalog

Primary key, Partition key: Id - Number.

Bảng Forum

Primary key, Partition key: Name - String.

Bảng Thread

Primary key gồm:

Partition key, ForumName - String.

Sort key, Subject - String.

Bảng Table

Primary key gồm:

Partition key, Id - String.

Sort key, ReplyDateTime - String.

Thêm index tên PostedBy-Message-Index:

Partition key, PostedBy - String.

Sort key, Message - String.

Gán Projected attributes:All.

Thêm dữ liệu vào bảng dùng AWS CLI

```
aws dynamodb batch-write-item --request-items file://ProductCatalog.json  
aws dynamodb batch-write-item --request-items file://Forum.json  
aws dynamodb batch-write-item --request-items file://Thread.json  
aws dynamodb batch-write-item --request-items file://Reply.json
```

Phân 4. AWS CLI với DynamoDB

1. Liệt kê bảng trong CSDL

```
aws dynamodb list-tables
```

2. Mô tả thành phần trong Table

```
aws dynamodb describe-table --table-name Music
```

3. Tạo Provisioned Table

```
aws dynamodb create-table  
--table-name Music  
--attribute-definitions  
  AttributeName=Artist,AttributeType=S  
  AttributeName=SongTitle,AttributeType=S  
--key-schema  
  AttributeName=Artist,KeyType=HASH  
  AttributeName=SongTitle,KeyType=RANGE  
--provisioned-throughput  
  ReadCapacityUnits=10,WriteCapacityUnits=5
```

4. Tạo On-Demand Table

```
aws dynamodb create-table  
--table-name Music  
--attribute-definitions  
  AttributeName=Artist,AttributeType=S  
  AttributeName=SongTitle,AttributeType=S  
--key-schema  
  AttributeName=Artist,KeyType=HASH  
  AttributeName=SongTitle,KeyType=RANGE  
--billing-mode=PAY_PER_REQUEST
```

5. Cập nhật, thay đổi thông tin Table

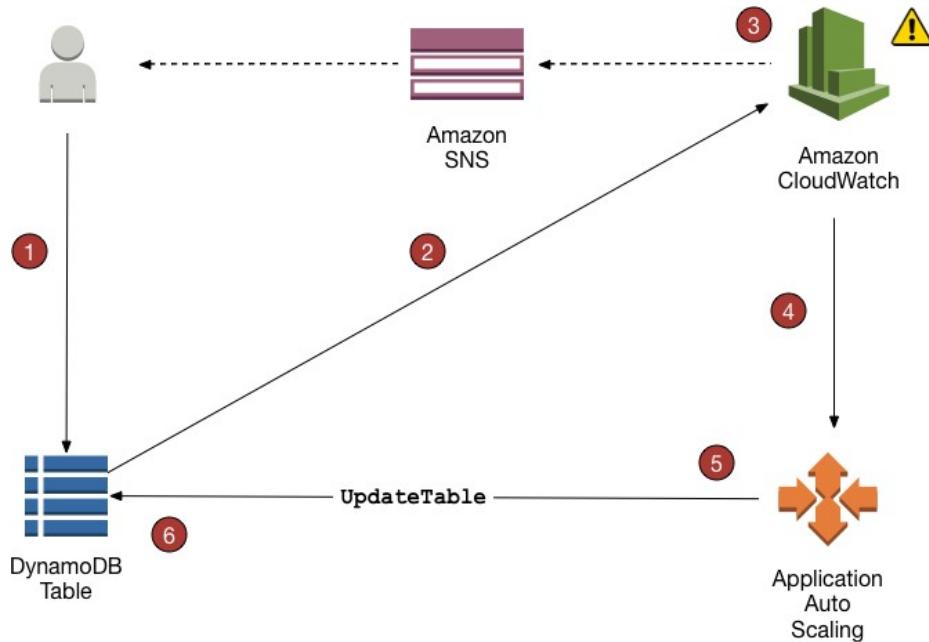
```
aws dynamodb update-table --table-name Music  
--provisioned-throughput ReadCapacityUnits=20,WriteCapacityUnits=10
```

```
aws dynamodb update-table --table-name Music  
--billing-mode PAY_PER_REQUEST
```

6. Xóa Table

```
aws dynamodb delete-table --table-name Music
```

7. Auto Scaling với bảng trong DynamoDB



8. Reading an Item

```
aws dynamodb get-item  
--table-name ProductCatalog  
--key '{"Id":{"N":"1"}}'
```

```
aws dynamodb get-item  
--table-name ProductCatalog  
--key '{"Id":{"N":"1"}}'  
--consistent-read  
--projection-expression "Description, Price, RelatedItems"  
--return-consumed-capacity TOTAL
```

```
aws dynamodb get-item
```

```

--table-name ProductCatalog
--key file://key.json
--projection-expression "Description, RelatedItems[0], ProductReviews.FiveStar"

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "Comment"

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "#c"
--expression-attribute-names '{"#c":"Comment"}'

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "Safety.Warning"

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "#sw"
--expression-attribute-names '{"#sw":"Safety.Warning"}'

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "ProductReviews.OneStar"

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "#pr1star"
--expression-attribute-names '{"#pr1star":"ProductReviews.OneStar"}'

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "#pr.#1star"
--expression-attribute-names '{"#pr":"ProductReviews", "#1star":"OneStar"}'

```

aws dynamodb get-item

```
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "ProductReviews.FiveStar, ProductReviews.ThreeStar,
ProductReviews.OneStar"

aws dynamodb get-item
--table-name ProductCatalog
--key '{"Id":{"N":"123"}}'
--projection-expression "#pr.FiveStar, #pr.ThreeStar, #pr.OneStar"
--expression-attribute-names '{"#pr":"ProductReviews"}
```

```
aws dynamodb batch-get-item
--request-items file://request-items.json
```

9. Thao tác chèn, cập nhật và xóa Item

```
aws dynamodb put-item
--table-name Thread
--item file://item.json
```

```
aws dynamodb put-item
--table-name ProductCatalog
--item file://item.json
--condition-expression "attribute_not_exists(Id)"
```

```
aws dynamodb update-item
--table-name Thread
--key file://key.json
--update-expression "SET Answered = :zero, Replies = :zero, LastPostedBy = :lastpostedby"
--expression-attribute-values file://expression-attribute-values.json
--return-values ALL_NEW
```

```
aws dynamodb delete-item
--table-name Thread
--key file://key.json
```

```
aws dynamodb delete-item
--table-name ProductCatalog
--key '{"Id": {"N": "456"}}'
--condition-expression "attribute_not_exists(Price)"
```

```
aws dynamodb delete-item
--table-name ProductCatalog
--key '{"Id": {"N": "456"}}'
```

```
--condition-expression "attribute_exists(ProductReviews.OneStar)"
```

```
aws dynamodb delete-item
--table-name ProductCatalog
--key '{"Id":{"N":"456"}}'
--condition-expression "(ProductCategory IN (:cat1, :cat2)) and (Price between :lo and :hi)"
--expression-attribute-values file://values.json
```

```
aws dynamodb update-item
--table-name ProductCatalog
--key '{"Id": {"N": "456"}}'
--update-expression "SET Price = Price - :discount"
--condition-expression "Price > :limit"
--expression-attribute-values file://values.json
```

```
aws dynamodb update-item
--table-name ProductCatalog
--key '{"Id":{"N":"789"}}'
--update-expression "SET ProductCategory = :c, Price = :p"
--expression-attribute-values file://values.json
--return-values ALL_NEW
```

```
aws dynamodb update-item
--table-name ProductCatalog
--key '{"Id":{"N":"789"}}'
--update-expression "SET RelatedItems = :ri, ProductReviews = :pr"
--expression-attribute-values file://values.json
--return-values ALL_NEW
```

```
aws dynamodb update-item
--table-name ProductCatalog
--key '{"Id": {"N": "789"}}'
--update-expression "SET RelatedItems[1] = :ri"
--expression-attribute-values file://values.json
--return-values ALL_NEW
```

10. Query (<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Query.html>)

To specify the search criteria, you use a **key condition expression**—a string that determines the items to be read from the table or index.

You must specify the partition key name and value as an equality condition.

You can optionally provide a second condition for the sort key (if present). The sort key condition must use one of the following comparison operators:

- $a = b$ — true if the attribute a is equal to the value b
- $a < b$ — true if a is less than b
- $a \leq b$ — true if a is less than or equal to b
- $a > b$ — true if a is greater than b
- $a \geq b$ — true if a is greater than or equal to b
- $a \text{ BETWEEN } b \text{ AND } c$ — true if a is greater than or equal to b , and less than or equal to c .

The following function is also supported:

- `begins_with(a, substr)`— true if the value of attribute a begins with a particular substring

```
aws dynamodb query
--table-name Thread
--key-condition-expression "ForumName = :name"
--expression-attribute-values '{":name":{"S":"Amazon DynamoDB"}}'
```

```
aws dynamodb query
--table-name Thread
--key-condition-expression "ForumName = :name and Subject = :sub"
--expression-attribute-values file://values.json
```

```
aws dynamodb query
--table-name Reply
--key-condition-expression "Id = :id and begins_with(ReplyDateTime, :dt)"
--expression-attribute-values file://values.json
```

Lọc truy vấn

```
aws dynamodb query
--table-name Thread
--key-condition-expression "ForumName = :fn"
--filter-expression "#v >= :num"
--expression-attribute-names '{"#v": "Views"}'
--expression-attribute-values file://values.json
aws dynamodb query --table-name Movies
--projection-expression "title"
--key-condition-expression "#y = :yyyy"
--expression-attribute-names '{"#y": "year"}'
--expression-attribute-values '{":yyyy":{"N":"1993"}}'
--page-size 5
--debug
```

11. Scan

```
aws dynamodb scan
  --table-name Thread
  --filter-expression "LastPostedBy = :name"
  --expression-attribute-values '{"name":{"S":"User A"}}'
aws dynamodb scan
  --table-name Movies
  --projection-expression "title"
  --filter-expression 'contains(info.genres,:gen)'
  --expression-attribute-values '{"gen":{"S":"Sci-Fi"}}'
  --page-size 100
  --debug
```

12. Thực hiện và giải thích trên bảng MusicCollection

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SQLtoNoSQL.ReadData.Query.html>

- ```
aws dynamodb create-table --table-name MusicCollection \
 --attribute-definitions AttributeName=Artist,AttributeType=S
 AttributeName=SongTitle,AttributeType=S \
 --key-schema AttributeName=Artist,KeyType=HASH
 AttributeName=SongTitle,KeyType=RANGE \
 --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
 --endpoint-url http://localhost:8000
```
- ```
aws dynamodb list-tables --endpoint-url http://localhost:8000
aws dynamodb --endpoint-url http://localhost:8000 put-item --table-name MusicCollection \
  --item '{"Artist": {"S": "Bring me the Horizon"}, "SongTitle": {"S": "Sleepwalking"}, \
  "AlbumTitle": {"S": "Sempiternal"}}'
```
- ```
aws dynamodb --endpoint-url http://localhost:8000 get-item --table-name MusicCollection \
 --key '{"Artist": {"S": "Bring me the Horizon"}, "SongTitle": {"S": "Sleepwalking"}}'
```
- ```
aws dynamodb --endpoint-url http://localhost:8000 get-item --table-name MusicCollection \
  --attributes-to-get ['AlbumTitle', 'SongTitle'] \
  --key '{"Artist": {"S": "Bring me the Horizon"}, "SongTitle": {"S": "Sleepwalking"}}'
```
- ```
aws dynamodb --endpoint-url http://localhost:8000 get-item --table-name MusicCollection \
 --key '{"Artist": {"S": "Bring me the Horizon"}, "SongTitle": {"S": "Sleepwalking"}}'
```

```
--projection-expression "AlbumTitle, SongTitle" \
--key '{"Artist": {"S": "Bring me the Horizon"}, "SongTitle": {"S": "Sleepwalking"}}'
```

```
aws dynamodb batch-write-item --request-items file://music-table/batch-write-songs.json --
endpoint-url http://localhost:8000
```

- `aws dynamodb --endpoint-url http://localhost:8000 scan --table-name MusicCollection`
- `aws dynamodb --endpoint-url http://localhost:8000 query --select ALL_ATTRIBUTES \
--table-name MusicCollection \
--key-condition-expression "Artist = :a" \
--expression-attribute-values '{":a":{"S":"AC/DC"}}'`
- `aws dynamodb --endpoint-url http://localhost:8000 query --select ALL_ATTRIBUTES \
--table-name MusicCollection \
--key-condition-expression "Artist = :a and SongTitle = :t" \
--expression-attribute-values '{":a":{"S":"AC/DC"}, ":t":{"S":"You Shook Me All Night Long"}}'`
- `aws dynamodb --endpoint-url http://localhost:8000 query --select ALL_ATTRIBUTES \
--table-name MusicCollection \
--key-condition-expression "Artist = :a and begins_with(SongTitle, :t)" \
--expression-attribute-values '{":a":{"S":"The Beatles"}, ":t":{"S": "h"}}'`
- `aws dynamodb --endpoint-url http://localhost:8000 query --select ALL_ATTRIBUTES --table-
name MusicCollection --key-condition-expression "Artist = :a and AlbumTitle = :t" --
expression-attribute-values '{":a":{"S":"AC/DC"}, ":t":{"S": "Back in Black"}}'`
- `aws dynamodb --endpoint-url http://localhost:8000 update-table --table-name MusicCollection \
--attribute-definitions AttributeName=Artist,AttributeType=S \
AttributeName=SongTitle,AttributeType=S AttributeName=AlbumTitle,AttributeType=S \
--global-secondary-index-updates "Create={"IndexName": "album-index", "KeySchema": [{"AttributeName": "Artist", "KeyType": "HASH"}, {"AttributeName": "AlbumTitle", "KeyType": "RANGE"}], "Projection": {"ProjectionType": "INCLUDE", "NonKeyAttributes": "AlbumTitle"}, "ProvisionedThroughput": {"ReadCapacityUnits": 1, "WriteCapacityUnits": 1}}"`
- `aws dynamodb --endpoint-url http://localhost:8000 query \
--select ALL_ATTRIBUTES \
--table-name MusicCollection \`

```
--index-name album-index |
--key-condition-expression "Artist = :a and AlbumTitle = :t" |
--expression-attribute-values '{":a":{"S":"AC/DC"}, ":t": {"S": "Back in Black"}}'
```

## Phần 5. Dùng DynamoDB với Java

Cho cấu trúc tài liệu JSON về Movies

```
{
 "year": 2013,
 "title": "Rush",
 "info": {
 "directors": ["Ron Howard"],
 "release_date": "2013-09-02T00:00:00Z",
 "rating": 8.3,
 "genres": [
 "Action",
 "Biography",
 "Drama",
 "Sport"
],
 "image_url": "http://ia.media-imdb.com/images/M/MV5BMTQyMDE0MTY0OV5BMl5BanB
 "plot": "A re-creation of the merciless 1970s rivalry between Formula One r
 "rank": 2,
 "running_time_secs": 7380,
 "actors": [
 "Daniel Bruhl",
 "Chris Hemsworth",
 "Olivia Wilde"
]
 }
},
```

### 1. Thực hiện việc tạo bảng Movies với DynamoDB Local bằng Java

Thư viện

```
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
```

Tạo đối tượng AmazonDynamoDB kết nối vào Endpoint của DynamoDB Local

Tạo bảng Movies với thuộc tính khóa year (HASH), title (RANGE).

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "local"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

String tableName = "Movies";

try {
 System.out.println("Attempting to create table; please wait...");
 Table table = dynamoDB.createTable(tableName,
 Arrays.asList(new KeySchemaElement("year", KeyType.HASH), // Partition
 // key
 new KeySchemaElement("title", KeyType.RANGE)), // Sort key
 Arrays.asList(new AttributeDefinition("year", ScalarAttributeType.N),
 new AttributeDefinition("title", ScalarAttributeType.S)),
 new ProvisionedThroughput(10L, 10L));
 table.waitForActive();
 System.out.println("Success. Table status: " +
table.getDescription().getTableStatus());
}

catch (Exception e) {
 System.err.println("Unable to create table: ");
 System.err.println(e.getMessage());
}
```

Đưa dữ liệu vào bảng Movies từ tập tin JSON

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "local"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

JsonParser parser = new JsonFactory().createParser(new File("moviedata.json"));

JsonNode rootNode = new ObjectMapper().readTree(parser);
Iterator<JsonNode> iter = rootNode.iterator();

ObjectNode currentNode;

while (iter.hasNext()) {
 currentNode = (ObjectNode) iter.next();
```

```

int year = currentNode.path("year").asInt();
String title = currentNode.path("title").asText();

try {
 table.putItem(new Item().withPrimaryKey("year", year, "title",
title).withJSON("info",
 currentNode.path("info").toString()));
 System.out.println("PutItem succeeded: " + year + " " + title);

}
catch (Exception e) {
 System.err.println("Unable to add movie: " + year + " " + title);
 System.err.println(e.getMessage());
 break;
}
}
parser.close();

```

## 2. Thực hiện CRUD.

- Tạo mới, chèn

```

AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "local"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

int year = 2015;
String title = "The Big New Movie";

final Map<String, Object> infoMap = new HashMap<String, Object>();
infoMap.put("plot", "Nothing happens at all.");
infoMap.put("rating", 0);

try {
 System.out.println("Adding a new item...");
 PutItemOutcome outcome = table
 .putItem(new Item().withPrimaryKey("year", year, "title", title).withMap("info",
infoMap));

 System.out.println("PutItem succeeded:\n" + outcome.getPutItemResult());

}
catch (Exception e) {
 System.err.println("Unable to add item: " + year + " " + title);
 System.err.println(e.getMessage());
}

```

- Đọc dữ liệu

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

int year = 2015;
String title = "The Big New Movie";

GetItemSpec spec = new GetItemSpec().withPrimaryKey("year", year, "title", title);

try {
 System.out.println("Attempting to read the item...");
 Item outcome = table.getItem(spec);
 System.out.println("GetItem succeeded: " + outcome);
}

catch (Exception e) {
 System.err.println("Unable to read item: " + year + " " + title);
 System.err.println(e.getMessage());
}
```

- Cập nhật dữ liệu

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

int year = 2015;
String title = "The Big New Movie";

UpdateItemSpec updateItemSpec = new UpdateItemSpec().withPrimaryKey("year", year,
 "title", title)
 .withUpdateExpression("set info.rating = :r, info.plot=:p, info.actors=:a")
 .WithValueMap(new ValueMap().withNumber(":r", 5.5).withString(":p", "Everything
happens all at once."))
 .withList(":a", Arrays.asList("Larry", "Moe", "Curly")))
 .withReturnValues(ReturnValue.UPDATED_NEW);
```

```

try {
 System.out.println("Updating the item...");
 UpdateItemOutcome outcome = table.updateItem(updateItemSpec);
 System.out.println("UpdateItem succeeded:\n" + outcome.getItem().toJSONPretty());
}

catch (Exception e) {
 System.err.println("Unable to update item: " + year + " " + title);
 System.err.println(e.getMessage());
}

```

- Xóa

```

AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
 AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

int year = 2015;
String title = "The Big New Movie";

DeleteItemSpec deleteItemSpec = new DeleteItemSpec()
 .withPrimaryKey(new PrimaryKey("year", year, "title",
 title)).withConditionExpression("info.rating <= :val")
 .WithValueMap(new ValueMap().withNumber(":val", 5.0));

// Conditional delete (we expect this to fail)

try {
 System.out.println("Attempting a conditional delete...");
 table.deleteItem(deleteItemSpec);
 System.out.println("DeleteItem succeeded");
}
catch (Exception e) {
 System.err.println("Unable to delete item: " + year + " " + title);
 System.err.println(e.getMessage());
}

```

3. Truy vấn dữ liệu theo điều kiện.

- Truy vấn dữ liệu

```

import java.util.HashMap;
import java.util.Iterator;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;

```

```

import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.ItemCollection;
import com.amazonaws.services.dynamodbv2.document.QueryOutcome;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.spec.QuerySpec;

```

```

AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
 .withEndpointConfiguration(new
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "us-west-2"))
 .build();

DynamoDB dynamoDB = new DynamoDB(client);

Table table = dynamoDB.getTable("Movies");

HashMap<String, String> nameMap = new HashMap<String, String>();
nameMap.put("#yr", "year");

HashMap<String, Object> valueMap = new HashMap<String, Object>();
valueMap.put(":yyyy", 1985);

QuerySpec querySpec = new QuerySpec().withKeyConditionExpression("#yr =
:yyyy").withNameMap(nameMap)
 .withValueMap(valueMap);

ItemCollection<QueryOutcome> items = null;
Iterator<Item> iterator = null;
Item item = null;

try {
 System.out.println("Movies from 1985");
 items = table.query(querySpec);

 iterator = items.iterator();
 while (iterator.hasNext()) {
 item = iterator.next();
 System.out.println(item.getNumber("year") + ": " + item.getString("title"));
 }
}

catch (Exception e) {
 System.err.println("Unable to query movies from 1985");
 System.err.println(e.getMessage());
}

valueMap.put(":yyyy", 1992);
valueMap.put(":letter1", "A");
valueMap.put(":letter2", "L");

```

```

querySpec.withProjectionExpression("#yr, title, info.genres, info.actors[0]")
 .withKeyConditionExpression("#yr = :yyyy and title between :letter1 and
:letter2").withNameMap(nameMap)
 .withValueMap(valueMap);

try {
 System.out.println("Movies from 1992 - titles A-L, with genres and lead actor");
 items = table.query(querySpec);

 iterator = items.iterator();
 while (iterator.hasNext()) {
 item = iterator.next();
 System.out.println(item.getNumber("year") + ": " + item.getString("title") + " "
+ item.getMap("info"));
 }
}

} catch (Exception e) {
 System.err.println("Unable to query movies from 1992:");
 System.err.println(e.getMessage());
}
}

```

*Tham khảo phần kết nối thông qua Java Project (Console)*

B0. Cài AWS Toolkit for Eclipse

B1. Dùng Dependency và Repository cho Project

```

<dependencyManagement>
 <dependencies>
 <dependency>
 <groupId>com.amazonaws</groupId>
 <artifactId>aws-java-sdk-bom</artifactId>
 <version>1.11.245</version>
 <type>pom</type>
 <scope>import</scope>
 </dependency>
 </dependencies>
</dependencyManagement>
<dependencies>
 <dependency>
 <groupId>com.amazonaws</groupId>
 <artifactId>aws-java-sdk-dynamodb</artifactId>
 </dependency>
 <dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>3.8.1</version>
 <scope>test</scope>
 </dependency>
</dependencies>

```

## B2. Tạo bảng

Khóa chính bao gồm các thuộc tính:

- year –partition key. ScalarAttributeType là N cho số.
- title –sort key. The ScalarAttributeType là S cho chuỗi.

Cấu hình Endpoint:

```
AwsClientBuilder.EndpointConfiguration("http://localhost:8000", "local")
```

## B3. Lấy dữ liệu, đưa vào bảng.

## B4. Thực hiện các thao tác cập nhật, truy vấn dữ liệu.

### Phần 6. Bài tập DynamoDB với .NET

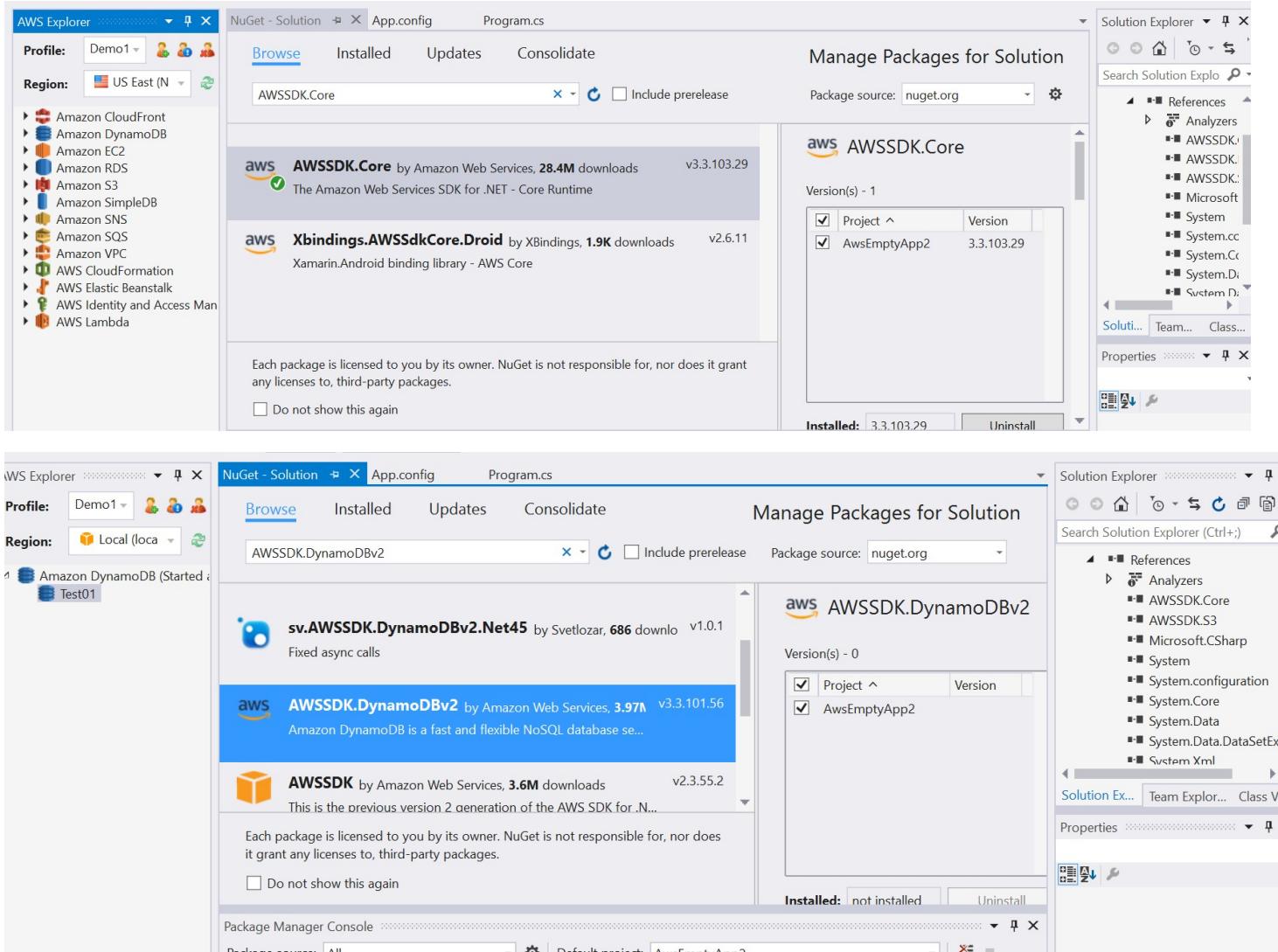
Cho cấu trúc tài liệu JSON về Movies

```
{
 "year": 2013,
 "title": "Rush",
 "info": {
 "directors": ["Ron Howard"],
 "release_date": "2013-09-02T00:00:00Z",
 "rating": 8.3,
 "genres": [
 "Action",
 "Biography",
 "Drama",
 "Sport"
],
 "image_url": "http://ia.media-imdb.com/images/M/MV5BMTQyMDE0MTY0OV5BMl5BanB
 "plot": "A re-creation of the merciless 1970s rivalry between Formula One r
 "rank": 2,
 "running_time_secs": 7380,
 "actors": [
 "Daniel Bruhl",
 "Chris Hemsworth",
 "Olivia Wilde"
]
 }
},
```

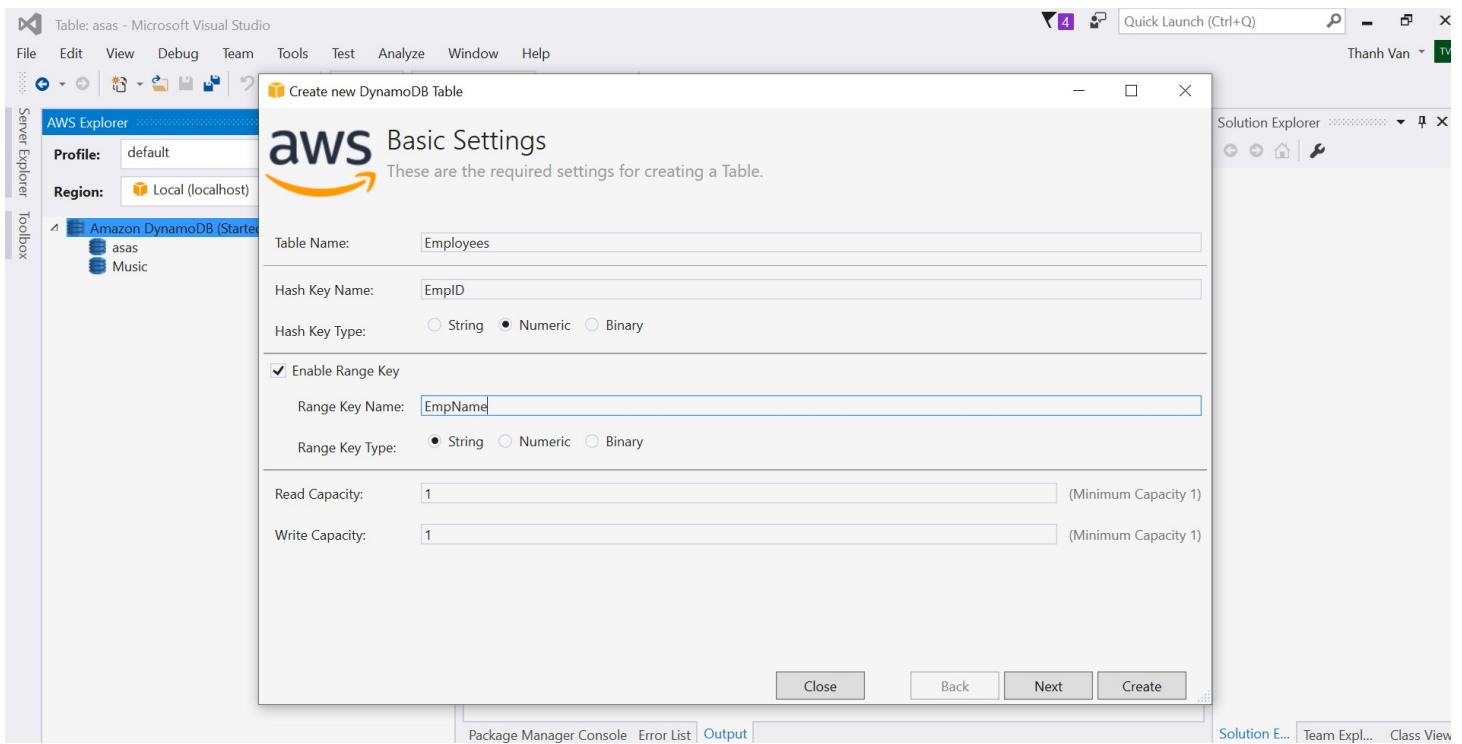
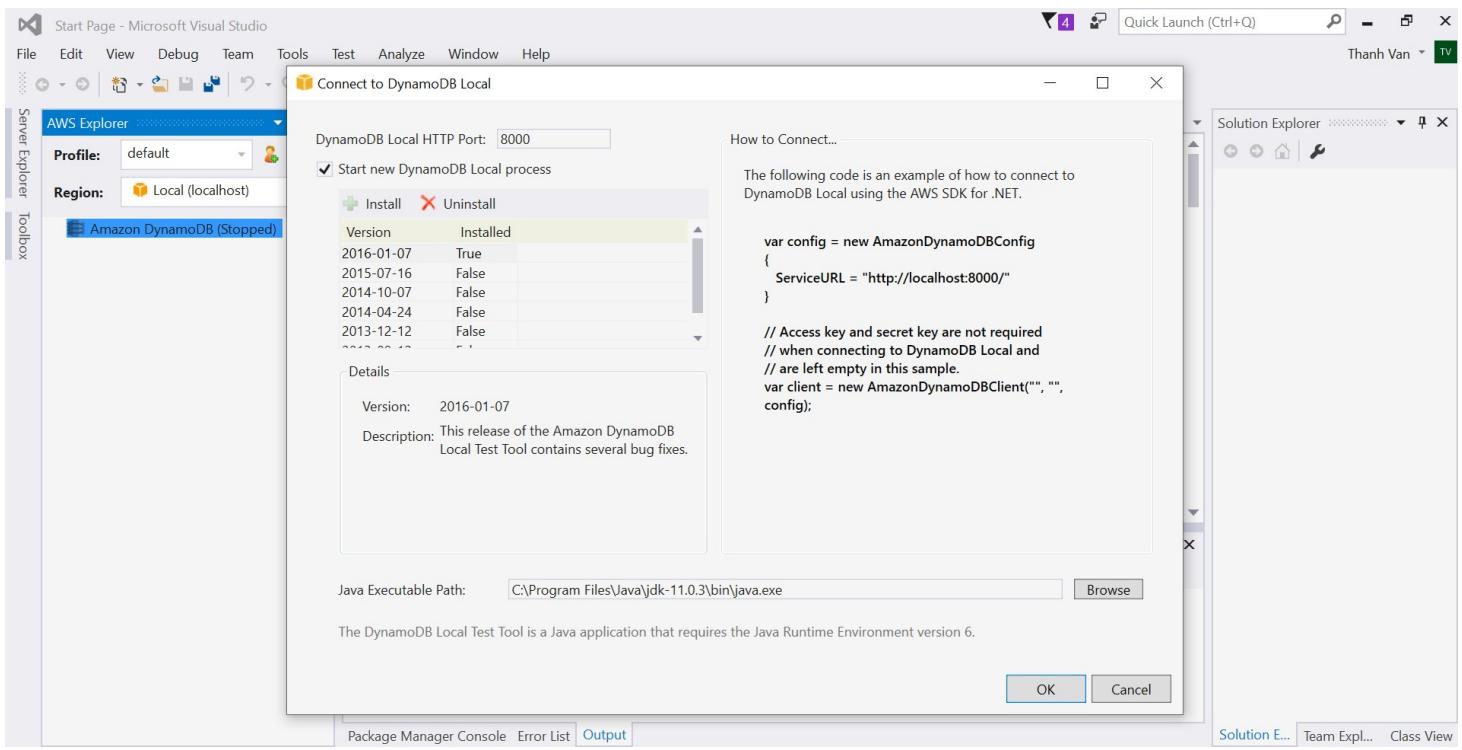
## B0. Cài đặt AWS Toolkit for Visual Studio

<https://aws.amazon.com/visualstudio/>

## B1. Tạo AWS Empty Project, dùng Nuget download các thư viện liên quan.



## B2. Dùng DynamoDB Local, service có sẵn cho Visual Studio (không dùng DynamoDB Local chạy bên ngoài)



## App.Config

```
<?xml version="1.0"?>
<configuration>
 <appSettings>
 <add key="AWSProfileName" value="default"/>
 <add key="AWSRegion" value="local" />
 </appSettings>
```

```
</configuration>
```

## Packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
 <package id="AWSSDK.Core" version="3.3.103.29" targetFramework="net452" />
 <package id="AWSSDK.DynamoDBv2" version="3.3.101.56" targetFramework="net452" />
 <package id="AWSSDK.S3" version="3.3.104.17" targetFramework="net452" />
 <package id="Newtonsoft.Json" version="12.0.2" targetFramework="net452" />
</packages>
```

## B3. Tạo Client

```
public static bool createClient(bool useDynamoDBLocal)
{
 if(useDynamoDBLocal)
 {
 operationSucceeded = false;
 operationFailed = false;

 bool localFound = false;
 try
 {
 using (var tcp_client = new TcpClient())
 {
 var result = tcp_client.BeginConnect("localhost", 8000, null, null);
 localFound = result.AsyncWaitHandle.WaitOne(3000); // Wait 3 seconds
 tcp_client.EndConnect(result);
 }
 }
 catch
 {
 localFound = false;
 }
 if(!localFound)
 {
 Console.WriteLine("\n ERROR: DynamoDB Local ");
 operationFailed = true;
 return (false);
 }

 // If DynamoDB-Local does seem to be running, so create a client
 Console.WriteLine(" -- Setting up a DynamoDB-Local client");
 AmazonDynamoDBConfig ddbConfig = new AmazonDynamoDBConfig();
 ddbConfig.ServiceURL = "http://localhost:8000";
 try { client = new AmazonDynamoDBClient(ddbConfig); }
 catch(Exception ex)
 {
 Console.WriteLine(" FAILED to create a DynamoDBLocal client; " + ex.Message);
 operationFailed = true;
 return false;
 }
 }
}
```

```

else
{
try { client = new AmazonDynamoDBClient(); }
catch(Exception ex)
{
 Console.WriteLine(" FAILED to create a DynamoDB client; " + ex.Message);
 operationFailed = true;
}
}
operationSucceeded = true;
return true;
}

```

B4. Thao tác tạo bảng với DynamoDB.

```

AmazonDynamoDBClient client = new AmazonDynamoDBClient();
string tableName = "ProductCatalog";

var request = new CreateTableRequest
{
 TableName = tableName,
 AttributeDefinitions = new List<AttributeDefinition>()
 {
 new AttributeDefinition
 {
 AttributeName = "Id",
 AttributeType = "N"
 }
 },
 KeySchema = new List<KeySchemaElement>()
 {
 new KeySchemaElement
 {
 AttributeName = "Id",
 KeyType = "HASH" //Partition key
 }
 },
 ProvisionedThroughput = new ProvisionedThroughput
 {
 ReadCapacityUnits = 10,
 WriteCapacityUnits = 5
 }
};

var response = client.CreateTable(request);

```

B5. Lấy dữ liệu, đưa vào bảng.

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.NET.03.html>

- Đọc dữ liệu từ JSON
- Đưa vào bảng Sync/Async

B5. Thực hiện các thao tác cập nhật, truy vấn dữ liệu.

```

QueryOperationConfig config = new QueryOperationConfig();
config.Filter = new QueryFilter();
config.Filter.AddCondition("year", QueryOperator.Equal, new DynamoDBEntry[] { 1992 });
 config.Filter.AddCondition("title", QueryOperator.Between, new DynamoDBEntry[] { "B",
"Hz" });
 config.AttributesToGet = new List<string> { "year", "title", "info" };
config.Select = SelectValues.SpecificAttributes;

```

Ngoài ra tham khảo:

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/CodeSamples.DotNet.html>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/LowLevelDotNetWorkingWithTables.html>

Dùng Project mẫu với AWS Toolkit for .NET.

Phản 7. Cập nhật nhật ký thực hiện đồ án

Phản 8. Lập kế hoạch kiểm thử cho ứng dụng (Kiểm thử chức năng, Kiểm thử giao diện, Kiểm thử chu trình nghiệp vụ, ...)

Tham khảo mẫu, hoặc dùng mẫu tương đương. Phản kê hoạch kiểm thử để trong phụ lục file báo cáo.

## TEST PLAN

<TÊN ĐỀ TÀI>

### I. Giới thiệu

#### 1.1. Mục tiêu

Mục tiêu của kế hoạch kiểm thử.

- <Lập kế hoạch cụ thể kiểm thử cho ứng dụng XX, đảm bảo việc kế hoạch kiểm thử được thực hiện đúng kế hoạch và đầy đủ, phân chia công việc trong nhóm phù hợp và hợp lý>
- <Nâng cao mức độ tiện lợi của ứng dụng>
- <Đảm bảo ứng dụng không còn lỗi khi đưa ra cho người sử dụng>

## 1.2. Phân tích ứng dụng

- *Đối tượng sử dụng Website?*
- *Mục tiêu của Website dùng để thực hiện các chức năng gì?*
- *Website hoạt động như thế nào?*
- *Phần mềm/phần cứng/thiết bị mà ứng dụng sử dụng?*

## 1.3. Phạm vi kiểm thử

- *Theo yêu cầu ứng dụng, để tài XX chỉ tập trung vào kiểm thử tất cả các chức năng và giao diện bên ngoài của website (trên các trình duyệt khác nhau như Firefox, Chrome, Microsoft Edge, Safari).*
- *Không thực hiện kiểm thử mức độ hiệu suất ứng dụng, logic cơ sở dữ liệu.*

## 1.4. Các ràng buộc về quy trình kiểm thử

- *Môi trường kiểm thử, các điều kiện liên quan.*

TT	Tài nguyên	Mô tả
1	Máy chủ	<i>Server cài đặt ứng dụng Web sẽ được kiểm thử. Database Server Application Server</i>
2	Công cụ	<i>Mô tả công cụ sử dụng để kiểm thử.</i>
3	Mạng	<i>Mạng LAN hoặc Internet</i>
4	Máy tính Client	<i>Mô tả môi trường mà máy Client dùng để chạy ứng dụng.</i>
5	Các tài nguyên khác	<i>Tài khoản AWS (AWS Free tier hoặc AWS Starter)</i>

Bảng 1. Môi trường kiểm thử

- *Các ràng buộc về tài nguyên, lịch trình, công cụ.*

## 1.5. Rủi ro

Rủi ro	Giải pháp

<i>Thành viên trong nhóm thiếu các kỹ năng cần thiết để kiểm thử website ứng dụng.</i>	<i>Tự học hỏi, tìm hiểu thêm, tham gia các khóa học liên quan đến kiểm thử để nâng cao kỹ năng của các thành viên nhóm</i>
<i>Lịch trình thực hiện đề tài trong thời gian ngắn, công việc nhiều, công nghệ tiếp cận là mới; rất khó để hoàn thành ứng dụng đúng hạn</i>	<i>Đặt mức độ ưu tiên kiểm thử cho từng hoạt động kiểm thử.</i>
<i>Nhóm trưởng có kỹ năng quản lý kém, không điều động được thành viên nhóm</i>	<i>Nhóm trưởng tham gia các khóa học liên quan đến quản lý.</i>
<i>Thiếu hợp tác giữa các thành viên ảnh hưởng tiêu cực đến năng suất làm việc của từng thành viên</i>	<i>Hợp nhóm, phân chia lại chức năng phù hợp với khả năng của từng thành viên.</i>
<i>Ước lượng chi phí cho ứng dụng chưa hợp lý và vượt chi phí.</i>	<i>Xác định phạm vi rõ ràng khi bắt đầu ứng dụng, lưu ý khi lập kế hoạch dự án và phải theo dõi tiến độ trong quá trình làm.</i>

Bảng 2. Các rủi ro

## II. Yêu cầu kiểm thử

### 2.1. Danh sách các chức năng kiểm thử

<i>STT</i>	<i>Chức năng</i>	<i>Mô tả văn tắt chức năng (outline)</i>	<i>Ước lượng số lượng tình huống kiểm kiểm thử (test case).</i>	<i>Ghi chú</i>
<i>1.</i>	<i>Các chức năng ở mức tổng thể</i>		<i>5</i>	
<i>2.</i>				
<i>3.</i>				
<i>4.</i>				

Bảng 3. Danh sách các chức năng kiểm thử

### 2.2. Điều kiện chấp nhận

*Danh sách các tiêu chí để xác định mức chất lượng kiểm thử là đủ để chuyển sang giai đoạn thử nghiệm tiếp theo*

- *Phạm vi bao phủ của Test (Test coverage)*

- *Successful Test coverage*
- *Số lượng các trường hợp kiểm thử (Đơn vị / Tích hợp / Các trường hợp thử nghiệm hệ thống)*
- *Số lượng lỗi/Trọng số lỗi*

### III. Kỹ thuật kiểm thử

#### 3.1. Kiểm thử đơn vị

*<Phân công công việc cho từng thành viên trong nhóm, thành viên thực hiện các unit nào, kiểm thử các unit đó>*

#### 3.2. Kiểm thử Module/chức năng

<b>Mục tiêu</b>	<i>&lt;Đảm bảo chức năng kiểm thử với mục tiêu thích hợp, bao gồm dữ liệu đầu vào, navigation, quá trình xử lý và kết quả nhận được&gt;</i>
<b>Kỹ thuật</b>	<i>&lt;Thực thi mỗi use case, luồng hoạt động cho use-case , hoặc chức năng, dùng dữ liệu đúng, dữ liệu không đúng để xác định: - Kết quả mong muốn khi dữ liệu đưa vào là đúng - Các thông báo lỗi, cảnh báo hiển thị khi dữ liệu không chính xác đưa vào. - Quy tắc nghiệp vụ áp dụng cho trường hợp test. - Sử dụng các công cụ kiểm thử (Test tool) ...&gt;</i>
<b>Tiêu chí hoàn thành chức năng</b>	<i>- &lt;Tất cả các kế hoạch kiểm thử cho chức năng được thực hiện - Chính sửa các lỗi đã phát hiện. &gt;</i>
<b>Điều kiện đặc biệt.</b>	<i>&lt;Xác định hoặc mô tả các mục hoặc vấn đề (bên trong hoặc bên ngoài) mà có ảnh hưởng đến việc triển khai và thực hiện kiểm thử chức năng &gt;</i>

Bảng 4. Kiểm thử chức năng

#### 3.3. Kiểm thử giao diện

<b>Mục tiêu</b>	<i>&lt;Kiểm thử các vấn đề: Di chuyển từ cửa sổ giao diện này sang giao diện khác, từ field này sang field khác dùng phím tab, di chuyển chuột hoặc các phím tổ hợp Đổi tượng cửa sổ, menu, kích cỡ, vị trí, trạng thái. &gt;</i>
<b>Kỹ thuật</b>	<i>&lt; Tạo hoặc sửa đổi các kiểm thử cho mỗi cửa sổ để xác minh điều hướng và trạng thái đối tượng thích hợp cho từng cửa sổ ứng dụng &gt;</i>

<b>Tiêu chí hoàn thành chức năng</b>	<Each window successfully verified to remain consistent with benchmark version or within acceptable standard>
<b>Điều kiện đặc biệt</b>	<Not all properties for custom and third party objects can be accessed.>

Bảng 5. Kiểm thử giao diện ứng dụng

### 3.4. Kiểm thử hệ thống

<Kiểm thử hệ thống thuộc loại Black Box Testing

- *Kiểm thử hệ thống được thực hiện khi hệ thống đã được tích hợp đầy đủ các chức năng bao gồm cả các thiết bị bên ngoài, kiểm thử các thành phần tương tác với nhau và với toàn bộ hệ thống.*
- *Tạo kịch bản kiểm thử cuối cùng.*
  - *Kiểm thử mọi đầu vào và đầu ra mong muốn.*
- *Kiểm thử trải nghiệm của người sử dụng với ứng dụng.*

## IV. Kế hoạch nguồn nhân lực thực hiện kiểm thử cho ứng dụng

<b>STT</b>	<b>Thành viên</b>	<b>Vai trò</b>	<b>Nhiệm vụ</b>	<b>Ước lượng thời gian thực hiện</b>
1		<i>Tên Test Manager</i>	<i>Quản lý toàn bộ việc kiểm thử ứng dụng . Xác định tài nguyên phù hợp cho việc kiểm thử.</i>	
2		<i>Tester</i>	<i>Thực hiện các kiểm thử, log kết quả, cáo cáo các lỗi.</i>	
3		<i>Developer thực hiện Test</i>	<i>Thực hiện các test cases, test program, test suite...</i>	
4		<i>Test Administrator</i>	<i>Xây dựng và đảm bảo môi trường kiểm thử, tài nguyên được quản lý và duy trì.</i>	

			<i>Hỗ trợ Tester sử dụng môi trường kiểm thử để thực hiện kiểm thử.</i>	
5			<p><i>Thành viên SQA</i></p> <p><i>Phụ trách đảm bảo chất lượng cho ứng dụng phần mềm/website.</i></p> <p><i>Kiểm thử để xác nhận xem quy trình kiểm thử có đáp ứng các yêu cầu đã được xác định hay không.</i></p>	

*Bảng 6 Kế hoạch nhân sự*

# BÀI TẬP TUẦN 05. 06. 07. MÔN CÔNG NGHỆ MỚI TRONG PTUDCNTT

Mục tiêu:

- Lập trình ứng dụng với Node.js
- Cập nhật nhật ký thực hiện đồ án.
- Chính sửa nội dung báo cáo phần phân tích thiết kế.

Yêu cầu:

- Đã có tài khoản AWS.
- Dùng Visual Studio Code hoặc WebStorm hoặc Eclipse with Node.js plug-in
- Sử dụng Postman cho ứng dụng RESTful API

## Phần 1. Cài đặt Node.js, cấu hình các biến môi trường.

B1. Cài Node.js <https://nodejs.org/en/>



### Downloads

Latest LTS Version: 10.16.3 (includes npm 6.9.0)

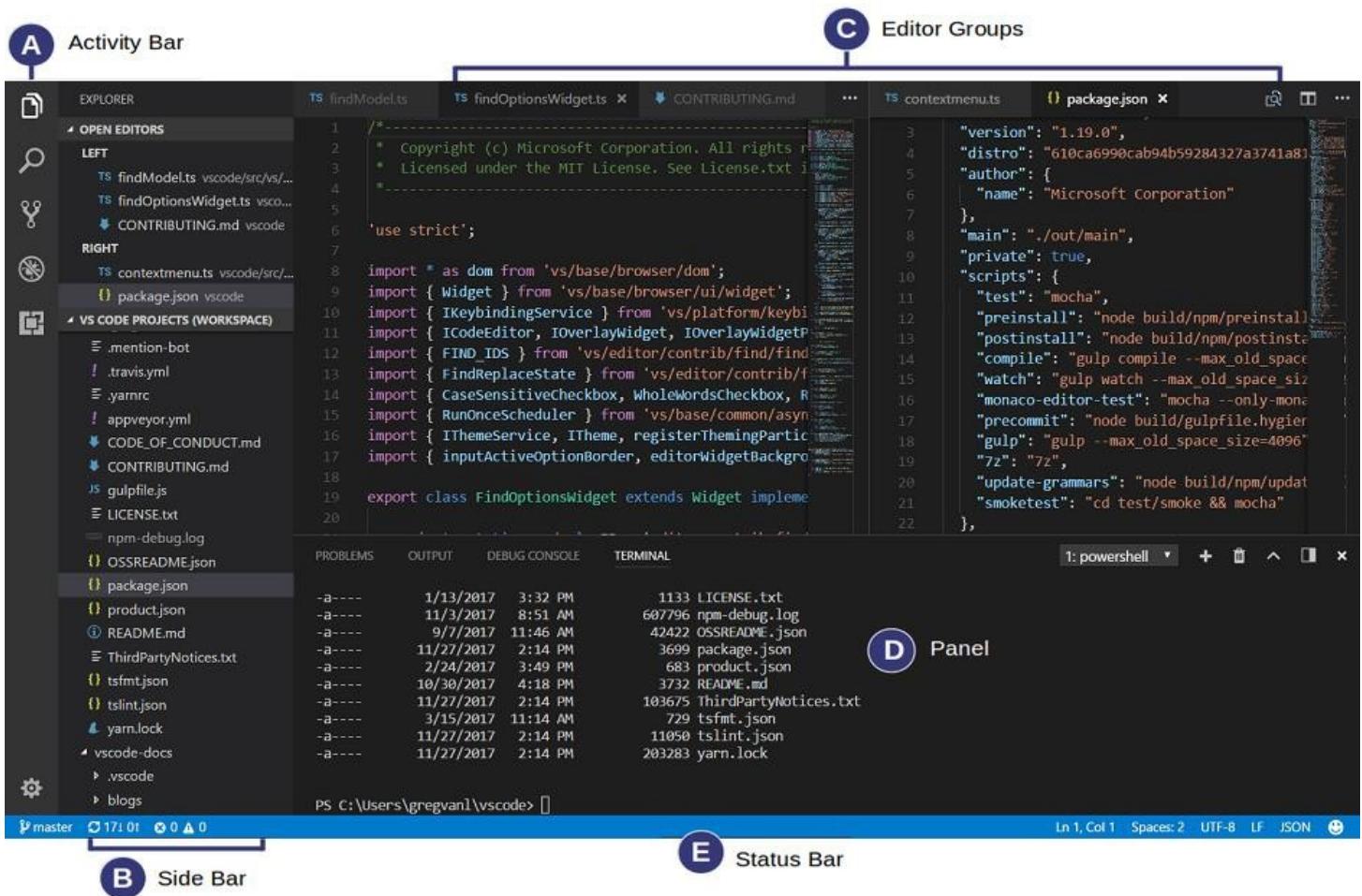
Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Windows Installer (.msi)	32-bit	64-bit
node-v10.16.3-x64.msi		

Windows Binary (.zip)	32-bit	64-bit
node-v10.16.3.zip		

B2. Kiểm tra các biến môi trường cho node, npm

B3. Visual Studio Code IDE cho Node.js



#### B4. Tạo Project cho Node.js

Tập tin package.json (dùng `npm init`, mặc định dùng `npm init --yes`)

- Các thông tin liên quan đến Project
- Liệt kê danh sách các packages mà project sử dụng.
- Gồm version cho từng package, dùng semantic versioning rules (<https://docs.npmjs.com/about-semantic-versioning>)

Code status	Stage	Rule	Example version
First release	New product	Start with 1.0.0	1.0.0
Backward compatible bug fixes	Patch release	Increment the third digit	1.0.1

<b>Code status</b>	<b>Stage</b>	<b>Rule</b>	<b>Example version</b>
Backward compatible new features	Minor release	Increment the middle digit and reset last digit to zero	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit and reset middle and last digits to zero	2.0.0

For example, to specify acceptable version ranges up to 1.0.4, use the following syntax:

- Patch releases: **1.0** or **1.0.x** or **~1.0.4**
- Minor releases: **1** or **1.x** or **^1.0.4**
- Major releases: **\*** or **x**

*Phần 2. Để xuất được 3 công nghệ phù hợp với ứng dụng quản lý thông tin sử dụng các công nghệ của AWS*

- Compute (EC2, Elastic Beanstalk)
- Storage (S3, EFS)
- Database (DynamoDB, RDS)
- Networking and Content Delivery (Route53)
- Security, Identity, & Compliance (IAM)

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/nodejs-dynamodb-tutorial.html>

### Phân 3. Thực hiện các ứng dụng với Node.js

#### Bài 1. Bài tập căn bản, xử lý sự kiện với Node.js

- Blocking và non-blocking

```
var fs = require("fs");

var data = fs.readFileSync("companies.txt");

console.log(data.toString());
console.log("Another task!");
```

```
var fs = require("fs");

fs.readFile("companies.txt", function (err, data) {
 if (err) return console.error(err);
 console.log(data.toString());
});

console.log("Another task!");
```

- Event Loop

```
var events = require("events");
// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();

// Create an Event Handler
var connectHandler = function connected() {
 console.log("Connect successfully!");

 // Fire/active an event (Kích hoạt sự kiện data_received)
 eventEmitter.emit("data_received");
}

// Assign an event 'connection' with Event Handler
eventEmitter.on("connection", connectHandler);

// Assign the 'data_received' event with anonymous function
eventEmitter.on("data_received", function(){
 console.log("Data received successfully!");
});

// Fire/active the event 'connection'
eventEmitter.emit("connection");
```

```
console.log("Finish!");
```

- **EventEmitter** là lớp trong Node.js có chức năng chính là phát sinh sự kiện.

Phương thức của EventEmitter:

1	<b>addListener(event, listener)</b>
2	<b>on(event, listener)</b>
3	<b>once(event, listener)</b> Sự kiện kích hoạt chỉ 1 lần
4	<b>removeListener(event, listener)</b>
5	<b>removeAllListeners([event])</b> Xóa tất cả Listener của một sự kiện
6	<b>setMaxListeners(n), n=0: không giới hạn số lượng Listener.</b>
7	<b>listeners(event)</b> Return mảng bao gồm danh sách cácListener cho một event.
8	<b>emit(event, [arg1], [arg2], [...])</b> Thực thi từng Listener với các tham số đã cho. Return true nếu sự kiện có các Listener, và false nếu không có.

```
var events = require("events");
var eventEmitter = new events.EventEmitter();

// listener #1
var listner1 = function listner1() {
 console.log("listner1 is executed.");
}

// listener #2
var listner2 = function listner2() {
 console.log("listner2 is executed.");
}

// Assign the 'connection' event with listner1 function
eventEmitter.addListener("connection", listner1);

// Assign the 'connection' event with listner2 function
eventEmitter.on("connection", listner2);

var eventListeners = require("events").EventEmitter.ListenerCount(eventEmitter, "connection");
console.log(eventListeners + " Event Listener is listening to " dang lang nghe su kien
'connection');

// Fire the connection event
```

```

eventEmitter.emit("connection");

// Remove the binding of listner1 function
eventEmitter.removeListener("connection", listner1);
console.log("At the current time, Listner1 did not listen to the events.");

// Fire the connection event
eventEmitter.emit("connection");

eventListeners = require("events").EventEmitter.ListenerCount(eventEmitter, "connection");
console.log(eventListeners + " Event Listener is listening to 'connection' event.");

console.log("Finish!");

```

- Stream

- Readable – Stream dùng để đọc
- Writable – Stream dùng để ghi
- Duplex – Stream dùng để đọc và ghi
- Transform – Tương tự Duplex Stream, tuy nhiên output được tính toán trên dữ liệu input.

```

var fs = require("fs");
var data = "";

// Create a Readable Stream with encoding UTF8
var readerStream = fs.createReadStream("companies.txt");
readerStream.setEncoding("UTF8");

// Handle the events related to Stream (event: data, end, and error)
readerStream.on("data", function(chunk) {
 data += chunk;
});

readerStream.on("end",function(){
 console.log(data);
});

readerStream.on("error", function(err){
 console.log(err.stack);
});

console.log("Finish!");

```

```

var fs = require("fs");
var data = "This is stream Tiếng Việt Unicode";

```

```

// Create a Writable Stream
var writerStream = fs.createWriteStream("output.txt");
writerStream.write(data,"UTF8");
writerStream.end();

// Handle the events related to Stream --> finish, and error
writerStream.on("finish", function() {
 console.log("Finish writing.");
});

writerStream.on("error", function(err){
 console.log(err.stack);
});

console.log("Finish!");

```

- Đọc ghi tập tin

```

var fs = require("fs");

// Asynchronous method
fs.readFile("input.txt", function (err, data) {
 if (err) {
 return console.error(err);
 }
 console.log("Using Async method: " + data.toString());
});

// Synchronous method
var data = fs.readFileSync("input.txt");
console.log("Using Sync method: " + data.toString());

console.log("Finish!");

```

## Bài 2. Tạo HTTP Server

- Tạo Simple HTTP Server

```

var http = require("http");
http.createServer(function (req, res) {
 res.writeHead(200, {"Content-Type": "text/html"});
 res.write("Hello World!");
 res.end();
}

```

```
}).listen(8080);
console.log("Server is running at port 8080");
```

- Xử lý với Query String

```
var http = require("http");
var url = require("url");

http.createServer(function (req, res) {
 res.writeHead(200, {"Content-Type": "text/html"});
 var q = url.parse(req.url, true).query;
 var txt = q.year + " " + q.month;
 res.end(txt);
}).listen(8080);

console.log("Server is running at port 8080");
```

- Server HTTP ở mức tổng quát

```
var http = require("http");
var fs = require("fs");
var url = require("url");

// Create a server
http.createServer(function (request, response) {
 // Parse the request containing file name
 var pathname = url.parse(request.url).pathname;

 // Print the name of the file for which request is made.
 console.log("Request for " + pathname + " received.");

 // Read the requested file content from file system
 fs.readFile(pathname.substr(1), function (err, data) {
 if (err) {
 console.log(err);

 // HTTP Status: 404 : NOT FOUND
 // Content Type: text/plain
 response.writeHead(404, {"Content-Type": "text/html"});
 } else {
 //Page found
 // HTTP Status: 200 : OK
 // Content Type: text/plain
 response.writeHead(200, {"Content-Type": "text/html"});

 // Write the content of the file to response body
 }
 });
});
```

```

 response.write(data.toString());
 }

 // Send the response body
 response.end();
});

}).listen(8081);

console.log("Server running at http://127.0.0.1:8081/");

```

- Tạo Client truy cập vào Server

```

var http = require("http");

// Options to be used by request
var options = {
 host: "localhost",
 port: "8081",
 path: "/index.htm"
};

// Callback function is used to deal with response
var callback = function(response) {
 // Continuously update stream with data
 var body = "";
 response.on("data", function(data) {
 body += data;
 });

 response.on("end", function() {
 // Data received completely.
 console.log(body);
 });
}

// Make a request to the server
var req = http.request(options, callback);
req.end();

```

### Bài 3. Express Framework trong Node.js

Các module kèm theo với Express:

**body-parser** – dùng để xử lí JSON, text, mã hóa URL.

**cookie-parser** – Phân tích cookie và chuyển đến các req.cookies

**multer** – xử lý phần multipart/form-data.

- Routing trong Express

```
var express = require("express");
var app = express();

// get()
app.get("/", function (req, res) {
 console.log("GET Request Homepage");
 res.send("Hello GET");
})

// post()
app.post("/", function (req, res) {
 console.log("POST Request Homepage");
 res.send("Hello POST");
})

// delete() DELETE Request v
app.delete("/del_user", function (req, res) {
 console.log("DELETE Request: /del_user");
 res.send("Hello DELETE");
})

// GET Request /list_user page.
app.get("/list_user", function (req, res) {
 console.log("GET Request: /list_user");
 res.send("Page Listing");
})

// GET Request theo pattern ab**cd
app.get("/ab*cd", function(req, res) {
 console.log("GET request: /ab*cd");
 res.send("Page Pattern Match");
})

var server = app.listen(8081, function () {
 var host = server.address().address;
 var port = server.address().port;

 console.log("Server is running at", host, port);
})
```

## Bài 4. Quản lý Session với Node.js

### B1. Template giao diện

```
<html>
<head>
<title>Session Management in NodeJS using Node and Express</title>
<script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
 var email,pass;
 $("#submit").click(function(){
 email=$("#email").val();
 pass=$("#password").val();
 /*
 * Perform some validation here.
 */
 $.post("/login",{email:email,pass:pass},function(data){
 if(data=="done"){
 window.location.href="/admin";
 }
 });
 });
});
</script>
</head>
<body>
<input type="text" size="40" placeholder="Type your email" id="email">

<input type="password" size="40" placeholder="Type your password" id="password">

<input type="button" value="Submit" id="submit">
</body>
</html>
```

### B2. Server xử lý Session

```
express = require('express');
session = require('express-session');
bodyParser = require('body-parser');
router = express.Router();
app = express();

app.use(session({secret: "sshhhhh", saveUninitialized: true, resave: true}));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(express.static(__dirname + '/views'));
```

```

var sess; // global session, NOT recommended

router.get('/',(req,res) => {
 sess = req.session;
 if(sess.email) {
 return res.redirect('/admin');
 }
 res.sendFile("index.html");
});

router.post('/login',(req,res) => {
 sess = req.session;
 sess.email = req.body.email;
 res.end("done");
});

router.get('/admin',(req,res) => {
 sess = req.session;
 if(sess.email) {
 res.write(`<h1>Hello ${sess.email} </h1>
`);
 res.end(`Logout`);
 }
 else {
 res.write(`<h1>Please login first.</h1>`);
 res.end(`Log in`);
 }
});

router.get('/logout',(req,res) => {
 req.session.destroy((err) => {
 if(err) {
 return console.log(err);
 }
 res.redirect('/');
 });
});

app.use("/", router);

app.listen(process.env.PORT || 3000,() => {
 console.log(`App Started on PORT ${process.env.PORT || 3000}`);
});

```

## Bài 5. Upload tập tin với Node.js

### B1. Tạo file HTML template

```
<html>
 <head>
 <title>File upload Node.</title>
 </head>
 <body>
 <form id="uploadForm"
 enctype="multipart/form-data"
 action="/api/photo"
 method="post">
 <input type="file" name="userPhoto" />
 <input type="submit" value="Upload Image" name="submit">

 </form>
 </body>
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
 <script
src="http://cdnjs.cloudflare.com/ajax/libs/jquery.form/3.51/jquery.form.min.js"></script>
 <script>
$(document).ready(function() {
 $("#uploadForm").submit(function() {
 $("#status").empty().text("File is uploading...");
 $(this).ajaxSubmit({
 error: function(xhr) {
 status("Error: " + xhr.status);
 },
 success: function(response) {
 console.log(response)
 $("#status").empty().text(response);
 }
 });

 return false;
 });
});
 </script>
</html>
```

### B2. Tạo tập tin server.js xử lý get/post cho

```

var express = require("express");
var multer = require("multer"); // Multer is a node.js middleware for handling
// multipart/form-data, which is primarily used for uploading files.

var app = express();

var storage = multer.diskStorage({
 destination: function (req, file, callback) {
 callback(null, './uploads');
 },
 filename: function (req, file, callback) {
 callback(null, file.fieldname + '-' + Date.now());
 }
});

var upload = multer({ storage : storage}).single("userPhoto");
app.get('/', function(req,res){
 res.sendFile(__dirname + "/index.html");
});

app.post('/api/photo',function(req,res){
 upload(req,res,function(err) {
 if(err) {
 return res.end("Error uploading file.");
 }
 res.end("File is uploaded");
 });
});

app.listen(3000,function(){
 console.log("Working on port 3000");
});

```

## Bài 6. RESTful API trong Node.js (dùng Express)

```

var express = require("express");
var app = express();
var fs = require("fs");
var port = process.env.PORT;

app.get('/products', function (req, res) {
 fs.readFile(__dirname + "/data/" + "products.json", "utf8", function (err, data) {
 console.log(data);
 res.end(data);
 });
})

```

```

var server = app.listen(port, function () {
 var host = server.address().address
 var port = server.address().port

 console.log("Example app listening at http://%s:%s", host, port)
})

```

## Bài 7. Node.js với DynamoDB Local

Tham khảo hướng dẫn với AWS JavaScript SDK:

<https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/>

Tạo bảng với khóa HASH và RANGE

```

var AWS = require("aws-sdk");

AWS.config.update({
 region: "us-west-2",
 endpoint: "http://localhost:8000"
});

var dynamodb = new AWS.DynamoDB();

var params = {
 TableName : "Movies",
 KeySchema: [
 { AttributeName: "year", KeyType: "HASH"}, //Partition key
 { AttributeName: "title", KeyType: "RANGE" } //Sort key
],
 AttributeDefinitions: [
 { AttributeName: "year", AttributeType: "N" },
 { AttributeName: "title", AttributeType: "S" }
],
 ProvisionedThroughput: {
 ReadCapacityUnits: 10,
 WriteCapacityUnits: 10
 }
};

dynamodb.createTable(params, function(err, data) {
 if (err) {
 console.error("Unable to create table. Error JSON:", JSON.stringify(err, null, 2));
 } else {

```

```

 console.log("Created table. Table description JSON:", JSON.stringify(data, null, 2));
 }
});
```

*Dưa dữ liệu vào bảng*

```

var AWS = require("aws-sdk");
var fs = require("fs");

DATA_FILE = "./data/moviedata.json";

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://127.0.0.1:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient();

console.log("Importing movies into DynamoDB...");

var allMovies = JSON.parse(fs.readFileSync(DATA_FILE, "utf8"));

allMovies.forEach(function(movie) {
 var params = {
 TableName: "Movies",
 Item: {
 year: movie.year,
 title: movie.title,
 info: movie.info
 }
 };

 docClient.put(params, function(err, data) {
 if (err) {
 console.error("Unable to add movie", movie.title,
 ". Error JSON:", JSON.stringify(err, null, 2));
 } else {
 console.log("Put item successful:", movie.title);
 }
 })
})
```

*Thêm Item và Table*

```
var AWS = require("aws-sdk");
```

```

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://localhost:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient();

var table = "Movies";
var year = 2015;
var title = "The Big New Movie";

var params = {
 TableName:table,
 Item: {
 year: year,
 title: title,
 info: {
 plot: "Nothing happens at all.",
 rating: 0
 }
 }
};

console.log("Adding a new item...");

docClient.put(params, function(err, data) {
 if (err) {
 console.error("Unable to add item. Error JSON:", JSON.stringify(err, null, 2));
 } else {
 console.log("Added item:", JSON.stringify(data, null, 2));
 }
});

```

Truy vấn dữ liệu (Query phải có PrimaryKey (HASH và RANGE Key))

```

var AWS = require("aws-sdk");

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://127.0.0.1:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient();

```

```

console.log("Querying for movies from 1985.");
var params = {
 TableName : "Movies",
 KeyConditionExpression: "#yr = :yyyy",
 ExpressionAttributeNames:{
 "#yr": "year" // 'year' is an AWS reserved name
 },
 ExpressionAttributeValues: {
 ":yyyy": 1985 // because cannot use literals in any expression
 }
};

docClient.query(params, function(err, data) {
 if (err) {
 console.error("Unable to query. Error:", JSON.stringify(err, null, 2));
 } else {
 console.log("Query succeeded.");
 data.Items.forEach(function(item) {
 console.log(" -", item.year + ": " + item.title);
 //console.log("JSON:", JSON.stringify(item, null, 2));
 });
 }
});

```

*Scan dữ liệu (Không cần Primary Key)*

```

var AWS = require("aws-sdk");

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://localhost:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient();

var params = {
 TableName: "Movies",
 ProjectionExpression: "#yr, title, info.rating",
 FilterExpression: "#yr between :start_yr and :end_yr",
 ExpressionAttributeNames: {
 "#yr": "year",
 },
 ExpressionAttributeValues: {

```

```

 ":start_yr": 1950,
 ":end_yr": 1959
 }
};

console.log("Scanning Movies table...");

docClient.scan(params, onScan);

function onScan(err, data) {
 if (err) {
 console.error("Unable to scan the table. Error JSON:", JSON.stringify(err, null, 2));
 } else {
 // print all the movies
 console.log("Scan succeeded.");
 data.Items.forEach(function(movie) {
 console.log(
 movie.year + ":",
 movie.title, "- rating:", movie.info.rating);
 });

 // continue scanning if we have more movies, because
 // scan can retrieve a maximum of 1MB of data
 if (typeof data.LastEvaluatedKey != "undefined") {
 console.log("Scanning for more...");
 params.ExclusiveStartKey = data.LastEvaluatedKey;
 docClient.scan(params, onScan);
 }
 }
}

```

## Cập nhật dữ liệu

```

var AWS = require("aws-sdk");

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://localhost:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient()

var table = "Movies";
var year = 2015;
var title = "The Big New Movie";

```

```

// Update the item, unconditionally
var params = {
 TableName:table,
 Key:{
 year: year,
 title: title
 },
 UpdateExpression: "set info.rating = :r, info.plot=:p, info.actors=:a",
 ExpressionAttributeValues:{
 ":r": 5.5,
 ":p": "Everything happens all at once.",
 ":a": ["Larry", "Moe", "Curly"]
 },
 ReturnValues: "UPDATED_NEW"
};

console.log("Updating the item...");

docClient.update(params, function(err, data) {
 if (err) {
 console.error("Unable to update item. Error JSON:", JSON.stringify(err, null, 2));
 } else {
 console.log("UpdateItem succeeded:", JSON.stringify(data, null, 2));
 }
});

```

Xóa dữ liệu

```

var AWS = require("aws-sdk");

AWS.config.update({
 region: "us-east-1",
 endpoint: "http://localhost:8000"
 //endpoint: "https://dynamodb.us-east-1.amazonaws.com"
});

var docClient = new AWS.DynamoDB.DocumentClient();

var table = "Movies";
var year = 2015;
var title = "The Big New Movie";

var params = {
 TableName: table,
 Key:{

```

```

 year: year,
 title: title
},
/*
ConditionExpression: "info.rating <= :val",
ExpressionAttributeValues: {
 ":val": 5.0
}
*/
};

console.log("Attempting a conditional delete...");

docClient.delete(params, function(err, data) {
 if (err) {
 console.error("Unable to delete item. Error JSON:", JSON.stringify(err, null, 2));
 } else {
 console.log("DeleteItem succeeded:", JSON.stringify(data, null, 2));
 }
});

```

## Bài 8. Bài tập Bookstore với DynamoDB

- Cho phép bán sách trực tuyến (hiển thị thông tin sách)
  - Cho phép tìm kiếm dữ liệu theo tên, tác giả. Tìm kiếm nâng cao với nhiều điều kiện.
  - <Cho phép Signin dùng Facebook, Google, and Amazon accounts>.
- Tham khảo phần JavaScript:  
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>  
<http://www.ecma-international.org/ecma-262/10.0/index.html>

Template literals (Template strings)

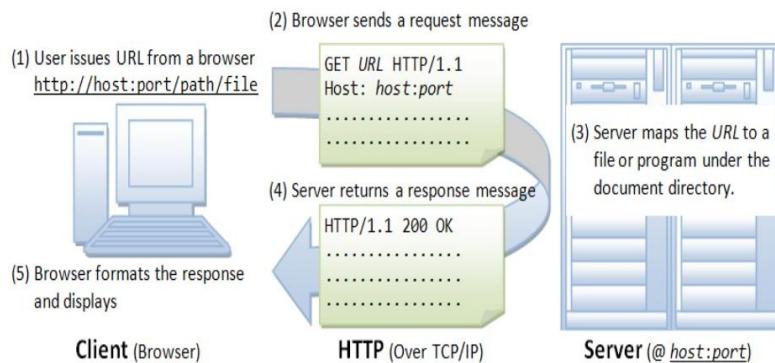
Cú pháp `Text \${javascript\_expression} other text...`

```
let firstName = "Peter";
let lastName = "Parker";
```

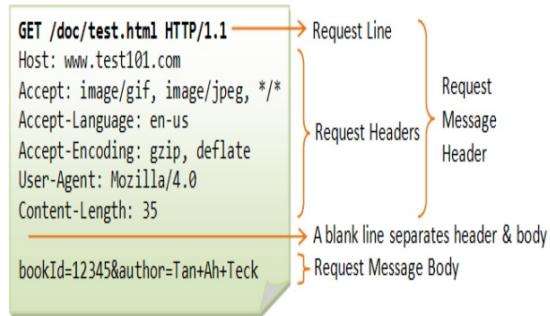
```
console.log(`Hello everyone! My name is ${firstName} ${lastName}.`);
```

Khai báo biến dùng **const** và **let** (thay vì dùng **var**)

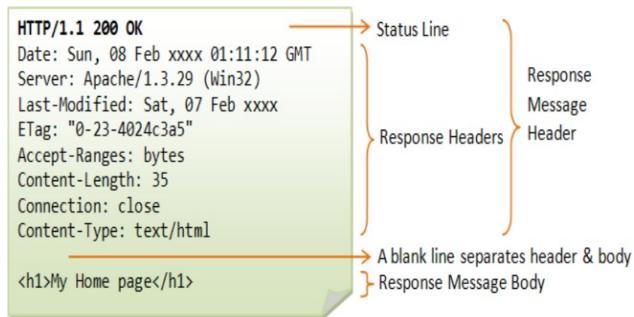
- HTTP Request/Response



- HTTP Request Message Format



## ▪ HTTP Response Message Format



## ▪ Http Response Status Code

- 100 Continue: The server received the request and is in the process of giving the response.
- **200 OK:** The request is fulfilled.
- 301 Move Permanently: The resource requested for has been permanently moved to a new location. The URL of the new location is given in the response header called Location. The client should issue a new request to the new location. Application should update all references to this new location.
- 400 Bad Request: Server could not interpret or understand the request, probably syntax error in the request message.

GET /index.html HTTTTTTP/1.0

GET test.html HTTP/1.0

- 401 Authentication Required: The requested resource is protected, and requires client's credential (username/password). The client should re-submit the request with his credential (username/password).
- 403 Forbidden: Server refuses to supply the resource, regardless of identity of client.
- 404 Not Found: The requested resource cannot be found in the server. GET /t.html HTTP/1.0
- 405 Method Not Allowed: The request method used, e.g., POST, PUT, DELETE, is a valid method. However, the server does not allow that method for the resource requested.
- 408 Request Timeout.

- 500 Internal Server Error: Server is confused, often caused by an error in the server-side program responding to the request.
- 501 Method Not Implemented: The request method used is invalid (could be caused by a typing error, e.g., "GET" misspell as "Get"). get /test.html HTTP/1.0
- 502 Bad Gateway: Proxy or Gateway indicates that it receives a bad response from the upstream server.
- 503 Service Unavailable: Server cannot response due to overloading or maintenance. The client can try again later.
- 504 Gateway Timeout: Proxy or Gateway indicates that it receives a timeout from an upstream server.

Tham khảo các bước thực hiện Bookstore

### B1. Thao tác với CSDL NoSQL

Tạo bảng với các khóa Partition Key và Sort Key

```
const AWS = require("aws-sdk");

AWS.config.update({
 region: "local",
 endpoint: "http://localhost:8000"
});

let dynamodb = new AWS.DynamoDB();

let params = {
 TableName: "Books",
 KeySchema: [
 {AttributeName: "name", KeyType: "HASH"},
 {AttributeName: "year", KeyType: "RANGE"}
],
 AttributeDefinitions: [
 {AttributeName: "name", AttributeType: "S"},
 {AttributeName: "year", AttributeType: "N"}
],
 ProvisionedThroughput: {
 ReadCapacityUnits: 10,
 WriteCapacityUnits: 10
 }
};

dynamodb.createTable(params, (err, data) => {
 if(err){
 console.error(`Something went wrong ${JSON.stringify(err,null,2)}`);
 }else{
 console.log(`Created table ${JSON.stringify(data, null, 2)}`);
 }
});
```

Lấy dữ liệu từ tập tin JSON

```
const AWS = require("aws-sdk");
const fs = require("fs");

AWS.config.update({
 region: "local",
 endpoint: "http://localhost:8000"
});
```

```

let docClient = new AWS.DynamoDB.DocumentClient();

console.log("Start importing");

let allBooks = JSON.parse(fs.readFileSync(__dirname+'/bookdata.json', 'utf-8'));

allBooks.forEach((book) => {
 let params = {
 TableName: "Books",
 Item: {
 "name": book.name,
 "year": book.year,
 "type": book.type,
 "author": book.author
 }
 };
 docClient.put(params,(err, data) => {
 if (err) {
 console.error(`Unable to add book ${book.title}, ${JSON.stringify(err, null, 2)}`);
 }else{
 console.log(`Book created ${book.name}`);
 }
 });
});

```

Scan dữ liệu từ bảng Books (lấy hết data không dùng Primary Key)

```

const AWS = require('aws-sdk');

AWS.config.update({
 region: 'local',
 endpoint: 'http://localhost:8000',
});

const docClient = new AWS.DynamoDB.DocumentClient();
const params = {
 TableName: "Books",
};
console.log("Scan ...");

docClient.scan(params, onScan);
function onScan(err, data) {
 if (err) {
 console.error("Unable to scan the table. Error JSON:", JSON.stringify(err, null, 2));
 }
}

```

```

} else {
 console.log("Scan succeeded.");
 data.Items.forEach((book) => {
 console.log(book.name, book.year, book.type, book.author);
 });
}

if (typeof data.LastEvaluatedKey !== "undefined") {
 console.log("Scanning for more...");
 params.ExclusiveStartKey = data.LastEvaluatedKey;
 docClient.scan(params, onScan);
}
}
}
}

```

## B2. Các hàm chung xử lý với AWS DynamoDB

```

const aws = require("aws-sdk");
const form_functions = require("./function_form");

aws.config.update({
 region: "local",
 endpoint: "http://localhost:8000"
});

let docClient = new aws.DynamoDB.DocumentClient();

function getAllItems(res) {
 let params = {
 TableName: "Books"
 };

 let scanObject = {};

 docClient.scan(params, (err, data) => {
 if (err) {
 scanObject.err = err;
 } else {
 scanObject.data = data;
 }
 form_functions.listTable(scanObject, res);
 });
}

function searchItems(year, name, res) {
 let params = {

```

```

 TableName: "Books"
};

let queryObject = {};

console.log(year + "" + name);

if (year) {
 if (name) {
 params.KeyConditionExpression = "#y = :year and #n =:name";
 params.ExpressionAttributeNames = {
 "#y": "year",
 "#n": "name"
 };
 params.ExpressionAttributeValues = {
 ":year": Number(year),
 ":name": String(name)
 };
 docClient.query(params, (err, data) => {
 if (err) {
 queryObject.err = err;
 } else {
 queryObject.data = data;
 }
 form_functions.listTable(queryObject, res);
 });
 }
} else {
 params.FilterExpression = "#y = :year";
 params.ExpressionAttributeNames = { "#y": "year" };
 params.ExpressionAttributeValues = { ":year": Number(year) };
 docClient.scan(params, (err, data) => {
 if (err) {
 queryObject.err = err;
 } else {
 queryObject.data = data;
 }
 form_functions.listTable(queryObject, res);
 });
}
}

else if (!year) {
 if (name) {
 params.FilterExpression = "#n = :name";
 params.ExpressionAttributeNames = { "#n": "name" };
 params.ExpressionAttributeValues = { ":name": String(name) };
 docClient.scan(params, (err, data) => {

```

```

 if (err) {
 queryObject.err = err;
 } else {
 queryObject.data = data;
 }
 form_functions.listTable(queryObject, res);
 });
}
}

function createItem(year, name, type, author, res) {
let params = {
 TableName: "Books",
 Item: {
 name: String(name),
 year: Number(year),
 type: String(type),
 author: String(author)
 }
};
docClient.put(params, (err, data) => {
 if (err) {
 form_functions.addNewForm(res);
 res.write("<h5 style='color:red;'>All fields are required!</h5>");
 }
 else {
 res.writeHead(302, { "Location": "/" });
 }
 res.end();
});
}

function updateItem(year, name, type, author, res) {
let params = {
 TableName: "Books",
 Key: {
 "name": String(name),
 "year": Number(year)
 },
 UpdateExpression: "set #t = :type, #a = :author",
 ExpressionAttributeNames: {
 "#t": "type",
 "#a": "author"
 },
 ExpressionAttributeValues: {

```

```

 ":type": String(type),
 ":author": String(author)
},
ReturnValues: "UPDATED_NEW"
};

docClient.update(params, (err, data) => {
if (err) {
 form_functions.editForm(year, name, type, author, res);
 res.write("<h5 style='color:red;'>All fields are required!</h5>");
} else {
 res.writeHead(302, { "Location": "/" });
}
res.end();
});

}

function deleteItem(year, name, res) {
let params = {
TableName: "Books",
Key: {
 "name": String(name),
 "year": Number(year)
}
};

docClient.delete(params, (err, data) => {
if (err) {
 console.error("Unable to delete item. Error JSON:", JSON.stringify(err, null, 2));
} else {
 res.writeHead(302, { "Location": "/" });
}
res.end();
});
}

module.exports = {
getAllItems: getAllItems,
searchItems: searchItems,
createItem: createItem,
updateItem: updateItem,
deleteItem: deleteItem
};

```

### B3. Tạo giao diện HTML

Form tạo mới Book

```

<form action="/create" method="get">
 <label>
 Year:<input name="year" type="text" />
 </label>

 <label>
 Title:<input name="name" type="text" />
 </label>

 <label>
 Type:<input name="type" type="text" />
 </label>

 <label>
 Author:<input name="author" type="text" />
 </label>

 <input type="submit" value="Save" />
 <input type="reset" value="Reset" />
</form>

```

*Form update Book theo Primary Key (year, title) (Hàm xử lý sẽ đọc file, thay thế giá trị value của các field. Các HASH Key, RANGE Key để ché đố readonly)*

```

<form action="/save" method="get">

 <label>
 Year:<input name="year" type="text" readonly="readonly"/>
 </label>

 <label>
 Title:<input name="name" type="text" readonly="readonly"/>
 </label>

 <p>Year & Title: Primary Key(Hashkey Sortkey)

 </p>
 <label>
 Type:<input name="type" type="text"/>
 </label>

 <label>
 Author:<input name="author" type="text"/>
 </label>

 <input type="submit" value="Save"/>
</form>

```

*Form Search*

```

<form action="/search" method="get">
 <label>
 Year:<input name="year" type="text" />
 </label>


```

```

<label>
 Title:<input name="name" type="text" />
</label>

<input type="submit" value="Search" />
</form>

```

#### B4. Các hàm chung xử lý các thao tác giao diện

```

var fs = require('fs');

function displaySearchForm(res) {
 var data = fs.readFileSync('../Views/SearchForm.html', 'utf-8');
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write(data);
}

function listTable(obj, res) {
 let tableHeader = "<table border='1px solid black"><tr><th>Year</th><th>Movie name</th><th>Movie type</th><th>Author</th>...</th></tr>";
 res.write(tableHeader);
 if (obj.err) {
 res.write("<h5 style='color:red;'>Error:: ${obj.err}</h5>");
 res.write("<tr><td colspan='5'>Nothing to show</td></tr>");
 } else {
 if (obj.data.Items.length === 0) {
 res.write("<tr><td colspan='5'>Nothing to show</td></tr>");
 }
 obj.data.Items.forEach((book) => {
 res.write(`<tr><td>${book.year}</td><td>${book.name}</td>
 <td>${book.type}</td><td>${book.author}</td>
 <td>Edit
 Delete</td></tr>`);
 });
 }
 res.write("</table>");
 res.end();
}

function addNewForm(res) {
 let data = fs.readFileSync('../Views/AddNewForm.html', 'utf-8');
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write(data);
}

function editForm(year, name, type, author, res) {
}

```

```

let data = fs.readFileSync("./Views/EditForm.html", "utf-8");
res.writeHead(200, {"Content-Type": "text/html"});
data = replaceYearValue(data, year);
data = replaceNameValue(data, name);
data = replaceTypeValue(data, type);
data = replaceAuthorValue(data, author);
res.write(data);
}

function replaceYearValue(data, year) {
let str = "<input name='year' type='text' readonly='readonly' />";
let index = data.indexOf(str) + str.length - 2;
return data.substr(0, index) + ` value="${year}" ` + data.substr(index);
}

function replaceNameValue(data, name) {
let str = "<input name='name' type='text' readonly='readonly' />";
let index = data.indexOf(str) + str.length - 2;
return data.substr(0, index) + ` value="${name}" ` + data.substr(index);
}

function replaceTypeValue(data, type) {
let str = "<input name='type' type='text' />";
let index = data.indexOf(str) + str.length - 2;
return data.substr(0, index) + ` value="${type}" ` + data.substr(index);
}

function replaceAuthorValue(data, author) {
let str = "<input name='author' type='text' />";
let index = data.indexOf(str) + str.length - 2;
return data.substr(0, index) + ` value="${author}" ` + data.substr(index);
}

module.exports = {
 displaySearchForm: displaySearchForm,
 addNewForm: addNewForm,
 editForm: editForm,
 listTable: listTable
};

```

### B5. Tao Server voi Node.js

```

var http = require("http");
var url = require("url");
var port = 9999;

```

```

var form_function = require("./function_form"); // Các hàm xử lý giao diện
var aws_function = require("./function_aws"); // Các hàm thao tác với NoSQL

http.createServer((req, res) => {
 let urlObject = url.parse(req.url, true);
 let pathName = urlObject.pathname;
 let data = urlObject.query;
 let year = data.year;
 let name = data.name;
 let type = data.type;
 let author = data.author;

 if (pathName === "/") {
 form_function.displaySearchForm(res);
 aws_function.getAllItems(res);
 }
 else if (pathName === "/search") {
 if (!year && !name) {
 res.writeHead(302, { "Location": "/" });
 res.end();
 }
 else {
 form_function.displaySearchForm(res);
 aws_function.searchItems(year, name, res);
 }
 }
 else if (pathName === "/new") {
 form_function.addNewForm(res);
 res.end();
 }
 else if (pathName === "/create") {
 aws_function.createItem(year, name, type, author, res);
 }
 else if (pathName === "/edit") {
 form_function.editForm(year, name, type, author, res);
 res.end();
 }
 else if (pathName === "/save") {
 aws_function.updateItem(year, name, type, author, res);
 }
 else if (pathName === "/delete") {
 aws_function.deleteItem(year, name, res);
 }
}) .listen(port, () => {
 console.log(`Server starting at port ${port}`);
});

```

## *Bài 9. Thực hiện Bài 8 dùng Express Framework*

### *Bài 10. Bài tập quản lý thông tin tin tức với DynamoDB*

Thông tin về tin tức cho 1 tạp chí online bao gồm các thông tin như Id, NewsTitle, PublishDate, Image, Content, Author và một số thông tin khác.

Tác giả của tin tức có thể là cộng tác viên, và cần lưu trữ thông tin như AuthorTitle, AuthorName, AuthorAddress.

Xây dựng ứng dụng bằng Node.js và DynamoDB cho phép thêm, xóa, sửa, liệt kê và tìm kiếm theo tiêu đề của tin tức. Mục tiêu hỗ trợ người dùng tìm kiếm theo tiêu đề (NewsTitle) của tin tức, tiêu đề của tin tức là duy nhất cho tạp chí online này.

Phân 4. *Chỉnh sửa nội dung báo cáo phân tích, thiết kế.*

Phân 5. *Cập nhật nhật ký thực hiện đồ án.*

# BÀI TẬP TUẦN 08. 09. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT

Mục tiêu:

- Tìm hiểu các công cụ phát triển ứng dụng với điện toán đám mây Visual Studio (.NET), Eclipse (Java), Visual Studio Code (Node.js)
- Cập nhật nhật ký thực hiện đồ án.
- Hoàn tất nội dung tập tin báo cáo đồ án.
- Kiểm thử ứng dụng.

Yêu cầu:

- Đã có tài khoản AWS.
- Cài đặt các AWS Toolkit cho Eclipse và .NET.
- Sử dụng Postman cho ứng dụng RESTful API

## Phần 1.

Tìm hiểu các công cụ phát triển ứng dụng với điện toán đám mây và cài đặt các AWS Toolkit cho Eclipse và .NET.

- Visual Studio (.NET)
- Eclipse (Java)
- Visual Studio Code (Node.js)

Phần 2. Thực hiện các bài tập phát triển ứng dụng với .NET/Java kết nối các services của AWS  
(Thực hiện 2 files bài tập với .NET và Java)

Phần 3. Hoàn tất tài liệu báo cáo của đồ án

- Lý thuyết nền tảng
- Phân tích, thiết kế đề tài (UML)
- Hiện thực đề tài
- Tài liệu tham khảo (liệt kê và trích dẫn tài liệu đúng theo quy định)

Phần 4. Thực hiện kiểm thử ứng dụng (kiểm thử đề tài)

# BÀI TẬP TUẦN 10. MÔN CÔNG NGHỆ MỚI TRONG PTUD CNTT

Mục tiêu:

- Hiểu và thực hiện được với các công cụ Developer và Management của AWS
- Cập nhật nhật ký thực hiện đồ án.
- Hoàn tất code của đồ án.
- Đóng gói ứng dụng, demo

Yêu cầu:

- Đã có tài khoản AWS. Sau khi thực hiện xong báo cáo, cần đóng tài khoản AWS.
- Sử dụng Postman cho kiểm thử các dữ liệu của ứng dụng RESTful API

Phần 1.

Tìm hiểu các công cụ hỗ trợ quản trị ứng dụng với điện toán đám mây

- Database Backup
- Management Tools

Phần 2. Thực hiện kiểm thử ứng dụng (kiểm thử đề tài)

Phần 3. Đóng gói ứng dụng, demo.