# **Xây dựng cơ sở kiến ​​thức kịp thời với Amazon Bedrock**

Các công ty phần mềm dưới dạng dịch vụ (SaaS) quản lý nhiều bên thuê phải đối mặt với một thách thức quan trọng: trích xuất hiệu quả những thông tin chi tiết có ý nghĩa từ các bộ sưu tập tài liệu khổng lồ trong khi vẫn kiểm soát được chi phí. Các phương pháp tiếp cận truyền thống thường dẫn đến chi tiêu không cần thiết cho các tài nguyên lưu trữ và xử lý chưa sử dụng, ảnh hưởng đến cả hiệu quả hoạt động và lợi nhuận. Các tổ chức cần các giải pháp có thể mở rộng quy mô tài nguyên xử lý và lưu trữ một cách thông minh dựa trên các mô hình sử dụng bên thuê thực tế trong khi vẫn duy trì được sự cô lập dữ liệu. Các hệ thống thế hệ tăng cường truy xuất (RAG) truyền thống tiêu thụ các tài nguyên có giá trị bằng cách thu thập và duy trì các nhúng cho các tài liệu có thể không bao giờ được truy vấn, dẫn đến chi phí lưu trữ không cần thiết và giảm hiệu quả của hệ thống. Các hệ thống được thiết kế để xử lý số lượng lớn bên thuê vừa và nhỏ có thể vượt quá giới hạn về cấu trúc chi phí và cơ sở hạ tầng hoặc có thể cần sử dụng các triển khai theo kiểu silo để giữ thông tin và cách sử dụng của từng bên thuê riêng biệt. Thêm vào sự phức tạp này, nhiều dự án có bản chất tạm thời, với công việc được hoàn thành không liên tục, dẫn đến dữ liệu chiếm không gian trong các hệ thống cơ sở kiến ​​thức có thể được các bên thuê đang hoạt động khác sử dụng.

Để giải quyết những thách thức này, bài đăng này trình bày một giải pháp cơ sở tri thức đúng lúc giúp giảm mức tiêu thụ không sử dụng thông qua xử lý tài liệu thông minh. Giải pháp này chỉ xử lý tài liệu khi cần và tự động loại bỏ các tài nguyên không sử dụng, do đó các tổ chức có thể mở rộng kho lưu trữ tài liệu của mình mà không làm tăng chi phí cơ sở hạ tầng theo tỷ lệ.

Với kiến ​​trúc đa thuê bao có giới hạn có thể cấu hình cho mỗi thuê bao, nhà cung cấp dịch vụ có thể cung cấp các mô hình định giá theo tầng trong khi vẫn duy trì sự cô lập dữ liệu nghiêm ngặt, khiến nó trở nên lý tưởng cho các ứng dụng SaaS phục vụ nhiều khách hàng có nhu cầu khác nhau. Hết hạn tài liệu tự động thông qua Thời gian tồn tại (TTL) đảm bảo hệ thống vẫn tinh gọn và tập trung vào nội dung có liên quan, trong khi làm mới TTL cho các tài liệu thường xuyên truy cập duy trì hiệu suất tối ưu cho thông tin quan trọng. Kiến trúc này cũng giúp giới hạn số lượng tệp mà mỗi thuê bao có thể tiếp nhận tại một thời điểm cụ thể và tốc độ mà thuê bao có thể truy vấn một tập hợp các tệp. Giải pháp này sử dụng các công nghệ không có máy chủ để giảm bớt chi phí vận hành và cung cấp khả năng mở rộng tự động, do đó các nhóm có thể tập trung vào logic kinh doanh thay vì quản lý cơ sở hạ tầng. Bằng cách sắp xếp các tài liệu thành các nhóm với bộ lọc dựa trên siêu dữ liệu, hệ thống cho phép truy vấn theo ngữ cảnh mang lại kết quả có liên quan hơn trong khi vẫn duy trì ranh giới bảo mật giữa các thuê bao. Tính linh hoạt của kiến ​​trúc hỗ trợ tùy chỉnh cấu hình thuê bao, tốc độ truy vấn và chính sách lưu giữ tài liệu, giúp thích ứng với các yêu cầu kinh doanh đang thay đổi mà không cần phải thiết kế lại đáng kể.

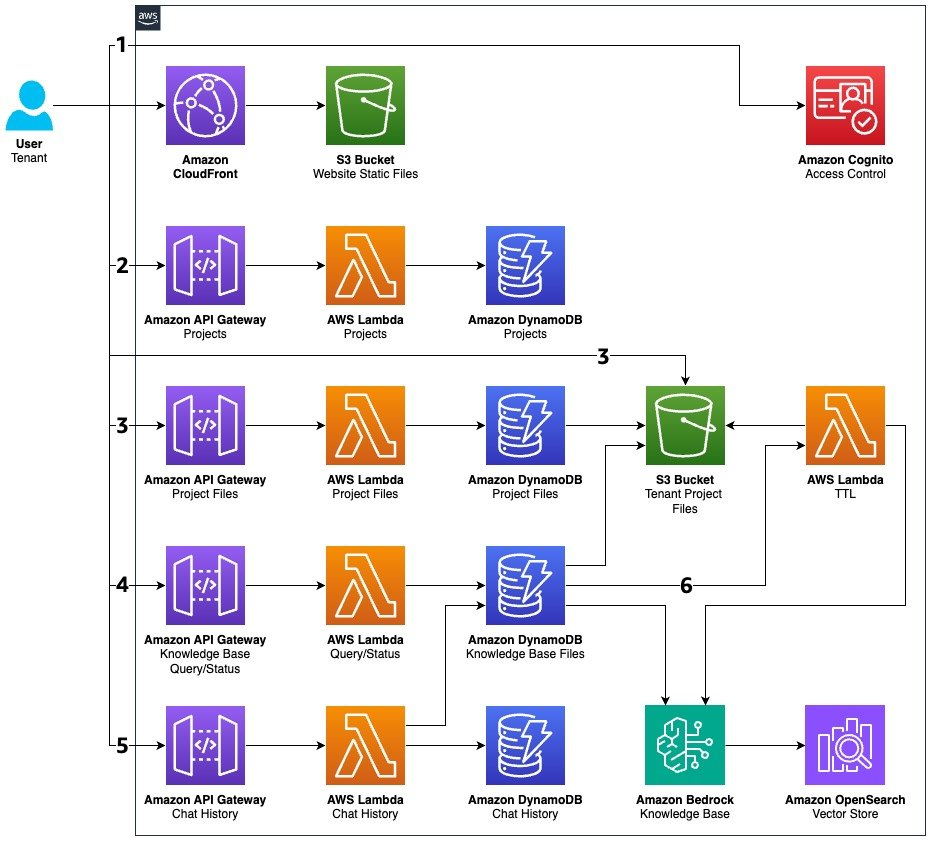
## **Tổng quan về giải pháp**

Kiến trúc này kết hợp một số dịch vụ AWS để tạo ra giải pháp cơ sở kiến ​​thức đa thuê bao, tiết kiệm chi phí, xử lý tài liệu theo yêu cầu. Các thành phần chính bao gồm:

* Cơ sở kiến ​​thức dựa trên vectơ – Sử dụng [Amazon Bedrock](https://aws.amazon.com/bedrock) và [Amazon OpenSearch Serverless](https://aws.amazon.com/opensearch-service/features/serverless/) để xử lý và truy vấn tài liệu hiệu quả
* Thu thập tài liệu theo yêu cầu – Triển khai xử lý kịp thời bằng cách sử dụng loại nguồn dữ liệu Amazon Bedrock [CUSTOM](https://docs.aws.amazon.com/bedrock/latest/userguide/custom-data-source-connector.html)
* Quản lý TTL – Tự động dọn dẹp các tài liệu không sử dụng bằng [tính năng TTL](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) trong [Amazon DynamoDB](https://aws.amazon.com/dynamodb)
* Cô lập nhiều bên thuê – Thực thi phân tách dữ liệu an toàn giữa người dùng và tổ chức với giới hạn tài nguyên có thể định cấu hình

Giải pháp cho phép kiểm soát chi tiết thông qua lọc dựa trên siêu dữ liệu ở cấp độ người dùng, người thuê và tệp. Hệ thống theo dõi DynamoDB TTL hỗ trợ cấu trúc giá theo tầng, trong đó người thuê có thể trả tiền cho các khoảng thời gian TTL, giới hạn thu thập tài liệu và tỷ lệ truy vấn khác nhau.

Sơ đồ sau minh họa các thành phần chính và quy trình làm việc của giải pháp.



Quy trình làm việc bao gồm các bước sau:

1. Người dùng đăng nhập vào hệ thống, hệ thống sẽ gắn ID người thuê vào người dùng hiện tại để gọi đến cơ sở kiến ​​thức Amazon Bedrock. Bước xác thực này rất quan trọng vì nó thiết lập ngữ cảnh bảo mật và đảm bảo các tương tác tiếp theo được liên kết đúng với đúng người thuê. ID người thuê trở thành phần siêu dữ liệu cơ bản cho phép cô lập nhiều người thuê và quản lý tài nguyên phù hợp trong toàn bộ quy trình làm việc.
2. Sau khi xác thực, người dùng tạo một dự án sẽ đóng vai trò là một container cho các tệp họ muốn truy vấn. Bước tạo dự án này thiết lập cấu trúc tổ chức cần thiết để quản lý các tài liệu liên quan với nhau. Hệ thống tạo siêu dữ liệu phù hợp và tạo các mục nhập cơ sở dữ liệu cần thiết để theo dõi mối liên kết của dự án với người thuê cụ thể, cho phép kiểm soát truy cập và quản lý tài nguyên phù hợp ở cấp độ dự án.
3. Với một dự án được thiết lập, người dùng có thể bắt đầu tải lên các tệp. Hệ thống quản lý quy trình này bằng cách tạo các URL được ký trước để tải lên tệp an toàn. Khi các tệp được tải lên, chúng được lưu trữ trong [Amazon Simple Storage Service](http://aws.amazon.com/s3) (Amazon S3) và hệ thống tự động tạo các mục trong DynamoDB liên kết từng tệp với cả dự án và người thuê. Mối quan hệ ba chiều này (tệp-dự án-người thuê) rất cần thiết để duy trì sự cô lập dữ liệu phù hợp và cho phép truy vấn hiệu quả sau này.
4. Khi người dùng yêu cầu tạo cuộc trò chuyện với cơ sở kiến ​​thức cho một dự án cụ thể, hệ thống sẽ bắt đầu thu thập các tệp dự án bằng nguồn dữ liệu CUSTOM. Đây là nơi bắt đầu quá trình xử lý just-in-time. Trong quá trình thu thập, hệ thống áp dụng giá trị TTL dựa trên khoảng thời gian TTL cụ thể theo từng tầng của người thuê. TTL đảm bảo các tệp dự án vẫn khả dụng trong phiên trò chuyện trong khi thiết lập khuôn khổ để tự động dọn dẹp sau đó. Bước này thể hiện cốt lõi của chiến lược xử lý theo yêu cầu, vì các tệp chỉ được xử lý khi cần thiết.
5. Mỗi phiên trò chuyện sẽ chủ động cập nhật TTL cho các tệp dự án đang được sử dụng. Quản lý TTL động này đảm bảo các tệp được truy cập thường xuyên vẫn nằm trong cơ sở kiến ​​thức trong khi cho phép các tệp ít được sử dụng hết hạn một cách tự nhiên. Hệ thống liên tục làm mới các giá trị TTL dựa trên mức sử dụng thực tế, tạo ra sự cân bằng hiệu quả giữa tính khả dụng của tài nguyên và tối ưu hóa chi phí. Phương pháp này duy trì hiệu suất tối ưu cho nội dung được sử dụng tích cực trong khi giúp ngăn ngừa lãng phí tài nguyên cho các tài liệu chưa sử dụng.
6. Sau khi phiên trò chuyện kết thúc và giá trị TTL hết hạn, hệ thống sẽ tự động xóa các tệp khỏi cơ sở kiến ​​thức. Quá trình dọn dẹp này được kích hoạt bởi [Amazon DynamoDB Streams](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html) theo dõi các sự kiện hết hạn TTL, kích hoạt chức năng [AWS Lambda](http://aws.amazon.com/lambda) để xóa các tài liệu đã hết hạn. Bước cuối cùng này giảm tải cho cụm OpenSearch Serverless cơ bản và tối ưu hóa tài nguyên hệ thống, đảm bảo cơ sở kiến ​​thức vẫn tinh gọn và hiệu quả.

## **Điều kiện tiên quyết**

Bạn cần những điều kiện tiên quyết sau đây trước khi có thể tiến hành giải pháp. Đối với bài đăng này, chúng tôi sử dụng us-east-1AWS Region.

* Một tài khoản AWS đang hoạt động có quyền tạo tài nguyên trongus-east-1
* Giao [diện dòng lệnh AWS](http://aws.amazon.com/cli) (AWS CLI) [đã được cài đặt](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html)
* Bộ [công cụ phát triển đám mây AWS](https://aws.amazon.com/cdk/) (AWS CDK) [đã được cài đặt](https://docs.aws.amazon.com/cdk/v2/guide/getting-started.html)
* Đã cài đặt Git để sao chép kho lưu trữ

## **Triển khai giải pháp**

Hoàn tất các bước sau để triển khai giải pháp:

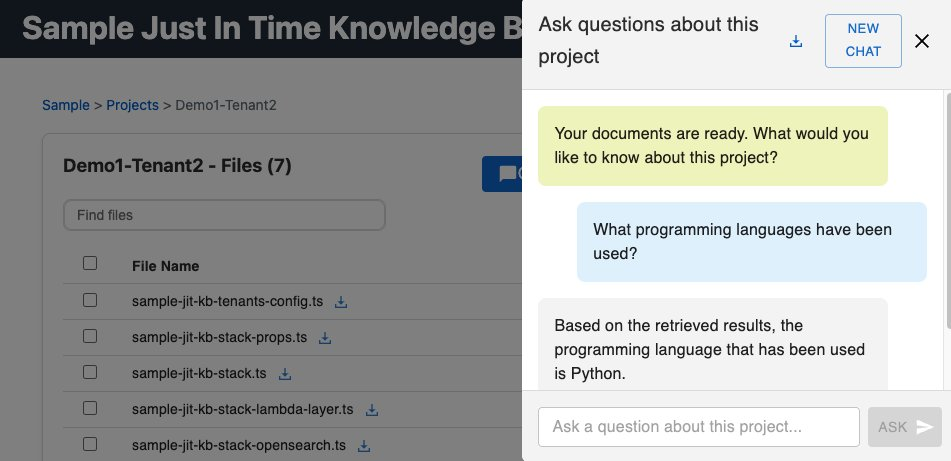
1. Tải xuống dự án AWS CDK từ [kho lưu trữ GitHub](https://github.com/aws-samples/sample-just-in-time-knowledge-base) .
2. Cài đặt các phụ thuộc của dự án:
3. npm run install:all
4. Triển khai giải pháp:
5. npm run deploy
6. Tạo người dùng và đăng nhập vào hệ thống sau khi xác thực email của bạn.

## **Xác thực cơ sở kiến ​​thức và chạy truy vấn**

Trước khi cho phép người dùng trò chuyện với tài liệu của mình, hệ thống thực hiện các bước sau:

* Thực hiện kiểm tra xác thực để xác định xem tài liệu có cần được nhập hay không. Quá trình này diễn ra minh bạch với người dùng và bao gồm kiểm tra trạng thái tài liệu trong DynamoDB và cơ sở kiến ​​thức.
* Xác thực rằng các tài liệu bắt buộc đã được tiếp nhận thành công và lập chỉ mục đúng cách trước khi cho phép truy vấn.
* Trả về cả câu trả lời do AI tạo ra và trích dẫn có liên quan đến tài liệu nguồn, duy trì khả năng truy xuất nguồn gốc và trao quyền cho người dùng xác minh tính chính xác của phản hồi.

Ảnh chụp màn hình sau đây minh họa ví dụ về cách trò chuyện với tài liệu.



Khi xem phương pháp ví dụ sau đây để nhập tệp, hãy lưu ý cách thông tin tệp được lưu trữ trong DynamoDB với giá trị TTL để hết hạn tự động. Cuộc gọi tài liệu cơ sở kiến ​​thức nhập bao gồm siêu dữ liệu cần thiết (ID người dùng, ID người thuê và dự án), cho phép lọc chính xác các tệp của người thuê này trong các hoạt động tiếp theo.

# Ingesting files with tenant-specific TTL values

def ingest\_files(user\_id, tenant\_id, project\_id, files):

# Get tenant configuration and calculate TTL

tenants = json.loads(os.environ.get('TENANTS'))['Tenants']

tenant = find\_tenant(tenant\_id, tenants)

ttl = int(time.time()) + (int(tenant['FilesTTLHours']) \* 3600)

# For each file, create a record with TTL and start ingestion

for file in files:

file\_id = file['id']

s3\_key = file.get('s3Key')

bucket = file.get('bucket')

# Create a record in the knowledge base files table with TTL

knowledge\_base\_files\_table.put\_item(

Item={

'id': file\_id,

'userId': user\_id,

'tenantId': tenant\_id,

'projectId': project\_id,

'documentStatus': 'ready',

'createdAt': int(time.time()),

'ttl': ttl # TTL value for automatic expiration

}

)

# Start the ingestion job with tenant, user, and project metadata for filtering

bedrock\_agent.ingest\_knowledge\_base\_documents(

knowledgeBaseId=KNOWLEDGE\_BASE\_ID,

dataSourceId=DATA\_SOURCE\_ID,

clientToken=str(uuid.uuid4()),

documents=[

{

'content': {

'dataSourceType': 'CUSTOM',

'custom': {

'customDocumentIdentifier': {

'id': file\_id

},

's3Location': {

'uri': f"s3://{bucket}/{s3\_key}"

},

'sourceType': 'S3\_LOCATION'

}

},

'metadata': {

'type': 'IN\_LINE\_ATTRIBUTE',

'inlineAttributes': [

{'key': 'userId', 'value': {'stringValue': user\_id, 'type': 'STRING'}},

{'key': 'tenantId', 'value': {'stringValue': tenant\_id, 'type': 'STRING'}},

{'key': 'projectId', 'value': {'stringValue': project\_id, 'type': 'STRING'}},

{'key': 'fileId', 'value': {'stringValue': file\_id, 'type': 'STRING'}}

]

}

}

]

)

Trong quá trình truy vấn, bạn có thể sử dụng siêu dữ liệu liên quan để xây dựng các tham số đảm bảo bạn chỉ truy xuất các tệp thuộc về đối tượng thuê cụ thể này. Ví dụ:

filter\_expression = {

"andAll": [

{

"equals": {

"key": "tenantId",

"value": tenant\_id

}

},

{

"equals": {

"key": "projectId",

"value": project\_id

}

},

{

"in": {

"key": "fileId",

"value": file\_ids

}

}

]

}

# Create base parameters for the API call

retrieve\_params = {

'input': {

'text': query

},

'retrieveAndGenerateConfiguration': {

'type': 'KNOWLEDGE\_BASE',

'knowledgeBaseConfiguration': {

'knowledgeBaseId': knowledge\_base\_id,

'modelArn': 'arn:aws:bedrock:us-east-1::foundation-model/amazon.nova-pro-v1:0',

'retrievalConfiguration': {

'vectorSearchConfiguration': {

'numberOfResults': limit,

'filter': filter\_expression

}

}

}

}

}

response = bedrock\_agent\_runtime.retrieve\_and\_generate(\*\*retrieve\_params)

## **Quản lý vòng đời tài liệu với TTL**

Để tối ưu hóa hơn nữa việc sử dụng tài nguyên và chi phí, bạn có thể triển khai hệ thống quản lý vòng đời tài liệu thông minh bằng tính năng DynamoDB (TTL). Điều này bao gồm các bước sau:

1. Khi một tài liệu được đưa vào cơ sở kiến ​​thức, một bản ghi sẽ được tạo ra với giá trị TTL có thể cấu hình được.
2. TTL này sẽ được làm mới khi tài liệu được truy cập.
3. Luồng DynamoDB với các bộ lọc cụ thể cho các sự kiện hết hạn TTL được sử dụng để kích hoạt chức năng dọn dẹp Lambda.
4. Hàm Lambda xóa các tài liệu đã hết hạn khỏi cơ sở kiến ​​thức.

Xem đoạn mã sau:

# Lambda function triggered by DynamoDB Streams when TTL expires items

def lambda\_handler(event, context):

"""

This function is triggered by DynamoDB Streams when TTL expires items.

It removes expired documents from the knowledge base.

"""

# Process each record in the event

for record in event.get('Records', []):

# Check if this is a TTL expiration event (REMOVE event from DynamoDB Stream)

if record.get('eventName') == 'REMOVE':

# Check if this is a TTL expiration

user\_identity = record.get('userIdentity', {})

if user\_identity.get('type') == 'Service' and user\_identity.get('principalId') == 'dynamodb.amazonaws.com':

# Extract the file ID and tenant ID from the record

keys = record.get('dynamodb', {}).get('Keys', {})

file\_id = keys.get('id', {}).get('S')

# Delete the document from the knowledge base

bedrock\_agent.delete\_knowledge\_base\_documents(

clientToken=str(uuid.uuid4()),

knowledgeBaseId=knowledge\_base\_id,

dataSourceId=data\_source\_id,

documentIdentifiers=[

{

'custom': {

'id': file\_id

},

'dataSourceType': 'CUSTOM'

}

]

)

## **Cô lập nhiều người thuê với các mức dịch vụ theo tầng**

Kiến trúc của chúng tôi cho phép cô lập nhiều đối tượng thuê bao tinh vi với nhiều cấp độ dịch vụ:

* Lọc tài liệu theo từng đối tượng thuê – Mỗi truy vấn bao gồm các bộ lọc dành riêng cho người dùng, đối tượng thuê và tệp, cho phép hệ thống giảm số lượng tài liệu được truy vấn.
* Giá trị TTL có thể cấu hình – Các tầng thuê bao khác nhau có thể có cấu hình TTL khác nhau. Ví dụ:
  + Gói miễn phí: Năm tài liệu được tiếp nhận với TTL 7 ngày và năm truy vấn mỗi phút.
  + Gói tiêu chuẩn: 100 tài liệu được tiếp nhận với TTL 30 ngày và 10 truy vấn mỗi phút.
  + Gói cao cấp: tiếp nhận 1.000 tài liệu với TTL 90 ngày và 50 truy vấn mỗi phút.
  + Bạn có thể cấu hình các giới hạn bổ sung, chẳng hạn như tổng số truy vấn mỗi tháng hoặc tổng số tệp được thu thập mỗi ngày hoặc mỗi tháng.

## **Dọn dẹp**

Để dọn dẹp các tài nguyên được tạo trong bài đăng này, hãy chạy lệnh sau từ cùng vị trí mà bạn đã thực hiện bước triển khai:

npm run destroy

## **Phần kết luận**

Kiến trúc cơ sở tri thức just-in-time được trình bày trong bài đăng này chuyển đổi quản lý tài liệu trên nhiều đối tượng thuê bằng cách chỉ xử lý tài liệu khi được truy vấn, giảm mức tiêu thụ chưa sử dụng của các hệ thống RAG truyền thống. Việc triển khai không có máy chủ này sử dụng Amazon Bedrock, OpenSearch Serverless và tính năng DynamoDB TTL để tạo ra một hệ thống tinh gọn với quản lý vòng đời tài liệu thông minh, giới hạn đối tượng thuê có thể định cấu hình và cô lập dữ liệu nghiêm ngặt, điều này rất cần thiết đối với các nhà cung cấp SaaS cung cấp mô hình định giá theo tầng.

Giải pháp này giải quyết trực tiếp các hạn chế về cấu trúc chi phí và cơ sở hạ tầng của các hệ thống truyền thống, đặc biệt là đối với các triển khai xử lý nhiều đối tượng thuê bao vừa và nhỏ với các dự án tạm thời. Kiến trúc này kết hợp xử lý tài liệu theo yêu cầu với quản lý vòng đời tự động, cung cấp một nguồn tài nguyên có khả năng mở rộng, tiết kiệm chi phí, giúp các tổ chức tập trung vào việc trích xuất thông tin chi tiết thay vì quản lý cơ sở hạ tầng, đồng thời duy trì ranh giới bảo mật giữa các đối tượng thuê bao.

Sẵn sàng triển khai kiến ​​trúc này chưa? Mã mẫu đầy đủ có sẵn trong [kho lưu trữ GitHub](https://github.com/aws-samples/sample-just-in-time-knowledge-base) .

### **Về tác giả**

Steven Warwick là Kiến trúc sư giải pháp cao cấp tại AWS, nơi ông lãnh đạo các hoạt động tương tác với khách hàng để thúc đẩy việc áp dụng đám mây thành công và chuyên về kiến ​​trúc SaaS và các giải pháp AI tạo sinh. Ông tạo ra nội dung giáo dục bao gồm các bài đăng trên blog và mã mẫu để giúp khách hàng triển khai các phương pháp hay nhất và đã lãnh đạo các chương trình về chủ đề GenAI cho các kiến ​​trúc sư giải pháp. Steven mang nhiều thập kỷ kinh nghiệm về công nghệ vào vai trò của mình, giúp khách hàng đánh giá kiến ​​trúc, tối ưu hóa chi phí và phát triển bằng chứng khái niệm.