

# A Conversation with your Architecture

Kevin Hakanson & Jenna Pederson

#OSN2025

What is your experience with diagrams?

# Who are we?

## Kevin Hakanson

- Solutions Architect at AWS\*
- 30+ years as developer and architect
- UST alumni (MS in Software Engineering)
- Diagram “snob”
- Humorous\*

*\* (opinions are my own)*

## Jenna Pederson

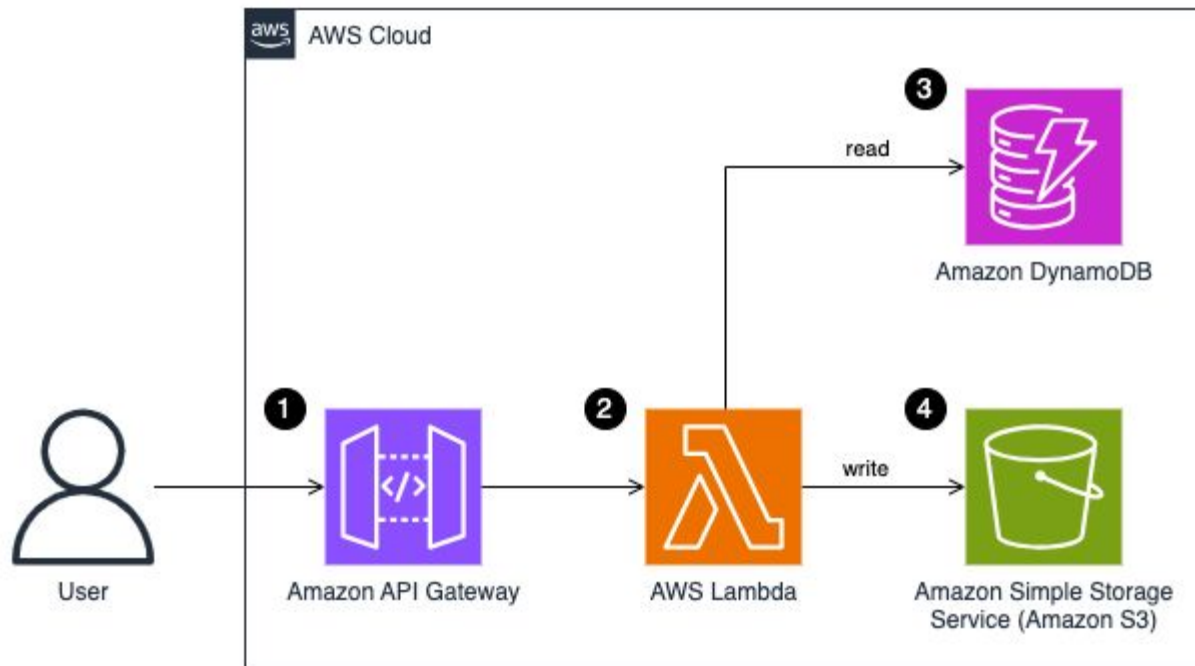
- Developer Advocate at Pinecone
- Full-stack developer before it was cool
- Consumes a lot of diagrams
- Creates diagrams, but NOT A DESIGNER

# Diagrams are hard

Reading, analyzing, creating, maintaining, making informed decisions from, and communicating through diagrams

# Let's play a game

We're going to show a diagram on the next slide and we need you to analyze it.



# We also asked Claude to analyze the diagram

It took 9 seconds.

```
1 # Set up the conversation agent and select model
2 png_conversation = Conversation(CLAUDE_V35_SONNET, bedrock_client)
3 png_conversation.system("You are an expert in AWS services and serverless architecture.")
4
5 # the PNG export from draw.io
6 response = png_conversation.chat(
7     "Describe the key components and services in the attached diagram.",
8     filename="apiendpoint.drawio.png"
9 )
```

✓ 8.9s

Python

The diagram illustrates a serverless architecture using AWS services. Here are the key components and services shown:

1. User: Represented by a simple icon, indicating the starting point of the interaction.
2. Amazon API Gateway: This is the entry point for the user's requests. It manages and routes API calls to the appropriate backend services.
3. AWS Lambda: Shown as the next step after API Gateway. Lambda is a serverless compute service that runs code in response to events and automatically manages the underlying compute resources.
4. Amazon DynamoDB: Depicted with a "read" arrow from Lambda. DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
5. Amazon Simple Storage Service (Amazon S3): Shown with a "write" arrow from Lambda. S3 is an object storage service offering industry-leading scalability, data availability, security, and performance.

The flow of the architecture appears to be:

1. The user interacts with the API Gateway.
2. API Gateway triggers a Lambda function.
3. The Lambda function reads data from DynamoDB.
4. The Lambda function then writes data to Amazon S3.

This serverless architecture allows for building scalable, event-driven applications without managing servers. It leverages AWS's managed services to handle infrastructure concerns, allowing developers to focus on writing code and business logic.



# Implementation technologies

- Large Language Models (LLMs)
- Prompts (user and system)
- Multi-modal models for image understanding
- Retrieval-Augmented Generation (RAG)
- Knowledge Base integration
- Vector databases for semantic search
- MCPs and tools
- Agents and orchestration



Amazon Nova Canvas prompt:

“A software project team is collaborating around a whiteboard. A developer is drawing a cloud architecture on the whiteboard. A security engineer is holding a clipboard with a checklist of items. A project manager is on her laptop, working on a spreadsheet.”

# Onboarding & understanding

New developer joining a complex project

#OSN2025

In the past:

- Meetings, trainings
- Reading outdated docs and diagrams
- Reverse engineering code and infrastructure
- Sifting through git commits
- Finding and waiting for time from expert

# What questions are we asking?

- How are we thinking about this problem? Why did we design it this way?
- What are some of the technical challenges that we had to address with this next generation serverless architecture?
- Why are namespaces so important today?
- How does data flow for the write path? The query path?
- What is adaptive caching and how does it work?
- What are some of the access patterns for this type of workload?

# Evolving an architecture

Building out a new feature,  
integrating a new service

#OSN2025

In the past:

- Reality differs from documentation
- Find historical examples of similar solutions, in the code, in other systems we maintain, in the industry
- Match against internal standards and architectural decision records (ADRs) stored in the project, on a wiki, or on a post-it note
- Architecture reviews with internal “experts”

# What questions are we asking?

- How did our last project solve this problem?
- We need to add a recommendation engine. Where should we integrate it?
- Should we roll our own or use a 3rd party service?
- How would the first option affect our database load?
- What happens if the database has increased latency or goes down completely? Does that mean we can't take orders, too?

# Security analysis

Incident response, third-party  
integration security review

#OSN2025

In the past:

- Review a checklist
- Analyze diagrams for potential vulnerabilities and then go re-analyze the actual infra/code
- Highlight trust boundaries and data flows
- Evaluate against security best practices
- Run pen tests

# What questions are we asking?

- After a breach, which other services and data could be compromised through this entry point?
- We're adding Stripe payments - what are the security implications?
- What are all the trust boundaries that change when we move to AWS?
- Who are our vendors and do we need to audit any licensing agreements?



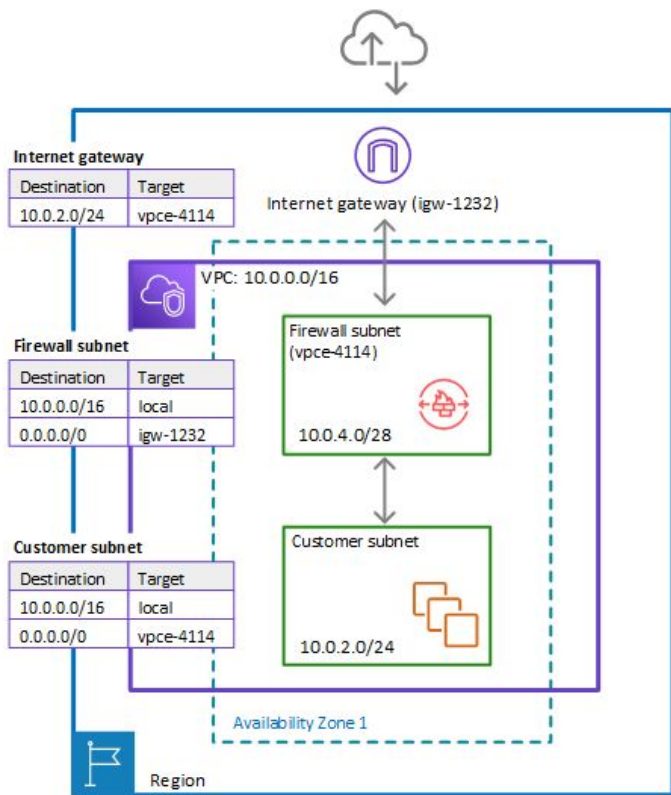
# State of AI & diagrams today

#OSN2025

Today, we're seeing:

- Text-to-diagram generation
- Diagram-to-text description
- Basic analysis capabilities
- Starting to see this integrated into developer tools like IDEs and coding assistants

# “describe the key components and services in the diagram”



This diagram illustrates a typical AWS VPC (Virtual Private Cloud) network architecture. Here are the key components and services depicted:

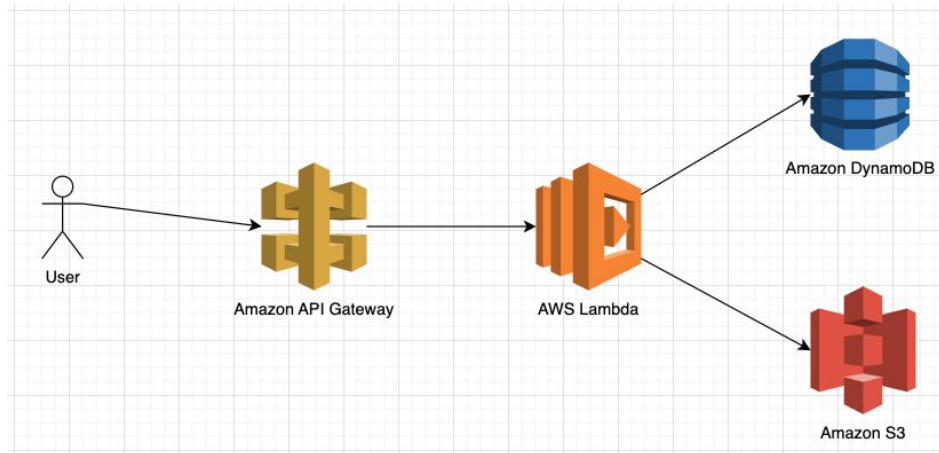
1. **VPC:** The main container (10.0.0.0/16) that encompasses the entire private network infrastructure.
2. **Internet Gateway (IGW):** Represented by the cloud icon at the top, allowing communication between the VPC and the internet.
3. **Subnets:** Three distinct subnets are shown within the VPC:
  - Firewall subnet (10.0.4.0/28)
  - Customer subnet (10.0.2.0/24)
  - Another subnet labeled as "Firewall subnet" (vpc-4114)
4. **Route Tables:** Depicted as tables showing destinations and targets for network traffic:
  - Internet Gateway route table
  - Firewall subnet route table
  - Customer subnet route table
5. **Availability Zone:** The entire VPC setup is shown within a single Availability Zone (AZ1).
6. **Region:** The VPC is contained within an AWS Region, as indicated by the "Region" label at the bottom.
7. **Security:** The presence of a firewall subnet suggests security measures are in place to control traffic flow.
8. **Connectivity:** Arrows between components indicate network connections and potential traffic flow.

This architecture demonstrates a segmented network design with controlled internet access, which is typical for enhancing security and managing network traffic in cloud environments. The use of separate subnets for different purposes (firewall, customer) allows for more granular control over network access and security policies.

# What do you think about this diagram?

An architecture follows a serverless approach, where the AWS Lambda function handles the application logic, and the Amazon API Gateway acts as the entry point for user requests. The Amazon DynamoDB database stores structured data, while the Amazon S3 service stores unstructured data or files.

Create a DrawIO formatted XML that demonstrates this architecture.



# Context is key

System Prompt

Formatting Rules

Golden Examples

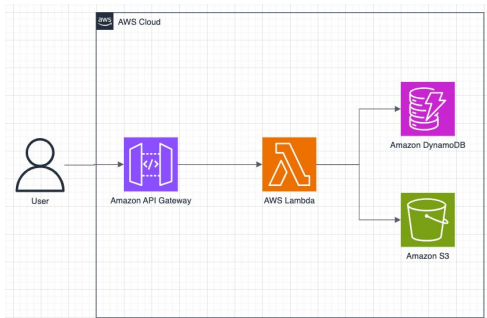
User Prompt

```
1 with open('system-example1.txt', 'r') as file:
2     system_example1 = file.read()
3 with open('system-example2.txt', 'r') as file:
4     system_example2 = file.read()
5
6 expert_system_prompt = f"""
7 You are an expert in software architecture with excellent skills using draw.io diagram tool.
8 Your goal is to provide draw.io formatted XML using the mxgraph.aws4 objects which demonstrates the most detailed AWS architecture requested by the user.
9
10
11 Rules you must follow while building your AWS diagrams using draw.io formatted XML:
12
13 <rules>
14 - Always prefer to use most recent AWS Icons from 2024
15 - Use the correct AWS visual branding and iconography
16 - Availability zones are placed completely inside of regions and VPC groups.
17 - Subnets must not touch other subnets, and availability zones must not touch other availability zones
18 - Make sure the availability zones are big enough to fit all of the applicable subnets inside
19 - Internet gateways must be along one of the edges of the VPC shape, preferably in the top middle unless stated otherwise
20 - Only use mxgraph.aws4 shapes in the diagrams
21 - There should never be text hidden behind other objects
22 </rules>
23
24 Here are 2 examples of how to respond in a standard interaction.:
25
26 {system_example1}
27 {system_example2}
28
29
30
31 Make sure the output follows all of the rules provided to you for creating draw.io formatted XML.
32 The output xml should use:
33 - a short id
34 - use {datetime.now(timezone.utc).isoformat(timespec='milliseconds')} + 'Z' for any modified attribute
35 - All objects must use mxgraph.aws4 shapes. Please include the object name in the drawing
36 The response must absolutely only contain xml code
37 """
```

Python

```
expert_conversation.system(expert_system_prompt)
```

```
response = expert_conversation.chat("""
Create a diagram for an architecture following a serverless approach,
where the AWS Lambda function handles the application logic,
and the Amazon API Gateway acts as the entry point for user requests.
The Amazon DynamoDB database stores structured data,
while the Amazon S3 service stores unstructured data or files.
""")
```



# Amazon Q Developer



**Olivier Lemaitre** • Following

GenAI Powered Architect who helps you build on AWS

1mo • Edited •

If your are interested, I have been invited to share live my experience with Diagrams & [#AmazonQDeveloper](#), with the [Believe In Serverless](#) community on April 15th next week :)

That will be streamed online (LinkedIn & YouTube). It will be in English, but it should be easy to follow by french ears I guess 😊

I will be using [#AmazonQDeveloper](#) and its features, but what I will explain isn't limited to this coding assistant, it's opened to anyone using one or curious about this one in particular.

**Amazon Q  
Developer**



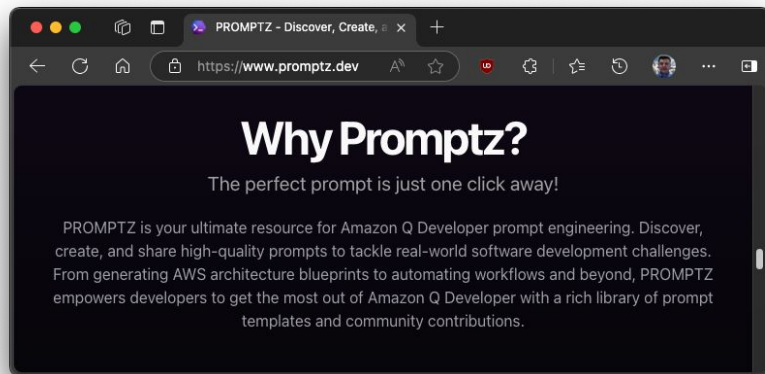
**Christian Bonzelet** • Following

AWS Solutions Architect at Bundesliga 🏆 | AWS Community ...

[Visit my website](#)

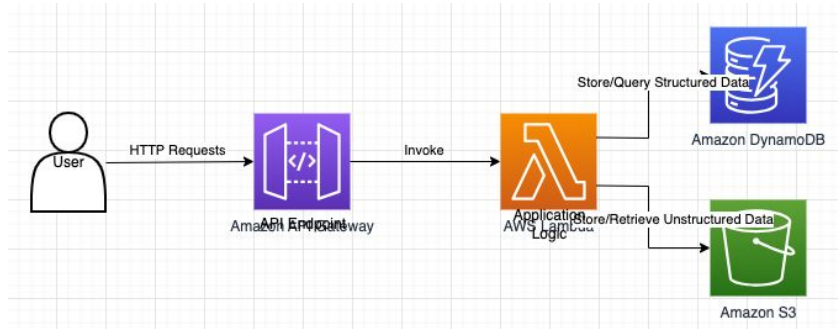
5mo • Edited •

I have another great example of how Amazon Q Developer increased my productivity. Have you tried using Amazon Q Developer to create code from diagrams? You should. Here is why.

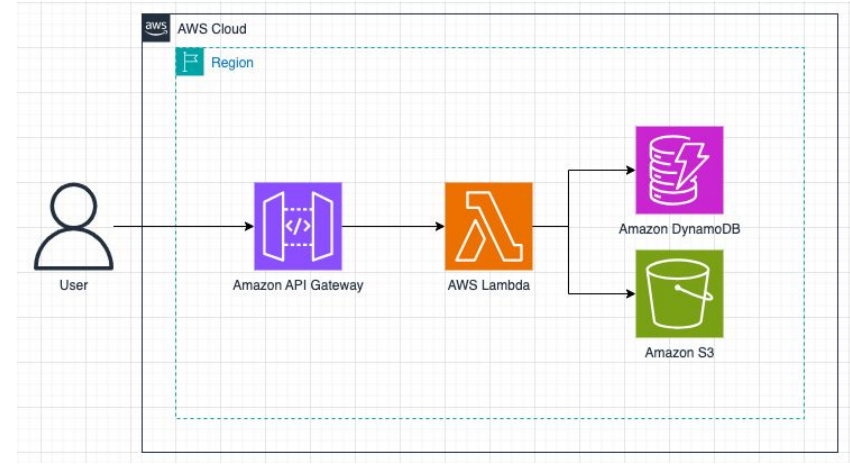


# Amazon Q CLI “chat”

Just User Prompt



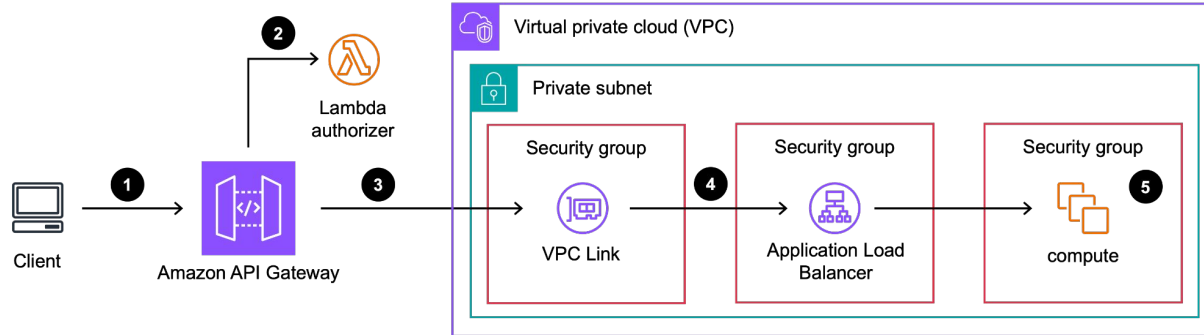
User Prompt with context added to:  
`.amazonq/rules/drawio-standards.md`



#OSN2025

# Vibe Architecting

create an AWS CDK project with the following components: 1) a API Gateway HTTP API targeting an ALB as the integration. 2) create a lambda custom authorizer for the HTTP API that adds data to the context. 3) have the integration from REST API to ALB add the context as a custom HTTP header. 4) create a new VPC for the ALB, and add a VPC Link to the HTTP API. 5) add another lambda function as the target for the ALB and print out the custom header. 6) recommend any security group logic to only allow HTTP traffic from the API Gateway HTTP API



What is your experience using AI tools with diagrams?



# But this is not perfect

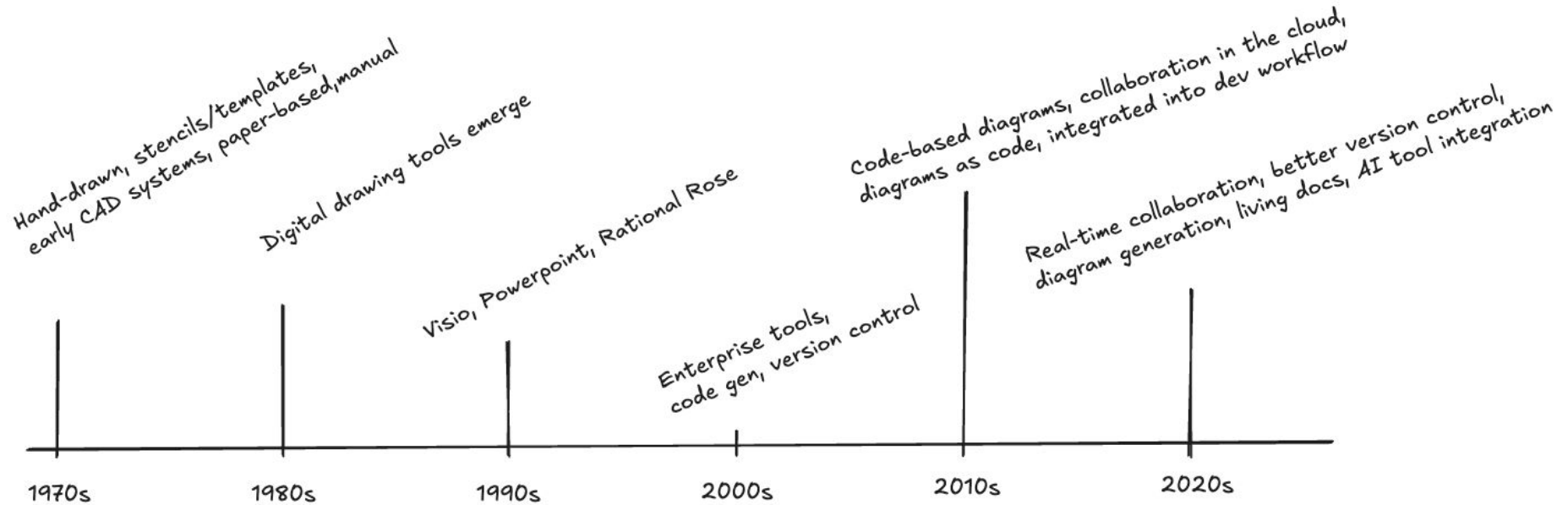
## Benefits

- Time savings in creating initial diagrams
- Consistent visual representation
- Basic analysis capabilities

## Limitations

- Limited understanding of your company's context - business domain, patterns, ADRs, standards, rules
- Still requires human validation
- Not deeply integrated with other artifacts

# This workflow is changing



# Context is key

#OSN2025

# Beyond diagrams

Integrated knowledge

#OSN2025

Architectural understanding comes from multiple sources:

- Diagram images (visual structure) or file formats (text / as-code)
- Source code and comments (implementation)
- Infrastructure as Code (deployment)
- ADRs and design documents (decisions)
- Security policies (constraints)
- Change history (evolution)
- Usage / Observability (metrics)

# Kevin's OSN 2023 presentation

## Software Architecture and Design InfoQ Trends Report - April 2023

“Architecture Decision Records (ADRs) are now commonly recognized as a way to document and communicate design decisions.”

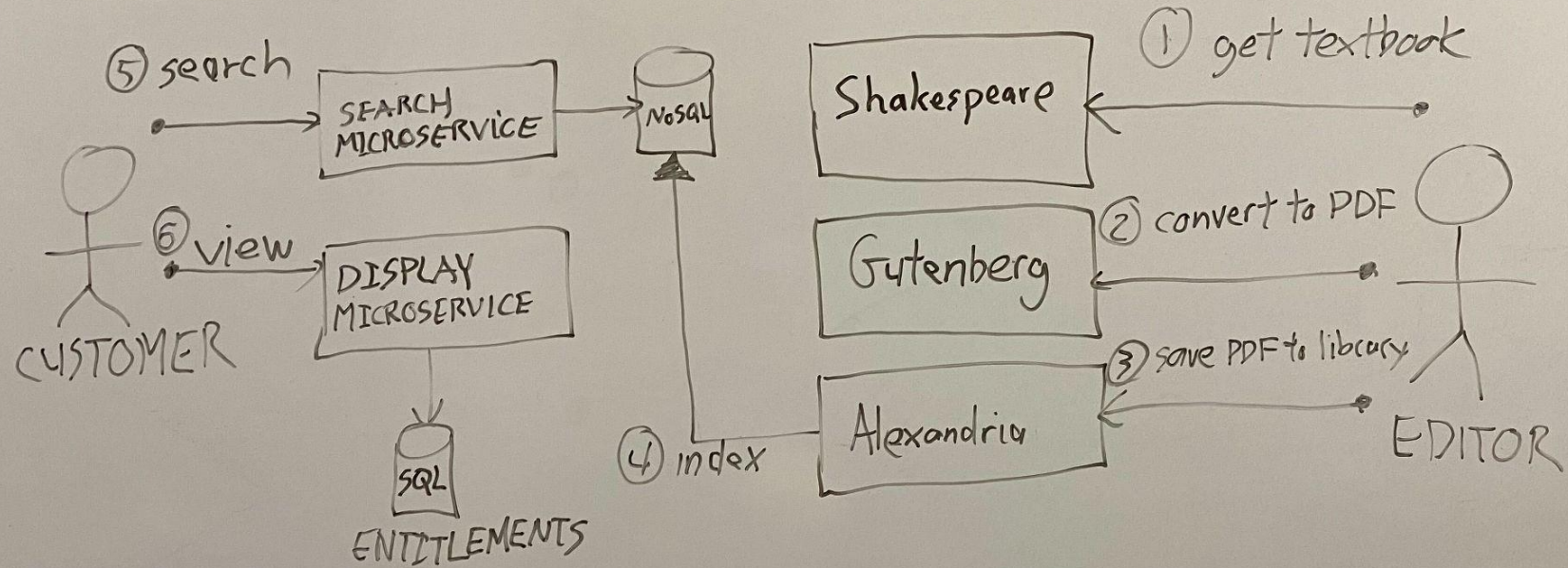
“Architects are always looking for improvements on how to document, communicate, and understand decisions. This may be another area where large language models will play a role in the future, acting as forensic archeologists to comb through ADRs and git history.”

The future  
is now

## Kevin's recent Q Developer chat

can you review the changes since git commit 6a7ebc6adaa1b5fc63b31e7ef0ae10b32a2239e0 and provide a summary? if a similar change was made to multiple files, just give me a count of the number of files and short description. If a major change was made to a file, list out that file path and summary of the major change.

# Whiteboard architecture of a content publishing system

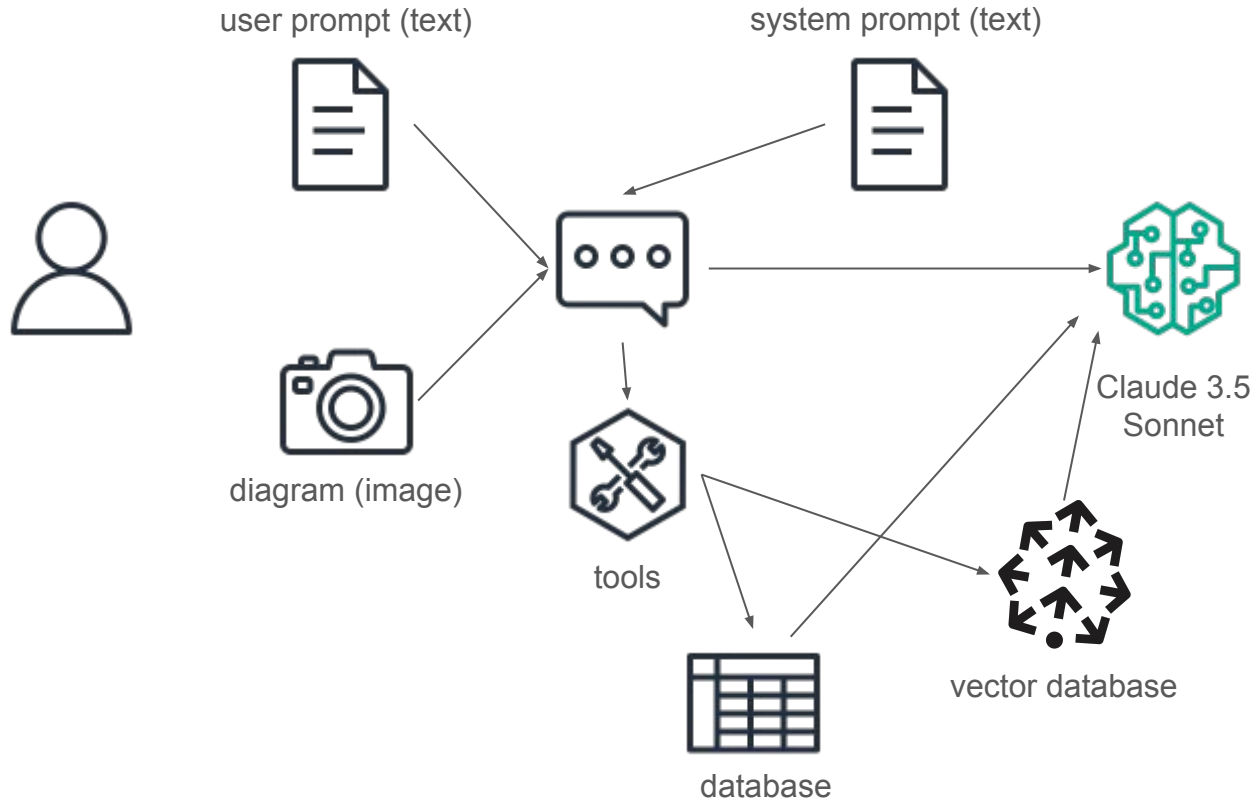


“describe the data flow of this diagram based on the order of the numbered circles”

1. Get Textbook: The process begins with the Editor creating or retrieving a textbook from the Shakespeare service, which is the editorial platform.
2. Convert to PDF: The textbook is then sent to the Gutenberg service, which converts it into a PDF format.
3. Save PDF to Library: The converted PDF is saved to the Alexandria service, which acts as the library for storing published textbooks.
4. Index: The newly saved textbook in Alexandria is indexed, likely updating the NoSQL database to make the content searchable.
5. Search: When a Customer initiates a search, their request is processed by the Search Microservice, which queries the NoSQL database to find relevant content.
6. View: After finding the desired content, the Customer can view it through the Display Microservice. This microservice likely checks the SQL Entitlements database to ensure the customer has the right to access the content before displaying it.



# Knowledge integration in action



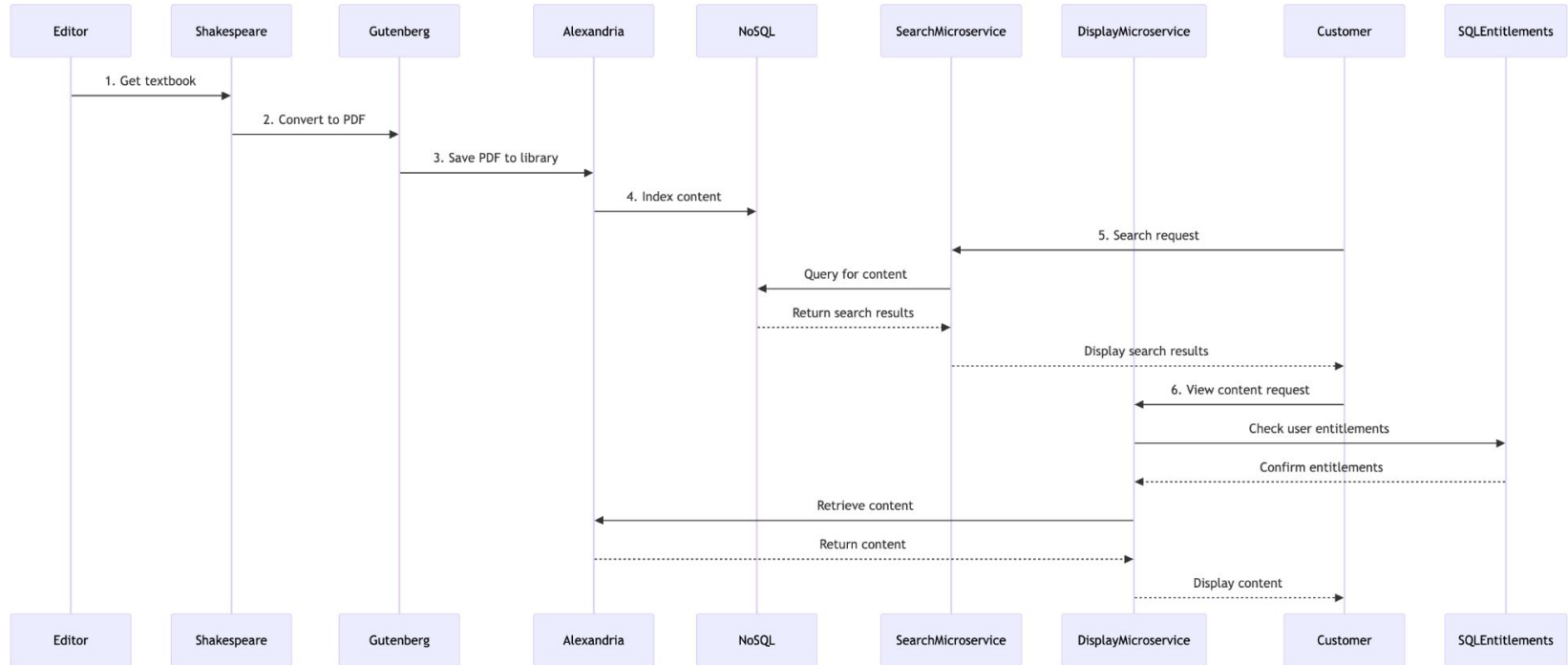
# Diagrams-as-Code

“There are benefits to using [diagrams as code] tools over the heavier alternatives, including easy version control and the ability to generate the DSLs from many sources.”

Thoughtworks Technology Radar Blip (Oct 28, 2020)

**Mermaid** is a JavaScript based diagramming and charting tool that renders Markdown-inspired text definitions to create and modify diagrams dynamically.

“convert into a Mermaid formatted sequence diagram”



# Multi-perspective visualization

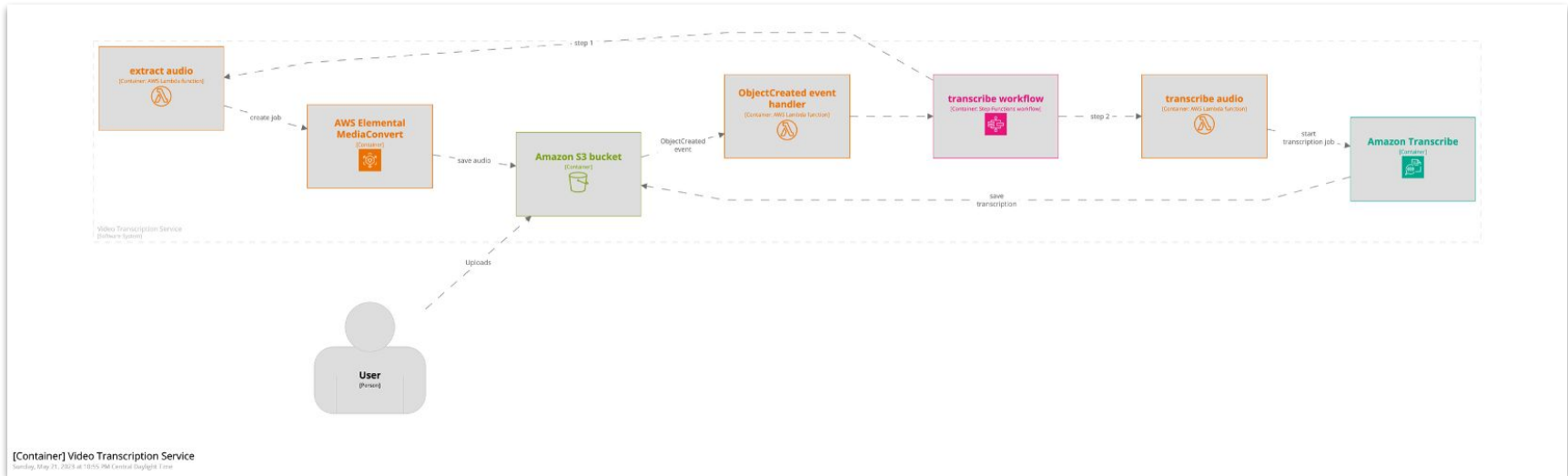
Different stakeholders need  
different views

- Business: High-level capabilities
- Security: Trust boundaries
- Operations: Deployment topology
- Developers: Component interfaces
- Product: Processes
- UX: User journeys

# C4 model for visualizing software architecture

1. System context diagram
2. Container diagram
3. Component diagram
4. Code diagram

Structurizr builds upon "diagrams as code", allowing you to create multiple software architecture diagrams from a single model.



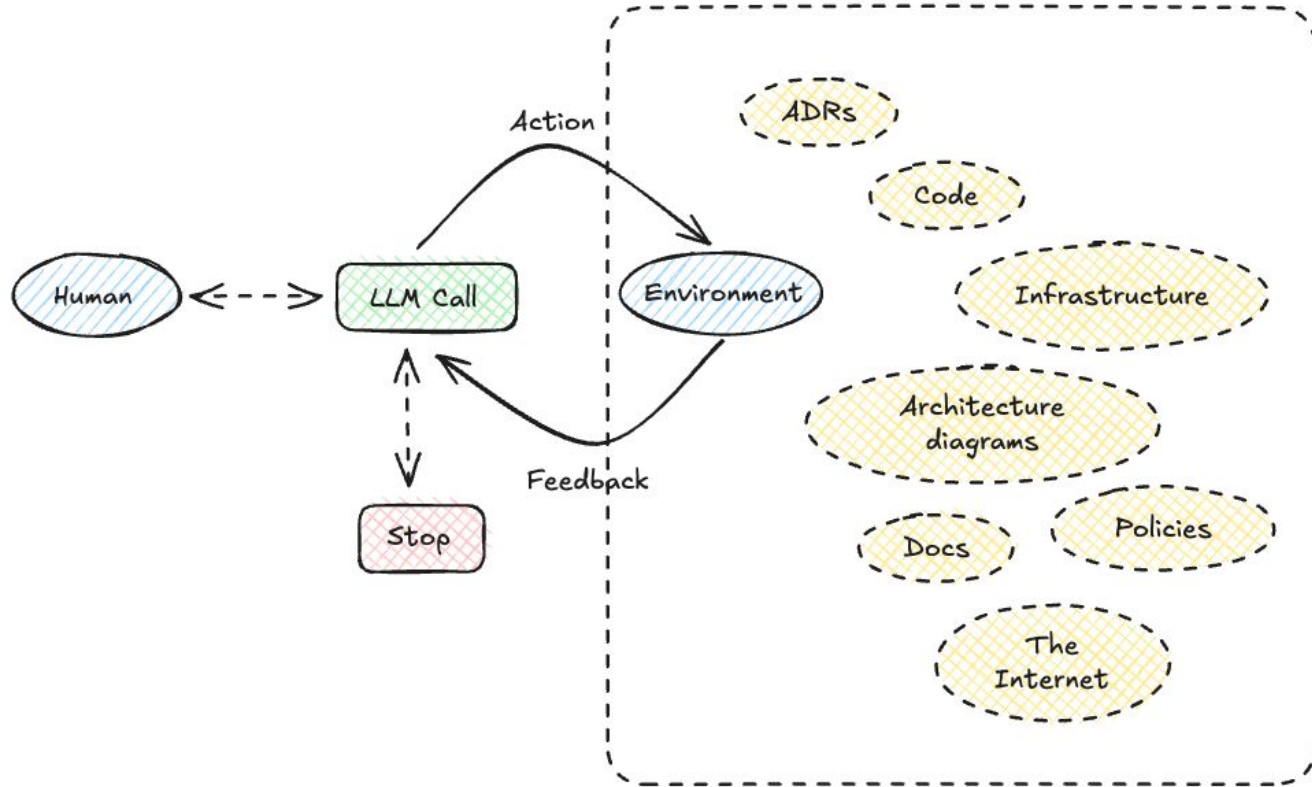
# Autonomous Agents

To scale tasks

#OSN2025

- Deploy an agent or a team of agents to:
  - Evaluate decisions
  - Migrate a codebase when an SDK/API changes
  - Respond to incidents and troubleshoot issues
  - Evaluate which is the right chunking approach, embedding model, vector dimension, or search strategy based on usage patterns

# Autonomous agent



# Practical takeaways

- Know where your data is and what state it is in
- Know what good code/diagrams/results look right for you
- Know what you're trying to accomplish
- Know the tools in your tool box and how to use them/what they are for
- Try things out
- Push past the hype



# Thank you!

Any questions?



/in/kevinhakanson



/in/jennapederson

#OSN2025

Please share your feedback



<https://pulse.aws/survey/ZQYU6TTY>