

Amazon Q Developer

The most capable generative AI-powered assistant for software development



Michael Lin

Sr. Solutions Architect
Amazon Web Services



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AI coding assistants' adoption

By 2028, 75% of enterprise software engineers will use AI code assistants, up from less than 10% in early 2023.

Gartner® Press Release, Gartner Says 75% of Enterprise Software Engineers Will Use AI Code Assistants by 2028, April 11, 2024,
<https://www.gartner.com/en/newsroom/press-releases/2024-04-11-gartner-says-75-percent-of-enterprise-software-engineers-will-use-ai-code-assistants-by-2028>. GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved.

Amazon Q Developer



Helps developers and IT professionals build faster across the entire software development lifecycle (SDLC)

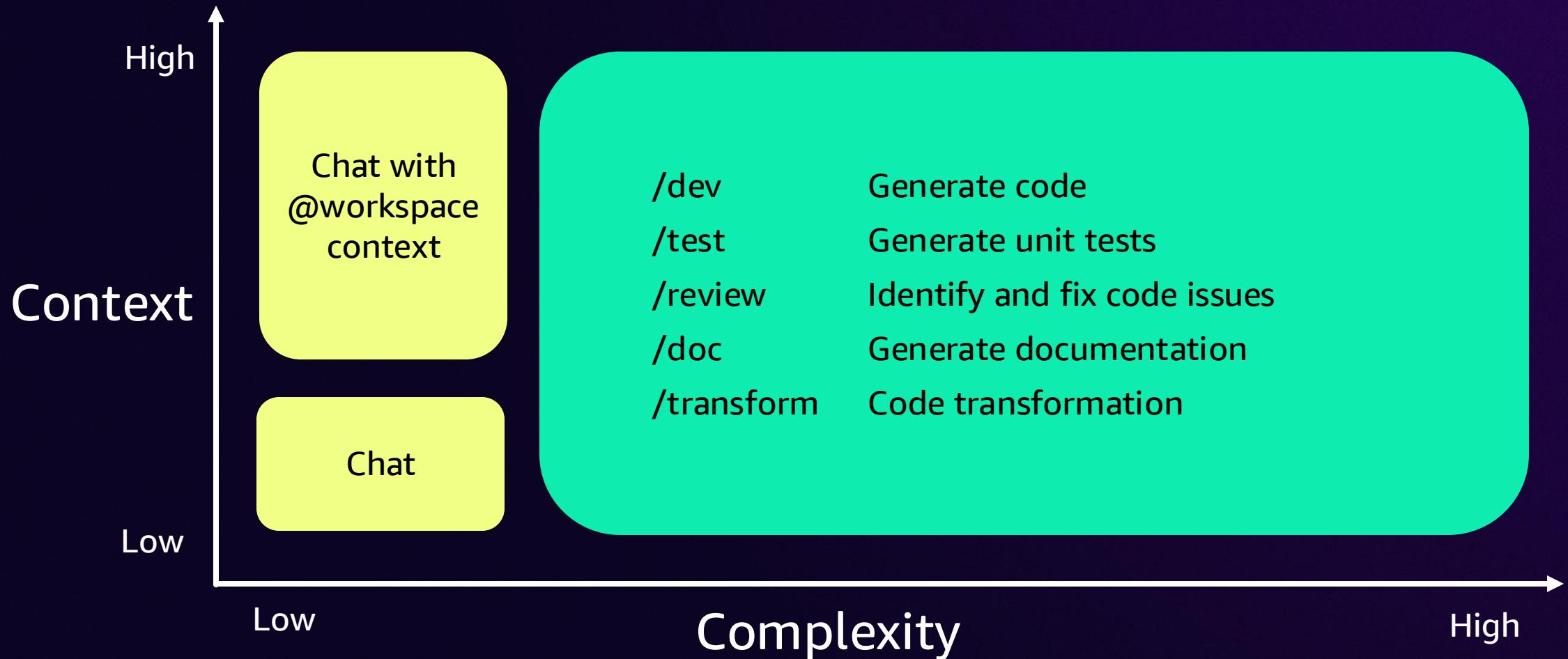
Most accurate coding recommendations

Agents can autonomously help you implement features, refactor code, perform software upgrades and more

Best-in-class security vulnerability scanning and remediation

Amazon Q is built with security and privacy in mind from the start, making it easier for organizations to use generative AI safely.

Positioning of Amazon Q Developer and agents



Recent Updates on Agentic Coding in CLI and IDE

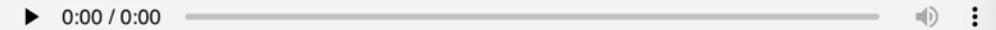
A lightning fast, new agentic coding experience within the Amazon Q Developer CLI

by Brian Beach | on 06 MAR 2025 | in [Amazon Q](#), [Amazon Q Developer](#), [Announcements](#) | [Permalink](#) | [Share](#)

Earlier today, [Amazon Q Developer](#) announced [an enhanced CLI agent](#) within the [Amazon Q command line interface \(CLI\)](#). With this announcement, Q Developer brings the latest agentic experience to the CLI that provide a more dynamic, interactive coding experience that works with you, and iteratively makes changes based on your feedback. Amazon Q Developer can now use the information in your CLI environment to help you read and write files locally, query AWS resources write code, or automatically debug issues.

Amazon Q Developer elevates the IDE experience with new agentic coding experience

by Elizabeth Fuentes | on 02 MAY 2025 | in [Amazon Q Developer](#), [Announcements](#), [AWS CLI](#), [Developer Tools](#), [Featured](#), [Launch](#), [News](#) | [Permalink](#) | [Comments](#) | [Share](#)



Today, [Amazon Q Developer](#) introduces a new, interactive, agentic coding experience that is now available in the [integrated development environments \(IDE\)](#) for [Visual Studio Code](#). This experience brings interactive coding capabilities, building upon existing prompt-based features. You now have a natural, real-time collaborative partner working alongside you while writing code, creating documentation, running tests, and reviewing changes.

<https://aws.amazon.com/blogs/devops/introducing-the-enhanced-command-line-interface-in-amazon-q-developer/>
<https://aws.amazon.com/blogs/aws/amazon-q-developer-elevates-the-ide-experience-with-new-agentic-coding-experience/>

Latest MCP Support in CLI and Context-Aware IDE

Extend the Amazon Q Developer CLI with Model Context Protocol (MCP) for Richer Context

by Brian Beach | on 29 APR 2025 | in [Amazon Aurora](#), [Amazon Q Developer](#), [Announcements](#), [PostgreSQL compatible](#), [RDS for PostgreSQL](#) | [Permalink](#) |  Share

Earlier today, [Amazon Q Developer](#) announced [Model Context Protocol \(MCP\) support](#) in the command line interface (CLI). Developers can connect external data sources to Amazon Q Developer CLI with MCP support for more context-aware responses. By integrating MCP tools and prompts into Q Developer CLI, you get access to an expansive list of pre-built integrations or any MCP Servers that support `stdio`. This extra context helps Q Developer write more accurate code, understand your data structures, generate appropriate unit tests, create database documentation, and execute precise queries, all without needing to develop custom integration code. By extending Q Developer with MCP tools and prompts, developers can execute development tasks faster, streamlining the developer experience. At AWS, we're committed to supporting popular open source protocols for agents like Model Context Protocol (MCP) proposed by Anthropic. We'll continue to support this effort by extending this functionality within the Amazon Q Developer IDE plugins in the coming weeks.

Use Model Context Protocol with Amazon Q Developer for context-aware IDE workflows

by Ritik Khatwani | on 12 JUN 2025 | in [Amazon Q](#), [Amazon Q Developer](#), [Announcements](#), [Developer Tools](#), [Generative AI](#) | [Permalink](#) |  Share

Earlier today, [Amazon Q Developer](#) announced Model Context Protocol (MCP) support in their Integrated Development Environment (IDE) plugins for Visual Studio Code and JetBrains. This allows developers to connect external tools or MCP servers to Q Developer, enabling more context-aware responses and complex workflows. MCP support has already been available in [Amazon Q Developer for Command Line](#) since April 29, 2025.

<https://aws.amazon.com/blogs/devops/extend-the-amazon-q-developer-cli-with-mcp/>
<https://aws.amazon.com/blogs/devops/use-model-context-protocol-with-amazon-q-developer-for-context-aware-ide-workflows/>



Amazon Q dashboard

METRICS WE TRACK

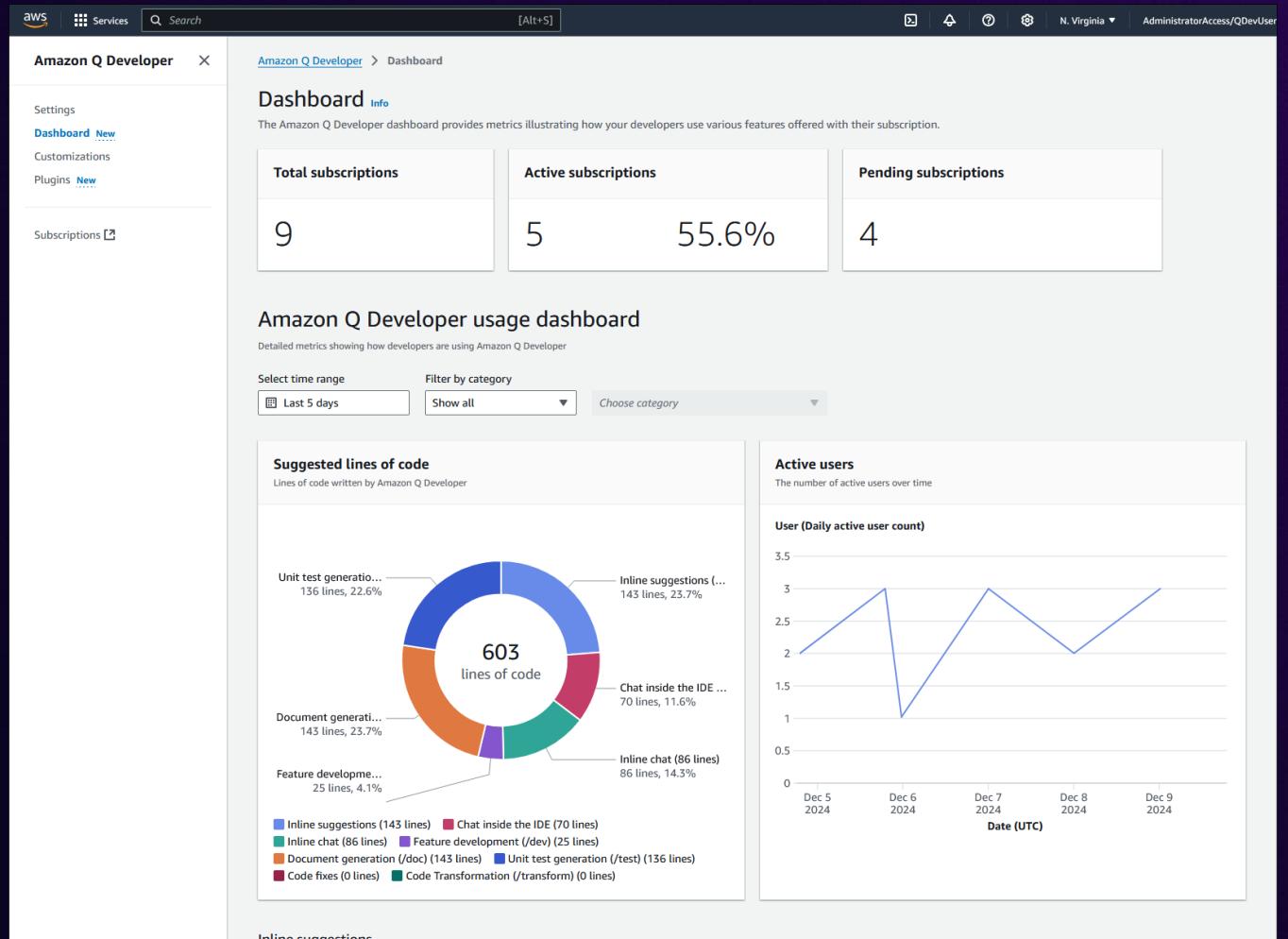
User activity

Lines of code generated

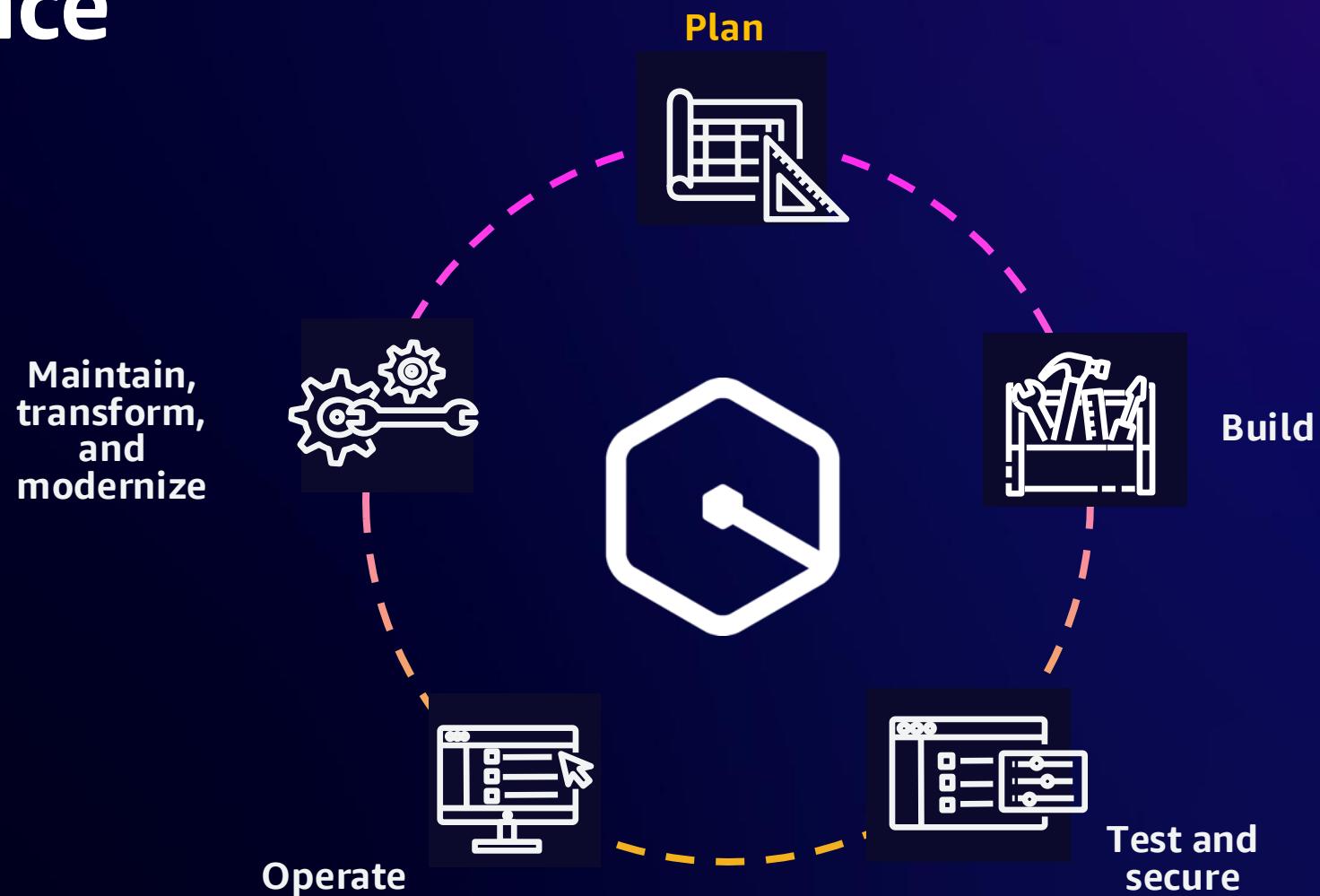
Acceptance rate

Percentage of code written
by Amazon Q Developer

Accepted recommendations
with references



Amazon Q Developer transforms the development experience



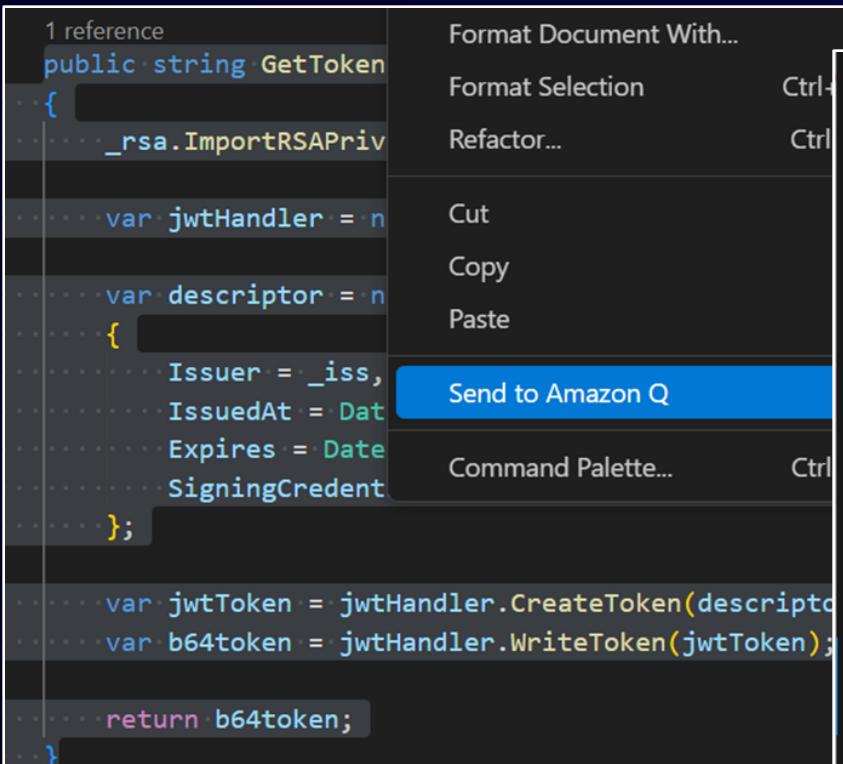
Code understanding



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Code explanation

HAVE Q FOR DEVELOPER EXPLAIN LEGACY CODE WRITTEN BY DEVS LONG GONE...



The provided code block is a method named `GetToken()` that generates a JSON Web Token (JWT) using the RSA algorithm. Here's a breakdown of what the code is doing:

1. The method starts by importing the RSA private key into an `RSACryptoServiceProvider` object named `_rsa`.
2. It then creates a `JwtSecurityTokenHandler` object, which is used to create and manipulate JWT tokens.
3. Next, it creates a `SecurityTokenDescriptor` object, which contains the following properties:
 - `Issuer`: The issuer of the JWT token, stored in the `_iss` field.
 - `IssuedAt`: The time the token was issued, set to 30 seconds before the current time.
 - `Expires`: The time the token expires, set to 10 minutes after the current time.
 - `SigningCredentials`: The signing credentials used to sign the JWT token, which include the RSA private key and the SHA256 algorithm.
4. The `JwtSecurityTokenHandler` is then used to create a JWT token based on the `SecurityTokenDescriptor`.
5. Finally, the method converts the JWT token to a base64-encoded string and returns it.

This code is likely used to generate a JWT token that can be used for authentication or authorization purposes, such as in a web application or API. The RSA private key is used to sign the token, ensuring its integrity and authenticity.



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using @workspace to identify files

The screenshot shows the Amazon Q: Chat interface. The user has asked: "@workspace what are the main files that interact with Amazon Bedrock and provide an explanation about what each one does?". The AI response is as follows:

Based on the provided code snippets, there are several key files that interact with Amazon Bedrock. Let's break them down and explain what each one does:

1. backend/lib/backend-stack.ts

This file is crucial for setting up the infrastructure for Amazon Bedrock integration. It defines the AWS CDK stack and configures various AWS resources. Here's what it does:

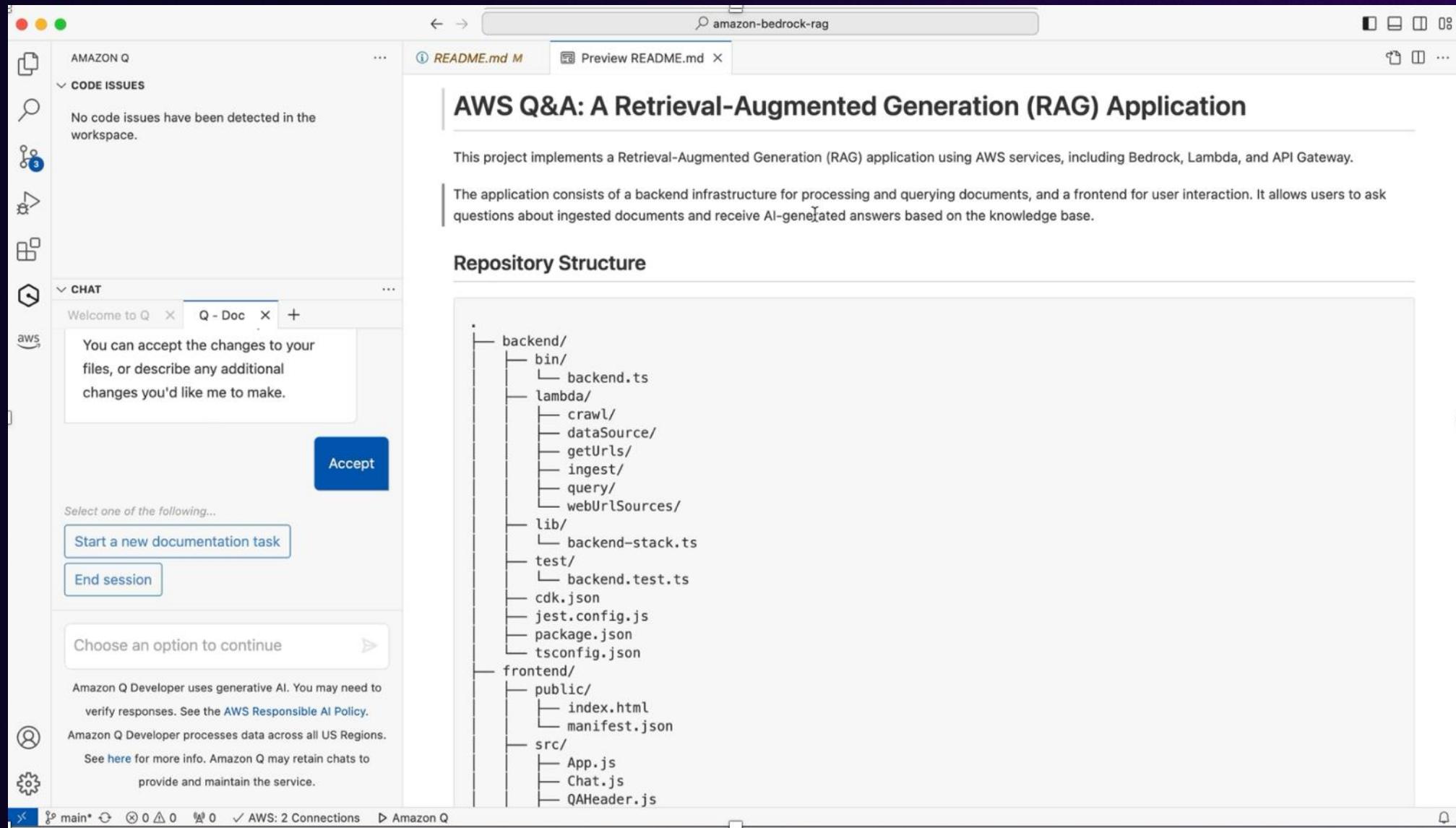
- Creates an Amazon Bedrock Knowledge Base
- Sets up an S3 bucket as a data source for the Knowledge Base
- Configures an

At the bottom, there is a footer note: "Amazon Q Developer uses generative AI. You may need to verify responses. See the [AWS Responsible AI Policy](#)".

On the right side of the interface, there is a large gray X watermark. The interface includes standard file navigation icons (File, Edit, View, Insert, Tools, Window, Help) and a status bar at the bottom.

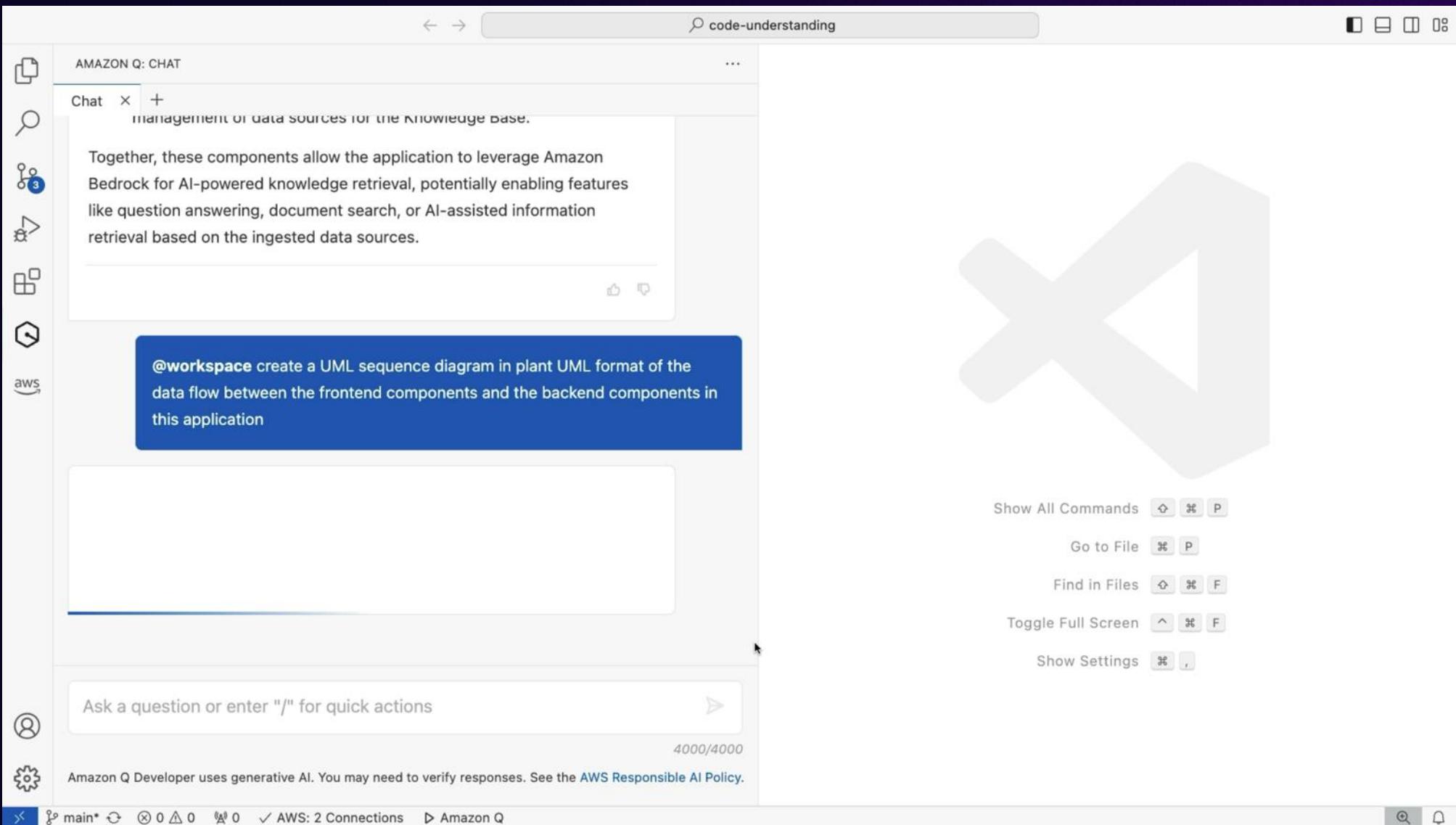


Amazon Q Developer agent for documentation

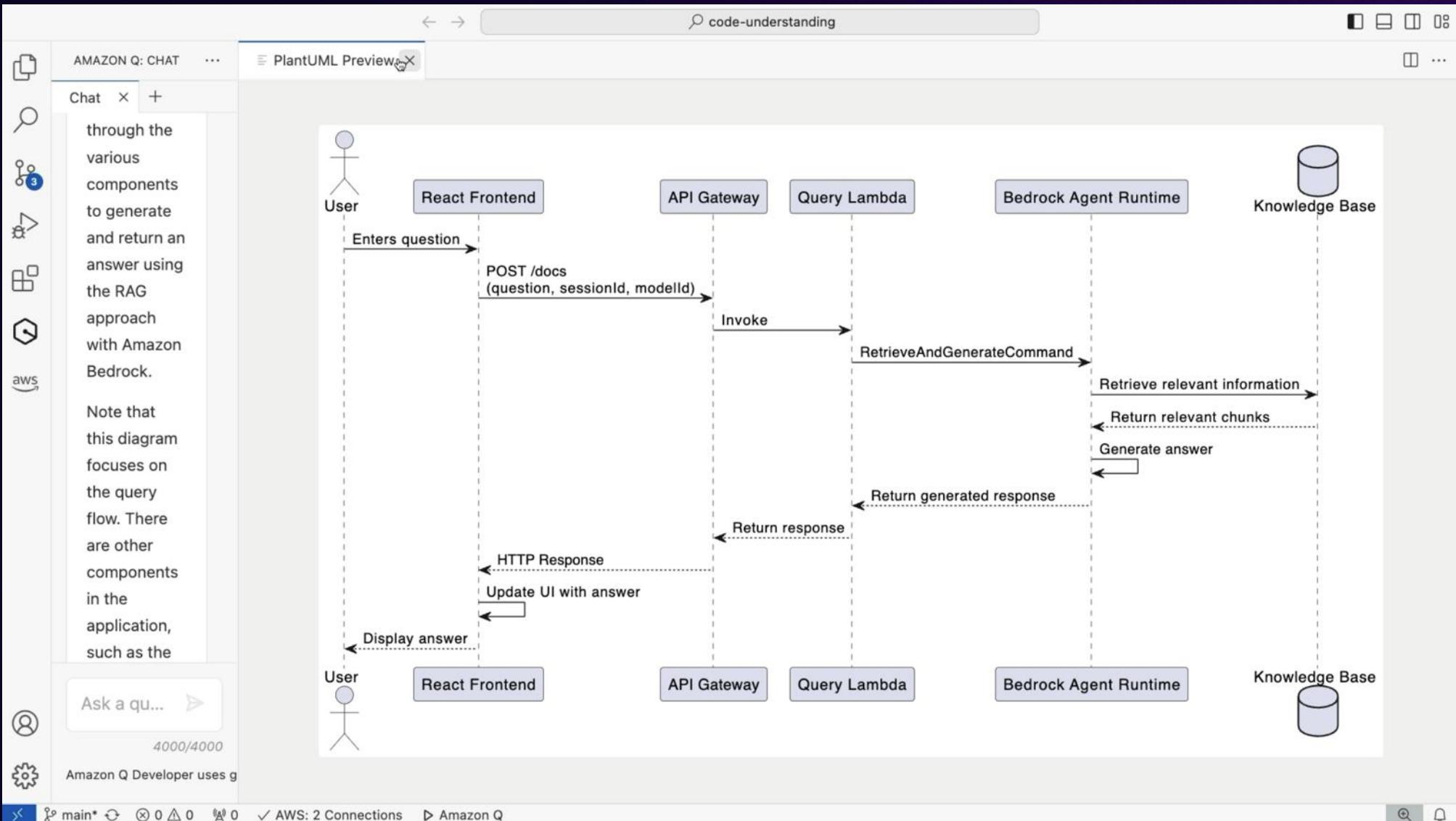


Visualization

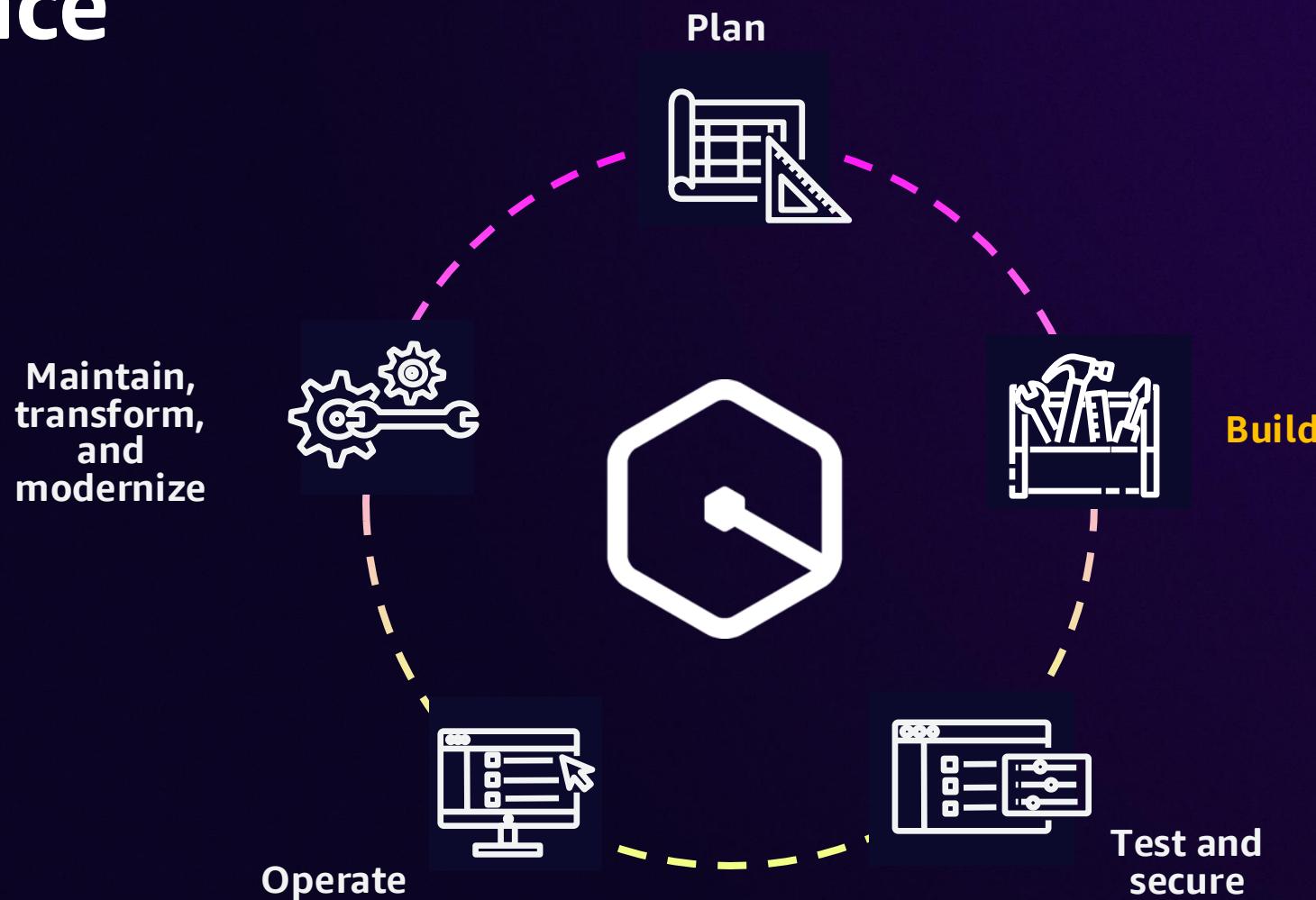
Create a sequence diagram with PlantUML



Create a sequence diagram with PlantUML

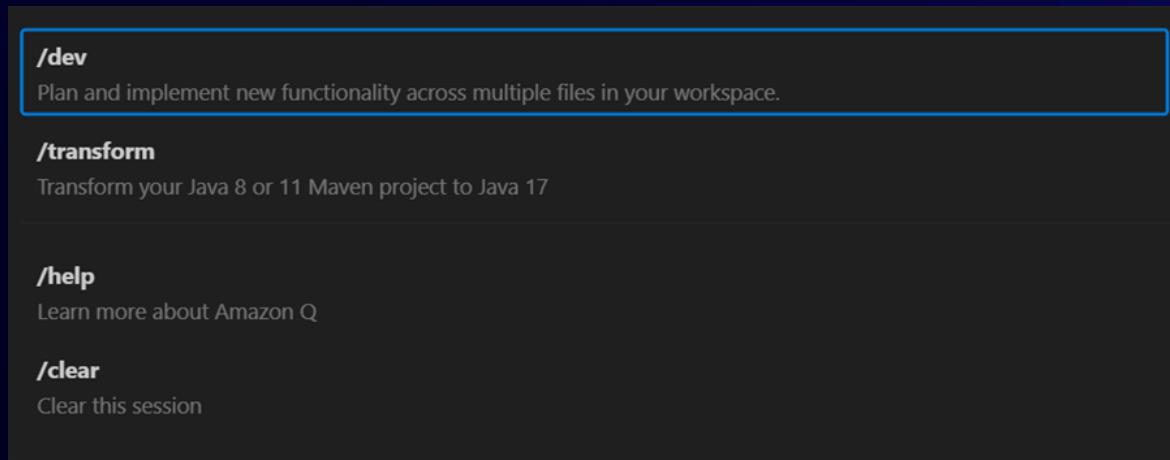


Amazon Q Developer transforms the development experience

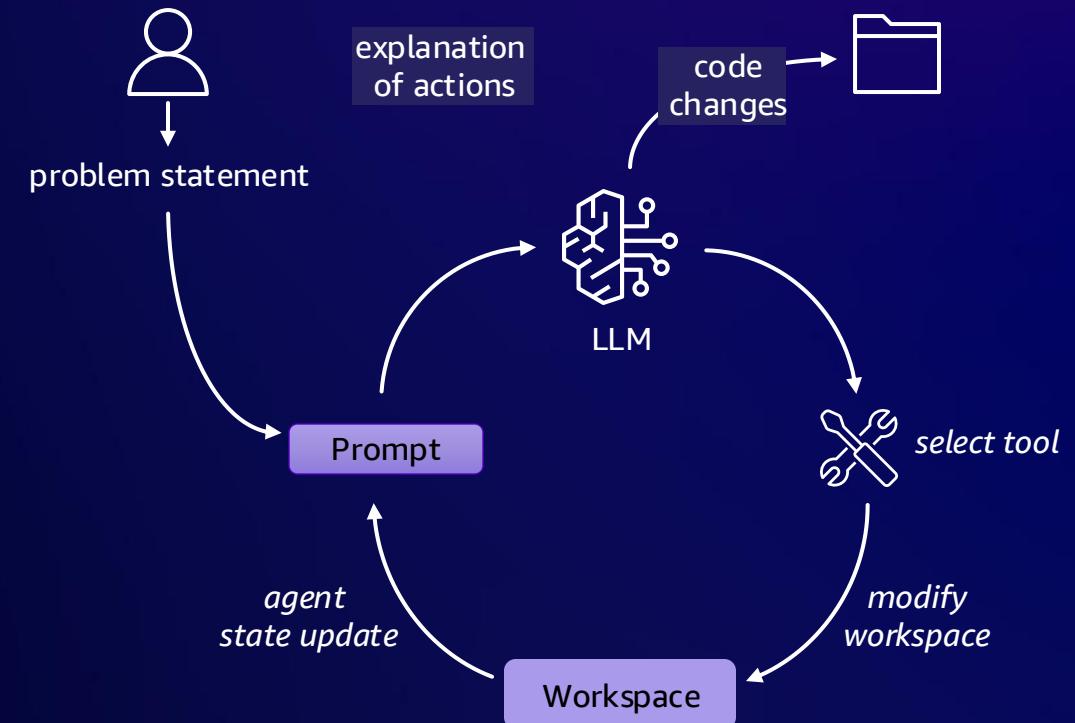


Developer agent for software development

PLAN AND IMPLEMENT NEW FUNCTIONALITY ACROSS MULTIPLE FILES IN YOUR WORKSPACE



Currently available in the Amazon Q extension for VS Code and in Amazon CodeCatalyst. Support is planned for other IDEs.





AMAZON Q: CHAT

Chat x Q - Dev x +

Welcome to feature development.

I can generate code to implement new functionality across your workspace. We'll start by discussing an implementation plan, and then we can review and regenerate code based on your feedback.

To get started, describe the task you are trying to accomplish.

Create a class to save a log of the tokens generated to dynamodb

Ok, let me create a plan. This may take a few minutes.

1. Create a new class for logging token generation to DynamoDB

1. Create a new file: `TokenLogger.cs`

2. Import the required namespaces:

- `Amazon.DynamoDBv2`

- `Amazon.DynamoDBv2.DataModel`

3. Define a new class `TokenLogger` that implements `IDisposable`

4. Define a constructor that initializes the DynamoDB client and table name

5. Define a method `LogToken` with the following input parameters:

- `string token`

- `DateTime expirationTime`

6. Inside `LogToken`, create a new instance of a DynamoDB model class

How can this plan be improved?

C# JwtBuilder.cs

C# Function.cs X

C# Function.cs > Function

```
1  using Amazon.Lambda.Core;
2  using Amazon.Lambda.APIGatewayEvents;
3  using Amazon.SimpleNotificationService;
4  using Amazon.SimpleNotificationService.Model;
5  using Octokit;
6  using System.Security.Cryptography;
7  using System.Text.Json;
8  using System.Linq;
9
10 // Assembly attribute to enable the Lambda function's JSON input to be converted into a .NET
11 [assembly: LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambda
12
13 namespace orgbranchprotection;
14
15 1 reference
16 public class Function
17 {
18     3 references
19     private static IAmazonSimpleNotificationService? client;
20     3 references
21     private static JwtBuilder? jwtBuilder;
22     4 references
23     private static AccessToken? accessToken;
24     2 references
25     private readonly long Installation = 11111111;
26     2 references
27     private readonly string SnsTopicArn = "";
28
29     //test comment
30
31     0 references
32     public Function()
33     {
34         if (jwtBuilder != null) return;
35     }
36 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG

NuGet

Determining projects to restore...

All projects are up-to-date for restore.

Describe the task
Q tells you its plan to accomplish the task

1. Create a new class for logging token generation to DynamoDB

1. Create a new file: `TokenLogger.cs`
2. Import the required namespaces:
 - `Amazon.DynamoDBv2`
 - `Amazon.DynamoDBv2.DataModel`
3. Define a new class `TokenLogger` that implements `IDisposable`
4. Define a constructor that initializes the DynamoDB client and table name
5. Define a method `LogToken` with the following input parameters:
 - `string token`
 - `DateTime expirationTime`
6. Inside `LogToken`, create a new instance of a DynamoDB model class representing a log entry
7. Save the log entry to the DynamoDB table using the client

2. Update the existing `JwtBuilder` class

1. `./JwtBuilder.cs`
2. Import the `TokenLogger` class
3. Instantiate the `TokenLogger` in the constructor
4. In the `GetToken` method, call the `LogToken` method of `TokenLogger` with the generated token and its expiration time

3. Add tests for the `TokenLogger` class

1. Create a new file: `TokenLoggerTests.cs`
2. Import the required namespaces for unit testing
3. Define a test class `TokenLoggerTests`

4. Define a test method `TestLogToken` with the following steps:

- Arrange:
 1. Set up mock DynamoDB client and table
 2. Create an instance of `TokenLogger`
 - Act:
 - Call `LogToken` with a sample token and expiration time
 - Assert:
 - Verify that the log entry was saved to the DynamoDB table
4. Update the project file to include the new files
 1. `./orgbranchprotection.csproj`
 2. Add the new `TokenLogger.cs` and `TokenLoggerTests.cs` files to the project
 5. Update the deployment configuration file for DynamoDB access
 1. `./aws-lambda-tools-defaults.json`
 2. Add the required DynamoDB permissions and environment variables for the DynamoDB table name
 6. Update the GitHub workflow or deployment scripts to run the new tests
 1. Modify the existing GitHub Actions workflow or deployment scripts
 2. Add a step to run the new `TokenLoggerTests` unit tests

By following these steps, a new class `TokenLogger` will be introduced to log token generation events to a DynamoDB table. The existing `JwtBuilder` class will be updated to use the `TokenLogger`. Additionally, unit tests will be added to verify the functionality of the `TokenLogger` class. The project file and deployment configuration will be updated to include the new files and DynamoDB access respectively. Finally, the GitHub Actions workflow or deployment scripts will be modified to run the new tests.

Note: To handle current demand, Q will use a smaller, higher-speed model for code generation.

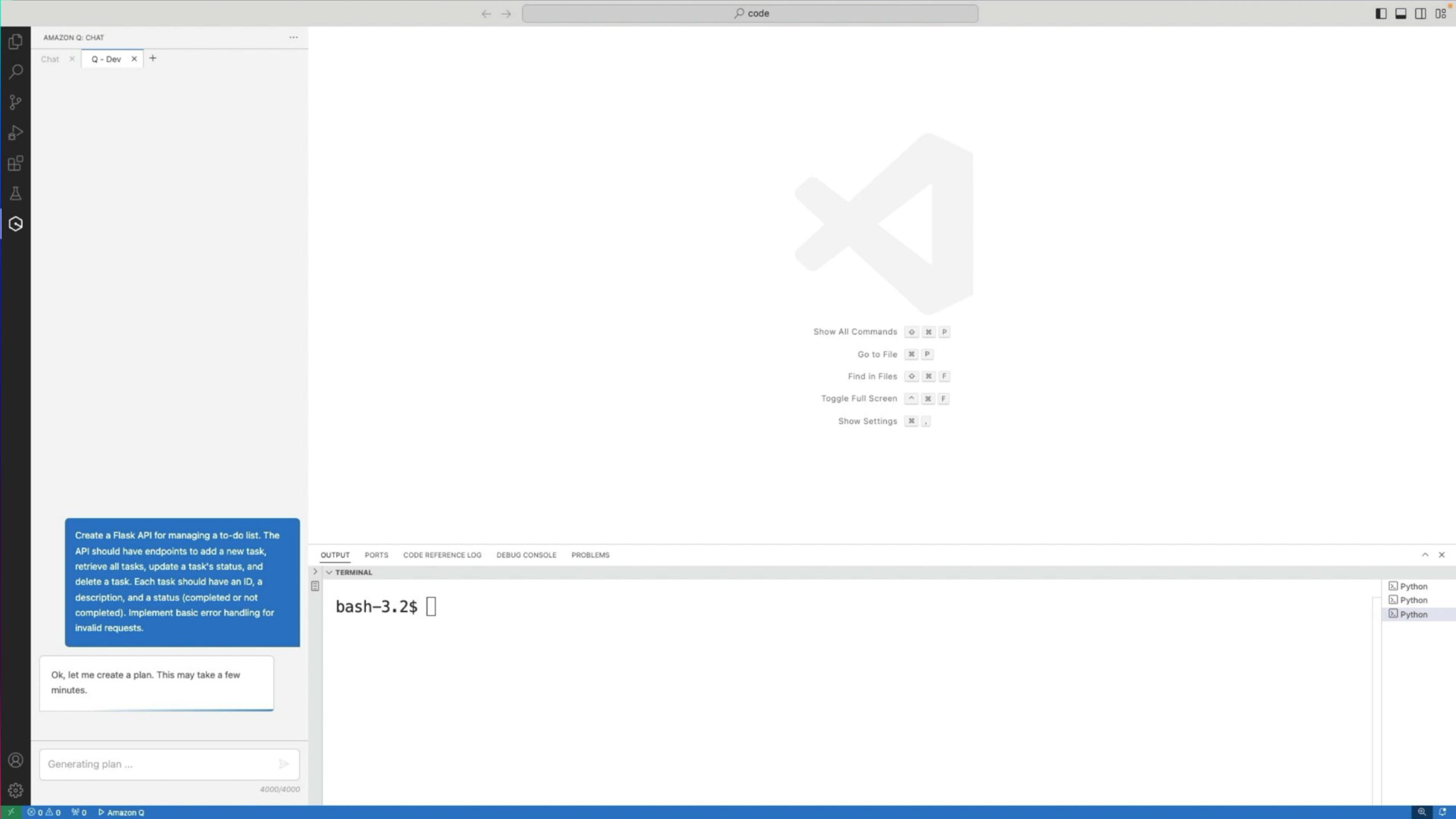


Sample prompt



Create a Flask API for managing a to-do list. The API should have endpoints to add a new task, retrieve all tasks, update a task status, and delete a task.

Ok, let me create a plan. This may take a few minutes.



AMAZON Q: CHAT

Chat Q - Dev +

← →

code

□ □ □



Show All Commands ⌘ ⌘ P

Go to File ⌘ P

Find in Files ⌘ ⌘ F

Toggle Full Screen ⌘ ⌘ F

Show Settings ⌘ ,

OUTPUT PORTS CODE REFERENCE LOG DEBUG CONSOLE PROBLEMS

> TERMINAL

bash-3.2\$

Create a Flask API for managing a to-do list. The API should have endpoints to add a new task, retrieve all tasks, update a task's status, and delete a task. Each task should have an ID, a description, and a status (completed or not completed). Implement basic error handling for invalid requests.

Ok, let me create a plan. This may take a few minutes.

Generating plan ...

4000/4000

Python
Python
Python



AMAZON Q: CHAT

Chat X Q - Dev X +

I suggest the following changes:

app.py:

1. Import required modules
 - o import Flask, request, jsonify from flask
 - o Import uuid for generating unique IDs
2. Initialize Flask app
3. Create an empty list to store tasks

2. Create a route for adding a new task
 - o Use POST method
 - o Generate a unique ID using uuid
 - o Get task description from request JSON
 - o Create a new task dictionary with ID, description, and status
 - o Append the new task to the tasks list
 - o Return the created task with a 201 status code
2. Implement error handling for invalid requests
 - Check if description is provided and is a non-empty string
 - Return an error message with a 400 status code if invalid
3. Create a route for retrieving all tasks
 - o Use GET method
 - o Return the list of all tasks
4. Create a route for updating a task's status
 - o Use PUT method
 - o Get task ID and new status from

How can this plan be improved? ➤

4000/4000



Show All Commands ⌘ ⌘ P
Go to File ⌘ P
Find in Files ⌘ ⌘ F
Toggle Full Screen ⌘ ⌘ F
Show Settings ⌘ ,

OUTPUT PORTS CODE REFERENCE LOG DEBUG CONSOLE PROBLEMS

> ▾ TERMINAL

bash-3.2\$

Python Python Python



The image shows a screenshot of a code editor interface, likely Visual Studio Code, displaying a Python file named `app.py`. The code implements a simple REST API for managing tasks using the Flask framework.

```
1 from flask import Flask, request, jsonify
2 import uuid
3
4 app = Flask(__name__)
5
6 tasks = []
7
8 @app.route('/tasks', methods=['POST'])
9 def add_task():
10     data = request.json
11     if not data or 'description' not in data or not isinstance(data['description'], str) or data
12         ['description'].strip() == '':
13             return jsonify({"error": "Invalid task description"}), 400
14
15     task = {
16         "id": str(uuid.uuid4()),
17         "description": data['description'],
18         "status": "not completed"
19     }
20     tasks.append(task)
21     return jsonify(task), 201
```

Below the code editor, there is a terminal window titled "TERMINAL" showing a bash prompt:

```
bash-3.2$
```

The status bar at the bottom of the screen displays various system and application icons, including the Amazon logo, a search icon, and the text "Amazon Q".

← → Home Workspaces API Network

Search Postman

Invite ⚙️ 🔔 Upgrade

My Workspace New Import GET Get Tasks POST Add Task +

Amazon Q LIL / Add Task

POST http://127.0.0.1:5000/tasks

Params Authorization Headers (8) Body * Pre-request Script Tests Settings Cookies

Body: JSON

```
1 {  
2   "description": "check in code to repo",  
3   "status": "not started"  
4 }
```

Send Save

POST 3.5-Add User

POST 3.5-Get User

POST Add Task

GET Get Tasks

Amazon Q LIL

ChatGPT ChatGPT Action NASA APIs NDC OpenAI API

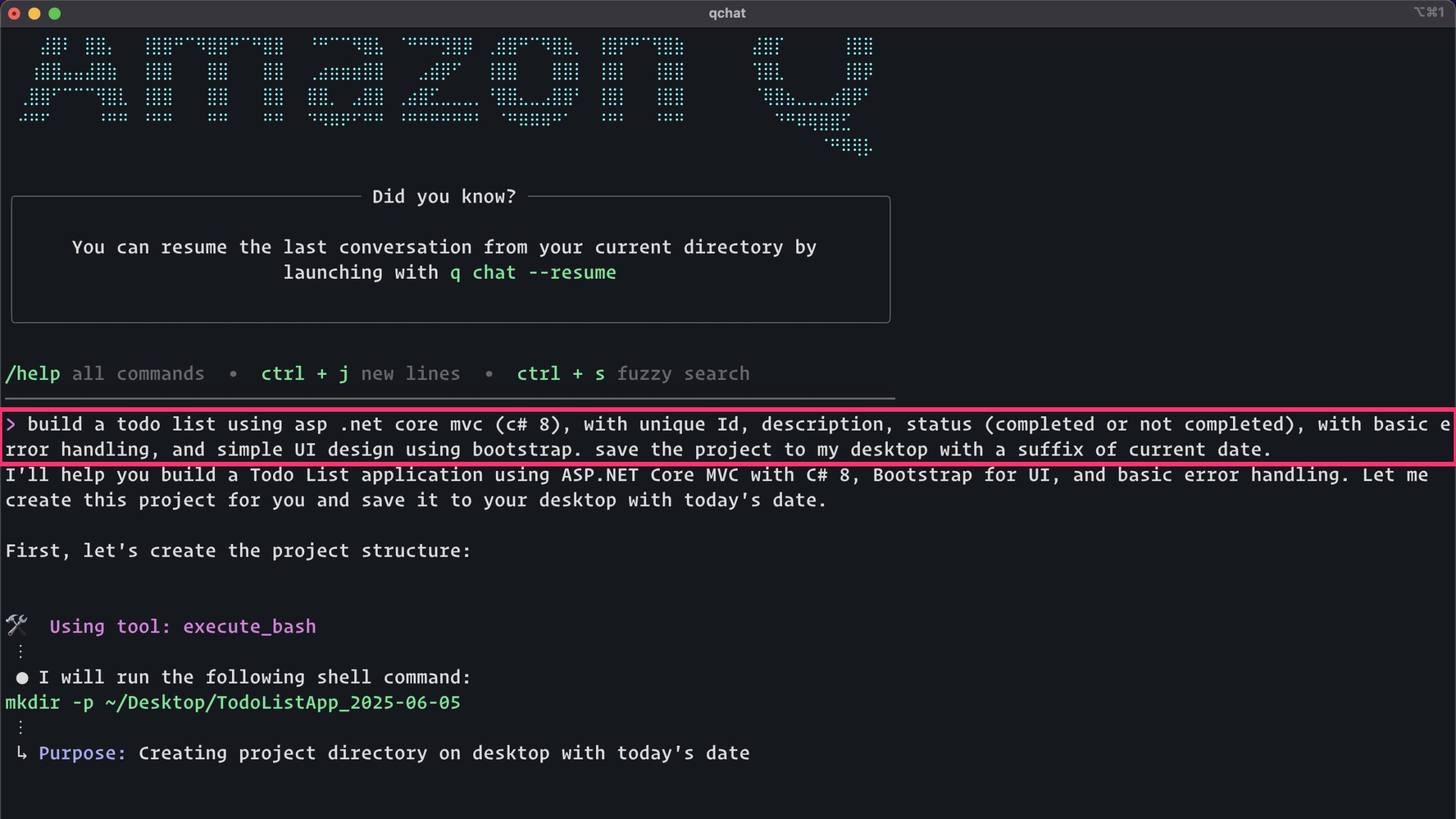
Body Cookies Headers (5) Test Results

Status: 201 CREATED Time: 5 ms Size: 292 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {  
2   "description": "check in code to repo",  
3   "id": "d84309ef-8d00-4a1d-940a-8e987ac93b0f",  
4   "status": "not completed"  
5 }
```

Postbot Runner Start Proxy Cookies Vault Trash





Restored /Users/mba/Desktop/TodoListApp_2025-06-05/TodoListApp/TodoListApp.csproj (in 32 ms).
Restore succeeded.

● Completed in 3.855s

Now, let's create the Todo model, controller, and views. I'll start by creating the Todo model:

🛠 Using tool: fs_write

● Path: /Users/mba/Desktop/TodoListApp_2025-06-05/TodoListApp/Models/Todo.cs

```
+ 1: using System.ComponentModel.DataAnnotations;
+ 2:
+ 3: namespace TodoListApp.Models
+ 4: {
+ 5:     public class Todo
+ 6:     {
+ 7:         public int Id { get; set; }
+ 8:
+ 9:         [Required(ErrorMessage = "Description is required")]
+10:        [StringLength(100, ErrorMessage = "Description cannot be longer than 100 characters")]
+11:        public string Description { get; set; } = string.Empty;
+12:
+13:        public bool IsCompleted { get; set; }
+14:    }
```



● Completed in 0.1s

I've successfully created a Todo List application using ASP.NET Core MVC with C# 8 and saved it to your desktop with today's date (2025-06-05). Here's what I've built:

Features of the Todo List App

- **Todo Model** with:
 - Unique Id (auto-generated)
 - Description (required, with validation)
 - Status (IsCompleted boolean flag)
- **CRUD Operations**:
 - Create new todo items
 - View all todo items
 - Edit existing todo items
 - Delete todo items
 - Toggle completion status
- **Error Handling**:
 - Try-catch blocks in all controller actions
 - Validation error messages
 - User-friendly error notifications
- **UI Design**:
 - Bootstrap 5 for responsive layout
 - Bootstrap Icons for visual elements
 - Card-based design for todo items
 - Color-coded status indicators
 - Confirmation dialogs for delete operations

Project Structure

- **Models**: Todo.cs with validation attributes
- **Data**: TodoContext.cs for Entity Framework Core



- User-friendly error notifications
- **UI Design**:
 - Bootstrap 5 for responsive layout
 - Bootstrap Icons for visual elements
 - Card-based design for todo items
 - Color-coded status indicators
 - Confirmation dialogs for delete operations

Project Structure

- **Models**: Todo.cs with validation attributes
- **Data**: TodoContext.cs for Entity Framework Core
- **Controllers**: TodosController with CRUD operations and error handling
- **Views**: Index, Create, and Edit views with Bootstrap styling
- **Database**: SQLite database for data persistence

How to Run the Application

1. Navigate to the project directory:

```
bash  
cd ~/Desktop/TodoListApp_2025-06-05/TodoListApp
```

2. Run the application:

```
bash  
dotnet run
```

3. Open your browser and go to <https://localhost:5001> or <http://localhost:5000>

The application is ready to use! You can create, edit, delete, and toggle the status of your todo items with a clean, responsive

Todo List

+ Add New Todo



study verilog tutorial



Edit



Delete



go pick up kids from painting class at 7pm



Edit



Delete



buy ice cream from Costee



Edit



Delete

Total: 3 | Completed: 1 | Pending: 2

Customize code recommendations

The screenshot shows a Java code editor with a file named `FoodDeliveryFulfillment.java`. The code is annotated with several labels:

- Natural language prompt**: A callout pointing to a specific line of code.
- Generated in-line code based on your company's internal code bases**: A callout pointing to another part of the code.

```
1 // Process a list of unassigned food deliveries around the driver's current location
2
3 package anycompany.fooddelivery.fulfillment;
4 import anycompany.delivery.Delivery;
5 import anycompany.delivery.DeliveryService;
6 import anycompany.driver.Driver;
7 import anycompany.driver.DriverLocationService;
8
9 public class FoodDeliveryFulfillment {
10     private DeliveryService deliveryService;
11     private DriverLocationService driverLocationService;
12
13     //Process all the unassigned deliveries
14     public void processUnassignedDeliveries() {
15         List<Delivery> unassignedDeliveries = deliveryService.getUnassignedDeliveries();
16         //Iterate over all the unassigned deliveries and assign them to a driver
17         for (Delivery delivery : unassignedDeliveries) {
18             //Get the nearest drivers for delivery location
19             List<Driver> drivers = driverLocationService.getDriver(delivery.getLocation());
20             for (Driver driver : drivers) {
21                 //Assign delivery to driver and send notification
22                 boolean isAssigned = deliveryService.assignDeliveryToDriver(delivery, driver);
23                 if (isAssigned) {
24                     deliveryService.notifyDelivery(delivery);
25                     driverLocationService.notifyDriver(driver);
26                     break;
27                 }
28             }
29         }
30     }
31
32     ...
33 }
34 }
```

Generate code based on your private repositories

Use Amazon Q Developer chat in the IDE to ask questions about your internal code base

Help understand your internal code base

Customized code suggestions with Amazon Q



National Australia Bank

“

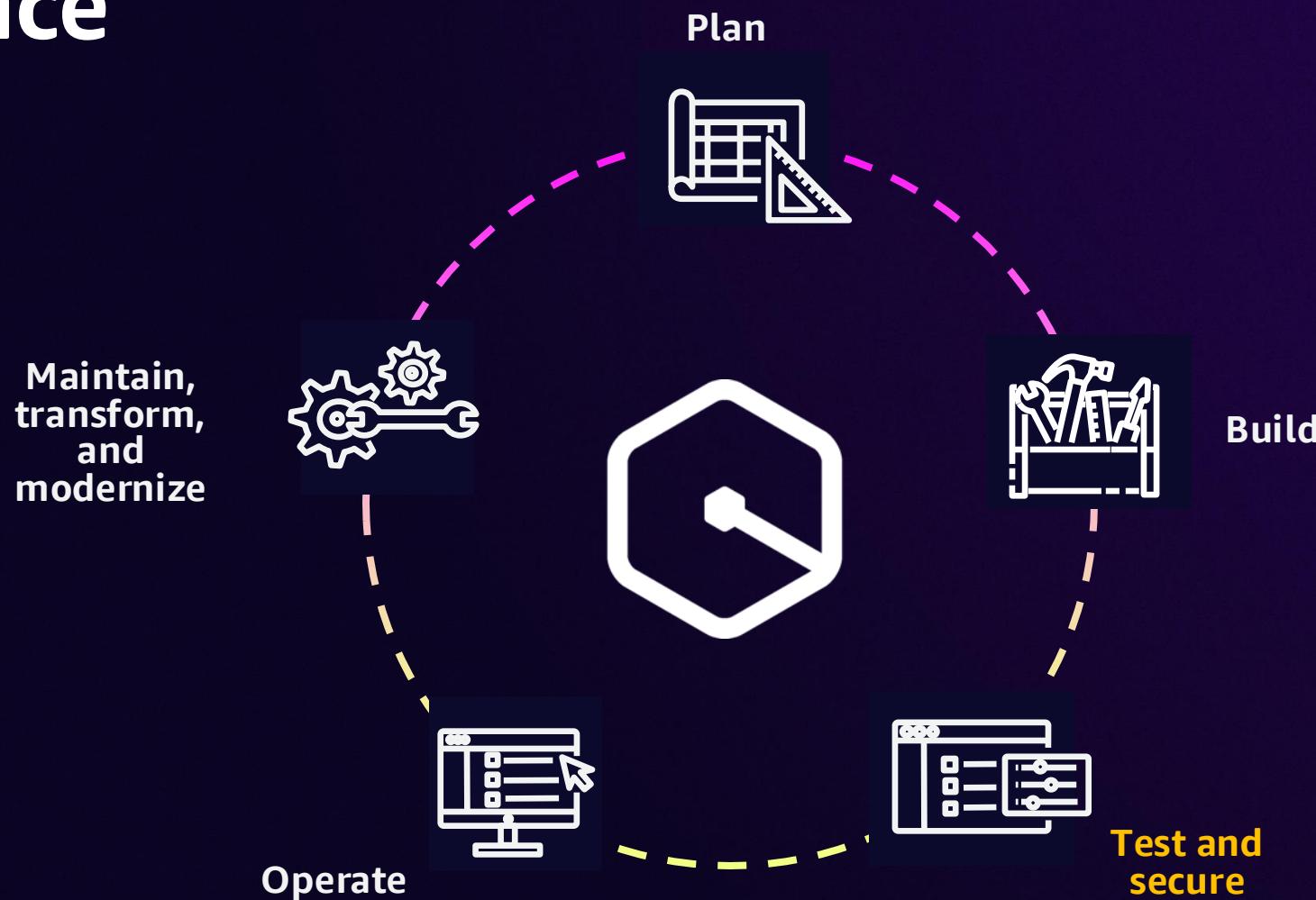
At NAB, our developers use Amazon Q Developer's generative AI-powered assistant to generate real-time coding suggestions. NAB has now added specific suggestions using the Amazon Q customisation capability that are tailored to the NAB coding standards. **We have 500 developers using Amazon Q Developer with customisation enabled. We're seeing increased acceptance rates of 60% with customisation, and we expect this to grow with Amazon Q Developer chat, which makes it even easier to interact using natural language.** ”

Andrew Brydon

Executive for Enterprise Engineering, National Australia Bank



Amazon Q Developer transforms the development experience



Unit test generation (/test)

Amazon Q Developer agent for unit tests

☰ /test

Included in your Q Developer Agent subscription

I can generate unit tests for your active file.

After you select the functions or methods I should focus on, I will:

1. Generate unit tests
2. Place them into relevant test file

To learn more, check out our [user guide](#).



← →

demo-feat-2048-game

□ □ □ 0%



AMAZON Q

...

□ ...

> CODE ISSUES

CHAT

Chat X Q - Test X +

To learn more, check out our User Guide.

Sure. This may take a few minutes. I'll share updates here as I work on this.

Generating unit tests for the following methods in app.py

new_game

make_move

get_game_state



Generating unit tests... 71%

Cancel

Amazon Q Developer uses generative AI. You may need to verify responses. See the [AWS Responsible AI Policy](#). Amazon Q Developer

app.py game.py

```
src > app > app.py
46     async def new_game() -> dict:
47         """Start a new game and return the initial state."""
48         game = Game2048()
49         game_id = str(len(games))
50         games[game_id] = game
51         return {"game_id": game_id, "state": game.get_state()}
52
53 @app.post("/game/move")
54 async def make_move(move_request: MoveRequest) -> dict:
55     """Make a move in the specified direction."""
56     game_id = move_request.game_id
57     if game_id not in games:
58         raise HTTPException(status_code=404, detail="Game not found")
59
60     game = games[game_id]
61     try:
62         moved = game.move(move_request.direction)
63         return {"moved": moved, "state": game.get_state()}
64     except ValueError as e:
65         raise HTTPException(status_code=400, detail=str(e))
66
67 @app.get("/game/{game_id}")
68 async def get_game_state(game_id: str) -> dict:
69     """Get the current state of a game."""
70     if game_id not in games:
71         raise HTTPException(status_code=404, detail="Game not found")
72     return games[game_id].get_state()
```



app.py

test_app.py (Generated by Amazon Q) X

game.py

tmp > aws-toolkit-vscode > q-testgen > resultArtifacts > tests > test_app.py

```
40+
41+def test_get_game_state_empty_game_id(test_client):
42+    """
43+        Test get_game_state with an empty game ID.
44+
45+        response = test_client.get("/game/")
46+        assert response.status_code == 404
47+        assert "detail" in response.json()
48+
49+
50+def test_get_game_state_existing_game():
51+    """
52+        Test get_game_state when the game_id exists in games.
53+        It should return the game state.    }
54+
55+        game_id = "test_game_id"
56+        mock_game = MagicMock(spec=Game2048)
57+        mock_game.get_state.return_value = {
58+            "board": [[0, 2, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]],
59+            "score": 0,
60+            "game_over": False
61+        }
62+
63+        games[game_id] = mock_game
64+
65+        with TestClient(app) as client:
66+            response = client.get(f"/game/{game_id}")
67+
68+        assert response.status_code == 200
69+        assert response.json() == mock_game.get_state.return_value
70+
71+        # Clean up the test game
72+        del games[game_id]
```

Automated code reviews (/review)

Amazon Q Developer agent for code reviews



/review

Included in your Q Developer subscription

I can review your workspace for vulnerabilities and issues.

After you begin a review, I will:

1. Review all relevant code in your workspace or your current file
2. Provide a list of issues for your review

You can then investigate, fix, or ignore issues.

To learn more, check out our [User Guide](#).

The screenshot shows the Amazon Q Developer interface integrated with a code editor. The top navigation bar includes tabs for 'AMAZON Q', 'CODE ISSUES', and 'CHAT'. The 'CHAT' tab is active, showing a 'Q - Review' panel. This panel displays a message about the /review command, stating it's included in the Q Developer subscription and can review the workspace for vulnerabilities and issues. It lists steps for the review process and provides links to the User Guide and workspace/active file reviews.

The main content area shows a code editor with a FastAPI application. The code defines a FastAPI app, adds middleware for CORS, and handles game creation and moves. A specific line of code for handling moves is highlighted with a tooltip asking if the user wants to make a move request.

```
src > app > app.py > ...
28
29     app = FastAPI(title="2048 Game API", debug=stage == "development")
30
31     app.add_middleware(
32         CORSMiddleware,
33         allow_origins=origins,
34         allow_credentials=True,
35         allow_methods=["*"],
36         allow_headers=["*"],
37     )
38     # Store active games
39     games = {}
40
41     class MoveRequest(BaseModel):
42         direction: str
43         game_id: str
44
45     @app.post("/game/new")
46     async def new_game() -> dict:
47         """Start a new game and return the initial state."""
48         game = Game2048()
49         game_id = str(len(games))
50         games[game_id] = game
51         return {"game_id": game_id, "state": game.get_state()}
52
53     @app.post("/game/move")
54     async def make_move(move_request: MoveRequest):
55         """Make a move (parameter) move_request: MoveRequest"""
56         game_id = move_request.game_id
57         if game_id not in games:
58             raise HTTPException(status_code=404, detail="Game not found")
59
60         game = games[game_id]
61         try:
62             moved = game.move(move_request.direction)
63             return {"moved": moved, "state": game.get_state()}
64         except ValueError as e:
65             raise HTTPException(status_code=400, detail=str(e))
66
67     @app.get("/game/{game_id}")
68
```

The screenshot shows the Amazon Q developer interface integrated with a code editor. The top navigation bar includes tabs for 'demo' and 'AMAZON Q'. The left sidebar features icons for file, search, and code review, along with sections for 'CODE ISSUES', 'CHAT', and 'Ask a question or enter "/" for quick actions'. The 'CODE ISSUES' section lists critical, high, medium, low, and info-level issues found in the code. The 'CHAT' section provides a summary of the issues found. The main area displays a Python file named 'app.py' with code related to a FastAPI application for a 2048 Game API. The code handles creating new games and making moves.

```
src > app > app.py > new_game
27
28
29     app = FastAPI(title="2048 Game API", debug=stage == "development")
30
31     app.add_middleware(
32         CORSMiddleware,
33         allow_origins=origins,
34         allow_credentials=True,
35         allow_methods=["*"],
36         allow_headers=["*"],
37     )
38     # Store active games
39     games = {}
40
41     class MoveRequest(BaseModel):
42         direction: str
43         game_id: str
44
45     @app.post("/game/new")
46     async def new_game() -> dict:
47         """Start a new game and return the initial state."""
48         game = Game2048()
49         game_id = str(len(games))
50         games[game_id] = game
51         return {"game_id": game_id, "state": game.get_state()}
52
53     @app.post("/game/move")
54     async def make_move(move_request: MoveRequest) -> dict:
55         """Make a move in the specified direction."""
56         game_id = move_request.game_id
57         if game_id not in games:
58             raise HTTPException(status_code=404, detail="Game not found")
59
60         game = games[game_id]
61         try:
62             moved = game.move(move_request.direction)
63             return {"moved": moved, "state": game.get_state()}
64         except ValueError as e:
65             raise HTTPException(status_code=400, detail=str(e))
```

The screenshot shows a code editor interface with several tabs at the top: `app.py`, `test_app.py`, and `game.py`. The `app.py` tab is active, displaying Python code for a FastAPI application. The code includes a `CORSMiddleware` configuration and two API endpoints: `/game/new` and `/game/move`.

In the bottom right corner of the editor, there is a "Code Issue Details" panel. It displays the following information:

CWE-1333 - Package Vulnerability High

The `py` library through 1.11.0 for Python allows remote attackers to conduct a ReDoS (Regular expression Denial of Service) attack via a Subversion repository with crafted info data, because the `InfoSvn` argument is mishandled. Note: This has been disputed by multiple third parties as not being reproduceable and they argue this is not a valid vulnerability.

Relevant link: <https://github.com/advisories/GHSA-w596-4wvx-j9j6>

Common Weakness	Detector library
Enumeration (CWE) CWE-1333	-

File path
poetry.lock [Ln 8]

Actions: [Generate Fix](#) [Explain](#) [Ignore](#) [Ignore All](#)

```
27
28 app = FastAPI(title="2048 Game API", debug=stage == "development")
29
30 app.add_middleware(
31     CORSMiddleware,
32     allow_origins=origins,
33     allow_credentials=True,
34     allow_methods=["*"],
35     allow_headers=["*"],
36 )
37
38 # Store active games
39 games = {}
40
41 class MoveRequest(BaseModel):
42     direction: str
43     game_id: str
44
45 @app.post("/game/new")
46 async def new_game() -> dict:
47     """Start a new game and return the initial state."""
48     game = Game2048()
49     game_id = str(len(games))
50     games[game_id] = game
51     return {"game_id": game_id, "state": game.get_state()}
52
53 @app.post("/game/move")
54 async def make_move(move_request: MoveRequest) -> dict:
55     """Make a move in the specified direction."""
56     game_id = move_request.game_id
57     if game_id not in games:
58         raise HTTPException(status_code=404, detail="Game not found")
59
60     game = games[game_id]
61     try:
62         moved = game.move(move_request.direction)
63         return {"moved": moved, "state": game.get_state()}
64     except ValueError as e:
65         raise HTTPException(status_code=400, detail=str(e))
```

demo

Code Issue Details

```
app.py 5 poetry.lock: Original ↔ poetry.lock test_app.py M ↑ ↓ ¶ ⌂ ...
```

1 # This file is automatically @generated by Poetry 1.8.3 and should not be change
2
3 [[package]]
4 name = "annotated-types"
5 version = "0.7.0"
6 description = "Reusable constraint types to use with typing.Annotated"
7 optional = false
8 python-versions = ">=3.8"
8+ python-versions = ">=3.9"
9 files = [
10 {file = "annotated_types-0.7.0-py3-none-any.whl", hash = "sha256:1f02e8b43a8"},
11 {file = "annotated_types-0.7.0.tar.gz", hash = "sha256:aff07c09a53a08bc8cfcc"}]
12]
13
14 [[package]]
15 name = "anyio"
16 version = "4.4.0"
17 description = "High level compatibility layer for multiple asynchronous event loops"
18 optional = false
19 python-versions = ">=3.8"
20 files = [
21 {file = "anyio-4.4.0-py3-none-any.whl", hash = "sha256:c1b2d8f46a8a812513012"},
22 {file = "anyio-4.4.0.tar.gz", hash = "sha256:5aadc6a1bbb7cdb0bede386cac5e294"}]
23]
24
25 [package.dependencies]
26 idna = ">=2.8"
27 sniffio = ">=1.1"
28
29 [package.extras]
30 doc = ["Sphinx (>=7)", "packaging", "sphinx-autodoc-typehints (>=1.2.0)", "sphinxcontrib-apidoc (>=0.1.12)"]
31 test = ["anyio[trio]", "coverage[toml] (>=7)", "exceptiongroup (>=1.2.0)", "hyperset (>=0.1.12)"]
32 trio = ["trio (>=0.23)"]
33
34 [[package]]
35 name = "attrs"
36 version = "24.2.0"
37 description = "Classes Without Boilerplate"
38 optional = false
39 python-versions = ">=3.7"
40 files = [

CWE-1333 - Package Vulnerability High

The py library through 1.11.0 for Python allows remote attackers to conduct a ReDoS (Regular expression Denial of Service) attack via a Subversion repository with crafted info data, because the InfoSvnCommand argument is mishandled. Note: This has been disputed by multiple third parties as not being reproducible and they argue this is not a valid vulnerability.

Relevant link: <https://github.com/advisories/GHSA-w596-4wvx-j9j6>

Common Weakness Enumeration (CWE)	Detector library
CWE-1333	-

File path
[poetry.lock \[Ln 8\]](#)

Suggested code fix preview

```
version = "0.7.0"  
description = "Reusable constraint types to use with typing.Annotated"  
optional = false  
python-versions = ">=3.9"  
files = [  
    {file = "annotated_types-0.7.0-py3-none-any.whl", hash = "sha256:1f02e8b43a8"},  
    {file = "annotated_types-0.7.0.tar.gz", hash = "sha256:aff07c09a53a08bc8cfcc"}]
```

Open diff Copy

Accept Fix Regenerate Fix Explain Ignore Ignore All

Code review spans across various types of code issues

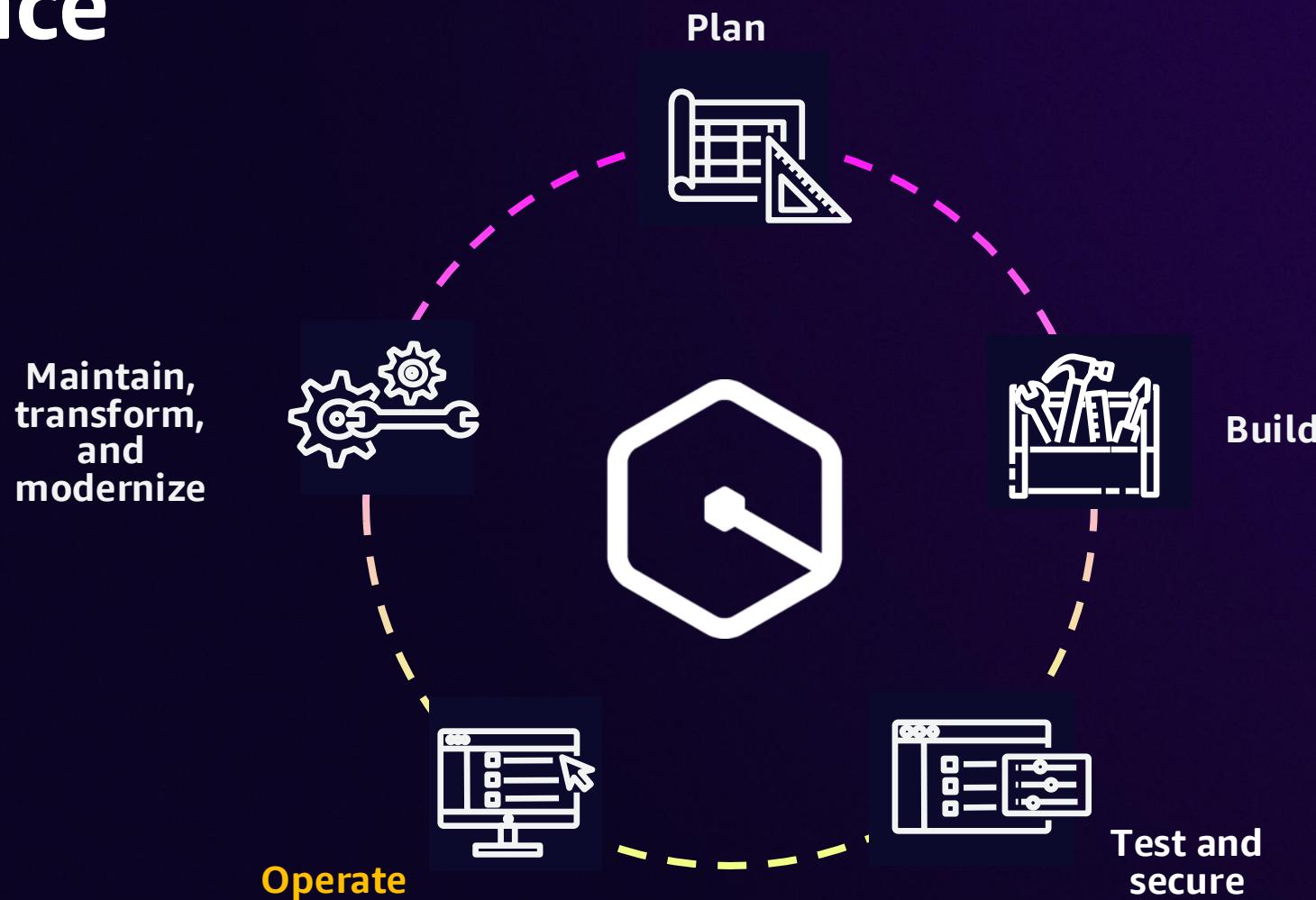
Amazon Q reviews your code for the following types of code issues:

- **SAST scanning — Detect security vulnerabilities in your source code.** Amazon Q identifies various security issues, such as resource leaks, SQL injection, and cross-site scripting.
- **Secrets detection — Prevent the exposure of sensitive or confidential information in your code.** Amazon Q reviews your code and text files for secrets such as hardcoded passwords, database connection strings, and usernames. Secrets findings include information about the unprotected secret and how to protect it.
- **IaC issues — Evaluate the security posture of your infrastructure files.** Amazon Q can review your infrastructure as code (IaC) code files to detect misconfiguration, compliance, and security issues.
- **Code quality issues — Ensure your code is meeting quality, maintainability, and efficiency standards.** Amazon Q generates code issues related to various quality issues, including but not limited to performance, machine learning rules, and AWS best practices.
- **Code deployment risks — Assess risks related to deploying code.** Amazon Q determines if there are any risks to deploying or releasing your code, including application performance and disruption to operations.
- **Software composition analysis (SCA) — Evaluate third-party code.** Amazon Q examines third-party components, libraries, frameworks, and dependencies integrated into your code, ensuring third-party code is secure and up to date.

<https://docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/code-reviews.html#issue-types>



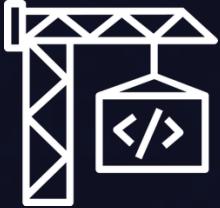
Amazon Q Developer transforms the development experience



Cloud Operations use cases for Amazon Q Developer

Build

(in the AWS Management Console)



Get expert guidance for building and deploying on AWS

- Ask for expert advice
- Receive actionable recommendations (e.g., Amazon SageMaker)
- Identify and resolve common errors in the console
- Generate Infrastructure as Code (IaC)

Operate



Improve operational efficiency

- Discover AWS resources
- Investigate and remediate operational issues
- Assisted incident investigation
- Third-party integrations (Wiz, Datadog)

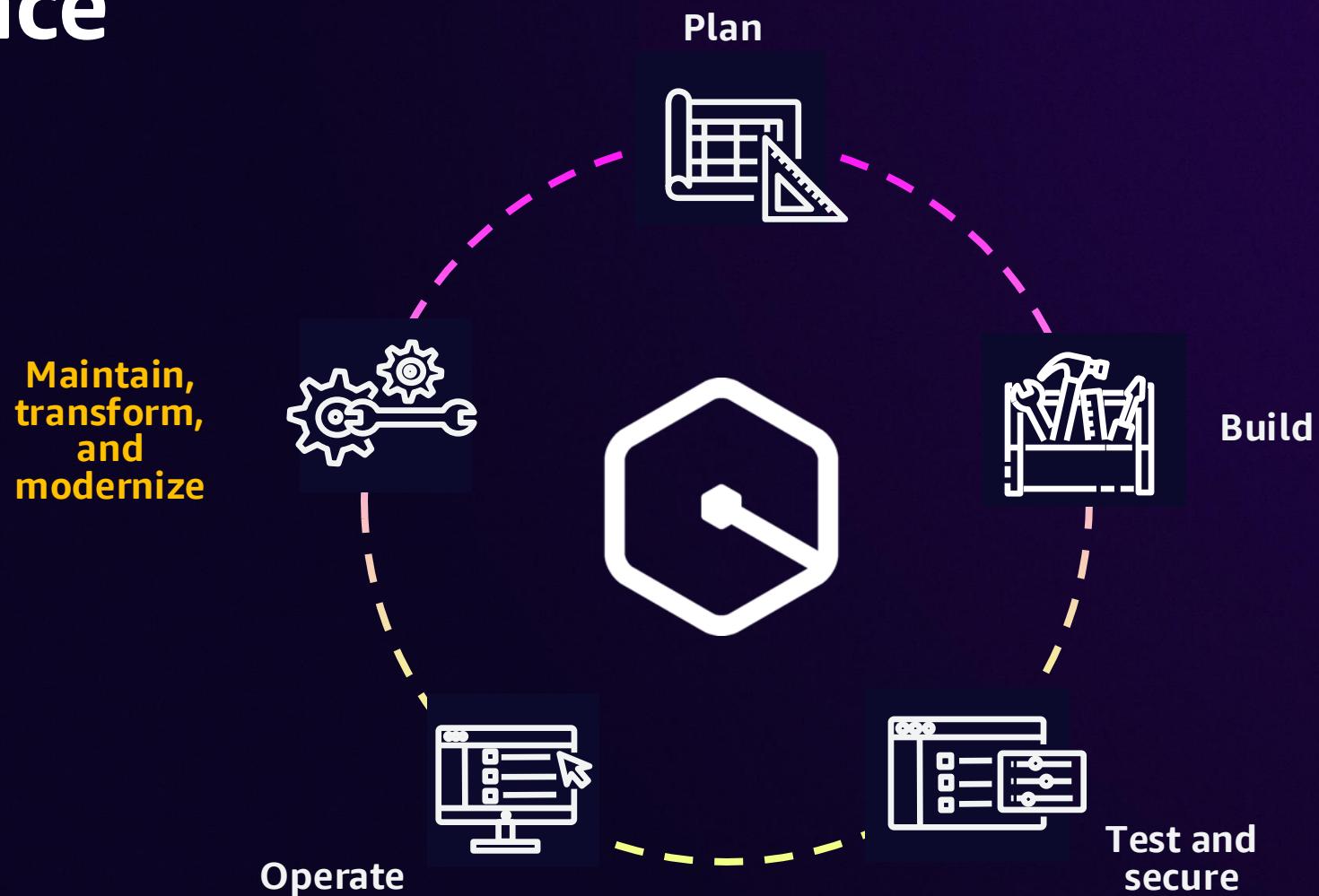
Optimize



Understand and optimize cost

- Investigate your AWS bill
- Identify trends and optimization opportunities
- Get EC2 Instance sizing advice

Amazon Q Developer transforms the development experience





Canada Life - Transforming API security with Amazon Q Developer | Amazon Web Services

481 views • 3 months ago



Amazon Web Services ✓

Canada Life is an insurance and financial services company. To address existing security vulnerabilities, the company decided to ...

CC

https://www.youtube.com/watch?v=_Cp7Jno9AQc

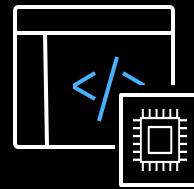
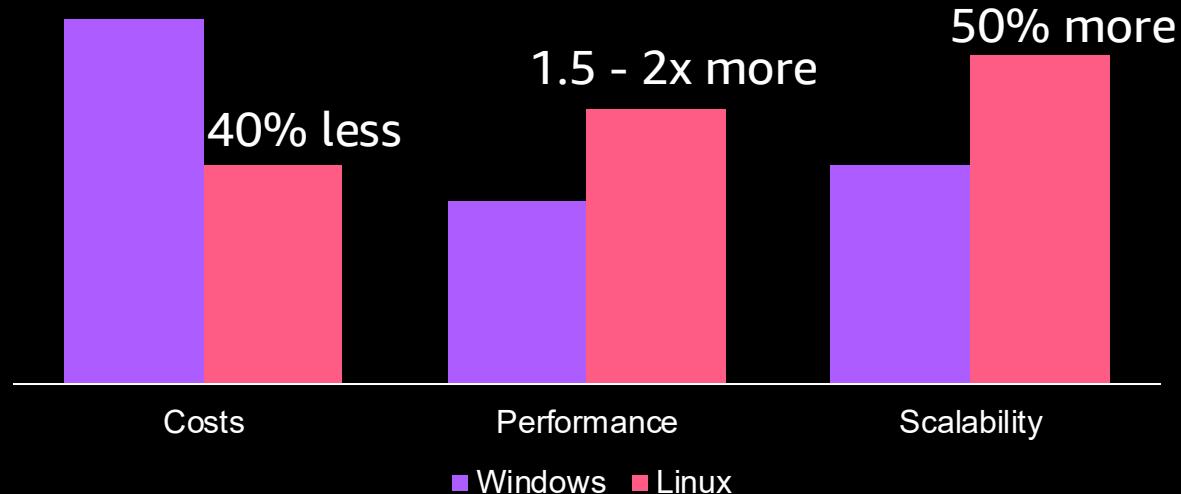
AWS Transform for .NET

The first agentic AI service for
modernizing .NET applications at scale

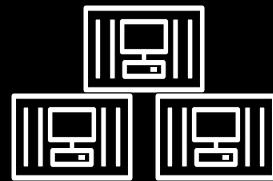


Why modernize .NET apps from Windows to Linux?

Customers have reduced their Windows usage by over 70% to adopt Linux



Compatible with x86-64
and ARM64



Lightweight containers



Serverless architecture
with Lambda



Modernizing can cut compute costs substantially

Examples of EC2 hourly pricing for the same instance type/size

<u>Instance</u>	<u>Windows</u>	<u>Linux</u>	<u>Savings vs. Windows</u>
m7i.2xlarge	\$0.7712	\$0.4032	47.7% savings
c7i.4xlarge	\$1.450	\$0.714	50.8% savings
r7i.8xlarge	\$3.5888	\$2.1168	41.0% savings

Additional savings by moving to Graviton instances

<u>Instance</u>		<u>Linux</u>	<u>vs. Intel+Windows</u>	<u>vs. Intel+Linux</u>
m7g.2xlarge	—	\$0.3264	57.7% savings	19% savings
c7g.4xlarge	—	\$0.5781	60.1% savings	19% savings
r8g.8xlarge	—	\$1.88512	47.5% savings	11% savings

*Ohio region, July 2024



INTRODUCING AWS Transform for .NET

The first agentic AI-powered experience for modernizing .NET applications at scale

GENERALLY AVAILABLE

Multifunctional team collaboration

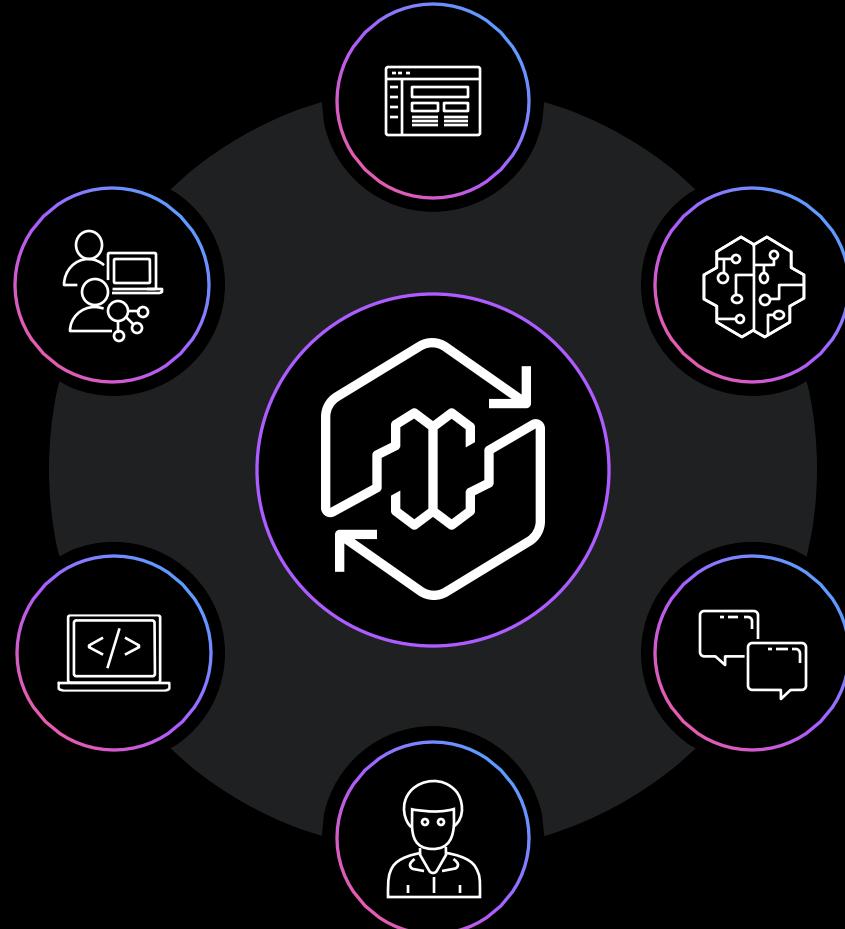
IDE experience for developers

A unified web experience

Foundational LLMs

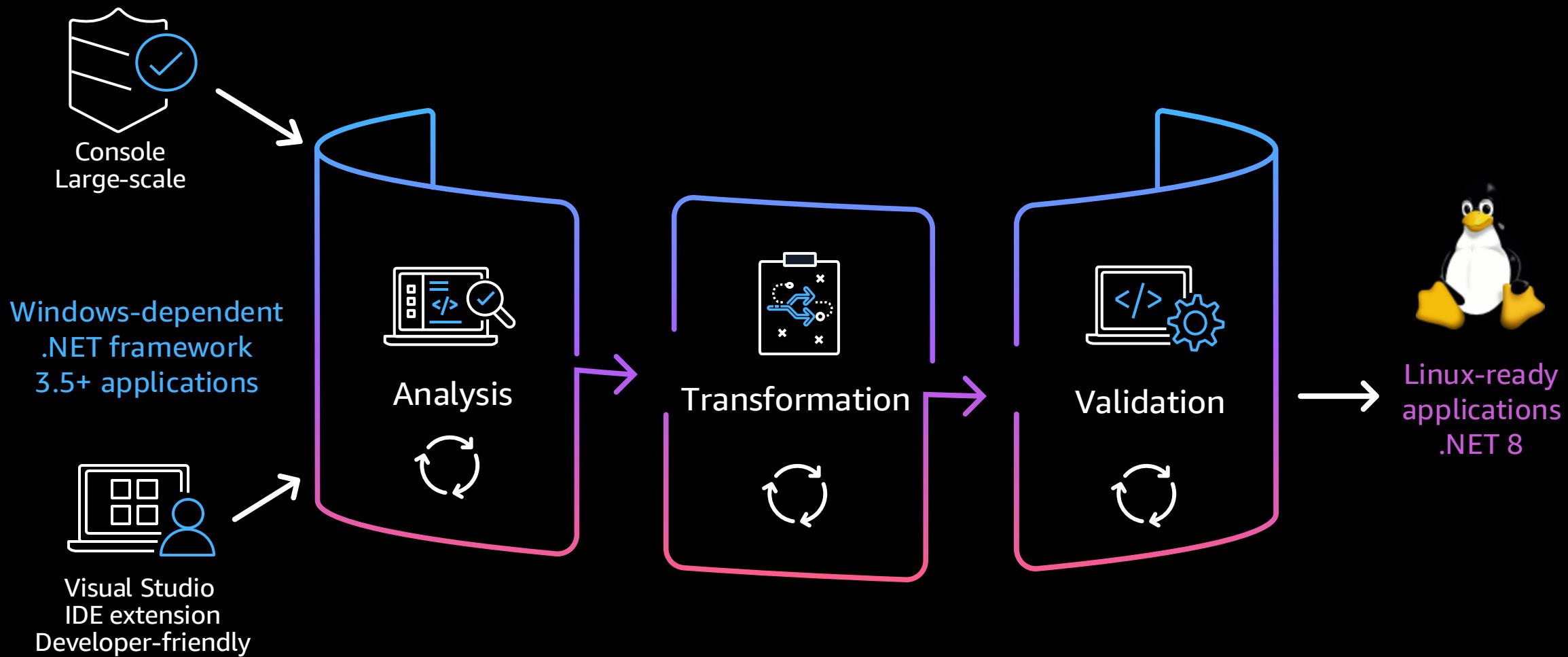
.NET domain-expert AI agent

Human in the Loop



.NET Transformation agent

Agent for .NET application porting to Linux



What's supported in upgrading .NET code in AWS Transform?

Supported .NET project types:

- Libraries
- Applications written in C#
- Console applications
- Web API (ASP.NET) Web API
- Business Logic Layers of SPA (Single Page Application) backends
- Model View Controller (MVC) applications including front-end Razor views
- Windows Communication Foundation (WCF) services
- Unit test projects (NUnit, xUnit, and MSTest)
- Projects with provided cross-platform versions for third-party or private NuGet packages. If a cross-platform equivalent is missing or unavailable, AWS Transform .NET will attempt a best-effort conversion.

AWS Transform does not transform the following:

- WebForms (.aspx), WinForms, Blazor UI components
- Win32 DLLs that don't have core compatible libraries
- Applications written in F# or VB.NET
- Applications already in .NET 8.0+
- AWS Transform will not modify the original repo branches, and can only write to a separate target branch specified in your transformation plan.

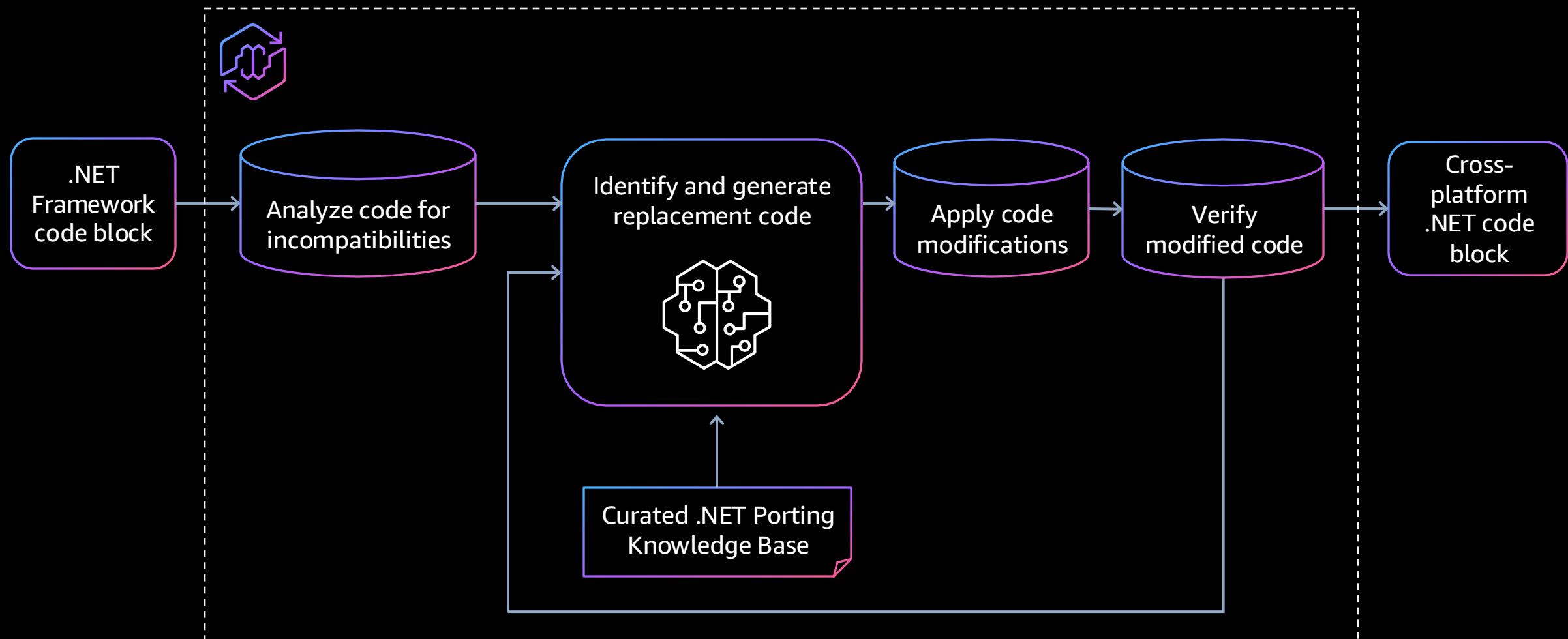
<https://docs.aws.amazon.com/transform/latest/userguide/dotnet.html>



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How .NET Transformation works?

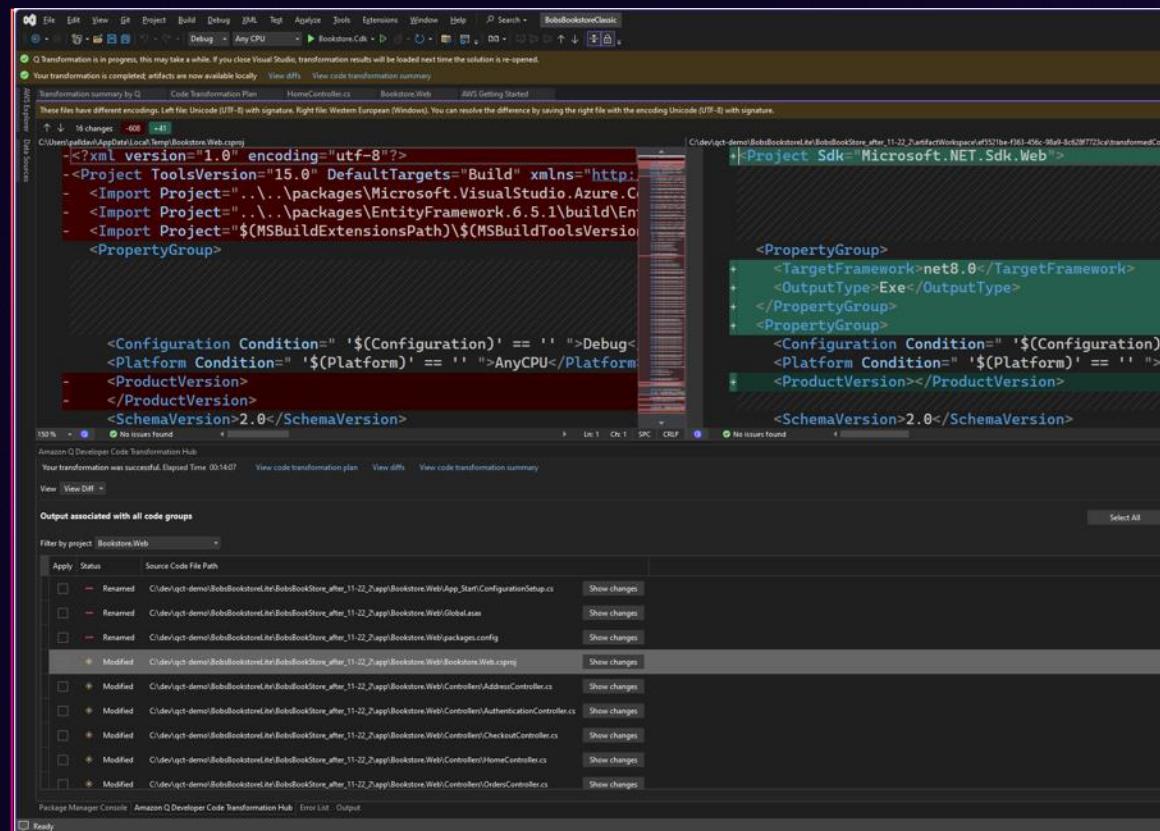
Analyze, transform, and validate



Two experiences for .NET porting

THE SAME GENERATIVE AI AGENT CAPABILITIES

.NET porting in Visual Studio IDE (for developers)

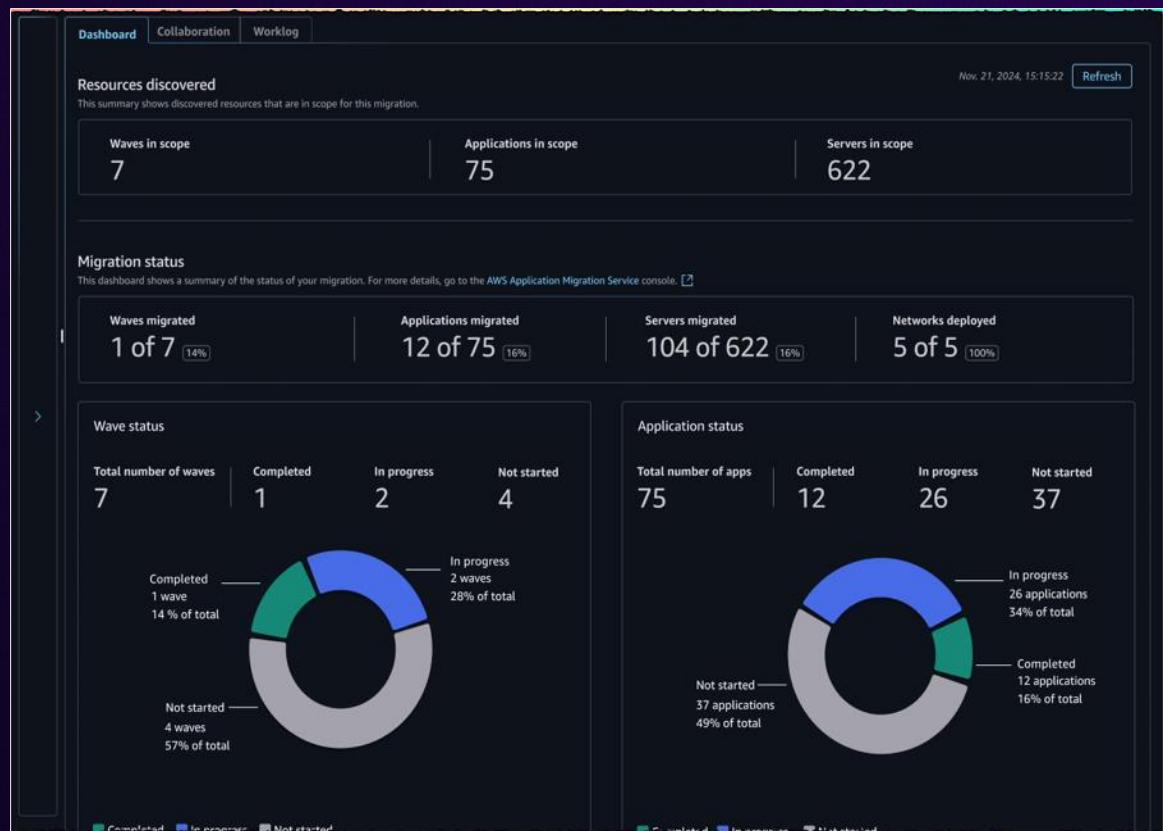


Amazon Q Developer



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

.NET porting at large-scale using a web experience (for IT and modernization teams)



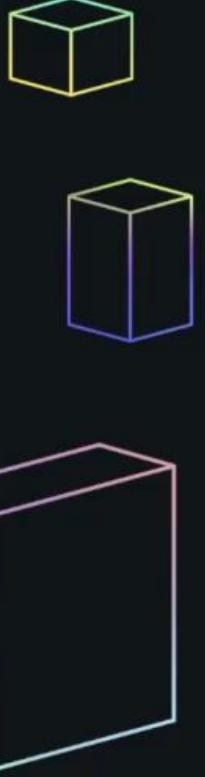
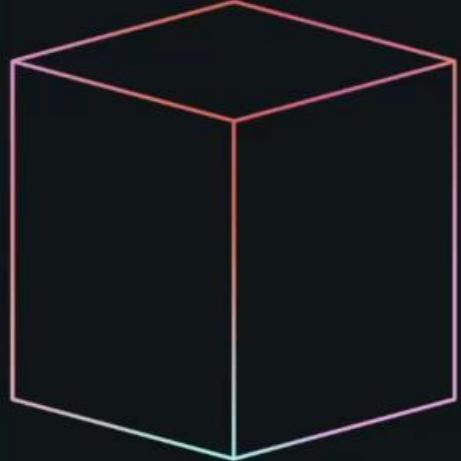
AWS Transform

Console Experience

DotNetDemo-workspace [Edit](#)

This is your workspace description. Workspace is a collection of jobs, artifacts, connectors and collaborators. You can chat to Q about any of these concepts.

[Jobs](#) [Artifacts](#)



Create your first transformation job with AWS Transform

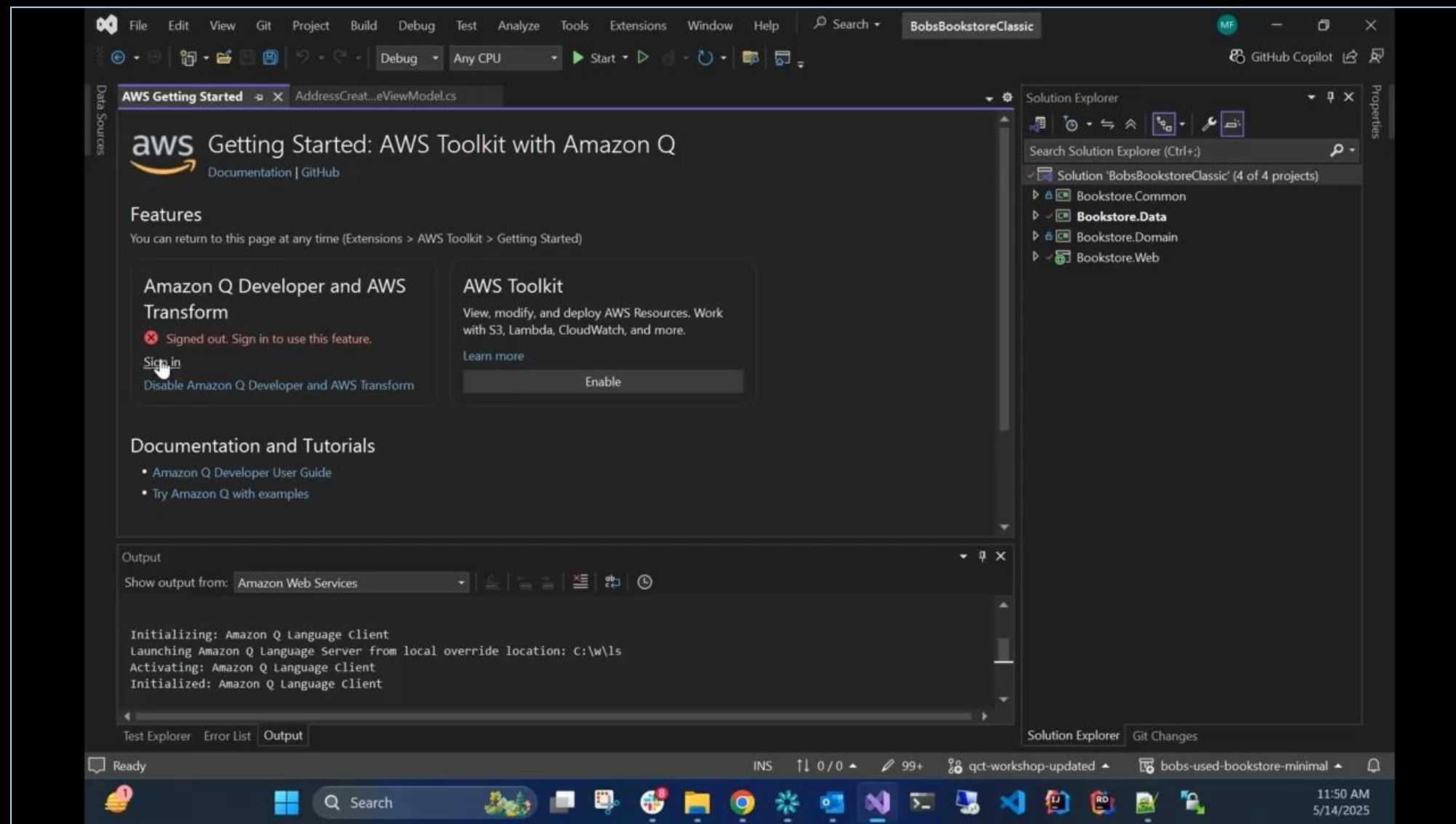
Within a workspace, you and the collaborators you invite can create and perform transformation jobs. When creating a job, AWS Transform will work with you to define an objective for the job. Based on your objective, AWS Transform will:

1. Generate a plan
2. Start the job
3. Ask for resources or clarification when needed

[Create a job in AWS Transform](#)

© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

IDE Experience



Thomson Reuters turbocharged .NET modernization with AWS Transform

CHALLENGES

Thomson Reuters was struggling with **expensive**, time-consuming **maintenance** of legacy .NET Framework applications that were impeding innovation and competing with their roadmap priorities.

SOLUTIONS

The company implemented AWS Transform, an agentic AI-powered tool designed for **large-scale** .NET modernization that enables parallel processing through both **web interface** and **IDE integration**.

RESULTS

- › Modernized 4x faster: **1.5M lines of code/month**
- › **30% cost savings** moving to Linux
- › 70% reduction in technical debt
- › Transformation time: **2 weeks** vs. months

"AWS Transform felt like an extension of our team—constantly learning, optimizing, and helping us move faster."

Matt Dimich

VP, Platform Engineering Enablement at Thomson Reuters

CUSTOMER PROFILE



INDUSTRY
Financial Services

REGION
United States/Global

As a global tech and AI leader powering the legal, tax, and compliance industries, Thomson Reuters is always looking for smarter, faster ways to build. That means staying ahead of the curve—and teaming up with partners like AWS to push what's possible.



Q Developer Java upgrades



V8, v11, v17, v21



V17, v21



Build & test
Java 8,11,17,21



Apply Knowledge Base



Build & Test
Java 17,21



Fix Errors with GenAI



Continuous Maintenance

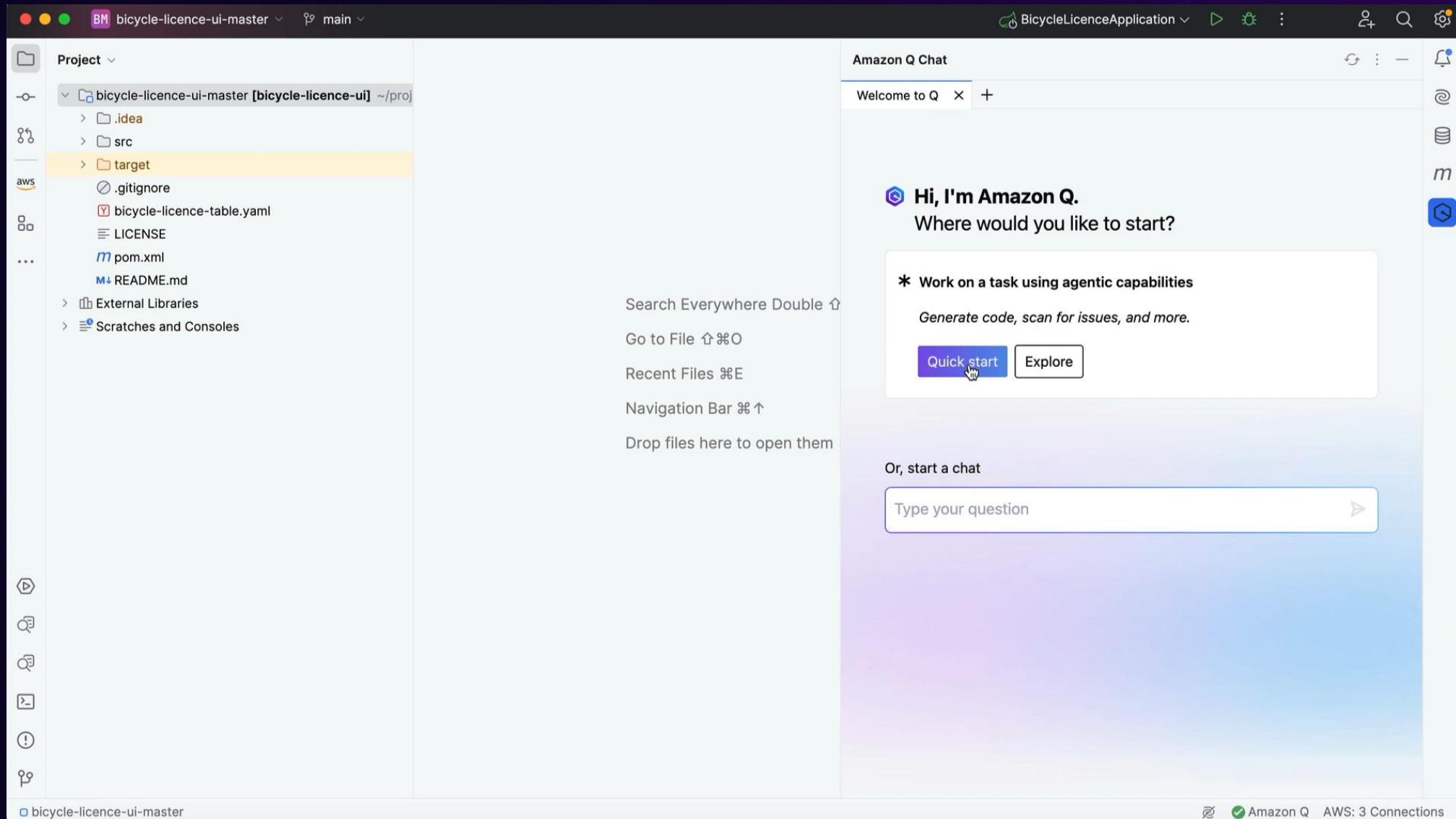
Bicycle licence application

The screenshot shows a web application interface. At the top left is the GOV.UK logo. Below it, a navigation bar contains the text "Bicycle Licence". The main title "Bicycle Licence" is displayed prominently in large, bold, black font. Below the title, a subtitle "A place to administer licences to ride a bicycle" is shown in a smaller, bold, black font. A welcome message "Welcome to the Bicycle Licence Application." follows. A bulleted list of actions includes "create a licence", "update a licence", and "view history of a licence". At the bottom left is a green button labeled "Start now >".



[Bicycle license application](#)
Java 11 application using
Spring Framework and
DynamoDB to demonstrate
code transformation agent

Run the transformation agent



Generate transformation plan

The screenshot shows a browser window with the following elements:

- Left Sidebar:** Project navigation for "bicycle-licence-ui-master" showing "main" branch. Items include .idea, src, target (selected), .gitignore, bicycle-licence-table.yaml, LICENSE, pom.xml, README.md, External Libraries, and Scratches and Consoles.
- Top Bar:** Project name "bicycle-licence-ui-master", branch "main", and a search bar.
- Header:** "Plan to Transform your project" tab.
- Amazon Q Chat:** A sidebar titled "Amazon Q Chat" with a message history:
 - "proposed changes."
 - "I was able to build your module and will start uploading your code."
 - "I'm starting to transform your code. It can take 10 to 30 minutes to upgrade your code, depending on the size of your module. To monitor progress, go to the Transformation Hub."
 - "If I run into any issues, I might pause the transformation to get input from you on how to proceed."
- Main Content Area:** "Code Transformation plan by Amazon Q".
 - Summary:** Amazon Q reviewed your code and generated a transformation plan. Amazon Q will suggest code changes according to the plan, and you can review the updated code before accepting changes to your files.
 - Statistics:** Lines of code in your application: 749, Dependencies to be replaced: 5, Deprecated code instances to be replaced: 0, Files to be changed: 5.
 - Planned transformation changes:** Amazon Q will use the proposed changes as guidance during the transformation. The final code updates might differ from this plan. [Read more](#).
 - Step 1 - Update JDK version, dependencies and related code:** Amazon Q will attempt to update the JDK version and change the following dependencies and related code.

Dependency	Action	Current version	Target version
jakarta.servlet:jakarta.servlet-api	ADD	-	6.x
javax.servlet;javax.servlet-api	ADD	-	-
javax.servlet;javax.servlet-api	REMOVE	-	-
org.junit.vintage:junit-vintage-engine	REMOVE	-	-
org.mockito:*	UPDATE	-	4.x
org.springframework.boot:spring-boot...	UPDATE	2.1.8.RELEASE	3.3.x
 - Step 2 - Upgrade deprecated code:** Amazon Q will attempt to replace the following instances of deprecated code.

Deprecated code	Files to be changed
- Bottom Navigation:** Find, pom.xml, and AWS connections.

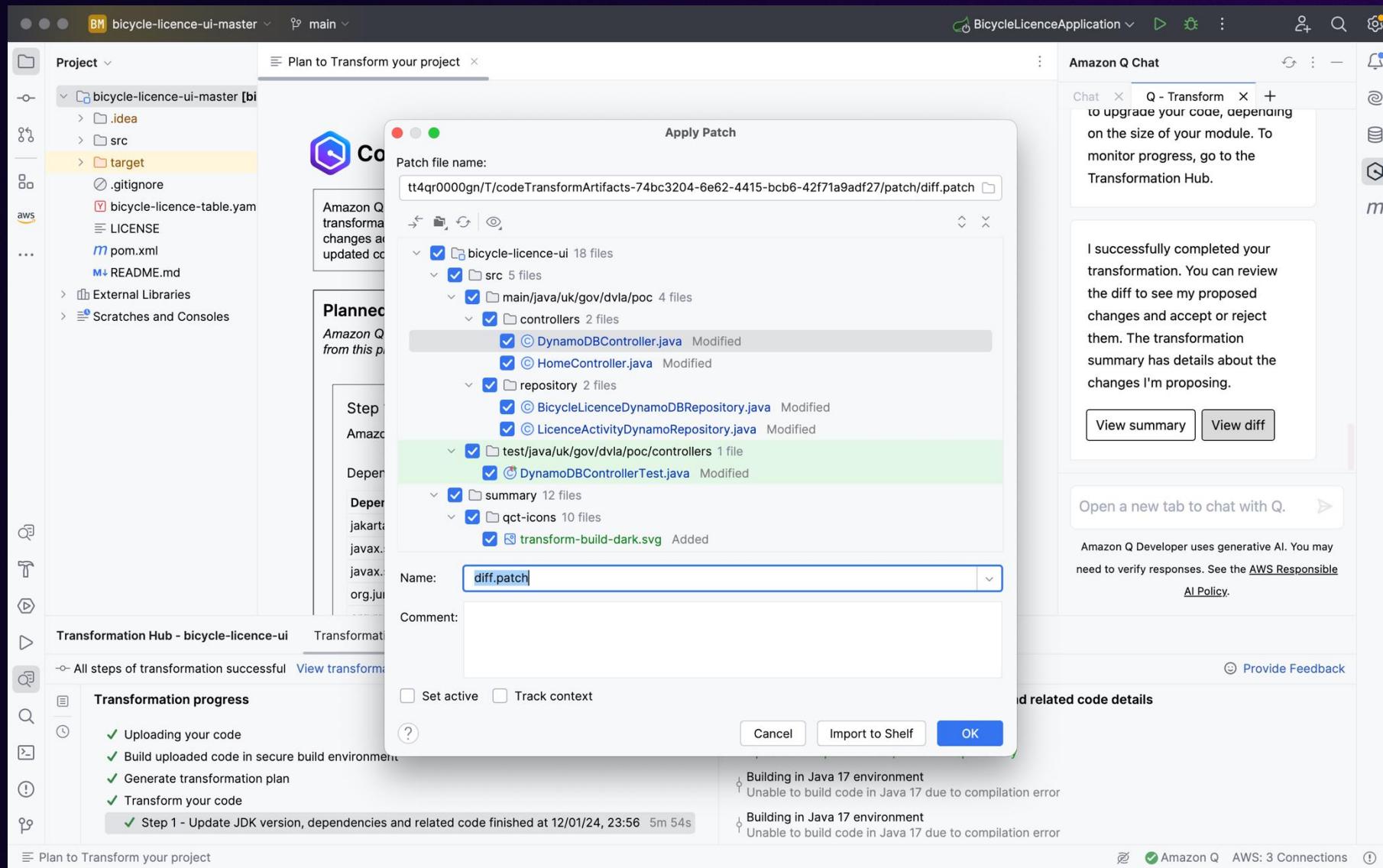


Build and test in Java 17

The screenshot shows the AWS Code Transformation plan by Amazon Q interface. The top navigation bar includes tabs for 'main' and 'BicycleLicenceApplication'. On the left, a sidebar shows a project structure for 'bicycle-licence-ui-master' with files like .idea, src, target, .gitignore, bicycle-licence-table.yaml, LICENSE, pom.xml, and README.md. A central panel displays the 'Code Transformation plan by Amazon Q' results, stating that Amazon Q reviewed the code and generated a transformation plan. It provides statistics: Lines of code in your application: 749, Dependencies to be replaced: 5, Deprecated code instances to be replaced: 0, and Files to be changed: 5. Below this, the 'Transformation Hub - bicycle-licence-ui' section shows a successful transformation process with four steps: Uploading your code, Build uploaded code in secure build environment, Generate transformation plan, and Transform your code. The final step, 'Transform your code', is expanded to show sub-tasks: Step 1 - Update JDK version, dependencies and related code details, Step 2 - Upgrade deprecated code, Step 3 - Apply code changes and resolve compilation errors, and Step 4 - Finalize code changes and generate transformation summary. The transformation summary indicates a running time of 12m 1s. To the right, there's an 'Amazon Q Chat' window titled 'Q - Transform' and a sidebar with various icons.



View code transformation diff



Implementing new interface method

The screenshot shows a Java code editor with a diff view between two versions of a repository class, `BicycleLicenceDynamoDBRepository.java`. The left column represents the **Base version** (2024-11-22), and the right column represents the **Transformed version**.

Base version (Left):

```
11  import lombok.extern.slf4j.Slf4j;
11  import lombok.extern.log4j.Log4j2;
12  import org.springframework.data.repository.CrudRepository;
13  import org.springframework.stereotype.Repository;
14  import uk.gov.dvla.poc.model.BicycleLicence;
15  import com.amazonaws.regions.Regions;
16
17  import java.util.*;
18  import java.util.stream.Collectors;
19  import java.util.stream.StreamSupport;
20
21  @Slf4j
22  @Repository
23  public class BicycleLicenceDynamoDBRepository implements CrudRepository<BicycleLicence, String> {
24
25  }
```

Transformed version (Right):

```
11  import lombok.extern.slf4j.Slf4j;
11  import lombok.extern.log4j.Log4j2;
12  import org.springframework.data.repository.CrudRepository;
13  import org.springframework.stereotype.Repository;
14  import uk.gov.dvla.poc.model.BicycleLicence;
15  import com.amazonaws.regions.Regions;
16
17  import java.util.*;
18  import java.util.stream.Collectors;
19  import java.util.stream.StreamSupport;
20
21  @Slf4j
22  @Repository
23  public class BicycleLicenceDynamoDBRepository implements CrudRepository<BicycleLicence, String> {
24
25  @Override
26  public void deleteAllById(Iterable<? extends String> ids) {
27      log.info("Deleting multiple licence documents by IDs from the DynamoDB table...");
28      List<BicycleLicence> licencesToDelete = StreamSupport.stream(idsspliterator(), false)
29          .map(this::findById)
30          .filter(Optional::isPresent)
31          .map(Optional::get)
32          .collect(Collectors.toList());
33      deleteAll(licencesToDelete);
34  }
35
36  private final ObjectMapper mapper = new ObjectMapper();
37}
```

The transformation includes:

- Imports for `lombok.extern.slf4j.Slf4j` and `lombok.extern.log4j.Log4j2`.
- Annotations `@Slf4j` and `@Repository`.
- A new implementation of the `deleteAllById` method using `StreamSupport`.



Code transformation summary

The screenshot shows the Amazon Q Code Transformation Summary interface. The main content area displays a summary of the transformation process, including:

- Lines of code in your application: 749
- Transformation duration: 11 min(s)
- Planned dependencies replaced: 2 of 6
- Additional dependencies added: 4
- Planned deprecated code instances replaced: 0 of 0
- Files changed: 6
- Build status in Java 17: SUCCEEDED

Below this is a Table of Contents:

1. Build log summary
2. Planned dependencies replaced
3. Additional dependencies added
4. Deprecated code replaced
5. Other changes
6. All files changed
7. Next steps

The Build log summary section shows a snippet of the build log:

```
Amazon Q successfully built the upgraded code in Java 17. Here is a relevant snippet from the build log. To view the full build log, open buildCommandOutput.log
```

The Maven build was successful and ran 6 tests without any failures, errors or skips. The tests ran the DynamoDBController, logging info messages about updating licenses and ch

The Planned dependencies replaced section shows the following table:

Dependency	Action	Previous version in Java 8	Current version in Java 17
jakarta.servlet:jakarta.servlet-api	Added	-	6.1.0
org.springframework.boot:spring-boot-starter-parent	Updated	2.1.8.RELEASE	3.3.6

On the right side, there is an Amazon Q Chat window with the following history:

- Q - Transform + use mvn test-compile.
- Okay, I will run unit tests when building your module.
- I was able to build your module and will start uploading your code.
- I'm starting to transform your code. It can take 10 to 30 minutes to upgrade your code, depending on the size of your module. To monitor progress, go to the Transformation Hub.
- I successfully upgraded your code to Java 17. You can review the diff to see my proposed changes and accept or reject them. The transformation summary has details about the files I updated.

At the bottom, there are buttons for "View summary" and "View diff".



Amazon's internal use of Amazon Q



CEO Andy Jassy reported in a recent [LinkedIn post](#) that the internal developer tooling team used Q Developer's new code transformation capability—specifically integrating it with internal development systems—to automate the process of upgrading over 30,000 applications from Java 8 or 11 to Java 17, saving over 4,500 developer years compared to manual upgrades.

Source: <https://news.a2z.com/contents/38885081>

LinkedIn post: https://www.linkedin.com/posts/andy-jassy-8b1615_one-of-the-most-tedious-but-critical-tasks-activity-7232374162185461760-AdSz/?utm_source=share&utm_medium=member_desktop

Getting Started



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Deployment decisions

Free tier



Code faster with code suggestions in the IDE and CLI



Use where you work: Your IDE, CLI, the AWS Management Console, etc.



Limited monthly access of advanced features

Pro tier



Manage users and policies with enterprise access controls



Customize Amazon Q to your code base



High limits of advanced features

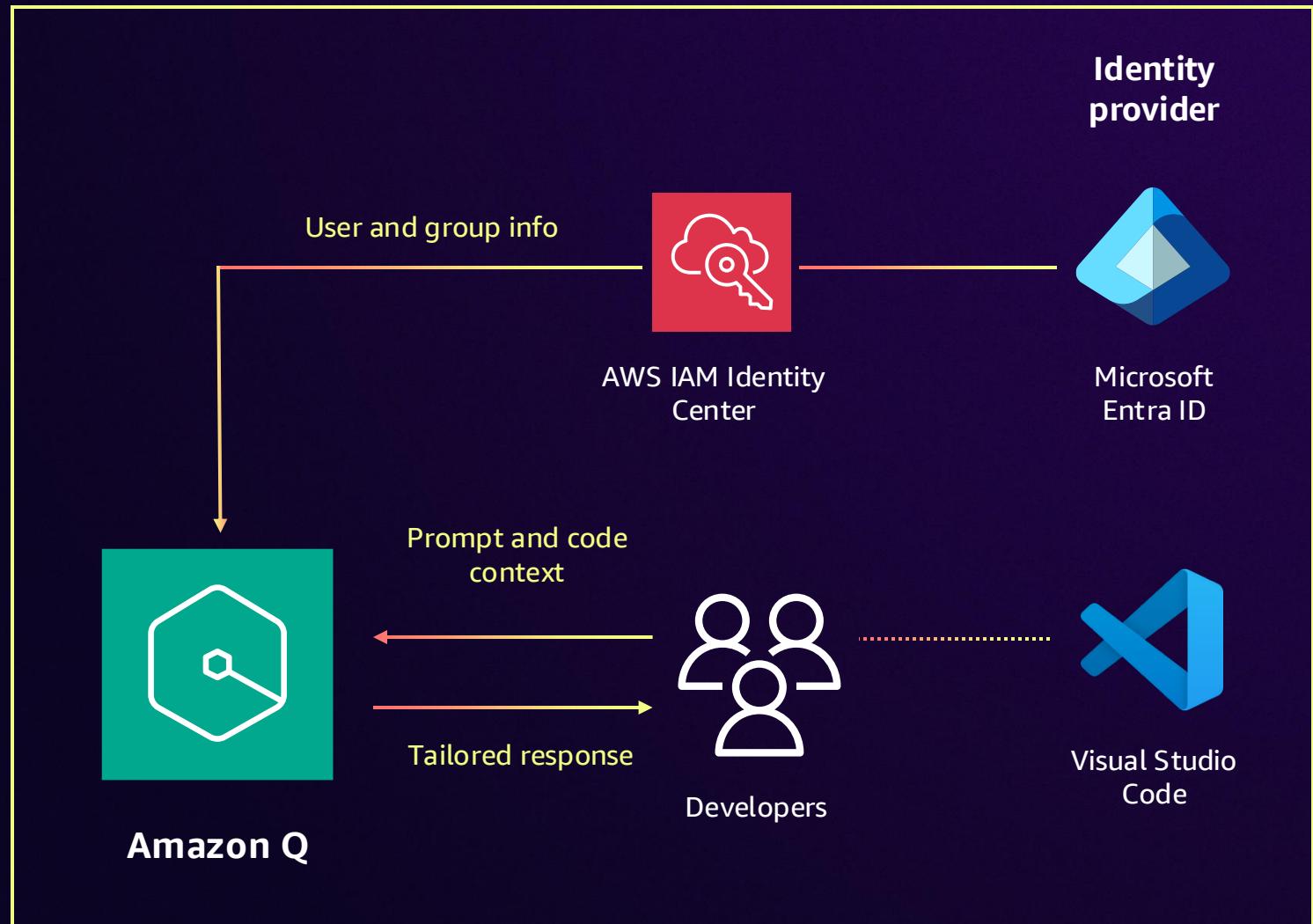
Single sign-on (SSO) integration

User management

Seamlessly match users' corporate credentials and provide access to Amazon Q

Security

Developers securely access and use Amazon Q Developer



Accelerate Evaluation with Workshop Exercises

aws workshop studio

Amazon Q Developer Workshop - Building the Q-Words App

- Introduction
- ▶ Setup and Configuration
- ▶ Code Comprehension and Summarization
- ▶ Code Debugging, Testing, and Optimization
- ▶ Natural Language to New Feature
- ▶ Application Modernization and Transformation
- ▶ Generative AI for Accelerating Software Development
- Resource Cleanup
- Wrapping Up: Next Steps and

Amazon Q Developer Workshop - Building the Q-Words ...

Amazon Q Developer Workshop - Building the Q-Words App



The workshop page displays the title "Amazon Q Developer Workshop - Building the Q-Words App". Below the title is a large image featuring the Amazon Q Developer logo, which consists of the text "Amazon Q Developer" and "Your generative AI-powered assistant" next to a stylized hexagonal icon.

aws workshop studio

Easily port .NET Frameworks apps to Linux with Amazon Q Developer .NET transformation capabilities

- Access Lab Environment
- ▶ Setup
- ▶ Lab 1 - IDE
- ▶ Lab 2 - Web Experience
- Cleanup
- Summary

AWS Documentation Homepage ↗

Content preferences

Language

Easily port .NET Frameworks apps to Linux with Amazon ...

Easily port .NET Frameworks apps to Linux with Amazon Q Developer .NET transformation capabilities

In this workshop, gain hands-on experience porting a .NET Framework application to cross-platform .NET. Use Amazon Q Developer .NET transformation capabilities, a modernization tool powered by generative AI that significantly reduces the time to migrate applications from Windows Server to Linux. Migrating to Linux helps you avoid additional licensing fees and gain performance and security benefits.

<https://catalog.workshops.aws/qwords/en-US>
<https://catalog.workshops.aws/qdevtransformdotnet/en-US>



Developers can be more productive

Accelerate workflow

Reduce repetitive work

Focus on higher value activities



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

THE WEB THE API THE ML THE AI THE DB THE FS

Thank you!

Michael Lin

linmicht@amazon.com



© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.