

# Next-Generation Software Development

**Jeff Barr**

(he/him)

[jbarr@amazon.com](mailto:jbarr@amazon.com)

VP & Chief Evangelist, AWS



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

# Hello, I am Jeff

## • Who I Am

- VP and Chief Evangelist, AWS
- Husband, parent, grandparent
- Technologist
- Maker (3D printing, LEGO, electronics)
- BS - Computer Science (1985)
- Masters of Communication in Digital Media (2013)
- Professional developer since 1979
- Spoke at first JAWS meeting in 2010



## • What I Do

- AWS News Blog (2004-2024)
- Social media
- AWS OnAir live stream
- Global events
- Customer meetings



# February 23, 2010 First JAWS-UG Meeting



# Introducing Angelo Shirahata, President AWSJ

- President of AWSJ since November 2024
- Education:
  - Keio University
  - Bond University (Australia)
- Career:
  - Nissho Iwai Corporation
  - General Electric
  - Schneider Electric



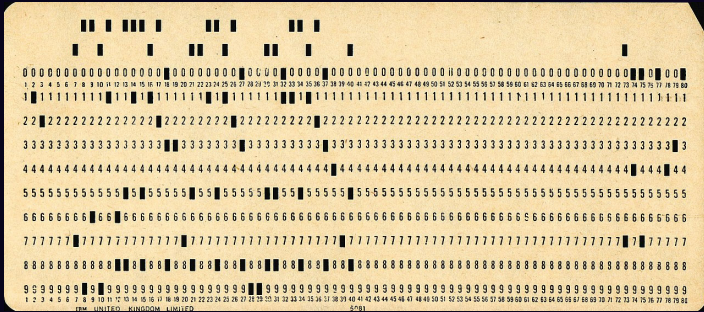




© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# My development background

- Mainframe - PL/I
- PDP-8 - BASIC
- PDP-11 – C
- Microcomputers – 8080, 6502, 68000
- C, PHP, Python, Perl



```
ec2-3-212-214-72.compute-1.amazonaws.com - PuTTY
File Edit Options Buffers Tools C Help
Highlight (CurCursor);
if (THE_ITEM (*CurCursor, CurCursor -> Index).Flags & FL_SELECT)
    Select (CurCursor);
    Cue (DirCue_Msg);
    if (FakeReload () == ERROR)
        return (ERROR);
    }
    break;

case K_HELP:
#endif SECURITY
    if (tm_testbit(sec_map, MAIN_HELP))
    {
        if (SecNote(NULL) == ERROR)
            return(ERROR);
        break;
    }
#endif

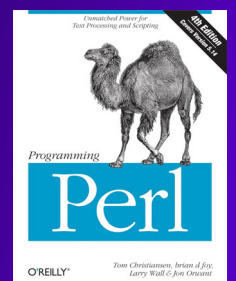
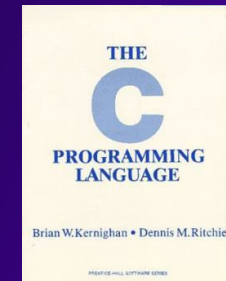
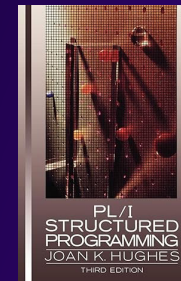
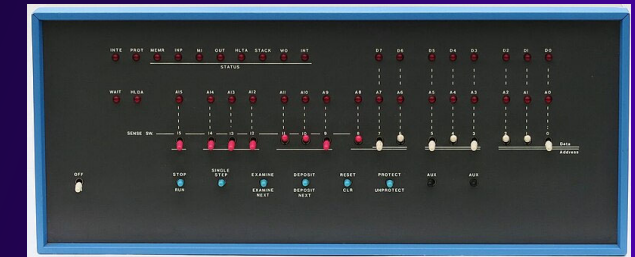
give_help:
Unhighlight (CurCursor);
if (ui_help (CurCursor == &DirCursor ? DirDirHlp : DirFileHlp, 0) ==
    ERROR)
    return (ERROR);
    Highlight (CurCursor);
    Cue (DirCue_Msg);
    break;

case K_CMD:
#endif SECURITY
    if (tm_testbit(sec_map, MAIN_CMD))
    {
        if (SecNote(NULL) == ERROR)
            return(ERROR);
        break;
    }
#endif

/*
 * Do command processing. Since things like Ad Hoc commands
 * and Shell already do a smart reload, and we would like to
 * do a smart reload here, rather than put up a reload message,
 * we use the global DidReload. To prevent us from re-triggering
 * the smart reload in the above cases where it had already
 * been called.
 */
DidReload = FALSE;
SayReload = FALSE;
Unhighlight (CurCursor);
if ((RetVal = Command (FileCursor.SelCount, DirCursor.SelCount)) ==
    ERROR)
{
    KeyEditing = FALSE;
    break;
}

Highlight (CurCursor);
Cue (DirCue_Msg);

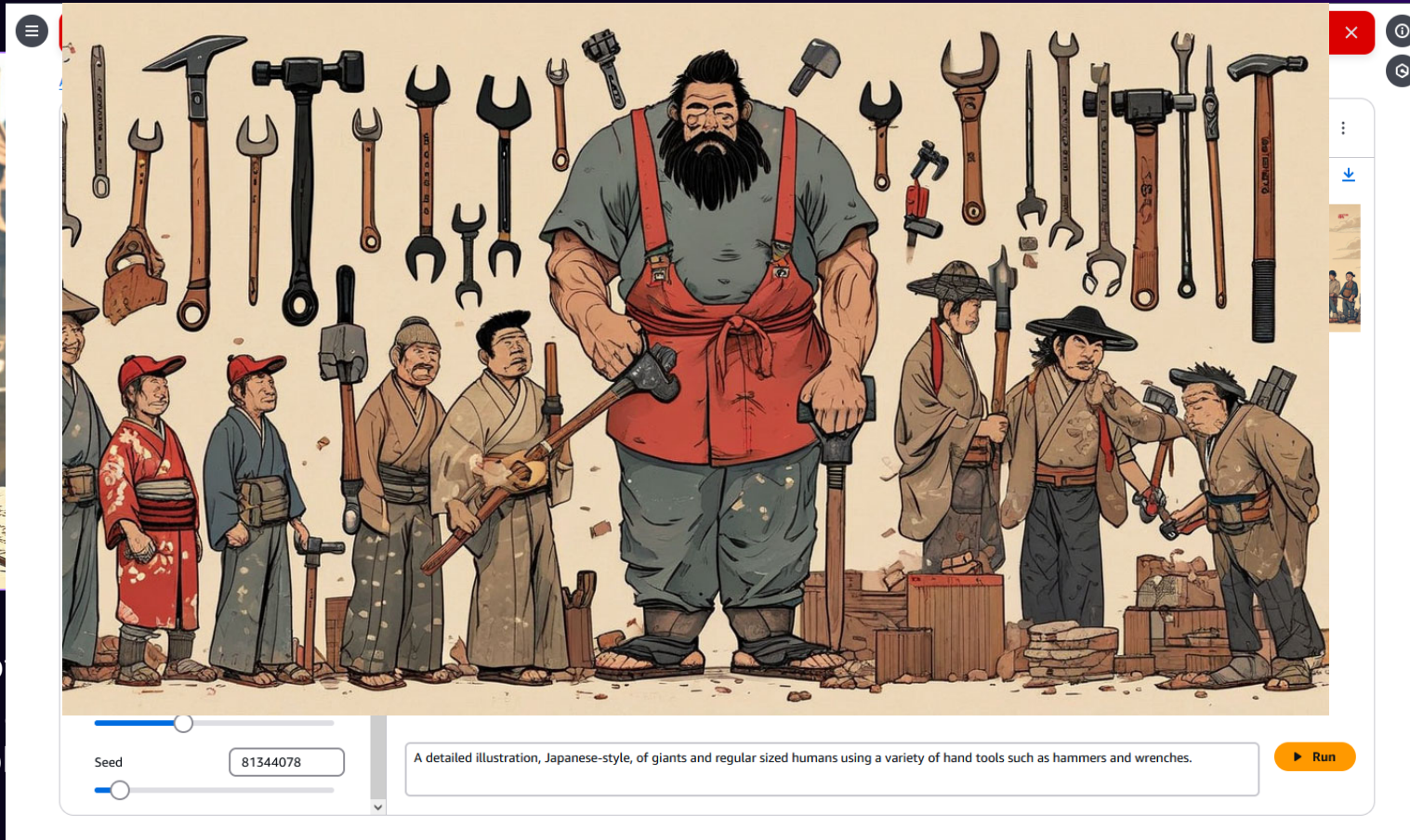
/*
 * If we did not do a smart reload and SayReload was not
 * set, then we should check if a reload is needed.
 * Otherwise, if SayReload was set, and we haven't done a smart
 * reload, do it now, and manipulate the display appropriately.
 */
if (DidReload == FALSE)
```



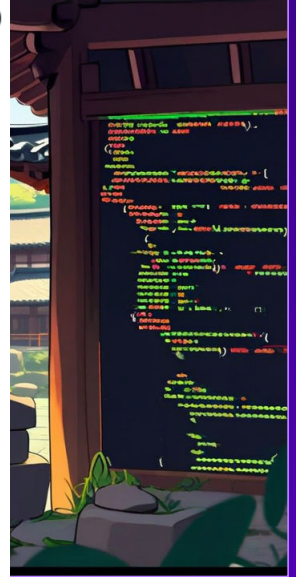
# Images produced by Amazon Bedrock & Amazon Nova Canvas



“a female so  
hammer and  
scene loo



“A detailed illustration, Japanese-style, of giants and  
regular sized humans using a variety of hand tools such as  
hammers and wrenches.”



ode using a  
n stone. The  
e 1600s.”

# Let's Dive In!



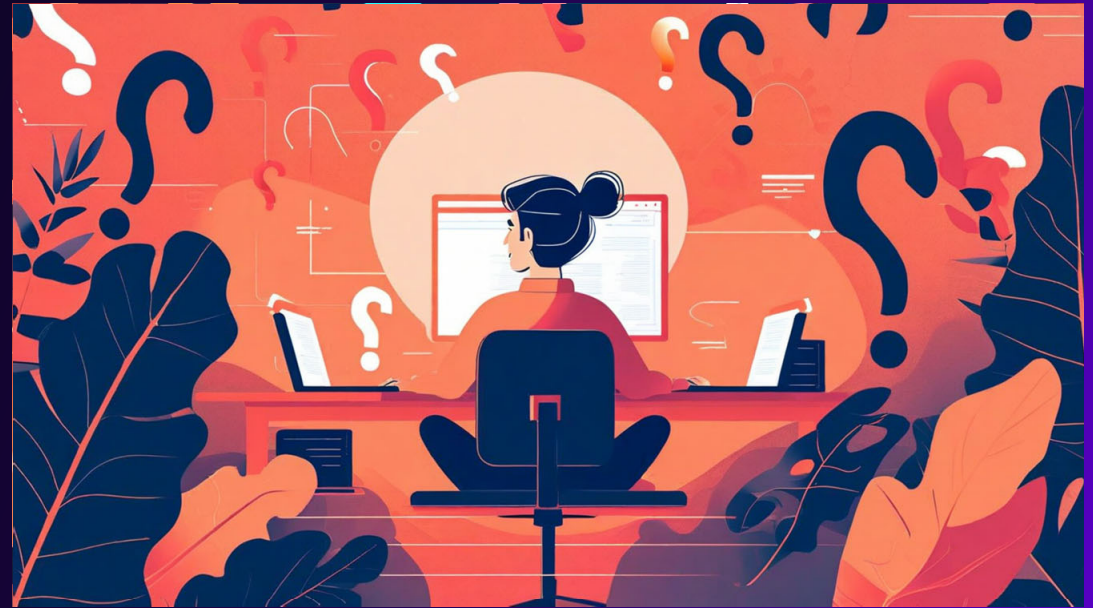
© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.





# An exceptionally interesting point in time

- The practice of software development continues to evolve
- Many developers express a feeling of uncertainty
- How did we get here?
- Where are we going?
- Where do I fit in?
- How do we prepare?



“A software developer surrounded by question marks.”





# Where are we today?

- [Some/Many] developers are creating space-age programs using outdated tools and processes!
- There's room for everyone:
  - Informal / self-taught
  - CS-educated / professional coder
  - Math-educated / formal approach

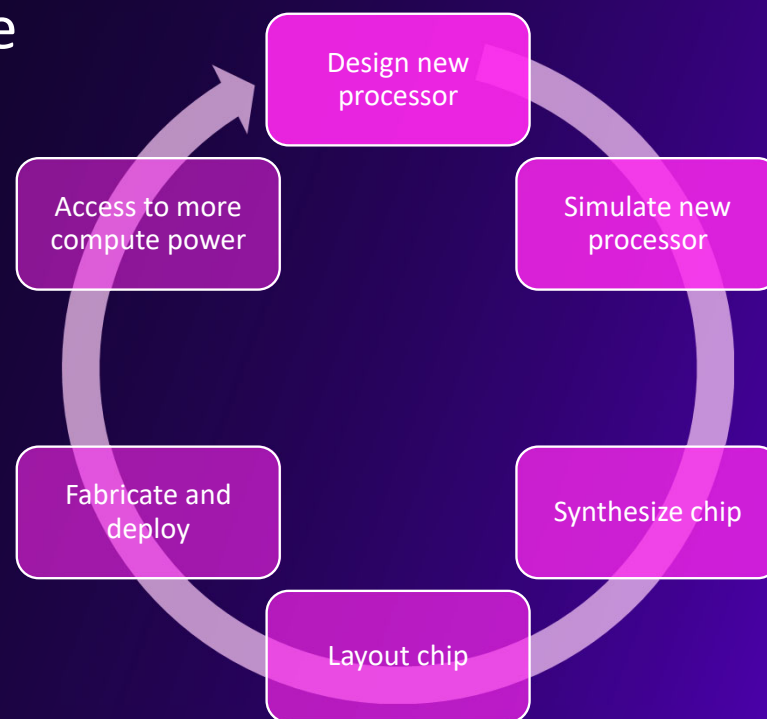


“a female software developer writing code using a hammer and a chisel to carve the code in stone. The scene looks like a Japanese city in the 1600s.”



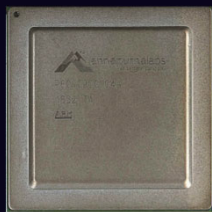
# The Amazon virtuous cycle in action for hardware

- Each generation of hardware accelerates production of the next one
- Time between generations shrinks
- Each generation more powerful & capable

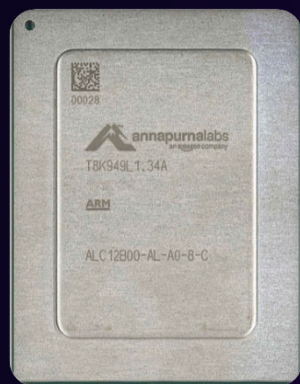


# Four generations of AWS Graviton chips

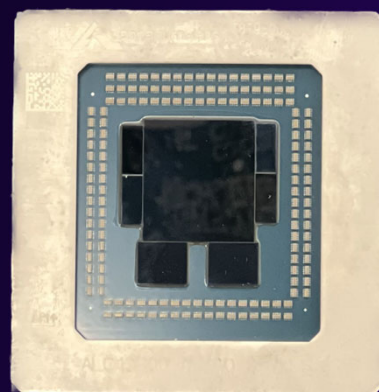
Graviton  
2018



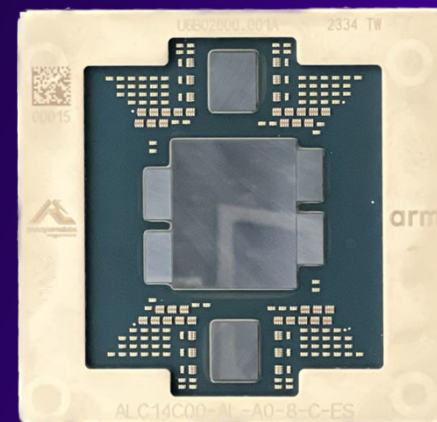
Graviton2  
2019



Graviton3  
2021



Graviton4  
2023

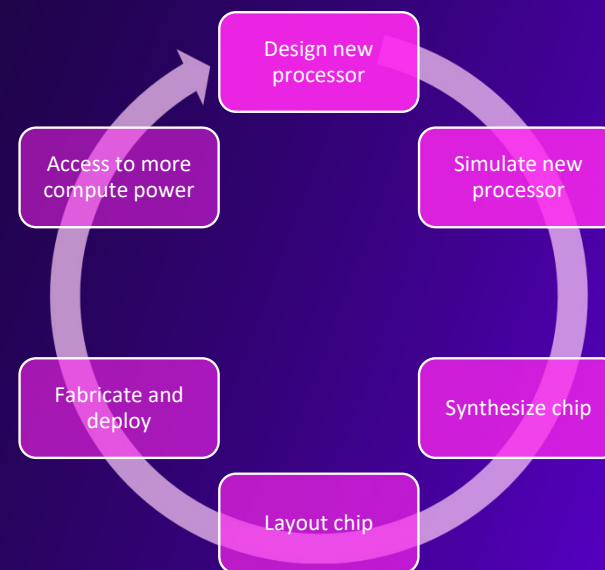


# Graviton Improves EDA Workloads

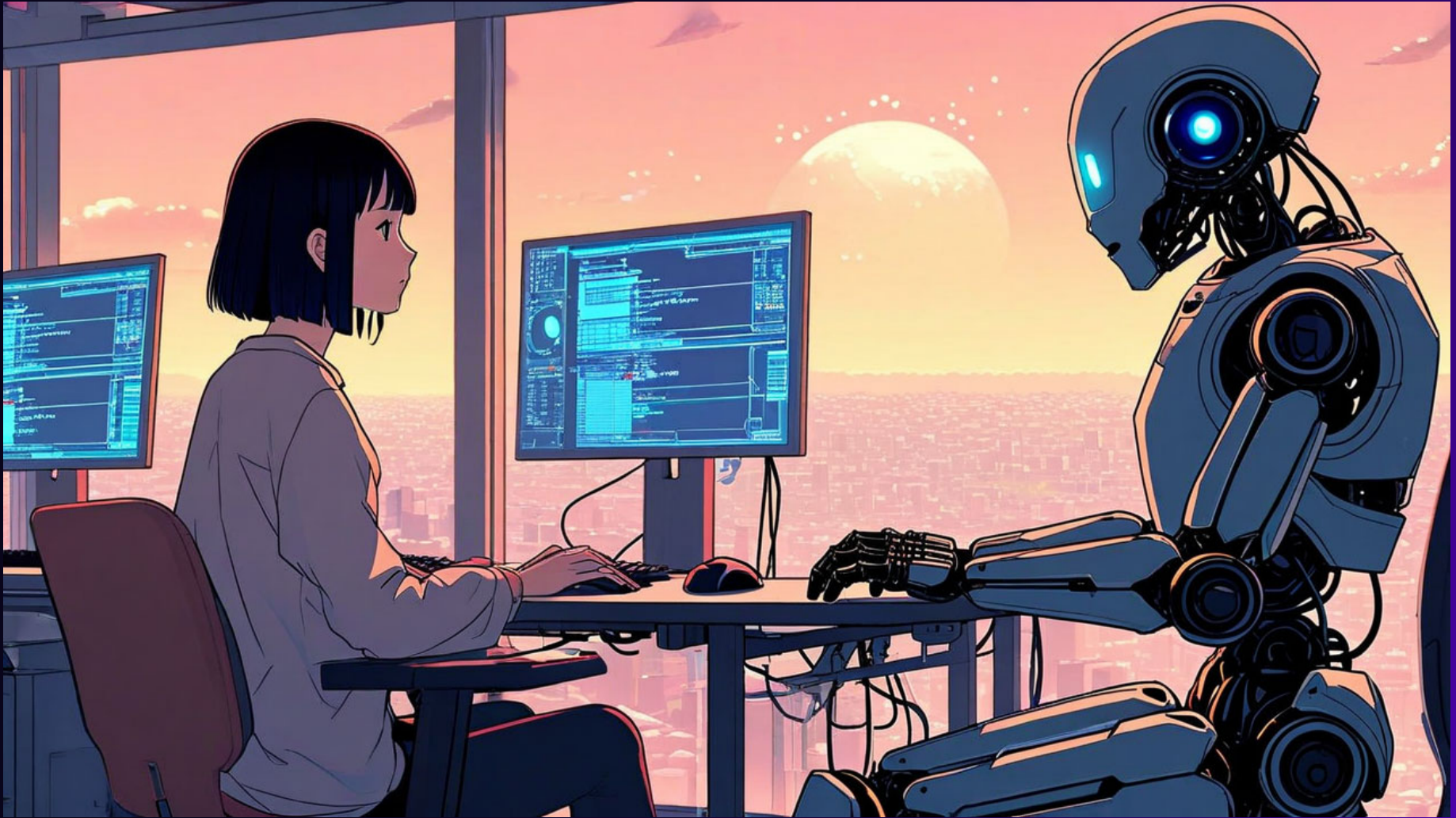
100s of  
Millions of  
CPU Hours  
to simulate and test  
designs

Over  
\$100 Million  
to design and build  
a modern CPU

**Arm Validates IP 1,000X+ Faster with  
Solido on AWS Graviton2**





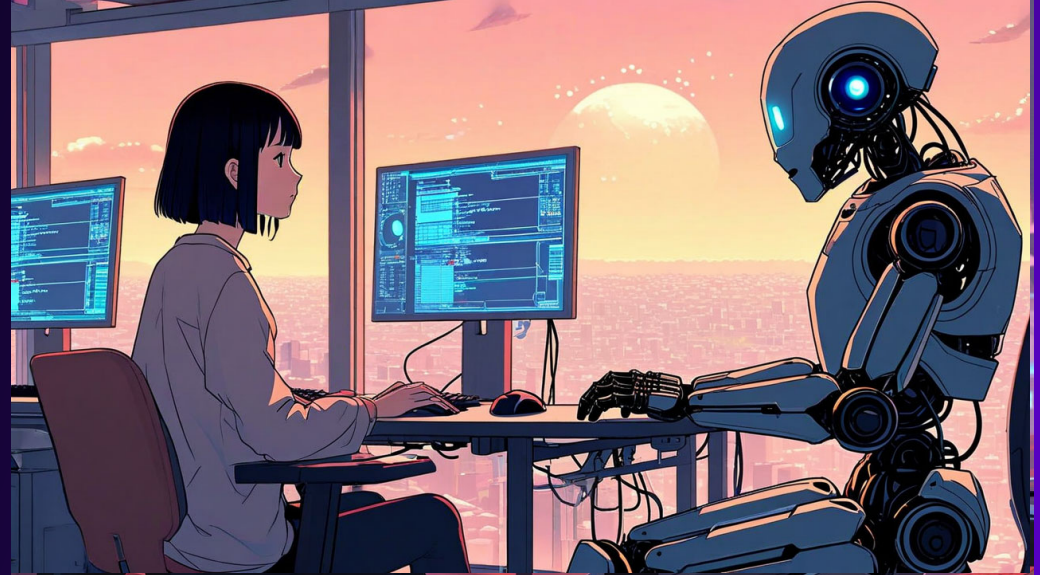




# A similar virtuous cycle for software

- Better computers support better and richer software
- We have been using software to build software since the beginning
- AI-powered coding assistants are just the next step
- Make you more powerful
- An assistant, not a replacement

“A female Japanese software developer sitting next to a robot, **wiht** both of them at their own computers, all in a japanese manga sci-fi landscape”



# The future is here but not yet evenly distributed

- Where are you today?

Hand-writing  
instructions  
using 0's and 1's

Coding in  
assembly  
language

Coding in a  
higher-level  
language

Using an  
AI-powered  
coding assistant

Using formal  
methods to  
validate your  
code

Using a  
Quantum  
Computer



# Developers and code

- Developer's responsibility:
  - Produce code that meets self-imposed or customer-driven requirements
- Desirable code attributes:
  - Correct – Behaves as expected
  - Fast – Makes efficient use of resources
  - Maintainable – Easy to update
  - “Good” (a qualitative measure)



“developer and source code at a white board,  
in a futuristic Japanese setting”



# It's all about the tools!



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



# #define DeveloperTools "..."

A developer tool is anything made from bits that helps you to do your job:

- Language feature
- Executable (editor, compiler)
- Function or class library
- Training & reference material
- ...



"A futuristic Japanese-style library with books, racks of computers, keyboards, some hand tools, and more. There are three people sitting at a table in the middle."

# Why developer tools matter

- Tools let us “stand on the shoulder of giants” (Isaac Newton)
- Tools build on and compose with other tools
- Tools improve with (and take advantage of) available compute power, and grow more powerful themselves as a result



“A detailed illustration, Japanese-style, of giants and regular sized humans using a variety of hand tools such as hammers and wrenches.”

# Tools Over Time

Assembler

Editor

Compiler

Lint

Debugger

CASE tools

IDE

Source code control

Profiler

Indenting

Colorization

Syntax-directed editing

Auto-complete

IntelliSense

Programmability

Single step

Disassemble

Breakpoints

Call stack

Symbols

Base compiler

Optimizing compiler

Checkout compiler

Instructional compiler

Diagramming

Modeling languages

Flow charts

Visual programming

# Tools Over Time

Assembler

Editor

Compiler

Lint

Debugger

CASE tools

IDE

Source code control

Profiler

Diagramming Base compiler

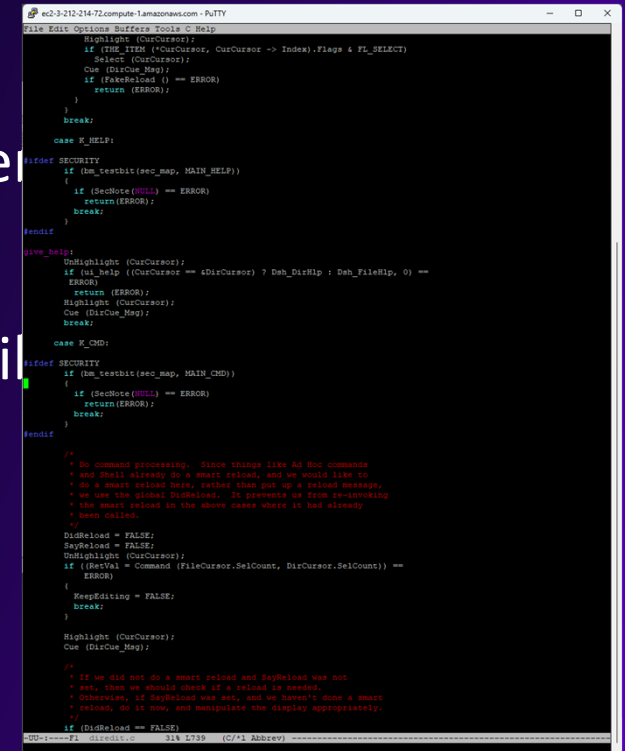
Object-oriented programming languages

System-level structured editing compiler

Abstract programming functional compiler

Symbolic

Programmability



```
File Edit Options Buffers Tools C Help
Highlight (CurCursor);
if (THE_ITEM (*CurCursor, CurCursor -> Index).Flags & FL_SELECT)
    Select (CurCursor);
    Cue (DisCue_Map);
    if (FakeReload () == ERROR)
        return (ERROR);
    }
    break;
case K_HELP:
#ifdef SECURITY
    if (tm_testbit(sec_map, MAIN_HELP))
    {
        if (SecNote(NULL) == ERROR)
            return(ERROR);
        break;
    }
#endif
give_help:
Unhighlight (CurCursor);
if (hi_help ((CurCursor == 4DisCursor) ? Deb_Help : Deb_FileMap, 0) ==
    ERROR)
    return (ERROR);
    Highlight (CurCursor);
    Cue (DisCue_Map);
    break;
case K_CMD:
#ifdef SECURITY
    if (tm_testbit(sec_map, MAIN_CMD))
    {
        if (SecNote(NULL) == ERROR)
            return(ERROR);
        break;
    }
#endif
/*
 * No command processing. Since things like fd Noc commands
 * and Noc already do a smart reload, and we would like to
 * do a smart reload here, rather than put up a reload message,
 * we use the global DisReload. It prevents us from re-invoking
 * the smart reload in the above cases where it had already
 * been called.
 */
DisReload = FALSE;
SayReload = FALSE;
Unhighlight (CurCursor);
if ((SetVal = Command (FileCursor.SelCount, DisCursor.SelCount)) ==
    ERROR)
{
    KeepEditing = FALSE;
    break;
}
Highlight (CurCursor);
Cue (DisCue_Map);
/*
 * If we did not do a smart reload and SayReload was not
 * set, then we should check if a reload is needed.
 * Otherwise, if SayReload was set, and we haven't done a smart
 * reload, do it now, and manipulate the display appropriately.
 */
if (DisReload == FALSE)
```





# Language features

## Compile time

- Structured programming
- Data types
- Objects
- Exceptions
- Memory safety

## Run time

- Array bounds checking
- Exception handling
- Memory management / GC
- Overflow detection
- Assertion

**Each one helps us to do a better job and to write better code**

# Packaging and reusable code

- Macro libraries
- Standard libraries
- Classes
- Open source



“happy software developers on the beach, looking out at a glorious rainbow made up of source code fragments.”

# And my point is....

- As humans, we conceptualize, build, and use tools
- Then we use them to build even more tools
- As developers we do the same
- Tools are good
- Respect (don't fear) progress

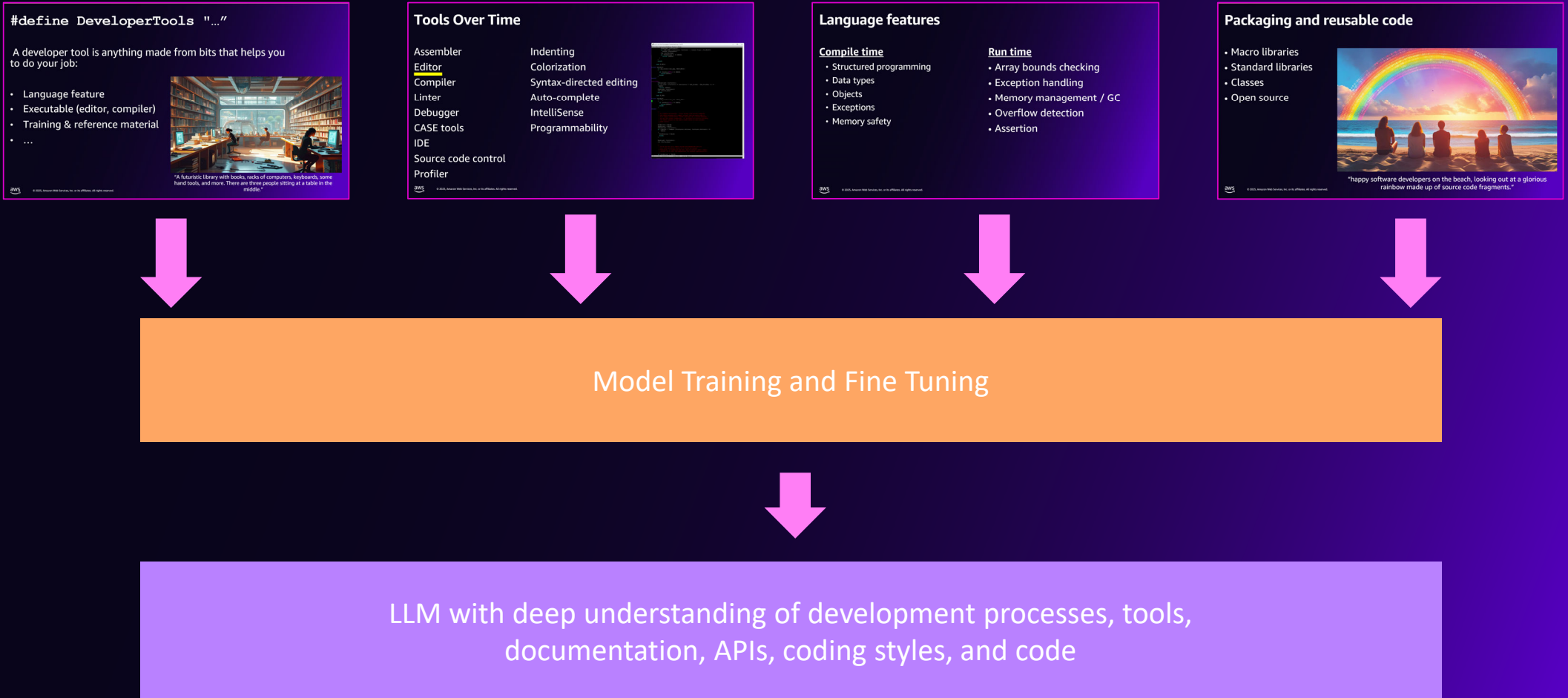


© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.





# Mix it all together



# What's Next / What's New / What's Now?



“a highway that leads to a space-age city. There are several styles of signs that say "New" along the highway, and some floating in the air. Make it look really, really amazing!”

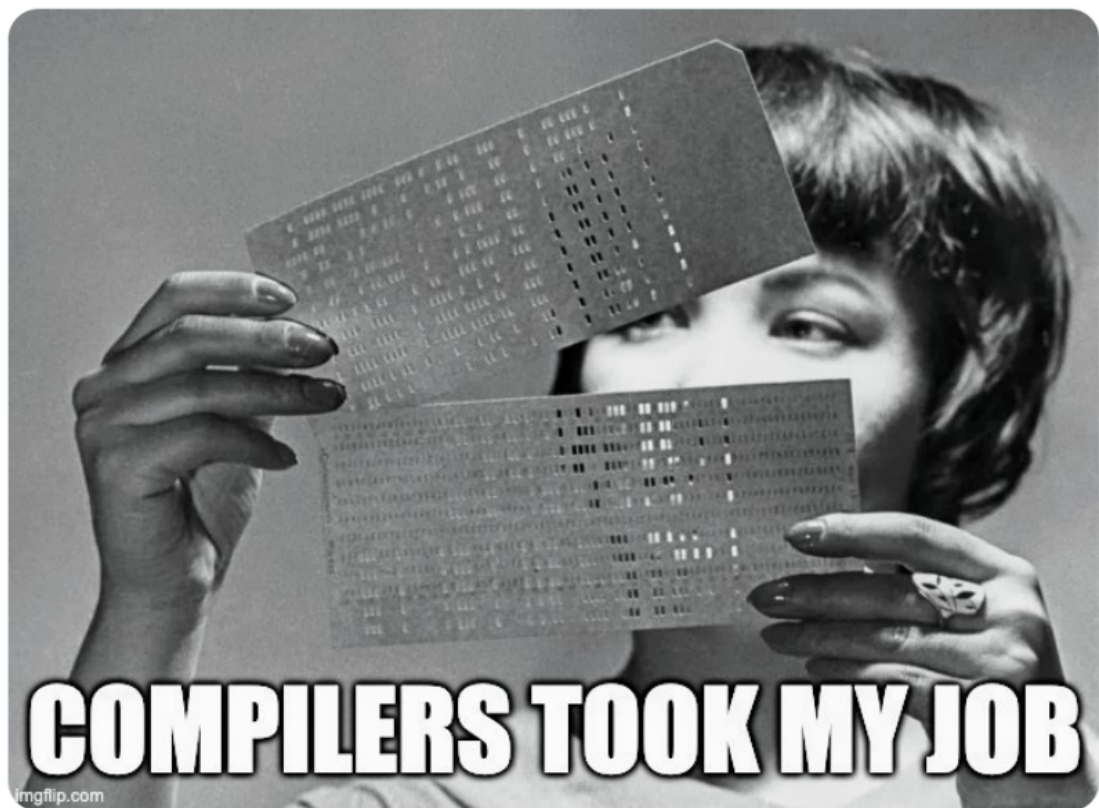


Ryan Els

@RyanEls4



Software developers in the 60s 🤔



“The image in the post humorously references the historical impact of compilers on software development jobs, reflecting on how automation and technological advancements have historically displaced certain roles, similar to concerns today about AI and automation.”

“The post plays on the current anxiety about AI taking jobs, drawing a parallel with the 1960s when compilers began to automate tasks previously done manually, suggesting a cycle of technological displacement in the tech industry.”



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



# What's next / What's new / What's now?

## AI-Powered Coding Assistant

- Logical next step
- Make you a better developer
- From intent to code
- Increase efficiency
- Another tool for your toolbox
- Builds on what you know
- Amazon Q Developer

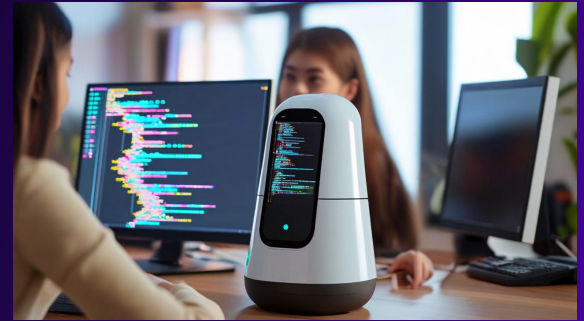
## Formal Verification

- Mathematical basis
- Spec vs implementation
- Prove programs correct
- Automated reasoning and other leading edge techniques
- Heavily used at Amazon
- Will become mainstream



# Amazon Q Developer

## AI-Powered Coding Assistant



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

# Create an app with Amazon Q Developer

```
*Untitled - Notepad
File Edit Format View Help
I want to create a full-stack web application for Sports Event Management with the following features:

Frontend (React.js):

1. Authentication system
  - React bootstrap index.html and index.js files
  - Login page
  - Registration page
  - Protected routes
  - JWT token management

2. User Interface
  - Modern, responsive design using Material-UI
  - Navigation bar with login/logout functionality
  - Add a logo
  - Dashboard layout

3. Sports Event Management
  - Display list of sports events (Football, Cricket, Tennis, Table Tennis, Badminton, Basketball, Handball, Hockey)
  - CRUD operations for events (Create, Read, Update, Delete)
  - Event details including:
    * Event name
    * Event Date
    * Teams (Team 1 vs Team 2)
    * Scores
    * Venue
    * Sport type
    * Winner
  - Filtering and sorting capabilities
  - Form validation
  - Confirmation dialogs for delete operations

4. Role-based Access Control
  - Admin users: Full CRUD access
  - Regular users: Read-only access
  - User profile management
```

## Backend (Node.js/Express):

1. API Architecture
  - RESTful API design
  - MVC pattern
  - Error handling middleware
  - Input validation
  - CORS configuration
2. Authentication
  - JWT-based authentication
  - Password hashing
  - Role-based authorization middleware
3. Database (In Memory DB)
  - User schema
  - Add an admin user in advance in db in config file
  - Sports events schema
4. Features
  - CRUD operations for events
  - User management
  - Error logging
  - Input sanitization
  - Rate limiting

Please provide guidance on how to implement this application with best practices and secure coding standards

Ln 42, Col 17 100% Windows (CRLF) UTF-8

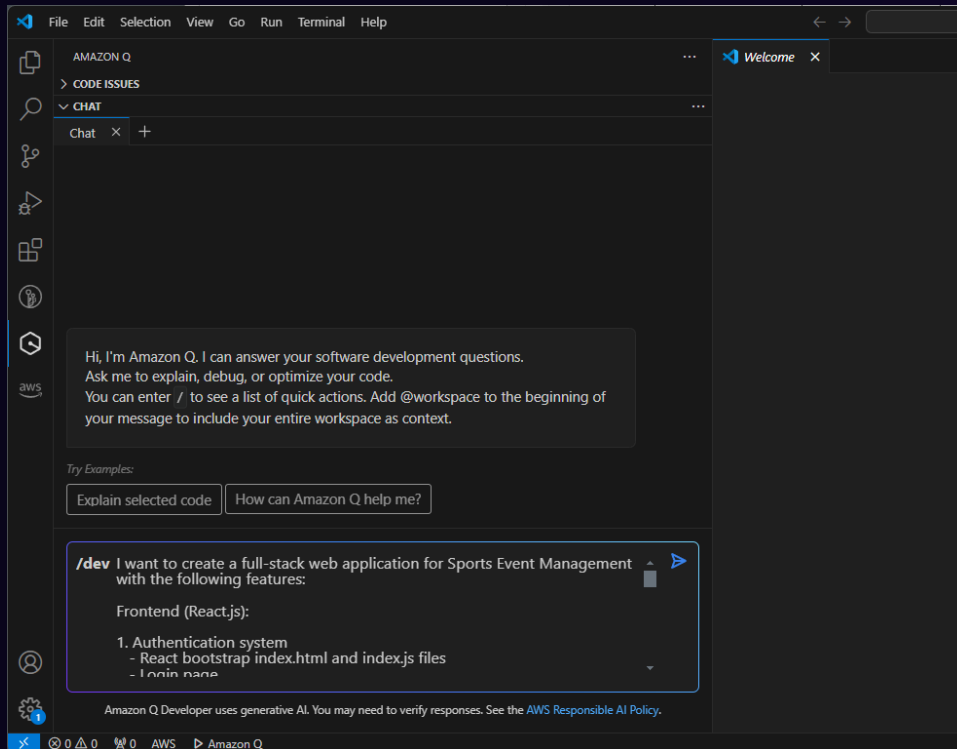


Sanjeev Kumar

<https://www.youtube.com/watch?v=06rB0yNrJT8>

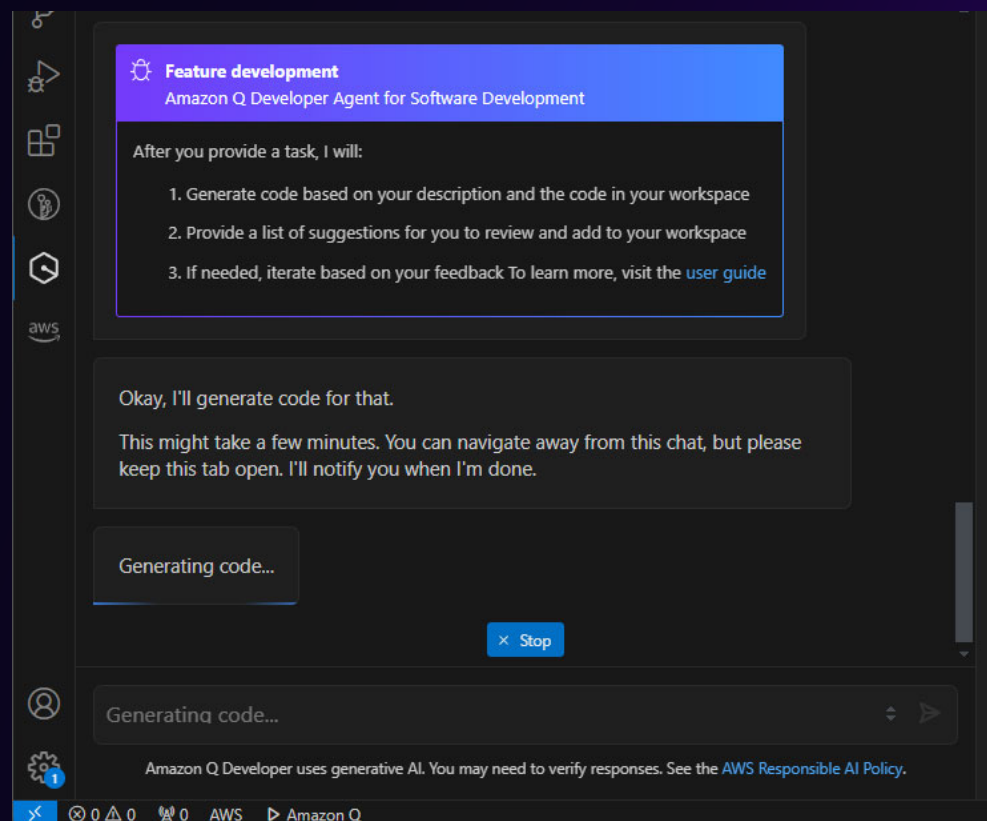


# Amazon Q Developer in action

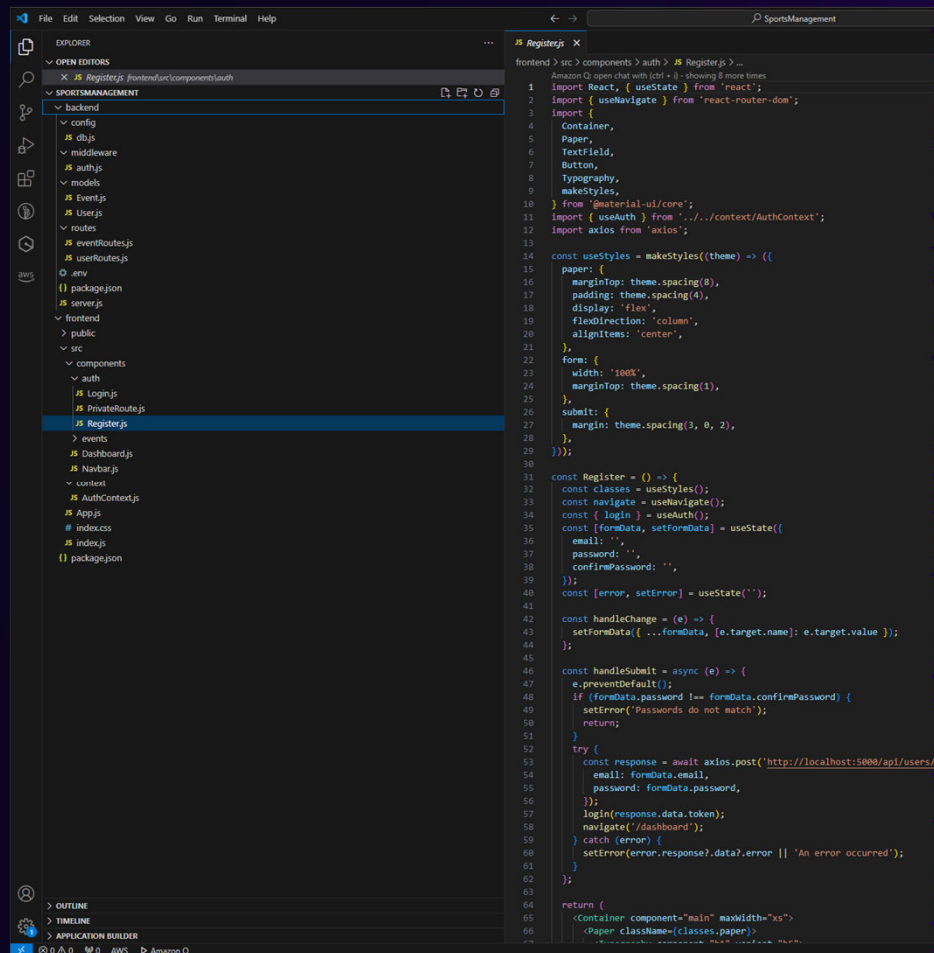


- Create plan
- Look for all features
- Write code
- Create project structure
- Add features and files
- Create summary of changes
- Ask user to accept code

# Amazon Q Developer generating code



# Amazon Q Developer generated code

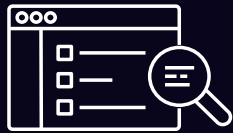


```
1 import React, { useState } from 'react';
2 import { useNavigate } from 'react-router-dom';
3 import {
4   Container,
5   Paper,
6   TextField,
7   Button,
8   Typography,
9   makeStyles,
10 } from '@material-ui/core';
11 import { useAuth } from '../context/AuthContext';
12 import axios from 'axios';
13
14 const useStyles = makeStyles((theme) => ({
15   paper: {
16     marginTop: theme.spacing(8),
17     padding: theme.spacing(4),
18     display: 'flex',
19     flexDirection: 'column',
20     alignItems: 'center',
21   },
22   form: {
23     width: '100%',
24     marginTop: theme.spacing(1),
25   },
26   submit: {
27     margin: theme.spacing(3, 0, 2),
28   },
29 }));
30
31 const Register = () => {
32   const classes = useStyles();
33   const navigate = useNavigate();
34   const [ login ] = useAuth();
35   const [formData, setFormData] = useState({
36     email: '',
37     password: '',
38     confirmPassword: '',
39   });
40   const [error, setError] = useState('');
41
42   const handleChange = (e) => {
43     setFormData({ ...formData, [e.target.name]: e.target.value });
44   };
45
46   const handleSubmit = async (e) => {
47     e.preventDefault();
48     if (formData.password !== formData.confirmPassword) {
49       setError("Passwords do not match");
50       return;
51     }
52     try {
53       const response = await axios.post('http://localhost:5000/api/users/register', {
54         email: formData.email,
55         password: formData.password,
56       });
57       login(response.data.token);
58       navigate('/dashboard');
59     } catch (error) {
60       setError(error.response?.data?.error || 'An error occurred');
61     }
62   };
63
64   return (
65     <Container component="main" maxWidth="xs">
66       <Paper className={classes.paper}>
```



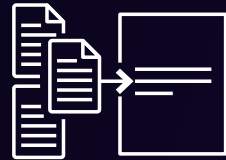


# Amazon Q Developer top capabilities



## Build faster

Accelerate tasks across the entire SDLC with generative AI-powered capabilities and agents



## Operate on AWS

Get expert guidance for building on AWS, managing and optimizing AWS cloud resources, and diagnosing and troubleshooting errors.



## Transform workloads

Modernize .NET, mainframe, VMware, and Java workloads at enterprise-scale to optimize processes and reduce costs



## Leverage Data and AI

Get guidance to quickly and easily build analytics, AI/ML, and generative AI applications.

**Faster Innovation**



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

# Amazon Q Developer for SDLC

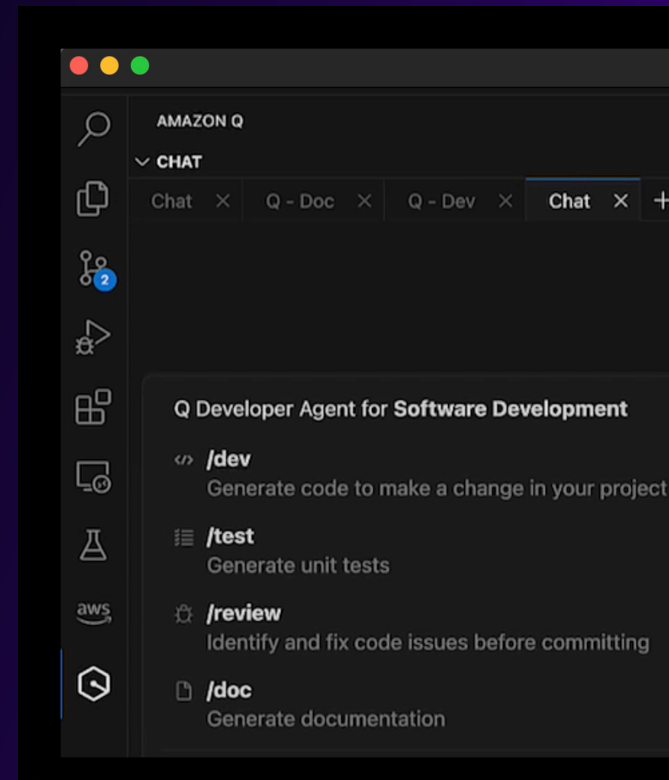
GENERATING CODE REVIEW, TESTS, DOCUMENTATION, GITLAB INTEGRATION, AND OPS HELP

With announcements presented this year, Amazon Q Developer got 3 new agent capabilities:

1. Enhancing documentation in codebases via **/doc** command.
2. Supporting code reviews to detect and resolve security and code quality issues via **/review**.
3. Generating unit tests automatically and improving test coverage **/test**.

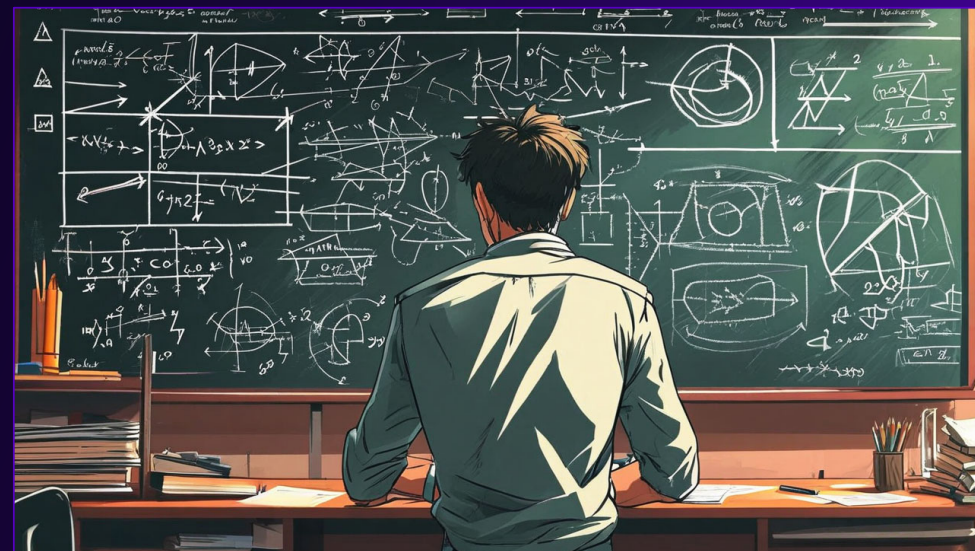
All of the above in your preferred IDE or GitLab Duo with Amazon Q (*preview*), which is one of the most popular enterprise DevOps platforms.

Additionally, there is a new capability in Amazon Q Developer to **investigate and remediate operational issues** (*preview*).



# Formal verification

- Mathematically or logically rigorous
- Proof of correctness
- From academia to industry
- Value grows with complexity of system



“A mathematician at a chalk board, working on a complex problem that has a lot of mathematical formulas, diagrams, lines, and arrows, in a manga style.”



## Machine Learning

Large amount  
of data  
(examples of  
what to predict)



Training



Model



Predictions  
(inference)

The model can generalize  
to unseen data but is not  
100% correct

## Automated Reasoning

Specific  
information  
(a company policy,  
a business process,  
an operational workflow,  
a checklist,  
programming code, ...)



Translation  
(by a human,  
a computer program,  
or an LLM)



Formal  
language  
(logic)



Ask  
questions  
(symbolic  
reasoning)

Results are accurate,  
sound, and transparent if  
assumptions are correct

# SAT/SMT solvers

- Propositional satisfiability problem (**SAT**) solvers work with propositional logic formulas to determine if there exists an assignment of true/false values to variables that makes the entire formula true (satisfiable).
- For example, consider the formula:

$$(x \vee y) \wedge (\neg x \vee z) \wedge (\neg y \vee \neg z)$$

- A SAT solver would determine if there's any way to assign true/false to x, y, and z to make this true.

# SAT/SMT solvers

- Satisfiability modulo theories (**SMT**) solvers extend SAT solving to handle richer logical theories beyond Boolean logic. They can reason about integer and real arithmetic, arrays and data structures, strings, etc.

- For example, an SMT solver can handle formulas like:

$(x > y) \wedge (y > 0) \wedge (x + y < 10)$  where **x** and **y** are integers

- Most modern SMT solvers actually incorporate SAT solvers as their core engine.



# Automated reasoning tools

**CPROVER** – Perform Bounded Model Checking and SatAbs verification on C programs. Detect buffer overflows, pointer safety, and assertions.

**Dafny** – Verification-aware programming language, blending automated reasoning with familiar programming idioms.

**Kani Rust Verifier** – Open source verification checking for Rust programs using a SAT solver.

**Zelkova** – SMT solver used at Amazon for S3 Block Public Access, IAM Access Analyzer, VPC Reachability Analyzer, and more.



# Where do you go from here?

- Continue to learn
- Try an AI-powered coding assistant
- Start to learn about automated reasoning and formal verification



## Do more with AWS Builder ID

Access 600+ free courses, connect with fellow AWS builders in the community, and build with tools like Amazon Q Developer—all with your single Builder ID





# Thank you!

## Questions?

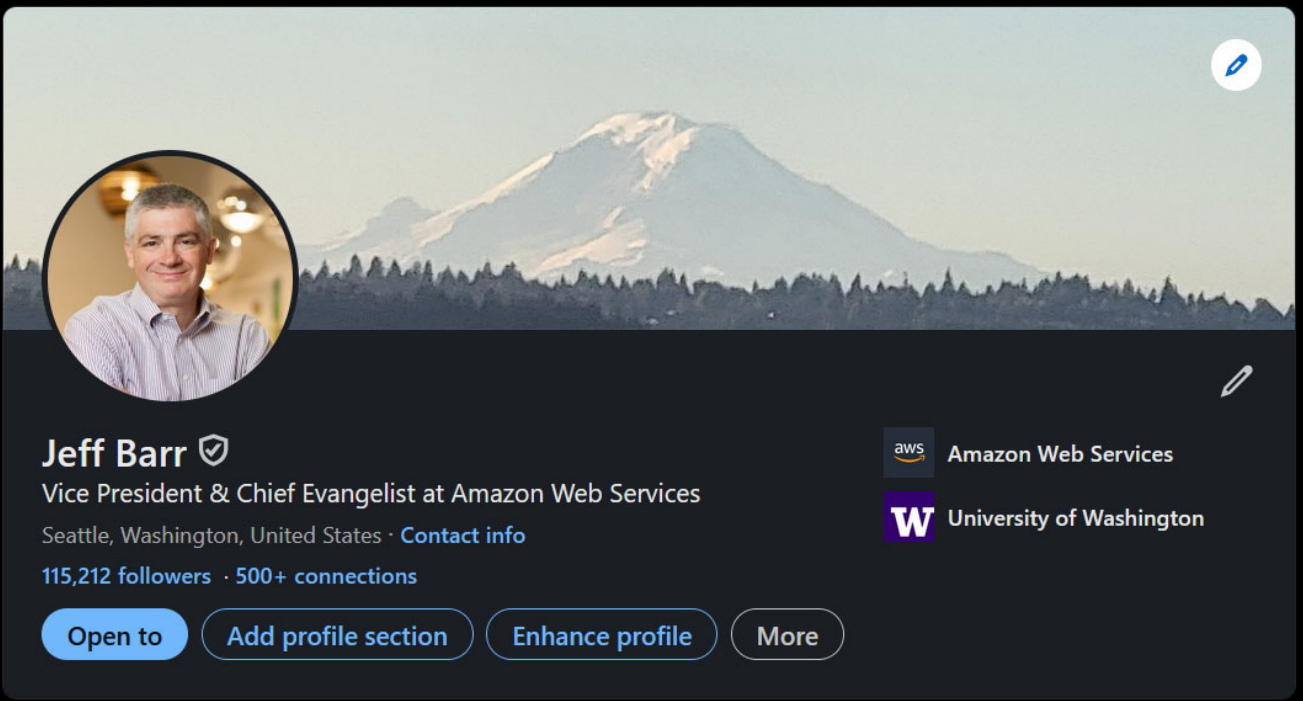
**Jeff Barr**

@jeffbarr

jbarr@amazon.com



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

A screenshot of a LinkedIn profile for Jeff Barr. The profile card has a dark background. At the top is a banner image of a snow-capped mountain. Below the banner is a circular profile picture of Jeff Barr, a man with grey hair wearing a light blue button-down shirt. To the right of the profile picture is a small blue icon of a pencil. Below the profile picture, the name 'Jeff Barr' is displayed with a blue verification checkmark. Underneath the name is the title 'Vice President & Chief Evangelist at Amazon Web Services' and the location 'Seattle, Washington, United States' followed by a link to 'Contact info'. Below this, it shows '115,212 followers · 500+ connections'. At the bottom of the profile card are four buttons: 'Open to', 'Add profile section', 'Enhance profile', and 'More'. To the right of the profile information, there are two logos: the AWS logo followed by 'Amazon Web Services' and the University of Washington logo followed by 'University of Washington'. A small blue icon of a pencil is also visible to the right of the logos.