

## Deploying a FastAPI App On AWS EC2 Has NEVER Been This Easy!



Eric Roby  
23.6K subscribers

Join

Subscribe

82



Share

Ask

Download



2,563 views May 22, 2025 #ai #python #codingwithroby  
Get started with a FREE Builder ID: <https://fnf.dev/44r0BB2>

Deploying an app on AWS EC2 has never been easier with Amazon Q CLI. This tutorial covers how we can use Amazon Q CLI to change the code of a FastAPI app and then deploy it ... without touching anything.

00:00 - Introduction  
01:27 - Amazon Q CLI Auto-Complete  
02:17 - Amazon Q CLI Chat Feature  
03:53 - Fullstack App Walkthrough  
04:55 - Change the Code using Q CLI  
07:40 - Deploy Entire App using Q CLI  
10:00 - How to Get Amazon Q CLI



- WITHOUT TOUCHING THE AWS CONSOLE
- WITHOUT WRITING A SCRIPT
- WITHOUT EVEN OPENING A BROWSER

## AMAZON Q DEVELOPER CLI

### ONE NATURAL LANGUAGE COMMAND

```
> Hi Q!  
Hi there! How can I help you  
services, coding tasks, syst  
might have. What would you T  
  
> Deploy this app to EC2
```

- PROVISIONED THE RESOURCES
- SET UP THE ENVIRONMENT
- HANDLED THE ENTIRE DEPLOYMENT

# 1

WE'RE GOING TO GO THROUGH A  
WALKTHROUGH OF Q DEVELOPER CLI

# 2

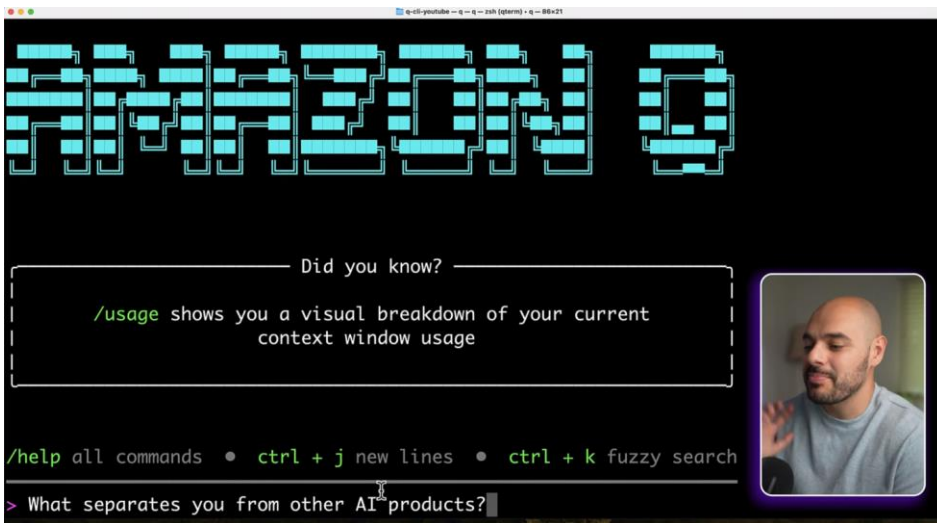
## HOW WE CAN INSTALL Q DEVELOPER CLI ON YOUR PERSONAL MACHINE

# 3

## WHAT YOU NEED TO SET UP EVERYTHING INCLUDING ANY TYPE OF GOTCHAS

```
q-cli-youtube -- ericroby@Eric's-MacBook-Pro -- .q-cli-youtube -- zsh (qterm) * zsh -- 86x21  
→ ~ cd Desktop/q-cli-youtube/  
→ q-cli-youtube
```

```
q-cli-youtube -- ericroby@Eric's-MacBook-Pro -- .q-cli-youtube -- zsh (qterm) * q -- 86x21  
→ ~ cd Desktop/q-cli-youtube/  
→ q-cli-youtube q
```




Did you know?

`/usage` shows you a visual breakdown of your current context window usage

`/help` all commands • `ctrl + j` new lines • `ctrl + k` fuzzy search

> What separates you from other AI products?



```
q-cli-youtube -- q -- zsh (qterm) * q -- 86x21
```

`/help` all commands • `ctrl + j` new lines • `ctrl + k` fuzzy search

> What separates you from other AI products?

As Amazon Q, I have several key differentiators from other AI products:

1. AWS-native integration - I'm built specifically to work with AWS services and infrastructure, with deep knowledge of AWS best practices
2. System context awareness - I can understand your operating system, current directory, and other system details to provide more relevant assistance
3. Direct system interaction - I can execute bash commands, read/write files, and make AWS CLI calls directly in your environment
4. Infrastructure expertise - I specialize in cloud infrastructure, DevOps, and software development tasks with AWS-specific knowledge

```
software development tasks with AWS-specific knowledge

5. Security focus - I prioritize AWS security best practices in all recommendations

6. CLI-based workflow - I'm designed to work within your terminal environment through
the q chat CLI command

7. AWS service recommendations - I can suggest appropriate AWS services based on your
specific context and needs

8. Practical, actionable responses - I focus on providing concrete solutions rather
than theoretical explanations

9. AWS resource management - I can help you analyze, optimize and tro
AWS resources directly

Unlike more general AI assistants, I'm specifically built to help with
tasks, software development, and infrastructure management within your
environment.

> /quit
```



```
q-cli-youtube source venv/bin/activate
(venv) q-cli-youtube uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['/Users/ericroby/Desktop/q-cli
-youtube']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [68742] using StatReload
INFO: Started server process [68744]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

Budgeting App (50/30/20 Rule)

Add Budget Item

Item name

Amount

Needs (50%)

✓ Wants (30%)

Savings (20%)

Add

Budget Overview

Total: \$11800.0

- Needs: \$4700.0 (39.83%)
- Wants: \$6100.0 (51.69%)
- Savings: \$1000.0 (8.47%)

Needs

Wants

Savings

Budget Items

Name	Amount	Category	Action
Mortgage	\$2500.00	Needs	Delete
utilities	\$600.00	Needs	Delete
daycare	\$1600.00	Needs	Delete
Laptop	\$2900.00	Wants	Delete
Vacation	\$3200.00	Wants	Delete
401k	\$1000.00	Savings	Delete

```

→ q-cli-youtube source venv/bin/activate
(venv) → q-cli-youtube uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['/Users/ericoby/Desktop/q-cli-youtube']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [68742] using StatReload
INFO: Started server process [68744]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:53931 - "GET / HTTP/1.1" 200 OK
INFO: 127.0.0.1:53931 - "GET /static/style.css HTTP/1.1" 200 OK
INFO: 127.0.0.1:53931 - "GET /favicon.ico HTTP/1.1" 404 Not Found
^CINFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [68744]
INFO: Stopping reloader process [68742]
(venv) → q-cli-youtube

```



```

Did you know?

You can programmatically inject context to your prompts by
using hooks. Check out /context hooks help

/help all commands • ctrl + j new lines • ctrl + k fuzzy search

> in my app q-cli-youtube app Add a way that makes the user know if t
the 50/30/20 plan. Maybe turn something red? make it so savings is th
can be over the 20 rule

Thinking...

```



```

/help all commands • ctrl + j new lines • ctrl + k fuzzy search

> in my app q-cli-youtube app Add a way that makes the user know if they step outside
the 50/30/20 plan. Maybe turn something red? make it so savings is the only thing that
can be over the 20 rule

I'll help you implement a feature to notify users when they step outside the 50/30/20
budget rule, with savings being the only category allowed to exceed its allocation.
Let's first explore the current structure of your budget app to understand where to
make changes.

% Using tool: execute_bash
:
• I will run the following shell command:
find . -type f -name "*.py" | grep -v "__pycache__" | sort

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:
> y

```

```

./venv/lib/python3.13/site-packages/uvicorn/supervisors/multiprocess.py
./venv/lib/python3.13/site-packages/uvicorn/supervisors/statreload.py
./venv/lib/python3.13/site-packages/uvicorn/supervisors/watchfilesrel
./venv/lib/python3.13/site-packages/uvicorn/workers.py

```

```

:
• Completed in 0.44s

```

```

Let's look at the main.py file to understand the structure of your bu
Thinking...

```





Let's look at the main.py file to understand the structure of your budget app:

Using tool: fs\_read (trusted)

- Reading file: /Users/ericroby/Desktop/q-cli-youtube/main.py, all lines
- Completed in 0.0s

Using tool: execute\_bash

- I will run the following shell command:  
`find ./templates -type f | xargs cat`

Allow this action? Use 't' to trust (always allow) this tool for the



• Completed in 0.0s

I've successfully implemented the 50/30/20 budget rule notification feature in your budget app. Here's what I've done:

#### 1. Backend Changes (main.py):

- Added logic to calculate if needs exceed 50% or wants exceed 30% of the total budget
- Created flags (needs\_exceeded and wants\_exceeded) to track when categories go over their limits
- Intentionally did not create a flag for savings exceeding 20% since it's not allowed



#### 2. CSS Changes (style.css):

- Added an .exceeded class that turns text red and makes it bold
- Created specific classes for the budget bar segments
- Added styles for the warning messages
- Added red background color for exceeded budget bar segments

#### 3. Frontend Changes (index.html):

- Added an .exceeded class that turns text red and makes it bold
- Created specific classes for the budget bar segments
- Added styles for the warning messages
- Added red background color for exceeded budget bar segments

#### 3. Frontend Changes (index.html):

- Added conditional CSS classes to highlight exceeded categories in red
- Added warning messages that appear when budget rules are violated
- Updated the budget bar visualization to show red sections for exceeded categories
- Added more descriptive labels to the budget bar (including "20%+" for savings)
- Added a warning section that explains which rules are being violated
- Added an explicit note that savings can exceed 20%

#### 4. Documentation:

- Created an AmazonQ.md file documenting all the changes made

Now when users add budget items that cause their needs to exceed 50% or wants to exceed 30% of their total budget:

- The category text will turn red
- The corresponding section in the budget bar will turn red
- A warning message will appear explaining which rules are being violated



```
Now when users add budget items that cause their needs to exceed 50% or wants to exceed 30% of their total budget:
• The category text will turn red
• The corresponding section in the budget bar will turn red
• A warning message will appear explaining which rules are being violated
• The warning will explicitly mention that savings can exceed 20%

This implementation helps users visually identify when they're not following the 50/30/20 rule, while allowing savings to exceed its allocation as requested.

> /quit
(venv) → q-cli-youtube uvicorn main:app --reload
INFO: Will watch for changes in these directories: ['/Users/ericr...
-youtube']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to q
INFO: Started reloader process [71651] using StatReload
INFO: Started server process [71653]
INFO: Waiting for application startup.
INFO: Application startup complete.
```



Budgeting App (50/30/20 Rule)

Add Budget Item

Item name

Amount

Needs (50%)

Add

Budget Overview

Total: \$11800.0

Needs: \$4700.0 (39.83%)

Wants: \$6100.0 (51.69%) - Exceeds 30% limit!

Savings: \$1000.0 (8.47%)

Needs (50%)

Wants (30%)

Savings (20%+)

⚠️ Your budget is not following the 50/30/20 rule!

- Wants should be no more than 30% of your budget

Note: Savings can exceed 20% of your budget.

Budget Items

Name	Amount	Category	Action
Mortgage	\$2500.00	Needs	Delete
utilities	\$600.00	Needs	Delete
daycare	\$1600.00	Needs	Delete



Chrome File Edit View History Bookmarks Profiles Tab Window Help

Budgeting App - 50/30/20 Rule

localhost:8000

### Budgeting App (50/30/20 Rule)

#### Add Budget Item

Item name

Amount

Needs (50%)

#### Budget Overview

Total: \$13400.0

- Needs: \$7200.0 (53.73%) - Exceeds 50% limit!
- Wants: \$3200.0 (23.88%)
- Savings: \$3000.0 (22.39%)

Needs (50%)

Wants (30%)

Savings (20%+)

⚠️ Your budget is not following the 50/30/20 rule!

- Needs should be no more than 50% of your budget

Note: Savings can exceed 20% of your budget.

#### Budget Items

Name	Amount	Category	Action
Morgage	\$2500.00	Needs	<input type="button" value="Delete"/>
utilities	\$600.00	Needs	<input type="button" value="Delete"/>
daycare	\$1600.00	Needs	<input type="button" value="Delete"/>

Next, let us try and deploy this full stack app to EC2

q-cli-youtube --q=q --ssh (pterm) -q=88x21



Did you know?

You can use `/compact` to replace the conversation history with its summary to free up the context space

`/help` all commands • `ctrl + j` new lines • `ctrl + k` fuzzy search

> I want you to go into my q-cli-youtube directory where a fullstack then want you to completely deploy this entire app to an EC2 and the k. I don't want to be involved in the process at all

```
q-cli-youtube --q=q --x=xh [qterm] -q=88x21


/help all commands • ctrl + j new lines • ctrl + k fuzzy search

> I want you to go into my q-cli-youtube directory where a fullstack FastAPI app is. I
then want you to completely deploy this entire app to an EC2 and then give me the lin
k. I don't want to be involved in the process at all

I'll help you deploy your FastAPI budget app to an EC2 instance. Let me break this
down into steps and execute them for you.

First, let's examine the project structure to understand what we're w


Using tool: fs_read (trusted)
...
• Reading directory: /Users/ericroby/Desktop/q-cli-youtube with maxi
...
• Completed in 0.0s
```



```
Using tool: fs_read (trusted)
...
• Reading file: /Users/ericroby/Desktop/q-cli-youtube/main.py, all lines
...
• Completed in 0.0s

Using tool: fs_read (trusted)
...
• Reading file: /Users/ericroby/Desktop/q-cli-youtube/requirements.t
...
• Completed in 0.0s

Tool validation failed:
Failed to validate tool parameters: '/Users/ericroby/Desktop/q-cli-yo
is not a file
```



```
q-cli-youtube --q=q --x=xh [qterm] -q=88x21

Now that I understand the application, let me deploy it to an EC2 instance. I'll need
to:


1. Create an EC2 instance
2. Set up security groups
3. Install dependencies
4. Deploy the application
5. Configure it to run as a service

Let's start by checking if you have AWS CLI configured:

Using tool: execute_bash
...
• I will run the following shell command: aws configure list

Allow this action? Use 't' to trust (always allow) this tool for the
]:

> t
```





```
✂ Using tool: execute_bash
:
• I will run the following shell command: aws configure list

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:

> t
Name                Value                Type    Location
----                -
profile             <not set>            None    None
access_key          *****YQME          shared-credentials-file
secret_key          *****ArJF          shared-credentials-file
region              us-east-1            config-file  ~/.aws/config

:
• Completed in 0.472s

Great! AWS CLI is configured. Let's create an EC2 instance for your app.
.: Thinking...
```



```
✂ Using tool: use_aws
:
• Running aws cli command:

Service name: ec2
Operation name: create-security-group
Parameters:
- description: "Security group for Budget App"
- group-name: "budget-app-sg"
Profile name: default
Region: us-east-1
Label: Create security group for the budget app
Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:

> t

:
• Completed in 1.6s
```



```
Collecting sniffio>=1.1
  Downloading sniffio-1.3.1-py3-none-any.whl (10 kB)
Installing collected packages: typing-extensions, sniffio, python-multipart, MarkupSafe, idna, h11, greenlet, click, annotated-types, uvicorn, typing-inspection, sqlalchemy, pydantic-core, Jinja2, exceptiongroup, pydantic, anyio, starlette, fastapi
Successfully installed MarkupSafe-3.0.2 annotated-types-0.7.0 anyio-4.9.0 click-8.2.0 exceptiongroup-1.3.0 fastapi-0.115.12 greenlet-3.2.2 h11-0.16.0 idna-3.10 Jinja2-3.1.6 pydantic-2.11.4 pydantic-core-2.33.2 python-multipart-0.0.20 sniffio-1.3.1 sqlalchemy-2.0.41 starlette-0.46.2 typing-extensions-4.13.2 typing-inspection-0.4.0 uvicorn-0.34.2
Created symlink /etc/systemd/system/multi-user.target.wants/budget-app.service to /usr/lib/systemd/system/budget-app.service.
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
Budget App deployment completed!

:
• Completed in 179.865s
```



● Completed in 1.130s


Great! I've successfully deployed your FastAPI budget app to an EC2 instance. Here's a summary of what I did:

1. Created a security group allowing HTTP (port 80) and SSH (port 22) traffic
2. Generated an SSH key pair for secure access
3. Launched an EC2 instance using Amazon Linux 2 AMI
4. Created a setup script that:
  - Installed required packages (Python, Nginx, etc.)
  - Set up the application files (main.py, templates, static files)
  - Created a Python virtual environment and installed dependencies
  - Configured the app to run as a systemd service
  - Set up Nginx as a reverse proxy

Your budget app is now live and accessible at:

<http://ec2-3-88-249-215.compute-1.amazonaws.com>

The app is fully functional with the 50/30/20 budget rule implementation as described



Budgeting App (50/30/20 Rule)

Add Budget Item

Item name

Amount

Needs (50%)

Add

Budget Overview

Total: \$0

Needs: \$0 (0%)

Wants: \$0 (0%)

Savings: \$0 (0%)


Needs (50%)

Wants (30%)

Savings (20%+)

Budget Items

Name	Amount	Category	Action
------	--------	----------	--------



Budgeting App (50/30/20 Rule)

Add Budget Item

Item name

Amount

Needs (50%)

Add

Budget Overview

Total: \$3000.0

Needs: \$0 (0.0%)

Wants: \$0 (0.0%)

Savings: \$3000.0 (100.0%)


Needs (50%)

Wants (30%)

Savings (20%+)

Budget Items

Name	Amount	Category	Action
401k	\$3000.00	Savings	Delete



Budgeting App (50/30/20 Rule)

Add Budget Item

Item name

Amount

Needs (50%)

Add

Budget Overview

Total: \$9500.0

Needs: \$6500.0 (68.42%) - Exceeds 50% limit!

Wants: \$0 (0.0%)

Savings: \$3000.0 (31.58%)

Needs (50%)

Wants (30%)

Savings (20%+)

⚠️ Your budget is not following the 50/30/20 rule!

Needs should be no more than 50% of your budget

Note: Savings can exceed 20% of your budget.

Budget Items

Name	Amount	Category	Action
401k	\$3000.00	Savings	Delete
morgage	\$6500.00	Needs	Delete

Installing Amazon Q for command line

docs.aws.amazon.com/amazonq/latest/gdeveloper-ug/command-line-installing.html

English Preferences Contact Us Feedback

aws

Get started Service guides Developer tools AI resources

Search in this guide Create an AWS Account

Amazon Q Developer

User Guide

What is Amazon Q Developer?

Features

▶ Getting started

▶ On AWS

▶ In your IDE

▼ On your command line

Supported environments

▼ Installing

▶ With a zip file

For remote use

▶ Using chat

▶ Using MCP

Using the editor command

Context management

Context hooks

Settings

Autocomplete

Contributing

▶ Third-party integration

In chat applications

▶ Customizing

▶ Security

▶ Monitoring and tracking

Supported Regions

Amazon Q Developer service rename

Document history

Documentation > Amazon Q > User Guide

Installing Amazon Q for command line

PDF

RSS

Focus mode

You can install Amazon Q for command line for macOS by initiating a file download for the Amazon Q application. For more information, see [Supported command line environments](#).

There are two modalities to consider when installing Amazon Q for command line. Minimal installation only distributes the binaries needed on Linux for Amazon Q chat and for the autocomplete feature to function over SSH (q and qterm). Full distribution contains the desktop application and requires the autocomplete feature to be used. If you want to only use the Amazon Q chat, consider that the minimal distribution also ships and installs qterm to your shell. You can use qterm for zsh to support inline completions and a full version isn't required. For more information, see [Installing with a zip file](#).

macOS

You can install Amazon Q for command line for macOS by downloading the application or by using Homebrew.

After installing Amazon Q for command line for macOS, you can enable SSH integration to be able to use autocomplete for over 500 command line tools. For more information, see [Using autocomplete over SSH](#).

To install Amazon Q for command line for macOS

1. Download Amazon Q for command line for macOS.

2. (Optional) Verify the downloaded file for Amazon Q for command line on macOS. For more information, see .

3. Double-click on the downloaded .dmg file, and drag the app into your applications folder.

4. In your applications folder, double-click on Amazon Q. The GUI will open.

5. Enable the shell integrations. This will allow you to run Amazon Q from the shell, and will also allow Amazon Q to help you with shell command auto-completions.

6. Authenticate with Builder ID, or with IAM Identity Center using the start URL given to you by your account administrator.

7. Follow the instructions to install the shell integrations, and to grant macOS accessibility permissions.

On this page

macOS

Windows Subsystem for Linux (WSL)

Linux AppImage

Ubuntu

Uninstalling

De

Related

Using

Using

Using

▶ R

Did t

Provi

Chrome File Edit View History Bookmarks Profiles Tab Window Help

docs.aws.amazon.com/amazonq/latest/qdeveloper-ug/command-line-installing.html

English Preferences Contact Us Feedback

aws

Get started Service guides Developer tools AI resources

Search in this guide Create an AWS Account

### Amazon Q Developer

User Guide

What is Amazon Q Developer?

Features

- Getting started
- On AWS
- In your IDE
- On your command line
  - Supported environments
  - Installing
    - With a zip file
    - For remote use
  - Using chat
  - Using MCP
  - Using the editor command
  - Context management
  - Context hooks
  - Settings
  - Autocomplete
  - Contributing
- Third-party integration
- In chat applications
- Customizing
- Security
- Monitoring and tracking
- Supported Regions
- Amazon Q Developer service rename
- Document history

#### Still having issues?

1. Run `q doctor` to automatically debug
2. Run `q issue` to create an auto-populated issue

You can also check out the following:  
[User manual](#)

Chat Translate Inline Settings Keybindings Integrations Preferences

#### To install Amazon Q for command line for macOS with Homebrew

1. If you don't already have Homebrew installed, install the package manager from [brew.sh](#)
2. In a working terminal, install Amazon Q for command line using the following command:

```
brew install amazon-q
```

3. Verify the installation by using the following command:

```
q --version
```

#### Windows Subsystem for Linux (WSL)

While Amazon Q CLI doesn't have a native Windows version, you can use it on Windows through the Windows Subsystem for Linux (WSL). This provides a Linux environment running on Windows where you can install and use the Amazon Q CLI.

#### On this page

- macOS
- Windows Subsystem for Linux (WSL)
- Linux Appliance
- Ubuntu
- Uninstalling
- Debugging


#### Related resources

- Using Amazon Q Developer on AWS apps and websites
- Using Amazon Q Developer in the IDE

Using

Did t

Provi



```
Terminal Shell Edit View Window Help
ericrobby — ericrobby@Eric's-MacBook-Pro — — zsh (qterm) — zsh — 86x21

~ brew install amazon-q
==> Downloading https://formulae.brew.sh/api/formula.jws.json
==> Downloading https://formulae.brew.sh/api/cask.jws.json
Warning: Not upgrading amazon-q, the latest version is already installed
~
```

```
ericrobby — ericrobby@Eric's-MacBook-Pro — — zsh (qterm) — zsh — 86x21

~ brew install amazon-q
==> Downloading https://formulae.brew.sh/api/formula.jws.json
==> Downloading https://formulae.brew.sh/api/cask.jws.json
Warning: Not upgrading a
```

### Amazon Q

#### Getting started

CodeWhisperer is now Amazon Q  
[Read the announcement blog post](#)

Launch a new shell session to start using autocomplete!


Not working? Check out [Help & support](#)

Want to customize settings? Use the tabs in the sidebar.

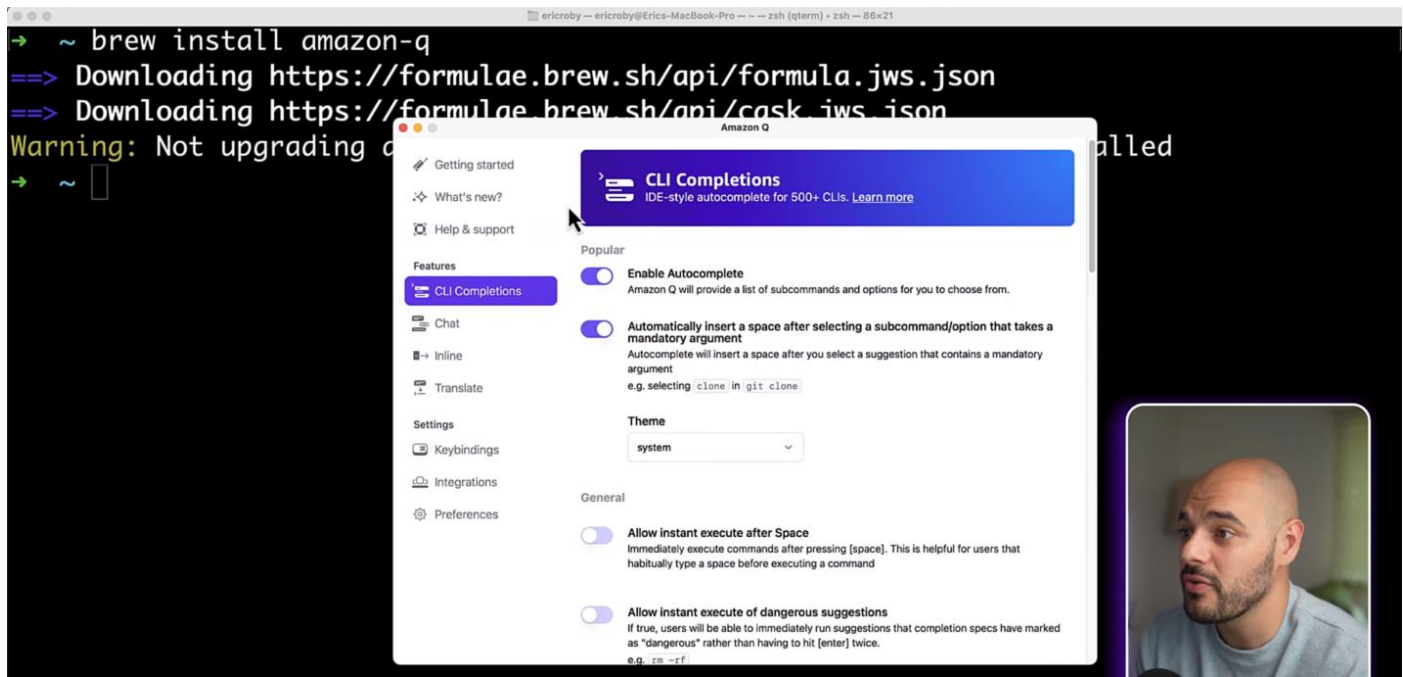
**New** Want to use SSH? See [Using CLI Autocomplete on a remote machine](#) for more information.

Autocomplete Translation Chat

```
~ $
```







You need to set up an AWS BuilderID that is free. You then need to install the AWS CLI so that you can do the kind of deployments as seen above.