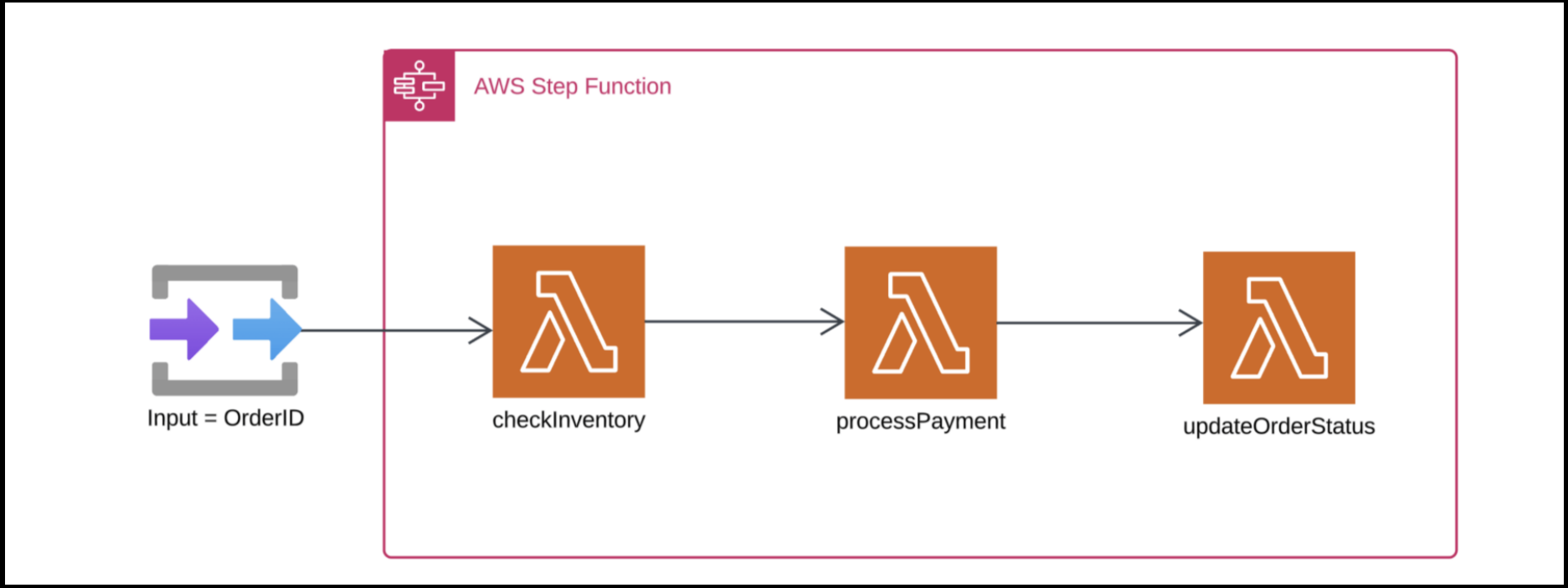


SERVERLESS ORDER FLOW WITH AWS LAMBDA & STEP FUNCTIONS

🚀 🛒 📦 ⚙️

BY KHUSHI NANDWANI

📌 INTRODUCTION

In the modern cloud ecosystem, serverless architecture empowers developers to build and deploy applications without managing infrastructure. This hands-on project walks you through creating a simplified serverless order-processing system using AWS Lambda and Step Functions. You'll build three Lambda functions that simulate inventory checking, payment processing, and order status updating. These are orchestrated via AWS Step Functions to create a seamless order workflow.

🧰 PREREQUISITES

- AWS Account with administrative access
- Basic knowledge of AWS Lambda & IAM
- Familiarity with JavaScript (Node.js)
- Understanding of JSON structure and API workflows

📚 PROJECT OVERVIEW

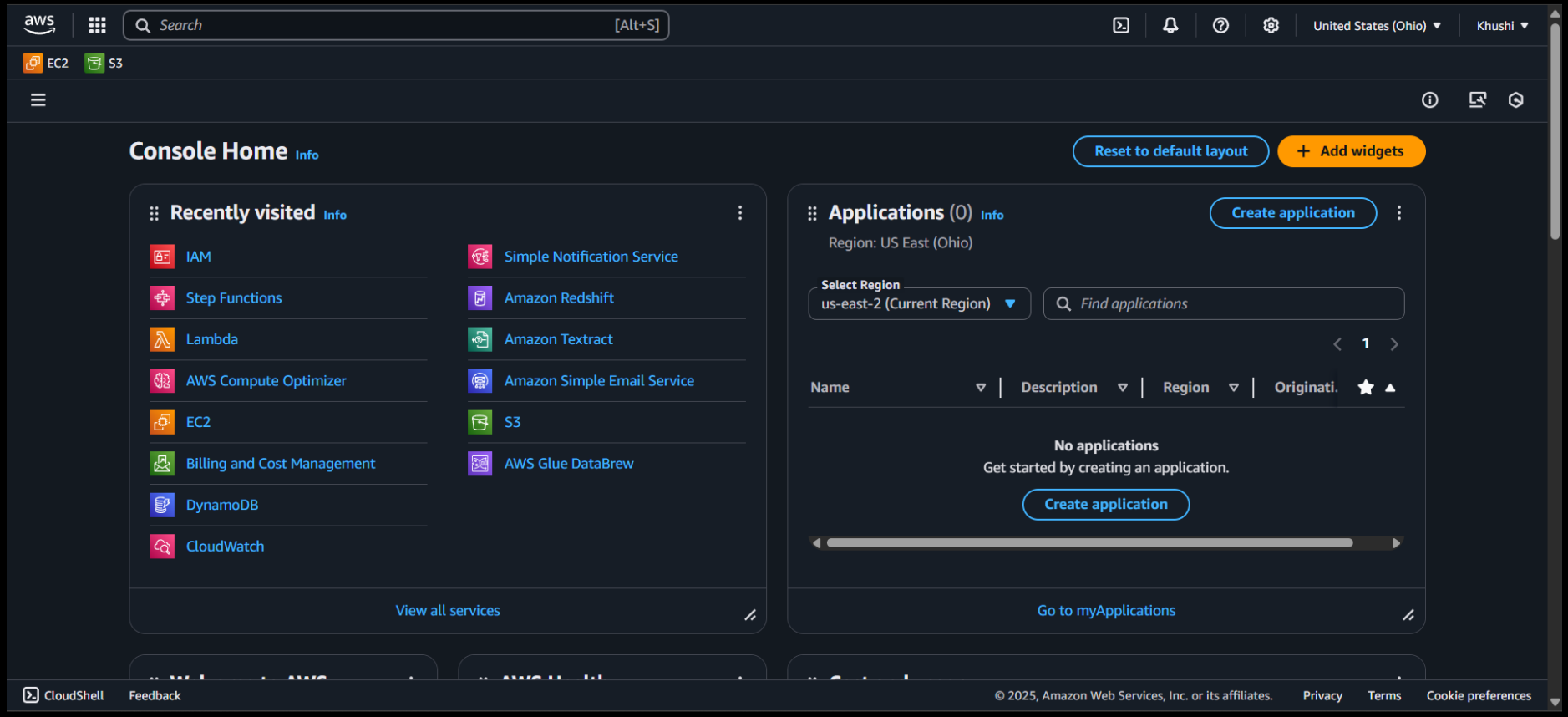
This project implements an automated serverless order flow using AWS services. You will:

- Create three Lambda functions:
  - checkInventory
  - processPayment
  - updateOrderStatus
- Use AWS Step Functions to orchestrate these tasks in sequence.
- Execute the state machine with sample input to simulate the end-to-end process.
- Visualize execution and debug via graphical and tabular representations.
- Clean up resources to avoid unnecessary billing.

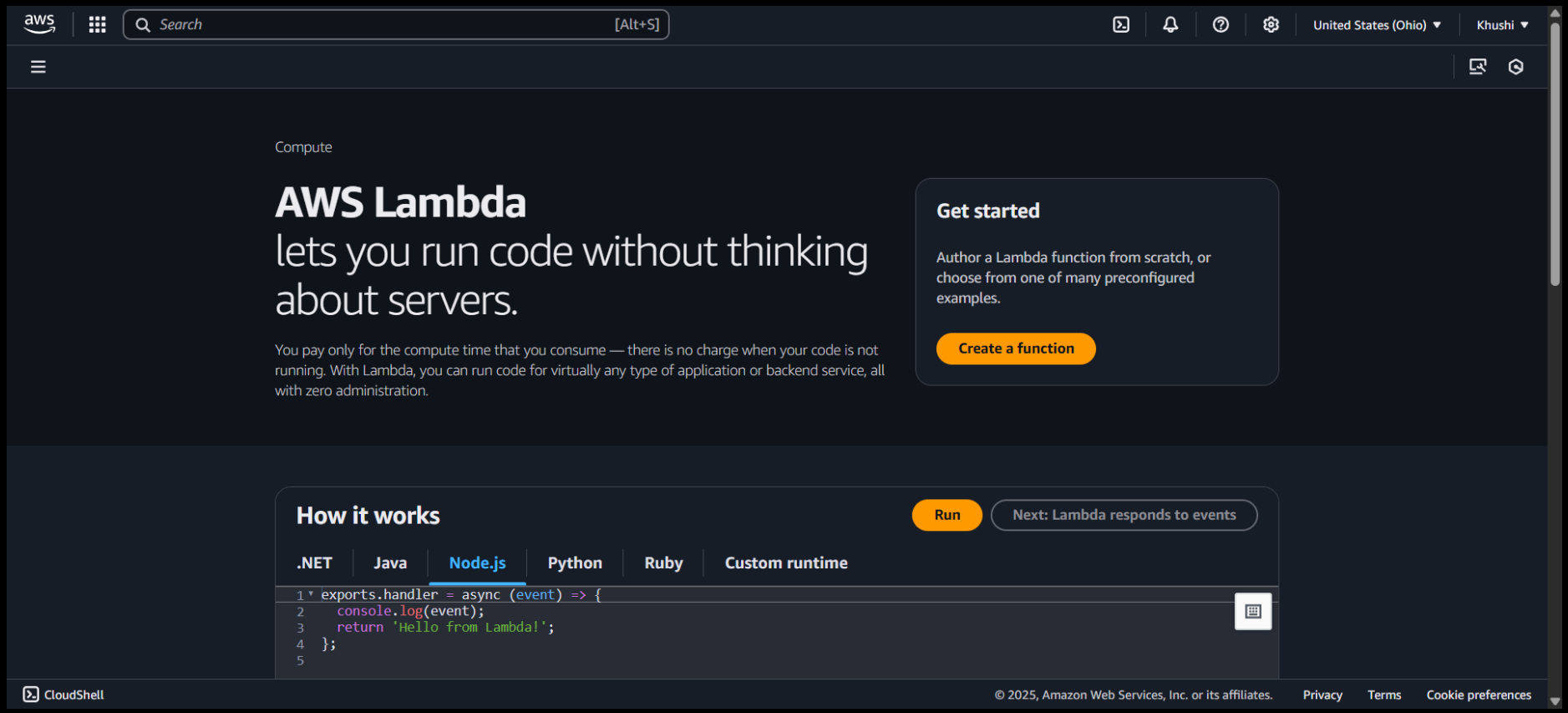
🧭 STEP-BY-STEP INSTRUCTIONS

I . CREATE LAMBDA FUNCTION: CHECKINVENTORY 🔍

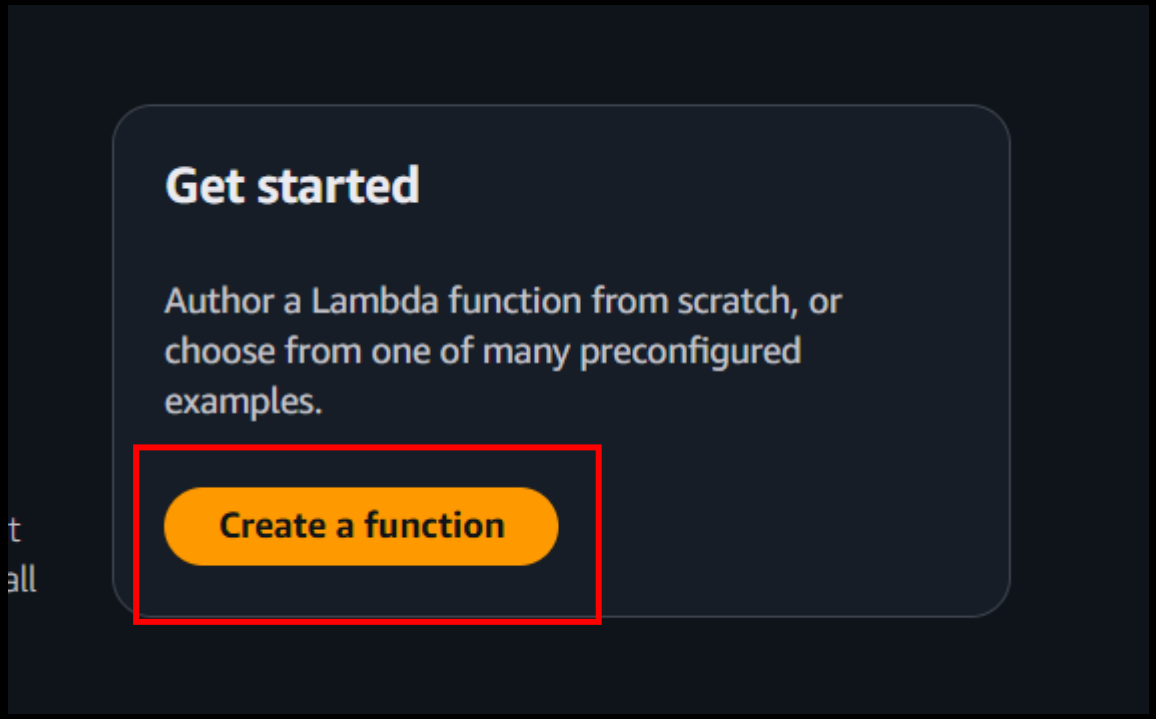
1. Log in to your AWS Console



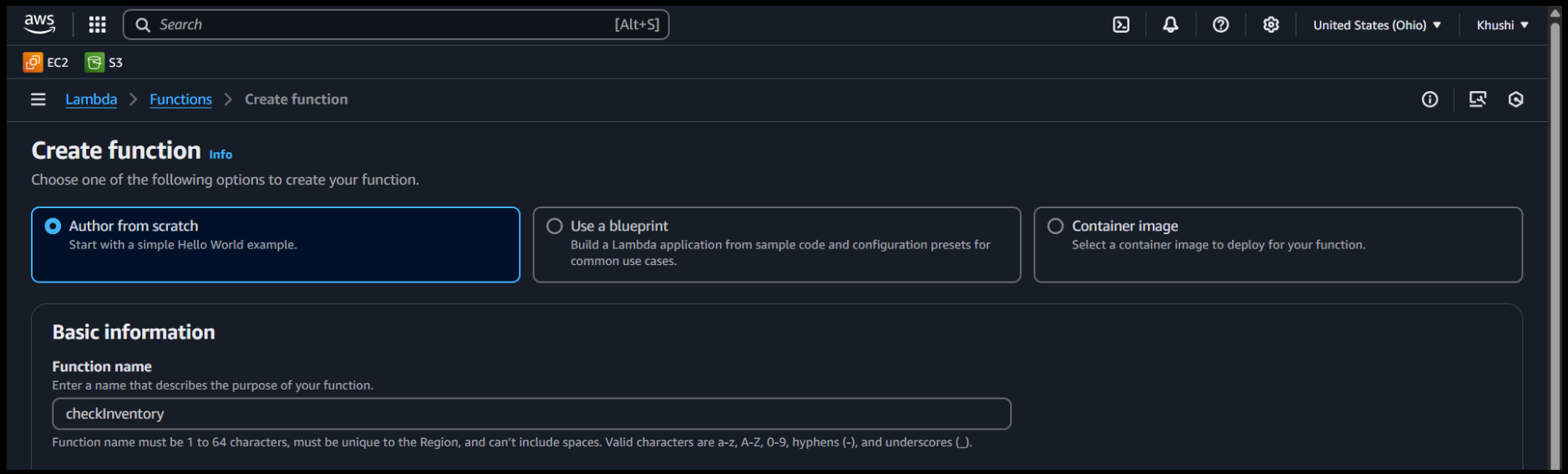
2. Navigate to AWS Lambda Service



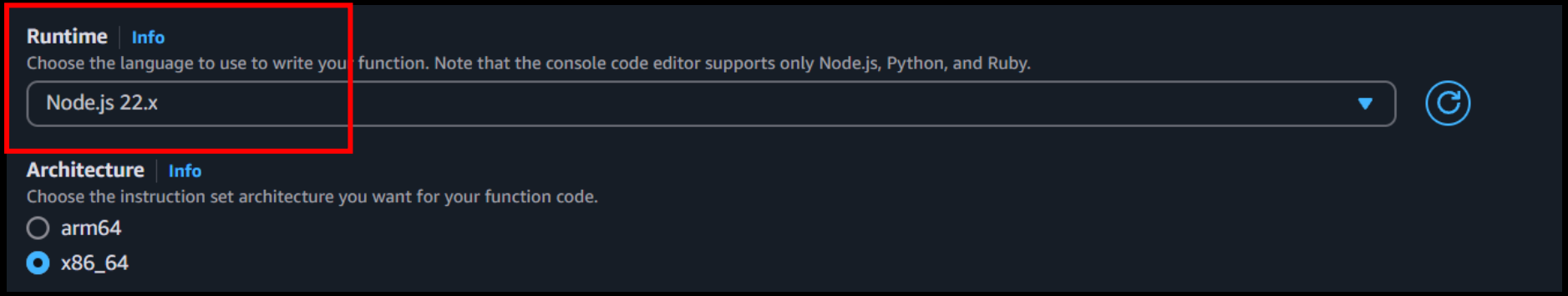
3. Click on "Create a Function"



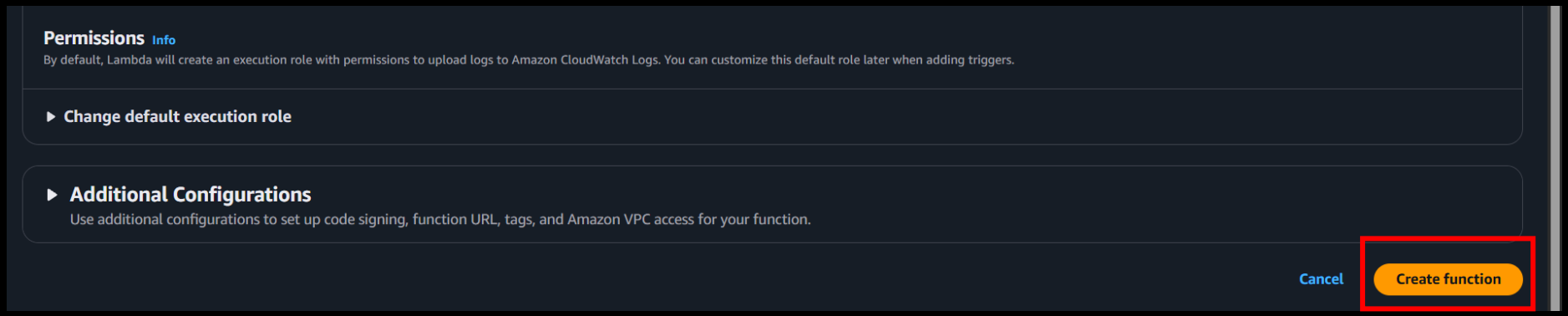
4. Name it **checkInventory**

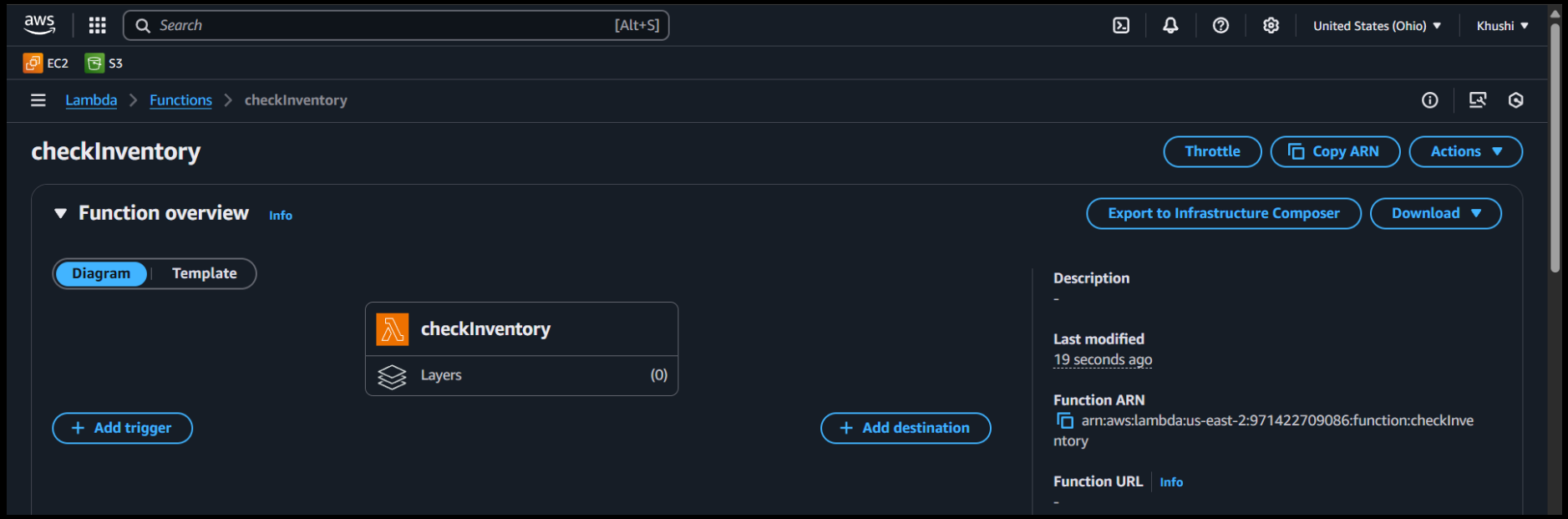


5. Select runtime: Node.js 20.x

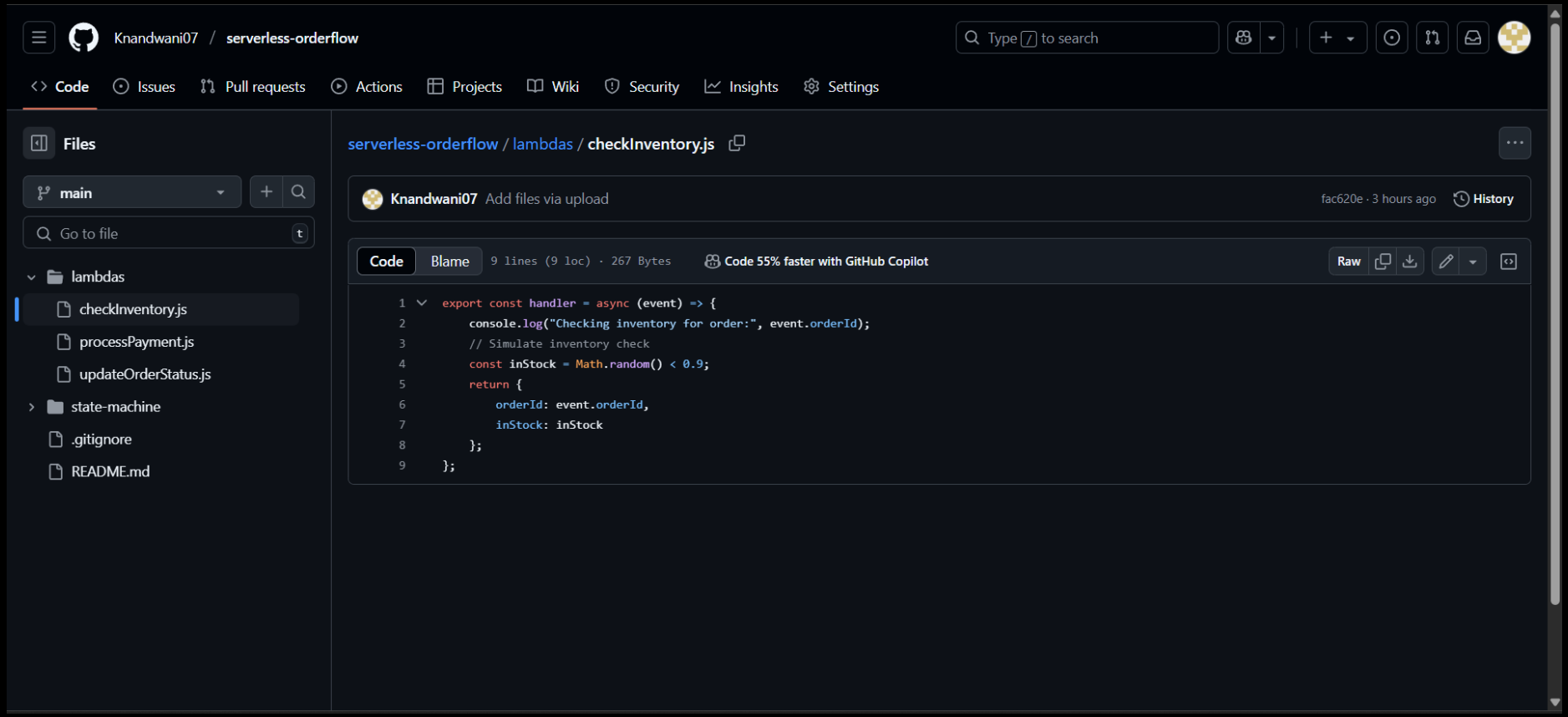


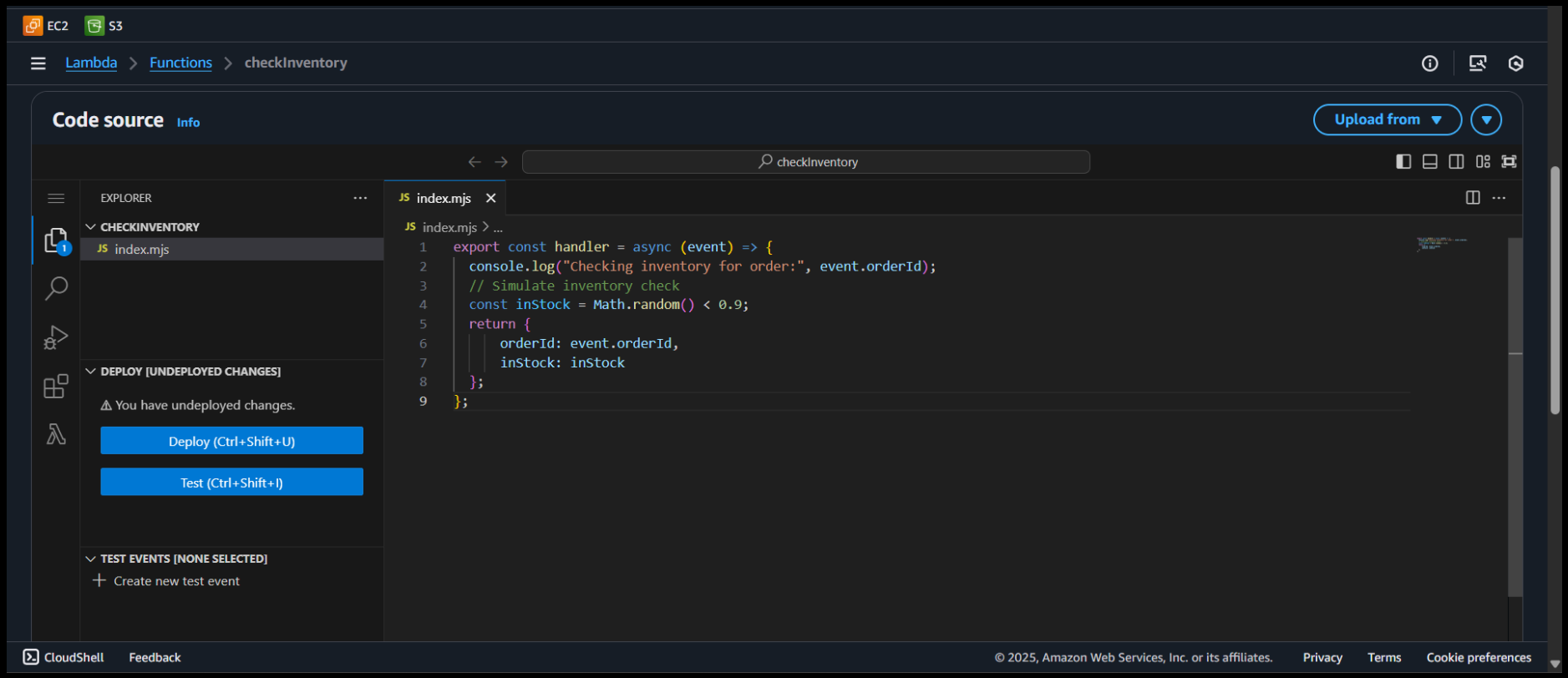
6. Click “Create Function”



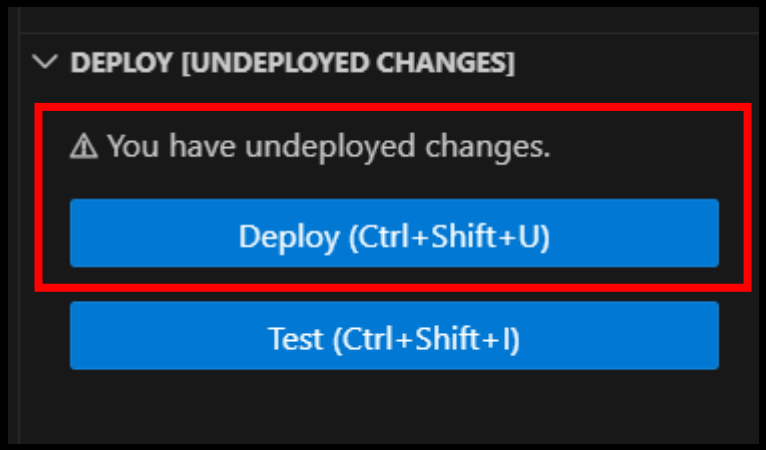


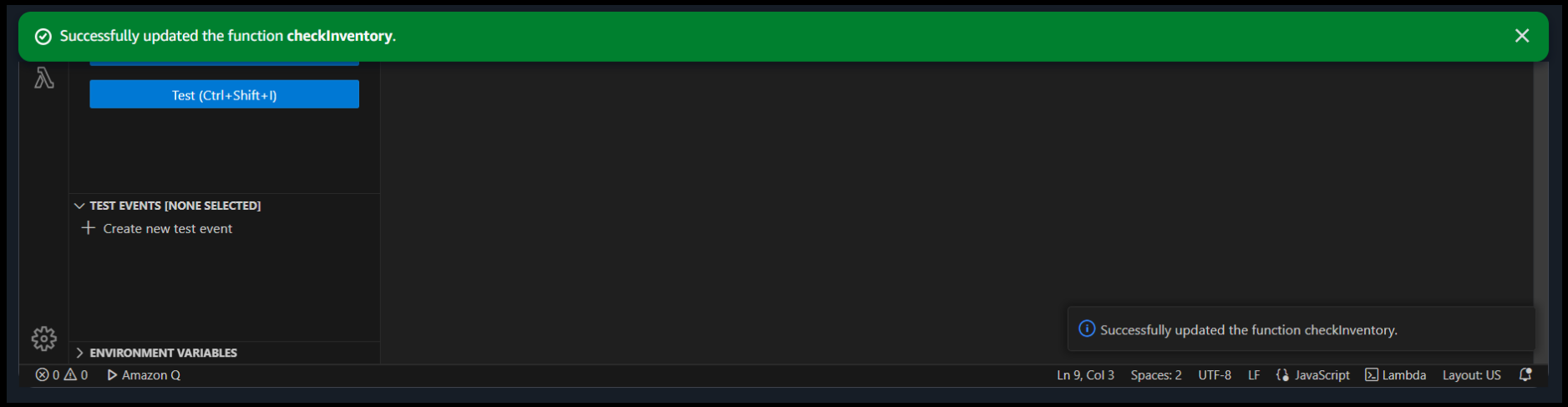
7. In the code editor, paste the checkInventory Lambda code




EC2
S3
Lambda > Functions > checkInventory
Code source Info
Upload from
checkInventory
EXPLORER
CHECKINVENTORY
index.mjs
DEPLOY [UNDEPLOYED CHANGES]
You have undeployed changes.
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
export const handler = async (event) => {
console.log("Checking inventory for order:", event.orderId);
// Simulate inventory check
const inStock = Math.random() < 0.9;
return {
orderId: event.orderId,
inStock: inStock
};
};
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

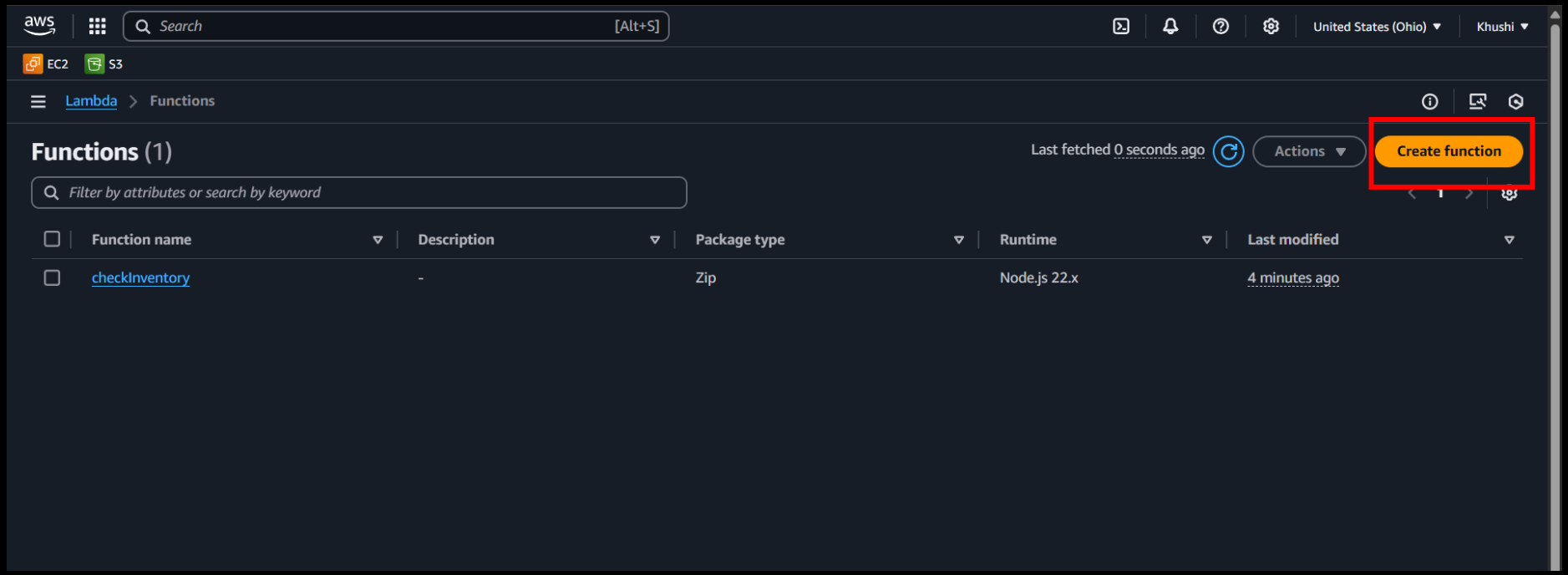
8. Click on “Deploy”





II . CREATE LAMBDA FUNCTION: PROCESSPAYMENT 💳

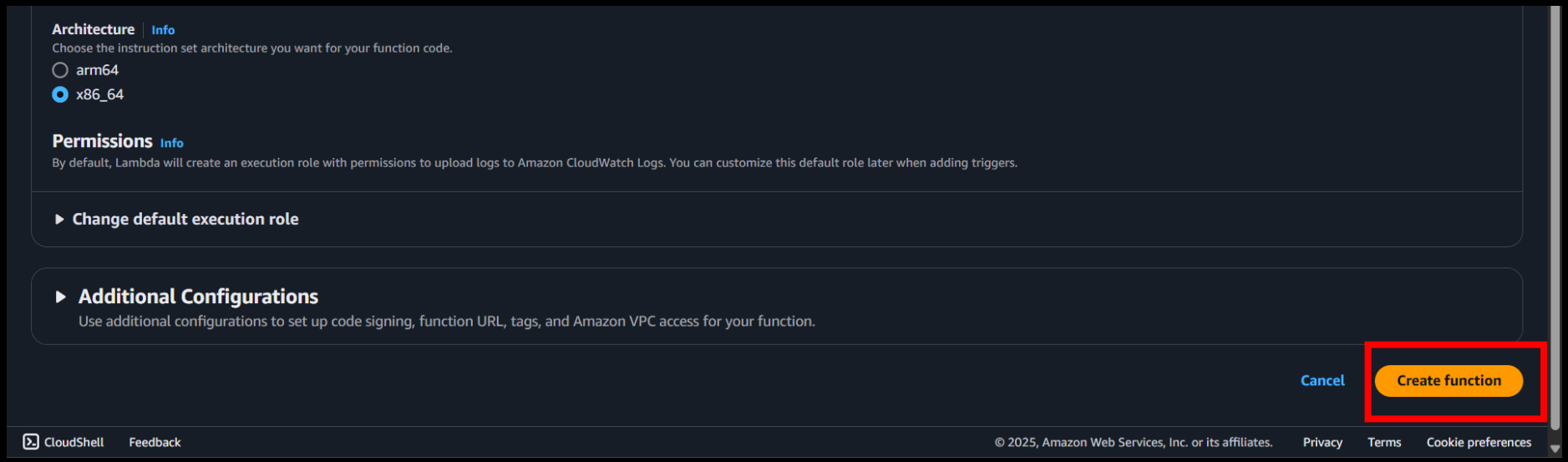
1. Go back to Functions and click "Create Function"

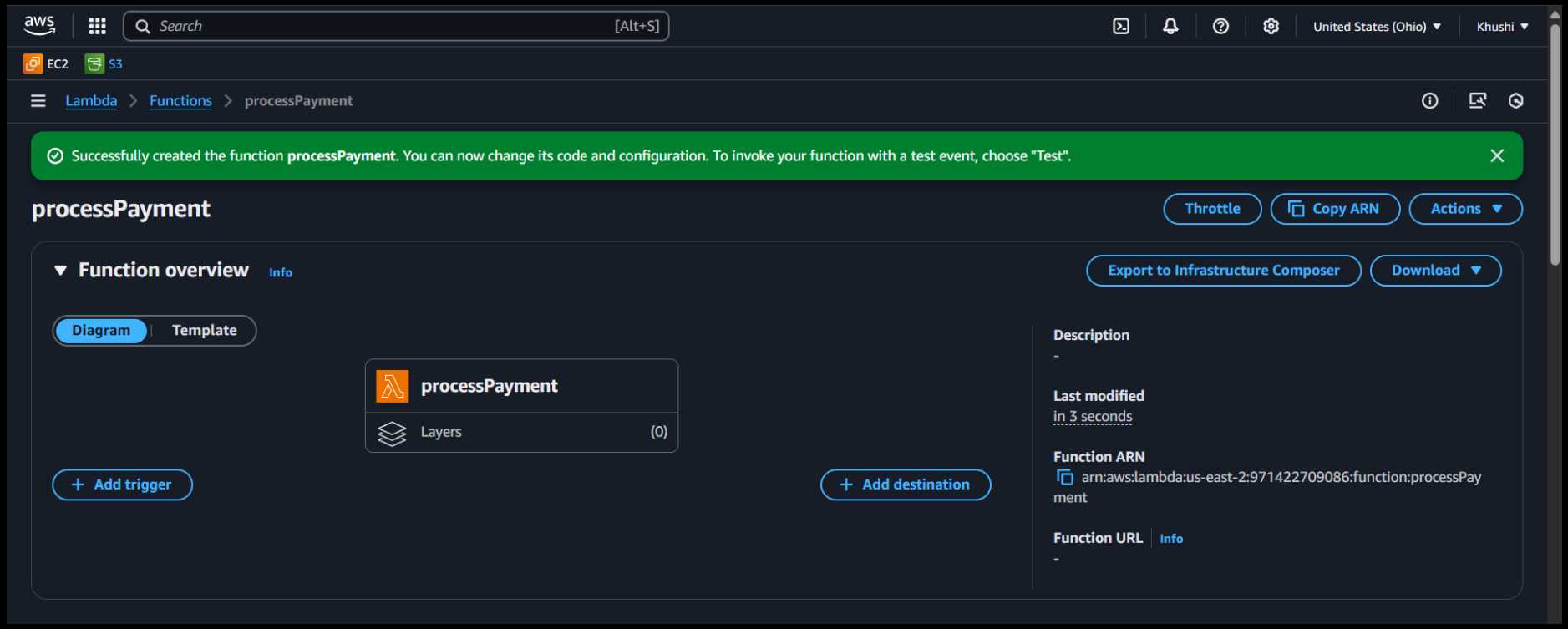


2. Name the function **processPayment**

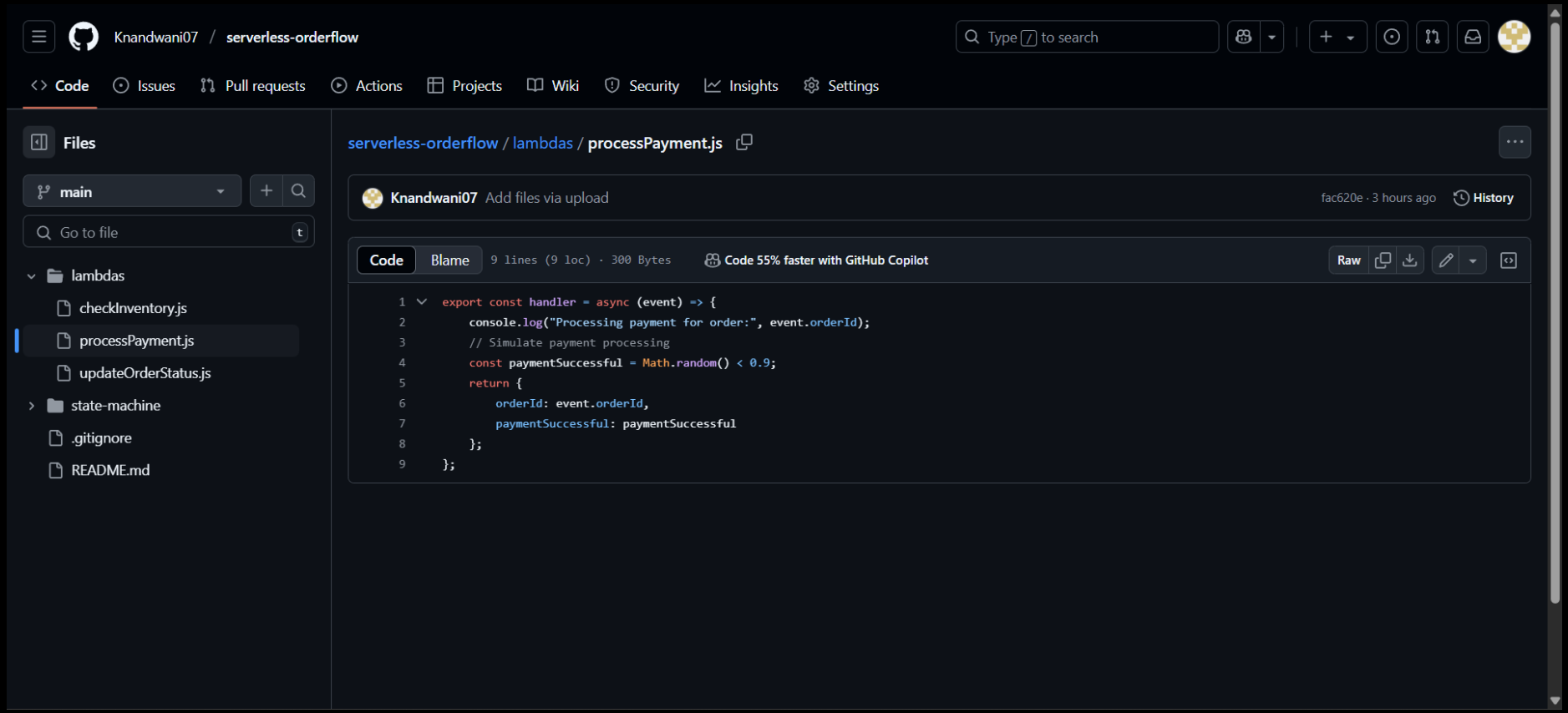
3. Runtime: Node.js 20.x (default)

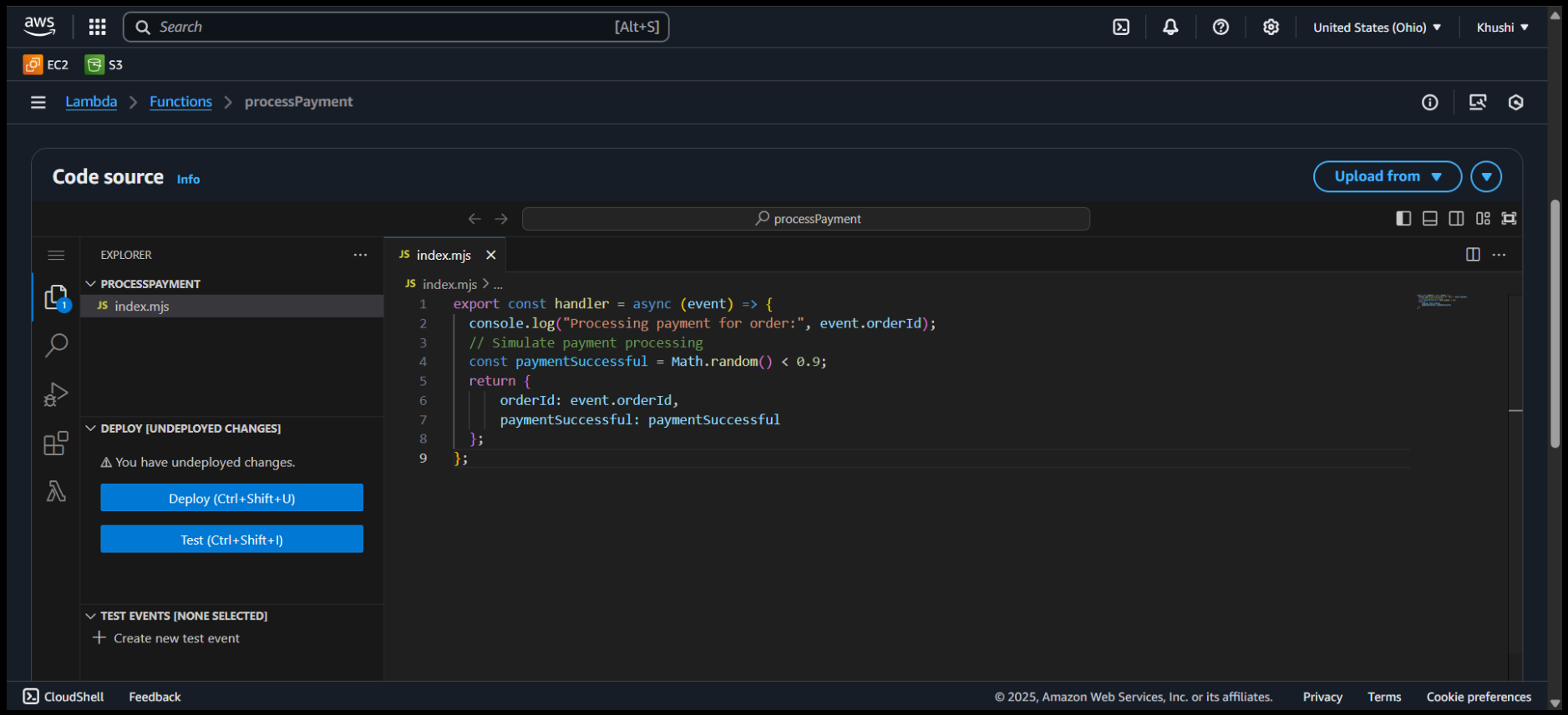
4. Click “Create Function”



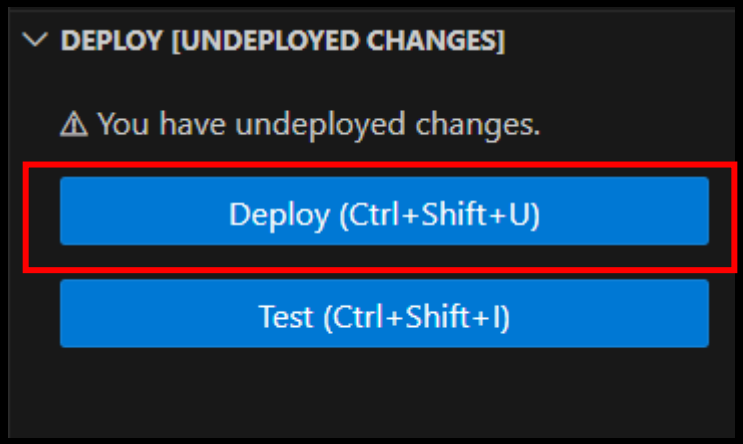

aws
Search
[Alt+S]
United States (Ohio)
Khushi
EC2
S3
Lambda
Functions
processPayment
Successfully created the function processPayment. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
processPayment
Throttle
Copy ARN
Actions
Function overview
Info
Export to Infrastructure Composer
Download
Diagram
Template
processPayment
Layers
(0)
Add trigger
Add destination
Description
-
Last modified
in 3 seconds
Function ARN
arn:aws:lambda:us-east-2:971422709086:function:processPayment
Function URL
Info
-

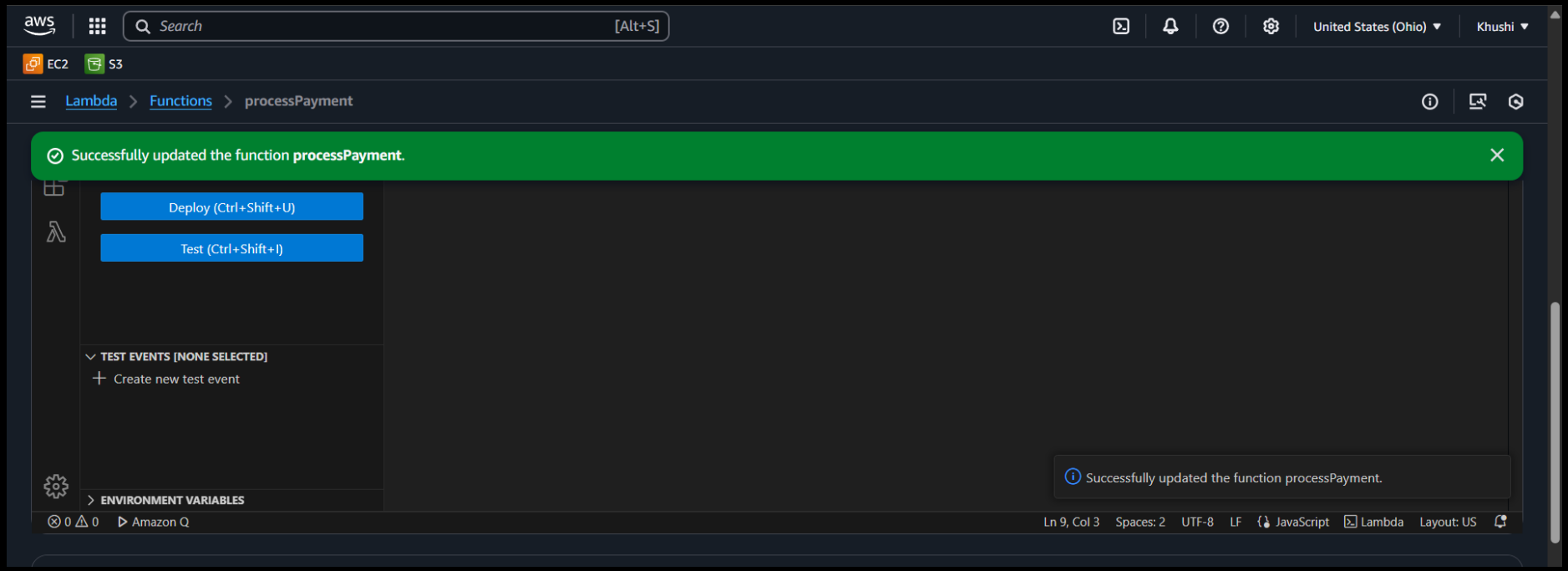
5. Paste the code in the editor




aws
Search
[Alt+S]
United States (Ohio)
Khushi
EC2
S3
Lambda
Functions
processPayment
Code source
Info
Upload from
processPayment
EXPLORER
PROCESSPAYMENT
index.mjs
DEPLOY [UNDEPLOYED CHANGES]
You have undeployed changes.
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
export const handler = async (event) => {
console.log("Processing payment for order:", event.orderId);
// Simulate payment processing
const paymentSuccessful = Math.random() < 0.9;
return {
orderId: event.orderId,
paymentSuccessful: paymentSuccessful
};
};
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

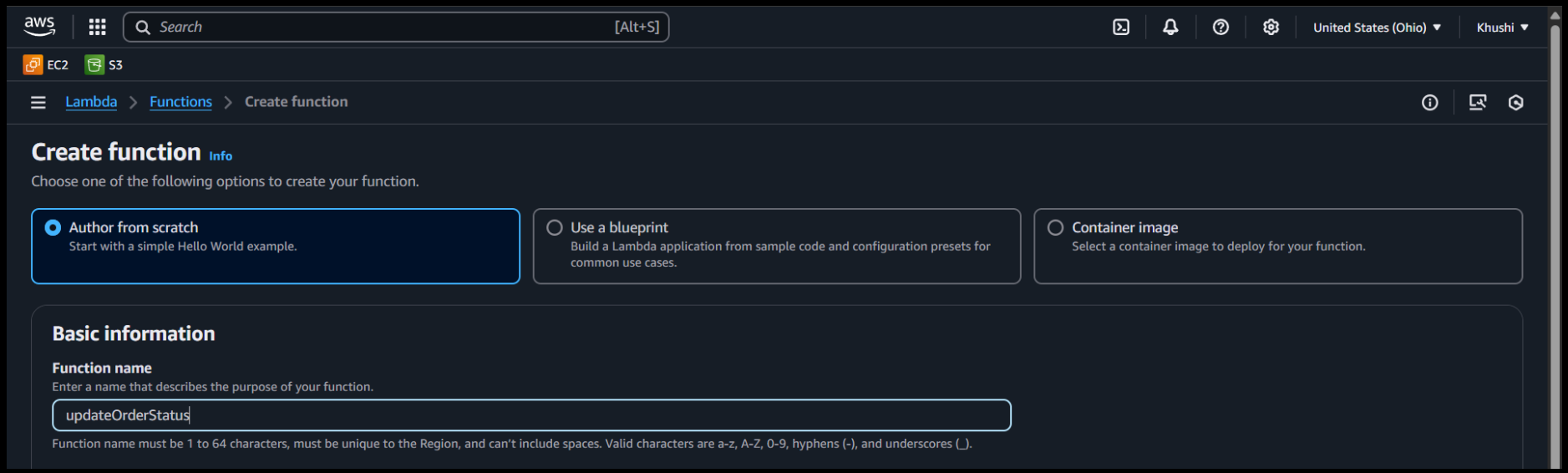
6. Click on “Deploy”



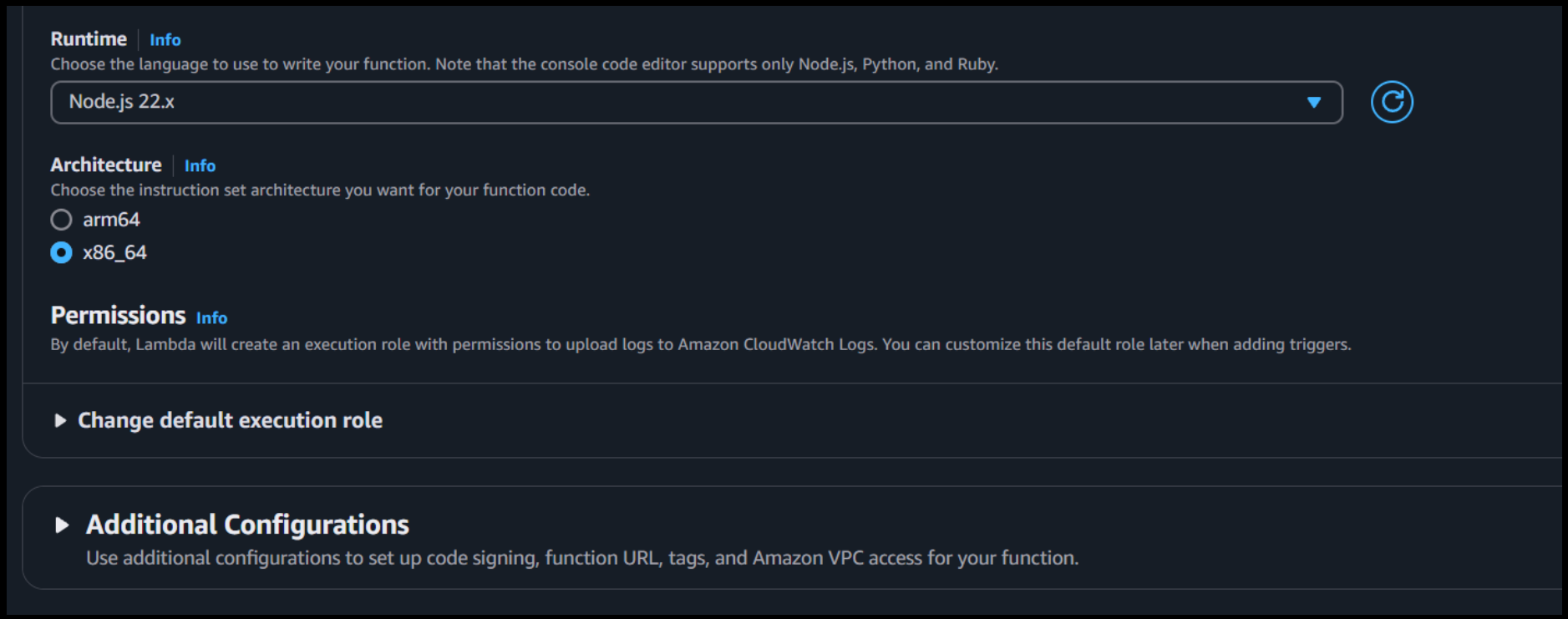

aws
Search
[Alt+S]
United States (Ohio)
Khushi
EC2
S3
Lambda
Functions
processPayment
Successfully updated the function processPayment.
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
ENVIRONMENT VARIABLES
Successfully updated the function processPayment.
Amazon Q
Ln 9, Col 3
Spaces: 2
UTF-8
LF
JavaScript
Lambda
Layout: US

Ⅲ. CREATE LAMBDA FUNCTION: UPDATEORDERSTATUS 📓

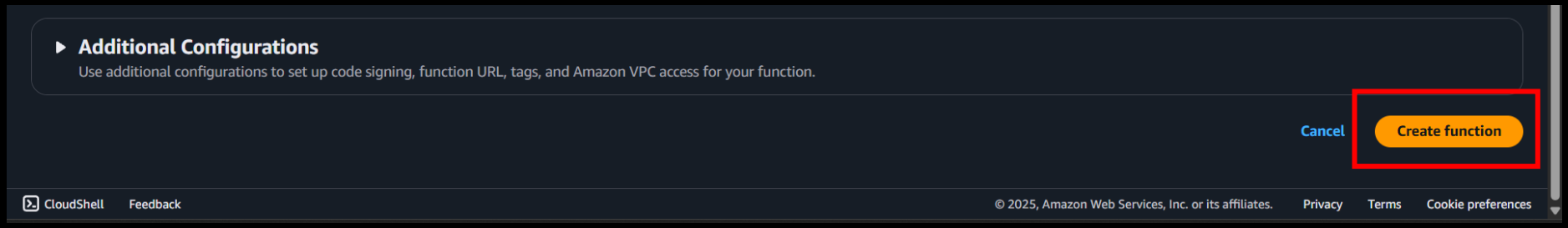
1. Create another Lambda function named **updateOrderStatus**

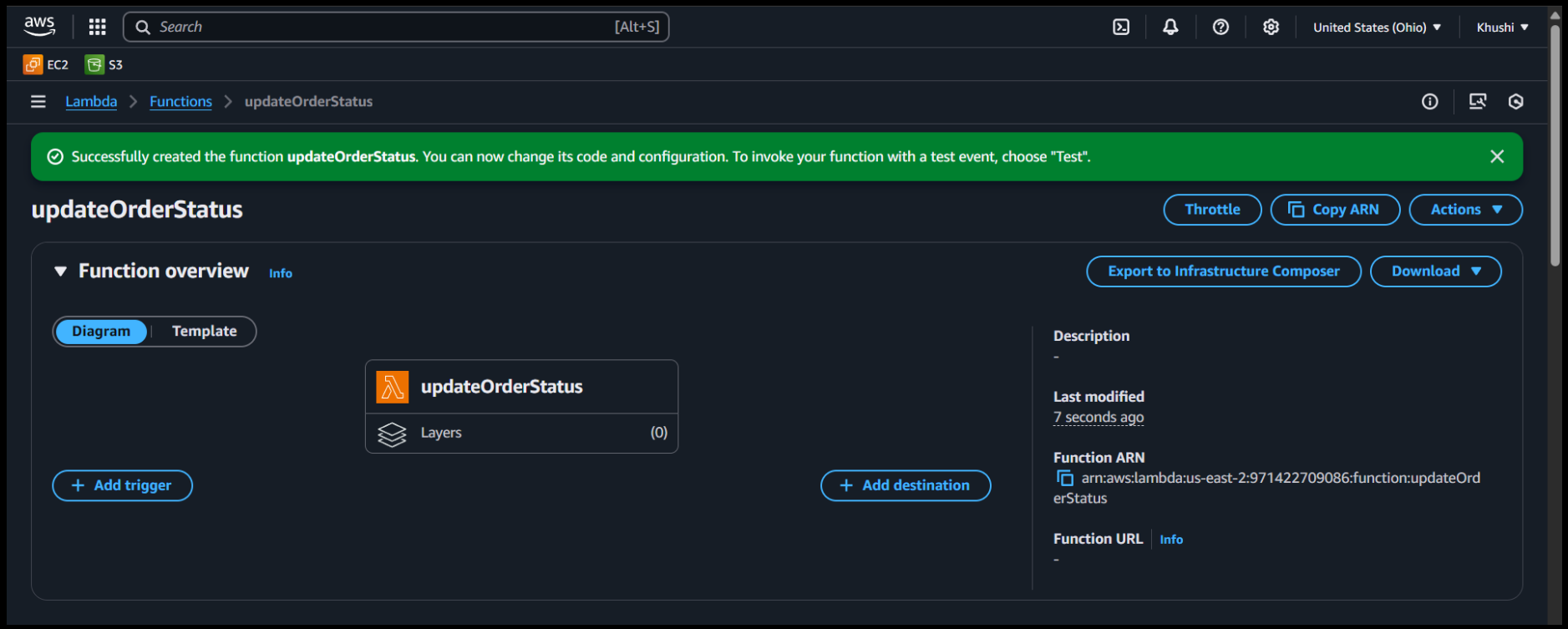


2. Leave all the other settings as default

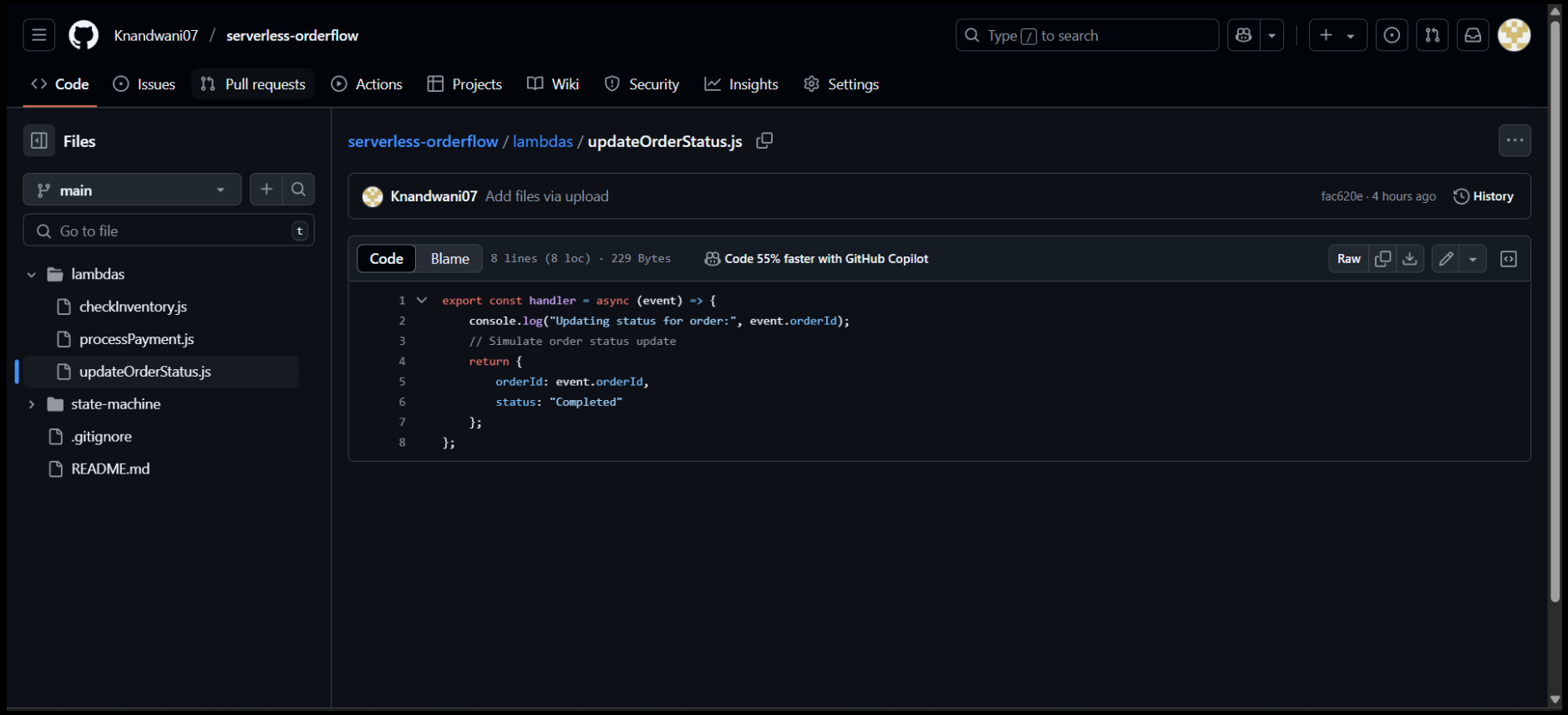


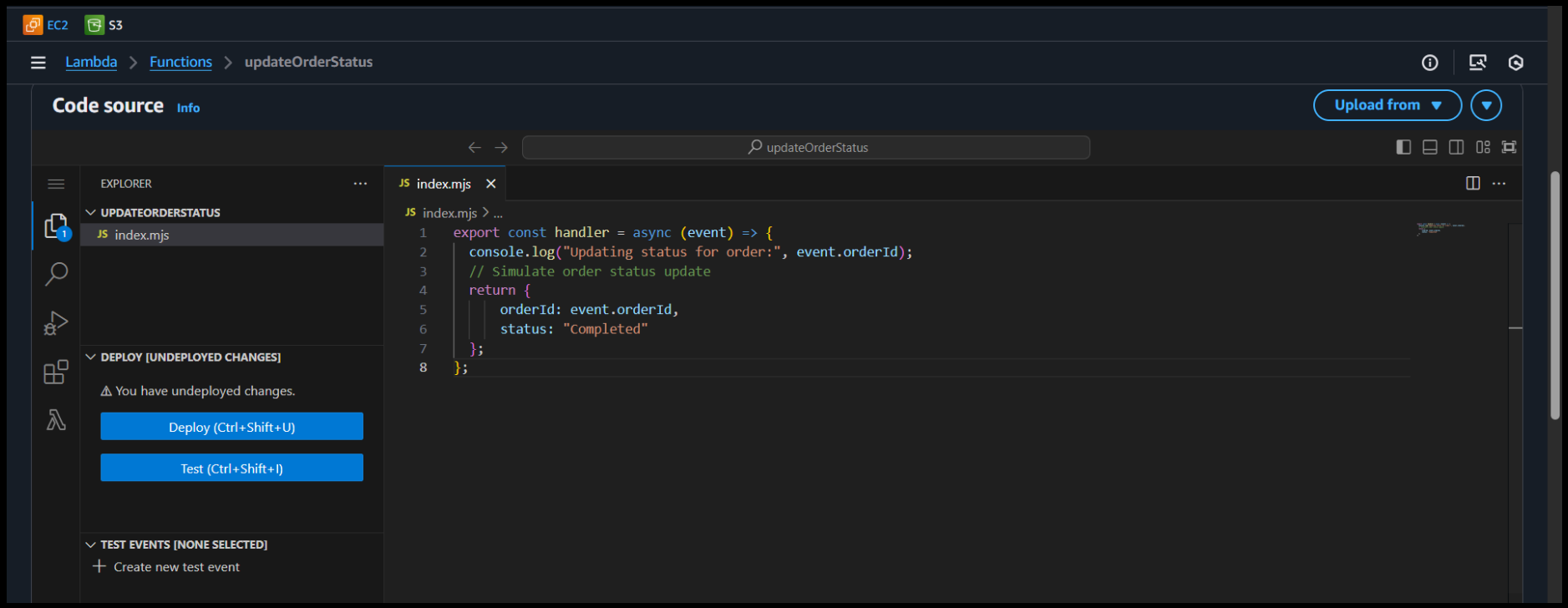
3. Click “Create Function”



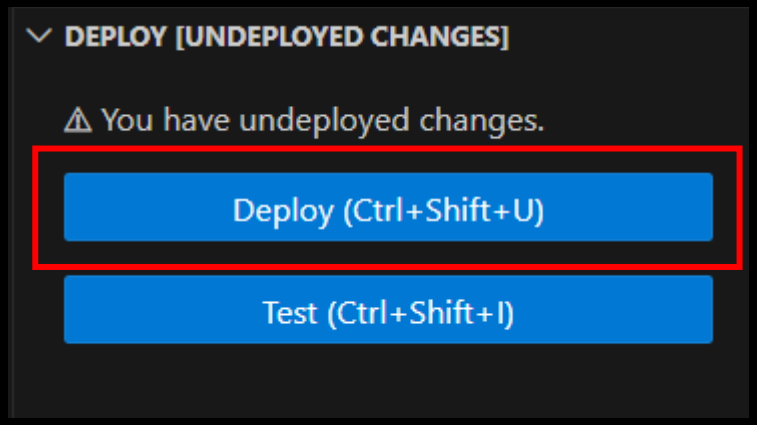

aws
Search
[Alt+S]
United States (Ohio)
Khushi
EC2
S3
Lambda
Functions
updateOrderStatus
Successfully created the function updateOrderStatus. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
updateOrderStatus
Throttle
Copy ARN
Actions
Function overview
Info
Export to Infrastructure Composer
Download
Diagram
Template
updateOrderStatus
Layers
(0)
Add trigger
Add destination
Description
-
Last modified
7 seconds ago
Function ARN
arn:aws:lambda:us-east-2:971422709086:function:updateOrderStatus
Function URL
Info
-

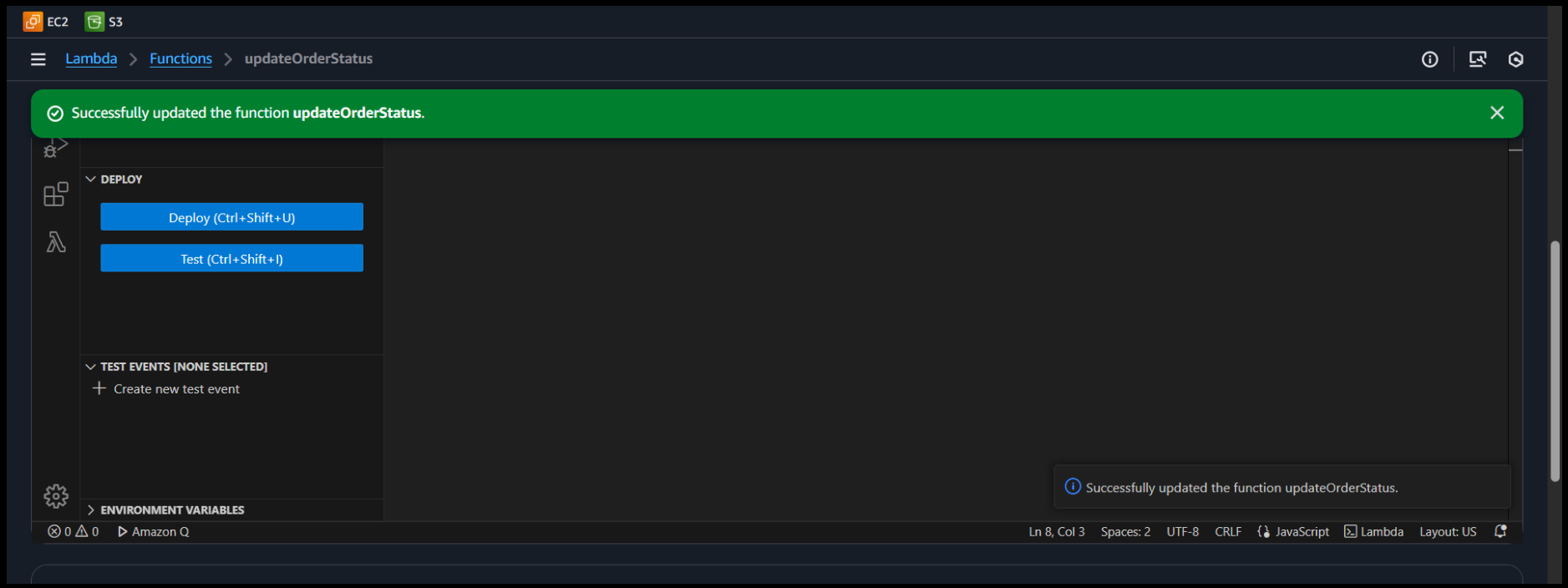
4. Paste the code in the editor




EC2
S3
Lambda > Functions > updateOrderStatus
Code source Info
Upload from
updateOrderStatus
EXPLORER
UPDATEORDERSTATUS
index.mjs
DEPLOY [UNDEPLOYED CHANGES]
You have undeployed changes.
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
index.mjs
index.mjs > ...
export const handler = async (event) => {
console.log("Updating status for order:", event.orderId);
// Simulate order status update
return {
orderId: event.orderId,
status: "Completed"
};
};

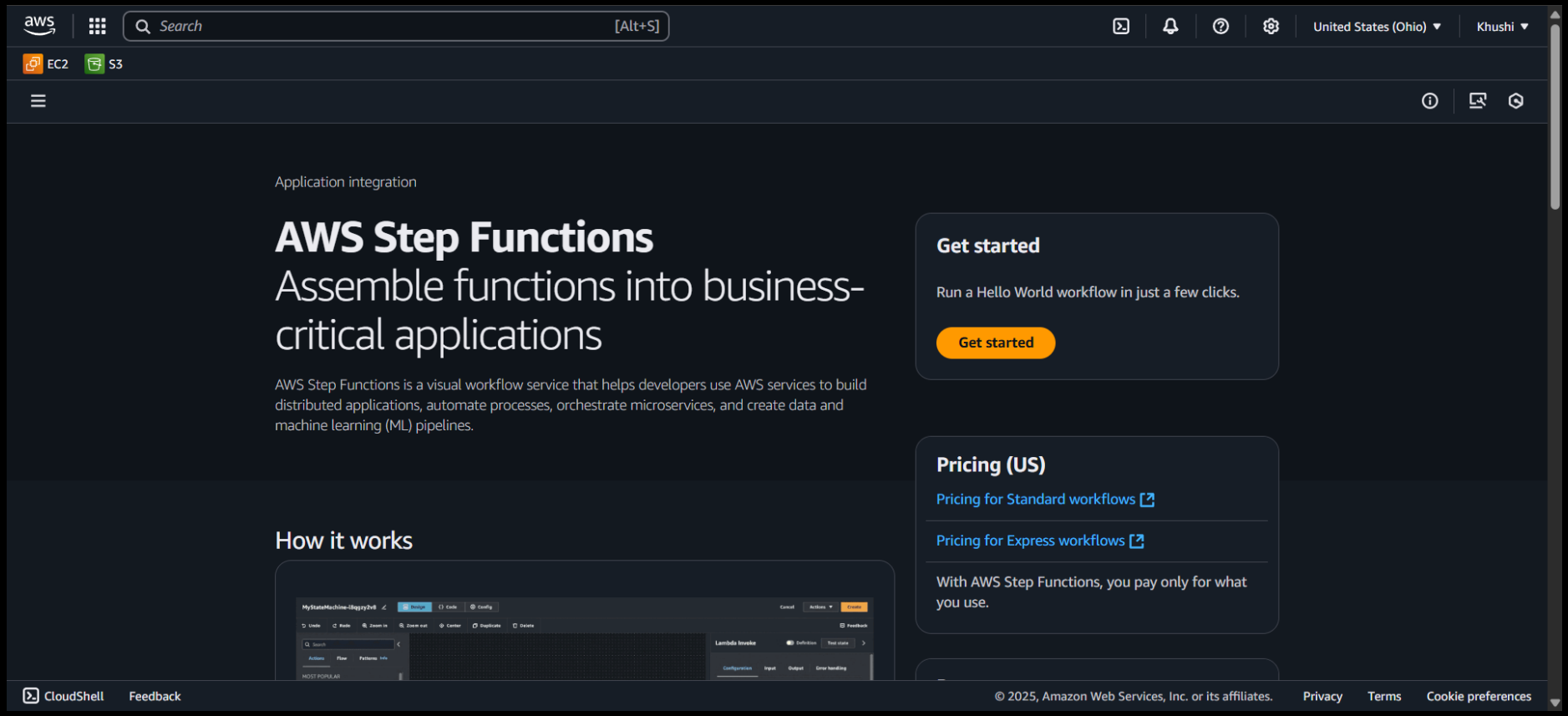
5. Click on “Deploy”



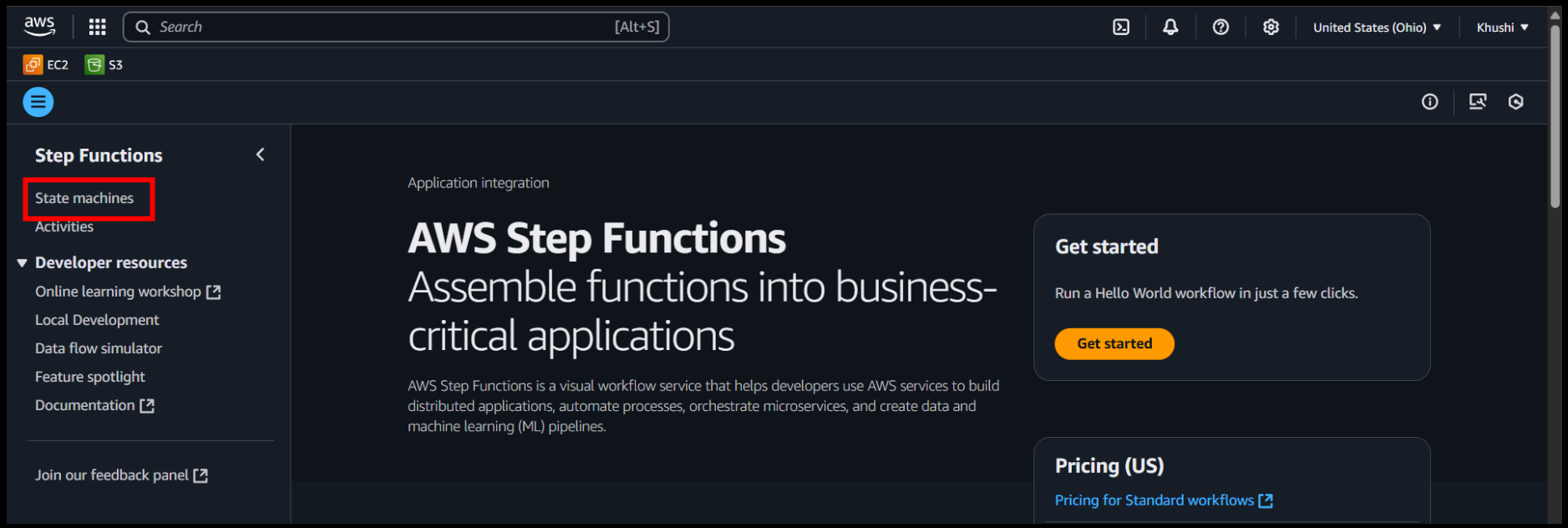

EC2
S3
Lambda
Functions
updateOrderStatus
Successfully updated the function updateOrderStatus.
DEPLOY
Deploy (Ctrl+Shift+U)
Test (Ctrl+Shift+I)
TEST EVENTS [NONE SELECTED]
Create new test event
ENVIRONMENT VARIABLES
Successfully updated the function updateOrderStatus.
0 0
Amazon Q
Ln 8, Col 3
Spaces: 2
UTF-8
CRLF
JavaScript
Lambda
Layout: US

IV. CONFIGURE AWS STEP FUNCTIONS 🤖

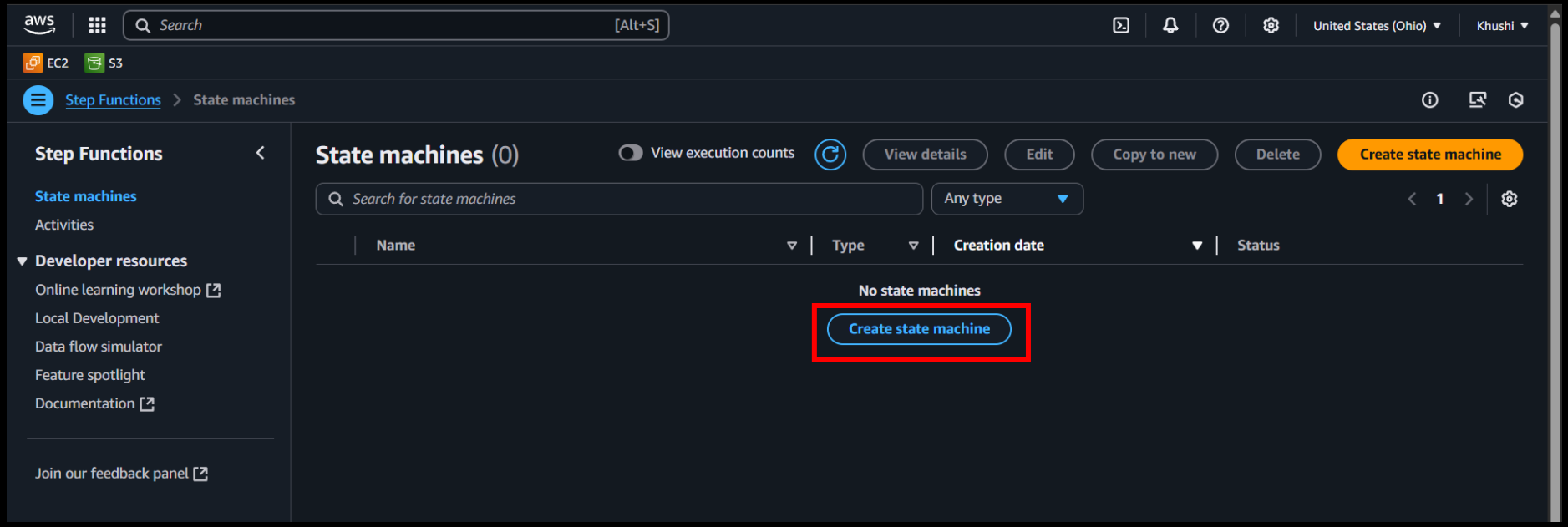
1. Go to AWS Step Functions from the Console



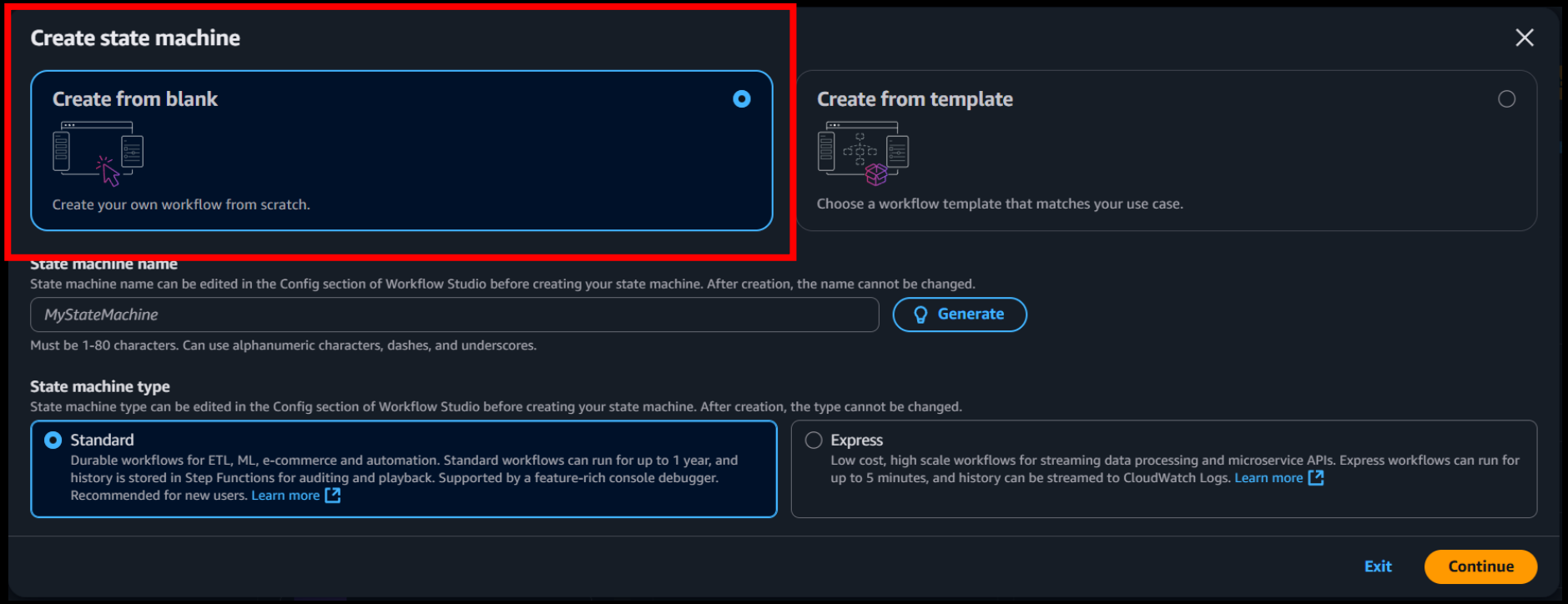
2. In the sidebar, click on “State Machines”



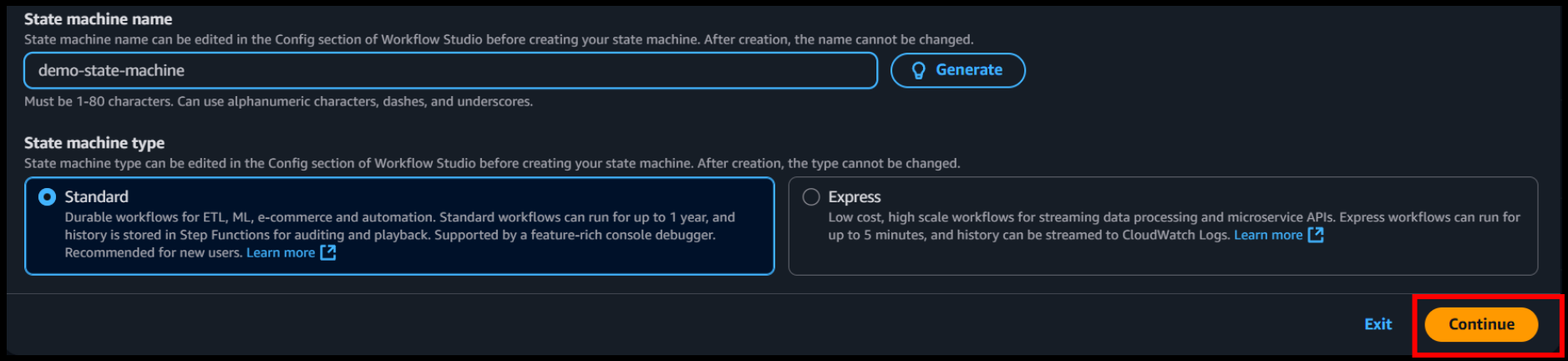
3. Click “Create State Machine”



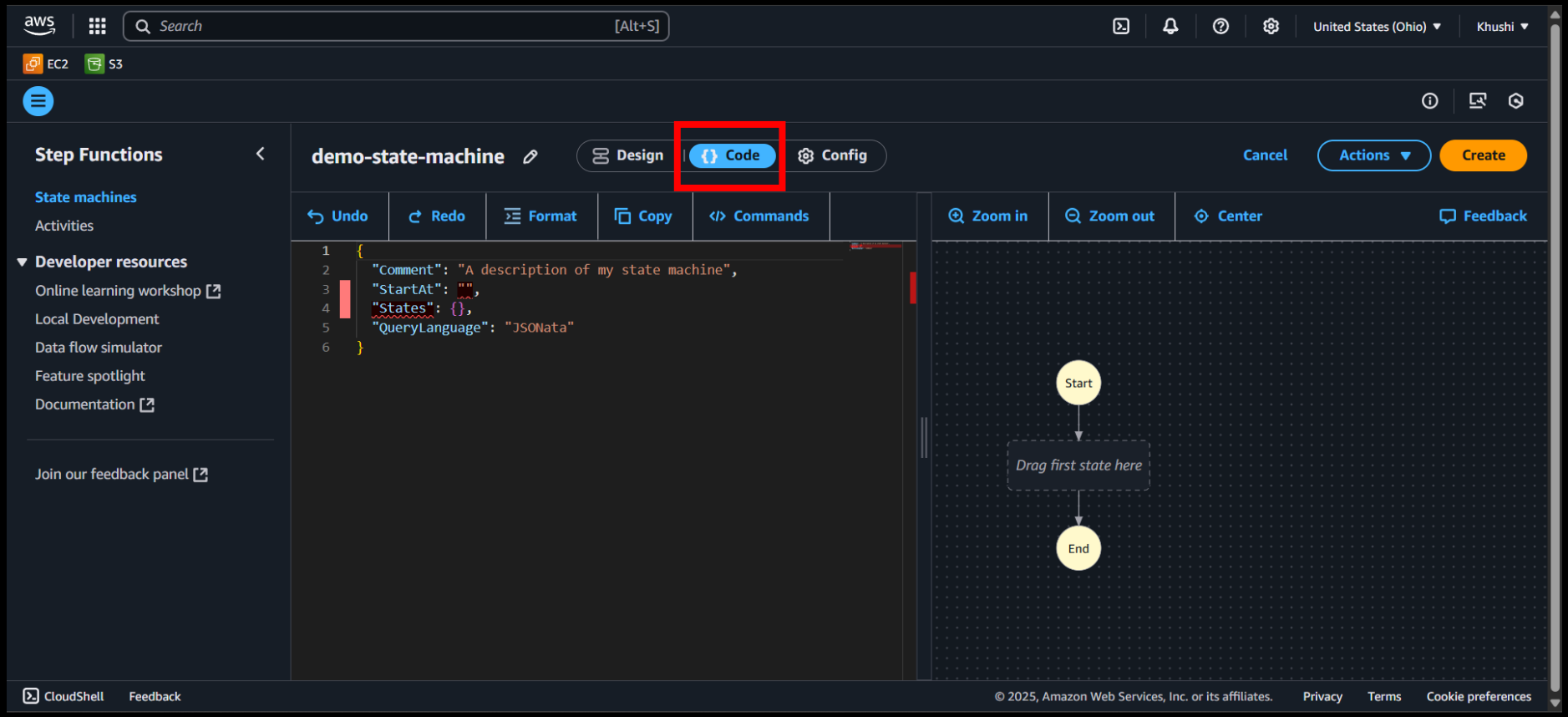
4. Choose “Blank Template” and click “Select”



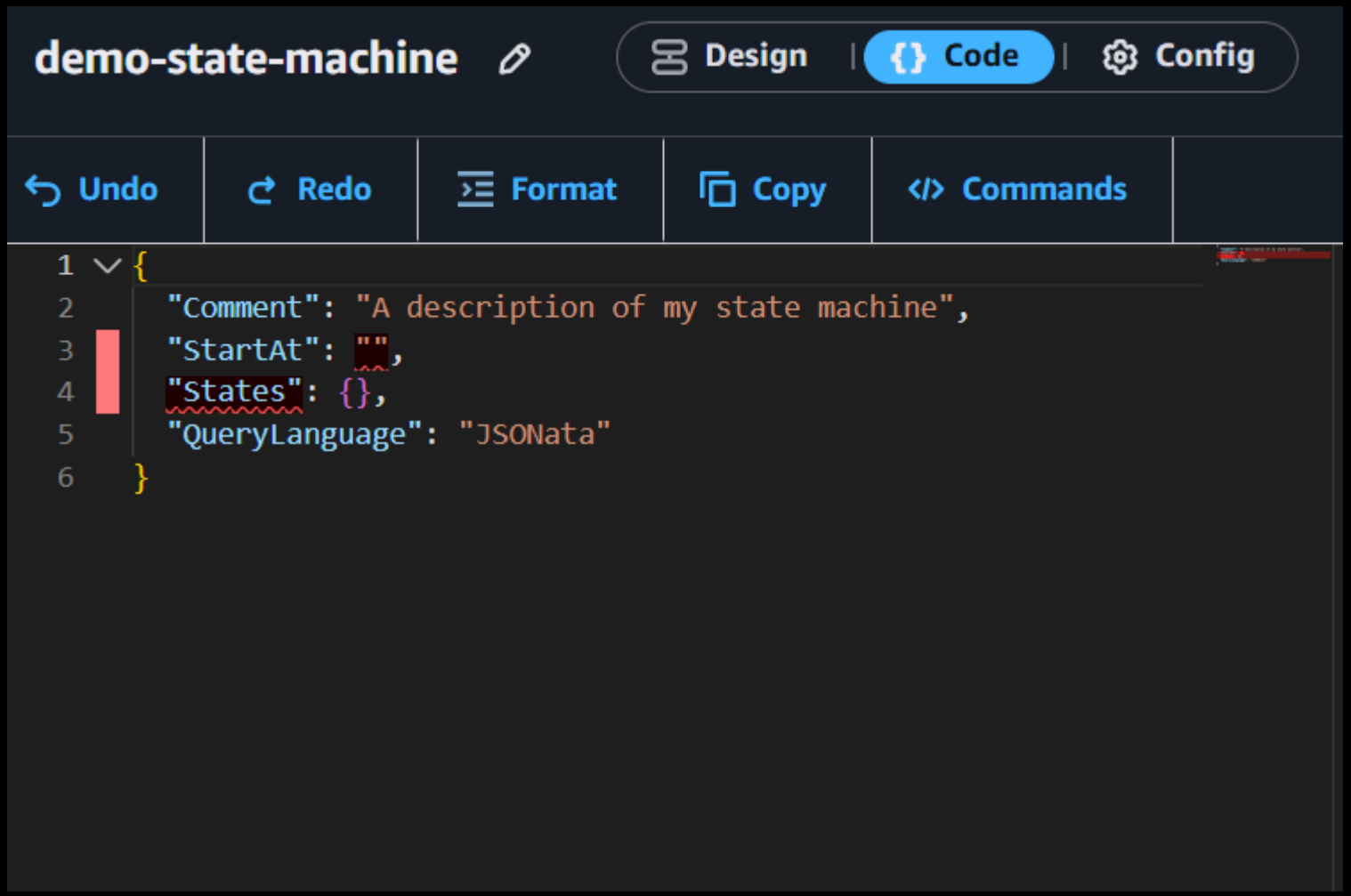
5. Give your state-machine a name and click on continue

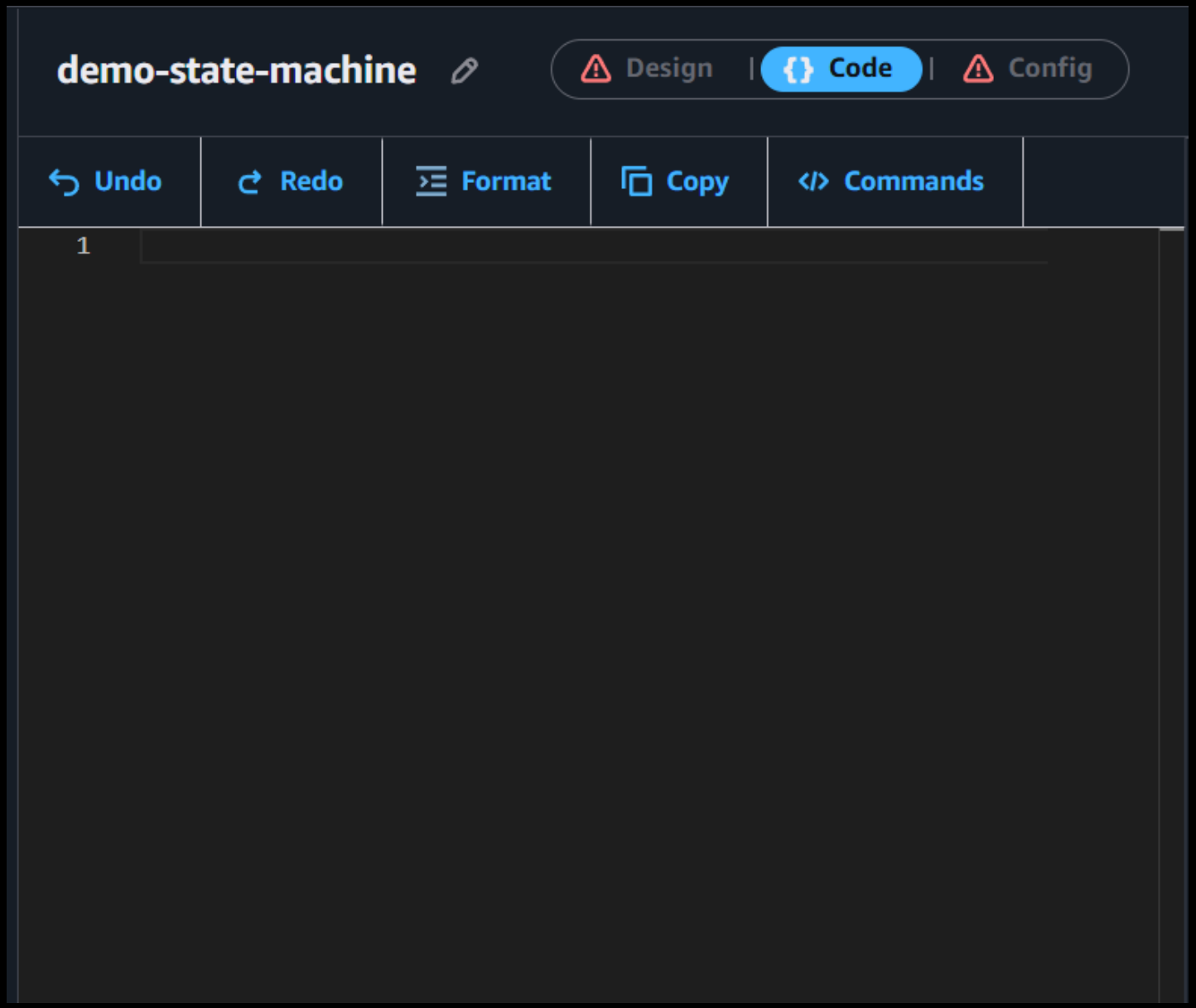


6. Go to the “Code” tab

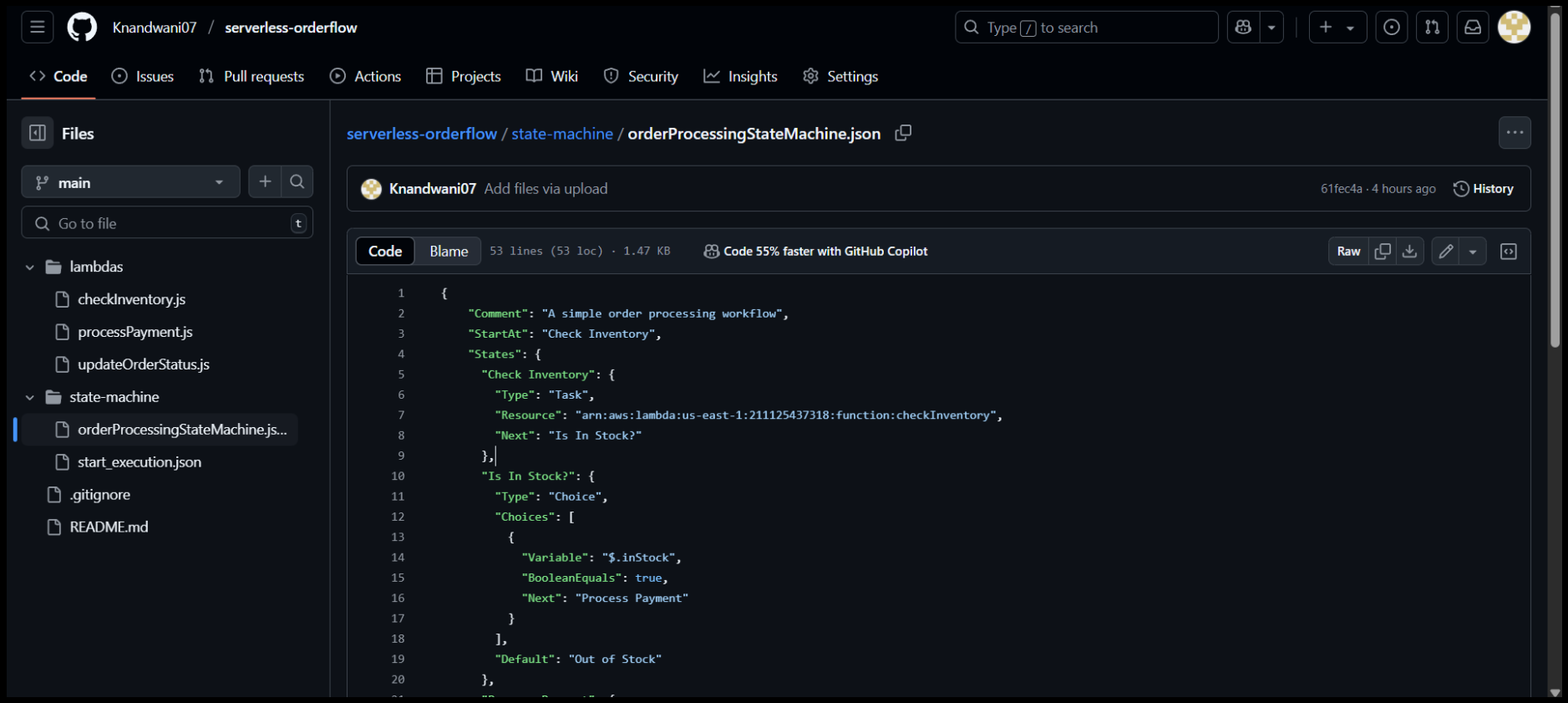


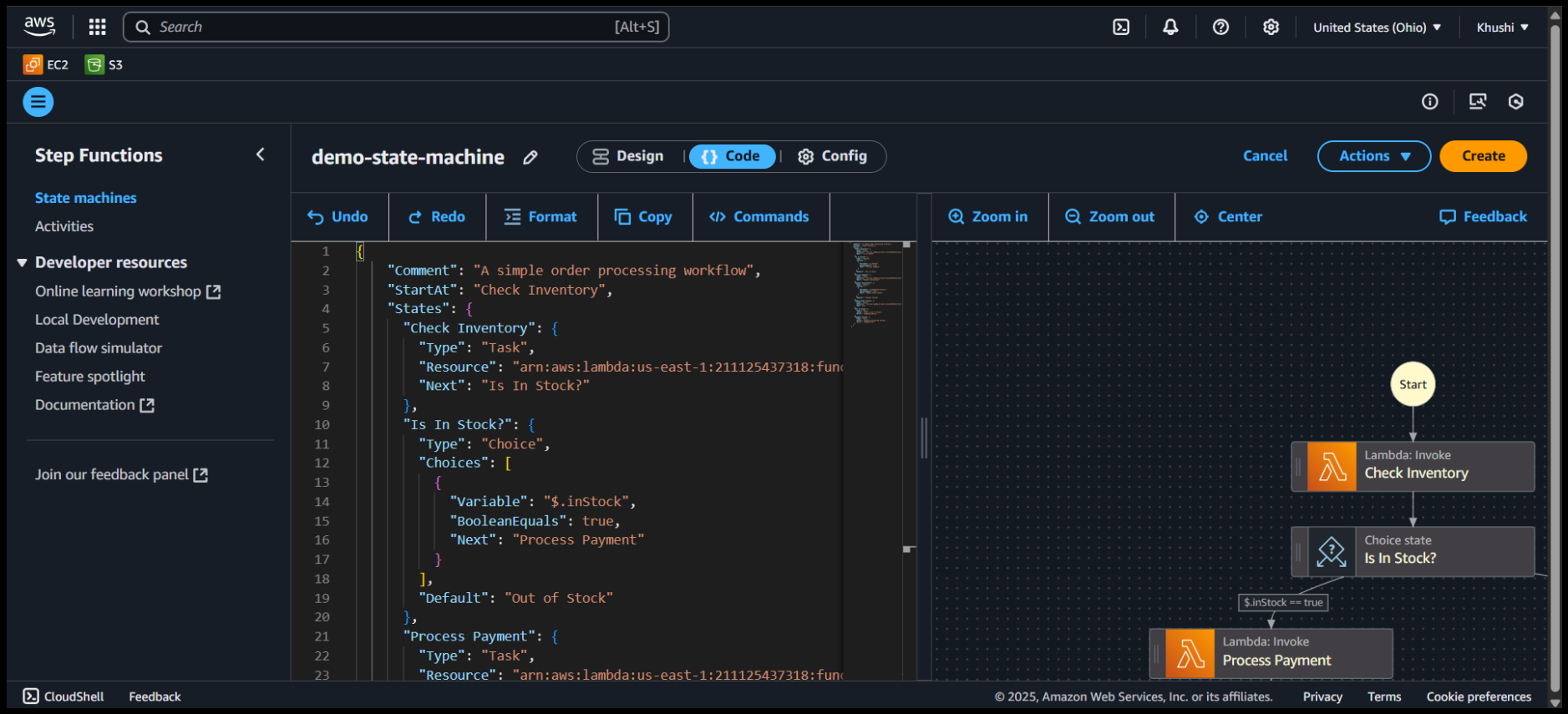
7. Remove default code

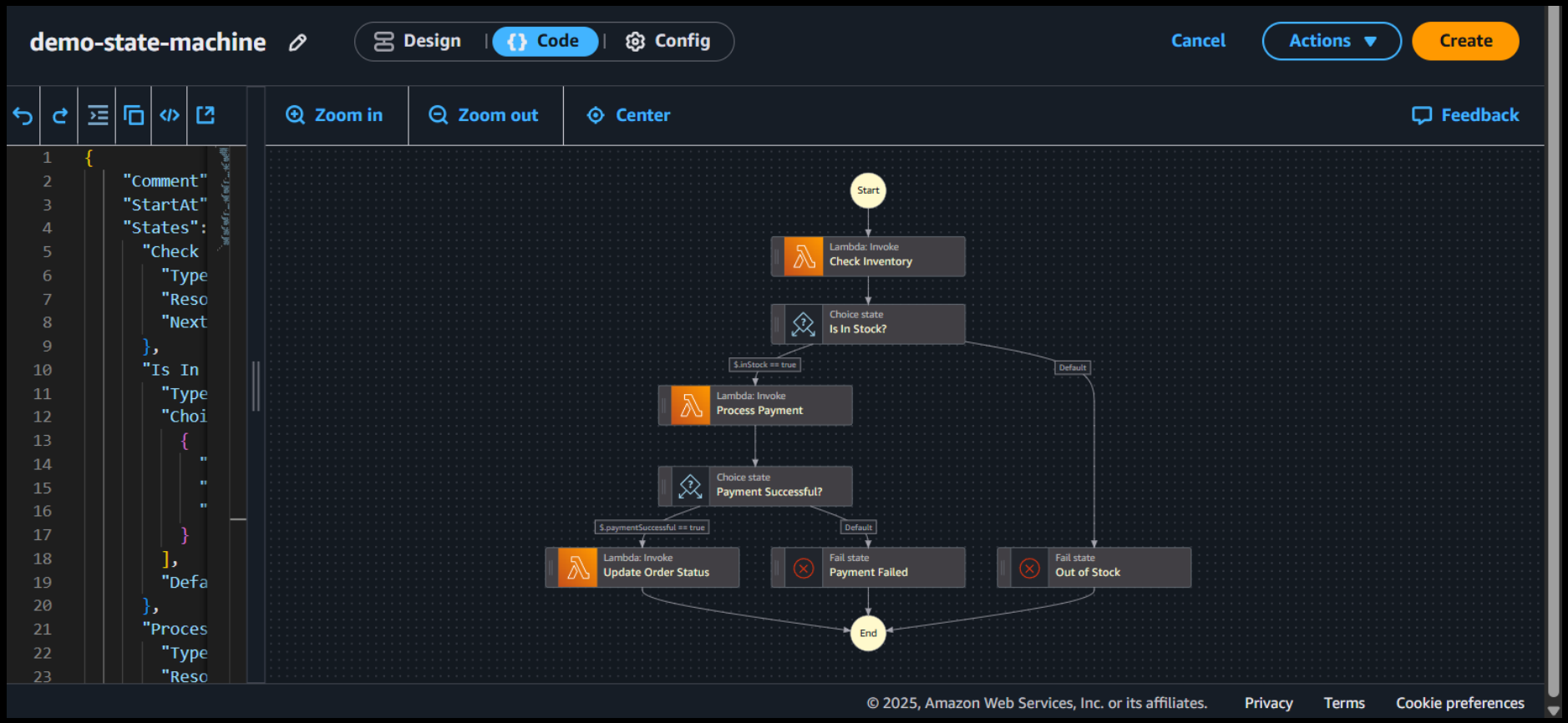



demo-state-machine
Design
Code
Config
Undo
Redo
Format
Copy
Commands
1

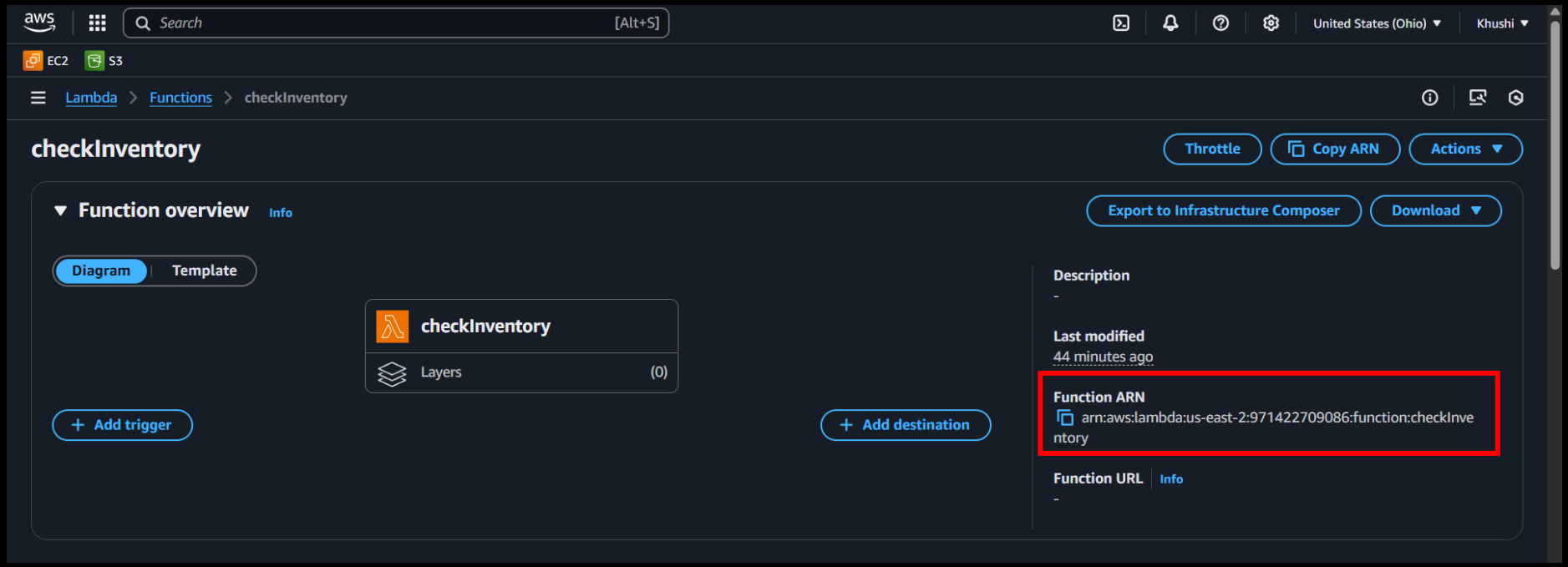
8. Paste your Step Functions JSON code



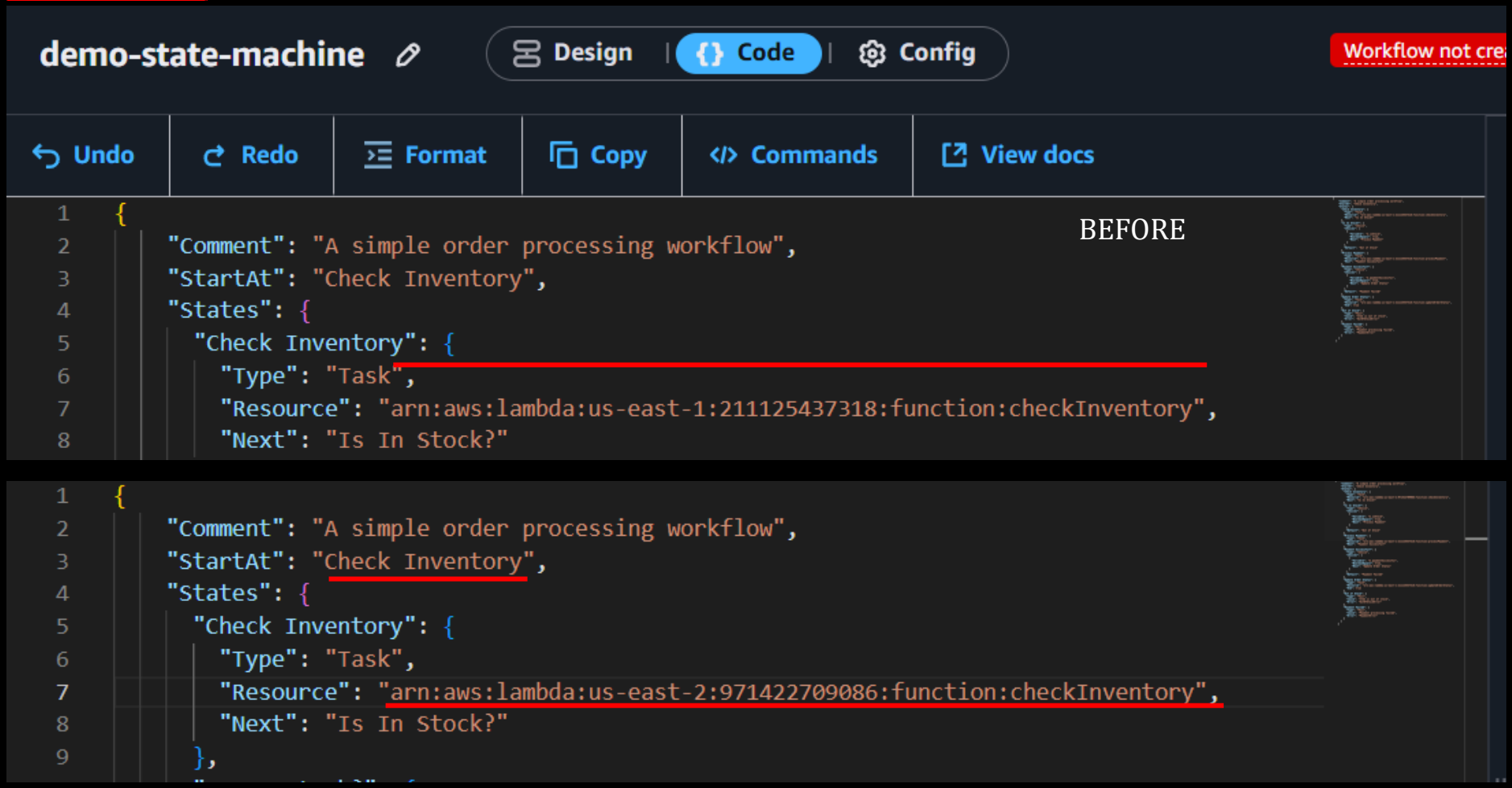

aws
Search
[Alt+S]
United States (Ohio)
Khushi
EC2
S3
Step Functions
State machines
Activities
Developer resources
Online learning workshop
Local Development
Data flow simulator
Feature spotlight
Documentation
Join our feedback panel
demo-state-machine
Design
Code
Config
Cancel
Actions
Create
Undo
Redo
Format
Copy
Commands
Zoom in
Zoom out
Center
Feedback
{
"Comment": "A simple order processing workflow",
"StartAt": "Check Inventory",
"States": {
"Check Inventory": {
"Type": "Task",
"Resource": "arn:aws:lambda:us-east-1:211125437318:func
"Next": "Is In Stock?"
},
"Is In Stock?": {
"Type": "Choice",
"Choices": [
{
"Variable": "$.inStock",
"BooleanEquals": true,
"Next": "Process Payment"
}
],
"Default": "Out of Stock"
},
"Process Payment": {
"Type": "Task",
"Resource": "arn:aws:lambda:us-east-1:211125437318:func
Start
Lambda: Invoke
Check Inventory
Choice state
Is In Stock?
$.inStock == true
Lambda: Invoke
Process Payment
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


demo-state-machine
Design
Code
Config
Cancel
Actions
Create
Zoom in
Zoom out
Center
Feedback
{
"Comment"
"StartAt"
"States":
"Check
"Type
"Reso
"Next
},
"Is In
"Type
"Choi
{
"
"
"
}
],
"Defa
},
"Proces
"Type
"Reso
Start
Lambda: Invoke
Check Inventory
Choice state
Is In Stock?
$.inStock == true
Default
Lambda: Invoke
Process Payment
Choice state
Payment Successful?
$.paymentSuccessful == true
Default
Lambda: Invoke
Update Order Status
Fail state
Payment Failed
Fail state
Out of Stock
End
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

9. Replace the placeholder ARN values with actual Lambda ARNs
    - Copy ARN from each Lambda function

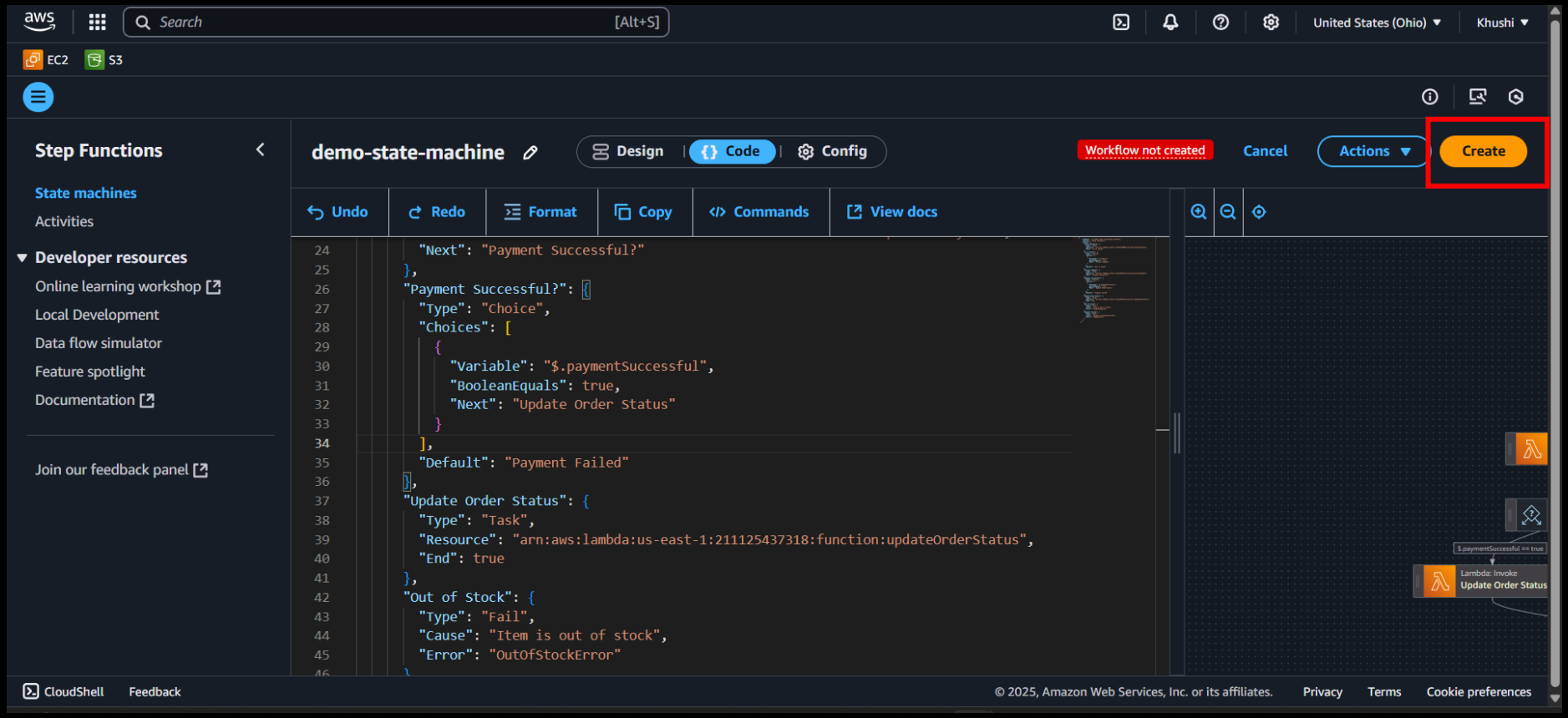


- Replace only the account ID and function names if other parts match

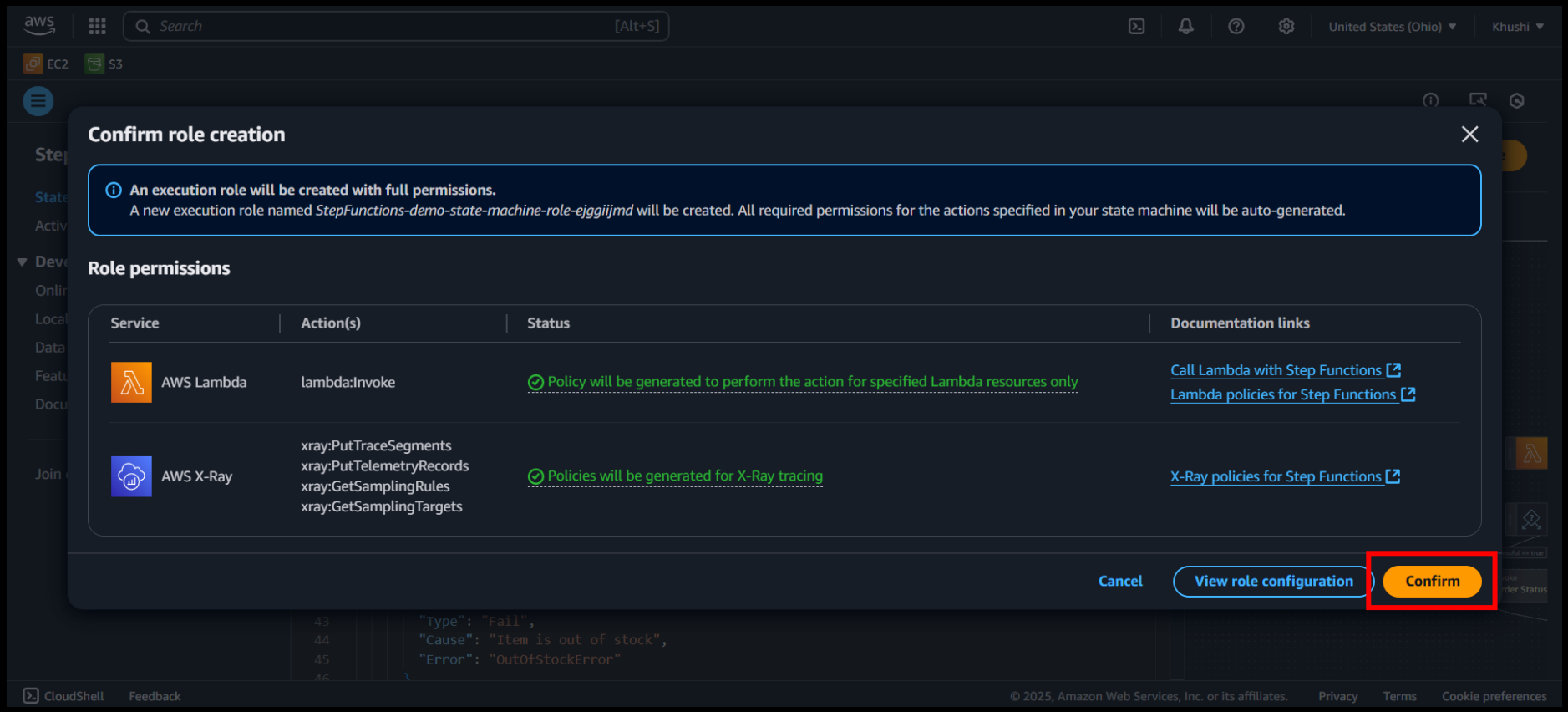


Change for the other 2 functions too!

10. Click “Create State Machine”

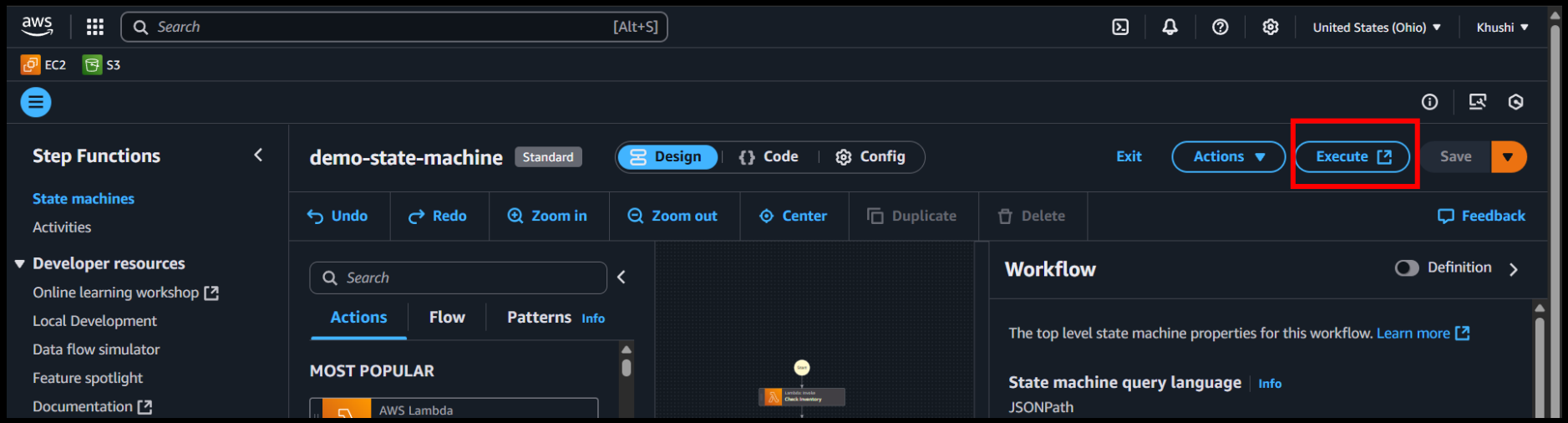


11. When prompted, confirm IAM role creation

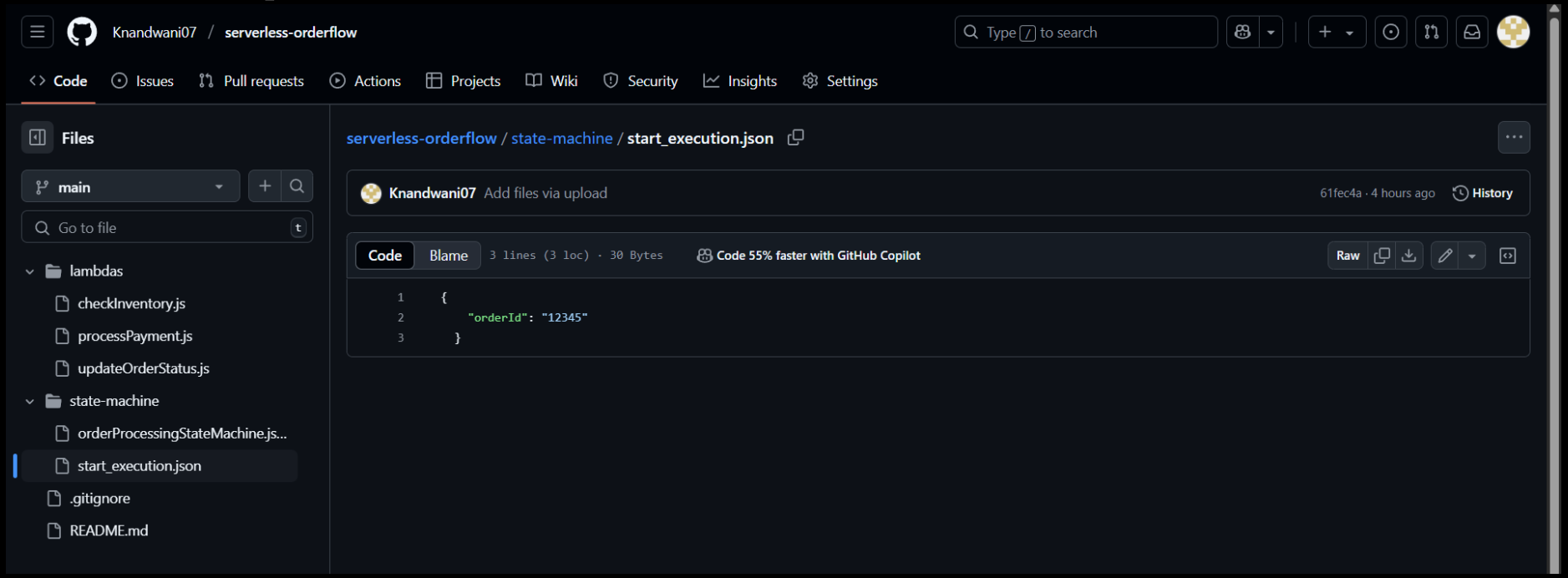


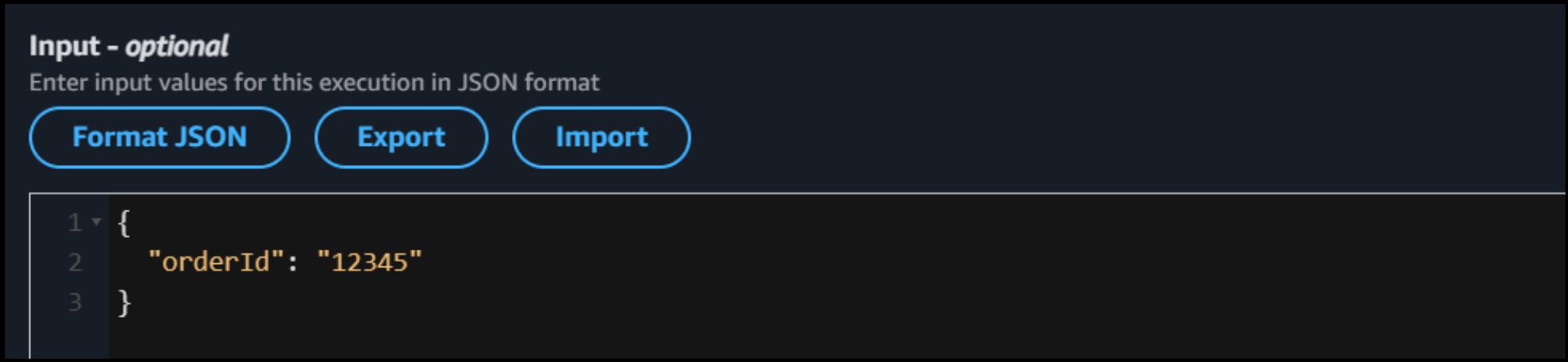
V. EXECUTE THE STATE MACHINE 🚀

1. Click On “**Execute**”



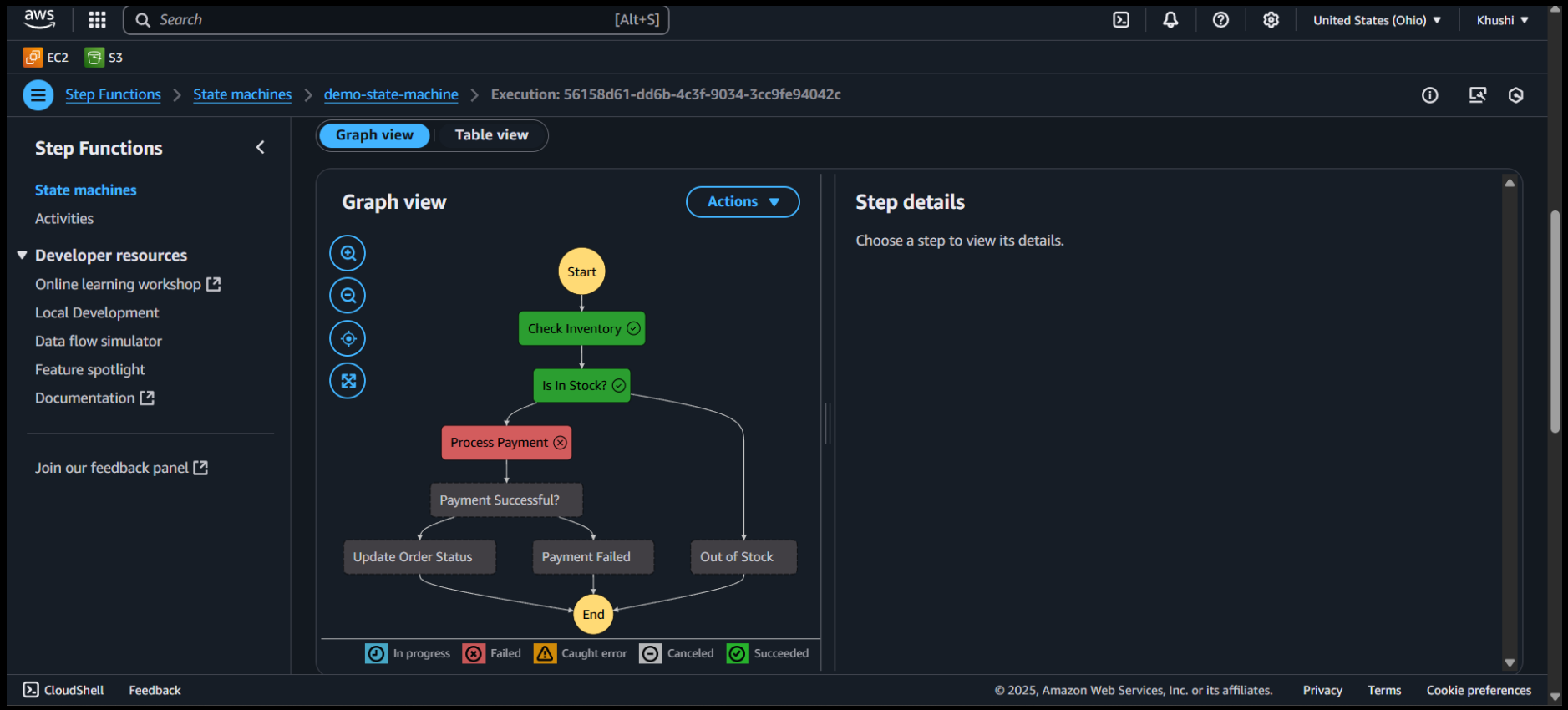
2. Enter JSON input



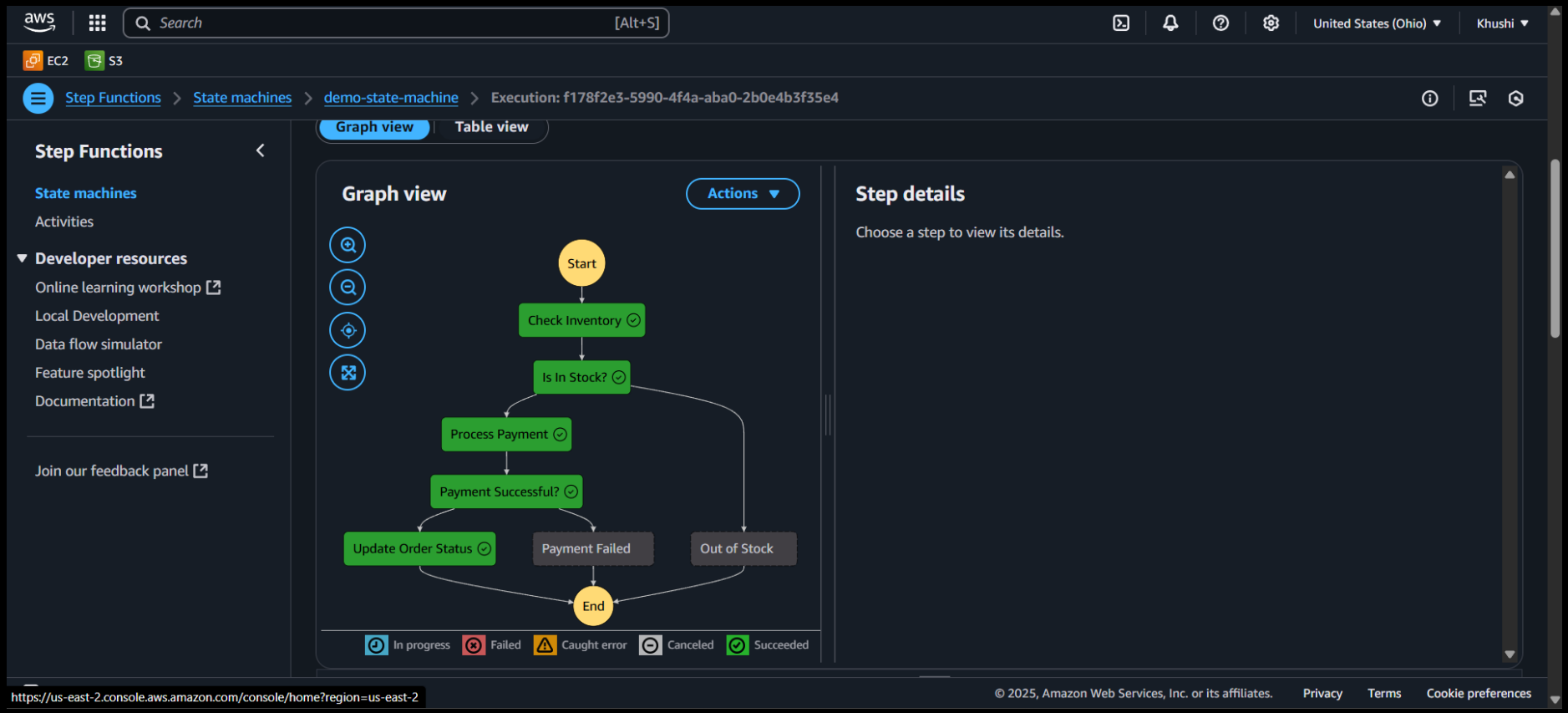


3. Click “**Start Execution**”

4. Observe execution flow via graph and table views

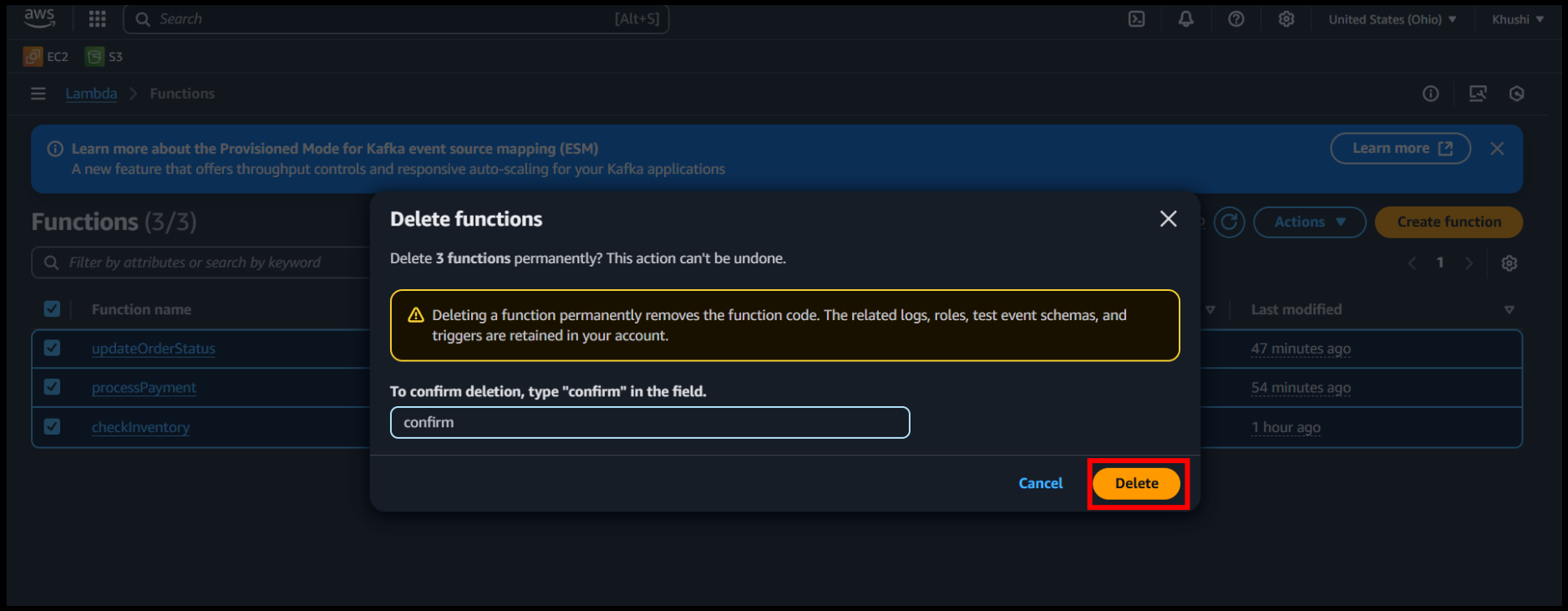


5. You may run additional executions to test different scenarios

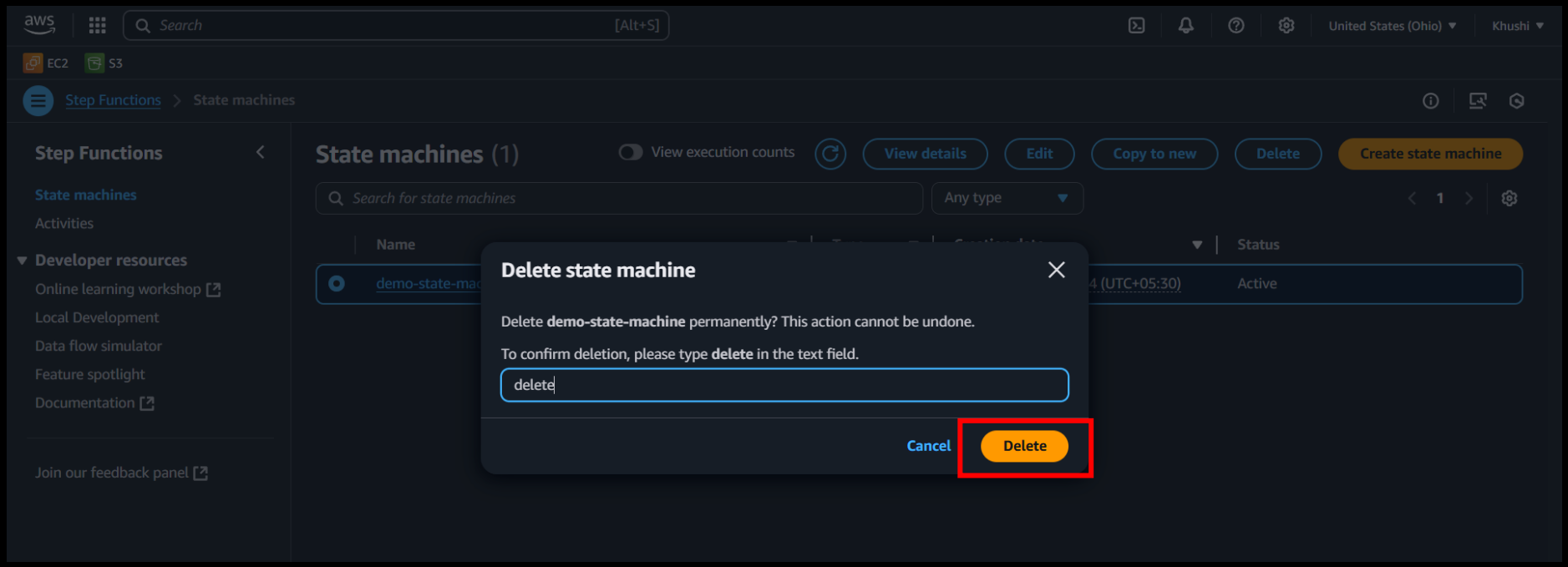


VI. DELETION & CLEANUP 🧹

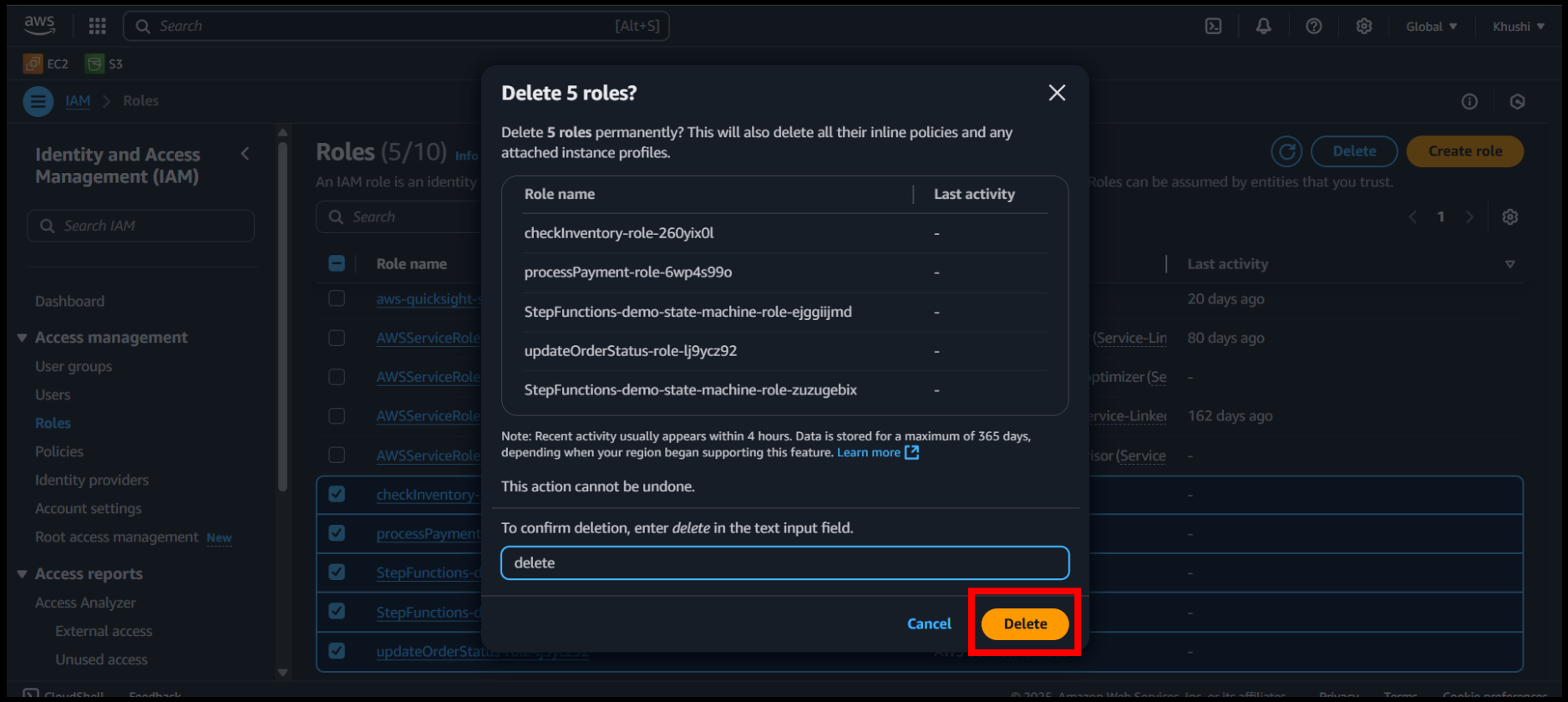
1. Delete all three Lambda functions



2. Delete the state machine from AWS Step Functions



3. Remove IAM roles that were automatically created



⁘ CONCLUSION

This project provided a practical walkthrough of designing a serverless order pipeline using AWS Lambda and Step Functions. From defining isolated microservices to integrating them into a cohesive state machine, you now have a blueprint for creating scalable, event-driven architectures. Cleaning up resources post-testing ensures a lean and cost-effective deployment strategy.

“Automate the repetitive, so you can focus on the innovative.”