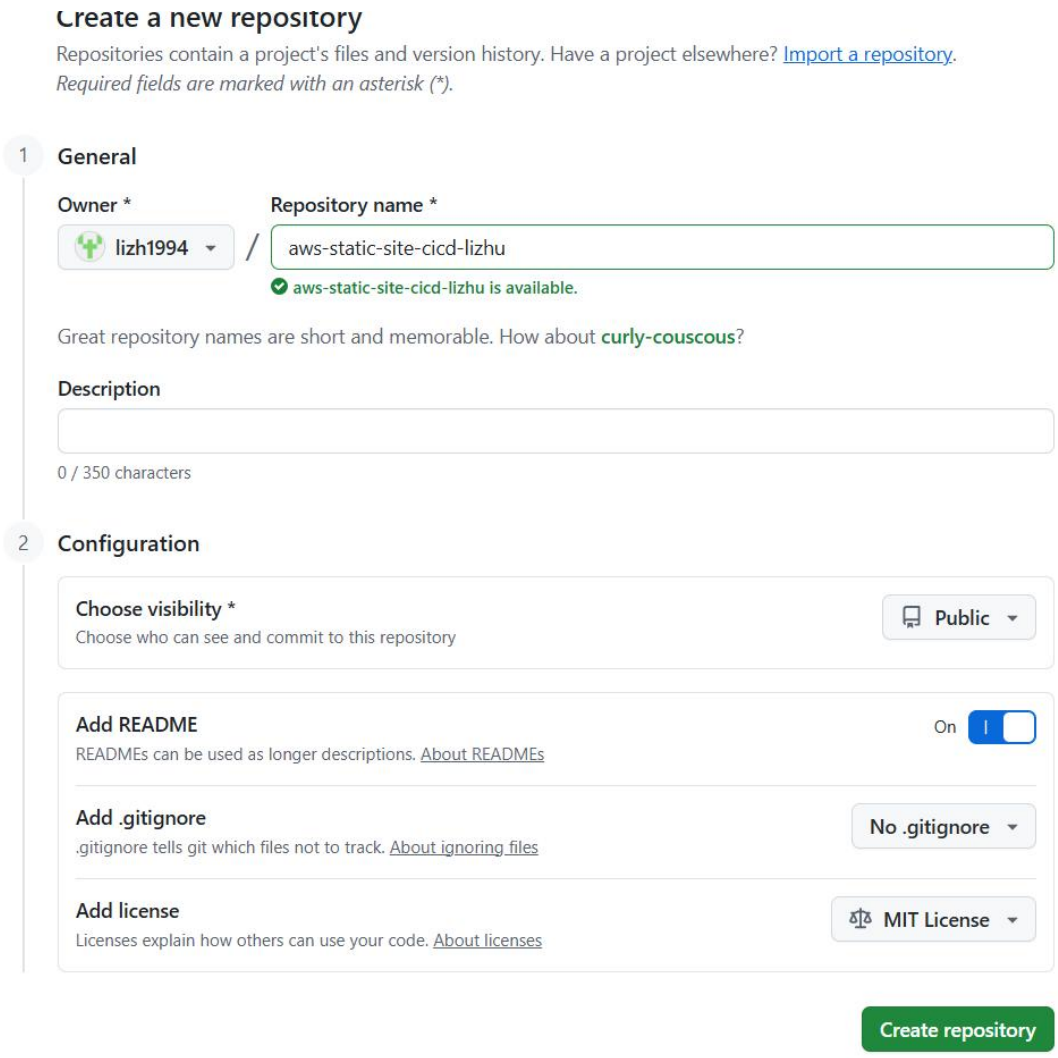
Project Documentation: AWS Static Website CI/CD Pipeline

1. Project Setup Overview

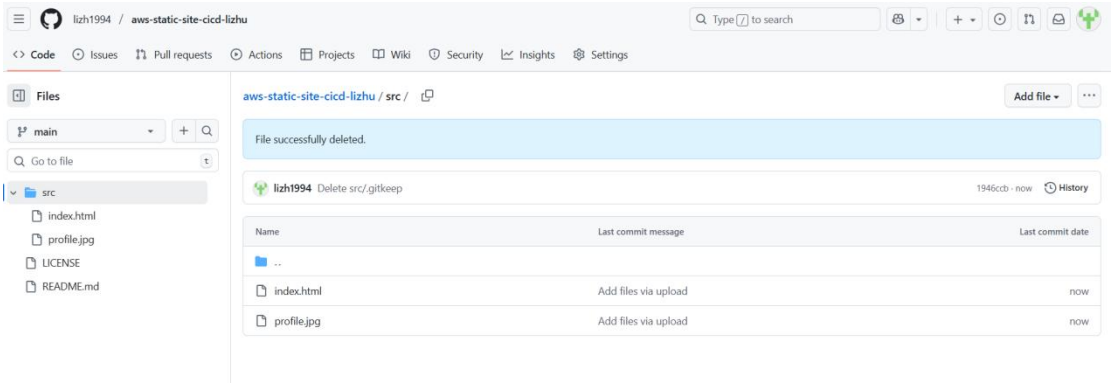
Goal: Build a CI/CD pipeline to deploy a static website (HTML, CSS, JS) on AWS.
Key AWS services used: S3, CodePipeline, CodeBuild, IAM, CloudFront, CloudWatch, SNS.
GitHub repository as the source of pipeline.

2. Prepare the Source Repository

Set up a GitHub repository to store the static website files (index.html and images). This repository will serve as the source for the CI/CD pipeline.

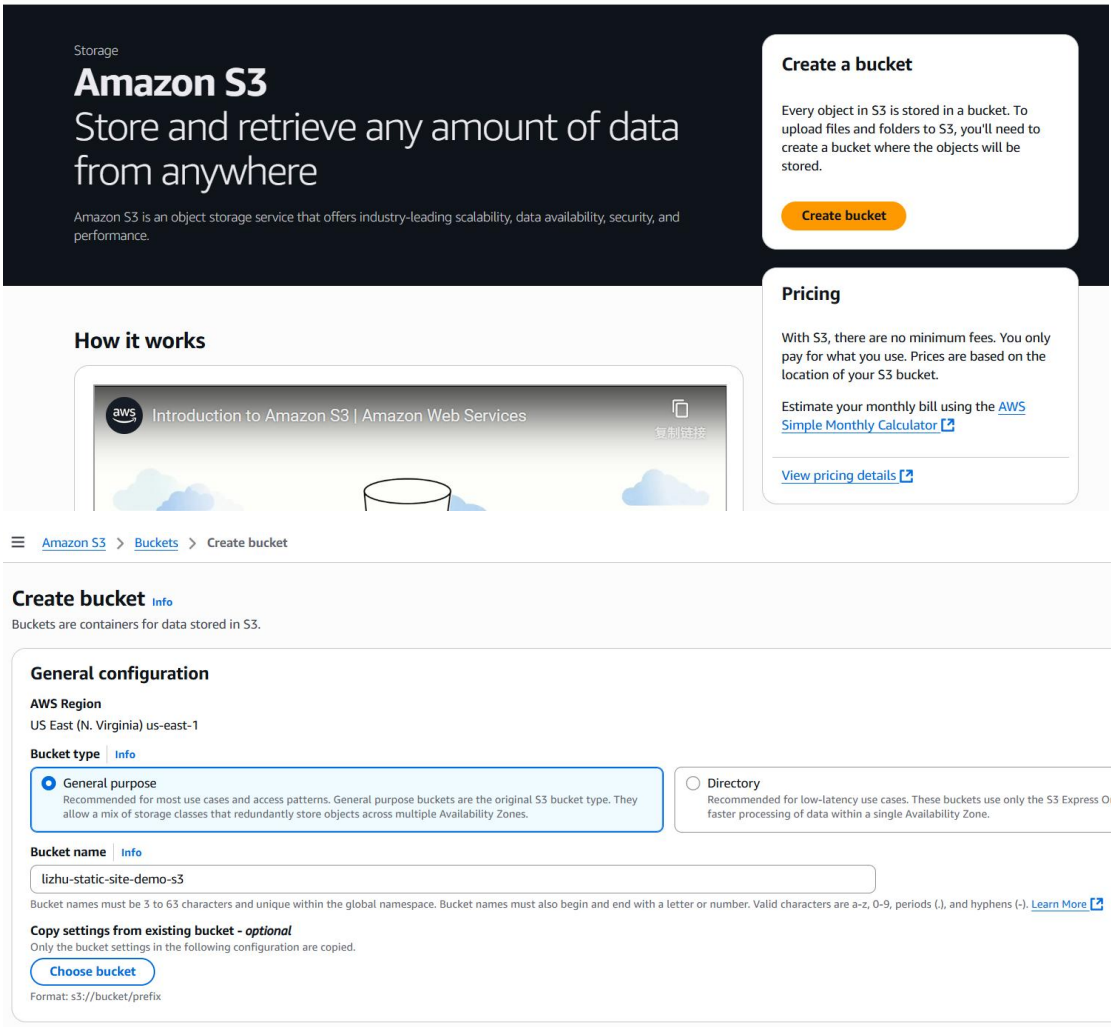


Add initial static website files (index.html, images), commit and push changes.



3. Create and Configure S3 Buckets

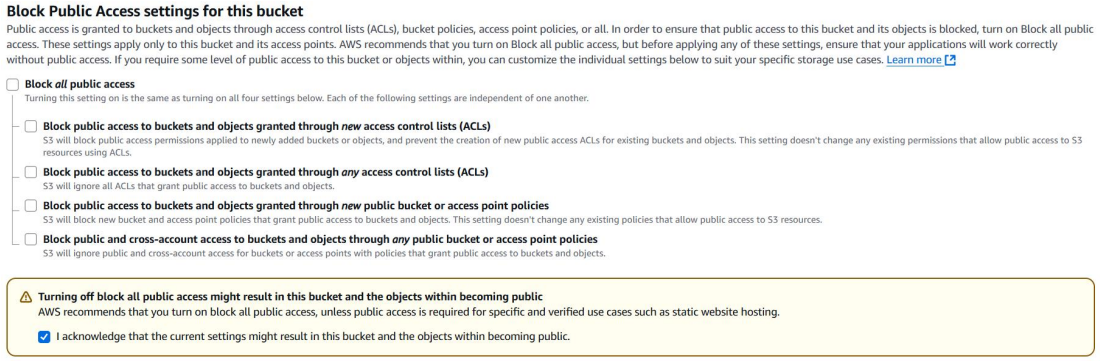
Create an S3 bucket for static website hosting.

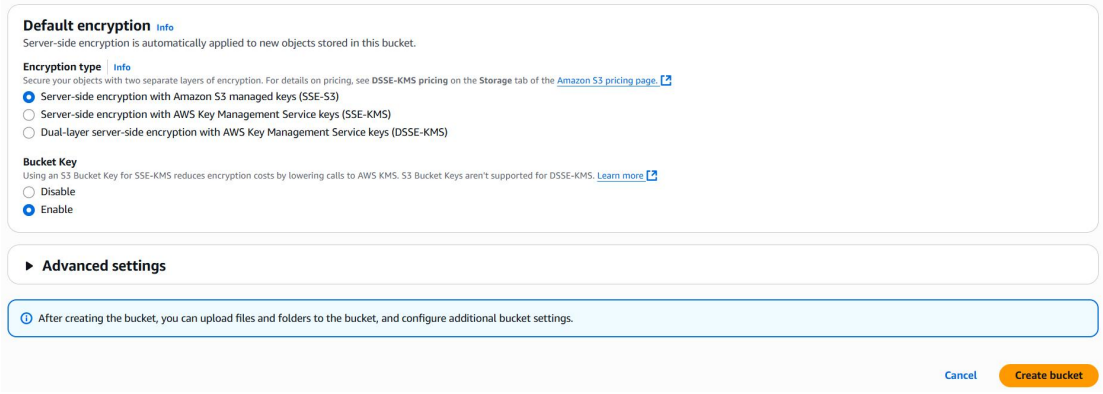


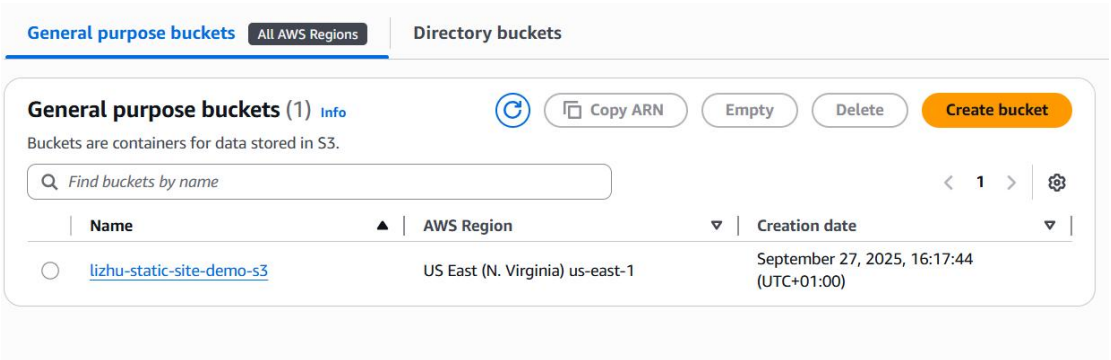
Configuration:

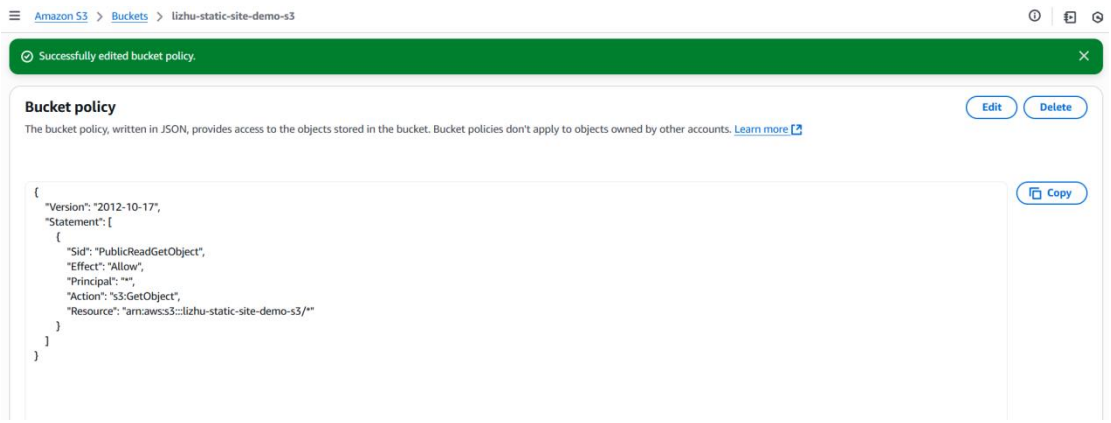
- Block Public Access: unchecked (required for static hosting).
- Versioning: enabled.

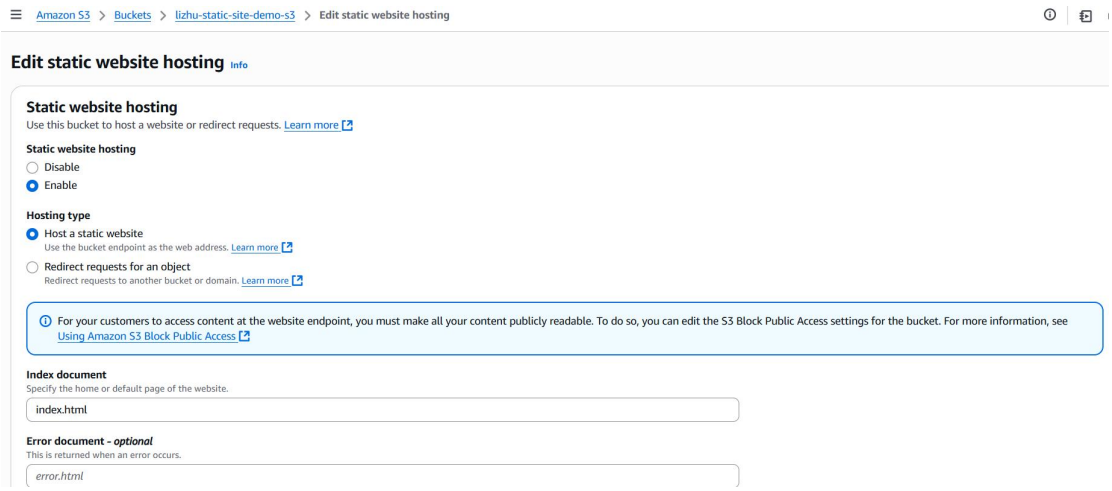
- Encryption: enabled with SSE-S3.
- Bucket Policy: restricted to s3:GetObject only to allow CloudFront access.



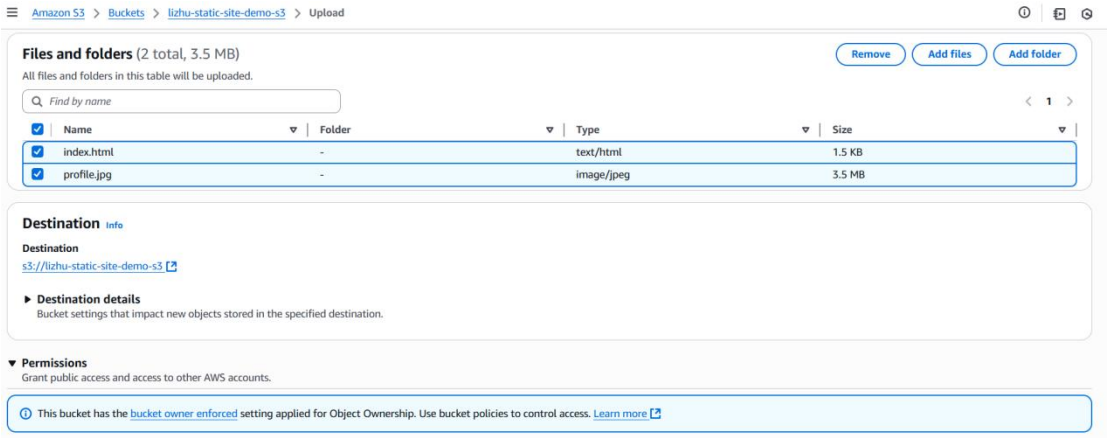


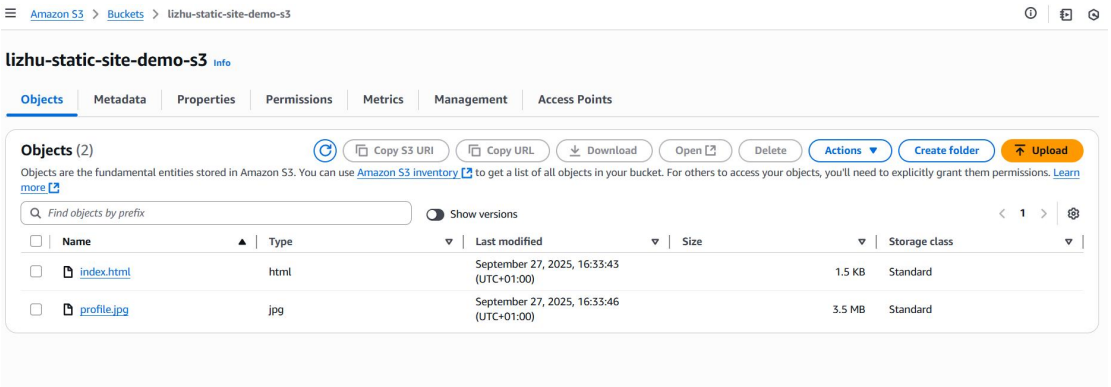


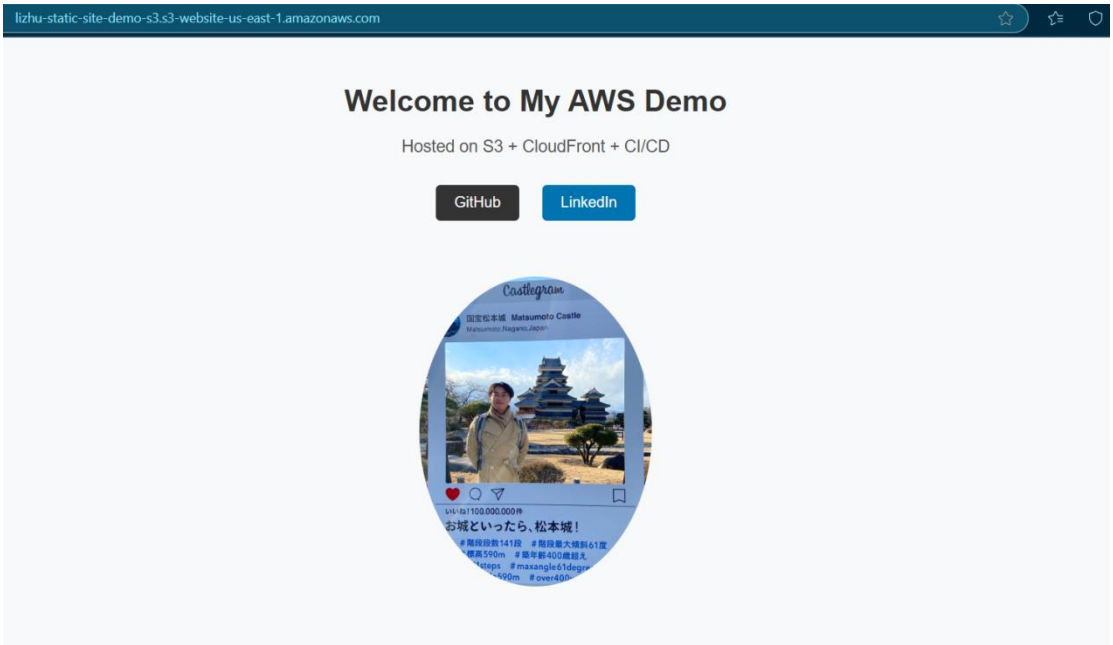




Upload the initial index.html and images, and test the static site via the S3 object URL.







In this project, the deploy S3 bucket has a bucket policy that grants public read access (s3:GetObject) for website visitors.
At the same time, the CodeBuild IAM Role is attached with permissions to read/write/delete/list objects in the deploy bucket.

Key Point:
- Bucket Policy controls who (principals such as public, other accounts) can access the bucket.
- IAM Role Policy controls what the CodeBuild service itself can do.
- The final permission = intersection of both (the request must be allowed by both IAM + Bucket policy).

Why this matters:
This design ensures public visitors only have read access, while CodeBuild has full update capability. It' s a clear example of IAM + Bucket Policy collaboration, aligning with the principle of least privilege.

4. Configure CloudFront Distribution and Functions

Create a CloudFront distribution with the Deploy Bucket as the origin.

Networking & Content Delivery

Amazon CloudFront

Securely deliver content with low latency and high transfer speeds

Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency and high transfer speeds.

Get started with CloudFront

Enable accelerated, reliable and secure content delivery for Amazon S3 buckets, Application Load Balancers, Amazon API Gateway APIs, and more in 5 minutes or less.

Create a CloudFront distribution

AWS Free Tier

- 1 TB of data transfer out
- 10,000,000 HTTP or HTTPS requests
- 2,000,000 CloudFront Function invocations
- Each month, always free

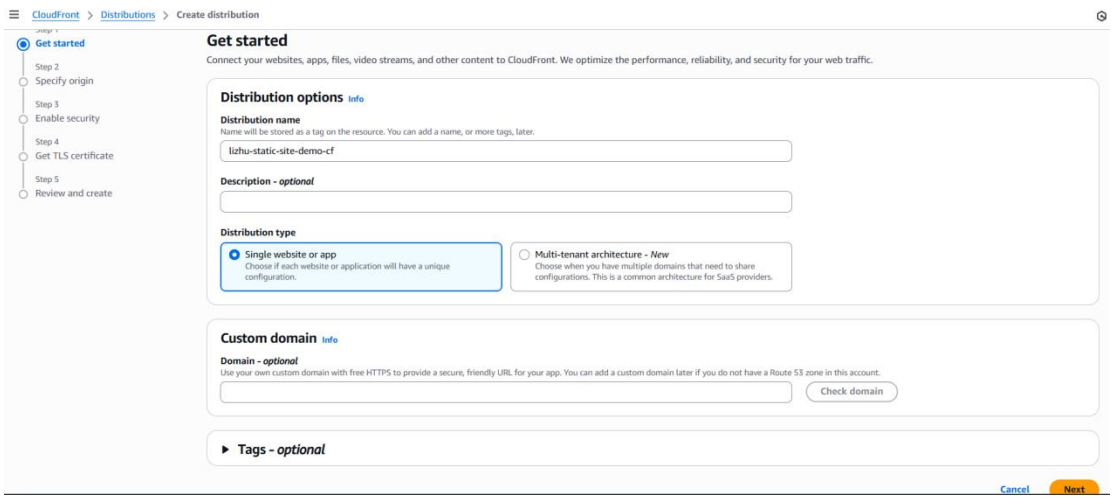
Benefits and features

Reduce latency

The CloudFront network has 225+ points of presence (PoPs) that are connected by fully redundant, parallel 100 GbE fiber delivering ultra-low latency performance

Improve security

Use CloudFront for perimeter protection, traffic encryption, and access controls. AWS Shield Standard defends traffic transmitted through CloudFront from



Specify origin

Origin type

Your origin is where your content (such as a website or app) lives. CloudFront works with AWS-based origins and origins hosted on other cloud providers.

- **Amazon S3** — Deliver static assets like files and images, statically generated websites or single page applications (SPA).
- **Elastic Load Balancer** — Deliver applications hosted behind ELB such as dynamic websites, web services, and APIs.
- **API Gateway** — Deliver API endpoints for REST APIs hosted on API Gateway.
- **Elemental MediaPackage** — Deliver end-to-end live events or video on demand (VOD).
- **VPC origin** — Deliver applications and content hosted within private VPCs, such as EC2 instances and Application Load Balancers.
- **Other** — Refer to any AWS or non-AWS origin through its publicly resolvable URL.

Origin

S3 origin

Choose an AWS origin, or enter your origin's domain name. Learn more

lizhu-static-site-demo-s3.s3-website-us-east-1.amazonaws.com

Browse S3

Origin path - optional

The directory path within your origin where your content is stored. Learn more

/path

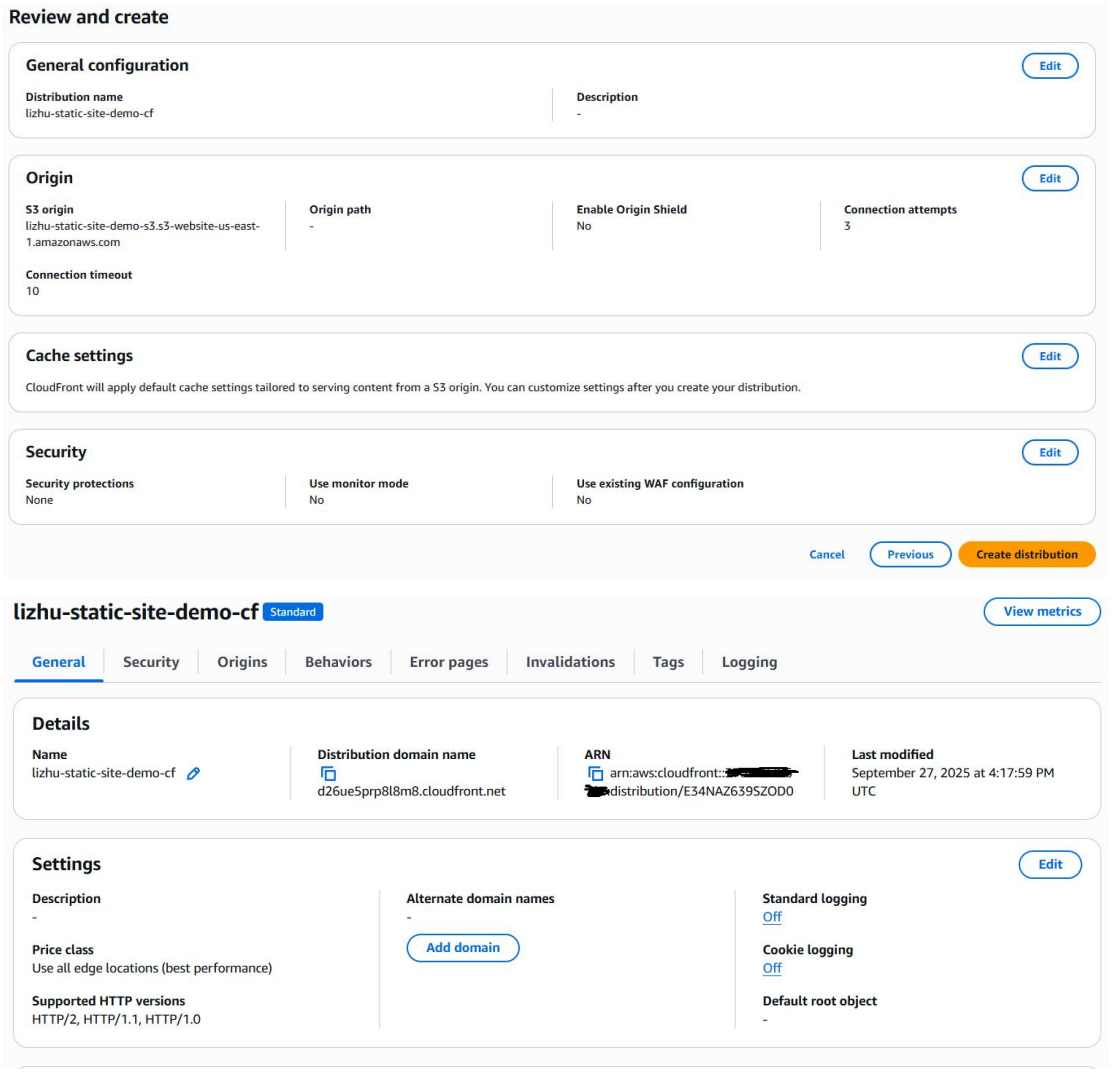
Settings Info

CloudFront provides default origin and cache settings based on what origin you selected. View default settings for S3

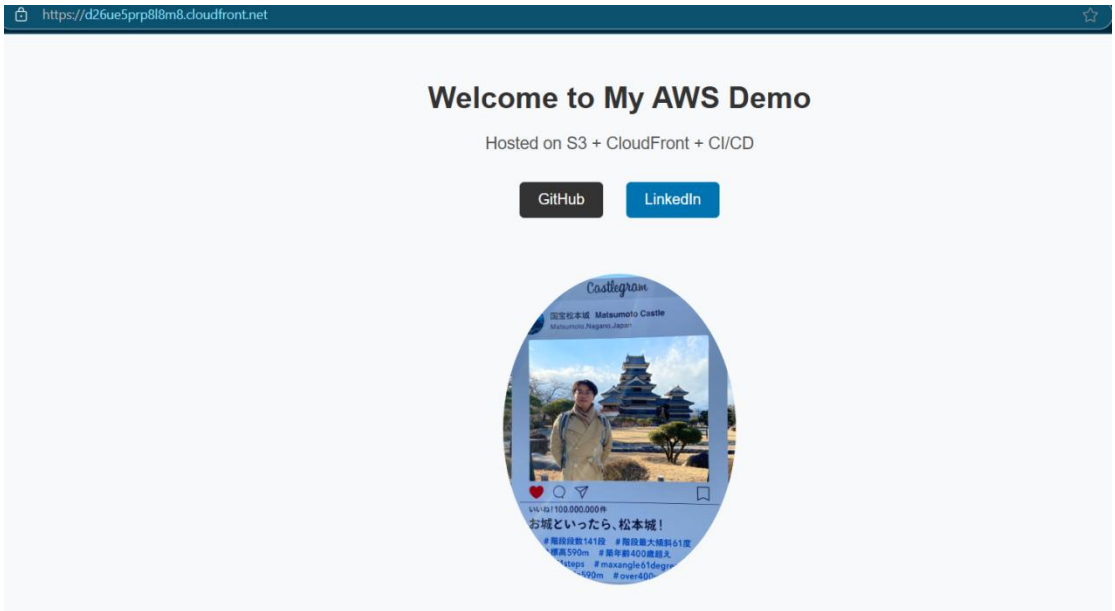
Origin settings

Origin settings control how CloudFront connects to the specified origin.

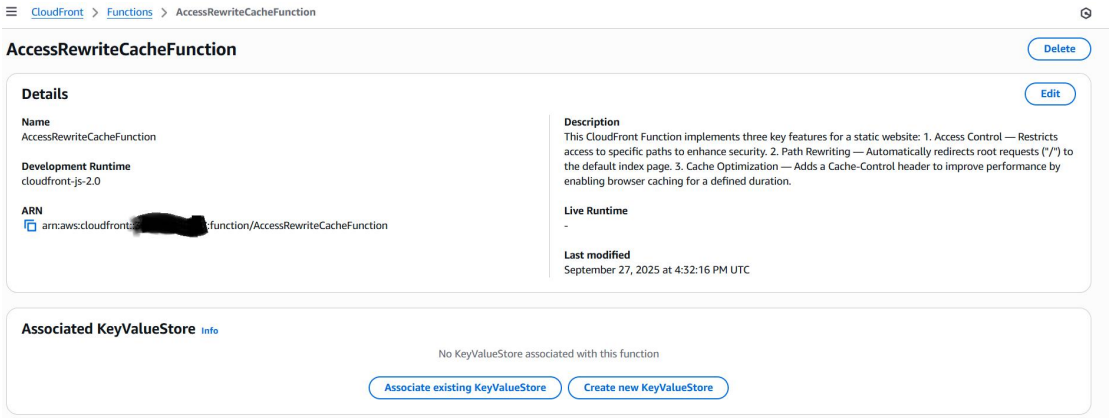
- Use recommended origin settings
- Customize origin settings

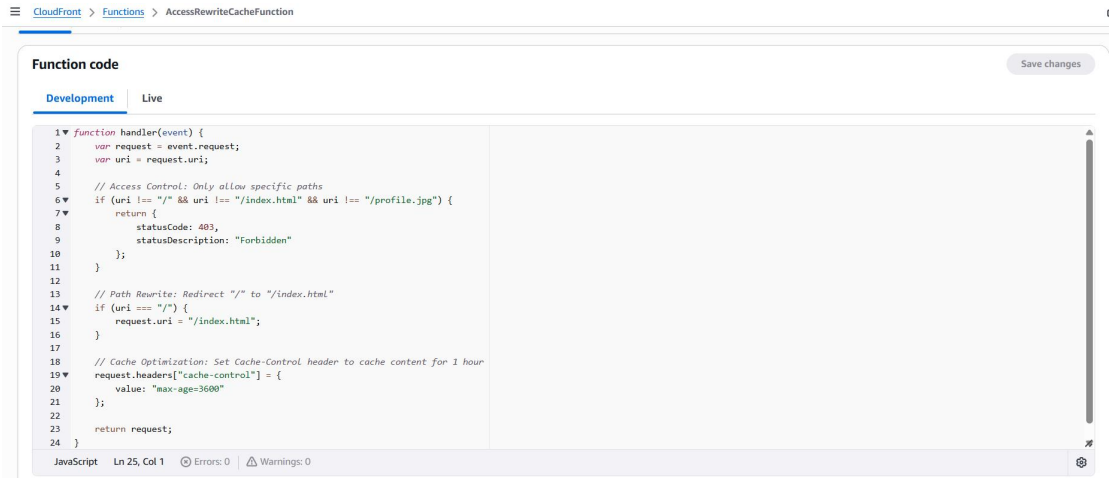


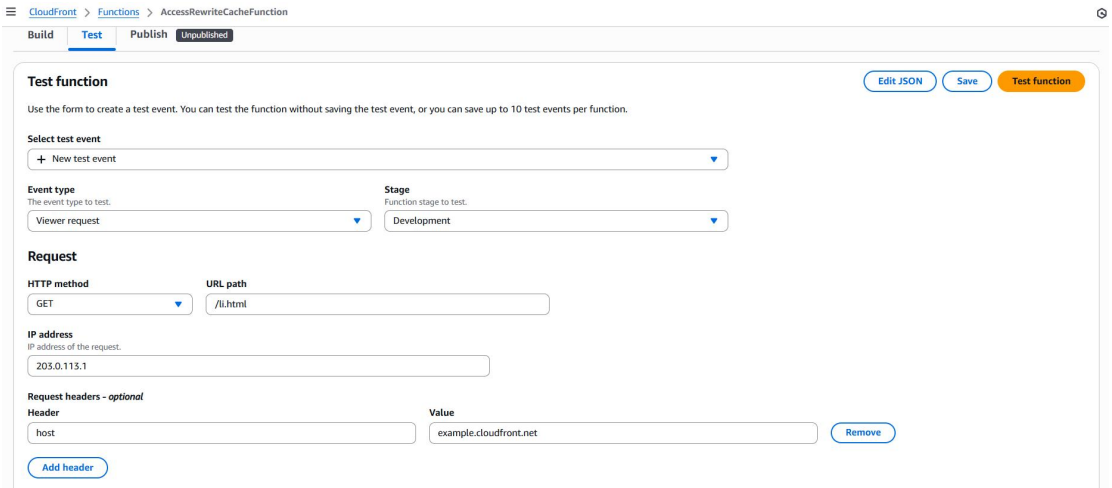
Test site access via the CloudFront distribution domain.

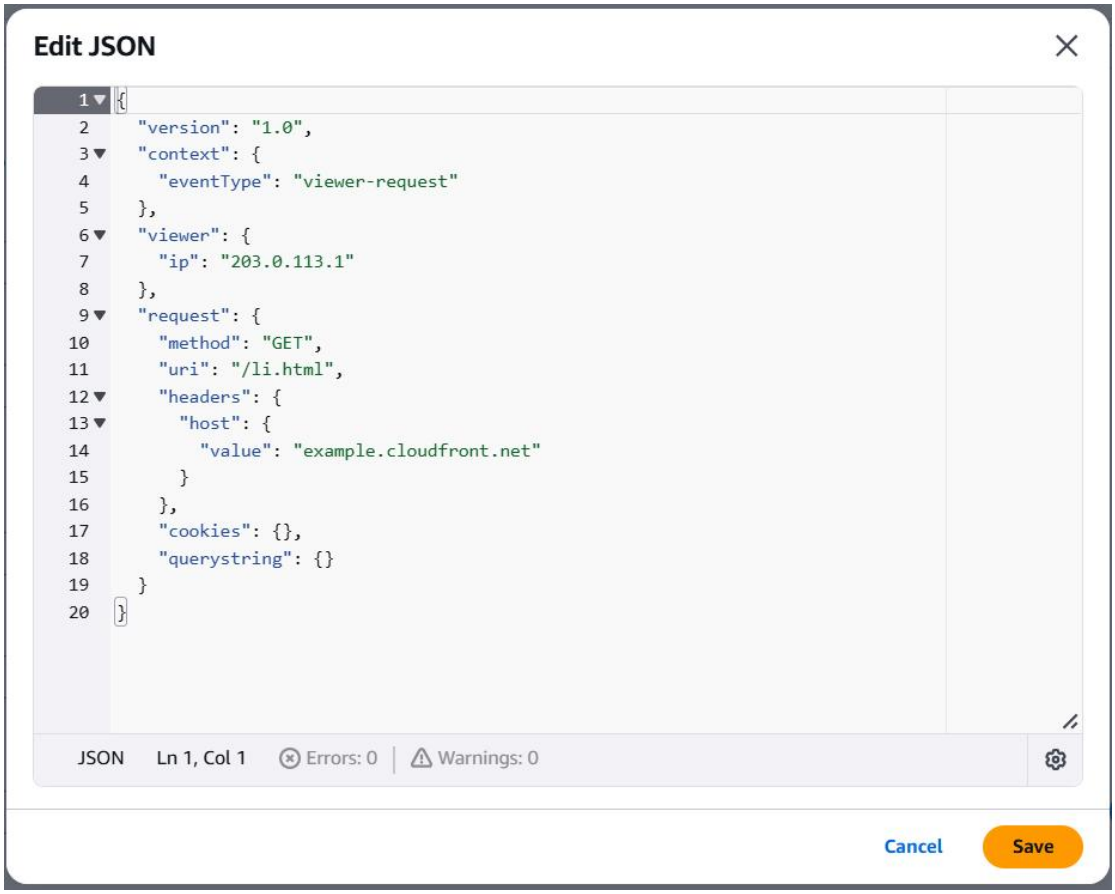


Add three CloudFront Functions:
- Path rewriting
- Simple access control
- Caching optimization

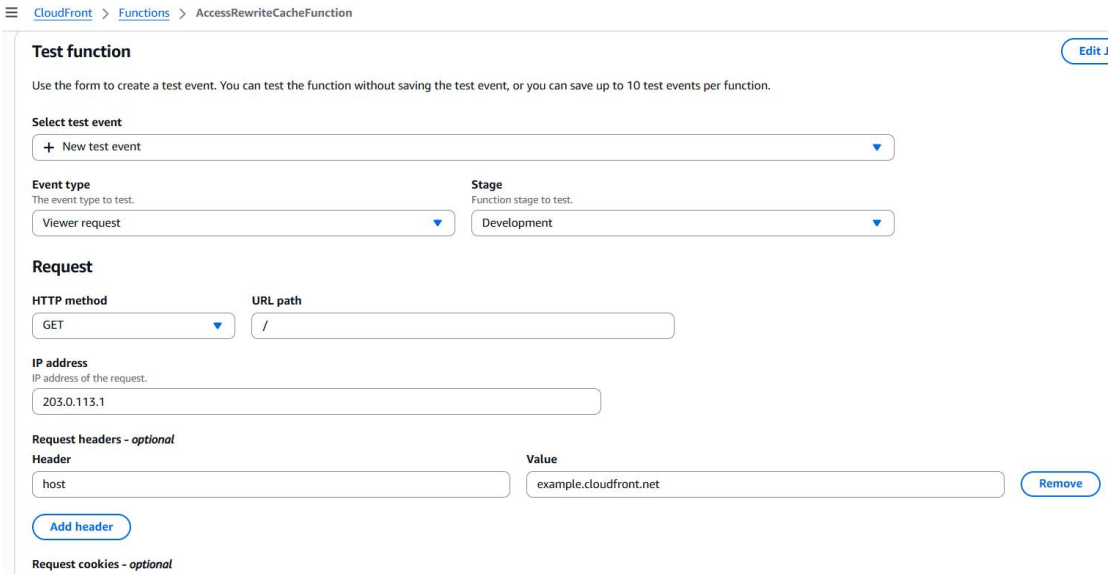




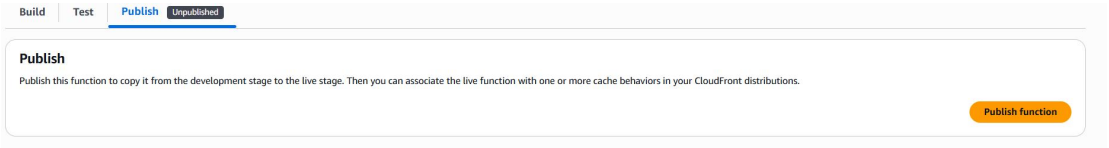


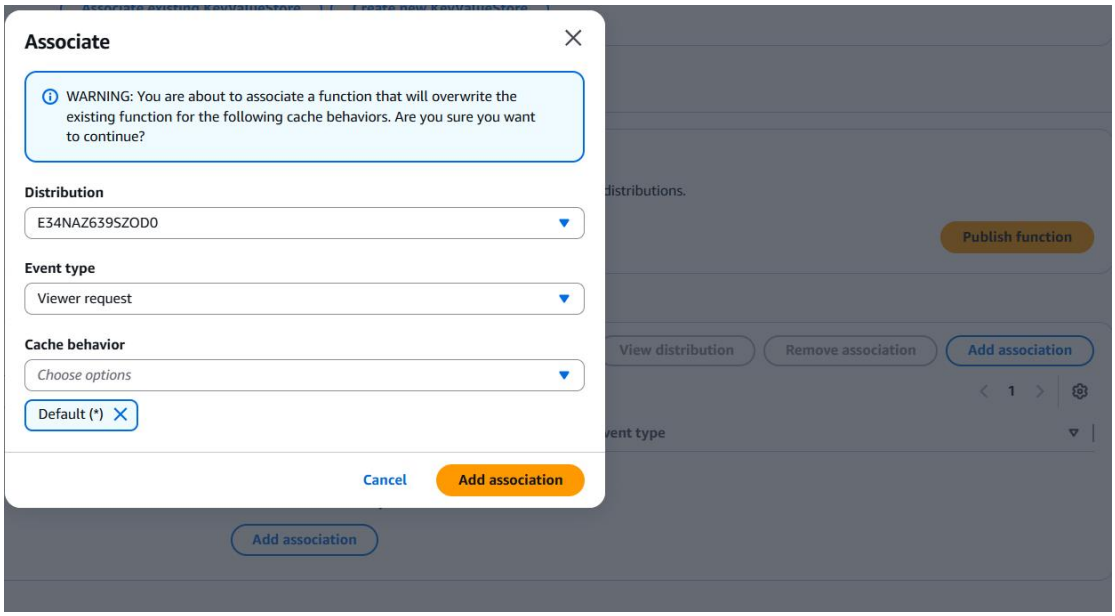

Edit JSON
{
"version": "1.0",
"context": {
"eventType": "viewer-request"
},
"viewer": {
"ip": "203.0.113.1"
},
"request": {
"method": "GET",
"uri": "/li.html",
"headers": {
"host": {
"value": "example.cloudfront.net"
}
},
"cookies": {},
"querystring": {}
}
}
JSON
Ln 1, Col 1
Errors: 0
Warnings: 0
Cancel
Save

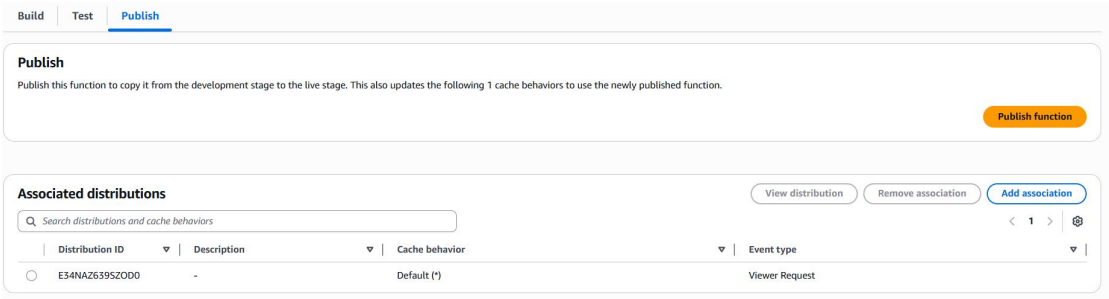

Execution result
Status
403 Forbidden
Stage
DEVELOPMENT
Compute utilization Info
4
Output
{
"response": {
"statusCode": 403,
"statusDescription": "Forbidden",
"headers": {},
"cookies": {}
}
}
Execution logs


CloudFront > Functions > AccessRewriteCacheFunction
Test function
Edit
Use the form to create a test event. You can test the function without saving the test event, or you can save up to 10 test events per function.
Select test event
+ New test event
Event type
The event type to test.
Viewer request
Stage
Function stage to test.
Development
Request
HTTP method
GET
URL path
/
IP address
IP address of the request.
203.0.113.1
Request headers - optional
Header
host
Value
example.cloudfront.net
Remove
Add header
Request cookies - optional

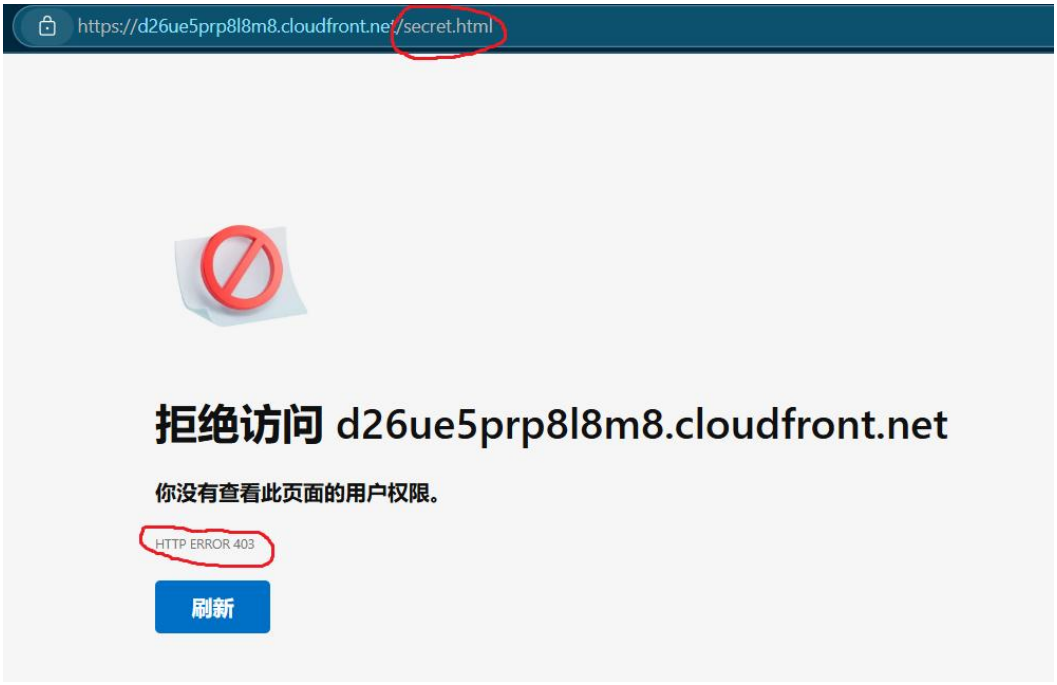

Execution result
Status
Succeeded
Stage
DEVELOPMENT
Compute utilization Info
4
Output
{
"request": {
"method": "GET",
"uri": "/index.html",
"querystring": {},
"headers": {
"cache-control": {
"value": "max-age=3600"
},
"host": {
Execution logs

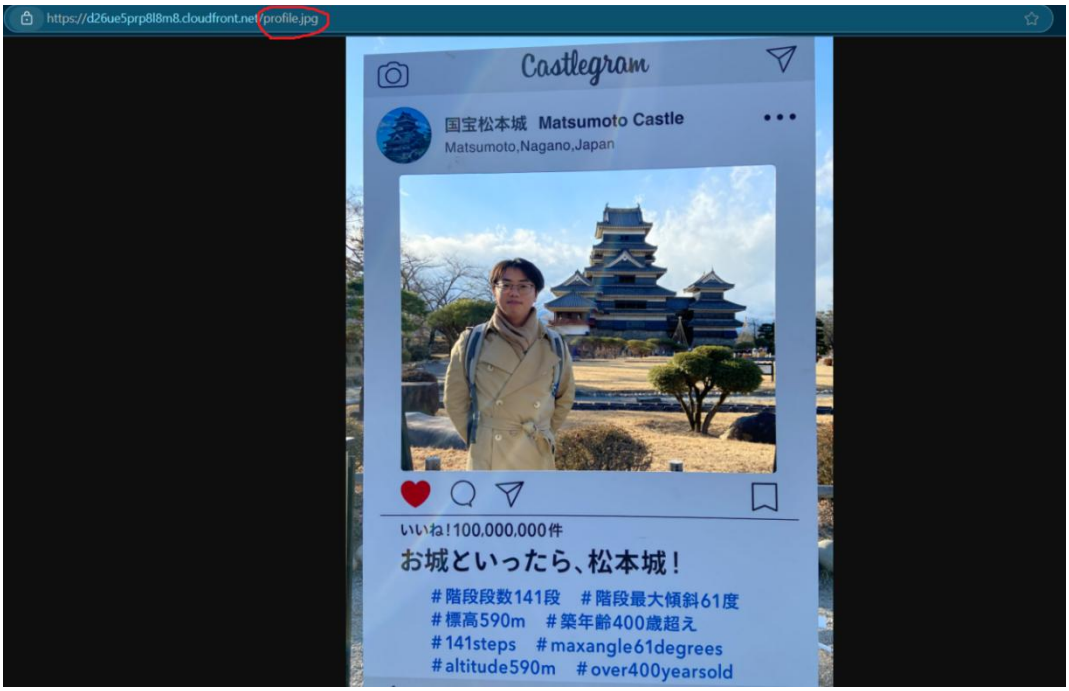

Build
Test
Publish
Unpublished
Publish
Publish this function to copy it from the development stage to the live stage. Then you can associate the live function with one or more cache behaviors in your CloudFront distributions.
Publish function


Associate
WARNING: You are about to associate a function that will overwrite the existing function for the following cache behaviors. Are you sure you want to continue?
Distribution
E34NAZ639SZOD0
Event type
Viewer request
Cache behavior
Choose options
Default (*)
Cancel
Add association
distributions.
Publish function
View distribution
Remove association
Add association
1
vent type
Add association



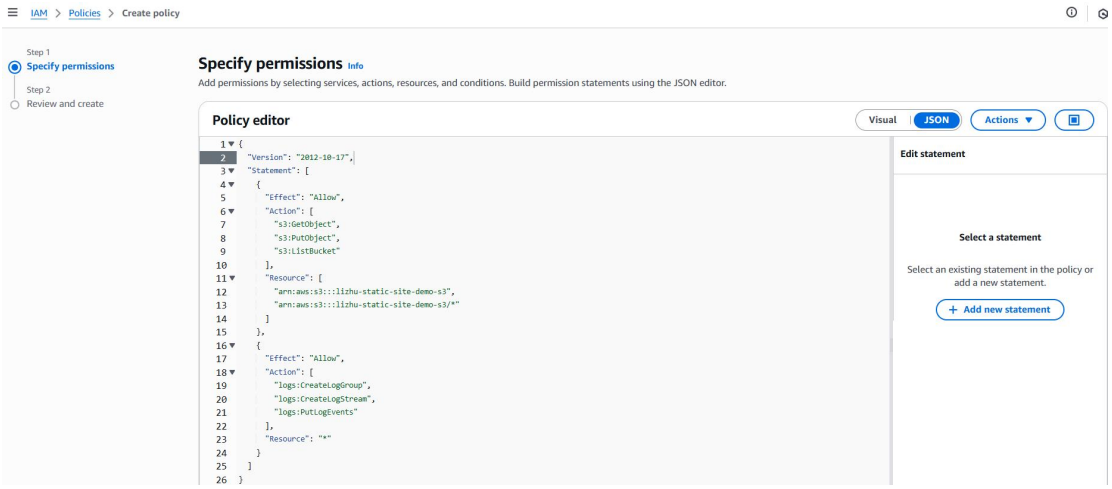
Test the function on the website

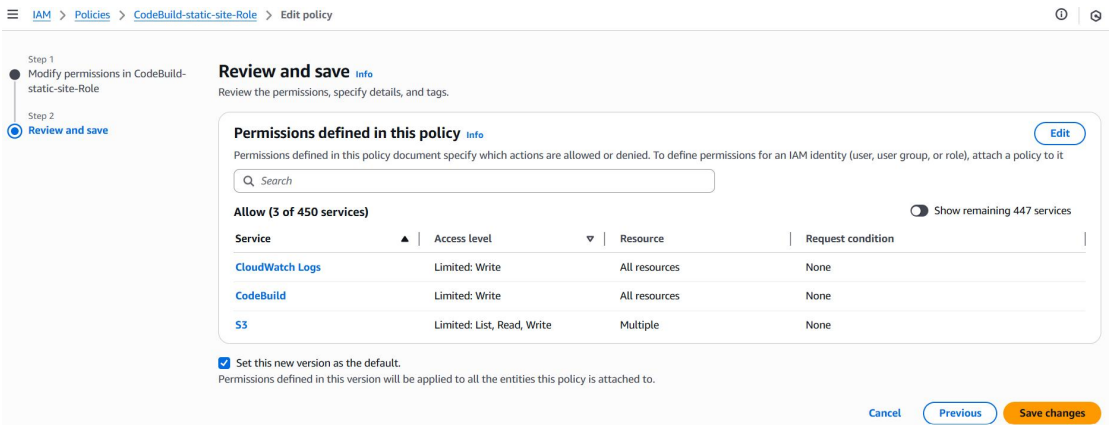




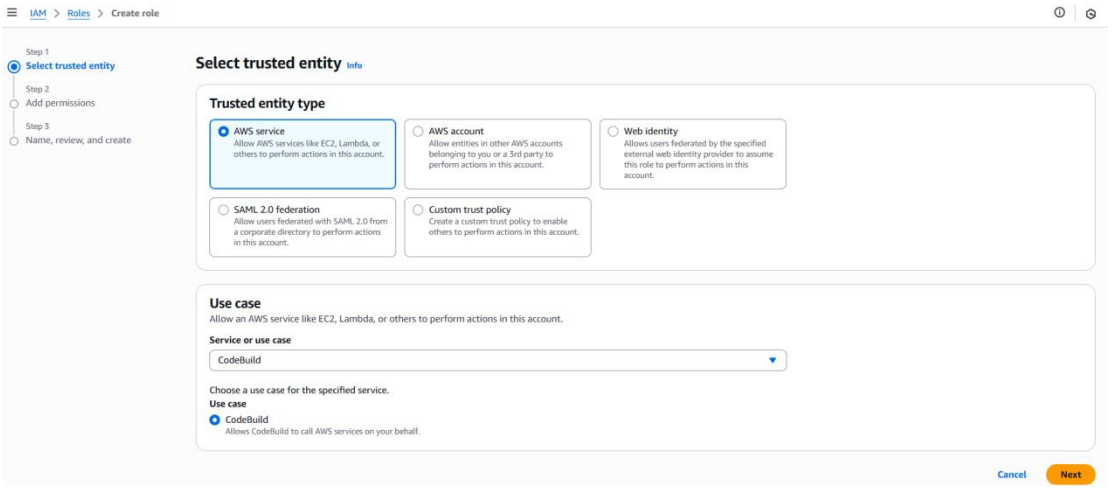
5. Create IAM Policy and CodeBuild Role

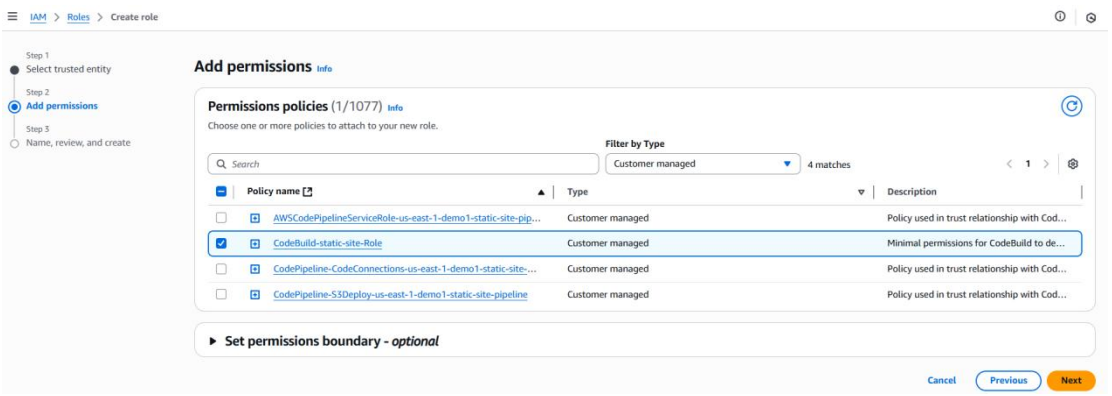
Define a custom IAM policy with least privilege permissions (read/write artifact bucket, full sync with Deploy Bucket, logging, reporting).

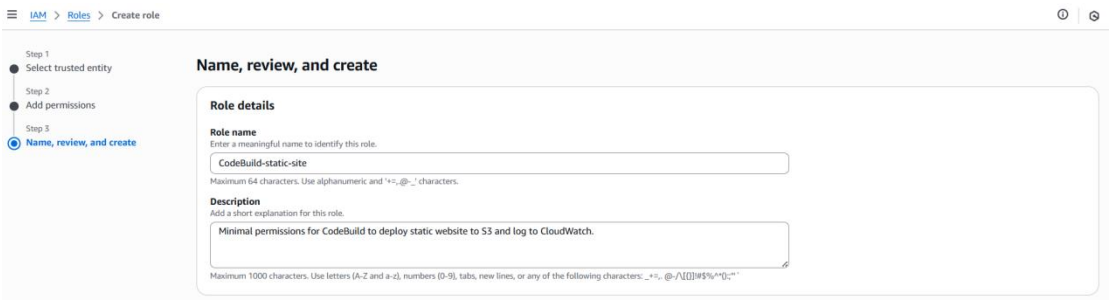




Attach the policy to a new IAM Role (CodeBuild-static-site) for use in the build process.

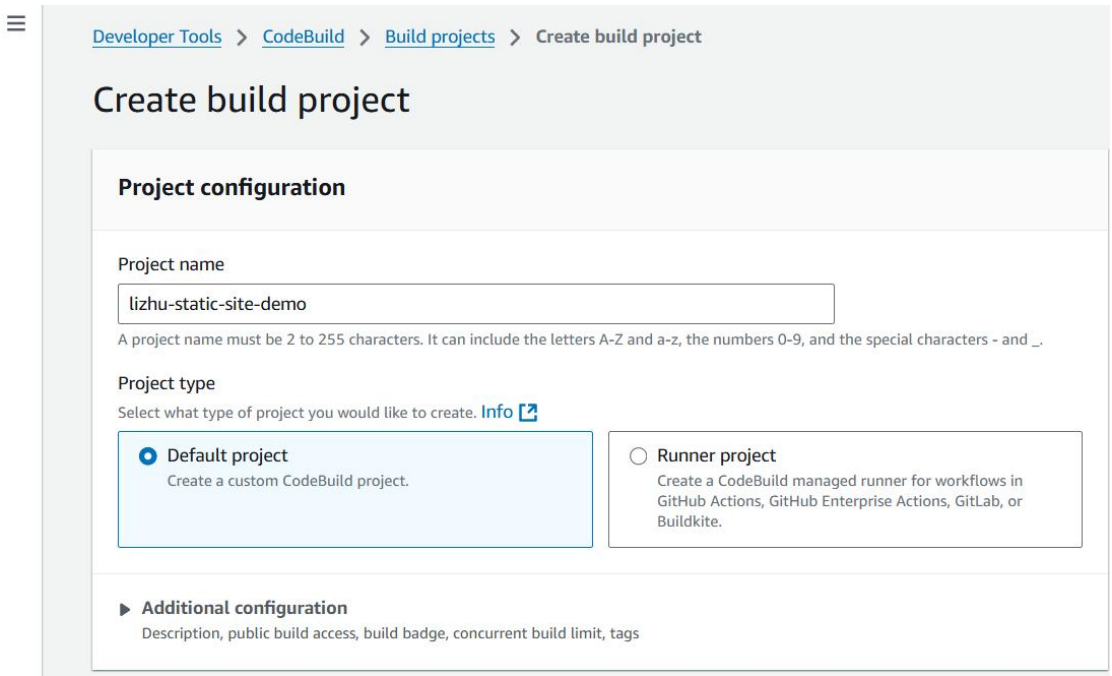






6. Create the CodeBuild Project

Create a CodeBuild project with inline build commands.



Source 1 - Primary

Source provider

GitHub

Credential

Your account is successfully connected through OAuth using CodeBuild managed token. Manage account credentials.

Use override credentials for this project only

Repository

- Repository in my GitHub account
- Public repository
- GitHub scoped webhook

Repository URL

https://github.com/lizh1994/aws-static-site-cicd-lizhu

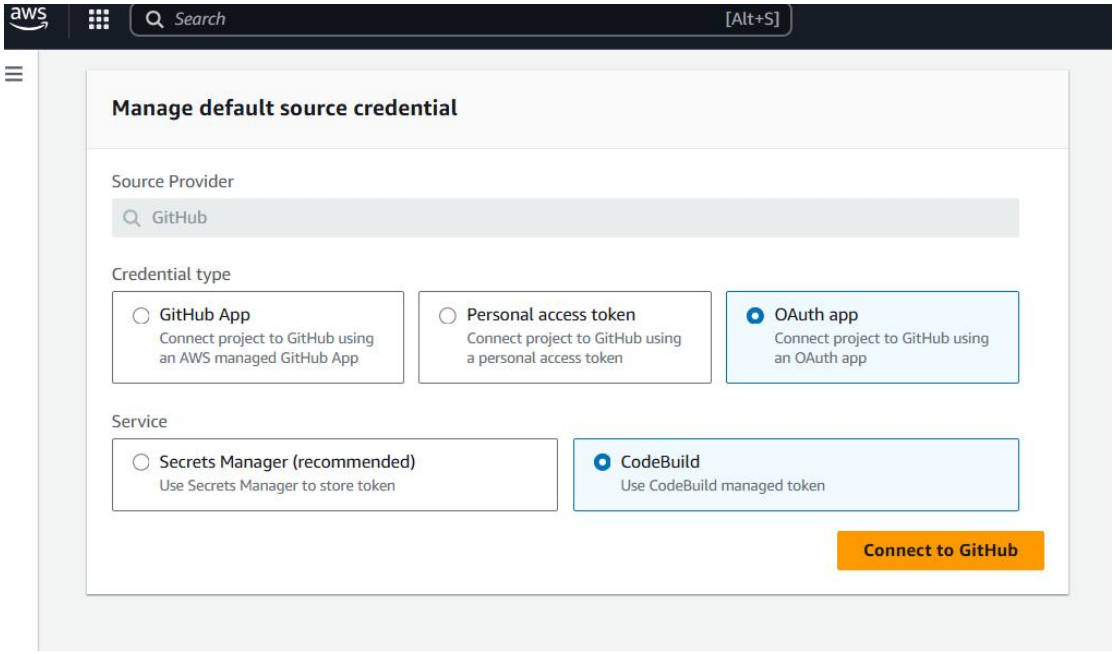
https://github.com/<user-name>/<repository-name>

Source version - *optional* Info

Enter a pull request, branch, commit ID, tag, or reference and a commit ID.

▶ **Additional configuration**

Git clone depth, Git submodules



Environment

Provisioning model Info

- (●) On-demand
  Automatically provision build infrastructure in response to new builds.
- ( ) Reserved capacity
  Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

- (●) Managed image
  Use an image managed by AWS CodeBuild
- ( ) Custom image
  Specify a Docker image

Compute

- (●) EC2
  Optimized for flexibility during action runs
- ( ) Lambda
  Optimized for speed and minimizes the start up time of workflow actions

Running mode

- (●) Container
  Running on Docker container
- ( ) Instance
  Running on EC2 instance directly

Search [Alt+S]

Runtime(s)

Standard

Image

aws/codebuild/amazonlinux-x86_64-standard:5.0

Image version

Always use the latest image for this runtime version

[ ] Use GPU-enhanced compute

Service role

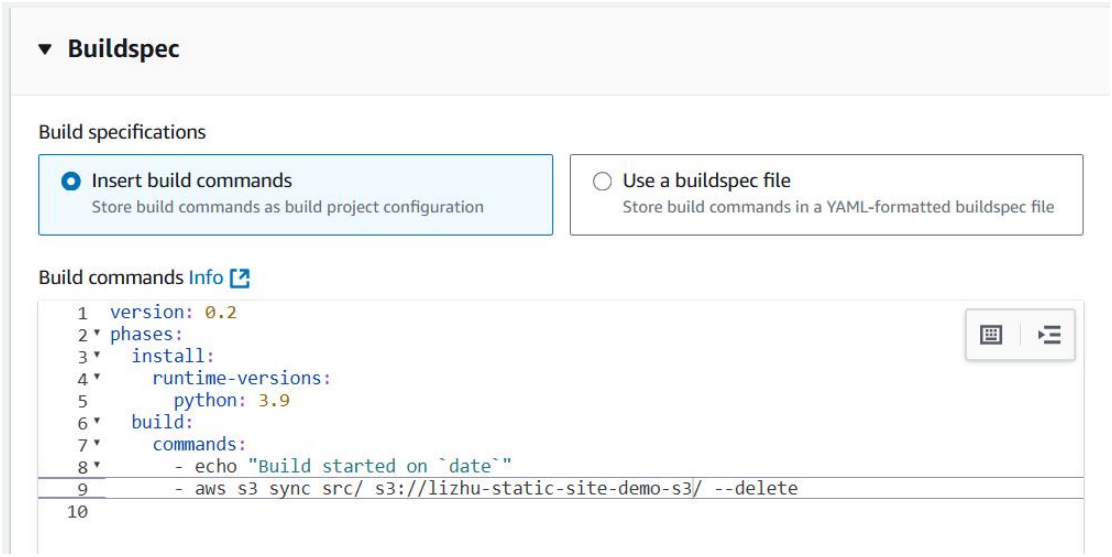
- ( ) New service role
  Create a service role in your account
- (●) Existing service role
  Choose an existing service role from your account

Role ARN

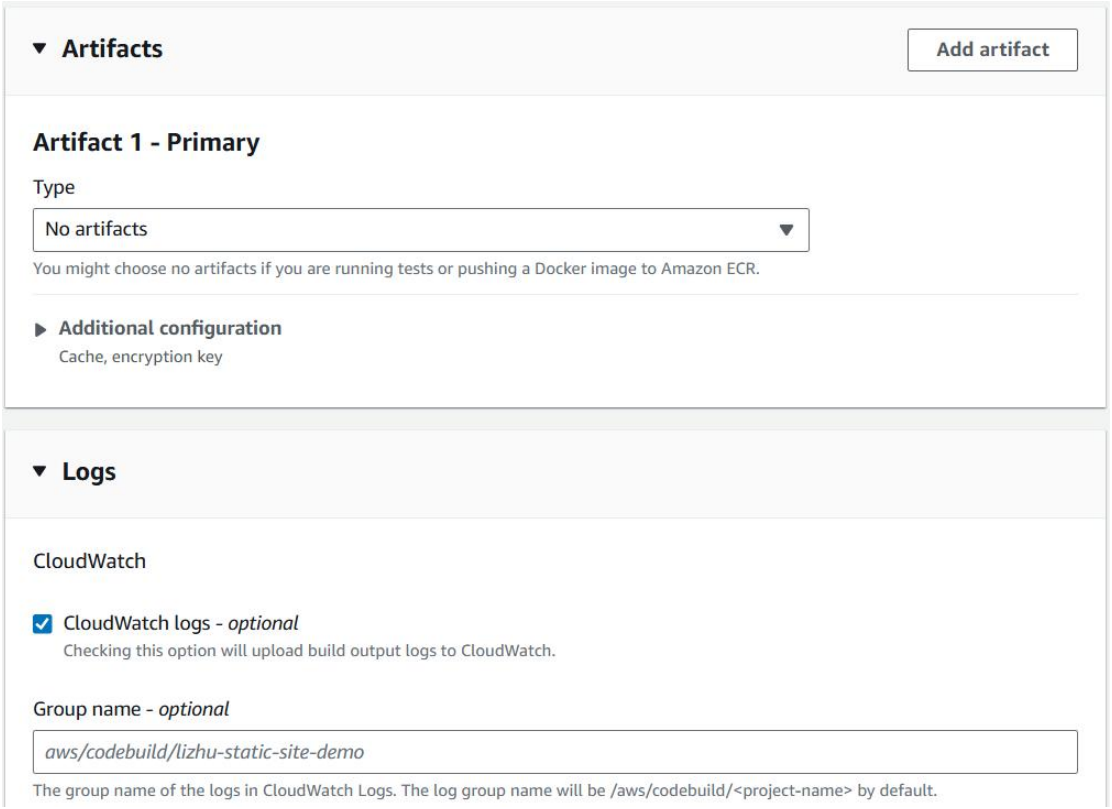
arn:aws:iam::[redacted]:role/CodeBuild-static-site

[ ] Allow AWS CodeBuild to modify this service role so it can be used with this build project.

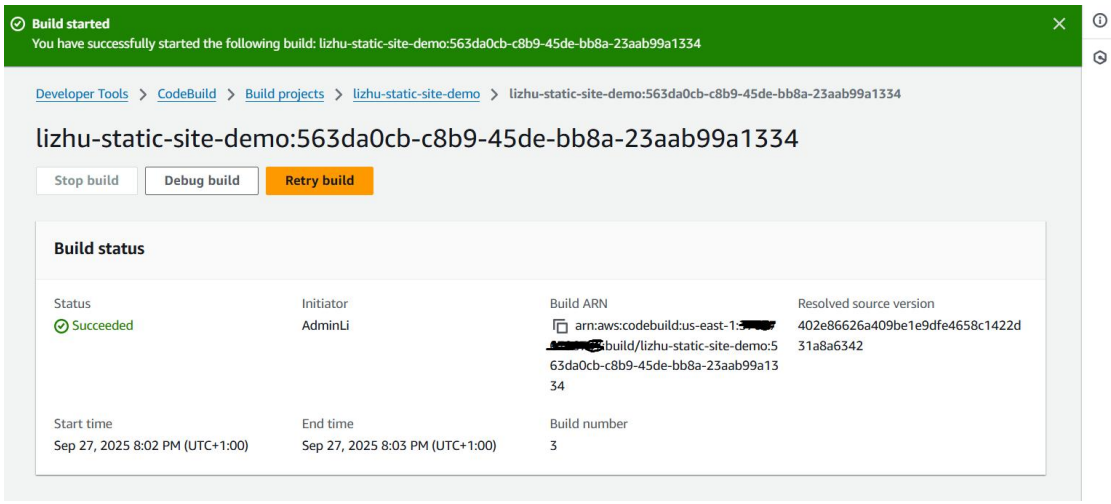
▶ **Additional configuration**
Timeout, privileged, certificate, VPC, compute type, environment variables, file systems, auto-retry, registry credential

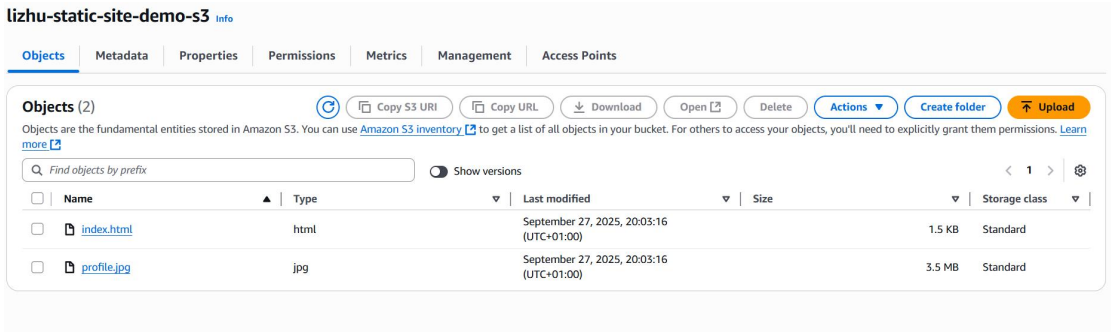


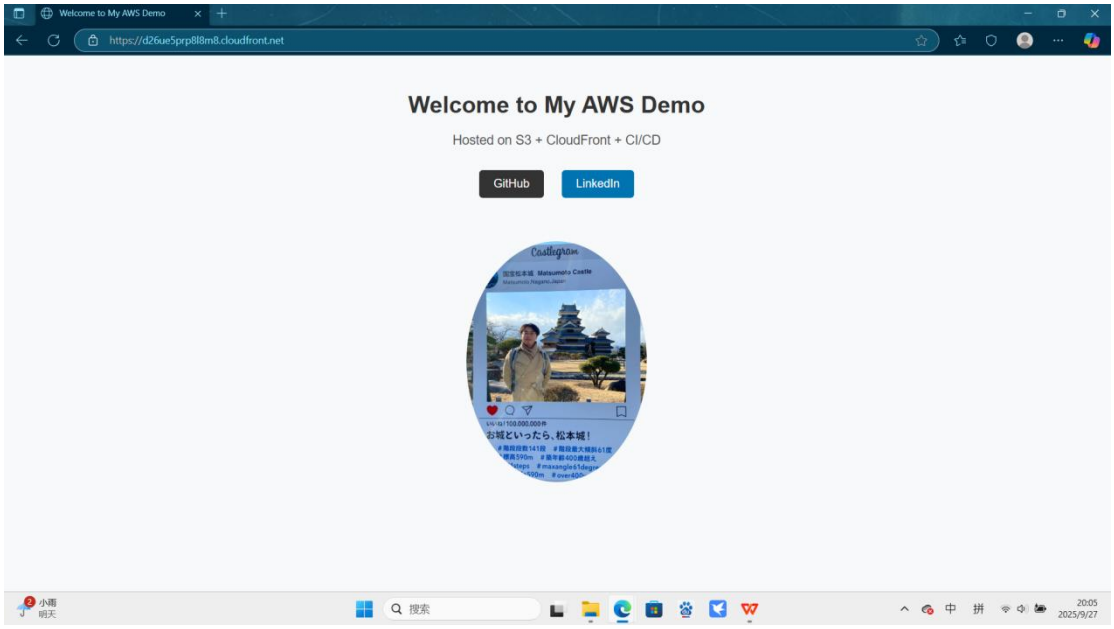
Initially, no artifact output was configured.



Test the build: confirmed that updated files were written to the Deploy Bucket.

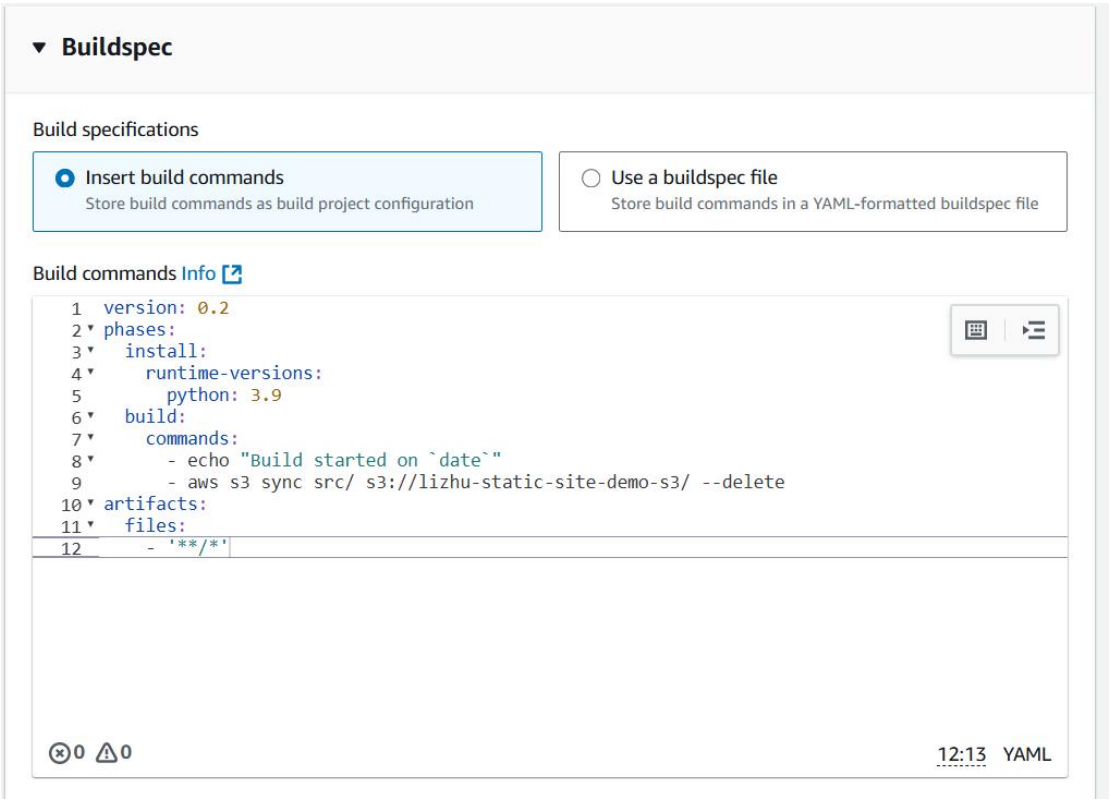




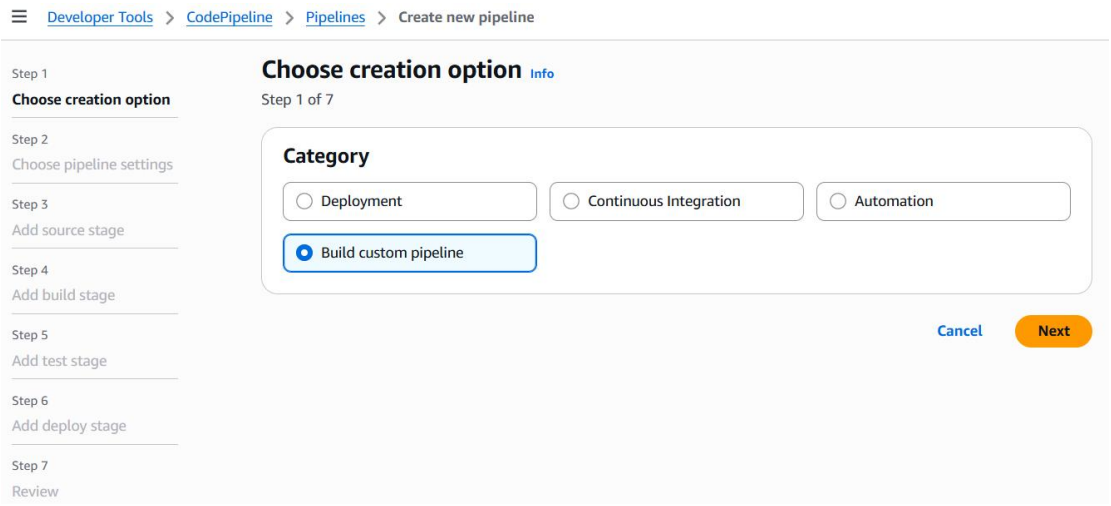


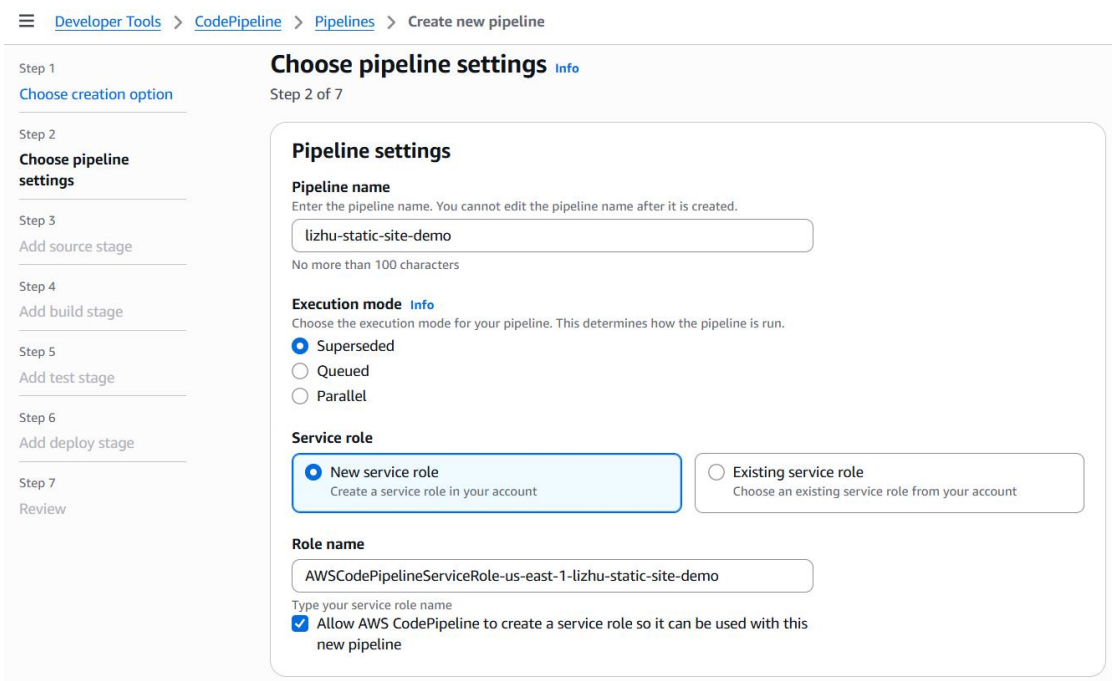
7. Create CodePipeline

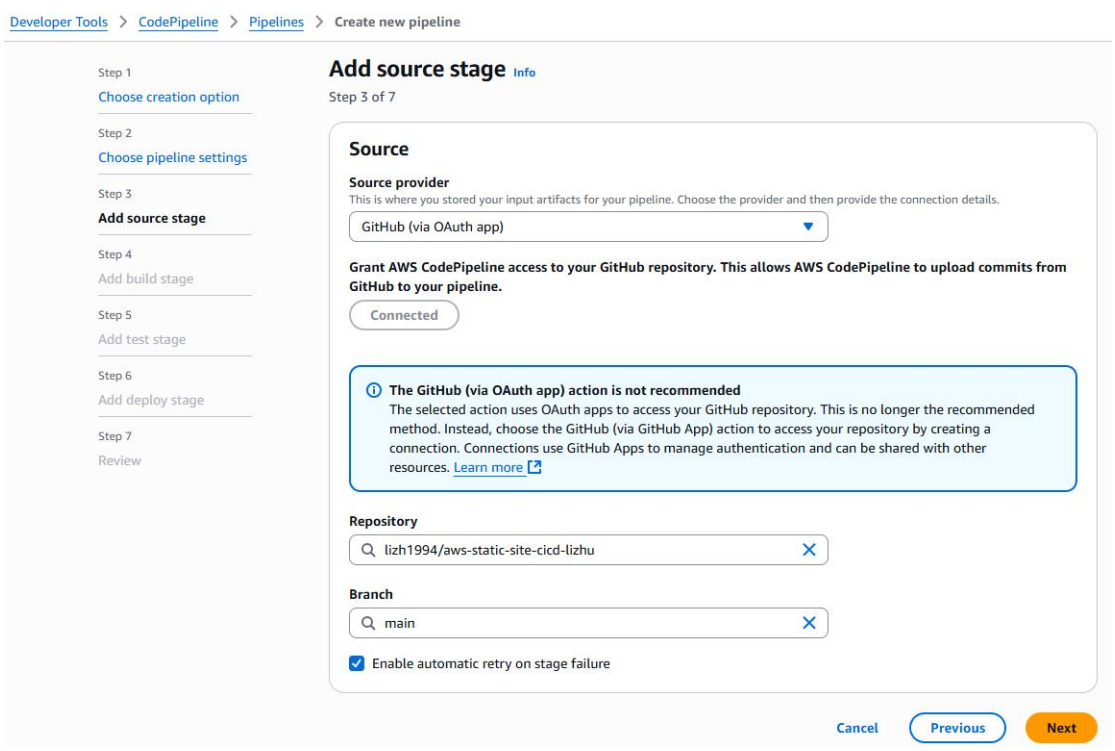
Update the CodeBuild project with build commands that generate build artifacts for the Deploy stage.

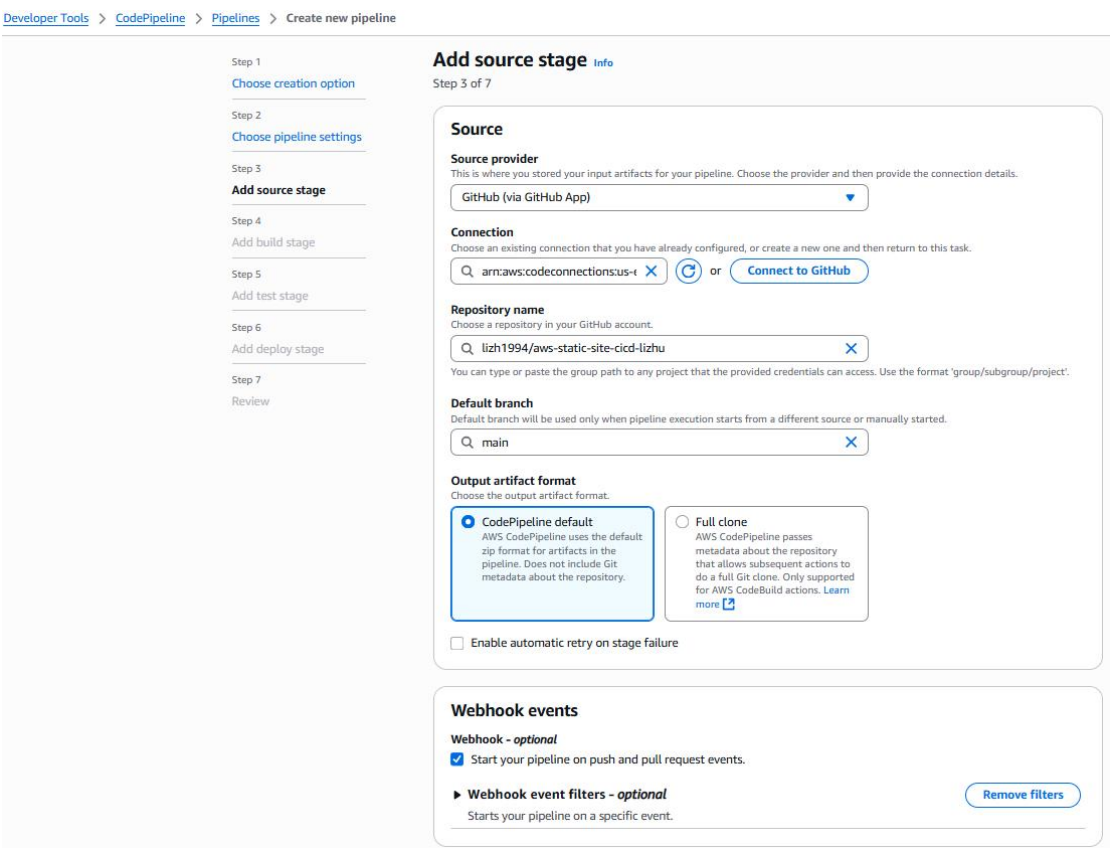


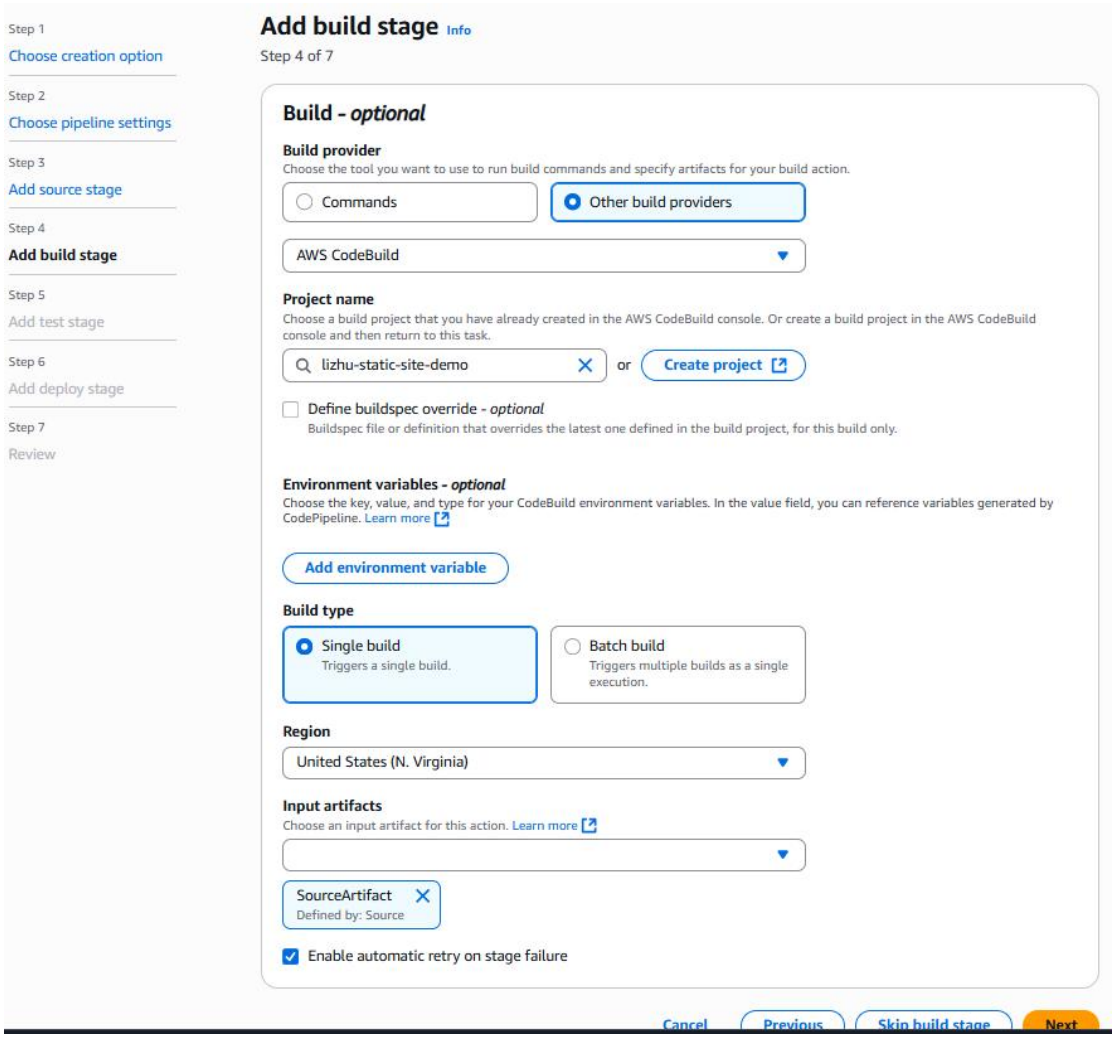
Create the CodePipeline with three stages:
- Source (GitHub) - Trigger: push events.
- Build (CodeBuild) - Uses the updated CodeBuild project.
- Deploy (S3) - Deploys artifacts to the Deploy Bucket.

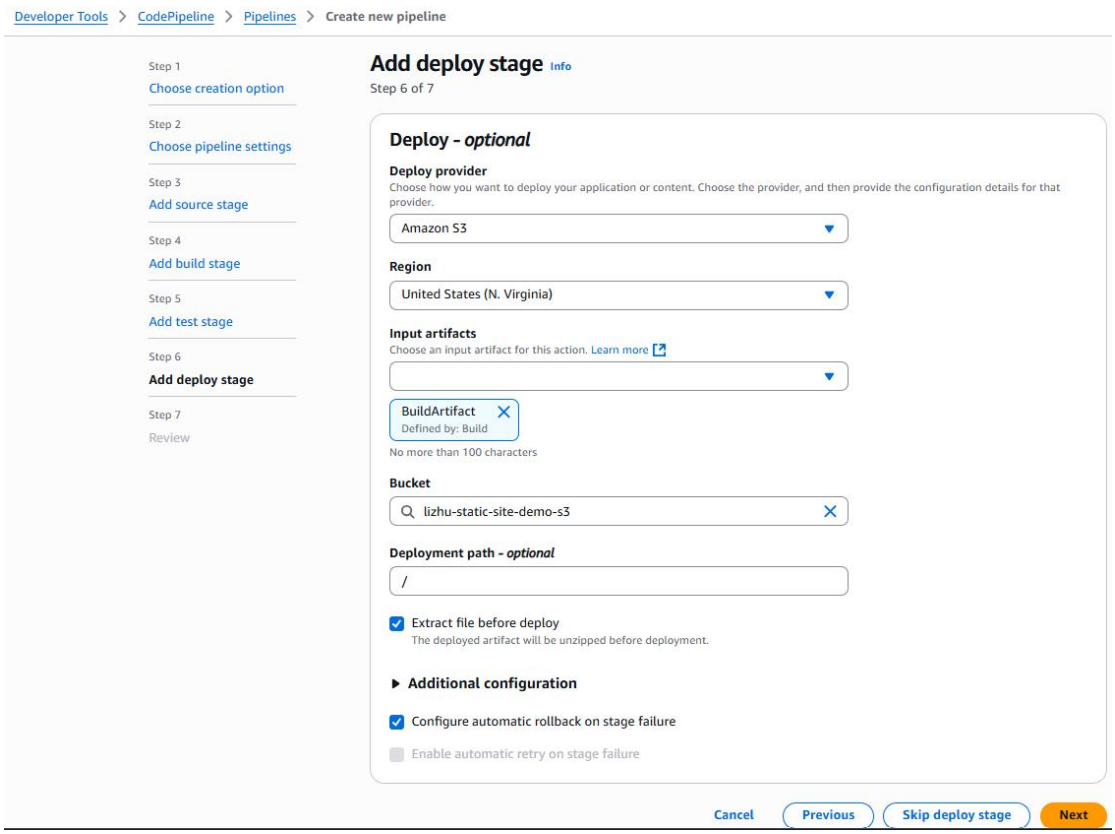


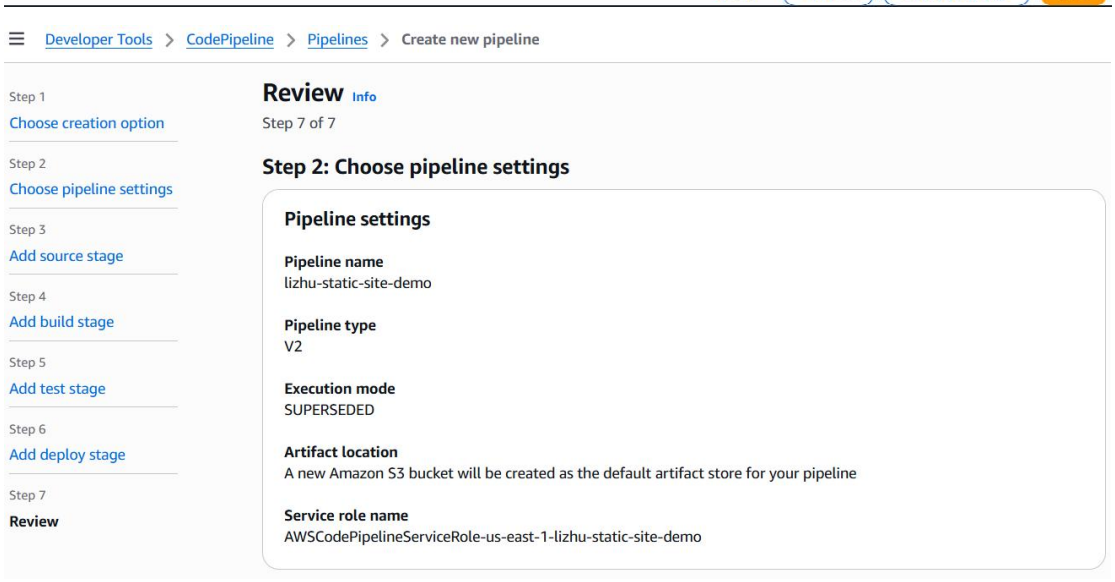

Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Choose pipeline settings Info
Step 2 of 7
Pipeline settings
Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.
lizhu-static-site-demo
No more than 100 characters
Execution mode Info
Choose the execution mode for your pipeline. This determines how the pipeline is run.
Superseded
Queued
Parallel
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role name
AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo
Type your service role name
Allow AWS CodePipeline to create a service role so it can be used with this new pipeline


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add source stage Info
Step 3 of 7
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (via OAuth app)
Grant AWS CodePipeline access to your GitHub repository. This allows AWS CodePipeline to upload commits from GitHub to your pipeline.
Connected
The GitHub (via OAuth app) action is not recommended
The selected action uses OAuth apps to access your GitHub repository. This is no longer the recommended method. Instead, choose the GitHub (via GitHub App) action to access your repository by creating a connection. Connections use GitHub Apps to manage authentication and can be shared with other resources. Learn more
Repository
lizh1994/aws-static-site-cicd-lizhu
Branch
main
Enable automatic retry on stage failure
Cancel
Previous
Next


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add source stage Info
Step 3 of 7
Source
Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.
GitHub (via GitHub App)
Connection
Choose an existing connection that you have already configured, or create a new one and then return to this task.
arn:aws:codeconnections:us-
or
Connect to GitHub
Repository name
Choose a repository in your GitHub account.
lizh1994/aws-static-site-cicd-lizhu
You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.
Default branch
Default branch will be used only when pipeline execution starts from a different source or manually started.
main
Output artifact format
Choose the output artifact format.
CodePipeline default
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.
Full clone
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. Learn more
Enable automatic retry on stage failure
Webhook events
Webhook - optional
Start your pipeline on push and pull request events.
Webhook event filters - optional
Remove filters
Starts your pipeline on a specific event.


Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add build stage Info
Step 4 of 7
Build - optional
Build provider
Choose the tool you want to use to run build commands and specify artifacts for your build action.
Commands
Other build providers
AWS CodeBuild
Project name
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.
lizhu-static-site-demo
or
Create project
Define buildspec override - optional
Buildspec file or definition that overrides the latest one defined in the build project, for this build only.
Environment variables - optional
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. Learn more
Add environment variable
Build type
Single build
Triggers a single build.
Batch build
Triggers multiple builds as a single execution.
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
SourceArtifact
Defined by: Source
Enable automatic retry on stage failure
Cancel
Previous
Skip build stage
Next


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Add deploy stage Info
Step 6 of 7
Deploy - optional
Deploy provider
Choose how you want to deploy your application or content. Choose the provider, and then provide the configuration details for that provider.
Amazon S3
Region
United States (N. Virginia)
Input artifacts
Choose an input artifact for this action. Learn more
BuildArtifact
Defined by: Build
No more than 100 characters
Bucket
lizhu-static-site-demo-s3
Deployment path - optional
/
Extract file before deploy
The deployed artifact will be unzipped before deployment.
Additional configuration
Configure automatic rollback on stage failure
Enable automatic retry on stage failure
Cancel
Previous
Skip deploy stage
Next


Developer Tools > CodePipeline > Pipelines > Create new pipeline
Step 1
Choose creation option
Step 2
Choose pipeline settings
Step 3
Add source stage
Step 4
Add build stage
Step 5
Add test stage
Step 6
Add deploy stage
Step 7
Review
Review Info
Step 7 of 7
Step 2: Choose pipeline settings
Pipeline settings
Pipeline name
lizhu-static-site-demo
Pipeline type
V2
Execution mode
SUPERSEDED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo

Step 3: Add source stage

Source action provider

Source action provider
GitHub (via GitHub App)

OutputArtifactFormat
CODE_ZIP

DetectChanges
true

ConnectionArn
arn:aws:codeconnections:us-east-1:370970796707:connection/0340bf7e-ef52-49a9-bbaa-b2c4a54e7685

FullRepositoryId
lizh1994/aws-static-site-cicd-lizhu

Default branch
main

Enable automatic retry on stage failure
Disabled

Trigger configuration

You can add additional pipeline triggers after the pipeline is created.

Trigger type
No filter

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 4: Add build stage

Build action provider

Build action provider
AWS CodeBuild

ProjectName
lizhu-static-site-demo

Commands
-

Enable automatic retry on stage failure
Enabled

Step 6: Add deploy stage

Deploy action provider

Deploy action provider
Amazon S3

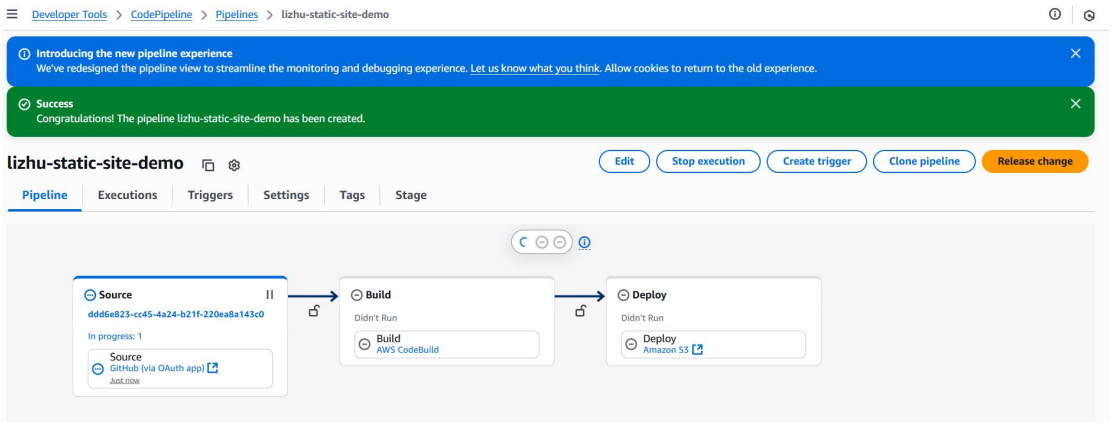
Extract
true

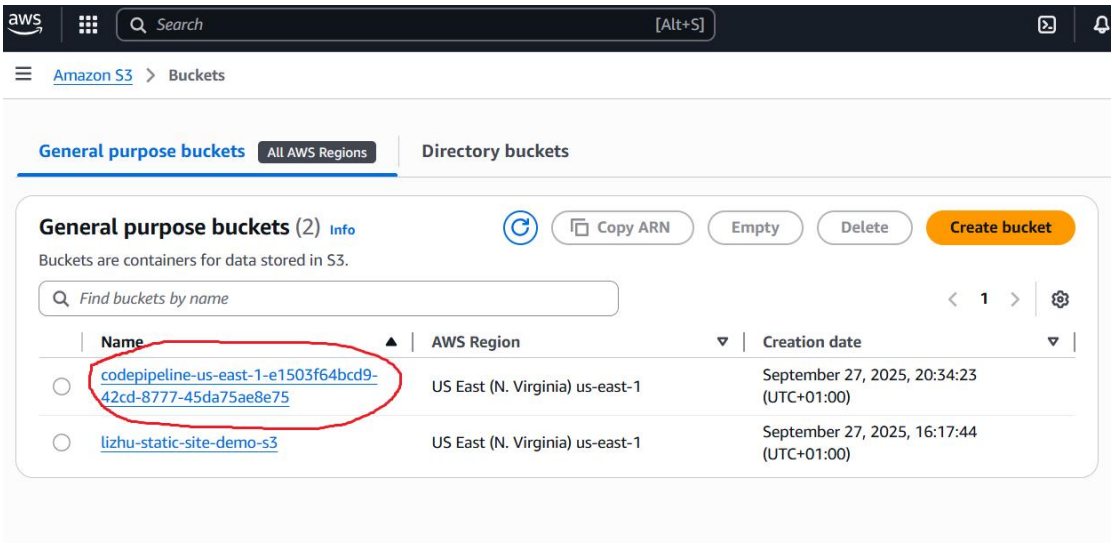
BucketName
lizhu-static-site-demo-s3

ObjectKey
/

Configure automatic rollback on stage failure
Enabled

Enable automatic retry on stage failure
Disabled

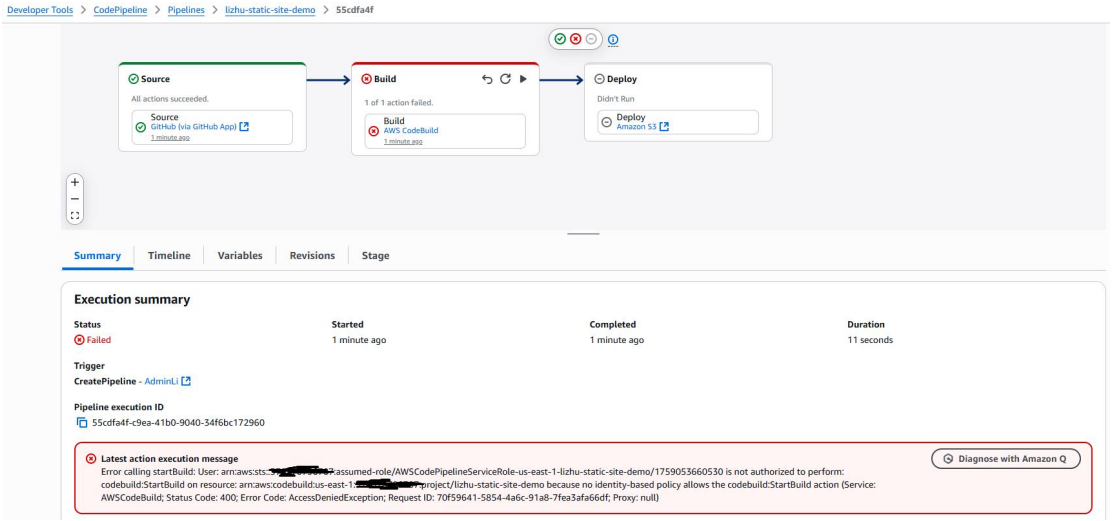




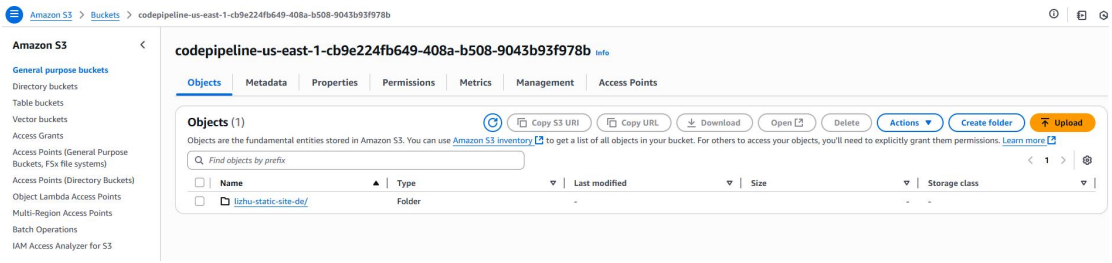
Note: Pipeline Artifact Bucket was automatically created by AWS to handle intermediate artifacts.

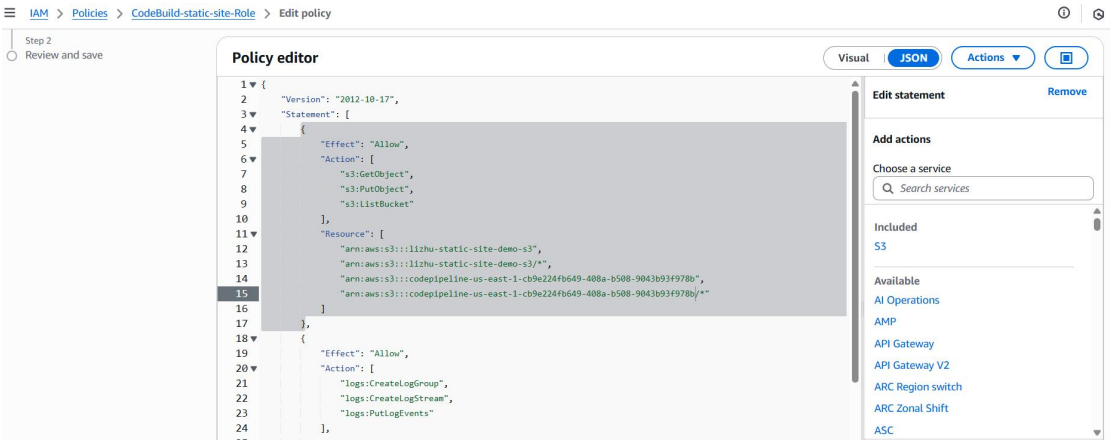
8. Troubleshooting

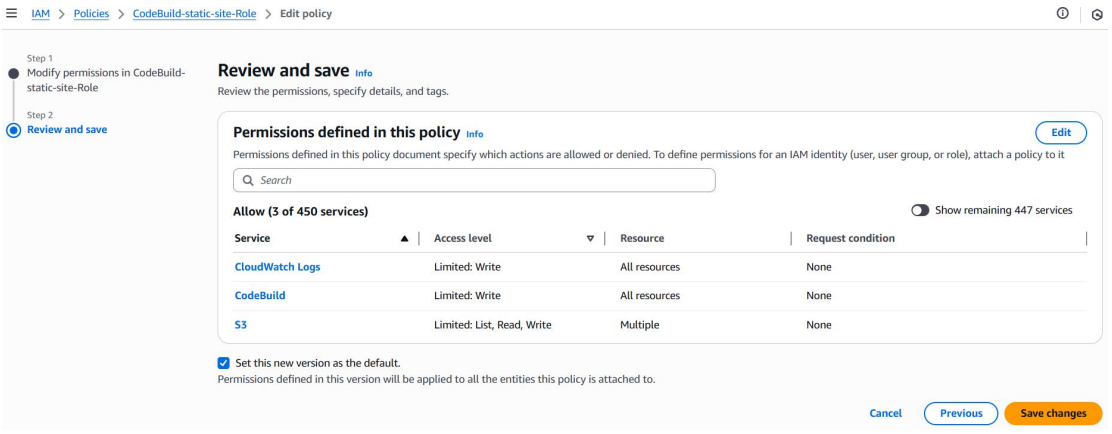
The first pipeline execution will always fail because the pipeline artifact bucket is just created, and the CodeBuild role does not yet have permissions for it.

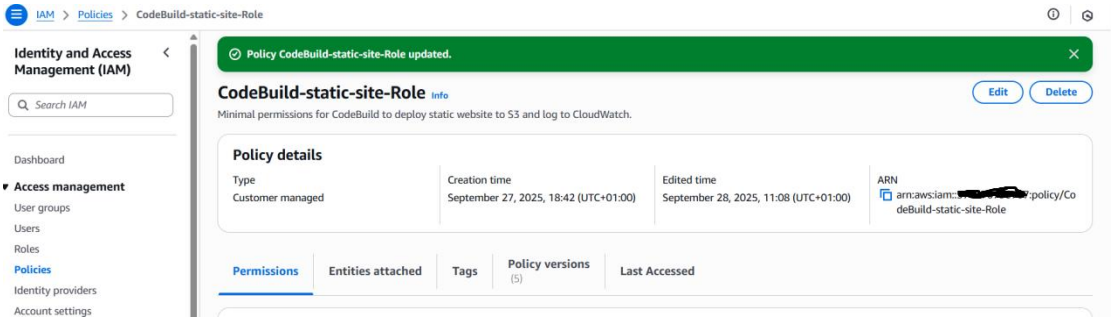


Step 1: Fix Initial Failure
- Go to the CodeBuild IAM Role.
- Add permissions for the newly created pipeline artifact bucket (Read/Write/List).
- Save and re-deploy the pipeline.



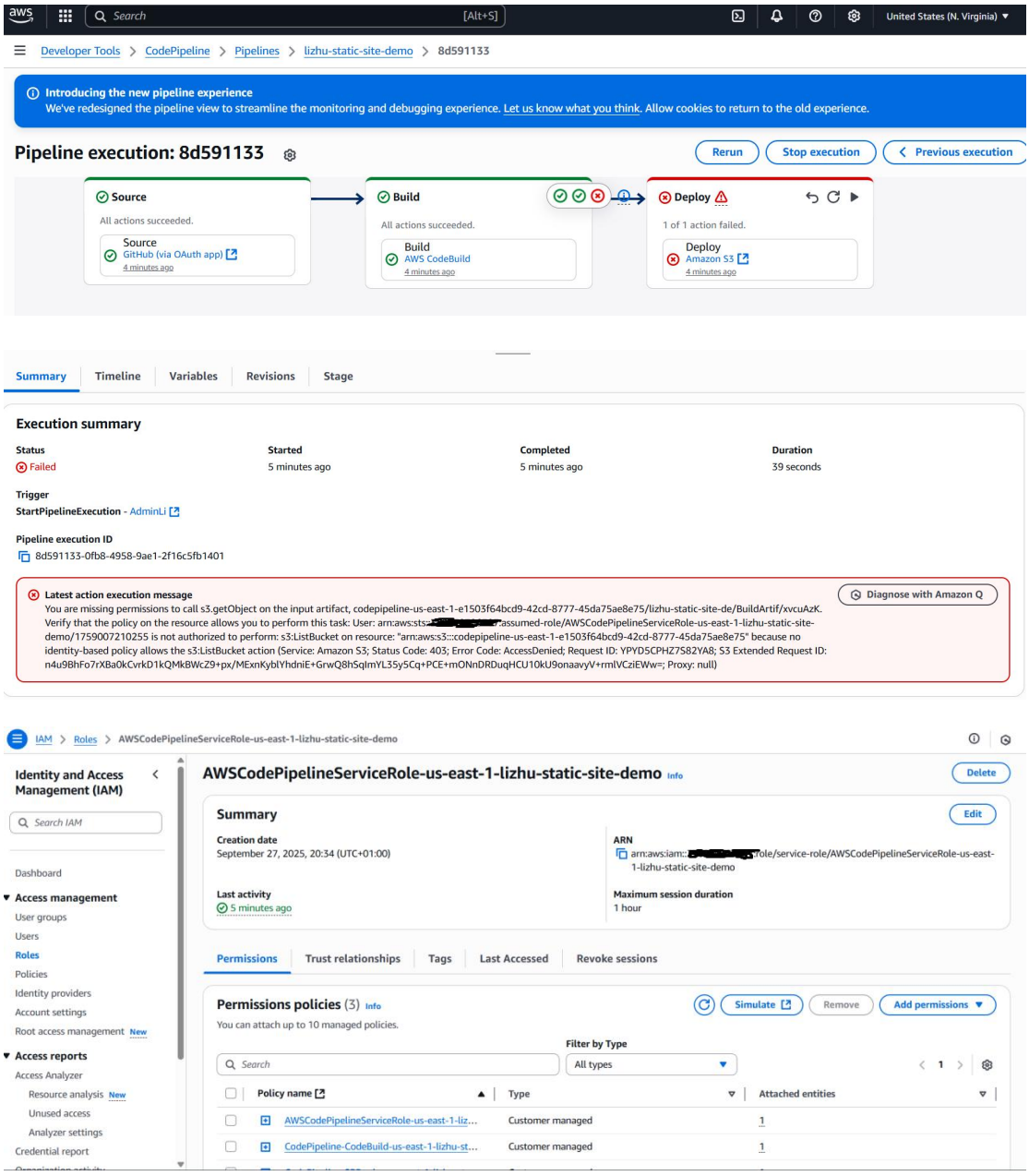


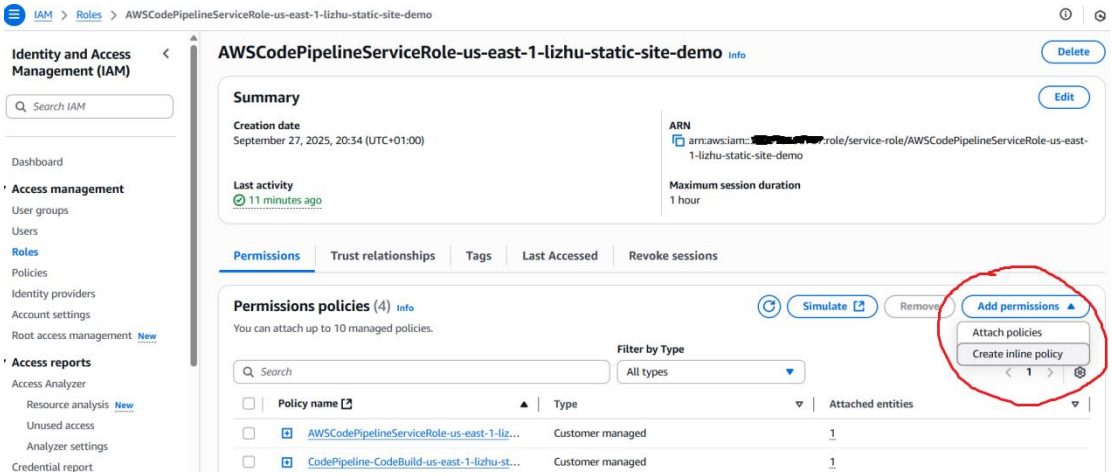


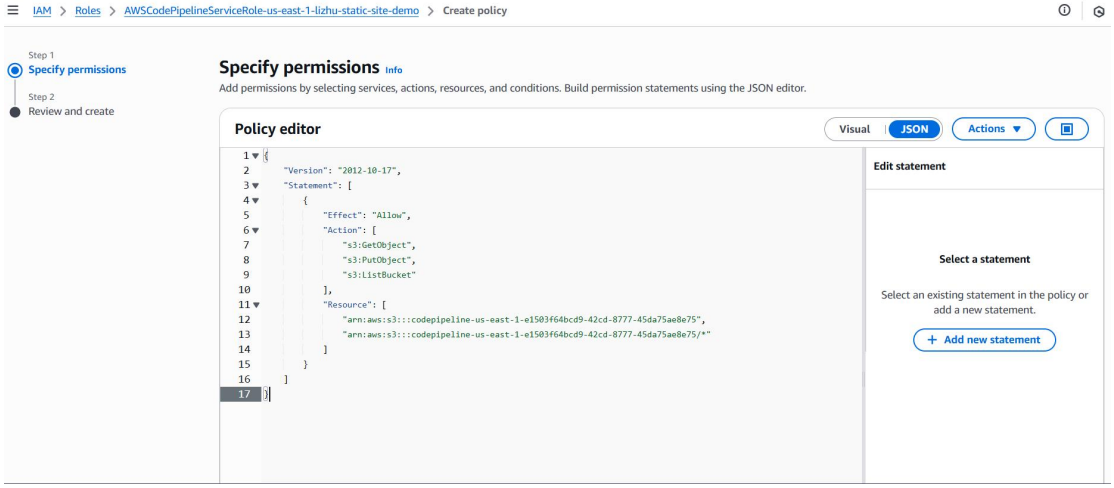


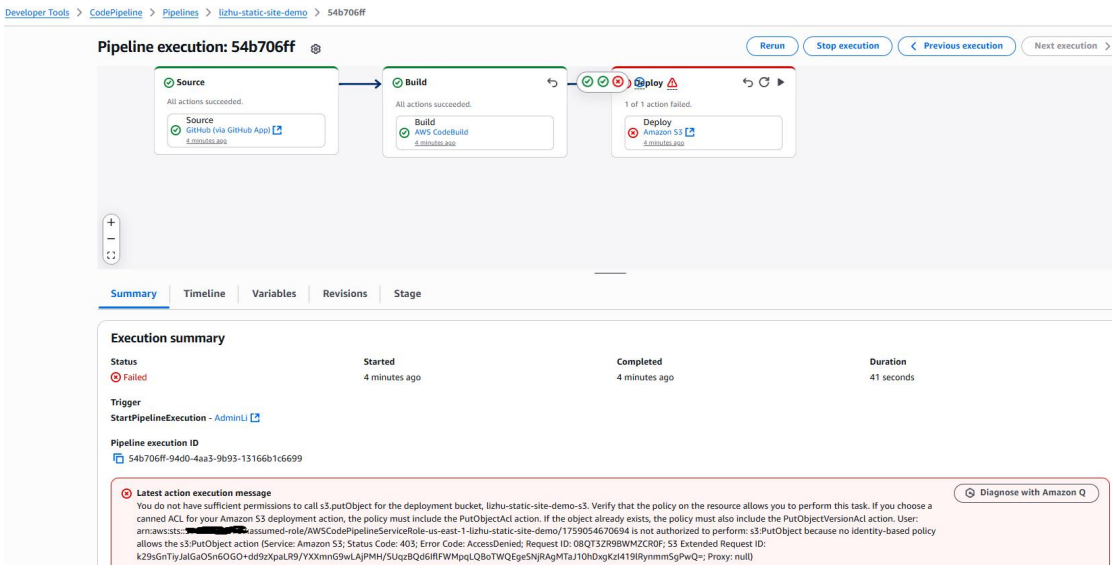
Step 2: If Issues Persist

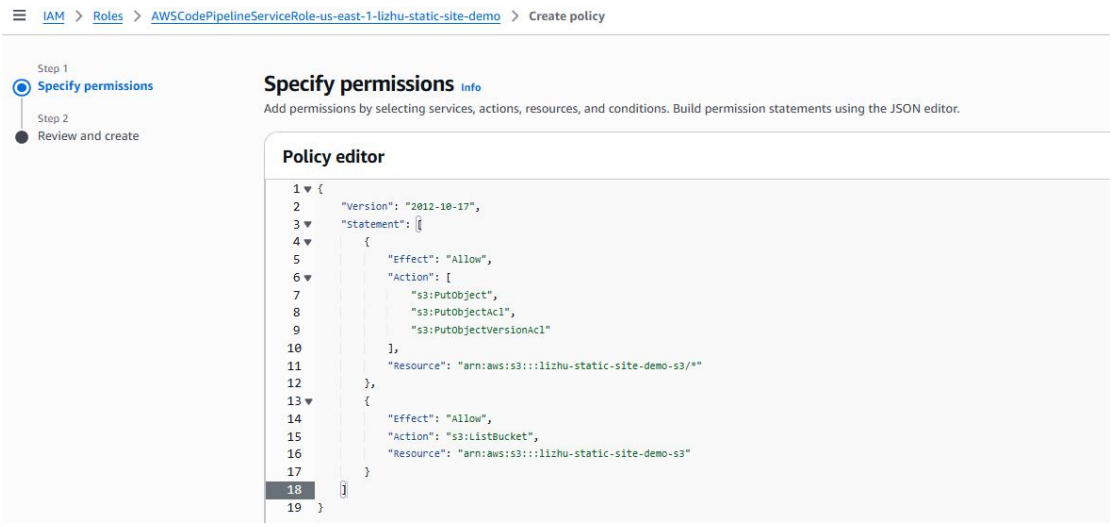
- CodeBuild permissions: Ensure the role includes full access (Get/Put/Delete/List) to the deploy S3 bucket and necessary access to the pipeline bucket.
- CodePipeline permissions: If the pipeline cannot pass artifacts between stages, add an inline policy to the CodePipeline role with the required s3:* actions on the artifact bucket.

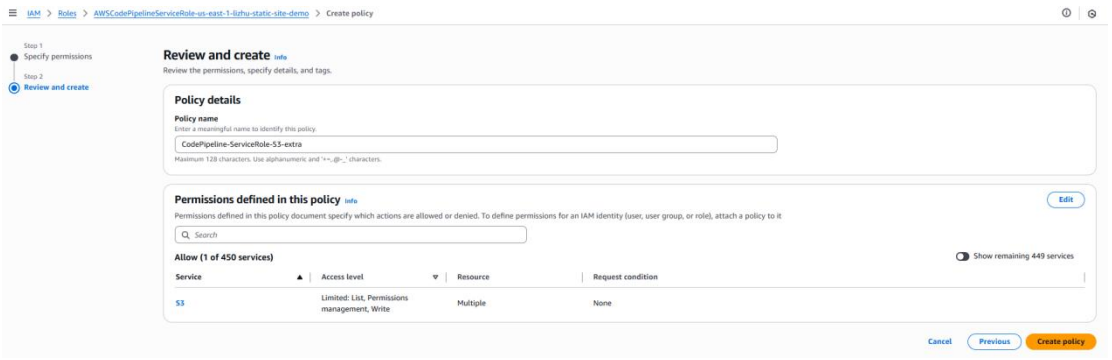



IAM > Roles > AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo Info
Delete
Summary
Edit
Creation date
September 27, 2025, 20:34 (UTC+01:00)
ARN
arn:aws:iam::[redacted]:role/service-role/AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo
Last activity
11 minutes ago
Maximum session duration
1 hour
Permissions
Trust relationships
Tags
Last Accessed
Revoke sessions
Permissions policies (4) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add permissions
Attach policies
Create inline policy
Filter by Type
Search
All types
Policy name
Type
Attached entities
AWSCodePipelineServiceRole-us-east-1-liz...
Customer managed
1
CodePipeline-CodeBuild-us-east-1-lizhu-st...
Customer managed
1


IAM > Roles > AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"s3:GetObject",
"s3:PutObject",
"s3:ListBucket"
],
"Resource": [
"arn:aws:s3:::codepipeline-us-east-1-e1503f64bcd9-42cd-8777-45da75ae8e75",
"arn:aws:s3:::codepipeline-us-east-1-e1503f64bcd9-42cd-8777-45da75ae8e75/*"
]
}
]
}
Edit statement
Select a statement
Select an existing statement in the policy or add a new statement.
Add new statement


Developer Tools > CodePipeline > Pipelines > lizhu-static-site-demo > 54b706ff
Pipeline execution: 54b706ff
Rerun
Stop execution
Previous execution
Next execution
Source
All actions succeeded.
Source
GitHub (via GitHub App)
4 minutes ago
Build
All actions succeeded.
Build
AWS CodeBuild
4 minutes ago
Deploy
1 of 1 action failed.
Deploy
Amazon S3
4 minutes ago
Summary
Timeline
Variables
Revisions
Stage
Execution summary
Status
Failed
Started
4 minutes ago
Completed
4 minutes ago
Duration
41 seconds
Trigger
StartPipelineExecution - AdminLi
Pipeline execution ID
54b706ff-94d0-4aa3-9b93-13166b1c6699
Latest action execution message
You do not have sufficient permissions to call s3.putObject for the deployment bucket, lizhu-static-site-demo-s3. Verify that the policy on the resource allows you to perform this task. If you choose a canned ACL for your Amazon S3 deployment action, the policy must include the PutObjectAcl action. If the object already exists, the policy must also include the PutObjectVersionAcl action. User: arn:aws:sts::[redacted]:assumed-role/AWSCodePipelineServiceRole-us-east-1-lizhu-static-site-demo/1759054670694 is not authorized to perform: s3:PutObject because no identity-based policy allows the s3:PutObject action (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied; Request ID: 08QT3ZR9BWMZCR0F; S3 Extended Request ID: k29sGnTiyJalGaOSn6OGO+dd9zXpaLR9/YXXmnG9wLAjPMH/SUqzBQd6IfIFWMpqLQBoTWQEgeSNjRAgMTaJ10hDxgKzI419lRynnmSgPwQ=; Proxy: null)
Diagnose with Amazon Q

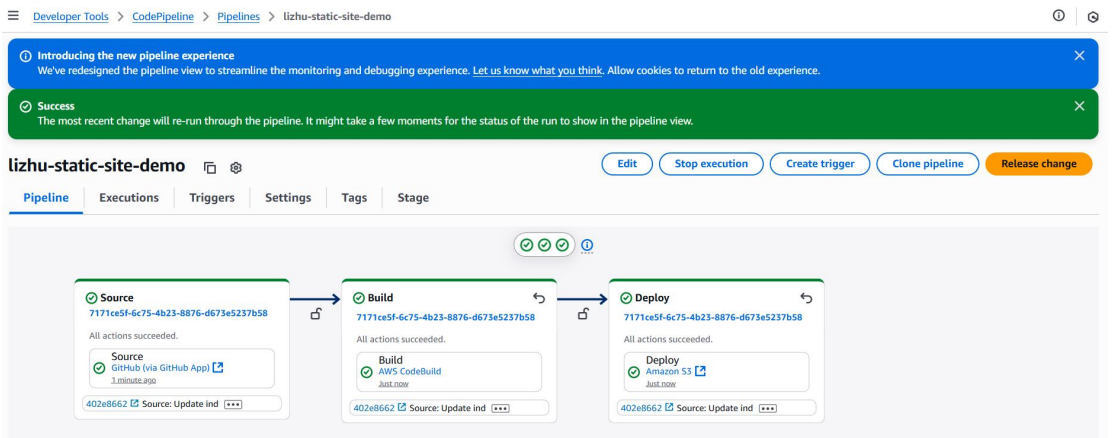


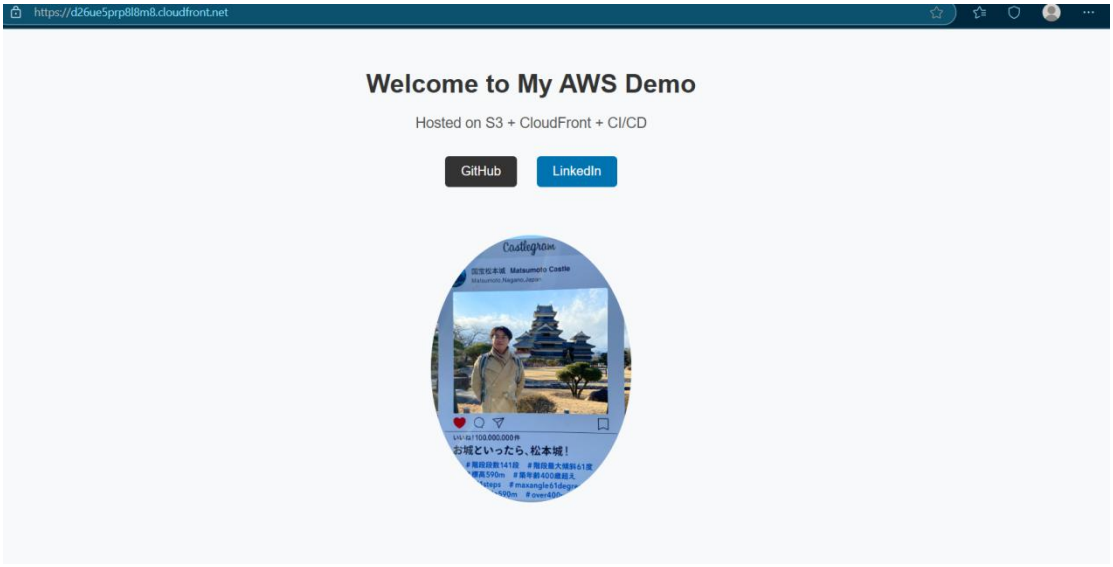


Most failures at this stage are caused by IAM misconfigurations or artifact exchange issues between services.

9. Validate Deployment

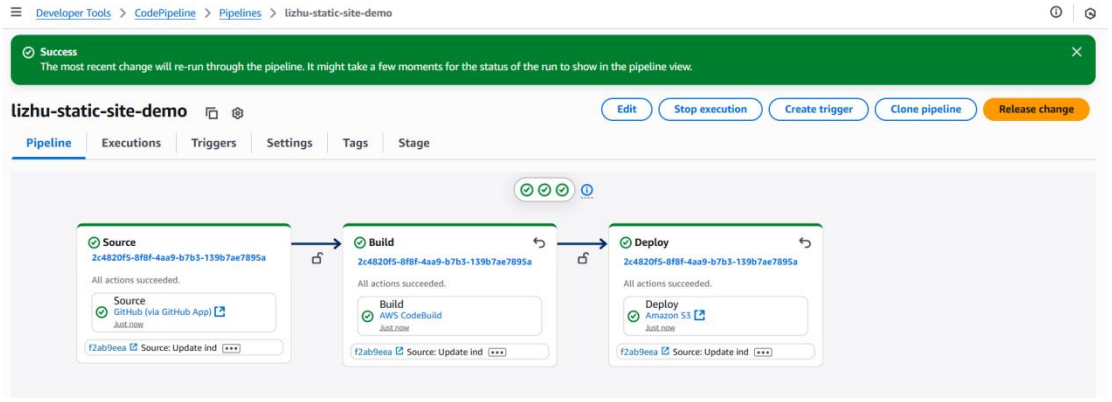
After successful deployment, test the CloudFront distribution to confirm site updates.

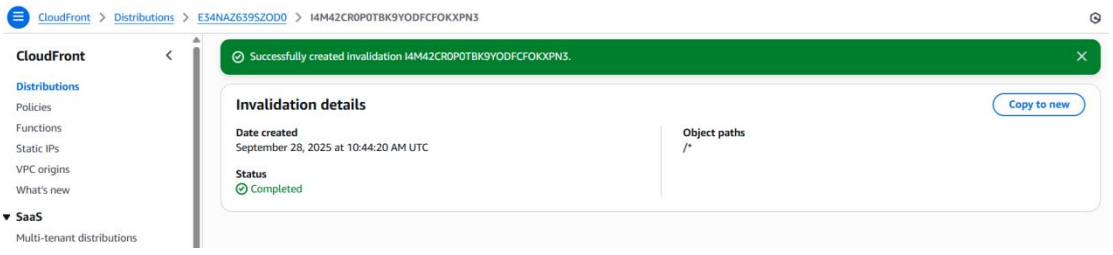


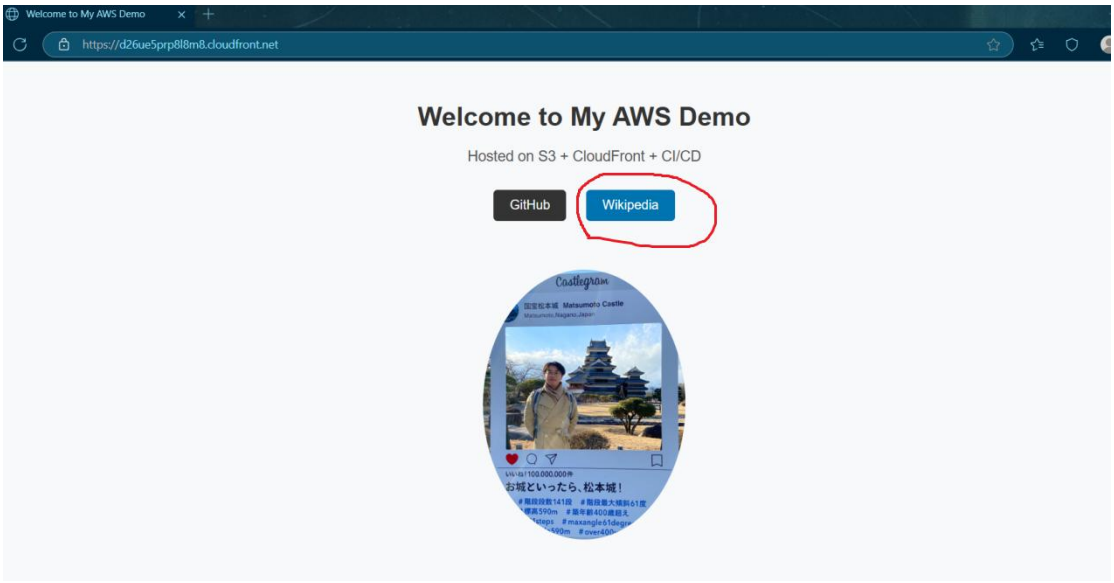


Modify index.html, commit the change to GitHub, and confirm the pipeline automatically redeploys the updated version. I replaced LinkedIn by Wikipedia, and you can find in the following screenshots the website changed.

```
50      <div class="button-container">
51          <a class="button github" href="https://github.com/lizh1994/aws-static-site-cicd-lizhu" target="_blank">GitHub</a>
52          <a class="button" href="https://en.wikipedia.org/wiki/Main_Page" target="_blank">Wikipedia</a>
53      </div>
```

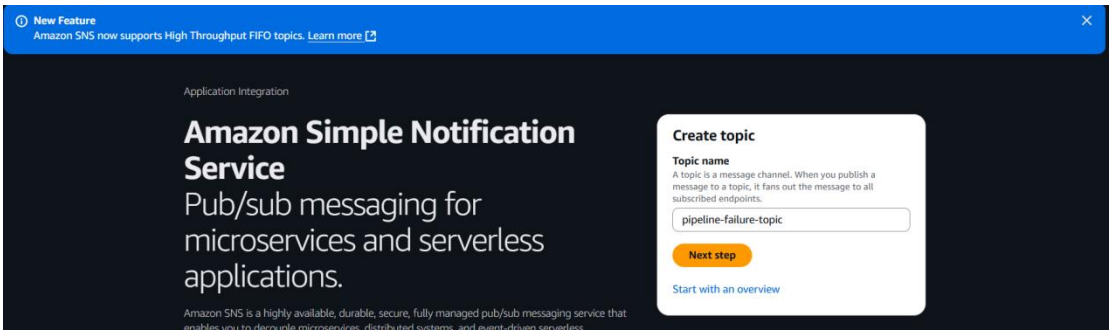


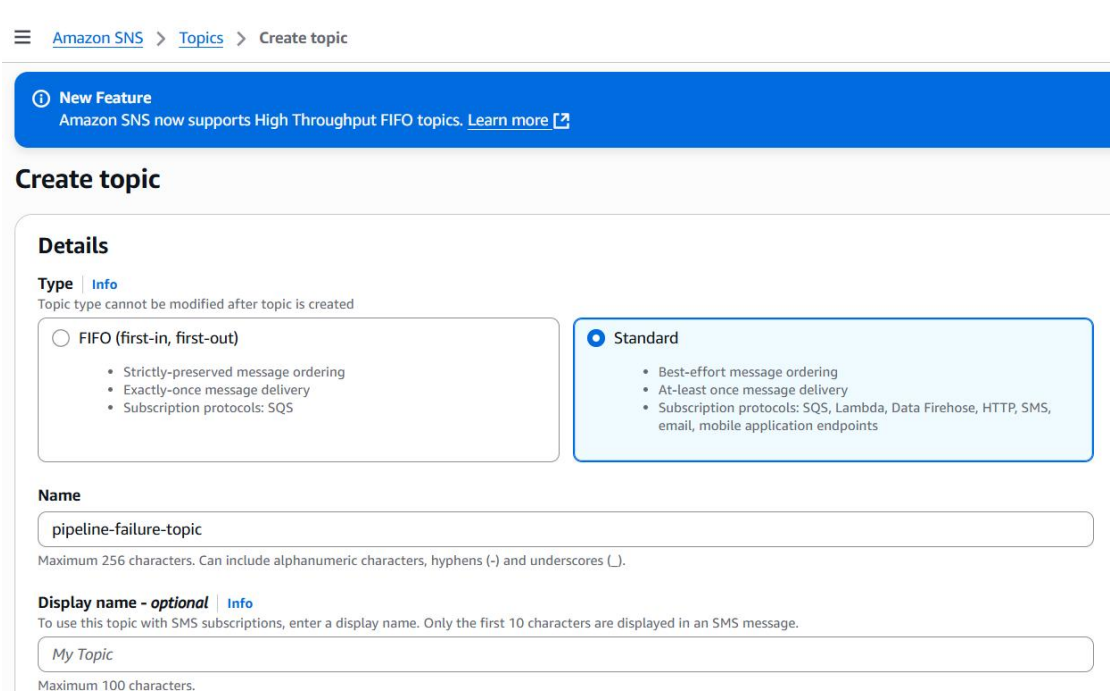


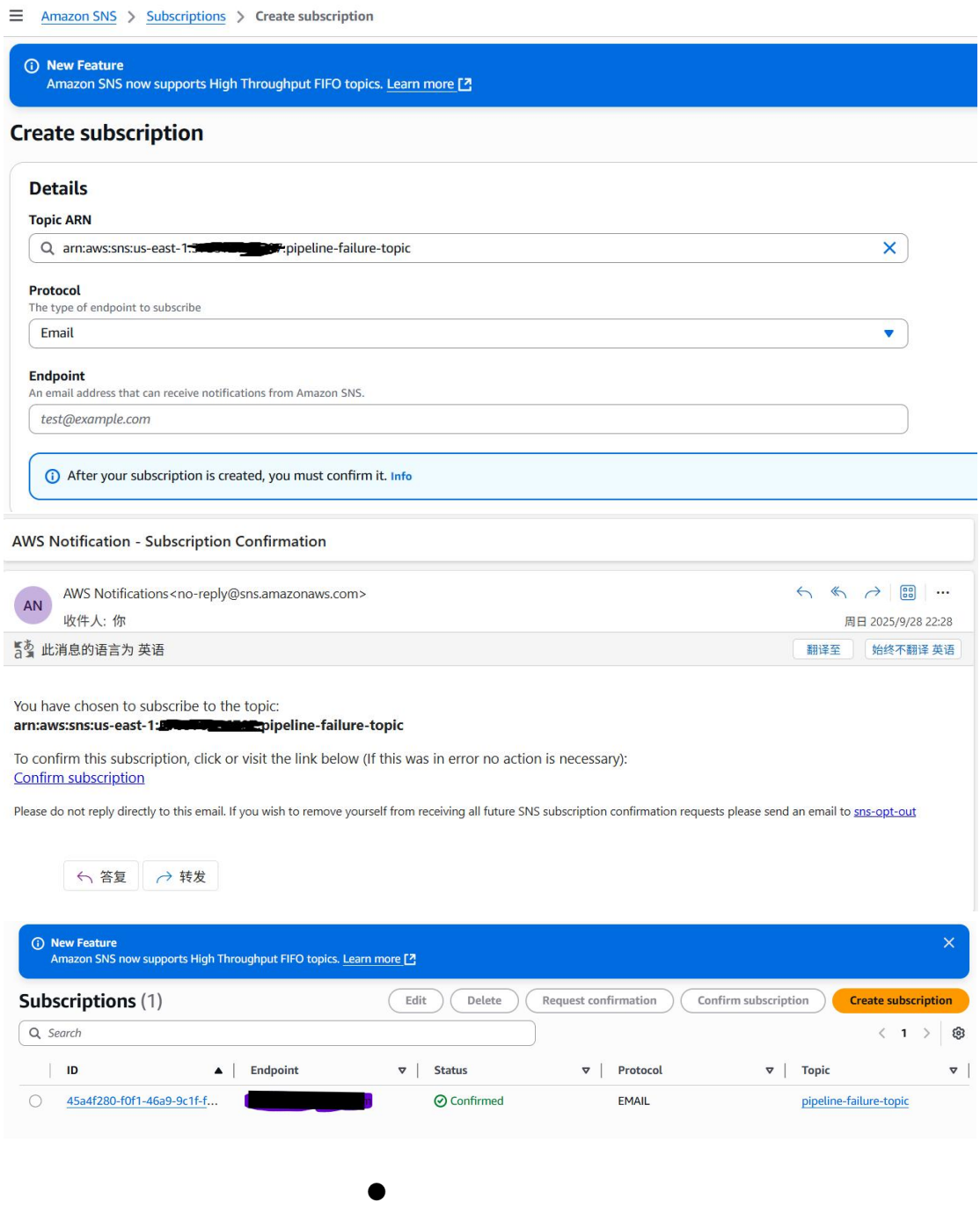


10. Add Monitoring & Notifications

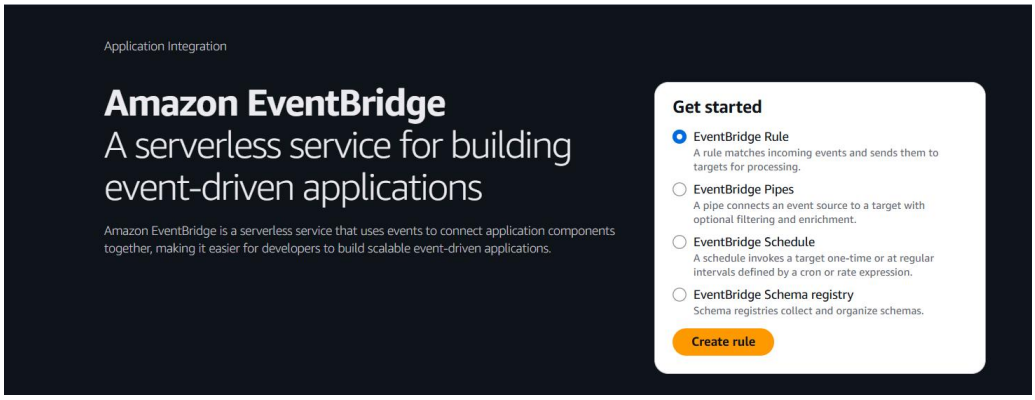
Create an SNS Topic and subscribe a personal email.

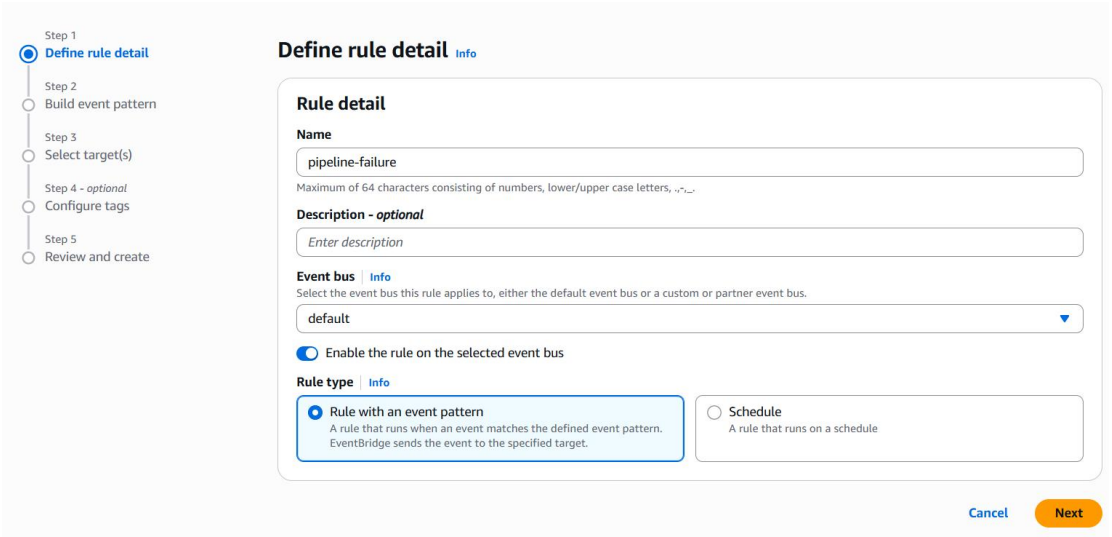


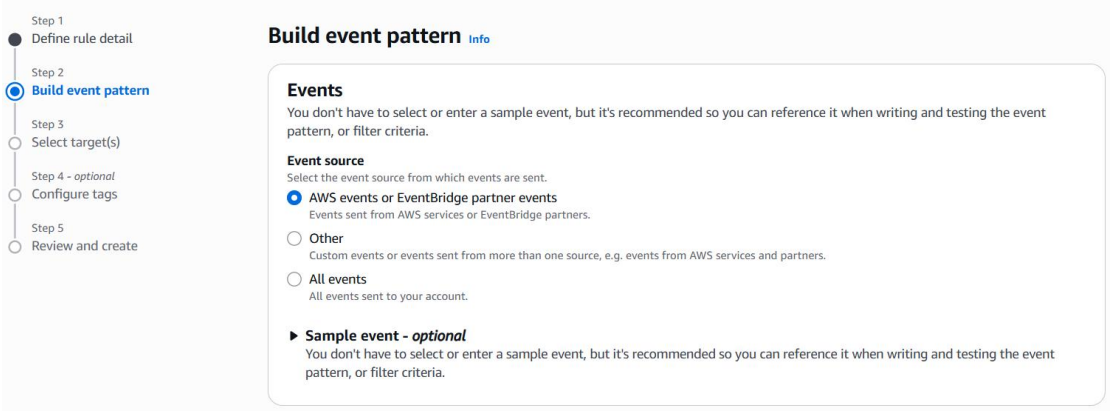


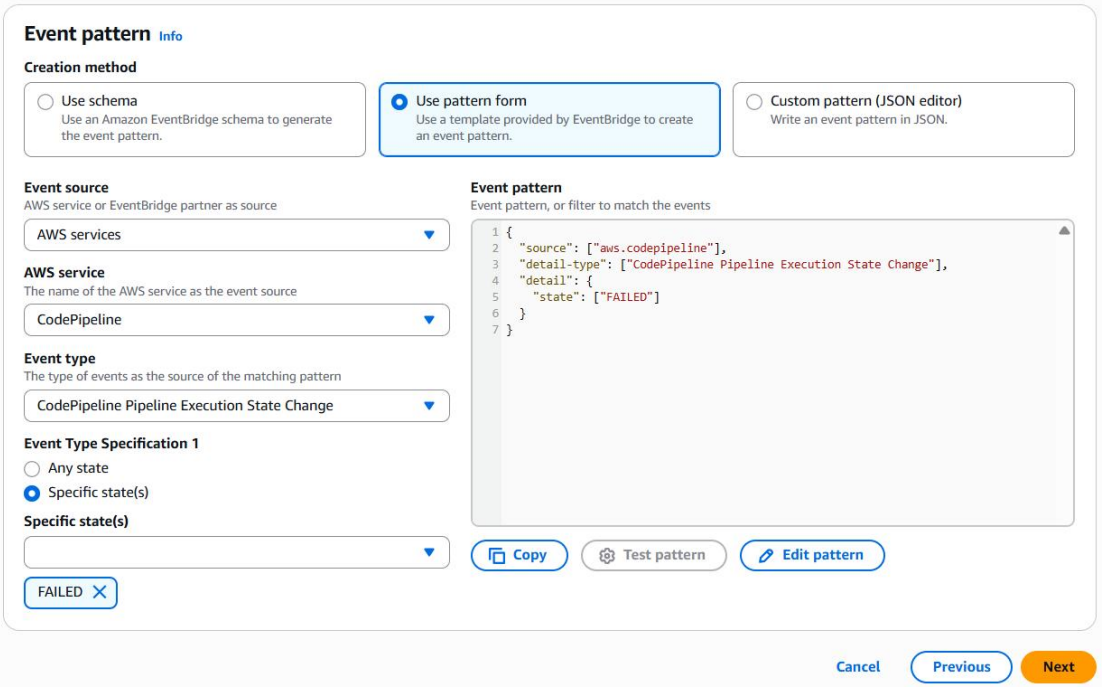


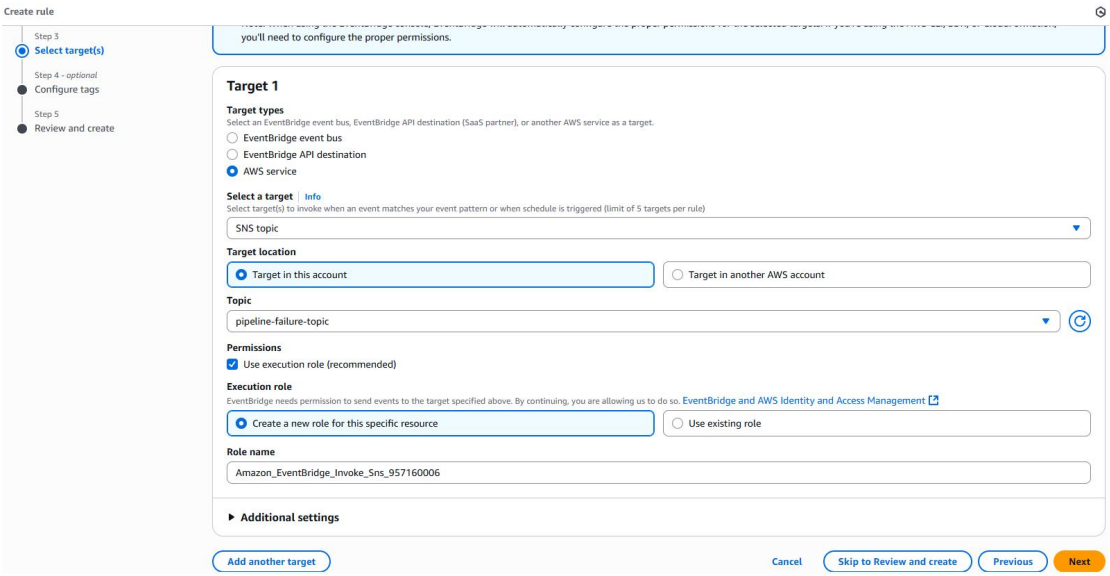
Set up EventBridge to detect pipeline failure events.

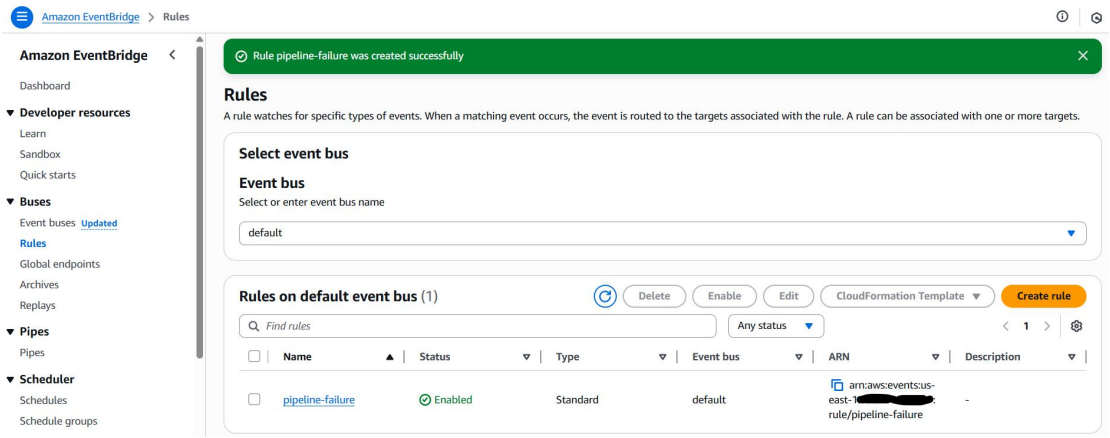



Step 1
Define rule detail
Step 2
Build event pattern
Step 3
Select target(s)
Step 4 - optional
Configure tags
Step 5
Review and create
Define rule detail Info
Rule detail
Name
pipeline-failure
Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.
Description - optional
Enter description
Event bus Info
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.
default
Enable the rule on the selected event bus
Rule type Info
Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.
Schedule
A rule that runs on a schedule
Cancel
Next

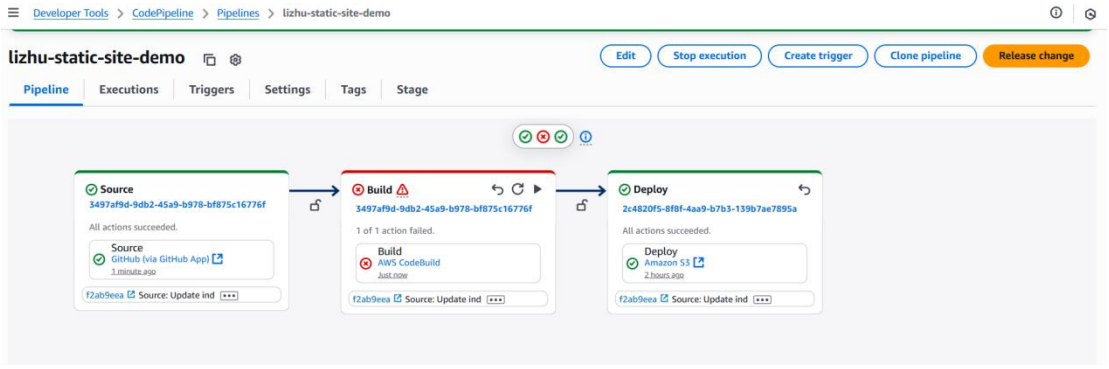

Step 1
Define rule detail
Step 2
Build event pattern
Step 3
Select target(s)
Step 4 - optional
Configure tags
Step 5
Review and create
Build event pattern Info
Events
You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.
Event source
Select the event source from which events are sent.
AWS events or EventBridge partner events
Events sent from AWS services or EventBridge partners.
Other
Custom events or events sent from more than one source, e.g. events from AWS services and partners.
All events
All events sent to your account.
Sample event - optional
You don't have to select or enter a sample event, but it's recommended so you can reference it when writing and testing the event pattern, or filter criteria.

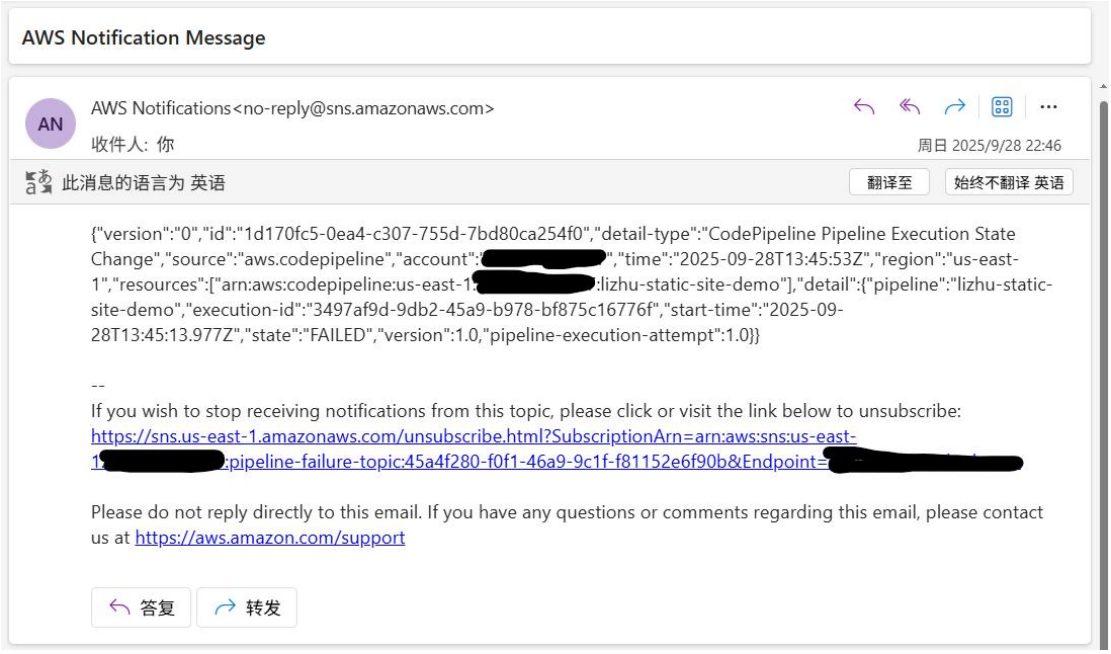

Event pattern Info
Creation method
Use schema
Use an Amazon EventBridge schema to generate the event pattern.
Use pattern form
Use a template provided by EventBridge to create an event pattern.
Custom pattern (JSON editor)
Write an event pattern in JSON.
Event source
AWS service or EventBridge partner as source
AWS services
AWS service
The name of the AWS service as the event source
CodePipeline
Event type
The type of events as the source of the matching pattern
CodePipeline Pipeline Execution State Change
Event Type Specification 1
Any state
Specific state(s)
Specific state(s)
FAILED
Event pattern
Event pattern, or filter to match the events
{
"source": ["aws.codepipeline"],
"detail-type": ["CodePipeline Pipeline Execution State Change"],
"detail": {
"state": ["FAILED"]
}
}
Copy
Test pattern
Edit pattern
Cancel
Previous
Next





Tested by intentionally breaking IAM permissions, I removed the s3:DeleteObject here → confirmed alert email was received.





11. Lessons Learned

11. 1 IAM Permissions

During the project, I gained a much clearer understanding of how IAM roles and policies work. Striking the right balance between least privilege and task feasibility is essential:

- Roles must be granted just enough access to complete their tasks, but not more.
- This requires knowing exactly what each AWS service will do, and how it interacts with others.
- Permissions must also be scoped at the right level of granularity.

Many of my initial failures were caused by insufficient permissions. These issues forced me to gradually clarify the boundaries of each role's access, which deepened my understanding of IAM in practice.

11. 2 Troubleshooting Skills

This was my first time building an AWS demo project end-to-end, so failures were

frequent. Through this process I learned how to:
- Read error logs carefully.
- Identify the root cause by correlating logs with service behavior.
- Apply targeted fixes instead of random trial and error.

Learning to debug AWS services systematically has been one of the most valuable outcomes of this project.

11. 3 Service Interactions

Another key lesson was to always map out how services exchange data and interact with each other. Missing or unclear interactions often caused failures.
For example, in my first pipeline runs the deploy stage failed repeatedly. After verifying permissions, I realized the real problem was that the build stage never produced a build artifact output. Without that artifact, CodePipeline could not hand anything over to the deploy stage.
Understanding these service interactions up front prevents wasted effort and unnecessary debugging.

Final Reflection

Most importantly, this project reminded me that hands-on practice is irreplaceable. Designing demos, facing failures, and solving problems along the way provided insights no tutorial could give me.
The key is to practice with curiosity, ask questions when things break, and keep the mindset that failure is part of the learning process.