# AWS DevOps CI/CD Checkride – Step‑by‑Step Guide (with Code & Rationale)

This document is a clear, end‑to‑end walkthrough you can use for a **checkride/demo**. It includes the **what**, **why**, and **how**, plus the **exact code** and **UI clicks**. The scenario: a simple Python AWS Lambda function deployed via **CodePipeline → CodeBuild → CloudFormation/SAM**, with GitHub as the source.

## 1) Architecture at a Glance

**Flow:** GitHub (source) → CodePipeline (orchestrator) → CodeBuild (package with SAM) → S3 packaging bucket (stores code bundle) → CloudFormation/SAM (deploys) → Lambda (runtime) → API Gateway (HTTP endpoint).

**Why this setup?** - **CodePipeline** ensures repeatable, auditable automation with stages and approvals. - **CodeBuild** provides ephemeral, isolated build workers and first‑class AWS CLI/SAM support. - **S3 packaging bucket** is required by aws cloudformation package to upload code artifacts referenced by CodeUri. - **CloudFormation/SAM** gives immutable, versioned infrastructure as code, drift detection, and easy rollback. - **Lambda + API Gateway** fits a simple stateless “Hello” service and keeps infra minimal for a demo.

## 2) Repo Layout (minimal)

/ (repo root)  
├─ buildspec.yaml # Build & package instructions  
├─ samTemplate.yaml # SAM template (IaC)  
└─ simple-hello-lambda-code/  
 ├─ lambda\_handler.py # App code  
 └─ requirements.txt # (optional deps)

## 3) Final, Working Code

### 3.1 Lambda handler

# simple-hello-lambda-code/lambda\_handler.py  
import json  
  
def lambda\_handler(event, context):  
 print("hello from pipeline demo")  
 return {"statusCode": 200, "body": json.dumps({"message": "Hello from Lambda!"})}

### 3.2 SAM template (lets SAM create the role, adds API trigger)

# samTemplate.yaml  
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
Description: Simple Hello Lambda – deployed via CodePipeline/CodeBuild/SAM  
  
Resources:  
 SampleHelloApp:  
 Type: AWS::Serverless::Function  
 Properties:  
 FunctionName: sample-hello-app  
 Handler: lambda\_handler.lambda\_handler  
 Runtime: python3.10  
 CodeUri: simple-hello-lambda-code/  
 MemorySize: 1024  
 Timeout: 10  
 Policies:  
 - AWSLambdaBasicExecutionRole  
 Events:  
 GetHello:  
 Type: Api  
 Properties:  
 Path: /hello  
 Method: GET  
  
Outputs:  
 HelloApiUrl:  
 Description: API endpoint for the demo  
 Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/hello"

### 3.3 Buildspec (no hard‑coded bucket; uses env var; single‑line CLI)

# buildspec.yaml  
version: 0.2  
phases:  
 install:  
 runtime-versions:  
 python: 3.10  
 pre\_build:  
 commands:  
 - pip install -r simple-hello-lambda-code/requirements.txt -t simple-hello-lambda-code/ || true  
 - echo "PACKAGE\_BUCKET=$PACKAGE\_BUCKET"  
 - aws s3api head-bucket --bucket "$PACKAGE\_BUCKET" || { echo "Packaging bucket not found"; exit 1; }  
 build:  
 commands:  
 - aws cloudformation package --template-file samTemplate.yaml --s3-bucket $PACKAGE\_BUCKET --output-template-file outputSamTemplate.yaml  
artifacts:  
 files:  
 - outputSamTemplate.yaml  
 discard-paths: yes  
cache:  
 paths:  
 - '/root/.cache/pip/\*\*/\*'

**Why single‑line command?** Multi‑line with \ can break in build shells due to CRLF/whitespace, causing the CLI to think args are missing.

## 4) One‑Time AWS Prereqs (Console)

1. **CodeStar Connection to GitHub**: Developer Tools → **Connections** → Create (**GitHub via GitHub App**). Used by CodePipeline Source.
2. **S3 Packaging Bucket** *(eu‑north‑1 to match pipeline)*: S3 → Create bucket → e.g., simple-hello-packages-eu-north-1-<accountId>.
3. **IAM – CodeBuild service role permissions** (read artifact bucket, read/write packaging bucket). Minimal inline policy:

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {"Effect":"Allow","Action":["s3:GetObject","s3:GetObjectVersion","s3:ListBucket"],  
 "Resource":["arn:aws:s3:::<PIPELINE\_ARTIFACT\_BUCKET>","arn:aws:s3:::<PIPELINE\_ARTIFACT\_BUCKET>/\*"]},  
 {"Effect":"Allow","Action":["s3:PutObject","s3:GetObject","s3:ListBucket"],  
 "Resource":["arn:aws:s3:::<PACKAGING\_BUCKET>","arn:aws:s3:::<PACKAGING\_BUCKET>/\*"]}  
 ]  
}

1. **IAM – CodePipeline role permissions** (trigger CodeBuild & use CodeStar connection). Minimal inline policy:

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {"Effect":"Allow","Action":["codebuild:StartBuild","codebuild:BatchGetBuilds"],  
 "Resource":"arn:aws:codebuild:eu-north-1:<ACCOUNT\_ID>:project/simple-hello-build"},  
 {"Effect":"Allow","Action":["codestar-connections:UseConnection"],  
 "Resource":"arn:aws:codestar-connections:eu-north-1:<ACCOUNT\_ID>:connection/<ID>"}  
 ]  
}

## 5) Create the CodeBuild Project (Console)

* **Name:** simple-hello-build
* **Source provider:** **CodePipeline** (important; the pipeline provides the source ZIP)
* **Environment:** Managed, Amazon Linux 2, aws/codebuild/standard:7.0, Small
* **Service role:** create new (then attach the inline policy above)
* **Buildspec:** *Use a buildspec file* → buildspec.yaml
* **Artifacts:** **CodePipeline**
* **(Optional default) Environment variable:** PACKAGE\_BUCKET = your packaging bucket (**Type: Plaintext**)

**Why provider=CodePipeline?** Guarantees the same revision (commit) the Source stage produced; prevents drift between Source and Build.

## 6) Create/Edit the CodePipeline (Console)

### Stages

1. **Source**
   * Provider: GitHub (via CodeStar Connection)
   * Repo: srinivasadhu/simple-hello-app
   * Branch: main
   * Output artifact: SourceOutput
2. **Build**
   * Provider: CodeBuild → Project: simple-hello-build
   * **Input artifact:** SourceOutput
   * **Output artifact:** BuildOutput
   * **Environment variable override:** PACKAGE\_BUCKET = your packaging bucket (**Plaintext**)
3. **Deploy**
   * Provider: CloudFormation
   * Region: eu-north-1
   * **Action mode:** Create or update a stack
   * **Stack name:** simple-hello-app-stack
   * **Input artifact:** BuildOutput
   * **Template file:** outputSamTemplate.yaml
   * **Capabilities:** CAPABILITY\_IAM, CAPABILITY\_NAMED\_IAM, CAPABILITY\_AUTO\_EXPAND

**Why AUTO\_EXPAND?** Required for the AWS::Serverless-2016-10-31 transform used by SAM.

## 7) Demo Script (Checkride)

1. **Show architecture** (diagram slide). Summarize the flow.
2. **Open GitHub repo**: point out samTemplate.yaml, buildspec.yaml, and handler code.
3. **Open packaging bucket** (S3): show it exists and is in eu‑north‑1.
4. **Open CodeBuild project**: highlight *Source=CodePipeline*, buildspec, and env var PACKAGE\_BUCKET.
5. **Open CodePipeline**: show the three stages and how BuildOutput feeds Deploy.
6. **Trigger a run**: click **Release change** (or push a small commit).
7. **Tail Build logs**: watch aws cloudformation package … upload to S3 and emit outputSamTemplate.yaml.
8. **Deploy**: show CloudFormation stack events; demonstrate **Outputs → HelloApiUrl**.
9. **Test API**: open the URL; show JSON {"message":"Hello from Lambda!"}.
10. **(Optional) Explain rollback**: updates create change sets; failed deploys roll back to last good state.

## 8) Common Pitfalls You Can Call Out (and fixes)

* **Using a folder name instead of a real S3 bucket** for --s3-bucket → create a **dedicated bucket** and pass as **Plaintext** env var.
* **Buildspec using line continuations** that break → use a **single line** command.
* **CodeBuild Source=GitHub** while also using CodePipeline → set **Source=CodePipeline**.
* **Env var Type=Parameter** but value is a literal → set Type=**Plaintext**.
* **Missing S3 permissions for CodeBuild role** → add minimal inline policy shown above; include KMS rights if the artifact bucket is encrypted.
* **Deploy fails with SAM transform** → ensure Deploy has **CAPABILITY\_AUTO\_EXPAND** and the template path is BuildOutput::outputSamTemplate.yaml.
* **Pipeline role can’t start builds** → grant codebuild:StartBuild to the pipeline role for the project ARN.

## 9) How You’d Scale / Improve (talking points)

* **Multiple environments**: dev → staging → prod with manual approvals and parameter overrides per stage.
* **Testing**: add unit tests in pre\_build; fail fast.
* **Security**: least‑privilege IAM, artifact KMS encryption, connection scopes, and restricted bucket policies.
* **Observability**: CloudWatch dashboards, alarms on Lambda errors/latency, X‑Ray for tracing.
* **Quality gates**: static scans (bandit, cfn‑nag), linting, policy‑as‑code.
* **Blue/Green or Canary**: SAM hooks or CodeDeploy for Lambda.

## 10) Clean‑up (cost control)

* Delete **CloudFormation stack** simple-hello-app-stack.
* Delete the **pipeline** and **CodeBuild** project.
* Empty and delete the **packaging bucket** and **artifact bucket**.

## Appendix A – Minimal IAM JSON Snippets

### A.1 CodeBuild service role (replace bucket names)

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {"Effect":"Allow","Action":["s3:GetObject","s3:GetObjectVersion","s3:ListBucket"],  
 "Resource":["arn:aws:s3:::<PIPELINE\_ARTIFACT\_BUCKET>","arn:aws:s3:::<PIPELINE\_ARTIFACT\_BUCKET>/\*"]},  
 {"Effect":"Allow","Action":["s3:PutObject","s3:GetObject","s3:ListBucket"],  
 "Resource":["arn:aws:s3:::<PACKAGING\_BUCKET>","arn:aws:s3:::<PACKAGING\_BUCKET>/\*"]},  
 {"Effect":"Allow","Action":["logs:CreateLogGroup","logs:CreateLogStream","logs:PutLogEvents"],"Resource":"\*"}  
 ]  
}

### A.2 CodePipeline role (build + connection)

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {"Effect":"Allow","Action":["codebuild:StartBuild","codebuild:BatchGetBuilds"],  
 "Resource":"arn:aws:codebuild:eu-north-1:<ACCOUNT\_ID>:project/simple-hello-build"},  
 {"Effect":"Allow","Action":["codestar-connections:UseConnection"],  
 "Resource":"arn:aws:codestar-connections:eu-north-1:<ACCOUNT\_ID>:connection/<ID>"}  
 ]  
}

## Appendix B – Optional: Single‑Template Pipeline (IaC)

If you want to provision the CI/CD infra itself via CloudFormation later, you can lift the pipeline template you used earlier. For the checkride, the **console‑driven** setup keeps the story crisp and easy to demo.

## Appendix C – Quick Q&A (for the panel)

* **Why SAM and not raw CloudFormation?** SAM simplifies serverless authoring and transforms to CFN; less boilerplate, faster demos.
* **Why separate packaging bucket?** cloudformation package must upload code somewhere durable; separating it from artifact bucket is cleaner for access control and retention.
* **How do you roll back?** CloudFormation keeps stack history and will roll back on failure; prior template/artifacts remain in S3.
* **How to add approvals?** Insert a **Manual approval** action between Build and Deploy; or multi‑account promotion with cross‑account roles.
* **How to secure?** Restrictive IAM, KMS for buckets, scoped CodeStar connection, VPC for Lambda if needed, and Secrets Manager for secrets.

**That’s your checkride script and reference.** Open this doc during the session and follow Sections 7–8 verbatim; keep Appendix C handy for questions.