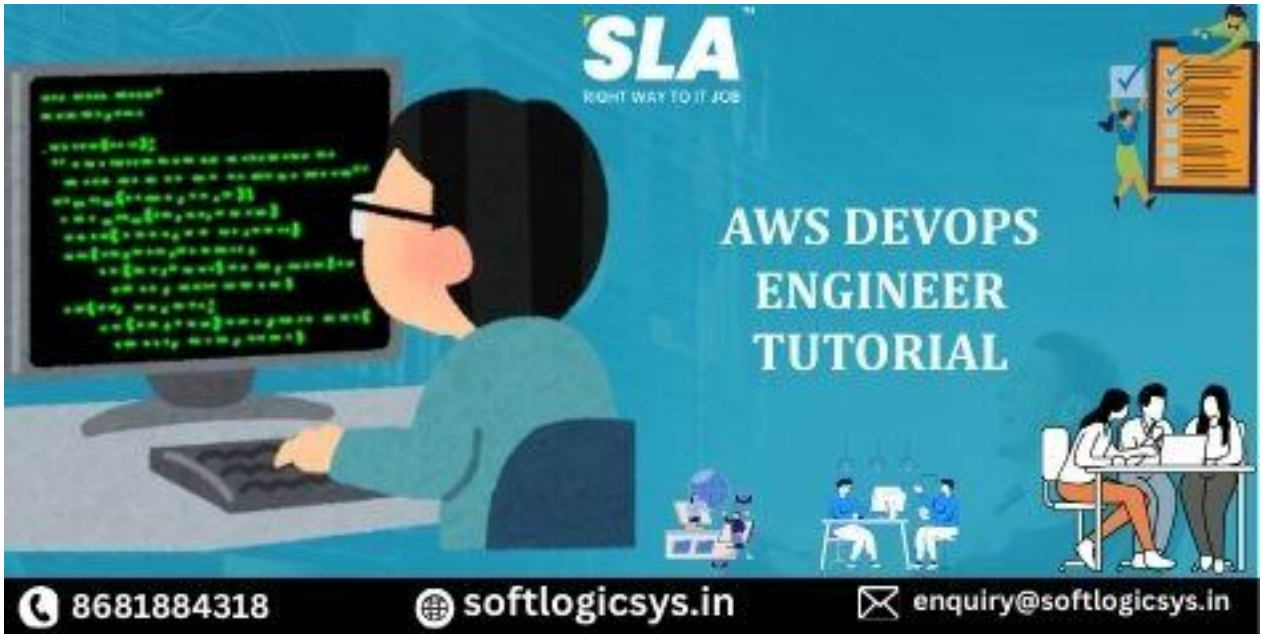

AWS DevOps Engineer Tutorial for Beginners

Introduction

Are you a beginner who is finding it difficult to stitch together dozens of cloud tools for your deployment pipeline? Do the constant infrastructure changes create manual errors and delays?

AWS DevOps provides a rich set of integrated services-for example, CodeCommit, CodeBuild, CodeDeploy, and CodePipeline-that automate the entire software release lifecycle directly on the world's leading cloud platform. This tutorial will walk you through committing your code all the way to automated cloud deployment using AWS and will simplify Continuous Integration/Continuous Delivery, better known as CI/CD.

Ready to deploy automatically to the cloud? Download the complete **AWS DevOps Course Syllabus** to get started with your automation!

Why Students or Freshers Learn AWS DevOps?

AWS DevOps Skills bring the following valuable career benefits:

- **Cloud Market Dominance:** AWS is, without question, the leading cloud platform provider; knowledge of its DevOps tooling CodePipeline and CodeBuild features highly in the marketplace.
- **Natively Integrated Tools:** AWS delivers a fully integrated set of services both for CI/CD and infrastructure management, making the automation of complex deployments easier directly on the cloud platform.
- **Infrastructure as Code (IaC):** Proficiency in infrastructure provisioning and management using CloudFormation or Terraform on AWS is critical for efficient and repeatable resource management.
- **High Demand for Cloud Automation:** Companies urgently need engineers who can reliably build, deploy, and scale applications within the AWS environment.

Ready to secure your cloud engineering job? Download our **Top AWS DevOps Interview Questions and Answers** to showcase expertise in cloud automation.

Step-by-Step AWS DevOps Tutorial for Beginners

This AWS DevOps tutorial will walk you through setting up your environment and building a simple continuous integration/continuous deployment pipeline to automate the deployment of a static website to Amazon S3 using key AWS services: CodeCommit, CodeBuild, and CodePipeline.

Because AWS DevOps is a cloud service, “installation” will consist of configuring your local tools and access to your AWS account.

Step 1: Account Creation & Installation on Local

You need an AWS account and local tools to interact with the platform.

1.1 Create an AWS Account

Action: Create an AWS Free Tier account on the official Amazon Web Services website. This will give you free access to the services used in this tutorial, within certain limits.

1.2 Configure IAM User Access

Security Best Practice: Never use the root account for daily tasks. Create an IAM (Identity and Access Management) user.

- **AWS Console:** Go to **IAM** service.
- Click Users, then click **Create User**.
- **Permissions:** Attach the following policies: IAMFullAccess (temporarily for setup), AmazonS3FullAccess, AWSCodeCommitFullAccess, AWSCodeBuildAdminAccess, and AWSCodePipeline_FullAccess.
- **Security Credentials:** After the user has been created, generate an Access Key ID and Secret Access Key. Store these credentials in a safe place—you'll need them in the next step.

1.3 Install AWS CLI (Command Line Interface)

AWS CLI is a method through which AWS services can be administered from your computer.

- **Action:** Download and install the latest **AWS CLI** for your operating system.
- **Verification:** Open terminal/command prompt and execute: *aws –version*

1.4 AWS CLI Configuration

Set up your local machine using the credentials you saved in Step 1.2.

aws configure

Input: Enter your Access Key ID, Secret Access Key, preferred region (e.g., us-east-1), and output format (e.g., json).

Step 2: Preparing the Application & Repository

What we need is a simple application to deploy and a private Git repository to store this application.

2.1 Creating a Static Website

Create a folder on your local machine called my-static-website and add a simple HTML file inside of it.

File: *index.html*

```
<!DOCTYPE html>

<html>

<head>

    <title>AWS DevOps CI/CD Demo</title>

</head>

<body>

    <h1>Welcome to My Automated AWS Deployment!</h1>

    <p>This page was deployed automatically via AWS CodePipeline.</p>

</body>

</html>
```

2.2 Create the CodeCommit Repository

CodeCommit is AWS's fully managed Git repository service.

- **AWS Console:** Now, go to CodeCommit.
- Click **Create Repository**.
- **Name:** Name it MyWebsiteRepo.
- **Action:** Once created, follow the instructions to configure HTTPS Git Credentials or SSH keys specific to IAM to access CodeCommit.

2.3 Push Code to CodeCommit

- **Local PC:** Change into your my-static-website directory.
- Initialize Git, commit the files and push to CodeCommit using the HTTPS URL that was provided by CodeCommit.

git init

git add .

git commit -m "Initial static website files"

git remote add origin YOUR_CODECOMMIT_REPO_URL

git push -u origin master

- **Verification:** Check the CodeCommit console. Your index.html should be visible.

Step 3: Setting up the Deployment Target (Amazon S3)

Static website files are hosted using Amazon S3 (Simple Storage Service).

3.1 Create an S3 Bucket

- **AWS Console:** Proceed to S3.

- Click **Create bucket**.
- **Name:** Enter a name that is unique worldwide, such as my-devops-website-app-12345.
- **Settings:** Uncheck Block all public access (you have to accept this security risk for a public website)
- **Action:** Create the bucket.

3.2 Enable Static Website Hosting

- In the S3 console, select your new bucket.
- Go to the **Properties** tab.
- Scroll down to **Static website hosting** and click **Edit**.
- Enable **static website hosting**.
- **Index document:** Set this to *index.html*.
- **Action:** Save your changes. Note the Endpoint URL – this is where your website will be accessible.

3.3 Set Bucket Policy

You must allow public read access to the files in the bucket.

1. In the S3 console, click on the Permissions tab.
2. Under Bucket policy, click Edit and paste the following JSON policy. Replace YOUR_BUCKET_NAME with your actual bucket name:

```
{

  “Version”: “2012-10-17”,

  “Statement”: [

    {
```

```
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME/*"
        }
    ]
}
```

Save the policy. Now your S3 bucket is ready for deploying your application.

Step 4. Setting Up Build Service (CodeBuild)

CodeBuild compiles source code, runs tests, and produces deployable artifacts. Since we are deploying a simple static site, the major function performed by CodeBuild is to zip the files and create an artifact.

4.1 Creating a Build Project

- **AWS Console:** Open the CodeBuild console.
- Click '**Create build project**'.
- **Project Name:** Name it *MyWebsiteBuildProject*.

4.2 Configuration Source

- **Source provider:** Choose **AWS CodeCommit**.
- **Repository:** Select *MyWebsiteRepo*.

- **Branch:** Select *master*.

4.3 Environment Configuration

- **Environment image:** Select **Managed image**.
- **Operating system:** Choose **Amazon Linux 2**.
- **Runtime(s):** Choose *Standard*.
- **Image:** Use the latest standard image, e.g., *aws/codebuild/amazonlinux2-x86_64:3.0.*
- **Service Role:** Choose Create a new service role. AWS will create an IAM role for CodeBuild to act on.

4.4 Buildspec File

The *buildspec.yml* file defines the commands that CodeBuild runs. It needs to be placed in the root of your CodeCommit repository.

- **Buildspec:** Choose Use a buildspec file.

4.5 Artifacts

- **Type:** Select *Amazon S3*.
- **Bucket:** Choose your S3 bucket (*my-devops-website-app-12345)*.
- **Name:** Give it a name like *website-artifact*.
- **Packaging:** Choose *ZIP*.

4.6 Action: Create the Buildspec File Locally

Create the file buildspec.yml in your my-static-website folder.

 # buildspec.yml

version: 0.2

phases:

```
  install:
    commands:
      # Optional: Install any necessary build tools (not needed for static site)
      - echo Nothing to install.
  pre_build:
    commands:
      - echo Running pre-build checks…
  build:
    commands:
      - echo Starting build process…
      # No compilation needed; the build is just creating the artifact
      - echo Zipping files…
  post_build:
    commands:
      - echo Build complete.
artifacts:
  files:
```

 # Specify the files to include in the artifact zip (all files in the root)

*– '**/*'*

base-directory: ./

4.7 Push the Buildspec

Commit and push the buildspec.yml file to CodeCommit.

git add buildspec.yml

git commit -m "Added buildspec.yml for CodeBuild"

git push origin master

Step 5: Setting up the Deployment Pipeline – CodePipeline

CodePipeline manages the entire release process, from the source to build and deployment stages.

5.1 Create a New Pipeline

- **AWS Console**, go to *CodePipeline*.
- Click **Create pipeline**.
- **Pipeline Name:** Name it *MyWebsiteDeploymentPipeline*.
- **Service Role:** Choose New service role. (AWS will create an IAM role for CodePipeline).

5.2 Stage 1: Source

- **Source Provider:** Choose *AWS CodeCommit*.
- **Repository Name:** Select *MyWebsiteRepo*.
- **Branch Name:** Choose *master*.

- **Change Detection:** Choose *AWS CodePipeline*. It leverages CloudWatch Events to detect changes immediately.

5.3 Stage 2: Build

- **Build Provider:** Choose *AWS CodeBuild*.
- **Region:** Choose your *region*.
- **Project Name:** Select *MyWebsiteBuildProject*.

5.4 Stage 3: Deploy

- **Deploy provider:** Select *Amazon S3*.
- **Region:** Select your *region*.
- **Bucket:** Choose your S3 bucket (*my-devops-website-app-12345)*.
- **Extraction:** Verify Extract file before deploy – this unzips the artifact from CodeBuild into the bucket.

5.5 Action: Create the Pipeline

Review and click **Create pipeline**.

Step 6: Testing the Full CI/CD Pipeline

The pipeline will immediately trigger upon creation using the latest code coming from CodeCommit.

6.1 Tracking the Initial Run

CodePipeline Console: Observe the graphical flow. You should see the pipeline move through:

- **Source:** Success; CodeCommit pulls the code.
- **Build:** Success – CodeBuild zips the files.
- **Deploy:** Success (CodePipeline extracts the files to S3).

6.2 Validate the Website

- **Action:** From your S3 bucket properties (Step 3.2), copy the Endpoint URL and paste that into your browser.
- **Expected Result:** You can now see the content of your *index.html* file.

6.3 Test Continuous Delivery

1. **Local Computer:** Make a minor edit to index.html, for instance edit the title or paragraph text.

<h1>SUCCESS! This is the automated deployment V2!</h1>

* Pipeline triggered at: [Current Timestamp] *

2. **Commit and Push:**

git add .

git commit -m “Updated website content for V2”

git push origin master

3. **Monitor:** Verify the CodePipeline console. A very new execution should begin automatically.
4. **Verify:** Once the pipeline displays success, refresh the S3 website endpoint. The fresh V2 content from CodeCommit to deployment in the cloud.

You don’t just need to get comfortable with basic static deployments; you need to start tackling real-world complexity in order to reinforce your skills.

Download the complete **AWS DevOps Challenges and Solutions** Guide! Practice exercises involving creating environment-specific S3 buckets, adding Lambda functions

for serverless deployments, and using CloudFormation or Terraform for Infrastructure as Code (IaC), with comprehensive solutions.

Real Time Examples for AWS DevOps Tutorial for Beginners

The mastering of DevOps tools on AWS allows for scale and speed in the deployment and management of applications on the cloud.

Immutable Infrastructure Deployment – CodePipeline & CloudFormation Solution

- **Scenario:** The company should launch a new version of its application on a consistent set of new servers, not altering the existing ones.
- **DevOps on AWS Applied:**
    - **CodePipeline** orchestrates the process; this means that **CodeCommit** (source) triggers **CodeBuild** (build).
    - The **CD stage** then provisions a new, clean **Autoscaling Group or ECS cluster** using an **AWS CloudFormation** template-
        - **Ex:** Infrastructure as Code.
    - **CodeDeploy** deploys the application after the new infrastructure is deemed healthy.
- **Goal:** Eliminate configuration drift and ensure environment consistency from development to production.

Serverless Function CI/CD

- **Scenario:** A team develops dozens of small serverless functions, AWS Lambdas, that need to be independently deployed quickly without managing any servers.
- **AWS DevOps Applied:**
    - Source code for each Lambda function is in **CodeCommit**.

    - **CodePipeline** detects changes, and CodeBuild packages the function code along with all dependencies.
    - Deployment by **CodeDeploy** (or Serverless Application Model – SAM) ensures zero downtime during the update of the function.
- **Goal:** Enable rapid iteration and deployment for microservices and event-driven architectures.

Containerized Application Release: EKS/ECS & CodePipeline

- **Scenario:** A large application running inside Docker containers managed by Amazon EKS (Kubernetes) or ECS requires new container images to be built and then pushed to the registry.
- **AWS DevOps Applied:**
    - **CodeBuild** compiles the code and builds a new Docker image, then pushes that to Amazon ECR (Elastic Container Registry).
    - The **CodePipeline** automatically updates the EKS or ECS service definition, commanding the orchestrator to pull the new container image and perform the rollout.
- **Goal:** Automate the lifecycle of a container from source code to a running service. Ready to build and deploy your own cloud services?

Download our curated list of **AWS DevOps Project Ideas** and get moving on automating IaC, serverless, and container deployments!

FAQs About AWS DevOps Tutorial for Beginners

1. Is AWS or DevOps easy to learn?

Neither is inherently “easy,” but AWS would probably be more structured to learn in the beginning because it’s focused on services and configuration. DevOps, on the other hand, is a broad methodology that requires mastery over coding, scripting, multiple tools like Git, Jenkins, Docker, and cloud platforms such as AWS.

2. Can I learn AWS in 1 week?

One week is too little to master AWS. You can learn the basics, work your way around the console, and understand core services like S3 and EC2. True knowledge requires deep, hands-on practice across networking, security (IAM), databases, and infrastructure automation.

3. How difficult is AWS to learn?

AWS is not easy to learn because its scope is so huge and always changing. The difficulty lies in the breadth of services-over 200. It requires consistent hands-on practice to understand how to combine services securely and cost-effectively, not just memorizing features.

4. Can we learn DevOps without AWS?

Yes, you can. DevOps is a methodology that can be applied to any platform: Azure, Google Cloud, on-premises. But since cloud platforms like AWS are standard for modern deployments, learning DevOps with AWS is highly recommended as far as job readiness is concerned.

5. Is DevOps heavy coding?

DevOps involves heavy scripting and automation rather than heavy coding, which is involved in developing a complex application. You need to be good at languages like Python, Bash, or PowerShell to automate infrastructure (IaC) and configure CI/CD pipelines.

6. Is DevOps stressful job?

It can be stressful. DevOps engineers are often on-call and responsible for the stability and reliability of production systems. The rapid pace of deployment cycles further increases the level of stress due to the pressure of quickly troubleshooting failures in complex environments.

7. What are the 7 phases of DevOps?

These 7 phases represent the continuous lifecycle: Plan, Code, Build-Continuous Integration, Test, Release-Continuous Delivery, Deploy, Operate/Monitor-Continuous Monitoring and Feedback. This cycle ensures continuous improvement and automation.

8. Can a fresher learn AWS?

Yes, a fresher can and should learn AWS. They should start by understanding concepts with Cloud Practitioner and then move on to Solutions Architect – Associate level. This shows their valuable set of skills for an entry-level cloud or development role.

9. What is the AWS job salary?

AWS DevOps salary for freshers in India are very competitive, from entry-level Cloud Engineer positions to highly paid Cloud Architects. Based on different certifications, years of experience, and locations, US senior roles can get up to $150,000+ annually.

10. Which job is best in AWS?

The best job is subjective, but AWS Solutions Architect and AWS DevOps Engineer are generally the most in-demand and highest-paying. These require deep knowledge of multiple services, architecture design, and automation skills.

Conclusion

Well Done! You have completed the AWS DevOps core journey, from automating the flow from CodeCommit to CodePipeline, through deploying to S3. You have mastered the essential concepts of CI/CD that are germane to any cloud career. This proficiency with native AWS tools will also set you up for real-world challenges like Immutable Infrastructure and Serverless deployments. Now you should take your knowledge to the next level with more complex tools like CloudFormation and EKS.

Ready to become an expert in automated cloud delivery? Enroll in our advanced **AWS DevOps Course in Chennai** to master the use of IaC, security, and complex multi-region deployments!