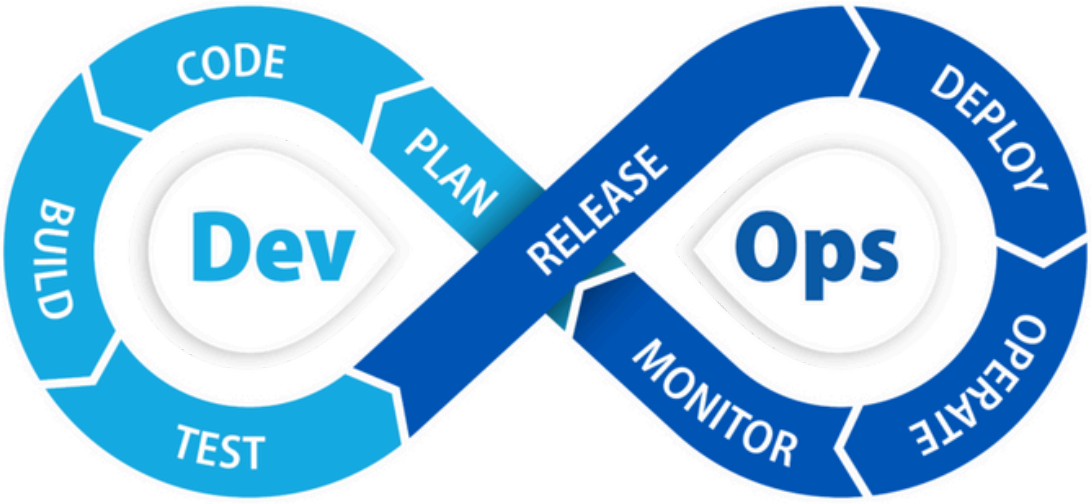
Notes

AWS Project Documentation

Building a Complete CI/CD Pipeline

(Java Web Application Deployment on AWS with DevOps Automation)



@sAnandha

Notes

SET UP A WEB APP IN THE CLOUD

What is CI / CD ?

CI/CD (Continuous Integration and Continuous Deployment) is a DevOps process where developers frequently push code to a shared repository, which then automatically goes through testing, building, and deployment steps. This helps deliver updates faster and with fewer errors.

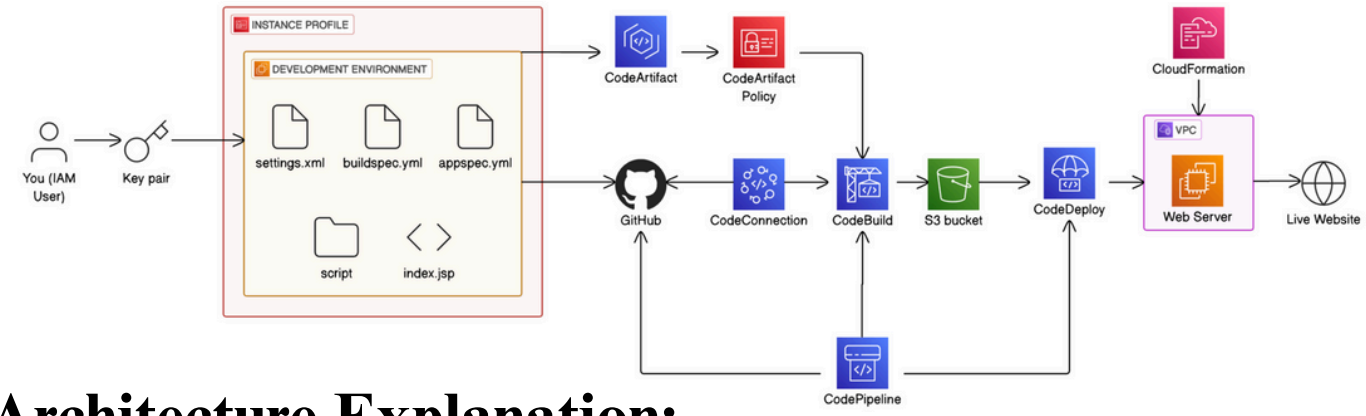
Example: Imagine you're updating your personal portfolio site. When you commit the change to GitHub, a CI/CD pipeline automatically checks for errors, builds the site, and deploys it to your AWS S3 bucket. Your site is updated online in seconds—no manual work needed!

Some key uses of CI/CD:

1. ✅ **Faster Releases** – Automates testing and deployment so updates reach users quickly.
2. 🐞 **Early Bug Detection** – Catches errors during integration before they go live.
3. 👥 **Team Collaboration** – Helps multiple developers work on the same codebase safely.
4. 🔒 **Improved Reliability** – Reduces the risk of human error during manual deployments.

Notes

ARCHITECTURE DIAGRAM :



Architecture Explanation:

- **Development Stage**: IAM User accesses Development Environment using key pairs, containing configuration files (settings.xml, buildspec.yml, appspec.yml) and application code (script, index.jsp)
- **Version Control**: Code is pushed to GitHub repository which serves as the central source code management system
- **Artifact Management**: CodeArtifact stores and manages application dependencies and build artifacts, controlled by CodeArtifact Policy for access permissions
- **CI/CD Pipeline**: CodePipeline orchestrates the workflow, with CodeConnection linking GitHub to CodeBuild for automated compilation and testing
- **Storage & Deployment:** Build artifacts are stored in S3 bucket, then CodeDeploy automatically deploys the application to target infrastructure
- **Infrastructure & Delivery**: CloudFormation provisions VPC and Web Server resources, making the application accessible as a Live Website to end users

Notes

Tools and Services Used :

👤 IAM User

- AWS Identity and Access Management (IAM) lets you securely manage access to AWS services.
- Used by developers to authenticate and access AWS resources.
- Generates credentials and key pairs for secure SSH access.
- Important for assigning roles, permissions, and policies in CI/CD.

🔑 Key Pair

- Used to securely connect to EC2 instances via SSH.
- Ensures that only authorized users can access the development environment.
- Stored safely by the developer for login.
- Paired with the IAM role for enhanced security.

💻 Development Environment (Local Machine or EC2)

- Where developers write and test code.
- Contains files like index.jsp, buildspec.yml, appspec.yml, settings.xml, and shell scripts.
- Uses tools like VS Code for development.
- Prepares code and configurations for CI/CD pipeline.

Notes

📄 Files Used

- **index.jsp**: A Java-based web page.
- **buildspec.yml**: Instructions for AWS CodeBuild (compile/build process).
- **appspec.yml**: Instructions for AWS CodeDeploy (deployment steps).
- **settings.xml**: Configuration for Maven or Java project dependencies.
- **script**: Shell script used for automation during build/deploy.

👨‍💻 VS Code

- Code editor for writing and managing the application.
- Used to modify index.jsp, .yml, .xml, and shell scripts.
- Offers version control features to push code to GitHub.

🐙 GitHub

- Source code repository where your project is stored.
- Acts as the trigger source for the CodePipeline.
- Integrates with CodeConnections to securely connect with AWS.
- Ensures version control and history tracking.

Notes

🔗 CodeConnection

- Connects GitHub securely to AWS CodePipeline.
- Enables automatic detection of code changes (webhooks).
- Ensures secure OAuth-based integration.
- Key to integrating external GitHub repos into AWS CI/CD.

🧰 CodeBuild

- Fully managed build service that compiles source code and runs tests.
- Uses buildspec.yml to define the build commands.
- Pulls dependencies from CodeArtifact.
- Stores the build output (artifact) in S3.

📦 CodeArtifact

- Secure artifact (package) repository used by CodeBuild.
- Stores dependencies (e.g., Maven packages for Java).
- Ensures versioned, safe, and internal package distribution.
- Controlled by CodeArtifact Policy for secure access.

Notes

🪣 S3 Bucket

- Used to store build artifacts (e.g., WAR/JAR files, zipped websites).
- Acts as a central location before deployment.
- CodeDeploy pulls code from here.
- Can also host static websites or be versioned.

🚀 CodeDeploy

- Deploys code to EC2 instances or on-prem servers.
- Uses appspec.yml to define deployment logic (e.g., install, stop/start services).
- Works alongside EC2 and S3.
- Ensures zero-downtime deployments with rollback support.

🔧 CodePipeline

- Orchestrates the entire CI/CD workflow.
- Connects GitHub → CodeBuild → S3 → CodeDeploy.
- Automatically triggers builds and deployments on code updates.
- Helps maintain fast and reliable delivery of applications.

Notes

🧱 CloudFormation

- Infrastructure as Code (IaC) service that creates AWS resources from templates.
- Used to automatically provision VPC, EC2, IAM Roles, and more.
- Ensures reproducibility and reduces manual setup.

🧠 VPC (Virtual Private Cloud)

- Isolated network for your EC2 instances.
- Provides control over IP range, subnets, and firewall rules.
- Ensures security and connectivity to the internet/load balancers.

🖥️ Web Server (EC2 Instance)

- Hosts the deployed application (like index.jsp).
- Runs inside the VPC for security.
- Receives updated code via CodeDeploy.
- Forms the backbone of your live website.

🌐 Live Website

- Final output of the whole CI/CD pipeline.
- Hosted on EC2 or can be served via CloudFront if needed.
- Reflects latest committed and deployed code from GitHub.

Notes

Full CI/CD Web App Deployment Pipeline on AWS

This project involves building a Java web application, storing it in GitHub, managing its packages securely with CodeArtifact, compiling with CodeBuild, deploying with CodeDeploy, and automating everything using CodePipeline.

🔶 Phase 1: Set Up a Web App in the Cloud

📌 https://learn.nextwork.org/projects/aws-devops-vscode

Description:

Learn how to set up your development environment, build a Java-based web application, and prepare it for deployment in the cloud using AWS EC2.

✅ Key Outcomes:

- Create and test a Java web app locally
- Connect with AWS using IAM credentials
- Set up EC2 instance to host your application
- SSH into EC2 and run sample apps

🔶 Phase 2: Connect a GitHub Repo with AWS

📌 https://learn.nextwork.org/projects/aws-devops-github

Description:

Securely connect your GitHub repository with AWS services to trigger automated builds and deployments.

Notes

✅ **Key Outcomes:**

- Link GitHub with AWS CodePipeline
- Create and manage CodeConnections
- Automatically trigger pipeline on code push
- Version control integration with CI/CD

🔶 **Phase 3: Secure Packages with CodeArtifact**

📌 https://learn.nextwork.org/projects/aws-devops-codeartifact-updated

Description:

Store and manage Java dependencies securely using AWS CodeArtifact, and integrate them into your Maven build process.

✅ **Key Outcomes:**

- Create CodeArtifact domain and repo
- Configure Maven (settings.xml) to use CodeArtifact
- Upload and pull dependencies securely
- Integrate with CodeBuild for secure dependency management

🔶 **Phase 4: Continuous Integration with CodeBuild**

📌 https://learn.nextwork.org/projects/aws-devops-codebuild-updated

Description:

Set up AWS CodeBuild to automate compiling and packaging of your application, creating production-ready artifacts.

Notes

✅ **Key Outcomes:**

- Configure buildspec.yml for build steps
- Compile Java code and generate WAR/ZIP
- Upload artifacts to S3 for deployment
- View real-time build logs via CloudWatch

🔶 **Phase 5: Deploy a Web App with CodeDeploy**

📌 https://learn.nextwork.org/projects/aws-devops-codedeploy-updated

Description:

Use AWS CodeDeploy to automate the deployment of your application to EC2 instances, with lifecycle hooks and rollback support.

✅ **Key Outcomes:**

- Write appspec.yml for deployment logic
- Deploy app to EC2 using CodeDeploy agent
- Monitor deployment status and logs
- Handle post-deploy steps and rollback if needed

🔶 **Phase 6: Build a CI/CD Pipeline with AWS**

📌 https://learn.nextwork.org/projects/aws-devops-codepipeline-updated

Description:

Automate the entire workflow with CodePipeline, integrating GitHub (source), CodeBuild (build), and CodeDeploy (deploy) into one continuous delivery system.

✅ **Key Outcomes:**

- Design and deploy a complete CI/CD pipeline
- Automate source → build → deploy sequence
- Monitor pipeline execution stages
- Enable full DevOps lifecycle with minimal manual steps

Notes

SAMPLE OUTPUT :