Service Name

AWS CodeBuild

Service Overview

AWS CodeBuild is a fully managed build service in the cloud that compiles source code, runs unit tests, and produces artifacts ready for deployment. It eliminates the need to provision, manage, and scale build servers while providing prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. CodeBuild automatically scales to meet peak build demands and charges based on build minutes consumed. The service integrates seamlessly with AWS CodePipeline for continuous delivery workflows and supports various source providers including GitHub, Bitbucket, AWS CodeCommit, and Amazon S3. It offers preconfigured environments for Microsoft Windows and Linux, with the flexibility to bring custom build environments using Docker containers.

Key Use Cases

- Continuous Integration: Automatically compile source code, run tests, and validate code quality as part of CI/CD pipelines
- Automated Testing: Execute unit tests, functional tests, and integration tests with visual reporting of test case results
- Multi-Environment Builds: Build applications across different programming languages and platforms using preconfigured or custom Docker environments
- Parallel Processing: Run test suites concurrently using parallel test execution to significantly reduce build times
- Container Image Building: Build and publish Docker images to Amazon ECR with dedicated Docker server capabilities and persistent caching

Features

- **Fully Managed Build Service**: Eliminates need to set up, patch, update, and manage build servers with automatic scaling
- **Multiple Source Providers**: Supports GitHub, GitHub Enterprise, Bitbucket, AWS CodeCommit, and Amazon S3 as source providers
- **Preconfigured Build Environments**: Provides ready-to-use environments for popular programming languages and build tools
- **Custom Docker Environments**: Allows bringing customized build environments as Docker containers
- **Parallel Test Execution**: Enables running test suites concurrently to reduce build times significantly

- **Test Reporting**: Generates comprehensive reports for unit, functional, and integration tests with visual results
- **VPC Integration**: Can operate within Amazon VPC for integration with private services and databases
- **Docker Server Capability**: Provides dedicated Docker server with persistent cache for faster container builds
- **Build Caching**: Supports caching of build dependencies to improve performance across builds

Value Proposition

AWS CodeBuild provides significant advantages over self-managed build infrastructure by eliminating server management overhead and offering automatic scaling without capacity planning. The pay-per-use model ensures cost efficiency as customers only pay for build minutes consumed rather than maintaining idle infrastructure:

- CodeBuild's seamless integration with AWS developer tools creates comprehensive CI/CD pipelines, while preconfigured environments for popular languages reduce setup time
- Advanced features like parallel test execution, Docker server capabilities, and persistent caching dramatically improve build performance
- The service's ability to operate within VPCs enables secure integration with existing infrastructure, making it ideal for enterprises requiring both flexibility and security in their build processes

Service Integration

AWS CodeBuild integrates deeply with AWS CodePipeline as a build and test stage in continuous delivery workflows, enabling automated code release processes. It works seamlessly with AWS CodeCommit for source control and AWS CodeArtifact for artifact management:

- Integration with Amazon S3 provides artifact storage and source input capabilities, while AWS Identity and Access Management (IAM) controls access to builds and projects
- Amazon CloudWatch and CloudWatch Logs provide comprehensive monitoring and logging of build activities
- CodeBuild also integrates with Amazon SNS for build notifications, AWS Systems Manager Parameter Store and AWS Secrets Manager for secure environment variable storage, and can operate within Amazon VPC for private resource access
- Additional integrations include Amazon ECR for container image storage and AWS Lambda for serverless build operations

Onboarding and Offboarding

Getting started with AWS CodeBuild involves creating a build project that defines source code location, build environment, and output settings. Customers begin by planning their build requirements, including determining source repository providers, build commands, required runtimes and tools, and any additional AWS resources needed:

- The service supports source code from CodeCommit, Amazon S3, GitHub, and Bitbucket
- Users create a buildspec file in YAML format that instructs CodeBuild on build transformation processes
- Build projects can be created through the AWS console, CLI, or SDKs, with options to configure environment settings, artifact outputs, and logging
- CodeBuild provides step-by-step tutorials for common scenarios, including Java applications with Maven, demonstrating the complete build lifecycle from source upload to artifact retrieval

Getting Started Guide: https://docs.aws.amazon.com/codebuild/latest/userguide/getting-started-overview.html

Data Removal

When offboarding from AWS CodeBuild, customers can delete build projects, builds, and associated resources using the AWS console, CLI, or SDKs. Build projects with existing builds and resource policies require removal of the policy and deletion of builds before project deletion. The batch-delete-builds API allows deletion of multiple builds simultaneously. All build artifacts stored in S3 buckets, CloudWatch logs, and test reports are retained according to their respective service retention policies and must be managed separately. Complete data removal requires deleting all associated resources including IAM roles, S3 artifacts, and CloudWatch log groups created for CodeBuild operations.

Backup/Restore and Disaster Recovery

AWS CodeBuild inherently provides resilience through its serverless, fully managed architecture that operates across multiple Availability Zones. Build configurations and project definitions are automatically replicated across AWS infrastructure. For disaster recovery, customers should implement Infrastructure as Code using AWS CloudFormation or CDK to recreate CodeBuild projects and configurations. Source code repositories and build artifacts should be backed up separately through their respective services. Build history and logs are retained according to CloudWatch retention policies, and customers can configure cross-region replication for critical build artifacts stored in S3 buckets.

Backup Capabilities

CodeBuild does not directly integrate with AWS Backup service as it is a serverless, fully managed service. However, the service maintains build project configurations and metadata through AWS's standard infrastructure resilience:

- Customers are responsible for backing up source code repositories, build artifacts in S3, and any custom Docker images in ECR
- Build specifications and project configurations should be version-controlled and backed up through Infrastructure as Code practices
- CloudWatch logs and build history are retained according to configured retention periods, providing historical build information for recovery purposes

Disaster Recovery

CodeBuild does not directly integrate with AWS Elastic Disaster Recovery as it is a serverless service with inherent multi-AZ resilience. The service automatically handles infrastructure failures and provides high availability through AWS's managed infrastructure:

- For comprehensive disaster recovery, customers should implement cross-region strategies by replicating build projects using CloudFormation templates, maintaining source code repositories in multiple regions, and storing critical build artifacts in geographically distributed S3 buckets
- Build configurations can be quickly recreated in alternate regions using Infrastructure as Code approaches, ensuring business continuity during regional outages

Recovery Objectives

CodeBuild supports customer RPO and RTO objectives through its serverless architecture that eliminates infrastructure recovery time. Build projects and configurations can be instantly recreated from Infrastructure as Code templates, achieving near-zero RTO for project recreation. Source code and artifacts should be replicated according to business requirements to meet RPO objectives. The service's integration with multi-region source repositories and S3 cross-region replication enables customers to maintain low RPO targets. For critical builds, customers can implement automated cross-region project creation and maintain standby configurations to minimize recovery time during disaster scenarios.

Service Constraints

Service Quotas: https://docs.aws.amazon.com/codebuild/latest/userguide/limits.html

FAQ: https://aws.amazon.com/codebuild/faqs/

Technical Requirements

Service Documentation:
https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html

User Guide: https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html

API Reference:
https://docs.aws.amazon.com/codebuild/latest/APIReference/Welcome.html

Pricing Overview

Please see the AWS UK G Cloud 15 Pricing Document accompanying this service in the Digital Marketplace.

Public Pricing: https://aws.amazon.com/codebuild/pricing/

Cost Optimization

AWS CodeBuild offers several unique cost optimization opportunities through its pay-per-use pricing model where customers only pay for build minutes consumed. Reserved capacity instances provide cost savings for predictable workloads with minimum usage commitments:

- The service's auto-scaling eliminates costs of idle infrastructure common with self-managed build servers
- Parallel test execution reduces total build time, directly decreasing costs
- Docker server capability with persistent caching significantly reduces build times for container-based workflows
- Customers can optimize costs by selecting appropriate compute types, using efficient build scripts, implementing build caching strategies, and leveraging the free tier offering 100 build minutes monthly for small instances
- Strategic use of spot pricing for non-critical builds and proper resource right-sizing further optimize expenses

Savings Considerations

Primary cost savings with CodeBuild come from eliminating the need to provision, manage, and maintain dedicated build infrastructure, reducing both capital and operational expenses. The pay-per-use model ensures customers avoid costs of idle build servers, paying only for actual build time consumed:

- Automatic scaling prevents over-provisioning while meeting peak demands efficiently
- Advanced features like parallel test execution and Docker server caching dramatically reduce build durations, directly translating to lower costs

- The service eliminates server maintenance, patching, and update costs while providing enterprise-grade build capabilities
- Integration with other AWS services creates operational efficiencies that reduce overall development lifecycle costs
- Organizations typically see 30-60% cost reduction compared to self-managed build infrastructure while gaining improved performance and reliability