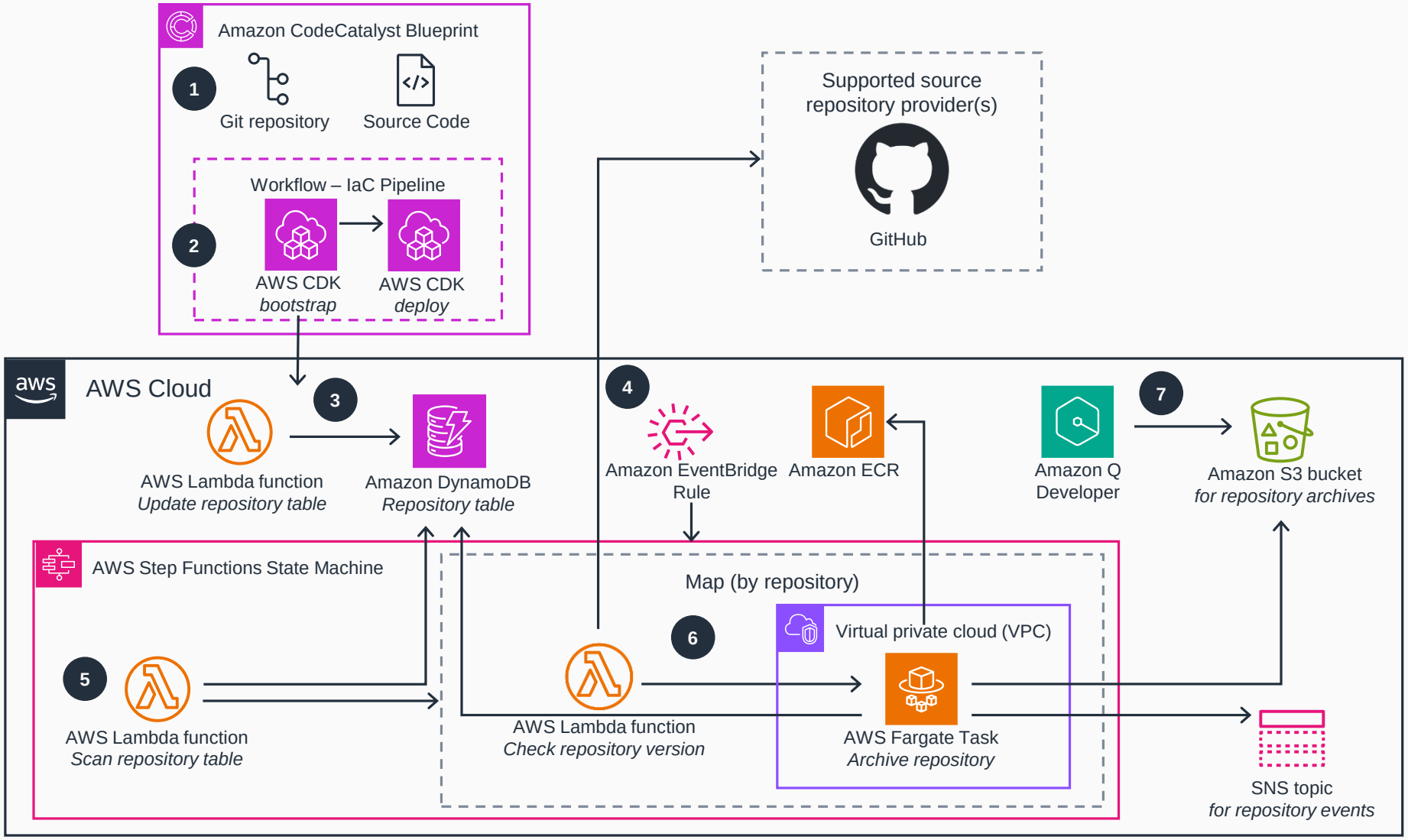
# Guidance for Creating a Customized Coding Companion with Amazon Q Developer

This architecture diagram shows how to aggregate public, open-source repositories for an Amazon Q Developer enterprise




1. Initialize a new project in **Amazon CodeCatalyst** with this blueprint to create a new Git repository. It is preconfigured with the source code and a workflow to deploy to an AWS account environment.

2. All changes in the Git repository invoke a workflow which builds and deploys all infrastructure-as-code (IaC) related to the architecture, using an **AWS Cloud Development Kit (AWS CDK)**, which includes a command line interface (CLI) tool for working with your **AWS CDK** apps and stacks.

3. An **AWS Lambda** trigger function examines which repositories should be included in the archival process, and updates metadata in the **Amazon DynamoDB** repository table.

4. An **Amazon EventBridge** rule invokes an **AWS Step Functions** state machine to process on a regular interval.

5. A **Lambda** function scans the repository table in **DynamoDB** for enabled repositories, and starts a parallel process for each repository using a Map state.

6. For each repository in the Map state, a **Lambda** function checks open-source providers (such as GitHub) for a new release. If a new release is available, a subsequent **AWS Fargate** task is launched within a customer-managed virtual private cloud (VPC). It uses an image from **Amazon Elastic Container Registry (Amazon ECR)**, downloads, and extracts the release archive in an **Amazon Simple Storage Service (Amazon S3)** bucket. The task updates the repository table in **DynamoDB**. A message is also published to an **Amazon Simple Notification Service (Amazon SNS)** topic.

7. An **Amazon Q Developer** customization can be trained using the repository archives in **Amazon S3**, allowing developers to use AI-generated code suited to their coding language and style.

*Reviewed for technical accuracy October 30, 2023*


**AWS Reference Architecture**