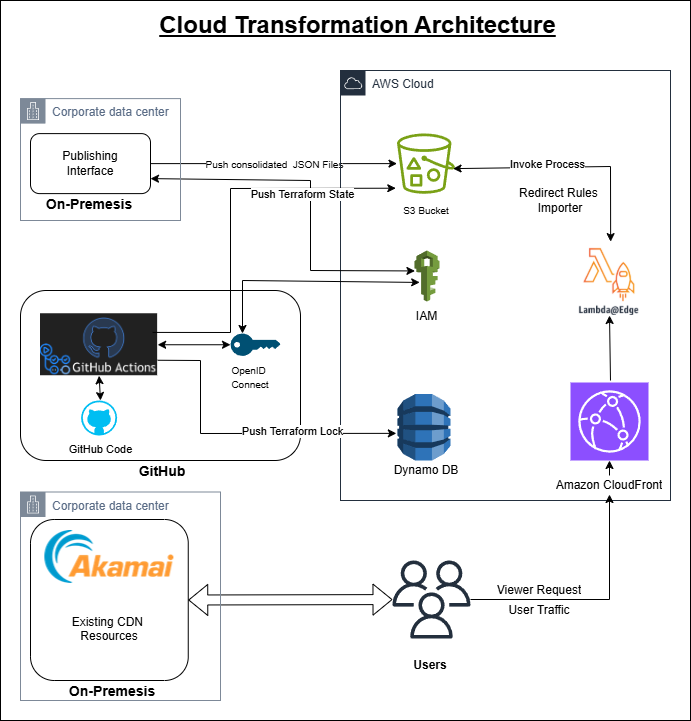
**Architecture - Flow Explained**



The diagram depicts the **"Cloud Transformation Architecture"**, which integrates various components like **GitHub**, **AWS Cloud services**, **Akamai CDN**, and **corporate on-premises infrastructure** to manage and transform cloud-based workflows efficiently. Here's a breakdown of the architecture:

**Components and Flow**

**1. Corporate Data Center (On-Premises)**

* **Publishing Interface**:
  + On-premises users use the Publishing Interface to push configurations (like JSON files) for cloud and CDN transformations.
  + These configurations are consolidated and sent to the AWS S3 bucket for further processing.
* **Akamai (Existing sample.com CDN Resources)**:
  + Akamai serves as the legacy Content Delivery Network (CDN) for sample.com, delivering static and cached content to users. It integrates with the new architecture but exists outside the AWS Cloud.

**2. GitHub (Code and Automation)**

* **GitHub Code**:
  + Acts as the central repository for managing Infrastructure as Code (IaC) using **Terraform**.
  + Developers push changes (e.g., Terraform scripts) to GitHub.
* **GitHub Actions**:
  + CI/CD automation workflows are triggered here to perform tasks like validating, packaging, and deploying Terraform changes.
  + **OpenID Connect** is used to securely authenticate GitHub workflows with AWS services without needing long-lived credentials.
* **Pushing Terraform Files**:
  + Terraform state files and lock files (used to track infrastructure states and dependencies) are pushed to AWS services for centralized management.

**3. AWS Cloud Services**

* **S3 Bucket**:
  + Receives consolidated JSON files from the Publishing Interface.
  + Stores these files, which are consumed by the **Redirect Rules Importer** to configure redirection rules in CloudFront and Lambda@Edge.
* **IAM**:
  + Provides fine-grained access control for different AWS components, ensuring secure interaction between GitHub, S3, DynamoDB, and Lambda@Edge.
* **DynamoDB**:
  + Stores the Terraform lock file, ensuring distributed consistency and state management when multiple teams or processes work on the same infrastructure.
* **Lambda@Edge**:
  + Processes incoming viewer requests at the CloudFront edge locations.
  + Executes custom logic (e.g., applying redirect rules from JSON files, personalizing content) closer to users for better performance.
* **Amazon CloudFront**:
  + Serves as the primary Content Delivery Network (CDN) for sample.com.
  + Distributes content globally with low latency.
  + Integrates with Lambda@Edge to dynamically handle requests and responses based on redirect rules or custom logic.

**4. Users**

* Users make requests to sample.com via CloudFront.
* CloudFront processes these requests with the help of Lambda@Edge (for redirects or custom handling) and responds with content.
* Some user traffic might still go through Akamai (legacy CDN).

**Workflow Summary**

1. **Publishing & State Management**:
   * Corporate users push configurations via the Publishing Interface.
   * GitHub manages Terraform IaC, and workflows ensure updates are pushed to S3, DynamoDB, and other services securely.
2. **Rule Import and Processing**:
   * Consolidated JSON files are stored in S3.
   * The **Redirect Rules Importer** fetches these files and applies redirect logic using Lambda@Edge.
3. **Content Delivery**:
   * Users' requests are routed through Amazon CloudFront.
   * Lambda@Edge dynamically processes requests (e.g., applying redirect rules or adding custom headers).
   * CloudFront delivers content globally with low latency.
4. **Legacy Integration**:
   * Akamai serves as a fallback for existing CDN resources not yet migrated to the new CloudFront-based architecture.

**Key Features of the Architecture**

* **Seamless Integration**:
  + Combines on-premises systems, GitHub workflows, and AWS services for efficient cloud transformation.
* **Automation and Security**:
  + GitHub Actions and OpenID Connect ensure secure and automated deployment pipelines.
* **Global Reach**:
  + Amazon CloudFront and Lambda@Edge ensure low-latency content delivery and dynamic request handling at edge locations.
* **State Management**:
  + DynamoDB and Terraform lock files ensure consistent management of infrastructure.