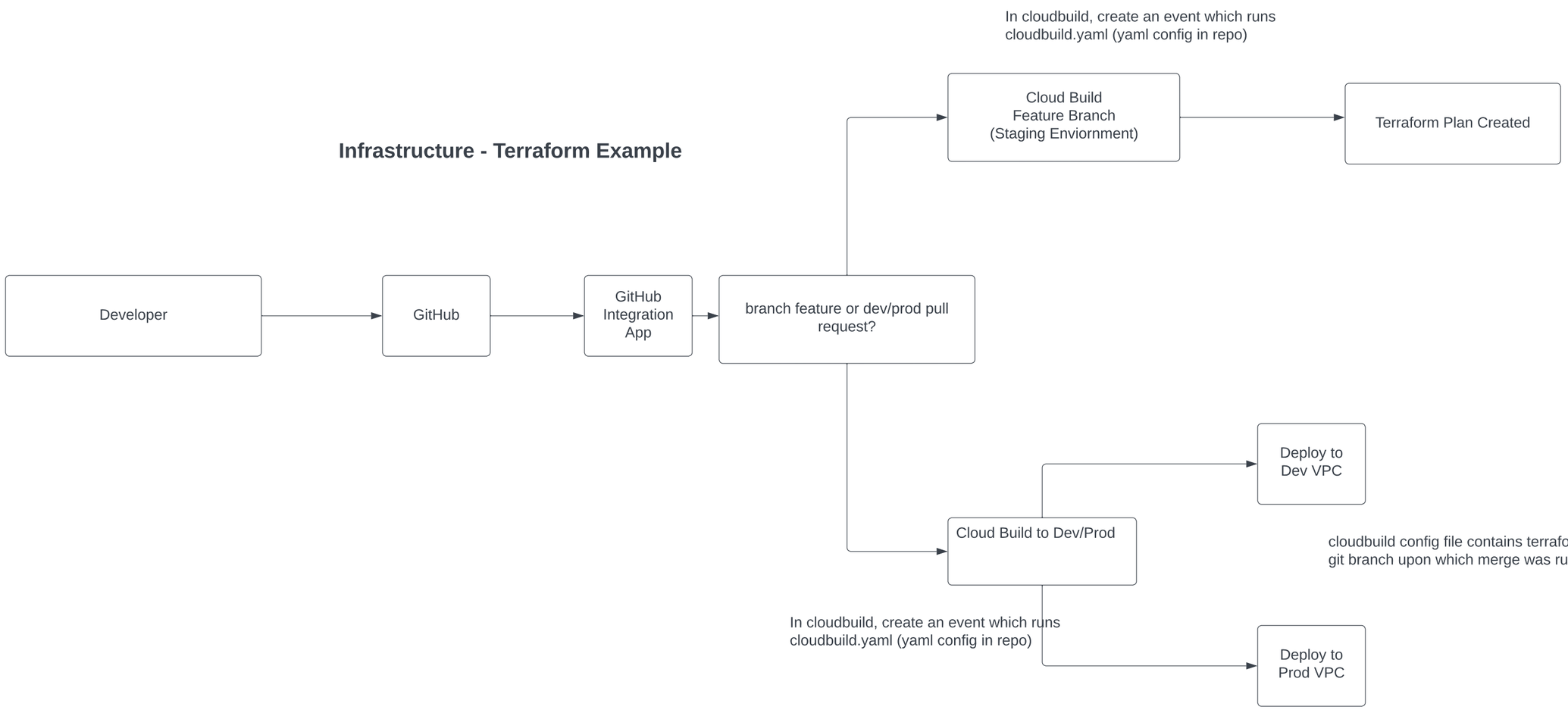
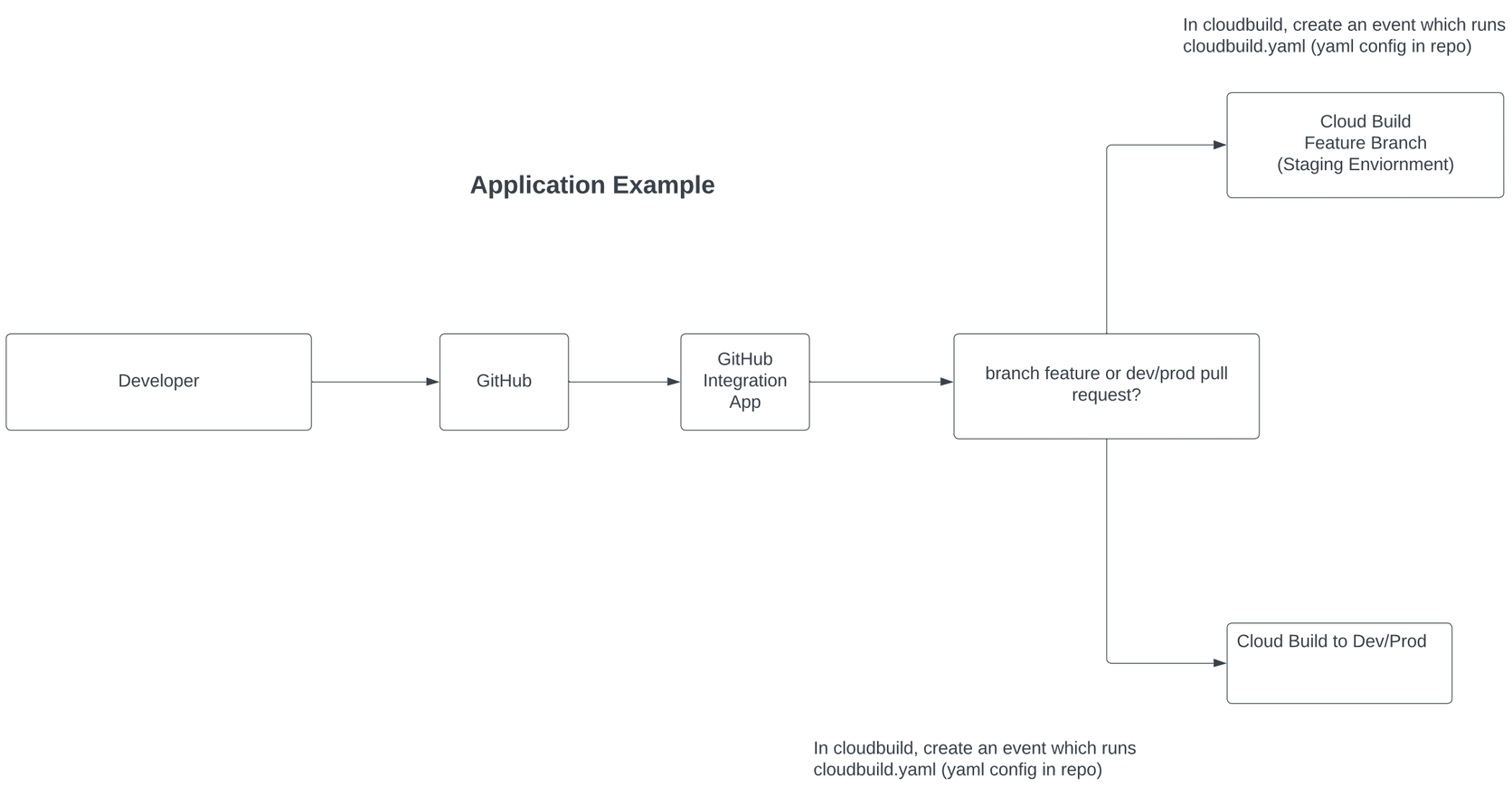
CI/CD Infrastructure and Application Pipelines
Sean Corzo 8/2024

Infrastructure - Terraform Example



Application Example



Staging Environment Setups for Feature Branches

Development teams often use alternative approaches to provide isolated environments for individual developers or feature branches:

Shared Development/Staging Environments: Developers often share a common development or staging environment that mimics the production setup. This reduces complexity and allows developers to collaborate effectively.

Branch-Based Isolation: CI/CD tools and infrastructure can be set up to deploy feature branches to separate isolated environments within a shared VPC. This isolates the changes from different developers while still utilizing the same networking infrastructure.

Docker Containers: Developers can use Docker containers or Kubernetes pods to encapsulate their applications along with their dependencies, which can run in shared environments without requiring separate VPCs.

Temporary Environments: For testing purposes, developers can provision temporary environments or resource instances within an existing VPC. These resources can be spun up and torn down as needed.

IAM and Access Control: Access control using IAM can ensure that developers have appropriate permissions to only the resources they are working on, while still sharing the same VPC.

Cloud Build typical Build/Deploy

When someone publishes a feature branch in a CI/CD application pipeline using Google Cloud Build, the typical setup involves the following steps:

Code Push to Version Control: A developer pushes code changes to a feature branch in the version control system (e.g., Git).

Trigger Setup: A Cloud Build trigger is set up to monitor the repository for changes. Triggers can be configured to respond to specific events, like a new commit or a pull request.

Build Configuration (cloudbuild.yaml): The repository contains a configuration file named `cloudbuild.yaml` that defines the steps to be executed when a build is triggered. This configuration specifies how the code should be built, tested, and deployed.

Build Process: When the trigger is activated due to code changes in the feature branch, Cloud Build reads the `cloudbuild.yaml` file and follows the defined instructions. The build process typically includes steps such as:

Checking out code: Cloning the repository and checking out the specific commit or branch.

Building: Compiling code, generating artifacts, and preparing the application for testing and deployment.

Testing: Running automated tests to ensure the quality and functionality of the code changes.

Packaging: Creating deployment-ready packages or artifacts.

Optional Steps: Running additional tasks like linting, code analysis, security scanning, etc.

Deployment (Optional): Depending on the CI/CD pipeline's configuration, Cloud Build can trigger a deployment to a staging environment or even directly to production. Deployment might involve deploying containers to Google Kubernetes Engine (GKE), updating serverless functions, deploying App Engine applications, etc.

Notifications and Reporting: Cloud Build provides notifications and reporting features that can alert developers, teams, or stakeholders about the build status and any issues that might arise during the process. This can include success, failure, or warnings.

Rollback (Optional): In case of issues discovered post-deployment, the pipeline can be configured to automatically trigger a rollback to the previous working version of the application.