# Domain 2: Building and Testing Applications - 13%

## 2.1 Use Google Cloud SDK and Command Line Tools

- Common Tools
  - gcloud: Manage GCP resources via CLI (deploy apps, configure IAM)
  - gsutil: Manage Cloud Storage buckets and objects
  - bq: Interact with BigQuery datasets, run SQL queries
  - firebase: Deploy and manage Firebase projects and hosting
- Best Practices
  - Use gcloud CLI for scripting infrastructure automation
  - Set project config using `gcloud config set project`
  - Automate authentication using service account key files (`gcloud auth activate-service-account`)

## 2.2 Write and Execute Unit and Integration Tests

- Types of Tests
  - Unit Test: Test individual functions; tools include JUnit, Mocha, pytest
  - Integration Test: Test components together (e.g., DB + API); tools include Postman, REST Assured
  - End-to-End (E2E): Simulate real user flow; tools include Cypress, Selenium
- GCP Testing Environments
  - Use emulators for Firestore, Pub/Sub, Cloud Functions (locally)
  - Use Test Lab for Android apps
  - Create ephemeral environments via Infrastructure as Code (IaC) tools like Terraform

## 2.3 Use GCP Tools for CI/CD Pipelines

- CI/CD Key Components
  - CI: Cloud Build, GitHub Actions for automating building, testing, packaging
  - Artifact: Artifact Registry to store Docker images and libraries
  - CD: Cloud Deploy, Spinnaker to release to GKE, Cloud Run, or App Engine
- Sample cloudbuild.yaml (CI Pipeline)
  - Install dependencies with npm
  - Run tests with npm
  - Build Docker image tagged `gcr.io/$PROJECT_ID/my-app`
  - Store image in `gcr.io/$PROJECT_ID/my-app`

## 2.4 Debug Application Issues in Development and Production

- Tools for Debugging
  - Cloud Logging: Check logs for errors and info
  - Cloud Debugger: Inspect variables in live app
  - Cloud Trace: Diagnose latency and bottlenecks
  - Error Reporting: Aggregate and alert on exceptions
- Best Practices
  - Use structured logging (JSON format) for queryable logs
  - Use labels and request tracing IDs for better observability
  - Monitor cold starts, memory, CPU limits for Cloud Run/Functions

## 2.5 Manage Dependencies and Build Tools

- Dependency Management by Language
  - Node.js: npm, yarn
  - Python: pip, requirements.txt
  - Java: Maven, Gradle
  - Go: go.mod, go.sum
- Best Practices
  - Use version pinning (==, ~) to avoid unexpected updates
  - Store libraries and Docker images in Artifact Registry
  - Perform vulnerability scanning on images and dependencies

## Recap Checklist

- Use CLI tools (gcloud, gsutil, bq) effectively in scripts and builds
- Write and execute unit, integration, and E2E tests
- Build CI/CD pipeline using Cloud Build and Cloud Deploy
- Use GCP debugging tools (Cloud Debugger, Logging, Trace) comfortably
- Manage dependencies securely and efficiently across languages

## Recommended Study Resources

- Cloud Build Docs: https://cloud.google.com/build/docs
- Artifact Registry Guide: https://cloud.google.com/artifact-registry
- Google Cloud Debugging Tools: https://cloud.google.com/tools/cloud-debugger
- Testing Best Practices: https://cloud.google.com/architecture/testing-code-in-production

Presented with xmind