Google Cloud

Webinar | AMER & EMEA

Community

TechTalks



October 8, 2026

Shift-Left API Development

Automated Testing & CI/CD with the Apigee Emulator

https://goo.gle/Recordings

Apigee

Housekeeping Notes

- 🎥 This session is being recorded so you can revisit it on demand later.
- 🔇 Please remain muted during the presentation
- 💬 Feel free to share questions anytime through the chat.
- 🎤 We'll open the floor for discussion and live questions **at the end** of the session.
- 👍 Use reactions and emojis to engage throughout the presentation.
- 🤝 Please help us keep the space respectful and welcoming for everyone.

Speaker



Nigel Walters

Google Cloud Platform Sales Engineer

GOOGLE CLOUD APIGEE DEVELOPER ENABLEMENT

Automated Testing of Apigee Proxies & Deployments

Accelerating API lifecycle feedback loops using the Apigee Local Emulator, CI/CD pipelines, and Cloud Run preview sandboxes.

- Local Apigee Runtime
- GitHub Actions CI/CD
- Cloud Run Previews

DEVELOPMENT CHALLENGES & PARADIGM SHIFT

The Shift-Left Testing Imperative

The Cloud Feedback Loop

Deploying proxy revisions to remote Apigee X or hybrid environments takes 30–90 seconds per iteration, creating deployment bottlenecks.

Disrupts shared dev orgs and exhausts quota

Sub-Second Local Testing

The local emulator deploys bundles in under 2 seconds on developer workstations, providing instant assertion feedback on every code change.

Zero Cloud Costs and no network round trips

Automated Quality Gates

Integrate deterministic policy validation directly into CI/CD pipelines to catch syntax, policy, and routing defects prior to merge.

Isolate state and reset cleanly

TESTING METHODOLOGY

The Apigee Testing Pyramid

03 **Live Cloud Runtime QA (Apigee X / Hybrid)**
UAT, end-to-end multi-region latency, mTLS handshakes, real IAM, and backend services.
Production Gate

02 **Apigee Local Emulator (Local / CI / Cloud Run)**
Automated regression suites, policy assertions, flow variables, KVMs, and debug trace checks.
Fast Feedback

01 **Static Unit & Linter Checks**
Syntax validation, schema verification, XML structure, and apigeelint pre-commit hooks.
Code Quality

RUNTIME ARCHITECTURE

Apigee Local Emulator Capabilities

Authentic Message Processor

Executes actual Apigee runtime logic including JavaScript policies, XML transformations, AssignMessage, and Condition evaluations.

Docker Image: google/apigee-emulator

Embedded Cassandra Datastore

Persists key-value maps (KVMs), target server definitions, and developer app credentials locally without external dependencies.

Self-contained local state persistence

Local Management APIs

Exposes REST endpoints on port 8080 to deploy proxy bundles, inspect revisions, query configuration, and export transaction traces.

Deploy in < 2 seconds via REST

Safe Sandboxing & Reset

Trigger a clean state wipe via REST between test runs, guaranteeing zero test cross-contamination and reproducible CI builds.

Endpoint: POST /v1/emulator/reset

CONFIGURATION & DECLARATIVE TESTING

Proxy Structure & Deployment Manifests

DEPLOYMENT MANIFEST (YAML)

deployment-1.yaml

Specifies the exact runtime binding, environment parameters, and target mocks for automated execution.

```yaml
environments:
  - name: test-env
    proxies:
      - name: test-proxy
        basepath: /v1/test
        bundle: ./src/test-proxy.zip
    targetservers:
      - name: mock-backend
        host: mocktarget.apigee.net
```

Declaratively mirrors production environment topology

POLICY & LOGIC VERIFICATION

JS-AddHelloWorld

Validates that custom JavaScript, flow conditions, and policy transformations mutate headers and bodies correctly.

```javascript
// JS-AddHelloWorld.js
var response = context.getVariable("response.content");
var payload = JSON.parse(response || "{}");
payload.status = "success";
payload.validatedBy = "ApigeeEmulator";
context.setVariable("response.content",
  JSON.stringify(payload));
```

Enables assertions on exact policy output and variable values

DEVELOPER EXECUTION

Step-by-Step Local Testing Workflow

1

Start Emulator

Launch containerized Apigee runtime with port mappings for proxy execution.

```
docker run -d -p 8080:8080 …
```

2

Deploy Assets

Run deploy.sh to package proxy bundles and apply manifest.

```
./deploy.sh deployment-1.yaml
```

3

Tester Service

Launch apigee-emulator-service to host UI and assertion engine.

```
./local.sh start
```

4

Run Assertions

Trigger automated assertions via REST to evaluate status codes and traces.

```
POST /api/tests/run
```

PIPELINE AUTOMATION

Continuous Integration with GitHub Actions

AUTOMATED PULL REQUEST VALIDATION

Every pull request runs automated emulator assertions inside GitHub Actions runners before code review.

- Spins up Docker container in standard GitHub runner
- No cloud credentials or IAM permissions required
- Executes regression assertions in under 20 seconds

Catches breaking policy changes before cloud deployment

CI PIPELINE WORKFLOW DEFINITION

.github/workflows/test.yml

```
jobs:
  test-proxies:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Start Apigee Emulator
        run: docker run -d -p 8080:8080 google/apigee-emulator
      - name: Deploy & Run Test Suite
        run: |
          ./deploy.sh deployment-1.yaml
          curl -f -X POST http://localhost:8080/v1/tests/run
```

Deterministic exits fail the CI pipeline upon assertion breach

COLLABORATIVE SANDBOXES

Cloud Run Ephemeral Preview Environments

Deploy Service to Cloud Run

Package apigee-emulator-service as a container and deploy to Cloud Run for automated scale-to-zero test hosting.

../cloudrun.sh deploy

Shared Preview Sandboxes

Provide frontend teams and QA engineers with live HTTPS endpoints to test proxies against branch revisions before merging.

Instant branch URLs for stakeholders

Interactive Trace Recording

Inspect execution flow, policy variable mutations, and request/response transformations directly via the web tester UI.

Trace JSON download & visual inspection

BEST PRACTICES & GOVERNANCE

Enterprise QA Strategy

Rapid Developer Feedback

Shift 80% of proxy validation from remote cloud deployments to local emulator assertions to accelerate development cycles.

Automate CI Quality Gates

Enforce emulator assertion suites in GitHub Actions or Cloud Build to prevent broken proxy logic from entering main branches.

Shareable Preview Sandboxes

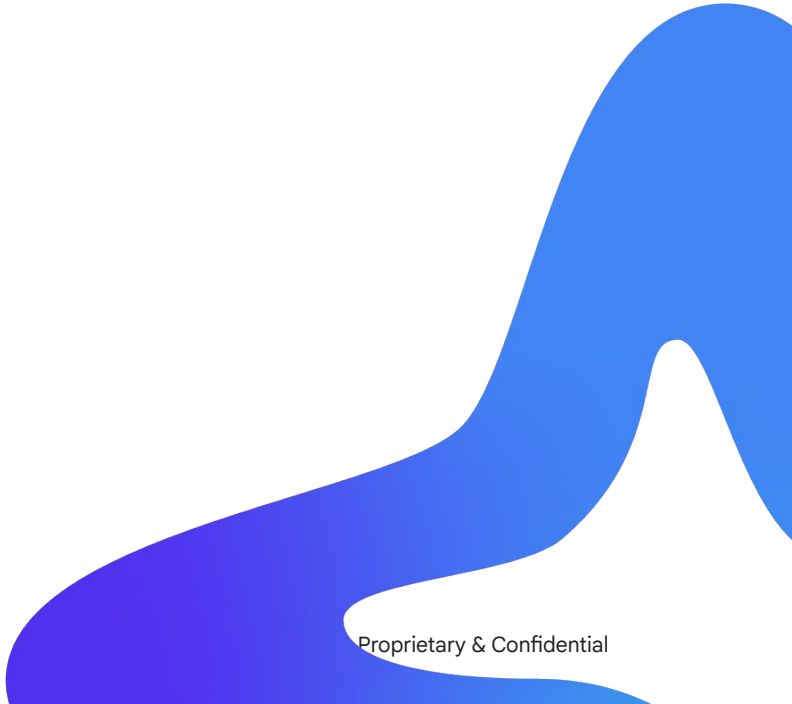
Leverage Cloud Run preview environments for cross-functional validation with frontend developers and QA partners.

Balanced Verification

Reserve live Apigee X/Hybrid environments for true end-to-end integration, performance benchmarking, and security sign-off.

DEMO

Using the Apigee Emulator locally and via Cloud Run



Proprietary & Confidential

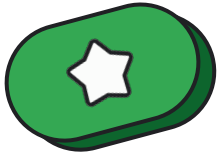
Q&A

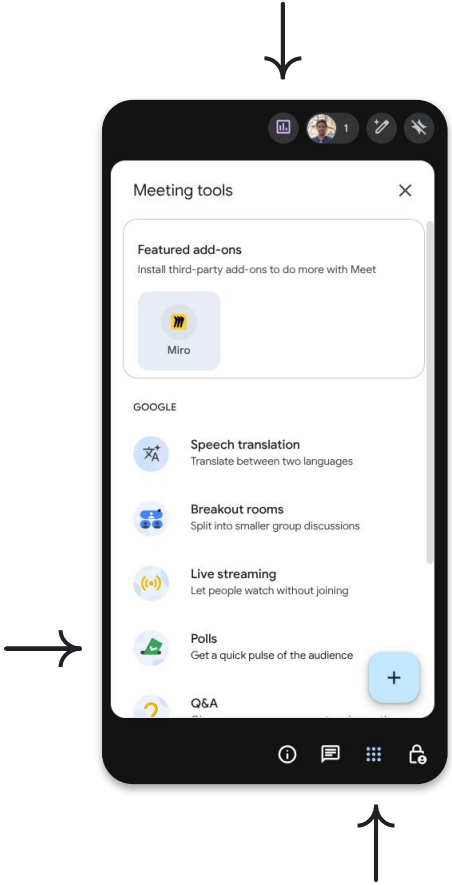


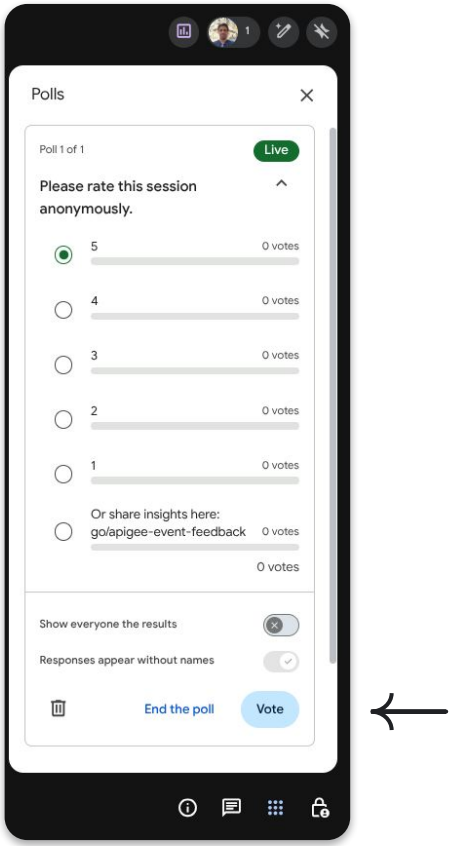
Please share your feedback!

Help us make our sessions even better!

Please provide your **anonymous feedback** by rating this session.







Catch up!

New Community Content

- → Securing WebSockets for Agentic & Real-Time Architectures ↗
- → Apigee Trace Viewer for Fast & Easy Trace Parsing ↗
- → Automated Testing of Apigee Proxies and Deployments with the Apigee Emulator ↗
- → Dynamic Model Routing with Apigee X ↗
- → Apigee Hybrid 1.17.0 introduces Model Context Protocol (MCP) support

Upcoming Events

- → Oct 22 | What's new and next in Apigee
- → Oct 29 | From prompt to production: End-to-end AI workflows in Apigee

Google Cloud

Community



Thank You