Canary Deployment Strategy: Feature Flag + Dial-Up Routing

# 📌 Problem Statement

Partners requested the ability to test new versions of APIs with zero impact on existing traffic and have a safe rollback path if issues are found — all without affecting other consumers.

# ⚠️ Current Limitation

Our PCF-based blue-green strategy is effective for full cutovers, but doesn't support partner-controlled rollout/versioning. Also, internal consumers directly access PCF endpoints, limiting flexibility.

# ✅ Proposed Solution

Use a combination of:  
- Feature Flag Header (from partner) → Controls version toggle.  
- Dial-Up Flag (Apigee Developer App attribute) → Governs whether partner is eligible for canary access.  
  
Together they route traffic to:  
- Main App (v1)  
- Canary App (v2, v3, …)

# 🎯 Logic Matrix

|  |  |  |
| --- | --- | --- |
| Dial-Up Flag | Feature Flag | Routed To |
| false | any | Main App |
| true | false | Main App |
| true | true (new) | Canary App |

# 🔧 Client-Side Changes

- Send `featureFlag` header in requests (value = `new`).  
- Nothing changes for default (v1) access.

# 🏗️ Synchrony-Side Changes

- Add `dialup=true/false` to Apigee Developer App.  
- Deploy separate PCF app for each version.  
- Modify routing logic in Apigee based on the flag combo.

# 📦 Long-Term Vision

- Reusable architecture for future v3, v4 rollouts.  
- Optional: Routing microservice for internal consumers.  
- Minimal change for partners and downstream apps.  
- Potential support for multi-product traffic control.

# 🏢 Industry-Backed Model

Companies like Netflix, Google, Amazon, and Microsoft already use canary deployments to enable progressive rollouts and partner-controlled testing.