CSC+Google Tech Talk
9/17/2014

Sign-in: **http://goo.gl/UKnvyX**

Building a mobile platform

Wesley Tarle (wt@google.com)

About Wes

- UWaterloo '08
- Joined Google 3 years ago
- Google+ platform TL
  - Sign-in with Google
  - Google Play services
  - G+ and Games widgets and APIs
- Google Mobile Platform

A story

Why a mobile platform?

- Increased scope of impact (scale across many developers)
- Improved security and privacy for users
- Spam and abuse considerations
- App quality improves

Challenges

- Quality of static portions
- Binary compatibility of dynamic parts
- Performance considerations (native types)
- Keeping track of clients (who's calling what)
- Difficult to test

Mobile platform parts

- Android
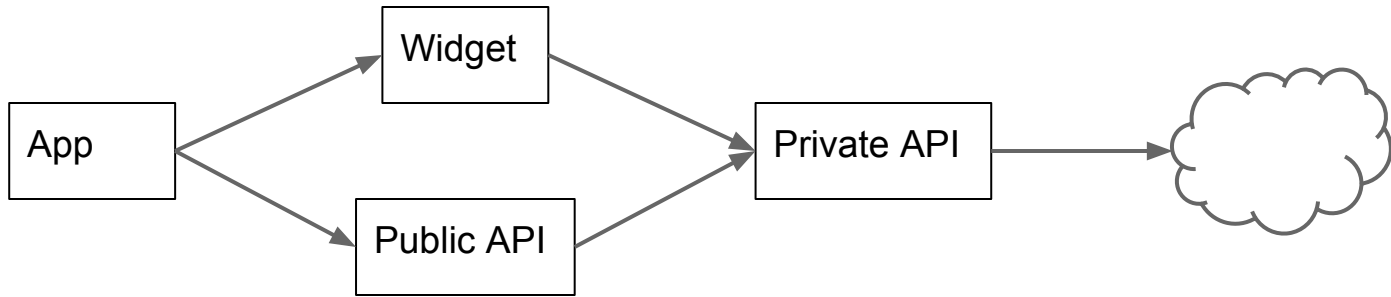- iOS
- Mobile web
- REST/RPC

API or Widget

API
- Quotas, spam & abuse checks
- Allows custom UX
- Restrictive access to data

Widget
- Native experience
- Controlled UX
- Complete access to data

App → Widget → Private API → (cloud)
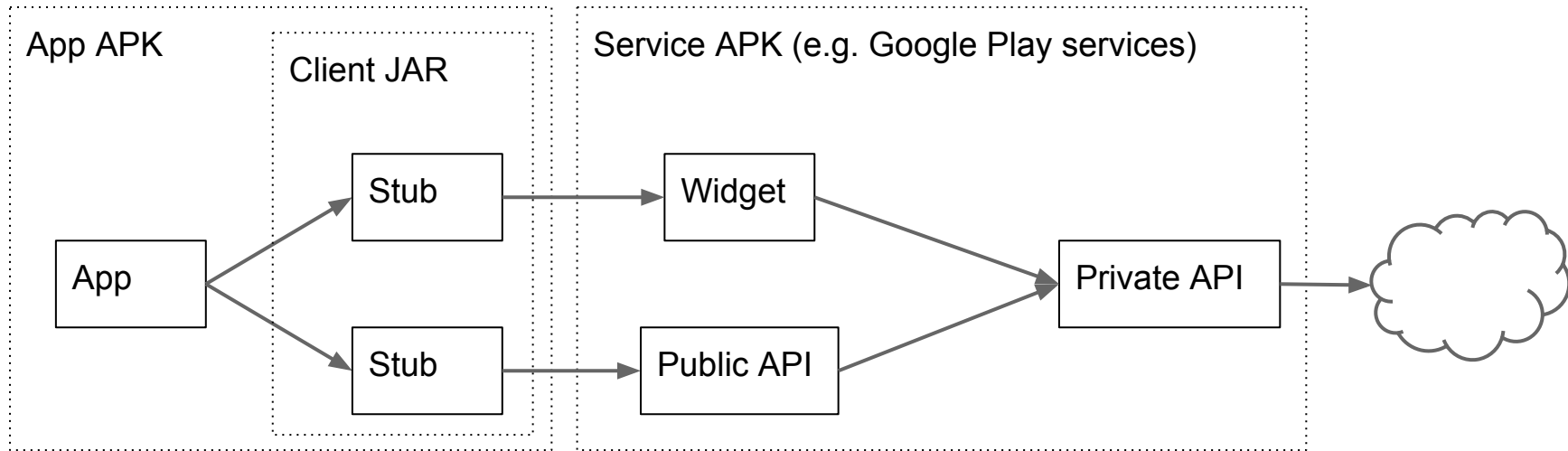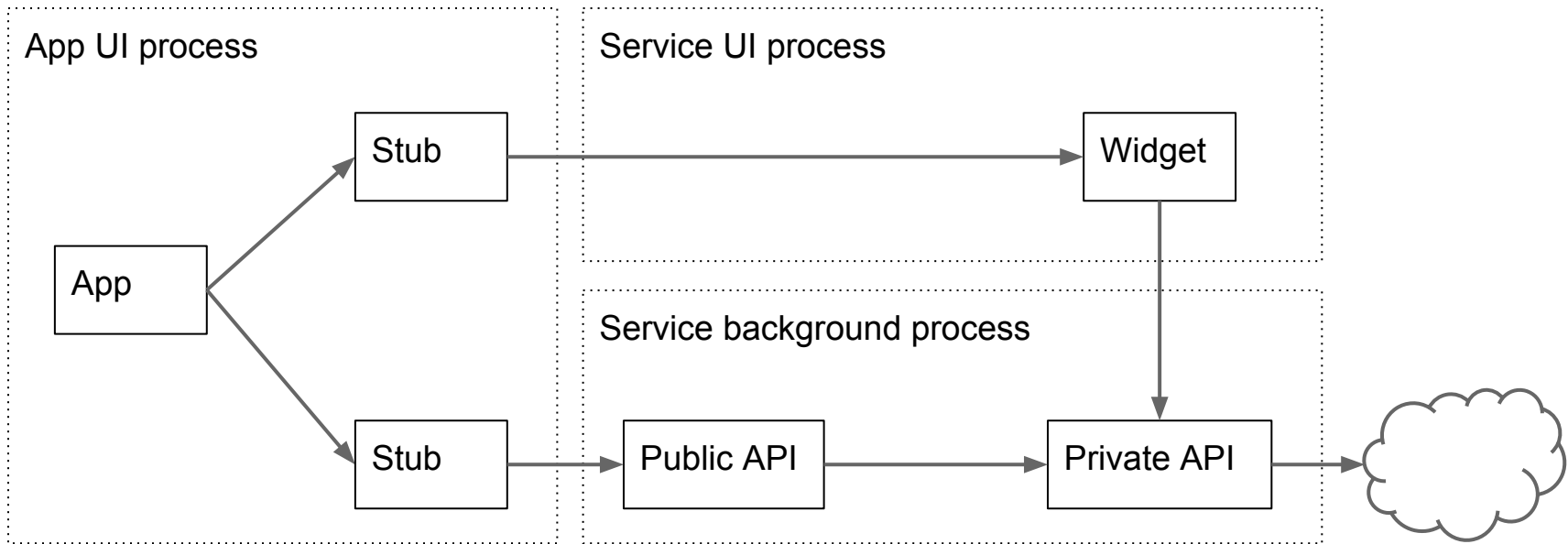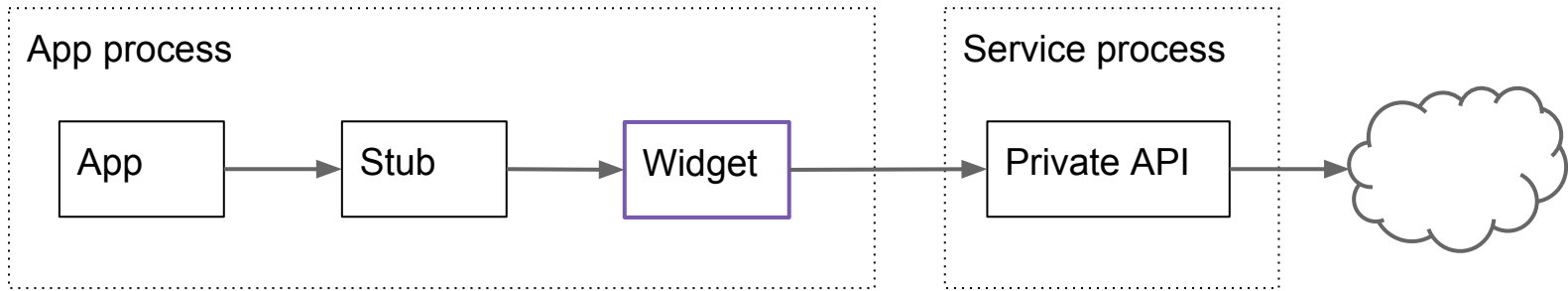App → Public API → Private API

API and Widget

Android

- Binder provides IPC mechanisms
  - Minimal code in client .jar
  - Binary compatibility requirement
  - Multi-process architectures
- Intents and activities
  - Allow for privileged implementations of widgets
  - Serve from master APK

App APK

Client JAR

Service APK (e.g. Google Play services)

App

Stub

Stub

Widget

Public API

Private API

Android physical model

App UI process

Service UI process

Service background process

App

Stub

Stub

Widget

Public API

Private API

Android process model

App process
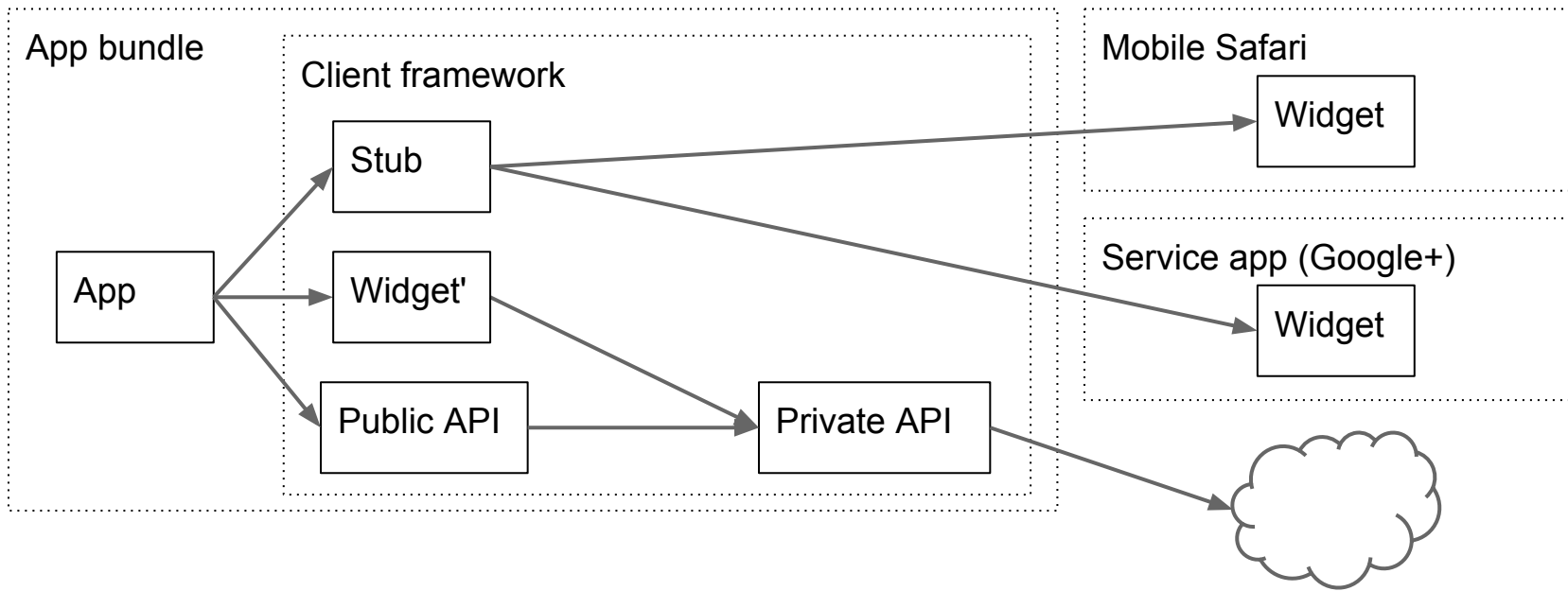
App → Stub → Widget

Service process

Private API

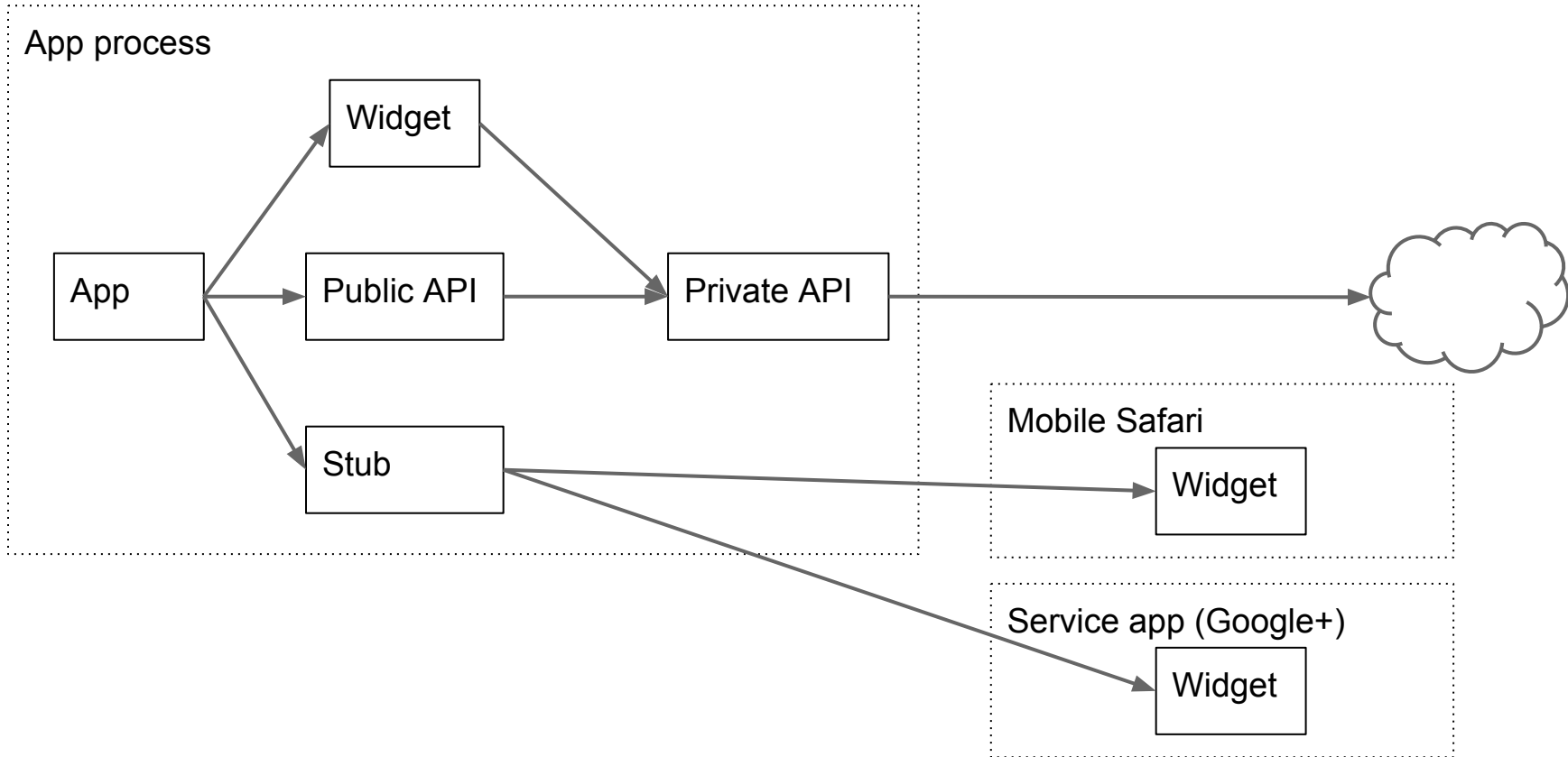Android in-process widget

Another story

iOS

- No general IPC mechanism
  - Single-process execution or traumatic UX
  - Libraries distrubuted statically (no binary compatibility requirement)
- Basic support for intenting
  - Defined "schemes" for going from app-to-app
  - No background services from third-party developers

iOS physical model

App process

- App
- Widget
- Public API
- Private API
- Stub

Mobile Safari

- Widget

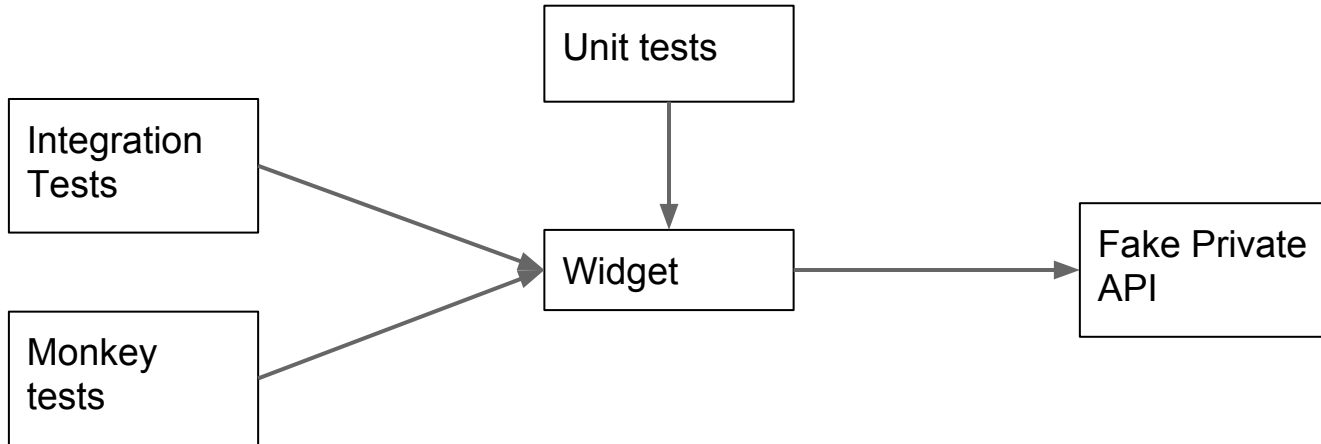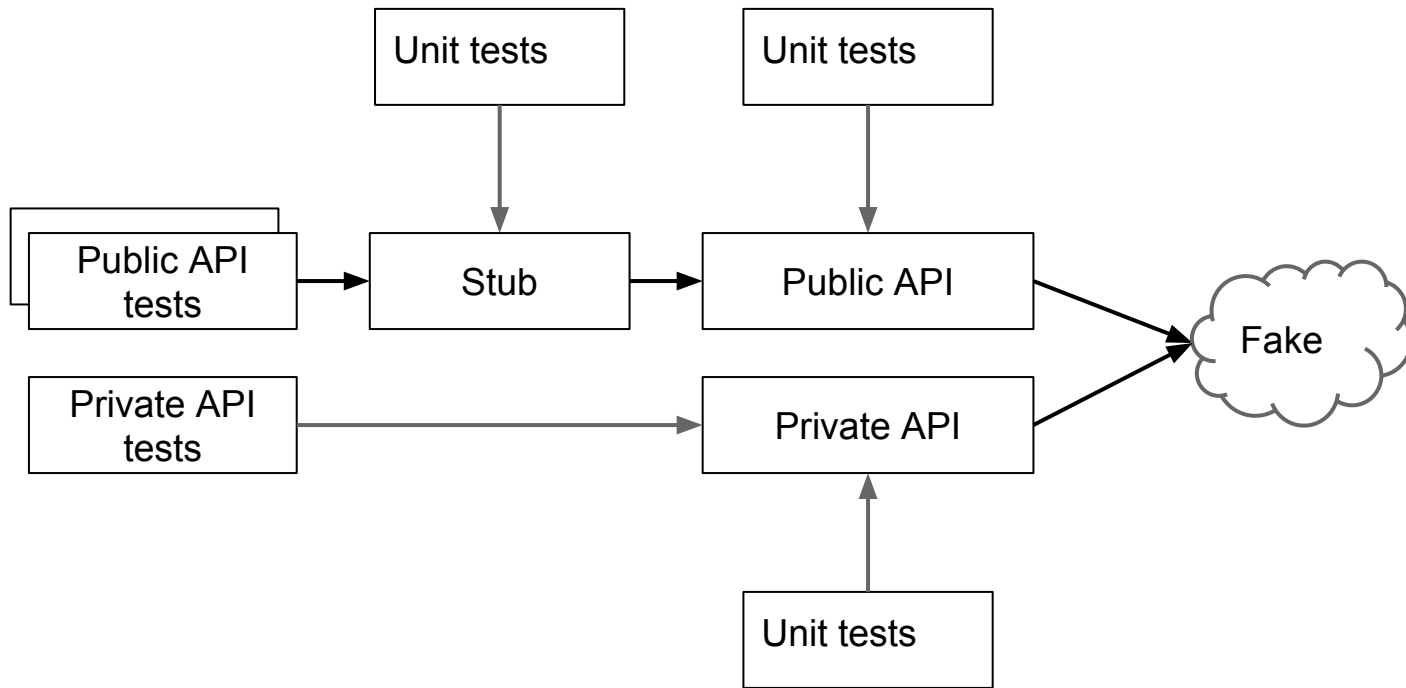**Service app (Google+)**
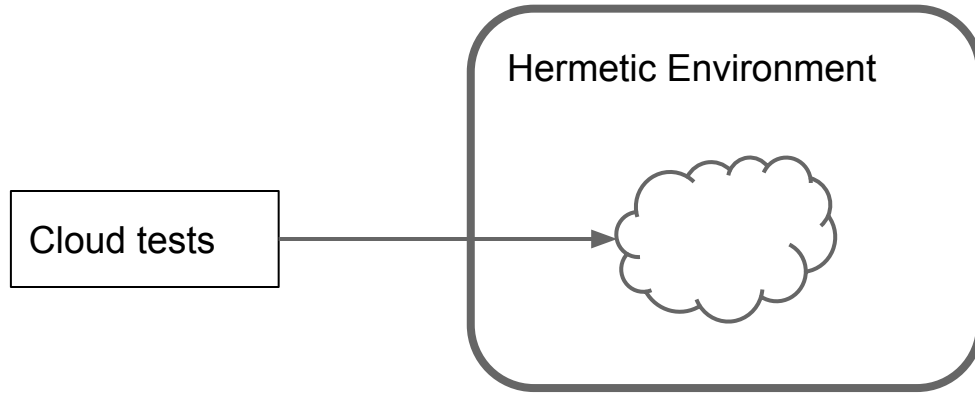
- Widget

iOS process model

Testing

- Challenges: multiple processes, multiple platforms (device + cloud)
- Approach: decouple components by faking parts
- Definitions: unit test, API test, integration test, monkey test. 80/15/5
- Manual test harness

Unit tests

Integration Tests

Monkey tests

Widget

Fake Private API

Testing Widgets

Testing APIs (Backwards Compatibility)

Testing Cloud

Thank You! Questions?

uwaterloostudents@google.com