# How to Build a Self-Hosted Device Farm Using Appium

From zero mobile tests to 1,000+ daily tests on 16 real devices

WeTest | 2026

# About Us

## Anna Kalypso Podimata

QA Ops Engineer

Electrical and Computer Engineering graduate. Worked as IT operations, network and cybersecurity engineer in order to finally find my passion in QA Ops engineering.

*I built the farm. She used it.*

## Georgianna Makraki

QA Automation Engineer

Applied informatics graduate. Extensive experience in various aspects of Software testing. Currently working on automating native app tests using Appium.

*She built the farm. I cherish it.*

**"We went from 0 automated mobile tests to 1,000+ daily runs — on hardware we own."**

# Agenda

1. Why self-hosted?
2. The evolution — how we got here (and what failed)
3. The architecture today
4. CI/CD integration — Nightly Pipeline Flow
5. Demo
6. Challenges & hard-won solutions

# Why Self-Hosted?

## 9 Domain Apps

- Domain-specific applications
- Tangled business logic

## High Volume of Tests

- Sanity: ~400 test cases
- Regression: 1000+ test cases

## 16 Parallel Devices

- Mobile app tests are slow
- Need for parallelization

## Cut Time Everywhere

- Pre-install APKs
- Disable feature hints via ADB
- Quick logins via shared_prefs

# Why Self-Hosted?

| | | Cloud Provider | Self-Hosted |
|---|---|---|---|
| Getting Started | | ✓ Instant setup | ✗ Setup required |
| Cost | ⚠ | ≈ Zero upfront / Grows with scale | ≈ HW investment / Fixed at scale |
| Device Selection | | ≈ Large pool | ≈ Focused set |
| Device Control | ⚠ | ✗ Limited access | ✓ Full ownership |
| Scale Up | | ✗ Linear cost ↑ | ✓ Plug in more |

# Evolution

**Stage 0**

**Manual Only**

Zero automated tests

Everything by hand

**Stage 1**

**2 Emulators on VM**

First automation attempt

Emulators on a VM

✗ Don't replicate real UX

✗ VM too weak / No pipeline

**Stage 2**

**5 Devices + Mac Mini**

Real devices, proof of concept

2 Appium container per device

✗ Mac underpowered

✗ Too many containers

**Stage 3**

**10 Devices + Grid**

Ubuntu server

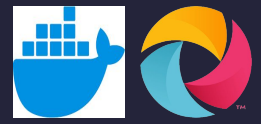
2 containers per device

**SeleniumConf 2025:**

**Discovered device-farm plugin**

**Today**

**16 Devices + 1 Container**



device-farm plugin

Full nightly pipeline

# The Architecture Today

## R2D2 Server (Ubuntu Host)

*DOCKER CONTAINERS*

**noviappium-hub**
port 4723 — device-farm plugin

**mitmproxy-hub**
proxy & webui & capture ports

*SYSTEMD SERVICES*

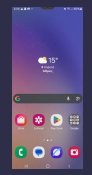
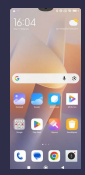
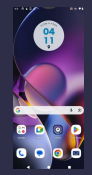
**device-manager API**
port 8001 — FastAPI + SQLite

**adb-server**
port 5037 — manages 16 USB devices

**USB Hub**

**16 PHYSICAL ANDROID DEVICES**

| 7x Samsung | 5x Xiaomi | 4x Motorola |
|---|---|---|
| Galaxy A54 | Redmi | Moto G54 |

## Key Wins

**1 container**
Replaces 32 containers + Selenium Grid

**16 sessions**
Concurrent test executions

**USB only**
WiFi was a lesson we learned the hard way

**Custom API**
Device health, APK install, proxy mapping

# Custom Services

## Device manager API

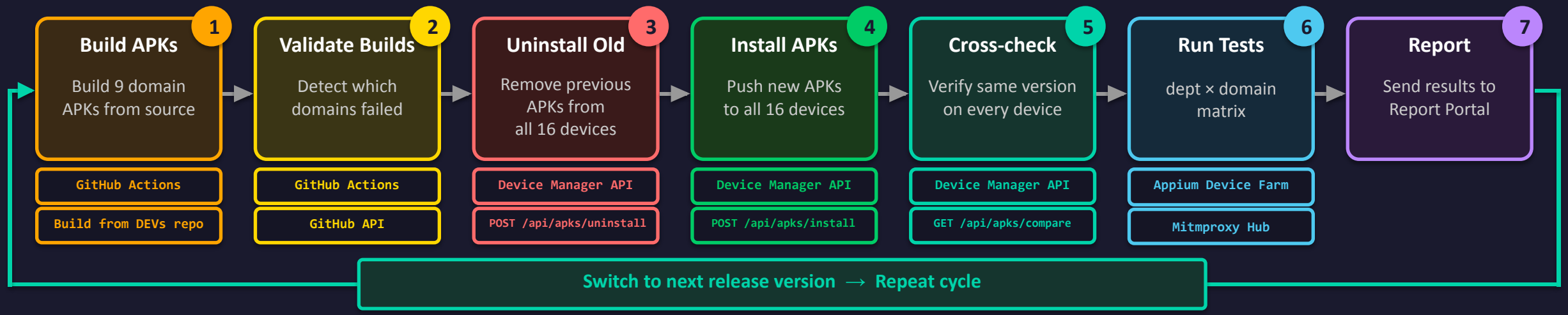
A lightweight FastAPI service that turns manual device management into API calls.

| Endpoint | What it does |
|---|---|
| `POST /api/apks/install-all` | Install APKs on ALL devices |
| `POST /api/apks/uninstall-all` | Uninstall APKs on ALL devices |
| `GET /api/devices/{device_serial}/packages` | Get installed APKs |
| `GET /api/apks/compare/staging` | Verify APK versions match across devices |

**The key insight: ADB + a small custom API turns device preparation from a manual chore into a single API call in your CI pipeline.**

## Monitor network traffic (Proxy)

# Nightly Pipeline Flow

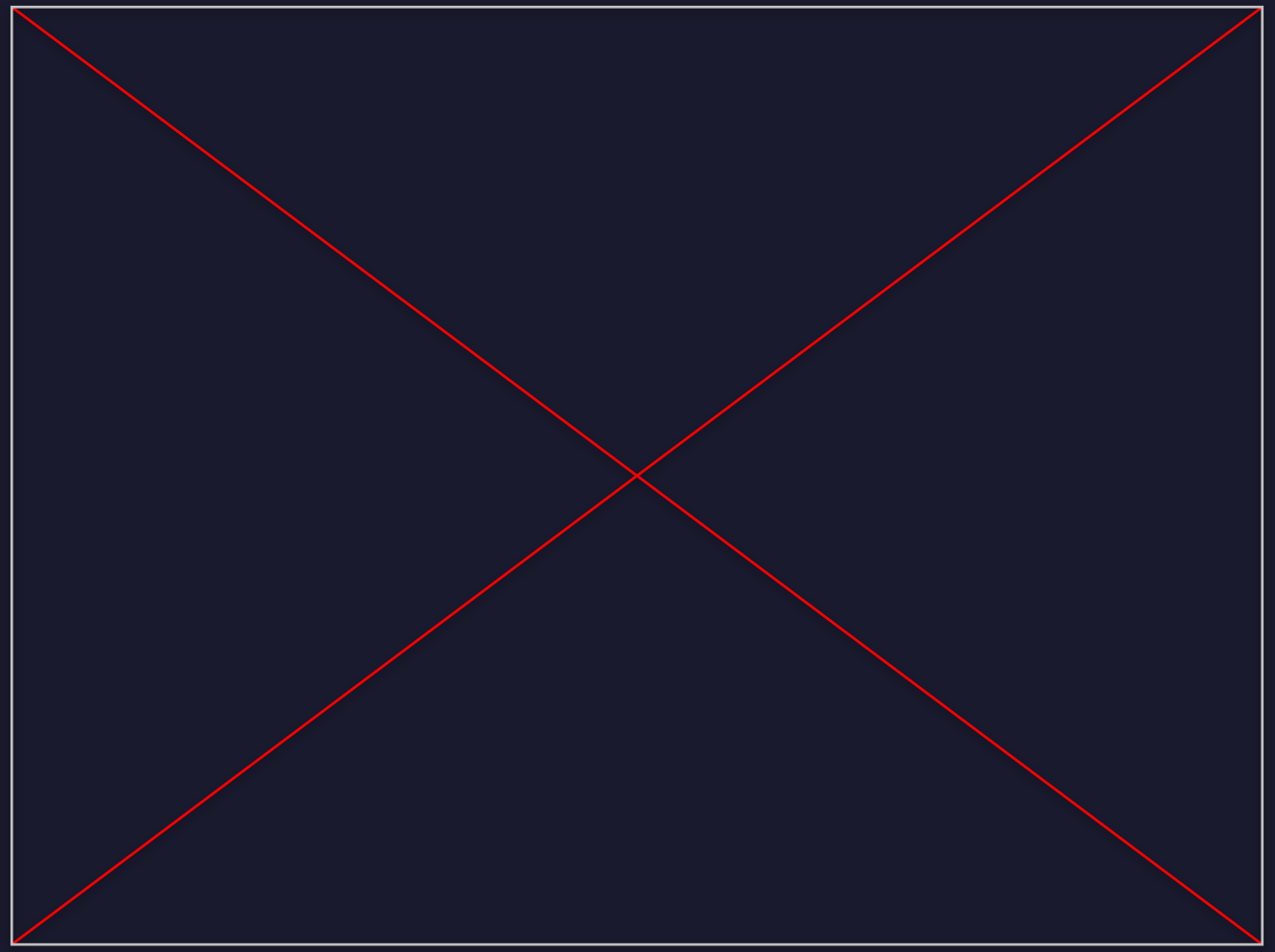



## What changes between iterations:

| Iteration | Release Version | Test Scope | Purpose |
|---|---|---|---|
| 1st cycle | Previous Release | Sanity | Quick check on previous release |
| 2nd cycle | Current Release | Regression | Full regression on latest release |

If a domain's APK fails to build, it is automatically excluded from that cycle's test matrix — no manual intervention

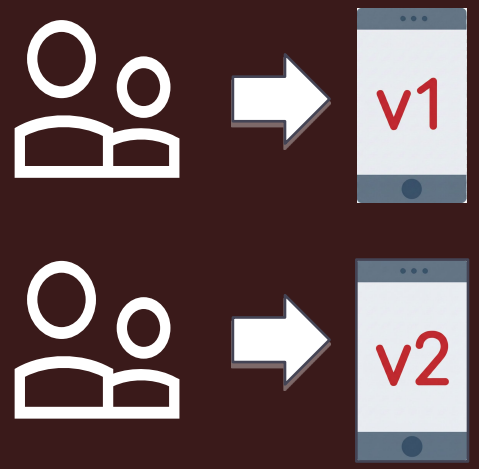
# Demo Time



# Challenges: Backwards Compatibility

## The Problem

**App in the store is NOT force-updated**




Different users in production can run different versions

## The Solution

**Maintain in our tests 2 versions (current + 1 prior) with some adjustments.**

## The Result

**1,000+**

test executions daily across 2 app versions on 16 real devices

# Challenges: Making Tests Fast

### 📦 Install APKs Once

Pre-install before nightly run, not per-test

**~30s saved per test**

### ✨ Disable All Animations

```
animator_duration_scale 0
window_animation_scale 0
transition_animation_scale 0
```

**No more 20–30s waits**

### 💬 Dismiss Feature Hints

ADB commands to clear first-run dialogs and onboarding screens

**Skip one-time popups**

### 🔑 Grant Permissions

```
adb shell pm grant ... before run
```

**No dialog interruptions**

### 🔌 USB, Not WiFi

Learned the hard way — wireless ADB adds seconds to every command

**Night & day difference**

### ⚡ Parallel via Matrix

GitHub Actions matrix runs across all 16 devices simultaneously

**16x throughput**

# Challenges: Hybrid Apps & WebViews

## ✗ The Problem on Physical Devices

**Switching to WebView context hung indefinitely or was flaky**

Chromedriver starts, attaches the DevTools socket —
but the WebView isn't stable enough to accept the session

```
driver.switchContext("WEBVIEW_gr.novibet.sports.betting.uat");
// ⏳ Chromedriver starts... ready... POST /session...
// ⚠️  timeout of 240000ms exceeded
// ⚠️  Chrome DevTools Protocol never connected
```

**Root Causes**

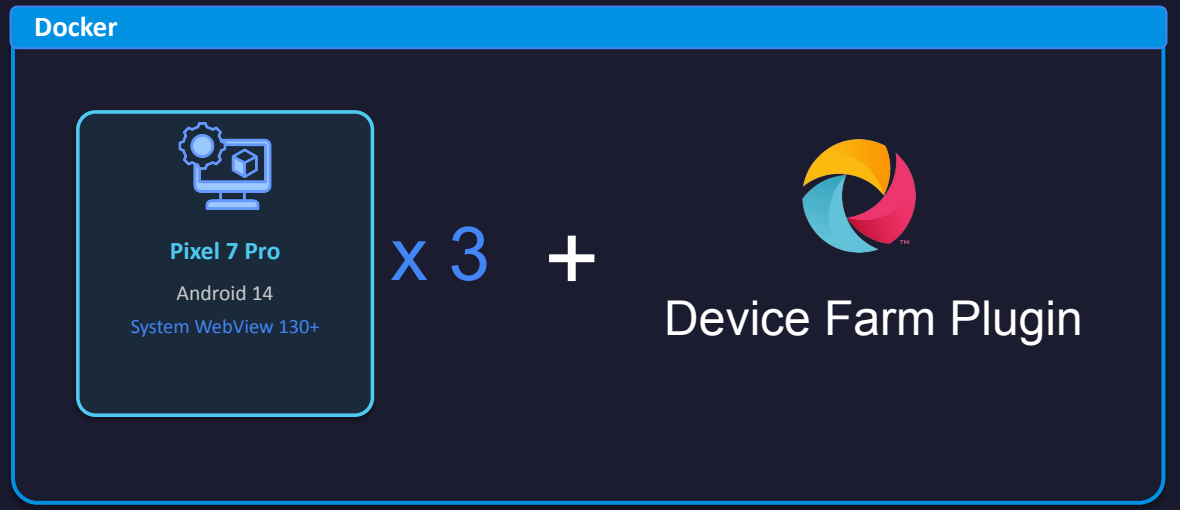
Difficult to guarantee WebView is stable at test runtime

16 devices × 3 manufacturers = impossible to control webview updates

## ✓ The Solution: Dedicated Emulator Farm

**Emulators give a consistent, controllable environment**

Pinned WebView version, WebView was very stable when the test switches context.




Acceptable ?

✓ ~8% of tests need switch to Webview

✓ Mostly tests that involve external integration

# Key Takeaways

1. **Mobile test automation is not magic**
   It’s ADB, Docker, and Appium — stitched together with a few custom scripts.

2. **ADB is your best friend**
   Pre-install APKs, disable animations, grant permissions, control device state programmatically.

3. **Start small, iterate**
   We went from 2 emulators on a VM to 16 real devices. Each stage taught us what to fix next.

4. **Self-hosted is viable at scale**
   1,000+ daily executions on hardware you control, with full CI/CD integration.

5. **Physical + Emulators = complementary**
   Don’t pick one. Use both for what they’re good at.

# Questions? • Thank you!

Technical Documentation

## Questions? • Thank you!

### Anna Kalypso Podimata

QA Ops Engineer

Electrical and Computer Engineering graduate. Worked as IT operations, network and cybersecurity engineer in order to finally find my passion in QA Ops engineering.

*I built the farm. She used it.*

### Georgianna Makraki

QA Automation Engineer

Applied informatics graduate. Extensive experience in various aspects of Software testing. Currently working on automating native app tests using Appium.

*She built the farm. I cherish it.*