



testDevices.scaleUp();

Thoughts about mobile testing on the cloud

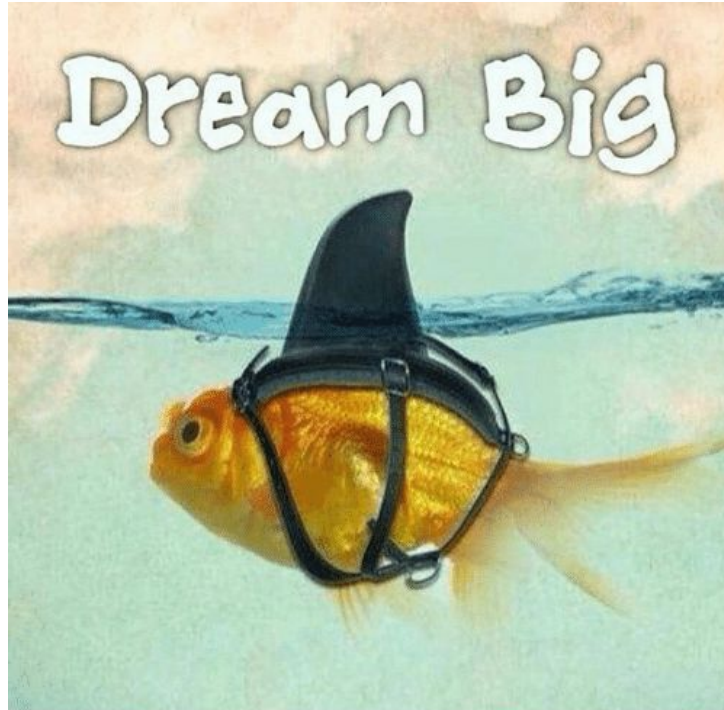
Where are we?



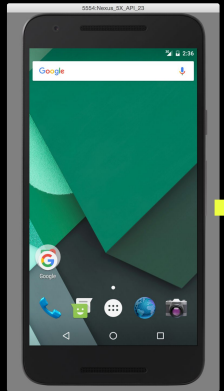
- Test automation PoC ✓
- Some test cases ✓
- Techstack stable and framework ready (“version 1”) ✓
- Initial thoughts about CI ✓

- **Now what next?**

Next question: How can we scale this?



Next question: How can we scale this?



SAUCELABS



pCloudy.com

On premise vs. cloud



- Full control
 - Number of devices
 - Types of devices
 - Software libraries
 - ...

- No maintenance required
- Fixed costs
- Support



- Maintenance
 - replace devices
 - fix issues
 - infrastructure
 - ...



- Less flexibility
 - Available devices
 - Software update cycle? (Appium, Java)

Two approaches towards cloud testing



#1: Client-side execution

Step 1: Upload your APP to the cloud

Step 2: Create a WebDriver instance

```
capabilities.setCapability("app", appUrl);  
new AppiumDriver("http://cloud.com/wd/hub", capabilities);
```

Step 3: Execute your tests as before

For example, Browserstack, Sauce Labs, pCloudy...



Two approaches towards cloud testing



#2: Server-side execution

Step 1: Upload your APP to the cloud

Step 2: Package and upload your tests to the cloud

Step 3: Tests get executed on the cloud

For example, AWS Devicefarm (public cloud)



Client- vs. server-side execution



- Easier migration
 - Execution flow remains the same
 - Reporting and CI remain the same
- More control
 - Software updates on the client side
 - easier to use a mixed/hybrid mode (=use different cloud providers or local + cloud)



- Network latency

- Performance (maybe)

- Trust and compliance
- Integration overhead
 - Test results and reporting
 - Network challenges













What's your use-case?

- **Devices**
 - specific models, manufacturers, OS versions
- **Parallelisation**
 - How to distribute n tests across m devices efficiently (each test runs once)?
 - How to run n tests on m devices in parallel (each test runs m times)?
- **Time constraints, execution speed, waiting times**
 - run tests once a day (“nightly tests”) → not time-critical
 - run tests multiple times a day to verify changes → highly time-critical
- **Client- vs. server execution mode**
- **API support**
- **Cost and pricing models**
- **Future use-cases**

Device models



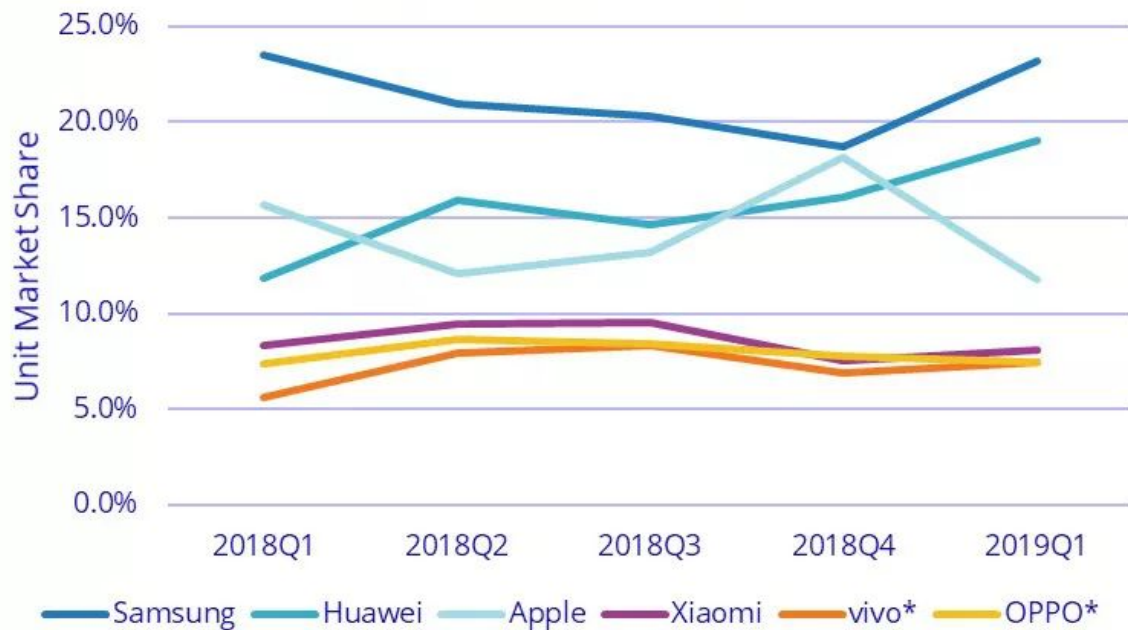
Device Vendor	Rank	Traffic %
Apple	1	45.62 
Samsung	2	17.19 
HTC	3	9.55 
Asus	4	5.36 
LG	5	5.36 
Sony	6	4.46 
Oppo	7	4.37 
Xiaomi	8	3.17 
Huawei	9	0.89 
Vivo	10	0.76 

- In Taiwan Apple and Samsung cover 63% of the market (US 81%, UK 87%, Singapore 82%, Australia 89% etc.)
- What about Huawei, Xiaomi, Oppo, Vivo, One Plus, Lava, Karbonn, Micromax, Symphony, Walton...?

Device models



Worldwide Top 5 Smartphone Companies,
2019Q1 Unit Market Share



- Huawei (13,3%) surpassed Apple (11,9%) in market share in Q2/2018
- Q1/2019: 19% global market share



Market specifics

- India: 20% Xiaomi (#3) = 2/3 are neither Apple nor Samsung
- Myanmar (2018): 19% Oppo (#2), 18% Vivo (#3), 13% Xiaomi (#4), 11% Huawei (#5) = 70% not Apple and Samsung
- Pakistan: 11% QMobile (#2)
- Bangladesh: 14% Symphony (#2)

Sources:

<https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter>, <https://deviceatlas.com/>

Device models



- AWS Device Farm doesn't have any Huawei, Xiaomi, Oppo or Vivo models (in their public cloud)
- Browserstack only supports Samsung, Google and Apple phones (and one OnePlus model)

Sources:

<https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter>, <https://deviceatlas.com>, https://www.browserstack.com/list-of-browsers-and-platforms/app_automate, <http://awsdevicefarm.info>

Demos



We will run the same login test on AWS Devicefarm, Browserstack and locally.

AWS Devicefarm

- package the test configuration into a test spec and upload to AWS
- upload the APK file to AWS
- package our tests (+framework) and upload to AWS
- select an available device (using device filters)
- execute the tests and process the results

Browserstack

- upload the APK file to Browserstack
- configure the execution (this includes the device selection) using `Capabilities`
- execute the tests and process the results



www.justtestlah.qa

<https://medium.com/@mart.schneider/mobile-test-automation-using-aws-device-farm-6bcf825fa27d>