EMULATORS IN ACTION
HOW TO RUN UI TESTS ON YOUR CI

ABOUT ME

ABOUT ME

- Daniel Hartwich
- Android Developer at XING
- Twitter: KiLLyA_
- GitHub: dhartwich1991

LOOKING FOR NEW COLLEAGUES

- 1 (Senior) Android Developer - Platform Team
- 1 Automation Android Developer - Mobile Releases Team

WHAT IS IN THIS?

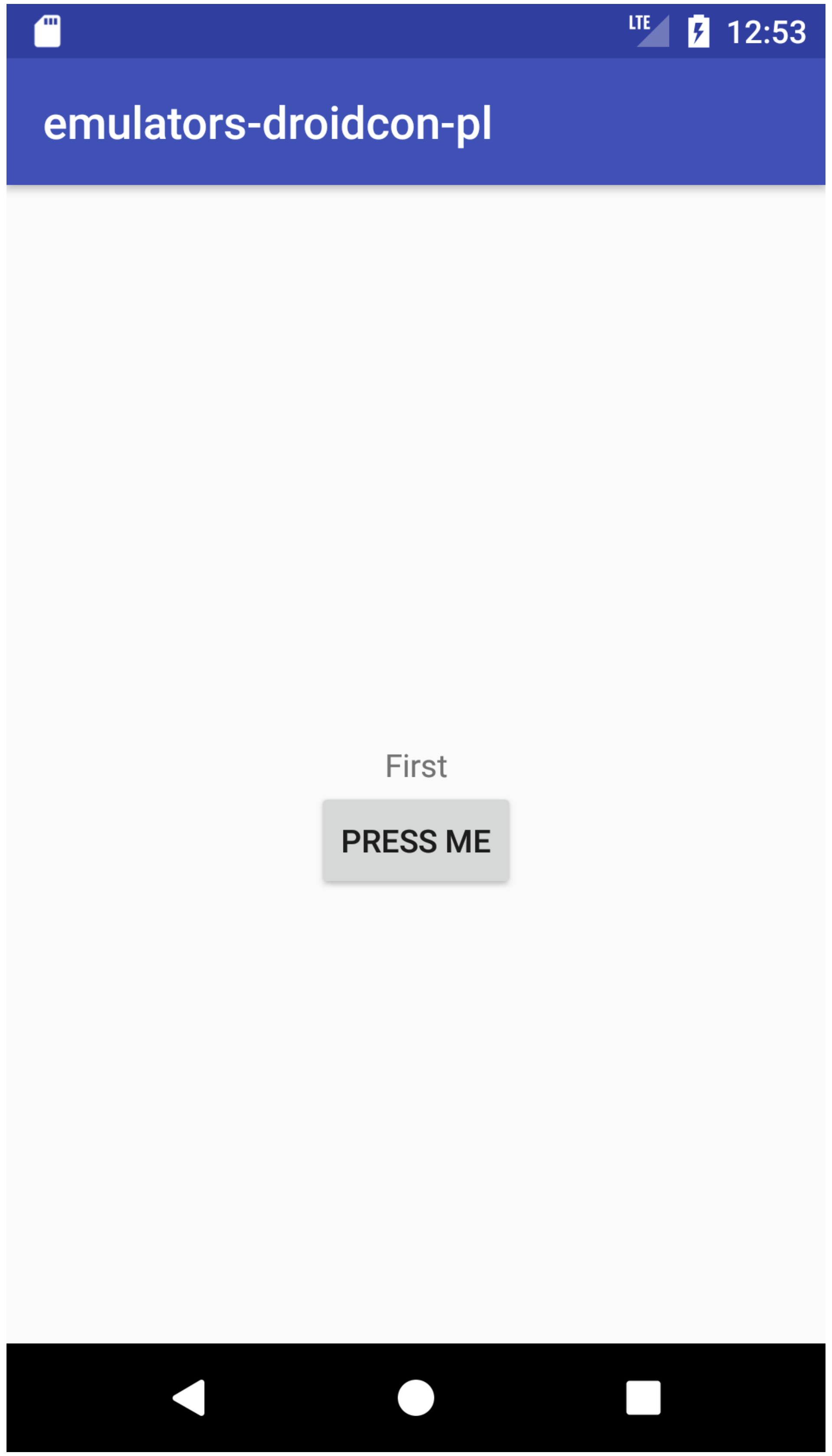
- UI Tests in Android
- Spoon 🥄
- Working setup at XING
- UI Tests on CI (Jenkins)
- Fastlane
- The solution (TM)

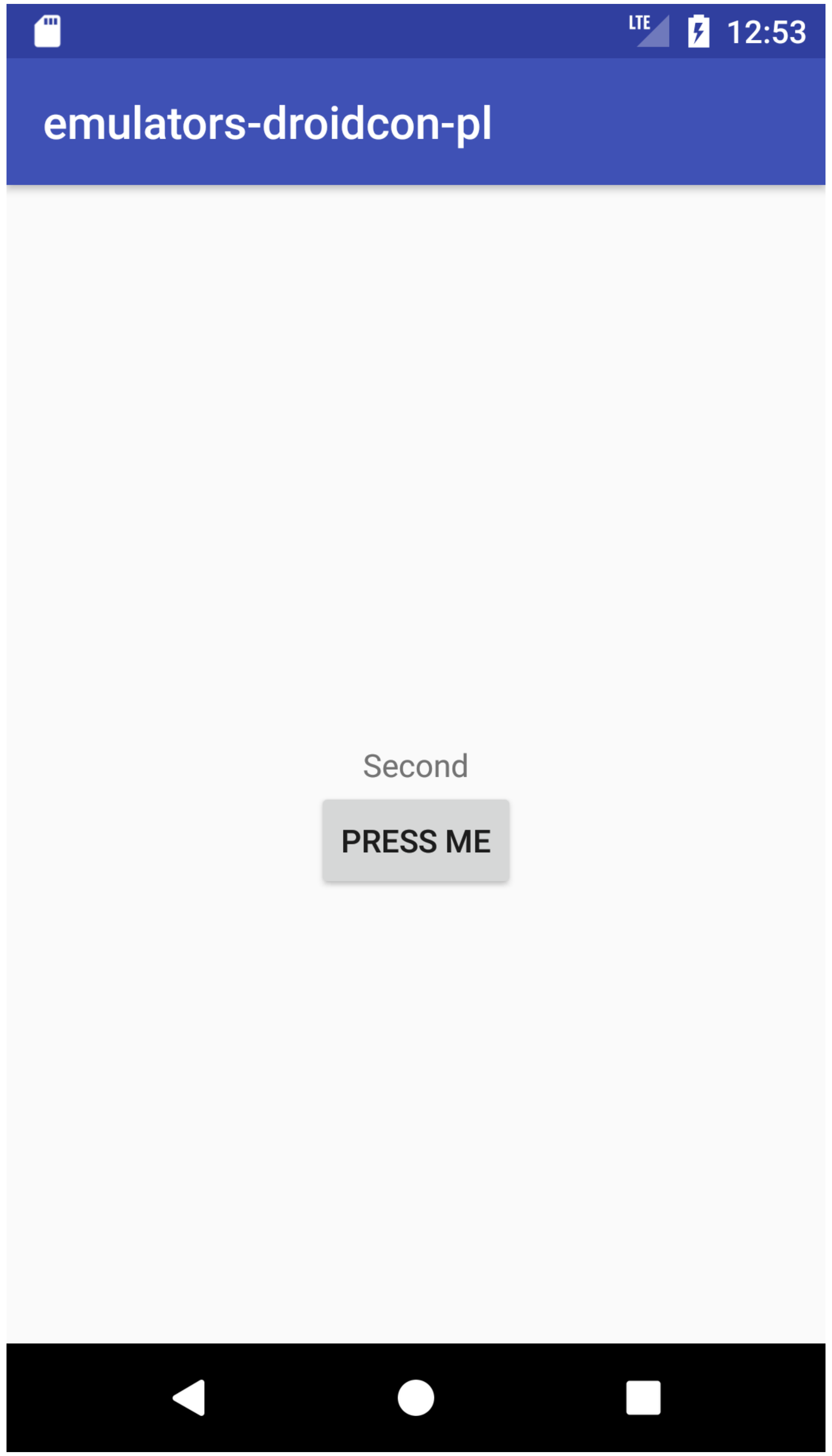
UI TESTS IN ANDROID

- UI Tests
- Espresso

```
@Test fun changesTextWhenClickingButton() {
        onView(withId(R.id.press_me_button))
                .perform(click())
        onView(withId(R.id.change_text_text)).check(matches(withText("First")))
        onView(withId(R.id.press_me_button))
                .perform(click())
        onView(withId(R.id.change_text_text)).check(matches(withText("Second")))
    }
```

- Simple view assertion
- When I click one button should show different text


12:53
emulators-droidcon-pl
First
PRESS ME


12:53
emulators-droidcon-pl
Second
PRESS ME

✅

UI TESTS

- (usually) quick to execute
- test UI of specific screen
- Run them with `./gradlew cAT`
- Is this enough?
- Yes...but!

SPOON 🥄

SPOON

- Distribute tests to different devices
- possibility to specify those devices
- Run on different versions of Android
- Take screenshots during critical parts of your tests
- Save important files like DBs
- Test sharding!

SOUNDS GOOD, SOUNDS FUN, BUT HOW?

- Gradle Spoon plugin (recommended)
- `classpath 'com.stanfy.spoon:spoon-gradle-plugin:1.2.2'`
- ⚠️ some problems with Android Studio 3.0
- Guys are working on it. There is a snapshot (2.0) available at the moment
- `spoonDebugAndroidTest`

HOW WE WORK AT XING

- ~25 developers
- Split in independent feature teams
- Release Trains 🚂
- Code Freeze every 2 weeks
- Followed by a Release + Rollout (20% -> 50% -> 100%)
- Work on feature branches, merge to master
- Current # of open PRs: ~40

- We use Jenkins CI
- Running Unit Tests, Static Analyzers, assemble different build types etc.
- On Every PR
- Run UI Tests on CI

- Huge load on Jenkins
- Waiting for long running UI tests
- Jenkins without UI (only raw metal - Linux machines)
- Tests were failing / slow / flaky
- shell scripts magicians? 🎩

INTEGRATIONWITHSPOON.SH

```bash
#!/usr/bin/env bash

# Configure pre-conditions
PACKAGE_NAME="com.xing.android"
AVD_NAME="integration-tests"
PORT=${1-6000}

##############################################################################

# Ensure the Test APK is built already.
TEST_APK_FILE="core-app/build/outputs/apk/core-app-debug-androidTest.apk"
if [ ! -f "${TEST_APK_FILE}" ]
then
    echo "Test APK doesn't exist, aborting. Make sure you run ./gradlew :core-app:assembleDebug :core-app:assembleDebugAndroidTest"
exit
else
    echo "androidTest APK Exists, continuing"
fi

# Calculate the Serial Number of the emulator instance
SERIAL=emulator-${PORT}

echo "Creating (forceful) AVD with name ${AVD_NAME}"
# We have to echo "no" because it will ask us if we want to use a custom hardware profile, and we don't.
echo "no" | android create avd \
    -n "${AVD_NAME}" \
    -k "system-images;android-22;default;x86_64" \
    -f
echo "AVD ${AVD_NAME} created."

# Start the Android Emulator
# "2>&1" combines stderr and stdout into the stdout stream
START_EMULATOR="/opt/android-sdk-linux/tools/emulator \
    -avd ${AVD_NAME} \
    -netspeed full \
    -netdelay none \
    -no-skin \
    -no-window \
    -gpu guest \
    -port ${PORT}"

echo $START_EMULATOR
$START_EMULATOR 2>&1 &

# Ensure Android Emulator has booted successfully before continuing
EMU_BOOTED='unknown'
MAX_RETRY_COUNT=27
while [[ ${EMU_BOOTED} != *"stopped"* ]]; do
    sleep 7
    EMU_BOOTED=`adb -s ${SERIAL} shell getprop init.svc.bootanim || echo unknown`

    # Exit if the emulator didin't start in 140 seconds.
    MAX_RETRY_COUNT=$(($MAX_RETRY_COUNT - 1))
    if [[ $MAX_RETRY_COUNT -eq 0 ]]; then
        echo "Emulator startup timeout. Aborting"
        exit 1
    fi
done

duration=$(( SECONDS - start ))
echo "Android Emulator started after $duration seconds."

# Use the Spoon utility as a test runner
SPOON_COMMAND="./gradlew --no-daemon spoonDebugAndroidTest -PspoonDevice=emulator-${PORT}"
echo "Running: ${SPOON_COMMAND}"
${SPOON_COMMAND}
```

KILL-EMULATOR.SH

```
#!/usr/bin/env bash

##############################################################
#
# KILL-EMULATOR
#
# Kills an Android emulator which requires authentication.
# It works by opening a telnet session and authenticates, before issuing the
# kill command.
#
# Usage: `kill-emulator.sh <port>`
# where <port> is optional (defaults to 6000)
#
# Since SDK Tools 25.1.6, the Android emulator has required authentication
# before any commands can be run on it. This breaks commands such as
# `adb emu kill`.
#
# References:
# - https://developer.android.com/studio/run/emulator-commandline.html#console-session
# - https://code.google.com/p/android/issues/detail?id=21021#
#
##############################################################

# Read port form the console
PORT=${1-6000}
# Read token for emulator
TOKEN=$(<$HOME/.emulator_console_auth_token)

# Notify user that everything is going to be OK
echo "Killing emulator on port $PORT with auth token $TOKEN"

# Start telnet and pray that it will work
TELNET=`(
    echo "auth $TOKEN";
    sleep 1;
    echo "kill";
    sleep 1
) | telnet localhost $PORT | grep "OK: killing emulator, bye bye"`
if [ "$?" -ne 0 ]; then
  echo "Couldn't kill emulator $PORT. Aborting"
  exit 1
else
  echo "Emulator dead"
  exit 0
fi
```

"# Start telnet and pray that it will work"
- kill-emulator.sh -

Why?

- Unmaintainable
- What if you die?
- What if one of the scripts fails
- Too many cases you can't handle
- Flaky / Slow
- People will not trust in tests
- And bother you a lot...

PROBLEM: HOW DO YOU CREATE/MANAGE EMULATORS ETC. ON JENKINS?

FASTLANE (TO THE RESCUE)

FASTLANE

- ruby tool to handle tedious tasks
- mainly focussed on releasing applications
- super cool
- has lots of plugins
- huge community
- "over *10,391,703* Developer Hours Saved"

- `sudo gem install fastlane -NV`
- `fastlane init` **inside your existing project**
- **Ready to go!**
- **Create your 'lanes' (definitions of tasks) inside** `Fastfile`

Espresso lane

```
desc "Run UI tests using default test runner"
lane :espresso_test do
  gradle(task: "cAT")
end
```

Run command: `fastlane espresso_test`

spoon lane

```
desc "Run UI tests using spoon"
  lane :espresso_spoon_test do
    gradle(task: "spoonDebugAndroidTest")
  end
```

Run command: `fastlane espresso_spoon_test`

- How does this help us?
  - it doesn't
- we still have the same problem with emulators
  - what to do?
  - Plugin magic

FASTLANE-PLUGIN-AUTOMATED-TEST-EMULATOR-RUN

- Wraps gradle/shell tasks
- Creates and manages emulators
- easy to configure
- start multiple emulators
- `fastlane add_plugin automated_test_emulator_run`
- create AVD(emulator) config using JSON

```
{
    "avd_list": [
        {
          "avd_name": "Test-Emulator-API23-Nexus-5-1",

          "create_avd_package": "system-images;android-23;google_apis;x86_64",
          "create_avd_device": "Nexus 5X",
          "create_avd_tag": "google_apis",
          "create_avd_abi": "x86_64",
          "create_avd_additional_options": "",
          "create_avd_hardware_config_filepath": "~/Android/AVD_Snapshots/Nexus_5X_API_23/Test-Emulator-API23-Nexus-5-1.ini",

          "launch_avd_port": "",
          "launch_avd_snapshot_filepath": "~/Android/AVD_Snapshots/Nexus_5X_API_23/Nexus_5X_API_23_SNAPSHOT.img",
          "launch_avd_launch_binary_name": "emulator",
          "launch_avd_additional_options": "-gpu on"
        },
        {
          "avd_name": "Test-Emulator-API23-Nexus-5-2",

          "create_avd_package": "system-images;android-26;google_apis;x86_64",
          "create_avd_device": "Nexus 5X",
          "create_avd_tag": "google_apis",
          "create_avd_abi": "x86_64",
          "create_avd_additional_options": "",
          "create_avd_hardware_config_filepath": "~/Android/AVD_Snapshots/Nexus_5X_API_26/Test-Emulator-API26-Nexus-5-2.ini",

          "launch_avd_port": "",
          "launch_avd_snapshot_filepath": "~/Android/AVD_Snapshots/Nexus_5X_API_23/Nexus_5X_API_26_SNAPSHOT.img",
          "launch_avd_launch_binary_name": "emulator",
          "launch_avd_additional_options": "-gpu on"
        }
    ]
}
```

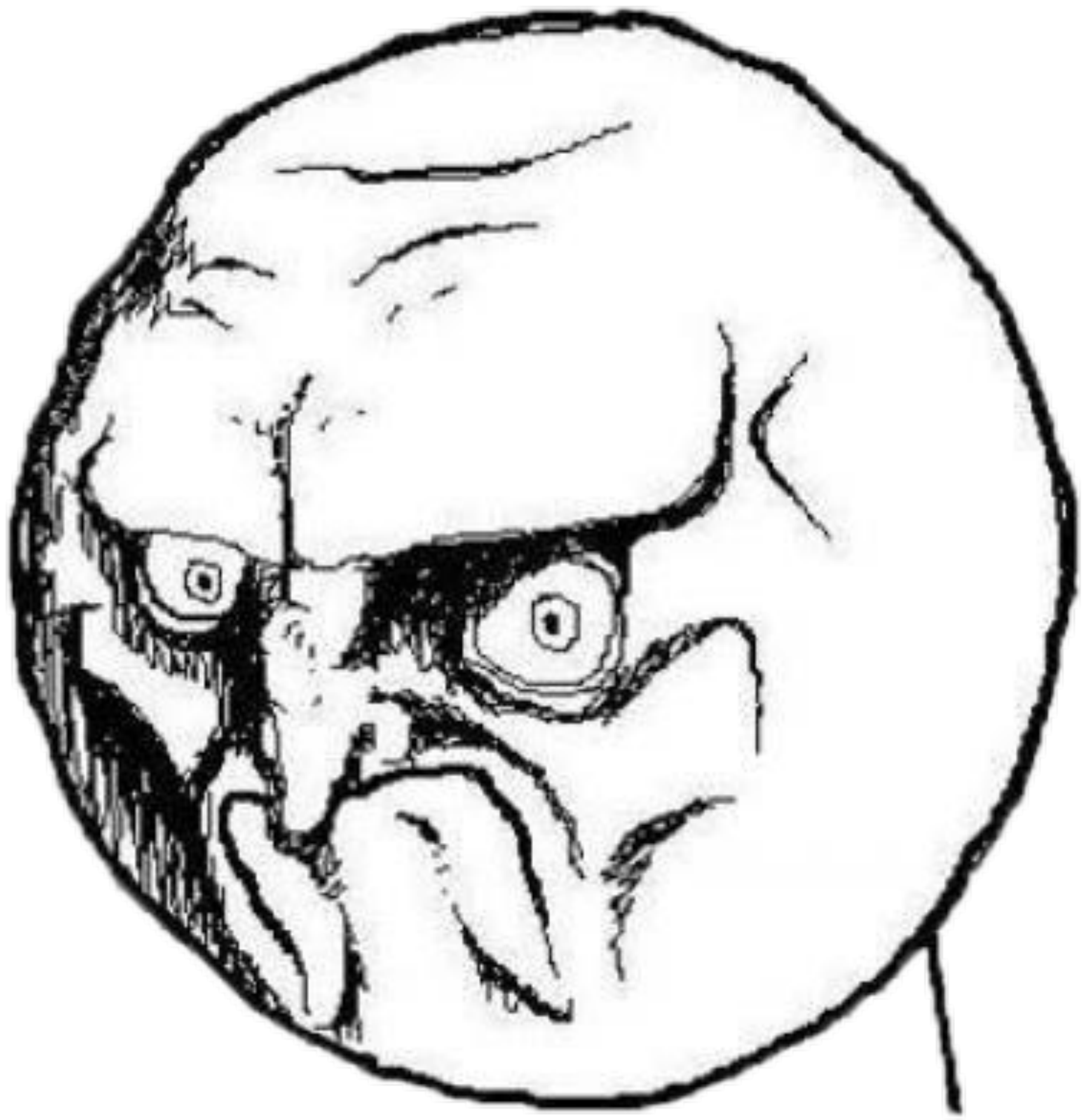
- You can configure everything from here you would normally need to do from command line
- It is easy to read
- Other people can maintain/tweak it
- It scales (why not use 3,4 or 5 emulators?)

- Now we can create lanes to use this plugin
- desc "Runs tests with AVD setup according to JSON file config with usage of spoon."

```
lane :Automation_EmulatorRun_Spoon do
  automated_test_emulator_run(
    AVD_setup_path: "fastlane/avdconfig/AVD_setup.json",
    gradle_task:"spoonDebugAndroidTest"
  )
end
```

- Now let's run it.
- `fastlane Automation_EmulatorRun_Spoon`

- Yayyyy!
- This handles emulator creation
- We can run multiple emulators with different versions
- easy to maintain
- are we done?



NO.

- We are still ~25 developers
- We have 4 Jenkins slaves
- 1 slave = 1 computer
- 8 nodes per slave (same computer)
- multiple ui test jobs can run on the same jenkins slave

- spoon tests are executed on all connected devices
- Tests on the same node interfere with each other
- Tests get flaky again.
- Tests are still slow

REMEMBER!!! - YOU CAN SPECIFY TARGET DEVICES ON SPOON

- We don't know what are the names of the emulators that get created by the plugin
  - Something with emulator-${portNumber}
- How do we connect our spoon test run with the created emulators?

Spoon configuration (build.gradle)

```
spoon {
            debug = true
            shard = true
            devices = ['emulator-5556', 'emulator-5558']
}
```

- You can specify target devices using the devices array

MHHHH 🤔

- This is it!
- We need a way to set devices from our emulator plugin
- It knows which ports it assigned
- And thus can pass the names to the gradle task it executes

- The current plugin does not support this.
- But it's open source
- So we can tweak it!

```
# Launching tests
          shell_task = "#{params[:shell_task]}" unless params[:shell_task].nil?
          gradle_task = "#{params[:gradle_task]}" unless params[:gradle_task].nil?
          spoon_task = "#{params[:spoon_task]}" unless params[:spoon_task].nil?
```

- Created `spoon_task`

TWEAKING THE PLUGIN

```
            ports = Array.new
            spoon_devices = " -PspoonDevice="
            spoon_devices = spoon_devices + "emulator-" + avd_schemes[0].launch_avd_port.to_s
            for i in 1...avd_schemes.length
              ports << avd_schemes[i].launch_avd_port
              spoon_devices = spoon_devices + ",emulator-" + avd_schemes[i].launch_avd_port.to_s
            end

            gradle_spoon_task = params[:spoon_task]
gradle_spoon_task = gradle_spoon_task + spoon_devices
```

- Create a new lane inside your Fastfile
- use `spoon_task` instead of `gradle_task`
- -pSpoonDevice will be passed to the task that is executed
- We need to read this value in build.gradle
- And configure our spoon to run on the passed emulators

./gradlew spoonDebugAndroidTest -pSpoonDevice=emulator-5556, emulator-5558

```
spoon {
          if (project.hasProperty('spoonDevice')) {
              devices = []
              project.properties['spoonDevice'].split(',').each {
                  devices += [it]
              }
          }
```

- **This reads -pSpoonDevice flag**
- **and propagates `devices[]` with emulator names**

- Now the tests will execute only on the devices created for the fastlane task we execute
- Tasks are now independent from each other
- Less failure due to interference
- Test runs get way faster
- Developers like if they don't need to retry to run the tests multiple times

✅

MISSION ACCOMPLISHED

✅

▶ ?

REMAINING PROBLEMS

- Sometimes can still be flaky (ADB issues)
- Emulators still don't get shut down properly
- Too much load on jenkins
- No retrying of flaky tests
- Still not perfect, but improving the current situation

THANK YOU ❤

QUESTIONS?