iOS Build that SCALES

ME in emojis

tobias hutzler

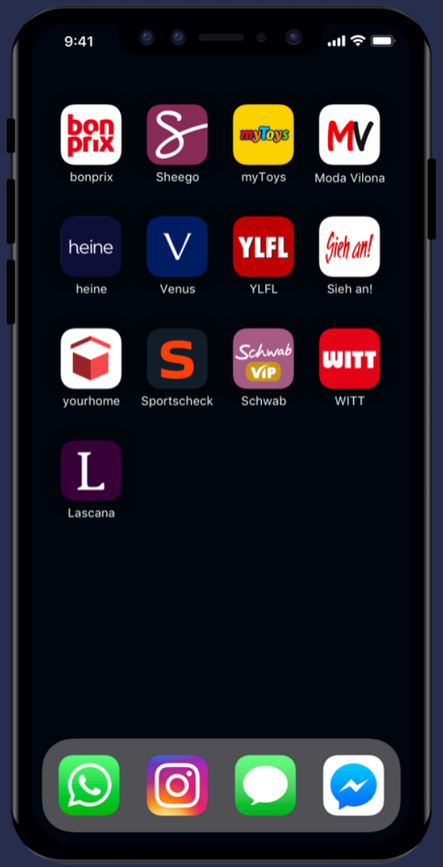
engineering lead @ otto group

AGENDA

- Challenges
- Requirements
- Our Approach

CHALLENGES *What we do*

[App Skeleton]



CHALLENGES continued

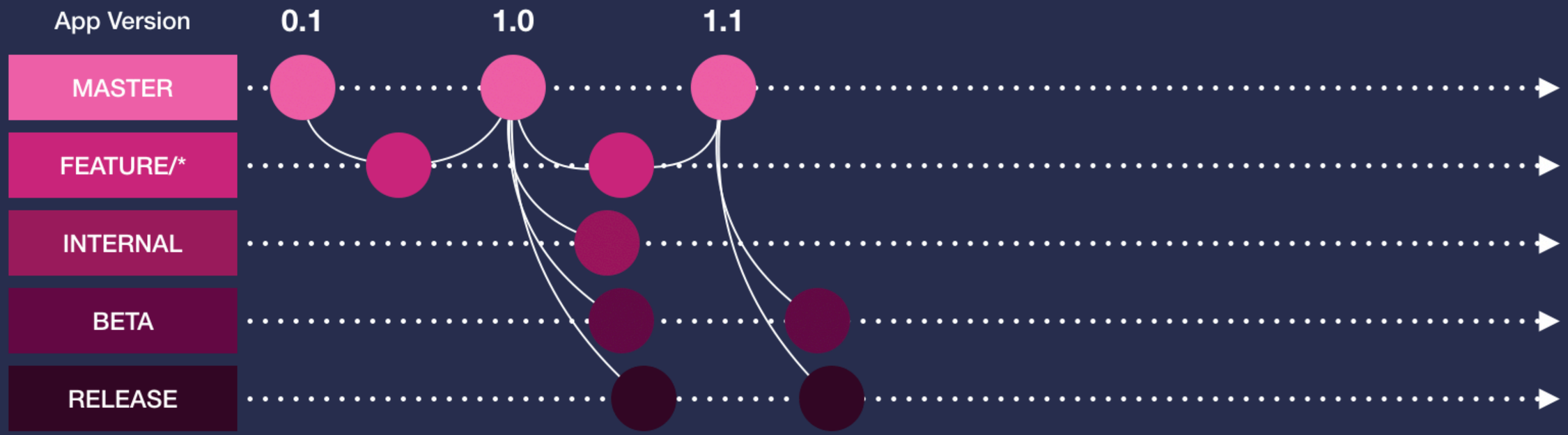
- build and ship dev, internal, beta, release
- support different environments and versions (Xcode, MacOS, etc.)
- signing for diffrent companies
- parallel builds

REQUIREMENTS CI Pipeline

- all things automated (AMAP)
- one git push shipping
- stable and flexible
- keep it simple

CI/CD techstack

CI/CD pipelines & trigger branches



FASTLANE lanes & steps

BRANCHES	FASTLANE LANE	STEPS
wildcard	default	build > test > slack
beta	beta	build > test > sign (enterprise certificate) > create release notes > upload to Crashlytics (customer test group) > slack
internal	internal	build > test > sign (enterprise certificate) > create release notes > upload to Crashlytics (interal test group) > slack
nightly	nightly	build > test > sign (company certificate) > slack
release	release	build > test > sign (company certificate) > itc upload > slack

FASTLANE *test lane*

```
desc "Run Test Suite"
private_lane :test do |options|

  test_device = 'iPhone 7'
  reset_simulator(test_device)
  scan(
      clean: true,
      scheme: "app",
      device: test_device,
      configuration: "Debug",
      skip_build: true
  )
end
```

FASTLANE upload to crashlytics

```
desc "upload to crashlytics"
private_lane :crashlytics_upload do |options|
  groups = options[:groups] || "mobile_lab_qa"
  crashlytics(
    groups: groups,
    api_token: fabric_api_key,
    build_secret: fabric_build_secret,
    notes: release_notes
  )
  upload_symbols_to_crashlytics(api_token: fabric_api_key)
end
```

FASTLANE beta lane

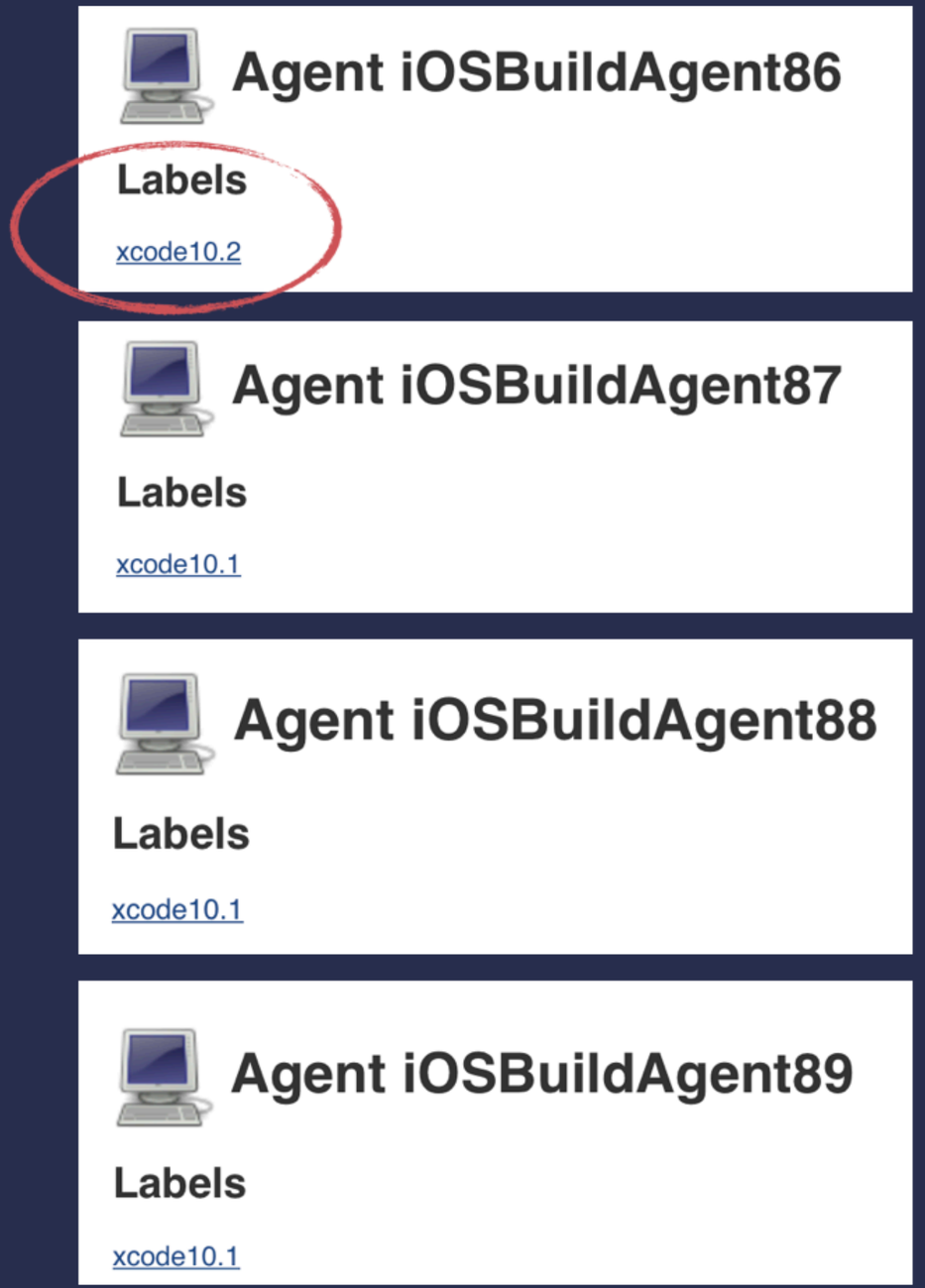
```
desc "Build a beta release, upload it to crashlytics and distribute it to all beta testers."
lane :beta do
  puts "Hi my name is beta!"
  test
  update_app_version
  install_beta_certificates(readonly: true)
  gym(
    export_method: "enterprise",
    configuration: "Beta",
    export_options: { compileBitcode: false },
    scheme: "app",
    sdk: "iphoneos",
    clean: true
  )
  crashlytics_upload(groups: $crashlytics_beta_groups)
  post_to_slack
end
```

JENKINS PIPELINES Define Steps

```
pipeline {
    stages {
        stage('Stage Checkout') {
            steps {
                deleteDir()
                checkout scm
                sh 'git submodule update --init'
                sh 'git fetch --tags'
            }
        }

        stage ('Stage run Build-Pipeline') {
            steps {
                echo "Run pipeline"
                sh "./build.sh"
            }
        }
    }
}
```

JENKINS PIPELINES Run on dedicated Agent

```
pipeline {
    agent { label 'xcode10.2' }
    options {
        ansiColor('xterm')
    }
    environment {
    SLACK_HOOK_URL          = credentials('slack_hook_url')
        FABRIC_API_KEY          = credentials('vns_fabric_api_key')
        FABRIC_BUILD_SECRET = credentials('vns_fabric_build_secret')
        ITC_DELIVER                 = credentials('mobilelab_itc_deliver')
        MATCH_PASSWORD          = credentials('match_password')
        }
    stages {
        stage('Stage Checkout') {
            steps {
                deleteDir()
                checkout scm
                sh 'git submodule update --init'
                sh 'git fetch --tags'
            }
        }

        stage ('Stage run Build-Pipeline') {
            steps {
                echo "Run pipeline"
                sh "./build.sh"
            }
        }
    }
}
```



JENKINS PIPELINES We ❤ nice output

```
pipeline {
    agent { label 'xcode10.1' }
    options {
        ansiColor('xterm')
    }
    environment {
    SLACK_HOOK_URL          = credentials('slack_hook_url')
        FABRIC_API_KEY          = credentials('vns_fabric_api_key')
        FABRIC_BUILD_SECRET = credentials('vns_fabric_build_secret')
        ITC_DELIVER                 = credentials('mobilelab_itc_deliver')
        MATCH_PASSWORD          = credentials('match_password')
        }
    stages {
        stage('Stage Checkout') {
            steps {
                deleteDir()
                checkout scm
                sh 'git submodule update --init'
                sh 'git fetch --tags'
            }
        }

        stage ('Stage run Build-Pipeline') {
            steps {
                echo "Run pipeline"
                sh "./build.sh"
            }
        }
    }
}
```

```
[11:43:42]: ------------------------------
[11:43:42]: --- Step: default_platform ---
[11:43:42]: ------------------------------
[11:43:42]: Driving the lane 'ios select_lane' 🚀
[11:43:42]: Hi before all! I'm going to check if we have a clean state
[11:43:42]: ----------------------------------
[11:43:42]: --- Step: ensure_git_status_clean ---
[11:43:42]: ----------------------------------
[11:43:42]: $ git status --porcelain
[11:43:42]: Git status is clean, all good! 💪
[11:43:42]: --------------------------------
[11:43:42]: --- Step: clear_derived_data ---
[11:43:42]: --------------------------------
[11:43:42]: Derived Data path located at: /Users/****/Library/Developer/Xcode/DerivedData
[11:43:43]: Successfully cleared Derived Data ♻️
[11:43:43]: Hi my name is select_lane and i will immediately switch to a lane!
[11:43:43]: ------------------------
[11:43:43]: --- Step: git_branch ---
[11:43:43]: ------------------------
[11:43:43]: Hey I am master! And not special! Lets run on default!
[11:43:43]: ---------------------------------------
[11:43:43]: --- Step: Switch to ios default lane ---
[11:43:43]: ---------------------------------------
[11:43:43]: Cruising over to lane 'ios default' 🚘
[11:43:43]: Hi my name is default!
[11:43:43]: ----------------------------------
[11:43:43]: --- Step: Switch to ios test lane ---
[11:43:43]: ----------------------------------
[11:43:43]: Cruising over to lane 'ios test' 🚘
[11:43:44]: Resetting iPhone 7
[11:43:45]: ------------------
[11:43:45]: --- Step: scan ---
[11:43:45]: ------------------
[11:43:45]: $ xcodebuild -showBuildSettings -workspace ./iOSAppLascana.xcworkspace -scheme app -configuration Debug
[11:43:48]: $ xcodebuild -showBuildSettings -workspace ./iOSAppLascana.xcworkspace -scheme app -configuration Debug
```

JENKINS PIPELINES *Secrets*

```
pipeline {
    agent { label 'xcode10.1' }
    options {
        ansiColor('xterm')
    }
    environment {
    SLACK_HOOK_URL          = credentials('slack_hook_url')
        FABRIC_API_KEY          = credentials('vns_fabric_api_key')
        FABRIC_BUILD_SECRET = credentials('vns_fabric_build_secret')
        ITC_DELIVER                 = credentials('mobilelab_itc_deliver')
        MATCH_PASSWORD          = credentials('match_password')
        }
    stages {
        stage('Stage Checkout') {
            steps {
                deleteDir()
                checkout scm
                sh 'git submodule update --init'
                sh 'git fetch --tags'
            }
        }

        stage ('Stage run Build-Pipeline') {
            steps {
                echo "Run pipeline"
                sh "./build.sh"
            }
        }
    }
}
```

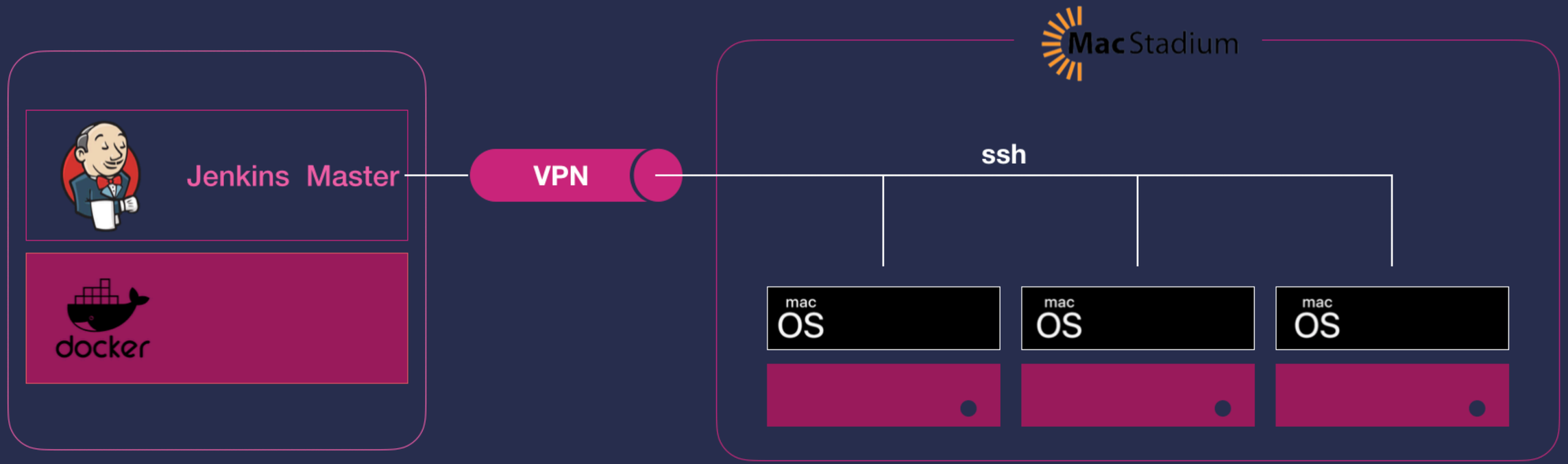
JENKINS PIPELINES That's all

```
pipeline {
    agent { label 'xcode10.1' }
    options {
        ansiColor('xterm')
    }
    environment {
    SLACK_HOOK_URL          = credentials('slack_hook_url')
        FABRIC_API_KEY          = credentials('vns_fabric_api_key')
        FABRIC_BUILD_SECRET = credentials('vns_fabric_build_secret')
        ITC_DELIVER                 = credentials('mobilelab_itc_deliver')
        MATCH_PASSWORD          = credentials('match_password')
        }
    stages {
        stage('Stage Checkout') {
            steps {
                deleteDir()
                checkout scm
                sh 'git submodule update --init'
                sh 'git fetch --tags'
            }
        }

        stage ('Stage run Build-Pipeline') {
            steps {
                echo "Run pipeline"
                sh "./build.sh"
            }
        }
    }
}
```

BUILD.SH

```
#!/usr/bin/env bash
PROJECT_DIR=app

# 1. Check if bundle is installed if not install it

BUNDLE_BIN=$(which bundle)

if [ -e $BUNDLE_BIN ]; then
    echo "Bundle installed in $BUNDLE_BIN"
else
    echo "Installing Bundler"
    gem update
    gem install bundler
fi

# 2. Install gem

bundle install

# 3. Change into project folder

cd $PROJECT_DIR

# 4. Run fastlane with bundle

BRANCH=$(git branch | sed -n -e 's/^\* \(.*\)/\1/p')
echo "Running on Branch $BRANCH"
bundle exec fastlane select_lane
```

CI/CD architecture



HAPPY BUILDING  

THANK YOU

tobias.hutzler@ottogroup.com