Automating the release process

Jean and Lucas

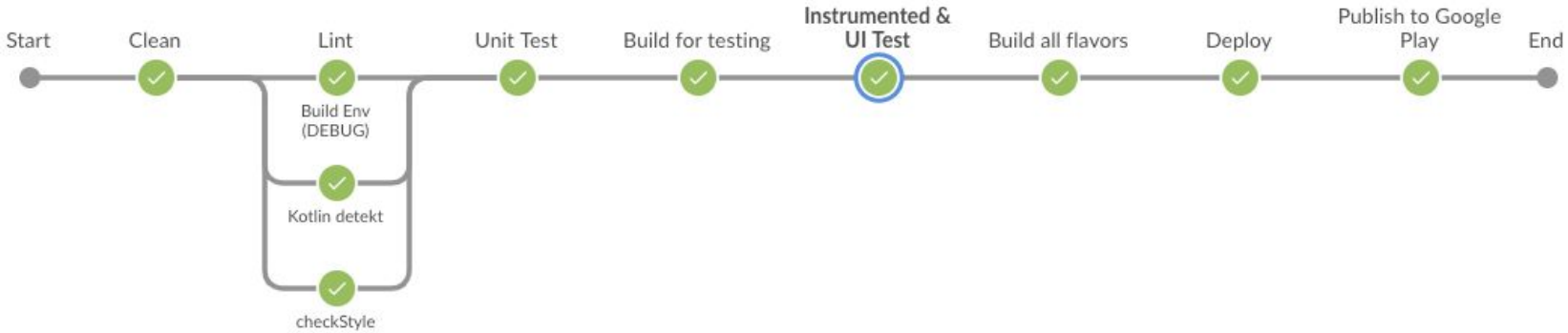
Growth Session #25 - XXV - June 20-21 2019

Intro

The CI/CD Jenkins pipeline for Android Red Planet

- Pipeline execution - 2hrs on average
- 2 pipelines execution per pull request
- Server only executes max of 2 in parallel
- Comes merging time after crunch time, waiting for up to 8 hrs is not uncommon, e.g. 4 merged PRs

Android RP Jenkins Server



Goals

Deliver faster, deliver easier (sabai sabai!)

- When preparing a release, still a few things done manually, e.g.
- Clerical tasks of creating proper gitflow branches
- Updating version code
- Phraseapp pull
- Localization diff, etc.

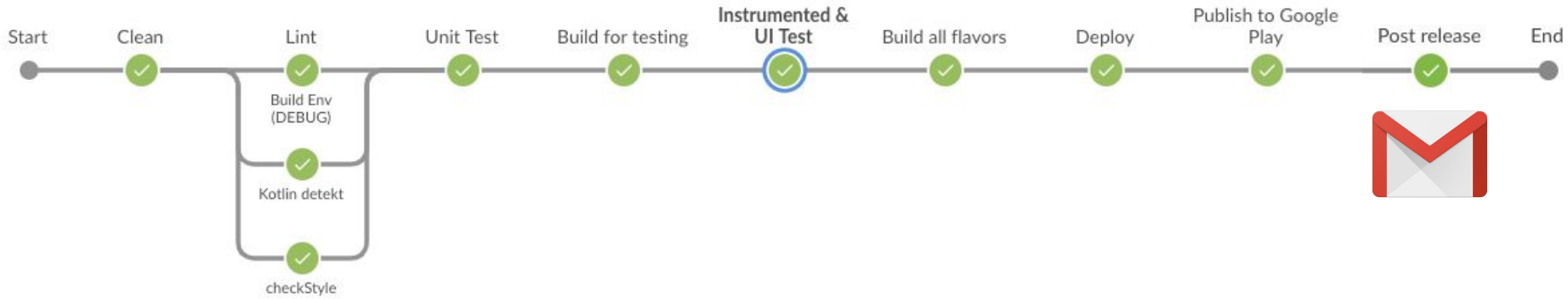
Release experience

- A checklist to go through
- Almost all clerical tasks
- Primed for automation

Beta release check list

1. Make sure all PR were merged
2. Make sure all string changes (add/update/remove) were up to dated on `phraseapp` from all merged PR to last beta
3. Pull latest `develop`, checkout new `beta` branch from `develop` (such as `beta/2.19.0-beta1`) and push to `origin`
4. Checkout new chore `prepare beta` branch (such as `chore/prepare-2.19.0-beta1`) and push to `origin`
- Create version update commit
- Create `phrase pull` commit: Goto `Tune` folder and do `phraseapp pull` command, check and ignore any unrelated string changes does not apply into current beta version
5. Create localization diff:
- Checkout and pull `release` branch (such as `release/2.19.0`) -> last beta
- Checkout back to `prepare beta` branch
- Run string diff script `./script/html_string_diff.sh` command at project root folder with `release` branch as `old beta` and `prepare beta` as `new beta` branch.
- A new `Localization-Diff.zip` file will be created at project root folder
6. Open `Prepare 2.19.0 beta1` PR to pull changes from `prepare beta` branch to `beta` branch, wait for review and merge
7. Open `Release 2.19.0 beta1` PR to pull changes from `beta` branch to `release` branch, wait for review and merge
8. Prepare beta notifier email
- Clone the content from our last emails
- Update `All APKs Folder` gdrive folder link: open current link, create new `Beta folder`, create a share link for this folder as `Anyone with the link can view` mode and update the new link in email.
- Update jira beta version filter for tester, e.g. https://jira.redplanethotels.com:8443/issues/?jql=project%20%3D%20AN%20AND%20status%20%3D%20%22Ready%20for%20Testing%22%20AND%20fixVersion%20%3D%20%222.19.0%20Build%20195%20(Beta%204)%22
- Update beta version name, build number and link to jira filter above
- Attach `Localization-Diff.zip` file into email
9. Wait for CI running on `release` branch to be finished, this CI will deployed **new version onto Fabric Beta + build apks** (StagingDebug, Staging_x86Debug, ProductionInternalRelease, ProductionInternal_x86Release)
- Upload all 4 apks from CI artifacts to gdrive `Beta folder` created above
- Send out new beta notifier email 🚀
10. Create PR to bring changes from `beta` branch back to `develop` (edited)

Android RP Jenkins Server



Demo

```
➜ ~ rimble --help
Usage: rimble [OPTIONS] COMMAND [ARGS]...

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  android-rp
➜ ~ rimble android-rp --help
Usage: rimble android-rp [OPTIONS]

Options:
  -d, --dev-branch TEXT   The name of the develop branch. Usually 'develop'
                          [required]
  -f, --repo-folder TEXT  The name of the folder to use when cloning the
                          repository  [required]
  -r, --repo-url TEXT     The full URL of the GitHub repository  [required]
  -a, --app-folder TEXT   The name of the app's folder, e.g. Tune  [required]
  -v, --version TEXT      Release version, e.g. 2.19.0  [required]
  -b, --beta TEXT         Beta version, e.g. 6  [required]
  -u, --username TEXT     Your GitHub username  [required]
  -p, --password TEXT     Your GitHub password  [required]
  --help                  Show this message and exit.
➜ ~
```

Step 1: Clerical tasks

Rimble - Release Nimble

A CLI to automate the admin work required for a mobile app release

⇒ Potential to be applied for all Nimble releases

- ios-rp
- android-the1
- ios-the1

Step 2: Communicate the release details

Remail - Release email

A python script to act as a stage in a Jenkins CI to draft a release email

⇒ Potential to be in applied all projects' pipelines

Achievement and Progress

- Identified all components required to make this work
- How those components will interact together
- Started to glue them in a comprehensive fashion

Next Steps

- Make release steps to be configurable, akin to phraseapp pull and **.yaml**
- Use service account oauth2
- Finish automated release flow for RP Android
- Apply process for other projects e.g. RP iOS 🥶
- Last but not least, formalize role of **Release Manager**

Automate everything!

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit #2402-03, Klong Toei, Wattana, Bangkok 10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower

28 Queen's Road, Central, Hong Kong

nimble