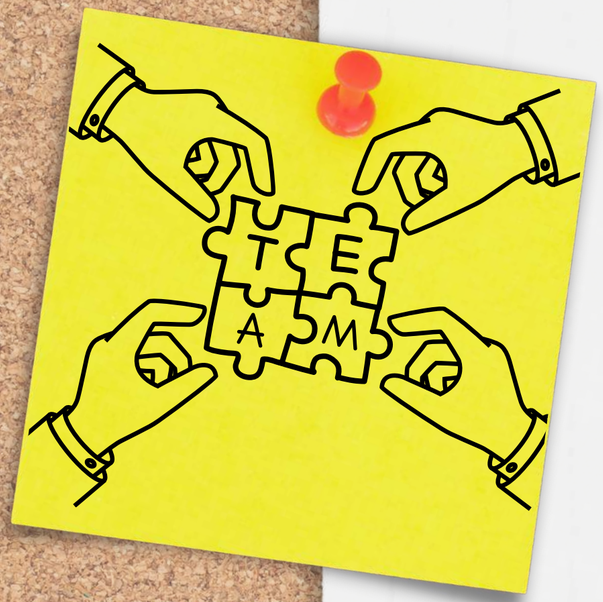


Beyond CODE

Building a Stronger Development Team

Nikos Kleidis

Mobile app tech lead @ Ferryhopper

About me

2012 - 2015
FullStack VB/Java

2015 - 2020
React/React-Native

2020 - 2023
Senior React Engineer

2023 - now
Mobile app tech lead

Nikos Kleidis

Code quality

- Follow the team's practices
- Perform code reviews
- Find room for improvements

developer todos...

Implement features

- Read the specs
- Implement designs
- Fix potential bugs

Tech lead todos...

Code quality

- Set the best practices
- Look for new trends in the react-native field
- Code review all the PRs
- Find room for improvements

Manage dev team

- Set up daily meetings
- Do 1-1s to get feedback
- Address individual wishes

Talk with QA team

- Do they know what to test?
- Do they have trouble with the communication?
- Are they adding reproduction steps to the tickets they open?

Talk with Product

- Are the specs written well?
- What is a feature's lifecycle?
- Are the features being delayed?

Organise sprints

- How many tasks can the team handle?
- Who adds the story points?
- How can I squeeze in tech debt into the sprint?

Pipelines / Automations

- Are task statuses updated automatically?
- Does deployment need manual steps?
- What happens when a PR is merged?

Steps

1. TALK WITH MY TEAM
2. TALK WITH QA
3. TALK WITH PRODUCT
4. BRAINSTORM
5. PERFORM ACTIONS
6. TALK AGAIN WITH EVERYONE!

App lifecycle

ClickUp

Everything starts will a well defined ticket!

01

1. Code is pushed to Github
2. PR is code reviewed
3. Code is merged to master

02

App Center

1. Connected to each PR
2. Builds code
3. Distribute code to groups by email

03

BrowserStack

1. AppCenter sends builds to BrowserStack
2. Builds are tested by QA team

04

1. Bundles are uploaded to Google Play Console and App Store Connect
2. Deployment is made by following the steps on the browser

QA

- We don't know the branch that we are testing
- We don't know what fixes the new build contains

Dev team

- Ticket status is updated manually by the devs (IN_PROGRESS -> CODE_REVIEW -> READY_FOR_QA)

git history

- commit messages don't follow a certain pattern
- hard to do git blame

Dev team

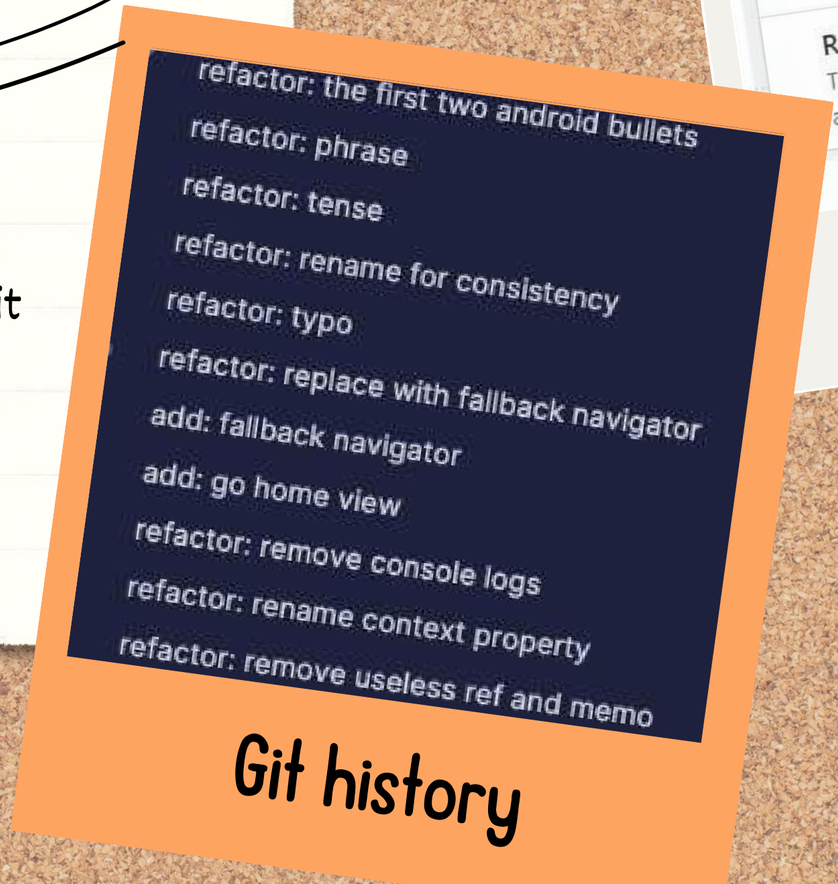
- Release process has a lot of manual steps
- ClickUp tickets are marked one by one on a new release

Product

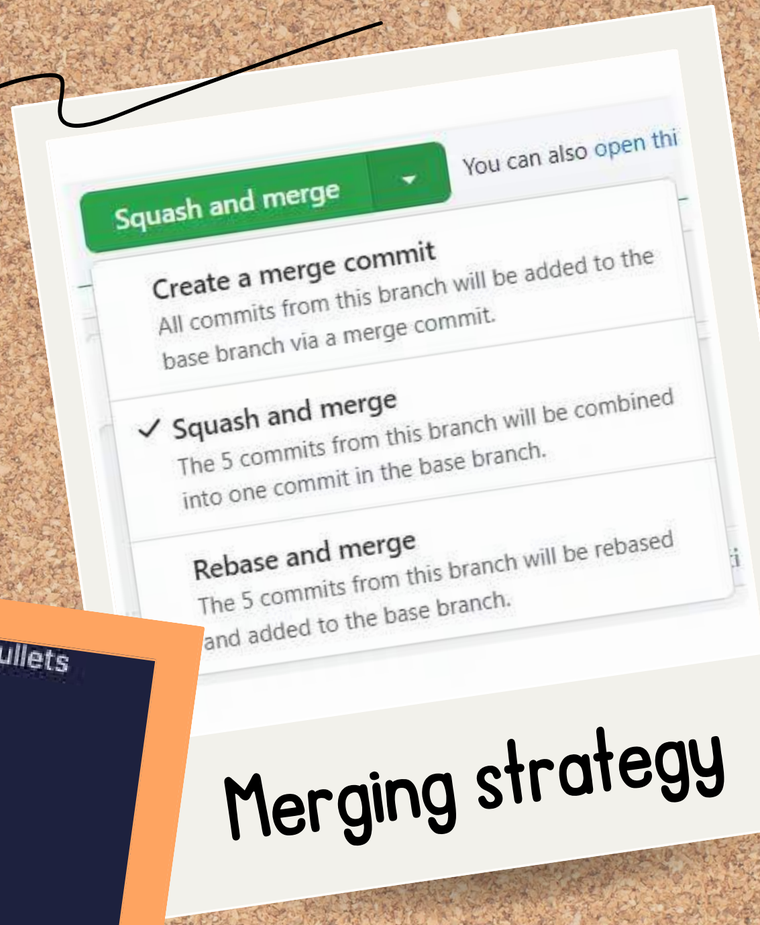
- We don't know the version that the ticket was released at
- We don't know what will be included in the upcoming version

How to merge

- Team likes to do small commits
- Should we merge or squash commits?
- Ticket number not mentioned on commit message - Hard to git blame



Git history



Merging strategy

The new git history

refactor: the first two android bullets

refactor: phrase

refactor: tense

refactor: rename for consistency

refactor: typo

refactor: replace with fallback navigator

add: fallback navigator

add: go home view

refactor: remove console logs

refactor: rename context property

refactor: remove useless ref and memo

before

- ✅ One commit per PR
- ✅ More descriptive git blame
- ✅ Linear git history

MOB-3190 MOB-3206 - Special offer fixes (#2959)

MOB-3169 Fix dob error event v3 (#2956)

MOB-3198 - Add knip to identify unused code (#2949)

MOB-3190 - Add special offer events (#2955)

MOB-2852 Fix the navigation (#2950)

MOB-2768, MOB-2885 - Add special offer api (#2896)

MOB-3194 - Fix images not found (#2943)

MOB-2787- Fix language on calendar (#2953)

MOB-3196 Disable the left arrow when the min date is s

MOB-3169 Fix dob event logging. (#2948)

MOB-2556 - Scroll to top safely (#2942)

MOB-2852 Open calendar modal when clicking on the c

MOB-3169 Fix missing events (#2941)

after

QA struggles....

- What is the environment?
- How can I test the same code on Android and iOS?
- What tickets are fixed in this build?

A new version of **Ferryhopper for iOS** is available.

Ferryhopper
1.31.0 (1688131447)
for iOS

You need to add a device to your account before you can start testing this app.

Add device

What's new

refactor: formatting

Manage your notification settings

AppCenter build

How can I fix this?

- There is an script called appcenter-pre-build.sh script. What can I do with it?
- My scripting knowledge is limited... (I am great with "cd" and "ls" commands though!)
- BUT... I have a couple of friends that can help me.....

Meet the gang!

1. They know scripting!
2. They can think fast!
3. They have time to listen!

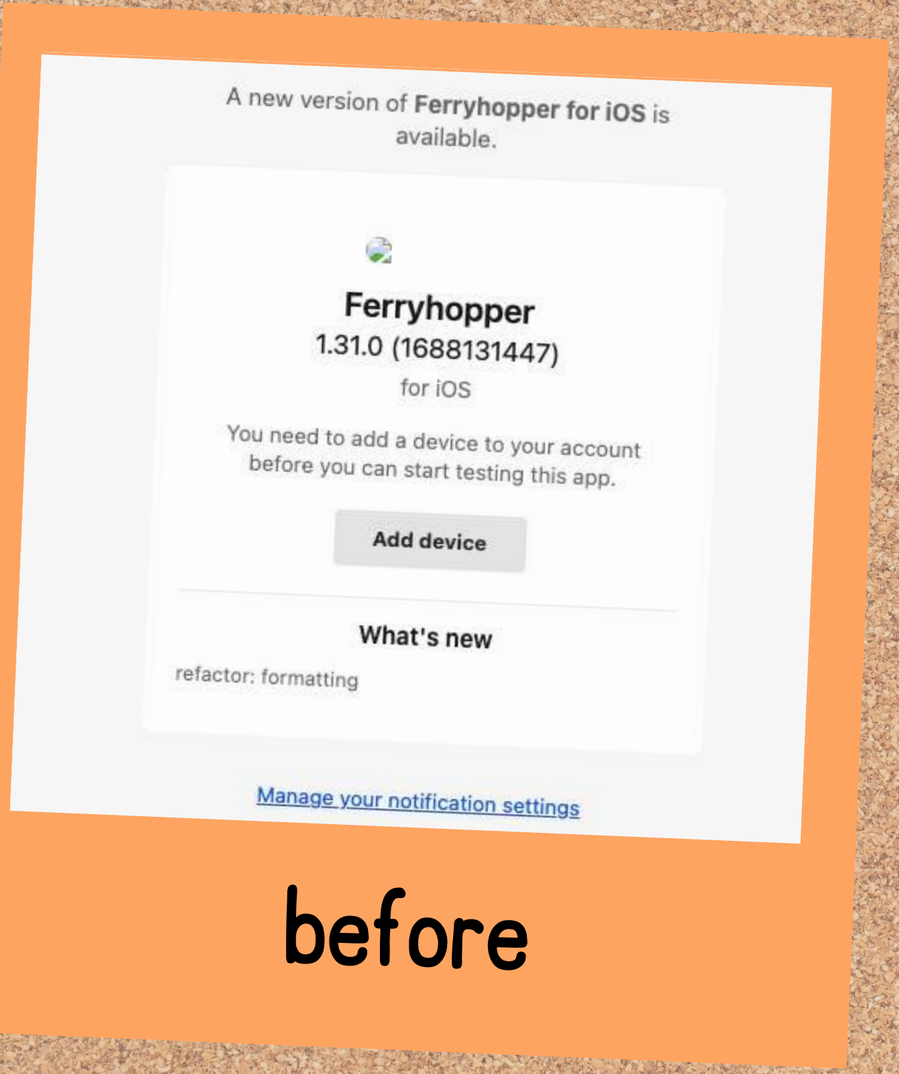
ChatGPT

friend by correspondence

Copilot

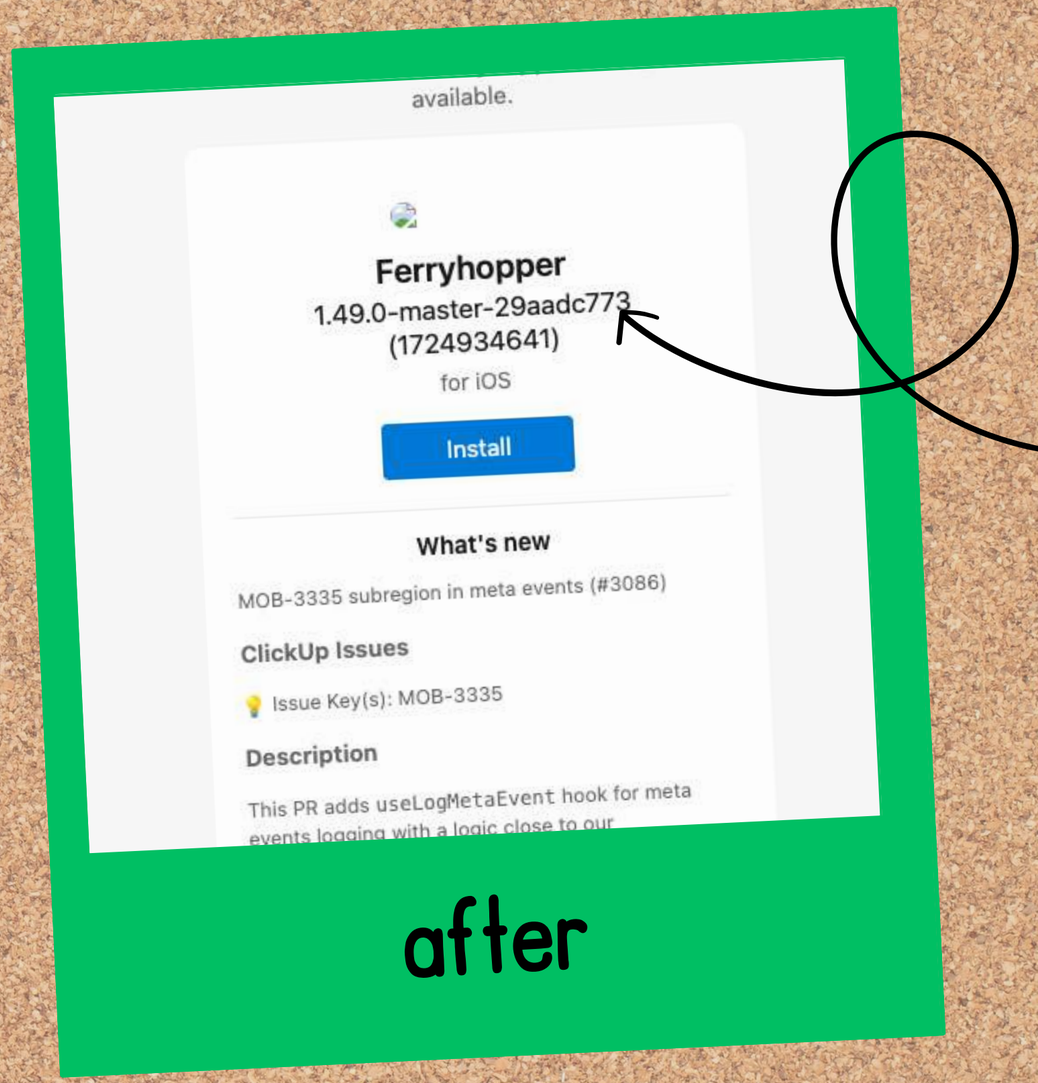
coworker

The new AppCenter Builds



before

- branch is visible
- commit hash is visible.
- Ticket ID and description



after

QA team is now happy!

Automations

- Does ClickUp offer automatic status updates based on PR updates? YES!
- So there is a Github-ClickUp integration, right? YES!
- Does it work?? YES and NO...

GitHub

Code repo

ClickUp

Project management platform

What can be done?

- I've heard that it can be done with Github Actions...
- Can I write it in JavaScript? YES
- Is there a ClickUp API to use for status updates? YES!
- Who can help me with the Github Action though? The gang!!
- How hard can it be?
- LET'S DO IT!!

.github/workflows/status-update.yml

```yaml
name: 'Update ClickUp status'

on:
  pull_request:
    types:
      - opened
      - edited
      - closed
      - ready_for_review

jobs:
  update_clickup_status:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: npm install
        run: npm i @actions/core @actions/github

      - name: Update ClickUp status
        run: npx zx ./scripts/status-update-action.mjs
        env:
          CLICKUP_API_KEY: ${{ secrets.CLICKUP_API_KEY }}
```

./scripts/status-update-action.mjs

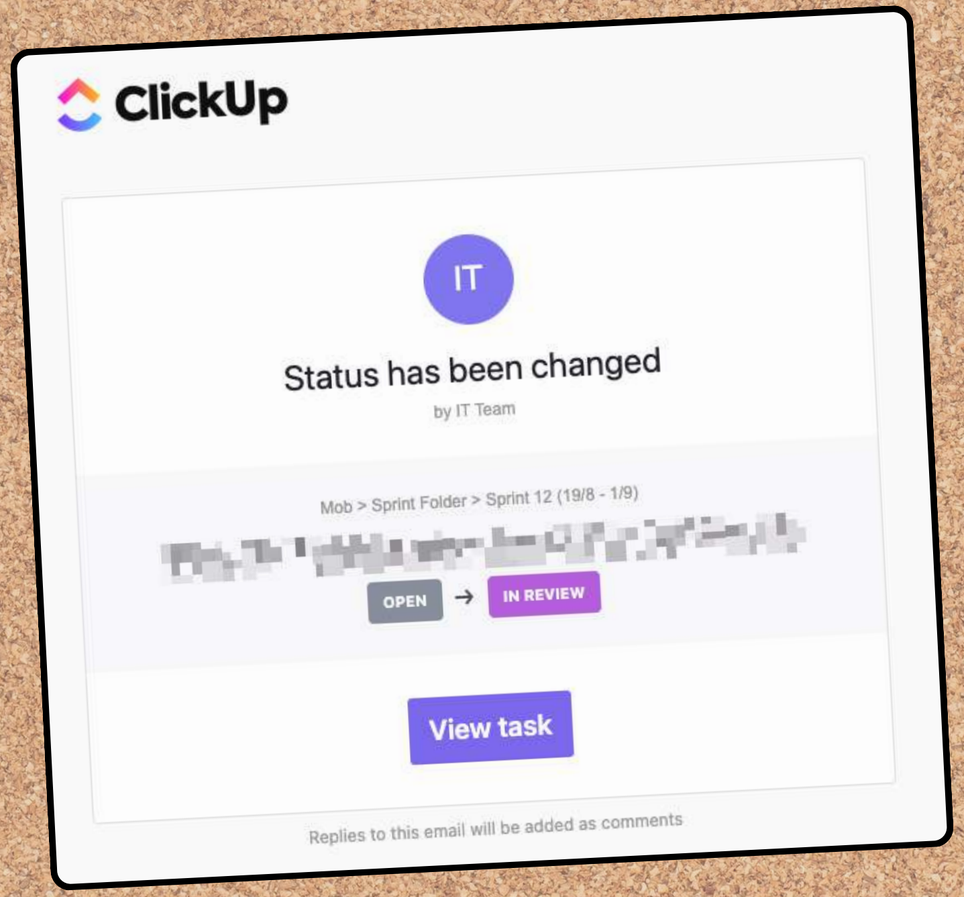
```javascript
// Retrieve pull request data
const pullRequestData = context.payload.pull_request;
const targetBranch = pullRequestData.base.ref;
const isDraft = pullRequestData.draft;
const isMerged = pullRequestData.merged;
const isClosed = pullRequestData.state === 'closed' && !isMerged;
const shouldSkipQA = isSkipQAChecked(pullRequestData.body);

// decide on next action
let newTaskStatus;
if (isClosed) {
  newTaskStatus = 'open';
} else if (isDraft) {
  newTaskStatus = 'in progress';
} else if (isMerged && targetBranch === MAIN_BRANCH) {
  newTaskStatus = shouldSkipQA ? 'completed' : 'qa';
} else {
  newTaskStatus = 'in review';
}

// get tasks to update from title
const taskIds = getTaskIds(pullRequestData.title);

// Call the ClickUp API for every task
taskIds.forEach(taskId =>
  setStatusToTask({ taskId, status: newTaskStatus })
);
```

Automatic status updates



Release process...

A lot of manual steps for updating the version name...

Tag ClickUp tickets one by one!!

Updating the native files

1. Grab the version from package.json
2. Set the version to the native files
3. Increase native versions by 1

Script ready!

```
zx scripts/update-native-versions.mjs
```

Tag tickets automatically

1. Grab the version from package.json
2. Find new commits `git log stable..HEAD --no-merges --pretty=oneline`
3. Extract ticket numbers from commits
4. Use the ClickUp API to tag the tickets

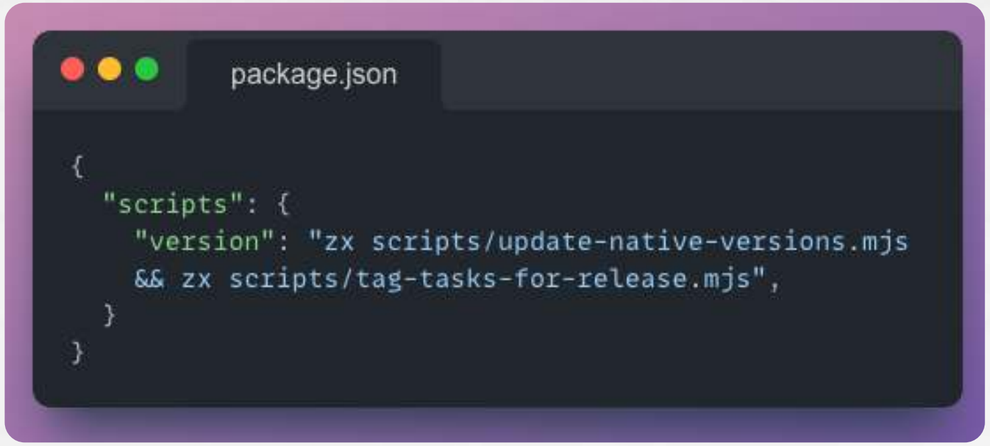
Script ready!

```
zx scripts/tag-tasks-for-release.mjs
```

Meet npm version!

npm version major | minor | patch

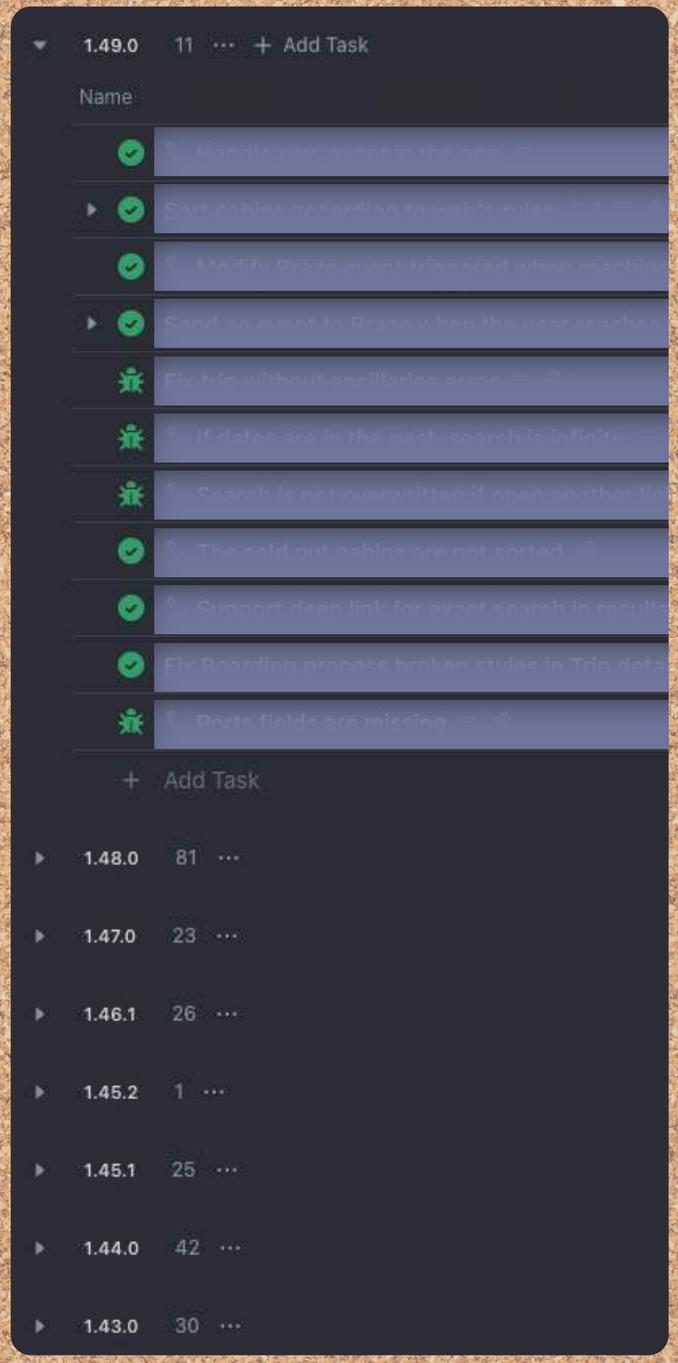
1. Bump the version number in package.json
2. Run the version script in package.json to update native versions and tag ClickUp tickets. (Don't forget to git add .)
3. Commit the changes
4. Tag the previous commit with the new version name



Now we just need to push!

The result looks great!

1. Great overview of what is released each time!
2. Automatically tag big releases!
3. Time to release decreased significantly!



SUCCESS!!!

Wait a sec...

Product is calling...

- P: hey! Nice tool you got there!
- D: Thanks! Bye!
- P: Wait a sec... Do you have 2 minutes to talk?
- D: Pfff ok!
- P: Could we enhance your tool so we know what tickets will be on the upcoming release before you create the new version? 🙃

Easier than I thought!

1. We know that when a commit goes to master branch, it will be part of the next release
2. In the Github Action we know the target branch
3. We know how to set the version using ClickUp API
4. Let's set the version to "next" on these tickets

scripts/status-update-action.mjs

```
if (isMerged && targetBranch === "master")
    shouldTagForNextRelease = true;
// ...
// Change status for tasks
// ...
if (shouldTagForNextRelease) {
  taskIds.forEach(taskId =>
    setAppVersionToTask({
      taskId,
      appVersion: "next"
    })
  )
}
```

Happy Product

Happy life!

Integrated workflows

- Github actions tag ClickUp tickets and update their statuses

GitHub ⇄ ClickUp

Clear communication

- QA team knows what they are testing
- Product knows what will be included in the upcoming release
- Dev team can use git blame more effectively

Release faster

- Native files are auto-updated
- ClickUp tickets are tagged automatically
- Easier to release leads to more frequent releases

Continuous Improvement

- The journey doesn't end here.
- We've made great strides, but there's always more we can do.
- Continuous imrovement is key to long-term success.

Be the change

- Change starts with you.
- Identify areas for improvement.
- Take action and make a difference.

Thank you!!

nikoskleidis