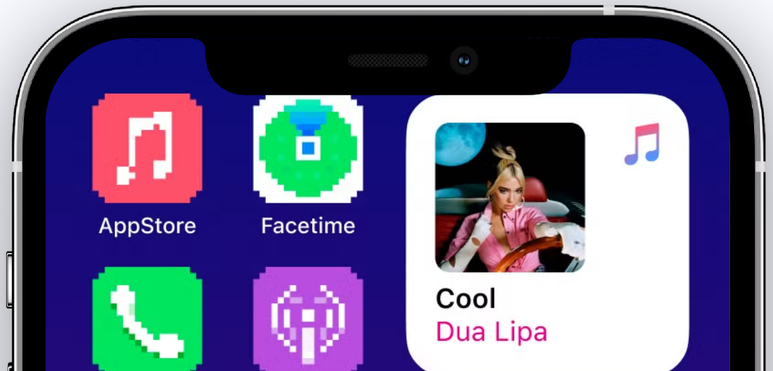
AREEB
TECHNOLOGY

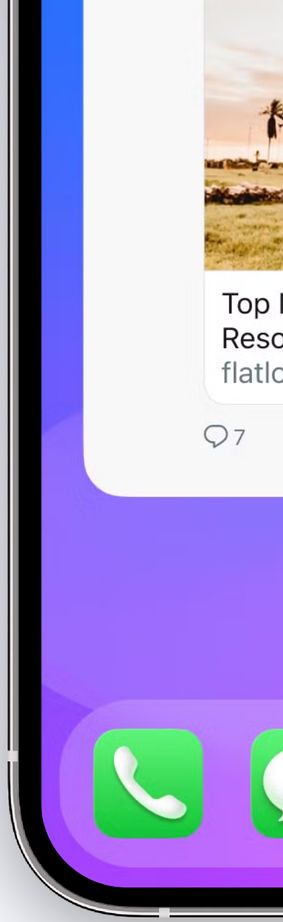
Mobile Releases

Managing The Chaos

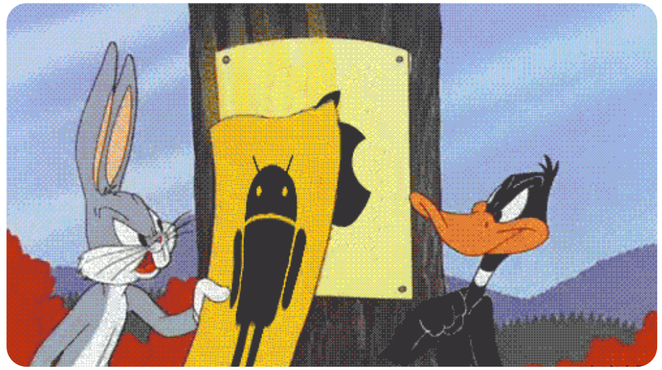
By Diaa Hassan

The chaos of mobile releases



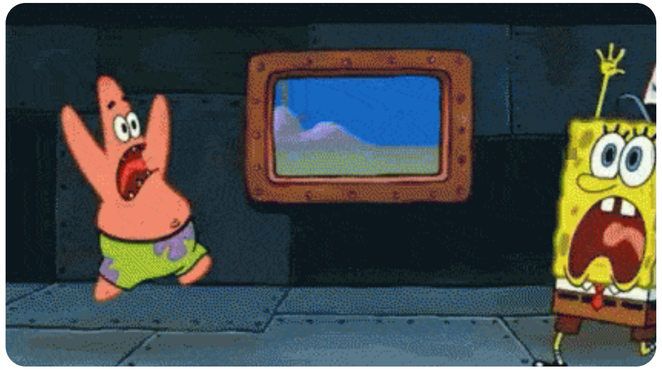


The chaos of mobile releases



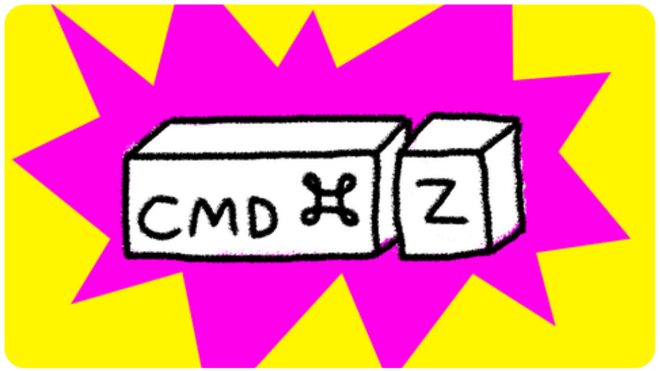
Fragmented Ecosystem

Just ship it !!



Multiple dependencies

Monitoring post-release



Can you roll-back ?!!

Bringing Order To Chaos

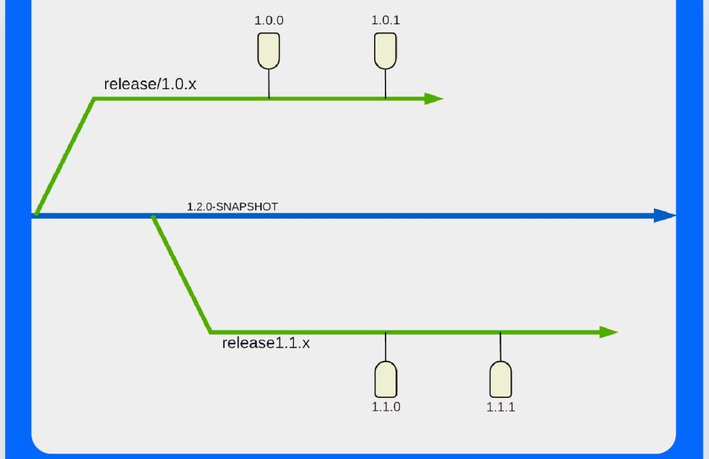
#1

You need a **Release Strategy** yesterday

AND DON'T STICK TO IT

ALWAYS IMPROVE

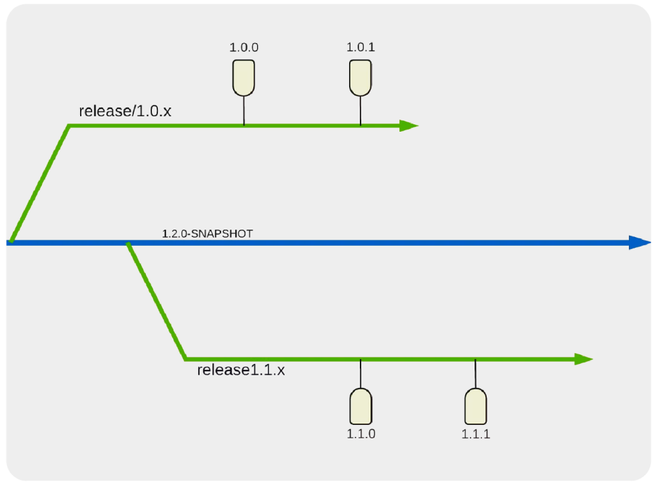
Common release strategies for mobile apps



Feature-based

It’s ready when it’s ready

Common release strategies for mobile apps



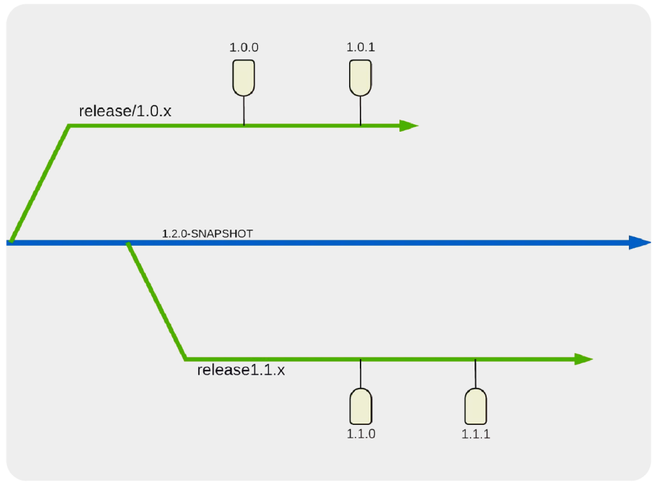
Feature-based

It’s ready when it’s ready

Time-based

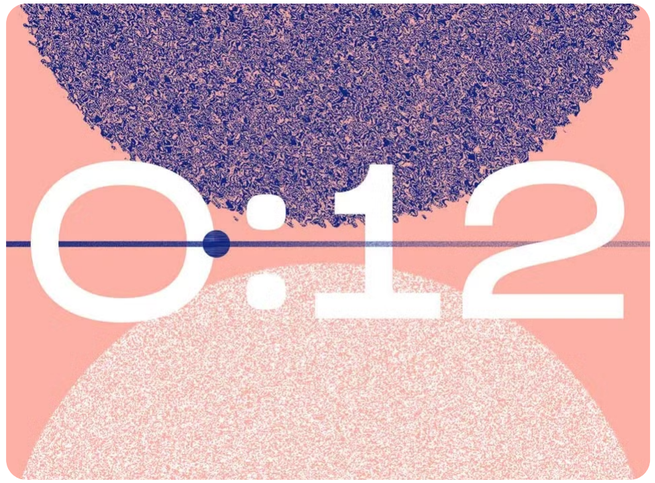
Hitting deadlines

Common release strategies for mobile apps



Feature-based

It’s ready when it’s ready

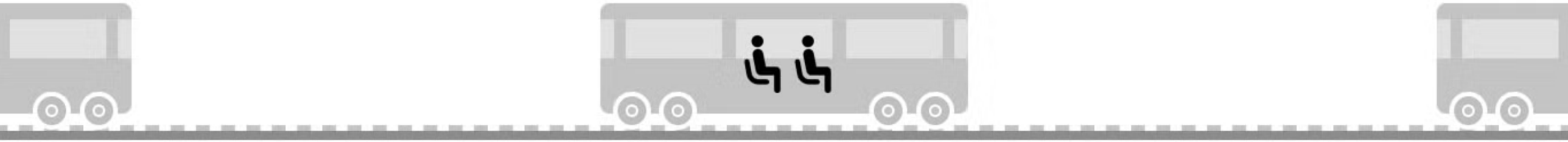


Time-based

Hitting deadlines

Release train

Leaving with or without you



Plan features and bug fixes

Implement features

Feature boarding release train

Feature missed release train

Released to app stores

2

Assign a Release Manager

|

AND ROTATE !!!

3

Have a clear
Documentation
of the process

4

Allow **enough** time for **Testing**

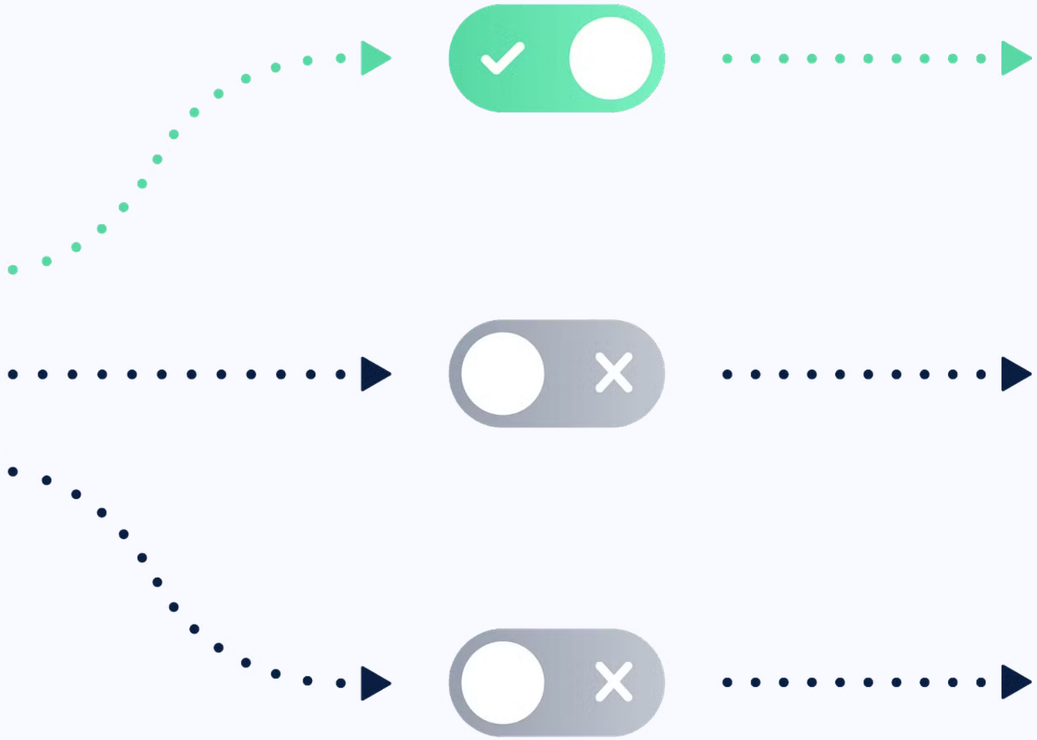
5

Enable **Feature** Flagging

New feature

Feature Flags

Customers



6

Use Phased Releases

Day of Phased Release	Percentage of Users
1	1%
2	2%
3	5%
4	10%
5	20%
6	50%
7	100%

#7

Monitor and React Quickly

Phased release

50%

Adoption rate

33%

Crash-free sessions

98.225%

User ratings

average

3.1

View home page

per session

1.21

View product page

per DAU

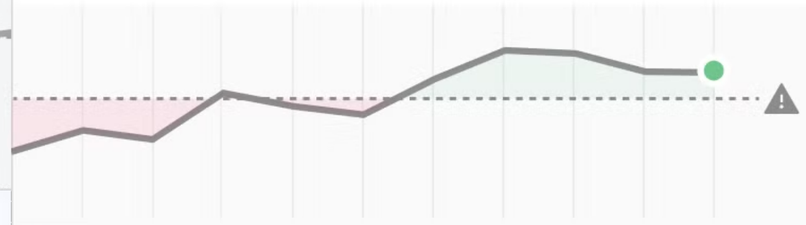
0.84

Crash-free users

96.430%

Checkout

0.06 per daily active user



Cold App Launch timeline

App Launch

	Apdex	P50	Change
Cold App Launch	0.35	5.87 sec	20K
Hot App Launch	0.75	103.181 sec	50K
Warm App Launch	0.75	103.181 sec	40K

Crashes

99.80%

ANRs

99.80%

Cold app launch

0.55 Fair

Network

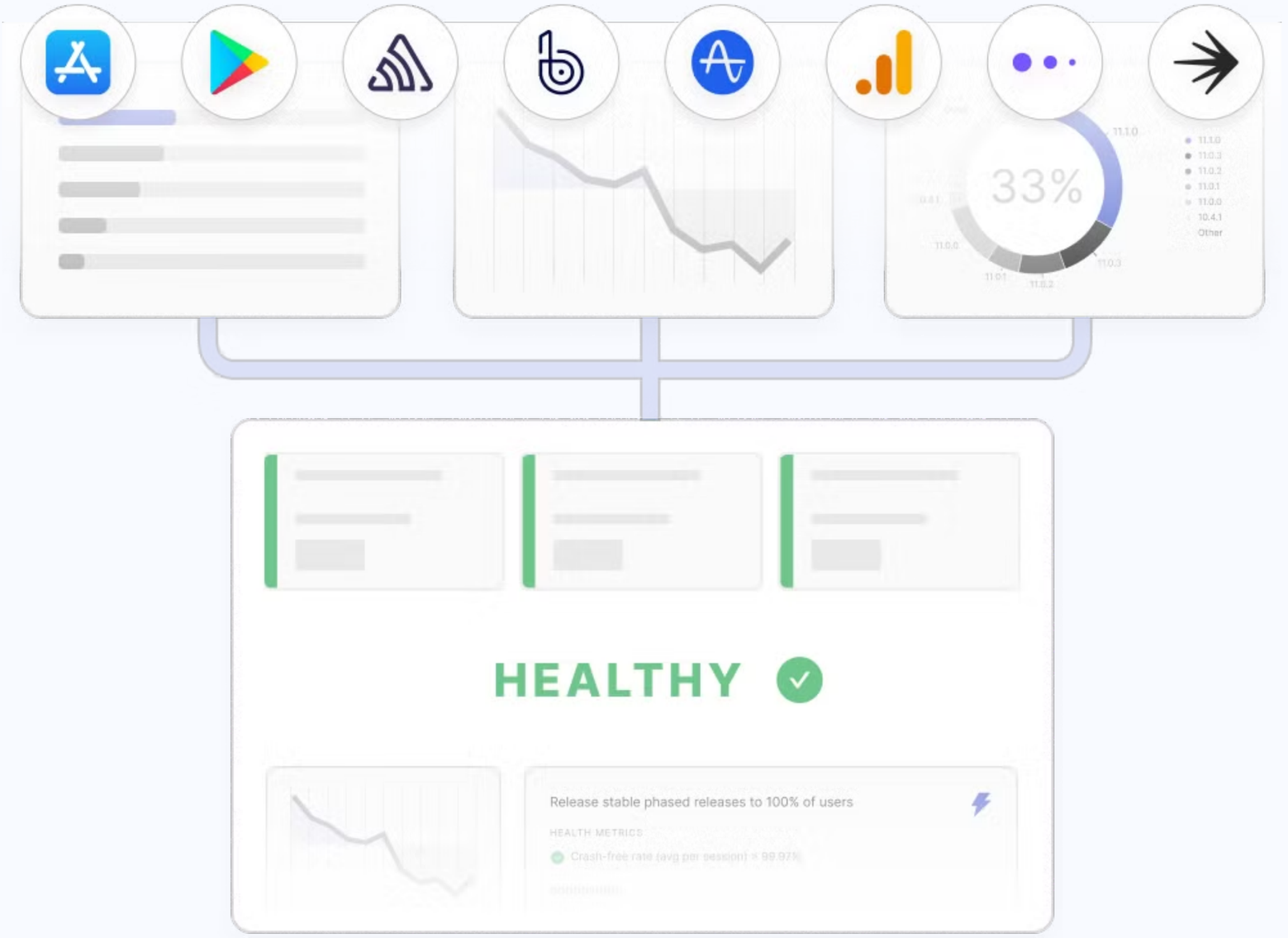
0.98 Excellent

Screen loading

0.94 Good

UI hangs

0.22 Unacceptable


33%
HEALTHY
Release stable phased releases to 100% of users
HEALTH METRICS

8

Have a built-in **Rollback Strategies/Tools**

READY

Prepare rollback releases

STEPS

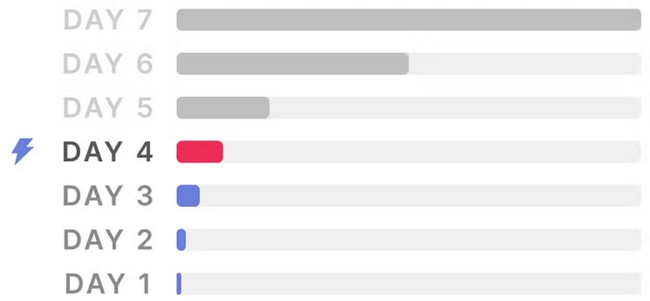
- Re-signing of 11.0.0 build completed
- Submitted
- Approved for release
- Waiting for manual release

- Re-sign build for rollbacks — Completed
- Upload build artifacts for distribution — Completed
- Upload dSYMs to stability monitoring — Completed
- Submit app for review — Completed

Phased release HALTED BY RUNWAY

10% OF USERS

Today is **Day 4**
of a 7-day rollout.



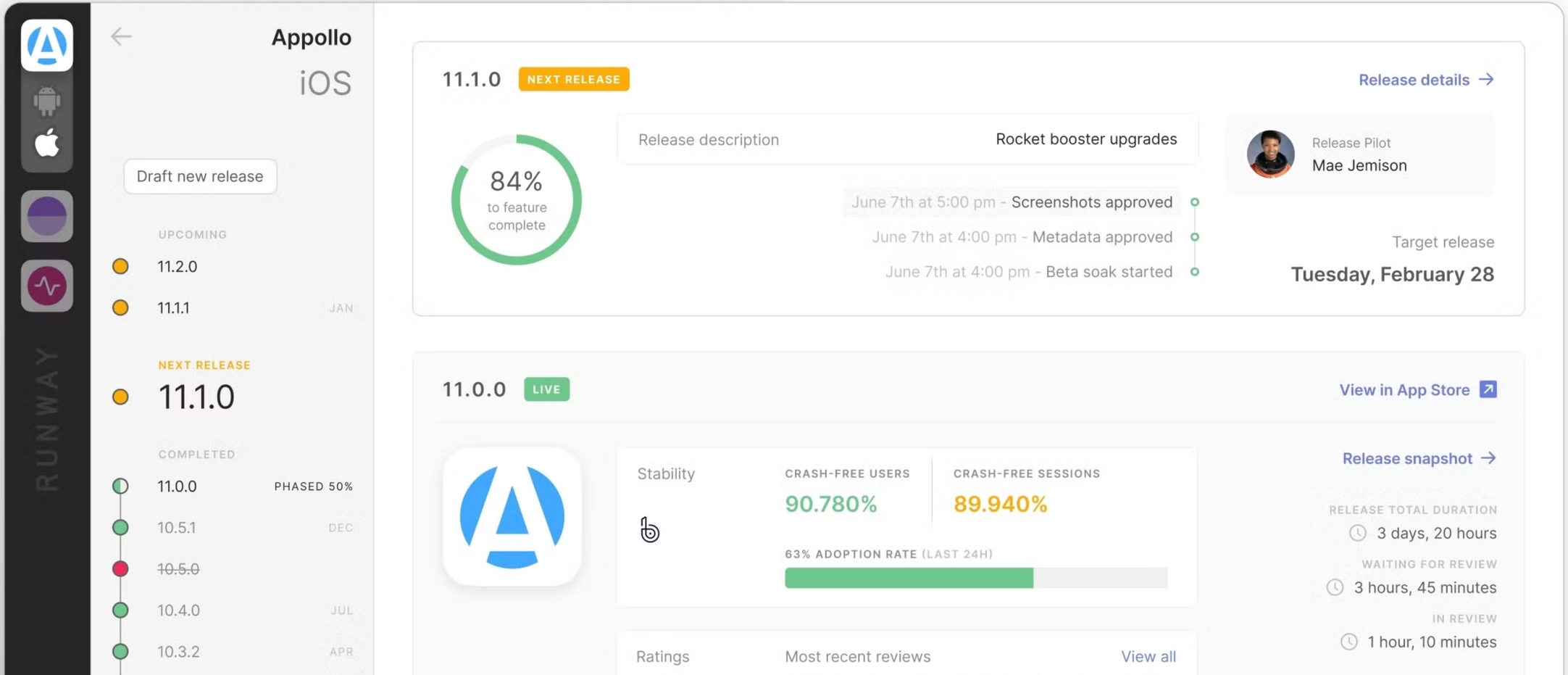
Resume rollout ···

Start a hotfix

Deploy rollback...

RUNWAY

Coordinate and automate your team's mobile app releases.



Automation should be your Best Friend

AUTOMATING THE KICK-OFF OF THE RELEASE

```yaml
name: Kick release off
on:
  schedule:
    - cron: 0 17 * * 0 # every sunday at 5pm UTC

jobs:
  kick-release-off:
    runs-on: ubuntu-latest
    steps:
      - name: Create and push branch release/${{ env.VERSION }}
        run: |
          git config --global user.name "CI/CD Automator"
          git config --global user.email "ci@yourawesomeorg.com"
          git checkout -b release/${{ env.VERSION }}
          git commit -a -m "Updating version to ${{ env.VERSION }}"
          git push --set-upstream origin release/${{ env.VERSION }}
```

AUTOMATING EXPIRED CERTIFICATE RENEWAL

```yaml
name: iOS Certificate Renewal

on:
  schedule:
    - cron: '0 0 24 12 *'

jobs:
  renew-certificates:
    runs-on: macos-latest
    steps:
      - name: Renew certificates
        env:
          MATCH_PASSWORD: ${{ secrets.MATCH_PASSWORD }}
          GIT_AUTHORIZATION: ${{ secrets.GIT_AUTHORIZATION }}
        run: |
          fastlane match nuke development
          fastlane match development --readonly
          fastlane match nuke distribution
          fastlane match appstore --readonly
```

match

AUTOMATING EXPIRED CERTIFICATE RENEWAL

```yaml
name: iOS Certificate Renewal

on:
  schedule:
    - cron: '0 0 24 12 *'  # Run at 00:00 on December 24th (Merry christmas 😈 )

jobs:
  renew-certificates:
    runs-on: macos-latest
    steps:
      - name: Renew certificates
        env:
          MATCH_PASSWORD: ${{ secrets.MATCH_PASSWORD }}
          GIT_AUTHORIZATION: ${{ secrets.GIT_AUTHORIZATION }}
        run: |
          fastlane match nuke development
          fastlane match development --readonly
          fastlane match nuke distribution
          fastlane match appstore --readonly
```

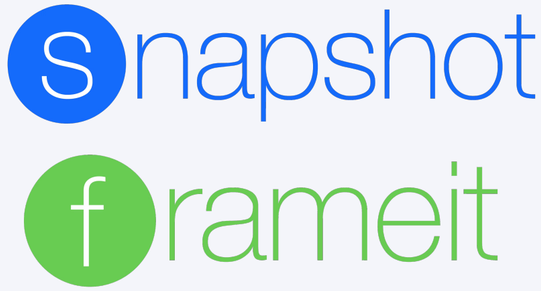
match

AUTOMATING SCREENSHOTS TAKING

```
lane :screenshots do
  snapshot(
    scheme: "YourAppScheme",
    devices: [
      "iPhone 8",
      "iPhone 13 Pro",
      "iPhone 15 Pro Max",
      "iPad Pro (12.9-inch) (5th generation)"
    ],
    languages: ["en-US"],
    output_directory: "./fastlane/screenshots",
    clear_previous_screenshots: true,
    concurrent_simulators: true,
    stop_after_first_error: true
  )
end
```

AUTOMATING SCREENSHOTS TAKING

```
lane :screenshots do
  snapshot(
    scheme: "YourAppScheme",
    devices: [
      "iPhone 8",
      "iPhone 13 Pro",
      "iPhone 15 Pro Max",
      "iPad Pro (12.9-inch) (5th generation)"
    ],
    languages: ["en-US"],
    output_directory: "./fastlane/screenshots",
    clear_previous_screenshots: true,
    concurrent_simulators: true,
    stop_after_first_error: true
  )

  frameit(
    path: "./fastlane/screenshots",
    white: true
  )
end
```



AUTOMATING SCREENSHOTS TAKING

```
lane :screenshots do
  snapshot(
    ....
  )

  frameit(
    ...
  )

  deliver(
    skip_binary_upload: true,
    skip_metadata: true,
    skip_app_version_update: true,
    force: true,
    submission_information: {
      add_id_info_uses_idfa: false
    }
  )
end
```

snapshot

frameit

deliver

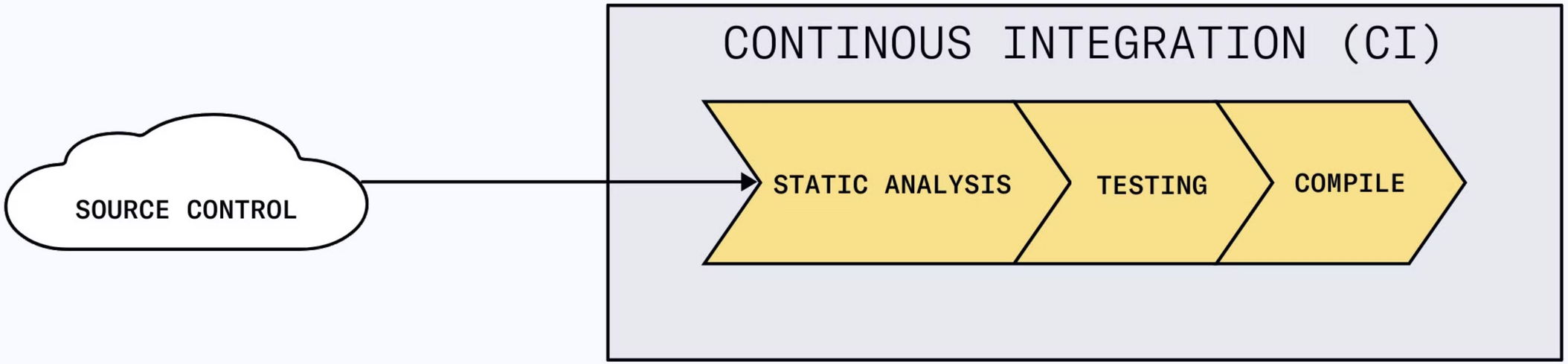
AUTOMATING SCREENSHOTS TAKING

```
lane :screenshots do
  snapshot(
    ....
  )

  frameit(
    ...
  )

  deliver(
    skip_binary_upload: true,
    skip_metadata: true,
    skip_app_version_update: true,
    force: true,
    submission_information: {
      add_id_info_uses_idfa: false
    }
  )
end
```

```
name: iOS Screenshots Renewal

on:
  # for manual trigger
  workflow_dispatch:

jobs:
  screenshots:
    runs-on: macos-latest
    steps:
    - name: Take screenshots and upload
      env:
        ... needed env variables ...
      run: fastlane screenshots
```

How your **CI/CD pipelines** should look like?

The CI pipeline



- Catch issues early
- Automate tests
- Reduce Human Error
- Improve collaboration

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

```
desc "Run Linting"
lane :lint do
  swiftlint(
    mode: :lint,
    output_file: "swiftlint.result.json",
    reporter: "json",
    config_file: ".swiftlint.yml",
    raise_if_swiftlint_error: true,
    ignore_exit_status: false
  )
end
```

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

Security

hard-coded secrets, weak encryption, and insecure network requests.

```
desc "Check any security issues"
lane :security_check do
  # you can use MobSF security scanner
  sh ("security_check_script.sh" )
end
```

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

Security

hard-coded secrets, weak encryption, and insecure network requests.

Dependency Check

unused dependencies, outdated dependencies, and conflicting dependencies.

```
desc "Check for outdated dependencies"
lane :dependency_check do
  cocoapods (
    repo_update: true
  )
  sh ("pod outdated" )
end
```

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

Security

hard-coded secrets, weak encryption, and insecure network requests.

Dependency Check

unused dependencies, outdated dependencies, and conflicting dependencies.

Accessibility Check

missing labels, low contrast, and absent accessibility traits.

```
lane :accessibility_check do
  // You can use Xcode's Accessibility Inspector
  sh("accessibility_check_script.sh")
end
```

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

Security

hard-coded secrets, weak encryption, and insecure network requests.

Dependency Check

unused dependencies, outdated dependencies, and conflicting dependencies.

Accessibility Check

missing labels, low contrast, and absent accessibility traits.

```
desc "Run all static analysis"
lane :static_analysis do
  lint
  security_check
  dependency_check
  accessibility_check
end
```

Static analysis

Linting (code style)

missing semicolons, incorrect indentation, and inconsistent naming conventions.

Security

hard-coded secrets, weak encryption, and insecure network requests.

Dependency Check

unused dependencies, outdated dependencies, and conflicting dependencies.

Accessibility Check

missing labels, low contrast, and absent accessibility traits.

```yaml
name: iOS Static Analysis - CI

on:
  pull_request:
    branches: [ develop ]

jobs:
  build:
    runs-on: macos-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up Ruby
      uses: ruby/setup-ruby@v1

    - name: Install Fastlane
      run: gem install fastlane

    - name: Install dependencies
      run: bundle install

    - name: Build iOS app
      run: fastlane static_analysis
```

Testing

Unit tests
ensure that individual components of your application function as intended. By validating each component in isolation, it is easier to catch regressions early in the development cycle. ***XCTest*** *for iOS and* ***JUnit*** *and* ***Mockito*** *for Android*

```
desc "Running unit tests"
lane :tests do
  spm(
    command: "generate-xcodeproj"
  )
  run_tests(
    project: "./<NameOfYourPackage>.xcodeproj",
    devices: ["iPhone 14 Pro Max"]
  )
  sh("rm", "-rf", "<NameOfYourPackage>.xcodeproj")
end
```

Testing

Unit tests

ensure that individual components of your application function as intended. By validating each component in isolation, it is easier to catch regressions early in the development cycle. *XCTest for iOS and JUnit and Mockito for Android*

Integration tests

confirm that different application components work together correctly. ***Espresso** for Android and **XCUITest** for iOS*

```
desc "Running unit tests"
lane :tests do
  spm(
    command: "generate-xcodeproj"
  )
  run_tests(
    project: "./<NameOfYourPackage>.xcodeproj",
    devices: ["iPhone 14 Pro Max"]
  )
  sh("rm", "-rf", "<NameOfYourPackage>.xcodeproj")
end
```

Testing

Unit tests

ensure that individual components of your application function as intended. By validating each component in isolation, it is easier to catch regressions early in the development cycle. *XCTest for iOS and JUnit and Mockito for Android*

Integration tests

confirm that different application components work together correctly. *Espresso for Android and XCUITest for iOS*

End-to-end tests

essential for simulating real-world user interactions with your app. They cover the entire application flow, enabling developers to detect issues from the user's perspective. **Appium** and **Detox**

```
desc "Running e2e tests"
lane :e2e_tests do
  appium(
    app_path:  "appium/apps/TargetApp.app",
    spec_path: "appium/spec",
    platform:  "iOS",
    caps: {
      versionNumber: "9.1",
      deviceName:    "iPhone 6"
    },
    appium_lib: {
      wait: 10
    }
  )
end
```

Compiling

The lane should look something like this:

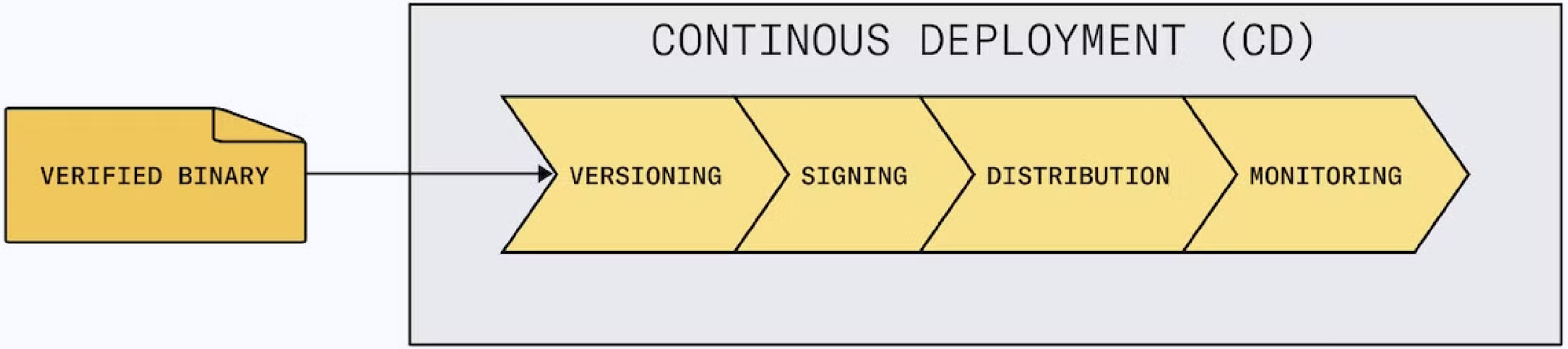
```
desc "Compile and build the app"
lane :build do
  gym(
    scheme: "YourSchemeName",
    clean: true,
    output_directory: "./build",
    output_name: "YourAppName.ipa"
  )
end
```

To execute it run:

```
> fastlane build
```

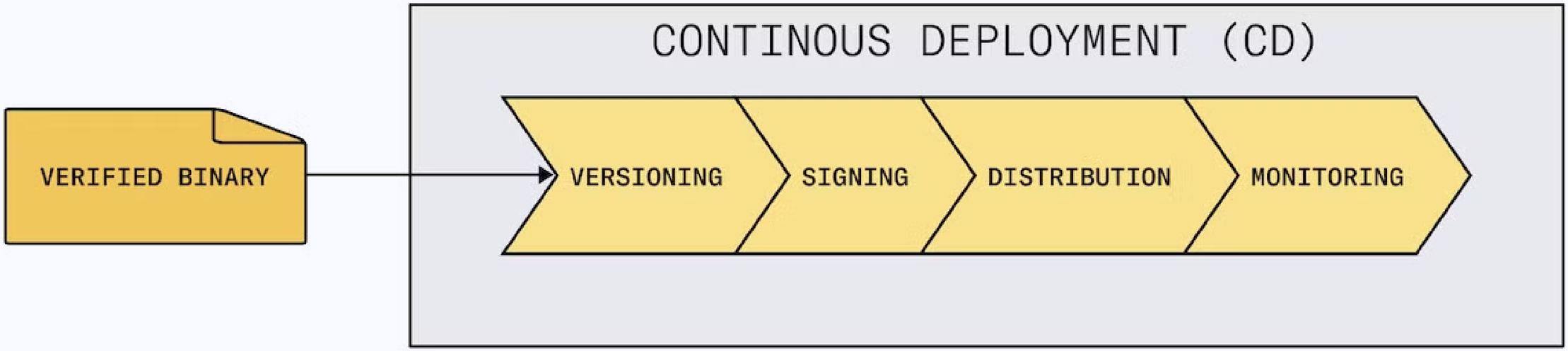
```
name: iOS Build - CI

on:
  pull_request:
    branches: [ develop ]

jobs:
  build:
    runs-on: macos-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up Ruby
      uses: ruby/setup-ruby@v1

    - name: Install Fastlane
      run: gem install fastlane

    - name: Install dependencies
      run: bundle install

    - name: Build iOS app
      run: fastlane build
```

The CD pipeline




```
desc "Increment the app version patch"
lane :bumpPatch do
  increment_version_number(
    bump_type: "patch"
  )
end
```

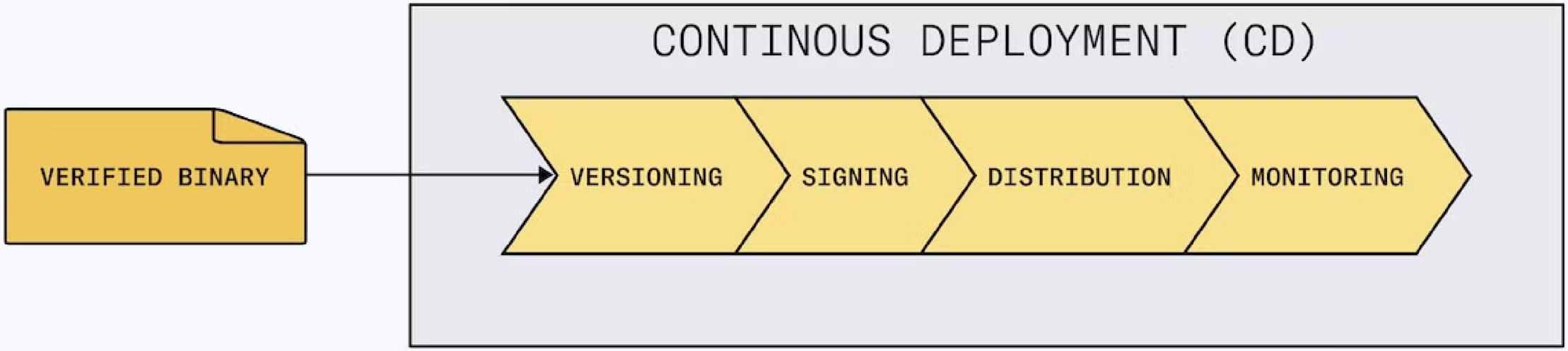
The CD pipeline




```
desc "Increment the app version patch"
lane :bumpPatch do
  increment_version_number(
    bump_type: "patch"
  )
end
```

```
desc "Increment the app version minor"
lane :bumpMinor do
  increment_version_number(
    bump_type: "minor"
  )
end
```

The CD pipeline



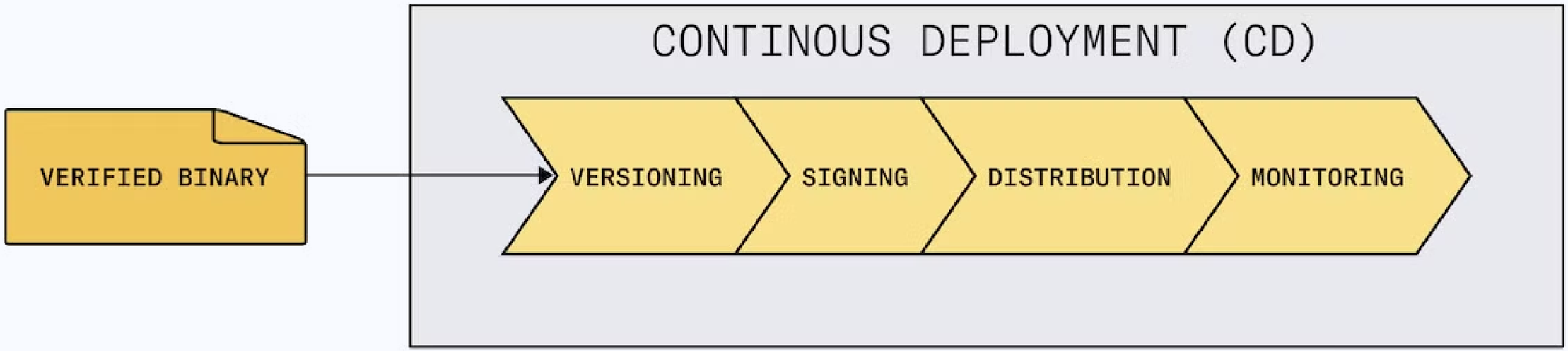

```
desc "Increment the app version patch"
lane :bumpPatch do
  increment_version_number(
    bump_type: "patch"
  )
end
```

```
desc "Increment the app version major"
lane :bumpMajor do
  increment_version_number(
    bump_type: "major"
  )
end
```

```
esc "Increment the app version minor"
ane :bumpMinor do
 increment_version_number(
   bump_type: "minor"
 )
nd
```

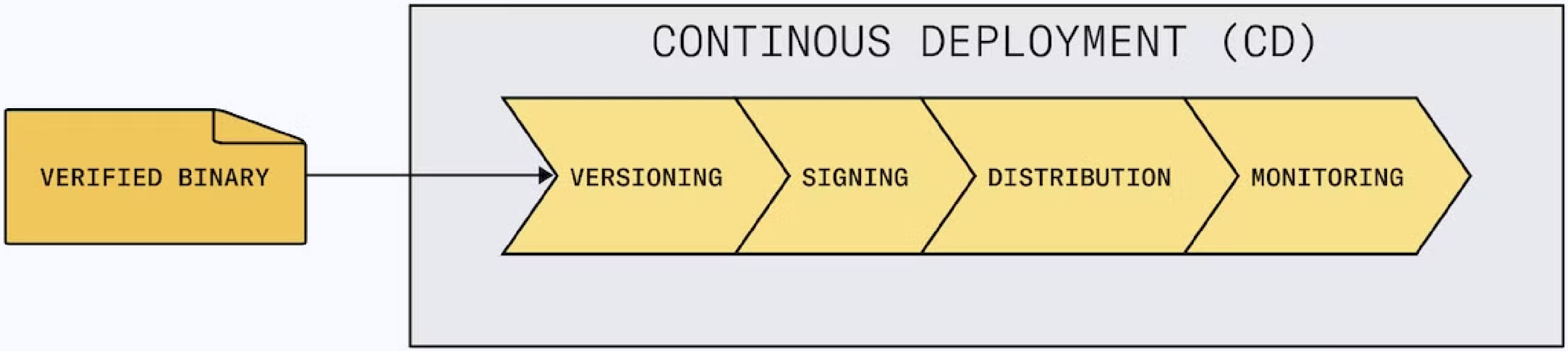
Try Pitch

The CD pipeline



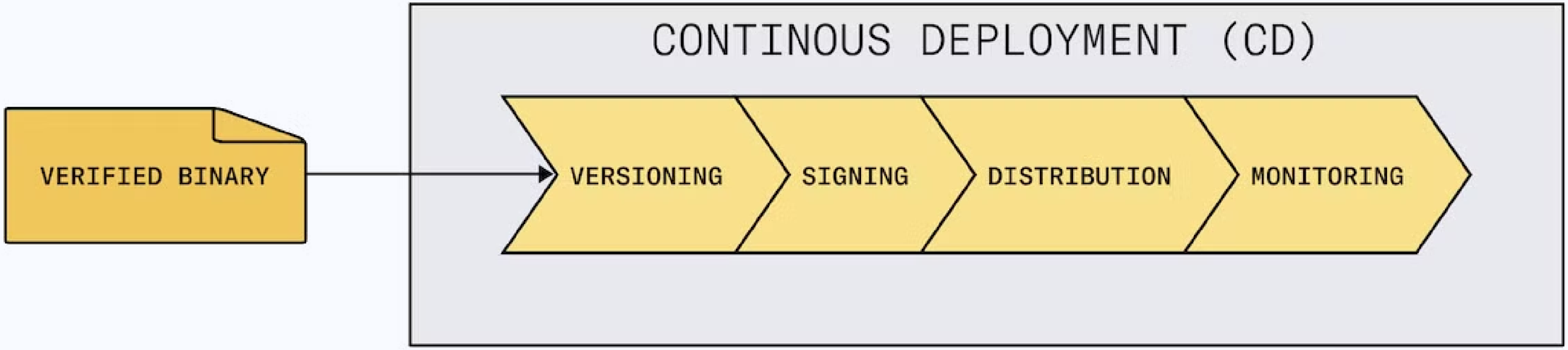
match

The CD pipeline



- Beta
- Ad-Hoc
- Production

The CD pipeline



THANK YOU

And Happy Releases 🎉

@AREEB TECHNOLOGY

@DIAA HASSAN

@SWIFT CAIRO

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)