skyscanner

Release iOS apps

#1 – to millions of users
#2 – every second week
#3 – with confident

Csaba Szabo
Senior Test Engineer

Skyscanner iOS app – our user base

Size

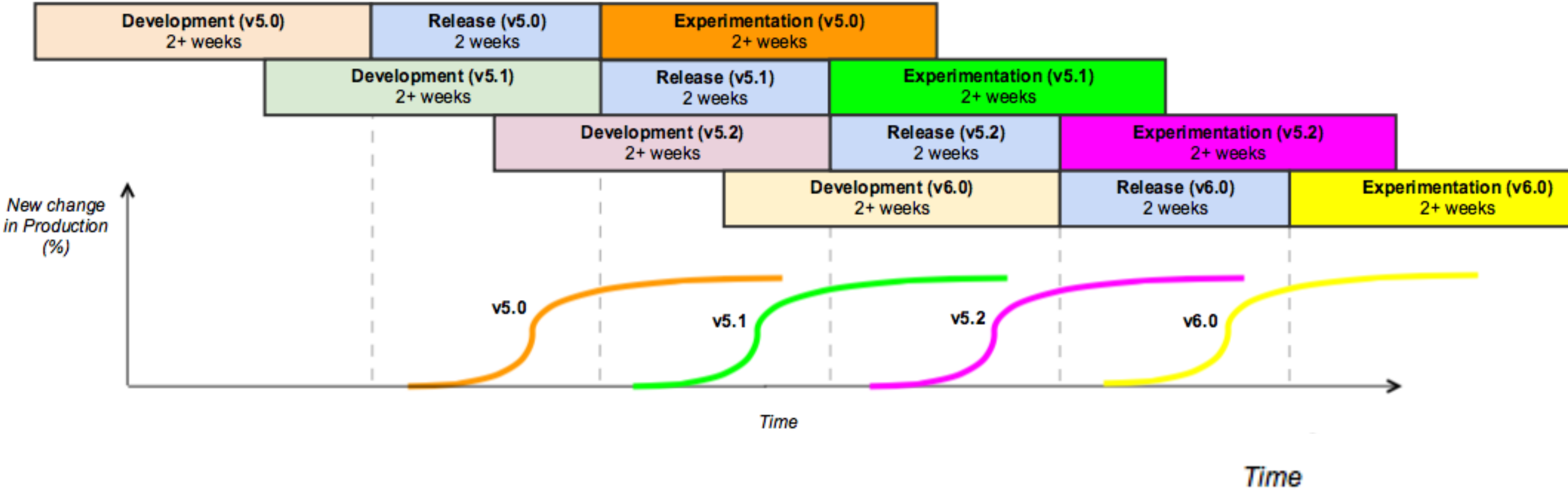
Budapest < monthly users < Hungary

Diversity

World-wide, all time zones
Over 30 languages

Setting the pace

”Release changes to production every second week”



Development (v5.0)
2+ weeks

Release (v5.0)
2 weeks

Experimentation (v5.0)
2+ weeks

Development (v5.1)
2+ weeks

Release (v5.1)
2 weeks

Experimentation (v5.1)
2+ weeks

Development (v5.2)
2+ weeks

Release (v5.2)
2 weeks

Experimentation (v5.2)
2+ weeks

Development (v6.0)
2+ weeks

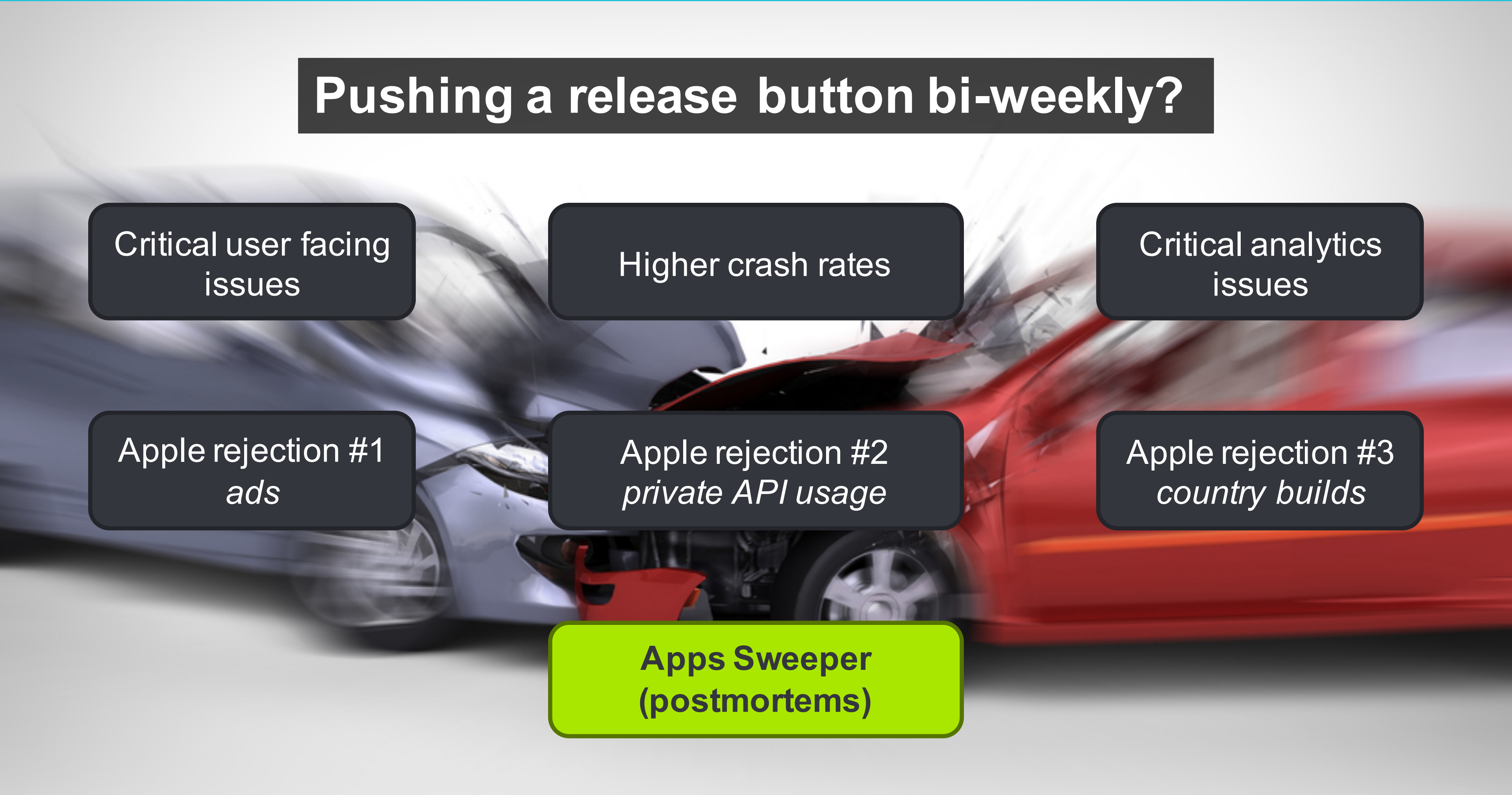
Release (v6.0)
2 weeks

Experimentation (v6.0)
2+ weeks

Pushing a release button bi-weekly?

Like driving a fast car without seatbelt

Pushing a release button bi-weekly?



Critical user facing issues

Higher crash rates

Critical analytics issues

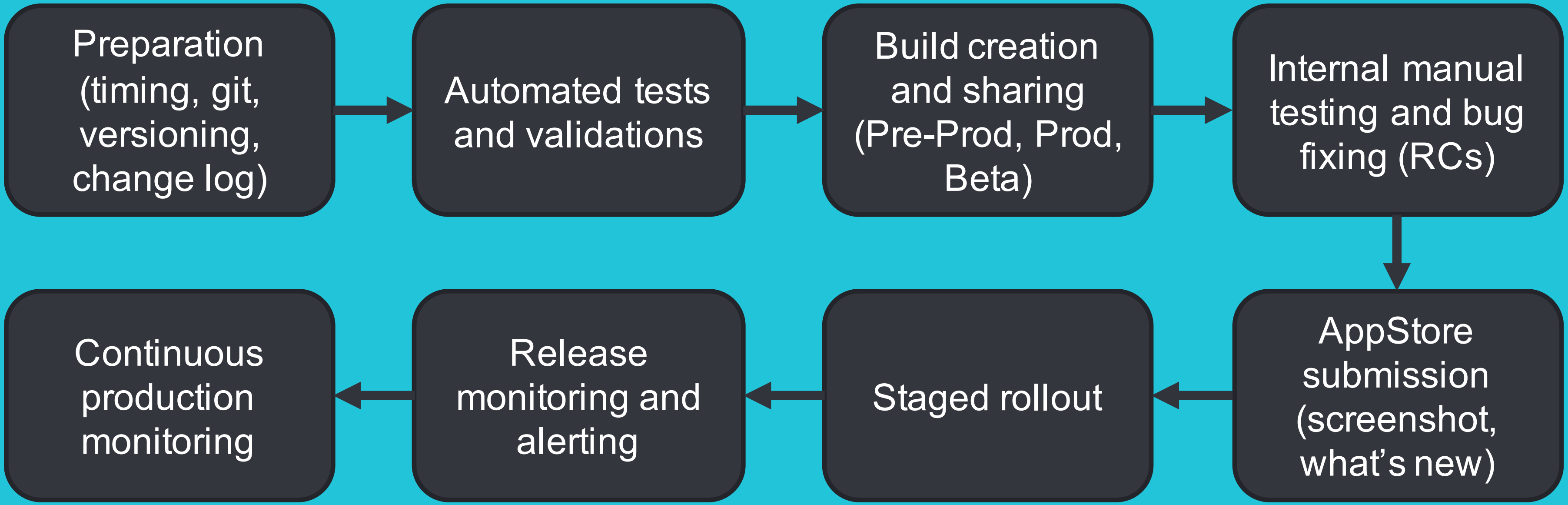
Apple rejection #1
ads

Apple rejection #2
private API usage

Apple rejection #3
country builds

Apps Sweeper (postmortems)

How one release looks like



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

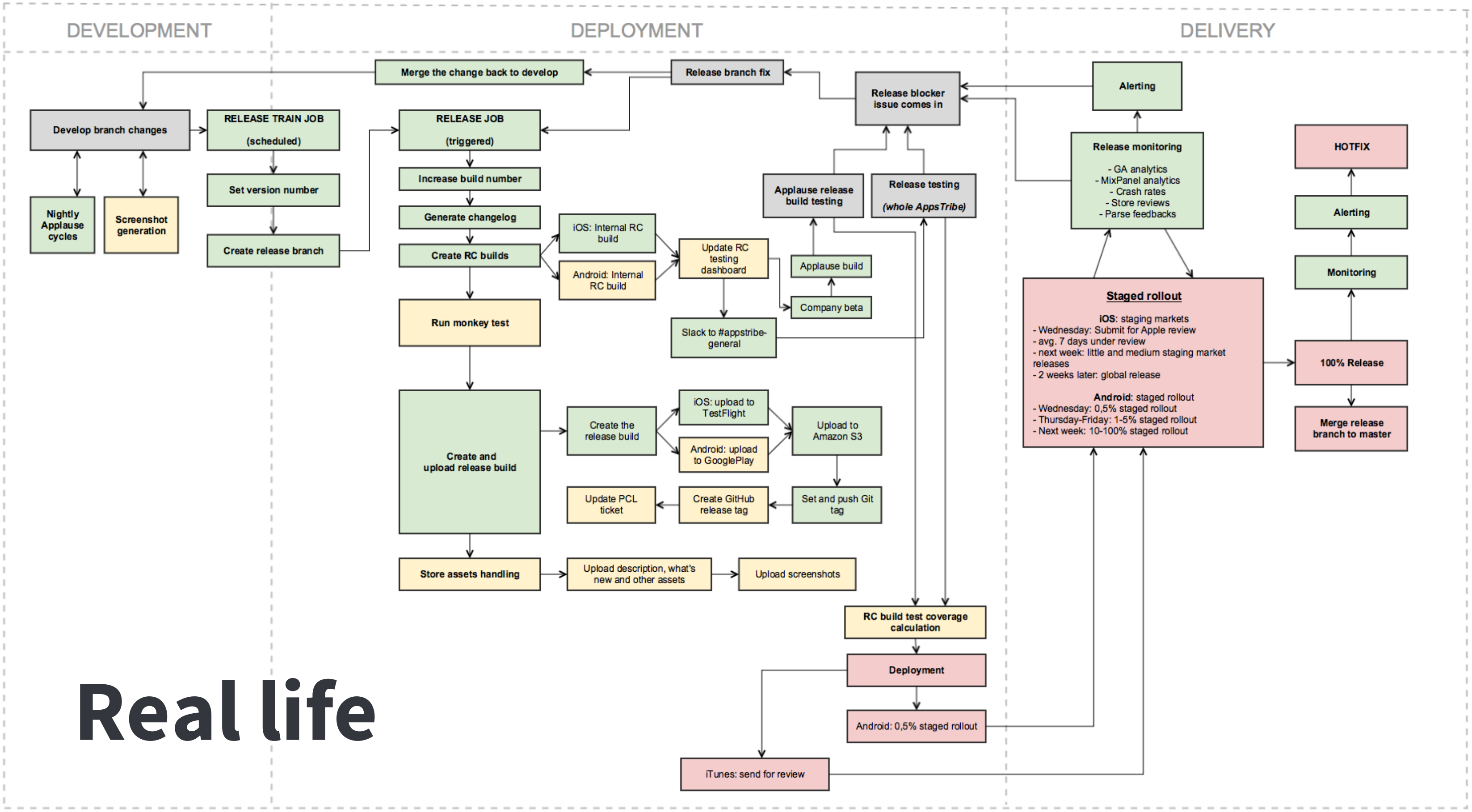
Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what's new)

Staged rollout

Release monitoring and alerting

Continuous production monitoring



DEVELOPMENT

DEPLOYMENT

DELIVERY

Merge the change back to develop

Release branch fix

Release blocker issue comes in

Alerting

Develop branch changes

RELEASE TRAIN JOB (scheduled)

RELEASE JOB (triggered)

Nightly Applause cycles

Screenshot generation

Set version number

Create release branch

Increase build number

Generate changelog

Create RC builds

iOS: Internal RC build

Android: Internal RC build

Update RC testing dashboard

Slack to #appstribe-general

Applause release build testing

Release testing *(whole AppsTribe)*

Applause build

Company beta

Release monitoring

- GA analytics
- MixPanel analytics
- Crash rates
- Store reviews
- Parse feedbacks

HOTFIX

Alerting

Monitoring

100% Release

Merge release branch to master

Run monkey test

Create and upload release build

Create the release build

iOS: upload to TestFlight

Android: upload to GooglePlay

Upload to Amazon S3

Set and push Git tag

Create GitHub release tag

Update PCL ticket

Store assets handling

Upload description, what's new and other assets

Upload screenshots

RC build test coverage calculation

Deployment

Android: 0,5% staged rollout

iTunes: send for review

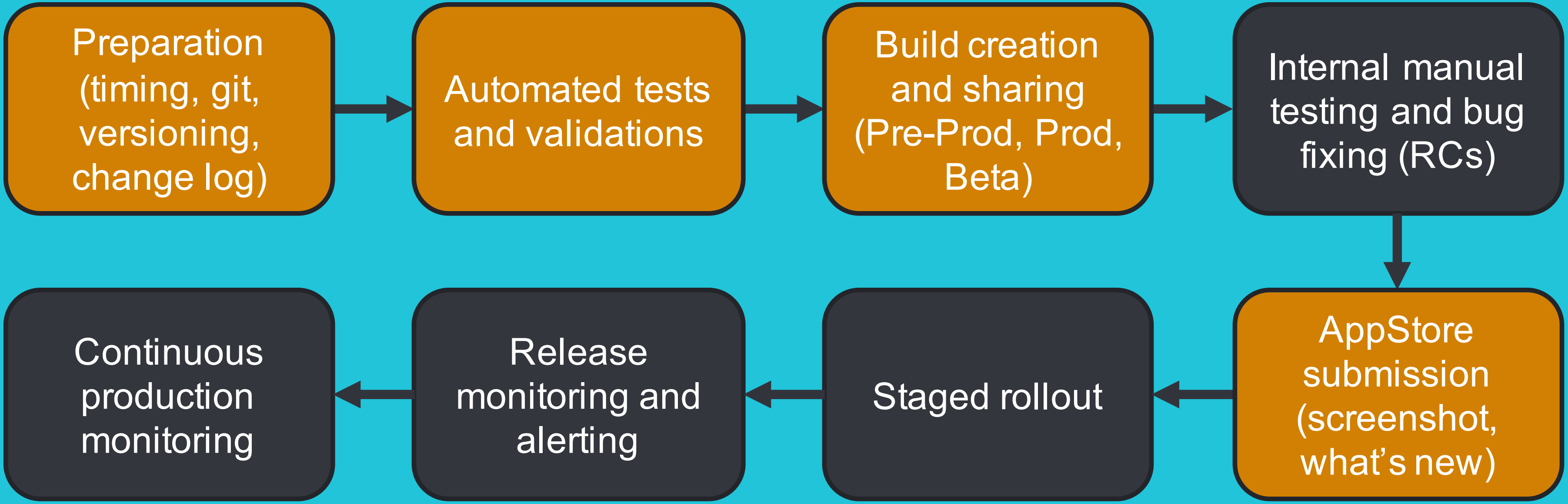
Staged rollout

iOS: staging markets
- Wednesday: Submit for Apple review
- avg. 7 days under review
- next week: little and medium staging market releases
- 2 weeks later: global release

Android: staged rollout
- Wednesday: 0,5% staged rollout
- Thursday-Friday: 1-5% staged rollout
- Next week: 10-100% staged rollout

Real life

#1 - Release process automation



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

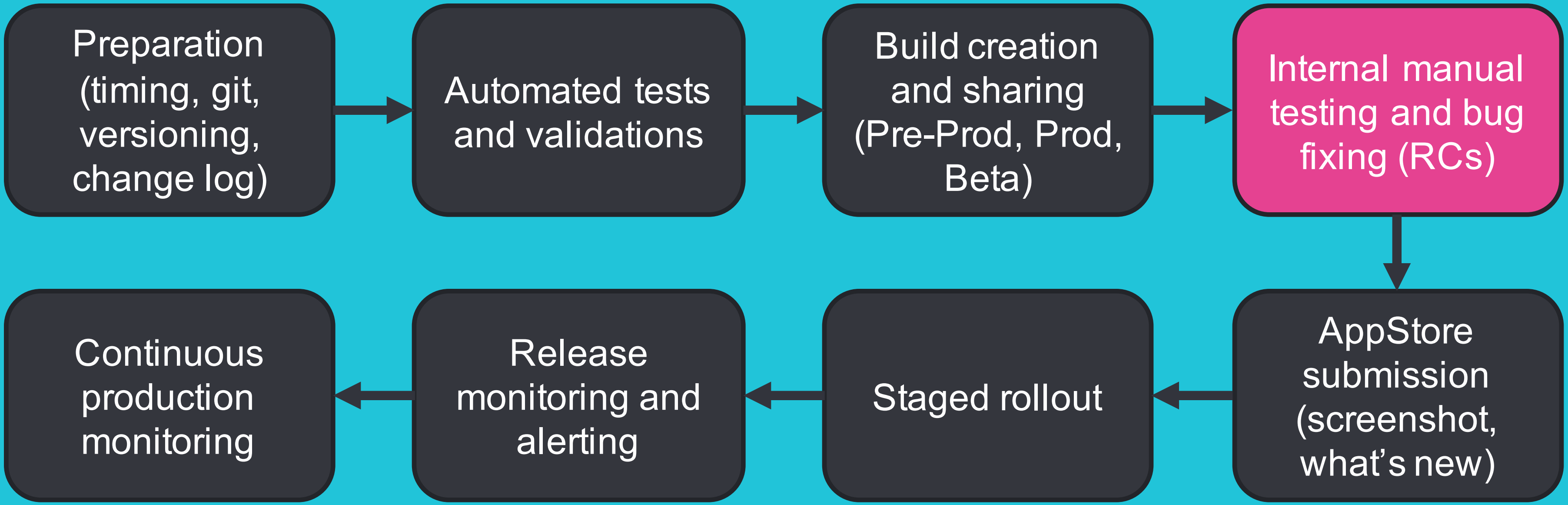
AppStore submission (screenshot, what’s new)

Staged rollout

Release monitoring and alerting

Continuous production monitoring

#2 - Stability period



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

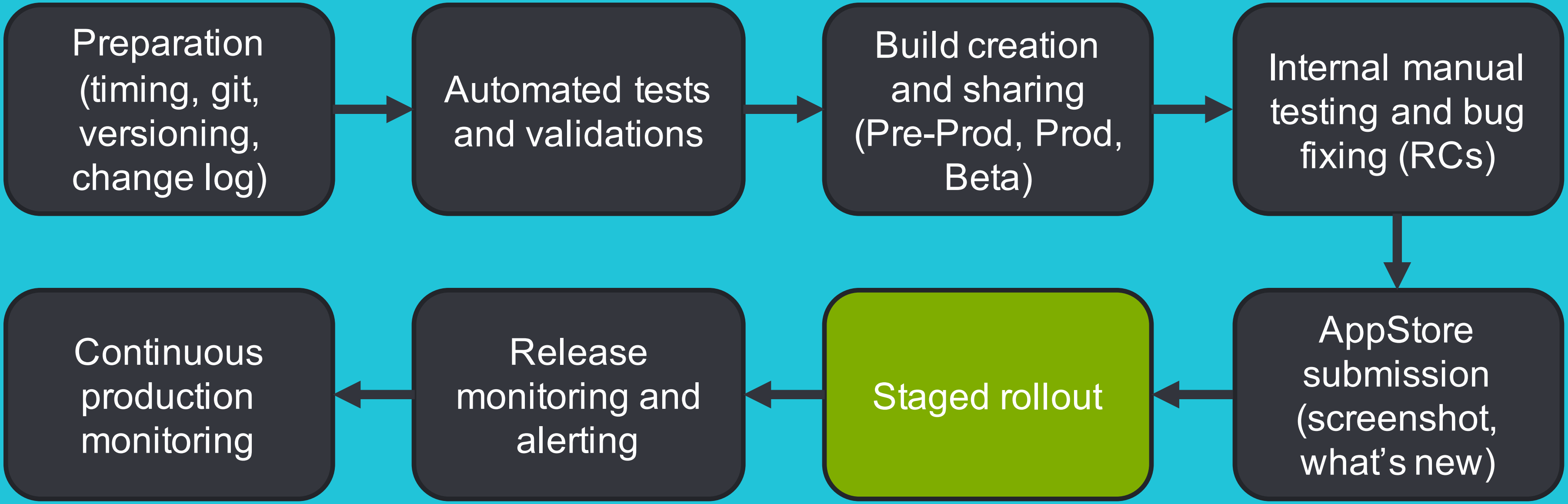
AppStore submission (screenshot, what's new)

Staged rollout

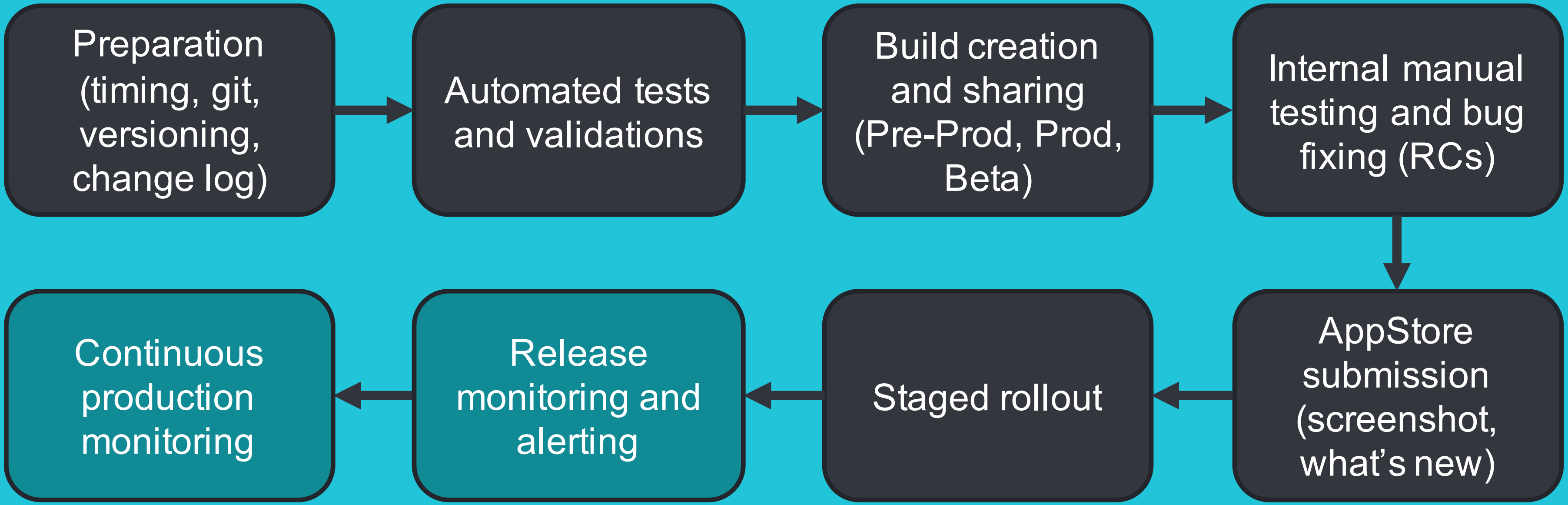
Release monitoring and alerting

Continuous production monitoring

#3 - Staged rollout


Preparation (timing, git, versioning, change log)
Automated tests and validations
Build creation and sharing (Pre-Prod, Prod, Beta)
Internal manual testing and bug fixing (RCs)
AppStore submission (screenshot, what’s new)
Staged rollout
Release monitoring and alerting
Continuous production monitoring

#4 - Monitoring and alerting



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

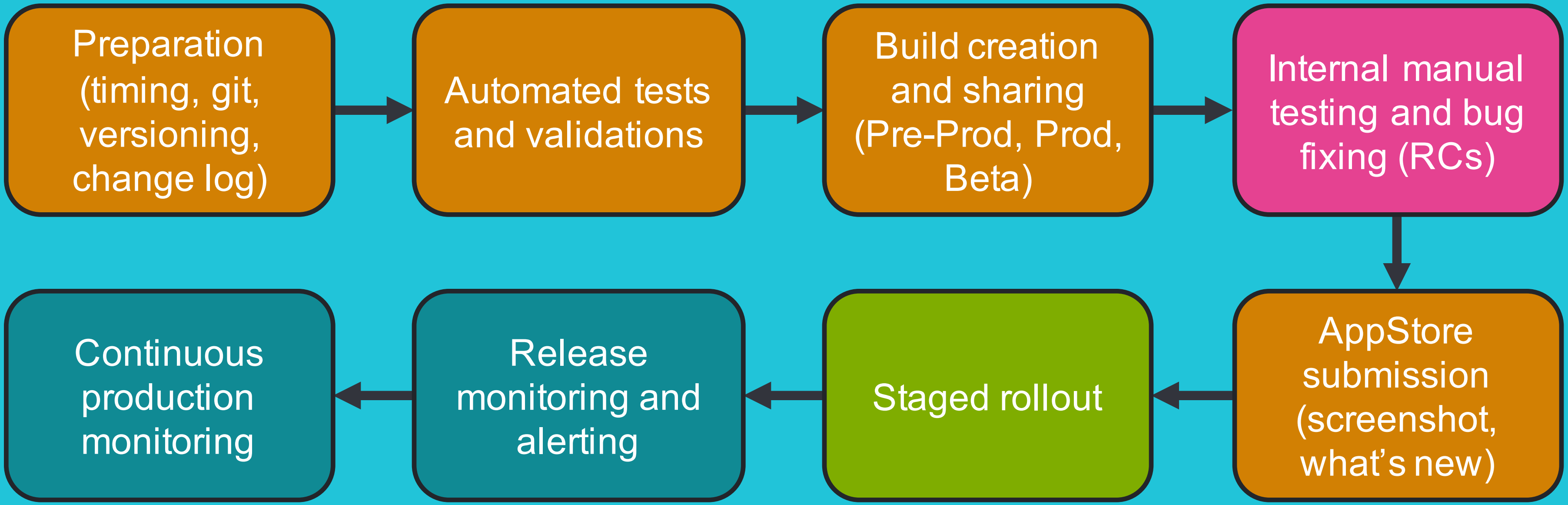
AppStore submission (screenshot, what's new)

Staged rollout

Release monitoring and alerting

Continuous production monitoring

Let’s see them



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

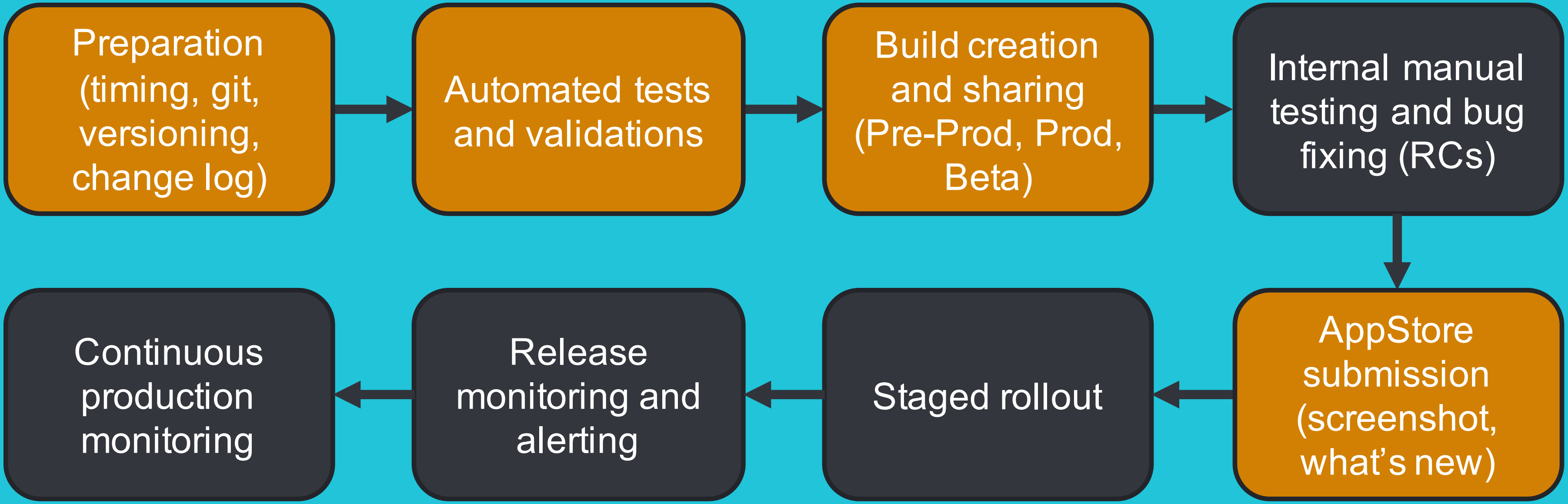
AppStore submission (screenshot, what’s new)

Staged rollout

Release monitoring and alerting

Continuous production monitoring

#1 - Release process automation



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what’s new)

Staged rollout

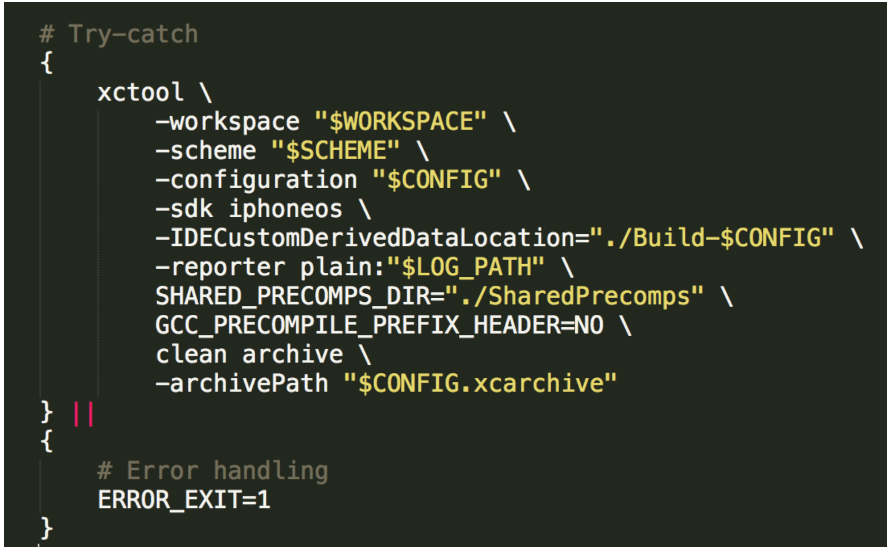
Release monitoring and alerting

Continuous production monitoring

How to automate the whole release process?

Tooling and environment

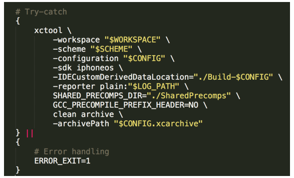
Build scripts



```
# Try-catch
{
    xctool \
        -workspace "$WORKSPACE" \
        -scheme "$SCHEME" \
        -configuration "$CONFIG" \
        -sdk iphoneos \
        -IDECustomDerivedDataLocation="./Build-$CONFIG" \
        -reporter plain:"$LOG_PATH" \
        SHARED_PRECOMPS_DIR="./SharedPrecomps" \
        GCC_PRECOMPILE_PREFIX_HEADER=NO \
        clean archive \
        -archivePath "$CONFIG.xcarchive"
} ||
{
    # Error handling
    ERROR_EXIT=1
}
```

How to automate the whole release process?

- **Tooling and environment**
  - CI system (Jenkins + custom tool)
  - CI slave environment difficulties: gitcache, pod cache, certificates, maintenance
- **Build scripts**
  - Custom script -> Fastlane
  - git (branching, tagging, merging back changes from release to develop)
  - xcodebuild (xctool, gym)
  - iTunesConnect (Spaceship, deliver, pilot, …)
  - Various other scripts (dashboard, hockeyapp, slack, analytics, ...)



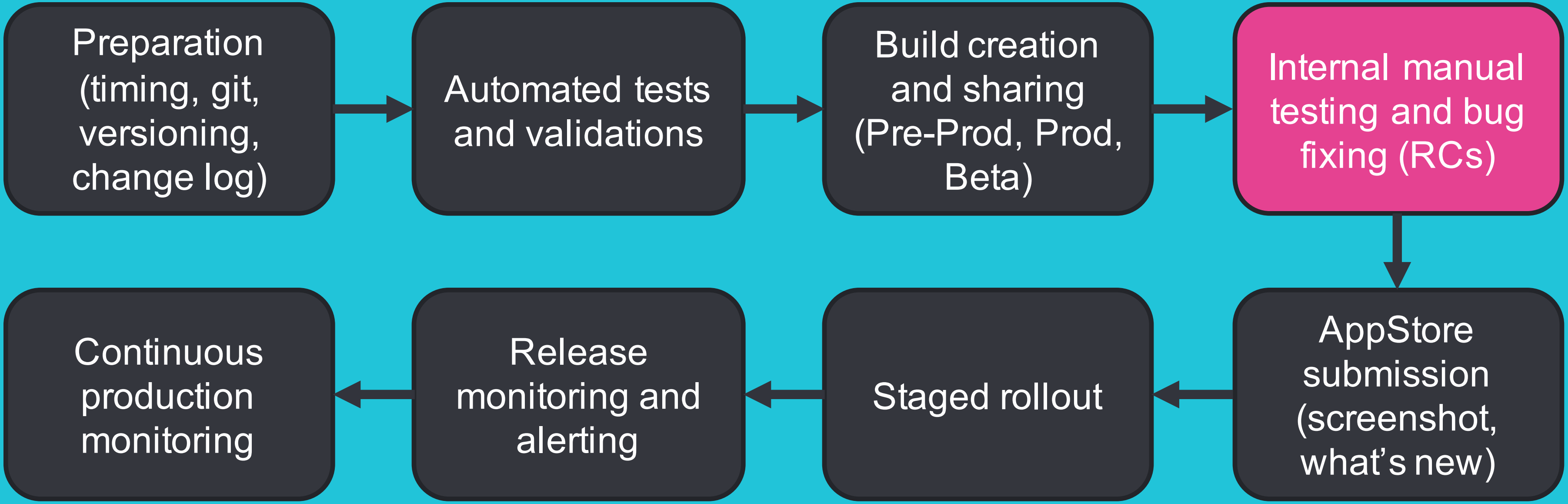
```
# Try-catch
{
    xctool \
        -workspace "$WORKSPACE" \
        -scheme "$SCHEME" \
        -configuration "$CONFIG" \
        -sdk iphoneos \
        -IDECustomDerivedDataLocation="./Build-$CONFIG" \
        -reporter plain:"$LOG_PATH" \
        SHARED_PRECOMPS_DIR="./SharedPrecomps" \
        GCC_PRECOMPILE_PREFIX_HEADER=NO \
        clean archive \
        -archivePath "$CONFIG.xcarchive"
} ||
{
    # Error handling
    ERROR_EXIT=1
}
```

iTunes Connect – good and bad side

- **Good side**
  - TestFlight (*not for our size*)
  - App Analytics (*not for our size*)
- **Bad side**
  - No staged rollout
  - Store review process (~a week)
  - Limited hotfix options
  - No rollback option
  - Cannot update screenshots under review
  - Unstable API
  - Still doesn't support many localizations (e.g. HU)

skyscanner

#2 - Stability period


Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what’s new)

Staged rollout

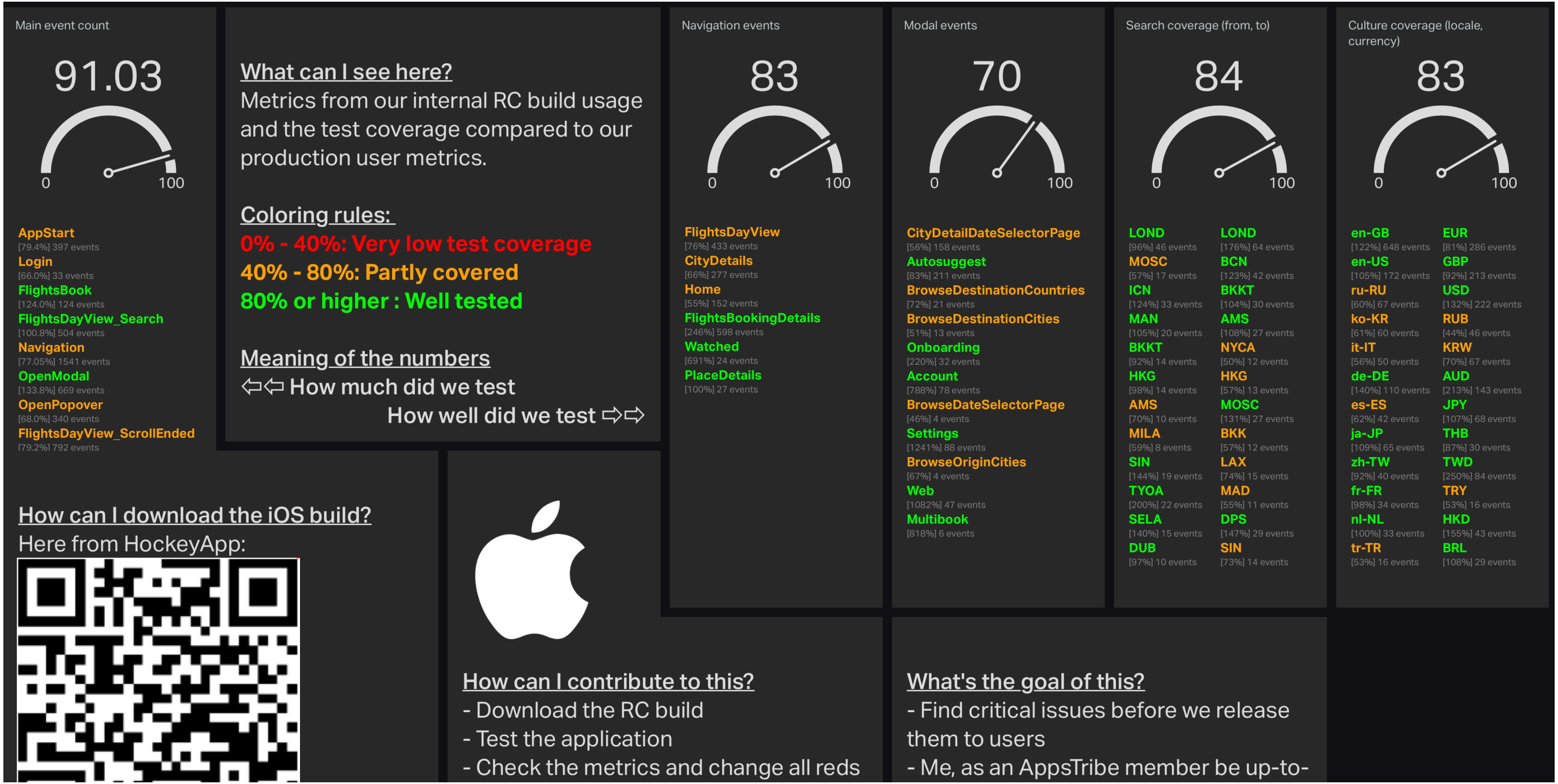
Release monitoring and alerting

Continuous production monitoring

Stability period

Internal testing
”dogfooding”

Crowdsourced testing
with Applause



Main event count

91.03

AppStart
[79.4%] 397 events
Login
[66.0%] 33 events
FlightsBook
[124.0%] 124 events
FlightsDayView_Search
[100.8%] 504 events
Navigation
[77.05%] 1541 events
OpenModal
[133.8%] 669 events
OpenPopover
[68.0%] 340 events
FlightsDayView_ScrollEnded
[79.2%] 792 events

What can I see here?

Metrics from our internal RC build usage and the test coverage compared to our production user metrics.

Coloring rules:

0% - 40%: Very low test coverage

40% - 80%: Partly covered

80% or higher : Well tested

Meaning of the numbers

⇦⇦ How much did we test

How well did we test ⇨⇨

Navigation events

83

FlightsDayView
[76%] 433 events
CityDetails
[66%] 277 events
Home
[55%] 152 events
FlightsBookingDetails
[246%] 598 events
Watched
[691%] 24 events
PlaceDetails
[100%] 27 events

Modal events

70

CityDetailDateSelectorPage
[56%] 158 events
Autosuggest
[83%] 211 events
BrowseDestinationCountries
[72%] 21 events
BrowseDestinationCities
[51%] 13 events
Onboarding
[220%] 32 events
Account
[788%] 78 events
BrowseDateSelectorPage
[46%] 4 events
Settings
[1241%] 88 events
BrowseOriginCities
[67%] 4 events
Web
[1082%] 47 events
Multibook
[818%] 6 events

Search coverage (from, to)

84

LOND
[96%] 46 events
MOSC
[57%] 17 events
ICN
[124%] 33 events
MAN
[105%] 20 events
BKKT
[92%] 14 events
HKG
[98%] 14 events
AMS
[70%] 10 events
MILA
[59%] 8 events
SIN
[144%] 19 events
TYOA
[200%] 22 events
SELA
[140%] 15 events
DUB
[97%] 10 events

LOND
[176%] 64 events
BCN
[123%] 42 events
BKKT
[104%] 30 events
AMS
[108%] 27 events
NYCA
[50%] 12 events
HKG
[57%] 13 events
MOSC
[131%] 27 events
BKK
[57%] 12 events
LAX
[74%] 15 events
MAD
[55%] 11 events
DPS
[147%] 29 events
SIN
[73%] 14 events

Culture coverage (locale, currency)

83

en-GB
[122%] 648 events
en-US
[105%] 172 events
ru-RU
[60%] 67 events
ko-KR
[61%] 60 events
it-IT
[56%] 50 events
de-DE
[140%] 110 events
es-ES
[62%] 42 events
ja-JP
[109%] 65 events
zh-TW
[92%] 40 events
fr-FR
[98%] 34 events
nl-NL
[100%] 33 events
tr-TR
[53%] 16 events

EUR
[81%] 286 events
GBP
[92%] 213 events
USD
[132%] 222 events
RUB
[44%] 46 events
KRW
[70%] 67 events
AUD
[213%] 143 events
JPY
[107%] 68 events
THB
[87%] 30 events
TWD
[250%] 84 events
TRY
[53%] 16 events
HKD
[155%] 43 events
BRL
[108%] 29 events

How can I download the iOS build?

Here from HockeyApp:

How can I contribute to this?

- Download the RC build
- Test the application
- Check the metrics and change all reds

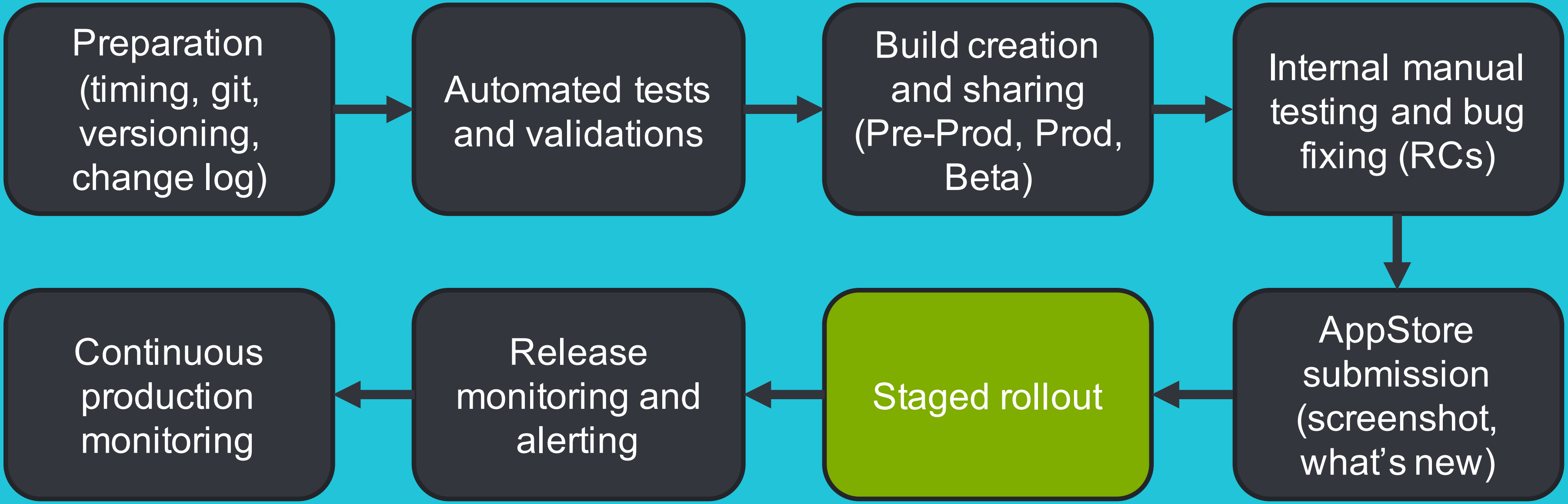
What's the goal of this?

- Find critical issues before we release them to users
- Me, as an AppsTribe member be up-to-

Stability period

- **Internal testing (dogfooding)**
  - Only critical bugfixes are accepted
  - Frequency is a key or a headwind
  - Internal RC testing dashboard
- **Crowdsourced testing**
  - With Applause - half-professional testers from all around the world
  - Testing our develop and release branches too
  - Coverage and flexibility is the key
- **Takes 2 days – 2 weeks**

#3 - Staged rollout


Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what's new)

Staged rollout

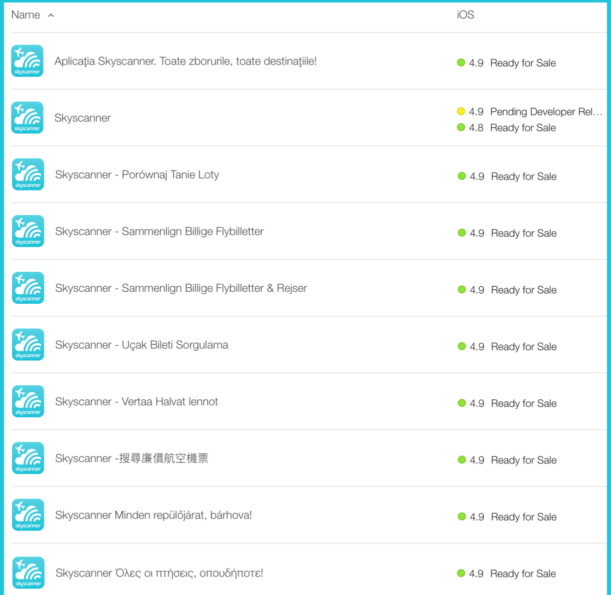
Release monitoring and alerting

Continuous production monitoring

How to do iOS staged rollout?

#1 - Build validation

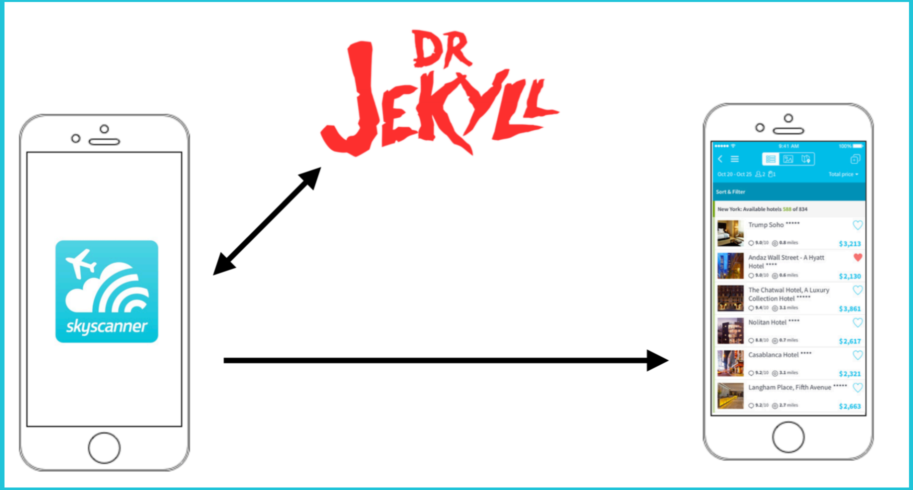
country market build



Name ^	iOS
Aplicaţia Skyscanner. Toate zborurile, toate destinaţiile!	● 4.9 Ready for Sale
Skyscanner	● 4.9 Pending Developer Rel… ● 4.8 Ready for Sale
Skyscanner - Porównaj Tanie Loty	● 4.9 Ready for Sale
Skyscanner - Sammenlign Billige Flybilletter	● 4.9 Ready for Sale
Skyscanner - Sammenlign Billige Flybilletter & Rejser	● 4.9 Ready for Sale
Skyscanner - Uçak Bileti Sorgulama	● 4.9 Ready for Sale
Skyscanner - Vertaa Halvat lennot	● 4.9 Ready for Sale
Skyscanner -搜尋廉價航空機票	● 4.9 Ready for Sale
Skyscanner Minden repülőjárat, bárhova!	● 4.9 Ready for Sale
Skyscanner Όλες οι πτήσεις, οπουδήποτε!	● 4.9 Ready for Sale

#2 - Feature validation

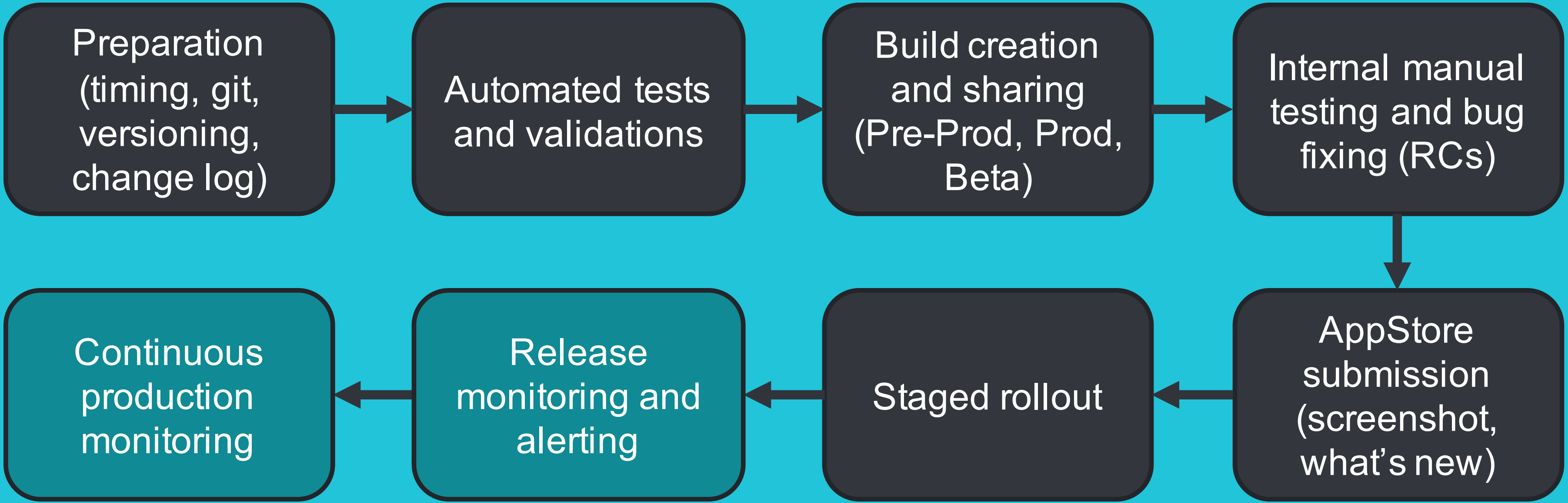
every feature is an experiment



How to do iOS staged rollout?

- **#1 – Build validation (country specific builds)**
  - We have 11 different apps in the store: 1 main, 10 country specific (like HU)
  - Release to one market -> measure -> release to more -> measure -> release globally
  - It's a **technical dept**, so we migrate most of these apps
- **#2 – Feature validation *(every feature is an experiment)***
  - All features are behind a feature flag
  - Unfinished features are released with OFF flag
  - Every feature is an experiment
  - With custom experimentation tool (Dr Jekyll), previously with MixPanel

#4 - Monitoring and alerting



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what’s new)

Staged rollout

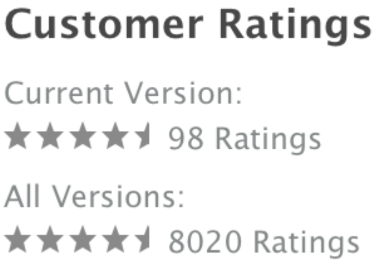
Release monitoring and alerting

Continuous production monitoring

Pushing "Release This Version" – with confident

We know nothing

push the button and check only AppStore reviews



Customer Ratings

Current Version:

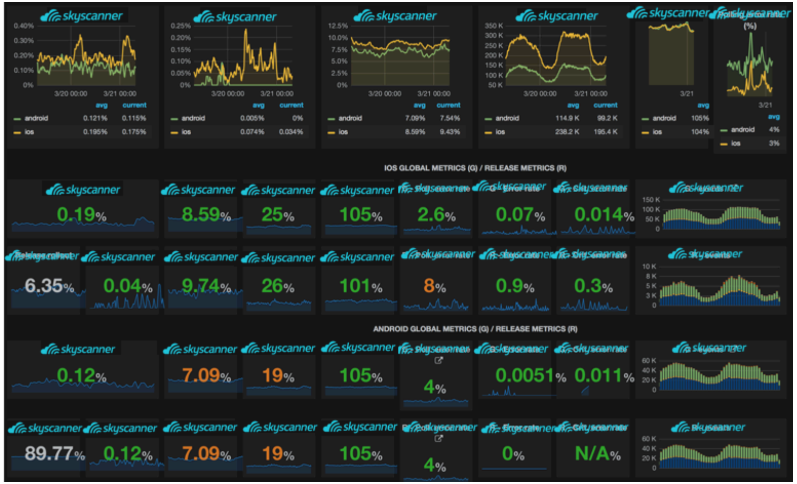
★★★★½ 98 Ratings

All Versions:

★★★★½ 8020 Ratings

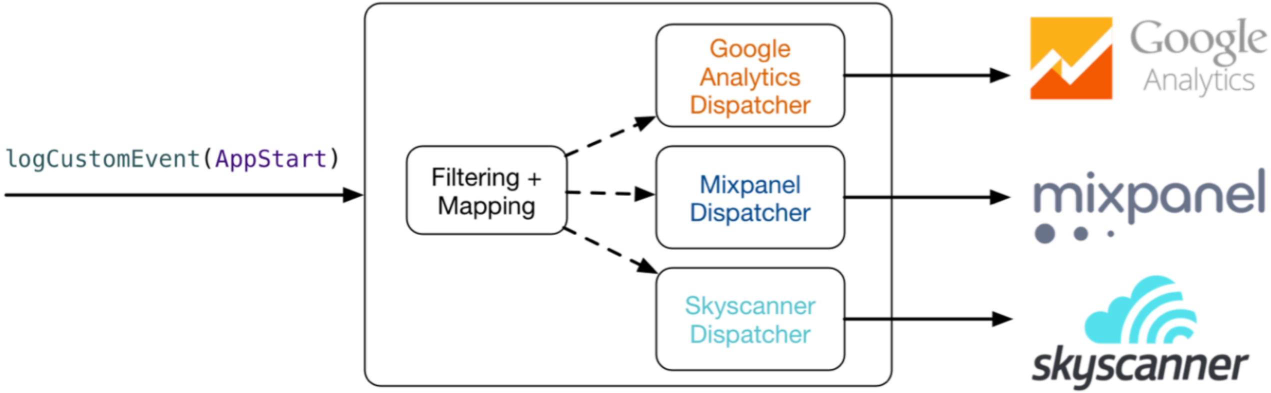
We know everything

get alerts if anything goes wrong



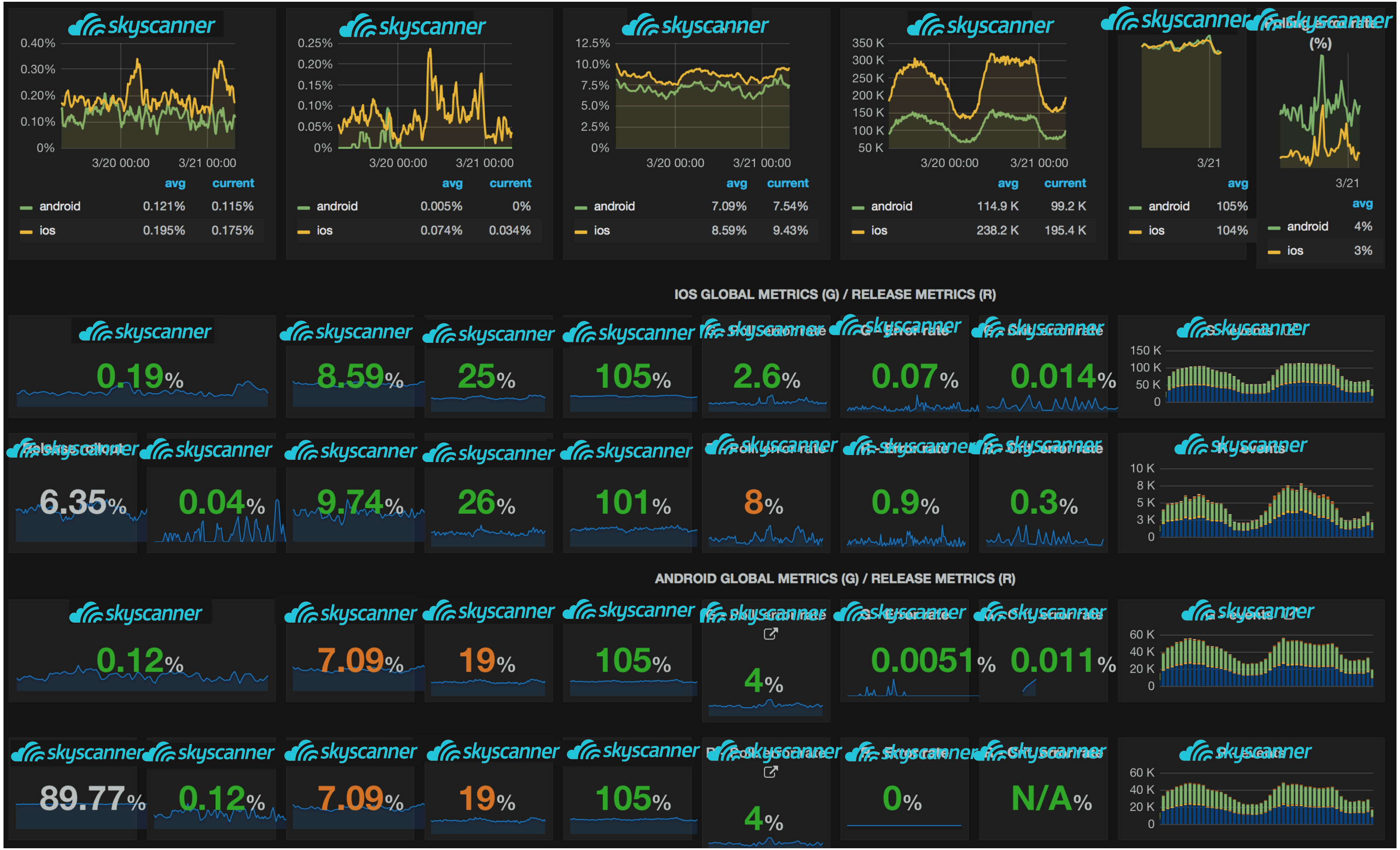
Pushing ”Release This Version” – with confident

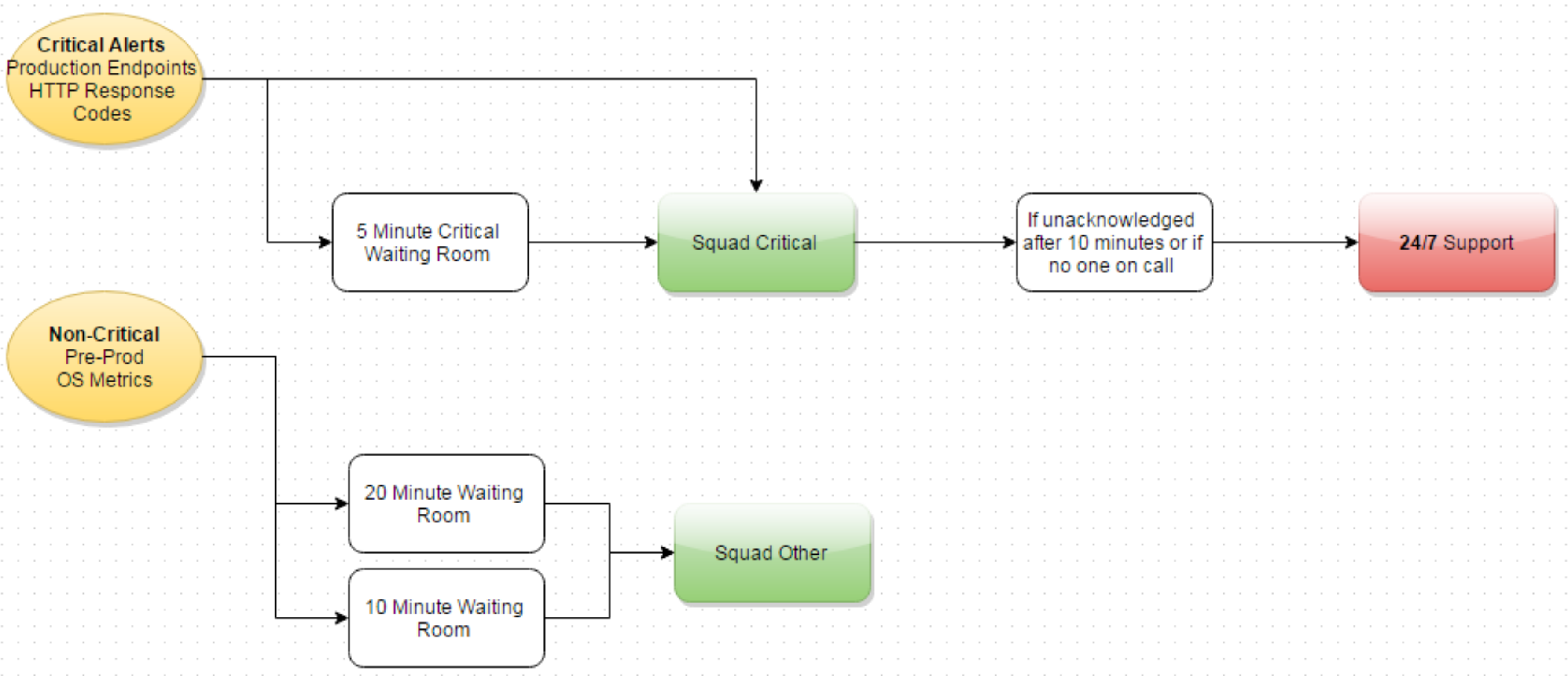
- **Real-time monitoring and alerting system**
  - On our internal data platform (Apache Kafka based)
  - We’re logging everything with the right context



Pushing ”Release This Version” – with confident

- **Real-time monitoring and alerting system**
  - On our internal data platform (Apache Kafka based)
  - We’re logging everything with the right context
  - Real-time metric calculation (pushed to Graphite)
  - Dashboards and alerts based on Graphite metrics
  - Alerts are handled in VictorOps





Critical Alerts
Production Endpoints
HTTP Response Codes

5 Minute Critical Waiting Room

Squad Critical

If unacknowledged after 10 minutes or if no one on call

24/7 Support

Non-Critical
Pre-Prod
OS Metrics

20 Minute Waiting Room

10 Minute Waiting Room

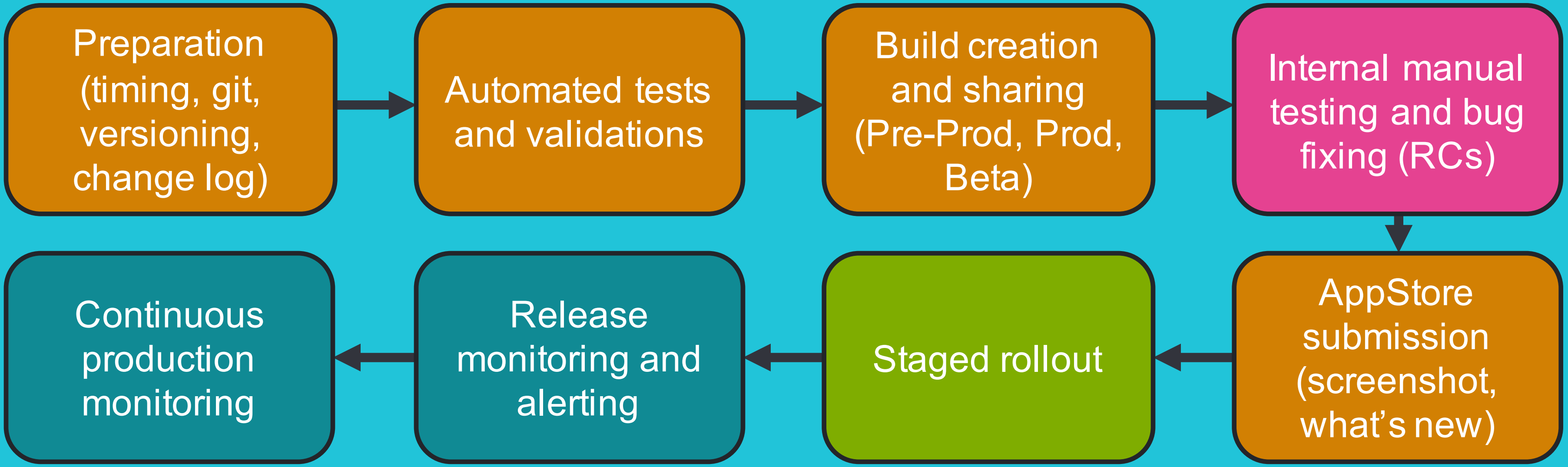
Squad Other

So now you know it – how to release iOS apps

#1 – to millions of users

#2 – every second week

#3 – with confident



Preparation (timing, git, versioning, change log)

Automated tests and validations

Build creation and sharing (Pre-Prod, Prod, Beta)

Internal manual testing and bug fixing (RCs)

AppStore submission (screenshot, what's new)

Staged rollout

Release monitoring and alerting

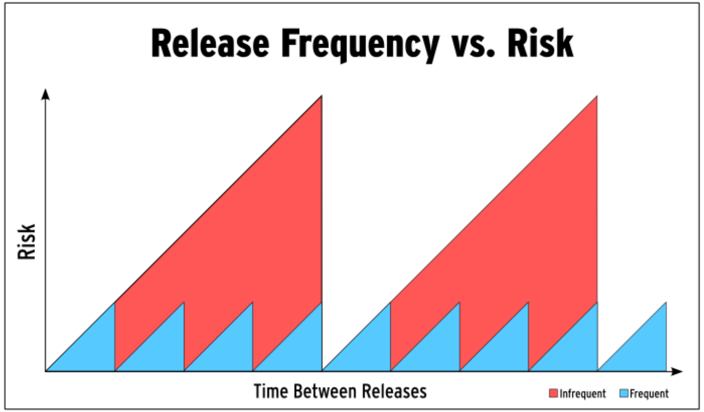
Continuous production monitoring

Release management - iOS vs. Android vs. Web

iOS
pain

Android
more flexible

Web
10000 releases / day



Thank you – questions?

Csaba Szabo
(*Senior Test Engineer*)
csaba.szabo@skyscanner.net

Come and work with us!
http://www.skyscanner.net/jobs/

Release iOS apps

#1 – to millions of users
#2 – every second week
#3 – with confident

skyscanner

Links

- Jenkins CI - https://jenkins-ci.org/
- Fastlane - https://github.com/fastlane/fastlane
- Applause - http://www.applause.com/
- Google dogfooding - http://googletesting.blogspot.hu/2014/01/the-google-test-and-development.html
- MixPanel analytics - https://mixpanel.com/
- Analytics and Data Driven Development in App - http://codevoyagers.com/2016/02/17/analytics-and-data-driven-development-in-apps/
- Apache Kafka - http://kafka.apache.org
- Graphite - http://graphite.readthedocs.org/en/latest/overview.html
- Grafana - http://grafana.org
- Seyren - https://github.com/scobal/seyren
- VictorOps - http://victorops.com