Android Architecture components

Presented by
Mohamed Wael

Outline

- What is Jetpack?
- What is Android Jetpack?
  - Workmanager
  - Navigation
  - Android KTX

JitPack
Easy to use package repository for Git

JetPack

- JitPack is a novel package repository for JVM and Android projects.
- It builds Git projects on demand and provides you with ready-to-use artifacts.
- If you want your library to be available to the world, there is no need to go through project build and upload steps.
- All you need to do is push your project to GitHub and JitPack will take care of the rest. That’s really it!

How to publish to JetPack

- Create the required library
- In root build.gradle

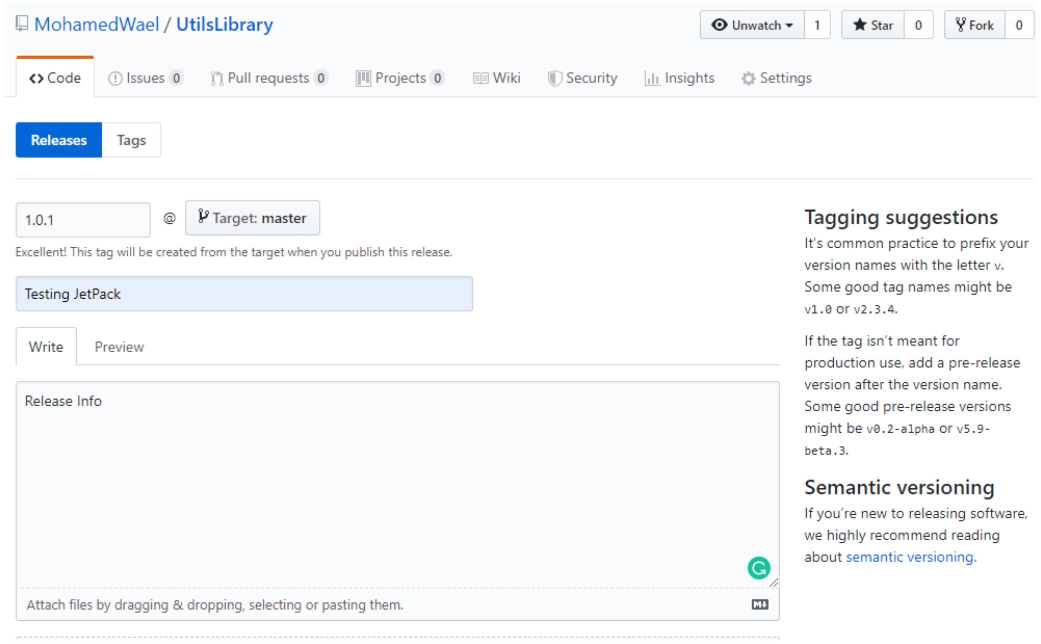
```
dependencies {
    ...
    classpath 'com.github.dcendents:android-maven-gradle-plugin:2.1'
```

- In your library/build.gradle

```
apply plugin: 'com.github.dcendents.android-maven'
group='com.github.UserName'
```

How to publish to JetPack

- Push it to Github
- Create a release



MohamedWael / UtilsLibrary

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

Releases | Tags

1.0.1 @ Target: master

Excellent! This tag will be created from the target when you publish this release.

Testing JetPack

Write | Preview

Release Info

Attach files by dragging & dropping, selecting or pasting them.

Tagging suggestions

It's common practice to prefix your version names with the letter v. Some good tag names might be `v1.0` or `v2.3.4`.

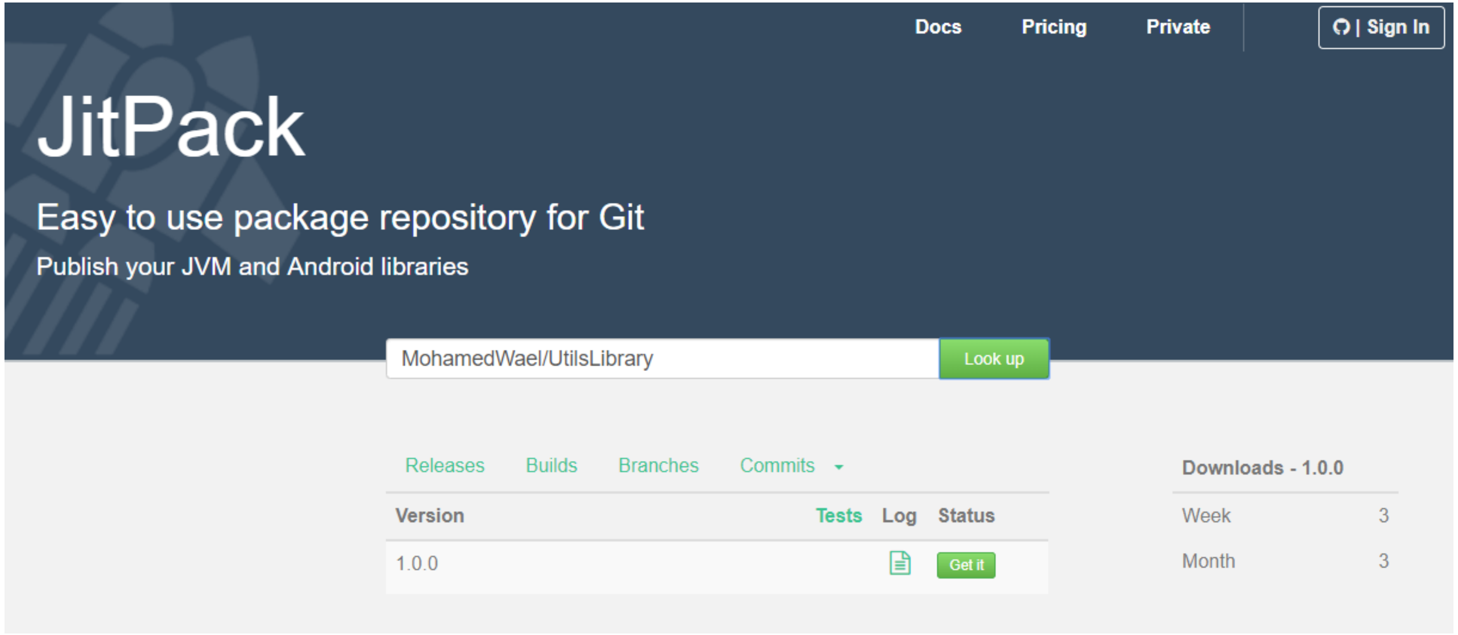
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `v0.2-alpha` or `v5.9-beta.3`.

Semantic versioning

If you're new to releasing software, we highly recommend reading about semantic versioning.

How to publish to JetPack

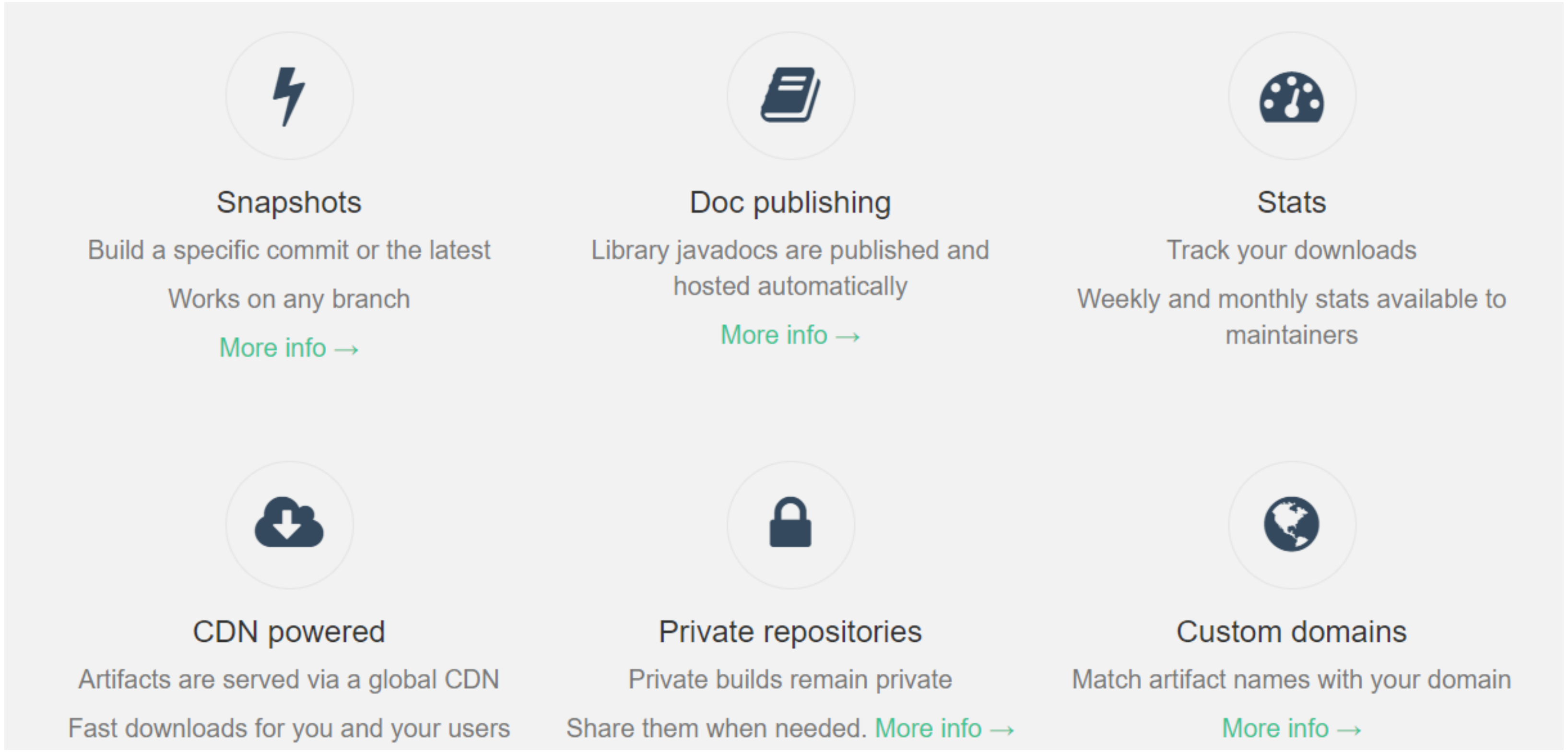
- Now, head to https://jitpack.io/ and enter the address of your repository in the search box and you will receive the address of your library!



Version	Tests	Log	Status
1.0.0		🗎	Get it

Downloads - 1.0.0	
Week	3
Month	3

JetPack features



Snapshots

Build a specific commit or the latest

Works on any branch

More info →

Doc publishing

Library javadocs are published and hosted automatically

More info →

Stats

Track your downloads

Weekly and monthly stats available to maintainers

CDN powered

Artifacts are served via a global CDN

Fast downloads for you and your users

Private repositories

Private builds remain private

Share them when needed. More info →

Custom domains

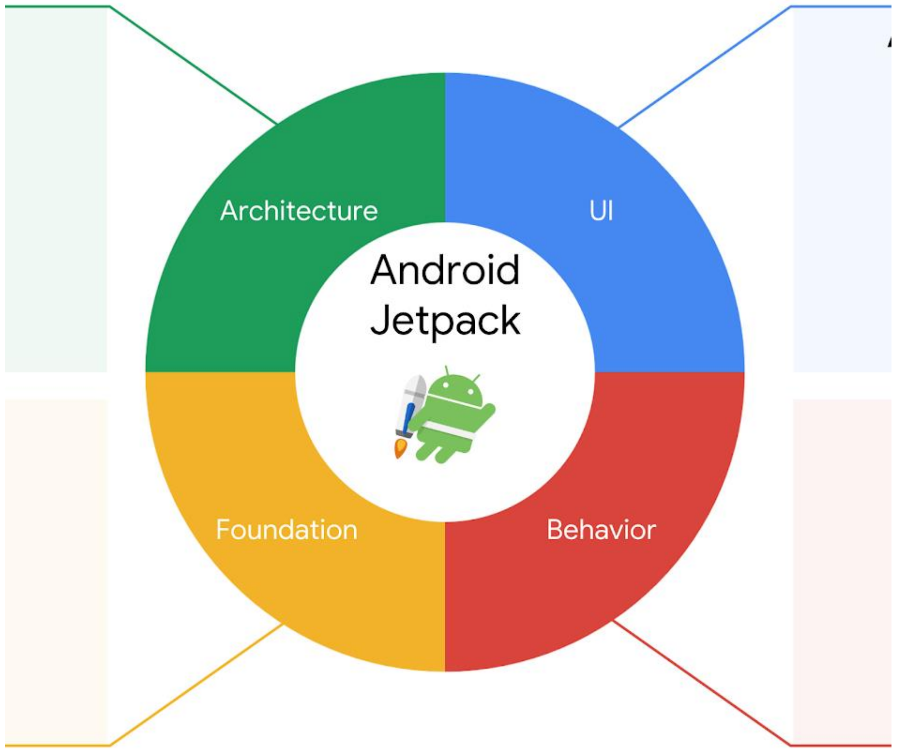
Match artifact names with your domain

More info →

What is Android Jetpack?

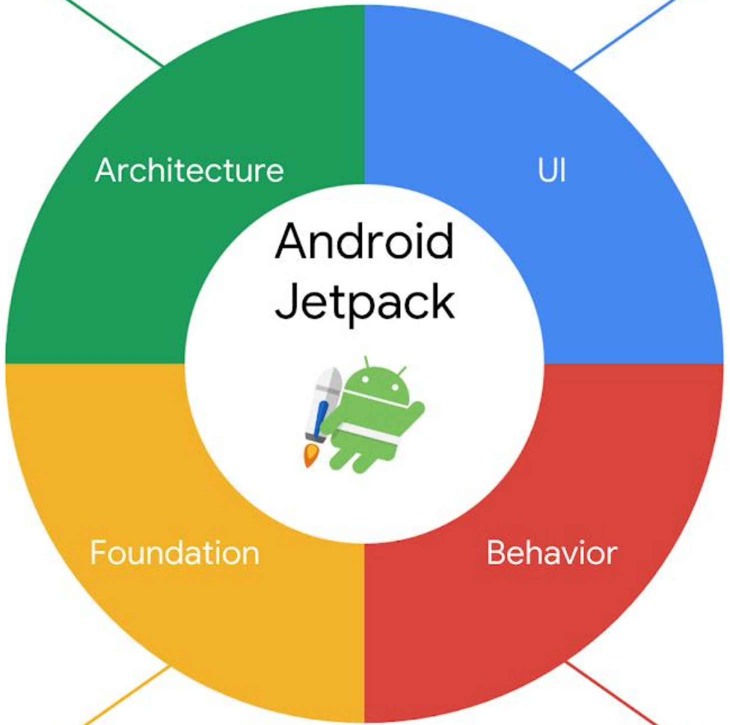
Android Jetpack

- Android Jetpack is a set of components, tools and guidance to make great Android apps.
- The Android Jetpack components bring together the existing Support Library and Architecture Components and arranges them into four categories



Data Binding
Lifecycles
LiveData
Navigation *new!*
Paging *new!*
Room
ViewModel
WorkManager *new!*

Animation & Transitions
Auto, TV & Wear
Emoji
Fragment
Layout
Palette



AppCompat
Android KTX *new!*
Multidex
Test

Download Manager
Media & Playback
Permissions
Notifications
Sharing
new! Slices

WorkManager

- The WorkMananager component is a powerful new library that provides a one-stop solution for **constraint-based background jobs** that need guaranteed execution, replacing the need to use things like jobs or SyncAdapters.

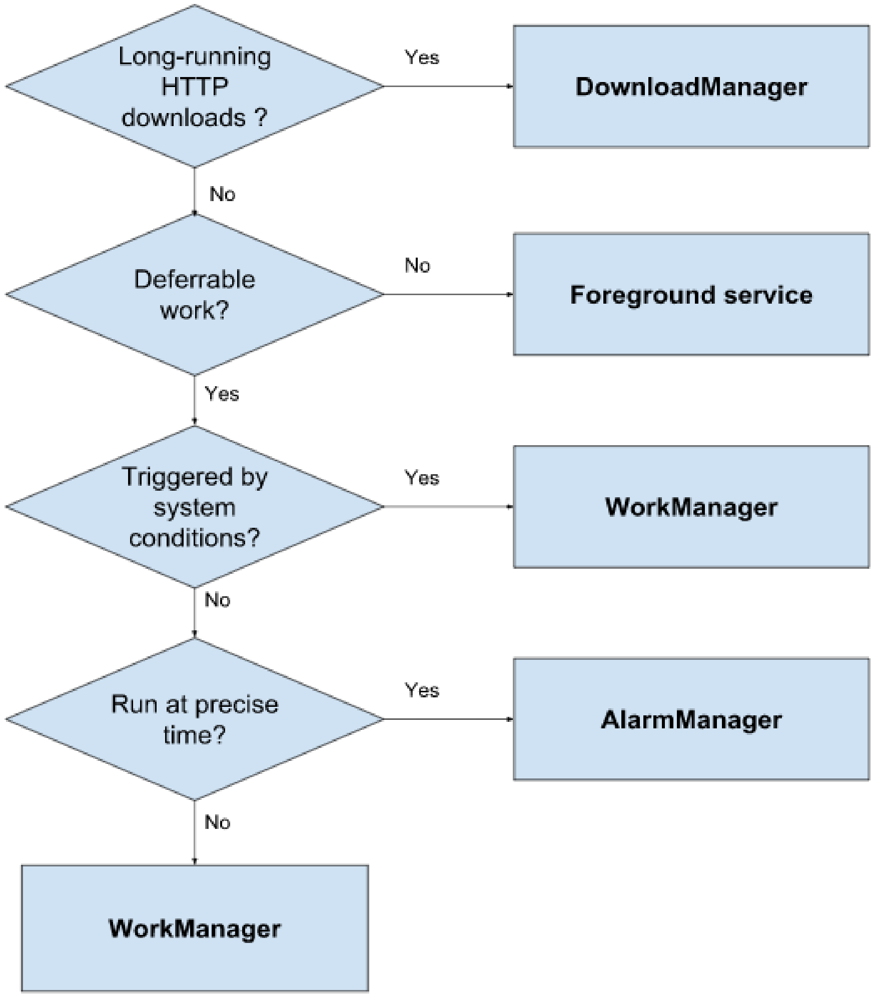
- It is intended for tasks that are **deferrable** — that is, not required to run immediately — and **required to run reliably** even if the **app exits** or the **device restarts**. For example:
  - Sending logs or analytics to backend services
  - Periodically syncing application data with a server

- Backwards compatible up to API 14
  - Uses JobScheduler on devices with API 23+
  - Uses a combination of BroadcastReceiver + AlarmManager on devices with API 14-22

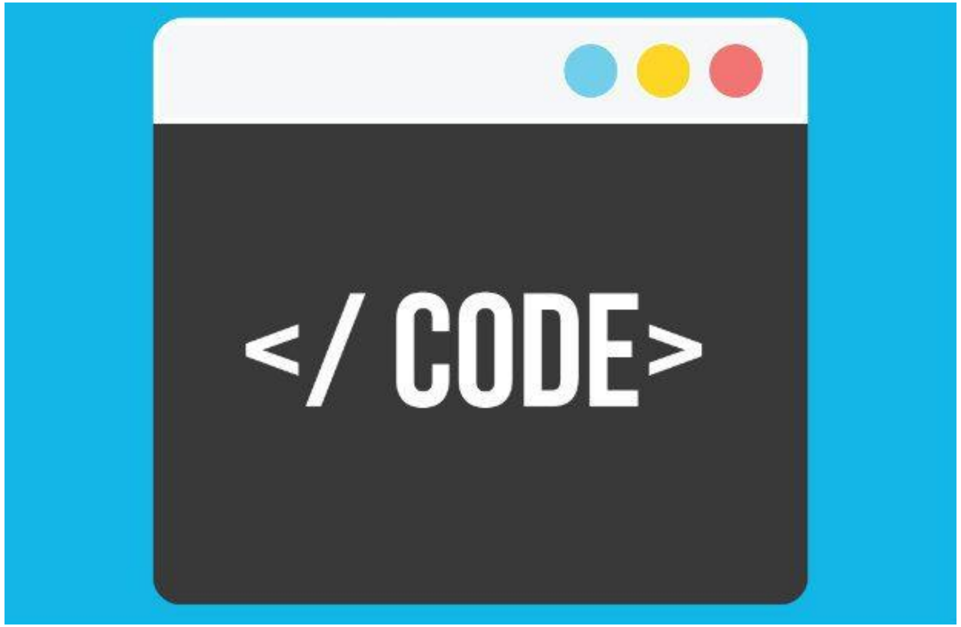
WorkManager

- Add work constraints like network availability or charging status
- Schedule asynchronous one-off or periodic tasks
- Monitor and manage scheduled tasks
- Chain tasks together
- Ensures task execution, even if the app or device restarts
- Adheres to power-saving features like Doze mode
- Find more about WorkManger here

WorkManager



WorkManager



WorkManager

- Adding work manager in app/build.gradle

```
        dependencies {
            ...
            implementation "androidx.work:work-runtime-ktx:2.2.0"
```

- WorkManager Basics
- **Worker:** This is where you put the code for the actual work you want to perform in the background. You'll extend this class and override the **doWork()** method.
- **WorkRequest**: This represents a request to do some work. You'll pass in your Worker as part of creating your WorkRequest.
- When making the WorkRequest you can also specify things like **Constraints** on when the Worker should run.

WorkManager

- WorkManager Basics
- **WorkManager**: This class actually schedules your WorkRequest and makes it run.
- It schedules WorkRequests in a way that spreads out the load on system resources, while honoring the constraints you specify.

- Creating the first worker:
- In the package workers, create a new class called BlurWorker and extend **Worker**.

```
class BlurWorker(ctx: Context, params: WorkerParameters) : Worker(ctx, params) {
    override fun doWork(): Result {}
}
```

- Implement the required work in the **doWork()** method

WorkManager

- Creating the first worker:
- In your viewModel create the workManager instance

```
private val workManager = WorkManager.getInstance(application)
```

- Enqueue the WorkRequest in WorkManager
- There are two types of WorkRequests:
- **OneTimeWorkRequest**: A WorkRequest that will only execute once.
- **PeriodicWorkRequest**: A WorkRequest that will repeat on a cycle.

WorkManager

- Enqueue the WorkRequest in WorkManager
- We only want the blurWorker to be executed once when the Go button is clicked.
- The applyBlur method is called when the Go button is clicked,
- so create a **OneTimeWorkRequest** from BlurWorker there.
- Then, using your WorkManager instance enqueue your WorkRequest.

```
internal fun applyBlur(blurLevel: Int) {
    workManager.enqueue(OneTimeWorkRequest.from(BlurWorker::class.java))
}
```

Resources

- What is Android JetPack: https://android.jlelse.eu/what-is-android-jetpack-737095e88161
- android-workmanager: https://github.com/googlecodelabs/android-workmanager
- Background Work with WorkManager - Kotlin: https://codelabs.developers.google.com/codelabs/android-workmanager-kt/#0