



**What's new in  
Android?**

## **AGENDA**

- Permissionless photo picker
- Figma to Jetpack Compose by Relay
- Jetpack Compose
- Jetpack Compose Animation
- Detekt Library
- Concat Adapter
- Want to join us?

# itAgenturen heroes 🚀



 android

 android

 android



## Permissionless Photo-picker



The photo picker provides -

- a browsable
- searchable UI interface

that presents the user with their media library, sorted by

- date from newest to oldest.

This tool provides a safe, built-in way for users to select images and videos without needing to grant your app access to their entire media library.

**it** Agenturen

## PAIN POINTS 🙄

Before Android 13,

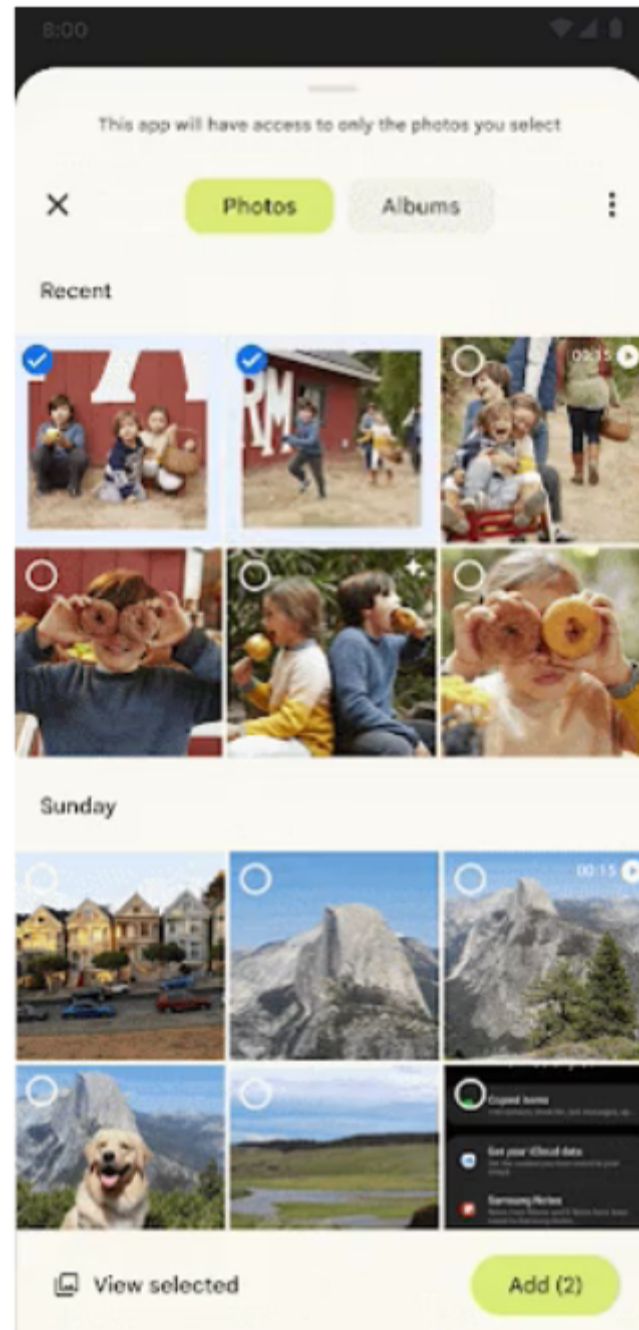
- **Approve READ/WRITE** storage permission just to access the images/videos from your app.
- You need to **add boiler-plate code** to handle those permissions on all the screens of your android app.
- What if the user chooses "**Never Ask Again**" when requesting storage permission?
- Users will see a system's UI gallery picker that feels like it isn't well integrated with App's UX and is **not user-friendly**.

## PAIN KILLERS

From Android 13,

- **NO more READ/WRITE** storage permission to access the images/videos from your app.
- **NO boiler-plate code** to handle those permissions on all your Android app screens.
- **GOOD LOOKING Photo Picker** and more User friendly.
- **Easy-Peasy integration** with more customizations, such as
  - See only photos
  - See only videos
  - Set a maximum number of items to select.

**it** Agenturen



it Agenturen

**Easy-Peasy  
Photo-Picker  
Integration**



**it** Agenturen



## TO PICK SINGLE IMAGE/VIDEO

```
val pickMedia = registerForActivityResult(PickVisualMedia()) { uri ->
    // Callback is invoked after the user selects a media item or closes the
    // photo picker.
    if (uri != null) {
        Log.d("PhotoPicker", "Selected URI: $uri")
    } else {
        Log.d("PhotoPicker", "No media selected")
    }
}

// Include only one of the following calls to launch(), depending on the types
// of media that you want to allow the user to choose from.

// Launch the photo picker and allow the user to choose images and videos.
pickMedia.launch(PickVisualMediaRequest(PickVisualMedia.ImageAndVideo))

// Launch the photo picker and allow the user to choose only images.
pickMedia.launch(PickVisualMediaRequest(PickVisualMedia.ImageOnly))

// Launch the photo picker and allow the user to choose only videos.
pickMedia.launch(PickVisualMediaRequest(PickVisualMedia.VideoOnly))

// Launch the photo picker and allow the user to choose only images/videos of a
// specific MIME type, such as GIFs.
val mimeType = "image/gif"
pickMedia.launch(PickVisualMediaRequest(PickVisualMedia.SingleMimeType(mimeType)))
```

## TO PICK MULTIPLE IMAGE/VIDEO

```
val pickMultipleMedia =
    registerForActivityResult(PickMultipleVisualMedia(5)) { uris ->
        // Callback is invoked after the user selects media items or closes the
        // photo picker.
        if (uris.isNotEmpty()) {
            Log.d("PhotoPicker", "Number of items selected: ${uris.size}")
        } else {
            Log.d("PhotoPicker", "No media selected")
        }
    }

// For this example, launch the photo picker and allow the user to choose images
// and videos. If you want the user to select a specific type of media file,
// use the overloaded versions of launch(), as shown in the section about how
// to select a single media item.
pickMultipleMedia.launch(PickVisualMediaRequest(PickVisualMedia.ImageAndVideo))
```

## TO CHECK IF PHOTO-PICKER IS AVAILABLE.

```
private fun isPhotoPickerAvailable(): Boolean {
    return if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        true
    } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
        getExtensionVersion(Build.VERSION_CODES.R) >= 2
    } else {
        false
    }
}

fun handlePhotoPickerLaunch() {
    if (isPhotoPickerAvailable()) {
        // To launch the system photo picker, invoke an intent that includes the
        // ACTION_PICK_IMAGES action. Consider adding support for the
        // EXTRA_PICK_IMAGES_MAX intent extra.
    } else {
        // Consider implementing fallback functionality so that users can still
        // select images and videos.
    }
}
```

it Agenturen

## **Figma to Jetpack Compose by Relay**

### What and Why Relay?

- Relay provides instant handoff of Android UI components between designers and developers.
- Designers use the Relay for Figma plugin to annotate and package UI components for developer use, including information about layout, styling, dynamic content, and interaction behavior.
- Developers use the Relay for Android Studio plugin to import UI Packages and generate pixel-perfect Jetpack Compose code. This process provides instant layout and styling implementation and speeds up the process of binding data.

## **Key Points :)**

The relay consists of three parts, each of which must be installed separately:

- Relay for Figma Plugin
- Relay for Android Studio Plugin
- Relay Gradle Plugin

Note: Use the latest version of the Relay Gradle plugin to get access to all the latest features.

# Install the Relay for Figma plugin

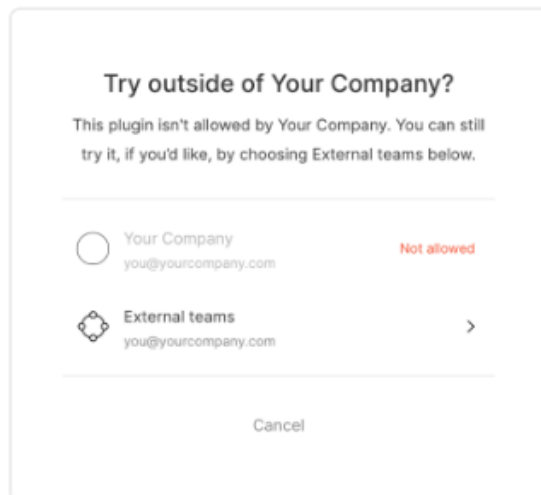
The Relay for Figma plugin is published to Figma's community plugin marketplace. To install the plugin:

1. Go to the [Relay for Figma plugin page](#) on Figma's website.

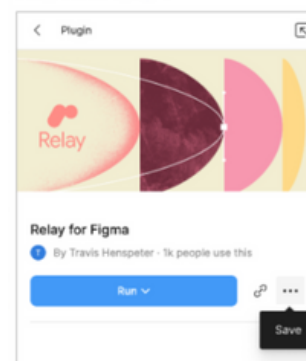
2. On the top right of the page, click **Try it out**.



3. You may encounter a dialog asking if you want to use the plugin. In the dialog, select your organization. In rare cases, your organization may disable public plugins. If so, then select the **External teams** option.



This opens a Figma file with the plugin information page displayed. Click ... and select **Save** to ensure you can easily reuse the plugin.



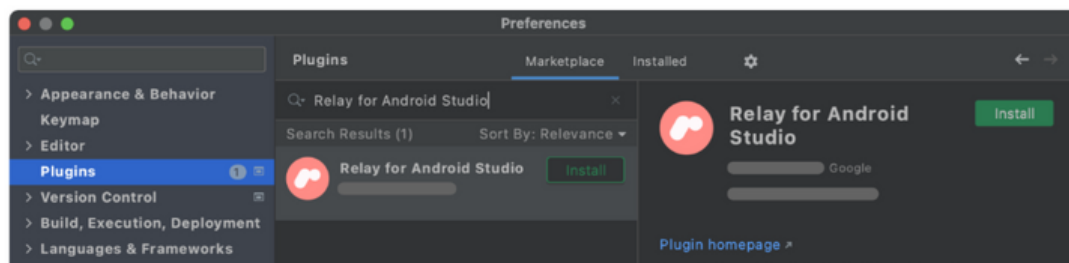
Now, click **Run**:

You can view the plugin running in the Figma canvas:



# Install the Relay for Android Studio plugin

1. Open Android Studio.
2. From the main menu:
  - For macOS, select **Android Studio > Preferences**.
  - For Windows and Linux, select **File > Settings**.
3. Select **Plugins**.
4. Select the Marketplace tab, search for: **"Relay for Android Studio"**, and select the plugin published by Google:



5. Click **Install**.
6. When the installation is complete, click **Restart IDE**.
7. To confirm that the installation succeeded, open **Preferences** or **Settings**, then go to **Plugins**. You should see **Relay for Android Studio** listed in the Installed section.

To ensure that Android Studio, and by extension Gradle itself, downloads and installs the Relay Gradle plugin, you need to specify the plugin as a dependency in your project, specifically in the `build.gradle` file in your module's directory. This is usually in `app/build.gradle`.

At the top of the `app/build.gradle` file is a section listing all of the plugins in use. Add the Relay Gradle plugin in this list, and Gradle handles downloading and installing it. For example:

```
plugins {  
    id 'com.android.application'  
    id 'kotlin-android'  
    id 'com.google.relay' version '0.3.00'  
}
```

Also, make sure your `settings.gradle` file has the following section:

```
pluginManagement {  
    repositories {  
        gradlePluginPortal()  
        google()  
        mavenCentral()  
    }  
}
```

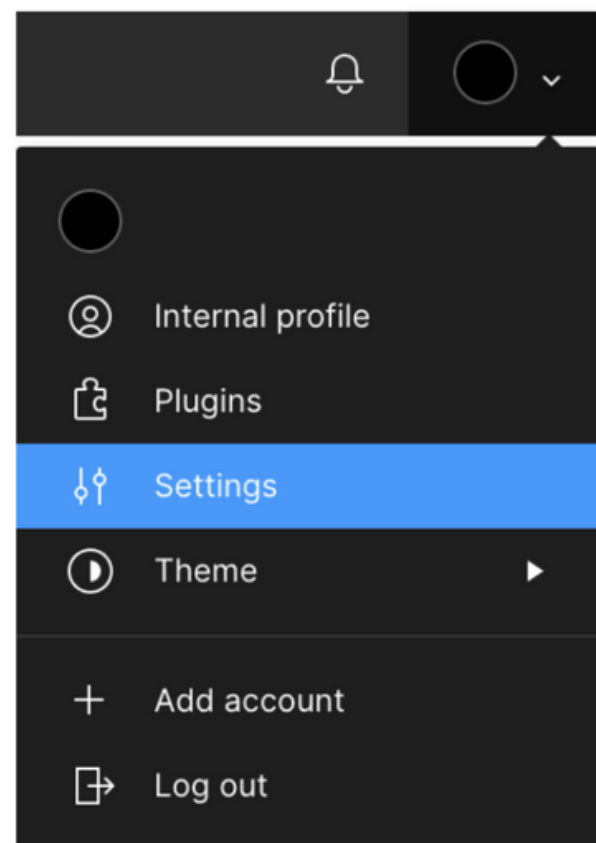
## Setup Figma Access

The relay requires a Figma personal access token to download your Figma designs and convert them to code. If you don't already have one:

Open Figma and log into Figma.

Go to the file browser.

Click your account icon and select Settings.



genturen



## Setup Figma Access

Scroll down to the Personal access tokens section.

Enter a token description, for example, “Relay” and type Return. A token is generated.

Click Copy this token

### Personal access tokens

Personal access tokens allow you to access your own data via the API. Do not give out your personal access tokens to anybody who you don't want to access your files.

Create a new personal access token:

Relay

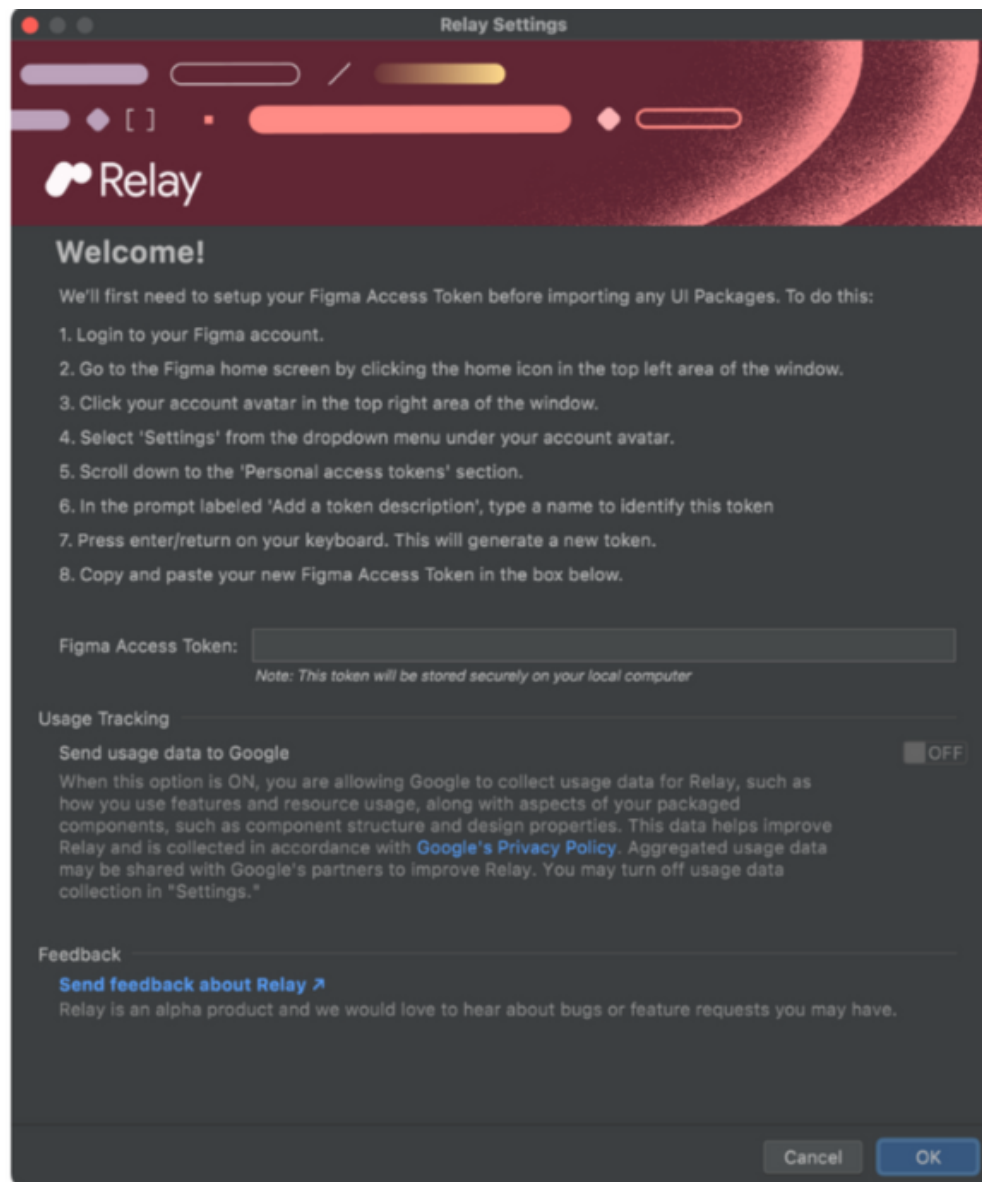
```
1111111-xx11xx11-xx11-1111x-xx1x-  
11x111xx11x1
```

[Copy this token.](#) This is your only chance to do so!

enturen

## Setup Figma Access

- In Android Studio, from the main menu, select Tools > Relay Settings
- If you have previously set up a Figma access token on macOS, you may see the following dialog. This dialog lets you know Android Studio uses your existing Figma access token.
- Enter your password, and click Always Allow..
- Paste your Figma access token into the Figma Access Token input. (If this is not your first time setting up a Figma access token, you may see an Existing Figma Access Token input.)



it Agenturen

## Quick overview on how to work with Figma and Android Studio

### In Figma:

- Design a UI component in Figma.
- Create a UI Package using the Figma plugin.
- Save a named version of the design.
- Share the Figma URL with the developers.

### In Android Studio:

- Using the Figma URL, import the UI Package into a project.
- Generate Jetpack Compose code by building the project.
- Add a reference to the generated composable in the project's UI code.
- Run the project.

# Jetpack Compose

Development Host

Android Studio

Compose Compiler Plugin

Kotlin Compiler

Tech Stack

Device

Compose UI Material

Compose UI Foundation

Compose UI Core


Compose Runtime

it Agenturen

## Composable Methods

```
@Composable  
fun BookLabel() {  
}
```

```
@Composable  
fun BookLabel(book: Book) {  
}
```



All Composable functions must have this annotation; this annotation informs the Compose compiler that this function is intended to convert data into UI

@Composable

```
fun BookLabel( book : Book ) {  
    Text( "${book.quantity}+ ${book.name}" )  
}
```

@Composable

```
fun BookLabel( book : Book ) {  
    if ( book.quantity > 0 ) {  
        Text( "${book.quantity} + ${book.name}" )  
    }  
}
```

it Agenturen

## Jetpack Compose State

```
@Composable
fun ChangeBookName() {
    Column ( modifier = Modifier.padding (16.dp)) {
        var name by remember { mutableStateof(" ") }
        Text (
            text = " Book Name : , $name ",
            modifier = Modifier.padding (bottom = 16.dp) ,
            style = MaterialTheme.typography.h5
        )
        OutlinedTextField (
            value = name ,
            onValueChange = { name = it },
            label = { Text (" Book Name") }
        )
    }
}
```



@Composable

fun BookScreen() {

**var** book **by rememberSaveable** { mutableStateOf(" ") }

    ChangeBookName( name = name, onNameChange = { name = it } )

}

**it** Agenturen

## Quick Demo



**it** Agenturen

## Compose Layout Basics

@Composable

```
fun BookNames() {  
    Text("Introduction to Kotlin")  
    Text("Performance in Android ")  
}
```



Performance in Android

```
@Composable
```

```
fun BookNames() {
```

```
    Column {
```

```
        Text("Introduction to Kotlin")
```

```
        Text("Performance in Android ")
```

```
    }
```

```
}
```

Introduction to kotlin  
Performance in Android

```

@Composable
fun BookCard() {
    Row(verticalAlignment = Alignment.CenterVertically) {
        Image(
            painter = painterResource(R.drawable.book),
            modifier = Modifier.height(150.dp)
        )
        Column(
            Modifier.clickable(onClick = { })
                .padding(16.dp)
        ) {
            Text(
                text = "Android development with kotlin",
                style = TextStyle(fontSize = 24.sp,
                    shadow = Shadow(
                        color = Color.Blue,
                        blurRadius = 3f
                    )
                )
            )
            Text(
                text = "Hardik Trivedi",
                Modifier
                    .padding(top = 10.dp),
                fontSize = 25.sp
            )
        }
    }
}

```



Android development  
with kotlin

Hardik Trivedi

it Agenturen

**@Composable**

```
fun BookSelected() {  
    Box { this: BoxScope  
        Image(  
            painter = painterResource(R.drawable.book), contentDescription: "Book Cover",  
            modifier = Modifier.height(150.dp)  
        )  
        OutlinedButton(onClick = { },  
            modifier= Modifier.align(Alignment.Center).height(50.dp),  
            shape = CircleShape,  
            border= BorderStroke(5.dp, Color( color: 0xFF0F9D58)),  
            contentPadding = PaddingValues(0.dp),  
            colors = ButtonDefaults.outlinedButtonColors(contentColor = Color.Green)  
        ) { this: RowScope  
            Icon(Icons.Default.Check ,  
                contentDescription = "Book Selected",  
                tint=Color.Green)  
        }  
    }  
}
```



**it** Agenturen

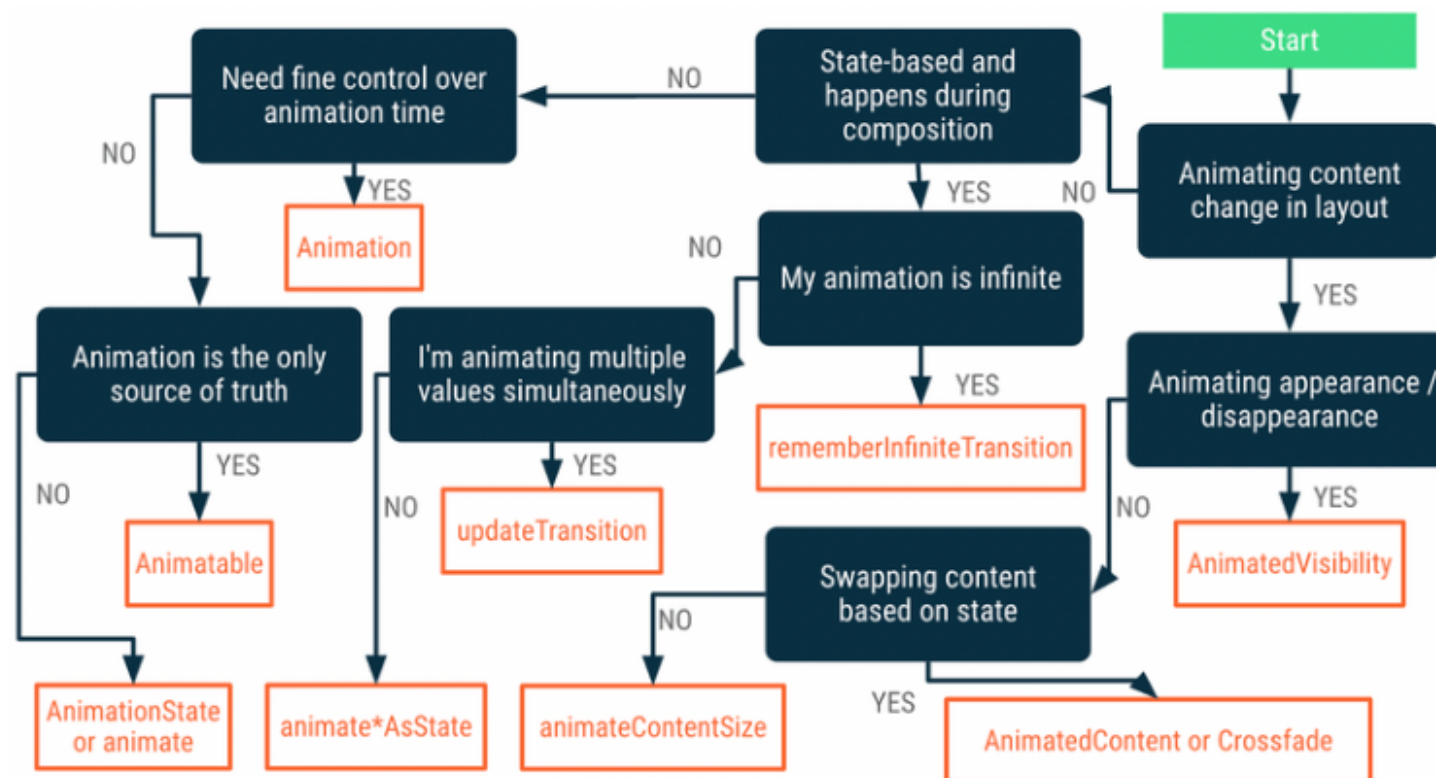
# Jetpack Compose Animation

## Animation Reimagined Compose Animation APIs

Jetpack Compose provides powerful and extensible APIs that make it easy to implement various animations in your app's UI.

- Declarative and interruptible
- Easy to use
- Tooling support

The diagram below helps you decide what API to use to implement your animation.





## *animatedColorAsState()* Animation API

```
@Composable
fun AnimateAsStateDemo() {
    var blue by remember { mutableStateOf(value:true) }
    val color by animateColorAsState(if (blue) Blue else Orange)

    Column { this: ColumnScope
        Button(
            onClick = { blue = !blue }
        ) { this: RowScope
            Text(text = "CHANGE COLOR")
        }
        Spacer(modifier = Modifier.preferredHeight(16.dp))

        Box(
            modifier = Modifier
                .preferredSize(128.dp)
                .background(color)
        )
    }
}
```

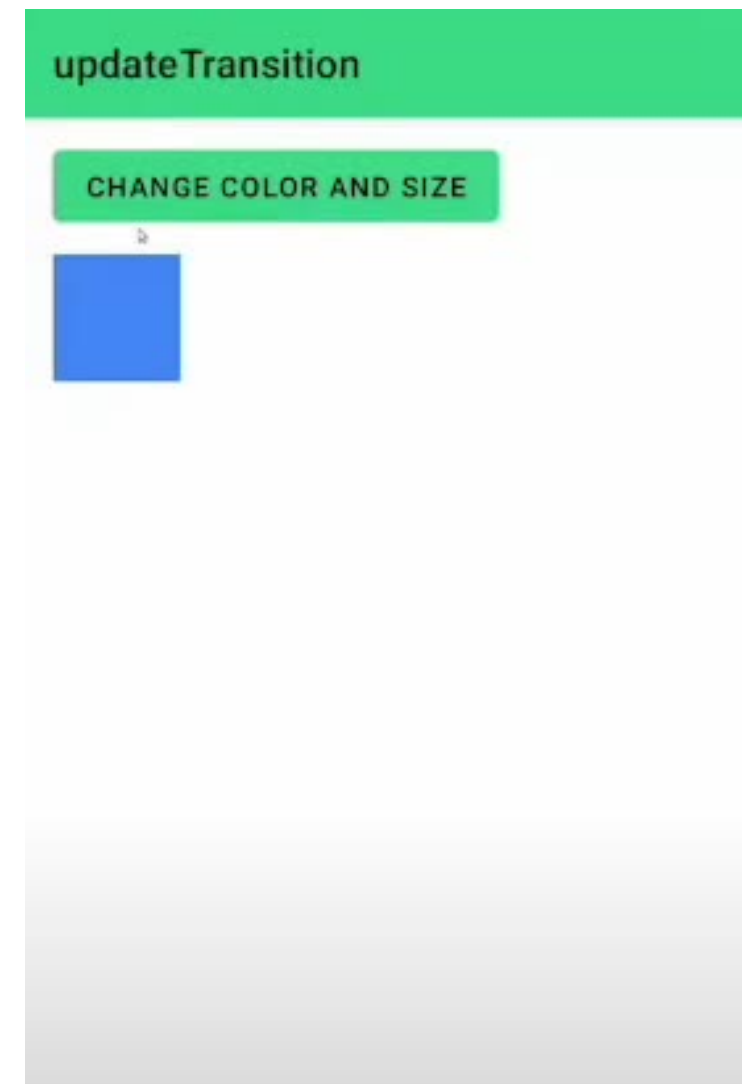
animate\*AsState

CHANGE COLOR



## Animate multiple values at the same time

```
fun UpdateTransitionDemo() {  
    var boxState by remember { mutableStateOf(BoxState.Small) }  
    val transition = updateTransition(targetState = boxState)  
    val color by transition.animateColor() { state ->  
        when (state) {  
            BoxState.Small -> Blue ^animateColor  
            BoxState.Large -> Orange ^animateColor  
        }  
    }  
    val size by transition.animateDp() { state ->  
        when (state) {  
            BoxState.Small -> 64.dp  
            BoxState.Large -> 128.dp  
        }  
    }  
    Column { this: ColumnScope  
        Button(  
            onClick = {  
                boxState = when (boxState) {  
                    BoxState.Small -> BoxState.Large  
                    BoxState.Large -> BoxState.Small  
                }  
            }  
        ) { this: RowScope  
            Text(text = "CHANGE COLOR AND SIZE")  
        }  
    }  
}
```



# AnimatedVisibility & AnimatedContent

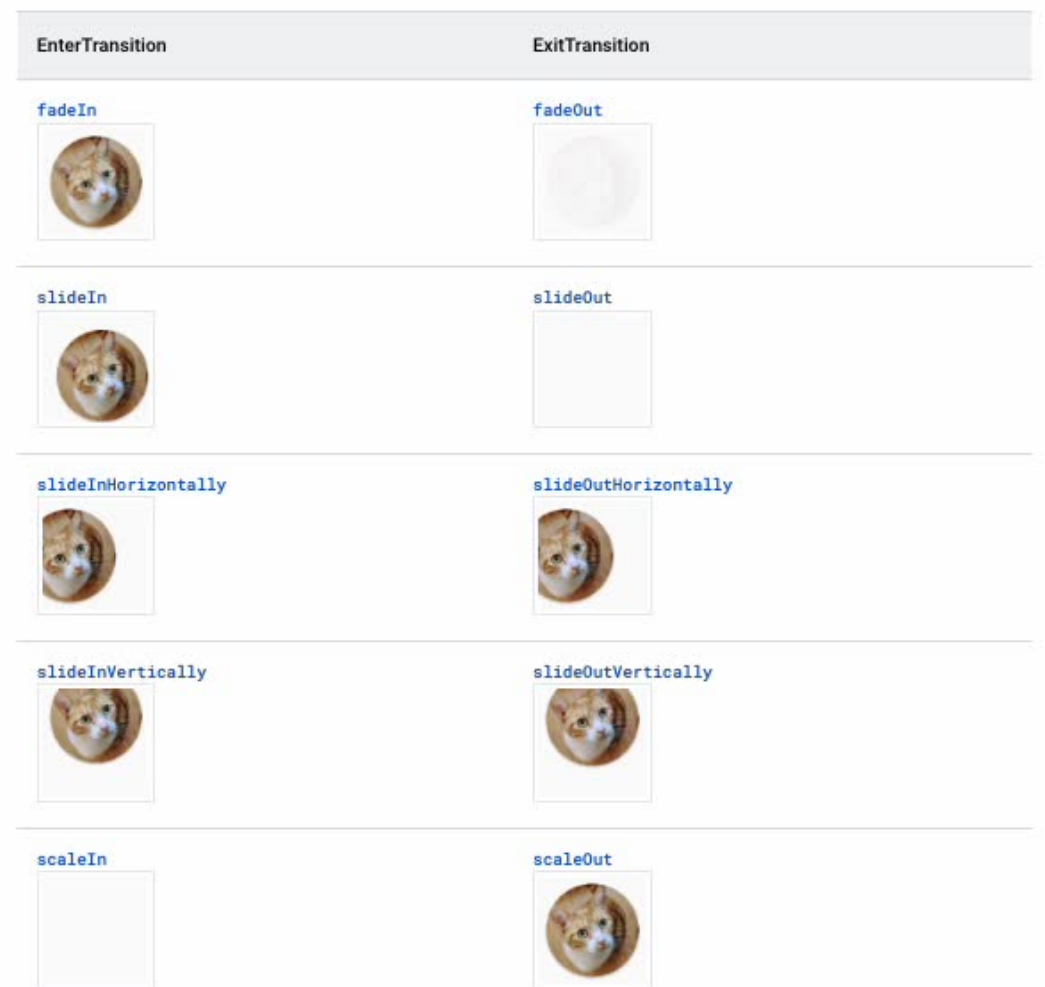
EnterTransition and ExitTransition examples

AnimatedVisibility

Enter and exit if its child

AnimatedContent

Transaction  
between  
content  
changes.



## **Detekt Library**

We are not perfect, right? Somewhere we make mistakes and get silly comments on our PR, like removing commented code, unwanted line space, code not properly formatted, or something like this, especially since we are working with big companies. So Detekt will come up with all these solutions.

When it comes to the code analysis tool, in Java, there are plenty of tools like pmd, findbugs, checkstyle, SonarQube. But when it comes to Kotlin, we have only a few proper options, like Detekt and Klint.

With Klint, the rules can't be modified, and the Gradle plugins are not easy to integrate.

Detekt is a Kotlin-specific linting tool that allows us to enforce codebase documentation and formatting standards and trace bugs before they do any harm.

## How to add Detekt Library in your project?

We need to add it to the top-level build.gradle under the all projects section so it is applied to all of our modules & we also add the Detekt plugin under the plugins section, as shown in the below code snippet.

```
plugins {  
    id "io.gitlab.arturbosch.detekt" version "<detekt-version>"  
}  
  
allprojects {  
    ...  
    apply plugin: "io.gitlab.arturbosch.detekt"  
  
    detekt {  
        toolVersion = "<detekt-version>"  
        config = files("$rootDir/config/detekt/detekt.yml")  
    }  
}
```

## Setup a config

Detekt is highly configurable to meet your requirements. To generate a configuration file with various rules, we need to run the below command:

```
./gradlew detektGenerateConfig
```

Open the newly generated file in `<project-root>/config/detekt/detekt.yml` to start playing with various configuration options. This file contains various options, and we can enable or disable them based on our requirements.

## Running detekt Terminal Command

Now that you have the detekt plugin set up, run the following command on your terminal:

```
./gradlew detekt
```

For example, if we run the detekt on the this DetektSample class, we will get the below error and the build will fail. The error is self-explanatory, the threshold for LongParameterList is defined as 6 but in the implementation, we have 10 parameters which is violating the rule.

```
> Task :app:detekt
Ruleset: complexity - 20min debt
    LongParameterList - 10/6 - [test] at /<file_path> /DetektSample.kt:4:13

Overall debt: 20min

Build failed with 1 weighted issues (threshold defined was 0).
```

```
class DetektSample {

    fun sampleMethod(
        param1: String,
        param2: String,
        param3: String,
        param4: String,
        param5: String,
        param6: String,
        param7: String,
        param8: String,
        param9: String,
        param10: String
    ) {
        //method body here
    }
}
```

## Concat Adapter

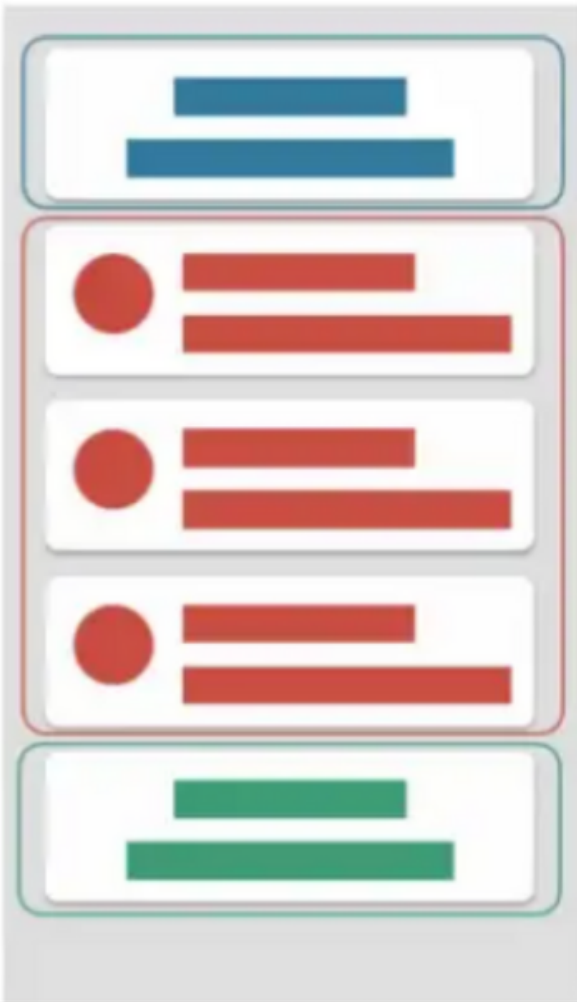
Concat Adapter is a class that was introduced in recycler view dependency (recyclerview:1.2.0- alpha02 )

Concat Adapter provides the ability to handle multiple adapters with a single recycler view.

Benefits of Concat Adapter:

- Keep our layouts simple, this is with fewer lines of XML, therefore, readable and easier to maintain.
- Each adapter has a single responsibility and can be reusable.
- Adapt to changes: adding new features and applying changes to the screen will be easier.



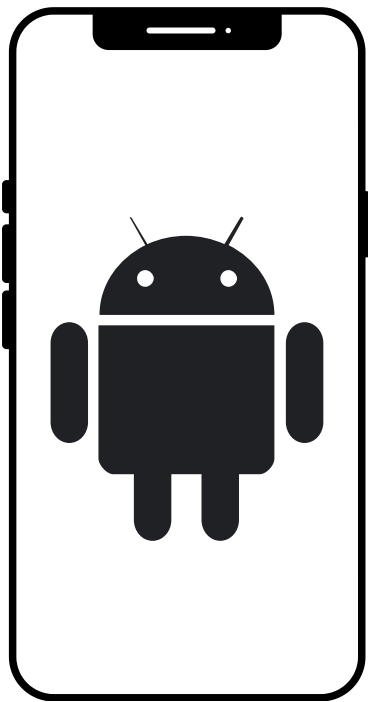


FirstAdapter

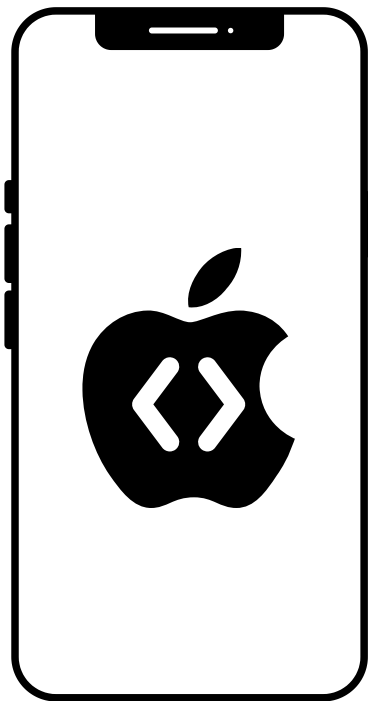
SecondAdapter

ThirdAdapter

**Want to join itAgenturen or have any questions?**



**Android Engineer**



**IOS Engineer**



**[info@itagenturen.com](mailto:info@itagenturen.com)**

**it Agenturen**