Chapitre 9 : Déployer et Sécuriser des Applications Android

Déployer et Sécuriser des Applications Mobiles sous Android

Ce chapitre aborde les étapes essentielles du déploiement et de la sécurisation des applications Android, des phases finales indispensables avant leur mise à disposition des utilisateurs. Il met en lumière les bonnes pratiques permettant d'assurer la fiabilité, la protection des données et la conformité aux exigences de Google Play. Nous aborderons les points suivants :

1	2	3	4
Préparer l'application : Debug vs Release	Signer l'application	Tester avant publication	Importer l'application sur Google Play Console

Qu'est-ce que le déploiement d'applications mobiles ?

Le déploiement d'une application mobile est un processus en plusieurs étapes, crucial pour rendre votre produit accessible aux utilisateurs. Il ne s'agit pas seulement de publier l'application, mais d'une série d'actions allant de la préparation à la maintenance post-lancement.

1

Préparation de l'Application

Cette phase inclut la finalisation du code, les tests rigoureux pour assurer la qualité, la conformité aux directives des plateformes (comme Google Play Store) et la signature numérique de l'application pour garantir son authenticité.

2

Distribution et Publication

Il s'agit de soumettre l'application aux marchés d'applications (par exemple, le Google Play Store). Cela implique de configurer la page du produit, de gérer les versions et de choisir les canaux de distribution appropriés pour atteindre votre public cible.

3

Surveillance et Maintenance

Après le lancement, il est essentiel de suivre les performances de l'application, de recueillir les retours des utilisateurs, de publier des mises à jour régulières pour ajouter des fonctionnalités ou corriger des bugs, et de s'assurer de la sécurité continue de l'application.

La sécurisation des applications mobiles

La sécurisation d'une application mobile consiste à protéger l'application, les données et les utilisateurs contre les accès non autorisés, les fuites d'informations et les attaques, tout au long de son cycle de vie.

Sécurisation et intégrité de l'application Android

La sécurisation d'une application Android repose d'abord sur la protection de l'application elle-même, incluant son code source, les bibliothèques intégrées, le mécanisme de signature et le processus de mise à jour. La signature des APK ou AAB garantit l'authenticité de l'application et permet à Android de vérifier que les mises à jour proviennent bien du même développeur. Par ailleurs, l'utilisation de techniques d'obfuscation du code telles que R8 ou ProGuard permet de rendre le code plus difficile à analyser ou à modifier, renforçant ainsi la protection contre la rétro-ingénierie et les attaques malveillantes.

Protection et sécurisation des données de l'application

Les données manipulées par une application Android constituent un élément critique de sa sécurité, qu'il s'agisse de données locales (bases SQLite, fichiers), de données échangées sur le réseau, ou de données sensibles comme les tokens d'authentification et les mots de passe. Pour garantir leur confidentialité et leur intégrité, plusieurs mécanismes sont essentiels : le chiffrement des données (par exemple avec AES) pour le stockage local, l'utilisation du Android Keystore pour protéger les clés cryptographiques, et la sécurisation des communications réseau via HTTPS / TLS afin d'éviter l'interception ou la modification des données en transit.

Sécurité centrée sur l'utilisateur

La sécurité centrée sur l'utilisateur concerne la gestion de son identité, de ses sessions et des permissions accordées à l'application. Elle vise à s'assurer que seules les personnes autorisées peuvent accéder aux fonctionnalités et aux données sensibles. Cela passe par des mécanismes d'authentification forte (mot de passe robuste, MFA), une gestion rigoureuse des permissions Android selon le principe du moindre privilège, ainsi que l'intégration de solutions de biométrie (empreinte digitale, reconnaissance faciale) pour renforcer l'accès sécurisé tout en améliorant l'expérience utilisateur.

Sécurité des communications entre l'application et l'extérieur

La sécurité des communications concerne les échanges de données entre l'application et les serveurs ou API distants. Elle vise à garantir la confidentialité, l'intégrité et l'authenticité des informations transmises sur le réseau. Cela repose notamment sur l'utilisation de TLS (HTTPS) pour chiffrer les échanges, le certificate pinning pour s'assurer que l'application communique uniquement avec des serveurs de confiance, et des mécanismes de protection contre les attaques de type Man-in-the-Middle (MITM), fréquentes sur les réseaux publics ou non sécurisés.

Publier votre application

Le terme "publication" désigne le processus général qui met vos applications Android à la disposition des utilisateurs. Voici les étapes à suivre lorsque vous publiez une application Android :

Préparer l'application en vue de sa publication

Au cours de l'étape de préparation, vous compilez une version de votre application.

Publier l'application

Au cours de l'étape de publication, vous assurez la promotion, la vente et la distribution de la version finale de votre application que les utilisateurs peuvent télécharger et installer sur leur appareil Android.

Publier votre application

Le terme "publication" est lié à ces termes clés :

Build

Le build est le processus automatisé qui transforme le code source d'une application mobile en un livrable exécutable (APK ou AAB). Dans le contexte du déploiement Android, le build inclut :

- compilation du code
- résolution des dépendances
- optimisation (R8 / ProGuard)
- signature
- génération du livrable final

Un build variant correspond à une version spécifique de l'application générée lors de la compilation, résultant de la combinaison d'un type de build (*debug* ou *release*) et, s'ils existent, de product flavors. Chaque build variant est conçu pour un contexte précis (développement, test, production, version gratuite ou payante) et génère un APK ou AAB distinct, comme par exemple *debug*, *release*, *freeDebug* ou *paidRelease*.

Debug

Le mode Debug est une configuration de build destinée au développement et aux tests, facilitant l'analyse et le débogage de l'application. Ses caractéristiques sont les suivants :

- signature avec clé debug
- logs activés
- pas ou peu d'optimisation
- installable directement

Il n'est jamais utilisé pour la production.

Release (version de production)

Une version release est une version finale de l'application, destinée aux utilisateurs finaux et prête à être publiée.

Une build **release** :

- est signée avec une clé sécurisée
- n'est pas débogable
- utilise des optimisations (suppression du code inutile)
- ne contient pas de logs de debug
- Garantit la sécurité, les mises à jour futures, la performance et la stabilité
- APK signé (usage local ou interne)
- AAB signé (publication Play Store)

Publier votre application

On peut distribuer des applications Android aux formats suivants :

APK (Android Package)

Un **APK (Android Package)** est le fichier bianaire final installable sur un appareil Android.
Il contient tout ce qui est nécessaire à l'exécution de l'application.

Un APK est une archive qui regroupe :

- le code compilé (`.dex`)
- les ressources (images, layouts, chaînes de caractères…)
- le manifeste Android (`AndroidManifest.xml`)
- les bibliothèques natives
- la signature numérique

Installable directement sur un appareil Android

Peut être installé via Google Play ou manuellement

Toujours utilisé techniquement, mais **n'est plus le format principal de publication sur Google Play**

AAB (Android App Bundle)

Un **AAB (Android App Bundle)** est un format de distribution et de publication, introduit par Google pour optimiser la diffusion des applications Android.

Il n'est pas installable directement sur un appareil.

Fournit à Google Play tous les éléments de l'application

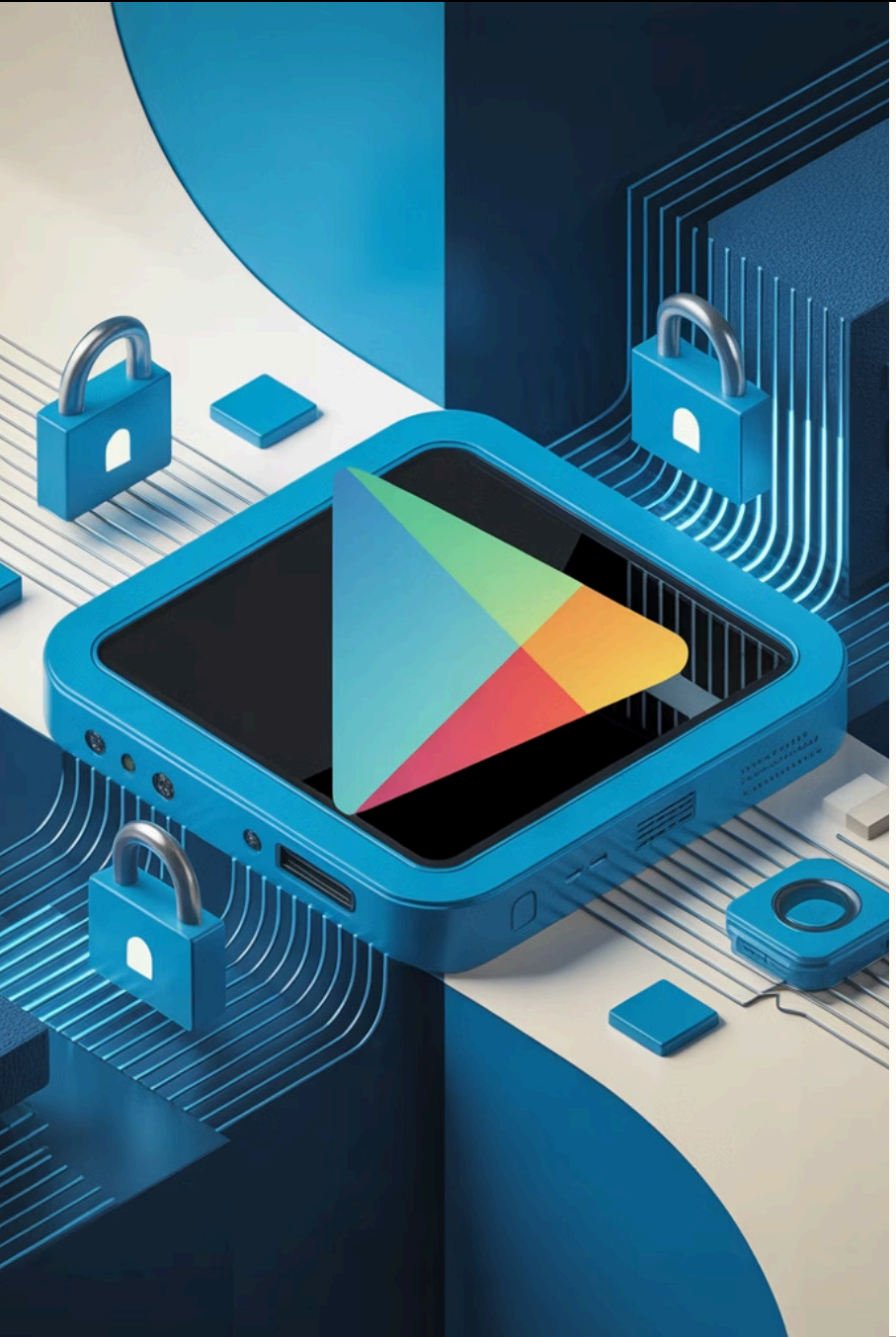
Permet à Google de générer des APK optimisés par appareil

Réduit de la taille des applications et Optimisation par type d'appareil

Sécurité renforcée (Play App Signing)

Format obligatoire sur Google Play pour les nouvelles applications (depuis 2021)

Le développeur signe l'AAB et Google Play vérifie la signature, génère des APK spécifiques (langue, ABI, densité écran et signe les APK finaux pour les utilisateurs



Préparer la publication de votre application

1. **Configurer la publication de votre application**
2. **Compiler et signer la version finale de votre application**
3. **Tester la version de votre application**
4. **Mettre à jour les ressources de l'application pour la publier**
5. **Préparer les serveurs et services distants dont votre application dépend**

ETAPE 1 : Configurer la publication de votre application

S'assurer que la journalisation (logs) est désactivée ou supprimée

- Supprimer les logs sensibles en production
- Ou les encapsuler dans un test de debug :

```
if (BuildConfig.DEBUG) {
    Log.d("TAG", "Message de debug")
}
```

Ainsi, aucun log ne sera exécuté en version release.

Vérifier que `debuggable` est à `false`

Fichier :

```
app/build.gradle.kts
```

vérifier :

```
android {
    buildTypes {
        getByName("release") {
            isDebuggable = false
            isMinifyEnabled = true
            isShrinkResources = true
        }
    }
}
```

Pour garantir que l'application n'est pas attachable par un debugger.

ETAPE 1 : Configurer la publication de votre application

Définir les informations relatives à la version de votre application

Le système Android applique la compatibilité des versions du système, comme indiqué par le paramètre `minSdk` dans les fichiers de compilation. Ce paramètre permet à une application de spécifier l'API système minimale avec laquelle elle est compatible.

```
android {
  namespace = "com.example.testapp"
  compileSdk = 33

  defaultConfig {
     applicationId = "com.example.testapp"
     minSdk = 24
     targetSdk = 33
     versionCode = 1
     versionName = "1.0"
     ...
  }
  ...
}
...
```

Définir les valeurs de la version

Vous pouvez définir des valeurs par défaut pour ces paramètres en les incluant dans le bloc `defaultConfig {}`, imbriqué dans le bloc `android {}` du fichier `build.gradle` ou `build.gradle.kts` de votre module. Vous pouvez ensuite ignorer ces valeurs par défaut pour différentes versions de votre application en définissant des valeurs distinctes pour chaque type de compilation ou type de produit. Le fichier suivant affiche les paramètres `versionCode` et `versionName` dans le bloc `defaultConfig {}`, ainsi que le bloc `productFlavors {}`.

Ces valeurs sont ensuite fusionnées dans le fichier manifeste de votre application au cours du processus de compilation.

```
android {
   ...
   defaultConfig {
      ...
      versionCode = 2
      versionName = "1.1"
   }
   productFlavors {
      create("demo") {
         ...
         versionName = "1.1-demo"
      }
      create("full") {
         ...
      }
   }
}
```

ETAPE 2 : Compiler et signer la version finale de votre application

Vous pouvez utiliser les fichiers de compilation Gradle avec le type de compilation *release* pour compiler et signer la version finale de votre application.

Android exige que toutes les applications Android soient signées à l'aide d'un certificat numérique.
Aujourd'hui, la publication sur Google Play se fait via un **Android App Bundle (AAB)**, à partir duquel Google génère et signe les **APK optimisés** destinés aux appareils des utilisateurs.

Les procédures à suivre pour signer une nouvelle application sur Google Play :

1. Générer une clé d'importation et un keystore
2. Signer votre application avec votre clé d'importation
3. Configurer la signature d'application Play

Préparer la publication de votre application

ETAPE 3 : Tester la version de votre application

Avant de distribuer votre application, vous devez tester la version à publier sur au moins un appareil de la taille d'un téléphone portable et un appareil de la taille d'une tablette. **Firebase Test Lab** est utile pour effectuer des tests sur une grande variété d'appareils et de configurations.

ETAPE 4 : Mettre à jour les ressources de l'application pour la publier

Assurez-vous que toutes les ressources de l'application, telles que les fichiers multimédias et les images, sont mises à jour et incluses avec l'application ou transférées sur les serveurs de production appropriés.

ETAPE 5 : Préparer les serveurs et services distants dont votre application dépend

Si votre application dépend de serveurs ou de services externes, assurez-vous qu'ils sont sécurisés et prêts pour la production.

Sur Google Play Console

1	2	3
Compte sur la place de marché Si vous n'en avez pas déjà, vous devez créer un compte sur la place de marché d'applications que vous souhaitez utiliser.	**Icône de l'application** Vous devez également créer une icône pour votre application.	**Contrat de licence utilisateur final (CLUF)** Et préparer un contrat de licence utilisateur final (CLUF) pour votre organisation, votre propriété intellectuelle et vous-même.

Signature de l'application

La signature d’une application Android est un mécanisme de sécurité basé sur la cryptographie qui permet :

- d’identifier l’auteur de l’application
- de garantir l’intégrité du code
- d’autoriser les mises à jour d’une application existante

Signature de l application

Keystores et clés

Keystores

Les keystores Java (.jks ou .keystore) sont des fichiers binaires servant de dépôts de certificats et de clés privées. Ils permettent de :

- Stocker les clés utilisées pour signer l’application
- Garantir que seul le propriétaire peut signer des mises à jour

Un alias est un nom logique qui identifie une clé spécifique à l’intérieur d’un keystore. Un keystore peut contenir plusieurs clés .

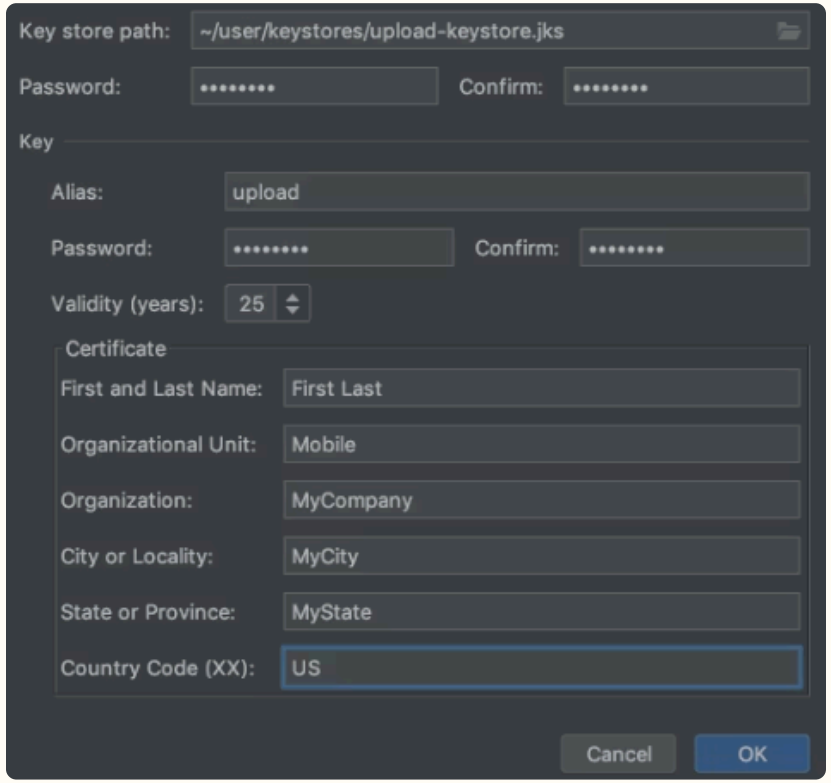
Clés

Clé de signature d'application : elle permet de signer les fichiers APK installés sur l'appareil d'un utilisateur. Conformément au modèle de mises à jour sécurisées Android, la clé de signature reste la même pendant toute la durée de vie de votre application.

Clé d'importation : elle sert à signer l'app bundle ou le fichier APK avant de l'importer pour la signature d'application avec Google Play.

Signature de l' application

ETAPE 1 : Générer une clé d'importation et un keystore



1. Dans la barre de menu, cliquez sur **Build > Generate Signed Bundle/APK** .
2. Dans la boîte de dialogue **Generate Signed Bundle or APK** (Générer un app bundle/APK signé), sélectionnez **Android App Bundle (ce qui est recommandé)**, puis cliquez sur **Next** (Suivant).
3. Sous **Key Store Path** (Chemin du keystore), cliquez sur **Create new**.
4. Dans la fenêtre **New Key Store** (Nouveau keystore), renseignez les champs concernant votre keystore et votre clé.

Signature de l' application

ETAPE 1 : Générer une clé d'importation et un keystore

1. **Keystore**
    - **Key store path** (Chemin d'accès) : sélectionnez l'emplacement où votre keystore doit être créé. Ajoutez également un nom de fichier avec l'extension `.jks` à la fin du chemin d'accès.
    - **Password** (Mot de passe) : créez et confirmez un mot de passe sécurisé pour votre keystore.
2. **Clé**
    - **Alias:** : saisissez un nom permettant d'identifier votre clé.
    - **Password** (Mot de passe) : créez et confirmez un mot de passe sécurisé pour votre clé. Utilisez le même mot de passe que pour votre keystore.
    - **Validity (years)** (Validité en années) : définissez la durée de validité de votre clé, en années. Votre clé doit être valide pendant au moins 25 ans pour que vous puissiez signer des mises à jour d'applications avec la même clé pendant toute la durée de vie de votre application.
    - **Certificate** (Certificat) : saisissez certaines informations vous concernant pour votre certificat. Ces informations n'apparaissent pas dans votre application, mais sont incluses dans votre certificat avec l'APK.
3. Une fois le formulaire rempli, cliquez sur OK.

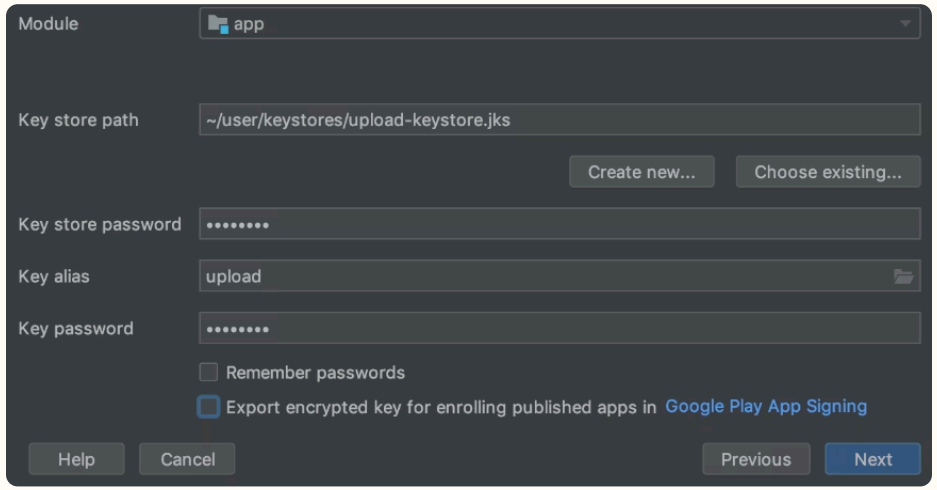
Si vous souhaitez créer et signer votre application avec votre clé d'importation, passez à la section Signer votre application avec votre clé d'importation. Si vous souhaitez uniquement générer la clé et le keystore, cliquez sur Annuler.

Signature de l' application

ETAPE 2 : Signer votre application avec votre clé d'importation

Pour signer votre application avec Android Studio, procédez comme suit :

1. Si la boîte de dialogue Generate Signed Bundle or APK (Générer un app bundle/APK signé) n'est pas ouverte, cliquez sur Build > Generate Signed Bundle/APK (Créer > Générer un app bundle/APK signé).
2. Dans la boîte de dialogue Generate Signed Bundle or APK (Générer un app bundle/APK signé), sélectionnez Android App Bundle ou APK, puis cliquez sur Next (Suivant).
3. Sélectionnez un module dans la liste déroulante.
4. Indiquez le chemin d'accès à votre keystore et à l'alias de votre clé, puis saisissez leurs mots de passe respectifs.
5. Cliquez sur Next (Suivant).



Signature de l' application

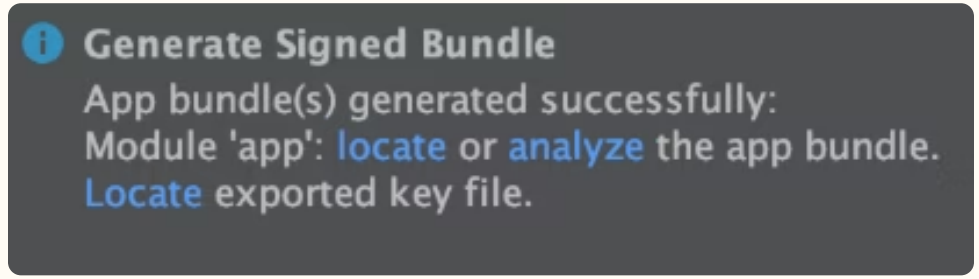
ETAPE 2 : Signer votre application avec votre clé d'importation

Pour signer votre application avec Android Studio, procédez comme suit :

6. Dans la fenêtre suivante, sélectionnez un dossier de destination pour votre application signée. Ensuite, sélectionnez le type de compilation et, si nécessaire, choisissez le ou les types de produit.

7. Si vous créez et signez un APK, vous devez sélectionner les versions de signature que votre application doit accepter.

8 . Cliquez sur Créer.



Une fois qu'Android Studio a terminé de compiler votre application signée, vous pouvez la localiser (localize) ou l'analyser (analyze) en cliquant sur l'option appropriée dans la notification pop-up

Signature de l' application

ETAPE 3 : Utiliser la signature d'application Play

Configurer votre application

1. Connectez-vous à la **Play Console**.
2. Créez votre nouvelle release en suivant les instructions pour la préparer et la déployer.
3. Après avoir choisi un canal de publication, configurez la signature d'application dans la section **App signing** (Signature d'application).

Une version est une combinaison d'une ou de plusieurs révisions d'application que vous préparez avant de déployer une application ou la mise à jour d'une application. Vous pouvez créer une version dans trois canaux de tests différents ou dans le canal de production :

- **Canal de tests ouverts** : les versions de tests ouverts sont accessibles aux testeurs sur Google Play. Les utilisateurs peuvent rejoindre vos tests depuis votre fiche Play Store.
- **Canal de tests fermés** : les versions de tests fermés sont accessibles à un nombre limité de testeurs que vous sélectionnez et qui testent une version préliminaire de votre application et vous font part de leurs commentaires.
- **Canal de tests internes** : les versions de tests internes sont accessibles aux testeurs de votre choix (100 maximum).
- **Canal de production** : les versions de production sont accessibles à tous les utilisateurs de Google Play dans les pays de votre choix.

Signature de l' application

Que se passe-t-il derrière l'assistant graphique d'Android Studio

Quand on clique dans Android Studio sur : Build → Generate Signed Bundle / APK , Android Studio ne compile rien lui-même. Il agit comme une interface graphique qui déclenche trois mécanismes techniques précis.

1 Création et gestion de la clé de signature (keytool)

- Il appelle l'outil Java standard `keytool`
- Il génère :
- un keystore (.jks)
- une clé privée
- un certificat auto-signé

Android Studio masque cet outil, mais le mécanisme est universel et indépendant de l'IDE.

3 Compilation réelle de l'application (Gradle)

Gradle est l'outil qui :compile le code, applique les règles de sécurité, signe l'application et génère l'AAB ou l'APK.

Quand on clique sur *Generate Signed Bundle*, Android Studio exécute en arrière-plan :

./gradlew bundleRelease pour un AAB destiné à être transformé par Google Play

ou

./gradlew assembleRelease pour un APK prêt à installer

2 Configuration de la signature (signingConfigs)

Associer un keystore, une clé et un mot de passe à une version de compilaton (`release`)

Il écrit automatiquement dans `build.gradle.kts`, une configuration du système de build (Gradle) :

```
signingConfigs {
    create("release") {
        storeFile = file("my-release-key.jks")
        storePassword = "****"
        keyAlias = "mykey"
        keyPassword = "****"
    }
}
```

```
buildTypes {
    getByName("release") {
        signingConfig = signingConfigs.getByName("release")
    }
}
```

Signature de l' application

Que se passe-t-il derrière le bouton d'Android Studio

Compilation réelle de l'application (Gradle)

1 Commande principale

```
./gradlew bundleRelease
```

- Cette commande génère un Android App Bundle (AAB) destiné à Google Play.
- Elle ne produit pas directement un APK installable, mais un fichier `.aab` optimisé.

3 Résultat concret

Après exécution, tu trouveras :

```
app/build/outputs/bundle/release/app-release.aab
```

- Ce fichier est optimisé pour Google Play.
- Google Play peut générer à partir de ce bundle :
  - APK universel
  - APKs fractionnés par appareil, ABI, densité, langue

2 Que fait Gradle derrière ?

`bundleRelease` n'est pas une tâche isolée. Elle déclenche un graphe de tasks. Voici les principales :

Task Gradle	Rôle
compileReleaseJavaWithJavac	Compile le code Java/Kotlin pour le build type Release
mergeReleaseResources	Fusionne toutes les ressources (images, layouts, strings)
processReleaseManifest	Prépare le manifeste Android
packageRelease	Prépare les fichiers pour le bundle
signReleaseBundle	Applique la signature avec le keystore de release
bundleRelease	Assemble enfin le .aab final

> Chaque task dépend des précédentes : si `compileReleaseJavaWithJavac` échoue, le bundle ne peut pas être généré.

Signer votre application

Facteurs à prendre en compte concernant la signature

Il est recommandé de signer une application avec le même certificat pendant toute sa durée de vie. Les raisons sont les suivantes :

Mises à jour

Lorsque le système installe une mise à jour d'application, il compare le ou les certificats de la nouvelle version à ceux de la version existante. Le système n'autorise la mise à jour que si les certificats correspondent. Si vous signez la nouvelle version avec un certificat différent, vous devez attribuer un nom de package différent à l'application. Dans ce cas, l'utilisateur installe cette nouvelle version comme une nouvelle application.

Modularité

Android autorise les APK signés par le même certificat à s'exécuter dans le même processus, si les applications le demandent, afin que le système les traite comme une seule et même application. Procéder ainsi vous permet de déployer votre application sous forme de modules que les utilisateurs peuvent mettre à jour indépendamment.

Autorisations de partager du code ou des données

Android fournit des autorisations basées sur la signature, afin qu'une application puisse exposer des fonctionnalités à une autre application signée avec un certificat spécifié. En signant plusieurs APK avec le même certificat et en utilisant les vérifications d'autorisation basées sur la signature, vos applications peuvent partager du code et des données de manière sécurisée.

Signature de l' application

Facteurs à prendre en compte concernant la signature

Sécuriser votre clé

La sécurisation de la clé de signature d'application est cruciale pour protéger votre identité de développeur et la confiance des utilisateurs. Si la clé est compromise, un tiers peut distribuer des applications malveillantes au nom de votre application, voler des données ou attaquer le système. La clé privée est indispensable pour toutes les mises à jour futures, et sa perte empêche la publication de nouvelles versions. Pour la protéger, il est recommandé de choisir des mots de passe forts, de ne jamais partager la clé ou le keystore, et de conserver le keystore en lieu sûr.

Supprimer les informations de signature de vos fichiers de compilation

Par défaut, Android Studio peut inclure vos informations de signature (keystore, mots de passe, alias) en texte clair dans les fichiers `build.gradle`.

Ceci représente un **risque de sécurité**, surtout si le code est partagé ou open source.

Il faut toujours ajouter le keystore dans .gitignore en cas d'utilisation de GIT.

Signature de l' application

Facteurs à prendre en compte concernant la signature

Comment supprimer les informations de signature de vos fichiers de compilation

Solution : utiliser un fichier séparé

- Créez un fichier `keystore.properties` à la racine du projet (ou ailleurs selon vos besoins CI/CD).
- Stockez-y vos informations de signature :

```
storePassword=myStorePassword
keyPassword=myKeyPassword
keyAlias=myKeyAlias
storeFile=myStoreFileLocation
```

Chargement des propriétés dans build.gradle

- Chargez `keystore.properties` dans votre `build.gradle` avant le bloc `android {}` :

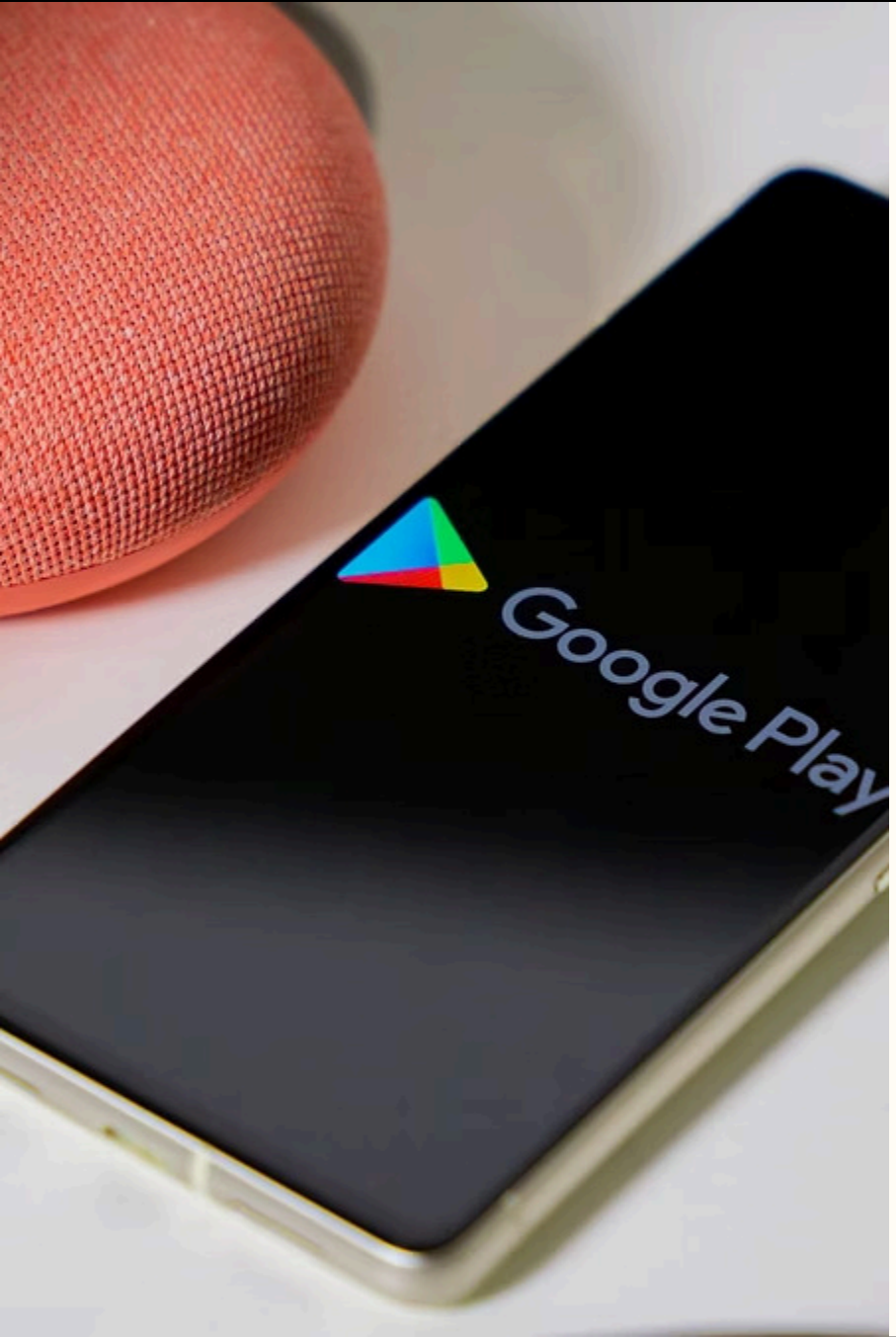
```
def keystorePropertiesFile = rootProject.file("keystore.properties")
def keystoreProperties = new Properties()
keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
```

- Référencez ensuite ces propriétés dans le bloc `signingConfigs` :

```
android {
    signingConfigs {
        config {
            keyAlias keystoreProperties['keyAlias']
            keyPassword keystoreProperties['keyPassword']
            storeFile file(keystoreProperties['storeFile'])
            storePassword keystoreProperties['storePassword']
        }
    }
}
```

Compilation sécurisée

- Sélectionnez le build variant et compilez via Build > Build Bundle(s)/APK(s).
- Les fichiers générés (`build/outputs/`) ne contiennent plus d’informations sensibles et peuvent être inclus dans un contrôle de code source.



Importer votre application dans la Play Console

Une fois que vous avez signé la version de votre application, vous devez l'importer sur Google Play pour l'inspecter, la tester et la publier.

Importer votre application dans la Play Console

Inspecter les APK à l'aide des dernières versions et bundles

Si vous importez votre application en tant qu'Android App Bundle, la Play Console génère automatiquement des APK divisés et des APK multiples pour toutes les configurations d'appareil compatibles avec votre application.

Dans la Play Console, vous pouvez utiliser la section "Derniers bundles" de la page "Dernières versions et bundles" pour afficher tous les artefacts APK générés par Google Play, inspecter des données telles que les appareils compatibles et les économies de taille des APK, et télécharger les APK générés pour les déployer et les tester localement.

Il faut s'enregistrer au service Signature d'application Play. Celui-ci est obligatoire depuis août 2021 pour importer et signer toutes les nouvelles applications.

Il faut s'assurer que votre application respecte les exigences de taille de Google Play. Google Play accepte une taille de téléchargement totale cumulée de 4 Go. Cette taille inclut tous les modules et packs d'éléments d'installation.

Importer votre application dans la Play Console

Mettre à jour votre app bundle

Pour mettre à jour votre application une fois importée dans la Play Console, vous devez augmenter le numéro de la version inclus dans le module de base, puis créer et importer un nouvel app bundle. Google Play génère ensuite les APK mis à jour avec les nouveaux codes de version et les distribue aux utilisateurs selon les besoins.

Importer votre application dans la Play Console

Préparer et déployer une version

Étape 1 : Créez une version

Une version est une combinaison d'une ou de plusieurs révisions d'application que vous préparez avant de déployer une application ou la mise à jour d'une application. Vous pouvez créer une version dans trois canaux de tests différents ou dans le canal de production :

Canal de tests ouverts :

les versions de tests ouverts sont accessibles aux testeurs sur Google Play. Les utilisateurs peuvent rejoindre vos tests depuis votre fiche Play Store.

Canal de tests fermés :

les versions de tests fermés sont accessibles à un nombre limité de testeurs que vous sélectionnez et qui testent une version préliminaire de votre application et vous font part de leurs commentaires.

Canal de tests internes :

les versions de tests internes sont accessibles aux testeurs de votre choix (100 maximum).

Canal de production :

les versions de production sont accessibles à tous les utilisateurs de Google Play dans les pays de votre choix.

Importer votre application dans la Play Console

Préparer et déployer une version

Étape 2 : Préparez la version de votre application

Une version est une combinaison d'une ou de plusieurs révisions d'application que vous préparez avant de déployer une application ou la mise à jour d'une application. Vous pouvez créer une version dans trois canaux de tests différents ou dans le canal de production :

Utiliser le service Signature d'application Play

Ajoutez vos app bundles

Attribuez un nom à la version et Saisissez les notes de version

Avec le service Signature d'application Play, Google gère et protège à votre place votre clé de signature d'application, et l'utilise pour signer des APK de distribution optimisés et générés à partir de vos app bundles. Ce service stocke votre clé de signature d'application sur l'infrastructure sécurisée de Google et propose des options de mise à niveau pour renforcer la sécurité.

Importer votre application dans la Play Console

Préparer et déployer une version

Étape 3 : Vérifiez et déployez votre version

Préparer les conditions de publication

Avant le déploiement, il faut compléter la fiche Play Store, renseigner le contenu de l'application (confidentialité, accès aux données) et définir le prix. Sans ces éléments, la version ne peut pas être soumise à l'examen ni publiée.

Vérifier et finaliser la version à déployer

Dans la Play Console, sélectionnez le canal de test (ou production), puis vérifiez la version brouillon. L'écran *Prévisualiser et confirmer* permet de détecter les erreurs bloquantes, qui doivent être corrigées avant toute publication

Gérer l'examen et le déploiement progressif

Selon le type de modification, vous pouvez soit enregistrer les changements pour les soumettre plus tard, soit publier immédiatement. Pour une mise à jour, il est possible de choisir un pourcentage d'utilisateurs afin de réaliser un déploiement progressif.

Lancer le déploiement auprès des utilisateurs

Une fois toutes les vérifications effectuées, cliquez sur *Lancer le déploiement*. Pour une première publication en production, l'application devient accessible à tous les utilisateurs Google Play dans les pays sélectionnés après validation.

Importer votre application dans la Play Console

Préparer et déployer une version

Étape 4 : Vérifiez les informations de la version

Identification et état de la version

Cette section indique le nom de la version, le canal de déploiement (production, test ouvert, test fermé) et son état actuel (brouillon, en cours, publié, suspendu).
Elle permet de savoir rapidement où en est chaque version de l'application.

Informations de déploiement

On y trouve la date de la dernière mise à jour ainsi que les pays ou régions ciblés par le déploiement. Cela aide à vérifier la disponibilité géographique de l'application et à gérer des déploiements progressifs ou régionaux.

Contenu technique de la version

Cette partie liste les App Bundles (AAB) et APK associés à la version : nouveaux fichiers, fichiers conservés ou désactivés.
Elle permet de contrôler exactement quels artefacts Android sont utilisés par Google Play.

Suivi et historique de la version

On accède ici à la présentation de la version (installations, mises à jour, performances), aux notes de version, et à l'historique de déploiement.
Ces données servent à analyser l'impact de la version et à suivre toutes les actions effectuées (diffusion, pause, reprise).

Canaux de distribution et stores Android

Il existe plusieurs options pour distribuer les applications Android, allant de la publication sur le Play Store aux stores alternatifs.

Canaux de distribution et stores Android

En dehors du Google Play Store, il existe plusieurs autres stores Android, chacun ayant ses propres procédures de publication et règles.

1

Samsung Galaxy Store

- Store officiel des appareils Samsung
- Nécessite un compte développeur Samsung
- Upload direct des APK signés via le portail Samsung Developer
- Supporte tests internes et versions publiques
- Parfois des optimisations spécifiques pour les appareils Samsung

2

Huawei AppGallery

- Store officiel pour les appareils Huawei
- Nécessite un compte développeur Huawei
- Upload d'APK ou App Bundle signé
- Intégration avec Huawei Mobile Services (HMS) si nécessaire
- Possibilité de déploiement par étapes et tests beta

3

Autres stores alternatifs

- Amazon Appstore : pour tablettes Fire et smartphones Amazon
- Xiaomi GetApps, Oppo App Market, Vivo App Store : stores régionaux ou fabricants
- Chaque store demande :
  - Compte développeur dédié
  - Signature APK spécifique
  - Soumission via leur portail
- Certains ont des règles de contenu et sécurité différentes du Play Store

Points communs

- Tous exigent que l'application soit signée
- Tous permettent des tests internes avant publication
- Les APK/AAB doivent respecter les guidelines de sécurité et compatibilité
- La Play Console n'est pas utilisée, chaque store a son propre portail

Canaux de distribution et stores Android

Importer votre application dans la Play Console n'est pas la seule option, mais c'est la méthode officielle et recommandée pour la publication sur Google Play.

1

Play Console (méthode officielle)

- Importer un **AAB ou APK signé**
- **Configurer fiche Play Store, canaux de test ou production**
- Obligatoire pour la publication publique

2

Firebase App Distribution

- Permet de distribuer des builds signés (APK ou AAB) à des testeurs internes ou externes
- Idéal pour tests alpha/bêta rapides avant publication
- Ne publie pas sur le Play Store

3

Distribution manuelle (sideloading)

- Installation directe de l'APK sur un appareil via USB ou téléchargement
- Utile pour tests internes ou démonstrations
- Non recommandée pour un déploiement public (sécurité limitée, absence de mise à jour automatique)

4

CI/CD + Play Developer API

- Automatisation de l'import et du déploiement depuis un serveur (GitHub Actions, GitLab CI, Jenkins)
- Permet de uploader directement un AAB signé sur la Play Console via API
- Toujours lié à la Play Console, mais sans interface manuelle

Pour la publication officielle sur Google Play, la Play Console est obligatoire. Les autres méthodes servent surtout aux tests internes, beta ou distribution hors Play Store.

Conclusion : Déploiement et Sécurisation d'Applications Android

Points clés à retenir

Préparation

Comprendre les distinctions entre les versions Debug et Release, la configuration adéquate, la gestion des logs et le versioning.

Signature

Maîtriser les Keystores, la gestion des clés et certificats, ainsi que les bonnes pratiques de sécurité via des commandes bash.

Publication

Utiliser efficacement la Play Console, préparer les App Bundles (AAB) et APK, et effectuer des tests internes rigoureux avant le lancement.

Déploiement

Gérer les différentes versions et canaux de déploiement, assurer un suivi précis des performances et planifier les mises à jour.

Processus complet

Le déploiement et la sécurisation d'applications Android sont des étapes fondamentales qui exigent une attention méticuleuse à chaque phase. D'une préparation rigoureuse de la version à la gestion stratégique de sa publication et de son déploiement via la Play Console, chaque détail compte. La signature sécurisée de l'application est un pilier central, garantissant l'intégrité et l'authenticité du code. En respectant ces pratiques, les développeurs peuvent non seulement offrir une expérience utilisateur fiable, mais aussi protéger leurs applications et leurs utilisateurs contre les menaces potentielles.