



СМР НА ОС АВРОПА

С нуля до альфы

Глазков Денис

Старший инженер-разработчик отдела разработки ОС

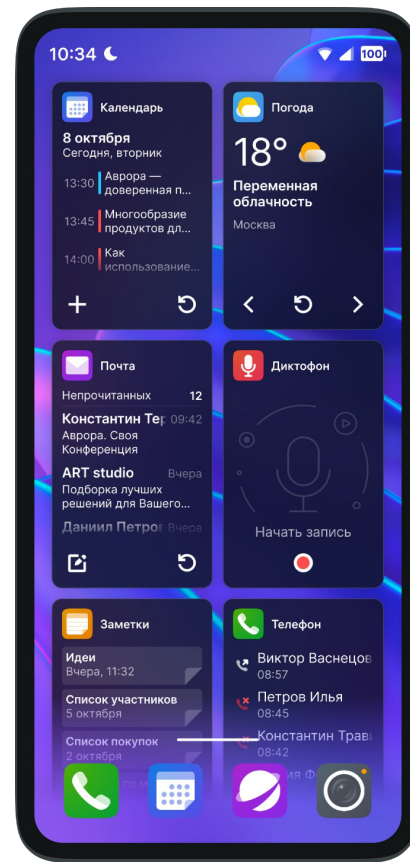


План доклада

1. Мобильная ОС Аврора
2. В предыдущих сериях ...
3. Архитектура SMP с точки зрения портирования
4. Разработка на SMP под ОС Аврора
5. Будущие планы

Мобильная операционная система «Аврора»

Отечественная мобильная операционная система, основанная на ядре Linux, с фокусом на **безопасность и эргономику**



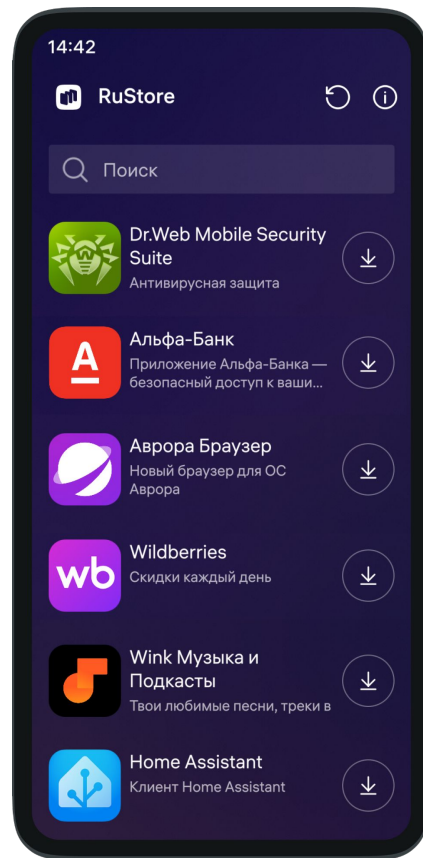
Экосистема приложений

С 2023 года ведется активная работа в развитие поддержки кроссплатформенных фреймворков на ОС Аврора.

Поддерживаемые фреймворки



В процессе адаптации



В предыдущих сериях ...

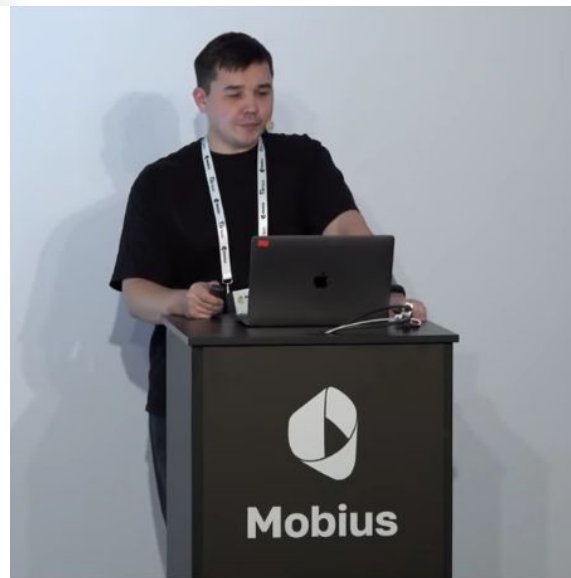
Mobius 2025

Текущий статус адаптации CMP

Демо приложение

На данный момент реализовано:

- Создание графического окна
- Сборку приложения под ОС Аврора
- Сборку итогового RPM пакета
- Запуск приложения в эмуляторе



Денис Глазков

Открытая Мобильная Платформа

А В Р О Р А | С В О Я С И С Т Е М А

А В Р О Р А | С В О Я С И С Т Е М А

Mobius 2025

Текущий статус адаптации CMP

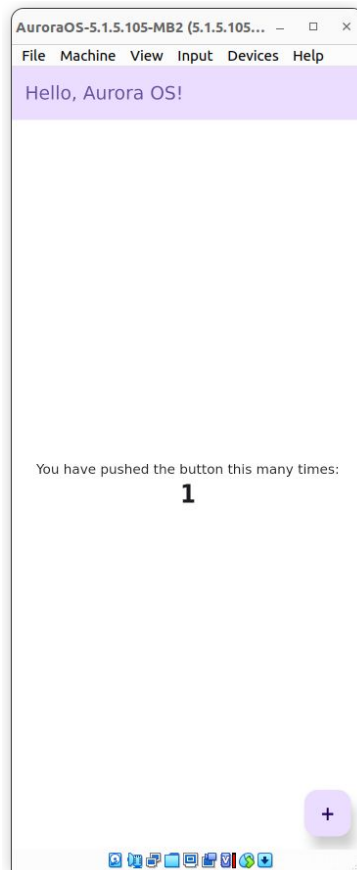
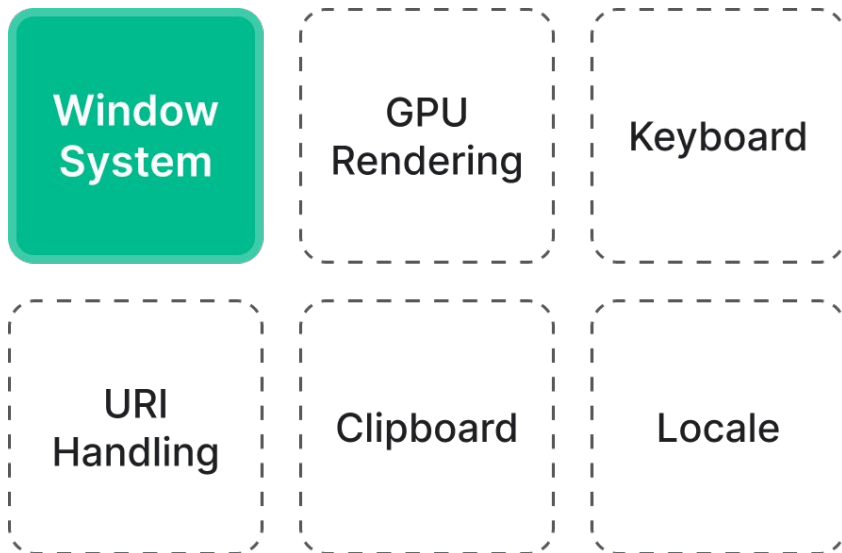
Демо приложение

На данный момент реализовано:

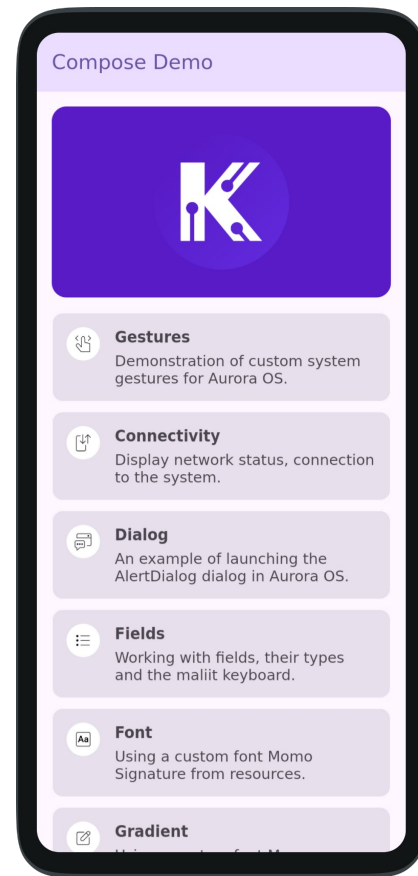
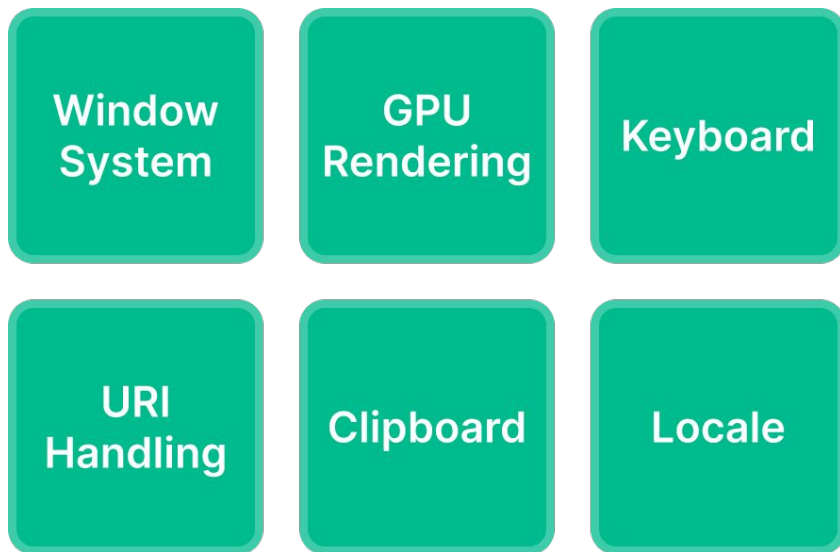
- Создание графического окна
- Сборку приложения под ОС Аврора
- Сборку итогового RPM пакета
- Запуск приложения в эмуляторе



Статус CMP на конец 2025 года



Статус CMP в середине 2026 года

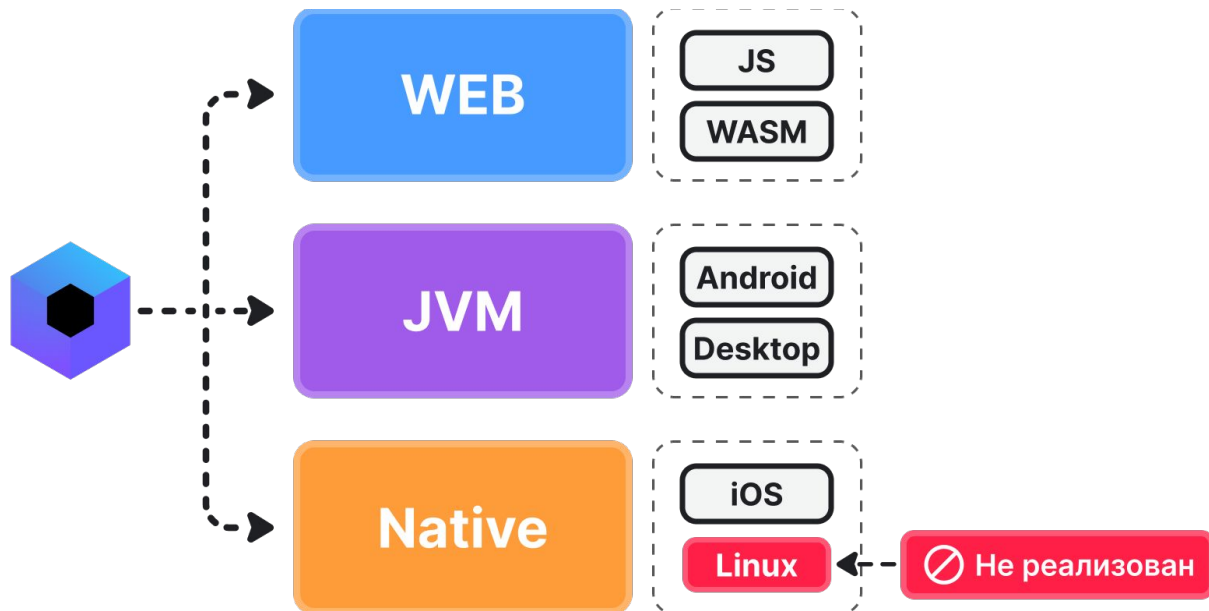




Источник: <https://imgur.com/gallery/this-is-fine-Jqtp4Bl>

Архитектура СМР с точки зрения портирования

СМР не поддерживает нативный Linux таргет из коробки



Три столпа CMP

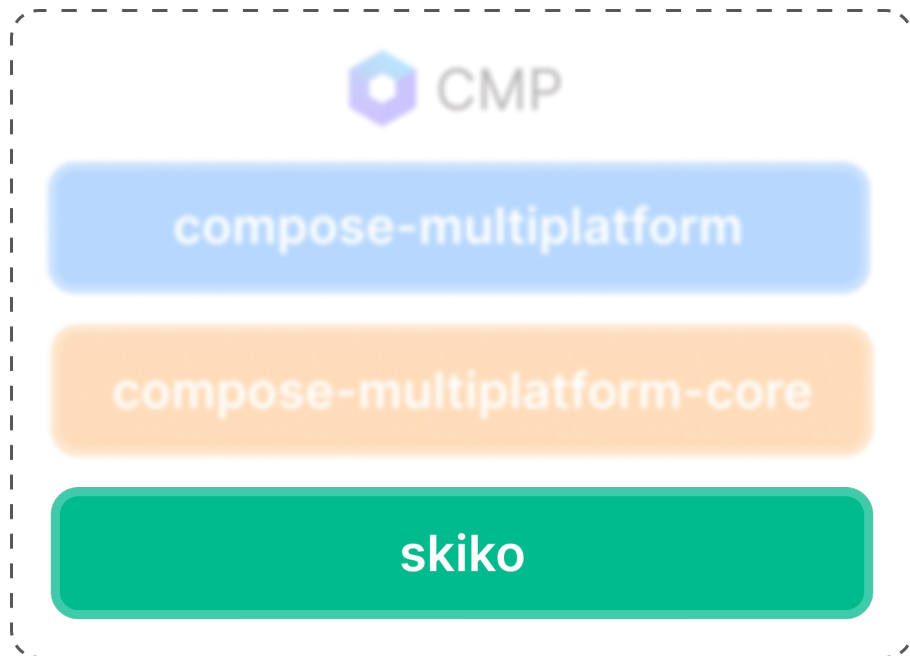


compose-multiplatform

compose-multiplatform-core

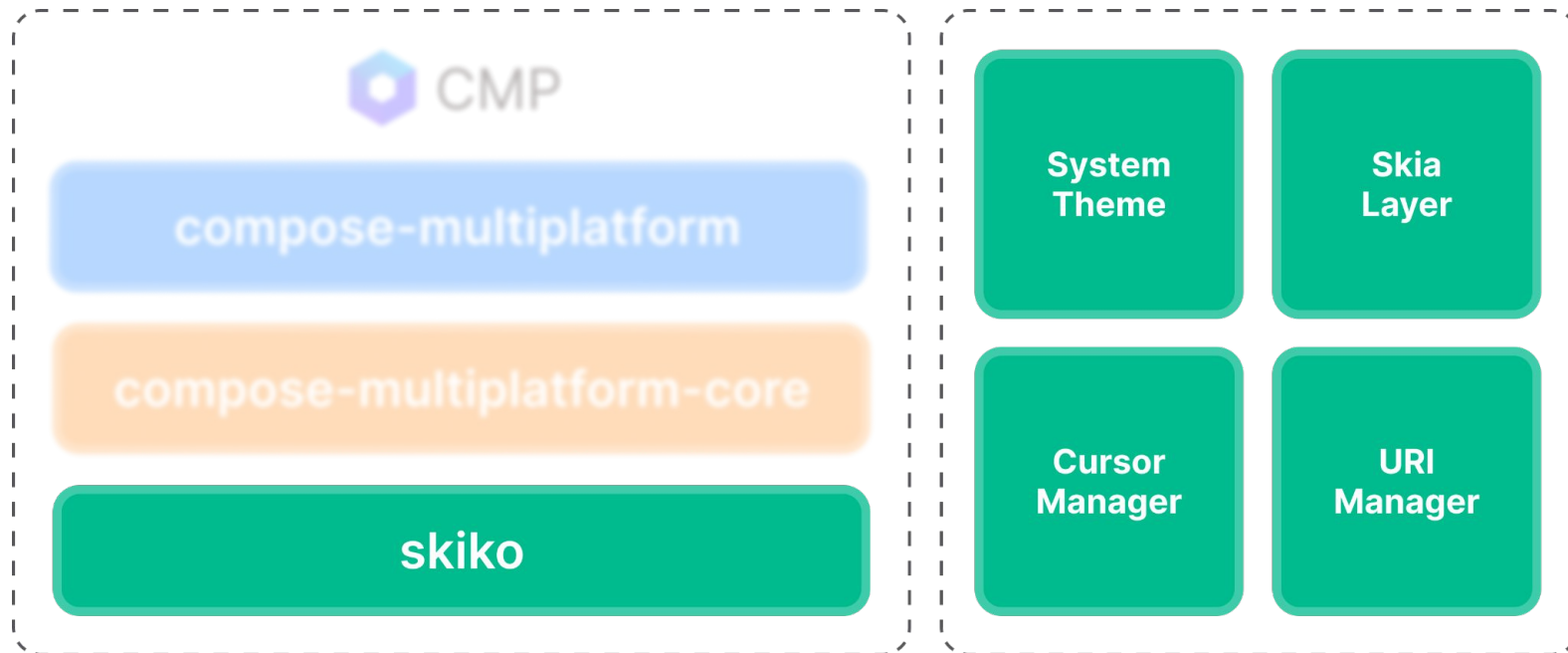
skiko

Библиотека skiko

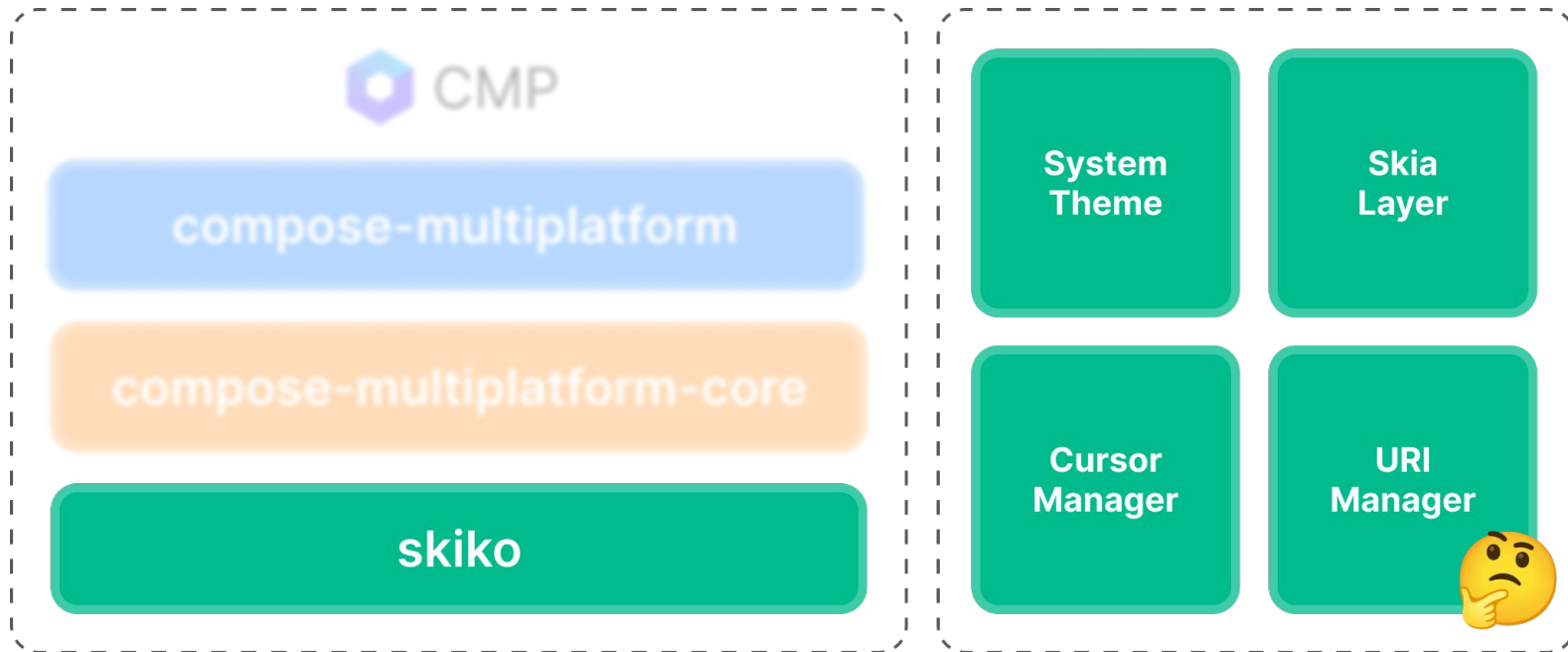


Библиотека,
предоставляющая
Kotlin биндинги для API
библиотеки 2D-графики **Skia**

Библиотека skiko



Библиотека skiko

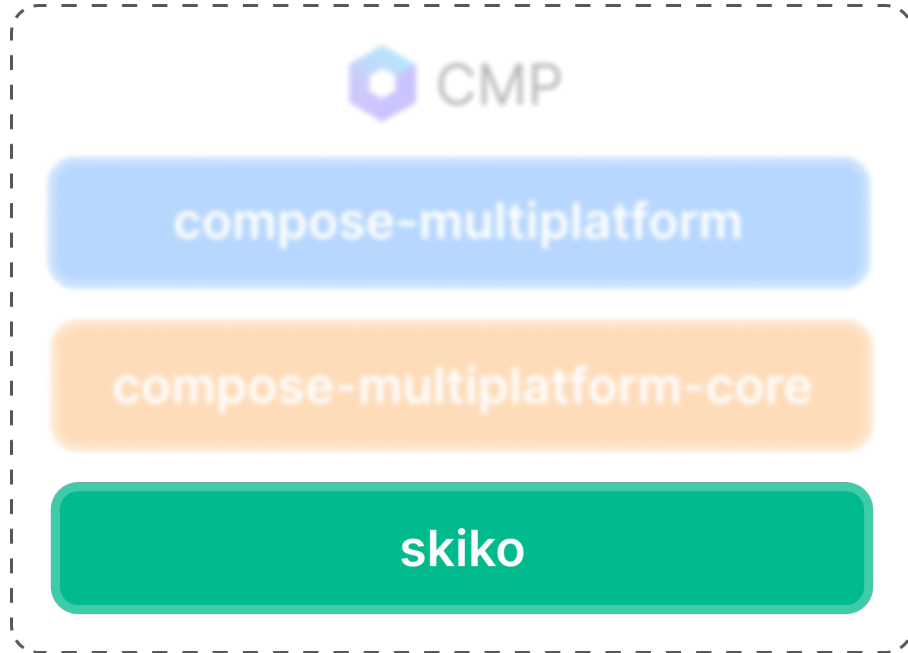


Библиотека skiko

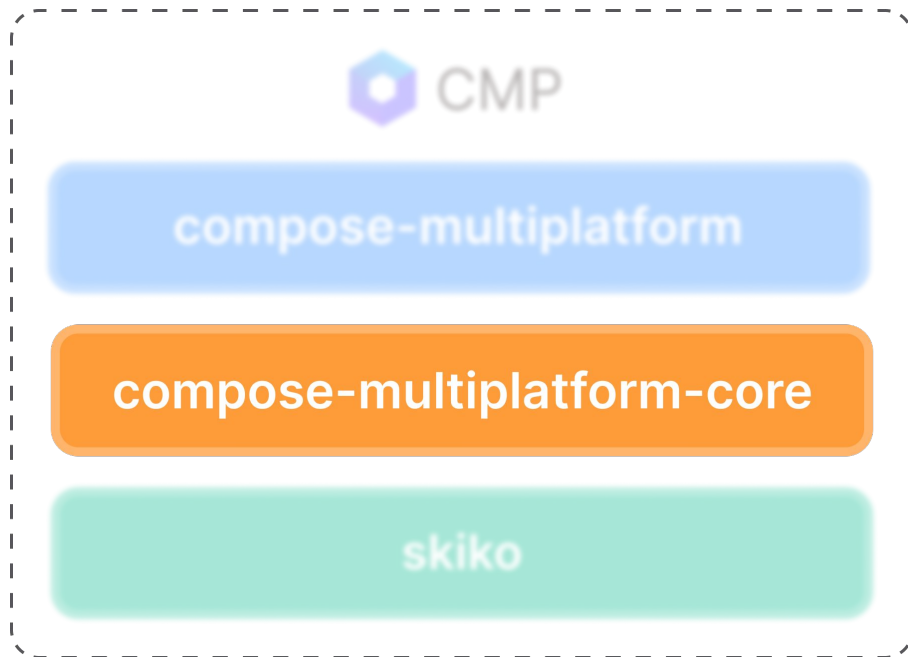
Интерфейс для реализации рендеринга на платформе

```
expect open class SkiaLayer {  
    var renderApi: GraphicsApi  
    val contentScale: Float  
    val pixelGeometry: PixelGeometry  
    var fullscreen: Boolean  
    var transparency: Boolean  
    val component: Any?  
    var renderDelegate: SkikoRenderDelegate?  
    fun attachTo(container: Any)  
    fun detach()  
    fun needRedraw()  
    internal fun draw(canvas: Canvas)  
}
```

Библиотека skiko

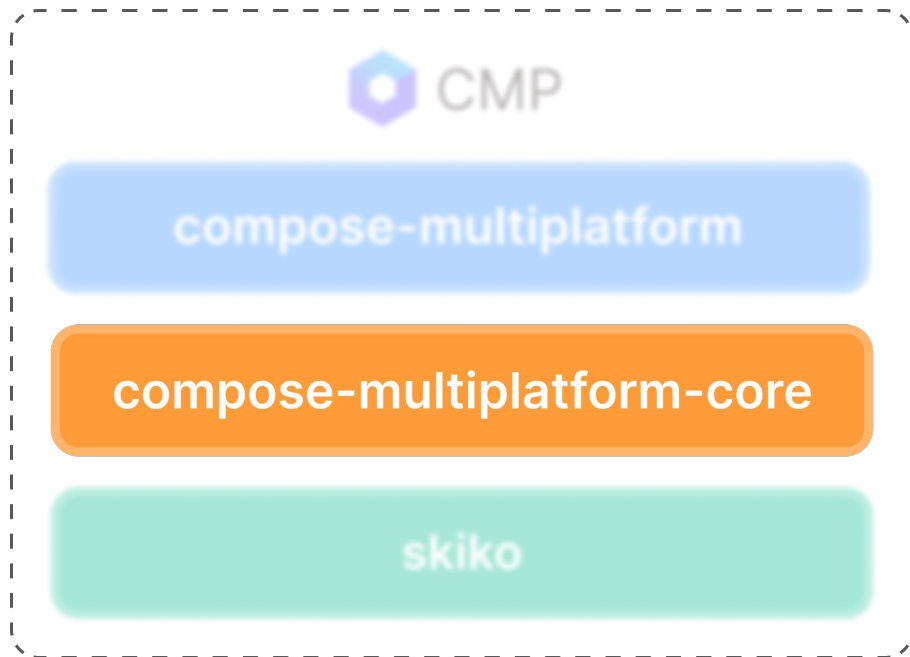


Набор библиотек compose-multiplatform-core



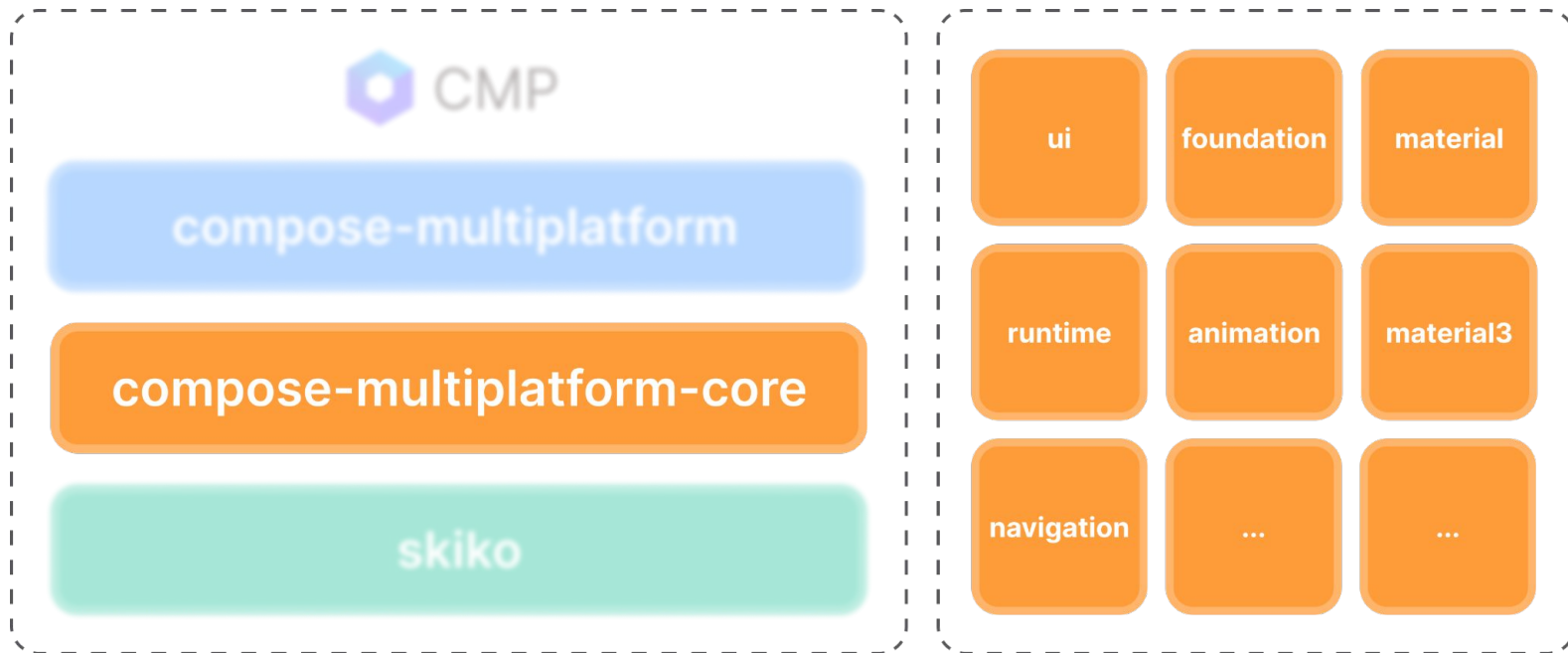
Набор CMP библиотек реализующий **основной функционал фреймворка CMP** (ui, runtime, material, ...)

Набор библиотек compose-multiplatform-core



Огромная монорепа, которая индексируется пол часа, из-за которой мне пришлось купить 64 гб оперативки, и в которой ничего не понятно, особенно как добавить новый таргет 😁

Набор библиотек compose-multiplatform-core



Набор библиотек `compose-multiplatform-core`

Создание окна и настройка рендеринга для платформы

```
fun Window(content: @Composable () → Unit) {  
    private val layer = SkiaLayer()  
    private val scene = CanvasLayersComposeScene(invalidate = layer::needRedraw)  
  
    init {  
        layer.renderDelegate = object : SkikoRenderDelegate {  
            override fun onRender(canvas: Canvas, w: Int, h: Int, time: Long) {  
                scene.render(canvas.asComposeCanvas(), time)  
            }  
        }  
        scene.setContent { content() }  
    }  
}
```

Набор библиотек `compose-multiplatform-core`

Интеграция с виртуальной клавиатурой платформы

```
internal class AuroraTextInputService : PlatformTextInputService {  
    override fun showSoftwareKeyboard(ime: ImeOptions) {  
        when(ime.keyboardType) {  
            KeyboardType.Text    → AuroraKeyboard.openType(AuroraKeyboardType.Text)  
            KeyboardType.Number → AuroraKeyboard.openType(AuroraKeyboardType.Number)  
            else                  → AuroraKeyboard.open()  
        }  
    }  
}  
  
override fun hideSoftwareKeyboard() {  
    AuroraKeyboard.close()  
}  
}
```

Набор библиотек `compose-multiplatform-core`

Конфигурация эффекта скролла для платформы

```
class AuroraOverscrollEffect() : OverscrollEffect {  
    override val isInProgress: Boolean  
    override fun applyToScroll(...): Offset { ... }  
    override suspend fun applyToFling(...) { ... }  
}
```

Набор библиотек `compose-multiplatform-core`

Реализация открытия URI для платформы

```
class AuroraUriHandler() : UriHandler {  
    override fun openUri(uri: String) {  
        if (!UriLauncher.open(uri)) {  
            throw IllegalArgumentException("Can't open $uri.")  
        }  
    }  
}
```

Набор библиотек `compose-multiplatform-core`

И много интересных флагов, разбросанных по проекту ...

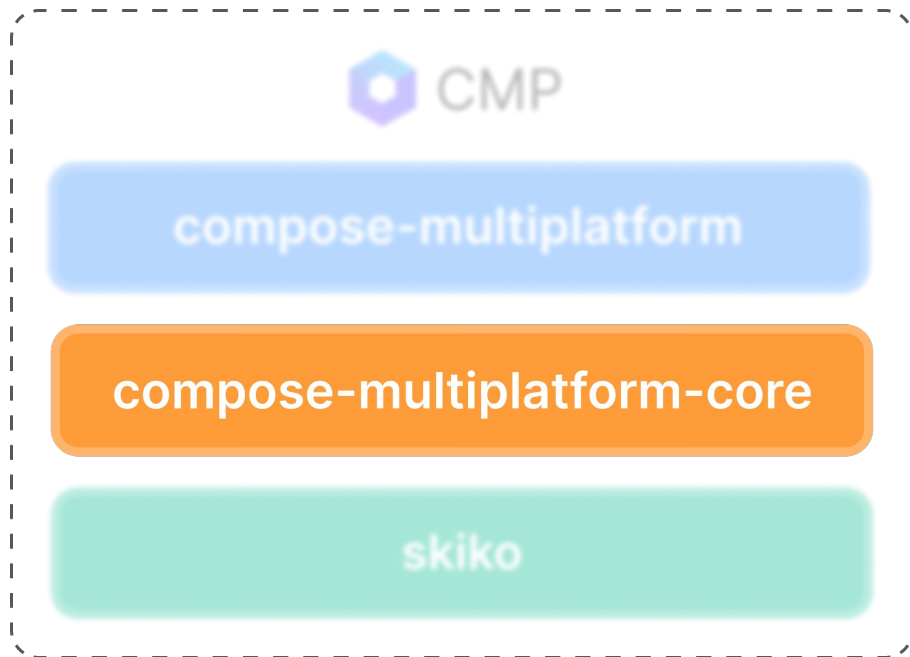
```
/**  
 * This is a temporary workaround and should be removed  
 * after proper mouse handling is settled (b/171402426).  
 */  
internal actual val isInTouchMode: Boolean = true
```

Набор библиотек `compose-multiplatform-core`

И много интересных флагов, разбросанных по проекту ...

```
// TODO https://youtrack.jetbrains.com/issue/CMP-5814/ ...  
internal actual fun isRequestFocusOnClickEnabled(): Boolean = false
```

Набор библиотек compose-multiplatform-core



Проект compose-multiplatform



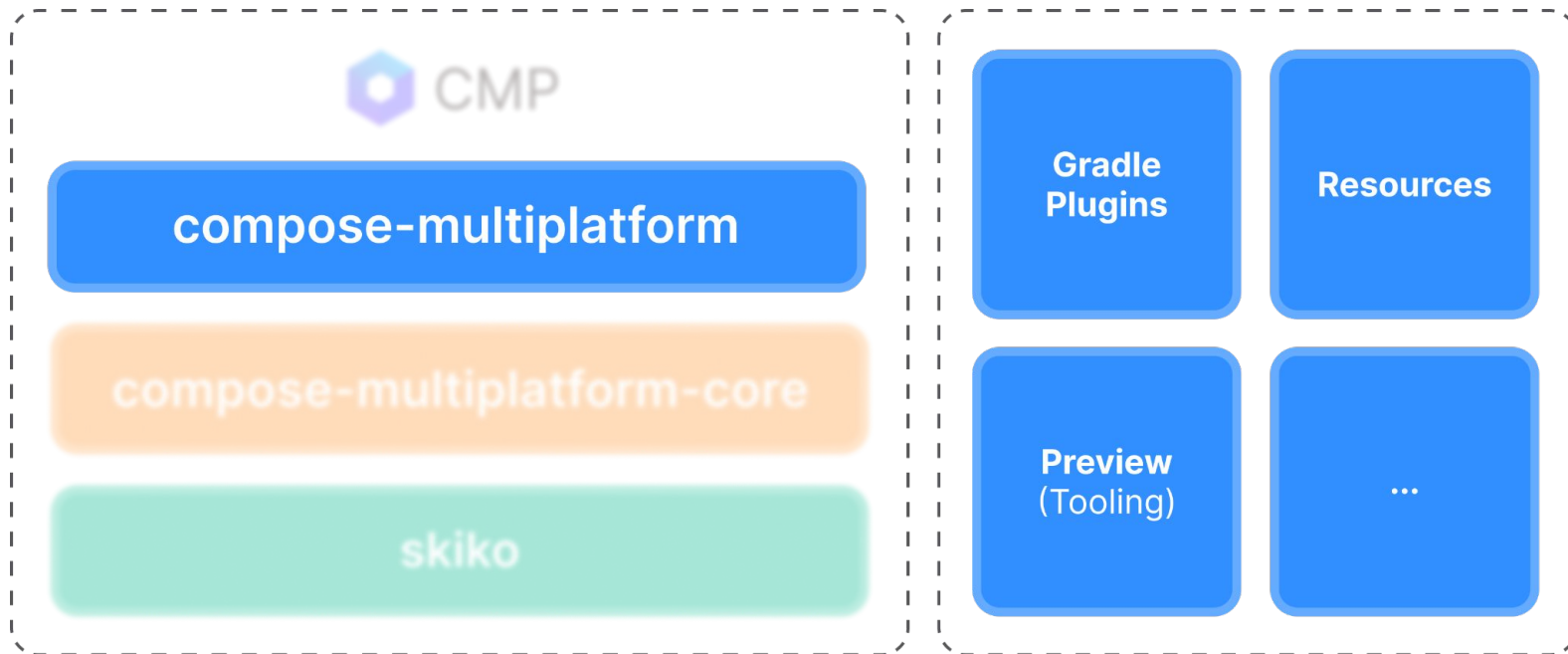
compose-multiplatform

compose-multiplatform-core

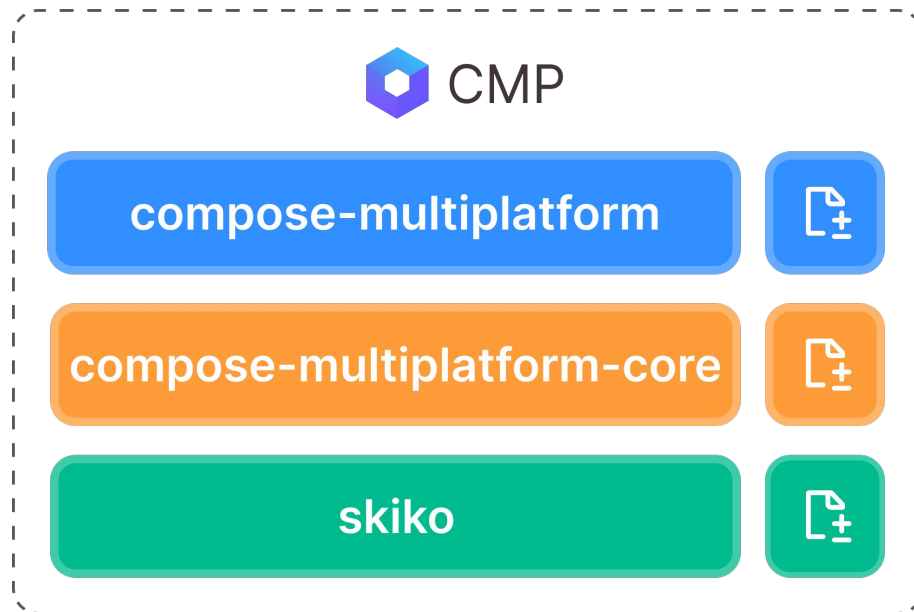
skiko

Gradle плагины и тулинг для удобства сборки проектов и разработки на CMP

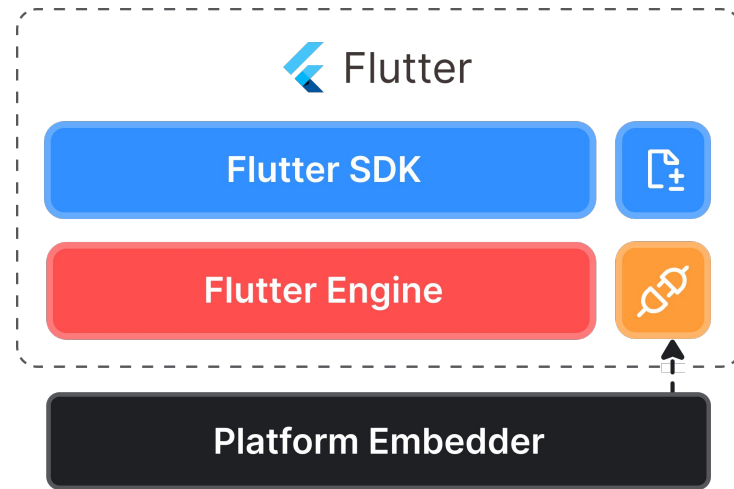
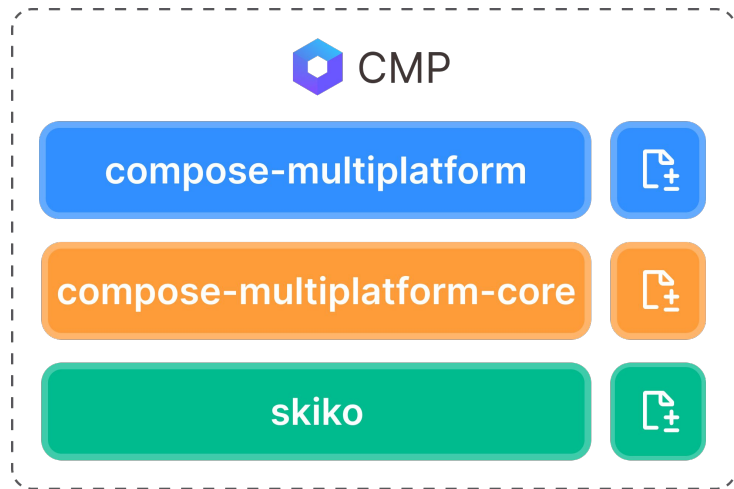
Проект compose-multiplatform



Интеграция новой платформы



Интеграция новой платформы





Альфа версия СМР для ОС Аврора

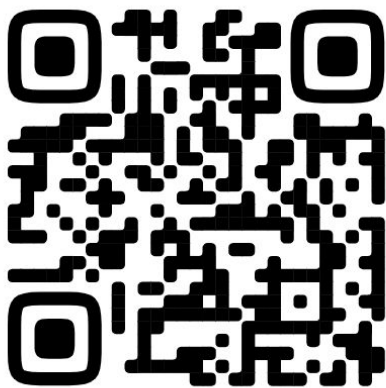


Альфа версия SMP для ОС Аврора





Бета программа



Aurora Developers



Аврора SDK

Альфа версия 0.0.4

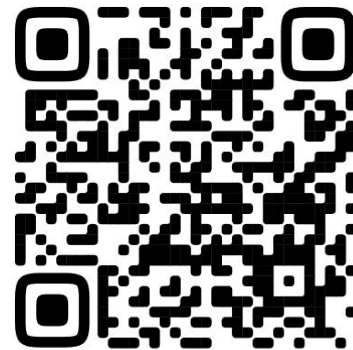
Аврора **Multiplatform**

Делитесь кодом на ваших условиях

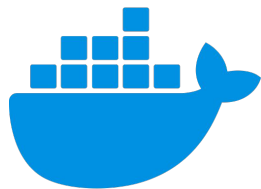
Ваш Kotlin Multiplatform / Compose Multiplatform проект теперь работает и на ОС Аврора.
Используйте стандартный KMP/СMP, а мы обеспечим совместимость с платформой Аврора.

Быстрый старт

Документация



Документация



Шаг 1

Установка Docker Desktop



<https://docs.docker.com/get-started/get-docker>

Install Docker Desktop on Mac

Copy as Markdown



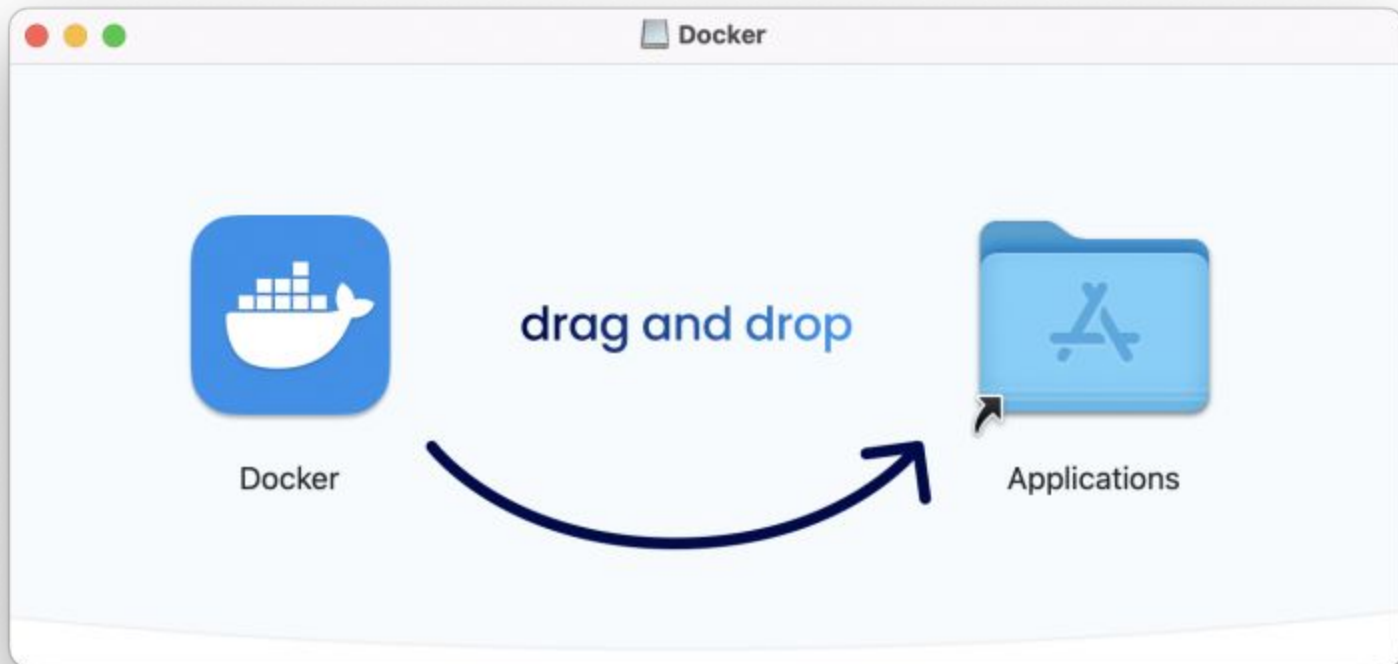
Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees or more than \$10 million USD in annual revenue) requires a [paid subscription](#).

This page provides download links, system requirements, and step-by-step installation instructions for Docker Desktop on Mac.

Docker Desktop for Mac with Apple silicon

Docker Desktop for Mac with Intel chip





Шаг 2

Установка Аврора SDK (BT)



Шаг 2

Установка Аврора SDK (MB2)



Шаг 2

Установка Аврора SDK (BT)



https://developer.auroraos.ru/downloads/archive/sdk_bt



Select Components

Please select the components you want to install.

Default

Select All

Deselect All

▼ Aurora SDK

Aurora SDK

- Aurora IDE
 - Aurora IDE Tools
- Build tools
- Documentation
- Emulators
- Examples
- Maintenance tool
 - Qt Linguist
 - Qt QmlLive

This component will occupy approximately 10.80 GB on your hard disk drive.

< Back

Next >

Cancel

Installing Aurora SDK



 6%

Installing component Aurora IDE

Show Details

< Back

Install

Cancel



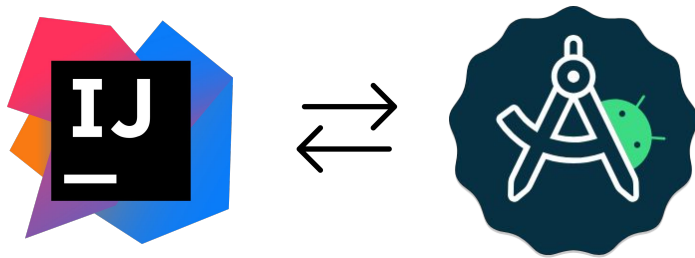
Шаг 3

Подготовка Maven Local



<https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-maven>

```
1 # Download
2 $ wget -O aurora-maven.tar.gz https://hub.mos.ru/auroraos/kotlin-multiplatform/aurora-
  maven/-/archive/aurora-0.0.4/aurora-maven-aurora-0.0.4.tar.gz
3
4 # Extract
5 $ tar -xzf aurora-maven.tar.gz
6 $ mkdir -p ~/.m2/
7 $ mv aurora-maven-* ~/.m2/repository
```



Для разработки можно использовать
любимую IDE



СМР для ОС Аврора подходит только для
разработки под ОС Аврора



Процесс разработки СМР приложения с
единственным таргетом - ОС Аврора

Шаг 4. Создание нового проекта

Search

New Project

- Java
- Kotlin
- Groovy
- Empty Project

Generators

- Maven Archetype
- Spring Boot
- JavaFX
- Ktor
- Kotlin Multiplatform**
- HTML
- Android
- Vite
- Multi-module Project

Name:

Location:

Project will be created in: ~/Documents/cmp.aurora.example

Create Git repository

Group:

Artifact:

JDK:

Android
With Compose Multiplatform UI framework based on Jetpack Compose

iOS

UI Implementation

With Compose Multiplatform UI framework, or use SwiftUI

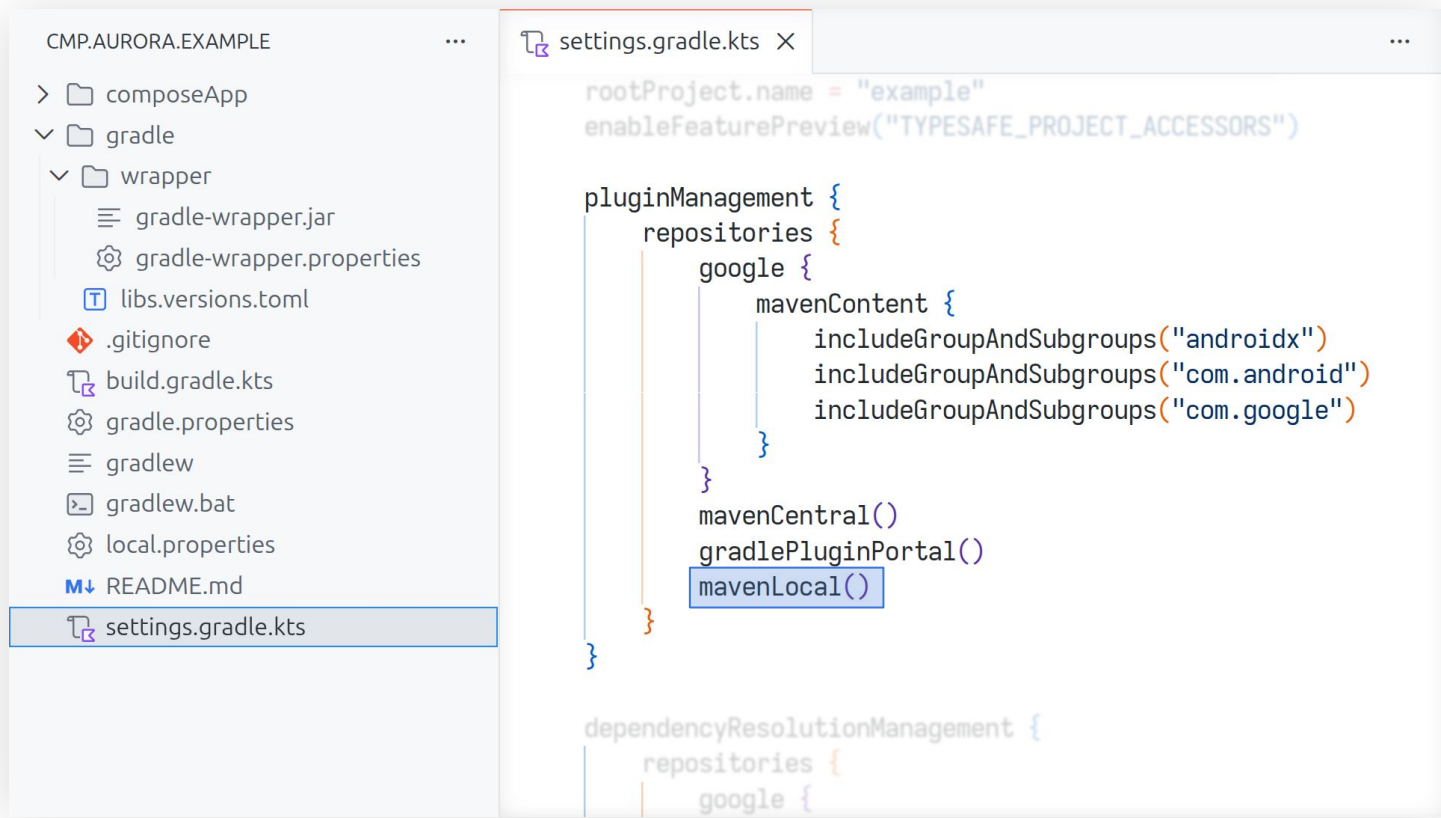
Desktop
With Compose Multiplatform UI framework

Web

UI Implementation

With Compose Multiplatform UI framework, or use React

Шаг 5. Подключение Maven Local



The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows the project structure for CMP.AURORA.EXAMPLE, with the settings.gradle.kts file selected. The code editor displays the contents of settings.gradle.kts, where the mavenLocal() method is highlighted in a blue box.

```
rootProject.name = "example"
enableFeaturePreview("TYPESAFE_PROJECT_ACCESSORS")

pluginManagement {
    repositories {
        google {
            mavenContent {
                includeGroupAndSubgroups("androidx")
                includeGroupAndSubgroups("com.android")
                includeGroupAndSubgroups("com.google")
            }
        }
        mavenCentral()
        gradlePluginPortal()
        mavenLocal()
    }
}

dependencyResolutionManagement {
    repositories {
        google {
```

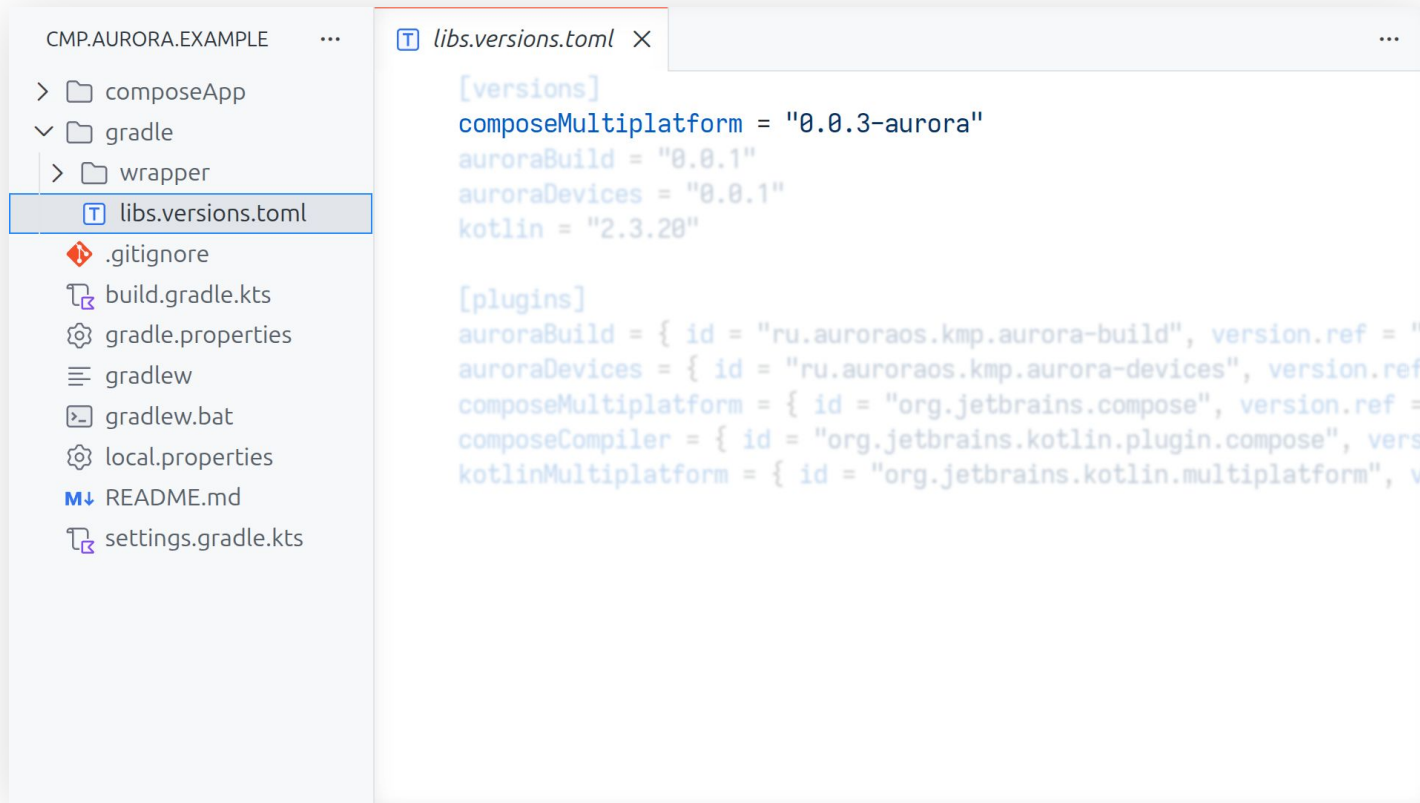
Шаг 5. Подключение Maven Local

The screenshot shows an IDE window with a project named CMP.AURORA.EXAMPLE. The left sidebar displays the project structure, with the `settings.gradle.kts` file selected. The main editor displays the content of `settings.gradle.kts`, showing the configuration for `dependencyResolutionManagement`. The `mavenLocal()` method is highlighted with a blue box, indicating its addition to the `repositories` list.

```
dependencyResolutionManagement {  
    repositories {  
        google {  
            mavenContent {  
                includeGroupAndSubgroups("androidx")  
                includeGroupAndSubgroups("com.android")  
                includeGroupAndSubgroups("com.google")  
            }  
            mavenLocal()  
            mavenCentral()  
        }  
    }  
}
```

Below the `dependencyResolutionManagement` block, the `include(":composeApp")` line is visible.

Шаг 6. Изменение версии CMP

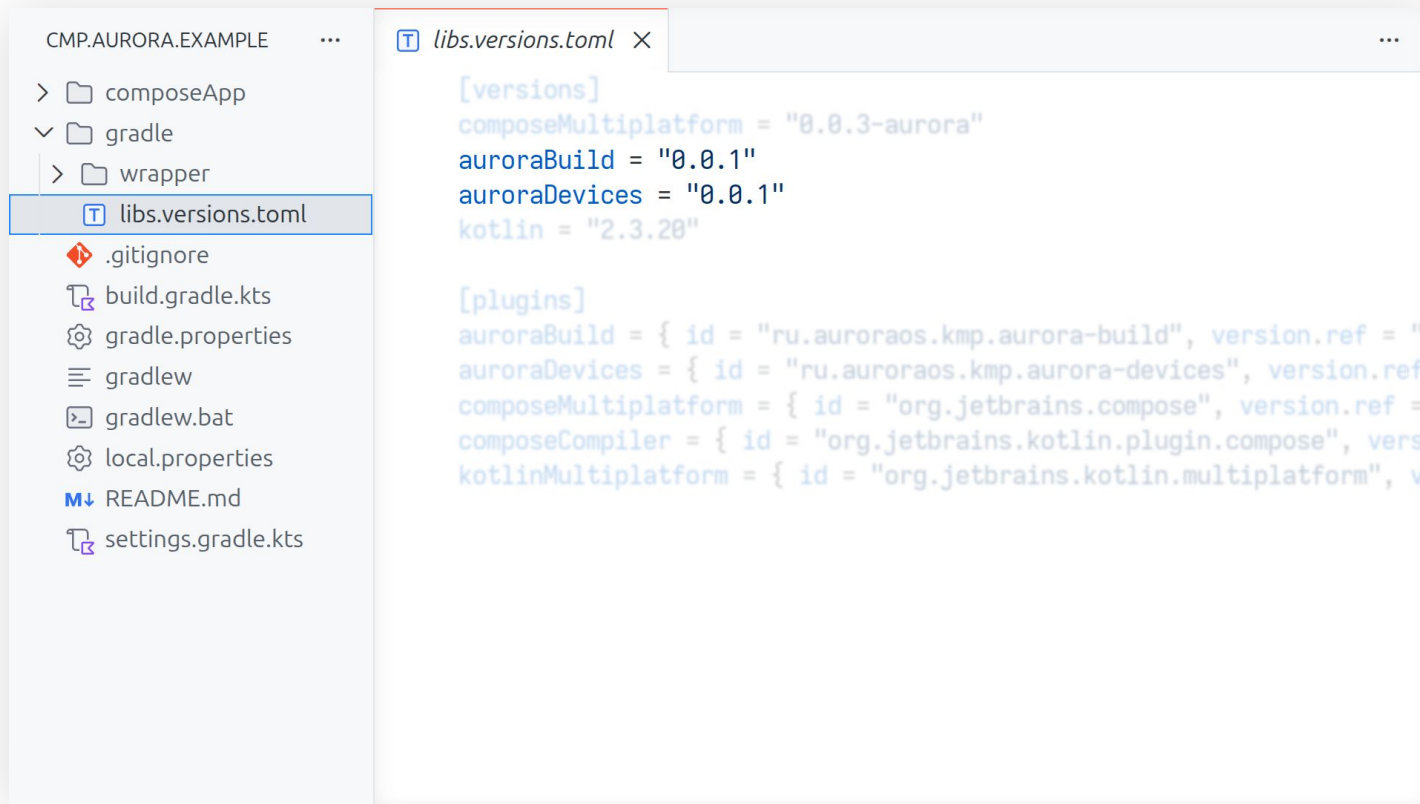


```
libs.versions.toml

[versions]
composeMultiplatform = "0.0.3-aurora"
auroraBuild = "0.0.1"
auroraDevices = "0.0.1"
kotlin = "2.3.20"

[plugins]
auroraBuild = { id = "ru.auroraos.kmp.aurora-build", version.ref = "auroraBuild" }
auroraDevices = { id = "ru.auroraos.kmp.aurora-devices", version.ref = "auroraDevices" }
composeMultiplatform = { id = "org.jetbrains.compose", version.ref = "composeMultiplatform" }
composeCompiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
kotlinMultiplatform = { id = "org.jetbrains.kotlin.multiplatform", version.ref = "kotlin" }
```

Шаг 7. Добавление плагинов для ОС Аврора

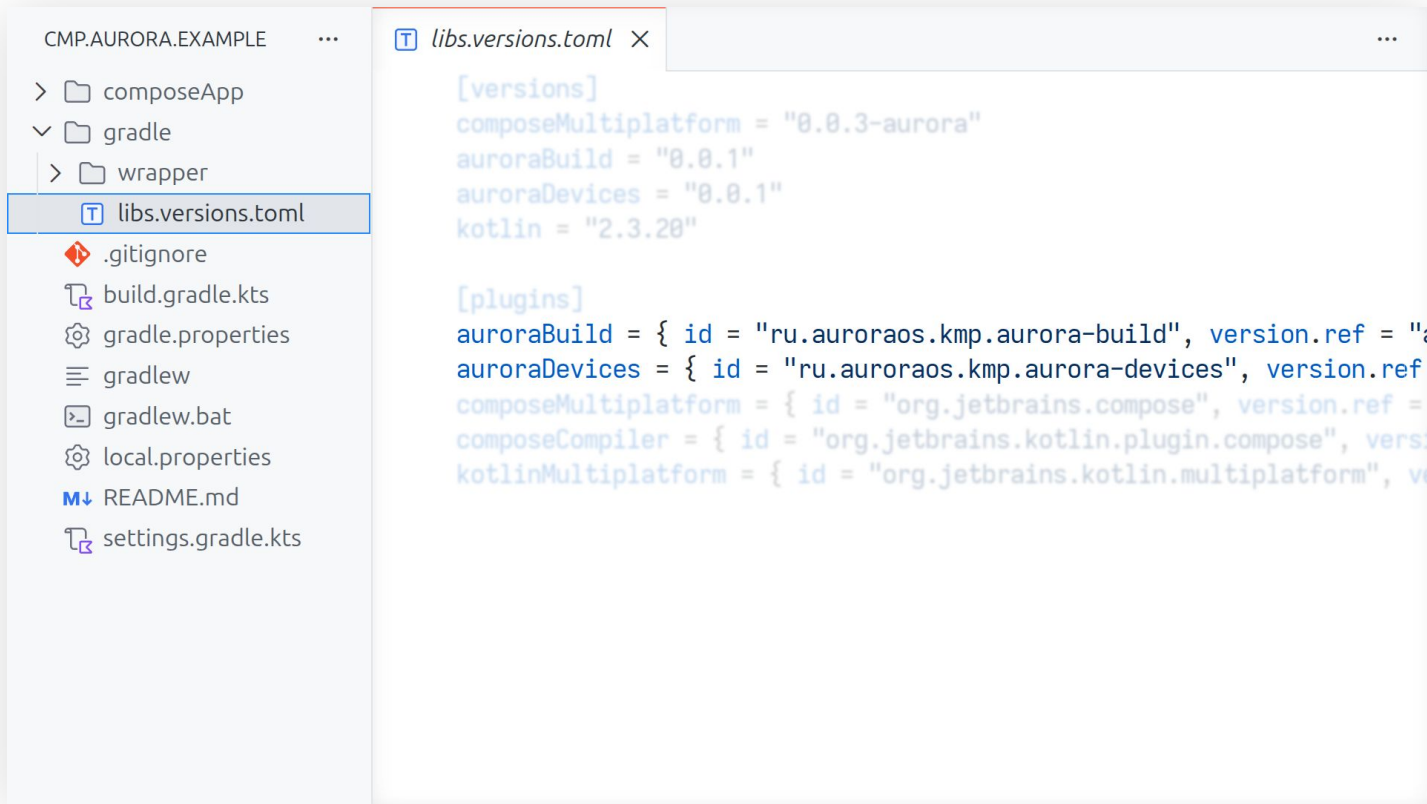


```
lib: CMP.AURORA.EXAMPLE ...
├── composeApp
├── gradle
├── wrapper
├── libs.versions.toml
├── .gitignore
├── build.gradle.kts
├── gradle.properties
├── gradlew
├── gradlew.bat
├── local.properties
├── README.md
├── settings.gradle.kts

[versions]
composeMultiplatform = "0.0.3-aurora"
auroraBuild = "0.0.1"
auroraDevices = "0.0.1"
kotlin = "2.3.20"

[plugins]
auroraBuild = { id = "ru.auroraos.kmp.aurora-build", version.ref = "auroraBuild" }
auroraDevices = { id = "ru.auroraos.kmp.aurora-devices", version.ref = "auroraDevices" }
composeMultiplatform = { id = "org.jetbrains.compose", version.ref = "composeMultiplatform" }
composeCompiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
kotlinMultiplatform = { id = "org.jetbrains.kotlin.multiplatform", version.ref = "kotlin" }
```

Шаг 7. Добавление плагинов для ОС Аврора

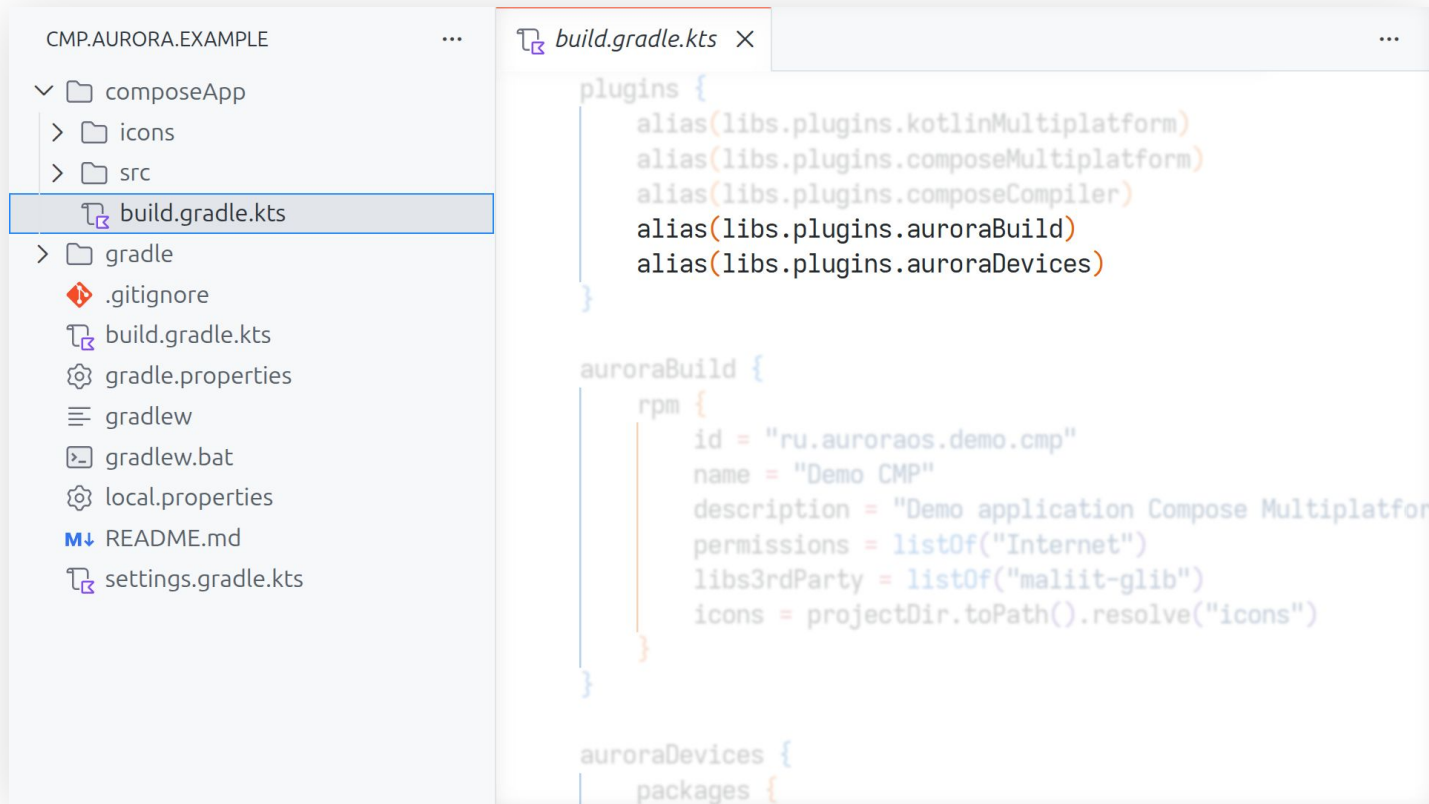


The screenshot shows an IDE interface with a file explorer on the left and a code editor on the right. The file explorer shows the project structure for CMP.AURORA.EXAMPLE, with the file `libs.versions.toml` selected. The code editor displays the following content:

```
[versions]
composeMultiplatform = "0.0.3-aurora"
auroraBuild = "0.0.1"
auroraDevices = "0.0.1"
kotlin = "2.3.20"

[plugins]
auroraBuild = { id = "ru.auroraos.kmp.aurora-build", version.ref = "auroraBuild" }
auroraDevices = { id = "ru.auroraos.kmp.aurora-devices", version.ref = "auroraDevices" }
composeMultiplatform = { id = "org.jetbrains.compose", version.ref = "composeMultiplatform" }
composeCompiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
kotlinMultiplatform = { id = "org.jetbrains.kotlin.multiplatform", version.ref = "kotlin" }
```

Шаг 7. Добавление плагинов для ОС Аврора



The screenshot shows an IDE window for a project named CMP.AURORA.EXAMPLE. The left sidebar displays a file tree with the following structure:

- composeApp
 - icons
 - src
 - build.gradle.kts (selected)
- gradle
- .gitignore
- build.gradle.kts
- gradle.properties
- gradlew
- gradlew.bat
- local.properties
- README.md
- settings.gradle.kts

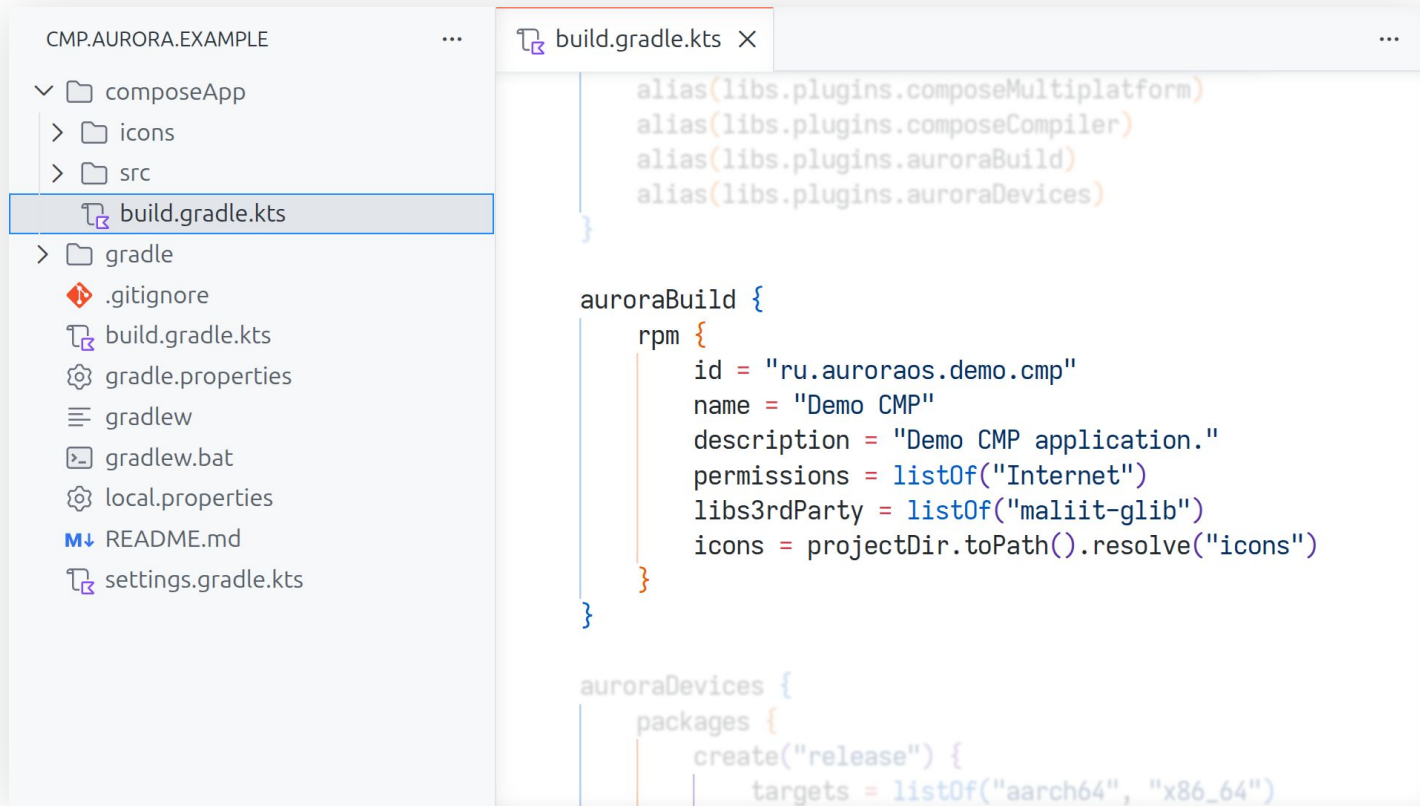
The main editor displays the content of the selected `build.gradle.kts` file:

```
plugins {  
    alias(libs.plugins.kotlinMultiplatform)  
    alias(libs.plugins.composeMultiplatform)  
    alias(libs.plugins.composeCompiler)  
    alias(libs.plugins.auroraBuild)  
    alias(libs.plugins.auroraDevices)  
}
```

```
auroraBuild {  
    rpm {  
        id = "ru.auroraos.demo.cmp"  
        name = "Demo CMP"  
        description = "Demo application Compose Multiplatform"  
        permissions = listOf("Internet")  
        libs3rdParty = listOf("maliit-glib")  
        icons = projectDir.toPath().resolve("icons")  
    }  
}
```

```
auroraDevices {  
    packages {
```

Шаг 8. Конфигурация сборки приложения



The screenshot displays an IDE interface with a project explorer on the left and a code editor on the right. The project explorer shows the file structure for 'CMP.AURORA.EXAMPLE', with 'build.gradle.kts' selected. The code editor shows the following Kotlin Gradle build script:

```
alias(libs.plugins.composeMultiplatform)
alias(libs.plugins.composeCompiler)
alias(libs.plugins.auroraBuild)
alias(libs.plugins.auroraDevices)
}

auroraBuild {
    rpm {
        id = "ru.auroraos.demo.cmp"
        name = "Demo CMP"
        description = "Demo CMP application."
        permissions = listOf("Internet")
        libs3rdParty = listOf("maliit-glib")
        icons = projectDir.toPath().resolve("icons")
    }
}

auroraDevices {
    packages {
        create("release") {
            targets = listOf("aarch64", "x86_64")
        }
    }
}
```

Шаг 8. Конфигурация сборки приложения

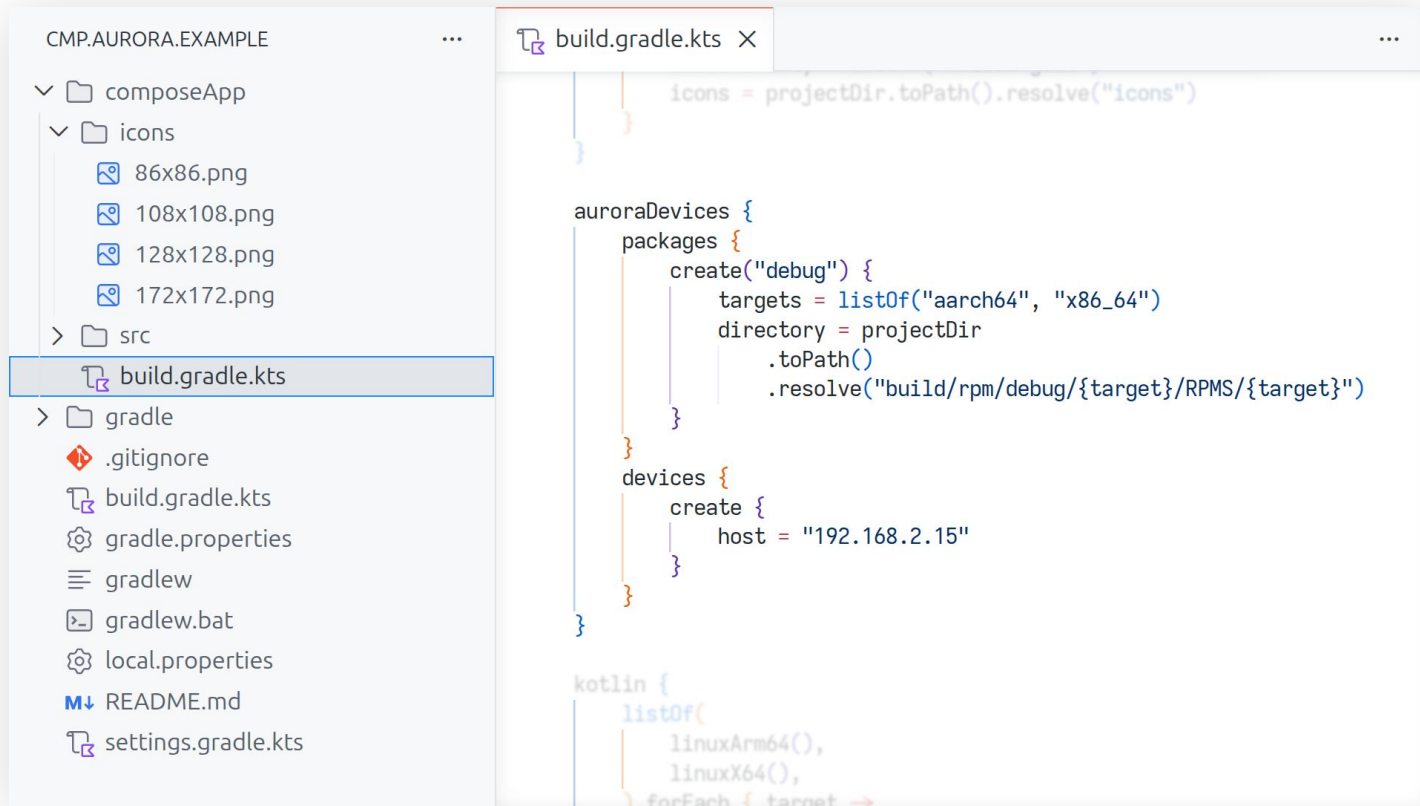
The screenshot displays an IDE interface for configuring a build script. The left sidebar shows the project structure for 'CMP.AURORA.EXAMPLE', with the 'icons' folder selected. The main editor shows the 'build.gradle.kts' file with the following configuration:

```
alias(libs.plugins.composeMultiplatform)
alias(libs.plugins.composeCompiler)
alias(libs.plugins.auroraBuild)
alias(libs.plugins.auroraDevices)
}

auroraBuild {
    rpm {
        id = "ru.auroraos.demo.cmp"
        name = "Demo CMP"
        description = "Demo CMP application."
        permissions = listOf("Internet")
        libs3rdParty = listOf("maliit-glib")
        icons = projectDir.toPath().resolve("icons")
    }
}

auroraDevices {
    packages {
        create("release") {
            targets = listOf("aarch64", "x86_64")
        }
    }
}
```

Шаг 9. Конфигурация девайсов



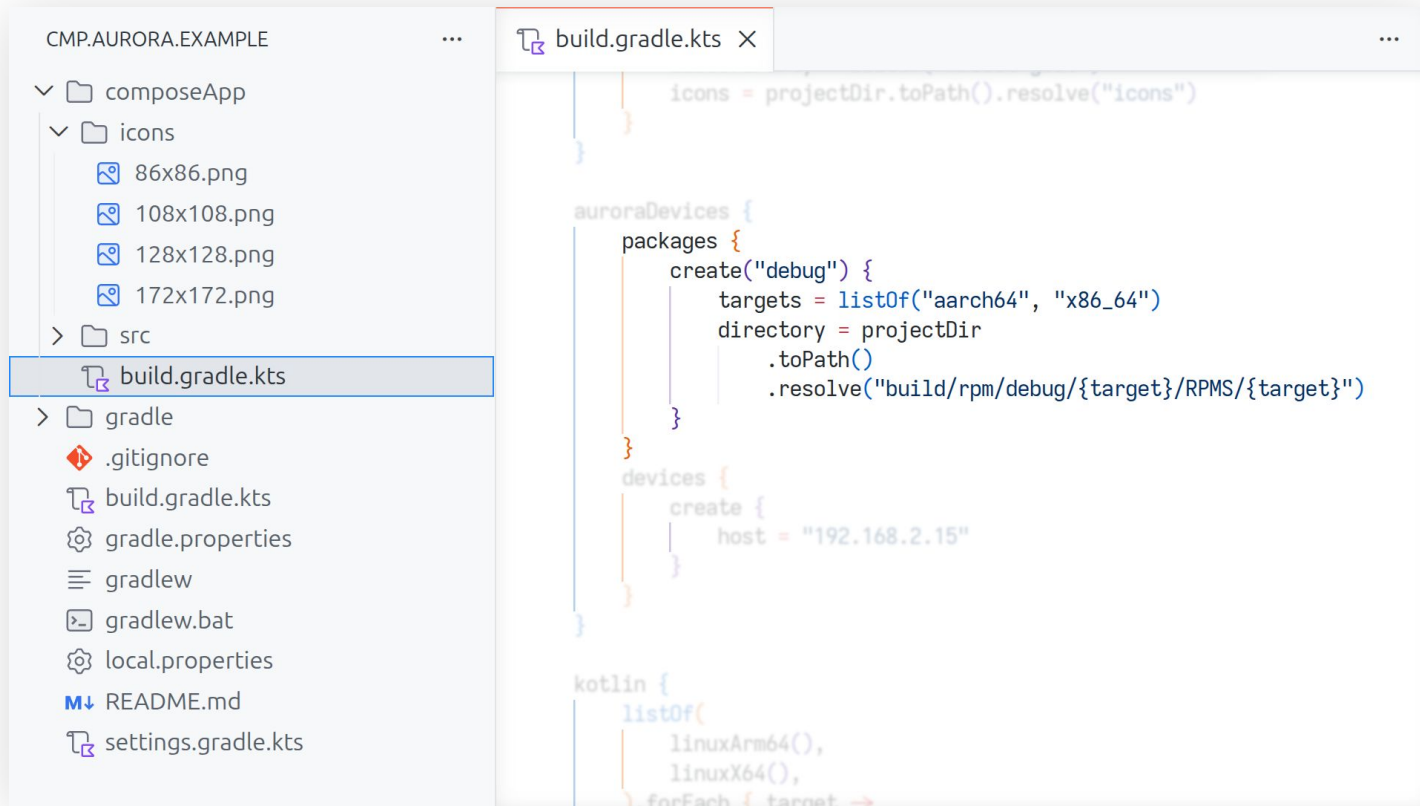
The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'CMP.AURORA.EXAMPLE' with folders like 'composeApp', 'icons', and 'src', and files like 'build.gradle.kts'. The code editor displays the 'build.gradle.kts' file with the following configuration:

```
icons = projectDir.toPath().resolve("icons")

auroraDevices {
    packages {
        create("debug") {
            targets = listOf("aarch64", "x86_64")
            directory = projectDir
                .toPath()
                .resolve("build/rpm/debug/{target}/RPMs/{target}")
        }
    }
    devices {
        create {
            host = "192.168.2.15"
        }
    }
}

kotlin {
    listOf(
        linuxArm64(),
        linuxX64(),
    ).forEach { target ->
```

Шаг 9. Конфигурация девайсов



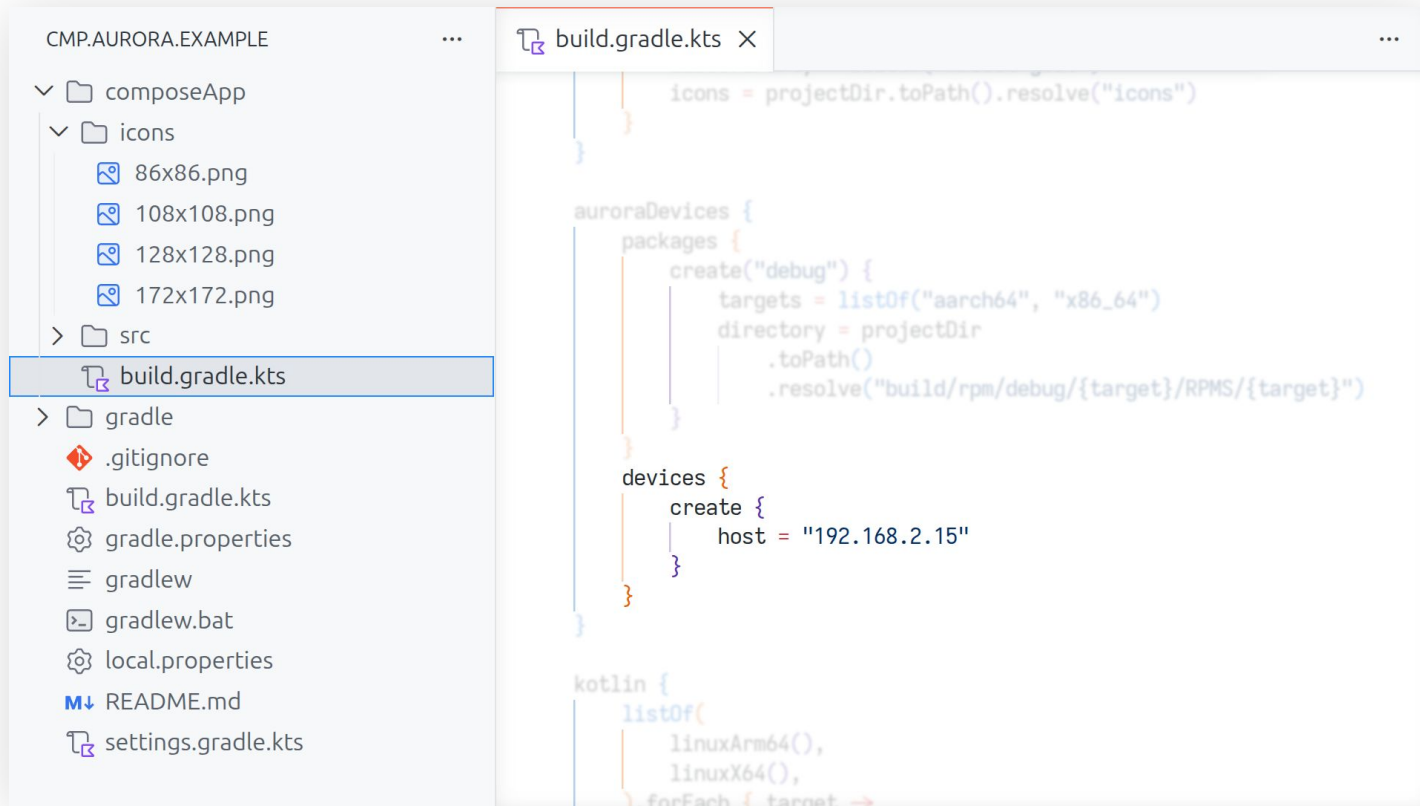
The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows the project structure for CMP.AURORA.EXAMPLE, with the build.gradle.kts file selected. The code editor displays the following Kotlin code:

```
icons = projectDir.toPath().resolve("icons")
}
}

auroraDevices {
    packages {
        create("debug") {
            targets = listOf("aarch64", "x86_64")
            directory = projectDir
                .toPath()
                .resolve("build/rpm/debug/{target}/RPMs/{target}")
        }
    }
    devices {
        create {
            host = "192.168.2.15"
        }
    }
}

kotlin {
    listOf(
        linuxArm64(),
        linuxX64(),
    ).forEach { target ->
```

Шаг 9. Конфигурация девайсов



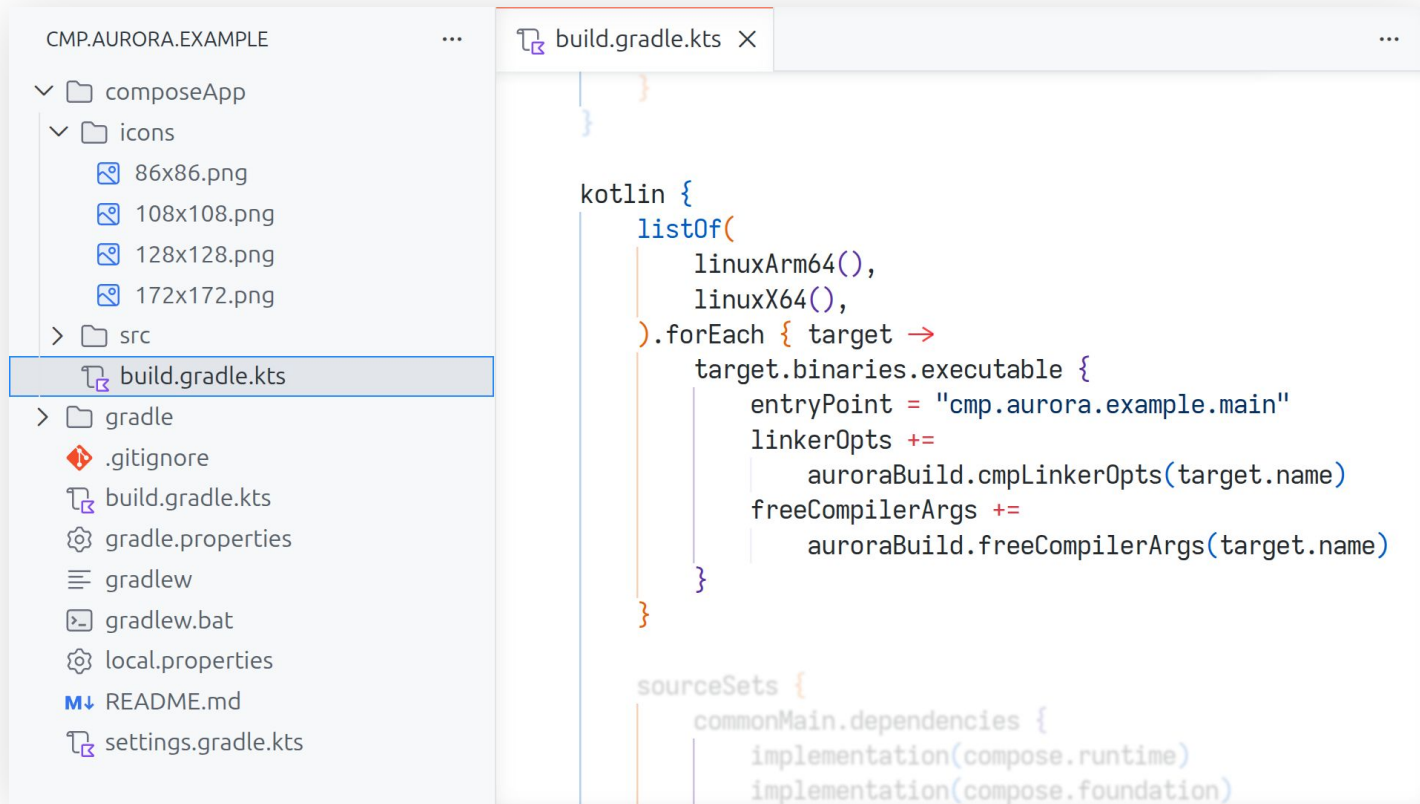
The image shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'CMP.AURORA.EXAMPLE' with folders 'composeApp' and 'icons' (containing '86x86.png', '108x108.png', '128x128.png', '172x172.png') and a 'src' folder. The 'build.gradle.kts' file is selected. The code editor shows the following Kotlin code:

```
icons = projectDir.toPath().resolve("icons")
}
}

auroraDevices {
    packages {
        create("debug") {
            targets = listOf("aarch64", "x86_64")
            directory = projectDir
                .toPath()
                .resolve("build/rpm/debug/{target}/RPMS/{target}")
        }
    }
    devices {
        create {
            host = "192.168.2.15"
        }
    }
}

kotlin {
    listOf(
        linuxArm64(),
        linuxX64(),
    ).forEach { target ->
```

Шаг 10. Конфигурация тергетов



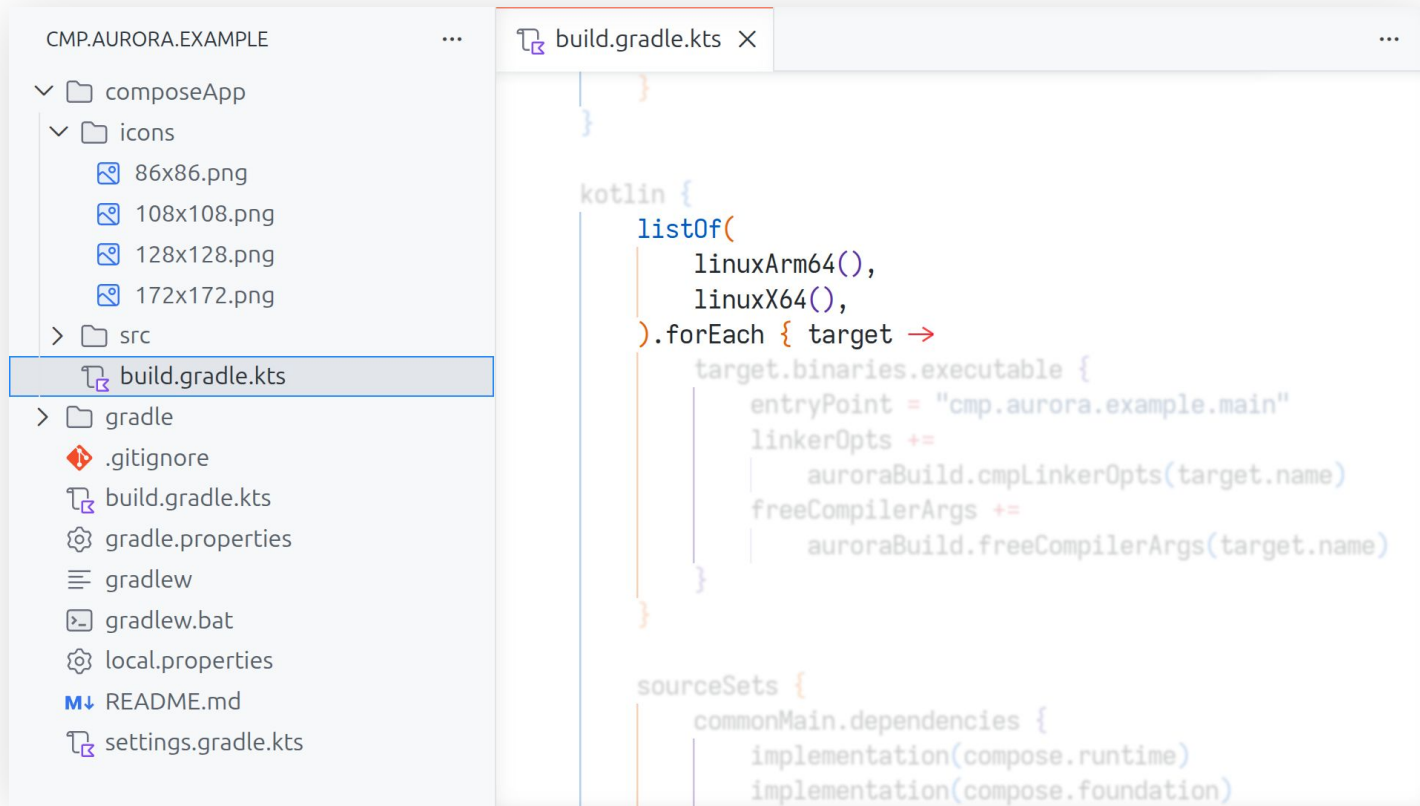
The screenshot shows an IDE window for a project named "CMP.AURORA.EXAMPLE". The left sidebar displays a file tree with the following structure:

- composeApp
 - icons
 - 86x86.png
 - 108x108.png
 - 128x128.png
 - 172x172.png
 - src
 - build.gradle.kts** (selected)
- gradle
 - .gitignore
 - build.gradle.kts
 - gradle.properties
 - gradlew
 - gradlew.bat
 - local.properties
 - README.md
 - settings.gradle.kts

The main editor window displays the content of the selected "build.gradle.kts" file. The code is as follows:

```
    }  
}  
  
kotlin {  
    listOf(  
        linuxArm64(),  
        linuxX64(),  
    ).forEach { target ->  
        target.binaries.executable {  
            entryPoint = "cmp.aurora.example.main"  
            linkerOpts +=  
                auroraBuild.cmpLinkerOpts(target.name)  
            freeCompilerArgs +=  
                auroraBuild.freeCompilerArgs(target.name)  
        }  
    }  
}  
  
sourceSets {  
    commonMain.dependencies {  
        implementation(compose.runtime)  
        implementation(compose.foundation)
```

Шаг 10. Конфигурация тергетов



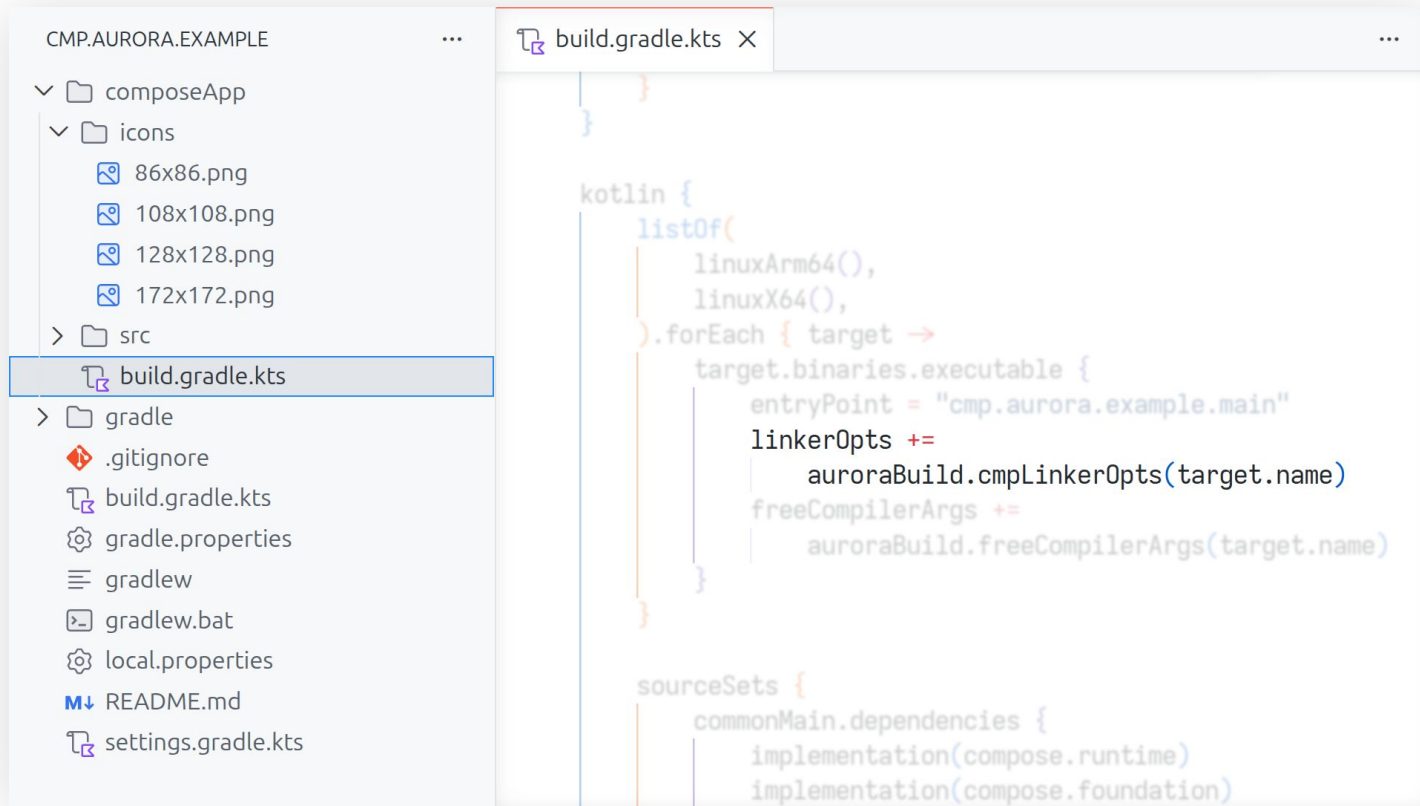
The screenshot shows an IDE window for a project named "CMP.AURORA.EXAMPLE". The left sidebar displays a file tree with the following structure:

- composeApp
 - icons
 - 86x86.png
 - 108x108.png
 - 128x128.png
 - 172x172.png
 - src
 - build.gradle.kts** (selected)
 - gradle
 - .gitignore
 - build.gradle.kts
 - gradle.properties
 - gradlew
 - gradlew.bat
 - local.properties
 - README.md
 - settings.gradle.kts

The main editor window shows the content of the selected "build.gradle.kts" file. The visible code is as follows:

```
    }  
}  
  
kotlin {  
    listOf(  
        linuxArm64(),  
        linuxX64(),  
    ).forEach { target ->  
        target.binaries.executable {  
            entryPoint = "cmp.aurora.example.main"  
            linkerOpts +=  
                auroraBuild.cmpLinkerOpts(target.name)  
            freeCompilerArgs +=  
                auroraBuild.freeCompilerArgs(target.name)  
        }  
    }  
}  
  
sourceSets {  
    commonMain.dependencies {  
        implementation(compose.runtime)  
        implementation(compose.foundation)
```

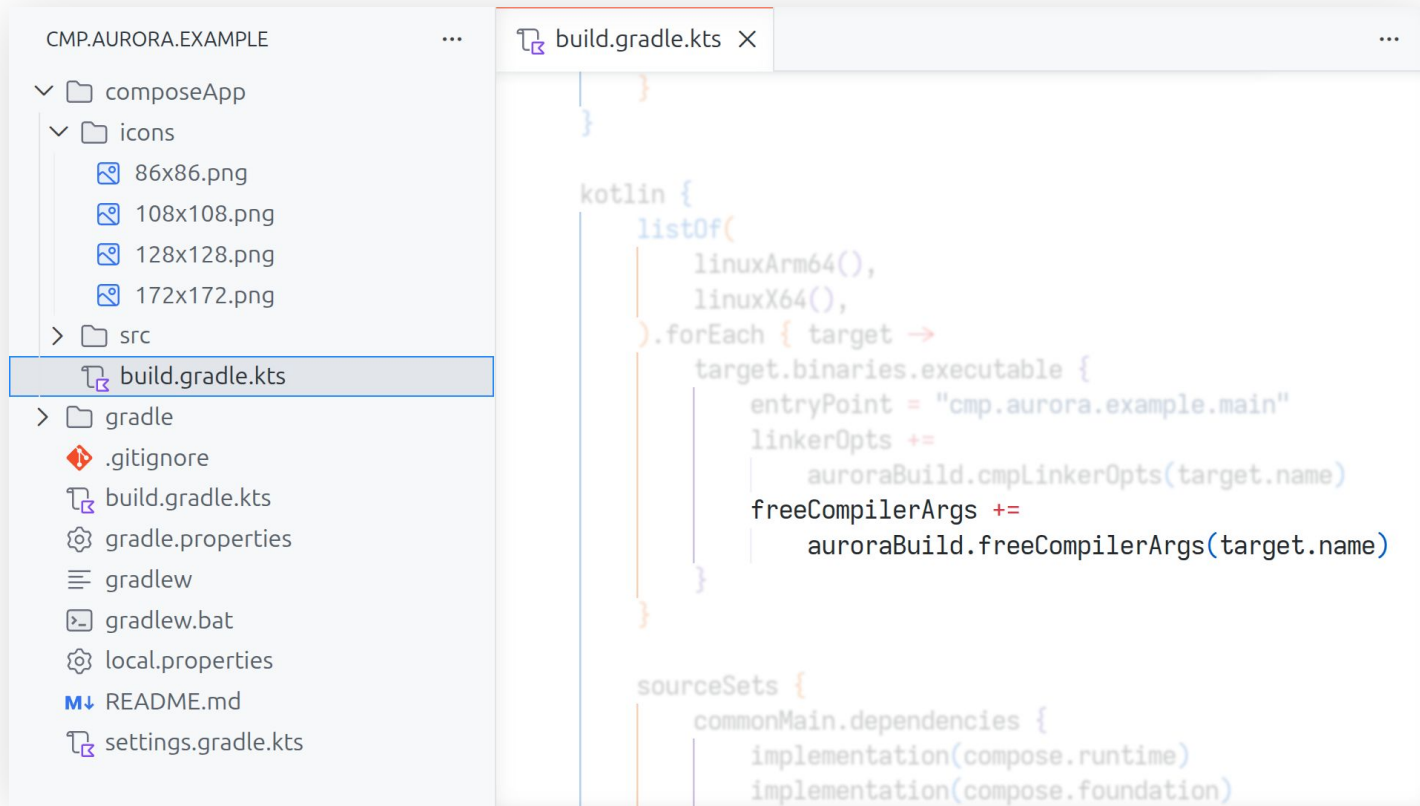
Шаг 10. Конфигурация тергетов



The screenshot shows an IDE window for a project named CMP.AURORA.EXAMPLE. The left sidebar displays the project structure, with the `build.gradle.kts` file selected. The main editor area shows the following Kotlin code:

```
    }  
  }  
  
  kotlin {  
    listOf(  
      linuxArm64(),  
      linuxX64(),  
    ).forEach { target ->  
      target.binaries.executable {  
        entryPoint = "cmp.aurora.example.main"  
        linkerOpts +=  
          auroraBuild.cmpLinkerOpts(target.name)  
        freeCompilerArgs +=  
          auroraBuild.freeCompilerArgs(target.name)  
      }  
    }  
  }  
  
  sourceSets {  
    commonMain.dependencies {  
      implementation(compose.runtime)  
      implementation(compose.foundation)
```

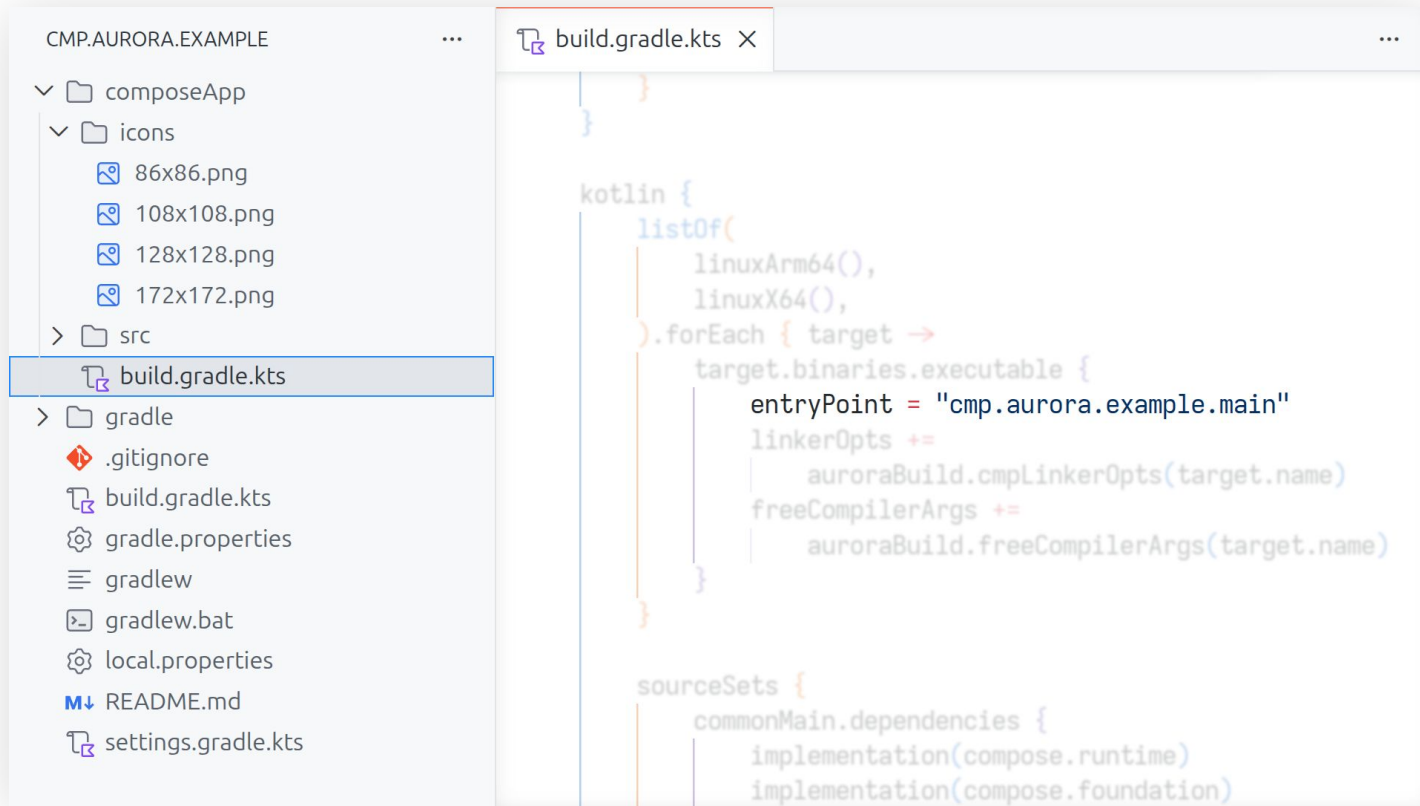
Шаг 10. Конфигурация тергетов



The screenshot shows an IDE window for a project named CMP.AURORA.EXAMPLE. The left sidebar displays the project structure, with the `build.gradle.kts` file selected. The main editor shows the following Kotlin code:

```
    }  
  }  
  
  kotlin {  
    listOf(  
      linuxArm64(),  
      linuxX64(),  
    ).forEach { target ->  
      target.binaries.executable {  
        entryPoint = "cmp.aurora.example.main"  
        linkerOpts +=  
          auroraBuild.cmpLinkerOpts(target.name)  
        freeCompilerArgs +=  
          auroraBuild.freeCompilerArgs(target.name)  
      }  
    }  
  }  
  
  sourceSets {  
    commonMain.dependencies {  
      implementation(compose.runtime)  
      implementation(compose.foundation)
```

Шаг 10. Конфигурация тергетов



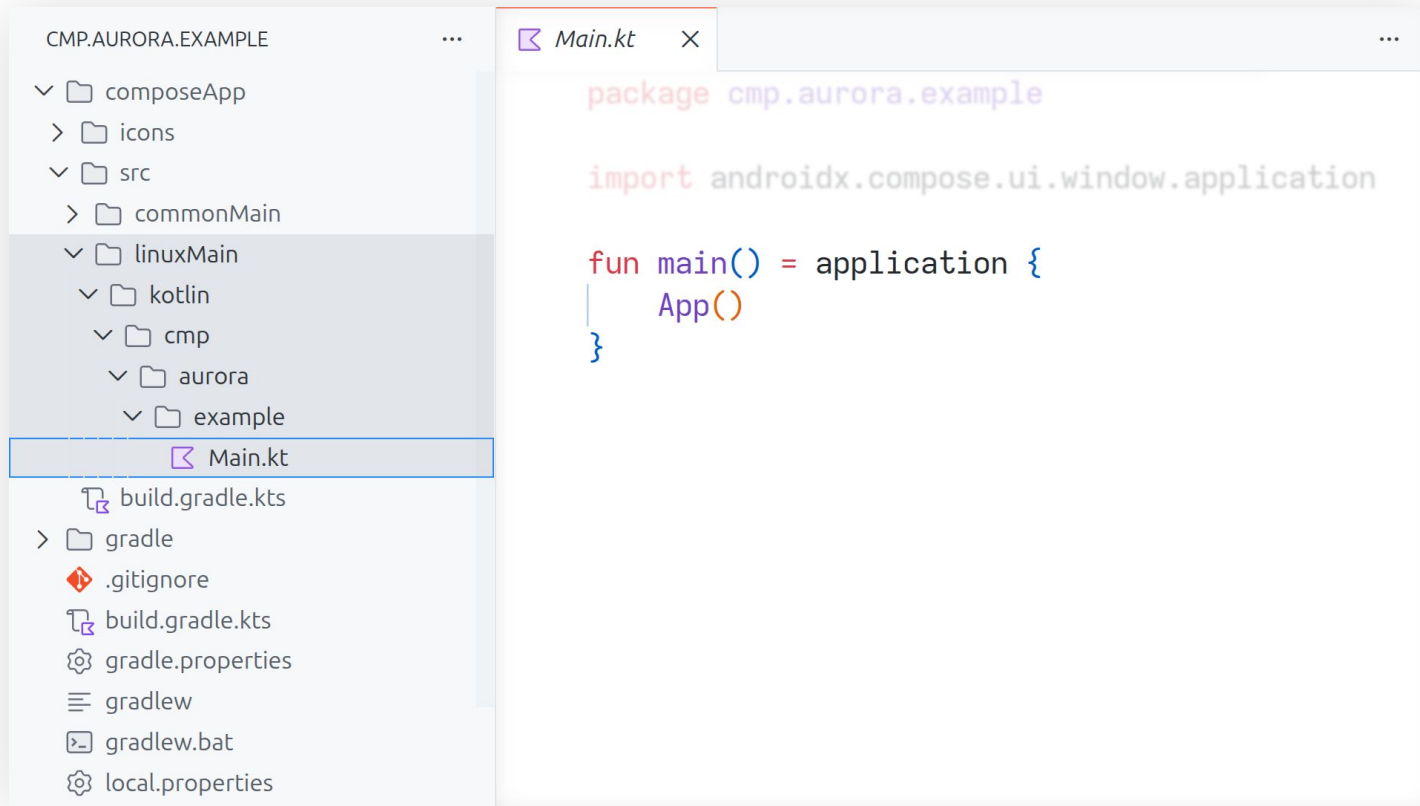
The screenshot shows an IDE window for a project named "CMP.AURORA.EXAMPLE". The left sidebar displays a file tree with the following structure:

- composeApp
 - icons
 - 86x86.png
 - 108x108.png
 - 128x128.png
 - 172x172.png
 - src
 - build.gradle.kts** (selected)
- gradle
 - .gitignore
 - build.gradle.kts
 - gradle.properties
 - gradlew
 - gradlew.bat
 - local.properties
 - README.md
 - settings.gradle.kts

The main editor window shows the content of the selected "build.gradle.kts" file. The code is as follows:

```
    }  
  }  
}  
  
kotlin {  
  listOf(  
    linuxArm64(),  
    linuxX64(),  
  ).forEach { target ->  
    target.binaries.executable {  
      entryPoint = "cmp.aurora.example.main"  
      linkerOpts +=  
        auroraBuild.cmpLinkerOpts(target.name)  
      freeCompilerArgs +=  
        auroraBuild.freeCompilerArgs(target.name)  
    }  
  }  
}  
  
sourceSets {  
  commonMain.dependencies {  
    implementation(compose.runtime)  
    implementation(compose.foundation)
```

Шаг 10. Конфигурация тергетов



The screenshot shows an IDE window for a project named CMP.AURORA.EXAMPLE. The left sidebar displays a file tree with the following structure:

- composeApp
 - icons
 - src
 - commonMain
 - linuxMain
 - kotlin
 - cmp
 - aurora
 - example
 - Main.kt**

- gradle
- .gitignore
- build.gradle.kts
- gradle.properties
- gradlew
- gradlew.bat
- local.properties

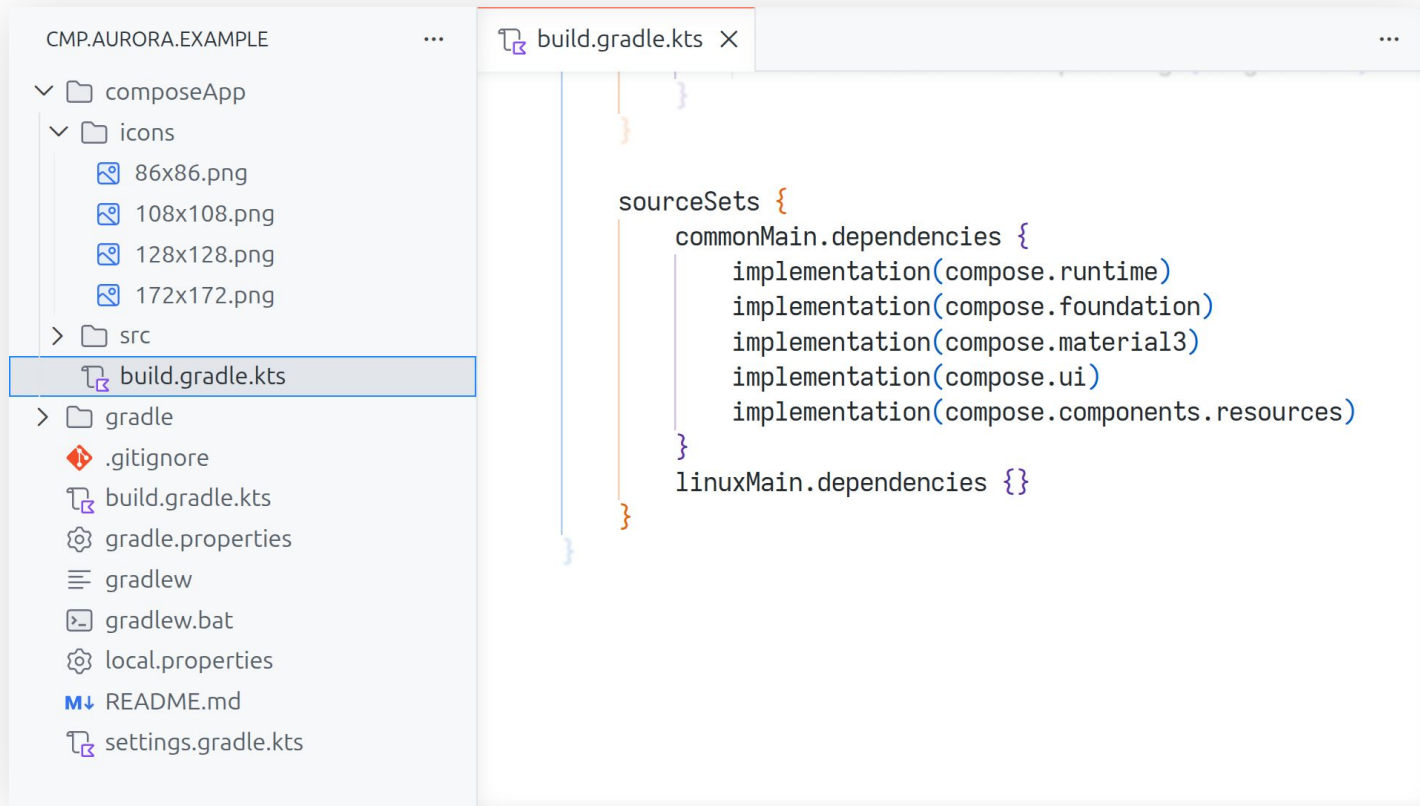
The main editor window shows the content of the selected `Main.kt` file:

```
package cmp.aurora.example

import androidx.compose.ui.window.application

fun main() = application {
    App()
}
```

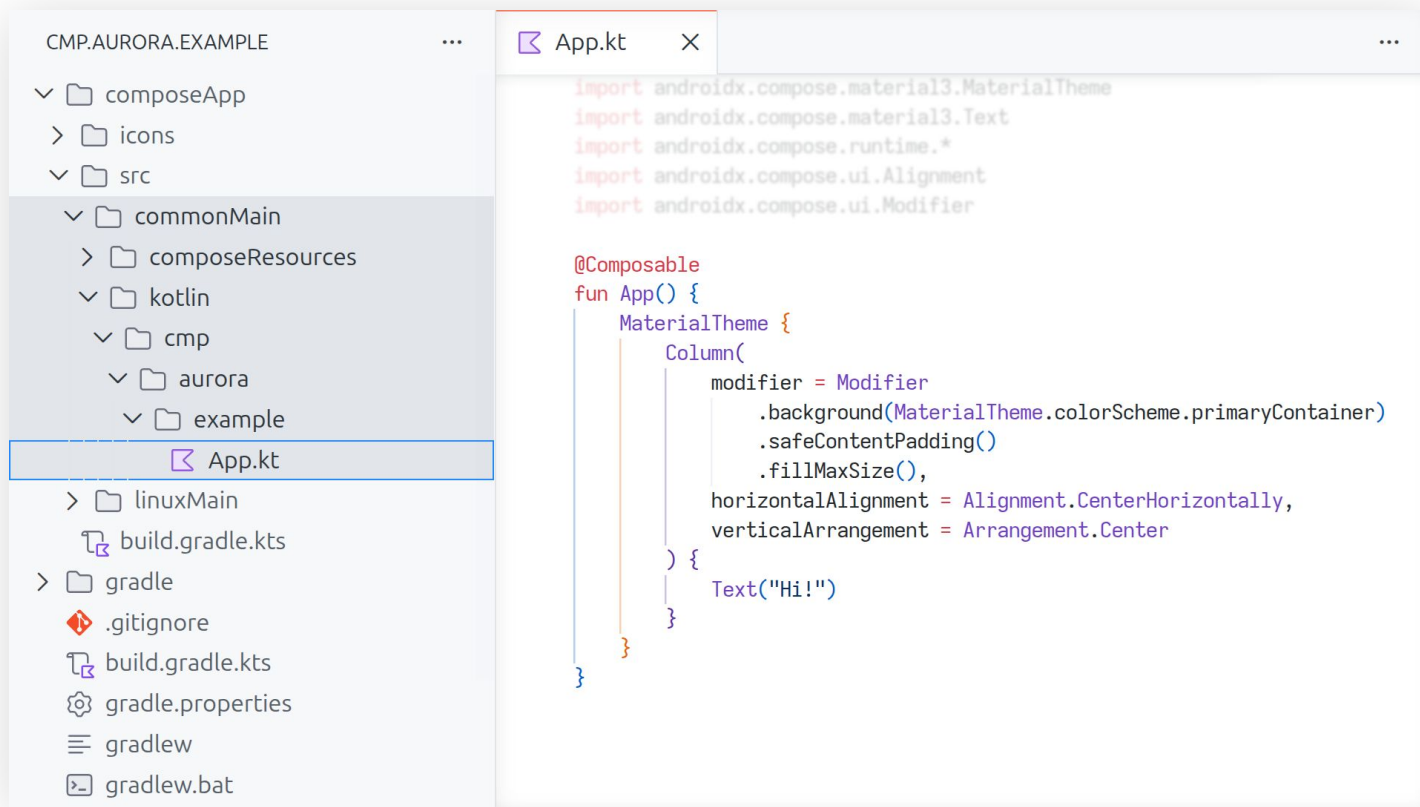
Шаг 11. Конфигурация зависимостей



The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'CMP.AURORA.EXAMPLE' with folders 'composeApp' and 'icons' (containing PNG files) and a 'src' folder. The 'build.gradle.kts' file is selected. The code editor shows the following Kotlin code:

```
sourceSets {  
    commonMain.dependencies {  
        implementation(compose.runtime)  
        implementation(compose.foundation)  
        implementation(compose.material3)  
        implementation(compose.ui)  
        implementation(compose.components.resources)  
    }  
    linuxMain.dependencies {}  
}
```

Шаг 12. Реализация приложения



```
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier

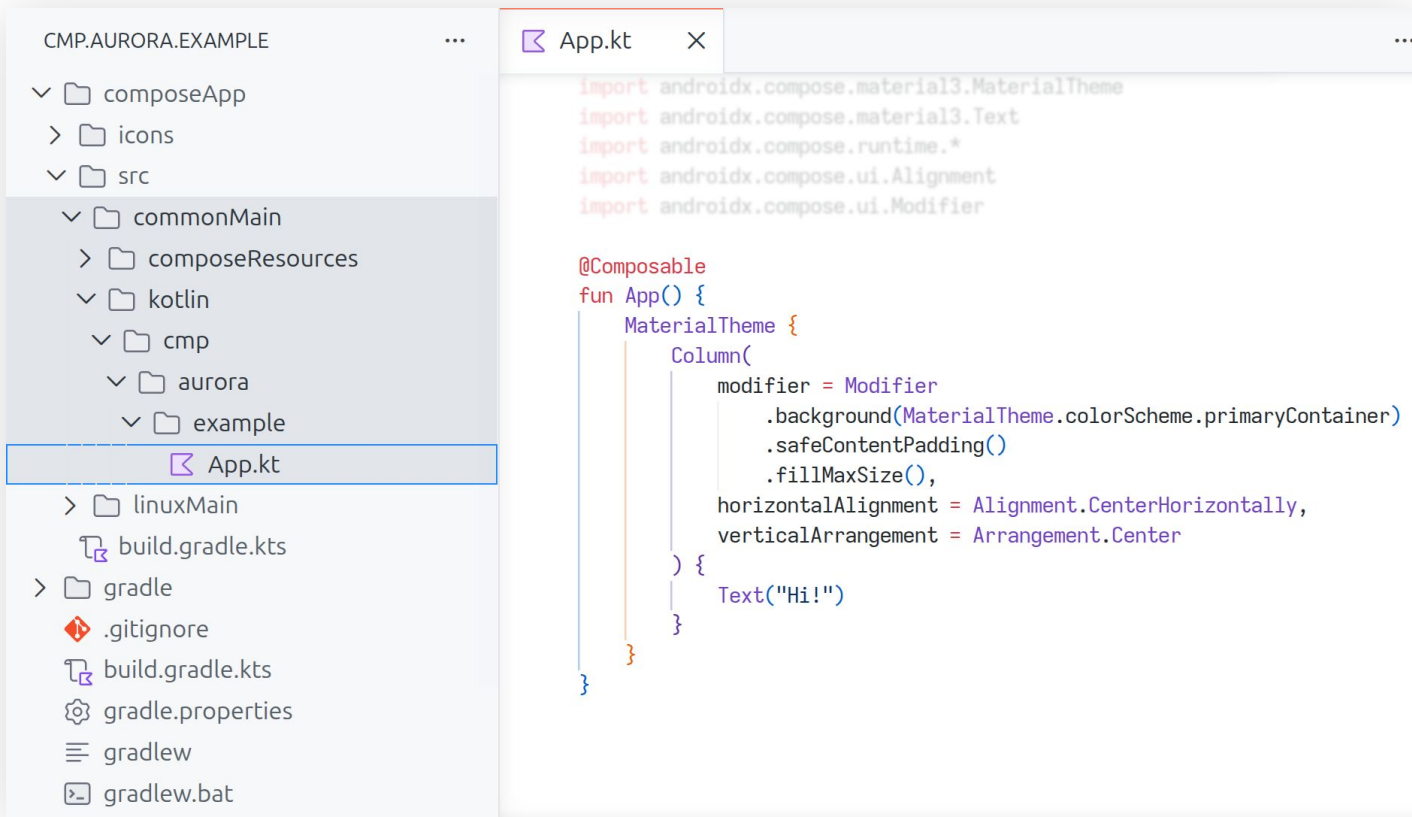
@Composable
fun App() {
    MaterialTheme {
        Column(
            modifier = Modifier
                .background(MaterialTheme.colorScheme.primaryContainer)
                .safeContentPadding()
                .fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text("Hi!")
        }
    }
}
```

Шаг 13. Сборка и запуск приложения

```
Сборка и запуск проекта

./gradlew :composeApp:buildDebugPipeline
./gradlew :composeApp:runDebugOnDevice
./gradlew :composeApp:runDebugOnEmulator
```

Шаг 13. Сборка и запуск приложения



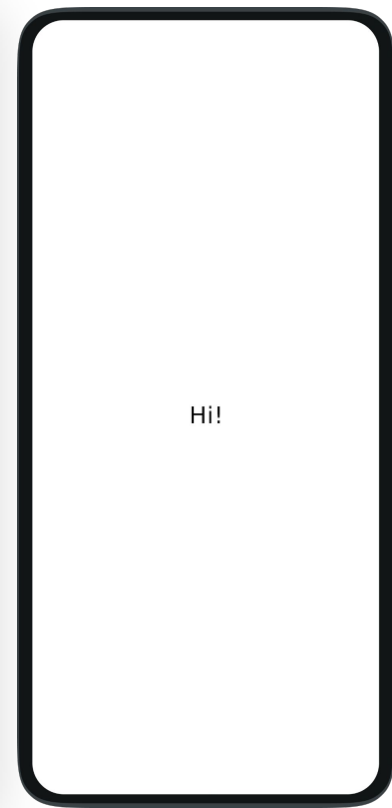
The screenshot shows an IDE window with the following structure:

- Project: CMP.AURORA.EXAMPLE
- Directories:
 - composeApp
 - icons
 - src
 - commonMain
 - composeResources
 - kotlin
 - cmp
 - aurora
 - example
 - App.kt (selected)

- Files:
- linuxMain
 - build.gradle.kts
- gradle
 - .gitignore
 - build.gradle.kts
 - gradle.properties
 - gradlew
 - gradlew.bat

```
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier

@Composable
fun App() {
    MaterialTheme {
        Column(
            modifier = Modifier
                .background(MaterialTheme.colorScheme.primaryContainer)
                .safeContentPadding()
                .fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Text("Hi!")
        }
    }
}
```





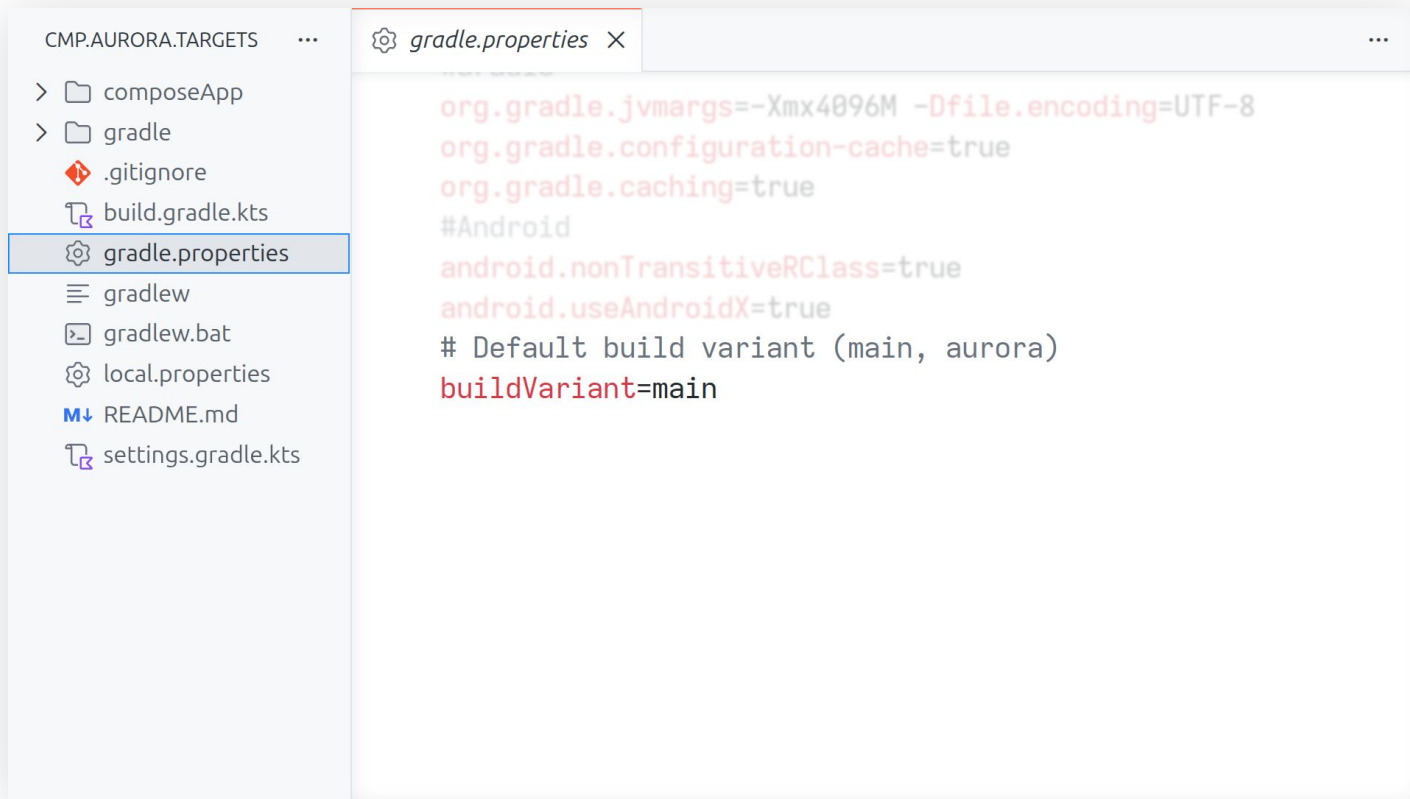
Demo CMP

<https://hub.mos.ru/auroraos/kotlin-multiplatform/demos/demo-cmp>



Как добавить поддержку ОС Аврора в
существующий проект?

Шаг 1. Добавление свойства Gradle



Шаг 2. Настройка параметризированной сборки



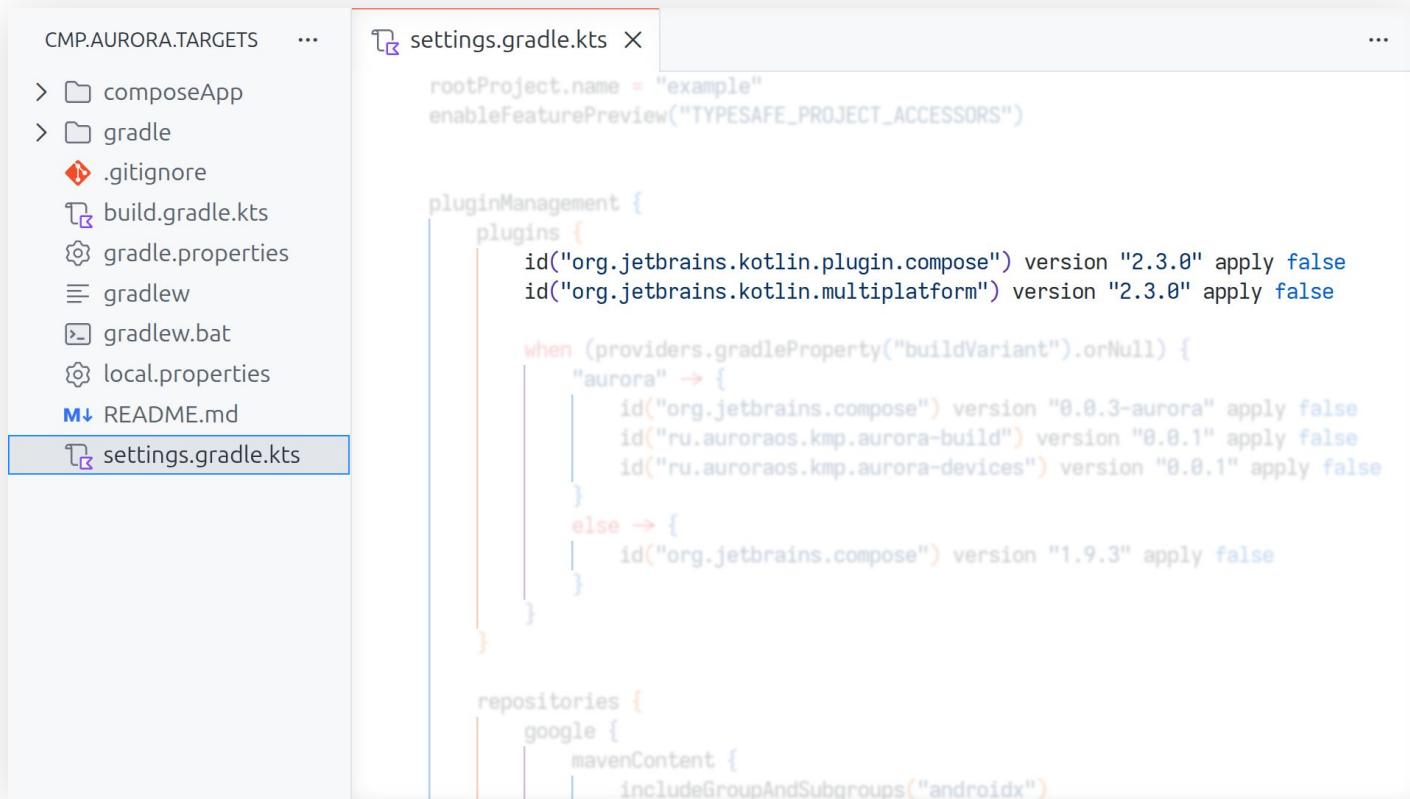
```
rootProject.name = "example"
enableFeaturePreview("TYPESAFE_PROJECT_ACCESSORS")

pluginManagement {
    plugins {
        id("org.jetbrains.kotlin.plugin.compose") version "2.3.0" apply false
        id("org.jetbrains.kotlin.multiplatform") version "2.3.0" apply false

        when (providers.gradleProperty("buildVariant").orNull) {
            "aurora" -> {
                id("org.jetbrains.compose") version "0.0.3-aurora" apply false
                id("ru.auroraos.kmp.aurora-build") version "0.0.1" apply false
                id("ru.auroraos.kmp.aurora-devices") version "0.0.1" apply false
            }
            else -> {
                id("org.jetbrains.compose") version "1.9.3" apply false
            }
        }
    }
}

repositories {
    google {
        mavenContent {
            includeGroupAndSubgroups("androidx")
        }
    }
}
```

Шаг 2. Настройка параметризированной сборки



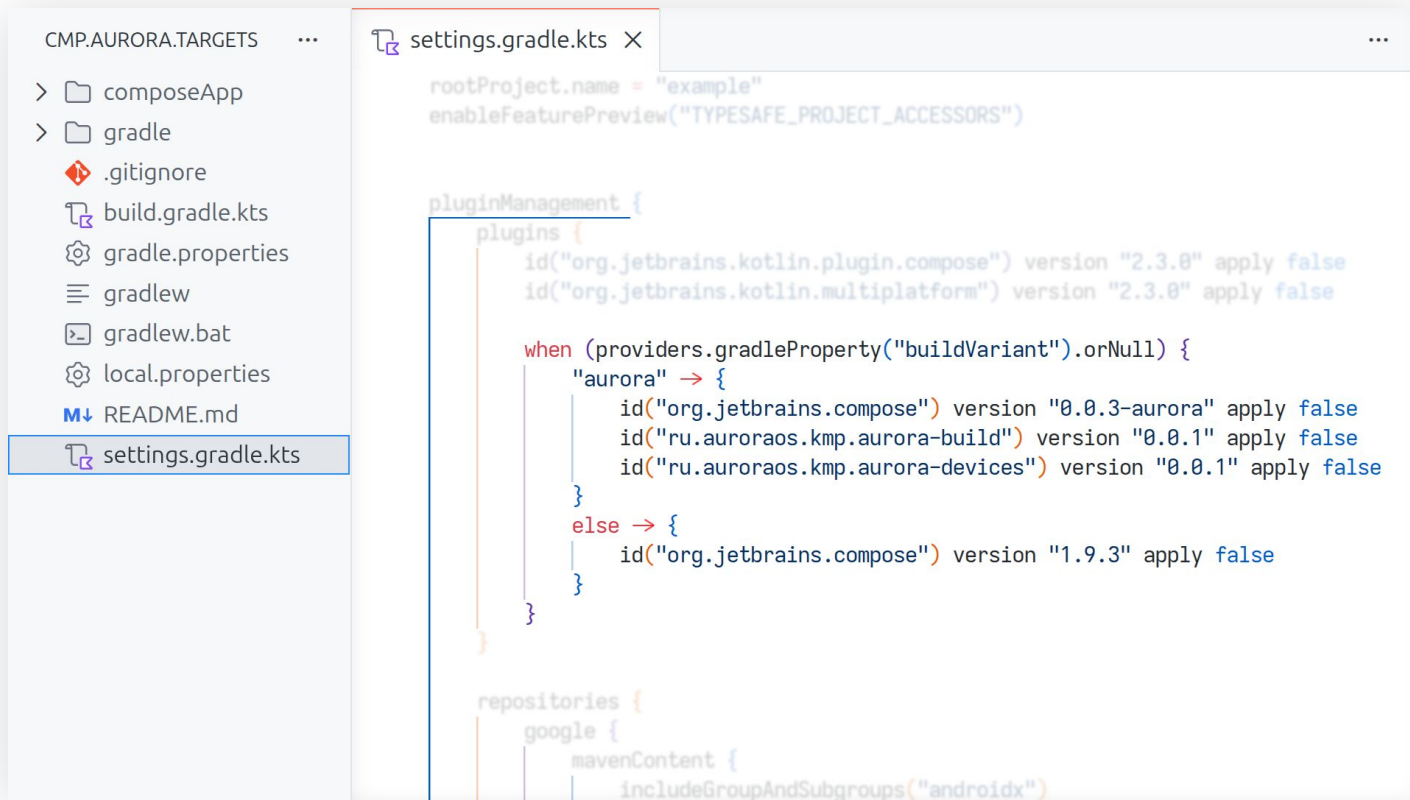
```
rootProject.name = "example"
enableFeaturePreview("TYPESAFE_PROJECT_ACCESSORS")

pluginManagement {
    plugins {
        id("org.jetbrains.kotlin.plugin.compose") version "2.3.0" apply false
        id("org.jetbrains.kotlin.multiplatform") version "2.3.0" apply false

        when (providers.gradleProperty("buildVariant").orNull) {
            "aurora" -> {
                id("org.jetbrains.compose") version "0.8.3-aurora" apply false
                id("ru.auroraos.kmp.aurora-build") version "0.8.1" apply false
                id("ru.auroraos.kmp.aurora-devices") version "0.8.1" apply false
            }
            else -> {
                id("org.jetbrains.compose") version "1.9.3" apply false
            }
        }
    }
}

repositories {
    google {
        mavenContent {
            includeGroupAndSubgroups("androidx")
        }
    }
}
```

Шаг 2. Настройка параметризированной сборки



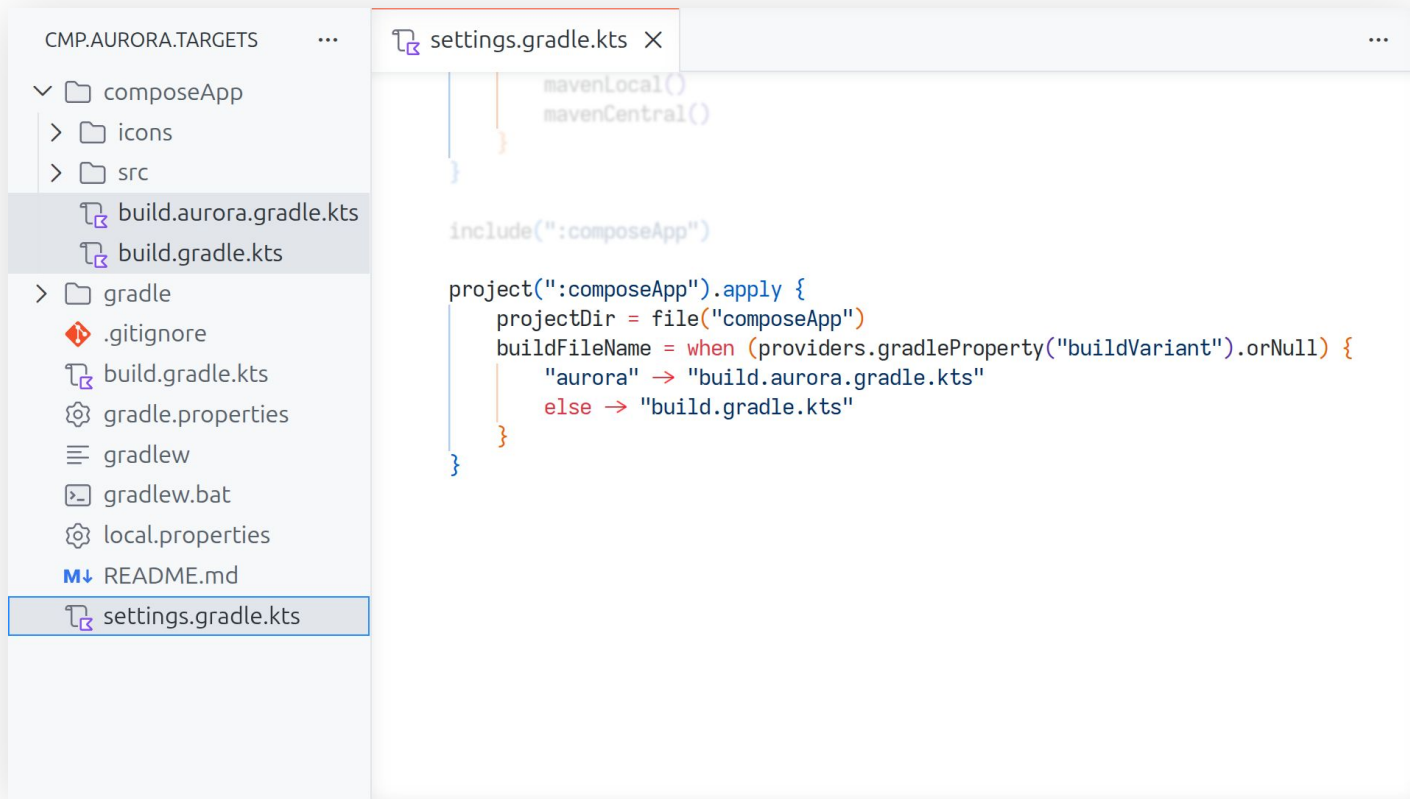
```
rootProject.name = "example"
enableFeaturePreview("TYPESAFE_PROJECT_ACCESSORS")

pluginManagement {
    plugins {
        id("org.jetbrains.kotlin.plugin.compose") version "2.3.0" apply false
        id("org.jetbrains.kotlin.multiplatform") version "2.3.0" apply false

        when (providers.gradleProperty("buildVariant").orNull) {
            "aurora" -> {
                id("org.jetbrains.compose") version "0.0.3-aurora" apply false
                id("ru.auroraos.kmp.aurora-build") version "0.0.1" apply false
                id("ru.auroraos.kmp.aurora-devices") version "0.0.1" apply false
            }
            else -> {
                id("org.jetbrains.compose") version "1.9.3" apply false
            }
        }
    }
}

repositories {
    google {
        mavenContent {
            includeGroupAndSubgroups("androidx")
        }
    }
}
```

Шаг 2. Настройка параметризированной сборки



The screenshot shows an IDE window with the file explorer on the left and the settings.gradle.kts file open in the editor. The file explorer shows the project structure with folders like composeApp, icons, and src, and files like build.aurora.gradle.kts, build.gradle.kts, gradle, .gitignore, build.gradle.kts, gradle.properties, gradlew, gradlew.bat, local.properties, README.md, and settings.gradle.kts. The settings.gradle.kts file is selected and its content is displayed in the editor.

```
settings.gradle.kts

mavenLocal()
mavenCentral()

include(":composeApp")

project(":composeApp").apply {
    projectDir = file("composeApp")
    buildFileName = when (providers.gradleProperty("buildVariant").orNull) {
        "aurora" -> "build.aurora.gradle.kts"
        else -> "build.gradle.kts"
    }
}
```

Шаг 3. Параметризованная сборка проекта

Параметризованная сборка проекта

```
./gradlew :composeApp:buildDebugPipeline -PbuildVariant=aurora
```



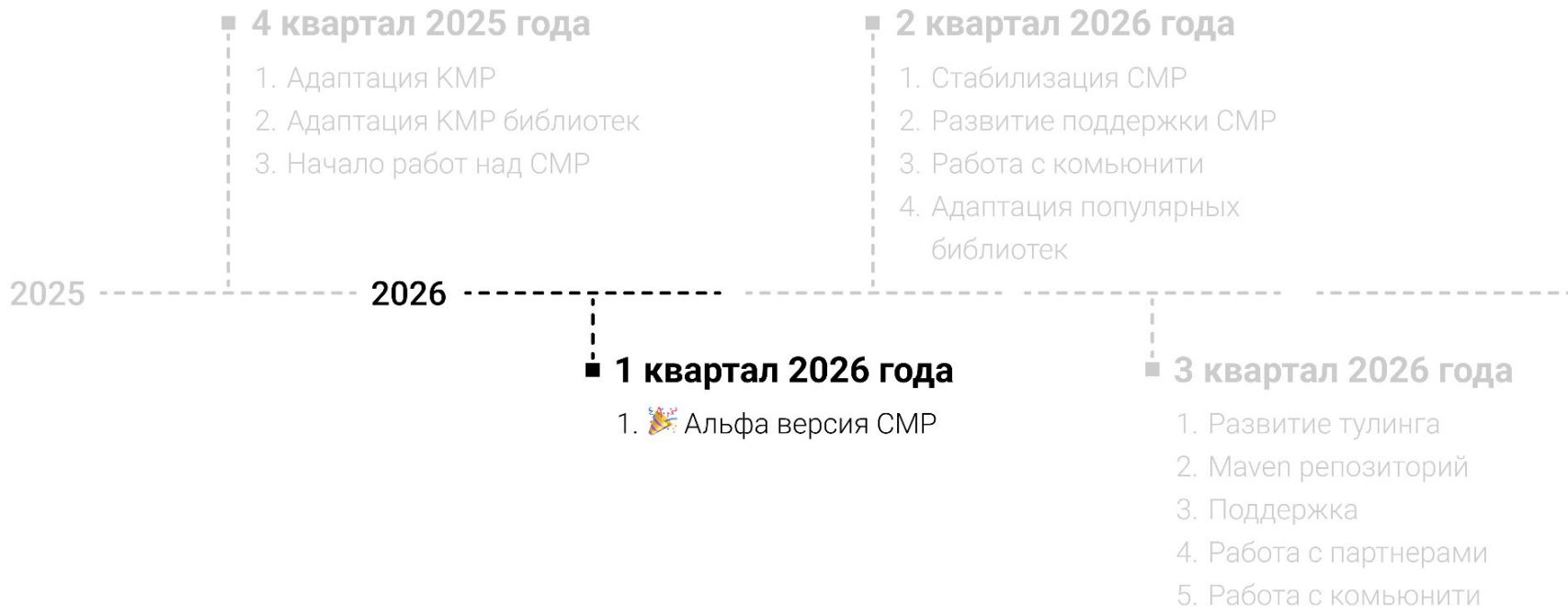
Demo Targets CMP

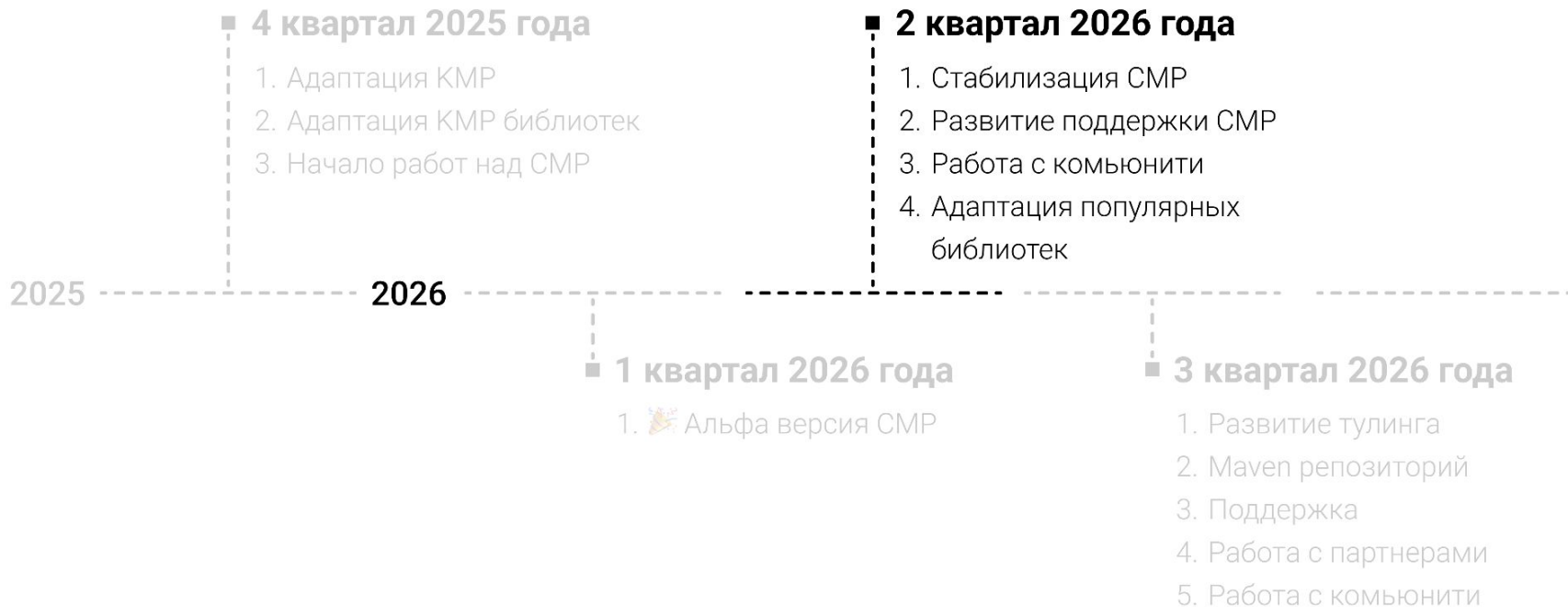
<https://hub.mos.ru/auroraos/kotlin-multiplatform/demos/demo-targets-cmp>



Roadmap по адаптации СМР для ОС Аврора







Список адаптированных библиотек

SQLDelight
v2.0.2

Koin
v4.0.0

Ktor
v3.4.1

BuildConfig
v0.15.1

Kermit
v2.1.0

Kamel
v1.0.9

datetime
v0.7.1

...

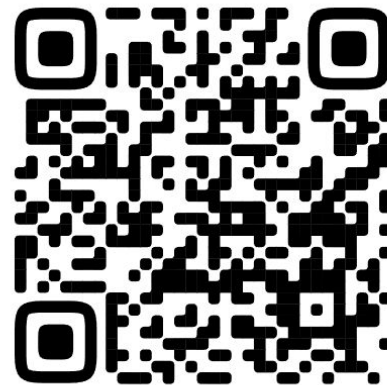




Бета программа



Aurora Developers



Документация

Заключение

- Аврора — современная быстро развивающаяся мобильная операционная система
- Мы нацелены на предоставление удобных инструментов для экосистемы
- Мы приветствуем партнеров в совместном развитии платформы
- Фреймворк KMP/СMP уже находится в альфа версии и продолжает развиваться

