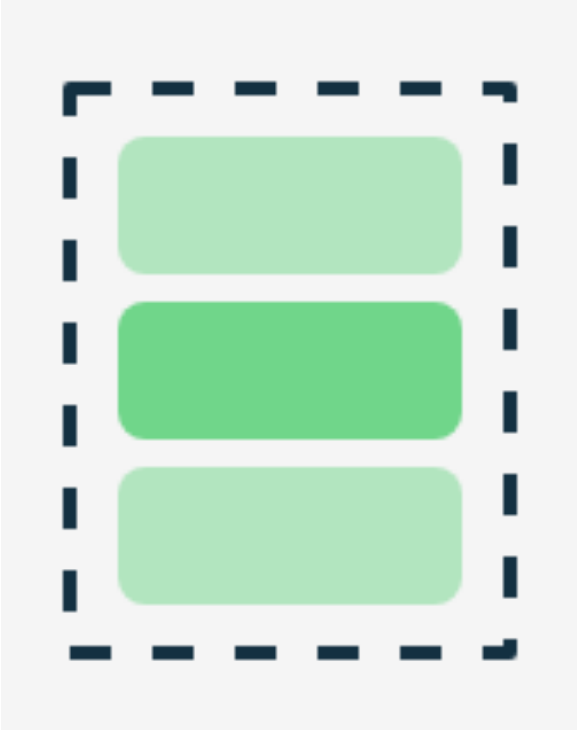
# Layout Dasar jetpack Compose

# Komponen Tataletak Dasar

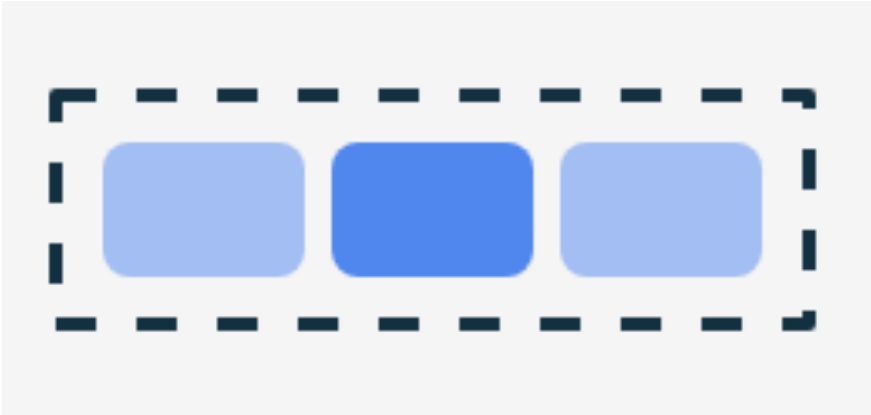
- ***Column***, digunakan untuk menyusun elemen widget secara vertical
- ***Row***, digunakan untuk menyusun elemen widget secara horizontal
- ***Box***, digunakan untuk menyusun object secara bertindihan

# Column



```
@Composable
fun TataletakColumn() {
    Column() {
        Text(text = "Elemen1")
        Text(text = "Elemen2")
        Text(text = "Elemen3")
        Text(text = "Elemen4")
        Text(text = "Elemen5")
    }
}
```

# Row



```
@Composable
fun TataletakRow(modifier: Modifier) {
    Row(modifier = modifier.fillMaxWidth(), horizontalArrangement = Arrangement.SpaceEvenly) {
        Text(text = "Elemen1")
        Text(text = "Elemen2")
        Text(text = "Elemen3")
        Text(text = "Elemen4")
        Text(text = "Elemen5")
    }
}
```

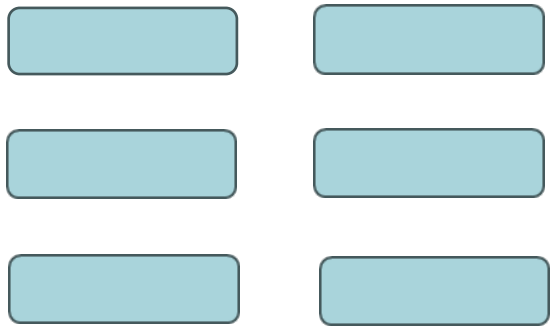
# Box

```
@Composable
fun TataletakBox(modifier: Modifier) {
    Box(
        modifier = modifier
            .fillMaxHeight()
            .fillMaxWidth(), contentAlignment = Alignment.Center
    ) {
        Text(text = "Box 1")
        Text(text = "Column 1")
        Text(text = "Row 1")
        Text(text = "Box 2")
        Text(text = "Column 2")
    }
}
```

# Column-Row

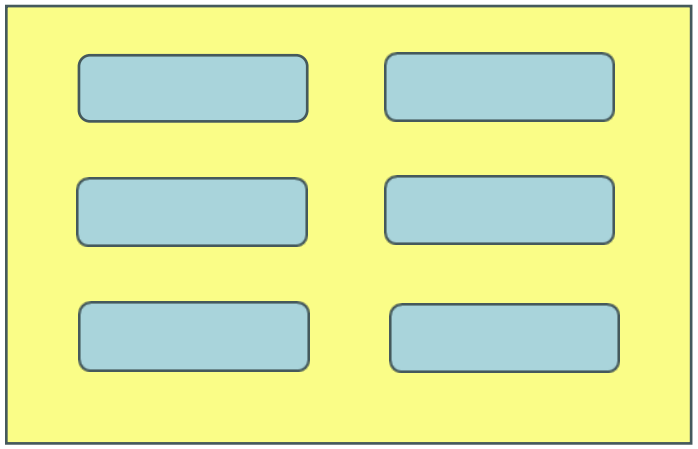
```
@Composable
fun TataletakColumnRow(modifier: Modifier) {
    Column() {
        Row(modifier = modifier.fillMaxWidth(), horizontalArrangement = Arrangement.SpaceEvenly) {
            Text(text = "Column1_Row1")
            Text(text = "Column1_Row2")
            Text(text = "Column1_Row3")
        }
        Row(modifier = modifier.fillMaxWidth(), horizontalArrangement = Arrangement.SpaceEvenly) {
            Text(text = "Column2_Row1")
            Text(text = "Column2_Row2")
            Text(text = "Column2_Row3")
        }
    }
}
```

# Row-Column



```
@Composable
fun TataletakRowColumn(modifier: Modifier) {
        Row(modifier = modifier.fillMaxWidth(), horizontalArrangement = Arrangement.SpaceEvenly) {
            Column(){
                Text(text = "Row1_Column1")
                Text(text = "Row1_Column2")
                Text(text = "Row1_Column3")
            }
            Column(){
                Text(text = "Row2_Column1")
                Text(text = "Row2_Column2")
                Text(text = "Row2_Column3")
            }
        }
}
```

# Box-Column-Row



```
@Composable
fun TataletakBoxColumnRow(modifier: Modifier) {
    Column {
        Box(
            modifier = modifier
                .fillMaxWidth()
                .height(110.dp)
                .background(color = Color.Yellow), contentAlignment = Alignment.Center
        ) {
            Column() {
                Row(
                    modifier = modifier.fillMaxWidth(),
                    horizontalArrangement = Arrangement.SpaceEvenly
                ) {
                    Text(text = "Column1_Row1")
                    Text(text = "Column1_Row2")
                    Text(text = "Column1_Row3")
                }
                Row(
                    modifier = modifier.fillMaxWidth(),
                    horizontalArrangement = Arrangement.SpaceEvenly
                ) {
                    Text(text = "Column2_Row1")
                    Text(text = "Column2_Row2")
                    Text(text = "Column2_Row3")
                }
            }
        }

    }
}
```

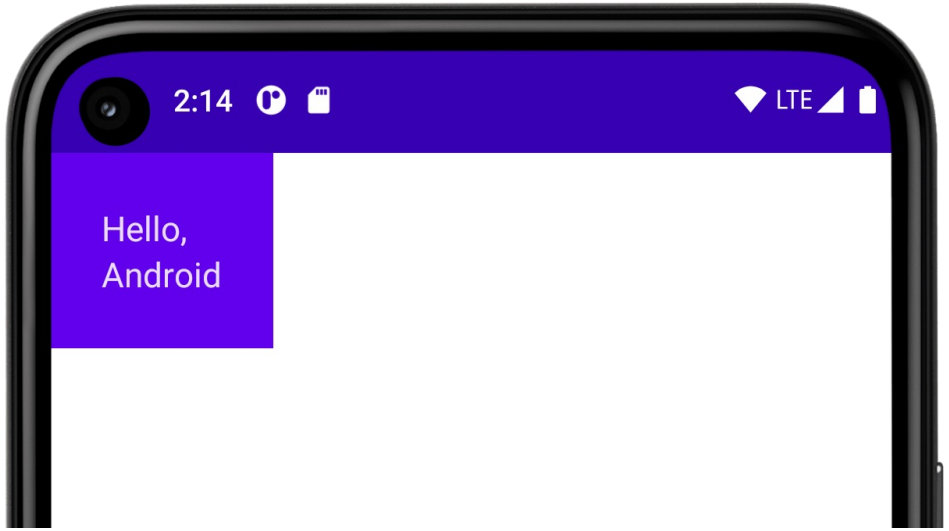
# Modifier

Modifier memungkinkan untuk melakukan hal-hal berikut:

- Mengubah ukuran, tata letak, perilaku, dan tampilan composable
- Menambahkan informasi, seperti label aksesibilitas
- Memproses input pengguna
- Menambahkan interaksi tingkat tinggi, seperti membuat elemen yang dapat diklik, dapat di-scroll, dapat ditarik, atau dapat di-zoom

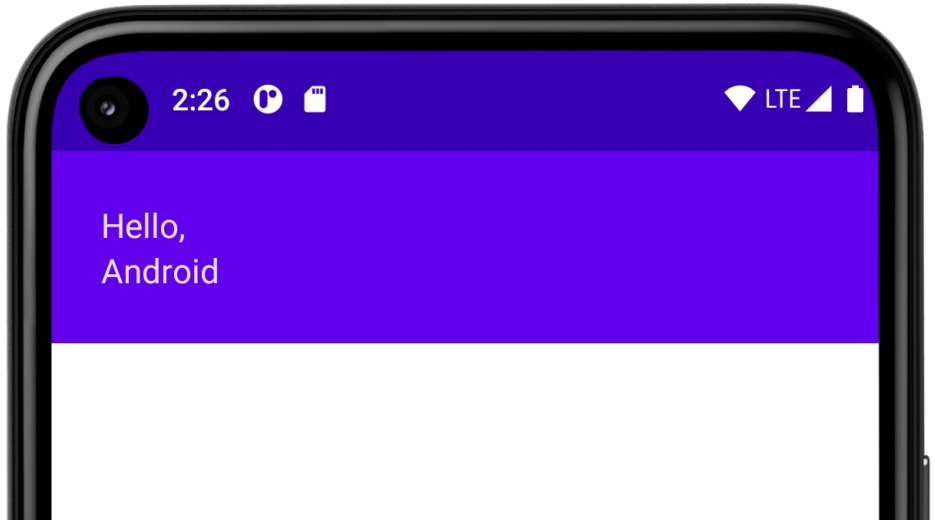
# Modifier

```
@Composable
private fun Greeting(name: String) {
    Column(modifier = Modifier.padding(24.dp)) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```



# Modifier

```
@Composable
private fun Greeting(name: String) {
    Column(modifier = Modifier
            .padding(24.dp)
            .fillMaxWidth()
) {
        Text(text = "Hello,")
        Text(text = name)
    }
}
```



# Modifier

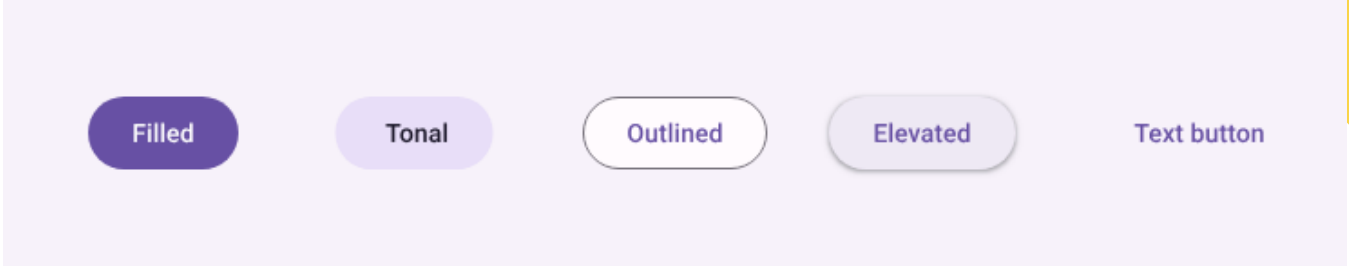
## Daftar Modifier

- Actions (Tindakan)
- Alignment (Perataan)
- Animation (Animasi)
- Border (Batas)
- Drawing (Gambar)
- Focus (Fokus)
- Graphics (Grafis)
- Keyboard
- Layout (Tataletak)
- Padding (Ruang)
- Pointer (Pointer)
- Position (Posisi)
- Semantics (Semantik)
- Scroll
- Size (Ukuran)
- Testing
- Transformations
- Other (Lainnya)

# Material components Jetpack Compose


Scaffold : Gunakan Scaffold yang dapat dikomposisikan untuk memberikan struktur pada layar Anda.

# Button




| Jenis | Tampilan | Tujuan |
|---|---|---|
| Filled | Latar belakang solid dengan teks yang kontras. | Tombol penekanan tinggi. Ini adalah untuk tindakan utama dalam aplikasi, seperti "kirim" dan "simpan". Efek bayangan menekankan pentingnya tombol. |
| Tonal | Warna latar belakang bervariasi sesuai dengan platform. | Juga untuk tindakan utama atau signifikan. Tombol yang terisi memberikan lebih banyak bobot visual dan fungsi yang sesuai seperti "tambahkan ke keranjang" dan "Login". |
| Outlined | Tonjol dengan memiliki bayangan. | Sesuai dengan peran yang serupa dengan tombol tonal. Tingkatkan elevasi agar tombol terlihat lebih jelas. |
| Elevated | Menampilkan batas tanpa isian. | Tombol penekanan sedang, yang berisi tindakan yang penting tetapi bukan utama. Tombol ini cocok dipasangkan dengan tombol lain untuk menunjukkan tindakan alternatif sekunder seperti "Batal" atau "Kembali". |
| Text Button | Menampilkan teks tanpa latar belakang atau batas. | Tombol dengan penekanan rendah, ideal untuk tindakan yang kurang penting seperti link navigasi, atau fungsi sekunder seperti "Pelajari Lebih Lanjut" atau "Lihat detail". |

# Gambar dan grafik di Compose

Di Android, ada beberapa cara berbeda untuk merender sesuatu secara visual di layar, baik menggunakan vektor atau bitmap ataupun langsung menggambar dengan kanvas di layar

# Memuat gambar

**Memuat gambar dari disk**

Gunakan composable Image untuk menampilkan grafik di layar.
Untuk memuat gambar (misalnya: PNG, JPEG, WEBP) atau aset vektor dari disk,
gunakan painterResource API dengan referensi gambar Anda.
Anda tidak perlu mengetahui jenis aset, cukup gunakan painterResource di pengubah Image atau paint.

```
Image(
    painter = painterResource(id = R.drawable.dog),
    contentDescription = stringResource(id = R.string.dog_content_description)
)
```

# Memuat gambar

**Memuat gambar dari internet**

Untuk memuat gambar dari internet, ada beberapa library pihak ketiga yang tersedia untuk membantu Anda menangani proses ini. Library pemuatan gambar melakukan banyak tugas berat bagi Anda; Library ini menangani caching (sehingga Anda tidak perlu mendownload gambar beberapa kali) dan logika jaringan untuk mendownload gambar dan menampilkannya di layar.

Misalnya, untuk memuat gambar dengan Coil dari Instacart, tambahkan library ke file gradle Anda, dan gunakan AsyncImage untuk memuat gambar dari URL:

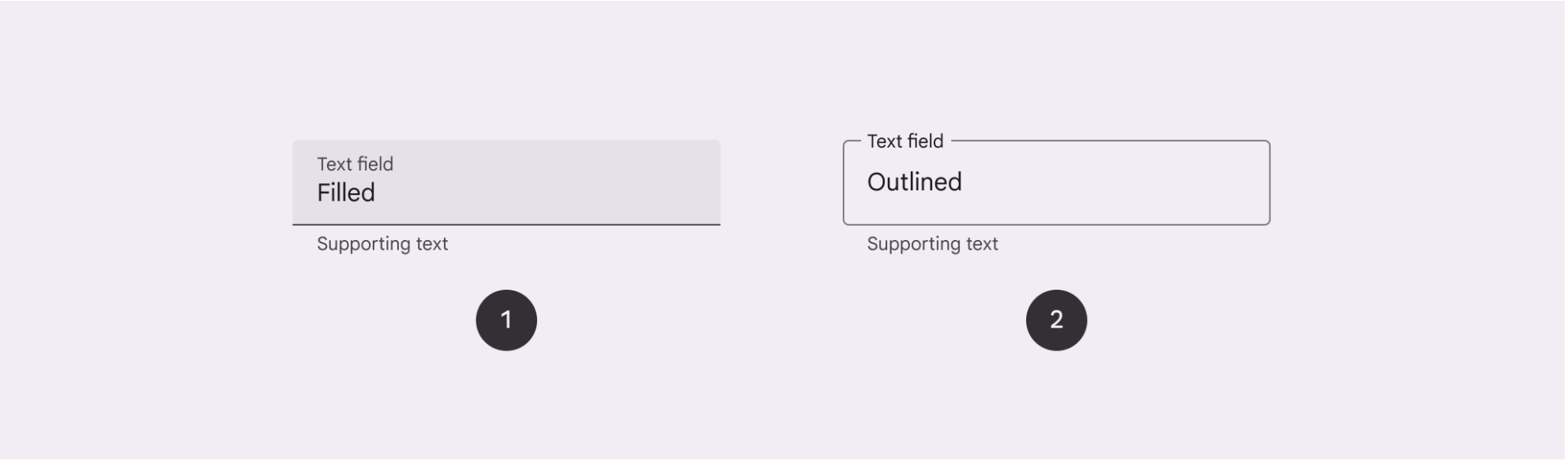
```
AsyncImage(
    model = "https://example.com/image.jpg",
    contentDescription = "Translated description of what the image contains"
)
```

# Menangani input pengguna

TextField memungkinkan pengguna memasukkan dan mengubah teks

TextField adalah penerapan Desain Material :

- Gaya default adalah filled
- OutlinedTextField adalah versi gaya outline



# FilledTextField

```
@Composable
  fun SimpleFilledTextFieldSample() {
     var text by remember { mutableStateOf("Hello") }

     TextField(
        value = text,
        onValueChange = { text = it },
        label = { Text("Label") }
     )
  }
```

# FilledTextField

```
@Composable
fun SimpleOutlinedTextFieldSample() {
    var text by remember { mutableStateOf("") }

    OutlinedTextField(
        value = text,
        onValueChange = { text = it },
        label = { Text("Label") }
    )
}
```

# Thank you

Haris Setyawan