



**MODUL PEMROGRAMAN MOBILE
CIM 430**

**MODUL PERTEMUAN 14
TESTING AND DEPLOYMENT APPLICATIONS**

DISUSUN OLEH :

7174 – SAWALI WAHYU, S.KOM, M.KOM

8126 – JEFRY SUNUPURWA ASRI, S.KOM, M.KOM

**UNIVERSITAS ESA UNGGUL
FAKULTAS ILMU KOMPUTER
TAHUN 2021**

FLUTTER – TESTING AND DEPLOYMENT APPLICATIONS

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu :

1. Mahasiswa mampu Konsep dasar pengujian aplikasi berbasis mobile
2. Mahasiswa mampu memahami macam macam fungsi testing aplikasi
3. Mahasiswa mampu membuat deployment dalam aplikasi mobile android



Testing dan Deployment Application

Pengujian adalah fase yang sangat penting dalam siklus hidup pengembangan suatu aplikasi. Ini memastikan bahwa aplikasi berkualitas tinggi. Pengujian membutuhkan perencanaan dan pelaksanaan yang cermat. Ini juga merupakan fase pengembangan yang paling memakan waktu.

Jenis Pengujian

1. Pengujian Alpha

Pengujian ini adalah pengujian *end-to-end* pertama untuk memastikan produk memenuhi persyaratan bisnis dan berfungsi dengan benar.

Alpha testing dilakukan untuk mengidentifikasi masalah yang tidak terdeteksi selama proses pengembangan. Selain itu, pengujian ini juga dilakukan untuk memahami *usability* dan *reliability* produk.

Ada tiga fase dalam *alpha testing* :

- *Pre-alpha testing*, yaitu fase untuk memastikan sistem aplikasi dapat diteruskan ke fase pengujian berikutnya.
- *Alpha testing*, yakni ketika aplikasi diuji secara keseluruhan. Tim *developer* menguji semua fitur sistem dan mengidentifikasi masalah-masalah yang muncul akibat pemakaian.
- *Post-alpha testing* yang dilakukan secara paralel. Pada fase ini, tim *developer* akan memperbaiki masalah-masalah yang muncul selama pengujian.

Secara umum, tiga jenis proses pengujian Alpha yaitu untuk menguji aplikasi secara teknis sepenuhnya. Mereka adalah sebagai berikut :

1) Pengujian Unit

Pengujian unit adalah metode termudah untuk menguji aplikasi. Ini didasarkan pada memastikan kebenaran sepotong kode (fungsi, secara umum) dari

metode kelas. Tapi, itu tidak mencerminkan lingkungan nyata dan selanjutnya, merupakan pilihan paling sedikit untuk menemukan bug.

Contoh Dokumen Unit Testing :

Identifikasi	BLC-02		
Nama Butir Uji	Menambahkan Data Barang		
Tujuan	Memeriksa apakah data baru barang yang ditambahkan terekam dalam tabel pemesanan atau tidak		
Kondisi Awal	<ul style="list-style-type: none">Tabel Barang sudah adaUser sudah membuka halaman menu untuk penambahan data barang		
Tanggal Pengujian	24 Mei 2013		
Penguji	Pegawai		
Skenario			
1. Ketikan data barang yang akan dibuat			
2. Klik tombol save jika sudah selesai diketik secara lengkap dan benar			
Hasil			
Data yang diberikan	Yang Diharapkan	Pengamatan	Kesimpulan
Nama Barang= Wafer Tango Supplier= PT. Orang Tua Jumlah= 100 Harga= 2500 Satuan= grm Netto= 50	Data barang terekam ke table data barang	<ul style="list-style-type: none">Field ID barang sudah terekam secara otomatisTombol Submit dapat di klik jika semua field data barang sudah terisi secara lengkap dan benarPada <i>database grid</i>, data baru ditempatkan di record terakhir	Ok
Catatan			

Contoh Dokumen Pengujian Unit :

No	No. Jalur	Data Input	Expected Result	Result	Status
1.	1 – 2 – 3 – 4 – 5	File document berextensi pdf atau doc	File terupload dan menampilkan pesan berhasil upload	File Berhasil terupload dan menampilkan pemberitahuan bahwa upload berhasil	Valid
2.	1 – 2 – 5	Tidak ada file document yang dipilih tapi menekan tombol upload	Menampilkan pemberitahuan dan Kembali ke halaman upload	Menampilkan pemberitahuan tetapi dan kembali ke halaman upload	Valid
3	1 – 3 – 4 – 5	File yang di upload tidak berekstensi pdf atau doc	Menampilkan pemberitahuan dan Kembali ke halaman upload	Menampilkan pemberitahuan cek file kembali dan kembali kehalaman upload	Valid

Tabel 5. Tabel Pengujian Unit

2) Pengujian Widget

Pengujian widget didasarkan pada memastikan kebenaran pembuatan widget, rendering, dan interaksi dengan widget lain seperti yang diharapkan. Ini melangkah lebih jauh dan menyediakan lingkungan hampir real-time untuk menemukan lebih banyak bug.

3) Pengujian integrasi

Pengujian integrasi melibatkan pengujian unit dan pengujian widget bersama dengan komponen eksternal aplikasi seperti database, layanan web, dll., Ini mensimulasikan atau mengecek lingkungan nyata untuk menemukan hampir semua bug, tetapi ini adalah proses yang paling rumit.

Contoh Dokumen Pengujian Integrasi :

Test Scenario ID			Test Case ID				
Test Case Description			Test Priority				
Pre-Requisite			Post-Requisite				
Test Execution Steps:							
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments

Flutter memberikan dukungan untuk semua jenis pengujian. Ini memberikan dukungan ekstensif dan eksklusif untuk pengujian Widget. Pada bab ini, kita akan membahas pengujian widget secara detail.

Pengujian Widget

Kerangka pengujian flutter menyediakan metode `testWidgets` untuk menguji widget. Ini menerima dua argumen -

- Deskripsi tes
- Kode uji

```
testWidgets('test description: find a widget', '<test code>');
```

Langkah-langkah yang Terlibat

Pengujian Widget melibatkan tiga langkah berbeda -

- Render widget di lingkungan pengujian.
- `WidgetTester` adalah kelas yang disediakan oleh framework pengujian Flutter untuk membuat dan merender widget. Metode `pumpWidget` dari kelas `WidgetTester` menerima widget apa pun dan merendernya di lingkungan pengujian.

```
testWidgets('finds a specific instance', (WidgetTester tester)
  async {
    await tester.pumpWidget(MaterialApp(
      home: Scaffold(
        body: Text('Hello'),
      ),
    ));
  });
```

- Menemukan widget, yang perlu kita uji.
 - Flutter framework menyediakan banyak opsi untuk menemukan widget yang dirender di lingkungan pengujian dan umumnya disebut Finders. Finder yang paling sering digunakan adalah `find.text`, `find.byKey`, dan `find.byWidget`.
 - `find.text` menemukan widget yang berisi teks yang ditentukan.
`find.text('Hello')`
 - `find.byKey` menemukan widget dengan kunci spesifiknya.
`find.byKey('home')`

- `find.byWidget` menemukan widget dengan variabel instance-nya.

`find.byWidget(homeWidget)`

- Memastikan widget berfungsi seperti yang diharapkan.
- Flutter framework menyediakan banyak opsi untuk mencocokkan widget dengan widget yang diharapkan dan biasanya disebut *Matchers*. Kita bisa menggunakan metode ekspektasi yang disediakan oleh framework pengujian untuk mencocokkan widget, yang kita temukan di langkah kedua dengan widget yang kita harapkan dengan memilih salah satu pencocok. Beberapa pencocokan penting adalah sebagai berikut.
 - `findOneWidget` - memverifikasi bahwa satu widget ditemukan.
`expect(find.text('Hello'), findsOneWidget);`
 - `findNothing` - memverifikasi tidak ada widget yang ditemukan
`expect(find.text('Hello World'), findsNothing);`
 - `findWidgets` - memverifikasi lebih dari satu widget ditemukan.
`expect(find.text('Save'), findsWidgets);`
 - `findNWidgets` - memverifikasi N jumlah widget yang ditemukan.
`expect(find.text('Save'), findsNWidgets(2));`

Kode tes lengkapnya adalah sebagai berikut -

```
testWidgets('finds hello widget', (WidgetTester tester)
  async { await tester.pumpWidget(MaterialApp(
    home: Scaffold(
      body: Text('Hello'),
    ),
  ));
  expect(find.text('Hello'), findsOneWidget);
});
```

Di sini, kami merender widget `MaterialApp` dengan teks Halo menggunakan widget `Teks` di tubuhnya. Kemudian, kami menggunakan `find.text` untuk menemukan widget tersebut dan kemudian mencocokkannya menggunakan `foundOneWidget`.

Contoh Kerja

Mari kita membuat aplikasi flutter sederhana dan menulis tes widget untuk lebih memahami langkah-langkah yang terlibat dan konsepnya.

- Buat aplikasi flutter baru, `flutter_test_app` di Android studio.

- Buka widget_test.dart di folder uji. Ini memiliki contoh kode pengujian seperti yang diberikan di bawah ini -

```
testWidgets('Counter increments smoke test', (WidgetTester
tester) async {
  // Build our app and trigger a frame.
  await tester.pumpWidget(MyApp());

  // Verify that our counter starts at 0.
  expect(find.text('0'), findsOneWidget);
  expect(find.text('1'), findsNothing);

  // Tap the '+' icon and trigger a frame.
  await tester.tap(find.byIcon(Icons.add));
  await tester.pump();

  // Verify that our counter has incremented.
  expect(find.text('0'), findsNothing);
  expect(find.text('1'), findsOneWidget);
});
```

- Di sini, kode pengujian melakukan fungsi berikut -
 - Merender widget MyApp menggunakan tester.pumpWidget.
 - Memastikan bahwa penghitung awalnya nol menggunakan findOneWidget dan findsNothing matchers.
 - Menemukan tombol kenaikan penghitung menggunakan metode find.byIcon.
 - Ketuk tombol kenaikan penghitung menggunakan metode tester.tap.
 - Pastikan penghitung ditingkatkan menggunakan findOneWidget dan findsNothing matchers.
- Mari kita ketuk lagi tombol kenaikan penghitung dan kemudian periksa apakah penghitung dinaikkan menjadi dua.


```
await tester.tap(find.byIcon(Icons.add)); await
tester.pump();
expect(find.text('2'), findsOneWidget);
```
- Klik menu Run.
- Klik tes dalam opsi widget_test.dart. Ini akan menjalankan tes dan melaporkan hasilnya di jendela hasil.



2. Pengujian Beta

Beta testing adalah peluang bagi konsumen untuk menggunakan produk di lingkungan *development*.

Tujuan dari pengujian adalah mengungkap *bug* atau masalah apa pun sehingga dapat diatasi sebelum produk dirilis.

Berbeda dengan *alpha testing*, pengujian pada tahap ini dilakukan untuk mengidentifikasi masalah yang tidak muncul dalam kondisi yang terkontrol. Untuk itulah konsumen dilibatkan sebagai *beta tester*.

Pengujian Secara Penerimaan *User (User Acceptance Testing)*

User Acceptance Testing (UAT) merupakan proses verifikasi bahwa solusi yang dibuat dalam sistem sudah sesuai untuk pengguna.

UAT (User Acceptance Test) adalah suatu proses pengujian yang dilakukan oleh pengguna dengan hasil output sebuah dokumen hasil uji yang dapat dijadikan bukti bahwa software sudah diterima dan sudah memenuhi kebutuhan yang diminta.

Dalam tahapan ini, pengujian sistem dilakukan untuk menentukan apakah sistem telah memenuhi kebutuhan pengguna dan dapat mendukung semua skenario bisnis dan pengguna. **UAT** dilakukan oleh client dan end-user.

Tujuannya adalah untuk memberikan keyakinan bahwa sistem disampaikan memenuhi persyaratan bisnis baik sponsor dan pengguna.

Ada lima jenis *beta testing* yang biasa dilakukan untuk menguji produk.

- 1) *Closeted beta testing*, yakni pengujian yang melibatkan sejumlah pengguna terpilih. Biasanya pengujian ini dibatasi dengan beberapa kriteria tertentu.
- 2) *Open beta testing* yang dilakukan secara terbuka tanpa batasan kriteria tertentu. Pengujian jenis ini biasanya dilakukan sebagai *follow up* dari *closeted beta testing*.
- 3) *Technical beta testing*, dilakukan untuk menemukan bug yang kompleks dan memberikan laporan kepada tim teknis.
- 4) *Focused beta testing*, yakni pengujian yang dilakukan untuk mendapatkan *feedback* seputar fitur produk tertentu. Pengujian ini dilakukan dengan merilis produk ke publik.

Contoh Dokumen UAT :

Dokumen Uji Terima Tirta ERP - Distribusi						Hal : 8 dari 10	
						No Dok :	
						Aplikasi : Tirta ERP - Distribusi	
ID Pengujian	Deskripsi Pengujian	Prosedur Pengujian	Data Masukan	Keluaran yang Diharapkan	Hasil yang Didapat	Hasil Uji	
						Diterima	Diterima dengan Catatan
	Penjualan → Realisasi Pengiriman → Per Kendaraan						
A.3.4.1	Butir Pengujian Laporan Customer → Customer v.1						
	Pengujian Laporan Customer → Customer v.1						
A.3.4.2	Butir Pengujian Laporan Customer → Status Customer v.1						
	Pengujian Laporan Customer → Status Customer v.1						
A.3.4.3	Butir Pengujian Laporan Customer → Rute Pengiriman						
	Pengujian Laporan Customer → Rute Pengiriman						

Project Manager

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. This system helps me to manage my project.				3	1
2. This sytems helps me to manage my project reporting				1	3
3. This sytem helps me to adhere to Best Practice standards of project management				2	2
4. This system allows me to have a good idea of how all my projects are doing					4

Management

5. The World View of the system allows me to have a good idea of how all my projects are doing					4
6. The World View of the system highlights projects that need my attention based on location				1	3
7. The Chart View of the system allows me to have a good idea of how all my projects are doing					4
8. The Chart View of the system highlights projects that need my attention based on project revenue				1	3

3. Pengujian Gamma


Tahap pengujian yang terakhir dilakukan sebelum produk diluncurkan adalah gamma testing. Pengujian ini dilakukan untuk memastikan bahwa produk benar-benar siap untuk diluncurkan ke pasar.

Fokus dari pengujian ini adalah keamanan dan fungsionalitas produk. Oleh karena itu, setelah pengujian ini dilakukan, tidak ada perubahan lain kecuali perubahan krusial yang mempengaruhi fungsi dan keamanan produk.

Contoh Dokumen Pengujian Fungsional :

Functional Test Plan Template

Project Name:
Project Sponsor:
Service Owner:
Project Manager:
Document Date:



SAN FRANCISCO
STATE UNIVERSITY

1. Functional Test Plan Scope	
In Scope	Out of Scope
In Scope <i>List functions that are tested.</i>	Out of Scope <i>List functions that are not tested.</i>

2. Functional Test Plan Assumptions and Constraints	
Functional Test Plan Assumptions	
Assumption <i>List the functional test plan assumptions.</i>	

Functional Test Plan Constraints	
Constraint <i>List the functional test plan constraints.</i>	

U

3. Functional Test Team Roles & Responsibilities		
Name	Roles	Responsibilities
Name <i>List names of people involved in functional testing.</i>		
Name <i>Add more rows if needed.</i>		

4. Functional Test Entry Criteria	
ID	Criteria
4.1	Entry Criteria <i>Factors that must be present to enable the start of the functional test. Example: Testing environment/ data is available.</i>

5. Functional Test Cases	
ID	Test Cases
5.1	Test Case <i>Identify the test cases along with the expected results.</i> <i>Example:</i> Test Procedure: Login with a corporate user account. Username: abc Password: abc Expected Results: An error will be displayed for the wrong credentials.

Sumber : San Francisco State University

Contoh Dokumen Pengujian Keamanan :

CYBER SECURITY THREAT/VULNERABILITY ASSESSMENT

HUMAN THREATS	Impact (0-6)	Probability (0-5)	Score (Impact x Probability)
1. Human Error			
• Accidental destruction, modification, disclosure, or incorrect classification of information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Ignorance: inadequate security awareness, lack of security guidelines, lack of proper documentation, lack of knowledge	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Workload: Too many or too few system administrators, highly pressured users	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Users may inadvertently give information on security weaknesses to attackers	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Incorrect system configuration	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Security policy not adequate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Security policy not enforced	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Security analysis may have omitted something important or be wrong	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. Dishonesty: Fraud, theft, embezzlement, selling of confidential agency information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. Attacks by "social engineering"			
• Attackers may use telephone to impersonate employees to persuade users/administrators to give user name/passwords/modem numbers, etc.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
• Attackers may persuade users to execute Trojan Horse programs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. Abuse of privileges/trust	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
GENERAL THREATS	Impact (0-6)	Probability (0-5)	Score (Impact x Probability)
1. Unauthorized use of "open" computers/Laptops	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. Mixing of test and production data or environments	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. Introduction of unauthorized software or hardware	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

DEPLOYMENT APPLICATION

Deployment adalah kegiatan yang bertujuan untuk menyebarkan **aplikasi** yang telah dikerjakan oleh para pengembang. Penyebarannya dapat melalui beragam cara tergantung dari jenis **aplikasinya**.

Penyebaran atau deployment dalam perangkat lunak dan pengembangan web pada dasarnya bertujuan untuk mem-push, mendorong perubahan atau update (pembaruan) dari satu lingkungan penerapan ke lingkungan penerapan (application environment) lainnya.

PROSES DEPLOYMENT

Proses *deployment* dan penerapan *software* atau perangkat lunak dapat diringkas dalam 3 (tiga) tahapan umum yaitu *preparation* (persiapan), *testing* (pengujian), dan *deployment* itu sendiri.

1) Preparation

Dalam tahap persiapan ini, *developer* atau pengembang harus mengumpulkan semua kode yang akan disebarkan bersama dengan pustaka (*library*) lain seperti file konfigurasi, atau sumber daya yang diperlukan agar aplikasi berfungsi.

2) Testing

Sebelum pembaruan dapat di-*push* ke lingkungan *live*, *update* atau pembaruan harus diterapkan ke server pengujian tempat pembaruan dapat dikenakan serangkaian pengujian otomatis yang telah dikonfigurasi sebelumnya.

3) Deployment

Pengembang dapat menjalankan serangkaian *script* atau skrip untuk memperbarui database yang relevan sebelum perubahan dapat diterapkan.

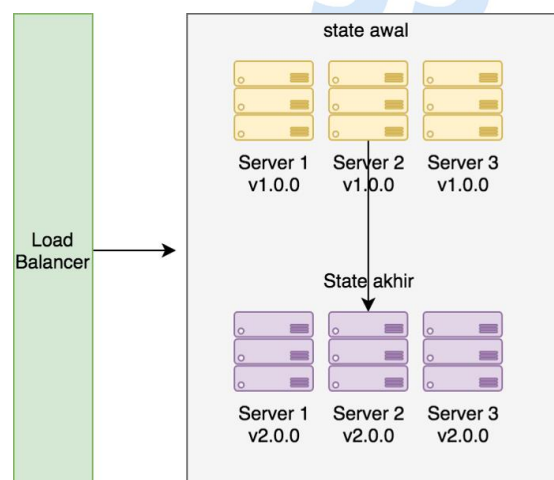
terdapat beberapa jenis deployment strategi yang sudah cukup sering digunakan oleh banyak orang. Yaitu,

1) Big-Bang deployment strategy (atau sering disebut Replace/Recreate deployment strategy).

Hal yang paling mudah dilakukan. Konsepnya sangat simple. Big-Bang strategy, atau sering juga disebut seperti Replace deployment, atau Recreate, dsb. Yang dimana sifatnya menimpa dan mengganti (mereplace) aplikasi yang aktif secara langsung.

Ciri-ciri dari deployment ini adalah,

- 1) Aplikasi yang akan dideploy biasanya sudah dibuat dalam satu package, yang nantinya diupload ke server.
- 2) Package aplikasi yang diupload akan diekstrak dan dijalankan sesuai jenis aplikasi, jika compiled maka binary hasil compilednya dijalankan di server, jika interpreted seperti PHP, Python, maka source codenya di copy dan dijalankan di Webserver.
- 3) Orang yang menggunakan metode ini, biasanya sangat optimis aplikasinya tidak akan down/error.
- 4) Dan jika terjadi down, biasanya terjadi chaos



Kelebihan :

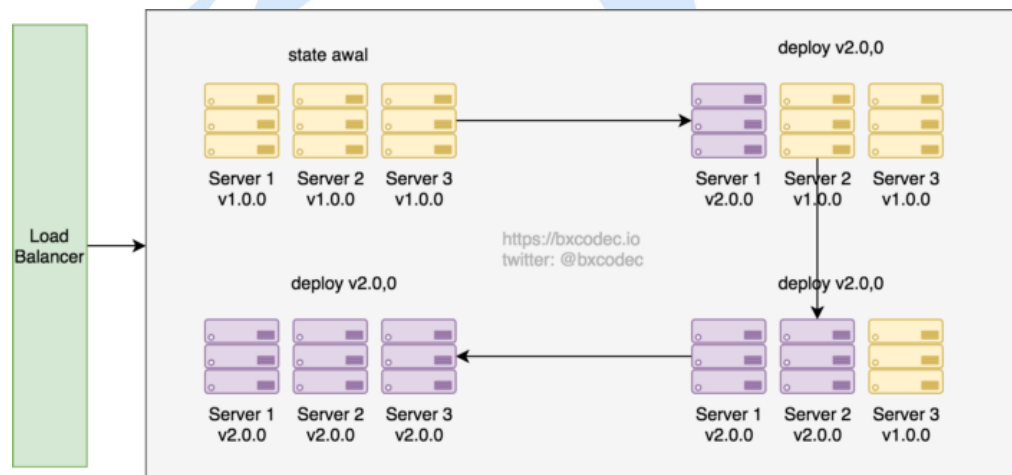
- 1) Mudah di-implementasikan. Cara klasik, tinggal replace.
- 2) Perubahan kepada sistem langsung 100% secara instan.

Kekurangan : Terlalu beresiko, rata-rata *downtime* cukup lama.

2) Rollout deployment strategy.

Rollout deployment jika dianalogikan seperti sebuah bus besar yang memiliki 8 ban(roda) dibelakang. Kiri dan kanan masing-masing ada 4 roda.

Sama halnya ketika *deployment* aplikasi. Dengan metode ini, kita melakukan deployment secara bertahap per-server yang hidup. Dan jika satu server saja langsung error, kita dapat langsung *rollback* tanpa melanjutkan *deploy* kesemua server.



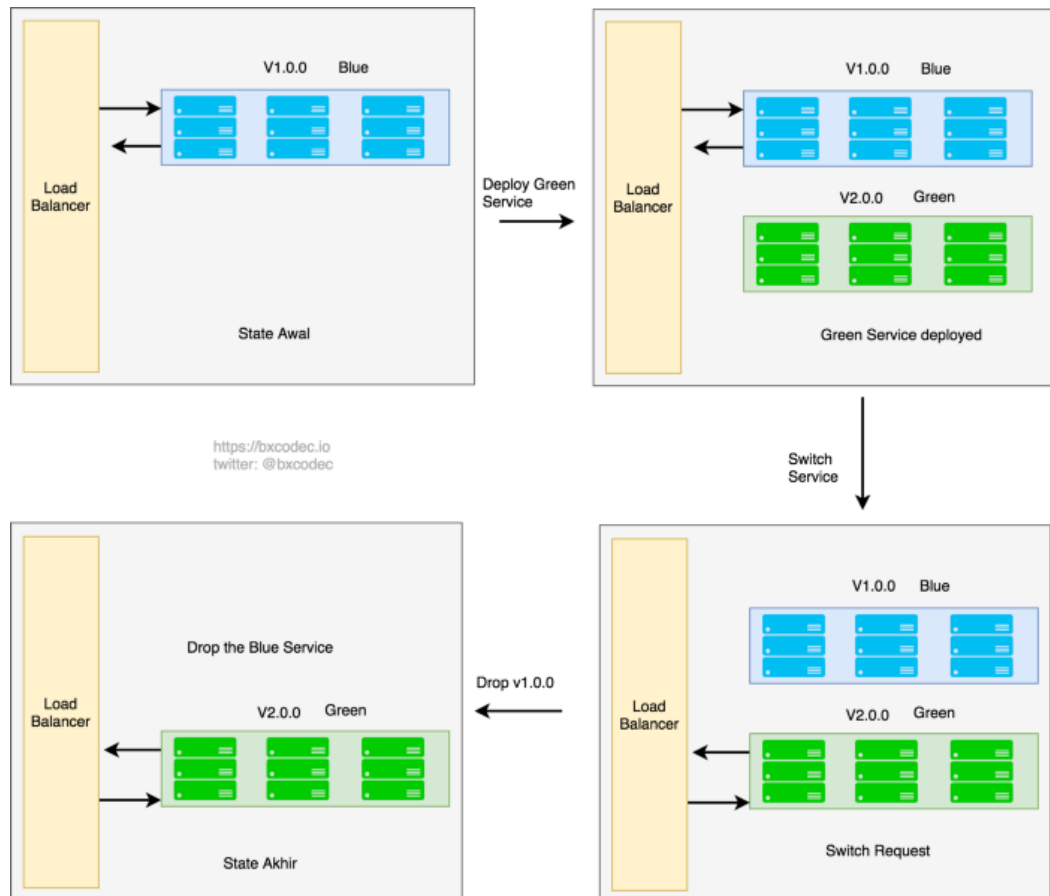
Kelebihan : Lebih aman dan less downtime dari versi sebelumnya

Kekurangan :

- 1) Akan ada 2 versi aplikasi berjalan secara barengan sampai semua server terdeploy, dan bisa membuat bingung. Seperti kita ketahui, untuk management versioning itu sedikit susah dan cukup buat pusing kepala
- 2) Karena sifatnya perlahan satu persatu, untuk deployment dan rollback lebih lama dari yang Bigbang, karena prosesnya perlahan-lahan sampai semua server terkena efeknya.
- 3) Tidak ada kontrol request. Server yang baru ke-deploy dengan aplikasi versi baru, langsung mendapat request yang sama banyaknya dengan server yang lain. Sehingga jika terjadi error, juga dapat menyebabkan kerugian besar.

3) Blue/Green deployment strategy.

Blue/Green Deployment yaitu membuat satu environment yang serupa dengan yang sedang aktif/live, kemudian dapat dilakukan switching request ke environment baru tersebut.



Kelebihan:

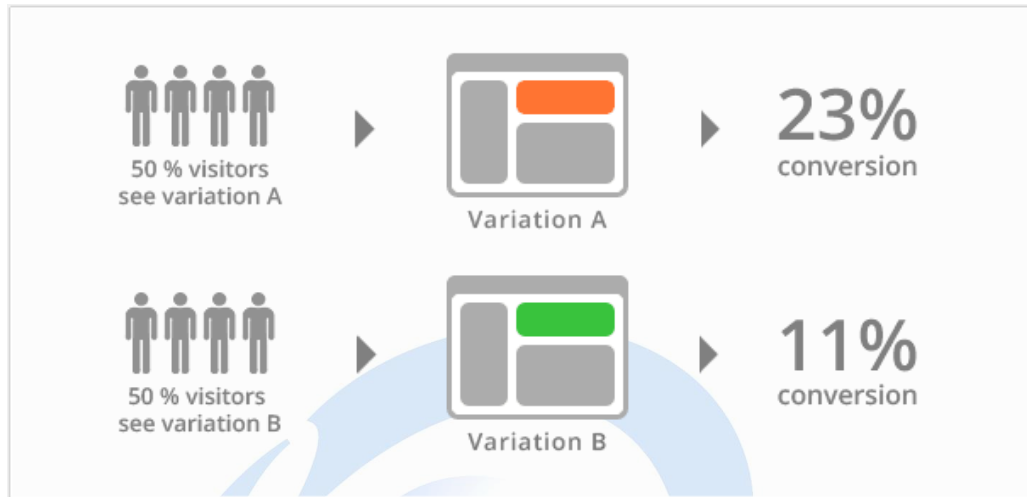
- 1) Perubahan sangat cepat, sekali *switch service* langsung berubah 100%.
- 2) Tidak ada issue beda versi pada *service* seperti yang terjadi pada Rollout Deployment.

Kekurangan:

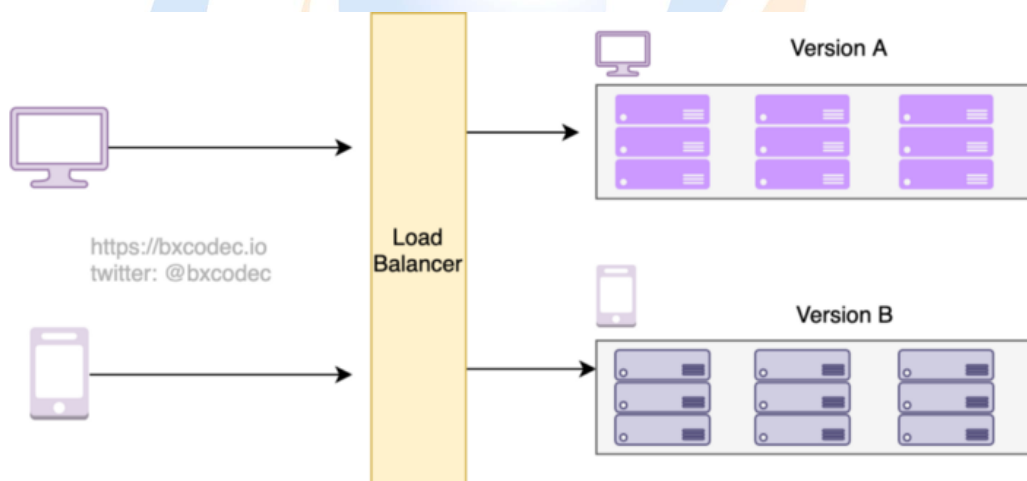
- 1) *Resource* yang dibutuhkan lebih banyak. Karena untuk setiap *deployment* kita harus menyediakan *service* yang serupa *environment*nya dengan yang sedang berjalan di *production*.
- 2) Testing harus benar-benar sangat diprioritaskan sebelum di *switch*, aplikasi harus kita pastikan aman dari *request* yang tiba-tiba banyak.

4) A/B deployment strategy.

A/B deployment lebih ke user sentris. Setengah user akan menerima fitur/layout A dan setengah lagi mendapat fitur/layout B, sehingga setiap user bisa mendapatkan tampilan yang beda.



Pengelompokan fitur/layout tersebut biasanya dibuat berdasarkan spesifikasi yang ada pada user, seperti lokasi, gender, dsb.



Kelebihan : Traffic request terkontrol dan terarah pada masing-masing server

Kekurangan :

- 1) Susah di implementasikan, karena sangat user sentris.
- 2) Troubleshooting jika terjadi error sedikit susah jika logging nya terpusat pada satu tempat. Sehingga untuk kasus A/B testing, setiap server harus memiliki mekanisme dan storage logging tersendiri untuk mempermudah tracing dan troubleshooting jika terjadi error.

Deployment - Aplikasi Android

- Ubah nama aplikasi menggunakan android: label entry di file manifest android. File manifest aplikasi Android, AndroidManifest.xml terletak di <app dir> / android / app / src / main. Ini berisi seluruh detail tentang aplikasi android. Kita bisa mengatur nama aplikasi menggunakan android: label entry.
- Ubah ikon peluncur menggunakan android: entri ikon dalam file manifest.
- Masuk aplikasi menggunakan opsi standar jika perlu.
- Aktifkan Proguard dan Obfuscation menggunakan opsi standar, jika perlu.
- Buat file APK rilis dengan menjalankan perintah di bawah ini –

```
cd /path/to/my/application flutter  
build apk
```

Anda dapat melihat output seperti yang ditunjukkan di bawah ini -

Initializing gradle... 8.6s

Resolving dependencies... 19.9s

Calling mockable JAR artifact transform to create file:

/Users/.gradle/caches/transforms-1/files-1.1/android.jar/

c30932f130afbf3fd90c131ef9069a0b/android.jar with input

/Users/Library/Android/sdk/platforms/android-28/android.jar Running Gradle

task 'assembleRelease'...

Running Gradle task 'assembleRelease'... Done

85.7s

Built build/app/outputs/apk/release/app-release.apk (4.8MB).

- Instal APK di perangkat menggunakan perintah berikut -

```
flutter install
```

- Publikasikan aplikasi ke Google Playstore dengan membuat appbundle dan dorong ke playstore menggunakan metode standar.

```
flutter build appbundle
```

Deployment - Aplikasi iOS

- Daftarkan aplikasi iOS di *App Store Connect* menggunakan metode standar. Simpan = **Bundle ID yang** digunakan saat mendaftarkan aplikasi.
- Perbarui Nama tampilan dalam pengaturan proyek XCode untuk mengatur nama aplikasi.
- Perbarui Bundle Identifier dalam pengaturan proyek XCode untuk mengatur id bundel, yang kami gunakan pada langkah 1.
- Tanda kode seperlunya menggunakan metode standar.
- Tambahkan ikon aplikasi baru seperlunya menggunakan metode standar.
- Hasilkan file IPA menggunakan perintah berikut -
flutter build ios
- Sekarang, Anda dapat melihat output berikut -
Building com.example.MyApp for device (ios-release)...
Automatically signing iOS for device deployment using specified development team in Xcode project:
Running Xcode build.....23.5s
.....
- Uji aplikasi dengan memasukkan aplikasi, file IPA ke TestFlight menggunakan metode standar.
- Terakhir, dorong aplikasi ke *App Store* menggunakan metode standar.

Esa Unggul