



Back End Development 1 ○

+

Sesi 22



**Code Coverage, +
Deployment + Final
Projects**

Code Coverage, Deployment + Final Projects - Sesi 22

Introductions

Codecoverage adalah sebuah tools yang mengukur efektivitas dari unit testing, dan untuk menunjukkan seberapa lengkap code yang telah ditulis yang sesuai dengan process business yang telah dicover oleh unit test.

Dengan coverage kita bisa mengetahui berapa persen kode yang telah di testing dan yang belum, dan untuk mendapatkan hasil yang PASS atau berhasil minimal semua code yang tercover minimal 90%.

Jacoco and SonarQube Jacoco adalah sebuah library opensource untuk aplikasi Java. Dibuat oleh tim Eclemma berdasarkan hasil riset mereka selama bertahun-tahun.

Untuk detail nya bisa diakses di situs nya langsung di <https://www.eclemma.org/jacoco/>. Jadi aplikasi Jacoco akan melakukan generate result dari codecoverage yang telah kita buat pada unit testing.

Java Code Coverage for Eclipse

JaCoCo Java Code Coverage Library

JaCoCo is a free code coverage library for Java, which has been created by the Eclemma team based on the lessons learned from using and integration existing libraries for many years.



oeg jacoco report

Element	Instruction Coverage	Missed Classes	Missed Methods	Missed Blocks	Missed Lines
org.jacoco.agent.jar	91%	10 / 20	14 / 128	80 / 214	117 / 380
org.jacoco.agent.lib	91%	0 / 0	0 / 40	0 / 98	0 / 105
org.jacoco.core	70%	1 / 7	0 / 26	12 / 58	28 / 90
org.jacoco.core.lib	96%	2 / 30	0 / 40	10 / 58	57 / 140
org.jacoco.core.runtime	97%	1 / 1	1 / 7	0 / 40	2 / 35
Total	84%	14 / 58	14 / 234	102 / 470	204 / 750

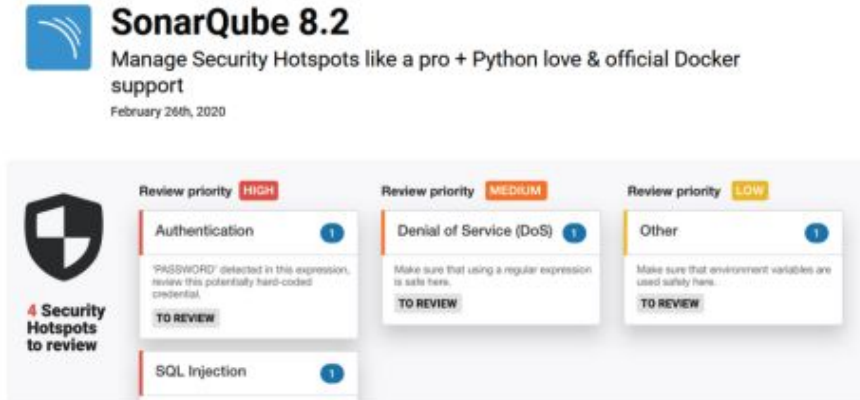
Code Coverage, Deployment + Final Projects - Sesi 22

Sonarqube

Sonarqube adalah sebuah tool yang dilakukan untuk menginspeksi kualitas dari code yang telah ditulis untuk melihat bug, code smells, security vulnerabilities.

Hasil report yang dihasilkan dari jacoco kemudian akan dibaca oleh Sonarqube yang selanjutnya akan ditampilkan secara informative menggunakan webbrowser.

Pada pertemuan ini kita menggunakan Sonarqube free edition sonarqube-7.9.1 Community Edition dan kalian bisa unduh secara langsung pada situs nya <https://www.sonarqube.org/downloads/>



SonarQube 8.2
Manage Security Hotspots like a pro + Python love & official Docker support
February 26th, 2020

4 Security Hotspots to review

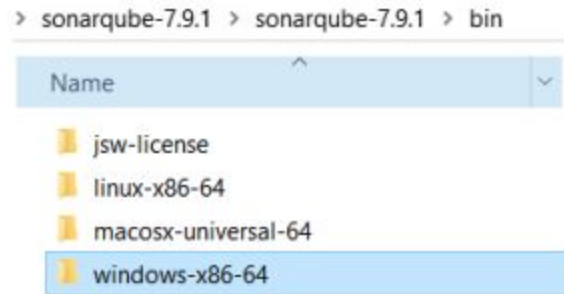
Review priority	HIGH	MEDIUM	LOW
Authentication	1	Denial of Service (DoS)	1
TO REVIEW			
SQL Injection	1		Other
TO REVIEW			

Code Coverage, Deployment + Final Projects - Sesi 22

Setting Integration with Jacoco & Sonarcube

Pada sesi ini akan menggunakan kembali code yang telah dibuat unit testing pada sesi sebelumnya yang fokus pada unit testing service layer dan controller layer.

Setelah Sonarqube di unduh kemudian ekstrak ke dalam folder. Untuk menjalankan server sonarqube masuk ke direktori sonarqube di bin/ , disana terdapat tipe dari sonarqube sesuai dengan system operasi, pada sesi kali ini kita akan menggunakan **windows-x86-64**.



Klik file StartSonar.bat dan akan muncul window berupa command line untuk melihat log dari server sonarqube.

```
wrapper | --> Wrapper Started as Console  
wrapper | Launching a JVM...
```

```
jvm 1 | Wrapper (Version 3.2.3) http://wrapper.tanukisoftware.org  
jvm 1 | Copyright 1999-2006 Tanuki Software, Inc. All Rights Reserved.  
jvm 1 |  
jvm 1 | 2021.12.02 13:12:16 INFO app[][o.s.a.AppFileSystem] Cleaning or creating temp directory  
D:\DEV_SOFTWARE\sonarqube-7.9.1\sonarqube-7.9.1\temp
```

.....

```
jvm 1 | 2021.12.02 13:13:00 INFO app[][o.s.a.SchedulerImpl] Process[web] is up  
jvm 1 | 2020.03.02 13:13:00 INFO app[][o.s.a.ProcessLauncherImpl] Launch process[[key='ce', ipcIndex=3,  
logFilenamePrefix=ce]] from [D:\DEV_SOFTWARE\sonarqube-7.9.1\sonarqube-7.9.1]: C:\Program  
Files\AdoptOpenJDK\jdk-11.0.5.10-hotspot\bin\java -Djava.awt.headless=true - Dfile.encoding=UTF-8  
-Djava.io.tmpdir=D:\DEV_SOFTWARE\sonarqube-7.9.1\sonarqube-7.9.1\temp  
--add-opens=java.base/java.util=ALL-UNNAMED -Xmx512m -Xms128m -  
XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|127.*|[:1] -cp  
./lib/common/*;D:\DEV_SOFTWARE\sonarqube-7.9.1\sonarqube-7.9.1\lib\jdbc\h2\h2-1.3.176.jar  
org.sonar.ce.app.CeServer  
D:\DEV_SOFTWARE\sonarqube-7.9.1\sonarqube-7.9.1\temp\sqprocess13396780188926114200properties
```



jvm 1 | 2021.12.02 13:13:09 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up

jvm 1 | 2021.12.02 13:13:09 INFO app[][o.s.a.SchedulerImpl] SonarQube is up

pada potongan log diatas di baris terakhir, terlihat bahwa server sonarqube sudah berhasil running.

Tambahkan Jacoco Plugin pada file Pom.xml

Tambahkan properties berikut dalam file pom.xml proj

```
<jacoco.version>0.8.3</jacoco.version>
<sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
<sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>
<sonar.jacoco.reportPath>${project.basedir}/../target/jacoco.exec</sonar.jacoco.
reportPath>
<sonar.language>java</sonar.language>
```



Setelah itu, masih dalam file pom.xml tambahkan plugin Jacoco didalam tag plugin.

```
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>${jacoco.version}</version>
  <configuration>
    <skip>${maven.test.skip}</skip>
    <destFile>${basedir}/target/coverage-reports/jacoco-unit.exec</destFile>
    <dataFile>${basedir}/target/coverage-reports/jacoco-unit.exec</dataFile>
    <output>file</output>
    <append>true</append>
    <excludes>
      <exclude>*MethodAccess</exclude>
    </excludes>
  </configuration>
  <executions>
    <execution>
      <id>jacoco-initialize</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
      <phase>test-compile</phase>
    </execution>
    <execution>
      <id>jacoco-site</id>
      <phase>verify</phase>
      <goals>
        <goal>report</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```



file pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.12.RELEASE</version>
    <relativePath/> <!-- Lookup parent from repository -->
  </parent>
  <groupId>id.learn</groupId>
  <artifactId>webservices-restful</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>webservices-restful</name>
  <description>Demo project for Spring Boot</description>

  <properties>
    <java.version>1.8</java.version>
    <problem-spring-web.version>0.25.0</problem-spring-web.version>

    <jacoco.version>0.8.3</jacoco.version>
    <sonar.java.coveragePlugin>jacoco</sonar.java.coveragePlugin>
    <sonar.dynamicAnalysis>reuseReports</sonar.dynamicAnalysis>

    <sonar.jacoco.reportPath>${project.basedir}/../target/jacoco.exec</sonar.jacoco.
reportPath>
    <sonar.language>java</sonar.language>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
```



```

        <optional>true</optional>
      </dependency>

      <!-- Random String generator -->
      <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.9</version>
      </dependency>

      <!-- For Testing-->
      <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
        <exclusions>
          <exclusion>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
          </exclusion>
        </exclusions>
      </dependency>
      <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <scope>test</scope>
      </dependency>
      <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-params</artifactId>
        <scope>test</scope>
      </dependency>
      <dependency>
        <groupId>org.junit.platform</groupId>
        <artifactId>junit-platform-launcher</artifactId>
        <version>1.3.2</version>
        <scope>test</scope>
      </dependency>
      <dependency>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
        <version>5.3.2</version>
        <scope>test</scope>
      </dependency>
      <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <scope>test</scope>
      </dependency>

      <!-- mapper -->
      <dependency>
        <groupId>org.zalando</groupId>
        <artifactId>problem-spring-web-starter</artifactId>
        <version>${problem-spring-web.version}</version>
        <type>pom</type>
      </dependency>
    </dependencies>

```



```

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
    <plugin>
      <groupId>org.jacoco</groupId>
      <artifactId>jacoco-maven-plugin</artifactId>
      <version>${jacoco.version}</version>
      <configuration>
        <skip>${maven.test.skip}</skip>
        <destFile>${basedir}/target/coverage-reports/jacoco-
unit.exec</destFile>
        <dataFile>${basedir}/target/coverage-reports/jacoco-
unit.exec</dataFile>
        <output>file</output>
        <append>true</append>
        <excludes>
          <exclude>*MethodAccess</exclude>
        </excludes>
      </configuration>
      <executions>
        <execution>
          <id>jacoco-initialize</id>
          <goals>
            <goal>prepare-agent</goal>
          </goals>
          <phase>test-compile</phase>
        </execution>
        <execution>
          <id>jacoco-site</id>
          <phase>verify</phase>
          <goals>
            <goal>report</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>

```



Konfigurasi diatas adalah konfigurasi yang dibutuhkan jacoco untuk menggenerate codecoverage.

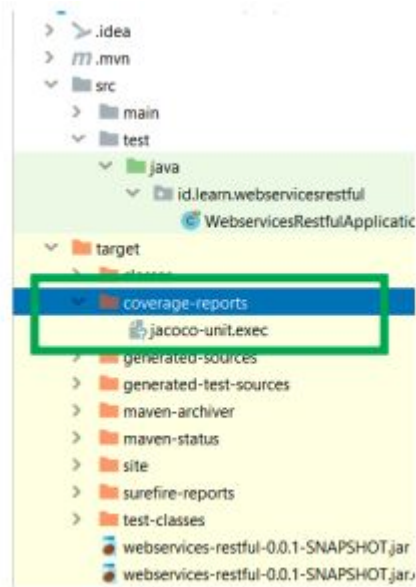
Buka terminal atau command prompt dan build projeknya dengan sintak berikut dan tunggu sampai proses build selesai.

mvn clean install

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 25.581 s  
[INFO] Finished at: 2020-03-02T13:20:39+07:00  
[INFO] -----
```

Kembali ke IDE, setelah projek di build lihat pada struktur folder nya di bawah folder target, terdapat folder coverage-reports yang berisi jacoco-unit.exec.

file itulah yang nantinya akan dibaca oleh Sonarqube untuk menginspeksi code yang sudah dibuat.



Code Coverage, Deployment + Final Projects - Sesi 22

Integrate Jacoco & Sonarqube

Masih dalam terminal atau command prompt yang sama, kemudian ketikkan sintak berikut untuk melakukan generate code quality.

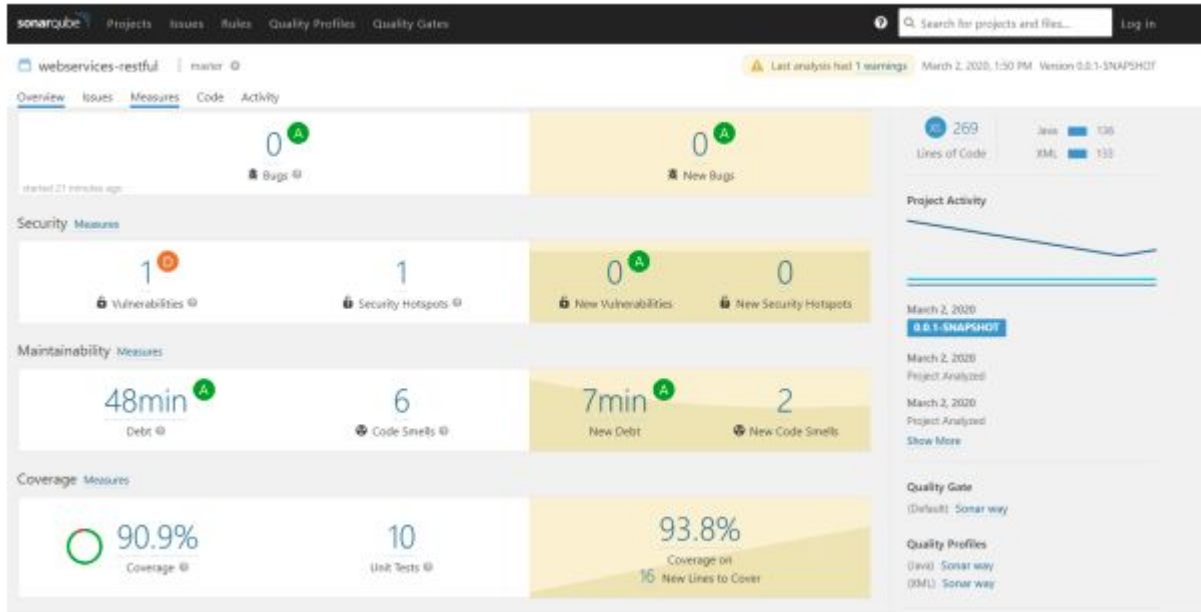
mvn sonar:sonar

tunggu sampai selesai, sampai keluar link yang bisa kita gunakan untuk mengakses codecoverage report.

```
[INFO] Sensor XML Sensor [xml] (done) | time=149ms
[INFO] 1/1 source files have been analyzed
[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=4ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=57ms
[INFO] No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
[INFO] 5 files had no CPD blocks
[INFO] Calculating CPD for 4 files
[INFO] CPD calculation finished
[INFO] Analysis report generated in 109ms, dir size=100 KB
[INFO] Analysis report compressed in 88ms, zip size=30 KB
[INFO] Analysis report uploaded in 622ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=id.learn%3Awebservices-restful
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/api/ce/task?id=KACZ72K0THMARRK3B5M1D
[INFO] Analysis total time: 9.691 s
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:06 min
```



Copy link tersebut kemudian paste pada browser



Pada windows tersebut kita bisa melihat informasi dari code yang telah ditulis, mulai dari security, maintainability dan coverage. Terlihat codecoverage bernilai 90.9% yang artinya bahwa kode kita sudah bisa dikatakan berkualitas. Jika angka tersebut di klik maka akan masuk ke halaman detail nya.

webservices-restful | master

Last analysis had 1 warnings March 2, 2020, 1:50 PM Version 0.0.1-SNAPSHOT

Overview Issues Measures **Code** Activity

Search for files...

webservices-restful » src » main/java/ld/learn/websearvicesrestful

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
main/java/ld/learn/websearvicesrestful	136	0	1	5	1	90.9%	0.0%
controller	34	0	1	1	0	100%	0.0%
model	49	0	0	1	0	93.3%	0.0%
repository	5	0	0	0	0	—	0.0%
service	39	0	0	3	0	100%	0.0%
WebServicesRestfulApplication.java	9	0	0	0	1	33.3%	0.0%

Dan untuk melihat lebih detail lagi, bisa menekan folder-folder diatas misalkan melihat codecoverage pada class ProductServiceImpl.

webservices-restful | master

Last analysis had 1 warnings March 2, 2020, 1:50 PM Version 0.0.1-SNAPSHOT

Overview Issues Measures **Code** Activity

Search for files...

webservices-restful » src » main/java/ld/learn/websearvicesrestful » service » impl

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
impl	30	0	0	2	0	100%	0.0%
ProductServiceImpl.java	30	0	0	2	0	100%	0.0%



	Lines	Coverage	Bug	Vulnerability	Code Smell	Security Hotspot
src/_webservicestestful/service/impl/ProductServiceImpl.java	40	100%	0	0	2	0

```
1 package id.learn.webservicestestful.service.impl;
2
3 import id.learn.webservicestestful.model.Product;
4 import id.learn.webservicestestful.repository.ProductRepository;
5 import id.learn.webservicestestful.service.ProductService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 import java.util.List;
10
11 @Service
12 public class ProductServiceImpl implements ProductService {
13
14     @Autowired
15     ProductRepository productRepository;
16
17     @Override
18     public List<Product> findAllProducts() {
19         return productRepository.findAll();
20     }
21
22     @Override
23     public Product findById(Long id) {
24         Product product = productRepository.findById(id).orElse(null);
25
26         return product;
27     }
28
29     @Override
30     public Product saveOrUpdateProduct(Product product) {
31         return productRepository.save(product);
32     }
33
34     @Override
35     public void deleteProduct(Long id) {
36         Product product = productRepository.findById(id).orElse(null);
37         productRepository.delete(product);
38     }
39 }
40
```





Deployment

+



Apa itu Deployment

Apa itu Deployment ?

Sebuah Istilah untuk kegiatan menyebarkan aplikasi baik itu website maupun mobile apps yang telah selesai dikerjakan oleh programmer.

Secara tidak sadar teman-teman telah melakukan deployment tetapi pada LOCAL Environment yaitu <http://127.0.0.1:8080>

Nah Deployment juga bisa dilakukan pada environment cloud : entah itu AWS/Google Cloud dan banyak lagi

Pada sesi kali ini kita akan menggunakan environment cloud yang di kenal dengan nama HEROKU.

Apa itu Heroku ?

Cloud Platform yang mendukung bahasa pemrograman seperti node JS, PHP, Ruby, Python, Java dan banyak lagi.

Kenapa HEROKU ?

Heroku dapat terintegrasi dengan database dimana cocok dengan yang sudah dipelajari sejauh ini.



Cara kerja nya bagaimana ?

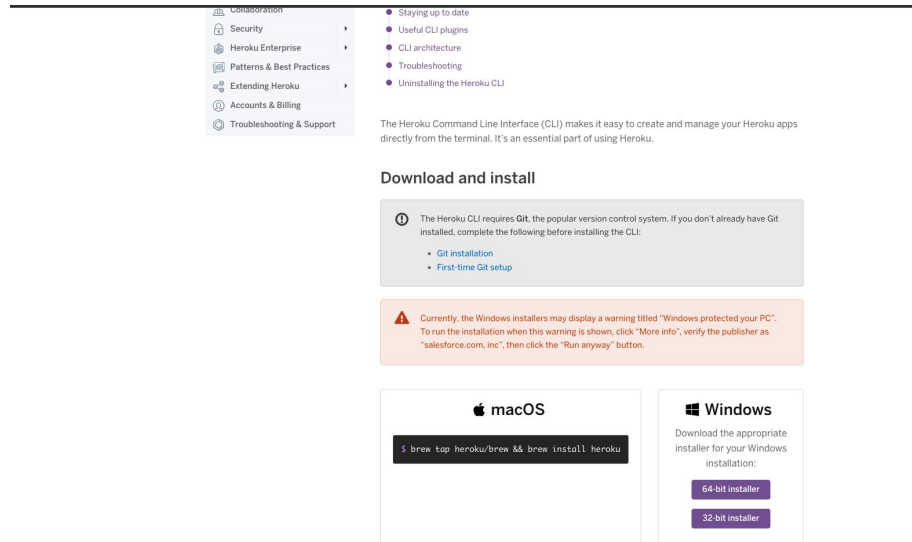
Selain Heroku terintegrasi dengan database juga terintegrasi langsung dengan GITHUB.

Dimana ketika kita melakukan push ke repository github maka heroku dapat langsung men-deploy aplikasi kita sehingga bisa dilihat oleh semua orang yang memiliki akses internet.

Mudah Bukan ?

Oh iya silahkan daftar dulu sebagai member di Heroku <https://signup.heroku.com/> ya.

Dan lakukan installasi HEROKU CLI <https://devcenter.heroku.com/articles/heroku-cli>



The screenshot shows the Heroku CLI installation page. On the left is a navigation menu with links: Collaboration, Security, Heroku Enterprise, Patterns & Best Practices, Extending Heroku, Accounts & Billing, and Troubleshooting & Support. The main content area has a list of links: Staying up to date, Useful CLI plugins, CLI architecture, Troubleshooting, and Uninstalling the Heroku CLI. Below this is a paragraph explaining that the Heroku CLI makes it easy to create and manage apps directly from the terminal. A section titled 'Download and install' contains a warning that Git is required and links to 'Git installation' and 'First-time Git setup'. Another warning states that Windows installers may display a 'Windows protected your PC' message and advises verifying the publisher as 'salesforce.com, inc.'. At the bottom, there are two boxes for 'macOS' and 'Windows'. The macOS box shows the command `$ brew tap heroku/brew && brew install heroku`. The Windows box instructs to download the appropriate installer and provides buttons for '64-bit installer' and '32-bit installer'.

Code Coverage, Deployment + Final Projects - Sesi 22

Integrate Jacoco & Sonarqube

We need to use ClearDB Mysql Addon on Heroku

First, we prepare the Spring Boot for heroku like so:

- git init to initialise the repository
- git add . to add all files in folder
- git commit -m 'Commit name': commit changes
- heroku create: to create heroku app

Next, you need to add mysql DB like so:

```
heroku addons:create cleardb:ignite
```

Now that you have add clearDB, you need the connection url. To get this, type like so:

```
heroku config
```

It should return your connection string.

You need to replace the `spring.datasource.url` value with the value return from heroku config.

One last thing though, to make it work, add a database config file like so:

```
3  import com.zaxxer.hikari.HikariConfig;
4  import com.zaxxer.hikari.HikariDataSource;
5  import org.springframework.beans.factory.annotation.Value;
6  import org.springframework.context.annotation.Bean;
7  import org.springframework.context.annotation.Configuration;
8
9  import javax.sql.DataSource;
10
11 @Configuration
12 public class DatabaseConfig {
13
14     @Value("${spring.datasource.url}")
15     private String dbUrl;
16
17     @Bean
18     public DataSource dataSource() {
19         HikariConfig config = new HikariConfig();
20         config.setJdbcUrl(dbUrl);
21         return new HikariDataSource(config);
22     }
23 }
24
25 public class config {
```



It's time to deploy to heroku

```
git push heroku master
```

At the end of this process, you should have your app deployed to Heroku
To test, you remove the localhost and port number. For example to get all bucketlists on a deployed app, the url is like this:

```
https://tranquil-mountain-81706.herokuapp.com/
```

