



**Release with confidence**

ANDROID



บริษัท สยามชานาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Your goal ?

Release with confidence

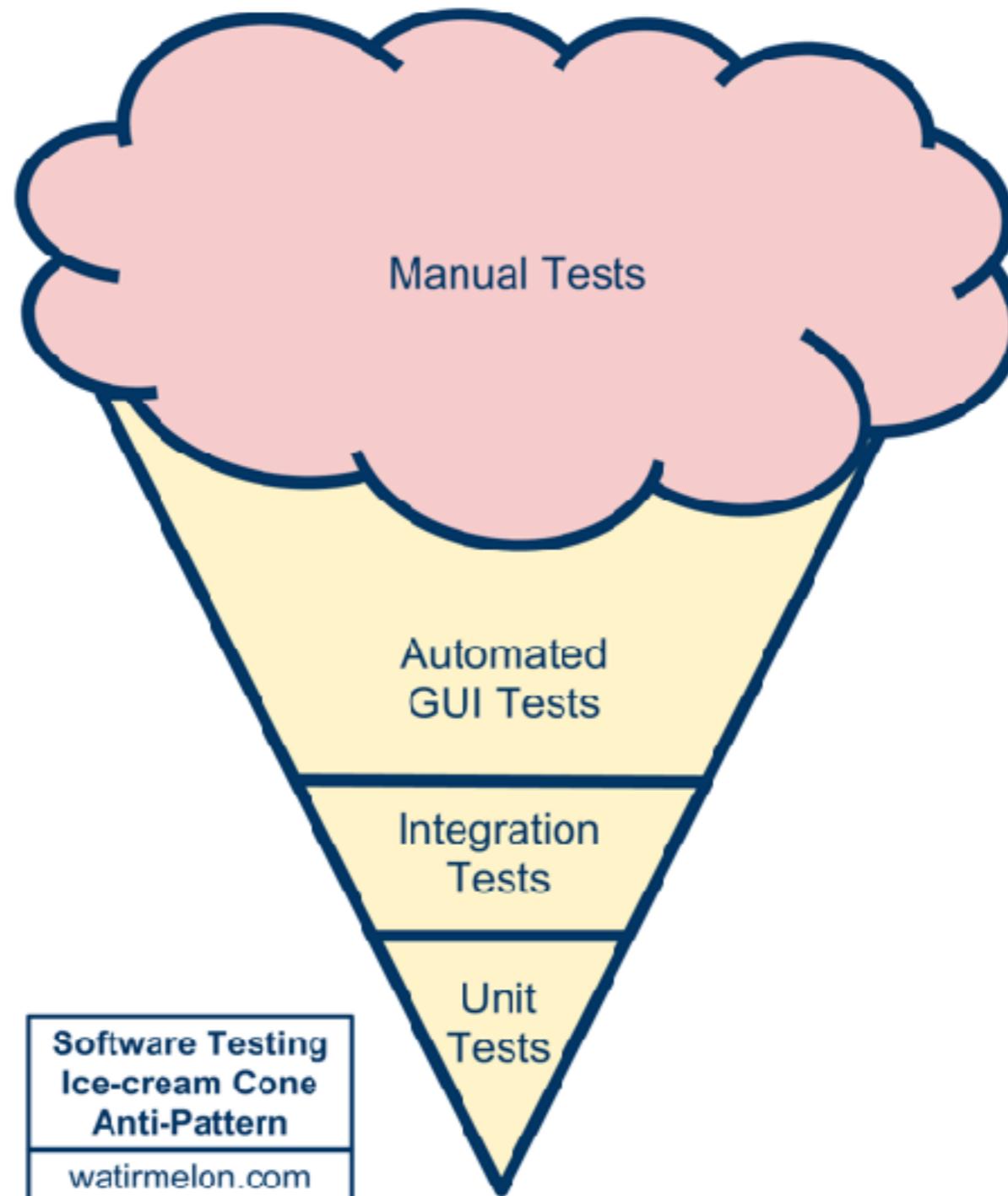


บริษัท สยามชานาญกิจ จำกัด และเพื่อนพ้องน้องพี่

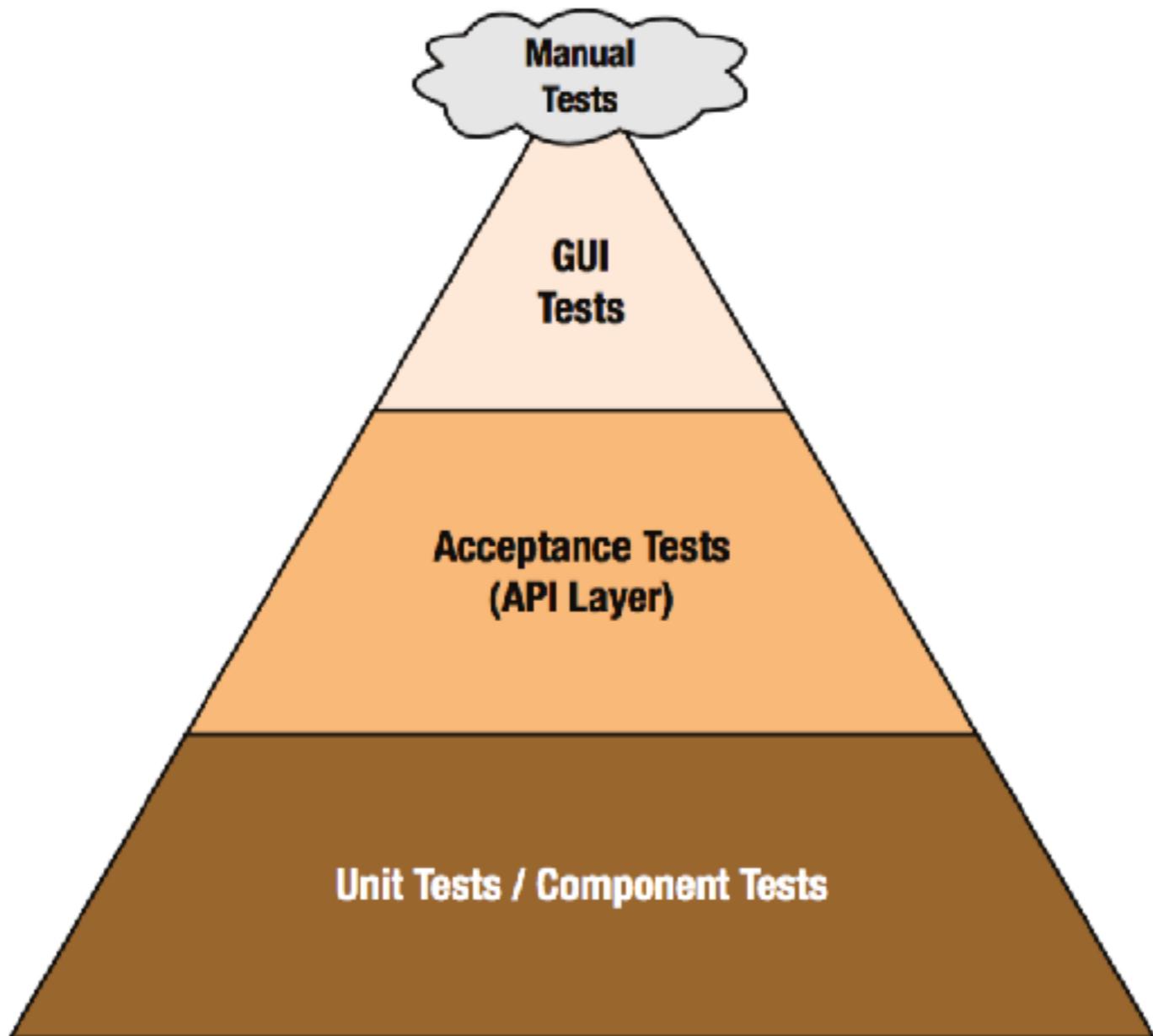
# Testing



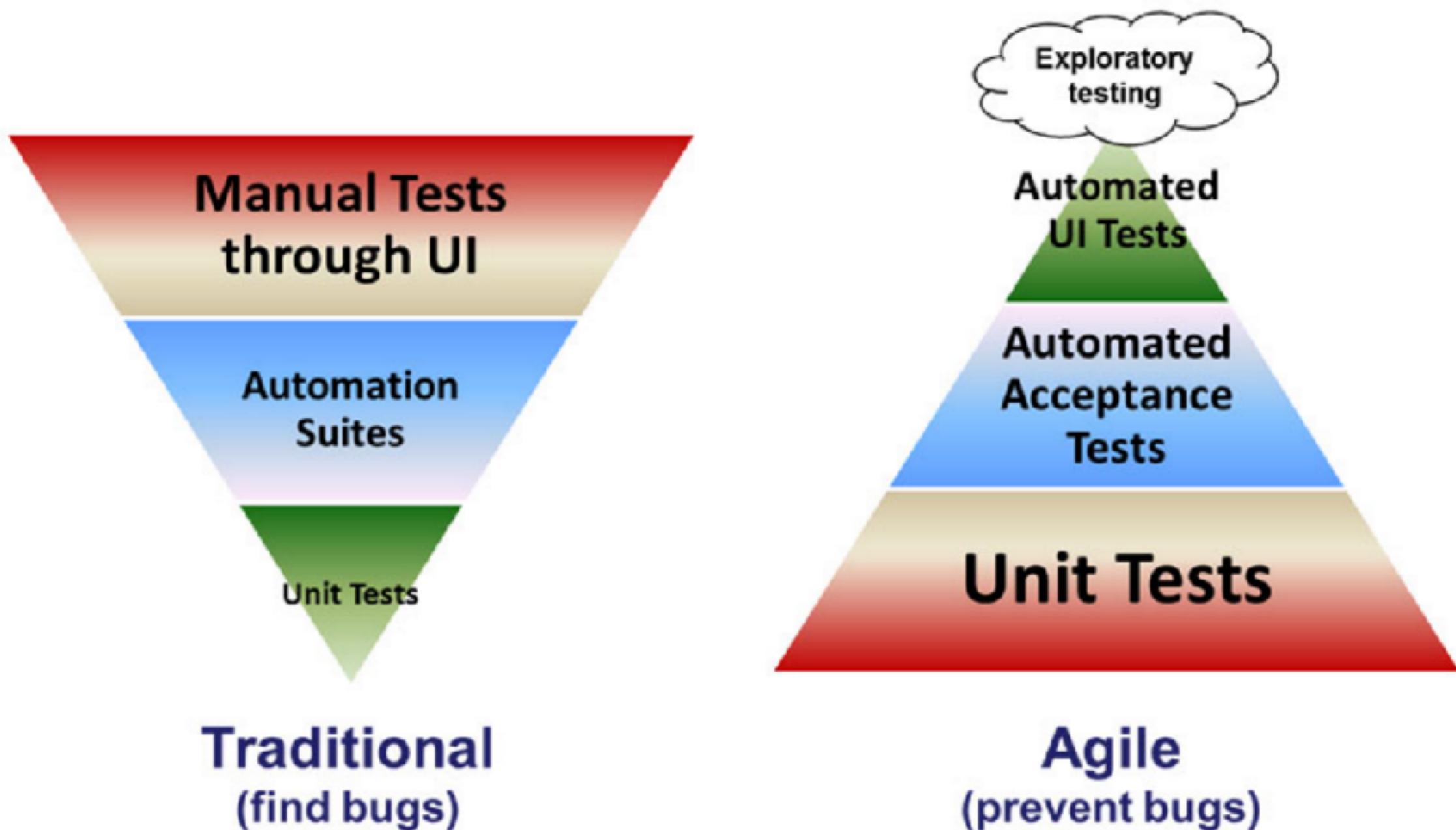
# Testing



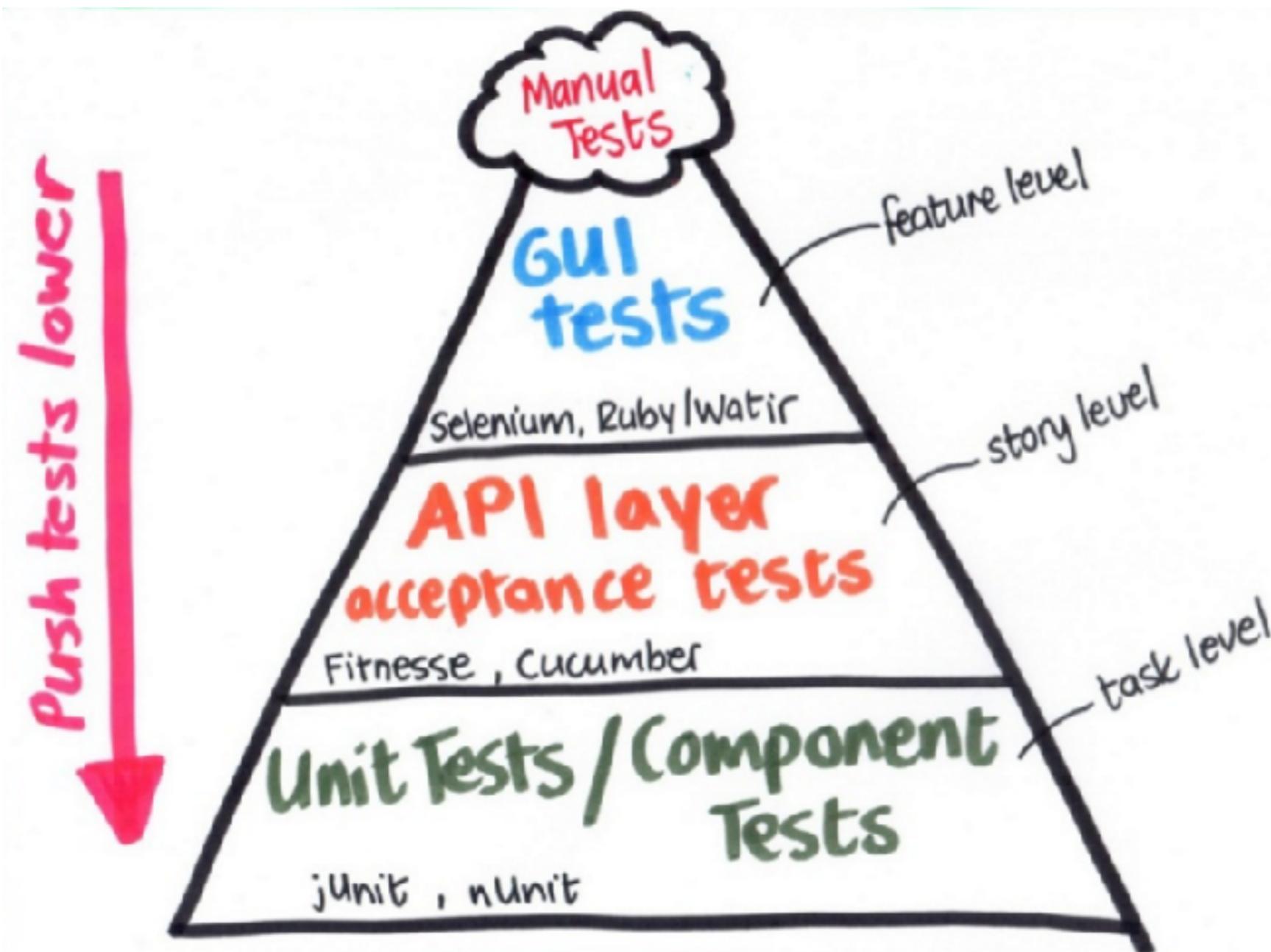
# Testing pyramid



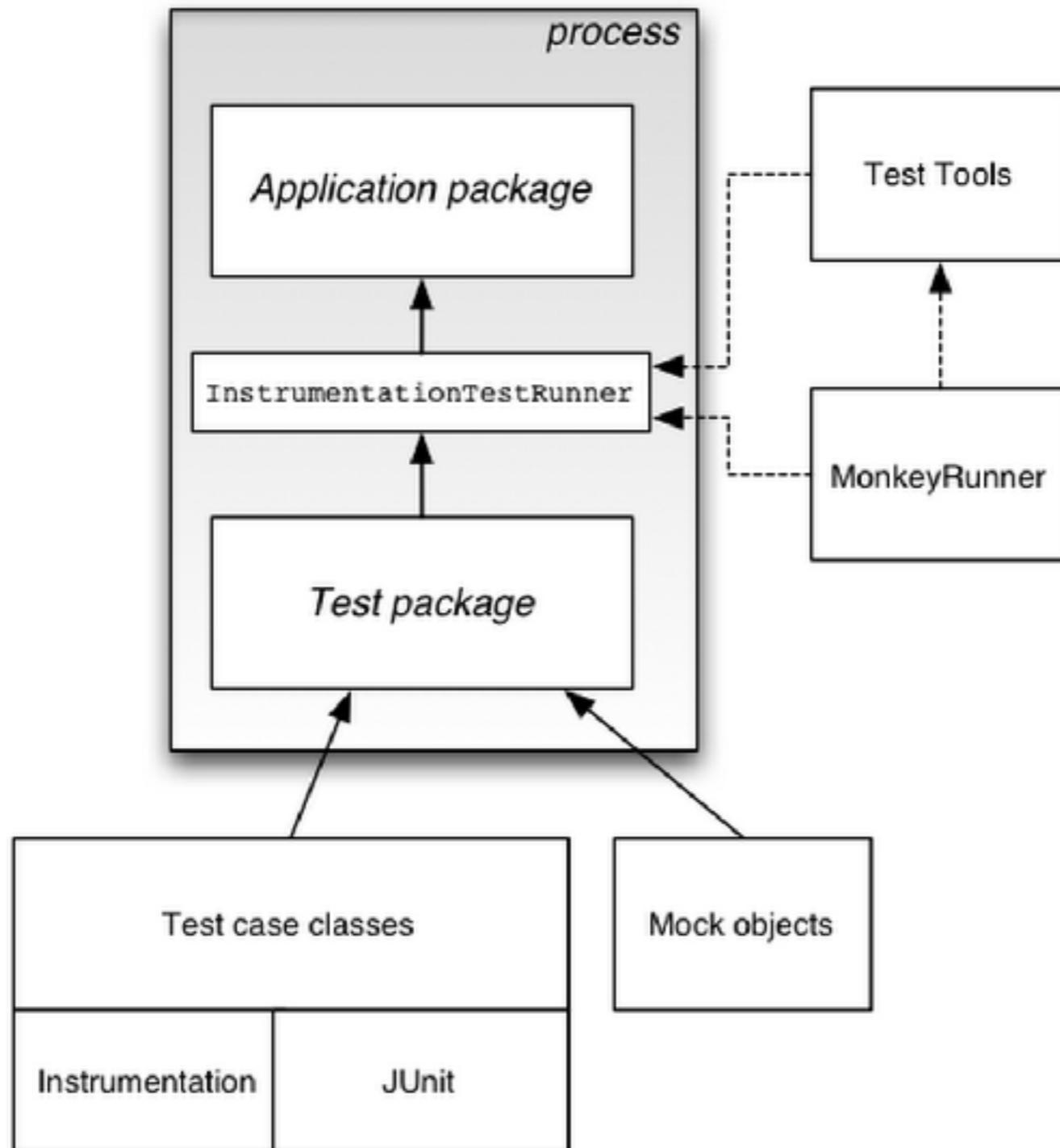
# Testing pyramid



# Testing pyramid



# Testing for android



# Testing for android

1. ต้องการ Emulator/Device
2. ไม่ต้องการ Emulator/Device



# 3rd-party tools



Hamcrest



Jenkins

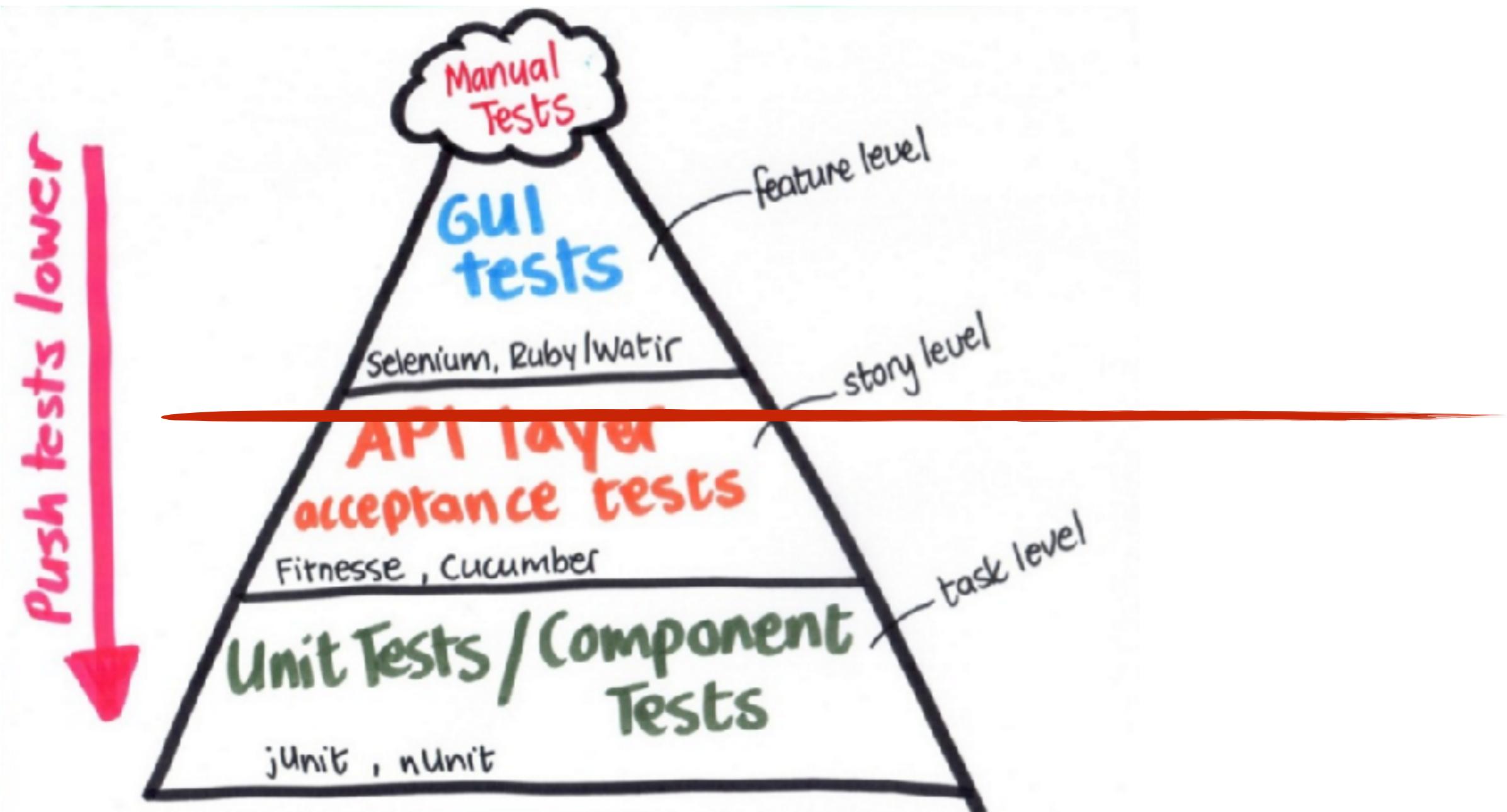
Stubby4J



JFrog Artifactory



# Testing for android



# Test sizes

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

<http://googletesting.blogspot.com/2010/12/test-sizes.html>



# We need small test !!

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

<http://googletesting.blogspot.com/2010/12/test-sizes.html>



# Unit testing

Code เพื่อใช้ทดสอบ Code



# Unit testing

ไม่ต้องการ Emulator ในการทดสอบ  
เร็วแน่นอน !!



# Why Unit testing

Catch more mistakes  
earlier in the development process



# Why Unit testing

Confidently make more changes



# Why Unit testing

Build in regression testing



# Why Unit testing

Extend the life of your codebase



# Good Unit Testing

Fast

Isolated

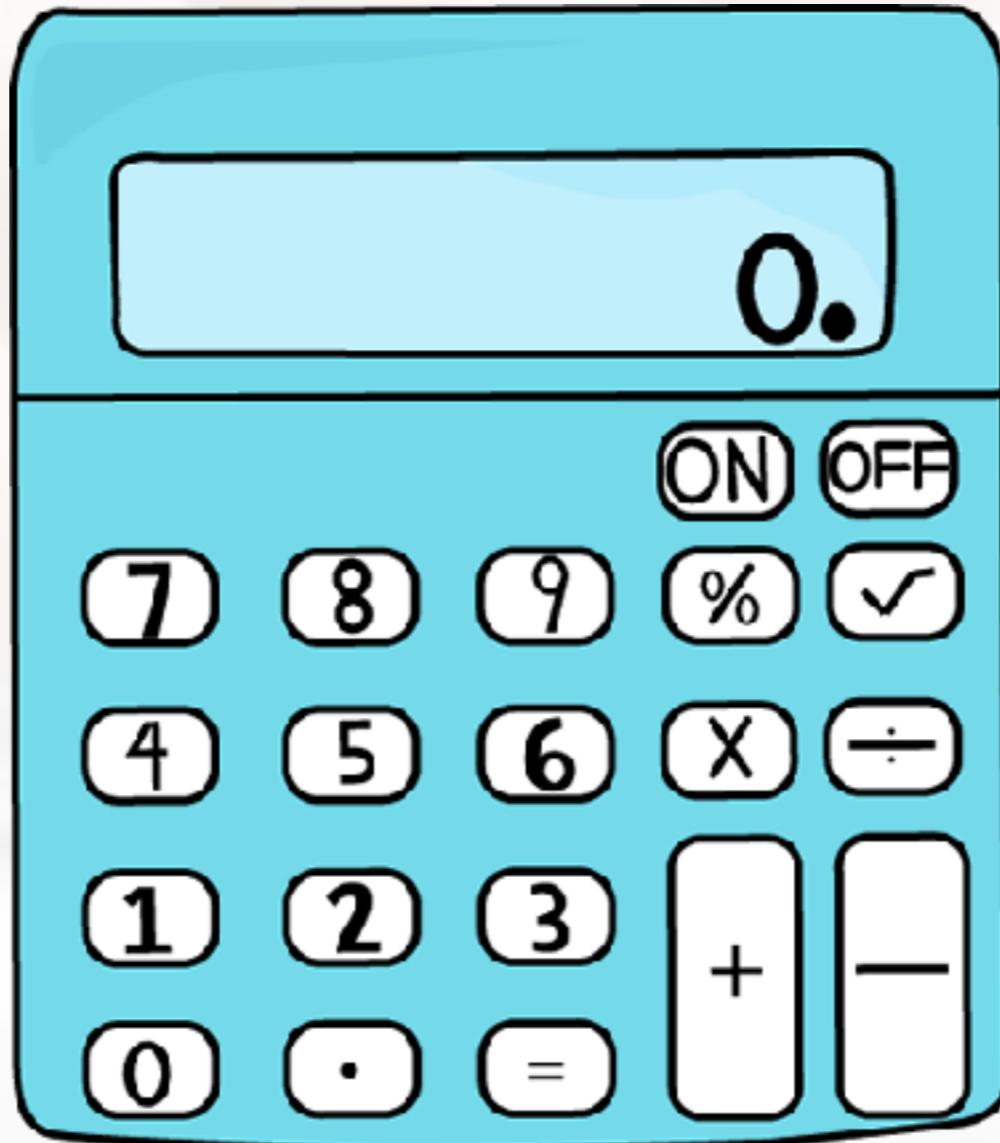
Repeatable

Self-verifying

Timely



# Workshop



# Workshop

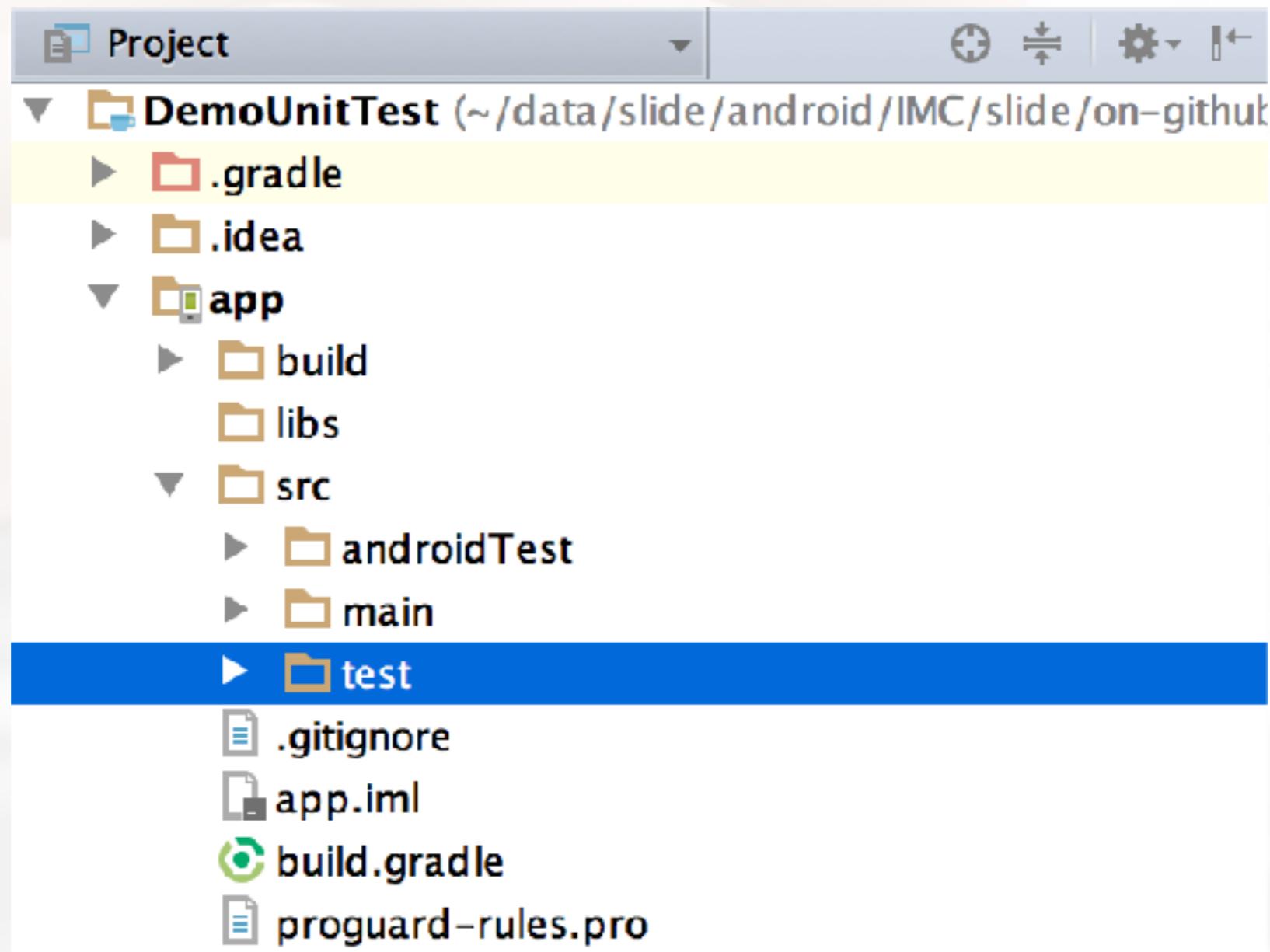
สร้าง project ใหม่ ชื่อ Calculator

เรียนรู้การสร้าง Unit test (jUnit4)

Code coverage



# Project structure

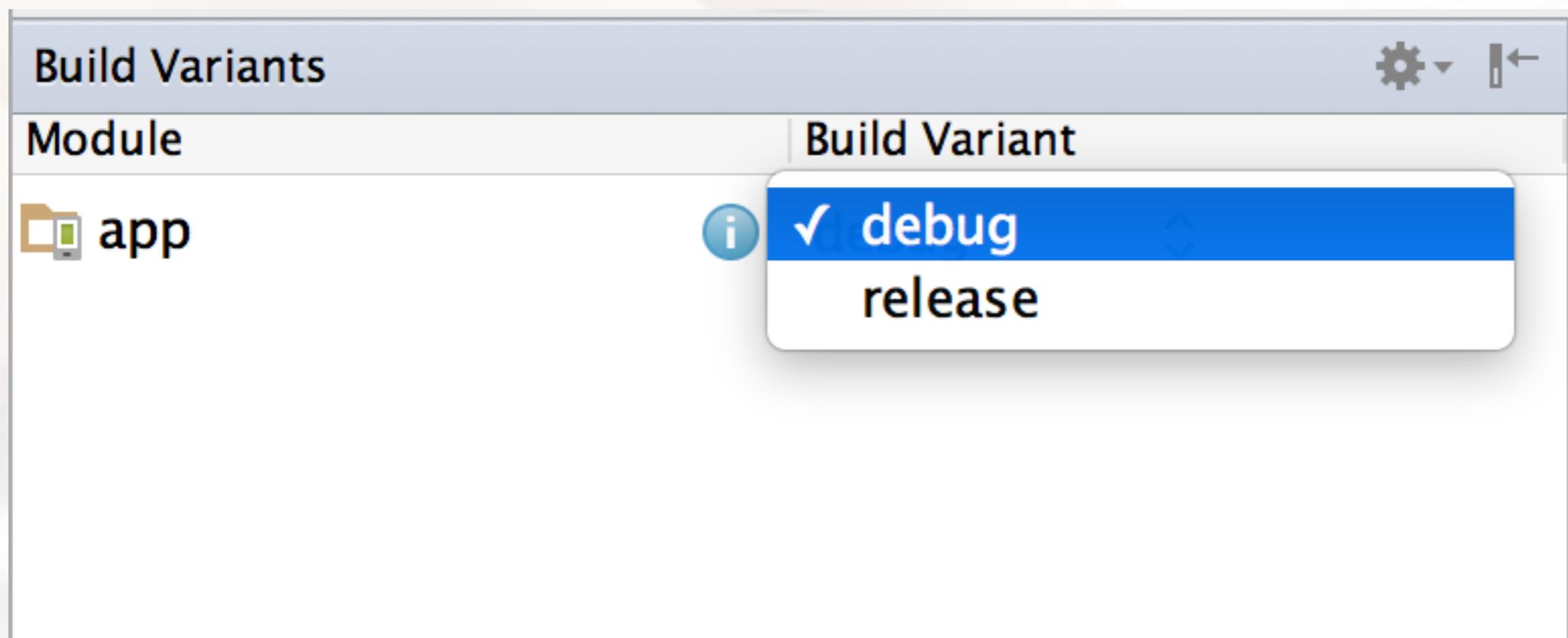


# build.gradle

```
dependencies {  
    compile fileTree(dir: 'libs', include:  
        testCompile 'junit:junit:4.12'  
        compile 'com.android.support:appcompat:  
    }  
}
```



# Build variant



# First unit test

```
public class CalculatorUnitTest {  
  
    @Test  
    public void addition_isCorrect() throws Exception {  
        //Arrange  
        Calculator calculator = new Calculator();  
        //Act  
        double actualResult0fAdd = calculator.add(1, 2);  
        //Assert  
        assertEquals(3, actualResult0fAdd, 0);  
    }  
}
```



# Naming test class and method

```
public class MyClassShould {  
    @Test public void  
        do_something_interesting() {  
    }  
}
```

Start with a verb.

Express what the class should be able to do.

Reads as a full sentence



# Naming test class and method

```
public class BankAccountShould{  
  
    @Test public void  
        have_balance_of_zero_when_created() {  
            BankAccount bankAccount = new BankAccount();  
  
            assertThat(bankAccount.balance(), is(0));  
        }  
  
    @Test public void  
        have_the_balance_increased_after_a_deposit() {  
given    BankAccount bankAccount = new BankAccount();  
  
when    bankAccount.deposit(10);  
  
then    assertThat(bankAccount.balance(), is(10));  
        }  
    }  
}
```



# Test creation order

```
public class BankAccountShould{  
    @Test public void  
        have_balance_increased_after_a_deposit() {  
  
    given BankAccount bankAccount = new BankAccount();  
  
    when bankAccount.deposit(10);  
  
    then assertThat(bankAccount.balance(), is(10));  
}  
}
```

Step 1: Name the class

Step 2: Name the method

Step 3: Define what you are testing

Step 4: Trigger the code

Step 5: Setup

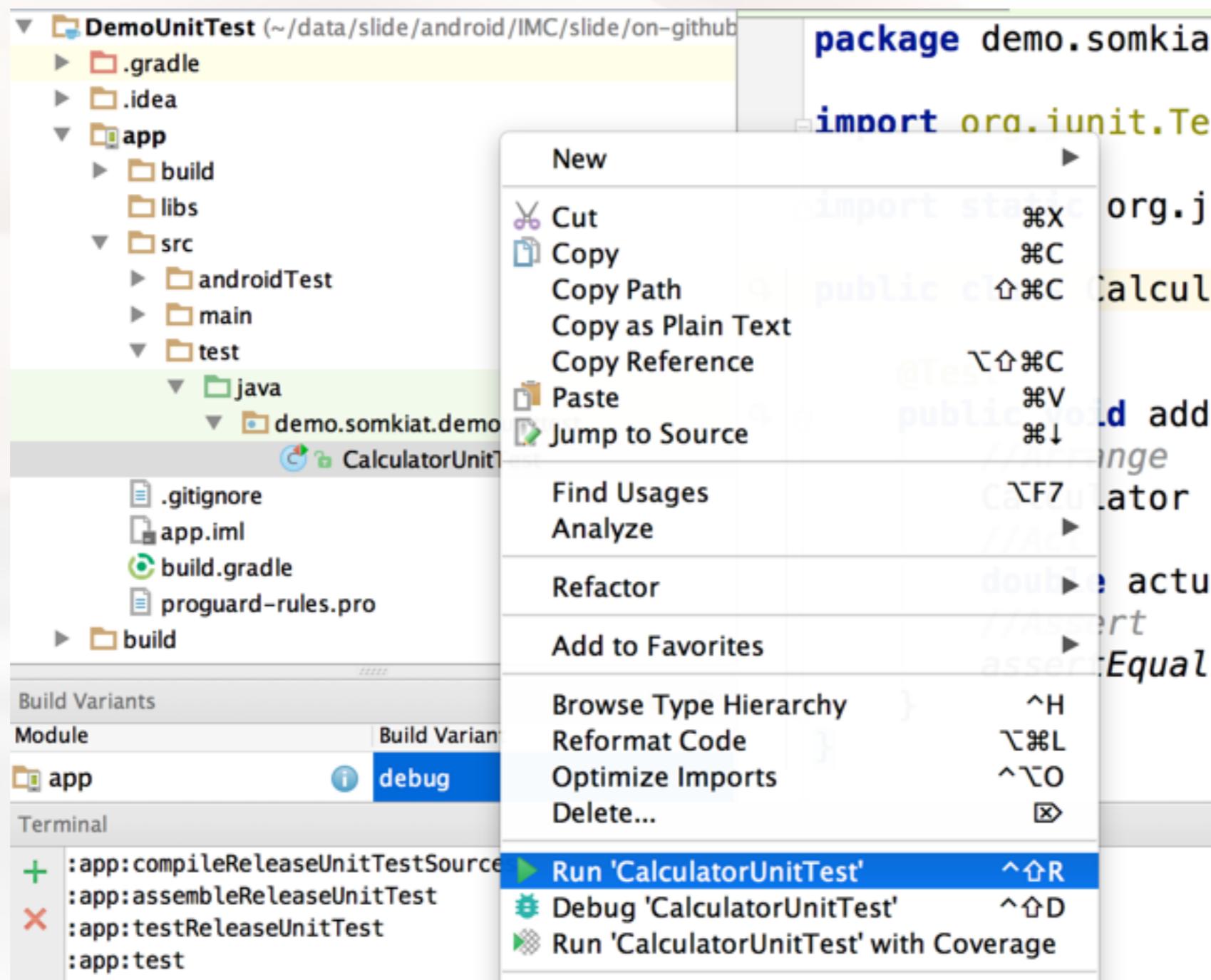
given

when

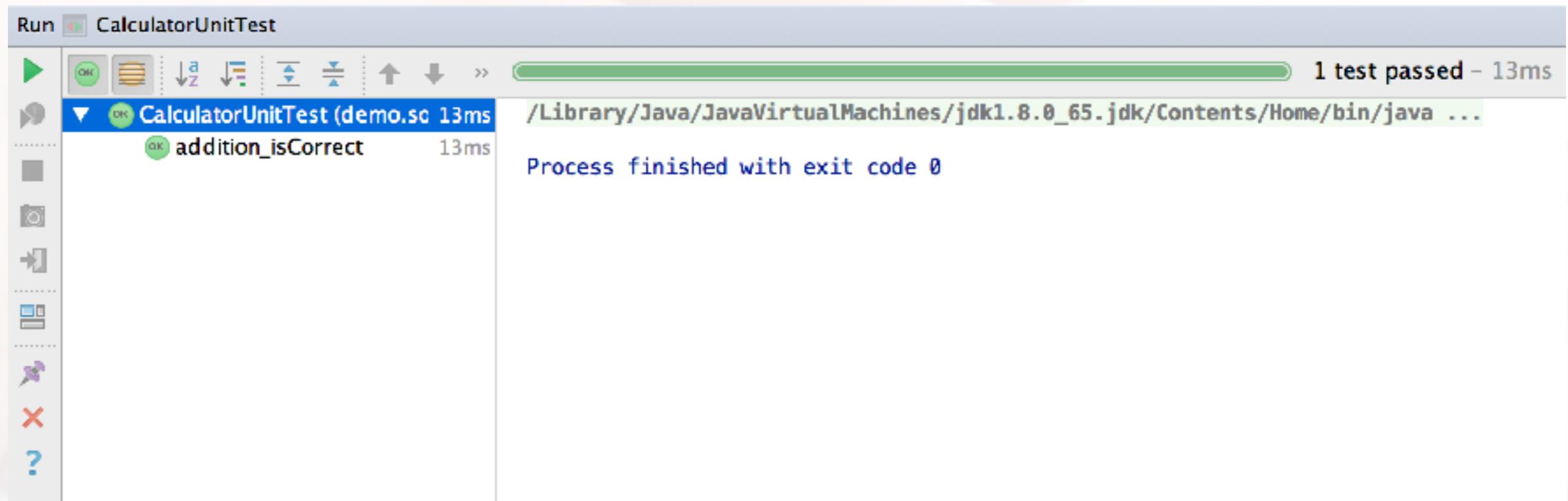
then



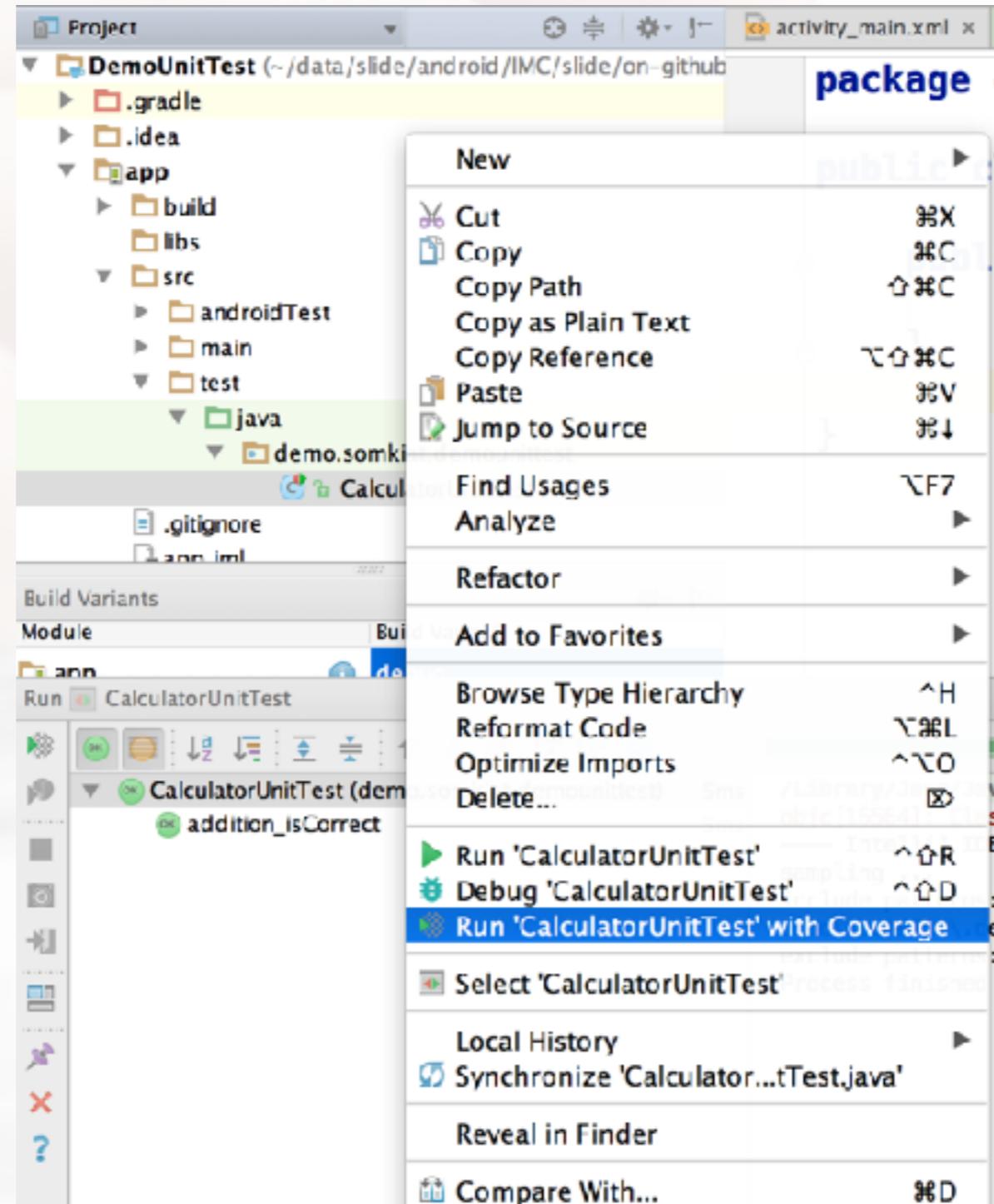
# Run unit test



# Run unit test



# Run unit test with coverage



# Run unit test with coverage

Coverage CalculatorUnitTest			
↑ 5% classes, 3% lines covered in package 'demo.somkiat.demounittest'			
Element	Class, %	Method, %	Line, %
BuildConfig	0% (0/1)	0% (0/1)	0% (0/2)
Calculator	100% (1/1)	100% (1/1)	100% (2/2)
MainActivity	0% (0/1)	0% (0/1)	0% (0/4)
R	0% (0/14)	0% (0/1)	0% (0/43)



# Run unit test

`./gradlew test --continue`



# Show report

## Test Summary

1 tests      0 failures      0 ignored      0.002s duration

100%  
successful

Packages

Classes

Package	Tests	Failures	Ignored	Duration	Success rate
<a href="#">demo.somkiat.demounittest</a>	1	0	0	0.002s	100%

Generated by [Gradle 2.10](#) at May 7, 2016, 2:50:21 PM



# Run unit test with coverage

```
./gradlew createDebugCoverageReport
```



# Enable code coverage

## build.gradle

```
buildTypes {  
    debug {  
        testCoverageEnabled = true  
    }  
    release {  
        minifyEnabled false  
        proguardFiles getDefaultProguardFile("proguard-android.txt"),  
        "proguard-rules.pro"  
    }  
}
```



# Show report

 [debug](#) >  [demo.somkiat.demounittest](#)

## demo.somkiat.demounittest

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Mi
 <a href="#">MainActivity</a>		0%		n/a	2	2	
 <a href="#">Calculator</a>		0%		n/a	2	2	
Total	17 of 17	0%	0 of 0	n/a	4	4	



# Run unit test with coverage

```
./gradlew createDebugCoverageReport
```

Working with instrumented test (AndroidTest) !!



# Run unit test with coverage

Let's start to hack jacoco !!



# Enable coverage for unit test

## build.gradle

```
apply plugin: 'jacoco'

task jacocoTestReport(type: JacocoReport, dependsOn:

    reports {
        xml.enabled = true
        html.enabled = true
    }

    jacocoClasspath = configurations['androidJacocoA

    def fileFilter = ['**/R.class', '**/R*.class',
    def debugTree = fileTree(dir: "${buildDir}/inter
    def mainSrc = "${project.projectDir}/src/main/ja

    sourceDirectories = files([mainSrc])
    classDirectories = files([debugTree])
    executionData = files("${buildDir}/jacoco/testDe
```



# Run unit test with coverage

```
./gradlew jacocoTestReport
```



# Show report

app > demo.somkiat.demounittest

## demo.somkiat.demounittest

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed
 <a href="#">MainActivity</a>		0%		n/a	2	2	4	4	
 <a href="#">Calculator</a>		100%		n/a	0	2	0	2	
Total	10 of 17	41%	0 of 0	n/a	2	4	4	6	



# Jacoco android gradle

## jacoco-android-gradle-plugin

[build](#) [passing](#) [Codecov](#) [98%](#) [Download](#) [0.1.1](#)

A Gradle plugin that adds fully configured `JacocoReport` tasks for unit tests of each Android application and library project variant.

### Why

In order to generate JaCoCo unit test coverage reports for Android projects you need to create `JacocoReport` tasks and configure them by providing paths to source code, execution data and compiled classes. It can be troublesome since Android projects can have different flavors and build types thus requiring additional paths to be set. This plugin provides those tasks already configured for you.

<https://github.com/arturdm/jacoco-android-gradle-plugin>



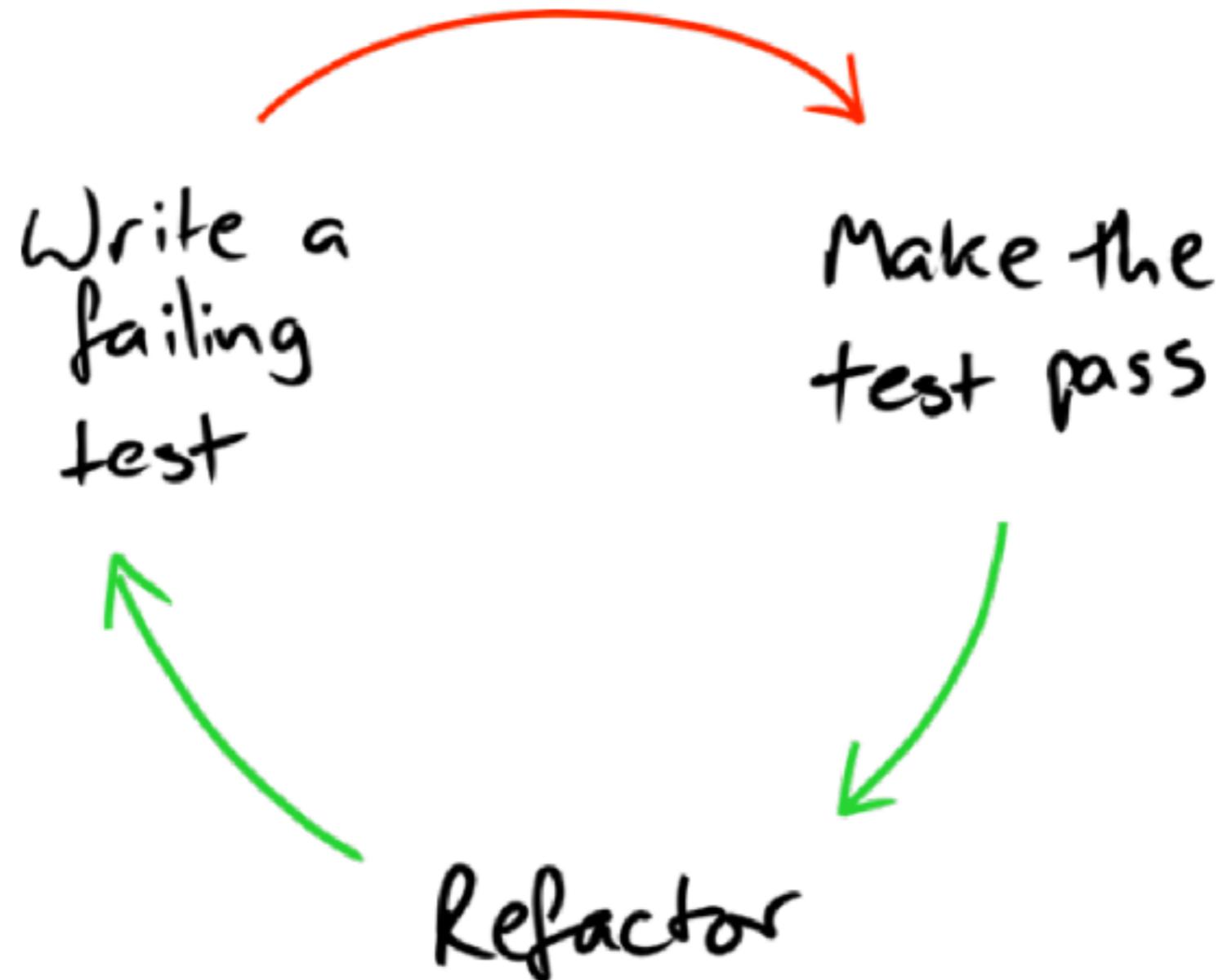
# ແບບຝຶກຫັດ

ບວກ ລບ ຄູລ ຮາຮ

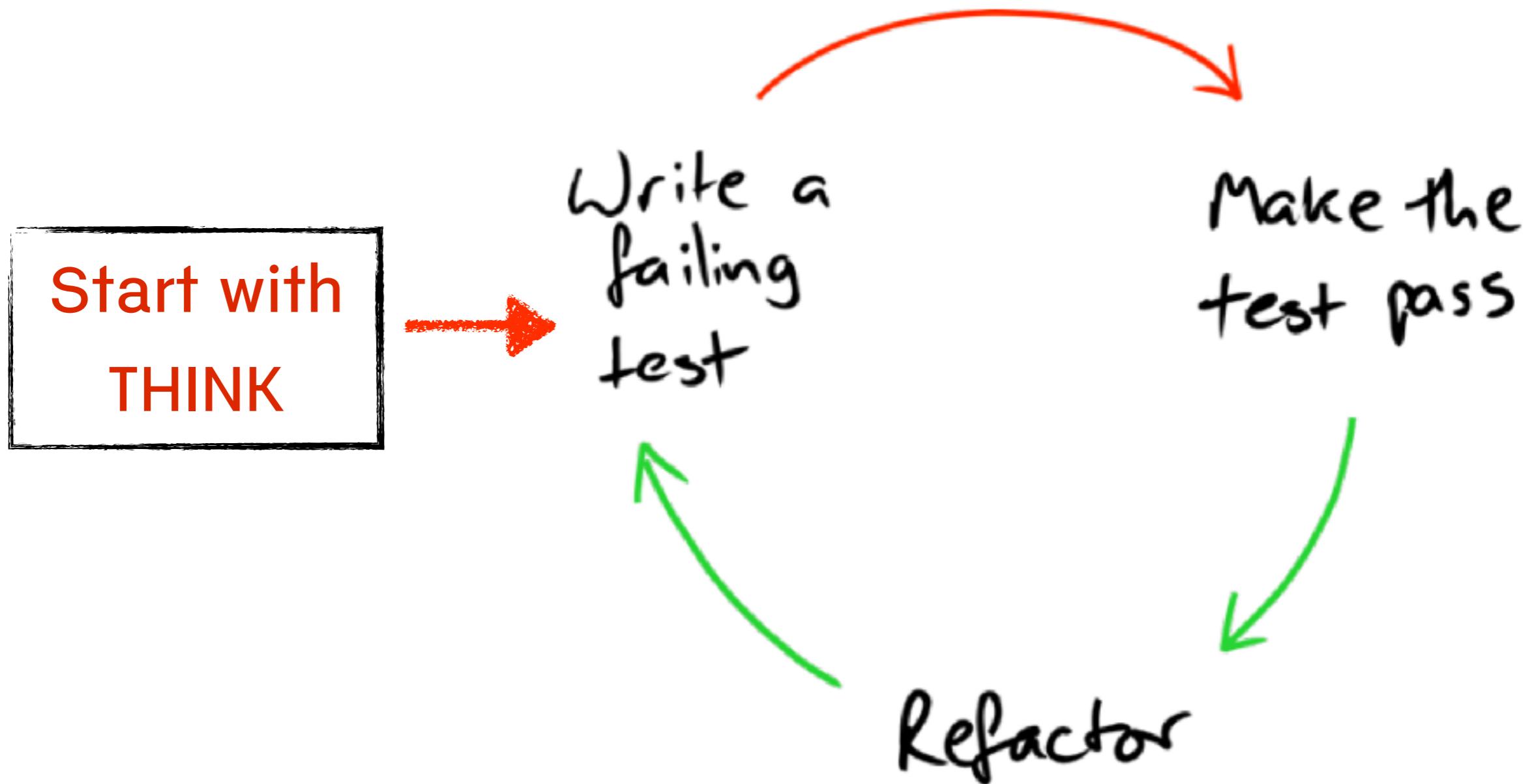
Code coverage = 100% !!



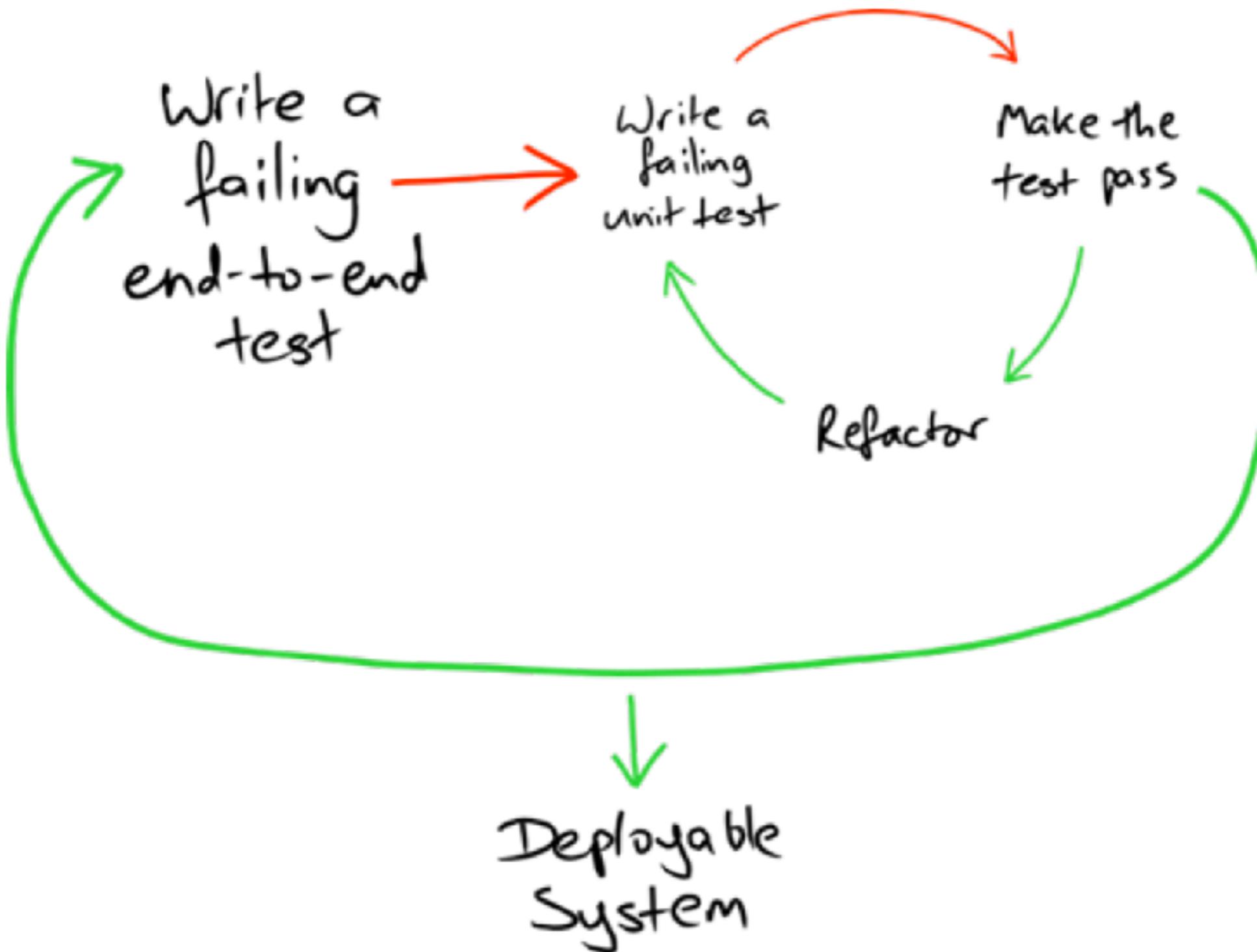
# ทำดีๆ



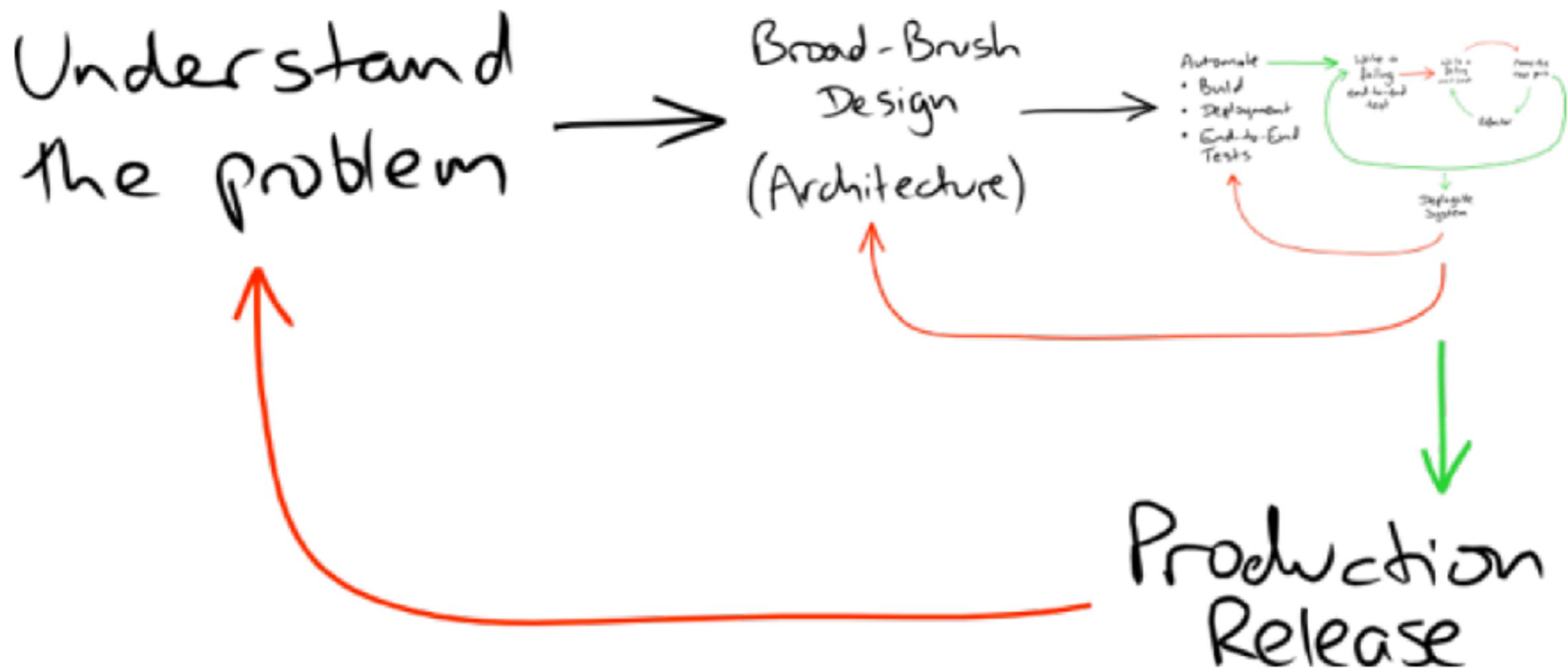
# คิดก่อนทำดีดี



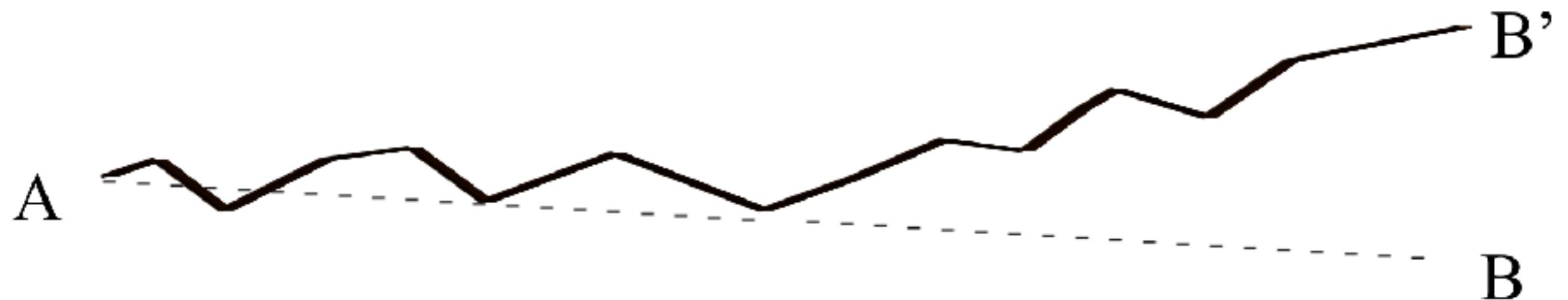
# ACCEPTANCE TEST DRIVEN DEVELOPMENT



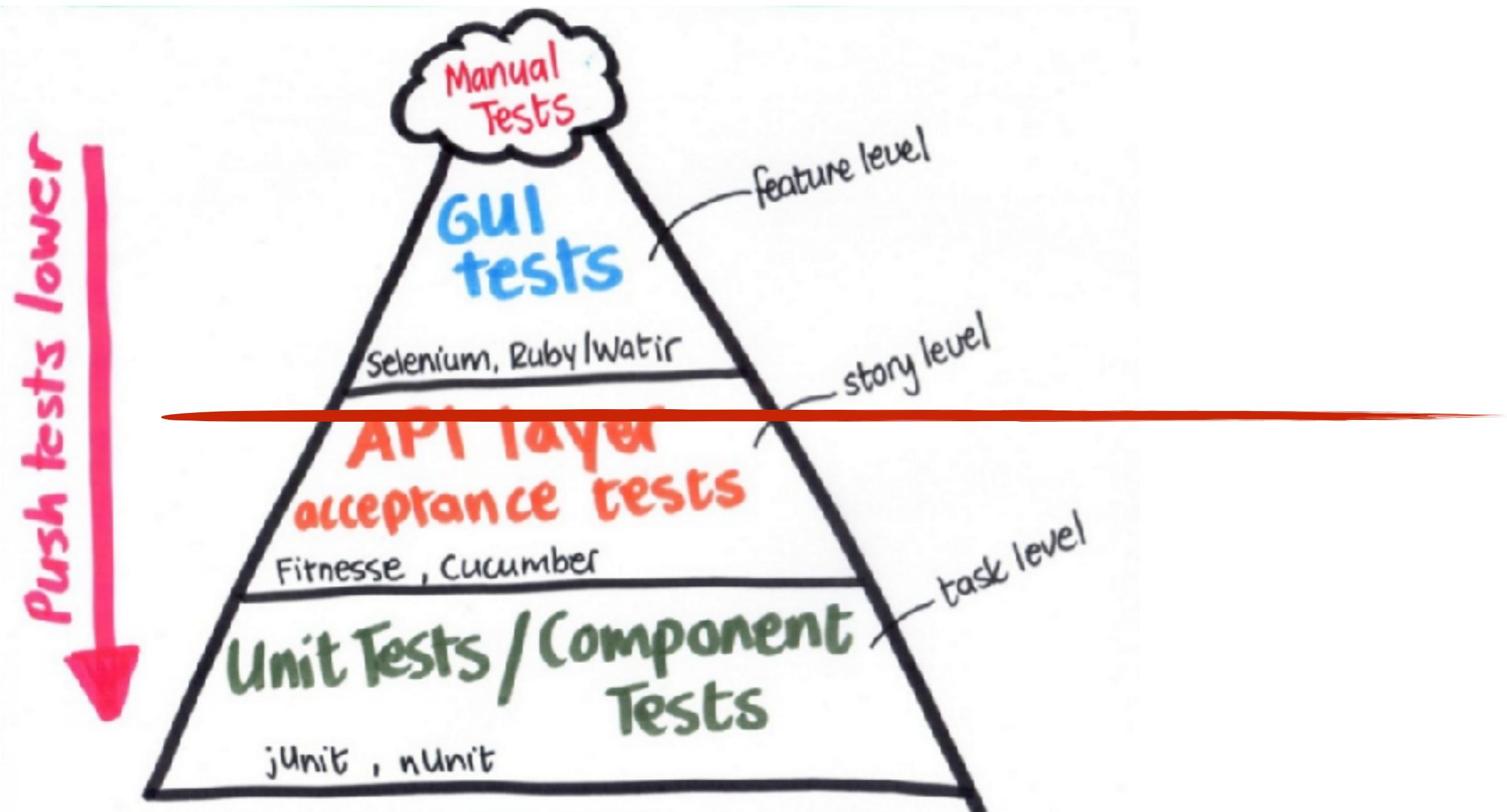
# LARGER FEEDBACK LOOP



# Small step



# Testing for android



# UI testing

ทำให้แน่ใจว่า App ใช้งานได้  
บน device ต่าง ๆ



# UI testing

**Looks good but can't use !!**



# UI testing

ต้องการทดสอบการทำงานของ Activity

ใช้เวลาการทดสอบนาน

จำนวนการทดสอบมีเท่าที่จำเป็น



# UI testing frameworks



UI Automator

Calabash.sh



Robotium



# UI testing



<https://google.github.io/android-testing-support-library/docs/espresso/index.html>



บริษัท สยามชานาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Espresso ?

Android Testing Support Library

Simulate user interactions

Automatic synchronization of test action  
with app UI



Who needs sleep?



# Espresso components

ViewMatcher

ViewAction

ViewAssertion



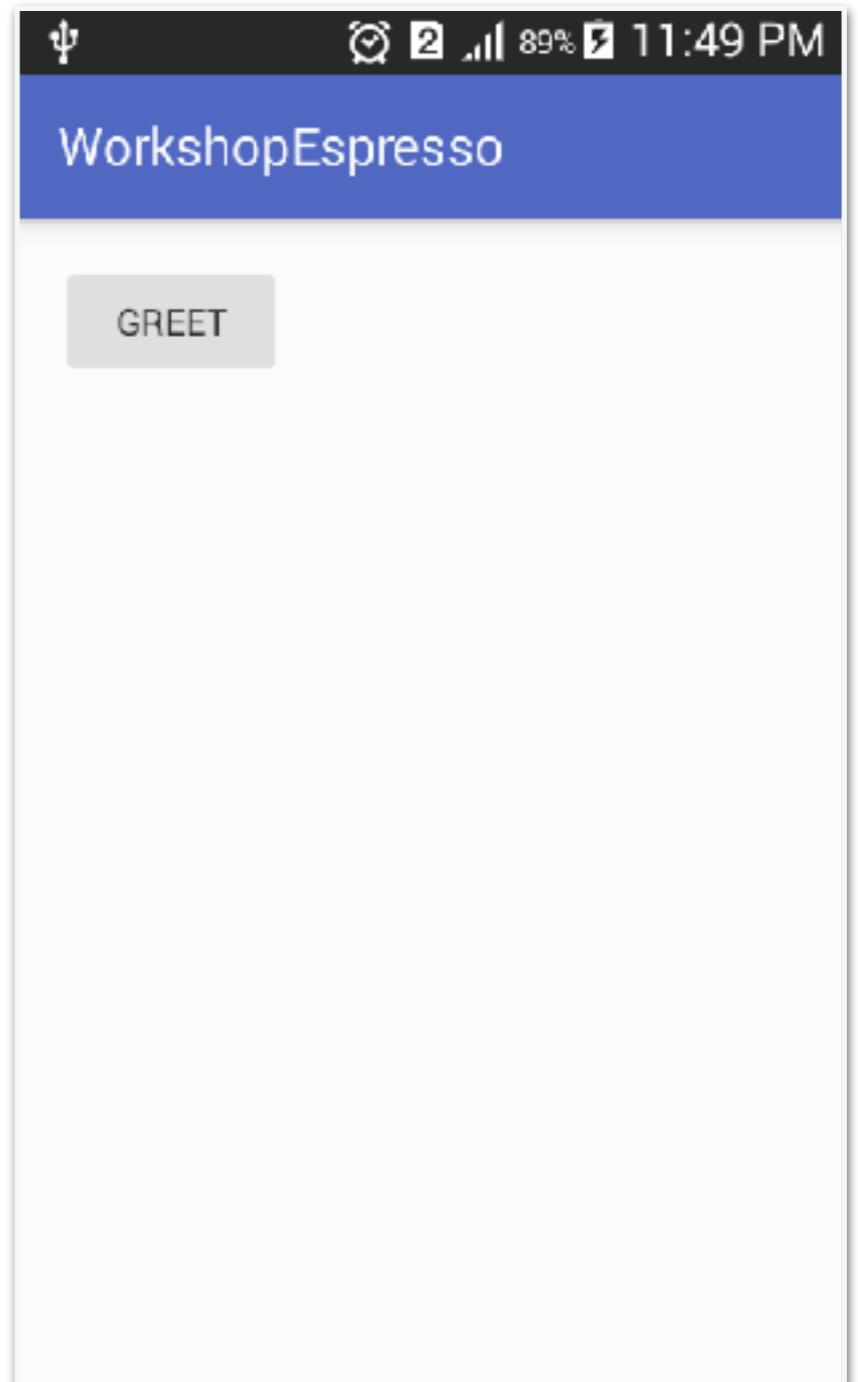
# Espresso

```
onView(ViewMatcher)  
    .perform(ViewAction)  
    .check(ViewAssertion)
```



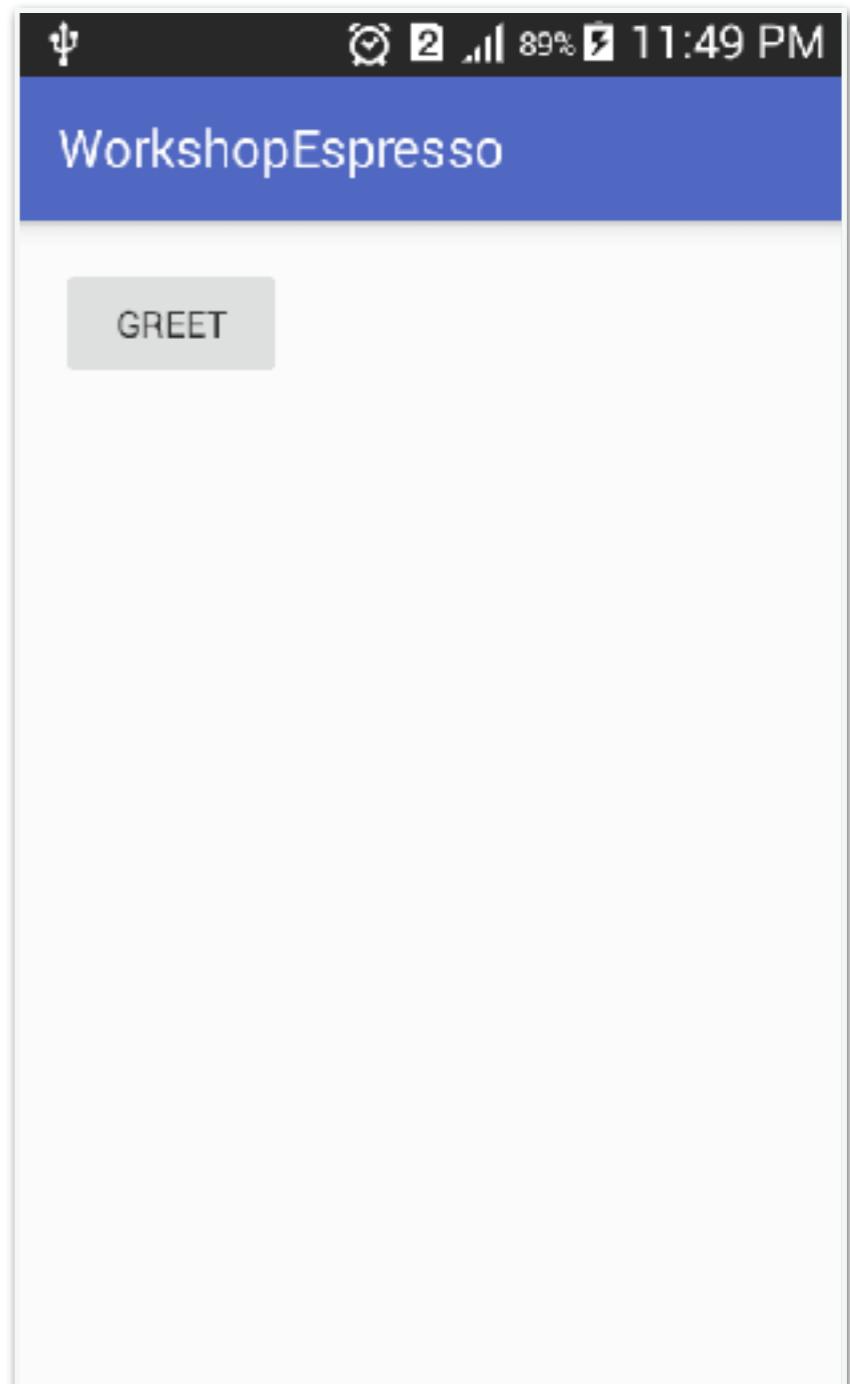
# Hello Espresso

```
onView(ViewMatcher)  
    .perform(ViewAction)  
    .check(ViewAssertion)
```



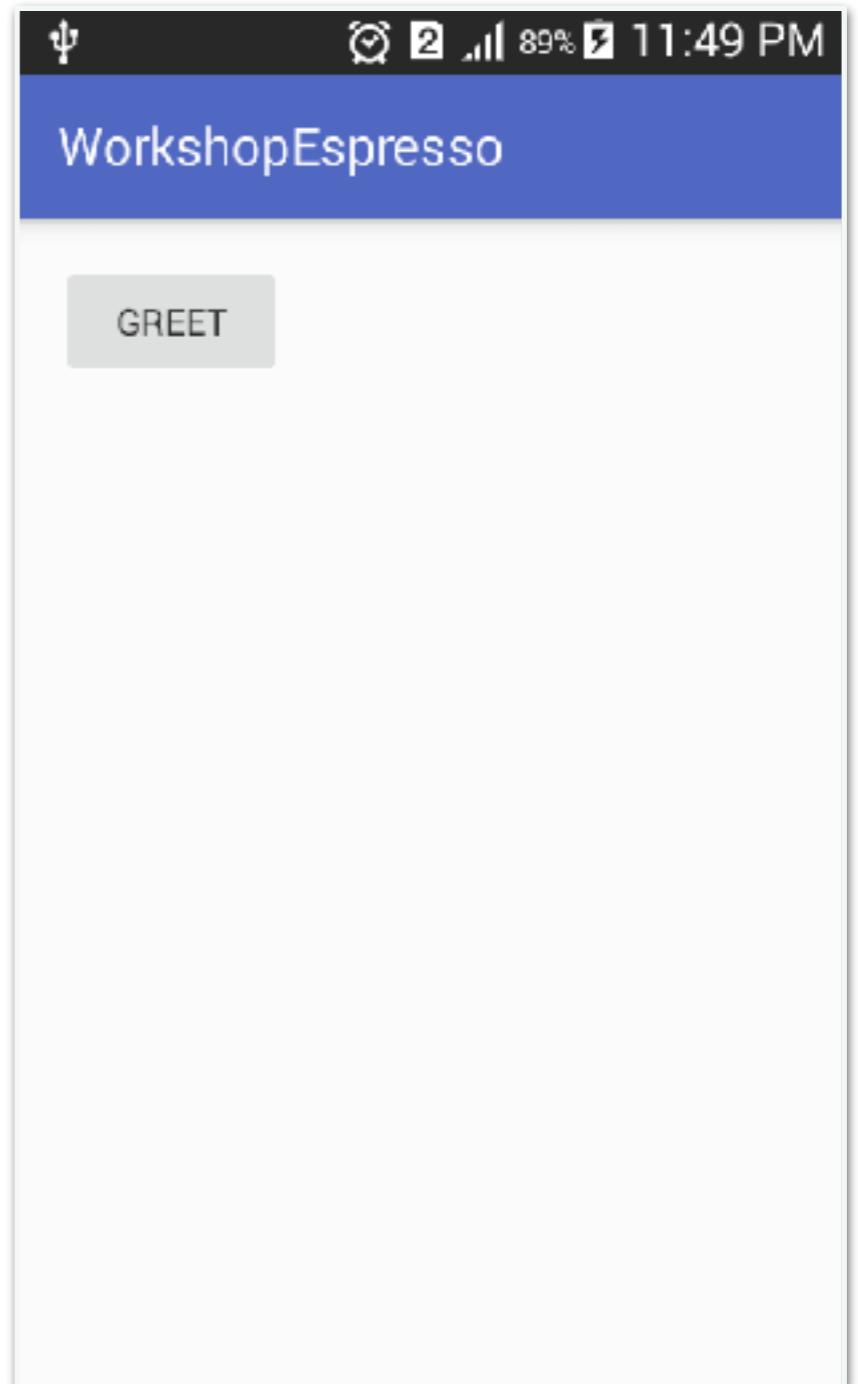
# ViewMatcher

```
onView(withId(R.id.greet_button))  
    .perform(ViewAction)  
    .check(ViewAssertion)
```



# ViewAction

```
onView(withId(R.id.greet_button))  
    .perform(click())  
    .check(ViewAssertion)
```



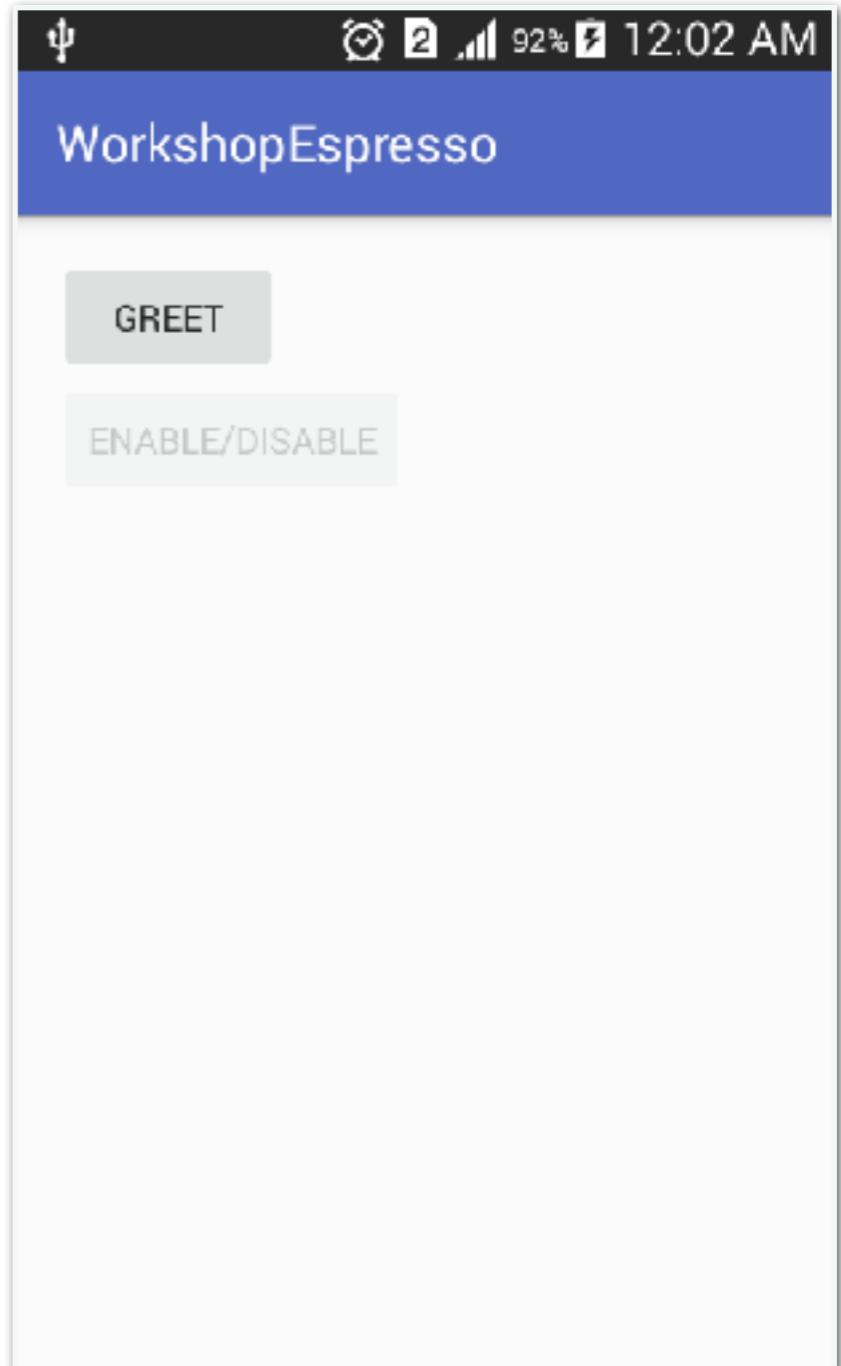
# ViewAssertion

```
onView(withId(R.id.greet_button))  
    .perform(click())  
    .check(matches(withText("สวัสดี")))
```



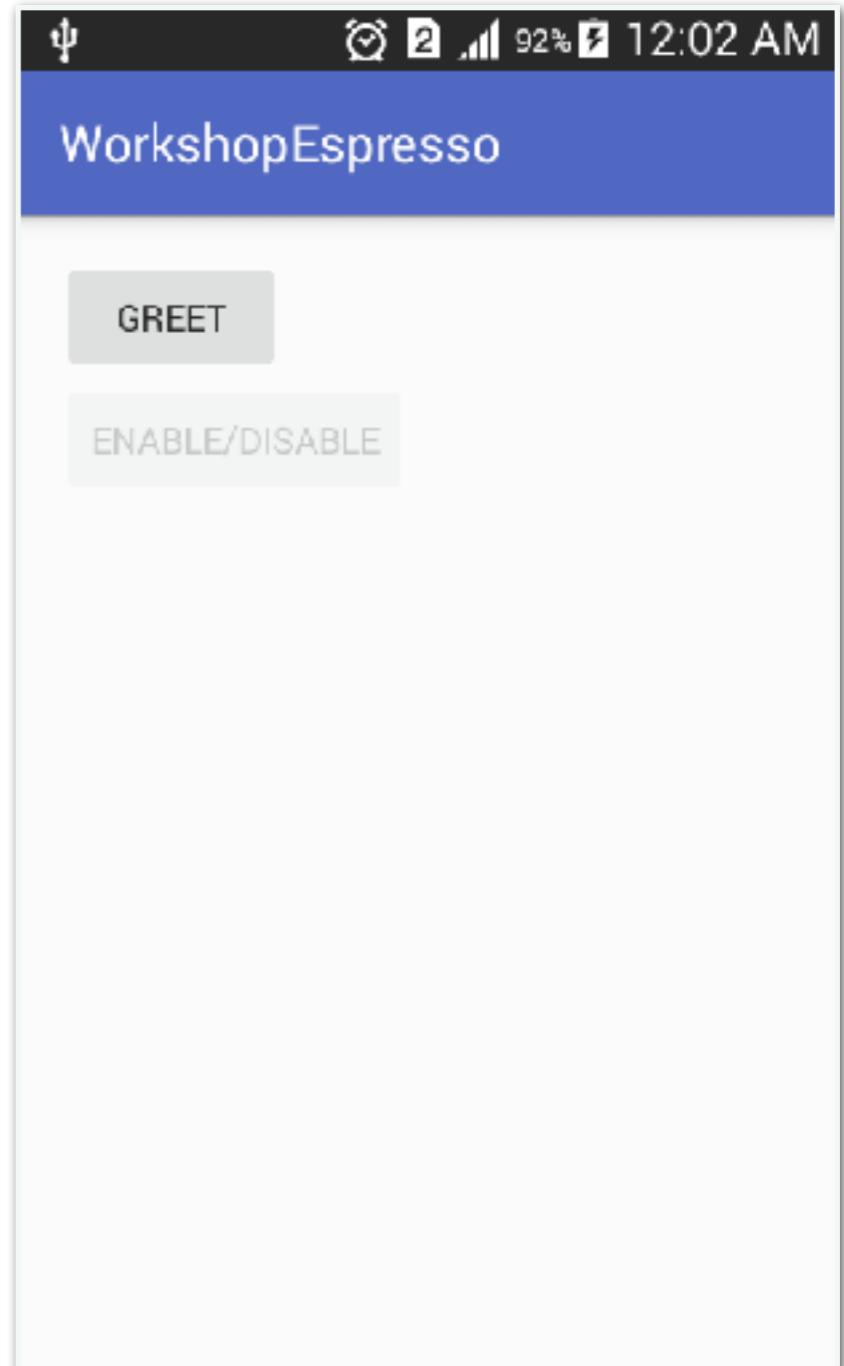
# ViewAssertion

```
onView(withId(R.id.state_button))  
    .perform(click())  
    .check(matches(not(isEnabled())))
```



# Hamcrest

```
onView(withId(R.id.state_button))  
    .perform(click())  
    .check(matches(not(isEnabled())))
```



# Espresso cheatsheet

## View Matchers

### USER PROPERTIES

```
withId(...)  
withText(...)  
withTagKey(...)  
withTagValue(...)  
hasContentDescription(...)  
withContentDescription(...)  
withHint(...)  
withSpinnerText(...)  
hasLinks()  
hasEllipsizeText()  
hasMultilineText()
```

### HIERARCHY

```
withParent(Matcher)  
withChild(Matcher)  
hasDescendant(Matcher)  
isDescendantOfA(Matcher)  
hasSibling(Matcher)  
isRoot()
```

### INPUT

```
supportsInputMethods(...)  
hasIMEAction(...)
```

### UI PROPERTIES

```
isDisplayed()  
isCompletelyDisplayed()  
isEnabled()  
hasFocus()  
isClickable()  
isChecked()  
isNotChecked()  
withEffectiveVisibility(...)  
isSelected()
```

### OBJECT MATCHER

```
allOf(Matchers)  
anyOf(Matchers)  
is(...)  
not(...)  
endsWith(String)  
startsWith(String)  
instanceOf(Class)
```

### CLASS

```
isAssignableFrom(...)  
withClassName(...)
```

### ROOT MATCHERS

```
isFocusable()  
isTouchable()  
isDialog()  
withDecorView()  
isPlatformPopup()
```

### SEE ALSO

Preference matchers  
Cursor matchers  
Layout matchers

## Data Options

```
inAdapterView(Matcher)  
atPosition(Integer)  
onChildView(Matcher)
```

## View Actions

### CLICK/PRESS

```
click()  
doubleClick()  
longClick()  
pressHack()  
pressIMEActionButton()  
pressKey([int/EspressoKey])  
pressMenuKey()  
closeSoftKeyboard()  
openLink()
```

### GESTURES

```
scrollTo()  
swipeLeft()  
swipeRight()  
swipeUp()  
swipeDown()
```

### TEXT

```
clearText()  
typeText(String)  
typeTextIntoFocusedView(String)  
replaceText(String)
```

## View Assertions

### MATCHES

```
matches(Matcher)  
doesNotExist()  
selectedDescendantsMatch(...)
```

### LAYOUT ASSERTIONS

```
noEllipsizeText(Matcher)  
noMultilineButtons()  
noOverlaps([Matcher])
```

### POSITION ASSERTIONS

```
isLeftOf(Matcher)  
isRightOf(Matcher)  
isLeftAlignedWith(Matcher)  
isRightAlignedWith(Matcher)  
isAbove(Matcher)  
isBelow(Matcher)  
isBottomAlignedWith(Matcher)  
isTopAlignedWith(Matcher)
```

<https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/index.html>



# Hamcrest cheatsheet

## General purpose

```
is(T)
equalTo(T)
not(T) : Matcher<T>
anything()
anything(String) : Matcher<Object>
any(Class<T>)
instanceOf(Class<?>)
isA(Class<T>) : Matcher<T>
nullValue()
nullValue(Class<T>)
notNullValue()
notNullValue(Class<T>)
sameInstance(T)
theInstance(T) : Matcher<T>
isIn(Collection<T>)
isIn(T[])
isOneOf(T...)
hasToString(String)
hasToString(Matcher<? super String>) : Matcher<T>
```

## Combining multiple matchers

```
is(Matcher<T>)
not(Matcher<T>) : Matcher<T>
allOf(Matcher<? super T>...)
allOf(Iterable<Matcher<? super T>>)
anyOf(Matcher<? super T>...)
anyOf(Iterable<Matcher<? super T>>) : Matcher<T>
both(Matcher<? super LHS>)
either(Matcher<? super LHS>) : Matcher<LHS>
describedAs(String, Matcher<T>, Object...) : Matcher<T>
```

## Strings

```
containsString(String)
startsWith(String)
endsWith(String) : Matcher<String>
equalToIgnoringCase(String)
equalToIgnoringWhiteSpace(String) : Matcher<String>
isEmptyString()
isEmptyOrNullString() : Matcher<String>
stringContainsInOrder(Iterable<String>) : Matcher<String>
```

## Iterables

```
everyItem(Matcher<U>) : Matcher<Iterable<U>>
hasItem(T)
hasItem(Matcher<? super T>) : Matcher<Iterable<? super T>>
hasItems(T...)
hasItems(Matcher<? super T>...) : Matcher<Iterable<T>>
emptyIterable() : Matcher<Iterable<? extends E>>
emptyIterableOf(Class<E>) : Matcher<Iterable<E>>
contains(E...)
contains(Matcher<? super E>...)
contains(Matcher<? super E>)
contains(List<Matcher<? super E>>)
containsInAnyOrder(T...)
containsInAnyOrder(Collection<Matcher<? super T>>)
containsInAnyOrder(Matcher<? super T>...)
containsInAnyOrder(Matcher<? super E>)
iterableWithSize(Matcher<? super Integer>)
iterableWithSize(int) : Matcher<Iterable<E>>
```

## Collections

```
hasSize(int)
hasSize(Matcher<? super Integer>)
empty() : Matcher<Collection<? extends E>>
emptyCollectionOf(Class<E>) : Matcher<Collection<E>>
```

## Arrays

```
array(Matcher<? super T>...)
hasItemInArray(T)
hasItemInArray(Matcher<? super T>) : Matcher<T[]>
arrayContaining(E...)
arrayContaining(List<Matcher<? super E>>)
arrayContaining(Matcher<? super E>...) : Matcher<E[]>
arrayContainingInAnyOrder(E...)
arrayContainingInAnyOrder(Matcher<? super E>...)
arrayContainingInAnyOrder(Collection<Matcher<? super E>>)
arrayWithSize(int)
arrayWithSize(Matcher<? super Integer>)
emptyArray() : Matcher<E[]>
```

## Maps

```
hasEntry(K, V)
hasEntry(Matcher<? super K>, Matcher<? super V>)
hasKey(K)
hasKey(Matcher<? super K>) : Matcher<Map<? extends K, ? extends V>>
hasValue(V)
hasValue(Matcher<? super V>) : Matcher<Map<?, ? extends V>>
```

## Beans

```
hasProperty(String)
hasProperty(String, Matcher<?>)
samePropertyValuesAs(T) : Matcher<T>
```

## Comparables

```
comparesEqualTo(T extends Comparable<T>)
greaterThan(T extends Comparable<T>)
greaterThanOrEqualTo(T extends Comparable<T>)
lessThan(T extends Comparable<T>)
lessThanOrEqualTo(T extends Comparable<T>) : Matcher<T>
```

## Numbers

```
closeTo(double, double) : Matcher<Double>
closeTo(BigDecimal, BigDecimal) : Matcher<BigDecimal>
```

## Classes

```
typeCompatibleWith(Class<T>) : Matcher<java.lang.Class<?>>
```

## EventObjects

```
eventFrom(Object)
eventFrom(Class<? extends EventObject>, Object)
: Matcher<EventObject>
```

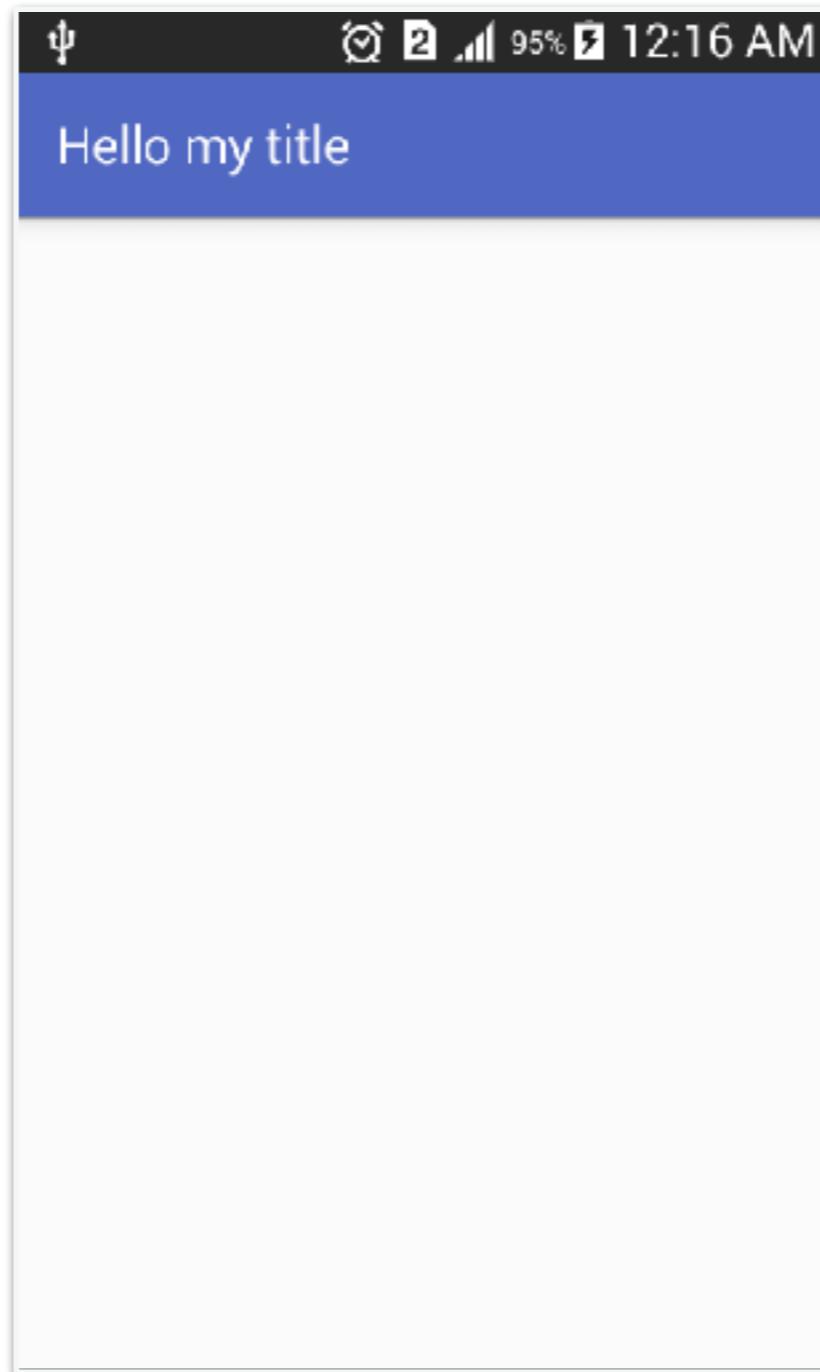
## DOM

```
hasXPath(String)
hasXPath(String, NamespaceContext)
hasXPath(String, Matcher<String>)
hasXPath(String, NamespaceContext, Matcher<String>)
: Matcher<org.w3c.dom.Node>
```

Created by Marc Philipp, <http://www.marcphilipp.de>  
This work is licensed under a Creative Commons  
Attribution-ShareAlike 3.0 Unported License,  
<http://creativecommons.org/licenses/by-sa/3.0/>



# Matchers with Toolbar title



# Espresso with onData

```
onData(ObjectMatcher)  
    .DataOptions  
    .perform(ViewAction)  
    .check(ViewAssertion)
```

## *Data Options*

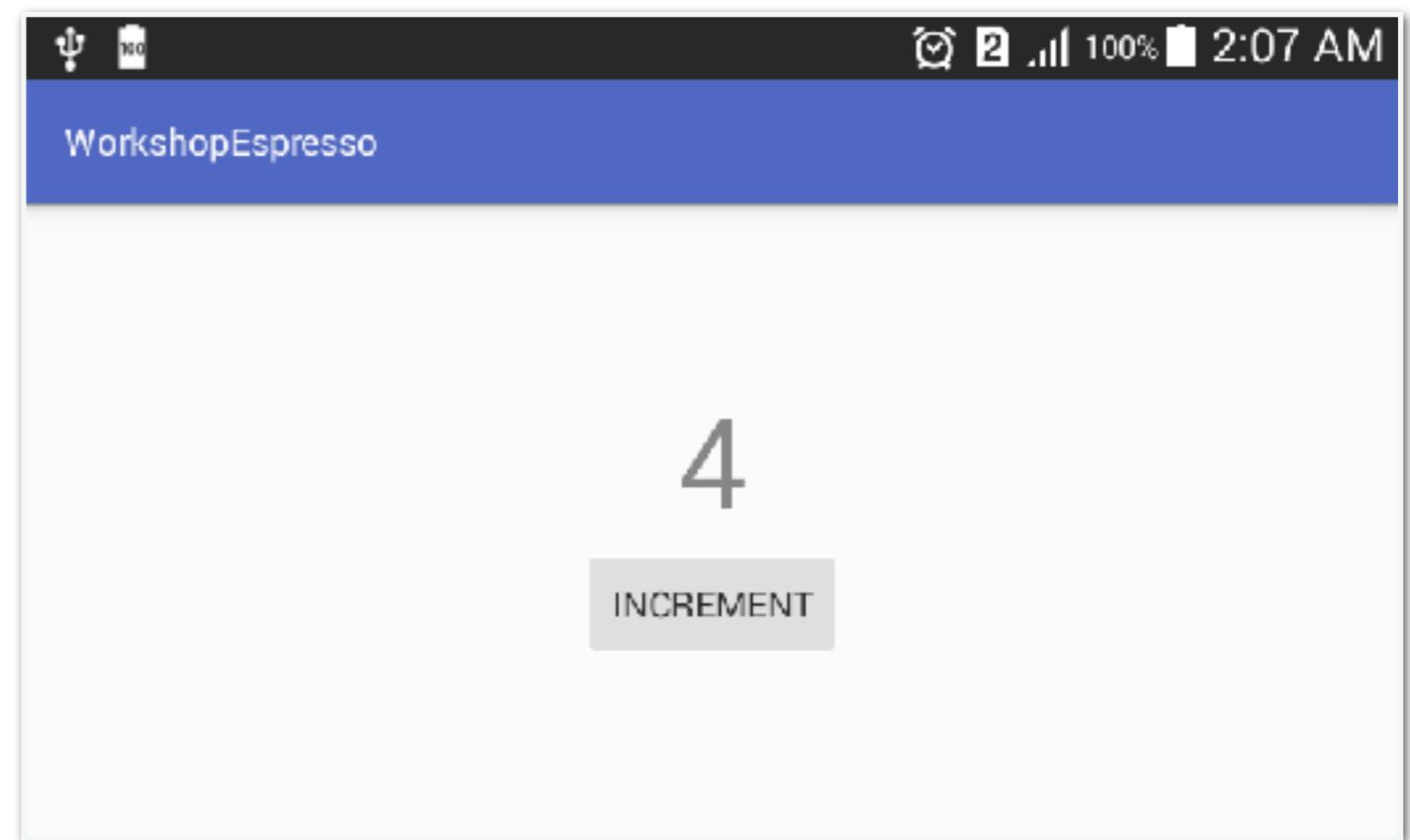
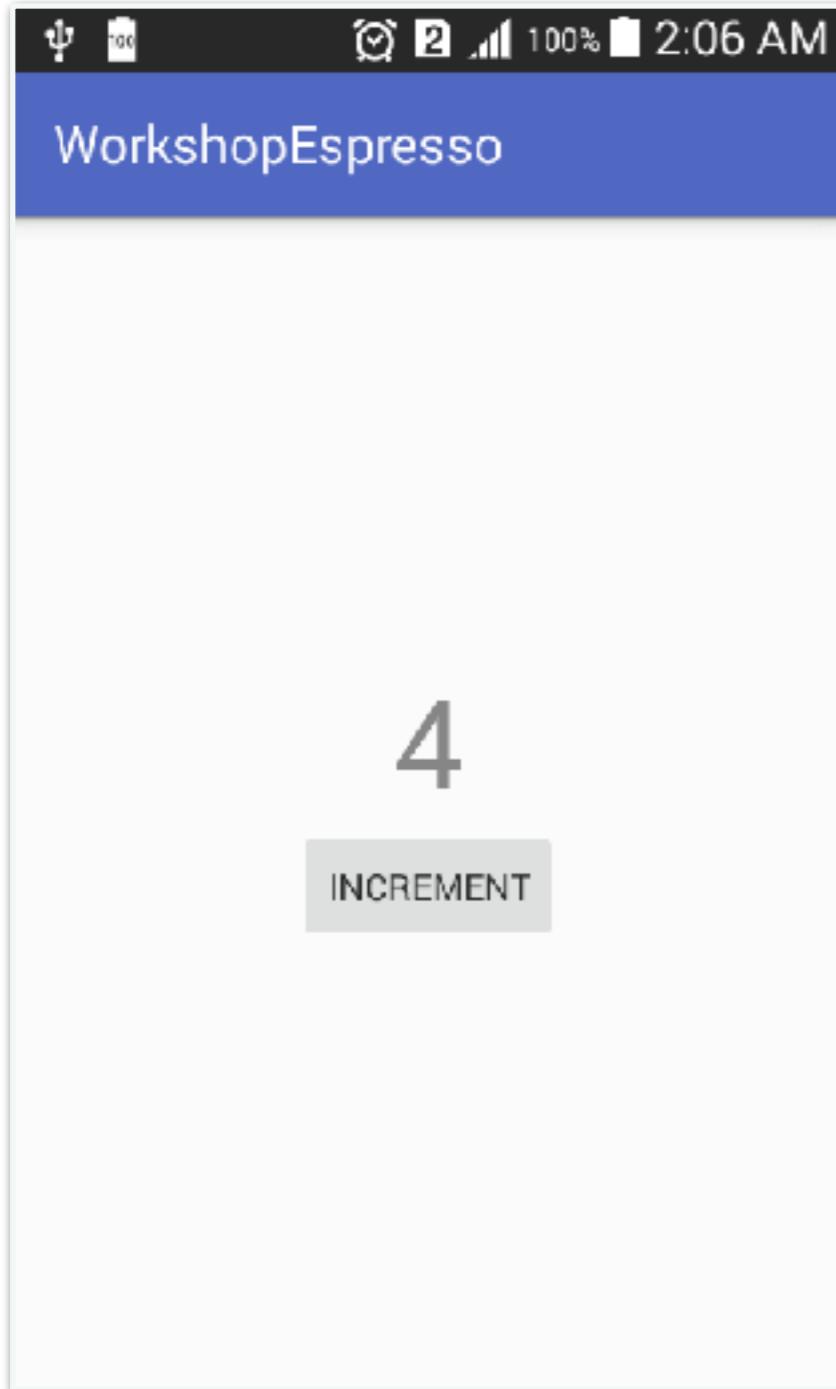
```
inAdapterView(Matcher)  
atPosition(Integer)  
onChildView(Matcher)
```



# Listview



# Rotate screen



# Example



# Example

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class CalculatorAddTest {

    public static final String ONE = "1";
    public static final String TWO = "2";
    public static final String RESULT = "3.0";

    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
        new ActivityTestRule<MainActivity>(MainActivity.class);

    @Test
    public void calculatorAdd() {
        onView(withId(R.id.operand_one_edit_text)).perform(typeText(ONE));
        onView(withId(R.id.operand_two_edit_text)).perform(typeText(TWO));
        onView(withId(R.id.operation_add_button)).perform(click());
        onView(withId(R.id.operation_result_text_view)).check(matches(withText(RESULT)));
    }
}
```



# Workshop

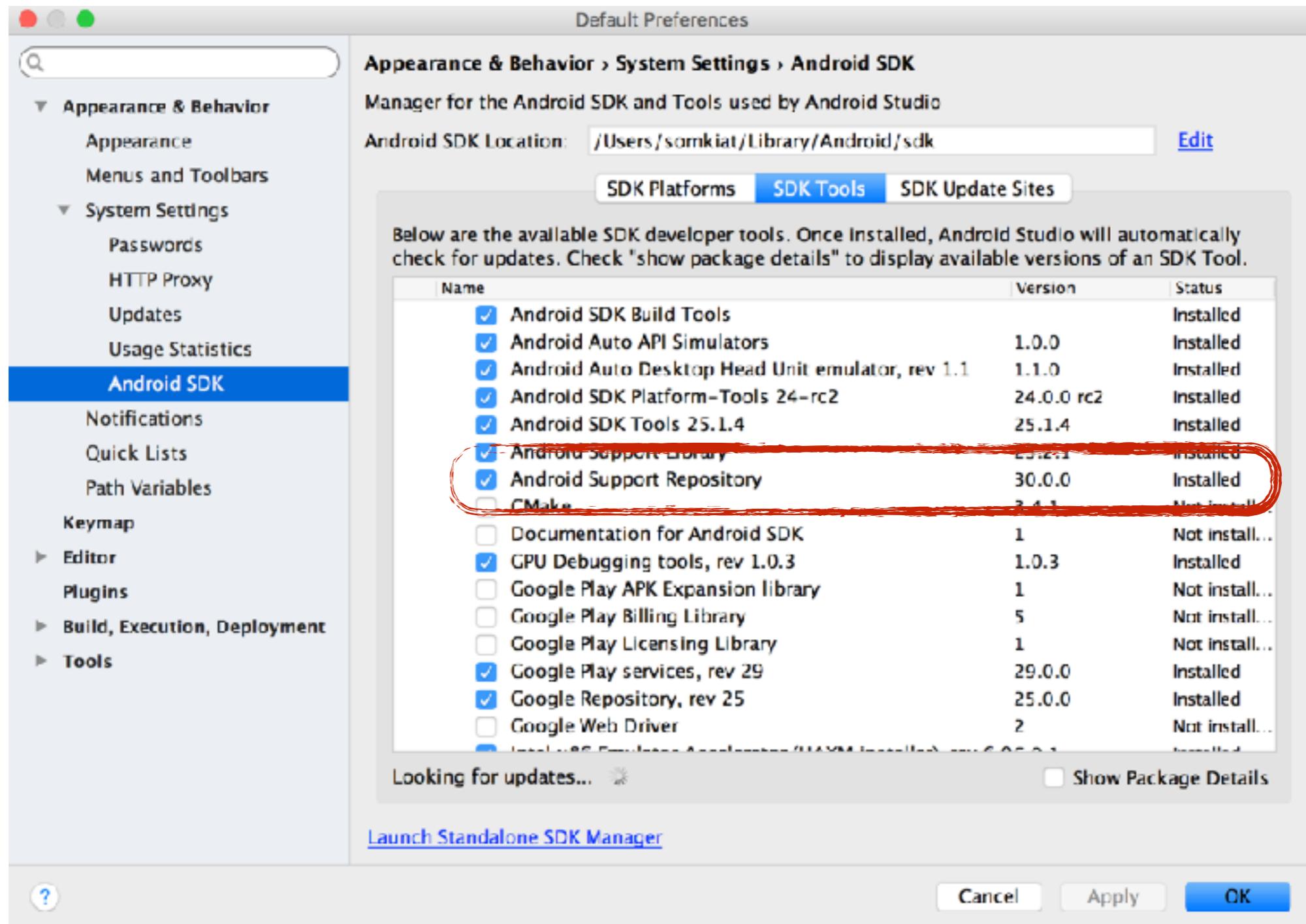
Configuration espresso in project

เรียนรู้การสร้าง UI test

Code coverage



# ติดตั้ง Android support repository



# เพิ่ม Espresso library

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    compile 'com.android.support:appcompat-v7:23.3.0'  
    compile 'com.android.support:support-annotations:23.3.0'  
    //Unit testing  
    testCompile 'junit:junit:4.12'  
    //UI testing with Espresso  
    androidTestCompile 'com.android.support:support-annotations:23.3.0'  
    androidTestCompile 'com.android.support.test:runner:0.5'  
    androidTestCompile 'com.android.support.test:rules:0.5'  
    androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.2'  
}
```



# Add instrumentation runner

```
android {  
    compileSdkVersion 23  
    buildToolsVersion '24.0.0 rc3'  
  
    defaultConfig {  
        applicationId "demo.somkiat.demounittest"  
        minSdkVersion 15  
        targetSdkVersion 23  
        versionCode 1  
        versionName "1.0"  
  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
}
```



# First UI test

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class CalculatorAddTest {

    public static final String ONE = "1";
    public static final String TWO = "2";
    public static final String RESULT = "3.0";

    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
        new ActivityTestRule<MainActivity>(MainActivity.class);

    @Test
    public void calculatorAdd() {
        onView(withId(R.id.operand_one_edit_text)).perform(typeText(ONE));
        onView(withId(R.id.operand_two_edit_text)).perform(typeText(TWO));
        onView(withId(R.id.operation_add_button)).perform(click());
        onView(withId(R.id.operation_result_text_view)).check(matches(withText(RESULT)));
    }
}
```



# Run UI test

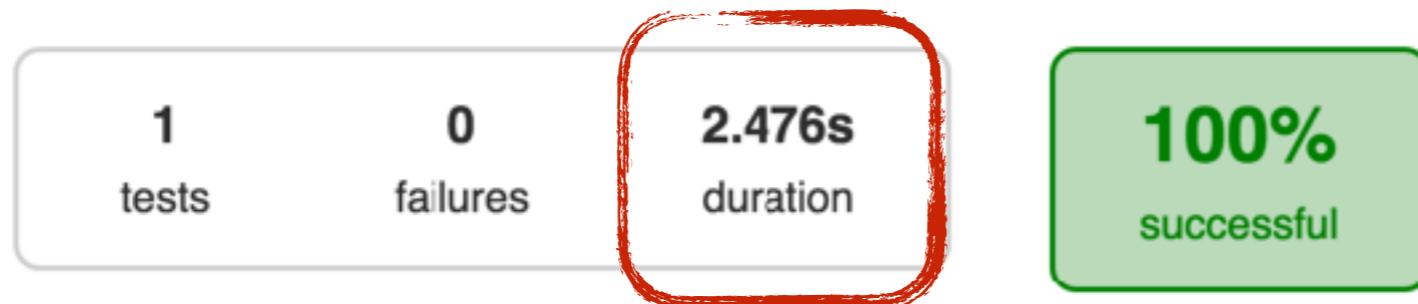
`./gradlew connectedAndroidTest`

`./gradlew cAT`



# Show report

## Test Summary



Packages

Classes

Package	Tests	Failures	Duration	Success rate
<a href="#">demo.somkiat.demounittest</a>	1	0	2.476s	100%

Generated by [Gradle 2.10](#) at May 9, 2016, 6:22:50 AM



# Show coverage report

 [debug](#) >  [demo.somkiat.demounittest](#)

## demo.somkiat.demounittest

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Line
 <a href="#">MainActivity</a>		85%		50%	1	6	
 <a href="#">Calculator</a>		100%		n/a	0	2	
Total	13 of 91	86%	1 of 2	50%	1	8	



# ແບບຝຶກຫັດ

ບວກ ລບ ຄູລ ຮາຮ

Code coverage = 100% !!



# Merge coverage unit + ui

```
def fileFilter = ['**/R.class', '**/R$.class', '**/BuildConfig.*', '**/Manifest*']
def debugTree = fileTree(dir: "${buildDir}/intermediates/classes/debug", excludes)
def mainSrc = "${project.projectDir}/src/main/java"

sourceDirectories = files([mainSrc])
classDirectories = files([debugTree])
executionData = files([
    "${buildDir}/jacoco/testDebugUnitTest.exec",
    "${buildDir}/outputs/code-coverage/connected/coverage.ec"
])
```



# Merge coverage unit + ui

```
./gradlew clean  
createDebugCoverageReport jacocoTestReport
```



# Merge coverage unit + ui

 app >  demo.somkiat.demounittest

## demo.somkiat.demounittest

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty
 <a href="#">MainActivity</a>		85%		50%	1	6
 <a href="#">Calculator</a>		100%		n/a	0	3
Total	13 of 95	86%	1 of 2	50%	1	9



# Android unit testing

ใช้งาน JUnit 4.12

<https://github.com/junit-team/junit4>



# Android assertions

Assertion	Description
assertEquals	Test that two values are the same
assertTrue	Test Boolean condition is true
assertFalse	Test Boolean condition is false
assertNull	Check that the object is null
assertNotNull	Check that the object is not null
assertSame	Test that both values refer to the same object reference
assertNotSame	Test that both values do not refer to the same object reference
assertThat	Test that the first value (object) matches the second value (or matcher)
fail	Test should always fail



# Unit options

@Before

@After

@BeforeClass

@AfterClass

@Test

@Test(expected=exception)

@Test(timeout=ms)



# Grouping tests

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

<http://googletesting.blogspot.com/2010/12/test-sizes.html>



# Example

```
@SmallTest  
@Test  
public void addition_isCorrect() throws Exception {  
    assertEquals(3, calculator.add(1, 2), 0);  
}  
  
@MediumTest  
@Test  
public void subtraction_isCorrect() throws Exception {  
    assertEquals(1, calculator.sub(2, 1), 0);  
}  
  
@LargeTest  
@Test  
public void divide_isCorrect() throws Exception {  
    assertEquals(2, calculator.div(2, 1), 0);  
}
```



# ระดมสมอง

Unit test ?

UI test ?

<https://plus.google.com/+AndroidDevelopers/posts/TPy1EeSaSg8>



บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Run with instrumentation runner

```
defaultConfig {  
    applicationId "demo.somkiat.demounittest"  
    minSdkVersion 15  
    targetSdkVersion 23  
    versionCode 1  
    versionName "1.0"  
  
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    testInstrumentationRunnerArgument "size", "medium,large"  
}  
-
```



# Run test

```
./gradlew clean cAT
```



# Parameterized tests

```
@Test  
public void addition_isCorrect() throws Exception {  
    assertEquals(3, calculator.add(1, 2), 0);  
    assertEquals(3, calculator.add(2, 1), 0);  
    assertEquals(10, calculator.add(8, 2), 0);  
    assertEquals(5, calculator.add(-1, 6), 0);  
}
```

ได้กลิ่นอะไรใหม่ ?



# Parameterized tests

```
@Test  
public void addition_isCorrect() throws Exception {  
    assertEquals(3, calculator.add(1, 2), 0);  
    assertEquals(3, calculator.add(2, 1), 0);  
    assertEquals(10, calculator.add(8, 2), 0);  
    assertEquals(5, calculator.add(-1, 6), 0);  
}
```

มา Refactor code กัน



# 1. Add @RunWith(Parameterized.class)

```
@RunWith(Parameterized.class)
public class CalculatorAddUnitTest {
    Calculator calculator = new Calculator();
```



## 2. Add datas

```
@RunWith(Parameterized.class)
public class CalculatorAddUnitTest {
    Calculator calculator = new Calculator();

    @Parameters
    public static List<Object[]> data() {
        return Arrays.asList(new Object[][] {
            {1, 2, 3},
            {2, 1, 3},
            {8, 2, 10},
            {-1, 6, 5},
        });
    }
}
```



# 3. Add constructor

```
@RunWith(Parameterized.class)
public class CalculatorAddUnitTest {
    Calculator calculator = new Calculator();

    public CalculatorAddUnitTest(int mOperandOne,
                                  int mOperandTwo,
                                  int mExpectedResult) {
        this.mOperandOne = mOperandOne;
        this.mOperandTwo = mOperandTwo;
        this.mExpectedResult = mExpectedResult;
    }
}
```



# 4. Use parameters in test

```
@RunWith(Parameterized.class)
public class CalculatorAddUnitTest {
    Calculator calculator = new Calculator();

    @Test
    public void addition_isCorrect(){
        assertEquals(this.mExpectedResult,
                    calculator.add(this.mOperandOne, this.mOperandTwo), 0);
    }
}
```



# Run test & Show report

The screenshot shows a Java IDE interface with a 'Run' tab selected. The title bar says 'Run CalculatorAddUnitTest'. The left sidebar has icons for file, project, build, run, test, and help. The main area displays a tree view of test results under 'CalculatorAddUnitTest (demo.somkiat.demounittes 1ms)'. It lists four test cases: [0], [1], [2], and [3], each with a green 'OK' status and a duration of 1ms or 0ms. To the right, a terminal window shows the command '/Library/Java/JavaVirtualMachines/jdk1.8.0\_65.jdk/Contents/Home/bin/java ...' and the message 'Process finished with exit code 0'. A progress bar at the top indicates 'All 4 tests passed - 1ms'.



# Run test & Show report

## Class demo.somkiat.demounittest.CalculatorAddUnitTest

[all](#) > [demo.somkiat.demounittest](#) > CalculatorAddUnitTest

4  
tests

0  
failures

0  
ignored

0.001s  
duration

100%  
successful

### Tests

Test	Duration	Result
addition_isCorrect[0]	0.001s	passed
addition_isCorrect[1]	0s	passed
addition_isCorrect[2]	0s	passed
addition_isCorrect[3]	0s	passed



# ແບບຝຶກຫັດ

ກຳການ run ຖຸກ ໃງ ກາຣທດສອບ  
ທັງ unit test ແລະ ui test



**With or without TDD,  
unit testing needs to become part of your development  
process**



# More testing

## Monkey testing



<http://developer.android.com/tools/help/monkey.html>



# Monkey testing

Stress testing

Command line tool

Random events



# How to use ?

```
$adb shell monkey -p <your package> -v <# of events>
```



# Workshop

```
$adb shell monkey -p demo.somkiat.demounittesting  
-v 20000
```

See result and fix it !!



# Good ideas in testing

**Multiple assertions per test are OK**

<http://www.slideshare.net/alacenski/android-testing-strategies>



# Good ideas in testing

Multiple assertions per test are OK

Test only one concept per test method



# Good ideas in testing

Multiple assertions per test are OK

Test only one concept per test method

You can write multiple test for one UI story



# Good ideas in testing

Multiple assertions per test are OK

Test only one concept per test method

You can write multiple test for one UI story

**Don't test other library's code**



# Good ideas in testing

Multiple assertions per test are OK

Test only one concept per test method

You can write multiple test for one UI story

Don't test other library's code

Look for ways to simplify your tests



# We need a continuous testing





Jenkins

Bamboo



TeamCity

> go™



Hudson



circleci

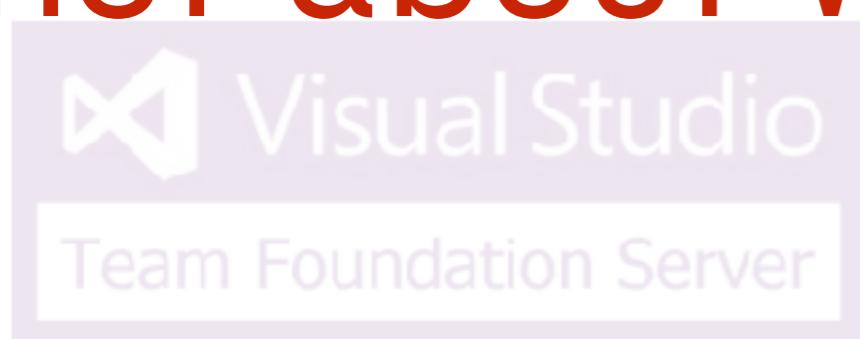




Jenkins

Bamboo

CI is about what people do  
not about what tools they use



Hudson



# CI is a practice

Discipline to integrate frequently



# CI is a practice

## Strive to make small change

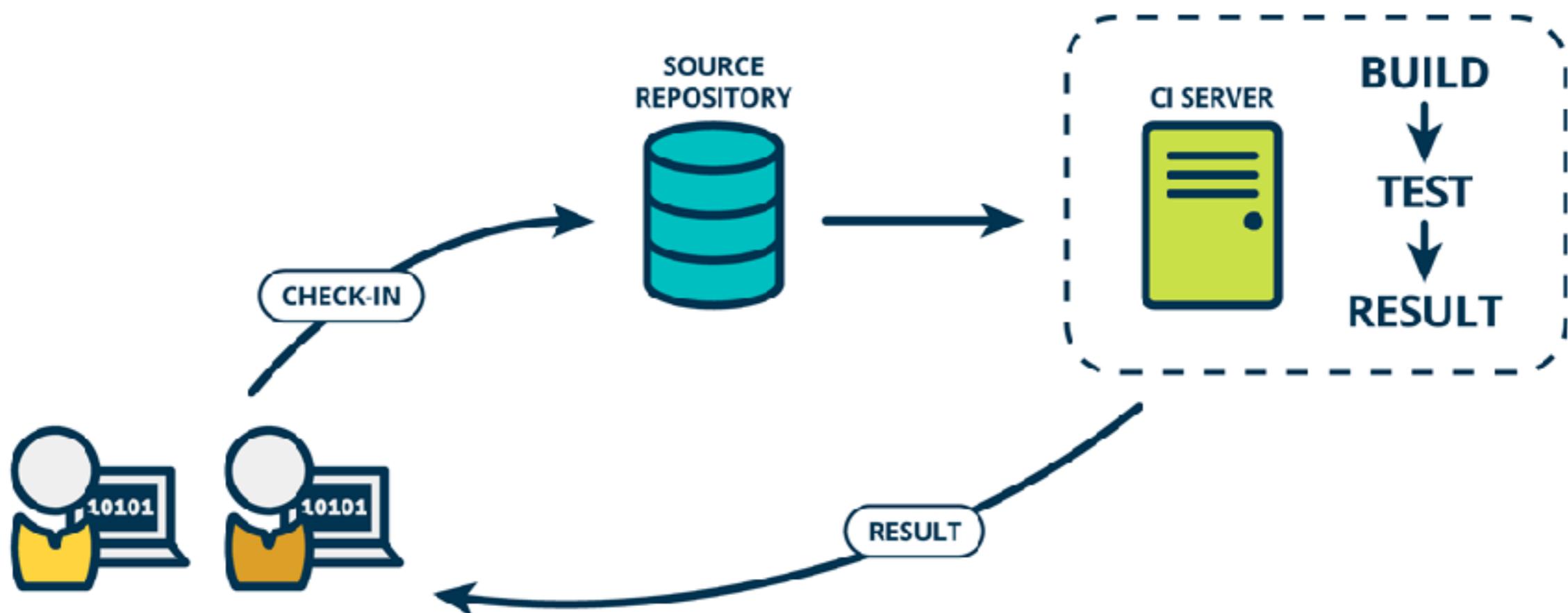


# CI is a practice

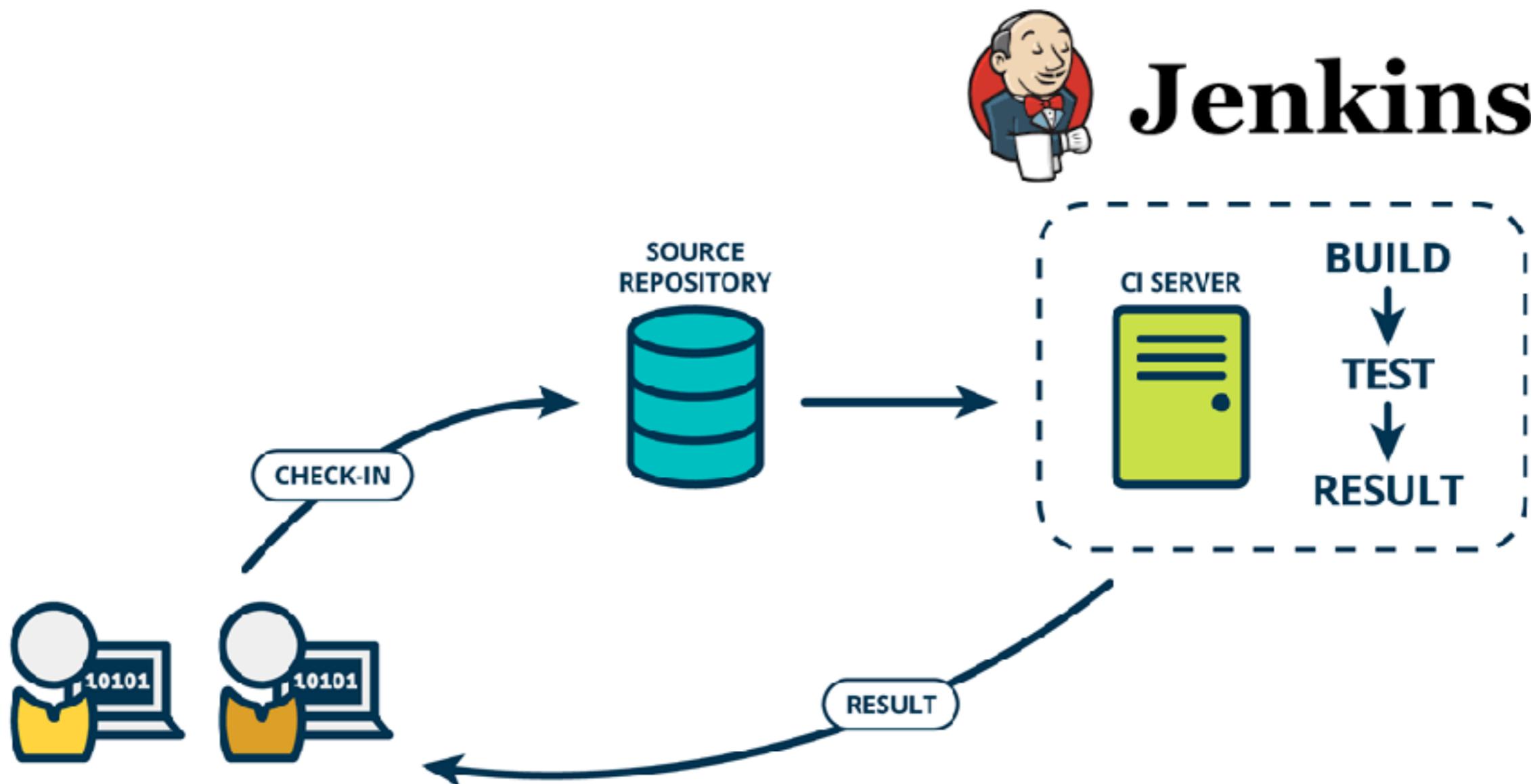
## Strive for **fast feedback**



# Continuous Integration



# Continuous Integration Server



# Workshop with Jenkins

ติดตั้ง Jenkins

ติดตั้ง plug-ins ต่าง ๆ

สร้างระบบการทดสอบแบบอัตโนมัติ



# Download Jenkins

Jenkins    [Downloads](#) ▾    [Participate](#) ▾    [Use-cases](#) ▾    [Blog](#)    [Documentation](#)    [Plugins](#)    [Wiki](#)    [Issues](#)    [Security](#)    [Press](#)

Conduct    Account

*Fork me on GitHub*



# Jenkins

Build great things at any scale

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

[Download Jenkins](#)

Get 1.651.1 LTS .war or the latest 2.1 weekly release

<https://jenkins-ci.org/>



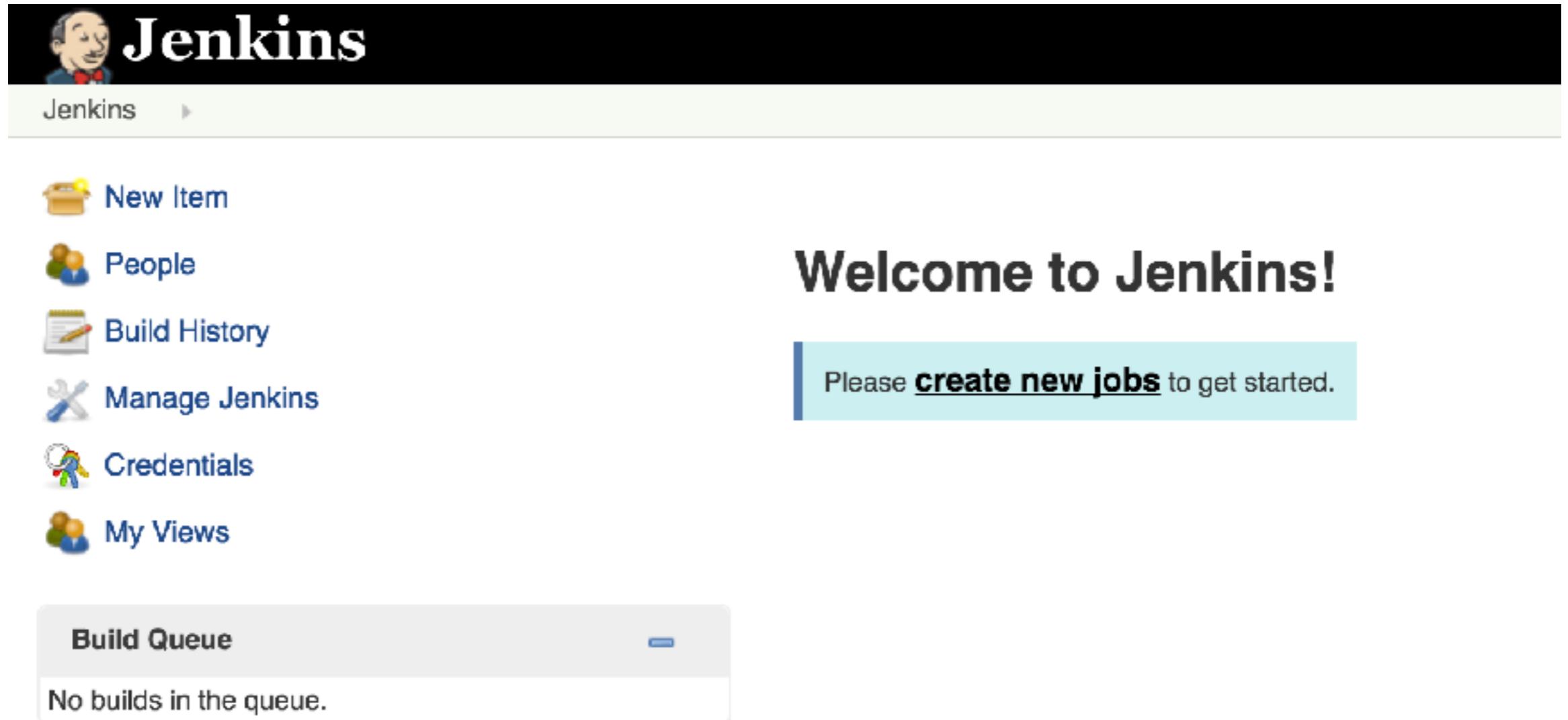
บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Start Jenkins

\$java -jenkins.war



# Welcome to Jenkins



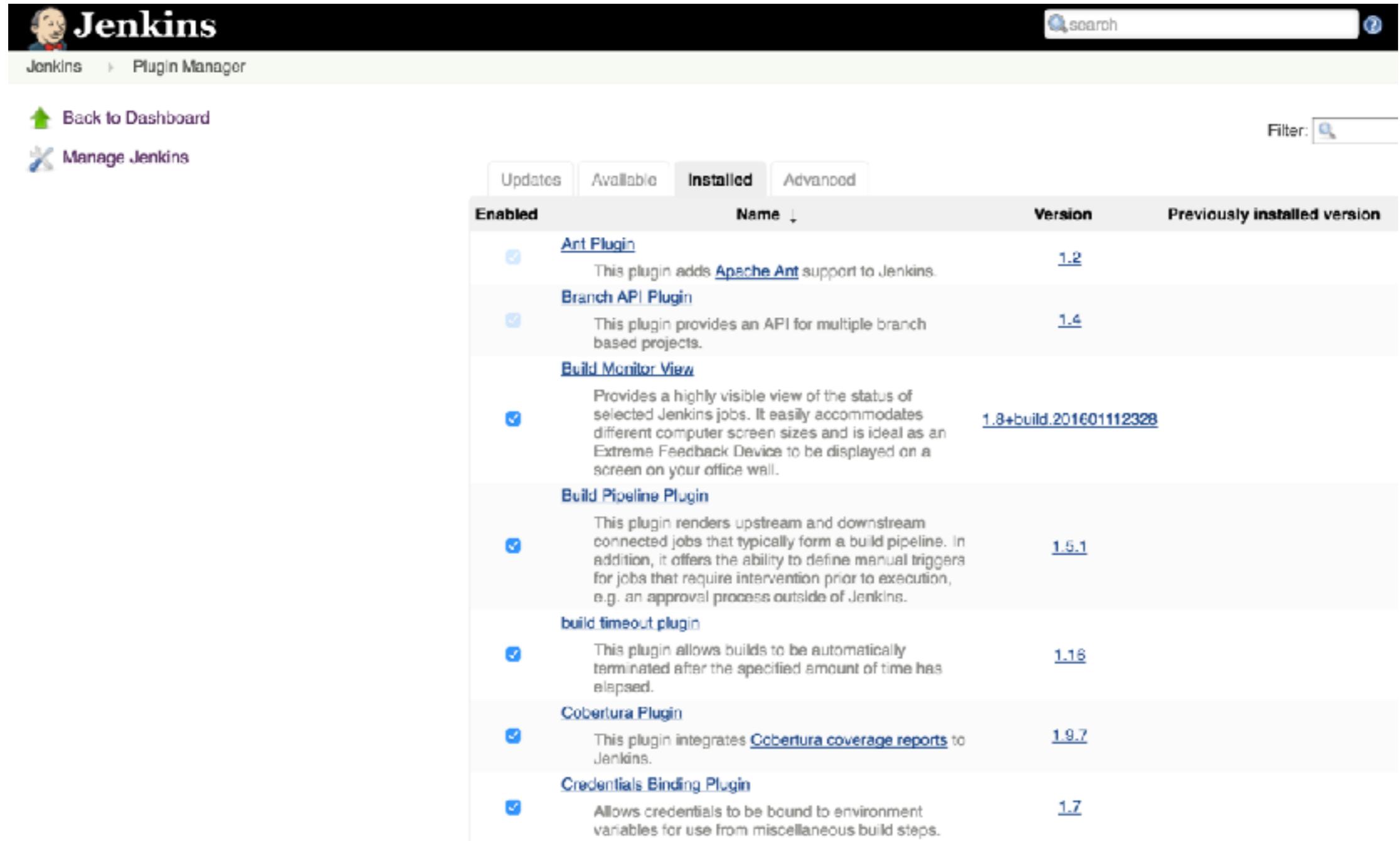
The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo. The main title "Welcome to Jenkins!" is centered above a teal box containing the text "Please [create new jobs](#) to get started." To the left of the main content is a sidebar with the following items:

- New Item
- People
- Build History
- Manage Jenkins
- Credentials
- My Views

Below the sidebar is a "Build Queue" section with the heading "Build Queue" and a message "No builds in the queue."



# Configure Jenkins



The screenshot shows the Jenkins Plugin Manager interface. At the top, there is a navigation bar with the Jenkins logo and a search bar. Below the navigation bar, there are links to 'Back to Dashboard' and 'Manage Jenkins'. On the right side, there is a 'Filter:' input field. The main area displays a table of installed plugins, with tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. The table has columns for 'Enabled', 'Name', 'Version', and 'Previously installed version'. The 'Installed' tab lists the following plugins:

Enabled	Name	Version	Previously installed version
<input checked="" type="checkbox"/>	<a href="#">Art Plugin</a> This plugin adds <a href="#">Apache Ant</a> support to Jenkins.	<a href="#">1.2</a>	
<input checked="" type="checkbox"/>	<a href="#">Branch API Plugin</a> This plugin provides an API for multiple branch based projects.	<a href="#">1.4</a>	
<input checked="" type="checkbox"/>	<a href="#">Build Monitor View</a> Provides a highly visible view of the status of selected Jenkins jobs. It easily accommodates different computer screen sizes and is ideal as an Extreme Feedback Device to be displayed on a screen on your office wall.	<a href="#">1.8+build.201601112328</a>	
<input checked="" type="checkbox"/>	<a href="#">Build Pipeline Plugin</a> This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.	<a href="#">1.5.1</a>	
<input checked="" type="checkbox"/>	<a href="#">build timeout plugin</a> This plugin allows builds to be automatically terminated after the specified amount of time has elapsed.	<a href="#">1.16</a>	
<input checked="" type="checkbox"/>	<a href="#">Cobertura Plugin</a> This plugin integrates <a href="#">Cobertura coverage reports</a> to Jenkins.	<a href="#">1.9.7</a>	
<input checked="" type="checkbox"/>	<a href="#">Credentials Binding Plugin</a> Allows credentials to be bound to environment variables for use from miscellaneous build steps.	<a href="#">1.7</a>	



# Create automated job

The screenshot shows the Jenkins dashboard with the title "Create automated job". On the left, there's a sidebar with links like "New Item", "People", "Build History", "Manage Jenkins", "Credentials", and "My Views". Below these are sections for "Build Queue" (empty) and "Build Executor Status". The main area is titled "Item name" with the value "01-PULL-CODE". It lists five options for job types: "Freestyle project" (selected), "Maven project", "Pipeline", "External Job", and "Folder". Each option has a brief description.

New Item

Item name **01-PULL-CODE**

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any kind of build step for something other than software build.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces configuration.

**Pipeline**  
Orchestrates long-running activities that can span multiple build slaves. Suitable for building and organizing complex activities that do not easily fit in free-style job type.

**External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, so you can use Jenkins as a dashboard of your existing automation system. See [the documentation](#).

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together under a single namespace, so you can have multiple things of the same name as long as they are in different folders.



# Configure job

Jenkins search ?

Jenkins > 01-PULL-CODE > configuration

find X Remove tabs

General Source Code Management Build Triggers Build Environment Bindings Build Post-build Actions

## General

Project name: 01-PULL-CODE

Description:

[Plain text] [Preview](#)

Discard Old Builds [?](#)

GitHub project [?](#)

This build is parameterized [?](#)

Throttle builds [?](#)

Disable Build (No new builds will be executed until the project is re-enabled.) [?](#)

Execute concurrent builds if necessary [?](#)

Restrict where this project can be run [?](#)

[Advanced...](#)

[Save All](#) [Apply All](#)



# Source code management

General    **Source Code Management**    Build Triggers    Build Environment    Bindings    Build    Post-build Actions

## Source Code Management

None  
 Git

**Repositories**

Repository URL  (**Please enter Git repository.**) X ?

Credentials

Advanced... Add Repository

**Branches to build**

Branch Specifier (blank for 'any')  X ?

Add Branch

**Repository browser**

**Additional Behaviours**

Subversion

Save All Apply All



# Build triggers

General   Source Code Management   **Build Triggers**   Build Environment   Bindings   Build   Post-build Actions

Repository browser   (Auto)   ?

Additional Behaviours   Add   ?

Subversion

---

## Build Triggers

- Trigger builds remotely (e.g., from scripts)   ?
- Build after other projects are built   ?
- Build periodically   ?
- Build when a change is pushed to GitHub   ?
- Poll SCM   ?

---

## Build Environment

- Delete workspace before build starts
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Use secret text(s) or file(s)   ?

---

**Save All**   **Apply All**



# Build triggers with poll SCM

## Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- Build when a change is pushed to GitHub ?
- Poll SCM ?

Schedule

\*\*\*\*\*|

?

**⚠ Do you really mean "every minute" when you say "\*\*\*\*\*"? Perhaps you meant "H \* \* \* \*" to poll once per hour**

Would last have run at Monday, May 9, 2016 at 9:57:10 PM Indochina Time; would next run at Monday, May 9, 2016 at 9:57:10 PM Indochina Time.

Ignore post-commit hooks

?



# Build

General   Source Code Management   Build Triggers   Build Environment   Bindings   **Build**   Post-build Actions

Schedule   `*****`   

**⚠ Do you really mean "every minute" when you say "\*\*\*\*\*"? Perhaps you meant "H \* \* \* \*" to poll once per hour**

Would last have run at Monday, May 9, 2016 at 9:57:10 PM Indochina Time; would next run at Monday, May 9, 2016 at 9:57:10 PM Indochina Time.

Ignore post-commit hooks  

## Build Environment

- Delete workspace before build starts
  - Abort the build if it's stuck
  - Add timestamps to the Console Output
  - Use secret text(s) or file(s)
- 

## Build

Add build step 

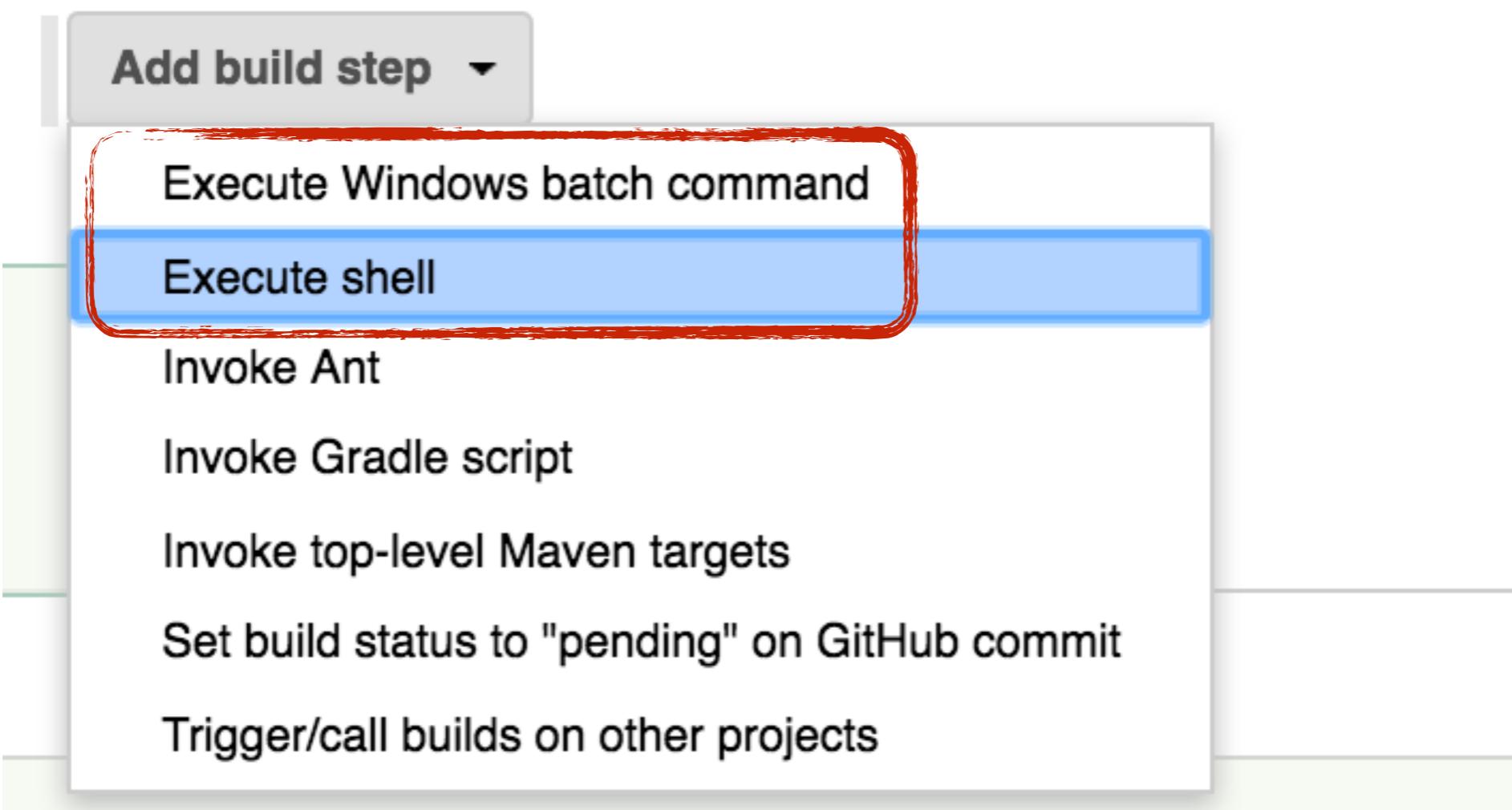
Save All

Apply All

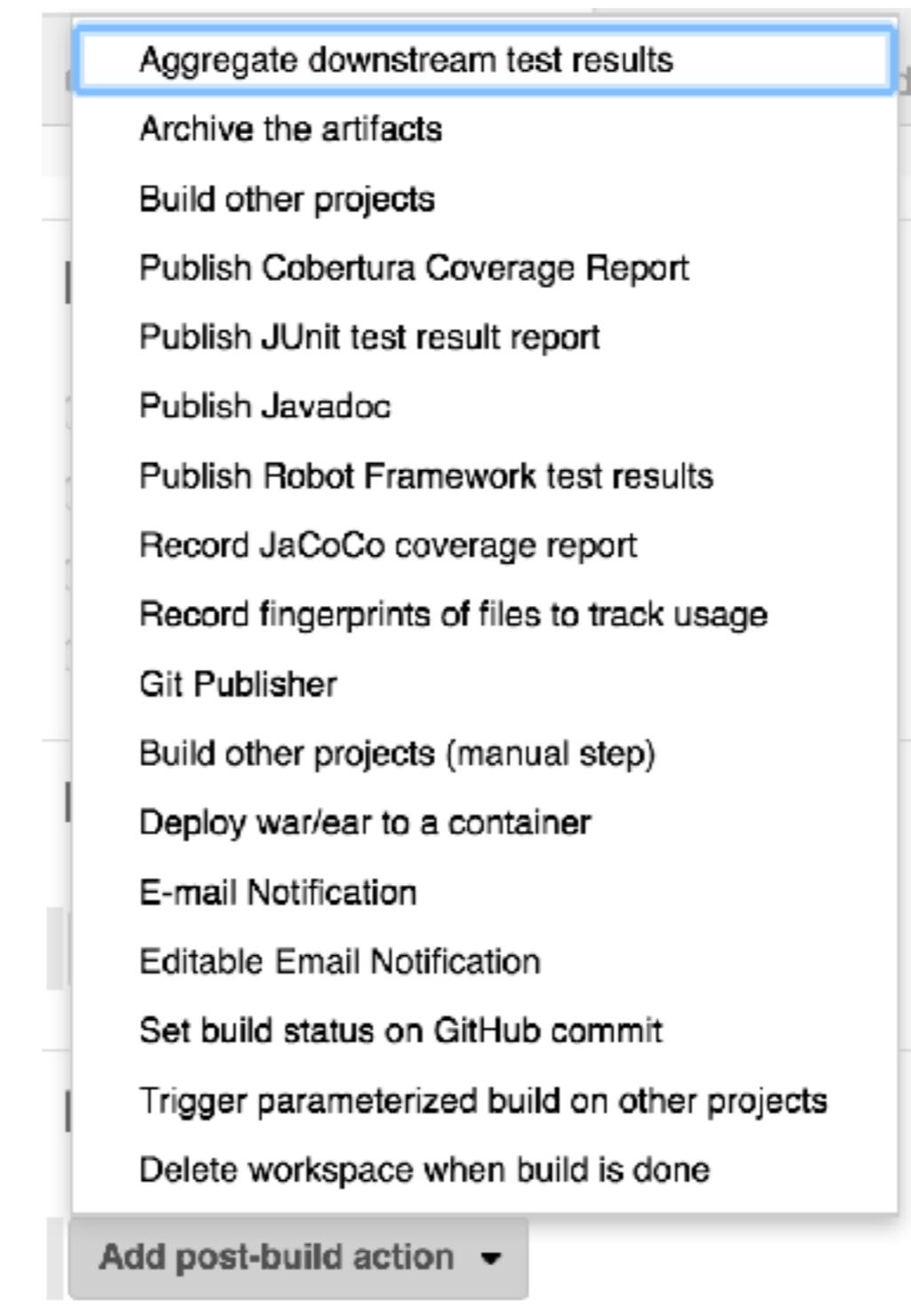


# Build with command line

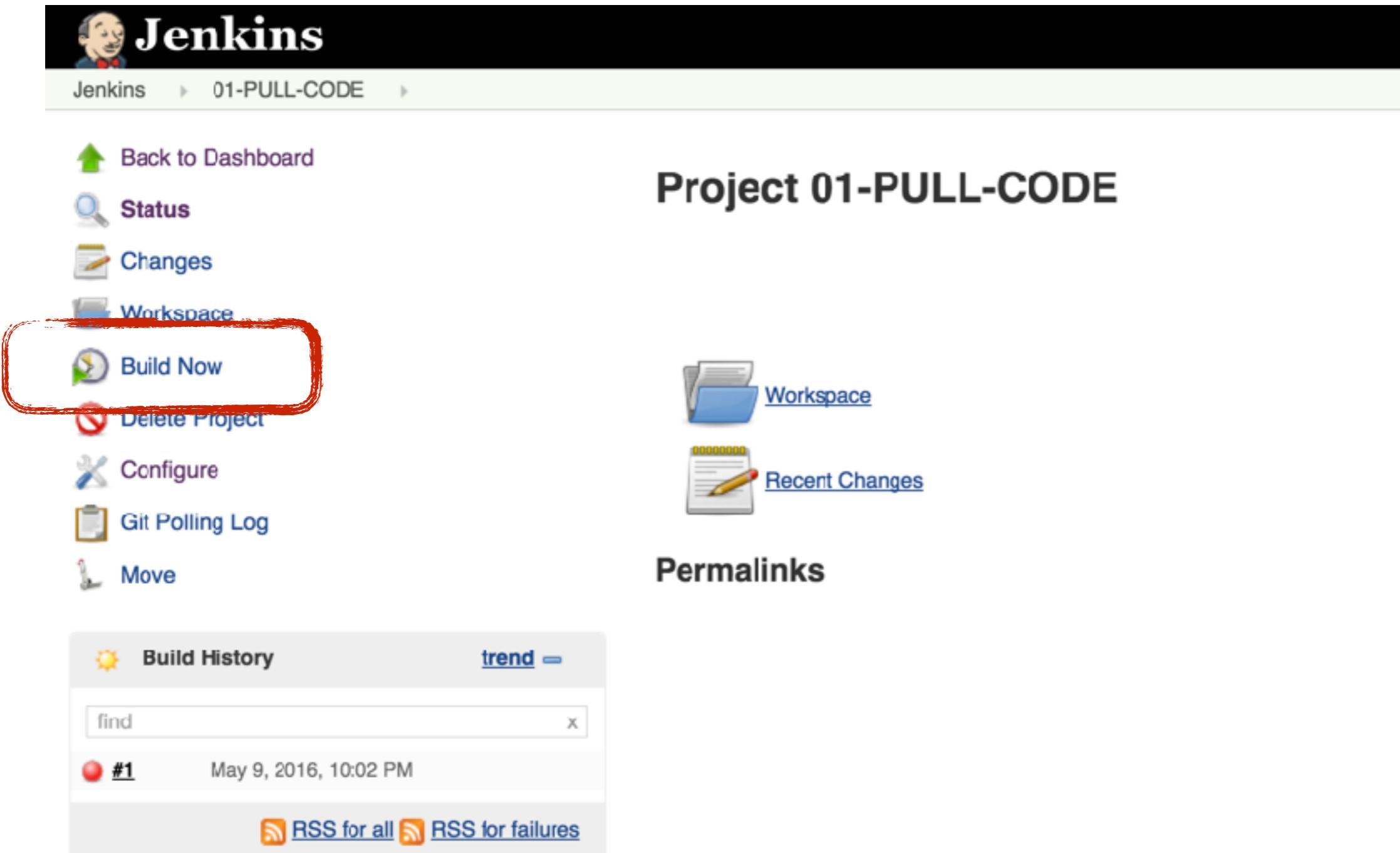
## Build



# Post build actions



# Save and Build Now !!



Jenkins

Jenkins > 01-PULL-CODE >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now**
- Delete Project
- Configure
- Git Polling Log
- Move

**Project 01-PULL-CODE**

[Workspace](#)

[Recent Changes](#)

**Permalinks**

**Build History**

trend —

find

#1 May 9, 2016, 10:02 PM

RSS for all RSS for failures



# แบบฝึกหัด

ออกแบบ Build pipeline ของระบบงาน

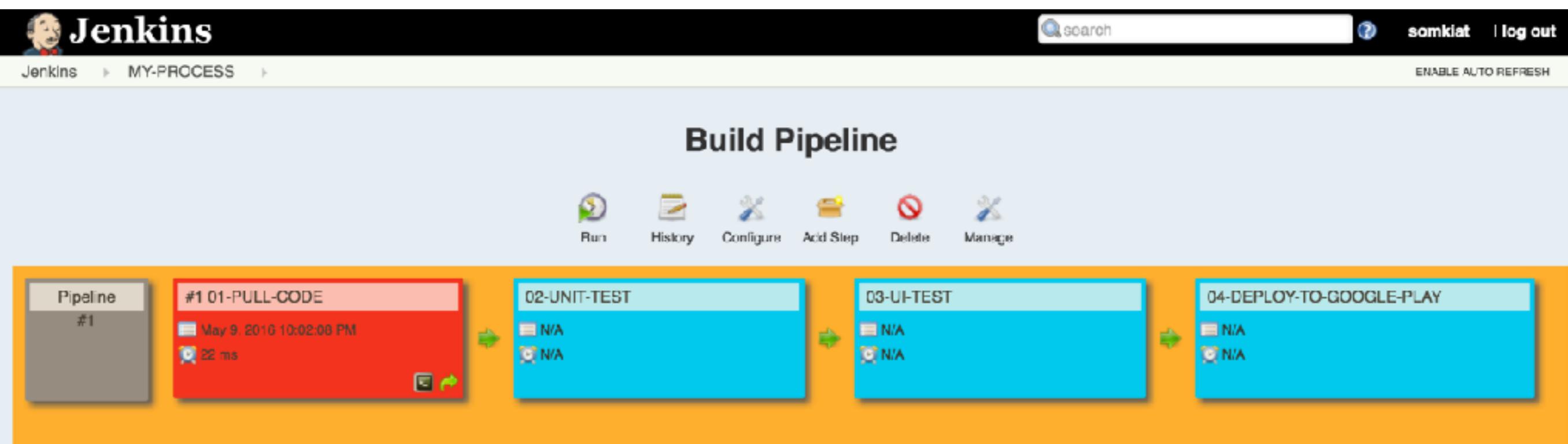


# ແບບືກ້ດ

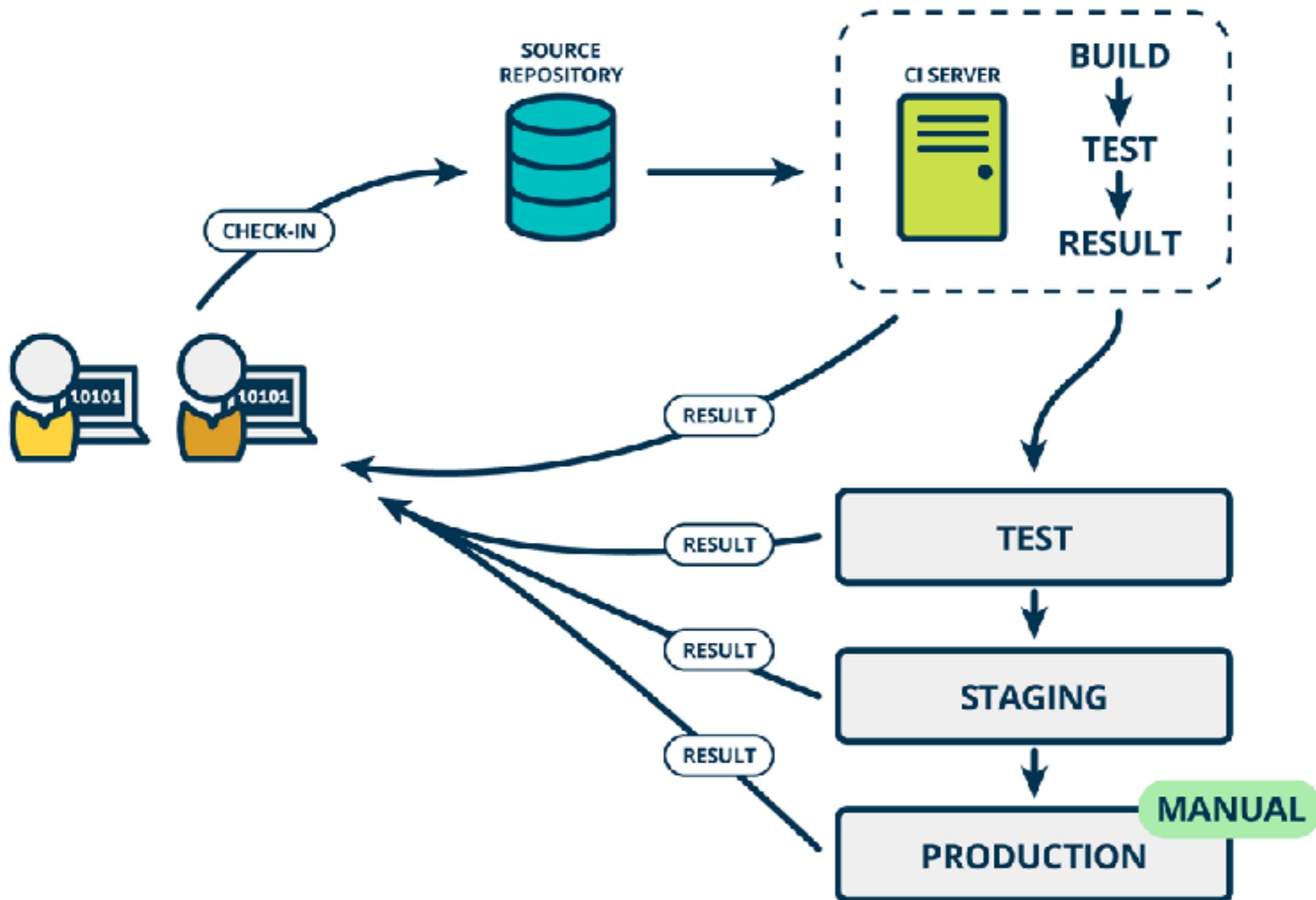
ສ້າງ Job ຕາມ build pipeline ໃນ Jenkins



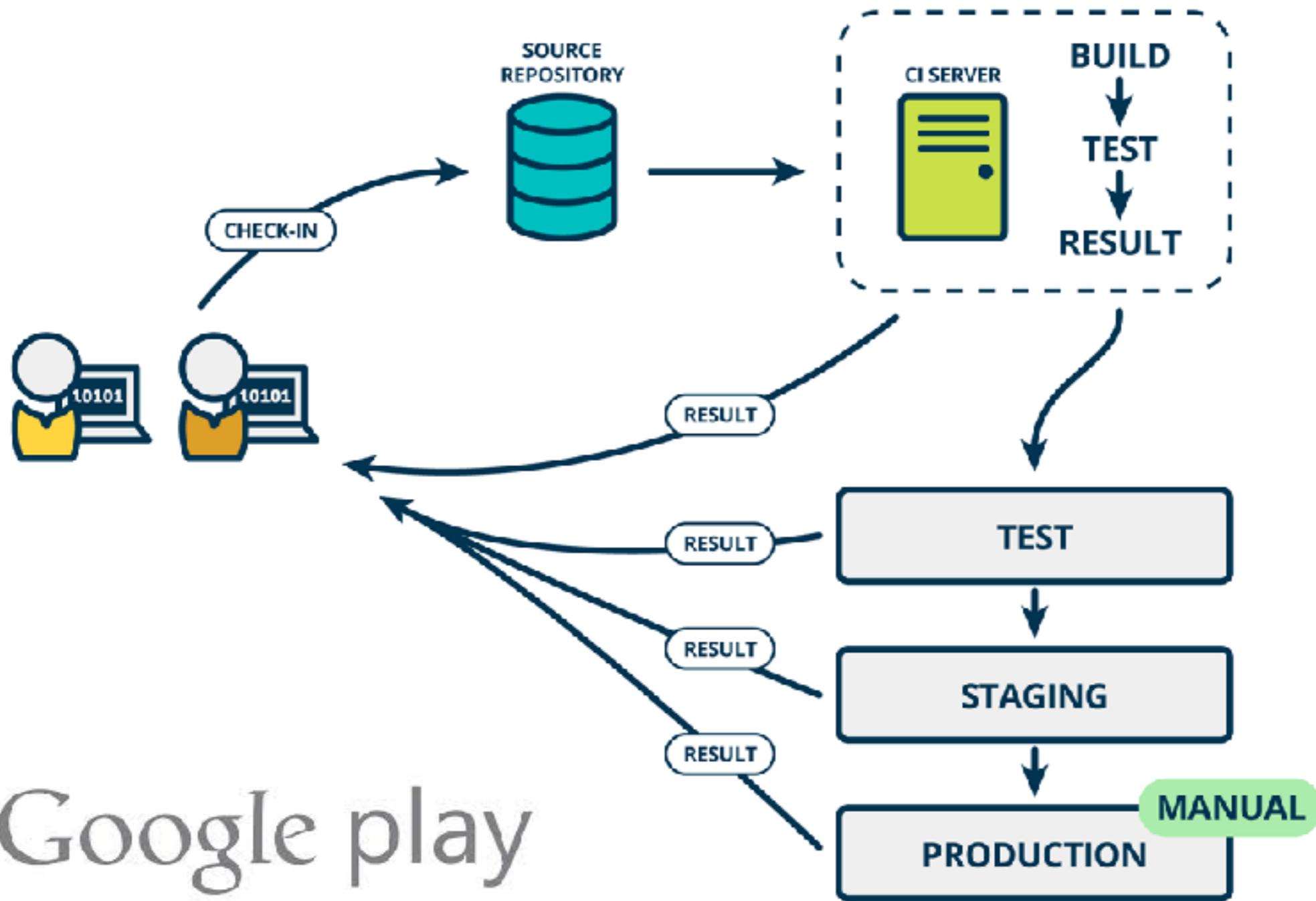
# Build pipeline



# Continuous Delivery



# Continuous Delivery



Google play



บริษัท สยามชานาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Fastlane.tools



GitHub

Automation done right

Used by thousands of mobile developers



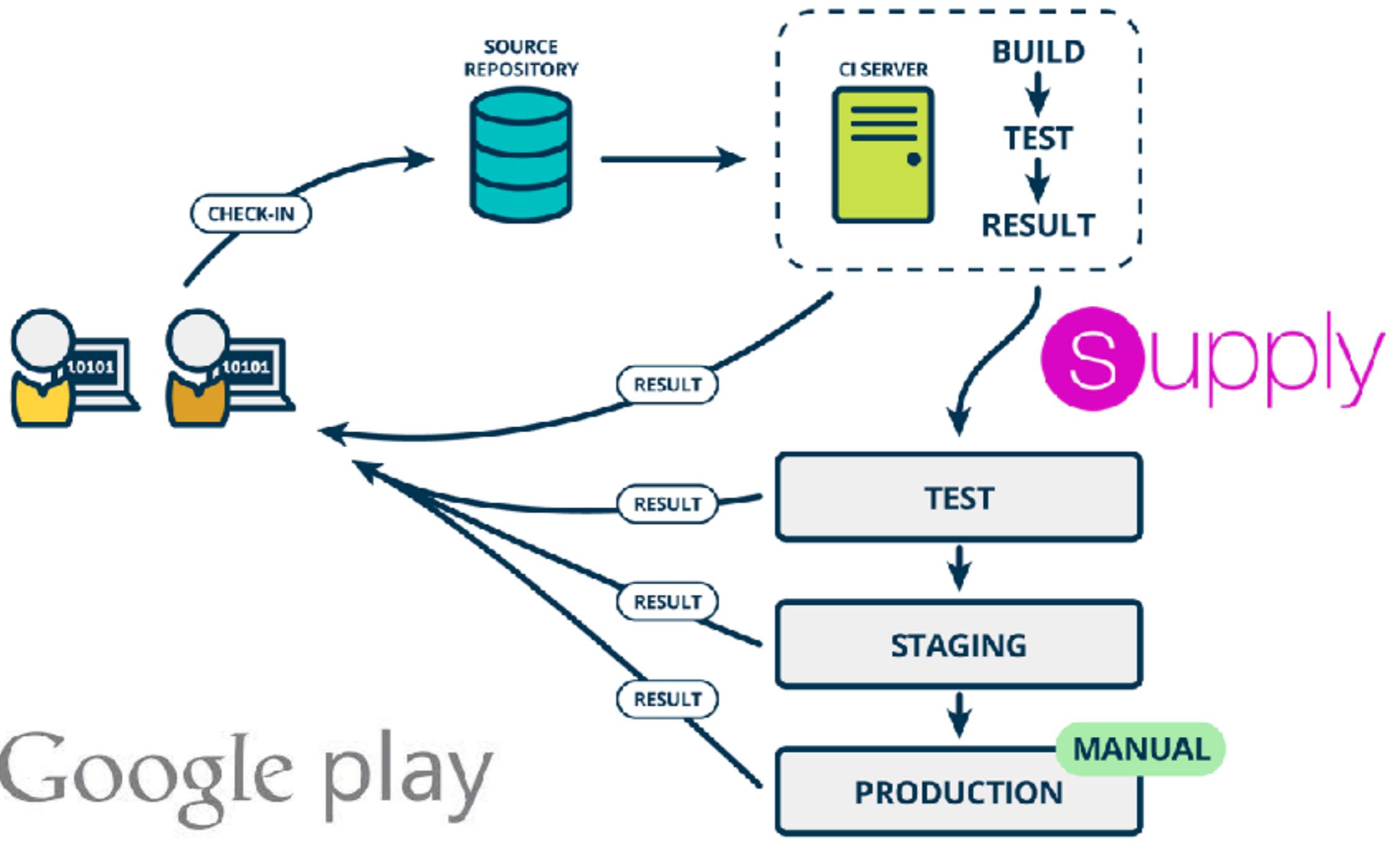
บริษัท สยามซัมนาณกิจ จำกัด และเพื่อนพ้องน้องพี่

# Installation

```
$sudo gem install fastlane --verbose
```



# Continuous Delivery



# Workshop with Supply

สมัคร Developer account

สร้าง Service account สำหรับ publish api

สร้าง Service account key

Upload APK to Google Play



# Create service account

<https://developers.google.com/android-publisher/>

The screenshot shows the 'API ACCESS' section of the Google Play Developer Console. On the left, there's a sidebar with icons for different settings: an Android phone, a game controller, a database, a gear (selected), and a warning triangle. The 'API access' option is highlighted with a teal bar. The main area has a heading 'API ACCESS' with a sub-section about the Google Play Developer Publishing API. It includes a note on security and a 'Learn more' link. Below this is a 'LINKED PROJECT' section showing 'Google Play Android Developer' linked with an 'Unlink' button. Another section shows 'Games Services Publishing API' with an 'On' button.

Google Play Developer Console

SETTINGS

- Account details
- User accounts & rights
- Activity log
- Email notifications
- API access**
- Linked accounts
- Developer page
- Cloud Test Lab

API ACCESS

The Google Play Developer Publishing API lets you publish and configure your apps from your own automated tools and processes. [Learn more](#)

Note on security: API users have access to perform actions similar to those available through the Google Play Developer Console. They must be used with the same care and managed with the same care as other Google Play developer console access credentials. Rights also apply to API requests.

LINKED PROJECT

Google Play Android Developer

Games Services Publishing API

Unlink

On



# Create service account

## SERVICE ACCOUNTS

Service accounts allow access to the Google Play Developer Publishing API on behalf of an application rather than accessing the API from an unattended server, such as an automated build server (e.g. Jenkins). All actions You can configure fine grained permissions for the service account on the 'User Accounts & Rights' page.

EMAIL

@developer.gserviceaccount.com

PERMISSIONS

**Grant access**

**Create Service Account**



# Create service account key

## API key

Identifies your project using a simple API key to check quota and access.  
For APIs like Google Translate.

## OAuth client ID

Requests user consent so your app can access the user's data.  
For APIs like Google Calendar.

## Service account key

Enables server-to-server, app-level authentication using robot accounts.  
For use with Google Cloud APIs.

## Help me choose

Asks a few questions to help you decide which type of credential to use.

[Create credentials ▾](#)



# Create service account key

## Create service account key

### Service account

New service account

Service account name 

fastlane-01

Service account ID

fastlane-01

@api-5228745590174066006-637549.iam.gserviceaccou



### Key type

Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.



JSON

Recommended



P12

For backward compatibility with code using the P12 format

**Create**

**Cancel**



# Create fastlane project

\$fastlane init



# Build signed APK

`./gradlew assembleRelease`



# Upload APK to Google play

Manual process



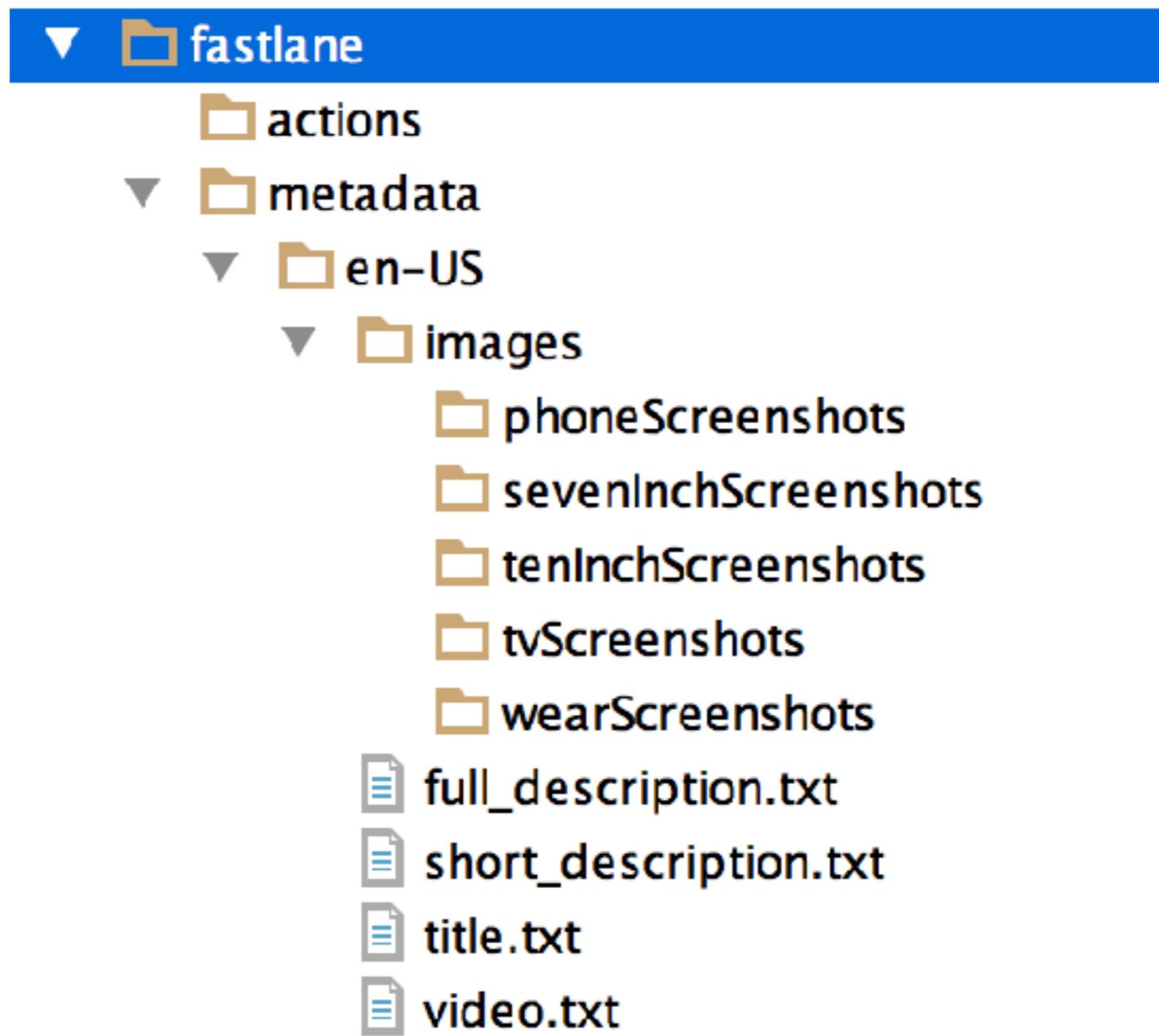
# Initial supply

\$supply init

```
Writing to metadata/en-US/title.txt...
Writing to metadata/en-US/short_description.txt...
Writing to metadata/en-US/full_description.txt...
Writing to metadata/en-US/video.txt...
Due to the limit of the Google Play API `supply` can't download your existing screenshots...
Due to the limit of the Google Play API `supply` can't download your existing feature graphics...
Downloading icon for en-US...
Downloading promoGraphic for en-US...
Downloading tvBanner for en-US...
Successfully stored metadata in 'metadata'
```



# Fastlane structure



# Deploy to alpha

```
desc "Deploy a new version to the Google Play"
lane :deploy do
  gradle(task: "assembleRelease")
  supply(track: "alpha")
end
```



# Deploy to alpha

\$fastlane deploy

Summary for supply 0.6.2	
track	alpha
package_name	demo.somkiat.demounittest
rollout	1.0
json_key	xxx-126555491470.json
apk	app/build/outputs/apk/app-release.apk
skip_upload_apk	false
skip_upload_metadata	false
skip_upload_images	false
skip_upload_screenshots	false

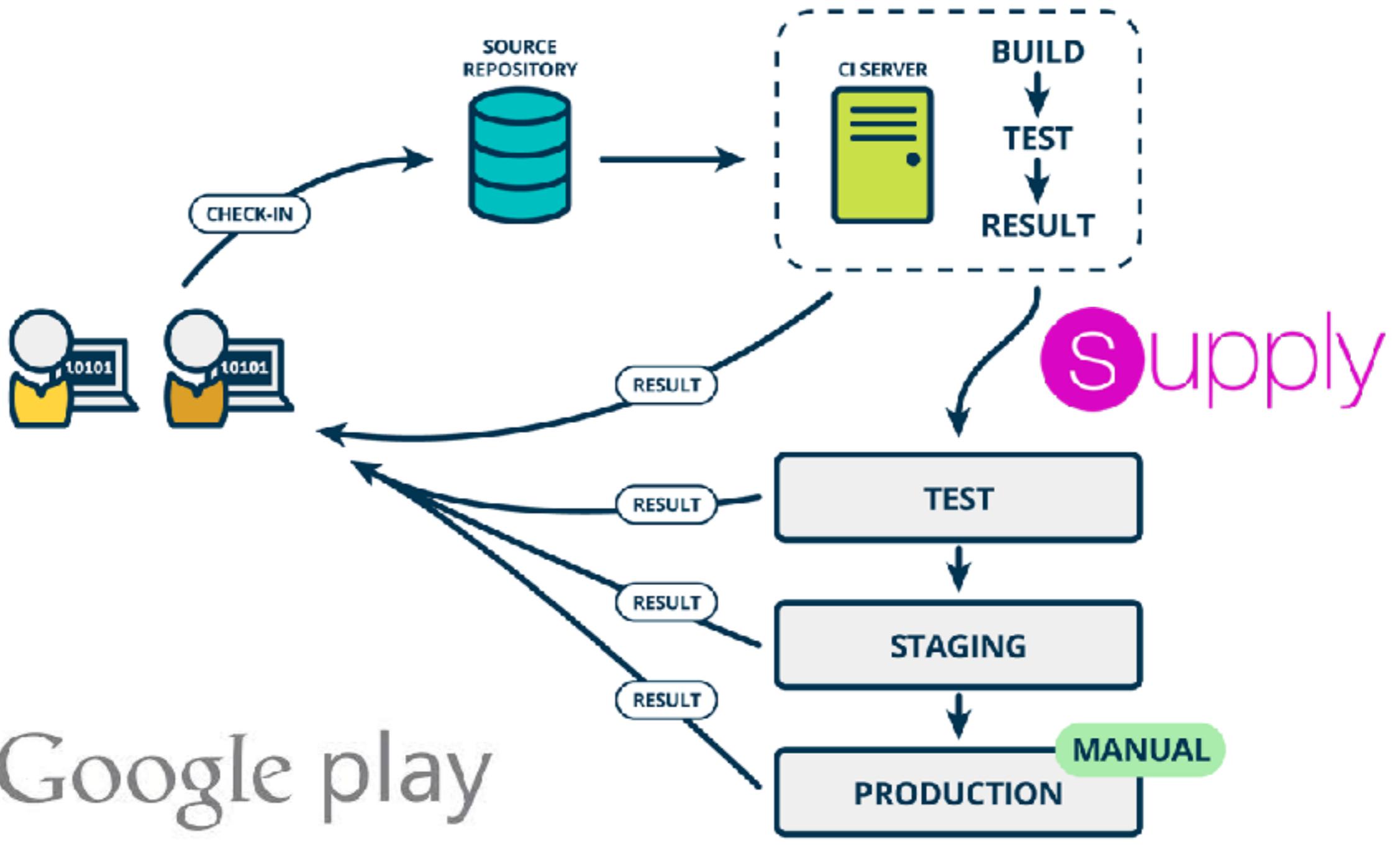
```
[12:21:33]: Preparing apk at path 'app/build/outputs/apk/app-release.apk' for upload...
[12:22:04]: Updating track 'alpha',...
[12:22:06]: Uploading all changes to Google Play...
[12:22:09]: Successfully finished the upload to Google Play
```

fastlane summary		
Step	Action	Time (in s)
1	Verifying required fastlane version	0
2	default_platform	0
3	gradle	15
4	supply	38

```
[12:22:09]: fastlane.tools finished successfully ━━
```



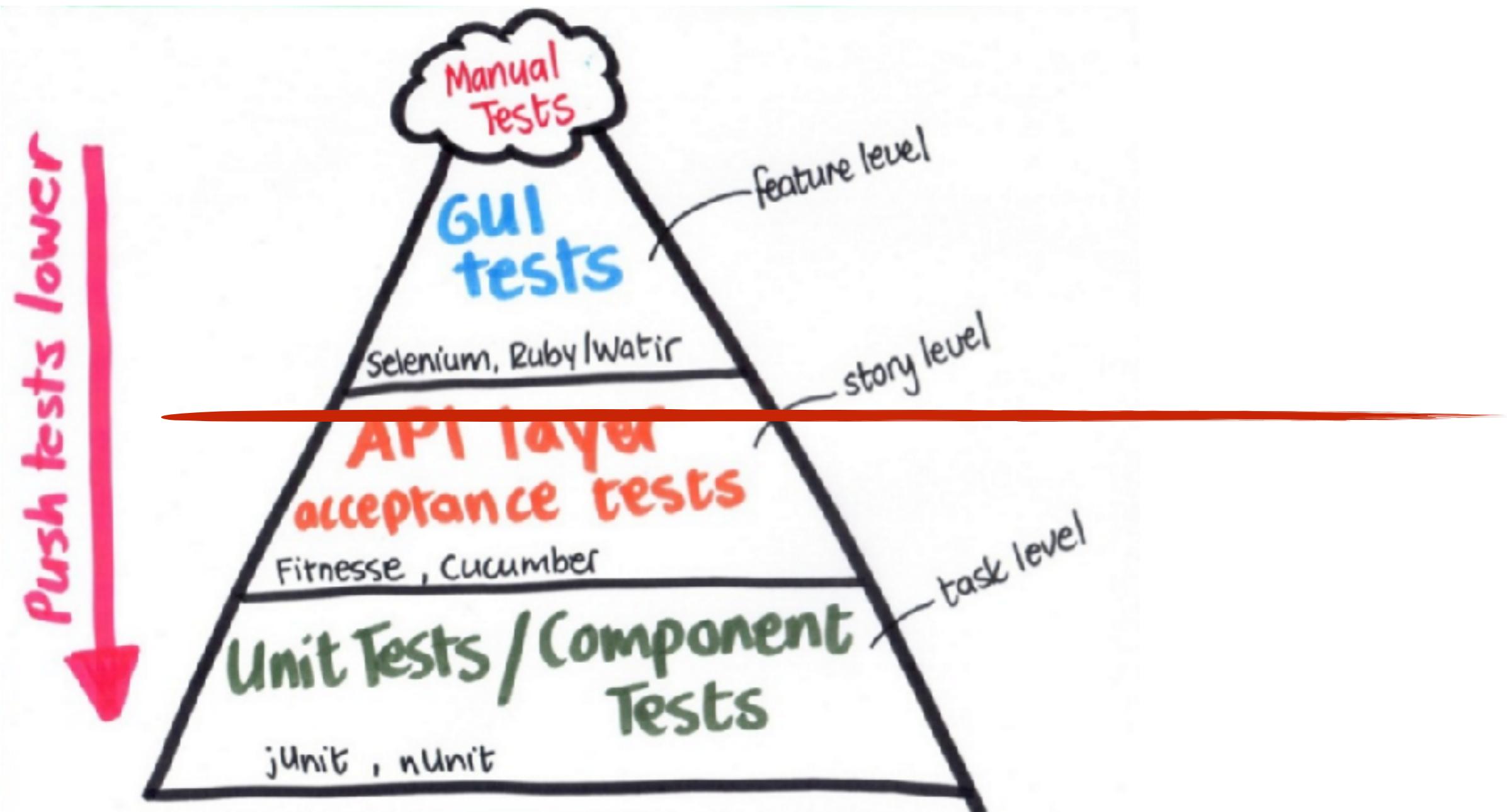
# Let's fun



# Mocking



# Testing for android



# We need small test !!

Feature	Small	Medium	Large
Network access	No	localhost only	Yes
Database	No	Yes	Yes
File system access	No	Yes	Yes
Use external systems	No	Discouraged	Yes
Multiple threads	No	Yes	Yes
Sleep statements	No	Yes	Yes
System properties	No	Yes	Yes
Time limit (seconds)	60	300	900+

<http://googletesting.blogspot.com/2010/12/test-sizes.html>



# Interact with others

Classes

Shared preferences

Time

SQLite database

Network



# Interact with others

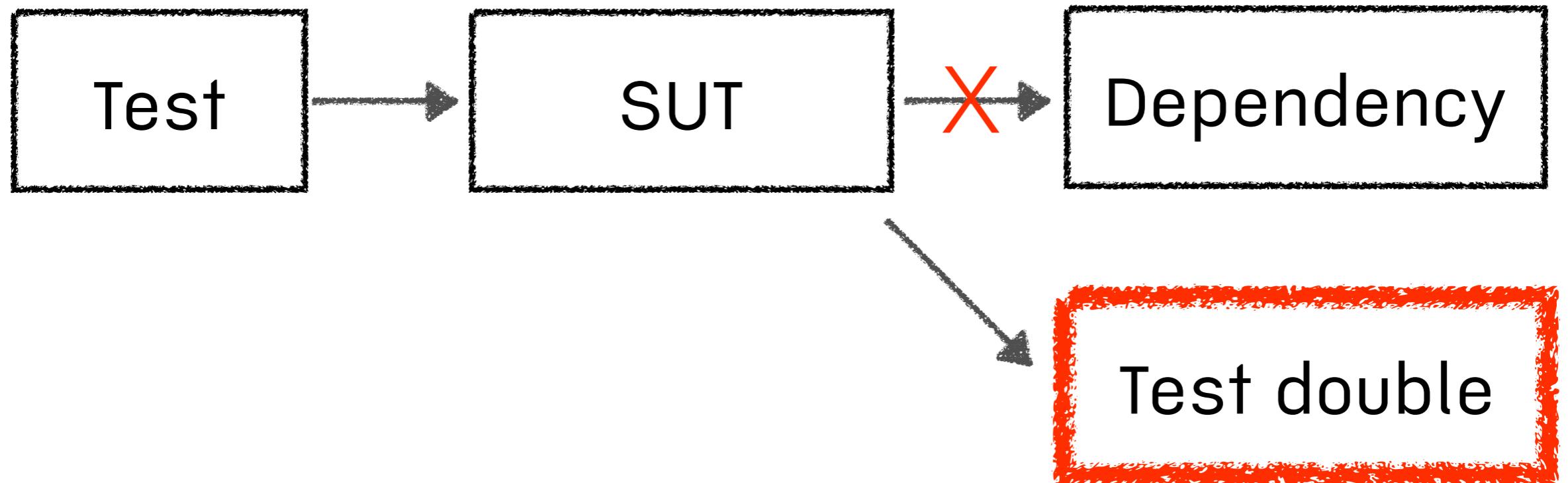
How to break dependencies ?



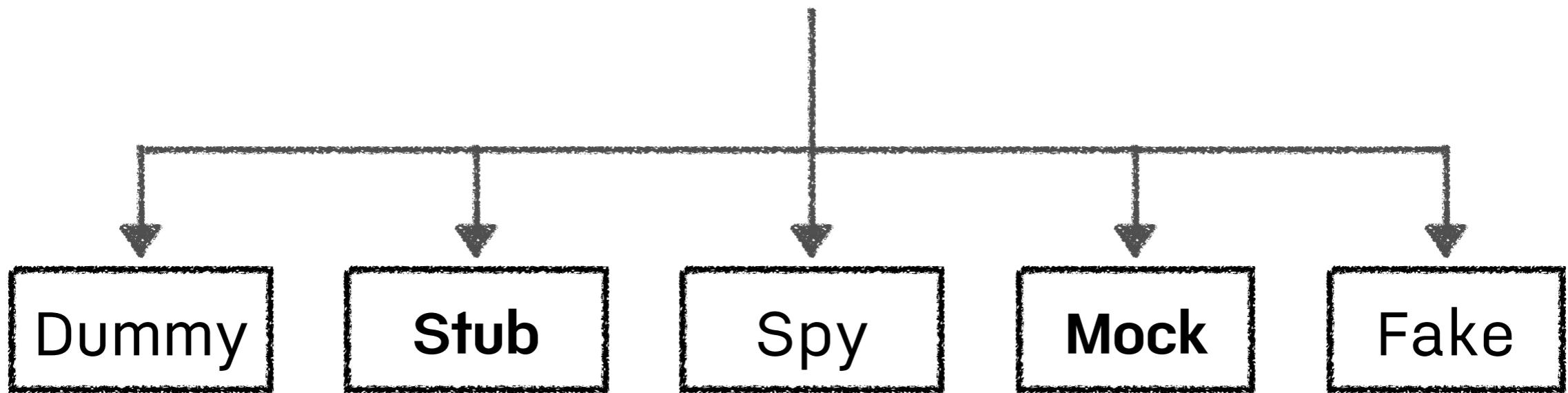
# Interact with others



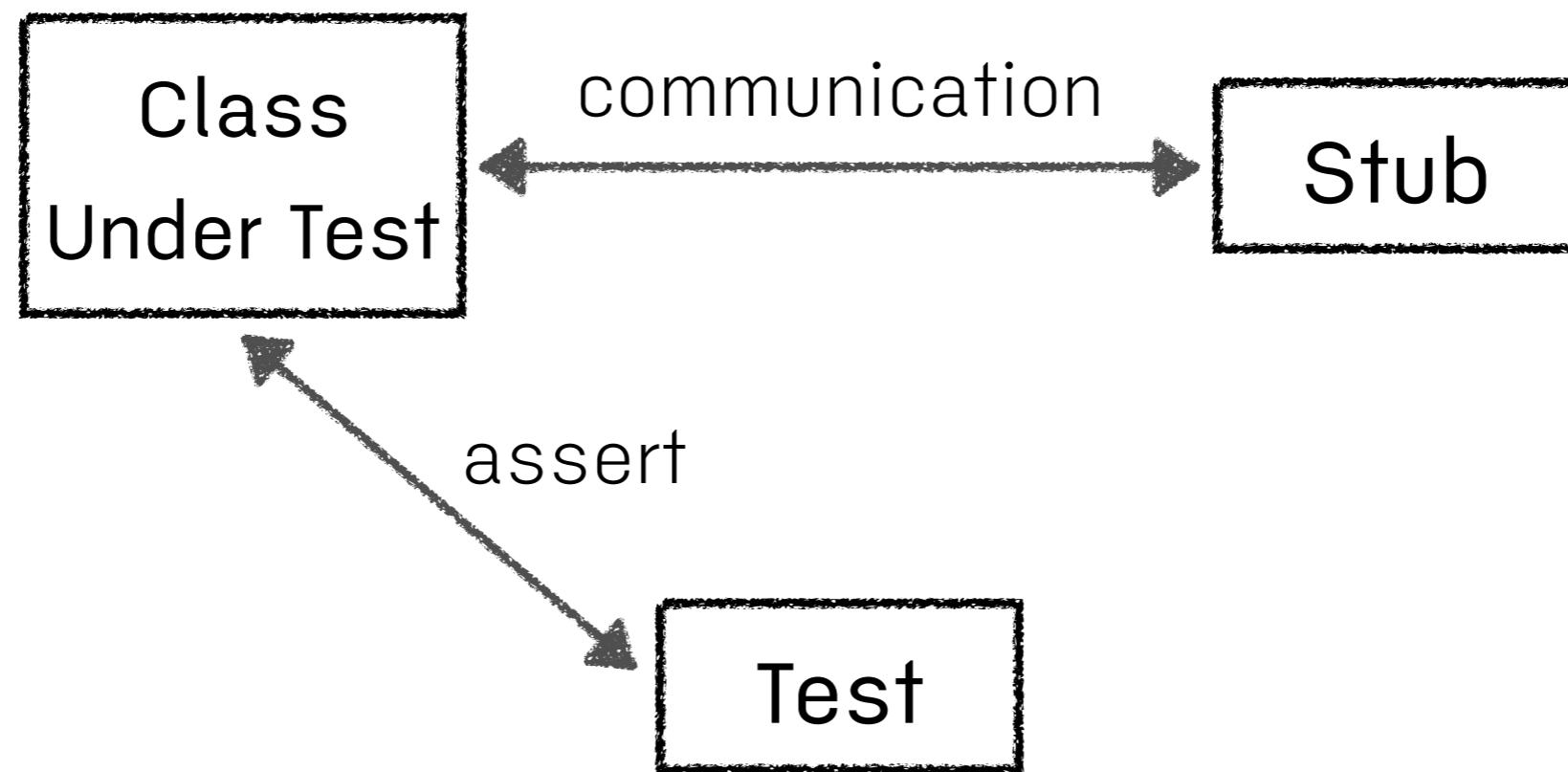
# Interact with others



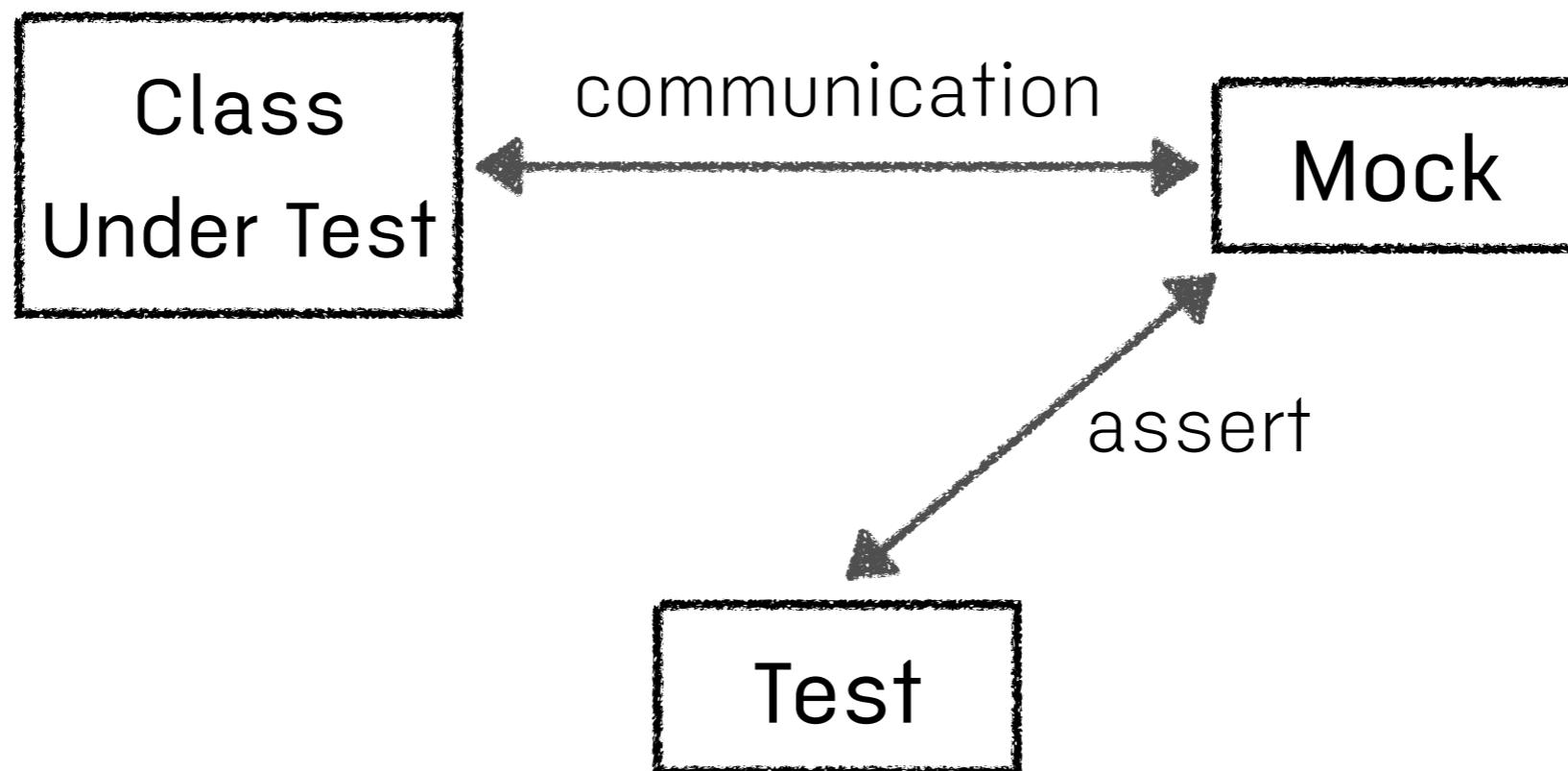
# Test double



# Stub



# Mock



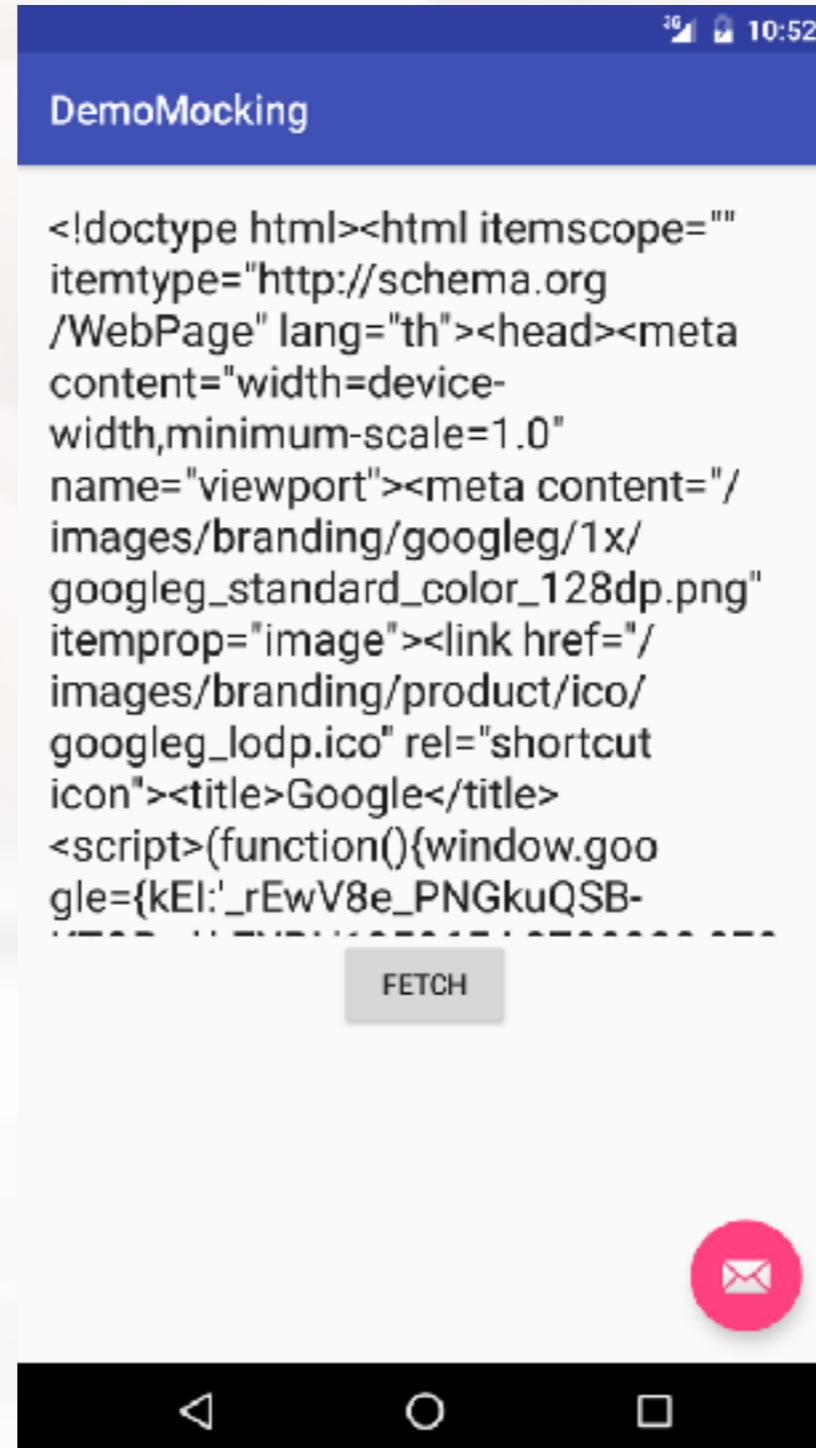
# Workshop with Mocking

Network with HttpURLConnection

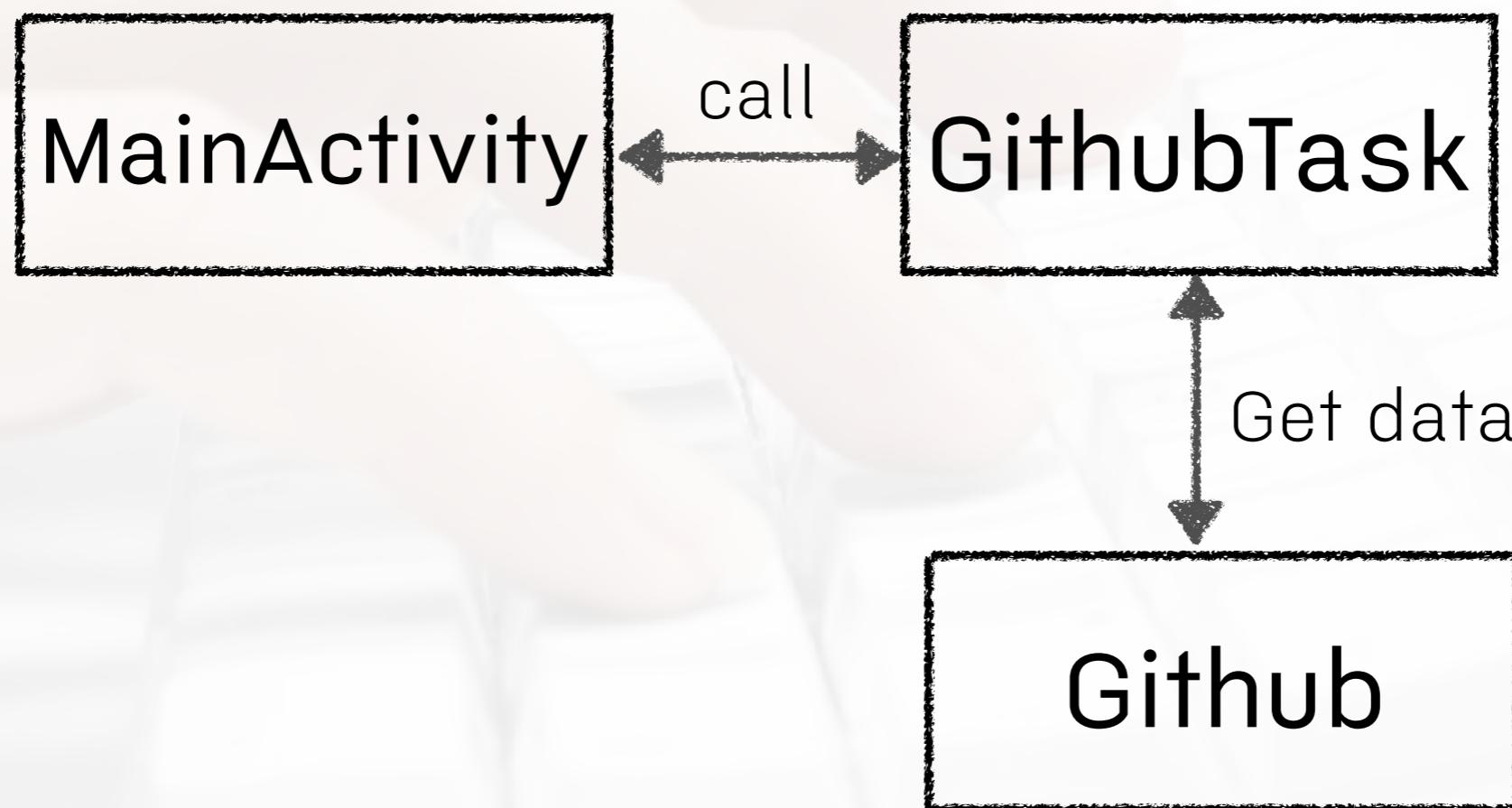
Network with Retrofit2



# HttpURLConnection



# HttpURLConnection



# Working with Mockito

```
dependencies {  
  
    //Unit test and Mocking  
    testCompile 'junit:junit:4.12'  
    testCompile 'org.hamcrest:hamcrest-library:1.3'  
    testCompile 'org.mockito:mockito-core:1.10.19'
```



# Example :: RunWith Mockito

```
@RunWith(MockitoJUnitRunner.class)
public class GithubTaskTest {

    public GitHubTask gitHubTask = Mockito.mock(GitHubTask.class);

    @Before
    public void setUp() {
        try {
            Mockito.when(gitHubTask.loadFromNetwork("http://www.google.com"))
                .thenReturn("<!doctype html><html itemscope=\"\" itemtype=\"\"");
        } catch (IOException e) {
            fail();
        }
    }

    @Test
    public void setGitHubTaskTest_ReturnsTrue() {
        try {
            assertThat(gitHubTask.loadFromNetwork("http://www.google.com"),
                containsString("doctype"));
        } catch (IOException e) {
            fail();
        }
    }
}
```



# Example :: Create mock

```
@RunWith(MockitoJUnitRunner.class)
public class GithubTaskTest {

    public GitHubTask gitHubTask = Mockito.mock(GitHubTask.class);

    @Before
    public void setUp() {
        try {
            Mockito.when(gitHubTask.loadFromNetwork("http://www.google.com"))
                .thenReturn("<!doctype html><html itemscope=\"\" itemtype=\"\"");
        } catch (IOException e) {
            fail();
        }
    }

    @Test
    public void setGitHubTaskTest_ReturnsTrue() {
        try {
            assertThat(gitHubTask.loadFromNetwork("http://www.google.com"),
                containsString("doctype"));
        } catch (IOException e) {
            fail();
        }
    }
}
```



# Example :: Create stub

```
@RunWith(MockitoJUnitRunner.class)
public class GithubTaskTest {

    public GitHubTask gitHubTask = Mockito.mock(GitHubTask.class);

    @Before
    public void setUp() {
        try {
            Mockito.when(gitHubTask.loadFromNetwork("http://www.google.com"))
                .thenReturn("<!doctype html><html itemscope=\"\" itemtype=\"\"");
        } catch (IOException e) {
            fail();
        }
    }

    @Test
    public void setGitHubTaskTest_ReturnsTrue() {
        try {
            assertThat(gitHubTask.loadFromNetwork("http://www.google.com"),
                containsString("doctype"));
        } catch (IOException e) {
            fail();
        }
    }
}
```



# Example :: Write test

```
@RunWith(MockitoJUnitRunner.class)
public class GithubTaskTest {

    public GitHubTask gitHubTask = Mockito.mock(GitHubTask.class);

    @Before
    public void setUp() {
        try {
            Mockito.when(gitHubTask.loadFromNetwork("http://www.google.com"))
                .thenReturn("<!doctype html><html itemscope=\"\" itemtype=\"\"");
        } catch (IOException e) {
            fail();
        }
    }

    @Test
    public void setGitHubTaskTest_ReturnsTrue() {
        try {
            assertThat(gitHubTask.loadFromNetwork("http://www.google.com"),
                containsString("doctype"));
        } catch (IOException e) {
            fail();
        }
    }
}
```

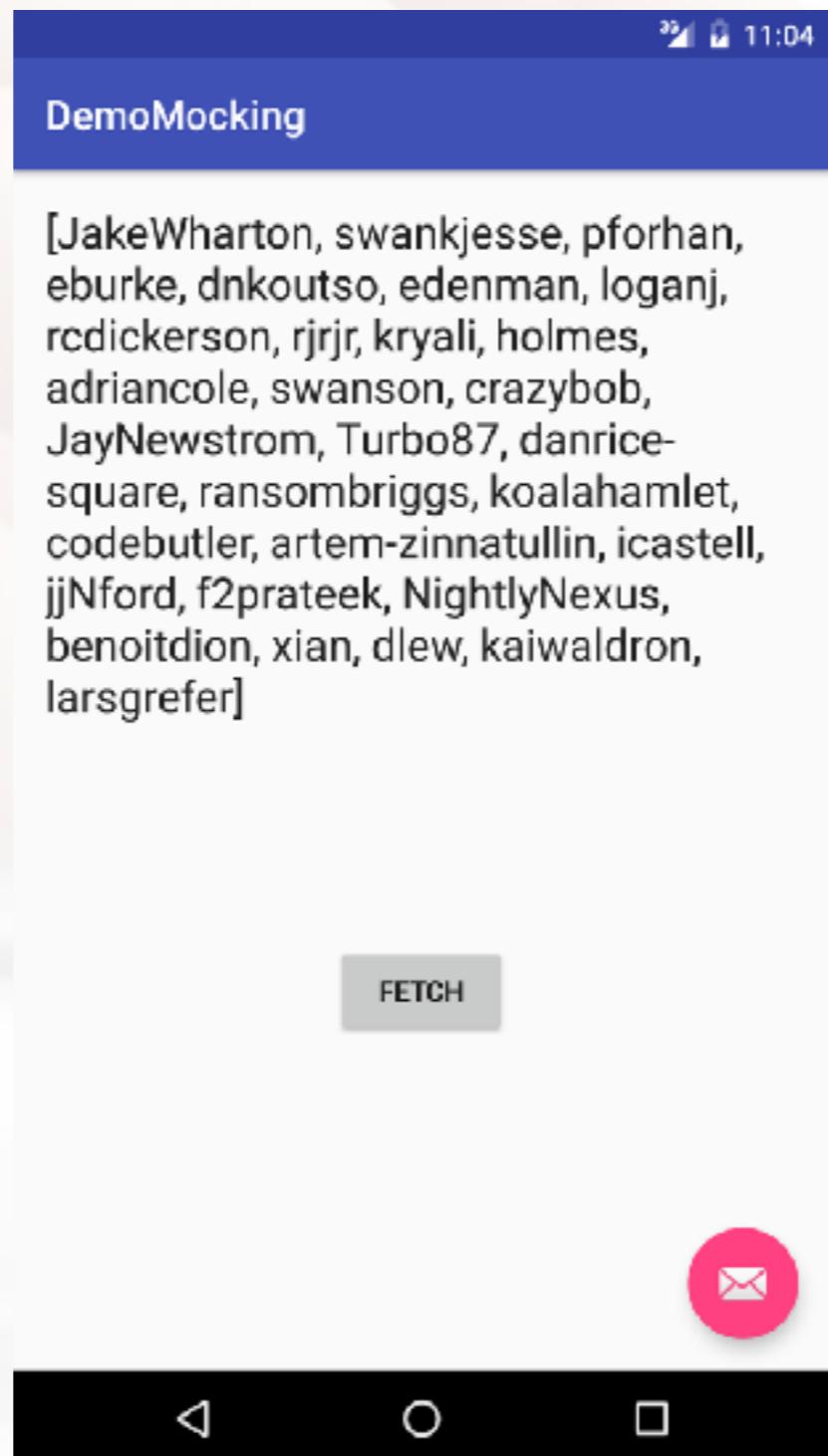


# Working with hardcoded

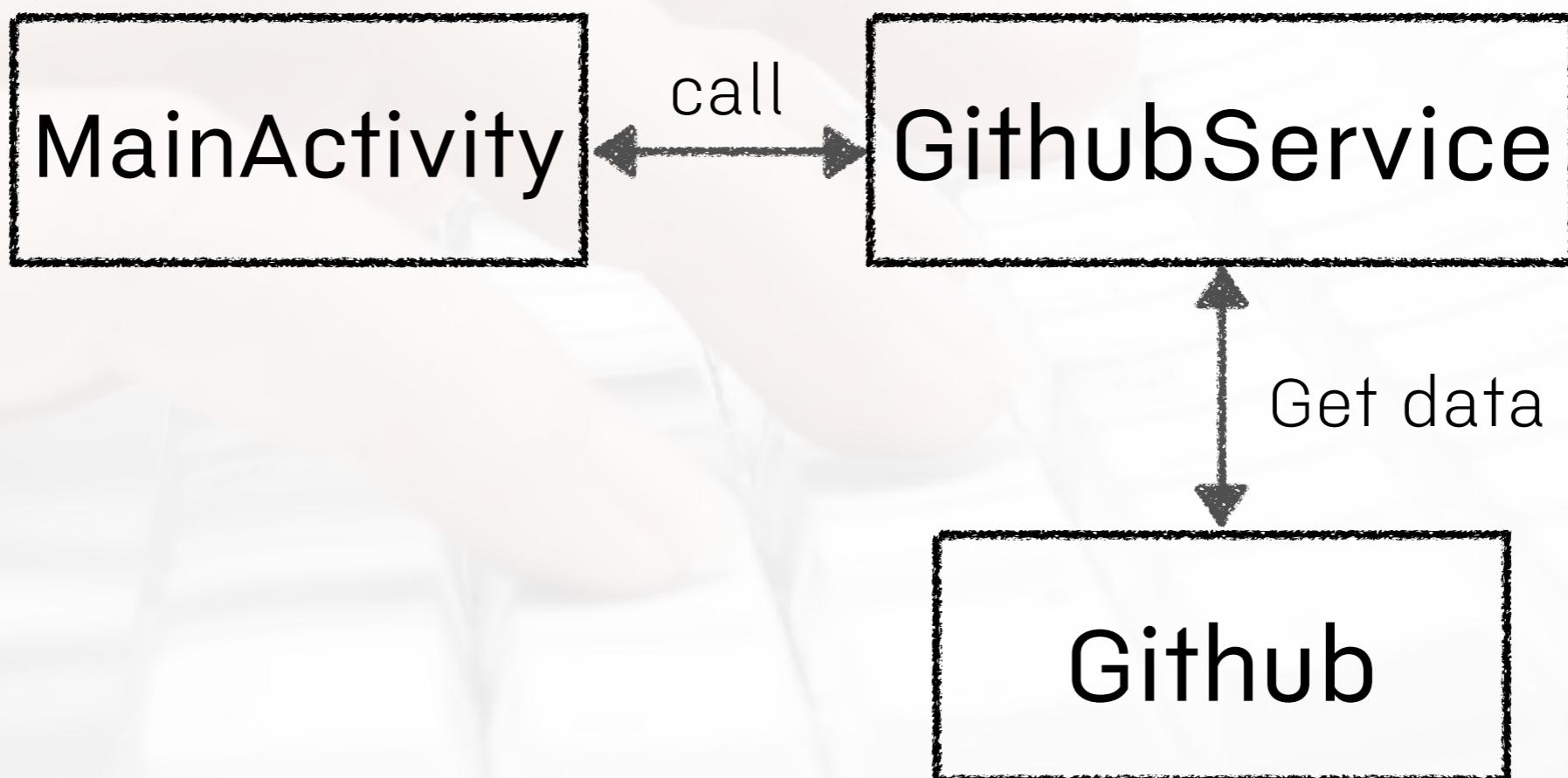
ลองทำดูสิ !!



# Retrofit2



# Retrofit2



# Workshop with Retrofit

MockWebServer

MockRetrofit



# Resources

[\*\*https://github.com/up1/course-automated-testing-for-android-app\*\*](https://github.com/up1/course-automated-testing-for-android-app)



# New topics



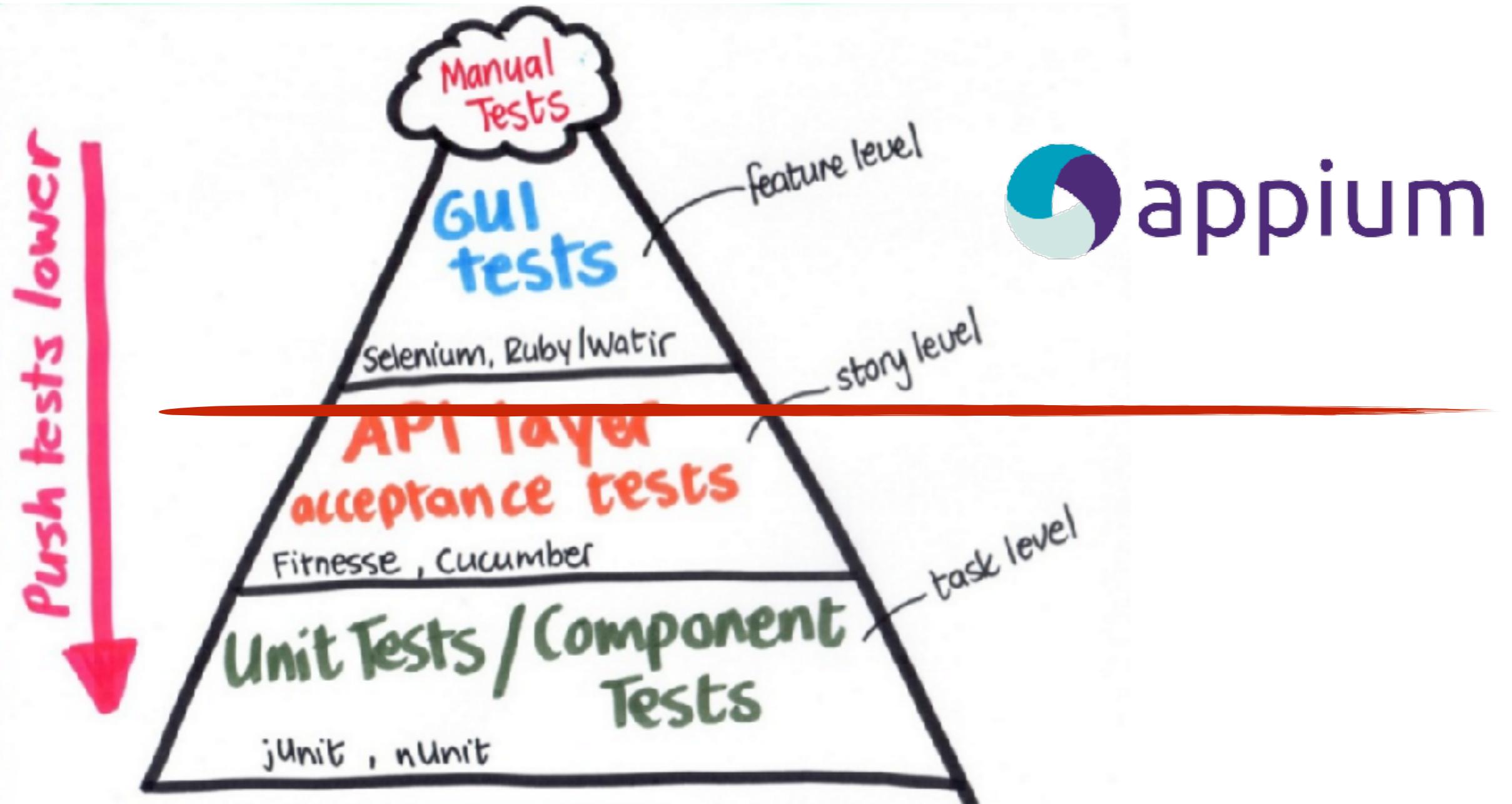
# Appium for Android app



<http://appium.io/>



บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่



# What is Appium ?

Test automation tool for mobile app

Open source

Support native, hybrid and mobile web

Run on actual device, emulator and simulator

Support android and iOS

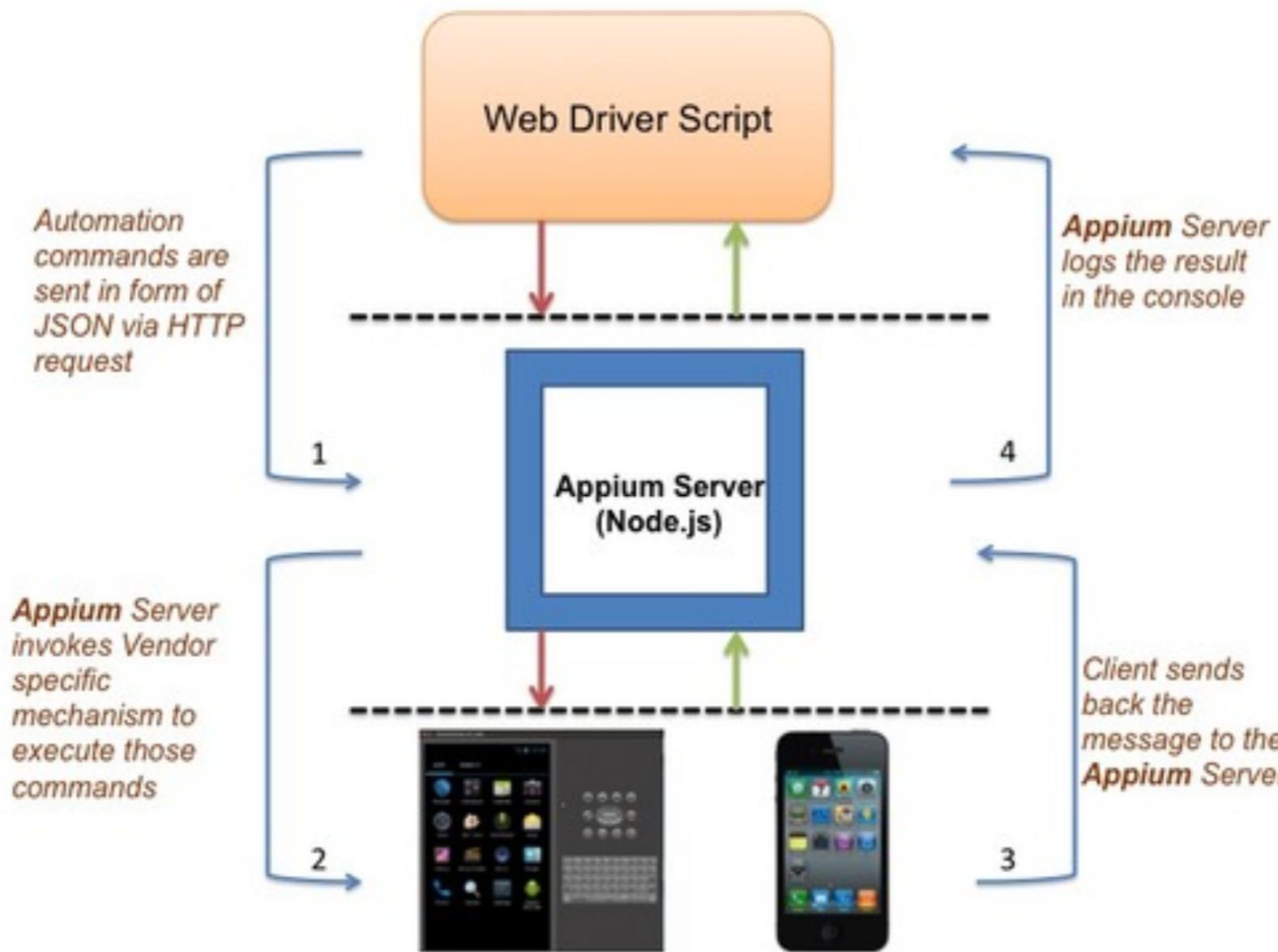


# Appium philosophy

You should be able to reuse code  
between android and iOS



# Appium architecture



<https://nishantverma.gitbooks.io/appium-for-android>



# Install appium server

via app

via NPM (Node package manager)

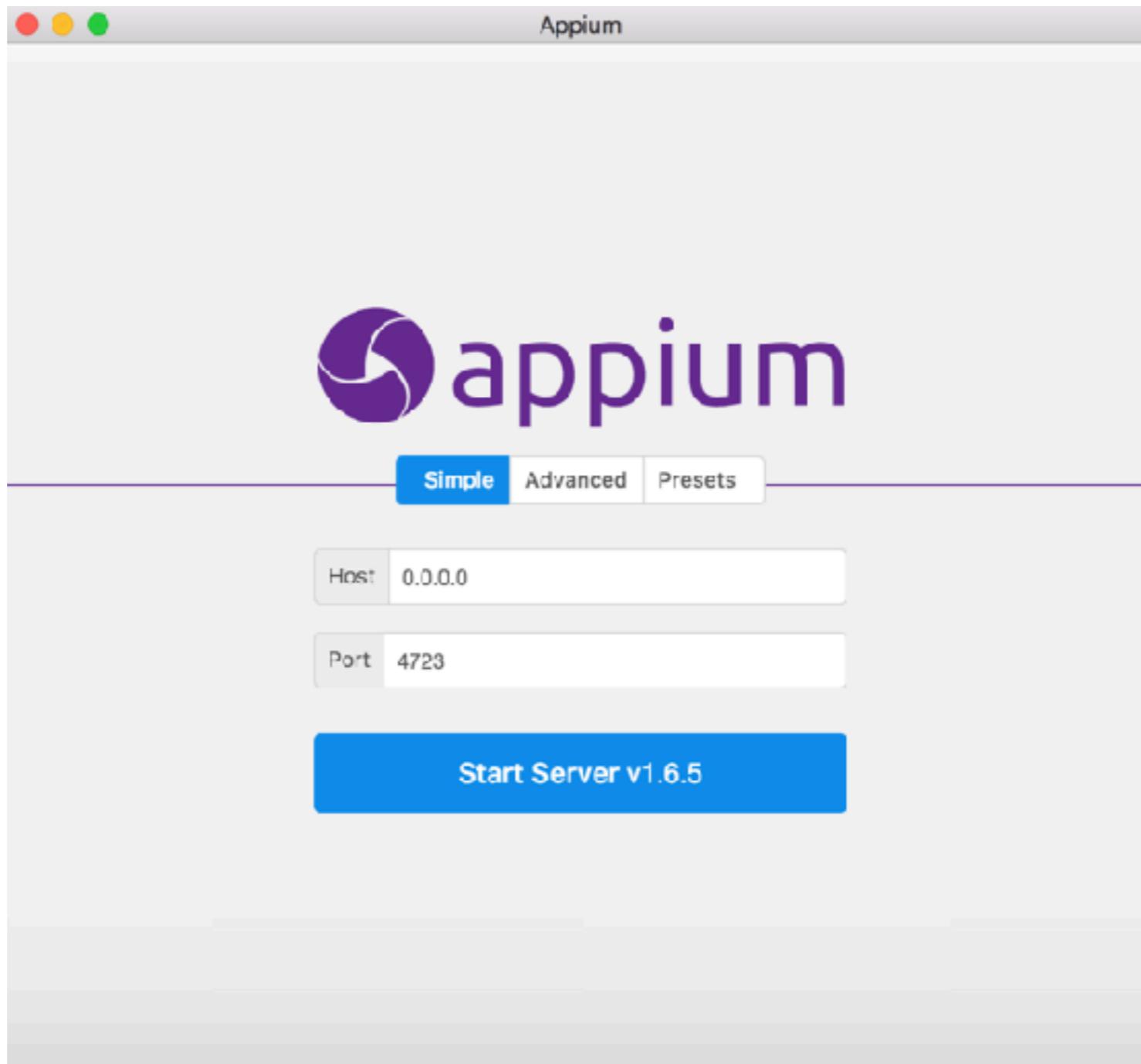


# Install via App

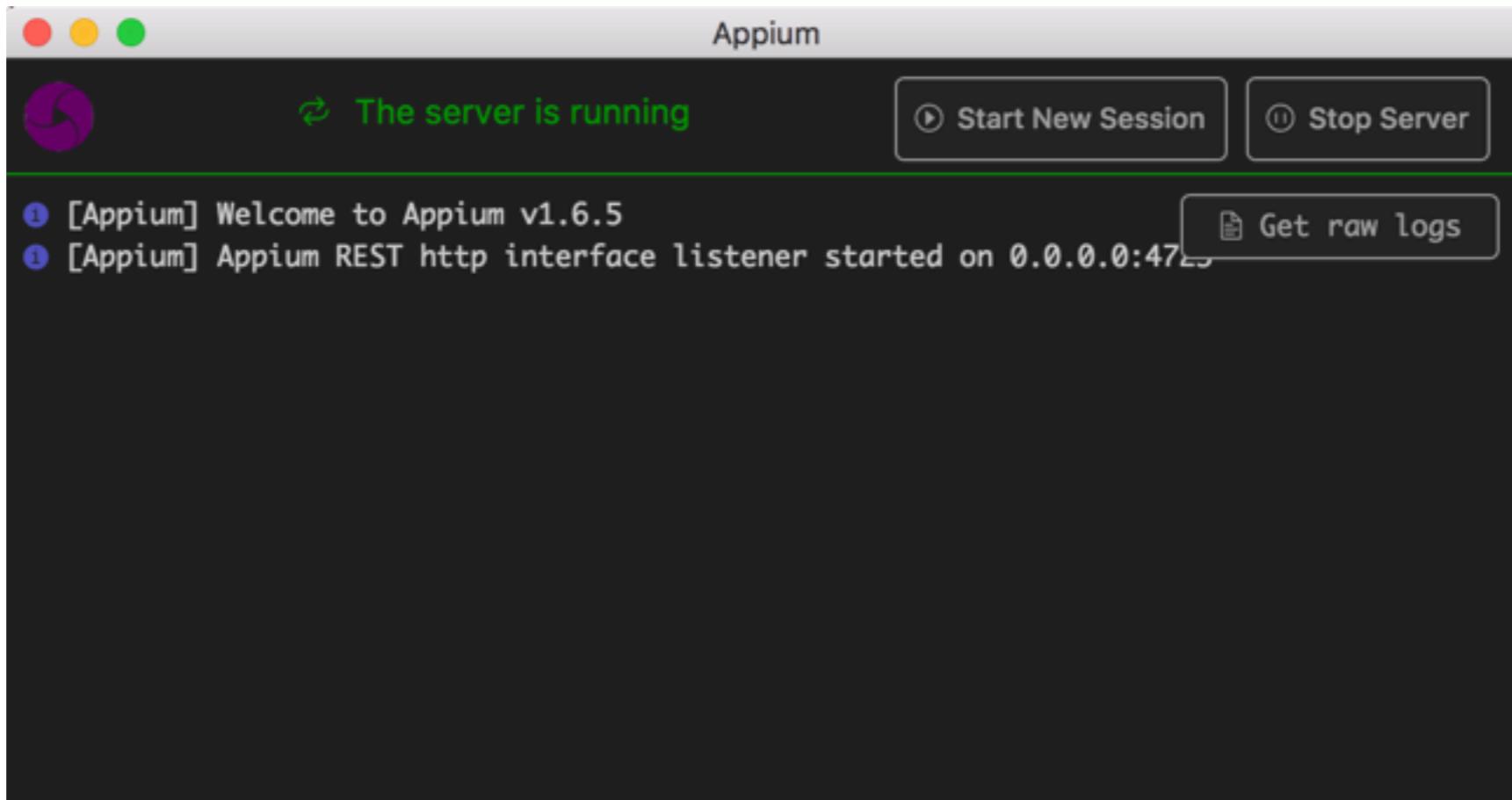
<http://appium.io/>



# Install via App



# Start server



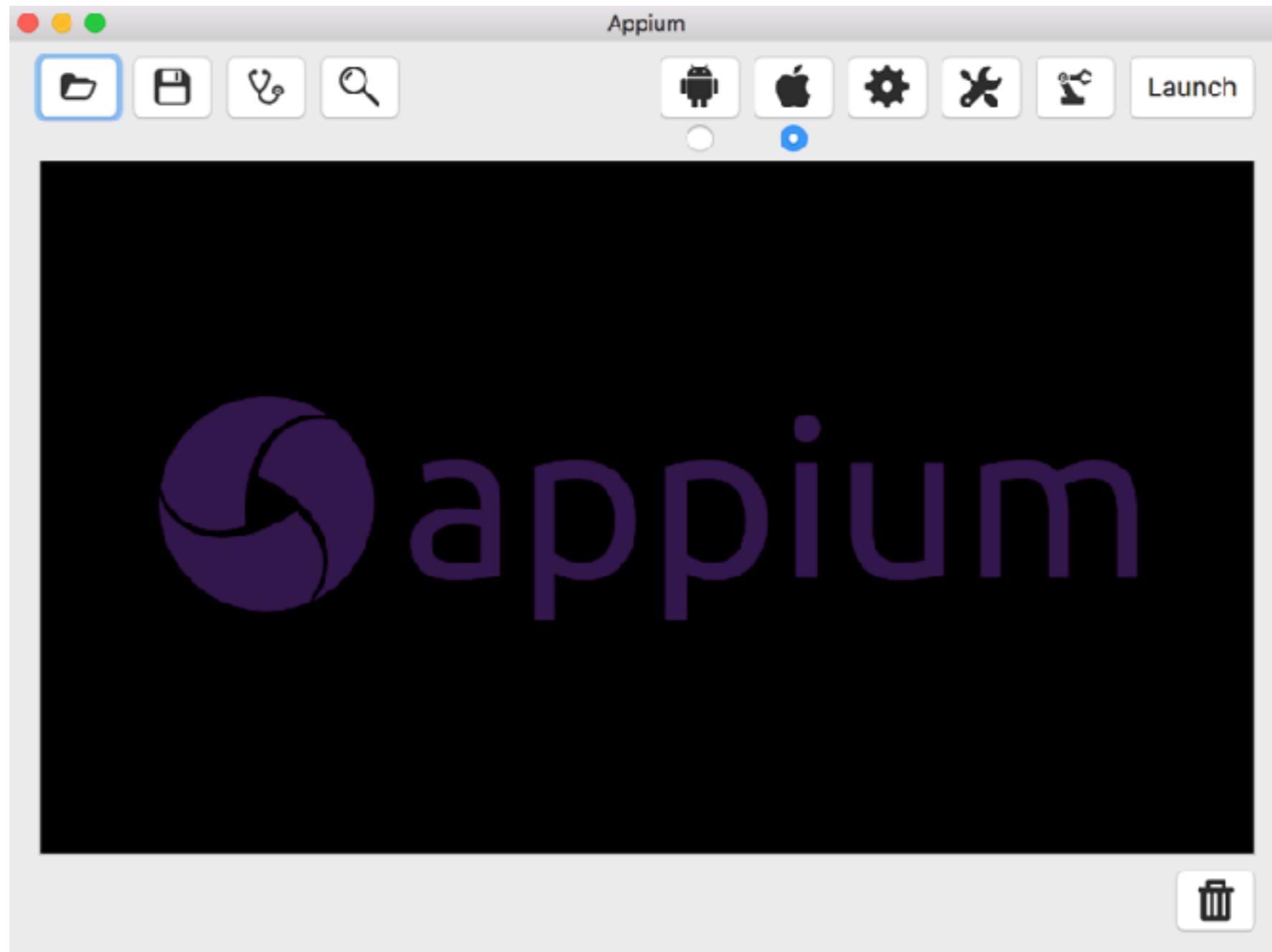
# Install via NPM

\$npm install -g appium

\$appium



# Install appium GUI



<https://bitbucket.org/appium/appium.app/downloads/appium.dmg>

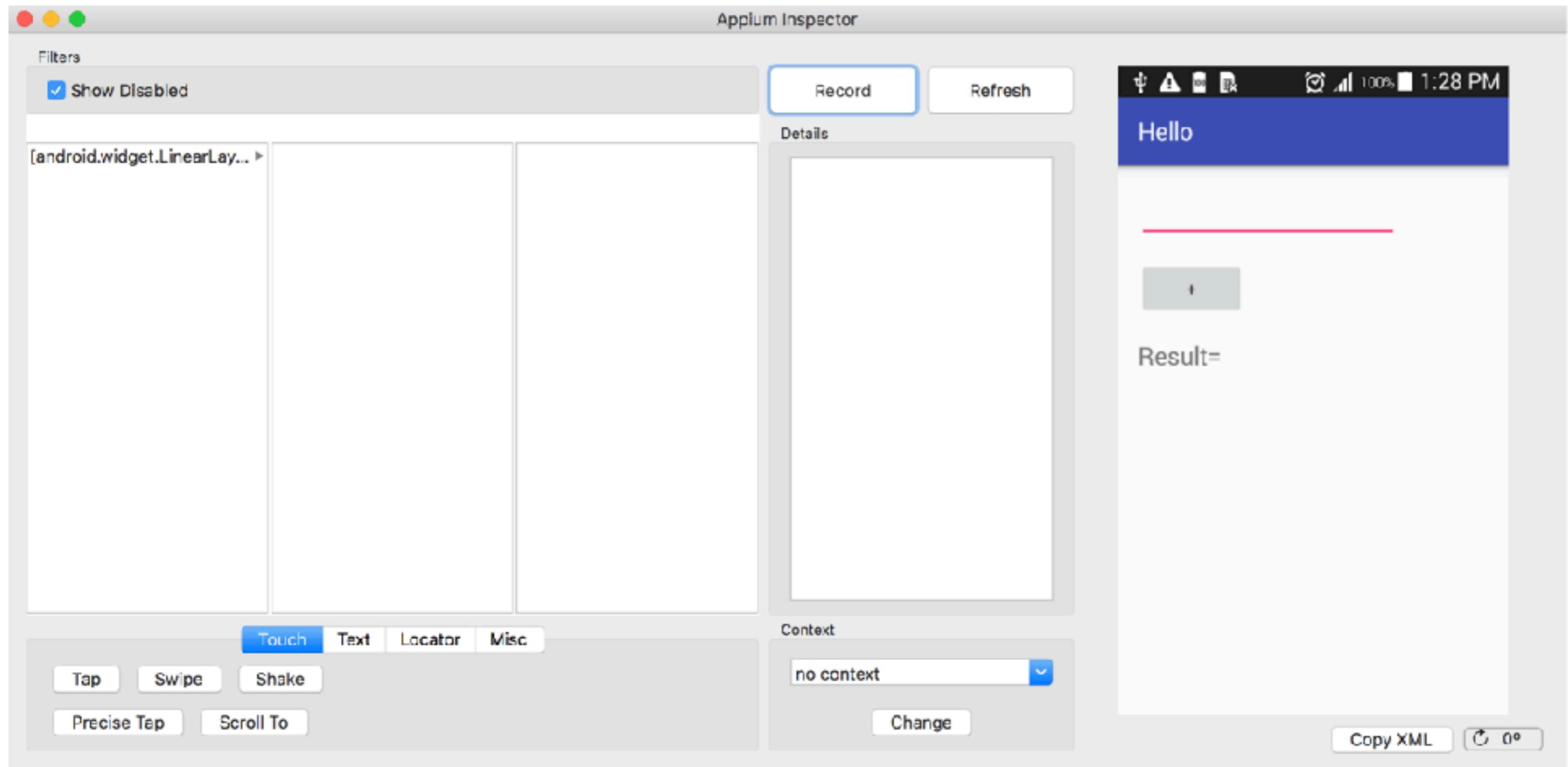


บริษัท สยามชนาญกิจ จำกัด และเพื่อนพ้องน้องพี่

# Setting for android



# Appium Inspector



# List of android devices

\$./adb devices

