twitch.tv/dancojocar
youtube.com/dancojocar

Lecture #14
Testing Frameworks & Exam Discussions

Mobile Applications
Fall 2023

Types of Testing Software

Smoke testing
System Integration Testing
Usability testing
Unit testing
User Acceptance testing
System Testing
Performance Testing
All Pairs testing
Security Testing
Regression Testing
Load Testing
Beta Testing
Ad-hoc testing
Soak Testing
Sanity Testing
Retesting
Black Box testing
Acceptance Testing
Risk-based Testing
Stress Testing
Backward Compatibility Testing
Glass box Testing
Localization Testing
Accessibility Testing
Static Testing
Boundary Value Testing
Agile Testing
Stability Testing
Negative Testing
API Testing
Big Bang Integration testing
Automated testing
Volume testing
Non-functional testing
Scalability Testing
Bottom up Integration testing
Pair Testing
Branch Testing
Gorilla Testing
Vulnerability Testing
Keyword-driven Testing
White box Testing
Browser compatibility Testing
Component Testing
Dynamic Testing
End-to-end Testing
Compatibility testing
Condition Coverage Testing
Exploratory Testing
Internationalization Testing
Happy path testing
Decision Coverage Testing
Integration Testing
Functional Testing
Fuzz Testing
Interface Testing
GUI (Graphical User Interface) testing

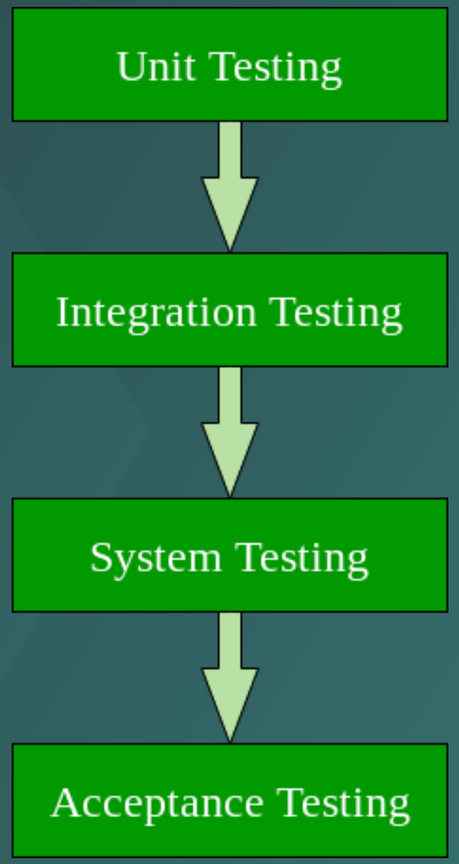
https://www.testingexcellence.com/types-of-software-testing-complete-list/

Types of Testing Software

Software Testing can be broadly classified into two types:

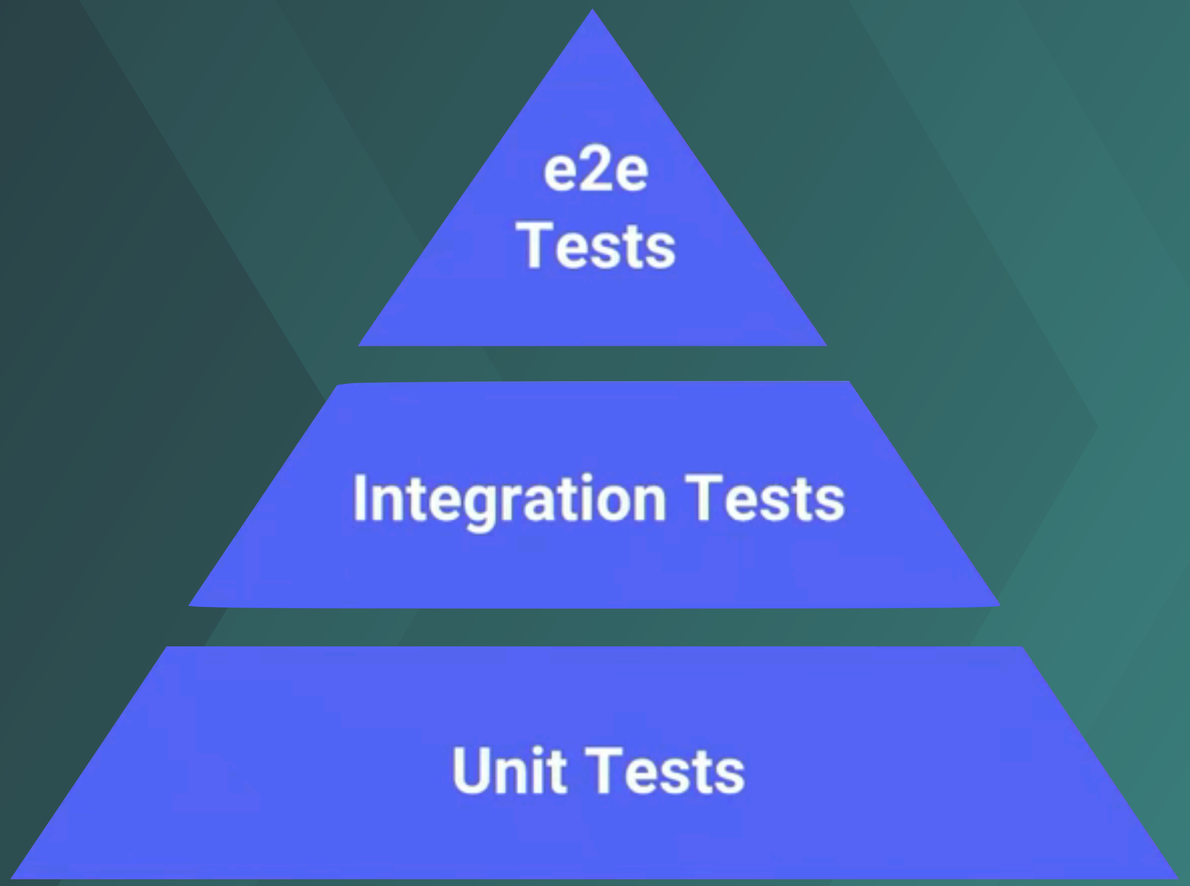
- Manual Testing
- Automation Testing

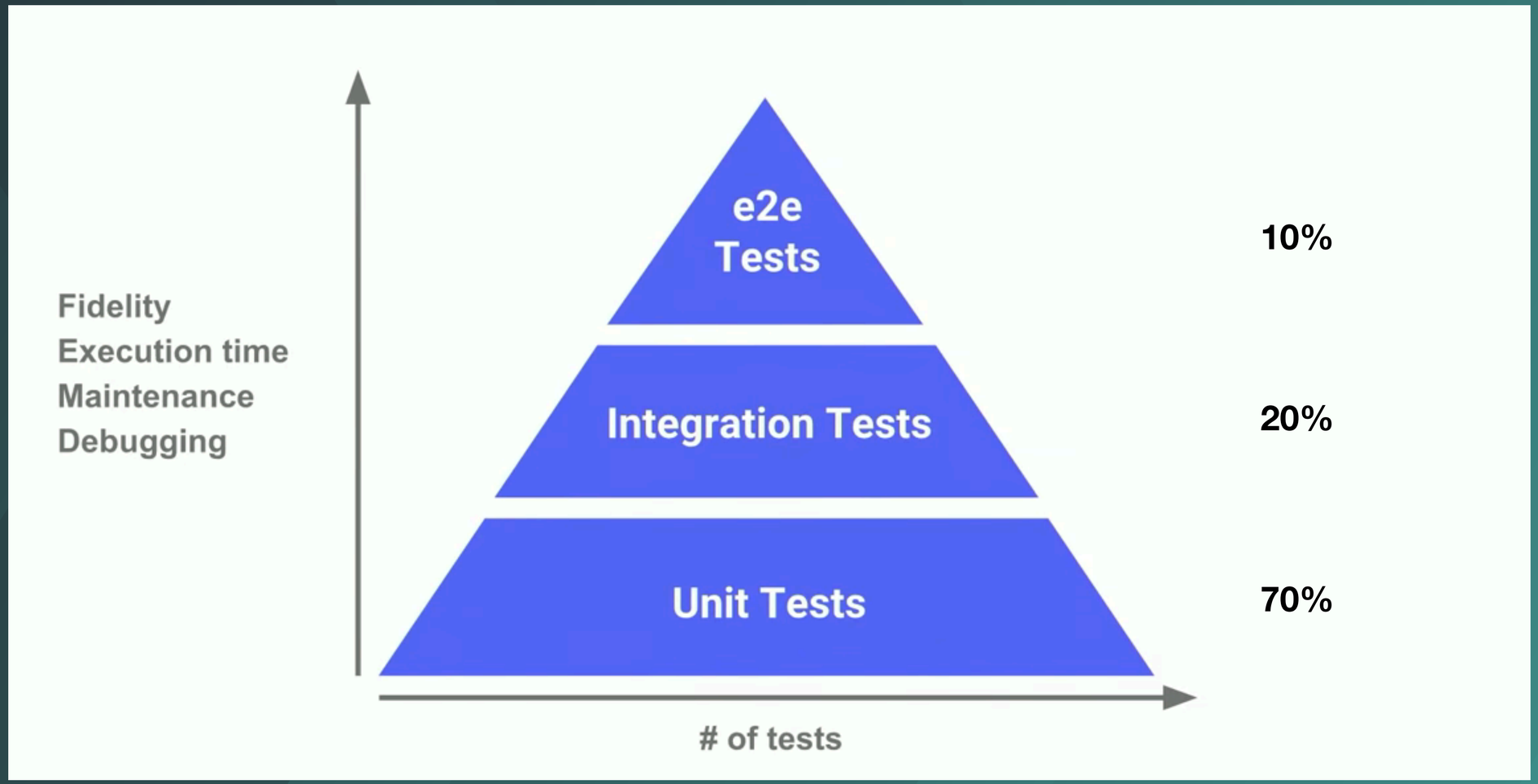
Automation Testing Levels

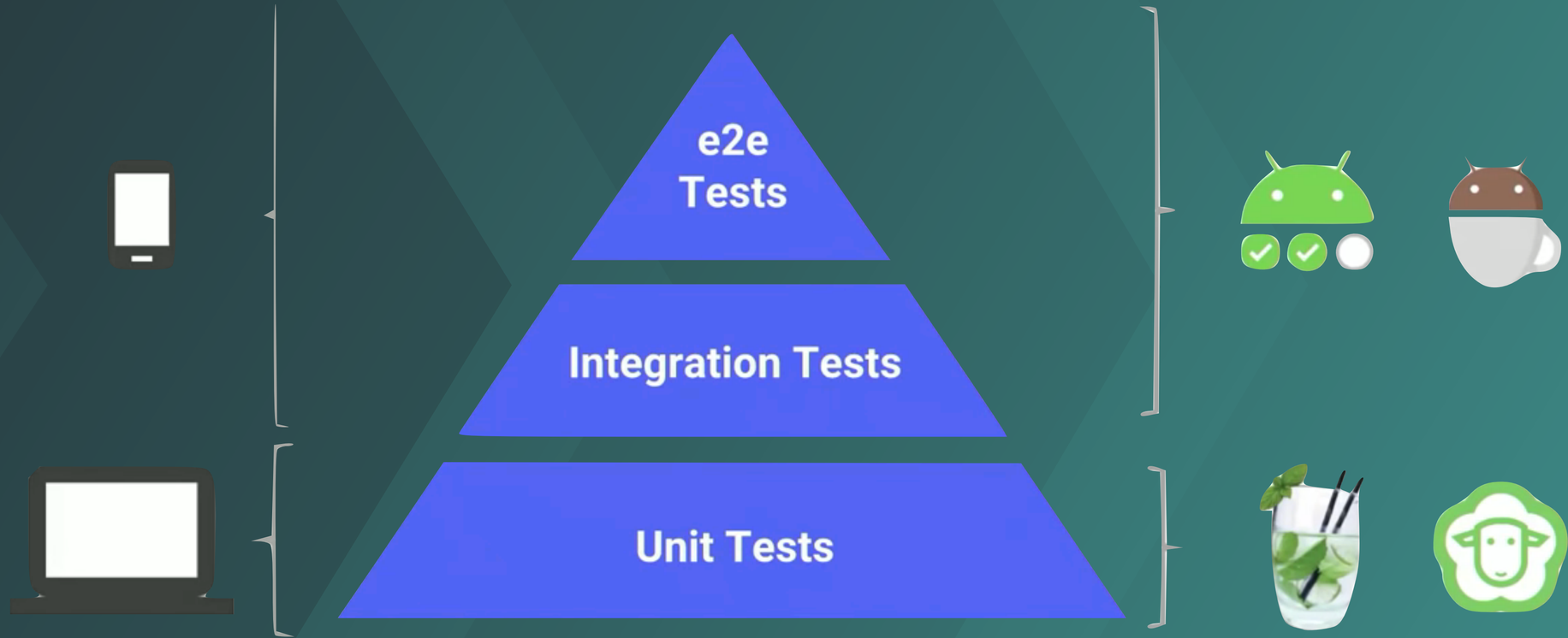


Advantages

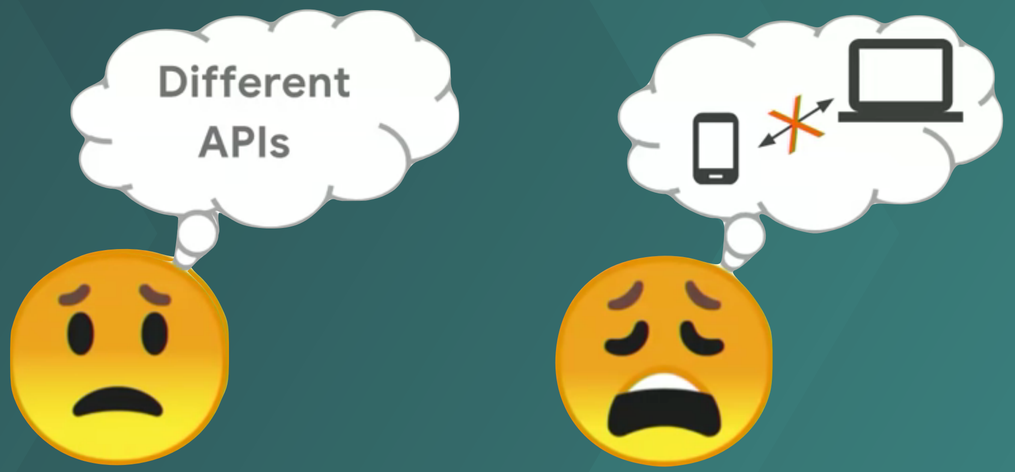
- Rapid feedback.
- Early failure detection.
- Safer code refactoring.
- Stable development velocity.


e2e
Tests
Integration Tests
Unit Tests


Fidelity
Execution time
Maintenance
Debugging
e2e
Tests
Integration Tests
Unit Tests
10%
20%
70%
# of tests


e2e
Tests
Integration Tests
Unit Tests

Test writing crisis



Test Structure

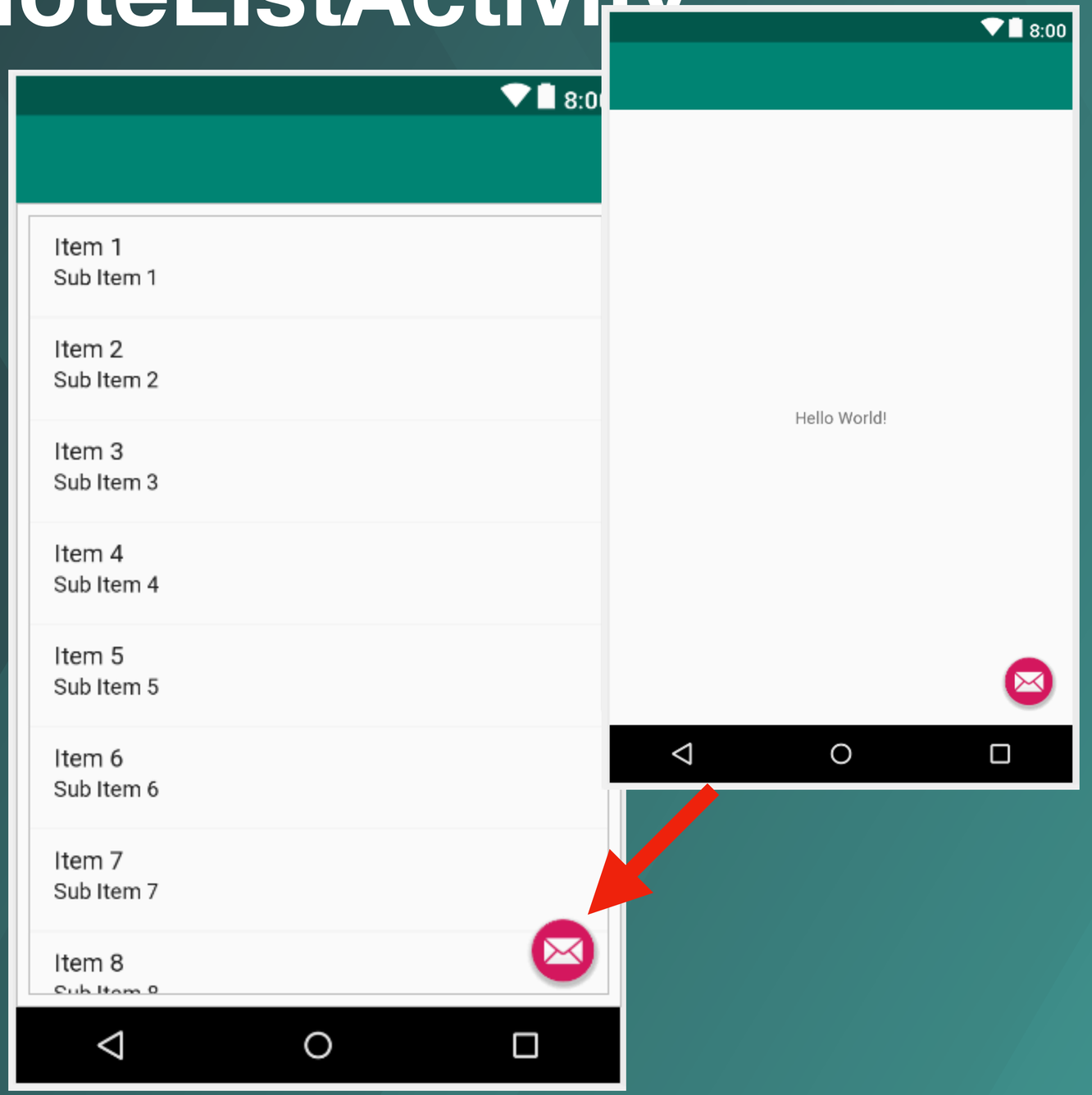
```
class WellStructuresTest{
  //..
  fun givenCondition_whenAction_thenShoulDo(){
    // GIVEN - Setup condition

    // WHEN - The tested action

    // THEN - An assertion to validate the action
  }
  //...
}
```

- Focus on specific behavior.
- Test behaviors independently.
- LESS is MORE. Keep tests understandable and in isolation.

NoteListActivity



Mockito

```
@RunWith(MockitoJUnitRunner::class)
class MockitoTest {
  @Spy var spyActivity = NoteListActivity()
  @Captor lateinit var intentCaptor: ArgumentCaptor<Intent>
  @Captor lateinit var clickCaptor:
            ArgumentCaptor<NoteListActivity.ClickHandler>

  fun testTitle() {
    `when`(spyActivity.findViewById(R.id.title))
      .thenReturn(mock<TextView>())

    clickCaptor.value.click()

    verify(spyActivity).startActivity(intentCaptor.capture())
  }
}
```

https://site.mockito.org

Robolectric

```
@RunWith(RobolecticTestRunner::class)
class RobolecticTest {

  @Test
  fun testTitle() {
    val activity =
      Roboelectic.setupActivity(NoteListActivity::class.java)

    ShadowView.clickOn(activity.findViewById(R.id.title))

    assertEquals(
      ShadowApplication.getInstance()
          .peekNextStartedActivity().action,
      "android.intend.action.EDIT"
    )
  }
}
```

http://robolectric.org

Espresso

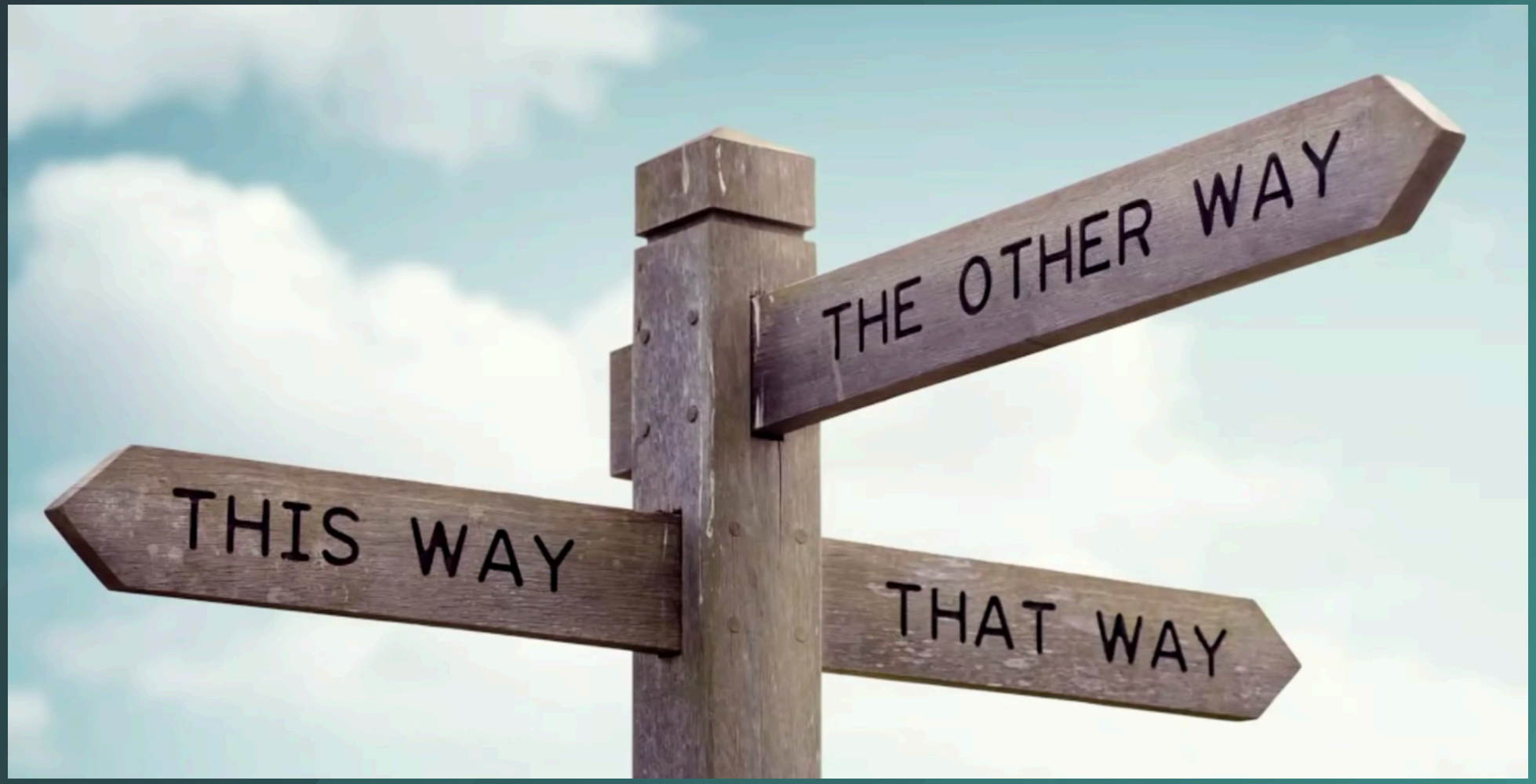
```
@RunWith(AndroidJUnit4::class)
class OnDeviceTest {

    @get:Rule
    val rule = ActivityTestRule(NoteListActivity::class.java)

    @Test
    fun testTitle() {
        onView(withId(R.id.fab)).perform(click())

        intended(hasAction(equalTo("android.intent.action.EDIT")))
    }
}
```

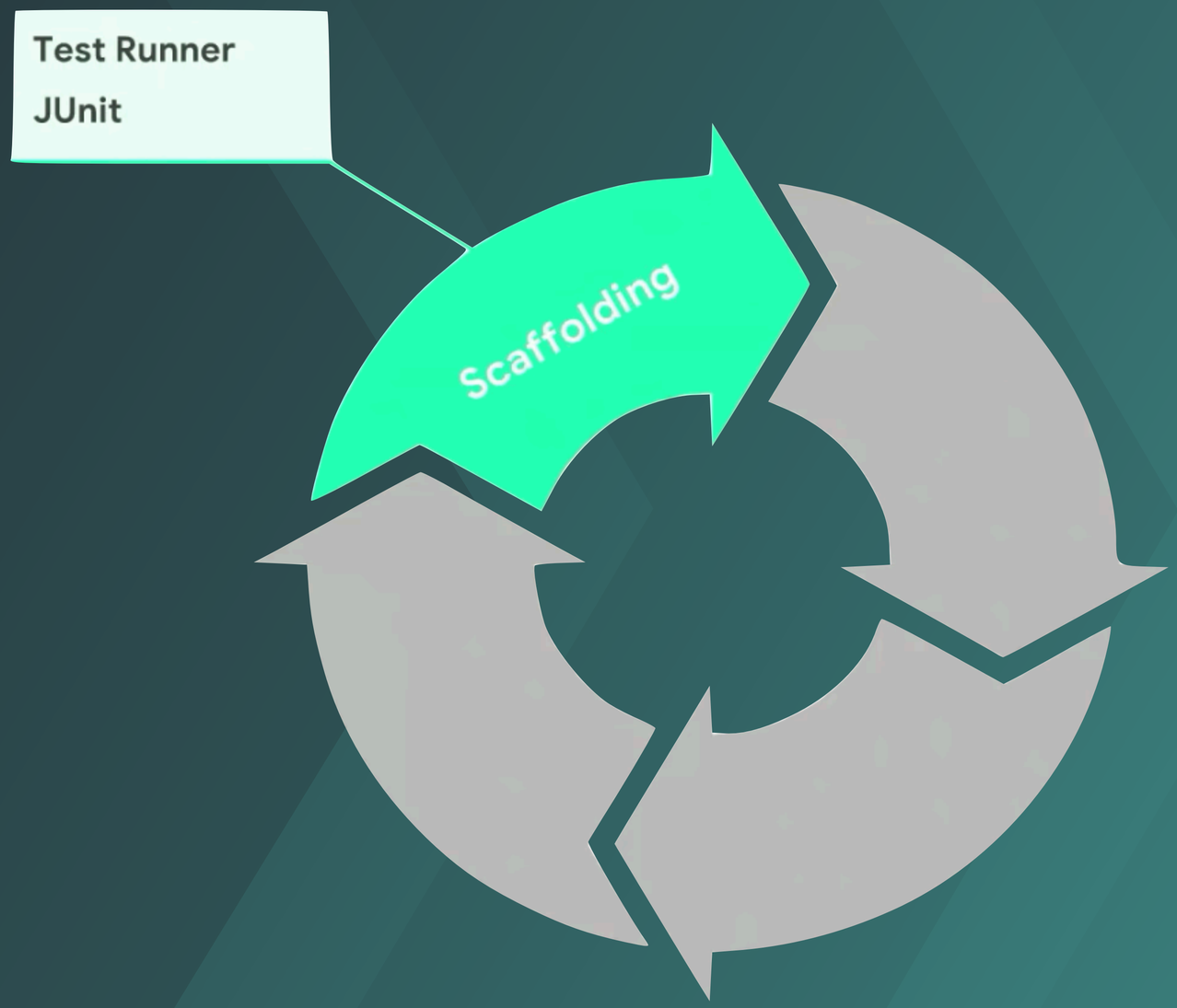
https://developer.android.com/training/testing/espresso


THE OTHER WAY
THIS WAY
THAT WAY

Android Test

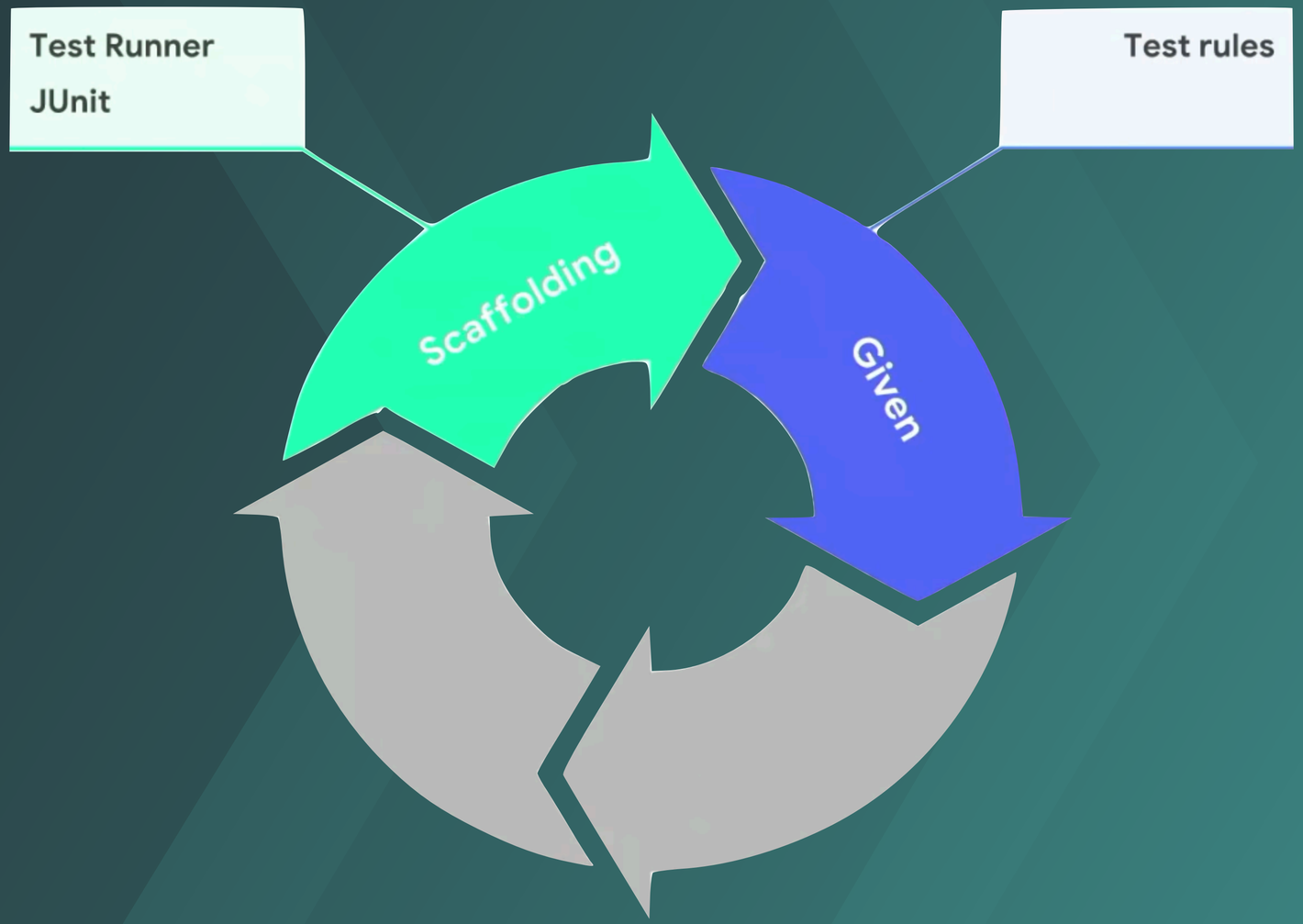
Part of Jetpack

- Includes existing libraries.
- New APIs and Kotlin.
- Available on/off device.
- Open source.


Test Runner
JUnit
Scaffolding

```
//SCAFFOLDING

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Before
  fun setup() {
    val context = InstrumentationRegistry.getTargetContext()
  }
}
```

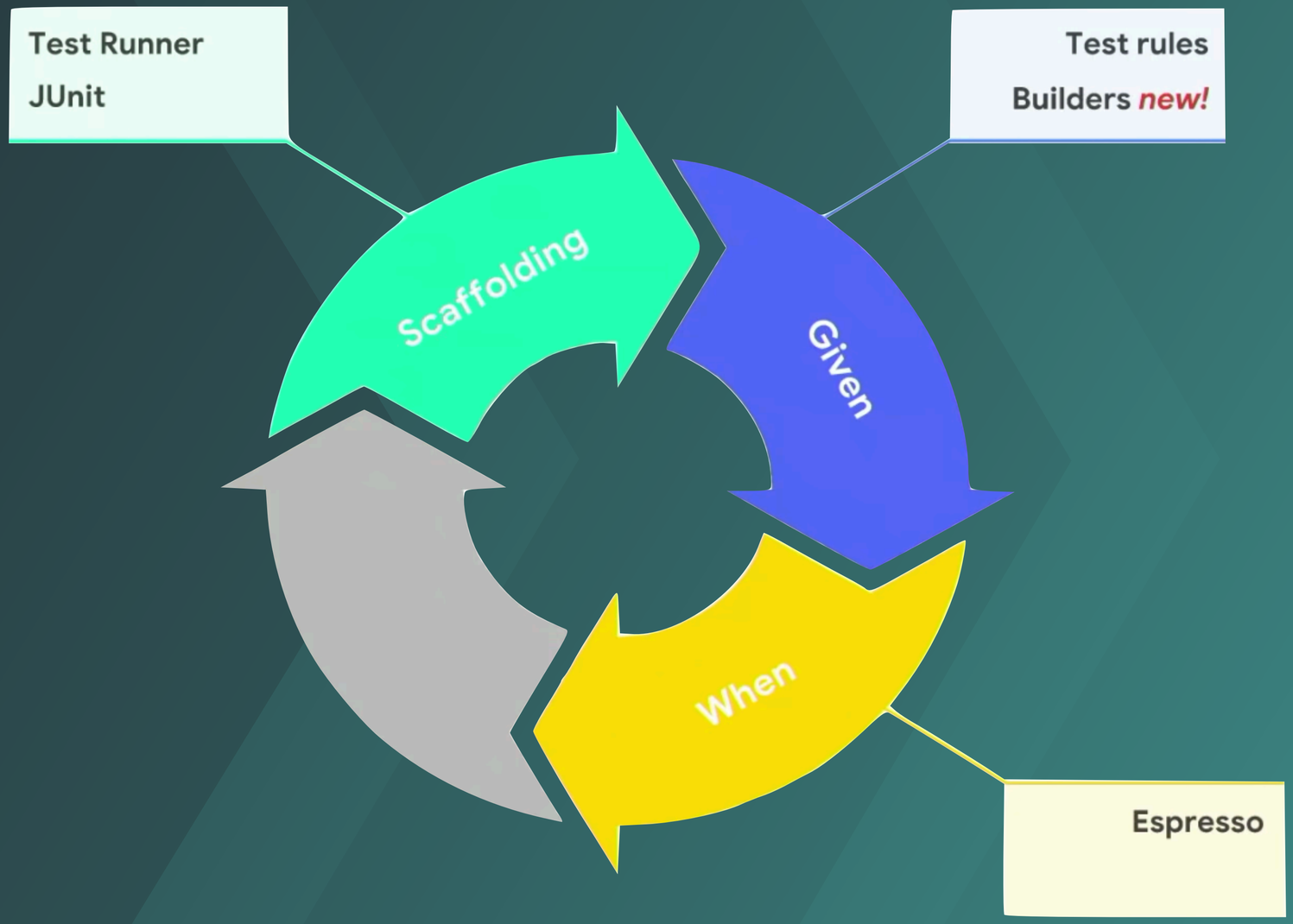

Test Runner
JUnit
Test rules
Scaffolding
Given

```
//GIVEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @get:Rule
  val rule = ActivityTestRule(NoteListActivity::class.java)
}
```

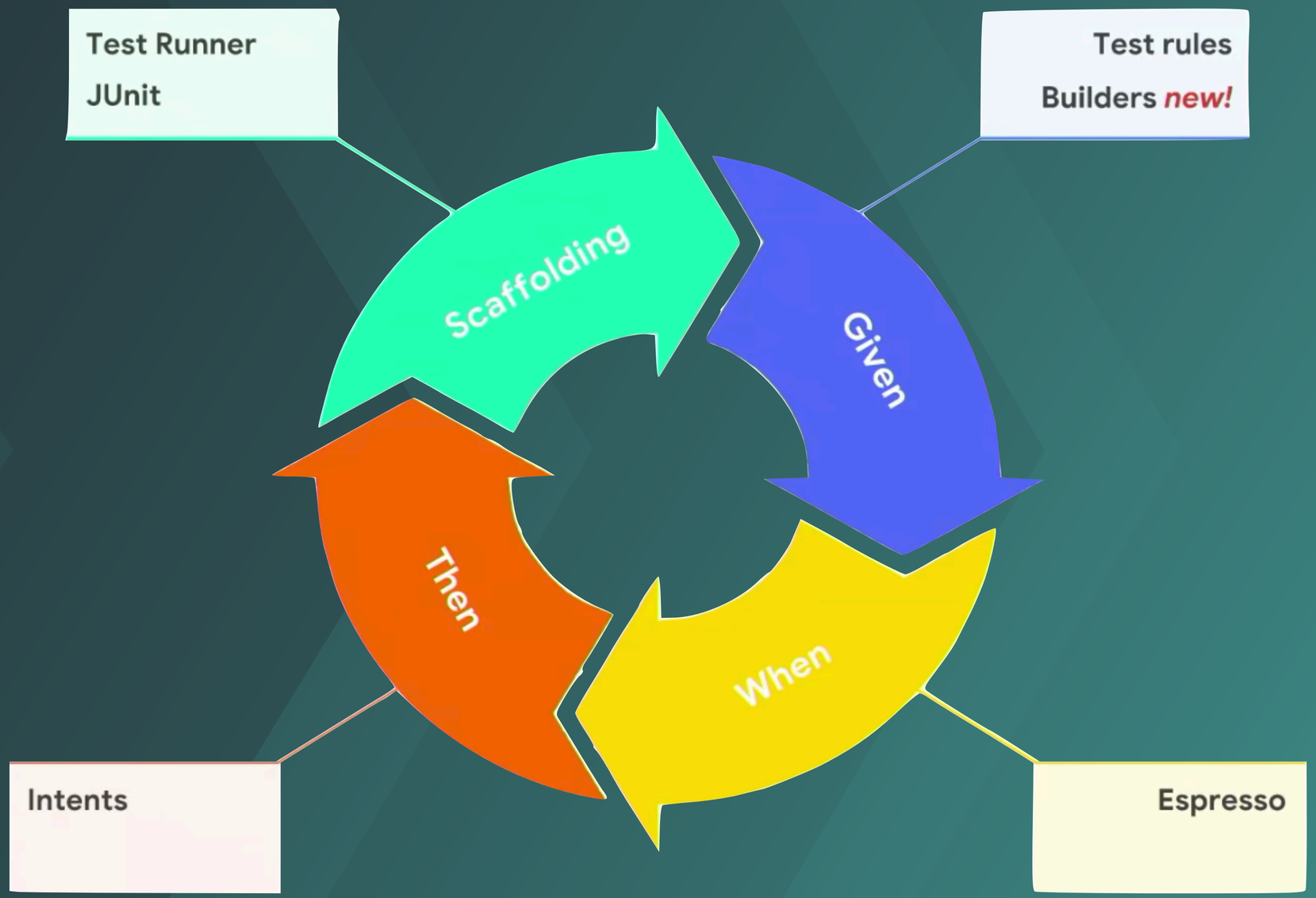

```
//GIVEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Test
  fun testMotionEvents() {
    val motionEvent =
            buildMotionEvent().setAction(MotionEvent.ACTION_DOWN)
  }
}
```


Test Runner
JUnit
Test rules
Builders new!
Scaffolding
Given
When
Espresso

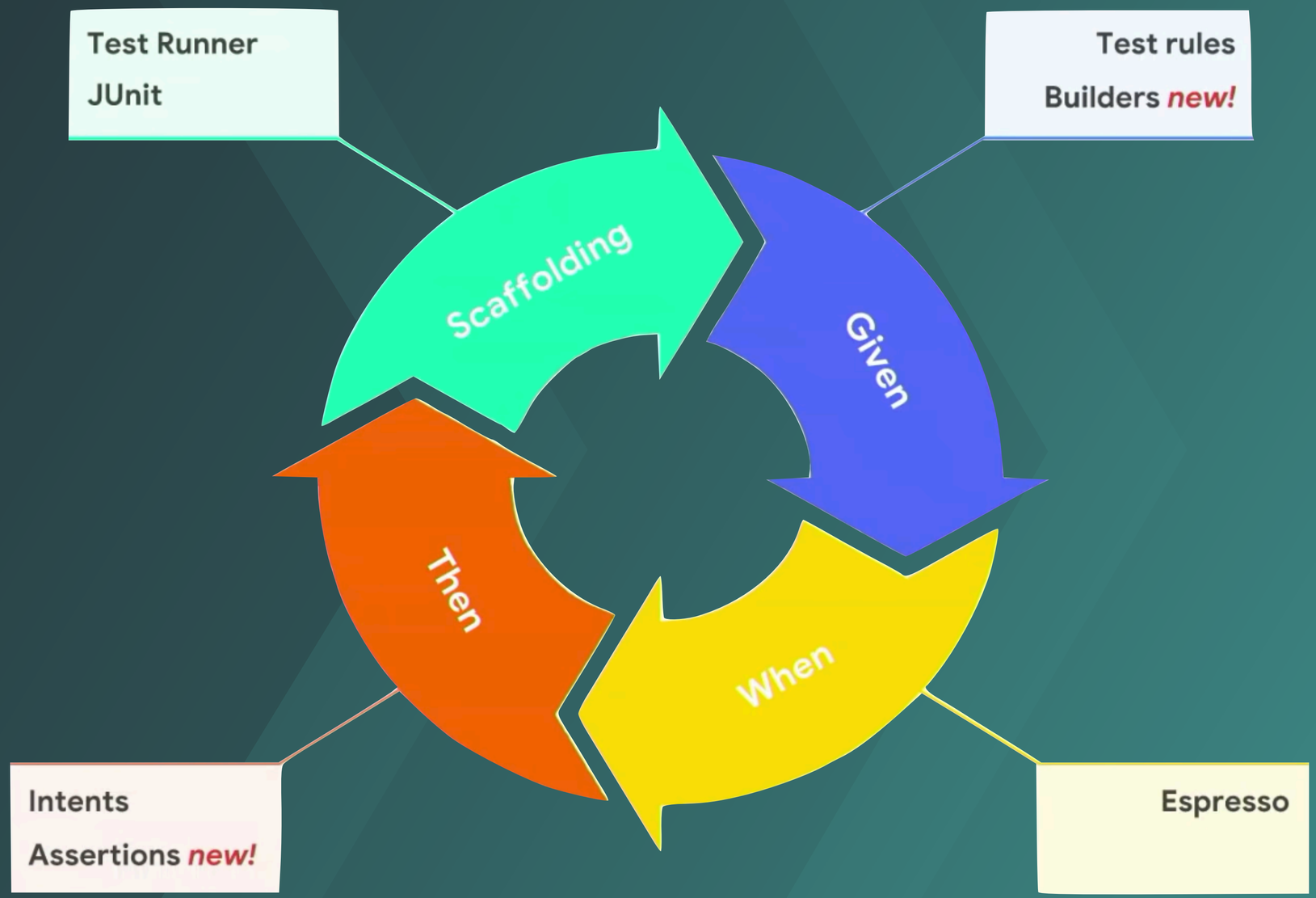
```
//WHEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Test
  fun testButtonClickSendsIntent() {
    onView(withId(R.id.fab)).perform(click())
  }
}
```


Test Runner
JUnit
Test rules
Builders new!
Scaffolding
Given
When
Then
Intents
Espresso

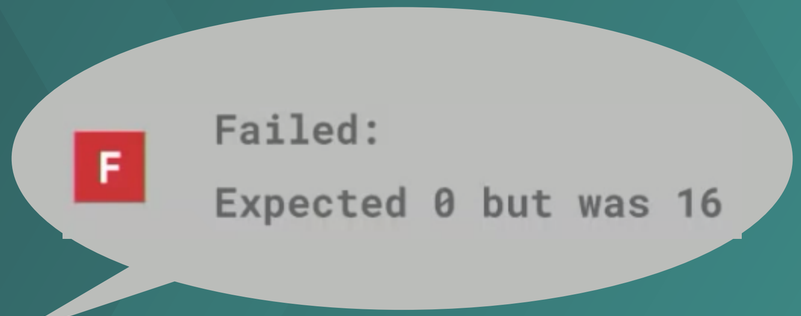
```
//THEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Test
  fun testButtonClickSendsIntent() {
    onView(withId(R.id.fab)).perform(click())
    intended(hasAction(equalTo("android.intent.action.EDIT")))
  }
}
```


Test Runner
JUnit
Test rules
Builders new!
Scaffolding
Given
When
Then
Intents
Assertions new!
Espresso

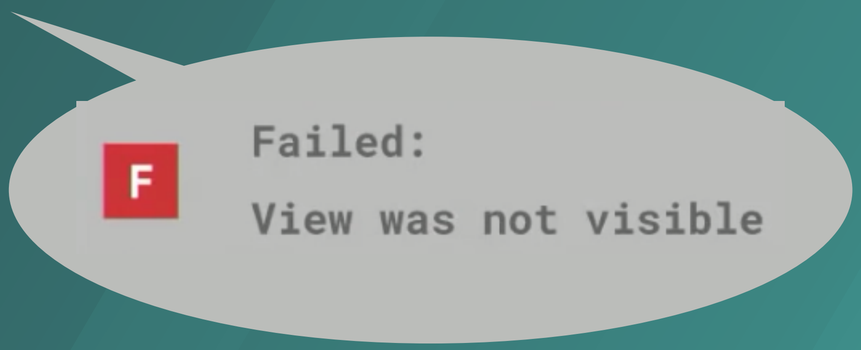
```
//THEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Test
  fun testVisibleView() {
    assertEquals(view.visibility, View.VISIBLE)
  }
}
```




```
//THEN

@RunWith(AndroidJUnit4::class)
class SimpleUnifiedTest {
  @Test
  fun testVisibleView() {
    assertThat(view).isVisible()
  }
}
```



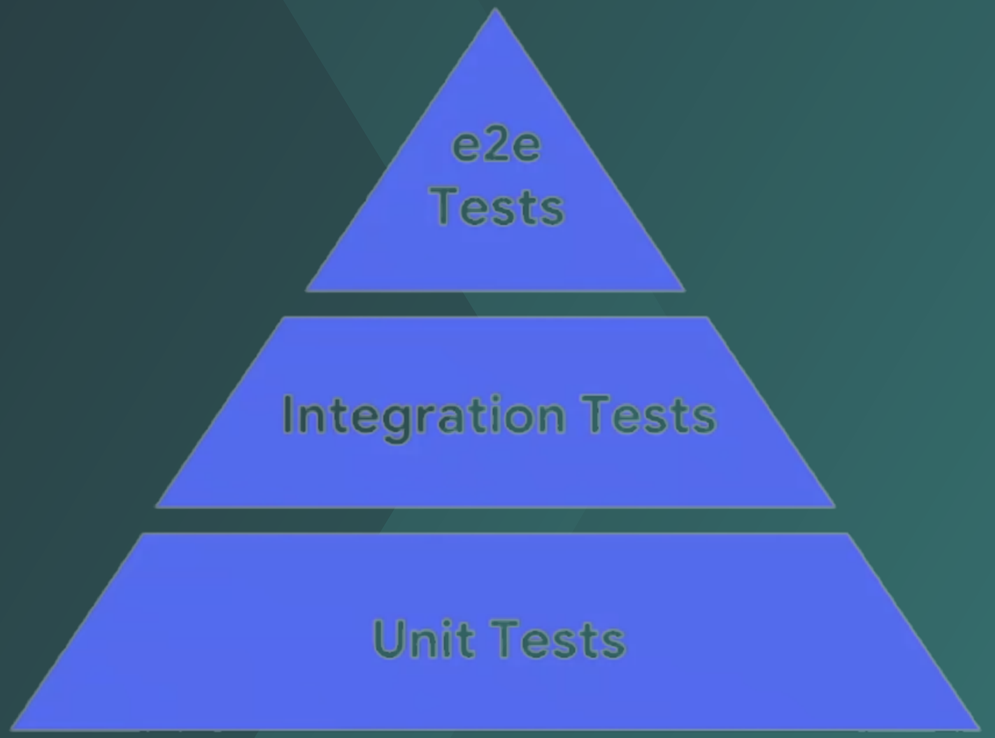
Canonical APIs

Kotlin

Reduces Boilerplate

Cross Environment

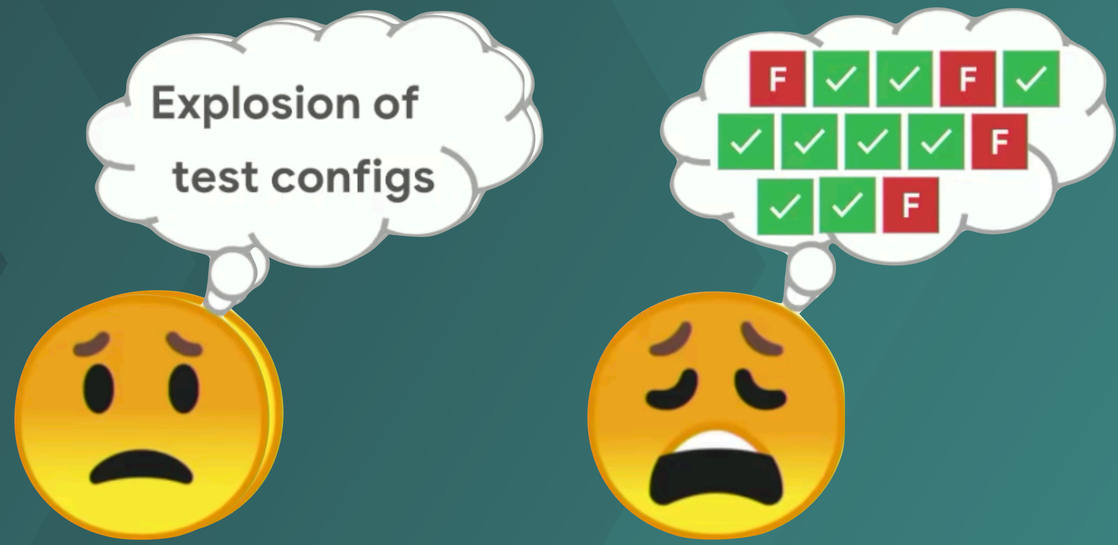
https://developer.android.com/training/testing



Android Test

Part of Jetpack

Test execution crisis



Project Nitrogen

A single entry point for tests

DO WHAT YOU CAN'T

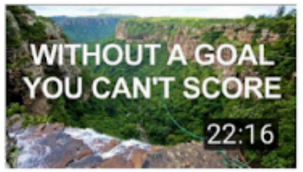
6.1M views

434K 7K Share Save Add to

CaseyNeistat

7M subscribers

SUBSCRIBE



DO MORE

CaseyNeistat

2.1M views

THE $21,000 FIRST CLASS AIRPLANE SEAT

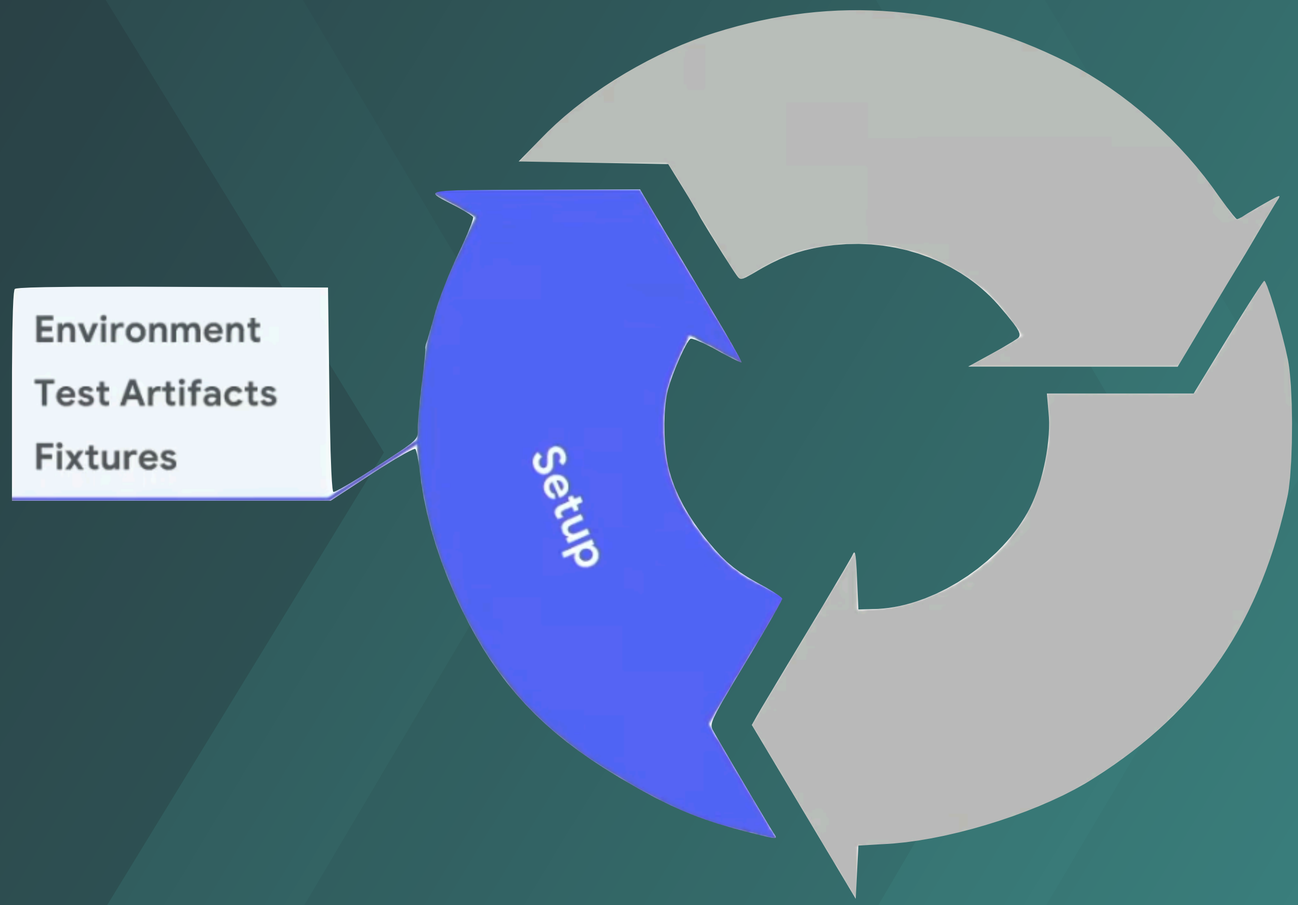
CaseyNeistat

42M views

FOLLOW YOUR DREAMS

CaseyNeistat

1.9M views


Environment
Test Artifacts
Fixtures
Setup


NITROGEN
Install APKs
App Under Test
Test Apk

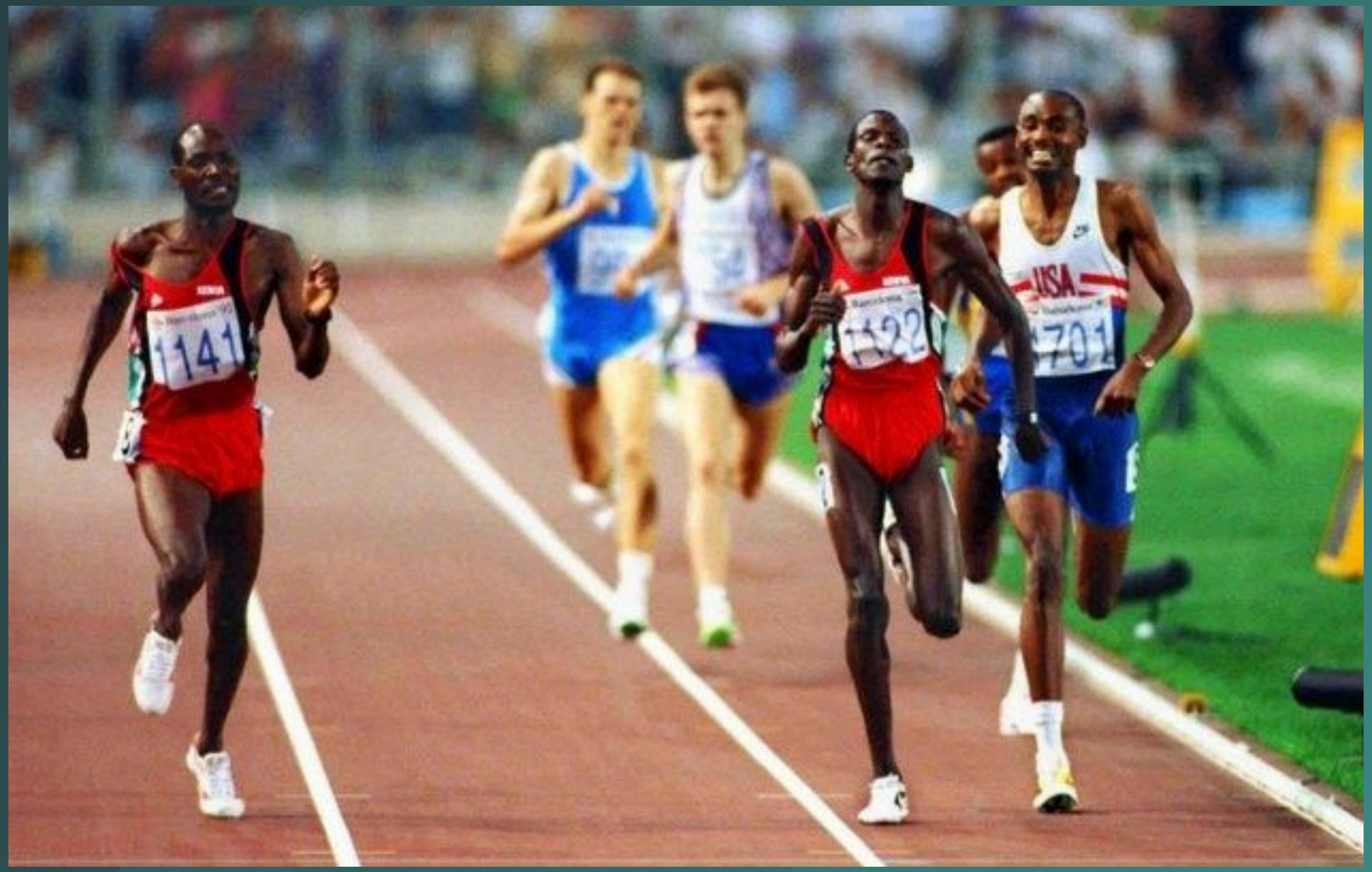

NITROGEN
Install APKs
App Under Test
Test Apk
Data Staging
Test data
Fixture Scripts


NITROGEN
Install APKs
App Under Test
Test Apk
Data Staging
Test data
Fixture Scripts
Test Execution
Android Test Orchestrator
Test Services

Running in the Real World

Sprint Running

Midd-Range Distance Running



Long Distance Running

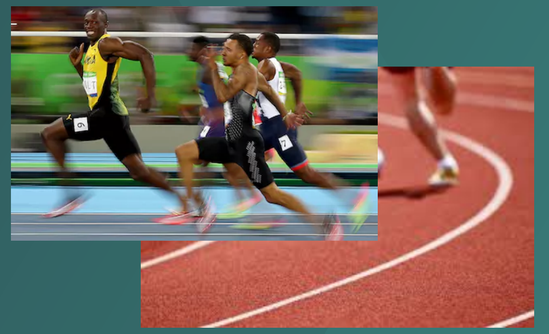
Track Running

Road Running

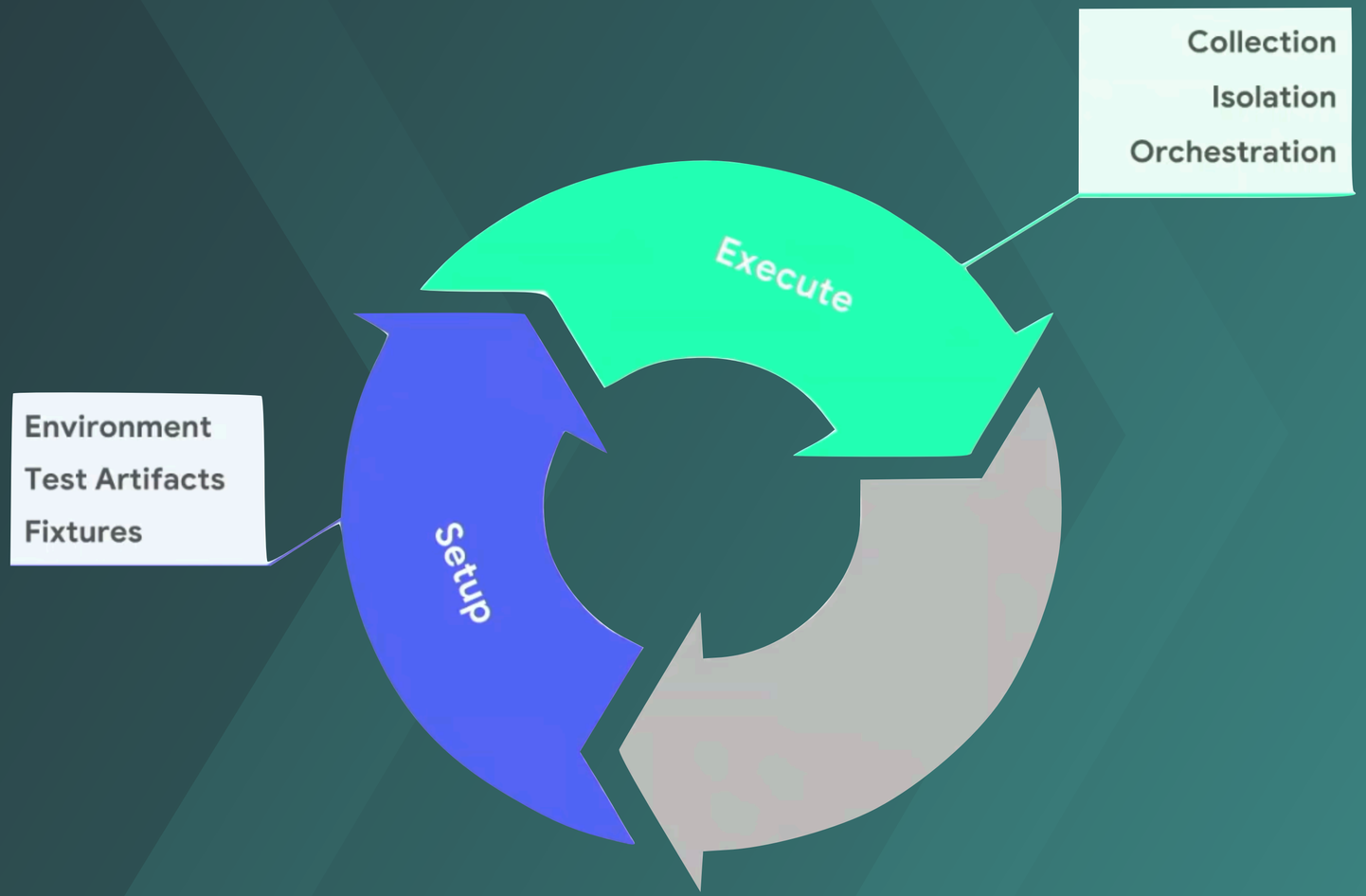
Cross Country Running

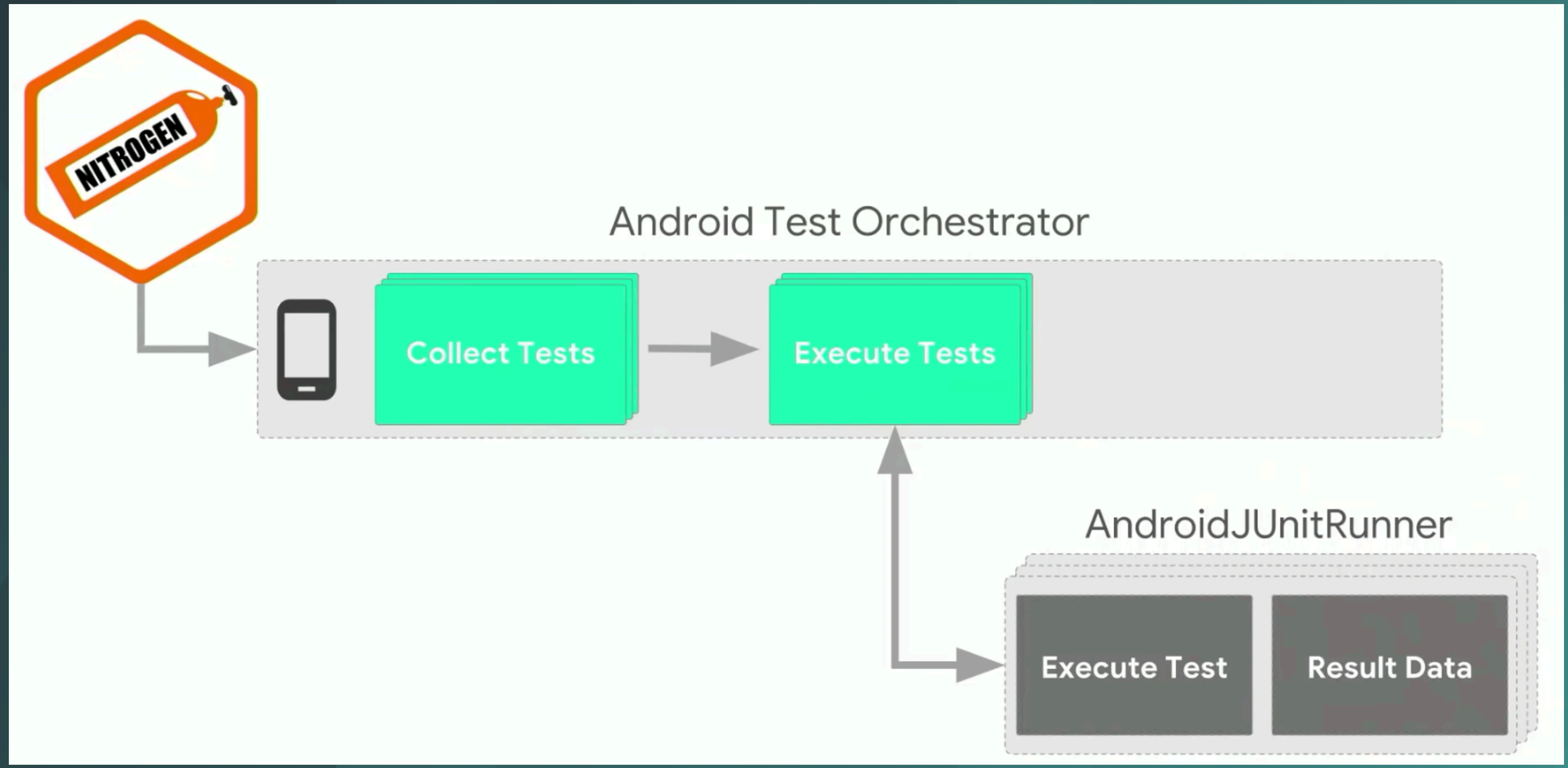
Mountain Running

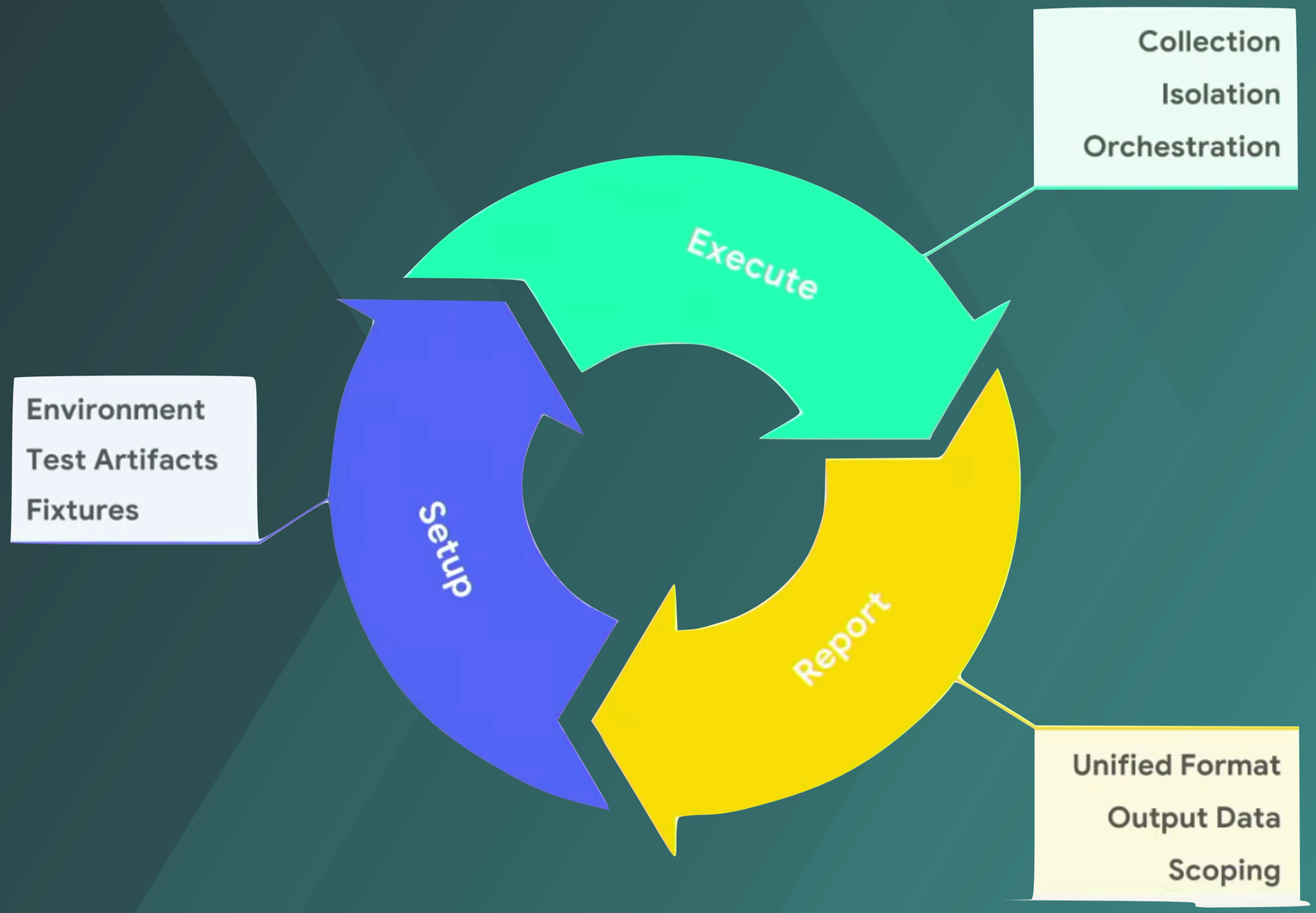
jUnit Test

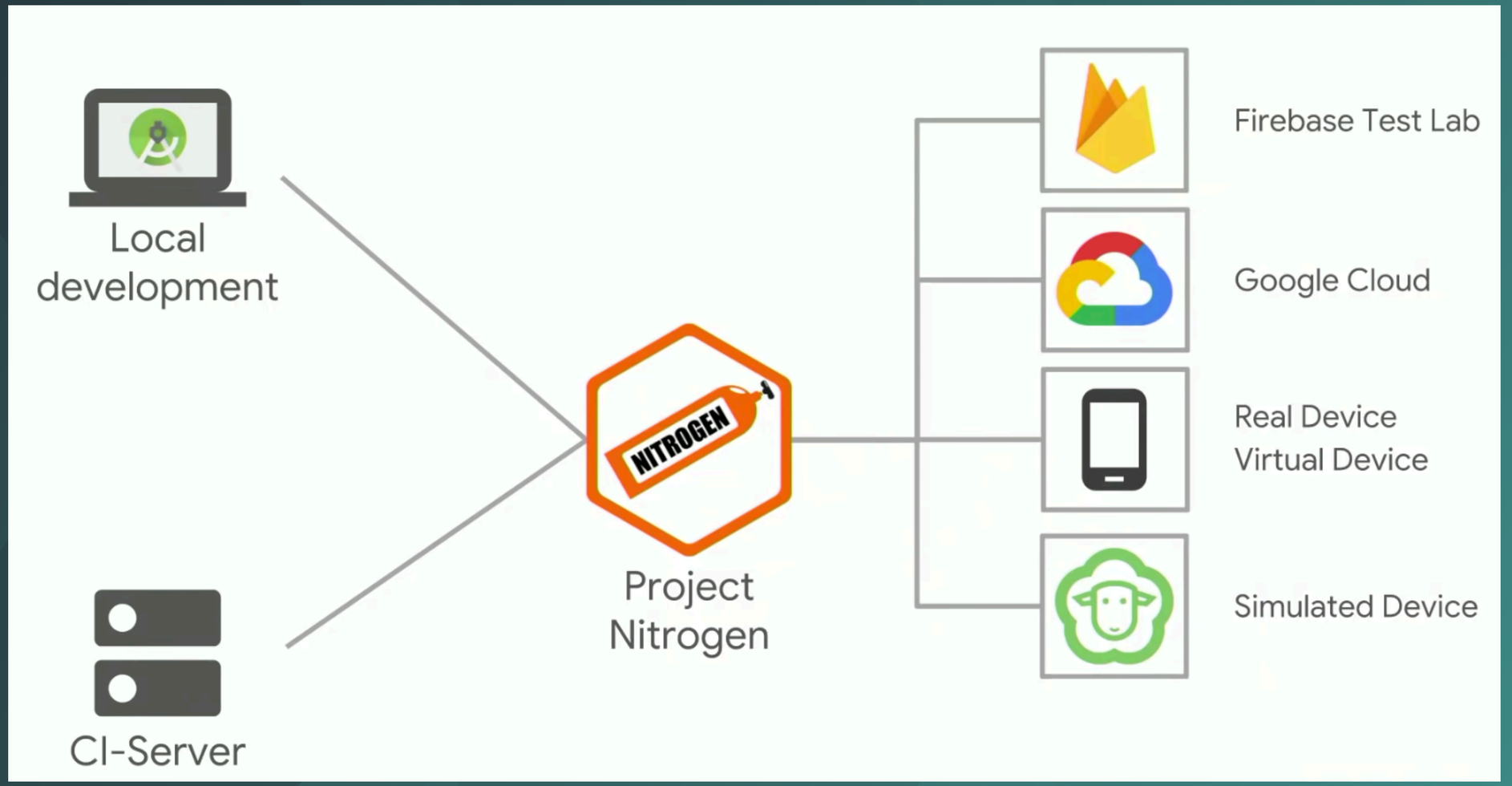


TestSuite

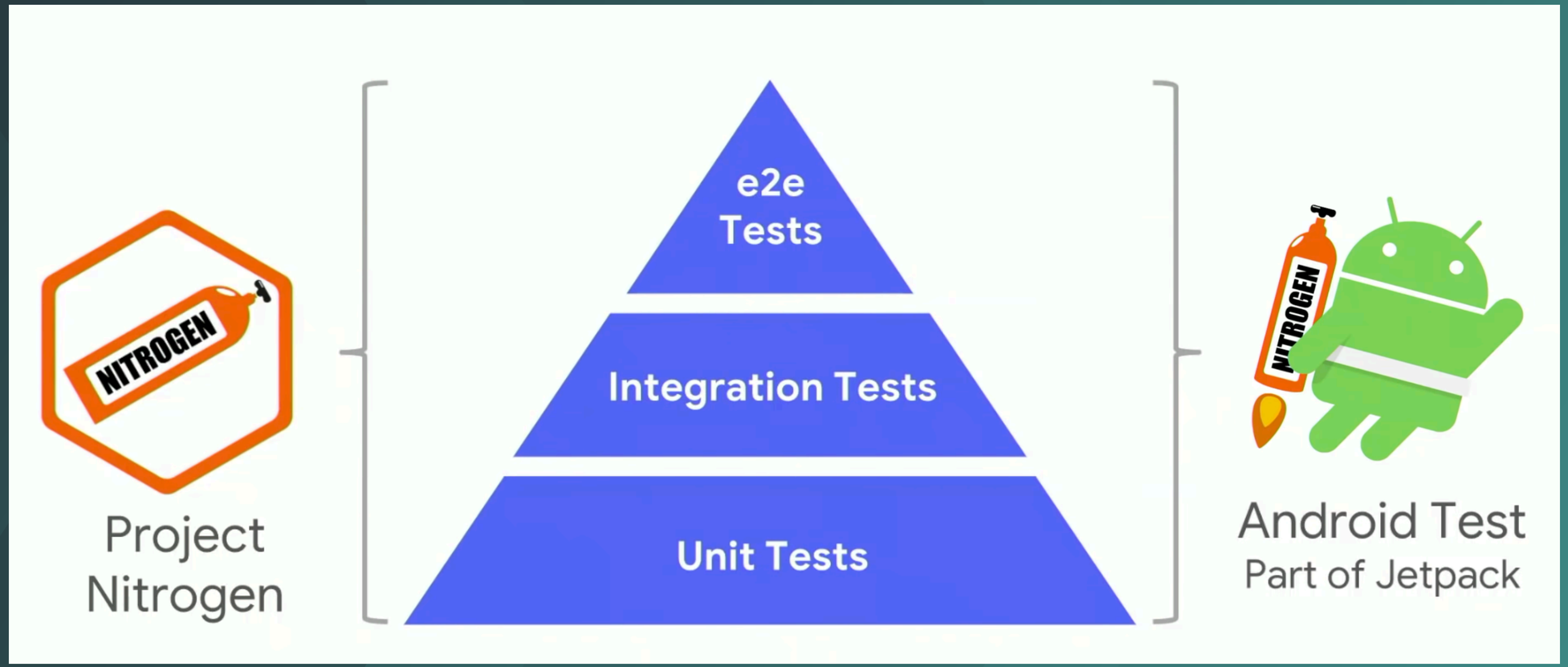

Collection
Isolation
Orchestration
Execute
Environment
Test Artifacts
Fixtures
Setup


NITROGEN
Android Test Orchestrator
Collect Tests
Execute Tests
AndroidJUnitRunner
Execute Test
Result Data


Collection
Isolation
Orchestration
Execute
Environment
Test Artifacts
Fixtures
Setup
Report
Unified Format
Output Data
Scoping


Local development
CI-Server
NITROGEN
Project Nitrogen
Firebase Test Lab
Google Cloud
Real Device
Virtual Device
Simulated Device

DEMO


NITROGEN
Project
Nitrogen
e2e
Tests
Integration Tests
Unit Tests
NITROGEN
Android Test
Part of Jetpack

Practical Exam

- IMPORTANT! Obey the exam scheduler.
- Request, by email, attendance reschedule in the secondary date if something **major** is not allowing you to attend the primary date.

Software Requirements

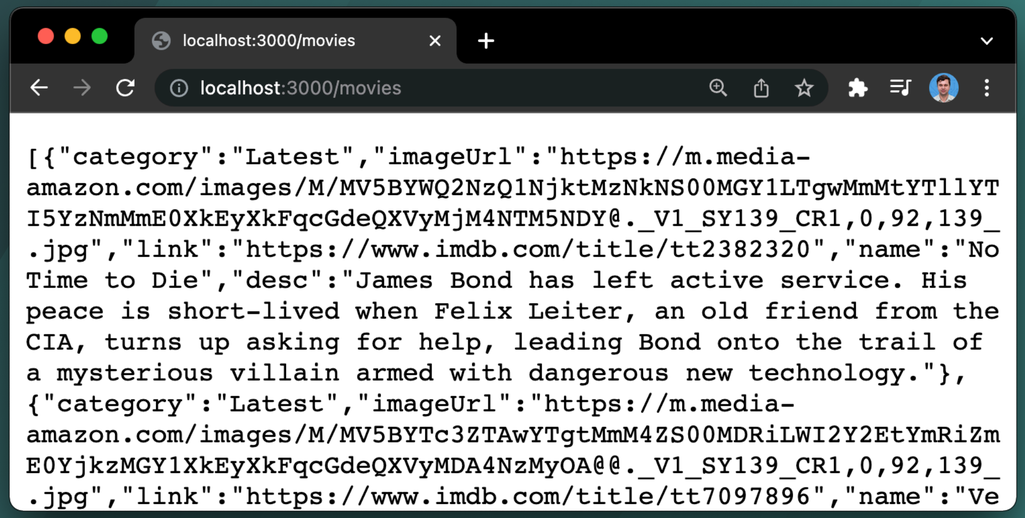
- Ensure that you have installed on your machine, the latest versions for:
  - Git
  - nodeJs
  - NPM
- ~~Are able to share your camera and desktop.~~

Execute the Server

- Ensure that you are able to execute the server from:
  - https://github.com/dancojocar/MA/tree/master/lectures/14/server
- Steps to execute the server:
  1. Update the dependencies: **npm install**
  2. Start the server: **npm start**
- The above two commands are executed under the **server** directory!
- Note: You should be running the latest version of node (v21.5.0) and npm (v10.2.4).

Verify the Server

- Using the browser of your choice go to:
    - http://localhost:3000/movies
- If everything is working properly you should receive a JSON similar to:



Practical Exam

- Two hours to solve the requirements.
- In this time:
  - You are **NOT** allowed to:
    - Discuss with anyone!
    - Disturb others!
  - You are allowed to:
    - Use all the materials available on your local machine or on the Internet. So, ensure that you are well prepared!
    - Request clarifications only from the exam supervisor.
- If found breaking at least one of the above rules you will be denied to take the current exam and the following ones held this year!

Lecture outcomes

- Introduction to the concepts and tools for building Android tests.
- Build complex unit tests with Android dependencies that cannot be satisfied with mock objects.
- Create tests to verify that the user interface behaves correctly for user interactions within a single app or for interactions across multiple apps.
- Create a stable and reusable testing harness to run performance tests.