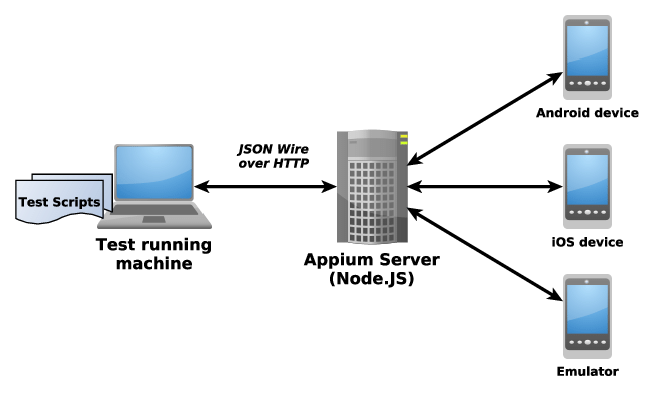
What is an Appium

**Appium** is an open-source, cross-platform tool for automating mobile application testing on Android and iOS devices. It supports native, hybrid, and mobile web apps etc. By leveraging the WebDriver protocol, Appium enables developers and testers to write test scripts in multiple programming languages (like Java, Python, Ruby, and JavaScript).

Appium is an opensource test automation framework used for mobile automation testing across different platforms such as iOS and Android. It also supports mobile web browser automation and has a wide variety of language support for automation.

# Appium Architecture



Appium is an HTTP server written in [**nodejs**](https://nodejs.org/en/) that exposes REST API. The client communicates with Appium server via REST API’s and it is handled by Mobile JSON Wire Protocol.

Step 1: Client libraries convert the user written commands to the **REST API** requests.

Step 2: These requests are sent to the Appium Server using ***Mobile JSON Wire Protocol***.

Step 3: Appium server forward these requests to target android device/emulator.

Step 4: These commands are interpreted by Appium flutter driver which converts them into flutter framework understandable because flutter is cross platform.

Step 4: The UIAutomator commands are now performed on the device/emulator via flutter Inspector.

Step 5: Device/emulator then reverts the outcome of the performed command to the Appium server via Appium Flutter Driver.

Step 6: Appium server forwards this response to the client.

Note: How Appium server determine the platform which we are testing. It will determine via the concept of desiredCapabilities.

## Placeholder and their meaning

## <ext-type>: This refers to the "Extension type". It is a mandatory parameter that specifies the type of extension being used. The value of `<ext-type>` can be either:

## driver: This indicates that the extension is a driver, which is a module that provides a specific functionality to Appium.

## plugin: This indicates that the extension is a plugin, which is a module that extends the functionality of Appium.

Example: appium driver list (In this case extension type is a driver).

Example: appium plugin list

## All the Extension CLI commands can be used with either drivers or plugins, so it is essential to specify the type of extension being used.

## <ext-name>: This refers to the "Extension name". It is a short name of the extension that can be found in a call to `appium <ext-type> list`. This name is distinct from:

## The NPM (Node Package Manager) package name of the extension.

## The "install spec" of the extension (which is explained below).

Example: npm install -g appium (In this case extension name is a npm).

## The `<ext-name>` is used to identify the extension and perform actions on it, such as installing, updating, or removing it.

## <install-spec>: This refers to the "Install specification". It is a string used to indicate what extension Appium should install. The `<install-spec>` can be in various formats, such as:

## A URL pointing to a GitHub repository or a npm package.

## A npm package name.

## A local file path.

## A version number.

## The `<install-spec>` provides the necessary information for Appium to download and install the extension.

## <install-source>: This refers to the method that Appium should use to install an extension. The `<install-source>` can be one of the following:

## npm: Install the extension from the npm registry.

## github: Install the extension from a GitHub repository.

## local: Install the extension from a local file or directory.

## url: Install the extension from a URL.

## The `<install-source>` determines the installation method and provides the necessary information for Appium to retrieve the extension.

## Universal programming language access

Appium is ultimately a Node.js program.

the WebDriver specification is actually an HTTP-based protocol, meaning it is designed to be used over a network rather than within the memory of a single program.

The main benefits of this "client-server" architecture are that it allows the automation implementer (the thing doing the automation, in this case the 'server') to be completely distinct from the automation runner (the thing defining what automation should be done, in what steps, etc..., in this case the 'client'). Basically, all the "hard stuff" (actually figuring out how to make automation happen on a given platform) can be handled in one place by the server, and "thin" client libraries can be written in any programming language which simply encode HTTP requests to the server in language-appropriate way.