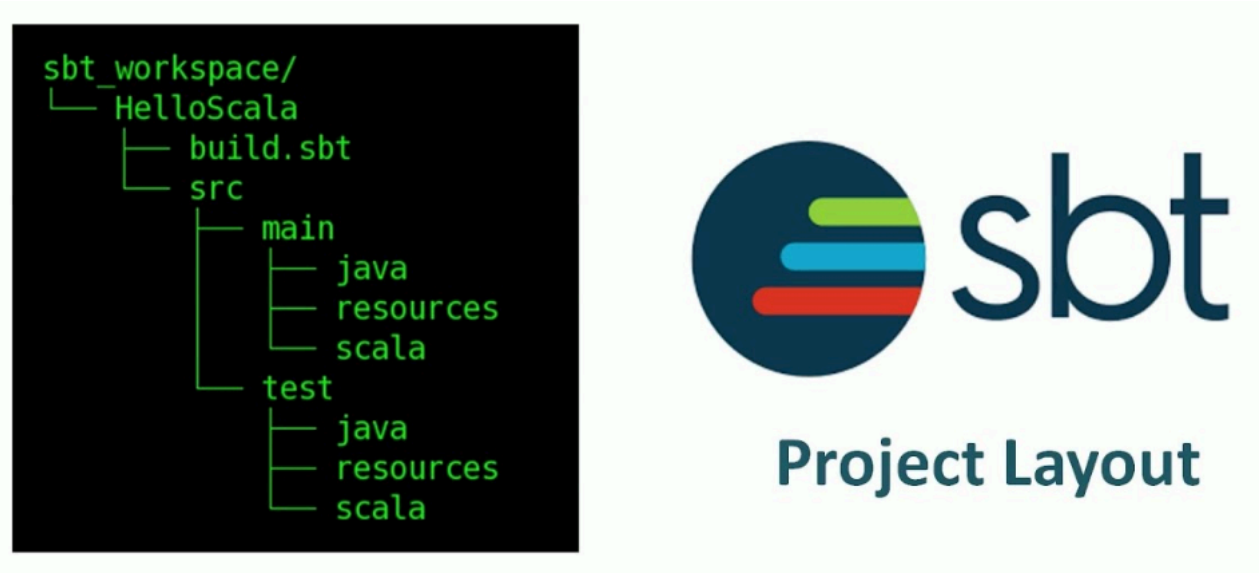
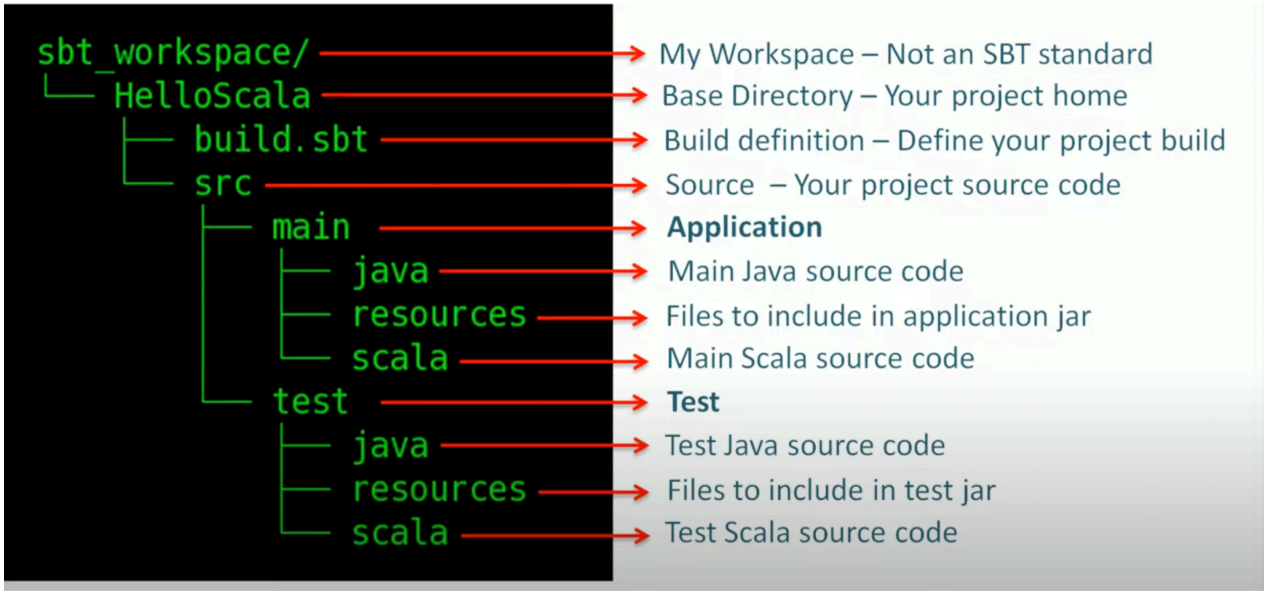
Scala Environment

Scala projects, particularly those built with sbt (Scala Build Tool), typically follow a **standardized directory structure**, similar to Maven projects. This structure promotes organization and simplifies build processes.

- SBT - Based on Maven.
- SBT_Workspace - Workspace Directory
- Hello Scala is my first project
- All projects should follow this structure
- Src - For source code - Main and Test
    - Application code
    - Test Code
- Can code in java and scala





✅ Scala Environment Overview

Scala runs on the **Java Virtual Machine (JVM)** and interoperates seamlessly with Java. The development environment typically includes:

- **JDK (Java Development Kit)** – Required to compile and run Scala.
- **Scala Compiler (`scalac`)** – Compiles `.scala` files to JVM bytecode.
- **Scala REPL (`scala`)** – An interactive shell for executing Scala code snippets.
- **Build Tools** – Typically **sbt (Scala Build Tool)**, or alternatives like Maven or Gradle.
- **IDE Support** – IntelliJ IDEA (with Scala plugin) is the most widely used.

✅ Setting Up the Scala Environment (Summary)

1. **Install Java (JDK 8 or above)**
   Recommended: JDK 11 or 17.
2. **Install Scala**
   - Via SDKMAN: `sdk install scala`
   - Or manually from https://www.scala-lang.org/download/
3. **Install sbt (Scala Build Tool)**
   - Via SDKMAN: `sdk install sbt`
   - Or from https://www.scala-sbt.org/

Verify Installations:

```
java -version
scala -version
scalac -version
sbt about
```

✅ Scala Folder Structure (Manual / sbt Project)

When you **create a Scala project using sbt**, a typical folder structure looks like this:

```
your-scala-project/
├── build.sbt                 <- Build definition file
├── project/                  <- sbt settings/plugins
│   └── build.properties
├── src/
│   ├── main/
│   │   ├── scala/            <- Scala source files
│   │   └── resources/        <- Non-code resources (conf,
json, etc.)
│   └── test/
│       ├── scala/            <- Scala test files
│       └── resources/        <- Test resources
└── target/                   <- Compiled output
(auto-generated)
```

✅ Key Files & Folders Explained

File/Folder	Purpose

`build.sbt`	Main build configuration file, like a `pom.xml` in Maven.
`project/build.properties`	Defines sbt version. Required to compile the project.
`src/main/scala/`	All production Scala source code lives here.
`src/main/resources/`	Configuration files, templates, static assets for main app.
`src/test/scala/`	Scala test code (ScalaTest, specs2, etc.).
`src/test/resources/`	Test-related configs, input data, mocks, etc.
`target/`	sbt's output directory — contains compiled classes, JARs, metadata.

✅ Simple `build.sbt` File Example

```
name := "HelloScalaProject"

version := "0.1"

scalaVersion := "2.13.12"

libraryDependencies += "org.scalatest" %% "scalatest" % "3.2.15" % Test
```

✅ Scala CLI Alternative

For small projects or scripts, you can use **Scala CLI**:

```
scala-cli run Main.scala
```

Structure with Scala CLI is flat. You can place `.scala` files in the root directory.

✅ Environment Variables (Optional)

Set in `.bashrc`, `.zshrc`, or `.bash_profile`:

```
export SCALA_HOME=/path/to/scala
export PATH=$SCALA_HOME/bin:$PATH
```

✅ REPL Usage

After installing Scala, run:

```
scala
```

You'll get an interactive shell to test code snippets:

```
scala> val x = 42
val x: Int = 42
```

✅ Compiling and Running Scala Manually

Save Code (e.g., `Hello.scala`)

```
object Hello extends App {
  println("Hello, Scala!")
}
```

1. **Compile**: `scalac Hello.scala`
2. **Run**: `scala Hello`
3. This generates `.class` files in the current directory.

Summary

Tool	Purpose
scalac	Scala compiler
scala	REPL and runner for Scala bytecode
sbt	Build tool like Maven/Gradle
scala-cli	Lightweight, script-friendly CLI tool
IntelliJ	IDE with strong Scala support
