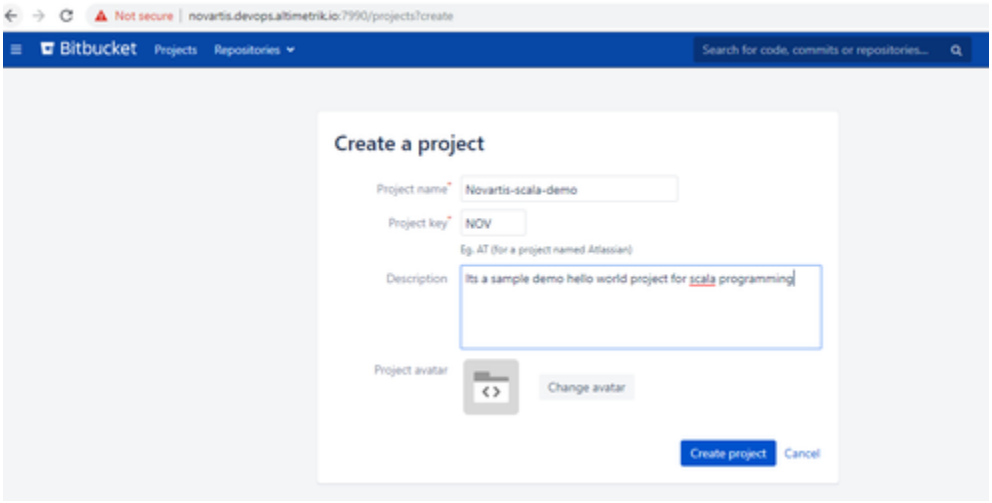
How to - Bitbucket repo setup - Create sample scala repo

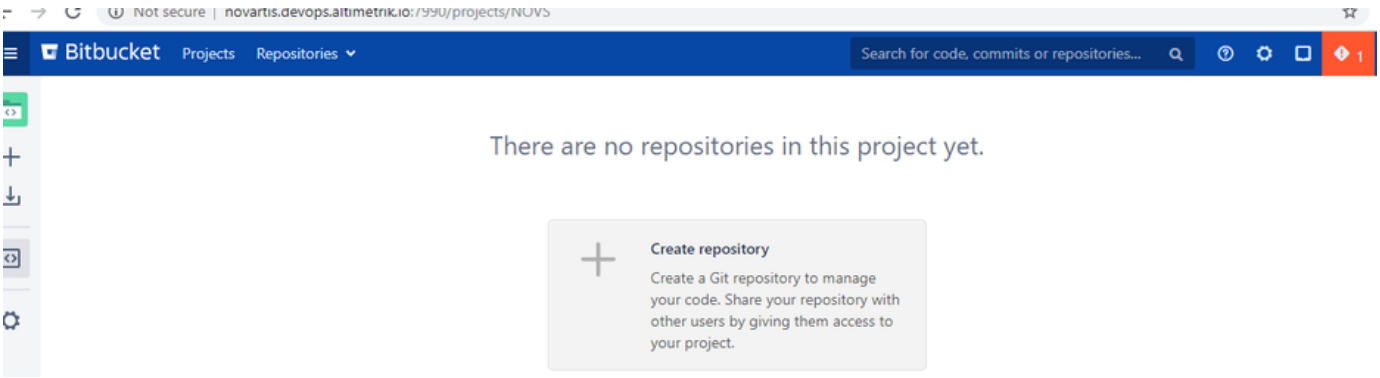
Bitbucket repository creation:

1. Go to project > Create project

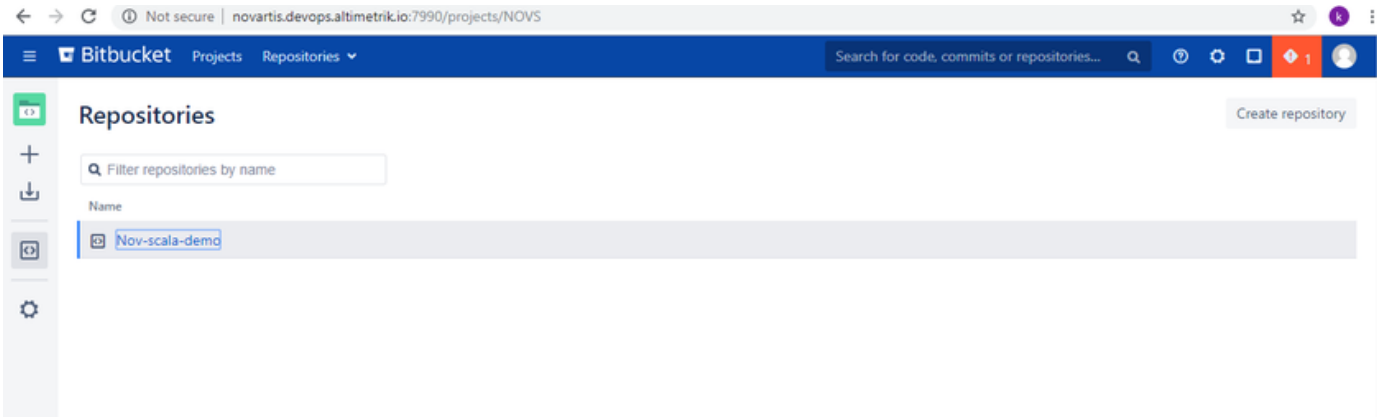
2. Create sample project name



3. Click on Create repository



4. Give the repository name and click on create repository



it will create with an empty repository.

To get started you will need to run below commands.

Configure Git for the first time

1. **Setting your Git username for every repository on your computer.open git bash and run below commands**

```
rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git config --global user.name "rajasekhar"

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git config --global user.email "rkendole@altimetrik.com"

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
```

To check whether username is added or not run git config command

```
rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git config --global user.name "rajasekhar"

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git config --global user.email "rkendole@altimetrik.com"

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge --skip -- %f
filter.lfs.process=git-lfs filter-process --skip
filter.lfs.required=true
credential.helper=manager
user.name=rajasekhar
user.email=rkendole@altimetrik.com

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$
```

Clone repository

1.If you want to simply clone this empty repository then run this command in your terminal.

git clone http://novartis.devops.altimetrik.io:7990/scm/novs/nov-scala-demo.git

```
rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ git clone http://novartis.devops.altimetrik.io:7990/scm/novs/nov-scala-demo.git
Cloning into 'nov-scala-demo'...
warning: You appear to have cloned an empty repository.

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project
$ ls
nov-scala-demo/
```

2.If you already have code ready to be pushed to this repository then run this in your terminal

```
cd nov-scala-demo
git init
```

3. Add one file README.md and below commands

```
git add .

  git commit -m "Initial Commit"

  git push -u origin master
```

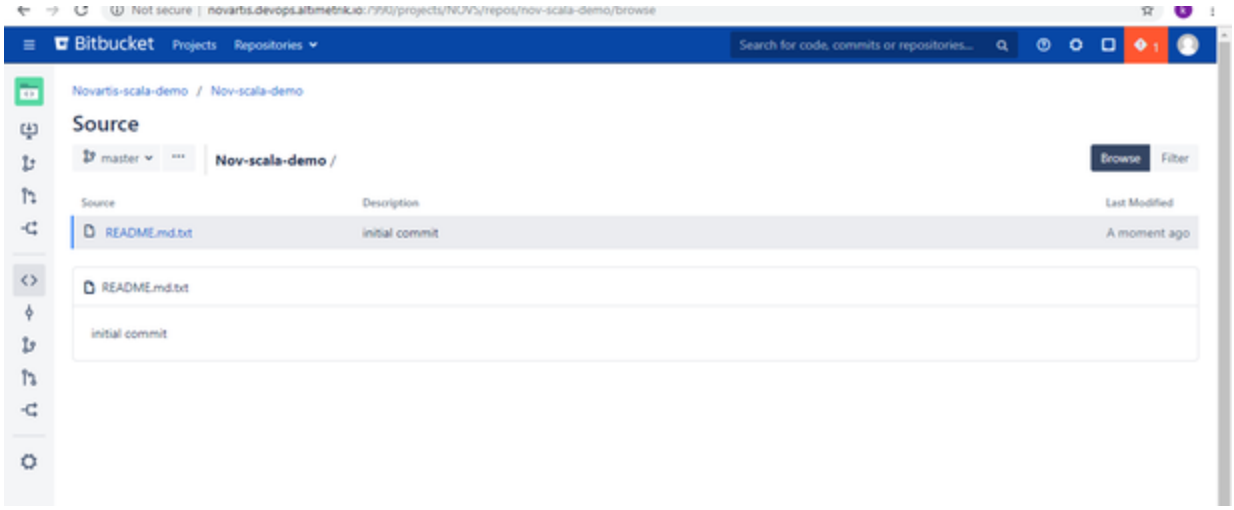
```
nothing added to commit but untracked files present (use "git add" to track)

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ git add .

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ git commit -m "initial commit"
[master (root-commit) 28b5e0f] initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md.txt

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 232 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://novartis.devops.altimetrik.io:7990/scm/novs/nov-scala-demo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

you can see initial commit in repository.



How to CREATE BRANCH:

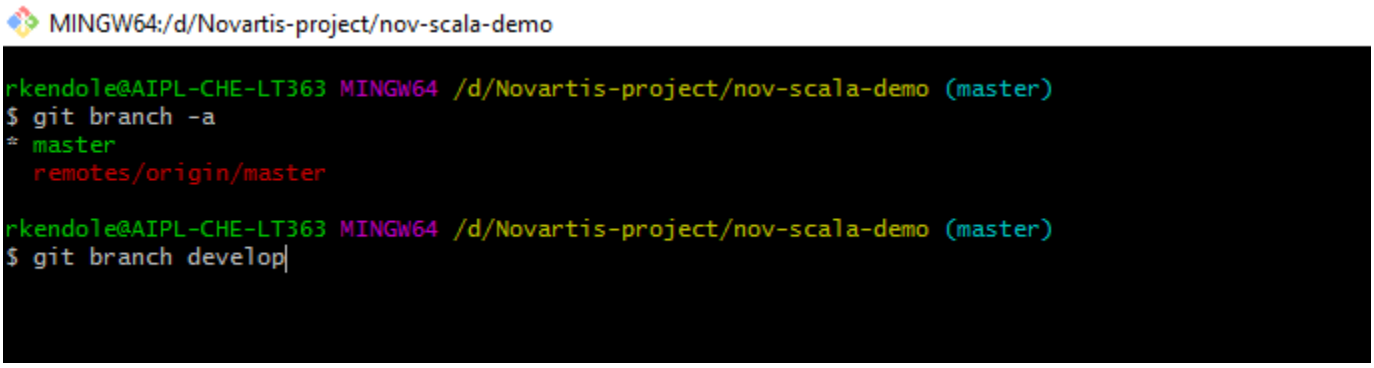
we can create branch in 2 ways

1. Manual using git commands
2. Bitbucket console

Using Git commands

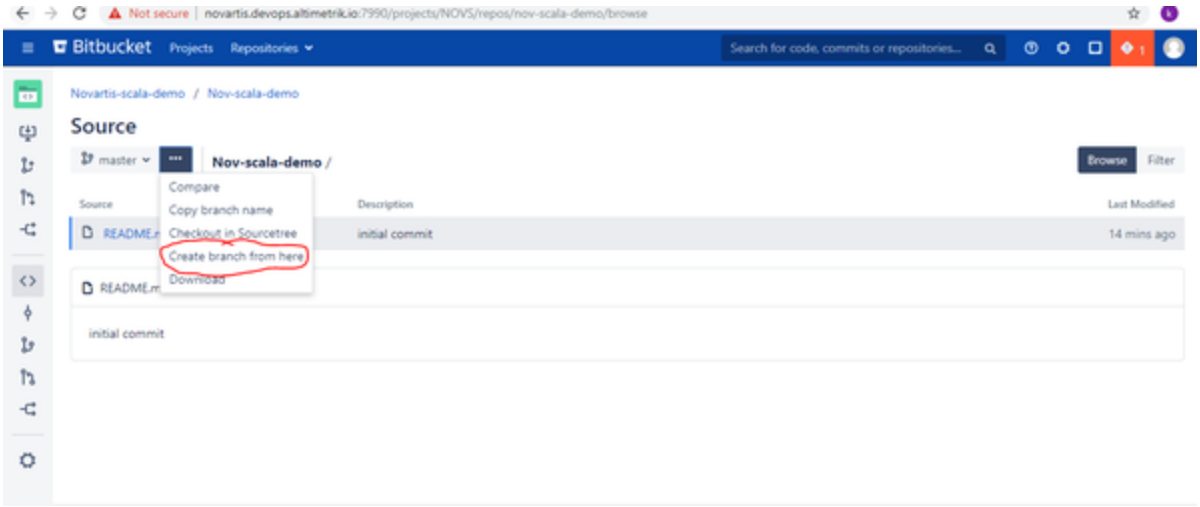
Go to bash terminal run below commands

git branch develop

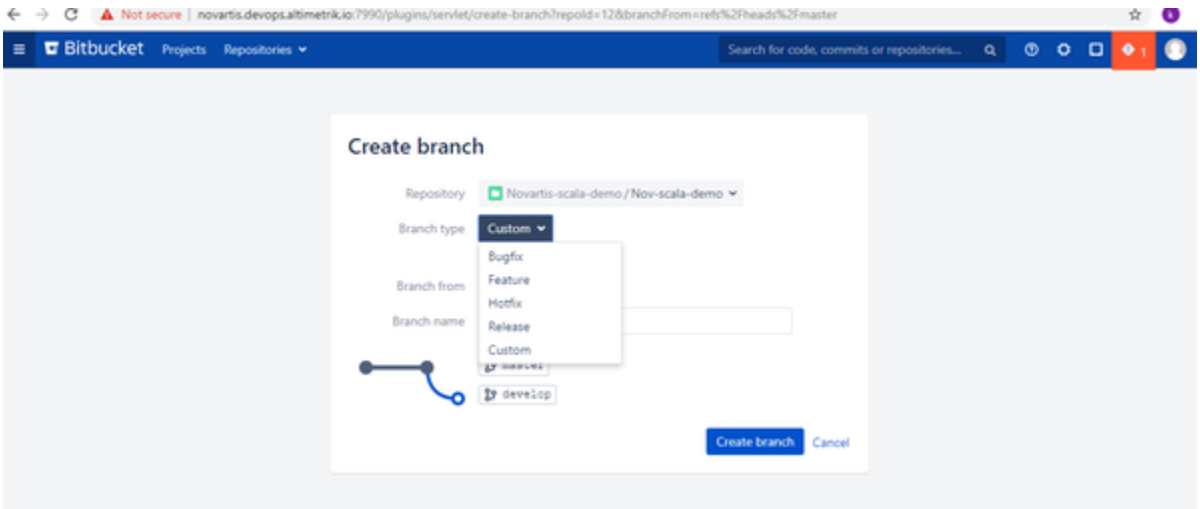


Using Bitbucket dashboard:

Go to repository >Create branch from here



2. Next give the branch name and choose branch type whether feature/hotfix/release and click on create branch



Branch has been created with develop branch.

Scala :

Scala combines object-oriented and functional programming in one concise, high-level language. Scala's static types help avoid bugs in complex applications, and its JVM and JavaScript runtimes let you build high-performance systems with easy access to huge ecosystems of libraries.

Install scala on windows:

PRE-REQUISITES:

1.OPERATING SYSTEM: RHEL/CENTOS/FEDORA AND UBUNTU/DEBIAN/LINUX MINT
2. JDK 1.8
3. SCALA 2.13.1

INSTALL SCALA:

1.Verify the JDK installation on your windows machine by typing the following commands in the command prompt

```
C:\Users\rkendole>java -version
java version "1.8.0_221"
Java(TM) SE Runtime Environment (build 1.8.0_221-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.221-b11, mixed mode)

C:\Users\rkendole>
```

2.Download Scala binaries from https://www.scala-lang.org/download/. The Scala installer file will be downloaded with .msi extension.
Run the extension file.
3. Set scala bin path in environment varibale
4. Check the version of scala.
scala -version

Scala Tools -

```
scala  -  Scala interactive interpreter

scalac -  Scala compiler
fsc    -  Scala resident compiler
scaladoc -  Scala API documentation generator

scalap   -   Scala classfile decoder


Run the command "scalac -help" to display the list of available
compiler options
```

Repository structure

```
scala/
+--build.sbt                The main sbt build script
+--lib/                     Pre-compiled libraries for the build
+--src/                     All sources
   +---/library             Scala Standard Library
   +---/reflect             Scala Reflection
   +---/compiler            Scala Compiler
   +---/intellij            IntelliJ project templates
+--spec/                    The Scala language specification
+--scripts/                 Scripts for the CI jobs (including building releases)
+--test/                    The Scala test suite
   +---/files               Partest tests
   +---/junit               JUnit tests
   +---/scalacheck          ScalaCheck tests
+--build/                   [Generated] Build output directory
```

Sample Hello World programming for Scala:

1.First lines of code

```
object Hello {
    def main(args: Array[String]) {
        println("Hello, world")
    }
}
```

Save the source code in repository in the name of Hello.scala.

2. **Compile the Scala code**

To run this use scalac command

```
one warning found

rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ scalac HelloWorld.txt
warning: there was one deprecation warning (since 2.13.0); re-run with -deprecation for details
one warning found
```

Scalac command creates two new files into current working directory.

- Hello$.class
- Hello.class

```
rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ ll
total 4
-rw-r--r-- 1 rkendole 1049089 669 Sep 19 16:13 'Hello$.class'
-rw-r--r-- 1 rkendole 1049089 617 Sep 19 16:13  Hello.class
-rw-r--r-- 1 rkendole 1049089  99 Sep 19 16:12  HelloWorld.txt
-rw-r--r-- 1 rkendole 1049089  14 Sep 19 12:46  README.md.txt
```

3. **RUN the the scala code**

Now you can run the `Hello` application with the `scala` command:

scala Hello

```
rkendole@AIPL-CHE-LT363 MINGW64 /d/Novartis-project/nov-scala-demo (master)
$ scala Hello
Hello,World
```

4. package the scala code

scalac HelloWorld.txt -d Hello.jar

How to compile, run, and package a Scala project with SBT:

SBT is an open source build tool in the Scala and java projects. If you're used to other build tools, you will be familiar with the commands:

- **clean**: Deletes files produced by the build, such as generated sources, compiled classes, and task caches.

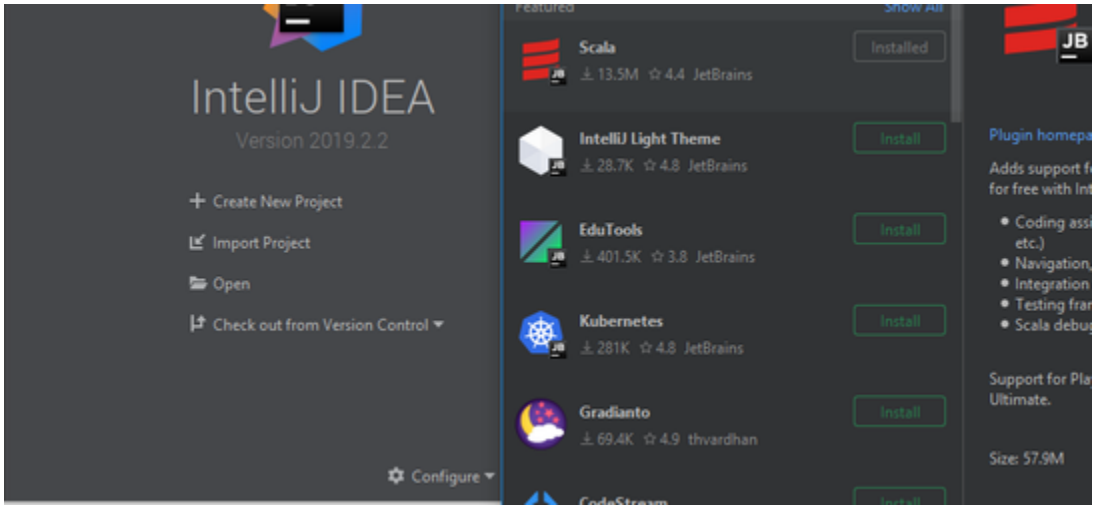
- **compile**: Compiles sources.
- **test**: Executes all tests.
- **package**: Produces the main artifact, such as a binary JAR. This is typically an alias for the task that actually does the packaging.
- **help**: Displays this help message or prints detailed help on requested commands (run 'help').
- **console**: Starts the Scala interpreter with the project classes on the classpath.

BUILDING A SCALA PROJECT WITH INTELLIJ AND SBT

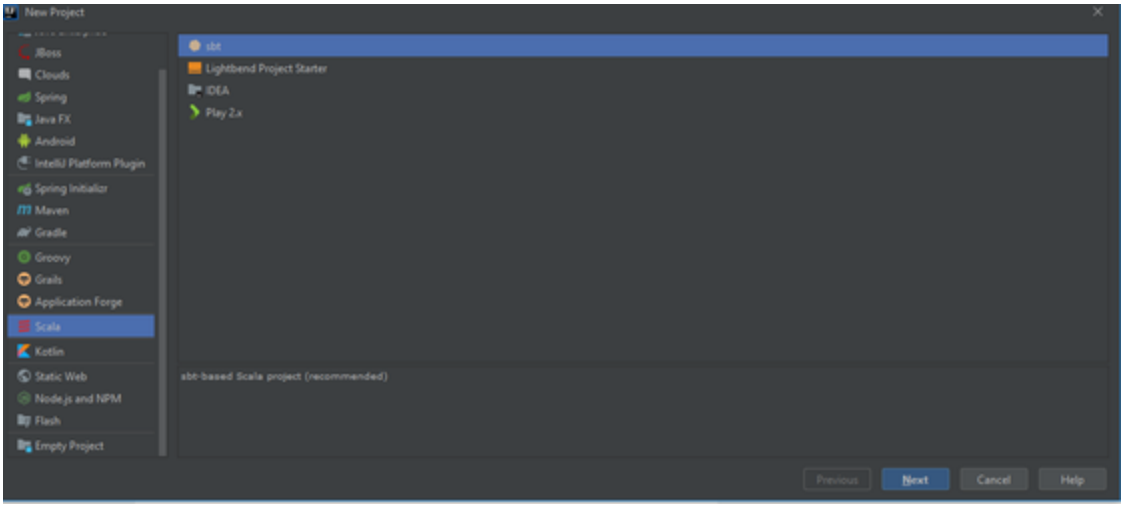
1. **Create the project**

Before creating project download scala plugin in configure section plugins

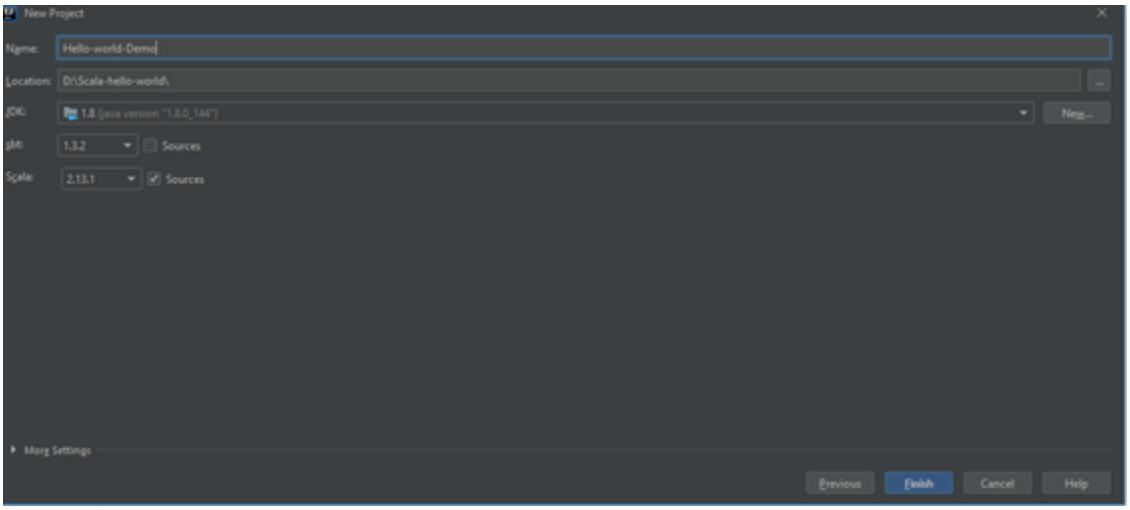
configure> plugins



then click Scala>sbt



click on next



Directory structure of sbt:

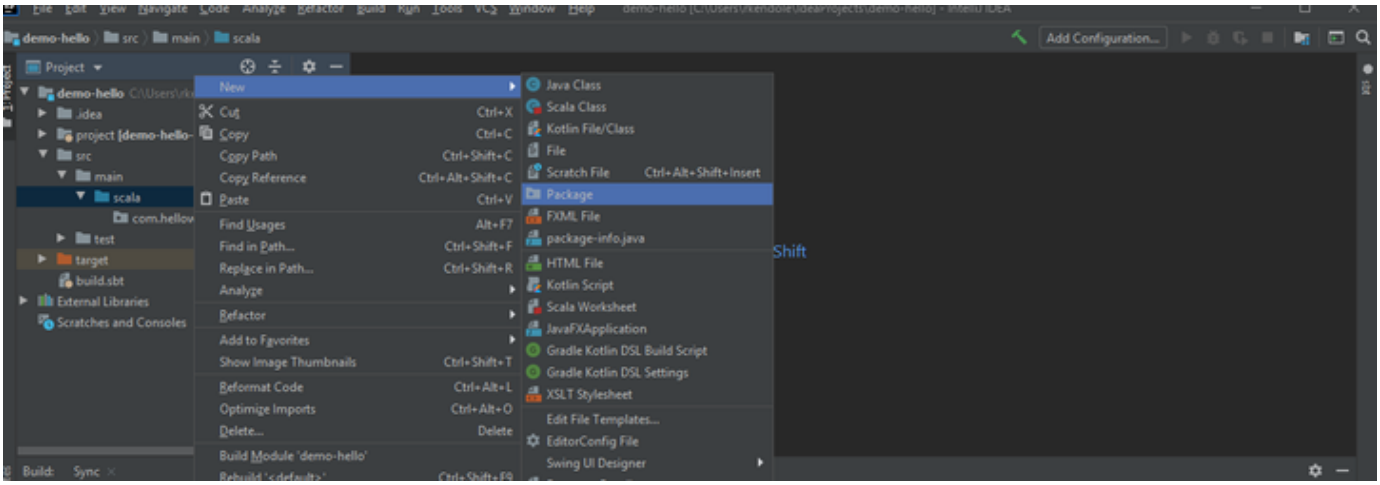
```
- .idea (IntelliJ files)
- project (plugins and additional settings for sbt)we need to build
build.sbt. This folder contains all the files needed for building
build.sbt.
- src (source files)
    - main (application code)
        - java (Java source files)
        - scala (Scala source files) <-- This is all we need for now
         - test (unit tests)
- target (generated files)contains artifacts from building the project
- build.sbt (build definition file for sbt) ex. project name,project
version,scala version, lib, etc
```

Writing Scala code:

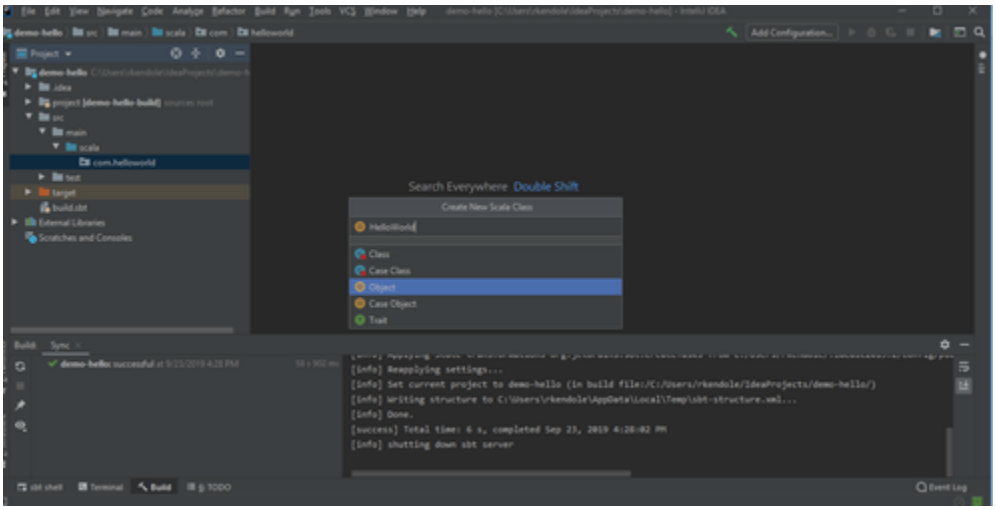
On the Project panel on the left, demo-hello src scala

Right click scala and select new package

let's create a package called `com.helloworld`



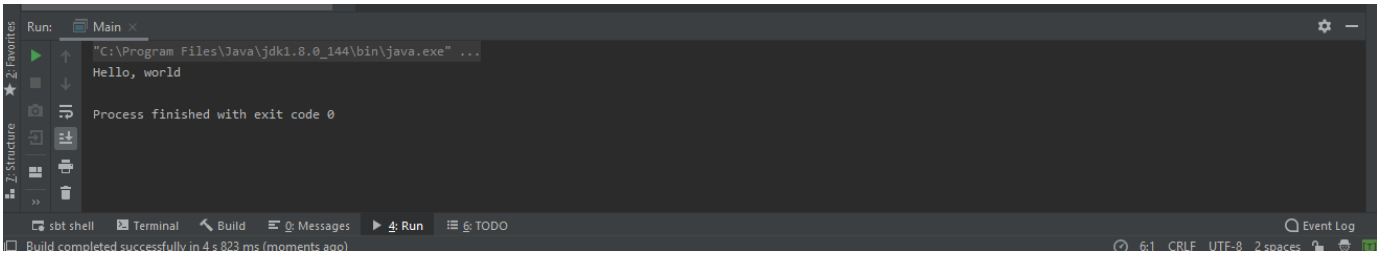
Then right-click on the package and select **New** -> **Scala Class**. We will name the class **HelloWorld**



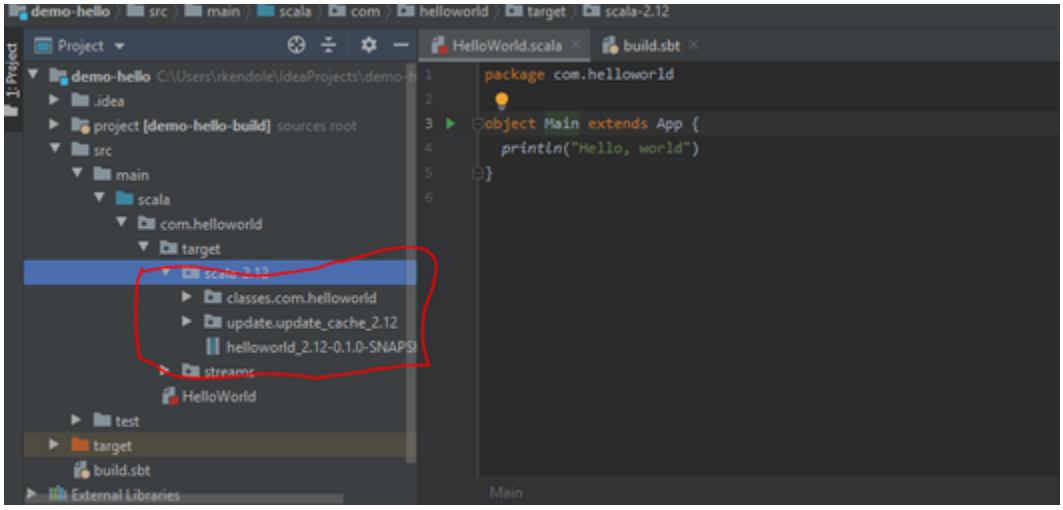

```
package com.helloworld

object Main extends App {
    println("Hello, world")
}
```

Right click anywhere and run main class



And the build artifact can be found in target/scala-2.13/classes



Unlike Java, in Scala, the file's package name doesn't have to match the directory name. For simple tests like this, you can place this file in the root directory of your SBT project.

From the root directory of the project, you can compile the project:

```
# compile the project

 sbt compile

# Run the project

 sbt run

# Package the project

sbt package
```


rkendole@AIPL-CHE-LT363 MINGW64 ~/IdeaProjects/demo-hello
$ sbt test
[info] Loading global plugins from C:\Users\rkendole\.sbt\1.0\plugins
[info] Loading project definition from C:\Users\rkendole\IdeaProjects\demo-hello
\project
[info] Loading settings for project demo-hello from build.sbt ...
[info] Set current project to demo-hello (in build file:/C:/Users/rkendole/IdeaP
rojects/demo-hello/)
[info] Compiling 1 Scala source to C:\Users\rkendole\IdeaProjects\demo-hello\tar
get\scala-2.13\classes ...
[info] Non-compiled module 'compiler-bridge_2.13' for Scala 2.13.1. Compiling...
8 warnings found
[info]   Compilation completed in 8.905s.
[info] Done compiling.
[success] Total time: 14 s, completed Sep 23, 2019 5:02:01 PM
rkendole@AIPL-CHE-LT363 MINGW64 ~/IdeaProjects/demo-hello
$ sbt package
[info] Loading global plugins from C:\Users\rkendole\.sbt\1.0\plugins
[info] Loading project definition from C:\Users\rkendole\IdeaProjects\demo-hello
\project
[info] Loading settings for project demo-hello from build.sbt ...
[info] Set current project to demo-hello (in build file:/C:/Users/rkendole/IdeaP
rojects/demo-hello/)
[success] Total time: 0 s, completed Sep 23, 2019 5:02:22 PM
rkendole@AIPL-CHE-LT363 MINGW64 ~/IdeaProjects/demo-hello
$ sbt compile
[info] Loading global plugins from C:\Users\rkendole\.sbt\1.0\plugins
[info] Loading project definition from C:\Users\rkendole\IdeaProjects\demo-hello\project
[info] Loading settings for project demo-hello from build.sbt ...
[info] Set current project to demo-hello (in build file:/C:/Users/rkendole/IdeaProjects/demo-hello/)
[info] Executing in batch mode. For better performance use sbt's shell
[success] Total time: 0 s, completed Sep 23, 2019 5:03:19 PM
rkendole@AIPL-CHE-LT363 MINGW64 ~/IdeaProjects/demo-hello
$