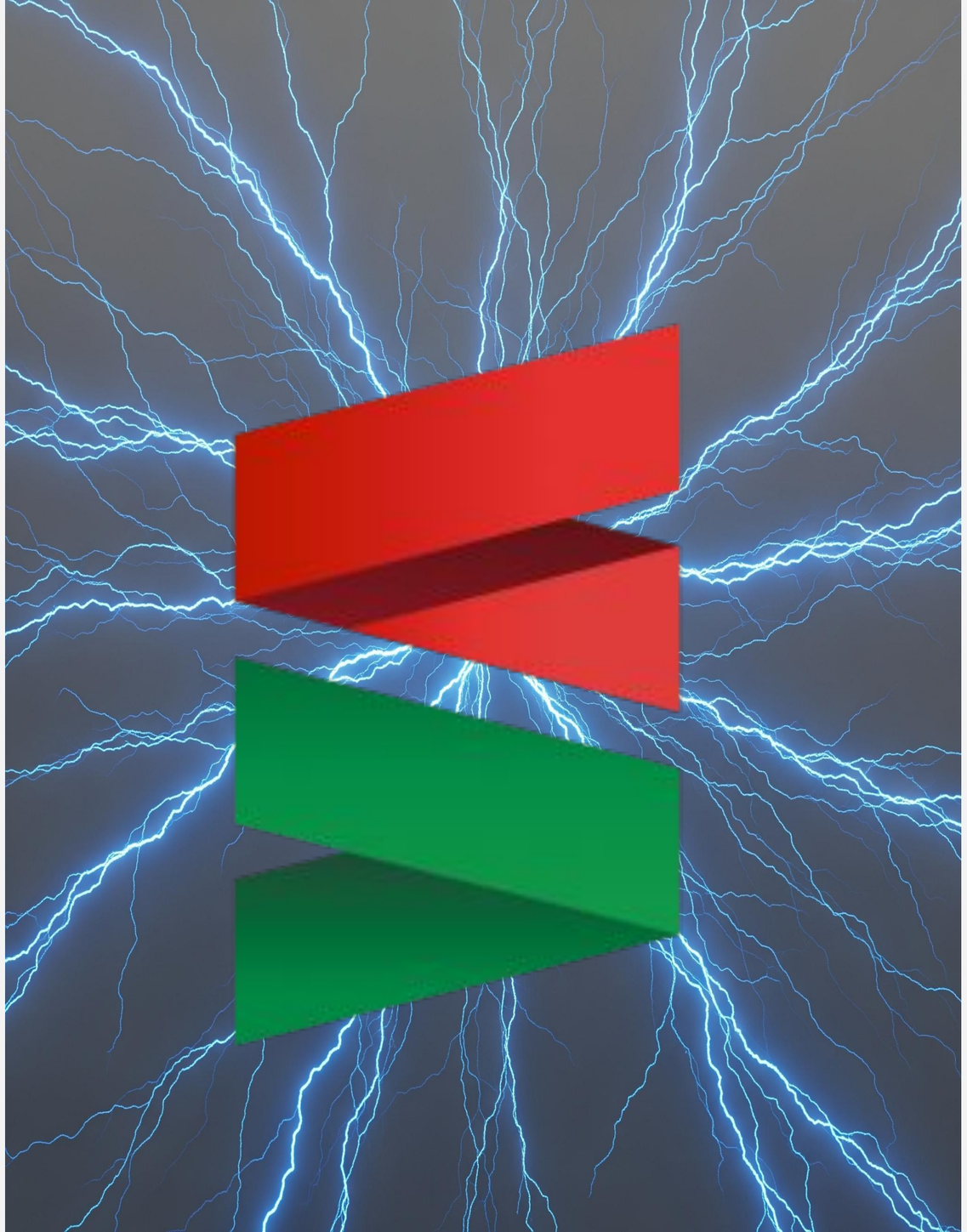
VIRTUSLAB

# Reach for the unlimited `--power` of Scala CLI

**Piotr Chabelski**
Product Manager @ Scala 3 Core Team



# Scala CLI

- a command-line tool to **compile**, **run** and **test** your Scala code (and more!)
- the official language runner as of Scala 3.5.0 (as per SIP-46)
- not quite a build tool (but almost!)

```
> scala -e 'println("Hello")'
Compiling project (Scala 3.7.2, JVM(23))
Compiled project (Scala 3.7.2, JVM(23))
Hello
```



# Why do we even need a `--power` option?

```
> scala package .
The package sub-command is restricted.
You can run it with the --power flag or turn
power mode on globally by running:
  scala config power true
```



# Types of Scala CLI features

## Stable

- doesn't require `--power`
- stable
- production ready

## Restricted

- requires `--power`
- also stable
- needs bureaucracy to reach its final form

## Experimental

- requires `--power`
- prints a warning
- no guarantees
- use at your own risk
- but do use and give feedback!



# How to run Scala CLI in `--power` mode?

- pass the **`--power` command line option**

```
> scala package . --power
```

- set the **`SCALA_CLI_POWER` environment variable** to true

```
> export SCALA_CLI_POWER=true
```

- set the **`power` config key** to true

```
> scala config power true
```



# Features

which I'll try to cover in this talk

01 **`fix` sub-command**
Refactoring & linting

02 **`--cross` compilation**
All the options in one command

03 **`--jmh` benchmarks**
Java Microbenchmark Harness integration

04 **`package` sub-command**
Let's build some artifacts

05 **`export` sub-command**
For when you need an actual build tool

06 **Markdown inputs support**
Why document the code, just code the document

07 **`--offline` mode**
For when you actually don't want the CLI to download stuff

08 **`publish` sub-command**
For publishing your work

09 **`bloop` sub-commands**
For when the build server acts up

# fix (experimental sub-command)

- --scalafix (enabled by default)

```
> scala fix . --power
Running built-in rules...
No need to migrate directives for a
single source file project.
Nothing to do.
Built-in rules completed.
Running scalafix rules...
Compiling project (Scala 3.7.2, JVM (23))
[warn] ./Hello.scala:5:7
[warn] unused local definition
[warn]   val unused = "unused" //removed
[warn]       ^^^^^^
Compiled project (Scala 3.7.2, JVM (23))
scalafix rules completed.
```

```
//Hello.scala
//> using scala 3
//> using options -Wunused:all
@main def hello() = {
  val unused = "unused" //removed
  println("Hello")
}
```

```
//.scalafix.conf
rules = [
  RemoveUnused
]
```



- **`--enable-built-in-rules` (enabled by default)**
  - (currently, migrating directives to a single file is the only one)

```
> scala fix . --power --scalafix=false
Running built-in rules...
Writing project.scala
Removing directives from 1.scala
Removing directives from 2.scala
Removing directives from 3.test.scala
Built-in rules completed.
```

```scala
//1.scala
//> using scala 3
//> using jvm 23
@main def hello() =
  println(os.pwd)
  println(Messages.hello)
```

```scala
//2.scala
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
object Messages:
  val hello = "Hello"
```

```scala
//3.test.scala
//> using dep org.scalameta::munit:1.1.1
class MyTests extends munit.FunSuite:
  test("Hello"):
    assertEquals(Messages.hello, "Hello")
```

- **`--enable-built-in-rules` (enabled by default)**
  - (currently, migrating directives to a single file is the only one)

```
//project.scala
//Main
//> using scala 3
//> using jvm 23
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5

//Test
//> using test.dep org.scalameta::munit:1.1.1
```

```
//1.scala
//> using scala 3
//> using jvm 23
@main def hello() =
  println(os.pwd)
  println(Messages.hello)
```

```
//2.scala
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
object Messages:
  val hello = "Hello"
```

```
//3.test.scala
//> using dep org.scalameta::munit:1.1.1
class MyTests extends munit.FunSuite:
  test("Hello"):
    assertEquals(Messages.hello, "Hello")
```

# --cross compilation (experimental)

```
> scala compile . --power --cross
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
Compiling project (Scala 3.7.2, Scala.js 1.19.0)
Compiled project (Scala 3.7.2, Scala.js 1.19.0)
Compiling project (Scala 3.3.6, JVM (23))
Compiled project (Scala 3.3.6, JVM (23))
Compiling project (Scala 3.3.6, Scala.js 1.19.0)
Compiled project (Scala 3.3.6, Scala.js 1.19.0)
Compiling project (Scala 2.13.16, JVM (23))
Compiled project (Scala 2.13.16, JVM (23))
Compiling project (Scala 2.13.16, Scala.js 1.19.0)
Compiled project (Scala 2.13.16, Scala.js 1.19.0)
```

```scala
//Hello.scala
//> using scala 3 3.lts 2.13
//> using platform jvm js
object Hello extends App {
  println("Hello")
}
```



# `--jmh` benchmarks (experimental)

Java Microbenchmark Harness integration

```
//Benchmark.scala
package bench
import java.util.concurrent.TimeUnit
import org.openjdk.jmh.annotations.*

@BenchmarkMode(Array(Mode.AverageTime))
@OutputTimeUnit(TimeUnit.NANOSECONDS)
@Warmup(iterations = 1, time = 100, timeUnit = TimeUnit.MILLISECONDS)
@Measurement(iterations = 3, time = 100, timeUnit = TimeUnit.MILLISECONDS)
@Fork(0)
class Benchmarks {
  @Benchmark
  def foo(): Unit = {
    (1L to 10000000L).sum
  }
}
```



# `--jmh` benchmarks (experimental)

```
> scala . --power --jmh
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
(...)
# JMH version: 1.37
# VM version: JDK 23.0.1, OpenJDK 64-Bit Server VM, 23.0.1+11
(...)
# Warmup Iteration   1: 1675.506 ns/op
Iteration   1: 33.108 ns/op
Iteration   2: 32.593 ns/op
Iteration   3: 31.114 ns/op

Result "bench.Benchmarks.foo":
  32.272 ±(99.9%) 18.882 ns/op [Average]
  (min, avg, max) = (31.114, 32.272, 33.108), stdev = 1.035
  CI (99.9%): [13.390, 51.154] (assumes normal distribution)


# Run complete. Total time: 00:00:00
```



# package (restricted sub-command)

- default format: **lightweight launcher JAR**

```
> scala package Hello.scala --power
Compiling project (Scala 3.7.2, JVM(23))
Compiled project (Scala 3.7.2, JVM(23))
Wrote ~/demo/hello, run it with
  ./hello
> ./hello
Hello
```

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```



- **`--library` JAR**

```
> scala package Hello.scala --power --library
Wrote ~/demo/hello.jar
> scala hello.jar
Hello
```

- **`--assembly` fat JAR**

```
> scala package Hello.scala --power --assembly
Wrote ~/demo/hello.jar, run it with
  ./hello.jar
> ./hello.jar
Hello
```

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```



- **`--native-image` (with GraalVM)**

```
> scala package Hello.scala --power --native-image
=========================================
GraalVM Native Image: Generating 'hello'
(executable)...
=========================================
(...)
-----------------------------------------
Produced artifacts:
~/demo/hello (executable)
~/demo/hello.build_artifacts.txt (txt)
=========================================
Finished generating hello in 17.8s.
Wrote ~/demo/hello, run it with
   ./hello
> ./hello
Hello
```

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```



- `--native` image (with Scala Native)

```
> scala package Hello.scala --power --native
Downloading compiler plugin
org.scala-native:::nscplugin:0.5.8
Downloading 2 dependencies
Compiling project (Scala 3.7.2, Scala Native 0.5.8)
Compiled project (Scala 3.7.2, Scala Native 0.5.8)
[info] Linking (multithreadingEnabled=true, disable if
not used) (1089 ms)
[info] Discovered 907 classes and 5564 methods after
classloading
(...)
Wrote ~/demo/hello, run it with
  ./hello
> ./hello
Hello
```

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```



- **`--js` Scala.js => `.js` file**

```
> scala package Hello.scala --power --js
Compiling project (Scala 3.7.2, Scala.js 1.19.0)
Compiled project (Scala 3.7.2, Scala.js 1.19.0)
Wrote ~/demo/hello.js, run it with
  node hello.js
> node hello.js
Hello
```

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```

- **`--docker` image**

```
> scala package Hello.scala --power --docker –docker-image-repository hello
Started building docker image with your application, it might take some time
Built docker image, run it with
  docker run hello:latest
> docker run hello:latest
Hello
```



- `--js` with `--js-emit-wasm` for Scala.js + WebAssembly

```
> scala package Hello.scala --power --js --js-emit-wasm
Compiling project (Scala 3.7.2, Scala.js 1.19.0)
Compiled project (Scala 3.7.2, Scala.js 1.19.0)
Wrote ~/wasm/main.js/main.js, run it with
  node ./main.js/main.js
> export DENO_V8_FLAGS=--experimental-wasm-exnref
> deno --allow-read main.js/main.js
Hello
```

```
//Hello.scala
//> using scala 3
//> using platform js
//> using jsEmitWasm
//> using jsModuleKind es
@main def hello() =
  println("Hello")
```

**Note: deno is a JavaScript runtime which supports all of the not-yet-stable features used with the Scala.js => WebAssembly implementation**



# OS-specific artifacts

(needs to be run on a compatible system)

```
//Hello.scala
//> using scala 3
@main def hello() =
  println("Hello")
```

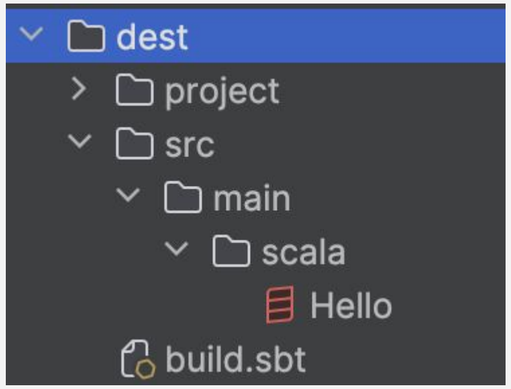
```
> scala package Hello.scala --power --deb #Debian
> scala package Hello.scala --power --rpm #RedHat
> scala package Hello.scala --power --pkg #MacOS
> scala package Hello.scala --power --msi #Windows
```



# export (experimental sub-command)

- --sbt (the default output)

```
> scala export . --power --sbt
Exporting to a sbt project...
Exported to: ~/demo/dest
```




```
//Hello.scala
//> using scala 3
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
@main def main() = println(os.pwd)
```

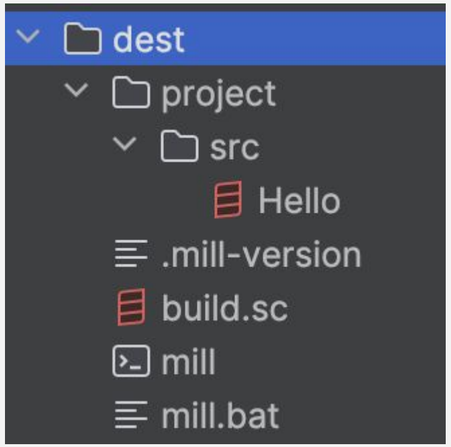
```
//build.sbt
scalaVersion := "3.7.2"
scalacOptions ++= Seq("-Wunused:all")
libraryDependencies +=
  "com.lihaoyi" %% "os-lib" % "0.11.5"
libraryDependencies +=
  "com.lihaoyi" %% "os-lib" % "0.11.5" % Test
```



- **`--mill` (currently up to Mill 0.12.15)**

```
> scala export . --power --mill
Exporting to a mill project...
Exported to: ~/demo/dest
```

```
//Hello.scala
//> using scala 3
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
@main def main() = println(os.pwd)
```



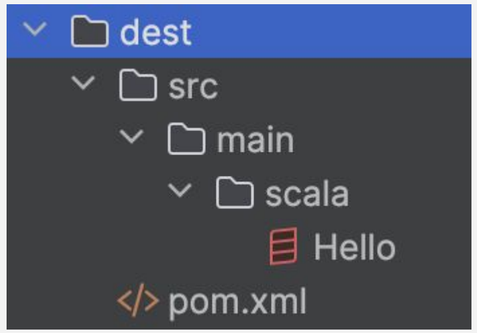

```
//build.sc
import mill._
import mill.scalalib._
object project extends ScalaModule {
  def scalaVersion = "3.7.2"
  def scalacOptions =
    super.scalacOptions() ++
      Seq("-Wunused:all")
  def ivyDeps = super.ivyDeps() ++ Seq(
    ivy"com.lihaoyi::os-lib:0.11.5"
  )
}
```



- **--mvn (yes, Maven)**

```
> scala export . --power --mvn
Exporting to a maven project...
Exported to: ~/demo/dest
```

```
//Hello.scala
//> using scala 3
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
@main def main() = println(os.pwd)
```




```
//pom.xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>maven-app</artifactId>
    <version>0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>com.lihaoyi</groupId>
            <artifactId>os-lib_3</artifactId>
            <version>0.11.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.scala-lang</groupId>
            <artifactId>scala3-library_3</artifactId>
            <version>3.7.2</version>
        </dependency>
        <dependency>
            <groupId>org.scala-lang</groupId>
            <artifactId>scala3-compiler_3</artifactId>
            <version>3.7.2</version>
        </dependency>
    </dependencies>
```

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <compilerArgs> </compilerArgs>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>4.9.1</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```



- **`--json` (for when you want your automation to consume a Scala CLI project configuration)**

```
> scala export . --power --json
```


```
{
  "projectVersion": "0.1.0-SNAPSHOT",
  "scalaVersion": "3.7.2",
  "platform": "JVM",
  "jvmVersion": "23",
  "scopes": [
    [
      "main",
      {
        "sources": ["~/demo/Hello.scala"],
        "scalacOptions": ["-Wunused:all"],
        "dependencies": [
          {
            "groupId": "com.lihaoyi",
            "artifactId": {
              "name": "os-lib",
              "fullName": "os-lib_3"
            },
            "version": "0.11.5"
          }
        ],
        "resolvers": ["https://repo1.maven.org/maven2"]
      }
    ]
  ],
  "scalaCliVersion": "1.8.5"
}
```


```
//Hello.scala
//> using scala 3
//> using options -Wunused:all
//> using dep com.lihaoyi::os-lib:0.11.5
@main def main() = println(os.pwd)
```



# Markdown inputs support (experimental)

- **run your Scala snippets directly in your `.md` files!**

```
> scala run . --power --md
Compiling project (Scala 3.3.6, JVM (23))
Compiled project (Scala 3.3.6, JVM (23))
Hello, world!
```

````
//hello.md
# A file with some content
Let's print a message to the
console!
```scala
//> using scala 3.lts
println("Hello, world!")
```
````



**Note, that all of the snippets from Markdown are technically scripts with a shared context.**

```
> scala run . --power --md
Compiling project (Scala 3.3.6, JVM (21))
Compiled project (Scala 3.3.6, JVM (21))
Hello, world!
```

**Also note how directives are global here.**

````
//hello.md
# A file with some content
Let's go with some Scala.

## First snippet
Here, we'll put directives and
initialize a variable.
```scala
//> using scala 3.lts
val message = "Hello, world!"
```

## Second snippet
Here, we'll print it.
```scala
//> using jvm 21
println(message)
```
````



## Don’t want a script snippet?
## Use the `raw` keyword!

```
> scala run . --power --md
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
Hello, world!
```

````
//hello.md
# Try a raw snippet!
This is treated as a .scala file
would, rather than a .sc script.
```scala raw
@main def hello() =
  println("Hello, world!")
```
````

**Note: without the `raw` keyword, this example would have an ambiguous main class.**



# ...how about we include tests in markdown, while we're at it

```
> scala test . --power --md
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
Test:
  + example test 0.005s
```

````
//hello.md
# Here's a test
And here's an in-depth explanation, why
are we asserting that true indeed is true.
```scala test
//> using dep org.scalameta::munit:1.1.1
class Test extends munit.FunSuite {
  test("example test") {
    assert(true)
  }
}
```
````

**Note: test snippets are a part of test scope, so they won't be run alongside other snippets.**



## It’s also possible to run remote markdown sources, like a blog

```
> scala https://gist.github.com/Gedochao/6415211eeb8ca4d8d6db123f83f0f839
  --power --md
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
Hello
```

**Note: remember that running remote inputs with Scala CLI has no sandboxing!**
**So be careful what you run.**



# `--offline` mode (experimental)

- pass the `--offline` **command line option**

```
> scala run . --power --offline
```

- set the `COURSIER_MODE` **environment variable** to `offline`

```
> export COURSIER_MODE=offline
```

- set the `-Dcoursier.mode` **property** to `offline`

```
> scala -Dcoursier.mode=offline run .
```



# `--offline` mode (experimental)

What it does:

- no artifacts' version validation
- defaults to `--server=offline` if Bloop artifacts unavailable
- no external JVMs, artifacts or dependencies downloaded

Requirement: prefill `coursier` cache

```
> cs install scala:3.7.2 scalac:3.7.2
> cs java --jvm 17
> cs fetch com.lihaoyi::os-lib::0.11.5
> cs fetch ch.epfl.scala:bloop-frontend_2.12:2.0.13
```



# publish (experimental sub-command)

```
//sonatype-conf.scala
//> using publish.user env:SONATYPE_USERNAME
//> using publish.password env:SONATYPE_PASSWORD

//> using publish.secretKey env:PGP_SECRET
//> using publish.secretKeyPassword env:PUBLISH_SECRET_KEY_PASSWORD

//> using publish.repository central
//> using publish.organization io.github.gedochao
//> using publish.name hello
//> using publish.version 0.0.1-SNAPSHOT

//> using publish.license Apache-2.0
//> using publish.url https://github.com/Gedochao/hello
//> using publish.scm github:Gedochao/hello
//> using publish.developer "Gedochao|Piotr Chabelski|https://github.com/Gedochao"
```



# publish local

```
> scala publish local . --power
Compiling project (Scala 3.7.2, JVM (23))
Compiled project (Scala 3.7.2, JVM (23))
Publishing io.github.gedochao:hello_3:0.0.1
Flag -classpath set repeatedly
1 warning found
 ✍️  Signed 5 files
 ✔ Computed 20 checksums
 🚚 Wrote 30 files

 👀 Check results at
  ~/.ivy2/local/io.github.gedochao/hello_3/0.0.1/
```



# `publish` supported targets

- `central` and `sonatype:snapshots` (the new **Sonatype Central Portal**)
- `ivy2-local`
- `github`
- any repository supported by `coursier` (like `jitpack`)
- any generic Maven repository
- any local directory



# bloop commands (restricted)

- **bloop/bloop start start the build server**

```
> scala bloop --power
Bloop server is running
```

- **bloop exit stops the build server**

```
> scala bloop exit --power
Stopped Bloop server.
```



# `bloop` commands (restricted)

- `bloop output` prints build server status

```
> scala bloop output --power
Ignoring SIGINT
Will truncate output file
~/Library/Caches/ScalaCli/bloop/daemon/output every 5 minutes
NGServer [UNKNOWN] started on local socket
~/Library/Caches/ScalaCli/bloop/daemon/socket.
```



# Try out Scala CLI nightlies for latest changes!

```
> scala --cli-version nightly version
Fetching Scala CLI 1.8.5-3-g038f11501-SNAPSHOT
Scala CLI version: 1.8.5-3-g038f11501-SNAPSHOT
Scala version (default): 3.7.2
```

**Note: nightlies have been restored in Scala CLI 1.8.5 / Scala 3.7.3**



# Thank You

**Piotr Chabelski**
Product Manager @ Scala 3 Core Team

pchabelski@virtuslab.com
https://github.com/Gedochao