

Modern Command Line Applications with Scala Native

Anton Sviridov

London, 2025

Table of contents

1. About me
2. Scala Native
3. Motivation for this talk
4. Sniper
5. Sniper
6. Modern CLI covenants
7. Components of a modern CLI
8. Distribution
9. Closing thoughts

About me

<https://blog.indoorvivants.com>

- Worked with Scala since 2014
- Advocated for Scala *everywhere* for years
- Maintainer and contributor to Scala Native ecosystem, the compiler, tools, and libraries
- Run a scala-centric blog

Github: [keynmol](#)

Twitter: [velvetbaldmime](#)

Bluesky: [@indoorvivants.com](#)

Posts tagged with

scala-native

Calling Scala Native from Java using FFM 2025-02-16

Let's point Java's newfangled foreign function and memory interface at a C portal into a Scala Native implementation. Zendaya meme will help explain.

Calling Java from Scala Native via JNI 2025-02-08

If you like suffering and have some Java libraries you just can't live without, your Scala Native application

Simple anti-toddler game with Scala Native and R 2025-02-08

To save my Slack and Discord from messages sent by a toddler, I need an e software. Let's build one with Raylib and Scala Native.

Scala Native and Swift: building Twotm8 MacOS a 2025-02-08

Let's build a Swift UI (MacOS) client app for Twotm8, using both Swift and

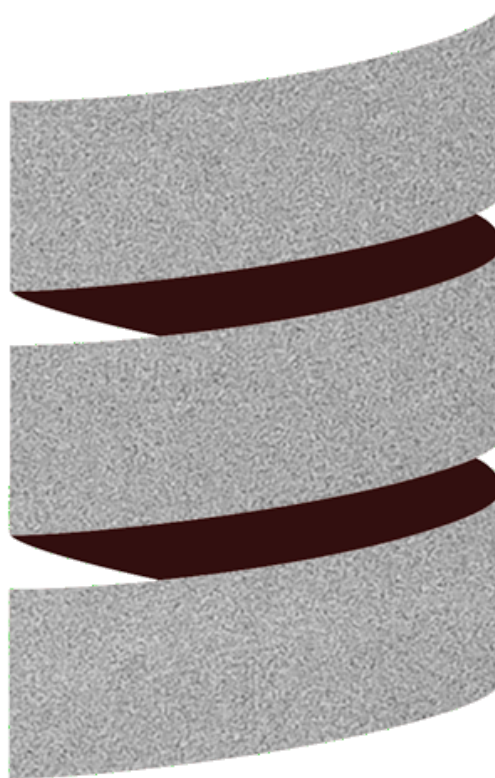
Twotm8 (p.4): Building the backend 2022-03-06

Armed with HTTP definitions and Postgres bindings, we build out the backe

Scala Native

<https://scala-native.org>

- Plugin to Scala compiler
- Build tools integrations
- Build **native** binaries, shared, and static libraries
- Interop with C libraries
- Compile C/C++/Assembly code alongside your Scala code
- Implements a large chunk of JDK-compatible APIs for cross-compilation



Getting started with Scala Native

- Install Clang and Scala CLI

```
//> using scala 3.7
//> using platform scala-native
//> using dep org.scala-lang.modules::scala-parallel-collections::1.2.0

import collection.parallel.CollectionConverters.*

@main def hello =
  Vector("L", "S", "U", "G").par.foreach(println(_))
```

- Package with `scala-cli package . -f -o ./hello-native`
- Run with `./hello-native`
- This is a self-contained native binary
- Multi-threading

Motivation for this talk

Multiple times a day I would run this incantation in order to test an idea, respond to a discord message, prepare a post, etc.

```
# go into a temp folder
cd $(mktemp -d) && \
  # create scala file
  touch test.scala && \
  # setup Metals configuration
  scala-cli setup-ide . && \
  # open editor
  nvim test.scala
```

Motivation for this talk

```
cd $(mktemp -d) && touch test.scala && scala-cli setup-ide . && nvim test.scala
```

- Very inflexible
- Relies on shell history
- Results are lost almost immediately
- No templates
- No search
- Can't be integrated into the rest of my workflow

Sniper

Fully-featured snippet management tool

```
> cd $(sniper new) && nvim .  
✓ Snippet description (max 240 characters) ... Hello, LSUG!  
? Select template to use ›  
  No template  
  ▶ scala3  
    c  
    leetcode  
    sbt-scala3-native  
    scala3-macro
```


Sniper

Fully-featured snippet management tool

```
> sniper open
? Select a snippet you want to see ›
  ▶ 72: Trying to get C macros values using libclang
    71: Bindings for SDL3
    68: Testing metals indexing
    67: Testing Math multiplyHigh on Scala Native for Andriy Plokhотnyuk
    66: Testing embed resources filters for scala native
    65: Testing Metals on 2.13 vs 3
    64: MCP proxy for Metals, STDIN to SSE
    63: Demonstrating union encoding with bindgen for https://github.com/scala-native/scala-native/issues/4339
    61: Testing the for comprehension desugaring
  ↓ 60: reproduction for inline given and nothing parameter in summon
```

Sniper

Fully-featured snippet management tool

- Manages folders automatically
- Templates and presets
- Moderately interactive CLI
- Trigram code search
- Embedded MCP server
- Alfred integration
- Shell completions
- TOML configuration

Choose your fighter

Unix approach

A selection of very small, self-contained tools, providing minimal functionality, designed to be combined together via text-based interface to achieve tasks

`ps` , `cat` , `ls` etc.

Kitchen sink approach

Single tool to do everything, hiding separate tasks and other CLI tools invocations under the hood

`kubectl` , `scala-cli` , `go` ,
`clang` , etc.

Choose your fighter

Sniper follows the kitchen sink approach.

Unix approach

A selection of very small, self-contained tools, providing minimal functionality, designed to be combined together via text-based interface to achieve tasks

`ps` , `cat` , `ls` etc.

Kitchen sink approach

Single tool to do everything, hiding separate tasks and other CLI tools invocations under the hood

`kubectl` , `scala-cli` , `go` ,
`clang` , etc.

Modern CLI covenants

- ☐ ☐ ☐ ☐ ☐

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data

-

-

-

-

-

-

-

-

-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
-
-
-
-
-
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
-
-
-
-
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
-
-
-
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
-
-
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
-
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
- Use interactive terminal features very sparingly
-
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
- Use interactive terminal features very sparingly
- Use known configuration formats, don't invent your own
-
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
- Use interactive terminal features very sparingly
- Use known configuration formats, don't invent your own
- Expect data to go out of sync and give user the power to fix it
-

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
- Use interactive terminal features very sparingly
- Use known configuration formats, don't invent your own
- Expect data to go out of sync and give user the power to fix it
- Expect and provide integration points

Modern CLI covenants

- Use standard directories (<https://dirs.dev/>) for config and data
- Bootstrap everything on first invocation
- Respect NO_COLORS (<https://no-color.org/>)
- Use nested subcommands in CLI arguments and provide shell completions
- Prefer self-contained binaries
- Confirmations for destructive actions
- Use interactive terminal features very sparingly
- Use known configuration formats, don't invent your own
- Expect data to go out of sync and give user the power to fix it
- Expect and provide integration points

There are many more soft and hard rules in order to be a useful and respectful guest on user's machine

Components of a modern CLI

- Argument parsing
- Configuration management
- Terminal interactivity
- Data storage
- AI slop for likes
- Integration with other tools

Ecosystem building blocks

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```

CLI parsing with decline-derive

<https://github.com/indoorvivants/decline-derive>

```
//> using dep com.indoorvivants::decline-derive::0.3.1

import decline_derive.*

enum Sub derives CommandApplication:
  case A(x: Int)
  case B(y: Option[String])

enum Command derives CommandApplication:
  @Help("Hello, LSUG!") case Hello(@Short("n") name: String)
  case Test(a: Sub)

@main def runCommand(args: String*): Unit =
  println(CommandApplication.parseOrExit[Command](args))
```


Configuration with toml-scala

<https://github.com/indoorvivants/toml-scala>

Resurrected from the dead, now with Scala 3 support thanks to community.

```
//> using dep com.indoorvivants::toml::0.3.0

import toml.*

case class Config(name: String, age: Int) derives Codec

@main def parse_toml =
  val string =
    """
    name = "John Doe"
    age = 30
    """
  println(Toml.parseAs[Config](string))
```

Configuration with toml-scala

<https://github.com/indoorvivants/toml-scala>

Resurrected from the dead, now with Scala 3 support thanks to community.

```
//> using dep com.indoorvivants::toml::0.3.0

import toml.*

case class Config(name: String, age: Int) derives Codec

@main def parse_toml =
  val string =
    """
    name = "John Doe"
    age = 30
    """
  println(Toml.parseAs[Config](string))
```

Configuration with toml-scala

<https://github.com/indoorvivants/toml-scala>

Resurrected from the dead, now with Scala 3 support thanks to community.

```
//> using dep com.indoorvivants::toml::0.3.0

import toml.*

case class Config(name: String, age: Int) derives Codec

@main def parse_toml =
  val string =
    """
    name = "John Doe"
    age = 30
    """
  println(Toml.parseAs[Config](string))
```

Configuration with toml-scala

<https://github.com/indoorvivants/toml-scala>

Resurrected from the dead, now with Scala 3 support thanks to community.

```
//> using dep com.indoorvivants::toml::0.3.0

import toml.*

case class Config(name: String, age: Int) derives Codec

@main def parse_toml =
  val string =
    """
    name = "John Doe"
    age = 30
    """
  println(Toml.parseAs[Config](string))
```

Terminal interactivity with cue4s

<https://github.com/neandertech/cue4s>

```
//> using dep tech.neander::cue4s::0.0.9

import cue4s.*

@main def prompts =
  Prompts.sync.use: prompts =>
    val name = prompts.text("What is your name?").getOrThrow
    val youOK = prompts.confirm(s"Are you ok, $name?").getOrThrow
    if !youOK then sys.error("You get better mate, come back later")
```

Available on JVM, JS, and Native

Terminal interactivity with cue4s

<https://github.com/neandertech/cue4s>

```
//> using dep tech.neander::cue4s::0.0.9

import cue4s.*

@main def prompts =
  Prompts.sync.use: prompts =>
    val name = prompts.text("What is your name?").getOrThrow
    val youOK = prompts.confirm(s"Are you ok, $name?").getOrThrow
    if !youOK then sys.error("You get better mate, come back later")
```

Available on JVM, JS, and Native

Terminal interactivity with cue4s

<https://github.com/neandertech/cue4s>

```
//> using dep tech.neander::cue4s::0.0.9

import cue4s.*

@main def prompts =
  Prompts.sync.use: prompts =>
    val name = prompts.text("What is your name?").getOrThrow
    val youOK = prompts.confirm(s"Are you ok, $name?").getOrThrow
    if !youOK then sys.error("You get better mate, come back later")
```

Available on JVM, JS, and Native

Terminal interactivity with cue4s

```
> scli run cue4s.scala
✓ What is your name? ... Anton
✓ Are you ok, Anton? ... no
Exception in thread "main" java.lang.RuntimeException: You get better mate, come back later
    at scala.sys.package$.error(package.scala:27)
    at cue4s$package$.prompts$$anonfun$1(cue4s.scala:9)
    at scala.runtime.function.JProcedure1.apply(JProcedure1.java:15)
    at scala.runtime.function.JProcedure1.apply(JProcedure1.java:10)
    at cue4s.SyncPromptsBuilder.use(SyncPrompts.scala:32)
    at cue4s$package$.prompts(cue4s.scala:6)
    at prompts.main(cue4s.scala:5)
```


Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

- Lorenzo Gabriele is *heroically* reimplementing JDBC for Scala Native, and publishing forked database libraries that work with it – Magnum and ScalaSql included, for Sqlite and DuckDB.
- Implementation is incomplete (JDBC is huge!) but you can already build real stuff with it
- The Scala library ships with sqlite3 embedded, so you don't need to install any dependencies
- **Excellent** opportunity to contribute to the ecosystem, highest ratio of impact to effort

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
//> using dep com.github.lolgab::magnum::1.3.1
//> using dep com.github.lolgab::scala-native-jdbc-sqlite::0.0.3
//> using platform scala-native

import com.augustnagro.magnum.*

@Table(SqliteDbType, SqlNameMapper.CamelToSnakeCase)
case class Vegetable(
  @Id id: Long,
  description: String
) derives DbCodec

@main def hello_jdbc =
  val sqlLogger = new SqlLogger:
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""
    override def log(successEvent: SqlSuccessEvent): Unit = ()
```

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
//> using dep com.github.lolgab::magnum::1.3.1
//> using dep com.github.lolgab::scala-native-jdbc-sqlite::0.0.3
//> using platform scala-native

import com.augustnagro.magnum.*

@Table(SqliteDbType, SqlNameMapper.CamelToSnakeCase)
case class Vegetable(
  @Id id: Long,
  description: String
) derives DbCodec

@main def hello_jdbc =
  val sqlLogger = new SqlLogger:
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""
    override def log(successEvent: SqlSuccessEvent): Unit = ()
```

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
//> using dep com.github.lolgab::magnum::1.3.1
//> using dep com.github.lolgab::scala-native-jdbc-sqlite::0.0.3
//> using platform scala-native

import com.augustnagro.magnum.*

@Table(SqliteDbType, SqlNameMapper.CamelToSnakeCase)
case class Vegetable(
  @Id id: Long,
  description: String
) derives DbCodec

@main def hello_jdbc =
  val sqlLogger = new SqlLogger:
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""
    override def log(successEvent: SqlSuccessEvent): Unit = ()
```

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
@Id id: Long,  
  description: String  
) derives DbCodec  
  
@main def hello_jdbc =  
  val sqlLogger = new SqlLogger:  
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""  
    override def log(successEvent: SqlSuccessEvent): Unit = ()  
  
  val dataSource = new org.sqlite.SQLiteDataSource()  
  dataSource.setUrl(s"jdbc:sqlite:./vegetables.db")  
  
  connect(Transactor(dataSource, sqlLogger)):  
    sql """  
      CREATE TABLE IF NOT EXISTS vegetable (  
        id INTEGER PRIMARY KEY,  
        description TEXT NOT NULL
```

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""
    override def log(successEvent: SqlSuccessEvent): Unit = ()

val dataSource = new org.sqlite.SQLiteDataSource()
dataSource.setUrl(s"jdbc:sqlite:./vegetables.db")

connect(Transactor(dataSource, sqlLogger)):
  sql """
    CREATE TABLE IF NOT EXISTS vegetable (
      id INTEGER PRIMARY KEY,
      description TEXT NOT NULL
    );
    """ .update.run()
val repo = Repo[Vegetable, Vegetable, Long]
repo.insert(Vegetable(5, "cucumber"))
println(repo.findById(5))
```

Sqlite with Magnum and JDBC reimplementation

<https://github.com/lolgab/scala-native-jdbc>

```
    override def exceptionMsg(exceptionEvent: SqlExceptionEvent): String = ""
    override def log(successEvent: SqlSuccessEvent): Unit = ()

val dataSource = new org.sqlite.SQLiteDataSource()
dataSource.setUrl(s"jdbc:sqlite:./vegetables.db")

connect(Transactor(dataSource, sqlLogger)):
  sql """
    CREATE TABLE IF NOT EXISTS vegetable (
      id INTEGER PRIMARY KEY,
      description TEXT NOT NULL
    );
  """ .update.run()
val repo = Repo[Vegetable, Vegetable, Long]
repo.insert(Vegetable(5, "cucumber"))
println(repo.findById(5))
```

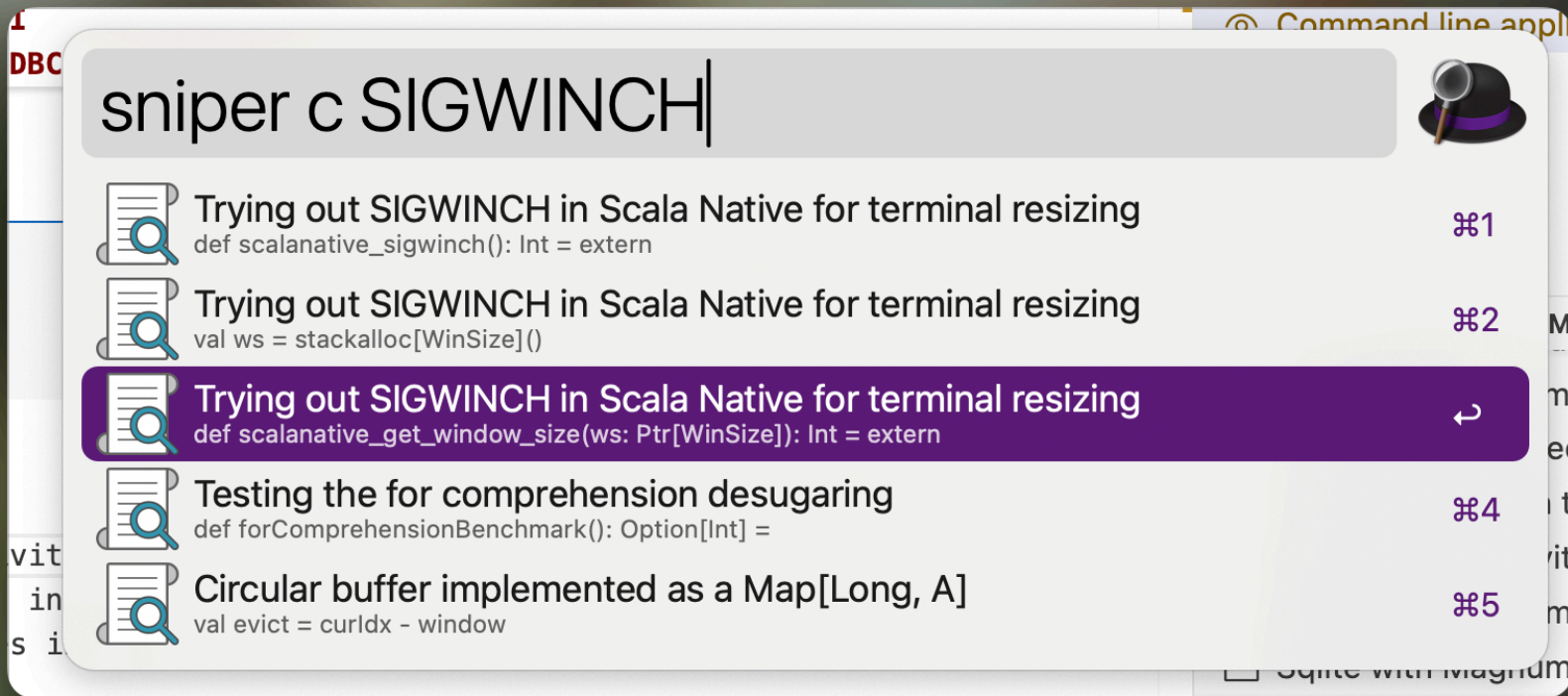
Alfred integration

- Alfred is the OG productivity app
- It has workflows that can invoke commands, and rudimentary JSON protocol
- It's JSON, so your choices in Scala are in the low hundreds

```
> sniper alfred prepare "n" | jq
{
  "items": [
    {
      "title": "scala3",
      "arg": "",
      "valid": false,
      "autocomplete": "new scala3 "
    },
    {
      "title": "c",
      "arg": "",
      "valid": false,
      "autocomplete": "new c "
    }
  ],
}
```

```
> sniper alfred prepare "n scala3 Hello, LSUG" | jq
{
  "items": [
    {
      "title": "Create snippet named `Hello, LSUG` using template `scala3`",
      "arg": "new:scala3:Hello, LSUG",
      "valid": true
    }
  ],
}
```


Alfred integration



Alfred integration

- All the logic is in the Sniper itself
- Alfred's workflow is a pretty static XML file
- Embed it!

```
//> using resourceDir ./resources  
//> using nativeEmbedResources
```

And then dump it directly out of the CLI:

```
sniper alfred workflow > Sniper.alfredworkflow && open Sniper.alfredworkflow
```

Alfred will open and install the workflow!

Model Context Protocol server with mcp

<https://github.com/indoorvivants/mcp>

```
//> using dep com.indoorvivants::mcp-quick::0.1.3

import mcp.*

@main def hello =
  val mcp = MCPBuilder
    .create()
    .handle(initialize): req =>
      InitializeResult(
        capabilities =
          ServerCapabilities(tools = Some(ServerCapabilities.Tools())),
        protocolVersion = req.protocolVersion,
        serverInfo = Implementation("scala-mcp", "0.0.1")
      )
    .run(SyncTransport.default)
end hello
```

To integrate with Claude, Cursor, etc.

A I recently did some C work in my snippets – what was it?

Use "snippets" to hint to Claude to use MCP

I'll check your snippets to see what C work you've been doing recently.

sniper_list

I can see a few snippets that involve C work. Let me search for C-related content to get more details:

sniper_code_search

Let me check the specific snippets that mention C files or C-related work:

sniper_read_file

sniper_read_file

sniper_read_file

sniper_read_file

Tool use

Based on your snippets, you've been working on two main C-related projects recently:

1. **Snippet #63: "Demonstrating union encoding with bindgen"** - This was for a Scala Native issue (#4339). You created a simple C header file (`test.h`) that demonstrates union encoding with a binary tree structure containing unions. The header defines:

- A `bwTRreeNode_t` struct with a single `uint8_t` field
- A `bwTree` struct with a union that can hold either a `uint64_t *` or `struct bwTRreeNode_t **`

Useful summary

Distribution

Distribution

- Distributing self-contained binaries is *very* easy compared to JVM applications
- Usually there are two problems:
 1. How to build for all platforms
 2. Where to put the binaries
 3. What to do with platforms that insist on dynamic linking and building from source.

Distribution

- Distributing self-contained binaries is *very* easy compared to JVM applications
- Usually there are two problems:
 1. How to build for all platforms
 2. Where to put the binaries
 3. What to do with platforms that insist on dynamic linking and building from source. **Put fingers in your ears, scream, and wait for them to go away**

Distribute via Github

- At the moment Github Actions is hard to beat
- Free runners for all major OSes and architectures
- Automatic releases, uploading binaries to release assets

Use my template to get started:

<https://github.com/indoorvivants/scala-native-binary-template>

Includes the entire release pipeline for all supported platforms.

Platform	Intel	Arm64
Linux	✓	✓
MacOS	✓	✓
Windows	✓	✗

Distribute via Github

Use my template to get started:

<https://github.com/indoorvivants/scala-native-binary-template>

v0.0.2

LatestCompare

keynmol released this 35 minutes ago



v0.0.2



0448326



Update ci.yml

▼ Assets

6

sniper-aarch64-apple-darwin	sha256:2ed399657158...	14.8 MB	4 minutes ago
sniper-aarch64-pc-linux	sha256:61d0850aeade...	15.9 MB	4 minutes ago
sniper-x86_64-apple-darwin	sha256:13c71f480e68...	14 MB	4 minutes ago
sniper-x86_64-pc-linux	sha256:97085c830992...	16.1 MB	4 minutes ago

Distribute via Coursier

- Coursier is a dependency manager for Scala that powers Mill, SBT, and Scala CLI
- It has a lesser known feature for distributing JVM and Native applications
- Your app *has* to be published to Maven Central or a compatible repository
- There are built-in registries at <https://github.com/coursier/apps/> for established applications
- You can define "channels" for your app distribution in JSON format

Distribute via Coursier

i.json

```
"sniper": {  
  "repositories": ["central"],  
  "dependencies": ["com.indoorvivants:sniper_native0.5_3:latest.stable"],  
  "launcherType": "scala-native",  
  "prebuiltBinaries": {  
    "x86_64-pc-linux": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-pc-linux.tar.gz",  
    "x86_64-pc-win32": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-pc-win32.tar.gz",  
    "x86_64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-apple-darwin.tar.gz",  
    "aarch64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-aarch64-apple-darwin.tar.gz",  
  }  
},
```

Deploy this file somewhere, e.g. `https://cs.indoorvivants.com/i.json`, and then run:

```
cs install sniper --channel
```

```
https://cs.indoorvivants.com/i.json
```

Distribute via Coursier

i.json

```
"sniper": {  
  "repositories": ["central"],  
  "dependencies": ["com.indoorvivants:sniper_native0.5_3:latest.stable"],  
  "launcherType": "scala-native",  
  "prebuiltBinaries": {  
    "x86_64-pc-linux": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-pc-linux.tar.gz",  
    "x86_64-pc-win32": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-pc-win32.tar.gz",  
    "x86_64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-apple-darwin.tar.gz",  
    "aarch64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-aarch64-apple-darwin.tar.gz",  
  }  
},
```

Deploy this file somewhere, e.g. `https://cs.indoorvivants.com/i.json`, and then run:

```
cs install sniper --channel
```

```
https://cs.indoorvivants.com/i.json
```

Distribute via Coursier

i.json

```
"sniper": {  
  "repositories": ["central"],  
  "dependencies": ["com.indoorvivants:sniper_native0.5_3:latest.stable"],  
  "launcherType": "scala-native",  
  "prebuiltBinaries": {  
    "x86_64-pc-linux": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-  
    "x86_64-pc-win32": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-  
    "x86_64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-x86_64-  
    "aarch64-apple-darwin": "https://github.com/indoorvivants/sniper/releases/download/v${version}/sniper-aa  
  }  
},
```

Deploy this file somewhere, e.g. `https://cs.indoorvivants.com/i.json`, and then run:

```
cs install sniper --channel
```

```
https://cs.indoorvivants.com/i.json
```

Distribute via homebrew

- Homebrew is the de-facto package manager for MacOS, and is gaining traction on Linux
- It has native support for "taps" – specially named Github repositories where you can put your "formulas"
- Mine is at <https://github.com/indoorvivants/homebrew-tap>

```
brew install indoorvivants/tap/sniper
```

Distribute via homebrew

- Formulas are defined in Ruby
- We can use pre-defined binaries, or build from source
- You can run the formula directly `brew install ./sniper.rb && brew test sniper`

```
class Sniper < Formula
  desc "Snippet management tool"
  # ...
  resource "binary" do
    on_arm do
      on_macos do
        url "https://github.com/keynmol/sniper/releases/download/v#{_version}/sniper-aarch64-#{_version}.tar.gz"
      end
    end
    on_linux do
      url "https://github.com/keynmol/sniper/releases/download/v#{_version}/sniper-aarch64-#{_version}.tar.gz"
    end
  end
end
```


Closing thoughts

It's still Scala after all

- It is remarkable that the entire project could well be a Scala JVM application
- Just change some dependencies, and remove the single "native" line:

```
val isTTY = scalanative.posix.unistd.isatty(1) == 1
```

- It is possible to do **so much** with the libraries and JDK implementations that are already available
- The ecosystem is rich, growing, and yearns for users

Conclusion

- The best time to start building CLIs in Scala Native is now
- Don't sit around waiting for something to be production ready – you are the production, you make that call
- Ecosystem is rich, growing, and yearns for users
- Scala Native itself desperately needs contributors:
 - Make optimiser much, much faster
 - We need incremental builds
 - JDK classes missing (VERY IMPORTANT NOTE)

Happy building!

- **Scala Native** <https://scala-native.org/en/stable/>
- **Sniper** <https://github.com/indoorvivants/sniper>
- **Blog** <https://blog.indoorvivants.com/>
- **Slides** <https://slides.indoorvivants.com/clis-with-scala-native>

Ask your company to sponsor Scala Center:

<https://scala.epfl.ch/corporate-membership.html>