Exploring the Scala's Tooling Ecosystem

choose your ~~poison~~ mithridate

What are tools used for?

Editing, compiling and testing.

What are tools used for?

Editing, compiling and testing, ...

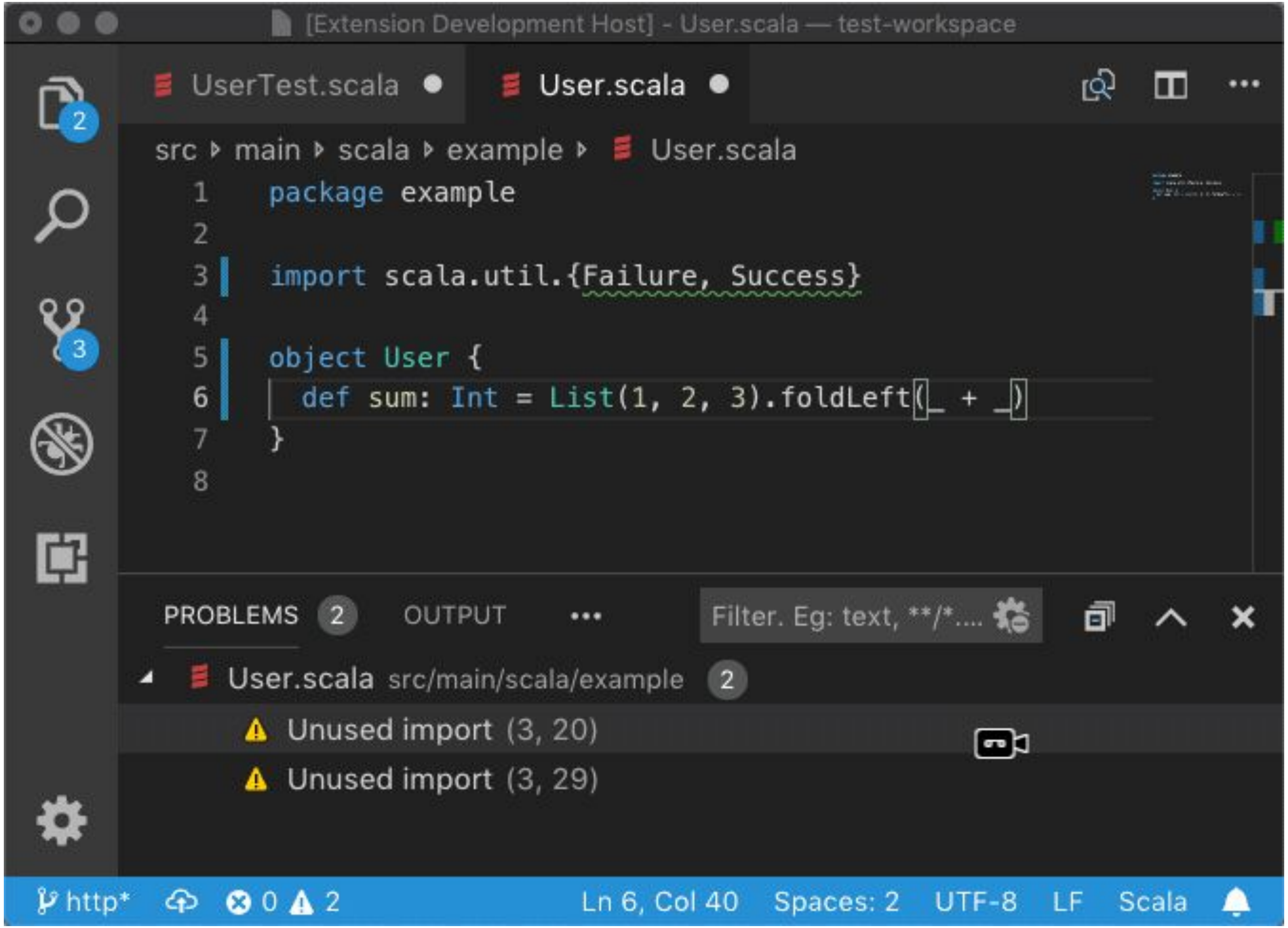
learning, experimenting, analysing, reviewing code, debugging, dependency management, publishing, versioning, benchmarking, creating other tools, ...

coffee, ...

why do we care ?

why do we care ?

Productivity.



[Extension Development Host] - User.scala — test-workspace
UserTest.scala
User.scala
src › main › scala › example › User.scala
PROBLEMS 2
OUTPUT
Filter. Eg: text, **/*....
User.scala src/main/scala/example 2

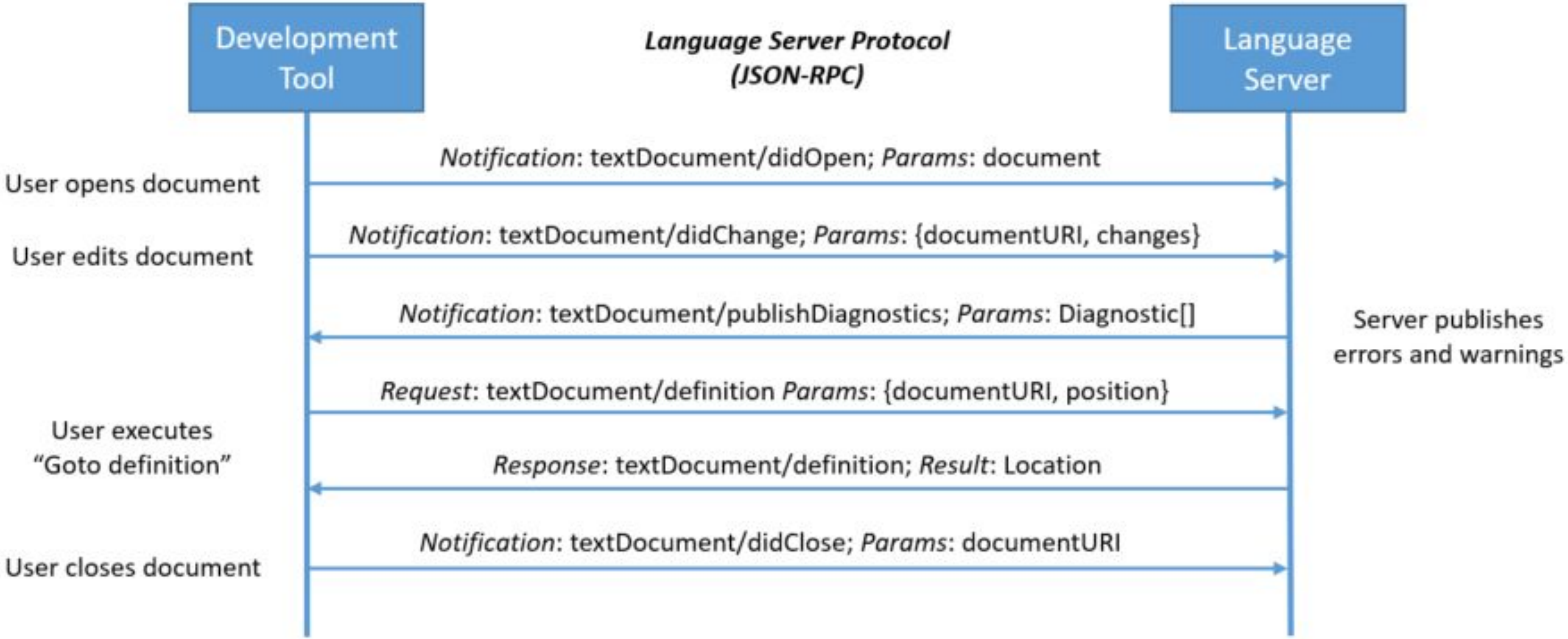
```scala
package example

import scala.util.{Failure, Success}

object User {
  def sum: Int = List(1, 2, 3).foldLeft(_ + _)
}

```

```
⚠ Unused import (3, 20)
⚠ Unused import (3, 29)
```

```
http*   ⊗ 0 ⚠ 2   Ln 6, Col 40   Spaces: 2   UTF-8   LF   Scala
```

LSP: Language Server Protocol

The Language Server Protocol defines the protocol used between an editor or an IDE and a language server that provides language features like auto complete, go to definition, find all references, etc.



Development Tool

Language Server Protocol (JSON-RPC)

Language Server

User opens document

Notification: textDocument/didOpen; *Params*: document

User edits document

Notification: textDocument/didChange; *Params*: {documentURI, changes}

Notification: textDocument/publishDiagnostics; *Params*: Diagnostic[]

Server publishes errors and warnings

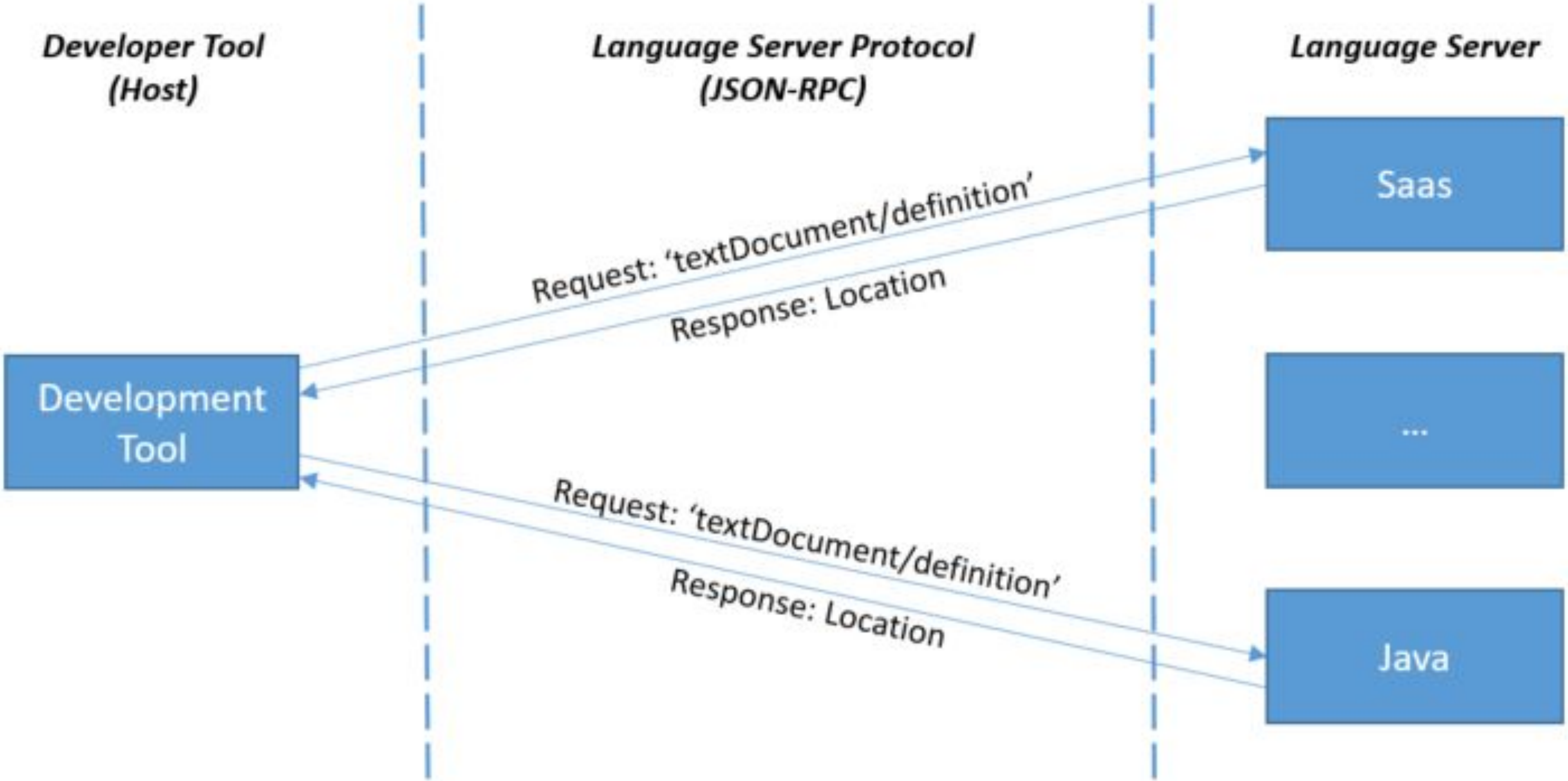
Request: textDocument/definition *Params*: {documentURI, position}

User executes "Goto definition"

Response: textDocument/definition; *Result*: Location

User closes document

Notification: textDocument/didClose; *Params*: documentURI


Developer Tool
(Host)
Language Server Protocol
(JSON-RPC)
Language Server
Saas
Request: 'textDocument/definition'
Response: Location
Development Tool
...
Request: 'textDocument/definition'
Response: Location
Java

Language Servers in Scala lang

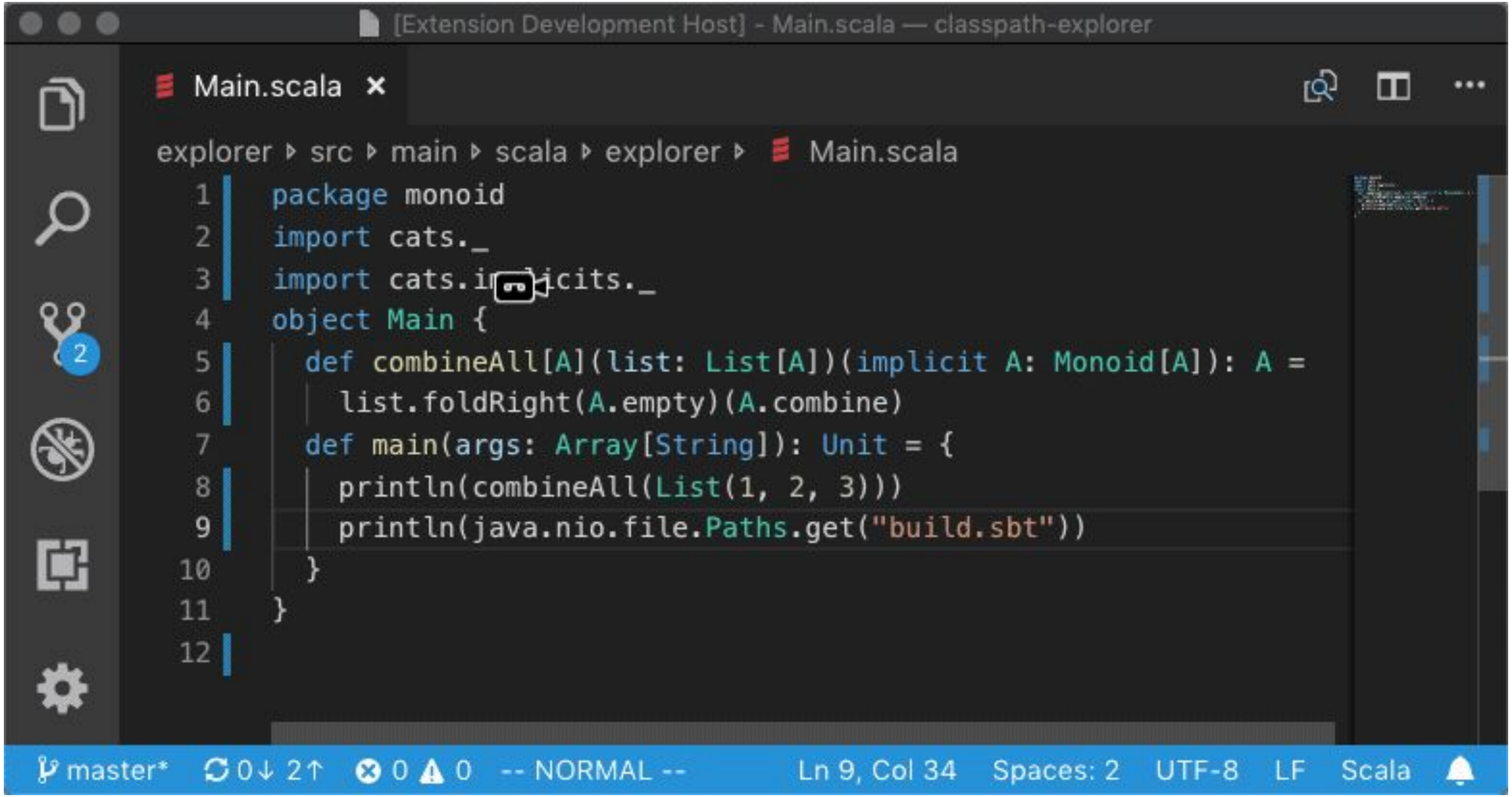
- Dragos-vscode-scala
- Metals
- Dotty LS
- SBT server (since SBT 1.x)
- Ensime

Editors/IDEs supporting the LSP

- IntelliJ
- Eclipse IDE
- VS code
- Atom
- Sublime Text 3
- vim8/neovim
- Emacs

See ls.org and ms.lsp for the latest on adoption.

Metals @scalameta


[Extension Development Host] - Main.scala — classpath-explorer
Main.scala ×
explorer › src › main › scala › explorer › Main.scala
2
master*
0↓ 2↑
0 ⚠ 0
-- NORMAL --
Ln 9, Col 34
Spaces: 2
UTF-8
LF
Scala

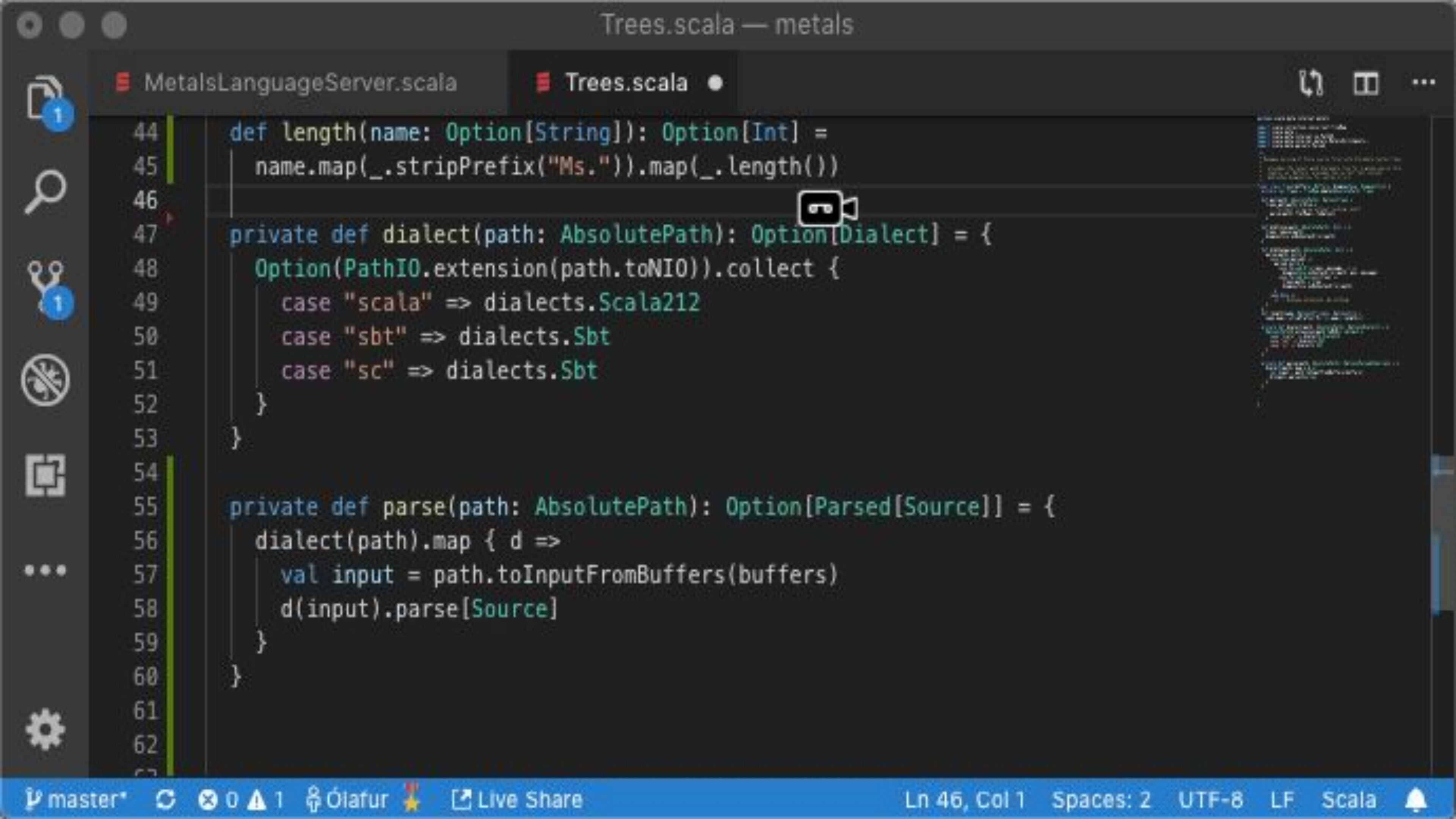
```scala
package monoid
import cats._
import cats.implicits._
object Main {
  def combineAll[A](list: List[A])(implicit A: Monoid[A]): A =
    list.foldRight(A.empty)(A.combine)
  def main(args: Array[String]): Unit = {
    println(combineAll(List(1, 2, 3)))
    println(java.nio.file.Paths.get("build.sbt"))
  }
}

```

Metals

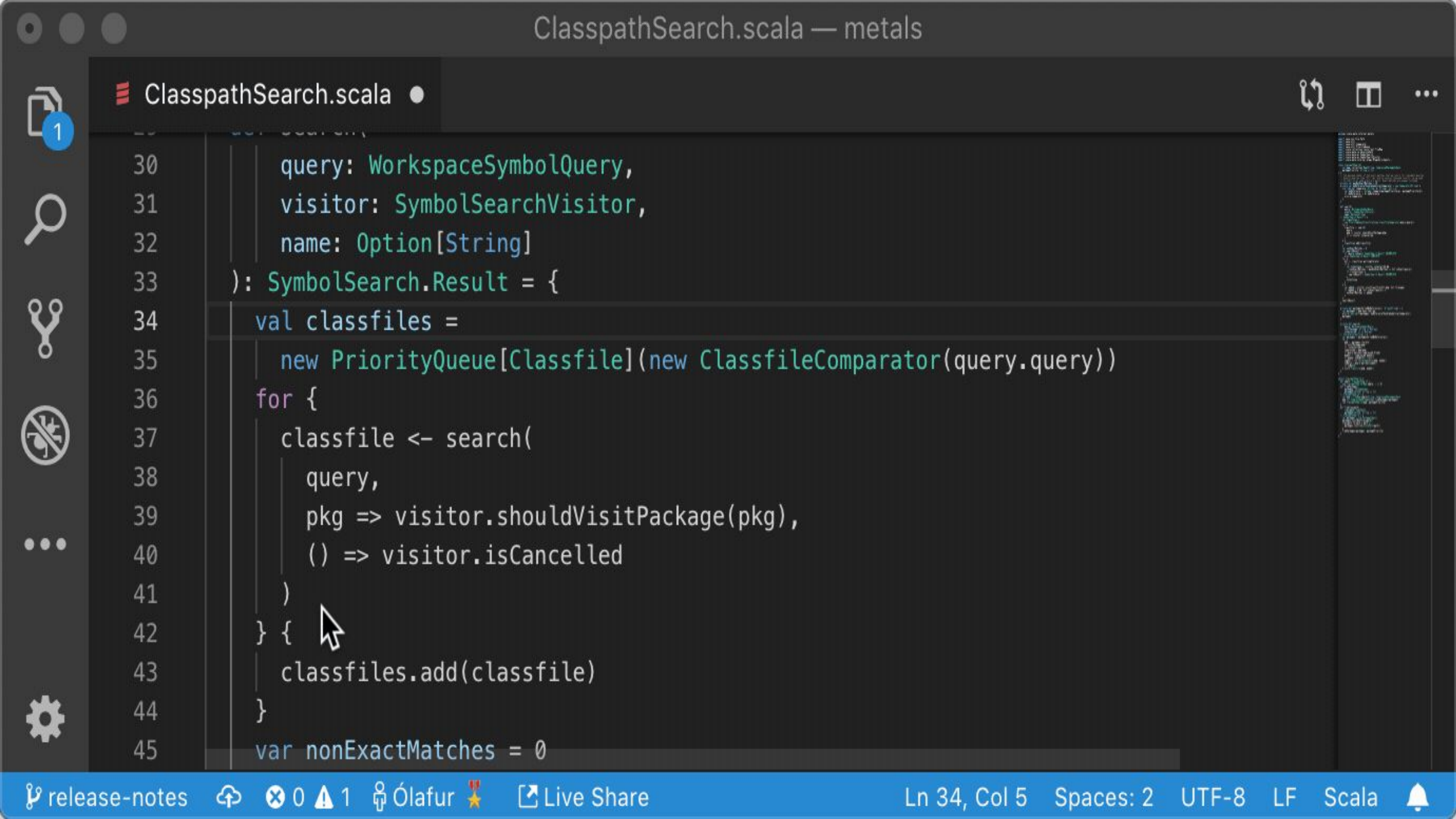
Works with: vs-code, atom, vim, Sublime text 3, Emacs.

Uses BSP to communicate LS and build tools.

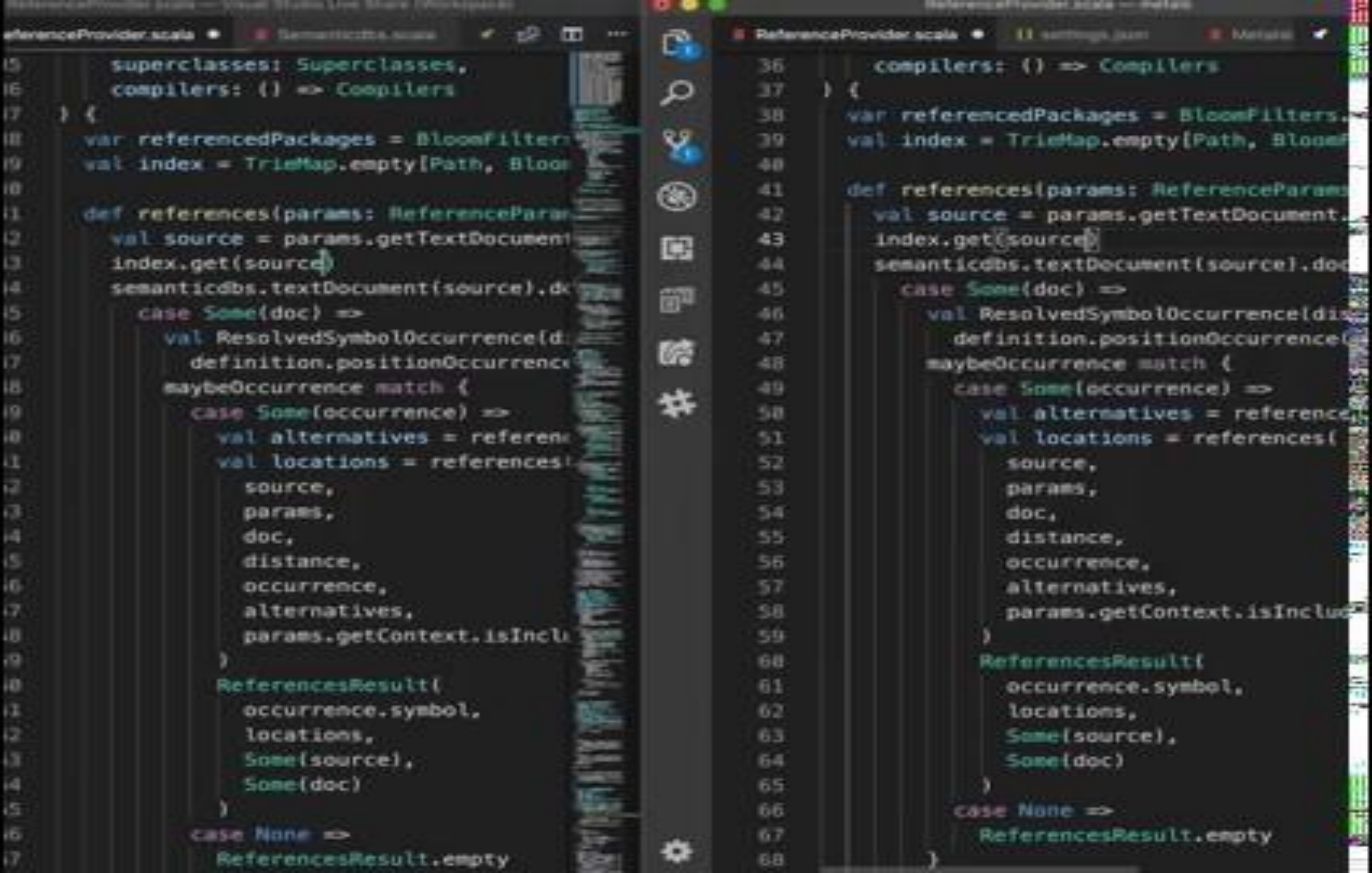
> *"Our goal in Metals is to support code navigation with as low CPU and memory overhead as possible without sacrificing rich functionality."*



```
  def length(name: Option[String]): Option[Int] =
    name.map(_.stripPrefix("Ms.")).map(_.length())

  private def dialect(path: AbsolutePath): Option[Dialect] = {
    Option(PathIO.extension(path.toNIO)).collect {
      case "scala" => dialects.Scala212
      case "sbt" => dialects.Sbt
      case "sc" => dialects.Sbt
    }
  }

  private def parse(path: AbsolutePath): Option[Parsed[Source]] = {
    dialect(path).map { d =>
      val input = path.toInputFromBuffers(buffers)
      d(input).parse[Source]
    }
  }
```



```scala
    query: WorkspaceSymbolQuery,
    visitor: SymbolSearchVisitor,
    name: Option[String]
): SymbolSearch.Result = {
  val classfiles =
    new PriorityQueue[Classfile](new ClassfileComparator(query.query))
  for {
    classfile <- search(
      query,
      pkg => visitor.shouldVisitPackage(pkg),
      () => visitor.isCancelled
    )
  } {
    classfiles.add(classfile)
  }
  var nonExactMatches = 0
```


ReferenceProvider.scala — Visual Studio Live Share (Workspace)
eferenceProvider.scala
Semanticdbs.scala
ReferenceProvider.scala — metals
ReferenceProvider.scala
settings.json
Metals

```scala
      superclasses: Superclasses,
      compilers: () => Compilers
  ) {
    var referencedPackages = BloomFilter
    val index = TrieMap.empty[Path, Bloom
    
    def references(params: ReferencePara
      val source = params.getTextDocument
      index.get(source)
      semanticdbs.textDocument(source).do
        case Some(doc) =>
          val ResolvedSymbolOccurrence(d
            definition.positionOccurrenc
          maybeOccurrence match {
            case Some(occurrence) =>
              val alternatives = referen
              val locations = references(
                source,
                params,
                doc,
                distance,
                occurrence,
                alternatives,
                params.getContext.isIncl
              )
              ReferencesResult(
                occurrence.symbol,
                locations,
                Some(source),
                Some(doc)
              )
            case None =>
              ReferencesResult.empty
```

```scala
      compilers: () => Compilers
  ) {
    var referencedPackages = BloomFilters.
    val index = TrieMap.empty[Path, BloomF
    
    def references(params: ReferenceParams
      val source = params.getTextDocument.
      index.get(source)
      semanticdbs.textDocument(source).doc
        case Some(doc) =>
          val ResolvedSymbolOccurrence(dis
            definition.positionOccurrence(
          maybeOccurrence match {
            case Some(occurrence) =>
              val alternatives = reference
              val locations = references(
                source,
                params,
                doc,
                distance,
                occurrence,
                alternatives,
                params.getContext.isInclud
              )
              ReferencesResult(
                occurrence.symbol,
                locations,
                Some(source),
                Some(doc)
              )
            case None =>
              ReferencesResult.empty
          }
```

SemanticDB

- Portable semantic APIs
- Data schema for semantic information about code.
- Relevant for developer tools.
- Protobuf, JSON, SQL.
- Do not require a running compiler.

BSP: Build Server Protocol

- Extends LSP
- Formalizes interaction between editors, LS, editors, build tools
- Bidirectional notifications,
- Client/server architecture.
- Clients are build tools, LS and editors/IDEs.
- Language agnostic.
- JSON-RPC-protobuf.

Bloop

- sbt, Maven, Gradle and Mill.
- IntelliJ and Metals (VS Code, Sublime, vim and Atom)
- Provides fast compile, test and run
- Has a built-in command-line tool
- Integrates with most JVM build tools
- Supports JVM, Scala.js and Scala Native

CBT: Chris' Build Tool

- Quite simple (even the source code!)
- Builds are programs (in Scala)
- JVM method invocation
- Reproducible builds
- Command line and shell integration
- Aims to make it easy for devs to be in control.
- Built-in Dotty support (experimental)

Mill

- Build tools as Pure functional programs
- Fast
- Flexible
- Targets with caching
- Inspectale cache graph
- Commands

Pants

- Source based dependency tool
- Aims to large code bases
- Monorepo environments
- Fast
- Scalable
- dependency inference from source-code imports

Bazel

- Reproducible builds
- Fast and correct builds
- Heavy caching
- Enforces build correctness
- Scalable
- Multi-language

Summary

Tools that work with my editor => LSP

Build integrations => BSP

Navigation => SemanticDB

Refactoring => scalafix

Diagnostics => BSP + LSP

Additional links...

https://www.scala-lang.org/blog/2018/02/14/tooling.html

https://github.com/scalacenter/tooling-working-groups/blob/master/meetings/2018-01-17/minutes.md

https://scala.sphere.it/#agenda

https://contributors.scala-lang.org/c/tooling

https://www.scala-lang.org/contribute/tools.html

https://scala.epfl.ch/projects.html

https://microsoft.github.io/language-server-protocol/

https://github.com/Microsoft/language-server-protocol

¡Gracias!

@unyagami