



How We Built Tools That Scale to Millions of Lines of Code

Eugene Burmako
Twitter, Inc.

6/20/2018



About me

- Founder of Scala macros, Scalameta and Rsc
- Member of the Scala Improvement Process committee
- PhD from Martin Odersky's lab at EPFL (2011-2016)
- Tech lead of the Advanced Scala Tools team at Twitter (2017-present)



Credits



Core contributors

Advanced Scala Tools team at Twitter:

- Eugene Burmako
- Shane Delmore
- Uma Srinivasan



Early adopters

- Build team
- Continuous Integration team
- Code Review team
- Core Data Libraries team
- Core Systems Libraries team
- Other folks at Twitter

Code

Issues 136

Pull requests 0

Insights

Settings

Pulse

Contributors

Community

Traffic

Commits

Code frequency

Dependency graph

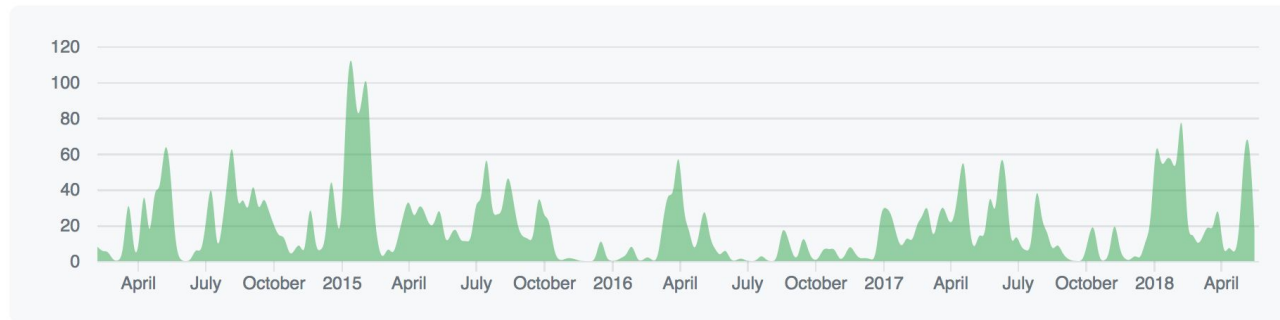
Network

Forks

Mar 2, 2014 – Jun 16, 2018

Contributions: Commits

Contributions to master, excluding merge commits



 **xeno-by** #1
 3,275 commits **441,055 ++** **420,749 --**

This line chart shows the commit activity for the user xeno-by. The y-axis ranges from 0 to 100, with a grid line at 50. The x-axis shows months from April to April. The chart displays an orange area representing the number of commits. There are several peaks, with the highest one occurring in late 2014, reaching approximately 110. Other significant peaks occur in early 2015, late 2016, and early 2018.

 **olafurpg** #2
 648 commits **113,877 ++** **86,816 --**

This line chart shows the commit activity for the user olafurpg. The y-axis ranges from 0 to 100, with a grid line at 50. The x-axis shows months from April to April. The chart displays an orange area representing the number of commits. There are several peaks, with the highest one occurring in late 2014, reaching approximately 50. Other significant peaks occur in early 2015, late 2016, and early 2018.



Problem statement



Huge codebase (ca. 2017)

- $\sim 2^{25}$ lines of human-written code
- $\sim 2^{16}$ targets



Need for semantic tooling (ca. 2017)

- Not enough to treat programs like text
- Need to understand semantics:
 - What does this identifier resolve to?
 - What are all the usages of this definition?
 - What is the type of this expression?
 - Etc etc.



Prioritized user asks (ca. 2017)

- Code browsing
- Code review
- Code evolution



State of semantic tooling (ca. 2017)

- Code browsing = IDEs, but IDEs couldn't load entire Twitter source
- Code review = Phabricator, which didn't have Scala integration
- Code evolution = scala-refactoring, which didn't have a maintainer
- Also, several proprietary solutions with varied Scala support



Advanced Scala Tools team

- Founded in June 2017
- Mission: “Raise the bar on what is possible for an effective Scala development environment both at Twitter and in the Scala community”
- Roadmap: improve code browsing, code review and code evolution in the Twitter development workflow



Existing semantic APIs



Existing semantic APIs (ca. 2017)

- Scala compiler internals
- Scala.reflect (thin wrapper over compiler internals)
- ScalaSignatures (serialization format for compiler internals)



Blocker #1: Learning curve

- Compiler internals span dozens of modules and thousands of methods
- Complicated data model and arcane preconditions for the APIs
- I did a PhD in Scalac internals, but still can't make sense of all that



Blocker #2: Scarce documentation

- Scala requires an extensive semantic API
- This requires lots and lots of documentation
- Even for `scala.reflect`, the documentation is significantly lagging behind



Blocker #3: Compiler instance

- Compiler internals require a compiler instance
- This means poor performance even for simple operations like “Go to definition” or “Find all usages”
- Tools that use Scala compiler internals either roll their own indexer or accept the limitations



Future semantic APIs



Future semantic APIs (ca. 2020)

- Scala.reflect is based on Scala compiler internals, so it was discarded
- Meet Tasty - serialization format for Dotty compiler internals
- Used in Dotty IDE and the upcoming Dotty macro system



library/src/scala/tasty/Tasty.scala

```
abstract class Tasty {  
  ...  
  
  // DefDef  
  type DefDef <: Definition  
  implicit def defDefClassTag: ClassTag[DefDef]  
  val DefDef: DefDefExtractor  
  
  ...  
}
```



library/src/scala/tasty/Universe.scala

```
trait Universe {  
  val tasty: Tasty  
  implicit val context: tasty.Context  
}
```

```
object Universe {  
  implicit def compilationUniverse: Universe = throw new  
  Exception("Not in inline macro.")  
}
```



compiler/.../CompilationUniverse.scala

```
import dotty.tools.dotc.core.Contexts.Context

class CompilationUniverse(val context: Context) extends
scala.tasty.Universe {
  val tasty: TastyImpl.type = TastyImpl
}
```



Summary

- In its current form, Tasty looks very similar to `scala.reflect`, but reimplemented for Dotty
- Still based on compiler internals
- Still underdocumented
- **Still requires a compiler instance**



Rolling our own semantic APIs



Scalameta (ca. 2013)

- Open-source metaprogramming library
- Created almost 5 years ago during my time at EPFL
- Focused on tool writers



Scalameta (ca. 2018)

- More than 10 projects
- More than 10000 commits
- More than 200 contributors
- Funded by Twitter and Scala Center



SemanticDB

- Data model for semantic information about programs
- Focused on what tool writers need from the compiler...
- ...not on what is convenient to expose in the compiler
- Collaboration between Eugene Burmako (a compiler writer) and Ólafur Páll Geirsson (a tool writer)



Interchange format

```
message TextDocument {  
  Schema schema = 1;  
  string uri = 2;  
  string text = 3;  
  Language language = 10;  
  repeated SymbolInformation symbols = 5;  
  repeated SymbolOccurrence occurrences = 6;  
  repeated Diagnostic diagnostics = 7;  
  repeated Synthetic synthetics = 8;  
}
```



Example

```
object Test {  
  def main(args: Array[String]): Unit = {  
    println("hello world")  
  }  
}
```



Workflow

```
$ scalac -Xplugin:our/plugin.jar Test.scala  
// Alternatively: metac Test.scala
```

```
$ find .  
./META-INF  
./META-INF/Test.scala.semanticdb  
./Test.scala
```



Payload

```
$ xxd META-INF/semanticdb/Test.scala.semanticdb
00000000: 0ae4 0408 0312 0a54 6573 742e 7363 616c .....Test.scal
00000010: 611a 596f 626a 6563 7420 5465 7374 207b a.Yobject Test {
00000020: 0a20 2064 6566 206d 6169 6e28 6172 6773 . def main(args
00000030: 3a20 4172 7261 795b 5374 7269 6e67 5d29 : Array[String])
00000040: 3a20 556e 6974 203d 207b 0a20 2020 2070 : Unit = {      p
00000050: 7269 6e74 6c6e 2822 6865 6c6c 6f20 776f rprintln("hello wo
00000060: 726c 6422 290a 2020 7d0a 7d0a 2a5b 0a1a rld").  }.}*[..
00000070: 5f65 6d70 7479 5f2e 5465 7374 2e6d 6169 _empty_.Test.mai
00000080: 6e28 292e 2861 7267 7329 1808 2a04 6172 n().(args)..*.ar
...
```



Payload

```
$ metap .
```

Summary:

```
Schema => SemanticDB v3
```

```
Uri => Test.scala
```

```
Text => non-empty
```

```
Language => Scala
```

```
Symbols => 3 entries
```

```
Occurrences => 7 entries
```




Symbols

```
_empty_.Test. => final object Test extends AnyRef { +1 decls }  
_empty_.Test.main(). => method main(args: Array[String]): Unit  
_empty_.Test.main().(args) => param args: Array[String]
```



Occurrences

```
[0:7..0:11): Test <= _empty_.Test.  
[1:6..1:10): main <= _empty_.Test.main().  
[1:11..1:15): args <= _empty_.Test.main().(args)  
[1:17..1:22): Array => scala.Array#  
[1:23..1:29): String => scala.Predef.String#  
[1:33..1:37): Unit => scala.Unit#  
[2:4..2:11): println => scala.Predef.println(+1).
```



To learn more

- Check out “SemanticDB for Scala developer tools” by Ólafur Páll Geirsson (ScalaSphere 2018)
- Detailed examples of SemanticDB payloads
- Introduction to CLI utilities to work with SemanticDB
- Overview of existing tools based on SemanticDB



Rolling our own semantic tools



Opensource tools

- Metadoc (code browsing)
- Metals (code browsing and interactive development)
- Scalafix (code linting and refactoring)

Developed by Ólafur Páll Geirsson and a community of opensource contributors based on Scalameta



Company-wide semantic index

- SemanticDB doesn't require a compiler instance
- Therefore can be made extremely fast even on huge codebases
- SQLite indexes take ~500Mb per 1Mloc and provide ~10ms query times
- Using different storage technology at Twitter, with similar characteristics



Company-wide language server

- Experimental LSP implementation backed by the semantic index
- Implements `textDocument/definition` and `textDocument/references`



Code browsing

- Experimental IntelliJ IDEA plugin with custom “Go to definition” and “Find references” powered by the company-wide language server
- Finally, an IDE that can handle the entire Twitter source



Code review

- Upstream improvements to DiffusionExternalSymbolsSource to take source positions into account
- Experimental implementation of a symbol source powered by the company-wide language server



Code evolution

- Upstream Scalafix, closely following cutting edge milestone builds
- Distributed Scalafix to run code rewrites across the entire Twitter source
- To learn more, check out “Scalafix @ Twitter scale” by Uma Srinivasan (Typelevel Summit Boston 2018)



Summary



Summary

- Advanced Scala Tools team was founded to improve code browsing, code review and code evolution in the Twitter development workflow
- We use SemanticDB - an opensource interchange format for semantic information developed by Eugene Burmako and Ólafur Páll Geirsson
- We have implemented experimental improvements to multiple areas of interest, integrating opensource and closed-source solutions



We are hiring!

- Are you interested in compilers and developer tools?
- Are you ready to get your hands dirty to make things happen?
- Drop Eugene Burmako an email: eburmako@twitter.com