# Scala Center updates, Q1 2019 Advisory Board Meeting

Scala Center team: Jorge Vicente Cantero, 100%; Ólafur Geirsson, 100%; Julien Richard-Foy, 50%; Alexandre Archambault, 100%; Darja Jovanovic, 100%; Sébastien Doeraene, 100%

## Initiatives worked on this cycle:

# At a glance

- coursier
- Bloop
- MOOCs
- Documentation
- Metals
- mdoc
- Scala.js
- Scala Days 2019 organisation
- SIP meetings

## coursier

@alexarchambault

coursier is a library to manage dependencies from Maven and Ivy repositories. It comes along with a CLI to conveniently interact with dependencies, and an sbt plugin, for coursier to replace Ivy in sbt.

We added a high-level API in coursier, aiming at being as simple to use as the CLI of coursier. We made adjustments to the coursier-based sbt launcher, so that it's ready for prime time. In the context of SCP-20, we're adding a "rules" mechanism to coursier,

allowing similar guards as the strict conflict manager of Ivy or the Maven enforcer plugin, and extending them in a more powerful way.

## High-level API

The high-level API aims at being as simple to use and featureful as the CLI of coursier, while retaining all the capabilities of the former low-level API. Plenty of features from the CLI, the sbt plugins, and the tests of coursier were moved to it, simplifying their code, and illustrating that the high-level API is powerful enough for advanced use cases.

Example of use

```
import coursier._
val a = Fetch()
  .addDependencies(dep"", dep"")
  .addRepository()
  .run()
```

`Resolve()` is also available along `Fetch()`. Both `Resolve()` and `Fetch()` have numerous methods allowing to adjust some or all resolution parameters. Methods like `future` or `io` can be called instead of `run` to run resolutions and downloads in the background.

The high-level API also makes it easier to add features to coursier, and have them be available straightaway in the CLI, the sbt plugins, and to users using the API. Rules described below, but also, in the future, more robust credential management or support for mirror repositories, are added with minimum overhead thanks to this.

The high-level API is already available in coursier `1.1.0-M12`. Expect a wider announcement soon.

## sbt launcher

coursier has had its own sbt launcher for some time already, able to fetch and run sbt, like the launcher of sbt itself. With its launcher, coursier resolves and downloads sbt on its own, which significantly speeds up its start up when an sbt version is run for the first time.

We added more tests to it, and made a few things more robust in it (related to former versions of sbt-coursier depending on things set up by the original sbt launcher, in particular).

A modified version of sbt-extras allows to run sbt as simply as before, but relying on the coursier-based launcher.

Nice speed up were seen on the Appveyor CI of coursier, which doesn't cache things between runs, and thus benefits from a faster initial start up of sbt (job durations going from 10-13 minutes to 8-10 minutes).

Expect a wider announcement soon (maybe after some extra minor adjustments in the output of the launcher).

## Resolution rules

In the context of SCP-20, we're adding a "rules" mechanism to coursier, allowing to enforce some constraints during or after resolutions. These rules can be

- enforcing some or all modules to have no evicted versions (all their dependees must explicitly depend on the selected version), like the strict conflict manager of Ivy does,
- require several modules to all have the same version, like it should be the case of the jackson modules,
- require some or all root dependencies not to have their version be bumped to a higher version,
- require some or all modules not to have a snapshot version,
- etc.

Optionally, some of these rules can try to automatically address the issue, like the second or third ones above.

Some of these rules are already implemented, and the high-level API of coursier already allows to use those. The CLI of coursier also accepts rules.

We'd like to allow to read those rules in external JSON files very soon, and have the sbt plugin of coursier accept rules, either via settings, and/or by reading a predefined file at the root of the project (`rules.json`?). An official announcement should follow right after that, to gather early feedback.

## General miscellaneous

@alexarchambault

- We discussed almond, our Scala kernel for Jupyter, and its internals at the Notebook Enterprise Summit, an enterprise-focused Jupyter conference.
- We did plenty of maintenance around almond, like
  - simplifying its run-from-sources instructions,
  - automatically publishing docker images for it,
  - automatically running and validating some example notebooks on its CI,
  - reworking the landing page of its website.
- We optimized some aspects of the resolution in coursier (parsing POM files with a SAX parser, replacing some regular expressions with optimized hand-crafted code).

## Bloop

@jvican

Bloop is a build server that provides fast compile, test and run capabilities to clients. This last quarter, we released `v1.1.1`, `v1.1.2`, `v1.2.0`, `v1.2.1`, `v1.2.2`, `v1.2.3`, `v1.2.4` and `v1.2.5`. Highlights of these versions include a bloop launcher, reliable compiler cancellation, cascade compilation, incremental reporting of compiler diagnostics via BSP, changes in CLI user ergonomics, better documentation and new installation options for Windows.

### Upcoming features

For the past two months, we have been working towards Bloop `v1.3.0` which includes state-of-the-art compiler-related improvements and aim to guarantee a safe usage of the build server to several clients.

## Reliable client isolation

Bloop `v1.3.0` will enable clients to use Bloop's build server with the certainty that none of the actions performed by other concurrent clients will interfere with their requests. This feature is called client isolation and is critical for correctness.

For example, picture two clients of a build server: one is a build tool and another one is an IDE. If the IDE starts to run an application and the build tool requires a concurrent compilation request, the JVM application will most likely crash when the second compilation changes class files in the classes directory, because the JVM lost the open file pointers to the loaded classes.

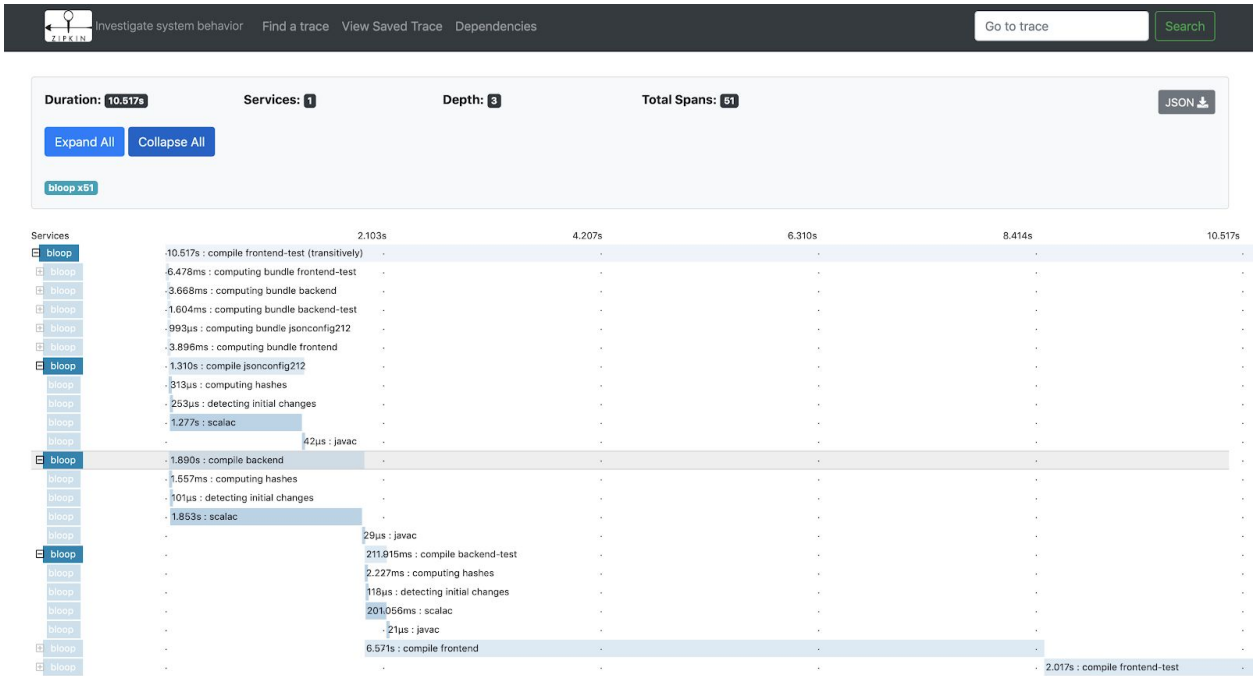
To achieve client isolation, Bloop returns to every client a set of unique compilation outputs (directories with any kind of compilation product). This operation has been designed to be as fast as possible.

## Deduplicated compilation

If two compilation requests happen concurrently, Bloop will compile the target only once if it can prove that the compilation inputs are the same. This reduces the time the build server compiles target significantly.

## Zipkin integration

Every compilation request can now be analyzed via Zipkin, a distributed tracing system. The traces registered in the system provide valuable insight to users to analyze where the bottleneck of their build compilation performance is.



## Faster incremental and no-op compiles

Bloop has now a compiler architecture that allows it to push as much IO work as possible to the background and perform it in parallel. Consequently, the cost of no-op compiles is now 5 times lower than before, while the price for incrementally compiling single files has halved. For example, Bloop can no-op compile the bloop build (with up to 32 targets) in 200ms and the `guardian/frontend` build (with ~15 targets and > 60.000 lines of code) in 450ms. The performance is not linear anymore because the more IO work there is, the more parallelism level we apply.

# MOOCs

@julienrf

We have deployed, tested, and launched the Programming Reactive Systems course on the edX.org platform!

We have started updating the content of the Functional Program Design course to make it more self-contained and consistent.

# Documentation

We have added a documentation page giving library authors a list of recipes to follow to automate boring things like running tests on pull requests using a continuous integration server, publishing releases, publishing online documentation or checking binary compatibility between minor releases.

We have added a documentation page showing how to enrich the collections that are part of the standard collections framework with user-defined operations.

# Metals

@olafurpg

Metals is a Scala language server that provides advanced code editing and code navigation support in Visual Studio Code, Vim, Sublime Text, Emacs and Atom. We released Metals v0.4 with several new features and bug fixes, see release notes for v0.4.0 and v0.4.4. Highlights of the v0.4 release include find symbol references, fuzzy

symbol search, code formatting with Scalafmt, document symbol outline and syntax errors as you type.

## Low-memory symbol indexing with bloom filters

One goal of Metals is to use little memory even when analyzing large codebases. We wrote a blog post on low-memory symbol indexing with bloom filters that explains the techniques Metals uses to enable fast response times for features like fuzzy symbol search and find symbol references while keeping memory usage low.

## Upcoming: code completions, parameter hints and show type

For the past four weeks, we have been working towards Metals v0.5 that adds several new features including code completions, parameter hints and show type at point. These features have been requested by many Metals users and we look forward to releasing them. There is a work-in-progress pull request that already implements a lot of functionality but work remains to stabilize the integration with the Scala presentation compiler before v0.5 can be released.

## mdoc

@olafurpg

The websites for Metals, Scalafix, Bloop, Coursier and Almond use mdoc, a documentation tool that evaluates Scala code examples in markdown files. In January, we released several improvements to mdoc that include a new sbt plugin with Docusaurus support and a Scala.js integration that enables library authors to write interactive documentation. See the release notes for v1.0.0, v1.1.0, v1.1.1 and v1.2.4. We also wrote a blog post fast typechecked markdown documentation with clear error messages that explains the techniques mdoc uses to speed up documentation generation by up to 27x compared to tut, an mdoc alternative.

## Scala.js

@sjrd

We mostly worked on general maintenance of Scala.js, notably with respect to the upcoming Scala 2.13.0-RC1: Scala.js is now green in the Scala community build for 2.13.

We also kept working on Scala.js 1.x.

## Scala Days 2019 organisation

@darjutak

Scala Center is organising the ScalaDays 10th edition, June 11-13 2019 at EPFL Switzerland, with crucial support and help from Lightbend and Dajana Günther, CEO at Trifork Germany GmbH.

### Published blogs

Blog detailing our vision published in January 2019, Scala Days 2019 - Celebrating Collaborative Success.

Blog asking for community's input for Phil Bagwell Award published in March 2019, Phil Bagwell Nominations. In only 2 days, we received ~70 recommendations.

### Milestones reached (in this quarter):

6. CfP closed, 250 submissions (in 2018 we received ~240 submissions, for both NA and EU editions; 2019 has only EU edition and more submissions than last year)

7. Four Platinum sponsorships signed, 10 Gold sponsors signing, ~20 in negotiations

7.1 First Scala spree Switzerland organised, 15th December

- ➢ At EPFL, 43 signups, 40 showed up (impressive show-up rate)
- ➢ We welcomed Scala developers from various cities in Switzerland on a last Saturday before Christmas

- ➢ We circulated a feedback form, and learned that lineup and format of an event are the most important for people to decide to participate and actually show up.
- ➢ Check out some cool photos

8. Program published 4th March 2019

- ➢ 48 speakers, 2 Keynote speakers, Community panel, and Diversity panel

9. Registration opened 22nd February 2019

- ➢ 15% tickets sold in the first week of opening

## Other

We've been contacted by SUG Organisers from different cities to help organise a special event for active group leaders during the Scala Days. This proposal turned into developing a great relationship between the Scala Center and organisers, as well as between the organisers themselves. Amongst other things, they will be a part of our diversity outreach. I've also explained how to create a proposal to the Advisory Board, as they were wondering how to do it.

## SIP Meetings

@darjutak and @sjrd

SIP meetings aim to cover all feature changes in Scala 3 by June 2019. For this reason, the Committee meets 3 times for 3-day meetings on top of the regular monthly public meetings. The first meeting of that kind was successfully organised in November 2018, the second one is coming up mid March, and the third one is scheduled beginning of June 2019.

- ➢ SIP meeting January 2019
- ➢ New batch up for community discussion
- ➢ Organisation of the extensive, face-to-face SIP meeting of March