~~A Vision~~ for Scala Builds

fun, fast, intuitive, composable, statically checked

I am releasing 1.0-beta today
CBT

Jan Christopher Vogt / @cvogt

https://github.com/cvogt/talk-2016-03-04

Who am I?

Jan Christopher Vogt

X.ai magically schedule meetings

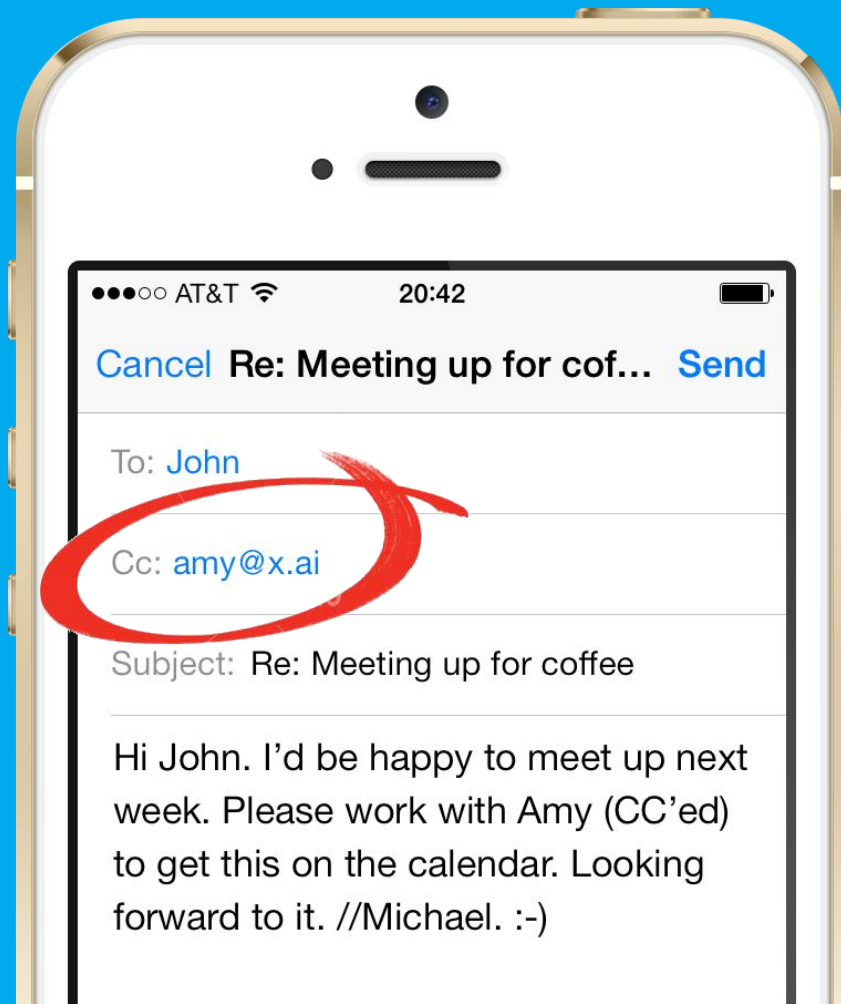Slick, Martin's lab, Compossible Records,
Scala Forklift, MavenSearch

What's the goal here?

x.ai

25 Engineers,
lots of Scala projects

CBT

Few, simple building blocks. Easy. Fast. Idiomatic Scala. Flexible and compositional. Easy code and contribution.

Building blocks

- build composition!
- Scala library of functionality
- builds scripts written in vanilla Scala
  (builds = classes, "tasks" = methods)
- configuration via inheritance
  (shallow layer)
- cli interop

Similar to SBT 0.7.
(back in 2011)

Fast

- resident JVM via Nailgun
- caching of dependencies
    - artifacts/pom on disk
    - classloader in memory
- run zinc only when changed
- concurrency (opt-in)
- native OSX file change triggers

Easy to understand and contribute

- focus!
- tiny: < 1500 LOC
- easy code! Scala beginners can understand.
- bootstraps from source: install = git clone

Supported features right now

- compile, run, test, scaladoc
- package jars
- publish to sonatype
- download jars from maven
- file change triggers
- build composition (dependent builds)

Missing features (likely easy to add)

1. configurable resolvers
2. cross scala version builds
3. scalajs support
4. repl

Rough edges

- some aspects of
    - usability
    - logging
- only tested on OSX
- only used by me :)

Live Demo

Getting started

check

- README.txt

Philosophy and design

Builds aren't special.
Just code.

Dependencies

-   zinc
-   Scala
-   BarbaryWatchService
-   ammonite-ops (still)

Custom maven resolver

- new implementation
- proper caching
- (some feature limitation still)
- Coursier as a drop-in

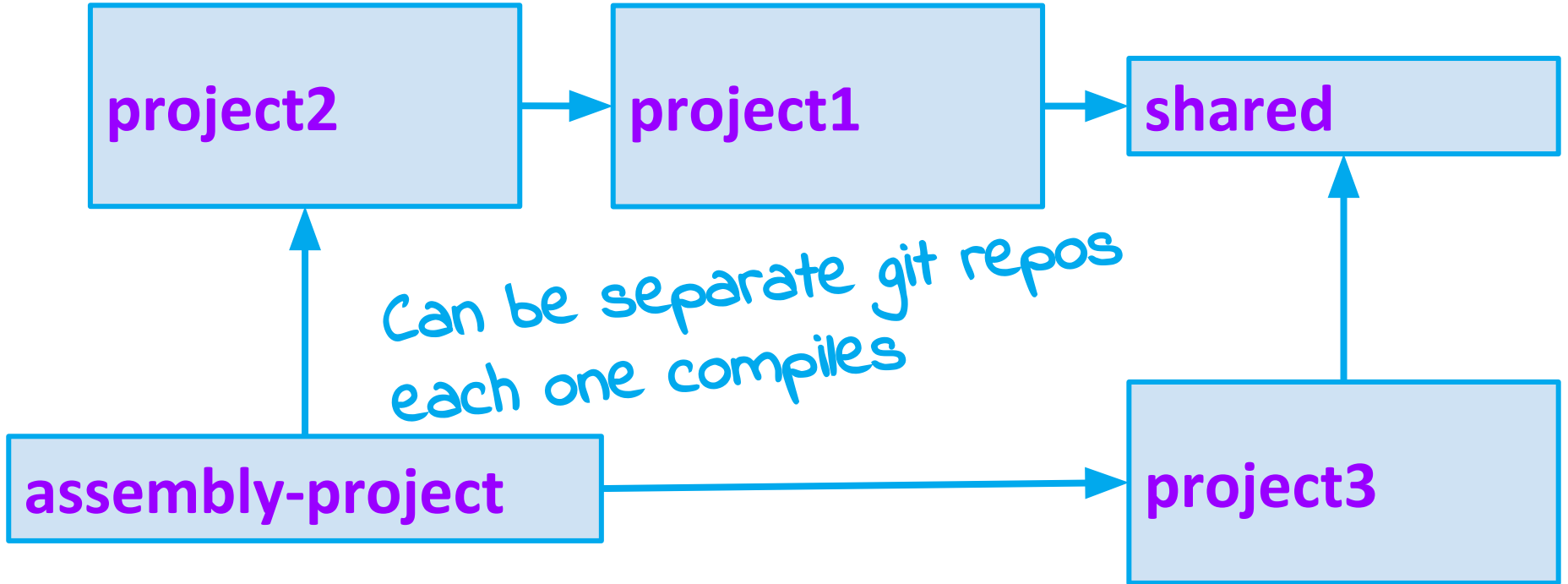# Fast re-compilation triggers

-   watches sources of project, build, CBT

Parallel task execution

- totally opt-in
- requires care regarding IO
- per-run result cache

Build composition

Build composition

Build composition

Simple building block, many uses:

1. Dependent builds
2. Multi-project builds
3. Tests
4. Builds of builds
5. Whenever A needs to compile before B

Same rules everywhere

Current limitations

Concurrent builds

- not supported yet
- should be fine via dependency graph
- requires internal care regarding caches

Re-configuration

Inheritance is static

Re-configuration through static code

Dynamic re-configuration

e.g. provide version from command line

But how?

Members are already hard-wired.

Ad-hoc re-configuration

e.g. automatically append "-SNAPSHOT" to the version number for snapshot builds

But how?

Code does not know the eventual class.

Candidate solutions

- messy, easy, powerful: mutable members
- clean, easy, limited: hard-coded use cases
- clean, hard, powerful: code-generation
  (it's a build tool after all)

Example: code-generation

```scala
class BasicBuild(...){
  def publishSnapshot: Unit = {
    val reconfiguredBuild = this.mixin(
      new BasicBuild{
        override def version = super.version+"-SNAPSHOT"
      }
    ): reconfiguredBuild
    b.publish
  }
}
```

How to contribute

The code is simple

- CBT is distributed as source. You clone it.
- check DEVELOPER_GUIDE.txt
- read the code: clean and easy.
- change it, CBT re-builds itself on use
- submit a PR

Write extensions

- very easy
- write traits to mix into builds
- publish as libs on maven
- depend on them in your BuildBuild

Diving into the source

If time permits.
Otherwise unconference session.

Near future work

- allow different CBT and Scala versions
- concurrent builds
- Coursier integration
- more tests
- Scala Days 2016 surprise feature

Future work

- git(hub) dependencies
- sbt interop

CBT?!?

Compositional Build Tool

Complex Build Tool ;)?

Cool Build Tool

CBT Build Tool

Compossible Build Tool
Composition Possible

CBT!

For now: "Chris' Build Tool"

```
if( year == 2017 && cbt.hasCommunity )
  cbt.setName(
     "Community Build Tool"
  )
```

NEScala special offer:
I'll make your CBT builds work.
Talk to me.

What's next

- This is BETA - let the bug hunting begin
- report issues or submit PRs (easy ;)).
- let's get it solid until Scala Days

```
git push
```

chris @ human.x.ai

Senior Backend Engineer

https://github.com/cvogt/cbt

Twitter: @cvogt @xdotai
Github: @cvogt
Slides, etc

we are hiring!