Accelerating developer productivity with Gradle

Gr8Conf US 2017

CRAIG ATKINSON, PRINCIPAL ENGINEER, GRADLE INC.

About me

- Craig Atkinson
- Principal Engineer @ Gradle, Inc.
- Gradle Enterprise
- craig@gradle.com
- @craigatk1
- github.com/craigatk

Agenda

- Intro
- Faster builds
  - Incremental compiler, build cache, etc.
- Easier local development
  - Composite builds, tooling API, etc.
- Deep build insights
  - Build scans, Gradle Enterprise

4.0M

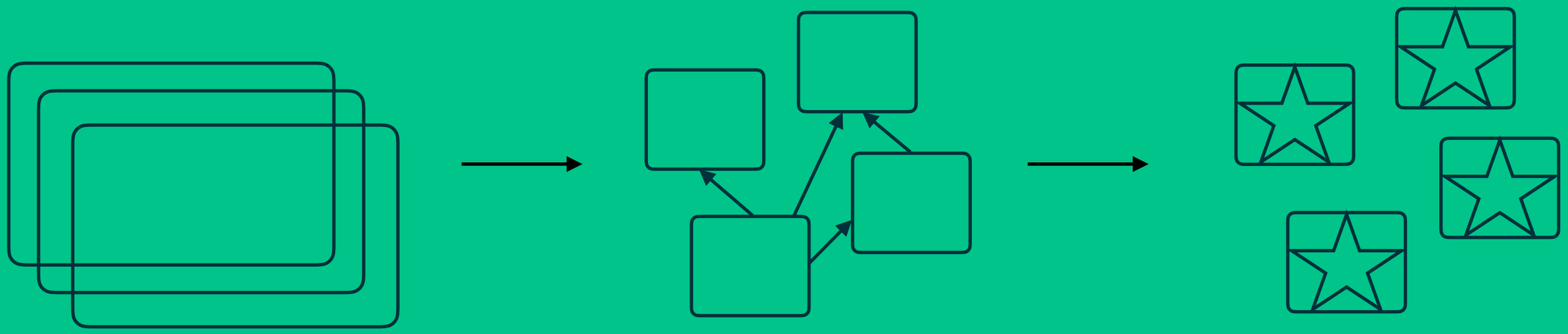
Downloads / Month

Top 20

Open-source Projects Worldwide (TechCruch)

Gradle 1-min intro

Gradle build execution



Gradle build scripts → Configure tasks → Execute tasks

2-phase build:

Configuration phase → build task graph

Execution phase → execute task graph

build.gradle

```
apply plugin: 'java-library'

repositories {
    jcenter()
}

dependencies {
    api 'org.apache.commons:commons-math3:3.6.1'
    implementation 'com.google.guava:guava:21.0'
    testImplementation 'junit:junit:4.12'
}

task greeting(type: DefaultTask) {
    doLast {
        println "Hello Gr8Conf US 2017!"
    }
}
```

Gradle 1-min intro

Demo

Incremental builds

Incremental builds

The fastest task is the one you don’t need to execute

Only re-run tasks affected by changes made between build executions

Keep output from up-to-date tasks

Incremental builds

Demo

Build cache

Build cache

- Reuse outcomes of **any** previous run
  - Rather than just the last
- Local cache and remote cache
- Task outputs are cached

Build cache

Calculate cache key from inputs, use output as cache value

Inputs —> Task —> Output

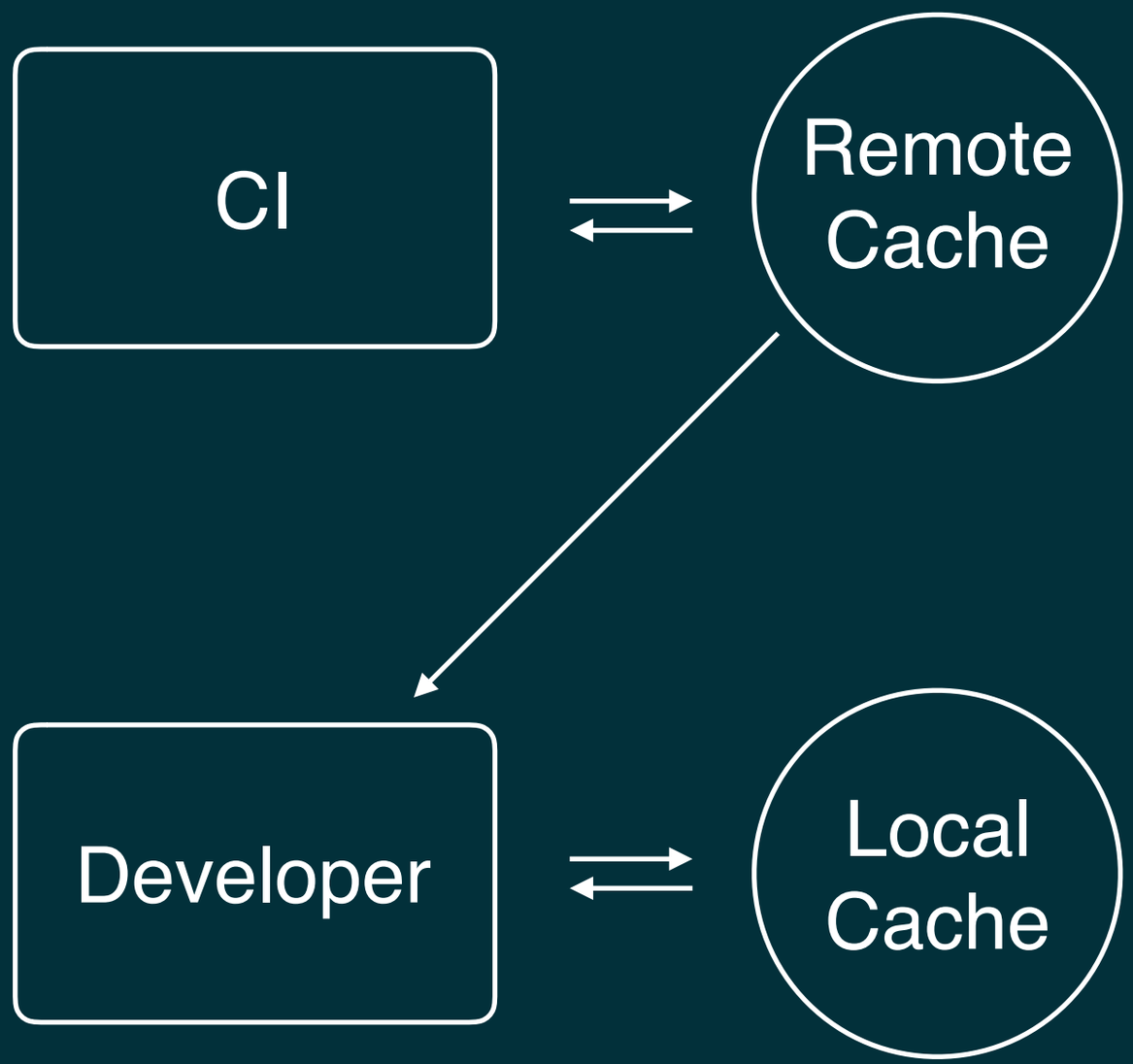
Example for Compile task:

- Cache key: hash(source files, compiler flags, etc.)
- Cache value: fileTree(class files)

Build cache

Demo

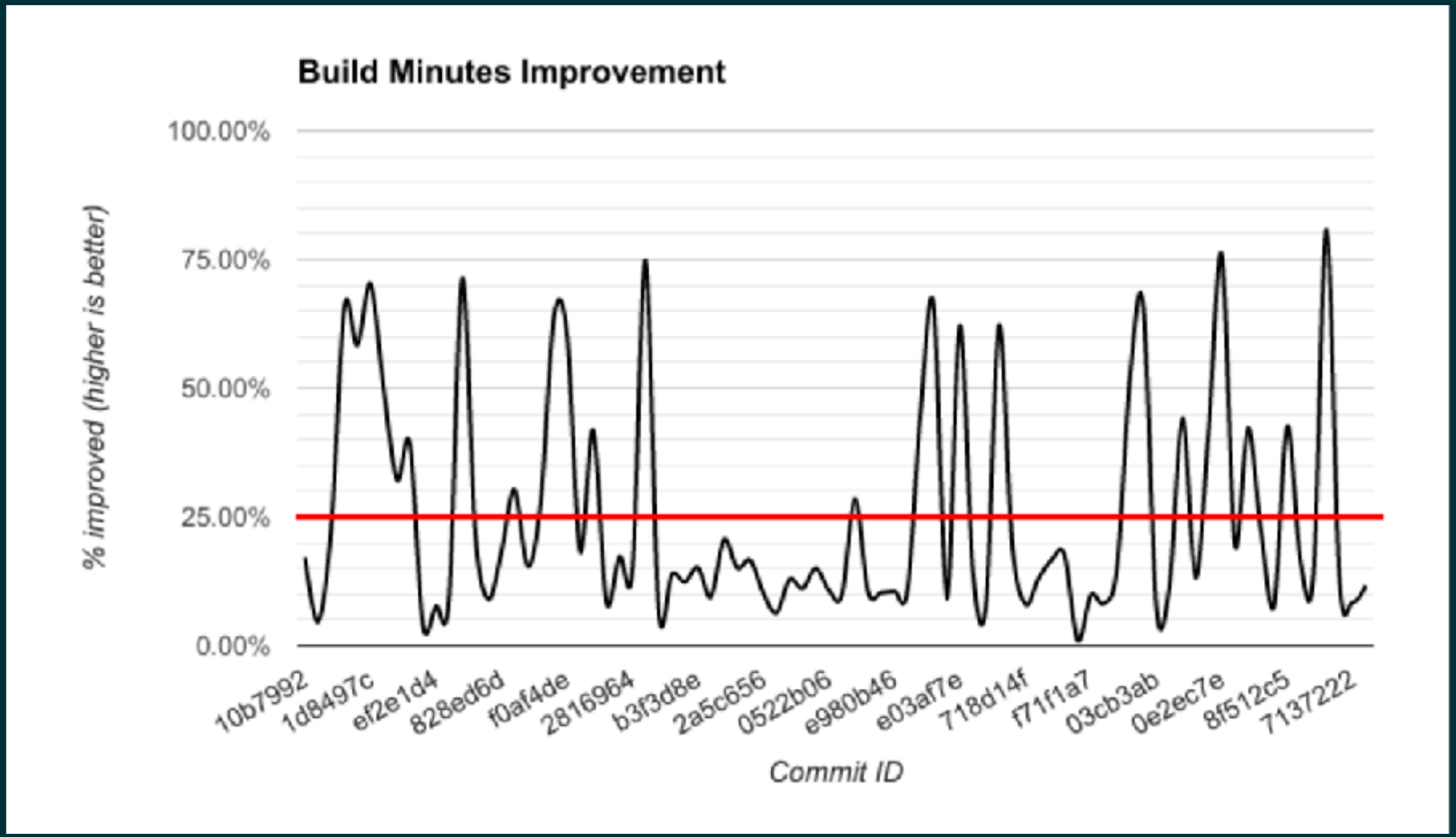
Build cache



Build cache

```
buildCache {
    local {
        enabled = !isCI
    }
    remote(HttpBuildCache) {
        url = "https://my.ge.server/cache/"
        push = isCI
    }
}
```

Build cache in Gradle build



Build cache resources

- Introduction to build cache blog post
- Extensive guide on using the build cache and improving the cacheability of your build
- User guide section on build cache
- Highly-performant, scalable build cache backend available in Gradle Enterprise
- Build cache node Docker image

Compile avoidance & incremental compiler

Compile avoidance & incremental compiler

Save time by only recompiling the **minimum number** of source files needed for a given change

Compile avoidance

recompile

unchanged

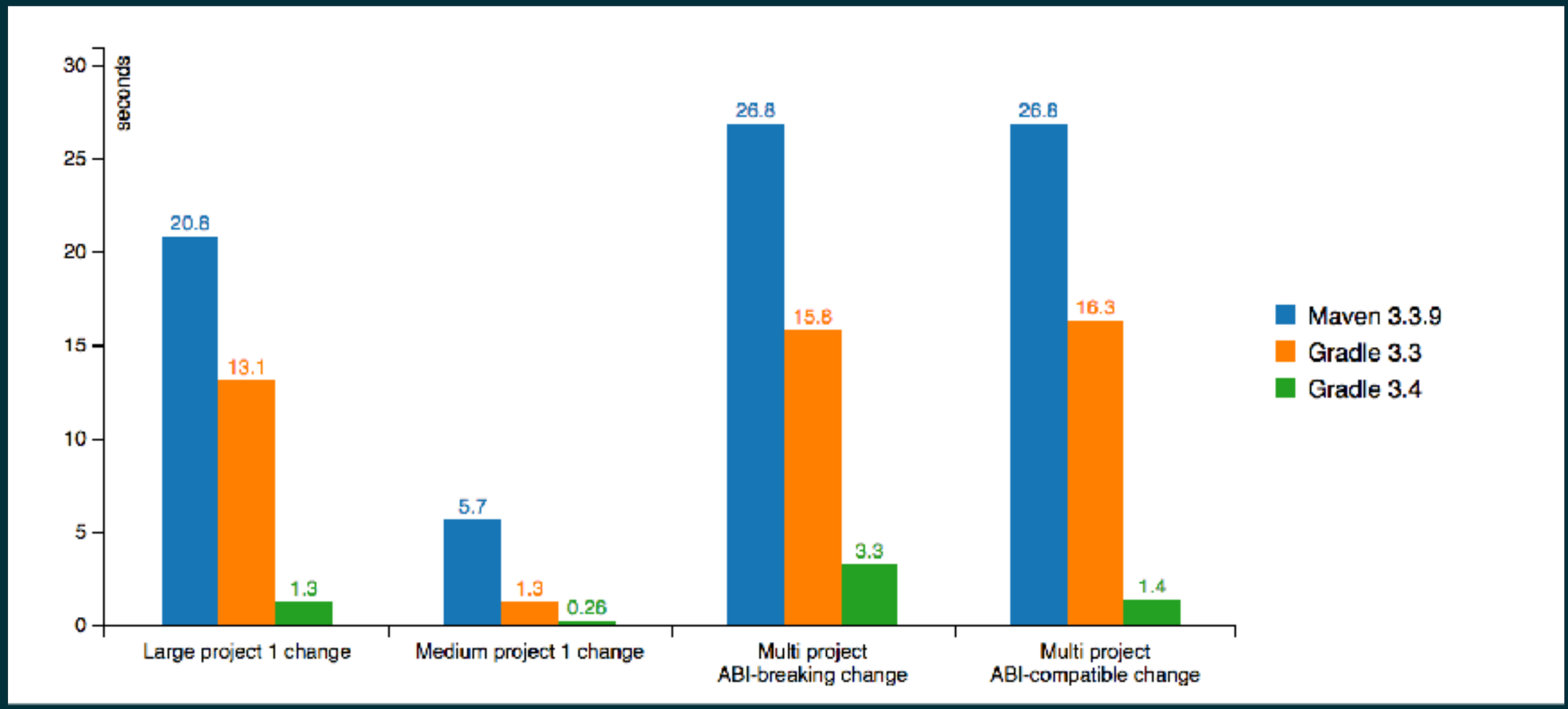
Incremental compiler

- Analyze class dependencies to optimize which classes are recompiled
- Fast in-memory cache of class ABIs inside daemon

Enable incremental compiler

```
tasks.withType(JavaCompile) {
    options.incremental = true
}
```

Compile avoidance & incremental compiler



Compile avoidance & incremental compiler

- Introduction to incremental compiler and compile avoidance
- Video of Gradle Summit presentation on incremental compilation
- User guide section on Java plugin
- Github repo with performance benchmark projects

General performance improvements

General performance improvements

- Faster configuration time
- Parallel dependency downloads
- Parallel task / action execution by default

Gradle daemon

Gradle daemon

Gradle builds executed much more quickly by a long-lived background process that avoids expensive bootstrapping and leverages caching

Gradle daemon resources

- Gradle daemon user docs section

Worker API

Worker API

- Previously tasks in different projects can run in parallel
- New API to run task actions in single project in parallel safely
- Parallel actions cannot mutate shared state

Example worker

```
import javax.inject.Inject

class ReverseFile implements Runnable {
    File fileToReverse
    File destinationFile

    @Inject
    public ReverseFile(File fileToReverse, File destinationFile) {
        this.fileToReverse = fileToReverse
        this.destinationFile = destinationFile
    }

    @Override
    public void run() {
        destinationFile.text = fileToReverse.text.reverse()
    }
}
```

```
class ReverseFiles extends SourceTask {
  final WorkerExecutor workerExecutor

  @OutputDirectory
  File outputDir

  // The WorkerExecutor will be injected by Gradle at runtime
  @Inject
  public ReverseFiles(WorkerExecutor workerExecutor) {
    this.workerExecutor = workerExecutor
  }

  @TaskAction
  void reverseFiles() {
    source.files.each { file ->
      workerExecutor.submit(ReverseFile.class) { WorkerConfiguration config ->
        // Use the minimum level of isolation
        config.isolationMode = IsolationMode.NONE

        // Constructor parameters for the unit of work implementation
        config.params = [ file, project.file("${outputDir}/${file.name}") ]
      }
    }
  }
}
```

Worker isolation levels

- NONE - runs in same thread, minimum isolation
- CLASSLOADER - runs in thread with isolated classloader
- PROCESS - runs in separate process, maximum isolation

Worker API resources

- Video of Gradle Summit presentation on worker API
- Worker API documentation

Continuous build

Continuous build

- Gradle watches for file changes, re-runs tasks
- Run Gradle with `-t`

Continuous build

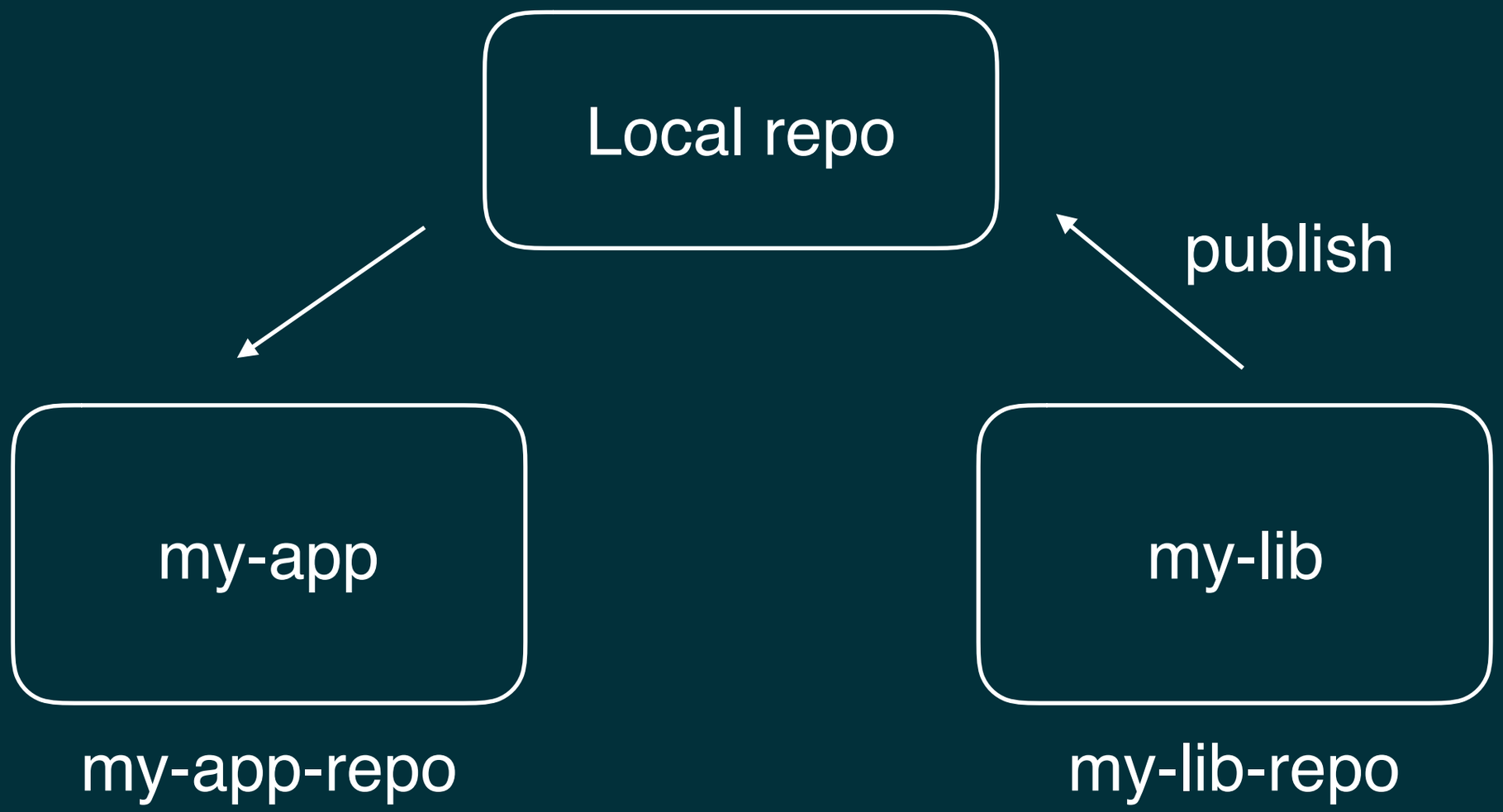
Demo

Continuous build resources

- Blog post introducing continuous build
- Continuous build user docs

Composite builds

Multi-repo change



Composite build

Composite builds

- Fix a bug in a library used by app
- Break down a monolith into multiple repos
- Consume latest state of libraries in integrations builds

Use composite builds

Command line

```
gradle --include-build ../my-utils run
```

settings.gradle

```
rootProject.name='my-app'

includeBuild '../my-utils'
```

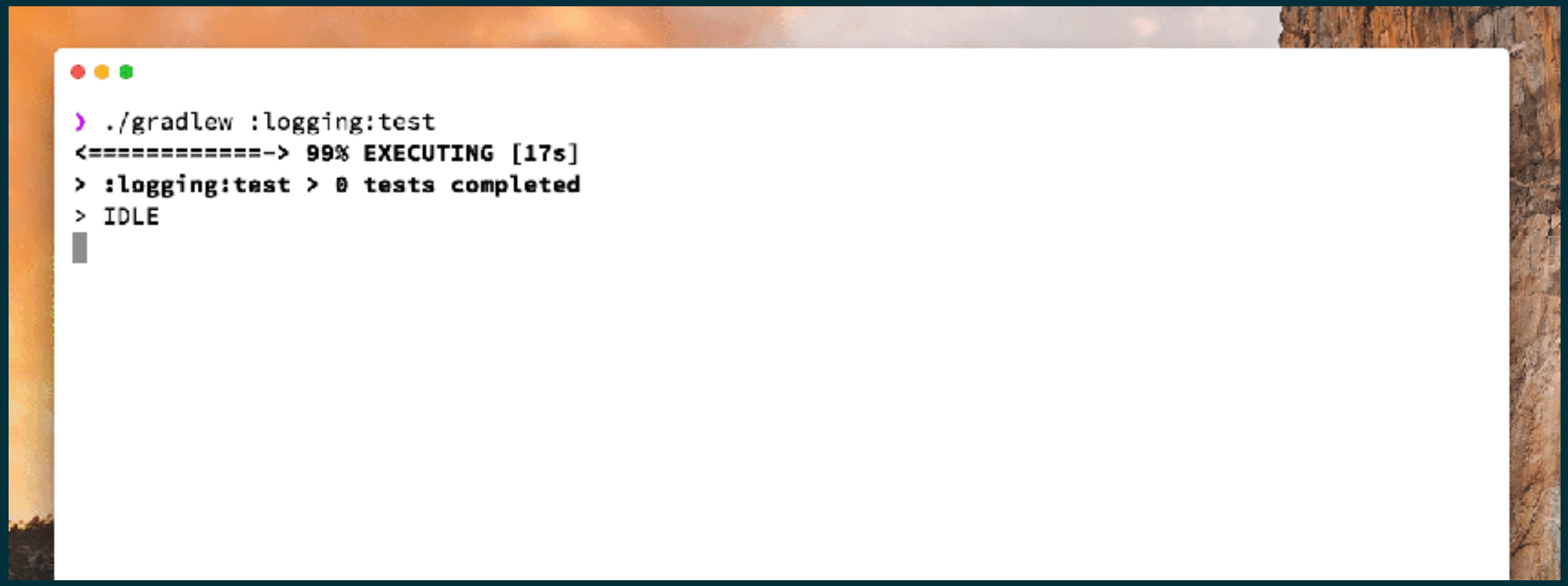
Composite build resources

- Introduction to composite builds
- Video of Gradle Summit presentation on composite builds
- Video on composite builds with IntelliJ IDEA
- User docs section on composite builds

New Gradle console

Parallel tasks console demo

Parallel tests console demo

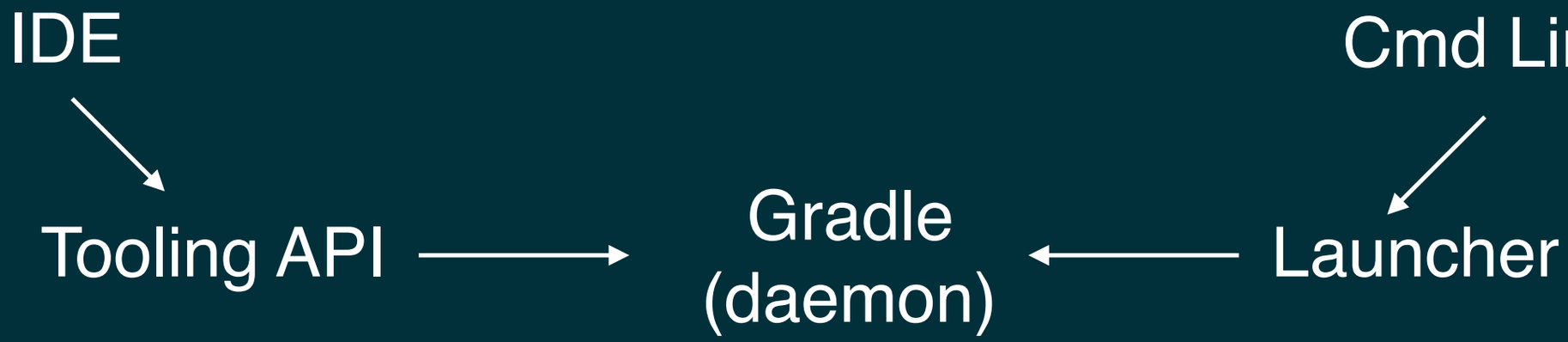


IDE integration

IDE integration

- Project setup & synchronization
- Task execution
- Test execution
- Build execution insights

Tooling API



IDE integration resources

- Buildship 2.0 blog post
- Buildship in Eclipse Marketplace
- IntelliJ/Gradle integration docs
- Gradle IDEA plugin docs

Build scans

Build scans

Deep insights into a build execution

- Details about build failures
- Visual timeline of which tasks ran and in which order
- Details on why tasks were executed (up-to-date reasons)
- Which dependencies were used
- Performance analysis of configuration, execution, etc.
- Attach custom data to your builds (Git commit, CI or local, Checkstyle errors, etc.)

Build scans

Improve build performance

- Quickly identify places in your build to optimize
- Find slowest tasks, long build configuration time, long dependency download times, etc.
- See which tasks are and aren’t cacheable
- Identify slowest tests

Build scans

Collaborate with colleagues and the community

- Easily share exactly what happened in when your build ran (tasks, tests, etc.)
- Build environment (JDK, OS, CLI switches, etc.)
- Share exact links to many different parts of the build scan (specific task, test, dependency, console output line, etc.)

Build scans

Demo

Build scan resources

- scans.gradle.com
- Get started with build scans
- Build scan plugin user manual

Build scans are a free service for everyone!

Gradle Enterprise

Gradle Enterprise

- Build scans + search + comparison + more
- Scalable, high-performance build cache backend
- Hosted on-premise

Search build scans

- Search based on project, tasks executed, start time, custom tags and values, etc.
- Find build scans to compare

Search build scans

Demo

Compare build scans

- Compare build scans to find differences between builds
  - Dependencies, task inputs, custom values, environment, etc.

Compare build scans

Demo

Build cache backend

- High-performance, scalable build cache backend
- Build cache backend supports multiple, distributed nodes

Build cache UI

Demo

Gradle Enterprise

- Gradle Enterprise is a commercial offering
- Learn more: gradle.com/enterprise

Summary

- Incremental builds
- Build cache
- Compile avoidance & incremental compiler
- Worker API
- Daemon
- Continuous builds
- Composite builds
- Tooling API / IDE integration
- Build scans
- Gradle Enterprise

Additional resources

- Online training (intro class is free!)
- Getting-started and topic-based guides
- User documentation
- Gradle forums

Q & A

Thank you!

Craig Atkinson