

# Gradle Optimization 101

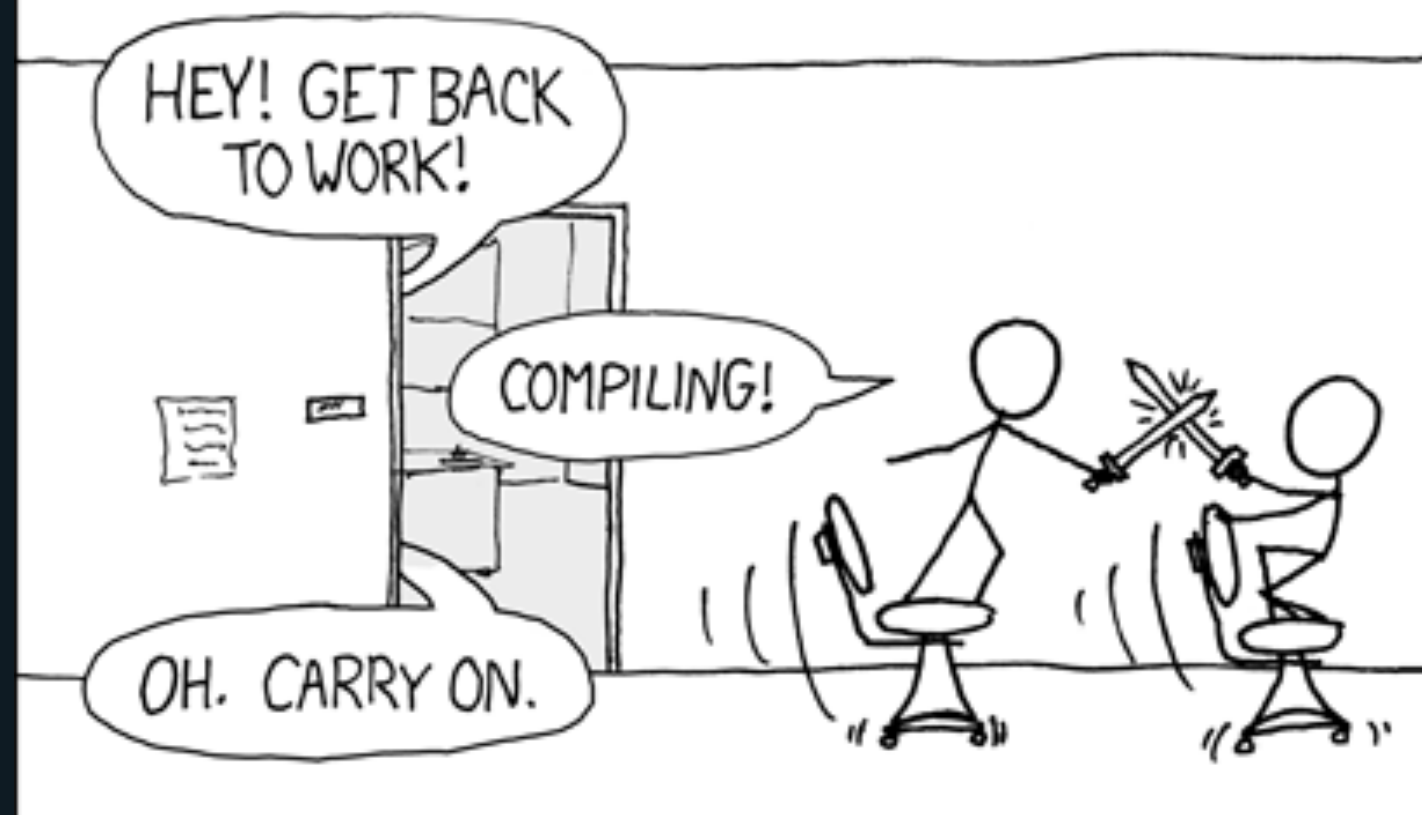


Gradle

# What is Gradle?

Gradle is an open-source build automation tool that helps in managing and automating the build process of software projects, specifically Java and Android projects.

THE #1 PROGRAMMER EXCUSE  
FOR LEGITIMATELY SLACKING OFF:  
"MY CODE'S COMPILING."



**How we can optimize it?**

# How we can optimize it?

- Don't use it

# How we can optimize it?

- Don't use it
- Use another build automation tool, like Bazel


# How we can optimize it?

- Don't use it
- Use another build automation tool, like Bazel
- Change language

# How we can optimize it?

- Don't use it
- Use another build automation tool, like Bazel
- Change language
- Change job



A close-up, high-contrast photograph of Santa Claus. He has a surprised or excited expression, with wide eyes and a slightly open mouth. His white beard and hair are prominent, and the lighting is warm, highlighting his face against a dark background.

***Fly you fools.***

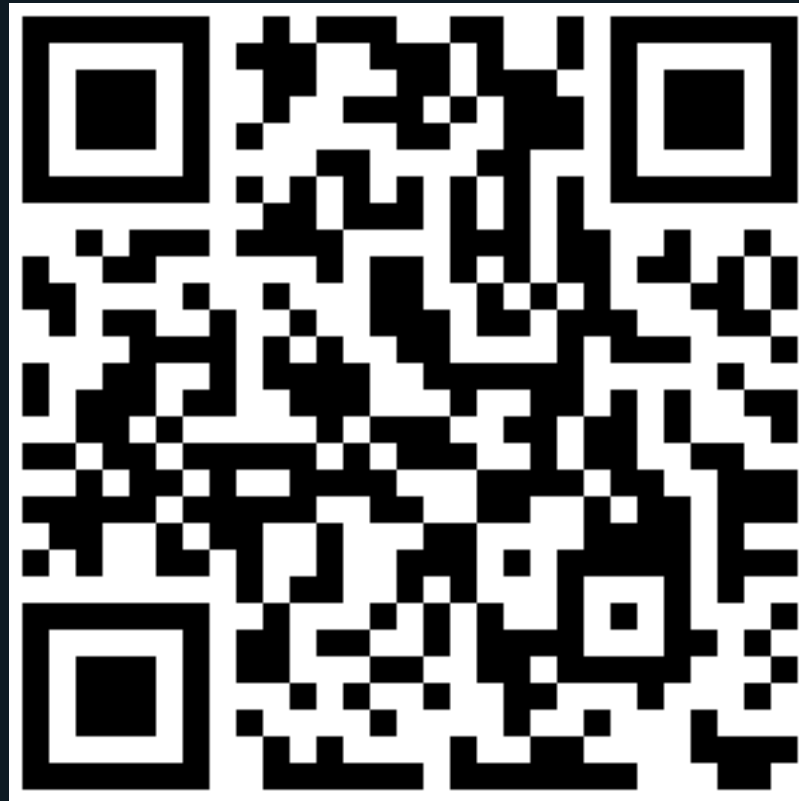
# How really optimize Gradle Build<sup>1</sup>

<sup>1</sup> Disclaimer: The use of Gradle for your project may result in headaches, confusion, depression, and an unexplained desire to pursue a career in farming. Please proceed with caution and seek guidance from experienced professionals.

**If you give a man a fish, you  
feed him for a day. If you teach  
a man to fish, you feed him for  
a lifetime.**

— Chinese proverb

# Learning Gradle with Gradle Trainings



# Modularize your projects

Modularizing your code allows the build system to compile only the modules you modify and cache those outputs for future builds.

Modularization also makes parallel project execution more effective when you enable that optimization.

# Maintain your module dependencies clean

For android projects, use Dependency Analysis Android Gradle Plugin



# Maintain your module dependencies clean

For others type of projects





# Optimize repository order

When Gradle resolves dependencies, it searches through each repository in the declared order.

To reduce the time spent searching for dependencies, declare the repository hosting the largest number of your dependencies first. T

his minimizes the number of network requests required to resolve all dependencies.



# Minimize repository count

Limit the number of declared repositories to the minimum possible for your build to work.

If you're using a custom repository server, create a virtual repository that aggregates several repositories together. Then, add only that repository to your build file.

# Enable parallel execution

To execute project tasks in parallel by default, add the following setting to the `gradle.properties` file in the project root or your Gradle home:

```
org.gradle.parallel=true
```

# Update versions

# Update versions

— Gradle

# Update versions

- Gradle
- Java

# Update versions

- Gradle
- Java
- Plugins

# Re-enable Gradle Daemon

The Gradle Daemon reduces build times by:

# Re-enable Gradle Daemon

The Gradle Daemon reduces build times by:

- caching project information across builds



# Re-enable Gradle Daemon

The Gradle Daemon reduces build times by:

- caching project information across builds
- running in the background so every Gradle build doesn't have to wait for JVM startup

# Re-enable Gradle Daemon

The Gradle Daemon reduces build times by:

- caching project information across builds
- running in the background so every Gradle build doesn't have to wait for JVM startup
- benefiting from continuous runtime optimization in the JVM

# Re-enable Gradle Daemon

The Gradle Daemon reduces build times by:

- caching project information across builds
- running in the background so every Gradle build doesn't have to wait for JVM startup
- benefiting from continuous runtime optimization in the JVM
- watching the file system to calculate exactly what needs to be rebuilt before you run a build

# Avoid Java Home Mismatch

Using a different JDK for command line builds vs Android Studio or IntelliJ builds will cause a new Gradle daemon to spawn. This will instantly double the memory being used by Gradle.

To fix this issue, we recommend setting your JAVA\_HOME environment variable and then using this JAVA\_HOME as the JDK used by Android Studio or IntelliJ.

# Profile Your Build

A proper build inspection helps you understand:

# Profile Your Build

A proper build inspection helps you understand:

- How long it takes to build your project

# Profile Your Build

A proper build inspection helps you understand:

- How long it takes to build your project
- Which parts of your build are slow

# Profile Your Build

A proper build inspection helps you understand:

- How long it takes to build your project
- Which parts of your build are slow
- A comparison point to better understand the impact of the changes recommended on this talk.



# How to profile a build?

Use gradle profile, build scan or gradle enterprise.

Here a blog post to start with gradle profile:



# Gradle Enterprise Build Validation Scripts

The purpose of the build validation scripts is to assist you in validating that your Gradle builds are in an optimal state in terms of maximizing work avoidance.



# Incremental Build Check

Incremental build is a Gradle optimization that skips running tasks that have previously executed with the same inputs.

If a task's inputs and its outputs have not changed since the last execution, Gradle skips that task.

Use the first build validation scripts to validates that a Gradle build is optimized for incremental building, implicitly invoked from the same location.

# Enable the build cache

The build cache is a Gradle optimization that stores task outputs for specific input.

When you later run that same task with the same input, Gradle retrieves the output from the build cache instead of running the task again.

To enable the build cache by default, add the following setting to the `gradle.properties` file:

```
org.gradle.caching=true
```

# Build Cache Fix

When it works, it's great! When it doesn't, it's frustrating at best (cache misses) and a major problem at worst (incorrect/corrupt cache entries).

For Android is recommended the Cache Fix Gradle Plugin:



# Local Build Cache Check

Use the second build validation scripts to validate that a Gradle build is optimized for local build caching when invoked from the same location.

Use the third build validation scripts to validate that a Gradle build is optimized for local build caching when invoked from different locations.

# Remote Build Cache

When multiple developers work on the same project, they don't just need to build their own changes: whenever they pull from version control, they end up having to build each other's changes as well.

```
boolean isCiServer = System.getenv().containsKey("CI")
```

```
buildCache {  
    remote(HttpBuildCache) {  
        url = 'https://example.com:8123/cache/'  
        push = isCiServer  
    }  
}
```

# Build Cache Node

Gradle provides a Docker image for a build cache node, which can connect with Gradle Enterprise for centralized management or used without a Gradle Enterprise installation with restricted functionality.





# Remote Build Cache Check

Use the fourth build validation scripts to validate that a Gradle build is optimized for remote build caching when invoked from different CI agents.

Use the fifth build validation scripts to validate that a Gradle build is optimized for remote build caching when invoked on CI agent and local machine.

# Enable Configuration Cache

You can cache the result of the configuration phase by enabling the configuration cache.

To enable the configuration cache by default, add the following setting to the `gradle.properties` file:

```
org.gradle.unsafe.configuration-cache=true
```

Build configuration inputs include:

Build configuration inputs include:

- Init scripts

Build configuration inputs include:

- Init scripts
- Settings scripts

Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts

Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts
- System properties used during the configuration phase

Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts
- System properties used during the configuration phase
- Gradle properties used during the configuration phase



Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts
- System properties used during the configuration phase
- Gradle properties used during the configuration phase
- Environment variables used during the configuration phase

Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts
- System properties used during the configuration phase
- Gradle properties used during the configuration phase
- Environment variables used during the configuration phase
- Configuration files accessed using value suppliers such as providers

Build configuration inputs include:

- Init scripts
- Settings scripts
- Build scripts
- System properties used during the configuration phase
- Gradle properties used during the configuration phase
- Environment variables used during the configuration phase
- Configuration files accessed using value suppliers such as providers
- buildSrc inputs, including build configuration inputs and source files

# Experiment with the JVM parallel garbage collect

Set the optimal JVM garbage collector used by Gradle in `gradle.properties`.

By default, JDK 8 use the parallel collector; JDK 9 or higher, use the G1 collector.

We recommend testing your Gradle builds with different garbage collector.

```
org.gradle.jvmargs=-XX:+UseParallelGC
```


# Increase the JVM heap size

If you observe slow builds, and in particular the garbage collection takes more than 15% of the build time, then you should increase the Java Virtual Machine (JVM) heap size.

In the `gradle.properties` file, set the limit to 4, 6, or 8 gigabytes as shown in the following example:

```
org.gradle.jvmargs=-Xmx6g
```



A close-up shot of a man's face, showing a look of intense shock or fear. His eyes are wide open, and his mouth is slightly agape. He has dark, curly hair and a small, dark mark on his left cheek. The background is filled with bright orange and yellow flames and thick, dark smoke, suggesting a scene of destruction or a fire. The lighting is dramatic, with the fire providing a strong light source.

*It's done.*



— NAME: Francesco Bonni





- **NAME:** Francesco Bonni
- **POSITION:** Senior Android Developer



- **NAME:** Francesco Bonni
- **POSITION:** Senior Android Developer
- **COMPANY:** Subito.it



- **NAME:** Francesco Bonni
- **POSITION:** Senior Android Developer
- **COMPANY:** Subito.it
- **EXPERTISE:** Creare presentazioni con pessime battute





**is Hiring!!**