Gradle Build Cache Deep Dive

Accelerating Developer Productivity with Faster Feedback Cycles

Gradle

Gradle Training Program
training@gradle.com

Objectives

- Understand benefits and how to use Gradle Build Cache
- Hands-on exercises to get you going

- Course pace picks up gradually

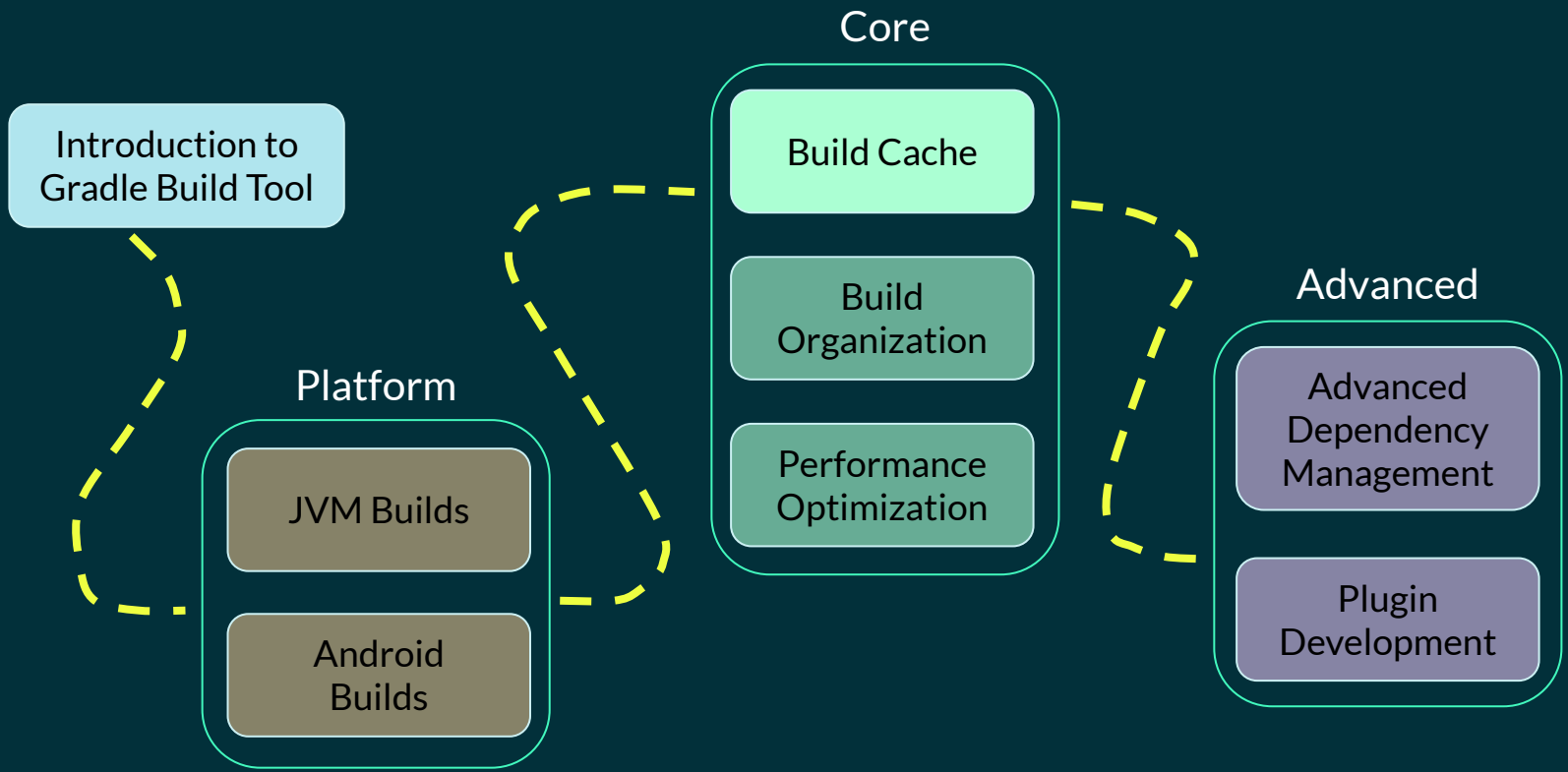
Prerequisites and Notes

- JDK 8+ and Gradle Build Tool installed
  - https://gradle.org/install/
- Gradle Build Tool experience
  - Knowledge of core concepts
  - Authoring build files
  - Kotlin experience a plus but not required
- Basic experience with Java software development
- Hands-on labs
  - READMEs will have instructions

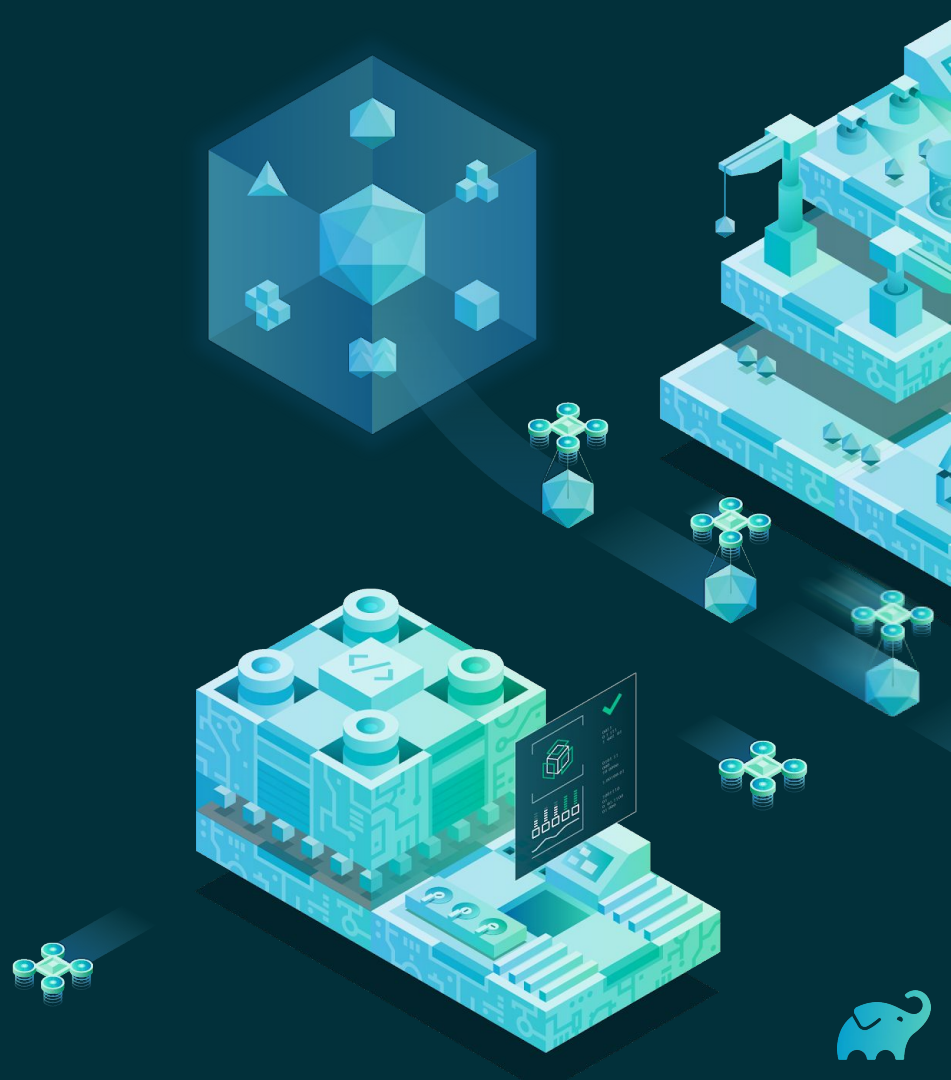
Agenda

- Historical Background
- Build Cache in Gradle
  - Incremental Builds
  - Local Cache
  - Remote Cache
- Enabling for Custom Tasks
- Troubleshooting Cache Misses
- Next Steps

Training Journey - Future Topics

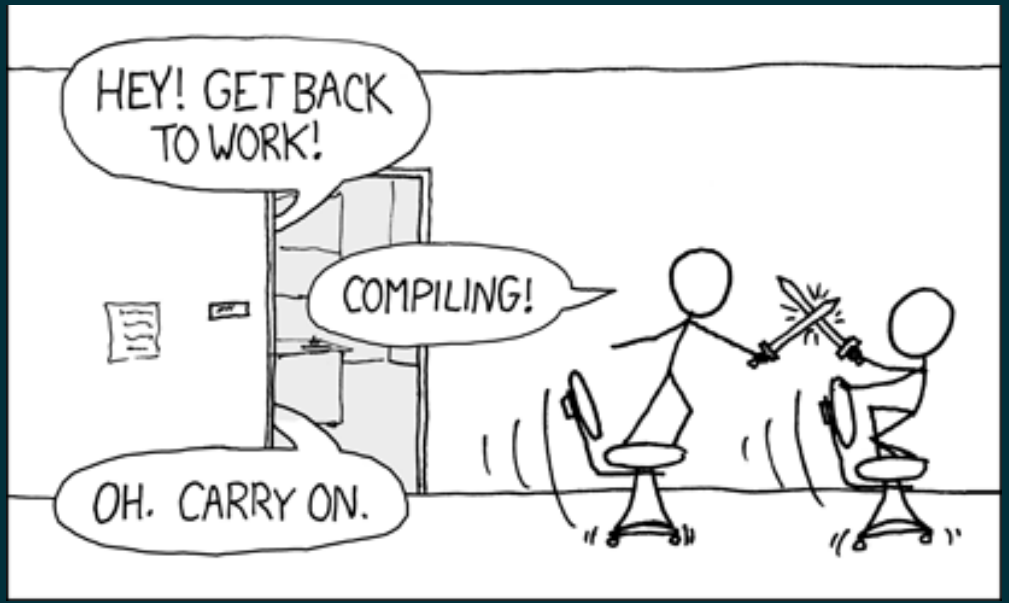


What is a Build Cache?



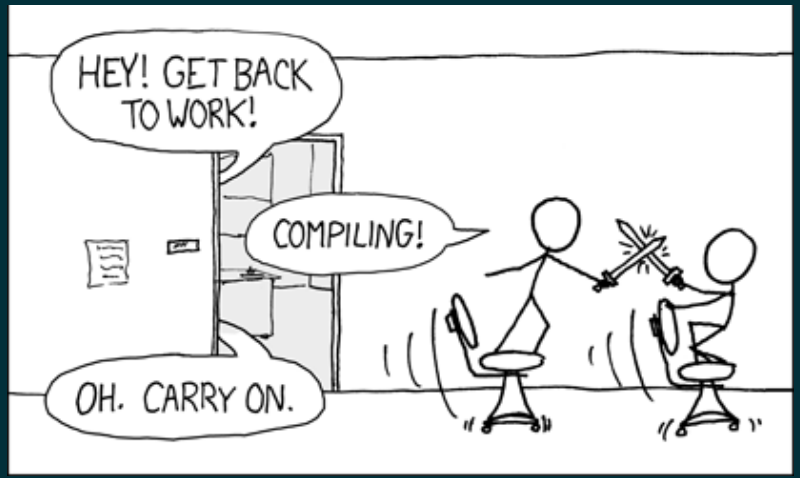
Historical Background

- As software applications have grown so have build durations

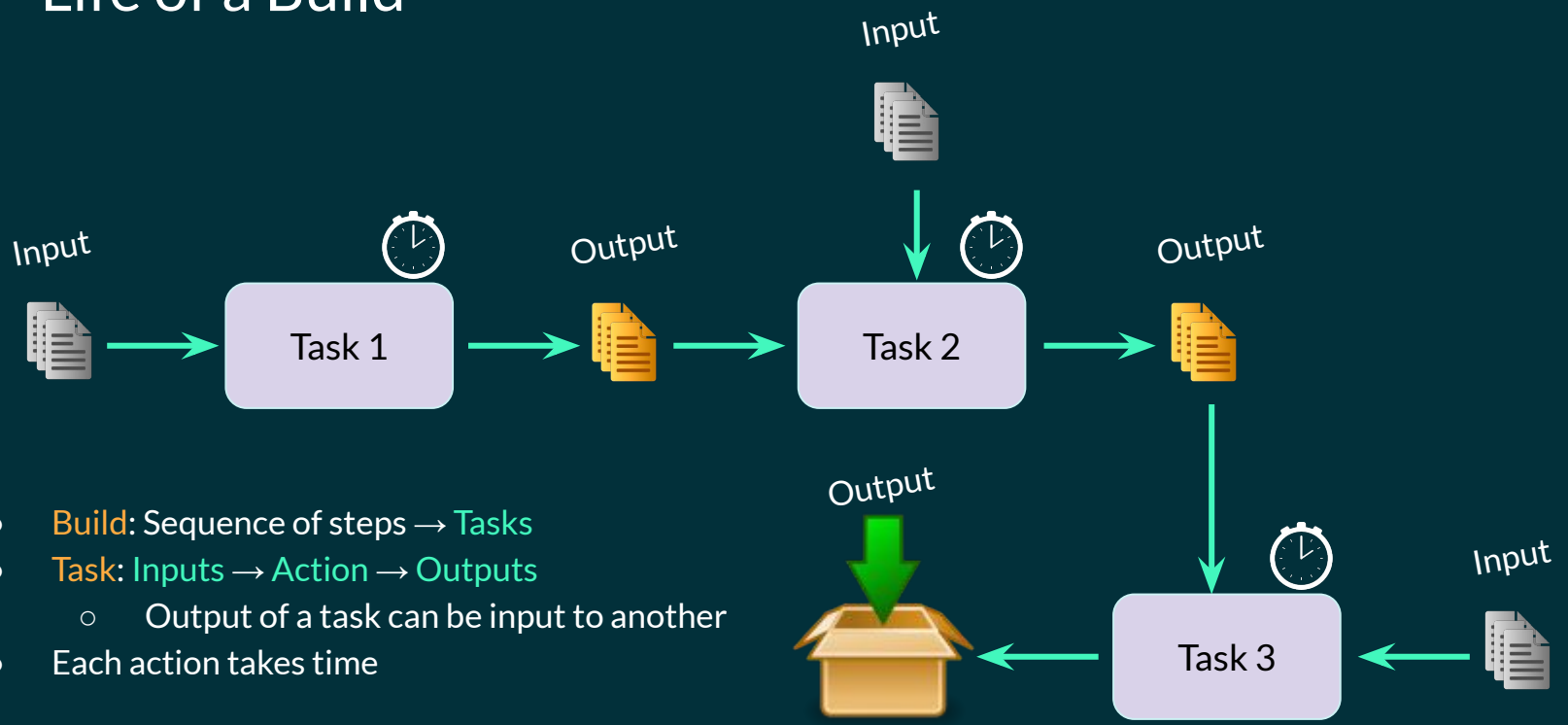


Historical Background

- Cases where builds take several hours
- Throwback to punch-card computing experience
  - Hand code to a person and come back hours later

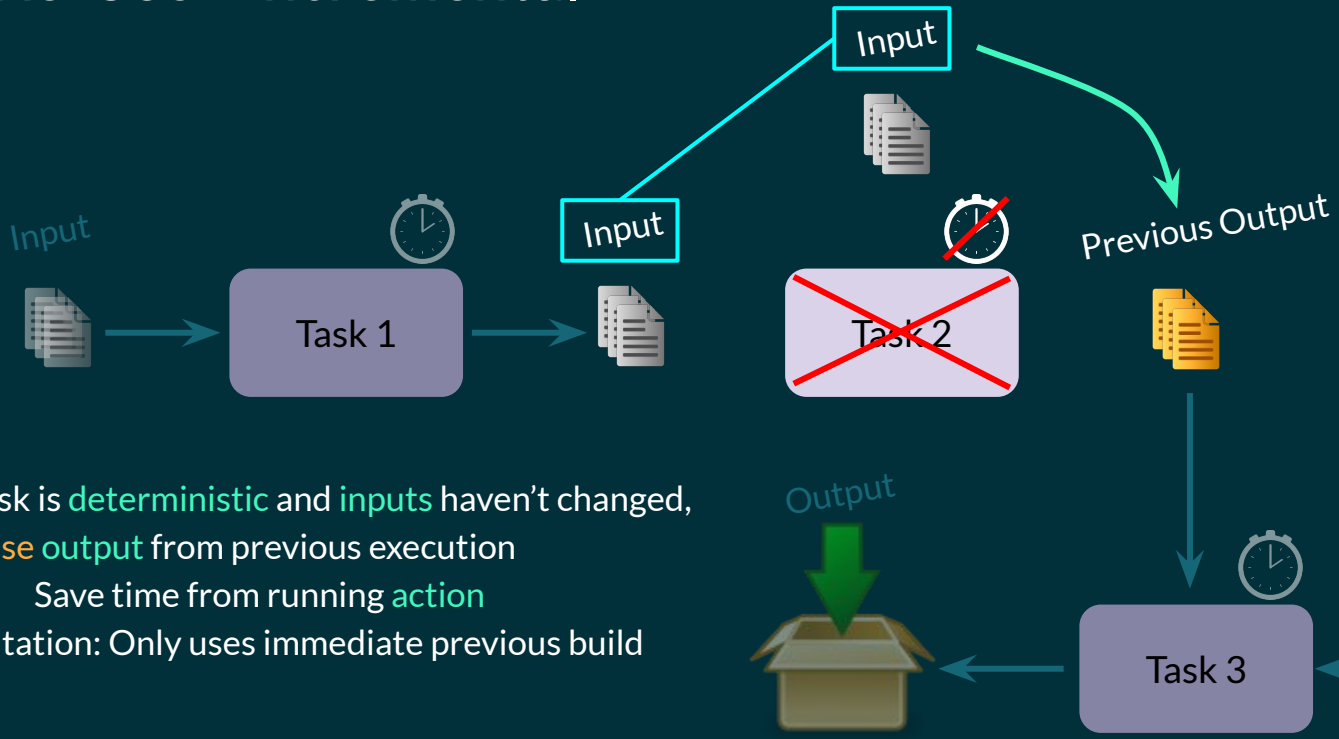


Life of a Build



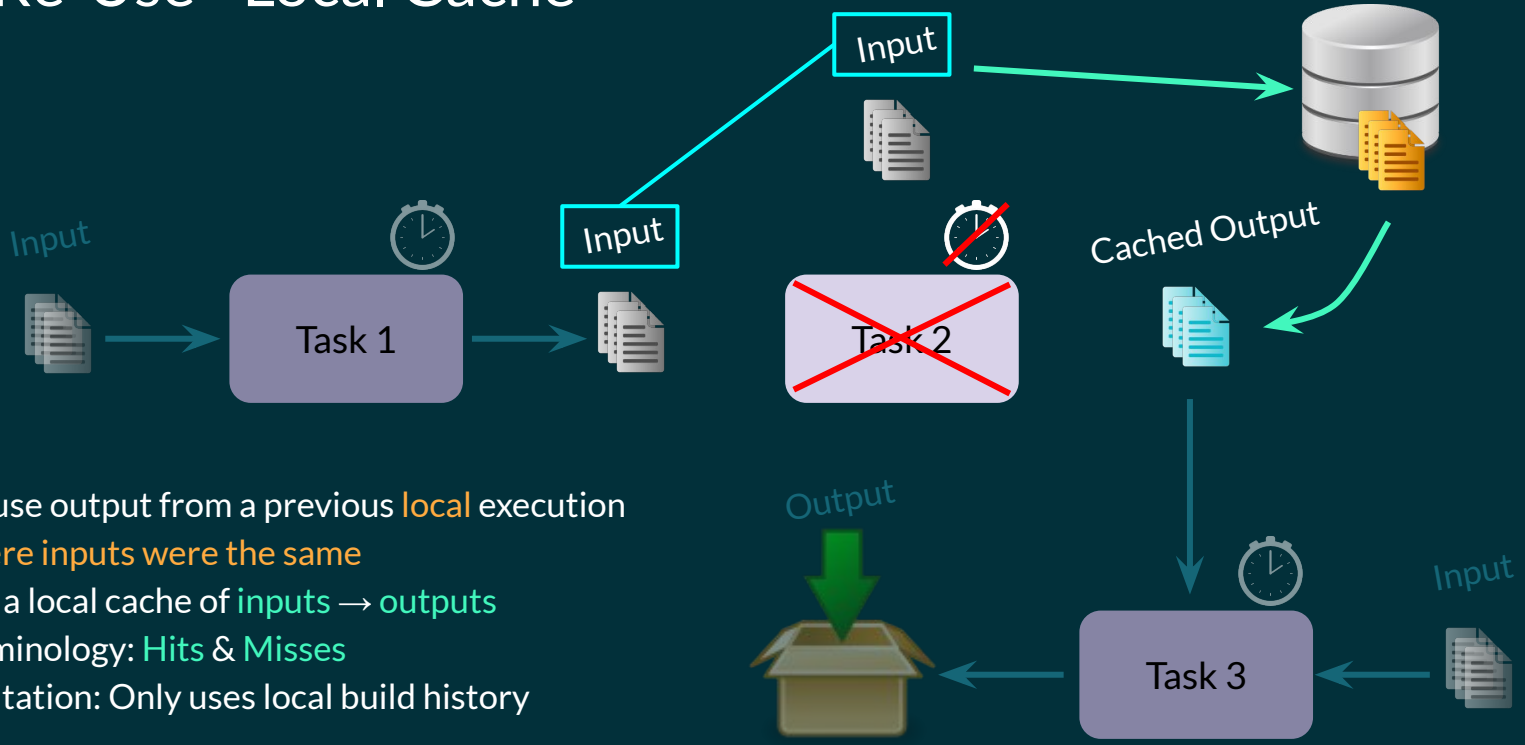
- Build: Sequence of steps → Tasks
- Task: Inputs → Action → Outputs
  - Output of a task can be input to another
- Each action takes time

Performance Ideas:
Re-Use - Incremental



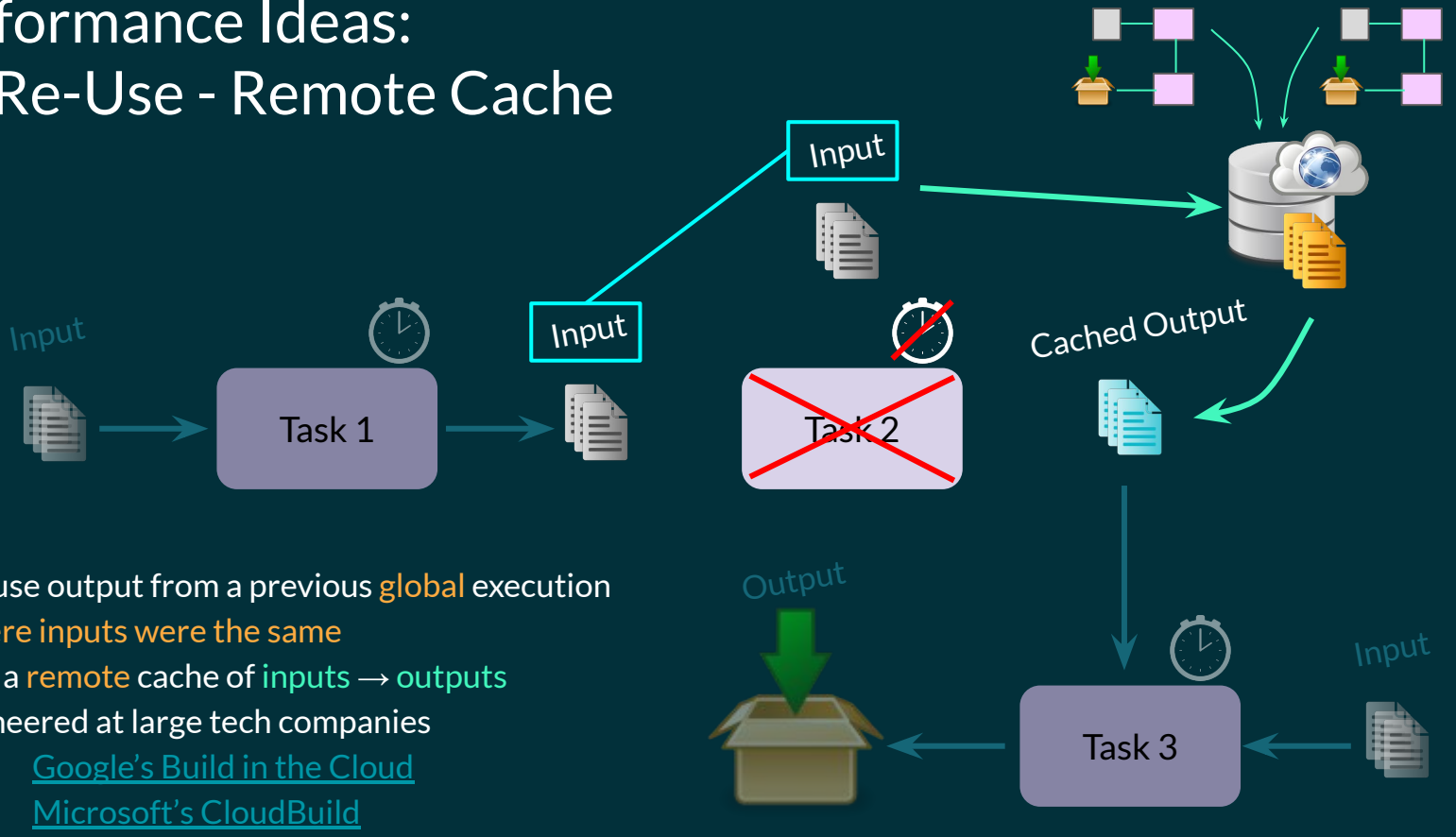
- If task is deterministic and inputs haven't changed, re-use output from previous execution
  - Save time from running action
- Limitation: Only uses immediate previous build

Performance Ideas:
Re-Use - Local Cache



- Re-use output from a previous local execution where inputs were the same
- Use a local cache of inputs → outputs
- Terminology: Hits & Misses
- Limitation: Only uses local build history

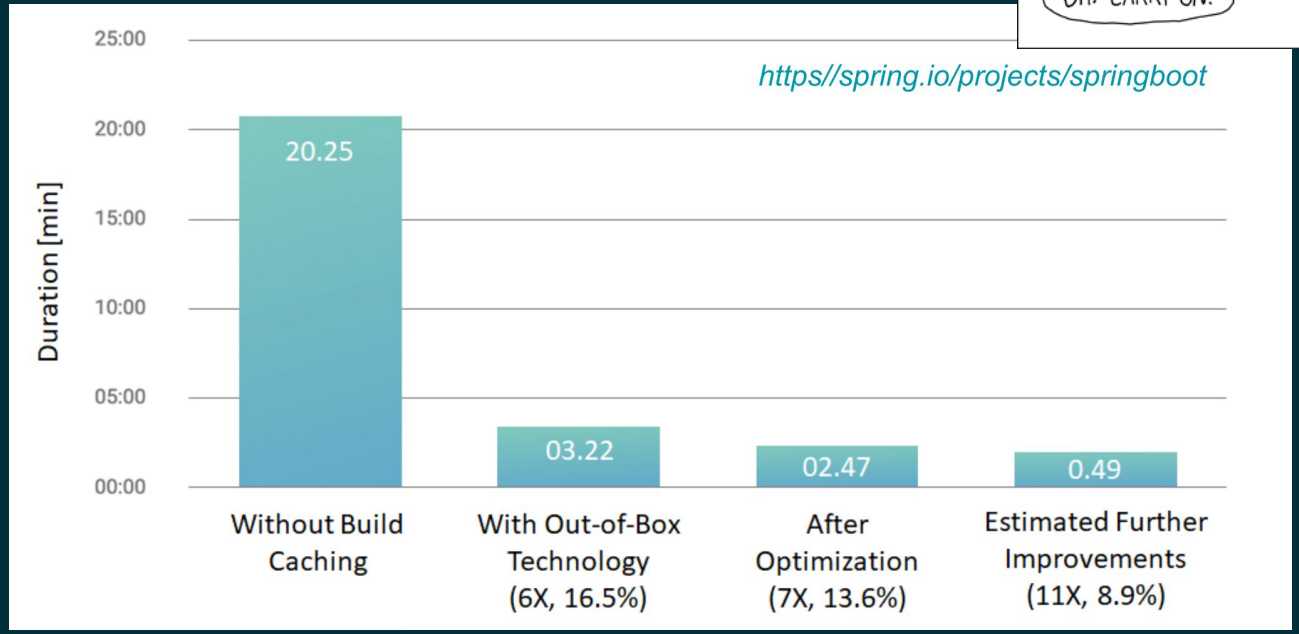
Performance Ideas:
Re-Use - Remote Cache



- Re-use output from a previous global execution where inputs were the same
- Use a remote cache of inputs → outputs
- Pioneered at large tech companies
  - Google's Build in the Cloud
  - Microsoft's CloudBuild

Serious Time Savings

https//spring.io/projects/springboot

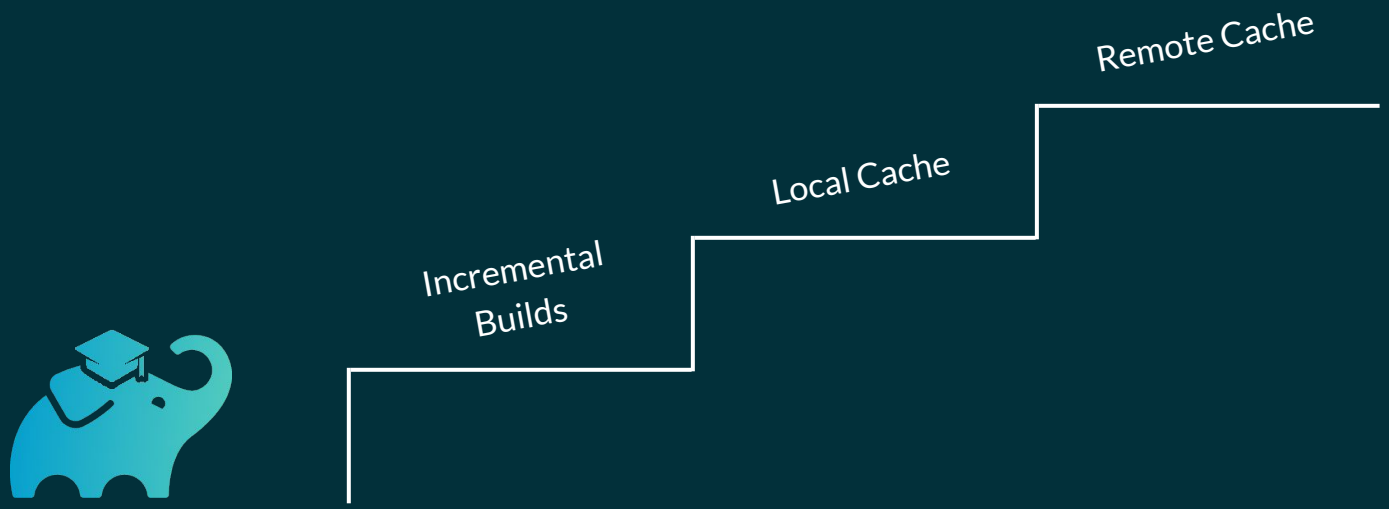


Build Cache in Gradle

- Historical Background ✓
- Build Cache in Gradle
  - Incremental Builds
  - Local Cache
  - Remote Cache
- Enabling with Custom Tasks
- Troubleshooting Cache Misses
- Next Steps

Build Cache in Gradle

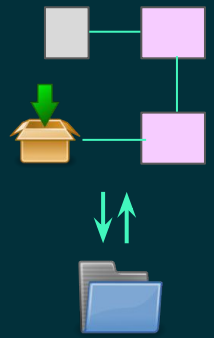
- Gradle wants to bring this performance feature to everyone
- Support for build cache added over time



Incremental Builds

- UP-TO-DATE outcome label → Task actions not executed, previous output used
- Gradle uses cached task outputs from build folder if inputs were same and task is deterministic

```
> Task :app:compileJava UP-TO-DATE
> Task :app:processResources NO-SOURCE
> Task :app:classes UP-TO-DATE
> Task :app:compileTestJava
> Task :app:processTestResources NO-SOURCE
> Task :app:testClasses
> Task :app:test
```



build/

- Inputs can be:
  - Files (including outputs from other tasks)
  - Configuration options
- Example: compileJava task inputs are source files and configuration like compile options

```
tasks.withType<JavaCompile> {
    options.isDebug = false
}
```

Incremental Builds: Limitations

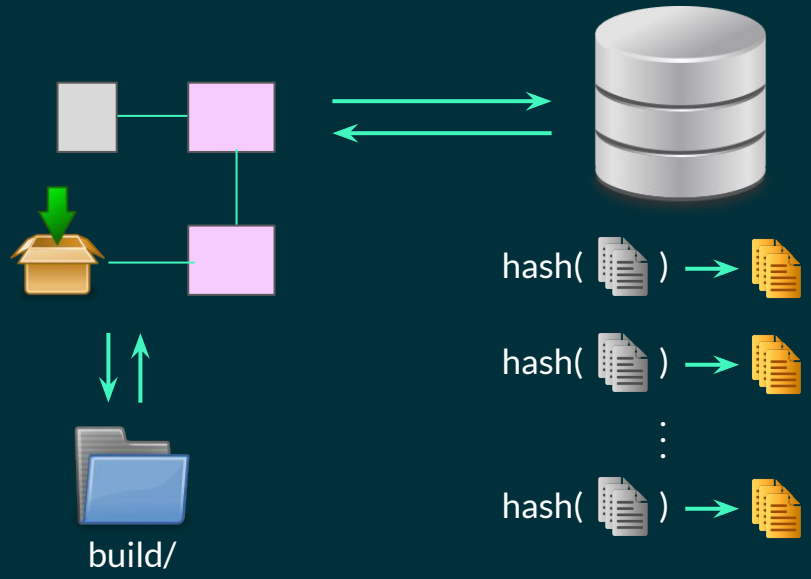
- Can refer to previous build outputs only
- Unsupported use-cases:
  - Switching branches
  - Clean build

Local Cache

- Historical Background ✓
- Build Cache in Gradle
  - Incremental Builds ✓
  - Local Cache
  - Remote Cache
- Enabling with Custom Tasks
- Troubleshooting Cache Misses
- Next Steps

Local Cache

- Store mappings of inputs → outputs
- Create build cache key from inputs
- Use in addition to incremental build support



Local Cache: Enable and Use

- Command-line:
  - ./gradlew compileJava --build-cache
- Update gradle.properties:
  - org.gradle.caching=true
  - Disable with --no-build-cache on command-line
- Debug:
  - ./gradlew compileJava -i
  - ./gradlew compileJava -Dorg.gradle.caching.debug=true

```
Build cache key for task ':app:compileJava' is 9a975a18f505e80f0ec6501e77da0299
```

- Default cache location:
  - ~/.gradle/caches/build-cache-1/
- Order of lookup:
  - Incremental build
  - Local cache

Local Cache: Configuration

- settings.gradle
  - Specify local configuration in buildCache
  - Location
    - Default: ~/.gradle/caches/build-cache-1/
    - Usually don't want to change this
  - Duration
    - Default: 7 days

```
buildCache {
    local {
        directory = File(rootDir, "build-cache")
        removeUnusedEntriesAfterDays = 30
    }
}
```

Local Cache: Limitations

- Can refer to outputs from builds done locally only
- Unsupported use-cases:
  - First build
  - Rapidly changing large project
    - Where you pull often
  - Back from vacation

Checkin Question

- In which file do we configure the retention duration for the Gradle build cache?
    a. gradle.properties
    b. build.gradle.kts
    c. settings.gradle.kts
    d. gradle-wrapper.properties

Checkin Question

- A task's action has executed even though the file inputs did not change, why could this be?
  a. The input → output entry expired in the cache
  b. The task was executed without the build cache option
  c. The task's configuration changed
  d. All of the above are valid reasons

Hands-on Exercise 1

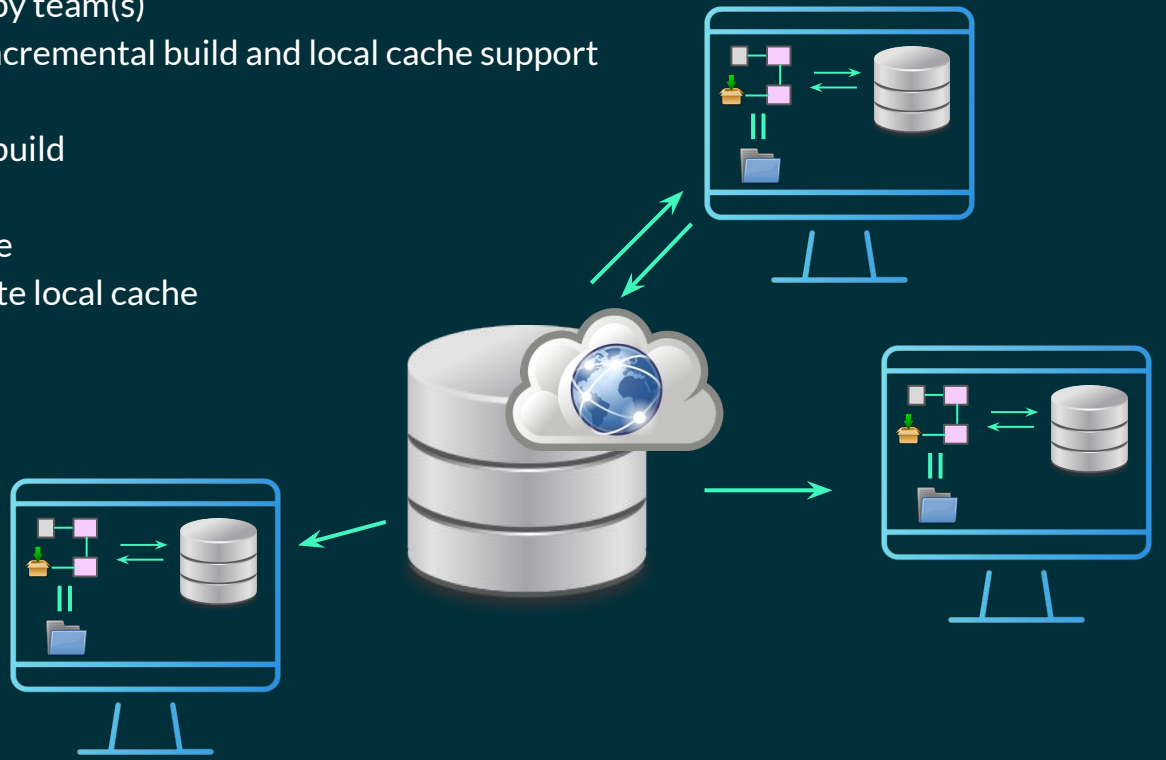
- Recap of Incremental Builds
- Enable and use Local Cache
- Order of lookup for cached outputs

Remote Cache

- Historical Background ✓
- Build Cache in Gradle
  - Incremental Builds ✓
  - Local Cache ✓
  - Remote Cache
- Enabling with Custom Tasks
- Troubleshooting Cache Misses
- Next Steps

Remote Cache

- Shared cache used by team(s)
- Use in addition to incremental build and local cache support
- Order of lookup:
  - Incremental build
  - Local cache
  - Remote cache
- Remote hits populate local cache



Remote Cache: Enable and Use

- settings.gradle
  - Specify remote configuration in buildCache
  - Don’t store credentials as plain-text in checked-in file
  - Use along with build scan for insights

```
buildCache {
  local {
    ...
  }
  remote<HttpBuildCache> {
    url = uri("https://enterprise-training.gradle.com/cache/")
    isPush = true
    credentials {
      username = System.getenv("CACHE_USERNAME")
      password = System.getenv("CACHE_PASSWORD")
    }
  }
}
```

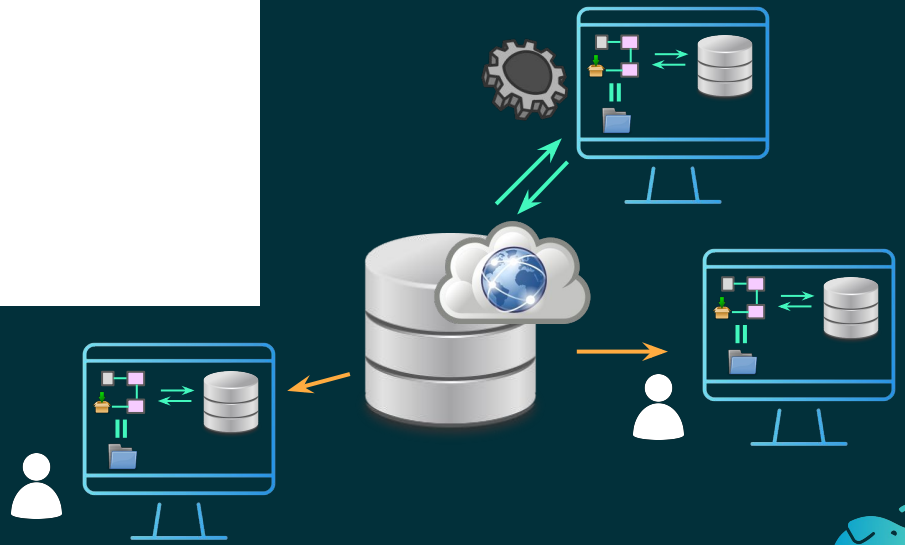
```
plugins {
  id("com.gradle.enterprise") version "3.8"
}
gradleEnterprise {
  buildScan {
    server = "https://enterprise-training.gradle.com"
    capture {
      isTaskInputFiles = true
    }
  }
}
```

Remote Cache: Sharing Strategy

- Only CI machines push to remote cache
  - Should have clean builds running
- Update configuration to specify who can push

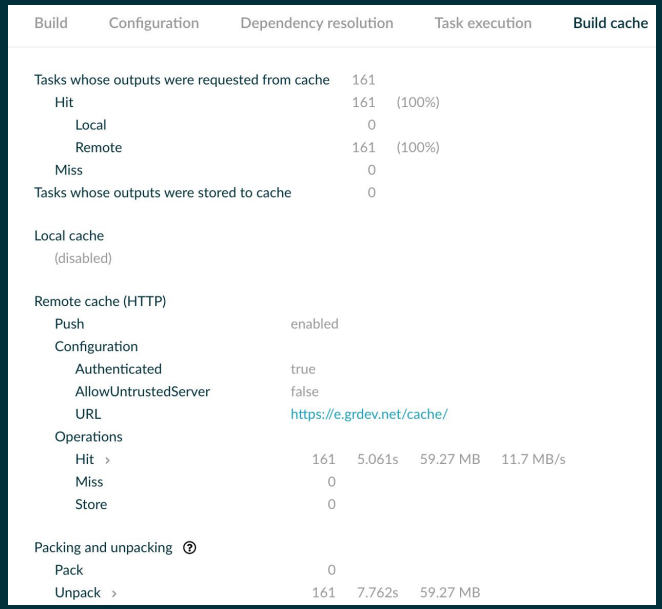
```
val isCiServer = System.getenv().containsKey("CI")

buildCache {
    local {
        ...
    }
    remote<HttpBuildCache> {
        ...
        isPush = isCiServer
    }
}
```



Remote Cache: Docker & Gradle Enterprise

- Docker image available
- Recommend using Gradle Enterprise for professional use
  - Provides high performance and scalable remote cache
  - Monitoring and insights
  - Geo-replicas for maximizing performance
  - Explore:
    - Cache hits/misses
    - Cache miss information
    - Compare build scans



Checkin Question

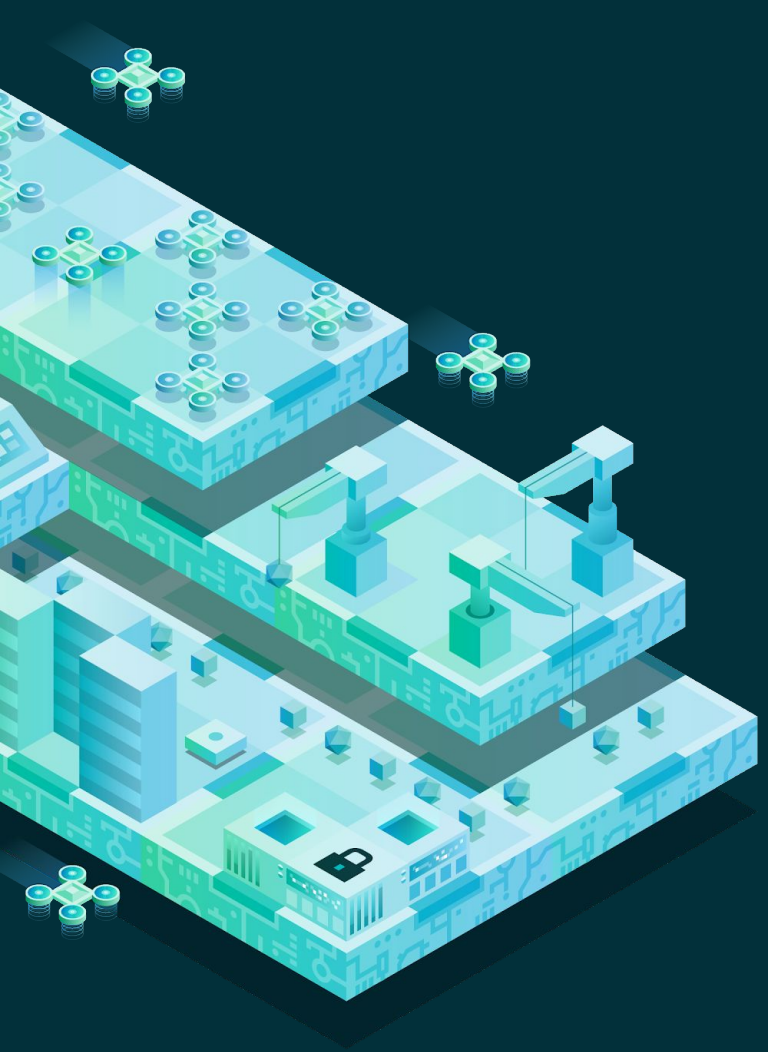
- What is the order in which Gradle looks for cached task output?
    a. Local, Remote, Incremental-build
    b. Local, Incremental-build, Remote
    c. Incremental-build, Remote, Local
    d. Incremental-build, Local, Remote

Checkin Question

- Why is it recommended only CI builds update the Gradle build cache?
  a. CI builds are faster
  b. CI machines are closer to the cache replicas
  c. Due to network access issues
  d. People do test/experimental builds which can clutter up the cache

Hands-on Exercise 2

- Enable and use remote caching
- Compare build scans to identify cause of cache misses



Enabling with Custom Tasks

- Historical Background ✓
- Build Cache in Gradle ✓
  - Incremental Builds ✓
  - Local Cache ✓
  - Remote Cache ✓
- Enabling with Custom Tasks
- Troubleshooting Cache Misses
- Next Steps

Custom Task: Step1 - Wiring Inputs & Outputs

```
abstract class Generate : DefaultTask() {

    @get:Input
    abstract val fileCount: Property<Int>

    @get:InputFile
    @get:PathSensitive(PathSensitivity.RELATIVE)
    abstract val inputFile: RegularFileProperty

    @get:OutputDirectory
    abstract val generatedFileDir: DirectoryProperty

    @TaskAction
    fun perform() {
        for (i in 1..fileCount.get()) {
            val text = File(inputFile.asFile.get().toURI()).readText()
            File(generatedFileDir.get().file("${i}.txt").asFile.toURI()).writeText("${text}-${i}")
        }
    }
}

tasks.register<Generate>("generate") {
    fileCount.set(2)
    inputFile.set(project.file("input.txt"))
    generatedFileDir.set(layout.buildDirectory.dir("gens"))
}
```

- Annotate inputs and outputs
- For file inputs, define path sensitivity
  - Normalization - docs
- Works with incremental build, but not cache
- Built-in cacheable tasks

Custom Task: Step 2: Making Cacheable

- Add annotation to enable caching for the task
- Observe cache key and other details using -i or -Dorg.gradle.caching.debug=true

```
@CacheableTask
abstract class Generate : DefaultTask() {

    ...
}

tasks.register<Generate>("generate") {
    ...
}
```

Custom Task: Using Runtime API

```
tasks.register<Zip>("zipTestResult") {
    // dependsOn("test")
    archiveFileName.set("test-results.zip")
    destinationDirectory.set(layout.buildDirectory)

    from(layout.buildDirectory.dir("test-results"))
    // from(tasks.test) { include("**/*.xml") }
}
tasks.named("zipTestResult") {
    // outputs.cacheIf { true }
    // inputs.files(tasks.test)
}
```

- Tasks using types already wired
- Build cache Runtime API docs
- Specify task dependencies using I/O
  - Inferred task dependencies
  - Linking outputs to inputs

```
tasks.register<Exec>("helloFile") {
    workingDir = layout.buildDirectory.asFile.get()
    commandLine("bash", "-c", "person=`cat ../name.txt`; echo \"hello \$person\" > hello.txt")

    /* outputs.cacheIf { true }

    inputs.file(layout.projectDirectory.file("name.txt"))
            .withPropertyName("helloInput")
            .withPathSensitivity(PathSensitivity.RELATIVE)

    outputs.file(layout.buildDirectory.file("hello.txt"))
            .withPropertyName("helloOutput")*/
}
```

Custom Task: Adding to Cache Key

- Can use Runtime API to customize cache key
- Observe cache key using -i or -Dorg.gradle.caching.debug=true

```
import java.net.InetAddress

...

@CacheableTask
abstract class Generate : DefaultTask() {

    ...
}

tasks.register<Generate>("generate") {
    ...

    inputs.property("hostName", InetAddress.getLocalHost().hostName)
    inputs.property("hostAddress", InetAddress.getLocalHost().hostAddress)
}
```

Hands-on Exercise 3

- Enable caching for Custom Task with existing wiring for inputs and outputs
- Using Runtime API to wire inputs and outputs

- Before starting - quick note on generateLocalUniqueValue task

Troubleshooting Cache Misses

- Historical Background ✓
- Build Cache in Gradle ✓
  - Incremental Builds ✓
  - Local Cache ✓
  - Remote Cache ✓
- Enabling with Custom Tasks ✓
- Troubleshooting Cache Misses
- Next Steps

Troubleshooting Cache Misses: Example

- Dependent project uses timestamp in manifest
- The test task will always execute

```
import java.time.format.DateTimeFormatter
import java.time.Instant

tasks.jar {
    manifest {
        attributes(
                "Implementation-Timestamp" to DateTimeFormatter.ISO_INSTANT.format(Instant.now())
        )
    }
}
```

- Build scan will show the reason
- Normalize to exclude single attribute

```
normalization {
    runtimeClasspath {
        metaInf {
            ignoreAttribute("Implementation-Timestamp")
        }
    }
}
```

Troubleshooting Cache Misses

- Task not caching properly?
  - Check cache key differences
  - Build scan or Gradle Enterprise insights
- Possible issues:
  - Absolute paths instead of relative
  - Inputs contain timestamps
  - Collection input has non-deterministic ordering
  - Overlapping outputs
  - External inputs, eg. system properties
  - File encoding
  - Line endings
  - Symlinks
  - Java version
    - Default behavior only tracks major version
    - Recommend using toolchains

Troubleshooting Cache Misses: Normalization

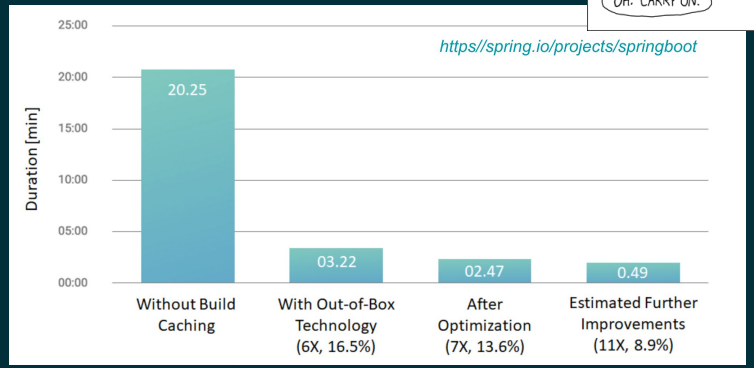
- Tell Gradle to ignore unnecessary information from inputs that keeps changing
  - Relative paths for inputs
  - Content normalization
    - eg. Ignoring timestamps
  - Normalization docs
- Alternatively make inputs deterministic
  - Not always an option

Next Steps

- Historical Background ✓
- Build Cache in Gradle ✓
  - Incremental Builds ✓
  - Local Cache ✓
  - Remote Cache ✓
- Enabling with Custom Tasks ✓
- Troubleshooting Cache Misses ✓
- Next Steps

Next Steps: Try it Out!

- Enable local cache
  - Configure retention duration
  - org.gradle.caching=true in gradle.properties
- Configure custom tasks to be cacheable
  - JVM and Android builds should work out-of-the-box
- Remote caching for team
  - Docker if small scale and have time
  - Gradle Enterprise for professional setting
  - Only CI machines push to remote cache



Summary

- Historical Background ✓
- Build Cache in Gradle ✓
  - Incremental Builds ✓
  - Local Cache ✓
  - Remote Cache ✓
- Enabling with Custom Tasks ✓
- Troubleshooting Cache Misses ✓
- Next Steps ✓

Thank you!

Objectives

- Understand benefits and how to use Gradle Build Cache
- Hands-on exercises to get you going

Resources

- https://docs.gradle.org/
- https://discuss.gradle.org/
- https://newsletter.gradle.com/
- https://plugins.gradle.org/
- https://gradle-community.slack.com/

Feedback

- training@gradle.com