# Build Cache Deep Dive

Improving the Developer Experience with Faster Feedback Cycles

## About the trainer

## Prequisites

- Skills
  - Good understanding of Java language
  - Basic understanding of Gradle concepts
- Tools
  - Java 8
  - Latest Gradle version

## Training content

- Build cache in context
- Understand the benefits of using the Gradle build cache
- Use and configure the build cache
- Tune build logic for maximum cacheability
- Maximize the benefits with Gradle Enterprise

## Training material

- Gradle Enterprise training instance
  @ https //enterprise-training.gradle.com
- Zip with hands-on labs and slides
  @ https //enterprise-training.gradle.com/build-cache-deep-dive

## Build scans

- Gathers details about build
- Generated and published with `--scan`
- Captures IP address and host name
- Published informa on will be publicly-available
- Can be deleted manually with minus icon in toolbar at the top

## Performance is key

- Faster build mes lead to faster feedback
- Faster feedback leads to better developer produc vity
- Better developer produc vity ships features quicker
- See blog post Quan fying the costs of builds

## Build cache in context

Build Cache is a key enabling technology for a new software development discipline called Developer Produc vity Engineering (DPE). DPE uses data analy cs and accelera on technologies to speed up software development processes—from builds to tes ng to CI—and make them more efficient.

- Faster feedback cycles
- More reliable and ac onable data
- A highly sa sfying developer experience

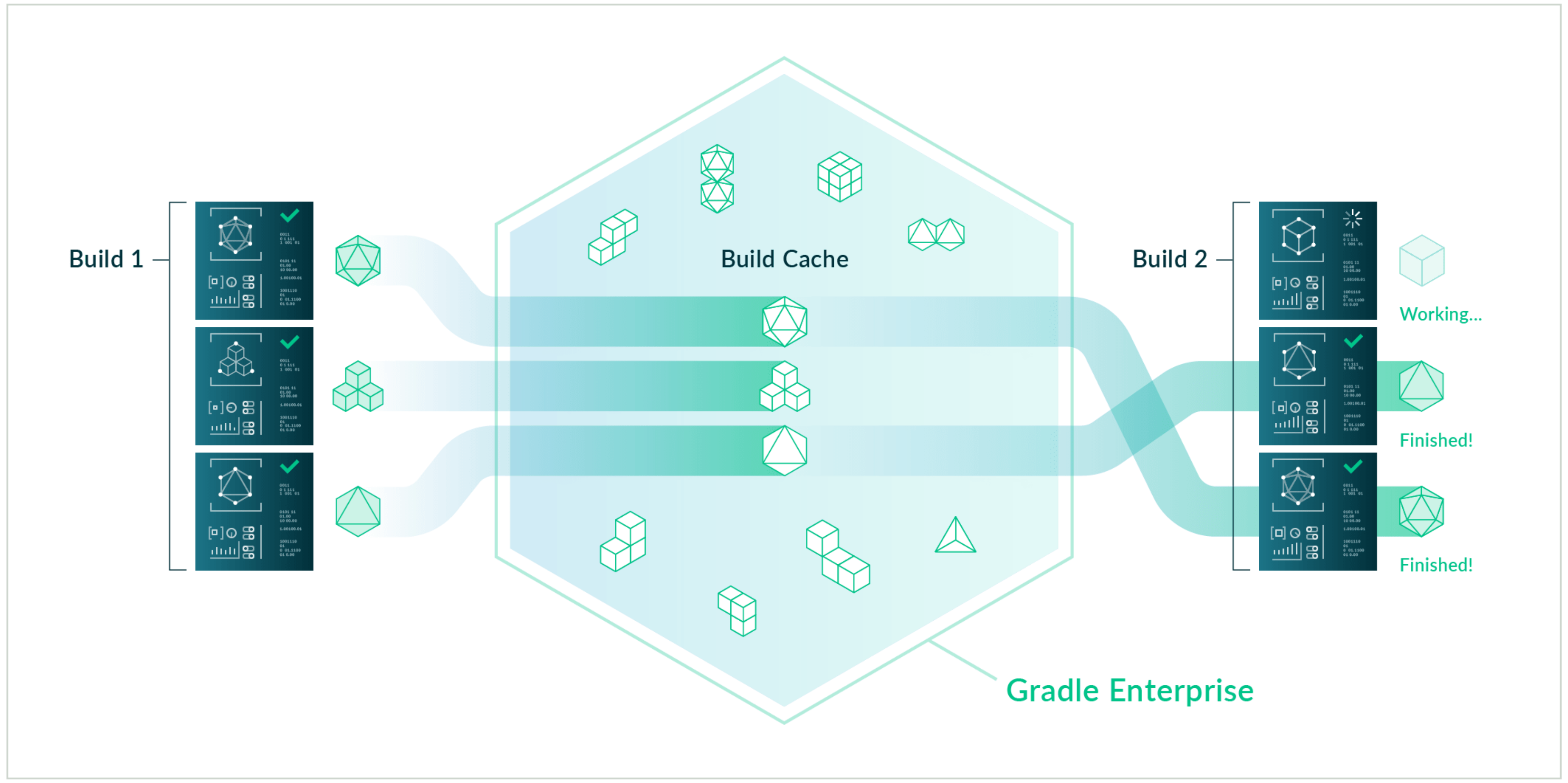
# Developer produc vity engineering solu on framework

| | | | | | |
|---|---|---|---|---|---|
| DPE addresses 5 primary **pain points**… | **IDLE/WAIT TIME** | **INEFFICIENT DEBUGGING** Build, test, CI | **FLAKY TEST & OTHER AVOIDABLE FAILURES** | **NO METRICS & KPI OBSERVABILITY** | **INEFFICIENT USE OF CI RESOURCES** |
| In order to deliver 5 major **benefits**… | FASTER FEEDBACK CYCLES | FASTER DEBUGGING | RELIABLE BUILDS & TESTS | CONTINUOUS LEARNING & IMPROVEMENT | CI COST & RESOURCE EFFICIENCY |
| The platform consists of 5 key **solutions…** | **BUILD & TEST ACCELERATOR** | **ROOT CAUSE ANALYSIS DATA** | **FAILURE ANALYTICS** | **TREND DASHBOARDS** | **CI RESOURCE OPTIMIZATION** |

❖ Build Cache

## Approaches for build avoidance

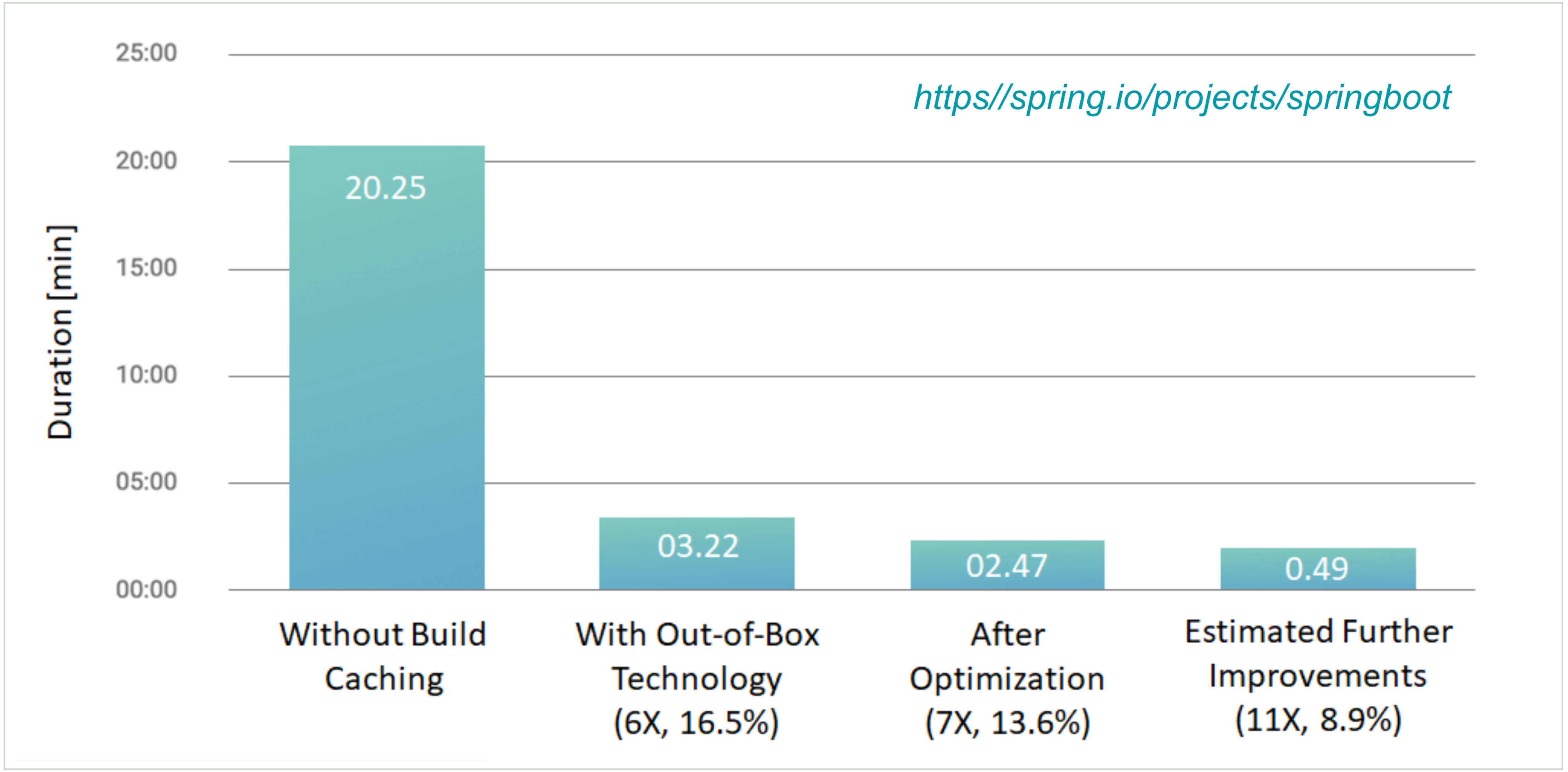
- **Incremental build** build avoidance in same workspace
- **Build cache** share build results across mul ple workspaces



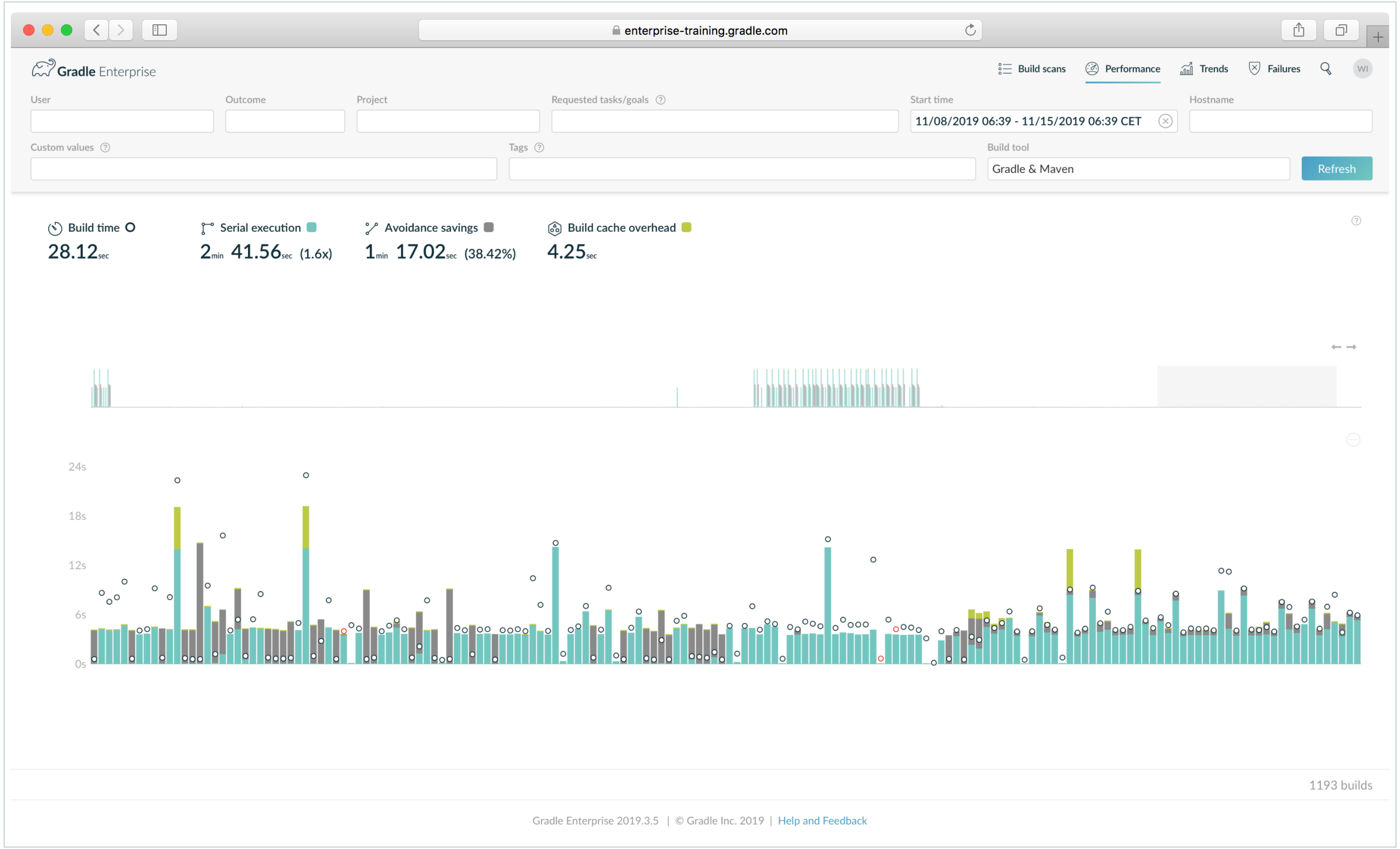
## Common use cases

- Speed up developers' builds when switching branches
- Share results between CI builds
- Accelerate developer builds by reusing CI results
- Driving posi ve developer behavior change with faster build & test feedback cycles

## Spring Boot build me for compile & unit tests



# Visualized savings in Gradle Enterprise



enterprise-training.gradle.com

Gradle Enterprise

Build scans | Performance | Trends | Failures

User:
Outcome:
Project:
Requested tasks/goals:
Start time: 11/08/2019 06:39 - 11/15/2019 06:39 CET
Hostname:
Custom values:
Tags:
Build tool: Gradle & Maven

Refresh

Build time: 28.12 sec
Serial execution: 2 min 41.56 sec (1.6x)
Avoidance savings: 1 min 17.02 sec (38.42%)
Build cache overhead: 4.25 sec

1193 builds

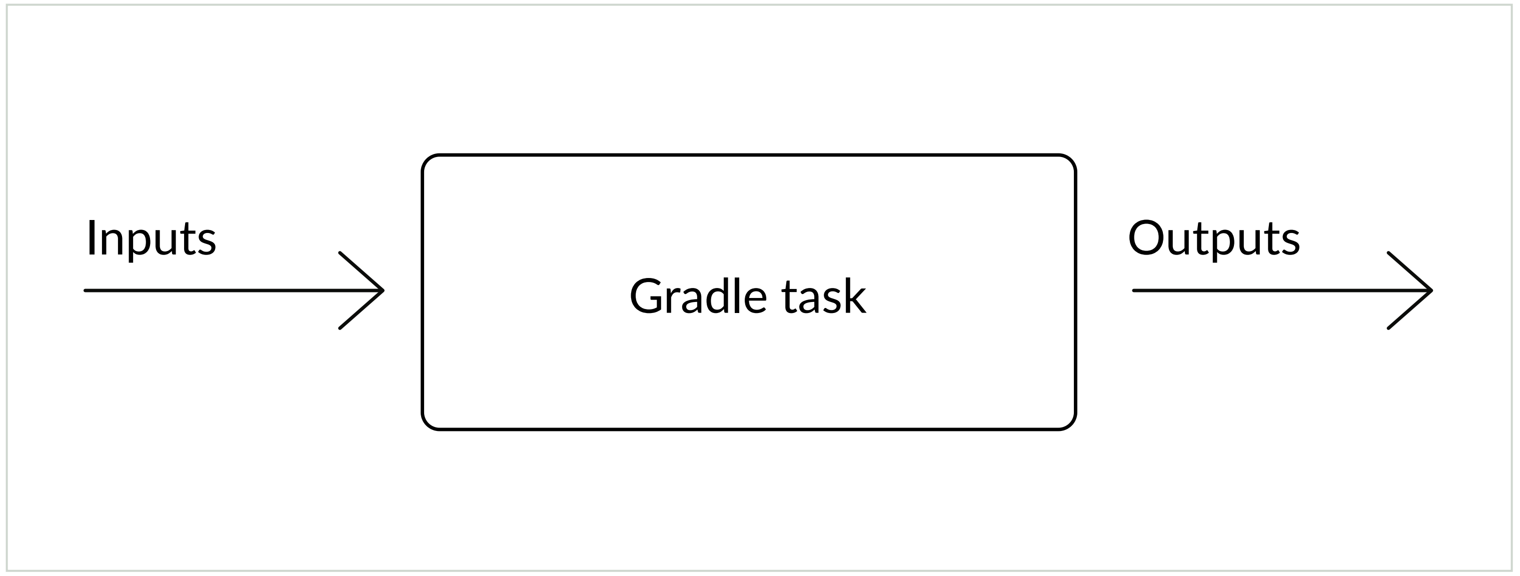
Gradle Enterprise 2019.3.5 | © Gradle Inc. 2019 | Help and Feedback

## Recap  incremental builds

- Important feature from the beginning
- Op  mized for single developer running the build
- Underlying mechanism for the build cache

## How does it work?

- Task needs to define inputs and outputs
- Hashes of inputs and outputs are stored on disk
- Ac  ons are only executed if inputs and/or outputs have changed



## Execu on marker in console

- Gradle marks task **`UP-TO-DATE`**
- Build summary indicates high-level sta s cs

```
$ gradle compileJava --console=verbose
:compileJava UP-TO-DATE

BUILD SUCCESSFUL in 0s
1 actionable task: 1 up-to-date
```

## Declaring inputs and outputs with annota ons

*Generate.groovy*

```
class Generate extends DefaultTask {
    @Input
    int fileCount = 10

    @OutputDirectory
    File generatedFileDir = project.file("${project.buildDir}/generate

    @TaskAction
    void perform() {
        for (int i=0; i<fileCount; i++) {
            new File(generatedFileDir, "${i}.txt").text = i
        }
    }
}
```

Assign annota ons to task proper es or getter methods for all of your custom task implementa ons.

## Declaring inputs and outputs with run me API

*build.gradle*

```
generate {
    inputs.property 'fileCount', 10
    outputs.dir project.file("${project.buildDir}/generated")
}
```

Use run me task API (see `TaskInputs` and `TaskOutputs`) if task source code cannot be changed easily.

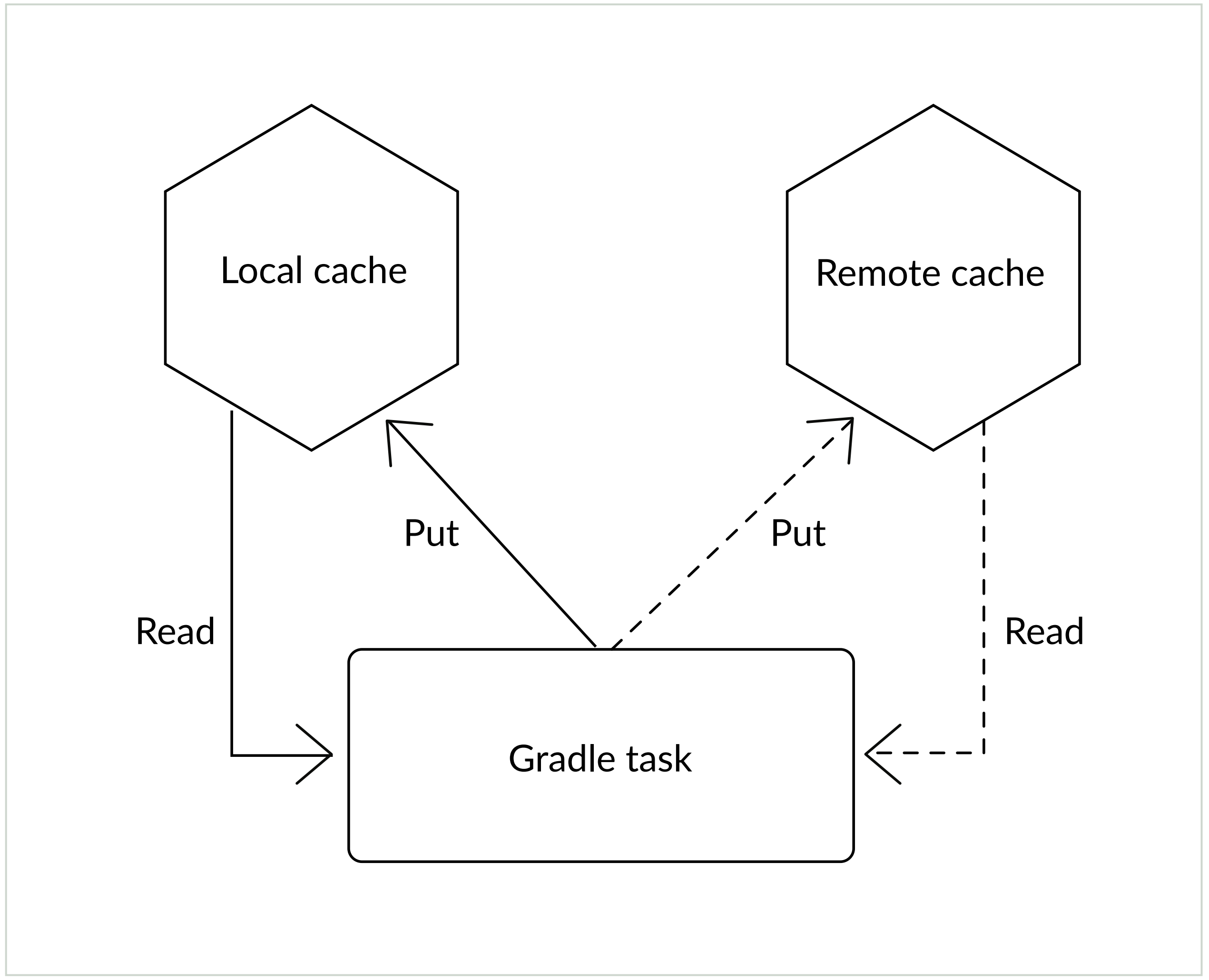
## What are the limita ons?

- Only uses the result of the previous execu on
- Restricted to execu on on single machine
- Cache is not shared among team members

## What is the build cache?

- Reuse build outputs of *any* previous execu on
- Reuse build outputs even if run with **`clean`** task
- Uniquely iden fies outputs of tasks by inputs
- First stable version with Gradle 4.0

# Different types of build caches



## Local build cache

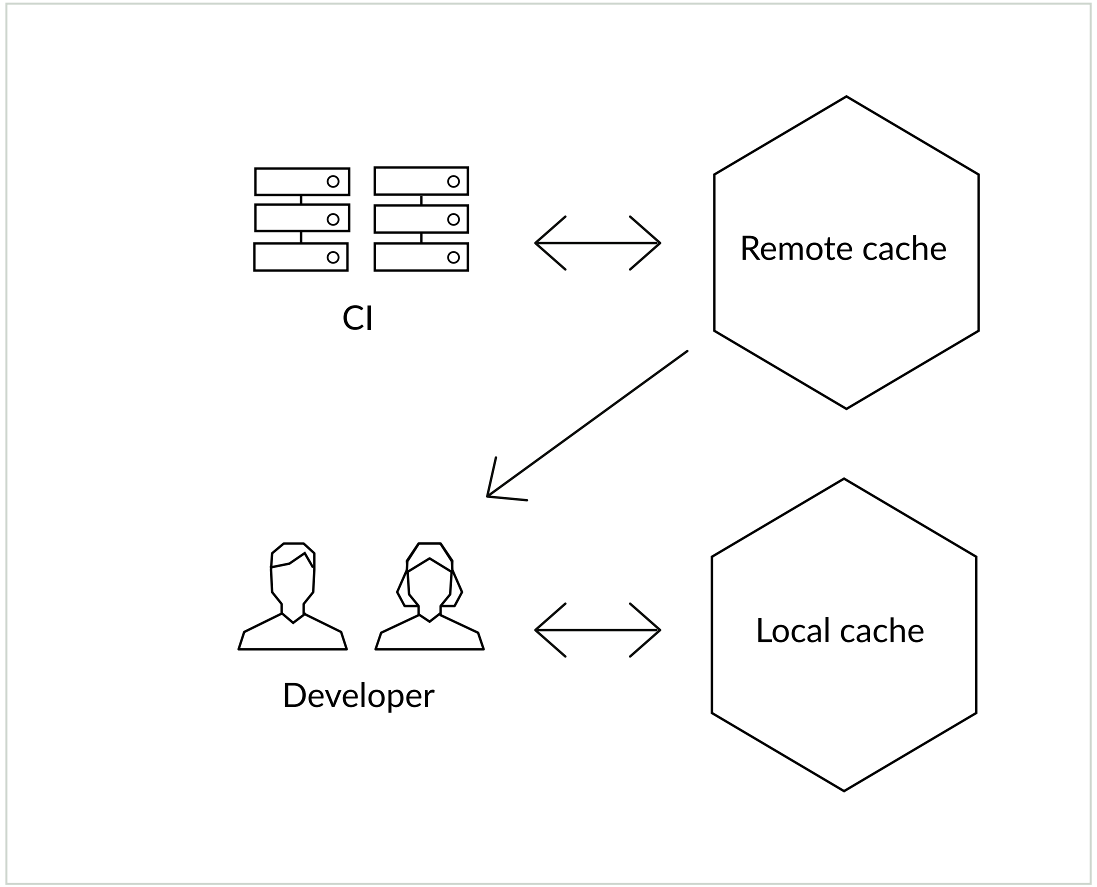
- Uses cache in directory on local machine
- Speeds up development for single developer or build agent
- Reuses build results when switching branches locally
- Par  cularly useful for Android variants

## Remote build cache

- Shared among different machines
- Speeds up development for the whole team
- Reuses build results among CI agents/jobs and individual developers

## Recommended sharing strategy

- only push to the shared cache from CI
- avoid sharing from developer machines



## Influencing factors

- Architecture of code
- Nature of change
- Are tasks cacheable?
- Do outputs change with every build?

## Using the build cache

## Enabling the cache

- This build invoca on only **`--build-cache`** command line op on
- All build invoca ons **`org.gradle.caching=true`** in **`gradle.properties`**
- The **`buildSrc`** project needs to be explicitly enabled for cacheability

```
gradle --build-cache clean assemble
```

## *Lab 01*

Using the local build cache

## Configuring the local build cache

*se ngs.gradle*

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

Domain class for configuring local cache **`DirectoryBuildCache`**

## Configuring the remote build cache

*se   ngs.gradle*

```
buildCache {
    remote(HttpBuildCache) {
        url = 'http://example.com:8123/cache/'
        credentials {
            username = 'build-cache-user'
            password = 'some-complicated-password'
        }
    }
}
```

Domain class for configuring remote cache **`HttpBuildCache`**

## Condi onal cache configura on

*se ngs.gradle*

```
def ciServer = System.getenv().containsKey('CI')

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

## Standardizing build cache configura on

*init.gradle*

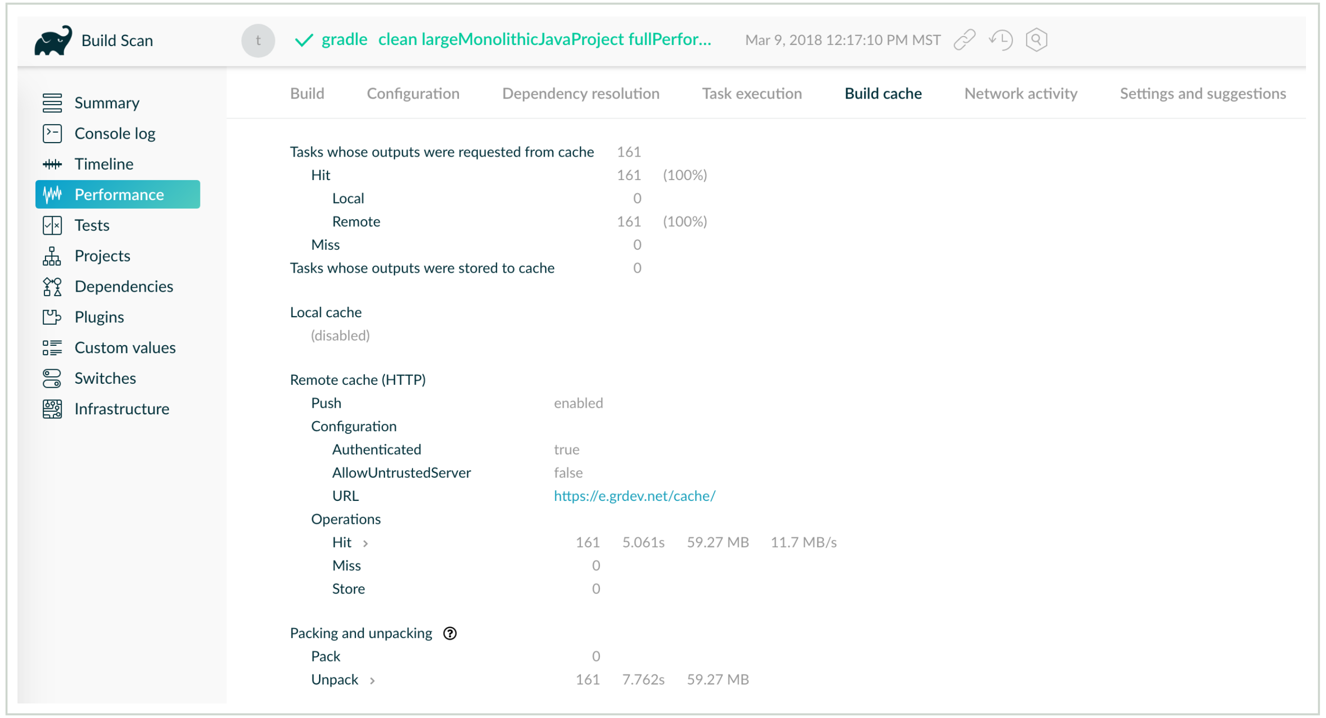
```
def ciServer = System.getenv().containsKey('CI')

gradle.settingsEvaluated { settings ->
    settings.buildCache {
        local {
            enabled = !ciServer
        }
        remote(HttpBuildCache) {
            url = 'https://example.com:8123/cache/'
            push = ciServer
        }
    }
}
```

## Compu ng the build cache key

- The task implementa on
- The task ac on implementa ons
- The names of the output proper es
- The names and values of task inputs

# Build cache opera ons

- Hit
- Miss
- Store
- Packing
- Unpacking



```
Tasks whose outputs were requested from cache   161
    Hit                                         161   (100%)
        Local                                     0
        Remote                                  161   (100%)
    Miss                                          0
Tasks whose outputs were stored to cache          0

Local cache
    (disabled)

Remote cache (HTTP)
    Push                          enabled
    Configuration
        Authenticated             true
        AllowUntrustedServer      false
        URL                       https://e.grdev.net/cache/
    Operations
        Hit ›                        161   5.061s   59.27 MB   11.7 MB/s
        Miss                           0
        Store                          0

Packing and unpacking ⓘ
    Pack                               0
    Unpack ›                         161   7.762s   59.27 MB
```

## *Lab 02*

Using the remote build cache

# What makes a task  cacheable ?

- Task needs to define inputs and outputs
- Task type implementa  on needs to declare `@CacheableTask` annota  on
- `@CacheableTask` is not inherited by subclasses
- Custom task types have to opt into cacheability

## Cacheability influencing factors

- Declared inputs and outputs
- Repeatable output
- Relocatability vs. absolute paths

## Built-in cacheable tasks

- Some but not all built-in Gradle tasks are cacheable
- Tasks involving copy opera  ons are usually not cacheable

## Enabling cacheability by annota on

*Generate.groovy*

```
@CacheableTask
class Generate extends DefaultTask {
    @Input
    int fileCount = 10

    @OutputDirectory
    File generatedFileDir = project.file("${project.buildDir}/generate

    @TaskAction
    void perform() {
        for (int i=0; i<fileCount; i++) {
            new File(generatedFileDir, "${i}.txt").text = i
        }
    }
}
```

Only applicable to custom task implementa ons!

## Enabling cacheability by run  me API

*build.gradle*

```
generateCode {
    outputs.cacheIf {
        // return boolean expression
    }
}
```

Ad-hoc tasks or tasks from plugins can determine cacheability via **`TaskOutputs.cacheIf(Spec)`**.

## Disabling cacheability by run me API

*build.gradle*

```
generateCode {
    outputs.doNotCacheIf('Actions produce volatile results') {
        true
    }
}
```

Disabling the cache for a task with **`TaskOutputs.doNotCacheIf(String, Spec)`** requires providing a reason.

## *Lab 03*

Equipping tasks with caching capabili es

## Troubleshoo ng the build cache

## Possible approaches

- Low-level troubleshoo ng
  - Iden fy task outcome with `--console=verbose`
  - Retrieve cache key informa on by changing the log level
  - Compare cache keys and root causes
- Visual and convenient troubleshoo ng
  - Create a build scan
  - Use GE deep insight features

## Info log level console informa on

```
$ gradle helloWorld --build-cache -i

> Task :helloWorld UP-TO-DATE
Build cache key for task ':helloWorld' is 16f4fbc007345a854d49302279d1
```

Info log level displays cache key generated for each task.

## Debug log level console informa  on

```
$ gradle helloWorld --build-cache -Dorg.gradle.caching.debug=true

> Task :helloWorld UP-TO-DATE
Appending taskClass to build cache key: HelloWorld_Decorated
Appending classLoaderHash to build cache key: 575dae0f1414d5dfd4ef14b6
Appending actionType to build cache key: HelloWorld_Decorated
Appending actionClassLoaderHash to build cache key: 575dae0f1414d5dfd4
Appending inputPropertyHash for 'message' to build cache key: f81fd65e
Appending outputPropertyName to build cache key: outputFile
Build cache key for task ':helloWorld' is 16f4fbc007345a854d49302279d1
```

Debug log level displays more detailed informa  on.

## Using build scans

- Task input comparison
- Task details (cache key, cacheability reason)
- Determining origin build of cache output
- Performance breakdown

## Requirements for cacheable tasks

## Repeatable task outputs

- Same inputs should produce the same outputs
- Byte-for-byte equivalent or seman  cally equivalent (with normaliza  on)

## Stable task inputs

- Inputs need to be stable over   me
- Poten  al source of vola  lity
  - Timestamps
  - Absolute file paths
  - Non-determinis  c ordering

## Path sensi vity

- File paths for input proper es are absolute by default
- Shared build results between machine requires exact same path
- Controllable via annota on `@PathSensitive`

```
@PathSensitive(PathSensitivity.RELATIVE)
@InputFiles
public FileTree getSources() {
    // ...
}
```

## Input normaliza on

- Task inputs between two execu ons are compared to determine cacheability
- Controllable via annota ons `@Classpath` and `@CompileClasspath`
- Example  For compile classpath Gradle extracts ABI signature from the classes on the classpath
- Configurable to ignore vola le files via `Project.normalization(Action)`

*build.gradle*

```
normalization {
    runtimeClasspath {
        ignore 'build-info.properties'
    }
}
```

## Handling cases affec ng cache correctness

## Overlapping outputs

- Two or more tasks write to the same directory
- Difficult for Gradle to determine which output belongs to which task
- Build scan renders reason for this case

## External inputs like system proper es

- System proper es often use absolute path
- Use rela ve path to fix

## File encoding

- Java tools use the system file encoding when no specific encoding specified
- Can cause incorrect builds
- Always set the file system encoding to avoid issues

## Line endings

- Important when build cache is shared across different OSes
- Set **`autocrlf=false`** if Git is used

## Symlinks

- Symlinks are not stored in build cache
- Uses actual file contents of the des na on of the link
- Some OSes (e.g. Windows) do not support symlinks
- Tasks will not be cacheable across different OSes

## Java versions

- Gradle tracks only the major version of Java as input
- Usually applicable to compila on and test tasks
- Vendor and the minor version may influence the bytecode
- Suggested to add vendor as an input to the corresponding tasks

## *Lab 04*

Handling cache misses

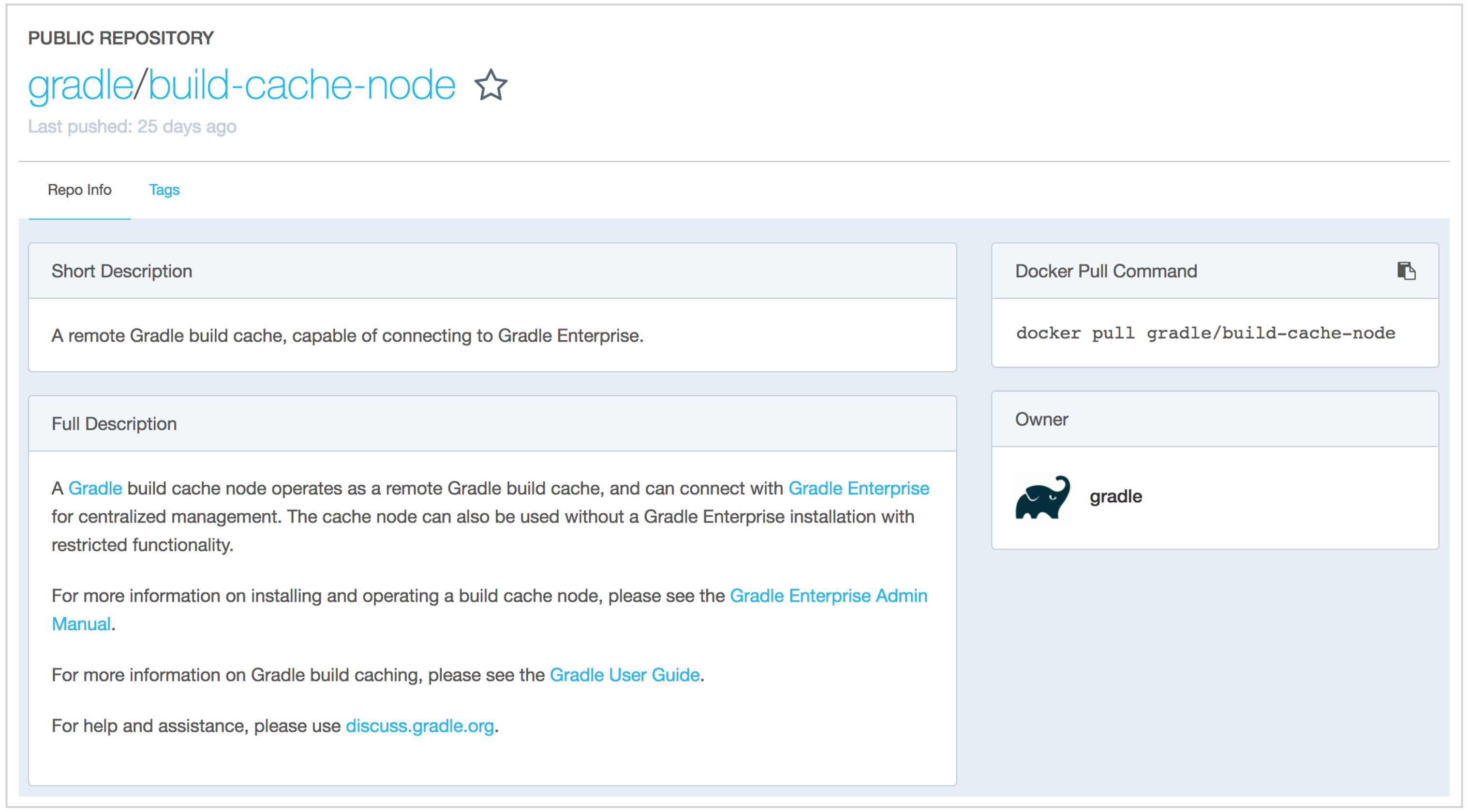
## Ge ng started with the build cache

## Recommended approach

- Equip tasks with inputs and outputs
- Use local build cache
- Set up remote build cache
- Roll out usage to team
- Use Gradle Enterprise for cache monitoring and op  miza  on
- Report from the field  Tableau using Gradle Enterprise

# Installing the remote build cache

- Build cache node available as Docker image or JAR file
- Docker image freely-available from Docker Hub
- Requires Docker installa on on host machine



PUBLIC REPOSITORY

gradle/build-cache-node

Last pushed: 25 days ago

Repo Info Tags

Short Description

A remote Gradle build cache, capable of connecting to Gradle Enterprise.

Full Description

A Gradle build cache node operates as a remote Gradle build cache, and can connect with Gradle Enterprise for centralized management. The cache node can also be used without a Gradle Enterprise installation with restricted functionality.

For more information on installing and operating a build cache node, please see the Gradle Enterprise Admin Manual.

For more information on Gradle build caching, please see the Gradle User Guide.

For help and assistance, please use discuss.gradle.org.

Docker Pull Command

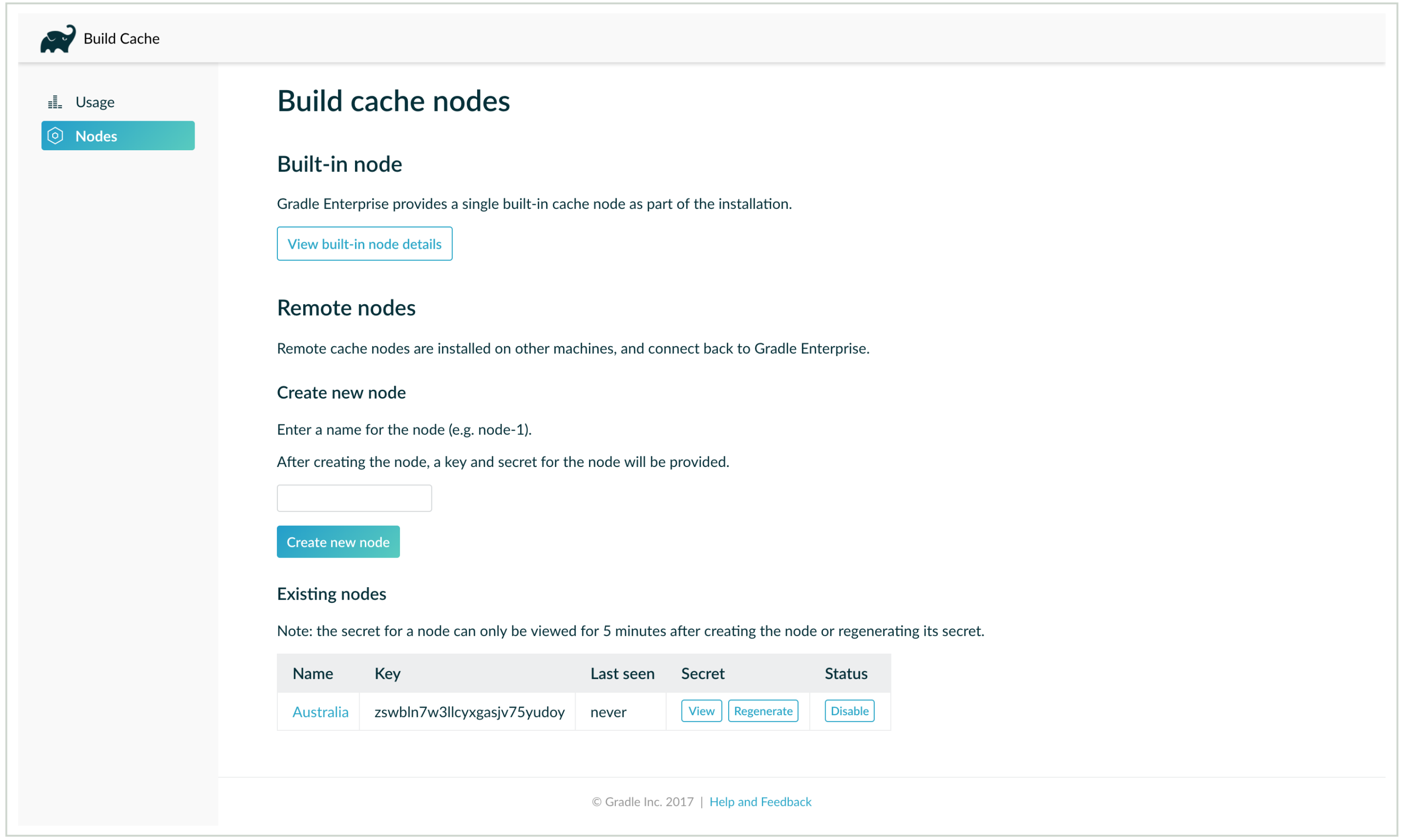
```
docker pull gradle/build-cache-node
```

Owner

gradle

## Connec ng to Gradle Enterprise

- Op onally register with Gradle Enterprise for centralized management
- Replica on capabili es for geographically distributed teams



Build Cache

Usage
Nodes

Build cache nodes

Built-in node

Gradle Enterprise provides a single built-in cache node as part of the installation.

View built-in node details

Remote nodes

Remote cache nodes are installed on other machines, and connect back to Gradle Enterprise.

Create new node

Enter a name for the node (e.g. node-1).

After creating the node, a key and secret for the node will be provided.

Create new node

Existing nodes

Note: the secret for a node can only be viewed for 5 minutes after creating the node or regenerating its secret.

| Name | Key | Last seen | Secret | Status |
|---|---|---|---|---|
| Australia | zswbln7w3llcyxgasjv75yudoy | never | View Regenerate | Disable |

© Gradle Inc. 2017 | Help and Feedback

## Wrap up

## Documenta on and resources

- gradle.com/enterprise/resources
- docs.gradle.org/current/userguide/build_cache.html
- guides.gradle.org/using-build-cache

## Video playlist

To review concepts and learn more about build cache and distributed tes ng, check out our brand new video playlist called Faster Feedback Cycles

tv.gradle.com/build-faster

## Try Gradle Enterprise for free

- Free 30-day trial
  - hosted by Gradle  zero installa  on, ready-to-go
  - on your infrastructure  quick setup, maximum control
- Technical support included

https //gradle.com/enterprise/trial

# Thank you