The importance of work avoidance

Etienne Studer

SVP of Engineering,
Gradle Enterprise

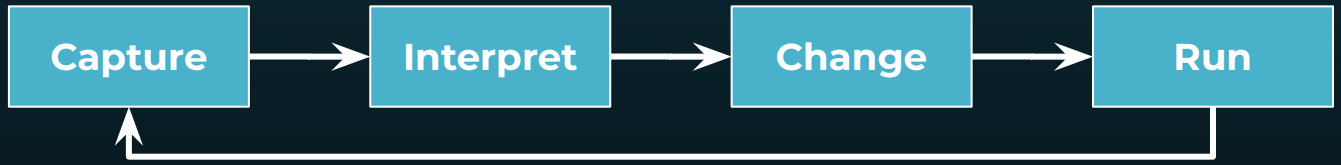
Jim Hurne

Lead Software Engineer,
Gradle Enterprise


Red Bull
Tezos
Tezos
TAG HEUER
1
Tezos
citrix
ARCTIC WOLF
infinitum
Mobil
Red Bull

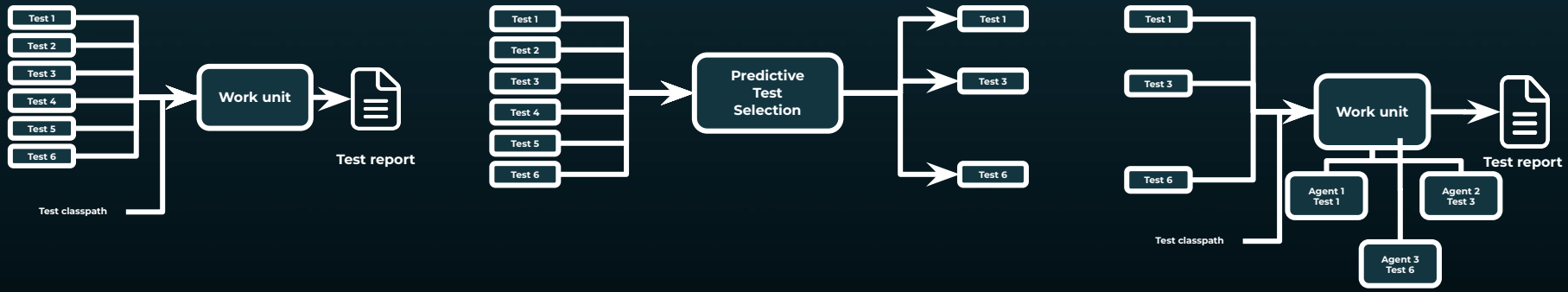
Important factors of DPE

- Capture actionable build metrics
- Understand and improve toolchain behavior
- Accelerate feedback cycles



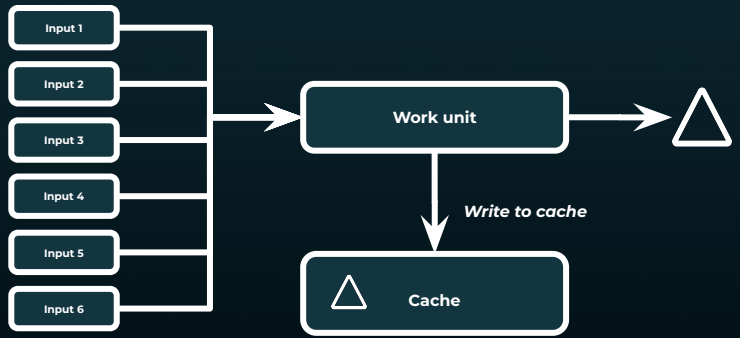
Accelerating feedback cycles

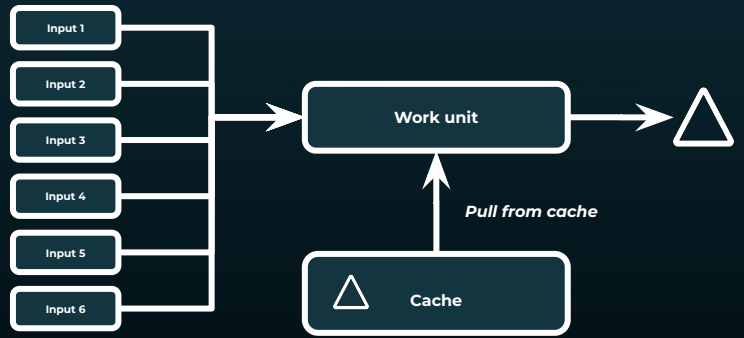
- Avoid work done before by reusing the results of previous work
- Avoid work for unrelated changes by skipping it
- Distribute the work for parallel execution
- Make the work perform faster



Avoiding work done before

- Typical unit of work is a task (Gradle), goal (Maven), or target (Bazel)
- Another unit of work is the Gradle configuration phase
- Output of unit of work is stored in local and/or remote cache
- Does not depend on commit ids or branch names



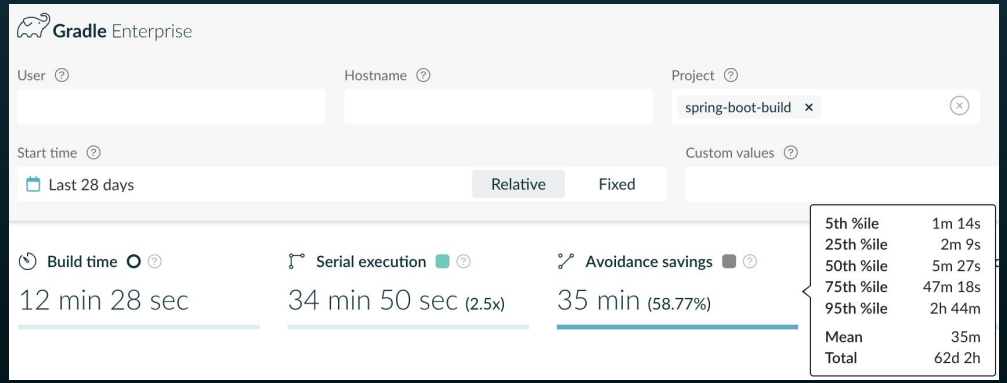


Optimizing for work avoidance

- Typically only some of the work avoidance potential is realized out-of-the-box
- Attaining a fully cacheable build requires investment
- Keeping a build fully cacheable requires investment
- The investment will pay back every time the build is run

Optimizing for work avoidance

- Spring Boot OSS project builds a modest ~100 times per day
- For every 24 hours passed, Spring Boot saves 53 hours or 6.5 FTE of build work

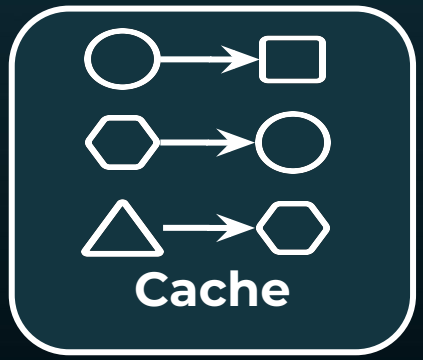


Categorizing build cache misses

 Timestamps	/.../... Absolute paths	 Operating system
 User/host information	1,3,2 Random code ordering	1.6.1 Version numbers

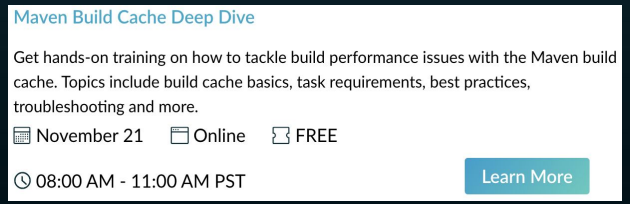
Categorizing build cache misses

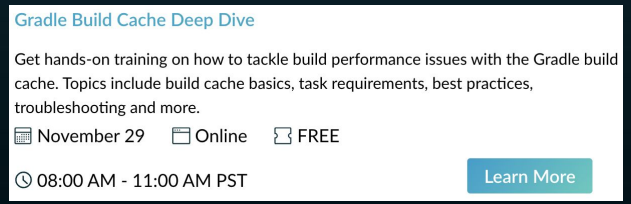
- Volatile inputs often caused by upstream work units creating volatile outputs
- Volatility may be masked by a sequence of cacheable producers and consumers
- Task output tracking is on the Gradle Enterprise roadmap



Taming build cache misses

- Removing volatile inputs
- Changing volatile inputs to stable inputs
- Normalizing volatile inputs





i. the process of optimizing developer productivity should itself be productive

ii. there is always more that we could optimize than we have time and resources available

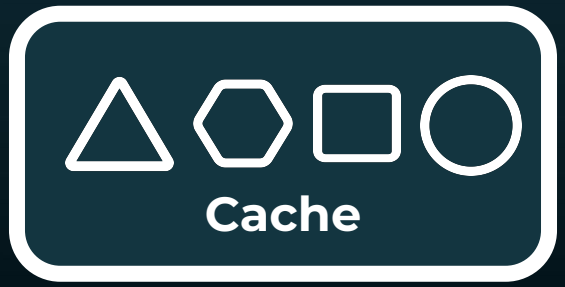
Attaining a fully cacheable build

- Run experiments
  - Measurable
  - Controlled
  - Reproducible
  - Automated

Attaining a fully cacheable build

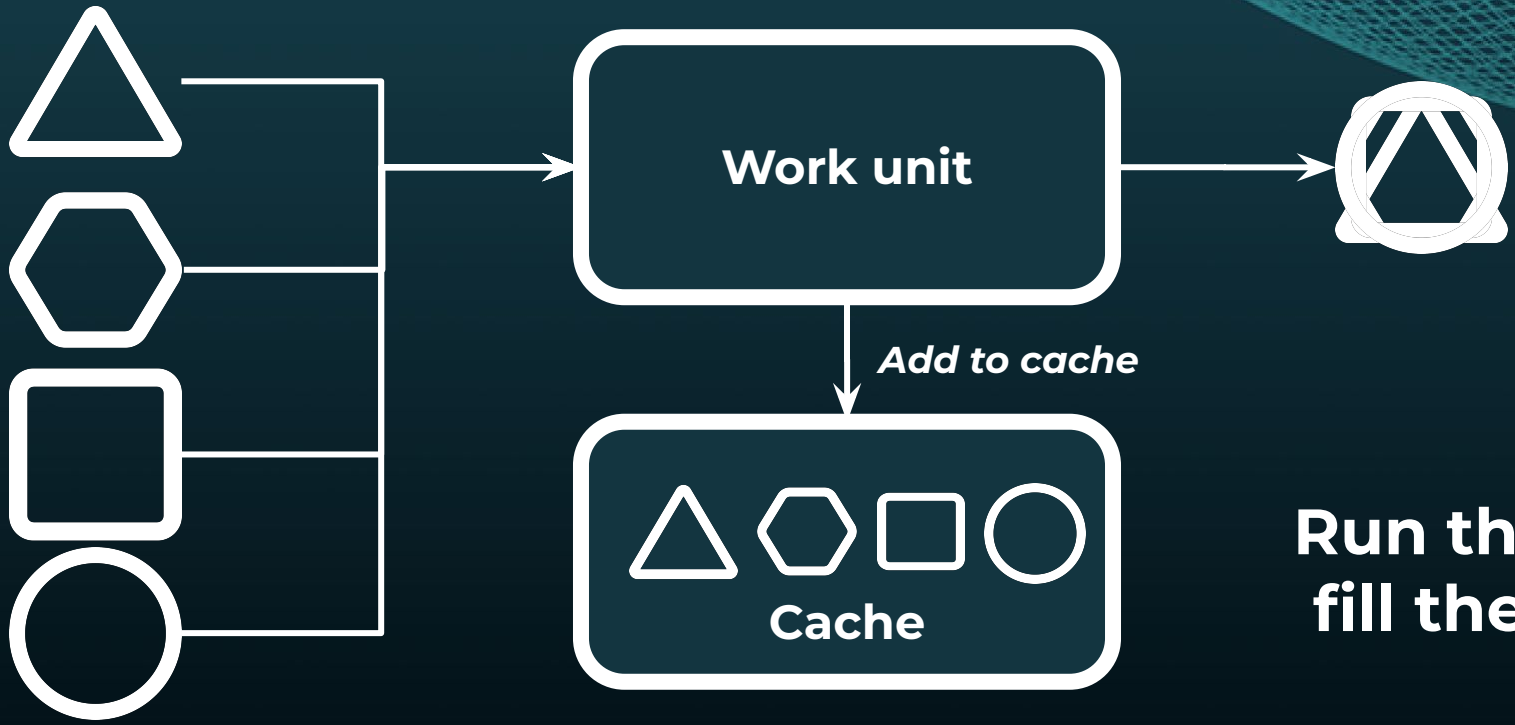
- Leverage build validation scripts, build scans, and build comparison
  - Local – Local experiment
  - CI – CI experiment
  - CI – Local experiment

Attaining a fully cacheable build



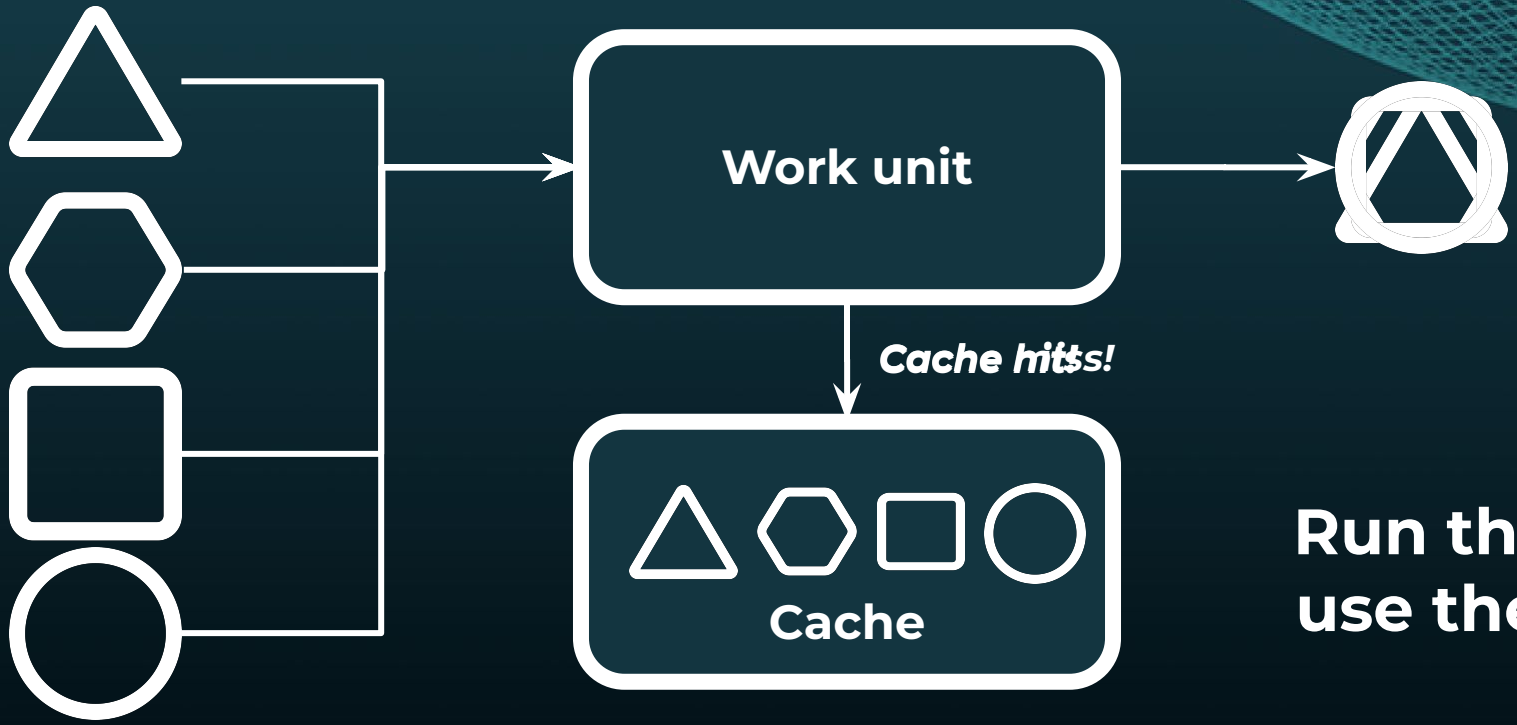
Clear the cache

Attaining a fully cacheable build



Run the build, fill the cache

Attaining a fully cacheable build



Run the build, use the cache

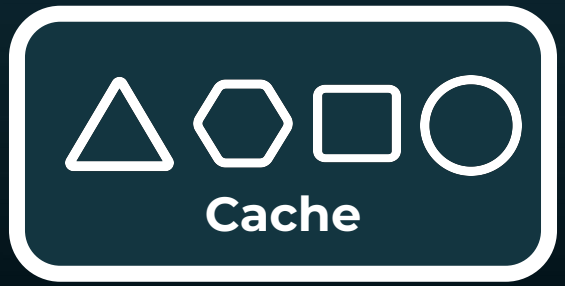
Attaining a fully cacheable build

Cache hit

Cache hit

Cache miss

Cache hit

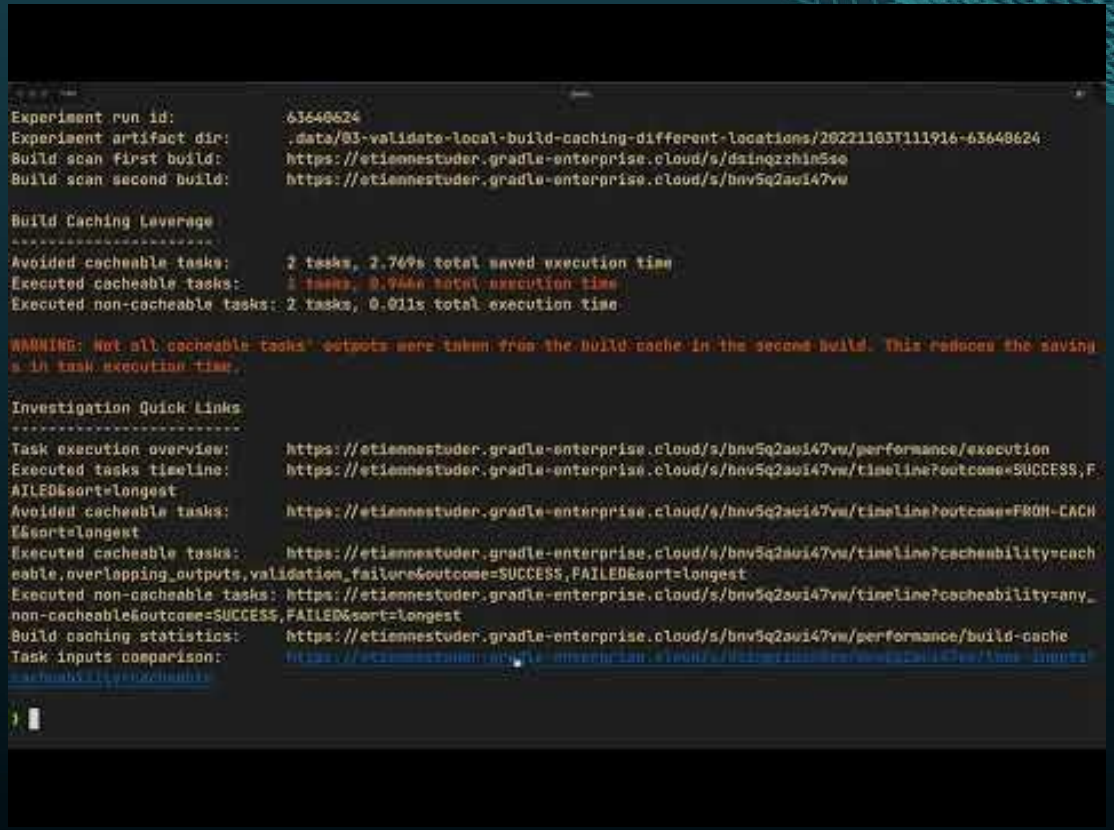


Investigate cache misses

Attaining a fully cacheable build

- Demo
  - Run automated experiment
  - Identify and investigate build cache misses
  - Fix root causes

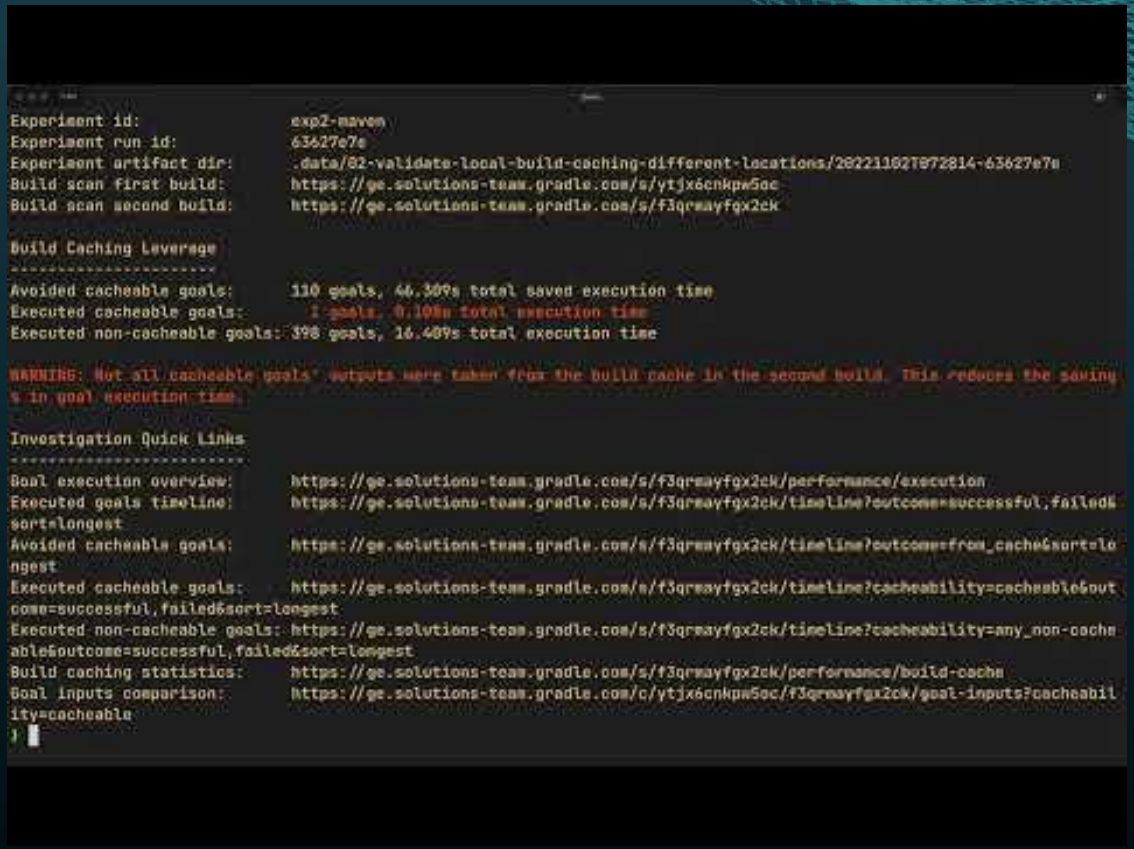
Attaining a fully cacheable build



Attaining a fully cacheable build

- Demo
  - Run automated experiment
  - Connect build ad-hoc to Gradle Enterprise
  - Understand realized and potential work avoidance savings

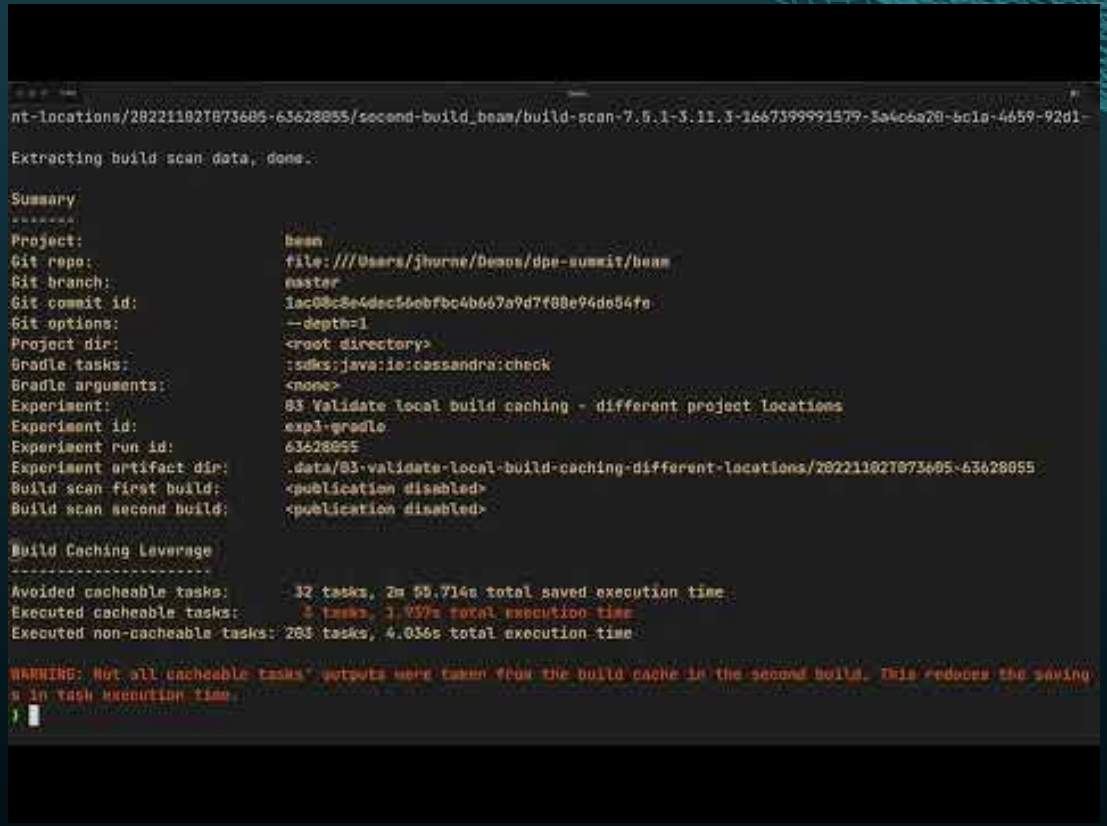
Attaining a fully cacheable build



Attaining a fully cacheable build

- Demo
  - Run automated experiment
  - Capture build data ad-hoc with Gradle Enterprise plugin/extension
  - Understand realized and potential work avoidance savings

Attaining a fully cacheable build



Extrapolating experiment results

- Total work avoidance savings = Realized savings + potential savings
- Max. build time savings ~= (Total work avoidance savings) / Work unit parallelization
- Average build time savings ~= Max. build time savings * (.35 to .65)

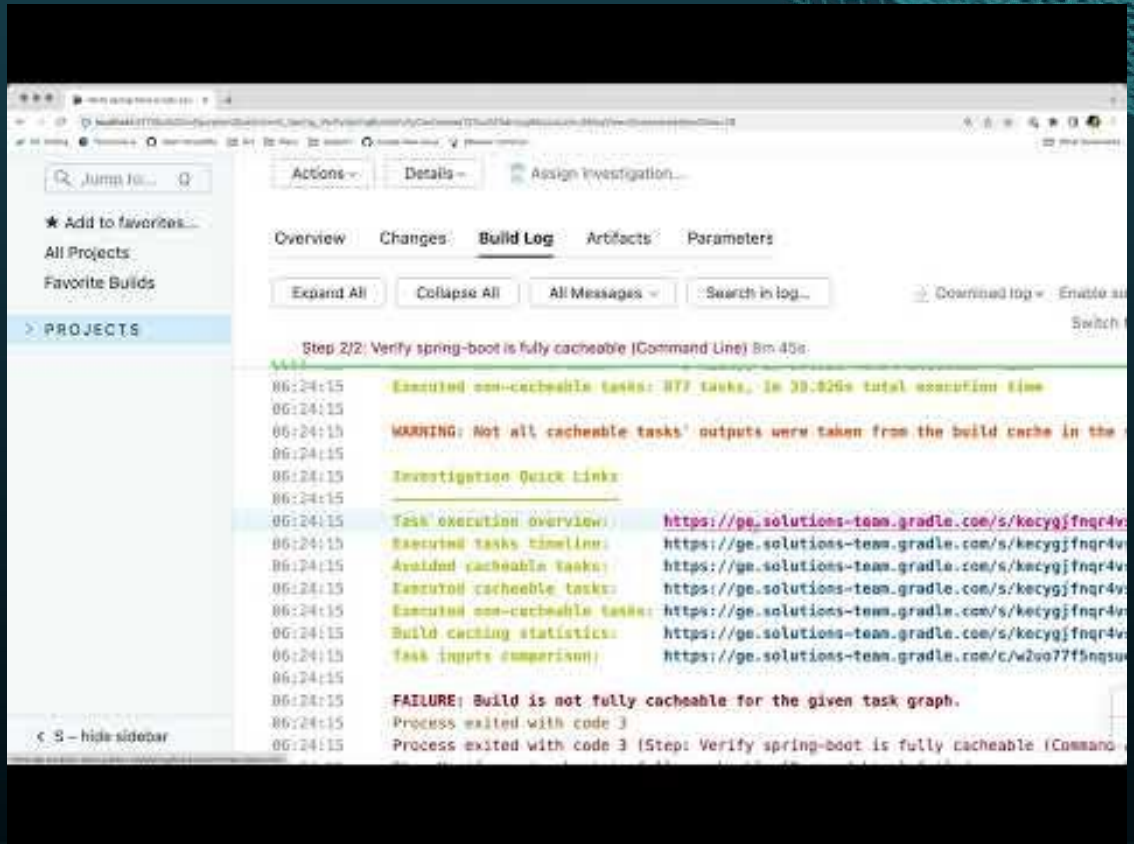
Maintaining a fully cacheable build

1. Use automation on CI to catch cacheability regressions
2. Run automation on a regular basis, triggered by build changes or timer
3. Fail if build not fully cacheable
    - **Run experiments** to catch regressions
    - **Compare historic builds** to catch regressions
    - **Inspect builds for build cache failures** to catch regressions
4. Repair regressions
    - Fix violating work units, make cacheable or mark as non-cacheable
    - Fix build cache access and configuration

Maintaining a fully cacheable build

- Demo
  - Run Local – Local experiments on CI to catch regressions

Maintaining a fully cacheable build



Maintaining a fully cacheable build

- Compare historic builds to catch regressions
  - Same project
  - Same commit id
  - Non-dirty builds
  - Compare cacheable tasks executed in both builds

Maintaining a fully cacheable build

spring-authorization-server	:spring-authorization-server-docs:asciidoctor	https://ge.spring.io/c/kgzhygoftphtq/yf64zqfxta5xe	**Volatile input**
spring-boot-build	:spring-boot-project:spring-boot-actuator-autoconfigure:asciidoctor	https://ge.spring.io/c/6skln6qftnq2g/d6b2kc3yfgir4	**Volatile input**
spring-boot-build	:spring-boot-project:spring-boot-actuator-autoconfigure:asciidoctorPdf	https://ge.spring.io/c/6skln6qftnq2g/d6b2kc3yfgir4	**Volatile input**
spring-boot-build	:spring-boot-project:spring-boot-docs:asciidoctor	https://ge.spring.io/c/dbi36jriqy7im/ha3s5a3rlltj2	**Volatile input**
spring-boot-build	:spring-boot-project:spring-boot-docs:asciidoctorMultipage	https://ge.spring.io/c/dbi36jriqy7im/ha3s5a3rlltj2	**Volatile input**
spring-boot-build	:spring-boot-project:spring-boot-docs:asciidoctorPdf	https://ge.spring.io/c/dbi36jriqy7im/ha3s5a3rlltj2	**Volatile input**
javadoc-plugin	:validatePlugins	https://ge.spring.io/c/vzkmbme7boyuy/6d7mcunkposza	**Eviction**
spring-authorization-server	:spring-authorization-server-docs:api	https://ge.spring.io/c/ejldvuvw7ushw/plvsiuinoiekw	**Eviction**

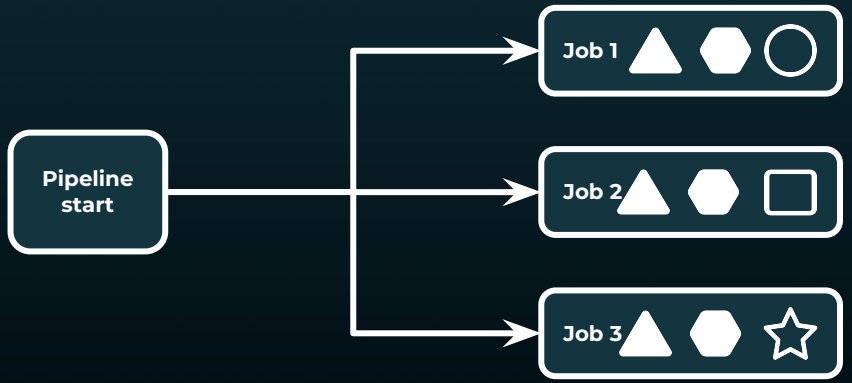
Maintaining a fully cacheable build

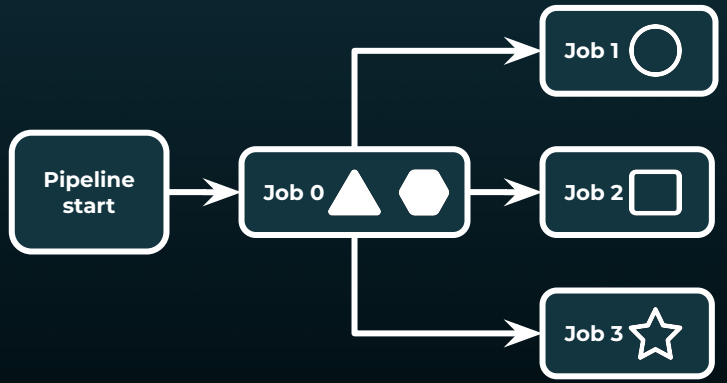
- Inspect builds for build cache failures to catch regressions
  - Build cache access and configuration issues often go unnoticed
  - Example: entity-too-large rejection by build cache disables build caching

Project	# local cache errors	# remote cache errors
spring-boot-build	1	4
spring-security	0	1
spring	0	4

Maintaining a fully cacheable build

- Observations
  - Negative avoidance savings
  - Cache entries that live longer in local cache on CI than in remote cache
  - CI pipelines not optimized to leverage build caching within same run





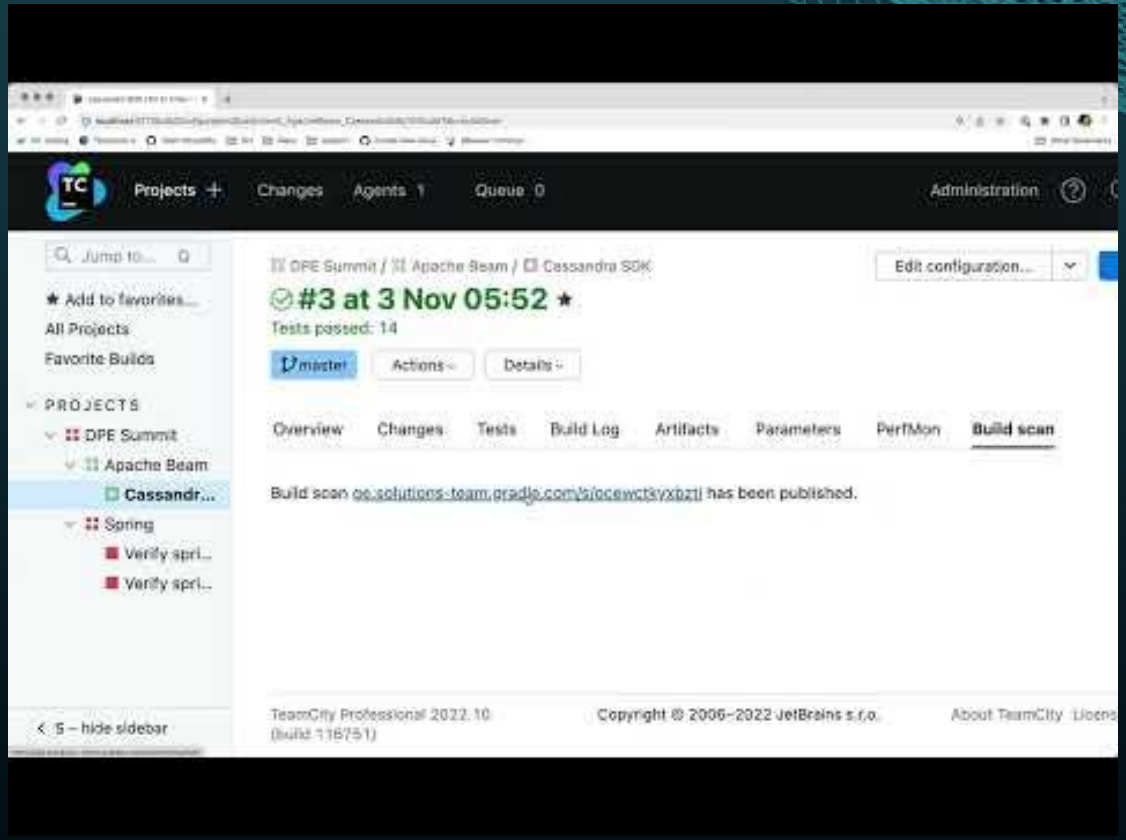
Dealing with many projects

1. Capture build data to make informed decision what projects to optimize
2. Measure potential impact of improving the work units' cacheability
3. Prioritize build improvements by their potential impact

Dealing with many projects

- Demo
  - Instrument all CI builds with build data capturing and remote build caching
  - Available for Jenkins, TeamCity, and soon Bamboo

Dealing with many projects



Dealing with many projects

- Prioritize build improvements by their potential impact

Dealing with many projects

Project	# builds	build time	t (serial)	t(avoided)	t (avoidable)	t (nonav)	% t(avoided)	% t (avoidab	% t (nonav)
spring-boot-build	5568	1196:06:58	3290:30:30	3486:42:49	2786:43:26	345:28:58	51%	41%	5%
spring	2296	122:17:24	398:23:40	455:29:21	340:08:10	16:28:53	53%	40%	2%
spring-security	3198	231:57:56	294:35:52	422:14:39	187:36:35	75:13:32	59%	26%	10%
spring-security-samples	341	41:52:52	78:09:18	3:57:32	59:54:18	15:27:03	5%	73%	19%
spring-session-build	184	23:44:51	35:08:27	0:00:00	0:00:00	35:08:01	0%	0%	100%
spring-kafka-dist	289	18:04:48	24:43:11	3:04:50	23:10:51	1:11:55	11%	83%	4%
spring-authorization-server	556	17:06:49	18:26:59	13:18:35	7:40:54	3:23:34	42%	24%	11%
spring-asciidoctor-backends	8	13:56:32	13:55:12	0:01:26	13:51:52	0:03:16	0%	99%	0%
spring-restdocs	167	11:37:59	12:59:47	47:59:36	2:51:26	8:40:21	79%	5%	14%
spring-session-data-geode-build	186	8:50:32	9:04:59	0:16:02	0:00:00	8:55:11	3%	0%	95%
spring-graphql	228	3:22:18	4:02:55	0:39:20	2:34:07	0:40:40	14%	55%	14%
dependency-management-plugin	26	1:35:59	1:30:45	0:53:54	1:26:42	0:00:55	37%	60%	1%
spring-shell	202	1:45:33	1:08:42	1:00:14	0:42:49	0:12:19	47%	33%	10%
javadoc-plugin	115	0:25:12	0:10:22	0:37:54	0:02:22	0:05:15	79%	5%	11%

Avoiding work is work – but it pays off very quickly!

Etienne Studer

SVP of Engineering,
Gradle Enterprise

Jim Hurne

Lead Software Engineer,
Gradle Enterprise

THANK YOU

Have any questions?

etienne@gradle.com
@etiennestuder
gradle.com/careers