Reproducible Builds with Bazel

GASPARE VITTA

Today’s agenda

01.
What is a reproducible build

02.
How Bazel can be useful

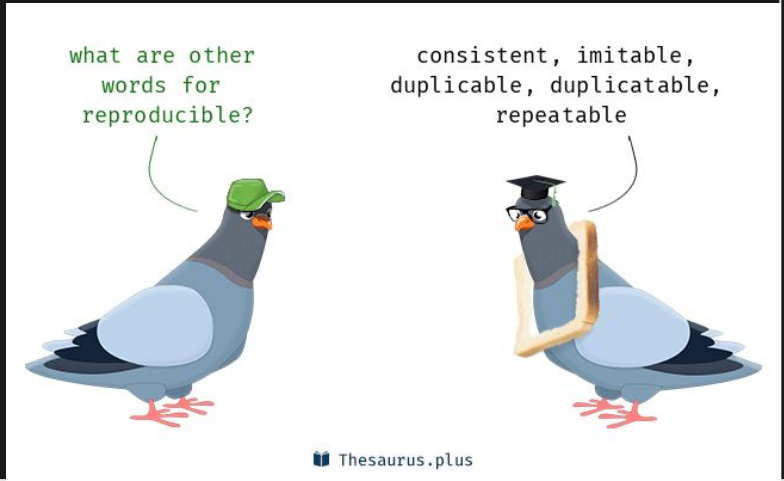
03.
How to create a reproducible Python environment

04.
A real life example

LET’S TAKE A STEP BACK:

What do reproducible means?

- Smaller attack surface
- Known artifact origin
- Faster build thanks to caching



Hermeticity

Unambiguous inputs
to the build

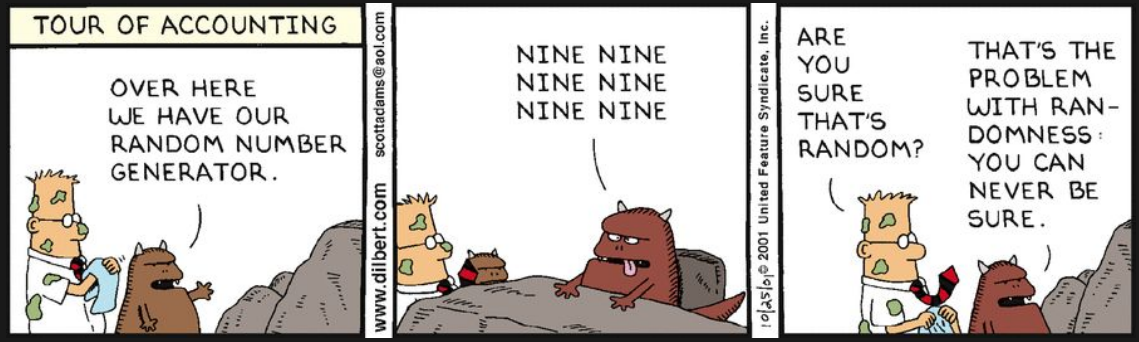
Any information needed by
the build is checked in the
source control

Hermeticity allows
cherry-picking

Internal Randomness

Don’t use timestamps, you don’t need them.

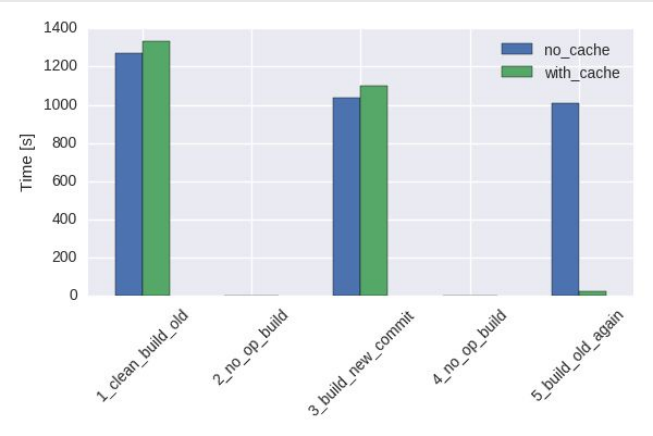
Initialize your variables.



Bazel

“A fast, scalable, multi-language and extensible build system” bazel.build

Hermeticity, Sandboxing, Caching.



https://nicolovaligi.com/faster-bazel-remote-caching-benchmark.html

Bazel concepts

WORKSPACE
BUILD
Bazel Rule
Package
Target

```
├── BUILD
├── WORKSPACE
├── requirements.txt
└── src
    ├── BUILD
    └── flask_app.py
```

```
bazel build //<package name>:target-name
```

WORKSPACE

```
workspace(name = "my_flask_app")

_configure_python_based_on_os = """

if [[ "$OSTYPE" == "darwin"* ]]; then

    ./configure --prefix=$(pwd)/bazel_install --with-openssl=$(brew
--prefix openssl)

else

    ./configure --prefix=$(pwd)/bazel_install

fi

"""
```

WORKSPACE

```
load("@bazel_tools//tools/build_defs/repo:http.bzl"
"http_archive")

# Fetch Python and build it from scratch

http_archive(

    name = "python_interpreter",

    build_file_content = """

exports_files(["python_bin"])

filegroup(

    name = "files",

    srcs = glob(["bazel_install/**"], exclude = ["**/* *"]),

    visibility = ["//visibility:public"],

)

""",
```

```
   patch_cmds = [

        "mkdir $(pwd)/bazel_install",

        _configure_python_based_on_os,

        "make",

        "make install",

        "ln -s bazel_install/bin/python3 python_bin",

    ],

    sha256 =
"dfab5ec723c218082fe3d5d7ae17ecbdebffa9a1aea4d64aa3a2ecdd2e795864",

    strip_prefix = "Python-3.8.3",

    urls =
["https://www.python.org/ftp/python/3.8.3/Python-3.8.3.tar.xz"],

)
```

WORKSPACE

```
# Fetch official Python rules for Bazel

http_archive(

    name = "rules_python",

    sha256 = "b6d46438523a3ec0f3cead544190ee13223a52f6a6765a29eae7b7cc24cc83a0",

    url =
"https://github.com/bazelbuild/rules_python/releases/download/0.1.0/rules_python-0.1.0.tar.gz",

)

load("@rules_python//python:repositories.bzl", "py_repositories")

py_repositories()
```

Bazel Toolchain for Python - BUILD

```
load("@rules_python//python:defs.bzl", "py_runtime", "py_runtime_pair")

py_runtime(

    name = "python3_runtime",

    files = ["@python_interpreter//:files"],

    interpreter = "@python_interpreter//:python_bin",

    python_version = "PY3",

    visibility = ["//visibility:public"],

)
py_runtime_pair(

    name = "py_runtime_pair",

    py2_runtime = None,

    py3_runtime = ":python3_runtime",

)
```

```
toolchain(

    name = "py_3_toolchain",

    toolchain = ":py_runtime_pair",

    toolchain_type =
"@bazel_tools//tools/python:toolchain
_type",

)
```

Bazel Toolchain for Python - WORKSPACE

```
# The Python toolchain must be registered ALWAYS at the end of the file

register_toolchains("//:py_3_toolchain")
```

First Bazelized Test - requirements.txt

```
attrs==20.3.0 --hash=sha256:31b2eced602aa8423c2aea9c76a724617ed67cf9513173fd3a4f03e3a929c7e6
chardet==3.0.4 --hash=sha256:fc323ffcaeaed0e0a02bf4d117757b98aed530d9ed4531e3e15460124c106691
idna==2.8 --hash=sha256:ea8b7f6188e6fa117537c3df7da9fc686d485087abf6ac197f9c46432f7e4a3c
more-itertools==8.2.0 --hash=sha256:5dd8bcf33e5f9513ffa06d5ad33d78f31e1931ac9a18f33d37e77a180d393a7c
packaging==20.3 --hash=sha256:82f77b9bee21c1bafbf35a84905d604d5d1223801d639cf3ed140bd651c08752
pluggy==0.13.1 --hash=sha256:966c145cd83c96502c3c3868f50408687b38434af77734af1e9ca461a4081d2d
py==1.8.1 --hash=sha256:c20fdd83a5dbc0af9efd622bee9a5564e278f6380fffcacc43ba6f43db2813b0
pyparsing==2.0.2 --hash=sha256:17e43d6b17588ed5968735575b3983a952133ec4082596d214d7090b56d48a06
pytest==5.4.1 --hash=sha256:0e5b30f5cb04e887b91b1ee519fa3d89049595f428c1db76e73bd7f17b09b172
six==1.15.0 --hash=sha256:8b74bedcbbbaca38ff6d7491d76f2b06b3592611af620f8426e82dddb04a5ced
wcwidth==0.1.9 --hash=sha256:cafe2186b3c009a04067022ce1dcd79cb38d8d65ee4f4791b8888d6599d1bbe1
```

First Bazelized Test - BUILD

```
# Third party libraries

load("@rules_python//python:pip.bzl", "pip_install")

pip_install(

    name = "py_deps",

    python_interpreter_target= "@python_interpreter//:python_bin",

    requirements = "//:requirements.txt",

  )
```

First Bazelized Test - compiler_version_test.py

```
import os

import platform

import sys

import pytest

class TestPythonVersion:

    def test_version(self):

        assert(os.path.abspath(os.path.join(os.getcwd(),"..", "python_interpreter", "python_bin")) in sys.executable)

        assert(platform.python_version() == "3.8.3")

if __name__ == "__main__":

    import pytest

    raise SystemExit(pytest.main([__file__]))
```

First Bazelized Test - BUILD

```
load("@rules_python//python:defs.bzl", "py_test")
load("@py_deps//:requirements.bzl", "requirement")
py_test(
    name = "compiler_version_test",
    srcs = ["compiler_version_test.py"],
    deps = [
        requirement("attrs"),
        requirement("chardet"),
        requirement("idna"),
        requirement("more-itertools"),
        requirement("packaging"),
        requirement("pluggy"),
        requirement("py"),
        requirement("pytest"),
        requirement("wcwidth"),
    ],
)
```

First Bazelized Test - Execute the Test

```
$ bazel test //test:compiler_version_test

Starting local Bazel server and connecting to it...

INFO: Analyzed target //test:compiler_version_test (31 packages loaded, 8550 targets configured).

INFO: Found 1 test target...

Target //test:compiler_version_test up-to-date:

  bazel-bin/test/compiler_version_test

INFO: Elapsed time: 172.459s, Critical Path: 3.10s

INFO: 2 processes: 2 darwin-sandbox.

INFO: Build completed successfully, 2 total actions

//test:compiler_version_test                                          PASSED in 0.6s

Executed 1 out of 1 test: 1 test passes.

INFO: Build completed successfully, 2 total actions
```

Flask Application - flask_app.py

```
import platform
import subprocess
import sys
from flask import Flask
def cmd(args):
    process = subprocess.Popen(args, stdout=subprocess.PIPE)
    out, _ = process.communicate()
    return out.decode('ascii').strip()
@app.route('/')
def python_versions():
    bazel_python_path = f'Python executable used by Bazel is: {sys.executable} <br/><br/>'
    bazel_python_version = f'Python version used by Bazel is: {platform.python_version()} <br/><br/>'
    host_python_path = f'Python executable on the HOST machine is: {cmd(["which", "python3"])} <br/><br/>'
    host_python_version = f'Python version on the HOST machine is: {cmd(["python3", "-c", "import platform; print(platform.python_version())"])}'
    python_string = (
        bazel_python_path
        + bazel_python_version
        + host_python_path
        + host_python_version
    )
    return python_string
if __name__ == '__main__':
    app.run()

app = Flask(__name__)
```

Flask Application - BUILD & requirements.txt

```
load("@rules_python//python:defs.bzl",
"py_binary")

load("@py_deps//:requirements.bzl",
"requirement")

py_binary(

    name = "flask_app",

    srcs = ["flask_app.py"],

    python_version = "PY3",

    deps = [

        requirement("flask"),

        requirement("jinja2"),

        requirement("werkzeug"),

        requirement("itsdangerous"),

        requirement("click"),

    ],
)
```

```
click==5.1 --hash=sha256:0c22a2cd5a1d741e993834df99133de07eff6cc1bf06f137da2c5f3bab9073a6

flask==1.1.2 --hash=sha256:8a4fdd8936eba2512e9c85df320a37e694c93945b33ef33c89946a340a238557

itsdangerous==0.24 --hash=sha256:cbb3fcf8d3e33df861709ecaf89d9e6629cff0a217bc2848f1b41cd30d360519

Jinja2==2.10.0 --hash=sha256:74c935a1b8bb9a3947c50a54766a969d4846290e1e788ea44c1392163723c3bd

MarkupSafe==0.23 --hash=sha256:a4ec1aff59b95a14b45eb2e23761a0179e98319da5a7eb76b56ea8cdc7b871c3

Werkzeug==0.15.5 --hash=sha256:87ae4e5b5366da2347eb3116c0e6c681a0e939a33b2805e2c0cbd282664932c4
```

Flask Application - Run the Application & localhost

```
$ bazel run //src:flask_app
INFO: Analyzed target //src:flask_app (10 packages loaded, 184 targets
configured).
INFO: Found 1 target...

Target //src:flask_app up-to-date:   bazel-bin/src/flask_app

INFO: Elapsed time: 7.430s, Critical Path: 1.12s

INFO: 4 processes: 4 internal.

INFO: Build completed successfully, 4 total actions

 * Serving Flask app "flask_app" (lazy loading)

 * Environment: production

 * Debug mode: off

 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
Python executable used by Bazel is:
/private/var/.../python_interpreter/python_bin

Python version used by Bazel is: 3.8.3

Python executable in the HOST machine is:
/Users/gaspare/.pyenv/versions/3.8.5/bin/pytho
n3

Python version in the HOST machine is: 3.8.5
```

Flask Application - Test Reproducibility

```
$ md5sum $(bazel info bazel-bin)/src/flask_app

2075a7ec4e8eb7ced16f0d9b3d8c5619

$ bazel clean --expunge_async

INFO: Starting clean.

$ bazel build //...

Starting local Bazel server and connecting to it...

INFO: Analyzed 4 targets (38 packages loaded, 8729 targets configured).

INFO: Found 4 targets...

INFO: Elapsed time: 183.923s, Critical Path: 1.65s

INFO: 7 processes: 7 internal.

INFO: Build completed successfully, 7 total actions

$ md5sum $(bazel info bazel-bin)/src/flask_app

2075a7ec4e8eb7ced16f0d9b3d8c5619
```

… Is It Really Reproducible?

Not really...

WORKSPACE

```
load("@bazel_tools//tools/build_defs/repo:http.bzl", "http_archive")

# Fetch Python and build it from scratch

http_archive(

    name = "python_interpreter",

    build_file_content = """

exports_files(["python_bin"])

filegroup(

    name = "files",

    srcs = glob(["bazel_install/**"], exclude = ["**/* *"]),

    visibility = ["//visibility:public"],

)

"""

,
```

```
   patch_cmds = [

        "mkdir $(pwd)/bazel_install",

        _configure_python_based_on_os,

        "make",

        "make install",

        "ln -s bazel_install/bin/python3 python_bin",

    ],

    sha256 =
"dfab5ec723c218082fe3d5d7ae17ecbdebffa9a1aea4d64aa3a2ecdd2e795864"
,

    strip_prefix = "Python-3.8.3",

    urls =
["https://www.python.org/ftp/python/3.8.3/Python-3.8.3.tar.xz"],

)
```

Now It Is Fully Really Reproducible

Use Docker

Use a precompiled binary

Use Nix Bazel rules

TAKEAWAYS:

What do reproducible means?

- Don't take for granted that your build is reproducible.
- Hermeticity enables cherry-picking.
- Inputs to the build must be versioned with source code.
- Internal randomness can be sneaky but must be removed.
- You have a working Python environment that is hermetic thanks to Bazel.
- You have seen how to compile a Flask Application in a reproducible way.

Link to Github: github.com/gasparev/reproducible-python-bazel

Thank you!

www.gasparevitta.com
@gasparevitta