arm

Integrating Bazel and `pre-commit`

Matt Clarkson
2025-05-22

Git Hooks

- Git hooks are a way to automate project checks
- Linting and formatting are good examples
- Git runs various stages of hooks from `.git/hooks` or another configured location
- Each stage is an executed Shell script
- A common stage is `pre-commit` which runs before a Git commit is completed
- Simple to execute various tools as part of a stage from Shell script
- Often, we need:
  - Multiple tools
  - Different languages
  - Dependencies that need to be installed
- Becomes complex to manage

`pre-commit` framework

- Available at https://pre-commit.com
- A Git hook framework written in Python
- Configured via a `.pre-commit-config.yaml`
- Can use published, pre-configured hooks
- `pre-commit` will hermetically setup hooks
  - Language runtime downloaded
  - Hook installed into isolated environment
  - Uses a shared user cache
- Hooks are only executed on matched files
  - Can use a regular expression on filename
  - Use a *type* such as `python`, `protobuf`, etc
- Possible to use *local* hooks
  - No dependencies are installed
- Has a lovely user experience
- Highly recommended framework!

```
repos:
- repo: https://github.com/pre-commit/pre-commit-hooks
  rev: v2.3.0
  hooks:
  - id: check-yaml
  - id: end-of-file-fixer
  - id: trailing-whitespace
- repo: https://github.com/psf/black
  rev: 22.10.0
  hooks:
  - id: black
- repo: local
  hooks:
  - id: check-requirements
    name: check requirements files
    language: system
    entry: python -m scripts.check_requirements --compare
    files: ^requirements.*\.txt$
```

Good, bad and the ugly

Good

- Hermetic tools
- Hooks run on matched files
- Excellent user experience
- Lots of published hooks

Bad

- Cache is local
- `pre-commit` needs to be installed locally
- Cannot share configurations

Ugly

- Language runtime downloaded for each hook
- Hooks are setup serially
- Long hook setup time

Bazel

- Use Bazel to hermetically install Python and `pre-commit`
- Each hook is `local` which calls into a Bazel run target
- Generate `.pre-commit-config.yaml` with Bazel
- Teach `pre-commit` Shell script to re-use Bazel downloads

Bazel

Use Bazel to hermetically install Python and `pre-commit`

- Need to expose `pre-commit` entrypoint
- Do not control `rules_python` Python version
- Use `uv` to generate a universal lockfile
- Generate `pip` hubs for each Python version
- Select hub with `select` statement
- `bazel run @pre-commit` is hermetic

Bazel

Each hook is `local` which calls into a Bazel run target

- Use the `pre_commit_hook` rule
- Bazel run target provided to `src` attribute
- Run target will use Bazel cache
- Remote cache to share dependencies
- Hook is shared with normal Bazel visibility rules

```
load(
    "@pre-commit//pre-commit/hook:defs.bzl",
    "pre_commit_hook",
)

pre_commit_hook(
    name = "format",
    src = "@buildifier_prebuilt//:buildifier",
    description = "Use `.buildifier.json` for config.",
    stages = [
        "@pre-commit//pre-commit/stage:pre-commit",
    ],
    summary = "Format Bazel files",
    types_or = [
        "@pre-commit//pre-commit/tag:bazel",
    ],
    visibility = ["//visibility:public"],
)
```

Bazel

Generate `.pre-commit-config.yaml` with Bazel

- Use the `pre_commit_config` rule
- Provide hooks to `srcs` attribute
- Generate configuration to `out` attribute
- `bazel run :config`

```
load(
    "@pre-commit//pre-commit/config:defs.bzl",
    "pre_commit_config",
)

pre_commit_config(
    name = "config",
    srcs = [
        "@pre-commit-hooks//buildifier",
    ],
    out = ".pre-commit-config.yaml",
)
```

Bazel

Teach `pre-commit` Shell script to re-use Bazel downloads

- Use the `pre_commit_install` rule
- Installs the generated configuration
- Runs `pre-commit install`
- Monkey patches extra code into the Shell script to call into hermetic `pre-commit` from Bazel

```
load(
    "@pre-commit//pre-commit/install:defs.bzl",
    "pre_commit_install",
)

pre_commit_install(
    name = "install",
    src = ":config",
)
```

`pre_commit` macro

- Does everything that is needed
- Put it in `hooks/BUILD.bazel`
  - `bazel run hooks:config`
  - `bazel run hooks:install`
  - `bazel run hooks`
- Add `bazel run hooks` to CI to enforce checks

```
load("@pre-commit//pre-commit:defs.bzl", "pre_commit")

pre_commit(
    name = "hooks",
    srcs = [
        "@pre-commit-hooks",
    ],
)
```

Try it out!

- Stable but not currently in Bazel Central Registry (BCR)
- Add the release registry endpoints:

```
https://gitlab.arm.com/bazel/pre-commit/-/releases/v${VERSION}/downloads
https://gitlab.arm.com/bazel/pre-commit-hooks/-/releases/v${VERSION}/downloads
```

arm

Merci
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Thank You
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה
ధన్యవాదములు
Köszönöm