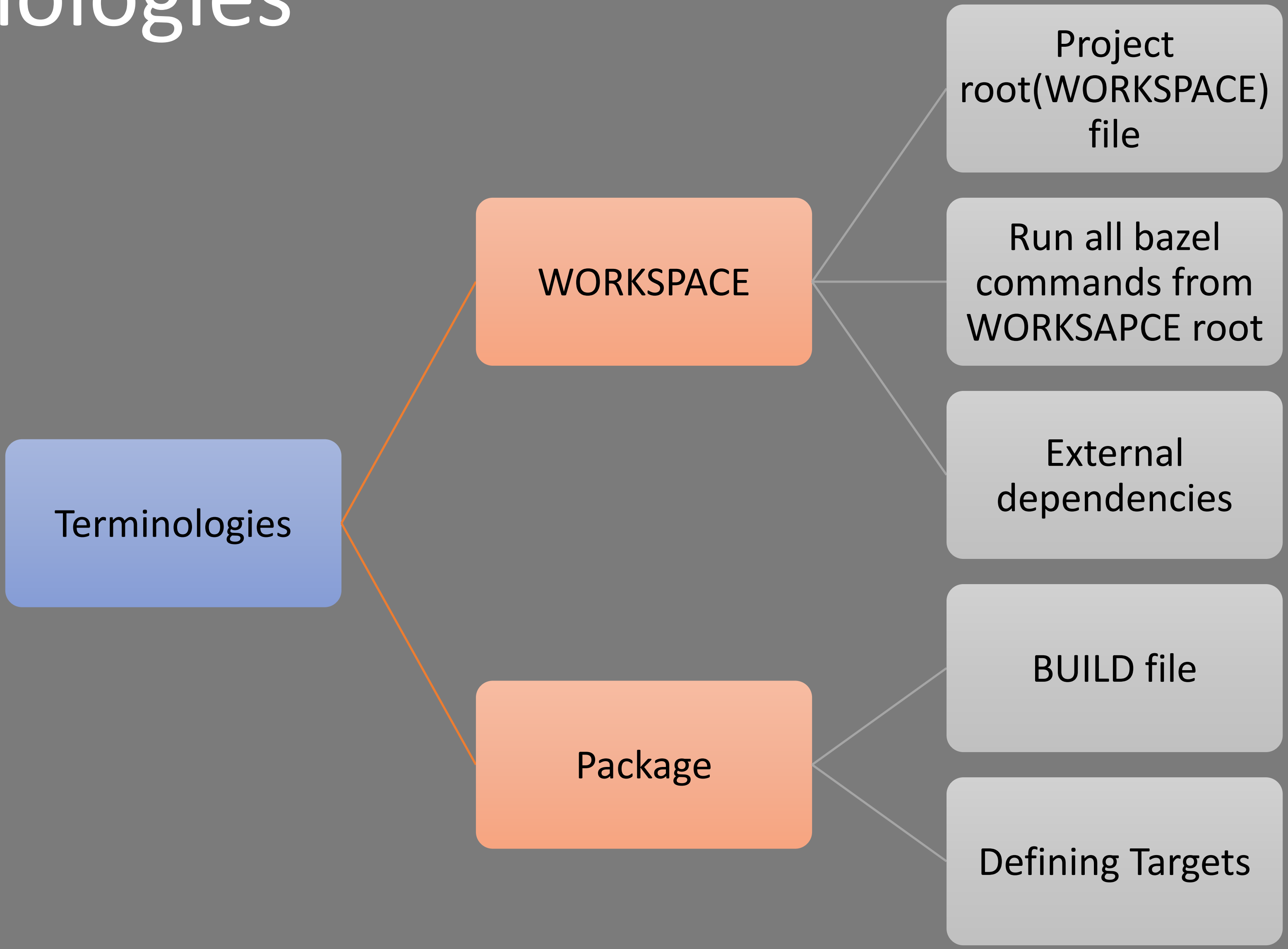
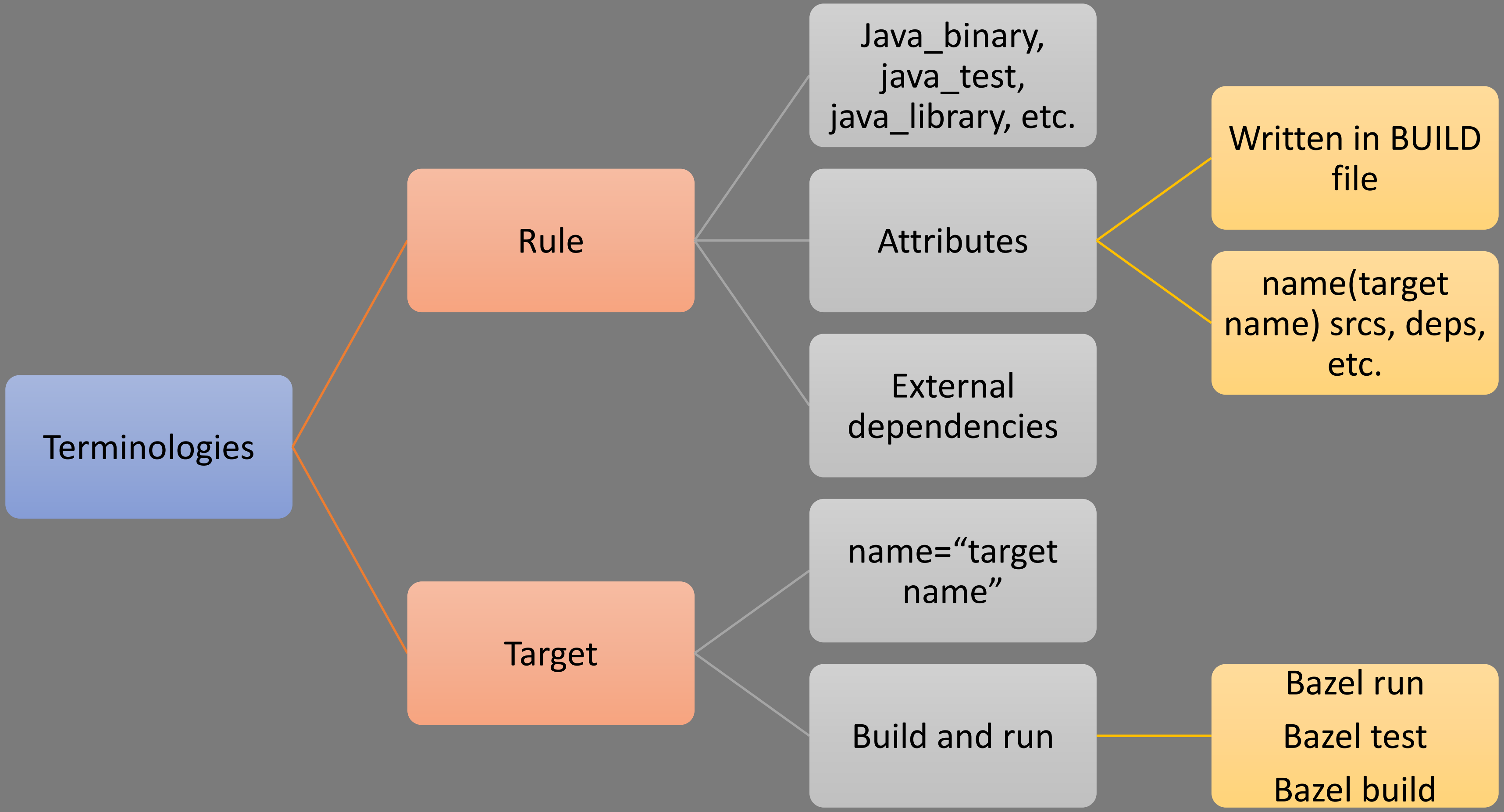
Bazel foundation course summary

- Bazel terminologies
- Bazel features
- Building the targets
- Different types of targets
- Third party/external dependencies
- Rules, Macros, Caching, platforms, selection
- Toolchains and queries

Terminologies



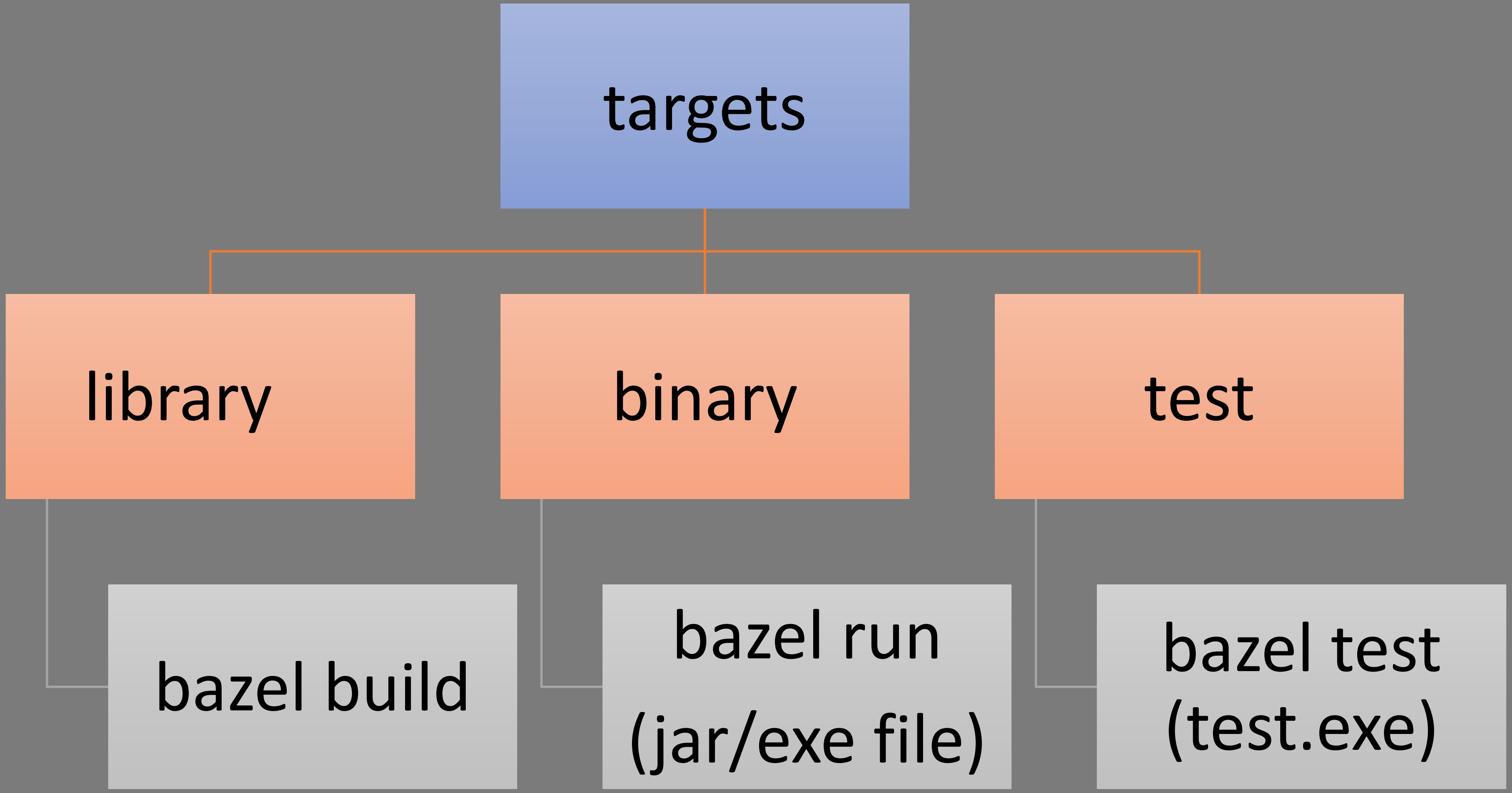
Terminologies



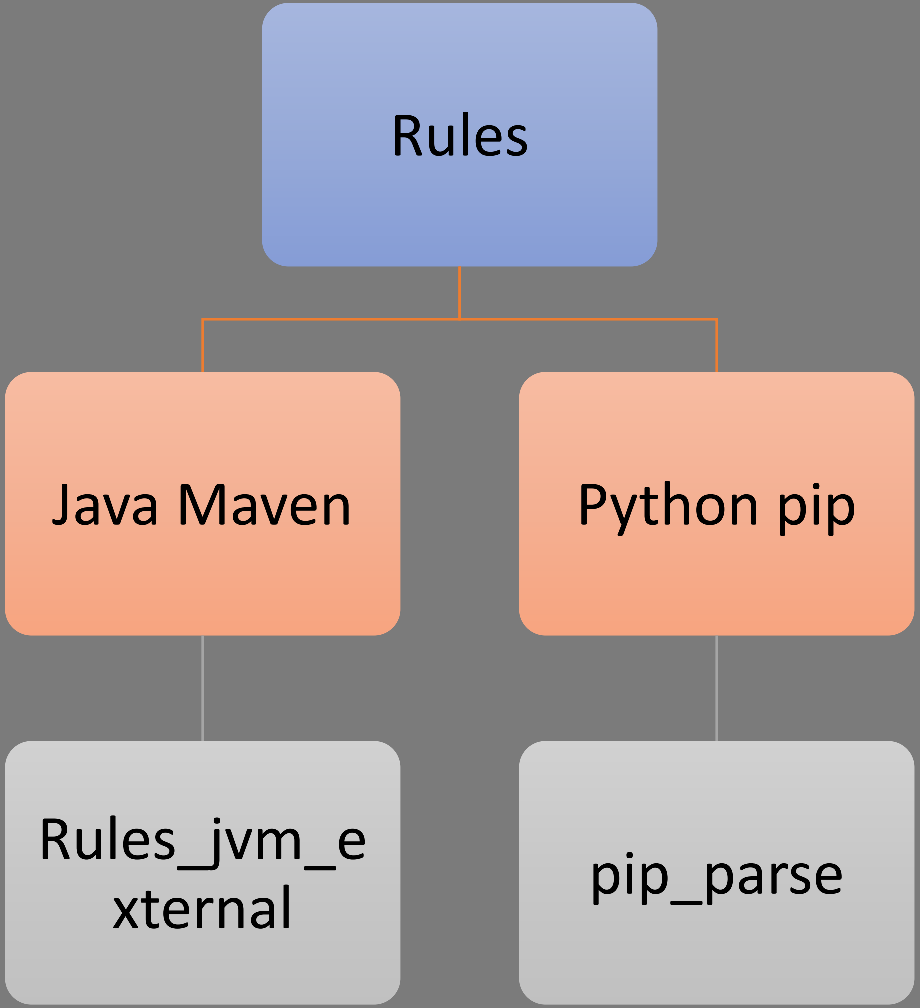
Features

- Sandbox environment
- Incremental builds through caching
- Hermetic and reproducible builds

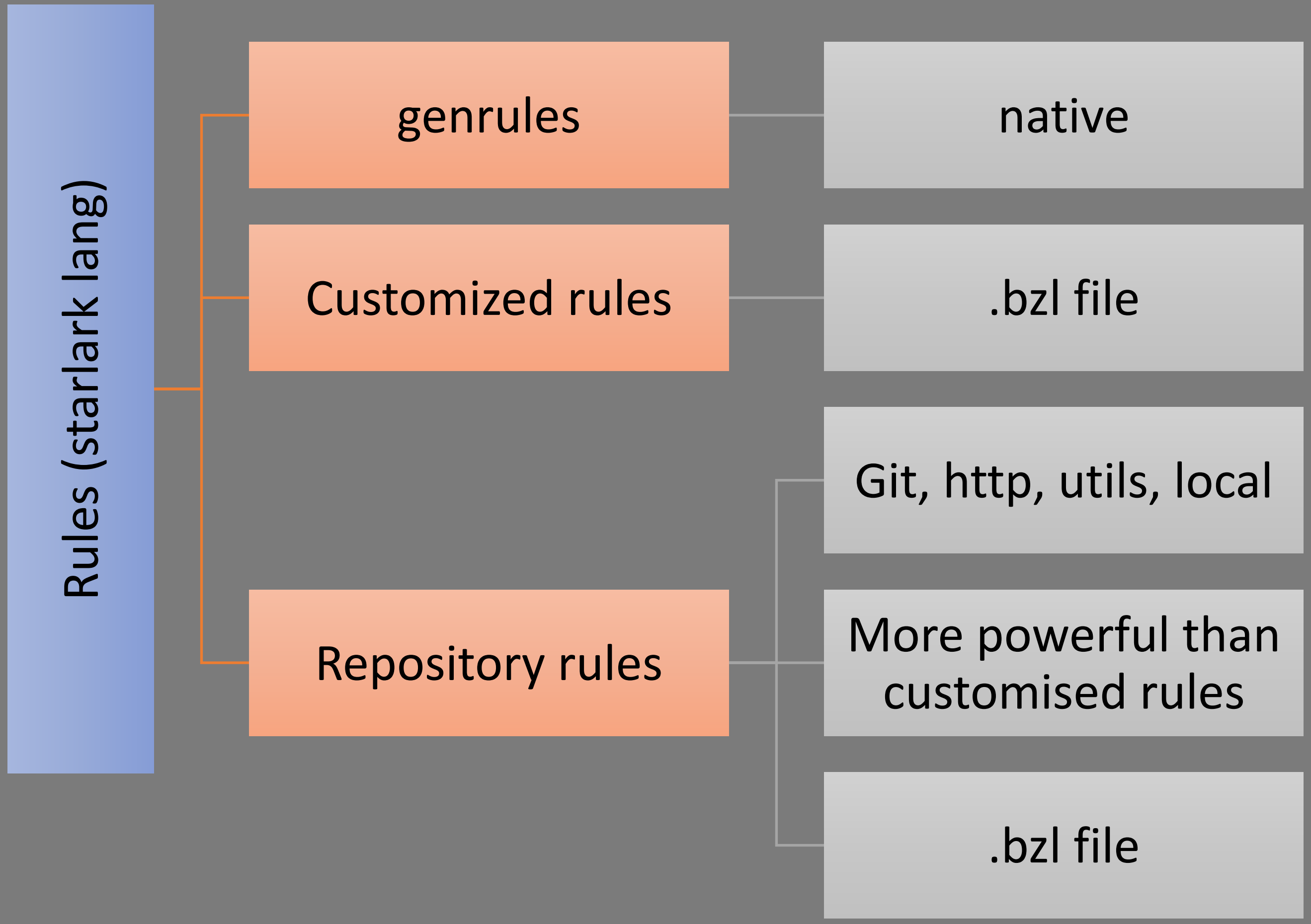
Targets


targets
library
binary
test
bazel build
bazel run (jar/exe file)
bazel test (test.exe)

Third party dependencies



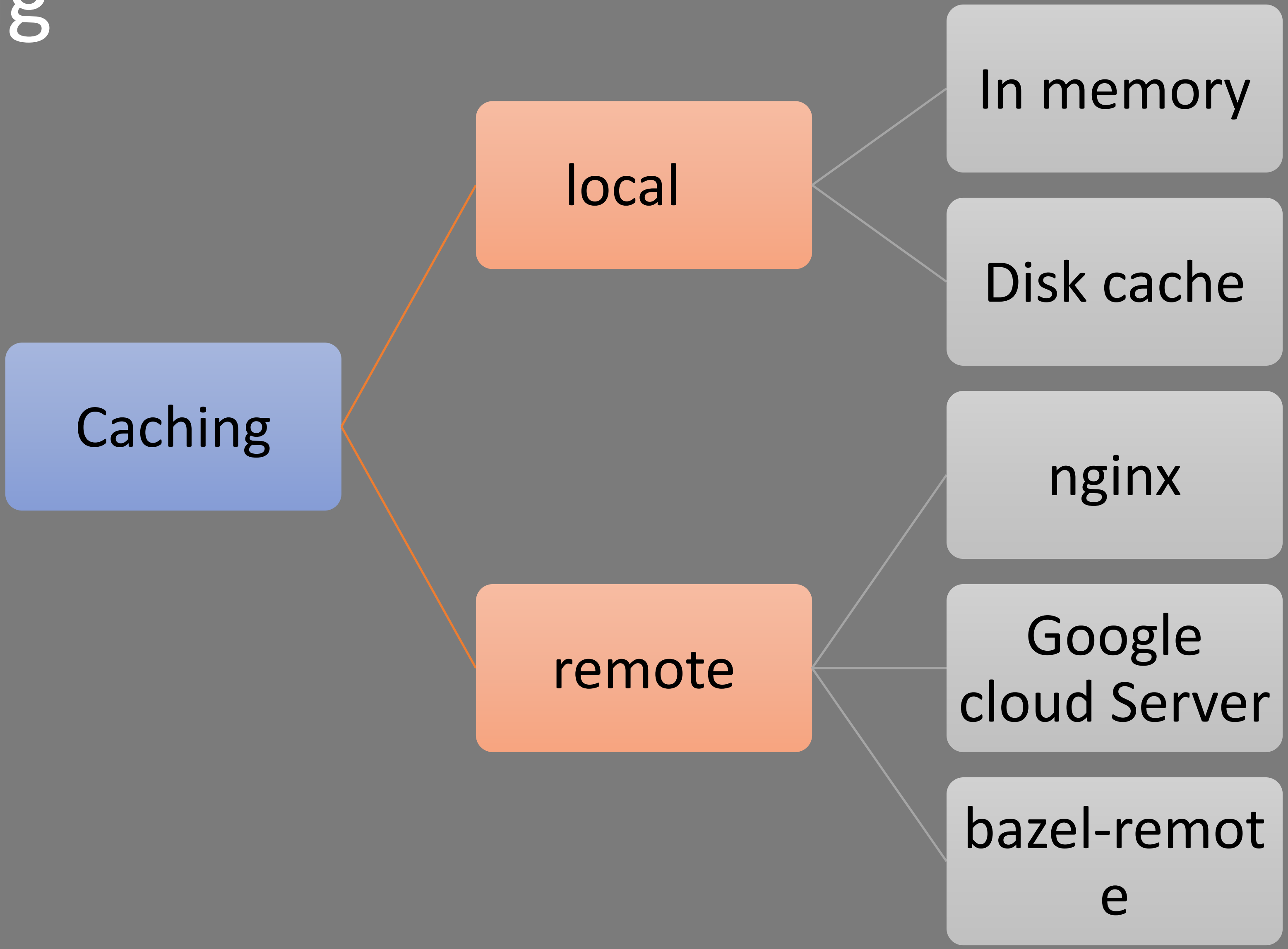
Rules


Rules (starlark lang)
genrules
native
Customized rules
.bzl file
Repository rules
Git, http, utils, local
More powerful than customised rules
.bzl file

Macros

- .bzl file
- Called from BUILD file
- Repeating use of rules
- Works in loading phase

Caching


Caching
local
remote
In memory
Disk cache
nginx
Google cloud Server
bazel-remote

Platforms

Host

- Bazel itself runs

Execution

- build tools execute build actions to produce intermediate and final outputs.

Target

- a final output resides and executes

Platforms

Single Platform Builds

- Host, execution and target are same

Cross platform builds

- Host and execution platforms are same, target is different

Multi platform builds

- Host, execution and target platforms are different

Platforms rule example

The following creates a platform named linux_x86, and says that it describes any environment that runs a Linux operating system on an x86_64 architecture with a glibc version of 2.25.

```
platform(
  name = "linux_x86",
  constraint_values = [
    "@platforms//os:linux",
    "@platforms//cpu:x86_64",
    ":glibc_2_25",
  ],
)
```

The target will not be built for any platform that doesn't satisfy all of the constraints. The following example restricts win_driver_lib.cc to 64-bit Windows.

```
cc_library(
  name = "win_driver_lib",
  srcs = ["win_driver_lib.cc"],
  target_compatible_with = [
    "@platforms//cpu:x86_64",
    "@platforms//os:windows",
  ],
)
```

Platforms + selection example

Use select() in combination with @platforms//:incompatible to express more complicated restrictions. For example, use it to implement basic OR logic. The following marks a library compatible with macOS and Linux, but no other platforms.

```
cc_library(
   name = "unixish_lib",
   srcs = ["unixish_lib.cc"],
   target_compatible_with = select({
      "@platforms//os:osx": [],
      "@platforms//os:linux": [],
      "//conditions:default":
["@platforms//:incompatible"],
   }),
)
```

Detecting incompatible targets using bazel cquery

```
$ cat example.cquery

def format(target):
  if "IncompatiblePlatformProvider" not in providers(target):
    return target.label
  return ""


$ bazel cquery //... --output=starlark
--starlark:file=example.cquery
```

Queries

Query

- Loading phase

cquery

- Analysis phase

aquery

- After analysis phase

Toolchains

Decouples rule logic from platform based selection of tools