

Zig packages, hermetically

Zig package manager integration in rules_zig

ANDREAS HERRMANN • BAZELCON 2026

Zig

A modern C



Compiles Zig, C and C++ for many targets

Zig logo CC BY-SA 4.0, Ziggy the Ziguana CC BY 4.0: [ziglang/logo](https://ziglang.org/logo)

Zig

A modern C



Compiles Zig, C and C++ for many targets



hermetic_cc_toolchain

```
zig cc
```

Zig as a C/C++ cross compiler

Zig logo CC BY-SA 4.0, Ziggy the Ziguana CC BY 4.0: [ziglang/logo](https://ziglang.org/logo)

rules_zig

Zig code as first-class Bazel targets

```
zig_library(  
    name = "lib",  
    main = "lib.zig",  
)  
  
zig_binary(  
    name = "app",  
    main = "main.zig",  
    deps = [":lib", ":some_cc_lib"],  
)
```

- Cross-compile via Bazel platforms
- Mix with C/C++ libraries
- Multiple Zig versions via transitions

Zig dependencies in rules_zig today

A hand-written BUILD file per dependency.

```
http_archive(  
    name = "translate-c",  
    build_file = "//third_party/...:translate-c.bazel",  
    sha256 = "948dbb0e...",  
    url = "https://.../0da7a16c...tar.gz",  
)
```

```
write_file(  
    name = "build_options_zig",  
    out = "build_options.zig",  
    content = ["""\n  
pub const debug_allocations: bool = false;  
"""],  
)  
  
zig_library(  
    name = "c_builtins",  
    main = "lib/c_builtins.zig",  
)
```

How Zig compiles, builds and fetches

main.zig

```
const clap = @import("clap");

pub fn main() !void {
    const res = try clap.parse(...);
    if (res.args.help > 0) ...
}
```

- One compilation unit

How Zig compiles, builds and fetches

main.zig

```
const clap = @import("clap");

pub fn main() !void {
    const res = try clap.parse(...);
    if (res.args.help > 0) ...
}
```

- One compilation unit

build.zig

```
const dep = b.dependency("clap", .{
    .target = target,
    .optimize = optimize,
});
exe.root_module.addImport(
    "clap", dep.module("clap"));
```

- In Zig, via `std.Build`

How Zig compiles, builds and fetches

main.zig

```
const clap = @import("clap");

pub fn main() !void {
    const res = try clap.parse(...);
    if (res.args.help > 0) ...
}
```

- One compilation unit

build.zig

```
const dep = b.dependency("clap", .{
    .target = target,
    .optimize = optimize,
});
exe.root_module.addImport(
    "clap", dep.module("clap"));
```

- In Zig, via `std.Build`

build.zig.zon

```
.fingerprint = 0xc96e70cfecd7faf1,
.dependencies = .{ .clap = .{
    .url = "https://...",
    .hash = "clap-0.12.0-oBaj...",
} },
```

- Fingerprint: the package
- Hash: one version

Zig's layers



`build.zig.zon`

composes packages



Package manager

fetches by hash



`build.zig`

a package



Build system

runs the build script



`main.zig`

modules



Compiler

one ZCU per artifact

The goal

app/build.zig.zon

```
.dependencies = .{  
    .clap = .{  
        .url = "https://.../05faf39...tar.gz",  
        .hash = "clap-0.12.0-oBajB4Xp...",  
    },  
},
```

MODULE.bazel

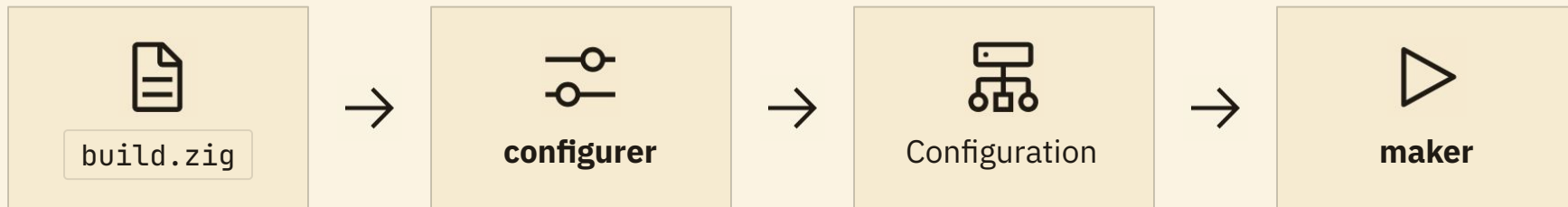
```
zig_packages = use_extension(  
    "@rules_zig//zig:packages.bzl", "zig_packages")  
zig_packages.from_file(  
    build_zig_zon = "//app:build.zig.zon")  
use_repo(zig_packages, "zig_deps")
```

app/BUILD.bazel

```
load("@zig_deps//:defs.bzl", "zig_deps")  
  
zig_binary(  
    name = "app",  
    main = "src/main.zig",  
    deps = zig_deps(),  
)
```

```
$ bazel build //app
```

Zig 0.17: configure ≠ make



Configure is its own phase: `zig build --print-configuration`

How it works

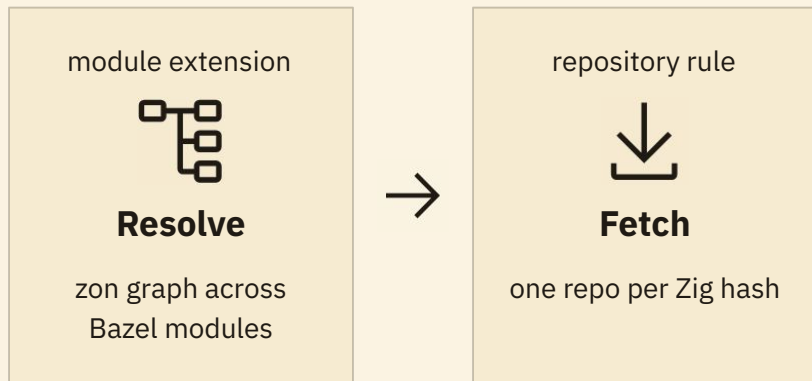
module extension



Resolve

zon graph across
Bazel modules

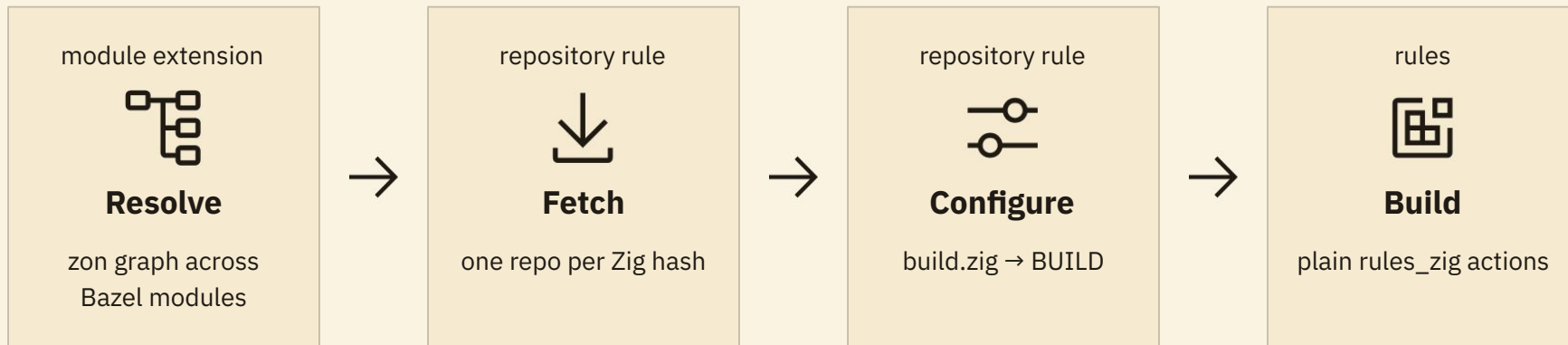
How it works



How it works



How it works



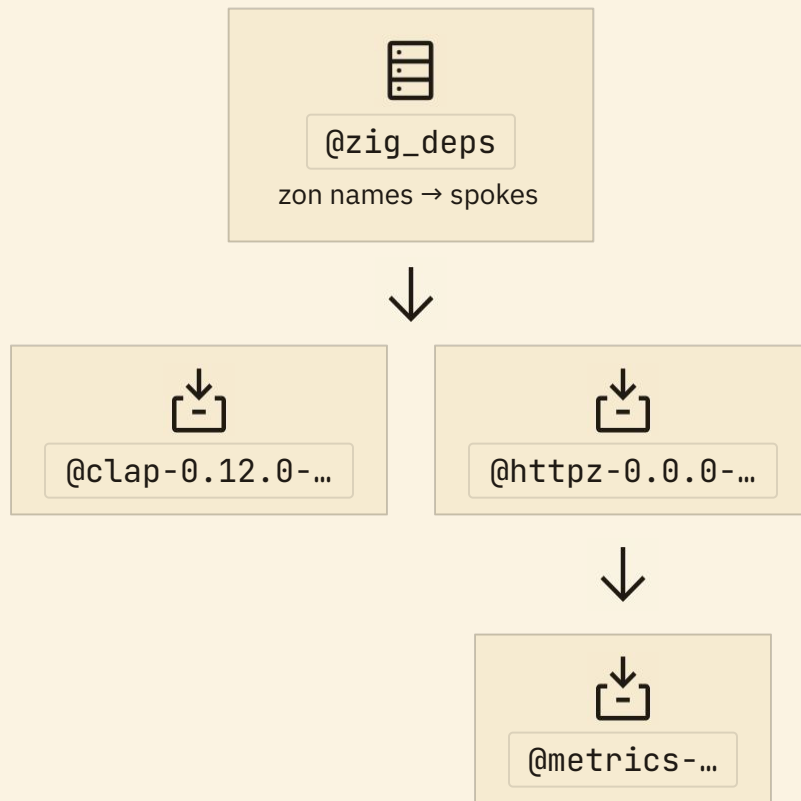
Hub and spokes

build.zig.zon

```
.clap = .{ .hash = "clap-0.12.0-..." },  
.httpz = .{ .hash = "httpz-0.0.0-..." },
```

BUILD.bazel

```
zig_binary(  
  name = "app",  
  main = "main.zig",  
  deps = [  
    zig_dep("clap"),  
    zig_dep("httpz"),  
  ],  
)
```



Cross-compilation and multiple configs

```
zig_packages.config(  
    name = "linux",  
    target = "x86_64-linux-gnu",  
)  
zig_packages.config(  
    name = "macos",  
    target = "aarch64-macos",  
    select_on = [  
        "@platforms//cpu:aarch64",  
        "@platforms//os:osx",  
    ],  
)  
zig_packages.configure(  
    configs = ["linux", "macos"],  
    fallback = "linux",  
)
```

- One configure run per config
- Differences → `select()`
- e.g. Bazel platform → `-Dtarget`

Limits, and Zig's build server

Limits

- Not every `build.zig` maps to Bazel
- e.g. host probes in configure
- e.g. writes to the source tree

Zig build server

- No exposed module names yet
- Options fixed per server
- Steps run by the maker

Next in Zig: build server clients via `zig maker` ([#36497](#))

Status and roadmap



Merged

Resolve and fetch

Configure → `zig_library`

`@zig_deps` `hub`



Next

C integration

Cross-compilation

Multiple configs



Roadmap

Lock file

Better configuration

Codegen steps

The recipe



Resolve in bzlmod

across Bazel modules



Fetch & configure

in a repo rule, emit `select()`



Build

with regular rules



Thank you

[hermeticbuild/rules_zig](#) · [#231](#) · [#232](#)

Andreas Herrmann · [yume.ag](#)