



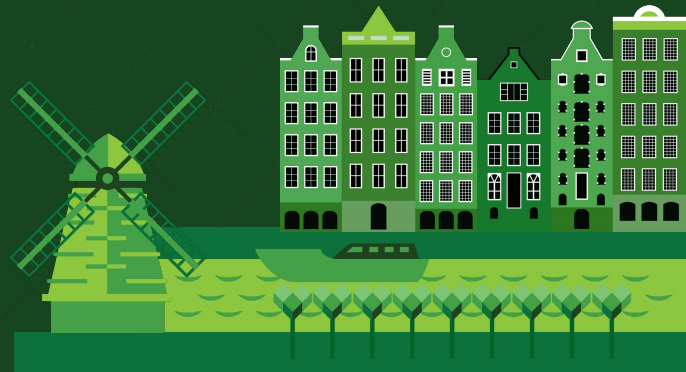
BazelCon 2026

More expressive platforms with `platforms_contrib`



Fabian Meumertzheim
Software Engineer, BuildBuddy
fabian@buildbuddy.io

#BazelCon



Beyond OS and CPU

```
platform(  
    name = "linux_x86_64",  
    constraint_values = [  
        "@platforms//os:linux",  
        "@platforms//cpu:x86_64",  
    ],  
)
```

kernel version?

```
toolchain(  
    name = "llvm_22",  
    exec_compatible_with = [  
        "@platforms//os:linux",  
        "@platforms//cpu:x86_64",  
    ],  
    ...  
)
```

glibc requirement?

```
cc_binary(  
    name = "blake3",  
    srcs = select({  
        "@platforms//cpu:x86_64": ...,  
        "//conditions:default": ...,  
    }),  
    ...  
)
```

AVX2?

Challenge

Interoperability requires shared constraints



BazelCon 2026

@platforms

- OS
 - CPU
-
- `@platforms//host`
 - (necessarily) very conservative about additions



@platforms_contrib

- OS → @platforms
- CPU → @platforms
- CPU microarchitecture & features
- libc flavor & version
- @platforms_contrib//host:{generic, native}
 - enables fully hermetic "-march=native"
- accepts complexity so you don't have to
 - refines_constraint_value (new feature in Bazel 9 & 10)
 - range constraints



CPU features

```
platform(  
    name = "linux_x86_64_v3",  
    constraint_values = [  
        "@platforms//os:linux",  
        "@platforms//cpu:x86_64",  
    ] + x86_64_level_constraints("v3"),  
)  
  
cc_library(  
    name = "matmul",  
    srcs = select({  
        "@platforms_contrib//cpu/x86_64/feature:avx2": ["matmul_avx2.c"], # selected  
        "//conditions:default": ["matmul_generic.c"],  
    }),  
)
```



libc versions

```
toolchain(  
  name = "legacy_toolchain",  
  exec_compatible_with = [  
    "@platforms_contrib//os/linux/libc/glibc:at_least_2.40_available",  
  ],  
  target_compatible_with = [  
    "@platforms_contrib//os/linux/libc/glibc:at_least_2.17_available",  
  ]  
)  
  
platform(  
  name = "ubuntu_20.04_with_musl", # matches as target, but not as exec  
  constraint_values = [  
    ...,  
  ] + glibc_version_constraints("2.31") + musl_version_constraints("1.2"),  
)
```



So that's a constraint, right? ...Right?

Rule of thumb

Does it affect what runs on a given machine?

- “Is glibc $\geq$ 2.35 available on my machine?”
 constraint
- “Does my code compile against musl?”
 obtain from the toolchain (`@rules_cc//cc/libc:musl`)
- “Should my toolchain pick glibc or musl when both are available?”
 free choice, can be an unshared Starlark flag



Roadmap

- Integrate with `@llvm`
- ARM constraints – help wanted!
- Version 1.0.0
- GPU features, PIE, OS kernel versions, ...

Give it a try and suggest improvements!

`bazel-contrib/platforms_contrib`

