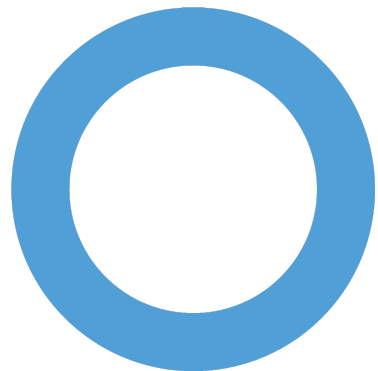


Universally Buildable Extensions





Omnigres

Extension Adoption

Application developers → Production operators

Developer Operating System

according to StackOverflow

45% use **Windows**

40% use **Linux**

30% use **macOS**

15% use **WSL**

(over 100% means overlapping)

Developer Operating System

according to JetBrains

61% use **Windows**

46% use **macOS**

45% use **Linux**

1% use **other**

(over 100% means overlapping)

Operating Out of a Container

- Mapping resources
 - (cores, memory, disk, networking, env vars, etc.)
- Linux-specific
- Example: No GPU support through Docker Desktop / Colima
- Container as a workaround for compile-time paths?

postgres.pm

(PoC)

Simple example

```
:- package(vector(Version), imports([git_tagged_revision_package(Version)])).  
  
git_repo("https://github.com/pgvector/pgvector").  
  
:- end_package.
```

- Infers versions (git tags)
- Infers build system (Makefile + C files)
 - Infers `make` and C compiler
- Infers metadata from META.json

Slightly more involved one

```
:- package(pg_curl(Version), imports([git_explicit_revision_package(Version)])).  
  
:- inherit(requires/1).  
  
git_repo("https://github.com/RekGRpth/pg_curl").  
  
git_revisions([  
    '502217c': '2.1.1',  
    % ...older versions omitted for now...  
]).  
  
requires(when(D := external_dependency(libcurl), version::match(D, '^7'))).  
  
:- end_package.
```

- Maps versions to commits
- Specifies a requirement
 - Solved using available "satisfiers" (pkgconfig, apt, homebrew, etc.)

High-level requirements
vs.
highly specific recipes

Build against minor versions?

Build against minor versions?

Perhaps

(and test, too)

16.0 → 16.1 → 16.2

New fields inserted in the middle

```
typedef struct BTScanOpaqueData
{
-     /* these fields are set by _bt_preprocess_keys(): */
+     /* all fields (except arraysStarted) are set by _bt_preprocess_keys(): */
    bool        qual_ok;           /* false if qual can never be satisfied */
+     bool        arraysStarted; /* Started array keys, but have yet to "reach
+                                     * past the end" of all arrays? */
    int          numberOfKeys; /* number of preprocessed scan keys */
```

New APIs

```
#define MaxArraySize ((Size) (MaxAllocSize / sizeof(Datum)))
...
+extern Relation try_index_open(Oid relationId, LOCKMODE lockmode);
...
+extern bool contain_mutable_functions_after_planning(Expr *expr);
```

16.0 → 16.1 → 16.2

Changes in inline behaviour

```
+      /*  
+      * Note: overflow is also possible when a = 0 and b < 0 (specifically,  
+      * when b = PG_INT64_MIN).  
+      */  
+      if ((a < 0 && b > 0 && a < PG_INT64_MIN + b) ||  
-          (a > 0 && b < 0 && a > PG_INT64_MAX + b))  
+          (a ≥ 0 && b < 0 && a > PG_INT64_MAX + b))
```

Unless you're going deep, you are probably ok
(but who knows?!)

Distribution-independent binary dependencies

- Static linking
- Bundling with `RPATH ($ORIGIN / @loader_path)`

Recap

- macOS/Windows DX
- Containers require resource mapping
- Build inferencing (postgres.pm)
- Static and RPATH dependencies
- PG minor version differences may be tricky