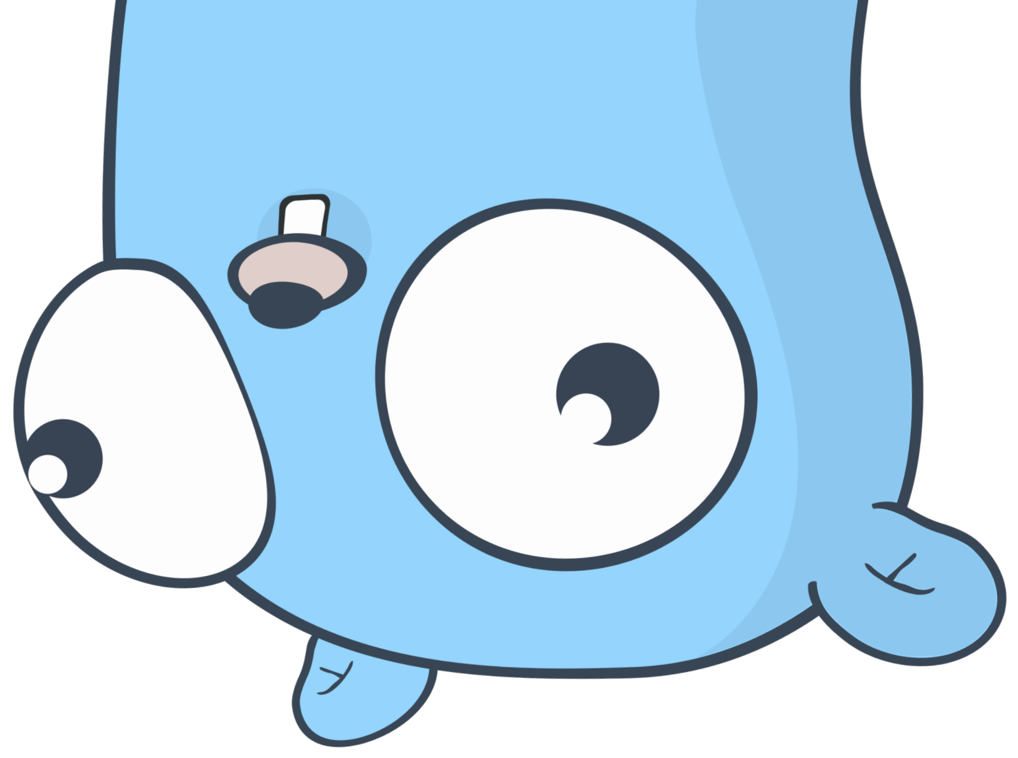
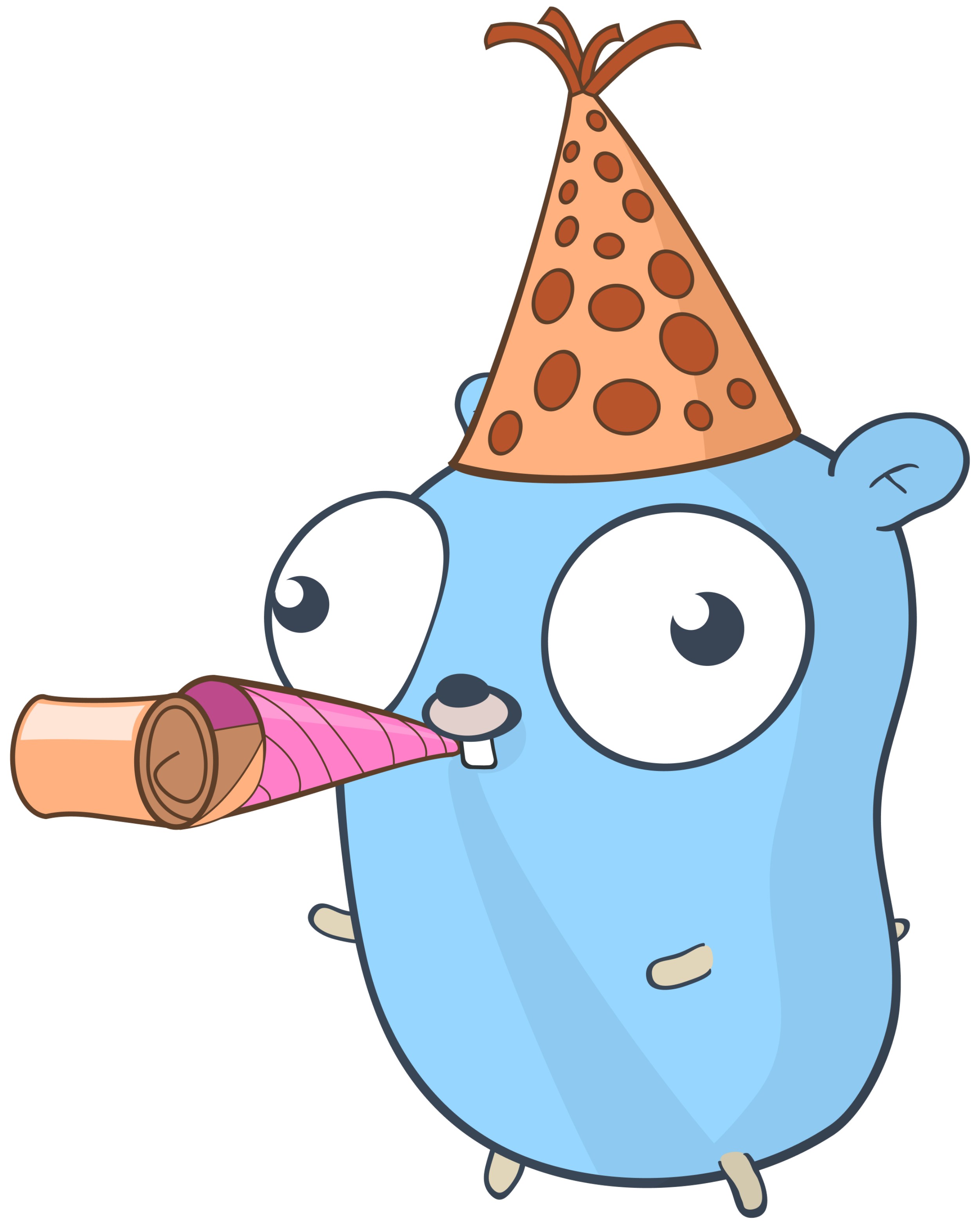


Safe, Fast, and Easy

Building a plugin system with WebAssembly

Kyle Conroy | conroy.org | @kyle_conroy



sqlc

sqlc

github.com/kyleconroy/sqlc | sqlc.dev

Compile SQL to type-safe code

How it works

How it works

1. You write SQL queries

How it works

1. You write SQL queries
2. You run `sqlc generate`, which outputs Go code with type-safe interfaces to those queries

How it works

1. You write SQL queries
2. You run `sqlc generate`, which outputs Go code with type-safe interfaces to those queries
3. You write application code that calls these methods

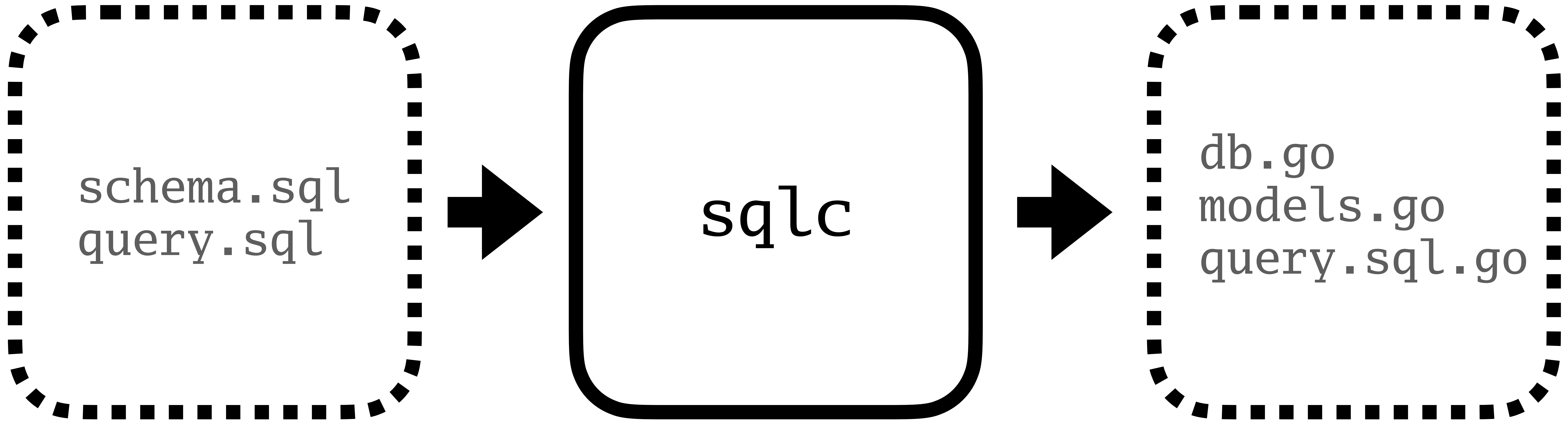
Quick example

```sql
CREATE TABLE authors (
  id   BIGSERIAL PRIMARY KEY,
  name text      NOT NULL,
  bio  text
);
```

Quick example

```sql
-- name: GetAuthor :one
SELECT * FROM authors
WHERE id = $1 LIMIT 1;
```

Quick example



Quick example

```go
ctx := context.Background()
queries := example.New(db) // *db.Sql

author, _ := queries.GetAuthor(ctx, 42)
fmt.Println(author.Name)
```

Column expansion

```
-- name: GetAuthor :one
SELECT * FROM authors
WHERE id = $1 LIMIT 1;
```

Column expansion

```
const getAuthQuery = `
SELECT id, name, bio FROM authors
WHERE id = $1 LIMIT 1;
`
```

Type inference

```
-- name: GetAuthor :one
SELECT * FROM authors
WHERE id = $1 LIMIT 1;
```

Type inference

```
func (q *Queries) GetAuthor(
    ctx context.Context,
    id int64,
) (Author, error) {
  ...
}
```

Catches typos

```
-- name: GetAuthor :one
SELECT first_name FROM authors
WHERE id = $1 LIMIT 1;
```

Advanced SQL

- Common Table Expressions
- Extensions
- Type inference for built-in functions
- Enums
- DDL

Supported databases

- PostgreSQL
- MySQL
- SQLite (beta)

GOPHERCON
2022

SQL
G ERCON
2022

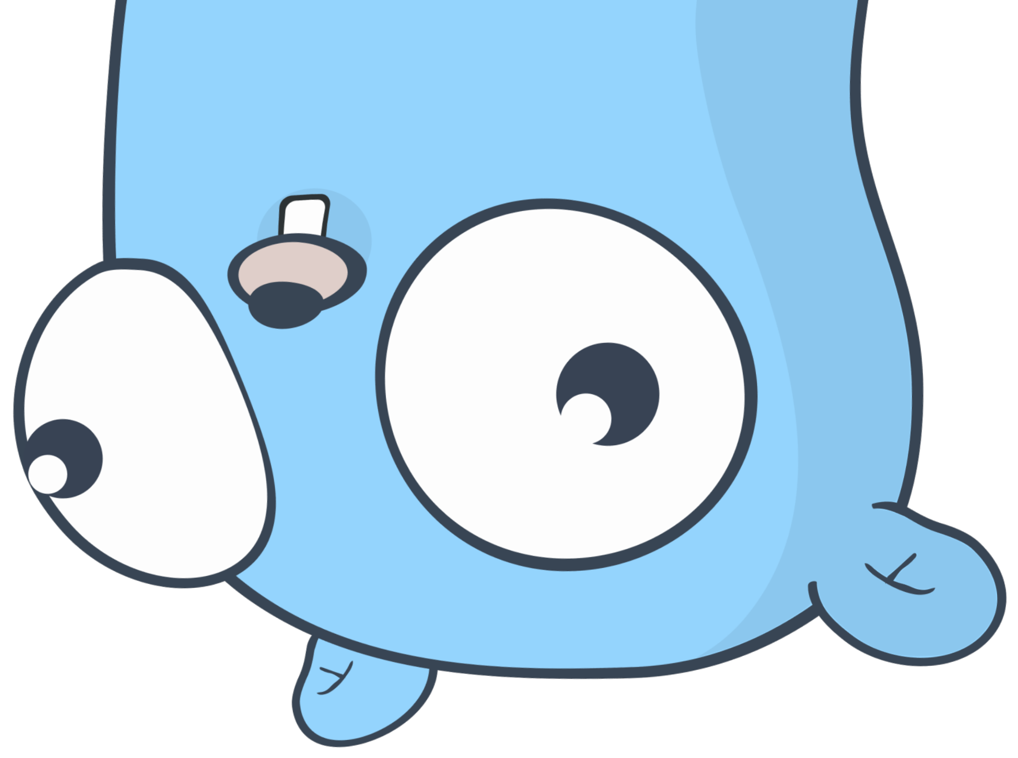
SQL
G ERCON
2022
ORACLE

Compile SQL to type-safe code

Creating magic with ASTs and parsers

Kyle Conroy | conroy.org | @kyle_conroy

Extending sqlc



Extending sqlc

- Changing existing codegen output
- Adding database engines
- Adding programming languages

Extending sqlc

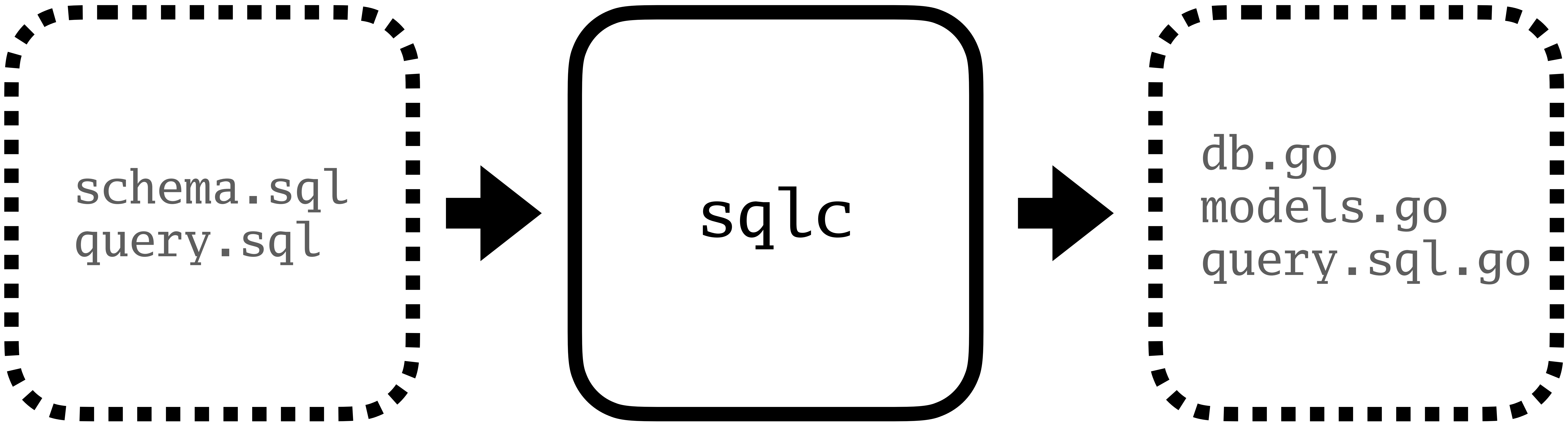
- **Changing existing codegen output**
    - Configuration file grew from a few options to over thirty
- Adding database engines
- Adding programming languages

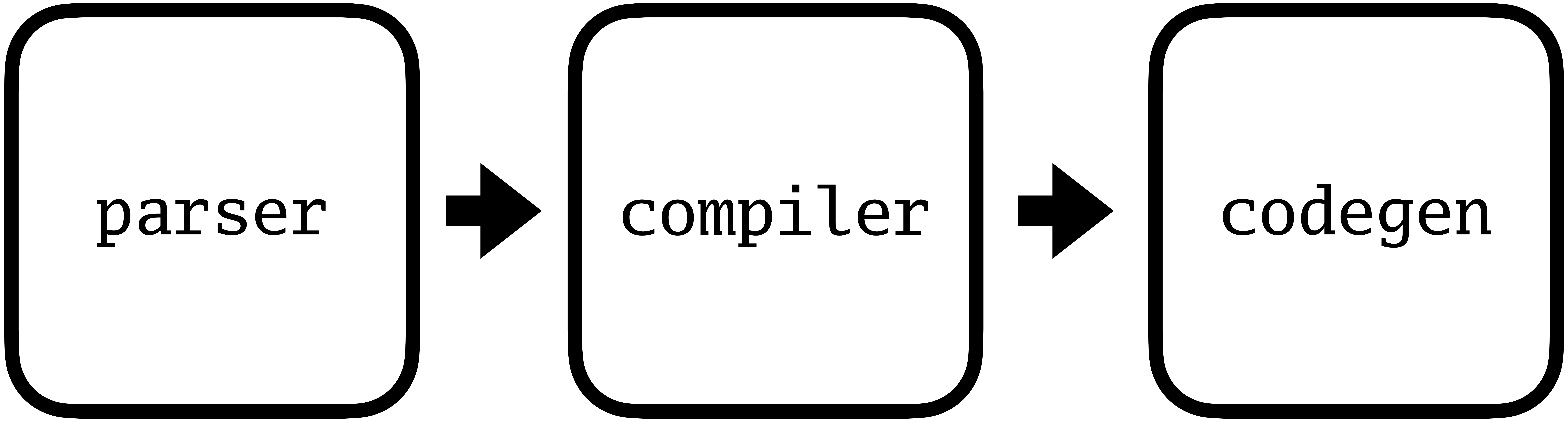
Extending sqlc

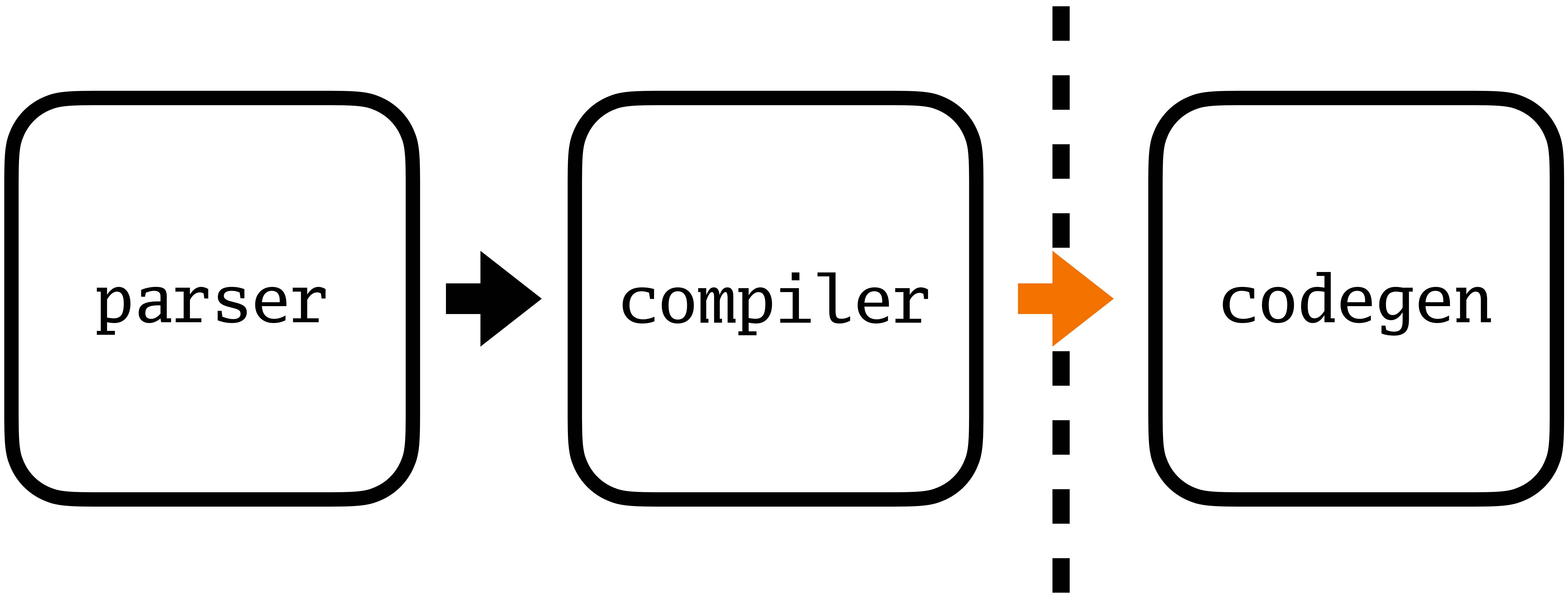
- Changing existing codegen output
- **Adding database engines**
  - Started with PostgreSQL but added MySQL a bit afterwards
- Adding programming languages

Extending sqlc

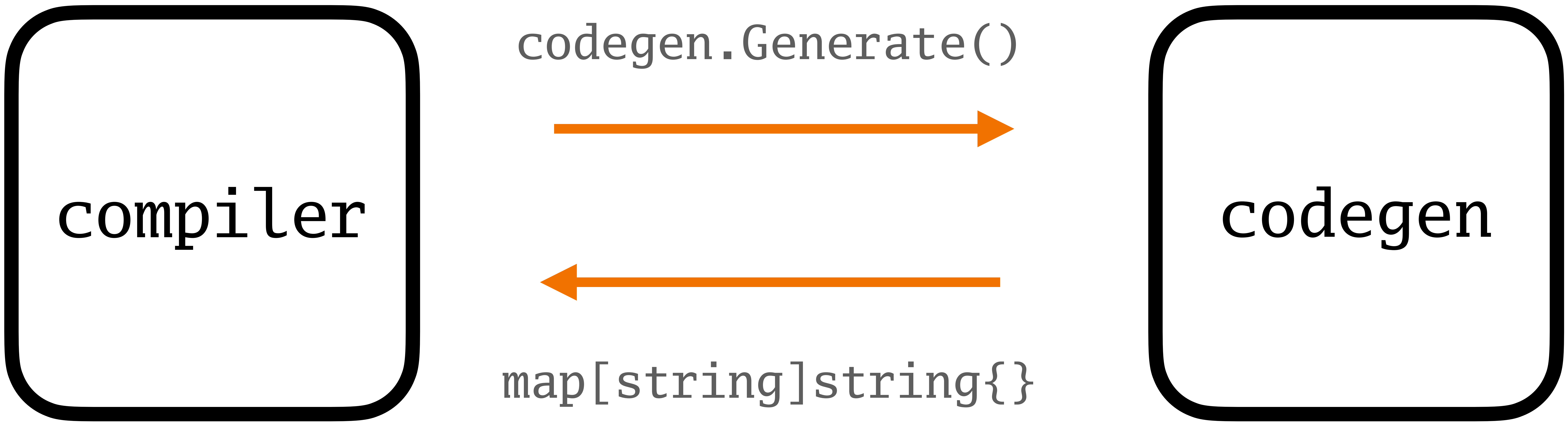
- Changing existing codegen output
- Adding database engines
- **Adding programming languages**
  - Early requests to merge support for Kotlin and Python


schema.sql
query.sql
sqlc
db.go
models.go
query.sql.go


parser
compiler
codegen


parser
compiler
codegen

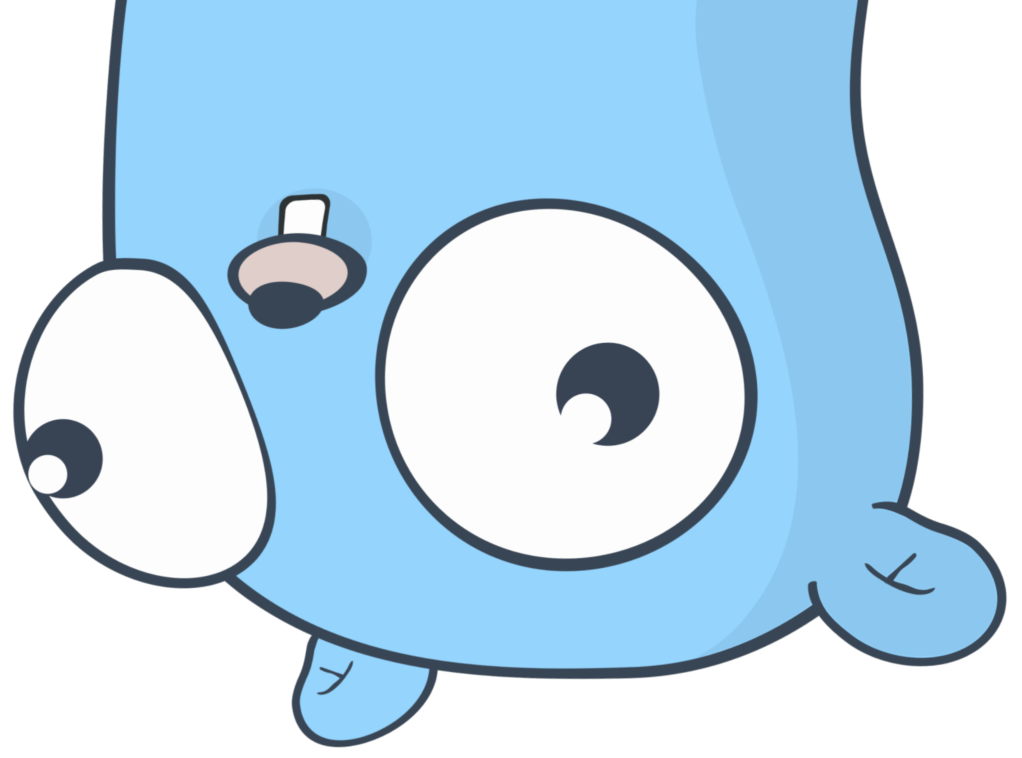
Starting point

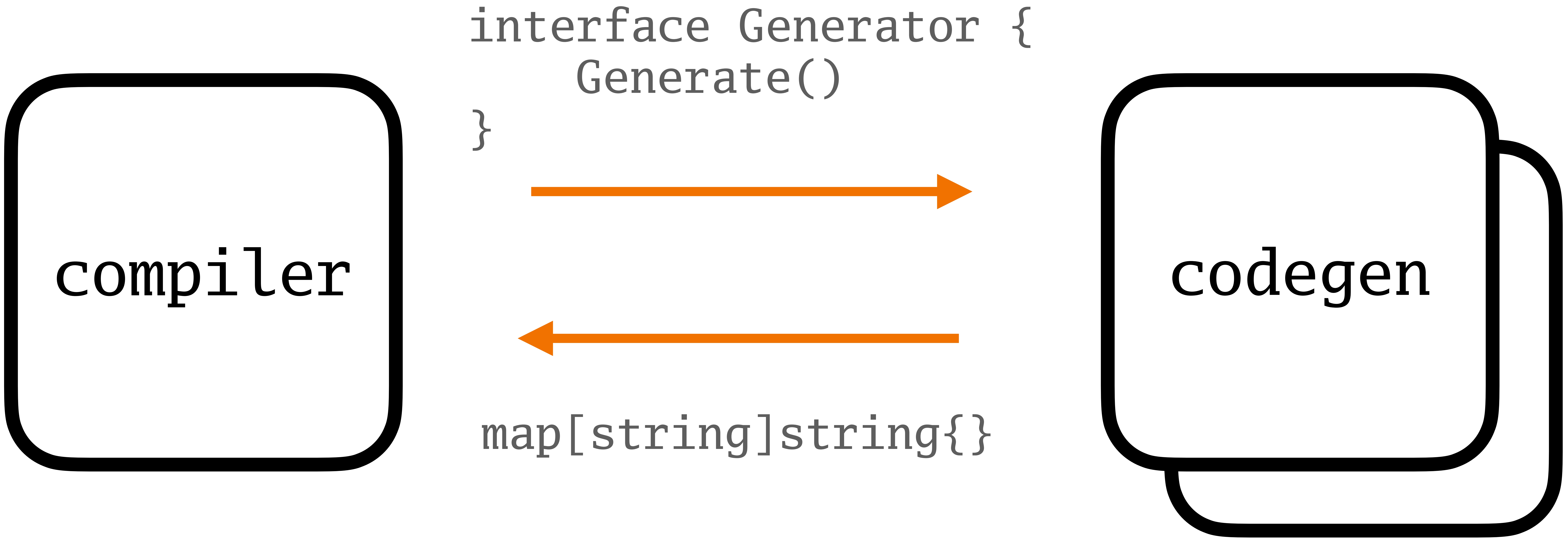


Starting point

- The codegen package only knew how to generate Go
- Imported a ton of internal shared packages
- Mainly implemented via text/template

Go packages





```
interface Generator {
    Generate()
}
```

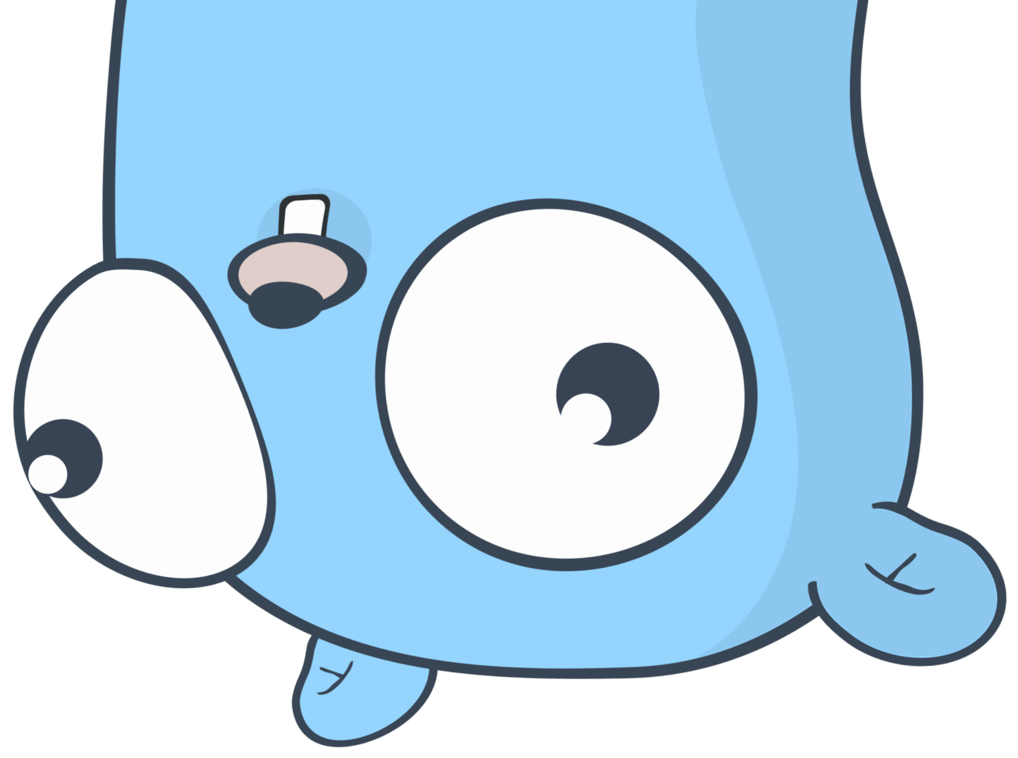
```
map[string]string{}
```

Go packages

- fir

Go packages

- Technically worked, merged Kotlin and Python support
- Many downsides
- Maintenance headaches



Requirements

Requirements

Context

- Command line tool, doesn't control the host environment
- Needs to support Linux, macOS, and Windows
- Plugins for specific languages are best written in that language

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Requirements

- Independent
- Safe
- Run anywhere
- Fast
- Familiar

Go packages

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
❌				

Go packages

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✖	⊖			

Go packages

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✗	—	✓		

Go packages

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✗	⊖	✓	✓	

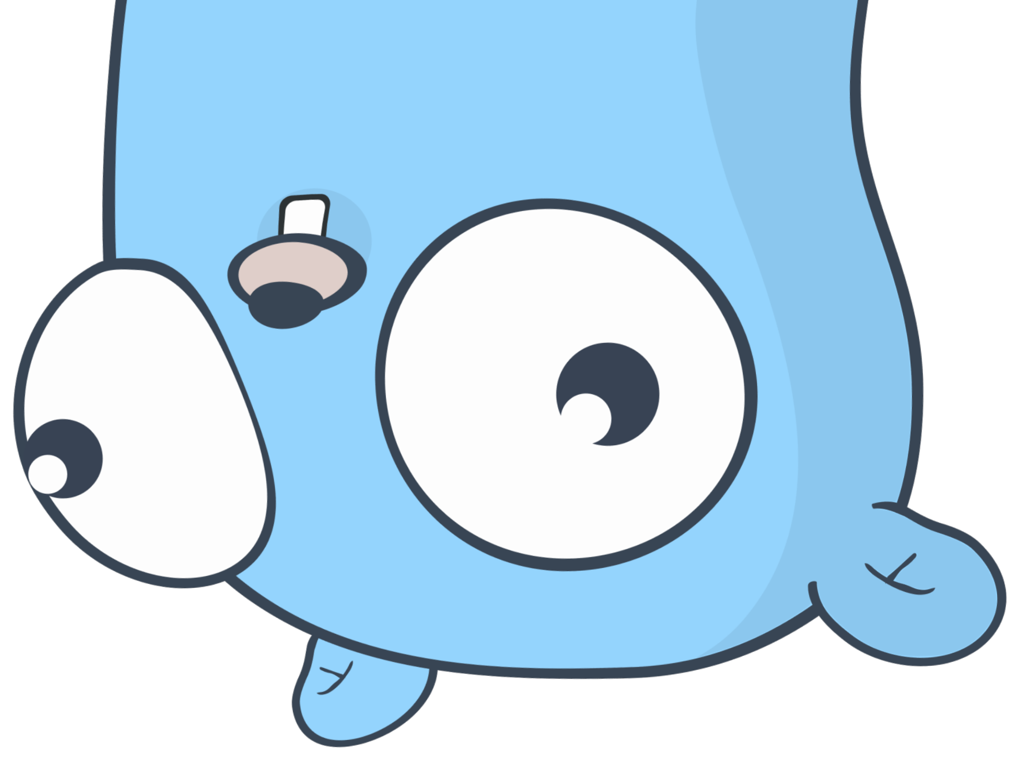
Go packages

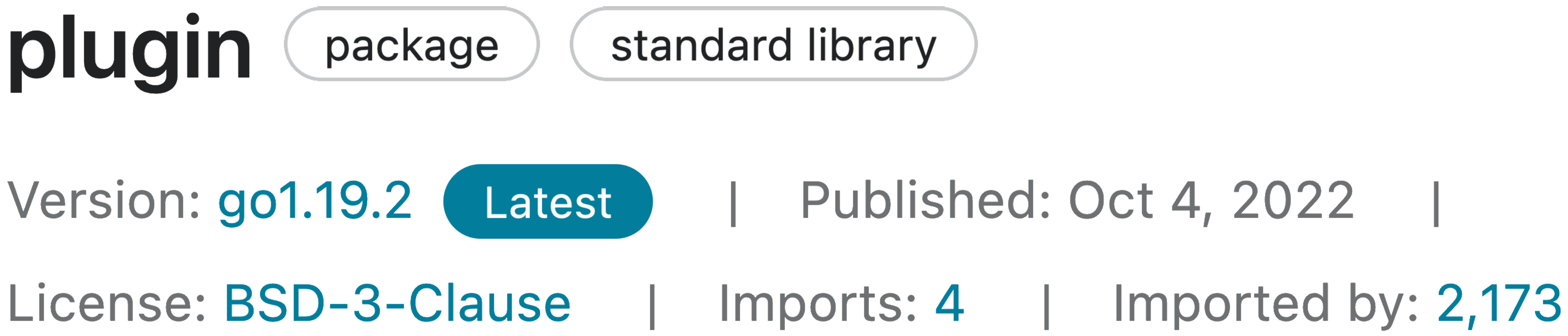
Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✗	−	✓	✓	✓

Attempts

- ~~Go packages~~

plugin

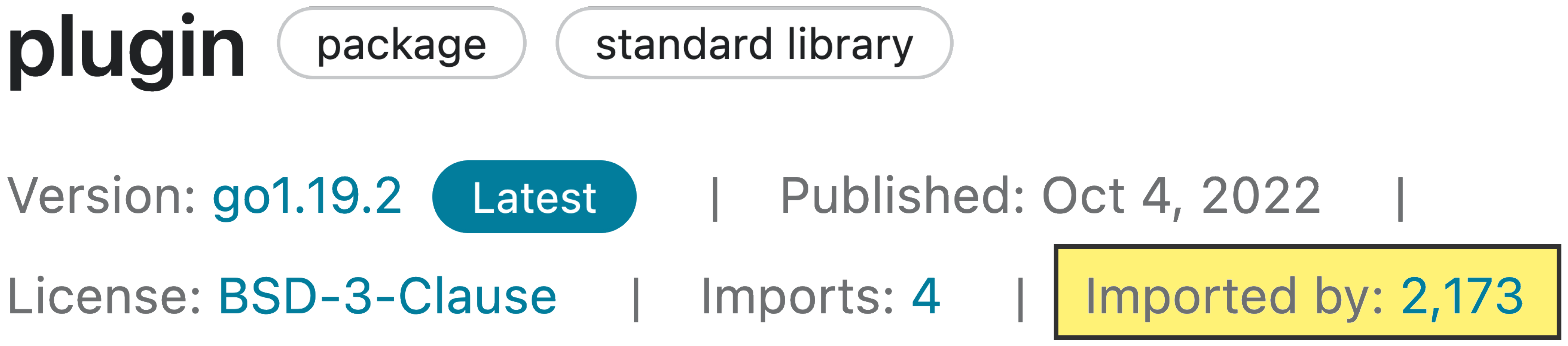




plugin package standard library

Version: go1.19.2 Latest | Published: Oct 4, 2022 |

License: BSD-3-Clause | Imports: 4 | Imported by: 2,173

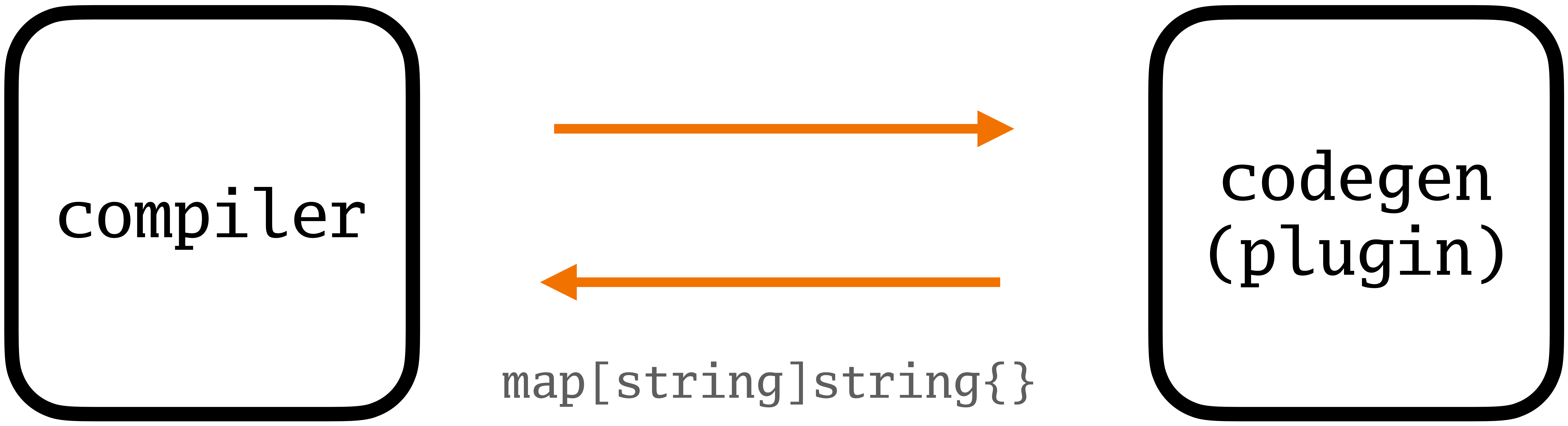


plugin package standard library

Version: go1.19.2 Latest | Published: Oct 4, 2022 |

License: BSD-3-Clause | Imports: 4 | Imported by: 2,173

```
p := plugin.Open("codegen.so")
f := p.Lookup("Generate")
f()
```



```
map[string]string{}
```

plugin

- "Currently plugins are only supported on Linux, FreeBSD, and macOS"
  - No Windows!

plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓				

plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗			

evil_😈_plugin

- Tries to find AWS credentials on your system
  - Looks in ~/.aws/credentials
  - Looks for AWS_SECRET_KEY
- And then send them off to the Cloud™

evil_😈_plugin

"PyPi python packages caught sending stolen AWS keys to unsecured sites"

https://www.bleepingcomputer.com/news/security/pypi-python-packages-caught-sending-stolen-aws-keys-to-unsecured-sites/

plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗		

plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	⊖	

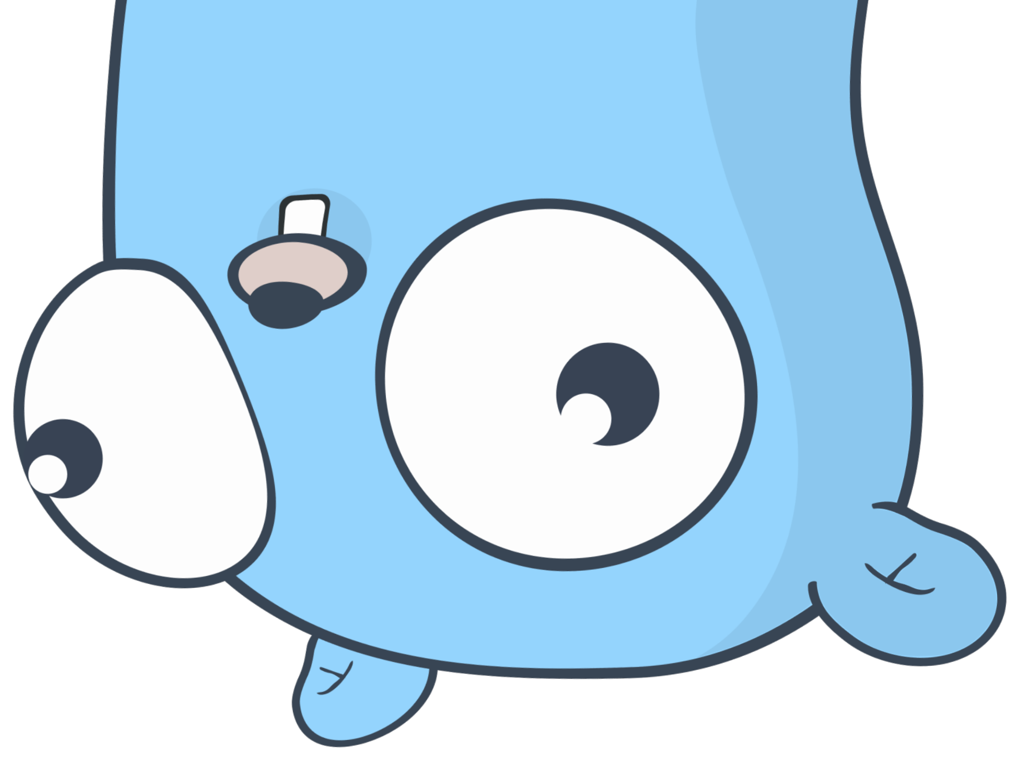
plugin

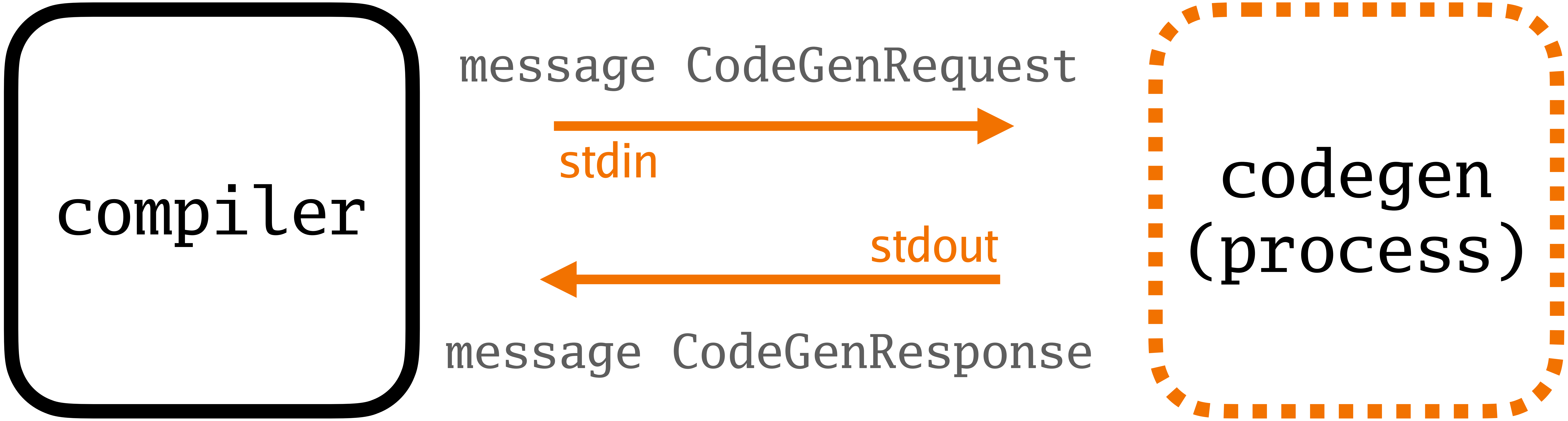
Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	⊖	⊖

Attempts

- ~~Go packages~~
- ~~plugin~~

os/exec




compiler
message CodeGenRequest
stdin
stdout
message CodeGenResponse
codegen
(process)

os/exec

- protoc uses this model
  - Lookup a plugin on $PATH
  - Start a process, send data via STDIN and read from STDOUT
- Well-understood pattern and supported across many languages

os/exec

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓				

os/exec

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗			

os/exec

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗		

os/exec

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	⊖	

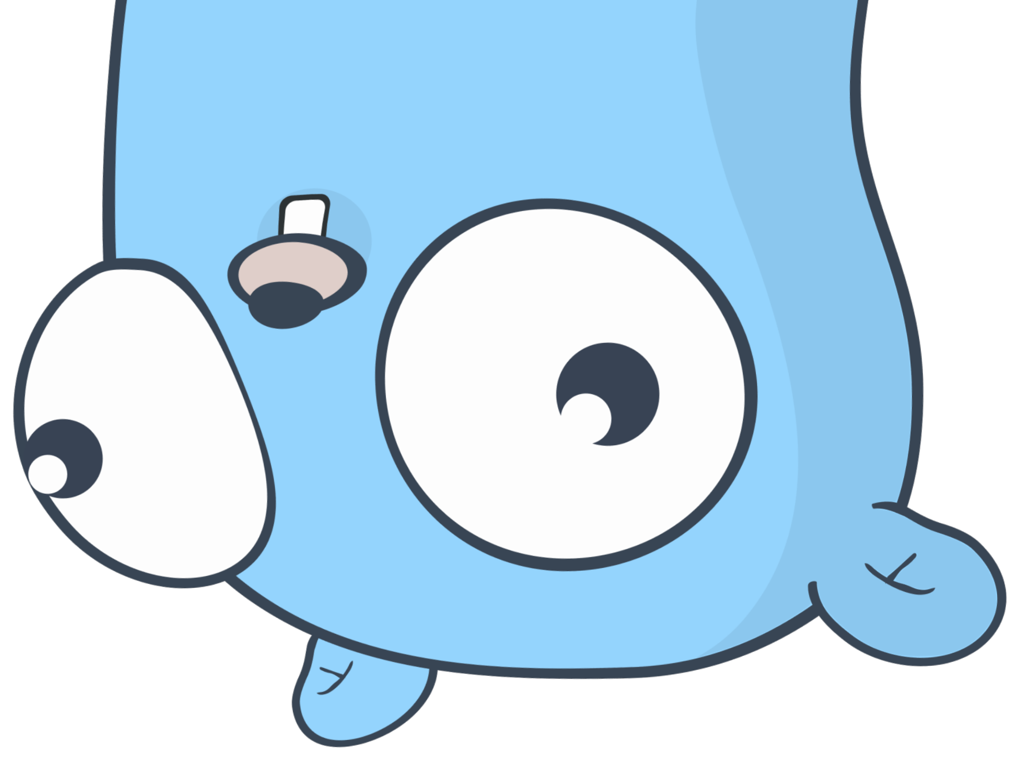
os/exec

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	⊖	✓

Attempts

- ~~Go packages~~
- ~~plugin~~
- ~~os/exec~~

hashicorp/go-plugin



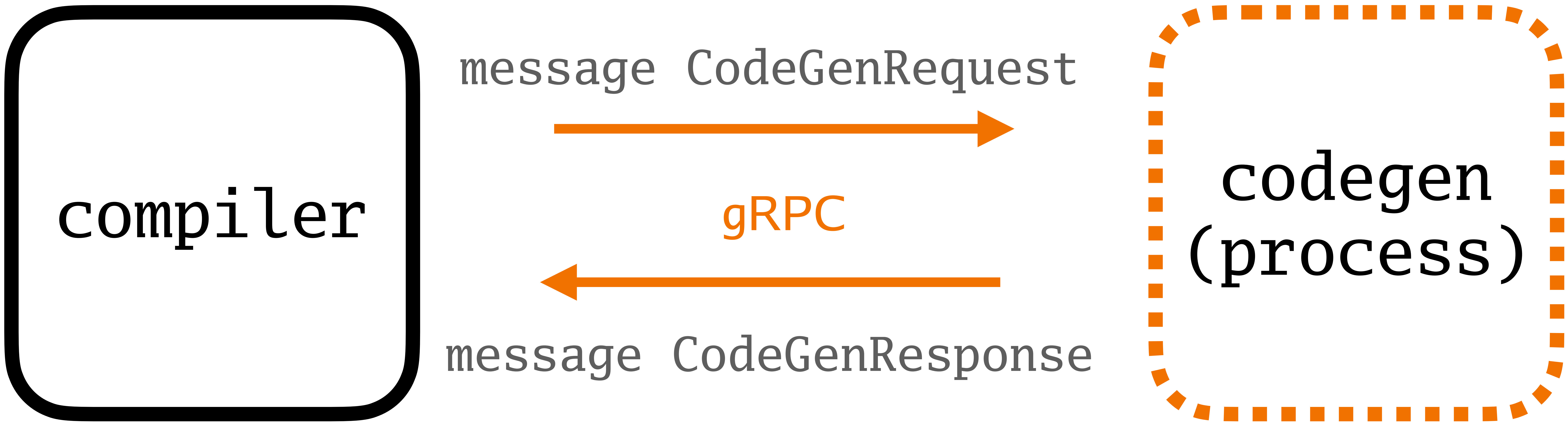
HashiCorp
Nomad

HashiCorp
Boundary

HashiCorp
Terraform

HashiCorp
Vault

HashiCorp
Packer


compiler
message CodeGenRequest
gRPC
message CodeGenResponse
codegen
(process)

hashicorp/go-plugin

- Plugins required to implement a large, complicated interface
    - At the end of the day it's just gRPC
    - I'm still not sure how easy it is to implement in a different language

hashicorp/go-plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓				

hashicorp/go-plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗			

hashicorp/go-plugin

- "Plugins can be **relatively** secure: The plugin only has access to the interfaces and args given to it, not to the entire memory space of the process. Additionally, go-plugin can communicate with the plugin over TLS."
- Don't be fooled!

hashicorp/go-plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗		

hashicorp/go-plugin

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	−	

hashicorp/go-plugin

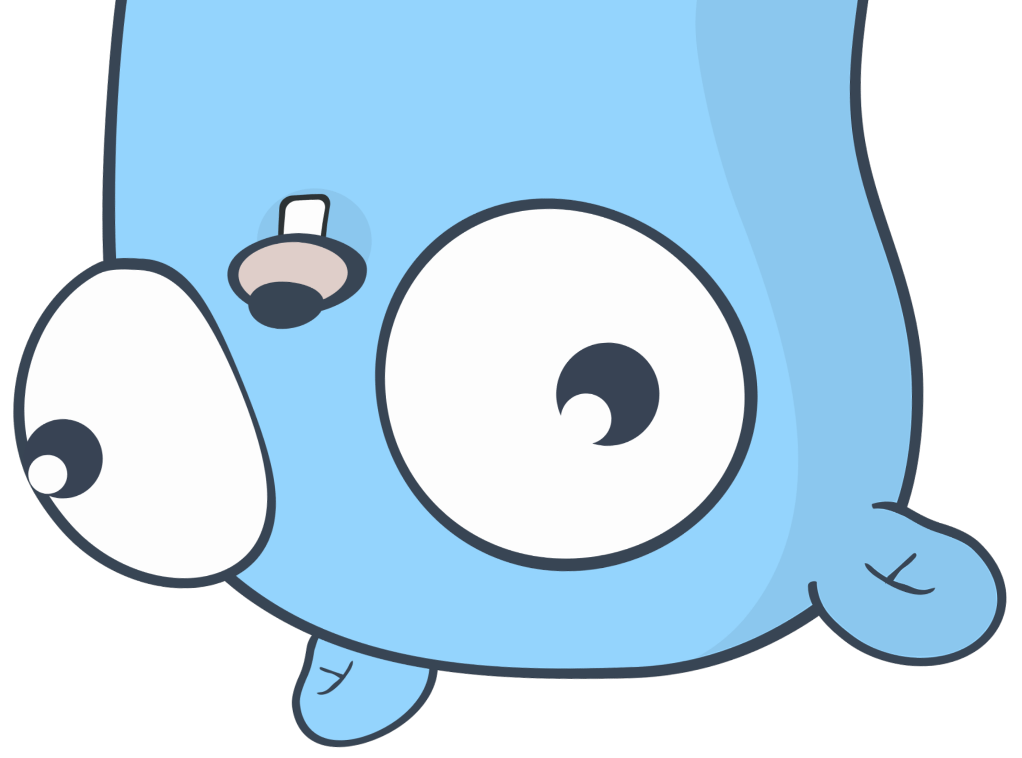
Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✗	✗	⊖	⊖

hashicorp/go-plugin

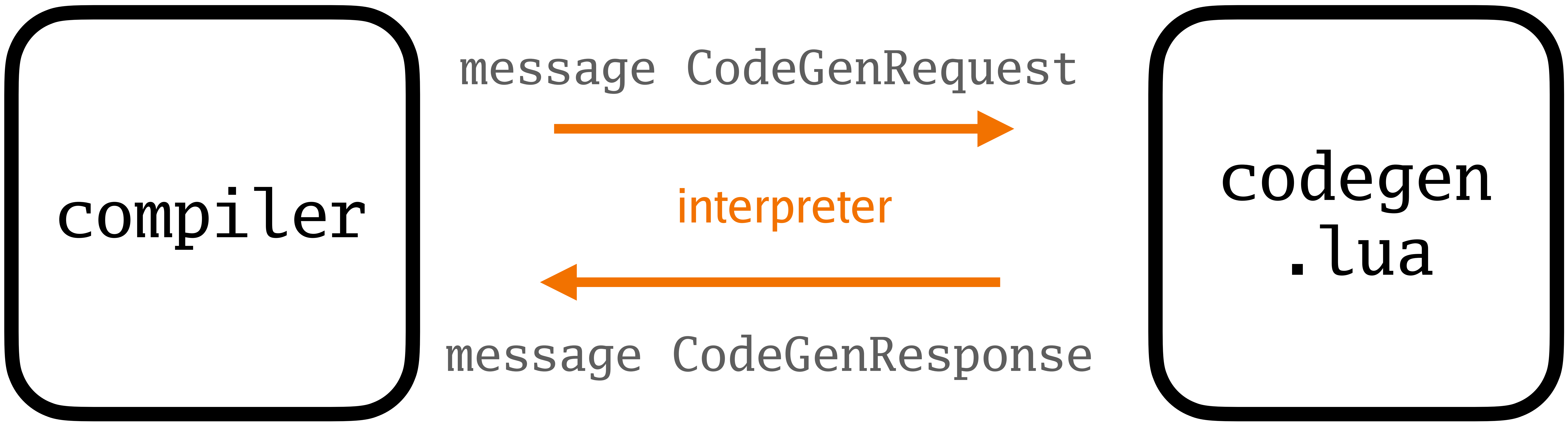
- Obviously not a bad choice!
- Powers an ecosystem many, many times larger than sqlc
- Much easier to secure if you're running the plugins in the cloud

Attempts

- ~~Go packages~~
- ~~plugin~~
- ~~os/exec~~
- ~~hashicorp/go-plugin~~



Embedded scripting


compiler
message CodeGenRequest
interpreter
message CodeGenResponse
codegen
.lua

Embedded scripting

- Lua / JavaScript / Python
- Have to find an interpreter that meets your needs

Embedded scripting

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
⊖				

Embedded scripting

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
⊖	⊖			

Embedded scripting

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
⊖	⊖	✓		

Embedded scripting

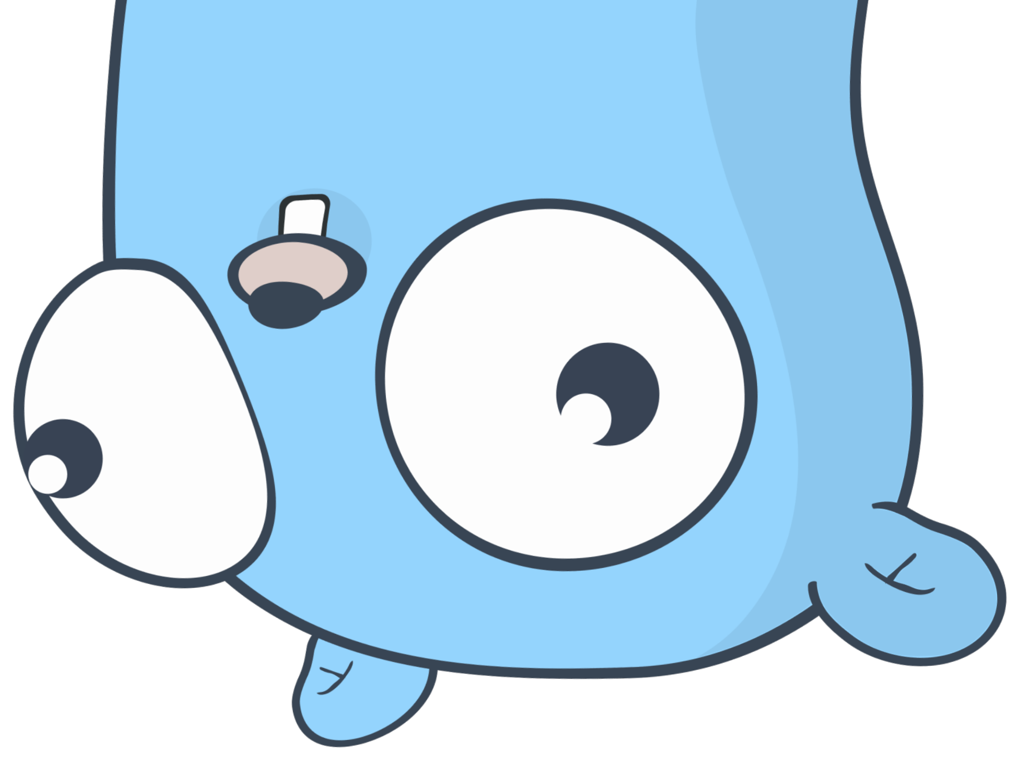
Independent?	Secure?	Run anywhere?	Fast?	Familiar?
⊖	⊖	✓	⊖	

Embedded scripting

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
⊖	⊖	✓	⊖	✗

Attempts

- ~~Go packages~~
- ~~plugin~~
- ~~os/exec~~
- ~~hashicorp/go-plugin~~
- ~~scripting~~



bytecodealliance/wasmtime-go

"A fast and secure runtime for WebAssembly"

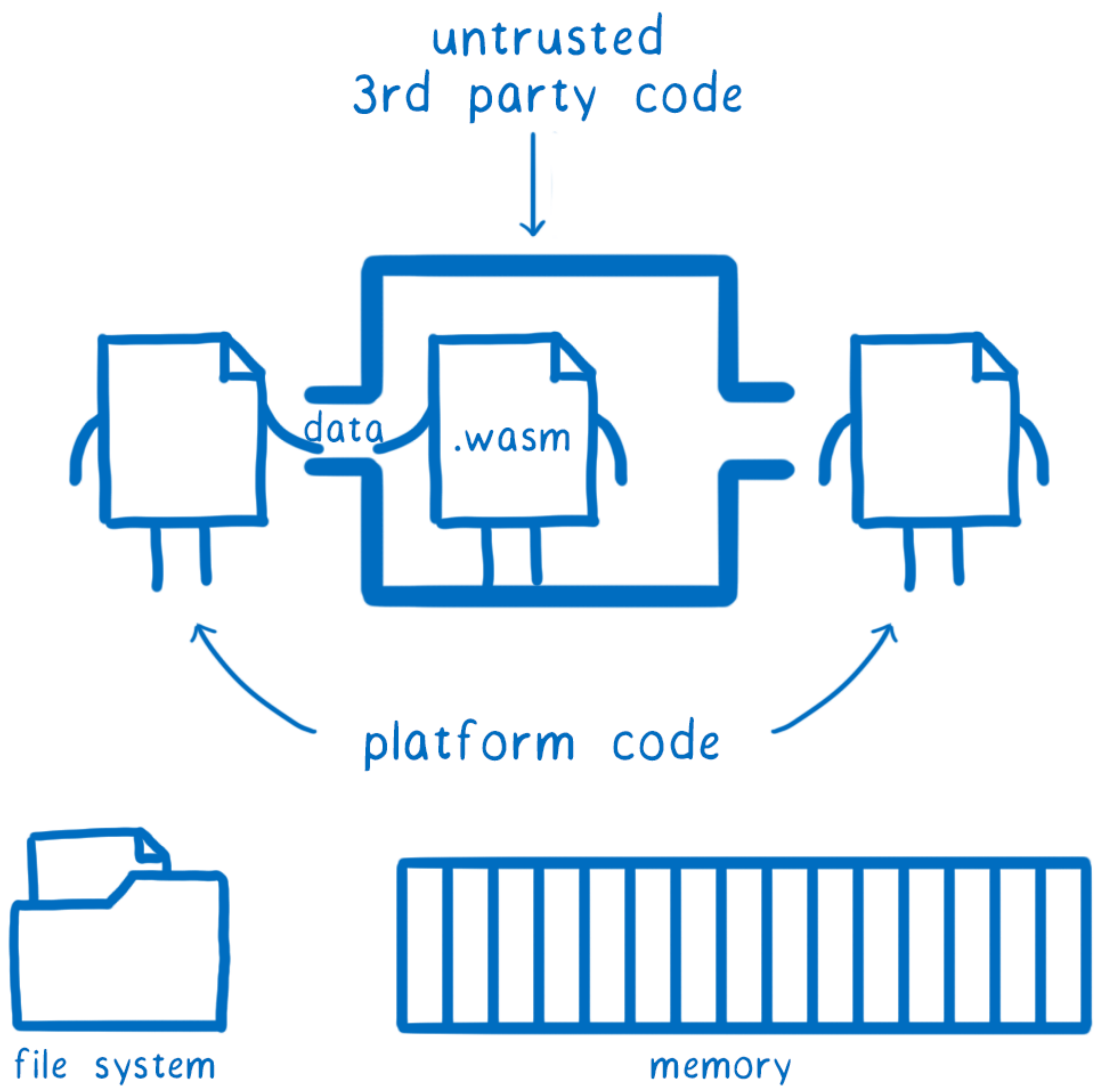
https://wasmtime.dev/

"WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications."

https://webassembly.org/

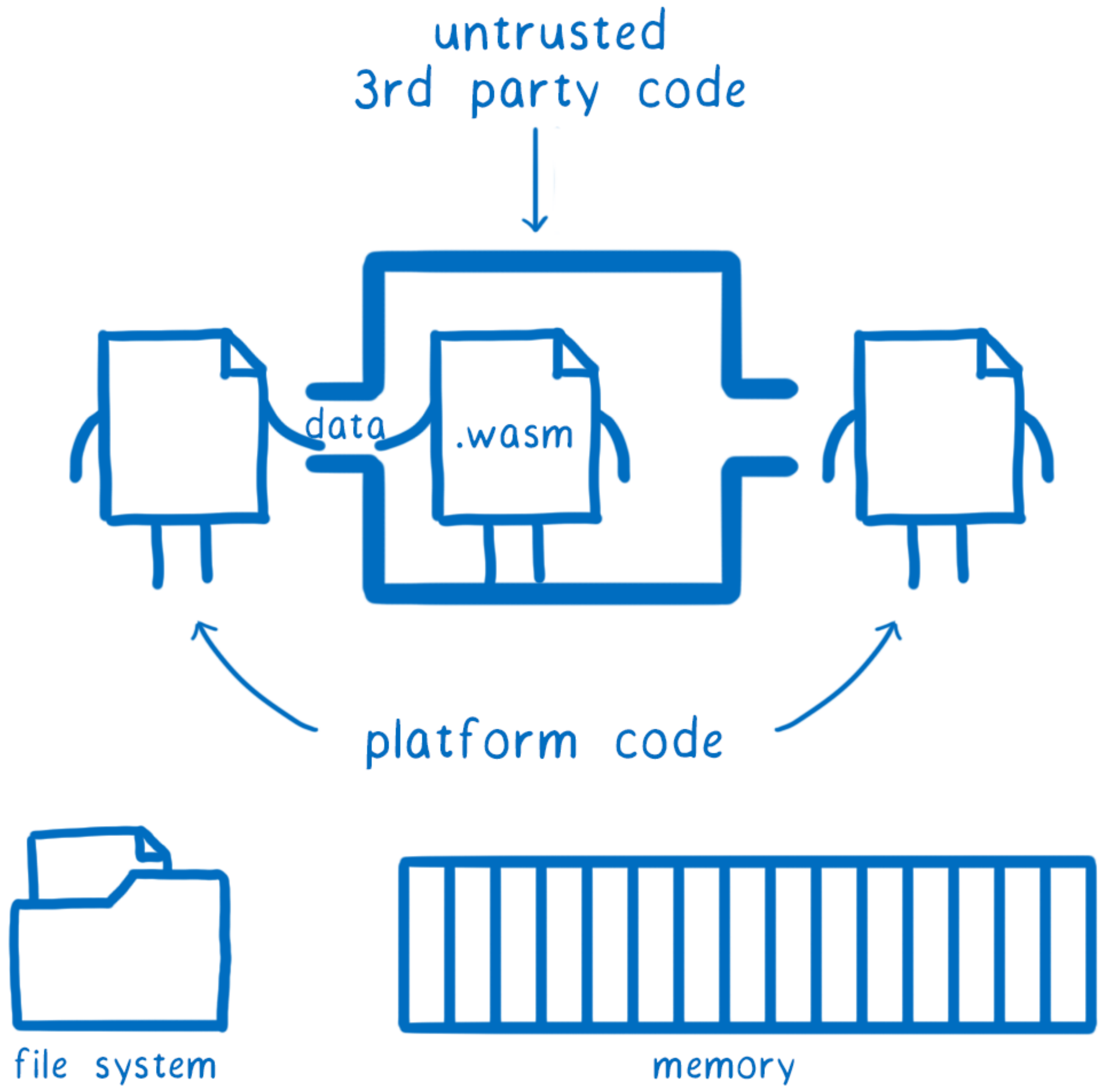
3rd Party Plugin systems

WebAssembly is great for platforms, where you often want to run 3rd party code so you can support many different specific use cases—for example, **plug-in** marketplaces where developers in the platform's **ecosystem** can share code with users."



3rd Party Plugin systems

WebAssembly is great for platforms, where you often want to run 3rd party code so you can support many different specific use cases—for example**, plug-in** marketplaces where developers in the platform's **ecosystem** can share code with users."



bytecodealliance/wasmtime-go

- Recently-released stable `1.0.0`!
- Platform-independent modules!
- Secure by default!

bytecodealliance/wasmtime-go

Security

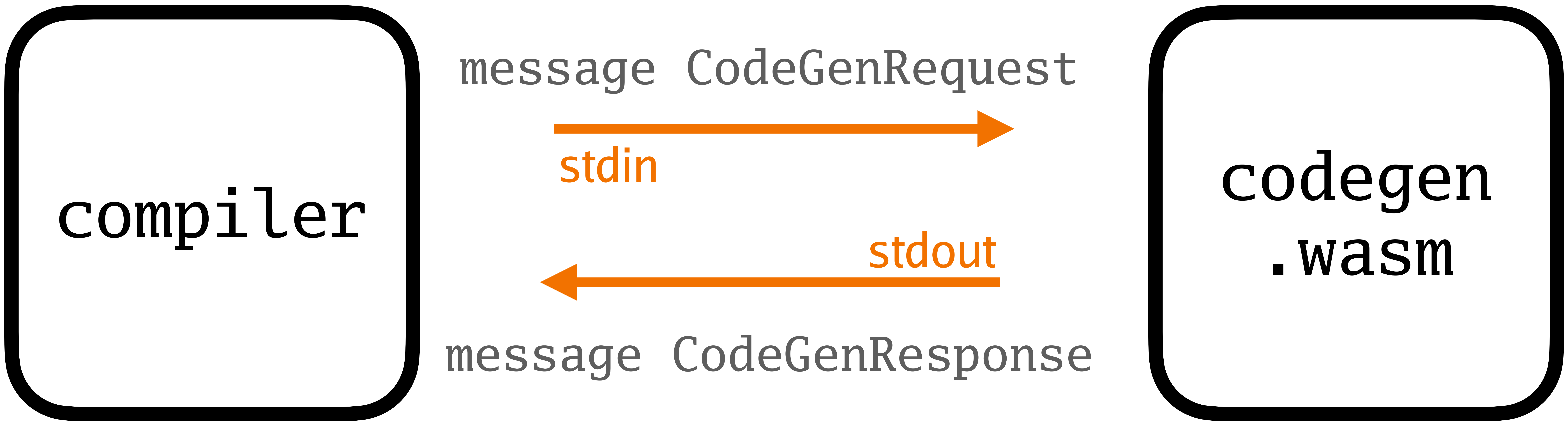
- Capability-based security system
- By default:
  - No filesystem access
  - No network access
  - No environment variables
- Can give access to specific directories

bytecodealliance/wasmtime-go

- Wasm is low-level; doesn't know about things like strings, objects, etc.
- Wasm-only interfaces are... intimidating

"WASI stands for WebAssembly System Interface. It's an API ... that provides access to several operating-system-like features, including files and filesystems..."

https://wasi.dev/


compiler
message CodeGenRequest
stdin
stdout
message CodeGenResponse
codegen
.wasm

bytecodealliance/wasmtime-go

- We've actually gone full circle back to `exec/cmd`
- Wasm + WASI is the same model as process-based plugins

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓				

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓			

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓	✓		

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓	✓	✓	

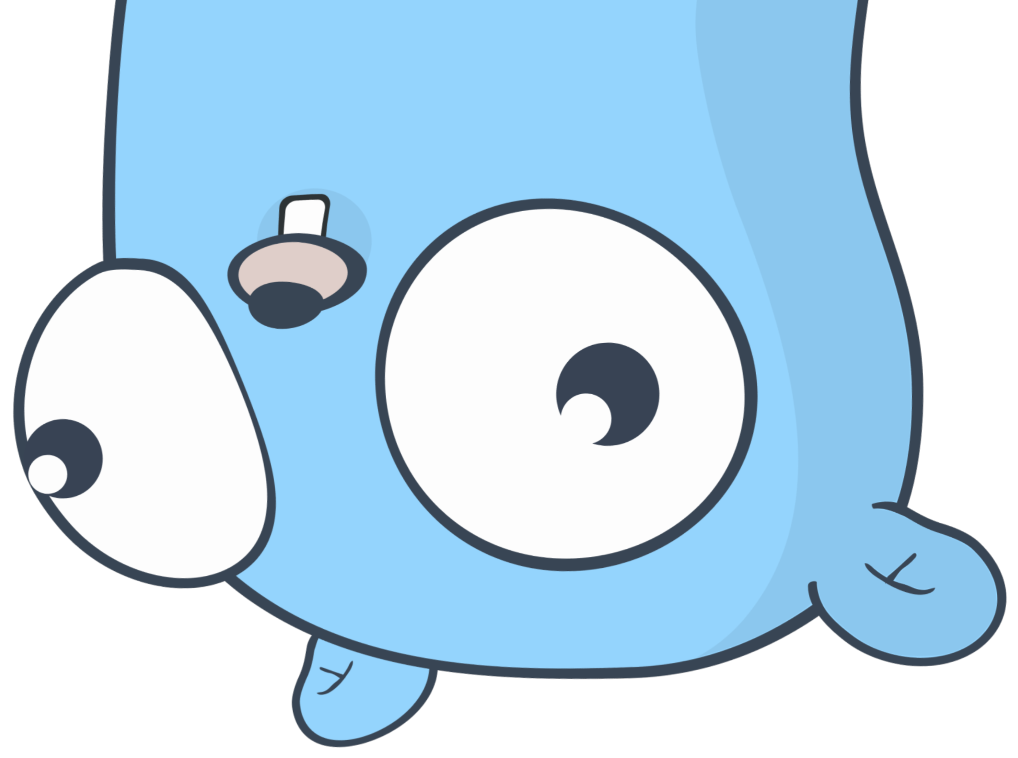
bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓	✓	✓	⊖

Attempts

- ~~Go packages~~
- ~~plugin~~
- ~~os/exec~~
- ~~hashicorp/go-plugin~~
- ~~scripting~~
- **wasmtime-go**

Implementation



Configuration

```yaml
version: '2'
plugins:
- name: greeter
  wasm:
    url: https://github.com/kyleconroy/.../sqlc-gen-greeter.wasm
    sha256: "afc486dac206......cd802424ad07"
```

Configuration

```yaml
sql:
- schema: schema.sql
  queries: query.sql
  engine: postgresql
  codegen:
  - out: gen
    plugin: greeter
```

How it works

1. Load the configuration file
2. Download each plugin's .wasm file (if necessary)
3. Create a wasmtime execution context (engine / module / linker)
4. Set up stdin, stdout, stderr
5. Call the context, read stdout

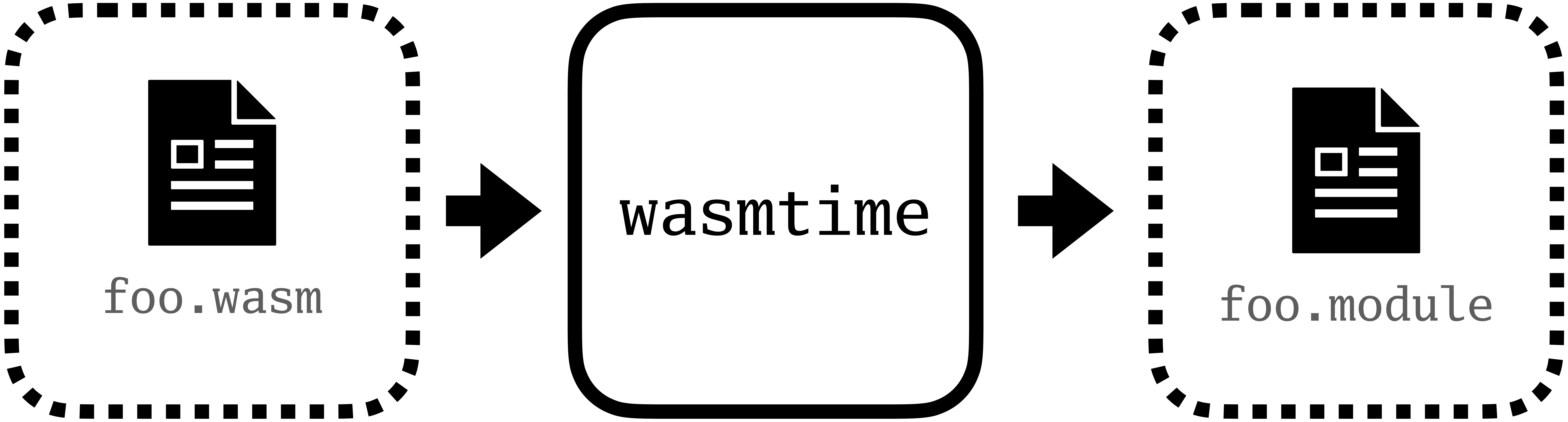
Speed

baseline	wasmtime-go
0.035s	0.955s

27x

slower

Speed (bump)



How it works (faster)

1. Load the configuration file
2. Download each plugin's .wasm file (if necessary)
3. **Turn the .wasm file into a module, save to disk**
4. Create a wasmtime execution context (engine / linker)
5. Set up stdin, stdout, stderr
6. Call the context, read stdout

Speed (up)

baseline	wasmtime-go
0.035s	0.055s

`internal/ext/wasm/wasm.go`

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓	✓	✓	⊖

bytecodealliance/wasmtime-go

Independent?	Secure?	Run anywhere?	Fast?	Familiar?
✓	✓	✓	✓	✗

GOOS=js GOARCH=wasm

- Go's builtin Wasm support does not support WASI
- So off to TinyGo we... go!

TinyGo, big problems

- TinyGo doesn't work with common serialization tools
  - No encoding/json
  - No encoding/xml
  - No protobuf

GOPHERCON

2022

2022
RUST
CONF
GO
ON

Crustaceans and Codegen

Building sqlc plugins in Rust

Kyle Conroy | conroy.org | @kyle_conroy

sqlc-gen-greeter

https://github.com/kyleconroy/sqlc-gen-greeter

- Written in Rust
- Extremely basic, just outputs "Hello world"

sqlc-gen-node-pg

https://github.com/tabbed/sqlc-gen-node-pg

- TypeScript output for sqlc
- Written in Rust
- Still a work-in-progress

Attempts

- ~~Go packages~~
- ~~plugin~~
- ~~os/exec~~
- ~~hashicorp/go-plugin~~
- ~~scripting~~
- **wasmtime-go**

Attempts

- ~~Go packages~~
- ~~plugin~~
- **os/exec**
- ~~hashicorp/go-plugin~~
- ~~scripting~~
- **wasmtime-go**

sqlc-gen-json

https://github.com/kyleconroy/sqlc/tree/main/cmd/sqlc-gen-json

- Process-based plugin written in Go
- Not all languages support Wasm and WASI
- No distribution built-in (on purpose)

2

asks

Serialize and deserialize protocol buffers

https://github.com/tinygo-org/tinygo/issues/2667

Go toolchain support for WASI

Thanks!

- Twitter: @kyle_conroy
- Email: kyle@conroy.org