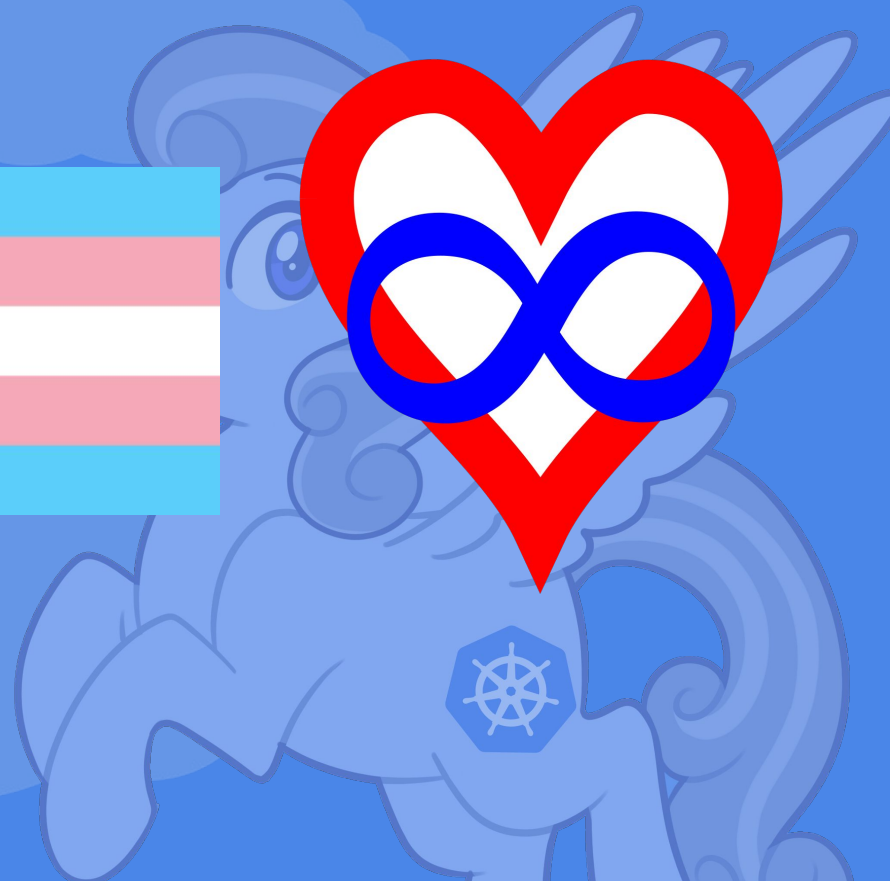
You got Database in my Cloud!

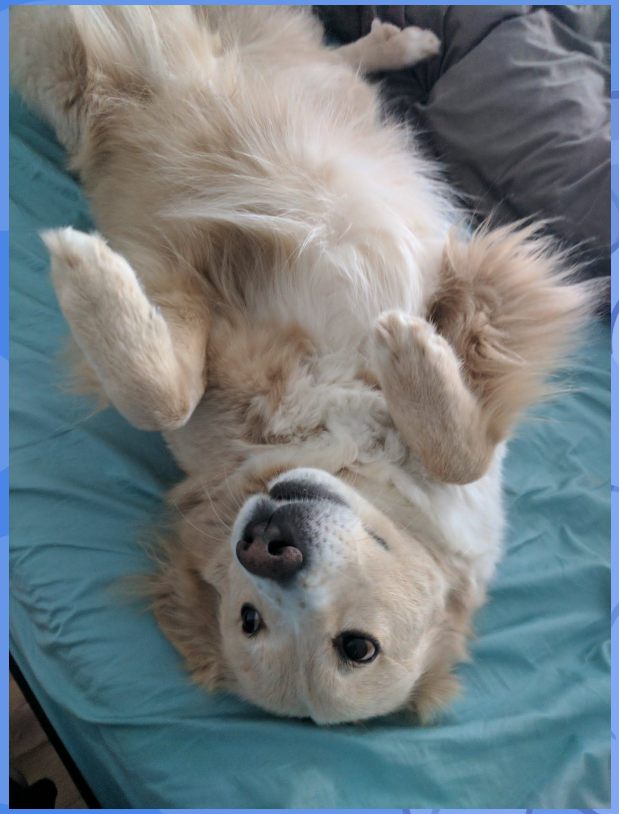
—

Kubernetes Foreign Data Wrapper for Postgres

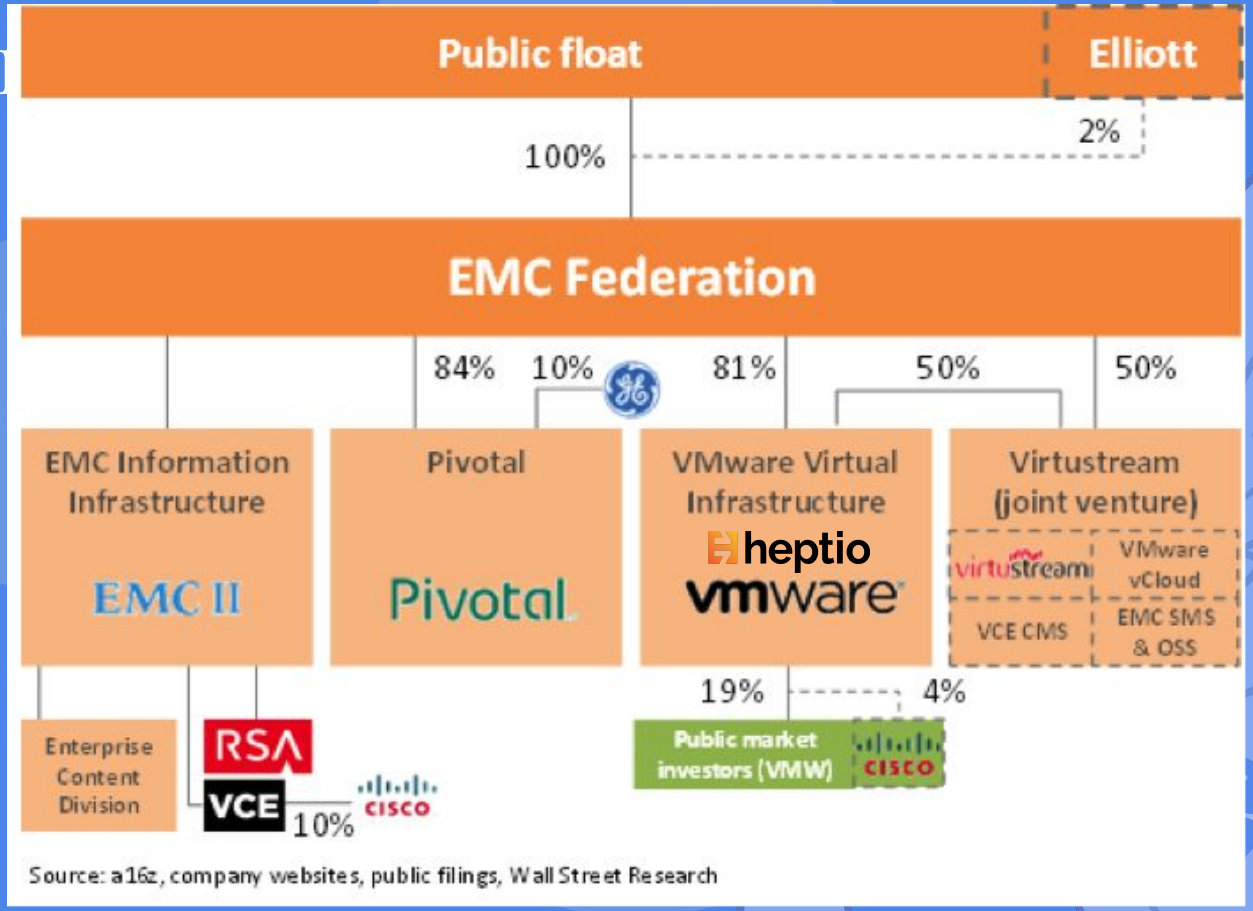
Who am I?



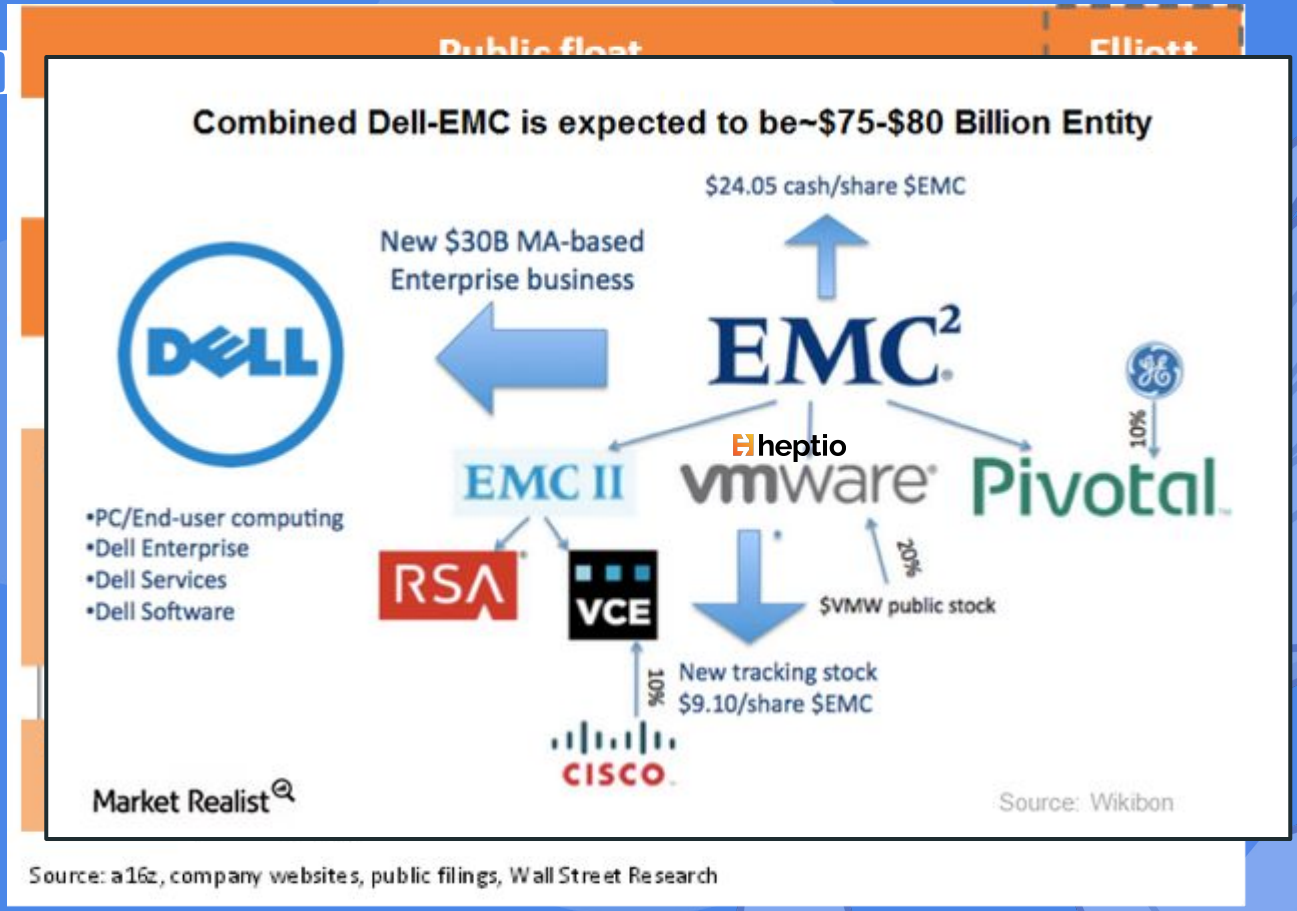
Who am I?



Who a


Public float
Elliott
100%
2%
EMC Federation
84%
10%
81%
50%
50%
EMC Information Infrastructure
EMC II
Pivotal
Pivotal
VMware Virtual Infrastructure
heptio
vmware
Virtustream (joint venture)
virtustream
VMware vCloud
VCE CMS
EMC SMS & OSS
19%
4%
Enterprise Content Division
RSA
VCE
10%
cisco
Public market investors (VMW)
cisco
Source: a16z, company websites, public filings, Wall Street Research

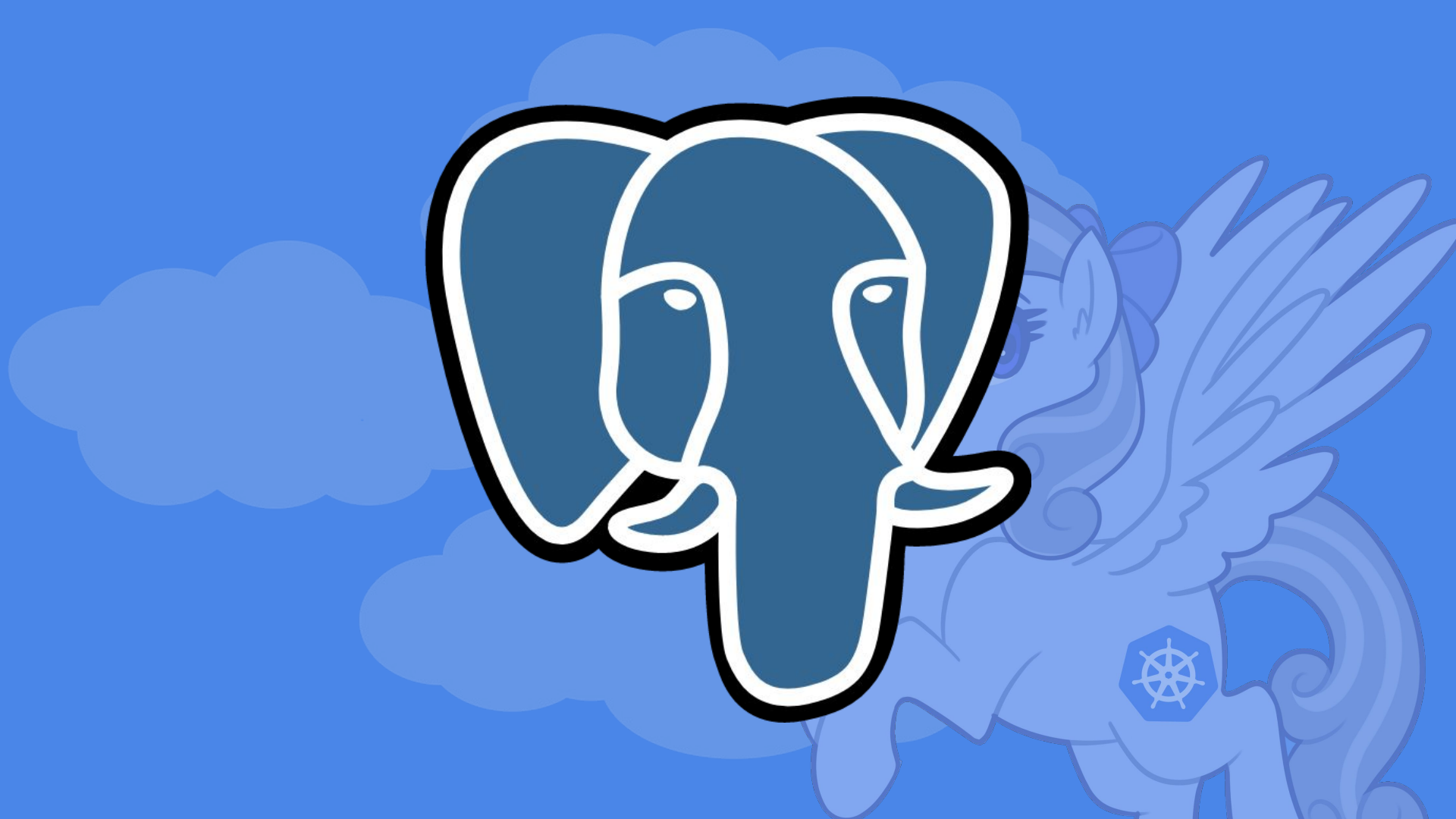
Who a



Source: a16z, company websites, public filings, Wall Street Research

Who am I?

(Stickers are available)



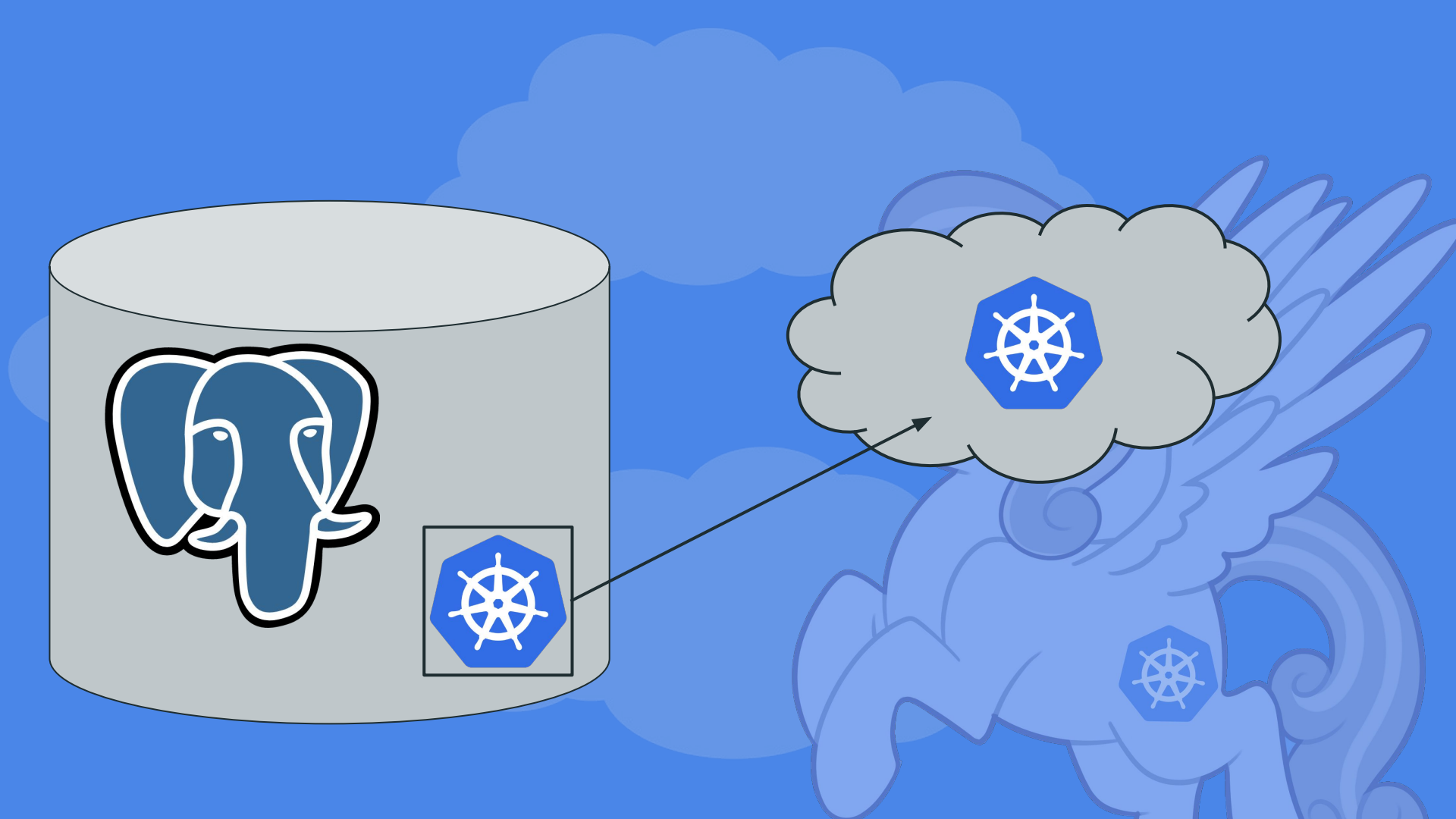

zéro un zéro
cero uno cero
零一零

What is a Foreign Data Wrapper?

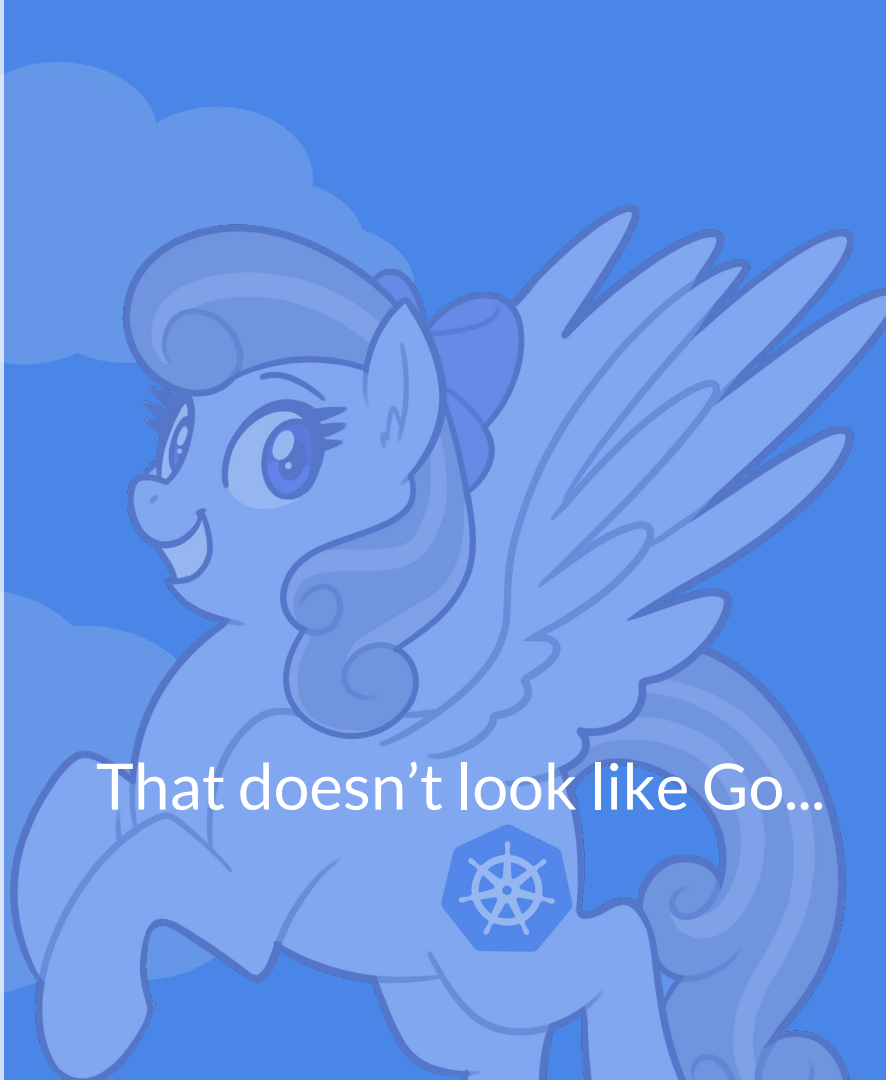
- ODBC
- SQLite

- Cassandra
- Redis

- Hue?
- FDWs?
- Kubernetes!

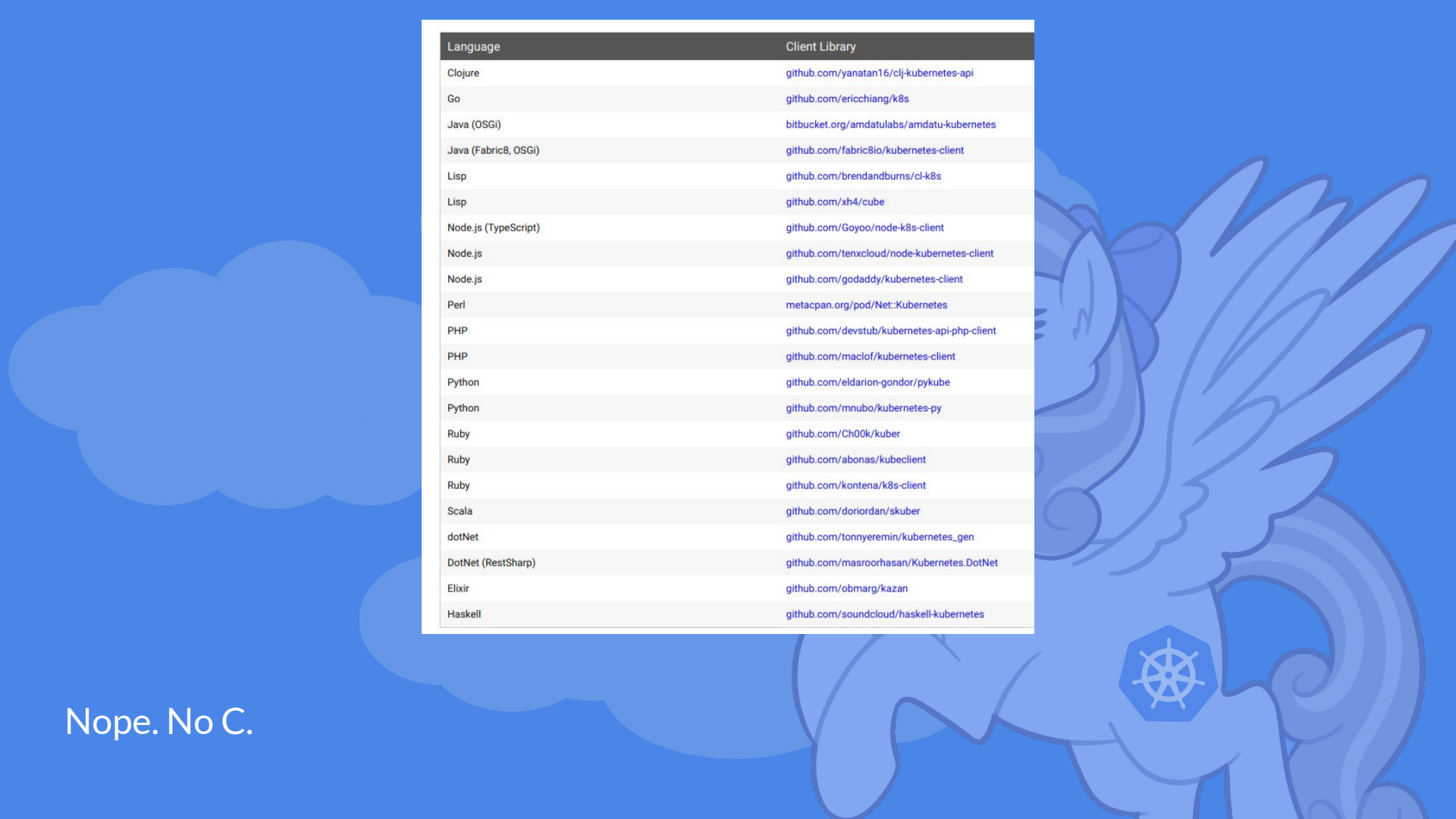


```
void
BeginForeignScan (ForeignScanState *node,
                  int eflags);

TupleTableSlot *
IterateForeignScan (ForeignScanState *node);

ForeignScan *
GetForeignPlan (PlannerInfo *root,
                RelOptInfo *baserel,
                Oid foreigntableid,
                ForeignPath *best_path,
                List *tlist,
                List *scan_clauses,
                Plan *outer_plan);
```

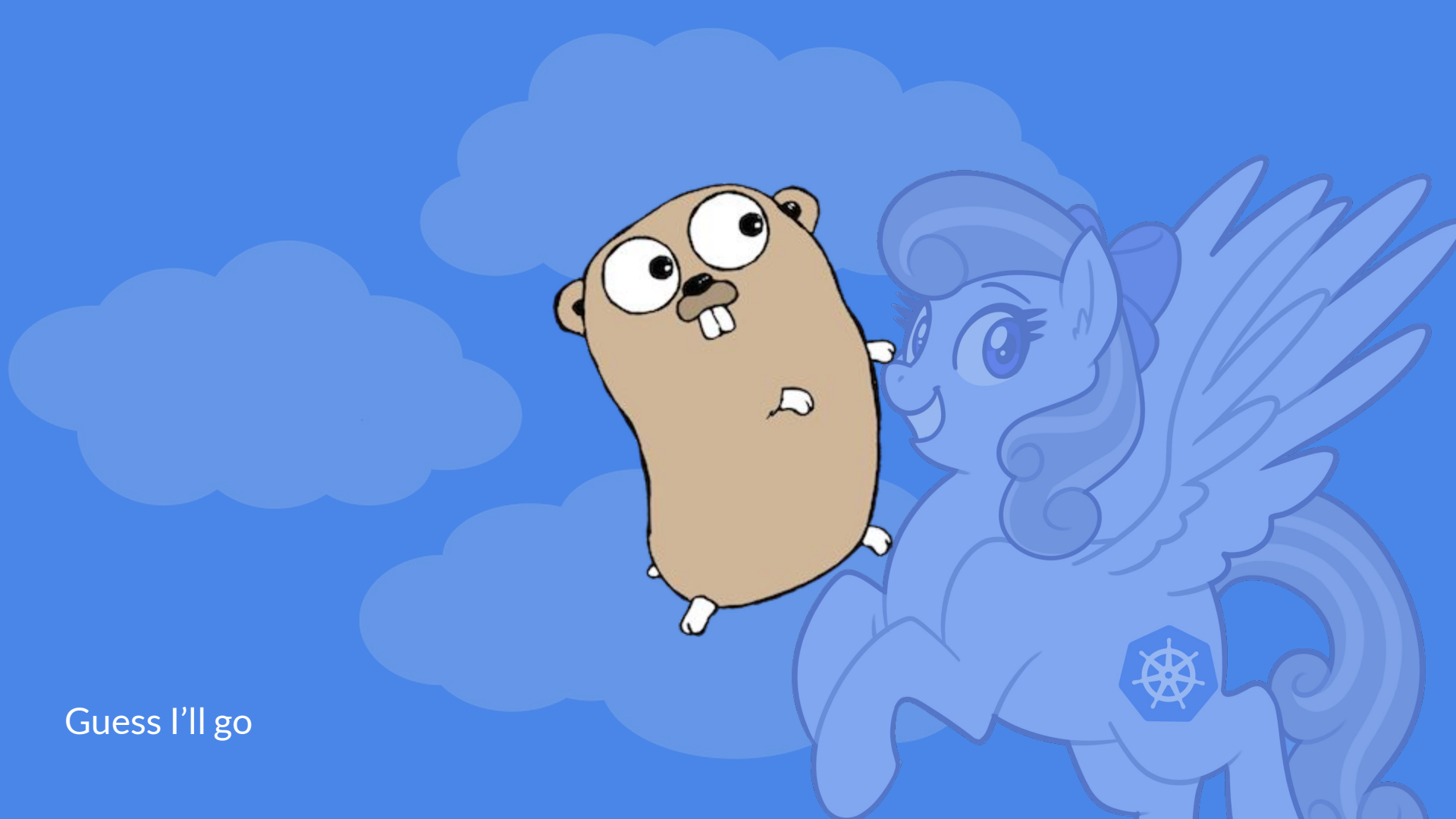



Why not Multicorn?

Language	Client Library
Clojure	github.com/yanatan16/clj-kubernetes-api
Go	github.com/ericchiang/k8s
Java (OSGi)	bitbucket.org/amdatulabs/amdatu-kubernetes
Java (Fabric8, OSGi)	github.com/fabric8io/kubernetes-client
Lisp	github.com/brendandburns/cl-k8s
Lisp	github.com/xh4/cube
Node.js (TypeScript)	github.com/Goyoo/node-k8s-client
Node.js	github.com/tenxcloud/node-kubernetes-client
Node.js	github.com/godaddy/kubernetes-client
Perl	metacpan.org/pod/Net::Kubernetes
PHP	github.com/devstub/kubernetes-api-php-client
PHP	github.com/maclof/kubernetes-client
Python	github.com/eldarion-gondor/pykube
Python	github.com/mnubo/kubernetes-py
Ruby	github.com/Ch00k/kuber
Ruby	github.com/abonas/kubeclient
Ruby	github.com/kontena/k8s-client
Scala	github.com/doriordan/skuber
dotNet	github.com/tonnyeremin/kubernetes_gen
DotNet (RestSharp)	github.com/masroorhasan/Kubernetes.DotNet
Elixir	github.com/obmarg/kazan
Haskell	github.com/soundcloud/haskell-kubernetes



Nope. No C.


Guess I’ll go

README.md

Go FDW for PostgreSQL

An experimental Go project template for building PostgreSQL Foreign Data Wrappers (FDW).

Tested with PostgreSQL v9.6 and Go 1.8.1 on Ubuntu x64.

Supports:

- Table scan
- EXPLAIN
- Table options

Contributions are welcome!

Getting started

Module entry point is defined in `fdw.go` file (see `SetTable`). This file contains a basic working example, so give it a try. Later you will need to rewrite it to suit your needs.

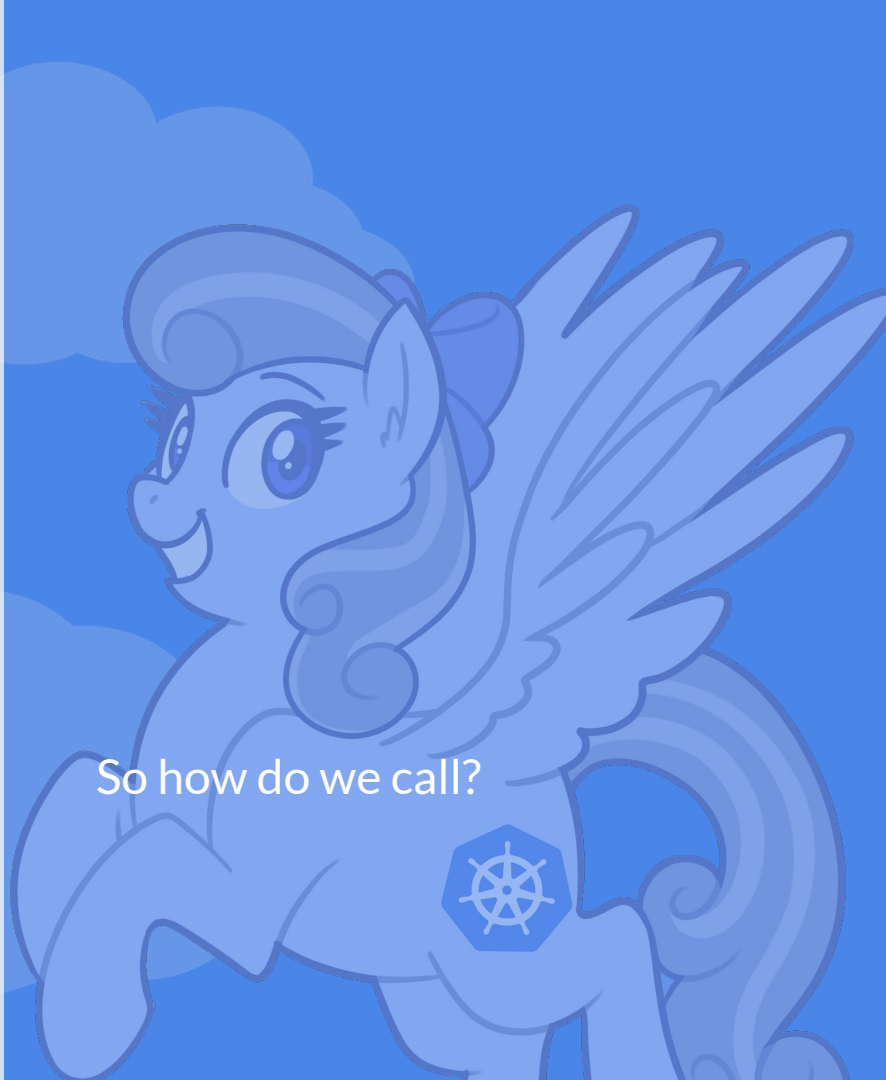
Success! Someone did the hard part for us!

Why not Multicorn?

```
// #include <stdio.h>
// #include <errno.h>
import "C"
```

```
//#cgo CFLAGS: -I/usr/include/postgresql/9.6/server
-I/usr/include/postgresql/internal
//#cgo LDFLAGS: -Wl,-unresolved-symbols=ignore-all
//#include "postgres.h"
//#include "funcapi.h"
<snip>
Import “C”
```

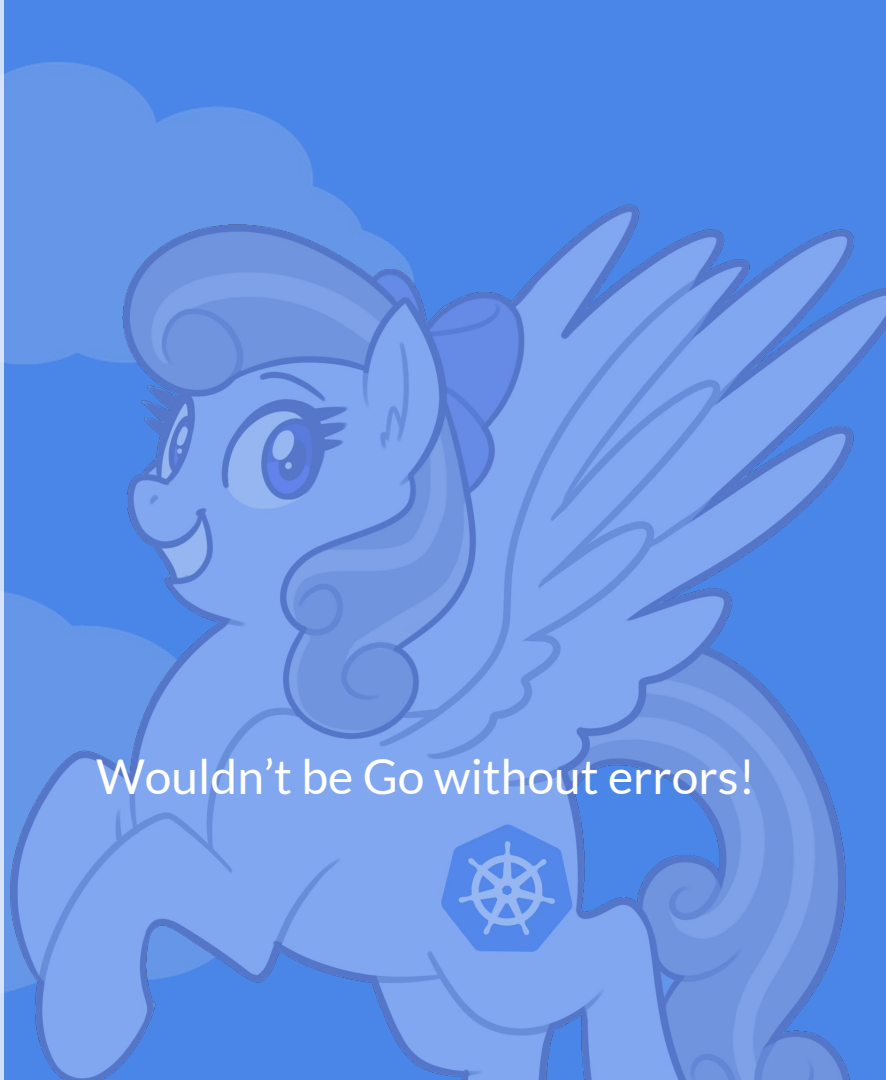
```
// Property adds a key-value property to results of EXPLAIN query.
func (e Explainer) Property(k, v string) {
        C.ExplainPropertyText(C.CString(k), C.CString(v), e.es)
}
```



```
//static Datum cStringGetDatum(const char *str) {
//   PG_RETURN_TEXT_P(CStringGetTextDatum(str));
//}
```

```
type Table interface {
    Stats(opts *Options) TableStats
    Scan(rel *Relation, opts *Options) Iterator
}

type Iterator interface {
    Next() []interface{}
    Reset()
    Close() error
}
```

```
type Table interface {
    Stats(opts *Options) (TableStats, error)
    Scan(rel *Relation, opts *Options) (Iterator, error)
}

type Iterator interface {
    Next() ([]interface{}, error)
    Reset()
    Close() error
}
```



```
CREATE SERVER IF NOT EXISTS kind
  FOREIGN DATA WRAPPER k8s_fdw
  OPTIONS (kubeconfig '/kubeconfig');

CREATE FOREIGN TABLE IF NOT EXISTS pods (
  name      text OPTIONS (alias 'metadata.name')
, namespace text OPTIONS (alias 'metadata.namespace')
)
  SERVER kind
  OPTIONS (
    namespace 'kube-system'
  , apiVersion 'v1'
  , kind 'Pod'
  );
```

```
CREATE SERVER IF NOT EXISTS kind
  FOREIGN DATA WRAPPER k8s_fdw
  OPTIONS (kubeconfig '/kubeconfig');

CREATE FOREIGN TABLE IF NOT EXISTS pods (
  name      text OPTIONS (alias 'metadata.name')
, namespace text OPTIONS (alias 'metadata.namespace')
)
  SERVER kind
  OPTIONS (
    namespace 'kube-system'
  , apiVersion 'v1'
  , kind 'Pod'
  );
```

```
$ psql --user postgres < test.sql
CREATE EXTENSION
CREATE SERVER
CREATE FOREIGN TABLE
```

```
$ psql --user postgres < test.sql
CREATE EXTENSION
CREATE SERVER
CREATE FOREIGN TABLE
ERROR:  couldn't get kubeconfig: couldn't get resource mapper:
could not get api group resources: Get
https://ec2-54-205-250-176.compute-1.amazonaws.com:6443/api?ti
meout=32s: net/http: invalid header field value "postgres:
postgres postgres [local]
SELECT\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00/v0.0.0
(linux/amd64) kubernetes/$Format" for key User-Agent
```

```
$ psql --user postgres < test.sql
CREATE EXTENSION
CREATE SERVER
CREATE FOREIGN TABLE
              name               |  namespace
---------------------------------+-------------
 coredns-6f685fffbf-7xrtp        | kube-system
 coredns-6f685fffbf-nzfn7        | kube-system
 etcd-master                     | kube-system
 kube-apiserver-master           | kube-system
 kube-controller-manager-master | kube-system
 kube-proxy-lk2mq                | kube-system
 kube-scheduler-master           | kube-system
```

That's boring. What else can you do?

How about JSONPATH?

text	the plain text	kind is {.kind}	kind is List
@	the current object	{@}	the same as input
. or []	child operator	{.kind} or {['kind']}	List
..	recursive descent	{..name}	127.0.0.1 127.0.0.2 myself e2e
*	wildcard. Get all objects	{.items[*].metadata.name}	[127.0.0.1 127.0.0.2]
[start:end :step]	subscript operator	{.users[0].name}	myself
[,]	union operator	{.items[*]['metadata.name', 'status.capacity']}	127.0.0.1 127.0.0.2 map[cpu:4] map[cpu:8]
?()	filter	{.users[?(@.name=="e2e")].user.password}	secret
range, end	iterate list	{range .items[*]}[{.metadata.name}, {.status.capacity}] {end}	[127.0.0.1, map[cpu:4]] [127.0.0.2, map[cpu:8]]
"	quote interpreted string	{range .items[*]}{.metadata.name}{'\t'}{end}	127.0.0.1 127.0.0.2

text	the plain text	kind is {.kind}
. or []	child operator	{.kind} or {['kind']}
[start:end :step]	subscript operator	{.users[0].name}

Why JSONPath?

Why JSONPath?

```
$ kubectl get pods -o=jsonpath='{@}'
$ kubectl get pods -o=jsonpath='{.items[0]}'
$ kubectl get pods -o=jsonpath='{.items[0].metadata.name}'
$ kubectl get pods -o=jsonpath='{range.items[*]}{.metadata.name}{"\t"}{.status.startTime}{"\n"}{end}'
```

```
CREATE FOREIGN TABLE IF NOT EXISTS pods (
  name      text OPTIONS (alias 'metadata.name')
, namespace text OPTIONS (alias 'metadata.namespace')
, container text OPTIONS (alias '{.spec.containers[0].image}')
)
  SERVER kind
  OPTIONS (
    namespace 'kube-system'
  , apiVersion 'v1'
  , kind 'Pod'
  );
```

```
SELECT * from PODS;
```

name	namespace	container
coredns-6f685fffbf-7xrtp	kube-system	k8s.gcr.io/coredns:1.2.6
coredns-6f685fffbf-nzfn7	kube-system	k8s.gcr.io/coredns:1.2.6
etcd-master	kube-system	k8s.gcr.io/etcd:3.2.24
kube-apiserver-master	kube-system	k8s.gcr.io/kube-apiserver:v1.12.1
kube-controller-manager-master	kube-system	k8s.gcr.io/kube-controller-manager:v1.12.1
kube-proxy-lk2mq	kube-system	k8s.gcr.io/kube-proxy:v1.12.1
kube-scheduler-master	kube-system	k8s.gcr.io/kube-scheduler:v1.12.1

That's fine and dandy, but not everything's a string!

`Labels`

object

Map of string keys and values that can be used to organize and categorize (scope and select) objects. May match selectors of replication controllers and services.

How about a map?

We’re gonna need a ~~bigger boat~~ Document Store

- JSON-based

We’re gonna need a ~~bigger boat~~ Document Store

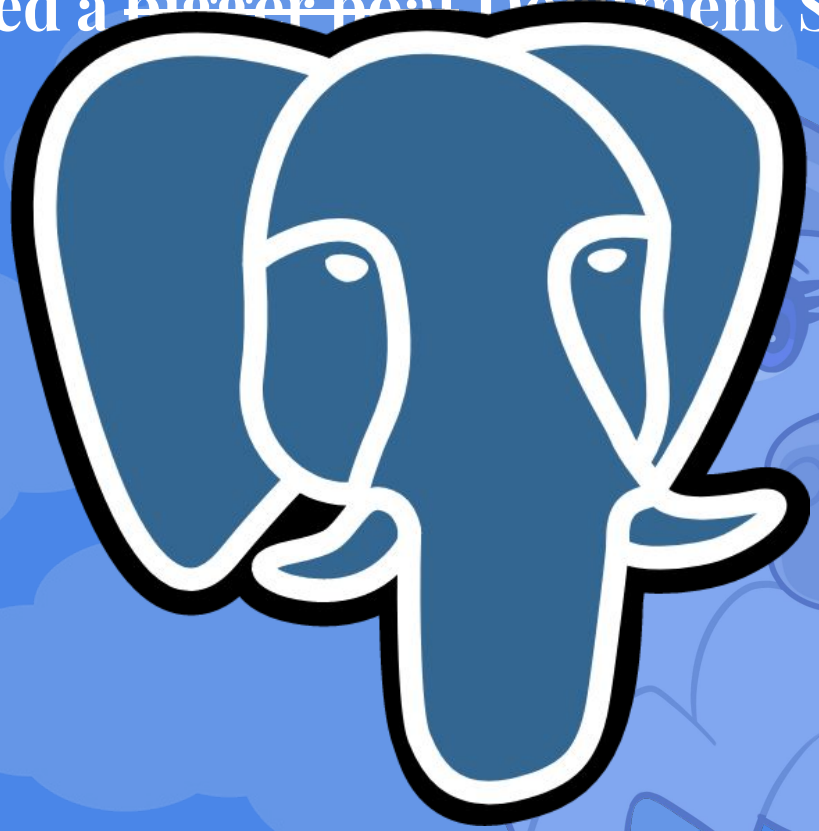
- JSON-based
- Sparse Indexes

We’re gonna need a ~~bigger boat~~ Document Store

- JSON-based
- Sparse Indexes
- Lighting Fast

We're gonna need a ~~bigger boat~~ Document Store

- JSON-based
- Sparse Indexes
- Lighting Fast



```
CREATE FOREIGN TABLE IF NOT EXISTS pods (
  name      text OPTIONS (alias 'metadata.name')
, namespace text OPTIONS (alias 'metadata.namespace')
, container text OPTIONS (alias '{.spec.containers[0].image}')
, labels   jsonb OPTIONS (alias 'metadata.labels')
)
  SERVER kind
  OPTIONS (
    namespace 'kube-system'
  , apiVersion 'v1'
  , kind 'Pod'
  );
```

```
SELECT name, labels FROM pods;
```

```
              name              |                                      labels
--------------------------------+--------------------------------------------------------------------------------------------------
 coredns-6f685fffbf-7xrtp       | {"k8s-app": "kube-dns", "pod-template-hash": "6f685fffbf"}
 coredns-6f685fffbf-nzfn7       | {"k8s-app": "kube-dns", "pod-template-hash": "6f685fffbf"}
 etcd-master                    | {"tier": "control-plane", "component": "etcd"}
 kube-apiserver-master          | {"tier": "control-plane", "component": "kube-apiserver"}
 kube-controller-manager-master | {"tier": "control-plane", "component": "kube-controller-manager"}
 kube-proxy-lk2mq               | {"k8s-app": "kube-proxy", "pod-template-generation": "1", "controller-revision-hash": "6cbfff58bb"}
 kube-scheduler-master          | {"tier": "control-plane", "component": "kube-scheduler"}
(7 rows)
```

```
SELECT name, container, labels->'component' AS component FROM pods;
```

```
             name              |                  container                  |         component
-------------------------------+---------------------------------------------+---------------------------
 coredns-6f685fffbf-7xrtp      | k8s.gcr.io/coredns:1.2.6                    |
 coredns-6f685fffbf-nzfn7      | k8s.gcr.io/coredns:1.2.6                    |
 etcd-master                   | k8s.gcr.io/etcd:3.2.24                      | "etcd"
 kube-apiserver-master         | k8s.gcr.io/kube-apiserver:v1.12.1           | "kube-apiserver"
 kube-controller-manager-master | k8s.gcr.io/kube-controller-manager:v1.12.1 | "kube-controller-manager"
 kube-proxy-lk2mq              | k8s.gcr.io/kube-proxy:v1.12.1               |
 kube-scheduler-master         | k8s.gcr.io/kube-scheduler:v1.12.1           | "kube-scheduler"
```

And for my final trick:

```
SELECT "deployments"."name" AS deployment_name
     , "replica_sets"."name" as replica_name
     , "pods"."name" AS pod_name
  FROM deployments
  JOIN replica_sets on "replica_sets"."name" LIKE "deployments"."name" || '-%'
  JOIN pods on "pods"."name" LIKE "replica_sets"."name" || '-%';
```

```
 deployment_name |    replica_name    |         pod_name
-----------------+--------------------+--------------------------
 coredns         | coredns-6f685fffbf | coredns-6f685fffbf-7xrtp
 coredns         | coredns-6f685fffbf | coredns-6f685fffbf-nzfn7
```

What *doesn't* work?

- Column types mostly ignored

What *doesn't* work?

- Column types mostly ignored
- If it's a not a number, string, or map...
  ...just kind of give up

What *doesn't* work?

- Column types mostly ignored
- If it's a not a number, string, or map...
  ...just kind of give up
- The codebase not being a not of spaghetti

Lessons from Production

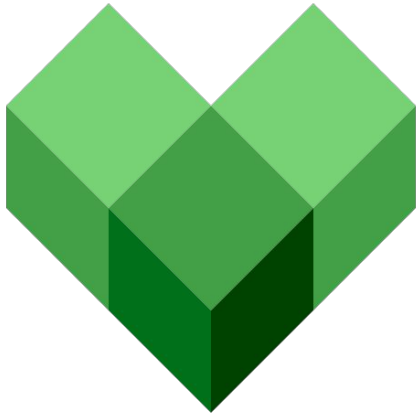
Lessons from ~~Production~~ The week I wrote this

Debugging is Hard!

- No log files
- Lots of Segfaults
- Postgres codebase large and intimidating

Bazel + Docker

- Bazel builds C + Go
- Docker container runs postgres



```
bazel run //:k8s_fdw_image

docker run \
 -v /tmp/config:/kubeconfig \
 --rm --name=k8s_fdw \
 bazel:k8s_fdw_image
```

Questions? Concerns?

Come find me! I'm the one with pink hair!

Twitter: @stillinbeta

Github: github.com/liztio/k8s_fdw

K8s.slack.io: @liz