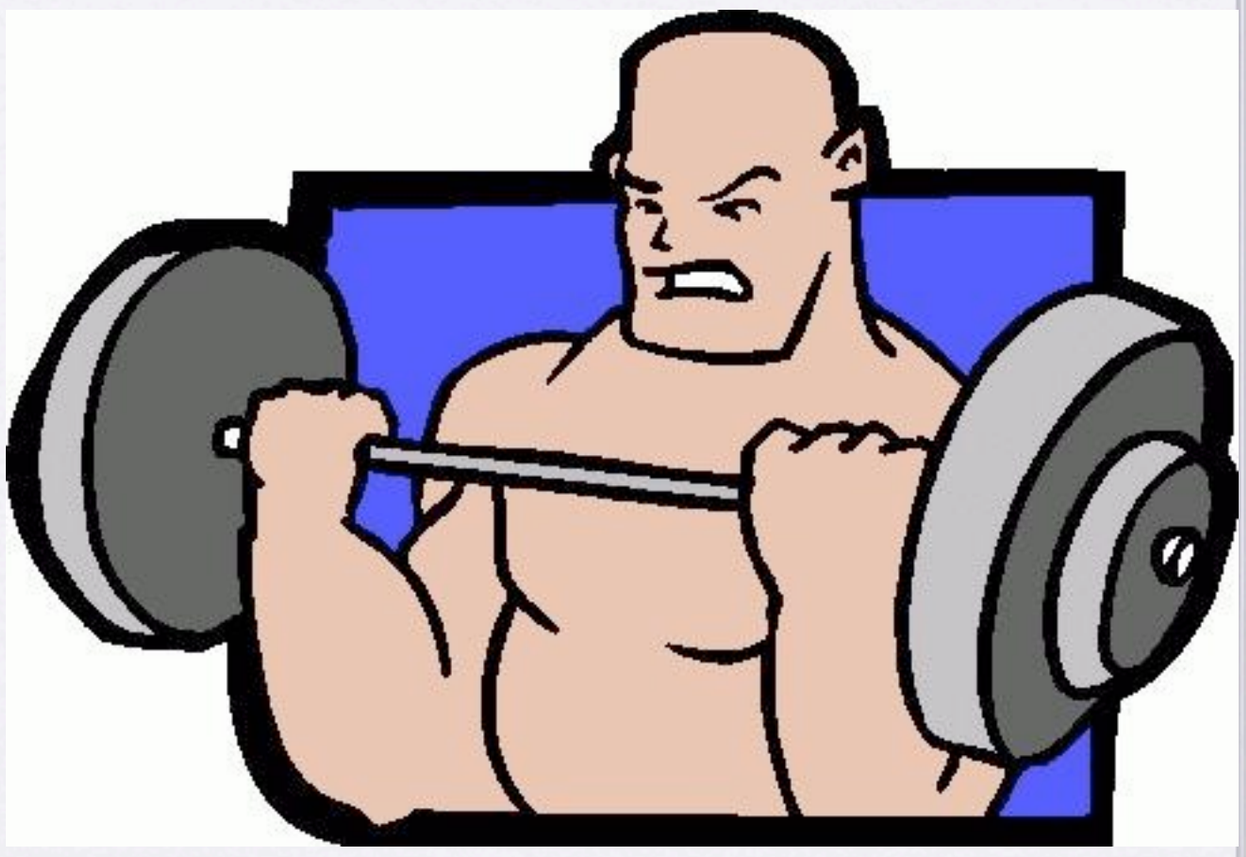
https://github.com/revmischa/pgnotify-demos

hey

User-Defined Placeholder Text

scaling with django

- Double-click to edit
- The cloud
- DevOps
- Dvorak



Message Queueing

- Ready-made components
- Infrastructure

Message Queueing

- Scalable
- Highly available
- Routing
- More event-based programming

Message Queueing

- Connect to PostgreSQL triggers
- Well-supported with HTML5 WebSockets and even MXHR
- Push notifications are cool and useful (demo)

ZeroMQ

- Fantastic library
- Supports TCP, UNIX IPC
- A bit too low-level

ZeroMQ

- Created my own PubSub server - ZeroMQ::PubSub
- Worked OK
- Not scalable
- SPOF
- Not PostgreSQL

PostgreSQL

- PubSub server built in!
- SPOF... so?
- PG SPOF > ZeroMQ::PubSub SPOF

LISTEN & NOTIFY

- # LISTEN foo;
- # NOTIFY foo, ‘hello foo!’;
- > Asynchronous notification "foo" with payload "hello, foo!" received from server process with PID 11095.
- Demo

C - libpq

```
while (notify = PQnotifies(conn)) {

    printf("NOTIFY of '%s' received from PID %d: '%s'\n",
            notify->relname, notify->be_pid, notify->extra);

    PQfreemem(notify);
}
```

Perl

- CPAN(minus)
- Prefork, async, whatever
- DBIx::Class
- Catalyst

Perl

- DBI + DBI::Pg
- AnyEvent
- AnyEvent::Pg
- AnyMQ
- AnyMQ::Pg

PSGI + Plack

- Catalyst
- Starman
- Feersum
- Others: Dancer, Mojolicious, Poet, Starlet...

Long-Polling

```
$ curl -v http://localhost:4000/_hippie/mxhr/mychannel
> GET /_hippie/mxhr/mychannel HTTP/1.1
> Host: localhost:4000
>
< HTTP/1.1 200 OK
< Content-Type: multipart/mixed; boundary="LjxCy7H9"
< Transfer-Encoding: chunked
<
--LjxCy7H9
Content-Type: application/json

{"client_id":"123","type":"hippie.pipe.set_client_id"}

--LjxCy7H9
Content-Type: application/json

{"type":"mychannel"}
```

WebSocket

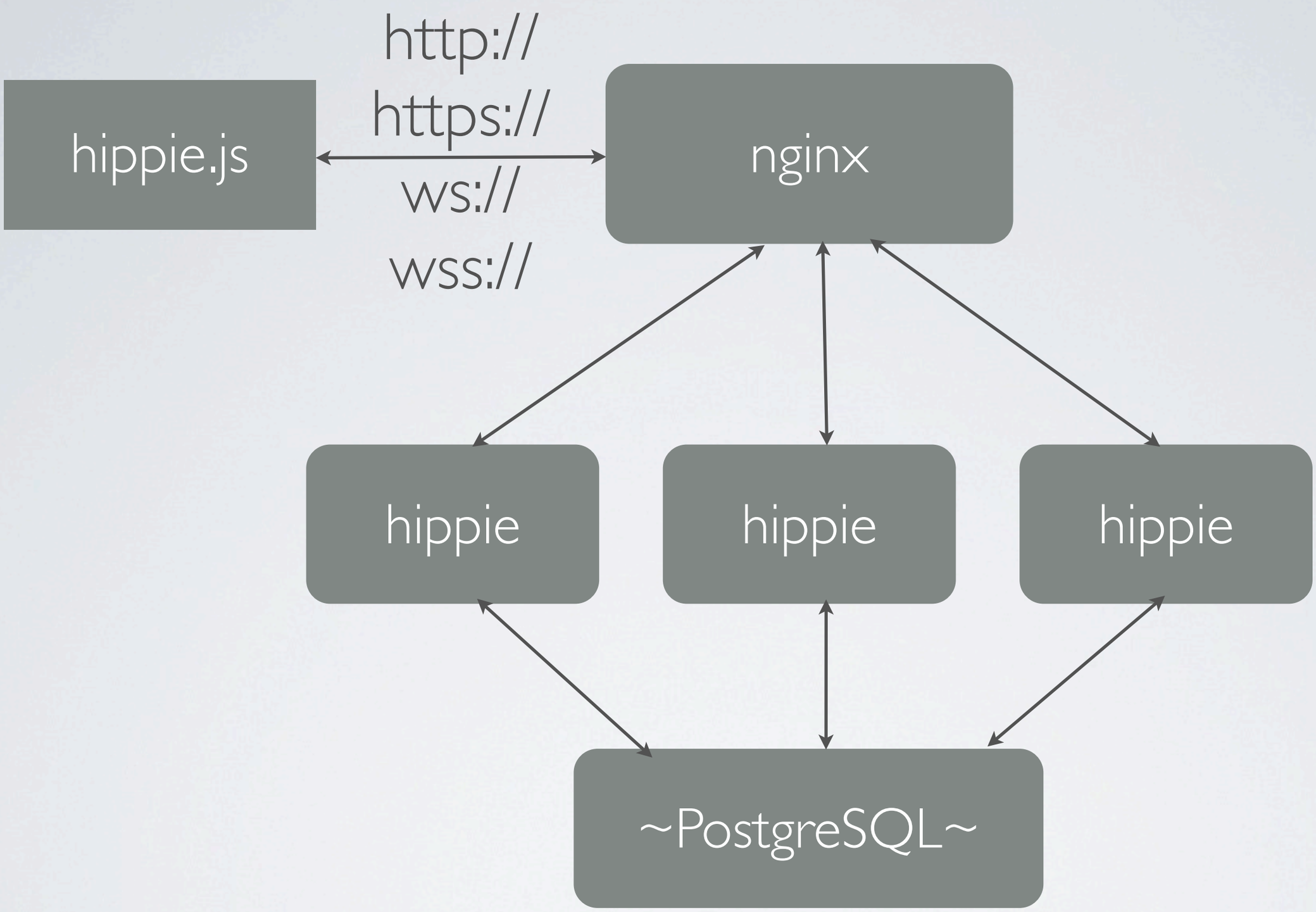
- Much cleaner than XHR
- Load balancing is trickier

Hippie::Pipe

- Supports WebSocket, XHR, Polling
- Uses AnyMQ
- AnyMQ::Pg https://github.com/revmischa/anymq-pg
- Web::Hippie::PubSub

Load Balancing/HA

- nginx for everything
- tcp_proxy


http://
https://
ws://
wss://
hippie.js
nginx
hippie
hippie
hippie
~PostgreSQL~

Location Update

```
CREATE EXTENSION postgis;

CREATE TABLE person (
    id serial PRIMARY KEY,
    name TEXT,
    loc GEOGRAPHY(POINT, 4326)
);
```

Location Update

```
...
PERFORM pg_notify('person_updated',
      '{"id": '
      || CAST(NEW.id AS text)
      || ', "loc": '
      || ST_AsGeoJSON(NEW.loc)
      || '}'
);
...
```

Location Update

```
CREATE TRIGGER person_update_notify
  AFTER UPDATE OR INSERT
  ON person FOR EACH ROW EXECUTE
  PROCEDURE update_person_table();
```

Location Update

```
INSERT INTO person (name,loc)
VALUES ('Mischa', ST_GeographyFromText(
'POINT(-122.... 37...)'));
```

```
UPDATE person
SET loc=ST_GeographyFromText(
'POINT(-122... 37...)');
```

Demo...

Other Applications

- Cache invalidation triggers
- Stuff