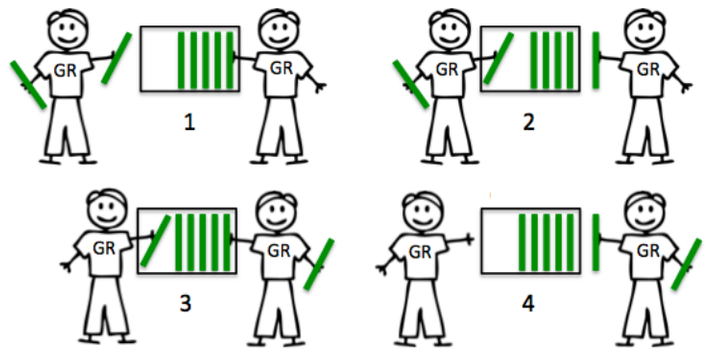
# Django Channels

Greg Schafer



# Outline


- Overview
- Context: Django history
- Context: WebSockets history/ecosystem
- Goals, Architecture, Considerations
- Demo + Code

# Overview

- Architecture change to Django
- Motivation = solution for websockets
- Andrew Godwin
- Separate library

- other protocols like http/2 (server push) and esp long-lived connection ones like websockets
- https://github.com/andrewgodwin/channels
    - While the goal is to merge this into Django proper, prototyping and iterating outside is much faster and more accessible for everyone involved
-

# History of Django

| Version | Date | Notes |
|---|---|---|
| 0.90[32] | 16 Nov 2005 | |
| 0.91[33] | 11 Jan 2006 | "new-admin" |
| 0.95[34] | 29 Jul 2006 | "magic removal" |
| 0.96[35] | 23 Mar 2007 | "newforms", testing tools |
| 1.0[36] | 3 Sep 2008 | API stability, decoupled admin, unicode |
| 1.1[37] | 29 Jul 2009 | Aggregates, transaction based tests |
| 1.2[38] | 17 May 2010 | Multiple db connections, CSRF, model validation |
| 1.3[39] | 23 Mar 2011 | Class based views, staticfiles |
| 1.4[40] | 23 Mar 2012 | Timezones, in browser testing, app templates. [41] |
| 1.5[42] | 26 Feb 2013 | Python 3 Support, configurable user model |
| 1.6[43] | 6 Nov 2013 | Dedicated to Malcolm Tredinnick, db transaction management, connection pooling. |
| 1.7[44] | 2 Sep 2014 | Migrations, application loading and configuration. |
| 1.8[45] | 1 Apr 2015 | Native support for multiple template engines. **Long-term support release, supported until at least April 2018** |
| 1.9[46] | 1 Dec 2015 | Automatic password validation. New styling for admin interface. |

- https://en.wikipedia.org/wiki/Django_(web_framework)#Version_history
- was started in 2003 at Lawrence Journal-World newspaper, released publicly in July 2005
- named after european jazz guitarist Django Reinhardt

# Django Release Schedule

| Release Series | Release Date | End of mainstream support[1] | End of extended support[2] |
|---|---|---|---|
| 1.9 | December 2015 | August 2016 | April 2017 |
| 1.10 | August 2016 | April 2017 | December 2017 |
| 1.11 LTS | April 2017 | December 2017 | Until at least April 2020 |
| 2.0 | December 2017 | August 2018 | April 2019 |

- Sidenote: Django 2.0 drops Python 2 support

https://www.djangoproject.com/weblog/2015/jun/25/roadmap/

# Channels History

- Proposed June 2015 (after v1.8)
- Targeted for August 2016 (v1.10)
- MOSS grant ($150k) in Dec 2015
- Concerns (complexity, latency, flexibility)
- Withdrawn May 2016
  - Now targeting 1.11 or 2.0

- https://groups.google.com/forum/#!topic/django-developers/mt1mTNvc0Ew
- https://www.djangoproject.com/weblog/2015/dec/11/django-awarded-moss-grant/
- Mailing list devs are very well-spoken and respectful
- “Channels is at best on par with the existing available approaches and at worst adds a bunch of latency, potentially dropped messages, and new points of failure while taking up more resources and locks everyone into using Redis”
  - https://groups.google.com/forum/#!searchin/django-developers/channels/django-developers/CZPvEE0WPi4/e1tGAYbYCwAJ
  - https://groups.google.com/forum/#!searchin/django-developers/channels/django-developers/pL2PMqT8xA8/xVLG7JePAQAJ
- Withdrawn: https://groups.google.com/forum/#!searchin/django-developers/channels/django-developers/QRd4v7OErT8/oTv4APoOAwAJ
  - Now aiming for 1.11 or 2.0

# WebSockets History

- Started in 2008, standardized in 2011
- Full-duplex TCP over port 80
- Preceded by Comet (usu. long-polling)
- Low overhead
- Upgrade header

- HTTP/1.1 uses keepalive to reuse same TCP connection
  - https://en.wikipedia.org/wiki/HTTP_persistent_connection
- overhead as low as 2 bytes per frame

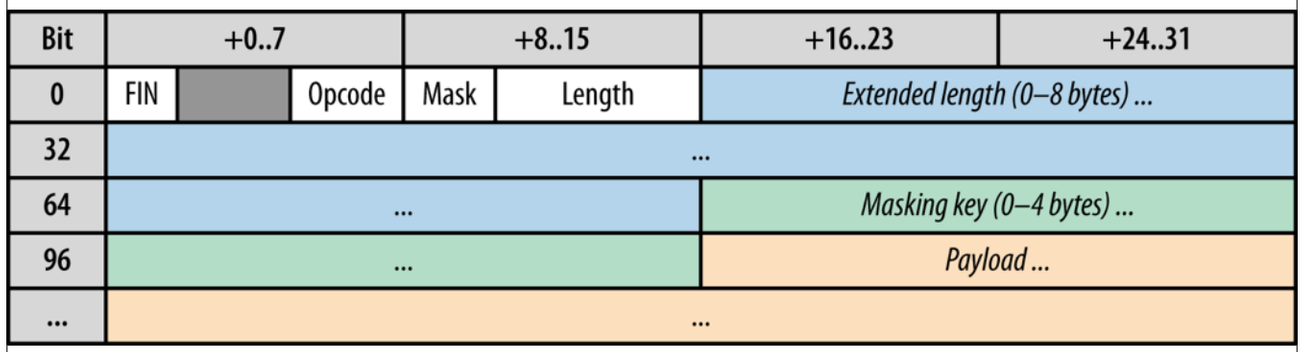
```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

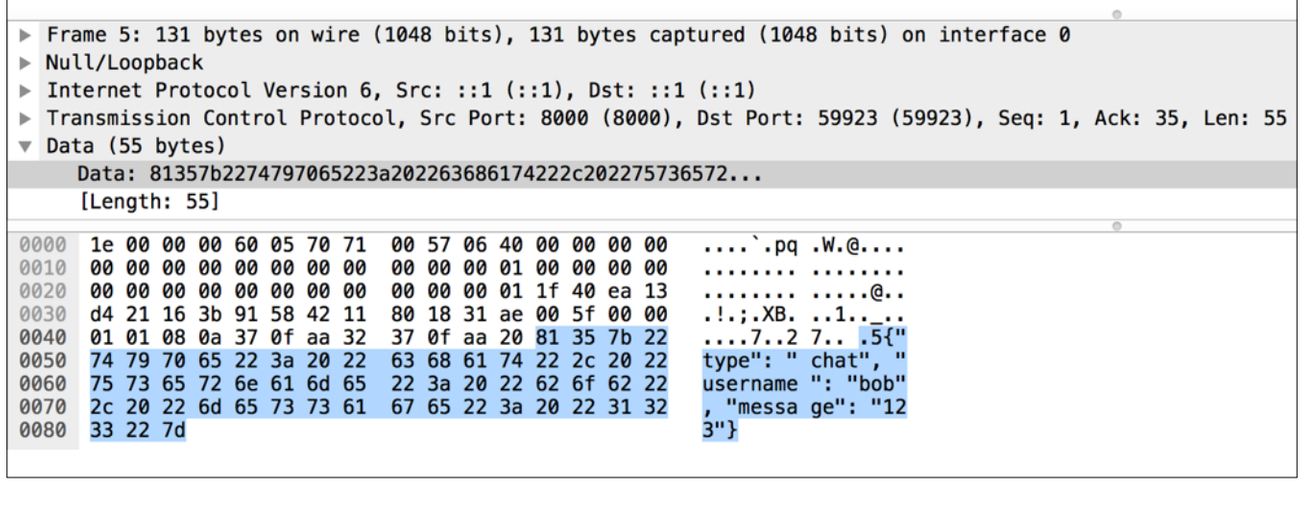
Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

- show chrome inspector

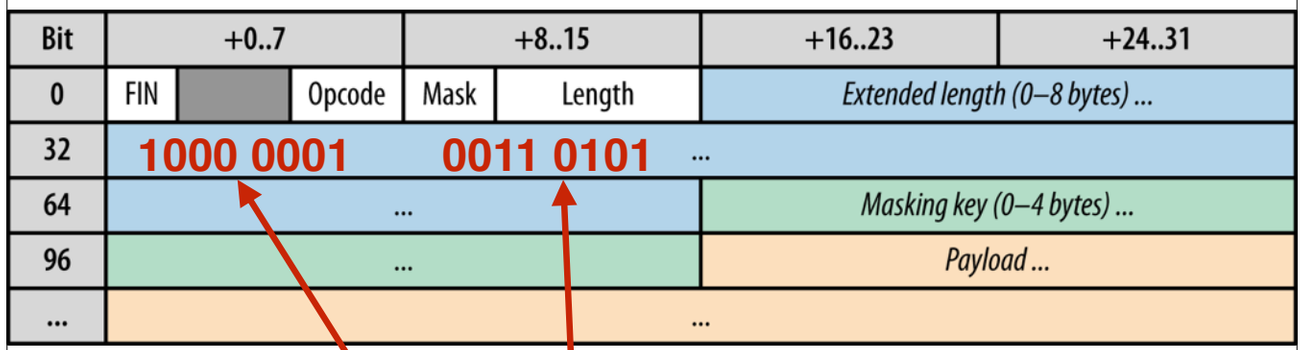
# Frames

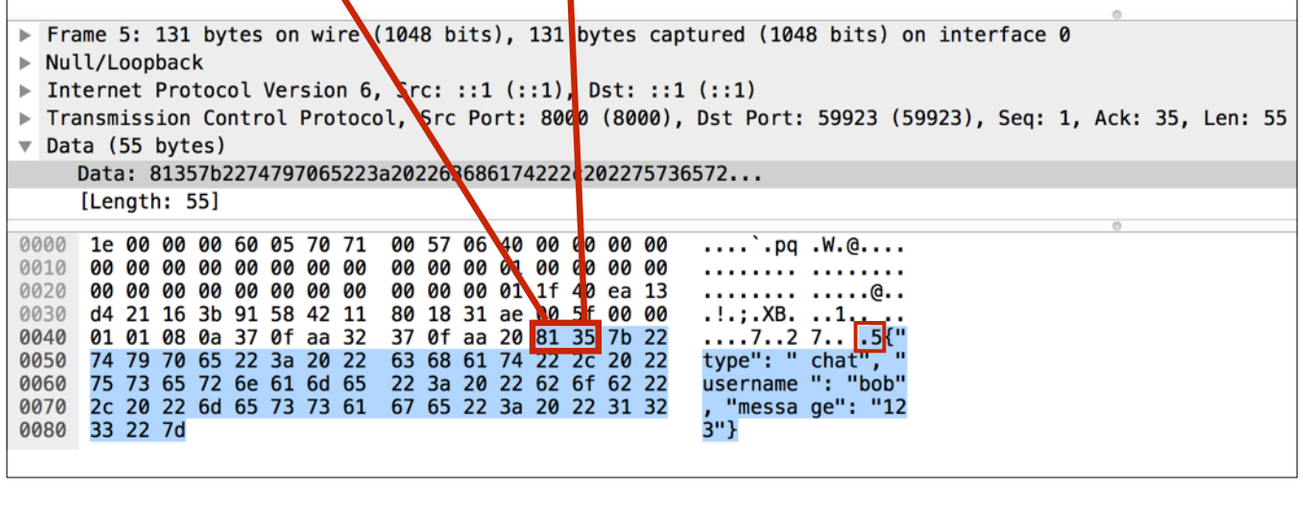



- show chrome inspector

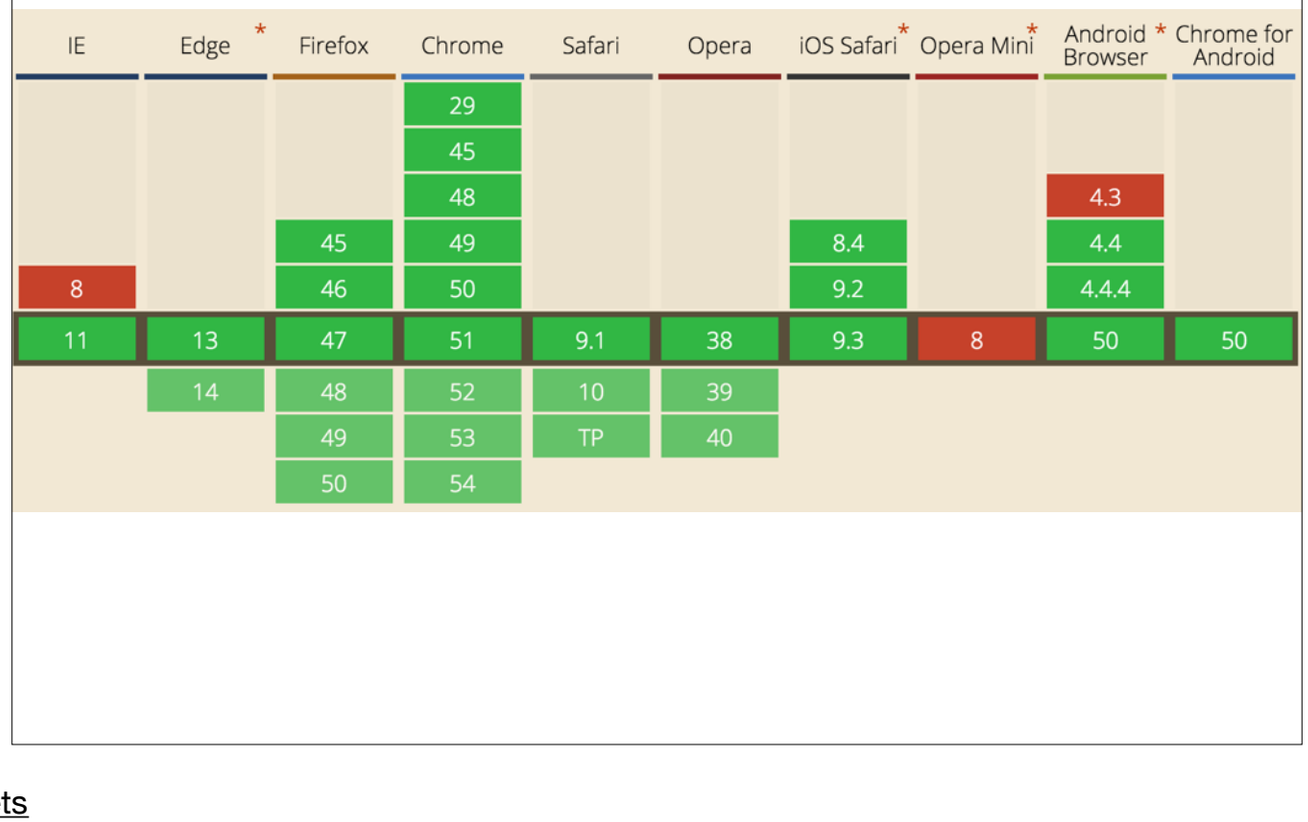
# Frames




- 81 = 1000 0001
  - FIN bit set, opcode = “text frame”
  - https://tools.ietf.org/html/rfc6455#page-65
- 35 = 0011 0101
  - no mask, length = 53 bytes
  - mask is for XORing w/ content to avoid proxy cache poisoning attacks
    - http://stackoverflow.com/a/14174429

# WebSockets Support




- http://caniuse.com/#feat=websockets

# WebSockets Ecosystem

- Rails
  - Rails 5 (June 22) introduced ActionCable
  - Faye, em-websocket, many gems
- Node (lots)
  - Faye, socket.io, Primus (abstraction layer)
- Python
  - Autobahn, crossbar.io

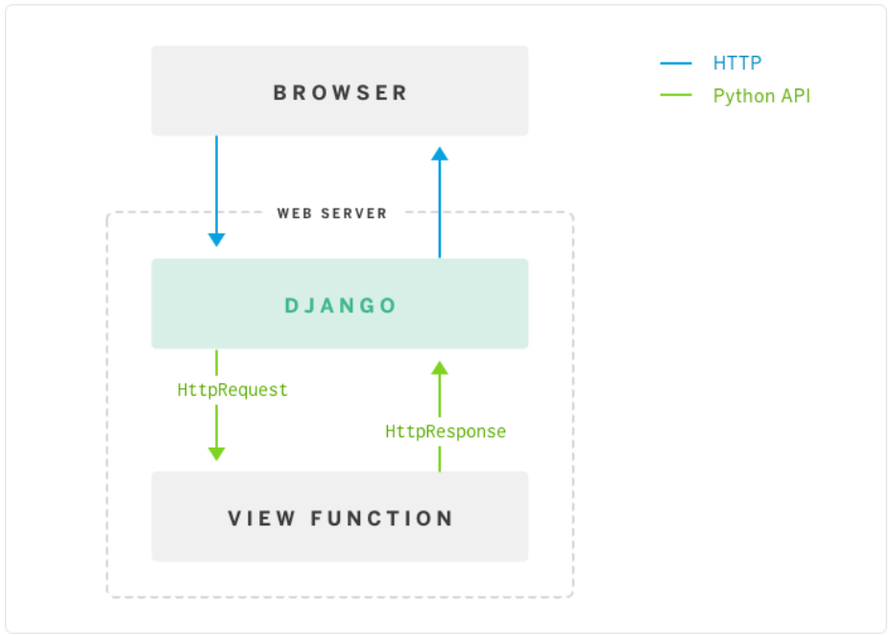
- ActionCable
  - Uses redis pubsub
  - Must be on same domain for cookie authentication
  - https://blog.heroku.com/real_time_rails_implementing_websockets_in_rails_5_with_action_cable
- Many stacks run http server and then a separate node server for websockets and intercommunicate via http API or some kind of pubsub (e.g. redis)
- Pusher.com is paid messaging SaaS
- Faye architecture: https://faye.jcoglan.com/architecture.html
- Ruby: https://www.sitepoint.com/websockets-in-the-ruby-ecosystem/
- Node: http://stackoverflow.com/a/16393046
- https://www.fullstackpython.com/websockets.html

# Goals of Channels

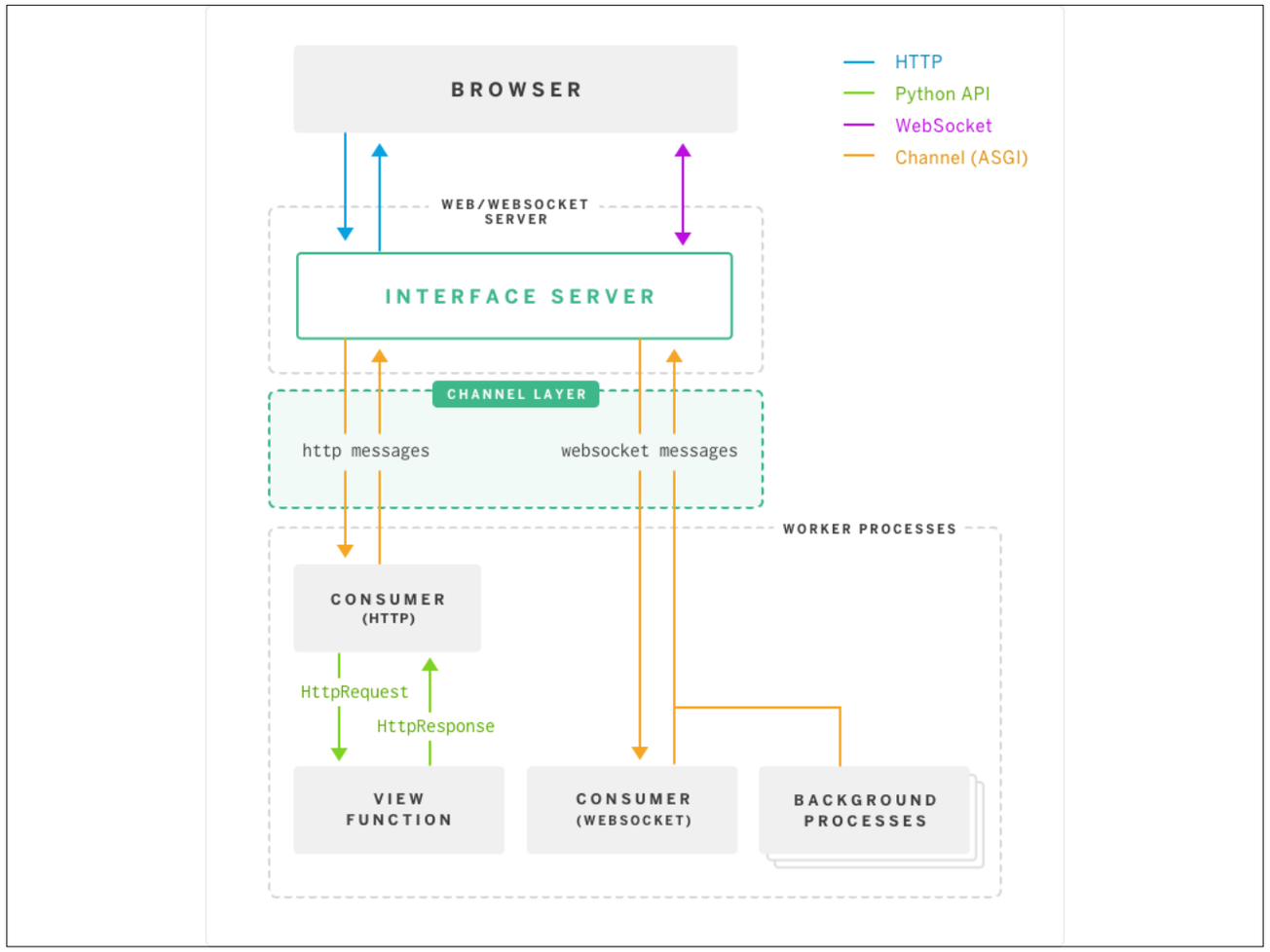
- Backwards compatible
- Simple (workers run synchronous code)
- Low latency + high throughput
  - Tradeoff = sacrifice guaranteed delivery
- Network transparent

- https://channels.readthedocs.io/en/latest/inshort.html
- https://channels.readthedocs.io/en/latest/concepts.html

# Architecture Change




- https://blog.heroku.com/archives/2016/3/17/in_deep_with_django_channels_the_future_of_real_time_apps_in_django
-




- channel layer = redis
  - https://channels.readthedocs.io/en/latest/backends.html
- consumers can be
  - http, sms, smtp, http/2, etc.
  - websocket handling, background tasks (a la celery, though beware considerations)
- changes django to be “event oriented”
- any consumer can process a message
- interface server is async (asyncio and Twisted), could be written in node or anything if it conforms to serialization standard
  - https://channels.readthedocs.io/en/latest/faqs.html#why-aren-t-you-using-node-go-etc-to-proxy-to-django

# Spec

- Channel layer interface (asgi_redis)
  - `send(channel, message)`
  - `receive_many(channels, block=False)`
- http://channels.readthedocs.io/en/latest/asgi.html#specification-details

- send puts message in redis
- receive_many shuffles channels, can block on getting a message
  - called by workers
    - workers can exclude channels or listen to only specific channels (priority)
- these are implemented as redis lua scripts (atomic w/ multiple operations)

# Groups

- Channel layer interface (asgi_redis)
  - `group_add(group, channel)`
  - `group_discard(group, channel)`
  - `send_group(group, message)`

- other extensions are flush and statistics
  - https://channels.readthedocs.io/en/latest/asgi.html#extensions
-

# Considerations

- Channel = ordered, first-in first-out queue with message expiry and **at-most-once delivery** to only one listener at a time
- Celery replacement? No

- https://channels.readthedocs.io/en/latest/concepts.html
- don’t use channels for reliable background processing
    - would need to add your own reliability mechanism
- high throughput example = player movement in a game
    - lost messages is no big deal

# Demo: Chat

- http://10.1.10.242:8000/
- Kill worker, inspect redis
  - Message timeout, no delivery
- Next steps
  - Models: http://channels.readthedocs.io/en/latest/getting-started.html#models

- git tag basic-chat
- redis commands
  - flushdb, keys, get, lrange, ttl

# Demo: Drawing

- Generalized handling
- Throughput

- git stash pop
- could generalize more
    - can use separate client-side channels, pointed at diff paths
    - routing.py can route based on path or any message attribute that's a string (e.g. http.request messages have method key)
        - http://channels.readthedocs.io/en/latest/getting-started.html#routing
- improvement is to buffer drawing and send on 30fps interval

# Questions?

https://github.com/grschafer/pythonweb-meetup-channels