Django Channels - WebSockets with Django

•••

Rafael Laverde

11 February 2017
Pycon Colombia

Rafael Laverde

@rlaverde

@rafa_laverde

- Junior Developer at Spyder-IDE (Continuum Analytics)
- 3+ years experience with python (django, scientific computing...)
- Also freelance.
- Trying to create Python Tunja.
- Software Libre advocate

Festival de Cultura Libre

4 - 7 ABRIL 2017
TALLERES - CONFERENCIAS - FOROS
FESTIVAL de
CULTURA LIBRE
SOFTWARE LIBRE
HARDWARE LIBRE
SEMILLAS LIBRES
LEGISLACION COPYLEFT
ARTE LIBRE
FILOSOFIA
http://cusol.uis.edu.co/clibre2/
fb.com/cusol
@cusol_org

http://cusol.uis.edu.co

cusol@uis.edu.co

@cusol_org

facebook/cusol

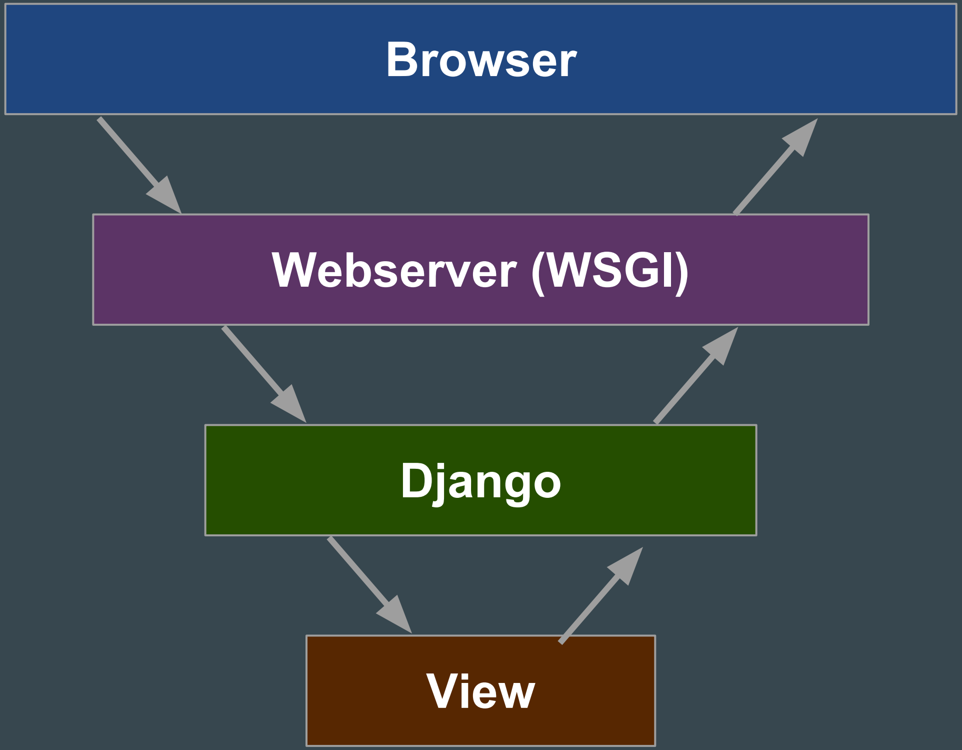
CUSOLUIS

Django, HTTP and WSGI

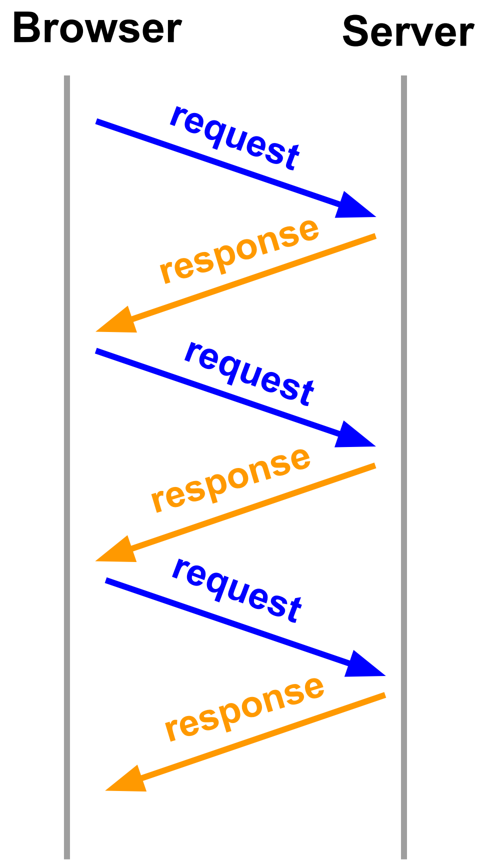
You take a request...

...and return a response

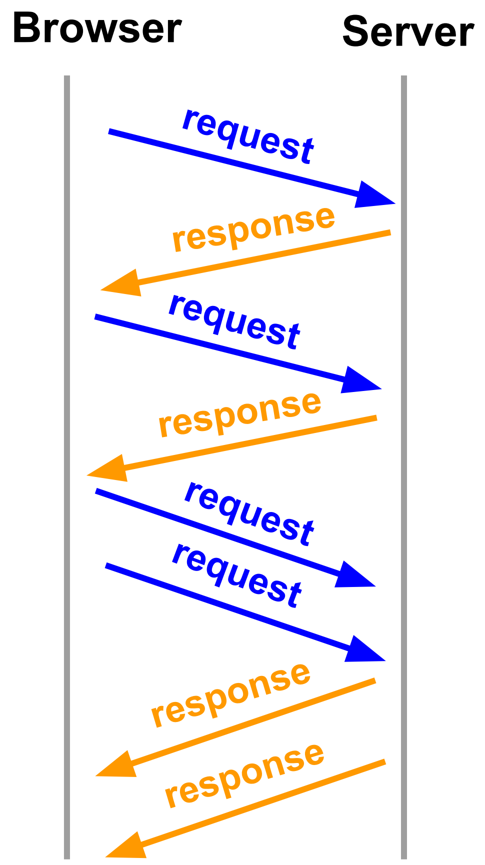
How django works



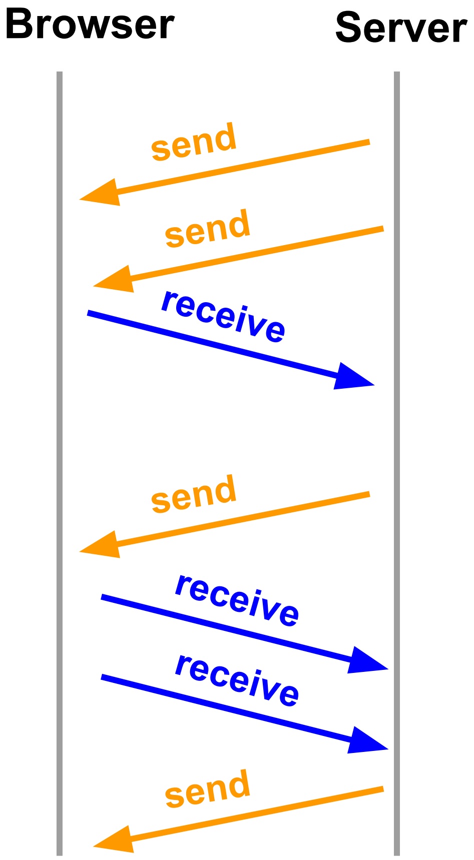
HTTP1


Browser
Server
request
response
request
response
request
response

HTTP2


Browser
Server
request
response
request
response
request
request
response
response

WebSockets


Browser
Server
send
send
receive
send
receive
receive
send

Django channels!!!

•••

Developer-friendly asynchrony for Django

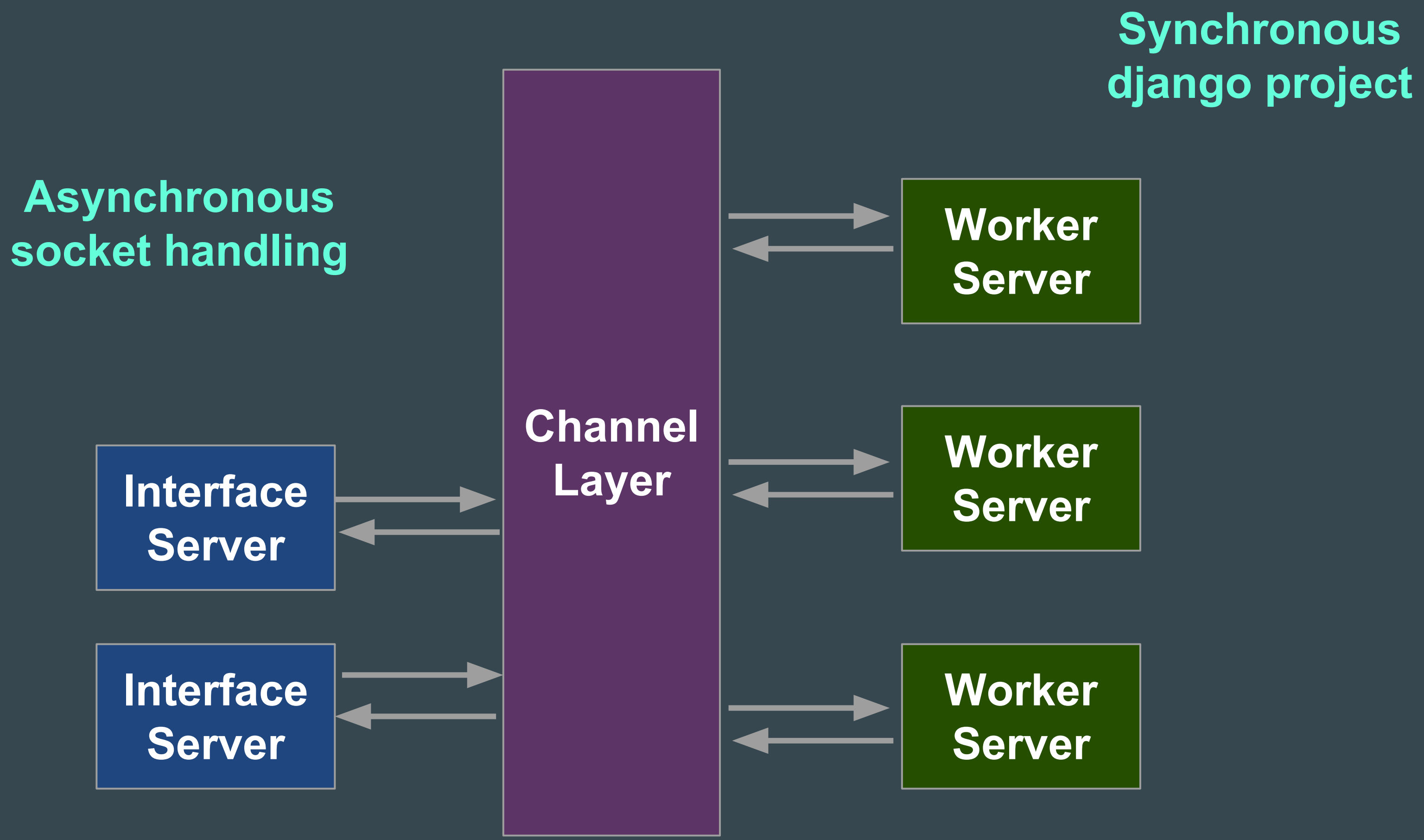
What is it?

Channels extends Django to add a new layer

Allows:

- WebSocket handling
- Background tasks

How?


Synchronous django project
Asynchronous socket handling
Channel Layer
Worker Server
Worker Server
Worker Server
Interface Server
Interface Server

How?

Change from running Django under a WSGI server, to running:

- An ASGI server, probably Daphne (**interface servers**)
- Django worker servers (**workers**)
- Something to route ASGI requests over, like Redis. (**channel backend**)

Channels Concepts

What is a channel

Ordered, FIFO queue, at-most-once delivery.

```
def my_consumer(message):
    pass

channel_routing = {
    "some-channel":
"myapp.consumers.my_consumer",
}
```

Channels

There are some useful default channels

- websocket.connect
- websocket.disconnect
- websocket.receive
- websocket.send

- http.request
- http.disconnect

Groups

Set of channels you broadcast to

```
Group("some_group").add(message.reply_channel)

Group("some_group").send({
        "text": json.dumps({
            "id": instance.id,
            "content": instance.content
        })
```

Install

```
pip install -U channels
```

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    ...
    'channels',
)
```

How a channels project looks like

```
liveblog/
    liveblog/
        settings.py
        ....
        asgi..p
        wsgi.py
        routing.py
posts/
    consumers.py
    models.py
    ...
```

routing.py

```
route("websocket.connect", connect_blog,
        path=r'^/liveblog/(?P<slug>[^/]+)/stream/$'),


route("websocket.disconnect", disconnect_blog,
        th=r'^/liveblog/(?P<slug>[^/]+)/stream/$'),


route("websocket.receive", save_post,
        path=r'^/liveblog/(?P<slug>[^/]+)/stream/$'),
```

Examples

Liveblog

People open a WebSocket when they open the page

Their Websocket is added to a group

When a Blog post is saved, the post pis send to the group

Liveblog

Their Websocket is added to a group

```
def connect_blog(message):
    group("liveblog").add(message.reply_channel)
```

When a Blog post is saved, the post is send to the group

```
class Post(models.Model):
  ...
  def save(self, *args, **kwargs):
    ...
      Group(“.liveblog”.send({
        "text": json.dumps(notification),
    })
```

Chat

People can send messages, and they send to everyone connected

When people connect they join a chat group

When we receive a message we send it pto the group

Chat

When people connect they join a chat group

```
def connect_blog(message):
    group("chat").add(message.reply_channel)
```

When we receive a message we send it to the group

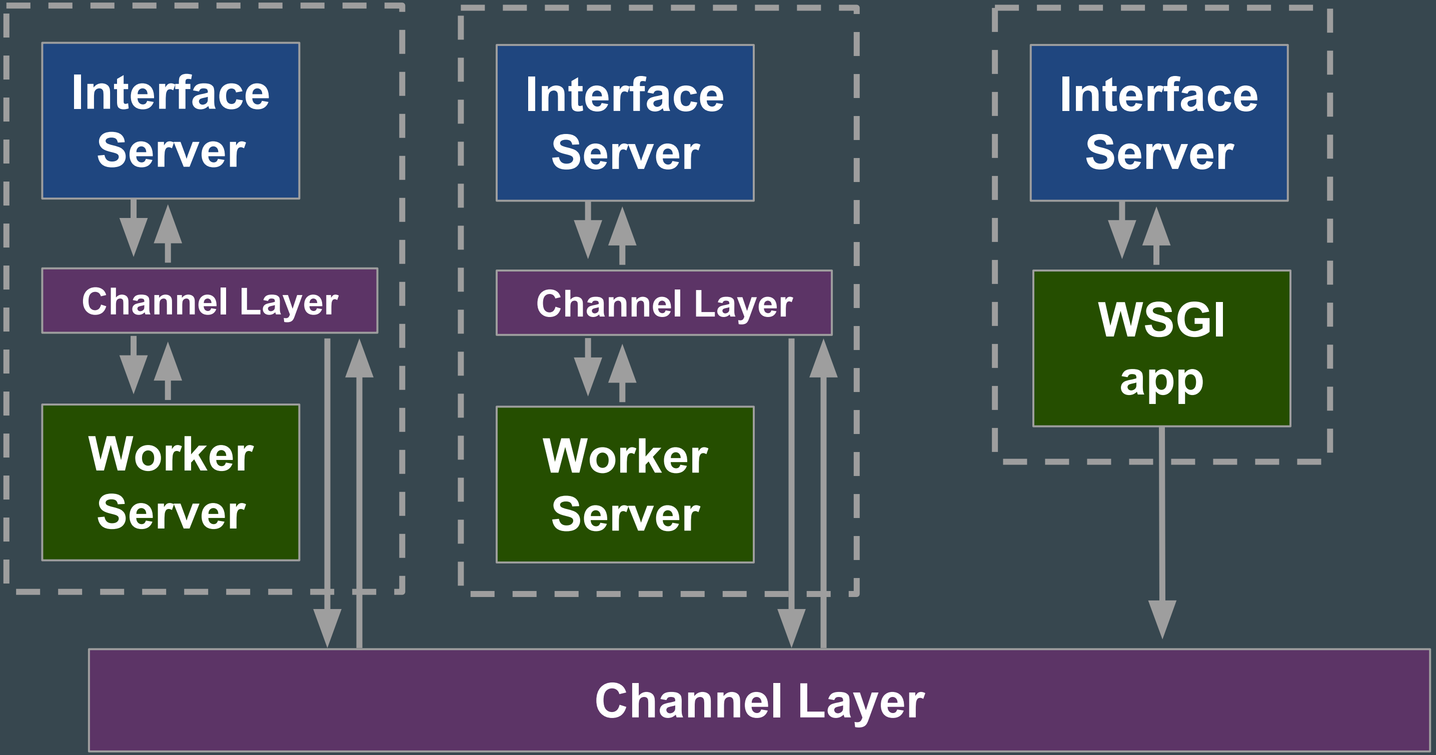
```
Group(“chat”).send({"text": json.dumps(notification),
```

Deploy!!

Architecture

- ASGI
  - Dapne
  - WSGI Adapter
- Backend
  - Redis
  - Posix
  - In-memory
- Worker
  - Django

WSGI and/or ASGI



Scalable

- Interface servers scale horizontally
- Worker server scale horizontally
- Channel layer could be the bottleneck (Sharding)

Alternatives

- websockets, Autoban (based in Asyncio, Twisted)
- Django-websockets (abandoned)
- Flask-websockets
- Tornado websocket

Channels is different!?!

Channels is different!!

Others options are a way of making a single Python process act **asynchronously**

In django-channels all the code you write for consumers runs **synchronously**.

Channels provide a **high-throughput** solution that is mostly reliable, rather than a **low-throughput** one that is nearly completely reliable.

Should I use channels?

- Scalable
- It’s becoming a mature project
- Support (an official Django project since September 2016)
- Mozilla sponsorship
- Community
- Documentation

Further Reading

https://channels.readthedocs.io

https://github.com/django/channels/

https://github.com/andrewgodwin/channels-examples

https://www.youtube.com/watch?v=2sEPipctTxw

Thank you!!

@rlaverde

@rafa_laverde