ActionCable

full-stack, real-time, publish-subscribe framework for Rails 5

Usage

- Chat
- Multiplayer Games
- Updates
- Feeds
- Collaborative Editing and Coding
- etc.

WebSocket

- TCP-based protocol
- Persistent (long-lived)
- Bi-directional (full-duplex)
- Designed to be implemented in web browsers and web servers
- Supports encrypted communication
- Introduces URI schemes: ws, wss

WebSocket: Standards

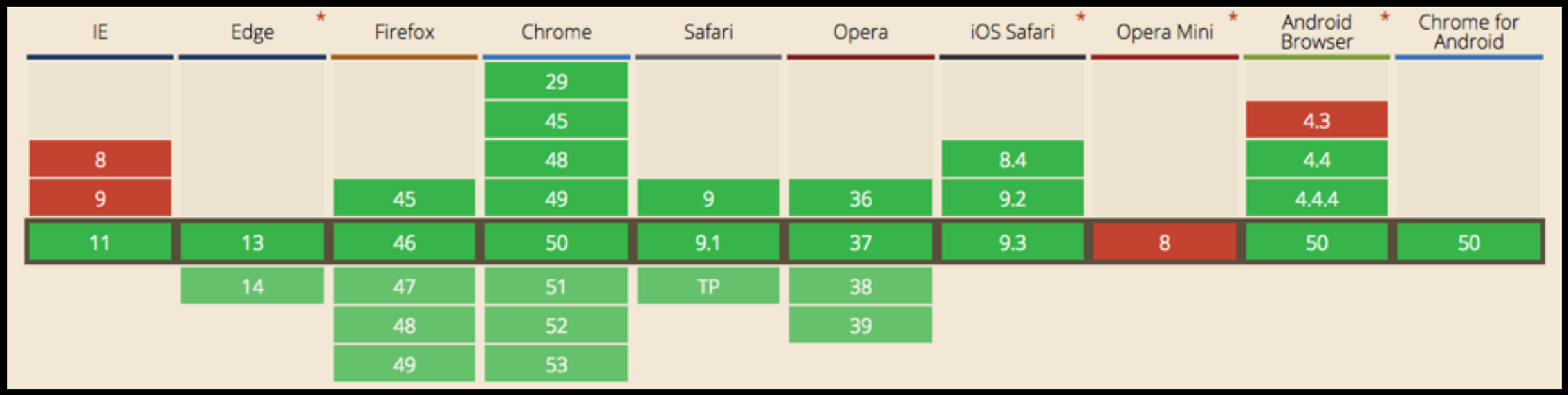
Standardised in 2011

IETF: RFC 6455

The WebSocket Protocol

W3C Web IDL

The WebSocket API



can-i-use-it

Yes!

WebSocket: Handshake

protocol switch from HTTP to WebSocket

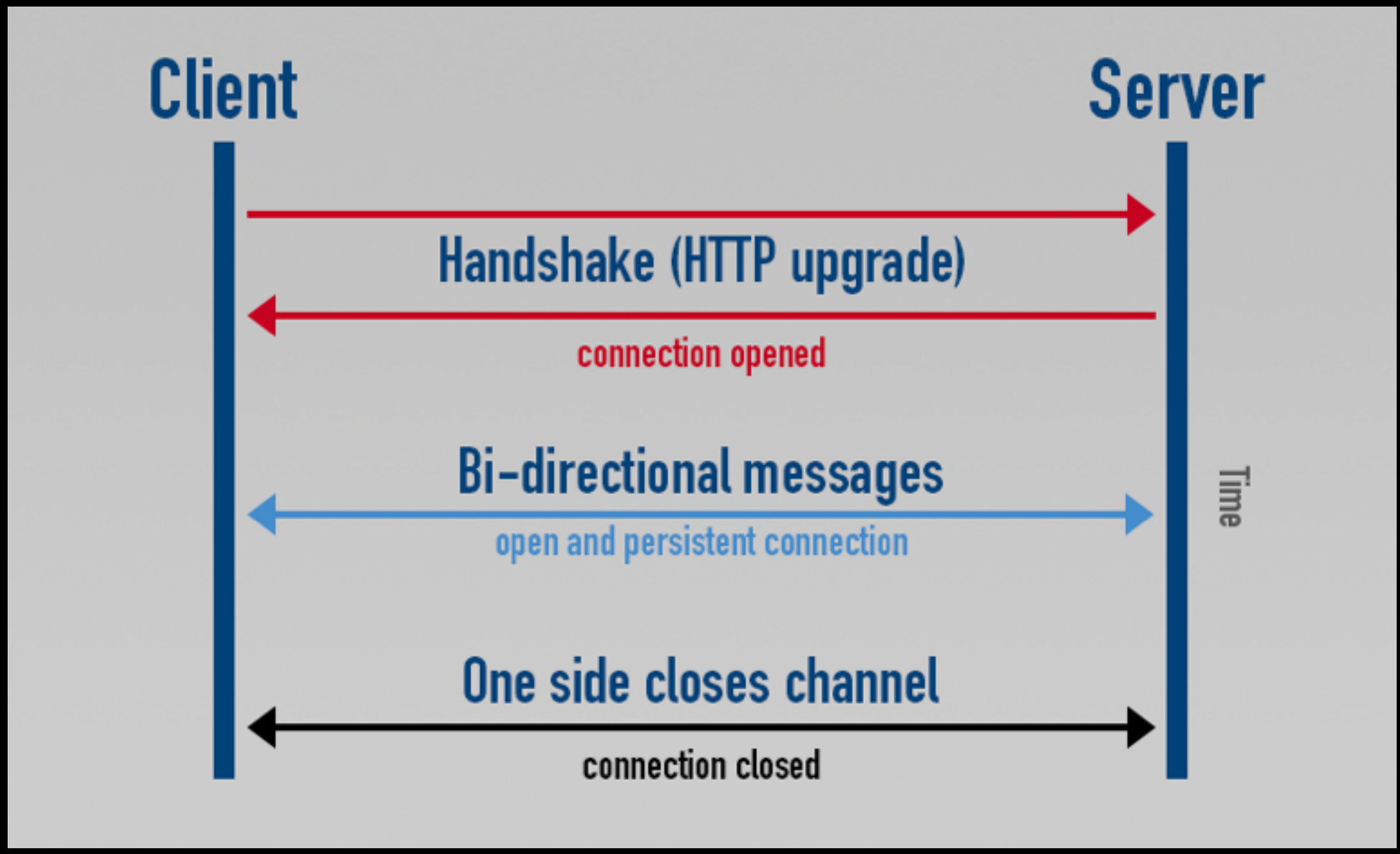
WebSocket: Handshake Example

CLIENT REQUEST

```
GET /cable HTTP/1.1
Host: zeroeleven.rs:3000
Cookie: session=N3Yyc0Z0cVdm
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

SERVER RESPONSE

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```



WebSocket

Visual Representation

WebSocket: API

```
var ws = new WebSocket("ws://zeroeleven.rs");

ws.onopen = function(evt) { alert("Connection open..."); };
ws.onmessage = function(evt) { alert("Received Message: " +
evt.data); };
ws.onclose = function(evt) { alert("Connection closed."); };

ws.send("Hello from Start-it!");
ws.close();
```

Terminology

Connection

Consumer

Channel

Subscriber

Subscription

Pub/Sub

Broadcasting

Pub/Sub Adapters

async

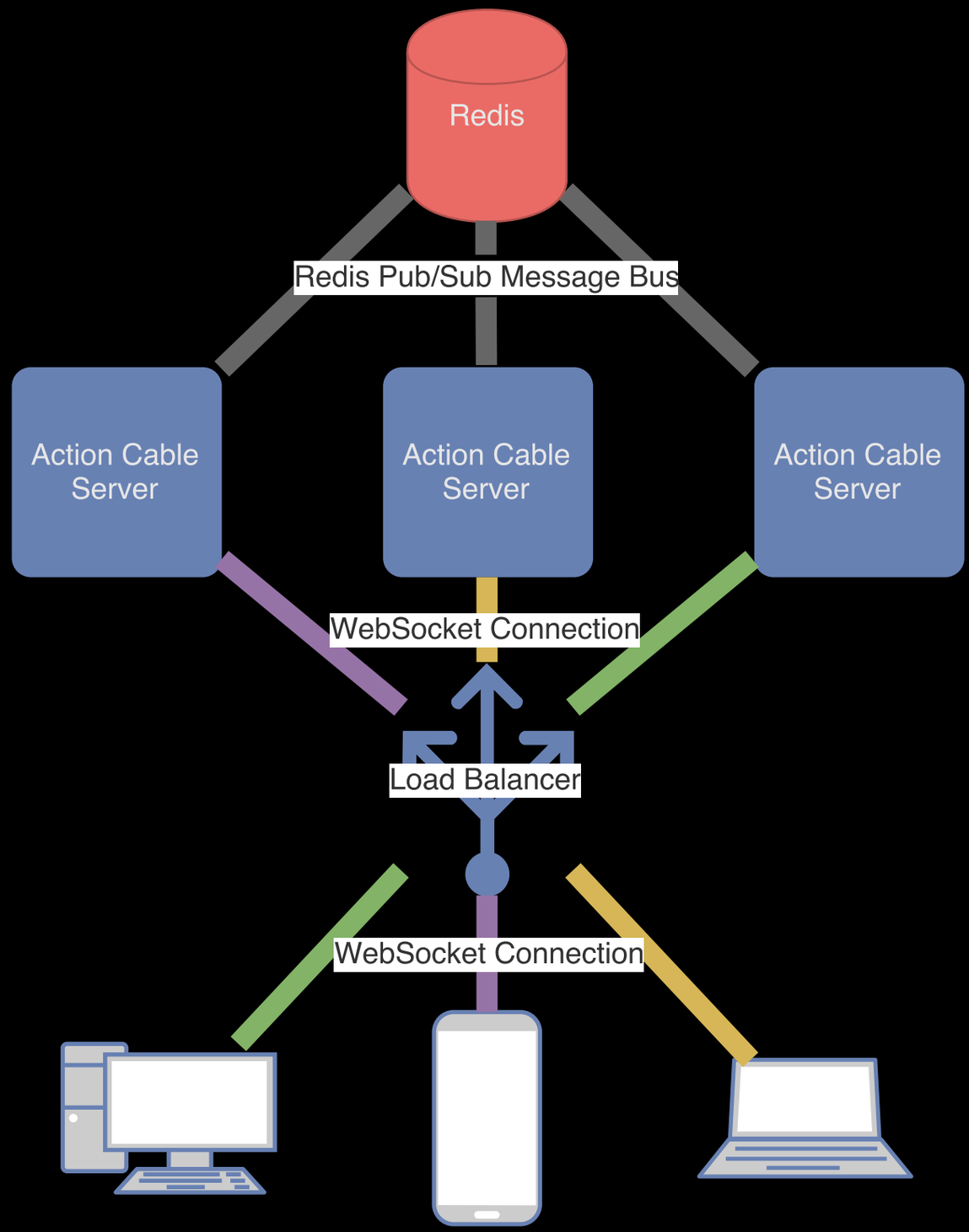
Redis

PostgreSQL

Running the ActionCable Server

In-App

Standalone



Visualisation of ActionCable in Production

```
$ rails new start-it
```

```
app/
  channels/
    application_cable/
      channel.rb
      connection.rb
```

```
app/
  assets/
    javascripts/
      cable.js
```

```ruby
# app/channels/application_cable/connection.rb

module ApplicationCable
  class Connection < ActionCable::Connection::Base
    identified_by :current_user

    def connect
      self.current_user = find_verified_user
    end

    def find_verified_user
      if verified_user = User.find_by(id: cookies.signed[:user_id])
        verified_user
      else
        reject_unauthorized_connection
      end
    end
  end
end
```

```
// app/assets/javascripts/cable.js

//= require action_cable
//= require_self
//= require_tree ./channels

(function() {
  this.App || (this.App = {});

  App.cable = ActionCable.createConsumer();

}).call(this);
```

```
$ rails generate channel room speak
```

```
app/
  channels/
    room_channel.rb
```

```
app/
  assets/
    javascripts/
      channels/
        room.coffee
```

```ruby
# app/channels/room_channel.rb

class RoomChannel < ApplicationCable::Channel
  def subscribed
    stream_from "room_channel"
  end

  def unsubscribed
  end

  def speak(data)
    ActionCable.server.broadcast 'room_channel', message:
data['message']
  end
end
```

```
# app/assets/javascripts/channels/room_channel.js

App.room = App.cable.subscriptions.create "RoomChannel",
  connected: ->

  disconnected: ->

  received: (data) ->
    $("#messages").append data["message"];

  speak: (message) ->
    @perform "speak", message: message

App.room.speak "SIR, YES SIR!"
```

Battleship Game Example