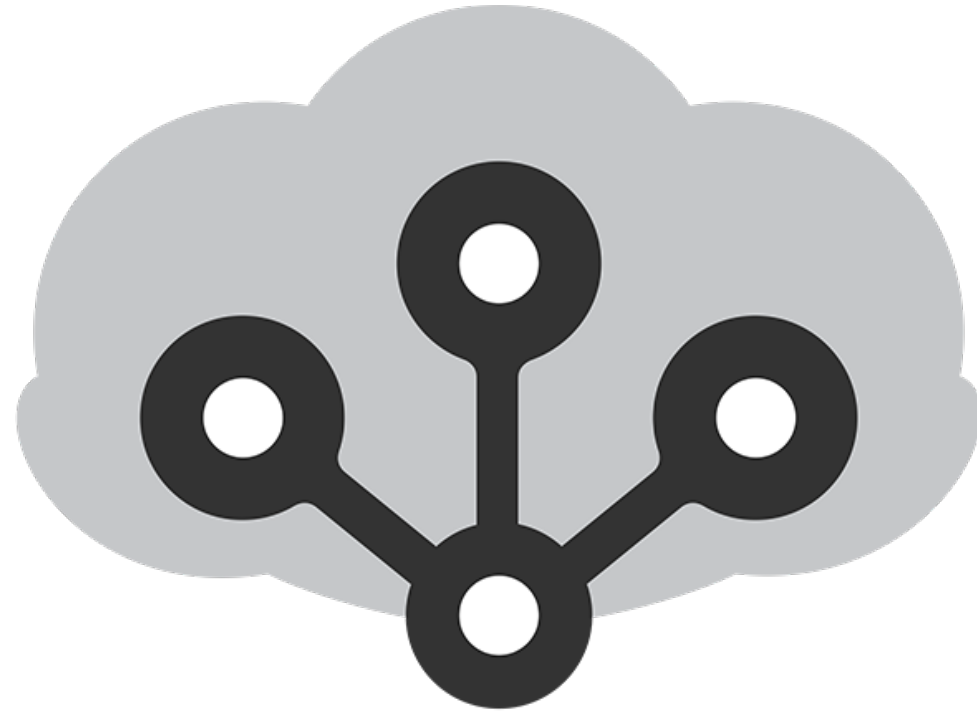


real-time
sweg



nodejitsu

observe.it



3rdeden



3rd-eden

with a dash

what does it mean to have
swag

sweg

socket.io

sweg

sackys

sweg

websocket

sweg

engine.io

sweg

google

browserchannel

why do you need
sweg

dude
websocket
all the things!

Chrome 20



Chrome 4
older protocol



Firefox 12



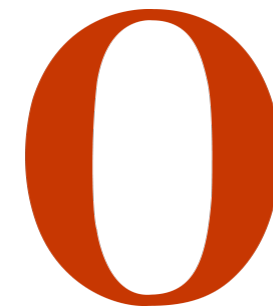
Firefox 4
older protocol



Opera 12.1



Opera 11
older protocol behind flag



Safari 6



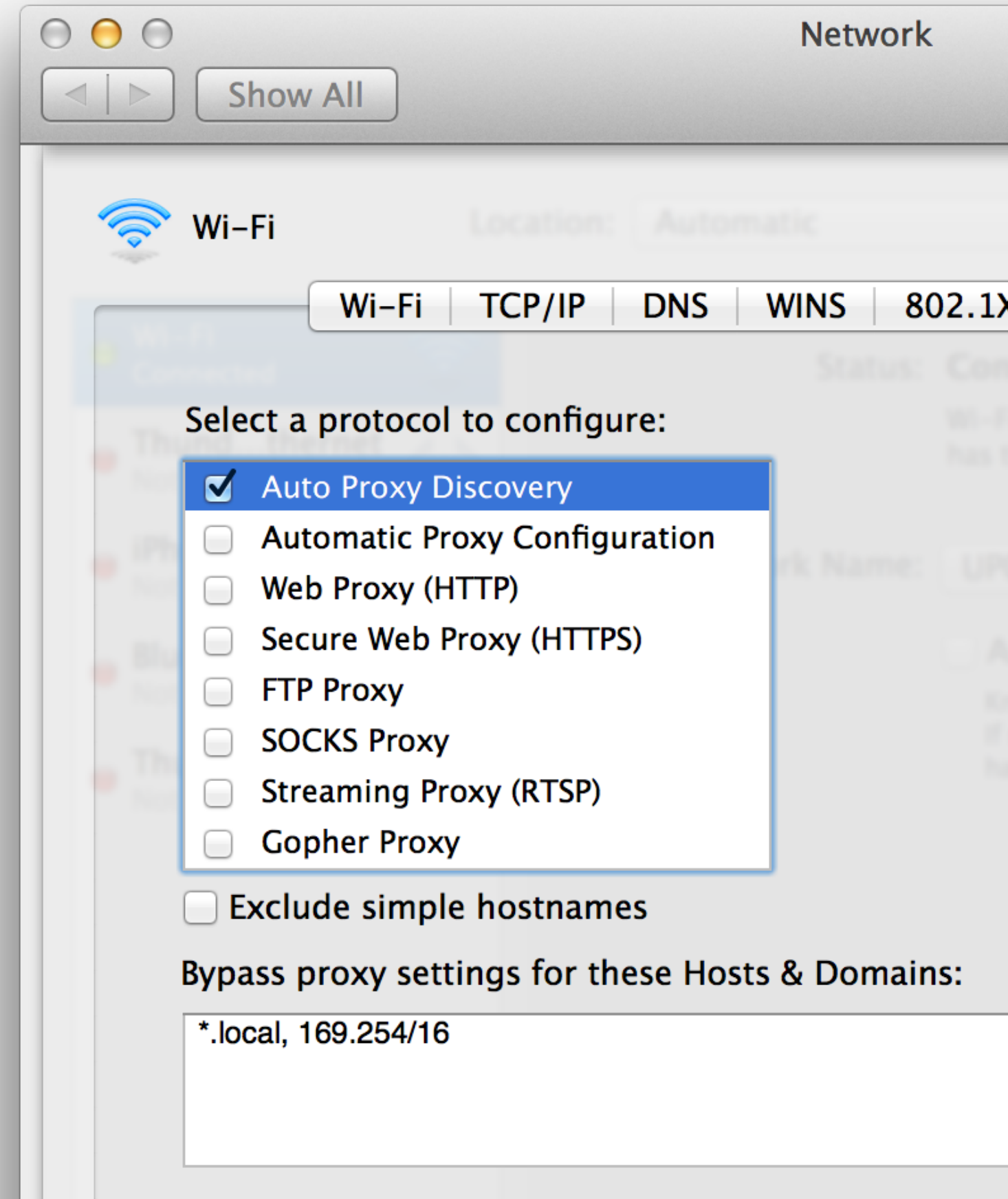
Safari 4.2
older protocol



Internet Explorer 10
finally



HTTP proxy settings in your network settings can cause a full browser crash.



Writing to a
can cause a full browser crash

Pressing
the WebSocket connection

Firefox can create
when you connect during

4G, 3G, LTE, mobile providers WTF
ARE YOU DOING?? —

Virus scanners such as AVG
WebSockets.

User, network and server firewalls
block

Load balancers don't understand and
block

```
new WebSocket("wss://url.io");
```

doesn't look so simple anymore

*but it has its
use cases*

low latency

binary

low bandwidth

don't care about older browsers

polling..
they see me
they hating

Removing spinners with `<iframes>`
for JSONP

Back/Forward & browse cache busting

Client -> Server
Server -> Client
heartbeats

Protocol invention

choosing your
sweg

socket.io 0.9

<http://github.com/automattic/socket.io>

- ✓ multiple transports
- ✓ cross domain
- ✗ invested with bugs
- ✗ poorly / not maintained / **left for dead**
- ✗ no message order guarantee
- ✗ dies behind firewall/virus scanners

Cross domain

Multiple transports

Sending average amounts of data

Not consumer facing

engine.io and socket.io 1.0

<http://github.com/automattic/engine.io>

- ✓ supports multiple transports
- ✓ cross domain
- ✓ upgrade instead of downgrade
- ✓ works behind firewalls & virusscanners
- ✗ not well battle tested yet
- ✗ no message order guarantee

Quick connections

Don't care much about latency

Cross browser

Binary needed

google's browserchannel

<https://github.com/josephg/node-browserchannel>

<https://code.google.com/p/closure-library/source/browse/closure/goog/net/browserchannel.js>

- ✓ multiple transports
- ✓ client maintained by google
- ✓ message order guaranteed
- ✓ works behind firewalls & virusscanners
- ✗ not cross domain
- ✗ no websocket support
- ✗ coffeescript on the server for node ._.
- ✗ not well documented & relatively unknown

Sending real-time updates

Cross browser

No need for binary data

Medium lived connections

Stability required over latency

sockjs

<https://github.com/sockjs>

- ✓ multiple transports
(tons of them)
- ✓ cross domain
- ✗ poor error handling
- ✗ no query string allowed for connect
- ✗ connection delay with firewalls
- ✗ poorly maintained
- ✗ in the way of developers

Sending lots of data using the most optimal transport

Cross domain

Cross browser

No need for binary

Long lived connected sessions

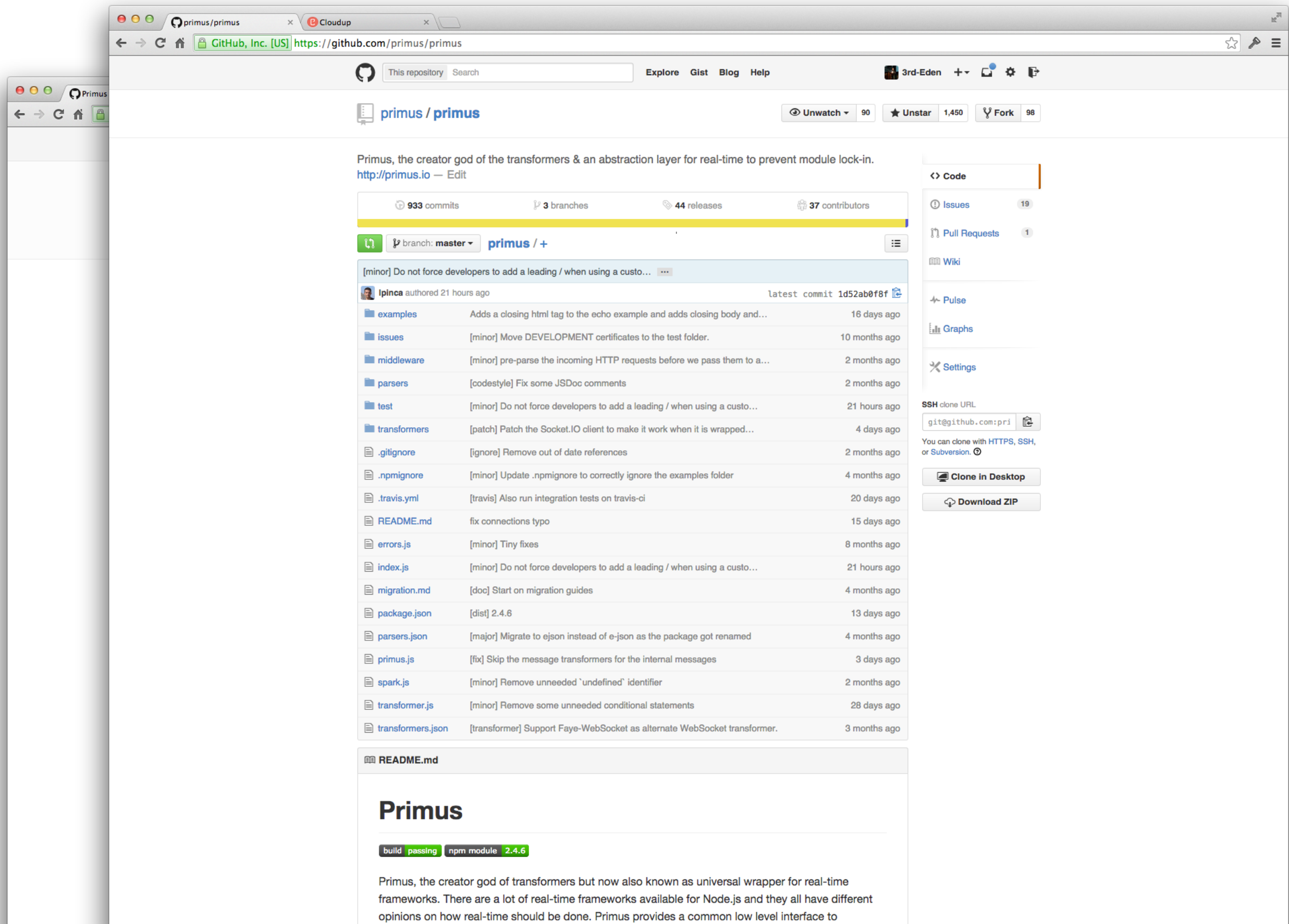
yolo
your only library option



primus is yolo for sweg

```
npm install primus
```


Primus wraps
real-time frameworks. So you can
focus on building apps.






```
cd your-awesome-project
```

```
$ npm install --save primus ws
```

```
echo "??"
```

```
echo "profit!"
```

```
vim index.js
```




```
'use strict';

var Primus = require("primus")
  , server = require("http").createServer(fn)
  , primus = new Primus(server, { transformer: "ws" });

primus.on("connection", function connection(spark) {
  console.log("connection received", spark.id);
  spark.write("ohai");

  spark.on("data", function data(msg) {
    console.log("received", msg);
  });
});

server.listen(8080);
```



```
<script src="http://localhost:8080/primus/primus.js"></script>
<script>
'use strict';


var primus = new Primus("http://localhost:8080");

primus.on("open", function connected() {
  console.log("connection opened");
  primus.write("ohai");
});

primus.on("data", function data(msg) {
  console.log("received", msg);
});
</script>
```




```
var primus = new Primus(server, {  
  transformer: "sockjs" // engine.io, socket.io etc  
});
```



```
module.exports = require("primus/transformer").extend({
  server: function () {
    // This is only exposed and ran on the server.
  },

  client: function () {
    // This is stringified and send/stored in the client.
    // Can be ran on the server, if used through Node.js
  },


  // Optional library for the front-end, assumes globals
  library: fs.readFileSync(__dirname + "./yourclientlib.js")
});
```




```
primus.on("end", function disconnected() {  
  console.log("connection ended");  
});  
  
primus.end();  
primus.write();  
  
fs.createReadStream(__dirname + '/index.html').pipe(spark, {  
  end: false  
});
```




```
var primus = new Primus(server, {  
  parser: "JSON" // JSON by default  
});
```




```
var primus = new Primus(server, {  
  parser: "EJSON" // or binary-pack or a third party module  
});
```



```
module.exports = {
  encoder: function (data, fn) {
    // encode data to a string.
  },


  decoder: function (data, fn) {
    // decode data to an object
  },

  // Optional library for the front-end, assumes globals
  library: fs.readFileSync(__dirname + "./yourclientlib.js")
};
```




```
primus.transform('incoming', function (packet) {  
  // This would transform all incoming messages to foo;  
  packet.data = 'foo';  
});
```

```
primus.transform('outgoing', function (packet) {  
  // This would transform all outgoing messages to foo;  
  packet.data = 'foo';  
});
```



```
var primus = new Primus("http://localhost:8080", {  
  strategy: "disconnect, online"  
});
```




```
var Primus = require("primus")
  , server = require("http").createServer(fn)
  , primus = new Primus(server, { transformer:"ws" });

primus.write("message"); // send message to all users

primus.forEach(function (spark) {
  // Or iterate over all connections, select the once you
  // want and only write to those


  spark.write("message");
});
```




```
// The long awaited Socket.IO 1.0 release with Primus:
```

```
var server = require("http").createServer(fn)  
  , primus = new Primus(server, { transformer: "engine.io" });
```

```
primus.use("emitter", "primus-emitter")  
      .use("multiplex", require("primus-multiplex"))  
      .use("primus-emitter", "primus-rooms");
```




```
module.exports = {  
  server: function () {  
    // This is only exposed and ran on the server.  
  },  
  
  client: function () {  
    // This is stringified and send/stored in the client.  
    // Can be ran on the server, if used through Node.js  
  },  
  
  // Optional library for the front-end, assumes globals  
  library: fs.readFileSync(__dirname + "./yourclientlib.js")  
};
```

```
var server = require("http").createServer(fn)
  , primus = new Primus(server, { transformer: "sockjs" });

primus.before("session", require("session-parse-module"))
  .before("middleware-name", "middleware-module-name");
```



```
var server = require("http").createServer(fn)
  , primus = new Primus(server, { transformer: "engine.io" });

primus.on("connection", function middlewarish(spark, next) {
  // do async stuff, all other "connection" events will not
  // be called until this one completes..
  next();
});
```



infinite use cases

lin



3rdeden