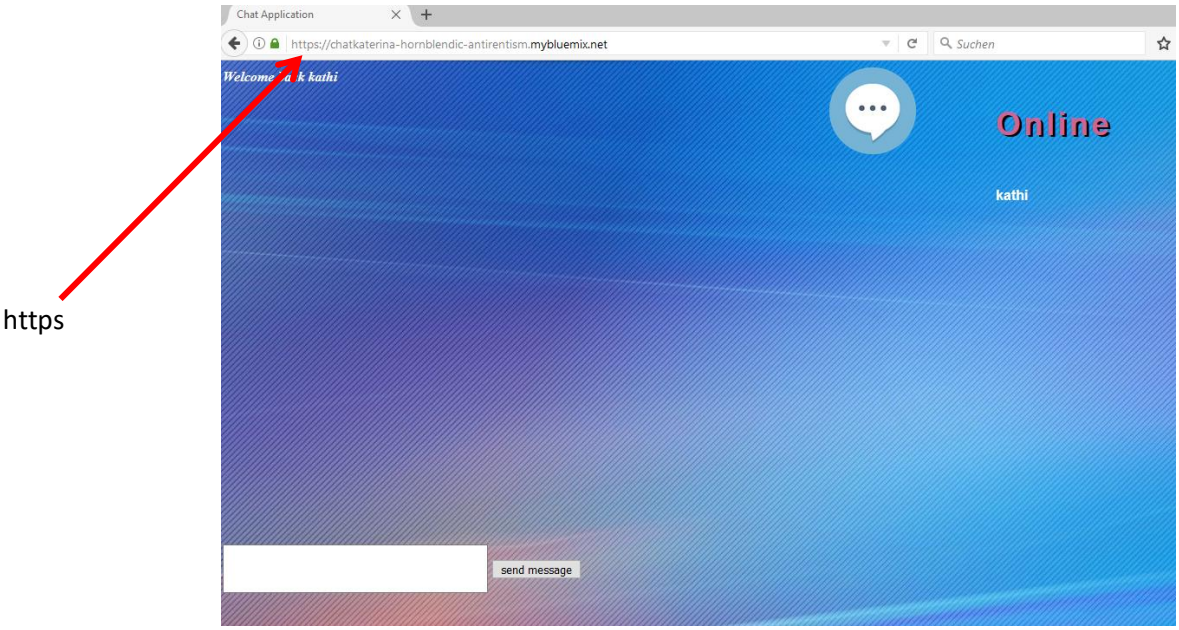
Cloud Computing – Exercise 4
Scalable Chat Server

Marek Mauser 731184
Katherina Kagelidou 731638

1. Rework from the last exercise

We were able to implement points that we unfortunately did not manage to achieve in the last exercise.

➔ Ensure data confidentiality and integrity by using TLS channels between client and server only.



Because the platform uses IBM DataPower which provides an endpoint for HTTP and HTTPS client traffic to the BlueMix Plattform, both urls wil work in our application. What we only have to do is:

```
app.enable('trust proxy');

// get to https
app.use(function(req, res, next) {
    if (req.secure) {
        next();
    } else {
        res.redirect('https://' + req.headers.host + req.url);
    }
});
```

to use the applicaton (url) and wrote https:// before every path. So we have reached our save connection.

2. Scalable Chat Server

➔ Analyse your code and identify any issue that prevents your current implementation from being run on multiple instances (scaled-out mode).

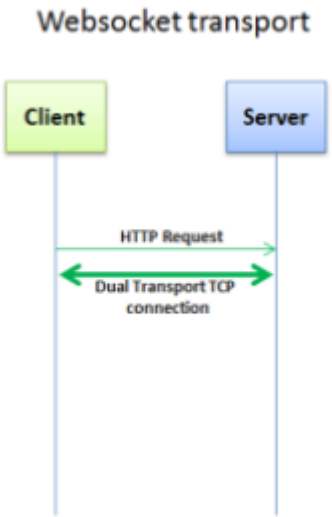
First of all the port would make problems because the instances should be run on different ports. Another point is that we don't have a database to allow every instance to access the same data and to be shared each other.

➔ Implement one of these solution on the Bluemix Cloud Platform.

Websocket transport - Client sends HTTP request first to Server and **Server establishes dual stateful connection** to send/recv anytime to/from Client.

```
var io = require('socket.io').listen(server, {transports: ['websocket']});
```

The WebSocket connection is stable, that means one connect and send/recv message wherever Client/Server wants to.



(vgl. https://mashhurs.wordpress.com/2016/09/30/polling-vs-websocket-transport/*)*

➔ Redis-service to allow more instances

```
//redis connection with out port, hostname and password
var client = redis.createClient(17981, 'pub-redis-17981.dal-05.1.sl.garantiadata.com', {no_ready_check: true});
client.auth('Qo49Fq6cSoqQzgYS', function (err) {
  if (err) throw err;
});

client.on('connect', function() {
  console.log('Connected to Redis');
});


var subscriber = redis.createClient(17981, 'pub-redis-17981.dal-05.1.sl.garantiadata.com');
subscriber.on("error", function(err) {
  console.error('There was an error with the redis client ' + err);
});
var publisher = redis.createClient(17981, 'pub-redis-17981.dal-05.1.sl.garantiadata.com');
publisher.on("error", function(err) {
  console.error('There was an error with the redis client ' + err);
});
```

Lession-Learnt

- How to create multiple instances with node js and socket.io
- Websocket for transport
- Redis with pub and sub to send date to all the connected clilents which have maybe different instances