

Simple Callcenter Platform with PHP



Astricon 2018, October 9th, Orlando, FL
Morten Amundsen / @mor10am

About me & **Teleperformance** each interaction matters

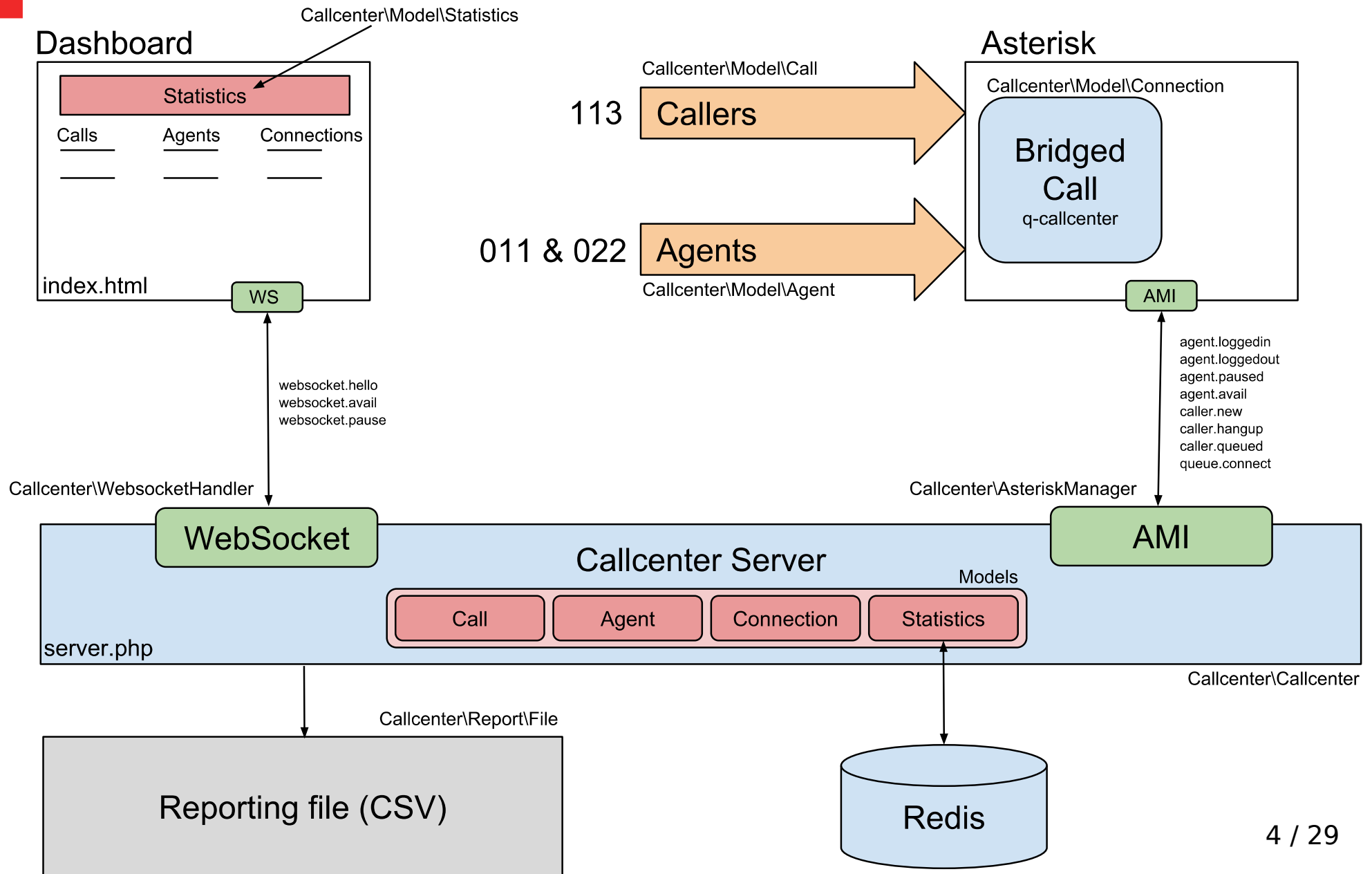
- From Norway, Working remote from Italy.
- Been with Teleperformance Nordic since 2003.
- Teleperformance is a contact center provider from France with presence in 76 countries and has over 220.000 employees.
- Teleperformance Nordic is serving clients in Norway, Sweden, Denmark and Finland.



Creating the callcenter

- Configuring Asterisk for agents and callers
- Create a PHP server process:
 - Communicate with Asterisk (AMI)
 - Communicate with Webpage (WS)
 - Create report
 - Keep callcenter state in Redis
 - Restore on server restart
- Create a simple dashboard in Vue.js
 - Calls, agents, connections and statistics
- Deploy to an Ubuntu VM with Ansible

Overview





Asterisk

- Asterisk 13 on Ubuntu 18.04 VM (universe)
- Main config files modified/used:
 - sip.conf
 - http.conf
 - manager.conf
 - queues.conf
 - extensions.conf



Dialplan

```
[public]  
exten => 113,1,goto(callcenter,s,1)  
exten => 011,1,goto(agent-login,s,1)  
exten => 022,1,goto(agent-logout,s,1)
```

Dialplan (Caller)

```
[callcenter]
exten => s,1,NoOp
    same => n,Answer
    same => n,Set(CHANNEL(language)=en)
    same => n,Set(CHANNEL(hangup_handler_push)=hangup-handler,s,1)
    same => n,UserEvent(CALLER)
    same => n,Queue(q-callcenter,,3600)
    same => n,Hangup(16)
```

```
[hangup-handler]
exten => s,1,noop
    same => n,UserEvent(CALLERHANGUP)
    same => n,GotoIf("${MEMBERINTERFACE}" = "")?skip
    same => n,UnpauseQueueMember(q-callcenter,${MEMBERINTERFACE})
    same => n(skip),Return
```

Interface of agent
if caller talked to one

Dialplan (Agent login)

```
[agent-login]
exten => s,1,noop
    same => n,Answer
    same => n,Set(CHANNEL(language)=en)
    same => n,Set(agentid=${CHANNEL(peername)})
    same => n,Set(queue-member=local/${agentid}@agent-connect)
    same => n,AddQueueMember(q-callcenter,${queue-member},,,${agentid})
    same => n,PauseQueueMember(q-callcenter,${queue-member})
    same => n,UserEvent(LOGGEDIN,agentid:${agentid},member:${queue-member})
    same => n,Playback(agent-loginok)
    same => n,hangup(16)
```

Channel peername is used as agent Id

Dialplan (Agent logout)


```
[agent-logout]
exten => s,1,noop
    same => n,Answer
    same => n,Set(CHANNEL(language)=en)
    same => n,Set(agentid=${CHANNEL(peername)})
    same => n,Set(queue-member=local/${agentid}@agent-connect)
    same => n,RemoveQueueMember(q-callcenter,${queue-member})
    same => n,UserEvent(LOGGEDOUT,agentid:${agentid},member:${queue-
member})
    same => n,Playback(agent-loggedoff)
    same => n,hangup(16)
```

Channel peername is used as agent Id

Dialplan (Bridge call)

Agent Id is the same
as the extension.
ex.: local/1234@agent-connect

```
[agent-connect]  
exten => _X.,1,Dial(SIP/${EXTEN},30,r)
```





Demo



PHP Server

- Libraries:
 - Ratchet (Websocket connection)
 - *<http://socketo.me/>*
 - React (Asterisk AMI TCP connection)
 - *<https://reactphp.org/>*
 - PAMI (Encoding/decoding AMI messages)
 - *<http://marcelog.github.io/PAMI/>*
 - *Redis (storage of server state)*

PHP Server

```
$loop = React\EventLoop\Factory::create();

$app = new Ratchet\App(
    'callcenter.local',           // HTTP hostname clients intend to connect to
    8080,                         // Port to listen on.
    '0.0.0.0',                   // IP address to bind to
    $loop                        // React\EventLoop to bind the application to
);

$websockethandler = new \Callcenter\WebSocketHandler();

$app->route(
    '/callcenter',               // The URI the client will connect to
    $websockethandler,           // Your application to server for the route
    ['*']                        // An array of hosts allowed to connect
);
```

PHP Server

```
$ami = new \React\Stream\DuplexResourceStream(  
    stream_socket_client('tcp://callcenter.local:5038'),  
    $loop  
);  
  
$asteriskmanager = new Callcenter\AsteriskManager($ami);  
  
$reportwriter = new \Callcenter\Report\File(__DIR__."/report.csv");  
  
$redis = new \Redis();  
$redis->connect('127.0.0.1');  
  
$callcenter = new Callcenter\Callcenter(  
    $websockethandler,  
    $asteriskmanager,  
    $reportwriter,  
    $redis  
);
```

PHP Server

```
$websockethandler->on('websocket.hello', [$callcenter, 'websocketHello']);
$websockethandler->on('websocket.avail', [$callcenter, 'websocketSetAgentAvail']);
$websockethandler->on('websocket.pause', [$callcenter, 'websocketSetAgentPause']);

$asteriskmanager->on('agent.loggedin', [$callcenter, 'agentLoggedIn']);
$asteriskmanager->on('agent.loggedout', [$callcenter, 'agentLoggedOut']);
$asteriskmanager->on('agent.paused', [$callcenter, 'agentPaused']);
$asteriskmanager->on('agent.avail', [$callcenter, 'agentAvail']);

$asteriskmanager->on('caller.new', [$callcenter, 'callNew']);
$asteriskmanager->on('caller.hangup', [$callcenter, 'callHangup']);
$asteriskmanager->on('caller.queued', [$callcenter, 'callQueued']);

$asteriskmanager->on('queue.connect', [$callcenter, 'callAndAgentConnected']);

$asteriskmanager->login('admin', 'password');

$app->run();
```

PHP Server Event

Event: **websocket.avail**

- Callcenter method “websocketSetAgentAvail” get event from WebsocketHandler
- Gets the agentid from this event
- Get existing agent or create a new
- Calls Asterisk Manager Interface method to unpause given agent

```
$websockethandler->on('websocket.avail', [$callcenter, 'websocketSetAgentAvail']);
```

```
public function websocketSetAgentAvail(CallcenterEvent $event) : void
{
    $agent = $this->getOrCreateAgent(
        $event->get('agentid'),
        $event->get('member', null)
    );

    $this->ami->unpauseAgent($agent->getMember());
}
```


PHP Server Event

Event: **call.new**

- Callcenter method “callNew” get event from AsteriskManager
- Create new call
- Send info about call to all connected websocket dashboards together with updated statistics.

```
$asteriskmanager->on('caller.new', [$callcenter, 'callNew']);
```

```
public function callNew(CallcenterEvent $event) : void
{
    $call = $this->getOrCreateCall($event->callerid, $event->uid);

    $this->websocket->sendtoAll(
        json_encode($call)."\n".
        $this->calcAndSerializeStats()."\n"
    );
}
```

PHP Server Event

Event: **queue.connect**

- Get Call and Agent objects, and update their status to INCALL
- Create new Connection model with Call and Agent
- Update dashboard with Agent, Call and Connection, plus updated statistics
- Send URL or Event to Agents computer. Not implemented.

```
$asteriskmanager->on('queue.connect', [$callcenter, 'callAndAgentConnected']);
```

```
public function callAndAgentConnected(CallcenterEvent $event) : void
{
    $agentid = $event->get('agentid');
    $calleruid = $event->get('calleruid');

    $agent = $this->agents[$agentid];
    $call = $this->calls[$calleruid];

    $this->setCallStatus($call, 'INCALL');

    $agent->setQueue($call->getQueue());
    $this->setAgentStatus($agent, 'INCALL');

    $conn = new Connection($call, $agent);

    $this->agentcallconnections[$conn->id] = $conn;

    $this->websocket->sendtoAll(
        json_encode($agent)."\n".json_encode($call)."\n".json_encode($conn)."\n".
        $this->calcAndSerializeStats()."\n"
    );

    /* !!! SEND URL OR EVENT TO AGENT DASHBOARD CRM OR OTHER SYSTEM !!! */
}
```

Report file

2018-09-17 15:11:07;AGENT;98427456;LOGGEDIN;12;
2018-09-17 15:11:11;CALL;3453439819;QUEUED;15;q-callcenter
2018-09-17 15:11:26;CALL;3453439819;ABANDON;0;q-callcenter
2018-09-17 15:11:39;CALL;3453439819;QUEUED;3;q-callcenter
2018-09-17 15:11:19;AGENT;98427456;AVAIL;23;q-callcenter
2018-09-17 15:11:42;CALL;3453439819;INCALL;12;q-callcenter
2018-09-17 15:11:54;CALL;3453439819;HANGUP;0;q-callcenter
2018-09-17 15:11:42;AGENT;98427456;INCALL;12;q-callcenter
2018-09-17 15:12:08;CALL;3453439819;QUEUED;1;q-callcenter
2018-09-17 15:12:09;CALL;3453439819;ABANDON;0;q-callcenter
2018-09-17 15:12:13;CALL;3453439819;QUEUED;2;q-callcenter
2018-09-17 15:11:54;AGENT;98427456;AVAIL;21;q-callcenter
2018-09-17 15:12:15;CALL;3453439819;INCALL;7;q-callcenter
2018-09-17 15:12:22;CALL;3453439819;HANGUP;0;q-callcenter
2018-09-17 15:12:15;AGENT;98427456;INCALL;7;q-callcenter
2018-09-17 15:12:22;AGENT;98427456;AVAIL;2;
2018-09-17 15:12:24;AGENT;98427456;PAUSED;49;

Dashboard

Agents 1 Calls 1 Connections 1 Received 21 Answered 12 Abandon 9 AHT 205 sec AQT 7 sec AAT 10 sec SLA 66.67%

Calls

3453439819 **INCALL**

q-callcenter

Agents

98427456 **INCALL**

q-callcenter

Connections

Call: 3453439819 Agent: 98427456

q-callcenter

- Built with:
 - Vue.js
 - <https://vuejs.org/>
 - Bootstrap CSS
 - <https://getbootstrap.com/>
 - Websocket

Dashboard (cont.)

```
var app = new Vue({
  el: '#app',
  conn: null,
  data: {
    calls: {},
    agents: {},
    connections: {},
    stats: {}
  },
  mounted: function() {
    this.conn = new WebSocket('ws://callcenter.local:8080/callcenter');

    this.conn.onopen = function (e) {
      this.send('HELLO');
    };

    this.conn.onmessage = function (e) {
      let lines = e.data.split("\n");

      lines.forEach(function (val, index) {
        if (val.length) {
          this.updateUI(val);
        }
      }, app)
    };
  },
});
```

Dashboard (cont.)

```
<div id="app">
  <div class="row">
    <stats :stats="stats" />
  </div>

  <div class="row">
    <div class="col-sm">
      <h3>Incoming calls</h3>
      <call v-for="call in calls" :call="call" :key="call.id"/>
    </div>

    <div class="col-sm">
      <h3>Agents</h3>
      <agent v-for="agent in agents" :agent="agent" :key="agent.agentid" />
    </div>

    <div class="col-sm">
      <h3>Connected calls</h3>
      <connection v-for="connection in
connections" :connection="connection" :key="connection.id"/>
    </div>

  </div>
</div>
```

Dashboard (cont.)

```
<call v-for="call in calls" :call="call" :key="call.id"/>
```

```
Vue.component('call', {  
  props: {  
    call: Object  
  },  
  template: '#call-template'  
});
```

```
<script type="text/x-template" id="call-template">  
  <div class="row">  
    <div class="col-sm">  
      <div class="alert alert-info">{{ call.callerid }} {{ call.status }}<span  
class="badge badge-secondary">{{ call.queue }}</span></div>  
    </div>  
  </div>  
</script>
```

Dashboard (cont.)

```
updateUI(msg) {  
    let obj = JSON.parse(msg);  
    if (!obj) return;  
    switch (obj.type) {  
        case 'AGENT':  
            this.addOrUpdateAgent(obj);  
            break;  
        case 'CALL':  
            this.addOrUpdateCall(obj);  
            break;  
        case 'CONNECT':  
            this.addOrUpdateConnection(obj);  
            break;  
        case 'STATS':  
            this.stats = obj;  
            break;  
        default:  
            break;  
    }  
}  
});
```


Dashboard (cont.)

```
addOrUpdateCall(call)
{
  if (call.status == 'HANGUP' || call.status == 'ABANDON') {
    this.$delete(this.calls, call.id);
    // connection has same Id as call
    this.$delete(this.connections, call.id);
  } else {
    this.$set(this.calls, call.id, call);
  }
},

addOrUpdateAgent(agent)
{
  if (agent.status == 'LOGGEDOUT') {
    this.$delete(this.agents, agent.agentid);
  } else {
    this.$set(this.agents, agent.agentid, agent);
  }
},

addOrUpdateConnection(connection)
{
  this.$set(this.connections, connection.id, connection);
}
```

Dashboard (cont.)

Messages sent over Websocket from PHP server to dashboard:

```
{
  "type": "CALL",
  "id": "1536414212.52",
  "callerid": "98427456",
  "status": "QUEUED",
  "queue": "q-callcenter",
  "time": 1536414212,
  "answered": 0
}
```

```
{
  "type": "AGENT",
  "agentid": "10092018",
  "status": "AVAIL",
  "queue": "",
  "time": 1536414159
}
```

```
{
  "type": "CONNECT",
  "id": "1536414129.44",
  "queue": "q-callcenter",
  "time": 1536414174,
  "agent": {
    "type": "AGENT",
    "agentid": "10092018",
    "status": "INCALL",
    "queue": "q-callcenter",
    "time": 1536414174
  },
  "call": {
    "type": "CALL",
    "id": "1536414129.44",
    "callerid": "98427456",
    "status": "INCALL",
    "queue": "q-callcenter",
    "time": 1536414174,
    "answered": 1
  }
}
```

```
{
  "type": "STATS",
  "calls_received": 8,
  "calls_abandoned": 1,
  "calls_answered": 7,
  "agents_online": 1,
  "calls_online": 0,
  "connections_online": 0,
  "average_handle_time": 11,
  "average_queue_time": 9,
  "average_abandoned_time": 6,
  "sla": 71.43
}
```

Ansible deployment

- Install PHP and Asterisk
- Copy files in place
 - Asterisk configuration files
 - PHP sources
 - HTML and JS
- Reload Asterisk
- Start CTI server

Ansible deployment (cont.)

```
$> ansible-playbook ansible/playbook.yml -i ansible/inventories/production
```

inventories/production:

```
[asterisk]  
callcenter.local
```

Playbook.yml:

```
- hosts: asterisk  
  remote_user: deploy  
  become: yes  
  roles:  
    - ansible-role-redis  
  ...  
- name: copy asterisk etc folder to project  
  synchronize:  
    src: asterisk/etc/  
    dest: /etc/asterisk/  
  notify:  
    - reload asterisk  
  
handlers:  
  - name: reload asterisk  
    shell: asterisk -rx'core reload'
```



Thank you!

Follow us

 /company/teleperformance

 /teleperformanceglobal

 @teleperformance

 @Teleperformance_group

 /teleperformance

 blog.Teleperformance.com



Contact me at **morten.amundsen@teleperformance.com**
or **@mor10am** on Twitter.

Sourcecode and slides at **<https://github.com/mor10am/callcenter>**