# COMEDY

Actors for flexible NodeJS scalability

## Victor Isaev

SAYMON project Team Lead

15 years of development experience (Java, C++, JavaScript).

Developing SAYMON for 4,5 years, using NodeJS for backend.

# NodeJS is single-threaded

## Tools for NodeJS scaling

> CLUSTER

> PM2

> COMEDY

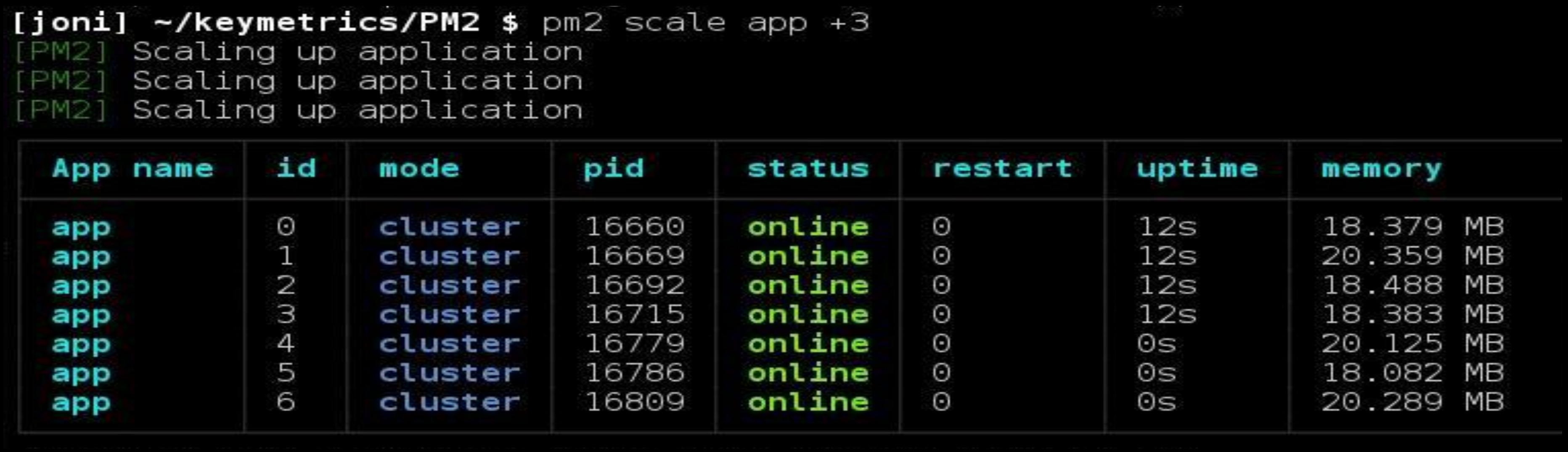
## CLUSTER

```
let cluster = require('cluster');

let http = require('http');

let numCPUs = 4;


if (cluster.isMaster) {

    for (var i = 0; i < numCPUs; i++) {

        cluster.fork();

    }

} else {

    http.createServer(function(req, res) {

        res.writeHead(200);

        res.end('process ' + process.pid);

    }).listen(8000);

}
```

# CLUSTER

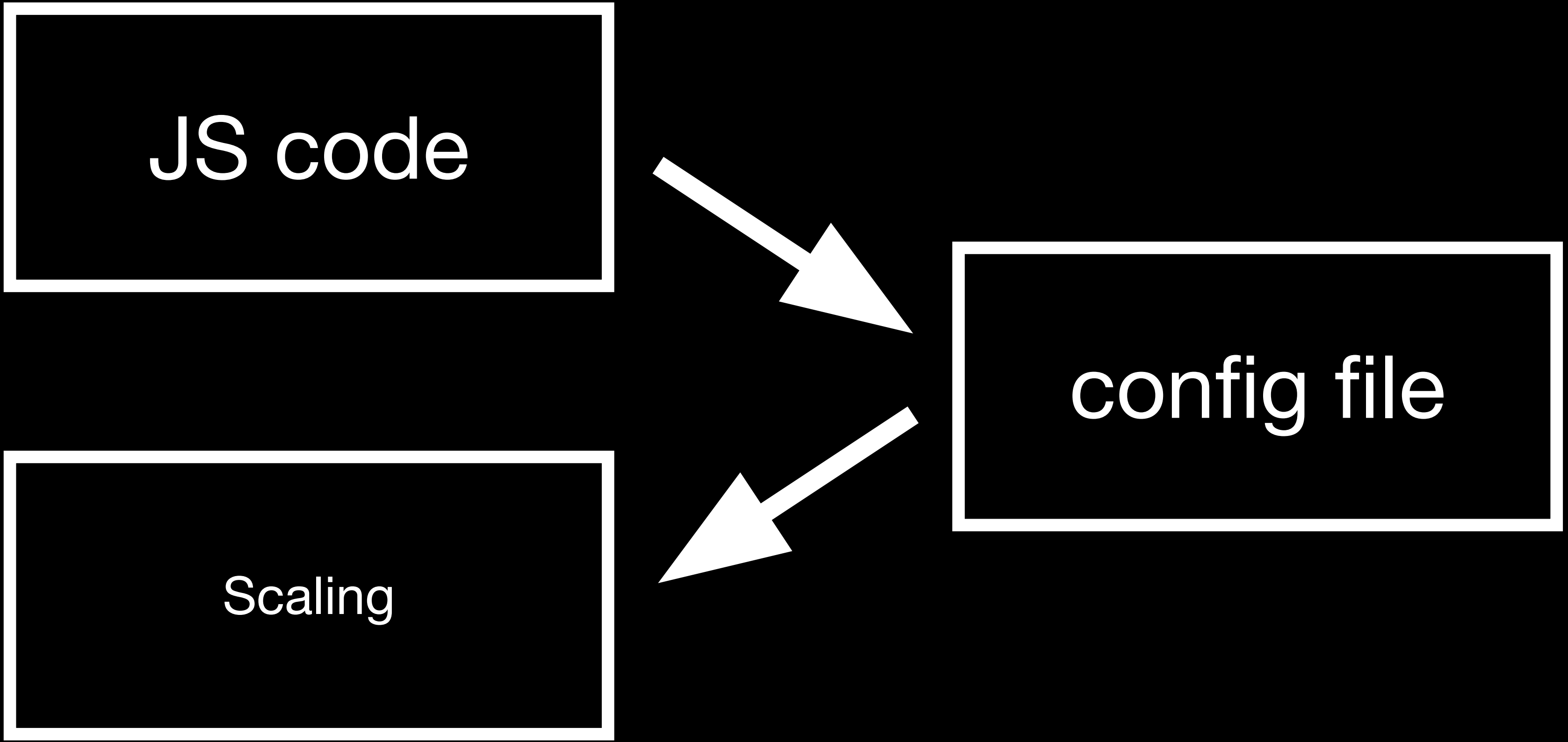
```
let cluster = require('cluster');
let http = require('http');
let numCPUs = 4;

if (cluster.isMaster) {
    for (var i = 0; i < numCPUs; i++) {
        cluster.fork();
    }
} else {
    http.createServer(function(req, res) {
        res.writeHead(200);
        res.end('process ' + process.pid);
    }).listen(8000);
}
```

## PM2



```
[joni] ~/keymetrics/PM2 $ pm2 scale app +3
[PM2] Scaling up application
[PM2] Scaling up application
[PM2] Scaling up application
 App name   id   mode      pid     status   restart   uptime   memory

 app        0    cluster   16660   online   0         12s      18.379 MB
 app        1    cluster   16669   online   0         12s      20.359 MB
 app        2    cluster   16692   online   0         12s      18.488 MB
 app        3    cluster   16715   online   0         12s      18.383 MB
 app        4    cluster   16779   online   0         0s       20.125 MB
 app        5    cluster   16786   online   0         0s       18.082 MB
 app        6    cluster   16809   online   0         0s       20.289 MB
```

## COMEDY

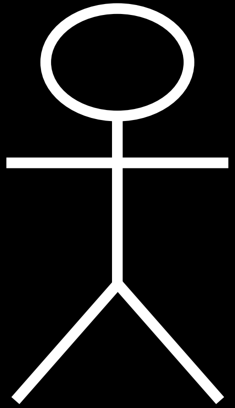


# WHAT IS AN ACTOR?

> receives messages

> sends messages

> spawns child actors



# COMEDY

JS code

```
var actors = require('comedy');


// Actor definition class.
class MyActor {
  sayHello(to) {
    console.log(`Hello, ${to}!`);
  }
}


actors()
    // Get a root actor reference.
  .rootActor()
    // Create a class-defined child actor.
  .then(rootActor => rootActor.createChild(MyActor))
  .then(myActor => {
    // Our actor is ready, we can send messages to it.
    myActor.send('sayHello', 'world');
  });
```

# COMEDY

## JS code

```
var actors = require('comedy');

// Actor definition class.
class MyActor {
  sayHello(to) {
    console.log(`Hello, ${to}!`);
  }
}

actors()
   // Get a root actor reference.
  .rootActor()
   // Create a class-defined child actor.
  .then(rootActor => rootActor.createChild(MyActor))
  .then(myActor => {
    // Our actor is ready, we can send messages to it.
    myActor.send('sayHello', 'world');
  });
```

## COMEDY

JS code

```
var actors = require('comedy');


// Actor definition class.
class MyActor {
  sayHello(to) {
    console.log(`Hello, ${to}!`);
  }
}


actors()
    // Get a root actor reference.
  .rootActor()
    // Create a class-defined child actor.
  .then(rootActor => rootActor. createChild (MyActor))
  .then(myActor => {
    // Our actor is ready, we can send messages to it.
    myActor.send('sayHello', 'world');
  });
```

## COMEDY

JS code

```
var actors = require('comedy');


// Actor definition class.
class MyActor {
  sayHello(to) {
    console.log(`Hello, ${to}!`);
  }
}


actors()
    // Get a root actor reference.
  .rootActor()
    // Create a class-defined child actor.
  .then(rootActor => rootActor.createChild(MyActor))
  .then(myActor => {
    // Our actor is ready, we can send messages to it.
    myActor.send('sayHello', 'world');
  });
```

## COMEDY

Config file

```
{
  "MyActor": {
    "mode": "forked",
    "clusterSize": 3
  }
}
```

## COMEDY

Result

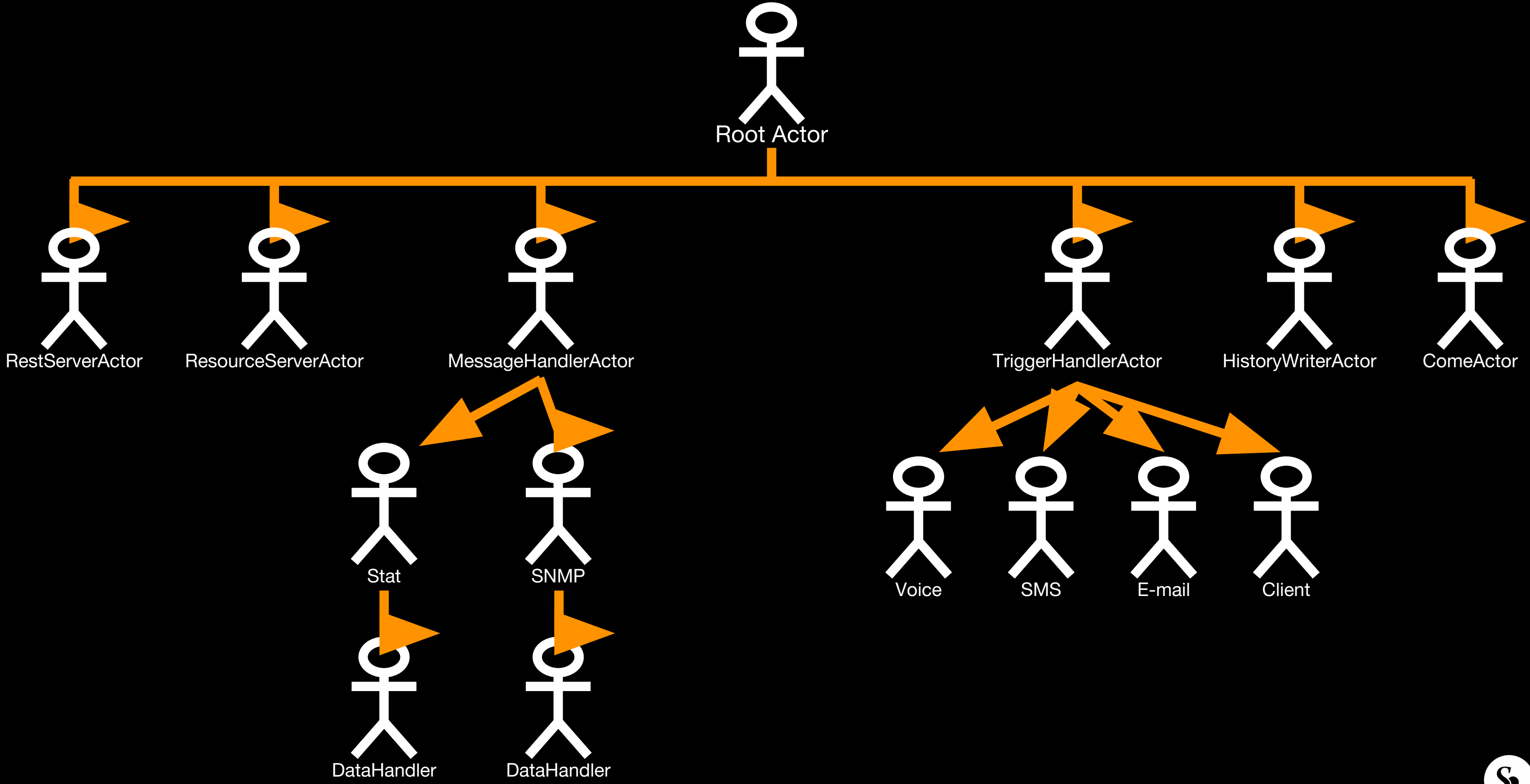
```
$ ps ax | grep node

11031 ?        Sl     0:00 node /tmp/simple-class.js

11041 ?        Sl     0:00 node /tmp/forked-actor-worker.js MyActor

11046 ?        Sl     0:00 node /tmp/forked-actor-worker.js MyActor

11048 ?        Sl     0:00 node /tmp/forked-actor-worker.js MyActor
```

## SCALING WITH ACTORS

> Describe your app in terms of actors (hierarchy)

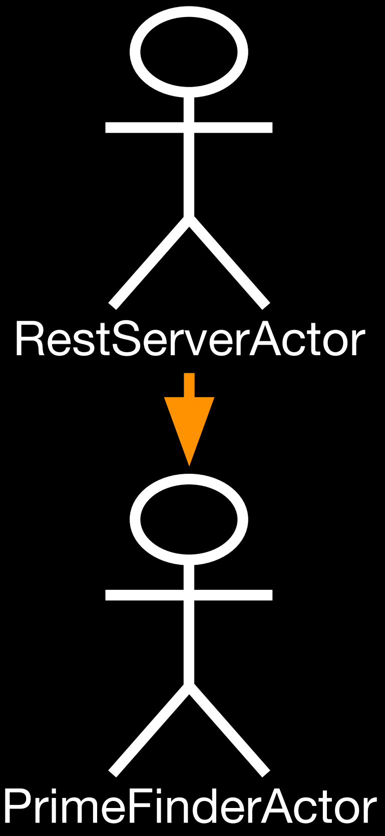
> Configure actors with config file

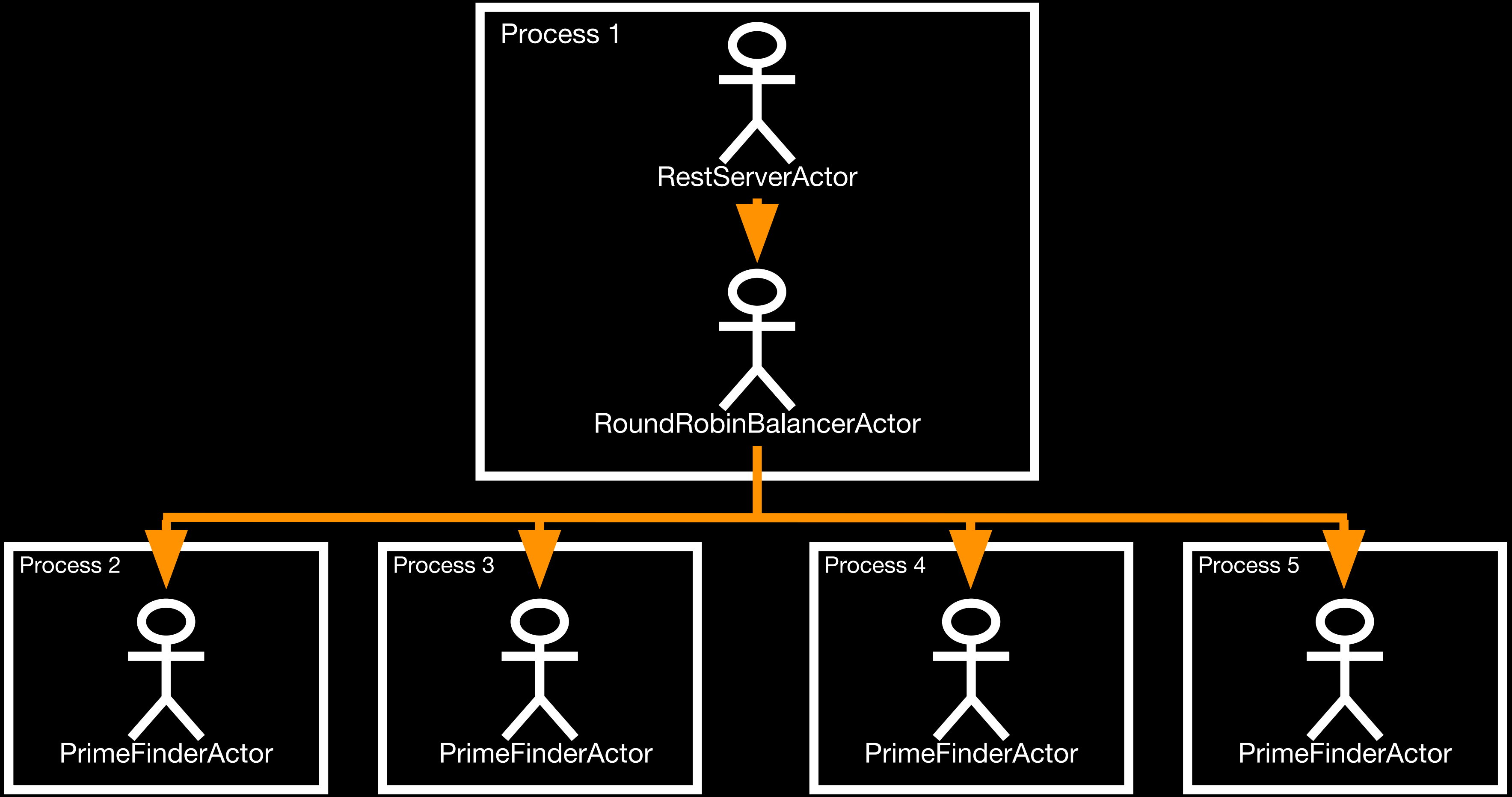
# ACTOR HIERARCHY EXAMPLE

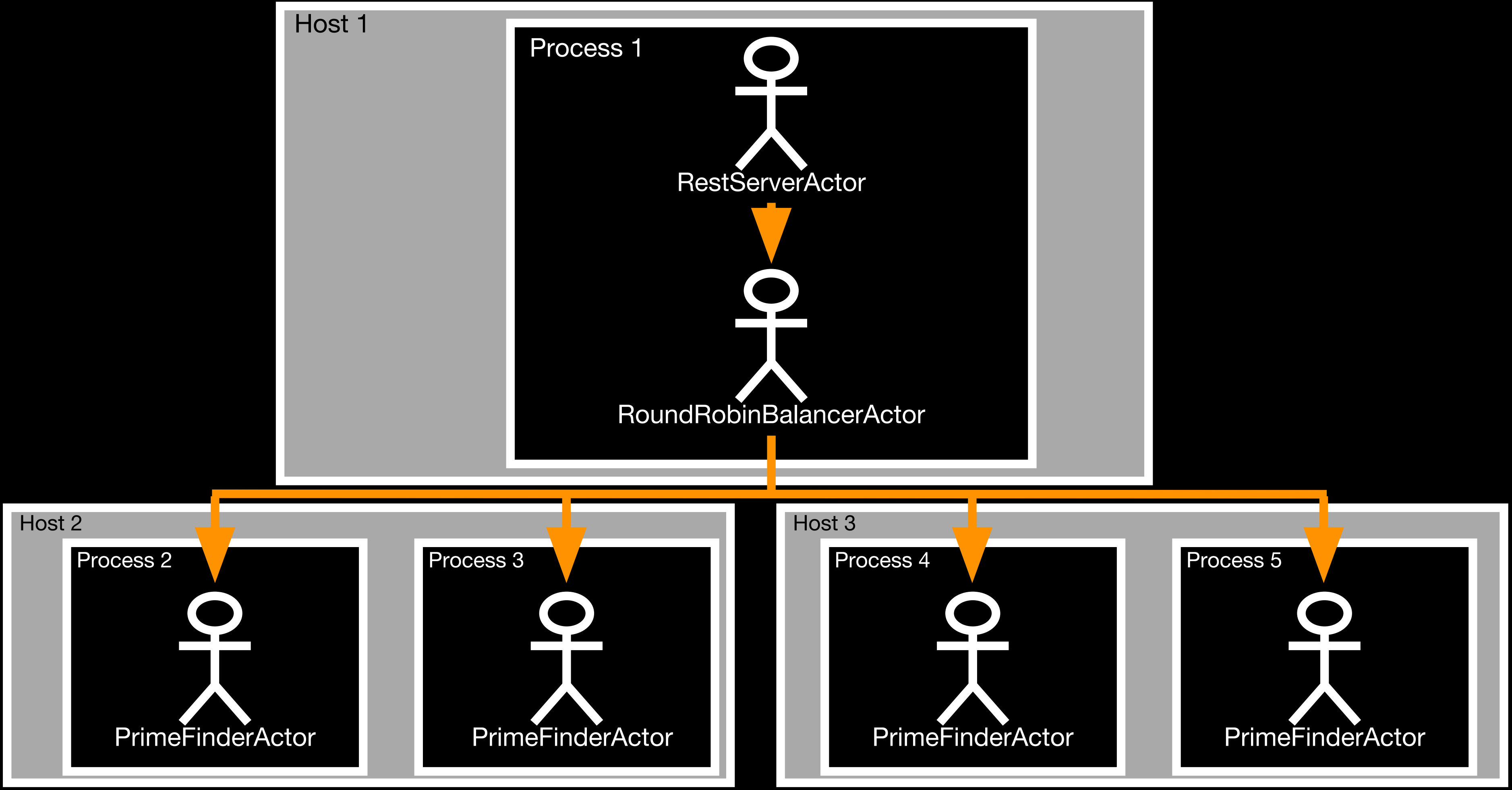


# DEMO

github.com/untu/comedy-presentation


RestServerActor
PrimeFinderActor


Process 1
RestServerActor
RoundRobinBalancerActor
Process 2
PrimeFinderActor
Process 3
PrimeFinderActor
Process 4
PrimeFinderActor
Process 5
PrimeFinderActor


Host 1
Process 1
RestServerActor
RoundRobinBalancerActor
Host 2
Process 2
PrimeFinderActor
Process 3
PrimeFinderActor
Host 3
Process 4
PrimeFinderActor
Process 5
PrimeFinderActor

# Victor Isaev

weekens@gmail.com

github.com/weekens