RICE®

Web Development

COMP 431 / COMP 531

Back-End Testing

Scott E Pollack, PhD

March 17, 2016

The Optical Society of America Rice Chapter Presents:

Dr. Scott Pollack

Two Sigma Investments

"From Black Holes and BECs to Quantitative Finance"

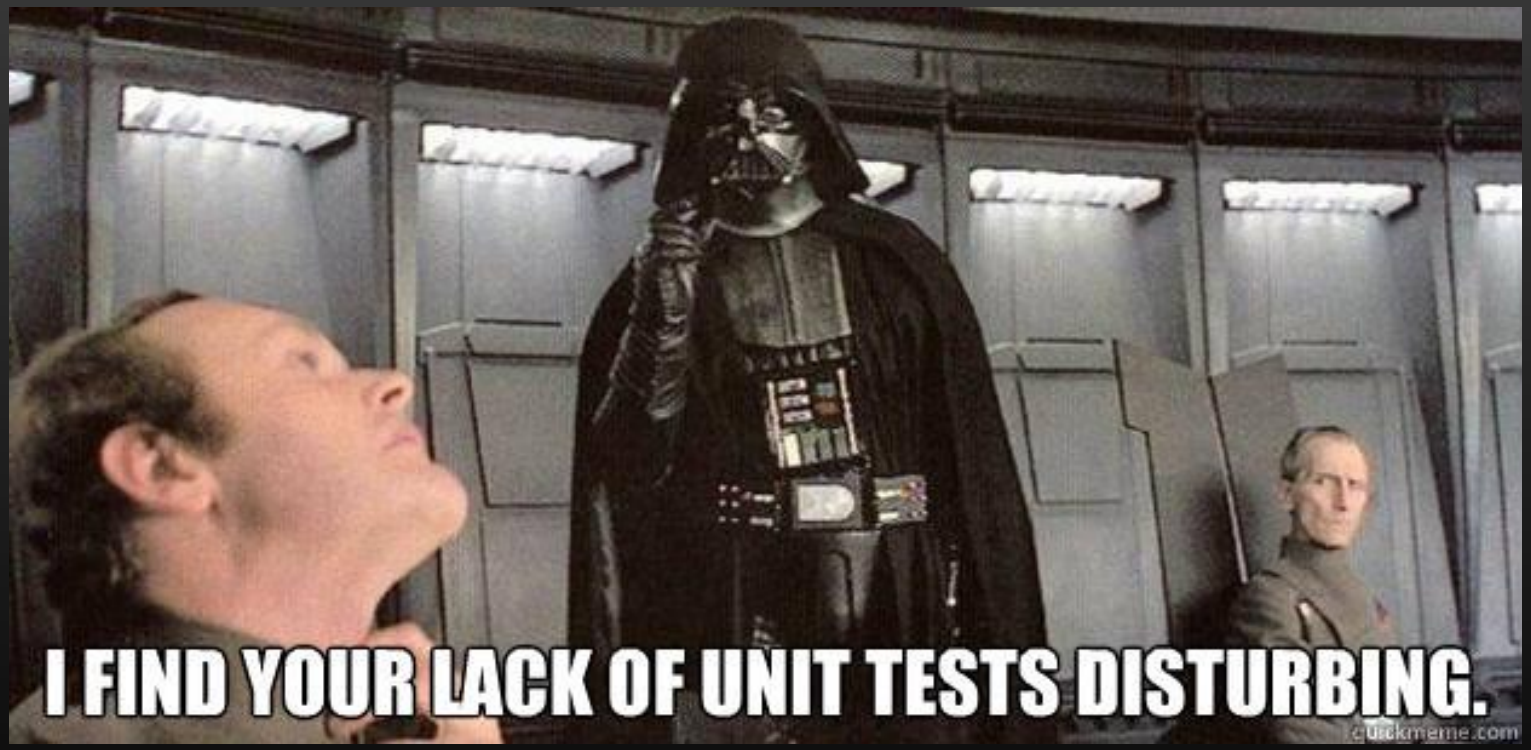
In his talk, Dr Pollack will describe how his background in experimental and computational physics and astrophysics prepared him for a successful career in a quantitative technology-focused industry, opportunities that exist, and some tips to help you succeed.

Thursday, March 17, 2016
Seminar 4:00 PM - 5:00PM
Pizza Social 5:00 PM - 6:00PM
Space Science and Technology, Room 337

RSVP at QR code or:
doodle.com/poll/iih5c5939gfbck76

Bio: Scott Pollack received his Physics Ph.D. in Precision Measurement and Gravitational Astrophysics from the University of Colorado at Boulder in 2005. Following a 3-year

Part II – Back End Development



Part II

~~Web Servers~~

~~Backend~~

~~Architecture~~

Unit Testing

Web Hosting

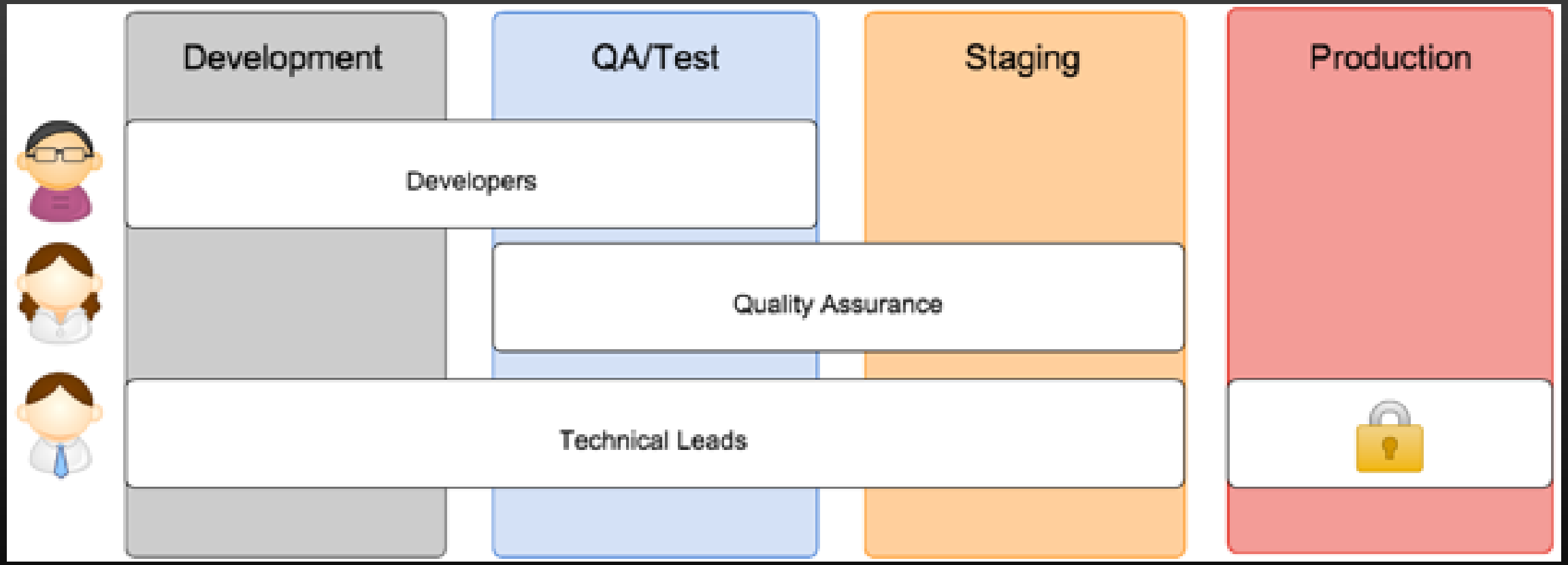
Databases

Homework Assignment 6

(Draft Back-End)

Due Thursday 3/24

Development to Production



http://blog.ashurex.com/2013/11/18/devops-agency/

Development to Production

Development
localhost
Everything's always broken
TDD and/or BDD

QA/Test
localhost or intranet
Continuous integration testing
end-to-end tests

Staging
hosted in wild with staging db
end-to-end test

Production
hosted in wild with prod db
Live with users

http://blog.codeship.com/jasmine-node-js-application-testing-tutorial/

Unit Testing Node: Set Up

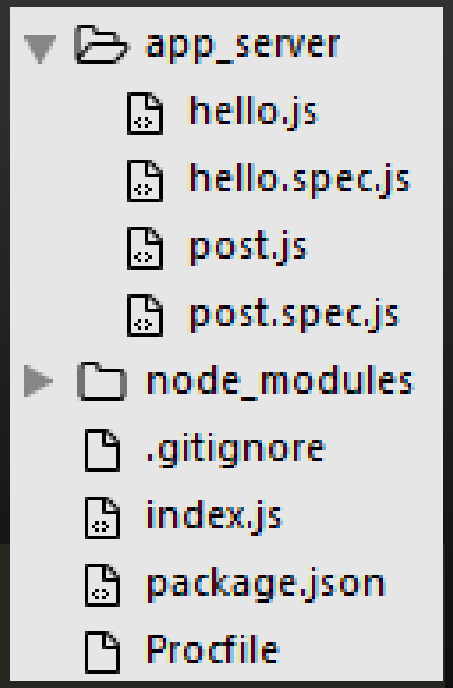
```
> npm install morgan --save
> npm install request jasmine-node --save-dev
```

morgan	logging framework
request	used to make http requests by the server
jasmine-node	nodeified jasmine for the back-end

add to *package.json* some "commands"

```
"scripts": {
  "start": "node index.js",
  "test": "./node_modules/.bin/jasmine-node app_server",
  "autotest": "./node_modules/.bin/jasmine-node --color --autotest app_server"
},
```

directory structure!



Spin up the server and autotester

```
# Do these each in a separate window

> npm start

> npm run autotest
```

Background…

```
1  //*************
2  // Demo Server
3  //*************
4
5  var express = require('express')
6  var bodyParser = require('body-parser')
7  var logger = require('morgan')
8
9  var app = express()
10 app.use(logger("default"))
11 app.use(bodyParser.json({ limit: '10mb' }))
12
13 require('./app_server/post.js').setup(app)
14 require('./app_server/hello.js').setup(app)
15
16 // Get the port from the environment, i.e., Heroku sets it
17 var port = process.env.PORT || 3000
18
19 var server = app.listen(port, function() {
20     console.log('Server listening at http://%s:%s',
21         server.address().address, server.address().port)
22 })
23
```

killnode

404: NOT found

Hello…

hello.js | hello.spec.js | index.js

```
1
2  exports.setup = function(app) {
3      app.get('/:user*?', helloUser)
4  }
5
6  function helloUser(req, res) {
7      var user = req.params.user
8      if (!user) { user = 'Somebody' }
9      res.send('Hello ' + user + '!')
10 }
```

Hello.spec!

hello.js | hello.spec.js | index.js

```
1
2  exports.setup = function(app) {
3      app.get('/:user*?', helloUser)
4  }
5
6  function helloUser(req, res) {
7      var user = req.params.user
8      if (!user) { user = 'Somebody' }
9      res.send('Hello ' + user + '!')
10 }
```

nodenom

hello.js | hello.spec.js | index.js

```
1  /*
2   * Test suite for hello.js
3   */
4  var request = require('request')
5  var hello = require('./hello.js')
6
7  function url(path) {
8      return "http://localhost:3000" + path
9  }
10
11 describe('Validate Hello Functionality', function() {
12
13     it('should say Hello Somebody!', function(done) {
14         request(url("/"), function(err, res, body) {
15             expect(res.statusCode).toBe(200);
16             expect(body).toEquals("Hello Sombody!");
17             done()
18         })
19     }, 200)
20
21     it('should say Hello Me!', function(done) {
22         request(url("/Me"), function(err, res, body) {
23             expect(res.statusCode).toBe(200);
24             expect(body).toEqual("Hello Me!");
25             done()
26         })
27     }, 200)
28 });
```

time out, how long we are going to wait for this tes

200 might be short


Sorry! WE'RE
UNIT TESTING

In-Class Exercise: Unit Test the Back-End

- For reference the example from the slides
  https://www.clear.rice.edu/comp431/sample/RiceBookServer/app_server/hello.spec.js

- Download:
  https://www.clear.rice.edu/comp431/sample/RiceBookServer/app_server/posts.spec.js
- Implement the four tests in *posts.spec.js*
  - We're doing test-driven-development
- Implement in posts.js two endpoints (maybe you already have these)
  GET /posts/:id
  POST /post
- Your implementations should pass the tests

***Turnin* posts.js *and* posts.spec.js**
COMP431-S16:inclass-18