ZERO TO PRODUCTION

WITH NODE.JS

KW@TWILIO.COM

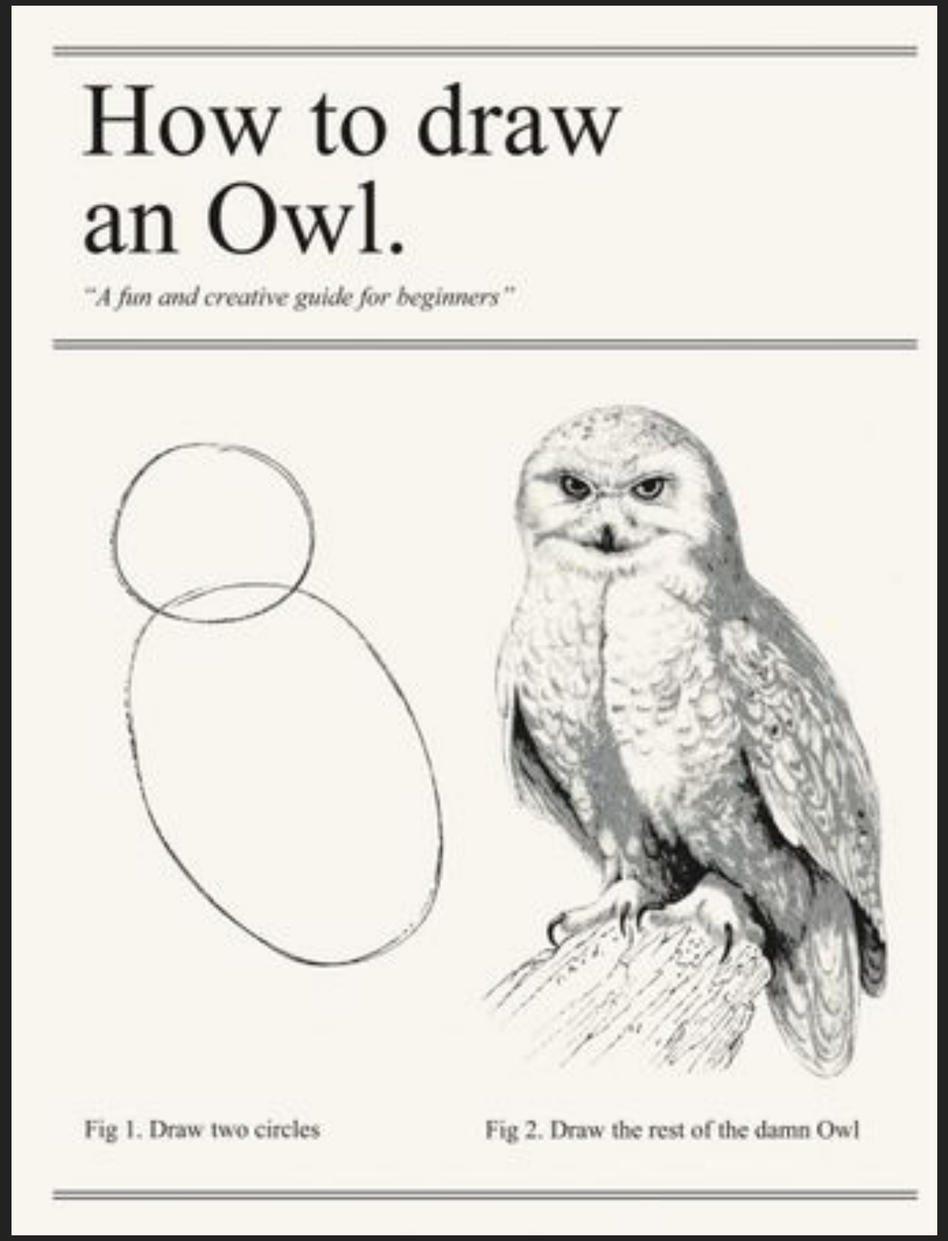
KEVIN WHINNERY

DAY ONE AGENDA

- Introductions
- Building a web server
- Build tools and automation
- Break
- Building the data tier
- Deploying to AWS
- Q&A

DAY TWO AGENDA

- Front end toolchain
- Building the front end with Vue.js
- Real time user interface
- Break
- Production monitoring and load testing
- Web analytics basics
- Q&A

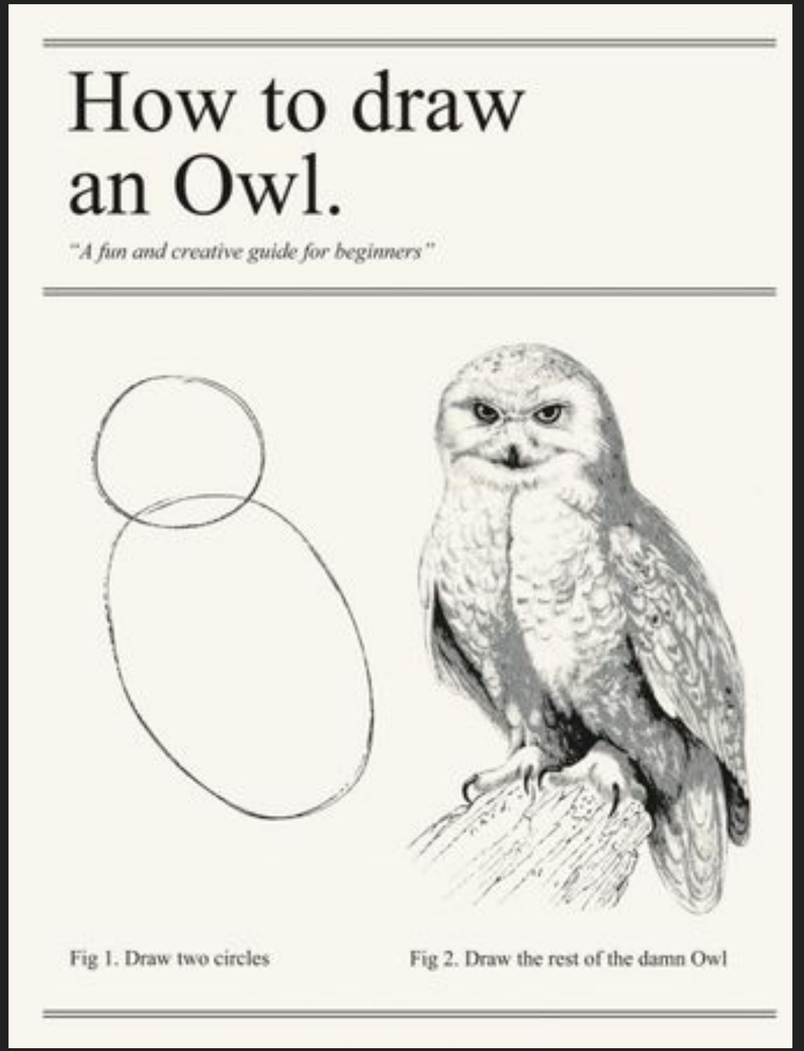


ZERO TO PRODUCTION WITH NODE.JS

INTRODUCTION

GOALS FOR THIS WORKSHOP

- Walk away with a productive starting point that
  - Is productive in development
  - Won’t fall over under modest load
- Mile wide, inch deep
- Exposure to ES2015 (and beyond) features, and tools to use it today
- Take time writing code



WHAT ARE YOUR GOALS?

OUR SAMPLE APPLICATION

- What it does is not important
- Mirrors techniques you might really use
- Doesn’t do everything you might do in production
- Google Style Guide
- Combines a set of technologies that are proven and work well together
- Feel free to riff

EXERCISE 0

SETTING UP SHOP

https://github.com/kwhinnery/todomvc-plusplus

```
'use strict';

const path = require('path');
const express = require('express');
const bodyParser = require('body-parser');
const routes = require('./controllers/routes');

let app = express();

// Configure view engine and views directory
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Configure middleware
app.use(bodyParser.urlencoded({ extended: false }));

// Static file serving happens everywhere but in production
if (process.env.NODE_ENV !== 'production') {
  let staticPath = path.join(__dirname, '..', '..', 'public');
  app.use('/static', express.static(staticPath));
}

// Mount application routes
routes(app);

// Export Express webapp instance
module.exports = app;
```

ZERO TO PRODUCTION WITH NODE.JS

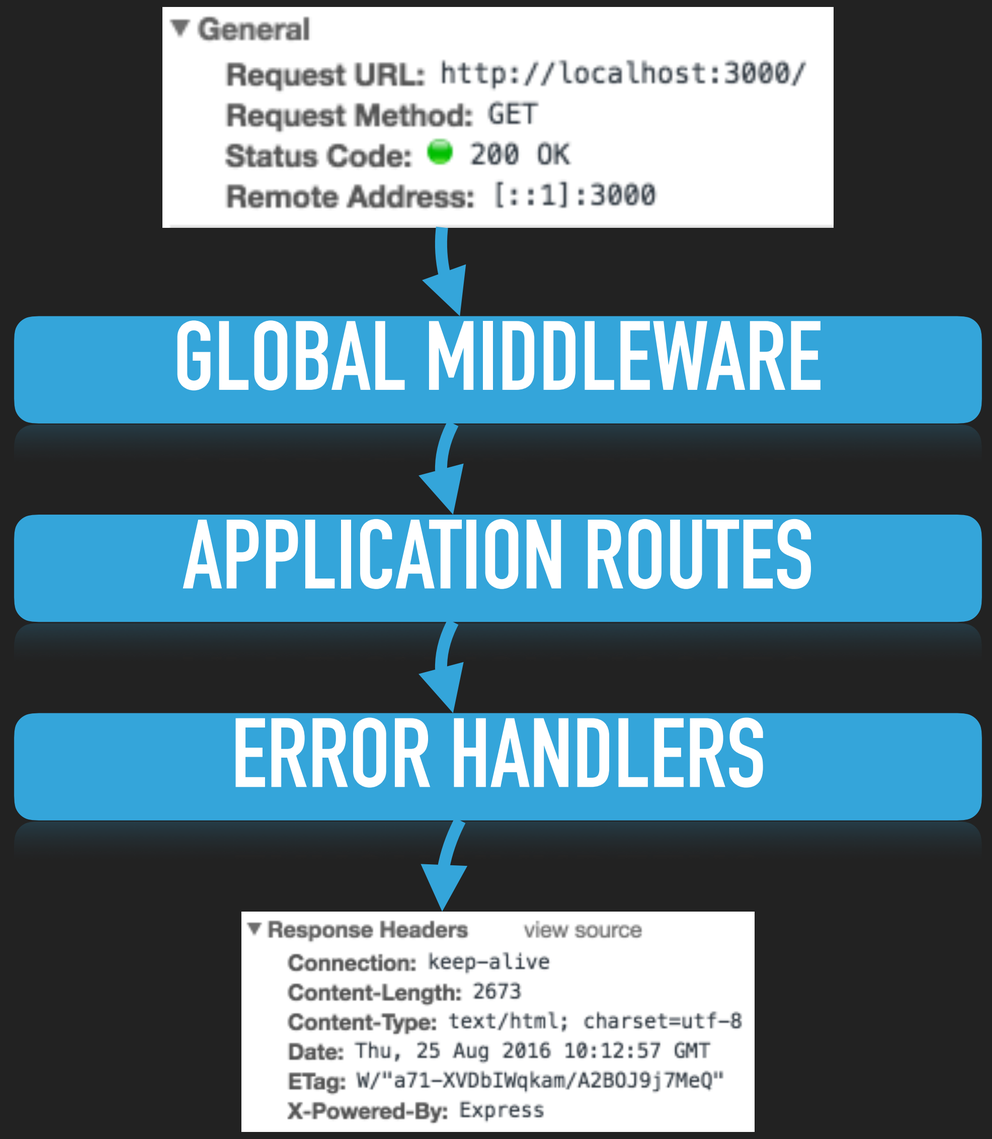
THE WEB SERVER

SERVING HTTP REQUESTS WITH EXPRESS

- Most popular Node.js module for routing HTTP requests to blocks of code in your application
- Batteries not included
- Non-magical

```
1  'use strict';
2
3  const path = require('path');
4  const express = require('express');
5  const bodyParser = require('body-parser');
6  const routes = require('./controllers/routes');
7
8  let app = express();
9
10 // Configure view engine and views directory
11 app.set('view engine', 'ejs');
12 app.set('views', path.join(__dirname, 'views'));
13
14 // Configure middleware
15 app.use(bodyParser.urlencoded({ extended: false }));
16
17 // Static file serving happens everywhere but in production
18 if (process.env.NODE_ENV !== 'production') {
19   let staticPath = path.join(__dirname, '..', '..', 'public');
20   app.use('/static', express.static(staticPath));
21 }
22
23 // Mount application routes
24 routes(app);
25
26 // Export Express webapp instance
27 module.exports = app;
28
```

MIDDLEWARE STACK


▼ General
Request URL: http://localhost:3000/
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:3000
GLOBAL MIDDLEWARE
APPLICATION ROUTES
ERROR HANDLERS
▼ Response Headers view source
Connection: keep-alive
Content-Length: 2673
Content-Type: text/html; charset=utf-8
Date: Thu, 25 Aug 2016 10:12:57 GMT
ETag: W/"a71-XVDbIWqkam/A2BOJ9j7MeQ"
X-Powered-By: Express

THE WEB SERVER

CODE DEMO

ALTERNATIVES TO EXPRESS

- Hapi
- Sails
- Koa

```
'use strict';

const path = require('path');
const express = require('express');
const bodyParser = require('body-parser');
const routes = require('./controllers/routes');

let app = express();

// Configure view engine and views directory
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// Configure middleware
app.use(bodyParser.urlencoded({ extended: false }));

// Static file serving happens everywhere but in production
if (process.env.NODE_ENV !== 'production') {
  let staticPath = path.join(__dirname, '..', '..', 'public');
  app.use('/static', express.static(staticPath));
}

// Mount application routes
routes(app);

// Export Express webapp instance
module.exports = app;
```

EXERCISE 1

HACKING ON EXPRESS

ZERO TO PRODUCTION WITH NODE.JS

BUILD TOOLS

NPM SCRIPTS AND GRUNT

- npm scripts
  - no additional installs/tools needed
  - local binary commands on $PATH
  - conventional
- Grunt tasks
  - mature plugin ecosystem
  - good at synchronous orchestration
  - framework for building tasks that work together

NPM SCRIPTS AND ELASTIC BEANSTALK

- npm scripts used during deploy lifecycle
- limited support for scripts:
  - `start`: create Node process for the instance
  - `prestart`: run prior to start command
  - `poststart`: run after start command

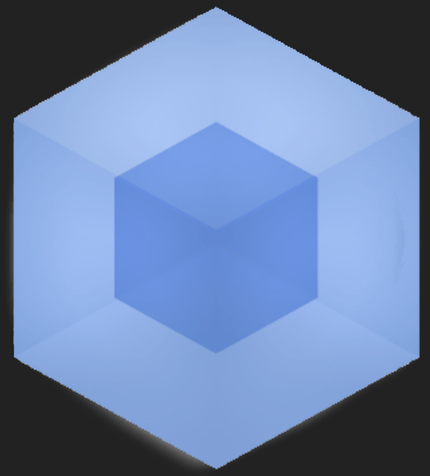
```
{
  "name": "todomvc-plusplus",
  "version": "1.0.1",
  "description": "Expanding on TodoMVC to move it
  "main": "index.js",
  "scripts": {
    "test": "NODE_ENV=test mocha src/server/test",
    "start": "node bin/server.js",
    "prestart": "sequelize db:migrate"
  },
```

BUILD TOOLS

CODE DEMO

ALTERNATIVES AND FRIENDS OF GRUNT

- Alternative: Gulp
  - Streaming tasks
  - Can be faster
  - IMHO: streaming interfaces harder to reason about/write tasks for
- Webpack (works great with Grunt)
  - Lots of awesome front end tricks
  - Not a general purpose task runner



EXERCISE 2

ENHANCING OUR BUILD TOOLS

ZERO TO PRODUCTION WITH NODE.JS

THE DATABASE

SEQUELIZE AND POSTGRESQL

- Sequelize
  - Most feature-rich Node.js ORM
  - Promise-based interface
  - Support for migrations
  - Smaller community, fewer resources
- PostgreSQL
  - Performant and feature-rich
  - Can use with Amazon RDS

ISN'T SQL FOR OLD PEOPLE?

- Possibly.
- Nothing against NoSQL databases
- NoSQL database support is better in Node.js (Mongoose is great)
- Main reasons for choosing SQL:
  - Postgres is awesome
  - RDS is also pretty awesome

THE DATABASE

CODE DEMO

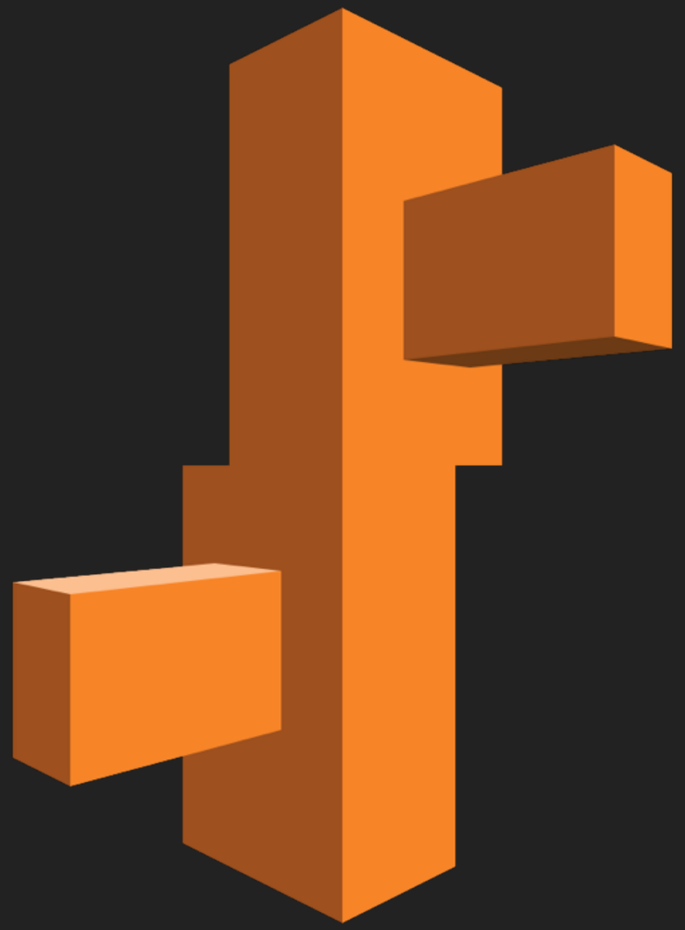
ALTERNATIVES

- MongoDB
  - Natively speaks JSON
  - Very fast
  - No migrations
- Mongoose
  - IMHO: Much better than any similar tool in Node.js right now
  - More resources

mongoose

EXERCISE 3

ENHANCING OUR DATA MODEL

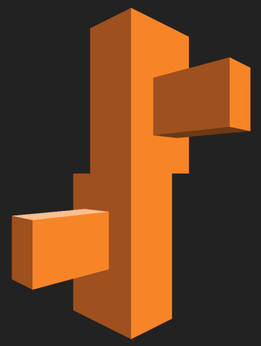


ZERO TO PRODUCTION WITH NODE.JS

PRODUCTION ENVIRONMENT

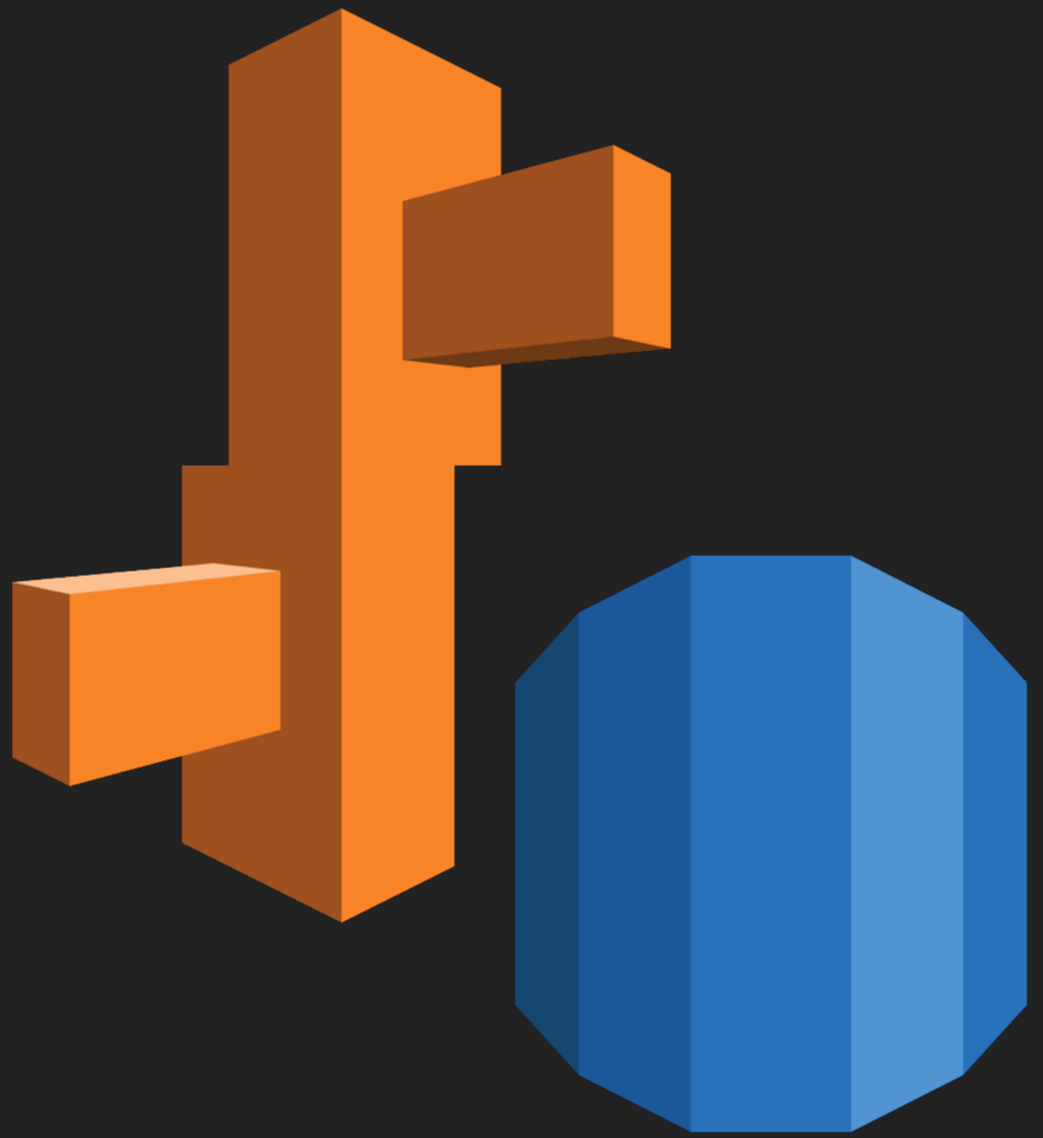

HEROKU

ZERO TO PRODUCTION WITH NODE.JS - PRODUCTION ENVIRONMENT

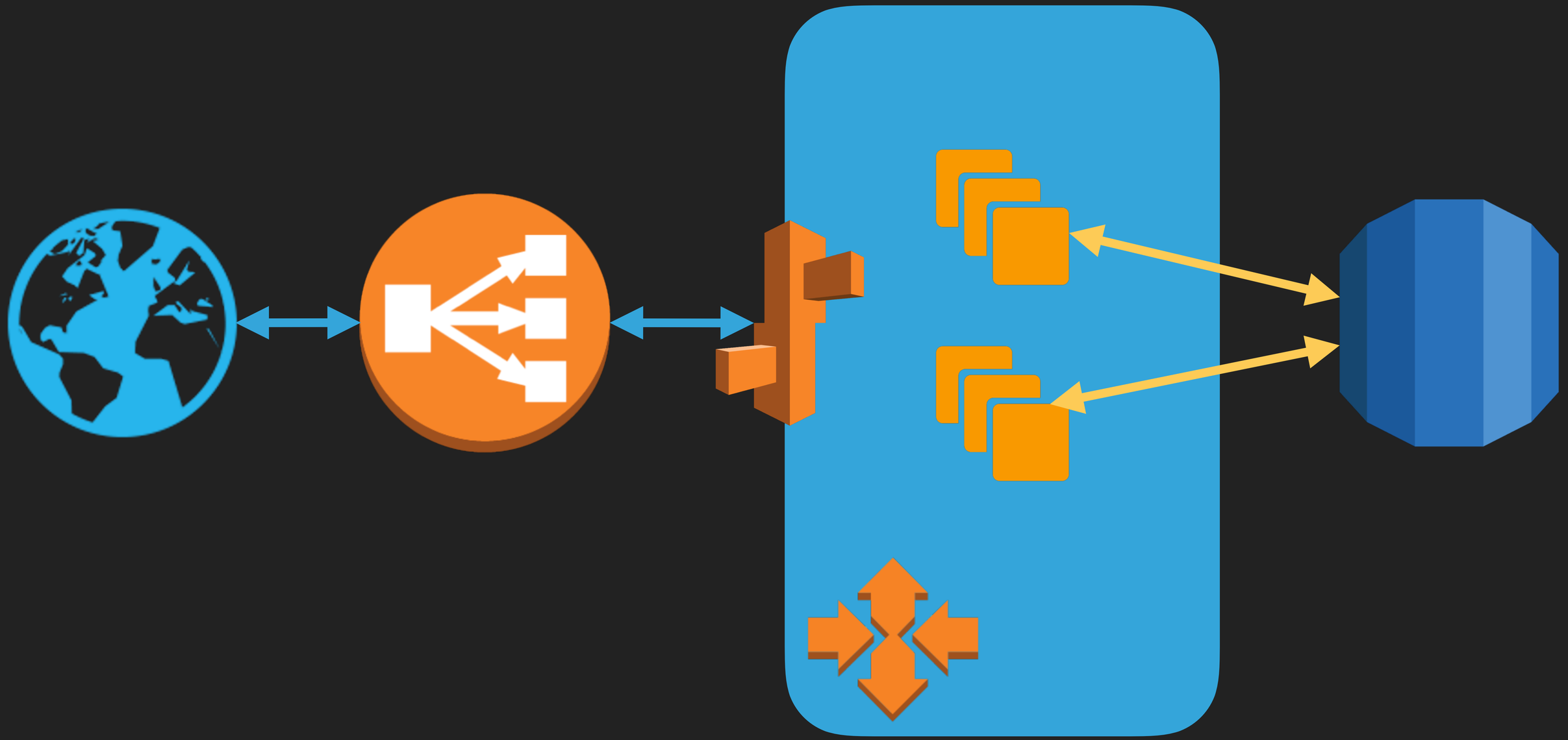


ELASTIC BEANSTALK AND RDS

- Elastic Beanstalk
  - More configurable than most PaaS
  - “Easy” to interact with other Amazon products (RDS, Route 53, S3, etc)
- RDS
  - Performant
  - Highly available
  - Managed updates and snapshots

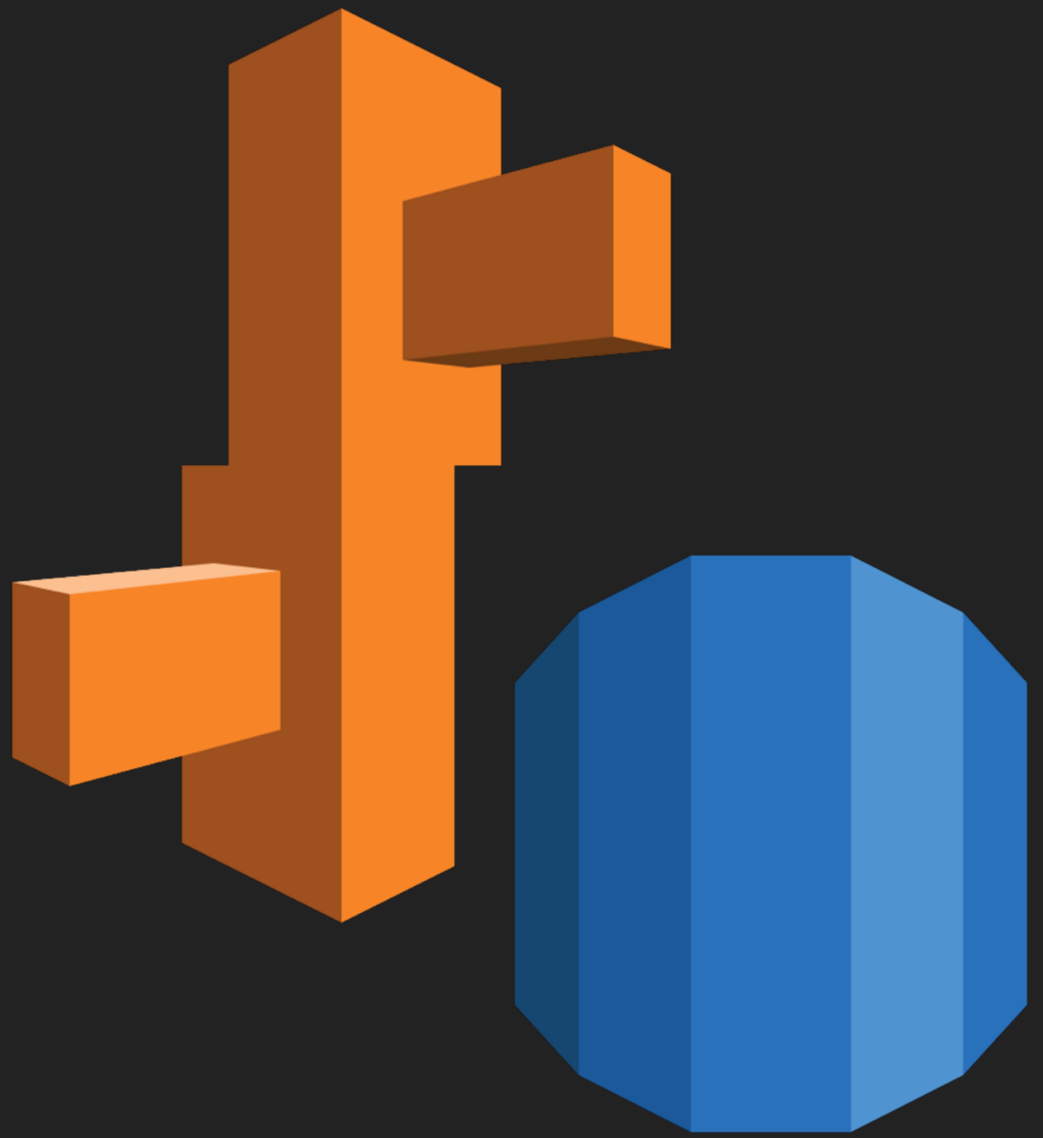


ZERO TO PRODUCTION WITH NODE.JS - PRODUCTION ENVIRONMENT



PROVISIONING AN ENVIRONMENT

- Select an AWS Region
- Create a user with sufficient permissions
- Create an EB environment (will be added to a security group)
- Create an RDS instance (add to same EB security group)
- Configure security group to allow incoming connections to Postgres
- Deploy application version
- Profit?



PRODUCTION ENVIRONMENT

DEPLOY DEMO

ALTERNATIVES

- Heroku / PaaS
  - Fully managed
  - Little DevOps
  - $$$ at scale
- Digital Ocean / EC2
  - Must write own deploy/orchestration software
  - More DevOps
  - Most cost efficient (in terms of compute resources)

EXERCISE 4

DEPLOYING THE APPLICATION