New Deployment

To create a new deployment of this project you will need three primary components: the Node.js application, a MySQL database, and a web server. We decided to use NGINX for our web server but this application will also work with Apache. Optionally, setup continuous integration with something like Travis-CI and setup DNS using Cloudflare for an added layer of protection from online threats like DDOS attacks.

We recommend using a process manager to run the node application. We like (and use) PM2 but you can also use Forever.

We recommend running this application on a VPS like those available from the Duke Innovation Colab or Digital Ocean. We are running our application on a Digital Ocean VPS running Ubuntu 16.04 LTS and would recommend that you use the same version of Ubuntu to mitigate compatibility issues. Once you have the VPS provisioned, go through the basic initial server security configurations (move ssh to a non-standard port, disable root ssh, etc.) and setup MySQL on the VPS. A great guide for installing and configuring MySQL can be found here. Once MySQL is installed and configured, install NGINX following the guide available here.

Connect to your MySQL database through a SQL client such as Sequel Pro and create a new database for the project. Load in the logical MySQL backup of the basic database from this backup here. This initializes the database tables and creates an admin user with username: admin password: initialadminpassword0987654321

The application expects the database and password hashing secret key to be stored in json configuration files in a config folder that is at the same level as the application folder. For example, our /var/www directory looks like this

```
buie@snorkel:/var/www$ ls
config  development  ecosystem.json  html  production  staging
```

where the config folder holds a `config.json` file and a `authconfig.json` file. The files should contain the following:

`config.json`

```
{
  "development": {
    "username": "dev_stage_mysql_username",
    "password": "dev_stage_mysql_password",
    "database": "dev_stage_mysql_database_name",
    "host": "dev_stage_mysql_hostname",
    "dialect": "mysql"
  },
  "staging": {
    "username": "staging_stage_mysql_username",
    "password": "staging_stage_mysql_password",
    "database": "staging_stage_mysql_database_name",
    "host": "staging_stage_mysql_hostname",
    "dialect": "mysql"
  },
  "production": {
    "username": "production_stage_mysql_username",
    "password": "production_stage_mysql_password",
    "database": "production_stage_mysql_database_name",
    "host": "production_stage_mysql_hostname",
    "dialect": "mysql"
  }
}
```

`authconfig.json`

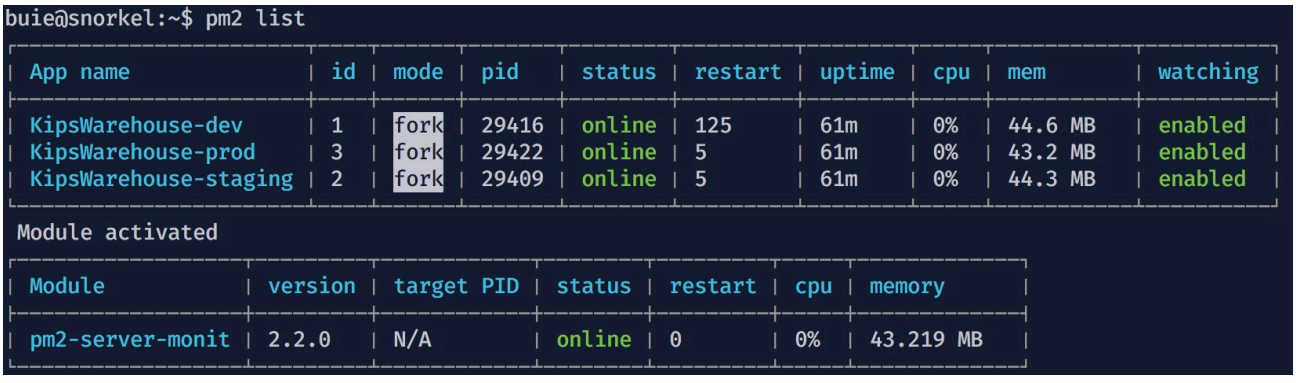
```
{
    "key": "jwt_secret_key"
}
```

Once NGINX and MySQL are running on the VPS, clone the repository into the `/var/www` directory, configure NGINX to point your URL to your newly created folder (likely called KipsWarehouse), and use PM2 to run the application. You can run the application directly using PM2 or use PM2 to manage a set of applications configured through an `ecosystem.json` file. More information about PM2 is available here. Our `ecosystem.json` file is below.

`ecosystem.json`

```
{
  "apps" : [{
    "name"        : "KipsWarehouse-dev",
    "script"      : "bin/www",
    "watch"       : true,
    "cwd"         : "/var/www/development",
    "env": {
      "ENV_STAGE": "dev",
      "NODE_ENV": "development",
      "PORT": "3020"
    }
  },{
    "name"        : "KipsWarehouse-staging",
    "script"      : "bin/www",
    "watch"       : true,
    "cwd"         : "/var/www/staging",
    "env": {
      "ENV_STAGE": "staging",
      "NODE_ENV": "staging",
      "PORT": "3010"
    }
  },{
    "name"        : "KipsWarehouse-prod",
    "script"      : "bin/www",
    "watch"       : true,
    "cwd"         : "/var/www/production",
    "env": {
      "ENV_STAGE": "prod",
      "NODE_ENV": "production",
      "PORT": "3000"
    }
  }]
}
```

After running your application and if all has been configured correctly you should be able to run `pm2 list` and see the status of your current application(s) running like in the photo below.



Here you can see the three instances of our application running. We ran the three instances on three different ports (3000, 3010, and 3020) and then use NGINX to map each subdomain to the correct port.

Working with our Existing Deployments

As mentioned above, our deployed version of this application, that is publicly visible at kipswarehouse.com, is hosted on a Digital Ocean VPS running Ubuntu 16.04 LTS. The DNS routing is handled by Cloudflare and our web server is NGINX. Our continuous deployment infrastructure automatically deploys to our VPS when pull requests are approved into development, staging, or master and handles restarting the node processes on the server. If for some reason the VPS is unexpectedly shutdown it may be necessary to either manually rerun the build/deployment through Travis-CI or to manually start the Node servers on the VPS. Either of these should bring the server back up. The Travis-CI build can be rerun through the web ui at Travis-CI.com. The node servers can be manually restarted by SSHing into the server and running

```
cd /var/www
```

```
pm2 start ecosystem.json
```

Run `pm2 list` to verify that the applications were all able to start. The database automatically starts on boot.

Disaster Recovery

In the event of catastrophic data loss it will be necessary to recover the database from a backups. Each of the databases (production and development) are backed up automatically. We store hourly backups for the last day, daily backups for the last week, weekly backups for the last month, and monthly backups for the last year. These backups are stored as Gzipped logically mysql dumps on an S3 bucket in London.

Download and unzip the backup you want to restore. Connect to your MySQL database through a SQL client such as Sequel Pro and create a new database for the project. Import the backup file you just downloaded. The database should be restored from backup.