deploying meteor with meteor up

reference

- http://code.krister.ee/hosting-multiple-instances-of-meteor-on-digitalocean/
- https://rtcamp.com/tutorials/nodejs/node-js-npm-install-ubuntu/
- https://gentlenode.com/journal/meteor-19-deploying-your-applications-in-a-snap-with-meteor-up-mup/41
- https://www.digitalocean.com/community/tutorials/how-to-set-up-http-authentication-with-nginx-on-ubuntu-12-10
- https://github.com/arunoda/meteor-up

vps

- Memory: 1G+
- Swap: Same as memory
- Disk size: 4G+
- CPUs: 1
- OS: Ubuntu 12.04 LTS

server setup

packages

```
> as root
```

install git, build tools and nginx

```
> apt-get remove -y apache2
> apt-get install -y git build-essential nginx
```

node repo

```
> apt-get install python-software-properties
> apt-add-repository ppa:chris-lea/node.js
> apt-get update
```

node.js/npm install

```
> apt-get install nodejs
```

meteor

```
> apt-get install -y curl
> curl https://install.meteor.com/ | sh
```

workstation setup

mup

install on local development workstation

```
> npm install -g mup
```

app initialization

```
> cd /..../[app_name]/
> mup init
```

- This will create two files: settings.json and mup.json
- Leave settings.js alone, customize mup.json as below
- Add mup.json file to your git ignore settings (global is recommended)

settings.json

```
{
 “public”: {}
}
```

mup.json

```
{
  // Server authentication info
  "servers": [
    {
      "host": "server.com", // Your server fqdn
      "username": "root",
      // pem file (ssh based authentication)
      "pem": "~/.ssh/id_rsa" // need to add your public rsa key on server first
    }
  ],

  // Install MongoDB in the server, does not destroy local MongoDB on future setup
  "setupMongo": true,

  // WARNING: Node.js is required! Only skip if you already have Node.js installed on server.
  "setupNode": false,

  // WARNING: If nodeVersion omitted will setup 0.10.29 by default. Do not use v, only version number.
  "nodeVersion": "0.10.33",

  // Install PhantomJS in the server
  "setupPhantom": true,

  // Application name (No spaces)
  "appName": "[app_name]",

  // Location of app (local directory on your dev workstation)
  "app": "~/..../[app_name]/",

  // Configure environment
  "env": {
    "ROOT_URL": "http://server.com", // your server fqdn
    "PORT": 3001 // change to avoid existing apps conflicting
  },

  // Meteor Up checks if the app comes online just after the deployment
  // before mup checks that, it will wait for no. of seconds configured below
  "deployCheckWaitTime": 15
}
```

preflight

If everything has been set up correctly, the following command in the /[app_name] directory

```
> cd ~/..../[app_name]
> mup setup
```

should yield output similar to this

```
Meteor Up: Production Quality Meteor Deployments
--------------------------------------------


Started TaskList: Setup (linux)
[server.com] - Installing PhantomJS
[server.com] ✔ Installing PhantomJS: SUCCESS
[server.com] - Setting up Environment
[server.com] ✔ Setting up Environment: SUCCESS
[server.com] - Copying MongoDB configuration
[server.com] ✔ Copying MongoDB configuration: SUCCESS
[server.com] - Installing MongoDB
[server.com] ✔ Installing MongoDB: SUCCESS
[server.com] - Configuring upstart
[server.com] ✔ Configuring upstart: SUCCESS
Completed TaskList: Setup (linux)
```

This is only necessary once, before first app deployment.

deploy!

If everything has been set up correctly, the following command in the [app_name] directory

```
cd ~/..../[app_name]
mup deploy
```

should yield output similar to this

```
Meteor Up: Production Quality Meteor Deployments
--------------------------------------------

Building Started: ~/..../[app_name]

Started TaskList: Deploy app '[app_name]' (linux)
[server.com] - Uploading bundle
[server.com] ✔ Uploading bundle: SUCCESS
[server.com] - Setting up Environment Variables
[server.com] ✔ Setting up Environment Variables: SUCCESS
[server.com] - Invoking deployment process
[server.com] ✔ Invoking deployment process: SUCCESS
Completed TaskList: Deploy app '[app_name]' (linux)
```

If so, your meteor app is now running at the specified port. For instance, http://server.com:3001 would work.

Note: Deployment bundles the meteor app using 'meteor bundle' then copies the bundle to /opt/[app_name]/ on the server and cycles the service on the server.

nginx proxying

To respond at normal HTTP port (80) once DNS is configured, configure nginx for the app.

```
/etc/nginx/conf.d/[fqdn].conf:
server {
 listen 80;

 server_name [fqdn];

 location / {
 proxy_pass http://localhost:3001;
 proxy_http_version 1.1;
 proxy_set_header Upgrade $http_upgrade;
 proxy_set_header Connection ‘upgrade’;
 proxy_set_header Host $host;
 proxy_cache_bypass $http_upgrade;
 }
}
```

The important parts are `server_name` and `proxy_pass`.

```
/etc/init.d/nginx restart
```

optional

HTTP auth on site

https://www.digitalocean.com/community/tutorials/how-to-create-a-ssl-certificate-on-nginx-for-ubuntu-12-04

next steps

- firehol to prevent external access to any port other than 22 or 443
- fail2ban or ssh brute force
- run each meteor app under it's own local user account as per https://www.digitalocean.com/community/tutorials/how-to-deploy-a-meteor-js-application-on-ubuntu-14-04-with-nginx