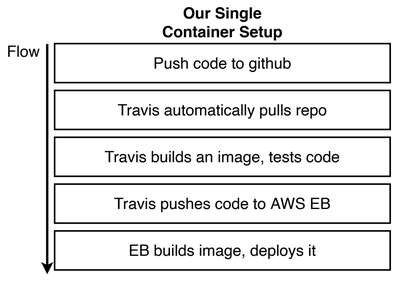
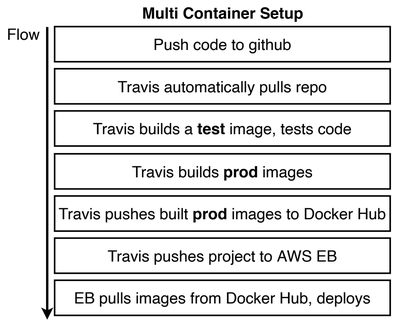
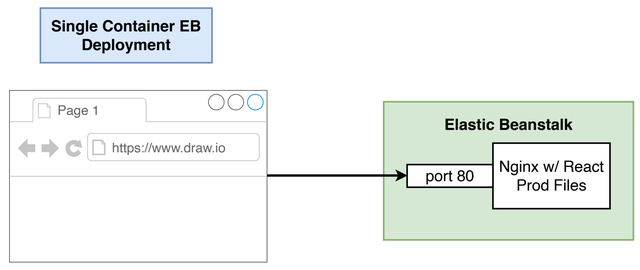
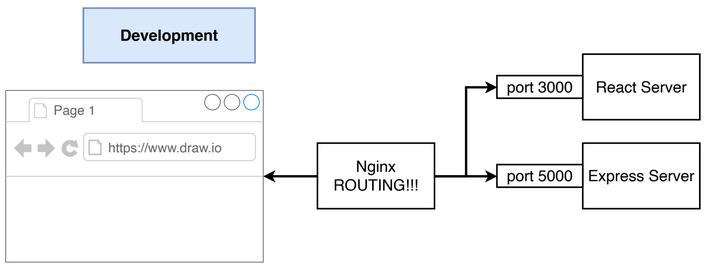
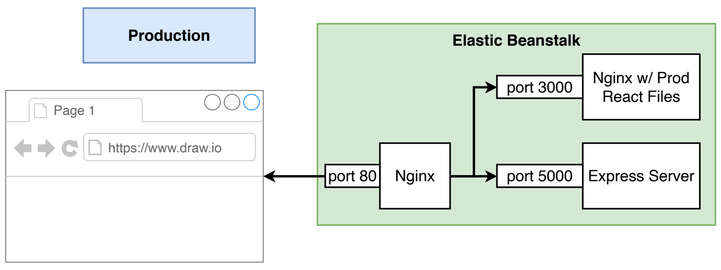
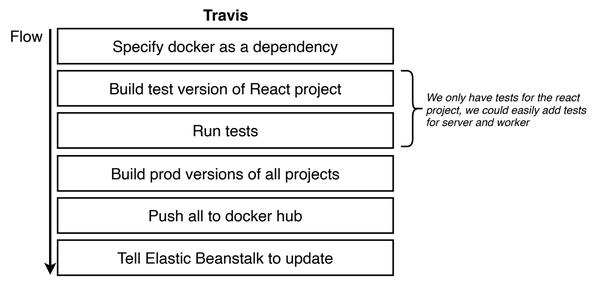

Our Single
Container Setup
Flow
Push code to github
Travis automatically pulls repo
Travis builds an image, tests code
Travis pushes code to AWS EB
EB builds image, deploys it

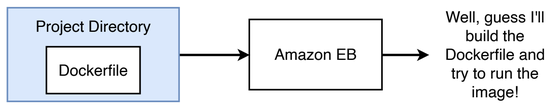

Multi Container Setup
Flow
Push code to github
Travis automatically pulls repo
Travis builds a **test** image, tests code
Travis builds **prod** images
Travis pushes built **prod** images to Docker Hub
Travis pushes project to AWS EB
EB pulls images from Docker Hub, deploys

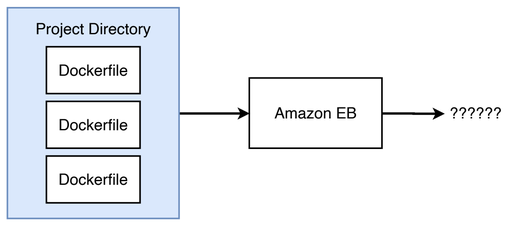

Single Container EB Deployment
Page 1
https://www.draw.io
Elastic Beanstalk
port 80
Nginx w/ React Prod Files

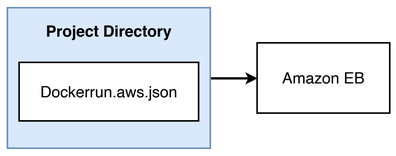

Development
Page 1
https://www.draw.io
Nginx
ROUTING!!!
port 3000
React Server
port 5000
Express Server

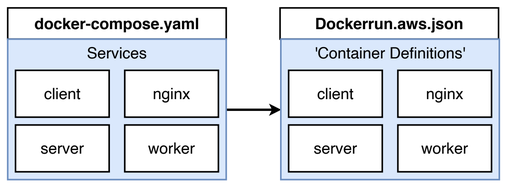

Production
Page 1
https://www.draw.io
Elastic Beanstalk
port 3000
Nginx w/ Prod
React Files
port 80
Nginx
port 5000
Express Server

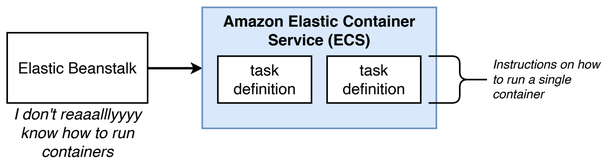

Travis
Flow
Specify docker as a dependency
Build test version of React project
Run tests
Build prod versions of all projects
Push all to docker hub
Tell Elastic Beanstalk to update
We only have tests for the react project, we could easily add tests for server and worker

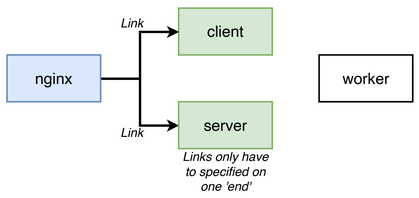

Project Directory
Dockerfile
Amazon EB
Well, guess I'll build the Dockerfile and try to run the image!

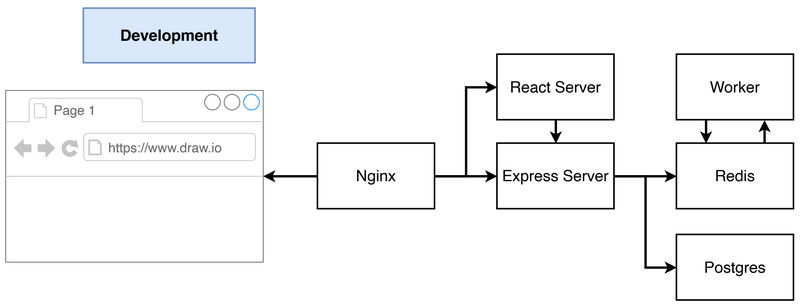

Project Directory
Dockerfile
Dockerfile
Dockerfile
Amazon EB
??????

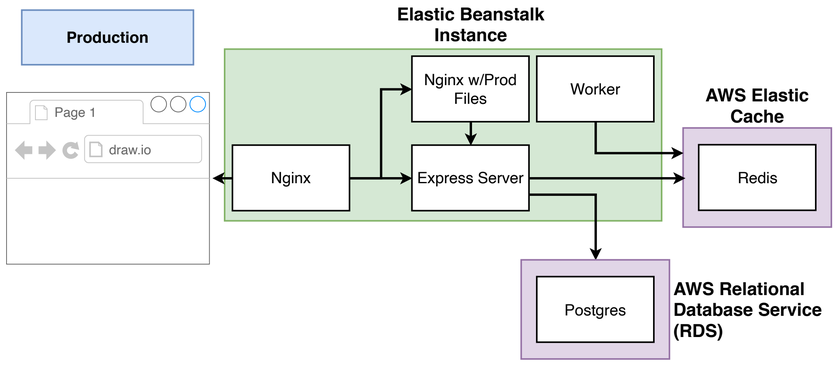

Project Directory
Dockerrun.aws.json
Amazon EB

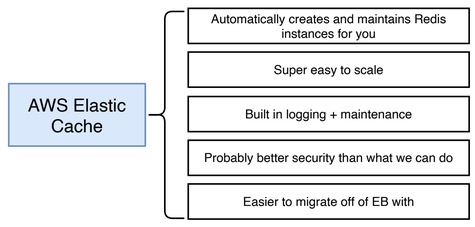

docker-compose.yaml
Services
client
nginx
server
worker
Dockerrun.aws.json
'Container Definitions'
client
nginx
server
worker

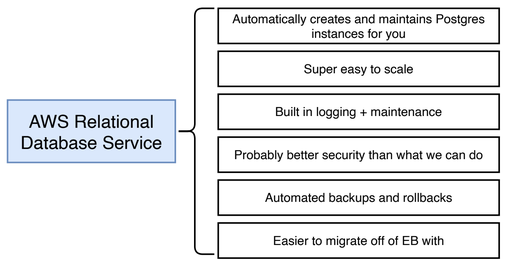

Elastic Beanstalk
I don't reaaalllyyyy know how to run containers
Amazon Elastic Container Service (ECS)
task definition
task definition
Instructions on how to run a single container


Link
client
nginx
worker
server
Link
Links only have to specified on one 'end'


Development
Page 1
https://www.draw.io
Nginx
React Server
Express Server
Worker
Redis
Postgres


Production
Elastic Beanstalk Instance
Page 1
draw.io
Nginx w/Prod Files
Worker
Nginx
Express Server
AWS Elastic Cache
Redis
Postgres
AWS Relational Database Service (RDS)

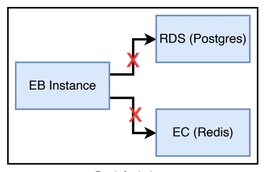

AWS Elastic Cache
Automatically creates and maintains Redis instances for you
Super easy to scale
Built in logging + maintenance
Probably better security than what we can do
Easier to migrate off of EB with


AWS Relational Database Service
Automatically creates and maintains Postgres instances for you
Super easy to scale
Built in logging + maintenance
Probably better security than what we can do
Automated backups and rollbacks
Easier to migrate off of EB with

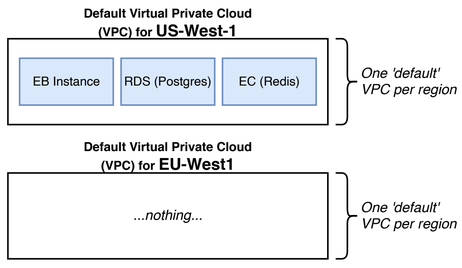
Having said all that....

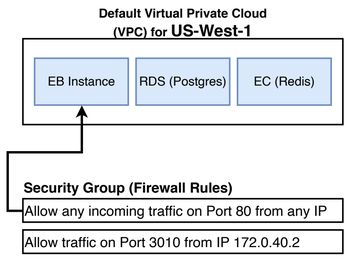
You might still want to know how to get your own DB running in a container in prod

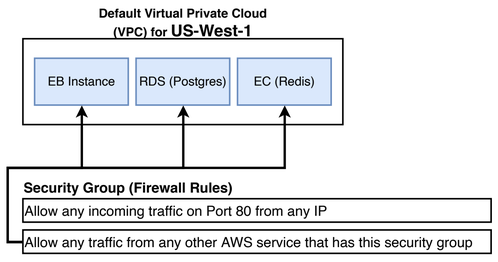
Our next project will use Redis + Postgres in containers



By default these services can't talk to eachother


Default Virtual Private Cloud
(VPC) for US-West-1
EB Instance
RDS (Postgres)
EC (Redis)
One 'default'
VPC per region
Default Virtual Private Cloud
(VPC) for EU-West1
...nothing...
One 'default'
VPC per region


Default Virtual Private Cloud
(VPC) for US-West-1
EB Instance
RDS (Postgres)
EC (Redis)
Security Group (Firewall Rules)
Allow any incoming traffic on Port 80 from any IP
Allow traffic on Port 3010 from IP 172.0.40.2


Default Virtual Private Cloud
(VPC) for US-West-1
EB Instance
RDS (Postgres)
EC (Redis)
Security Group (Firewall Rules)
Allow any incoming traffic on Port 80 from any IP
Allow any traffic from any other AWS service that has this security group