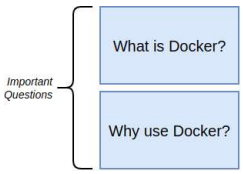
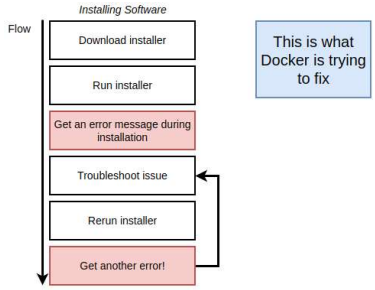
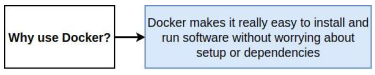
Docker

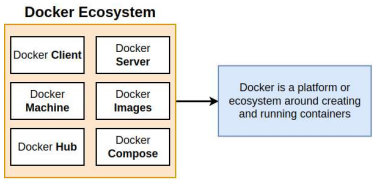
Sunday, June 22, 2025 7:21 PM



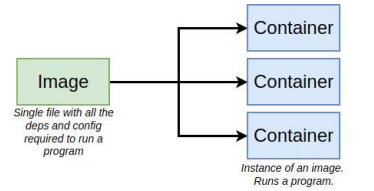


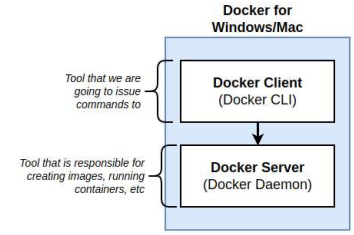

```
docker run -it redis
```



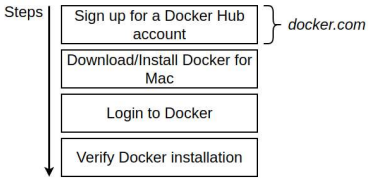


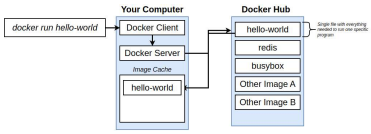

```
➜  prod git:(master) docker version
Client:
 Version:           18.06.0-ce
 API version:       1.38
 Go version:        go1.10.3
 Git commit:        0ffa825
 Built:             Wed Jul 18 19:05:26 2018
 OS/Arch:           darwin/amd64
 Experimental:      false

Server:
 Engine:
  Version:          18.06.0-ce
  API version:      1.38 (minimum version 1.12)
  Go version:       go1.10.3
  Git commit:       0ffa825
  Built:            Wed Jul 18 19:13:46 2018
  OS/Arch:          linux/amd64
```

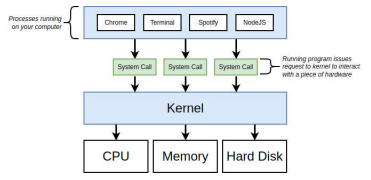


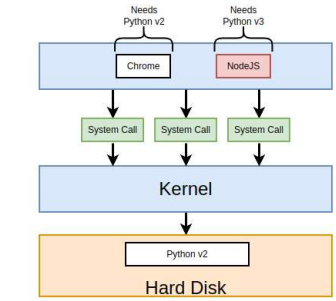


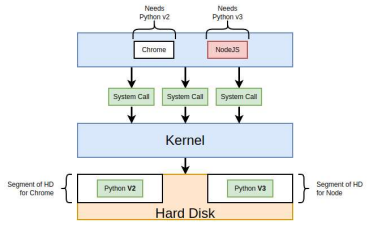

```
➜  prod git:(master) docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7af
acdc
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs th
```

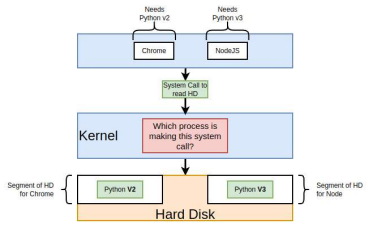


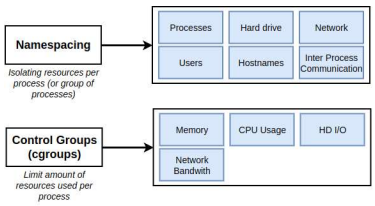


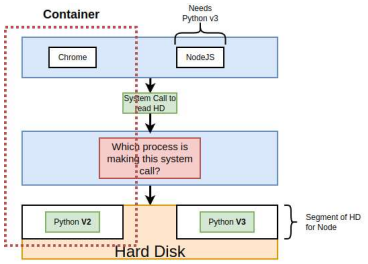


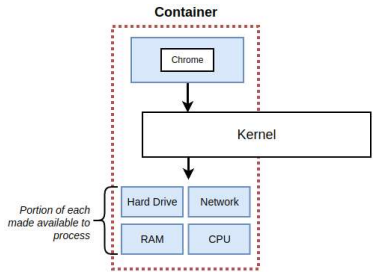

Needs Python v2
Needs Python v3
Chrome
NodeJS
System Call
System Call
System Call
Kernel
Python v2
Hard Disk


Needs Python v2
Needs Python v3
Chrome
NodeJS
System Call
System Call
System Call
Kernel
Segment of HD for Chrome
Python V2
Python V3
Segment of HD for Node
Hard Disk


Needs Python v2
Needs Python v3
Chrome
NodeJS
System Call to read HD
Kernel
Which process is making this system call?
Segment of HD for Chrome
Python V2
Python V3
Segment of HD for Node
Hard Disk


Namespacing
Isolating resources per process (or group of processes)
Processes
Hard drive
Network
Users
Hostnames
Inter Process Communication
Control Groups (cgroups)
Limit amount of resources used per process
Memory
CPU Usage
HD I/O
Network Bandwith


Container
Needs Python v3
Chrome
NodeJS
System Call to read HD
Which process is making this system call?
Python V2
Python V3
Segment of HD for Node
Hard Disk


Container
Chrome
Kernel
Portion of each made available to process
Hard Drive
Network
RAM
CPU

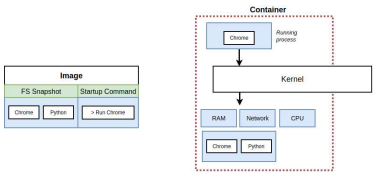
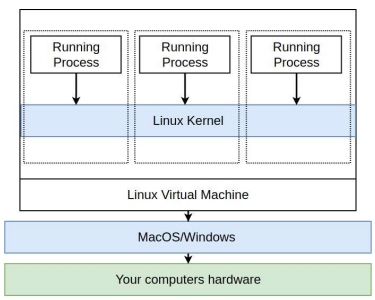
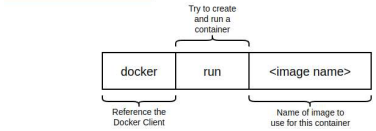
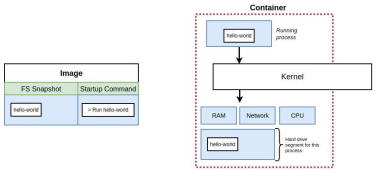


Image
FS Snapshot
Startup Command
Chrome
Python
> Run Chrome
Container
Chrome
Running process
Kernel
RAM
Network
CPU
Chrome
Python


Running Process
Running Process
Running Process
Linux Kernel
Linux Virtual Machine
MacOS/Windows
Your computers hardware

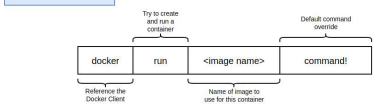
Creating and Running a Container from an Image

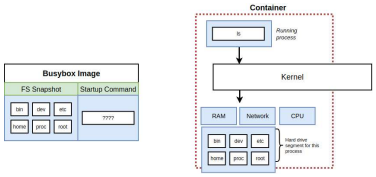




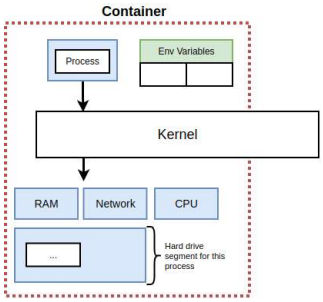

```
➜  prod git:(master) docker run busybox echo hi there
hi there
➜  prod git:(master) docker run busybox echo bye there
bye there
➜  prod git:(master) docker run busybox echo how are you
how are you
➜  prod git:(master)
```

Creating and Running a Container from an Image



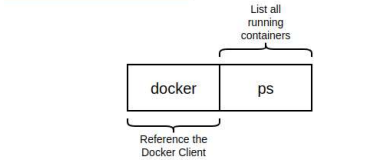



```
➜  prod git:(master) docker run busybox ls
bin
dev
etc
home
proc
root
sys
tmp
usr
var
➜  prod git:(master)
```




```
➜  prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
14aded86a0cd        busybox             "ping google.com"   25 seconds ago      Up 24 seconds                           epic_cori
➜  prod git:(master)
```

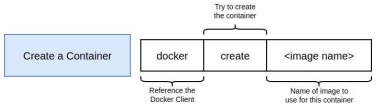
List all running containers




```
➜  prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
14aded86a0cd        busybox             "ping google.com"   25 seconds ago      Up 24 seconds                           epic_cori
➜  prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
➜  prod git:(master) docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
14aded86a0cd        busybox             "ping google.com"   About a minute ago  Exited (0) 30 seconds ago                       epic_cori
38515dd930b8        busybox             "echo hi there"     2 minutes ago       Exited (0) 2 minutes ago                        adoring_benz
715c4da179f0        busybox             "ping google.com"   6 minutes ago       Exited (0) 3 minutes ago                        cranky_ramanujan
8225c260befe        busybox             "echo hi there"     6 minutes ago       Exited (0) 6 minutes ago                        silly_cori
a4bc76929bcc        busybox             "sh"                38 minutes ago      Exited (0) 37 minutes ago                       quizzical_pasteur
0cf904079d76        busybox             "echo $hi"          45 minutes ago      Exited (0) 45 minutes ago                       mystifying_agnesi
2a9cc98827f0        busybox             "echo $hi"          45 minutes ago      Exited (0) 45 minutes ago                       wonderful_mcnulty
51601d3e0c2f        busybox             "echo $hi"          About an hour ago   Exited (0) About an hour ago                    stoic_pike
a5a736795b38        busybox             "echo bye"          About an hour ago   Exited (0) About an hour ago                    relaxed_mendeleev
0e14a36d7dcd        busybox             "echo bye"          About an hour ago   Exited (0) About an hour ago                    pedantic_swirles
42b183c7575d        busybox             "echo"              About an hour ago   Exited (0) About an hour ago                    pensive_jepsen
5b56a5aa310f        busybox             "echo MERGE_HEAD"   About an hour ago   Exited (0) About an hour ago                    mystifying_noether
f98a89c9e136        busybox             "echo"              About an hour ago   Exited (0) About an hour ago                    keen_torvalds
75d318ff6c52        busybox             "echo"              About an hour ago   Exited (0) About an hour ago                    compassionate_yonath
cdf7e291997e        busybox             "echo bye"          About an hour ago   Exited (0) About an hour ago                    wonderful_keller
c2d9a91b55b7        busybox             "echo"              About an hour ago   Exited (0) About an hour ago                    heuristic_nightingale
8e2fdacfa735        busybox             "echo none normal bg…"   About an hour ago   Exited (0) About an hour ago                    kind_nobel
2b526a85586f        busybox             "echo bye"          About an hour ago   Exited (0) About an hour ago                    modest_minsky
```

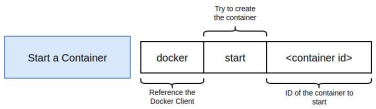
docker **run** = docker **create** + docker **start**

Create a Container

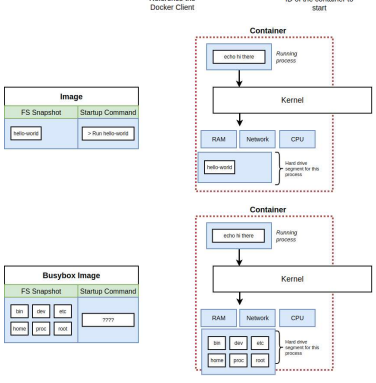



```
➜  prod git:(master) docker create hello-world
955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734ec95358f7ff4fec
➜  prod git:(master) docker start 955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734e
c95358f7ff4fec
955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734ec95358f7ff4fec
➜  prod git:(master) docker start -a 955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e7
34ec95358f7ff4fec
```

Start a Container

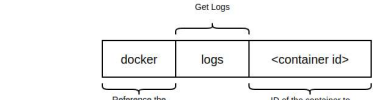



```
➜  prod git:(master) docker run busybox echo hi there
hi there
➜  prod git:(master) docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
  NAMES
37a8c45a23e1        busybox             "echo hi there"     4 seconds ago       Exited (0) 3 s
  pedantic_jepsen
8849c62dd8d5        busybox             "echo hi there"     About a minute ago  Exited (0) Abou
  thirsty_northcutt
➜  prod git:(master) docker start 37a8c45a23e1
37a8c45a23e1
➜  prod git:(master) docker start -a 37a8c45a23e1
hi there
➜  prod git:(master) docker start -a 37a8c45a23e1 echo bye there
you cannot start and attach multiple containers at once
➜  prod git:(master)
```



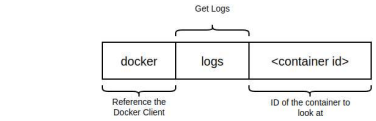

```
➜  prod git:(master) docker system prune
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N] y
Deleted Containers:
37a8c45a23e15afd71684ad0c8e26e328661c5dc0855609bbd66b861b0fb0d8e
8849c62dd8d512ab0485b174db5f4ca13f3fcc03fbe1de35324e77b494b6ed69

Total reclaimed space: 0B
➜  prod git:(master) docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED             STAT
US                  PORTS               NAMES
➜  prod git:(master) docker system prune
```

Get logs from a container

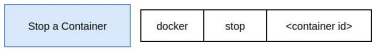



```
➜  prod git:(master) docker create busybox echo hi there
a2d69ef73f4dfade7670e093ee8a406116690cbb907cc9007f99de2b56100be1
➜  prod git:(master) docker start a2d69ef73f4dfade7670e093ee8a4061166
99de2b56100be1
a2d69ef73f4dfade7670e093ee8a406116690cbb907cc9007f99de2b56100be1
➜  prod git:(master) docker logs a2d69ef73f4dfade7670e093ee8a4061166
9de2b56100be1
hi there
➜  prod git:(master)
```

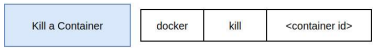
```
➜  prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STAT
```

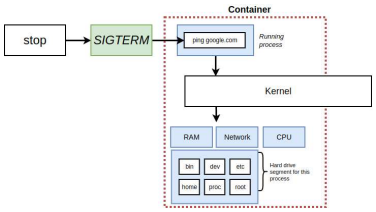


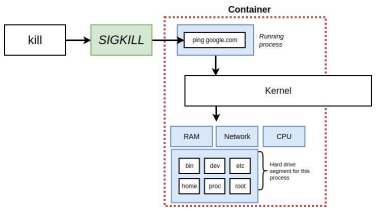

```
a2d69ef73f4dfade7670e093ee8a406116690cbb907cc9007f99de2b56100be1
➜  prod git:(master) docker logs a2d69ef73f4dfade7670e093ee8a406116690
9de2b56100be1
hi there
➜  prod git:(master)
```

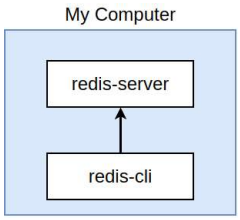



```
➜  prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
a8f48b4dc69a        busybox             "ping google.com"   3 minutes ago       Up 2 minutes                            quizzical_meninsky
➜  prod git:(master) docker stop a8f48b4dc69a
a8f48b4dc69a
➜  prod git:(master) docker start a8f48b4dc69a
a8f48b4dc69a
➜  prod git:(master) docker kill a8f48b4dc69a
a8f48b4dc69a
➜  prod git:(master)
```

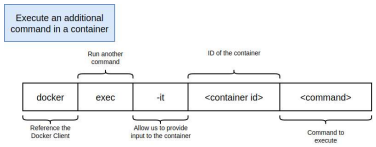


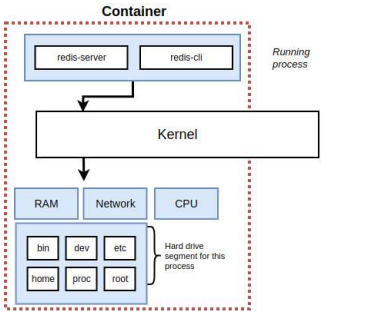




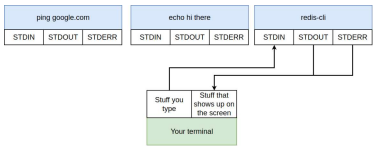



```
➜  prod git:(master) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
093b6e72ff2b        redis               "docker-entrypoint.s…"   3 minutes ago       Up 3 minutes        6379/tcp            jovial_leakey
➜  prod git:(master) ✗ docker exec -it 093b6e72ff2b redis-cli
127.0.0.1:6379> set myvalue 5
OK
127.0.0.1:6379> get myvalue
"5"
127.0.0.1:6379>
➜  prod git:(master) ✗
➜  prod git:(master) ✗ docker exec 093b6e72ff2b redis-cli
➜  prod git:(master) ✗
```

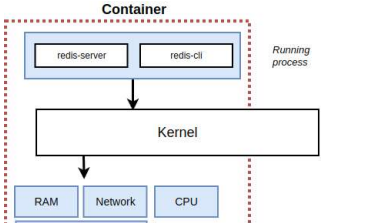


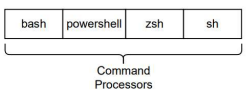



```
➜  prod git:(master) ✗ docker exec -i -t
➜  prod git:(master) ✗
➜  prod git:(master) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
093b6e72ff2b        redis               "docker-entrypoint.s…"   13 minutes ago      Up 13 minutes       6379/tcp            jovial_leakey
➜  prod git:(master) ✗ docker exec -i 093b6e72ff2b redis-cli
set myvalue 5
OK
get myvalue
5
```

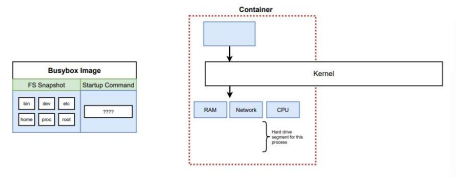


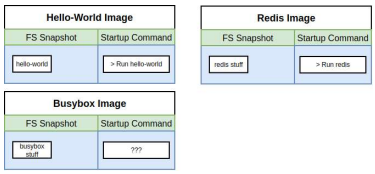

```
➜  prod git:(master) docker exec -it 4e3d15293585 sh
# cd ~/
# ls
# cd /
# ls
bin   data  etc   lib   media  opt   root  sbin  sys  usr
boot  dev   home  lib64 mnt    proc  run   srv   tmp  var
# echo hi there
hi there
# export b=5
# echo $b
5
# redis-cli
127.0.0.1:6379>
```

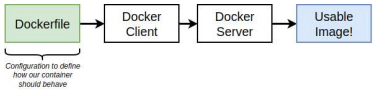




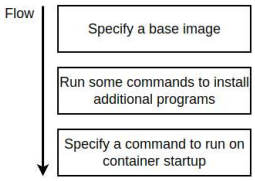

```
➜  prod git:(master) docker run -it busybox sh
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # ping google.com
PING google.com (172.217.11.238): 56 data bytes
```







Creating a Dockerfile

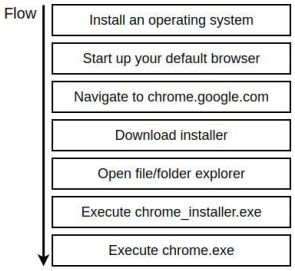


Create an image that runs redis-server

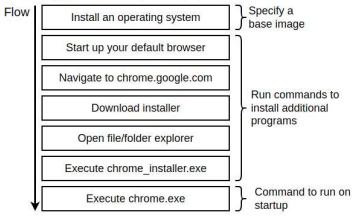
Goal

Writing a dockerfile == Being given a computer with no OS and being told to install Chrome

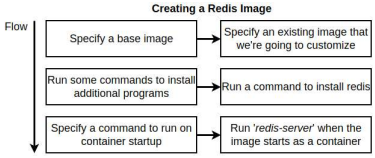
How do you install Chrome *on a computer with no operating system*?



How do you install Chrome *on a computer with no operating system*?



Creating a Redis Image



Why did we use alpine as a base image?

↓

Why do you use Windows, MacOS, or Ubuntu?

↓

They come with a preinstalled set of programs that are useful to you!

```
prod git:(master) mkdir redis-image
prod git:(master) cd redis-image
redis-image git:(master) code .
redis-image git:(master)
```

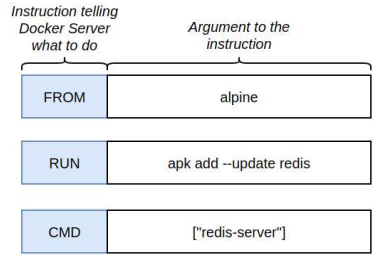
```
1  # Use an existing docker image as a base
2  FROM alpine
3
4  # Download and install a dependency
5  RUN apk add --update redis
6
7  # Tell the image what to do when it starts
8  # as a container
9  CMD ["redis-server"]
```

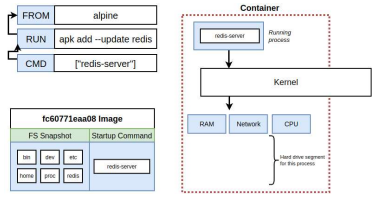
```
➜  prod git:(master) mkdir redis-image
➜  prod git:(master) cd redis-image
➜  redis-image git:(master) code .
➜  redis-image git:(master) docker build .
Sending build context to Docker daemon  2.048kB
Step 1/3 : FROM alpine
latest: Pulling from library/alpine
8e3ba11ec2a2: Pull complete
Digest: sha256:7043076348bf5040220df6ad703798fd8
93be117e430
Status: Downloaded newer image for alpine:latest
 ---> 11cd0b38bc3c
Step 2/3 : RUN apk add --update redis
```

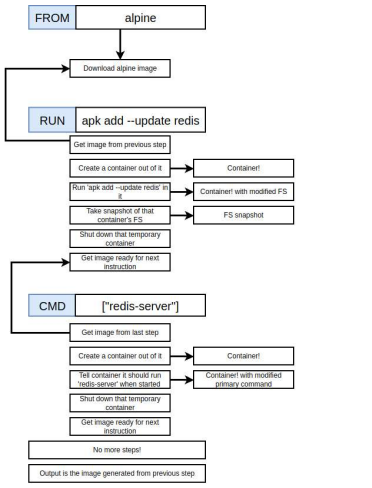
```
➜  redis-image git:(master) ✗ docker build .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM alpine
 ---> 11cd0b38bc3c
Step 2/4 : RUN apk add --update redis
 ---> Using cache
 ---> 38ec9aea7e10
Step 3/4 : RUN apk add --update gcc
 ---> Running in 8340604fcfb4
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/m
.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/c
INDEX.tar.gz
(1/11) Installing binutils (2.30-r5)
(2/11) Installing gmp (6.1.2-r1)
```

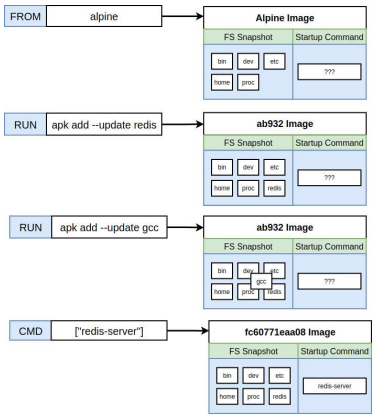
```
1  # Use an existing docker image as a base
2  FROM alpine
3
4  # Download and install a dependency
5  RUN apk add --update gcc
6  RUN apk add --update redis
7
8
9  # Tell the image what to do when it starts
10 # as a container
11 CMD ["redis-server"]
```

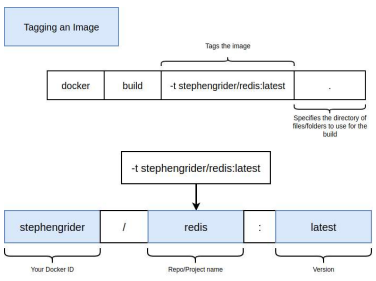
```
➜  redis-image git:(master) ✗ docker build .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM alpine
 ---> 11cd0b38bc3c
Step 2/4 : RUN apk add --update gcc
 ---> Using cache
 ---> d409bde64c12
Step 3/4 : RUN apk add --update redis
 ---> Using cache
 ---> 3266f6626ef3
Step 4/4 : CMD ["redis-server"]
 ---> Using cache
 ---> 7dfdfbcf1017
Successfully built 7dfdfbcf1017
➜  redis-image git:(master) ✗ docker run 7dfdfbcf1017
```

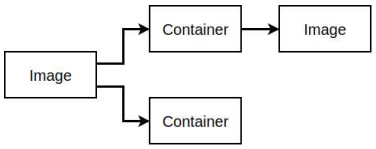


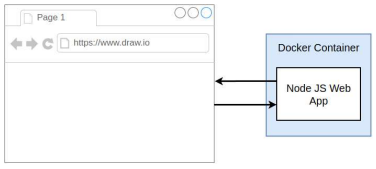








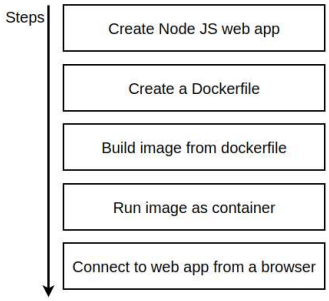





```
➜  redis-image git:(master) ✗ docker run busybox
➜  redis-image git:(master) ✗ docker build -t stephengrider/redis:lates
t .
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM alpine
 ---> 11cd0b38bc3c
Step 2/4 : RUN apk add --update gcc
 ---> Using cache
 ---> d409bde64c12
Step 3/4 : RUN apk add --update redis
 ---> Using cache
 ---> 3266f6626ef3
Step 4/4 : CMD ["redis-server"]
 ---> Using cache
 ---> 7dfdfbcf1017
Successfully built 7dfdfbcf1017
Successfully tagged stephengrider/redis:latest
➜  redis-image git:(master) ✗ docker run stephengrider/redis
```

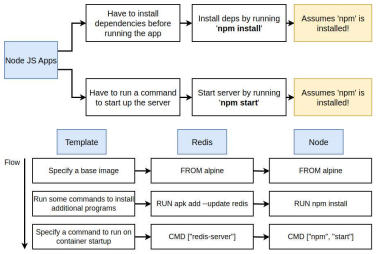
```
➜  redis-image git:(master) ✗ docker run -it alpine sh
/ # apk add --update redis
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/main/x86_64/APKINDEX.ta
r.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/community/x86_64/APKIND
EX.tar.gz
(1/1) Installing redis (4.0.10-r1)
Executing redis-4.0.10-r1.pre-install
Executing redis-4.0.10-r1.post-install
Executing busybox-1.28.4-r0.trigger
OK: 6 MiB in 14 packages
/ #
```

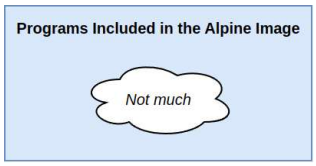
```
➜  redis-image git:(master) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
     STATUS              PORTS               NAMES
39075447a383        alpine              "sh"                48 seconds ago
     Up 47 seconds                           friendly_brown
➜  redis-image git:(master) ✗ docker commit -c 'CMD ["redis-server"]' 39075
447a383
sha256:0835898662dfae4b5a9d9e9961a0e89c2cbe55d895a0c0fbe6b7664f980d272b
➜  redis-image git:(master) ✗
```

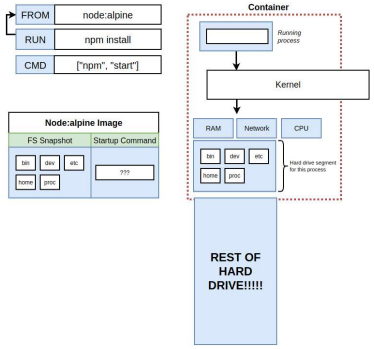


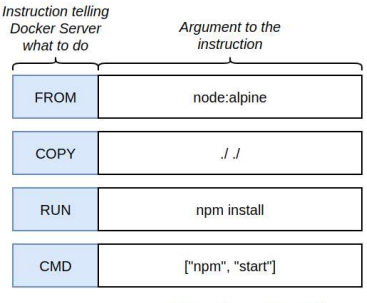
Disclaimer

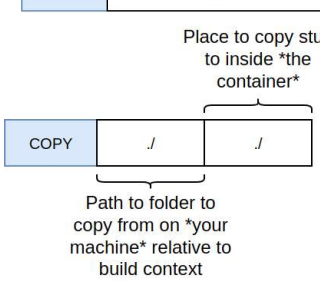
We're going to do a few things slightly wrong!

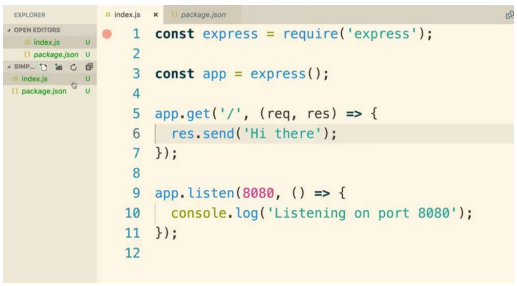








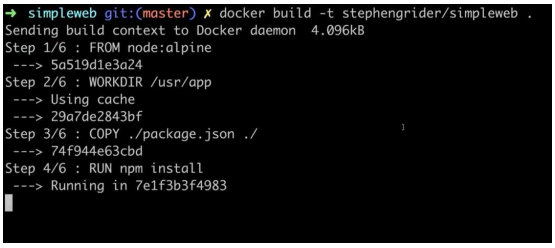




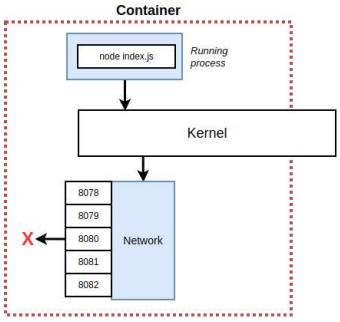
```
const express = require('express');

const app = express();

app.get('/', (req, res) => {
  res.send('Hi there');
});

app.listen(8080, () => {
  console.log('Listening on port 8080');
});
```

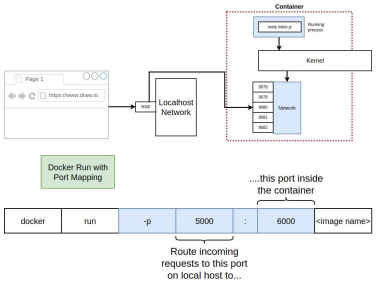
```
{
  "dependencies": {
    "express": "*"
  },
  "scripts": {
    "start": "node index.js"
  }
}
```

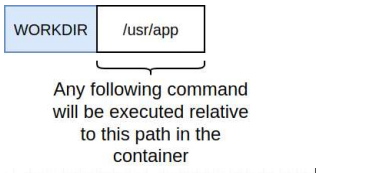
```
# Specify a base image
FROM node:alpine

WORKDIR /usr/app

# Install some depenendencies
COPY ./package.json ./
RUN npm install
COPY ./ ./

# Default command
CMD ["npm", "start"]
```

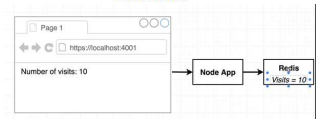


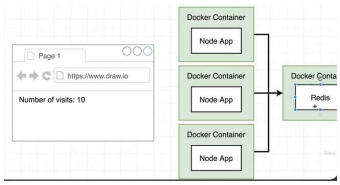
```
➜  simpleweb git:(master) ✗ docker build -t stephengrider/simpleweb .
Sending build context to Docker daemon  4.096kB
Step 1/6 : FROM node:alpine
 ---> 5a519d1e3a24
Step 2/6 : WORKDIR /usr/app
 ---> Using cache
 ---> 29a7de2843bf
Step 3/6 : COPY ./package.json ./
 ---> 74f944e63cbd
Step 4/6 : RUN npm install
 ---> Running in 7e1f3b3f4983
```

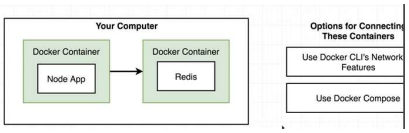


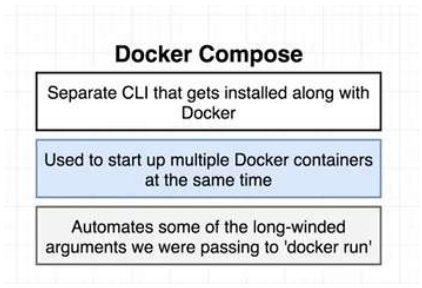


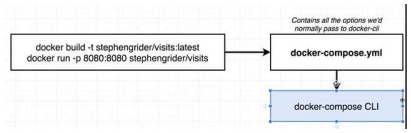


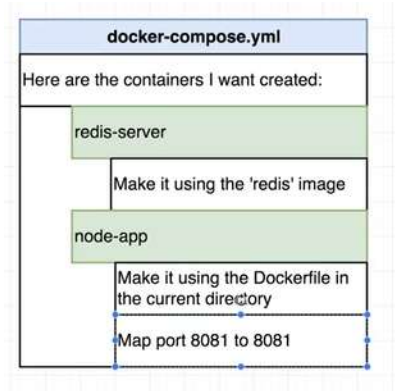












package.json

```
{
  "dependencies": {
    "express": "*",
    "redis": "2.8.0"
  },
  "scripts": {
    "start": "node index.js"
  }
}
```

package.json / index.js

```
const express = require('express');
const redis = require('redis');

const app = express();
const client = redis.createClient();
client.set('visits', 0);

app.get('/', (req, res) => {
  client.get('visits', (err, visits) => {
    res.send('Number of visits is ' + visits);
    client.set('visits', parseInt(visits) + 1);
  });
});

app.listen(8081, () => {
  console.log('Listening on port 8081');
});
```

Dockerfile

```
FROM node:alpine

WORKDIR '/app'

COPY package.json .
RUN npm install
COPY . .

CMD ["npm", "start"]
```

```
➜ visists git:(057-app) ✗ docker build -t stephengrider
s:latest .
```

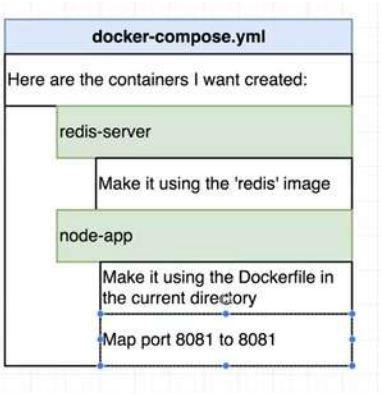
```
➜ visists git:(058-dockerfile) docker run stephengrider/visi
```

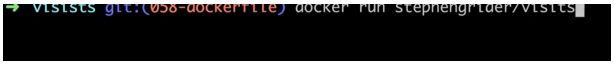
```
➜ visists git:(058-dockerfile) docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
be8881be8156: Downloading  11.02MB/22.49MB
```

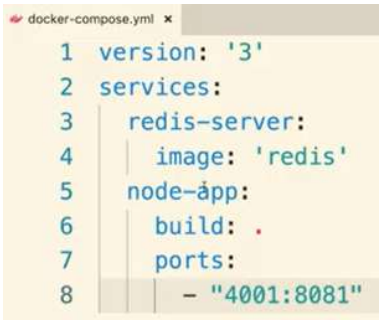
```
Last login: Tue Jul 24 16:13:51 on ttys000
➜ visists git:(058-dockerfile) docker run stephengrider/visits
```

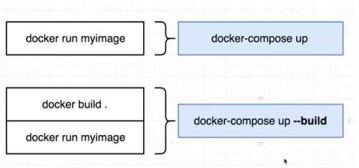
docker-compose.yml

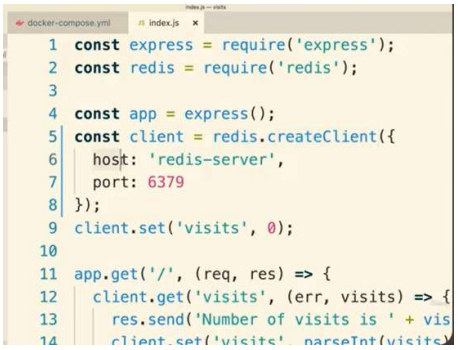
```
version: '3'
services:
  redis-server:
    image: 'redis'
  node-app:
    build: .
    ports:
      - "4001:8081"
```









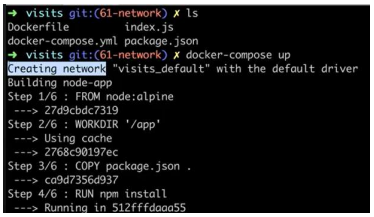


Launch in background

```
docker-compose up -d
```

Stop Containers

```
docker-compose down
```

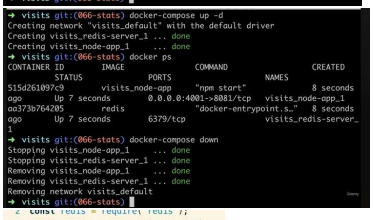


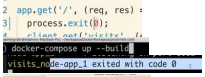
Status Codes

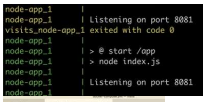
0	We exited and everything is OK
1, 2, 3, etc	We exited because something went wrong!

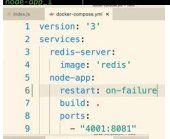
Restart Policies

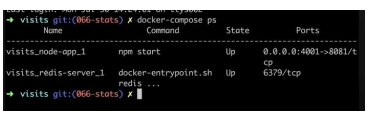
"no"	Never attempt to restart this . container if it stops or crashes
always	If this container stops "for any reason" always attempt to restart it
on-failure	Only restart if the container stops with an error code
unless-stopped	Always restart unless we (the developers) forcibly stop it

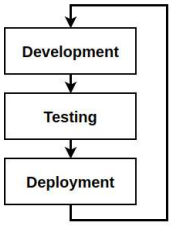


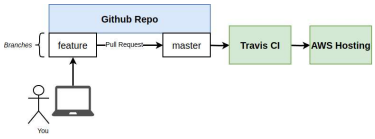


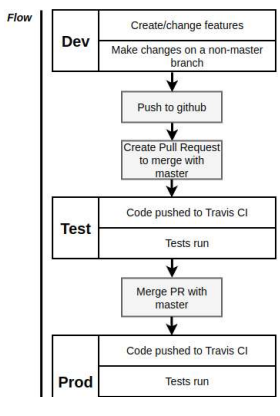


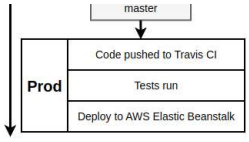












Something to notice...

- Last diagram didn't mention anything about Docker!
- Docker is a ***tool*** in a normal development flow
- Docker makes some of these tasks a lot easier

npm run start	Starts up a development server. *For development use only*
npm run test	Runs tests associated with the project
npm run build	Builds a **production** version of the application

In Development...

Docker Container

npm run ***start***

In Production...

Docker Container

npm run ***build***

```
visits_node-app_1       npm start              Up      0.0.0.0:4001->8081/tcp
visits_redis-server_1   docker-entrypoint.sh   Up      6379/tcp
                        redis ...
➜ visits git:(066-stats) ✗
```

```
➜ prod git:(066-stats) ✗ ls
diagrams    redis-image visits      web
➜ prod git:(066-stats) ✗ docker-compose ps
ERROR:
        Can't find a suitable configuration file in this directory or any
        parent. Are you in the right directory?

        Supported filenames: docker-compose.yml, docker-compose.yaml

➜ prod git:(066-stats) ✗
```

```
➜ ~ node -v
v8.11.3
➜ ~
```

Make sure to use npx as shown below!

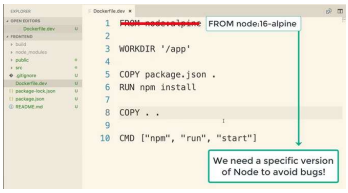
```
➜ prod git:(070-flow) ls
diagrams    redis-image visits      web
➜ prod git:(070-flow) npx create-react-app frontend

Creating a new React app in /Users/stephengrider/workspace/DockerWorkspace/
prod/frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

(                              ) ⸩ fetchMetadata: sill resolveWithNewModule esutils@2.
```

```
➜ frontend git:(070-flow) ✗ npm run test

> frontend@0.1.0 test /Users/stephengrider/w
rontend
> react-scripts test --env=jsdom
```

```
➜ frontend git:(070-flow) ✗ npm run build

> frontend@0.1.0 build /Users/stephengrider/w
rontend
> react-scripts build

Creating an optimized production build...
```

```
➜ frontend git:(070-flow) ✗ ls
README.md           node_modules        package.json        src
build               package-lock.json public
➜ frontend git:(070-flow) ✗ ls build
asset-manifest.json index.html          service-worker.js
favicon.ico         manifest.json       static
➜ frontend git:(070-flow) ✗ ls build/stic
ls: build/stic: No such file or directory
➜ frontend git:(070-flow) ✗ ls build/static
css   js    media
➜ frontend git:(070-flow) ✗ ls build/static/js
main.a0b7d8d3.js    main.a0b7d8d3.js.map
➜ frontend git:(070-flow) ✗
```

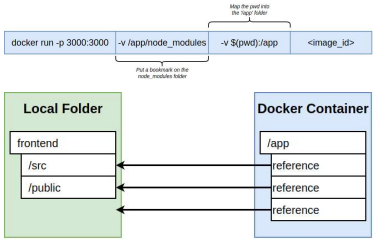
```
➜ frontend git:(070-flow) ✗ npm run start

> frontend@0.1.0 start /Users/stephengrider/
frontend
> react-scripts start
```




```
➜ frontend git:(070-flow) ✗ ls
Dockerfile.dev      build               package-lock.json public
README.md           node_modules        package.json      src
➜ frontend git:(070-flow) ✗ docker build .
unable to prepare context: unable to evaluate symlinks in Dockerfile path:
lstat /Users/stephengrider/workspace/DockerWorkspace/prod/frontend/Dockerfi
le: no such file or directory
➜ frontend git:(070-flow) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon  155.2MB
Step 1/6 : FROM node:alpine
 ---> 27d9cbdc7319
Step 2/6 : WORKDIR '/app'
 ---> Using cache
 ---> 2768a90197ac
Step 3/6 : COPY package.json .
```

```
➜ frontend git:(070-flow) ✗ open .
➜ frontend git:(070-flow) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon  155.2MB
Step 1/6 : FROM node:alpine
```

```
➜ frontend git:(070-flow) ✗ docker run -p 3000:3000 57f59038ad29

> frontend@0.1.0 start /app
> react-scripts start

Starting the development server...
```




```
➜ frontend git:(070-flow) ✗ docker build -f Dockerfile.dev .
➜ frontend git:(070-flow) ✗ docker run -p 3000:3000 -v ${pwd}:/app 68dabe9
a308e

> frontend@0.1.0 start /app
> react-scripts start

sh: react-scripts: not found
npm ERR! file sh
➜ frontend git:(070-flow) ✗ docker run -p 3000:3000 -v /app/node_modules -
v ${pwd}:/app 68dabe9a308e

> frontend@0.1.0 start /app
> react-scripts start

Starting the development server...
```

```
version: '3'
services:
  web:
    build: .
    ports:
      - "3000:3000"
    volumes:
      - /app/node_modules
      - .:/app
```

```
➜ frontend git:(070-flow) ✗ docker-compose up
Creating network "frontend_default" with the default driver
Building web
ERROR: Cannot locate specified Dockerfile: Dockerfile
➜ frontend git:(070-flow) ✗
```

```
version: '3'
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile.dev
    ports:
      - "3000:3000"
    volumes:
      - /app/node_modules
      - .:/app
```

```
➜ frontend git:(070-flow) ✗ docker-compose up
Building web
Step 1/6 : FROM node:alpine
 ---> 27d9cbdc7319
```

```
➜ frontend git:(070-flow) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon  557.1kB
```

```
➜ frontend git:(070-flow) ✗ docker run b8cb8d82171c npm run test

> frontend@0.1.0 test /app
> react-scripts test --env=jsdom

➜ frontend git:(070-flow) ✗ docker run -it b8cb8d82171c npm run test

> frontend@0.1.0 test /app
> react-scripts test --env=jsdom
```

```
PASS src/App.test.js
  ✓ renders without crashing (5ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
```

```
→ frontend git:(070-flow) ✗ docker run -it b8cb8d82171c npm run test

> frontend@0.1.0 test /app
> react-scripts test --env=jsdom
```

```
PASS  src/App.test.js
✓ renders without crashing (5ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.855s, estimated 1s
Ran all test suites matching /app/.

Watch Usage: Press w to show more.
```

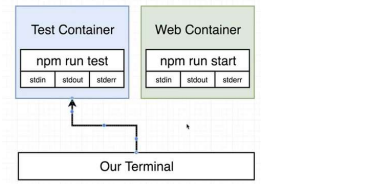
```
import ReactDOM from 'react-dom';
import App from './App';

it('renders without crashing', () => {
  const div = document.createElement('div');
  ReactDOM.render(<App />, div);
  ReactDOM.unmountComponentAtNode(div);
});

it('renders without crashing', () => {
  const div = document.createElement('div');
  ReactDOM.render(<App />, div);
  ReactDOM.unmountComponentAtNode(div);
});
```

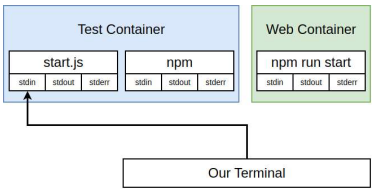
```
→ frontend git:(081-t) ✗ docker-compose up
Starting frontend_web_1 ... done
Attaching to frontend_web_1
```

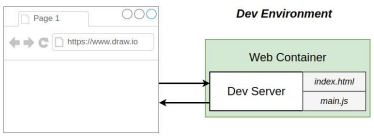
```
→ frontend git:(081-t) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS                    NAMES
0fe01eef2915        frontend_web        "npm run start"          36 minutes ago
    Up 43 seconds       0.0.0.0:3000->3000/tcp   frontend_web_1
→ frontend git:(081-t) ✗ docker exec -it 0fe01eef2915 npm run test

> frontend@0.1.0 test /app
> react-scripts test --env=jsdom
```

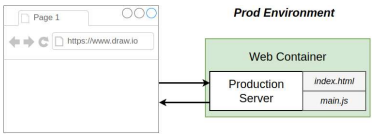
```
  tests:
    build:
      context: .
      dockerfile: Dockerfile.dev
    volumes:
      - /app/node_modules
      - .:/app
    command: ["npm", "run", "test"]
```

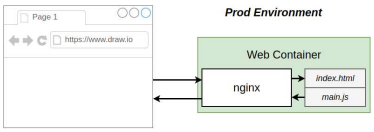
```
frontend git:(081-t) ✗ docker-compose up --build
→ frontend git:(081-t) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS                    NAMES
35eedb8be35c        frontend_web        "npm run start"          6 minutes ago
     Up 2 minutes        0.0.0.0:3000->3000/tcp   frontend_web_1
cf0e3950c7b5        frontend_tests      "npm run test"           6 minutes ago
     Up 2 minutes                                 frontend_tests_1
→ frontend git:(081-t) ✗ docker attach 35eedb8be35c
p
^
→ frontend git:(081-t) ✗ docker exec -it 35eedb8be35c sh
/app # ps
PID   USER     TIME  COMMAND
    1 root      0:00 npm
   17 root      0:00 node /app/node_modules/.bin/react-scripts start
   23 root      0:03 node /app/node_modules/react-scripts/scripts/start.js
   58 root      0:00 sh
   65 root      0:00 ps
/app #
```

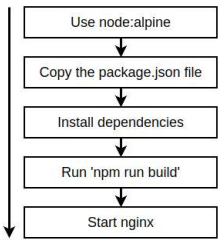


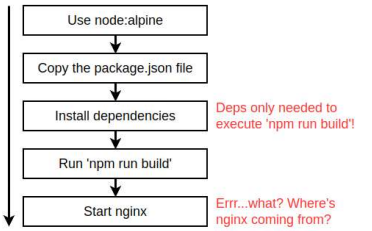


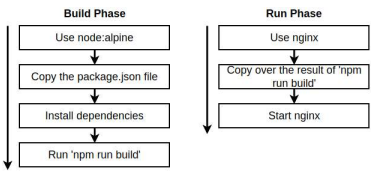












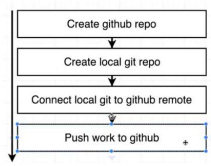

```
Dockerfile
1  FROM node:alpine as builder
2  WORKDIR '/app'
3  COPY package.json .
4  RUN npm install
5  COPY . .
6  RUN npm run build
7
8  /app/build <--- all the stuff
```

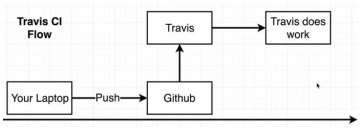
FROM node:16-alpine as builder

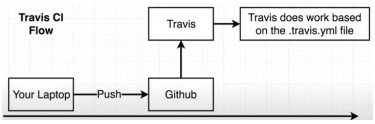
```
1  FROM node:alpine as builder
2  WORKDIR '/app'
3  COPY package.json .
4  RUN npm install
5  COPY . .
6  RUN npm run build
7
8  FROM nginx
9  COPY --from=builder /app/build /usr/share/nginx/html
```

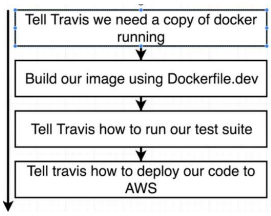
```
→ frontend git:(087-multi) docker build .
Sending build context to Docker daemon  1.181MB
Step 1/8 : FROM node:alpine as builder
 ---> 27d9cbdc7319
Step 2/8 : WORKDIR '/app'
→ frontend git:(087-multi) docker run -p 8080:80 539657aa6f75
```

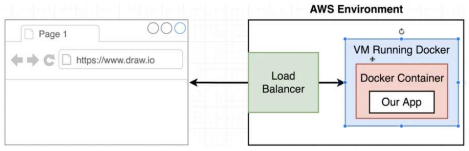
Github	Free!
Travis CI	Free!
AWS	Free, but credit card required

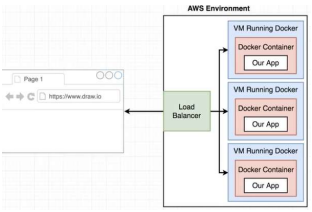






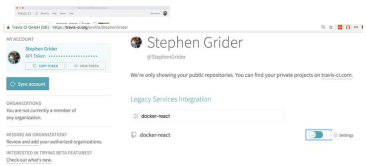







```
→ frontend git:(087-multi) git init
Initialized empty Git repository in /Users/stephengrider/workspace/DockerWo
rkspace/prod/frontend/.git/
→ frontend git:(master) ✗ git add .
→ frontend git:(master) ✗ git commit -m "initial commit"
```

```
→ frontend git:(master) git remote add origin git@github.com:StephenGrider
/docker-react.git
→ frontend git:(master) git push origin master
Counting objects: 21, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 132.22 KiB | 3.39 MiB/s, done.
Total 21 (delta 0), reused 0 (delta 0)
To github.com:StephenGrider/docker-react.git
 * [new branch]      master -> master
→ frontend git:(master)
```




```
1  sudo: required
2  services:
3    - docker
4
5  before_install:
6    - docker build -t  -f Dockerfile.dev .
```

```
→ frontend git:(master) ✗ docker run 3d318928e02c npm run test -- --covera
ge
```

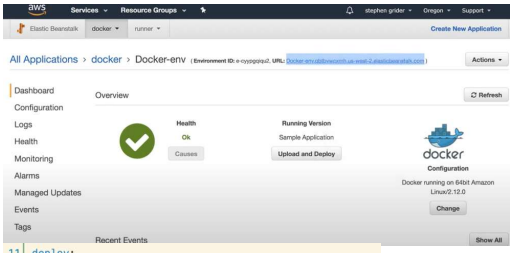
```
8  script:
9    - docker run -e CI=true stephengrider/docker-react npm run test
```

Updated script with required **-e CI=true** variable and omitting the coverage flag.

```
1  sudo: required
2  services:
3    - docker
4
5  before_install:
6    - docker build -t stephengrider/docker-react -f Dockerfile.dev .
7
8  script:
9    - docker run stephengrider/docker-react npm run test -- --coverage
```

```
→ frontend git:(master) ✗ git add .
→ frontend git:(master) ✗ git commit -m "added travis file"
[master bdc5813] added travis file
 1 file changed, 9 insertions(+)
 create mode 100644 .travis.yml
→ frontend git:(master) git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 391 bytes | 391.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:StephenGrider/docker-react.git
   b61ae2e..bdc5813  master -> master
→ frontend git:(master)
```

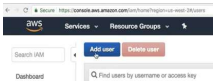
file:///C:/Users/ANKIT/Downloads/qBit/Udemy/Docker%20and%20Kubernetes/07%20-%20Continuous%20Integration%20and%20Deployment%20with%20AWS/009%20Elastic%20Beanstalk%20Setup%20and%20Configuration.html

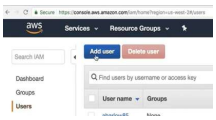



```
11 deploy:
12   provider: elasticbeanstalk
13   region: "us-west-2"
14
15   Docker-env.qbtbvwcxmh.us-west-2.elasticbeanstalk.com
```

```
All Applications › docker › Docker-env
11 deploy:
12   provider: elasticbeanstalk
13   region: "us-west-2"
14   app: "docker"
15   env: "Docker-env"
```

```
Amazon S3 › elasticbeanstalk-us-west-2-306476627547
.travis.yml
   - docker build -t stephengrider/docker-react -f Dockerfile.dev .
7
8  script:
9    - docker run stephengrider/docker-react npm run test -- --coverage
10
11 deploy:
12   provider: elasticbeanstalk
13   region: "us-west-2"
14   app: "docker"
15   env: "Docker-env"
16   bucket_name: "elasticbeanstalk-us-west-2-306476627547"
17   bucket_path: "docker"
18   on:
19     branch: master
20
```





Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* docker-react-travis-ci

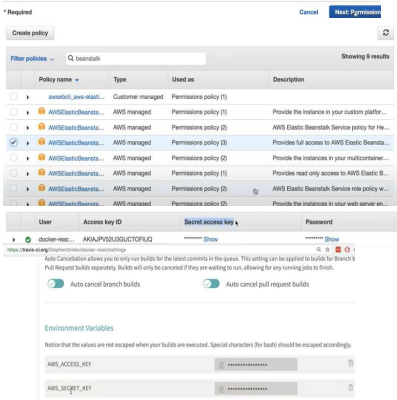
Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

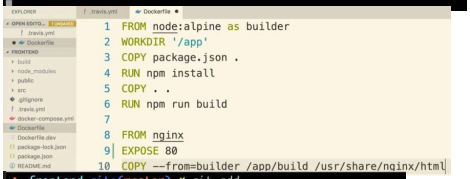
Access type* Programmatic access
Enables an access key ID and secret access key for the AWS API, CLI, SDK, and other development tools.

AWS Management Console access
Enables a password that allows users to sign-in to the AWS Management Console.

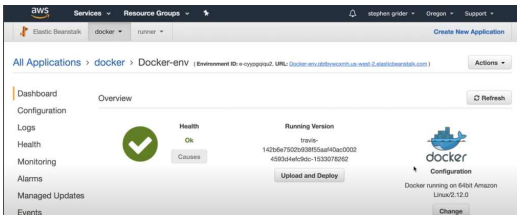



```yaml
deploy:
  provider: elasticbeanstalk
  region: "us-west-2"
  app: "docker"
  env: "Docker-env"
  bucket_name: "elasticbeanstalk-us-west-2-306476627547"
  bucket_path: "docker"
  on:
    branch: master
  access_key_id: $AWS_ACCESS_KEY
  secret_access_key: "$AWS_SECRET_KEY"
```

```
➜ frontend git:(master) code .
➜ frontend git:(master) gst
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .travis.yml

no changes added to commit (use "git add" and/or "git commit -a")
➜ frontend git:(master) ✗ git add .
➜ frontend git:(master) ✗ git commit -m "added travis deploy config"
[master faead22] added travis deploy config
 1 file changed, 15 insertions(+), 1 deletion(-)
➜ frontend git:(master) git push origin master
```




```
➜ frontend git:(master) ✗ git add .
➜ frontend git:(master) ✗ git commit -m "added expose 80"
[master 20debb8] added expose 80
 1 file changed, 1 insertion(+)
➜ frontend git:(master) git push origin master
```

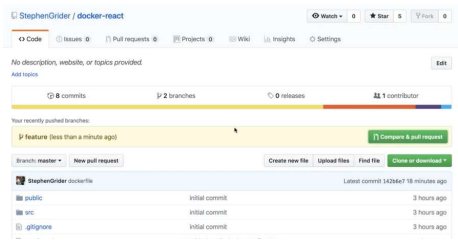


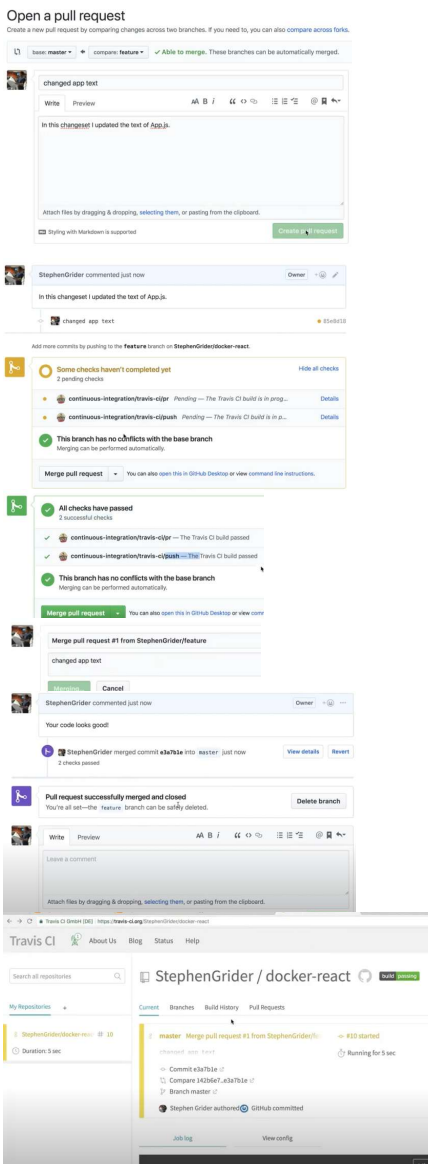

```
➜ frontend git:(master) git checkout -b feature
Switched to a new branch 'feature'
```

```
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo"
          <h1 className="App-title">Welcome to
        </header>
        <p className="App-intro">
          I was changed on the Feature branch
```

```
➜ frontend git:(master) git checkout -b feature
Switched to a new branch 'feature'
➜ frontend git:(feature) git add .
➜ frontend git:(feature) ✗ git commit -m "changed app text"
[feature 85e8d18] changed app text
 1 file changed, 1 insertion(+), 1 deletion(-)
➜ frontend git:(feature) git push origin feature
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
```

```
[feature 85e8d18] changed app text
 1 file changed, 1 insertion(+), 1 deletion(-)
➜  frontend git:(feature) git push origin feature
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 372 bytes | 372.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:StephenGrider/docker-react.git
 * [new branch]      feature -> feature
➜  frontend git:(feature)
```





Something to notice...

Last diagram didn't mention anything about Docker!

Docker is a ***tool*** in a normal development flow

Docker makes some of these tasks a lot easier

Don't forget to terminate the Elastic Beanstalk app!

Single Container Deployment Issues

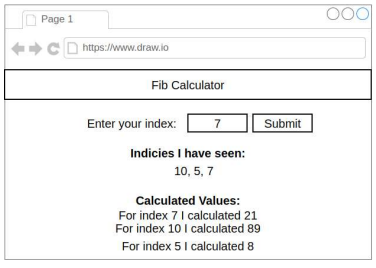
The app was simple - no outside dependencies

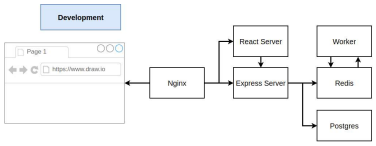
Our image was built multiple times

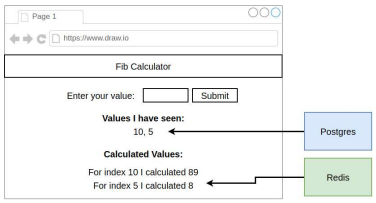
How do we connect to a database from a container?

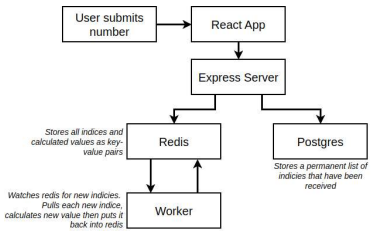
Fib Sequence:

Value	1	1	2	3	5	8	13	21	...
Index	0	1	2	3	4	5	6	7	









We're going to do just the JS side of the app right now

(no docker!)

If you don't care about Javascript stuff then you can skip the next few videos and download the completed app

```
→ prod git:(master) ls
diagrams    frontend    redis-image simpleweb
→ prod git:(master) mkdir complex
→ prod git:(master) cd complex
→ complex git:(master) mkdir worker
→ complex git:(master) code .
```

package.json

```
{
  "dependencies": {
    "nodemon": "1.18.3",
    "redis": "2.8.0"
  },
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon"
  }
}
```

keys.js

```
module.exports = {
  redisHost: process.env.REDIS_HOST,
  redisPort: process.env.REDIS_PORT
};
```

index.js

```
const keys = require('./keys');
const redis = require('redis');

const redisClient = redis.createClient({
  host: keys.redisHost,
  port: keys.redisPort,
  retry_strategy: () => 1000
});
const sub = redisClient.duplicate();
```

```
const redisClient = redis.createClient({
  host: keys.redisHost,
  port: keys.redisPort,
  retry_strategy: () => 1000
});
const sub = redisClient.duplicate();

function fib(index) {
  if (index < 2) return 1;
  return fib(index - 1) + fib(index - 2);
}

sub.on('message', (channel, message) => {
  redisClient.hset('values', message, fib(parseInt(message)));
});
sub.subscribe('insert');
```

```
→ worker git:(master) ✗ node index.js
module.js:549
    throw err;
    ^

Error: Cannot find module 'redis'
```

package.json

```
{
  "dependencies": {
    "express": "4.16.3",
    "pg": "8.0.3",
    "redis": "2.8.0",
    "cors": "2.8.4",
    "nodemon": "1.18.3"
  },
  "scripts": {
    "dev": "nodemon",
    "start": "node index.js"
  }
}
```

keys.js

```
module.exports = {
  redisHost: process.env.REDIS_HOST,
  redisPort: process.env.REDIS_PORT,
  pgUser: process.env.PGUSER,
  pgHost: process.env.PGHOST,
  pgDatabase: process.env.PGDATABASE,
  pgPassword: process.env.PGPASSWORD,
  pgPort: process.env.PGPORT
};
```

package.json

```
{
  "dependencies": {
    "express": "4.16.3",
    "pg": "7.4.3",
    "redis": "2.8.0",
    "cors": "2.8.4",
    "nodemon": "1.18.3",
    "body-parser": "*"
  },
  "scripts": {
    "dev": "nodemon",
    "start": "node index.js"
  }
}
```

index.js

```
const keys = require('./keys');

// Express App Setup
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(bodyParser.json());

// Postgres Client Setup
```

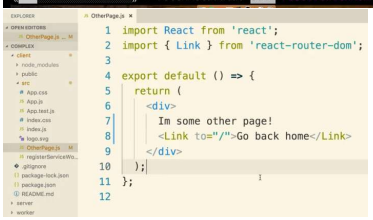
```
// Postgres Client Setup
const { Pool } = require('pg');
const pgClient = new Pool({
  user: keys.pgUser,
  host: keys.pgHost,
```

```
// Postgres Client Setup
```

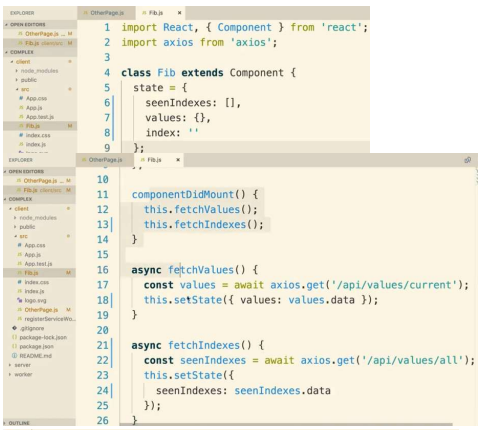
```
// Postgres Client Setup
const { Pool } = require('pg');
const pgClient = new Pool({
  user: keys.pgUser,
  host: keys.pgHost,
  database: keys.pgDatabase,
  password: keys.pgPassword,
  port: keys.pgPort
});
pgClient.on('error', () => console.log('Lost PG connection'));

pgClient
  .query('CREATE TABLE IF NOT EXISTS values (number INT)')
  .catch(err => console.log(err));

// Redis Client Setup
const redis = require('redis');
const redisClient = redis.createClient({
  host: keys.redisHost,
  port: keys.redisPort,
  retry_strategy: () => 1000
});
const redisPublisher = redisClient.duplicate();

// Express route handlers

app.get('/', (req, res) => {
  res.send('Hi');
});

app.get('/values/all', async (req, res) => {
  const values = await pgClient.query('SELECT * from values');

  res.send(values.rows);
});
app.get('/values/current', async (req, res) => {
  redisClient.hgetall('values', (err, values) => {
    res.send(values);
  });
});

app.post('/values', async (req, res) => {
  const index = req.body.index;

  if (parseInt(index) > 40) {
    return res.status(422).send('Index too high');
  }

  redisClient.hset('values', index, 'Nothing yet!');
  redisPublisher.publish('insert', index);
  pgClient.query('INSERT INTO values(number) VALUES($1)', [index]);

  res.send({ working: true });
});

app.listen(5000, err => {
  console.log('Listening');
});
```

Make sure to use npx as shown below!

```
➜  complex git:(master) ✗ npx create-react-app client

Creating a new React app in /Users/stephengrider/workspace/DockerWorkspace/prod/complex/client.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

(                    ) ⸩ fetchMetadata: sill resolveWithNewModule jest-c
```



```
import React from 'react';
import { Link } from 'react-router-dom';

export default () => {
  return (
    <div>
      Im some other page!
      <Link to="/">Go back home</Link>
    </div>
  );
};
```



```
import React, { Component } from 'react';
import axios from 'axios';

class Fib extends Component {
  state = {
    seenIndexes: [],
    values: {},
    index: ''
  };

  componentDidMount() {
    this.fetchValues();
    this.fetchIndexes();
  }

  async fetchValues() {
    const values = await axios.get('/api/values/current');
    this.setState({ values: values.data });
  }

  async fetchIndexes() {
    const seenIndexes = await axios.get('/api/values/all');
    this.setState({
      seenIndexes: seenIndexes.data
    });
  }

  handleSubmit = async (event) => {
    event.preventDefault();

    await axios.post('/api/values', {
      index: this.state.index
    });
    this.setState({ index: '' });
  };

  renderSeenIndexes() {
    return this.state.seenIndexes.map(({ number }) => number).join(', ');
  }

  renderValues() {
    const entries = [];
```

If you didn't write all the source code out in the last couple sections

Stay around, make sure you got checkpoint.zip from the last lecture

If you wrote all the code with me

Continue on!

Or, download the checkpoint.zip and use my copy of code (to avoid typos)

React App

Express Server

Worker

Need to make **dev** Dockerfiles for each

package.json #1

src
index.js #2

Copy over package.json

Run 'npm install'

Copy over everything else

Docker compose should set up a volume to 'share' files

Compose

postgres

What image to use?

```
renderValues() {
  const entries = [];
renderValues() {
  const entries = [];

  for (let key in this.state.values) {
    entries.push(
      <div key={key}>
        For index {key} I calculated {this.state.values[k
      </div>
    );
  }

  return entries;
}
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <label>Enter your index:</label>
        <input
          value={this.state.index}
          onChange={event => this.setState({ index: event
        />
        <button>Submit</button>
      </form>

      <h3>Indexes I have seen:</h3>
      {this.renderSeenIndexes()}

      <h3>Calculated Values:</h3>
      {this.renderValues()}
    </div>
  );
  }
}

export default Fib;
```

```
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.4.2",
    "react-dom": "^16.4.2",
    "react-scripts": "1.1.4",
    "react-router-dom": "4.3.1",
    "axios": "0.18.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
```

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom'
import OtherPage from './OtherPage';
import Fib from './Fib';
```

```
class App extends Component {
  render() {
    return (
      <Router>
        <div className="App">
          <header className="App-header">
            <img src={logo} className="App-logo" alt="logo" />
            <h1 className="App-title">Welcome to React</h1>
            <Link to="/">Home</Link>
            <Link to="/otherpage">Other Page</Link>
          </header>
          <div>
            <Route exact path="/" component={Fib} />
            <Route path="/otherpage" component={OtherPage} />
          </div>
        </Router>
```

```
→ prod git:(master) ✗ ls
diagrams    frontend    redis-image simpleweb
→ prod git:(master) ✗ mkdir complex
→ prod git:(master) ✗ cd complex
→ complex git:(master) ✗ open .
→ complex git:(master) ✗ ls
client server worker
```

```
FROM node:alpine
WORKDIR '/app'
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

FROM node:16-alpine

```
→ complex git:(master) code .
→ complex git:(master) ls
client server worker
→ complex git:(master) ✗ cd client
→ client git:(master) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon  561.7kB
Step 1/6 : FROM node:alpine
Step 6/6 : CMD ["npm", "run", "start"]
 ---> Running in f1760d1b6ea7
Removing intermediate container f1760d1b6ea7
 ---> 71088f27a8f5
Successfully built 71088f27a8f5
→ client git:(master) ✗ docker run 71088f27a8f5

> client@0.1.0 start /app
> react-scripts start

Starting the development server...
```

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "dev"]
```

FROM node:14.14.0-alpine

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "dev"]
```

FROM node:

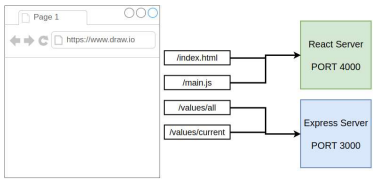
```
→ complex git:(master) ✗ ls
client server worker
```

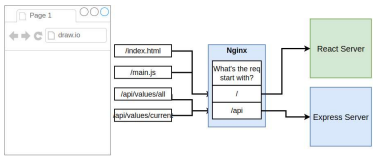
Compose

- postgres
  - What image to use?
- redis
  - What image to use?
- server
  - Specify build
  - Specify volumes
  - Specify env variables

Environment Variables

variableName=value	variableName
*Sets a variable in the container at *run time**	*Sets a variable in the container at *run time** *Value is taken from *your computer**

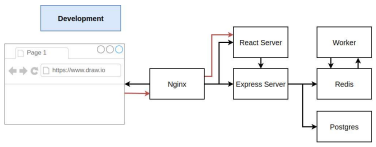




default.conf

Adds configuration rules to Nginx

- Tell Nginx that there is an 'upstream' server at client:3000
- Tell Nginx that there is an 'upstream' server at server:5000
- Listen on port 80
- If anyone comes to '/' send them to client upstream
- If anyone comes to '/api' send them to server upstream



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

```
5  COPY . .
6  CMD ["npm", "run", "dev"]
```

```
➜  complex git:(master) ✗ ls
client server worker
➜  complex git:(master) ✗ cd server
➜  server git:(master) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon  6.656kB
Step 1/6 : FROM node:alpine
```

```
➜  complex git:(master) ✗ cd worker
➜  worker git:(master) ✗ docker build -f Dockerfile.dev .
Sending build context to Docker daemon   5.12kB
Step 1/6 : FROM node:alpine
```

```
1  version: '3'
2  services:
3    postgres:
4      image: 'postgres:latest'
5    redis:
6      image: 'redis:latest'
7    server:
8      build:
9        dockerfile: Dockerfile.dev
10       context: ./server
11     volumes:
12       - /app/node_modules
13       - ./server:/app
14
14   environment:
15     - REDIS_HOST=redis
16     - REDIS_PORT=6379
17     - PGUSER=postgres
18     - PGHOST=postgres
19     - PGDATABASE=postgres
20     - PGPASSWORD=postgres_password
21     - PGPORT=5432
```

```
➜  complex git:(master) ✗ docker-compose up
Building server
Step 1/6 : FROM node:alpine
```

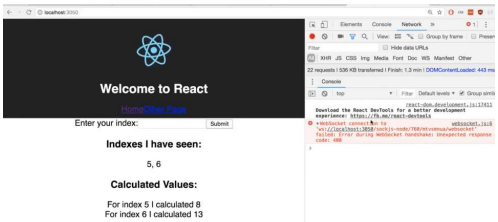
```
22   client:
23     build:
24       dockerfile: Dockerfile.dev
25       context: ./client
26     volumes:
27       - /app/node_modules
28       - ./client:/app
29   worker:
30     build:
31       dockerfile: Dockerfile.dev
32       context: ./worker
33     volumes:
34       - /app/node_modules
35       - ./worker:/app
```

```
1  upstream client {
2    server client:3000;
3  }
4
5  upstream api {
6    server api:5000;
7  }
8
9  server {
10   listen 80;
11
12   location / {
13     proxy_pass http://client;   :tp://client;
14   }
15
16   location /api {
17     rewrite /api/(.*) /$1 break;
18     proxy_pass http://api;
19   }
20 }
```

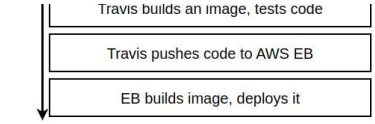
```
1  FROM nginx
2  COPY ./default.conf /etc/nginx/conf.d/default.conf
```

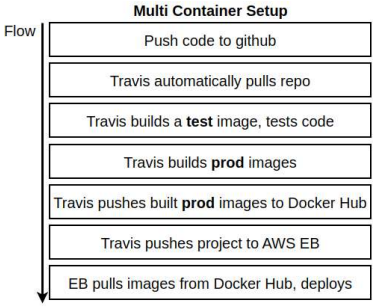
```
1  version: '3'
2  services:
3    postgres:
4      image: 'postgres:latest'
5    redis:
6      image: 'redis:latest'
7    nginx:
8      restart: always
9      build:
10       dockerfile: Dockerfile.dev
11       context: ./nginx
12     ports:
13       - '3050:80'
14   api:
15     build:
16       dockerfile: Dockerfile.dev
17       context: ./server
```

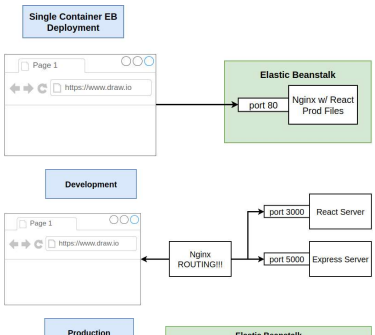
```
➜  complex git:(master) ✗ docker-compose up --build
Building worker
Step 1/6 : FROM node:alpine
```

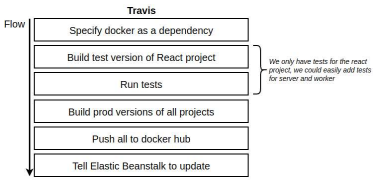


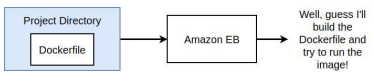

```
10   listen 80;
11
12   location / {
13     proxy_pass http://client;
14   }
15
16   location /sockjs-node {
17     proxy_pass http://client;
18     proxy_http_version 1.1;
19     proxy_set_header Upgrade $http_upgrade;
20     proxy_set_header Connection "Upgrade";
21   }
```

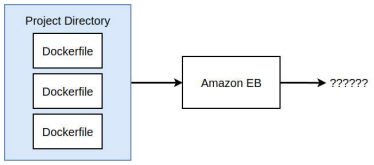


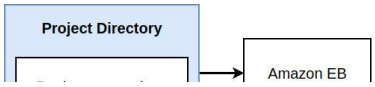













```
location /sockjs-node {
  proxy_pass http://client;
  proxy_http_version 1.1;
  proxy_set_header Upgrade $http_upgrade;
  proxy_set_header Connection "Upgrade";
}
```

```
➜ complex git:(master) ✗ docker-compose up --build
Building worker
```

```
FROM node:alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

```
FROM node:14.4.0-alpine
WORKDIR "/app"
COPY ./package.json ./
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

```
FROM nginx
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

it('renders without crashing', () => {});
```

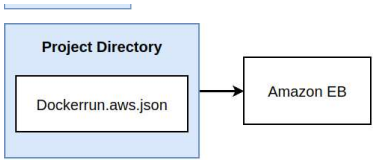
```
FROM node:alpine as builder
WORKDIR '/app'
COPY ./package.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx
EXPOSE 3000
COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
COPY --from=builder /app/build /usr/share/nginx/html
```

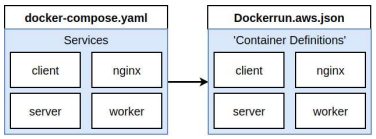
```
sudo: required
services:
  - docker

before_install:
  - docker build -t stephengrider/react-test -f ./client/Dockerfile.dev

script:
  - docker run stephengrider/react-test npm test -- --coverage

after_success:
  - docker build -t stephengrider/multi-client ./client
  - docker build -t stephengrider/multi-nginx ./nginx
  - docker build -t stephengrider/multi-server ./server
  - docker build -t stephengrider/multi-worker ./worker
```

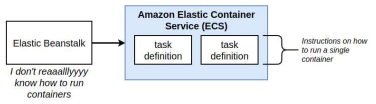
```
➜ complex git:(master) ✗ git init
Initialized empty Git repository in /Users/stephengrider/workspace/DockerWorkspace/prod/complex/.git/
➜ complex git:(master) ✗ git add .
➜ complex git:(master) ✗ git commit -m "initial commit"
➜ complex git:(master) git remote add origin git@github.com:StephenGrider/multi-docker.git
➜ complex git:(master) git remote -v
origin  git@github.com:StephenGrider/multi-docker.git (fetch)
origin  git@github.com:StephenGrider/multi-docker.git (push)
➜ complex git:(master) git push origin master
Counting objects: 41, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (40/40), done.
Writing objects: 100% (41/41), 137.63 KiB | 3.62 MiB/s, done.
Total 41 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To github.com:StephenGrider/multi-docker.git
 * [new branch]      master -> master
➜ complex git:(master)
```

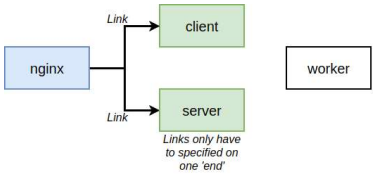
```
# Log in to the docker CLI
- echo "$DOCKER_PASSWORD" | docker login -u "$DOCKER_ID" --password-stdin
# Take those images and push them to docker hub
- docker push stephengrider/multi-client
- docker push stephengrider/multi-nginx
- docker push stephengrider/multi-server
- docker push stephengrider/multi-worker
```

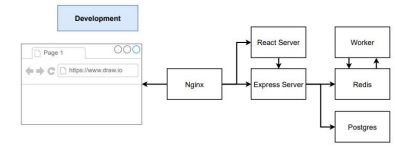
```
➜ complex git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working d

        modified:   .travis.yml

no changes added to commit (use "git add" and/or "git commit -a")
➜ complex git:(master) ✗ git add .
➜ complex git:(master) ✗ git commit -m "changed travis yml"
[master 59d70a2] changed travis yml
 1 file changed, 7 insertions(+)
➜ complex git:(master) git push origin master
```

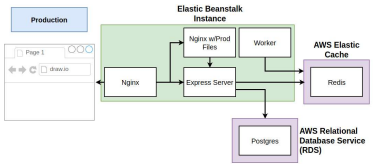


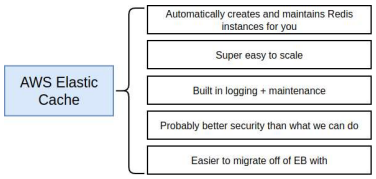


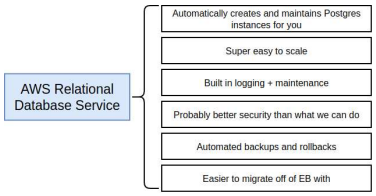


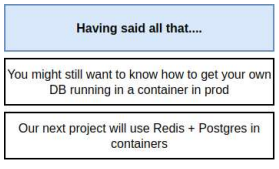


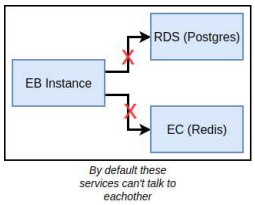


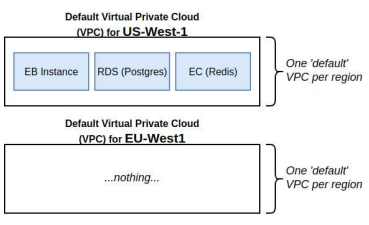













```
no changes added to commit (use "git add" and/or "git commit -a")
➜  complex git:(master) ✗ git add .
➜  complex git:(master) ✗ git commit -m "changed travis yml"
[master 59d70a2] changed travis yml
 1 file changed, 7 insertions(+)
➜  complex git:(master) git push origin master
```

```
EXPLORER
OPEN EDITORS 1 UNSAVED
  Dockerrun.aw... U
COMPLEX
  client
  nginx
    default.conf
    Dockerfile
    Dockerfile.dev
  server
  worker
  .travis.yml U
  docker-compose.yml
  Dockerrun.aws.jso... U
```

Dockerrun.aws.json

```
{
  "AWSEBDockerrunVersion": 2,
  "containerDefinitions": [
    {
      "name": "client",
      "image": "stephengrider/multi-client",
      "hostname": "client",
      "essential": false
    },
    {
      "name": "server",
      "image": "stephengrider/multi-server",
      "hostname": "api",
      "essential": false
    },
    {
      "name": "worker",
      "image": "stephengrider/multi-worker",
      "hostname": "worker",
      "essential": false
    },
    {
      "name": "nginx",
      "image": "stephengrider/multi-nginx",
      "hostname": "nginx",
      "essential": true,
      "portMappings": [
        {
          "hostPort": 80,
          "containerPort": 80
        }
      ],
      "links": ["client", "server"]
    }
  ]
}
```

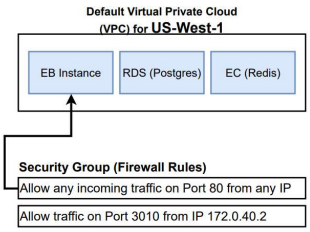
https://jsonlint.com

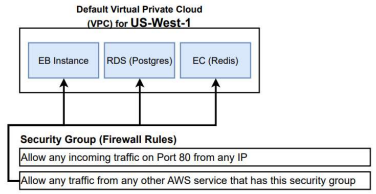
JSONLint - The JSON Validator

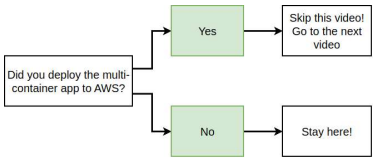
Validate JSON | Clear

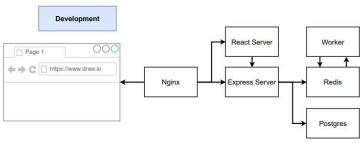
Results

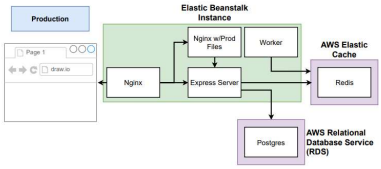
Valid JSON

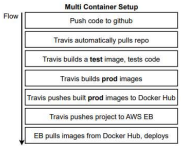

Default Virtual Private Cloud
(VPC) for US-West-1
EB Instance
RDS (Postgres)
EC (Redis)
Security Group (Firewall Rules)
Allow any incoming traffic on Port 80 from any IP
Allow traffic on Port 3010 from IP 172.0.40.2

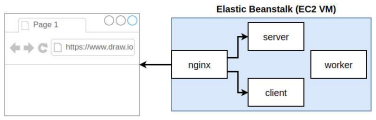

Default Virtual Private Cloud
(VPC) for US-West-1
EB Instance
RDS (Postgres)
EC (Redis)
Security Group (Firewall Rules)
Allow any incoming traffic on Port 80 from any IP
Allow any traffic from any other AWS service that has this security group

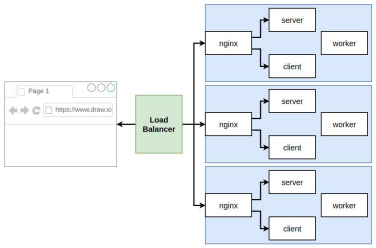

Did you deploy the multi-container app to AWS?
Yes
Skip this video! Go to the next video
No
Stay here!

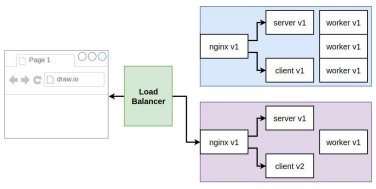

Development
Page 1
https://www.draw.io
Nginx
React Server
Express Server
Worker
Redis
Postgres

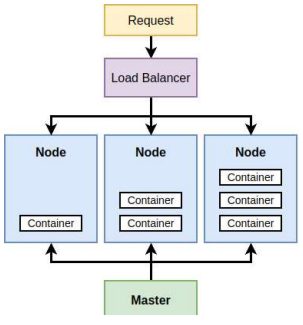

Production
Elastic Beanstalk Instance
Page 1
draw.io
Nginx
Nginx w/Prod Files
Worker
Express Server
AWS Elastic Cache
Redis
Postgres
AWS Relational Database Service (RDS)

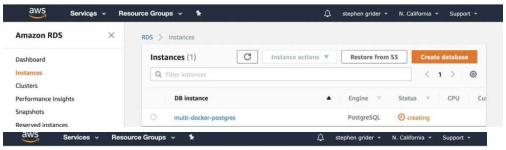

Multi Container Setup
Flow
Push code to github
Travis automatically pulls repo
Travis builds a test image, tests code
Travis builds prod images
Travis pushes built prod images to Docker Hub
Travis pushes project to AWS EB
EB pulls images from Docker Hub, deploys

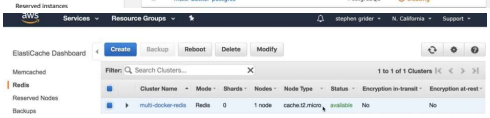

Elastic Beanstalk (EC2 VM)
Page 1
https://www.draw.io
nginx
server
worker
client

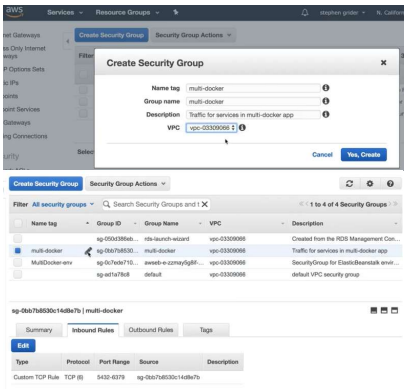

Page 1
https://www.draw.io
Load Balancer
nginx
server
worker
client

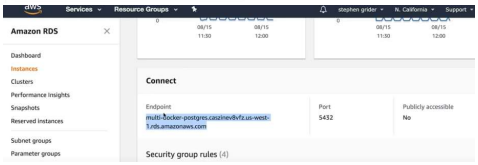

Page 1
draw.io
Load Balancer
nginx v1
server v1
client v1
client v2
worker v1

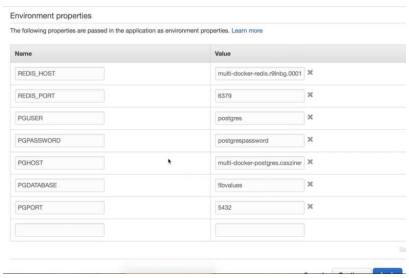

Request
Load Balancer
Node
Container
Master

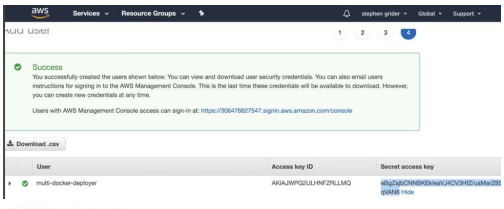

Amazon RDS
Instances
Create database
multi-docker-postgres

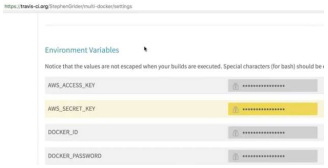

ElastiCache Dashboard
Create
Backup
Reboot
Delete
Modify
multi-docker-redis


Create Security Group
Name tag
Group name
Description
VPC
Cancel
Yes, Create

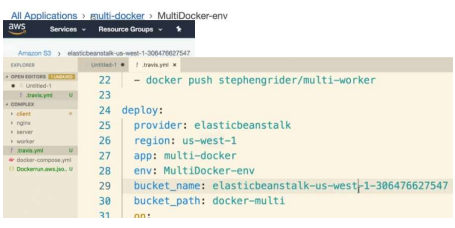

Amazon RDS
Connect
Endpoint
Port
Publicly accessible
Security group rules

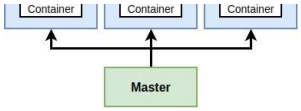

Environment properties
REDIS_HOST
REDIS_PORT
PGUSER
PGPASSWORD
PGHOST
PGDATABASE
PGPORT

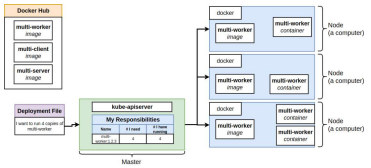

Success
Download .csv
User
Access key ID
Secret access key
multi-docker-deployer


Environment Variables
AWS_ACCESS_KEY
AWS_SECRET_KEY
DOCKER_ID
DOCKER_PASSWORD


Elastic Beanstalk
multi-docker


All Applications
multi-docker
MultiDocker-env
22 - docker push stephengrider/multi-worker
23
24 deploy:
25 provider: elasticbeanstalk
26 region: us-west-1
27 app: multi-docker
28 env: MultiDocker-env
29 bucket_name: elasticbeanstalk-us-west-1-306476627547
30 bucket_path: docker-multi





Important Takeaways

Kubernetes is a system to deploy containerized apps
Nodes are individual machines (or vm's) that run containers
Masters are machines (or vm's) with a set of programs to manage nodes
Kubernetes didn't build our images - it got them from somewhere else
Kubernetes (the master) decided where to run each container - each nodecan run a dissimilar set of containers
To deploy something, we update the desired state of the master with a config file
The master works constantly to meet your desired state

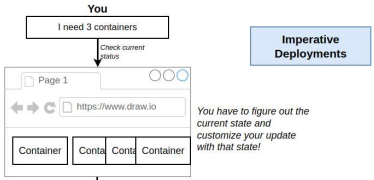
The important stuff

Imperative Deployments

'Do exactly these steps to arrive at this container setup'

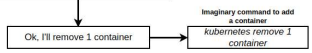
Declarative Deployments

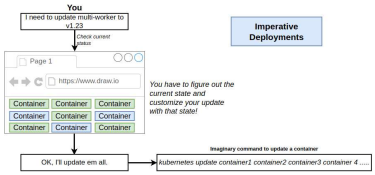
'Our container setup should look like this, make it happen'



Imperative Deployments

You have to figure out the current state and customize your update with that state!

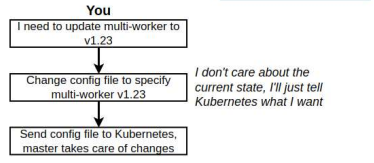




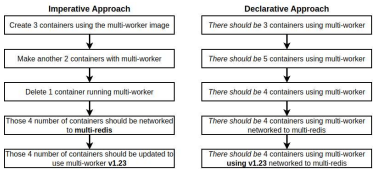
Imperative Deployments

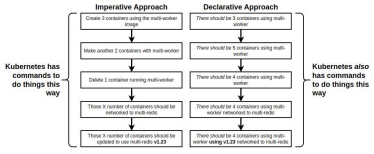
You have to figure out the current state and customize your update with that state!

Declarative

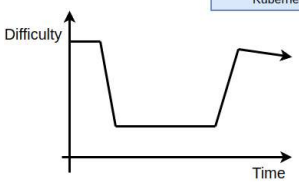


I don't care about the current state, I'll just tell Kubernetes what I want

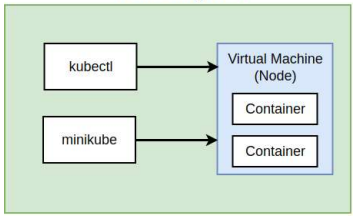


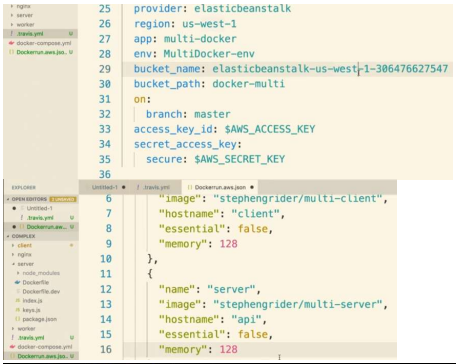


Difficulty curve of learning Kubernetes



Your Computer

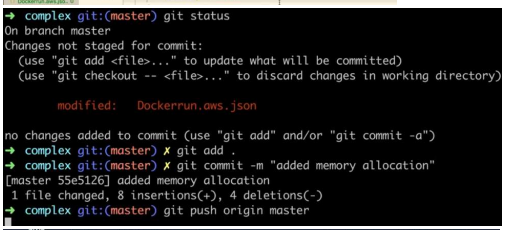




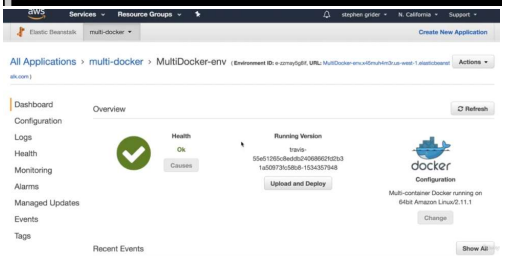
```
provider: elasticbeanstalk
region: us-west-1
app: multi-docker
env: MultiDocker-env
bucket_name: elasticbeanstalk-us-west-1-306476627547
bucket_path: docker-multi
on:
  branch: master
access_key_id: $AWS_ACCESS_KEY
secret_access_key:
  secure: $AWS_SECRET_KEY
```

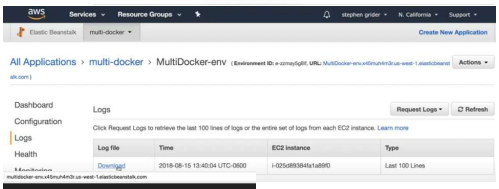


```
    "image": "stephengrider/multi-client",
    "hostname": "client",
    "essential": false,
    "memory": 128
  },
  {
    "name": "server",
    "image": "stephengrider/multi-server",
    "hostname": "api",
    "essential": false,
    "memory": 128
```



```
➜  complex git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Dockerrun.aws.json

no changes added to commit (use "git add" and/or "git commit -a")
➜  complex git:(master) ✗ git add .
➜  complex git:(master) ✗ git commit -m "added memory allocation"
[master 55e5126] added memory allocation
 1 file changed, 8 insertions(+), 4 deletions(-)
➜  complex git:(master) git push origin master
```



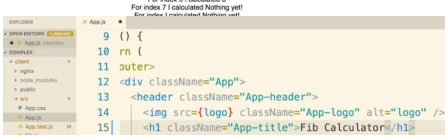


Indexes I have seen:

5, 7

Calculated Values:

For index 5 I calculated 8
For index 7 I calculated Nothing yet!



```
() {
rn (
outer>
<div className="App">
  <header className="App-header">
    <img src={logo} className="App-logo" alt="logo" />
    <h1 className="App-title">Fib Calculator</h1>
```

Local Kubernetes Development

Install Kubectl — CLI for interacting with our master

Install a VM driver virtualbox — Used to make a VM that will be your single node

Install minikube — Runs a single node on that VM

MacOS Install

Make sure Brew is installed	Download brew @ brew.sh
Install Kubectl	*brew install kubectl*
Install a VM driver virtualbox	Download VirtualBox @ virtualbox.org
Install minikube	*brew cask install minikube*
Start minikube	*minikube start*

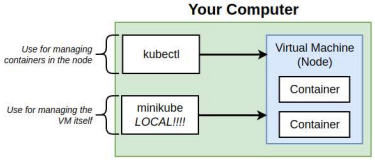
First Deployment

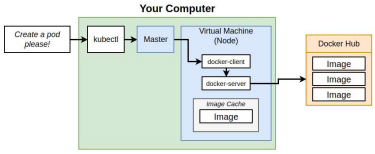
Copy the 'complex' project folder

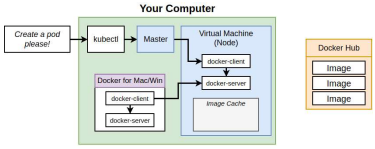
Create a 'deployment' file to deploy the **multi-client** image

Send the deployment file to master by using 'kubectl'

Your Computer







Configure the VM to Use Your Docker Server

eval $(minikube docker-env)

This only configures your current terminal window.

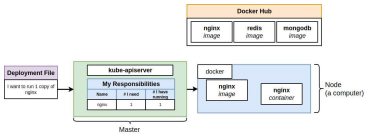
You can look up how to run a command in your terminal every time you open a window to fix this

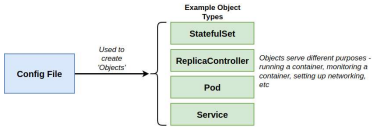
Why mess with Docker in the Node?

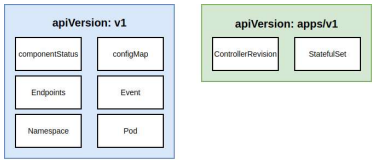
Use all the same debugging techniques we learned with Docker CLI — Many of these commands are available through kubectl

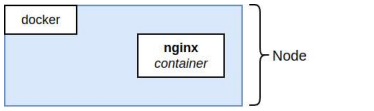
Manually kill containers to test Kubernetes ability to 'self-heal'

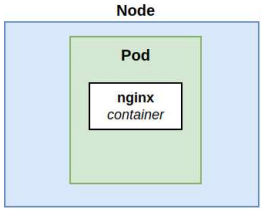
Delete cached images in the node

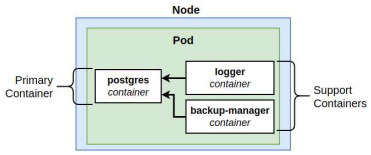


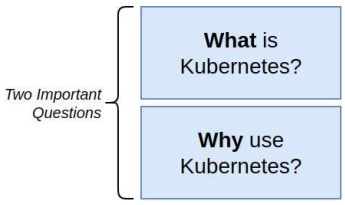


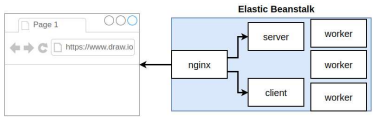


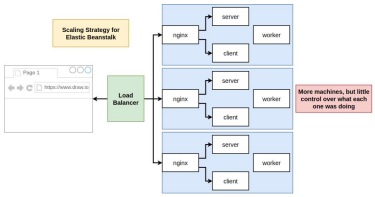


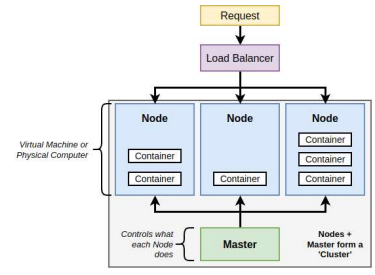

Node
Pod
nginx
container

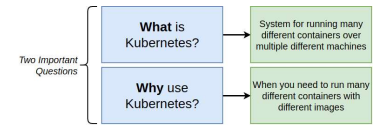

Node
Pod
Primary Container
postgres container
logger container
backup-manager container
Support Containers

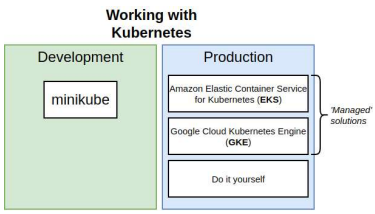

Two Important Questions
What is Kubernetes?
Why use Kubernetes?

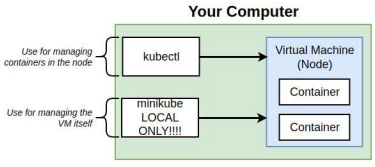

Elastic Beanstalk
Page 1
https://www.draw.io
nginx
server
client
worker
worker
worker

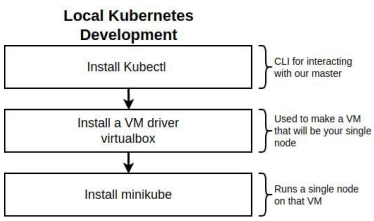

Scaling Strategy for Elastic Beanstalk
Load Balancer
More machines, but little control over what each one was doing

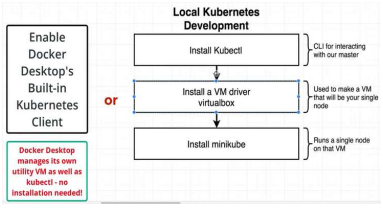

Request
Load Balancer
Virtual Machine or Physical Computer
Node
Container
Controls what each Node does
Master
Nodes + Master form a 'Cluster'


Two Important Questions
What is Kubernetes?
Why use Kubernetes?
System for running many different containers over multiple different machines
When you need to run many different containers with different images


Working with Kubernetes
Development
minikube
Production
Amazon Elastic Container Service for Kubernetes (EKS)
Google Cloud Kubernetes Engine (GKE)
Do it yourself
'Managed' solutions


Your Computer
Use for managing containers in the node
kubectl
Virtual Machine (Node)
Container
Container
Use for managing the VM itself
minikube LOCAL ONLY!!!!

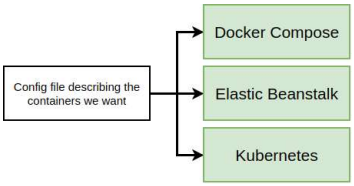

Local Kubernetes Development
Install Kubectl
CLI for interacting with our master
Install a VM driver virtualbox
Used to make a VM that will be your single node
Install minikube
Runs a single node on that VM

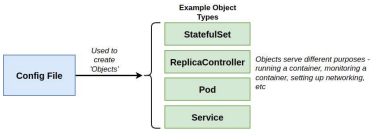


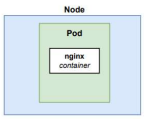
Goal

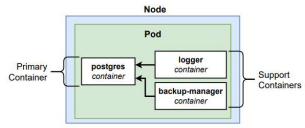
Get a simple container running on kubernetes locally

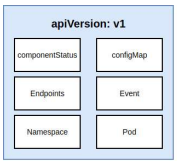
Then talk about how Kubernetes works and what its doing

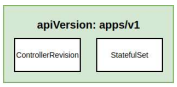




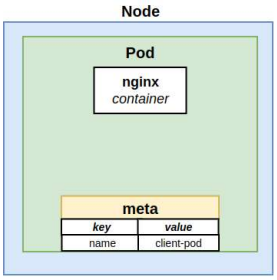




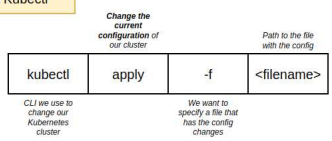




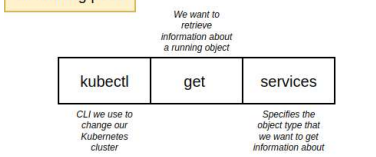
Each API version defines a different set of 'objects' we can use

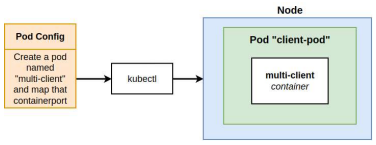


Feed a config file to Kubectl

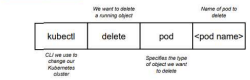


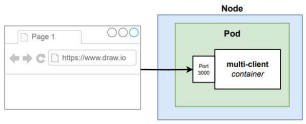
Print the status of all running pods



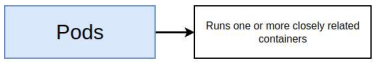


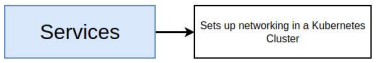
Remove a Pod



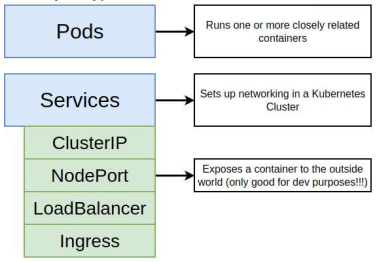


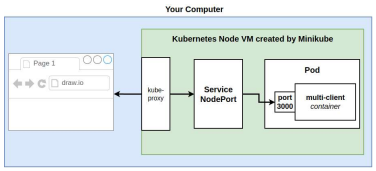
Object Types

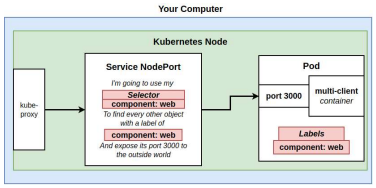


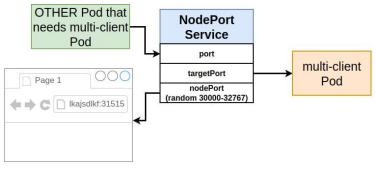


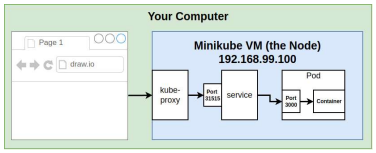
Object Types











Goal

Get the **multi-client** image running on our local Kubernetes Cluster running as a container

docker-compose.yaml

```
services:
  nginx:
```

```
➜ prod git:(master) ls
complex     diagrams    frontend    redis-image simpleweb
➜ prod git:(master) mkdir simplek8s
➜ prod git:(master) cd simple
cd: no such file or directory: simple
➜ prod git:(master) cd simplek8s
➜ simplek8s git:(master) code .
```

```
EXPLORER
OPEN EDITORS
  client-pod.y... U
  client-node-... U
SIMPLEK8S
  client-node-por... U
  client-pod.yaml U

client-pod.yaml ×   client-node-port.yaml

1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: client-pod
5    labels:
```

docker-compose.yaml

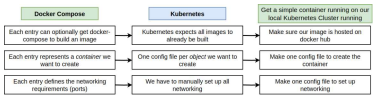
```
services:
  nginx:
    build
    ports
  worker:
    build
    ports
  client:
    build
    ports
```

Each entry can optionally get docker-compose to build an image

Each entry represents a *container* we want to create

Each entry defines the networking requirements (ports)

Docker Compose	Kubernetes
Each entry can optionally get docker-compose to build an image	Kubernetes expects all images to already be built
Each entry represents a *container* we want to create	One config file per *object* we want to create
Each entry defines the networking requirements (ports)	We have to manually set up all networking



Get detailed info about an object

We want to get detailed info | *Name of object*

kubectl	describe	<object type>	<object name>

CLI we use to change our Kubernetes cluster | *Specifies the type of object we want to get info about*

Old Goal	New Goal
Get the **multi-client** image running on our local Kubernetes Cluster running as a container	Update our existing pod to use the **multi-worker** image

Imperative	Declarative
Run a command to list out current running pods	Update our config file that originally created the pod
Run a command to update the current pod to use a new image	Throw the updated config file into kubectl

Updating an Object

Updated Config File → kubectl → Master → Virtual Machine (Node) → Pod *Name: client-pod* → **multi-client** *container*

Your Computer

Updating an Object

Updated Config File (Name: client-pod, Kind: pod, image: multi-worker) → kubectl → Master → Virtual Machine (Node) → Pod *Name: client-pod* → **multi-client** *container*

Your Computer

Pod Config

containers:	1
name:	client
port	3000
image	multi-worker

Can't be updated!

Object Types

Pods	Runs one or more closely related containers
Services	Sets up networking in a Kubernetes Cluster
Deployment	Maintains a set of identical pods, ensuring that they have the correct config and that the right number exists

Pods	Deployment
Runs a single set of containers	Runs a set of identical pods (one or more)
Good for one-off dev purposes	Monitors the state of each pod, updating as necessary
Rarely used directly in production	Good for dev
	Good for production

Deployment

Pod Template:

Pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: client-pod
  labels:
    component: web
spec:
  containers:
    - name: client
      image: stephengrider/multi-client
      ports:
        - containerPort: 3000
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: client-node-port
spec:
  type: NodePort
  ports:
    - port: 3050
      targetPort: 3000
      nodePort: 31515
  selector:
    component: web
```

```
➜ simplek8s git:(master) ✗ ls
client-node-port.yaml client-pod.yaml
➜ simplek8s git:(master) ✗ kubectl apply -f client-pod.yaml
pod/client-pod configured
➜ simplek8s git:(master) ✗ kubectl apply -f client-node-port.yaml
service/client-node-port configured
➜ simplek8s git:(master) ✗ kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
client-pod    1/1       Running   0          1h
➜ simplek8s git:(master) ✗ kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
client-node-port  NodePort    10.103.27.201   <none>        3050:31515/TCP   1h
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP          4d
➜ simplek8s git:(master) ✗
```

```
➜ simplek8s git:(master) ✗ minikube ip
192.168.99.100
➜ simplek8s git:(master) ✗
```

Docker Desktop Kubernetes users should access at localhost:31515

```
➜ simplek8s git:(master) ✗ docker kill 407737e06067
407737e06067
➜ simplek8s git:(master) ✗ docker ps
CONTAINER ID        IMAGE                          COMMAND
      CREATED             STATUS              PORTS               NAMES
28314cca7f3f        stephengrider/multi-client   "nginx -g 'daemon o
f…"   2 seconds ago       Up 2 seconds
➜ simplek8s git:(master) ✗ kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
client-pod    1/1       Running   1          1h
➜ simplek8s git:(master) ✗
➜ simplek8s git:(master) ✗ ls
client-node-port.yaml client-pod.yaml
➜ simplek8s git:(master) ✗ kubectl apply -f client-pod.yaml
pod/client-pod configured
➜ simplek8s git:(master) ✗ kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
client-pod    1/1       Running   2          19h
➜ simplek8s git:(master) ✗
NAME          READY     STATUS    RESTARTS   AGE
client-pod    1/1       Running   2          19h
➜ simplek8s git:(master) ✗ kubectl describe pod client-pod
Name:         client-pod
Namespace:    default
Node:         minikube/10.0.2.15
Start Time:   Mon, 20 Aug 2018 15:31:51 -0600
Labels:       component=web
Annotations:  kubectl.kubernetes.io/last-applied-configuration=
Version":"v1","kind":"Pod","metadata":{"annotations":{},"labels"
omponent":"web"},"name":"client-pod","namespace":"default"},"spe
"container...
Status:       Running
IP:           172.17.0.6
Containers:
  client:
    Container ID:   docker://9b7eabc7cabc2ef3a15b
6cc7b7847252e5da237d518d1
    Image:          stephengrider/multi-worker
```

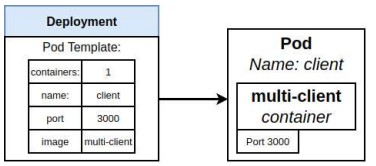
```
image: stephengrider/multi-worker
ports:
  - containerPort: 9999
```

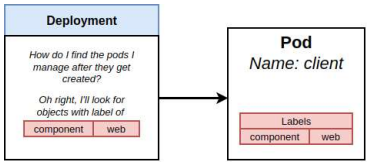
```
➜ simplek8s git:(master) ✗ kubectl apply -f client-pod.yaml
The Pod "client-pod" is invalid: spec: Forbidden: pod updates may no
t change fields other than `spec.containers[*].image`, `spec.initCon
tainers[*].image`, `spec.activeDeadlineSeconds` or `spec.tolerations
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: web
  template:
    metadata:
      labels:
        component: web
    spec:
      containers:
        - name: client
          image: stephengrider/multi-client
          ports:
            - containerPort: 3000
```

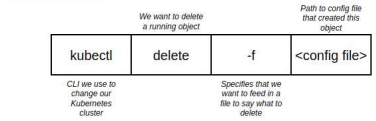
```
➜ simplek8s git:(master) ✗ kubectl get pods
NAME          READY     STATUS    RESTARTS   AGE
client-pod    1/1       Running   0          1m
➜ simplek8s git:(master) ✗ kubectl delete -f client-pod.yaml
pod "client-pod" deleted
➜ simplek8s git:(master) ✗ kubectl get pods
No resources found.
➜ simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment created
➜ simplek8s git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS
AGE
client-deployment-588947887b-km66g   1/1       Running   0
8s
➜ simplek8s git:(master) ✗
➜ simplek8s git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   1         1         1            1           38s
➜ simplek8s git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS
AGE
client-deployment-588947887b-km66g   1/1       Running   0
2m
```

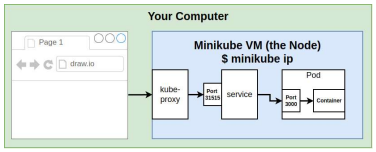
Good for production

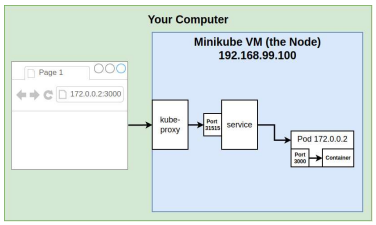


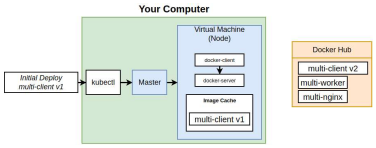


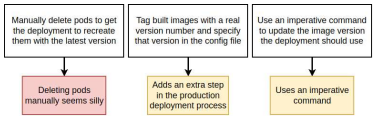
Remove an object



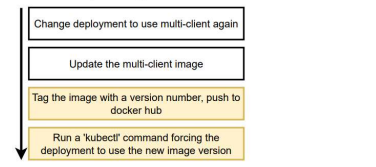


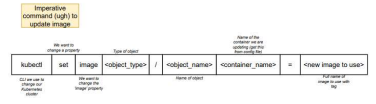


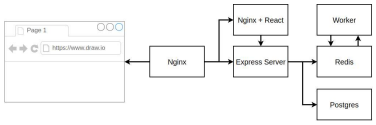




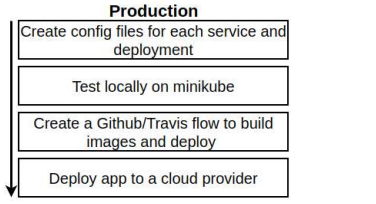
Update Image Version



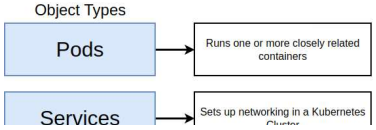




Path to Production



Object Types




```
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment     1         1         1            1           38s
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                  READY     STATUS    RESTARTS
AGE
client-deployment-588947887b-km66g    1/1       Running   0
2m
➜  simplek8s git:(master) ✗ minikube ip
192.168.99.100
```


```
➜  simplek8s git:(master) ✗ kubectl get pods -o wide
NAME                  READY     STATUS    RESTARTS   AGE       IP            NODE
client-deployment-588947887b-km66g   1/1       Running   0          9m        172.17.0.6   minikube
```

```
 10   template:
 11     metadata:
 12       labels:
 13         component: web
 14     spec:
 15       containers:
 16         - name: client
 17           image: stephengrider/multi-client
 18           ports:
 19             - containerPort: 9999
```

```
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
➜  simplek8s git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   1         1         1            1           20m
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                  READY     STATUS    RESTARTS   AGE
client-deployment-9b8bffb65-zrk5q     1/1       Running   0          26s
➜  simplek8s git:(master) ✗ kubectl describe pods
```

```
  1 apiVersion: apps/v1
  2 kind: Deployment
  3 metadata:
  4   name: client-deployment
  5 spec:
  6   replicas: 5
```

```
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
➜  simplek8s git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   5         5         5            5           22m
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                  READY     STATUS    RESTARTS   AGE
client-deployment-9b8bffb65-hqtt4     1/1       Running   0          32s
client-deployment-9b8bffb65-qw4g5     1/1       Running   0          32s
client-deployment-9b8bffb65-qx6fb     1/1       Running   0          32s
client-deployment-9b8bffb65-trmsr     1/1       Running   0          32s
client-deployment-9b8bffb65-zrk5q     1/1       Running   0          2m
```

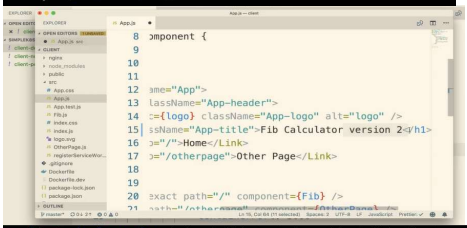
```
  7   selector:
  8     matchLabels:
  9       component: web
 10   template:
 11     metadata:
 12       labels:
 13         component: web
 14     spec:
 15       containers:
 16         - name: client
 17           image: stephengrider/multi-worker
 18           ports:
 19             - containerPort: 9999
```

```
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
➜  simplek8s git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   5         7         4            4           23m
➜  simplek8s git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   5         5         5            5           24m
➜  simplek8s git:(master) ✗
```

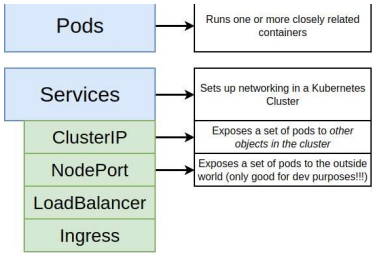
```
  5 spec:
  6   replicas: 1
  7   selector:
  8     matchLabels:
  9       component: web
 10   template:
 11     metadata:
 12       labels:
 13         component: web
 14     spec:
 15       containers:
 16         - name: client
 17           image: stephengrider/multi-client
 18           ports:
 19             - containerPort: 3000
```

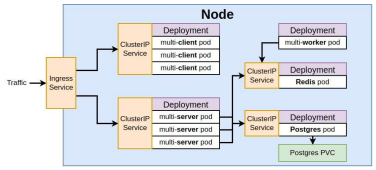
```
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
➜  simplek8s git:(master) ✗ minikube ip
192.168.99.100
➜  simplek8s git:(master) ✗
```

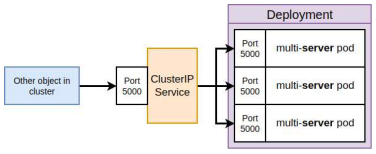

```
Last login: Tue Aug 21 13:57:40 on ttys003
➜  simplek8s git:(master) ✗ cd ..
➜  prod git:(master) ✗ ls
complex      diagrams     frontend     redis-image simplek8s    simpleweb
➜  prod git:(master) ✗ cd complex
➜  complex git:(master) ✗ ls
Dockerrun.aws.json docker-compose.yml server
client             nginx              worker
➜  complex git:(master) ✗ cd client
➜  client git:(master) ✗ code .
➜  client git:(master) ✗
```

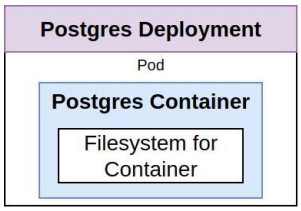


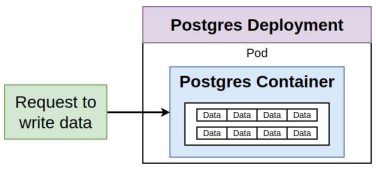

```
➜  client git:(master) ✗ docker build -t stephengrider/multi-client .
Sending build context to Docker daemon    574kB
Step 1/10 : FROM node:alpine as builder
 ---> 5a519d1e3a24
Step 2/10 : WORKDIR '/app'
 ---> Using cache
 ---> eb34bacff3ad
Step 3/10 : COPY ./package.json ./
 ---> Using cache
 ---> aa0eed977062
Step 4/10 : RUN npm install
 ---> Using cache
 ---> e61411b83dd1
Step 5/10 : COPY . .
```

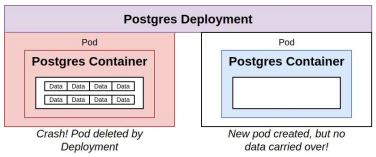


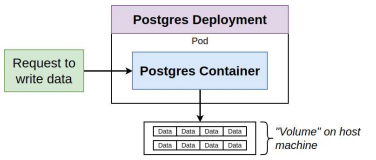


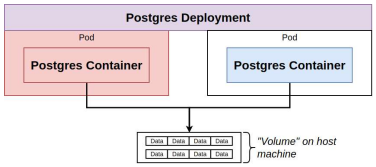


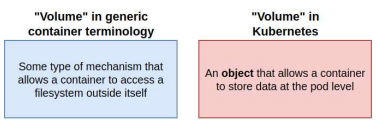


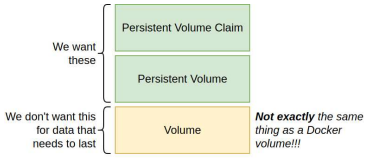


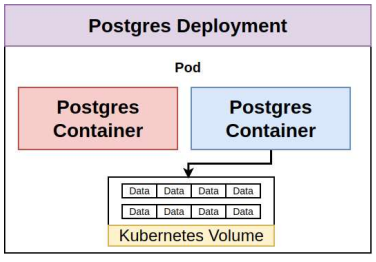













```
Step 3/10 : COPY ./package.json ./
 ---> Using cache
 ---> aa0eed977062
Step 4/10 : RUN npm install
 ---> Using cache
 ---> e61411b83dd1
Step 5/10 : COPY . .
 ---> 9058788e6880
Step 6/10 : RUN npm run build
 ---> Running in be8158ecf5ea
```

```
 ---> edf8c3b65171
Step 9/10 : COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> 63fea4bcfe12
Step 10/10 : COPY --from=builder /app/build /usr/share/nginx/html
 ---> 96aae3282fff
Successfully built 96aae3282fff
Successfully tagged stephengrider/multi-client:latest
➜  client git:(master) ✗ docker push stephengrider/multi-client
The push refers to repository [docker.io/stephengrider/multi-client]
51bfb905f2b3: Pushed
694a6abf5f0d: Layer already exists
08d25fa0442e: Layer already exists
a8c4aeeaa045: Layer already exists
cdb3f9544e4c: Layer already exists
latest: digest: sha256:5e2a93a2bcd1ab55756fca4de120ca3ad13ac7965ae1cd9435f6
cc9f464cec4e size: 1365
➜  client git:(master) ✗
```

We updated our image and pushed to docker hub now to update our deployment:
Don't do this, see below for solution

```
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment unchanged
➜  simplek8s git:(master) ✗
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-822hv   1/1       Running   0          38m
➜  simplek8s git:(master) ✗
```

```
➜  client git:(master) ✗ docker build -t stephengrider/multi-client:v4 .
```

```
 7    selector:
 8      matchLabels:
 9        component: web
10    template:
11      metadata:
12        labels:
13          component: web
14      spec:
15        containers:
16          - name: client
17            image: stephengrider/multi-client:v4
18            ports:
19              - containerPort: 3000
```

```
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-822hv   1/1       Running   0          38m
➜  simplek8s git:(master) ✗ kubectl apply -f client-deployment.yaml
```

```
 9    component: web
10  strategy:
11    type: RollingUpdate
12  template:
13    metadata:
14      labels:
15        component: web
16    spec:
17      containers:
18        - name: client
19          image: stephengrider/multi-client:${CLIENT_VERSION}
20          ports:
21            - containerPort: 3000
```

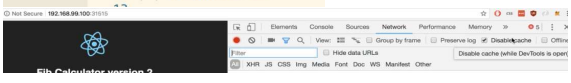
Not allowed

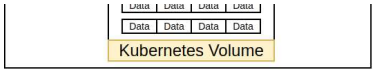
Solution to update image

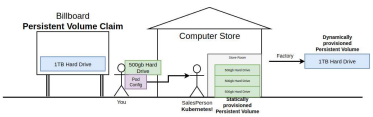
```
Last login: Tue Aug 21 16:18:36 on ttys002
➜  simplek8s git:(master) ✗ cd ..
➜  prod git:(master) ✗ ls
complex     diagrams    frontend    redis-image simplek8s   simpleweb
➜  prod git:(master) ✗ cd complex
➜  complex git:(master) ✗ cd client
➜  client git:(master) ✗ ls
Dockerfile          nginx               package.json
Dockerfile.dev      node_modules        public
README.md           package-lock.json src
➜  client git:(master) ✗ docker tag
➜  client git:(master) ✗
➜  client git:(master) ✗ docker build -t stephengrider/multi-client:v5 .
Sending build context to Docker daemon    574kB
Step 1/10 : FROM node:alpine as builder
 ---> 5a519d1e3a24
Step 2/10 : WORKDIR '/app'
 ---> Using cache
 ---> eb34bacff3ad
Step 3/10 : COPY ./package.json ./
 ---> Using cache
 ---> aa0eed977062
Step 4/10 : RUN npm install
 ---> Using cache
 ---> e61411b83dd1
Step 5/10 : COPY . .
```

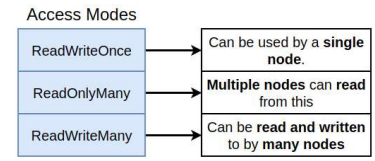
```
Step 9/10 : COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
 ---> Using cache
 ---> 63fea4bcfe12
Step 10/10 : COPY --from=builder /app/build /usr/share/nginx/html
 ---> Using cache
 ---> 96aae3282fff
Successfully built 96aae3282fff
Successfully tagged stephengrider/multi-client:v5
➜  client git:(master) ✗ docker push stephengrider/multi-client:v5
The push refers to repository [docker.io/stephengrider/multi-client]
51bfb905f2b3: Layer already exists
694a6abf5f0d: Layer already exists
08d25fa0442e: Layer already exists
a8c4aeeaa045: Layer already exists
cdb3f9544e4c: Layer already exists
v5: digest: sha256:5e2a93a2bcd1ab55756fca4de120ca3ad13ac7965ae1cd9435f6cc9f
464cec4e size: 1365
➜  client git:(master) ✗
➜  simplek8s git:(master) ✗ kubectl set image deployment/client-deployment
client=stephengrider/multi-client:v5
deployment.extensions/client-deployment image updated
➜  simplek8s git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-58d959cb75-rqvwg   1/1       Running   0          7s
➜  simplek8s git:(master) ✗ minikube ip
192.168.99.100
➜  simplek8s git:(master) ✗
```

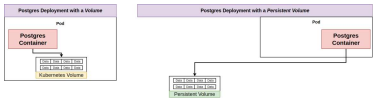
```
 1  apiVersion: v1
 2  kind: Service
 3  metadata:
 4    name: client-node-port
 5  spec:
 6    type: NodePort
 7    ports:
 8      - port: 3050
 9        targetPort: 3000
10        nodePort: 31515
11    selector:
12      component: web
```

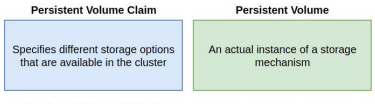


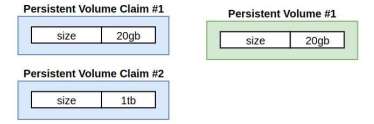


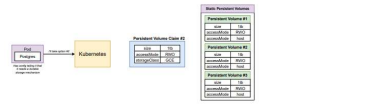


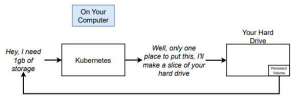


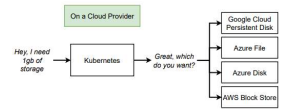


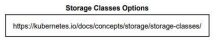


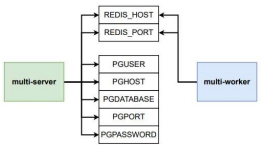


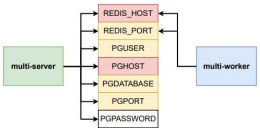


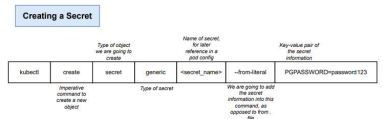


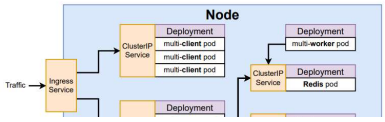




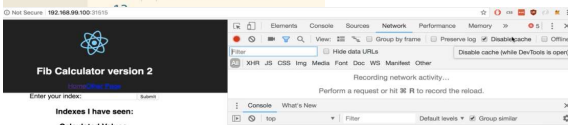









```
      component: web
```




```
➜ simplek8s git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
    STATUS              PORTS               NAMES
➜ simplek8s git:(master) eval $(minikube docker-env)
➜ simplek8s git:(master) minikube docker-env
export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.99.100:2376"
export DOCKER_CERT_PATH="/Users/stephengrider/.minikube/certs"
export DOCKER_API_VERSION="1.35"
# Run this command to configure your shell:
# eval $(minikube docker-env)
➜ simplek8s git:(master) docker ps
```

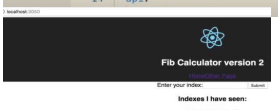
```
➜ simplek8s git:(master) docker logs cbee3ab1cf75
➜ simplek8s git:(master) docker exec -it cbee3ab1cf75 sh
# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
# exit
➜ simplek8s git:(master)
```

```
➜ simplek8s git:(master) kubectl get pods
NAME                                  READY     STATUS    RESTARTS   AGE
client-deployment-58d959cb75-rqvwg    1/1       Running   1          19h
➜ simplek8s git:(master) kubectl logs client-deployment-58d959cb75-rqvwg
➜ simplek8s git:(master) kubectl exec -it client-deployment-58d959cb75-rqv
wg sh
# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
# exit
➜ simplek8s git:(master) docker system prune -a
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all images without at least one container associated to them
        - all build cache
Are you sure you want to continue? [y/N]
untagged: stephengrider/multi-client
untagged: stephengrider/multi-client
untagged: stephengrider/multi-client
4b18978c580eedee048246932152e538e
deleted: sha256:79dbda5ee9d1a2dc1a8c
34eba
deleted: sha256:8ba6ddf184366c65ac2e
3dba8

Total reclaimed space: 2.373MB
➜ simplek8s git:(master)
```

```
➜ complex git:(master) ls
Dockerrun.aws.json docker-compose.yml server
client             nginx              worker
➜ complex git:(master) docker-compose up --build
Building worker
Step 1/6 : FROM node:alpine
 ---> 5a519d1e3a24
Step 2/6 : WORKDIR "/app"
```

```
➜ complex git:(master) docker-compose up
Starting complex_client_1   ... done
Starting complex_api_1      ... done
Starting complex_worker_1   ... done
Starting complex_postgres_1 ... done
Starting complex_redis_1    ... done
Starting complex_nginx_1    ... done
```

```
version: '3'
services:
  postgres:
    image: 'postgres:latest'
  redis:
    image: 'redis:latest'
  nginx:
    restart: always
    build:
      dockerfile: Dockerfile.dev
      context: ./nginx
    ports:
      - '3050:80'
  api:
```




```
➜ complex git:(master) ✗ cd ..
➜ prod git:(master) ✗ ls
complex                 redis-image
complex-elastic-beanstalk simplek8s
diagrams                simpleweb
frontend
➜ prod git:(master) ✗ cd complex
➜ complex git:(master) ✗
```

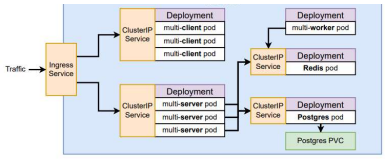
```
apiVersion: apps/v1
kind: Deployme
metadata:
  name: client
spec:
  replicas: 3
  selector:
    matchLabel
      component: web
  template:
    metadata:
      label:
        component: web
```

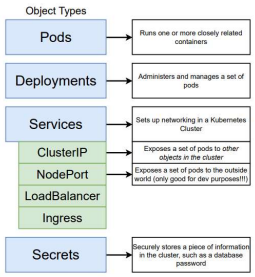
Ack, typo here! That should be "labels" with an "s" on the end

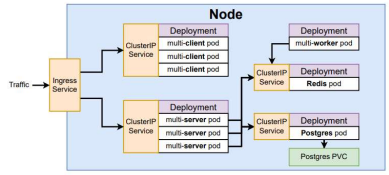
```
    spec:
      containers:
        - name: client
          image: stephengrider/multi-client
          ports:
            - containerPort: 3000
```

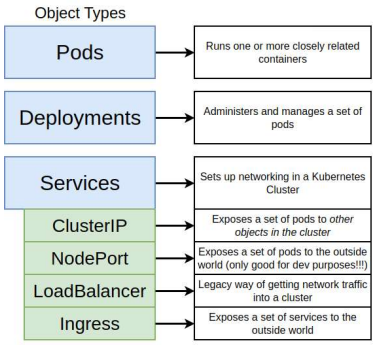
```
apiVersion: v1
kind: Service
metadata:
  name: client-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: web
  ports:
    - port: 3000
      targetPort: 3000
```

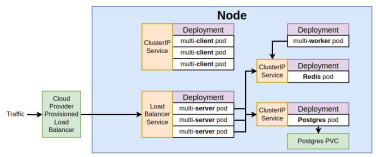
```
➜ complex git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   1         1         1            1           23h
➜ complex git:(master) ✗ kubectl delete deployment client-deployment
deployment.extensions "client-deployment" deleted
➜ complex git:(master) ✗ kubectl get deployments
No resources found.
➜ complex git:(master) ✗ kubectl get services
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
client-node-port    NodePort    10.103.27.201    <none>        3050:31515/TCP
```

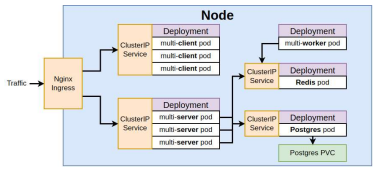


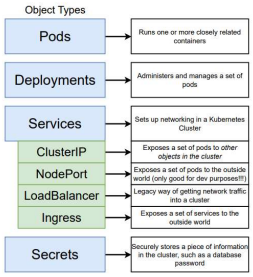


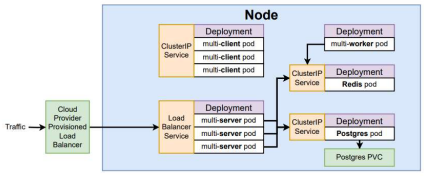


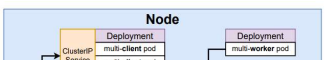













```
deployment.extensions "client-deployment" deleted
→ complex git:(master) ✗ kubectl get deployments
No resources found.
→ complex git:(master) ✗ kubectl get services
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
client-node-port    NodePort    10.103.27.201   <none>        3050:31515/TCP
   1d
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP
   5d
→ complex git:(master) ✗ kubectl delete service client-node-port\
>
service "client-node-port" deleted
→ complex git:(master) ✗
```

```
→ complex git:(master) ✗ kubectl apply -f k8s/client-deployment.yaml
→ complex git:(master) ✗ ls
client k8s    server worker
→ complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service created
error: error validating "k8s/client-deployment.yaml": error validating data
: ValidationError(Deployment.spec.template.metadata): unknown field "label"
 in io.k8s.apimachinery.pkg.apis.meta.v1.ObjectMeta; if you choose to ignor
e these errors, turn validation off with --validate=false
→ complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment created
→ complex git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   3         3         3            3           9s
→ complex git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-gfzlt   1/1       Running   0          16s
client-deployment-588947887b-hdccm   1/1       Running   0          16s
client-deployment-588947887b-r9p7l   1/1       Running   0          16s
→ complex git:(master) ✗ kubectl get services
NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT
(S)    AGE
client-cluster-ip-service  ClusterIP   10.109.237.218   <none>        3000
/TCP   47s
kubernetes                 ClusterIP   10.96.0.1        <none>        443/
TCP    5d
→ complex git:(master) ✗
```

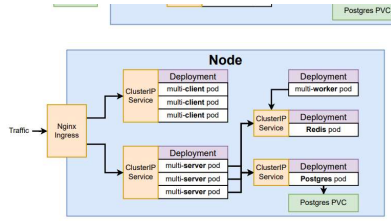
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: server-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      component: server
  template:
    metadata:
      labels:
        component: server
    spec:
      containers:
        - name: server
          image: stephengrider/multi-server
          ports:
            - containerPort: 5000
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: server-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: server
  ports:
    - port: 5000
      targetPort: 5000
```

```yaml
        - name: server
          image: stephengrider/multi-server
          ports:
            - containerPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: server-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: server
  ports:
    - port: 5000
      targetPort: 5000
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: worker-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: worker
  template:
    metadata:
      labels:
        component: worker
    spec:
      containers:
        - name: worker
          image: stephengrider/multi-worker
```

```
→ complex git:(master) ✗ ls
client k8s    server worker
→ complex git:(master) ✗ ls k8s
client-cluster-ip-service.yaml server-deployment.yaml
client-deployment.yaml         worker-deployment.yaml
server-cluster-ip-service.yaml
→ complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
service/server-cluster-ip-service created
deployment.apps/server-deployment created
deployment.apps/worker-deployment created
→ complex git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-gfzlt   1/1       Running   2          22h
client-deployment-588947887b-hdccm   1/1       Running   2          22h
client-deployment-588947887b-r9p7l   1/1       Running   2          22h
server-deployment-58f77c7889-5gjl4   1/1       Running   0          25s
server-deployment-58f77c7889-c8dmd   1/1       Running   0          25s
server-deployment-58f77c7889-grrzn   1/1       Running   0          25s
worker-deployment-778459cd54-sqjdv   1/1       Running   0          25s
→ complex git:(master) ✗ kubectl get deployments
NAME                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
client-deployment   3         3         3            3           22h
server-deployment   3         3         3            3           40s
worker-deployment   1         1         1            1           40s
→ complex git:(master) ✗ kubectl get services
NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT
(S)    AGE
client-cluster-ip-service  ClusterIP   10.109.237.218   <none>        3000
/TCP   22h
kubernetes                 ClusterIP   10.96.0.1        <none>        443/
TCP    6d
server-cluster-ip-service  ClusterIP   10.96.150.142    <none>        5000
/TCP   46s
→ complex git:(master) ✗
→ complex git:(master) ✗ kubectl logs server-deployment-58f77c7889-5gjl4

> @ start /app
> node index.js

Listening
```



We are using **ingress-nginx,** a community led project

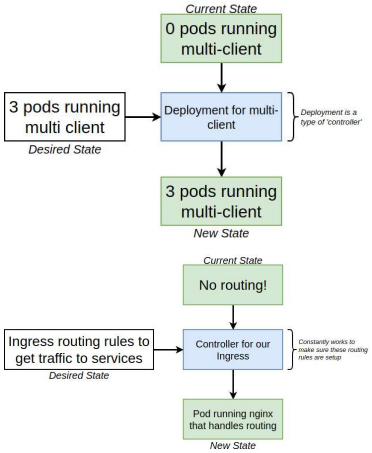
github.com/kubernetes/ingress-nginx

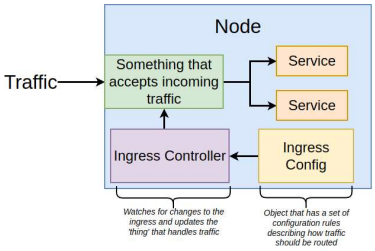
We are ***not*** using **kubernetes-ingress**, a project led by the company nginx

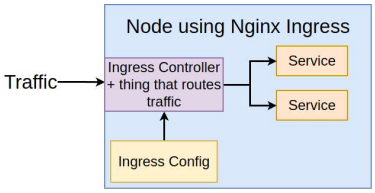
github.com/nginxinc/kubernetes-ingress

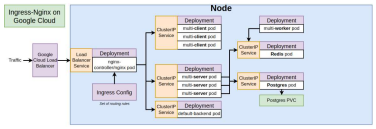
Setup of ingress-nginx changes depending on your environment (local, GC, AWS, Azure)

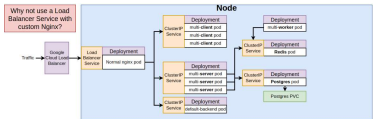
We are going to set up ingress-nginx on local and GC

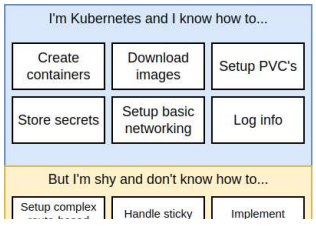













```
→ complex git:(master) ✗
→ complex git:(master) ✗ kubectl logs server-deployment-58f77c7889-5gjl4

> @ start /app
> node index.js

Listening
[ Error: connect ECONNREFUSED 127.0.0.1:5432
    at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1163:14)
  errno: 'ECONNREFUSED',
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 5432 ]
→ complex git:(master) ✗
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: redis
  template:
    metadata:
      labels:
        component: redis
    spec:
      containers:
        - name: redis
          image: redis
          ports:
            - containerPort: 6379
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: redis-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: redis
  ports:
    - port: 6379
      targetPort: 6379
```

```
→ complex git:(master) ls
client k8s    server worker
→ complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
service/redis-cluster-ip-service created
deployment.apps/redis-deployment created
service/server-cluster-ip-service unchanged
deployment.apps/server-deployment unchanged
deployment.apps/worker-deployment unchanged
→ complex git:(master) ✗
→ complex git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-gfzlt   1/1       Running   3          23h
client-deployment-588947887b-hdccm   1/1       Running   3          23h
client-deployment-588947887b-r9p7l   1/1       Running   3          23h
redis-deployment-666bf96bdd-78wsr    1/1       Running   0          15s
server-deployment-58f77c7889-5gjl4   1/1       Running   1          1h
server-deployment-58f77c7889-c8dmd   1/1       Running   1          1h
server-deployment-58f77c7889-grrzn   1/1       Running   1          1h
worker-deployment-778459cd54-sqjdv   1/1       Running   1          1h
→ complex git:(master) ✗
→ complex git:(master) ✗ kubectl get services
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT
(S)    AGE
client-cluster-ip-service   ClusterIP   10.109.237.218   <none>        3000
/TCP   23h
kubernetes                  ClusterIP   10.96.0.1        <none>        443/
TCP    6d
redis-cluster-ip-service    ClusterIP   10.103.196.215   <none>        6379
/TCP   23s
server-cluster-ip-service   ClusterIP   10.96.150.142    <none>        5000
/TCP   1h
→ complex git:(master) ✗
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      component: postgres
  template:
    metadata:
      labels:
        component: postgres
    spec:
      containers:
        - name: postgres
          image: postgres
          ports:
            - containerPort: 5432
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: postgres-cluster-ip-service
spec:
  type: ClusterIP
  selector:
    component: postgres
  ports:
    - port: 5432
      targetPort: 5432
```

```
→ complex git:(master) ✗ ls
client k8s    server worker
→ complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
service/postgres-cluster-ip-service created
deployment.apps/postgres-deployment created
service/redis-cluster-ip-service unchanged
deployment.apps/redis-deployment unchanged
service/server-cluster-ip-service unchanged
deployment.apps/server-deployment unchanged
deployment.apps/worker-deployment unchanged
→ complex git:(master) ✗ kubectl get pods
NAME                                 READY     STATUS              RESTARTS   AGE
client-deployment-588947887b-gfzlt   1/1       Running             3          1d
client-deployment-588947887b-hdccm   1/1       Running             3          1d
client-deployment-588947887b-r9p7l   1/1       Running             3          1d
postgres-deployment-599f4f6bf7-zlxjd 0/1       ContainerCreating   0          9s
redis-deployment-666bf96bdd-78wsr    1/1       Running             0          6m
server-deployment-58f77c7889-5gjl4   1/1       Running             1          1h
server-deployment-58f77c7889-c8dmd   1/1       Running             1          1h
server-deployment-58f77c7889-grrzn   1/1       Running             1          1h
worker-deployment-778459cd54-sqjdv   1/1       Running             1          1h
→ complex git:(master) ✗ kubectl get pods
→ complex git:(master) ✗ kubectl get services
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
client-cluster-ip-service   ClusterIP   10.109.237.218   <none>        3000/TCP
1d
kubernetes                  ClusterIP   10.96.0.1        <none>        443/TCP
6d
postgres-cluster-ip-service ClusterIP   10.102.187.175   <none>        5432/TCP
30s
redis-cluster-ip-service    ClusterIP   10.103.196.215   <none>        6379/TCP
7m
server-cluster-ip-service   ClusterIP   10.96.150.142    <none>        5000/TCP
```

networking

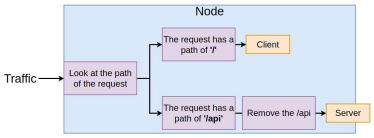
But I'm shy and don't know how to...

Setup complex route-based networking

Handle sticky sessions

Implement caching

So if you want to bring your own web server then that is OK :)



Prod Deployment

- Create Github Repo
- Tie repo to Travis CI
- Create Google Cloud project
- **Enable billing for the project**
- Add deployment scripts to the repo

Why Google Cloud?

- Google created Kubernetes!
- AWS only "recently" got Kubernetes support
- Far, far easier to poke around Kubernetes on Google Cloud
- Excellent documentation for beginners

Google Cloud Console

console.cloud.google.com

Travis Config File

- Install Google Cloud SDK CLI
- Configure the SDK with out Google Cloud auth info
- Login to Docker CLI
- Build the 'test' version of multi-client
- Run tests
- If tests are successful, run a script to deploy newest images
- Build all our images, tag each one, push each to docker hub
- Apply all configs in the 'k8s' folder
- Imperatively set latest images on each deployment

Don't Forget!

Some aspects of our app rely on imperative commands to work correctly!

- Create a Service Account
- Download service account credentials in a json file
- Download and install the Travis CLI
- Encrypt and upload the json file to our Travis account
- In travis.yml, add code to unencrypt the json file and load it into GCloud SDK

Travis CLI

github.com/travis-ci/travis.rb

Requires Ruby to be installed locally

Installing Ruby is a pain!

Let's get a Docker image that has Ruby pre-installed, then we can install travis CLI in there!

- `docker run -it -v ${pwd}:/app ruby:2.3 sh`
- `gem install travis --no-rdoc --no-ri`
- `gem install travis`
- `travis login`
- *Copy json file into the 'volumed' directory so we can use it in the container*
- `travis encrypt-file service-account.json -r StephenGrider/multi-k8s`

`docker run -it -v ${pwd}:/app ruby:2.3 sh`

Use curly braces on windows

```
1d
kubernetes                  ClusterIP   10.96.0.1        <none>        443/TCP
6d
postgres-cluster-ip-service ClusterIP   10.102.187.175   <none>        5432/TCP
30s
redis-cluster-ip-service    ClusterIP   10.103.196.215   <none>        6379/TCP
7m
server-cluster-ip-service   ClusterIP   10.96.150.142    <none>        5000/TCP
1h
➜  complex git:(master) ✗
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-persistent-volume-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

```
➜  complex git:(master) ✗ kubectl get storageclass
NAME                 PROVISIONER                AGE
standard (default)   k8s.io/minikube-hostpath   7d
➜  complex git:(master) ✗
➜  complex git:(master) ✗ kubectl describe storageclass
Name:                  standard
IsDefaultClass:        Yes
Annotations:           kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{"storageclass.beta.kubernetes.io/is-default-class":"true"},"labels":{"addonmanager.kubernetes.io/mode":"Reconcile"},"name":"standard","namespace":""},"provisioner":"k8s.io/minikube-hostpath"}
,storageclass.beta.kubernetes.io/is-default-class=true
Provisioner:           k8s.io/minikube-hostpath
Parameters:            <none>
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>
➜  complex git:(master) ✗
```

```
      labels:
        component: postgres
    spec:
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: database-persistent-volume-claim
      containers:
        - name: postgres
          image: postgres
          ports:
            - containerPort: 5432
          volumeMounts:
            - name: postgres-storage
              mountPath: /var/lib/postgresql/data
              subPath: postgres
```

```
➜  complex git:(master) ✗ ls
client k8s    server worker
➜  complex git:(master) ✗ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
persistentvolumeclaim/database-persistent-volume-claim created
service/postgres-cluster-ip-service unchanged
deployment.apps/postgres-deployment configured
service/redis-cluster-ip-service unchanged
deployment.apps/redis-deployment unchanged
service/server-cluster-ip-service unchanged
deployment.apps/server-deployment unchanged
deployment.apps/worker-deployment unchanged
➜  complex git:(master) ✗
```

```
➜  complex git:(master) ✗ kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
client-deployment-588947887b-gfzlt  1/1       Running   3          1d
client-deployment-588947887b-hdccm  1/1       Running   3          1d
client-deployment-588947887b-r9p7l  1/1       Running   3          1d
postgres-deployment-fdd46dbc8-dklfg 1/1       Running   0          19s
redis-deployment-666bf96bdd-78wsr   1/1       Running   0          23h
server-deployment-58f77c7889-5gjl4  1/1       Running   8          1d
server-deployment-58f77c7889-c8dmd  1/1       Running   8          1d
server-deployment-58f77c7889-grrzn  1/1       Running   8          1d
worker-deployment-778459cd54-sqjdv  1/1       Running   8          1d
```

```
➜  complex git:(master) ✗ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                STORAGECLASS   REASON   AGE
pvc-3e797a7b-a7d2-11e8-bb71-0800277ed29a   2Gi        RWO            Delete           Bound    default/database-persistent-volume-claim   standard               35s
➜  complex git:(master) ✗ kubectl get pvc
NAME                                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
database-persistent-volume-claim    Bound     pvc-3e797a7b-a7d2-11e8-bb71-0800277ed29a   2Gi        RWO            standard       2m
➜  complex git:(master) ✗
```

```
    template:
      metadata:
        labels:
          component: worker
      spec:
        containers:
          - name: worker
            image: stephengrider/multi-worker
            env:
              - name: REDIS_HOST
                value: redis-cluster-ip-service
              - name: REDIS_PORT
                value: 6379
```

```
              - containerPort: 5000
            env:
              - name: REDIS_HOST
                value: redis-cluster-ip-service
              - name: REDIS_PORT
                value: 6379
              - name: PGUSER
                value: postgres
              - name: PGHOST
                value: postgres-cluster-ip-service
              - name: PGPORT
                value: 5432
              - name: PGDATABASE
                value: postgres
```

```
➜  complex git:(master) ✗ kubectl create secret generic pgpassword --from-literal PGPASSWORD=12345asdf
secret/pgpassword created
➜  complex git:(master) ✗ kubectl get secrets
NAME                  TYPE                                  DATA      AGE
default-token-2c228   kubernetes.io/service-account-token   3         8d
pgpassword            Opaque                                1         11s
➜  complex git:(master) ✗
```

```
              value: postgres
              - name: PGHOST
                value: postgres-cluster-ip-service
              - name: PGPORT
                value: 5432
              - name: PGDATABASE
                value: postgres
              - name: PGPASSWORD
                valueFrom:
                  secretKeyRef:
                    name: pgpassword
                    key: PGPASSWORD
```

```
        - name: postgres
          image: postgres
          ports:
            - containerPort: 5432
          volumeMounts:
            - name: postgres-storage
```

travis login

Copy json file into the 'volumed' directory so we can use it in the container

travis encrypt-file service-account.json -r StephenGrider/multi-k8s

DELETE THE ORGINAL SERVICE_ACCOUNT>JSON FILE!!!!

docker build -t docker_id/multi-client -f ./client/Dockerfile ./client

Image

Tag	docker_id/mult-client:*latest*

kubectl set image deployment/multi-client-deployment client=docker_id/multi-client:latest

I was already running 'latest' so no change required!

docker build
-t docker_id/multi-client:latest
-t docker_id/multi-client:$GIT_SHA
-f ./client/Dockerfile ./client

Image

Tag	docker_id/mult-client:*latest*
Tag	docker_id/mult-client:*d859186b88466*

kubectl set image deployment/multi-client-deployment client=**docker_id/multi-client:d859186b88466**

Oh, new image, I'll use that!

Oh no, our app is breaking!!! What is the exact state of the codebase that our deployments are running?

Deployment is running multi-client:832ba92c

git checkout 832ba92c

Debug the app knowing the *exact* code that is running in production

docker build
-t docker_id/multi-client:latest
-t docker_id/multi-client:$GIT_SHA
-f ./client/Dockerfile ./client

Image

Tag	docker_id/mult-client:*latest*
Tag	docker_id/mult-client:*$SHA*

Time Passes...

New Engineer runs *'kubectl apply -f k8s'*

Deployment specifies config file with implicit 'latest' tag ---> latest version used *without* having to worry about specifying a SHA

Helm + Tiller

https://docs.helm.sh/using_helm/#quickstart-guide

Command we issue → Helm Client → Tiller Server

Role Based Access Control (RBAC)

Limits who can access and modify objects in our cluster

Enabled on Google Cloud by default

Tiller wants to make changes to our cluster, so it needs to get some permissions set

User Accounts

*Identifies a *person* administering our cluster*

Service Accounts

*Identifies a *pod* administering a cluster*

ClusterRoleBinding

Authorizes an account to do a certain set of actions across the entire cluster

RoleBinding

*Authorizes an account to do a certain set of actions in a *single namespace**

User Accounts

kubectl config view users

Service Accounts

kubectl config view serviceaccounts

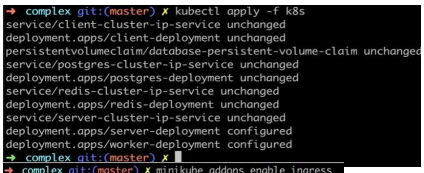
kubectl create serviceaccount --namespace kube-system tiller

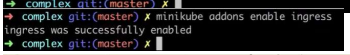
Create a new service account called tiller in the kube-system namespace

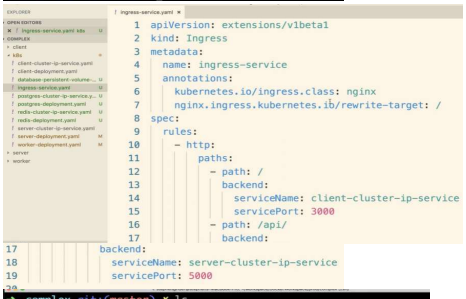
kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller

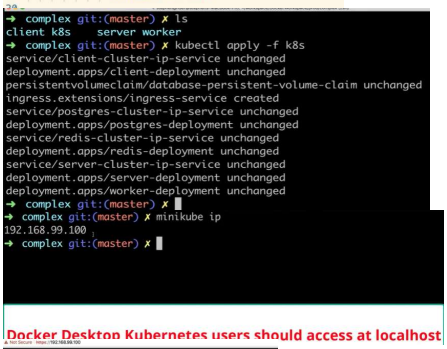
Create a new clusterrolebinding with the role 'cluster-admin' and assign it to service account 'tiller'

- [ ] Check out a branch
- [ ] Make changes
- [ ] Commit changes
- [ ] Push to github branch
- [ ] Create a PR
- [ ] Wait for tests to show up 'green'
- [ ] Merge the PR
- [ ] See changes appear on prod

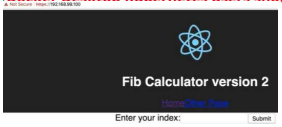








Docker Desktop Kubernetes users should access at localhost

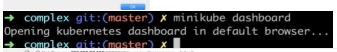


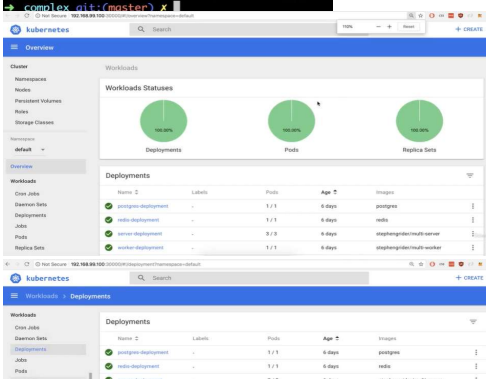
Indexes I have seen:

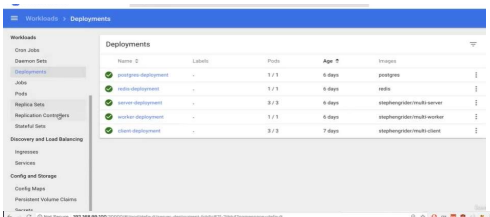
10

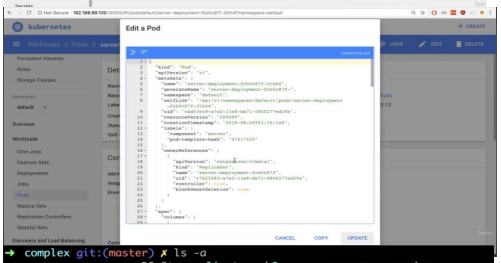
Calculated Values:

For index 10 I calculated 89



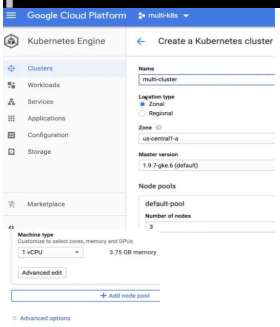


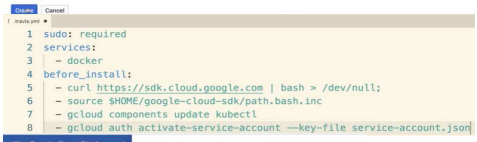


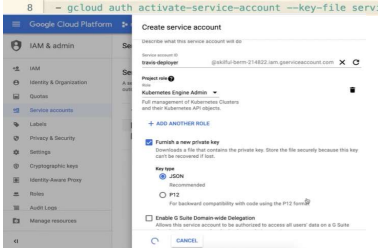


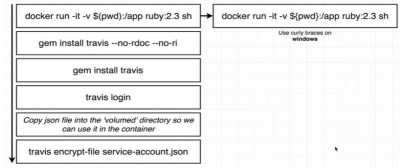

```
➜ complex git:(master) ✗ ls -a
.          ..         .DS_Store  client     k8s        server     worker
➜ complex git:(master) ✗ git init
Initialized empty Git repository in /Users/stephengrider/workspace/DockerWo
rkspace/prod/complex/.git/
➜ complex git:(master) ✗ git add .
➜ complex git:(master) ✗ git commit -m "initial commit"
```

```
➜ complex git:(master) ls -a
.          .DS_Store  client     server
..         .git       k8s        worker
➜ complex git:(master) git remote -v
➜ complex git:(master) git remote remove origin
fatal: No such remote: origin
➜ complex git:(master) git remote add origin git@github.com:StephenGrider/
multi-k8s.git
➜ complex git:(master) git remote -v
origin  git@github.com:StephenGrider/multi-k8s.git (fetch)
origin  git@github.com:StephenGrider/multi-k8s.git (push)
➜ complex git:(master) git push origin master
```










```
➜ complex git:(master) ✗ docker run -it -v $(pwd):/app ruby:2.3 sh
# ls
app  boot  etc   lib    media  opt   root  sbin  sys  usr
bin  dev   home  lib64  mnt    proc  run   srv   tmp  var
# cd app
# ls
client  k8s  server  worker
#
```

```
# gem install travis --no-rdoc --no-ri
Fetching: multipart-post-2.0.0.gem (100%)
Successfully installed multipart-post-2.0.0
```

```
# travis
Shell completion not installed. Would you like to install it now? |y| n
Usage: travis COMMAND ...

Available commands:

        accounts      displays accounts and their subscription status
        branches      displays the most recent build for each branch
```

```
# travis login
We need your GitHub login to identify you.
This information will not be sent to Travis CI, only to api.github.com.
The password will not be displayed.

Try running with --github-token or --auto if you don't want to enter your p
assword anyway.

Username: stephengrider
Password for stephengrider: ********
Two-factor authentication code for stephengrider: 843114
Successfully logged in as StephenGrider!
# ls
client  k8s  server  service-account.json  worker
# pwd
```

```
Username: stephengrider
Password for stephengrider: ********
Two-factor authentication code for stephengrider: 843114
Successfully logged in as StephenGrider!
# ls
client  k8s  server  service-account.json  worker
# pwd
/app
#
```

```
# travis encrypt-file service-account.json -r StephenGrider/multi-k8s
encrypting service-account.json for StephenGrider/multi-k8s
storing result as service-account.json.enc
storing secure env variables for decryption

Please add the following to your build script (before_install stage in your
 .travis.yml, for instance):

    openssl aes-256-cbc -K $encrypted_0c35eebf403c_key -iv $encrypted_0c35e
ebf403c_iv -in service-account.json.enc -out service-account.json -d

Pro Tip: You can add it automatically by running with --add.

Make sure to add service-account.json.enc to the git repository.
Make sure not to add service-account.json to the git repository.
Commit all changes to your .travis.yml.
#
```

```
# ls
client  server              service-account.json.enc
k8s     service-account.json  worker
# ls
client  k8s  server  service-account.json.enc  worker
#
```

```
➜  complex git:(master) ✗ ls
client                  server              worker
k8s                     service-account.json.enc
➜  complex git:(master) ✗ git add .
➜  complex git:(master) ✗ git commit -m "added encrypted service account fi
le"
[master d859186] added encrypted service account file
 2 files changed, 8 insertions(+)
 create mode 100644 .travis.yml
 create mode 100644 service-account.json.enc
➜  complex git:(master)
```

```
 10    - gcloud config set project skilful-berm-214822
 11    - gcloud config set compute/zone us-central1-a
 12    - gcloud container clusters get-credentials multi-cluster
 13    - echo "$DOCKER_PASSSWORD" | docker login -u "$DOCKER_USERNAME" --pas
 14    - docker build -t stephengrider/react-test -f ./client/Dockerfile.dev
 15
 16  script:
 17    - docker run stephengrider/react-test npm test -- --coverage
 18
 19  deploy:
 20    provider: script
 21    script: bash ./deploy.sh
 22    on:
 23      branch: master
```

```
1  docker build -t stephengrider/multi-client -f ./client/Dockerfile ./client
2  docker build -t stephengrider/multi-server -f ./server/Dockerfile ./server
3  docker build -t stephengrider/multi-worker -f ./worker/Dockerfile ./worker
4  docker push stephengrider/multi-client
5  docker push stephengrider/multi-server
6  docker push stephengrider/multi-worker
7  kubectl apply -f k8s
8  kubectl set image deployments/server-deployment server=stephengrider/mu
```

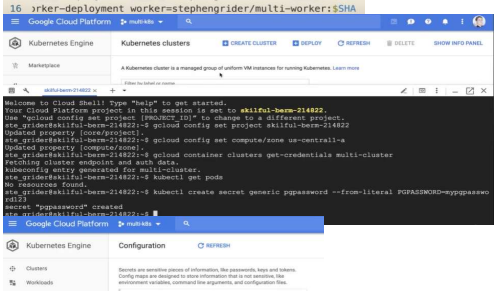
```
➜  complex git:(master) ✗ git rev-parse HEAD
d859186b88466ce00a259183ef0a5f98060ca301
➜  complex git:(master) ✗ git log
➜  complex git:(master) ✗
```

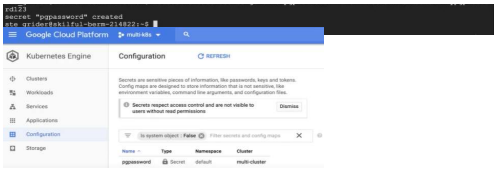
```
 1  sudo: required
 2  services:
 3    - docker
 4  env:
 5    global:
 6      - SHA=$(git rev-parse HEAD)
 7      - CLOUDSDK_CORE_DISABLE_PROMPTS=1
 8  before_install:
 9    - openssl aes-256-cbc -K $encrypted_0c35eebf403c_key -iv
10    - curl https://sdk.cloud.google.com | bash > /dev/null;
11    - source $HOME/google-cloud-sdk/path.bash.inc
12    - gcloud components update kubectl
13    - gcloud auth activate-service-account --key-file servic
14    - gcloud config set project skilful-berm-214822
15    - gcloud config set compute/zone us-central1-a
16    - gcloud container clusters get-credentials multi-cluste
17    - echo "$DOCKER_PASSSWORD" | docker login -u "$DOCKER_US
```

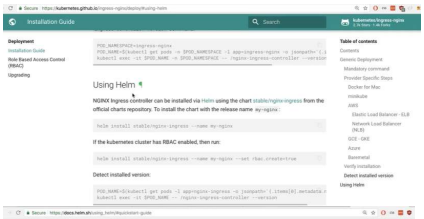
```
1  docker build -t stephengrider/multi-client:latest -t stephengrider/multi-client:$SHA -
2  docker build -t stephengrider/multi-server:latest -t stephengrider/multi-server:$SHA -
3  docker build -t stephengrider/multi-worker:latest -t stephengrider/multi-worker:$SHA -
4  docker push stephengrider/multi-client
```

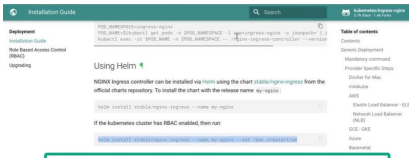
```
 1  docker build -t stephengrider/multi-client:latest -t stephengrider/mult
 2  docker build -t stephengrider/multi-server:latest -t stephengrider/mult
 3  docker build -t stephengrider/multi-worker:latest -t stephengrider/mult
 4
 5  docker push stephengrider/multi-client:latest
 6  docker push stephengrider/multi-server:latest
 7  docker push stephengrider/multi-worker:latest
 8
 9  docker push stephengrider/multi-client:$SHA
10  docker push stephengrider/multi-server:$SHA
11  docker push stephengrider/multi-worker:$SHA
12
13  kubectl apply -f k8s
14  kubectl set image deployments/server-deployment server=stephengrider/mu
15  kubectl set image deployments/client-deployment client=stephengrider/mu
16  kubectl set image deployments/worker-deployment worker=stephengrider/mu
```

```
 4
 5  -client:latest
 6  -server:latest
 7  -worker:latest
 8
 9  -client:$SHA
10  -server:$SHA
11  -worker:$SHA
12
13
14  :rver-deployment server=stephengrider/multi-server:$SHA
15  lient-deployment client=stephengrider/multi-client:$SHA
16  orker-deployment worker=stephengrider/multi-worker:$SHA
```

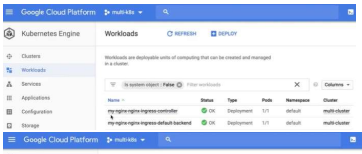


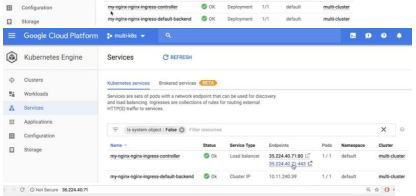






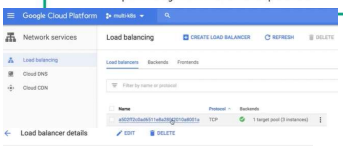
Important - The current install commands for Ingress Nginx with Helm can be found in the lecture note just before this video! Do not run the command that is shown here!



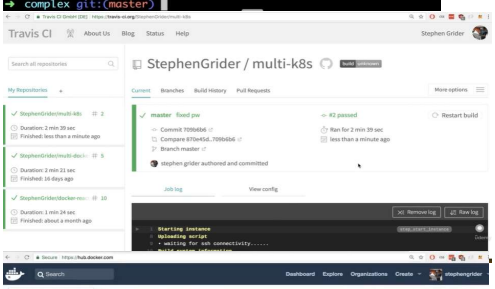


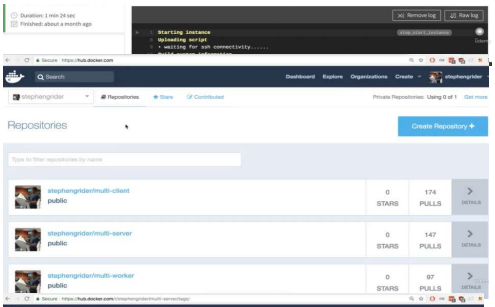

```
default backend - 404
```

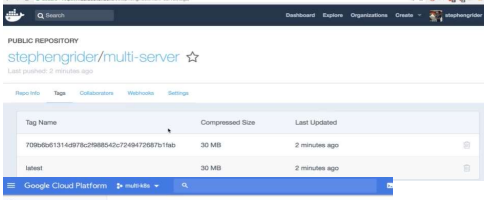
Important - A default backend no longer exists. You will now only see a **404 Not Found Error Page**. This is perfectly normal and expected!

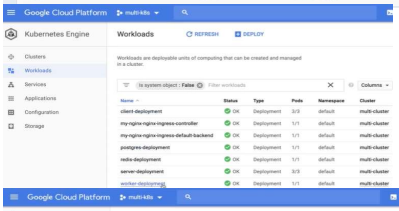


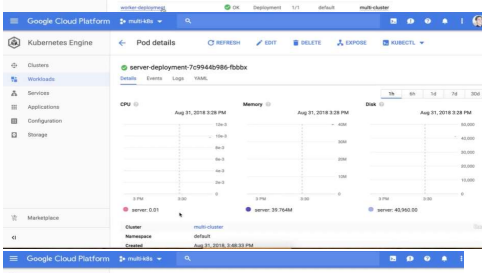

```
➜  complex git:(master) ✗ git add .
➜  complex git:(master) ✗ git commit -m "added deployment scripts"
[master 870e45d] added deployment scripts
 2 files changed, 35 insertions(+)
 create mode 100644 deploy.sh
➜  complex git:(master) git push origin master
Counting objects: 8, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 3.78 KiB | 3.78 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To github.com:StephenGrider/multi-k8s.git
   049231e..870e45d  master -> master
➜  complex git:(master)
```

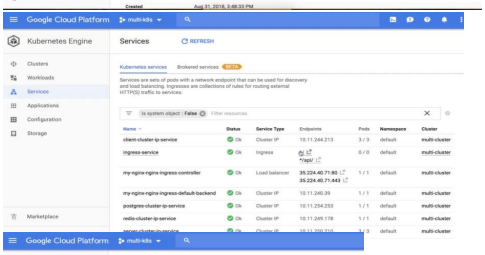


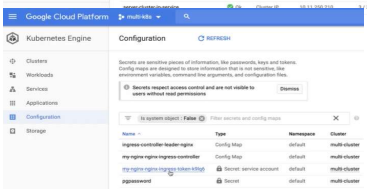


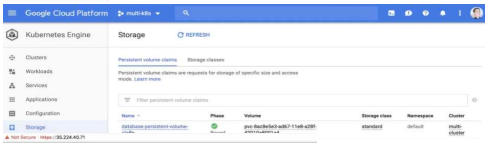


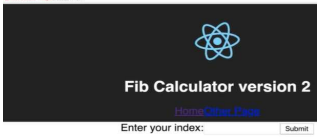




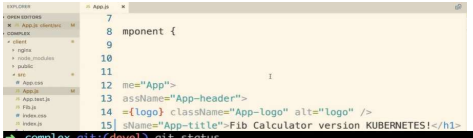






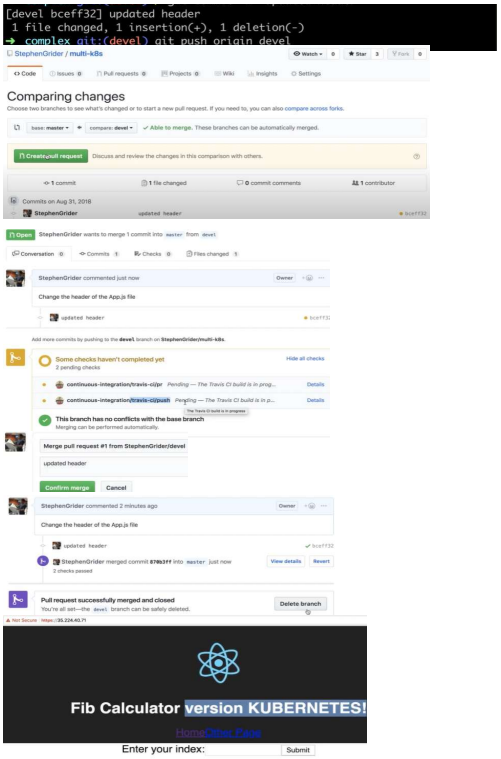


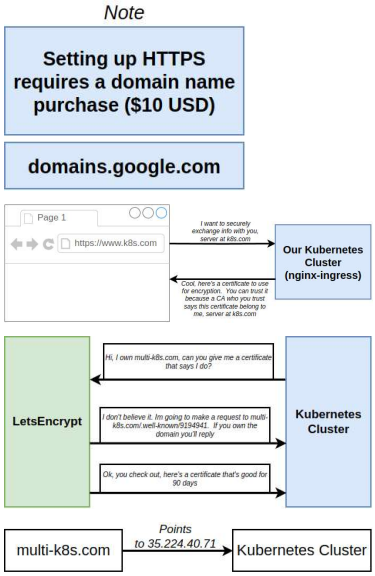

```
➜ complex git:(feature) ✗ git checkout -b devel
Switched to a new branch 'devel'
➜ complex git:(devel)
```

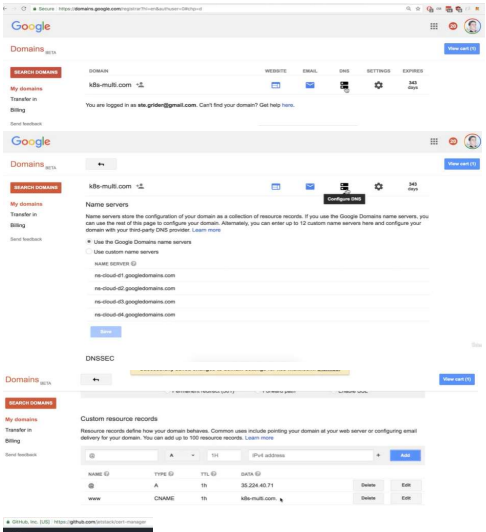


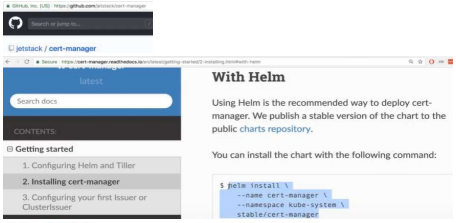

```
➜ complex git:(devel) git status
On branch devel
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   client/src/App.js

no changes added to commit (use "git add" and/or "git commit -a")
➜ complex git:(devel) ✗ git add .
➜ complex git:(devel) ✗ git commit -m "updated header"
[devel bceff32] updated header
 1 file changed, 1 insertion(+), 1 deletion(-)
➜ complex git:(devel) git push origin devel
```

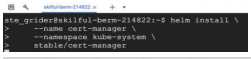
Comparing changes

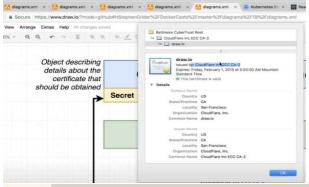













```yaml
apiVersion: certmanager.k8s.io/v1alpha1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: 'ste.grider@gmail.com'
    privateKeySecretRef:
      name: letsencrypt-prod
    http01: {}
```

```yaml
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
  name: k8s-multi-com-tls
spec:
```

```yaml
apiVersion: certmanager.k8s.io/v1alpha1
kind: Certificate
metadata:
  name: k8s-multi-com-tls
spec:
  secretName: k8s-multi-com
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
  commonName: k8s-multi.com
  dnsNames:
    - k8s-multi.com
    - www.k8s-multi.com
```

```yaml
    - www.k8s-multi.com
  acme:
    config:
      - http01:
          ingressClass: nginx
        domains:
          - k8s-multi.com
          - www.k8s-multi.com
```

```
➜  complex git:(devel) git checkout master
Switched to branch 'master'
➜  complex git:(master) ✗ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        k8s/certificate.yaml
        k8s/issuer.yaml

➜  complex git:(master) ✗ git add .
➜  complex git:(master) ✗ git commit -m "added certificate and issuer"
[master 60a5c23] added certificate and issuer
 2 files changed, 31 insertions(+)
 create mode 100644 k8s/certificate.yaml
 create mode 100644 k8s/issuer.yaml
➜  complex git:(master) git push origin master
To github.com:StephenGrider/multi-k8s.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'git@github.com:StephenGrider/multi-k8s
➜  complex git:(master) git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:StephenGrider/multi-k8s
 * branch            master     -> FETCH_HEAD
   709b6b6..870b3ff  master     -> origin/master
Merge made by the 'recursive' strategy.
 client/src/App.js | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
➜  complex git:(master) git push origin master
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (7/7), done.
```

```
ate_grider@skilful-berm-214822:~$ kubectl get certificates
NAME              AGE
k8s-multi-com-tls   2m
ate_grider@skilful-berm-214822:~$ kubectl describe certificates
  Normal  CertIssued  1m  cert-manager  Certificate issued successfully
ate_grider@skilful-berm-214822:~$
```

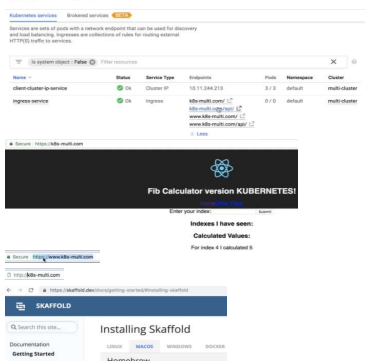
```
ate_grider@skilful-berm-214822:~$ kubectl get secrets
NAME                                  TYPE                                  DATA      AGE
default-token-mp55w                   kubernetes.io/service-account-token   3         2d
k8s-multi-com                         kubernetes.io/tls                     2         3m
my-nginx-nginx-ingress-token-k91q6    kubernetes.io/service-account-token   3         2h
pgpassword                            Opaque                                1         3h
```

```yaml
    kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/rewrite-target: /
    certmanager.k8s.io/cluster-issuer: 'letsencrypt-prod'
    nginx.ingress.kubernetes.io/ssl-redirect: 'true'
spec:
  tls:
    - hosts:
        - k8s-multi.com
        - www.k8s-multi.com
      secretName: k8s-multi-com
```

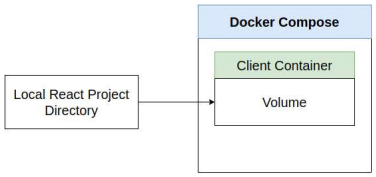
```yaml
        - k8s-multi.com
        - www.k8s-multi.com
      secretName: k8s-multi-com
  rules:
    - host: k8s-multi.com
      http:
        paths:
          - path: /
            backend:
              serviceName: client-cluster-ip-service
              servicePort: 3000
          - path: /api/
            backend:
              serviceName: server-cluster-ip-service
              servicePort: 5000
```

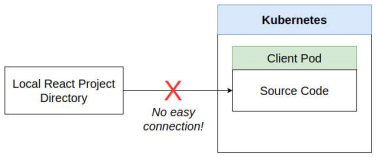
```yaml
          - path: /
            backend:
              serviceName: client-cluster-ip-service
              servicePort: 3000
          - path: /api/
            backend:
              serviceName: server-cluster-ip-service
              servicePort: 5000
    - host: www.k8s-multi.com
      http:
        paths:
          - path: /
            backend:
              serviceName: client-cluster-ip-service
              servicePort: 3000
```

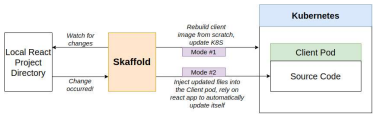
```
➜  complex git:(master) git add .
➜  complex git:(master) ✗ git commit -m "updated ingress"
[master 844b0ad] updated ingress
 1 file changed, 20 insertions(+), 1 deletion(-)
➜  complex git:(master) git push origin master
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 628 bytes | 628.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:StephenGrider/multi-k8s.git
   277f903..844b0ad  master -> master
➜  complex git:(master)
```



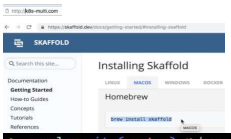
Docker Compose



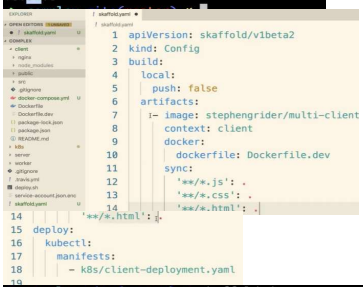




skaffold.dev/docs/getting-started

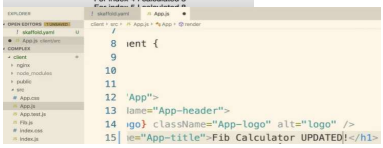



```
➜  complex git:(master) ✗ brew install skaffold
Warning: skaffold 0.32.0 is already installed and up-to-date
To reinstall 0.32.0, run `brew reinstall skaffold`
➜  complex git:(master) ✗ skaffold version
v0.32.0
```

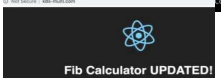


```
apiVersion: skaffold/v1beta2
kind: Config
build:
  local:
    push: false
  artifacts:
    - image: stephengrider/multi-client
      context: client
      docker:
        dockerfile: Dockerfile.dev
      sync:
        '**/*.js': .
        '**/*.css': .
        '**/*.html': .
deploy:
  kubectl:
    manifests:
      - k8s/client-deployment.yaml
```

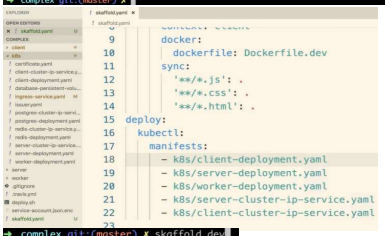
```
➜  complex git:(master) ✗ skaffold dev
WARN[0000] The semantics of sync has changed, the folder structure is no longer flattened but preserved (see https://skaffold.dev/docs/how-tos/filesync/). The likely impacted patterns in your skaffold yaml are: [**/*.js **/*.css **/*.html]
Generating tags...
 - stephengrider/multi-client -> stephengrider/multi-client:5818...-dirty
Tags generated in 68.845656ms
Starting build...
Found [docker-for-desktop] context, using local docker daemon.
Building [stephengrider/multi-client]...
```



```
Watching for changes every 1s...
[client-deployment-5c776dd854-cshz8 client] Compiled successfully!
[client-deployment-5c776dd854-gszqn client] Compiled successfully!
```

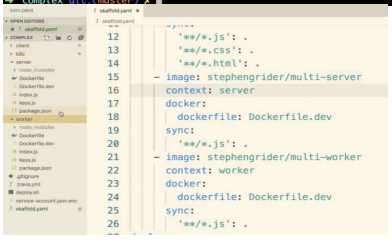



```
[client-deployment-5c776dd854-s6wsq client] Compiled successfully!
^Cpruning images...
WARN[0250] builder cleanup: pruning images: Error response from daemon: conflict: unable to delete 56eb85ba560e (cannot be forced) - image is being used by running container d79ec77122c9
Deploying k8s...
deployment.apps "client-deployment" deleted
Cleanup complete in 235.036398ms
➜  complex git:(master) ✗ kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
deployment-pgadmin-6cdcb6f58c-gxxl6   0/1     CrashLoopBackOff   18         97d
deployment-postgres-6f9bcd6857-kprd7   1/1     Running            1          110d
postgres-deployment-8d79c44d9-gjcxp   1/1     Running            0          1h
redis-deployment-666bf96bdd-zszbm     1/1     Running            0          1h
server-deployment-fcb5c87f-2sbhf      1/1     Running            0          44m
server-deployment-fcb5c87f-5bf2b      1/1     Running            0          44m
server-deployment-fcb5c87f-h68lw      1/1     Running            0          44m
worker-deployment-56b886cd5c-8pqbb    1/1     Running            0          1h
➜  complex git:(master) ✗
```

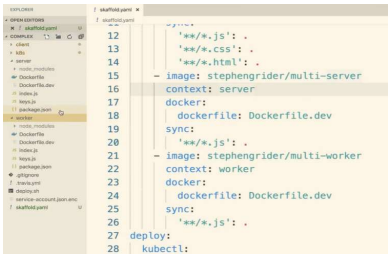


```
      docker:
        dockerfile: Dockerfile.dev
      sync:
        '**/*.js': .
        '**/*.css': .
        '**/*.html': .
deploy:
  kubectl:
    manifests:
      - k8s/client-deployment.yaml
      - k8s/server-deployment.yaml
      - k8s/worker-deployment.yaml
      - k8s/server-cluster-ip-service.yaml
      - k8s/client-cluster-ip-service.yaml
```

```
➜  complex git:(master) ✗ skaffold dev
^Cpruning images...
WARN[0022] builder cleanup: pruning images: Error response from daemon: conflict: unable to delete 16a600a55 (cannot be forced) - image is being used by running container 7db1c7567d7...
Cleaning up...
service "client-cluster-ip-service" deleted
deployment.apps "client-deployment" deleted
service "server-cluster-ip-service" deleted
deployment.apps "server-deployment" deleted
deployment.apps "worker-deployment" deleted
Cleanup complete in 622.023504ms
➜  complex git:(master) ✗ kubectl get pods
NAME                                READY   STATUS        RESTARTS   AGE
deployment-pgadmin-6cdcb6f58c-gxxl6   0/1     CrashLoopBackOff   18         97d
deployment-postgres-6f9bcd6857-kprd7   1/1     Running       1          110d
postgres-deployment-8d79c44d9-gjcxp   1/1     Running       0          1h
redis-deployment-666bf96bdd-zszbm     1/1     Running       0          1h
server-deployment-5b6d4d9f64-4xblz    1/1     Terminating   0          11s
server-deployment-5b6d4d9f64-llw6h    1/1     Terminating   0          17s
server-deployment-5b6d4d9f64-vh8bb    1/1     Terminating   0          25s
worker-deployment-6d7d75d8dd-bxv8j    1/1     Terminating   0          25s
➜  complex git:(master) ✗
```



```
      docker:
        dockerfile: Dockerfile.dev
      sync:
        '**/*.js': .
        '**/*.css': .
        '**/*.html': .
    - image: stephengrider/multi-server
      context: server
      docker:
        dockerfile: Dockerfile.dev
      sync:
        '**/*.js': .
    - image: stephengrider/multi-worker
      context: worker
      docker:
        dockerfile: Dockerfile.dev
      sync:
        '**/*.js': .
```


skaffold.yaml
12 '**/*.js': .
13 '**/*.css': .
14 '**/*.html': .
15 - image: stephengrider/multi-server
16 context: server
17 docker:
18 dockerfile: Dockerfile.dev
19 sync:
20 '**/*.js': .
21 - image: stephengrider/multi-worker
22 context: worker
23 docker:
24 dockerfile: Dockerfile.dev
25 sync:
26 '**/*.js': .
27 deploy:
28 kubectl: