

Docker and Kubernetes – The Complete Guide

By

Stephen Grider

Couse Link

<https://www.udemy.com/course/docker-and-kubernetes-the-complete-guide/>

Notes By

Naveen Kumar M

<https://www.linkedin.com/in/naveen-kumar-m-5a6960119/>

Table of Contents

Dive Into Docker !	10
Why use Docker?	10
Installing Software	10
Example installing redis using docker	11
Summary	12
What is Docker?	12
What is Image?	12
What is container?	13
Docker for Mac/Windows	13
Docker for Mac/Windows Architecture	13
Installing Docker for Mac	14
Steps for Installing Docker	14
Verifying Installation	14
Installing Docker with WSL2 on Windows 10 Home and Pro	14
Steps and Explanations	15
Using the Docker Client	16
Running docker run hello-world command	16
What's happing in the background?	16
What is Container?	17
How Operating System Works?	17
Imagine a scenario	17
Name spacing and Control Groups	19
Container in getting a picture	19
How is the Image eventually creating a Container?	20
Container Creation Process	20
How is Docker running on your computer?	21
Linux Virtual Machine	21
Checking OS Architecture	21
Manipulating Containers with the Docker Client	22
Docker Run in Detail	22
Creating and Running a Container from an Image	22
What is happening behind the scenes?	22
Overriding Default Commands	22
What happens behind the scenes?	23
Listing Running Containers	24
Container Lifecycle	25
Restarting Stopped Containers	26
Listing All Container	26
Starting the Container without Logging	26
Starting the Container with Logging	26
Removing the stopped container	26
docker system prune	26
Retrieving Log Outputs	27
Command Explanation	27
Command Execution	27
Stopping the container	28
Docker stop	28
Docker kill	29
Some Facts	29
Demo	29
Multi-Command Containers	30

Running a Redis using Docker	31
Executing Command in Running Container	32
Docker exec Command	32
Purpose of the “-it” flag	32
How processes run in Linux environment	33
Command “-it” flag	33
Getting a Command Prompt in Container	34
Command Processors	34
Enable Command Prompt	34
Starting with a Shell	35
Container Isolation	35
Demo and Explanation	35
Building Custom Images Through Docker Server	37
Creating Docker Images	37
Introducing Docker File	37
Docker File Creation Flow	38
Building a Docker Image	38
Creating a Docker File	38
Docker Build	38
Running an Image	39
Dockerfile Teardown	39
What is Base Image?	40
Analogy	40
Base Image	40
The Build Process in Detail	41
Brief Recap	43
Recap	43
Rebuild with Cache	44
Tagging an Image	46
Explanations	46
Executions	46
Manual Image Generation with Docker Commit	47
Explanations	47
Demo	47
Making Real Projects with Docker	48
Creating a Sample Project	48
Project Outline	48
Project Plan	48
Node Server Setup	48
Building a Docker Image for Our Project	49
npm Setup	49
Step for Building Docker Image for Our Node Application	49
Creating a Docker File	50
Building a Docker File	50
Exploring Public Images from Docker Hub	51
Example of Defining Version Along with Image	52
Updating the Dockerfile	53
Building Dockerfile Again	53
Copying Build Files	56
Explanations	56
Updating the docker file	56
Building the Docker Image	57
Building Docker Image with Tagging an Image Name	57

Running a Docker Image	57
Container Port Mapping	58
Port Mapping Command	58
Running the Command	58
Specifying the Working Directory	59
Opening the Shell Inside Container	59
Specifying the Working Directory in Dockerfile	59
Unnecessary Rebuilds	60
What happens we change the project files?	60
Controlling unnecessary rebuild	60
Docker Compose with Multiple Local Container	62
App Overview	62
Overview	62
Architecture	62
Problem with single container approach	63
Sperate container for redis	63
App Server Started Code	64
Code	64
Assembling a Docker File	65
Creating a docker file	65
Building an image	65
Introducing a Docker Compose	65
Running a docker image	65
Starting the redis server	66
Analyzing the issue	66
Options for establishing the network infrastructure	67
Docker Compose	67
Docker Compose File	68
What is YAML File	68
Creating a YAML File	68
Networking with Docker Compose	69
Establishing connection	69
Docker Compose Commands	70
Commands	70
Execution	71
Stopping docker compose containers	71
Typical docker commands to run and stop the containers.	72
Stopping the containers	72
Container Maintenance with Docker Compose	73
Making the server to crash for learning purpose	73
Building the image and starting the container	73
Application crashes when access the link	74
Automatic Container Restart	74
Status codes	74
Restart Policies	75
Updating the docker-compose.yaml	75
Running the container	75
Container status with Docker	76
Starting the container	76
Container status	76
Creating a Production Grade Workflow	78
Workflow	78
Development Workflow	78

Flow Specifics	78
Project Creation	79
Creating react app	79
Necessary Commands	79
Creating a Dev Dockerfile	81
Type of dockerfile	81
Creating a dockerfile	81
Building dockerfile	81
Starting the Container	82
Command	82
Docker Volume	83
Current Scenario	83
Syntax for setting up the volume	84
Running the commands	84
Shorthand with Docker Compose	86
Creating docker compose yaml file	86
Running the docker compose file	87
Overriding the Dockerfile Selection	87
Updating the docker compose yaml file	87
Running the image using docker-compose command	88
Do we need a copy?	88
Executing Tests	89
Live updating tests	89
Docker Compose for Running the Tests	90
Updating the docker file	90
Running the docker compose	90
Modifying the test file	91
Shortcomings on Testing	92
Why are test commands not accepted while container is running?	92
Docker attach	92
Need for Ngix	93
Npm run build	93
Development Environment	93
Production Environment	94
Multi-Step Docker Build	94
Multi Step Docker File	95
Implementing Multi Step Builds	95
Running Nginx	96
Building the Image	96
Running the Image	97
Continuous Integration and Deployment with AWS	99
Services Overview	99
Continuous Integration with Github Actions	99
Creating a Github Actions configuration file	99
Checking the build	99
AWS Elastic Beanstalk	100
What is it?	100
Creating an application in elastic beanstalk	100
Github Actions Configurations for Deployment	102
Creating a user for access in AWS IAM Service	102
Setting the secrets for repository	104
Updating the Github Actions deploy.yml file	104
Commit and Push	104

Exposing ports through docker file	106
Workflow with Github	106
Adding a new feature	106
New git branch	107
Compare and pull request	107
Merging the pull request	109
Terminating AWS Elastic beanstalk Environment	110
<i>Building Multi Container Application</i>	111
Single Container Deployment Issue	111
Fibonacci Sequence Generator	111
Application Overview	111
Application Architecture	112
Application Development	113
Worker	113
Server	115
Client	117
Dockerizing the Multi Services	119
Dockerizing the React App	119
Dockerizing the generic node app	120
Adding a postgres as a service	121
Docker compose config	122
Environment variable with docker compose	122
The worker and client services	123
Nginx Path Routing	124
Building a custom Nginx image	126
Starting up docker compose	126
<i>A Continuous Integration Workflow for Multiple Images</i>	127
Production Multi Container Deployment	127
Production Dockerfile	127
Continuous Integration with Github Actions	130
Successful Image Building	130
<i>Multi Container Deployment to AWS</i>	132
Multi container Definition Files	132
Forming Container Links	134
Using Amazon Linux 2 Platform	136
Create environment using the new platform	136
Rename the current docker-compose file	136
Create a production only docker-compose.yml file	137
Creating the EB Environment	139
Steps	139
Screenshots	139
<i>Architecture in Production Environment</i>	141
Overview of AWS VPC's and Security Group	142
VPC	142
Security group	143
RDS Database Creation	144
Steps	144
Screenshots	145
ElastiCache Redis Creation	148
Steps	148
Screenshots	148
Creating a Custom Security Group	151
Steps	151

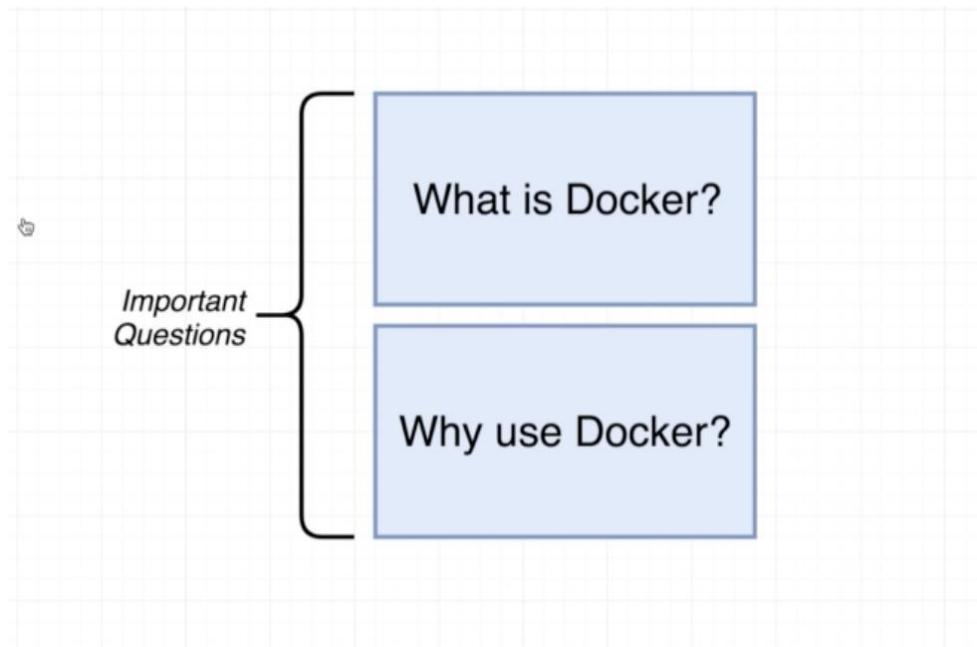
Screenshots	152
Applying Security Groups to ElastiCache	154
Steps	154
Applying Security Groups to RDS	156
Steps	156
Screenshots	157
Applying Security Groups to Elastic Beanstalk	159
Add AWS configuration details to Github Actions deploy.yml file's deploy script	162
Setting Environment Variables	163
Steps	163
Screenshots	164
IAM Keys for Deployment	166
Steps	166
Screenshots	167
AWS Keys in Github Actions	170
Deploying the App	170
Onwards to Kubernetes	172
The Why's and What's of Kubernetes	172
Why Kubernetes?	172
Scaling Challenge	173
Kubernetes cluster	174
Getting Started with Kubernetes	174
Kubernetes in Development and Production	174
Minikube Setup	175
Mapping Existing Knowledge	177
Adding Configuration File	178
Configuration file to create a container	178
Configuration file to set up networking	180
Running a container in Pod	180
Service Config File in Depth	181
Connecting to Running Container	183
The Entire Deployment Flow	185
Imperative and Declarative Deployments	187
Maintaining Set of Containers with Deployments	190
Updating Existing Object	190
Approach	190
Changing the config file	191
Applying the config file	191
Inspecting the specific pod	191
Limitation in config update	192
Deployment Object	193
Running containers with deployment	193
Deployment Config File	195
Applying Deployment	196
Use of service object	197
Scaling and changing the deployments	197
Scaling up pods	198
Updating Deployment Image	199
Rebuilding the Client Image	201
Imperatively updating a deployment image	204
Configure VM to Use Docker Server	205
A Multi Container App with Kubernetes	206

The Path to Production	206
Cluster IP vs Node Port	207
Kubernetes multi container application	207
Deploying the kubernetes configuration	208
The Need for Volume with Database	213
Kubernetes Volume	215
Volume vs Persistent Volume	217
Persistent Volume vs Persistent Volume Claim	217
Claim Config Files	220
Defining Environment Variable	223
Environment Variables	223
Updating the config file	224
Creating an encoded secret	225
Handling the traffic with Ingress Controller	227
Load Balancer Services	227
Quick notes in Ingress	228
Behind the Scenes of Ingress	229
Setting up Ingress Locally with Minikube	232
Ingress Config File	233
Kubernetes Production Deployment	235
Deployment Steps	235
Google Cloud	236
Creating a project in Google Cloud	236
Creating Kubernetes Engine and starting the cluster in GCP	237
Kubernetes Dashboard on Google Cloud	241
Github Actions	251
Steps for Creating a Service Account in GCP	253
Unique Tag for Build Image	257
Configuring the GCloud CLI on Cloud Console	260
Helm V3 Update	262
Kubernetes Security with RBAC	263
Assigning Tiller a Service Account	264
Ingress-Ngnix with Helm	265
The Result of Ingress Nginx	265
Verifying Deployment	267
A Workflow for Changing in Pods	269
HTTPS Setup with Kubernetes	273
Overview	273
HTTP Set Overview	273
Domain Purchase	273
Domain Name Setup	274
Cert Manager Install	276
How to write up Cert Manager	277
Certificate Config File	278
Deployment	278
Ingress Config for HTTPS	279
Google Cloud Cleanup	282
Local Environment Cleanup	283
Local Development with Skaffold	285
Local Development	285
Installing Skaffold	286
Skaffold Config File	286

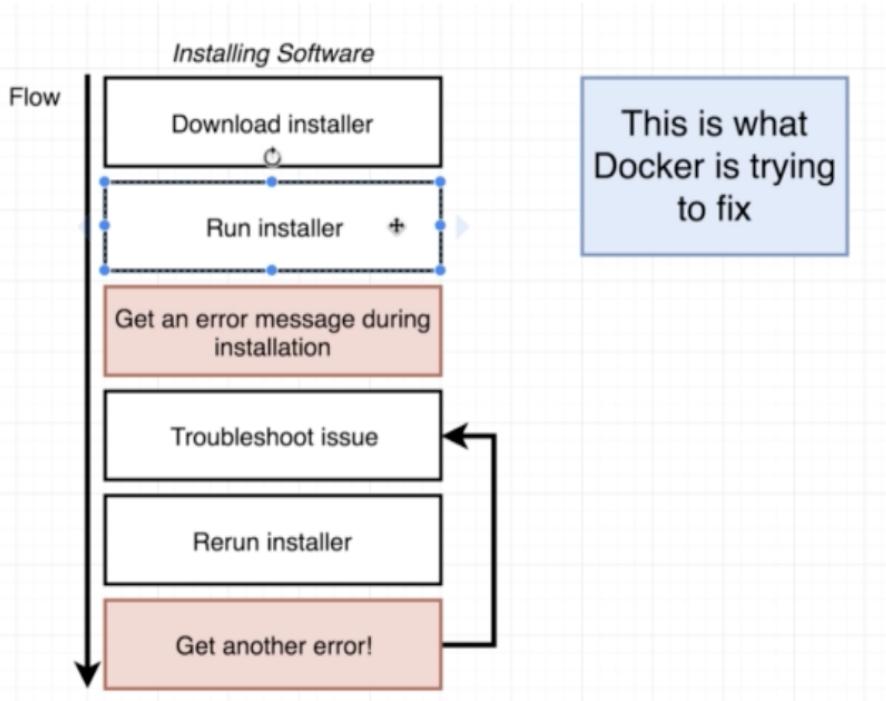
Running the Skaffold	288
Live Sync Changes	288

Dive Into Docker !

Why use Docker?



Installing Software



Docker wants to make it easy and straightforward for you to install and run software on computer (Web servers and cloud-based computing platform).

Example installing redis using docker

Redis is in-memory data store.

Below is the installation step.

Installation

Download, extract and compile Redis with:

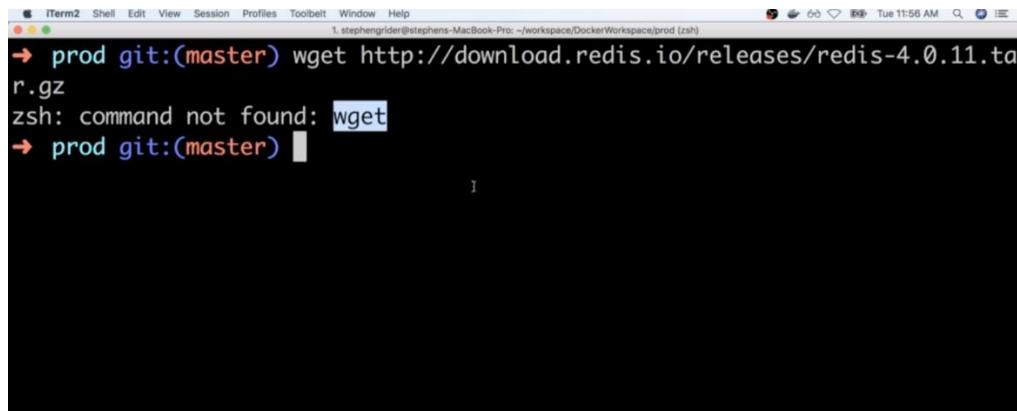
```
$ wget http://download.redis.io/releases/redis-4.0.11.tar.gz  
$ tar xzf redis-4.0.11.tar.gz  
$ cd redis-4.0.11  
$ make
```

The binaries that are now compiled are available in the `src` directory. Run Redis with:

```
$ src/redis-server
```

You can interact with Redis using the built-in client:

```
$ src/redis-cli  
redis> set foo bar  
OK  
redis> get foo  
"bar"
```



```
prod git:(master) wget http://download.redis.io/releases/redis-4.0.11.tar.gz  
zsh: command not found: wget
```

Got an error while downloading the package. We need to fix this before moving forward. This is the problem docker wants to solve.



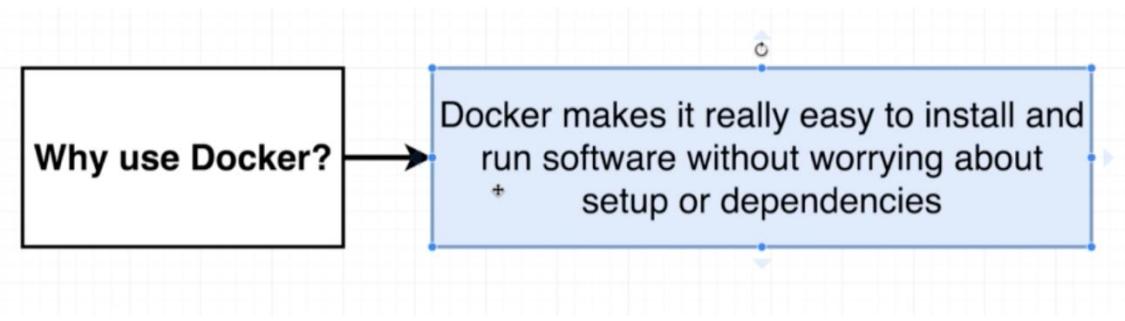
```
prod git:(master) docker run -it redis
```

```
is.conf
Redis 4.0.11 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 1

http://redis.io
```

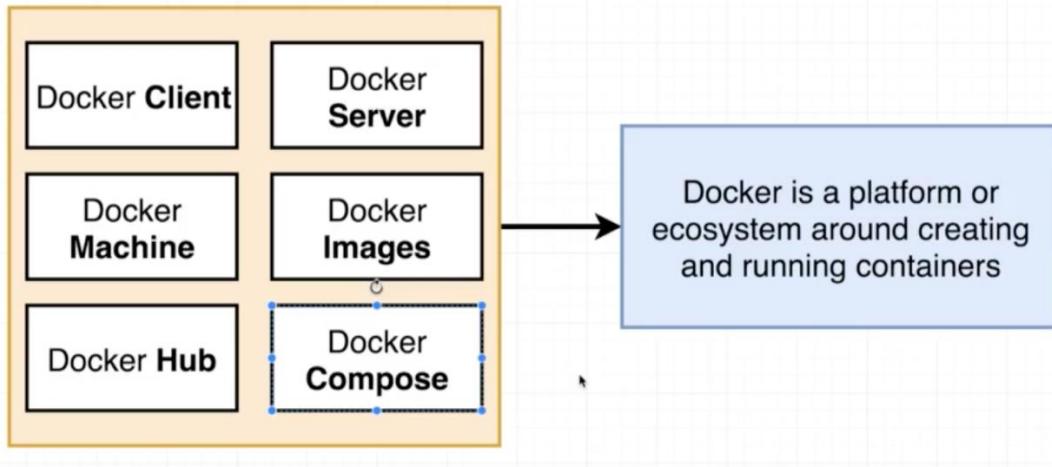
But using a docker, application is installed and up and running, with a single command.

Summary



What is Docker?

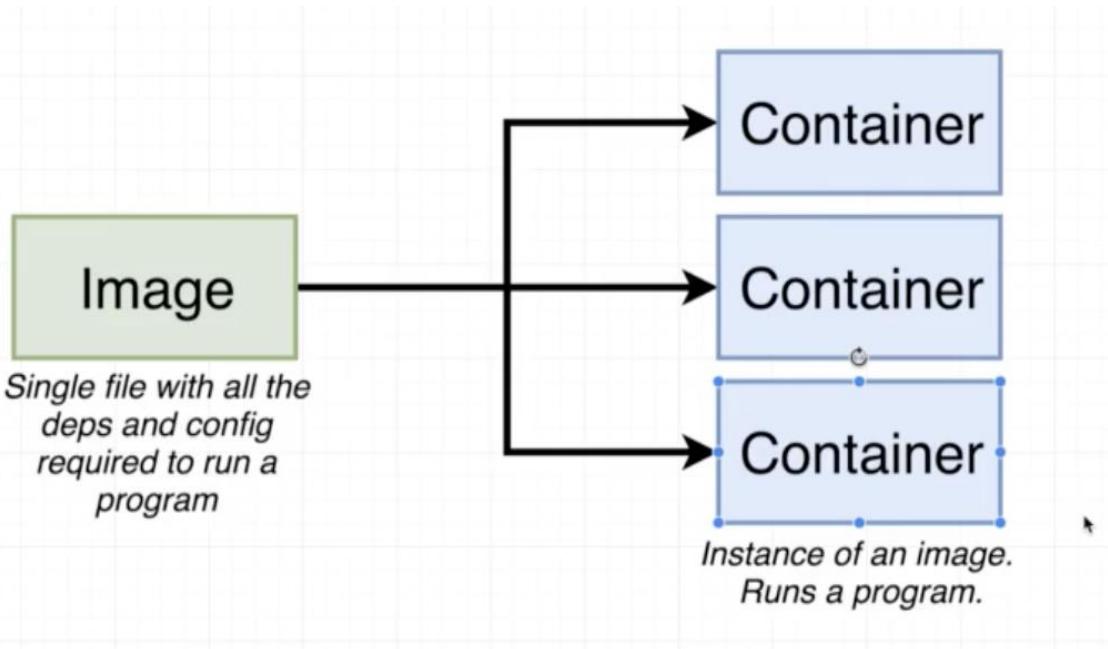
Docker Ecosystem



What is Image?

When running the docker command, docker cli reaches to docker hub and download the single file called image.

Here image refers to,



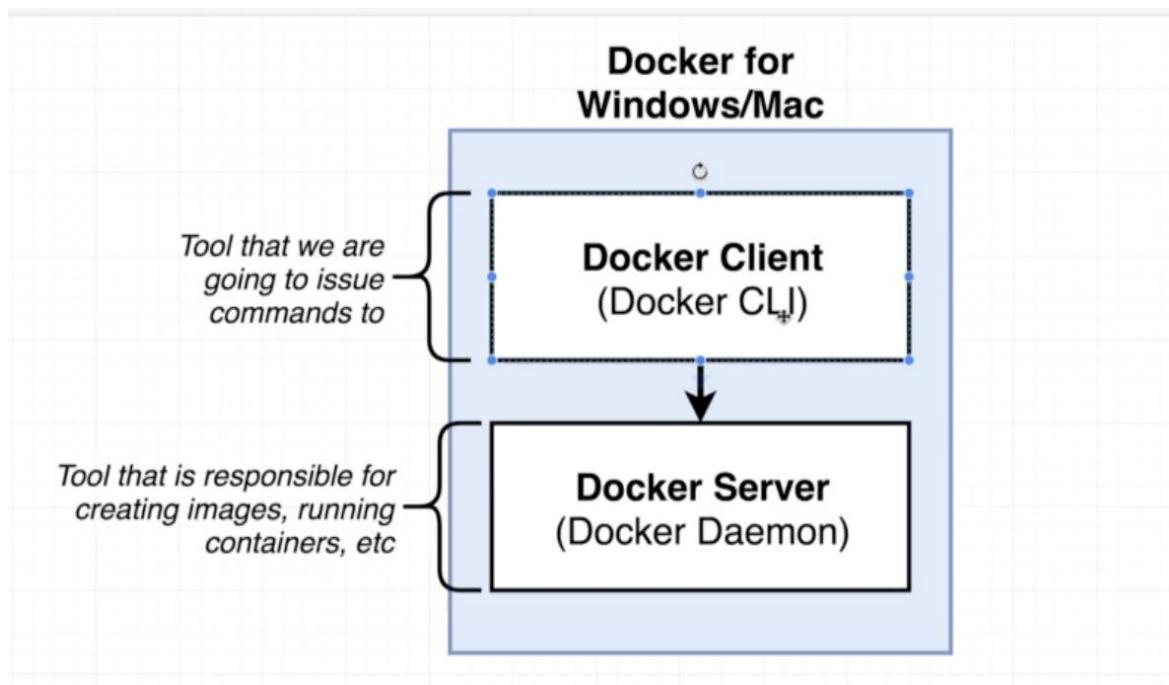
What is container?

We can use an image to create a container.

Container is running programs and has its own isolated set of hardware resources (memory, networking technology and hard drive).

Docker for Mac/Windows

Docker for Mac/Windows Architecture

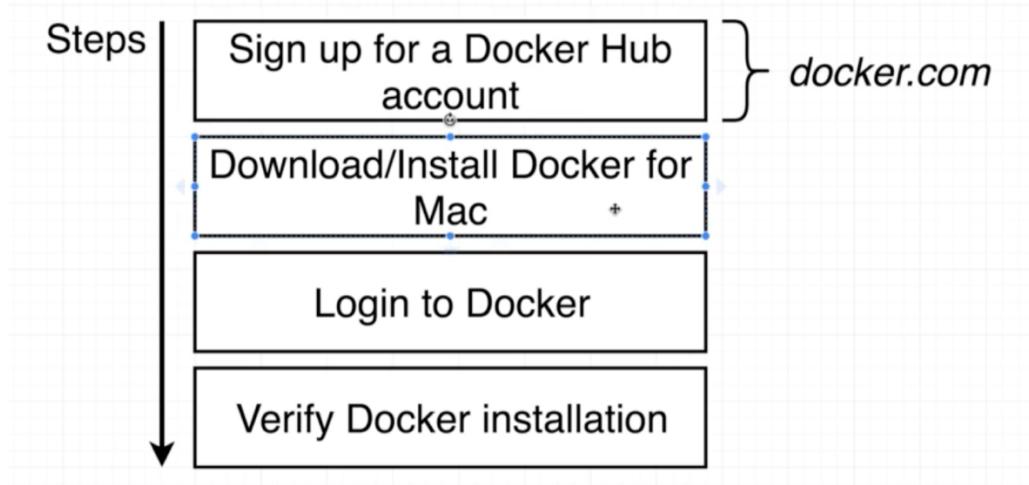


Docker client used to interact with Docker Server.

Docker Server is running behind the scenes.

Installing Docker for Mac

Steps for Installing Docker



Verifying Installation

```
→ ~ docker version
Client:
Cloud integration: 1.0.17
Version: 20.10.8
API version: 1.41
Go version: go1.16.6
Git commit: 3967b7d
Built: Fri Jul 30 19:55:20 2021
OS/Arch: darwin/amd64
Context: default
Experimental: true

Server: Docker Engine - Community
Engine:
Version: 20.10.8
API version: 1.41 (minimum version 1.12)
Go version: go1.16.6
Git commit: 75249d8
Built: Fri Jul 30 19:52:10 2021
OS/Arch: linux/amd64
Experimental: false
containerd:
Version: 1.4.9
GitCommit: e25210fe30a0a703442421b0f60afac609f950a3
runc:
Version: 1.0.1
GitCommit: v1.0.1-0-g4144b63
docker-init:
Version: 0.19.0
GitCommit: de40ad0
→ ~
```

Installing Docker with WSL2 on Windows 10 Home and Pro

Installing Docker with WSL2 on Windows 10 Home and Pro

updated 2-19-2021

WSL2

Windows 10 Pro and some Windows 10 Home users will be able to install Docker Desktop if their computer supports the Windows Subsystem for Linux (WSL2).

Please see this guide first which includes all of the steps to install and enable WSL2:

<https://docs.microsoft.com/en-us/windows/wsl/install-win10>

Once you have successfully installed and enabled WSL2 and then installed the Linux OS of your choice, continue to the Docker Desktop installation docs here:

<https://docs.docker.com/docker-for-windows/wsl/>

With Docker Desktop installed using WSL2 as a backend, you will now be able to follow along with the course. You will access the applications at localhost just like we do in the video lectures.

Important - A significant difference when using WSL2 is that you will need to create and run your project files from within the Linux filesystem, not the Windows filesystem. This will be very important in later lectures when we cover volumes.

You can access your Linux system by running `wsl` in the Windows Search Bar. Going forward, all Docker commands should be run within WSL2 and not on the Windows file system.

Using the Docker Client

Running docker run hello-world command

```
→ ~ docker run hello-world
docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
See 'docker run --help'.
→ ~ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:97a379f4ff88575512824f3b352bc03cd75e239179ee0fecc38e597b2209f49a
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 the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

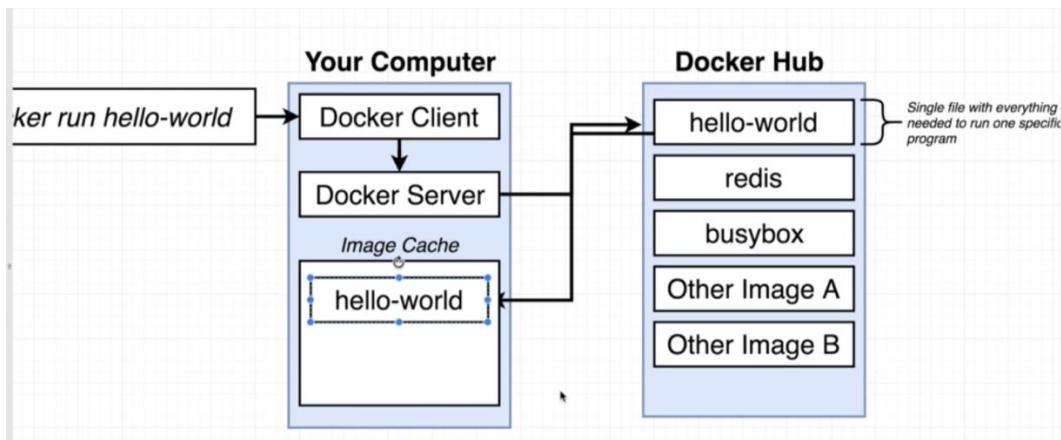
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

What's happening in the background?

1. “docker run hello-world” – docker command for running “hello-world” image.
2. When we run “docker run hello-world” command in terminal, it starts up the **Docker Client**.
3. **Docker Client** then communicates with **Docker Server** (also known as Docker daemon).
4. Docker server checks in **local image cache** that “hello-world” image already exists or not.
5. If the local image does not exist, docker server will communicate to docker hub to fetch the image.
6. Once received image from **docker hub**, it will be stored in image cache.
7. Next, **container** is created from the image and runs the program.
8. So, we are seeing the “Hello from Docker!” message in terminal as result program run.



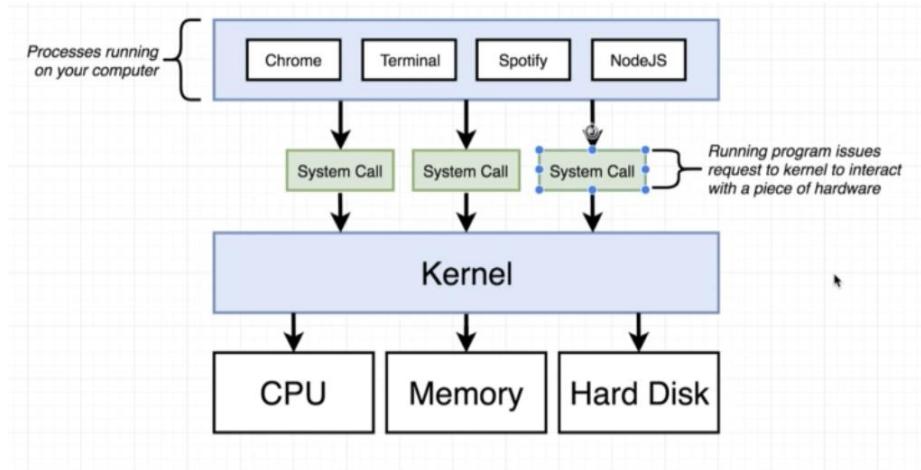
What is Container?

To understand the container, we need to understand how operating system works.

How Operating System Works?

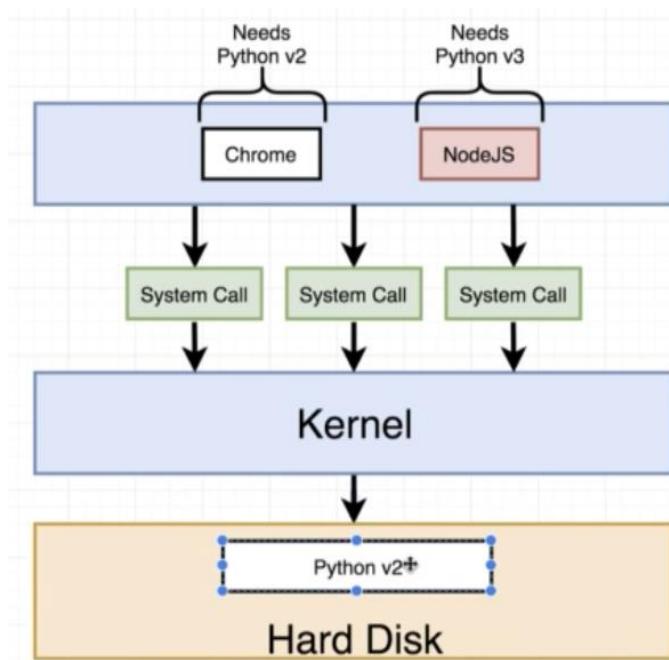
Kernel

- Running software process.
- Governs access between the programs running on computer and hardware connected to the computer.



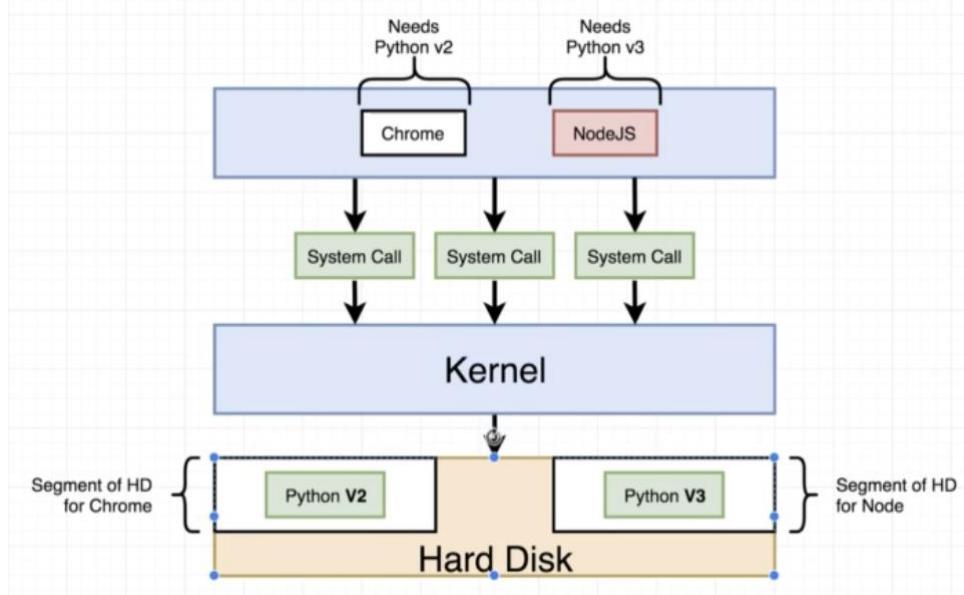
Imagine a scenario

- Two program is running on a computer one is chrome and other is NodeJS.
- Chrome require Python v2 and NodeJS require a Python V3 to be installed in the computer.
- But our computer is installed only with Python v2.

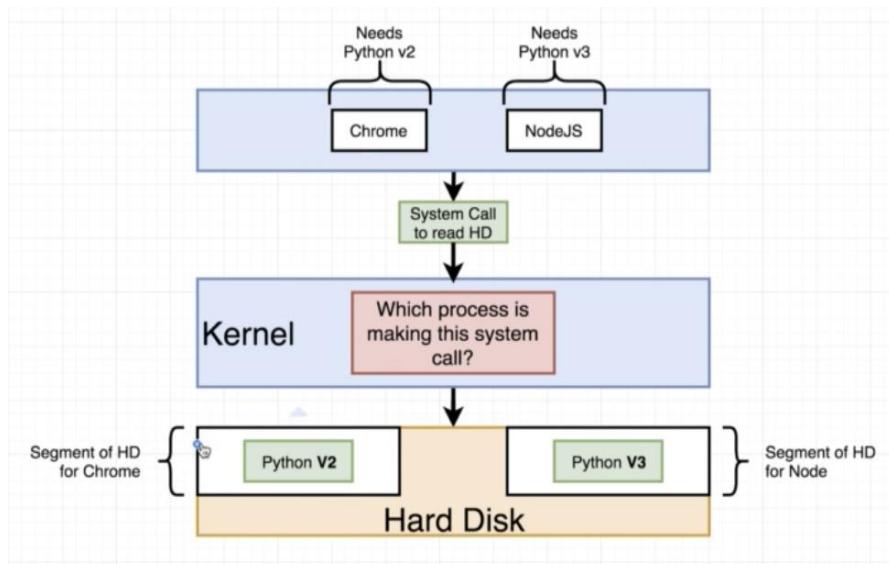


How can we solve this issue?

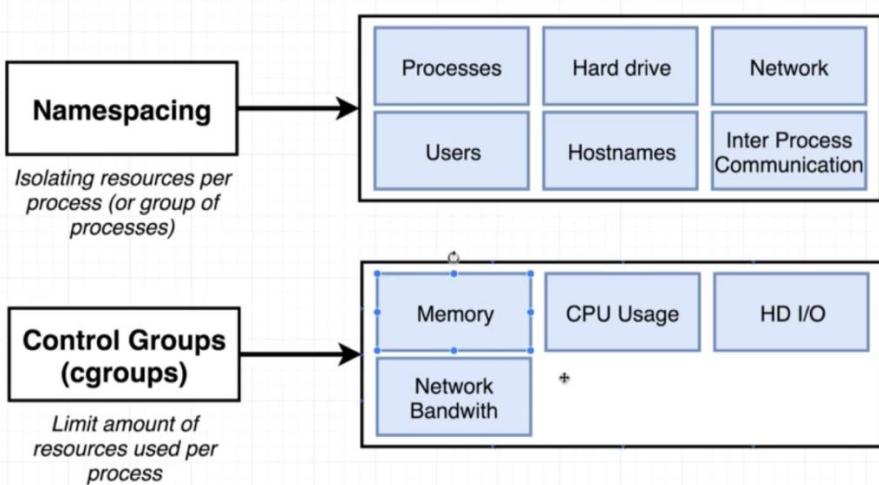
we can make use of the OS feature called name spacing.



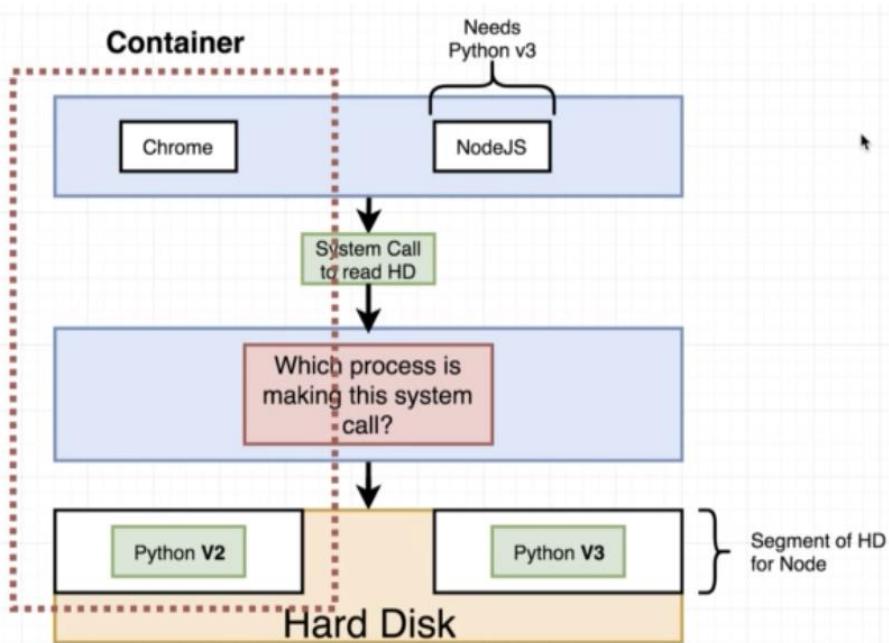
Based on the system call, kernel can able to assign right segment to the program.



Name spacing and Control Groups

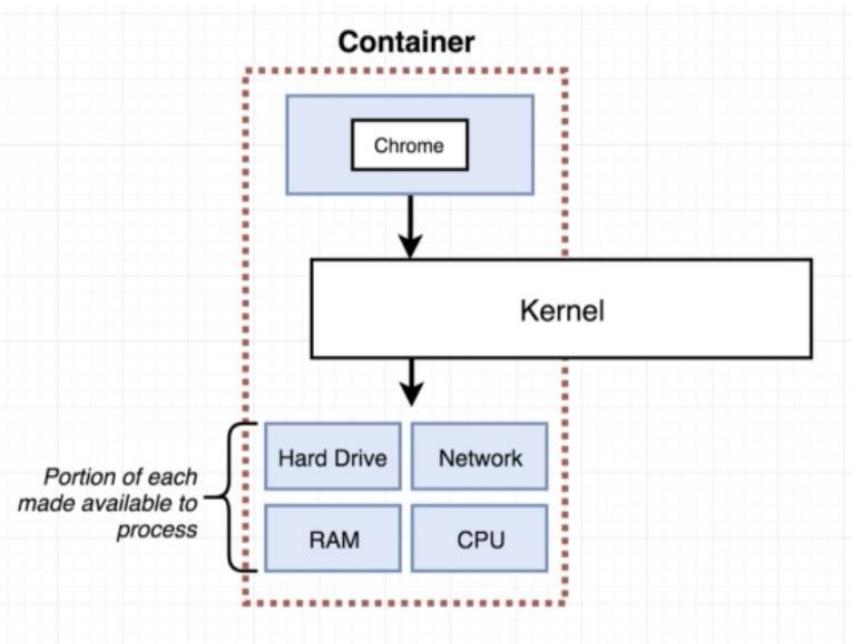


Container in getting a picture



Here, Container represent program with separate allocated resource.

Process or set of processes that have a grouping of resources specifically assigned to it.



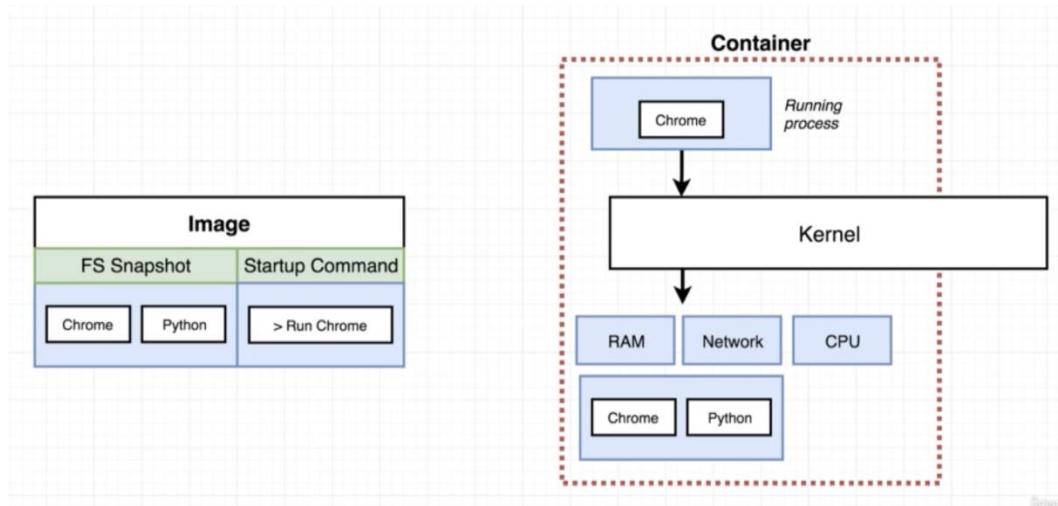
How is the Image eventually creating a Container?

Image contains,

- File System Snapshot - A copy case of directory and files. (Installable directories)
- Startup command - For running the program.

Container Creation Process

- Initially kernel assign an isolated set of resource specific to the container.
- File Snapshot from the images are placed inside the allocated segment of the hard drive.
- Then, startup command is executed and that creates instance of the process like example in Chrome, Python.

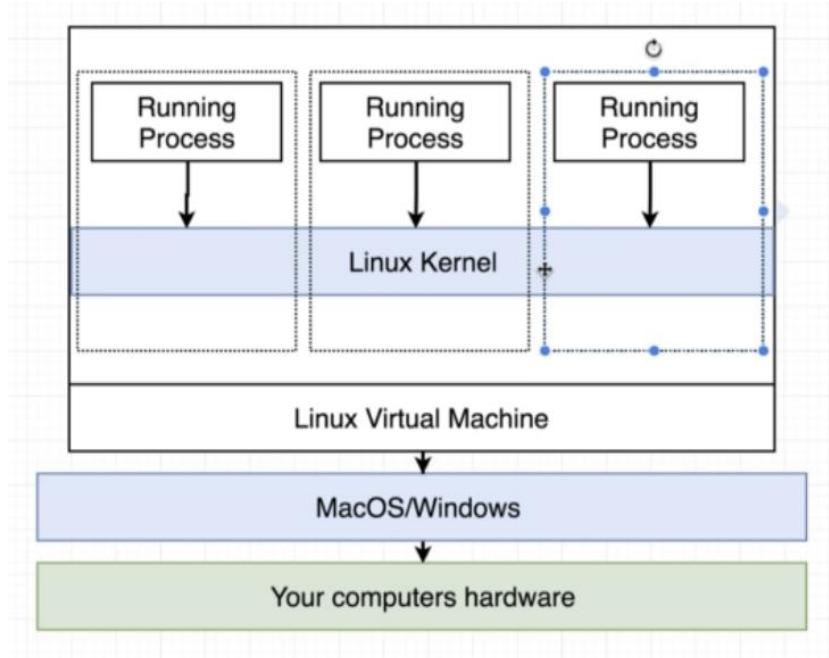


How is Docker running on your computer?

Linux Virtual Machine

Namespacing and Control Groups are specific to Linux operating system.

When we installed docker, we installed Linux virtual machine.



Checking OS Architecture

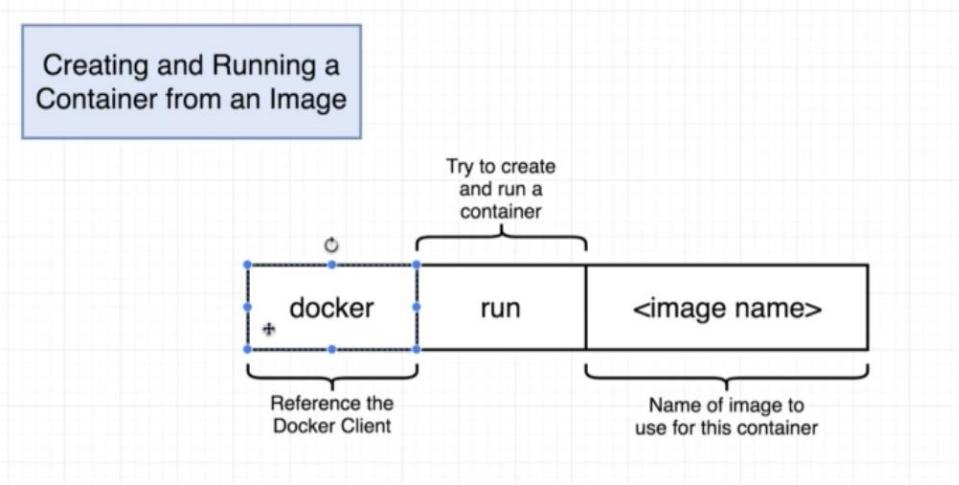
```
→ ~ docker version
Client:
Cloud integration: v1.0.22
Version: 20.10.12
API version: 1.41
Go version: go1.16.12
Git commit: e91ed57
Built: Mon Dec 13 11:46:56 2021
OS/Arch: darwin/amd64
Context: default
Experimental: true

Server: Docker Desktop 4.5.0 (74594)
Engine:
  Version: 20.10.12
  API version: 1.41 (minimum version 1.12)
  Go version: go1.16.12
  Git commit: 459d0df
  Built: Mon Dec 13 11:43:56 2021
  OS/Arch: linux/amd64
  Experimental: false
containerd:
  Version: 1.4.12
  GitCommit: 7b11cfaabd73bb80907dd23182b9347b4245eb5d
runc:
  Version: 1.0.2
  GitCommit: v1.0.2-0-g52b36a2
docker-init:
  Version: 0.19.0
  GitCommit: de40ad0
```

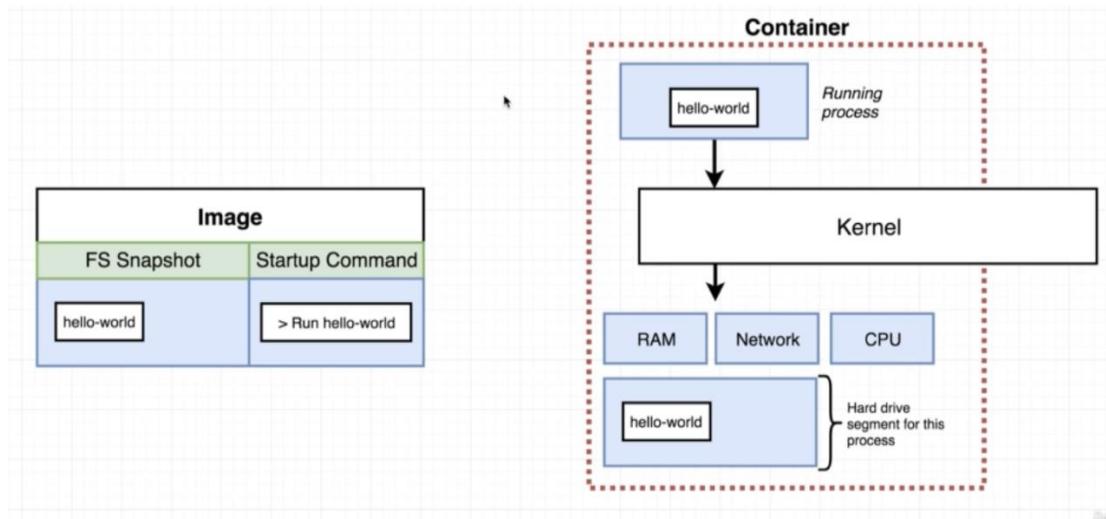
Manipulating Containers with the Docker Client

Docker Run in Detail

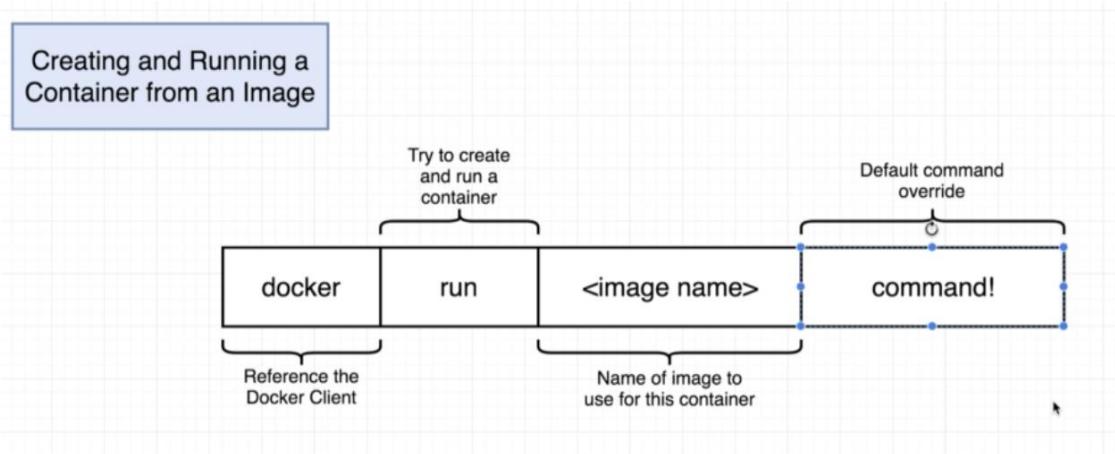
Creating and Running a Container from an Image



What is happening behind the scenes?



Overriding Default Commands

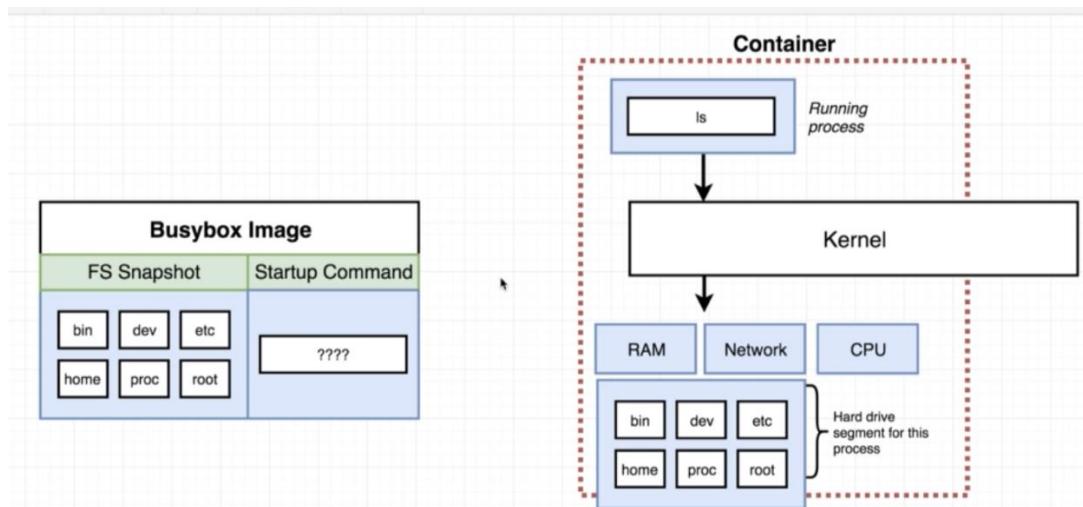


Alternate command to be executed after the container starts up. This alternate command overrides the existing start up command inside the image.

```
→ 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) docker run busybox ls
bin
dev
etc
home
proc
root
sys
tmp
usr
var
```

What happens behind the scenes?



Busybox image has a FS Snapshot is mentioned above. That is the reason why we are getting a list of same file system.

Command "ls" and "echo" executables are available in FS Snapshot. That's the reason why we are to see the output of those command.

If we don't have a FS for a given command, we will be getting an error like below.

```
→ prod git:(master) docker run hello-world ls
docker: Error response from daemon: OCI runtime create failed: container_li
nux.go:348: starting container process caused "exec: \"ls\": executable fil
e not found in $PATH": unknown.
→ prod git:(master) docker run hello-world echo hi there
docker: Error response from daemon: OCI runtime create failed: container_li
nux.go:348: starting container process caused "exec: \"echo\": executable f
ile not found in $PATH": unknown.
→ prod git:(master)
```

Listing Running Containers

docker ps

Shows running container.

```
→ prod git:(master) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STAT
US                  PORTS              NAMES
→ prod git:(master)
```

```
round-trip min/avg/max = 45.922/73.735/292.132 ms
→ ~ docker run busybox ping google.com
PING google.com (216.58.200.142): 56 data bytes
64 bytes from 216.58.200.142: seq=0 ttl=37 time=56.886 ms
64 bytes from 216.58.200.142: seq=1 ttl=37 time=57.832 ms
64 bytes from 216.58.200.142: seq=2 ttl=37 time=48.046 ms
64 bytes from 216.58.200.142: seq=3 ttl=37 time=54.966 ms
64 bytes from 216.58.200.142: seq=4 ttl=37 time=56.016 ms
64 bytes from 216.58.200.142: seq=5 ttl=37 time=46.098 ms
64 bytes from 216.58.200.142: seq=6 ttl=37 time=62.768 ms
64 bytes from 216.58.200.142: seq=7 ttl=37 time=63.596 ms
64 bytes from 216.58.200.142: seq=8 ttl=37 time=41.999 ms
```

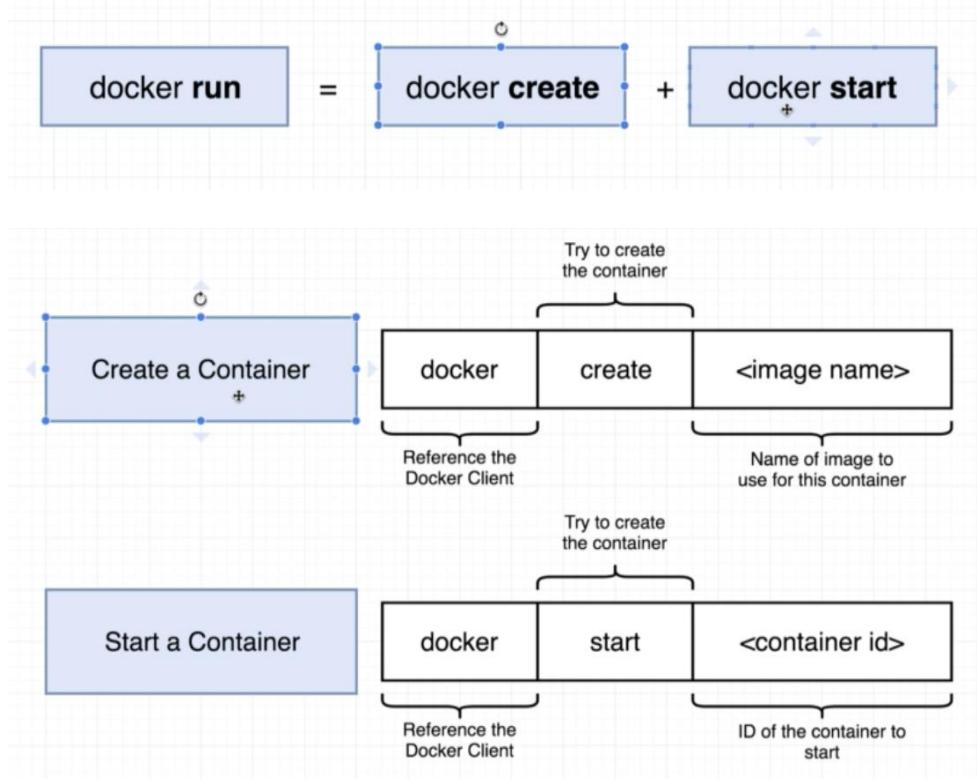
```
/Users/9/Downloads/Ramda.js/2311/c.001: No matches found.  Total:
→ ~ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS            PORTS      NAMES
d31ad06d36c8        busybox            "ping google.com"   7 seconds ago     Up 7 seconds
→ ~
```

docker ps --all

Shows all the container ever created.

```
→ ~ docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS            PORTS      NAMES
f72a680a4002        busybox            "ping google.com"   3 minutes ago    Up 3 minutes
d31ad06d36c8        busybox            "ping google.com"   4 minutes ago   Exited (0) 3 minutes ago
85a2afdf89457       hello-world        "/hello"          3 hours ago     Exited (0) 3 hours ago
→ ~
```

Container Lifecycle



A screenshot of an iTerm2 terminal window showing the following commands and output:

```
prod git:(master) docker create hello-world
9271d26374ff54f715ce137ab71b5d86a7e9af3638b51d0a470fd8765c55821e
prod git:(master) docker start -a 9271d26374ff54f715ce137ab71b5d86a7e9af3638b51d0a470fd8765c55821e
```

Output:
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 the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
\$ docker run -it ubuntu bash

-a => Watch for the output from the container and print it to the terminal.

A screenshot of an iTerm2 terminal window showing the following commands and output:

```
prod git:(master) docker create hello-world
955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734ec95358f7ff4fec
prod git:(master) docker start 955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734ec95358f7ff4fec
955c4d7671e4bdd68dea6d8a2deedac4a182a9d46a568e734ec95358f7ff4fec
prod git:(master) █
```

Restarting Stopped Containers

Listing All Container

```
→ ~ docker ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f72a680a4002 busybox "ping google.com" 3 minutes ago Up 3 minutes
d31ad06d36c8 busybox "ping google.com" 4 minutes ago Exited (0) 3 minutes ago
85a2af89457 hello-world "/hello" 3 hours ago Exited (0) 3 hours ago
→ ~
```

We can start the container which status is Exited (0).

Starting the Container without Logging

```
→ ~ docker ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f72a680a4002 busybox "ping google.com" 3 minutes ago Up 3 minutes
d31ad06d36c8 busybox "ping google.com" 4 minutes ago Exited (0) 3 minutes ago
85a2af89457 hello-world "/hello" 3 hours ago Exited (0) 3 hours ago
→ ~ docker start d31ad06d36c8
d31ad06d36c8
→ ~ docker ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f72a680a4002 busybox "ping google.com" 13 hours ago Exited (0) 13 hours ago
d31ad06d36c8 busybox "ping google.com" 13 hours ago Up 11 seconds
85a2af89457 hello-world "/hello" 16 hours ago Exited (0) 16 hours ago
→ ~
```

Starting the Container with Logging

```
→ ~ docker start -a d31ad06d36c8
64 bytes from 142.250.77.174: seq=159 ttl=37 time=64.694 ms
64 bytes from 142.250.77.174: seq=160 ttl=37 time=60.017 ms
64 bytes from 142.250.77.174: seq=161 ttl=37 time=60.130 ms
64 bytes from 142.250.77.174: seq=162 ttl=37 time=71.112 ms
64 bytes from 142.250.77.174: seq=163 ttl=37 time=58.364 ms
64 bytes from 142.250.77.174: seq=164 ttl=37 time=55.679 ms
64 bytes from 142.250.77.174: seq=165 ttl=37 time=64.265 ms
```

Once the container created, we cannot override the default command.

```
→ prod git:(master) docker start -a 37a8c45a23e1 echo bye there
you cannot start and attach multiple containers at once
→ prod git:(master) |
```

Removing the stopped container

docker system prune

“docker system prune” command

- Remove all containers.
- Remove all image build cache.

```

→ ~ docker ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f72a680a4002 busybox "ping google.com" 13 hours ago Exited (0) 13 hours ago
d31ad06d36c8 busybox "ping google.com" 13 hours ago Exited (0) 6 seconds ago
85a2af89457 hello-world "/hello" 16 hours ago Exited (0) 16 hours ago

→ ~ docker system prune
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

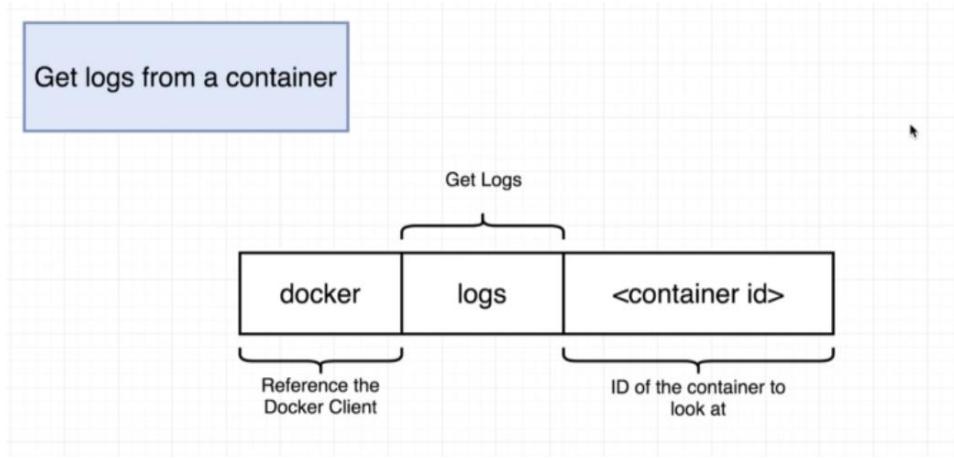
Are you sure you want to continue? [y/N] y
Deleted Containers:
f72a680a4002eb74e690a14270e59a430567a04400b6000c74fe3827d5c2ddd0
d31ad06d36c84074fc4aa5761472299dd05bd40d0cf2a7df70b713cc91a7cf
85a2af8945757a558cc82a3d3b7bc01c7ba7004e60bbdc5b203cc713ed8b1be

Total reclaimed space: 0B
→ ~ docker ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
→ ~

```

Retrieving Log Outputs

Command Explanation



Retrieve the logs emitted from the container.

Command Execution

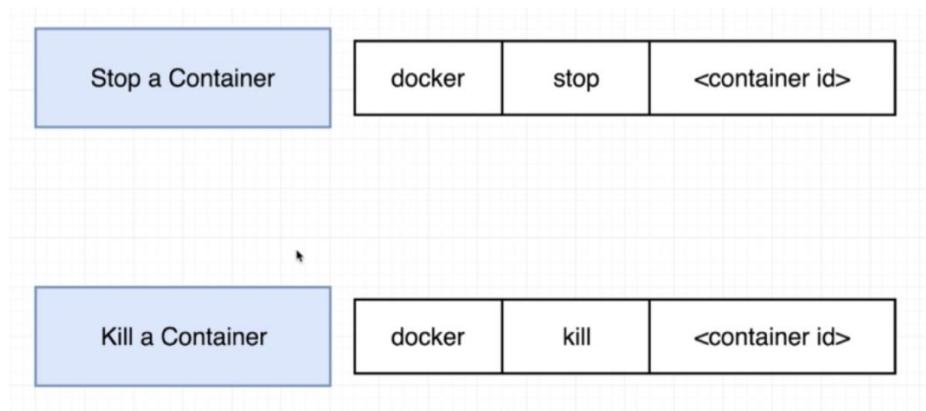
```

→ ~ docker create busybox echo hi there
0e30728dda9aedbb545e0d67c9b386385048b139d996b1ce94b777c9667b6445
→ ~ docker start 0e30728dda9aedbb545e0d67c9b386385048b139d996b1ce94b777c9667b6445
0e30728dda9aedbb545e0d67c9b386385048b139d996b1ce94b777c9667b6445
→ ~ docker log 0e30728dda9aedbb545e0d67c9b386385048b139d996b1ce94b777c9667b6445
docker: 'log' is not a docker command.
See 'docker --help'
→ ~ docker logs 0e30728dda9aedbb545e0d67c9b386385048b139d996b1ce94b777c9667b6445
hi there
→ ~

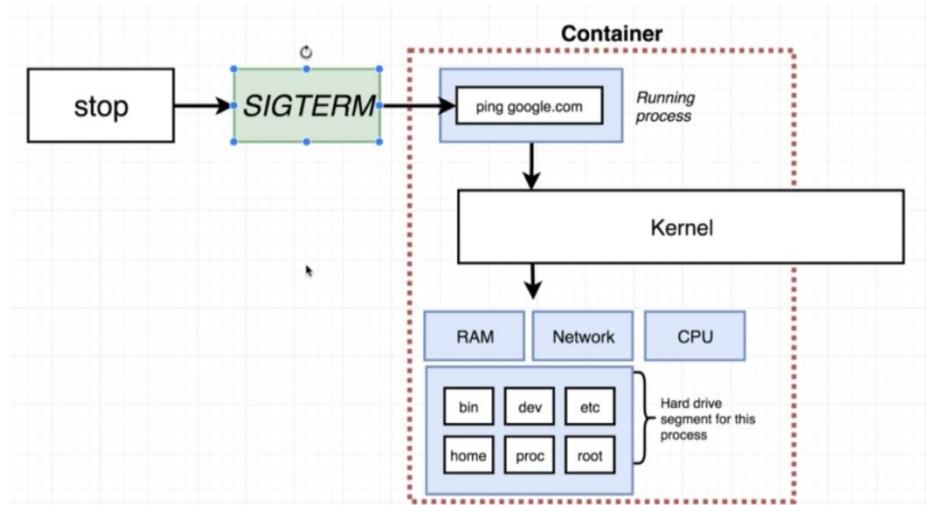
```

Stopping the container

We have two options

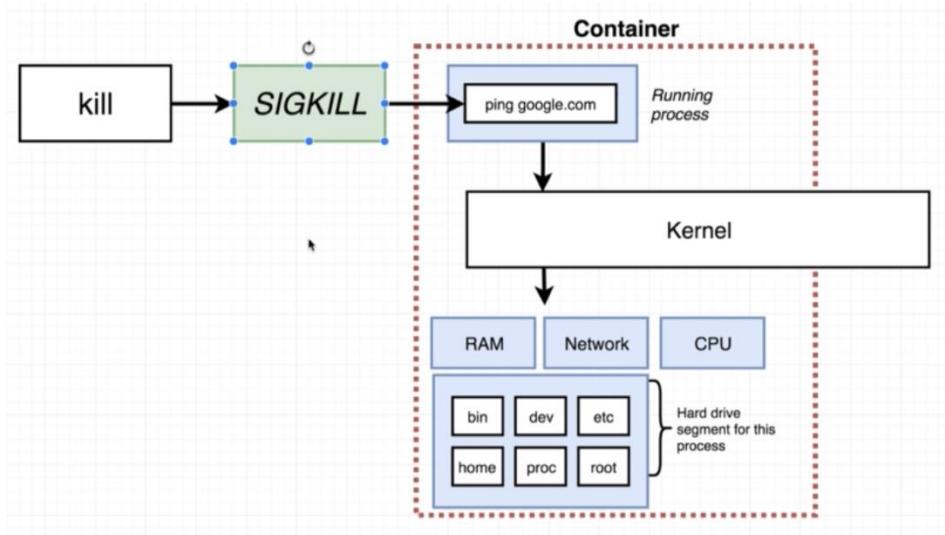


Docker stop



- Docker stop command send “SIGTERM” message to primary process in the container.
- SIGTERM – provides the process a time to shutdown process and do the necessary clean up activity.

Docker kill



- Docker kill command issue a SIGKILL message.
- It tells container process to shut down immediately.
- No relaxing time given for clean-up and so on.

Some Facts

Ideally, we will be stopping the container using stop command. If the container is locked up and not responding to the docker stop command, then we will be issuing the docker kill.

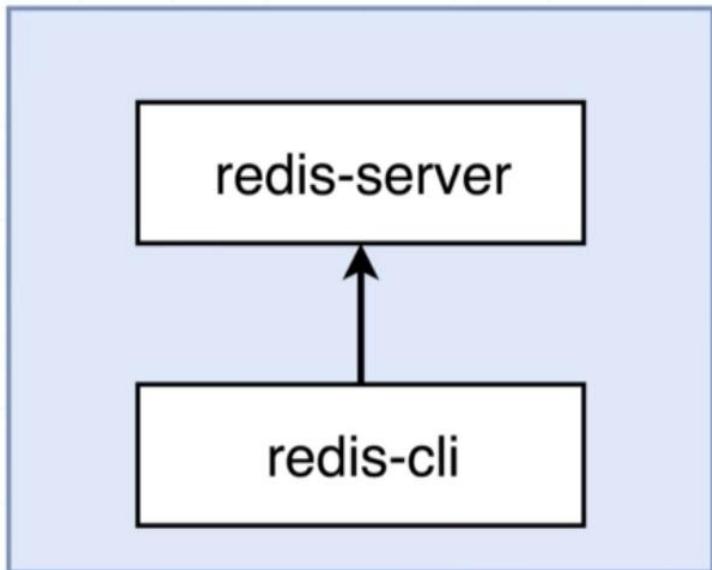
After issuing a docker stop command, if the docker did not stops docker automatically in 10 seconds, docker fallback to the docker kill command and executing the same.

Demo

```
→ ~ docker create busybox ping google.com
da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
→ ~ docker start da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
→ ~ docker logs da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
PING google.com (142.250.77.174): 56 data bytes
64 bytes from 142.250.77.174: seq=0 ttl=37 time=51.610 ms
64 bytes from 142.250.77.174: seq=1 ttl=37 time=51.901 ms
64 bytes from 142.250.77.174: seq=2 ttl=37 time=52.139 ms
64 bytes from 142.250.77.174: seq=3 ttl=37 time=72.741 ms
64 bytes from 142.250.77.174: seq=4 ttl=37 time=73.879 ms
64 bytes from 142.250.77.174: seq=5 ttl=37 time=53.557 ms
64 bytes from 142.250.77.174: seq=6 ttl=37 time=64.142 ms
64 bytes from 142.250.77.174: seq=7 ttl=37 time=61.993 ms
64 bytes from 142.250.77.174: seq=8 ttl=37 time=50.674 ms
64 bytes from 142.250.77.174: seq=9 ttl=37 time=60.838 ms
→ ~ docker stop da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
```

```
→ ~ docker start da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
→ ~ docker kill da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
da5c8008d8cef73ca5c263dd054e4e45dab5ce67377490d8c56d10ea85473adc
→ ~ █
```

My Computer



Redis is an in-memory data store.

A screenshot of an iTerm2 window titled "1. redis-server (redis-server)". The window displays the following text:

```
Running in standalone mode
Port: 6379
PID: 11868

http://redis.io

11868:M 07 Aug 15:58:52.414 # Server initialized
11868:M 07 Aug 15:58:52.414 * DB loaded from disk: 0.000 seconds
11868:M 07 Aug 15:58:52.414 * Ready to accept connections
```

A screenshot of an iTerm2 window showing a Redis client session. The session starts with the user's last login information and then proceeds with Redis commands:

```
Last login: Tue Aug  7 15:54:17 on ttys002
→ prod git:(master) ✘ redis-cli
127.0.0.1:6379> set mynumber 5
OK
127.0.0.1:6379> get mynumber
"5"
127.0.0.1:6379>
```

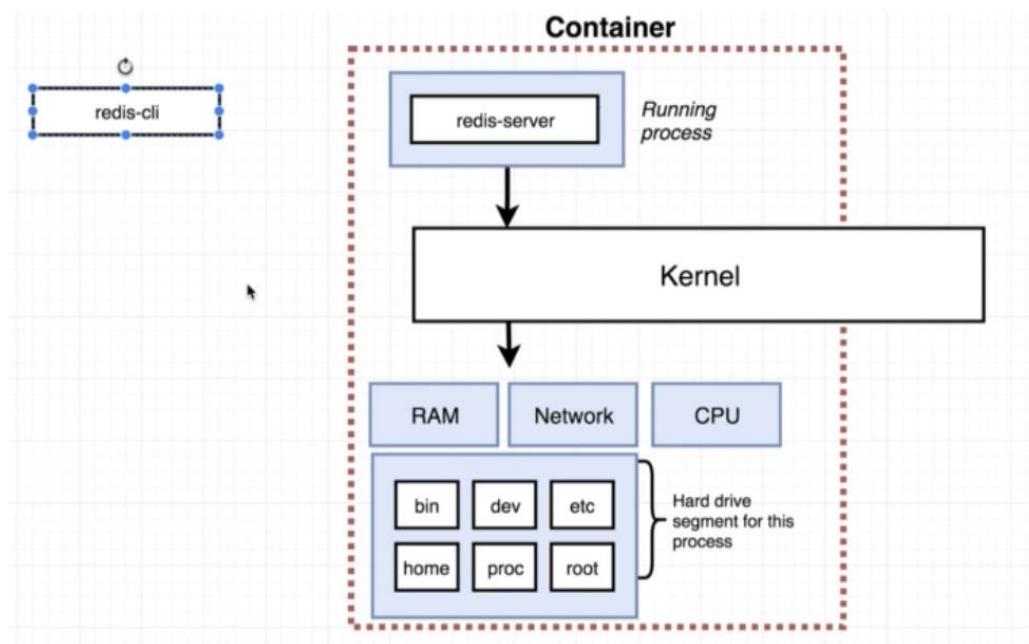
Running a Redis using Docker

Redis server

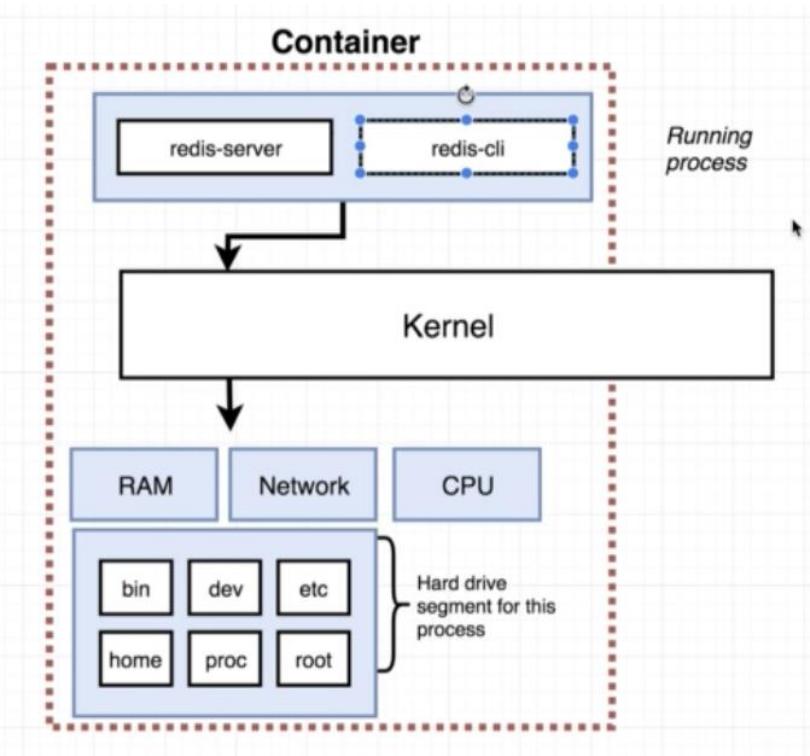
```
→ ~ docker run redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
5eb5b503b376: Pull complete
6530a7ea3479: Pull complete
91f5202c6d9b: Pull complete
9f1ac212e389: Pull complete
82c311187b72: Pull complete
da84aa65ce64: Pull complete
Digest: sha256:0d9c9aed1eb385336db0bc9b976b6b49774aee3d2b9c2788a0d0d9e239986cb3
Status: Downloaded newer image for redis:latest
1:C 27 Feb 2022 14:19:12.305 # o000o000o000 Redis is starting o000o000o000
1:C 27 Feb 2022 14:19:12.305 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 27 Feb 2022 14:19:12.305 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 27 Feb 2022 14:19:12.306 * monotonic clock: POSIX clock_gettime
1:M 27 Feb 2022 14:19:12.307 * Running mode=standalone, port=6379.
1:M 27 Feb 2022 14:19:12.307 # Server initialized
1:M 27 Feb 2022 14:19:12.308 * Ready to accept connections
```

```
→ ~ redis-cli
zsh: command not found: redis-cli
→ ~
```

We are trying to run the redis-cli command outside the container.

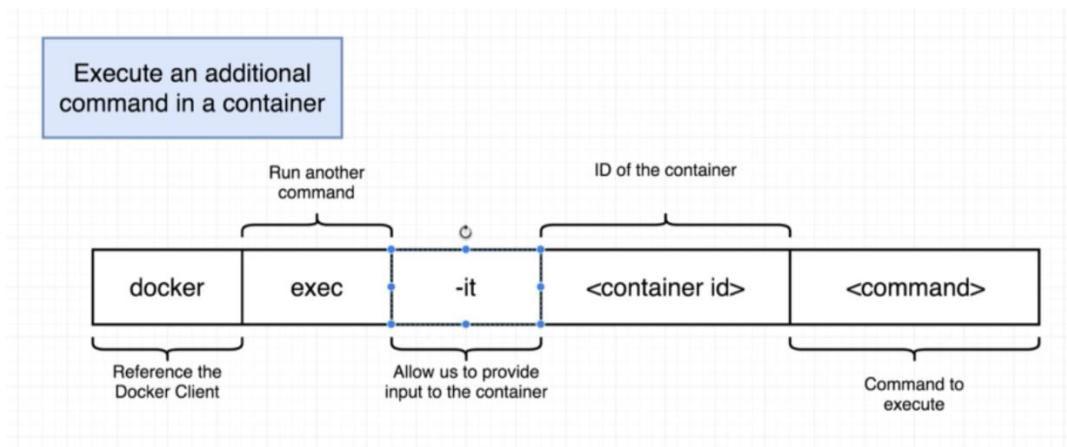


Redis-cli will work we are trying to run int same container.



Executing Command in Running Container

Docker exec Command



```

→ ~ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c3cc7e3c8642 redis "docker-entrypoint.s..." 12 minutes ago Up 12 minutes 6379/tcp elated_
cori
→ ~ docker exec -it c3cc7e3c8642 redis-cli
127.0.0.1:6379> set a 5
OK
127.0.0.1:6379> get a
"5"
127.0.0.1:6379>

```

Purpose of the “-it” flag

```

→ ~ docker exec -it c3cc7e3c8642 redis-cli
127.0.0.1:6379> set a 5
OK
127.0.0.1:6379> get a
"5"
127.0.0.1:6379>
→ ~ docker exec c3cc7e3c8642 redis-cli
→ ~

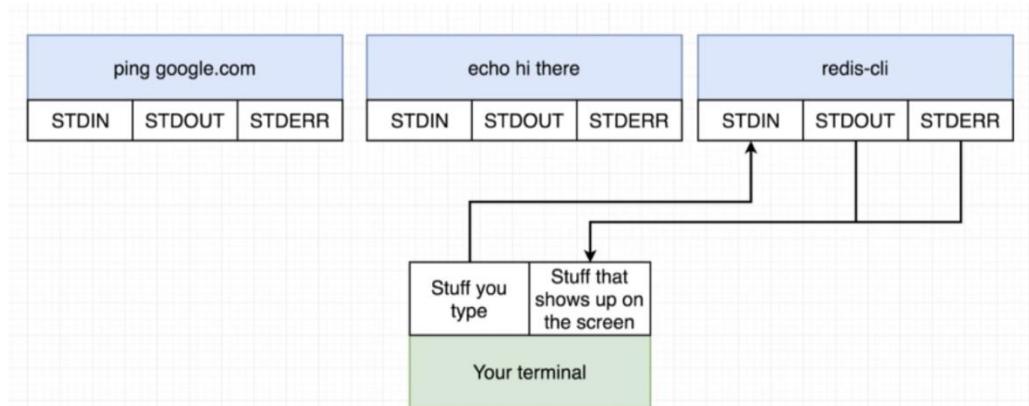
```

Running command without “-it” flag, could not enter redis-cli, so it just moved out.

How processes run in Linux environment

Every container will run in Linux virtual machine. So, all the processes in container are runs in Linux environment.

Every process we create in a Linux environment, three communication channels attached to it. That is STDIN, STDOUT and STDERR.



How does it relate to “-it” flag?

Command “-it” flag

We can split the “-it” flag into two parts.

-it = -i -t

-i ➔ Stuff we type will be sent to the process's STDIN communication channel.

-t ➔ This flag responsible for make the content nicely formatted on screen.

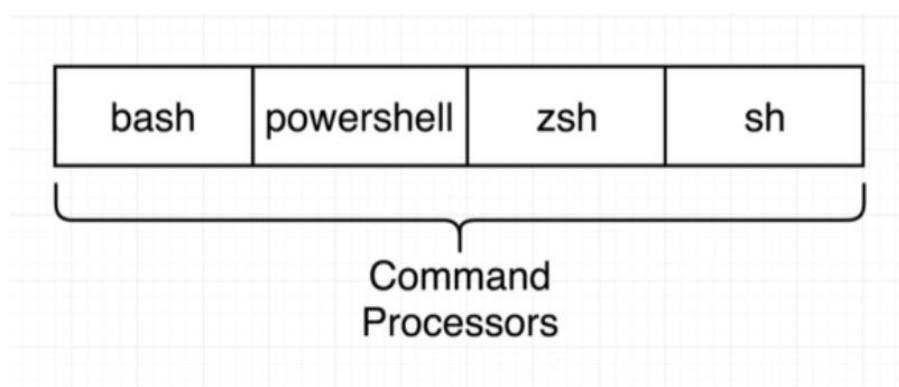
Without -t flag

```
→ prod git:(master) ✘ docker exec -i 093b6e72ff2b redis-cli
set myvalue 5
OK
get myvalue
5
```

Getting a Command Prompt in Container

Command Processors

We can enable the command processors for our container.



Enable Command Prompt

Enable the command prompt for container

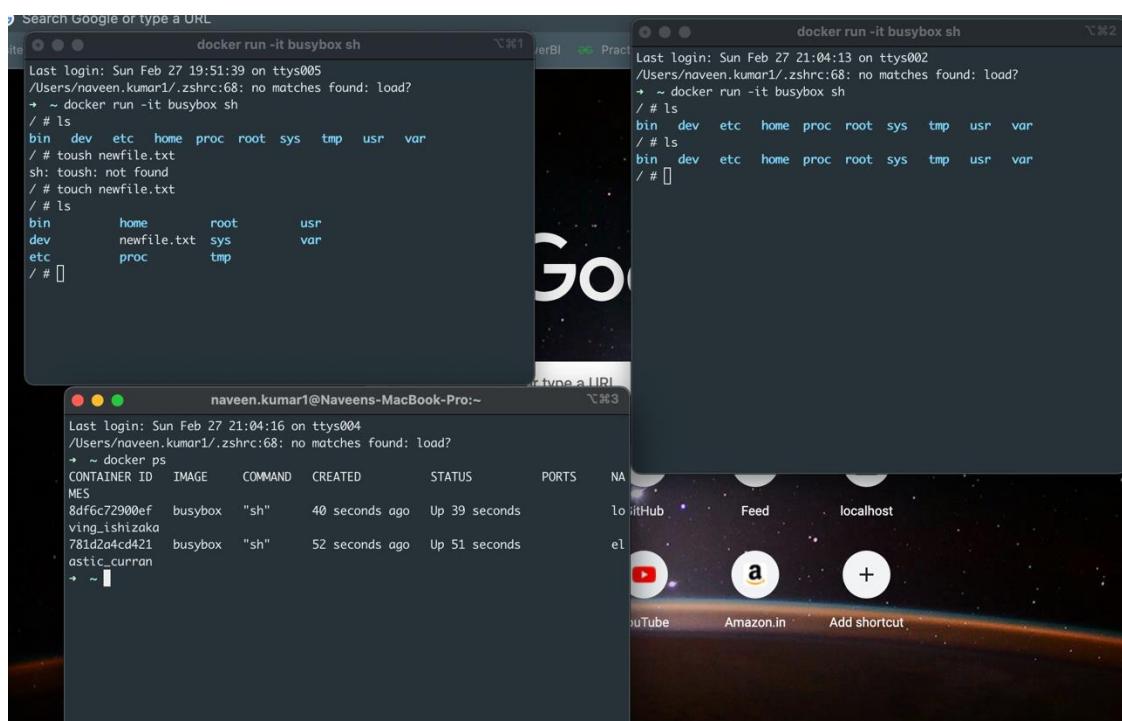
```
→ ~ docker exec -it c3cc7e3c8642 sh
# redis-cli
127.0.0.1:6379> set b 2
OK
127.0.0.1:6379> get b
"2"
127.0.0.1:6379>
# export a=5
# echo $a
5
# ls
# cd ~/
# ls
# cd /
# ls
bin  data  etc  lib   media  opt   root  sbin  sys  usr
boot dev   home lib64 mnt    proc   run   srv   tmp  var
#
```

Starting with a Shell

```
→ ~ docker run -it busybox sh
/ # ping google.com
PING google.com (142.250.67.46): 56 data bytes
64 bytes from 142.250.67.46: seq=0 ttl=37 time=50.944 ms
64 bytes from 142.250.67.46: seq=1 ttl=37 time=48.368 ms
64 bytes from 142.250.67.46: seq=2 ttl=37 time=37.944 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 37.944/45.752/50.944 ms
/ # echo hi there
hi there
/ # echo bye!!
bye!!
/ # ls
bin dev etc home proc root sys tmp usr var
/ # [
```

Container Isolation

Demo and Explanation



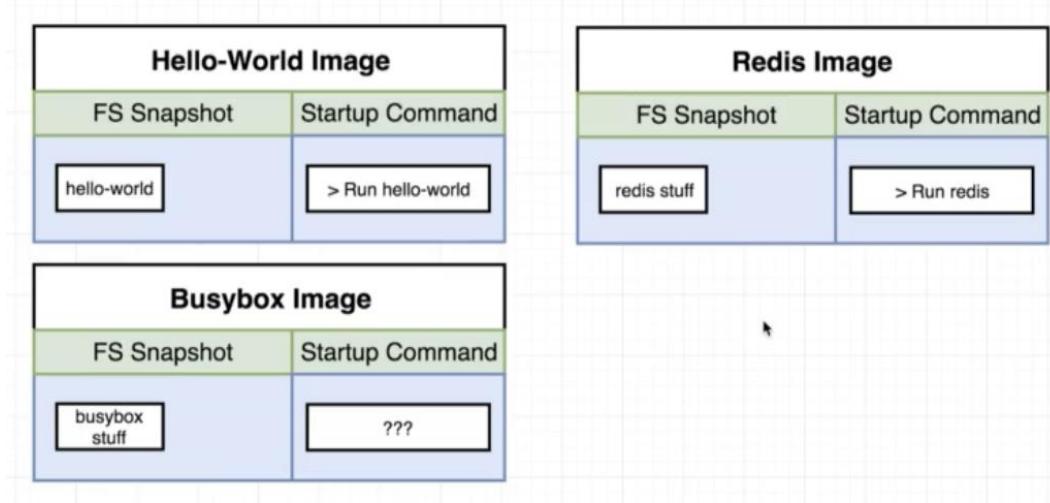
- We have created two different containers for the same image “busybox”.
- Created a new file called “newfile.txt” in the first container’s file systems.
- We tried to list all the files in second container’s file systems. We could see that “newfile.txt” file is not listed over there.

From this, we can conclude, each container having its own resources. Resources are not being shared among the containers.

Building Custom Images Through Docker Server

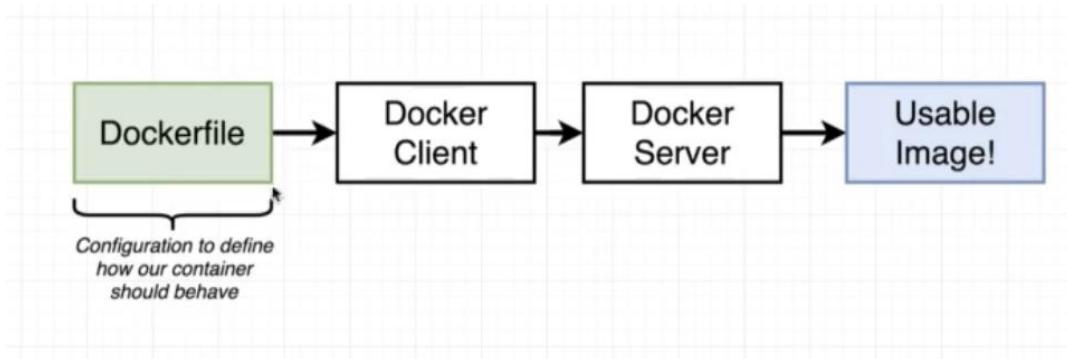
Creating Docker Images

Before we have been using an image created by other engineers.



Now, we are going to figure out how can we build own custom images.

Introducing Docker File



Docker File contains,

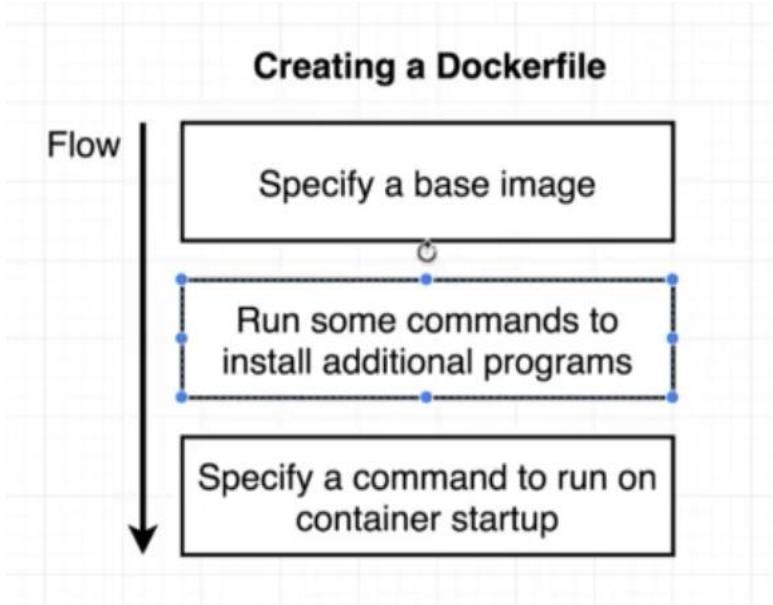
- What are the programs to run?
- What images does when starting a container?
- Configurations and dependencies required for program to run.

We will be passing the docker file to docker client (docker cli), that in turn pass it to docker server.

Docker server is what is doing the heavy lifting for us. It takes the docker file and look at all the lines of configurations and then build a usable image.

Then, that image is used for creating a container.

Docker File Creation Flow



Building a Docker Image

Creating a Docker File

```
FROM alpine
RUN apk add --update redis
CMD ["redis-server"]
```

Docker Build

```
+ REDIS_IMAGE docker build .
[+] Building 11.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 237B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:latest
=> [auth] library/alpine:pull token for registry-1.docker.io
=> [1/2] FROM docker.io/library/alpine@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300
=> => resolve docker.io/library/alpine@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300
=> sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300 1.64kB / 1.64kB
=> => sha256:e7d8de73db3d3fd9b2d63aa7f447a10fd0220b7cbf39803c803f2af9ba256b3 528B / 528B
=> => sha256:c059bfaa849c4d8e4aecaebe3a10c2d9b3d85f5165c66ad3a4d937758128c4d18 1.47kB / 1.47kB
=> => sha256:59bf1c3500f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3 2.82MB / 2.82MB
=> => extracting sha256:59bf1c3500f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3
=> [2/2] RUN apk add --update redis
=> exporting to image
=> => exporting layers
=> => writing image sha256:9183a3ffd60e420fc437e7d0914f395c128f4f1ae1b90dee8bcbe4529b41dc07
```

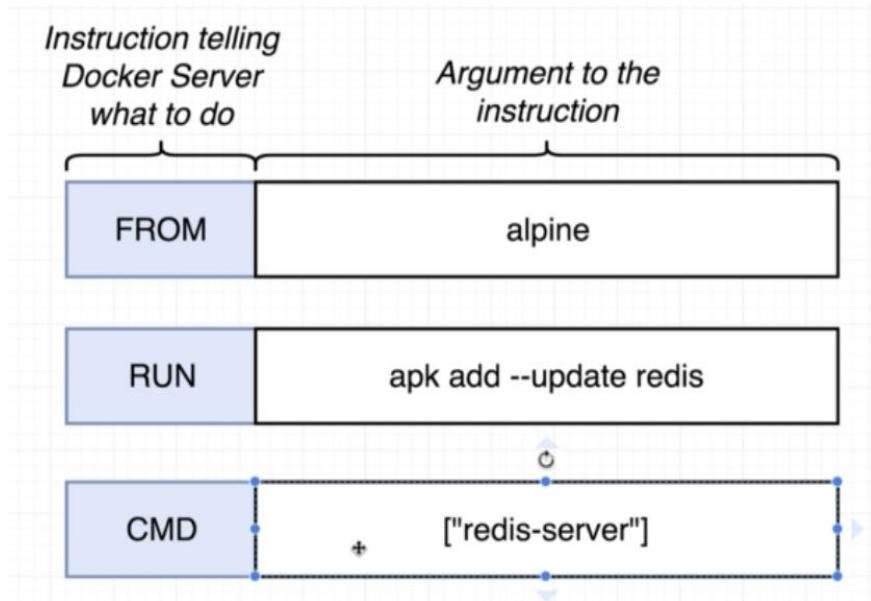
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

Running an Image

```
=> exporting to image
=> => exporting layers
=> => writing image sha256:9183a3ffd60e420fc437e7d0914f395c128f4f1ae1b90dee8bcbe4529b41dc07

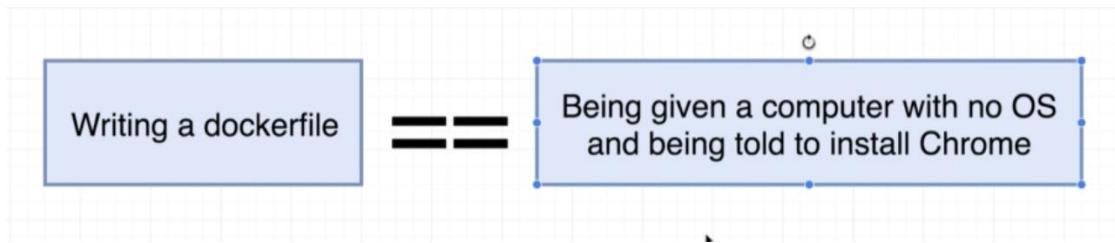
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ REDIS_IMAGE docker run 9183a3ffd60e420fc437e7d0914f395c128f4f1ae1b90dee8bcbe4529b41dc07
1:C 28 Feb 2022 19:57:15.736 # o000o000o000o Redis is starting o000o000o000o
1:C 28 Feb 2022 19:57:15.736 # Redis version=6.2.6, bits=64, commit=b39e1241, modified=0, pid=1, just
1:C 28 Feb 2022 19:57:15.736 # Warning: no config file specified, using the default config. In order
redis-server /path/to/redis.conf
1:M 28 Feb 2022 19:57:15.737 * monotonic clock: POSIX clock_gettime
1:M 28 Feb 2022 19:57:15.738 * Running mode=standalone, port=6379.
1:M 28 Feb 2022 19:57:15.738 # Server initialized
1:M 28 Feb 2022 19:57:15.739 * Ready to accept connections
```

Dockerfile Teardown

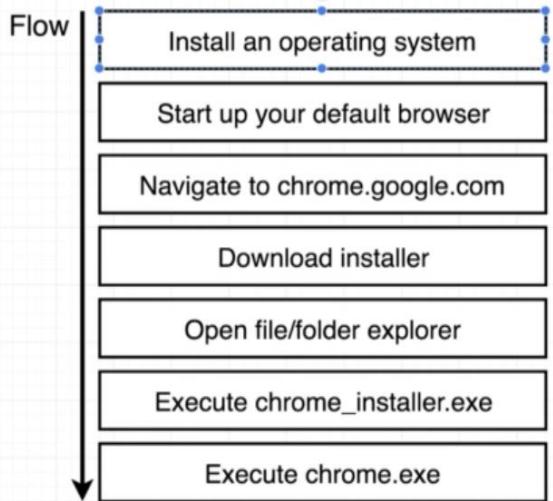


What is Base Image?

Analogy

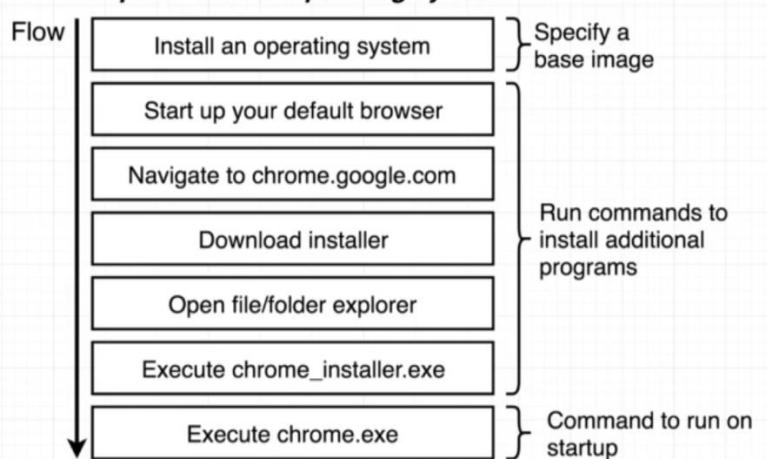


How do you install Chrome on a computer with no operating system?



Base Image

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



Purpose of giving a base image is initial starting point or initial set of programs used for configuring an image further.

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!

Base image contains all set of necessary programs to run, run commands.

The Build Process in Detail

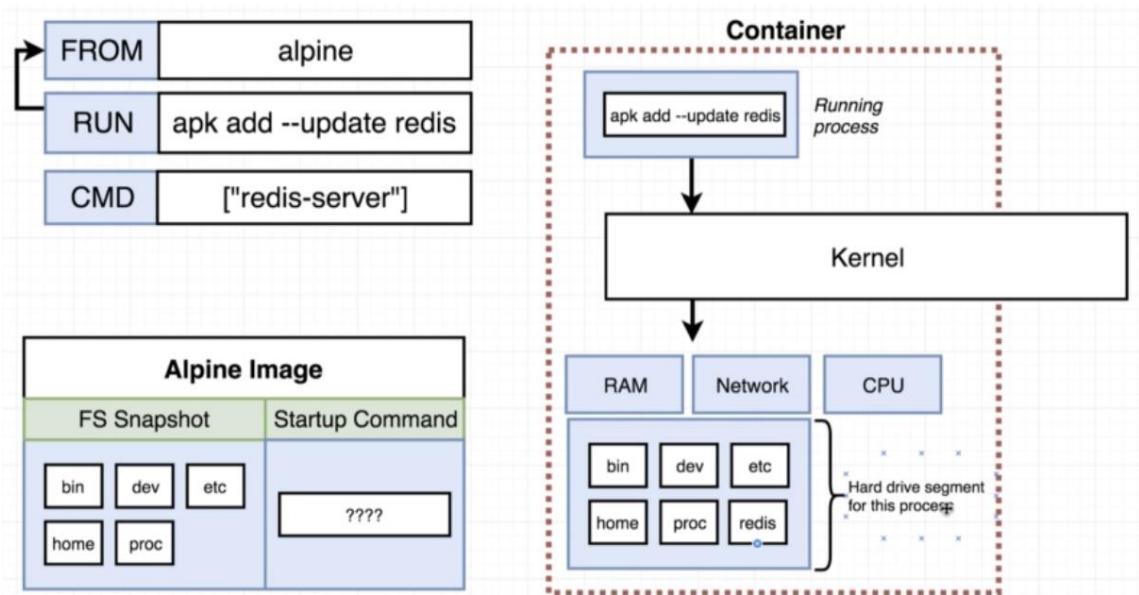
```
→ REDIS_IMAGE docker build .
[+] Building 11.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 237B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:latest
=> [auth] library/alpine:pull token for registry-1.docker.io
=> [1/2] FROM docker.io/library/alpine@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300
=> => resolve docker.io/library/alpine@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300
=> => sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300 1.64kB / 1.64kB
=> => sha256:e7d88de73db3d3fd9b2d63aa7f447a10fd0220b7cbf39803c803f2af9a256b3 528B / 528B
=> => sha256:c059bfaa849c4d8e4ecaeab310c2d9b3d85f5165c66ad3a4d937758128c4d18 1.47kB / 1.47kB
=> => sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3 2.82MB / 2.82MB
=> => extracting sha256:59bf1c3509f33515622619af21ed55bbe26d24913cedbca106468a5fb37a50c3
=> [2/2] RUN apk add --update redis
=> exporting to image
=> => exporting layers
=> => writing image sha256:9183a3ffd60e420fc437e7d0914f395c128f4f1ae1b90dee8bcbe4529b41dc07
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

What “docker build .” command is doing?

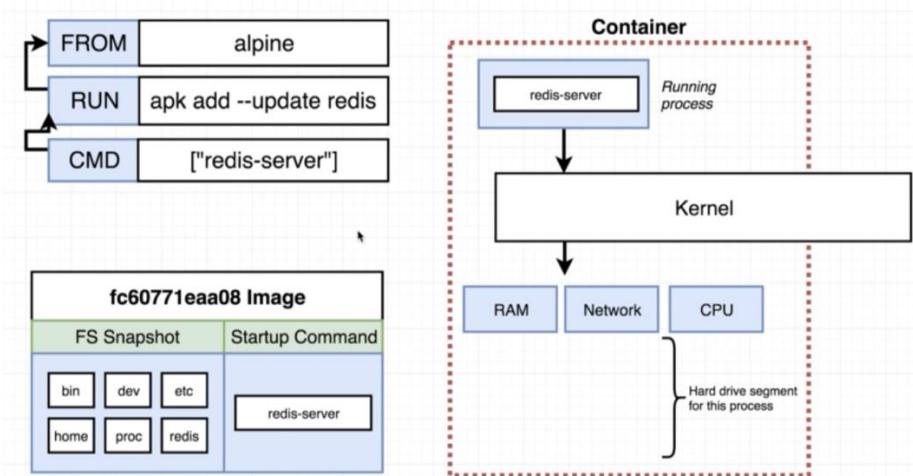
- Giving Dockerfile to docker cli.
- Build command takes the docker file and creates an image out of it.
- “.” in command specify the build context. Set of file and folders, that we want wrap and bring it into the container.

Building Process

- Checks local cache, whether base image is available or not.
- If it is not available, image will be download from the docker hub.
- Next step is running the run commands.
- For, running the num command, docker make use of the base image and creates temporary container out of it.

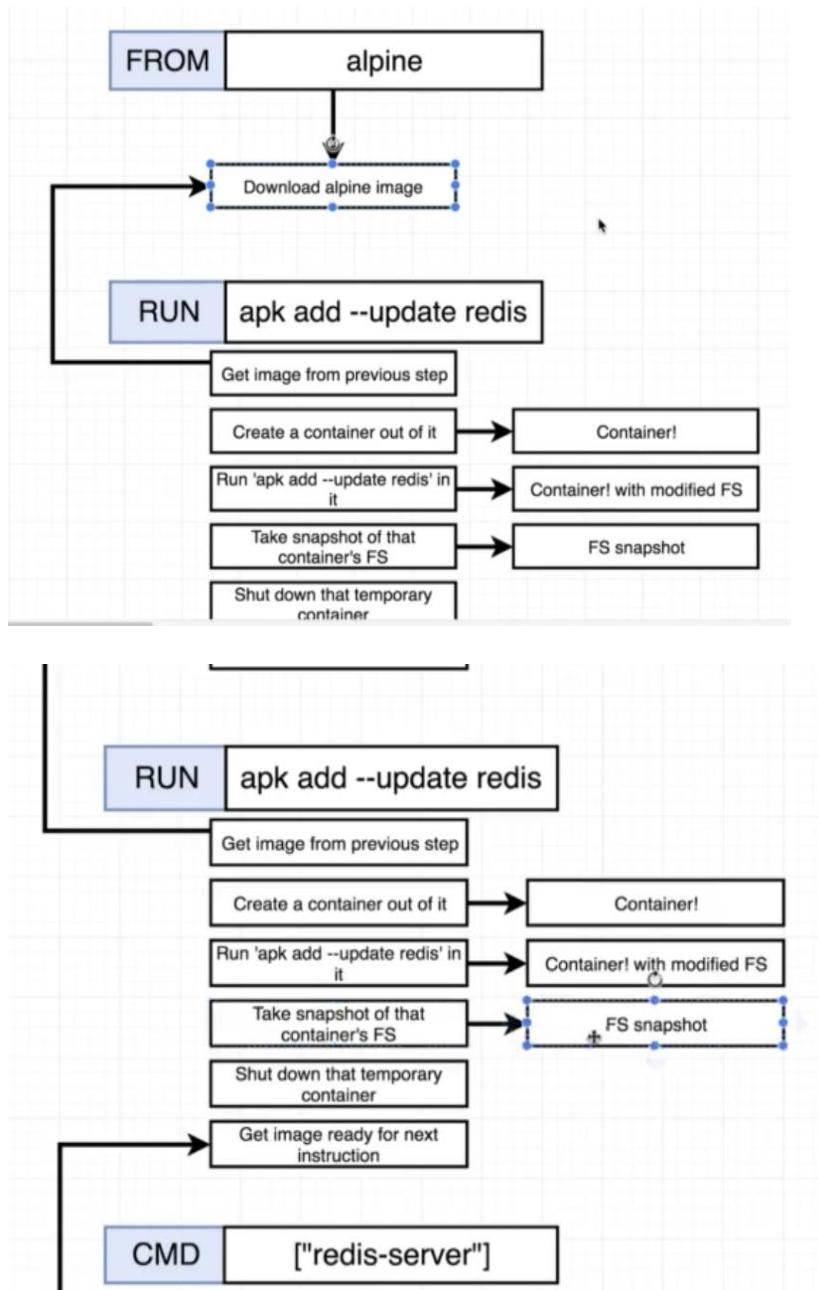


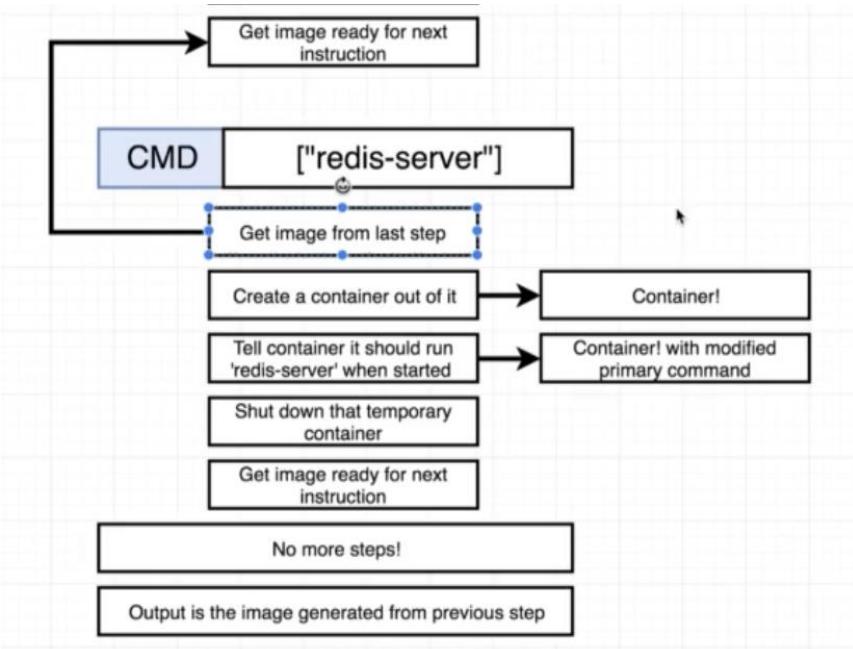
- Once the running command is installed, we are taking all the file system back from the container and stopping it.
- Run command installed file system is saved in temporary image.
- CMD command is updated as startup command in saved temporary image, that in turn creates a new image and gives the result as image id.



Brief Recap

Recap





Rebuild with Cache

```

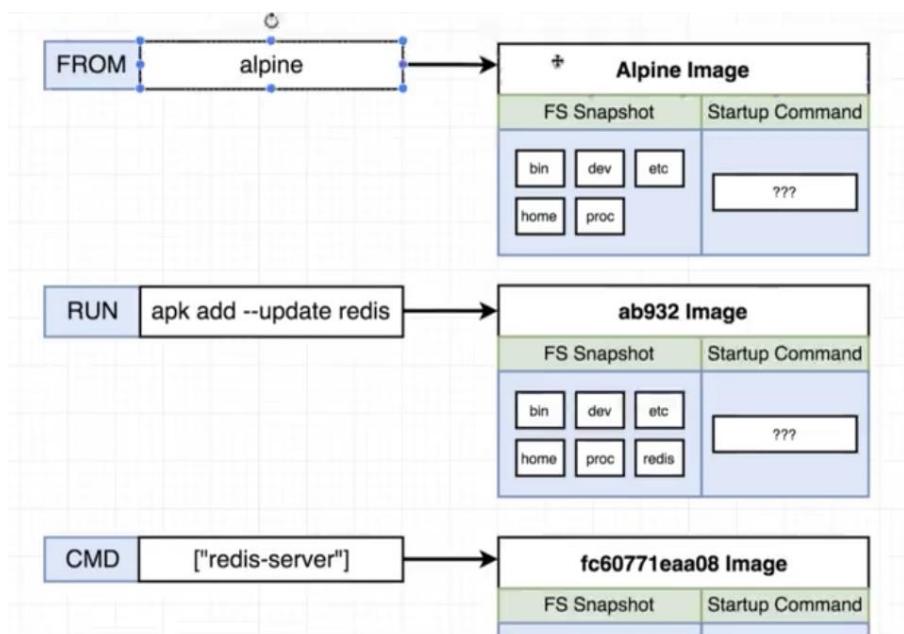
# Use an existing docker image as a base
FROM alpine

# Download and install a dependency
RUN apk add --update redis

# Tell the image what to do when it starts
# as a container
CMD ["redis-server"]

```

We have three instructions, for each instruction we will be getting the new image.



```
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 RUN apk add --update gcc
7
8 # Tell the image what to do I when it starts
9 # as a container
10 CMD ["redis-server"]
```

We updated the Dockerfile with one more RUN instruction.

For running the instruction, we are using the image. If the image is already created and stored in cache, we are reusing the same.

```
Successfully built b90e9fb1e7e0
→ 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
--> Using cache
--> a6d5a254e65b
Step 4/4 : CMD ["redis-server"]
--> Using cache
--> b90e9fb1e7e0
Successfully built b90e9fb1e7e0
```

If the series of the steps from the Dockerfile is same as before while building, we are reusing the image from cache.

Tagging an Image

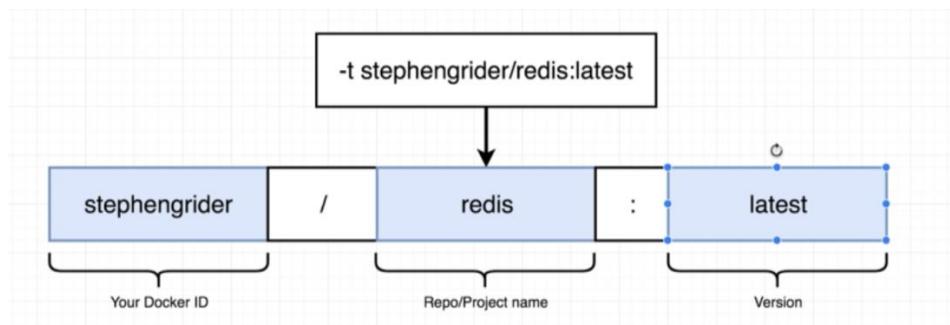
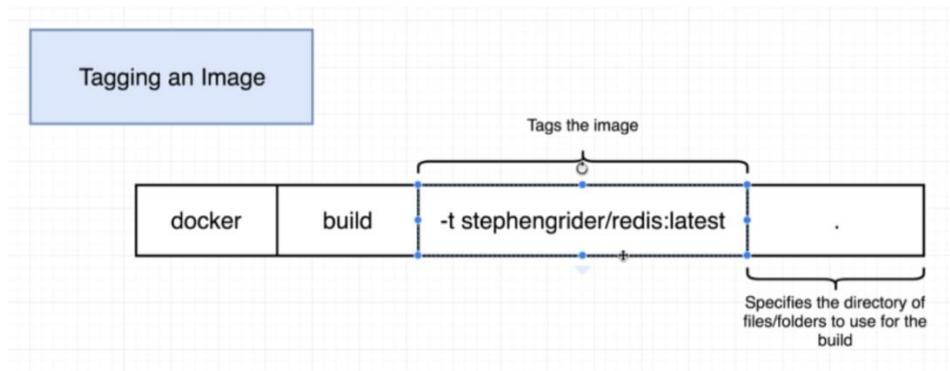
Explanations

Tagging an image name while building the Dockerfile.

Docker run hello-world

Docker run redis

Docker run busybox

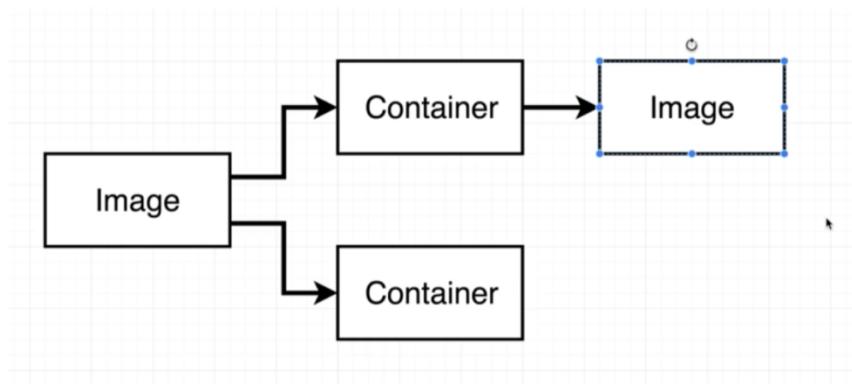


Executions

```
1. stephengrider@stephens-MacBook-Pro:~/workspace/DockerWorkspace/proj/redis-image (zsh)
→ 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
```

Manual Image Generation with Docker Commit

Explanations



We could create an Image from container. This approach is equivalent to building an image from docker file.

Note: Do not recommended to do it in projects.

Demo

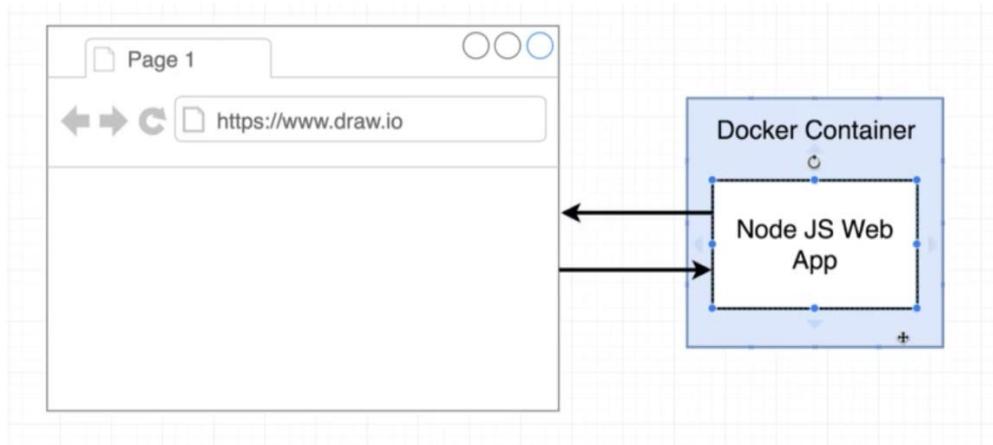
```
redis-image git:(master) ✘ docker run -it alpine sh
/ # apk add --update redis
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/main/x86_64/APKINDEX.t
r.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.8/community/x86_64/APKIND
EX.tar.gz
(1/1) Installing redis (4.0.10-r1)
Executing redis-4.0.10-r1.pre-install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

```
Last login: Wed Aug  8 13:07:25 on ttys002
redis-image git:(master) ✘ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            NAMES
39075447a383        alpine              "sh"               48 seconds ago   friendly_brown
Up 47 seconds
redis-image git:(master) ✘ docker commit -c 'CMD ["redis-server"]' 39075
447a383
sha256:0835898662afae4b5a9d9e9961a0e89c2cbe55d895a0c0fbe6b7664f980d272b
redis-image git:(master) ✘ docker run 0835898662af
```

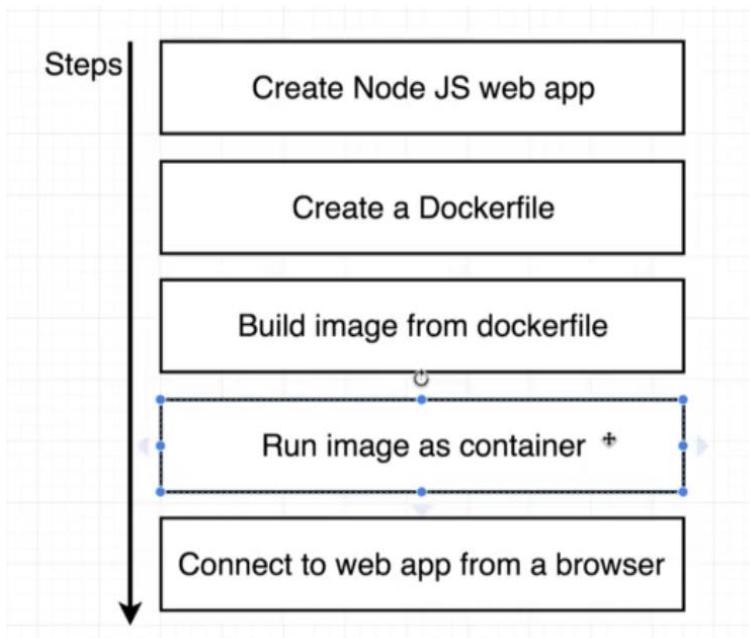
Making Real Projects with Docker

Creating a Sample Project

Project Outline



Project Plan



Node Server Setup

EXPLORER

01_simpleweb > package.json > {} scripts

```

1  {
2    "dependencies": {
3      "express": "*"
4    },
5    "scripts": {
6      "start": "node index.js"
7    }
8  }
9

```

01_simpleweb > index.js > app.get("/") callback

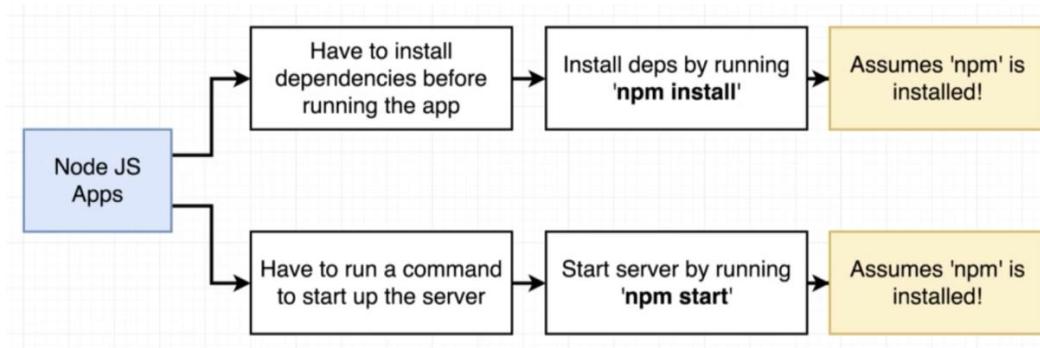
```

1  const express = require("express");
2
3  const app = express();
4
5  app.get("/", (req, res) => {
6    res.send("Hi There");
7  );
8
9  app.listen(8080, () => {
10   console.log("Listing on port 8080");
11 });
12

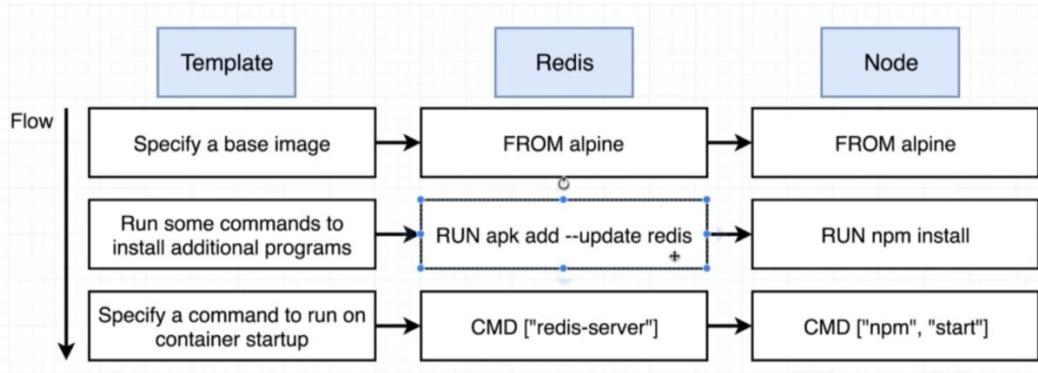
```

Buidling a Docker Image for Our Project

npm Setup



Step for Building Docker Image for Our Node Application



Creating a Docker File

```

> 00_redis_image
< 01_simpleweb
|   node_modules
|     Dockerfile
|     index.js
|     package-l...
|     package.json
|     .gitignore

```

```

1 #Specify a base image
2 FROM alpine
3
4 #Install some dependencies
5 RUN npm install
6
7 #Default command
8 CMD ["npm", "start"]

```

The terminal shows the directory structure of a project. It includes a '00_redis_image' folder, a '01_simpleweb' folder which contains a 'node_modules' folder and a 'Dockerfile'. The 'Dockerfile' is displayed in the terminal, showing its contents:

Building a Docker File

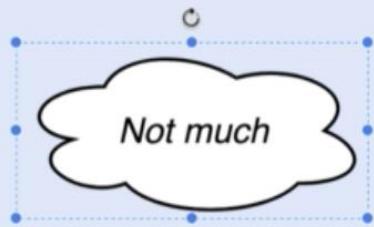
```

→ 01_simpleweb git:(main) ✘ docker build .
[+] Building 3.6s (6/6) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 157B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/alpine:lat 3.3s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> CACHED [1/2] FROM docker.io/library/alpine@sha256:21a3dea 0.0s
=> ERROR [2/2] RUN npm install                           0.2s
> [2/2] RUN npm install:
#6 0.206 /bin/sh: npm: not found
executor failed running [/bin/sh -c npm install]: exit code: 127
→ 01_simpleweb git:(main) ✘

```

Here we are getting npm not found error. Because our base image “alpine” file system does not have npm installed.

Programs Included in the Alpine Image



To solve this issue, we have two options. That is,

1. Find a different base image where npm pre-installed.
2. We can continue keeping “alpine” base image, install npm as dependency.

For this example, we are going with first option. For that, we need to find a base image which have a npm pre-installed.

Exploring Public Images from Docker Hub

<https://hub.docker.com/search?type=image>

The screenshot shows the Docker Hub search interface. The URL in the address bar is <https://hub.docker.com/search?type=image&category=base>. The search bar contains the query "Base Images". On the left, there are filters for "Images" (including "Verified Publisher" and "Official Images") and "Categories" (including "Analytics", "Application Frameworks", "Application Infrastructure", "Application Services", "Base Images" which is checked, "Databases", "DevOps Tools", "Featured Images", "Messaging Services", and "Monitoring"). The main search results display three images:

- ubuntu**: Official Image. Updated a month ago. 1B+ Downloads, 10K+ Stars. Tags: Container, Linux, PowerPC 64 LE, riscv64, IBM Z, 386, x86-64, ARM, ARM 64, Base Images, Operating Systems.
- alpine**: Official Image. Updated 3 months ago. 1B+ Downloads, 8.5K Stars. Tags: Container, Linux, IBM Z, 386, riscv64, PowerPC 64 LE, ARM 64, ARM, x86-64, Featured Images, Base Images, Operating Systems.
- busybox**: Official Image. Updated a month ago. 1B+ Downloads, 2.5K Stars. Tags: Container, Linux, IBM Z, 386, riscv64, PowerPC 64 LE, ARM 64, ARM, x86-64, Base Images.

The screenshot shows the Docker Hub interface for the 'node' image. At the top, there's a search bar with 'Search for great content', navigation links for 'Explore', 'Repositories', 'Organizations', 'Help', and a yellow 'Upgrade' button. Below the header, the breadcrumb navigation shows 'Explore > Official Images > node'. On the left, there's a Docker logo icon. The main content area features the 'node' image card, which includes a small icon of a ship with a stack of shipping containers, the name 'node', a green 'Official Image' badge with a checkmark, and a star icon. A brief description states: 'Node.js is a JavaScript-based platform for server-side and networking applications.' Below the description, there's a download count of '1B+' and a 'Copy and paste to pull this image' button with a 'docker pull node' command and a clipboard icon. There are also tabs for 'Container', 'Linux', 'ARM 64', 'PowerPC 64 LE', 'IBM Z', '386', 'x86-64', 'ARM', 'Application Infrastructure', and 'Official Image'. At the bottom of the card, there are links for 'Description', 'Reviews', and 'Tags'.

Versions available for node image

Supported tags and respective Dockerfile links

- 17-alpine3.14, 17.6-alpine3.14, 17.6.0-alpine3.14, alpine3.14, current-alpine3.14
- 17-alpine, 17-alpine3.15, 17.6-alpine, 17.6-alpine3.15, 17.6.0-alpine, 17.6.0-alpine3.15, alpine, alpine3.15, current-alpine, current-alpine3.15
- 17, 17-bullseye, 17.6, 17.6-bullseye, 17.6.0, 17.6.0-bullseye, bullseye, current, current-bullseye, latest
- 17-bullseye-slim, 17-slim, 17.6-bullseye-slim, 17.6-slim, 17.6.0-bullseye-slim, 17.6.0-slim, bullseye-slim, current-bullseye-slim, current-slim, slim
- 17-buster, 17.6-buster, 17.6.0-buster, buster, current-buster
- 17-buster-slim, 17.6-buster-slim, 17.6.0-buster-slim, buster-slim, current-buster-slim
- 17-stretch, 17.6-stretch, 17.6.0-stretch, current-stretch, stretch

Example of Defining Version Along with Image

```

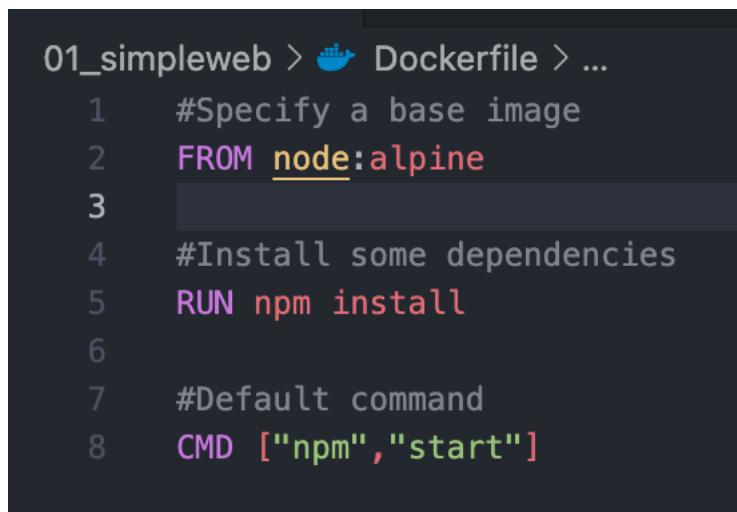
# Specify a base image
FROM node:alpine

# Install some dependencies
RUN npm install

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

```

Updating the Dockerfile



```

01_simpleweb > 🛠 Dockerfile > ...
1 #Specify a base image
2 FROM node:alpine
3
4 #Install some dependencies
5 RUN npm install
6
7 #Default command
8 CMD ["npm","start"]

```

Building Dockerfile Again

```

[+] Building 26.3s (6/6) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 162B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/node:alpine 4.7s
=> [auth] library/node:pull token for registry-1.docker.io  0.0s
=> [2/2] FROM docker.io/library/node:alpine@sha256:250e9a093b86 20.5s
=> => resolve docker.io/library/node:alpine@sha256:250e9a093b861 0.0s
=> => sha256:e4e269ec1fe432faa79436b86ac6a194f9bc19d 449B / 449B 1.4s
=> => sha256:250e9a093b861c330be2f4d1d224712d4e4 1.43kB / 1.43kB 0.0s
=> => sha256:13961fc7673e7a62de914c5978336408703 1.16kB / 1.16kB 0.0s
=> => sha256:eb56d56623e54e8819e79c9ee3c5cd5d4db 6.53kB / 6.53kB 0.0s
=> => sha256:13f72b2a902f68ec36db45f4738973f7 45.84MB / 45.84MB 17.9s
=> => sha256:3bd6b5da901231b85e31594a95a26439b24 2.34MB / 2.34MB 3.0s
=> => extracting sha256:13f72b2a902f68ec36db45f4738973f76e70a2e5 2.2s
=> => extracting sha256:3bd6b5da901231b85e31594a95a26439b2466fa0 0.1s
=> => extracting sha256:e4e269ec1fe432faa79436b86ac6a194f9bc19d6 0.0s
=> ERROR [2/2] RUN npm install                           1.0s
-----
> [2/2] RUN npm install:
#6 0.808 npm ERR! Tracker "idealTree" already exists
#6 0.810
#6 0.810 npm ERR! A complete log of this run can be found in:
#6 0.811 npm ERR!     /root/.npm/_logs/2022-03-03T08_04_31_026Z-debug-0.
log
-----
executor failed running [/bin/sh -c npm install]: exit code: 1

```

To solve this error,

npm ERR! idealTree already exists

To resolve this, add a WORKDIR right after the FROM instruction: (we will be adding this very soon anyway)

```
1 | FROM node:alpine
2 |
3 | WORKDIR /usr/app
```

Also, you will no longer get the error regarding the missing

Updating the Dockerfile,

```
01_simpleweb > 🛠 Dockerfile > ...
1 #Specify a base image
2 FROM node:alpine
3
4 #Work Directory
5 WORKDIR /usr/app
6
7 #Install some dependencies
8 RUN npm install
9
10 #Default command
11 CMD ["npm","start"]
```

Running a docker build again,

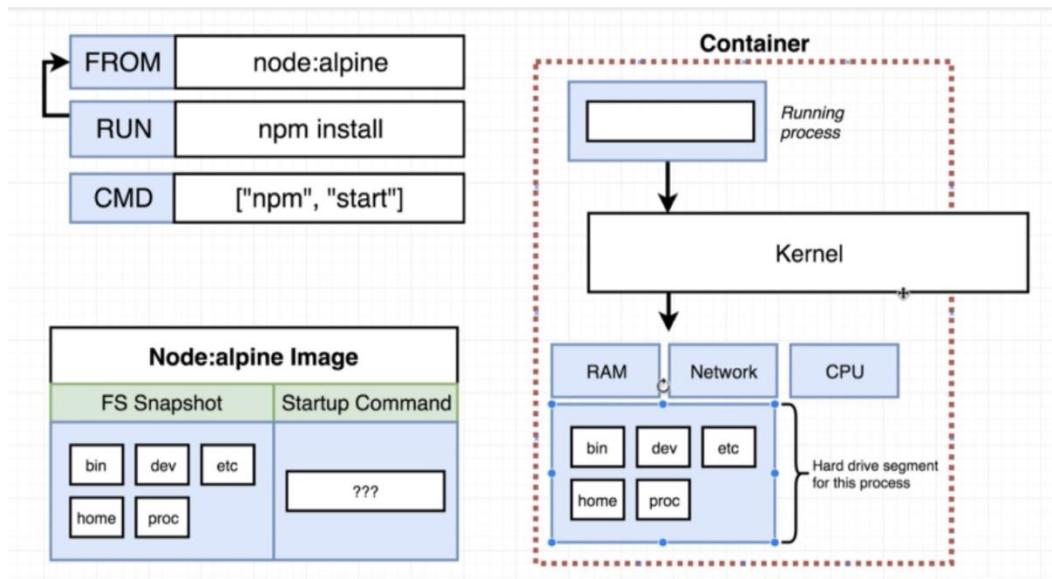
```

executor failed running [/bin/sh -c npm install]: exit code: 1
→ 01_simpleweb git:(main) ✘ docker build .
[+] Building 5.4s (7/7) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 198B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/node:alpine 4.5s
=> [auth] library/node:pull token for registry-1.docker.io   0.0s
=> CACHED [1/3] FROM docker.io/library/node:alpine@sha256:250e9a 0.0s
=> [2/3] WORKDIR /usr/app                                0.0s
=> ERROR [3/3] RUN npm install                           0.8s
-----
> [3/3] RUN npm install:
#7 0.799 npm ERR! code ENOENT
#7 0.800 npm ERR! syscall open
#7 0.800 npm ERR! path /usr/app/package.json
#7 0.801 npm ERR! errno -2
#7 0.803 npm ERR! enoent ENOENT: no such file or directory, open '/usr/a
pp/package.json'
#7 0.803 npm ERR! enoent This is related to npm not being able to find a
file.
#7 0.804 npm ERR! enoent
#7 0.805
#7 0.805 npm ERR! A complete log of this run can be found in:
#7 0.805 npm ERR!     /root/.npm/_logs/2022-03-03T08_09_51_956Z-debug-0.
log
-----
executor failed running [/bin/sh -c npm install]: exit code: 254
→ 01_simpleweb git:(main)

```

Why “no such file or directory, open '/usr/app/package.json'” ?

Because there is no, package.json file inside the container file system. We are using only the segment of the hard drive where container filesystem lies in. We are still have the rest of the hard drive part where package.json exist. So container is not communicating with rest of the hard drive part.



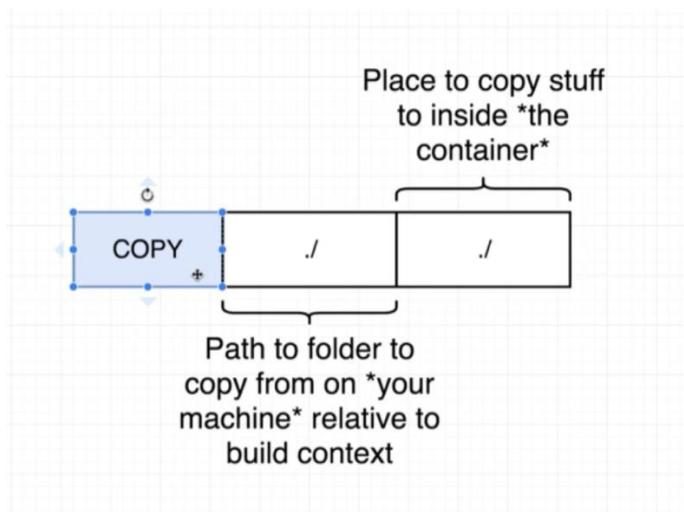
Our project files won't come into container by default. We need to explicitly specify in Dockerfile.

The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree under 'DOCKER_KUBE'. It includes a folder '00_redis_image', a folder '01_simpleweb' which contains 'node_modules', 'Dockerfile', 'index.js', 'package-lock.json', 'package.json', and '.gitignore'. The 'Dockerfile' tab is selected in the main editor area. The code in the editor is:

```
1 #Specify a base image
2 FROM node:alpine
3
4 #Work Directory
5 WORKDIR /usr/app
6
7 #Install some dependencies
8 RUN npm install
9
10 #Default command
11 CMD ["npm","start"]
```

Copying Build Files

Explanations



./ is a build context.

Updating the docker file

The screenshot shows the VS Code interface with the Dockerfile open. The 'COPY ./ ./' line is highlighted with a green selection bar. The code in the editor is:

```
1 #Specify a base image
2 FROM node:alpine
3
4 #Work Directory
5 WORKDIR /usr/app
6
7 #Install some dependencies
8 COPY ./ ./ RUN npm install
9
10 #Default command
11 CMD ["npm","start"]
```

Building the Docker Image

```
→ 01_simpleweb git:(main) ✘ docker build .
[+] Building 6.7s (10/10) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 209B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/node:alpine 4.2s
=> [auth] library/node:pull token for registry-1.docker.io   0.0s
=> [1/4] FROM docker.io/library/node:alpine@sha256:250e9a093b 0.0s
=> [internal] load build context                          0.1s
=> => transferring context: 1.76MB                      0.1s
=> CACHED [2/4] WORKDIR /usr/app                         0.0s
=> [3/4] COPY ./ ./                                       0.0s
=> [4/4] RUN npm install                                2.1s
=> exporting to image                                  0.1s
=> => exporting layers                                 0.1s
=> => writing image sha256:f95c2b702af5a79c1f0648f4a4421d9a9f 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ 01_simpleweb git:(main) ✘
```

Building Docker Image with Tagging an Image Name

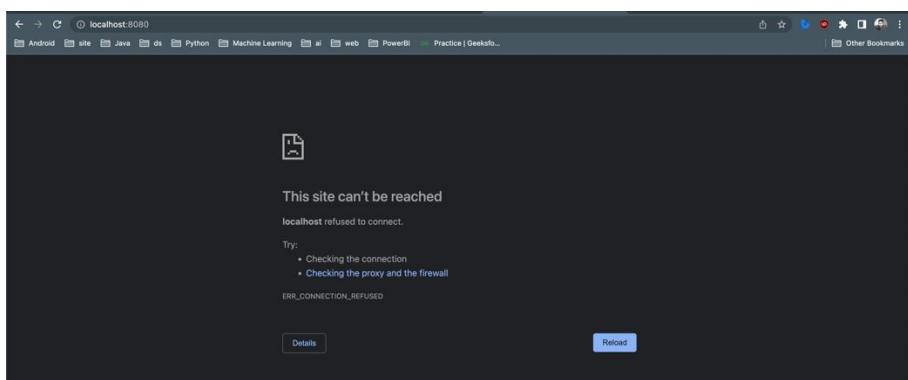
```
→ 01_simpleweb git:(main) ✘ docker build -t mnavveensmn/simpleweb .
[+] Building 1.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 37B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/node:alpine 1.3s
=> [internal] load build context                          0.0s
=> => transferring context: 20.96kB                      0.0s
=> [1/4] FROM docker.io/library/node:alpine@sha256:250e9a093b861c330be2f4d1d224712d4e 0.0s
=> CACHED [2/4] WORKDIR /usr/app                         0.0s
=> CACHED [3/4] COPY ./ ./                               0.0s
=> CACHED [4/4] RUN npm install                         0.0s
=> exporting to image                                  0.0s
=> => exporting layers                                 0.0s
=> => writing image sha256:f95c2b702af5a79c1f0648f4a4421d9a9fa08a0b31cc437d2ee5f282fd 0.0s
=> => naming to docker.io/mnavveensmn/simpleweb        0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ 01_simpleweb git:(main) ✘
```

Running a Docker Image

```
→ 01_simpleweb git:(main) ✘ docker run mnavveensmn/simpleweb
> start
> node index.js

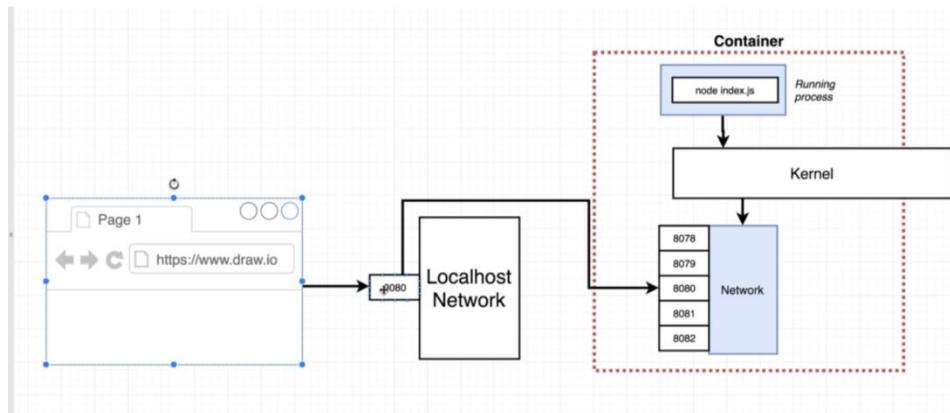
Listing on port 8080
```



But localhost:8080 is not running.

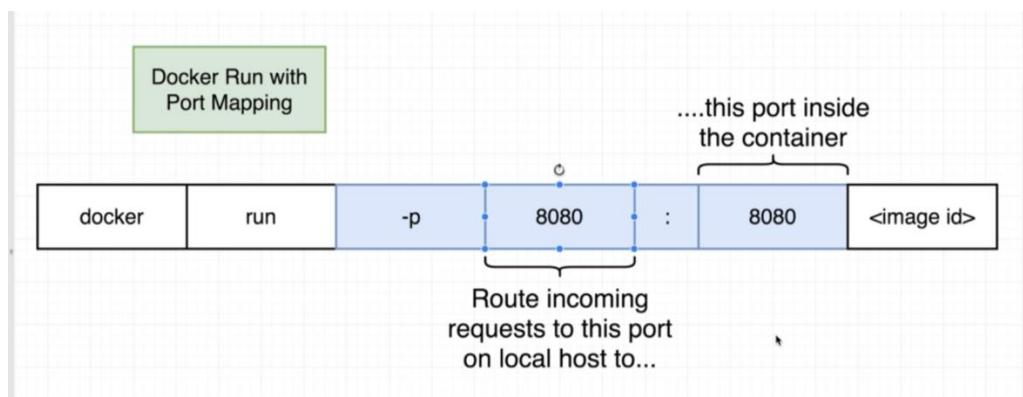
Container Port Mapping

Why we cannot be able to access the API?



Because container having an own isolated set of network ports. When we access the network in our machine, network traffic is not routed to container network port by default. We need to explicitly specify the mapping between machine port to container network port.

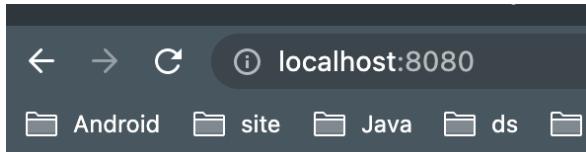
Port Mapping Command



Port on localhost and port inside the container does not have to be the same. It can be different. If we are changing the port inside the container, we need to make sure our app listen to the port.

Running the Command

```
tail -f /root/.npm/_logs/2022-05-05T07_55_20_882Z-debug-v.log
→ 01_simpleweb git:(main) ✘ docker run -p 8080:8080 mnaveensmn/simpleweb
> start
> node index.js
Listing on port 8080
```



Hi There

Specifying the Working Directory

Opening the Shell Inside Container

```
→ 01_simpleweb git:(main) ✘ docker run -it mnaveensmn/simpleweb sh
/usr/app # ls
Dockerfile      node_modules      package.json
index.js        package-lock.json
/usr/app # └
```

If we don't specify the working directory, all the files will be pasted into root directory of the container like below,

```
/ # ls
Dockerfile      mnt          sbin
bin            node_modules    srv
dev            opt           sys
etc            package-lock.json
home           package.json   usr
index.js       proc          var
lib             root
media          run
/ # └
```

Since specified the working directory, all the project files are pasted in /usr/app folder inside the container.

Specifying the Working Directory in Dockerfile

```
simpleweb > 🛠 Dockerfile > ...
1  #Specify a base image
2  FROM node:alpine
3
4  #Work Directory
5  WORKDIR /usr/app
6
7  #Install some dependencies
8  COPY ./ .
9  RUN npm install
10
11 #Default command
12 CMD ["npm","start"]
```

```

/usr/srv/mnaveen/kullen-1/123$ docker ps
→ DOCKER_KUBE git:(main) ✘ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS   NAMES
836d19f7a5c1      mnaveensmn/simpleweb "docker-entrypoint.s..."
22 minutes ago     Up 22 minutes          vigorous_wilson
→ DOCKER_KUBE git:(main) ✘ docker exec -it 836d19f7a5c1 sh
/usr/app # ls
Dockerfile          node_modules       package.json
index.js            package-lock.json
/usr/app # cd ..
/usr # s
sh: s: not found
/usr # ls
app    bin    lib    local   sbin   share
/usr # cd ..
/ # ls
bin    home   mnt   root    srv    usr
dev    lib     opt   run     sys    var
etc   media   proc   sbin   tmp
/ # 

```

Unnecessary Rebuilds

What happens we change the project files?

- If we are modifying the project file while container running, our changes won't reflect.
- To make the changes reflect, we need to rebuild the image.
- Example, if changed one of the project files, then we are rebuilding the image, docker also executes instructions which does not required for the changes to be reflected along with required instruction.
- We need to find a way to solve this problem.

Controlling unnecessary rebuild

```

01_simpleweb > Dockerfile > ...
1  #Specify a base image
2  FROM node:alpine
3
4  #Work Directory
5  WORKDIR /usr/app
6
7  #Install some dependencies
8  COPY ./package.json ./ ←
9  RUN npm install
10 COPY ./ ./
11
12 #Default command
13 CMD ["npm", "start"]

```

By specifying the COPY command for package.json, we will be running the npm install only when above instruction files get changed, in other word, package.json get changed. "npm install" won't get executed if we changed any other files.

After doing the changes, when we want to rebuild the image, as highlighted in the image, docker using it from the cache.

```
ulnerabilities and learn how to fix them
→ 01_simpleweb git:(main) ✘ docker build .
[+] Building 2.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 37B                      0.0s
=> [internal] load .dockerignore                         0.0s
=> => transferring context: 2B                          0.0s
=> [internal] load metadata for docker.io/library/node 2.0s
=> [1/5] FROM docker.io/library/node:alpine@sha256:250  0.0s
=> [internal] load build context                       0.0s
=> => transferring context: 21.16kB                   0.0s
=> CACHED [2/5] WORKDIR /usr/app                     0.0s
=> CACHED [3/5] COPY ./package.json ./                 0.0s
=> CACHED [4/5] RUN npm install                      0.0s
=> [5/5] COPY ./ ./                                 0.1s
=> exporting to image                                0.1s
=> => exporting layers                             0.1s
=> => writing image sha256:f786ceece9e0c5a64e0ec423db8 0.0s
```

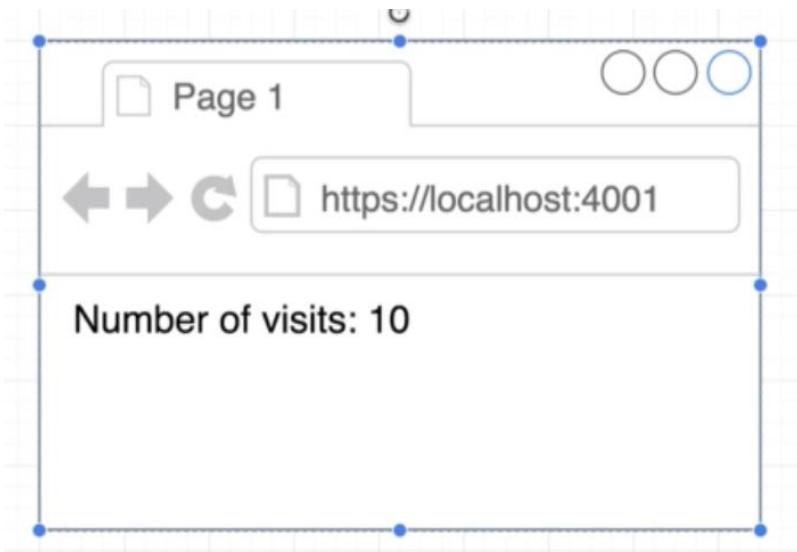
```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ 01_simpleweb git:(main) ✘ █
```

It's best practice to segment out the COPY operation.

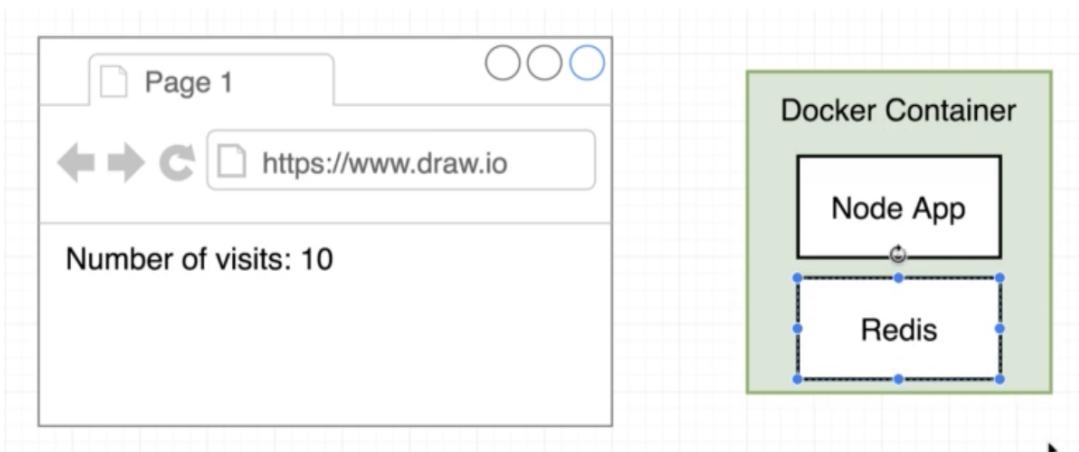
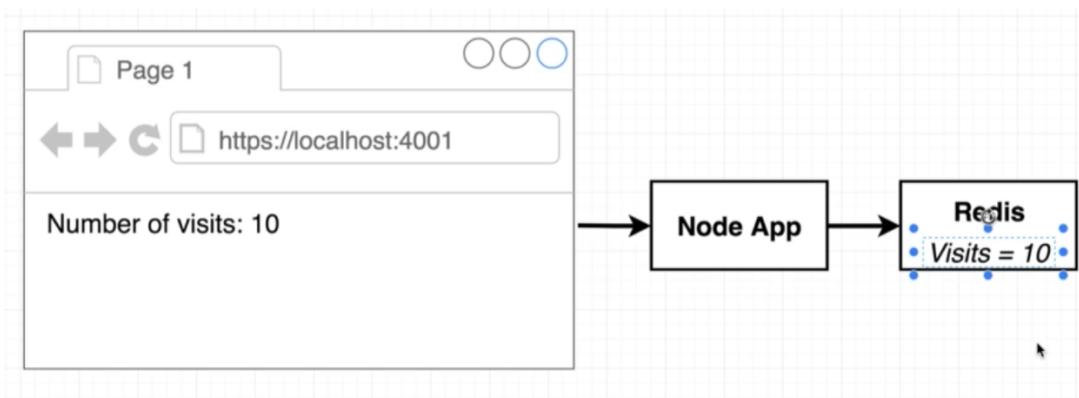
Docker Compose with Multiple Local Container

App Overview

Overview



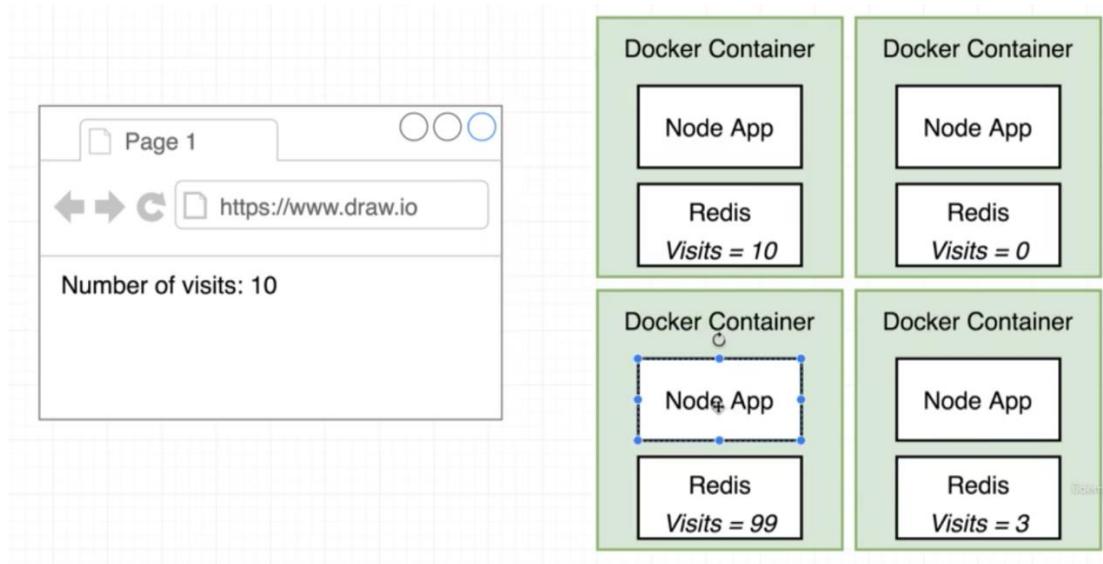
Architecture



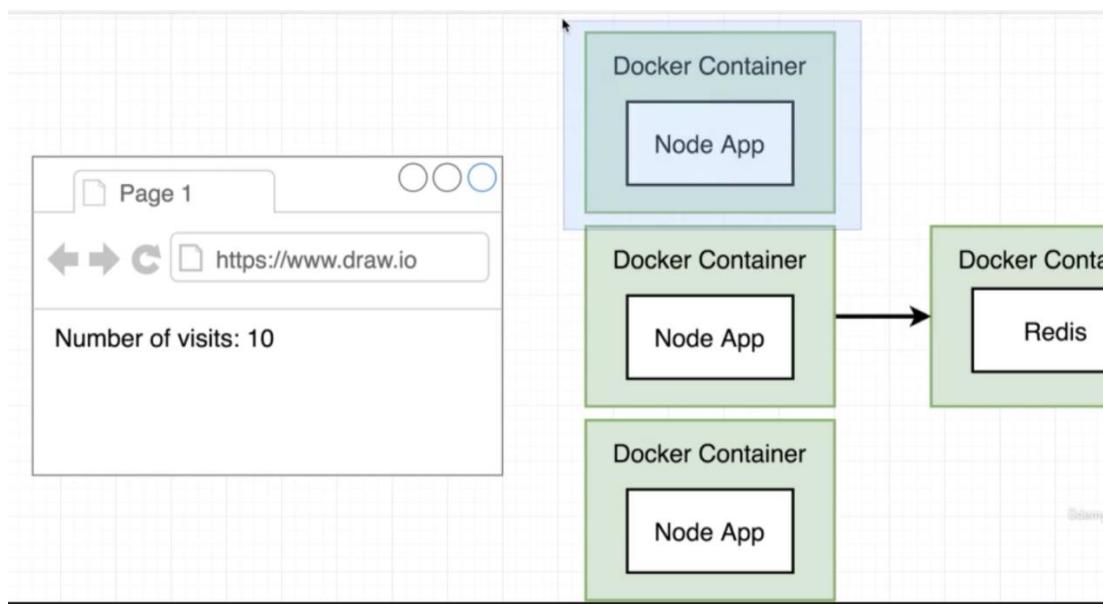
We can keep the Node App and Redis in single container.

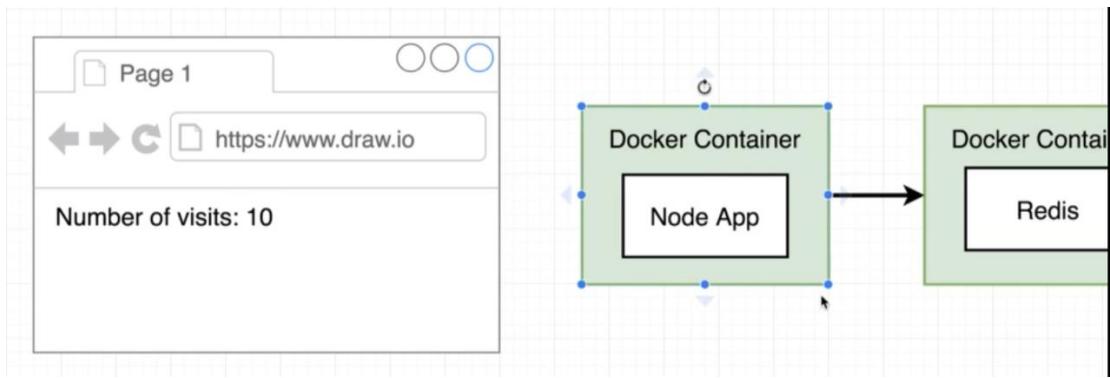
Problem with single container approach

When traffic of our webpage increase, we will be creating multiple docker container instances. So, each docker container having its own redis server and disconnected from each other. Because of this, each container having its own visit count instead of having a cumulative one.



Separate container for redis





Now, we are having a single docker container for redis and container for node app is connected to it. When the network traffic increase, we will be creating instances of only node app container and still all the instance is connected single redis container.

App Server Started Code

Code

```

✓ DOCKER_KUBE          02_user_visits > js index.js > ...
> └─ 00_redis_image
> └─ 01_simpleweb
└─ 02_user_vi... ●
  ├─ index.js   U
  └─ package.js... U
  └─ .gitignore

```

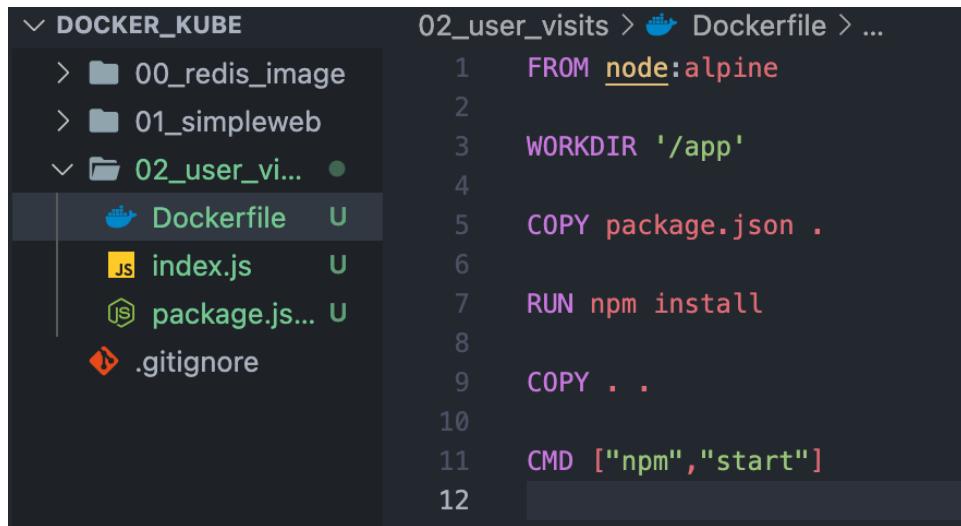
```

1  const express = require("express");
2  const redis = require("redis");
3
4  const app = express();
5  const client = redis.createClient();
6  client.set("visits", 0);
7
8  app.get("/", (req, res) => {
9    client.get("visits", (err, visits) => {
10      res.send("Number of visits is " + visits);
11      client.set("visits", parseInt(visits) + 1);
12    });
13  });
14
15  app.listen(8081, () => {
16    console.log("Listening on port 8081");
17  });

```

Assembling a Docker File

Creating a docker file



The screenshot shows a file explorer interface with a tree view on the left and a code editor on the right. The tree view shows a folder structure under 'DOCKER_KUBE': '00_redis_image', '01_simpleweb', and '02_user_visits'. '02_user_visits' is expanded, showing files: 'Dockerfile', 'index.js', 'package.json', and '.gitignore'. The 'Dockerfile' is selected and open in the code editor.

```
02_user_visits > Dockerfile > ...
1  FROM node:alpine
2
3  WORKDIR '/app'
4
5  COPY package.json .
6
7  RUN npm install
8
9  COPY . .
10
11 CMD ["npm","start"]
12
```

Building an image

```
→ 02_user_visits git:(main) ✘ docker build -t mnaveensmn/visits:latest .
[+] Building 12.8s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 144B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 739B
=> CACHED [1/5] FROM docker.io/library/node:alpine@sha256:250e9a093b861c330be2f4d1d22 0.0s
=> [2/5] WORKDIR /app
=> [3/5] COPY package.json .
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:d52bc2010e1451d67c31e9d9c9390b7be8f23cc3ed2dfb9d9c25aa79b0 0.0s
=> => naming to docker.io/mnaveensmn/visits:latest 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Introducing a Docker Compose

Running a docker image

```
ix them
→ 02_user_visits git:(main) ✘ docker run mnaveensmn/visits
> start
> node index.js

Listening on port 8081
node:events:505
    throw er; // Unhandled 'error' event
  ^

Error: connect ECONNREFUSED 127.0.0.1:6379
  at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1158:16)
Emitted 'error' event on RedisClient instance at:
  at RedisClient.on_error (/app/node_modules/redis/index.js:406:14)
  at Socket.<anonymous> (/app/node_modules/redis/index.js:279:14)
  at Socket.emit (node:events:527:28)
  at emitErrorNT (node:internal/streams/destroy:164:8)
  at emitErrorCloseNT (node:internal/streams/destroy:129:3)
  at processTicksAndRejections (node:internal/process/task_queues:83:21) {
  errno: -111,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 6379
}

Node.js v17.6.0
```

Error due to there is no redis running in the container.

Starting the redis server

```
→ DOCKER_KUBE git:(main) ✘ docker run redis
1:C 07 Mar 2022 04:03:38.487 # o000o000o000o Redis is starting o000o000o000o
1:C 07 Mar 2022 04:03:38.487 # Redis version=6.2.6, bits=64, commit=00000000, modified
=0, pid=1, just started
1:C 07 Mar 2022 04:03:38.487 # Warning: no config file specified, using the default co
nfig. In order to specify a config file use redis-server /path/to/redis.conf
1:M 07 Mar 2022 04:03:38.488 * monotonic clock: POSIX clock_gettime
1:M 07 Mar 2022 04:03:38.489 * Running mode=standalone, port=6379.
1:M 07 Mar 2022 04:03:38.489 # Server initialized
1:M 07 Mar 2022 04:03:38.490 * Ready to accept connections
[]
```

We just started the redis server.

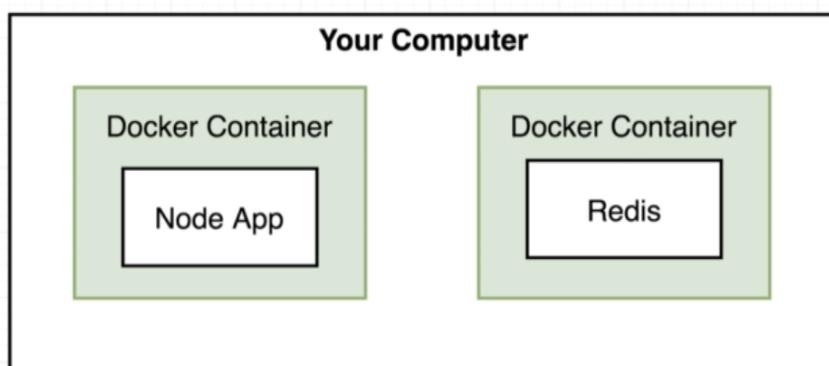
```
Node.js v17.6.0
→ 02_user_visits git:(main) ✘ docker run mnaveensmn/visits
> start
> node index.js

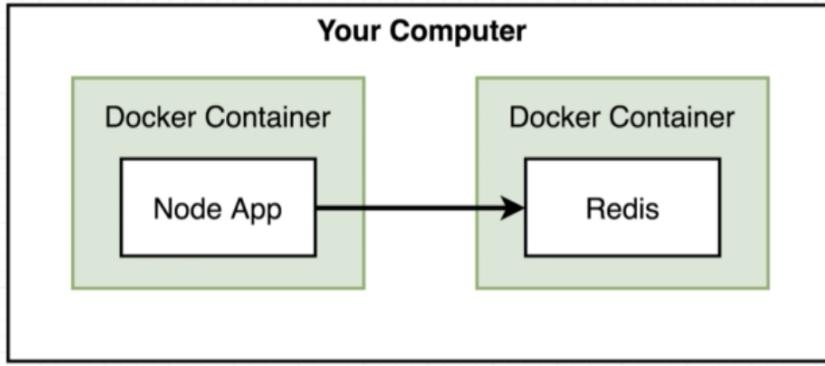
Listening on port 8081
node:events:505
    throw er; // Unhandled 'error' event
    ^
Error: connect ECONNREFUSED 127.0.0.1:6379
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1158:16)
Emitted 'error' event on RedisClient instance at:
    at RedisClient.on_error (/app/node_modules/redis/index.js:406:14)
    at Socket.<anonymous> (/app/node_modules/redis/index.js:279:14)
    at Socket.emit (node:events:527:28)
    at emitErrorNT (node:internalstreams/destroy:164:8)
    at emitErrorCloseNT (node:internalstreams/destroy:129:3)
    at processTicksAndRejections (node:internal/process/task_queues:83:21) {
  errno: -111,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '127.0.0.1',
  port: 6379
}
Node.js v17.6.0
```

Even though we have a redis server running our app still fails.

Analyzing the issue

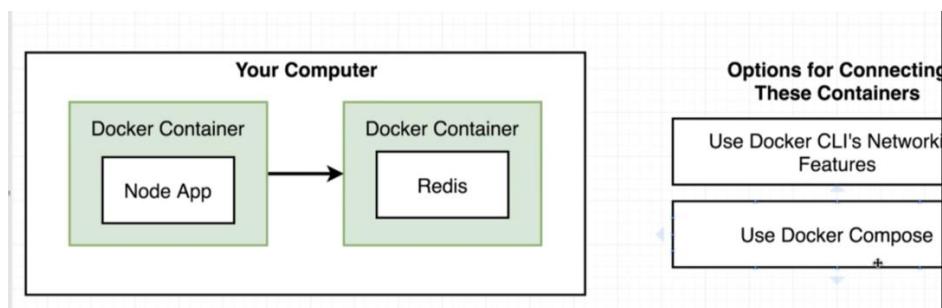
Node and redis server are running in separate container and not connected each other.





We need to find a way to establish the networking infrastructure between these two containers.

Options for establishing the network infrastructure



We have two options to look at.

Establishing the network infrastructure using Docker CLI's network feature required to do a hand full of job every time when start the docker cli.

Docker Compose

Docker Compose

Separate CLI that gets installed along with Docker

Used to start up multiple Docker containers at the same time

Automates some of the long-winded arguments we were passing to 'docker run'

```

→ 02_user_visits git:(main) ✘ docker-compose
Usage: docker compose [OPTIONS] COMMAND
Docker Compose
Options:
  --ansi string           Control when to print ANSI control
                          characters ("never"|"always"|"auto")
                          (default "auto")
  --compatibility
  --env-file string       Run compose in backward compatibility mode
  -f, --file stringArray  Specify an alternate environment file.
  --profile stringArray   Compose configuration files
  --project-directory string  Specify a profile to enable
  --project-name string    Specify an alternate working directory
                           (default: the path of the Compose file)
  -p, --project-name string Project name

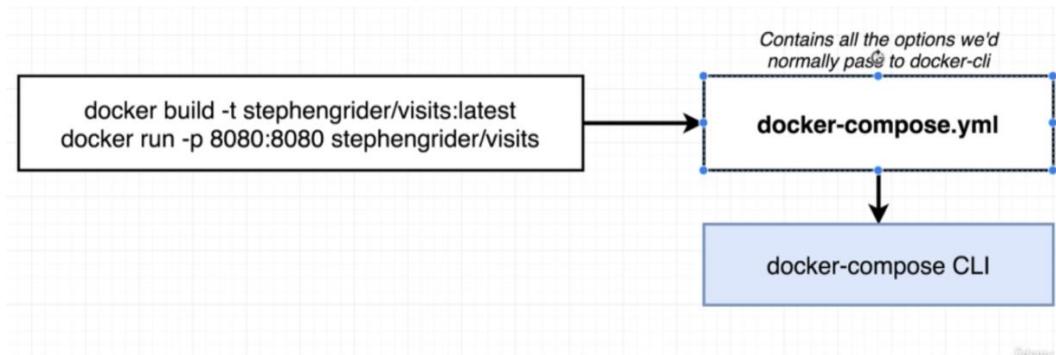
Commands:
  build      Build or rebuild services
  convert    Converts the compose file to platform's canonical format
  cp         Copy files/folders between a service container and the local filesystem
  create     Creates containers for a service.
  down       Stop and remove containers, networks
  events     Receive real time events from containers.
  exec       Execute a command in a running container.
  images    List images used by the created containers
  kill       Force stop service containers.
  logs      View output from containers
  ls        List running compose projects
  pause     Pause services
  port      Print the public port for a port binding.
  ps        List containers

```

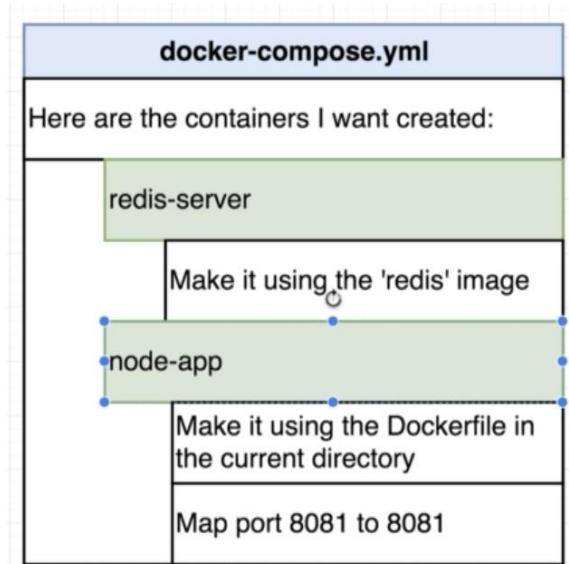
Docker compose avoids running the repetitive commands.

Docker Compose File

What is YAML File



Creating a YAML File



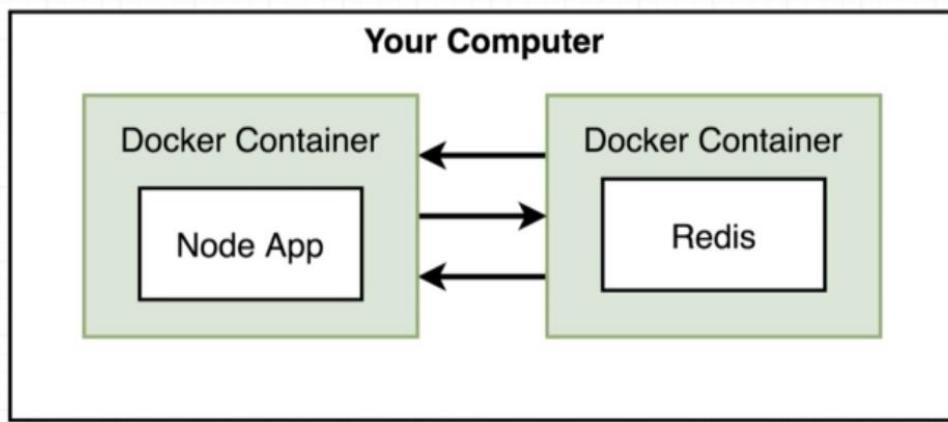
The screenshot shows a code editor interface with two tabs: 'Dockerfile U' and 'docker-compose.yaml U'. The left sidebar shows a project structure under 'DOCKER_KUBE' with folders '00_redis_image', '01_simpleweb', and '02_user_v...'. Inside '02_user_v...', there are files: 'Dockerfile', 'index.js', 'package.json', and '.gitignore'. The 'Dockerfile' tab is active, displaying the following content:

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

In the first service, we are using the image from docker hub. In the second service, we are building the image from Dockerfile.

Networking with Docker Compose

With the docker compose containers are free to share data with each other.



Establishing connection

Now, we have a docker container for Node App and Redis running, it's a time to establish the connection between Node App and Redis.

To establishing the connection between redis server and node app by specifying the redis service name (from the docker compose yaml file) and port while creating the redis client at the node app.

```

const express = require("express");
const redis = require("redis");

const app = express();
const client = redis.createClient({
  host: "redis-server",
  port: 6379,
});
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);
  });
});

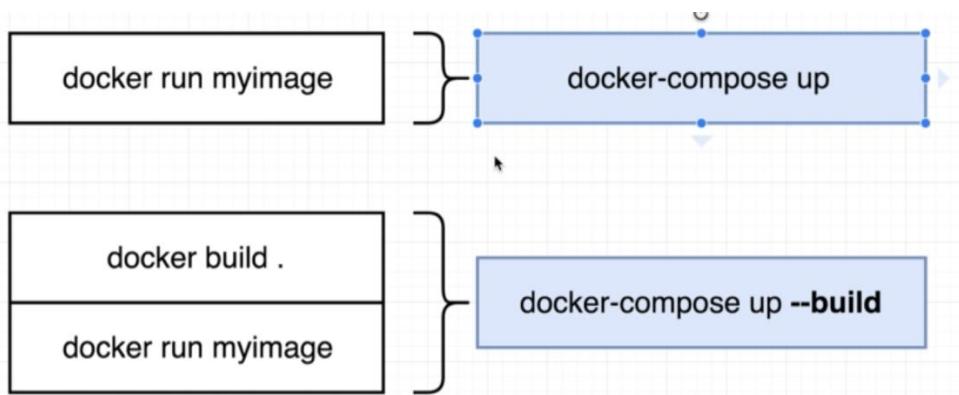
app.listen(8081, () => {
  console.log("Listening on port 8081");
});

```

Docker Compose Commands

One of the purposes of docker compose is to make easier to run the docker commands.

Commands

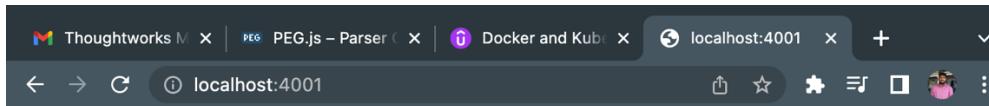


Execution

```
docker compose package.json
+ 02_user_visits git:(main) docker-compose up
[+] Building 4.2s (11/11) FINISHED
=> [internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 139B                      0.0s
=> [internal] load .dockerignore                         0.0s
=> => transferring context: 2B                          0.0s
=> [internal] load metadata for docker.io/library/node 4.0s
=> [auth] library/node:pull token for registry-1.docke 0.0s
=> [internal] load build context                       0.0s
=> => transferring context: 942B                      0.0s
=> [1/5] FROM docker.io/library/node:alpine@sha256:250 0.0s
=> CACHED [2/5] WORKDIR /app                           0.0s
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install                         0.0s
=> [5/5] COPY . .
=> exporting to image                     0.0s
=> => exporting layers                         0.0s
=> => writing image sha256:5fb572190352a673b3f67911f75 0.0s
=> => naming to docker.io/02_user_visits_node- 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 3/3
:: Network 02_user_visits_default     Created      0.0s
:: Container 02_user_visits-node-app-1 Created      0.1s
:: Container 02_user_visits-redis-server-1 Created      0.1s
Attaching to 02_user_visits-node-app-1, 02_user_visits-redis-server-1
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 07:30:15.246
# o000o000o000 Redis is starting o000o000o000
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 07:30:15.246
# Redis version=6.2.6, bits=64, commit=00000000, modified=0, p
id=1, just started
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 07:30:15.246
# Warning: no config file specified, using the default config.
# In order to specify a config file use redis-server /path/to/r
edis.conf
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 07:30:15.246
```

By using the docker compose, now two containers able to communicate.



Stopping docker compose containers

Using the docker compose we are working with multiple containers at the same time. So, we can start or stop all the containers at once.

Typical docker commands to run and stop the containers.

```
→ visits git:(066-stats) docker run -d redis
ba1220ea9edc743504aeaa5f198ae40fd7651558ab23b1746a45b1264f4c9acf
→ visits git:(066-stats) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
ba1220ea9edc        redis              "docker-entrypoint.s..."   13 seconds ago    Up 12 seconds     quizzical_fermat
→ visits git:(066-stats) docker stop ba1220ea9edc
ba1220ea9edc
→ visits git:(066-stats)
```

API 1.25+	
--cpus	Number of CPUs
--cpuset-cpus	CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems	MEMs in which to allow execution (0-3, 0,1)
--detach , -d	Run container in background and print container ID
--detach-keys	Override the key sequence for detaching a container
--device	Add a host device to the container
--device-cgroup-rule	Add a rule to the cgroup allowed devices list

Stopping the containers

Launch in background

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

Stop Containers

```
docker-compose down
```

```
/Users/rajanreddi/Downloads/PyCharm.app/Contents/plugins/docker/lib/python3.6/site-packages/docker/api/client.py:62: UserWarning: No matches found.  Code: None
  warnings.warn("no configuration file provided: %s" % code)
```

```
→ DOCKER_KUBE git:(main) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
e9fc15ac9e1e        02_user_visits_node-app   "docker-entrypoint.s..."   22 minutes ago   Up 22 minutes
es      0.0.0:4001->8081/tcp      02_user_visits-node-app-1
ea5396be2e61        redis              "docker-entrypoint.s..."   22 minutes ago   Up 22 minutes
es      6379/tcp          02_user_visits-redis-server-1
→ DOCKER_KUBE git:(main) docker-compose down
no configuration file provided: not found
→ DOCKER_KUBE git:(main) cd 02_user_visits
→ 02_user_visits git:(main) docker-compose down
[+] Running 3/3
  #: Container 02_user_visits-redis-server-1 Removed
  #: Container 02_user_visits-node-app-1      Rem...
  #: Network 02_user_visits_default           Removed
  0.2s
  0.7s
  0.1s
→ 02_user_visits git:(main) docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
→ 02_user_visits git:(main)
```

Container Maintenance with Docker Compose

What if container crash

Making the server to crash for learning purpose

Making the changes in the code.

```
user_visits > js index.js > app.get("/") callback
  const express = require("express");
  const redis = require("redis");
  const process = require("process");
  

---


  const app = express();
  const client = redis.createClient({
    host: "redis-server",
    port: 6379,
  });
  client.set("visits", 0);

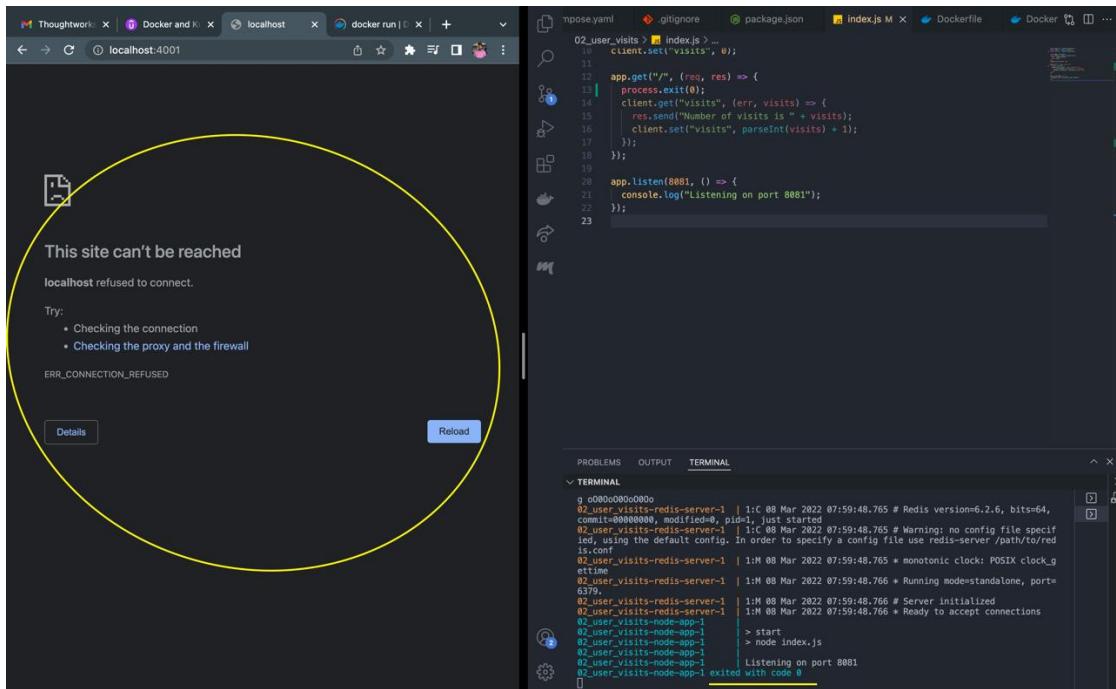
  app.get("/", (req, res) => {
    process.exit(0);
    client.get("visits", (err, visits) =>
      res.send("Number of visits is " + visits);
    client.set("visits", parseInt(visits));
  });
}

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

Building the image and starting the container

```
→ 02_user_visits git:(main) ✘ docker-compose up --build
[+] Building 4.0s (11/11) FINISHED
  => [internal] load build definition from Dockerfile          0.0s
  => => transferring dockerfile: 31B                          0.0s
  => [internal] load .dockerignore                           0.0s
  => => transferring context: 2B                            0.0s
  => [internal] load metadata for docker.io/library/alpine   3.9s
  => [auth] library/node:pull token for registry-1.docker.io 0.0s
  => [internal] load build context                         0.0s
  => => transferring context: 637B                         0.0s
  => [1/5] FROM docker.io/library/node:alpine@sha256:250e9a093b861c330be2f4d1d224712d4e4 0.0s
  => CACHED [2/5] WORKDIR /app                            0.0s
  => CACHED [3/5] COPY package.json .                      0.0s
  => CACHED [4/5] RUN npm install                         0.0s
  => [5/5] COPY . .                                     0.0s
  => exporting to image                                  0.0s
  => => exporting layers                                0.0s
```

Application crashes when access the link



As we can see, only Redis container is running. Node app container got crashed.

```
/usr/local/bin/docker ps
→ DOCKER_KUBE git:(main) ✘ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
00a3e3afdf31      redis              "docker-entrypoint.s..."   5 minutes ago    Up 5 minutes     6379/tcp      02
_user_visits-redis-server-1
→ DOCKER_KUBE git:(main) ✘
```

Automatic Container Restart

How to auto restart the container when it crashes.

Status codes

Docker can decide whether to restart or not based on the status from exit status codes.



Restart Policies

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

In yaml file "no" interpreted as a false. So, when specifying the "no" as a policy, we need to have a double or single quote.

Updating the docker-compose.yaml

We are specifying the restart policies for specific containers.

```
02_user_visits > ⚡ docker-compose.yaml
 1  version: "3"
 2  services:
 3    redis-server:
 4      image: "redis"
 5    node-app:
 6      restart: always
 7      build: .
 8      ports:
 9        - "4001:8081"
10
```

Running the container

```
↳ Ready to exec, by:
→ 02_user_visits git:(main) ✘ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
→ 02_user_visits git:(main) ✘ docker-compose up
[+] Running 3/3
  ● Network 02_user_visits_default          Created          0.0s
  ● Container 02_user_visits-redis-server-1 Created          0.1s
  ● Container 02_user_visits-node-app-1     Created          0.0s
Attaching to 02_user_visits-node-app-1, 02_user_visits-redis-server-1
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:22:58.101 # o000o000o00
0 Redis is starting 0000o000o000
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:22:58.101 # Redis versi
on=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:22:58.101 # Warning: no
 config file specified, using the default config. In order to specify a con
fig file use redis-server /path/to/redis.conf
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:22:58.102 * monotonic c
lock: POSIX clock_gettime
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:22:58.103 * Running mod
e=standalone, port=6379.
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:22:58.103 # Server init
ialized
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:22:58.103 * Ready to ac
cept connections
```

As we can see in the run log that, every time application exited with status code 0, it's start the container once again.

```

02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:22:58.103 * Ready
cept connections
02_user_visits-node-app-1    | > start
02_user_visits-node-app-1    | > node index.js
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | Listening on port 8081
02_user_visits-node-app-1 exited with code 0
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | > start
02_user_visits-node-app-1    | > node index.js
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | Listening on port 8081
02_user_visits-node-app-1 exited with code 0
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | > start
02_user_visits-node-app-1    | > node index.js
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | Listening on port 8081
02_user_visits-node-app-1 exited with code 0
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | > start
02_user_visits-node-app-1    | > node index.js
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | Listening on port 8081

```

When docker restarting the container, it is using the already created container.

Container status with Docker

Starting the container

```

→ 02_user_visits git:(main) docker-compose up
[+] Running 3/3
  ● Network 02_user_visits_default      Created          0.0s
  ● Container 02_user_visits-node-app-1 Created          0.0s
  ● Container 02_user_visits-redis-server-1 Created          0.1s
Attaching to 02_user_visits-node-app-1, 02_user_visits-redis-server-1
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:43:47.807 # 000000000000
0 Redis is starting 000000000000
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:43:47.807 # Redis versi
on=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
02_user_visits-redis-server-1 | 1:C 08 Mar 2022 08:43:47.807 # Warning: no
 config file specified, using the default config. In order to specify a com
fig file use redis-server /path/to/redis.conf
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:43:47.807 * monotonic c
lock: POSIX clock_gettime
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:43:47.808 * Running mod
e=standalone, port=6379.
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:43:47.808 # Server init
ialized
02_user_visits-redis-server-1 | 1:M 08 Mar 2022 08:43:47.808 * Ready to ac
cept connections
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | > start
02_user_visits-node-app-1    | > node index.js
02_user_visits-node-app-1    |
02_user_visits-node-app-1    | Listening on port 8081

```

Container status

NAME	COMMAND	SERVICE	STATUS
02_user_visits-node-app-1	"docker-entrypoint.s..."	node-app	running
02_user_visits-redis-server-1	"docker-entrypoint.s..."	redis-server	running

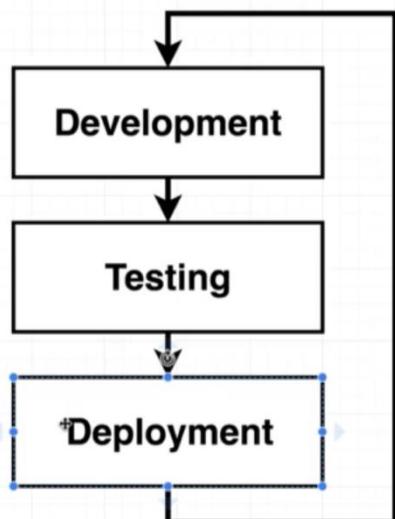
Docker-compose ps command require a yaml file. So, we cannot run this command outside the yaml file's directory.

```
6379/tcp
→ 02_user_visits git:(main) cd ..
→ DOCKER_KUBE git:(main) docker-compose ps
no configuration file provided: not found
→ DOCKER_KUBE git:(main) █
```

Creating a Production Grade Workflow

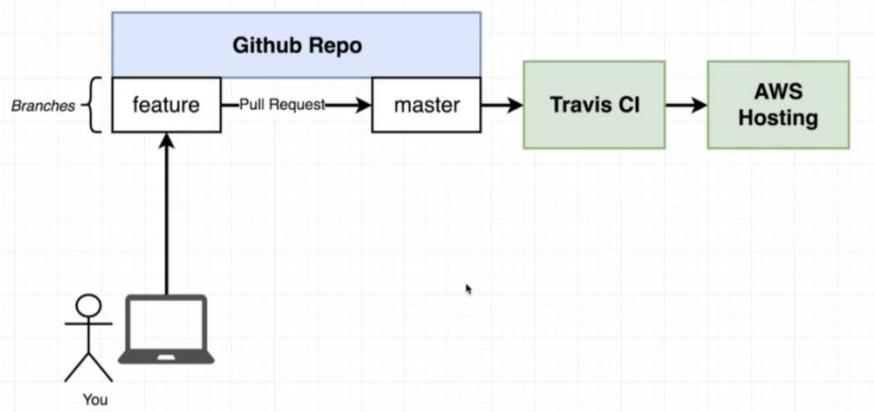
Workflow

Development Workflow



As our software expands with multiple features and functionalities, we have to do multiple deployments.

Flow Specifics



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

Project Creation

735902

Creating react app

```
/Users/naveen.kumar1/Workspace/no matches found, load.
→ DOCKER_KUBE git:(main) npx create-react-app 03_frontend-react-app

Creating a new React app in /Users/naveen.kumar1/Workspace/DOCKER_KUBE/03_frontend-react-app.

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

  ):: idealTree:03_frontend-react-app: sill idealTr
  ):: idealTree:03_frontend-react-app: sill idealTr
  ):: idealTree:03_frontend-react-app: sill idealTr
  ):: idealTree:babel-jest: timing idealTree:node_modules/babel-jest Complete
```

```
Success! Created 03_frontend-react-app at /Users/naveen.kumar1/Workspace/DOCKER_KUBE/03_frontend-react-app
Inside that directory, you can run several commands:
```

```
npm start
  Starts the development server.

npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!
```

We suggest that you begin by typing:

```
cd 03_frontend-react-app
npm start
```

Happy hacking!

Necessary Commands

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

```
PASS  src/App.test.js
  ✓ renders learn react link (30 ms)

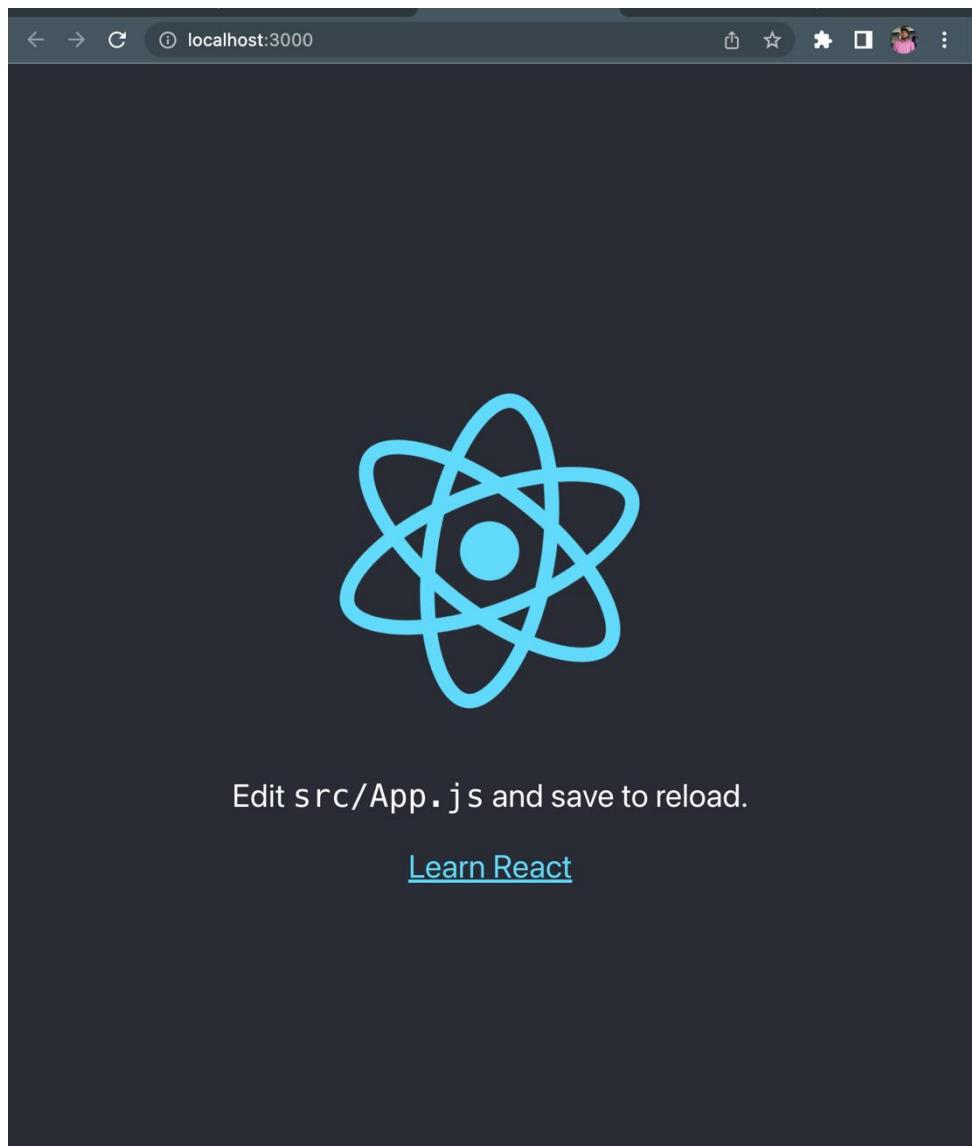
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        5.139 s
Ran all test suites related to changed files.

Watch Usage
> Press a to run all tests.
> Press f to run only failed tests.
> Press q to quit watch mode.
> Press p to filter by a filename regex pattern.
> Press t to filter by a test name regex pattern.
> Press Enter to trigger a test run.

→ 03_frontend-react-app git:(main) ✘ npm run build
> 03_frontend-react-app@0.1.0 build
> react-scripts build

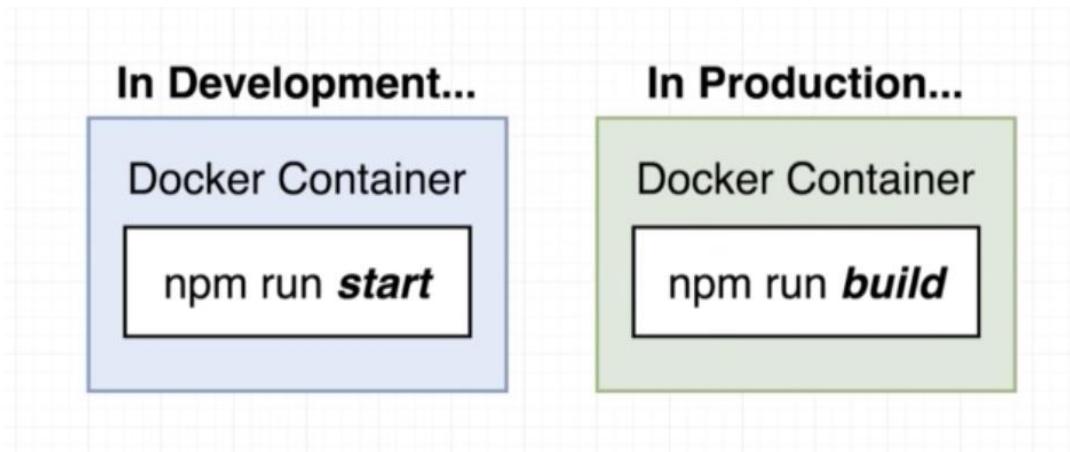
Creating an optimized production build...
Compiled successfully.
```

Npm start



Creating a Dev Dockerfile

Type of dockerfile



We can have a dockerfile for development and production.

Creating a dockerfile

Now, we are proceeding with creating a dockerfile for development environment. We named the file as "Dockerfile.dev" instead of "Dockerfile". It's a custom file name. With the custom docker file name, we need to specify the file name explicitly in docker build command.

A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project structure with folders like "00_redis_image", "01_simpleweb", "02_user_visits" (which is selected), and "03_frontend-react-app". Inside "03_frontend-react-app", there are sub-folders "build", "node_modules", "public", "src", ".gitignore", and files "Dockerfile.dev", "package-lock.json", "package.json", "README.md", and another ".gitignore". The "Dockerfile.dev" file is open in the main editor area, displaying the following Dockerfile content:

```
FROM node:16-alpine
WORKDIR '/app'
COPY package.json .
RUN npm install
COPY . .
CMD ["npm", "run", "start"]
```

Building dockerfile

```

create mode 100644 03_Frontend/react-app/package-lock.json
→ 03_frontend-react-app git:(main) ✘ docker build -f Dockerfile.dev .
Sending build context to Docker daemon 289.7MB
Step 1/6 : FROM node:16-alpine
16-alpine: Pulling from library/node
59bf1c3509f3: Already exists
683dd8c3cc08: Pull complete
ae5b2724f19b: Pull complete
39190df3f477: Pull complete
Digest: sha256:2c6c59cf4d34d4f937ddfcf33bab9d8bbad8658d1b9de7b97622566a52167f2b
Status: Downloaded newer image for node:16-alpine
--> 0e1547c0f4a4
Step 2/6 : WORKDIR '/app'
--> Running in 07b57c81335c
Removing intermediate container 07b57c81335c
--> 7a3f59dd43d4
Step 3/6 : COPY package.json .
--> f3682d8b6bba
Step 4/6 : RUN npm install
--> Running in d4b85933c1c8
npm WARN deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.
npm WARN deprecated source-map-resolve@0.6.0: See https://github.com/lydell/source-map-resolve
#deprecated

added 1237 packages, and audited 1238 packages in 2m

169 packages are looking for funding
  run `npm fund` for details

6 moderate severity vulnerabilities

```

When we build for second time, it ran really fast.

```

→ 03_frontend-react-app git:(main) ✘ docker build -t mnaveensmn/react-app:latest -f Dockerfile.dev .
[+] Building 2.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile.dev          0.0s
=> => transferring dockerfile: 40B                                0.0s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                0.0s
=> [internal] load metadata for docker.io/library/node:16-alpine 0.0s
=> [1/5] FROM docker.io/library/node:16-alpine                0.0s
=> [internal] load build context                           2.2s
=> => transferring context: 2.85MB                         2.1s
=> CACHED [2/5] WORKDIR /app                            0.0s
=> CACHED [3/5] COPY package.json .                      0.0s
=> CACHED [4/5] RUN npm install                         0.0s
=> CACHED [5/5] COPY . .                                0.0s
=> exporting to image                                 0.0s
=> => exporting layers                                0.0s
=> => writing image sha256:9280fae1d175b76ea685bedba0a8b73e88ad92cc6b57af61f5e 0.0s
=> => naming to docker.io/mnaveensmn/react-app:latest      0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

```

Starting the Container

Command

```
docker run -it -p 3000:3000 IMAGE_ID
```

```

Compiled successfully!

You can now view 03_frontend-react-app in the browser.

  Local:          http://localhost:3000
  On Your Network:  http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

assets by status 9.52 KiB [cached] 2 assets
assets by status 1.48 MiB [emitted]
  assets by chunk 1.48 MiB (name: main)
    asset static/js/bundle.js 1.48 MiB [emitted] (name: main) 1 related asset
    asset main.9fe084b7c6ee7d8087a3.hot-update.js 373 bytes [emitted] [immutable] [hmr]
  ] (name: main) 1 related asset
  assets by path *.json 699 bytes
    asset asset-manifest.json 671 bytes [emitted]
    asset main.9fe084b7c6ee7d8087a3.hot-update.json 28 bytes [emitted] [immutable] [hm
r]
  asset index.html 1.67 KiB [emitted]
Entrypoint main 1.48 MiB (1.5 MiB) = static/js/bundle.js 1.48 MiB main.9fe084b7c6ee7d8
087a3.hot-update.js 373 bytes 3 auxiliary assets
  cached modules 1.37 MiB [cached] 107 modules
  runtime modules 31.4 KiB 15 modules
  webpack 5.70.0 compiled successfully in 213 ms

```

We made a following changes to App.js file. But our changes not reflected in UI as container using the file system snapshot when the time image was built.

```

src/App.js
import logo from "./logo.svg";
import "./App.css";

Complexity is 4 Everything is cool!
function App() {
  return (
    <div className="App">
      <h1>Hello Docker !!</h1>
    </div>
  );
}

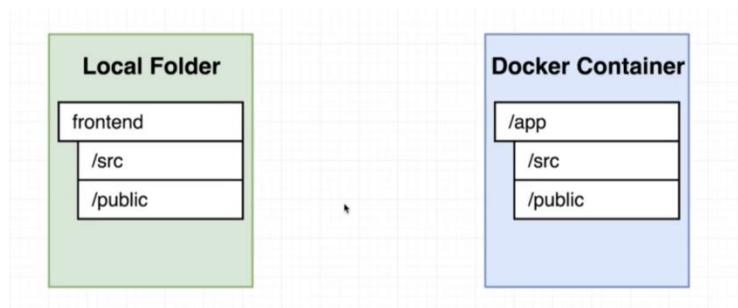
export default App;

```

How to make to the changes reflected immediately in the UI ?

Docker Volume

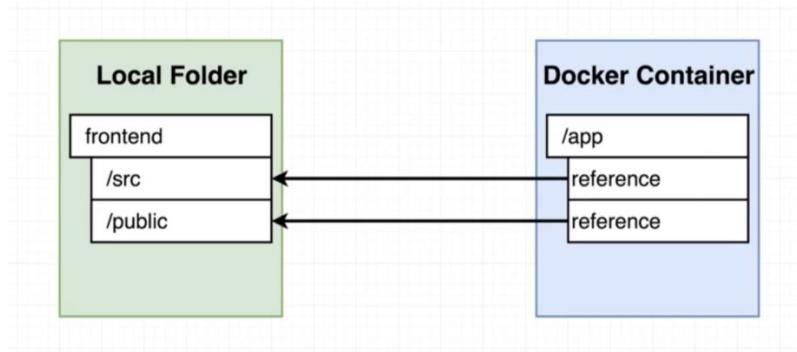
Current Scenario



In the current scenario, we are copying the file from local folder to docker container.

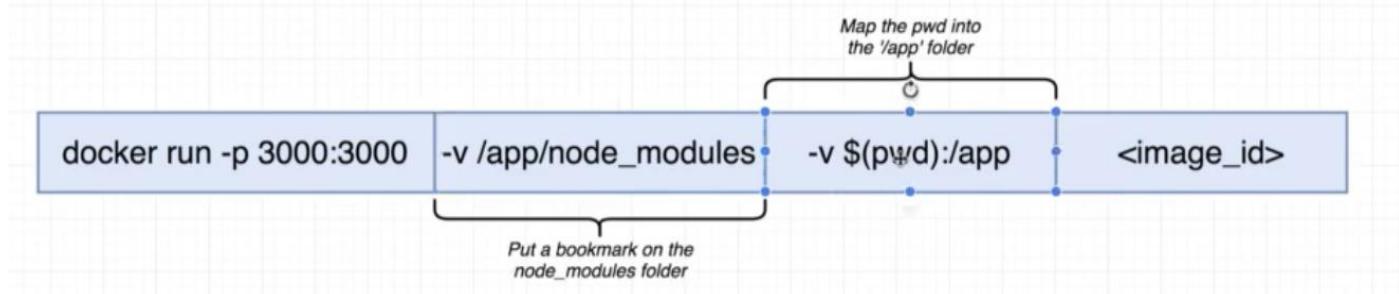
How to make the changes we make in the local folder get updated in the docker container?

- We can achieve this by using the docker volume. It's an out of the box feature in docker.
- With the docker volume, we have a placeholder in the container which has reference files in local folders.



- It is like mapping between folders insider and outside the container.

Syntax for setting up the volume

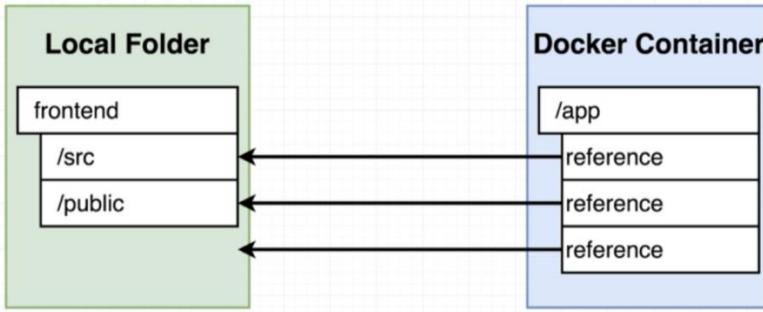


Running the commands

```
➜ frontend git:(070-flow) ✘ docker run -p 3000:3000 -v $(pwd):/app 68dabe9a308e
> frontend@0.1.0 start /app
> react-scripts start

sh: react-scripts: not found
npm ERR! file sh
npm ERR! code ELIFECYCLE
npm ERR! errno ENOENT
npm ERR! syscall spawn
npm ERR! frontend@0.1.0 start: `react-scripts start`
npm ERR! spawn ENOENT
npm ERR!
npm ERR! Failed at the frontend@0.1.0 start script.
npm ERR! This is probably not a problem with npm. There is likely additional
logging output above.
npm WARN Local package.json exists, but node_modules missing, did you mean "npm install"?
```

"-v \$(pwd):/app" command tells the docker that map the current directory to app folder inside the container.



When we initially run the docker run command there is no “node_module” folder. So, docker container pointing to empty reference. That’s the cause of the above issue.

```

See `docker run --help` .
→ 03_frontend-react-app git:(main) ✘ docker run -p 3000:3000 -v $(pwd):/app mnaveensmn/react-app

> 03_frontend-react-app@0.1.0 start
> react-scripts start

sh: react-scripts: not found
→ 03_frontend-react-app git:(main) ✘ docker run -p 3000:3000 -v /app/node_modules -v $(pwd):/app mnaveensmn/react-app

> 03_frontend-react-app@0.1.0 start
> react-scripts start

(node:26) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:26) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

Compiled successfully!

You can now view 03_frontend-react-app in the browser.

Local:          http://localhost:3000
On Your Network: http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

```

In the above image, we deleted the node_modules folder inside local folder and ran the first command. It given an error as react-scripts not found.

In the second command in the above image, we have given the “-v /app/node_modules” along with existing command. It’s telling the docker to not to look for any reference in the local folder. Because, by the time when we run image, we would already have the node_module folder inside the container.

Now, we can see that docker accepting the running changes from the local folder. This set up is called volume mounting system.

A screenshot of the VS Code interface. At the top, there are several tabs: 'Inbox - naveen', 'Thoughtworks', 'Docker and Kub...', 'React App', and a new tab. Below the tabs, the address bar shows 'localhost:3000'. The main area displays a browser window with the title 'Hellow Docker !!' and the content 'Hello Docker !!'. To the right of the browser window is the code editor showing 'App.js' with the following code:

```
1 import './App.css';
2
3 function App() {
4   return (
5     <div className="App">
6       <h1>Hello Docker !!</h1>
7     </div>
8   );
9 }
10
11 export default App;
```

Below the code editor is the 'TERMINAL' tab, which shows the output of a webpack build:

```
cached modules 1.36 MiB [cached] 105 modules
runtime modules 31.4 KiB 15 modules
./src/App.js 1.48 KiB [built] [code generated]
webpack 5.70.0 compiled successfully in 187 ms
Compiling...
Compiled successfully!
assets by status 6.95 KiB [cached] 1 asset
assets by chunk 1.48 MiB (name: main)
  asset static/js/bundle.js 1.47 MiB [emitted] (name: main) 1 related asset
    asset main.b3d25aae4bc6b5aa24d2.hot-update.js 2.82 KiB [emitted] [immutable] [hmr] (na
me: main) 1 related asset
assets by path *.json 611 bytes
  asset asset-manifest.json 583 bytes [emitted]
    asset main.b3d25aae4bc6b5aa24d2.hot-update.json 28 bytes [emitted] [immutable] [hmr]
asset index.html 1.67 KiB [emitted]
Entrypoint main 1.48 MiB (main.b3d25aae4bc6b5aa24d2.hot-update.js) = static/js/bundle.js 1.47 MiB main.b3d25aae4bc6b5aa24d2.hot-update.js 2.82 KiB [emitted] 3 auxiliary assets
cached modules 1.36 MiB [javascript] 31.4 KiB (runtime) [cached] 120 modules
runtime modules 31.4 KiB 15 modules
./src/App.js 1.48 KiB [built]
webpack 5.70.0 compiled successfully in 225 ms
Compiling...
Compiled successfully!
assets by status 1.48 MiB [cached] 2 assets
assets by path *.json 211 bytes
  asset index.html 1.67 KiB [emitted]
  asset asset-manifest.json 458 bytes [emitted]
cached modules 1.36 MiB [javascript] 31.4 KiB (runtime) [cached] 120 modules
./src/App.js 1.48 KiB [built]
webpack 5.70.0 compiled successfully in 110 ms
```

Shorthand with Docker Compose

Creating docker compose yaml file

A screenshot of the VS Code interface. On the left is the 'EXPLORER' sidebar, which shows a project structure under 'DOCKER_KUBE'. The 'src' folder contains files like 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'reportWebVitals.js', 'setupTests.js', '.gitignore', and 'docker-compose.yaml'. The 'docker-compose.yaml' file is currently selected and shows the following content:

```
1 version: "3"
2 services:
3   web:
4     build: .
5     ports:
6       - "3000:3000"
7     volumes:
8       - /app/node_modules
9       - .:/app
```

At the bottom of the screen is the 'TERMINAL' tab, which shows the output of a webpack build:

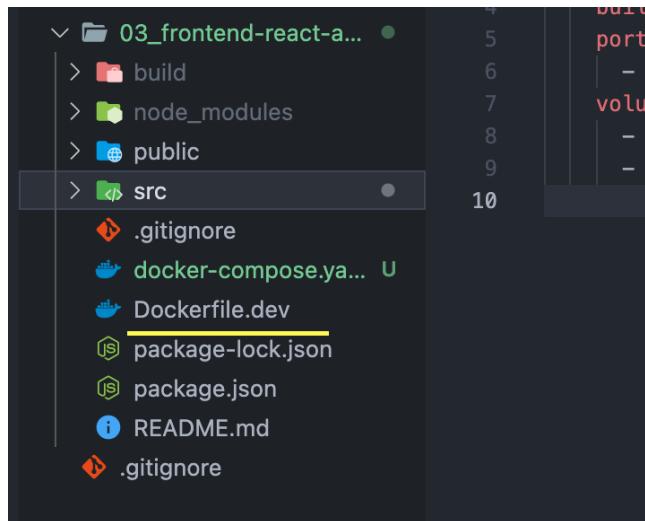
```
Compiled successfully!
assets by status 6.95 KiB [cached] 1 asset
assets by chunk 1.48 MiB (name: main)
  asset static/js/bundle.js 1.47 MiB [emitted] (name: main) 1 related asset
```

We are doing the volume mapping in the docker compose file.

Running the docker compose file

```
→ 03_frontend-react-app git:(main) ✘ docker-compose up
[+] Building 0.0s (1/2)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 2B
failed to solve: rpc error: code = Unknown desc = failed to solve with frontend dockerfile.v0: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount870413150/Dockerfile: no such file or directory
→ 03_frontend-react-app git:(main) ✘
```

When run the docker-compose, we are getting the error as failed to read docker.



Because, we are having the dockerfile with different name. (Instead of Dockerfile, it is Dockerfile.dev)

Overriding the Dockerfile Selection

Updating the docker compose yaml file

We are updating context and docker file required for building the image.

context => Path of the files system required for building the image.

dockerfile => Name of docker file inside the given context.

```
03_frontend-react-app > ⚡ docker-compose.yaml
1   version: "3"
2   services:
3     web:
4       build:
5         context: .
6         dockerfile: Dockerfile.dev
7       ports:
8         - "3000:3000"
9       volumes:
10      - /app/node_modules
11      - .:/app
```

“context: .” represent the current directory.

Running the image using docker-compose command

```
fatal: no such file or directory
→ 03_frontend-react-app git:(main) ✘ docker-compose up
[+] Building 0.2s (10/10) FINISHED
  => [internal] load build definition from Dockerfile.dev          0.0s
  => => transferring dockerfile: 152B                            0.0s
  => [internal] load .dockerignore                                0.0s
  => => transferring context: 2B                                0.0s
  => [internal] load metadata for docker.io/library/node:16-alpine 0.0s
  => [1/5] FROM docker.io/library/node:16-alpine                  0.0s
  => [internal] load build context                                0.1s
  => => transferring context: 1.67MB                            0.1s
  => CACHED [2/5] WORKDIR /app                                0.0s
  => CACHED [3/5] COPY package.json .                           0.0s
  => CACHED [4/5] RUN npm install                             0.0s
  => [5/5] COPY . .                                         0.0s
  => exporting to image                                       0.0s
  => => exporting layers                                     0.0s
  => => writing image sha256:97b9c8b770727139067e3b4c5d9ad5d51e66f325419aa426750d3 0.0s
  => => naming to docker.io/library/03_frontend-react-app_web 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how
to fix them
[+] Running 2/2
  #: Network 03_frontend-react-app_default  Created          0.1s
  #: Container 03_frontend-react-app-web-1  Created        12.3s
Attaching to 03_frontend-react-app-web-1
03_frontend-react-app-web-1 | 
03_frontend-react-app-web-1 | > 03_frontend-react-app@0.1.0 start
03_frontend-react-app-web-1 | > react-scripts start
03_frontend-react-app-web-1 | (node:26) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
03_frontend-react-app-web-1 | (Use `node --trace-deprecation ...` to show where the warning was created)
03_frontend-react-app-web-1 | (node:26) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
03_frontend-react-app-web-1 | Starting the development server...
03_frontend-react-app-web-1 | Compiled successfully!
03_frontend-react-app-web-1 | You can now view 03_frontend-react-app in the browser.
03_frontend-react-app-web-1 | Local:          http://localhost:3000
03_frontend-react-app-web-1 | On Your Network: http://172.18.0.2:3000
03_frontend-react-app-web-1 | Note that the development build is not optimized.
03_frontend-react-app-web-1 | To create a production build, use npm run build.
03_frontend-react-app-web-1 | assets by path static/js/*.js 1.48 MiB
03_frontend-react-app-web-1 |   asset static/js/bundle.js 1.47 MiB [emitted] (name: main) 1 related asset
03_frontend-react-app-web-1 |   asset static/is/node_modules_web-vitals_dist_web-vitals
```

Do we need a copy?

Even though we are specifying the volume mounting in the docker compose, we are still keeping the “COPY ..” instruction in the docker file. If we don’t want to use the docker compose in the future, this copy instruction in the docker file, gives hand.

Executing Tests



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

```
→ frontend git:(081-t) docker run -it 3debad3faf7e npm run test
```

```
PASS src/App.test.js
  ✓ renders without crashing (26ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.116s
Ran all test suites related to changed files.

Watch Usage
> Press p to filter by a filename regex pattern.
> Press t to filter by a test name regex pattern.
> Press q to quit watch mode.
> Press Enter to trigger a test run.
```

Live updating tests

```
Last login: Tue Jul 31 10:47:37 on ttys003
→ frontend git:(081-t) ✘ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              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
```

We can reuse the existing container to run the test. The running container is, ran from docker compose where we have a volume mounting. So, updating the test will be reflected in command prompt.

```

PASS  src/App.test.js
✓ renders without crashing (25ms)
✓ renders without crashing (1ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.17s
Ran all test suites related to changed files.

Watch Usage
> Press p to filter by a filename regex pattern.
> Press t to filter by a test name regex pattern.
> Press q to quit watch mode.
> Press Enter to trigger a test run.

```

Docker Compose for Running the Tests

Updating the docker file

```

03_frontend-react-app > 🛠 docker-compose.yaml
1  version: "3"
2  services:
3    web:
4      build:
5        context: .
6        dockerfile: Dockerfile.dev
7      ports:
8        - "3000:3000"
9      volumes:
10     - /app/node_modules
11     - .:/app
12    tests:
13      build:
14        context: .
15        dockerfile: Dockerfile.dev
16      volumes:
17        - /app/node_modules
18        - .:/app
19      command: ["npm", "run", "test"]
20

```



Running the docker compose

```

→ 03_frontend-react-app git:(main) ✘ docker-compose up --build
[+] Building 59.7s (15/16)
=> [03_frontend-react_app_tests internal] load build definition from Dockerfile.dev  0.0s
=> => transferring dockerfile: 35B  0.0s
=> [03_frontend-react_app_web internal] load build definition from Dockerfile.dev  0.0s
=> => transferring dockerfile: 152B  0.0s
=> [03_frontend-react_app_tests internal] load .dockerignore  0.0s
=> => transferring context: 2B  0.0s
=> [03_frontend-react_app_web internal] load .dockerignore  0.0s
=> => transferring context: 2B  0.0s
=> [03_frontend-react_app_web internal] load metadata for docker.io/library/node:16-a  0.0s
=> [03_frontend-react_app_web 1/5] FROM docker.io/library/node:16-alpine  0.0s
=> [03_frontend-react_app_tests internal] load build context  42.2s
=> => transferring context: 217.37MB  42.1s
=> [03_frontend-react_app_web internal] load build context  42.1s
=> => transferring context: 217.92MB  42.0s
=> CACHED [03_frontend-react_app_tests 2/5] WORKDIR /app  0.0s
=> CACHED [03_frontend-react_app_web 3/5] COPY package.json .  0.0s
=> CACHED [03_frontend-react_app_web 4/5] RUN npm install  0.0s

```

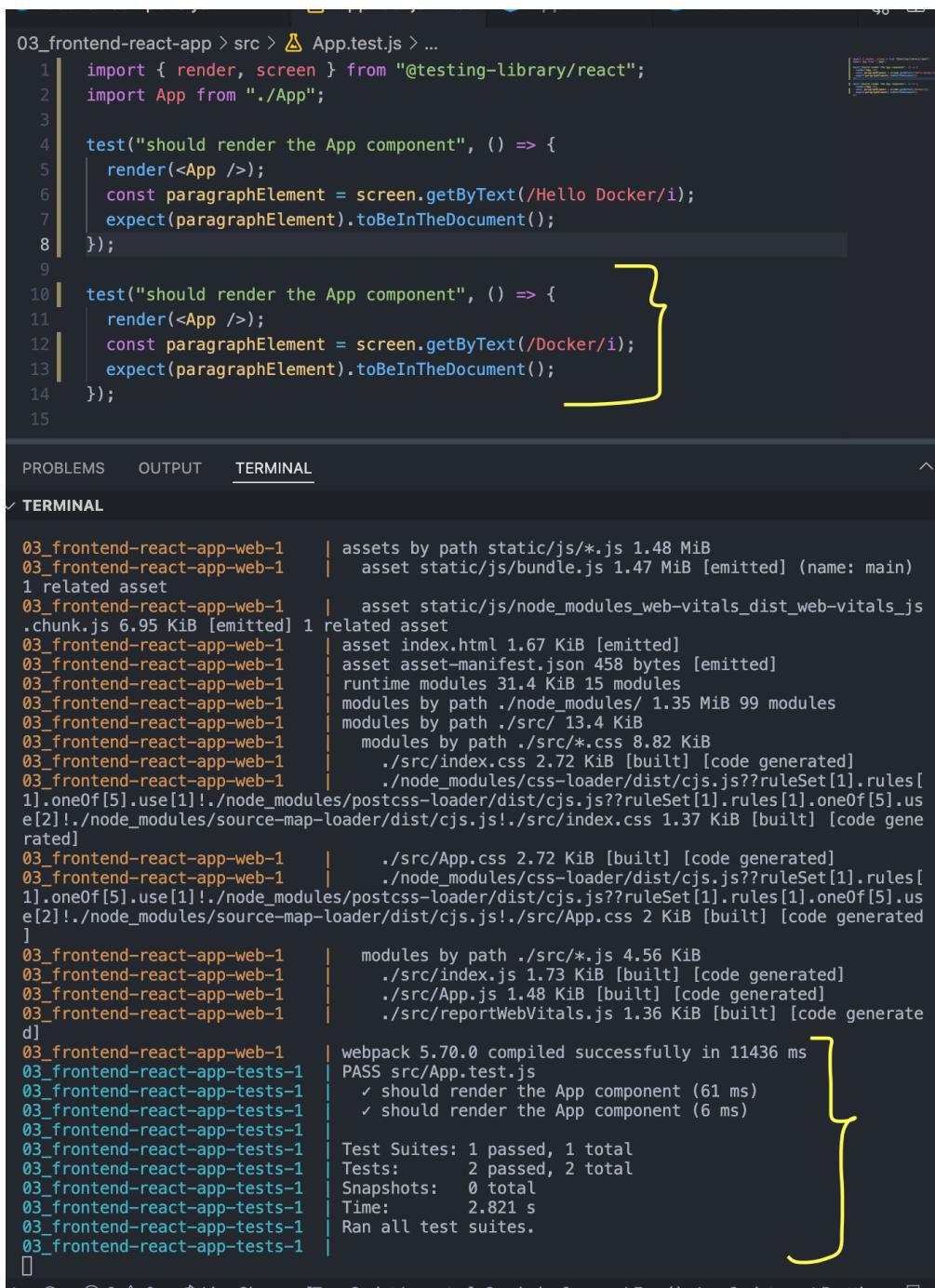
As we can see in the below image, that test is running as part of the docker compose.

```

tewares' option.
03_frontend-react-app-web-1 | (Use `node --trace-deprecation ...` to show where the warnin
g was created)
03_frontend-react-app-web-1 | [node:25] [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE
] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMi
ddlewares' option.
03_frontend-react-app-tests-1 | PASS src/App.test.js
03_frontend-react-app-tests-1 | ✓ should render the App component (63 ms)
03_frontend-react-app-tests-1 | Test Suites: 1 passed, 1 total
03_frontend-react-app-tests-1 | Tests: 1 passed, 1 total
03_frontend-react-app-tests-1 | Snapshots: 0 total
03_frontend-react-app-tests-1 | Time: 2.254 s
03_frontend-react-app-tests-1 | Ran all test suites.
03_frontend-react-app-tests-1 |
03_frontend-react-app-web-1 | Starting the development server...
03_frontend-react-app-web-1 | Compiled successfully!
03_frontend-react-app-web-1 |

```

Modifying the test file



```

03_frontend-react-app > src > App.test.js > ...
1 import { render, screen } from "@testing-library/react";
2 import App from "./App";
3
4 test("should render the App component", () => {
5   render(<App />);
6   const paragraphElement = screen.getByText(/Hello Docker/i);
7   expect(paragraphElement).toBeInTheDocument();
8 });
9
10 test("should render the App component", () => {
11   render(<App />);
12   const paragraphElement = screen.getByText(/Docker/i);
13   expect(paragraphElement).toBeInTheDocument();
14 });
15

```

PROBLEMS OUTPUT TERMINAL

TERMINAL

```

03_frontend-react-app-web-1 | assets by path static/js/*.js 1.48 MiB
03_frontend-react-app-web-1 |   asset static/js/bundle.js 1.47 MiB [emitted] (name: main)
1 related asset
03_frontend-react-app-web-1 |   asset static/js/node_modules_web-vitals_dist_web-vitals_js
.chunk.js 6.95 KiB [emitted] 1 related asset
03_frontend-react-app-web-1 |   asset index.html 1.67 KiB [emitted]
03_frontend-react-app-web-1 |   asset asset-manifest.json 458 bytes [emitted]
03_frontend-react-app-web-1 |   runtime modules 31.4 KiB 15 modules
03_frontend-react-app-web-1 |   modules by path ./node_modules/ 1.35 MiB 99 modules
03_frontend-react-app-web-1 |   modules by path ./src/ 13.4 KiB
03_frontend-react-app-web-1 |     modules by path ./src/*.*.css 8.82 KiB
03_frontend-react-app-web-1 |     ./src/index.css 2.72 KiB [built] [code generated]
03_frontend-react-app-web-1 |     ./node_modules/css-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].us
e[2]!./node_modules/source-map-loader/dist/cjs.js!./src/index.css 1.37 KiB [built] [code gene
rated]
03_frontend-react-app-web-1 |     ./src/App.css 2.72 KiB [built] [code generated]
03_frontend-react-app-web-1 |     ./node_modules/css-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].use[1]!./node_modules/postcss-loader/dist/cjs.js??ruleSet[1].rules[1].oneOf[5].us
e[2]!./node_modules/source-map-loader/dist/cjs.js!./src/App.css 2 KiB [built] [code generated]
03_frontend-react-app-web-1 |   modules by path ./src/*.*.js 4.56 KiB
03_frontend-react-app-web-1 |     ./src/index.js 1.73 KiB [built] [code generated]
03_frontend-react-app-web-1 |     ./src/App.js 1.48 KiB [built] [code generated]
03_frontend-react-app-web-1 |     ./src/reportWebVitals.js 1.36 KiB [built] [code generate
d]
03_frontend-react-app-web-1 | webpack 5.70.0 compiled successfully in 11436 ms
03_frontend-react-app-tests-1 | PASS src/App.test.js
03_frontend-react-app-tests-1 | ✓ should render the App component (61 ms)
03_frontend-react-app-tests-1 | ✓ should render the App component (6 ms)
03_frontend-react-app-tests-1 |
03_frontend-react-app-tests-1 | Test Suites: 1 passed, 1 total
03_frontend-react-app-tests-1 | Tests: 2 passed, 2 total
03_frontend-react-app-tests-1 | Snapshots: 0 total
03_frontend-react-app-tests-1 | Time: 2.821 s
03_frontend-react-app-tests-1 | Ran all test suites.

```

Updating the new test is directly reflected in the container run.

There is one drawback of this approach. Since it is part of the docker compose, we cannot execute any additional test command while container is running.

```
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.627s, estimated 1s
Ran all test suites related to changed files.
```

Watch Usage

- > Press p to filter by a filename regex pattern.
- > Press t to filter by a test name regex pattern.
- > Press q to quit watch mode.
- > Press Enter to trigger a test run.

```
Compiled successfully!
```

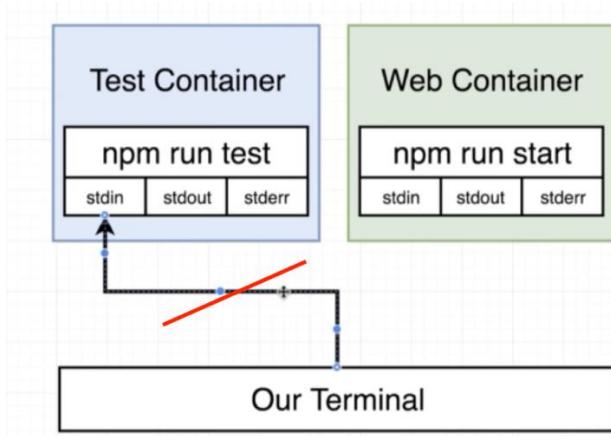


```
web_1  |
web_1  | Note that the development build is not optimized.
web_1  | To create a production build, use npm run build.
web_1  |
```

```
pp
q
t
w
|
```

Shortcomings on Testing

Why are test commands not accepted while container is running?



- We are typing the additional test command in our terminal, here we don't have the default setup to pass the command to Test Container's stdin.
- std{in,out,err} are process specific. Every process inside the container has its own std{in,out,err}.

Docker attach

Docker attach command used to attach terminal to std{in,out,err} of primary process of the container.

After we ran the "docker-compose up", we will be getting the container ID using "docker ps". Then, we will be passing the container ID to docker attach command.

```

/Users/naveen.kumar1/.zshrc:68: no matches found: load?
→ DOCKER_KUBE git:(main) docker ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
a33da6e988b5 03_frontend-react-app_tests "docker-entrypoint.s..." 2
  days ago Up 24 seconds 03_frontend-react-
app-tests-1
42da8371ea3d 03_frontend-react-app_web "docker-entrypoint.s..." 2
  days ago Up 24 seconds 0.0.0.0:3000->3000/tcp 03_frontend-react-
app-web-1
→ DOCKER_KUBE git:(main) docker attach a33da6e988b5
p
q
q

```

If we are trying to run the test commands, it is not working. Let's analyze why.

```

p web-1
→ DOCKER_KUBE git:(main) docker ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
a33da6e988b5 03_frontend-react-app_tests "docker-entrypoint.s..." 2
  days ago Up 5 seconds 03_frontend-react-a
pp-tests-1
42da8371ea3d 03_frontend-react-app_web "docker-entrypoint.s..." 2
  days ago Up 5 seconds 0.0.0.0:3000->3000/tcp 03_frontend-react-a
pp-web-1
→ DOCKER_KUBE git:(main) docker exec -it a33da6e988b5 sh
/app # ps
PID USER TIME COMMAND
 1 root 0:00 npm run test
 19 root 0:00 node /app/node_modules/.bin/react-scripts test
 26 root 0:01 /usr/local/bin/node /app/node_modules/react-script
 78 root 0:00 sh
 84 root 0:00 ps
/app #

```

There multiple process running in the container. When do the docker attach, terminal always attached to the primary process in the container.

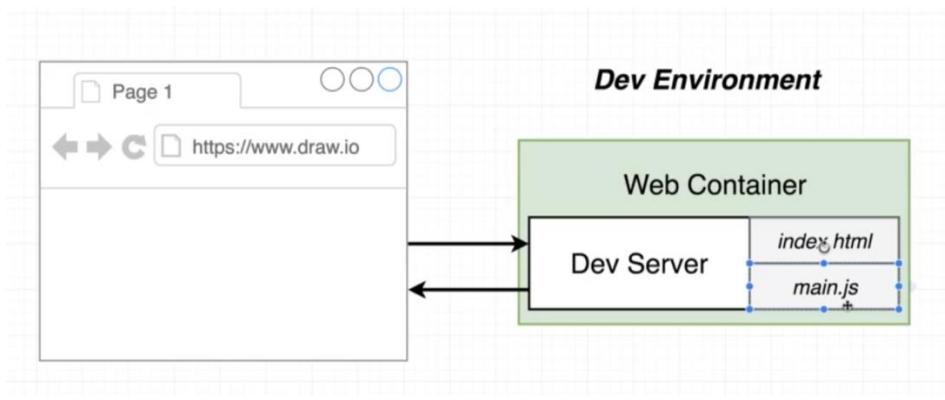
So, unfortunately, we cannot handle the test commands while container is running.

Need for Ngix

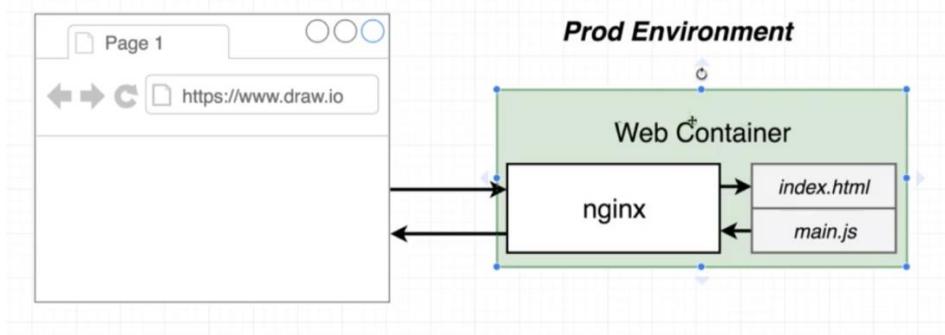
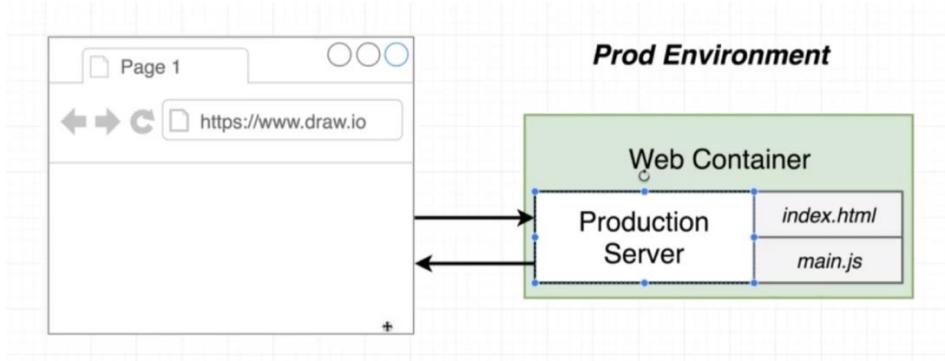
Npm run build

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

Development Environment

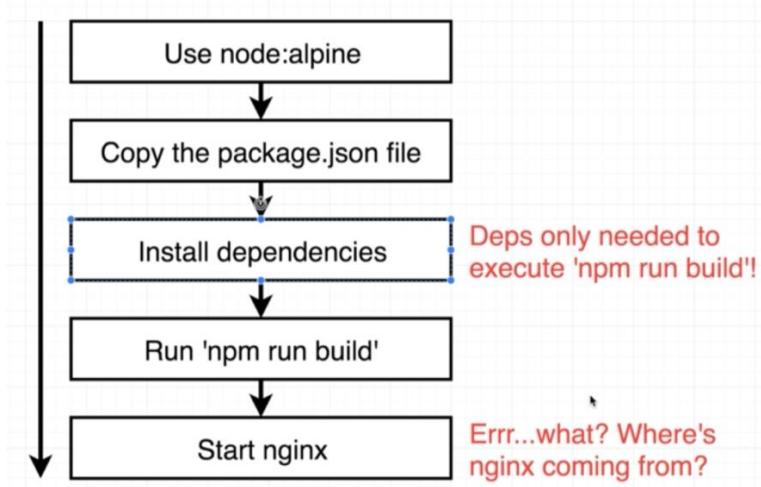


Production Environment



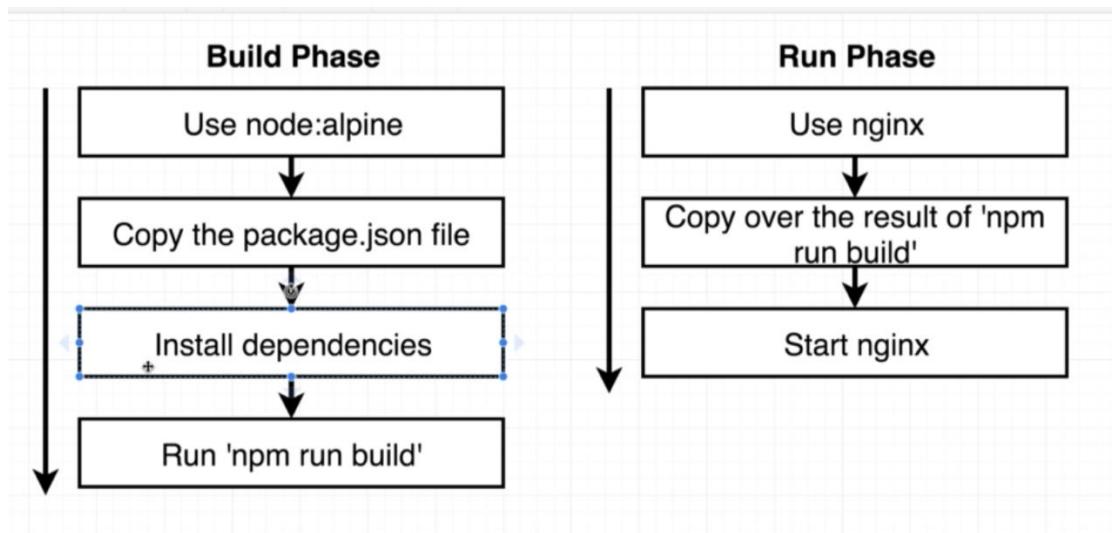
We are going to use the Engine X as a production server.

Multi-Step Docker Build



Once we build the application, dependencies are no longer required. We are using the ngnix for production server.

Multi Step Docker File



We are going to have two base images. One is for build phase and another one is for run phase.

Implementing Multi Step Builds

```
03_frontend-react-app > 🛠 Dockerfile > ...
1  FROM node:16-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
10
```

/usr/share/nginx/html => This the folder where nginx look for the files to run.

Running Nginx

Building the Image

The screenshot shows a terminal window with two tabs: "Dockerfile 03_frontend-react-app" and "Dockerfile 01_simpleweb". The "03_frontend-react-app" tab displays the Dockerfile content:

```
FROM node:16-alpine as builder
WORKDIR '/app'
COPY package.json .
RUN npm install
COPY . .
RUN npm run build

FROM nginx
COPY --from=builder /app/build /usr/share/nginx/html
```

The "TERMINAL" tab shows the output of the "docker build ." command:

```
[+] Building 21.2s (6/14)
[+] 1/2 [internal] load build definition from Dockerfile          0.0s
[+] 1/2 => transferring dockerfile: 218B                          0.0s
[+] 1/2 [internal] load .dockerignore                            0.0s
[+] 1/2 => transferring context: 2B                           0.0s
[+] 1/2 [internal] load metadata for docker.io/library/nginx:latest 4.9s
[+] 1/2 [internal] load metadata for docker.io/library/node:16-alpine 0.0s
[+] 1/2 [auth] library/nginx:pull token for registry-1.docker.io 0.0s
[+] 1/2 [internal] load build context                         16.2s
[+] 1/2 => transferring context: 48.57MB                      15.9s
[+] 1/2 [stage-1 1/2] FROM docker.io/library/nginx@sha256:1c13bc6de5dfca749c3779741 16.2s
[+] 1/2 => resolve docker.io/library/nginx@sha256:1c13bc6de5dfca749c377974146ac05256 0.0s
[+] 1/2 => sha256:c919045c4c2b0b0007c606e763ed2c830c7b1d038ce878a3c0 7.66kB / 7.66kB 0.0s
[+] 1/2 => sha256:4d3e1d15534ccf8091e19bd74215f6512abafb5b341122 24.12MB / 25.35MB 16.2s
[+] 1/2 => sha256:9eb164bd1d87b4b3da53bdda045dff49cf370c803b023c9b9ec2a 601B / 601B 3.0s
[+] 1/2 => sha256:1c13bc6de5dfca749c377974146ac05256791ca2fe1979fc8e 1.86kB / 1.86kB 0.0s
[+] 1/2 => sha256:f7a1c6dad28192bd417b78079d6ddc03cbca6d5ea46bba1 11.53MB / 31.37MB 16.2s
[+] 1/2 => sha256:2468d48e476b6a079eb646e87620f96ce1818ac0c5b3a84505 1.57kB / 1.57kB 0.0s
[+] 1/2 => sha256:59baa8b00c3c90731a08416338a21d113876a624553d34fbc717e6 893B / 893B 3.9s
[+] 1/2 => sha256:a41ae70ab6b499f58a869b3323ec69d8ecb47745aa811c5ee2ed58 664B / 664B 4.9s
[+] 1/2 => sha256:e3908122b958943f7ebb162d6d710262b8a5d50c203f7897d5 1.40kB / 1.40kB 5.8s
[+] 1/2 [builder 1/6] FROM docker.io/library/node:16-alpine           0.0s
```

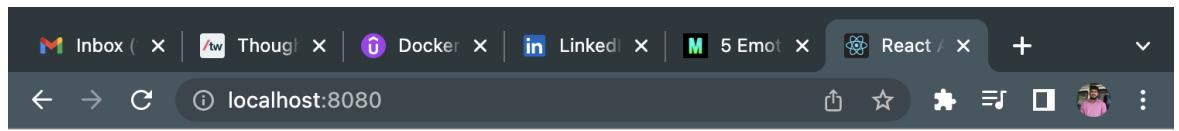
TERMINAL

```
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 218B                         0.0s
=> [internal] load .dockerignore                           0.0s
=> => transferring context: 2B                           0.0s
=> [internal] load metadata for docker.io/library/nginx:latest   4.9s
=> [internal] load metadata for docker.io/library/node:16-alpine    0.0s
=> [auth] library/nginx:pull token for registry-1.docker.io      0.0s
=> [internal] load build context                          39.0s
=> => transferring context: 217.37MB                     38.5s
=> [stage-1 1/2] FROM docker.io/library/nginx@sha256:1c13bc6de5dfca749c3779741 30.6s
=> => resolve docker.io/library/nginx@sha256:1c13bc6de5dfca749c377974146ac05256 0.0s
=> => sha256:c919045c4c2b0b0007c606e763ed2c830c7b1d038ce878a3c0 7.66kB / 7.66kB 0.0s
=> => sha256:4d3e1d15534ccf8091e19bd74215f6512ababfb5b341122 25.35MB / 25.35MB 16.6s
=> => sha256:9ebbe164bd1d87b4b3da53bdda045dff49cf370c803b023c9b9ec2a 601B / 601B 3.0s
=> => sha256:1c13bc6de5dfca749c377974146ac05256791ca2fe1979fc8e 1.86kB / 1.86kB 0.0s
=> => sha256:f7a1c6dad28192bd417b78079d6ddc03cbc6d5ea46bba1 31.37MB / 31.37MB 26.0s
=> => sha256:2468d48e476b6a079eb646e87620f96ce1818ac0c5b3a84505 1.57kB / 1.57kB 0.0s
=> => sha256:59baa8b00c3c90731a08416338a21d113876a624553d34fbc717e6 893B / 893B 3.9s
=> => sha256:a41ae70ab6b499f58a869b3323ec69d8ecb47745aa811c5ee2ed58 664B / 664B 4.9s
=> => sha256:e3908122b958943f7ebb162d6d710262b8a5d50c203f7897d5 1.40kB / 1.40kB 5.8s
=> => extracting sha256:f7a1c6dad28192bd417b78079d6ddc03cbc6d5ea46bba12769b235 2.2s
=> => extracting sha256:4d3e1d15534ccf8091e19bd74215f6512ababfb5b341122e91c1254 1.4s
=> => extracting sha256:9ebbe164bd1d87b4b3da53bdda045dff49cf370c803b023c9b9ec2a4 0.0s
=> => extracting sha256:59baa8b00c3c90731a08416338a21d113876a624553d34fbc717e6c 0.0s
=> => extracting sha256:a41ae70ab6b499f58a869b3323ec69d8ecb47745aa811c5ee2ed58c 0.0s
=> => extracting sha256:e3908122b958943f7ebb162d6d710262b8a5d50c203f7897d5afa60 0.0s
=> [builder 1/6] FROM docker.io/library/node:16-alpine           0.0s
=> CACHED [builder 2/6] WORKDIR /app                            0.0s
=> CACHED [builder 3/6] COPY package.json .                      0.0s
=> CACHED [builder 4/6] RUN npm install                         0.0s
=> [builder 5/6] COPY . .                                     12.9s
=> [builder 6/6] RUN npm run build                           18.1s
=> [stage-1 2/2] COPY --from=builder /app/build /usr/share/nginx/html 0.1s
=> exporting to image                                       0.0s
=> => exporting layers                                      0.0s
=> => writing image sha256:6966026fc67d8aeb7874f61d13d51575645e1fabc8007a8748cb 0.0s
```

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

Running the Image

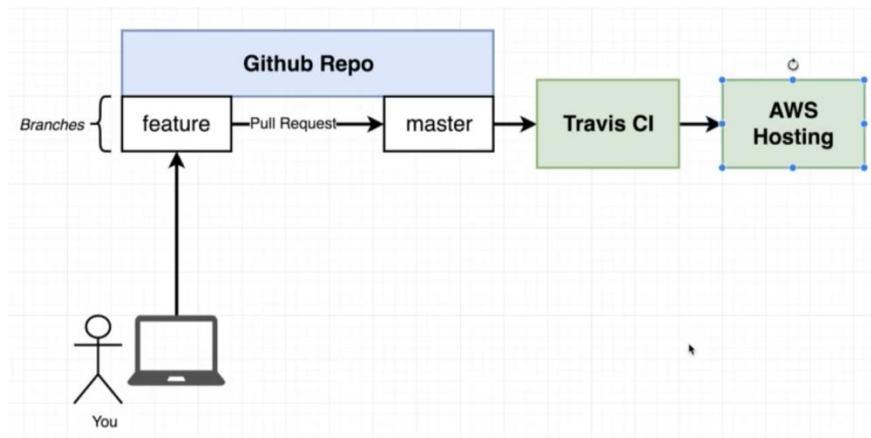
```
→ 03_frontend-react-app git:(main) ✘ docker run -p 8080:80 6966026fc67d8aeb7874f61d13d51575645e1fabc8007a8748cb
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/03/14 08:19:24 [notice] 1#1: using the "epoll" event method
2022/03/14 08:19:24 [notice] 1#1: nginx/1.21.6
2022/03/14 08:19:24 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/03/14 08:19:24 [notice] 1#1: OS: Linux 5.10.76-linuxkit
2022/03/14 08:19:24 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2022/03/14 08:19:24 [notice] 1#1: start worker processes
2022/03/14 08:19:24 [notice] 1#1: start worker process 31
2022/03/14 08:19:24 [notice] 1#1: start worker process 32
2022/03/14 08:19:24 [notice] 1#1: start worker process 33
2022/03/14 08:19:24 [notice] 1#1: start worker process 34
2022/03/14 08:19:24 [notice] 1#1: start worker process 35
2022/03/14 08:19:24 [notice] 1#1: start worker process 36
```



Hello Docker !!

Continuous Integration and Deployment with AWS

Services Overview



We will be using the Github Actions platform in the place of Travis CI.

Continuous Integration with Github Actions

Creating a Github Actions configuration file

```
REACT-APP-FOR-DOCKER
  .github > workflows > deploy.yaml
    name: Deploy Frontend
    on:
      push:
        branches:
          - main
    jobs:
      build:
        build:
          runs-on: ubuntu-latest
          steps:
            - uses: actions/checkout@v2
            - run: docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
            - run: docker build -t mnaveensmn/docker-react -f Dockerfile.dev .
            - run: docker run -e CI=true mnaveensmn/docker-react npm test
```

In this config, we are,

- Building the docker image “mnaveensmn/docker-react” using the Dockerfile.dev file.
- Running the build image with initial command as npm run test.

Checking the build

[mnaveensmn / docker-react](#) Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

✓ renamed folder to workflows Deploy Frontend #1

Summary Jobs

build

build succeeded 14 minutes ago in 6m 2s

> Set up job

> Run actions/checkout@v2

> Run docker login -u *** -p ***

> Run docker build -t ***/docker-react -f Dockerfile.dev .

> Run docker run -e CI=true ***/docker-react npm test

Build output.

AWS Elastic Beanstalk

What is it?

- Easiest way to get started with production docker instances.
- It must be appropriate when running a one container at a time.
- It automatically scales the application using load balancer (By creating multiple instances as required)

Creating an application in elastic beanstalk

aws Services Search for services, features, blogs, docs, [Option+S] N. Virginia Naveen Kumar M

Elastic Beanstalk

Environments Applications Change history

Elastic Beanstalk > Getting started

Create a web app

Create a new application and environment with a sample application or your own code. By creating an environment, you allow Amazon Elastic Beanstalk to manage Amazon Web Services resources and permissions on your behalf. [Learn more](#)

Application information

Application name Up to 100 Unicode characters, not including forward slash (/).

Application tags

Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. [Learn more](#)

Key	Value
<input type="text"/>	<input type="text"/>

[Remove tag](#) [Add tag](#) 50 remaining

Feedback English (US) ▾ Privacy Terms Cookie preferences

<https://docs.aws.amazon.com/console/elasticbeanstalk/tags> Amazon Internet Services Private Ltd. or its affiliates.

Platform

Platform
Docker

Platform branch
Docker running on 64bit Amazon Linux 2

Platform version
3.4.12 (Recommended)

Application code

Sample application
Get started right away with sample code.

Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

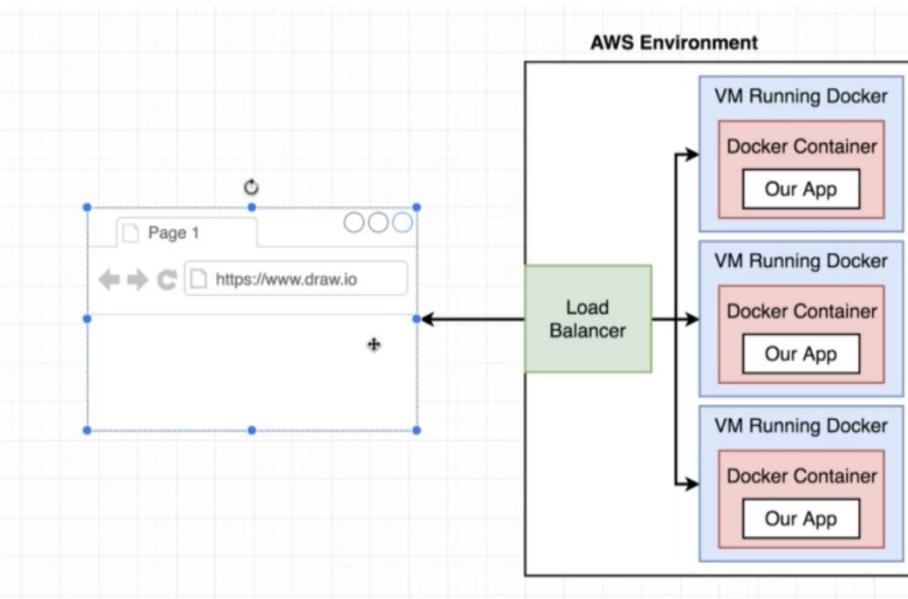
[Cancel](#) [Configure more options](#) [Create application](#)

Elastic Beanstalk > Environments > Dockerreact-env

Creating Dockerreact-env
This will take a few minutes. .

2:22pm Using elasticbeanstalk-us-east-1-218788053233 as Amazon S3 storage bucket for environment data.

2:22pm createEnvironment is starting.



Dockerreact-env

Dockerreact-env.eba-nmvxxup.us-east-1.elasticbeanstalk.com (e-5mqfbpj3ku)

Application name: docker-react

Health

Ok

Running version

Sample Application

Platform

Docker running on 64bit Amazon Linux 2/3.4.12

Github Actions Configurations for Deployment

Creating a user for access in AWS IAM Service

Identity and Access Management (IAM)

Access management

Users

Add users

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[Add another user](#)

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Select AWS credential type*
- Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
 - Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

- Console password*
- Autogenerated password
 - Custom password

Require password reset
User must create a new password at next sign-in
Users automatically get the **IAMUserChangePassword** policy to allow them to change their own password.

* Required

Cancel

Next: Permissions

Set permissions

Add user to group Copy permissions from existing user **Attach existing policies directly**

Get started with groups
You haven't created any groups yet. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. Get started by creating a group. [Learn more](#)

[Create group](#)

Set permissions boundary

Cancel Previous **Next: Tags**

Set permissions

Add user to group Copy permissions from existing user **Attach existing policies directly**

[Create policy](#)

Filter policies Showing 14 results

	Policy name	Type	Used as
<input checked="" type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None

AdministratorAccess-AWSElasticBeanstalk
Grants account administrative permissions. Explicitly allows developers and administrators to gain direct access to resources they need to manage AWS Elastic Beanstalk applications

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition. For details, choose [Show remaining](#). [Learn more](#)

Policy summary [JSON](#)

Service	Access level	Resource	Request condition
Allow (20 of 319 services) Show remaining 299			

Cancel Previous **Next: Tags**

Add user

1 2 3 4 **5**

Success
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://218788053233.signin.aws.amazon.com/console>

[Download .csv](#)

	User	Access key ID	Secret access key	Password	Email login instructions
<input checked="" type="checkbox"/>	docker-react...	██████████	***** Show	***** Show	Send email

Setting the secrets for repository

The screenshot shows the GitHub repository settings page for 'mnavenmsn/docker-react'. The left sidebar is collapsed, and the main area is titled 'Actions secrets'. It displays two sections: 'Environment secrets' (empty) and 'Repository secrets' (containing four secrets: AWS_ACCESS_KEY, AWS_SECRET_KEY, DOCKER_PASSWORD, and DOCKER_USERNAME, all updated 1 hour ago). A 'New repository secret' button is at the top right.

Updating the Github Actions deploy.yaml file

The screenshot shows the 'deploy.yaml' file in the GitHub Actions workflow section of the repository. The code defines a workflow named 'Deploy Frontend' that triggers on pushes to the 'main' branch. It uses an Ubuntu-latest runner, runs actions to checkout, login to Docker, build the Dockerfile, run tests, generate a deployment package (zip), and finally deploy to Elastic Beanstalk using the 'einaregilsson/beanstalk-deploy@v18' action with specific AWS credentials and environment variables.

```
.github > workflows > deploy.yaml
1 name: Deploy Frontend
2 on:
3   push:
4     branches:
5       - main
6
7   jobs:
8     build:
9       runs-on: ubuntu-latest
10      steps:
11        - uses: actions/checkout@v2
12        - run: docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
13        - run: docker build -t mnavenmsn/docker-react -f Dockerfile.dev .
14        - run: docker run -e CI=true mnavenmsn/docker-react npm test
15
16        - name: Generate deployment package
17          run: zip -r deploy.zip . -x '*.git*'
18
19        - name: Deploy to EB
20          uses: einaregilsson/beanstalk-deploy@v18
21          with:
22            aws_access_key: ${{ secrets.AWS_ACCESS_KEY }}
23            aws_secret_key: ${{ secrets.AWS_SECRET_KEY }}
24            application_name: docker-react
25            environment_name: Dockerreact-env
26            existing_bucket_name: elasticbeanstalk-us-east-1-218788053233
27            region: us-east-1
28            version_label: ${{ github.sha }}
29            deployment_package: deploy.zip
```

Commit and Push

Once we commit and push, we can see the workflow running in github actions.

The screenshot shows the GitHub Actions interface for a repository named 'mnavenmsn/docker-react'. A specific workflow run titled 'renamed folder to workflows Deploy Frontend #1' is displayed. The 'build' job is highlighted with a red underline, indicating it is the current focus. The job status is 'succeeded 20 minutes ago in 6m 2s'. The job steps listed are: Set up job, Run actions/checkout@v2, Run docker login -u *** -p ***, Run docker build -t ***/docker-react -f Dockerfile.dev ., Run docker run -e CI=true ***/docker-react npm test, Generate deployment package, Deploy to EB, Post Run actions/checkout@v2, and Complete job. Each step has a timestamp next to it: 1s, 1s, 0s, 1m 18s, 3s, 0s, 4m 35s, 0s, and 0s respectively. A 'Search logs' bar is at the top right, and a 'Re-run all jobs' button is at the bottom right.

The screenshot shows a Chrome browser window with multiple tabs open. The active tab displays a React application with the heading 'Hello Docker !!'. The browser's address bar shows the URL 'Not Secure | dockerreact-env.eba-nmvxxxup.us-east-1.elasticbeanstalk.com'. Other tabs include 'Thoughtworks Me', 'Thoughtworks - S', 'mnavenmsn (Na', 'renamed folder to', 'React App', 'Welcome', 'einaregilsson/bea', and 'bucket name in'. The status bar at the bottom right shows the date 'Sun 20 Mar 21:36'.

Exposing ports through docker file

There is no port in the container expose by default. We must expose manually in the docker file.

```
🚢 Dockerfile > ...
1  FROM node:16-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  EXPOSE 80
10 COPY --from=builder /app/build /usr/share/nginx/html
11 |
```

Workflow with Github

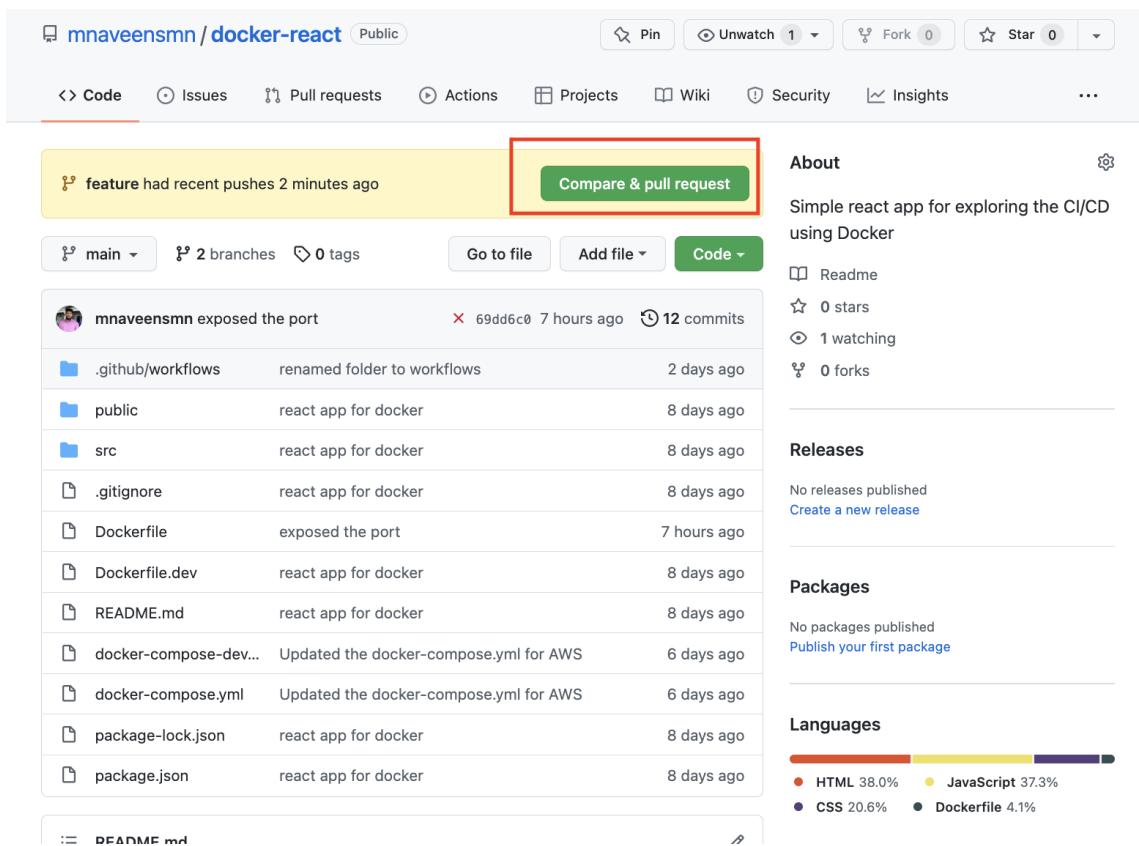
Adding a new feature

```
src > 📄 App.js > ...
1  import "./App.css";
2
3  Complexity is 5 Everything is cool!
4  function App() { █
5    return (
6      <div className="App">
7        <h1>Hello Docker !!</h1>
8        <h1>New Feature</h1>
9      </div>
10    );
11
12  export default App;
13 |
```

New git branch

```
        Checkout
→ react-app-for-docker git:(main) git checkout -b feature
Switched to a new branch 'feature'
→ react-app-for-docker git:(feature) ✘ git add .
→ react-app-for-docker git:(feature) ✘ git commit -m 'Added the new feature'
[feature cc257bd] Added the new feature
1 file changed, 1 insertion(+)
→ react-app-for-docker git:(feature) git push origin feature
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 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), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:     https://github.com/mnaveensmn/docker-react/pull/new/feature
remote:
To github.com:mnaveensmn/docker-react.git
 * [new branch]      feature -> feature
→ react-app-for-docker git:(feature) □
```

Compare and pull request



The screenshot shows a GitHub repository page for 'mnaveensmn / docker-react'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and a dropdown menu. A yellow banner at the top indicates 'feature had recent pushes 2 minutes ago'. Below the banner is a green 'Compare & pull request' button, which is highlighted with a red rectangular box. The main content area displays a list of recent commits:

Author	Commit Message	Date
mnaveensmn	exposed the port	69dd6c0 7 hours ago
.github/workflows	renamed folder to workflows	2 days ago
public	react app for docker	8 days ago
src	react app for docker	8 days ago
.gitignore	react app for docker	8 days ago
Dockerfile	exposed the port	7 hours ago
Dockerfile.dev	react app for docker	8 days ago
README.md	react app for docker	8 days ago
docker-compose-dev...	Updated the docker-compose.yml for AWS	6 days ago
docker-compose.yml	Updated the docker-compose.yml for AWS	6 days ago
package-lock.json	react app for docker	8 days ago
package.json	react app for docker	8 days ago

On the right side of the page, there are sections for 'About', 'Releases', 'Packages', and 'Languages'. The 'About' section describes the repository as a 'Simple react app for exploring the CI/CD using Docker'. The 'Languages' section shows a horizontal bar with the following data:

Language	Percentage
HTML	38.0%
JavaScript	37.3%
CSS	20.6%
Dockerfile	4.1%

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, it says "base: main" and "compare: feature". A green checkmark indicates "Able to merge. These branches can be automatically merged." The main area has a title "Added the new feature" and a heading "Added the new feature heading". Below the heading is a text input field with placeholder text "Attach files by dragging & dropping, selecting or pasting them." At the bottom right of this area is a green "Create pull request" button, which is highlighted with a red box. To the right of the main area, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (Use Closing keywords in the description to automatically close issues). A note at the bottom left says "Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)".

Added the new feature #1

The screenshot shows a GitHub pull request page for a pull request titled "Added the new feature #1". The pull request is from "mnavveensmn" to "main" from the "feature" branch. The commit history shows two commits from "mnavveensmn" with the message "Added the new feature". A note below the commits says "Add more commits by pushing to the feature branch on mnavveensmn/docker-react.". A green box highlights a message: "This branch has no conflicts with the base branch. Merging can be performed automatically." Below this is a green "Merge pull request" button. A note below the button says "You can also [open this in GitHub Desktop](#) or view [command line instructions](#)." The right side of the screen shows standard pull request settings: "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), "Projects" (None yet), "Milestone" (No milestone), and "Development" (Successfu... merging this pull request may close these issues.). There is also a "Notifications" section with a "Customize" link and an "Unsubscribe" button.

Merging the pull request

Added the new feature #1

Merged mnaveensmn merged 2 commits into `main` from `feature` 43 seconds ago

Conversation 0 Commits 2 Checks 0 Files changed 1

mnaveensmn commented 14 minutes ago Owner ...
No description provided.

mnaveensmn added 2 commits 22 minutes ago
Added the new feature cc257bd
Added the new feature bb559d4

mnaveensmn merged commit db759cd into main 43 seconds ago Revert

mnaveensmn commented 34 seconds ago Owner Author ...
Code looks good.

Pull request successfully merged and closed Delete branch
You're all set—the `feature` branch can be safely deleted.

Our pipeline automatically deploying the merged code

Tell us how to make GitHub Actions work better for you with three quick questions. Give feedback X

All workflows

Showing runs from all workflows

Filter workflow runs

3 workflow runs	Event	Status	Branch	Actor
Merge pull request #1 from mnaveensmn/fe... Deploy Frontend #3: Commit db759cd pushed by mnaveensmn	main	2 minutes ago In progress		
exposed the port Deploy Frontend #2: Commit 69dd6c0 pushed by mnaveensmn	main	8 hours ago 4m 16s		
renamed folder to workflows Deploy Frontend #1: Commit 8d9def8 pushed by mnaveensmn	main	2 days ago 6m 13s		

Dockerreact-env

Dockerreact-env.eba-nmvxxxup.us-east-1.elasticbeanstalk.com (e-5mqfbpj3ku)

Application name: docker-react

↻ Refresh

Actions ▾

Health

Ok

Causes

Running versiondb759cdca117e7151f3a00618b0
71bd1fc8ff3c8

Upload and deploy

PlatformDocker running on 64bit Amazon
Linux 2/3.4.12

Change

Not Secure | dockerreact-env.eba-nmvxxxup.us-east-1.elasticbeanstalk.com

Hello Docker !!

New Feature

Terminating AWS Elastic beanstalk Environment



Elastic Beanstalk is terminating your environment.

[View Events](#)**Dockerreact-env**

Dockerreact-env.eba-nmvxxxup.us-east-1.elasticbeanstalk.com (e-5mqfbpj3ku)

Application name: docker-react

↻ Refresh

Actions ▾

Health

Info

Causes

Running versiondb759cdca117e7151f3a00618b0
71bd1fc8ff3c8

Upload and deploy

PlatformDocker running on 64bit Amazon
Linux 2/3.4.12

Change

Building Multi Container Application

Single Container Deployment Issue

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?

Fibonacci Sequence Generator

Application Overview

Fibonacci Series

Fib Sequence:

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

Returns the fib sequence value for given index.

Page 1

https://www.draw.io

Fib Calculator

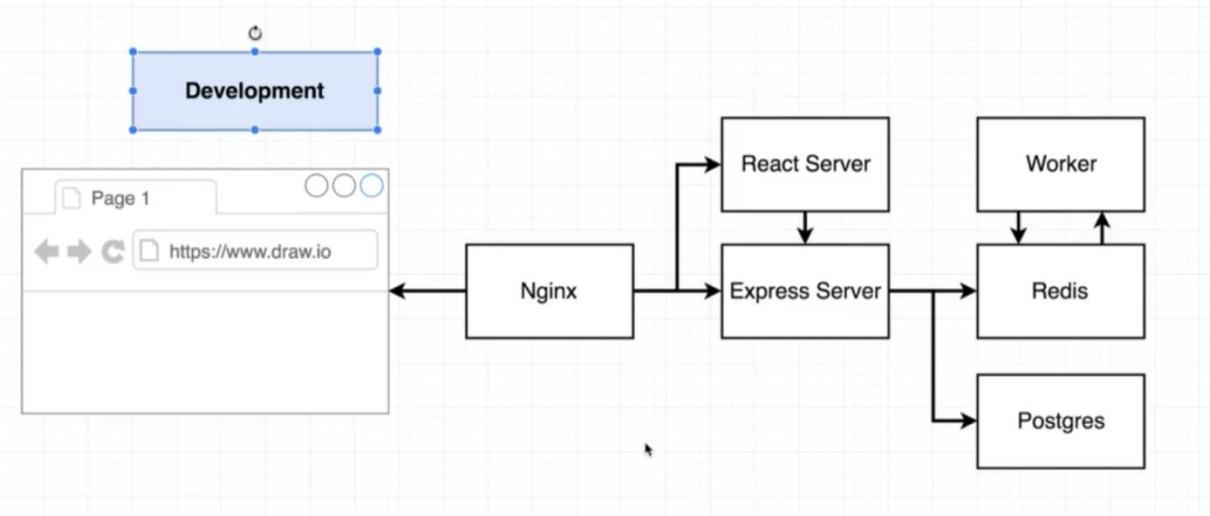
Enter your index: Submit

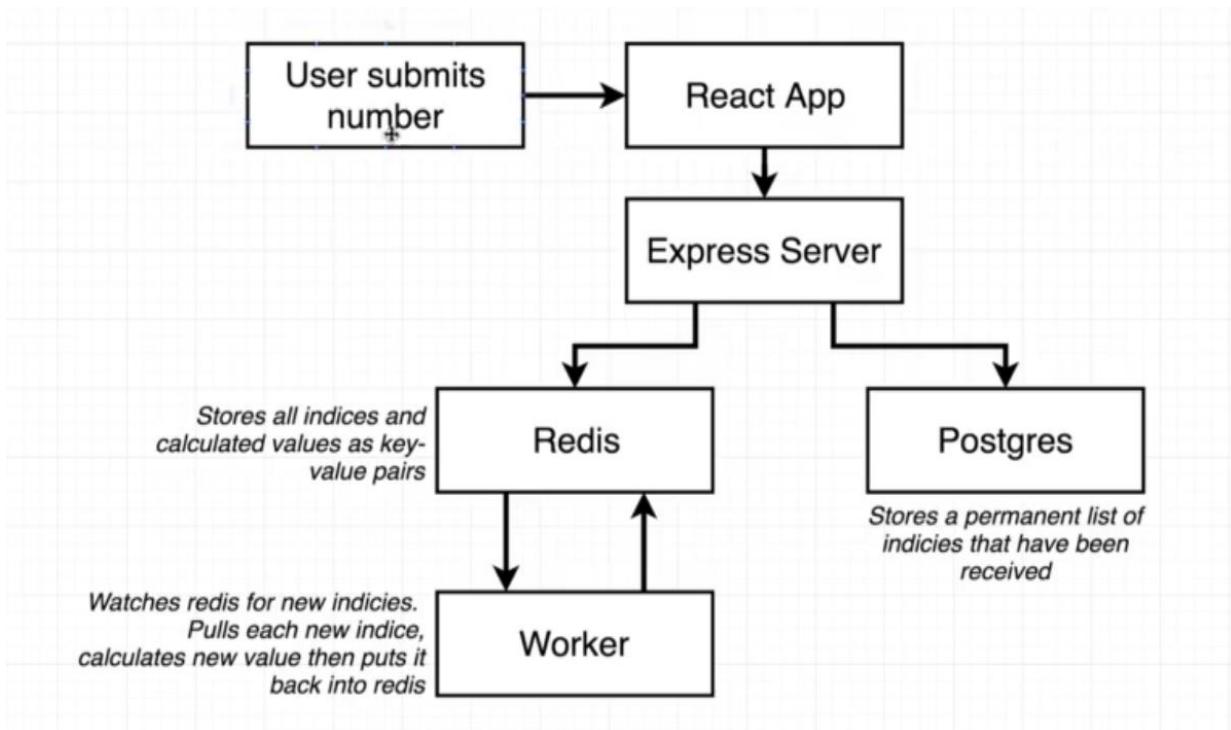
Indices I have seen:
10, 5

Calculated Values:

For index 10 I calculated 89
For index 5 I calculated 8

Application Architecture





Application Development

Worker

Adding the package.json file.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows a folder named "FIBONAC..." containing a "worker" folder, "node_modules", "index.js", "package-lock.json", and "package.json".
- Get Started** tab: Active tab.
- package.json** tab: Active tab.
- index.js** tab: Available tab.
- package.json** content (highlighted):

```

1  {
2   "dependencies": {
3     "nodemon": "1.18.3",
4     "redis": "2.8.0"
5   },
6   "scripts": {
7     "start": "node index.js",
8     "dev": "nodemon"
9   }
10 }
11

```

Configuring Redis and adding the Fibonacci series algorithm

```
worker > [js] keys.js > [?] <unknown>
```

```
1 module.exports = {
2   ...redisHost: process.env.REDIS_HOST,
3   ...redisPort: process.env.REDIS_PORT,
4 };
5
```

```
worker > [js] index.js > ...
```

```
1 const keys = require("./keys.js");
2 const redis = require("redis");
3
4 const redisClient = redis.createClient({
5   host: keys.redisHost,
6   port: keys.redisPort,
7   retry_strategy: () => 1000,
8 });
9
10 const subscription = redisClient.duplicate();
11
12 Complexity is 4 Everything is cool!
13 function fib(index) {
14   if (index < 2) {
15     return 1;
16   }
17   return fib(index - 1) + fib(index - 2);
18 }
19
20 subscription.on("message", (channel, messsgage) => {
21   redisClient.hset("value", messsgage, fib(parseInt(messsgage)));
22 });
23
24 subscription.subscribe("insert");
```

Server

Express and Postgres setup

```
server > js index.js > ...
1  const keys = require("./keys");
2
3  // Express App Setup
4  const express = require("express");
5  const bodyParser = require("body-parser");
6  const cors = require("cors");
7
8  //cors - cross version resource sharing
9  const app = express();
10 app.use(cors());
11 app.use(bodyParser.json());
12
13 //Postgress Client Set
14 const { Pool } = require("pg");
15 const pgClient = new Pool({
16   user: keys.pgUser,
17   host: keys.pgHost,
18   database: keys.pgDatabase,
19   password: keys.pgPassword,
20   port: keys.pgPort,
21 });
22
23 pgClient.on("connect", (client) => {
24   client
25     .query("CREATE TABLE IF NOT EXISTS values (number INT)")
26     .catch((err) => console.error(err));
27 });
28
```

```
server > js keys.js > ...
```

```
1  module.exports = {
2    redisHost: process.env.REDIS_HOST,
3    redisPort: process.env.REDIS_PORT,
4    pgUser: process.env.PGUSER,
5    pgHost: process.env.PGHOST,
6    pgDatabase: process.env.PGDATABASE,
7    pgPassword: process.env.PGPASSWORD,
8    pgPort: process.env.PGPORT,
9  };
10
```

Redis client setup and Express handler

```
// 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);
  });
});

Complexity is 3 Everything is cool!
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");
});
```

Client

Creating a react app

```
/Users/naveen.kumar1/Desktop/no-matches-found-load.
→ fibonacci-generator-web npx create-react-app client

Creating a new React app in /Users/naveen.kumar1/Desktop/fibonacci-generator-w
eb/client.

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

added 1372 packages in 3m

169 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...
npm WARN deprecated source-map-resolve@0.6.0: See https://github.com/lydell/source-m
ap-resolve#deprecated

added 38 packages in 9s

169 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...
```

Fetching the data in react app

```
Complexity is 7 It's time to do something...
class Fib extends Component { █
  state = {
    seenIndexes: [],
    values: {},
    index: "",
  };

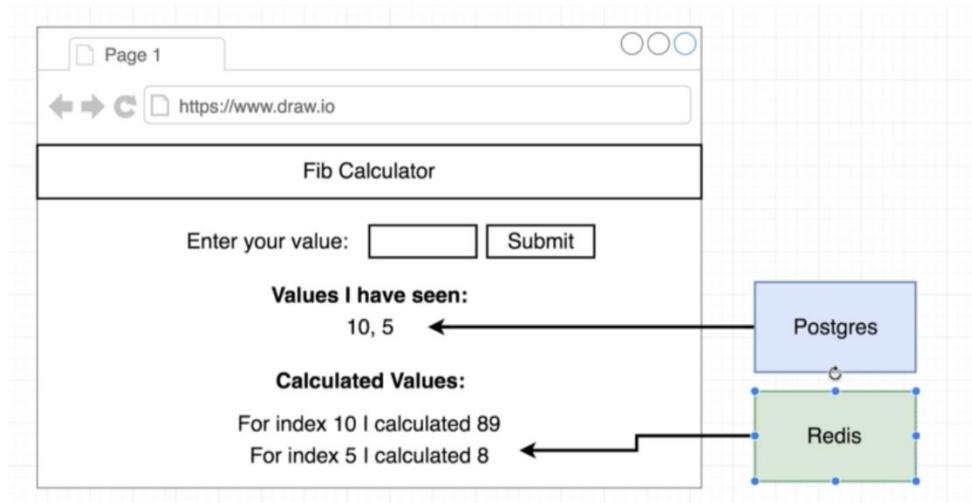
  componentDidMount() {
    this.fetchValues();
    this.fetchIndexes();
  }

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

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

Complexity is 7 It's time to do something...
render() { █
  return (
    <div>
      <form>
        <label>Enter your index:</label>
        <input />
        <button>Submit</button>
      </form>
    </div>
  );
}
```

Rendering logic



Routing in react app

```
> src > JS App.js > ...
import "./App.css";
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
import OtherPage from "./OtherPage";
import Fib from "./Fib";

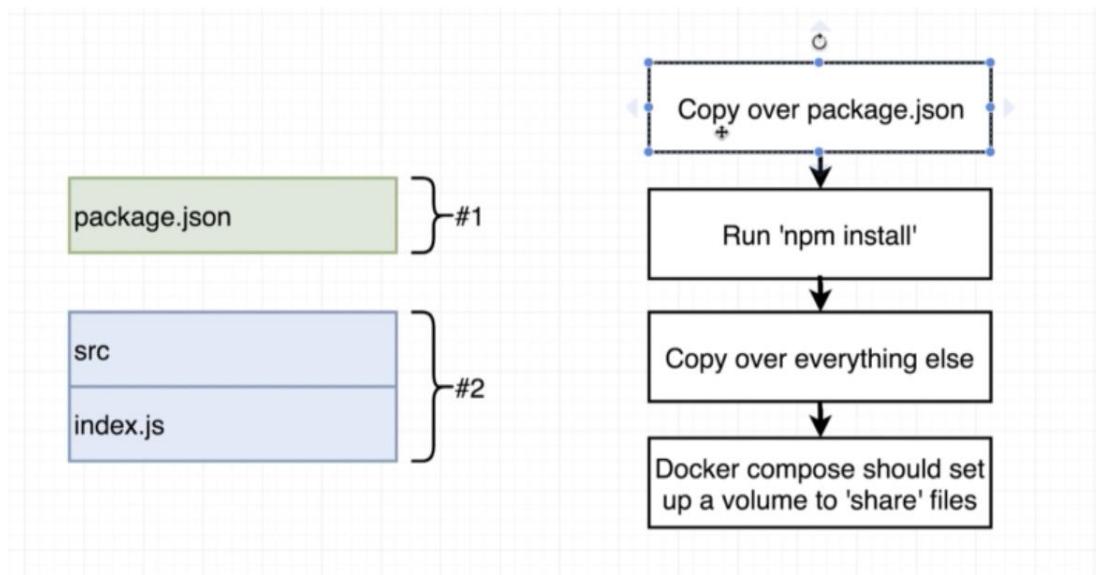
Complexity is 8 It's time to do something...
function App() {
  return (
    <Router>
      <div className="App">
        <Link to="/">Home</Link>
        <Link to="/otherpage">Other Page</Link>

        <Route exact path="/" component={Fib} />
        <Route exact path="/otherpage" component={OtherPage} />
      </div>
    </Router>
  );
}

export default App;
```

Dockerizing the Multi Services

Dockerizing the React App



Creating a dev dockerfile.

The screenshot shows the project structure and the content of the `Dockerfile.dev`:

```
client > Dockerfile.dev > ...
1  FROM node:16-alpine
2  WORKDIR '/app'
3
4  COPY ./package.json ./
5
6  RUN npm install -g npm@8.5.5
7
8  COPY . .
9
10 CMD ["npm", "run", "start"]
11
```

The project structure on the left includes:

- client:** Contains `node_modules`, `public`, `src`, `.gitignore`, and `Dockerfile.dev`.
- server:** Contains `node_modules` and `index.js`.

Building a dev docker file.

```

executor failed running [/bin/sh -c npm install]: exit code: 1
→ client docker build -f Dockerfile.dev .
[+] Building 25.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile.dev      0.0s
=> => transferring dockerfile: 176B                          0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                            0.0s
=> [internal] load metadata for docker.io/library/node:16-a 0.0s
=> [internal] load build context                           1.8s
=> => transferring context: 3.06MB                         1.8s
=> [1/5] FROM docker.io/library/node:16-alpine           0.0s
=> CACHED [2/5] WORKDIR /app                            0.0s
=> CACHED [3/5] COPY ./package.json ./                  0.0s
=> [4/5] RUN npm install -g npm@8.5.5                10.2s
=> [5/5] COPY . . .                                     7.0s
=> exporting to image                                    6.3s
=> => exporting layers                                 6.3s
=> => writing image sha256:5fd5f07f816a9960517b03f777ab8ad0 0.0s

```

Running the docker image.

```

→ client docker run 5fd5f07f816a9960517b03f777ab8ad0
> client@0.1.0 start
> react-scripts start

i [wds]: Project is running at http://172.17.0.2/
i [wds]: webpack output is served from
i [wds]: Content not from webpack is served from /app/public
i [wds]: 404s will fallback to /
Starting the development server...

→ client

```

Dockerizing the generic node app

Adding a docker file for server and worker project. It is a same command for both the dockerfiles.

```

{
  "client": {
    "Dockerfile.dev": [
      "FROM node:14.14.0-alpine",
      "WORKDIR '/app'",
      "COPY ./package.json ./",
      "RUN npm install",
      "COPY . .",
      "CMD ["npm", "run", "dev"]"
    ]
  },
  "server": {
    "Dockerfile.dev": [
      "FROM node:14.14.0-alpine",
      "WORKDIR '/app'",
      "COPY ./package.json ./",
      "RUN npm install",
      "COPY . .",
      "CMD ["npm", "run", "dev"]"
    ]
  }
}

```

Building a docker image

```

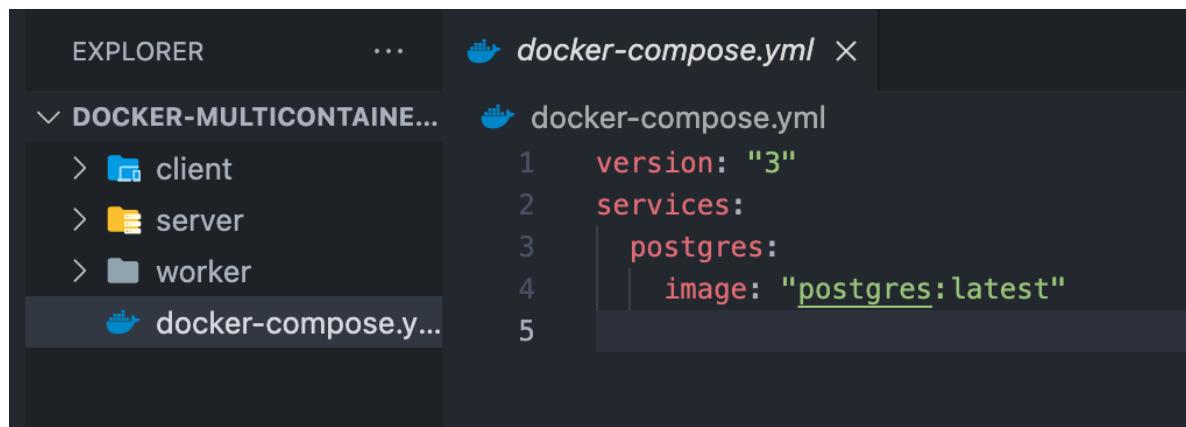
=> => extracting sha256:378c9540b231b522ab12144ad767b0abb1c 1.6s
=> => extracting sha256:fd5173d1722049170dadc1ec1e8a0653b8c 0.1s
=> => extracting sha256:72f138d46980234e4f5b2469131de318f07 0.0s
=> [internal] load build context 3.4s
=> => transferring context: 8.25MB 3.4s
=> [2/5] WORKDIR /app 0.1s
=> [3/5] COPY ./package.json ./ 0.0s
=> [4/5] RUN npm install 18.8s
=> [5/5] COPY . . 0.2s
=> exporting to image 0.5s
=> => exporting layers 0.5s
=> => writing image sha256:7e99c3a97a32f10f6e764a20eae11eb8 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ server cd ..
→ DOCKER-MULTICONTAINER-APP cd worker
→ worker docker build -f Dockerfile.dev .
[+] Building 13.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile.dev 0.0s
=> => transferring dockerfile: 164B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/node:14.1 1.6s
=> [1/5] FROM docker.io/library/node:14.14.0-alpine@sha256: 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 1.22kB 0.1s
=> CACHED [2/5] WORKDIR /app 0.0s
=> [3/5] COPY ./package.json ./ 0.0s
=> [4/5] RUN npm install 11.0s
=> [5/5] COPY . . 0.0s
=> exporting to image 0.4s
=> => exporting layers 0.4s
=> => writing image sha256:16395da1d2c8e4f1ff5a48b4cebfa808 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
→ worker []

```

Adding a postgres as a service

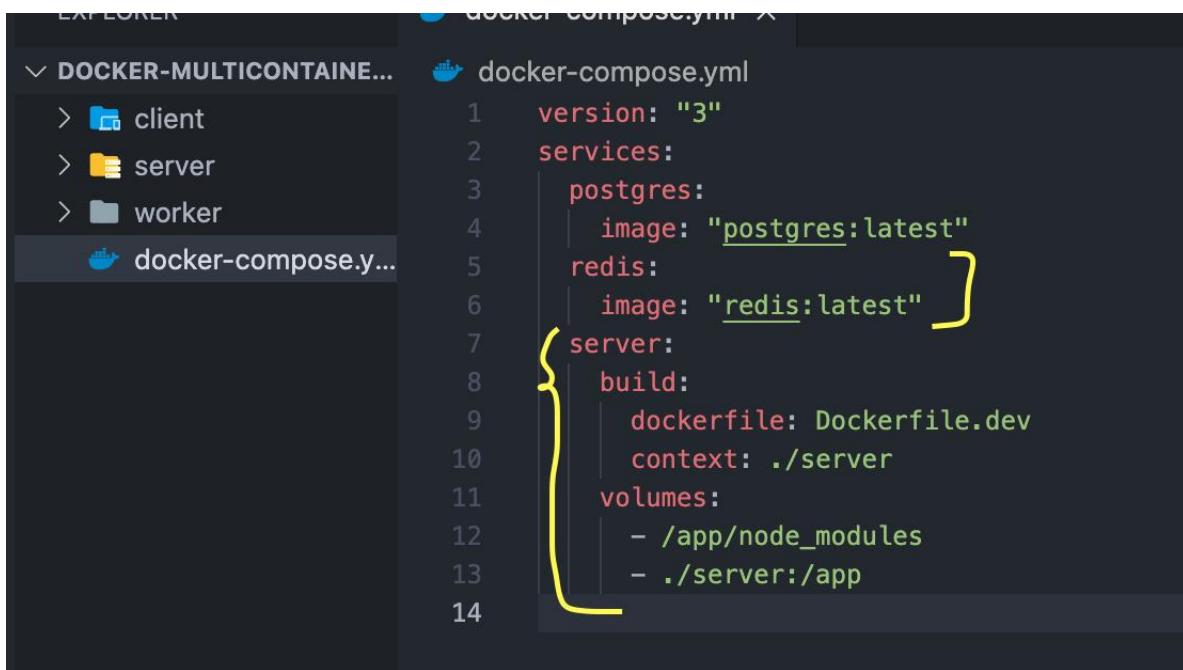


```

version: "3"
services:
  postgres:
    image: "postgres:latest"

```

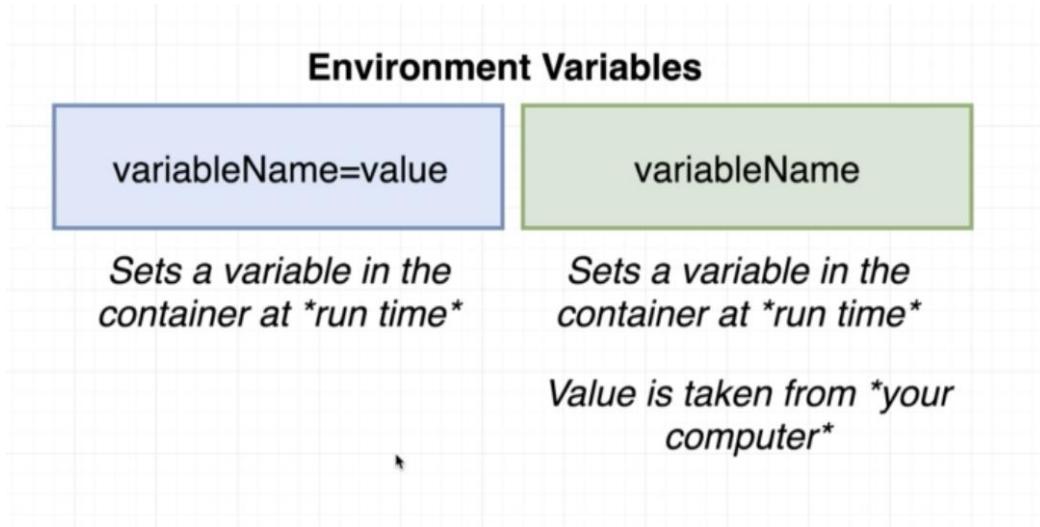
Docker compose config



```
version: "3"
services:
  postgres:
    image: "postgres:latest"
  redis:
    image: "redis:latest"
  server:
    build:
      dockerfile: Dockerfile.dev
      context: ./server
    volumes:
      - /app/node_modules
      - ./server:/app
```

Adding a redis service and server to docker compose.

Environment variable with docker compose



```
git docker-compose.yml
1 version: "3"
2 services:
3   postgres:
4     image: "postgres:latest"
5     environment:
6       - POSTGRES_PASSWORD=postgres_password
7   redis:
8     image: "redis:latest"
9   server:
10    build:
11      dockerfile: Dockerfile.dev
12      context: ./server
13    volumes:
14      - /app/node_modules
15      - ./server:/app
16    environment:
17      - REDIS_HOST=redis
18      - REDIS_PORT=6379
19      - PGUSER=postgres
20      - PGHOST=postgres
21      - PGDATABASE=postgres
22      - PGPASSWORD=postgres_password
23      - PGPORT=5432
24
```

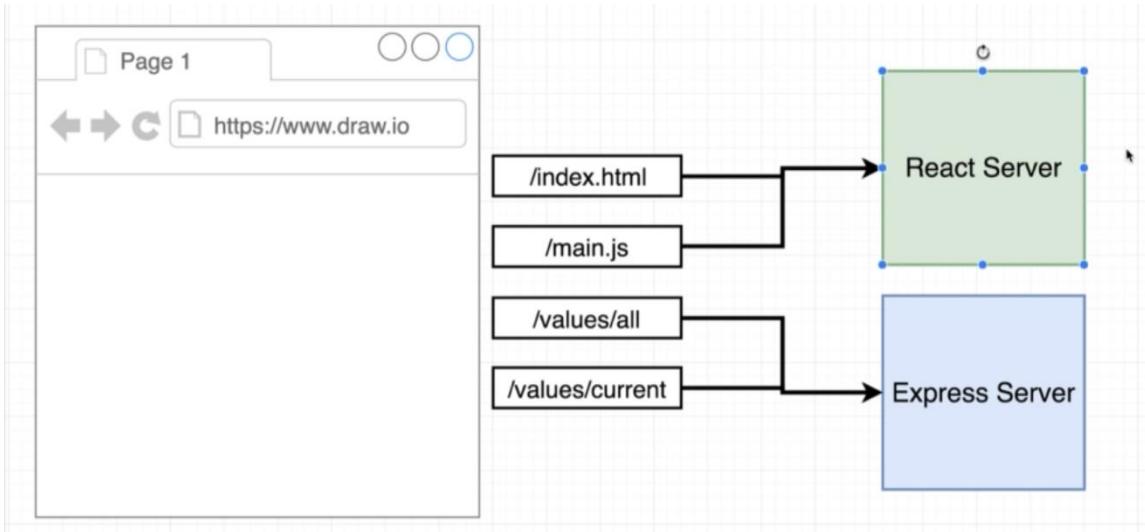
The worker and client services

```
git docker-compose.yml
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
```

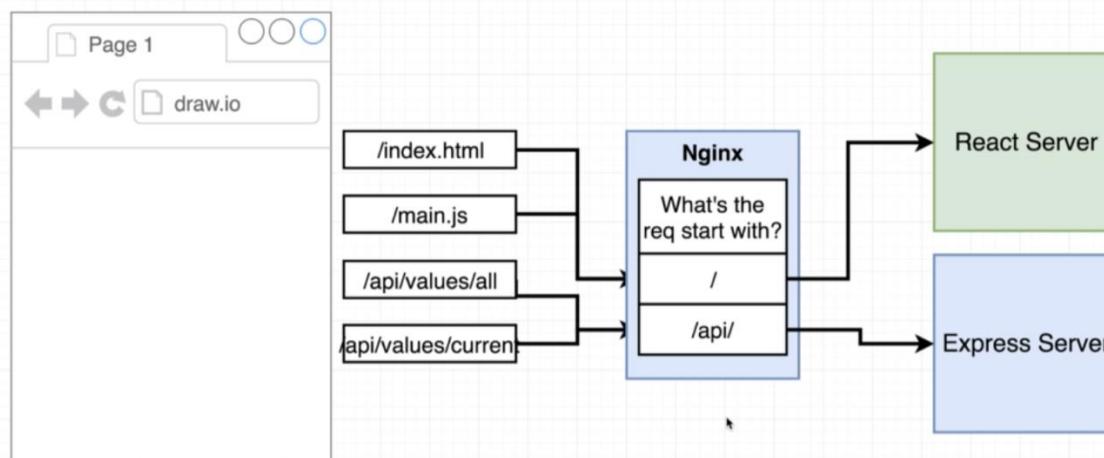
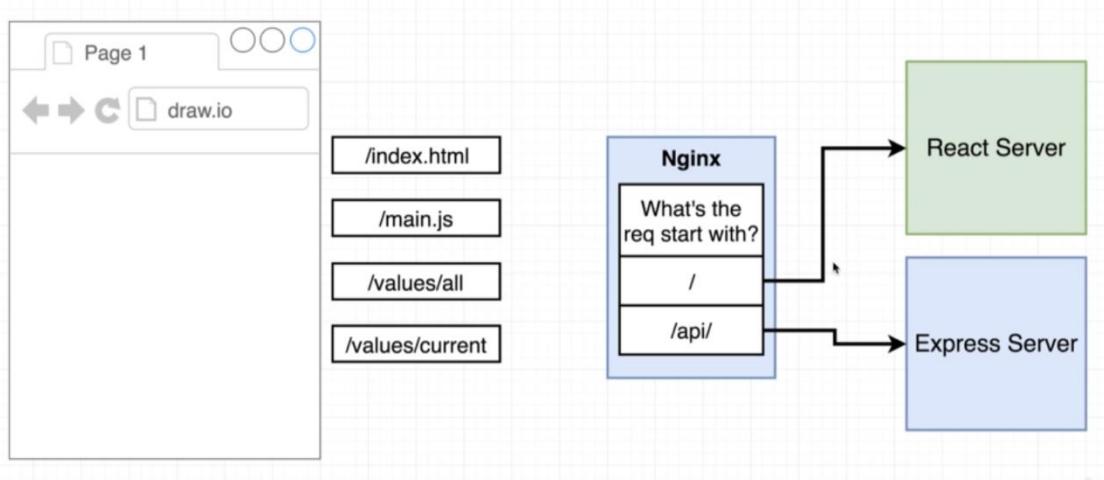


```
  - REDIS_PORT=6379
  - PGUSER=postgres
  - PGHOST=postgres
  - PGDATABASE=postgres
  - PGPASSWORD=postgres_password
  - PGPORT=5432
  client:
    stdin_open: true
    build:
      dockerfile: Dockerfile.dev
      context: ./client
    volumes:
      - /app/node_modules
      - ./server:/app
  worker:
    build:
      dockerfile: Dockerfile.dev
      context: ./worker
    volumes:
      - /app/node_modules
      - ./worker:/app
    environment:
      - REDIS_HOST=redis
      - REDIS_PORT=6379
```

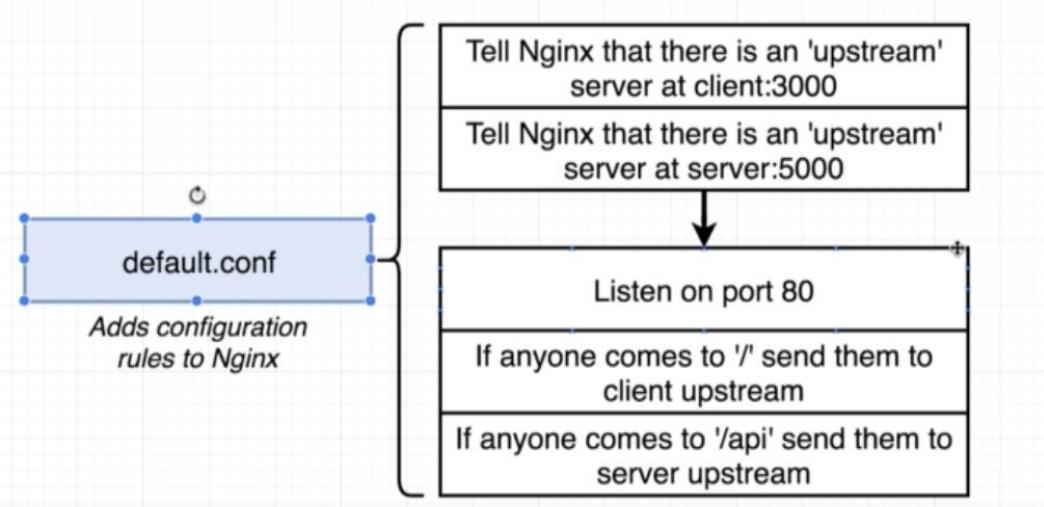
Nginx Path Routing



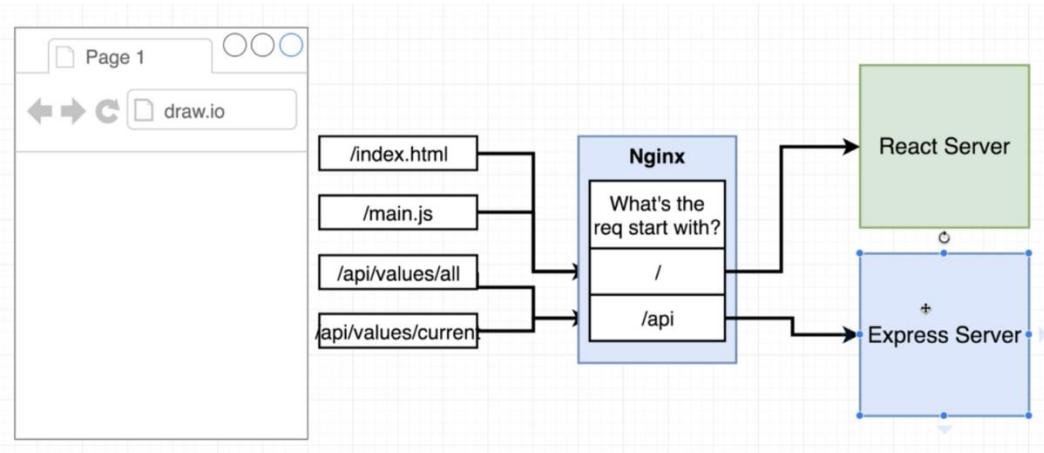
Nginx helps to route the request to right server.



If the request contains the api in url, request goes to express server. Otherwise, it goes to react server.



Adding a configuration file for nginx routing.



Nginx route the request to appropriate server.

```

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

```

Building a custom Nginx image

```
nginx > 🛠 Dockerfile.dev > ...
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      environment:
6        - POSTGRES_PASSWORD=postgres_password
7    nginx:
8      restart: always
9      build:
10        dockerfile: Dockerfile.dev
11        context: ./nginx
12      ports:
13        - "3050:80"
14    redis:
15      image: "redis:latest"
16    server:
17      build:
18        dockerfile: Dockerfile.dev
19        context: ./server
20      volumes:
21        - /app/node_modules
```

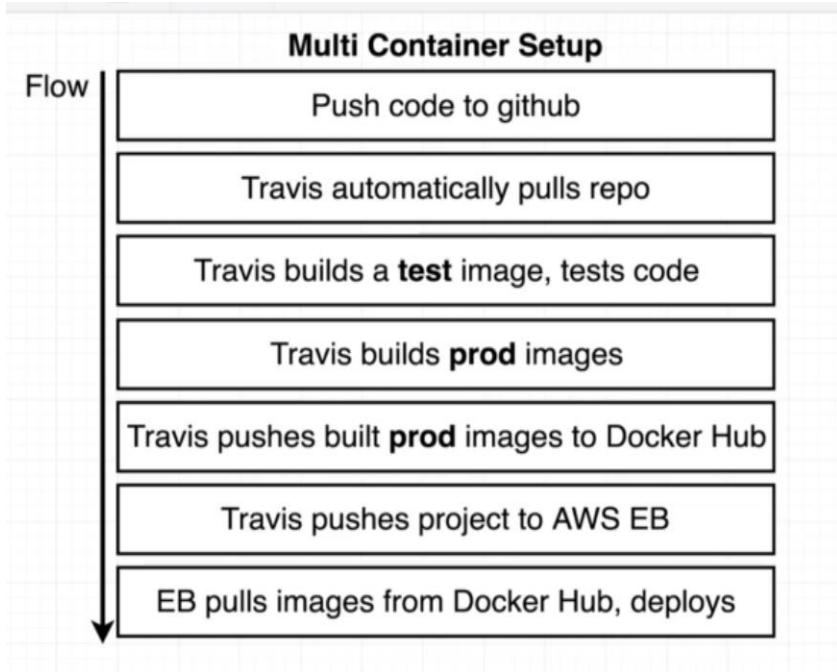


Starting up docker compose

```
TERMINAL
→ DOCKER-MULTICONTAINER-APP docker-compose up --build
Creating network "docker-multicontainer-app_default" with the default driver
Building nginx
[+] Building 4.4s (3/4)
=> [internal] load build definition from Dockerfile.dev          0.0s
=> => transferring dockerfile: 107B                            0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                0.0s
=> [internal] load metadata for docker.io/library/nginx:latest 4.3s
=> [auth] library/nginx:pull token for registry-1.docker.io   0.0s
```

A Continuous Integration Workflow for Multiple Images

Production Multi Container Deployment



We will be using the GitHub Actions in place of Travis.

Production Dockerfile

Worker dockerfile

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

Server Dockerfile

✓ DOCKER-MULTICONTAINER-APP

- > client
- > nginx
- ✓ server
 - > node_modules
 - Dockerfile
 - Dockerfile.dev
 - index.js
 - keys.js
 - package.json
- ✓ worker
 - > node_modules
 - Dockerfile
 - Dockerfile.dev

server > Dockerfile > COPY

```

1  FROM node:14.14.0-alpine
2  WORKDIR "/app"
3  COPY ./package.json ./
4  RUN npm install
5  COPY . .
6  CMD ["npm","run","start"]

```

Nginx Dockerfile

✓ DOCKER-MULTI... ▾

- > client
- ✓ nginx
 - default.conf
 - Dockerfile
 - Dockerfile.dev
- ✓ server
 - > node_modules
 - Dockerfile
 - Dockerfile.dev
 - index.js
 - keys.js

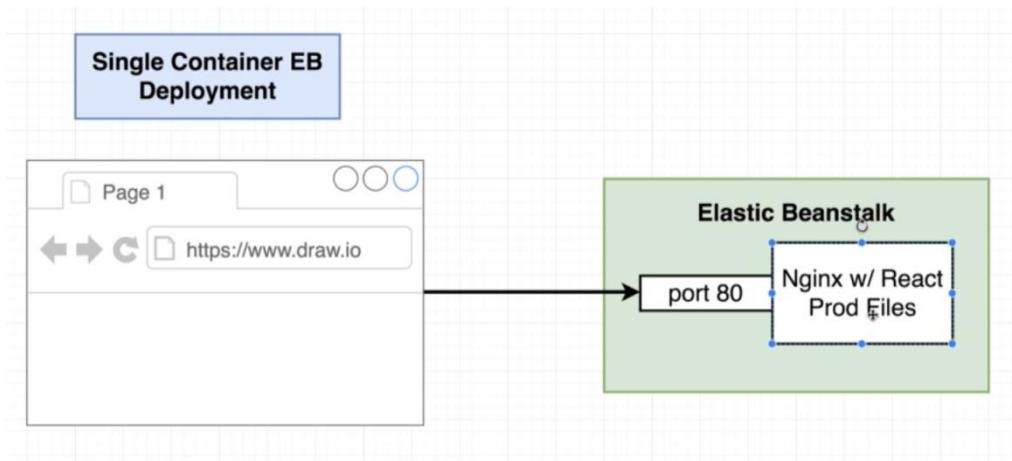
nginx > Dockerfile > ...

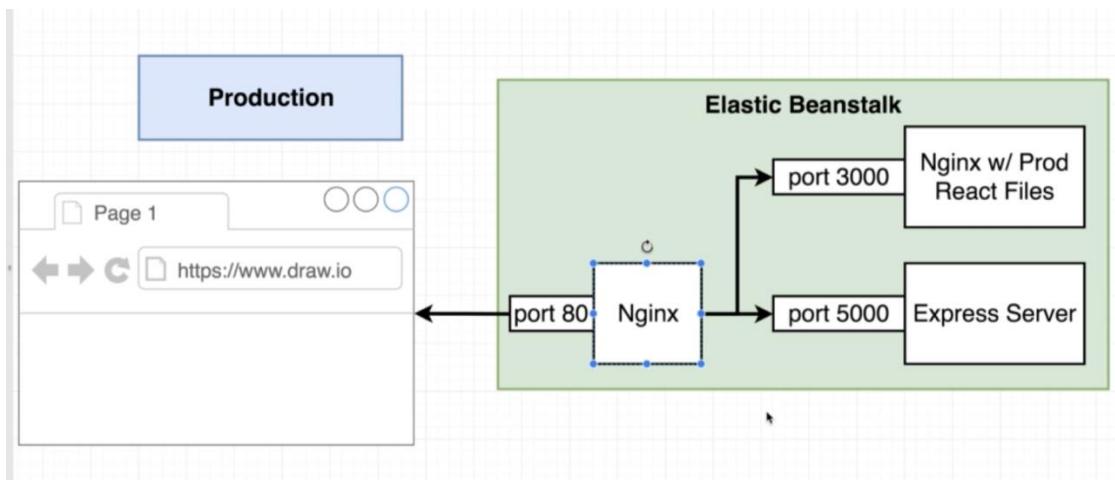
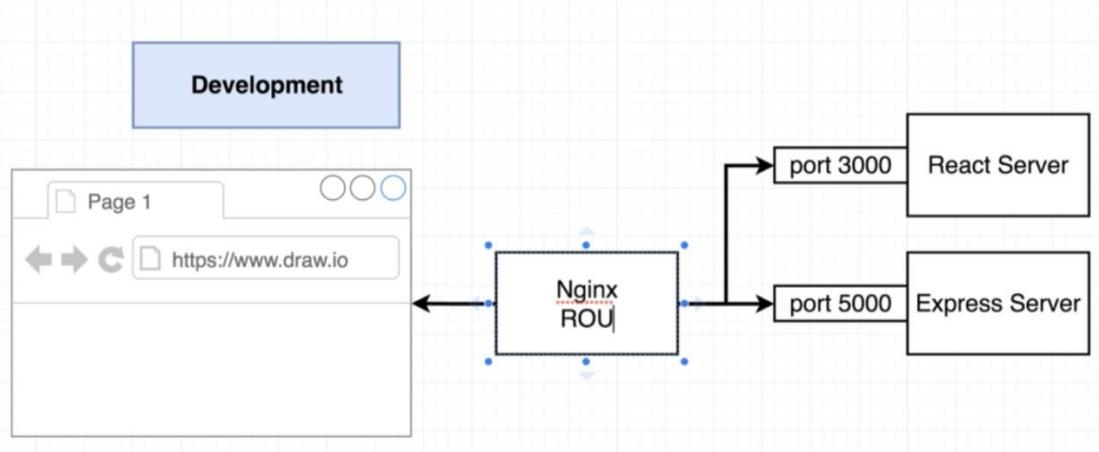
```

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

```

Mutliple Nginx Instances





Client Production Dockerfile

```

client > 🛠 Dockerfile > ...
1  FROM node:16-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  EXPOSE 3000
10 COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf
11 COPY --from=builder /app/build /use/share/nginx/html
12
  
```

Continuous Integration with Github Actions

```
.github > workflows > deploy.yaml
1   name: Deploy MultiDocker
2   < on:
3     < push:
4       < branches:
5         - main
6
7   < jobs:
8     < build:
9       runs-on: ubuntu-latest
10    < steps:
11      - uses: actions/checkout@v2
12      - run: docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
13      - run: docker build -t mnaveensmn/react-test -f ./client/Dockerfile.dev
14      - run: docker run -e CI=true mnaveensmn/react-test npm test
15
16      - run: docker build -t mnaveensmn/multi-client-10-14 ./client
17      - run: docker build -t mnaveensmn/multi-nginx-10-14 ./nginx
18      - run: docker build -t mnaveensmn/multi-server-10-14 ./server
19      - run: docker build -t mnaveensmn/multi-worker-10-14 ./worker
20
21      - run: docker push mnaveensmn/multi-client-10-14
22      - run: docker push mnaveensmn/multi-nginx-10-14
23      - run: docker push mnaveensmn/multi-server-10-14
24      - run: docker push mnaveensmn/multi-worker-10-14
25
26      - name: Generate deployment package
27        run: zip -r deploy.zip . -x '*.git*'
28
```

Successful Image Building

✓ added the github actions workflow script Deploy MultiDocker #1

Re-run all jobs Latest #2 ...

Summary

Jobs

build

build succeeded 3 minutes ago in 2m 34s

Search logs

Step	Description	Duration
> ✓ Set up job		1s
> ✓ Run actions/checkout@v2		1s
> ✓ Run docker login -u *** -p ***		0s
> ✓ Run docker build -t ***/react-test -f ./client/Dockerfile.dev ./client		1m 20s
> ✓ Run docker run -e CI=true ***/react-test npm test		4s
> ✓ Run docker build -t ***/multi-client-10-14 ./client		20s
> ✓ Run docker build -t ***/multi-nginx-10-14 ./nginx		1s
> ✓ Run docker build -t ***/multi-server-10-14 ./server		14s
> ✓ Run docker build -t ***/multi-worker-10-14 ./worker		18s
> ✓ Run docker push ***/multi-client-10-14		3s
> ✓ Run docker push ***/multi-nginx-10-14		2s
> ✓ Run docker push ***/multi-server-10-14		3s
> ✓ Run docker push ***/multi-worker-10-14		4s
> ✓ Generate deployment package		0s
> ✓ Post Run actions/checkout@v2		0s
> ✓ Complete job		0s

Build image has been pushed to docker repository

The screenshot shows a web-based Docker registry interface. At the top, there is a blue header bar with the GitHub logo, a search bar containing "Search for great content", navigation links for "Explore", "Repositories", "Organizations", "Help", and a yellow "Upgrade" button. Below the header, the user's profile "mnaveensmn" is displayed, along with a dropdown arrow and a search bar for "Search by repository name". A prominent blue "Create Repository" button is located on the right.

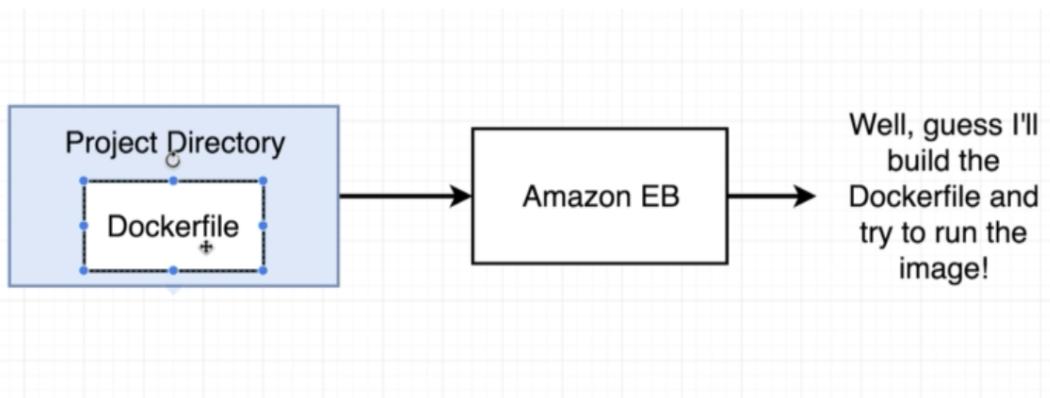
Below the user profile, four repository cards are listed:

- mnaveensmn / multi-worker-10-14**
Last pushed: 5 minutes ago
Not Scanned | 0 stars | 0 downloads | Public
- mnaveensmn / multi-server-10-14**
Last pushed: 5 minutes ago
Not Scanned | 0 stars | 0 downloads | Public
- mnaveensmn / multi-nginx-10-14**
Last pushed: 5 minutes ago
Not Scanned | 0 stars | 0 downloads | Public
- mnaveensmn / multi-client-10-14**
Last pushed: 5 minutes ago
Not Scanned | 0 stars | 0 downloads | Public

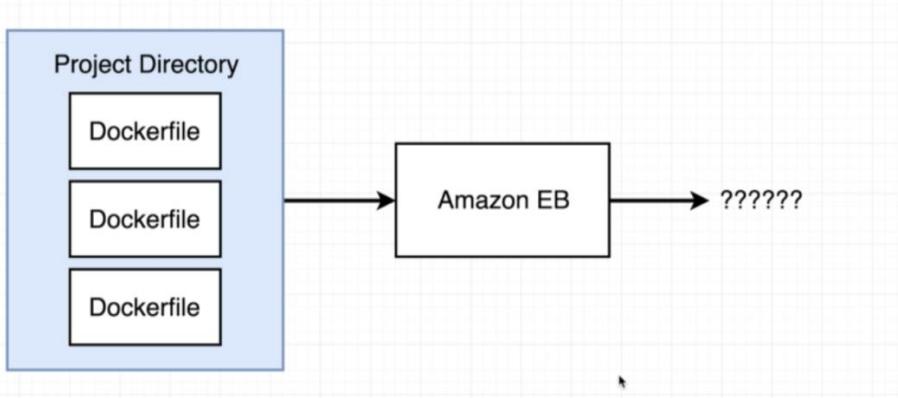
Multi Container Deployment to AWS

Multi container Definition Files

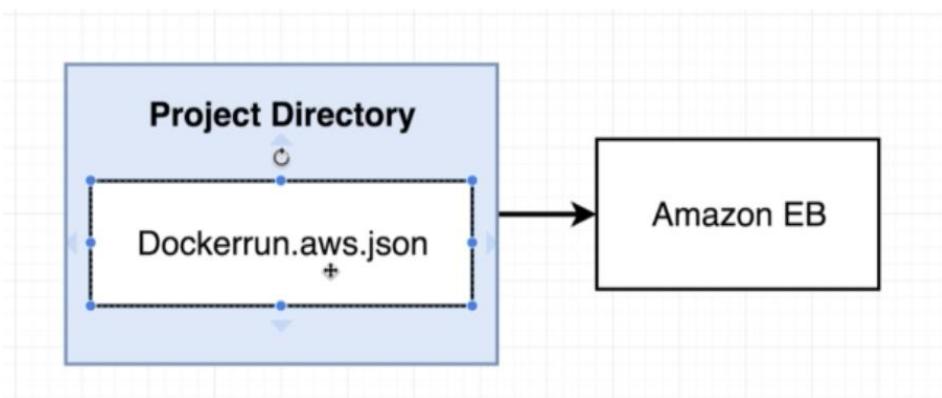
Single container scenario.



This time around we have a multi container.



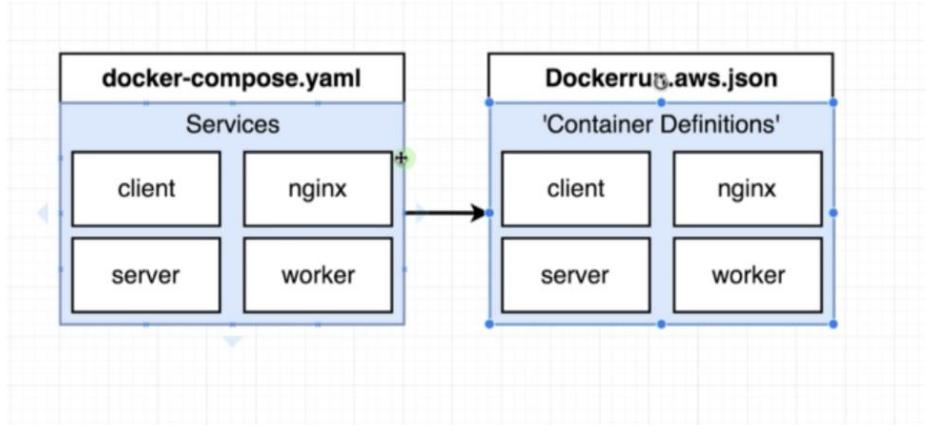
We are going to create a special file inside the project directory.



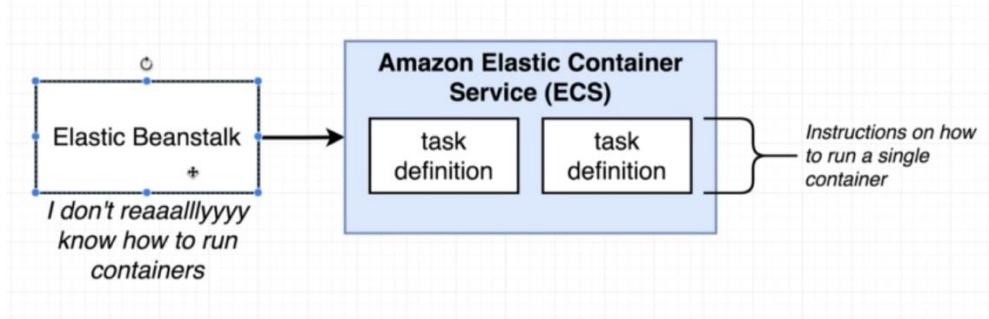
Dockerrun.aws.json file going to tell Amazon EB that how to work with multiple containers by specifying the information below.

1. Where to pull the image from?
2. What resource to allocate to each image?
3. How to set up a port mapping?
4. And some associated information.

It is like docker compose yaml file. The difference is, in docker compose file, we are giving the instruction to build the image from dockerfile. In dockerrun.aws.json file we are giving the instruction to where to pull the image from.



Elastic Beanstalk by default does not know the how to work with multiple containers.



ECS has definition file that contains instruction to run a single container.

[All](#) [Shopping](#) [Images](#) [News](#) [Videos](#) [More](#)

Settings Tools

About 125,000 results (0.45 seconds)

The **AWS::ECS::TaskDefinition** resource describes the **container** and **volume definitions** of an **Amazon Elastic Container Service (Amazon ECS) task**. You can specify which Docker images to use, the required resources, and other configurations related to launching the **task definition** through an **Amazon ECS** service or **task**.

AWS::ECS::TaskDefinition - AWS CloudFormation - AWS Documentation
<https://docs.aws.amazon.com/AWSCloudFormation/.../aws-resource-ecs-taskdefinition.htm...>

[About this result](#) [Feedback](#)

[Amazon ECS Task Definitions - Amazon Elastic Container Service](#)

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definitions.html ▾

Amazon ECS Task Definitions. A task definition is required to run Docker containers in Amazon ECS. Some of the parameters you can specify in a task definition include: The Docker images to use with the containers in your task. ... The Docker networking mode to use for the containers in your task.

[Task Definition Parameters](#)

The following task definition

[Example Task Definitions](#)

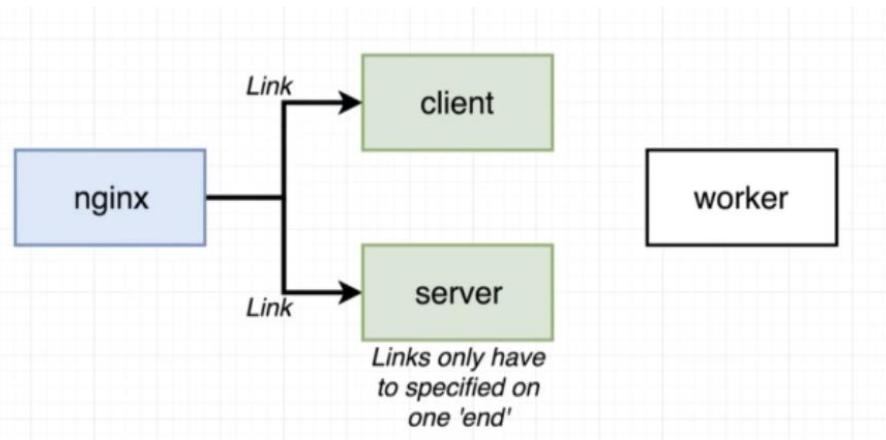
Example Task Definitions. Below are

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html

If essential container crashes, all other containers will be shut down.

At least one container marker as essential true.

Forming Container Links



```
{..} Dockerrun.aws.json > [ ] containerDefinitions > {} 3
```

```
1   {
2     "AWSEBDockerrunVersion": 2,
3     "containerDefinitions": [
4       {
5         "name": "client",
6         "image": "mnaveensmn/multi-client",
7         "hostname": "client",
8         "essential": false
9       },
10      {
11        "name": "server",
12        "image": "mnaveensmn/multi-server",
13        "hostname": "api",
14        "essential": false
15      },
16      {
17        "name": "worker",
18        "image": "mnaveensmn/multi-worker",
19        "hostname": "worker",
20        "essential": false
21      },
22      {
23        "name": "ngix",
24        "image": "mnaveensmn/multi-nginx",
25        "hostname": "nginx",
26        "essential": true,
27        "portMappings": [
28          {
29            ...
30          }
31        ],
32        "links": ["client", "server"]
33      }
34    ]
35  }
36}
37}
```

Using Amazon Linux 2 Platform

Create environment using the new platform

1. Create environment using the new platform

When creating our Elastic Beanstalk environment we need to select **Docker running on 64bit Amazon Linux 2**

Platform

Managed platform
Platforms published and maintained by Amazon Elastic Beanstalk. [Learn more](#)

Custom platform
Platforms created and owned by you.

Platform

Docker

Platform branch

Docker running on 64bit Amazon Linux 2

Platform version

3.4.4 (Recommended)

Rename the current docker-compose file

2. Rename the current docker-compose file

Rename the **docker-compose.yml** file to **docker-compose-dev.yml**. Going forward you will need to pass a flag to specify which compose file you want to build and run from:

```
docker-compose -f docker-compose-dev.yml up
docker-compose -f docker-compose-dev.yml up --build
docker-compose -f docker-compose-dev.yml down
```

Create a production only docker-compose.yml file

3. Create a production only docker-compose.yml file

The production compose file will follow closely what was written in the Dockerrun.aws.json. There are two major differences:

No Container Links: In the "["Forming Container Links"](#)" lecture we add the client and server services to the links array of the nginx service. Docker Compose will handle this container communication automatically for us.

Environment Variables: When using a compose file we will need to explicitly specify the environment variables each service will need access to. The value for each variable must match the corresponding variable names you have specified in the Elastic Beanstalk environment. The AWS variables are created in the "["Setting Environment Variables"](#)" lecture.

Note - You must not have a Dockerrun.aws.json file in your project directory. If AWS EBS sees this file the deployment will fail. If you have previously followed this course and deployed to the old Multi-container platform **you will need to delete this file before moving to the new platform!!!**

Complete docker-compose.yml file:

```
1 | version: "3"
2 | services:
3 |   client:
4 |     image: "cygnetops/multi-client-10-14"
5 |     mem_limit: 128m
6 |     hostname: client
7 |   server:
8 |     image: "cygnetops/multi-server-10-14"
9 |     mem_limit: 128m
10 |    hostname: api
11 |    environment:
12 |      - REDIS_HOST=$REDIS_HOST
13 |      - REDIS_PORT=$REDIS_PORT
14 |      - PGUSER=$PGUSER
15 |      - PGHOST=$PGHOST
16 |      - PGDATABASE=$PGDATABASE
17 |      - PGPASSWORD=$PGPASSWORD
18 |      - PGPORT=$PGPORT
19 |   worker:
20 |     image: "cygnetops/multi-worker-10-14"
21 |     mem_limit: 128m
22 |     hostname: worker
23 |     environment:
24 |       - REDIS_HOST=$REDIS_HOST
25 |       - REDIS_PORT=$REDIS_PORT
26 |   nginx:
27 |     image: "cygnetops/multi-nginx-10-14"
28 |     mem_limit: 128m
29 |     hostname: nginx
30 |     ports:
31 |       - "80:80"
```

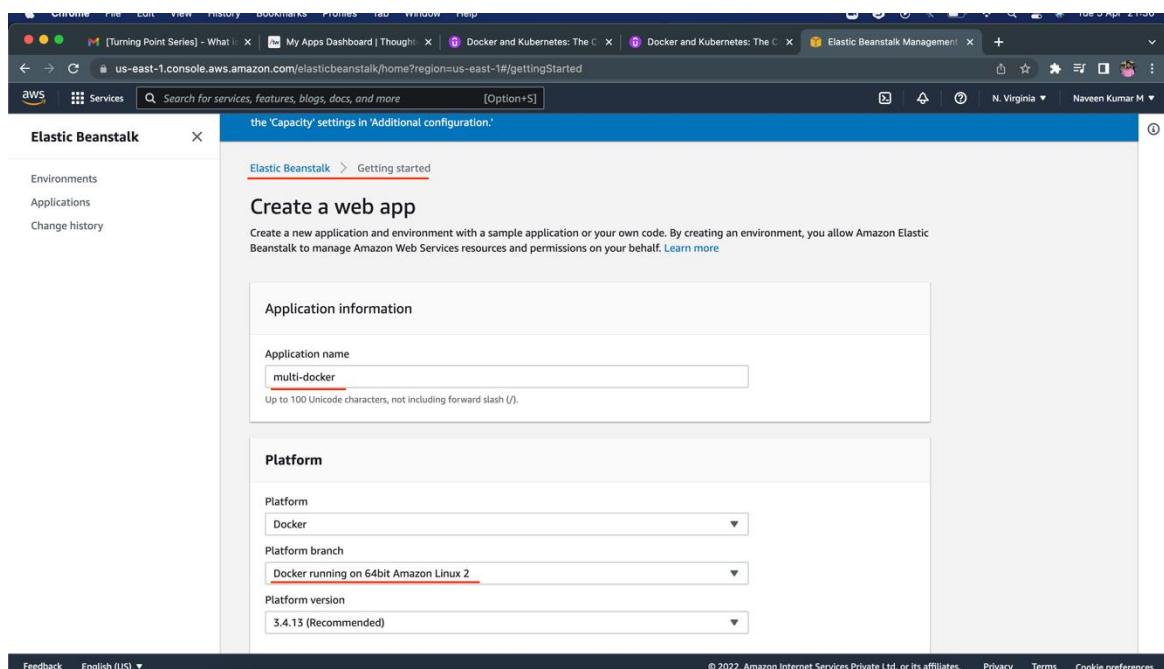
Creating the EB Environment

Steps

EBS Application Creation (If using Amazon Linux 2 Platform Platform)

1. Make sure you have followed the guidance in this [note](#).
2. Go to AWS Management Console and use Find Services to search for Elastic Beanstalk
3. Click "Create Application"
4. Set Application Name to 'multi-docker'
5. **Scroll down to Platform and select Docker**
6. The Platform Branch should be automatically set to **Docker Running on 64bit Amazon Linux 2.**
7. Click Create Application
8. You may need to refresh, but eventually, you should see a green checkmark underneath Health.

Screenshots



Platform

Platform

Docker

Platform branch

Docker running on 64bit Amazon Linux 2

Platform version

3.4.13 (Recommended)

Application code

Sample application
Get started right away with sample code.

Upload your code
Upload a source bundle from your computer or copy one from Amazon S3.

Cancel Configure more options **Create application**

us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/launchEnvironment?applicationName=multi-docker&environmentId=e-rm3gkdgkrf

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Elastic Beanstalk Environments Applications Change history

multi-docker Application versions Saved configurations

Multidocker-env

AWS Graviton now supported
AWS Graviton, an arm64-based processor, can offer up to 40% better price performance over the comparable x86 processor. To upgrade to an arm64 instance type, choose it in the 'Capacity' settings in 'Additional configuration.'

Elastic Beanstalk > Environments > Multidocker-env

Creating Multidocker-env
This will take a few minutes.

9:37pm Using elasticbeanstalk-us-east-1-218788053233 as Amazon S3 storage bucket for environment data.
9:37pm createEnvironment is starting.

Elastic Beanstalk > Environments > Multidocker-env

Multidocker-env
Multidocker-env.eba-ziwmpoj.us-east-1.elasticbeanstalk.com (e-rm3gkdgkrf)
Application name: multi-docker

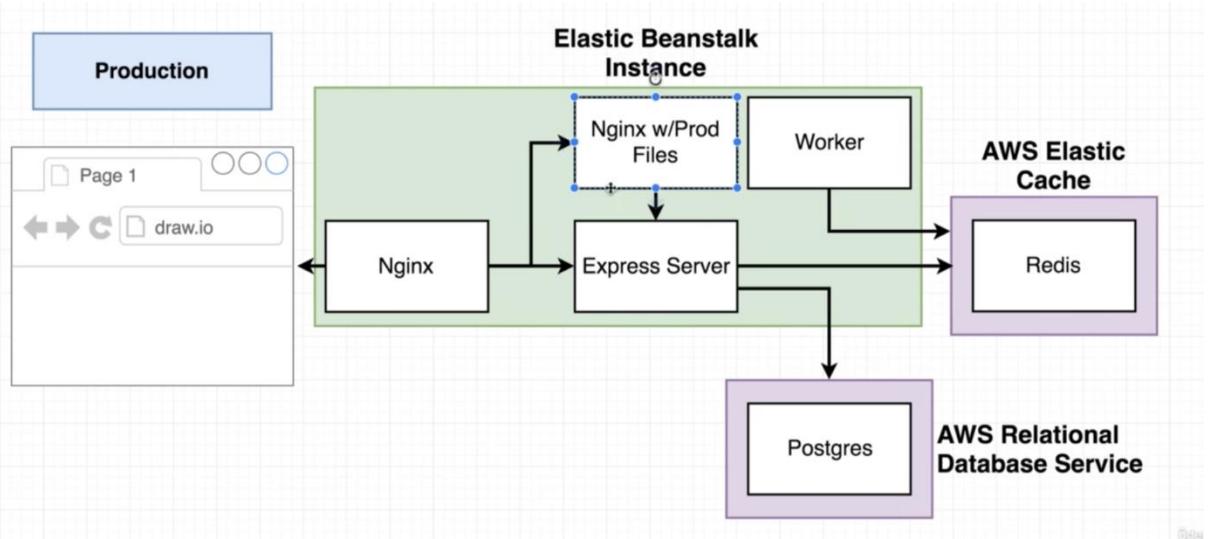
Health **Running version** **Platform**

Ok **Sample Application** Docker running on 64bit Amazon Linux 2/3.4.13

Causes **Upload and deploy** **Change**

Recent events **Show all**

Architecture in Production Environment



AWS Elastic Cache

Automatically creates and maintains Redis instances for you

Super easy to scale

Built in logging + maintenance

Probably better security than what we can do

Easier to migrate off of EB with

AWS Relational Database Service

Automatically creates and maintains Postgres instances for you

Super easy to scale

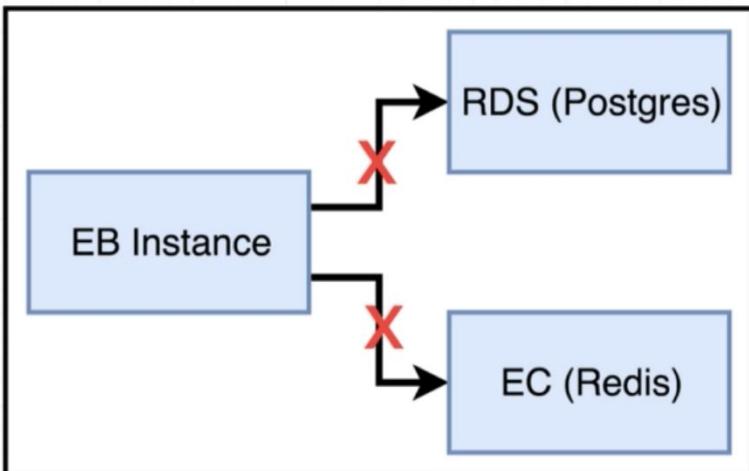
Built in logging + maintenance

Probably better security than what we can do

Automated backups and rollbacks

Easier to migrate off of EB with

Overview of AWS VPC's and Security Group



By default these services can't talk to each other

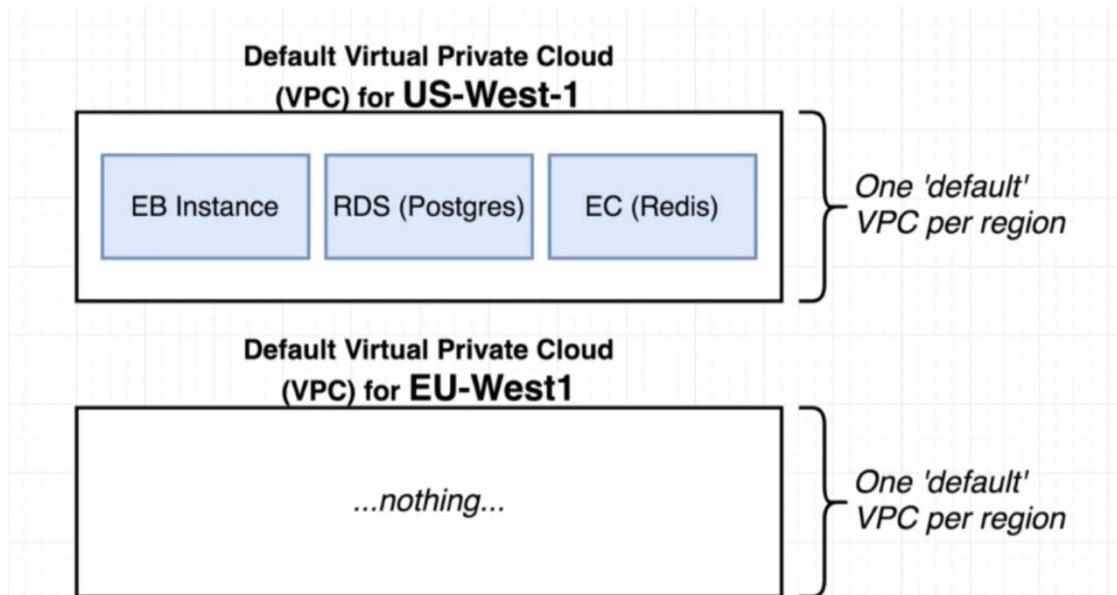
VPC

VPC > Your VPCs > vpc-0d0d5d4c21e769367

vpc-0d0d5d4c21e769367

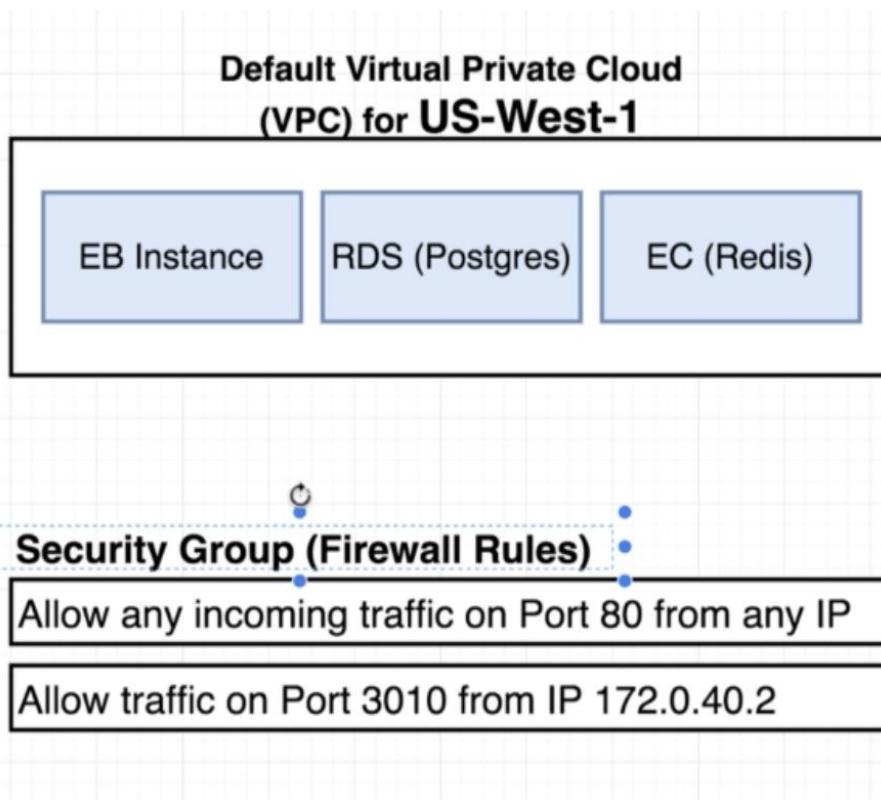
Actions ▾

Details		Info	
VPC ID	vpc-0d0d5d4c21e769367	State	Available
Tenancy		DNS hostnames	Enabled
Default		Main route table	rtb-0e7ff1dc07591ae1a
Default VPC		IPv4 CIDR	172.31.0.0/16
Yes		IPv6 pool	-
Route 53 Resolver DNS Firewall rule groups	-	Owner ID	218788053233
		DNS resolution	Enabled
		Main network ACL	acl-0530c1534c957b9b6
		IPv6 CIDR (Network border group)	-



Security group

To get the different service to connect to each other, we need to create the Security group.



AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

New VPC Experience Tell us what you think

Cloud Your VPCs Subnets Route Tables Internet Gateways Egress Only Internet Gateways Carrier Gateways DHCP Options Sets Elastic IPs Managed Prefix Lists Endpoints New Endpoint Services NAT Gateways Peering Connections

SECURITY Network ACLs **Security Groups**

NETWORK ANALYSIS Reachability Analyzer Network Access Analyzer

DNS FIREWALL

Security Groups (1/3) Info

Name	Security group ID	Security group name	VPC ID	Description
-	sg-06c52a3b446492146	default	vpc-0d0d5d4c21e769367	default VPC security group
<input checked="" type="checkbox"/> Multidocker-env	sg-0d0c442938a636007	awseb-e-rm3gkdgkrf-stack-AWSEBSecurityGrou...	vpc-0d0d5d4c21e769367	SecurityGroup for ElasticBeanstalk env
<input checked="" type="checkbox"/> Multidocker-env	sg-0f35fef4442782098	awseb-e-rm3gkdgkrf-stack-AWSEBLoadBalancer...	vpc-0d0d5d4c21e769367	Elastic Beanstalk created security group

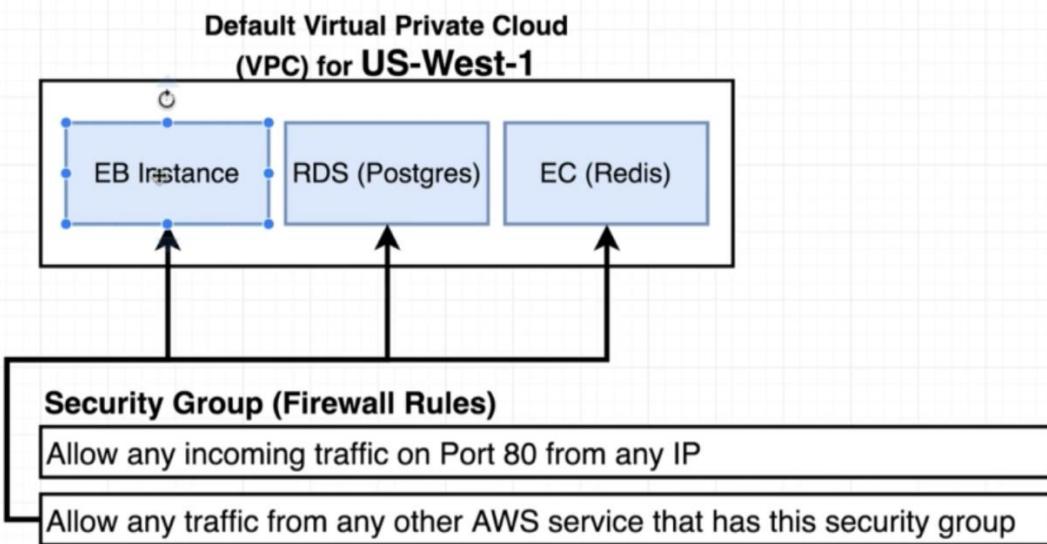
sg-0f35fef4442782098 - awseb-e-rm3gkdgkrf-stack-AWSEBLoadBalancerSecurityGroup-1CB2NV06JVVKY

Details Inbound rules Outbound rules Tags

You can now check network connectivity with Reachability Analyzer Run Reachability Analyzer

Inbound rules (1/1)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
<input checked="" type="checkbox"/> -	sgr-05c3944399470ed...	IPv4	HTTP	TCP	80	0.0



RDS Database Creation

Steps

RDS Database Creation

1. Go to AWS Management Console and use Find Services to search for RDS
2. Click Create database button
3. Select PostgreSQL
4. Change Version to the newest available v12 version (The free tier is currently not available for Postgres v13)
5. In Templates, check the Free tier box.
6. Scroll down to Settings.
7. Set DB Instance identifier to **multi-docker-postgres**
8. Set Master Username to **postgres**
9. Set Master Password to **postgrespassword** and confirm.
10. Scroll down to Connectivity. Make sure VPC is set to Default VPC
11. Scroll down to Additional Configuration and click to unhide.
12. Set Initial database name to **fibvalues**
13. Scroll down and click Create Database button

Screenshots

us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#

Amazon RDS

Dashboard Databases Query Editor Performance insights Snapshots Automated backups Reserved instances Proxies Subnet groups Parameter groups Option groups Custom engine versions Events Event subscriptions Recommendations 0 Certificate update

Try the new Amazon RDS Multi-AZ deployment option for MySQL and PostgreSQL
For your Amazon RDS for MySQL and PostgreSQL workloads, improve transactional commit latencies by 2x, experience faster failover typically less than 35 seconds and, get read scalability with two readable standby DB instances by deploying the Multi-AZ DB cluster [Learn more](#)

Create database Or, Restore Multi-AZ DB Cluster from Snapshot

Resources

You are using the following Amazon RDS resources in the US East (N. Virginia) region (used/quota)

DB Instances (0/40)	Parameter groups (0)
Allocated storage (0 TB/100 TB)	Default (0)
Click here to increase DB instances limit	
DB Clusters (0/40)	Custom (0/100)
Reserved instances (0/40)	Option groups (0)
Snapshots (0)	Default (0)
Manual (0/100)	Custom (0/20)
Automated (0)	Subnet groups (0/50)
Recent events (0)	Supported platforms VPC
Event subscriptions (0/20)	Default network vpc-0d0d5d4c21e769367

Additional information

Getting started with RDS Overview and features Documentation Articles and tutorials Data import guide for MySQL Data import guide for Oracle Data import guide for SQL Server New RDS feature announcements Pricing Forums

Database Preview Environment

Get early access to new DB engine versions, before they're generally available. The RDS database preview environment lets you work with upcoming

Engine options

Engine type [Info](#)

Amazon Aurora 

PostgreSQL 

MySQL 

MariaDB 

Oracle 

Microsoft SQL Server 

Version

PostgreSQL 12.10-R1 [Info](#)

Templates

Choose a sample template to meet your use case.

Production
Use defaults for high availability and fast, consistent performance.

Dev/Test
This instance is intended for development use outside of a production environment.

Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. [Info](#)

Settings

DB instance identifier [Info](#)

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

multi-docker-postgres

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

postgres

1 to 16 alphanumeric characters. First character must be a letter.

Auto generate a password

Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double quote) and @ (at sign).

Confirm password [Info](#)

Connectivity



Virtual private cloud (VPC) [Info](#)

VPC that defines the virtual networking environment for this DB instance.

Default VPC (vpc-0d0d5d4c21e769367)



Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

Subnet group [Info](#)

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

default



Public access [Info](#)

Yes

Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

No

RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

VPC security group

Choose a VPC security group to allow access to your database. Ensure that the security group rules allow the appropriate incoming traffic.

Choose existing

Choose existing VPC security groups

Create new

Create new VPC security group

▼ Additional configuration

Database options, encryption enabled, backup enabled, backtrack disabled, Performance Insights enabled, Enhanced Monitoring disabled, maintenance, CloudWatch Logs, delete protection disabled.

Database options

Initial database name [Info](#)

fibvalues

If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group [Info](#)

default.postgres12



Option group [Info](#)

default:postgres-12



us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#launch-dbinstance:gdb=false;isHermesCreate=true;s3-import=false

Services Search for services, features, blogs, docs, and more [Option+S]

Maintenance window [Info](#)
Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.
 Select window
 No preference

Deletion protection
 Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

[Learn more about AWS Free Tier.](#)

When your free usage expires or if your application usage exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page](#).

ⓘ You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

[Cancel](#) [Create database](#)

Creating database multi-docker-postgres

Your database might take a few minutes to launch.

[View credential details](#) [X](#)

RDS > Databases

Databases [Group resources](#) [Modify](#) [Actions ▾](#) [Restore from S3](#) [Create database](#)

Filter by databases [1](#)

DB identifier	Role	Engine	Region & AZ	Size	Status	CPU	Current az
multi-docker-postgres	Instance	PostgreSQL	-	db.t3.micro	Creating	-	

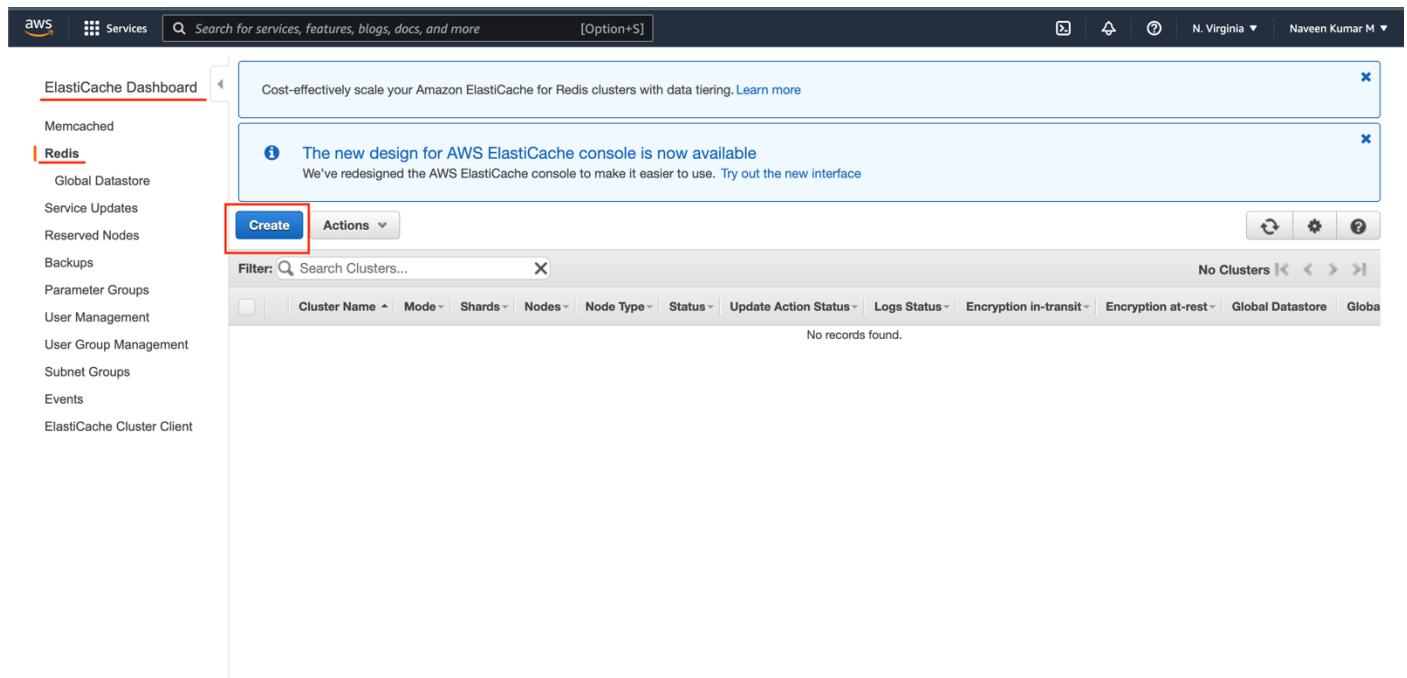
ElastiCache Redis Creation

Steps

ElastiCache Redis Creation

1. Go to AWS Management Console and use Find Services to search for ElastiCache
2. Click Redis in sidebar
3. Click the Create button
4. **Make sure Cluster Mode Enabled is NOT ticked**
5. In Redis Settings form, set Name to multi-docker-redis
6. Change Node type to 'cache.t2.micro'
7. Change Replicas per Shard to 0
8. Scroll down and click Create button

Screenshots



us-east-1.console.aws.amazon.com|elasticache/home?region=us-east-1#create-cluster:ct=redis:

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Create your Amazon ElastiCache cluster

- Cluster engine **Redis**
In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers Multi-AZ with Auto-Failover and enhanced robustness.
- Cluster Mode enabled**
- Memcached**
High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.

Location

Choose a location

- Amazon Cloud**
Use Amazon's cloud for your ElastiCache instances
- On-Premises**
Create your ElastiCache instances on AWS Outposts. You need to create a subnet ID on an Outpost first.

Redis settings

Ensure you have reviewed the five workload characteristics to consider when right sizing Amazon ElastiCache Redis clusters. [Learn more](#)

Name

Description

us-east-1.console.aws.amazon.com|elasticache/home?region=us-east-1#create-cluster:ct=redis:

Amazon Cloud
Use Amazon's cloud for your ElastiCache instances

On-Premises
Create your ElastiCache instances on AWS Outposts. You need to create a subnet ID on an Outpost first.

Select node type

Instance family	r6g	r6gd	m6g	t4g	r5	m5	r4	m4	r3	m3	t3
Node type	cache.t2.micro										
Memory (GiB)	0.5										
Network performance	Low to moderate										

Cancel Save

Parameter group default.redis6.x

Node type cache.r6g.large (13.07 GiB)

Number of replicas 2

Multi-AZ

Advanced Redis settings

us-east-1.console.aws.amazon.com|elasticache/home?region=us-east-1#create-cluster:ct=redis:

Engine version compatibility 6.2

Port 6379

Parameter group default.redis6.x

Node type cache.t2.micro (0.5 GiB)

Number of replicas 0

Multi-AZ
Multi-AZ can not be enabled when the number of replicas is set to 0. Select one or more replicas to enable Multi-AZ. [Learn more](#)

Advanced Redis settings

Subnet group is mandatory setting.

Subnet group: Create new

Name: multi-docker-subnet-group

Description: Description

VPC ID: vpc-0d0d5d4c21e769367

Subnet ID	Availability zone	CIDR Block
subnet-0cbdebab9172a127a	us-east-1c	172.31.80.0/20
subnet-02d07c707ad3c2729	us-east-1d	172.31.16.0/20
subnet-091229cf276ea46ea	us-east-1e	172.31.48.0/20
subnet-0a2bb1acd7b074622	us-east-1a	172.31.32.0/20
subnet-058bb8c04659c249b	us-east-1b	172.31.0.0/20
subnet-09dd5352f97705811	us-east-1f	172.31.64.0/20

Enable automatic backups:

Backup retention period: 1 day(s)

Backup window: No preference
 Specify backup window

Maintenance window: No preference
 Specify maintenance window

Topic for SNS notification: Disable notifications

Key	Value
Add key	Empty value

Create

The screenshot shows the AWS ElastiCache Redis console. On the left, there's a sidebar with options like Memcached, Redis, Global Datastore, Service Updates, Reserved Nodes, Backups, Parameter Groups, User Management, User Group Management, Subnet Groups, Events, and ElastiCache Cluster Client. The Redis section is selected. A banner at the top right says 'The new design for AWS ElastiCache console is now available'. The main area shows a table with one cluster entry:

	Cluster Name	Mode	Shards	Nodes	Node Type	Status	Update Action Status	Logs Status	Encryption in-transit	Encryption at-rest	Global Datastore
	multi-docker-redis	Redis	0	1 node	cache.t2.micro	creating	up to date	disabled	No	No	-

Creating a Custom Security Group

Steps

Creating a Custom Security Group

1. Go to AWS Management Console and use Find Services to search for VPC
2. Find the Security section in the left sidebar and click Security Groups
3. Click Create Security Group button
4. Set Security group name to multi-docker
5. Set Description to multi-docker
6. Make sure VPC is set to default VPC
7. Scroll down and click the Create Security Group button.
8. After the security group has been created, find the Edit inbound rules button.
9. Click Add Rule
10. Set Port Range to 5432-6379
11. Click in the box next to Source and start typing 'sg' into the box. Select the Security Group you just created.
12. Click the Save rules button

Screenshots

This screenshot shows the AWS VPC Security Groups list page. On the left, there's a navigation sidebar with various VPC-related options like New VPC Experience, DHCP Options Sets, Elastic IPs, Managed Prefix Lists, Endpoints, Endpoint Services, NAT Gateways, Peering Connections, SECURITY (Network ACLs, Security Groups), NETWORK ANALYSIS (Reachability Analyzer, Network Access Analyzer), DNS FIREWALL (Rule Groups, Domain Lists), and NETWORK FIREWALL (Firewalls, Firewall Policies, Network Firewall Rule Groups). The 'Security Groups' section is currently selected. The main area displays a table of security groups with columns: Name, Security group ID, Security group name, VPC ID, Description, and Owner. There are three entries: 'default' (sg-06c52a3b446492146), 'Multidocker-env' (sg-0d0c442938a636007), and another 'Multidocker-env' entry (sg-0f35fe4442782098). A red box highlights the 'Create security group' button at the top right of the table.

This screenshot shows the 'Create security group' wizard. The first step, 'Basic details', is displayed. It has fields for 'Security group name' (multi-docker), 'Description' (multi-docke), and 'VPC' (vpc-0d0d5d4c21e769367). Below this, the 'Inbound rules' step shows a message: 'This security group has no inbound rules.' A 'Add rule' button is available. The 'Outbound rules' step is partially visible at the bottom.

This screenshot shows the 'Create security group' wizard. The 'Outbound rules' step is the active one. It includes filters for Type (All traffic), Protocol (All), Port range (All), Destination (Custom, 0.0.0.0/0), and a 'Delete' button. A 'Add rule' button is also present. At the bottom, there's a 'Tags - optional' section with a note about tags and a 'Add new tag' button. A red box highlights the 'Create security group' button at the bottom right.

Inbound security group rules successfully modified on security group (sg-0354ea177013e5427 | multi-docker)

Details

VPC > Security Groups > sg-0354ea177013e5427 - multi-docker

sg-0354ea177013e5427 - multi-docker

Actions ▾

Security group name multi-docker	Security group ID sg-0354ea177013e5427	Description multi-docke	VPC ID vpc-0d0d5d4c21e769367
Owner 218788053233	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer Run Reachability Analyzer

Inbound rules (1/1)

Filter security group rules

Manage tags | Edit inbound rules

< 1 > ⌂

New VPC Experience Tell us what you think

DHCP Options Sets

Elastic IPs

Managed Prefix Lists

Endpoints New

Endpoint Services

NAT Gateways

Peering Connections

▼ SECURITY

Network ACLs

Security Groups

▼ NETWORK ANALYSIS

Reachability Analyzer

Network Access Analyzer

▼ DNS FIREWALL

Rule Groups New

Domain Lists New

▼ NETWORK FIREWALL

Firewalls

Firewall Policies

Network Firewall Rule Groups

▼ VIRTUAL PRIVATE

us-east-1.console.aws.amazon.com/vpc/home?region=us-east-1#SecurityGroup:group-id=sg-0354ea177013e5427

Services Search for services, features, blogs, docs, and more [Option+S]

Actions ▾

VPC > Security Groups > sg-0354ea177013e5427 - multi-docker

sg-0354ea177013e5427 - multi-docker

Details

Security group name multi-docker	Security group ID sg-0354ea177013e5427	Description multi-docke	VPC ID vpc-0d0d5d4c21e769367
Owner 218788053233	Inbound rules count 1 Permission entry	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer Run Reachability Analyzer

Inbound rules (1/1)

Filter security group rules

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-09b2dfc9bc0262f4b	-	Custom TCP	TCP	0

< 1 > ⌂

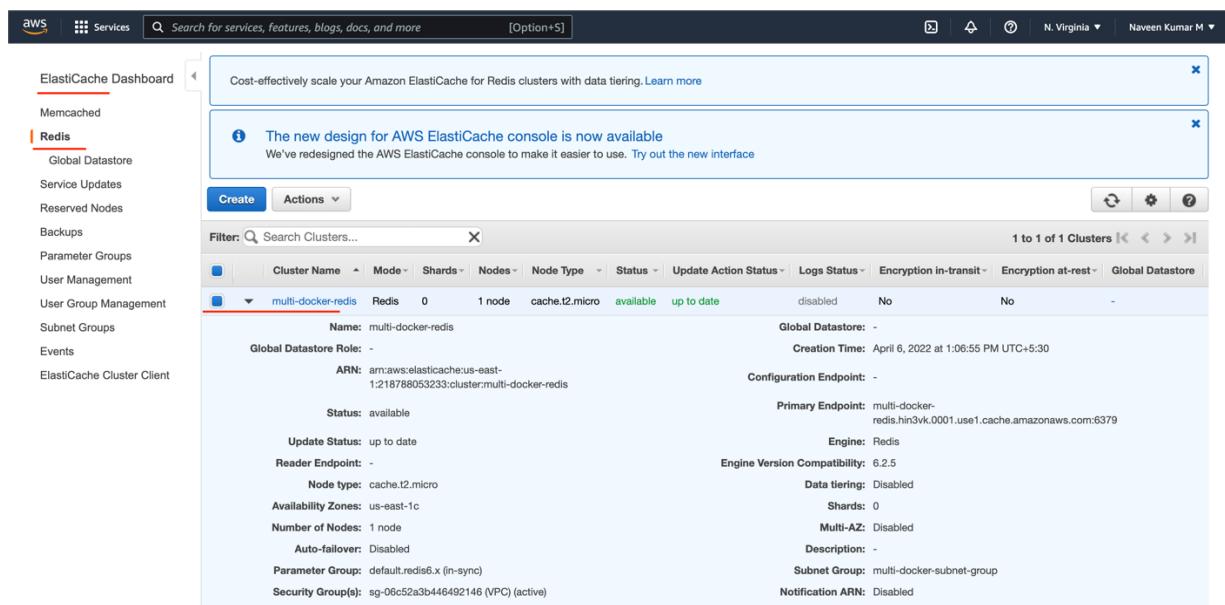
Applying Security Groups to ElastiCache

Steps

Applying Security Groups to ElastiCache

1. Go to AWS Management Console and use Find Services to search for ElastiCache
2. Click Redis in Sidebar
3. Check the box next to Redis cluster
4. Click Actions and click Modify
5. Click the pencil icon to edit the VPC Security group. Tick the box next to the new multi-docker group and click Save
6. Click Modify

Screenshots



Screenshot of the AWS ElastiCache Dashboard showing the new console design. A modal window displays a message about the redesign. The main interface shows a cluster named "multi-docker-redis" with details like engine (Redis), node type (cache.t2.micro), and configuration endpoint.

Actions dropdown menu:

- Backup
- Modify**
- Reboot
- Delete
- View/Manage Logs
- Apply Service Update
- View/Stop Update
- Setup Global Datastore
- Manage tags
- Migrate Data from Endpoint
- Stop Data Migration
- Manage Auto Scaling policies

Cluster Details:

- 1 node
- cache.t2.micro
- available
- up to date
- disabled
- No
- No

Configuration:

- Global Datastore: -
- Creation Time: April 6, 2022 at 1:06:55 PM UTC+5:30
- Configuration Endpoint: -
- Primary Endpoint: multi-docker-redis.hn3vk.0001.use1.cache.amazonaws.com:6379
- Engine: Redis
- Engine Version Compatibility: 6.2.5
- Data tiering: Disabled
- Shards: 0
- Multi-AZ: Disabled
- Description: -
- Subnet Group: multi-docker-subnet-group
- Notification ARN: Disabled

Parameter Group: default.redis6.x (in-sync)

Security Group(s): sg-06c52a3b446492146 (VPC) (active)

Screenshot of the AWS ElastiCache Modify Cluster dialog. The "VPC Security Group(s)" field is highlighted with a red circle. A modal window titled "Select Security groups" lists several security groups, with "multi-docker" selected. The "Save" button is highlighted with a red box.

Modify Cluster

Cluster Name: multi-docker-redis

Engine: redis

Engine Version Compatibility: 6.2.5

VPC Security Group(s): default (sg-06c52a3b446492146)

Parameter Group: default.redis6.x

Rescaling to R6gd nodes isn't supported. To use R6gd, create a new VPC security group.

Node Type: cache.t2.micro (555 MiB memory)

Enable Automatic Backups:

Backup Retention Period: 1 day(s)

Backup Window: 00:00 UTC-01:00

Cancel Save

Security Group(s): sg-06c52a3b446492146 (VPC) (active)

Notification ARN: Disabled

Screenshot of the AWS ElastiCache Modify Cluster dialog. The "VPC Security Group(s)" field is highlighted with a red circle. A modal window titled "Select Security groups" lists several security groups, with "multi-docker" selected. The "Save" button is highlighted with a red box.

Modify Cluster

Cluster Name: multi-docker-redis

Engine: redis

Engine Version Compatibility: 6.2.5

VPC Security Group(s): multi-docker (sg-0354ea177013e5427)

Parameter Group: default.redis6.x

Rescaling to R6gd nodes isn't supported. To use R6gd, create a new VPC security group.

Node Type: cache.t2.micro (555 MiB memory)

Enable Automatic Backups:

Backup Retention Period: 1 day(s)

Backup Window: 00:00 UTC-01:00

Cancel Save

Security Group(s): sg-06c52a3b446492146 (VPC) (active)

Notification ARN: Disabled

The screenshot shows the AWS ElastiCache Redis console. On the left sidebar, under the 'Redis' section, there are links for Global Datastore, Service Updates, Reserved Nodes, Backups, Parameter Groups, User Management, User Group Management, Subnet Groups, Events, and ElastiCache Cluster Client. The main content area displays a table with one cluster entry:

	Cluster Name	Mode	Shards	Nodes	Node Type	Status	Update Action Status	Logs Status	Encryption in-transit	Encryption at-rest	Global Datastore
<input checked="" type="checkbox"/>	multi-docker-redis	Redis	0	1 node	cache.t2.micro	available	up to date	disabled	No	No	-

Below the table, detailed information about the cluster is shown:

- Name:** multi-docker-redis
- Global Datastore Role:** -
- ARN:** arn:aws:elasticache:us-east-1:218788053233:cluster:multi-docker-redis
- Status:** available
- Update Status:** up to date
- Reader Endpoint:** -
- Node type:** cache.t2.micro
- Availability Zones:** us-east-1c
- Number of Nodes:** 1 node
- Auto-failover:** Disabled
- Parameter Group:** default.redis6.x (in-sync)
- Security Group(s):** sg-0354ea177013e5427 (VPC) (active)
- Maintenance Window:** fri:06:00-fri:07:00
- Ranking Window:** 00:00-01:00
- Global Datastore:** -
- Creation Time:** April 6, 2022 at 1:06:55 PM UTC+5:30
- Configuration Endpoint:** -
- Primary Endpoint:** multi-docker-redis.hn3vk.0001.use1.cache.amazonaws.com:6379
- Engine:** Redis
- Engine Version Compatibility:** 6.2.5
- Data Tiering:** Disabled
- Shards:** 0
- Multi-AZ:** Disabled
- Description:** -
- Subnet Group:** multi-docker-subnet-group
- Notification ARN:** Disabled
- Backup Retention Period:** 1 day(s)
- Encryption in-transit:** No

Applying Security Groups to RDS

Steps

Applying Security Groups to RDS

1. Go to AWS Management Console and use Find Services to search for RDS
2. Click Databases in Sidebar and check the box next to your instance
3. Click Modify button
4. Scroll down to Connectivity and add the new multi-docker security group
5. Scroll down and click the Continue button
6. Click Modify DB instance button

Screenshots

This screenshot shows the AWS RDS Databases page. On the left, there's a sidebar with various navigation options like Dashboard, Databases (which is selected and highlighted in orange), Query Editor, etc. The main content area is titled 'Databases' and shows a table of database instances. One instance, 'multi-docker-postgres', is highlighted with a red circle around its DB identifier column. The table includes columns for DB identifier, Role, Engine, Region & AZ, Size, Status, and CPU usage (4.75%). At the top right of the table, there are buttons for 'Group resources', 'Modify' (which is currently selected), 'Actions', 'Restore from S3', and 'Create database'.

This screenshot shows the 'Modify DB instance: multi-docker-postgres' configuration page. The left sidebar is identical to the previous screenshot. The main page has two main sections: 'Settings' and 'DB instance class'. In the 'Settings' section, the 'DB engine version' is set to '12.10'. The 'DB instance identifier' is set to 'multi-docker-postgres'. There's also a field for 'New master password'. In the 'DB instance class' section, the 'Burstable classes (includes t classes)' option is selected. Other options shown are 'Standard classes (includes m classes)' and 'Memory optimized classes (includes r and x classes)'. The URL in the browser bar is 'us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#modify-instance:id=multi-docker-postgres'.

us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#modify-instance:id=multi-docker-postgres

Amazon RDS

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Events

Event subscriptions

Recommendations 1

Certificate update

Availability & durability

Multi-AZ deployment [Info](#)

Create a standby instance (recommended for production usage)
Creates a standby in a different Availability Zone (AZ) to provide data redundancy, eliminate I/O freezes, and minimize latency spikes during system backups.

Do not create a standby instance

Connectivity

Subnet group: default-vpc-0dd0d5d4c21e769367

Security group: Choose security groups
multi-docker

Certificate authority: rds-ca-2019

Additional configuration

Database authentication

us-east-1.console.aws.amazon.com/rds/home?region=us-east-1#modify-instance:id=multi-docker-postgres

Amazon RDS

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Events

Event subscriptions

Recommendations 1

Certificate update

arn:aws:kms:us-east-1:218788053233:key/7cc346ef-5950-4bfb-8728-6482a3a0

Example: arn:aws:kms:<region>:<accountID>/key/<key-id>

Account: 218788053233

KMS key ID: 7cc346ef-5950-4bfb-8728-6482a3a08a1a

Warning: You can't change the KMS key after enabling Performance Insights.

Maintenance

Auto minor version upgrade [Info](#)

Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

DB instance maintenance window
The weekly time range during which system maintenance can occur.

Start day: Tuesday

Start time: 05 : 50 UTC

Duration: 0.5 hours

Deletion protection

Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

Cancel Continue

The screenshot shows the AWS RDS 'Modify DB instance' page for the database 'multi-docker-postgres'. On the left sidebar, under the 'Databases' section, the 'Security group' option is selected. In the main content area, there's a table showing the modification of the 'Security group' attribute from 'default' to 'multi-docker'. Below this, a section titled 'Scheduling of modifications' offers two options: 'Apply during the next scheduled maintenance window' (selected) and 'Apply immediately'. At the bottom right, there are 'Cancel', 'Back', and 'Modify DB instance' buttons, with the 'Modify DB instance' button being highlighted by a red box.

Applying Security Groups to Elastic Beanstalk

Applying Security Groups to Elastic Beanstalk

1. Go to AWS Management Console and use Find Services to search for Elastic Beanstalk
2. Click Environments in the left sidebar.
3. Click MultiDocker-env
4. Click Configuration
5. In the Instances row, click the Edit button.
6. Scroll down to EC2 Security Groups and tick box next to multi-docker
7. Click Apply and Click Confirm
8. After all the instances restart and go from No Data to Severe, you should see a green checkmark under Health.

us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environments

Elastic Beanstalk Services Search for services, features, blogs, docs, and more [Option+S]

N. Virginia Naveen Kumar M

All environments

Filter results matching the display values

Environment name	Health	Application name	Date created	Last modified	URL	Running versions	Platform	Platform state
Multidocker-env	Ok	multi-docker	2022-04-05 21:37:42 UTC+0530	2022-04-05 21:42:31 UTC+0530	Multidocker-env.eba-ziwmpj.us-east-1.elasticbeanstalk.com	Sample Application	Docker running on 64bit Amazon Linux 2	Support

us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/configuration?applicationName=multi-docker&environmentId=e-rm3gkdgkrf

Elastic Beanstalk Services Search for services, features, blogs, docs, and more [Option+S]

N. Virginia Naveen Kumar M

Configurations

Search for an option name or value

Category	Options	Actions
Software	Log streaming: disabled Proxy server: nginx Rotate logs: disabled X-Ray daemon: disabled	Edit
Instances	EC2 security groups: awseb-e-rm3gkdgkrf-stack-AWSEBSecurityGroup-1EABJ0IFNMT5 IMDSv1: disabled IOPS: container default Monitoring interval: 5 minute Root volume type: container default Size: container default Throughput: container default	Edit
	AMI ID: ami-05badaaaae806728e Availability Zones: Any Breach duration: 5 Capacity rebalancing: disabled Environment type: load balancing, auto scaling Instance types: t2.micro,t2.small Lower threshold: 2000000 Max: 4	

us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/configuration?applicationName=multi-docker&environmentId=e-rm3gkdgkrf N. Virginia N. Virginia Naveen Kumar M

Elastic Beanstalk

Environments Applications Change history

multi-docker Application versions Saved configurations

Multidocker-env Go to environment Configuration Logs Health Monitoring Alarms Managed updates Events Tags

Recent environments

The desired throughput to provision for the Amazon EBS root volume attached to your environment's EC2 instance MiB/s

Instance metadata service (IMDS)
Your environment's platform supports both IMDSv1 and IMDSv2. To enforce IMDSv2, disable IMDSv1. [Learn more](#)

Disable IMDSv1
With the current setting, the environment enables both IMDSv1 and IMDSv2.
 Disabled

EC2 security groups

Group name	Group ID	Name
awseb-e-rm3gkdgkrf-stack-AWSEBLoadBalancerSecurityGroup-1CB2NV06JVVKY	sg-0f35fef4442782098	Multidocker-env
awseb-e-rm3gkdgkrf-stack-AWSEBSecurityGroup-1EABJ00IFNMTS	sg-0d0c442938a636007	Multidocker-env
default	sg-06c52a3b446492146	-
multi-docker	sg-0354ea177013e5427	-

Cancel Continue **Apply**

us-east-1.console.aws.amazon.com/elasticbeanstalk/home?region=us-east-1#/environment/configuration?applicationName=multi-docker&environmentId=e-rm3gkdgkrf N. Virginia N. Virginia Naveen Kumar M

Elastic Beanstalk

Environments Applications Change history

multi-docker Application versions Saved configurations

Multidocker-env Go to environment Configuration Logs Health Monitoring Alarms Managed updates Events Tags

Recent environments

Elastic Beanstalk > Environments > Multidocker-env > Configuration

Service messages

warnings (1) errors

⚠ Changes to option SecurityGroups settings will not take effect immediately. Each of your existing EC2 instances will be replaced and your new settings will take effect then.

```
aws:autoscaling:launchconfiguration:SecurityGroups "awseb-e-rm3gkdgkrf-stack-AWSEBSecurityGroup-1EABJ00IFNMTS" to "multi-docker,awseb-e-rm3gkdgkrf-stack-AWSEBSecurityGroup-1EABJ00IFNMTS"
```

Cancel **Confirm**

Screenshot of the AWS Elastic Beanstalk console showing the environment "Multidocker-env". A message box at the top says "Elastic Beanstalk is updating your environment." Below it, the environment details are shown: "Multidocker-env" (e-rm3gkdgkrf), Application name: "multi-docker", Health: "Ok", Running version: "Sample Application", Platform: "Docker running on 64bit Amazon Linux 2/3.4.13". The "Recent events" section is empty.

Add AWS configuration details to Github Actions deploy.yml file's deploy script



```

    - run: docker build -t mnaveensmn/multi-worker-10-14 ./worker
    - run: docker push mnaveensmn/multi-client-10-14
    - run: docker push mnaveensmn/multi-nginx-10-14
    - run: docker push mnaveensmn/multi-server-10-14
    - run: docker push mnaveensmn/multi-worker-10-14

    - name: Generate deployment package
      run: zip -r deploy.zip . -x '*.git*'

    - name: Deploy to EB
      uses: einaregilsson/beanstalk-deploy@v18
      with:
        aws_access_key: ${{ secrets.AWS_ACCESS_KEY }}
        aws_secret_key: ${{ secrets.AWS_SECRET_KEY }}
        application_name: multi-docker
        environment_name: Multidocker-env
        existing_bucket_name: elasticbeanstalk-us-east-1-218788053233
        region: us-east-1
        version_label: ${{ github.sha }}
        deployment_package: deploy.zip
  
```

Setting Environment Variables

Steps

1. Go to AWS Management Console and use Find Services to search for Elastic Beanstalk
2. Click Environments in the left sidebar.
3. Click MultiDocker-env
4. Click Configuration
5. In the Software row, click the Edit button
6. Scroll down to Environment properties
7. In another tab Open up ElastiCache, click Redis and check the box next to your cluster. Find the Primary Endpoint and copy that value but omit the :6379
8. Set REDIS_HOST key to the primary endpoint listed above, remember to omit :6379
9. Set REDIS_PORT to 6379
10. Set PGUSER to postgres
11. Set PGPASSWORD to postgrespassword
12. In another tab, open up the RDS dashboard, click databases in the sidebar, click your instance and scroll to Connectivity and Security. Copy the endpoint.
13. Set the PGHOST key to the endpoint value listed above.
14. Set PGDATABASE to fibvalues
15. Set PGPORT to 5432
16. Click Apply button
17. After all instances restart and go from No Data, to Severe, you should see a green checkmark under Health.

Screenshots

Elastic Beanstalk > Environments > Multidocker-env > Configuration

Configurations

Software

- Log streaming: disabled
- Proxy server: nginx
- Rotate logs: disabled
- X-Ray daemon: disabled

Instances

- EC2 security groups: awseb-e-rm3gkdgkr-stack-AWSEBSecurityGroup-1EABJOOIFNMTS
- IMDSv1: disabled
- IOPS: container default
- Monitoring interval: 5 minute
- Root volume type: container default
- Size: container default
- Throughput: container default

AMI ID: ami-05badaaee806728e
Availability Zones: Any
Breach duration: 5
Capacity rebalancing: disabled
Environment type: load balancing, auto scaling
Instance types: t2.micro,t2.small
Lower threshold: 2000000
Max: 4

ElastiCache Dashboard

Redis

multi-docker-redis

Primary Endpoint: multi-docker-redis.hn3vk.0001.use1.cache.amazonaws.com:6379

Global Datastore Role: -
ARN: arn:aws:elasticache:us-east-1:218788053233:cluster:multi-docker-redis

Status: available

Update Status: up to date

Reader Endpoint: -

Node type: cache.t2.micro

Availability Zones: us-east-1c

Number of Nodes: 1 node

Auto-failover: Disabled

Parameter Group: default.redis6.x (in-sync)

Security Group(s): sg-0354ea17701e5427 (VPC) (active)

Elastic Beanstalk > Environments > Multidocker-env > Configuration

Environment properties

Name	Value
REDIS_HOST	multi-docker-redis.hn3vk.0001.use1.cache.amazonaws.com
REDIS_PORT	6379
PGUSER	postgres
PGPASSWORD	postgrespassword

Cancel **Continue** **Apply**

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Amazon RDS

- Dashboard
- Databases**
- Query Editor
- Performance insights
- Snapshots
- Automated backups
- Reserved instances
- Proxies

- Subnet groups
- Parameter groups
- Option groups
- Custom engine versions

- Events
- Event subscriptions

- Recommendations 1
- Certificate update

RDS > Databases > multi-docker-postgres

multi-docker-postgres

Summary

DB identifier multi-docker-postgres	CPU 5.38%	Status Available	Class db.t3.micro
Role Instance	Current activity 0.00 sessions	Engine PostgreSQL	Region & AZ us-east-1f

Modify Actions ▾

Connectivity & security Monitoring Logs & events Configuration Maintenance & backups Tags

Connectivity & security

Endpoint & port	Networking	Security
Endpoint multi-docker-postgres.cy0h9nb2gwvf.us-east-1.rds.amazonaws.com	Availability Zone us-east-1f	VPC security groups multi-docker (sg-0354ea177013e5427) Active
Port 5432	VPC vpc-0d0d5d4c21e769367	Public accessibility No
	Subnet group default-vpc-0d0d5d4c21e769367	Certificate authority multi-docker (sg-0354ea177013e5427)

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Elastic Beanstalk

- Environments
- Applications
- Change history

- multi-docker
 - Application versions
 - Saved configurations
- Multidocker-env
 - Go to environment
 - Configuration**
 - Logs
 - Health
 - Monitoring
 - Alarms
 - Managed updates
 - Events
 - Tags
- Recent environments

Environment properties

The following properties are passed in the application as environment properties. [Learn more](#)

Name	Value
REDIS_HOST	multi-docker-redis.hin3vk.0001.use1.cache.amazonaws.com
REDIS_PORT	6379
PGUSER	postgres
PGPASSWORD	postgrespassword
PGHOST	multi-docker-postgres.cy0h9nb2gwvf.us-east-1.rds.amazonaws.com
PGDATABASE	fibvalues
PGPORT	5432

Cancel Continue **Apply**

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Elastic Beanstalk

- Environments
- Applications
- Change history

- multi-docker
 - Application versions
 - Saved configurations
- Multidocker-env
 - Go to environment
 - Configuration
 - Logs
 - Health
 - Monitoring
 - Alarms
 - Managed updates
 - Events
 - Tags
- Recent environments

Elastic Beanstalk is updating your environment.

To cancel this operation select **Abort Current Operation** from the **Actions** dropdown.
[View Events](#)

Multidocker-env Multidocker-env.eba-z1wmappj.us-east-1.elasticbeanstalk.com (e-rm3gkdgkrf) Application name: multi-docker

Health	Running version	Platform
	Sample Application Upload and deploy	 Docker running on 64bit Amazon Linux 2/3.4.13 Change

Recent events

Time	Type	Details

Show all < 1 >

IAM Keys for Deployment

Steps

IAM Keys for Deployment

You can use the same IAM User's access and secret keys from the single container app we created earlier, or, you can create a new IAM user for this application:

1. Search for the "IAM Security, Identity & Compliance Service"
2. Click "Create Individual IAM Users" and click "Manage Users"
3. Click "Add User"
4. Enter any name you'd like in the "User Name" field.

eg: docker-multi-travis-ci

5. Tick the "Programmatic Access" checkbox
6. Click "Next:Permissions"
7. Click "Attach Existing Policies Directly"
8. Search for "beanstalk"
9. Tick the box next to "AdministratorAccess-AWSElasticBeanstalk"
10. Click "Next:Tags"
11. Click "Next:Review"
12. Click "Create user"
13. Copy and / or download the *Access Key ID* and *Secret Access Key* to use in the Travis Variable Setup.

Screenshots

This screenshot shows the AWS Identity and Access Management (IAM) service in the AWS Management Console. The left sidebar is titled 'Identity and Access Management (IAM)' and includes a search bar and a dashboard link. Under the 'Access management' section, 'Users' is selected, indicated by an orange underline. Other options include 'User groups', 'Roles', 'Policies', 'Identity providers', and 'Account settings'. The main content area is titled 'Users (0) Info' and contains a message: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' Below this is a search bar labeled 'Find users by username or access key'. A table header row is visible with columns for 'User name', 'Groups', 'Last activity', 'MFA', 'Password age', and 'Active key age'. A message at the bottom states 'No resources to display'. At the top right of the main content area are 'Delete' and 'Add users' buttons.

This screenshot is identical to the one above, showing the AWS IAM service in the AWS Management Console. The left sidebar and main content area are the same, including the 'Users' selection under 'Access management'. The difference is that the 'Add users' button in the top right corner of the main content area is now highlighted with a red box, drawing attention to it.

Add user

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* [Add another user](#)

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

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

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

* Required [Cancel](#) [Next: Permissions](#)

Add user

Set permissions

Add user to group Copy permissions from existing user [Attach existing policies directly](#) [Create policy](#)

Filter policies Showing 14 results

Policy name	Type	Used as
<input checked="" type="checkbox"/> AdministratorAccess-AWSElasticBeanstalk	AWS managed	None
<input type="checkbox"/> AWSElasticBeanstalkCustomPlatformforEC2Role	AWS managed	None
<input type="checkbox"/> AWSElasticBeanstalkEnhancedHealth	AWS managed	Permissions policy (1)
<input type="checkbox"/> AWSElasticBeanstalkManagedUpdatesCustomerRolePolicy	AWS managed	Permissions policy (1)
<input type="checkbox"/> AWSElasticBeanstalkMulticontainerDocker	AWS managed	Permissions policy (1)
<input type="checkbox"/> AWSElasticBeanstalkReadOnly	AWS managed	None
<input type="checkbox"/> AWSElasticBeanstalkRoleCore	AWS managed	None
<input type="checkbox"/> AWSElasticBeanstalkRoleCWL	AWS managed	None
<input type="checkbox"/> AWSElasticBeanstalkRoleECS	AWS managed	None

[Cancel](#) [Previous](#) [Next: Tags](#)

Add user

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="Add new key"/>	<input type="text"/>	Remove

You can add 50 more tags.

[Cancel](#) [Previous](#) [Next: Review](#)

Add user

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	docker-multi-github-actions-ci
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AdministratorAccess-AWSElasticBeanstalk

Tags

No tags were added.

Cancel Previous Create user

Add user

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://218788053233.signin.aws.amazon.com/console>

Download .csv

User	Access key ID	Secret access key
docker-multi-github-actions-ci	AKIAJTF4GIYDY2FE4GLFG	***** Show

Close

AWS Keys in Github Actions

The screenshot shows the 'Actions secrets' section of a GitHub repository settings page. On the left, a sidebar lists repository management options like General, Access, Collaborators, Moderation options, Code and automation, Security, and Integrations. The 'Secrets' option under 'Actions' is selected. The main area displays two sections: 'Environment secrets' (which is empty) and 'Repository secrets'. Under 'Repository secrets', four environment variables are listed: AWS_ACCESS_KEY, AWS_SECRET_KEY, DOCKER_PASSWORD, and DOCKER_USERNAME. Each entry includes an 'Update' button and a 'Remove' button.

Deploying the App

The screenshot shows a GitHub Actions run log for a workflow named 'Revert "updated the npm version" Deploy MultiDocker #7'. The workflow summary indicates it started 38s ago. The 'build' job is currently active, showing a list of steps: Set up job, Run actions/checkout@v2, Run docker login -u *** -p ***, Run docker build -t ***/react-test -f ./client/Dockerfile.dev ./client, Run docker run -e CI=true ***/react-test npm test, Run docker build -t ***/multi-client-10-14 ./client, Run docker build -t ***/multi-nginx-10-14 ./nginx, Run docker build -t ***/multi-server-10-14 ./server, Run docker push ***/multi-client-10-14, Run docker push ***/multi-nginx-10-14, Run docker push ***/multi-server-10-14, Run docker push ***/multi-worker-10-14, Generate deployment package, Deploy to EB, and Post Run actions/checkout@v2. A 'Cancel workflow' button is visible at the top right of the log view.

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia Naveen Kumar M

Elastic Beanstalk Environments Applications Change history

multi-docker Application versions Saved configurations

Multidocker-env Go to environment Configuration Logs Health Monitoring Alarms Managed updates Events Tags

Recent environments

Elastic Beanstalk > Environments > Multidocker-env

Elastic Beanstalk is updating your environment. To cancel this operation select Abort Current Operation from the Actions dropdown. View Events

Multidocker-env Multidocker-env.eba-ziwmpapp.us-east-1.elasticbeanstalk.com (e-rm3gkdgkrf) Application name: multi-docker

Health Info Causes

Running version Sample Application Upload and deploy

Platform Docker running on 64bit Amazon Linux 2/3.4.13 Change

Recent events Show all < 1 >

Time Type Details

github.com/mnaveensmn/docker-multi-container/runs/5855432586?check_suite_focus=true

Search or jump to... Pull requests Issues Marketplace Explore

mnavenmsn / docker-multi-container Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Revert "updated the npm version" Deploy MultiDocker #

Summary Jobs build

build Succeeded 1 minute ago in 5m 6s

Set up job Run actions/checkout@v2 Run docker login -u *** -p *** Run docker build -t ***/react-test -f ./client/Dockerfile.dev ./client Run docker run -e CINTRUE ***/react-test npm test Run docker build -t ***/multi-client-10-14 ./client Run docker build -t ***/multi-nginx-10-14 ./nginx Run docker build -t ***/multi-server-10-14 ./server Run docker build -t ***/multi-worker-10-14 ./worker Run docker push ***/multi-client-10-14 Run docker push ***/multi-nginx-10-14 Run docker push ***/multi-server-10-14 Run docker push ***/multi-worker-10-14 Generate deployment package Deploy to EB Post Run actions/checkout@v2 Complete job

Search logs

Not Secure — multidocker-env.eba-ziwmpapp.us-east-1.elasticbeanstalk.com

Home Link Other Page

Enter your index: Submit

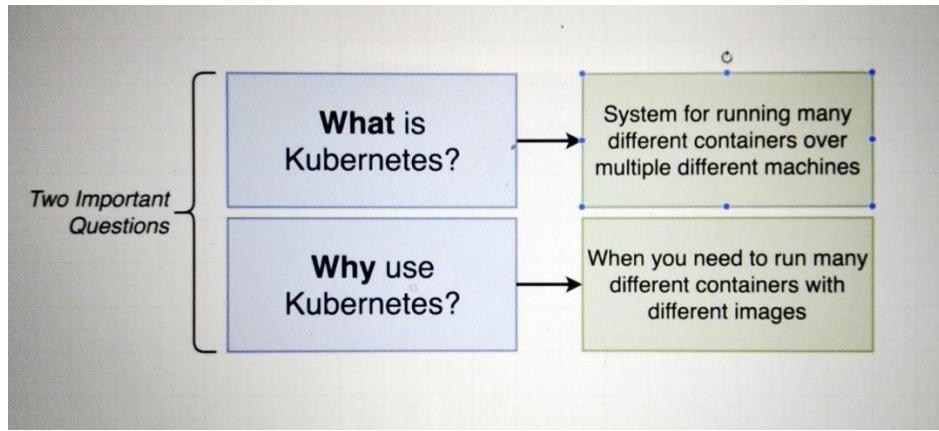
Indexes I have seen:

Calculated Values:

Onwards to Kubernetes

The Why's and What's of Kubernetes

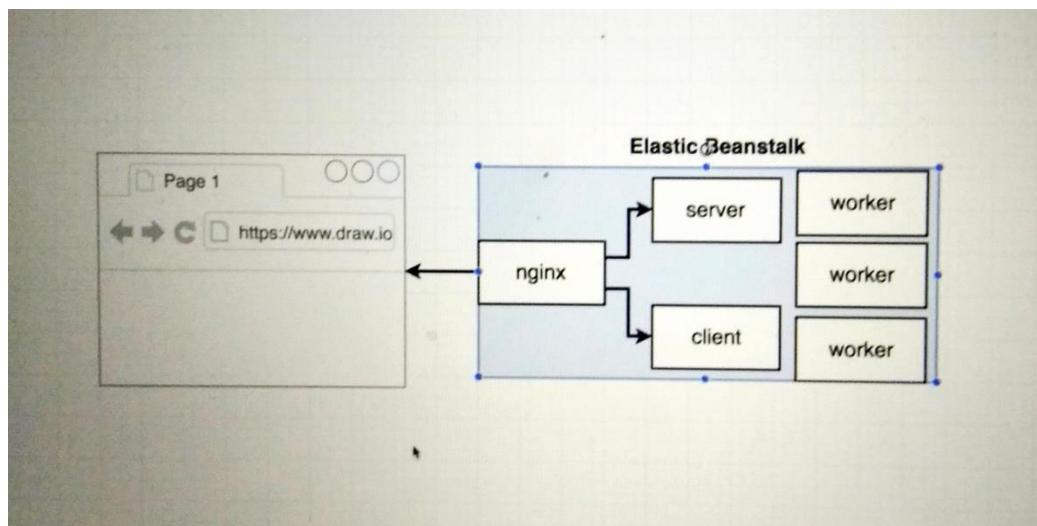
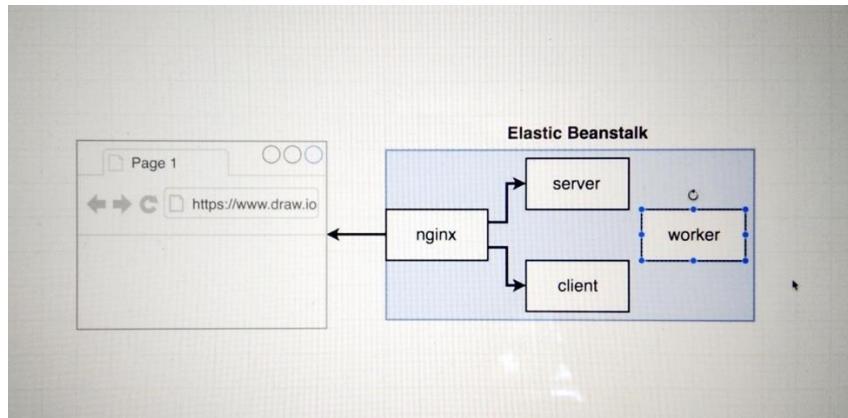
Why Kubernetes?



How do we scale our elastic beanstalk application?

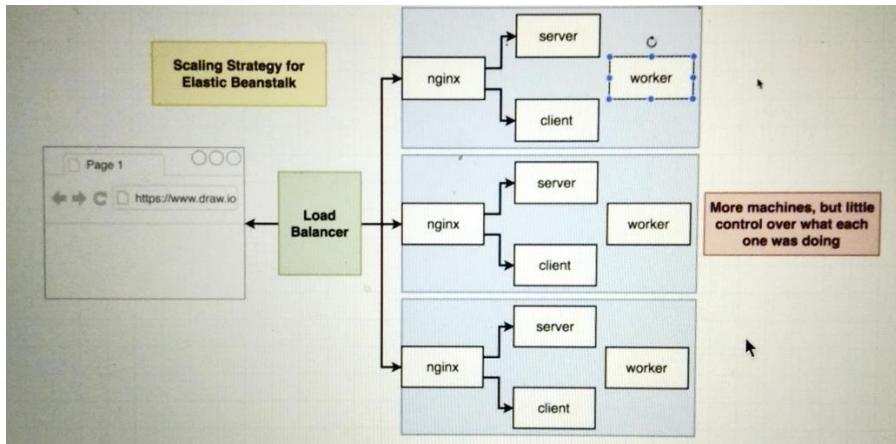
How do we manage the large traffic request for Fibonacci number?

The worker container doing the actual job of finding the Fibonacci number. If worker container does the replica, it would be easy for us to manage the traffic.

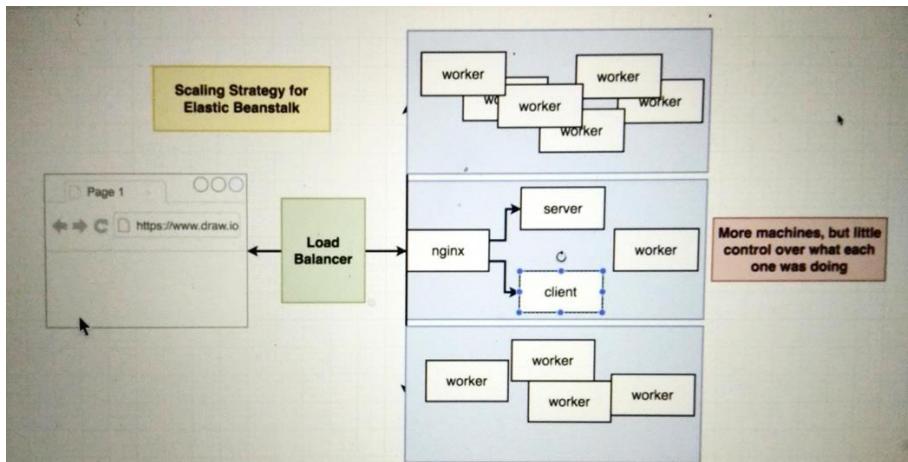


Scaling Challenge

Elastic beanstalk spins up additional copies of entire set of containers.



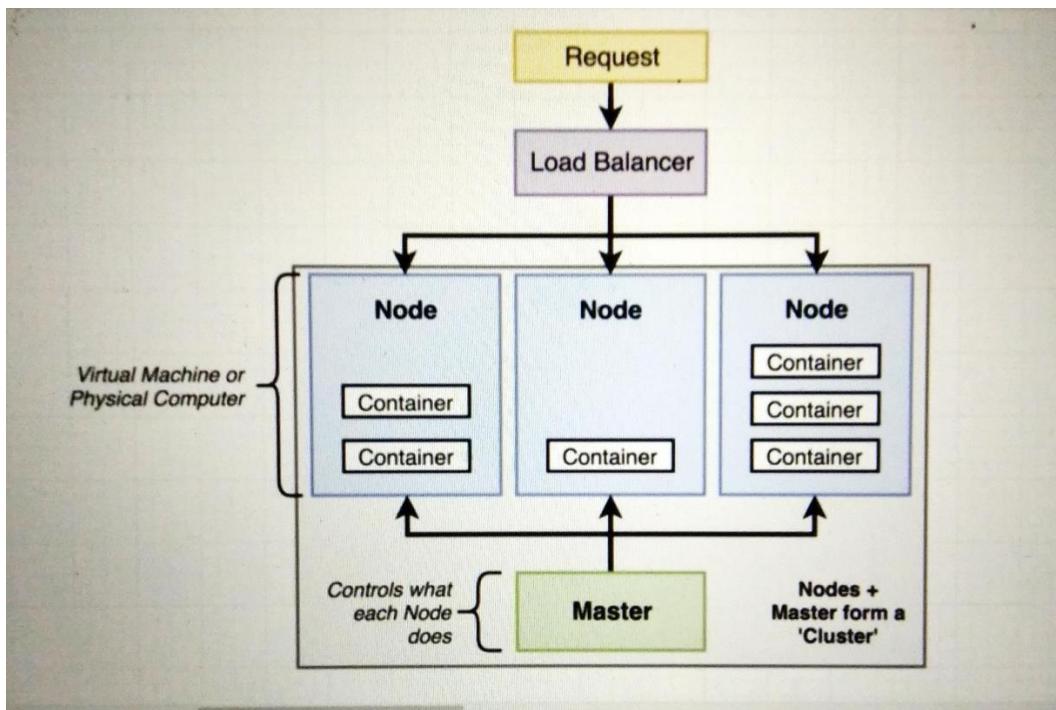
It will be useful when we have an additional copies of worker container instead of others.



Tons of additional copies worker container makes application efficient when we have a large traffic.

By using the Kubernetes, we can have an additional container and determine what these additional containers should be doing. Used to solve the scaling issue.

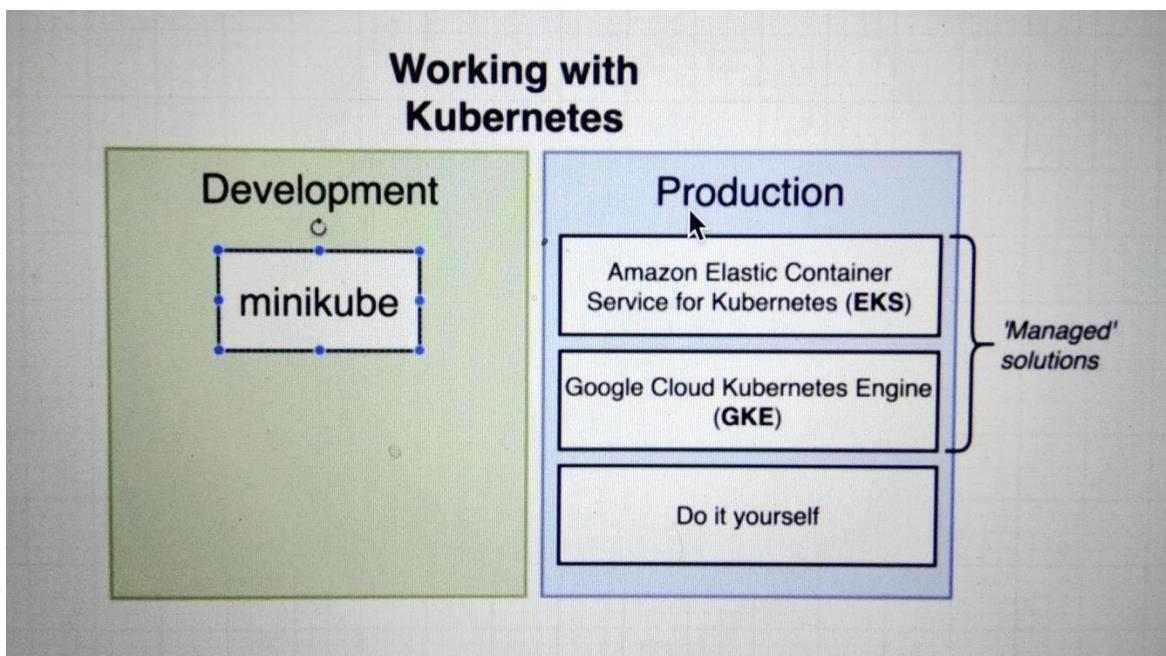
Kubernetes cluster

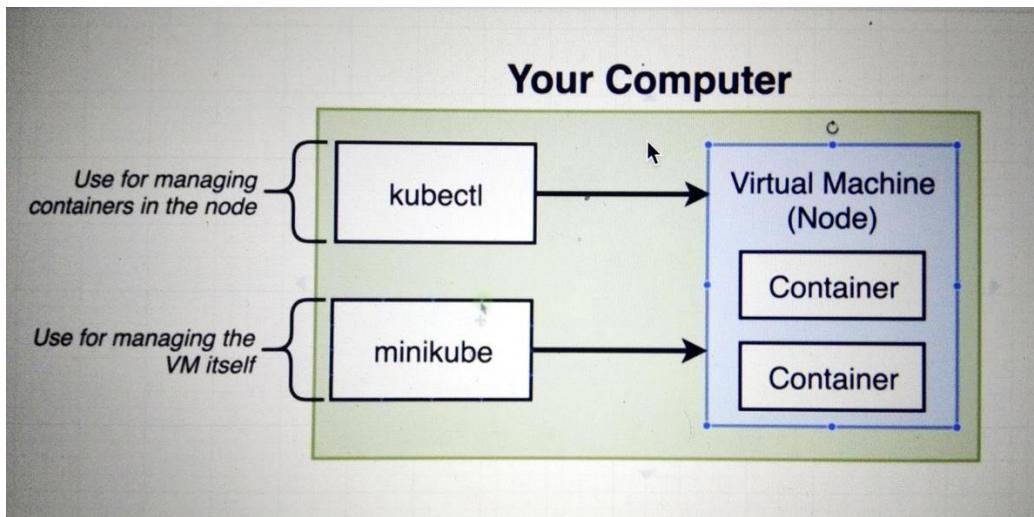


Kubernetes cluster is the assembly of master and one or more nodes. Each node will run different set of containers. Inside the node, containers can be of different types. These nodes are created and managed by master. We can send the docker commands to master. Then, master will take care of running the commands in different nodes.

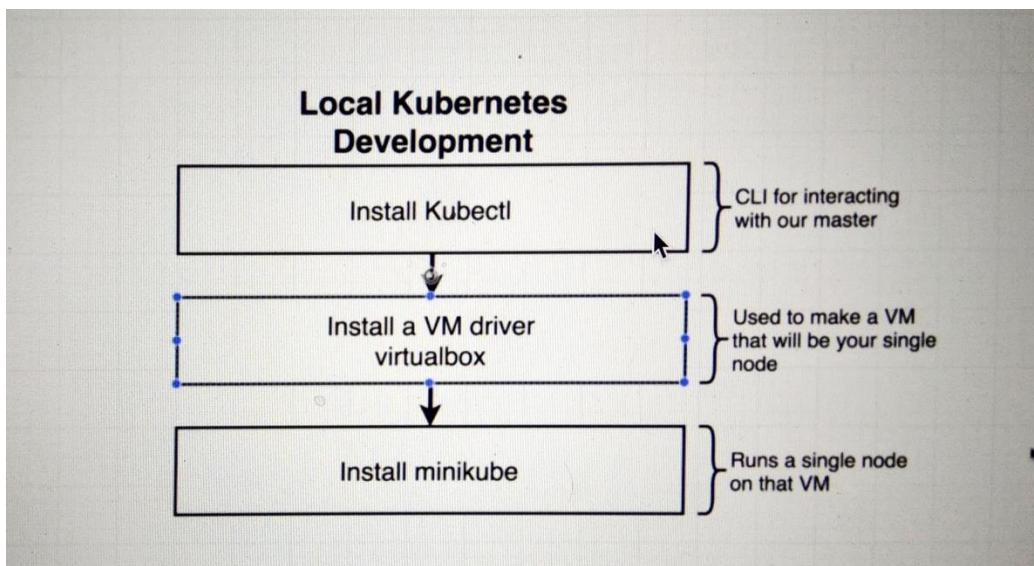
Getting Started with Kubernetes

Kubernetes in Development and Production





Minikube is only used in local.



Minikube Setup

Install Minikube with Homebrew

First, make sure you have Homebrew installed. If not, follow the instructions here:

<https://brew.sh/>

Then, in your terminal, run:

```
brew install minikube
```

Your output should look something like this (it will be ok if it is not exact)

Starting Minikube and Testing Installation

After you have successfully installed Minikube we need to start and test the cluster to make sure everything is working correctly.

1. Start with VM driver:

In your terminal, run:

```
minikube start --driver=hyperkit
```

2. Check Minikube Status

After you see a **Done!** message in your terminal, run `minikube status` to make sure the cluster is healthy. Pay particular attention that the **apiserver** is in a "**Running**" state.

```
➜ ~ minikube status
minikube
  type: Control Plane
  host: Running
  kubelet: Running
  apiserver: Running
  kubeconfig: Configured
  timeToStop: Nonexistent

➜ ~ █
```

3. Check kubectl

3. Check kubectl

Lastly, open up your terminal and make sure that you can run

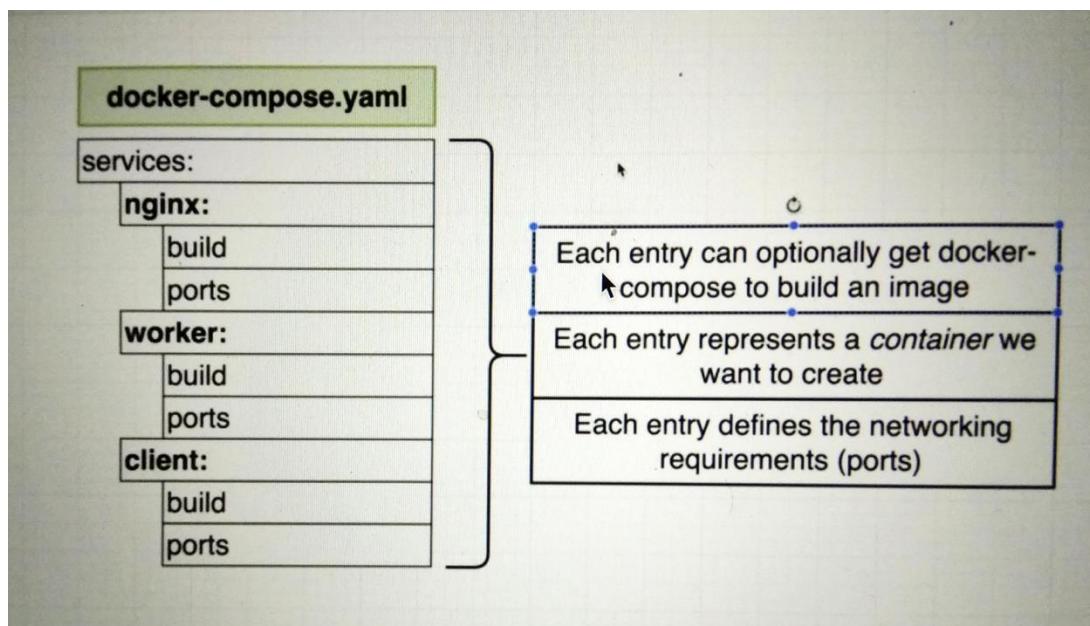
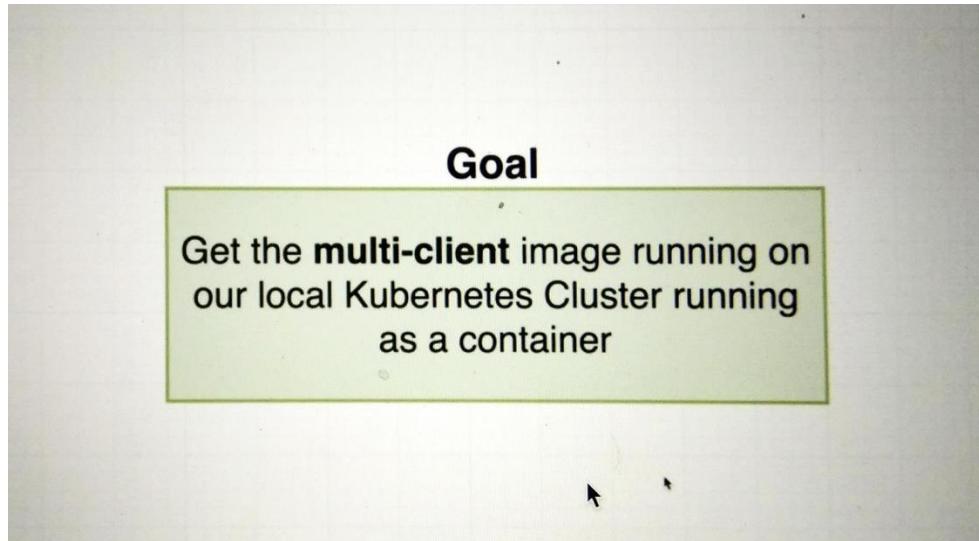
```
kubectl version
```

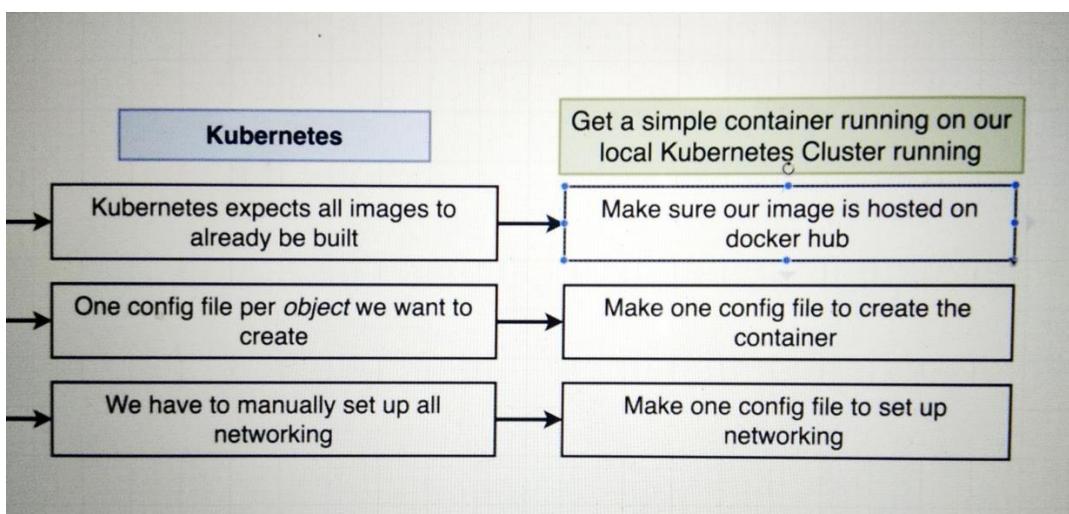
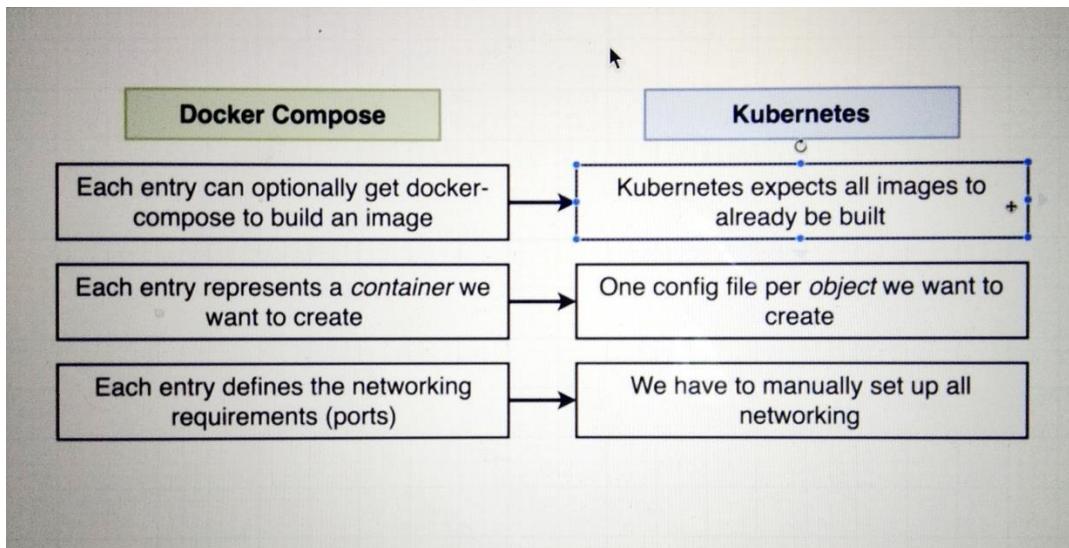
```
➜ ~ kubectl version
Client Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.3", GitCommit:"1e1
1e4a2108024935ecfc2912226cedeadf99df", GitTreeState:"clean", BuildDate:"2020-10-14T12:5
0:19Z", GoVersion:"go1.15.2", Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"19", GitVersion:"v1.19.3", GitCommit:"1e1
1e4a2108024935ecfc2912226cedeadf99df", GitTreeState:"clean", BuildDate:"2020-10-14T12:4
1:49Z", GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

Note - the client and server can be off by one minor version without error or issue.

Mapping Existing Knowledge

```
→ ~ kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
→ ~
```





Adding Configuration File

Configuration file to create a container

In terms of kubernetes, container is a object.

```


✓ DOCKER_KUBE



- > 00_redis_image
- > 01_simpleweb
- > 02_user_visits
- > 03_frontend-read... ●
- ✓ 04_simplek8s ●
- ↳ client-node-por... U
- ↳ client-pod.yaml U
- ↳ .gitignore



04_simplek8s > client-pod.yaml > {} spec > [ ] containers >

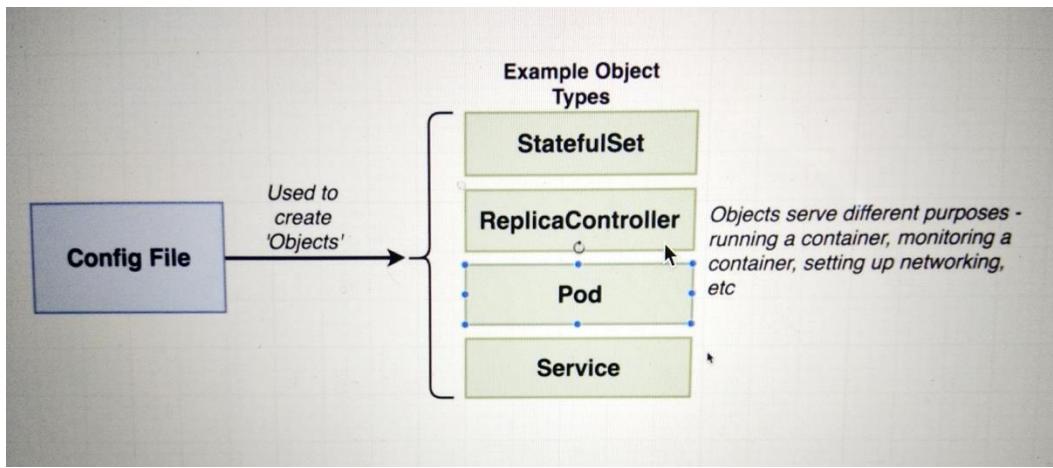


```

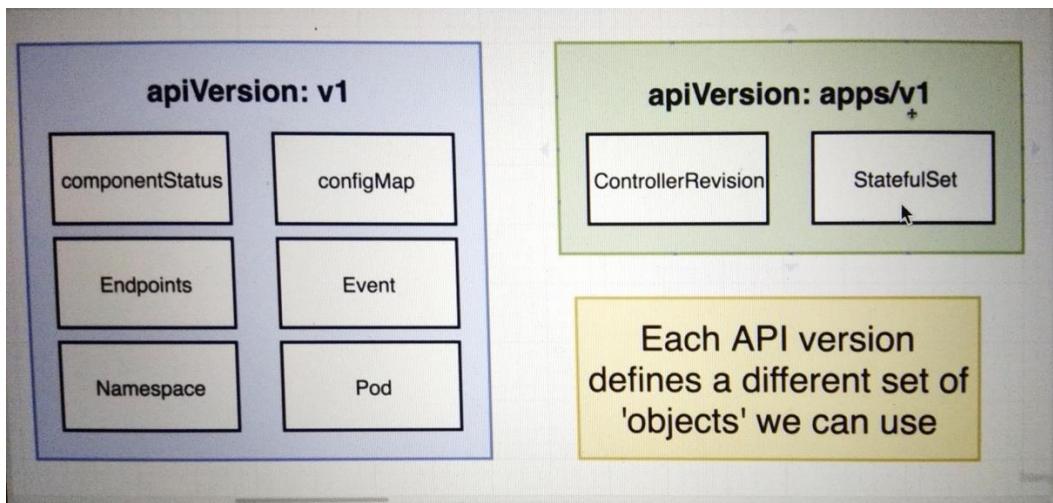
1 apiVersion: v1
2 kind: Pod
3 metadata:
4 name: client-pod
5 labels:
6 component: web
7 spec:
8 containers:
9 - name: client
10 image: stephengrider/multi-client
11 ports:
12 - containerPort: 3000
13

```


```



“Kind” property represents the object type. Pod object type is used to run a container.



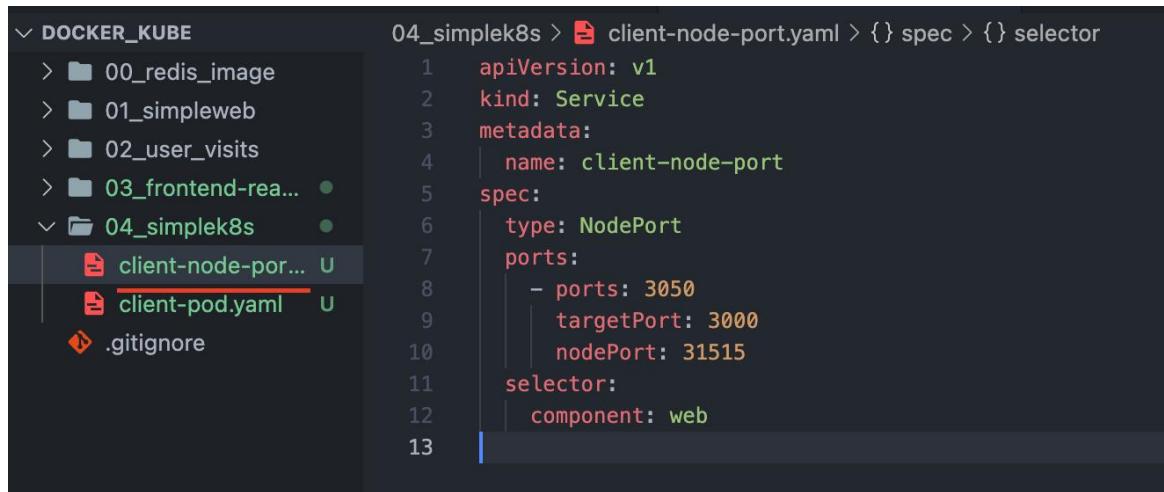
“apiVersion” defines a different set of objects we can use.

Spec – container property represents a name of the container and image repository and port we want to expose.

Meta-data – name property represent a name of the pod.

Labels-component property used to connect another config file (object). Here we specified the values as “web” that’s is used by network config file.

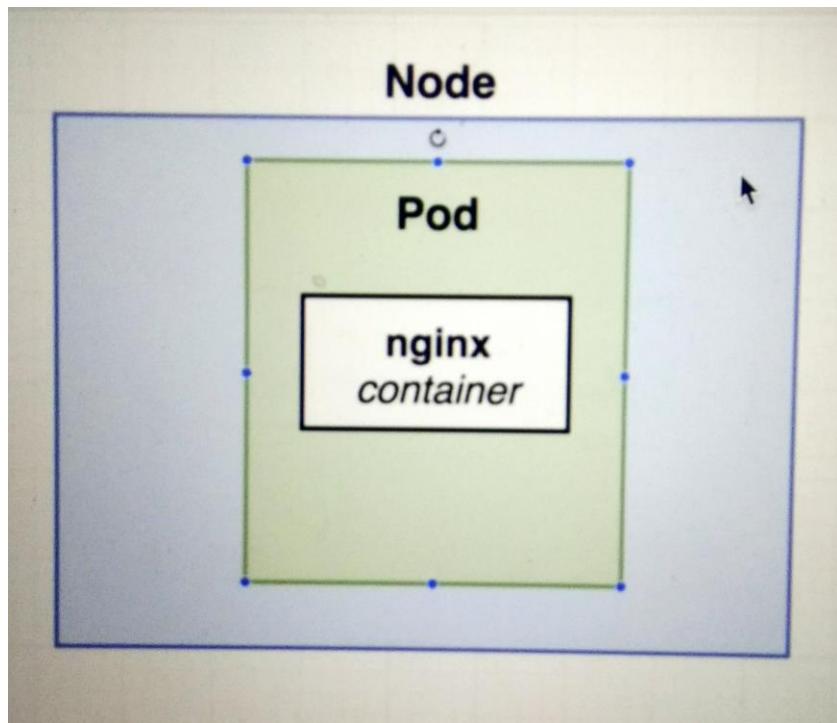
Configuration file to set up networking



The terminal shows a file structure under 'DOCKER_KUBE'. Inside '04_simplek8s', there are files 'client-node-port.yaml' and 'client-pod.yaml'. The 'client-node-port.yaml' file is displayed in a code editor:

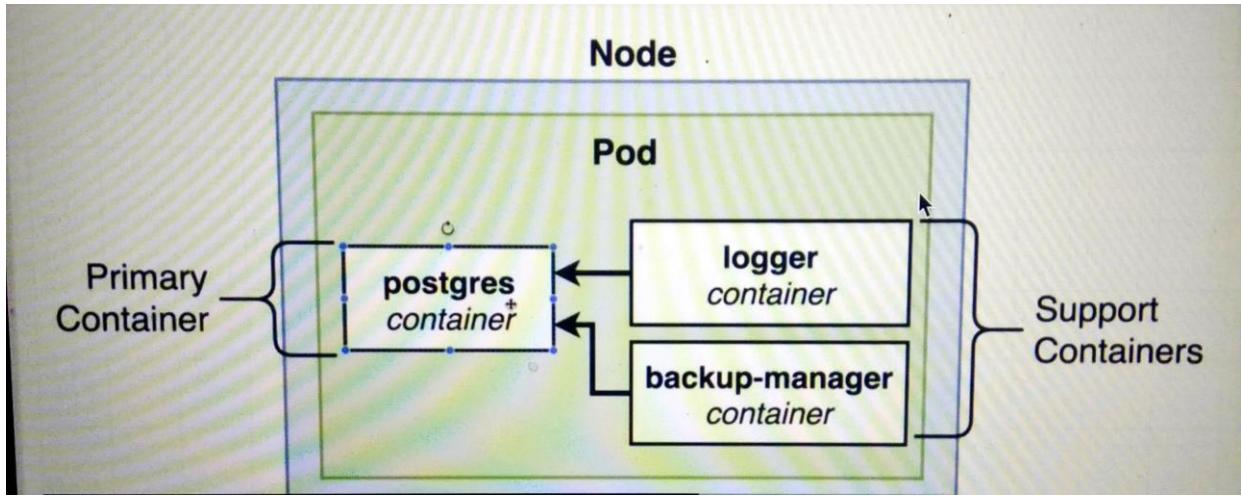
```
04_simplek8s > client-node-port.yaml > {} spec > {} selector
1   apiVersion: v1
2   kind: Service
3   metadata:
4     name: client-node-port
5   spec:
6     type: NodePort
7     ports:
8       - ports: 3050
9         targetPort: 3000
10        nodePort: 31515
11      selector:
12        component: web
13
```

Running a container in Pod



Minikube creates a virtual machine in our machine, that is called Node. This node will be later used by kubernetes to run some number of objects. The object is called Pod.

Pod is a grouping of container with a common purpose. Tightly coupled containers with close relations. If either of container crashes application will break.

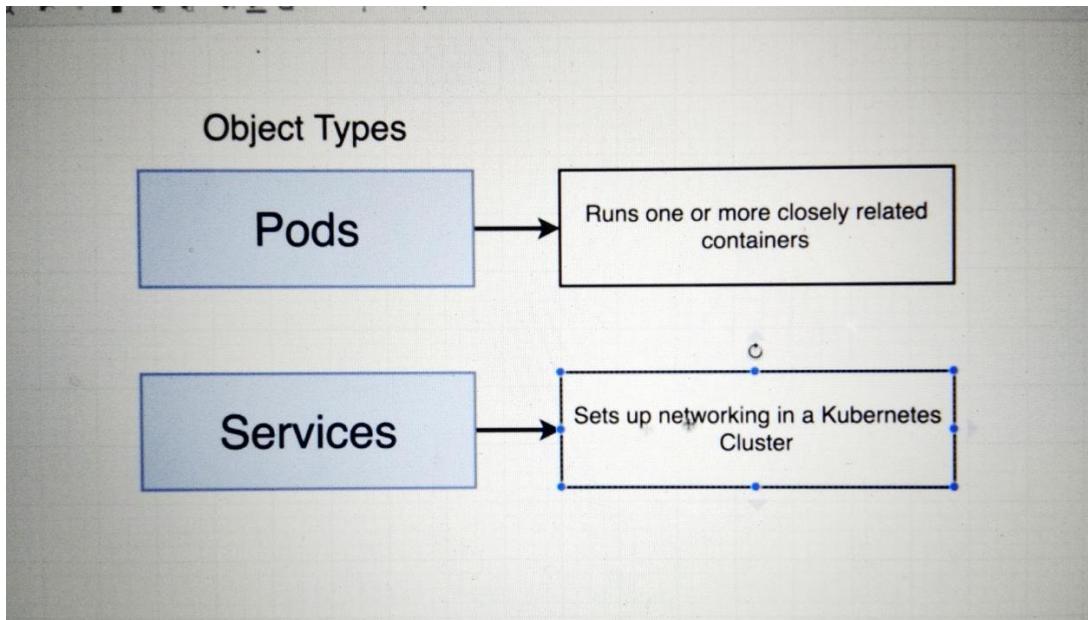


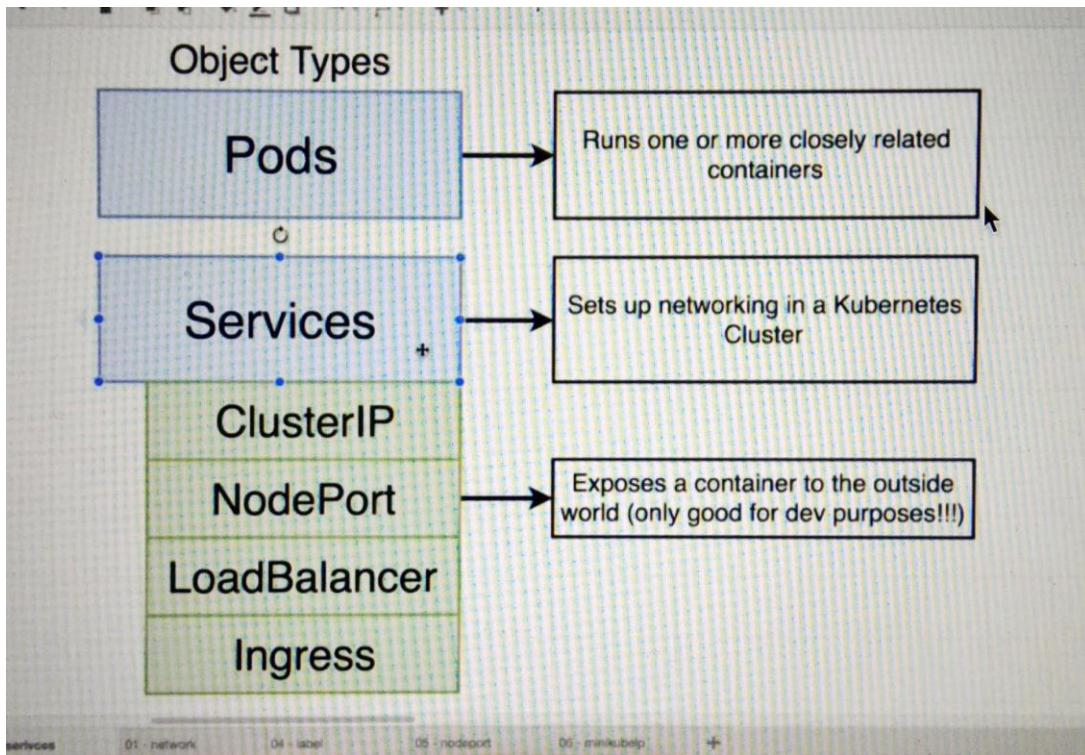
In the above application, if primary containers go away, support container won't work. So, this is the tightly coupled scenario.

In Kubernetes, we cannot deploy individual container by themselves. Anytime when we want to deploy container in the world of kubernetes, we need to make use pod.

Pod can run one or more container inside of it. These containers in the Pod are close relation with each other.

Service Config File in Depth



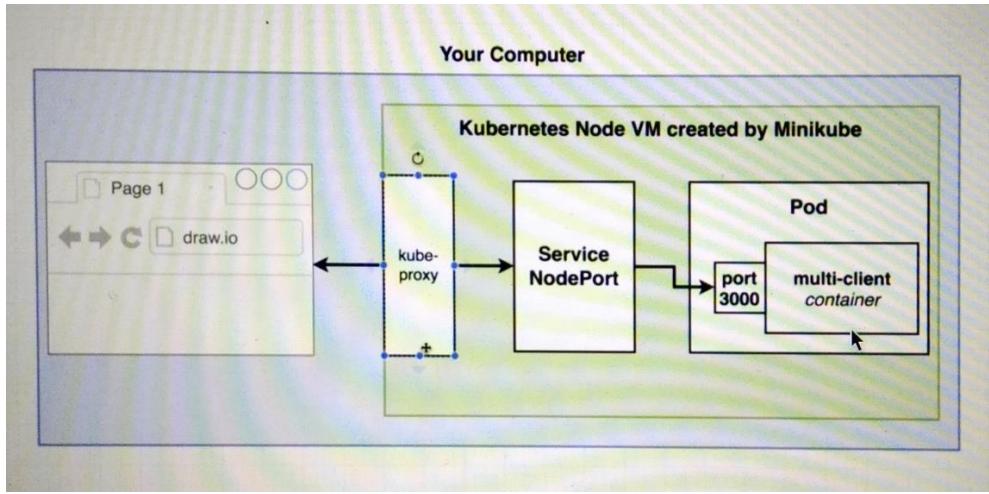


Service has four sub type.

```

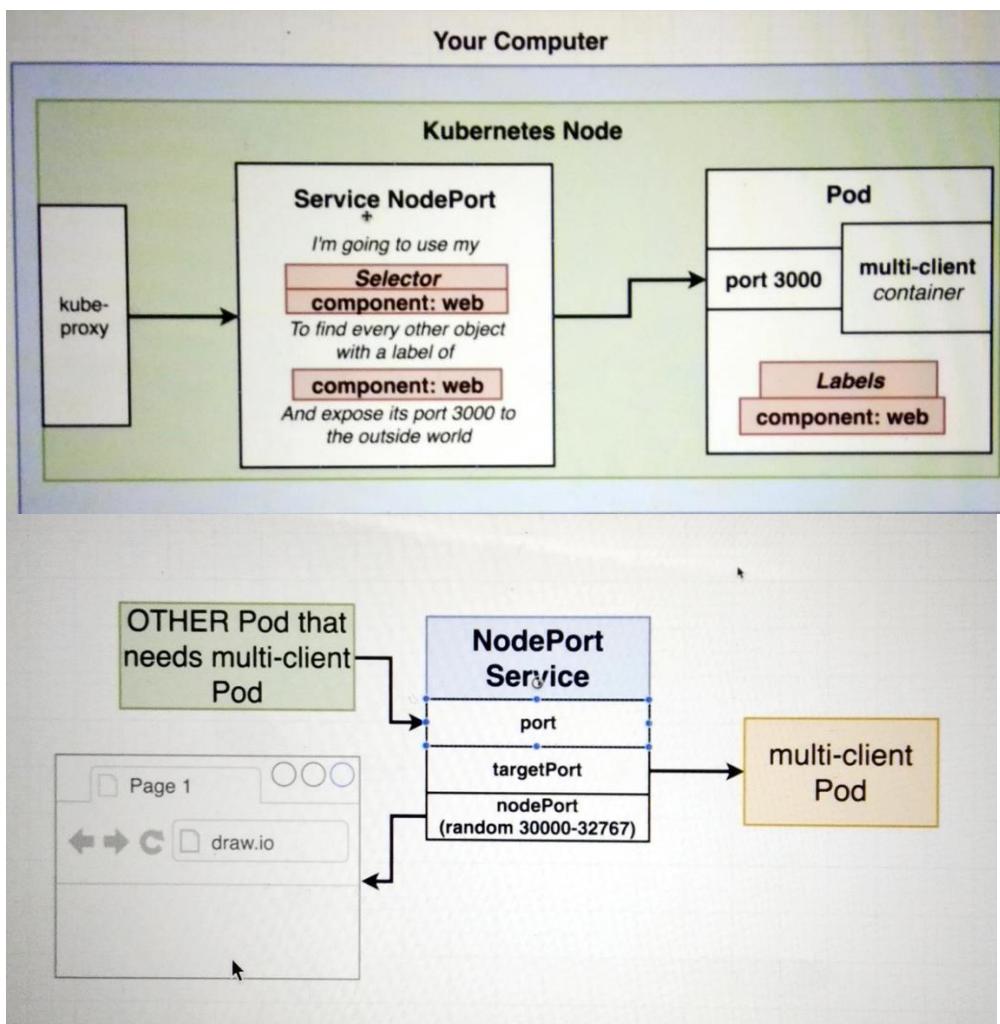
04_simplek8s > 📄 client-node-port.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: client-node-port
5  spec:
6    type: NodePort
7    ports:
8      - ports: 3050
9        targetPort: 3000
10       nodePort: 31515
11    selector:
12      component: web
13
  
```

For our example we used the NodePort type. It exposes the container to outside world which means accessing through web browser. NodePort is only suitable for development purpose.



Initial request goes to the kube-proxy. Next it navigated to Service NodePort. From there, request will be mapped to the container running inside the Pod.

Let's look at inside of the node more detail.

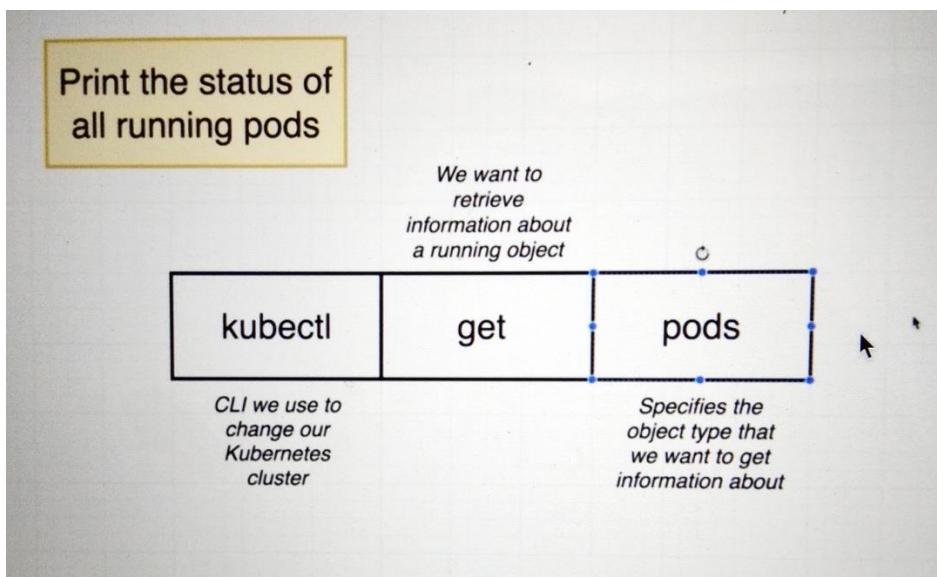
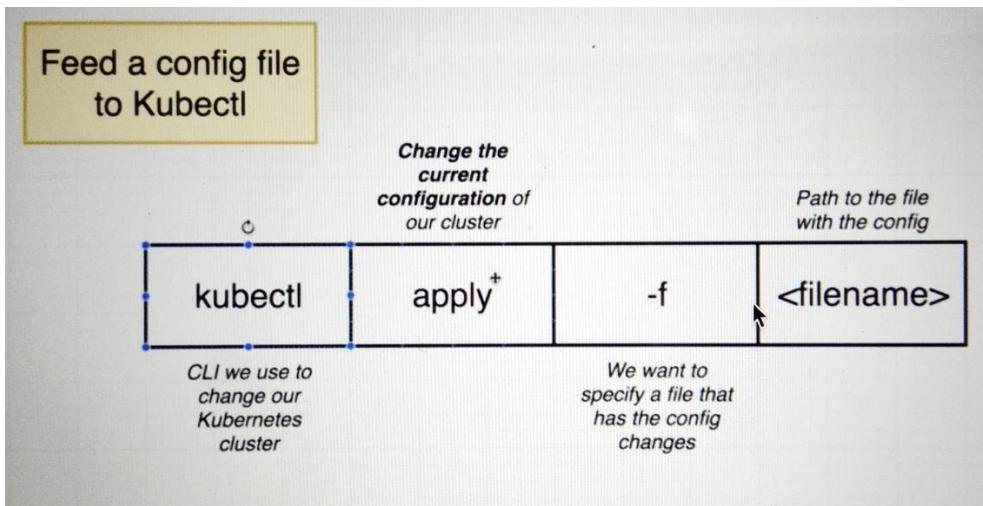


Port – used for communication between Pods.

targetPort – tells in which port request should be navigated to.

nodePort – used for sending request to Pod from browser.

Connecting to Running Container



DOCKER_KUBE

- > 00_redis_image
- > 01_simpleweb
- > 02_user_visits
- > 03_frontend-react-app
- > 04_simplek8s
 - client-node-port.yaml
 - client-pod.yaml
- .gitignore

TERMINAL

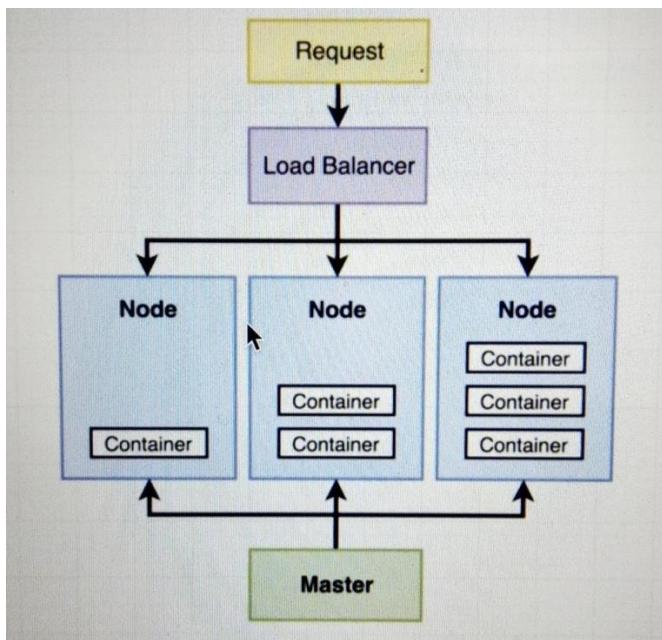
```
/Users/naveen.kumar1/.zshrc:68: no matches found: load?
→ DOCKER_KUBE git:(main) ✘ cd 04_simplek8s
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-pod.yaml
pod/client-pod created
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-node-port.yaml
service/client-node-port created
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
client-pod     1/1     Running   0          2m31s
→ 04_simplek8s git:(main) ✘
```

Kubectl get services

```
CLIENT pod 1/1 Running 0 2m31s
→ 04_simplek8s git:(main) ✘ kubectl get services
NAME           TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
client-node-port NodePort  10.99.156.34 <none>        3050:31515/TCP 4m30s
kubernetes     ClusterIP  10.96.0.1    <none>        443/TCP   4d16h
→ 04_simplek8s git:(main) ✘
```

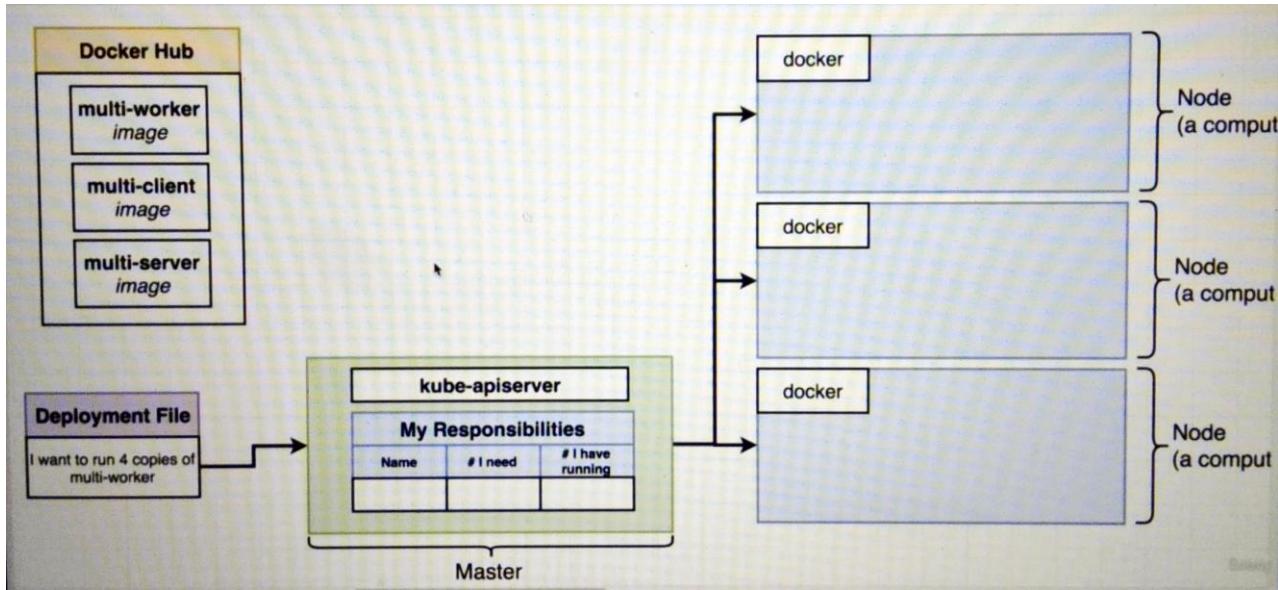
Output of running kubernetes cluster

The Entire Deployment Flow



```
ruben@ubu: ~ [1]: 10:56:01 ~ [root] ~ [root]
→ 04_simplek8s git:(main) ✘ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
e8880b2d3e85 stephengrider/multi-client "nginx -g 'daemon of..." 12 minutes ago Up 12 minutes
f-0f615c6792fe_0
→ 04_simplek8s git:(main) ✘ docker kill e8880b2d3e85
e8880b2d3e85
→ 04_simplek8s git:(main) ✘ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME READY STATUS RESTARTS AGE
client-pod 1/1 Running 1 (18s ago) 19m
→ 04_simplek8s git:(main) ✘ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
d6ffbf6793ab5 stephengrider/multi-client "nginx -g 'daemon of..." 3 minutes ago Up 3 minutes
0f615c6792fe_1
→ 04_simplek8s git:(main) ✘ █
```

Even though we are killing the docker container, container getting restarted automatically in kubernetes cluster.



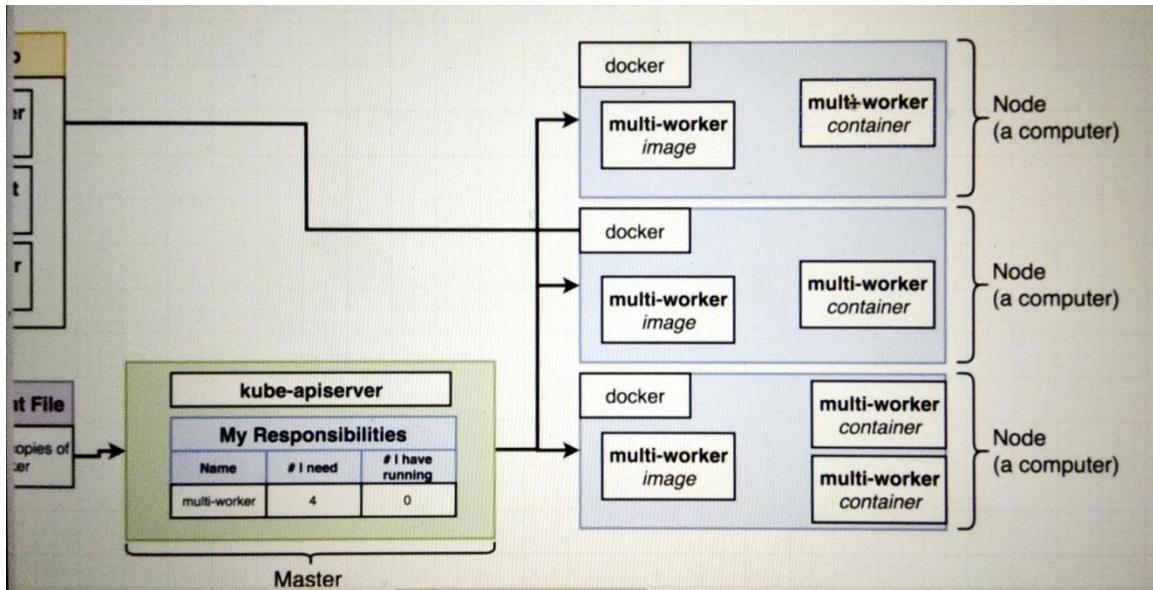
Here, Node represent virtual machine or computer that going to run some number of objects that we create inside the kubernetes cluster.

When we give “kubectl apply” command, configuration file passed on to master. On the master there is variety of different program that controls the entire kubernetes cluster. For an example, we are going to consider only kube-apiserver.

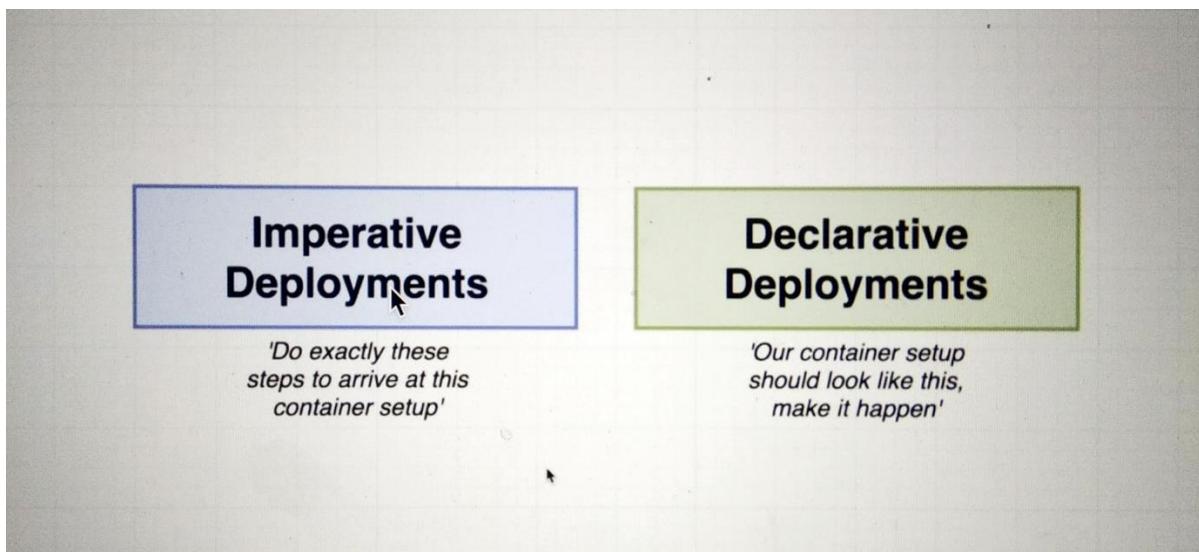
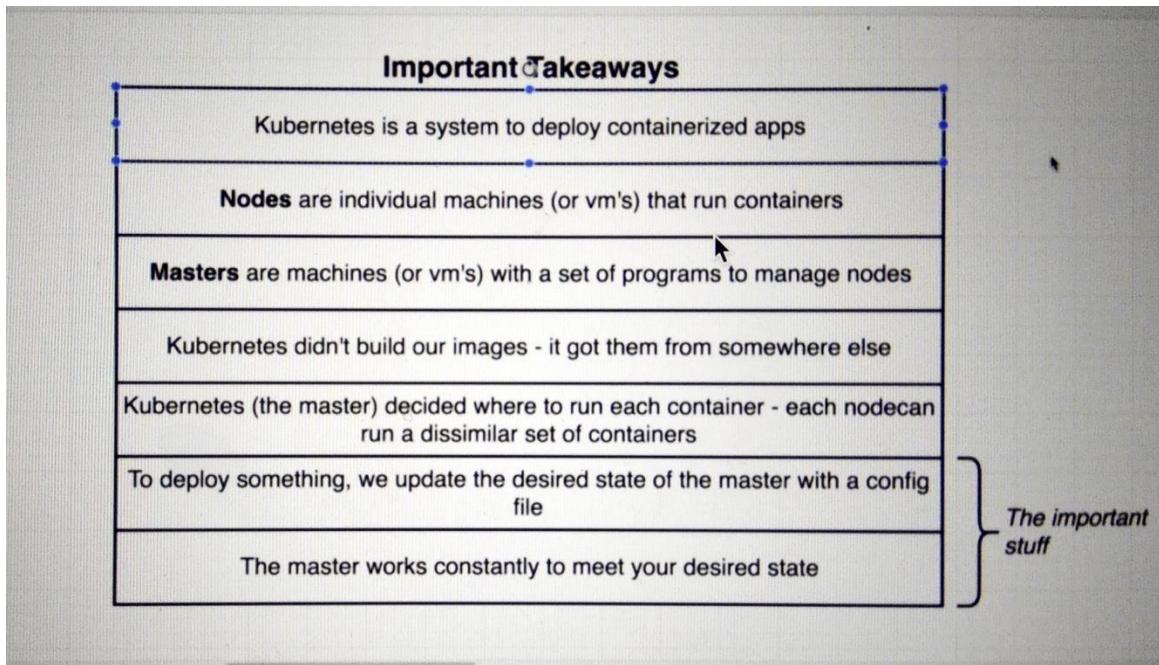
kube-apiserver responsible for monitoring entire nodes in kubernetes cluster. Master makes the list of responsibilities of all the commands that we give through config file.

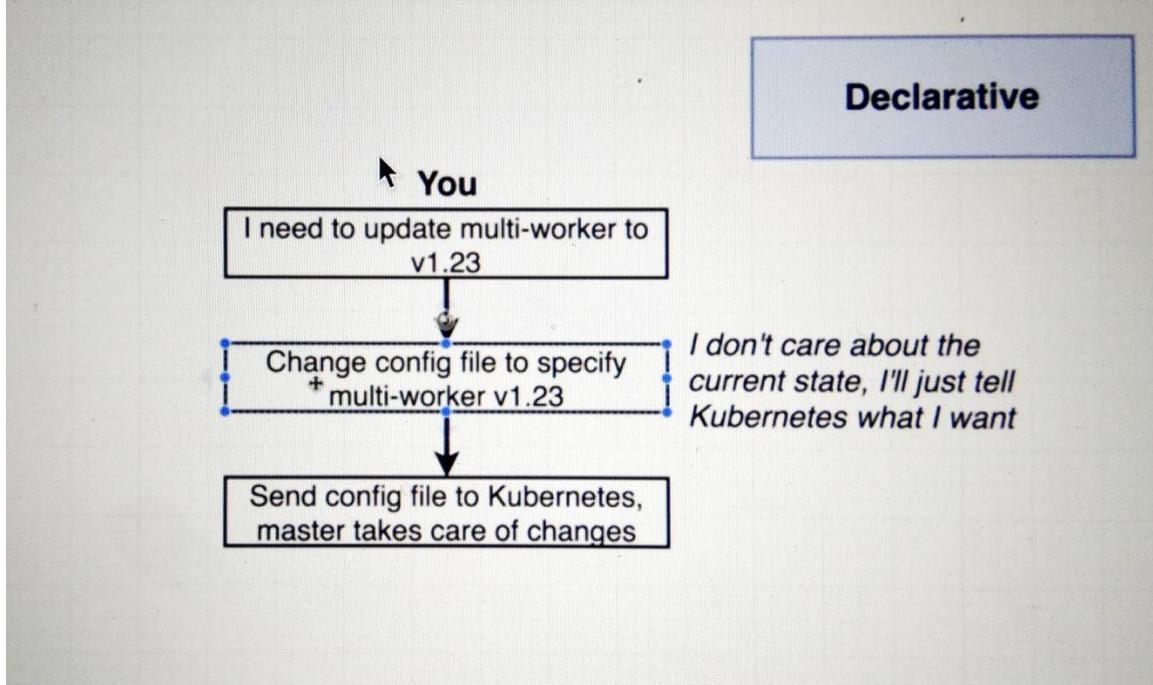
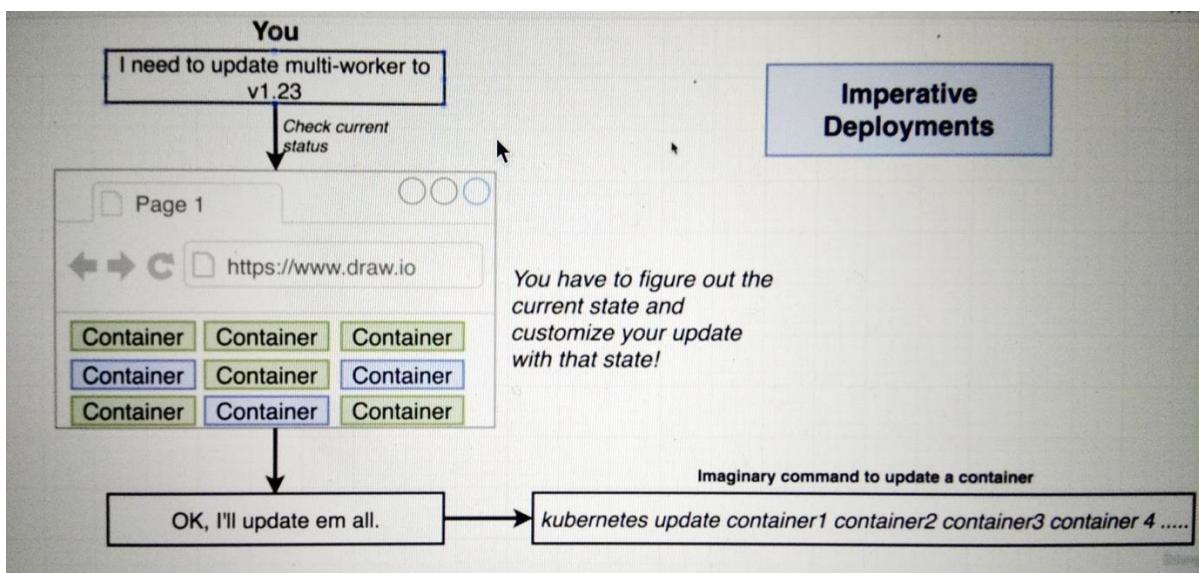
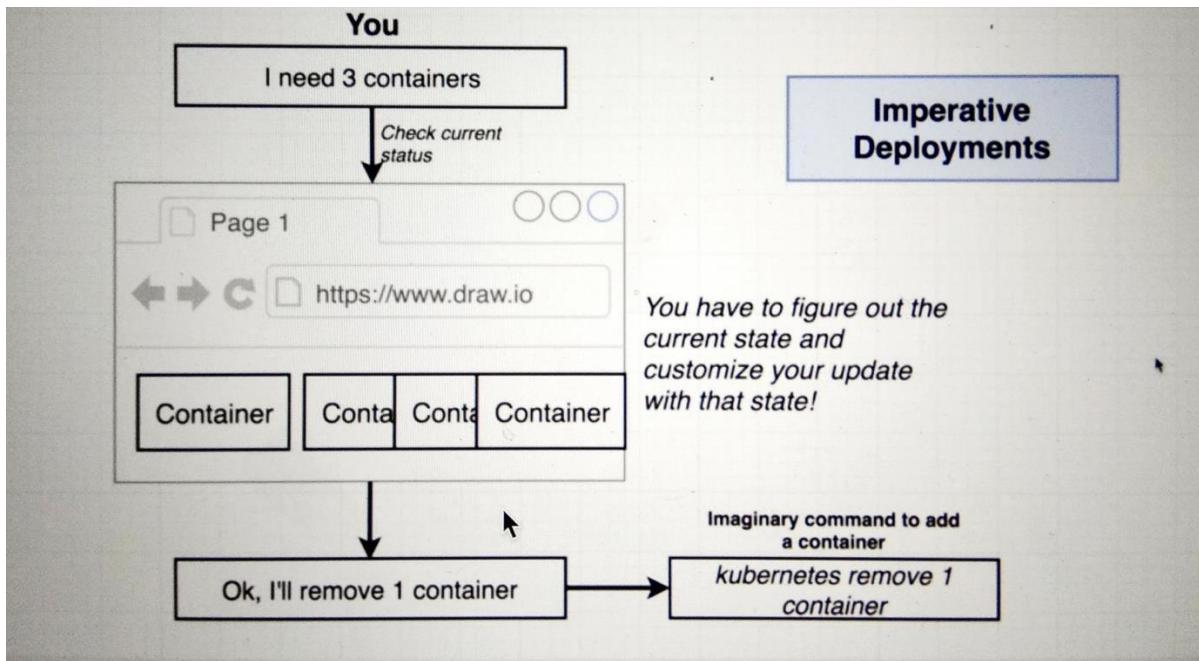
Master will check the responsibilities list, if it finds any mismatch between “# I need” and “# I have running”, it will run the respective command to match the same count. This is the reason why container recreated after we did the docker kill.

Docker inside the node will reach out docker hub and pull the image and run the docker.

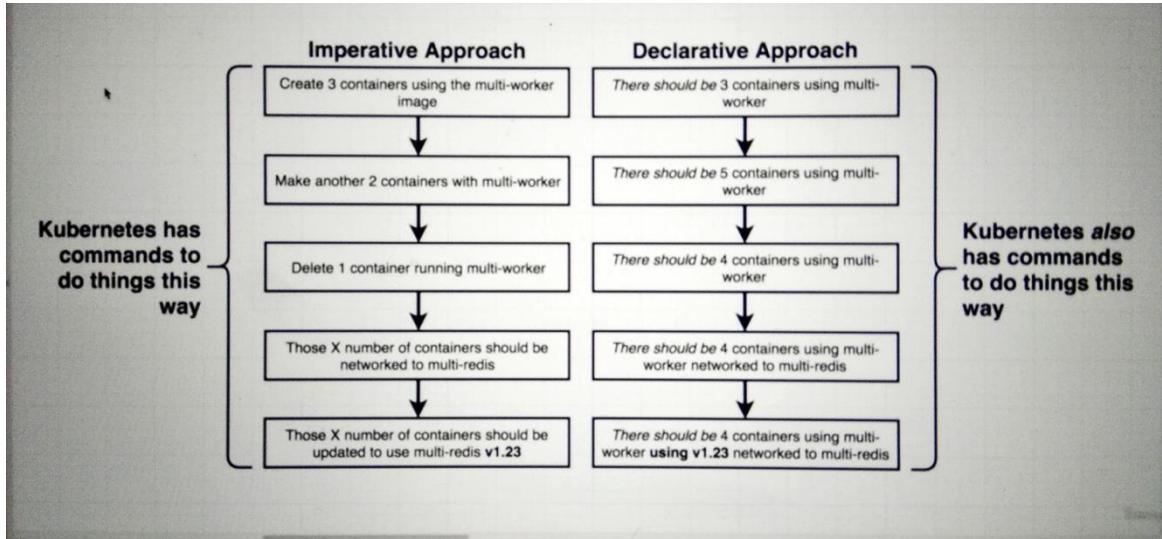


Imperative and Declarative Deployments





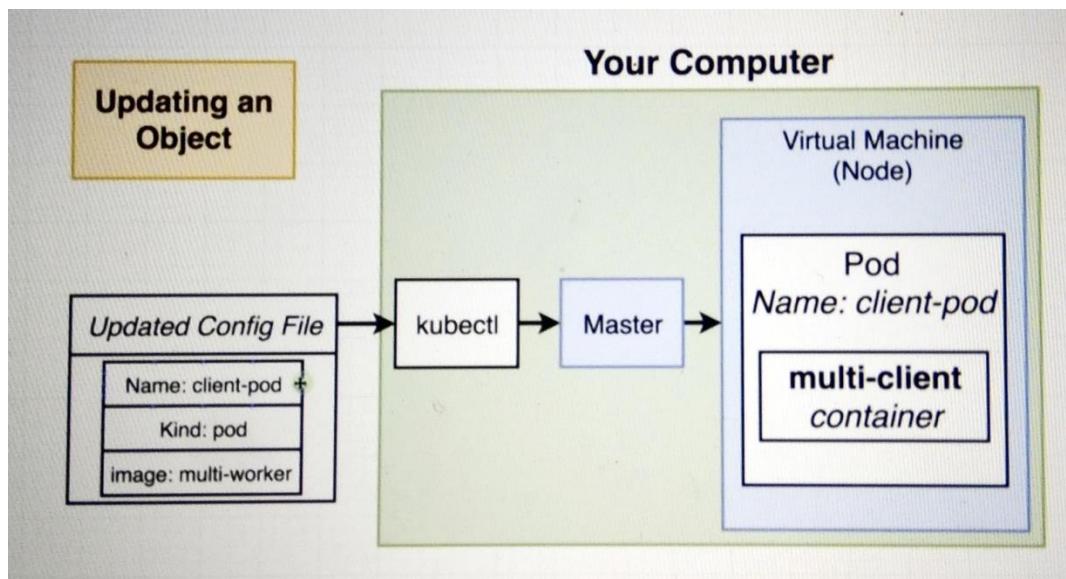
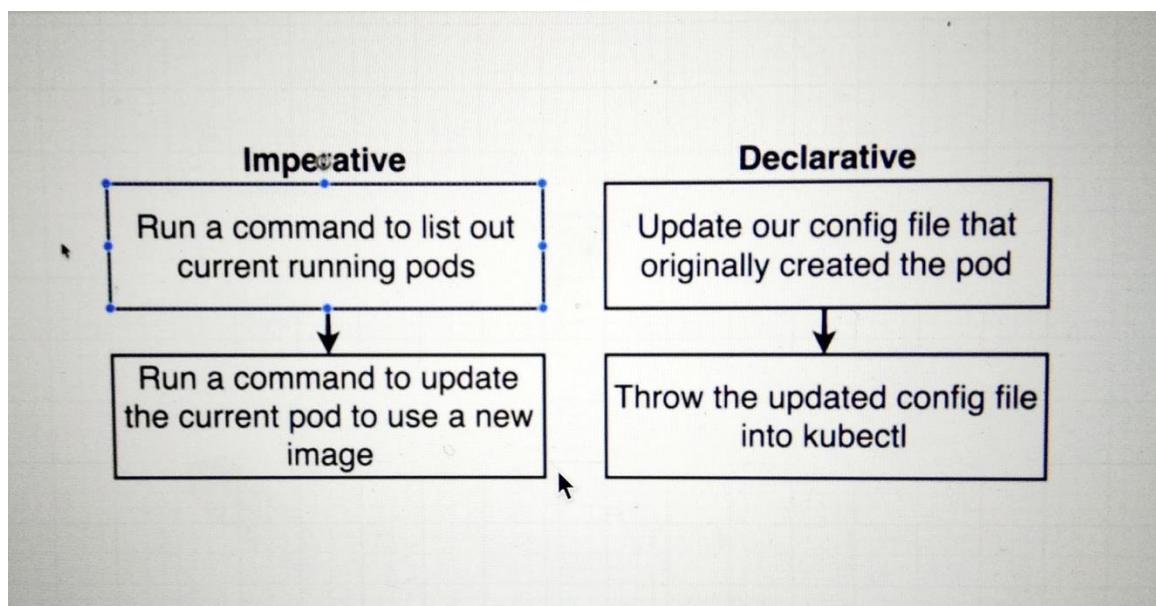
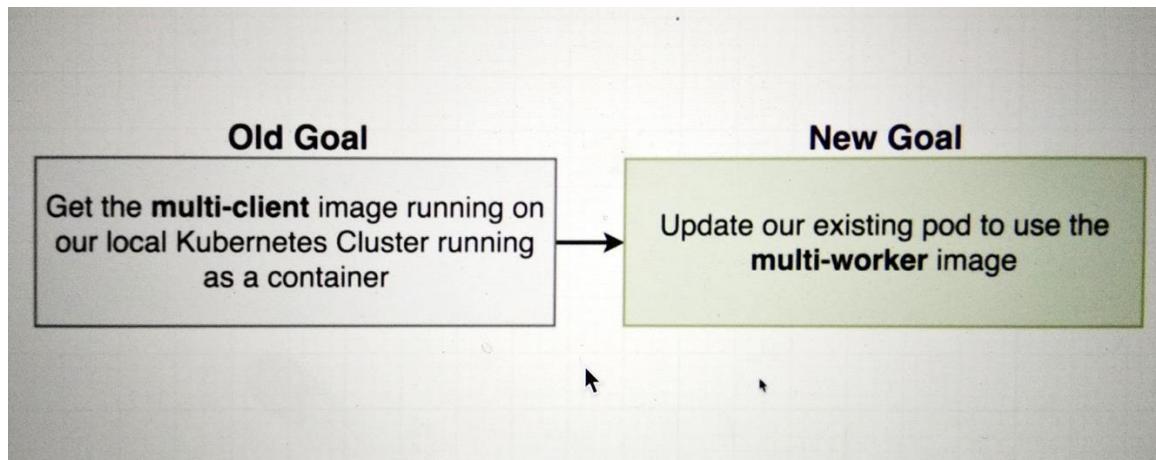
Comparatively declarative approach is an easy one.



Maintaining Set of Containers with Deployments

Updating Existing Object

Approach



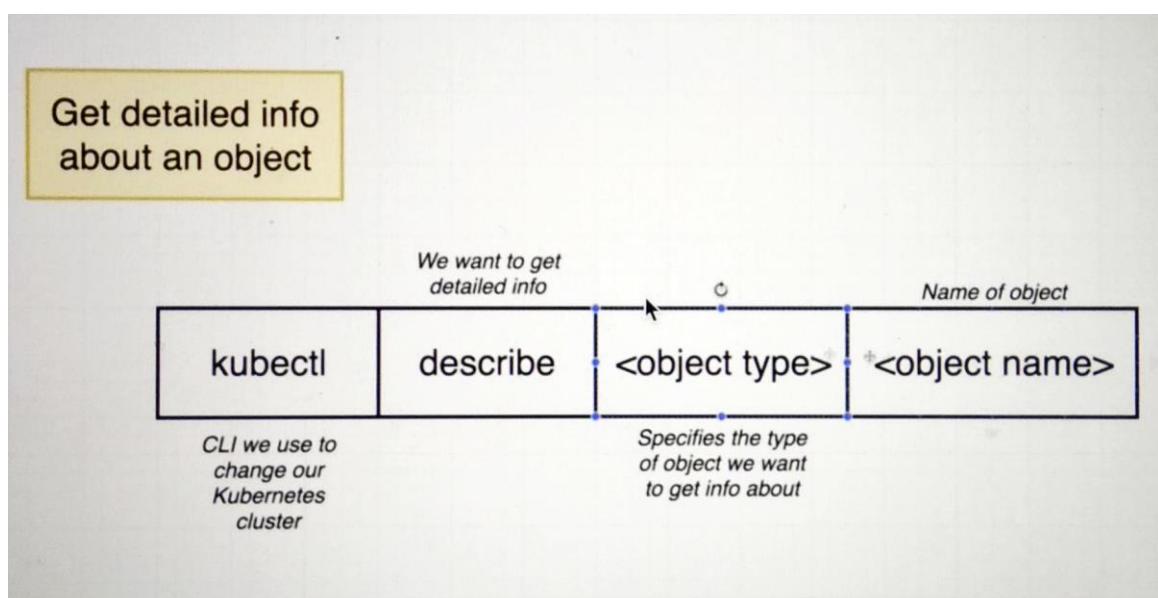
Changing the config file

```
04_simplek8s > 📄 client-pod.yaml
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: client-pod
5    labels:
6      component: web
7  spec:
8    containers:
9      - name: client
10     image: stephengrider/multi-worker
11     ports:
12       - containerPort: 3000
13
```

Applying the config file

```
+ 04_simplek8s git:(main) ✘ kubectl apply client-pod.yaml
error: must specify one -f and -k
+ 04_simplek8s git:(main) ✘ kubectl apply -f client-pod.yaml
pod/client-pod configured
+ 04_simplek8s git:(main) ✘ kubectl apply -f client-node-port.yaml
service/client-node-port unchanged
+ 04_simplek8s git:(main) ✘ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
client-pod  1/1     Running   3 (2m28s ago)  46h
+ 04_simplek8s git:(main) ✘ kubectl get services
NAME          TYPE     CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
client-node-port  NodePort  10.99.156.34  <none>      3050:31515/TCP  46h
kubernetes    ClusterIP  10.96.0.1    <none>      443/TCP   6d15h
+ 04_simplek8s git:(main) ✘ docker ps
CONTAINER ID   IMAGE           COMMAND           CREATED          STATUS          PORTS          NAMES
5d2fdcee2c13  stephengrider/multi-worker "npm run start"  14 seconds ago   Up 14 seconds   0.0.0.0:3000->3000/tcp
k8s_client_client-pod_default_5945b3b0-6f3d-4bbd-8def-0f615_c6792fe_4
+ 04_simplek8s git:(main) ✘
```

Inspecting the specific pod



```

l079218_4
→ 04_simplek8s git:(main) ✘ kubectl describe pod client-pod
Name:           client-pod
Namespace:      default
Priority:      0
Node:          docker-desktop/192.168.65.4
Start Time:    Sun, 17 Apr 2022 14:24:21 +0530
Labels:         component=web
Annotations:   <none>
Status:        Running
IP:            10.1.0.26
 IPs:
 IP: 10.1.0.26
Containers:
client:
  Container ID:  docker://5d2fdce2c13159e681055a0757b79753c75445b8ceb141a9329599c7f076c3b
  Image:         stephengrider/multi-worker
  Image ID:     docker-pullable://stephengrider/multi-worker@sha256:5fbab5f86e6a4d499926349a5f0ec032c42e7f7450acc98b053791df26dc4d2b
  Port:          3000/TCP
  Host Port:    0/TCP
  State:        Running
    Started:    Tue, 19 Apr 2022 13:19:51 +0530
  Last State:   Terminated
    Reason:     Completed
    Exit Code:  0
    Started:    Tue, 19 Apr 2022 13:17:41 +0530
    Finished:   Tue, 19 Apr 2022 13:19:19 +0530
  Ready:        True
  Restart Count: 4
  Environment:  <none>
  Mounts:
    /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-jfhgt (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready       True
  ContainersReady  True
  PodScheduled  True
Volumes:
kube-api-access-jfhgt:
  Type:        Projected (a volume that contains injected data from multiple sources)
  TokenExpirationSeconds: 3607
  ConfigMapName:  kube-root-ca.crt
  ConfigMapOptional: <nil>
  DownwardAPI:   true
QoS Class:      BestEffort
  QoS Class:    BestEffort
  Node-Selectors: <none>
  Tolerations:   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type      Reason     Age                    From      Message
  ----      ----     --                    ----      ---
  Normal    SandboxChanged  13h                  kubelet  Pod sandbox changed, it will be killed and re-created.
  Normal    Pulling      13h                  kubelet  Pulling image "stephengrider/multi-client"
  Normal    Pulled       13h                  kubelet  Successfully pulled image "stephengrider/multi-client" in 5.800051077s
  Normal    Created      13h                  kubelet  Created container client
  Normal    Started      13h                  kubelet  Started container client
  Normal    SandboxChanged  7m45s                kubelet  Pod sandbox changed, it will be killed and re-created.
  Normal    Pulling      7m45s                kubelet  Pulling image "stephengrider/multi-client"
  Normal    Pulled       7m40s                kubelet  Successfully pulled image "stephengrider/multi-client" in 4.16683186s
  Normal    Killing      6m2s                 kubelet  Container client definition changed, will be restarted
  Normal    Pulling      6m2s                 kubelet  Pulling image "stephengrider/multi-worker"
  Normal    Created      5m30s (x2 over 7m40s)  kubelet  Created container client
  Normal    Started      5m30s (x2 over 7m40s)  kubelet  Started container client
  Normal    Pulled       5m30s                kubelet  Successfully pulled image "stephengrider/multi-worker" in 31.712919588s

```

Limitation in config update

We are trying to update the container port in client-pod.yaml file.

```

04_simplek8s > 📄 client-pod.yaml
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: client-pod
5    labels:
6      component: web
7  spec:
8    containers:
9      - name: client
10        image: stephengrider/multi-worker
11        ports:
12          - containerPort: 9999
13

```

```

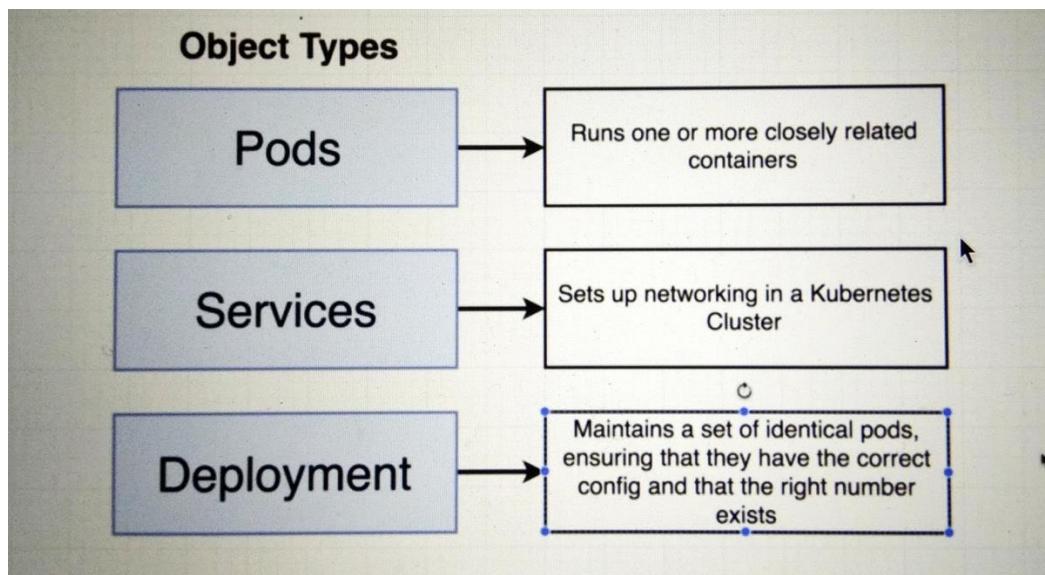
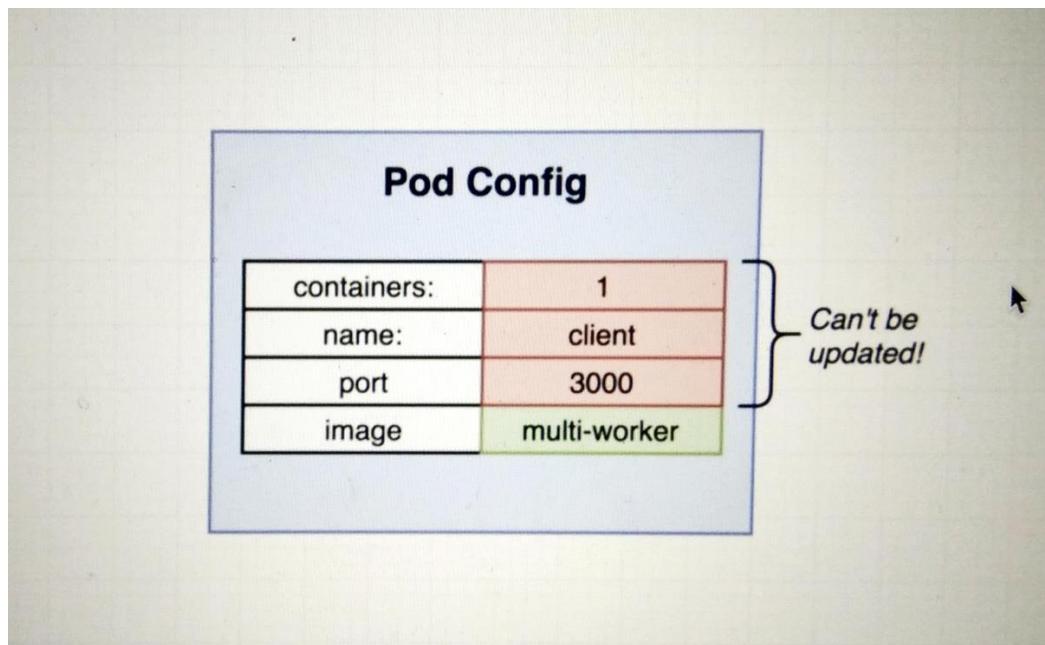
+-- Normal  Pinned          3m36s   kubelet  successfully pulled image "stephengridel/multi-worker" in 317/12315580s
+ 04_simplek8s git:(main) ✘ kubectl apply -f client-pod.yaml
The Pod "client-pod" is invalid: spec: Forbidden: pod updates may not change fields other than `spec.containers[*].image`, `spec.initContainers[*].image`, `spec.act
iveDeadlineSeconds`, `spec.tolerations` (only additions to existing tolerations) or `spec.terminationGracePeriodSeconds` (allow it to be set to 1 if it was previous
ly negative)
  core.PodSpec{
    Volumes:      [{"Name: "kube-api-access-jfhgt", VolumeSource: {Projected: &{Sources: {{ServiceAccountToken: &{ExpirationSeconds: 3607, Path: "token"}}, {Co
nfigMap: &{LocalObjectReference: {Name: "kube-root-ca.crt"}, Items: {{Key: "ca.crt", Path: "ca.crt"}}, {DownwardAPI: &{Items: {{Path: "namespace", FieldRef: &{API
Version: "v1", FieldPath: "metadata.namespace"}}, DefaultMode: &420}}}}, DefaultMode: &420}}}, {ContainerPort: 9999, Name: "", HostPort: 0, WorkingDir: ""}, Args: nil, InitContainers: nil, Containers: []core.Container{
      {
        ...
        // 3 identical fields
      }
    }
  }

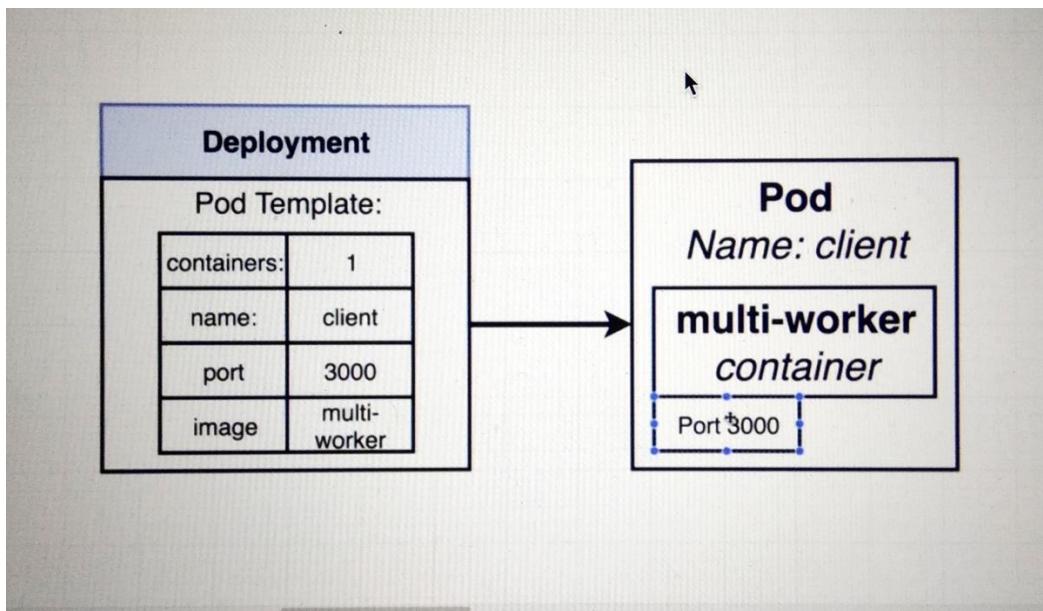
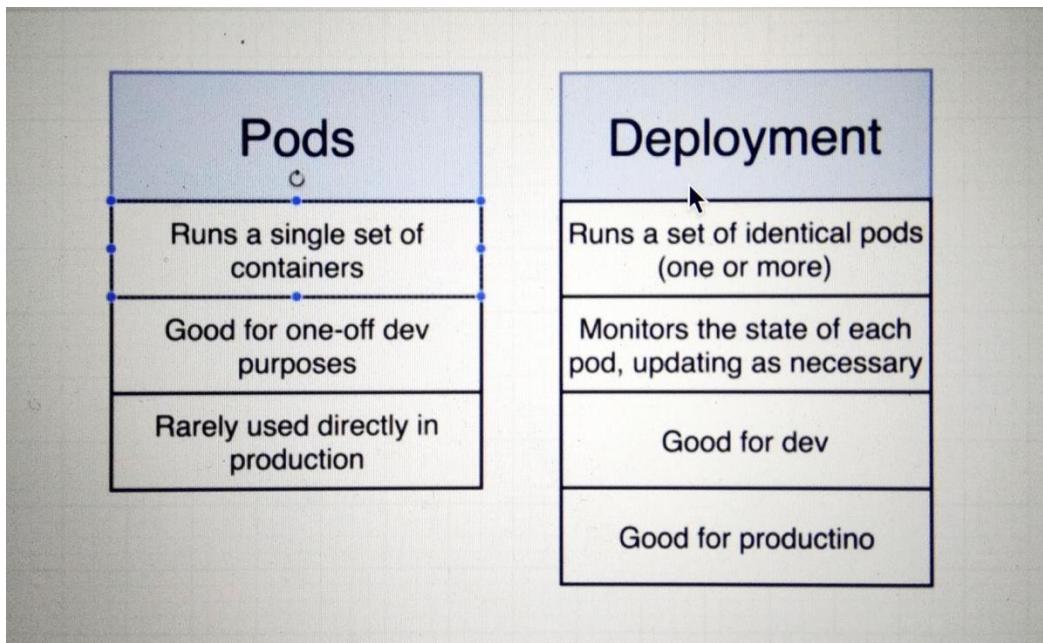
```

When we tried to apply the config file to kubernetes we get above error. Only specified property can be updated in config file and applied to kubernetes.

Deployment Object

Running containers with deployment





Deployment has a pod template which has all the configuration info. When this configuration gets deployed to kubernetes cluster, container will be created as per the configuration.

When there is a change in configuration, existing container will be killed and created again as per the configuration.

Deployment object constantly watching all the pods and make sure everything maintains a correct state.

Deployment Config File

```
04_simplek8s > client-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: client-deployment
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
20
```

appVersion: app/v1 => Available object type

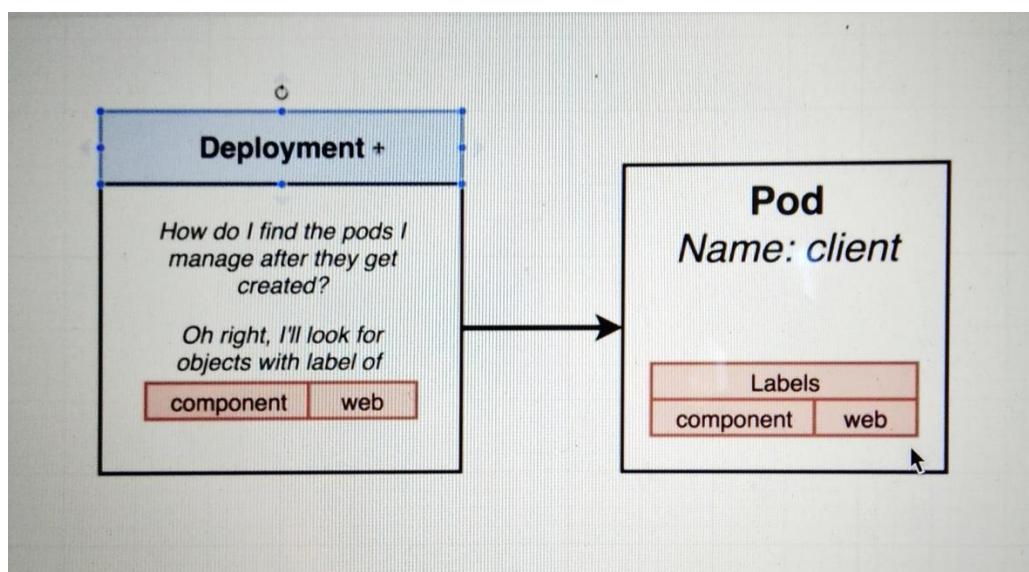
kind => type of object. Here it is Deployment

metadata: name: client-deployment => Name of the object.

Template => defines configuration for every pod that's get created.

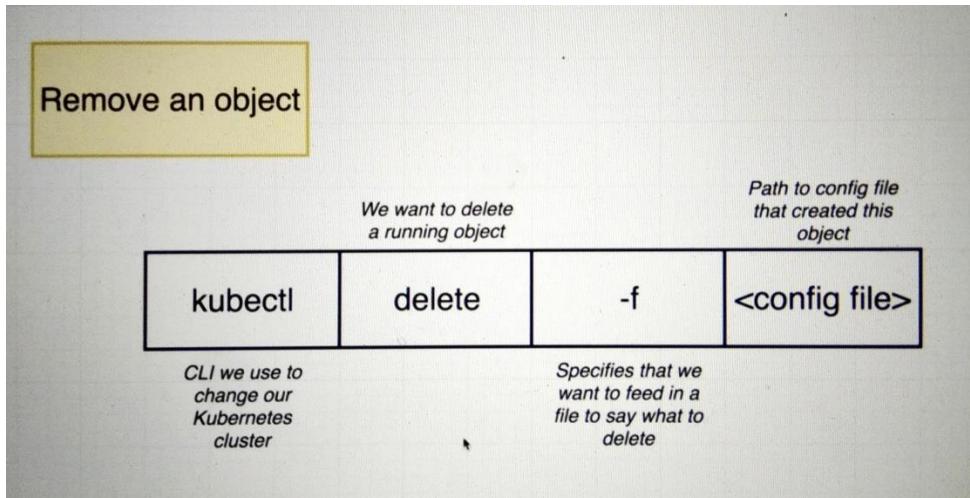
Replicas => represent a number of pods to create.

Selector work like below,



Applying Deployment

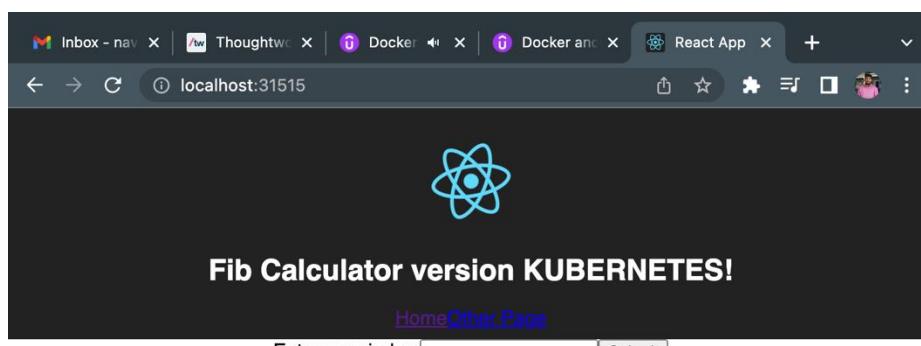
Removing existing object.



```
→ DOCKER_KUBE git:(main) ✘ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
client-pod     1/1     Running   22 (37s ago) 8d
→ DOCKER_KUBE git:(main) ✘ cd 04_simplek8s
→ 04_simplek8s git:(main) ✘ kubectl delete -f client-pod.yaml
pod "client-pod" deleted
→ 04_simplek8s git:(main) ✘ █
```

Applying deployment config file.

```
✗ command not found: kubectl
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment created
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
client-deployment-7cb6c958f7-6nq7h 1/1     Running   0          11s
→ 04_simplek8s git:(main) ✘ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment 1/1       1           1          96s
→ 04_simplek8s git:(main) ✘ █
```



Indexes I have seen:

Calculated Values:

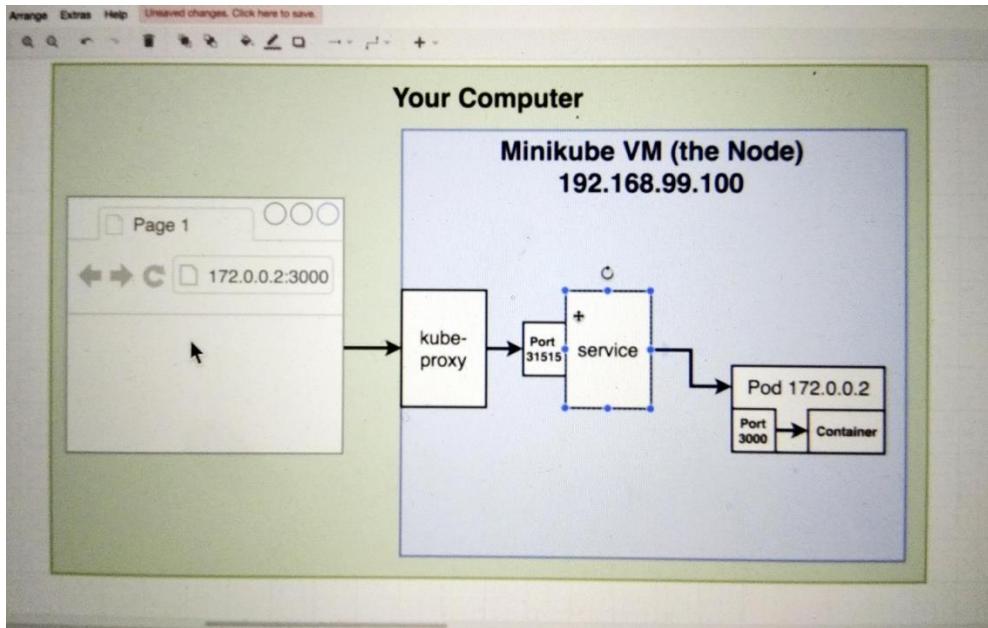
```

client deployment 1/1
+ 04_simplek8s git:(main) ✘ kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED NODE   READINESS
GATES
client-deployment-7cb6c958f7-6nq7h   1/1     Running   0        6m49s   10.1.0.39   docker-desktop <none>       <none>
+ 04_simplek8s git:(main) ✘

```

Every pod assigned with IP Address.

Use of service object



If try to access the pod directly using the pod IP address, on time of pod crash, pod will be deleted and created again. Newly created pod will be assigned with new IP address. In client side we must change the URL. This is the problem we face when every time pod gets created.

By using the service object, based on the selector, request will be navigated to right pod.

Scaling and changing the deployments

```

components: web
template:
  metadata:
    labels:
      component: web
  spec:
    containers:
      - name: client
        image: stephengrider/multi-client
        ports:
          - containerPort: 9999

```

Changed the container port to 9999.

```

DESKTOP-<none> ~ <none>
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME READY STATUS RESTARTS AGE
client-deployment-cb665f48d-pw4w9 1/1 Running 0 35s
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME READY STATUS RESTARTS AGE
client-deployment-cb665f48d-pw4w9 1/1 Running 0 47s
→ 04_simplek8s git:(main) ✘

```

We just applied the deployment config file to kubernetes. Pod is recreated again as we can see from AGE.

Scaling up pods

```

client-deployment.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: client-deployment
5 spec:
6   replicas: 5
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: 9999

```

```

client-deployment-cb665f48d-pw4w9 1/1 Running 0 47s
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME READY STATUS RESTARTS AGE
client-deployment-cb665f48d-486v5 0/1 ContainerCreating 0 2s
client-deployment-cb665f48d-7s4z5 0/1 ContainerCreating 0 2s
client-deployment-cb665f48d-hp8zd 0/1 ContainerCreating 0 2s
client-deployment-cb665f48d-pv8ng 0/1 ContainerCreating 0 2s
client-deployment-cb665f48d-pw4w9 1/1 Running 0 9m9s
→ 04_simplek8s git:(main) ✘ kubectl get deployment
NAME READY UP-TO-DATE AVAILABLE AGE
client-deployment 5/5 5 5 40m
→ 04_simplek8s git:(main) ✘

```

Updating the image in deployment config file.

```

COMPONENT: WEB
template:
  metadata:
    labels:
      component: web
spec:
  containers:
    - name: client
      image: stephengrider/multi-worker
      ports:
        - containerPort: 9999

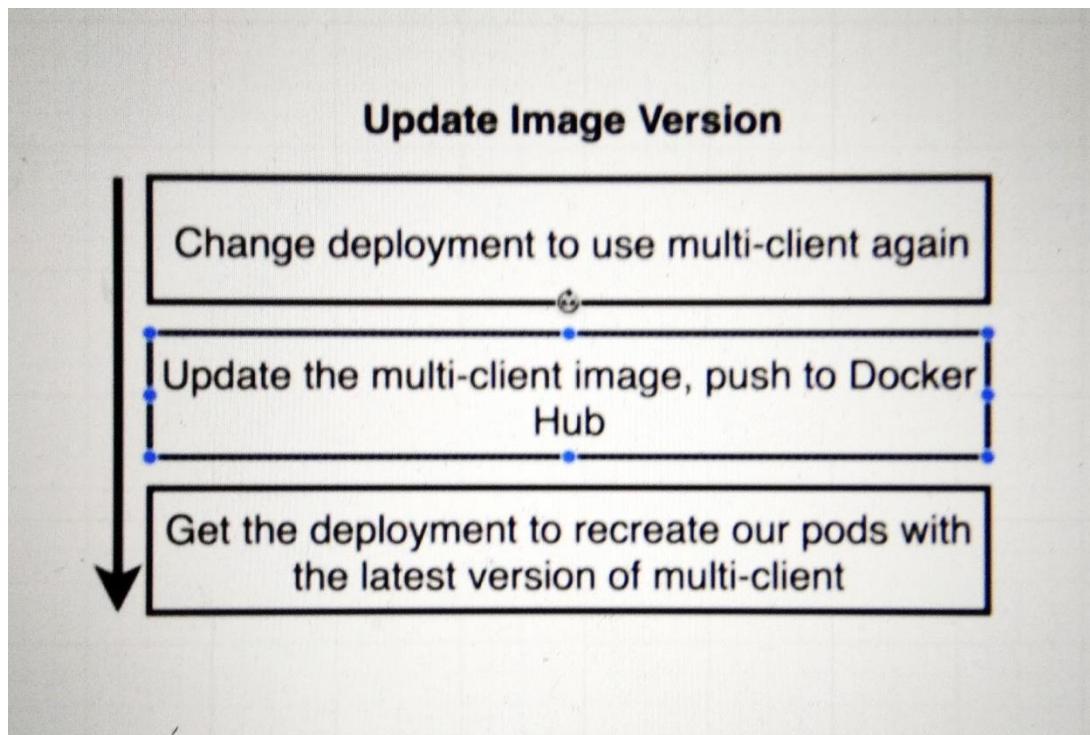
```

```
DESKTOP-simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
→ DESKTOP-simplek8s git:(main) ✘ kubectl get pods
→ DESKTOP-simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
→ DESKTOP-simplek8s git:(main) ✘ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  4/5      3           4           60m
→ DESKTOP-simplek8s git:(main) ✘ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  4/5      4           4           60m
→ DESKTOP-simplek8s git:(main) ✘ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  4/5      4           4           60m
```

```
client-deployment  4/5      5           4           60m
→ DESKTOP-simplek8s git:(main) ✘ kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  5/5      5           5           61m
→ DESKTOP-simplek8s git:(main) ✘ █
```

Once changed the image in deployment. Not all the pods recreated immediately. Initially, 3 pods recreated followed by 4 and 5.

Updating Deployment Image



Updating the latest version of the image.

We made the following changes to deployment config file.

```

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

```

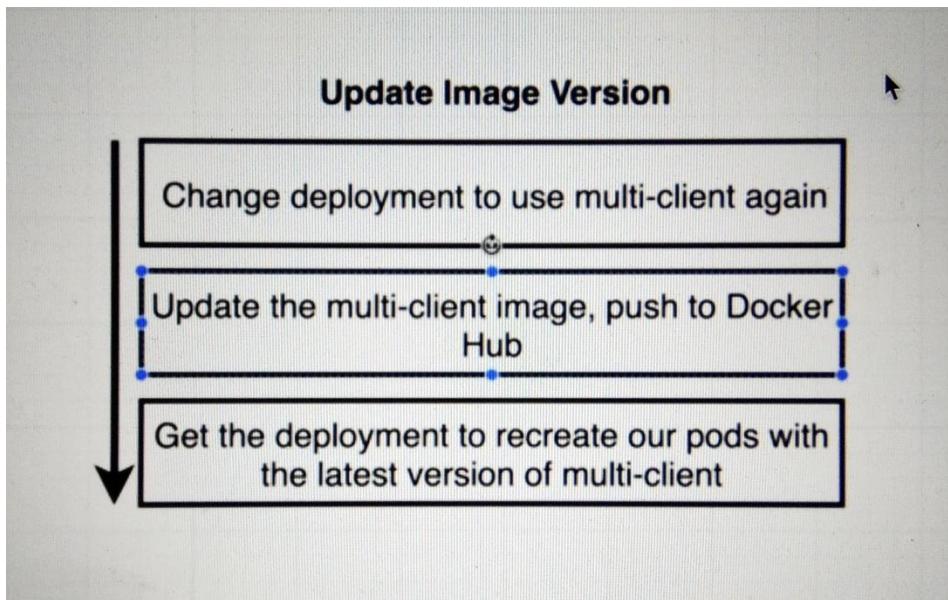
Applying the latest config file to kubernetes cluster and checking the result.

```

→ DOCKER_KUBE git:(main) ✘ cd 04_simplek8s
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
client-deployment-57f4f5b4c7-bf99m  1/1    Running   1 (20m ago)  46h
client-deployment-57f4f5b4c7-k9m4z  1/1    Running   1 (20m ago)  46h
client-deployment-57f4f5b4c7-mr8rf  1/1    Running   1 (20m ago)  46h
client-deployment-57f4f5b4c7-r7k8g  1/1    Running   1 (20m ago)  46h
client-deployment-57f4f5b4c7-snr4r  1/1    Running   1 (20m ago)  46h
→ 04_simplek8s git:(main) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment configured
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
client-deployment-57f4f5b4c7-k9m4z  1/1    Running   1 (21m ago)  46h
client-deployment-7cb6c958f7-kc52d  0/1    ContainerCreating   0           4s
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
client-deployment-57f4f5b4c7-k9m4z  1/1    Running   1 (21m ago)  46h
client-deployment-7cb6c958f7-kc52d  0/1    ContainerCreating   0           9s
→ 04_simplek8s git:(main) ✘
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
client-deployment-57f4f5b4c7-k9m4z  1/1    Running   1 (21m ago)  46h
client-deployment-7cb6c958f7-kc52d  0/1    ContainerCreating   0           11s
→ 04_simplek8s git:(main) ✘ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
client-node-port  NodePort    10.99.156.34 <none>       3050:31515/TCP  10d
kubernetes   ClusterIP  10.96.0.1    <none>       443/TCP       14d
→ 04_simplek8s git:(main) ✘ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
client-deployment-7cb6c958f7-kc52d  1/1    Running   0           28s
→ 04_simplek8s git:(main) ✘

```

Rebuilding the Client Image



```
App.js
8 component {
9
10
11
12   name="App">
13   className="App-header">
14   logo className="App-logo" alt="logo" />
15   ssName="App-title">Fib Calculator version 2</h1>
16   >/>Home</Link>
17   >/otherpage>Other Page</Link>
18
19
20   exact path="/" component={Fib} />
21   path="/otherpage" component={OtherPage} />
```

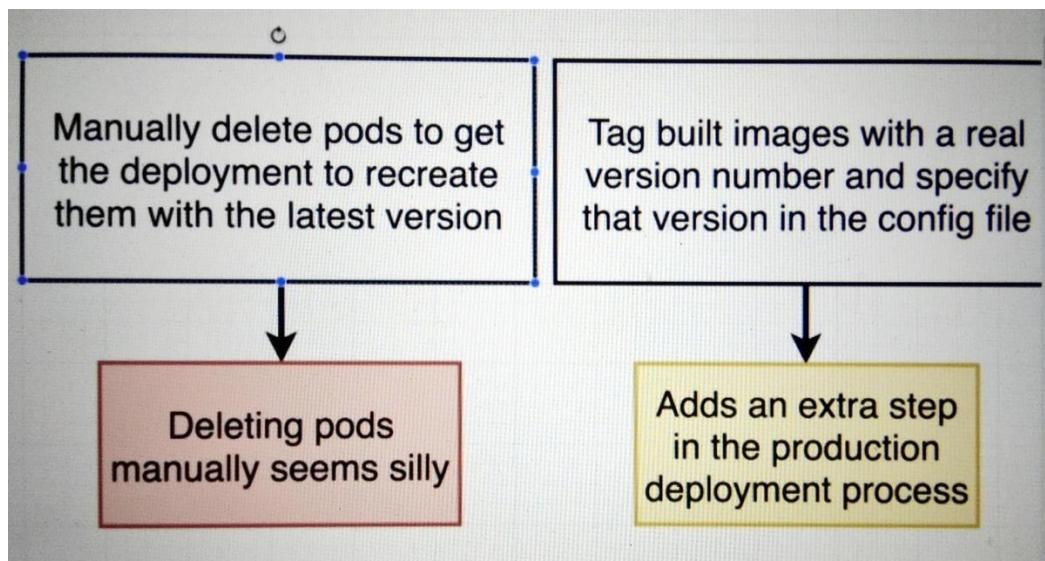
```
iTerm2 Shell Edit View Session Scripts Profiles Toolkit Window Help
1. docker build -t stephengrider/multi-client (docker)
→ client git:(master) ✘ docker build -t stephengrider/multi-client .
Sending build context to Docker daemon    574kB
Step 1/10 : FROM node:alpine as builder
--> 5a519d1e3a24
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 . .
--> 9058788e6880
Step 6/10 : RUN npm run build
--> Running in be8158ecf5ea
```

```
Step 8/10 : EXPOSE 3000
--> Using cache
--> edf8c36f5171
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
```

```
→ simplek8s git:(master) ✘ kubectl apply -f client-deployment.yaml
deployment.apps/client-deployment unchanged
→ simplek8s git:(master) ✘
```

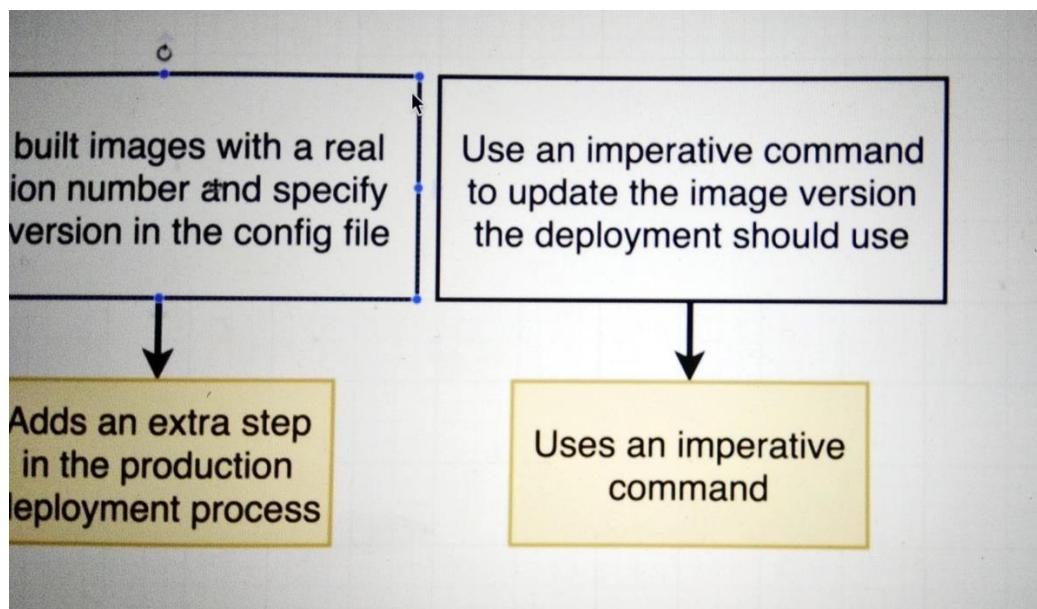
When we try to apply the config file, it says configuration file is unchanged. Our kubernetes cluster does not know, we have published new changes to our image.

There are three ways to overcome this problem.

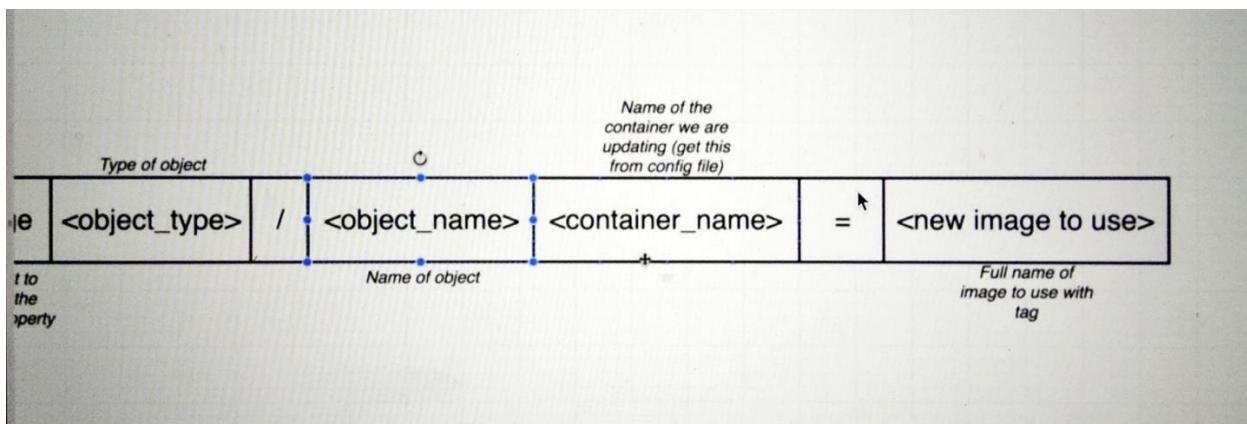
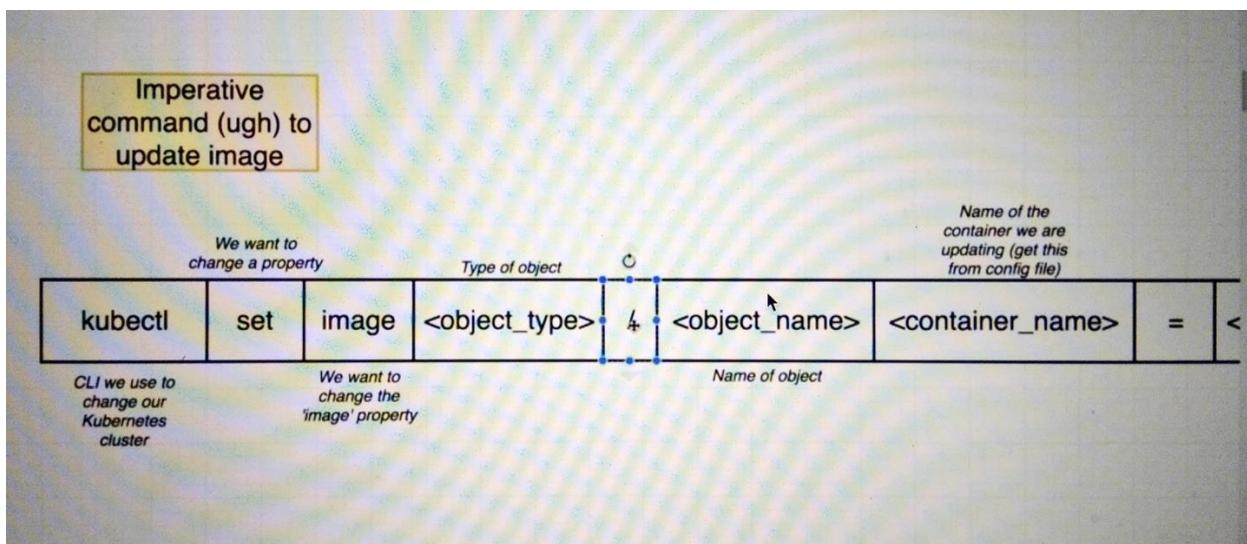
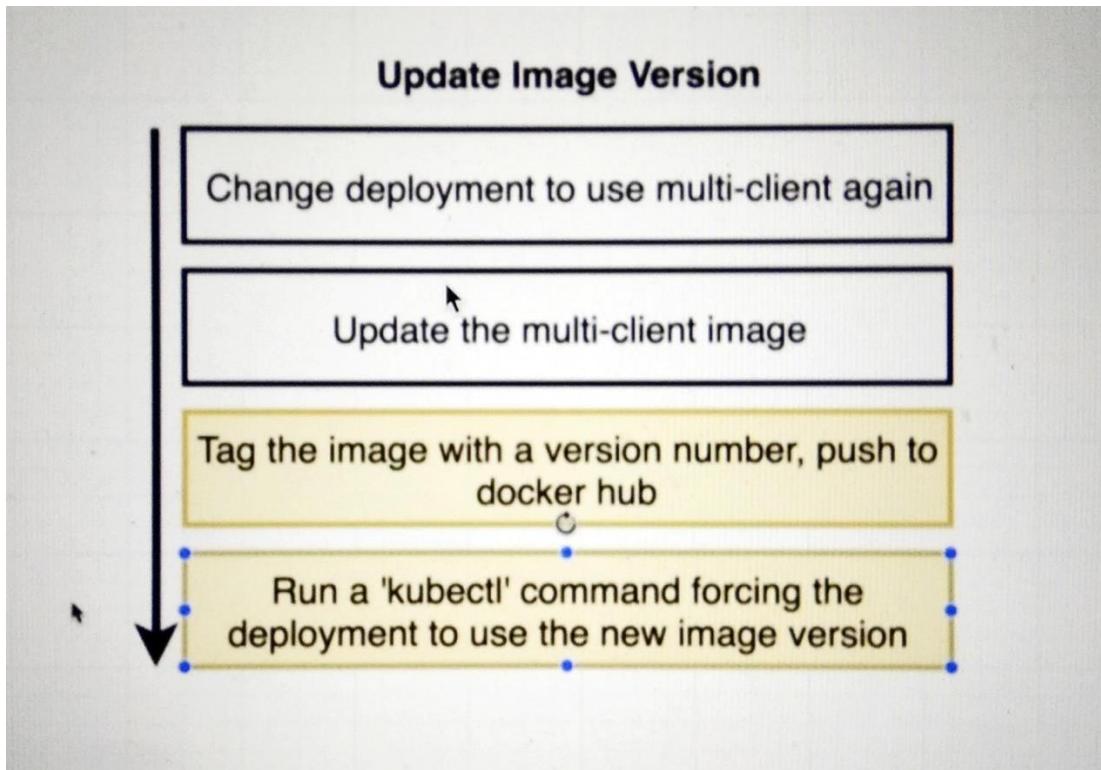


```
template:  
  metadata:  
    labels:  
      component: web  
spec:  
  containers:  
    - name: client  
      image: stephengrider/multi-client:v4  
    ports:  
      - containerPort: 3000
```

In the second approach, we must update the version number in config file whenever we build the image.



Imperatively updating a deployment image



```
→ simplek8s git:(master) ✘ kubectl set image deployment/client-deployment client=stephengrider/multi-client:v5
deployment.extensions/client-deployment: image updated
→ simplek8s git:(master) ✘
```

Configure VM to Use Docker Server

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

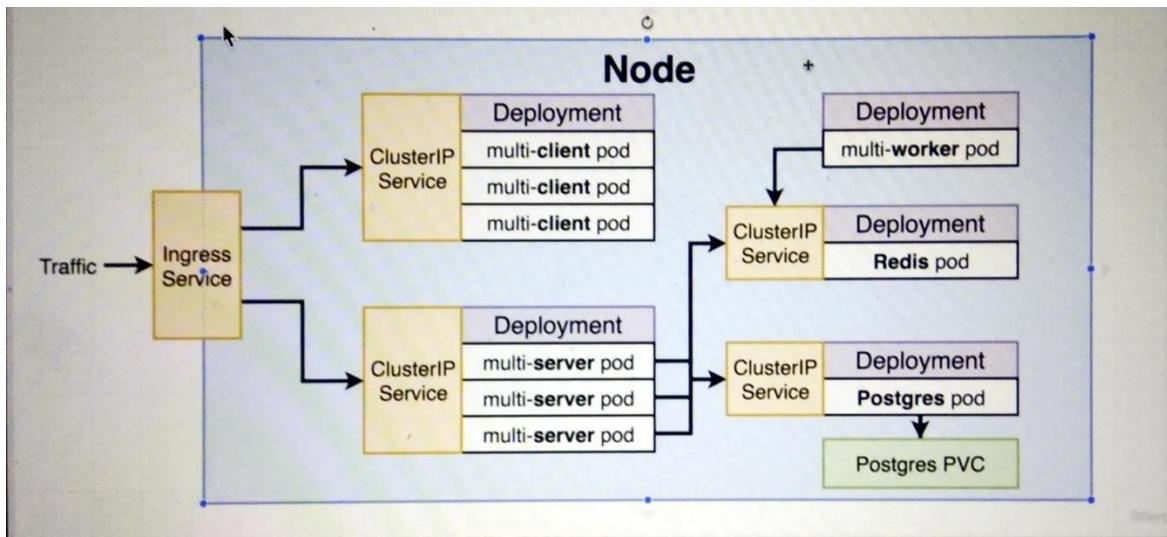
Many of these commands are available through kubectl

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

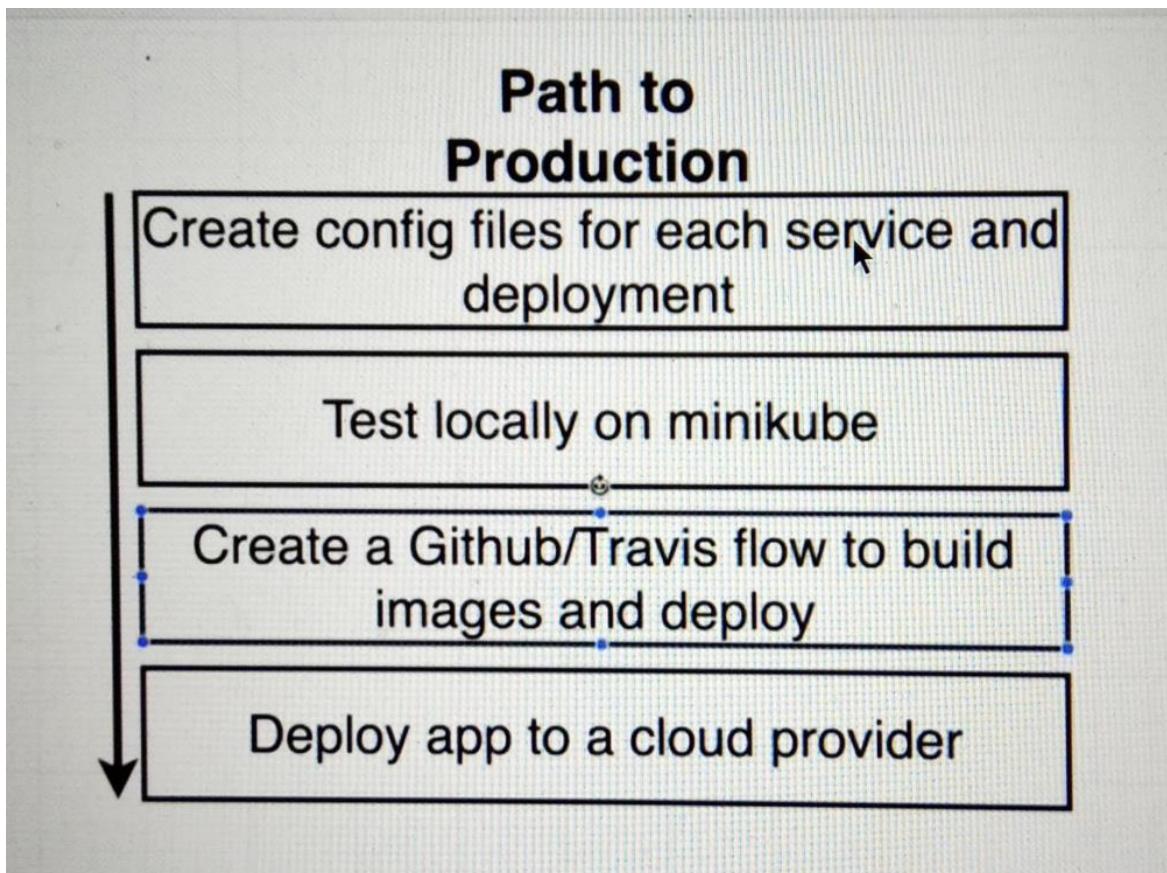
Delete cached images in the node

A Multi Container App with Kubernetes

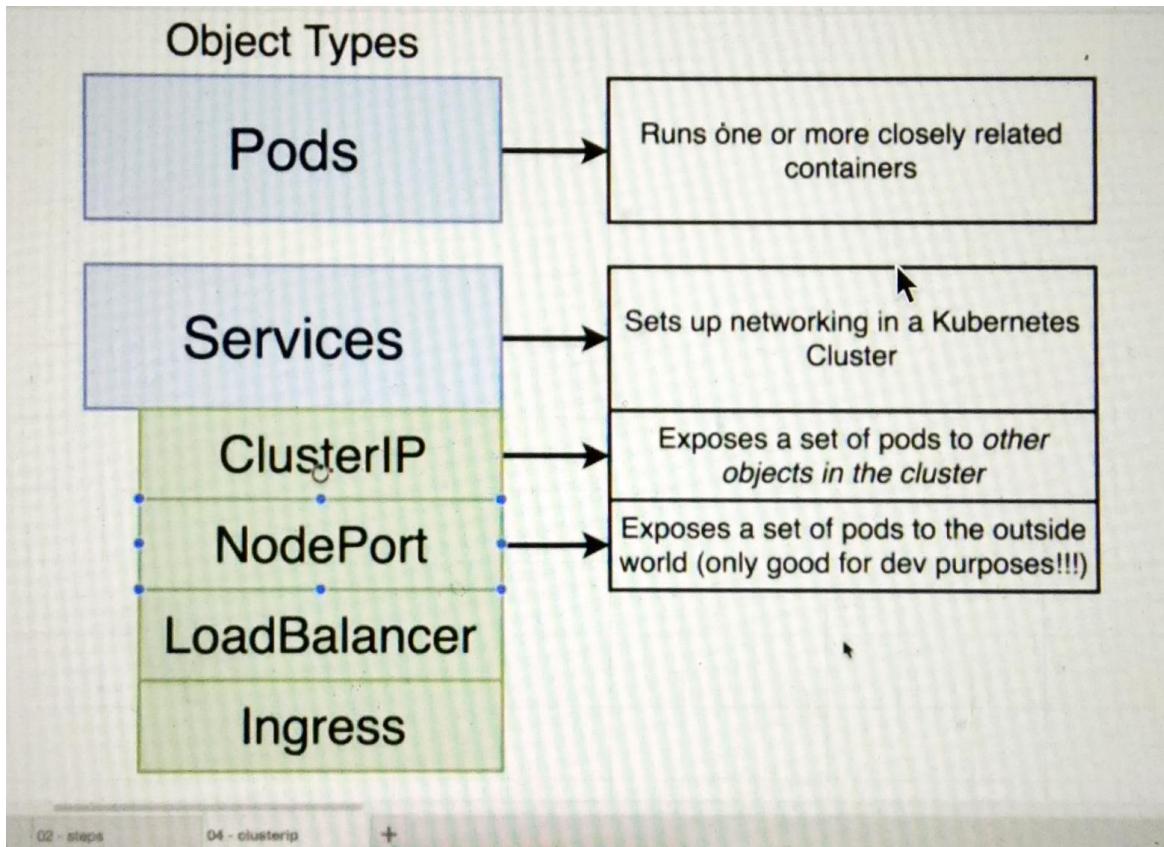
The Path to Production



Postgres PVC, Here PVC stands for Persistent Volume Claim.



Cluster IP vs Node Port



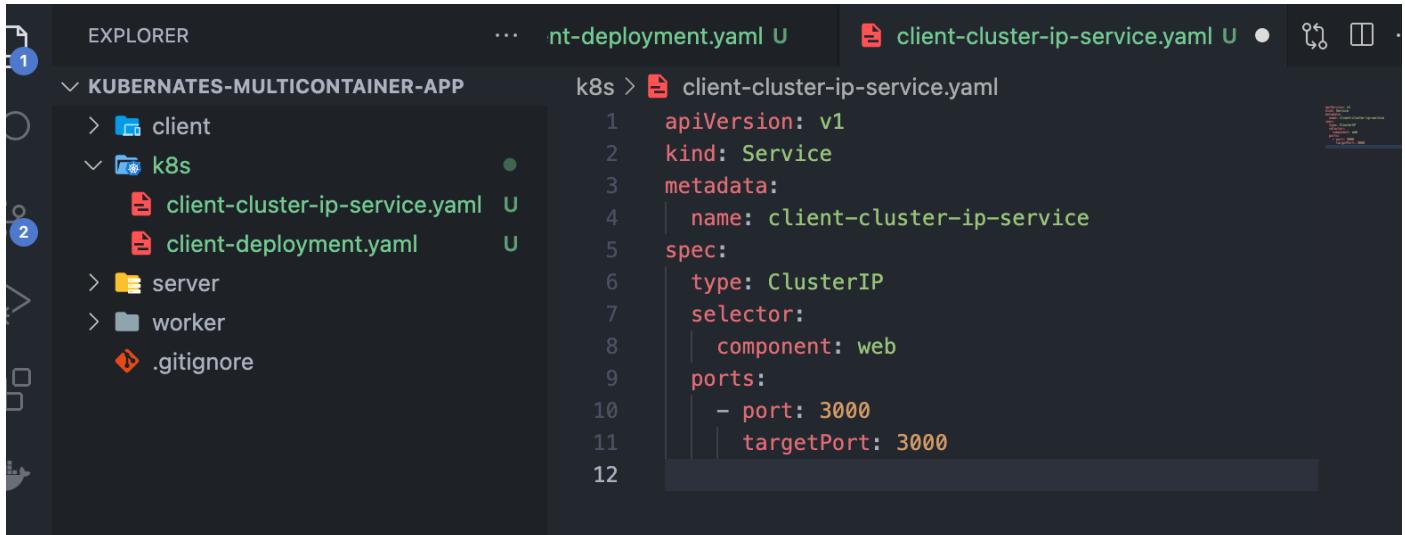
Kubernetes multi container application

Client deployment config file.

The screenshot shows a code editor with an "EXPLORER" sidebar and a main "client-deployment.yaml" file tab. The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: client-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      component: web
  template:
    metadata:
      labels:
        component: web
    spec:
      containers:
        - name: client
          image: stephengrider/multi-client
          ports:
            - containerPort: 3000
```

Client cluster IP service



The screenshot shows the VS Code interface with the Explorer sidebar open. The 'KUBERNATES-MULTICONTAINER-APP' folder is selected. Inside, there are several files and folders: 'client', 'k8s' (which contains 'client-cluster-ip-service.yaml' and 'client-deployment.yaml'), 'server', 'worker', and '.gitignore'. The 'client-cluster-ip-service.yaml' file is open in the main editor area. Its content is a YAML configuration for a Kubernetes Service:

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

Deploying the kubernetes configuration

Deleting Existing Deployment

```
/Users/naveen.kumar1/.zshrc:68: no matches found: load?
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  1/1     1          1          9d
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl delete deployment client-deployment
deployment.apps "client-deployment" deleted
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get deployments
No resources found in default namespace.
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

Deleting the node port services

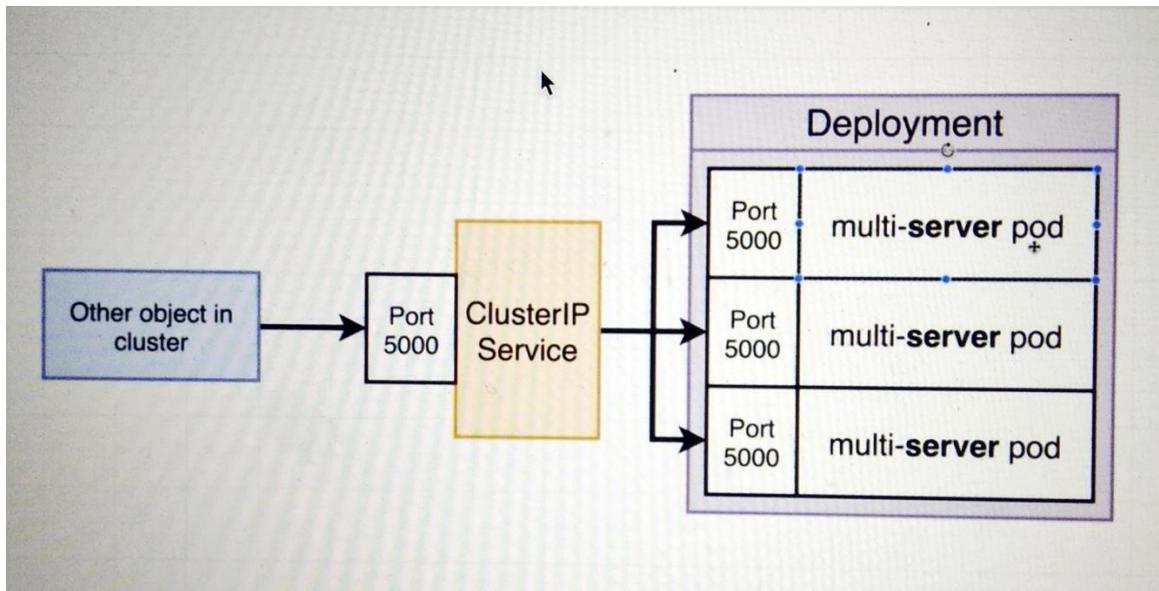
```
No resources found in default namespace.
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get services
NAME        TYPE        CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
client-node-port  NodePort    10.99.156.34  <none>        3050:31515/TCP  17d
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP       22d
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl delete services client-node-port
service "client-node-port" deleted
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get services
NAME        TYPE        CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP       22d
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

Deploying kubernetes configuration

```

→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
service/client-cluster-ip-service created
deployment.apps/client-deployment unchanged
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get deployments
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment   3/3     3            3           27s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
client-deployment-7cb6c958f7-mqf6c  1/1     Running   0          40s
client-deployment-7cb6c958f7-shgmv  1/1     Running   0          40s
client-deployment-7cb6c958f7-xpcgk  1/1     Running   0          40s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
client-cluster-ip-service   ClusterIP  10.98.174.100 <none>       3000/TCP  40s
kubernetes      ClusterIP  10.96.0.1    <none>       443/TCP   23d
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘

```



Express API Configuration

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure and files. The current file is `server-deployment.yaml`, which contains the following YAML configuration:

```

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

```

We can combine config file into a single file.

```
EXPLORER ... ~ip-service.yaml U server-config.yaml U X ⌂

KUBERNATES-MULTICONTAINER-APP
> client
< k8s
  client-cluster-ip-service.yaml U
  client-deployment.yaml U
  server-cluster-ip-service.... U
  server-config.yaml U
  server-deployment.yaml U
> server
> worker
.gitignore

k8s > server-config.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: server-deployment
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        component: server
10   template:
11     metadata:
12       labels:
13         component: server
14   spec:
15     containers:
16       - name: server
17         image: stephengrider/multi-server
18         ports:
19           - containerPort: 5000
20
21
22  apiVersion: v1
23  kind: Service
24  metadata:
25    name: server-cluster-ip-service
26  spec:
27    type: ClusterIP
28    selector:
29      component: server
30    ports:
31      - port: 5000
32      | targetPort: 5000
33
34
```

The Worker Deployment

```
EXPLORER ... ~ip-service.yaml U worker-deployment.yaml U ⌂

KUBERNATES-MULTI...
> client
< k8s
  client-cluster-ip-service.yaml U
  client-deployment.yaml U
  server-cluster-ip-service.... U
  server-deployment.yaml U
  worker-deployment.yaml U
> server
> worker
.gitignore

k8s > worker-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: worker-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        component: worker
10   template:
11     metadata:
12       labels:
13         component: worker
14   spec:
15     containers:
16       - name: worker
17         image: stephengrider/multi-worker
18
```

```

→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  3/3     3          3          51m
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ 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
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
client-deployment  3/3     3          3          51m
server-deployment  0/3     3          0          3s
worker-deployment  0/1     1          0          3s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
client-deployment-7cb6c958f7-mqf6c  1/1     Running   0          52m
client-deployment-7cb6c958f7-shgmv  1/1     Running   0          52m
client-deployment-7cb6c958f7-xpcgk  1/1     Running   0          52m
server-deployment-9bff8dfb-chc4b   1/1     Running   0          88s
server-deployment-9bff8dfb-rmvwq   1/1     Running   0          88s
server-deployment-9bff8dfb-s46s6   1/1     Running   0          88s
worker-deployment-666c96ffc5-tkdbj 1/1     Running   0          88s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ █

```

Creating and applying redis config file

```

k8s > 📄 redis-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: redis-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        component: redis
10   template:
11     metadata:
12       labels:
13         component: redis
14     spec:
15       containers:
16         - name: redis
17           image: redis
18           ports:
19             - containerPort: 6379
20

```

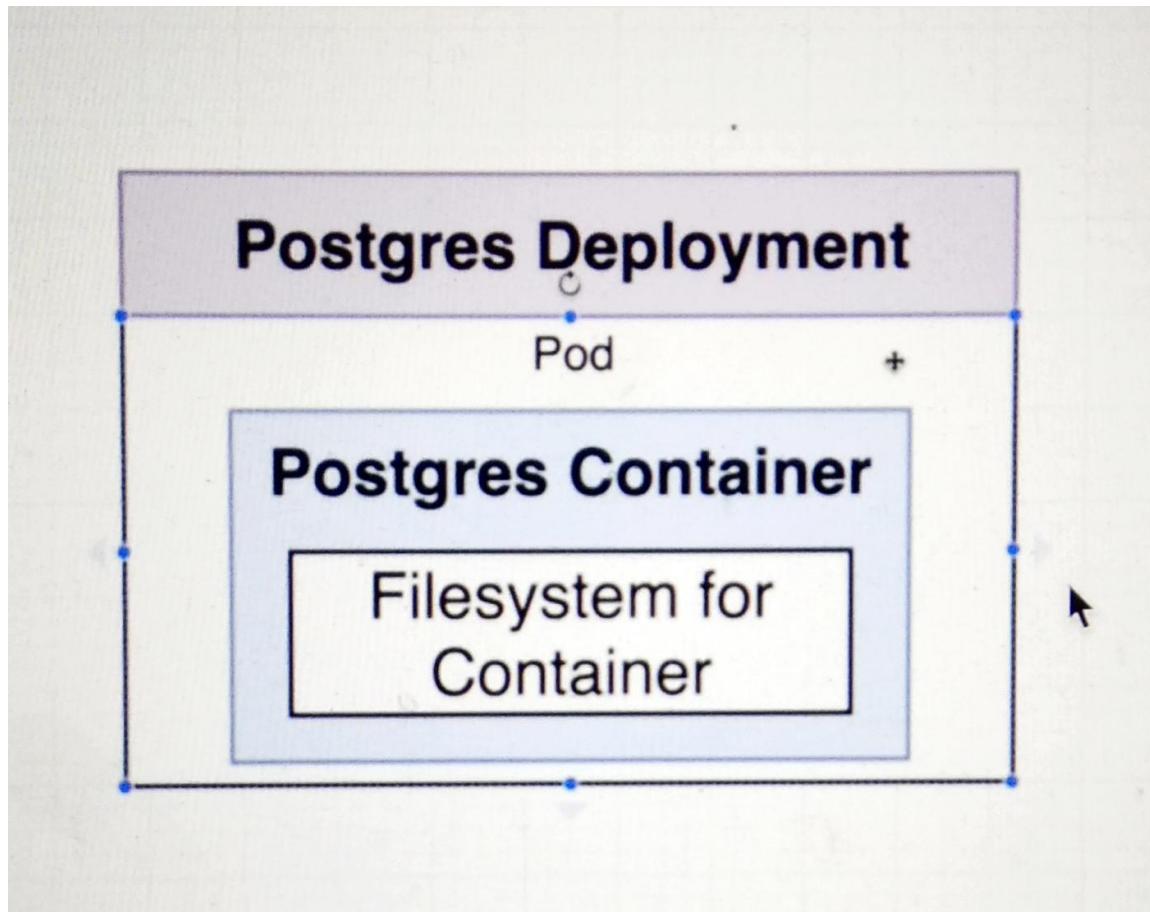
```
→ IN VERSION app/v1
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
service/redis-cluster-ip-service unchanged
deployment.apps/redis-deployment created
service/server-cluster-ip-service unchanged
deployment.apps/server-deployment unchanged
deployment.apps/worker-deployment unchanged
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

Creating and applying postgres config file

```
k8s > ⌂ postgres-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: postgres-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        component: postgres
10   template:
11     metadata:
12       labels:
13         component: postgres
14     spec:
15       containers:
16         - name: postgres
17           image: postgres
18           ports:
19             - containerPort: 5432
20
```

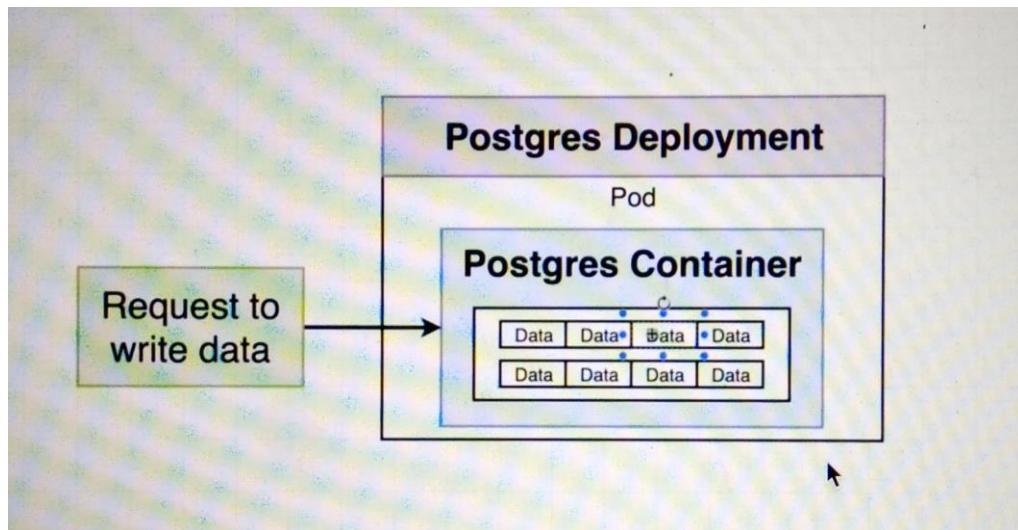
```
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
service/postgres-cluster-ip-service unchanged
deployment.apps/postgres-deployment created
service/redis-cluster-ip-service unchanged
deployment.apps/redis-deployment unchanged
service/server-cluster-ip-service unchanged
deployment.apps/server-deployment unchanged
deployment.apps/worker-deployment unchanged
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

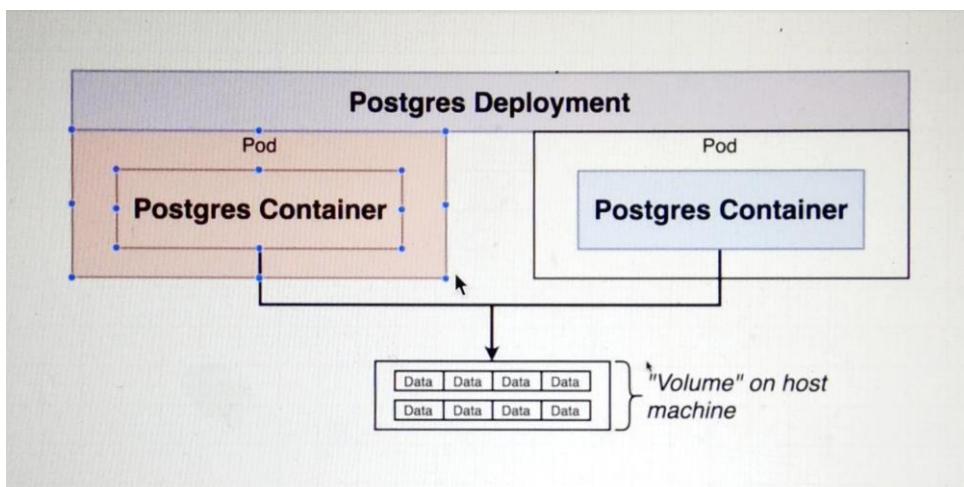
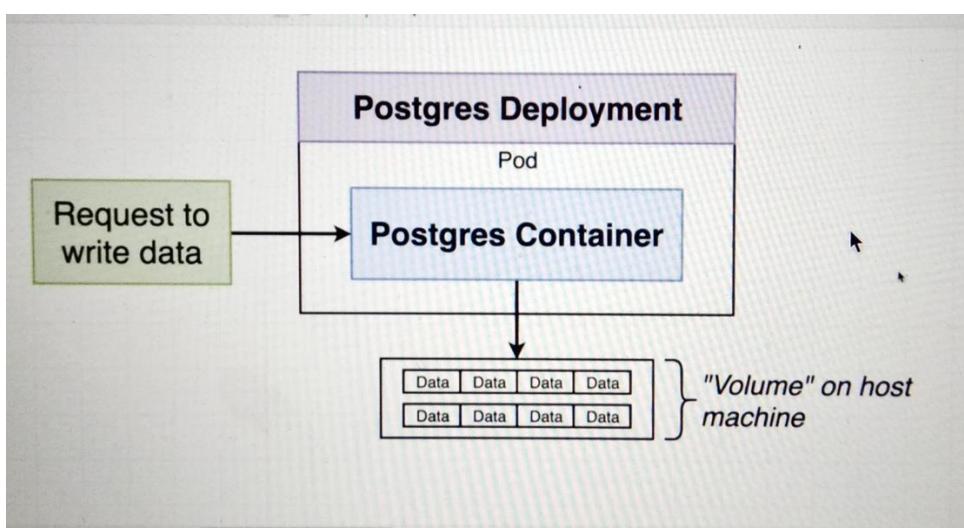
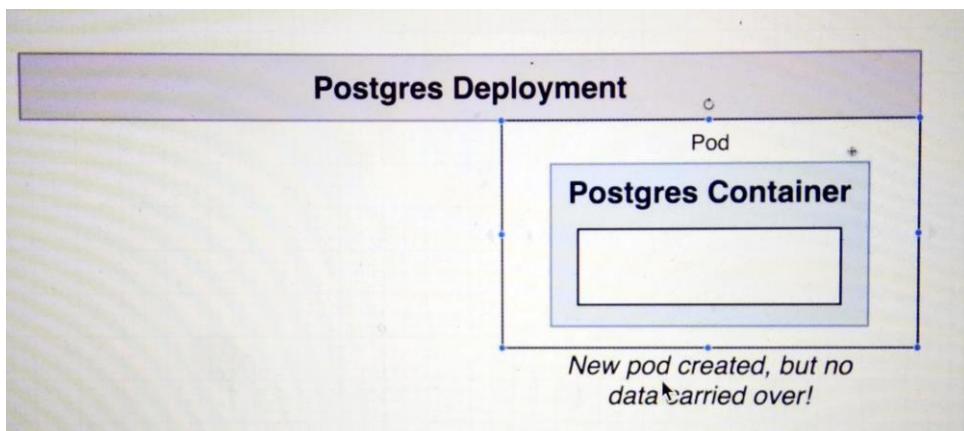
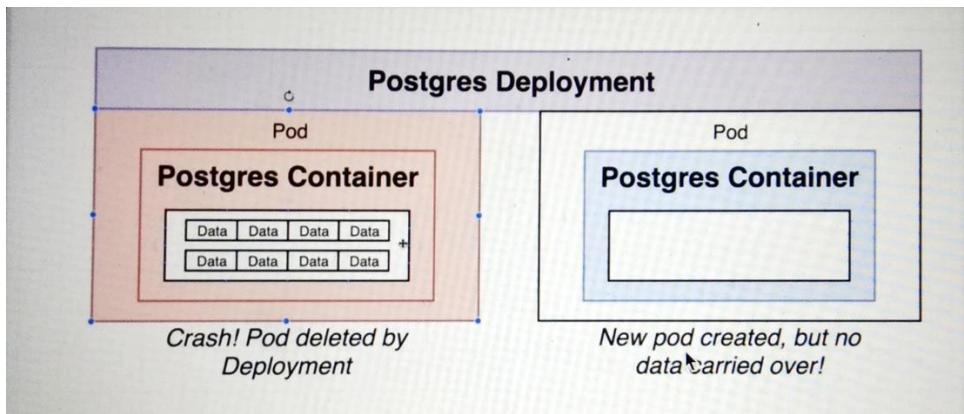
The Need for Volume with Database

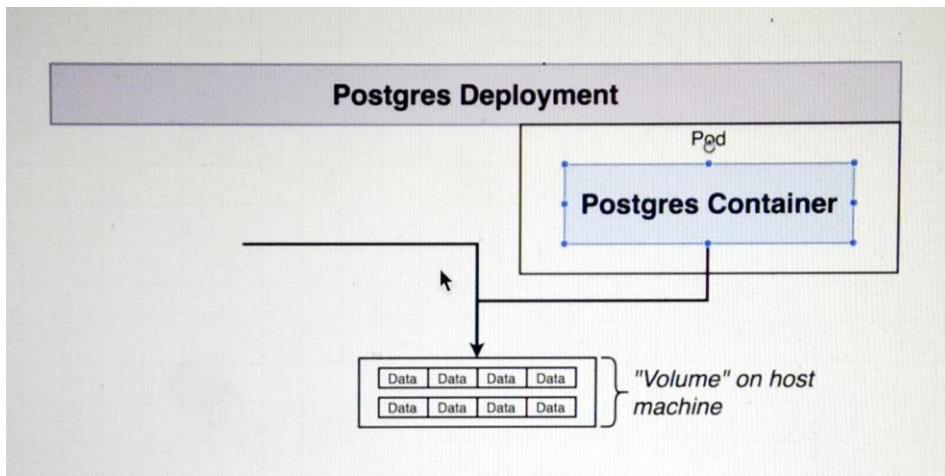


Volume is a consistent file system. Even if the pod crashes we have the data.

Running a multiple copies of postgres replicas, will access the same filesystem. It is the recipe for disaster. When we have replicas for database, we need to have a multiple configurations.







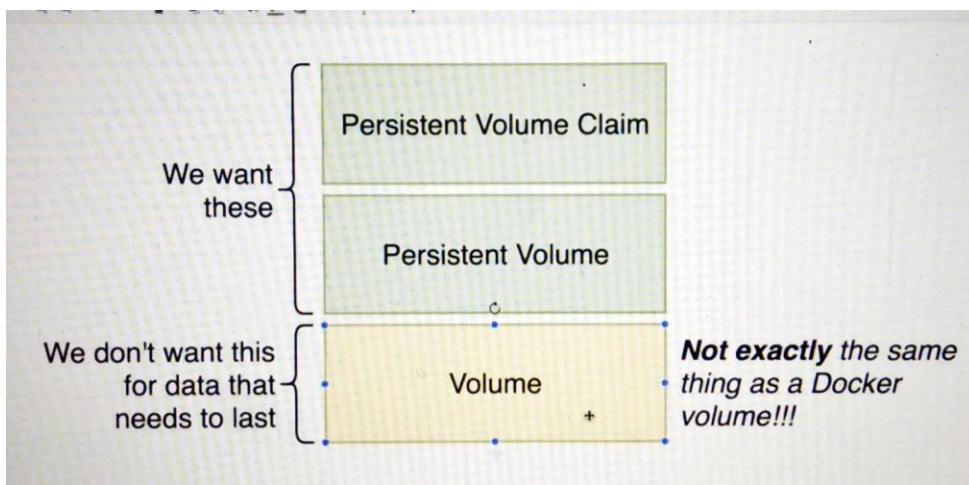
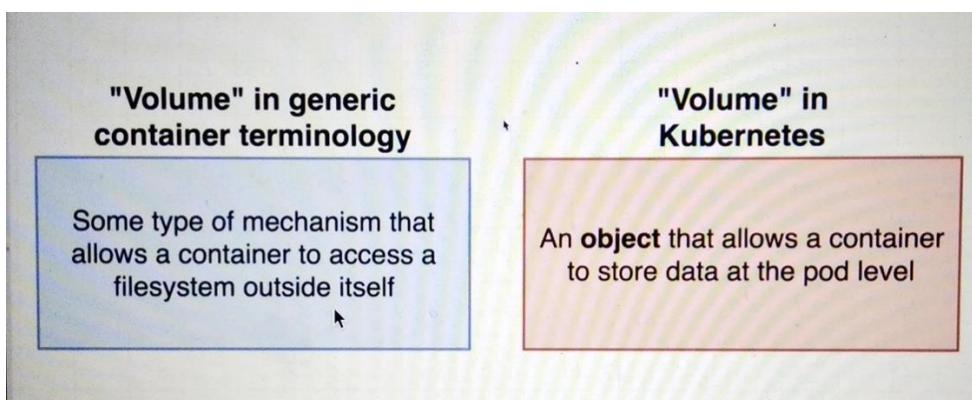
Kubernetes Volume

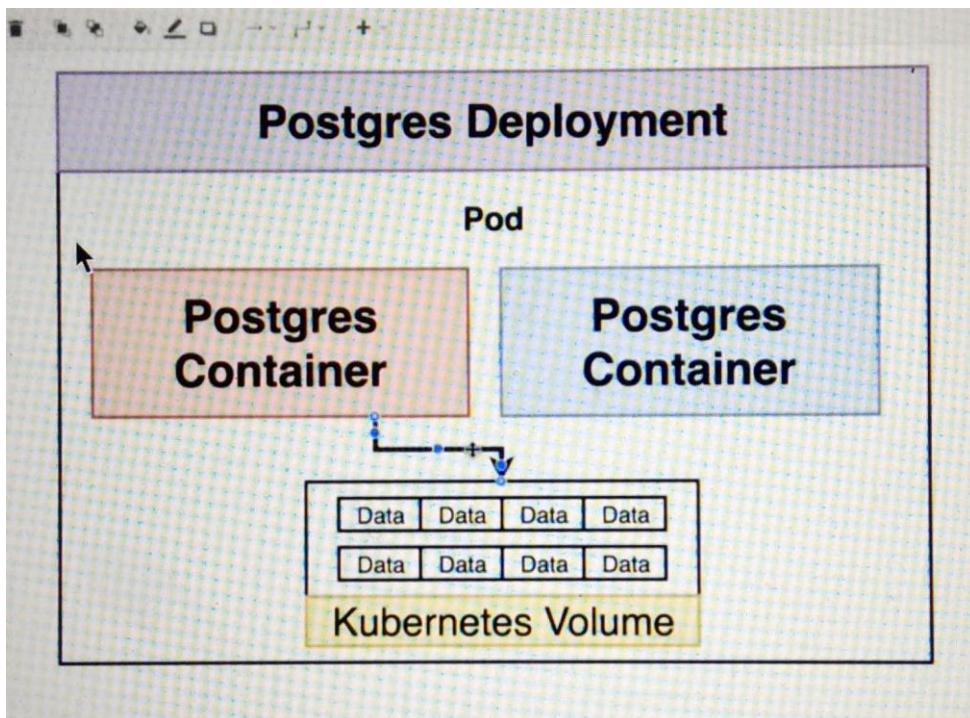
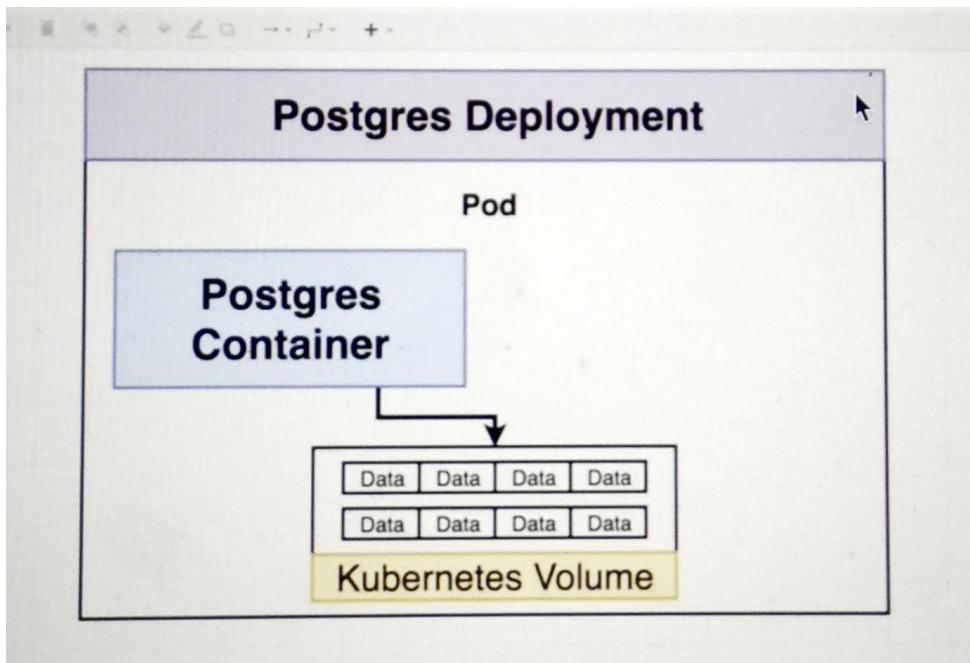
Storage tied directly to the specific pod.

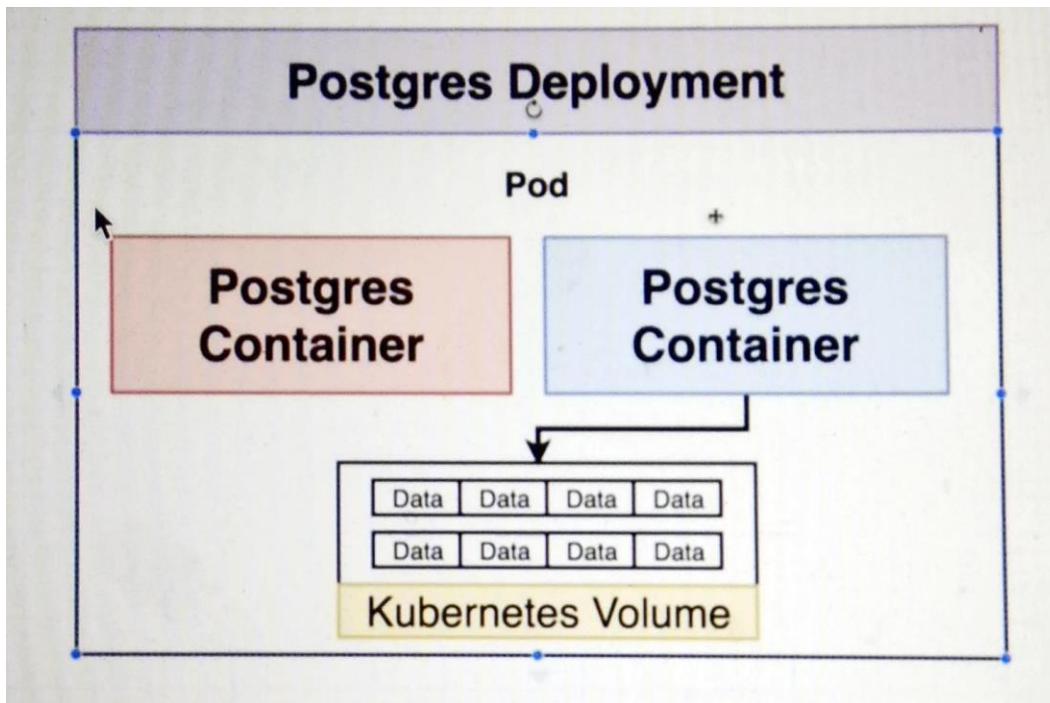
If any container crashes inside the pods, new container makes use of the existing data.

If the pod itself crashes, kubernetes volume will be lost.

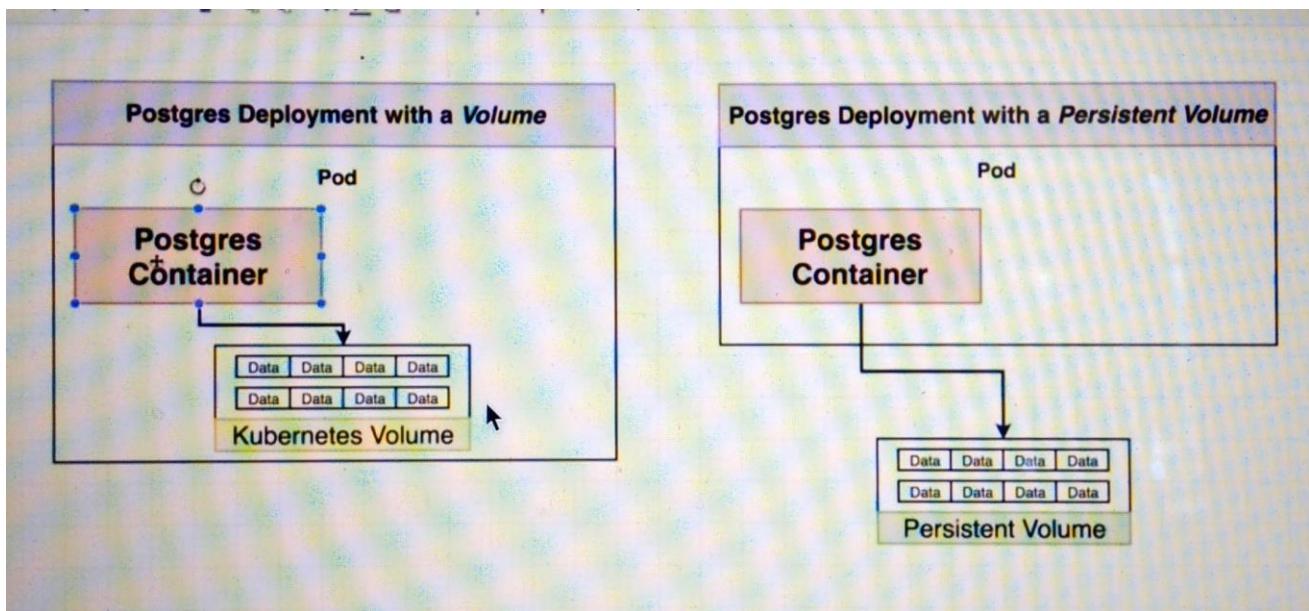
That is the reason we are going with Persistent volume claim.







Volume vs Persistent Volume

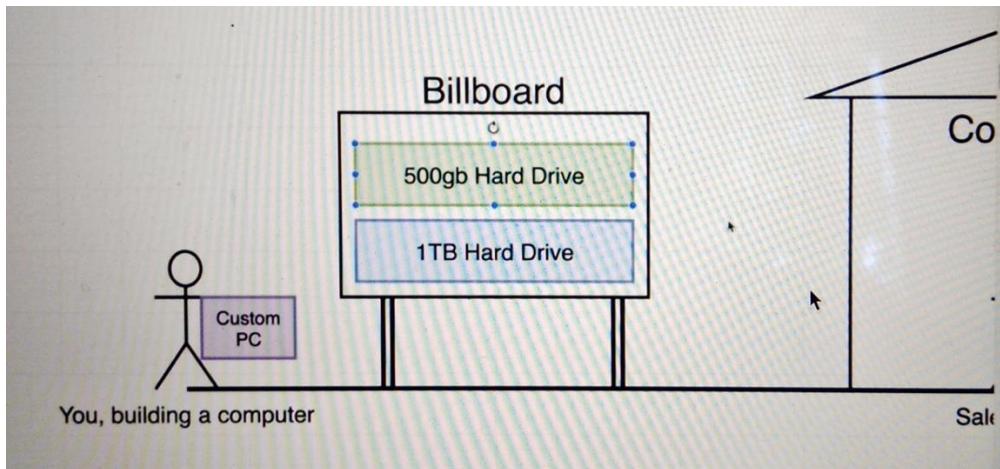


In case of persistent volume, even if the pod deleted or recreated, persistent volume stick around.

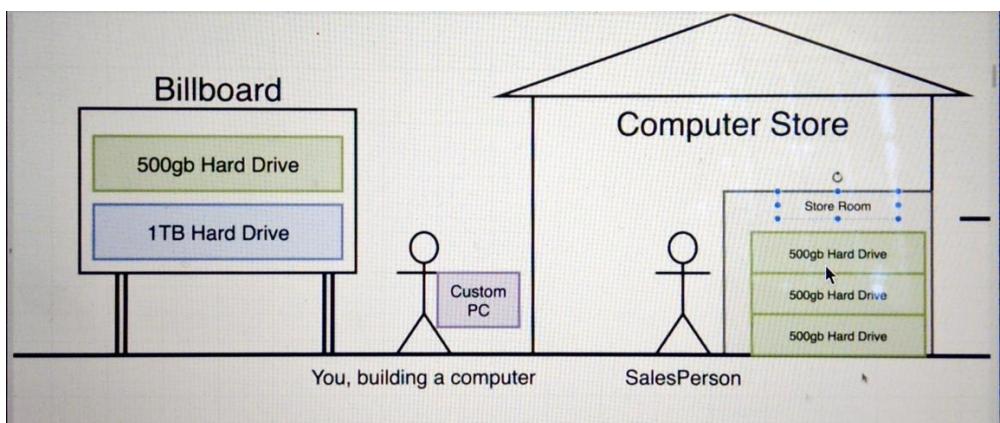
Persistent volume will last if the pod crashes.

Persistent Volume vs Persistent Volume Claim

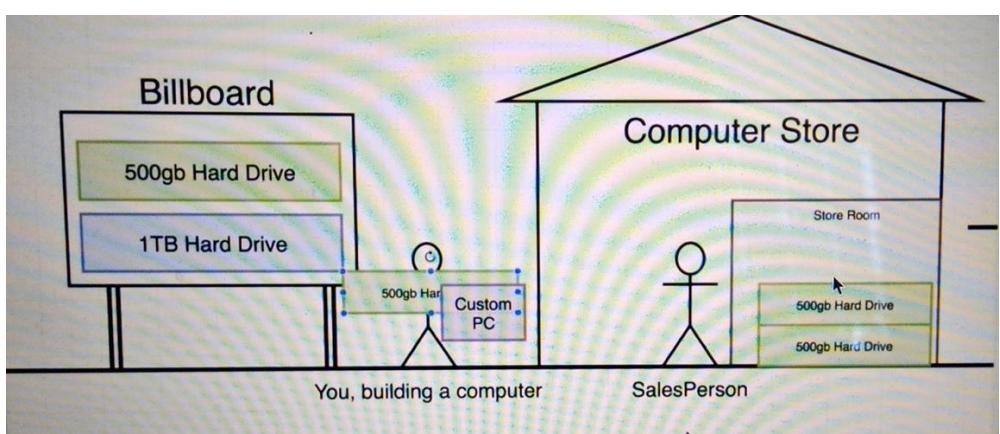
Customer sales person analogy for understanding persistent volume claim.



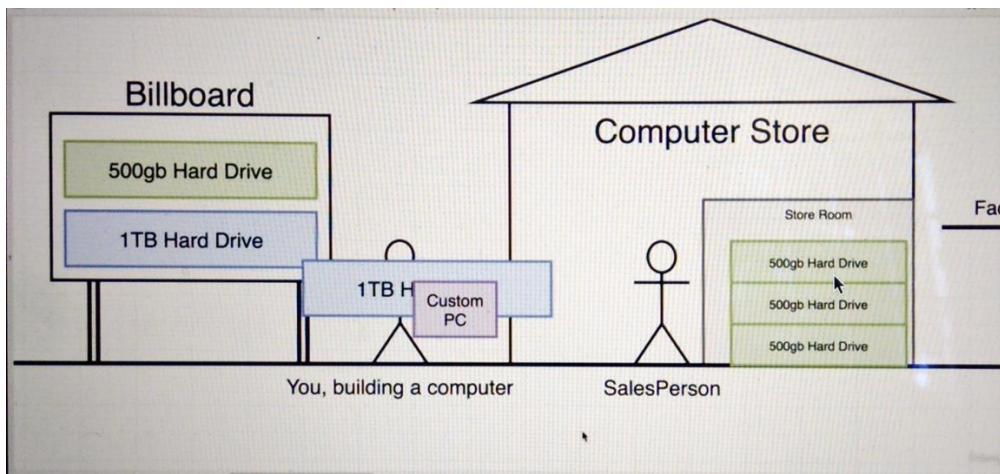
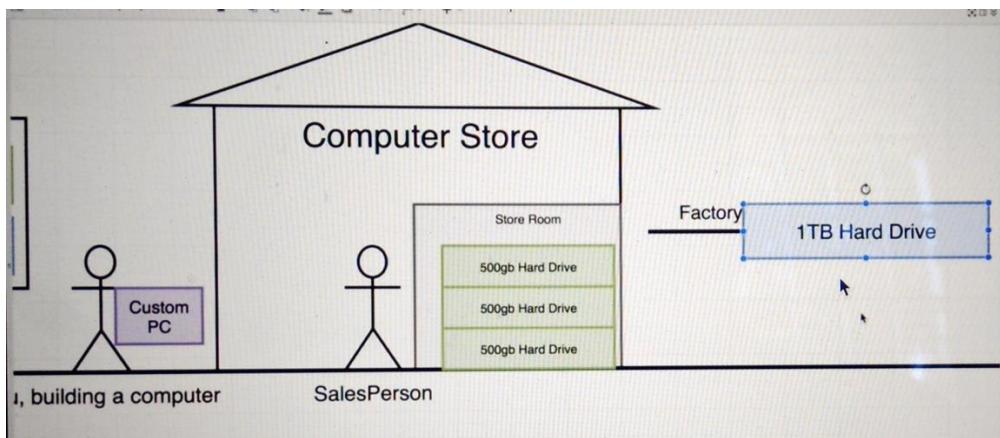
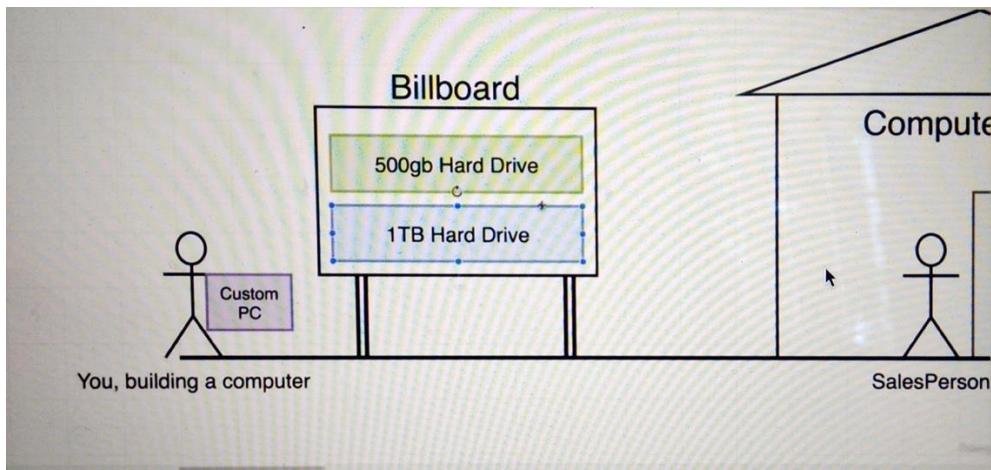
Asking for a 500gb hard drive.



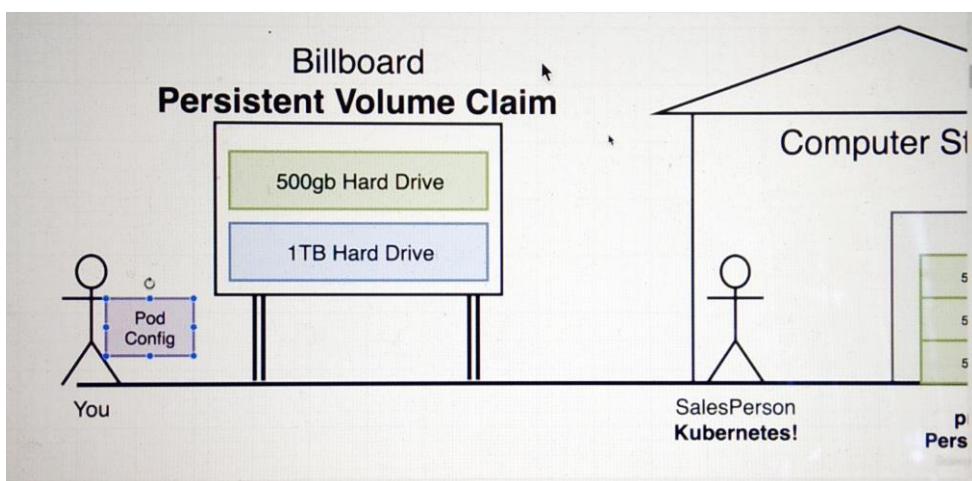
Available in the store. So purchased.

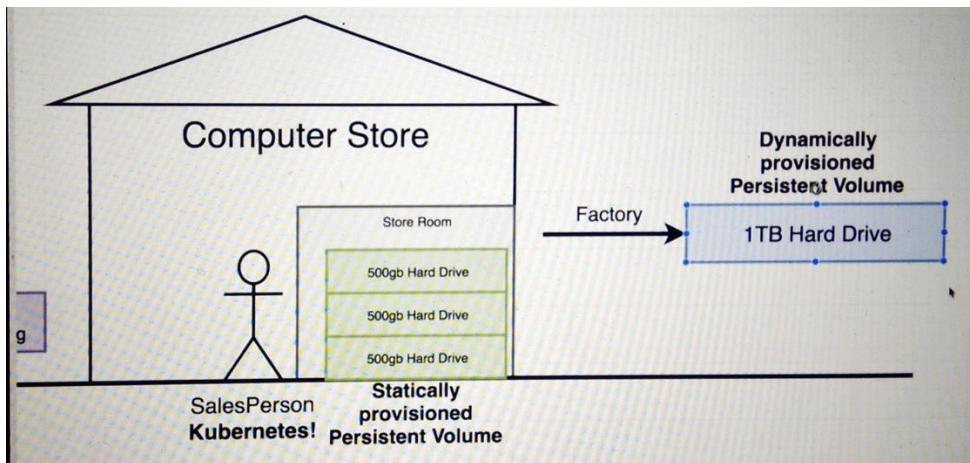
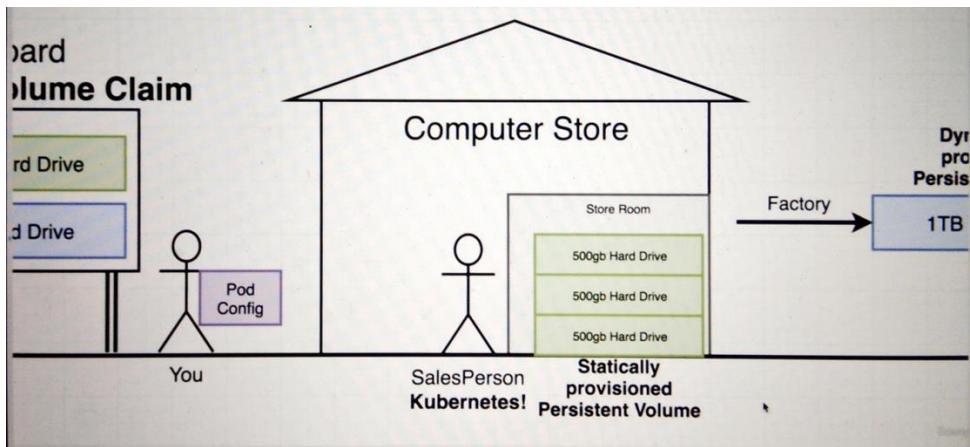


Coming to the store again. Now, we are asking for 1TB hard disk. But 1TB hard disk is not available in stock. So sales person immediate going to factory and manufacture it on spot. And Salesperson bring it back and handover to customer.



Kubernetes version of the same story with different terms



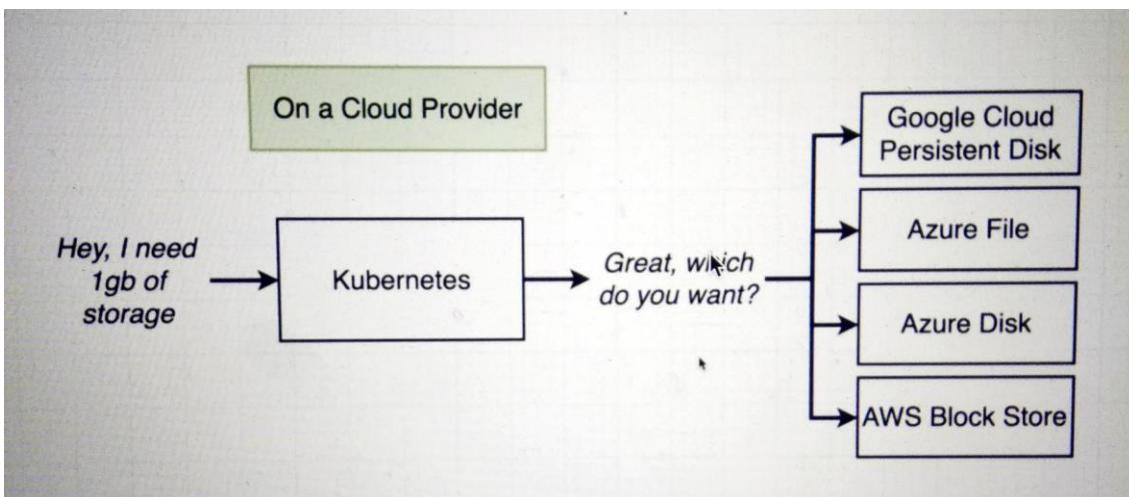
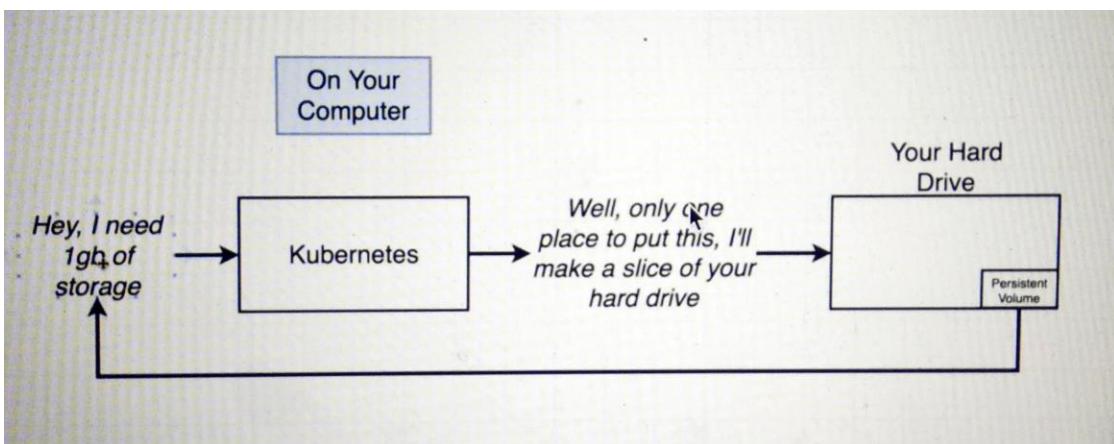
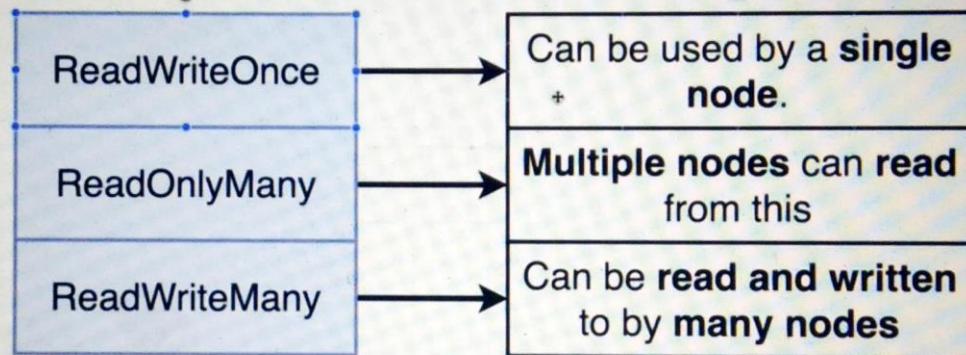


Persistent volume claim is an advertisement.

Claim Config Files

```
k8s >  database-persistent-volume-claim.yaml
 1  apiVersion: v1
 2  kind: PersistentVolumeClaim
 3  metadata:
 4    name: database-persistent-volume-claim
 5  spec:
 6    accessModes:
 7      - ReadWriteOnce
 8    resources:
 9      requests:
10        storage: 2Gi
11
```

Access Modes



Designating PVC in Pod Template

```

k8s > ⌂ postgres-deployment.yaml
 1  apiVersion: apps/v1
 2  kind: Deployment
 3  metadata:
 4    name: postgres-deployment
 5  spec:
 6    replicas: 1
 7    selector:
 8      matchLabels:
 9        component: postgres
10    template:
11      metadata:
12        labels:
13          component: postgres
14    spec:
15      volumes:
16        - name: postgres-storage
17          persistentVolumeClaim:
18            claimName: database-persistent-volume-claim
19      containers:
20        - name: postgres
21          image: postgres
22          ports:
23            - containerPort: 5432
24          volumeMounts:
25            - name: postgres-storage
26              mountPath: /var/lib/postgresql/data
27              subPath: postgres
28

```

Applying a PVC

```

~/Users/naveen.Kumar17.zshrc:68: no matches found: load?
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ 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
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘

```

Get persistent volume in kubernetes cluster

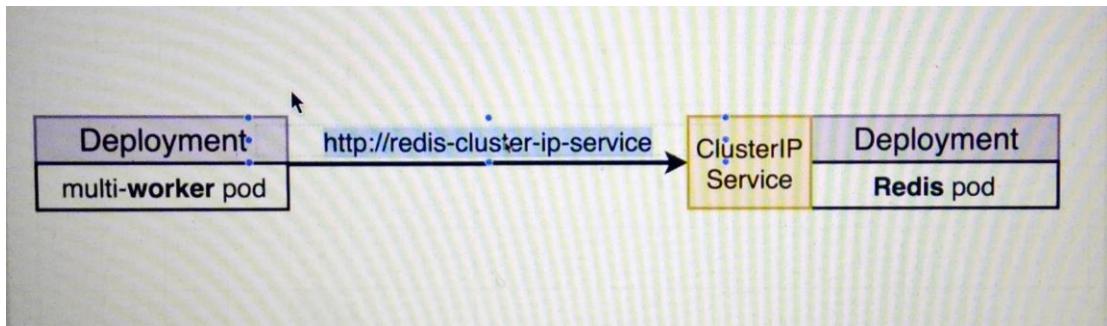
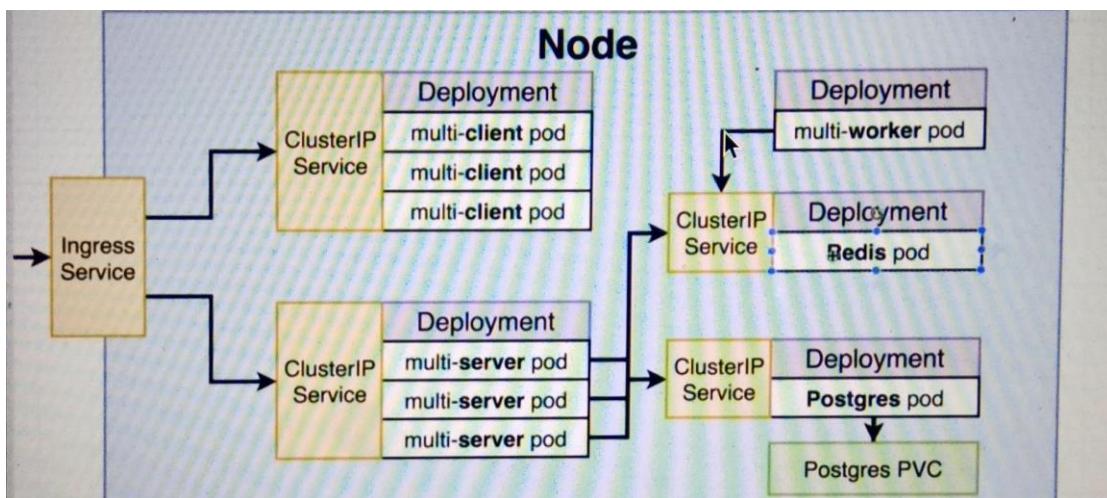
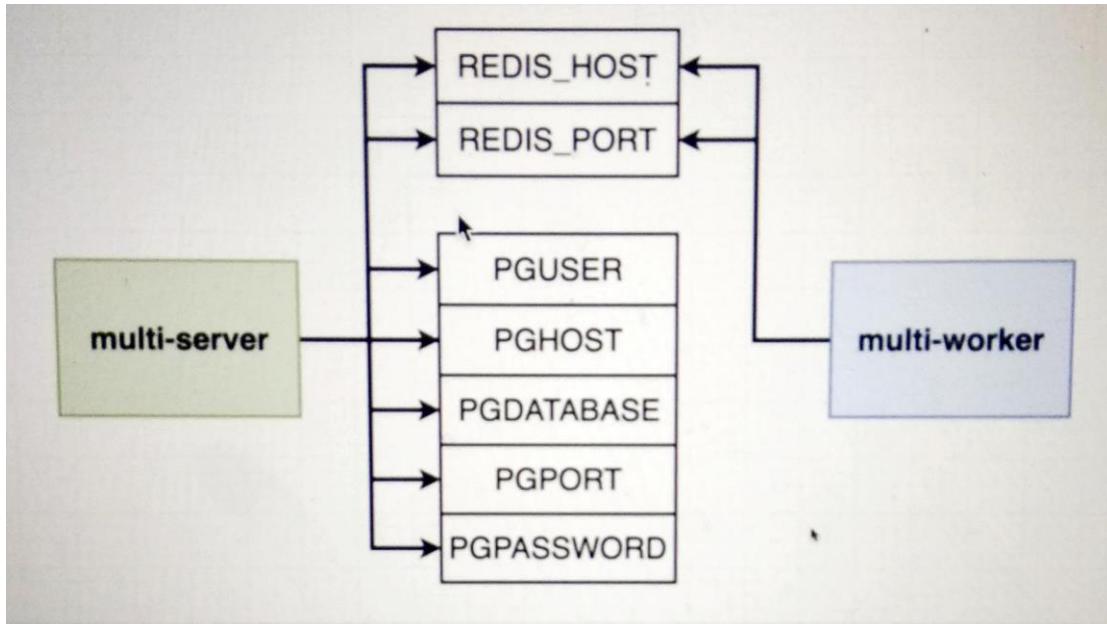
```

deployment.apps/worker deployment unchanged
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get pv
NAME                                     CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS   CLAI
M                                         STORAGECLASS  REASON AGE
pvc-d2b9d88c-743c-455b-9277-daed76e6caf2  2Gi        RWO   Delete           Bound   defa
ult/database-persistent-volume-claim   hostpath
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘

```

Defining Environment Variable

Environment Variables



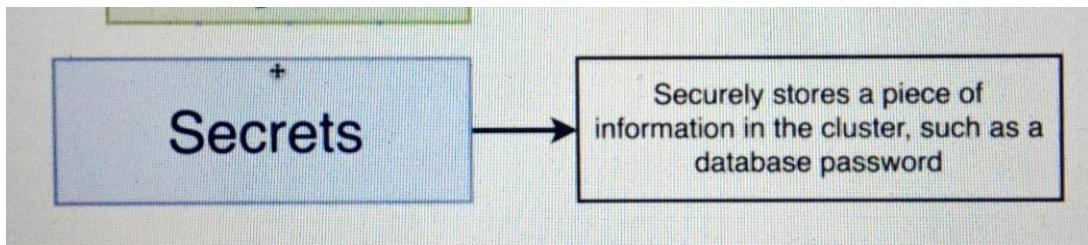
Updating the config file

```
k8s > ❁ worker-deployment.yaml
 6   replicas: 1
 7   selector:
 8     matchLabels:
 9       component: worker
10   template:
11     metadata:
12       labels:
13         component: worker
14   spec:
15     containers:
16       - name: worker
17         image: stephengrider/multi-worker
18         env:
19           - name: REDIS_HOST
20             value: redis-cluster-ip-service
21           - name: REDIS_PORT
22             value: 6379
```

```
k8s > ❁ server-deployment.yaml
 1   apiVersion: apps/v1
 2   kind: Deployment
 3   metadata:
 4     name: server-deployment
 5   spec:
 6     replicas: 3
 7     selector:
 8       matchLabels:
 9         component: server
10     template:
11       metadata:
12         labels:
13           component: server
14     spec:
15       containers:
16         - name: server
17           image: stephengrider/multi-server
18           ports:
19             - containerPort: 5000
20           env:
21             - name: REDIS_HOST
22               value: redis_cluster_ip_service
23             - name: REDIS_PORT
24               value: 6379
25             - name: PGUSER
26               value: postgres
27             - name: PGHOST
28               value: postgres-cluster-ip-service
29             - name: PGDATABASE
30               value: postgres
31
```

Creating an encoded secret

Kubernetes secret object



Kubernetes command for creating a secret

Creating a Secret						
		Type of object we are going to create		Name of secret, for later reference in a pod config	Key-value pair of the secret information	
kubectl	create	secret	generic	<secret_name>	--from-literal	key=value
Imperative command to create a new object			Type of secret	We are going to add the secret information into this command, as opposed to from .file		

Type of secret

- Tls
- Docker-registry
- Generic

```
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl create secret generic pgpassword --from-literal PGPASSWORD=qwerty123
secret/pgpassword created
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get secrets
NAME          TYPE           DATA   AGE
default-token-d549x  kubernetes.io/service-account-token  3      30d
pgpassword     Opaque          1      18s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

Updating the config file

```
k8s > server-deployment.yaml
14     spec:
15       containers:
16         - name: server
17           image: stephengrider/multi-server
18           ports:
19             - containerPort: 5000
20           env:
21             - name: REDIS_HOST
22               value: redis_cluster_ip_service
23             - name: REDIS_PORT
24               value: 6379
25             - name: PGUSER
26               value: postgres
27             - name: PGHOST
28               value: postgres-cluster-ip-service
29             - name: PGDATABASE
30               value: postgres
31             - name: POSTGRES_PASSWORD
32               valueFrom:
33                 secretKeyRef:
34                   name: pgpassword
35                   key: PGPASSWORD
36
```

PROBLEMS OUTPUT TERMINAL

Session contents restored from 09/05/2022 at 23:16:08

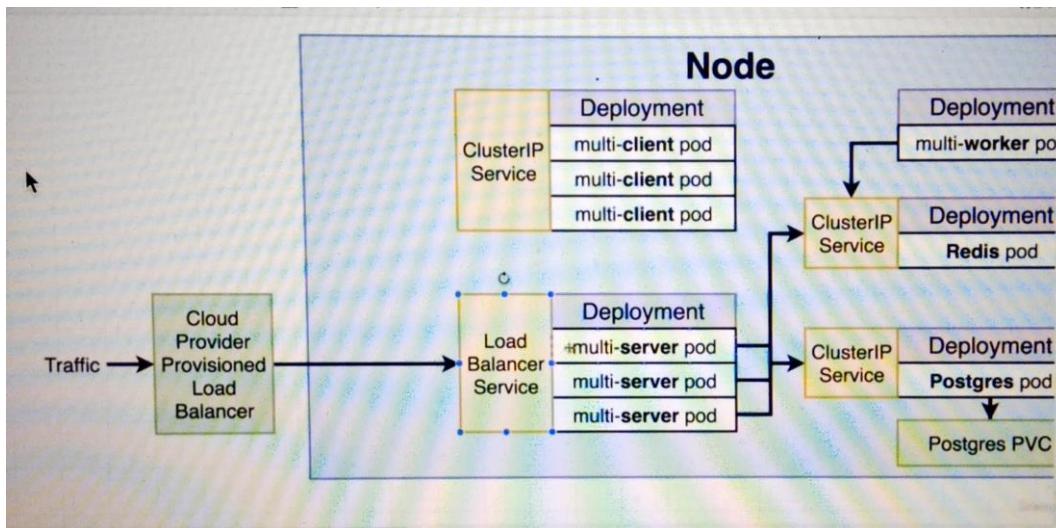
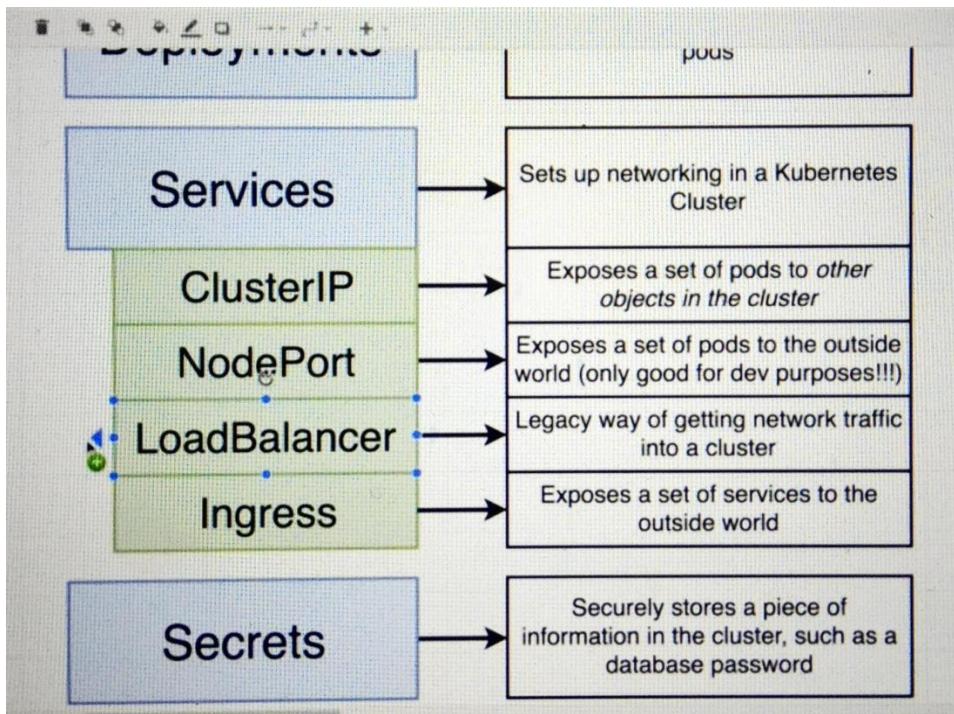
```
/Users/naveen.kumar1/.zshrc:68: no matches found: load?
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl create secret generic pgpassword --from-lit
er
al PGPASSWORD=qwerty123
secret/pgpassword created
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get secrets
```

Applying config file.

```
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
persistentvolumeclaim/database-persistent-volume-claim unchanged
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 configured
deployment.apps/worker-deployment configured
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```

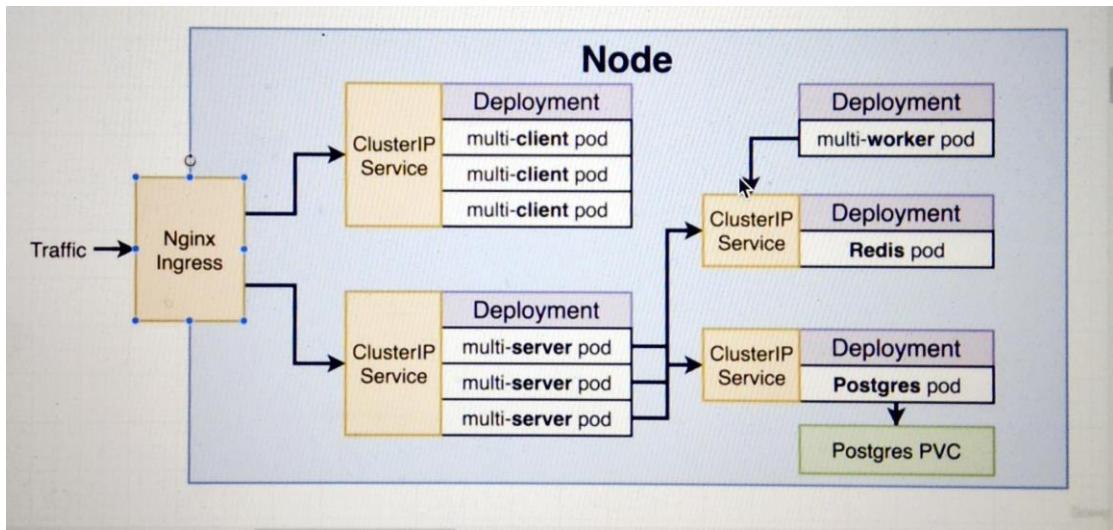
Handling the traffic with Ingress Controller

Load Balancer Services



Load balancer is specific to pods.

Quick notes in Ingress



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**

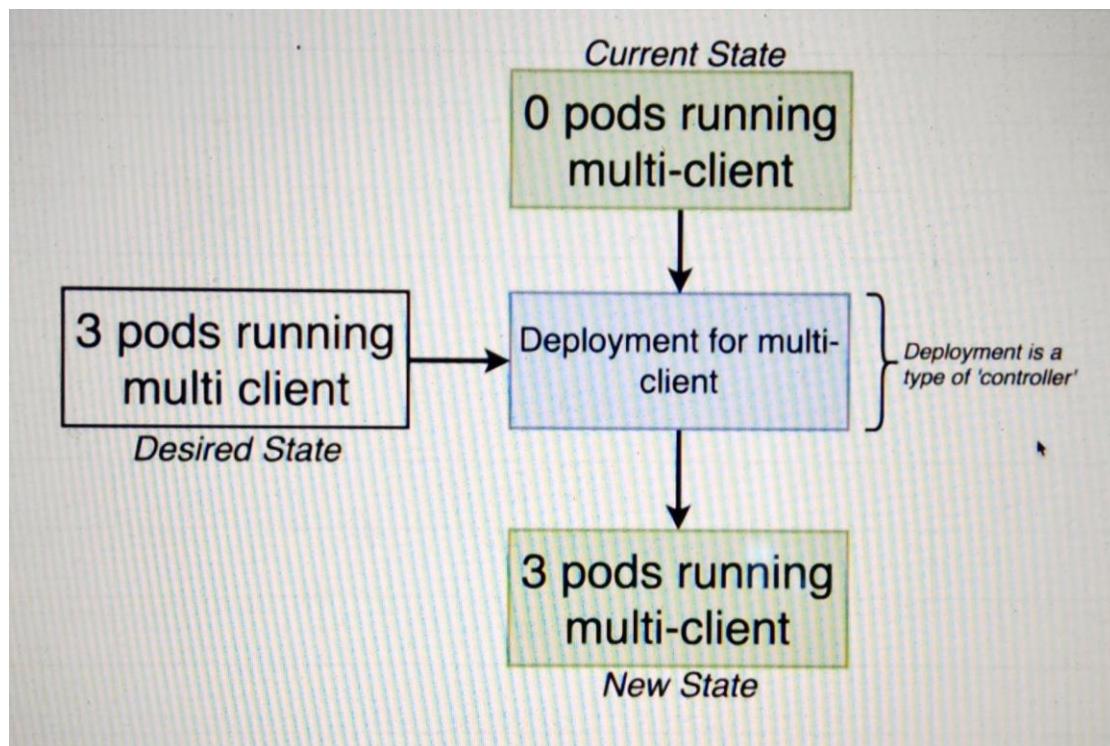
github.com/nginxinc/kubernetes-ingress

We are going to use ingress-nginx.

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

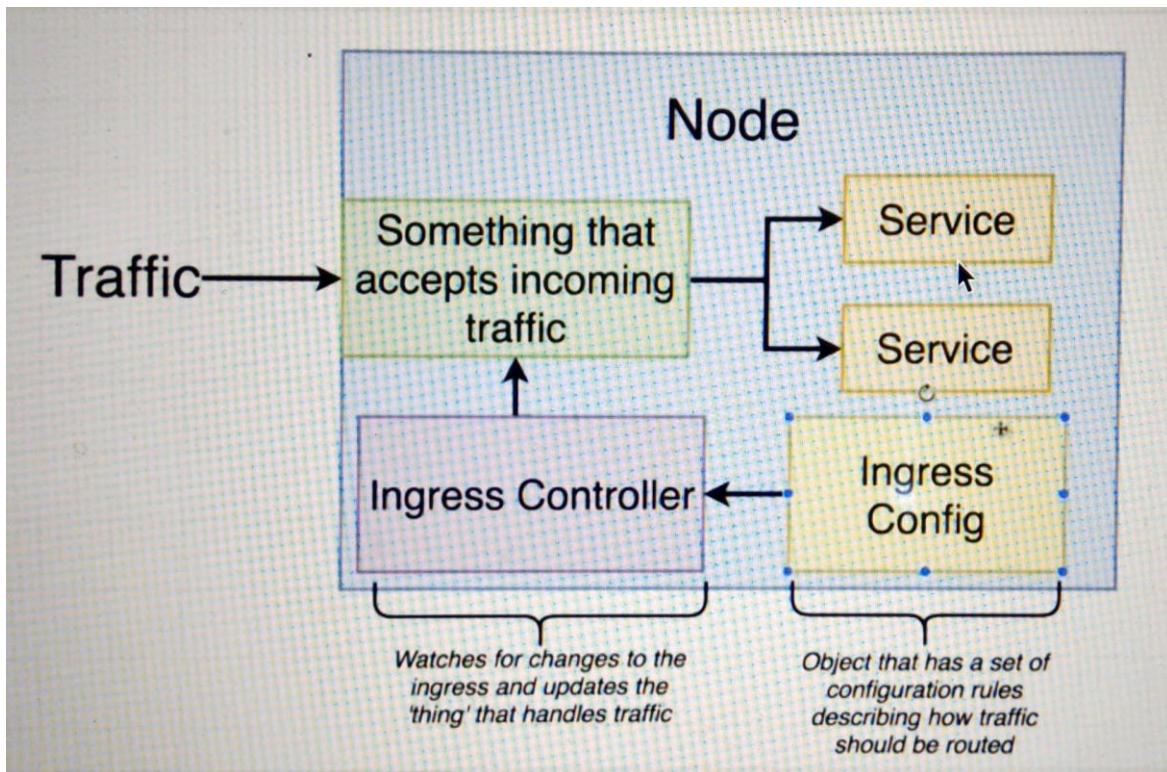
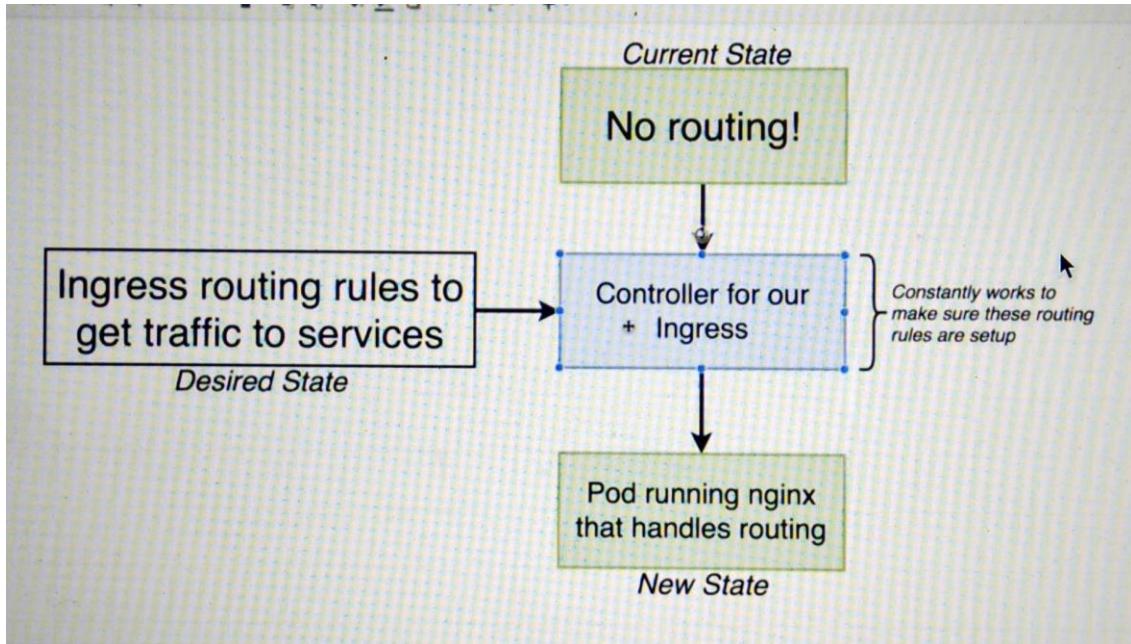
We are going to set up ingress-nginx on local and GC

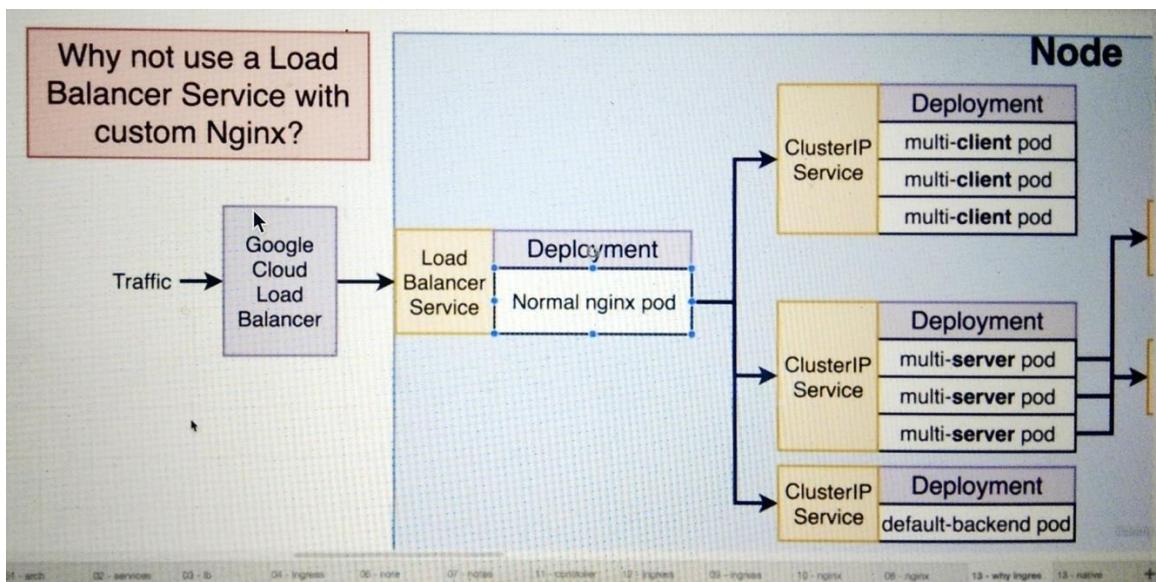
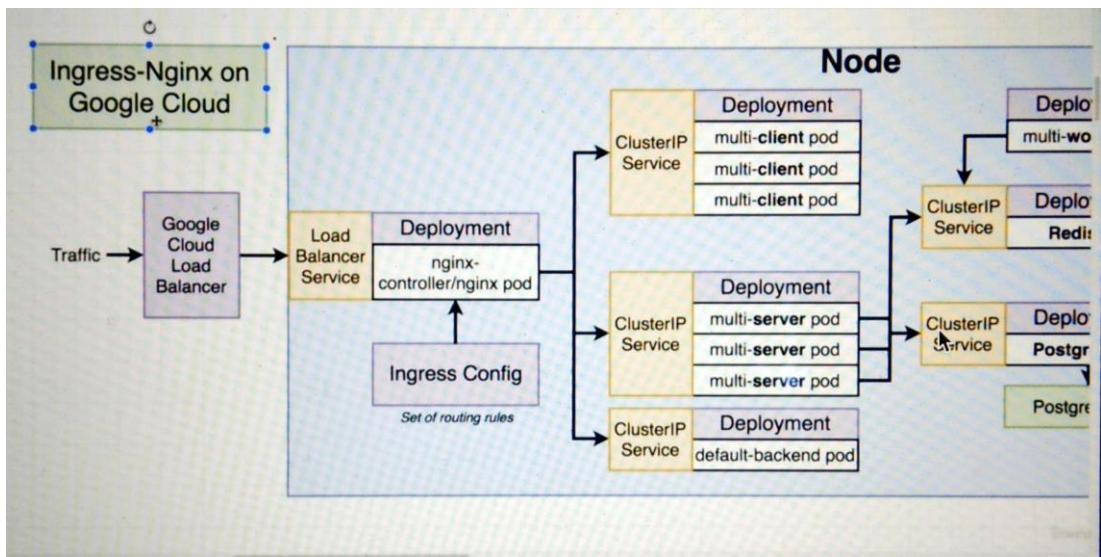
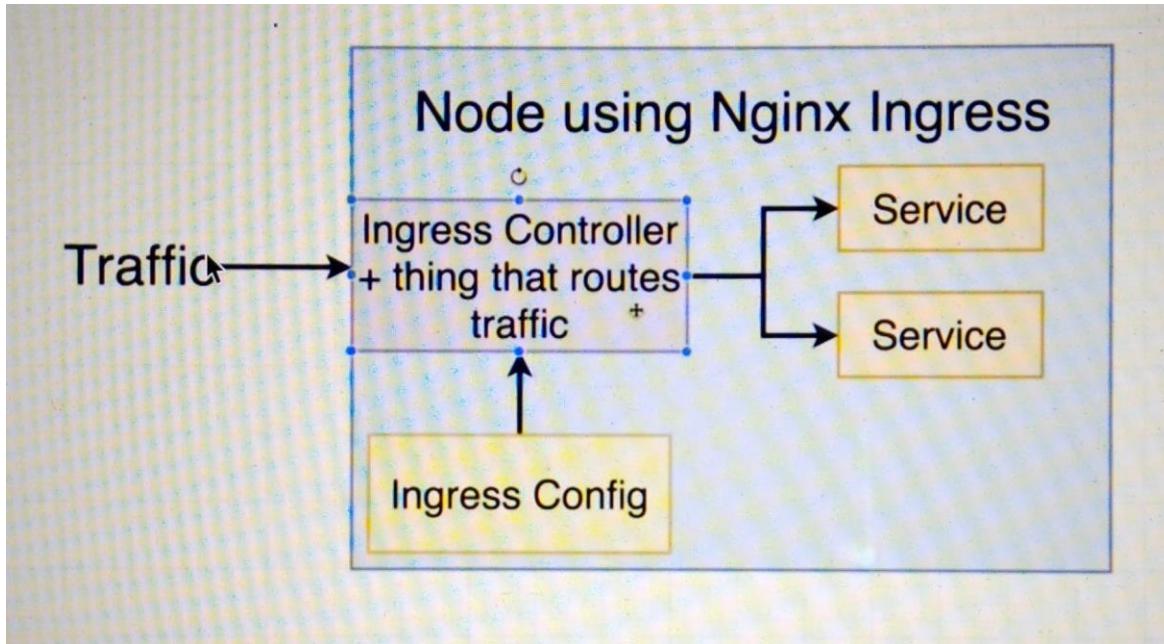
Behind the Scenes of Ingress



In kubernetes, controller means any type of object constantly runs in background to make the desired state.

Same thing applies to ingress also.





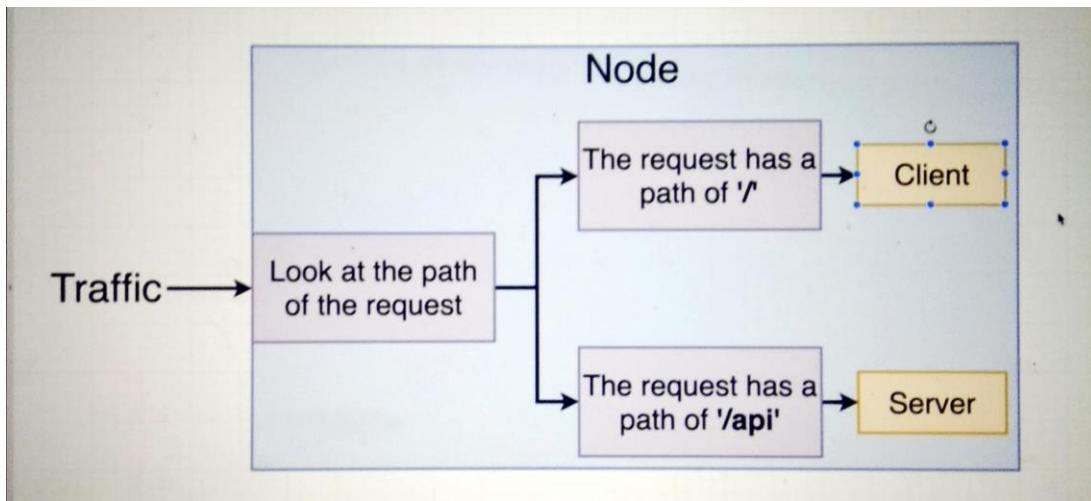
Nginx helps to route the request directly to pod, ability to bypass the multi-client pod.

<https://www.joyfulbikeshedding.com/blog/2018-03-26-studying-the-kubernetes-ingress-system.html>

Setting up Ingress Locally with Minikube

```
/Users/5/Desktop/Kubernetes/1.23.1/minikube: no matches found: load
> DOCKER_KUBE git:(main) ✘ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.2.0/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
DOCKER_KUBE git:(main)
```

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.2.0/deploy/static/provider/cloud/deploy.yaml>



Ingress Config File

```
k8s >  ingress-service.yaml
1  apiVersion: networking.k8s.io/v1
2  # UPDATE API
3  kind: Ingress
4  metadata:
5    name: ingress-service
6    annotations:
7      kubernetes.io/ingress.class: "nginx"
8      nginx.ingress.kubernetes.io/use-regex: "true"
9      # ADD ANNOTATION
10     nginx.ingress.kubernetes.io/rewrite-target: /$1
11     # UPDATE ANNOTATION
12 spec:
13   rules:
14     - http:
15       paths:
16         - path: /?(.*)
17           # UPDATE PATH
18           pathType: Prefix
19           # ADD PATHTYPE
20           backend:
21             service:
22               # UPDATE SERVICE FIELDS
23               name: client-cluster-ip-service
24               port:
25                 number: 3000
26         - path: /api/?(.*)
27           # UPDATE PATH
28           pathType: Prefix
29           # ADD PATHTYPE
30           backend:
31             service:
32               # UPDATE SERVICE FIELDS
33               name: server-cluster-ip-service
34               port:
35                 number: 5000
36   |
```

Applying the config file.

```
✗ command not found: Kubectl
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl apply -f k8s
service/client-cluster-ip-service unchanged
deployment.apps/client-deployment unchanged
persistentvolumeclaim/database-persistent-volume-claim unchanged
ingress.networking.k8s.io/ingress-service created
service/postgres-cluster-ip-service unchanged
deployment.apps/postgres-deployment unchanged
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
```



Fib Calculator version KUBERNETES!

[Home](#)[Other Page](#)

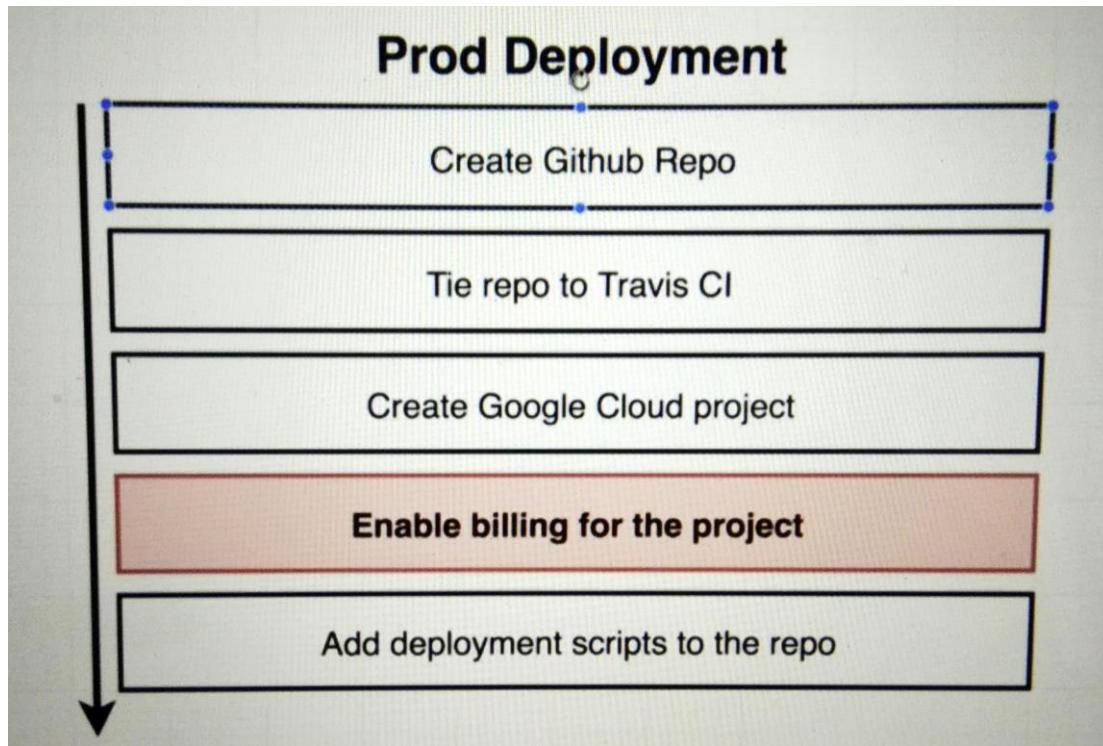
Enter your index:

Indexes I have seen:

Calculated Values:

Kubernetes Production Deployment

Deployment Steps



We have an option to avail free Google Cloud credits.

Free Google Cloud Credits

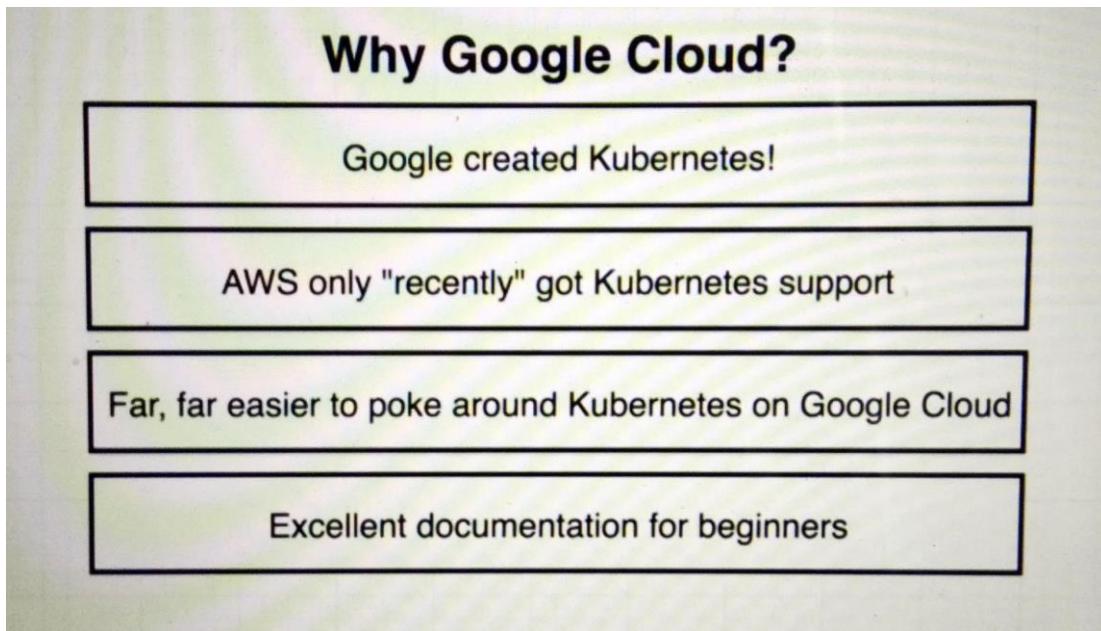
In the next section, we are going to start creating our project on Google Cloud.

Remember, creating Kubernetes clusters on Google Cloud costs real money! If you are sensitive to spending money, you can try getting some free Google Cloud credits using this link:

<https://cloud.google.com/free>

Important! If you already have a Google Cloud account you will have to sign out of it and create a new account in order to get those free credits!

Google Cloud



Creating a project in Google Cloud

The screenshot shows the Google Cloud Platform dashboard with the 'My First Project' dropdown menu open. A modal window titled 'Select a project' is displayed, containing a search bar and a list of recent projects. The 'NEW PROJECT' button is highlighted with a red box.

RECENT	STARRED	ALL
My First Project		
My Project		

Below the modal, the dashboard shows various service icons and a message about event-driven serverless functions.

New Project

You have 10 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name * [?](#)

Project ID: multi-k8s-350408. It cannot be changed later. [EDIT](#)

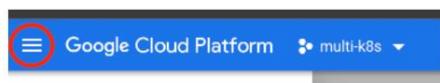
Location * [BROWSE](#)

Parent organization or folder

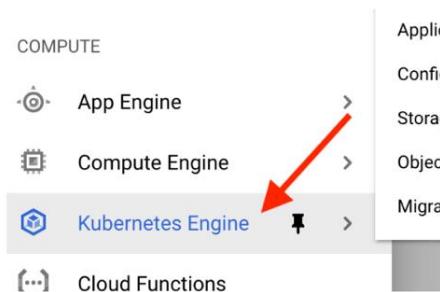
[CREATE](#) [CANCEL](#)

Creating Kubernetes Engine and starting the cluster in GCP

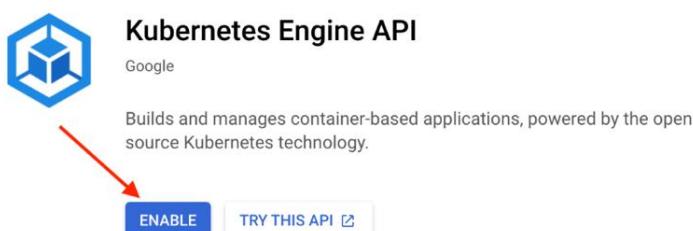
1. Click the Hamburger menu on the top left-hand side of the dashboard.



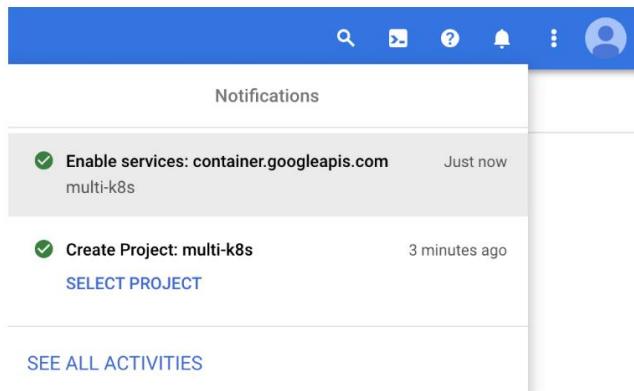
2. Click **Kubernetes Engine**



3. Click the **ENABLE** button to enable the Kubernetes API for this project.



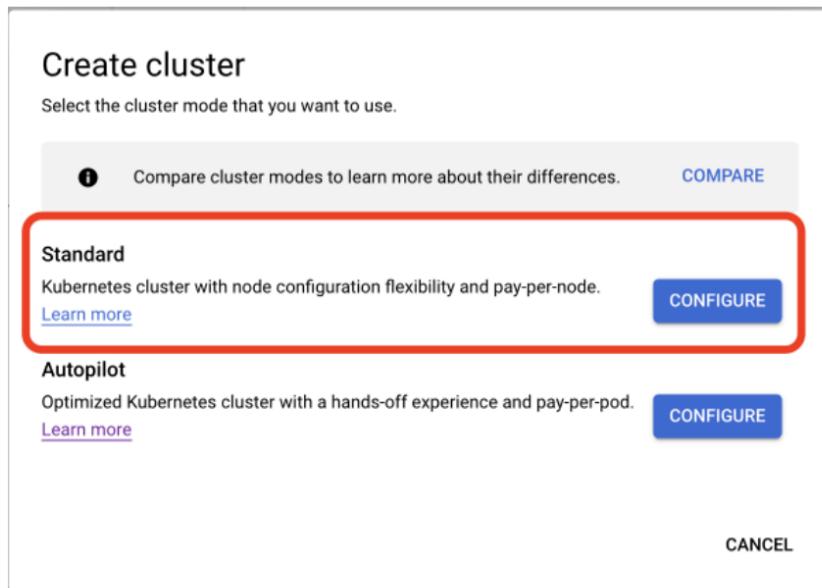
- After a few minutes of waiting, clicking the **bell** icon in the top right part of the menu should show a **green** checkmark for **Enable services: container.googleapis.com**



- If you refresh the page it should show a screen to create your first cluster. If not, click the hamburger menu and select **Kubernetes Engine** and then **Clusters**.
Once you see the screen below, click the **CREATE** button.



6. A **Create Cluster** dialog will open and provide two choices. Standard and Autopilot. Click the **CONFIGURE** button within the **Standard** cluster option



7. A form similar to the one shown in the video will be presented. Set the **Name** to **multi-cluster** (step 1). Confirm that the **Zone** set is actually near your location (step 2). The Node Pool that is discussed in the video is now found in a separate dropdown on the left sidebar. Click the downward-facing arrow to view the settings. No changes are needed here (step 3). Finally, click the **CREATE** button at the bottom of the form (step 4).

Create a Kubernetes cluster

Cluster basics

NODE POOLS

- default-pool **3**

CLUSTER

- Automation
- Networking
- Security
- Metadata
- Features

Name **1** multi-cluster

Location type
 Zonal
 Regional

Zone **2** us-central1-c

Specify default node locations **?**
 Current default: us-central1-c

Control plane version
 Choose a release channel for automatic management of your cluster's version and upgrade cadence. Choose a static version for more direct management of your cluster's version. [Learn more.](#)

- Static version
- Release channel

Release channel
 Regular channel (default)

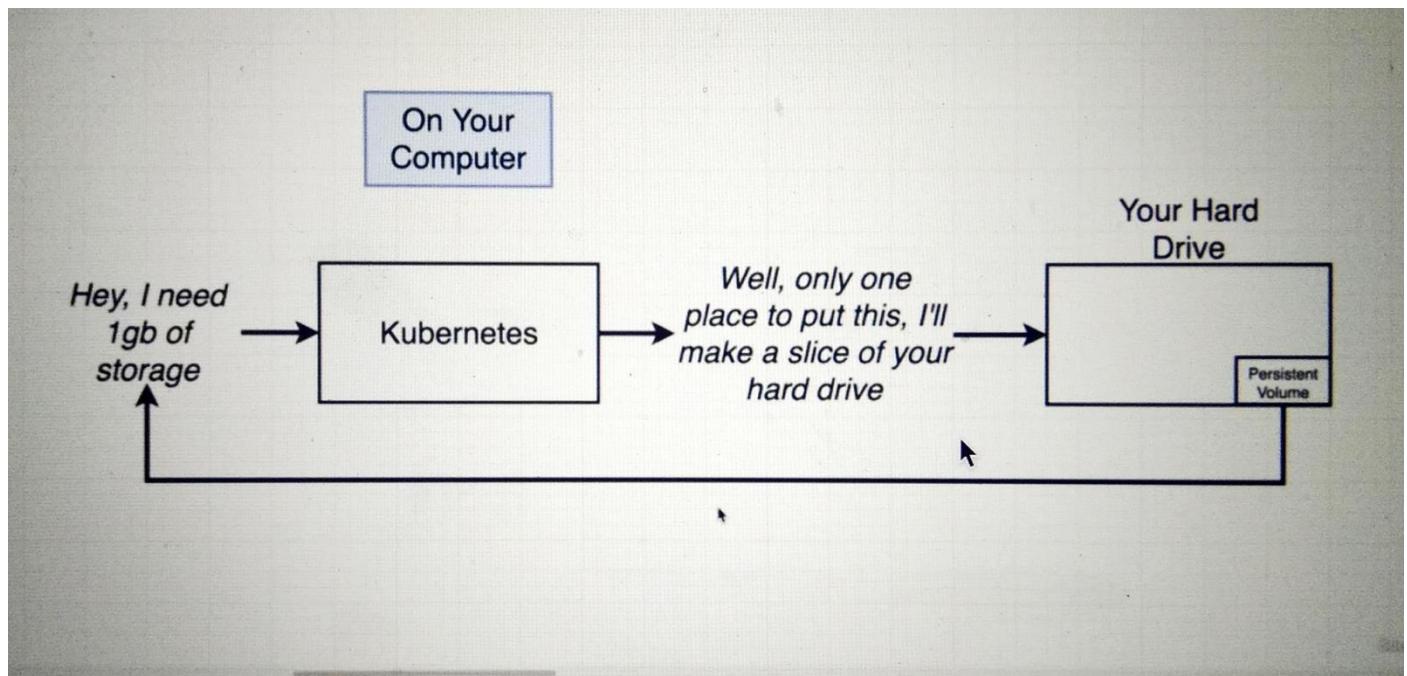
Version
 1.18.12-gke.1210 (default)

These versions have passed internal validation and are considered production-quality, but don't have enough historical data to guarantee their stability. Known issues

4 **CREATE** CANCEL Equivalent REST or COMMAND LINE

8. After a few minutes, the cluster dashboard should load and your multi-cluster should have a **green** checkmark in the table.

Kubernetes Dashboard on Google Cloud



Google Cloud Platform multi-k8s

Kubernetes Engine

Kubernetes clusters

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
multi-cluster	us-central1-a	3	3 vCPUs	11.25 GB		

Marketplace

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io diagrams.xml - draw.io diagrams.xml - draw.io StephenGrider/multi-k8s StephenGrider/multi-k8s Kubernetes Engine - multi-k8s Stephen

Secure https://console.cloud.google.com/kubernetes/clusters/details/us-central1-a/multi-cluster?project=skillful-berm-214822&folder&organizationId&tab=detail... Fri 10:12 AM

Google Cloud Platform multi-k8s Clusters EDIT DELETE DEPLOY CONNECT

Kubernetes Engine

Clusters Workloads Services Applications Configuration Storage Marketplace

	Master zone	us-central1-a
Node zones	us-central1-a	
Network	default	
Subnet	default	
VPC-native (alias IP)	Disabled	
Pod address range	10.8.0.0/14	
Stackdriver Logging	Enabled	
Stackdriver Monitoring	Enabled	
Private cluster	Disabled	
Master authorized networks	Disabled	
Network policy	Disabled	
Legacy authorization	Disabled	
Maintenance window	Any time	
Cloud TPU	Disabled	

Labels None

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io diagrams.xml - draw.io diagrams.xml - draw.io StephenGrider/multi-k8s StephenGrider/multi-k8s Kubernetes Engine - multi-k8s Stephen

Secure https://console.cloud.google.com/kubernetes/clusters/details/us-central1-a/multi-cluster?project=skillful-berm-214822&folder&organizationId&tab=detail... Fri 10:13 AM

Google Cloud Platform multi-k8s Clusters EDIT DELETE DEPLOY CONNECT

Kubernetes Engine

Clusters Workloads Services Applications Configuration Storage Marketplace

Node Pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit. [Learn more](#)

default-pool (3 nodes, version 1.9.7-gke.6)	
Name	default-pool
Size	3
Node version	1.9.7-gke.6
Node image	Container-Optimized OS (cos)
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Total cores	3 vCPUs
Total memory	11.25 GB
Automatic node upgrades	Disabled
Automatic node repair	Enabled

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x StephenGrider/multi-k8s x StephenGrider/multi-k8s x Kubernetes Engine - multi-k8s x Stephen

Secure https://console.cloud.google.com/kubernetes/workload?folder=8&organizationId=&project=skillful-berm-214822&workload_list_tableSize=50

Fri 10:13 AM

Google Cloud Platform multi-k8s

Kubernetes Engine Workloads  REFRESH

-  Clusters
-  **Workloads** 
-  Services
-  Applications
-  Configuration
-  Storage

Kubernetes Engine Deploy a containerized application

Deploy, manage, and scale containers on Kubernetes, powered by Google Cloud. Learn more

Deploy or **Show system workloads**

Marketplace

https://console.cloud.google.com/kubernetes/workload?project=skillful-berm-214822

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x StephenGrider/multi-k8s x StephenGrider/multi-k8s x Kubernetes Engine - multi-k8s x Stephen

Secure https://console.cloud.google.com/kubernetes/discovery?folder=8&organizationId=&project=skillful-berm-214822&service_list_tableSize=50

Fri 10:13 AM

Google Cloud Platform multi-k8s

Kubernetes Engine Services  REFRESH

-  Clusters
-  Workloads
-  **Services** 
-  Applications
-  Configuration
-  Storage

Kubernetes services Brokered services 

Kubernetes Engine Discovery & load balancing

Kubernetes discovered services, like load balancers, define a logical set of pods and a way to access them (a micro-service).

Learn more or **Show system objects**

Marketplace

https://console.cloud.google.com/kubernetes/discovery?project=skillful-berm-214822

Chrome File Edit View History Bookmarks People Window Help Fri 10:13 AM

diagrams.xml - draw.io diagrams.xml - draw.io diagrams.xml - draw.io StephenGrider/multi-k8s StephenGrider/multi-k8s Kubernetes Engine - multi-k8s Stephen

Secure https://console.cloud.google.com/kubernetes/application?folder=&organizationId=&project=skillful-berm-214822&application_list_tablesize=60

1

Google Cloud Platform multi-k8s

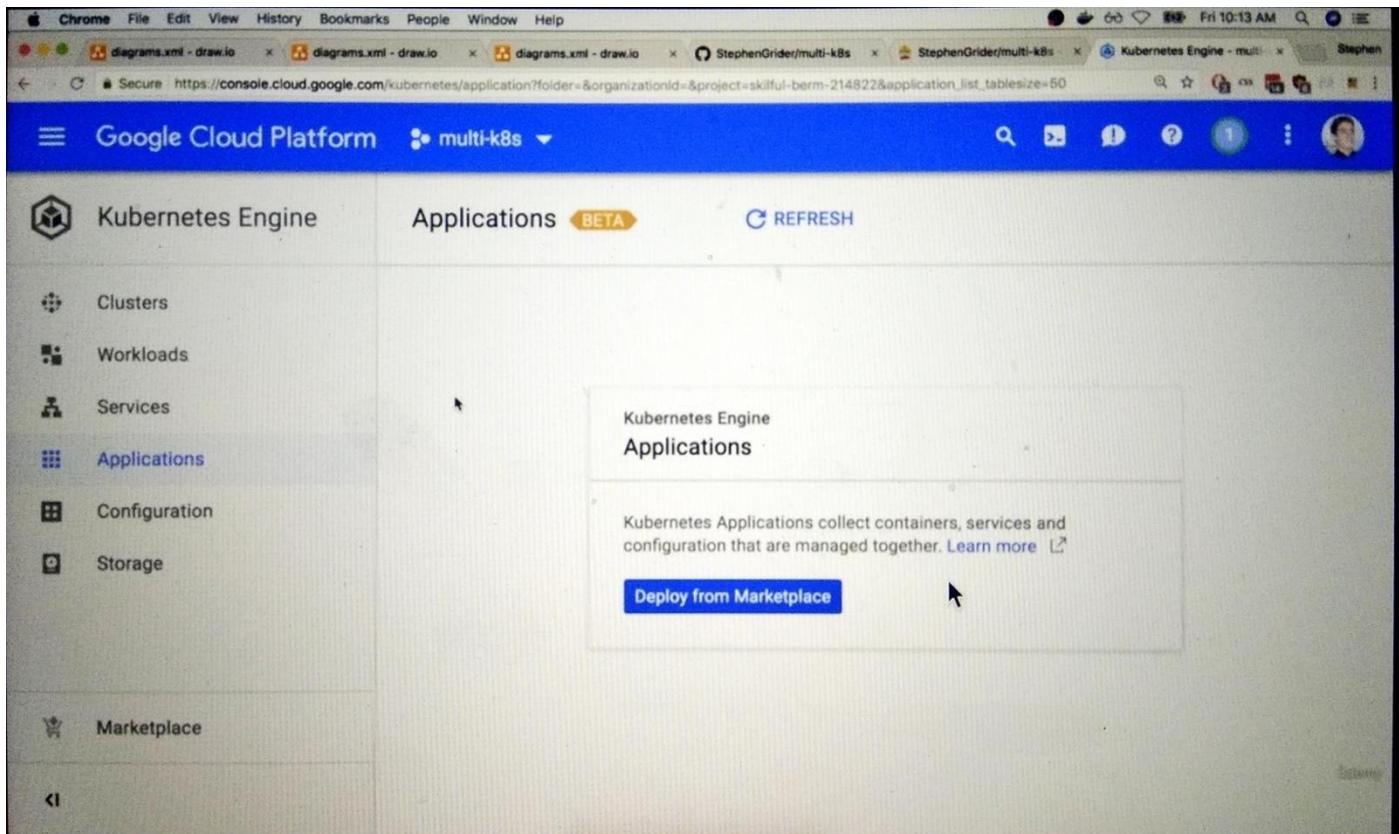
Kubernetes Engine Applications BETA REFRESH

Clusters Workloads Services Applications Configuration Storage Marketplace

Kubernetes Engine Applications

Kubernetes Applications collect containers, services and configuration that are managed together. Learn more ↗

Deploy from Marketplace



Chrome File Edit View History Bookmarks People Window Help Fri 10:14 AM

diagrams.xml - draw.io diagrams.xml - draw.io diagrams.xml - draw.io StephenGrider/multi-k8s StephenGrider/multi-k8s Kubernetes Engine - multi-k8s Stephen

Secure https://console.cloud.google.com/kubernetes/config?folder=&organizationId=&project=skillful-berm-214822&config_list_tablesize=50

1

Google Cloud Platform multi-k8s

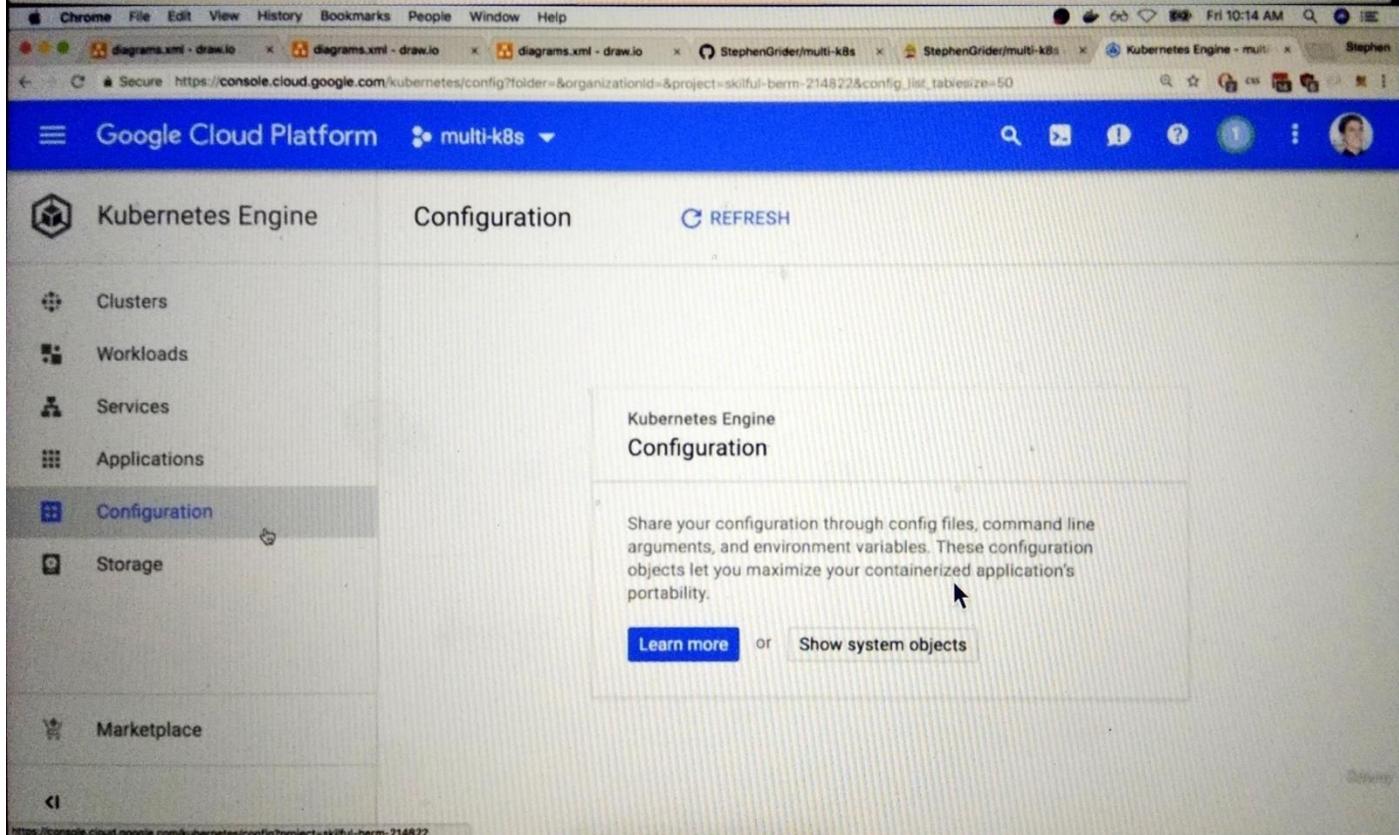
Kubernetes Engine Configuration REFRESH

Clusters Workloads Services Applications Configuration Storage Marketplace

Kubernetes Engine Configuration

Share your configuration through config files, command line arguments, and environment variables. These configuration objects let you maximize your containerized application's portability.

Learn more or Show system objects



Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x StephenGrider/multi-k8s x StephenGrider/multi-k8s x Kubernetes Engine - multi-k8s x

C Secure https://console.cloud.google.com/kubernetes/storage?folder=&organizationId=&project=skillful-berm-214822&tab=persistentVolumeClaims&persistent_v...

Fri 10:14 AM

Google Cloud Platform multi-k8s

Kubernetes Engine Storage REFRESH

- Clusters
- Workloads
- Services
- Applications
- Configuration
- Storage**

Persistent volume claims Storage classes

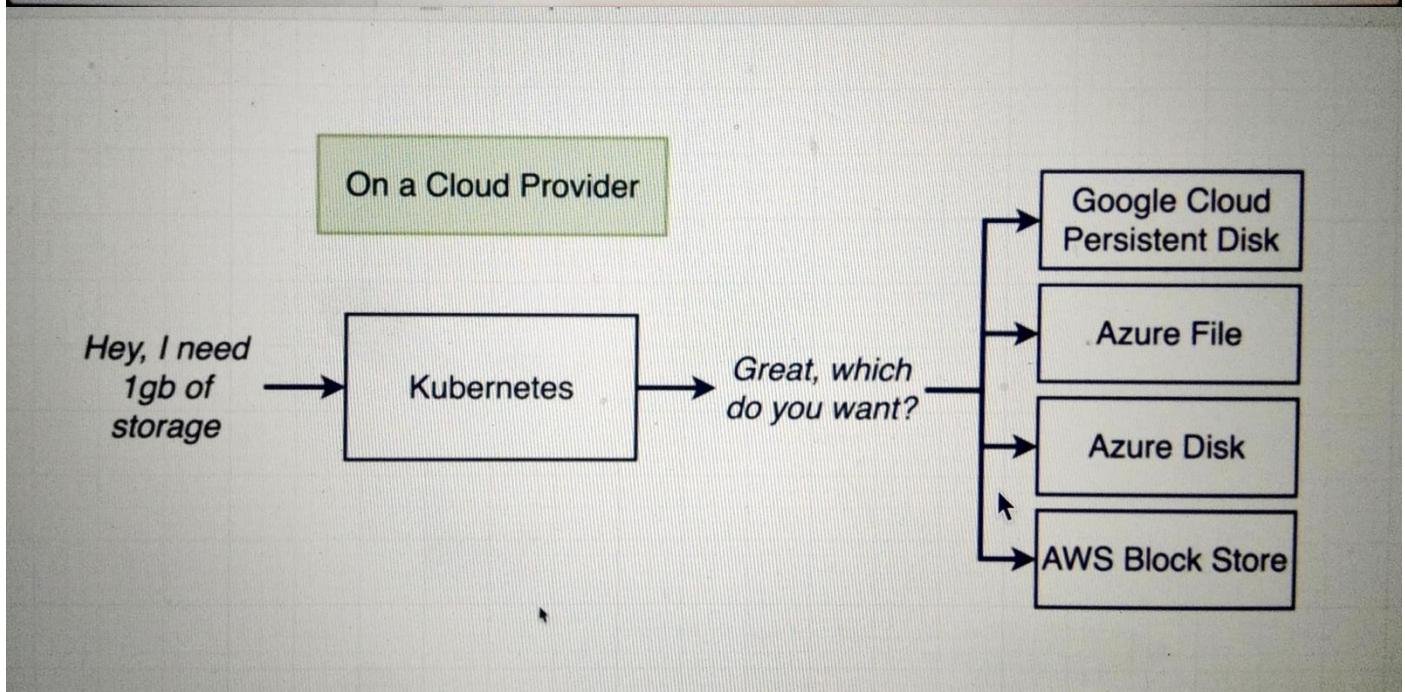
Kubernetes Engine Persistent volume claims

Persistent volume claims are a way of letting pods access the persistent volumes.

Learn more

Marketplace

This screenshot shows the Google Cloud Platform (GCP) Kubernetes Engine Storage interface. On the left, there's a sidebar with icons for Clusters, Workloads, Services, Applications, Configuration, and Storage. The Storage icon is highlighted. The main area has tabs for 'Persistent volume claims' (which is selected) and 'Storage classes'. A tooltip for 'Persistent volume claims' explains that they are a way of letting pods access persistent volumes. A 'Learn more' button is also present. The top navigation bar includes the GCP logo, project name 'multi-k8s', and a refresh button.



Secure https://console.cloud.google.com/kubernetes/storage?folder=&organizationId=&project=skillful-berm-214822&tab=storageClasses&persistent_volume_cl...  

☰ Google Cloud Platform • multi-k8s ▾

Kubernetes Engine Storage 

Clusters Persistent volume claims Storage classes

Workloads Storage classes are parameters for a class of storage for which persistent volumes are dynamically provisioned.

Services

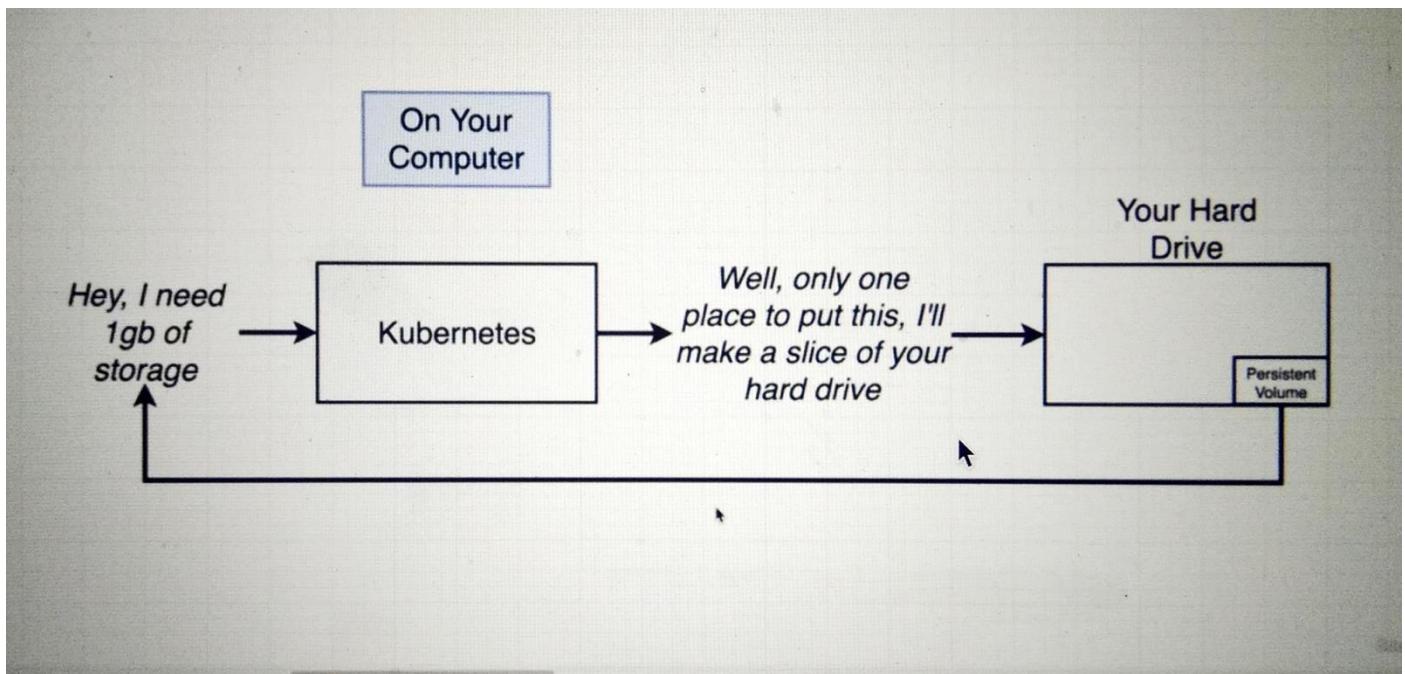
Applications Filter storage classes

Configuration

Storage Name Provisioner Type Zone Cluster

Name	Provisioner	Type	Zone	Cluster
standard	kubernetes.io/gce-pd	pd-standard		multi-cluster

Marketplace



Google Cloud Platform multi-k8s

Kubernetes Engine

Kubernetes clusters

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
multi-cluster	us-central1-a	3	3 vCPUs	11.25 GB		

Clusters Workloads Services Applications Configuration Storage

Marketplace

Google Cloud Platform multi-k8s

Kubernetes Engine

Clusters

Master zone	Node zones
us-central1-a	us-central1-a

Workloads Services Applications Configuration Storage

Marketplace

Google Cloud Platform multi-k8s

Kubernetes Engine Clusters EDIT DELETE DEPLOY CONNECT

Clusters Workloads Services Applications Configuration Storage Marketplace

Node Pools

Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher availability, or add node pools of different type machines. To add a node pool, click Edit. Learn more

default-pool (3 nodes, version 1.9.7-gke.6)

Name	default-pool
Size	3
Node version	1.9.7-gke.6
Node image	Container-Optimized OS (cos)
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Total cores	3 vCPUs
Total memory	11.25 GB
Automatic node upgrades	Disabled
Automatic node repair	Enabled

Google Cloud Platform multi-k8s

Kubernetes Engine Workloads REFRESH

Clusters **Workloads** Services Applications Configuration Storage Marketplace

Kubernetes Engine
Deploy a containerized application

Deploy, manage, and scale containers on Kubernetes, powered by Google Cloud. Learn more

Deploy or Show system workloads

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x StephenGrider/multi-k8s x StephenGrider/multi-k8s x Kubernetes Engine - multi-k8s x Stephen

Secure https://console.cloud.google.com/kubernetes/discovery?folder=&organizationId&project=skillful-berm-214822&serviceList_tableSize=50 Fri 10:13 AM

Google Cloud Platform multi-k8s

Kubernetes Engine Services REFRESH

Clusters Kubernetes services Brokered services **BETA**

Workloads Services **Services** Applications Configuration Storage

Kubernetes Engine Discovery & load balancing

Kubernetes discovered services, like load balancers, define a logical set of pods and a way to access them (a micro-service).

Learn more or Show system objects

Marketplace

https://console.cloud.google.com/kubernetes/discovery?project=skillful-berm-214822

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x StephenGrider/multi-k8s x StephenGrider/multi-k8s x Kubernetes Engine - multi-k8s x Stephen

Secure https://console.cloud.google.com/kubernetes/application?folder=&organizationId&project=skillful-berm-214822&applicationList_tableSize=50 Fri 10:13 AM

Google Cloud Platform multi-k8s

Kubernetes Engine Applications **BETA** REFRESH

Clusters Workloads Services Applications Configuration Storage

Kubernetes Engine Applications

Kubernetes Applications collect containers, services and configuration that are managed together. [Learn more](#)

Deploy from Marketplace

Marketplace

https://console.cloud.google.com/kubernetes/application?project=skillful-berm-214822

Chrome File Edit View History Bookmarks People Window Help

Secure https://console.cloud.google.com/kubernetes/config?folder=&organizationId=&project=skillful-berm-214822&config_list_tablesize=50 Fri 10:14 AM

Google Cloud Platform • multi-k8s

Kubernetes Engine Configuration

REFRESH

- Clusters
- Workloads
- Services
- Applications
- Configuration**
- Storage

Kubernetes Engine Configuration

Share your configuration through config files, command line arguments, and environment variables. These configuration objects let you maximize your containerized application's portability.

[Learn more](#) or [Show system objects](#)

Marketplace

https://console.cloud.google.com/kubernetes/storage?folder=&organizationId=&project=skillful-berm-214822&tab=persistentVolumeClaims&persistent_v... Fri 10:14 AM

Google Cloud Platform • multi-k8s

Kubernetes Engine Storage

REFRESH

- Clusters
- Workloads
- Services
- Applications
- Configuration
- Storage**

Persistent volume claims

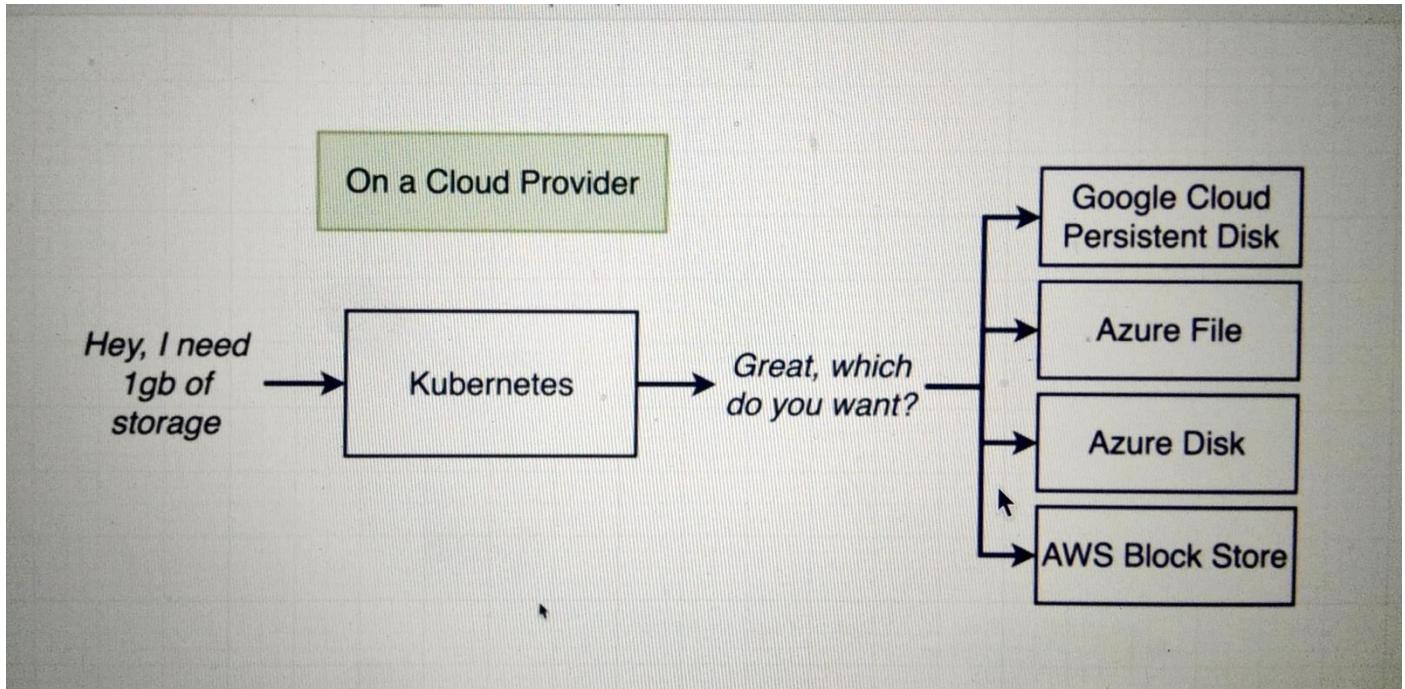
Storage classes

Kubernetes Engine Persistent volume claims

Persistent volume claims are a way of letting pods access the persistent volumes.

[Learn more](#)

Marketplace



Screenshot of the Google Cloud Platform Kubernetes Engine Storage page. The left sidebar shows navigation options: Clusters, Workloads, Services, Applications, Configuration, and Storage (which is selected). The main content area is titled "Storage" and includes a "REFRESH" button. It displays two tabs: "Persistent volume claims" and "Storage classes". A descriptive text states: "Storage classes are parameters for a class of storage for which persistent volumes are dynamically provisioned." Below this is a table titled "Filter storage classes" with columns: Name, Provisioner, Type, Zone, and Cluster. One row is visible: "standard" (Provisioner: kubernetes.io/gce-pd, Type: pd-standard, Cluster: multi-cluster).

Name	Provisioner	Type	Zone	Cluster
standard	kubernetes.io/gce-pd	pd-standard		multi-cluster

Github Actions

```

.github > workflows > 🛡 deployment.yaml
1   name: Deploy MultiK8s
2   on:
3     push:
4       branches:
5         - main
6
7   env:
8     SHA: $(git rev-parse HEAD)
9
10  jobs:
11    build:
12      runs-on: ubuntu-latest
13      steps:
14        - uses: actions/checkout@v3
15
16        - name: Test
17          run: |-
18            docker login -u ${{ secrets.DOCKER_USERNAME }} -p ${{ secrets.DOCKER_PASSWORD }}
19            docker build -t cygnetops/react-test -f ./client/Dockerfile.dev ./client
20            docker run -e CI=true cygnetops/react-test npm test
21
22        - name: Set Service Key
23          uses: "google-github-actions/auth@v0"
24          with:
25            credentials_json: "${{ secrets.GKE_SA_KEY }}"
26
27        - name: Set Project
28          uses: google-github-actions/setup-gcloud@v0
29          with:
30            project_id: multi-338920
31
32        - name: Auth
33          run: |-
34            gcloud --quiet auth configure-docker
35
36        - name: Get Credentials
37          uses: google-github-actions/get-gke-credentials@v0
38          with:
39            cluster_name: multi-cluster
40            location: us-central1-c
41

```

```

- name: Build
  run: |-
    docker build -t cygnetops/multi-client-k8s-gh:latest -t cygnetops/multi-client-k8s-gh:${{ env.SHA }} -f ./client/Dockerfile ./client
    docker build -t cygnetops/multi-server-k8s-pgfix-gh:latest -t cygnetops/multi-server-k8s-pgfix-gh:${{ env.SHA }} -f ./server/Dockerfile
    docker build -t cygnetops/multi-worker-k8s-gh:latest -t cygnetops/multi-worker-k8s-gh:${{ env.SHA }} -f ./worker/Dockerfile ./worker

- name: Push
  run: |-
    docker push cygnetops/multi-client-k8s-gh:latest
    docker push cygnetops/multi-server-k8s-pgfix-gh:latest
    docker push cygnetops/multi-worker-k8s-gh:latest

    docker push cygnetops/multi-client-k8s-gh:${{ env.SHA }}
    docker push cygnetops/multi-server-k8s-pgfix-gh:${{ env.SHA }}
    docker push cygnetops/multi-worker-k8s-gh:${{ env.SHA }}

- name: Apply
  run: |-
    kubectl apply -f k8s
    kubectl set image deployments/server-deployment server=cygnetops/multi-server-k8s-pgfix-gh:${{ env.SHA }}
    kubectl set image deployments/client-deployment client=cygnetops/multi-client-k8s-gh:${{ env.SHA }}
    kubectl set image deployments/worker-deployment worker=cygnetops/multi-worker-k8s-gh:${{ env.SHA }}

```

Steps for Creating a Service Account in GCP

1. Click the Hamburger menu on the top left-hand side of the dashboard, find **IAM & Admin**, and select **Service Accounts**. Then click the **CREATE SERVICE ACCOUNT** button.

The screenshot shows the 'Service accounts' page under the 'IAM & Admin' section. The left sidebar includes links for IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (which is selected and highlighted in blue), Labels, Settings, Privacy & Security, and Identity-Aware Proxy. The main area displays service accounts for the project 'multi-k8s'. It includes a brief description of what a service account is, a filter bar, and a table with columns for Email, Status, Name, Description, and Actions. Two service accounts are listed: 'compute@developer.gserviceaccount.com' and 'travis-deployer@steady-petal-307322.iam.gserviceaccount.com'. The 'Actions' column for the second account has a three-dot menu icon.

2. In the form that is displayed, set the **Service account name** to **travis-deployer** (step 1), then click the **CREATE** button (step 2).

The screenshot shows the 'Create service account' dialog. It consists of two main sections: 'Service account details' and 'Grant this service account access to project (optional)'.
In the 'Service account details' section:

- Service account name:** travis-deployer (highlighted with a red box labeled '1').
- Display name for this service account:** travis-deployer
- Service account ...:** travis-deployer @steady-petal-307322.iam.gserviceaccount.com (with a delete 'X' and refresh 'C' button).
- Service account description:** (empty field)

In the 'Grant this service account access to project (optional)' section:

- Grant users access to this service account (optional):** (empty field)

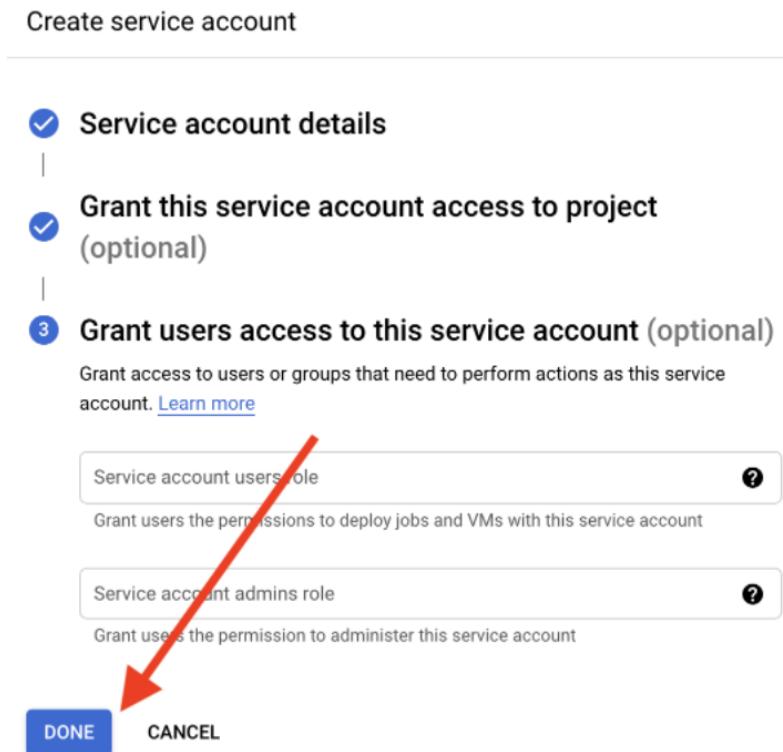
3. Click in the **Select a role** filter and scroll down to select **Kubernetes Engine** and then **Kubernetes Engine Admin**.

The screenshot shows a user interface for granting service account access. On the left, there's a sidebar with various project and organization management options. The main area has two sections: 'Service account details' (marked with a blue checkmark) and 'Grant this service account access to project (optional)'. In the 'Grant' section, there's a 'Select a role' dropdown with a 'Type to filter' input field. A dropdown menu lists several Kubernetes roles, with 'Kubernetes Engine Admin' highlighted and circled in red. Other listed roles include 'Kubernetes Engine Cluster Admin', 'Kubernetes Engine Cluster Viewer', 'Kubernetes Engine Developer', 'Kubernetes Engine Host Service Agent User', and 'Kubernetes Engine Viewer'. At the bottom of the dropdown is a 'MANAGE ROLES' link.

4. Make sure the filter now shows **Kubernetes Engine Admin** and then click **CONTINUE**

The screenshot shows the 'Create service account' process. It includes 'Service account details' (checked), 'Grant this service account access to project (optional)', and 'Grant users access to this service account (optional)'. In the 'Grant' section, the 'Role' dropdown is set to 'Kubernetes Engine Admin' (with a red arrow pointing to it). Below the dropdown is a description: 'Full management of Kubernetes Clusters and their Kubernetes API objects.' At the bottom of the 'Grant' section is a 'CONTINUE' button, which also has a red arrow pointing to it. The 'Grant users' section is partially visible at the bottom.

5. The Grant users access form is optional and should be skipped. Click the **DONE** button.



6. You should now see a table listing all of the service accounts including the one that was just created. Click the **three dots** to the right of the service account you just created. Then select **Manage Keys** in the dropdown.

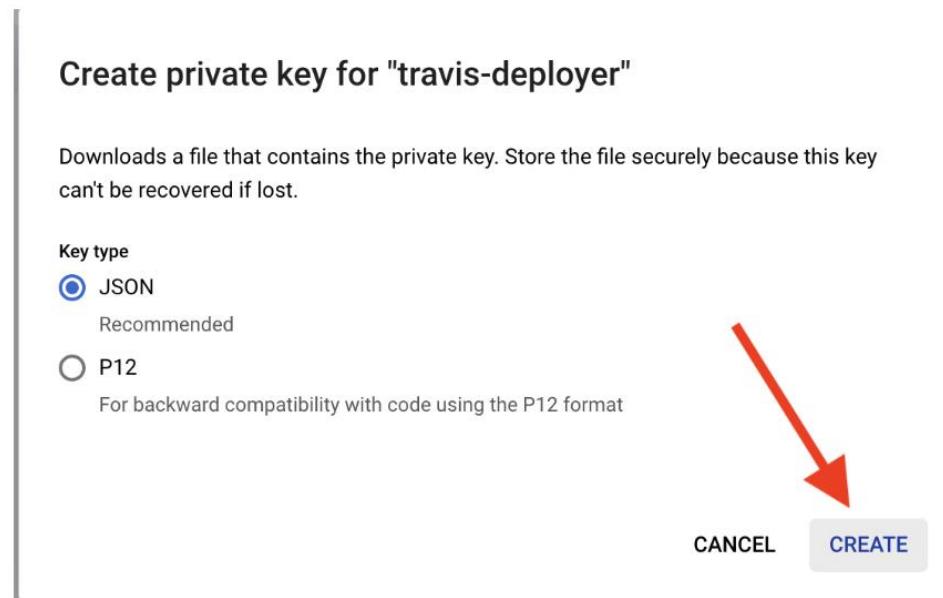
Service accounts for project "multi-k8s"					
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.					
Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.					
Filter <input type="text" value="Enter property name or value"/>	Status	Name ↑	Description	Actions	
<input type="checkbox"/> Email					
<input type="checkbox"/>  1020410104059-compute@googleapis.com	✓	Compute Engine default service account			
<input type="checkbox"/>  travis-deployer@steady-petal-307322.iam.gserviceaccount.com	✓	travis-deployer			
<input type="checkbox"/>  travis-deployer-395@steady-petal-307322.iam.gserviceaccount.com	✓	travis-deployer			

1: Manage details
2: Manage permissions
3: Manage keys
4: View metrics

7. In the **Keys** dashboard, click **ADD KEY** and then select **Create new key**.

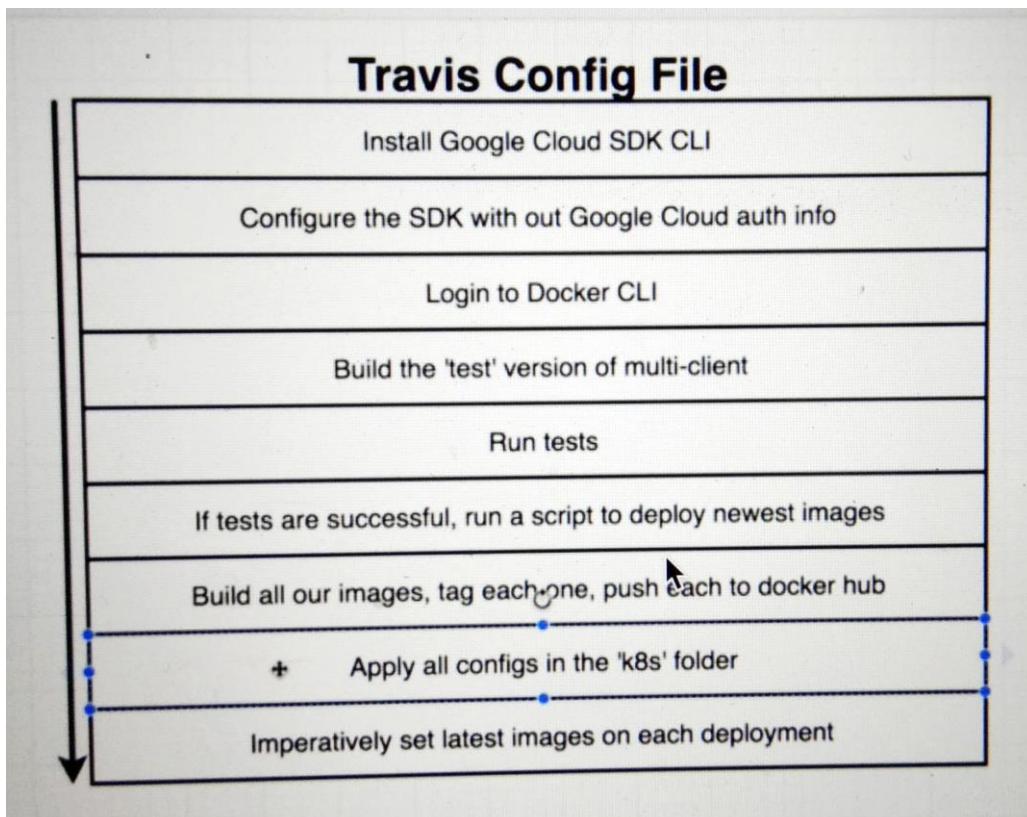
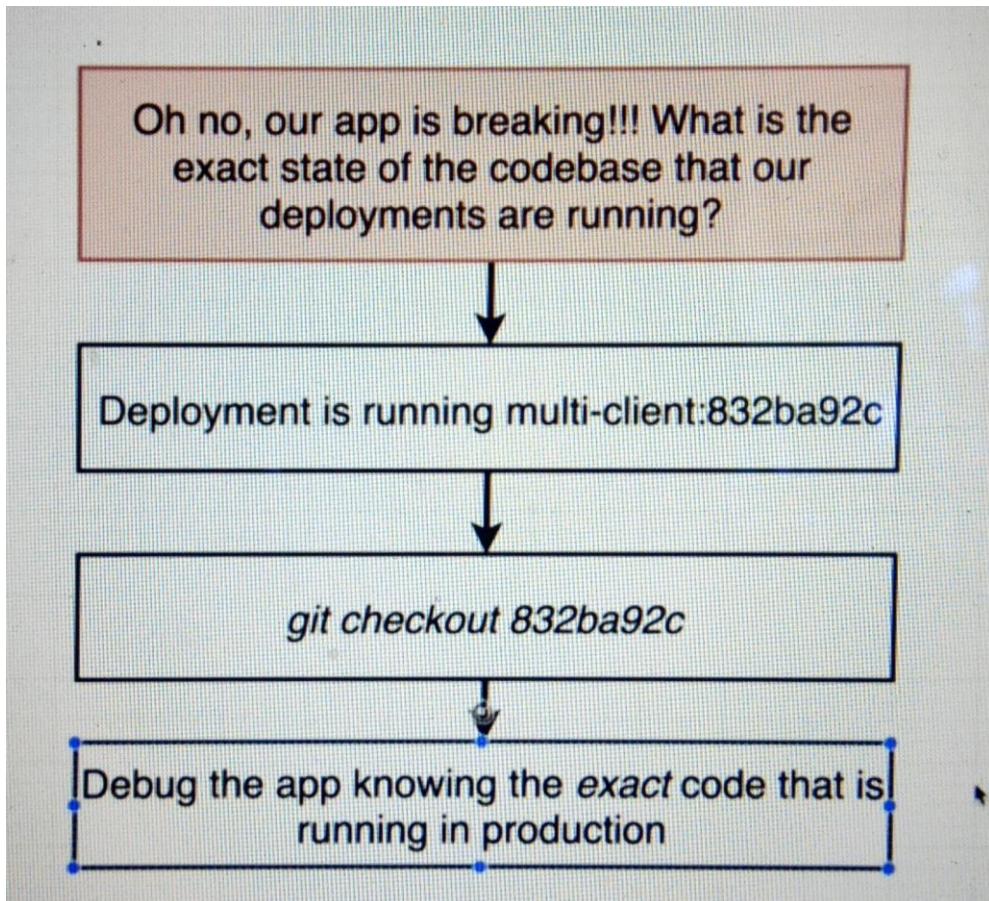
The screenshot shows the 'travis-deployer' project's Keys dashboard. At the top, there are tabs for DETAILS, PERMISSIONS, KEYS (which is underlined), METRICS, and LOGS. Below the tabs, the word 'Keys' is displayed. A note says: 'Add a new key pair or upload a public key certificate from an existing key pair. Please note that public certificates need to be in RSA_X509_PEM format. [Learn more about upload key formats](#)'. Another note says: 'Block service account key creation using [organization policies](#). [Learn more about setting organization policies for service accounts](#)'. A dropdown menu labeled 'ADD KEY' is open, showing three options: 'Create new key' (which has a red arrow pointing to it), 'Key creation date', and 'Key expiration date'. Below this, another option 'Upload existing key' is visible.

8. In the **Create private key** dialog box, make sure **Key type** is set to **JSON**, and then click the **CREATE** button.

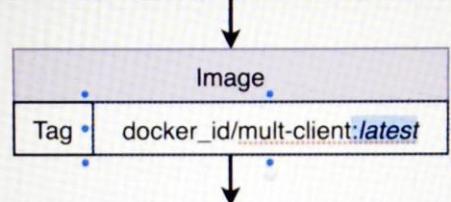


9. The JSON key file should now download to your computer.

Unique Tag for Build Image



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



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

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

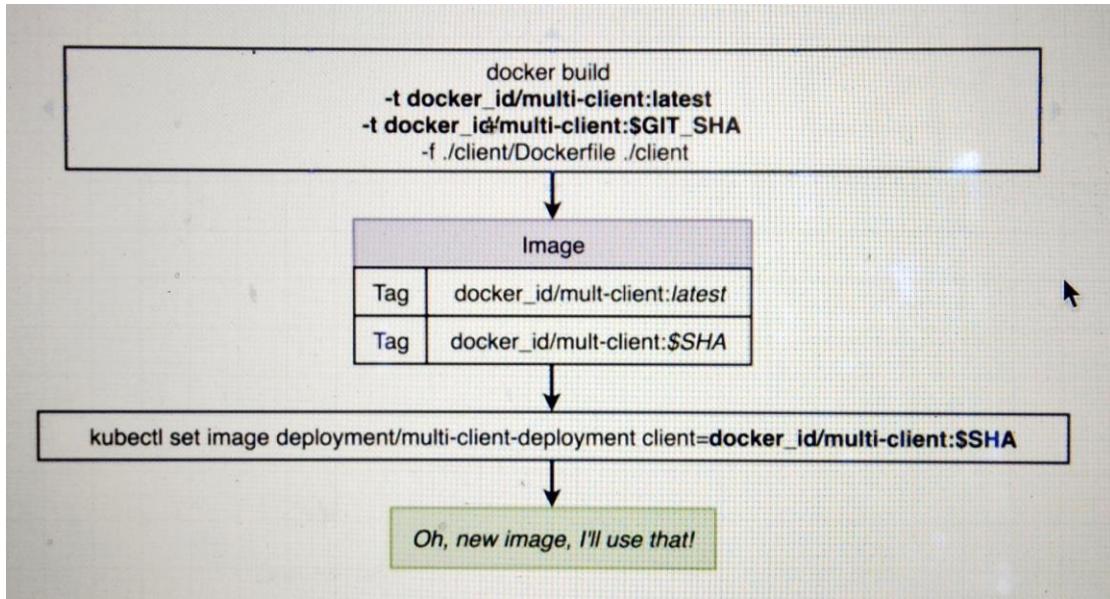
Update Image Version

Change deployment to use multi-client again

Update the multi-client image

Tag the image with a version number, push to
+ docker hub

Run a 'kubectl' command forcing the
deployment to use the new image version

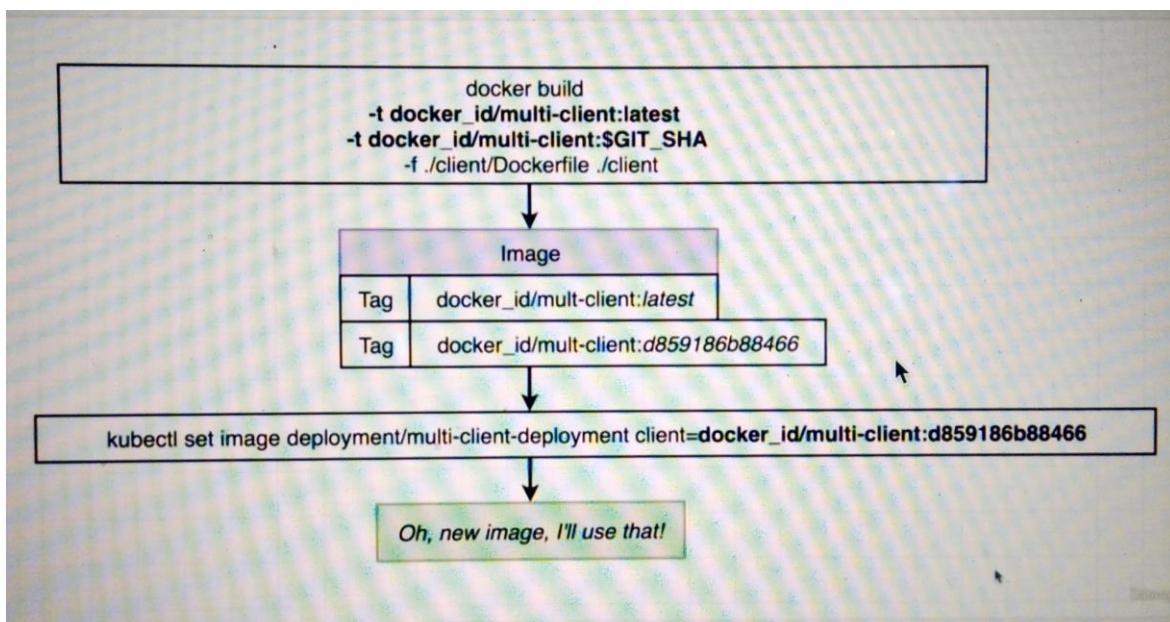


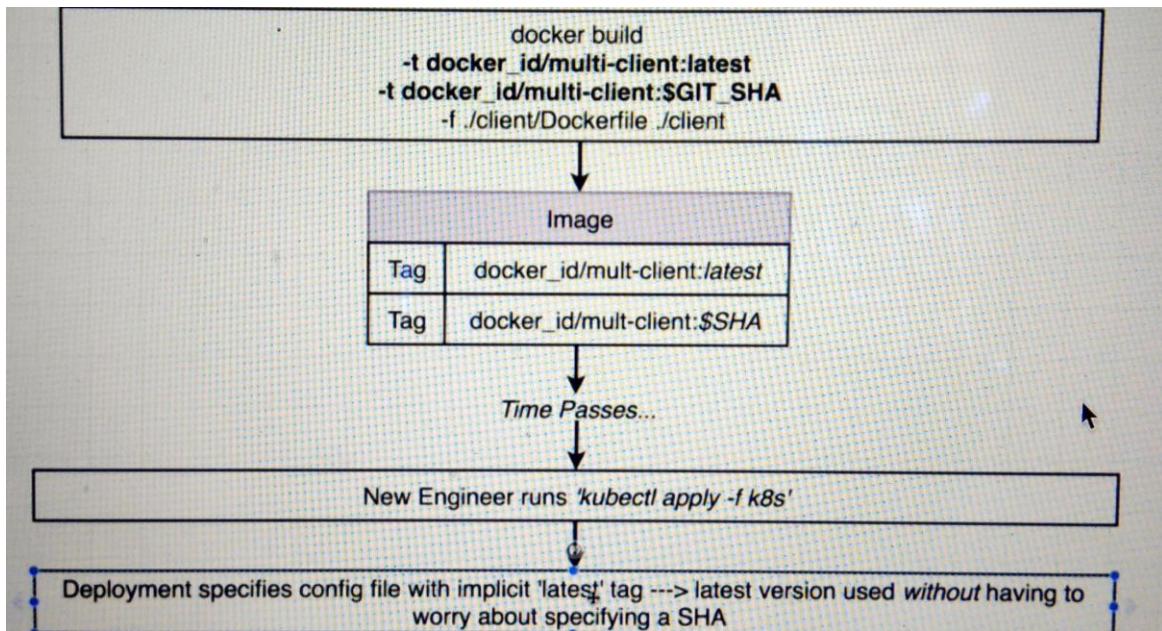
A screenshot of an iTerm2 terminal window. The title bar shows 'iTerm2 Shell View Session Scripts Profiles Toolbelt Window Help'. The main pane displays a git log output:

```

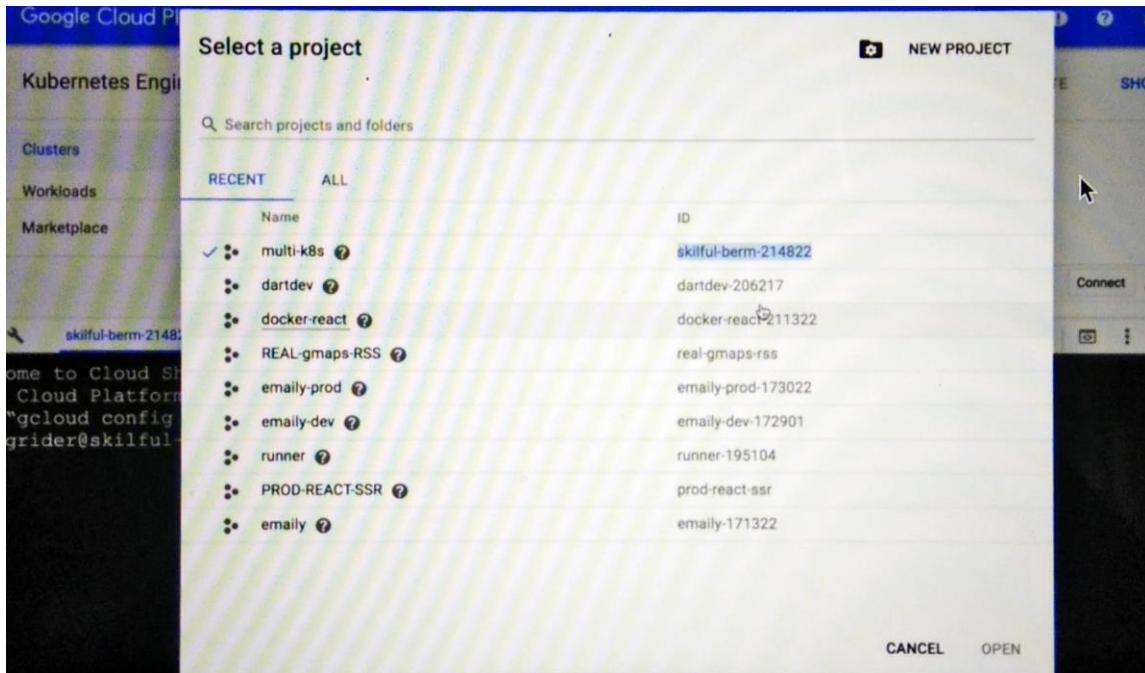
→ complex git:(master) ✘ git rev-parse HEAD
d859186b88466ce00a259183ef0a5f98060ca301
→ complex git:(master) ✘ git l
  
```

The terminal shows two commands: 'git rev-parse HEAD' which prints the commit hash, and 'git l' which is likely a typo for 'git log'.





Configuring the GCloud CLI on Cloud Console



Chrome File Edit View History Bookmarks People Window Help

Fri 2:25 PM

diagrams.xml - draw... diagrams.xml - draw... diagrams.xml - draw... diagrams.xml - draw... StephenGrider/multi... StephenGrider/multi... Kubernetes Engine

Secure https://console.cloud.google.com/kubernetes/list?project=skilful-berm-214822

Google Cloud Platform multi-k8s

Kubernetes Engine Kubernetes clusters + CREATE CLUSTER + DEPLOY ⌂ REFRESH ⌂ DELETE SHOW INFO

Clusters Workloads Marketplace

A Kubernetes cluster is a managed group of uniform VM instances for running Kubernetes. Learn more

Filter by label or name

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
multi-cluster	us-central1-a	3	3 vCPUs	11.25 GB		

Connect

skilful-berm-214822 ×

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to **skilful-berm-214822**.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
ste_grider@skilful-berm-214822:~\$ gcloud config set project skilful-berm-214822
Updated property [core/project].
ste_grider@skilful-berm-214822:~\$

Chrome File Edit View History Bookmarks People Window Help

Fri 2:25 PM

diagrams.xml - draw... diagrams.xml - draw... diagrams.xml - draw... diagrams.xml - draw... StephenGrider/multi... StephenGrider/multi... Kubernetes Engine

Secure https://console.cloud.google.com/kubernetes/list?project=skilful-berm-214822

Google Cloud Platform multi-k8s

Kubernetes Engine Kubernetes clusters + CREATE CLUSTER + DEPLOY ⌂ REFRESH ⌂ DELETE SHOW INFO PANEL

Clusters Workloads Marketplace

A Kubernetes cluster is a managed group of uniform VM instances for running Kubernetes. Learn more

Filter by label or name

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
multi-cluster	us-central1-a	3	3 vCPUs	11.25 GB		

Connect

skilful-berm-214822 ×

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to **skilful-berm-214822**.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
ste_grider@skilful-berm-214822:~\$ gcloud config set project skilful-berm-214822
Updated property [core/project].
ste_grider@skilful-berm-214822:~\$ gcloud config set compute/zone us-central1-a
Updated property [compute/zone].
ste_grider@skilful-berm-214822:~\$ gcloud container clusters get-credentials multi-cluster
Fetching cluster endpoint and auth data.
kubeconfig entry generated for multi-cluster.
ste_grider@skilful-berm-214822:~\$

Gdemy

Helm V3 Update

In the next lecture, we will be installing Helm. Helm v3 has since been released which is a major update, as it removes the use of Tiller. Please follow the updated instructions for this version below:

1. Install Helm v3:

In your Google Cloud Console run the following:

```
1 | curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3
2 | chmod 700 get_helm.sh
3 | ./get_helm.sh
4 |
```

link to the docs:

<https://helm.sh/docs/intro/install/#from-script>

2. Skip the commands run in the following lectures:

Helm Setup, Kubernetes Security with RBAC, Assigning Tiller a Service Account, and Ingress-Nginx with Helm. You should still watch these lectures and they contain otherwise useful info.

3. Install Ingress-Nginx:

In your Google Cloud Console run the following:

```
1 | helm repo add ingress-nginx https://kubernetes.github.io/ingress-
nginx
2 | helm install my-release ingress-nginx/ingress-nginx
3 |
```

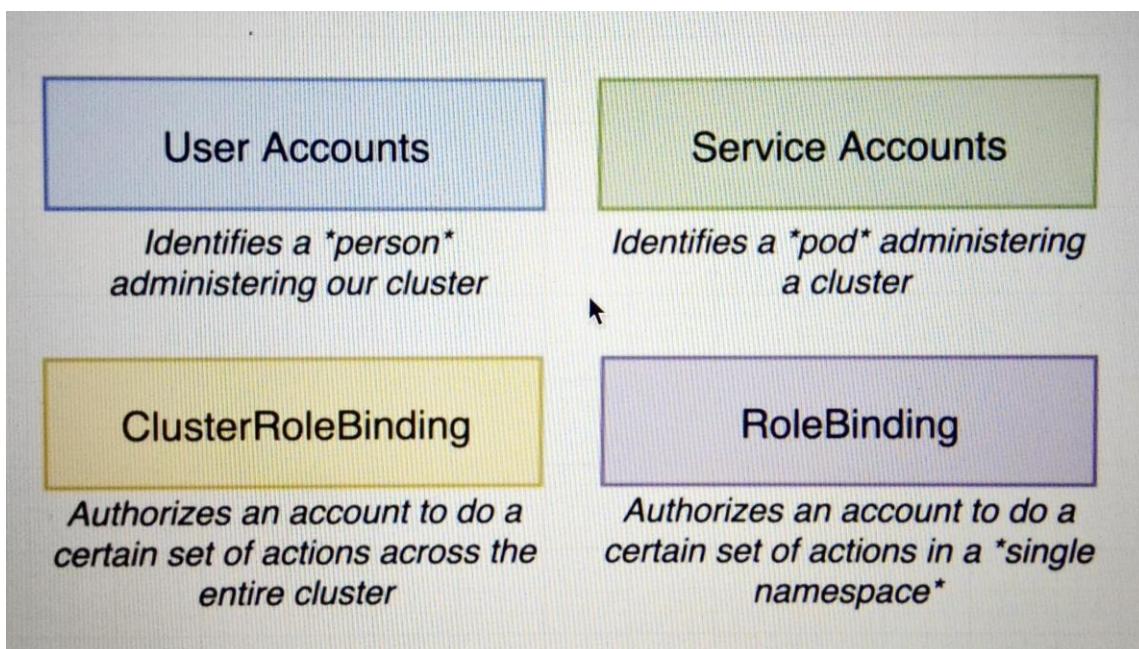
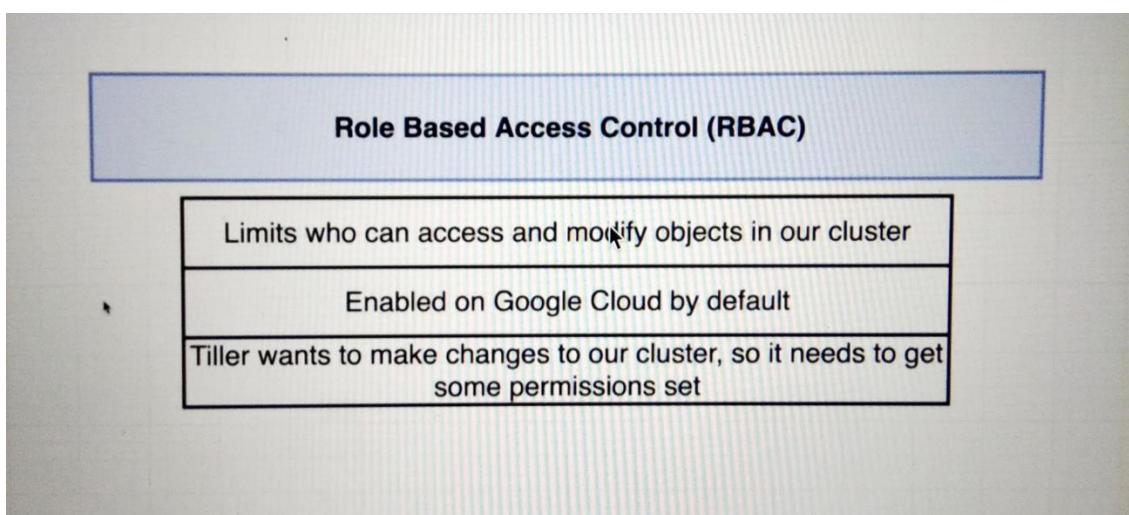
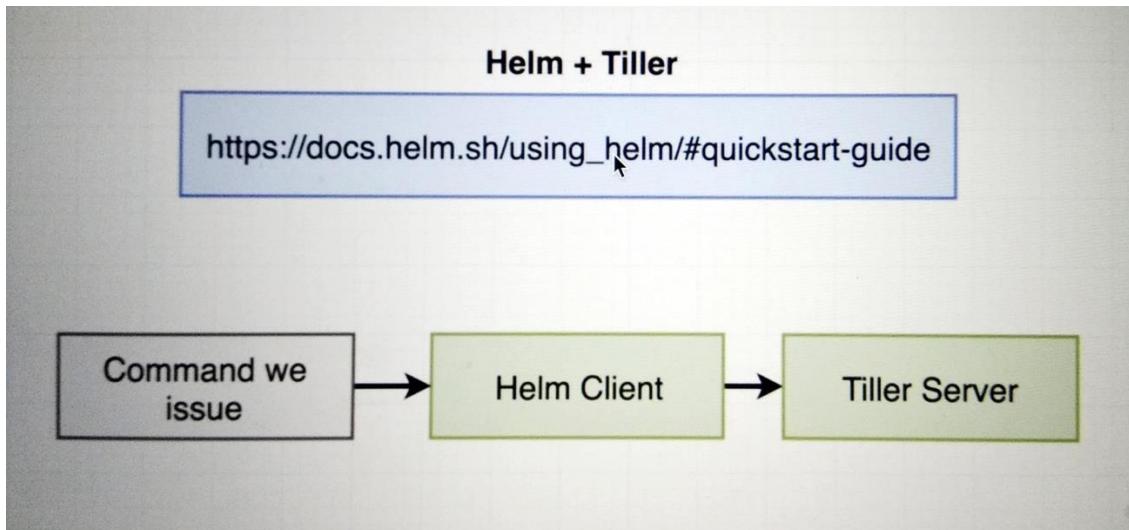
IMPORTANT: If you get an error such as *chart requires kubeVersion: >=1.16.0-0.....*

You may need to manually upgrade your cluster to at least the version specified:

```
gcloud container clusters upgrade YOUR_CLUSTER_NAME --
master --cluster-version 1.16
```

This should not be a long term issue since Google Cloud should handle this automatically:

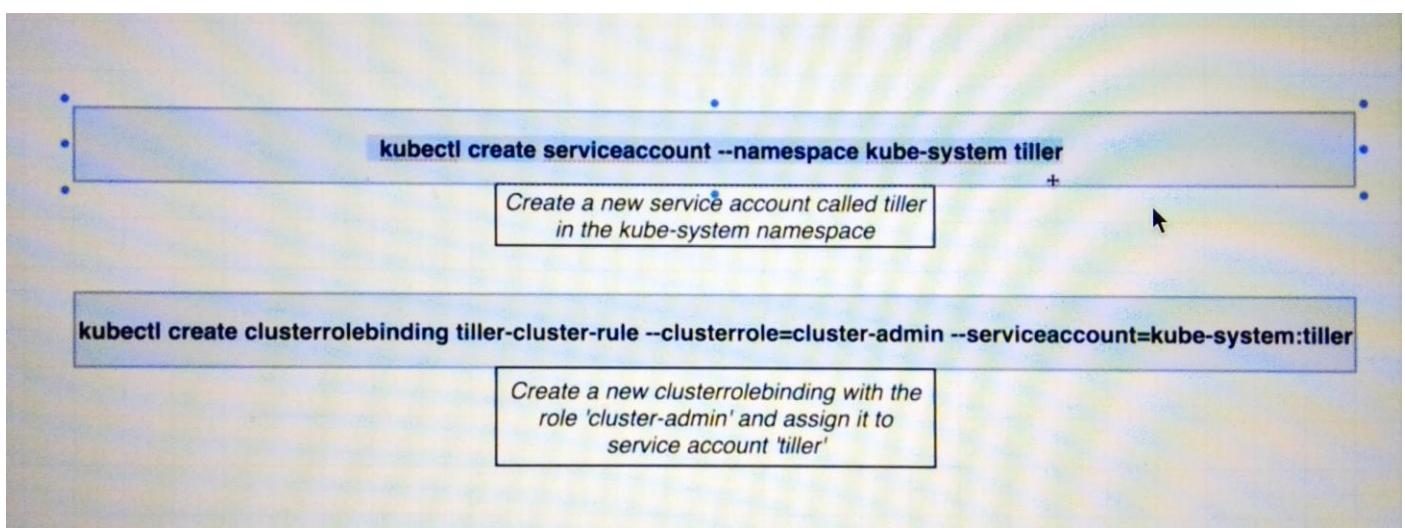
<https://cloud.google.com/kubernetes-engine/docs/how-to/upgrading-a-cluster>



The screenshot shows the Google Cloud Platform Kubernetes Engine interface. At the top, there are navigation links for 'Kubernetes Engine' and 'Kubernetes clusters', along with a 'CREATE CLUSTER' button. Below this, a 'Marketplace' section is visible. A central terminal window displays the following command output:

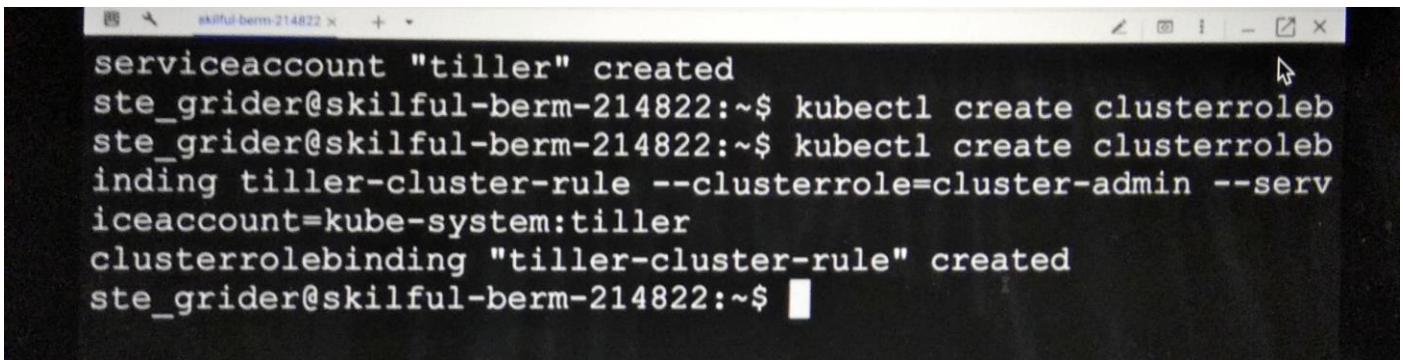
```
ste_grider@skilful-berm-214822:~$ kubectl create pod .alksdjlakjdsf^C
ste_grider@skilful-berm-214822:~$ kubectl get namespaces
NAME        STATUS   AGE
default     Active   1d
kube-public Active   1d
kube-system Active   1d
ste_grider@skilful-berm-214822:~$
```

Assigning Tiller a Service Account

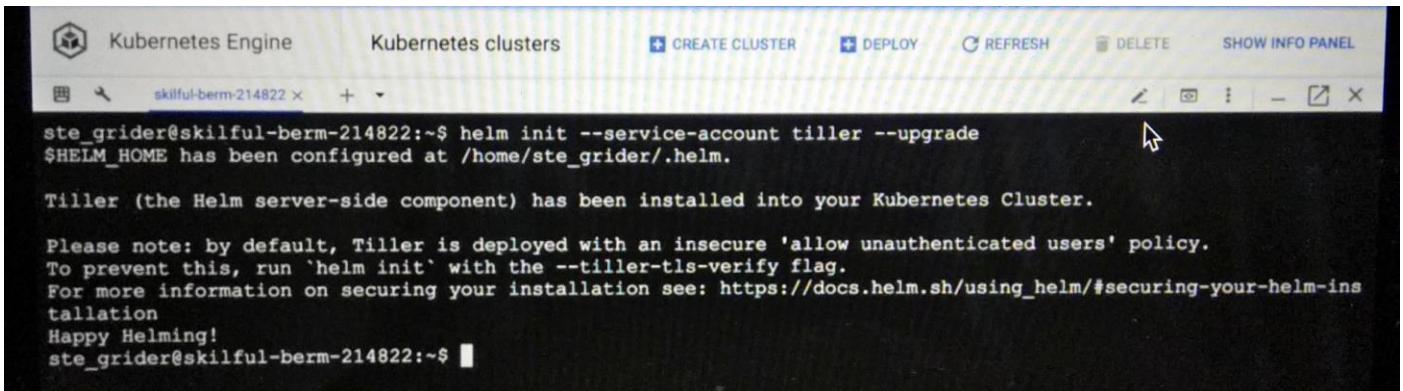


The screenshot shows the Google Cloud Platform Kubernetes Engine interface again. The terminal window now displays the successful execution of the commands from the previous screenshot:

```
ste_grider@skilful-berm-214822:~$ kubectl create serviceaccount --namespace kube-system tiller
serviceaccount "tiller" created
ste_grider@skilful-berm-214822:~$ kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
```



```
serviceaccount "tiller" created
ste_grider@skilful-berm-214822:~$ kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
clusterrolebinding "tiller-cluster-rule" created
ste_grider@skilful-berm-214822:~$
```



```
ste_grider@skilful-berm-214822:~$ helm init --service-account tiller --upgrade
$HELM_HOME has been configured at /home/ste_grider/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
Happy Helming!
ste_grider@skilful-berm-214822:~$
```

Ingress-Nginx with Helm

Quick Note about the Default Backend

In the next lecture, you will see the **Services** dashboard showing an ingress controller and default backend. A default backend no longer ships with ingress-nginx, so, if you only see a controller and you get a **404 Not Found** when visiting the IP address, this is perfectly expected.

The Result of Ingress Nginx



```
ste_grider@skilful-berm-214822:~$ helm install stable/nginx-ingress --name my-nginx --set rbac.create=true
```

```

http:
  paths:
    - backend:
        serviceName: exampleService
        servicePort: 80
      path: /
  # This section is only required if TLS is to be enabled for the Ingress
  tls:
    - hosts:
      - www.example.com
      secretName: example-tls

If TLS is enabled for the Ingress, a Secret containing the certificate and key must also be provided:

apiVersion: v1
kind: Secret
metadata:
  name: example-tls
  namespace: foo
data:
  tls.crt: <base64 encoded cert>
  tls.key: <base64 encoded key>
type: kubernetes.io/tls
ste_grider@skilful-berm-214822:~$ 

```

Google Cloud Platform multi-k8s

Kubernetes Engine Workloads

Clusters Workloads Services Applications Configuration Storage

Workloads are deployable units of computing that can be created and managed in a cluster.

Name	Status	Type	Pods	Namespace	Cluster
my-nginx-ingress-controller	OK	Deployment	1/1	default	multi-cluster
my-nginx-ingress-default-backend	OK	Deployment	1/1	default	multi-cluster

Google Cloud Platform multi-k8s

Kubernetes Engine Services

Clusters Workloads Services Applications Configuration Storage Marketplace

Kubernetes services Brokered services BETA

Services are sets of pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to services.

Name	Status	Service Type	Endpoints	Pods	Namespace	Cluster
my-nginx-ingress-controller	Ok	Load balancer	35.224.40.71:80 35.224.40.71:443	1 / 1	default	multi-cluster
my-nginx-ingress-default-backend	Ok	Cluster IP	10.11.240.39	1 / 1	default	multi-cluster

Google Cloud Platform multi-k8s

Network services Load balancer details

Load balancing a502ff2c0ad6511e8a28f42010a8001a

Cloud DNS

Cloud CDN

Frontend

Protocol	IP-Port	Network Tier
TCP	35.224.40.71:80-443	Premium

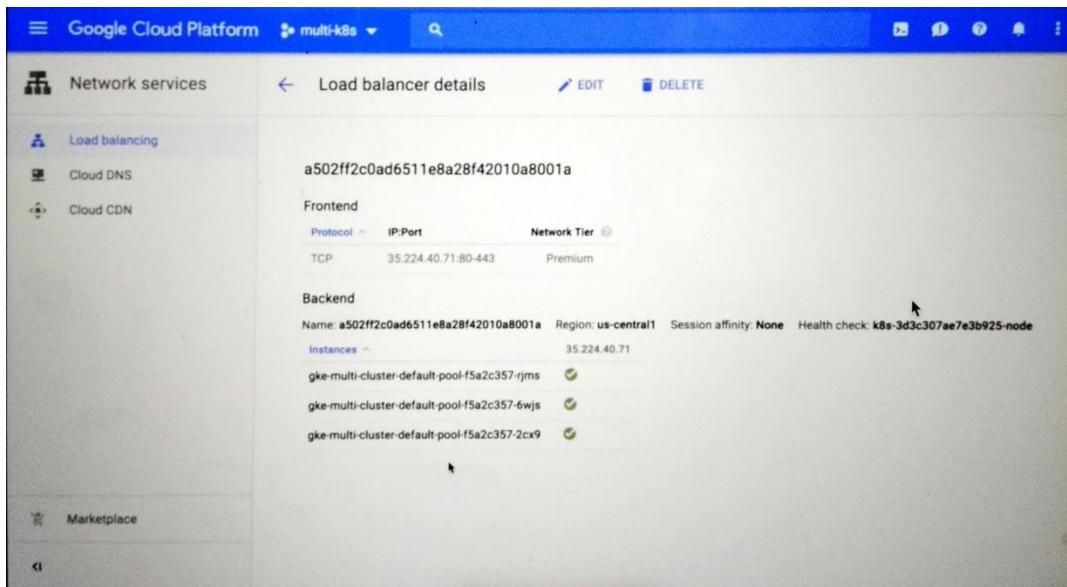
Backend

Name	Region	Session affinity	Health check
a502ff2c0ad6511e8a28f42010a8001a	us-central1	None	k8s-3d3c307ae7e3b925-node

Instances

- 35.224.40.71
- gke-multi-cluster-default-pool-f5a2c357-rljms
- gke-multi-cluster-default-pool-f5a2c357-6wjs
- gke-multi-cluster-default-pool-f5a2c357-2cx9

Marketplace



Verifying Deployment

Google Cloud Platform multi-k8s

Kubernetes Engine Deployments

Clusters Workloads Services Applications Configuration Storage Marketplace

Logs Container logs, Audit logs

Replicas 3 updated, 3 ready, 3 available, 0 unavailable

Pod specification Revision 2, containers: server

Active revisions

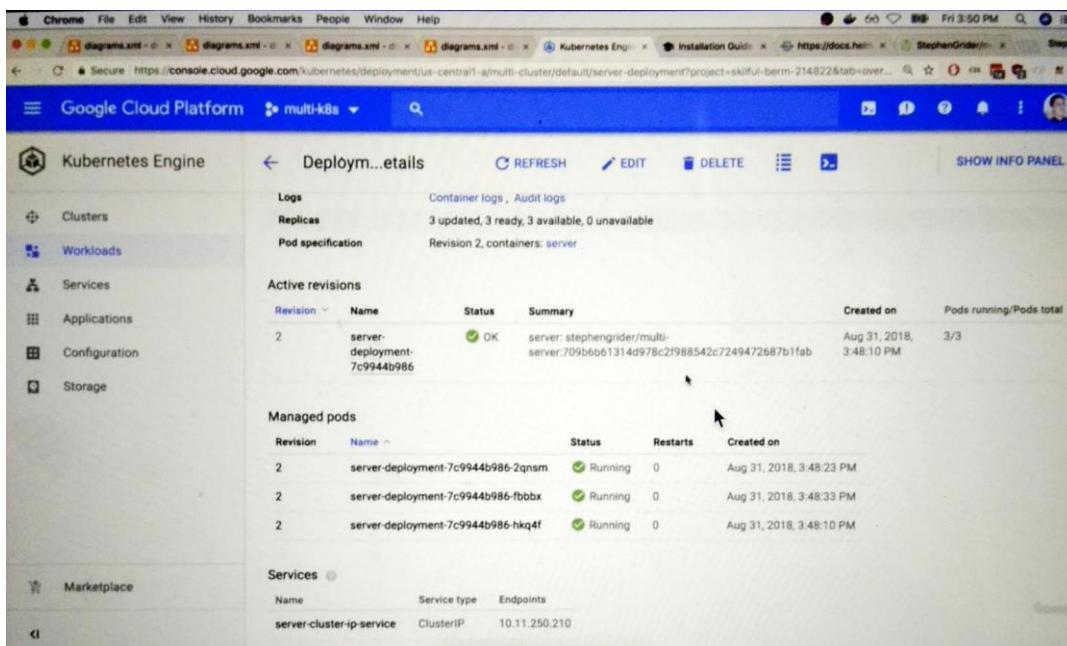
Revision	Name	Status	Summary	Created on	Pods running/Pods total
2	server-deployment-7c9944b986	OK	server: stephengrider/multi-server:709b6061314d978c2f988542c7249472687b1fb	Aug 31, 2018, 3:48:10 PM	3/3

Managed pods

Revision	Name	Status	Restarts	Created on
2	server-deployment-7c9944b986-2qnsm	Running	0	Aug 31, 2018, 3:48:23 PM
2	server-deployment-7c9944b986-fbbbx	Running	0	Aug 31, 2018, 3:48:33 PM
2	server-deployment-7c9944b986-hkq4f	Running	0	Aug 31, 2018, 3:48:10 PM

Services

Name	Service type	Endpoints
server-cluster-ip-service	ClusterIP	10.11.250.210
jonasse-service	Ingress	10.11.250.210



Travis CI StephenGrider / multi-k8s build unknown

Search all repositories

My Repositories

- StephenGrider/multi-k8s # 2 Duration: 2 min 39 sec Finished: less than a minute ago
- StephenGrider/multi-docker # 5 Duration: 2 min 21 sec Finished: 16 days ago
- StephenGrider/docker-readme # 10 Duration: 1 min 24 sec Finished: about a month ago

Current Branches Build History Pull Requests More options

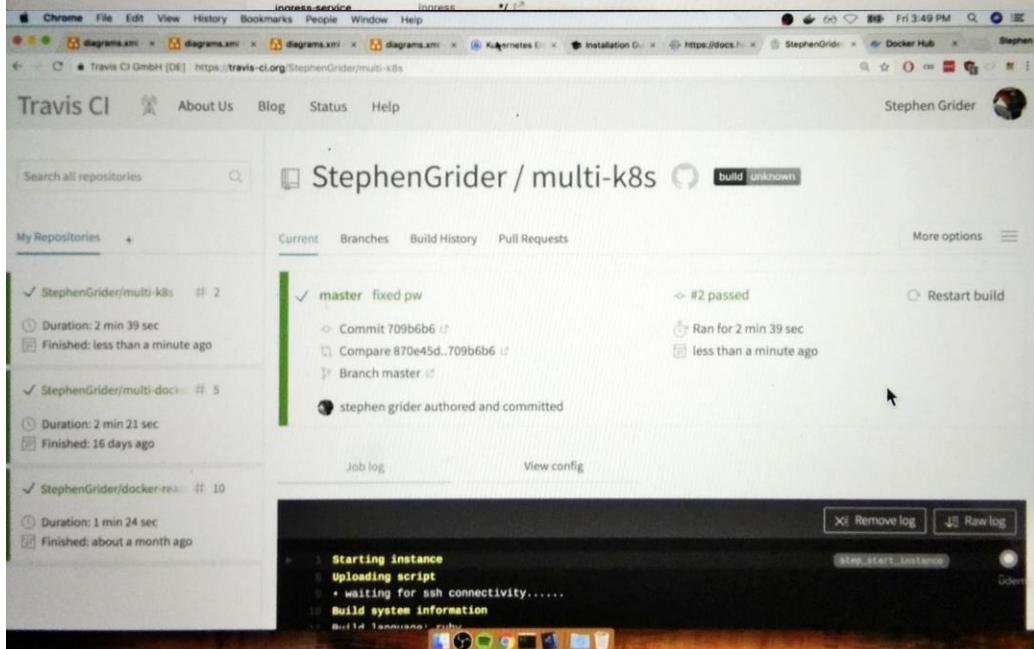
master fixed pw

- Commit 709b6b6
- Compare 870e45d...709b6b6
- Branch master

stephen grider authored and committed

Job log View config

```
Starting instance
Uploading script
+ waiting for ssh connectivity.....
Build system information
Build ID: 1anunno...zuhu
```



Google Cloud Platform multi-k8s

Kubernetes Engine Services

Kubernetes services **BETA**

Services are sets of pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to services.

Name	Status	Service Type	Endpoints	Pods	Namespace	Cluster
client-cluster-ip-service	Ok	Cluster IP	10.11.244.213	3 / 3	default	multi-cluster
ingress-service	Ok	Ingress	*:80 */api/:80	0 / 0	default	multi-cluster
my-nginx-ingress-controller	Ok	Load balancer	35.224.40.71:80 35.224.40.71:443	1 / 1	default	multi-cluster
my-nginx-ingress-default-backend	Ok	Cluster IP	10.11.240.39	1 / 1	default	multi-cluster
postgres-cluster-ip-service	Ok	Cluster IP	10.11.254.253	1 / 1	default	multi-cluster
redis-cluster-ip-service	Ok	Cluster IP	10.11.249.178	1 / 1	default	multi-cluster
server-cluster-ip-service	Ok	Cluster IP	10.11.250.210	3 / 3	default	multi-cluster

Google Cloud Platform multi-k8s

Kubernetes Engine Configuration

Secrets are sensitive pieces of information, like passwords, keys and tokens. Config maps are designed to store information that is not sensitive, like environment variables, command line arguments, and configuration files.

Secrets respect access control and are not visible to users without read permissions [Dismiss](#)

Name	Type	Namespace	Cluster
ingress-controller-leader-nginx	Config Map	default	multi-cluster
my-nginx-ingress-controller	Config Map	default	multi-cluster
my-nginx-ingress-token-k9lq6	Secret: service account	default	multi-cluster
pgpassword	Secret	default	multi-cluster

Google Cloud Platform multi-k8s

Kubernetes Engine Persistent volume claim details

database persistent-volume-claim

Cluster	multi-cluster
Namespace	default
Created	Aug 31, 2018, 3:48:05 PM
Labels	No labels set
Annotations	pv.kubernetes.io/bind-completed: yes pv.kubernetes.io/bound-by-controller: yes volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/gce-pd Show all annotations
Phase	Bound
Label selector	Empty label selector
Requested access modes	Read Write Once
Storage class	standard
Resources	Storage requested 2Gi capacity 2Gi
Volume	pvc-8ac8e5e3-ad67-11e8-a28f-42010a8001a4

Google Cloud Platform multi-k8s

Kubernetes Engine Persistent volume details

REFRESH EDIT DELETE

Clusters Workloads Services Applications Configuration Storage

Details YAML

pvc-8ac8e5e3-ad67-11e8-a28f-42010a8001a4

Cluster: multi-cluster
Created: Aug 31, 2018, 3:48:10 PM
Labels: failure-domain:region:us-central1 failure-domain:zone:us-central1-a
Annotations: kubernetes.io/createdby:gce-pd-dynamic-provisioner
pv.kubernetes.io/bound-by-controller: yes
pv.kubernetes.io/provisioned-by:kubernetes.io/gce-pd

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml (2) Not Secure https://35.224.40.71 Fri 3:53 PM

Fib Calculator version 2

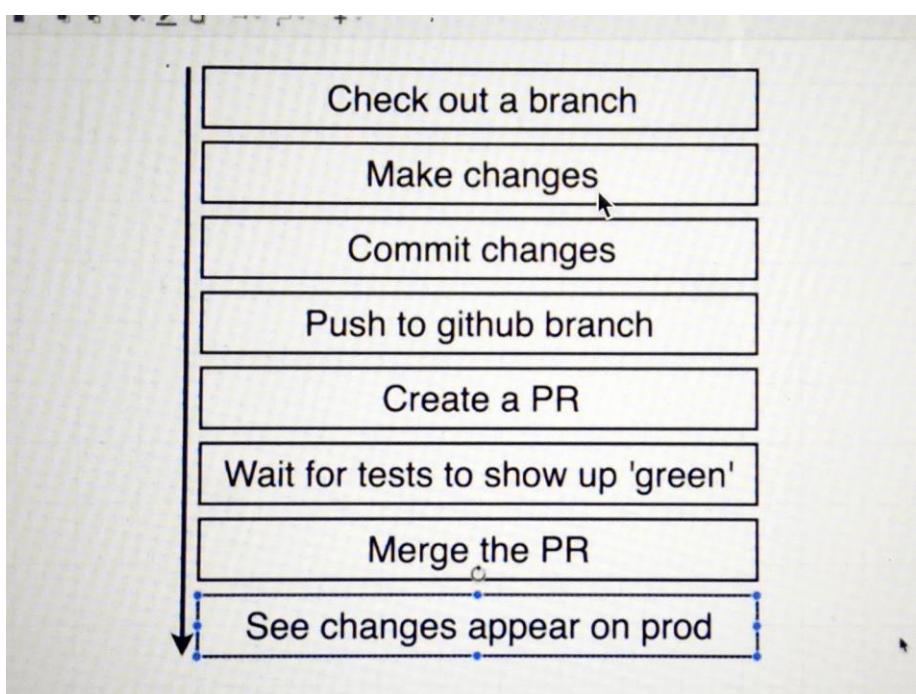
Home Other Page

Enter your index: Submit

Indexes I have seen:

Calculated Values:

A Workflow for Changing in Pods



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbar Window Help
1. stephengrider@stephens-MacBook-Pro: ~/workspace/DockerWorkspace/prod/complex [zsh]
→ complex git:(feature) ✘ git checkout -b devel
Switched to a new branch 'devel'
→ complex git:(devel) ┌
```

```
Code File Edit Selection View Go Debug Tasks Window Help
App.js — complex
OPEN EDITORS
COMPLEX
client
  nginx
  node_modules
  public
src
  App.css
  App.js
  App.test.js
  Fib.js
  index.css
  index.js
  logo.svg
  OtherPage.js
  registerServiceWorker.js
  Dockerfile
  Dockerfile.dev
  package-lock.json
  package.json
  README.md
k8s
  server
  worker
  .gitignore
OUTLINE
Ln 16, Col 70  Spaces: 2  UTF-8  LF  JavaScript  Prettier
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbar Window Help
1. stephengrider@stephens-MacBook-Pro: ~/workspace/DockerWorkspace/prod/complex (ssh)
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
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 423 bytes | 423.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To github.com:StephenGrider/multi-k8s.git
 * [new branch]      devel -> devel
→ complex git:(devel) ┌
```

A GitHub pull request page showing a merge from `devel` into `master`. The pull request has been updated by `StephenGrider`, who commented: "Change the header of the App.js file." A commit titled "updated header" has been pushed. The status bar indicates 2 pending checks. The branch has no conflicts with the base branch.

StephenGrider wants to merge 1 commit into master from devel.

Conversation 0 Commits 1 Checks 0 Files changed 1

StephenGrider commented just now

Change the header of the App.js file.

updated header bceff32

Add more commits by pushing to the `devel` branch on [StephenGrider/multi-k8s](#).

Some checks haven't completed yet

continuous-integration/travis-ci/pr continuous-integration/travis-ci/push

This branch has no conflicts with the base branch

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe

Chrome File Edit View History Bookmarks People Window Help

diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x diagrams.xml - draw.io x Kubernetes Engine x React App x StephenGrider/multi... x

Not Secure https://35.224.40.71 Fri 4:09 PM



Fib Calculator version KUBERNETES!

[Home](#) [Other Page](#)

Enter your index:

Indexes I have seen:

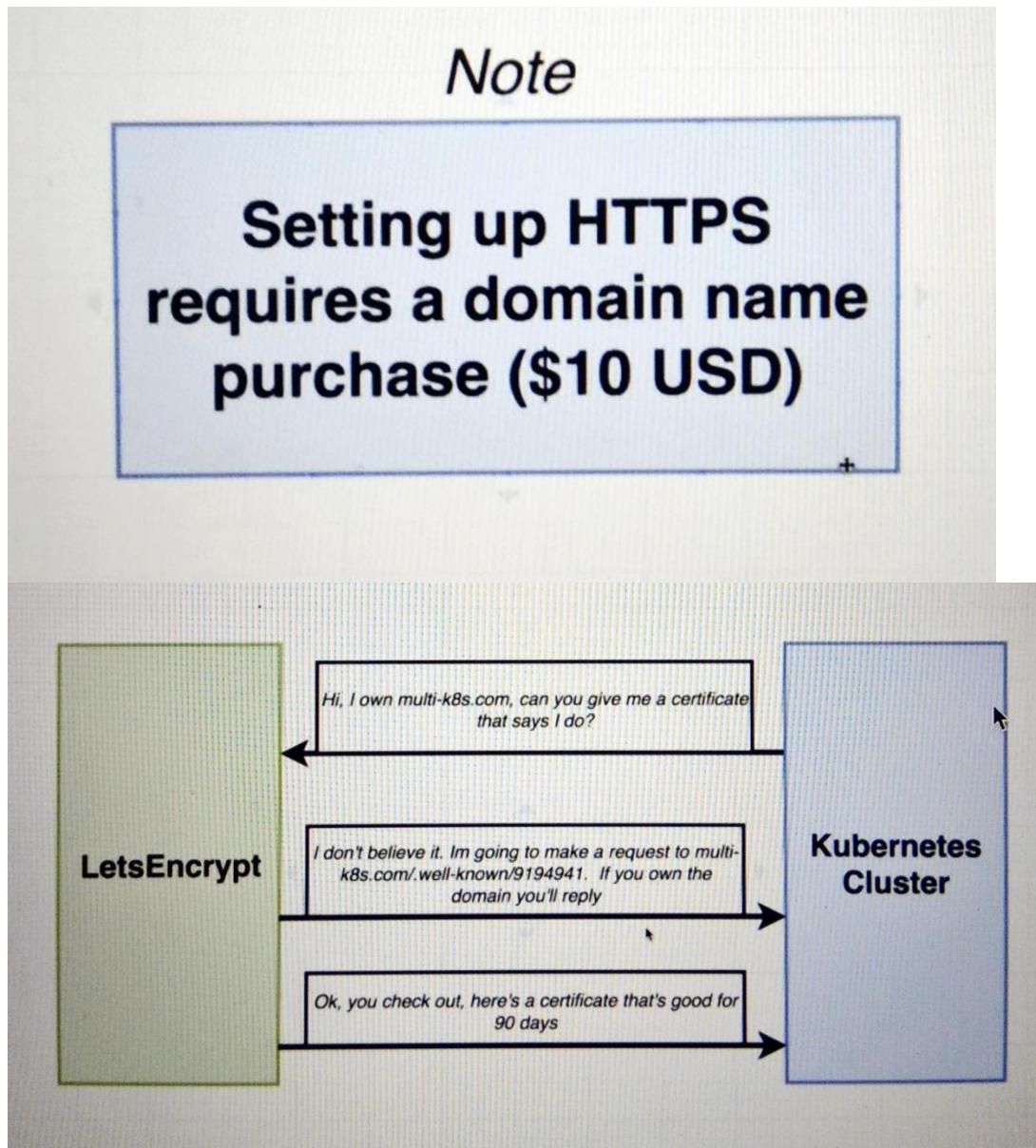
Calculated Values:

For index 4 I calculated 5

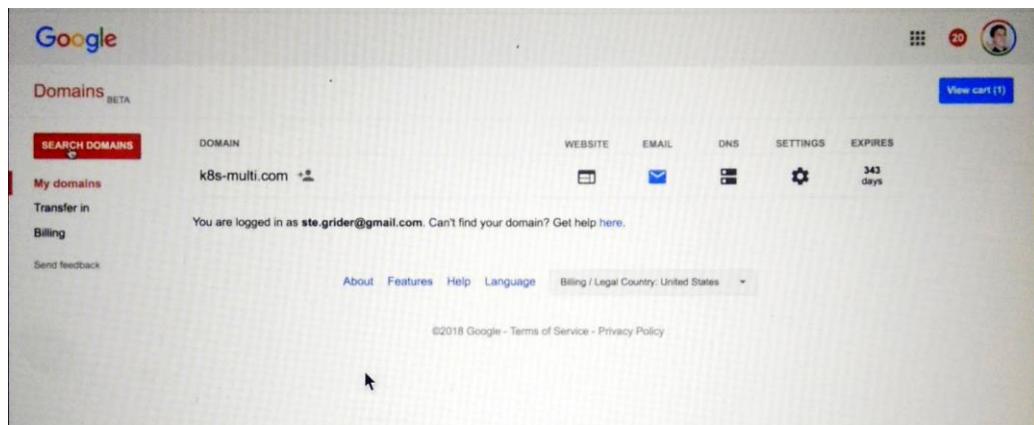
HTTPS Setup with Kubernetes

Overview

HTTP Set Overview



Domain Purchase



The image shows two screenshots from the Google Domains interface. The top screenshot displays a search results page for the query "laksdjflkasjdfasf". It lists several domain options, including ".com", ".net", ".org", ".cc", and ".info". The ".com" option is selected, and a message indicates it has been added to the cart. The bottom screenshot shows the details for "laksdjflkasjdfasf.com", including registration price (\$12/year), privacy protection status, and auto-renewal settings. A "GO TO CART" button is visible at the bottom.

Domain Name Setup

The image shows the Google Cloud Platform (GCP) Kubernetes Engine Services page. The sidebar on the left lists "Clusters", "Workloads", "Services", "Applications", "Configuration", and "Storage". The "Services" tab is selected, showing a list of services. The table includes columns for Name, Status, Service Type, Endpoints, Pods, Namespace, and Cluster. Services listed include "client-cluster-ip-service", "ingress-service", "my-nginx-ingress-controller", "my-nginx-ingress-default-backend", "postgres-cluster-ip-service", "redis-cluster-ip-service", and "server-cluster-ip-service", all associated with the "multi-cluster" namespace.

Google Domains BETA

Domains BETA

SEARCH DOMAINS

k8s-multi.com

Name servers

My domains Transfer in Billing Send feedback

Name servers store the configuration of your domain as a collection of resource records. If you use the Google Domains name servers, you can use the rest of this page to configure your domain. Alternately, you can enter up to 12 custom name servers here and configure your domain with your third-party DNS provider. Learn more

Configure DNS

343 days

• Use the Google Domains name servers
○ Use custom name servers

NAME SERVER

ns-cloud-d1.googledomains.com
ns-cloud-d2.googledomains.com
ns-cloud-d3.googledomains.com
ns-cloud-d4.googledomains.com

Save

DNSSEC

Google Domains BETA

Domains BETA

SEARCH DOMAINS

k8s-multi.com

WEBSITE EMAIL DNS SETTINGS EXPIRES

You are logged in as ste.grider@gmail.com. Can't find your domain? Get help here.

Red arrow pointing to the DNS icon

About Features Help Language Billing / Legal Country: United States

©2018 Google - Terms of Service - Privacy Policy

Google Domains BETA

Domains BETA

SEARCH DOMAINS

Custōm resource records

My domains Transfer in Billing Send feedback

Resource records define how your domain behaves. Common uses include pointing your domain at your web server or configuring email delivery for your domain. You can add up to 100 resource records. Learn more

NAME TYPE TTL DATA

IPv4 address

Add

No custom resource records set up yet. Learn how to set up resource records

About Features Help Language Billing / Legal Country: United States

©2018 Google - Terms of Service - Privacy Policy

The screenshot shows the Google Domains interface for managing domain settings. A success message at the top states "Successfully saved changes to domain settings for k8s-multi.com. Dismiss." Below this, there are sections for "My domains" and "Custom resource records". The "Custom resource records" section contains a table with one entry: "@ A 1h 35.224.40.71". A red banner at the bottom of the page says "Added the new custom resource records".

Below this, another screenshot shows the same interface after adding a second record. The table now includes two entries: "@ A 1h 35.224.40.71" and "www CNAME 1h k8s-multi.com".

Cert Manager Install

1. Add the Jetstack Helm repository

```
helm repo add jetstack https://charts.jetstack.io
```

2. Update your local Helm chart repository cache:

```
helm repo update
```

3. Install the cert-manager Helm chart:

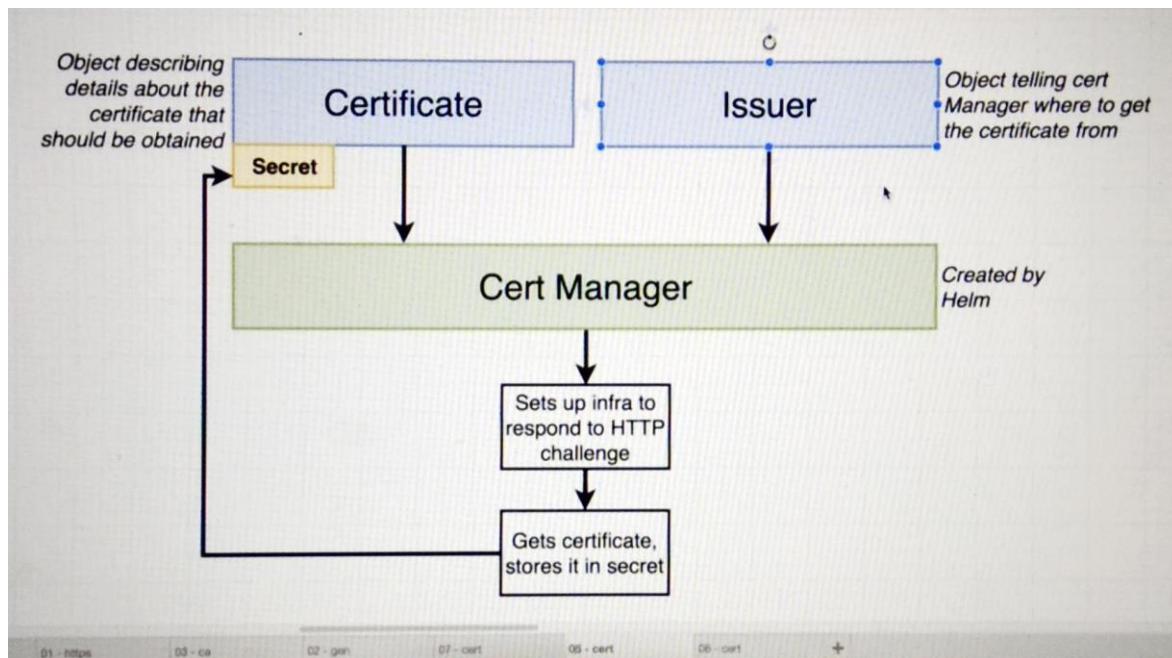
```
helm install \
cert-manager jetstack/cert-manager \
--namespace cert-manager \
--create-namespace \
--version v1.8.0 \
```

```
--set installCRDs=true
```

Official docs for reference:

<https://cert-manager.io/docs/installation/helm/#steps>

How to write up Cert Manager



Issuer Config File

```
1 | apiVersion: cert-manager.io/v1
2 | kind: ClusterIssuer
3 | metadata:
4 |   name: letsencrypt-prod
5 | spec:
6 |   acme:
7 |     server: https://acme-v02.api.letsencrypt.org/directory
8 |     email: "test@test.com"
9 |     privateKeySecretRef:
10 |       name: letsencrypt-prod
11 |     solvers:
12 |       - http01:
13 |         ingress:
14 |           class: nginx
```

Certificate Config File

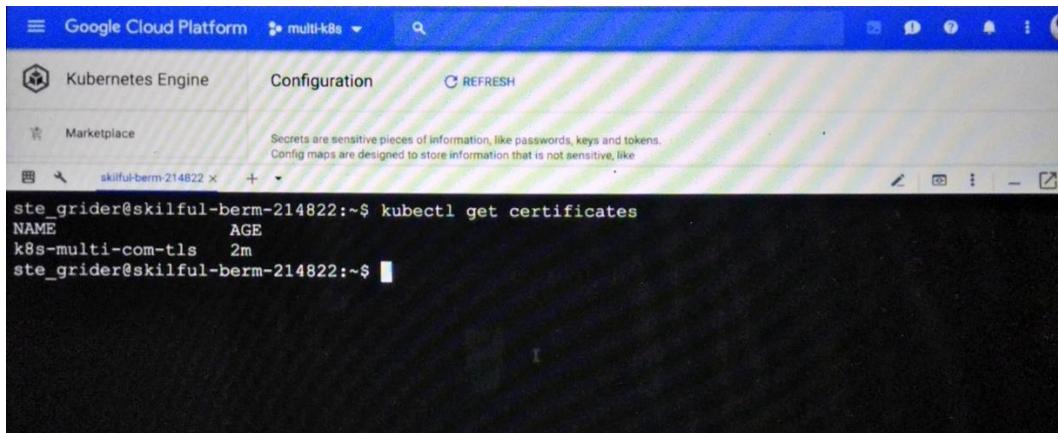
```
1 apiVersion: cert-manager.io/v1
2
3 kind: Certificate
4 metadata:
5   name: yourdomain-com-tls
6 spec:
7   secretName: yourdomain-com
8   issuerRef:
9     name: letsencrypt-prod
10    kind: ClusterIssuer
11   commonName: yourdomain.com
12   dnsNames:
13     - yourdomain.com
14     - www.yourdomain.com
```

Deployment

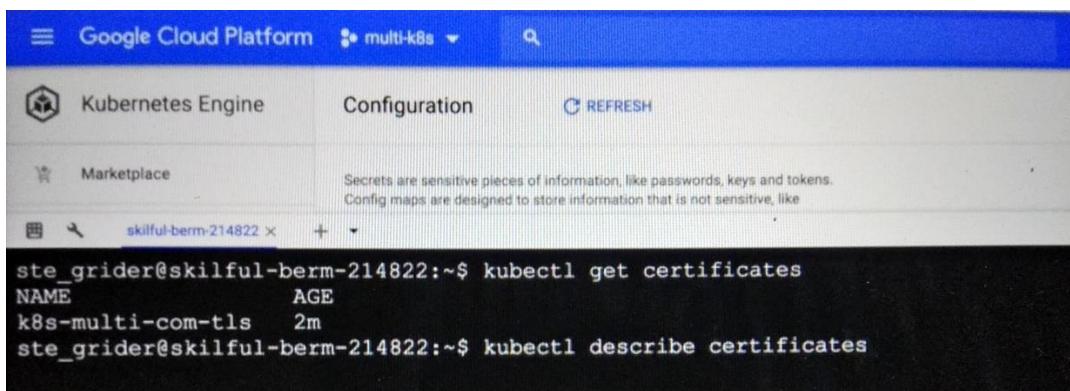
```
→ KUBERNATES-MULTICONTAINER-APP git:(main) git add .
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ git commit -m 'Https setup'

[main 448cc43] Https setup
 2 files changed, 28 insertions(+)
 create mode 100644 k8s/certificate.yaml
 create mode 100644 k8s/issuer.yaml
→ KUBERNATES-MULTICONTAINER-APP git:(main) git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 731 bytes | 731.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:mnavveensmn/kubernetes-multi-container.git
 3949ea5..448cc43 main -> main
→ KUBERNATES-MULTICONTAINER-APP git:(main) □
```

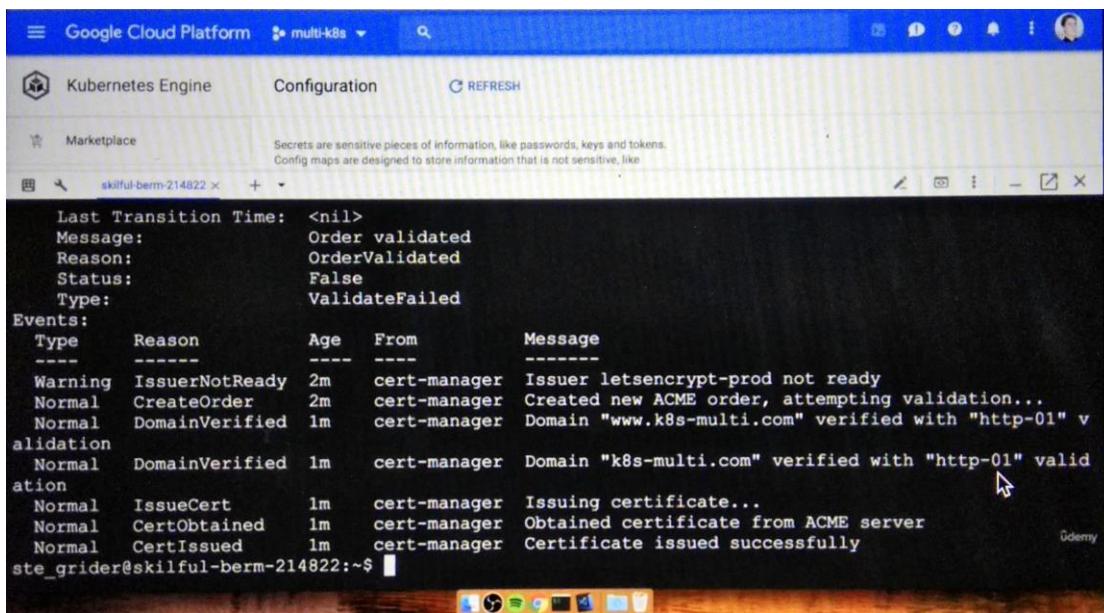
Ingress Config for HTTPS



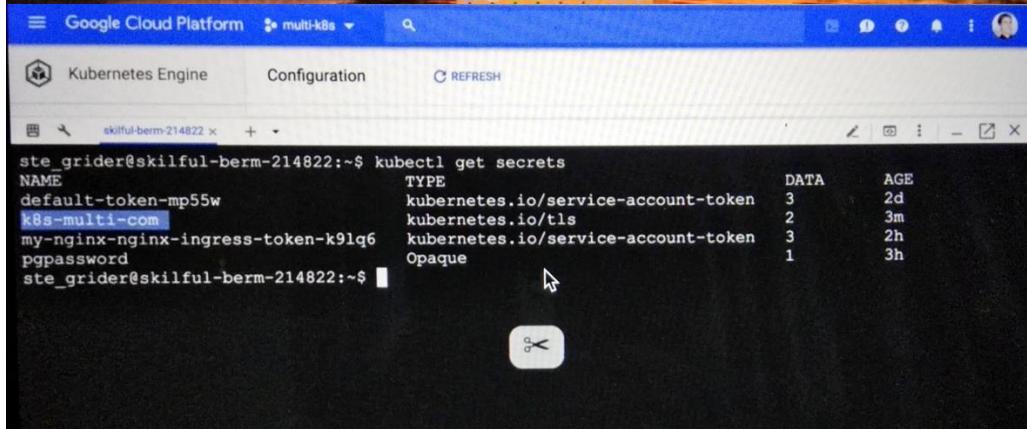
```
ste_grider@skilful-berm-214822:~$ kubectl get certificates
NAME          AGE
k8s-multi-com-tls  2m
ste_grider@skilful-berm-214822:~$
```



```
ste_grider@skilful-berm-214822:~$ kubectl get certificates
NAME          AGE
k8s-multi-com-tls  2m
ste_grider@skilful-berm-214822:~$ kubectl describe certificates
```



```
Last Transition Time: <nil>
Message:           Order validated
Reason:            OrderValidated
Status:             False
Type:              ValidateFailed
Events:
  Type    Reason     Age   From      Message
  ----  -----  ----  ----      -----
  Warning  IssuerNotReady  2m  cert-manager  Issuer letsencrypt-prod not ready
  Normal   CreateOrder    2m  cert-manager  Created new ACME order, attempting validation...
  Normal   DomainVerified 1m  cert-manager  Domain "www.k8s-multi.com" verified with "http-01" v
alidation
  Normal   DomainVerified 1m  cert-manager  Domain "k8s-multi.com" verified with "http-01" valid
ation
  Normal   IssueCert     1m  cert-manager  Issuing certificate...
  Normal   CertObtained   1m  cert-manager  Obtained certificate from ACME server
  Normal   CertIssued    1m  cert-manager  Certificate issued successfully
ste_grider@skilful-berm-214822:~$
```



```
ste_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-ingress-token-k9lq6  kubernetes.io/service-account-token  3     2h
pgpassword      Opaque                           1     3h
ste_grider@skilful-berm-214822:~$
```

Required Update for the HTTPS Ingress

In the upcoming lecture, we need to make one small change to one of the annotations:

Google Cloud Platform ➔ multi-k8s 🔍

Kubernetes Engine Services

REFRESH

Kubernetes services Brokered services BETA

Services are sets of pods with a network endpoint that can be used for discovery and load balancing. Ingresses are collections of rules for routing external HTTP(S) traffic to services.

	Name	Status	Service Type	Endpoints	Pods	Namespace	Cluster
	client-cluster-ip-service	Ok	ClusterIP	10.11.244.213	3 / 3	default	multi-cluster
	ingress-service	Ok	Ingress	k8s-multi.com/ ↗ www.k8s-multi.com/ ↗ www.k8s-multi.com/api/ ↗	0 / 0	default	multi-cluster
	my-nginx-ingress-controller	Ok	Load balancer	35.224.40.71:80 ↗ 35.224.40.71:443 ↗	1 / 1	default	multi-cluster
	my-nginx-ingress-default-backend	Ok	ClusterIP	10.11.240.39	1 / 1	default	multi-cluster
	postgres-cluster-ip-service	Ok	ClusterIP	10.11.254.253	1 / 1	default	multi-cluster
	redis-cluster-ip-service	Ok	ClusterIP	10.11.249.178	1 / 1	default	multi-cluster

The screenshot shows a browser window with multiple tabs open. The active tab is titled "Fib Calculator version KUBERNETES!". The page features a large blue glowing atom icon at the top center. Below it, the title "Fib Calculator version KUBERNETES!" is displayed in a bold, white, sans-serif font. A button labeled "Home" is visible above a text input field. The text input field contains the value "4" and is followed by a "Submit" button. Below the input field, the text "Indexes I have seen:" is displayed in bold. Underneath, the text "Calculated Values:" is also in bold. The output section shows the following text: "For index 4 I calculated 5".

k8s > [ingress-service.yaml](#) > {} spec > [] rules > {} 0 > {} http > [] paths > {} 1 > {} backend > {} s

```
1 apiVersion: networking.k8s.io/v1
2 # UPDATE API
3 kind: Ingress
4 metadata:
5   name: ingress-service
6   annotations:
7     kubernetes.io/ingress.class: "nginx"
8     nginx.ingress.kubernetes.io/use-regex: "true"
9     # ADD ANNOTATION
10    nginx.ingress.kubernetes.io/rewrite-target: /$1
11    # UPDATE ANNOTATION
12    cert-manager.io/cluster-issuer: "letsencrypt-prod"
13    nginx.ingress.kubernetes.io/ssl-redirect: "true"
14  spec:
15    tls:
16      - host:
17        - k8s-multi.com
18        - www.k8s-multi.com
19        secretName: k8s-multi-com
20    rules:
21      - host: k8s-multi.com
22        http:
23          paths:
24            - path: /?(.*)
25              # UPDATE PATH
26              pathType: Prefix
27              # ADD PATHTYPE
28              backend:
29                service:
30                  # UPDATE SERVICE FIELDS
31                  name: client-cluster-ip-service
32                  port:
33                    number: 3000
34            - path: /api/?(.*)
35              # UPDATE PATH
36              pathType: Prefix
37              # ADD PATHTYPE
38              backend:
39                service:
40                  # UPDATE SERVICE FIELDS
41                  name: server-cluster-ip-service
42                  port:
43                    number: 5000
44      - host: www.k8s-multi.com
45        http:
46          paths:
47            - path: /?(.*)
48              # UPDATE PATH
```

Google Cloud Cleanup

Time for some cleanup! **If you want to close down the Kubernetes cluster running on Google Cloud, do the following. Remember, you are paying for the running cluster!**

Steps to clean up:

1) Click the project selector on the top left of the page



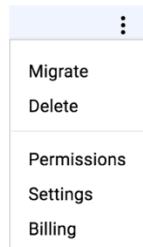
2) Click the 'gear' icon on the top right



3) Find your project in the list of projects that is presented, then click the three dots on the far right hand side



4) Click 'Delete'



5) Enter your project ID and click 'Shut Down'

Shut down project "multi-k8s"

When you shut down a project, this immediately happens:

- All billing and traffic serving stops
- You lose access to your entire project
- Project owners will be notified and can stop the shutdown within 30 days

The entire project will be scheduled to be deleted after 30 days.

To shut down project "multi-k8s", type your project ID: skilful-berm-214822

[CANCEL](#) [SHUT DOWN](#)

Local Environment Cleanup

You might want to also clean up some of the work you did on your local machine. Remember, we have a running Kubernetes cluster, and we have also built a ton of images.

Deleting Pods, Deployments, Services from the Multi K8's project

In the root project directory, run `kubectl delete -f k8s/`

Stopping Minikube

To stop Minikube, and the VM that it runs, run `minikube stop` . You can bring your local cluster back online at any time by running `minikube start`

To fully delete the cluster, run `minikube delete`

Stopping Running Containers

You might still have some containers running on your machine. Try a `docker ps` . You can then run `docker stop <container_id>` to clean up any running containers

Clearing the Build Cache

All the images that we built and ran during the course are cached on your local machine - they might be taking up to around 1GB of space. You can clean these up by running `docker system prune`

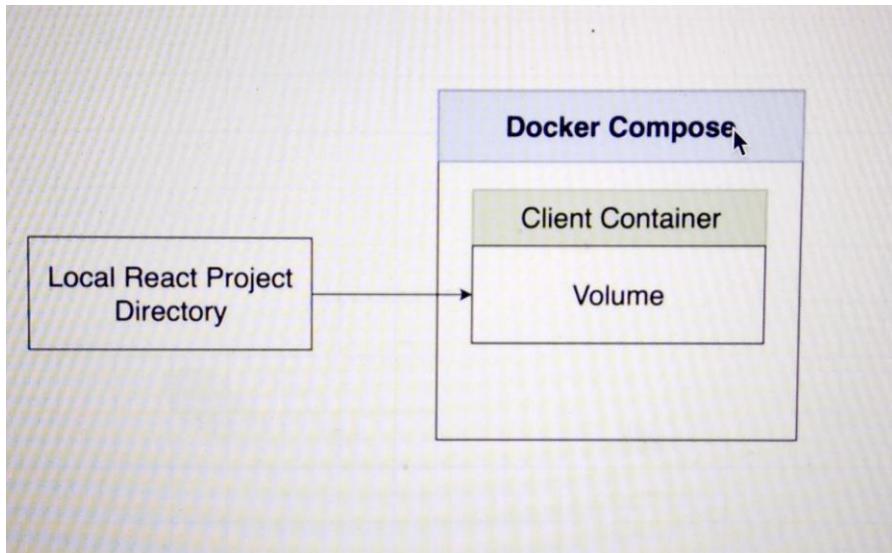
```
/Users/naveen.kumar1/.zshrc:68: no matches found: load?
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl delete -f k8s/
service "client-cluster-ip-service" deleted
deployment.apps "client-deployment" deleted
persistentvolumeclaim "database-persistent-volume-claim" deleted
ingress.networking.k8s.io "ingress-service" deleted
service "postgres-cluster-ip-service" deleted
deployment.apps "postgres-deployment" deleted
service "redis-cluster-ip-service" deleted
deployment.apps "redis-deployment" deleted
service "server-cluster-ip-service" deleted
deployment.apps "server-deployment" deleted
deployment.apps "worker-deployment" deleted
unable to recognize "k8s/certificate.yaml": no matches for kind "Certificate" in version "cert-manager.io/v1"
unable to recognize "k8s/issuer.yaml": no matches for kind "ClusterIssuer" in version "cert-manager.io/v1"
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ docker system prune
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
503f30532c3fb1adb01aaeaa4062d3be01cfcbcb413e14b44c3757dd25e6e38
adfc18b8e12cf56e89e9f44af322df5c862e125777c1a89c2524f9953b6c5aa
5fba0ac5035ad6ef00e105e58d5a7fb853137dd3f074446a72ff2e0f1ee1b272
854e0a8f351a6023bd6de187ee4e0d4cb8415d2f62be999327aaa454a8597d38
daa1b2f934430af6cd3c40baf63f3ade73ac0f1c36b2c35d1a32a703774e5c4b
f6021788dfb373c465b1ad01987549e9ce28c7e9e9a72eac9a97be439189b7f8
25003ed1cfb5afe75f7950435aecb9fe03cd501340541914380577ecf7bc91bc
c248a8af4be576b68cef1162fc308d9f4284a1dbbdd2367da783393536f472da
4e7acae273c9fe7af9efa9b12868a9894aeb2f83aaacf82ecdee3d311d94ff03
fe48a45fc7b8f85f0c3def1f5345adec23636983992329727725dcddb34a7
113b026b3df082c49b11c3d54341e0810ab969425b9503faa6ec8451e50e6987
ea8c71d5fe7107ed47eb5b33d14995545c5e078c5a50e2163d96bff094a9a3e7
430c1d8568c4d2af61171dcddd2638d5b6c7bcea87b16fc19ed078950ebbb850
f9f1b41a9bcfa1f07adfb554d6e7ec0fcf1b04b7708716d70ebbe4c763e233bc
ea041bf784a259c8319baa127bfcae2a9f321a916e8c2622e4b7418089365544
ddacf7d192e8cdb044213c216f35f0918acfecb53c3be19cbf80ba93ef5a0a79
c5d3c6ed1e2036ce7bcce81f0aec0039c347f36b711ed665d8a683dd1dcaf523
586909d422134ea883b7786ad4792d59311ffc0208548f95c8118605e5e8573d
509c88be88215f14b05bfa9ab292cf2bf1386219928a3fe6e52994a5a0277a74
78db62e76109a39d4e380411a16510dfbcdab860e5aa27e08d114b3884d204f

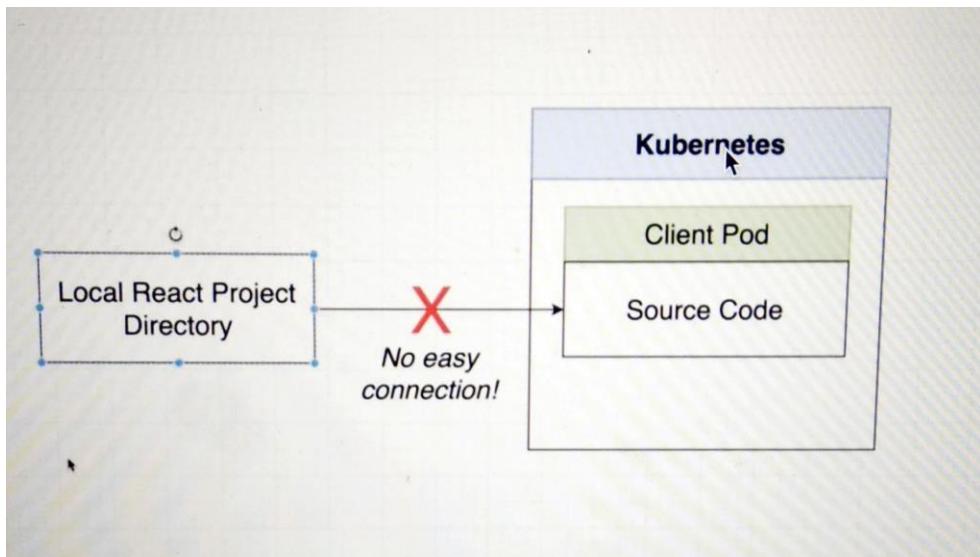
Deleted Images:
deleted: sha256:8618b223fb37f964ed78d2fa29a044b404d9d6009d17387634147445125b799e
deleted: sha256:b87800a6d0d1d07ced7d11f90a82a234cdb5715392404b8bb61e325e153a06d8
deleted: sha256:07b5158828ba0f137f0344e23cc1a7565df43164f4ec853db0298da0ee817df3
deleted: sha256:d852c578c64a1ac74c66c7e1159acb1a43aef0e94a5595fb20cc8b5ce1921232
deleted: sha256:f4576edb4e31506b600e94d5f6ba5495b7b29a91abd82ce58cd5072da111df41
deleted: sha256:f3682d8b6bba3b83e6c8ef3ecdf286f38dc73b67f17191318f977e1f0c3de374
deleted: sha256:a425ab92770316f1f83826993e3130319a84a4c90792f03344552d4a6c65fce8
deleted: sha256:7a3f59dd43d40d2592fb5d6469ab4819bfd5cc324a558fea5b0cc1fe2db6436
deleted: sha256:7bb989c513ca0e3029dd767fb10198261a2b4a4c114c5a3b0a2e68fd0cfad0d1
deleted: sha256:9dadae6f286e2295c5062383c7f4246d320f92dd44c533a97ff74522f3435fae
untagged: k8s.gcr.io/ingress-nginx/kube-webhook-certgen@sha256:64d8c73dca984af206adf9d
6d7e46aa550362b1d7a01f3a0a91b20cc67868660
deleted: sha256:c41e9fcadef5a291120de706b7dfa1af598b9f2ed5138b6dc9f79a68aad0ef4c
```

Local Development with Skaffold

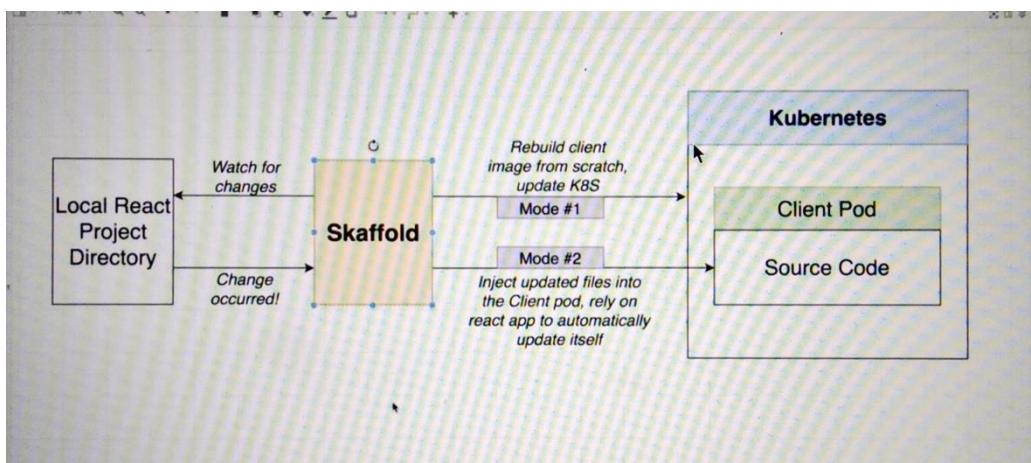
Local Development



With docker compose we have an option to see our local changes instantly in deployment output.



But with Kubernetes, we don't have the direct option. That's where Skaffold comes into play.



Installing Skaffold

```
/Users/naveen.Kumar1/.zshrc:68: no matches found: Load?
→ ~ brew install skaffold
Running `brew update --preinstall` ...


^Skaffold 1.32.0 is already installed but outdated (so it will be upgraded).
=> Downloading https://ghcr.io/v2/homebrew/core/skaffold/manifests/1.38.0
#####
 100.0%
=> Downloading https://ghcr.io/v2/homebrew/core/skaffold/blobs/sha256:86b4b0cf661910e5f405c55d5d6272a3a5bc5d5204d
=> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:86b4b0cf661910e5f405c55d5d627
#####
 100.0%
=> Upgrading skaffold
 1.32.0 -> 1.38.0

=> Pouring skaffold--1.38.0.big_sur.bottle.tar.gz
=> Caveats
zsh completions have been installed to:
 /usr/local/share/zsh/site-functions
=> Summary
🍺 /usr/local/Cellar/skaffold/1.38.0: 8 files, 88.3MB
=> `brew cleanup` has not been run in the last 30 days, running now...
```

Skaffold Config File

```
skaffold.yaml > {} build > [ ] artifacts > {} 1 > {} sync > [ ] manual  
  skaffold.yaml (v2beta28.json)  
1   apiVersion: skaffold/v2beta12  
2   kind: Config  
3   deploy:  
4     kubectl:  
5       manifests:  
6         - ./k8s/*  
7   build:  
8     local:  
9       push: false  
10    artifacts:  
11      - image: cygnetops/client-skaffold  
12        context: client  
13        docker:  
14          dockerfile: Dockerfile.dev  
15        sync:  
16          manual:  
17            - src: "src/**/*.js"  
18              dest: .  
19            - src: "src/**/*.css"  
20              dest: .  
21            - src: "src/**/*.html"  
22              dest: .  
23      - image: cygnetops/worker-skaffold  
24        context: worker  
25        docker:  
26          dockerfile: Dockerfile.dev  
27        sync:  
28          manual:  
29            - src: "*.js"  
30              dest: .  
31      - image: cygnetops/server-skaffold  
32        context: server  
33        docker:  
34          dockerfile: Dockerfile.dev  
35        sync:  
36          manual:  
37            - src: "*.js"  
38              dest: .  
39
```

Running the Skaffold

```
total reclaimed space: 3.77750B
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ skaffold d
ev
Listing files to watch...
- cynetops/client-skaffold
- cynetops/worker-skaffold
- cynetops/server-skaffold
Generating tags...
- cynetops/client-skaffold --> cynetops/client-skaffold:448cc43-dirty
- cynetops/worker-skaffold --> cynetops/worker-skaffold:448cc43
- cynetops/server-skaffold --> cynetops/server-skaffold:448cc43
Checking cache...
- cynetops/client-skaffold: Not found. Building
- cynetops/worker-skaffold: Not found. Building
- cynetops/server-skaffold: Not found. Building
Starting build...
Found [docker-desktop] context, using local docker daemon.
Building [cyne.../server-skaffold]...
Target platforms: [linux/amd64]
[+] Building 6.3s (8/10)
=> [internal] load build definition from Dockerfile.dev          0.0s
=> => transferring dockerfile: 242B                            0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                0.0s
=> [internal] load metadata for docker.io/library/node:14.14.0-alpine 3.5s
=> [auth] library/node:pull token for registry-1.docker.io      0.0s
=> [1/5] FROM docker.io/library/node:14.14.0-alpine@sha256:97c4cddbbf97299f5eda09 0.0s
[+] Building 6.4s (8/10)
=> => transferring context: 3.05kB                           0.0s
=> CACHED [2/5] WORKDIR /app                                0.0s
=> [3/5] COPY ./package.json ./                               0.0s
=> [4/5] RUN npm install                                     2.8s
```

Live Sync Changes

```
skaffold.yaml > {} build > [ ] artifacts > {} 0 > {} sync > [ ] manual > {} 2
14           dest: .
15           - src: "src/**/*.css"
16           dest: .
17           - src: "src/**/*.html"
18           dest: .
19           - image: cyne.../worker-skaffold
20             context: worker
21             docker:
22               dockerfile: Dockerfile.dev
23             sync:
24               manual:
25                 - src: "*.js"
26                 dest: .
27             - image: cyne.../server-skaffold
28               context: server
29               docker:
30                 dockerfile: Dockerfile.dev
31               sync:
32                 manual:
33                   - src: "*.js"
34                   dest: .
35   deploy:
36     kubectl:
37       manifests:
38         - k8s/client-deployment.yaml
39
```

```
34 |           dest: .
35 | deploy:
36 |   kubectl:
37 |     manifests:
38 |       - k8s/client-deployment.yaml
39 |       - k8s/server-deployment.yaml
40 |       - k8s/worker-deployment.yaml
41 |       - k8s/server-cluster-ip-service.yaml
42 |       - k8s/client-cluster-ip-service.yaml
43 | }
```

PROBLEMS OUTPUT TERMINAL

✗ TERMINAL

```
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
client-deployment-79678d66b7-kq8qs  1/1     Running   0          42s
client-deployment-79678d66b7-r4xln  1/1     Running   0          30s
client-deployment-79678d66b7-zxhqd  1/1     Running   0          58s
server-deployment-64c8997f95-kf8jd  1/1     Running   0          58s
server-deployment-64c8997f95-qz28k  1/1     Running   0          58s
server-deployment-64c8997f95-w7cg9  1/1     Running   0          58s
worker-deployment-7c9f59d99d-b45rv  1/1     Running   0          58s
→ KUBERNATES-MULTICONTAINER-APP git:(main) ✘
```