Docker Birmingham July 2020

Agenda (Rough)

- Remote ‘n’ Greet
- Matt Todd - "Dr. StrangeCloud or: How I Learned to Stop Worrying and Love the IDE"
- Docker Birmingham - Where Next?

https://github.com/docker-birmingham/policies

Next Meetup - Aug 5th 2020

Thanks to our Sponsor!

BlackCat /

Want to speak

(Don't be afraid to ask for help!)

Topic Ideas?

(Don't be afraid to ask for what you want!)

Formats?

(Workshops, hacks, 99s, lean circle)

Introductions

- Preferred Name!
- Why you’re here!
- If you could choose only one place to go on holiday for the rest of your life, where would it be and why?

"Dr. StrangeCloud or: How I Learned to Stop Worrying and Love the IDE"

On IDEs

Is this an IDE?

IDE’s are great for……

- Clear code display and workspace management
- Easy access to compile & build tools
- Code completion and introspection
- Access to documentation & libraries
- Handling large refactoring tasks
- Linting & Code quality analysis

Anything I’ve missed - what’s great about IDEs?

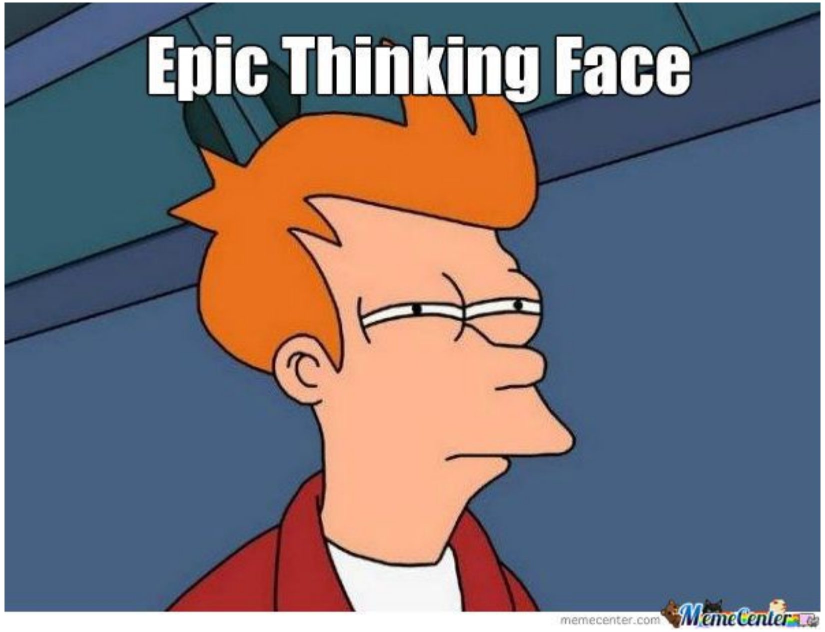
IDEs are not so great as they.....

- Are greedy with resources
- Rely on external tooling and apps to integrate with (versioning issues)
- No very portable between hosts
- Generally needs to run as a desktop UI (atom or similar)
- Not very configurable & closed source

Sounds Familiar?

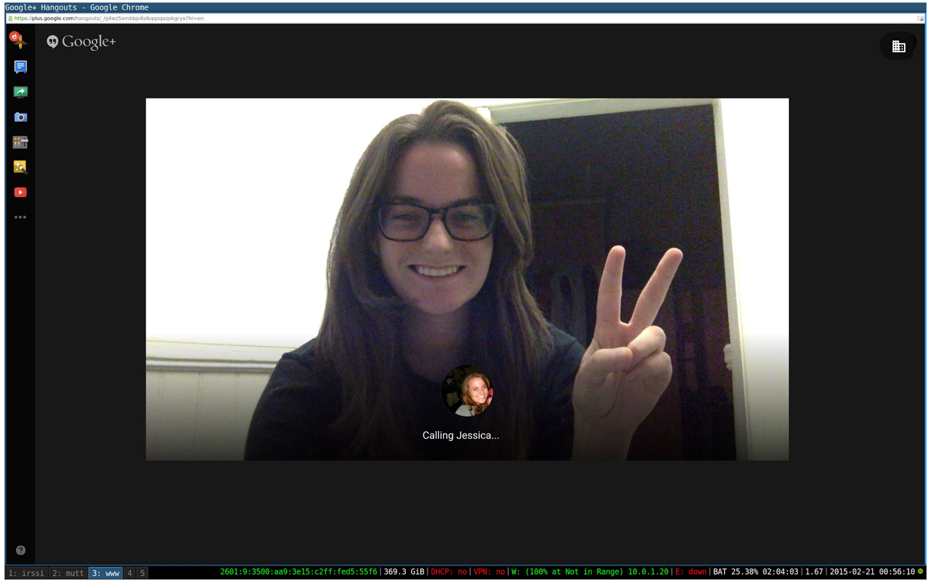
- What if the desktop IDE could be containerised?
- It would gain all the goodness of containerisation
- But what about that pesky Desktop

Containerise the IDE, Great……. Errrrr how?



Waaaaaay back when (2015)

https://blog.jessfraz.com/post/docker-containers-on-the-desktop/


Google+ Hangouts - Google Chrome
https://plus.google.com/hangouts/_/g4wz5xmbbjo6slkqqsqaqskgrya?hl=en
Google+
Calling Jessica...
1: irssi 2: mutt 3: www 4 5
2601:9:3500:aa9:3e15:c2ff:fed5:55f6 | 369.3 GiB | DHCP: no | VPN: no | W: (100% at Not in Range) 10.0.1.20 | E: down | BAT 25.38% 02:04:03 | 1.67 | 2015-02-21 00:56:10

Dockerised Spotify?

```
$ docker run -it \

    -v /tmp/.X11-unix:/tmp/.X11-unix \ # mount the X11 socket

    -e DISPLAY=unix$DISPLAY \ # pass the display

    --device /dev/snd \ # sound

    --name spotify \

    jess/spotify
```

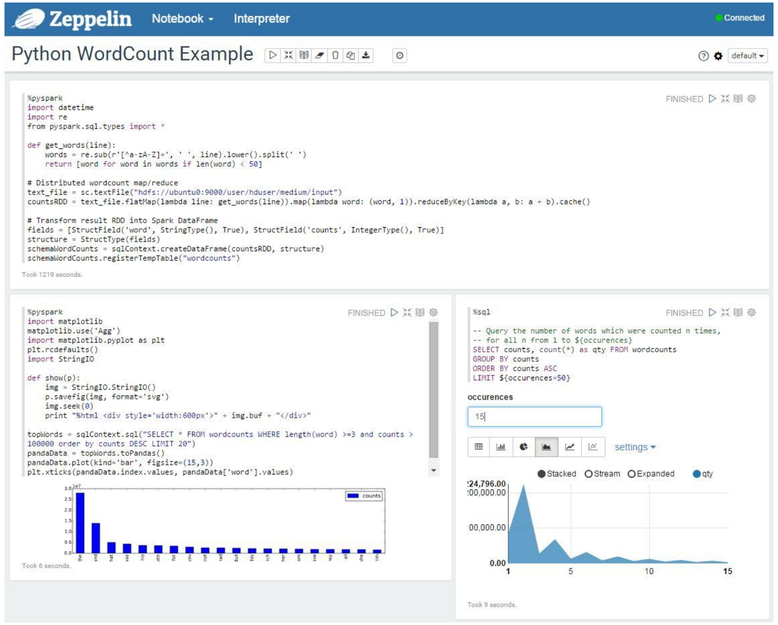
It sort of works.....

- X11 volume and port forwarding - Yuk!
- Device mapping to the containers......
- Terrifyingly bad UX experience
- Lack of consistent support for OSes (OSx, won’t work)

The rise of the web-based code editor

- Browsers capability dramatically increased in recent years
- Enabled sophisticated code editing without a desktop UI
- Games changer in terms of delivery

First Wave (Data Science)



Python WordCount Example

```
%pyspark
import datetime
import re
from pyspark.sql.types import *

def get_words(line):
    words = re.sub(r'[^a-zA-Z]+', ' ', line).lower().split(' ')
    return [word for word in words if len(word) < 50]

# Distributed wordcount map/reduce
text_file = sc.textFile("hdfs://ubuntu0:9000/user/hduser/medium/input")
countsRDD = text_file.flatMap(lambda line: get_words(line)).map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b).cache()

# Transform result RDD into Spark DataFrame
fields = [StructField('word', StringType(), True), StructField('counts', IntegerType(), True)]
structure = StructType(fields)
schemaWordCounts = sqlContext.createDataFrame(countsRDD, structure)
schemaWordCounts.registerTempTable("wordcounts")
```

Took 1219 seconds.

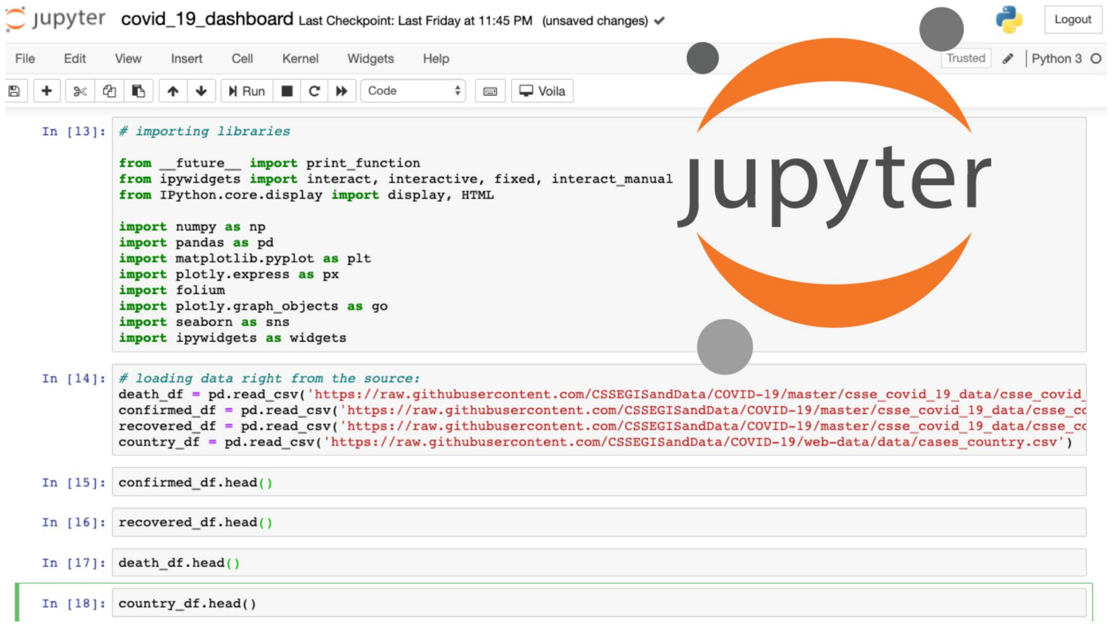
```
%pyspark
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
plt.rcdefaults()
import StringIO

def show(p):
    img = StringIO.StringIO()
    p.savefig(img, format='svg')
    img.seek(0)
    print "%html <div style='width:600px'>" + img.buf + "</div>"

topWords = sqlContext.sql("SELECT * FROM wordcounts WHERE length(word) >=3 and counts >
100000 order by counts DESC LIMIT 20")
pandaData = topWords.toPandas()
pandaData.plot(kind='bar', figsize=(15,3))
plt.xticks(pandaData.index.values, pandaData['word'].values)
```

Took 6 seconds.

```
%sql

-- Query the number of words which were counted n times,
-- for all n from 1 to ${occurences}
SELECT counts, count(*) as qty FROM wordcounts
GROUP BY counts
ORDER BY counts ASC
LIMIT ${occurences=50}
```

occurences: 15

Took 9 seconds.



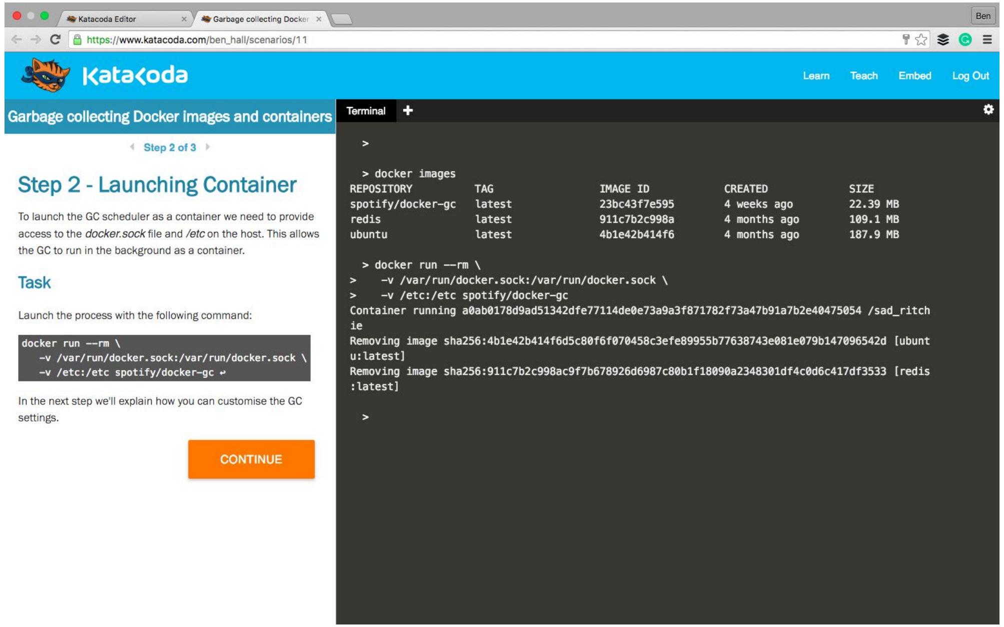
covid_19_dashboard Last Checkpoint: Last Friday at 11:45 PM (unsaved changes)

```
In [13]: # importing libraries

         from __future__ import print_function
         from ipywidgets import interact, interactive, fixed, interact_manual
         from IPython.core.display import display, HTML

         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         import plotly.express as px
         import folium
         import plotly.graph_objects as go
         import seaborn as sns
         import ipywidgets as widgets

In [14]: # loading data right from the source:
         death_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid
         confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_co
         recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_co
         country_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/web-data/data/cases_country.csv')

In [15]: confirmed_df.head()

In [16]: recovered_df.head()

In [17]: death_df.head()

In [18]: country_df.head()
```

First Wave

- Browser based REPL interpreters
- Easily containerised (it’s just a web-app after-all)
- Can build up a data and code toolbox which is easily shared and reproduced

Then came “SAAS” Interactive Environments


Katacoda Editor
Garbage collecting Docker
Ben
https://www.katacoda.com/ben_hall/scenarios/11
Katacoda
Learn
Teach
Embed
Log Out
Terminal

Garbage collecting Docker images and containers

Step 2 of 3

Step 2 - Launching Container

To launch the GC scheduler as a container we need to provide access to the *docker.sock* file and */etc* on the host. This allows the GC to run in the background as a container.

Task

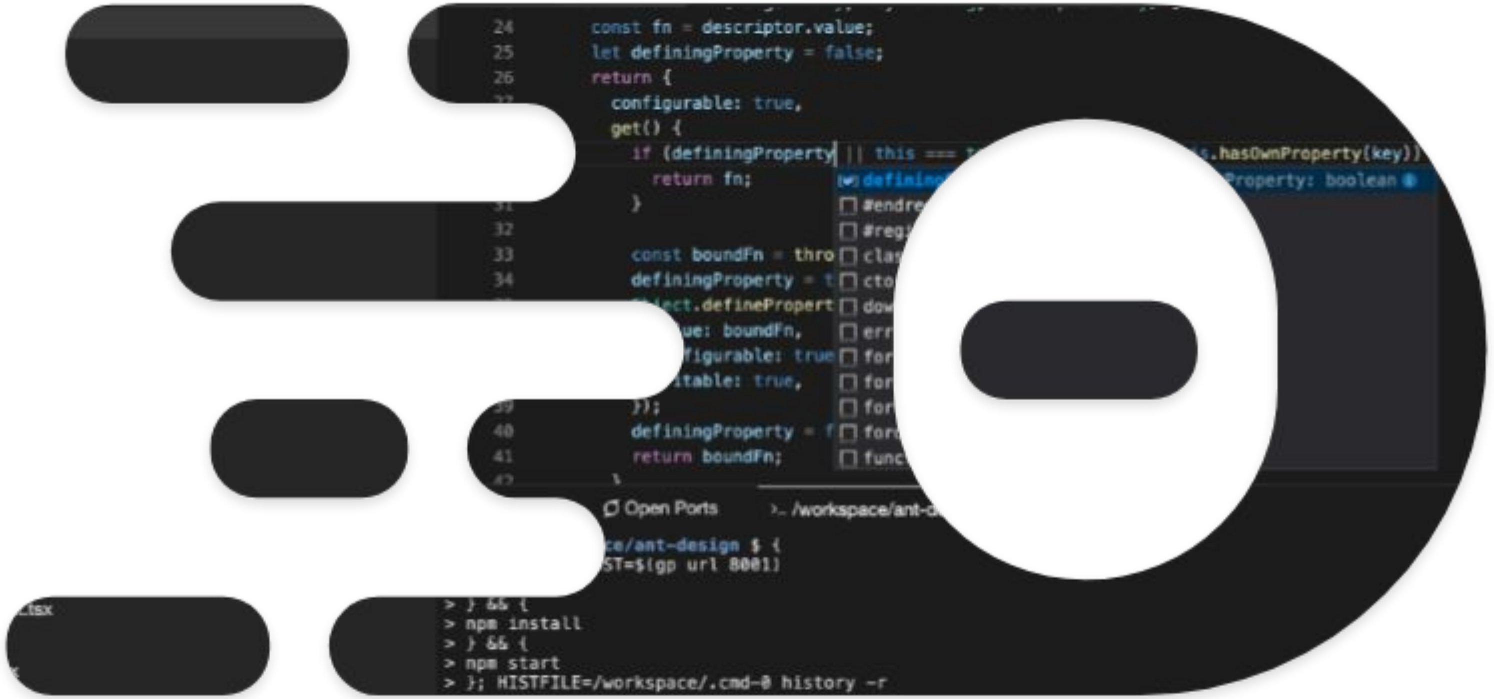
Launch the process with the following command:

```
docker run --rm \
   -v /var/run/docker.sock:/var/run/docker.sock \
   -v /etc:/etc spotify/docker-gc ↵
```

In the next step we'll explain how you can customise the GC settings.

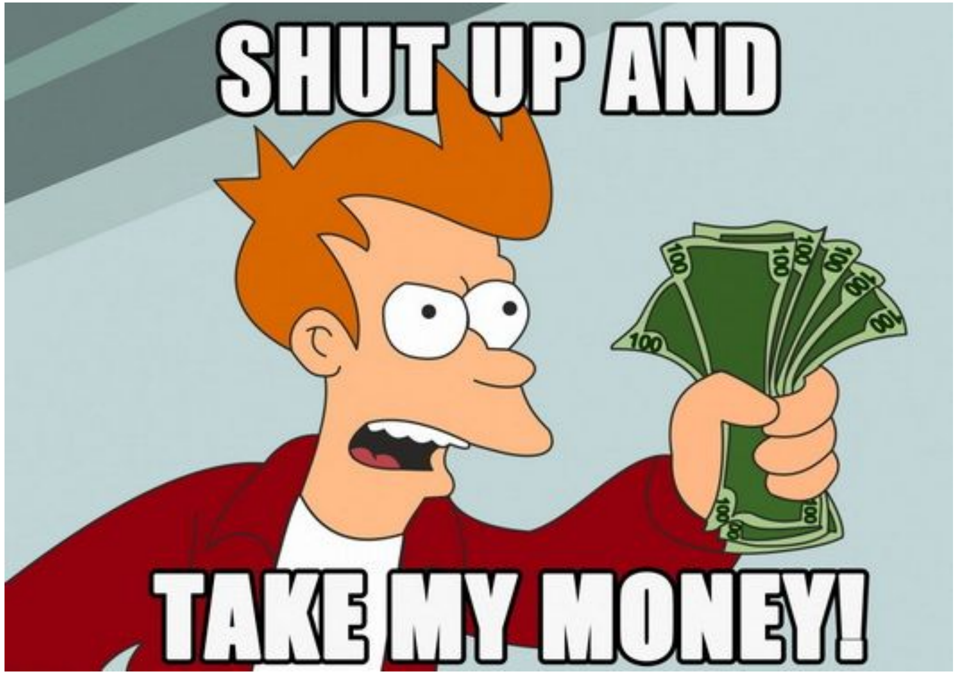
CONTINUE

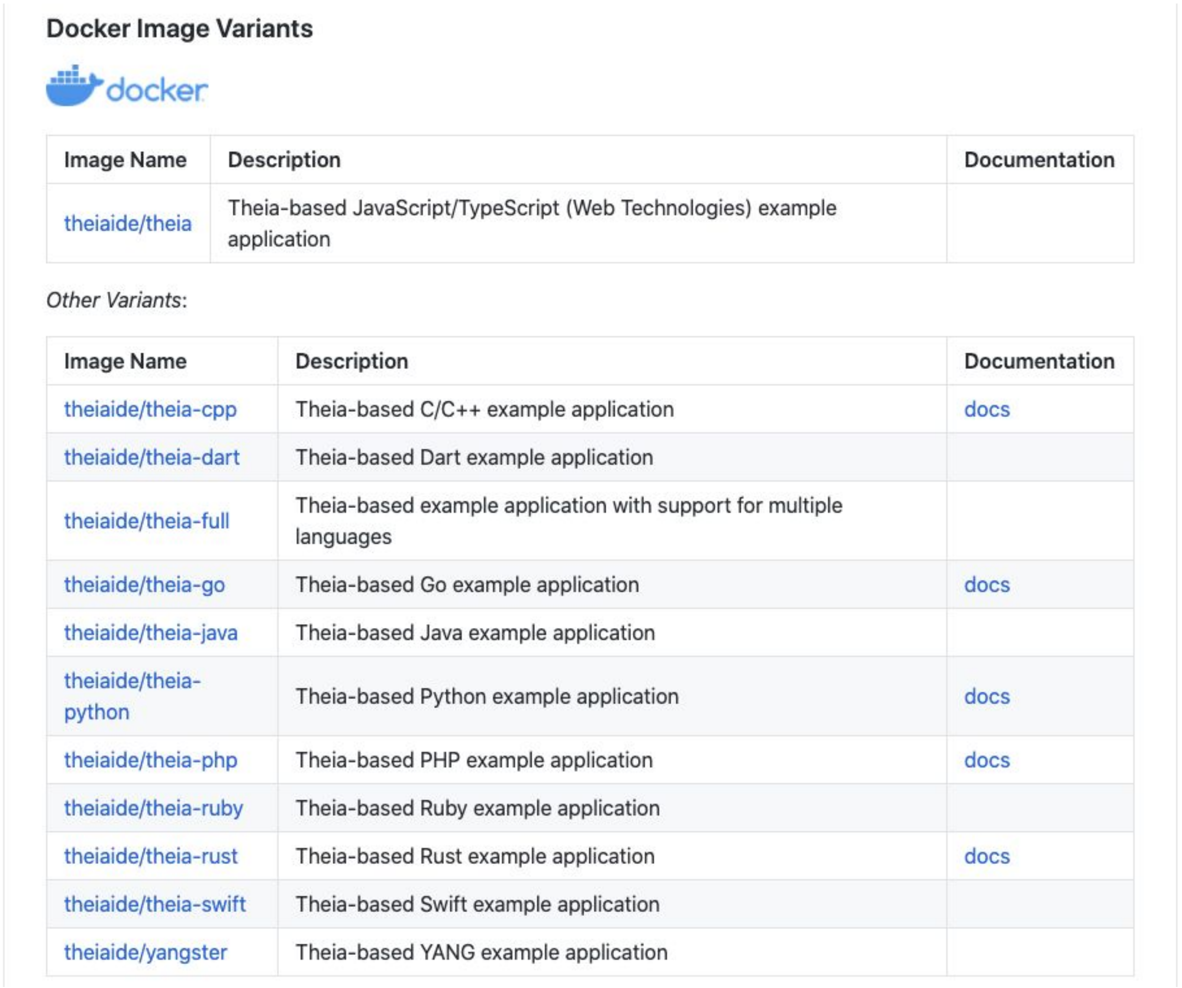
```
  >

  > docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
spotify/docker-gc   latest              23bc43f7e595        4 weeks ago         22.39 MB
redis               latest              911c7b2c998a        4 months ago        109.1 MB
ubuntu              latest              4b1e42b414f6        4 months ago        187.9 MB

  > docker run --rm \
>    -v /var/run/docker.sock:/var/run/docker.sock \
>    -v /etc:/etc spotify/docker-gc
Container running a0ab0178d9ad51342dfe77114de0e73a9a3f871782f73a47b91a7b2e40475054 /sad_ritch
ie
Removing image sha256:4b1e42b414f6d5c80f6f070458c3efe89955b77638743e081e079b147096542d [ubunt
u:latest]
Removing image sha256:911c7b2c998ac9f7b678926d6987c80b1f18090a2348301df4c0d6c417df3533 [redis
:latest]

  >
```



https://theia-ide.org/
https://github.com/eclipse-theia/theia

Dockerised Browser based UI you say?





Docker Image Variants

docker

Image Name	Description	Documentation
theiaide/theia	Theia-based JavaScript/TypeScript (Web Technologies) example application	

Other Variants:

Image Name	Description	Documentation
theiaide/theia-cpp	Theia-based C/C++ example application	docs
theiaide/theia-dart	Theia-based Dart example application	
theiaide/theia-full	Theia-based example application with support for multiple languages	
theiaide/theia-go	Theia-based Go example application	docs
theiaide/theia-java	Theia-based Java example application	
theiaide/theia-python	Theia-based Python example application	docs
theiaide/theia-php	Theia-based PHP example application	docs
theiaide/theia-ruby	Theia-based Ruby example application	
theiaide/theia-rust	Theia-based Rust example application	docs
theiaide/theia-swift	Theia-based Swift example application	
theiaide/yangster	Theia-based YANG example application	

Great, time for a spin.....

```
docker run -it \

  --init \

  -p 3000:3000 \

  -v "$(pwd):/home/project:cached" \

  theiaide/theia-java:next
```

Generate a (very, very old spring-boot app)

```
mvn archetype:generate -DarchetypeGroupId=org.springframework.boot
-DarchetypeArtifactId=spring-boot-sample-simple-archetype -DarchetypeVersion=1.0.2.RELEASE
-DartifactId=my-app -DgroupId=com.example -Dversion=0.1.0-SNAPSHOT -DinteractiveMode=false
```

Create a new react app…

```
$ npx create-react-app my-react

$ cd my-react

$ yarn start  (not this will conflict with theia port on 3000)
```

What’s so great about this?

- Pretty good IDE experience including auto-complete for languages built-in
- Build your own IDE and consistently versioned tooling
- Support multiple languages as you need
- Port this between any number of Docker Hosts
- Leverage Cloud compute such as GPUs and high spec machines
- Consistent enterprise or squad dev environments (cloud or local)

What’s not so great

- Slightly quirky IDE experience
- Some mental gymnastics needed to understand where you files may be locate
- Some performance issues when using code-completion

Thanks!,
Questions!

Docker Birmingham - Where Next?

Survey URI

https://forms.gle/2EPWyJo5pVMvXCM29