

```
1  #!/usr/bin/env python
2
3  import sys
4  import requests
5  from bs4 import BeautifulSoup
6  from urllib.parse import urljoin
7
8  def extract_links(url):
9      res = requests.get(url)
10     soup = BeautifulSoup(res.text, "html.parser")
11     base = url
12     # TODO: Update base if a <base> element is present with the href attribute
13     links = []
14     for link in soup.find_all("a"):
15         links.append({
16             "text": " ".join(link.text.split()) or "[IMG]",
17             "href": urljoin(base, link.get("href"))
18         })
19     return links
20
21 if __name__ == "__main__":
22     if len(sys.argv) != 2:
23         print("\nUsage: \n\t{} <URL>\n".format(sys.argv[0]))
24         sys.exit(1)
25     for link in extract_links(sys.argv[-1]):
26         print("{}({})".format(link["text"], link["href"]))
27
```

Modular
!

Highly
informative!

Usage
hints!

```
$ docker build -t linkextractor:v2 .
```

```
$ docker run --rm linkextractor:v2 https://docker.com
```

```
[[IMG]](https://docker.com/dockercon)
```

```
[Watch live >](https://docker.com/dockercon/register-livestream)
```

```
[[IMG]](https://docker.com/)
```

```
[Why Docker?](https://docker.com/why-docker)
```

```
[What is a Container?](https://docker.com/resources/what-container)
```

```
[Company](https://docker.com/company)
```

```
[Partners](https://docker.com/partners)
```

```
[Products](https://docker.com/products)
```

```
[Docker Enterprise](https://docker.com/products/docker-enterprise)
```

```
[Docker Hub](https://docker.com/products/docker-hub)
```

```
[Docker Community](https://docker.com/products/docker-community)
```



```
$ docker image ls linkextractor
```

```
linkextractor
```

```
v2
```

```
linkextractor
```

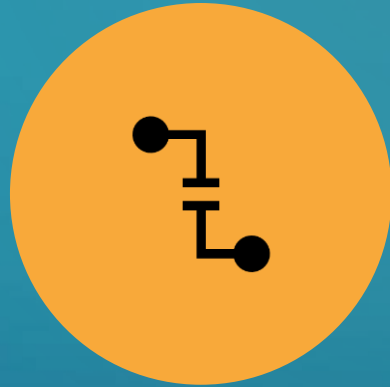
```
v1.1
```

```
linkextractor
```

```
v1
```



LINKEXTRACTOR ROADMAP



TURN IT INTO AN API SERVICE
INSTEAD OF A ONE-OFF
COMMAND



PROFIT!

1. "MAGICALLY" CODE MAIN.PY

```
$ cp cheats/main.py .
```

linkextractor.py

Dockerfile

main.py

×

```
1  #!/usr/bin/env python
2
3  from flask import Flask
4  from flask import request
5  from flask import jsonify
6  from linkextractor import extract_links
7
8  app = Flask(__name__)
9
10 @app.route("/")
11 def index():
12     return "Usage: http://<hostname>[:<prt>]/api/<url>"
13
14 @app.route("/api/<path:url>")
15 def api(url):
16     qs = request.query_string.decode("utf-8")
17     if qs != "":
18         url += "?" + qs
19     links = extract_links(url)
20     return jsonify(links)
21
22 app.run(host="0.0.0.0")
23
```

1. "MAGICALLY" CODE MAIN.PY

```
$ cp cheats/main.py .
```

linkextractor.py

Dockerfile

main.py



```
1  #!/usr/bin/env python
2
3  from flask import Flask
4  from flask import request
5  from flask import jsonify
6  from linkextractor import extract_links
7
8  app = Flask(__name__)
9
10 @app.route("/")
11 def index():
12     return "Usage: http://<hostname>[:<prt>]/api/<url>"
13
14 @app.route("/api/<path:url>")
15 def api(url):
16     qs = request.query_string.decode("utf-8")
17     if qs != "":
18         url += "?" + qs
19     links = extract_links(url)
20     return jsonify(links)
21
22 app.run(host="0.0.0.0")
23
```

That's
our
baby!

"MAGICALLY" CODE MAIN.PY

```
$ cp cheats/main.py .
```



"Go ahead
caller, I'm
listening."

linkextractor.py

Dockerfile

main.py

×

```
1  #!/usr/bin/env python
2
3  from flask import Flask
4  from flask import request
5  from flask import jsonify
6  from linkextractor import extract_links
7
8  app = Flask(__name__)
9
10 @app.route("/")
11 def index():
12     return "Usage: http://<hostname>[:<prt>]/api/<url>"
13
14 @app.route("/api/<path:url>")
15 def api(url):
16     qs = request.query_string.decode("utf-8")
17     if qs != "":
18         url += "?" + qs
19     links = extract_links(url)
20     return jsonify(links)
21
22 app.run(host="0.0.0.0")
23
```

That's
our
baby!

OPTIMIZE OUR DOCKERFILE

```
FROM python:3
```

```
LABEL maintainer="<your name>"
```

```
RUN pip install beautifulsoup4
```

```
RUN pip install requests
```

```
WORKDIR /app
```

```
COPY linkextractor.py /app/
```

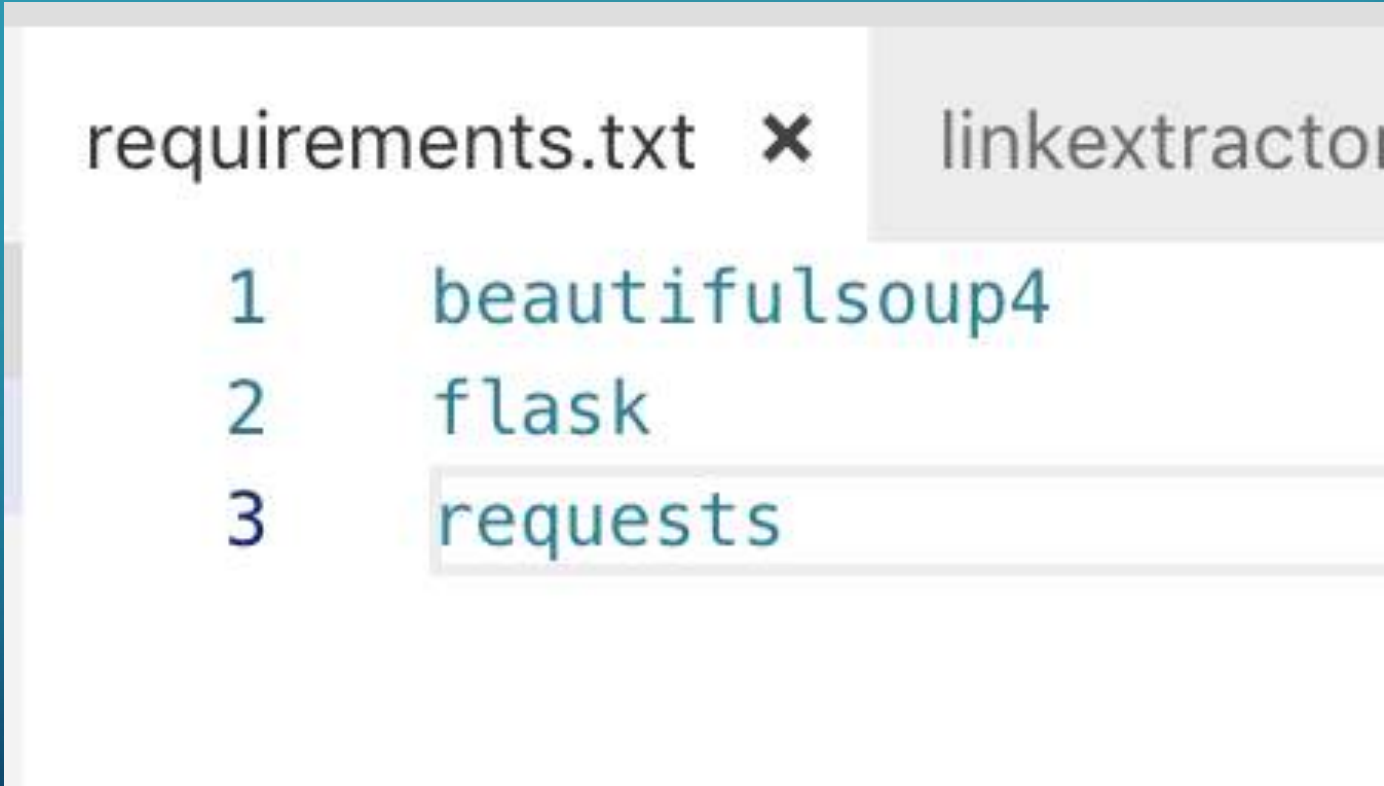
```
RUN chmod a+x linkextractor.py
```

```
ENTRYPOINT ["../linkextractor.py"]
```

} Eliminate extra
manual install steps

- Reduce layers
- Maintain dependencies the way we normally do in our language of choice

2. PYTHON REQUIREMENTS FILE



The image shows a code editor window with a tab labeled 'requirements.txt' and a close button 'X'. The editor contains three lines of code, each preceded by a line number (1, 2, 3). The dependencies listed are BeautifulSoup4, Flask, and Requests. The 'Requests' line is currently selected with a mouse cursor.

```
1 BeautifulSoup4
2 flask
3 requests
```

OPTIMIZE OUR DOCKERFILE

```
FROM python:3
```

```
LABEL maintainer="<your name>"
```

```
RUN pip install beautifulsoup4
```

```
RUN pip install requests
```


```
WORKDIR /app
```

```
COPY linkextractor.py /app/
```

```
RUN chmod a+x linkextractor.py
```

```
ENTRYPOINT ["./linkextractor.py"]
```


Copy all our code in
at once



OPTIMIZE OUR DOCKERFILE

```
FROM python:3  
LABEL maintainer="<your name>"  
  
RUN pip install beautifulsoup4  
RUN pip install requests  
  
WORKDIR /app  
COPY linkextractor.py /app/  
RUN chmod a+x linkextractor.py  
  
ENTRYPOINT ["../linkextractor.py"]
```

We no longer want
a single, one-off
command...we want
a service



OPTIMIZE OUR DOCKERFILE

```
FROM python:3
```

```
LABEL maintainer="<your name>"
```

```
WORKDIR /app
```

```
COPY requirements.txt /app/
```

```
RUN pip install -r requirements.txt
```

```
COPY *.py /app/
```

```
RUN chmod a+x *.py
```

```
CMD [ "./main.py" ]
```

Note the switch
from
ENTRYPOINT to
CMD

- ENTRYPOINT - a command to run every time
 - good for executables
- CMD - a *default* command for our container to run but it can be substituted

LINKEXTRACTOR V3 - API-IFIED

```
$ docker build -t linkextractor:v3 .  
  
$ docker run -d -p 5000:5000 --name=linkextractor  
linkextractor:v3  
  
$ curl localhost:5000  
Usage: http://<hostname>[:<prt>]/api/<url>  
  
$ curl localhost:5000/api/http://docker.com  
[{"href":"http://docker.com/dockercon","text":"[IMG]"}, {"href":"http://docker.com/dockercon/register-livestream","text":"Watch live  
>"}, {"href":"http://docker.com/","text":"[IMG]"}, {"href":"http://docker.com/why-docker","text":"Why Docker?"}, {"href":"http://docker.com/resources/what-container","text":"What is a
```

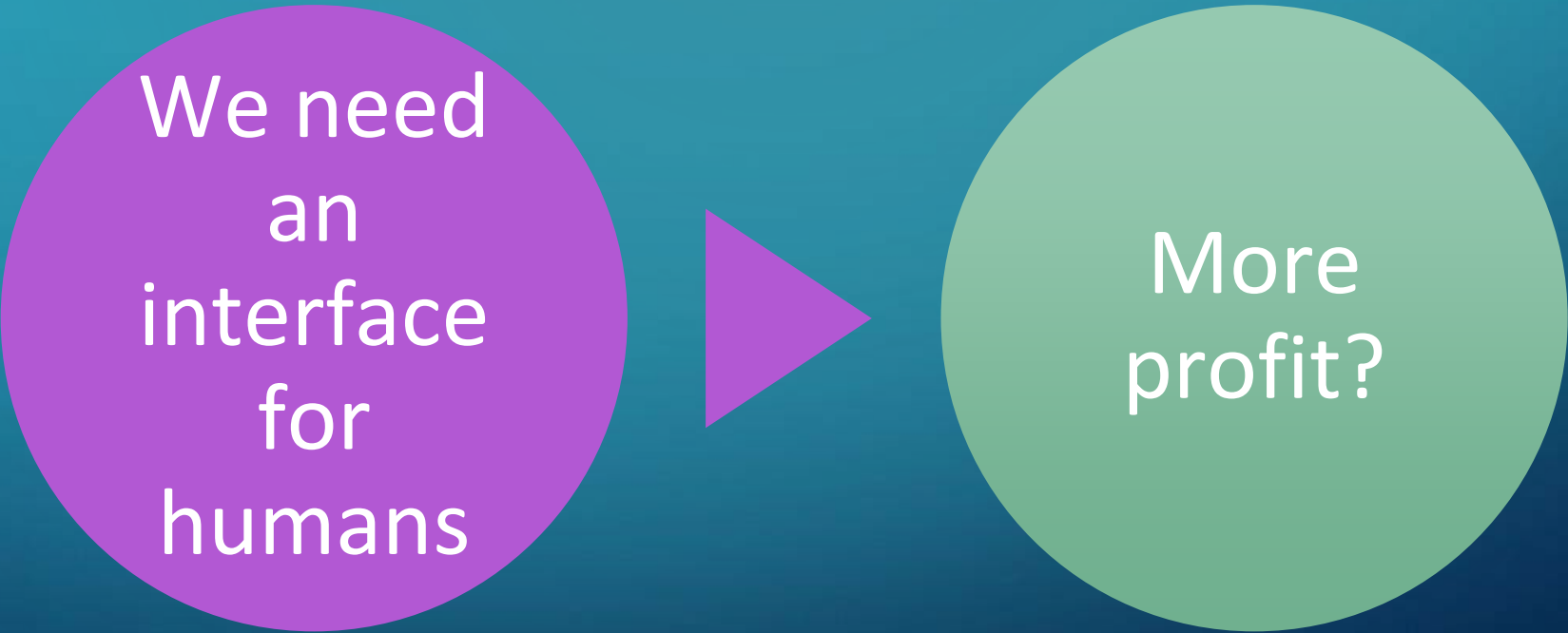
DUDE, WHERE ARE MY LOGS?

```
$ docker container logs linkextractor
* Serving Flask app "main" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production
environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
172.17.0.1 - - [01/May/2019 09:14:23] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [01/May/2019 09:14:37] "GET
/api/http://example.com HTTP/1.1" 200 -
172.17.0.1 - - [01/May/2019 09:16:33] "GET /api/http://docker.com
HTTP/1.1" 200 -
```


The background is a solid teal color. In the corners, there are decorative white line art elements resembling circuit boards or neural network connections. These include straight lines, right-angle turns, and small circles at the ends of the lines.

```
$ docker container rm -f linkextractor
```

NEW ROADMAP!



```
graph LR; A((We need an interface for humans)) --> B((More profit?))
```

We need
an
interface
for
humans

More
profit?

A decorative graphic consisting of blue circuit-like lines with small circles at the ends, extending horizontally from the left and right sides of the central text area.

TIME WARP

```
$ git checkout step4 -f
```