

# 7

## Build Image

Run on project directory

```
docker image build -t env-api .
```

Run image

```
docker run -p 80:80 env-api
```

## List Environment Variable

Code

```
@api.route('/list-env', methods = ['GET'])
def list_env():

    content_type = request.headers.get('Content-Type')

    if (content_type == 'application/json'):
        json_response= []

        for env in os.environ:
            json_response.append({
                "Env variable": env,
                "Env value": os.getenv(env),
            })
    else:

        json_response = {
            "success": "false",
            "message": "content-Type not supported"
        }

    return json.dumps(json_response)
```

## Request

```
GET /list-env
```

## CURL example

```
curl -XGET http://localhost/list-env | jq
```

```
[
  {
    "Env variable": "PATH",
    "Env value": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  },
  {
    "Env variable": "HOSTNAME",
    "Env value": "4f74b5a8104c"
  },
  {
    "Env variable": "LANG",
    "Env value": "C.UTF-8"
  },
  {
    "Env variable": "SSL_CERT_FILE",
    "Env value": "/etc/ssl/certs/ca-certificates.crt"
  },
  {
    "Env variable": "HOME",
    "Env value": "/root"
  }
]
```

## Set or Replace Environment Variable

### Code

```
@api.route('/set-env', methods = ['PUT'])
def set_env():

    content_type = request.headers.get('Content-Type')

    if (content_type == 'application/json'):
        json_data = request.json
        os.environ[json_data['Env variable']] = json_data['Env value']

        json_response = {
            "success": "true"
        }
    else:

        json_response = {
            "success": "false",
            "message": "content-Type not supported"
        }

    return json.dumps(json_response)
```

### Request

```
PUT /list-env
{
  "Env variable" : "test",
  "Env value" : "lalala"
}
```

## CURL example

```
curl -XPUT http://localhost/set-env \  
-H 'Content-Type: application/json' \  
-d '{"Env variable":"test","Env value":"lalala"}' | jq
```

```
{  
  "success": "true"  
}
```

```
[  
  {  
    "Env variable": "PATH",  
    "Env value": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"  
  },  
  {  
    "Env variable": "HOSTNAME",  
    "Env value": "4f74b5a8104c"  
  },  
  {  
    "Env variable": "LANG",  
    "Env value": "C.UTF-8"  
  },  
  {  
    "Env variable": "SSL_CERT_FILE",  
    "Env value": "/etc/ssl/certs/ca-certificates.crt"  
  },  
  {  
    "Env variable": "HOME",  
    "Env value": "/root"  
  },  
  {  
    "Env variable": "test",  
    "Env value": "lalala"  
  }  
]
```

# Unset Environment Variable

## Code

```
@api.route('/unset-env', methods = ['DELETE'])
def unset_env():

    content_type = request.headers.get('Content-Type')

    if (content_type == 'application/json'):
        json_data = request.json

        if json_data['Env variable'] in os.environ:
            del os.environ[json_data['Env variable']]

            json_response = {
                "success": "true"
            }

        else:
            json_response = {
                "success": "false",
                "message": "no env variable"
            }

    else:
        json_response = {
            "success": "false",
            "message": "content-Type not supported"
        }

    return json.dumps(json_response)
```

## Request

```
DELETE /unset-env
{
  "Env variable": "test"
}
```

## CURL example

```
curl -XDELETE http://localhost/unset-env \
-H 'Content-Type: application/json' \
-d '{"Env variable": "test"}' | jq
```

```
{
  "success": "true"
}
```

from

```
[
  {
    "Env variable": "PATH",
    "Env value": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  },
  {
    "Env variable": "HOSTNAME",
    "Env value": "4f74b5a8104c"
  },
  {
    "Env variable": "LANG",
    "Env value": "C.UTF-8"
  },
  {
    "Env variable": "SSL_CERT_FILE",
    "Env value": "/etc/ssl/certs/ca-certificates.crt"
  },
  {
    "Env variable": "HOME",
    "Env value": "/root"
  },
  {
    "Env variable": "test",
    "Env value": "lalala"
  }
]
```

to

```
[
  {
    "Env variable": "PATH",
    "Env value": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
  },
  {
    "Env variable": "HOSTNAME",
    "Env value": "4f74b5a8104c"
  },
  {
    "Env variable": "LANG",
    "Env value": "C.UTF-8"
  },
  {
    "Env variable": "SSL_CERT_FILE",
    "Env value": "/etc/ssl/certs/ca-certificates.crt"
  },
  {
    "Env variable": "HOME",
    "Env value": "/root"
  }
]
```

## Create or Replace File (json, yaml, env)

Code

```
@api.route('/file-env', methods = ['PUT'])
def file_env():
    content_type = request.headers.get('Content-Type')

    if (content_type == 'application/json'):

        json_data = request.json

        if json_data['Type'] == "json":

            json_array = []

            for env in os.environ:

                json_array.append({
                    env: os.getenv(env)
                })

            with open('/files/'+ json_data['File name'] + "." + json_data['Type'], 'w') as json_file:
                json.dump(json_array, json_file)

            json_response = {
                "success": "true",
                "message": json_data['Type'] + " file created file at /files/" + json_data['File name'] + "." + json_data['Type']
            }

        elif json_data['Type'] == "env":

            with open('/files/'+ json_data['File name'] + "." + json_data['Type'], 'w') as env_file:
                for env in os.environ:
                    env_file.write(env + "=" + os.getenv(env) + "\n")

            json_response = {
                "success": "true",
                "message": json_data['Type'] + " file created file at /files/" + json_data['File name'] + "." + json_data['Type']
            }

        elif json_data['Type'] == "yaml":

            yaml_array = []

            for env in os.environ:

                yaml_array.append({
                    env : os.getenv(env)
                })

            with open('/files/'+ json_data['File name'] + "." + json_data['Type'], 'w') as yaml_file:
                yaml.dump(yaml_array, yaml_file)

            json_response = {
                "success": "true",
                "message": json_data['Type'] + " file created file at /files/" + json_data['File name'] + "." + json_data['Type']
            }
        else:
            json_response = {
                "success": "false",
                "message": "wrong json parameter"
            }

    else:
        json_response = {
            "success": "false",
            "message": "content-Type not supported"
        }
    return json.dumps(json_response)
```



## Request

```
PUT /file-env
{
  "File name": "testfile",
  "Type": "json/env/yaml"
}
```

## CURL example

```
curl -XPUT http://localhost/file-env \
-H 'Content-Type: application/json' \
-d '{"File name":"testfile","Type":"env"}'
```

```
{
  "success": "true",
  "message": "env file created file at /files/test3.env"
}
```

## List File

### Code

```
@api.route('/list-file', methods = ['GET'])
def list_file():

    json_array= []

    for file in os.listdir("/files/"):

        extension = os.path.splitext("/files/" + file)[1]

        json_array.append({
            "File name": file,
            "File type": extension[1:]
        })

    return json.dumps(json_array)
```

### Request

```
GET /list-file
```

### CURL example

```
curl -XGET http://localhost/list-file | jq
```

```
[
  {
    "File name": "test3.env",
    "File type": "env"
  }
]
```

## Download File

### Code

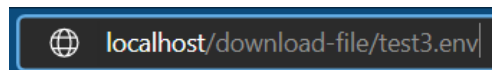
```
@api.route('/download-env/<filename>', methods = ['GET'])
def download_env(filename):
    return send_file('/files/' + filename, as_attachment=True)
```

### Request


```
GET /download-env/<filename>
```

### Download from browser

```
http://localhost/download/test3.env
```



### Downloads

 test3.env  
[Open file](#)

```
1 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
2 HOSTNAME=b82de4dd4ff4
3 LANG=C.UTF-8
4 SSL_CERT_FILE=/etc/ssl/certs/ca-certificates.crt
5 HOME=/root
```