FuncX Container Service

Ben Galewsky NCSA University of Illinois

Steve Wangen, Steve Goldstein University of Wisconsin, Data Science Institute

Aristana Scourtas, Ben Blaiszik University of Chicago/Globus Labs

National Center for Supercomputing Applications

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Introduction

- Dependency Management in FuncX
- Containers
- Introducing the Container Service
- dlHub Servables
- Example
- Next Steps

Dependency Management in FuncX

- Without container service, if your funcX function requires libraries that are not built-into your python runtime you will need a way to install them
  - Endpoint configuration offers worker initialization commands

```
provider=CondorProvider(
                worker_init='module load python/3.7.0;
                python3 -m venv parsl_env;
                source parsl_env/bin/activate;
                python3 -m pip install parsl',
                walltime="00:20:00",
            )
```

Containers

- There is a better way to manage dependencies for funcX workers: Containers!
  - Can specify operating system, shared packages, python version, and installed libraries
  - Aid reproducibility since these images are frozen at build time

Docker
- Very popular
- Images can be published to public repositories
- Security concerns when running on a host

Shifter/Singularity
- Supported on HPCs
- Images built and stored on HPC facility
- Designed to eliminate the security concerns of Docker

Dependency Management in FuncX With Containers

- Register a container with funcX and associate it with a function
  - A single container ID can be backed by both docker and singularity images

```
container_uuid = fxc.register_container("bengal1/pyhf-funcx:3.8-0.6.3",
                                        "Docker")

function = fxc.register_function(fitting_func,container_uuid=container_uuid)
```

Introducing the Container Service

- Building an image requires some technical skills
- The workflow to build and publish the image sit outside of funcX
- Container Service introduces new calls to the SDK
  - Build container from specification
    - Python Version
    - APT packages
    - Packages from pypi
    - Conda Packages
    - Archives of files to include in built image

Introducing the Container Service

- Building an image requires some technical skills
- The workflow to build and publish the image sit outside of funcX
- Container Service introduces new calls to the SDK
  - Build container from specification
    - Python Version
    - APT packages
    - Packages from pypi
    - Conda Packages
    - Archives of files to include in built image

```
class ContainerSpec:
    def __init__(
        self,
        name=None,
        description=None,
        apt=None,
        pip=None,
        conda=None,
        payload_url=None,
    ):
```

DLHub Servables

- DLHub is a popular service to find, share, publish, and run machine learning models uses funcX to execute *Servables* out of Docker containers
- The DLHub service has basic functionality for building these images from DLHub SDK
  - Not general purpose
  - Expensive to maintain as standalone service
- Code is being removed from DLHub service and migrated to funcX Container Service

Example

```
from funcx import ContainerSpec
from funcx.sdk.client import FuncXClient
fxc = FuncXClient()
    container_uuid = fxc.build_container(
        ContainerSpec(
            name="WineFileReader",
            pip=[
                "pandas"
            ],
            conda=[
                "python=3.10"
            ]
        )
    )
```

```
while True:
    status =
      fxc.get_container_build_status(container_uuid)

    print(f"status is {status}")
    if status in ["ready", "failed"]:
        break
    sleep(5)

print(fxc.get_container(container_uuid,
                        container_type="docker"))
```

Example Output

```
Building 4b23054f-2a5b-48f6-b073-b7eecb769cbd
status is building
status is ready
{'build_status': 'ready', 'build_stderr': ---> Using cache  --->
2017963b0295 Step 50/51 : ENTRYPOINT ["/usr/local/bin/repo2docker-
entrypoint"]  ---> Using cache  ---> d6833df2e41f Step 51/51 : CMD
["jupyter", "notebook", "--ip", "0.0.0.0"]  ---> Using cache  --->
a149d78035a9 {"aux": {"ID":
"sha256:a149d78035a93d8beb414d48e0b3d60d5c1b2777afcdba1e8037831fdd7c72c
0"}}Successfully built a149d78035a9 Successfully tagged funcx_4b23054f-
2a5b-48f6-b073-b7eecb769cbd:latest', 'container_uuid': '4b23054f-2a5b-
48f6-b073-b7eecb769cbd', 'location': 'docker.io/bengal1/funcx_4b23054f-
2a5b-48f6-b073-b7eecb769cbd:latest', 'name': 'WineFileReader', 'type':
'docker'}
```

Next Steps

- Service is in final internal testing. Will be released as part of funcX in October
- Only builds Docker Images
- DLHub being retrofitted to use this service

Thank You

Ben Galewsky

bengal1@Illinois.edu

This work was supported by the National Science Foundation under NSF Award Number: 1931306 "Collaborative Research: Framework: Machine Learning Materials Innovation Infrastructure".