Building a Freesurfer Gear

Outline

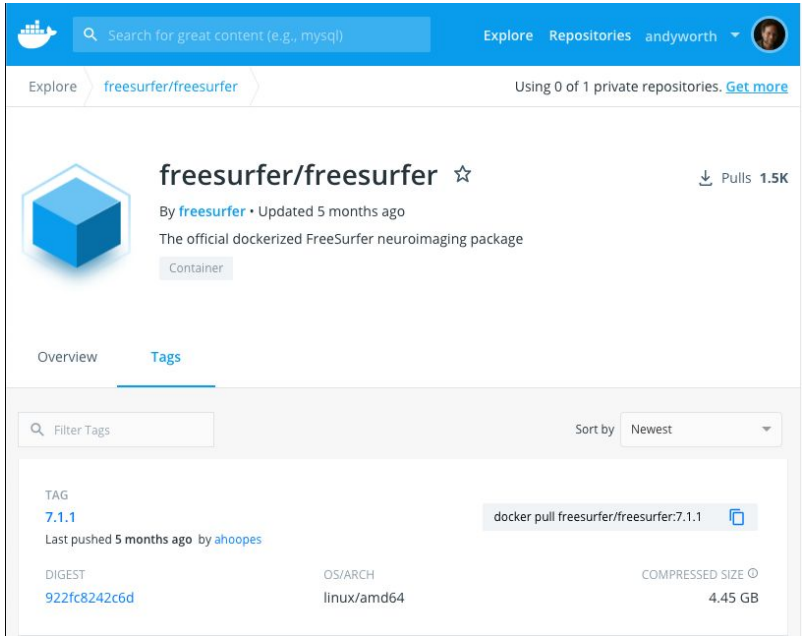
- Requirements
- Building
  - Manifest
  - Dockerfile
  - run.py
- Testing

Flywheel

Requirements

Freesurfer Code

- Now provided as a Docker image!



Possible Development Environment

- Python environment on your local machine for
  - Running run.py
  - Local “dry-run” testing
- pip install
  - flywheel-sdk~=14.5.0
  - flywheel-gear-toolkit~=0.1.1
- Flywheel CLI + Docker
  - Docker to build and test your Gear
  - CLI to upload to Flywheel

Flywheel

Outline

- Requirements
- Building
  - **Manifest**
  - Dockerfile
  - run.py
- Testing

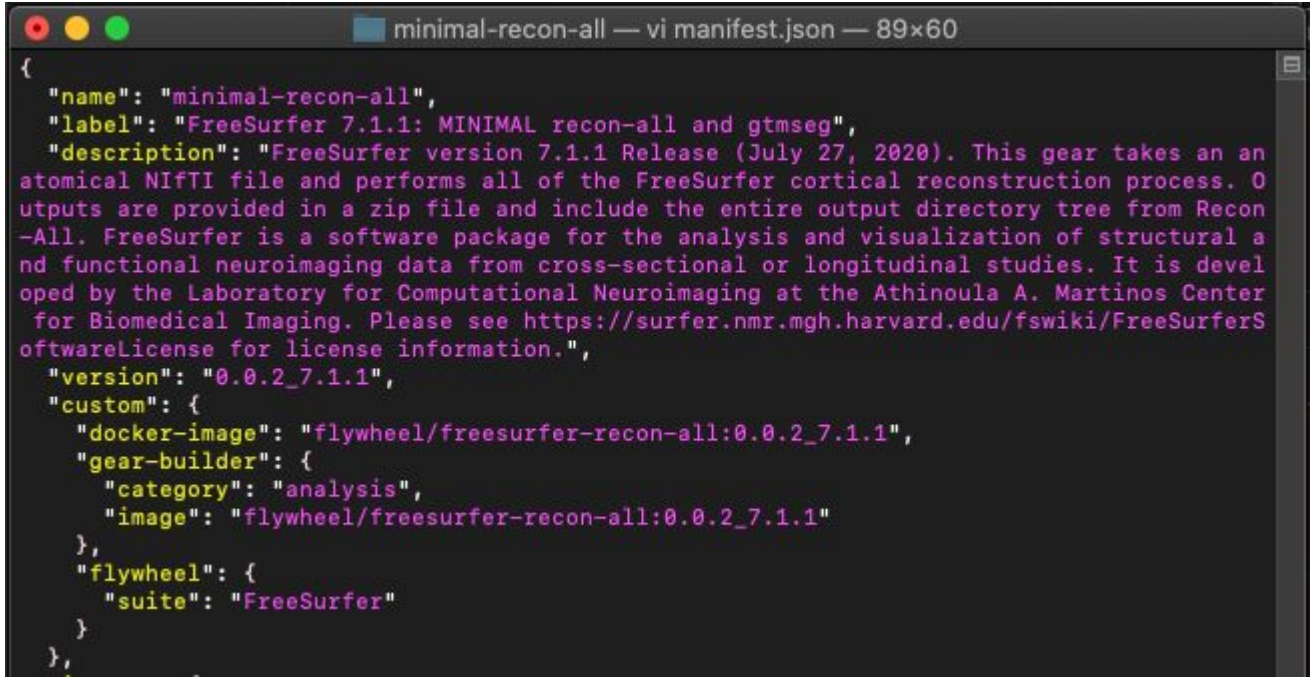
Flywheel

Building

Manifest

- Name
- Label
- Description
- Version(s)
-

minimal-recon-all — vi manifest.json — 89×60



```
{
  "name": "minimal-recon-all",
  "label": "FreeSurfer 7.1.1: MINIMAL recon-all and gtmseg",
  "description": "FreeSurfer version 7.1.1 Release (July 27, 2020). This gear takes an an
atomical NIfTI file and performs all of the FreeSurfer cortical reconstruction process. O
utputs are provided in a zip file and include the entire output directory tree from Recon
-All. FreeSurfer is a software package for the analysis and visualization of structural a
nd functional neuroimaging data from cross-sectional or longitudinal studies. It is devel
oped by the Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center
 for Biomedical Imaging. Please see https://surfer.nmr.mgh.harvard.edu/fswiki/FreeSurferS
oftwareLicense for license information.",
  "version": "0.0.2_7.1.1",
  "custom": {
    "docker-image": "flywheel/freesurfer-recon-all:0.0.2_7.1.1",
    "gear-builder": {
      "category": "analysis",
      "image": "flywheel/freesurfer-recon-all:0.0.2_7.1.1"
    },
    "flywheel": {
      "suite": "FreeSurfer"
    }
  },
```

Flywheel

Building

Manifest

-
- Inputs
  - NIfTI file
  - License
-

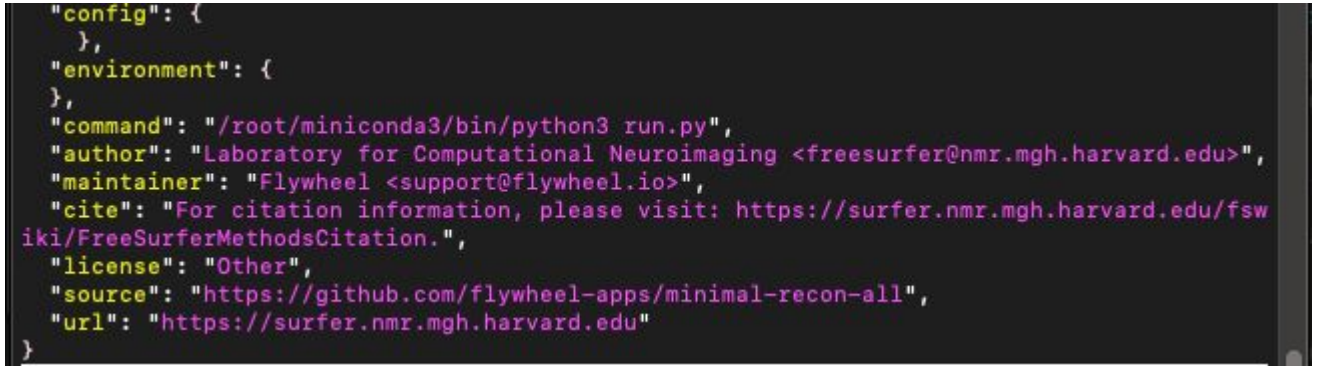
```
  "inputs": {
    "api-key": {
      "base": "api-key",
      "read-only": true
    },
    "anatomical": {
      "description": "Anatomical NIfTI file, DICOM archive, or previous freesurfer-recon-
all zip archive",
      "base": "file",
      "type": {
        "enum": [
          "nifti"
        ]
      }
    },
    "freesurfer_license": {
      "description": "FreeSurfer license file, provided during registration with FreeSurf
er. This file will by copied to the $FSHOME directory and used during execution of the Ge
ar.",
      "base": "file",
      "optional": true
    }
  },
```

Flywheel

Building

Manifest

-
- Config
- Environment
- **Command**
- Miscellaneous



Flywheel

Outline

- Requirements
- Building
  - Manifest
  - **Dockerfile**
  - run.py
- Testing

Flywheel

Building

Dockerfile

First try:

- Base image
- Environment
- Gear requirements
- Gear setup

minimal-recon-all — vi old0/Dockerfile — 89×60

```
FROM freesurfer/freesurfer:7.1.1 as base

LABEL maintainer="support@flywheel.io"

RUN source $FREESURFER_HOME/SetUpFreeSurfer.sh

# Save environment so it can be passed in when running recon-all.
RUN python -c 'import os, json; f = open("/tmp/gear_environ.json", "w"); json.dump(dict(o
s.environ), f)'

COPY requirements.txt /tmp
RUN pip install -r /tmp/requirements.txt && \
    rm -rf /root/.cache/pip

# Make directory for flywheel spec (v0)
ENV FLYWHEEL /flywheel/v0
WORKDIR ${FLYWHEEL}

# Copy executable/manifest to Gear
COPY manifest.json ${FLYWHEEL}/manifest.json
COPY run.py ${FLYWHEEL}/run.py

# Configure entrypoint
RUN chmod a+x ${FLYWHEEL}/run.py
ENTRYPOINT ["/flywheel/v0/run.py"]
```

```
DOCKER_IMAGE_NAME=`cat manifest.json | jq '.custom."gear-builder".image' | tr -d '"'`

docker build -f Dockerfile -t "${DOCKER_IMAGE_NAME}" .
```

Flywheel

Building

Dockerfile

```
docker run -it --rm \
    --volume "`pwd`:/src" \
    --volume "$HOME/.config/flywheel:/root/.config/flywheel" \
    "${ENTRY_POINT}" \
    "${TESTING_IMAGE}" \
    "$@"
```

Problems!

Run with ENTRY_POINT as /bin/bash to find out:

- Can't find a good Python
- Default Python is version 2
- Buried in Freesurfer is Python version 3.6 (don't want to mess with that)
- Pip is not installed anywhere

So, install another version of Python to run the gear code and keep it separated

Flywheel

Building

Dockerfile

Second try:

- Install Python 3.8
- *After* saving the environment
- *Before* installing packages

```
# Save environment so it can be passed in when running recon-all.
RUN python -c 'import os, json; f = open("/tmp/gear_environ.json", "w"); json.dump(dict(o
s.environ), f)'

# Install a version of python to run Flywheel code and keep it separate from the
# python that Freesurfer uses.  Saving the environment above makes sure it is not
# changed in the Flyfwheel environment.

# Set CPATH for packages relying on compiled libs (e.g. indexed_gzip)
ENV PATH="/root/miniconda3/bin:$PATH" \
    CPATH="/root/miniconda3/include/:$CPATH" \
    LANG="C.UTF-8" \
    PYTHONNOUSERSITE=1

RUN wget \
    https://repo.anaconda.com/miniconda/Miniconda3-py38_4.8.3-Linux-x86_64.sh \
    && mkdir /root/.conda \
    && bash Miniconda3-py38_4.8.3-Linux-x86_64.sh -b \
    && rm -f Miniconda3-py38_4.8.3-Linux-x86_64.sh

# Installing precomputed python packages
RUN conda install -y python=3.8.5 && \
    chmod -R a+rX /root/miniconda3; sync && \
    chmod +x /root/miniconda3/bin/*; sync && \
    conda build purge-all; sync && \
    conda clean -tipsy && sync

COPY requirements.txt /tmp
RUN pip install -r /tmp/requirements.txt && \
    rm -rf /root/.cache/pip
```

Now **run.py** runs in a conda environment inside a container (belt and suspenders)

Flywheel

Building

Dockerfile

minimal-recon-all — vi Dockerfile — 89×60

```
FROM freesurfer/freesurfer:7.1.1 as base

LABEL maintainer="support@flywheel.io"

RUN yum clean all -y \
  && yum update -y \
  && yum install -y unzip \
  && yum clean all -y

RUN source $FREESURFER_HOME/SetUpFreeSurfer.sh

# extra segmentations require matlab compiled runtime
RUN fs_install_mcr R2014b

# Save environment so it can be passed in when running recon-all.
RUN python -c 'import os, json; f = open("/tmp/gear_environ.json", "w"); json.dump(dict(o
s.environ), f)'
```

More problems!

- In Run.py, -brainstem-structures is now a script: segmentBS.sh
- The script requires a Matlab runtime to run
- And unzip as well

Now this command can be run in the container:

recon-all -i MPR_T1w.nii.gz -subjid sub-001 -3T -all && segmentBS.sh sub-001 && gtmseg --s sub-001

Flywheel

Outline

- Requirements
- Building
  - Manifest
  - Dockerfile
  - **run.py**
- Testing

Flywheel

Building

run.py

Use Flywheel Gear Toolkit

- Logging
- Access to inputs
- Access to config
- Helpful functions that improve your quality of life

minimal-recon-all — vi run.py — 89×60

```
#!/usr/bin/env python3
"""Run the gear: set up for and call command-line command."""

import json
import sys
from pathlib import Path

import flywheel_gear_toolkit
from flywheel_gear_toolkit.interfaces.command_line import exec_command
from flywheel_gear_toolkit.licenses.freesurfer import install_freesurfer_license
from flywheel_gear_toolkit.utils.zip_tools import zip_output

SUBJECTS_DIR = Path("/usr/local/freesurfer/subjects")
LICENSE_FILE = "/usr/local/freesurfer/license.txt"


def main(gtk_context):

    gtk_context.init_logging("debug")
    gtk_context.log_config()
    log = gtk_context.log

    acquisition_id =  gtk_context.config_json["inputs"]["anatomical"]["hierarchy"]["id"]
    file_name =  gtk_context.config_json["inputs"]["anatomical"]["location"]["name"]
    log.info(f"acquisition {acquisition_id} {file_name}")
```

Flywheel

Building

run.py

```
fw = gtk_context.client
full_file = fw.get_acquisition_file_info(acquisition_id, file_name)
field_strength = full_file.info.get('MagneticFieldStrength')
log.info(f"field_strength = {field_strength}")

# grab environment for gear (saved in Dockerfile)
with open("/tmp/gear_environ.json", "r") as f:
    environ = json.load(f)

install_freesurfer_license(gtk_context, LICENSE_FILE)

subject_id = fw.get_analysis(gtk_context.destination["id"]).parents.subject
subject = fw.get_subject(subject_id)
subject_id = subject.label

subject_dir = Path(SUBJECTS_DIR / subject_id)
work_dir = gtk_context.output_dir / subject_id
if not work_dir.is_symlink():
    work_dir.symlink_to(subject_dir)

anat_dir = Path("/flywheel/v0/input/anatomical")
anatomical_list = list(anat_dir.rglob("*.nii*"))
anatomical = str(anatomical_list[0])
```

SDK Calls:

- Get field strength
- Get subject label

Get FS environment

Gear Toolkit

- FS License
- Work directory
  (link to Freesurfer's "subject" directory)

Flywheel

Building

run.py

The command itself!

- recon-all
- segmentBS.sh
- gtmseg

```
# The main command line command to be run:
command = [
    "recon-all",
    "-i",
    anatomical,
    "-subjid",
    subject_id]
if field_strength == 3:
    command.append("-3T")
command += ["-all",
    "&&",
    "segmentBS.sh",
    subject_id,
    "&&",
    "gtmseg",
    "--s",
    subject_id,
]
```

Flywheel

Building

run.py

Execute command

- try/except
- subprocess
- Logs error

```
try:
    return_code = 0

    exec_command(
        command,
        environ=environ,
        dry_run=False,
        shell=True,
        cont_output=True,
    )

except RuntimeError as exc:
    log.critical(exc)
    log.exception("Unable to execute command.")
    return_code = 1
```

Flywheel

Building

run.py

Finish

- Save FS Subject dir
- Clean up

```
    # zip entire output/<subject_id> folder into
    #  <gear_name>_<subject_id>_<analysis.id>.zip
    zip_file_name = (
        gtk_context.manifest["name"]
        + f"_{subject_id}_{gtk_context.destination['id']}.zip"
    )
    if subject_dir.exists():
        log.info("Saving %s in %s as output", subject_id, SUBJECTS_DIR)
        zip_output(str(gtk_context.output_dir), subject_id, zip_file_name)
    else:
        log.error("Could not find %s in %s", subject_id, SUBJECTS_DIR)

    # clean up: remove symbolic link to subject so it won't be in output
    if work_dir.exists():
        log.debug('removing output directory "%s"', str(work_dir))
        work_dir.unlink()
    else:
        log.info("Output directory does not exist so it cannot be removed")

    log.info("Gear is done.  Returning %d", return_code)

    sys.exit(return_code)


if __name__ == "__main__":
    gear_toolkit_context = flywheel_gear_toolkit.GearToolkitContext()
    main(gear_toolkit_context)
```

Flywheel

Outline

- Requirements
- Building
  - Manifest
  - Dockerfile
  - run.py
- **Testing**

Flywheel

Testing

Local Testing inside Docker container

`./tests/bin/docker-test.sh`

```
USAGE="
Usage:
    $0 [OPTION...] [[--] TEST_ARGS...]
Run tests in a docker container.
Options:
    -h, --help      Print this help and exit
    -B, --no-build  Don't build the docker image (use existing)
    -s, --shell     Drop into the container with bash instead of normal entry
    -- TEST_ARGS    Arguments passed to tests.sh
"
```

Test Scripts

- Build
- Runs Tests

```
    docker run -it --rm \
        --volume "`pwd`:/src" \
        --volume "$HOME/.config/flywheel:/root/.config/flywheel" \
        "${ENTRY_POINT}" \
        "${TESTING_IMAGE}" \
        "$@"
```

Run inside container: `/src/tests/bin/tests.sh`

Finally, upload Gear: `fw gear upload`

Flywheel

The Whole Gear

1. Test locally,
2. Upload and test on platform
3. Download job
4. Test locally,
5. Upload for final end to end test

```
exec_command(
    command,
    environ=environ,
    dry_run=True,
    shell=True,
    cont_output=True,
)
```

```
C02DC1DFMD6M:minimal-recon-all andyworth % ls                                    main ■ ]
Dockerfile        README.md         manifest.json     run.py*
LICENSE           htmlcov/          requirements.txt  tests/
C02DC1DFMD6M:minimal-recon-all andyworth % tree tests                            main ■ ]
tests
├── Dockerfile
├── bin
│   ├── docker-test.sh
│   ├── fwutil_get_job.py
│   ├── pack-gear-tests.py
│   ├── tests.sh
│   └── unpack-gear-tests.py
├── conftest.py
├── data
│   └── gear_tests
│       ├── dry_run
│       │   ├── config.json
│       │   ├── input
│       │   │   └── anatomical
│       │   │       └── T1w_MPR_27.nii.gz
│       │   └── output
│       ├── dry_run.zip
│       ├── platform
│       │   ├── config.json
│       │   ├── input
│       │   │   ├── anatomical
│       │   │   │   └── sub-TOME3024_ses-Session2_acq-MPR_T1w.nii.gz
│       │   │   └── freesurfer_license
│       │   │       └── license.txt
│       │   └── output
│       └── platform.zip
├── integration_tests
│   └── test_run.py
└── requirements.txt
```

Testing

Integration Testing

platform

Run on Flywheel,

Use job # to create test

```
[C02DC1DFMD6M:gear_tests andyworth % ../../bin/fwutil_get_job.py 5fda7c1a2295c736acc02ee0 ]
Creating directory: /Users/andyworth/Flywheel/github/flywheel-apps/minimal-recon-all/test
s/data/gear_tests/minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc02ee0/input
Creating directory: /Users/andyworth/Flywheel/github/flywheel-apps/minimal-recon-all/test
s/data/gear_tests/minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc02ee0/output
Created directory: /Users/andyworth/Flywheel/github/flywheel-apps/minimal-recon-all/tests
/data/gear_tests/minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc02ee0/input/anatomical
Downloading: sub-TOME3024_ses-Session2_acq-MPR_T1w.nii.gz
Created directory: /Users/andyworth/Flywheel/github/flywheel-apps/minimal-recon-all/tests
/data/gear_tests/minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc02ee0/input/freesurfer_
license
Downloading: license.txt
Done!
[C02DC1DFMD6M:gear_tests andyworth % ls minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc0]
2ee0
config.json  input/       output/
[C02DC1DFMD6M:gear_tests andyworth % mv minimal-recon-all-0.0.1_7.1.1_5fda7c1a2295c736acc0]
2ee0 platform
[C02DC1DFMD6M:gear_tests andyworth % tree platform                                  main ■ ]
platform
├── config.json
├── input
│   ├── anatomical
│   │   └── sub-TOME3024_ses-Session2_acq-MPR_T1w.nii.gz
│   └── freesurfer_license
│       └── license.txt
└── output

4 directories, 3 files
```

```
fwutil_get_job.py 5fda7c1a2295c736acc02ee0
```

Flywheel

Testing

Integration Testing

`./tests/bin/docker-test.sh`

Example Test

```python
def test_platform_works(capfd, install_gear, print_captured, search_sysout):

    user_json = Path(Path.home() / ".config/flywheel/user.json")
    if not user_json.exists():
        TestCase.skipTest("", f"No API key available in {str(user_json)}")

    install_gear("platform.zip")

    with flywheel_gear_toolkit.GearToolkitContext(input_args=[]) as gtk_context:

        with pytest.raises(SystemExit) as excinfo:

            run.main(gtk_context)

        captured = capfd.readouterr()
        print_captured(captured)

        assert excinfo.type == SystemExit
        assert excinfo.value.code == 0
        assert search_sysout(captured, "-3T -all && segmentBS.sh sub-TOME3024")
```

tests/integration_tests/test_run.py

Flywheel

Testing

Integration Testing

platform

Run on Flywheel,

Use job # to create test

gear_tests — vi platform/config.json — 89×60


```
{
    "config": {},
    "inputs": {
        "anatomical": {
            "base": "file",
            "hierarchy": {
                "type": "acquisition",
                "id": "5dc091f669d4f3002d16f33e"
            },
            "location": {
                "name": "sub-TOME3024_ses-Session2_acq-MPR_T1w.nii.gz",
                "path": "/flywheel/v0/input/anatomical/sub-TOME3024_ses-Session2_acq-MPR_
T1w.nii.gz"
            },
            "object": {
                "info": {
                    "AcquisitionDateTime": "2017-09-06T09:14:12.392500",
```


```
vi platform/config.json
```

Flywheel

Complete Example

freesurfer-recon-all

https://github.com/flywheel-apps/freesurfer-recon-all

Code for this demo: https://github.com/flywheel-apps/minimal-recon-all

Testing based on https://github.com/flywheel-apps/bids-app-template

Flywheel


Flywheel