# Migrating Workflows to Pegasus 5.0


Ryan Tanaka
Programmer Analyst
tanaka@isi.edu

USC Viterbi
Information Sciences Institute

April 2, 2021

https://pegasus.isi.edu/

# You've installed Pegasus 5.0, next steps?

- Existing workflows written in the DAX3 format need to be updated to the new YAML format
  - Support for older formats will end in Pegasus 5.1 (~ 9 months from now)
- New workflows should be written using the new Python API
- Submit node must have Python3.5+
- Perform database upgrade

# Overview of Changes Affecting Workflow Development

- File Format Changes
- default data staging configuration is **condorio**
  - If using the default prior to v4.9, **sharedfs,** this needs to be explicitly set in your properties file `pegasus.data.configuration sharedfs`
- by default, output files have `register_replica=True`
- Default site catalog containing 2 sites is generated if none is given:
  - **local**
  - **condorpool**

| Entity | Pegasus 4.x | Pegasus 5.0 |
|---|---|---|
| Workflow | DAX (XML) | YAML |
| Site Catalog | XML | YAML |
| Transformation Catalog | Text | YAML |
| Replica Catalog | Text | YAML |

# Overview of Changes Affecting Workflow Development

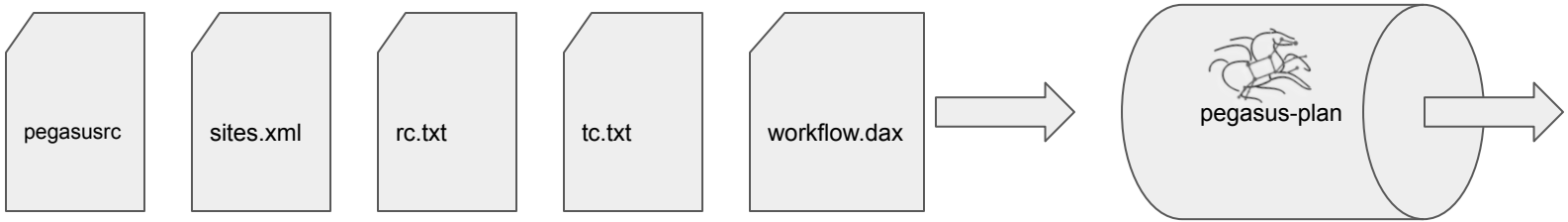
- Database upgrade required using `pegasus-db-admin update -a`
- Python API (formerly DAX3) reworked from the ground up
  - build properties file and catalogs programmatically
  - API usage simplified (adding inputs/outputs takes less lines of code)
  - plan/submit/analyze/monitor directly from workflow script

# Outline

- ~~Overview of Changes~~
- **Pegasus 4.X vs Pegasus 5.0 Workflow Scripts**
- Migrating to New Python API
- Useful Tips
- Resources

# Pegasus 4.x Workflow Script

- Shell script that invokes pegasus-plan at the end
- Catalogs generated either in shell script or already on filesystem
  - Transformation catalog and replica catalog may be omitted as they can be embedded into the workflow with limited functionality
- pegasusrc generated either in shell script or already on filesystem
- Workflow generation script typically invoked from within shell script to generate some <workflow name>.dax file



# Pegasus 4.x Workflow Script

plan.sh

```
#!/bin/bash

set -e
set -v

TOP_DIR=$( cd "$(dirname "${BASH_SOURCE[0]}")"; pwd -P )
cd $TOP_DIR

# build the dax generator
export CLASSPATH=.:`pegasus-config --classpath`
javac --release 8 BlackDiamondDAX.java

# generate the dax
java BlackDiamondDAX /usr blackdiamond.xml

# create the site catalog
cat >sites.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog" xmlns:xsi="http://www.w3.org/2001/X
ecatalog http://pegasus.isi.edu/schema/sc-4.0.xsd" version="4.0">
<site  handle="condorpool" arch="x86_64" os="LINUX" osrelease="" osversion="" glibc="">
    <profile namespace="env" key="PEGASUS_HOME" >/usr</profile>
    <profile namespace="condor" key="universe" >vanilla</profile>
    <profile namespace="pegasus" key="style" >condor</profile>
</site>
<site  handle="local" arch="x86_64" os="LINUX" osrelease="" osversion="7" glibc="">
    <directory  path="$HOME/workflows/scratch" type="shared-scratch" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/scratch">
        </file-server>
    </directory>
    <directory  path="$HOME/workflows/outputs" type="local-storage" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/outputs">
        </file-server>
    </directory>
</site>
</sitecatalog>
EOF

# plan the workflow
pegasus-plan \
    --output-sites local \
    --conf pegasusrc \
    blackdiamond.xml
```

# Pegasus 4.x Workflow Script

plan.sh

setup

```
#!/bin/bash

set -e
set -v

TOP_DIR=$( cd "$(dirname "${BASH_SOURCE[0]}")"; pwd -P )
cd $TOP_DIR

# build the dax generator
export CLASSPATH=.:`pegasus-config --classpath`
javac --release 8 BlackDiamondDAX.java

# generate the dax
java BlackDiamondDAX /usr blackdiamond.xml

# create the site catalog
cat >sites.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog" xmlns:xsi="http://www.w3.org/2001/X
ecatalog http://pegasus.isi.edu/schema/sc-4.0.xsd" version="4.0">
<site  handle="condorpool" arch="x86_64" os="LINUX" osrelease="" osversion="" glibc="">
    <profile namespace="env" key="PEGASUS_HOME" >/usr</profile>
    <profile namespace="condor" key="universe" >vanilla</profile>
    <profile namespace="pegasus" key="style" >condor</profile>
</site>
<site  handle="local" arch="x86_64" os="LINUX" osrelease="" osversion="7" glibc="">
    <directory  path="$HOME/workflows/scratch" type="shared-scratch" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/scratch">
        </file-server>
    </directory>
    <directory  path="$HOME/workflows/outputs" type="local-storage" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/outputs">
        </file-server>
    </directory>
</site>
</sitecatalog>
EOF

# plan the workflow
pegasus-plan \
    --output-sites local \
    --conf pegasusrc \
    blackdiamond.xml
```

# Pegasus 4.x Workflow Script

setup

generate workflow file

plan.sh

```
#!/bin/bash

set -e
set -v

TOP_DIR=$( cd "$(dirname "${BASH_SOURCE[0]}")"; pwd -P )
cd $TOP_DIR

# build the dax generator
export CLASSPATH=.:`pegasus-config --classpath`
javac --release 8 BlackDiamondDAX.java

# generate the dax
java BlackDiamondDAX /usr blackdiamond.xml

# create the site catalog
cat >sites.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog" xmlns:xsi="http://www.w3.org/2001/X
ecatalog http://pegasus.isi.edu/schema/sc-4.0.xsd" version="4.0">
<site  handle="condorpool" arch="x86_64" os="LINUX" osrelease="" osversion="" glibc="">
    <profile namespace="env" key="PEGASUS_HOME" >/usr</profile>
    <profile namespace="condor" key="universe" >vanilla</profile>
    <profile namespace="pegasus" key="style" >condor</profile>
</site>
<site  handle="local" arch="x86_64" os="LINUX" osrelease="" osversion="7" glibc="">
    <directory  path="$HOME/workflows/scratch" type="shared-scratch" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/scratch">
        </file-server>
    </directory>
    <directory  path="$HOME/workflows/outputs" type="local-storage" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/outputs">
        </file-server>
    </directory>
</site>
</sitecatalog>
EOF

# plan the workflow
pegasus-plan \
    --output-sites local \
    --conf pegasusrc \
    blackdiamond.xml
```

Diamond.java

```
/**
 *  Copyright 2007-2008 University Of Southern California
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *  http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing,
 *  software distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

import edu.isi.pegasus.planner.dax.*;

public class BlackDiamondDAX {

    /**
     * Create an example DIAMOND DAX
     * @param args
     */
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java ADAG <site_handle> <pegasus_location>
            System.exit(1);
        }

        try {
            Diamond(args[0]).writeToFile(args[1]);
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    }

    private static ADAG Diamond(String pegasus_location) throws Exception {

        ADAG dax = new ADAG("blackdiamond");

        File fb1 = new File("f.b1");
        File fb2 = new File("f.b2");
        File fc1 = new File("f.c1");
        File fc2 = new File("f.c2");
        File fd = new File("f.d");
    File fe1 = new File("f.e1");
    File fe2 = new File("f.e2");
        fd.setRegister(true);
```

# Pegasus 4.x Workflow Script

setup

generate workflow file

generate catalogs/pegasusrc

plan.sh

```
#!/bin/bash

set -e
set -v

TOP_DIR=$( cd "$(dirname "${BASH_SOURCE[0]}")"; pwd -P )
cd $TOP_DIR

# build the dax generator
export CLASSPATH=.:`pegasus-config --classpath`
javac --release 8 BlackDiamondDAX.java

# generate the dax
java BlackDiamondDAX /usr blackdiamond.xml

# create the site catalog
cat >sites.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog" xmlns:xsi="http://www.w3.org/2001/
ecatalog http://pegasus.isi.edu/schema/sc-4.0.xsd" version="4.0">
<site  handle="condorpool" arch="x86_64" os="LINUX" osrelease="" osversion="" glibc="">
    <profile namespace="env" key="PEGASUS_HOME" >/usr</profile>
    <profile namespace="condor" key="universe" >vanilla</profile>
    <profile namespace="pegasus" key="style" >condor</profile>
</site>
<site  handle="local" arch="x86_64" os="LINUX" osrelease="" osversion="7" glibc="">
    <directory  path="$HOME/workflows/scratch" type="shared-scratch" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/scratch">
        </file-server>
    </directory>
    <directory  path="$HOME/workflows/outputs" type="local-storage" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/outputs">
        </file-server>
    </directory>
</site>
</sitecatalog>
EOF

# plan the workflow
pegasus-plan \
    --output-sites local \
    --conf pegasusrc \
    blackdiamond.xml
```

Diamond.java

```
/**
 *  Copyright 2007-2008 University Of Southern California
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *  http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing,
 *  software distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

import edu.isi.pegasus.planner.dax.*;

public class BlackDiamondDAX {

    /**
     * Create an example DIAMOND DAX
     * @param args
     */
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java ADAG <site_handle> <pegasus_location>
            System.exit(1);
        }

        try {
            Diamond(args[0]).writeToFile(args[1]);
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    }

    private static ADAG Diamond(String pegasus_location) throws Exception {

        ADAG dax = new ADAG("blackdiamond");

        File fb1 = new File("f.b1");
        File fb2 = new File("f.b2");
        File fc1 = new File("f.c1");
        File fc2 = new File("f.c2");
        File fd = new File("f.d");
    File fe1 = new File("f.e1");
    File fe2 = new File("f.e2");
        fd.setRegister(true);
```

# Pegasus 4.x Workflow Script

setup

generate workflow file

generate catalogs/pegasusrc

invoke pegasus-plan

plan.sh

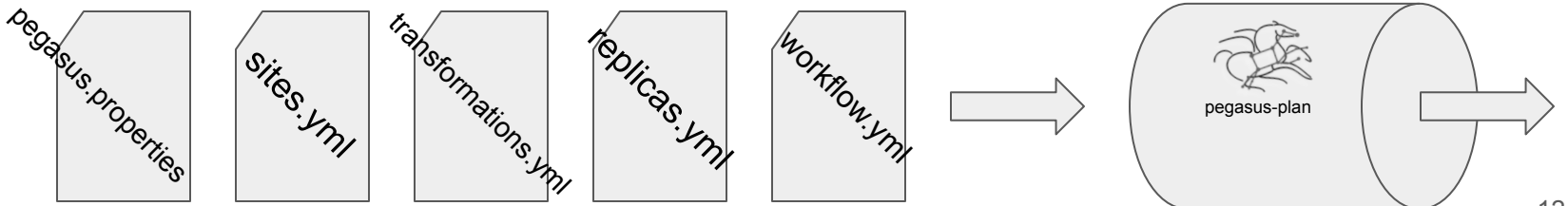
```
#!/bin/bash

set -e
set -v

TOP_DIR=$( cd "$(dirname "${BASH_SOURCE[0]}")"; pwd -P )
cd $TOP_DIR

# build the dax generator
export CLASSPATH=.:`pegasus-config --classpath`
javac --release 8 BlackDiamondDAX.java

# generate the dax
java BlackDiamondDAX /usr blackdiamond.xml

# create the site catalog
cat >sites.xml <<EOF
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog" xmlns:xsi="http://www.w3.org/2001/
ecatalog http://pegasus.isi.edu/schema/sc-4.0.xsd" version="4.0">
<site  handle="condorpool" arch="x86_64" os="LINUX" osrelease="" osversion="" glibc="">
    <profile namespace="env" key="PEGASUS_HOME" >/usr</profile>
    <profile namespace="condor" key="universe" >vanilla</profile>
    <profile namespace="pegasus" key="style" >condor</profile>
</site>
<site  handle="local" arch="x86_64" os="LINUX" osrelease="" osversion="7" glibc="">
    <directory  path="$HOME/workflows/scratch" type="shared-scratch" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/scratch">
        </file-server>
    </directory>
    <directory  path="$HOME/workflows/outputs" type="local-storage" free-size="" total-size="">
        <file-server  operation="all" url="file://$HOME/workflows/outputs">
        </file-server>
    </directory>
</site>
</sitecatalog>
EOF

# plan the workflow
pegasus-plan \
    --output-sites local \
    --conf pegasusrc \
    blackdiamond.xml
```

Diamond.java

```
/**
 *  Copyright 2007-2008 University Of Southern California
 *
 *  Licensed under the Apache License, Version 2.0 (the "License");
 *  you may not use this file except in compliance with the License.
 *  You may obtain a copy of the License at
 *
 *  http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing,
 *  software distributed under the License is distributed on an "AS IS" BASIS,
 *  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 *  See the License for the specific language governing permissions and
 *  limitations under the License.
 */

import edu.isi.pegasus.planner.dax.*;

public class BlackDiamondDAX {

    /**
     * Create an example DIAMOND DAX
     * @param args
     */
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java ADAG <site_handle> <pegasus_location>
            System.exit(1);
        }

        try {
            Diamond(args[0]).writeToFile(args[1]);
        }
        catch (Exception e) {
            e.printStackTrace();
        }

    }

    private static ADAG Diamond(String pegasus_location) throws Exception {

        ADAG dax = new ADAG("blackdiamond");

        File fb1 = new File("f.b1");
        File fb2 = new File("f.b2");
        File fc1 = new File("f.c1");
        File fc2 = new File("f.c2");
        File fd = new File("f.d");
    File fe1 = new File("f.e1");
    File fe2 = new File("f.e2");
        fd.setRegister(true);
```

# Pegasus 5.0 Workflow Script

- Single script using the Pegasus 5.0 Python API
- Programmatically generate:
  - pegasus.properties (you may have seen pegasusrc, same file)
  - Site Catalog (sites.yml)
    - default one will be used if none is picked up by pegasus-plan
  - Replica Catalog (replicas.yml)
  - Transformation Catalog (transformations.yml)
- Catalogs remain separate entities from the abstract workflow
- Plan and run directly from this script



# Pegasus 5.0 Workflow Script

workflow.py

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Pegasus 5.0 Workflow Script

generate pegasus.properties

workflow.py

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Pegasus 5.0 Workflow Script

generate pegasus.properties

generate Replica Catalog

workflow.py

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Pegasus 5.0 Workflow Script

generate pegasus.properties

generate Replica Catalog

generate Transformation Catalog

workflow.py

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Pegasus 5.0 Workflow Script

workflow.py

generate pegasus.properties

generate Replica Catalog

generate Transformation Catalog

generate workflow

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Pegasus 5.0 Workflow Script

workflow.py

generate pegasus.properties

generate Replica Catalog

generate Transformation Catalog

generate workflow

plan and run workflow

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties ###################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###############################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog########################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #####################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #################################################################
wf.plan(submit=True).wait()
```

# Outline

- ~~Overview of Changes~~
- ~~Pegasus 4.X vs Pegasus 5.0 Workflow Scripts~~
- **Migrating to New Python API**
- Useful Tips
- Resources

# Migrating Workflow Scripts to New Python API

- Migrate catalogs from 4.x format to YAML
  - Generate new catalogs using the Python API (recommended) **OR**
  - Convert existing catalogs using provided catalog converters
    - `pegasus-sc-converter -i sites.xml -o sites.yml`
    - `pegasus-tc-converter -i tc.txt -I Text -O YAML -o transformations.yml`
    - `pegasus-rc-converter -I File -O YAML -i rc.txt -o replicas.yml transformations.yml`
- Migrate workflow generation script from using `Pegasus.DAX3` to `Pegasus.api`

# Python API: Site Catalog

sites.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog http://pegasus.isi.edu/schema
             version="4.0">

    <site  handle="local" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/tmp/workflows/scratch">
            <file-server operation="all" url="file:///tmp/workflows/scratch"/>
        </directory>
        <directory type="local-storage" path="/tmp/workflows/outputs">
            <file-server operation="all" url="file:///tmp/workflows/outputs"/>
        </directory>
    </site>

    <site  handle="condor_pool" arch="x86_64" os="LINUX">
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="auxillary"/>
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="compute"/>
        <directory type="shared-scratch" path="/lustre">
            <file-server operation="all" url="gsiftp://smarty.isi.edu/lustre"/>
        </directory>
    </site>

    <site  handle="staging_site" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/data">
            <file-server operation="put" url="scp://obelix.isi.edu/data"/>
            <file-server operation="get" url="http://obelix.isi.edu/data"/>
        </directory>
    </site>

</sitecatalog>
```

generate_sc.py

```python
# create a SiteCatalog object
sc = SiteCatalog()

# create a "local" site
local = Site("local", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch and local storage directories to the site "local"
local_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/tmp/workflows/scratch")\
                           .add_file_servers(FileServer("file:///tmp/workflows/scratch", Operation.ALL))

local_local_storage_dir = Directory(Directory.LOCAL_STORAGE, path="/tmp/workflows/outputs")\
                          .add_file_servers(FileServer("file:///tmp/workflows/outputs", Operation.ALL))

local.add_directories(local_shared_scratch_dir, local_local_storage_dir)

# create a "condorpool" site
condorpool = Site("condorpool", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add job managers to the site "condorpool"
condorpool.add_grids(
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.AUXILLARY),
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.COMPUTE)
)

# create and add a shared scratch directory to the site "condorpool"
condorpool_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/lustre")\
                                .add_file_servers(FileServer("gsiftp://smarty.isi.edu/lustre", Operation.ALL))
condorpool.add_directories(condorpool_shared_scratch_dir)

# create a "staging_site" site
staging_site = Site("staging_site", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch directory to the site "staging_site"
staging_site_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/data")\
                                  .add_file_servers(
                                      FileServer("scp://obelix.isi.edu/data", Operation.PUT),
                                      FileServer("http://obelix.isi.edu/data", Operation.GET)
                                  )
staging_site.add_directories(staging_site_shared_scratch_dir)

# add all the sites to the site catalog object
sc.add_sites(
    local,
    condorpool,
    staging_site
)

# write the site catalog to the default path "./sites.yml"
sc.write()
```

# Python API: Site Catalog

sites.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog http://pegasus.isi.edu/schema
             version="4.0">

    <site  handle="local" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/tmp/workflows/scratch">
            <file-server operation="all" url="file:///tmp/workflows/scratch"/>
        </directory>
        <directory type="local-storage" path="/tmp/workflows/outputs">
            <file-server operation="all" url="file:///tmp/workflows/outputs"/>
        </directory>
    </site>

    <site  handle="condor_pool" arch="x86_64" os="LINUX">
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="auxillary"/>
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="compute"/>
        <directory type="shared-scratch" path="/lustre">
            <file-server operation="all" url="gsiftp://smarty.isi.edu/lustre"/>
        </directory>
    </site>

    <site  handle="staging_site" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/data">
            <file-server operation="put" url="scp://obelix.isi.edu/data"/>
            <file-server operation="get" url="http://obelix.isi.edu/data"/>
        </directory>
    </site>

</sitecatalog>
```

generate_sc.py

```python
# create a SiteCatalog object
sc = SiteCatalog()

# create a "local" site
local = Site("local", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch and local storage directories to the site "local"
local_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/tmp/workflows/scratch")\
                           .add_file_servers(FileServer("file:///tmp/workflows/scratch", Operation.ALL))

local_local_storage_dir = Directory(Directory.LOCAL_STORAGE, path="/tmp/workflows/outputs")\
                          .add_file_servers(FileServer("file:///tmp/workflows/outputs", Operation.ALL))

local.add_directories(local_shared_scratch_dir, local_local_storage_dir)

# create a "condorpool" site
condorpool = Site("condorpool", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add job managers to the site "condorpool"
condorpool.add_grids(
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.AUXILLARY),
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.COMPUTE)
)

# create and add a shared scratch directory to the site "condorpool"
condorpool_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/lustre")\
                                .add_file_servers(FileServer("gsiftp://smarty.isi.edu/lustre", Operation.ALL))
condorpool.add_directories(condorpool_shared_scratch_dir)

# create a "staging_site" site
staging_site = Site("staging_site", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch directory to the site "staging_site"
staging_site_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/data")\
                                  .add_file_servers(
                                      FileServer("scp://obelix.isi.edu/data", Operation.PUT),
                                      FileServer("http://obelix.isi.edu/data", Operation.GET)
                                  )
staging_site.add_directories(staging_site_shared_scratch_dir)

# add all the sites to the site catalog object
sc.add_sites(
    local,
    condorpool,
    staging_site
)

# write the site catalog to the default path "./sites.yml"
sc.write()
```

# Python API: Site Catalog

sites.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog http://pegasus.isi.edu/schema
             version="4.0">

    <site  handle="local" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/tmp/workflows/scratch">
            <file-server operation="all" url="file:///tmp/workflows/scratch"/>
        </directory>
        <directory type="local-storage" path="/tmp/workflows/outputs">
            <file-server operation="all" url="file:///tmp/workflows/outputs"/>
        </directory>
    </site>

    <site  handle="condor_pool" arch="x86_64" os="LINUX">
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="auxillary"/>
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="compute"/>
        <directory type="shared-scratch" path="/lustre">
            <file-server operation="all" url="gsiftp://smarty.isi.edu/lustre"/>
        </directory>
    </site>

    <site  handle="staging_site" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/data">
            <file-server operation="put" url="scp://obelix.isi.edu/data"/>
            <file-server operation="get" url="http://obelix.isi.edu/data"/>
        </directory>
    </site>

</sitecatalog>
```

generate_sc.py

```python
# create a SiteCatalog object
sc = SiteCatalog()

# create a "local" site
local = Site("local", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch and local storage directories to the site "local"
local_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/tmp/workflows/scratch")\
                            .add_file_servers(FileServer("file:///tmp/workflows/scratch", Operation.ALL))

local_local_storage_dir = Directory(Directory.LOCAL_STORAGE, path="/tmp/workflows/outputs")\
                            .add_file_servers(FileServer("file:///tmp/workflows/outputs", Operation.ALL))

local.add_directories(local_shared_scratch_dir, local_local_storage_dir)

# create a "condorpool" site
condorpool = Site("condorpool", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add job managers to the site "condorpool"
condorpool.add_grids(
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.AUXILLARY),
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.COMPUTE)
)

# create and add a shared scratch directory to the site "condorpool"
condorpool_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/lustre")\
                                .add_file_servers(FileServer("gsiftp://smarty.isi.edu/lustre", Operation.ALL))
condorpool.add_directories(condorpool_shared_scratch_dir)

# create a "staging_site" site
staging_site = Site("staging_site", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch directory to the site "staging_site"
staging_site_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/data")\
                                  .add_file_servers(
                                      FileServer("scp://obelix.isi.edu/data", Operation.PUT),
                                      FileServer("http://obelix.isi.edu/data", Operation.GET)
                                  )
staging_site.add_directories(staging_site_shared_scratch_dir)

# add all the sites to the site catalog object
sc.add_sites(
    local,
    condorpool,
    staging_site
)

# write the site catalog to the default path "./sites.yml"
sc.write()
```

# Python API: Site Catalog

sites.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sitecatalog xmlns="http://pegasus.isi.edu/schema/sitecatalog"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://pegasus.isi.edu/schema/sitecatalog http://pegasus.isi.edu/schema
             version="4.0">

    <site  handle="local" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/tmp/workflows/scratch">
            <file-server operation="all" url="file:///tmp/workflows/scratch"/>
        </directory>
        <directory type="local-storage" path="/tmp/workflows/outputs">
            <file-server operation="all" url="file:///tmp/workflows/outputs"/>
        </directory>
    </site>

    <site  handle="condor_pool" arch="x86_64" os="LINUX">
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="auxillary"/>
        <grid type="gt5" contact="smarty.isi.edu/jobmanager-pbs" scheduler="PBS" jobtype="compute"/>
        <directory type="shared-scratch" path="/lustre">
            <file-server operation="all" url="gsiftp://smarty.isi.edu/lustre"/>
        </directory>
    </site>

    <site  handle="staging_site" arch="x86_64" os="LINUX">
        <directory type="shared-scratch" path="/data">
            <file-server operation="put" url="scp://obelix.isi.edu/data"/>
            <file-server operation="get" url="http://obelix.isi.edu/data"/>
        </directory>
    </site>

</sitecatalog>
```

generate_sc.py

```python
# create a SiteCatalog object
sc = SiteCatalog()

# create a "local" site
local = Site("local", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch and local storage directories to the site "local"
local_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/tmp/workflows/scratch")\
                            .add_file_servers(FileServer("file:///tmp/workflows/scratch", Operation.ALL))

local_local_storage_dir = Directory(Directory.LOCAL_STORAGE, path="/tmp/workflows/outputs")\
                            .add_file_servers(FileServer("file:///tmp/workflows/outputs", Operation.ALL))

local.add_directories(local_shared_scratch_dir, local_local_storage_dir)

# create a "condorpool" site
condorpool = Site("condorpool", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add job managers to the site "condorpool"
condorpool.add_grids(
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.AUXILLARY),
    Grid(Grid.GT5, contact="smarty.isi.edu/jobmanager-pbs", scheduler_type=Scheduler.PBS, job_type=SupportedJobs.COMPUTE)
)

# create and add a shared scratch directory to the site "condorpool"
condorpool_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/lustre")\
                                    .add_file_servers(FileServer("gsiftp://smarty.isi.edu/lustre", Operation.ALL))
condorpool.add_directories(condorpool_shared_scratch_dir)

# create a "staging_site" site
staging_site = Site("staging_site", arch=Arch.X86_64, os_type=OS.LINUX)

# create and add a shared scratch directory to the site "staging_site"
staging_site_shared_scratch_dir = Directory(Directory.SHARED_SCRATCH, path="/data")\
                                    .add_file_servers(
                                        FileServer("scp://obelix.isi.edu/data", Operation.PUT),
                                        FileServer("http://obelix.isi.edu/data", Operation.GET)
                                    )
staging_site.add_directories(staging_site_shared_scratch_dir)

# add all the sites to the site catalog object
sc.add_sites(
    local,
    condorpool,
    staging_site
)

# write the site catalog to the default path "./sites.yml"
sc.write()
```

# Python API: Transformation Catalog

tc.txt

```
cont centos-pegasus{
    type "docker"
    image "docker:///rynge/montage:latest"
    mount "/Volumes/Work/lfs1:/shared-data/:ro"
    profile env "JAVA_HOME" "/opt/java/1.6"
}

tr example::keg:1.0 {

    profile env "APP_HOME" "/tmp/myscratch"
    profile env "JAVA_HOME" "/opt/java/1.6"

    site isi {
        pfn "/path/to/keg
        arch "x86"
        os "linux"
        type "INSTALLED"
        container "centos-pegasus"
    }
}
```

generate_tc.py

```
# create the TransformationCatalog object
tc = TransformationCatalog()

# create and add the centos-pegasus container
centos_cont = Container(
                "centos-pegasus",
                Container.DOCKER,
                "docker:///rynge/montage:latest",
                mounts=["/Volumes/Workf/lfs1:/shared-data/:ro"]
            ).add_profiles(Namespace.ENV, JAVA_HOME="/opt/java/1.6")

tc.add_containers(centos_cont)

# create and add the transformation
keg = Transformation(
        "keg",
        namespace="example",
        version="1.0",
        site="isi",
        pfn="/path/to/keg",
        is_stageable=False,
        container=centos_cont
    ).add_profiles(Namespace.ENV, APP_HOME="/tmp/myscratch", JAVA_HOME="/opt/java/1.6")

tc.add_transformations(keg)

# write the transformation catalog to the default file path "./transformations.yml"
tc.write()
```

# Python API: Transformation Catalog

tc.txt

```
cont centos-pegasus{
    type "docker"
    image "docker:///rynge/montage:latest"
    mount "/Volumes/Work/lfs1:/shared-data/:ro"
    profile env "JAVA_HOME" "/opt/java/1.6"
}

tr example::keg:1.0 {

    profile env "APP_HOME" "/tmp/myscratch"
    profile env "JAVA_HOME" "/opt/java/1.6"

    site isi {
        pfn "/path/to/keg
        arch "x86"
        os "linux"
        type "INSTALLED"
        container "centos-pegasus"
    }
}
```

generate_tc.py

```
# create the TransformationCatalog object
tc = TransformationCatalog()

# create and add the centos-pegasus container
centos_cont = Container(
                "centos-pegasus",
                Container.DOCKER,
                "docker:///rynge/montage:latest",
                mounts=["/Volumes/Workf/lfs1:/shared-data/:ro"]
            ).add_profiles(Namespace.ENV, JAVA_HOME="/opt/java/1.6")

tc.add_containers(centos_cont)

# create and add the transformation
keg = Transformation(
        "keg",
        namespace="example",
        version="1.0",
        site="isi",
        pfn="/path/to/keg",
        is_stageable=False,
        container=centos_cont
    ).add_profiles(Namespace.ENV, APP_HOME="/tmp/myscratch", JAVA_HOME="/opt/java/1.6")

tc.add_transformations(keg)

# write the transformation catalog to the default file path "./transformations.yml"
tc.write()
```

# Python API: Transformation Catalog

tc.txt

```
cont centos-pegasus{
    type "docker"
    image "docker:///rynge/montage:latest"
    mount "/Volumes/Work/lfs1:/shared-data/:ro"
    profile env "JAVA_HOME" "/opt/java/1.6"
}

tr example::keg:1.0 {

    profile env "APP_HOME" "/tmp/myscratch"
    profile env "JAVA_HOME" "/opt/java/1.6"

    site isi {
        pfn "/path/to/keg
        arch "x86"
        os "linux"
        type "INSTALLED"
        container "centos-pegasus"
    }
}
```

generate_tc.py

```
# create the TransformationCatalog object
tc = TransformationCatalog()

# create and add the centos-pegasus container
centos_cont = Container(
                "centos-pegasus",
                Container.DOCKER,
                "docker:///rynge/montage:latest",
                mounts=["/Volumes/Workf/lfs1:/shared-data/:ro"]
            ).add_profiles(Namespace.ENV, JAVA_HOME="/opt/java/1.6")

tc.add_containers(centos_cont)

# create and add the transformation
keg = Transformation(
        "keg",
        namespace="example",
        version="1.0",
        site="isi",
        pfn="/path/to/keg",
        is_stageable=False,
        container=centos_cont
    ).add_profiles(Namespace.ENV, APP_HOME="/tmp/myscratch", JAVA_HOME="/opt/java/1.6")

tc.add_transformations(keg)

# write the transformation catalog to the default file path "./transformations.yml"
tc.write()
```

# Python API: Replica Catalog

rc.txt

```
f.a file:///Volumes/data/inputs/f.a site="local"

f.b file:///Volumes/data/inputs/f.b site="local"
```

generate_rc.py

```
rc = ReplicaCatalog()

rc.add_replica(
        site="local",
        lfn="f.a",
        pfn="/Volumes/data/inputs/f.a"
    )

rc.add_replica(
        site="local",
        lfn="f.b",
        pfn="/Volumes/data/inputs/f.b"
    )

rc.write()
```

# Python API: Replica Catalog

rc.txt

```
f.a file:///Volumes/data/inputs/f.a site="local"

f.b file:///Volumes/data/inputs/f.b site="local"
```

generate_rc.py

```
rc = ReplicaCatalog()

rc.add_replica(
        site="local",
        lfn="f.a",
        pfn="/Volumes/data/inputs/f.a"
    )

rc.add_replica(
        site="local",
        lfn="f.b",
        pfn="/Volumes/data/inputs/f.b"
    )

rc.write()
```

# Python API: Replica Catalog

rc.txt

```
1 f.a file:///Volumes/data/inputs/f.a site="local"
2
3 f.b file:///Volumes/data/inputs/f.b site="local"
```

generate_rc.py

```
1  rc = ReplicaCatalog()
2
3  rc.add_replica(
4          site="local",
5          lfn="f.a",
6          pfn="/Volumes/data/inputs/f.a"
7      )
8
9  rc.add_replica(
10         site="local",
11         lfn="f.b",
12         pfn="/Volumes/data/inputs/f.b"
13     )
14
15 rc.write()
```

# Python API: Workflow

daxgen.py

```
diamond = ADAG("diamond")

a = File("f.a")
b1 = File("f.b1")
b2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                name="preprocess",
                version="4.0"
            )

preprocess.addArguments("-T60","-i",a,"-o",b1,b2)

preprocess.uses(a, link=Link.INPUT)

preprocess.uses(b1, link=Link.OUTPUT)
preprocess.uses(b2, link=Link.OUTPUT)

diamond.addJob(preprocess)
```

workflow.py

```
wf = Workflow("diamond")

fa = File("f.a")
fb1 = File("f.b1")
fb2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                transformation="preprocess",
                version="4.0"
            )

preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)

preprocess.add_inputs(fa)

preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)

wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
diamond = ADAG("diamond")

a = File("f.a")
b1 = File("f.b1")
b2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                name="preprocess",
                version="4.0"
            )

preprocess.addArguments("-T60","-i",a,"-o",b1,b2)

preprocess.uses(a, link=Link.INPUT)

preprocess.uses(b1, link=Link.OUTPUT)
preprocess.uses(b2, link=Link.OUTPUT)

diamond.addJob(preprocess)
```

workflow.py

```
wf = Workflow("diamond")

fa = File("f.a")
fb1 = File("f.b1")
fb2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                transformation="preprocess",
                version="4.0"
            )

preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)

preprocess.add_inputs(fa)

preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)

wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
diamond = ADAG("diamond")

a = File("f.a")
b1 = File("f.b1")
b2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                name="preprocess",
                version="4.0"
            )

preprocess.addArguments("-T60","-i",a,"-o",b1,b2)

preprocess.uses(a, link=Link.INPUT)

preprocess.uses(b1, link=Link.OUTPUT)
preprocess.uses(b2, link=Link.OUTPUT)

diamond.addJob(preprocess)
```

workflow.py

```
wf = Workflow("diamond")

fa = File("f.a")
fb1 = File("f.b1")
fb2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                transformation="preprocess",
                version="4.0"
            )

preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)

preprocess.add_inputs(fa)

preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)

wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
1  diamond = ADAG("diamond")
2
3  a = File("f.a")
4  b1 = File("f.b1")
5  b2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  name="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.addArguments("-T60","-i",a,"-o",b1,b2)
14
15 preprocess.uses(a, link=Link.INPUT)
16
17 preprocess.uses(b1, link=Link.OUTPUT)
18 preprocess.uses(b2, link=Link.OUTPUT)
19
20 diamond.addJob(preprocess)
```

workflow.py

```
1  wf = Workflow("diamond")
2
3  fa = File("f.a")
4  fb1 = File("f.b1")
5  fb2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  transformation="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)
14
15 preprocess.add_inputs(fa)
16
17 preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)
18
19 wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
1  diamond = ADAG("diamond")
2
3  a = File("f.a")
4  b1 = File("f.b1")
5  b2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  name="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.addArguments("-T60","-i",a,"-o",b1,b2)
14
15 preprocess.uses(a, link=Link.INPUT)
16
17 preprocess.uses(b1, link=Link.OUTPUT)
18 preprocess.uses(b2, link=Link.OUTPUT)
19
20 diamond.addJob(preprocess)
```

workflow.py

```
1  wf = Workflow("diamond")
2
3  fa = File("f.a")
4  fb1 = File("f.b1")
5  fb2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  transformation="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)
14
15 preprocess.add_inputs(fa)
16
17 preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)
18
19 wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
1  diamond = ADAG("diamond")
2
3  a = File("f.a")
4  b1 = File("f.b1")
5  b2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  name="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.addArguments("-T60","-i",a,"-o",b1,b2)
14
15 preprocess.uses(a, link=Link.INPUT)
16
17 preprocess.uses(b1, link=Link.OUTPUT)
18 preprocess.uses(b2, link=Link.OUTPUT)
19
20 diamond.addJob(preprocess)
```

workflow.py

```
1  wf = Workflow("diamond")
2
3  fa = File("f.a")
4  fb1 = File("f.b1")
5  fb2 = File("f.b2")
6
7  preprocess = Job(
8                  namespace="diamond",
9                  transformation="preprocess",
10                 version="4.0"
11             )
12
13 preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)
14
15 preprocess.add_inputs(fa)
16
17 preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)
18
19 wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
diamond = ADAG("diamond")

a = File("f.a")
b1 = File("f.b1")
b2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                name="preprocess",
                version="4.0"
            )

preprocess.addArguments("-T60","-i",a,"-o",b1,b2)

preprocess.uses(a, link=Link.INPUT)

preprocess.uses(b1, link=Link.OUTPUT)
preprocess.uses(b2, link=Link.OUTPUT)

diamond.addJob(preprocess)
```

workflow.py

```
wf = Workflow("diamond")

fa = File("f.a")
fb1 = File("f.b1")
fb2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                transformation="preprocess",
                version="4.0"
            )

preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)

preprocess.add_inputs(fa)

preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)

wf.add_jobs(preprocess)
```

# Python API: Workflow

daxgen.py

```
diamond = ADAG("diamond")

a = File("f.a")
b1 = File("f.b1")
b2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                name="preprocess",
                version="4.0"
            )

preprocess.addArguments("-T60","-i",a,"-o",b1,b2)

preprocess.uses(a, link=Link.INPUT)

preprocess.uses(b1, link=Link.OUTPUT)
preprocess.uses(b2, link=Link.OUTPUT)

diamond.addJob(preprocess)
```

workflow.py

```
wf = Workflow("diamond")

fa = File("f.a")
fb1 = File("f.b1")
fb2 = File("f.b2")

preprocess = Job(
                namespace="diamond",
                transformation="preprocess",
                version="4.0"
            )

preprocess.add_args("-T", "3", "-i", fa, "-o", fb1, fb2)

preprocess.add_inputs(fa)

preprocess.add_outputs(fb1, fb2, register_replica=True, stage_out=True)

wf.add_jobs(preprocess)
```

# Python API: Hierarchical Workflow

daxgen.py

```
# Create a abstract dag
adag = ADAG('local-hierarchy')

daxfile = File('blackdiamond.dax')
dax1 = DAX(daxfile)
# DAX jobs are called with same arguments passed,
# while planning the root level dax
dax1.addArguments('--output-site local')
dax1.addArguments('-vvv')
adag.addJob(dax1)


# this dax job uses a pre-existing dax file
# that has to be present in the replica catalog
daxfile2 = File('sleep.dax')
dax2 = DAX(daxfile2)
dax2.addArguments('--output-site local')
dax2.addArguments( '-vvv')
adag.addJob(dax2)
```

workflow.py

```
blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
blackdiamond_wf.add_args(
                    "--input-dir",
                    "input",
                    "--output-sites",
                    "local",
                    "-vvv"
                )

sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
sleep_wf.add_args("--output-sites", "local", "-vvv")

wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Hierarchical Workflow

daxgen.py

```
# Create a abstract dag
adag = ADAG('local-hierarchy')

daxfile = File('blackdiamond.dax')
dax1 = DAX(daxfile)
# DAX jobs are called with same arguments passed,
# while planning the root level dax
dax1.addArguments('--output-site local')
dax1.addArguments('-vvv')
adag.addJob(dax1)


# this dax job uses a pre-existing dax file
# that has to be present in the replica catalog
daxfile2 = File('sleep.dax')
dax2 = DAX(daxfile2)
dax2.addArguments('--output-site local')
dax2.addArguments( '-vvv')
adag.addJob(dax2)
```

workflow.py

```
blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
blackdiamond_wf.add_args(
                    "--input-dir",
                    "input",
                    "--output-sites",
                    "local",
                    "-vvv"
                )

sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
sleep_wf.add_args("--output-sites", "local", "-vvv")

wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Hierarchical Workflow

daxgen.py

```
# Create a abstract dag
adag = ADAG('local-hierarchy')

daxfile = File('blackdiamond.dax')
dax1 = DAX(daxfile)
# DAX jobs are called with same arguments passed,
# while planning the root level dax
dax1.addArguments('--output-site local')
dax1.addArguments('-vvv')
adag.addJob(dax1)


# this dax job uses a pre-existing dax file
# that has to be present in the replica catalog
daxfile2 = File('sleep.dax')
dax2 = DAX(daxfile2)
dax2.addArguments('--output-site local')
dax2.addArguments( '-vvv')
adag.addJob(dax2)
```

workflow.py

```
blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
blackdiamond_wf.add_args(
                    "--input-dir",
                    "input",
                    "--output-sites",
                    "local",
                    "-vvv"
                )

sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
sleep_wf.add_args("--output-sites", "local", "-vvv")

wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Hierarchical Workflow

daxgen.py

```
# Create a abstract dag
adag = ADAG('local-hierarchy')

daxfile = File('blackdiamond.dax')
dax1 = DAX(daxfile)
# DAX jobs are called with same arguments passed,
# while planning the root level dax
dax1.addArguments('--output-site local')
dax1.addArguments('-vvv')
adag.addJob(dax1)


# this dax job uses a pre-existing dax file
# that has to be present in the replica catalog
daxfile2 = File('sleep.dax')
dax2 = DAX(daxfile2)
dax2.addArguments('--output-site local')
dax2.addArguments( '-vvv')
adag.addJob(dax2)
```

workflow.py

```
blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
blackdiamond_wf.add_args(
                    "--input-dir",
                    "input",
                    "--output-sites",
                    "local",
                    "-vvv"
                )

sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
sleep_wf.add_args("--output-sites", "local", "-vvv")

wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Hierarchical Workflow

daxgen.py

```
# Create a abstract dag
adag = ADAG('local-hierarchy')

daxfile = File('blackdiamond.dax')
dax1 = DAX(daxfile)
# DAX jobs are called with same arguments passed,
# while planning the root level dax
dax1.addArguments('--output-site local')
dax1.addArguments('-vvv')
adag.addJob(dax1)


# this dax job uses a pre-existing dax file
# that has to be present in the replica catalog
daxfile2 = File('sleep.dax')
dax2 = DAX(daxfile2)
dax2.addArguments('--output-site local')
dax2.addArguments( '-vvv')
adag.addJob(dax2)
```

workflow.py

```
blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
blackdiamond_wf.add_args(
                    "--input-dir",
                    "input",
                    "--output-sites",
                    "local",
                    "-vvv"
                )

sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
sleep_wf.add_args("--output-sites", "local", "-vvv")

wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Hierarchical Workflow

daxgen.py

```
1  # Create a abstract dag
2  adag = ADAG('local-hierarchy')
3
4  daxfile = File('blackdiamond.dax')
5  dax1 = DAX(daxfile)
6  # DAX jobs are called with same arguments passed,
7  # while planning the root level dax
8  dax1.addArguments('--output-site local')
9  dax1.addArguments('-vvv')
10 adag.addJob(dax1)
11
12
13 # this dax job uses a pre-existing dax file
14 # that has to be present in the replica catalog
15 daxfile2 = File('sleep.dax')
16 dax2 = DAX(daxfile2)
17 dax2.addArguments('--output-site local')
18 dax2.addArguments( '-vvv')
19 adag.addJob(dax2)
```

workflow.py

```
1  blackdiamond_wf = SubWorkflow("blackdiamond.yml", is_planned=False)
2  blackdiamond_wf.add_args(
3                      "--input-dir",
4                      "input",
5                      "--output-sites",
6                      "local",
7                      "-vvv"
8                  )
9
10 sleep_wf = SubWorkflow("sleep.yml", is_planned=False)
11 sleep_wf.add_args("--output-sites", "local", "-vvv")
12
13 wf.add_jobs(blackdiamond_wf, sleep_wf)
```

# Python API: Adding Profiles

**Profiles can be applied to:**

- FileServer
- Site
- Container
- Transformation Site
- Transformation
- Job
- SubWorkflow
- Workflow

daxgen.py

```
1 # add a Profile Object
2 job.addProfile(Profile(Namespace.ENV,'PATH','/bin'))
3
4 # add a profile directly
5 job.profile(Namespace.CONDOR, "universe", "vanilla")
```

# Python API: Adding Profiles

daxgen.py

```
1 # add a Profile Object
2 job.addProfile(Profile(Namespace.ENV,'PATH','/bin'))
3
4 # add a profile directly
5 job.profile(Namespace.CONDOR, "universe", "vanilla")
```

Create a Profile object, then pass it to job.addProfile()

# Python API: Adding Profiles

daxgen.py

```
1 # add a Profile Object
2 job.addProfile(Profile(Namespace.ENV,'PATH','/bin'))
3
4 # add a profile directly
5 job.profile(Namespace.CONDOR, "universe", "vanilla")
```

Call job.profile(), and set the profile directly.

# Python API: Adding Profiles

workflow.py

```
1  from Pegasus.api import *
2
3  j = Job("office-hours")
4
5  # option 1: ensures valid key is entered, can set multiple profiles
6  j.add_pegasus_profile(clusters_num=1, stageout_clusters=1)
7  j.add_env(PATH="/bin", JAVA_HOME="/java/home")
8  j.add_condor_profile(universe="vanilla")
9  j.add_dagman_profile(retry=3)
10 j.add_globus_profile(count=1)
11 j.add_selector_profile(execution_site="condorpool")
12
13 # option 2: lower level, can set multiple profiles
14 j.add_profiles(Namespace.ENV, PATH="/bin", JAVA_HOME="/java/home")
15
16 # option 3: lower level, use when keys have non-alphanumeric characters
17 j.add_profiles(Namespace.CONDOR, key="+KeyName", value="val")
```

# Python API: Adding Profiles

workflow.py

```
1  from Pegasus.api import *
2
3  j = Job("office-hours")
4
5  # option 1: ensures valid key is entered, can set multiple profiles
6  j.add_pegasus_profile(clusters_num=1, stageout_clusters=1)
7  j.add_env(PATH="/bin", JAVA_HOME="/java/home")
8  j.add_condor_profile(universe="vanilla")
9  j.add_dagman_profile(retry=3)
10 j.add_globus_profile(count=1)
11 j.add_selector_profile(execution_site="condorpool")
12
13 # option 2: lower level, can set multiple profiles
14 j.add_profiles(Namespace.ENV, PATH="/bin", JAVA_HOME="/java/home")
15
16 # option 3: lower level, use when keys have non-alphanumeric characters
17 j.add_profiles(Namespace.CONDOR, key="+KeyName", value="val")
```

Call obj.add_<namespace>_profile(key1=val1, key2=val2, ...)

# Python API: Adding Profiles

workflow.py

```
1   from Pegasus.api import *
2
3   j = Job("office-hours")
4
5   # option 1: ensures valid key is entered, can set multiple profiles
6   j.add_pegasus_profile(clusters_num=1, stageout_clusters=1)
7   j.add_env(PATH="/bin", JAVA_HOME="/java/home")
8   j.add_condor_profile(universe="vanilla")
9   j.add_dagman_profile(retry=3)
10  j.add_globus_profile(count=1)
11  j.add_selector_profile(execution_site="condorpool")
12
13  # option 2: lower level, can set multiple profiles
14  j.add_profiles(Namespace.ENV, PATH="/bin", JAVA_HOME="/java/home")
15
16  # option 3: lower level, use when keys have non-alphanumeric characters
17  j.add_profiles(Namespace.CONDOR, key="+KeyName", value="val")
```

Call obj.add_profiles(Namespace.<NS>, key1=val1, …)

# Python API: Adding Profiles

workflow.py

```
1  from Pegasus.api import *
2
3  j = Job("office-hours")
4
5  # option 1: ensures valid key is entered, can set multiple profiles
6  j.add_pegasus_profile(clusters_num=1, stageout_clusters=1)
7  j.add_env(PATH="/bin", JAVA_HOME="/java/home")
8  j.add_condor_profile(universe="vanilla")
9  j.add_dagman_profile(retry=3)
10 j.add_globus_profile(count=1)
11 j.add_selector_profile(execution_site="condorpool")
12
13 # option 2: lower level, can set multiple profiles
14 j.add_profiles(Namespace.ENV, PATH="/bin", JAVA_HOME="/java/home")
15
16 # option 3: lower level, use when keys have non-alphanumeric characters
17 j.add_profiles(Namespace.CONDOR, key="+KeyName", value="val")
```

Call obj.add_profiles(Namespace.<NS>, key=”+-.something”, value=”val”)

# Python API: Putting It All Together

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################
wf.plan(submit=True).wait()
```

# Python API: Putting It All Together

import Pegasus.api; setup

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################################
wf.plan(submit=True).wait()
```

# Python API: Putting It All Together

create properties file (config)

```
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################
wf.plan(submit=True).wait()
```

pegasus.properties

# Python API: Putting It All Together

create replica catalog
(specify input files)

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################
wf.plan(submit=True).wait()
```

pegasus.properties

replicas.yml

# Python API: Putting It All Together

create transformation catalog (specify executables)

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################
wf.plan(submit=True).wait()
```

pegasus.properties

replicas.yml

transformations.yml

# Python API: Putting It All Together

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################
wf.plan(submit=True).wait()
```

build abstract workflow

pegasus.properties

replicas.yml

transformations.yml

# Python API: Putting It All Together

```python
#!/usr/bin/env python3
import logging
from pathlib import Path

from Pegasus.api import *

logging.basicConfig(level=logging.INFO)

TOP_DIR = Path(__file__).parent.resolve()

### Properties #######################################################################
props = Properties()
props["pegasus.mode"] = "development"
props.write()

### Replica Catalog###################################################################
rc = ReplicaCatalog()
rc.add_replica(site="local", lfn="input.txt", pfn=TOP_DIR / "input.txt")
rc.write()

### Transformation Catalog############################################################
tc = TransformationCatalog()
compute = Transformation(
            "compute.sh",
            site="local",
            pfn=TOP_DIR / "compute.sh",
            is_stageable=True
        )
tc.add_transformations(compute)
tc.write()

### Workflow #########################################################################
wf = Workflow("office-hours-workflow")

in_file = File("input.txt")
out_file = File("output.txt")
j1 = Job(compute).add_args("OH").add_inputs(in_file).add_outputs(out_file)
wf.add_jobs(j1)

### Plan and Run #####################################################################
wf.plan(submit=True).wait()
```

plan and run workflow

files will be consumed by **pegasus-plan**

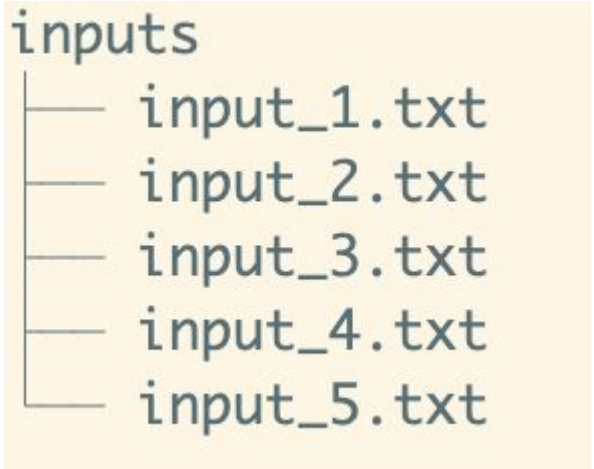
- pegasus.properties
- replicas.yml
- transformations.yml
- workflow.yml

# Outline

- ~~Overview of Changes~~
- ~~Pegasus 4.X vs Pegasus 5.0 Workflow Scripts~~
- ~~Migrating to New Python API~~
- **Useful Tips**
- Resources

# Useful Tips: Building Up the Replica Catalog

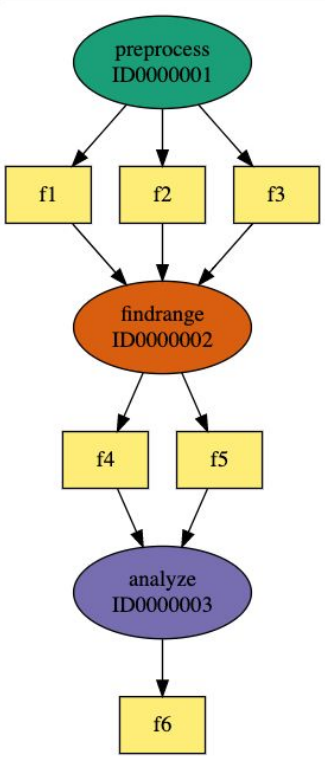
```python
from pathlib import Path

from Pegasus.api import *

rc = ReplicaCatalog()

input_dir = Path(".").parent / "inputs"
for p in input_dir.iterdir():
    rc.add_replica(site="local", lfn=p.name, pfn=p.resolve())

rc.write()
```




Loop through input directory and add all files to the Replica Catalog object.
(also see `pegasus-plan --input-dir dir1[,dir2,...]`)

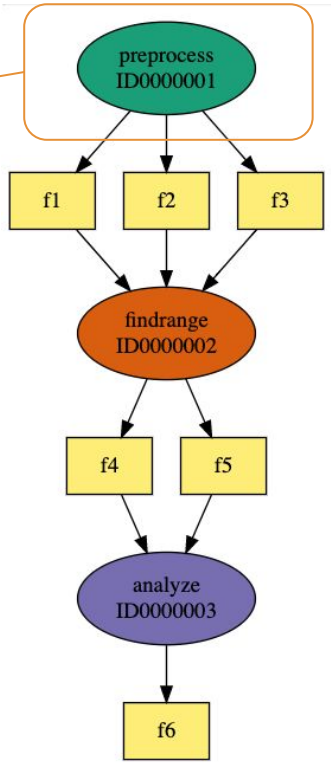
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



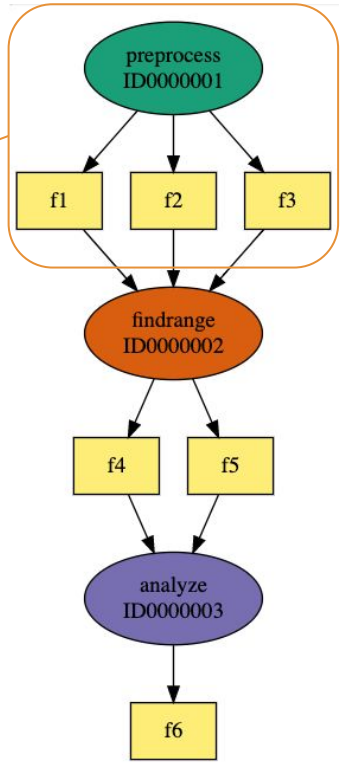
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



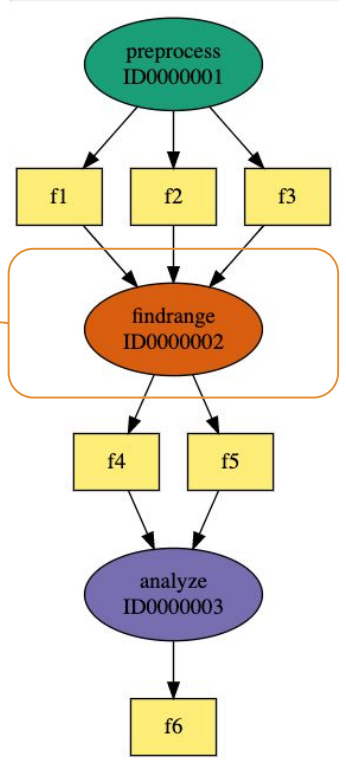
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



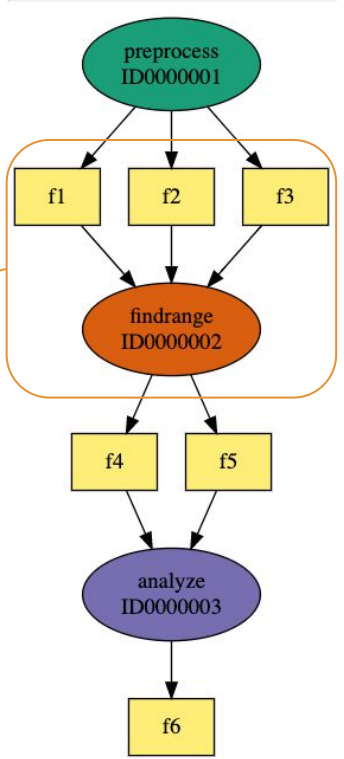
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



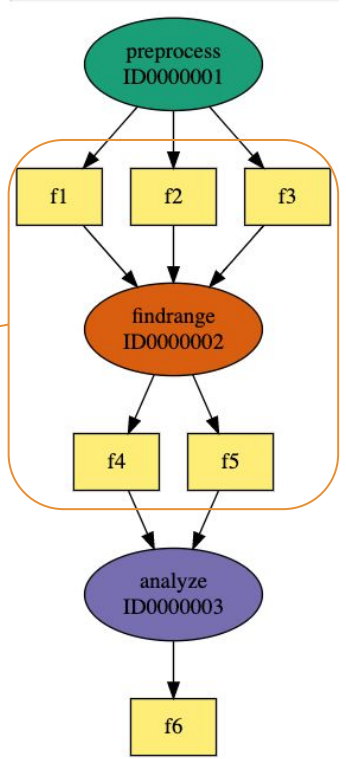
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



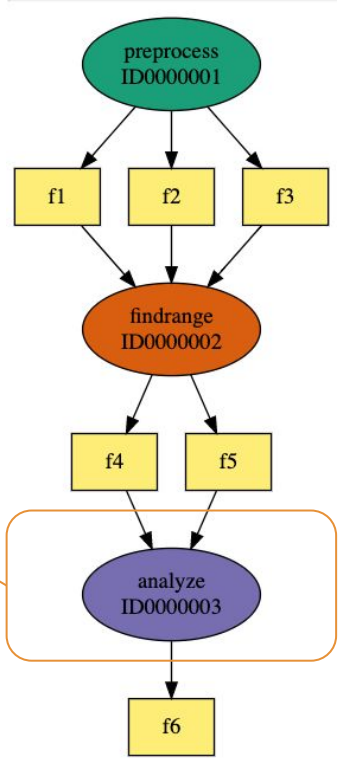
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



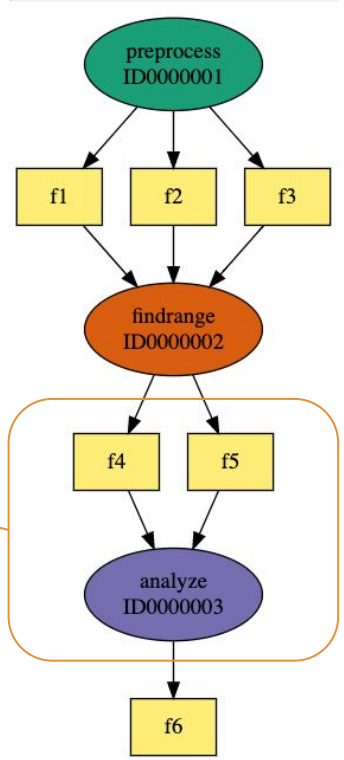
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



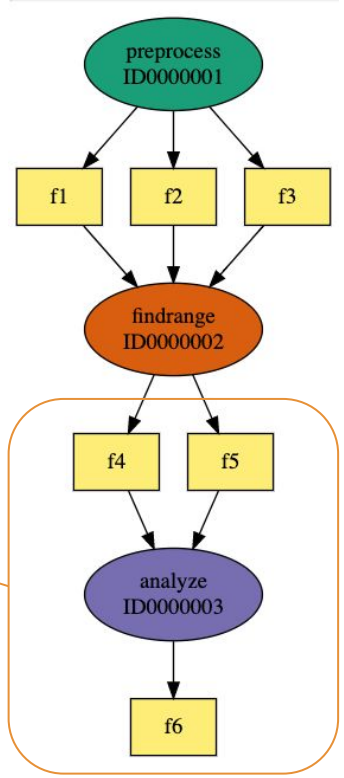
# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



# Useful Tips: Chaining Jobs Together

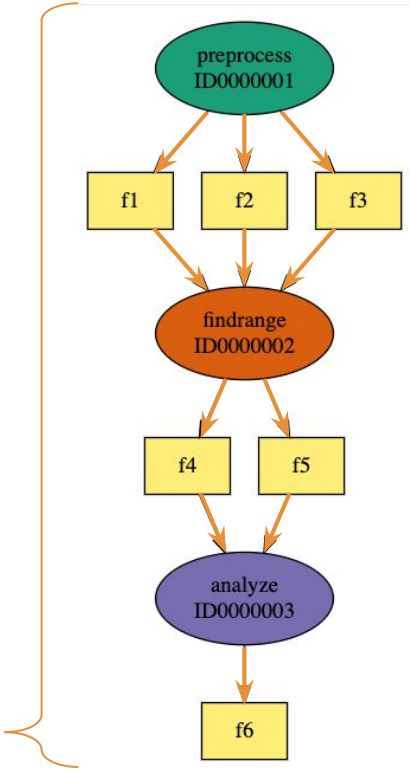
```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```



# Useful Tips: Chaining Jobs Together

```
1  wf = Workflow("office-hours")
2  job_1 = Job("preprocess")
3  job_1.add_outputs(File("f1"), File("f2"), File("f3"))
4
5  job_2 = Job("findrange")
6  job_2.add_inputs(*job_1.get_outputs())
7  job_2.add_outputs(File("f4"), File("f5"))
8
9  job_3 = Job("analyze")
10 job_3.add_inputs(*job_2.get_outputs())
11 job_3.add_outputs(File("f6"))
12
13 wf.add_jobs(job_1, job_2, job_3)
```

By default, dependencies are inferred based on input/output files created/used by each job.

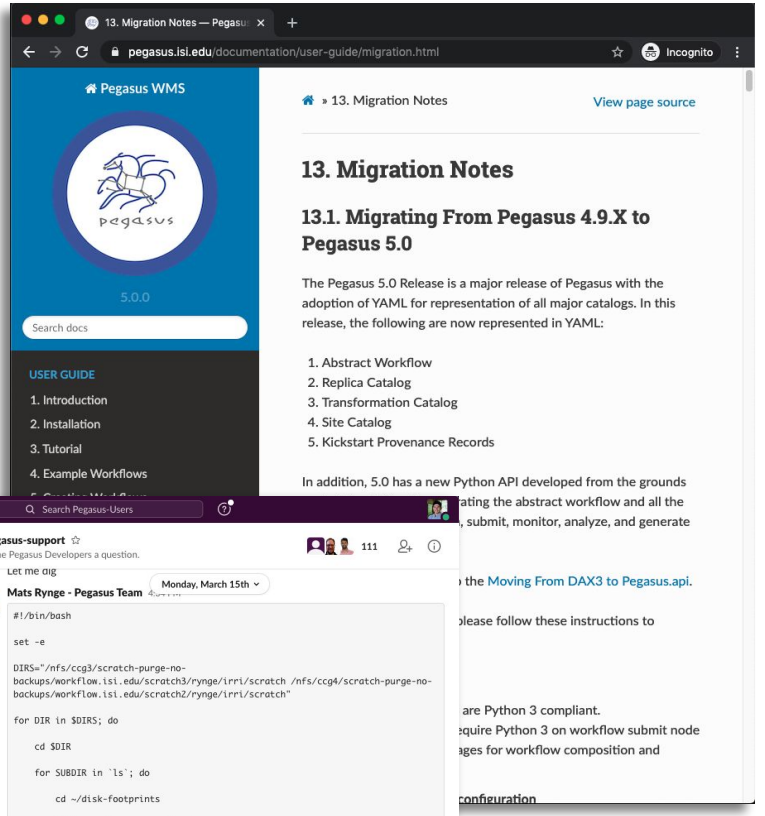


# Outline

- ~~Overview of Changes~~
- ~~Pegasus 4.X vs Pegasus 5.0 Workflow Scripts~~
- ~~Migrating to New Python API~~
- ~~Useful Tips~~
- **Resources**

# Resources

- Migration Notes
  - https://pegasus.isi.edu/documentation/user-guide/migration.html
- Python API Reference
  - https://pegasus.isi.edu/documentation/reference-guide/api-reference.html
- Pegasus Tutorial
  - https://pegasus.isi.edu/documentation/user-guide/tutorial.html
- Pegasus Team
  - https://pegasus-users.slack.com/archives/C01KBPSM64S



# Pegasus est. 2001

Automate, recover, and debug scientific computations

## Get Started

### Pegasus Website
https://pegasus.isi.edu/

### Users Mailing List
pegasus-users@isi.edu

### Pegasus Online Office Hours
https://pegasus.isi.edu/blog/online-pegasus-office-hours

*Bi-monthly basis on the second Friday of the month, where we address user questions and also apprise the community of new developments.*

### Pegasus Website
pegasus-support@isi.edu



USC Viterbi
*Information Sciences Institute*