Pegasus short tutorial

https://pegasus.isi.edu/

Pegasus has a number of features that contribute to its useability and effectiveness

- **Portability / Reuse**

  User created workflows can easily be run in different environments without alteration. Pegasus currently runs workflows on top of Condor, Grid infrastrucutures such as Open Science Grid and TeraGrid, Amazon EC2, Nimbus, and many campus clusters. The same workflow can run on a single system or across a heterogeneous set of resources.

- **Performance**

  The Pegasus mapper can reorder, group, and prioritize tasks in order to increase the overall workflow performance.

- **Scalability**

  Pegasus can easily scale both the size of the workflow, and the resources that the workflow is distributed over. Pegasus runs workflows ranging from just a few computational tasks up to 1 million. The number of resources involved in executing a workflow can scale as needed without any impediments to performance.

- **Provenance**

  By default, all jobs in Pegasus are launched via the kickstart process that captures runtime provenance of the job and helps in debugging. The provenance data is collected in a database, and the data can be summarised with tools such as pegasus-statistics, pegasus-plots, or directly with SQL queries.

- **Data Management**

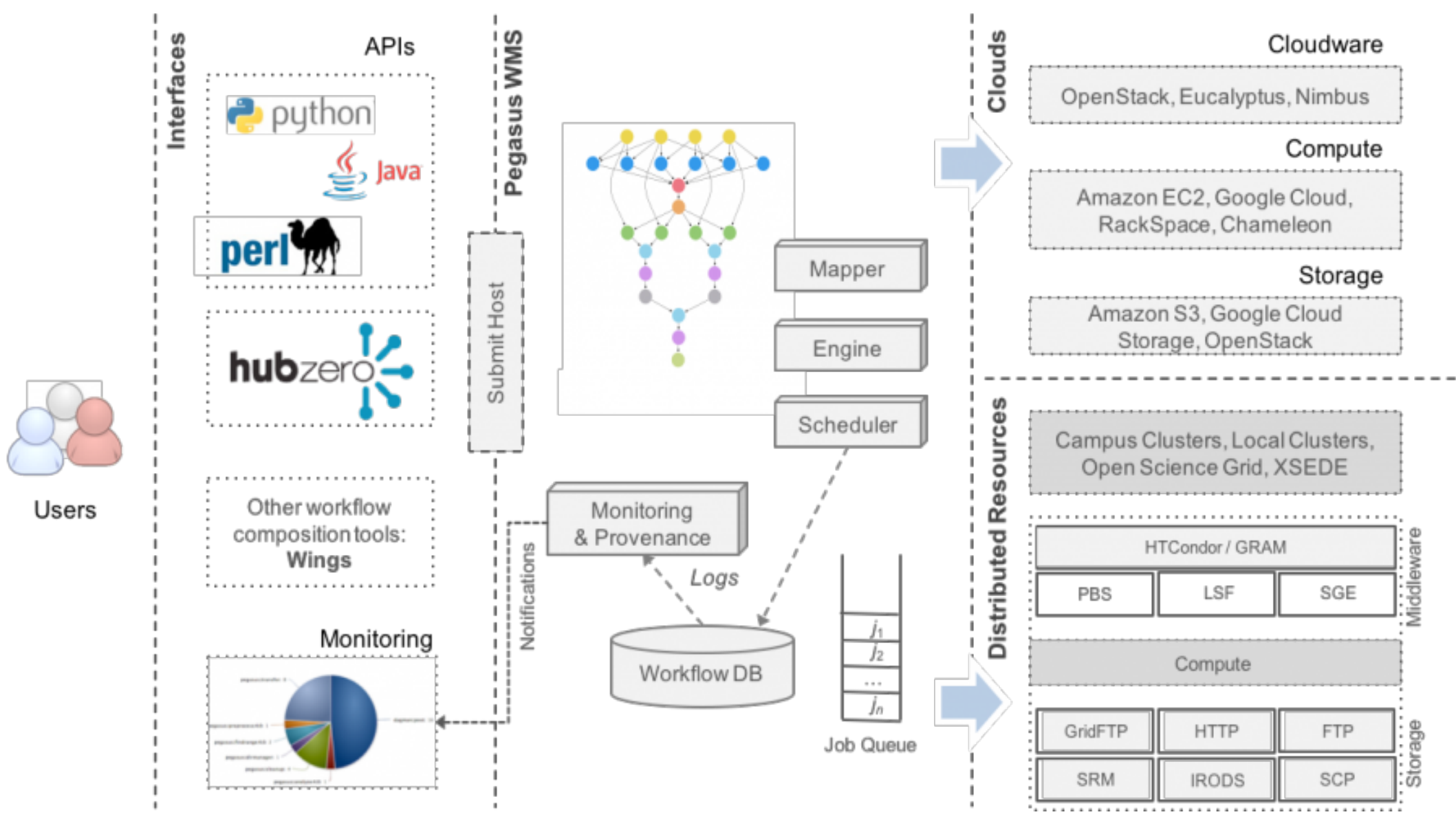
  Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxilliary jobs by the Pegasus planner.

- **Reliability**

  Jobs and data transfers are automatically retried in case of failures. Debugging tools such as pegasus-analyzer helps the user to debug the workflow in case of non-recoverable failure

- **Error Recovery**

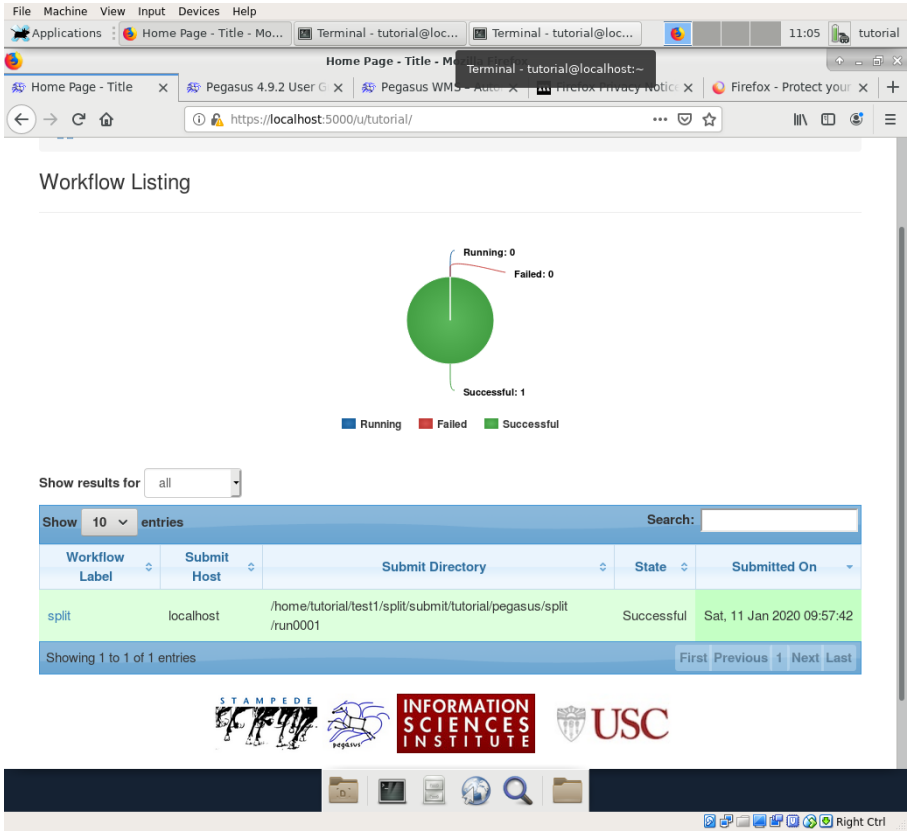
  When errors occur, Pegasus tries to recover when possible by retrying tasks, by retrying the entire workflow, by providing workflow-level checkpointing, by re-mapping portions of the workflow, by trying alternative data sources for staging data, and, when all else fails, by providing a rescue workflow containing a description of only the work that remains to be done. It cleans up storage as the workflow is executed so that data-intensive workflows have enough space to execute on storage-constrained resource. Pegasus keeps track of what has been done (provenance) including the locations of data used and produced, and which software was used with which parameters.

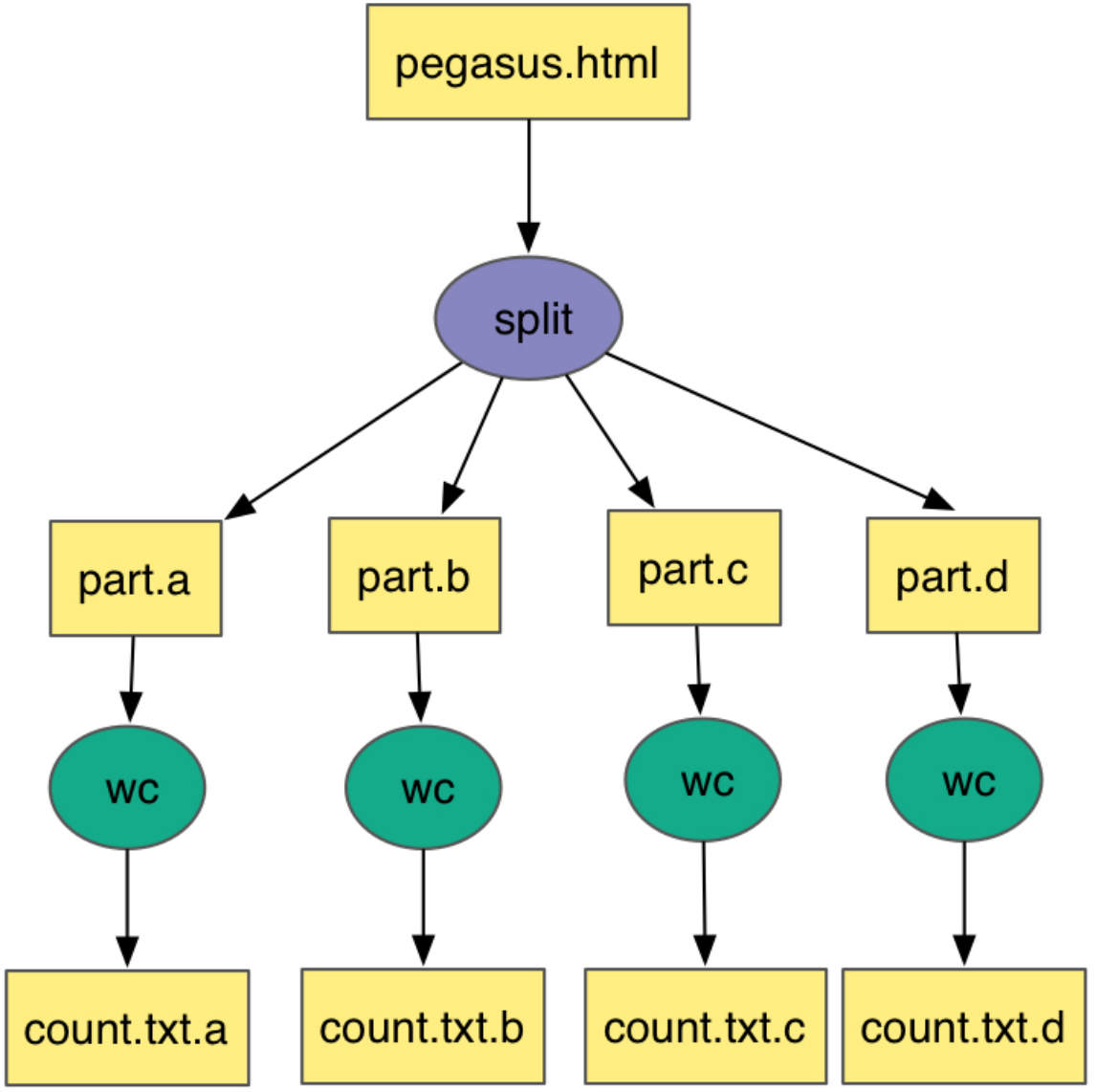

Interfaces
APIs
python
Java
perl
hubzero
Other workflow composition tools: Wings
Monitoring
Users
Pegasus WMS
Submit Host
Mapper
Engine
Scheduler
Monitoring & Provenance
Logs
Notifications
Workflow DB
Job Queue
Clouds
Cloudware
OpenStack, Eucalyptus, Nimbus
Compute
Amazon EC2, Google Cloud, RackSpace, Chameleon
Storage
Amazon S3, Google Cloud Storage, OpenStack
Distributed Resources
Campus Clusters, Local Clusters, Open Science Grid, XSEDE
HTCondor / GRAM
PBS
LSF
SGE
Middleware
Compute
GridFTP
HTTP
FTP
SRM
IRODS
SCP
Storage

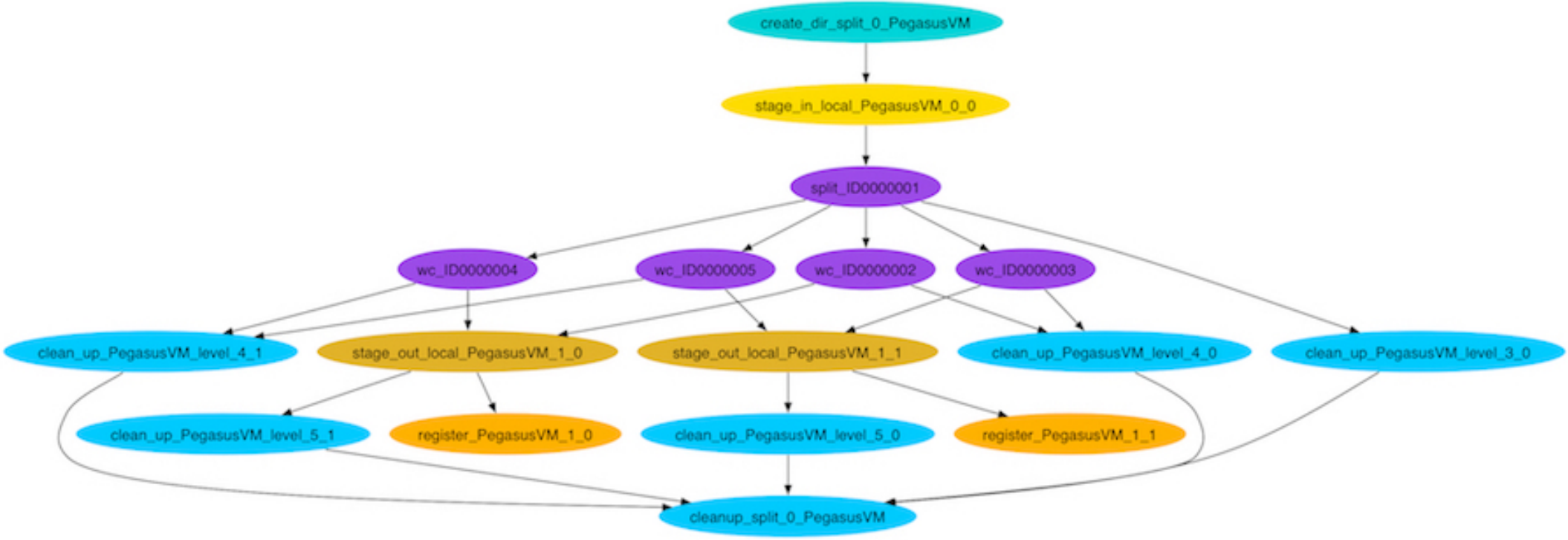
- Gallery of workflows: https://pegasus.isi.edu/workflow_gallery/
- Virtual box image (tutorial):

  demo...

```
[tutorial@localhost split]$ history
    1  ll
    2  mkdir test1
    3  cd test1
    4  pegasus-init split
    5  ll
    6  ll ./split
    7  cd split/
    8  ./generate_dax.sh split.dax
    9  ls -lt
   10  ./plan_dax.sh split.dax
   11  codor_status
   12  condor_status
   13  hostname
```




pegasus.html
split
part.a
part.b
part.c
part.d
wc
wc
wc
wc
count.txt.a
count.txt.b
count.txt.c
count.txt.d


create_dir_split_0_PegasusVM
stage_in_local_PegasusVM_0_0
split_ID0000001
wc_ID0000004
wc_ID0000005
wc_ID0000002
wc_ID0000003
clean_up_PegasusVM_level_4_1
stage_out_local_PegasusVM_1_0
stage_out_local_PegasusVM_1_1
clean_up_PegasusVM_level_4_0
clean_up_PegasusVM_level_3_0
clean_up_PegasusVM_level_5_1
register_PegasusVM_1_0
clean_up_PegasusVM_level_5_0
register_PegasusVM_1_1
cleanup_split_0_PegasusVM


Pegasus
Workflow Management
System