Gnosis Research Center

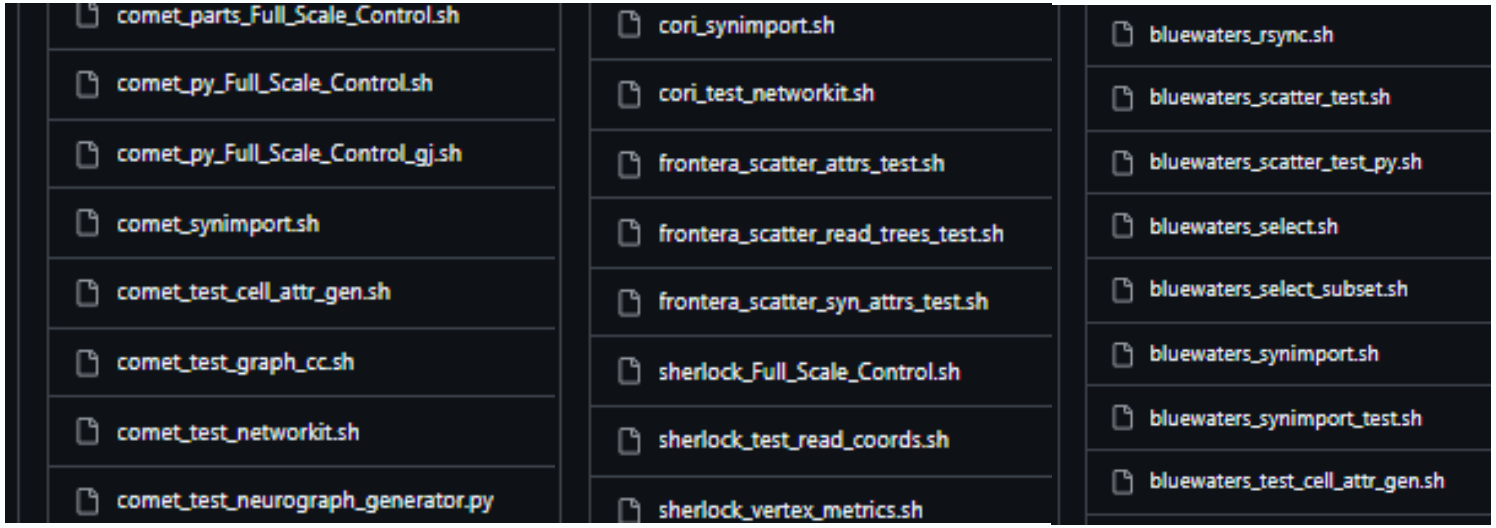
ILLINOIS INSTITUTE OF TECHNOLOGY

# Jarvis: Towards a Shared, User-Friendly, and Reproducible, I/O Infrastructure.

# Running scientific code is a pain

| Complex Parameter Space | Machine-Specific Configurations | Complex, Unstandardized Deployments |
|---|---|---|
| Applications and their dependencies often have many parameters that require expertise to configure<br><br>(e.g., HDF5 has thousands of configurable optimizations) | Applications often require specific knowledge of the machine and its environment to run<br><br>(e.g., network parameters, storage locations, etc.) | Experiments are often divided into many specialized scripts designed for particular machines and their environments<br><br>(e.g., the repo with a million bash scripts) |

Can't we make reproducibility easier than this?



# Jarvis

- Jarvis aims to be a deployment definition and executor cli/python library.
  - It enforces documentation of parameter.
  - Provides mechanism to make scripts hardware adaptable.
  - Leverages Python with support for HPC-specific interfaces (e.g. MPIexec).

# Jarvis Packages

- In Jarvis, we use “packages".
- Jarvis has three general pkg types:
  - Service: runs forever, until stopped.
  - Application: runs to a definite completion.
  - Interceptor: Used to intercept code (LD_Preload)
- Jarvis includes extensive utilities for handling program execution. This includes things like:
  - Executing MPI and PSSH commands in Python.
  - Hostfile and configuration file management
  - Wrappers around common bash commands.

```
def configure(self, **kwargs)
def start(self):
def stop(self):
def clean(self):
```

```
MpiExecInfo(nprocs=self.config['nprocs'],
            ppn=self.config['ppn'],
            hostfile=self.jarvis.hostfile,
            env=self.mod_env))
```

# Resource Graph: a Hardware Definition

- A record of the hardware and its configuration on a given cluster.
- It is sharable and reusable.
- There is a *mildly successful* automatic resource graph generator.

```yaml
fs:
- avail: 500GB
  dev_type: ssd
  device: /dev/sdb1
  fs_type: xfs
  host: localhost
  model: Samsung SSD 860
  mount: /mnt/ssd/${USER}
  parent: /dev/sdb
  shared: false
  uuid: 45b6abb3-7786-4b68-95d0-a8fac92e0d70
- avail: 900GB
  dev_type: hdd
  device: /dev/sdc1
  fs_type: xfs
  host: localhost
  model: ST1000LM049-2GH1
  mount: /mnt/hdd/${USER}
  parent: /dev/sdc
  shared: false
  uuid: 7857cbad-2e46-40c2-835a-b297bc5ee1d2
```

```yaml
net:
- domain: lo
  fabric: 127.0.0.1/32
  host: localhost
  protocol: FI_PROTO_RXM
  provider: tcp;ofi_rxm
  shared: false
  speed: 42949672960
  type: FI_EP_RDM
  version: '114.10'
- domain: enp47s0np0
  fabric: 172.25.0.0/16
  host: localhost
  protocol: FI_PROTO_RXD
  provider: udp;ofi_rxd
  shared: true
  speed: 42949672960
  type: FI_EP_RDM
  version: '114.10'
```

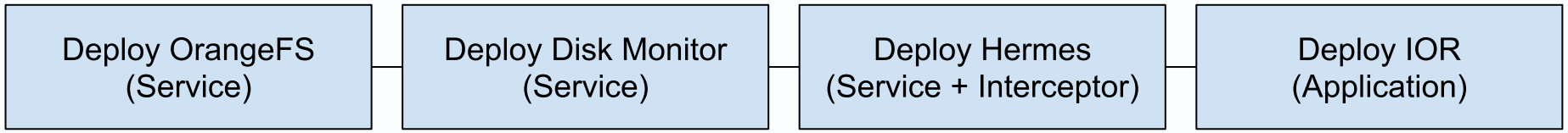
# Querying the Resource Graph

```
from jarvis_util import *
rg = ResourceGraph()
rg.find_storage(shared=True, condense=True)
```

- Build hardware-independent packages
- Answer questions like:
  - Are there local NVMes on the compute node?
  - Does the network support verbs?
  - What is the PFS mount point?

# Pipelines: composable storage deployments

- A pipeline specifies an ordered set of configured pkgs to execute.
- An example of a Jarvis pipeline would be as follows:




- Jarvis provides a CLI to create pipelines.
- Pipelines can then be executed, stopped, configured and managed.
- Pipelines hold individual configurations of each package and maintain a static record of the environment.
  - Pipelines are sharable.

# On-Going

# Community Survey

- We are looking to understand the expectations/requirements and hardware used by the storage community.
- Building a public cluster with diverse storage and accelerator hardware that can be managed with Jarvis.
- Running a survey, qr code plus pamphlets on the back.

# Jarvis status

- Continuous and on-going development on Jarvis.
- Small team, we welcome contributions to the core of Jarvis, but also packages.
- We are always happy to help anyone interested in using it or look at github issues.
- It has been used in multiple labs and university research clusters.

# Thank you!

Jarvis: https://github.com/grc-iit/jarvis-cd
Wiki: https://github.com/grc-iit/jarvis-cd/wiki

Contact: llogan@hawk.iit.edu and/or jcernudagarcia@hawk.iit.edu

# Lighting
# Quick Features

# Descriptive/Documented packages

```
(.venv) llogan@llogan-OMEN-by-HP-Gaming-Laptop-16-xf0xxx:~/Documents/Projects/jarvis-cd$ jarvis pkg help hermes_run
COMMAND: hermes_run   ...

Option Class:
Name             Default  Type    Description
-------------  ---------  ------  ------------------------------------------------------------------------------------
log_verbosity          1  int     Verbosity of the output, 0 for fatal, 1 for info
sleep                  0  int     How much time to sleep during start (seconds)
reinit             False  bool    Destroy previous configuration and rebuild
do_dbg             False  bool    Enable or disable debugging
dbg_port            4000  int     The port to use for debugging
stdout                    str     The file to use for holding output. Use stderr topipe to the same file as stderr.
stderr                    str     The file to use for holding error output. Use stdout to pipe to the same file as stdout.
hide_output        False  bool    Hide output of the runtime.
h,help             False  bool    Print help menu
```

```
Option Class: adapter
Name           Default    Type     Description
-------------  ---------  -------  ------------------------------------
i,include      []         ['str']  Specify paths to include
e,exclude      []         ['str']  Specify paths to exclude
adapter_mode   default    str      The adapter mode to use for Hermes
flush_mode     async      str      The flushing mode to use for adapters
page_size      1m         str      The page size to use for adapters

Option Class: buffer organizer
Name             Default  Type    Description
-------------  ---------  ------  --------------------------------------------------
recency_max            1  float   time before blob is considered stale (sec)
borg_min_cap           0  float   Capacity percentage before reorganizing can begin
flush_period        5000  int     Period of time to check for flushing (milliseconds)
```

```
{
  'name': 'nprocs',
  'msg': 'Number of processes to spawn',
  'type': int,
  'default': 4,
},
{
  'name': 'ppn',
  'msg': 'Processes per node',
  'type': int,
  'default': 16,
},
{
  'name': 'L',
  'msg': 'Grid size of cube',
  'type': int,
  'default': 32,
}
```

# Custom Repositories

```
my_org_name
└── my_org_name
    └── orangefs
        └── package.py
jarvis repo add /path/to/my_org_name
```

# Configuration management

```
<io name="SimulationOutput">
       <engine type="Plugin">
           <parameter key="PluginName" value="hermes" />
           <parameter key="ppn" value='##PPN##'/>
           <parameter key="VarFile" value="##VARFILE##"/>
           <parameter key="OPFile" value="##OPFILE##"/>
           <parameter key="db_file" value="##DBFILE##"/>
       </engine>
   </io>
```

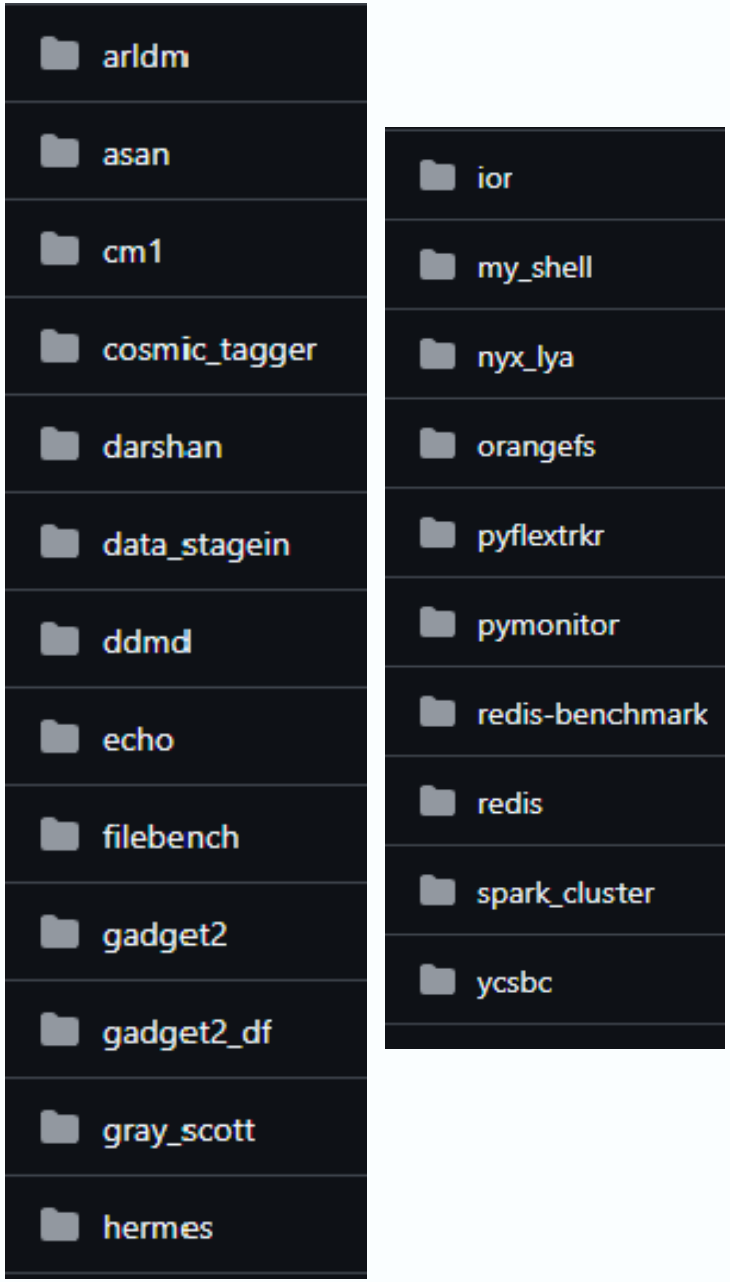
```
self.copy_template_file(
  f'{self.pkg_dir}/config/hermes.xml',
  self.adios2_xml_path,
  replacements={
    'PPN': self.config['ppn'],
    'VARFILE': self.var_json_path,
    'OPFILE': self.operator_json_path,
    'DBFILE': self.config['db_path'],
  }
)
```

# Python interface

```
from jarvis_cd.basic.pkg import Pipeline
pipeline = Pipeline().create(pipeline_id).build_env().save()
pipeline = Pipeline().load(pipeline_id=None)
pipeline.append(pkg_type, pkg_id=None, do_configure=True, **kwargs)
```

```
pkg = pipline.get_pkg('hermes')
for i in range(5):
  pkg.configure(n_procs=i*20).save()
  pipeline.run()
```

# Built-in Packages




- Applications, workflow, benchmarks
- Storage systems (Hermes, redis, orangeFS)
- Compute systems (spark)
- Support packages (darshan, asan)

# Minor things

```
1  jarvis pipeline sbatch job_name=test nnodes=4
2  jarvis pipeline pbs nnodes=2 system=other_system
```

```
do_dbg                False  bool     Enable or disable debugging
dbg_port               4000  int      The port to use for debugging
```