Education and Research Computing supported by DevOps DSL@ISE@IU

Gregor von Laszewski
(laszewski@gmail.com)
Fugang Wang
Allan Streib
Geoffrey Fox

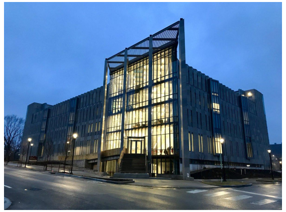
Ideas - Observations

- Support of a variety of users
- Research flexibility is not captured just by a data/compute center where we often find outdated software
- Templated state-of-the-art images to the rescue
- Containers for uniform execution environments
- Client tools such as **cloudmesh client** enhance DevOps experience

Intelligent Systems Engineering Department @ IU

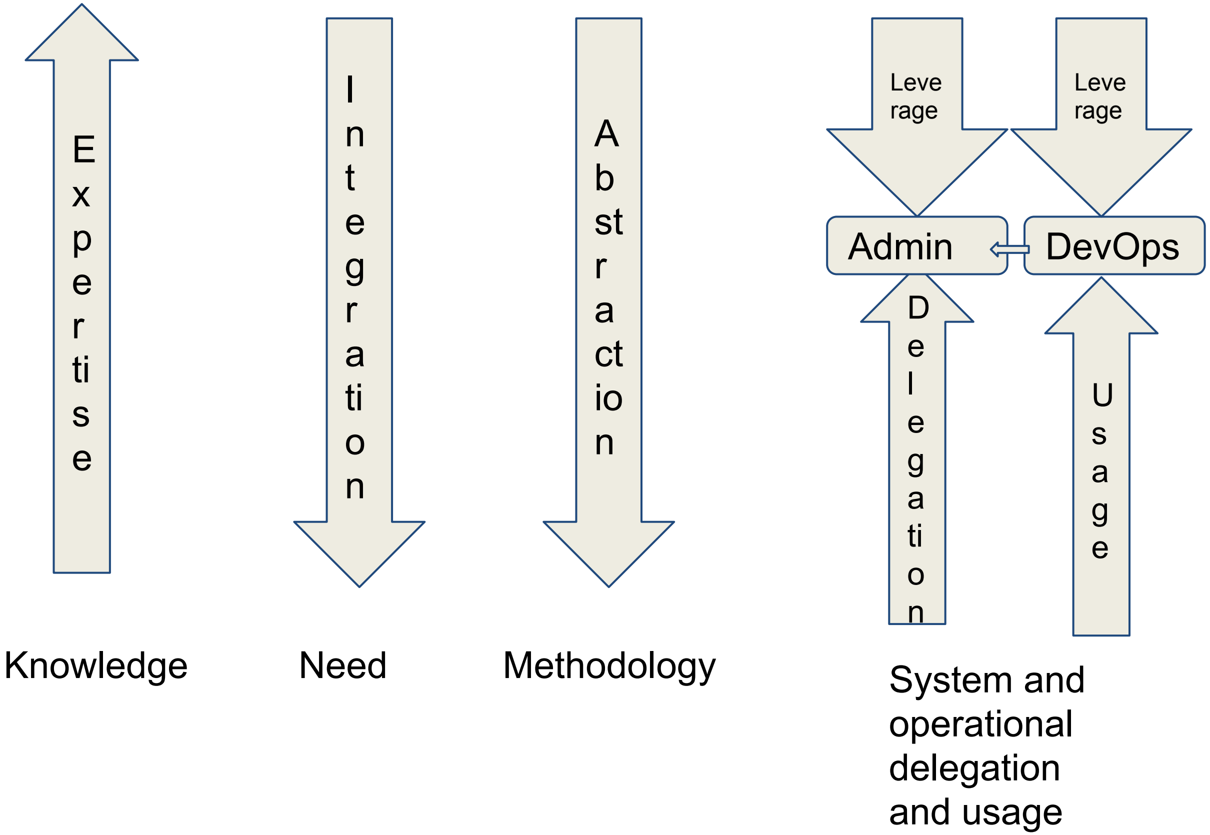
- ISE
  a. Interdisciplinary
  b. Collaborative Research
  c. Rich set of disciplines
     i. Bioengineering
     ii. Computer Engineering
     iii. Cyber-physical Systems
     iv. Environmental Engineering
     v. Intelligent Systems
     vi. Molecular and Nanoscale Engineering
     vii. Neuro-engineering
  d. Innovation with “**intelligent systems**”

- Topics
  a. Hardware & Software
  b. Sensor systems & Signal Processing
  c. High Performance Simulation
  d. Medical Devices
  e. Living Organisms
- Design Centered Approach
- Hands on
- Research Oriented
- Interfacing with other disciplines



Observations about Users

- Admin Community
- Admin
- Developer
- Student
- Researcher

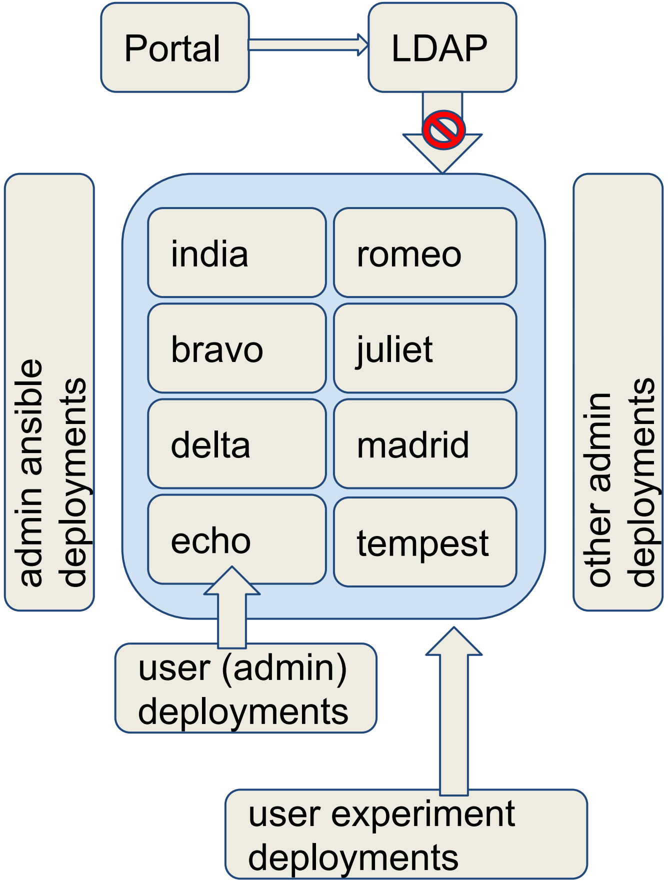


Systems Supporting Research (Staff=1 + .25)

Name	System Type	Use	# Nodes	# CPUs	# Cores	RAM (GB)	Storage (TB)
bravo	HP Proliant	Storage (112 TB Beegfs/IB)	16	32	128	3072	128
delta	SuperMicro GPU Cluster	GPU	16	32	192	1333	144
echo	SuperMicro Cluster	Containers (Kubernetes, Docker Swarm)	16	32	192	6144	192
juliet	SuperMicro HPC Cluster	MPI, MapReduce	128	256	3456	16384	1024 HDD; 50 SSD
romeo	SuperMicro cluster	GPU (K80, Volta)/Deep Learning	6	12	136	768	48 HDD; 2.4 SSD
tango	Penguin/Intel Xeon Phi/Omnipath	MPI, Applications, Distributed Systems Machine Learning/Data Analytics	64	64	4416	12800	205 HDD; 51 SSD
tempest	SuperMicro HPC Cluster/Omnipath	Applications, MPI, Distributed Systems	10	20	480	2560	25 HDD; 4 SSD
victor	SuperMicro HPC Cluster	Clouds, Containers	16	32	768	4096	128 HDD; 6.4 SSD

Research Compute Resources

- Deployment of Hosts
  - Software stack and system configuration using configuration management tools (Ansible)
    - Used for openstack cloud deployment
    - Used for container system deployment (docker swarm; kubernetes)
  - Kickstart/preseed OS installation (Centos; Ubuntu)
- Operation
  - Streamlined account/project application and setup starting from a web portal (SaaS approach)
  - Continuous delivery of user documentation generated from document source
  - User deployments, Long running experiment deployments



Summary: System Admin Usage

- Initial provisioning of new cluster nodes
  - PXE/Kickstart for RHEL installation
  - Ansible for OS configuration and software provisioning
    - Consistent, repeatable, Self-documenting configuration
  - Kickstart and Playbooks in github
- Recovery or upgrade of production nodes
  - Easily recover after disk failures or other problems.
  - Deploy updates and security patches automatically and consistently.

- Account provisioning
  - Automatic via portal after manual approval.
  - Automatic LDAP group management for permissions and access control.
  - Automatic SSH key management via user web portal and LDAP

Types of Resources

- DOE: Leadership class resources for most demanding applications
- XSEDE: large scale - medium scale resources using common software stack
- ISE/department
  - **<u>Experimental systems</u>** with **<u>newest</u>** software
    - typically not provided by Research Computing
  - Experiments that are dedicated to an application
  - Preparation for other systems
  - Education
  - **<u>Full control</u>** of the system

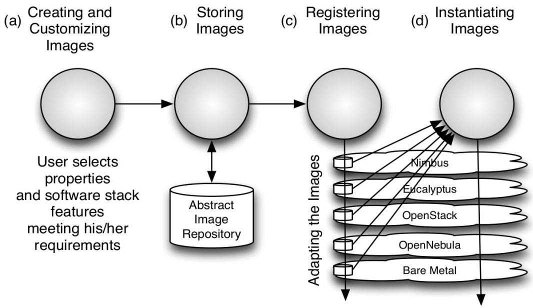
=> We need all three

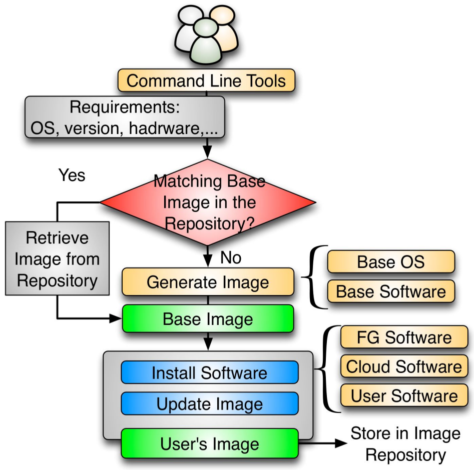
Rain: Results documented in series of papers

[R1]	Design of the FutureGrid Experiment Management Framework Gregor von Laszewski, Geoffrey C. Fox, Fugang Wang, Andrew J. Younge, Archit Kulshrestha, Gregory G. Pike, Warren Smith, Jens Voeckler, Renato J. Figueiredo, Jose Fortes et al. doi> 10.1109/GCE.2010.5676126
[R2]	Design of a Dynamic Provisioning System for a Federated Cloud and Bare-metal Environment, Gregor von Laszewski, Hyungro Lee, Javier Diaz, Fugang Wang, Koji Tanaka, Shubhada Karavinkoppa, Geoffrey C. Fox, and Tom Furlani, Proceeding FederatedClouds ‘12 Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit. doi> 10.1145/2378975.2378982
[R3]	Abstract Image Management and Universal Image Registration for Cloud and HPC Infrastructures, J. Diaz, Gregor von Laszewski, F. Wang and G. Fox. IEEE Cloud 2012, Honolulu, Hawaii, June 2012. doi> 10.1109/CLOUD.2012.94
[R4]	Comparison of Multiple Cloud Frameworks, Gregor von Laszewski, J. Diaz, F. Wang, and G. Fox. IEEE Cloud 2012, Honolulu, Hawaii, June 2012. doi> 10.1109/CLOUD.2012.104
[R5]	FutureGrid Image Repository: A Generic Catalog and Storage System for Heterogeneous Virtual Machine Images, J. Diaz, Gregor von Laszewski, F. Wang, A.J. Younge, and G. Fox, 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom2011), Athens, Greece, November 2011. doi> 10.1109/CloudCom.2011.85

Rain: Templated Images

- It was sufficient to generate templated images for different architectures
- We used DevOps to generate them
- Templates allow maintainability
  - (Fix the template and run anywhere)
- Baremetal deployments were heavily used





Rain

- Reprovisioned cluster or parts of the cluster to
  - MPI mode
  - OpenStack Mode
  - Hadoop Mode
- Concept of virtual clusters was highly useful

Today: NSF SDSC Comet

- Comet Large NSF cluster as part of XSEDE
- Virtual Cluster
  - based on Rocks
  - using REST interfaces to manage clusters
  - using **cloudmesh client** to easily interface
  - full control of the cluster by the user
  - there can be multiple clusters on the same bare metal machine
- cloudmesh: create me a cluster with 30 servers
  - Than users can essentially do what they want to do
  - e.g. cluster is “owned” by user
  - dynamical resource use (grow, shrink, suspend)
  - http://cloudmesh-client.readthedocs.io/en/latest/commands/command_comet.html#comet-command

Reminder: Why we use DevOps?

- Development:
  - Fast prototype and incremental development
- Test/QA
  - Continuous integration from distributed team members checkins
- Operation
  - Reproducible base system
  - Configuration management

DevOps: Lessons learned

- Reproducible Development environment
  - **Vanilla** OS in cloud or other VMs
  - **Configuration management** (e.g., **ansible**) for system setup and configuration
    - we used previously also chef and puppet
  - **Virtual software development environment**
    - in python we use virtualenv & pyenv
  - **Version control** (public github; IU GIT)
  - **Continuous Integration & Quality Assurance**
    - **travis, jenkins**
    - **system testing, application testing**

Observation

This is not just about ansible, chef, puppet, cfengine, …

It is how to leverage these tools including virtualized software environments such as in the case of python.

Why we use ansible?

- No contributions from students when we used chef and puppet
  – This is not a surprise: At university we teach python, ansible is done in python
- We used to find more stable “templates” in ansible than others.
  – (this may have changed)
- We use it to deploy container infrastructure

Evolving Focus

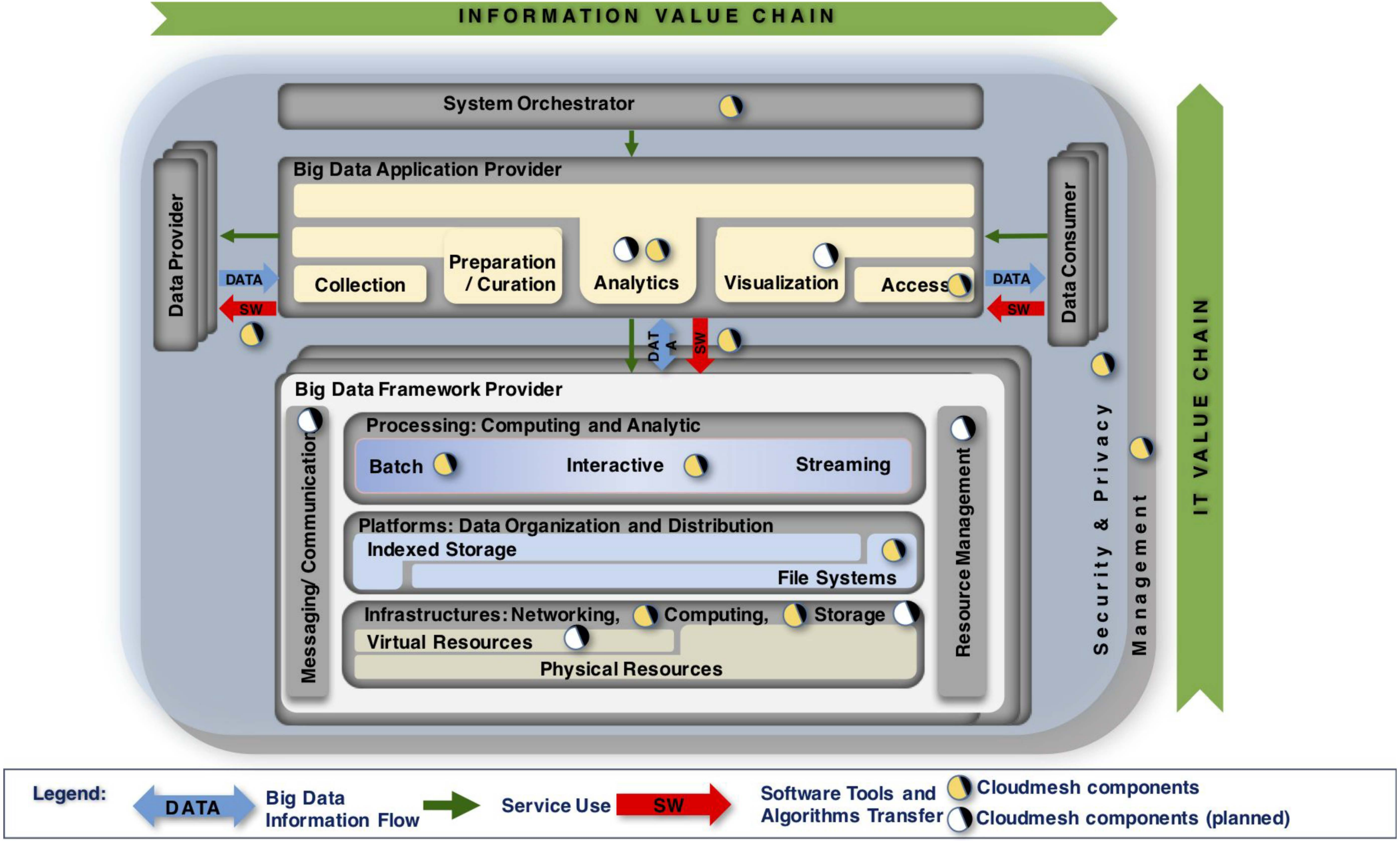
- Old Focus: OpenStack with DevOps
  - ansible scripts
- New focus: Use community resources for virtual machines
  - now we use **community resources** due to high demand on administration
  - with **cloudmesh client we can switch easily between virtual machine providers**
    - `cm cloud=chameleon; cm vm start`
      - vm defaults are defined on a user basis
    - `cm cloud=aws; cm vm start`

Evolving Focus

- New Focus: containers
  - This includes lxc, deployments of docker swarm and kubernetes supported by devops
- We still use templated images via Dockerfiles
- We can deploy them on OSX, Windows, Linux
  - developing clients becomes easier
  - services are supported and easier to deploy
  - Dockerfiles are easier than ansible (different focus)

DevOps in support of NIST Service Abstractions

- Working with NIST to define a BigData Reference Architecture
- Goals
  - Vendor independent
  - Useful Abstractions
  - Pluggable services
- OpenAPI to define REST services



INFORMATION VALUE CHAIN
System Orchestrator
Big Data Application Provider
Collection
Preparation / Curation
Analytics
Visualization
Access
Data Provider
Data Consumer
DATA
SW
DATA
SW
DATA
SW
Big Data Framework Provider
Messaging/ Communication
Processing: Computing and Analytic
Batch
Interactive
Streaming
Platforms: Data Organization and Distribution
Indexed Storage
File Systems
Infrastructures: Networking, Computing, Storage
Virtual Resources
Physical Resources
Resource Management
Security & Privacy
Management
IT VALUE CHAIN
Legend:
DATA
Big Data Information Flow
Service Use
SW
Software Tools and Algorithms Transfer
Cloudmesh components
Cloudmesh components (planned)

Cloudmesh in support of NIST

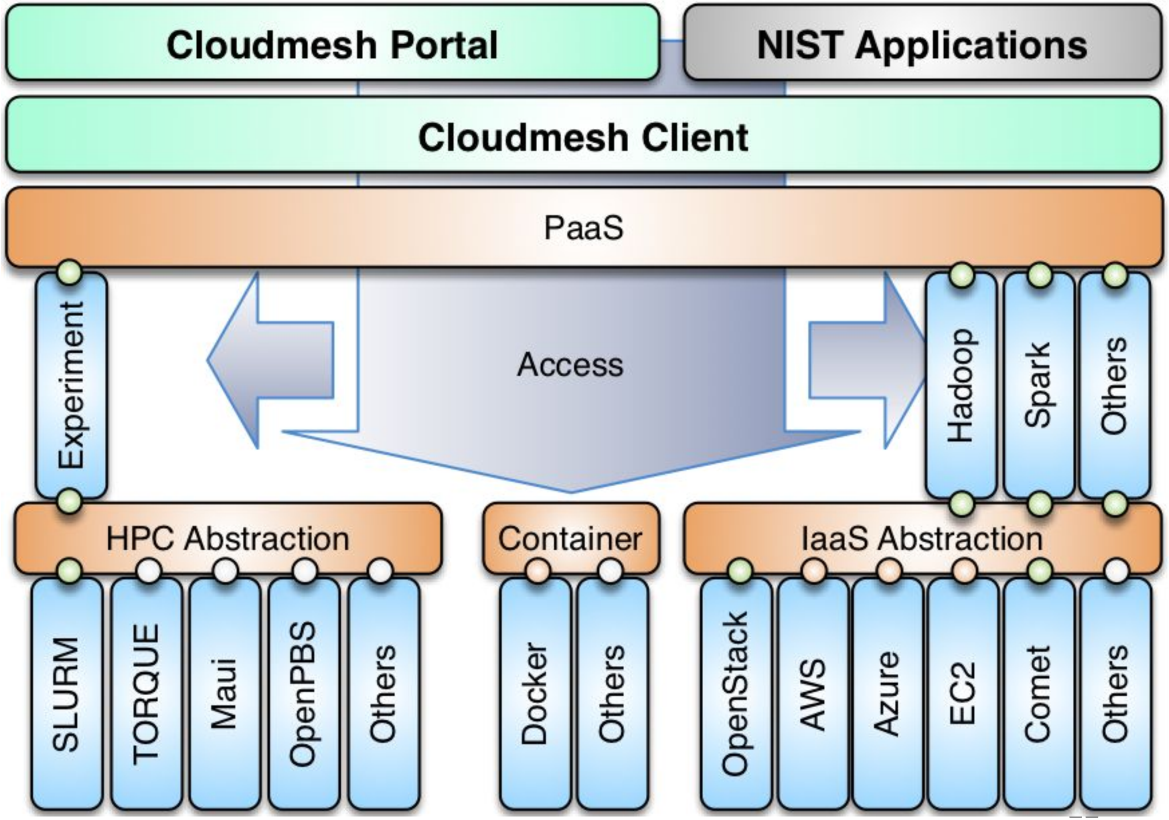
- Goal Working on Spec:
  - Volume 8: Big Data Reference Architecture
- Goal Cloudmesh:
  - Derive from spec service abstractions
  - Implement prototype
  - Deploy via DevOps
  - Test services
  - Accept contributions from community that contain deployment specs using DevOps

Cloudmesh Architecture and Use

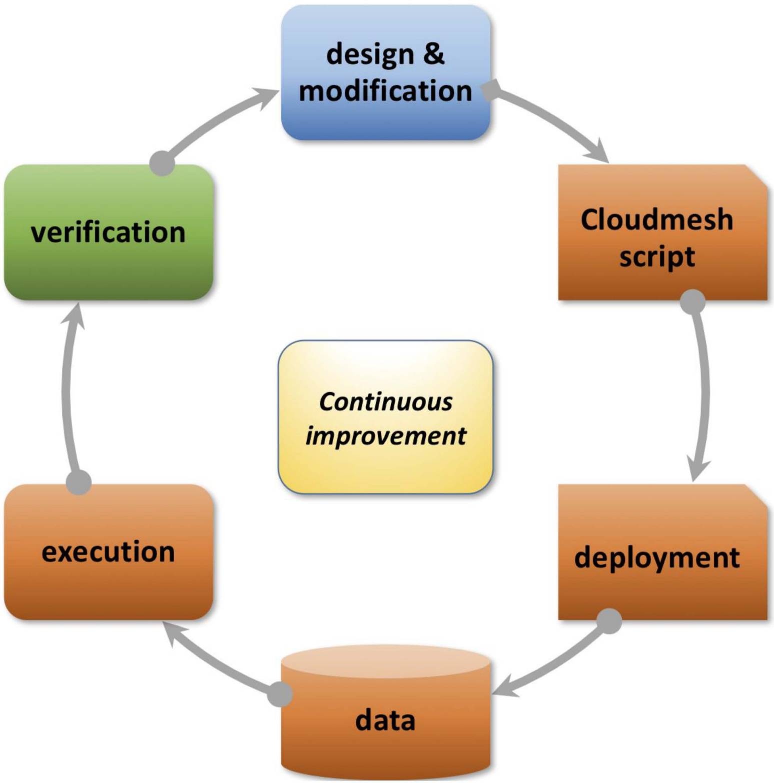
- Deployment
- Access
- Use

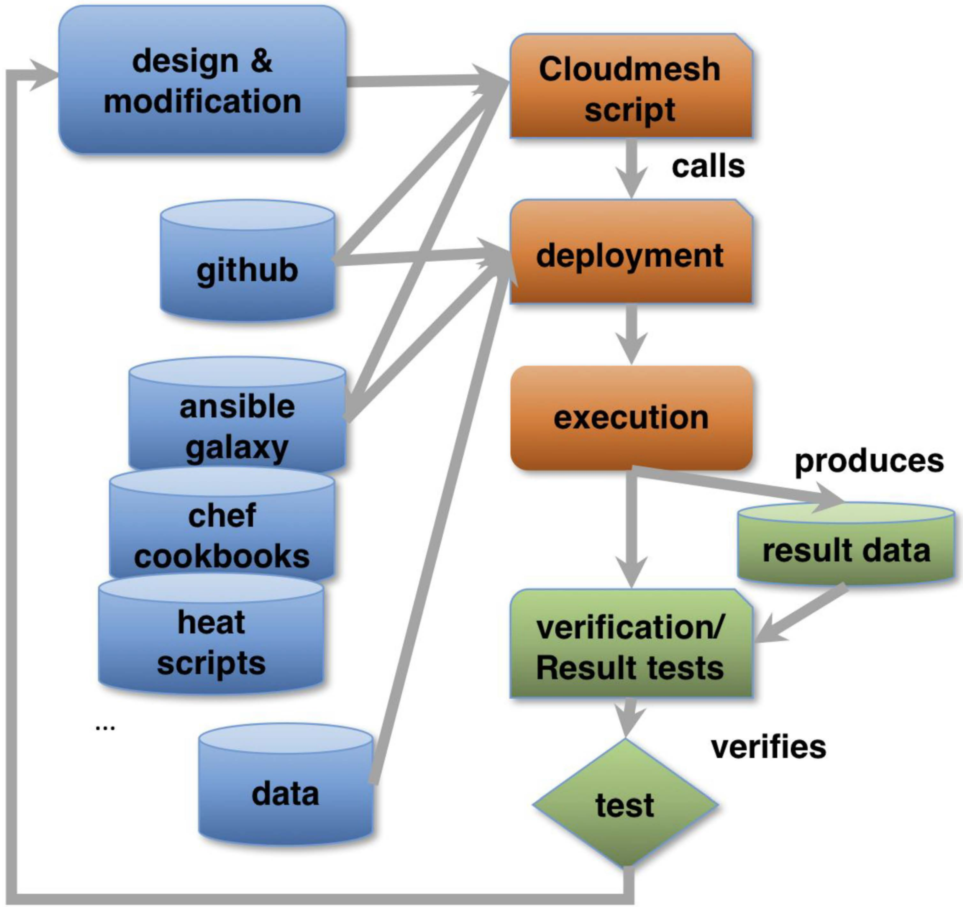
Goal:

```
cm --n 20
   --service hadoop
   --host echo
```



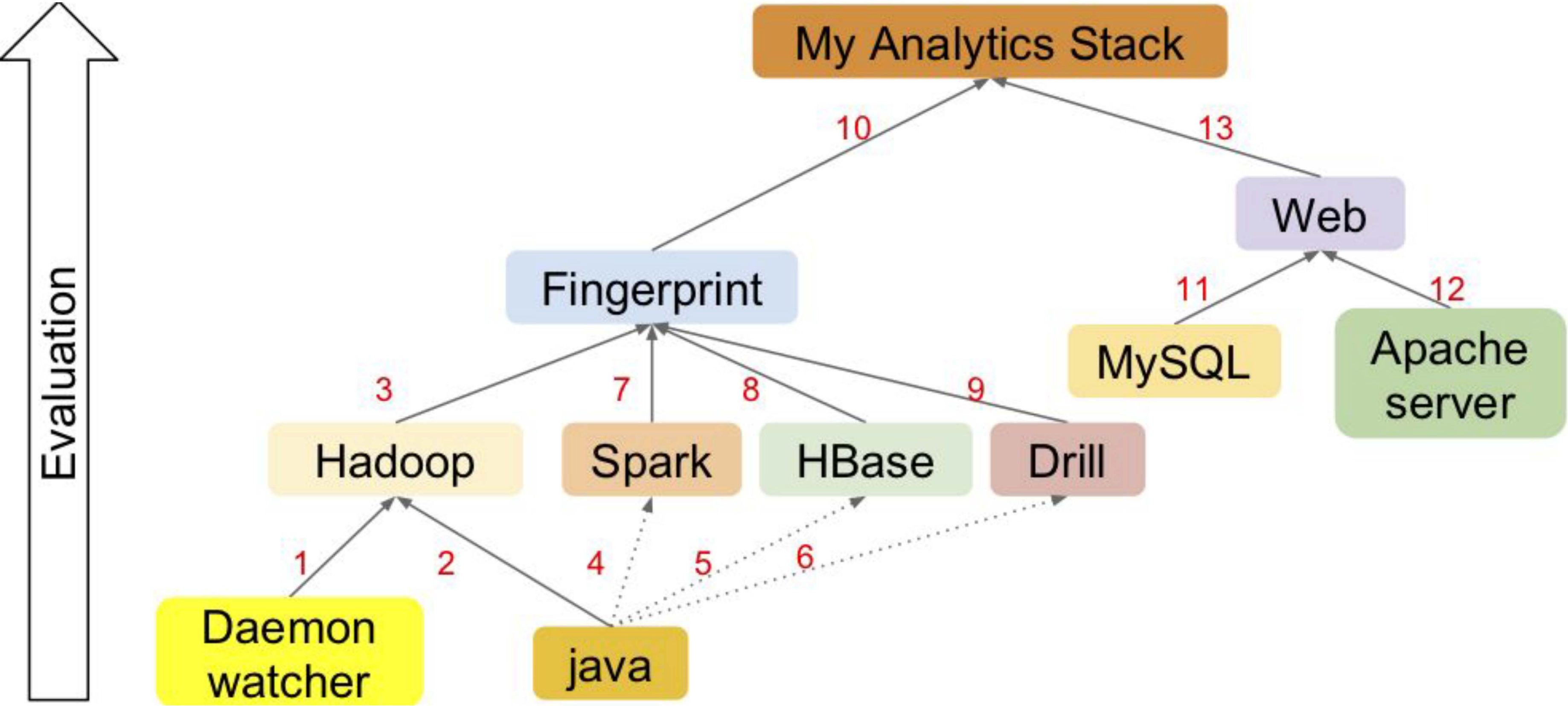
Continuous Improvement




design &
modification
Cloudmesh
script
calls
github
deployment
ansible
galaxy
execution
produces
chef
cookbooks
result data
heat
scripts
verification/
Result tests
...
verifies
data
test

Nist Usage Scenarios

- Implement usage scenarios for different disciplines while using deployment support with DevOps




```
cm deploy --n 20 --service fingerprint
```

NIST References

- https://github.com/cloudmesh-community/nist/tree/master/services
- https://github.com/cloudmesh-community/nist/blob/master/docs/dest/nistvol8-2.docx?raw=true
- https://laszewski.github.io/papers/vonLaszewski-nist.pdf

Summary

- DevOps supports users to configure sophisticated environments
- Templated images from vanilla image is supported by various DevOps Tools
- Testing applications is supported by DevOps
- DevOps is more than ansible, chef, puppet
- It is not sufficient to have a version of x in the data center, as that version is likely outdated
- We are reimplementing cloudmesh based on NIST experience.
  - add more abstractions
- Abstractions are needed to make it super simple. Cloudmesh to the rescue :-)

The Networking and Information Technology Research and Development (NITRD) Program

Mailing Address: NCO/NITRD, 2415 Eisenhower Avenue, Alexandria, VA 22314

Physical Address: 490 L'Enfant Plaza SW, Suite 8001, Washington, DC 20024, USA Tel: 202-459-9674, Fax: 202-459-9673, Email: nco@nitrd.gov, Website: https://www.nitrd.gov