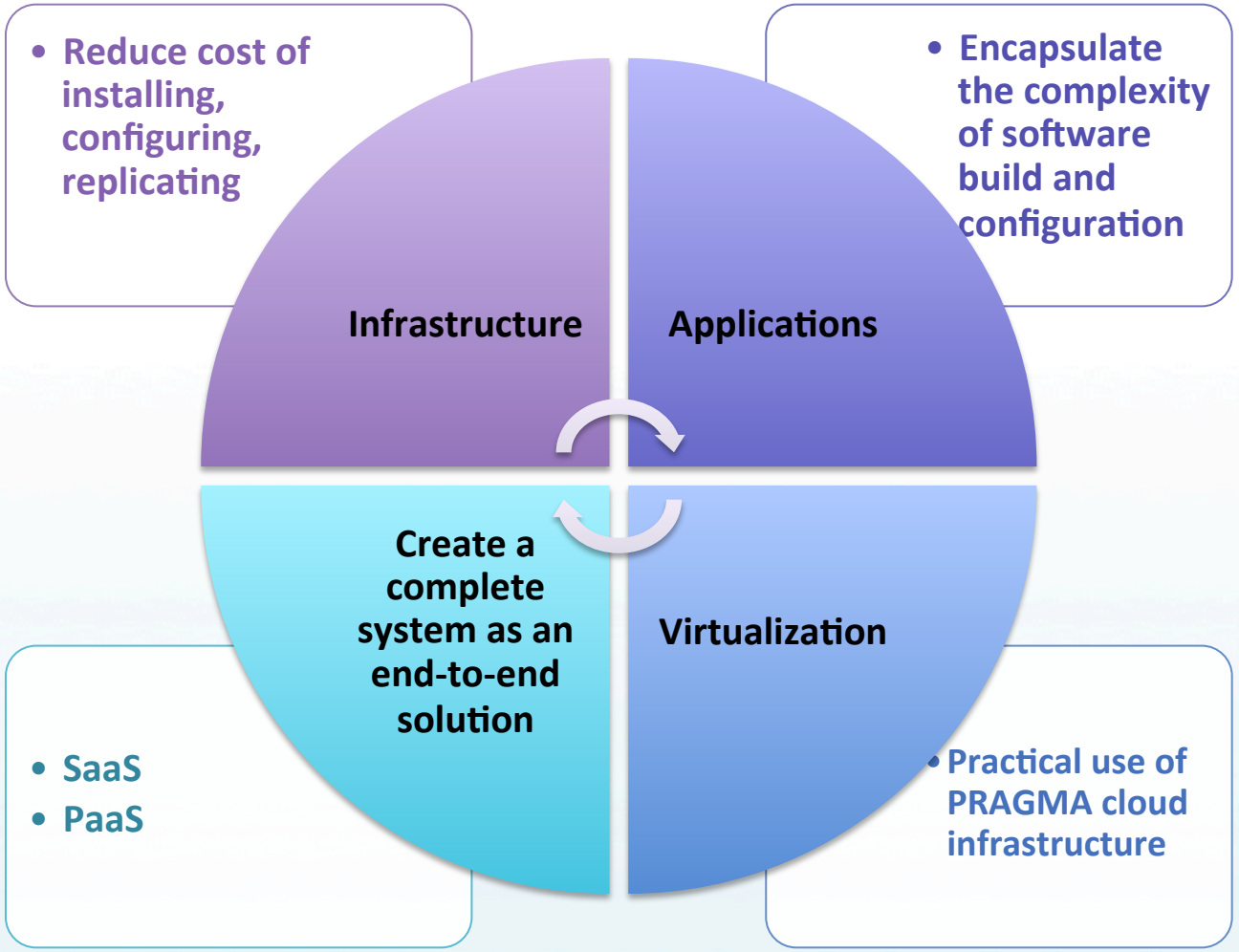
Applications and systems integration

Nadya Williams (UCSD) nadya@sdsc.edu

What we are trying to

- **Increase availability and flexibility of Software as a complete system**
  - reduce cost of installing/configuring/ replicating
  - ease burden of integrating hardware and software
- **Enable a fast "workflow" from software update to server availability:**
  - Minimize time spent on software build and configuration
  - Automate most hands-on tasks.
  - Essential: have test cases for all installed components and their configuration



Challenges

Server provisioning

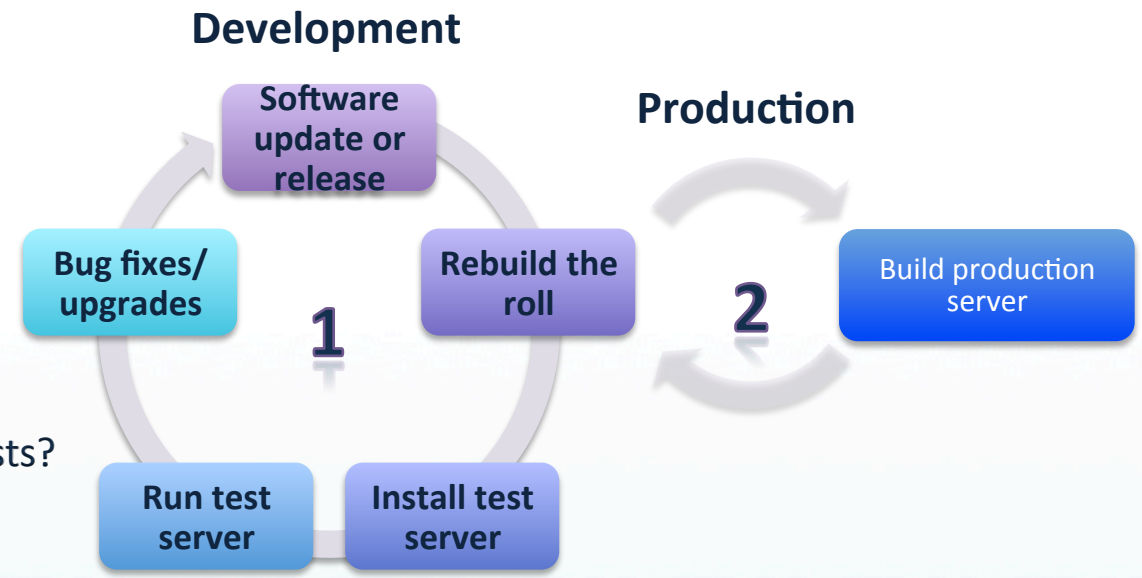
- automate, optimize

Managing, scaling and automating cloud applications

- Migrating from physical to virtual host
- Application packaging and distribution
- Virtual resource control: allocate memory, disk
- Environment is different during build and run time

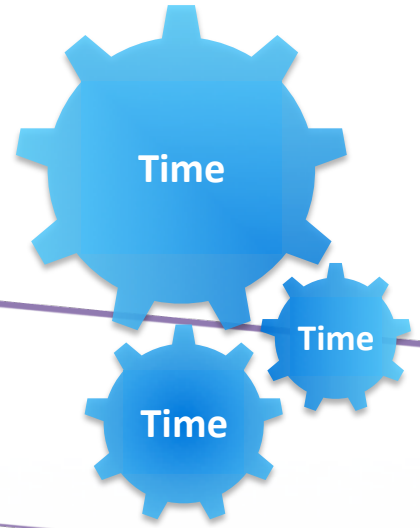
Unify physical and virtual

- Can we merge applications management for physical and virtual hosts?
- Can we build in reproducible way?
- How do we ensure that adequate virtual application performance is maintained ?
- How to tune performance
- How to troubleshoot problems



Anything done more than once need to be automated !

Identify *what* to automate and *when*
Which *tools* to use and *how*



Can not make any assumptions about the system

- Know your system real-time status: what is installed, what version, configuration
- Robust system: can reliably build and rebuild
- Enable operational efficiency and flexibility: deployment and operation of servers

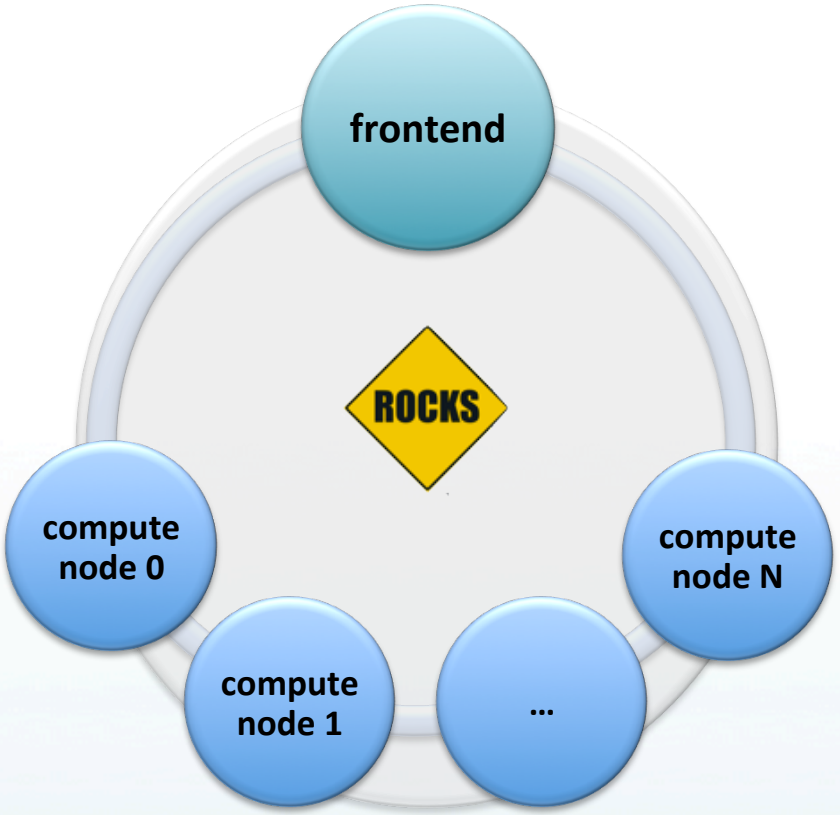
Automating operational, time consuming and repetitive tasks

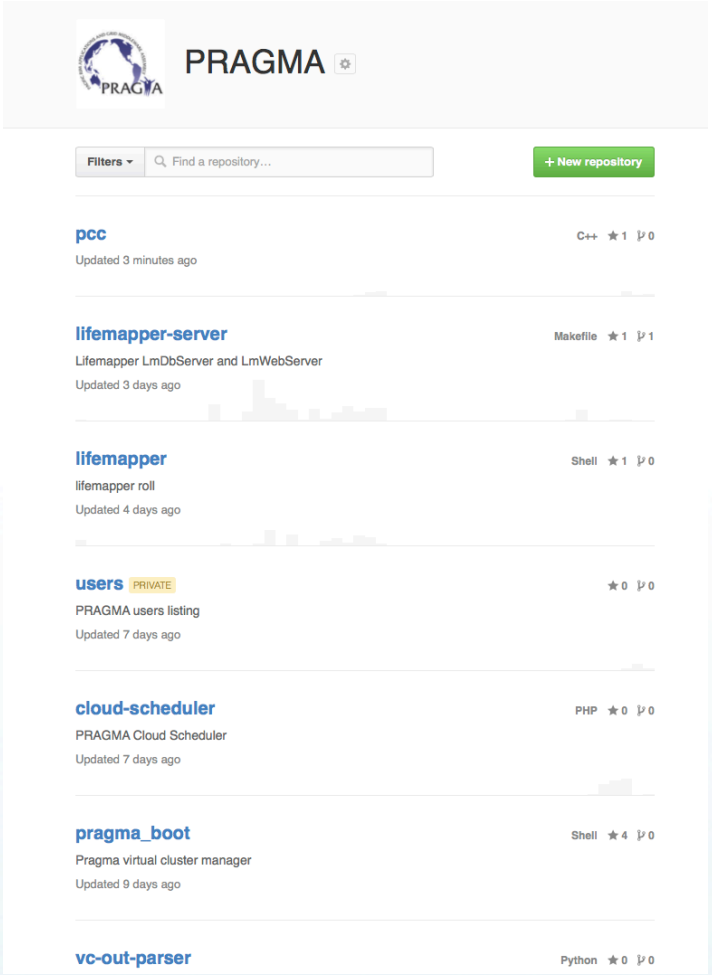
- Software build, configuration, customization
- Application "minimal resources" requirements
- Monitoring and troubleshooting

Automate service delivery - provide seamless integration of hw/sw

- Robust system: can reliably build and rebuild
- Cut build time from days to hours
- Provisioning in reproducible way - enables to clone multiple servers
- An absolute prerequisite for migration to the cloud

Cluster Overview

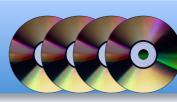




Cluster

- Create Frontend
- Create compute nods

Rolls

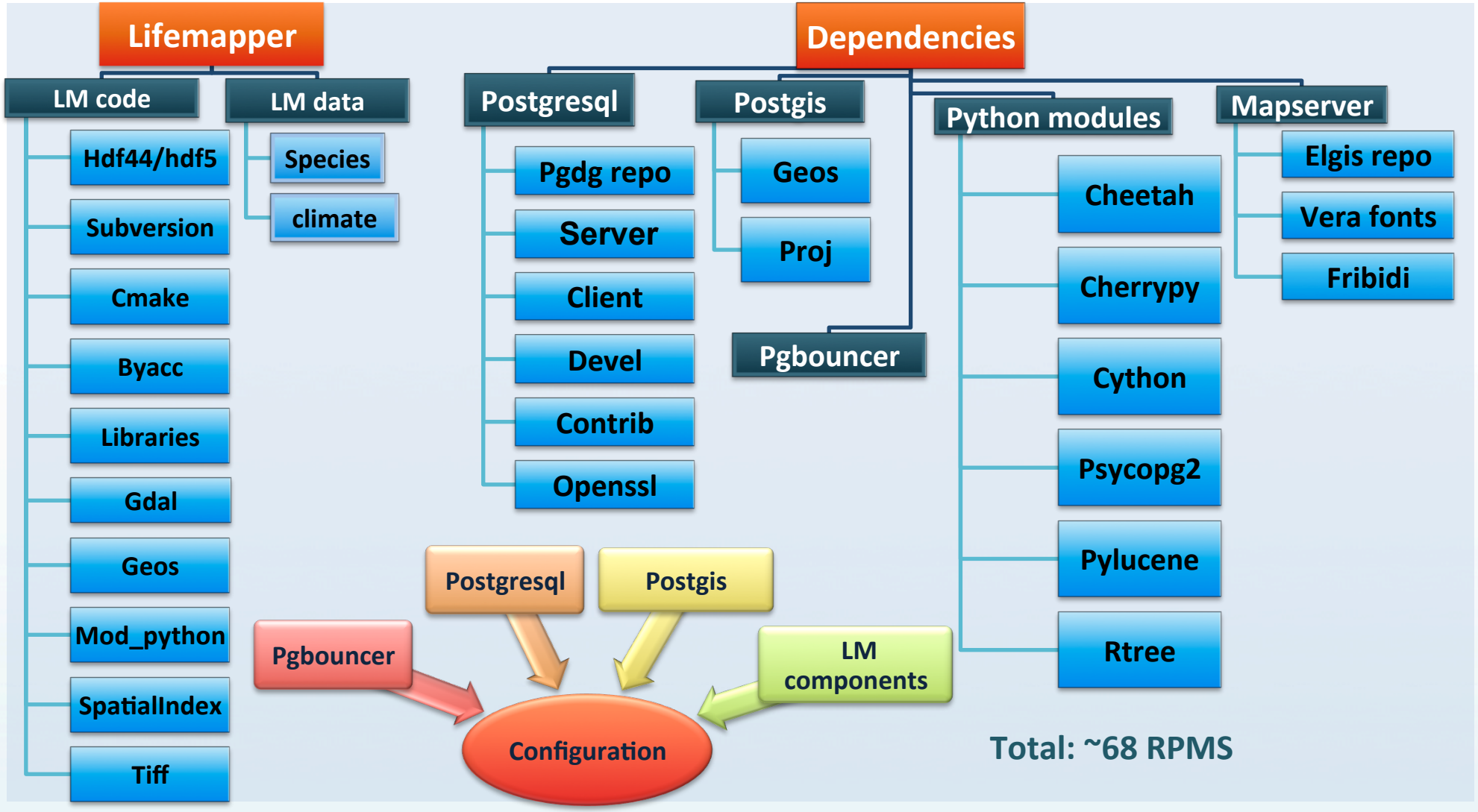


- kernel
- os
- hpc
- base
- ganglia
- sge
- web-server
- . . .

Hosting environment: KVM

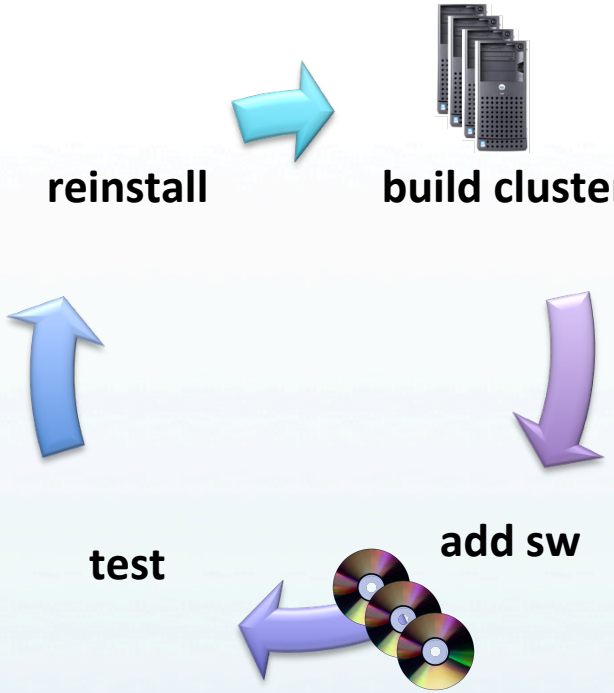
https://github.com/rocksclusters
https://github.com/pragmagrid

Roll Example: LifeMapper Server



PaaS and the Cloudscape

Virtual Cluster Deployment



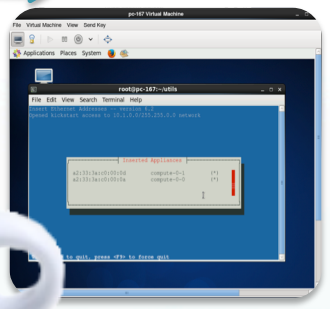
Advantages

1. Simplify development on a specific platform (ROCKS)
2. Minimize operational expense: from physical to virtual
3. Scalability: provide additional servers when needed
4. Portability: do once use many

Building Rocks Virtual Cluster

```
# rocks add cluster 67.58.51.168 1 fe-name=pc-168  \
        fe-container=pc-163 container-hosts=vm-container-0-9 \
        vlan=1051
# rocks start host vm pc-168
# virt-manager
```

```
# rocks start host vm pc-168
# insert-ethers
# rocks start host vm hosted-vm-0-0-0
```



Start install

Choose rolls

Cluster info

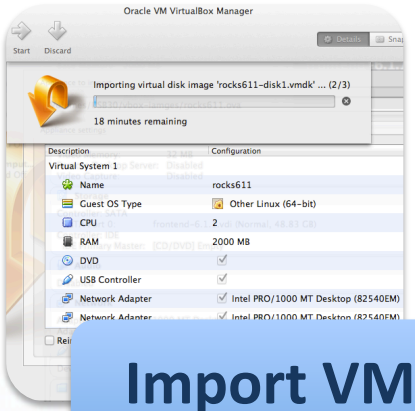
Root password

On virtual frontend

Frontend

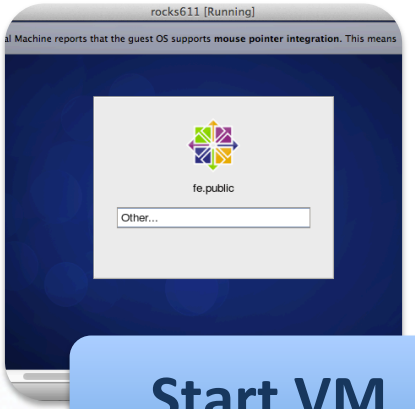
Compute nodes

Rocks Virtual Cluster: in VirtualBox on a laptop



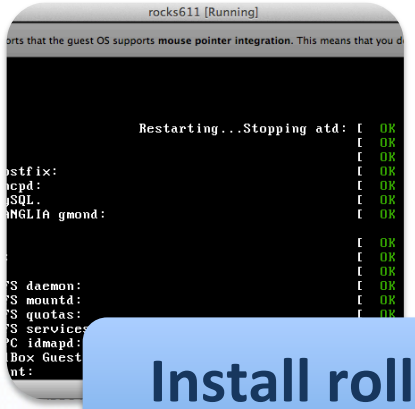
Import VM

- 5 min



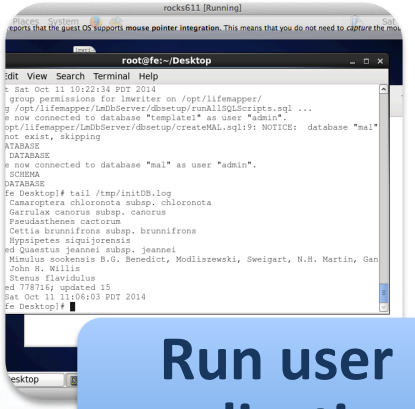
Start VM

- 3 min



Install roll

- 15 min



Run user application

Setup

Use

Steps:

1. Install VirtualBox, import VM, start VM
2. Install lifemapper-server roll
3. Use the image

Actions

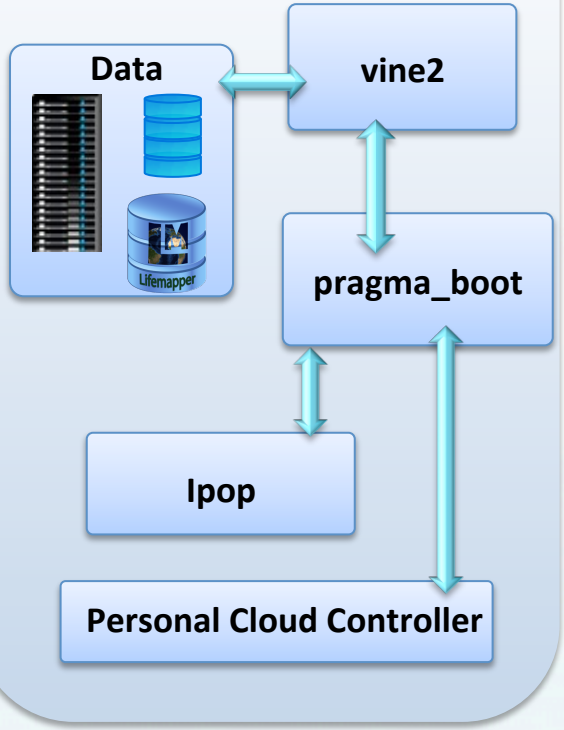
Automate Everything

- Installs
- configure
- Script repetitive tasks
- packages

Reuse

- Use software leverage
- Open source software
- Choose portability
- Small is good

Integrate



Summary - Corollary

A product that fails to make a meaningful impact on users ends up failing.

There must be a visible reward.

Solutions must be open, portable and interoperable without pushing users to replace their existing systems.

It is all about the application!

Acknowledgements

This work is funded in part by National Science Foundation and NASA grants

PRAGMA

US NSF 1234953

Rocks

US NSF OCI-1032778

US NSF OCI-0721623