Computer Monitoring with Prometheus & Grafana

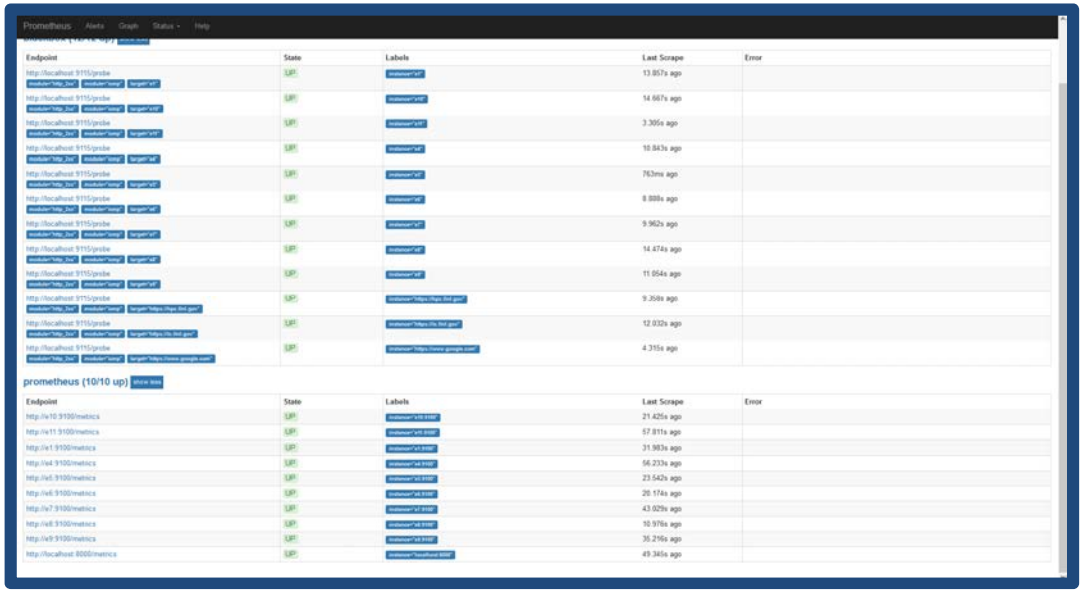
Brad Davis, Enikoe Bihari, Eliana Purcell
HPC Academy 2018

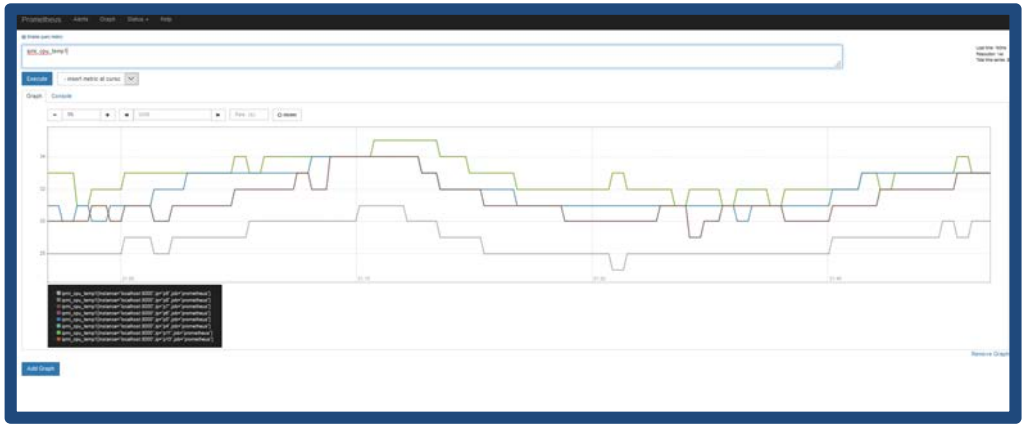
August 14, 2018

LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Prometheus

- Computer monitoring database software that stores data received from various exporters





Prometheus Configuration

```
global:
  scrape_interval: 15s

rule_files:
  - alert.rules.yml

alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093

scrape_configs:

  - job_name: 'prometheus'
    scrape_interval: 60s
    scrape_timeout: 30s
    static_configs:
      - targets: ['localhost:8000','e1:9100','e4:9100','e5:9100','e6:9100','e7:9100','e8:9100','e9:9100','e10:9100','e11:9100']

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx,icmp]
    static_configs:
      - targets:
        - https://www.google.com
        - https://hpc.llnl.gov
        - https://lc.llnl.gov
        - e1
        - e4
        - e5
        - e6
        - e7
        - e8
        - e9
        - e10
        - e11
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: localhost:9115
~
```

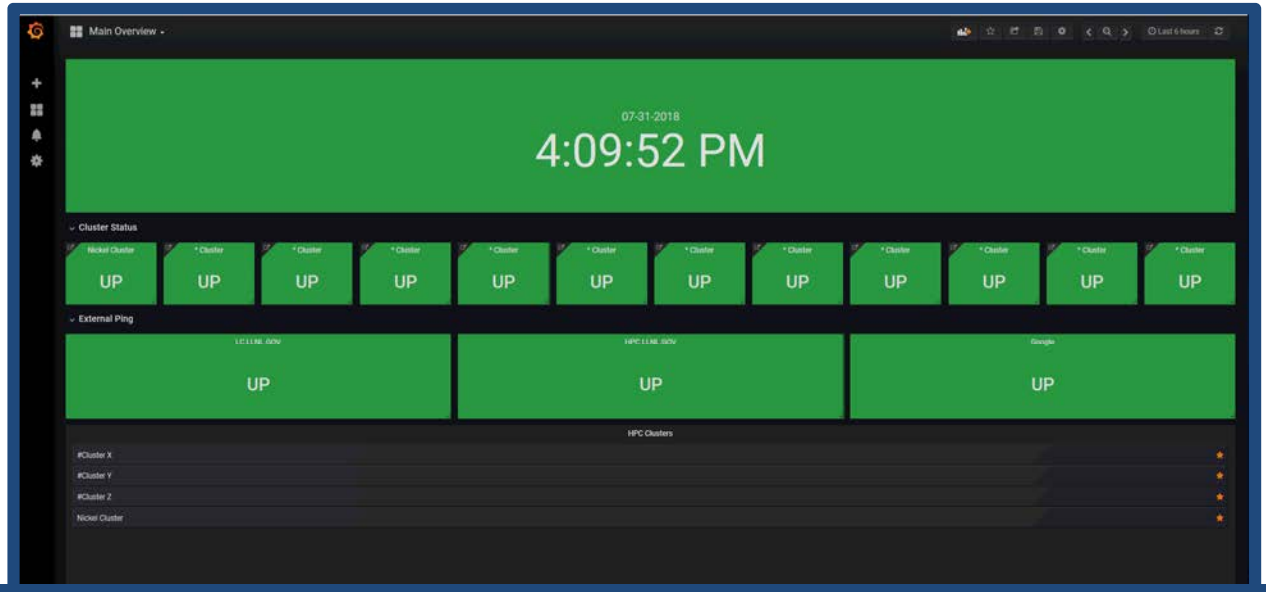
Exporters

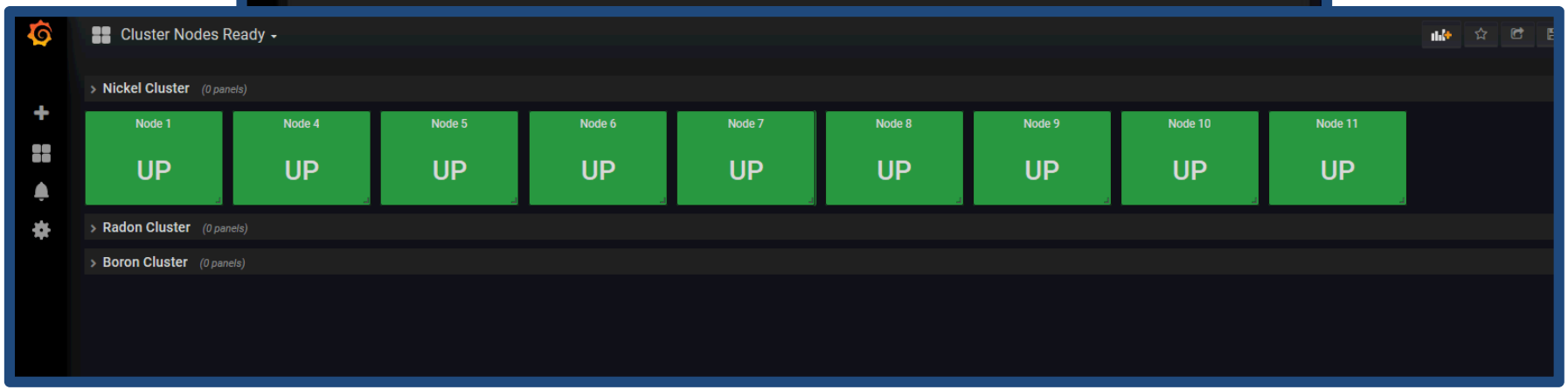
- Scrape various data from the nodes
  - Node Exporter
  - IPMI Exporter
  - Blackbox Exporter

Grafana

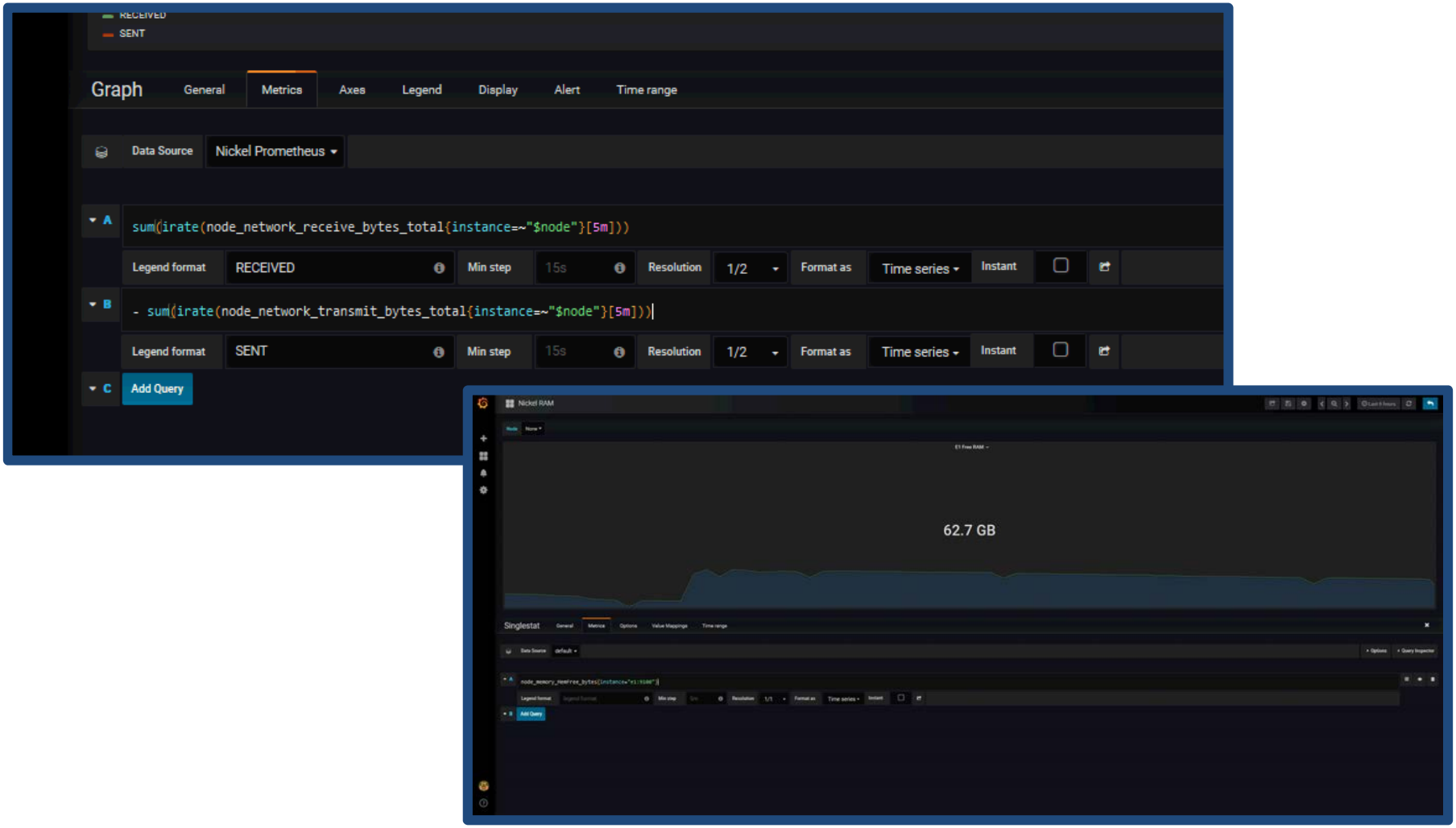
- Grafana is a versatile UI
- Displays the data from Prometheus on dashboards

Main Dashboard





Queries


RECEIVED
SENT
Graph
General
Metrics
Axes
Legend
Display
Alert
Time range
Data Source
Nickel Prometheus
A
sum(irate(node_network_receive_bytes_total{instance=~"$node"}[5m]))
Legend format
RECEIVED
Min step
15s
Resolution
1/2
Format as
Time series
Instant
B
- sum(irate(node_network_transmit_bytes_total{instance=~"$node"}[5m]))
Legend format
SENT
Min step
15s
Resolution
1/2
Format as
Time series
Instant
C
Add Query
Nickel RAM
62.7 GB
Singlestat

Drilldowns

Nickel CPU Temperatures
CPU 2
34.0 °C
34.0 °C
28.0 °C
28.0 °C
31.0 °C
31.0 °C
32.0 °C
32.0 °C
CPU 2 Temperature Heatmap
CPU 1

72 W
72 W
72 W
72 W
48 W
48 W

Alertmanager Configuration

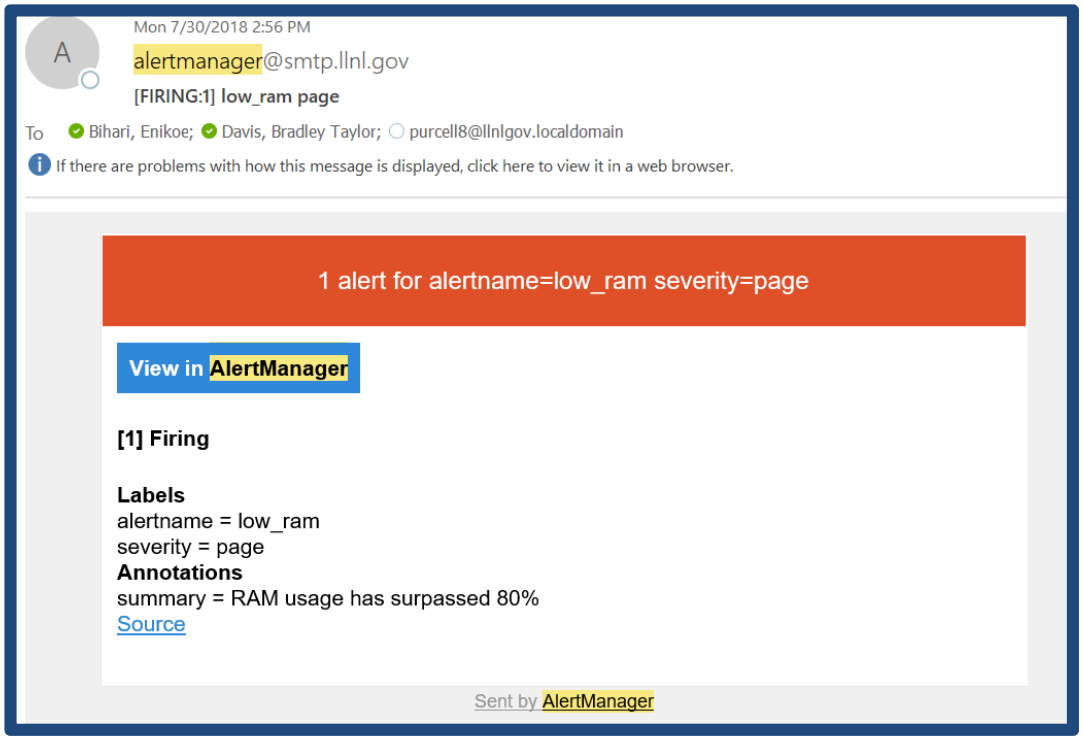
```
global:
  smtp_smarthost: 'localhost:25'
  smtp_from: 'alertmanager@smtp.llnl.gov'
  smtp_require_tls: false

route:
  group_by: ['alertname','instance','severity']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  receiver: nickel

  routes:
    - match:
        severity: critical
      receiver: TeslaMan


inhibit_rules:
- source_match:
    severity: 'critical'
  target_match:
    severity: 'page'

receivers:
  - name: 'nickel'
    email_configs:
      - to: 'bihari2@llnl.gov, davis282@llnl.gov, purcell8@llnlgov'

  - name: 'TeslaMan'
    email_configs:
      - to: 'dixon30@llnl.gov'

```

- Alertmanager groups and sends alerts when certain metrics reach a threshold
- Sends alerts over email and Slack to different groups depending on the type of alert

```
rule_files:
  - alert.rules.yml

alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093
```

```
groups:
- name: RAM
  rules:
  - alert: low_ram
    expr: sum(node_memory_MemFree_bytes) / sum(node_memory_MemTotal_bytes) < 0.20
    for: 1m
    labels:
      severity: "page"
    annotations:
      summary: "RAM usage has surpassed 80%"

  - alert: very_low_ram
    expr: sum(node_memory_MemFree_bytes) / sum(node_memory_MemTotal_bytes) < 0.05
    for: 1m
    labels:
      severity: "critical"
    annotations:
      summary: "RAM usage has surpassed 95%"

```

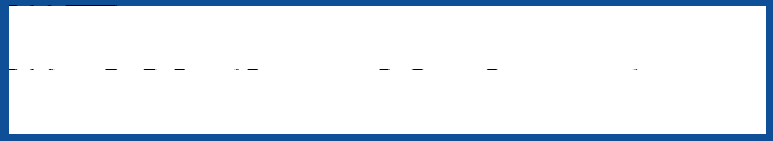
Alert Examples



Scalability With Ansible

- Automated the installation and configuration of Prometheus, Grafana, etc. with Ansible
- Allows for scalability for future use

Questions?



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Expanding Livmomi

Quinn Black
Maribel Cardiel
Jason Shortino
Matthew Xie

Mentors: Rigo Moreno Delgado and Mike Gilbert

August 14, 2018

LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

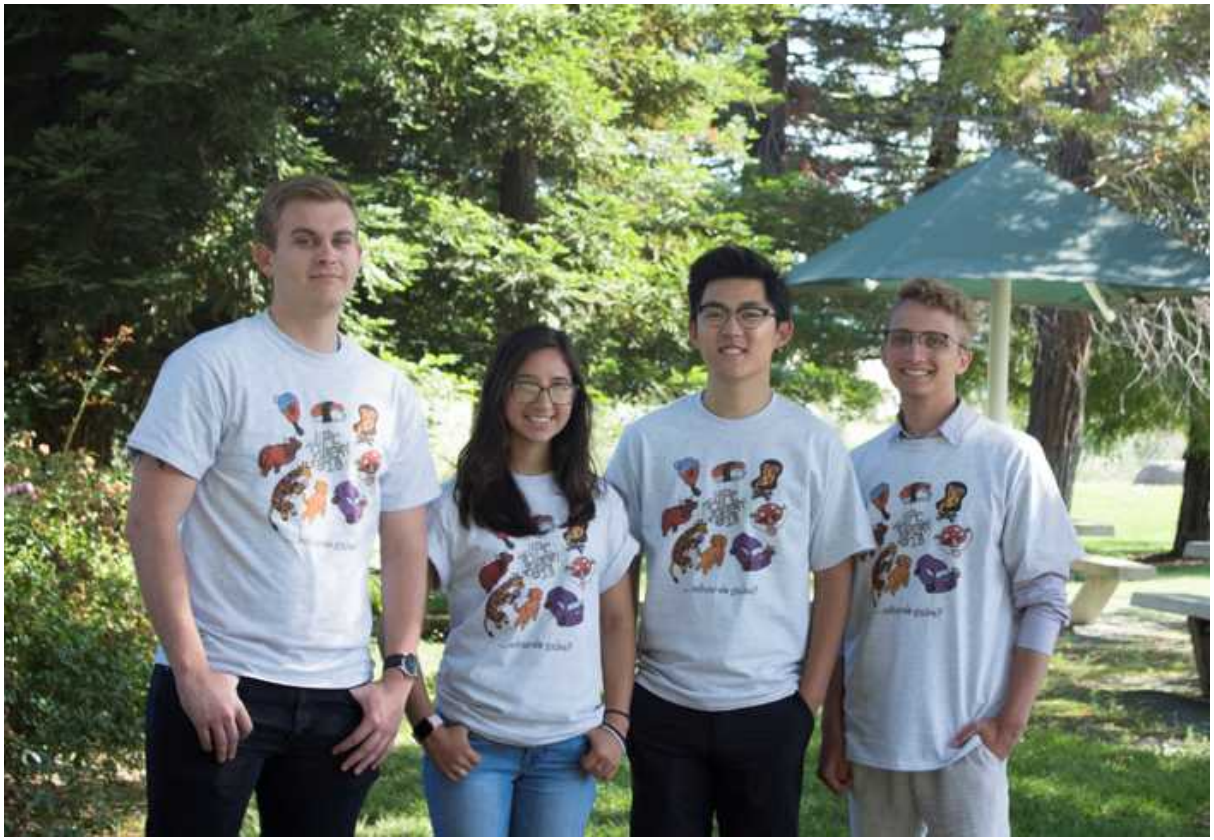
Our Group

Jason Shortino
CSU Long Beach

Maribel Cardiel
Stanford University

Matthew Xie
UC Berkeley

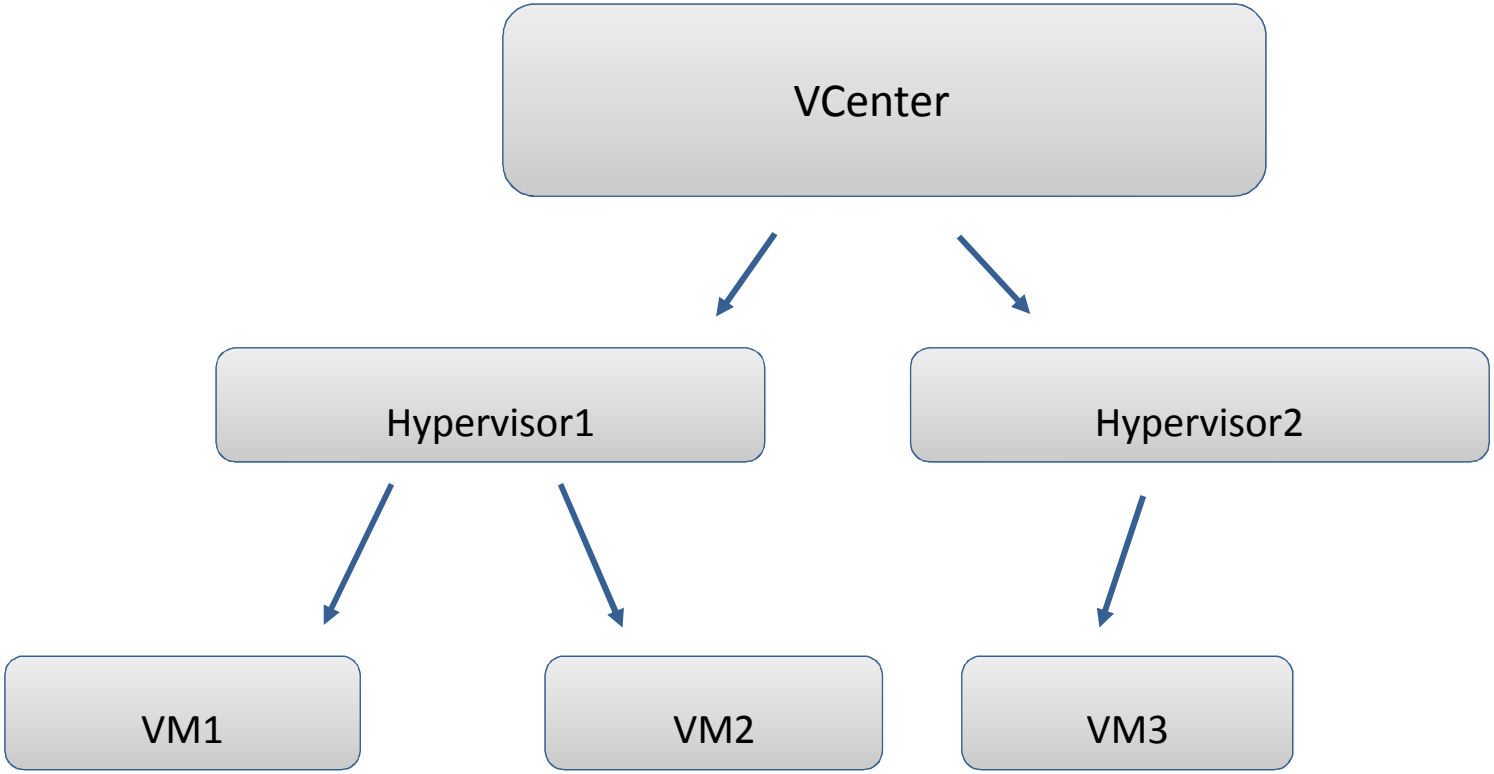
Quinn Black
Brigham Young University



Overview

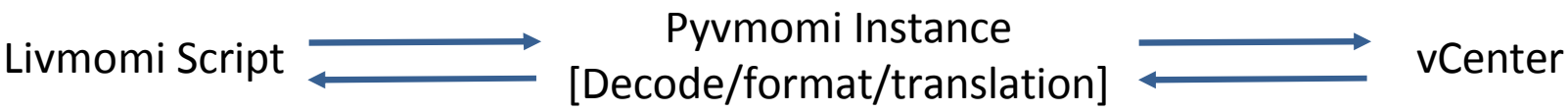
- What is Livmomi?
- Why does it matter?
- Expanding Feature-set Controls
- Challenges
- Research Topics
- Wrap Up

VMware Environment



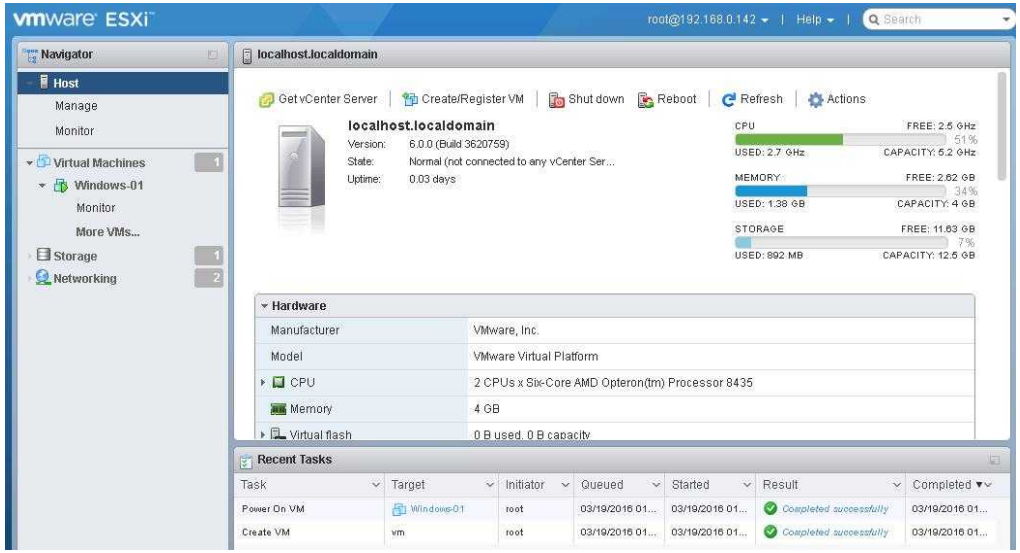
What is Livmomi?

- Livmomi: Set of scripts (developed within LC) written in Python using the Pyvmomi open-source API
  - Pyvmomi: Python VMware Managed Object Management Interface
  - Allows us to manage and gather information about VMs
- VMware environment consist of vCenters and hypervisors used for hosting and monitoring VMs



Why Does it Matter?

- What used to take minutes using VMware's vSphere GUI is done in seconds with Livmomi
- Increased efficiency and productivity in LC infrastructure clusters
- Various features are not supported via the GUI
- Who's using it
  - SAG, LC



Powering On and Off VMs

- New script that was added to the Livmomi suite
- Powers on and off VMs
- Supports hostlists and wildcards
- Returns status if operation is already complete

```
(venv)[cardiel2@lgw2:livmomi]$ ./power_vm.py --host 192.168.60.17 --user administrator --password [redacted] --vm lt-vm[1
-2,6] --operation power_on
Powering on VM:lt-vm1
VM: lt-vm1 has successfully powered on.
VM: lt-vm2 is already on
VM: lt-vm6 is already on
```

Retrieving Host Info From Specified VMs

- View the power states of VMs

```
(venv)[xie3@lgw2:livmomi]$ ./get_host_vms.py --host 192.168.60.17 --user administrator --password [redacted]
--human --state
192.168.60.16
    lt-vm7: PoweredOn
    lt-vm6: PoweredOn
    lt-vm5: PoweredOn
    lt-vm4: PoweredOn
    lt-vm3: PoweredOn
    lt-vm2: PoweredOn
    lt-vm1: PoweredOn
```

- Find a VM's host given its name

```
(venv)[xie3@lgw2:livmomi]$ ./get_host_vms.py --host 192.168.60.17 --user administrator --password [redacted]!
--human --vm lt-vm4
192.168.60.16
    lt-vm4
```

- All flags can be combined together

VM Snapshot Operations by Snapshot ID

- Snapshots previously were being identified by the name
- Now operations can be done by using the snapshot ID

```
Displaying list of snapshots on virtual machine lt-vm1
VM: lt-vm1; Snapshot name: Snapshot-2018-08-08; Snapshot ID: 11; Description: ; CreateTime: 2018-08-08 21:49:00.251965+00:00
VM: lt-vm1; Snapshot name: Snapshot-2018-08-08; Snapshot ID: 12; Description: ; CreateTime: 2018-08-08 21:49:06.031018+00:00
VM: lt-vm1; Snapshot name: Snapshot-2018-08-08; Snapshot ID: 13; Description: ; CreateTime: 2018-08-08 21:49:10.081025+00:00
The total number of snapshots for VM: lt-vm1 is 3
```

```
[shorti@lgw2:livmomi]$ ./snapshot_vm.py --vm lt-vm1 --operation remove --snap_id 13 --host 192.168.60.17
Removing snapshot Snapshot-2018-08-08 from VM: lt-vm1
```

Other Tasks and Features

- Created colorize() function to add color and highlights to output

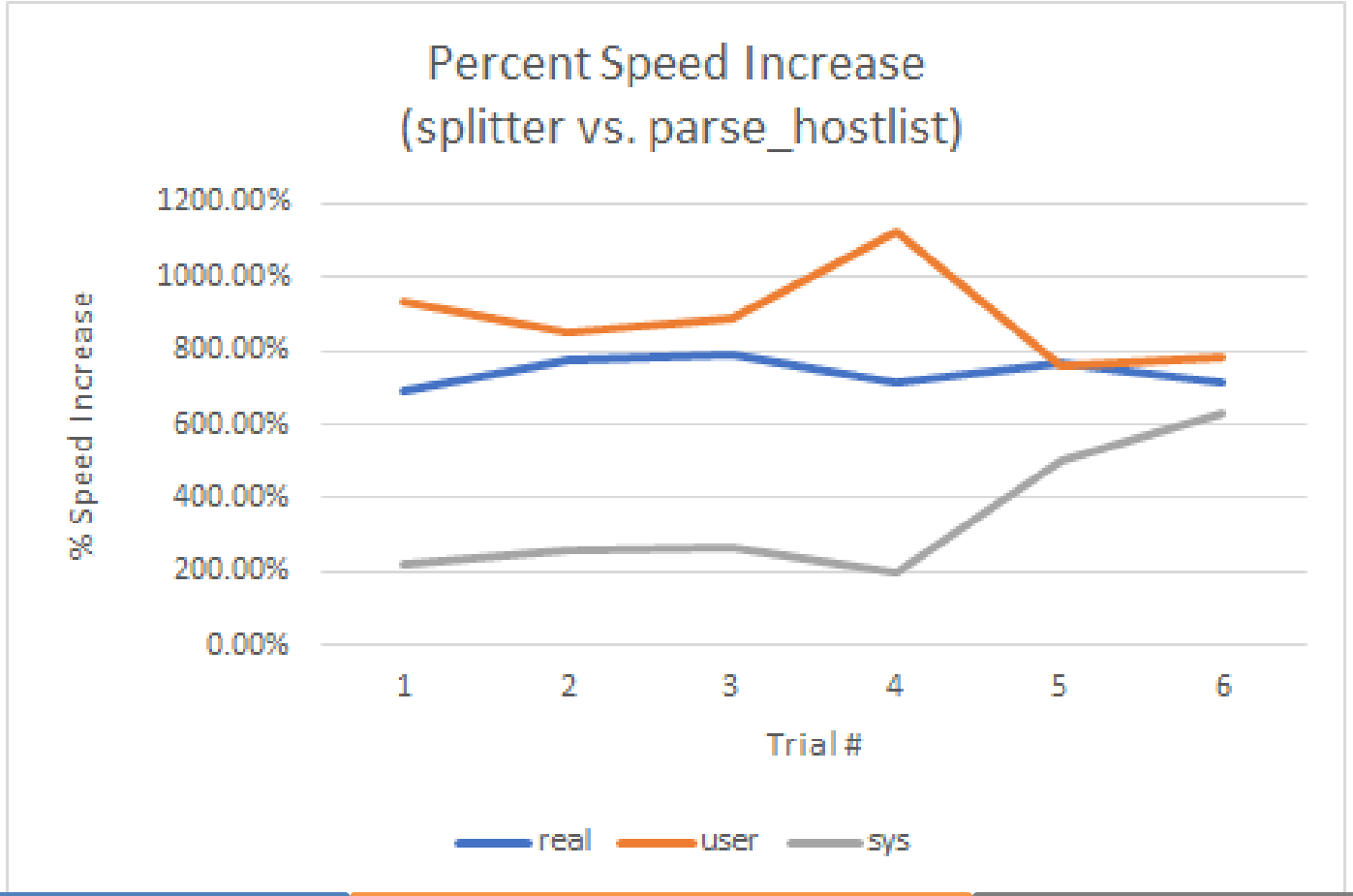
- Fixed parse_hostlist() function to support lists with names that include dashes

- Created splitter() to replace parse_hostlist()
  - Faster run times
  - Flexible user input formatting
  - Ordered output

```
lt-vm[1,2,3-5,9-10,50-90]
```

```
['lt-vm1', 'lt-vm2', 'lt-vm3', 'lt-vm4', 'lt-vm5', 'lt-vm9', 'lt-vm10', 'lt-vm50', 'lt-vm51',
'lt-vm52', 'lt-vm53', 'lt-vm54', 'lt-vm55', 'lt-vm56', 'lt-vm57', 'lt-vm58', 'lt-vm59', 'lt-vm
60', 'lt-vm61', 'lt-vm62', 'lt-vm63', 'lt-vm64', 'lt-vm65', 'lt-vm66', 'lt-vm67', 'lt-vm68', '
lt-vm69', 'lt-vm70', 'lt-vm71', 'lt-vm72', 'lt-vm73', 'lt-vm74', 'lt-vm75', 'lt-vm76', 'lt-vm7
7', 'lt-vm78', 'lt-vm79', 'lt-vm80', 'lt-vm81', 'lt-vm82', 'lt-vm83', 'lt-vm84', 'lt-vm85', 'l
t-vm86', 'lt-vm87', 'lt-vm88', 'lt-vm89', 'lt-vm90']
```

Improvement on Host Lists Parsing Time



Real: actual process run time beginning to end

User: time required by CPU to execute user-defined code

Sys: time required by CPU to execute system calls

Challenges

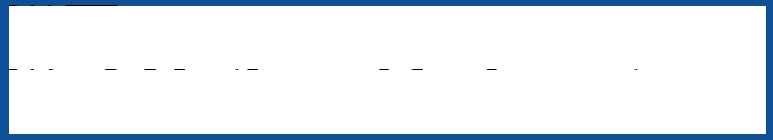
- Working with the Pyvmomi API and figuring out its intricacies
- Some of the Pyvmomi documentation is outdated
- Unable to use Git in the testing environment due to security limitations

Research Topics

- What's the best way to customize the OS on a newly cloned VM?
- Is it possible to migrate one vm to another environment all together?
- Create a script that will update a ESXi/Host

What's Next

- Wrap Livmomi up into a RPM for easier installation on LC systems
- Expand the Pyvmomi API
- Adding new functionality to Livmomi
- Implement Chris Moussa's parse_hostlist function



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

HPSS Deployment Automation

Livermore Computing

Alicja Gornicka & Gabriel Rusk
HPCCEA

August 2018

LLNL-PRES-XXXXXX

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

HPSS Team

Alicja Gornicka
University of California, Santa Cruz

Gabriel Rusk
Westminster College

WESTMINSTER
SALT LAKE CITY • UTAH

Objectives

I. Use a configuration management system (i.e. ansible) to integrate the configuration of an HPSS test environment

II. Use the configuration management with the cluster deployment to fully automate the HPSS install

III. Enable efficient testing of the HPSS environment

IV. Present packaged automation to the HPSS team

Our Project

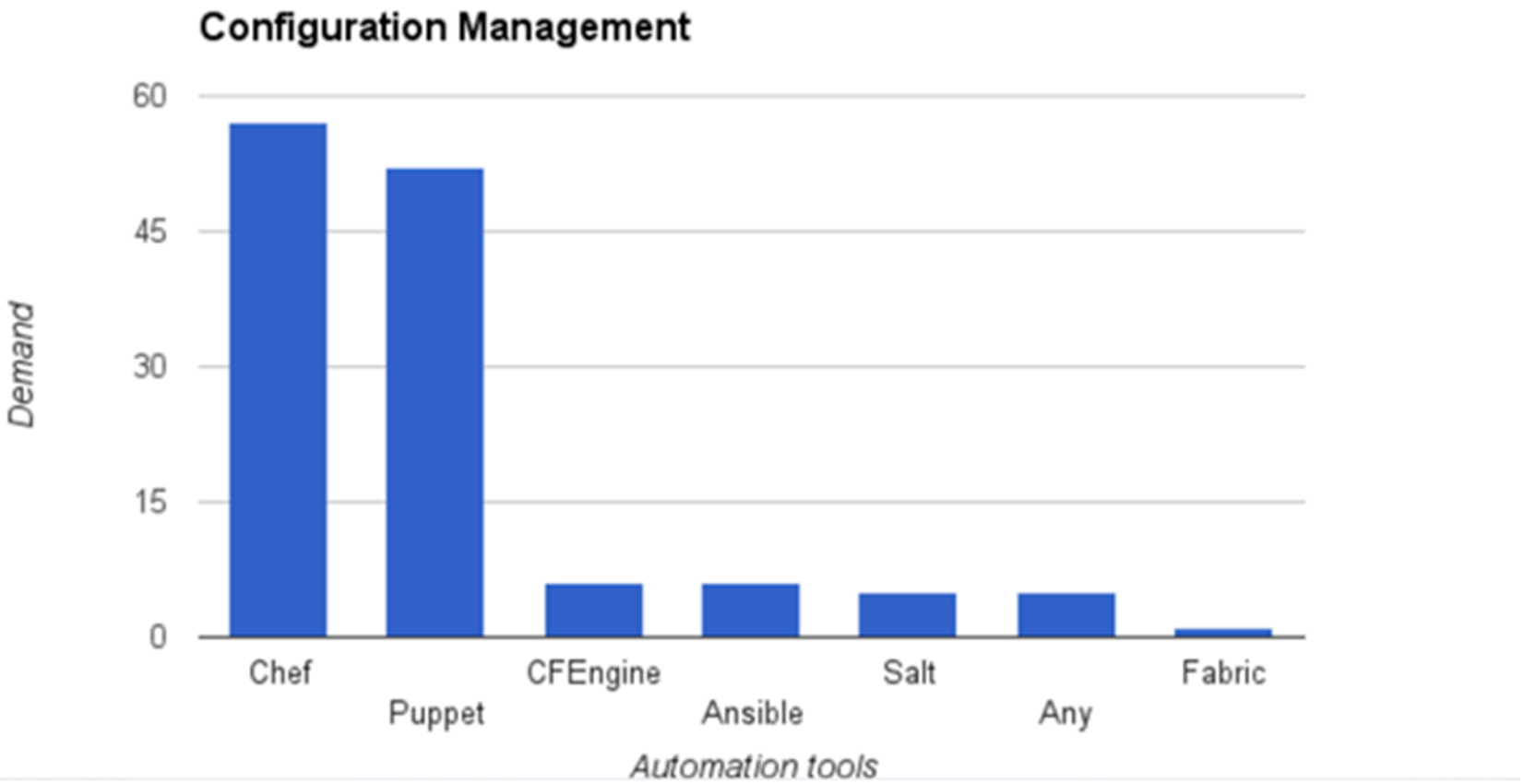
- Decide on which automation configuration management system to use
- Automate instructions from the HPSS deployment guide which do not require the GUI
- Automate the configuration of a test HPC cluster using ansible
- Introduce and teach ansible and its use to the HPC Academy members

High Performance Storage System

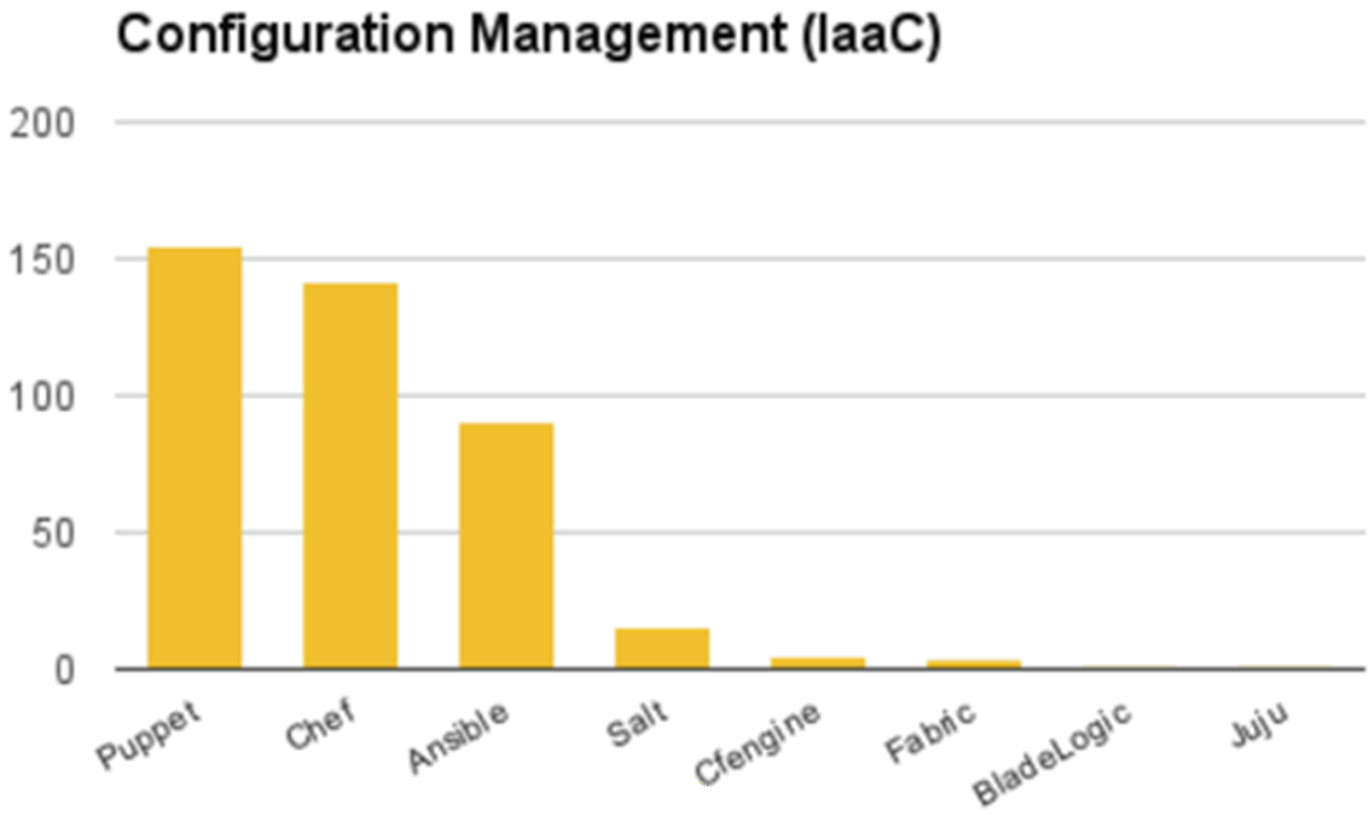
Ansible

- What is ansible?
  - Ansible is an open source configuration management tool that is quickly gaining popularity
  - Created to orchestrate multi-tier applications across clouds
  - Helps with software provisioning, configuration management, and application deployment
- Why ansible?
  - Created to counter difficulties of the other programs
  - The ansible team wanted to make something simple and easy to use so that people could just get and go
  - Well documented and has a great support community
  - All functions are performed over SSH

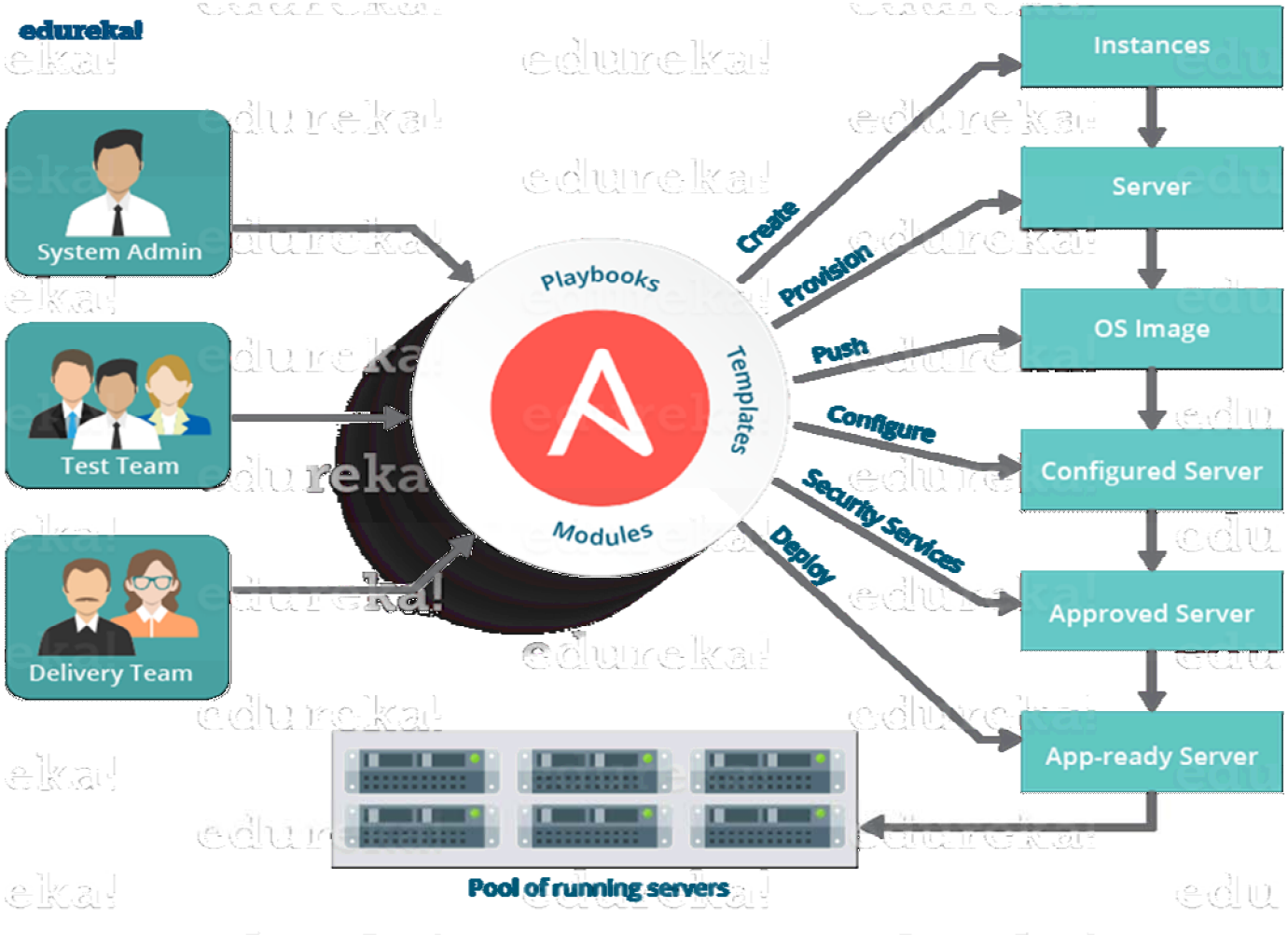
Ansible Usage in 2016



Ansible Usage in 2017



Ansible: Start to Finish



HPSS

- HPSS (High Performance Storage System) is a hierarchical file system software system designed to store and manage petabytes of data on disk and tape libraries in a network-centered, cluster-based environment
- HPSS is used in many large HPC sites (such as LLNL) to manage files on tape and disk

What we did

- Researched and studied what ansible is and how it works
- Implemented configuration tasks using ansible on the HPC test cluster
- Worked from an internal HPSS deployment document
  - Analyzed each section and split up the tasks
  - Automated most of sections 6-13 using ansible
    - Some sections could not be automated due to use of a GUI
    - Other sections were optional or used tools not required by LLNL
  - Used the HPC test cluster to test the automated HPSS deployment

The Automation of Section 7

7. Setting semaphore values

Adjust the semaphore values and sysctl settings (do this for core and movers):

1. Determine the amount of system memory.

```
# grep "MemTotal" /proc/meminfo
MemTotal: 49398860 kB
```

Memory in bytes = 49398860*1024 = 50584432640
Memory in GB = 49398860/1024/1024 = 47

2. Calculate the following variables which will be used to set the semaphore settings in /etc/sysctl.conf.

Table 1. Kernel Parameter Expressions

Memory_in_Bytes	MemTotal * 1024
Memory_in_GB	MemTotal / 1024 / 1024
shmmax	Memory_in_Bytes
shmall	2 * Memory_in_Bytes / 4096
shmmni	256 * Memory_in_GB
sem	256 256000 32 <1024 * Memory_in_GB>
msgmni	1024 * Memory_in_GB

The Automation of Section 7

```
# core memory variables
---
core_MemTotal: 65711772 # total memory from /proc/meminfo
core_Memory_in_Bytes: 67288854528 # core_MemTotal*1024
core_Memory_in_GB: 62 # core_MemTotal/1024/1024
core_shmmax: 67288854528 # core_Memory_in_Bytes
core_shmall: 32855886 # 2*Memory_in_Bytes
core_shmmni: 15872 # 256*Memory_in_GB
core_sem: 1644420584658272 # "256 256000 32 (1024*Memory_in_GB)"
core_msgmni: 64171 # 1024*Memory_in_GB
core_msgmnb: 65536
core_msgmax: 65536
core_space: 0

mover_MemTotal:
mover_Memory_in_Bytes:
mover_Memory_in_GB:
mover_shmmax:
mover_shmall:
mover_shmmni:
mover_sem:
mover_msgmni:
mover_msgmnb: 65536
mover_msgmax: 65536
mover_space: 0
~
~
```

```
---
- hosts: atest

  vars_files:
    - memVars.yml

  tasks:
    - name: append kernel variable to core sysctl.conf
      blockinfile:
        dest: /etc/sysctl.conf
        content: |
          # Controls the maximum shared segment size, in bytes
          kernel.shmmax = {{ core_Memory_in_Bytes }}

          # Controls the maximum number of shared memory segments, in pages
          kernel.shmall = {{ core_shmall }}
          kernel.shmmni = {{ core_shmmni }}
          kernel.sem = {{ core_sem }}
          kernel.msgmni = {{ core_msgmni }}
          kernel.msgmnb = 65536
          kernel.msgmax = 65536
          kernel.randomize_va_space = 0

    - name: commit changes
      command: /usr/sbin/sysctl -p
...
~
```

Challenges

- Getting the HPSS deployment documentation
- Time (haven't verified end to end deployment process)
- Understanding ansible, ansible syntax, and ansible error handling
- Bringing the HPC test cluster back up after a power glitch
- Documentation targeted someone familiar with HPSS and had access to the standard testing environment
- Wanted to use ansible modules rather than just using the ansible command line option
  - Figuring out how to do it the "ansible way"

Conclusion

- Ansible
  - Easy to use and quick to pick up
  - Must be very careful with syntax (tabs vs spaces)
  - Allows flexibility for what we wanted to do (HPSS and cluster build)
  - Good fit for automating the HPSS documentation
- Successfully automated required sections of the HPSS deployment document
  - On track to achieving goal of minimal human command line input
  - Once required variables are saved into the variable files, running one command installs the specified section file

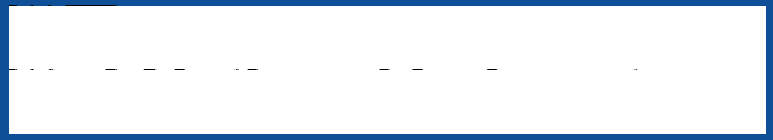
What's Next?

HPSS
High Performance Storage System

- Continue automation of the HPSS deployment
  - Clean up commenting and code
  - End to end testing and verification process
  - Adding in flags to allow user to specify which part they want to install
- Create a detailed README
  - How to use ansible to automate parts of the HPSS deployment documentation
  - Include what prerequisites are needed before running playbook
- Use ansible to automate the cluster install of the HPC test cluster
  - Fully automate the cluster build and package it for future reference
  - Make it easy to add in minor changes

Roll Credits

Thank you to **Jean, Dave, Thomas, Bryan, Rigo, Phil, Mike, and Geoff**
HPC Academy <3



Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.