BIOTEAM
Accelerate Science

AWS ParallelCluster: Tips, Tricks, and Lessons Learned

BioTeam 2022 Webinar Series

Welcome!

Today's Topics

- Why traditional HPC on AWS?
- Why ParallelCluster?
- Tips, Tricks & Lessons Learned
  - Operational Philosophy
  - Persistant Essentials
  - Node Configuration
  - Deployment Considerations
  - Post-deployment Acts
- Questions & Comments

Future Related Webinars

- *License-aware Schrödinger Integration with ParallelCluster*
  - September 14 2022
- *Enabling & Using the Slurm REST API with AWS ParallelCluster*
  - *TBD*

Acknowledgements

- AWS HPC Team
- AWS ParallelCluster Developers
- BioTeam's Karl Gutwin & Adam Kraut

BIOTEAM
Accelerate Science

Quick Intro

About Me

- Failed scientist turned HPC & research computing infrastructure nerd
- AWS user since EC2 was a private beta
- Reformed Grid Engine bigot

About This Webinar

- Moderated by Adam Kraut!
- Sorry! Not slick or high-level; this is personal
- **Promised actionable advice so you are going to get the dry and boring details delivered at a fast pace**
- **All mistakes in this presentation are my own**
- Slides are word-heavy because they are often passed around after the “live” event

chris@bioteam.net
Twitter: @chris_dag

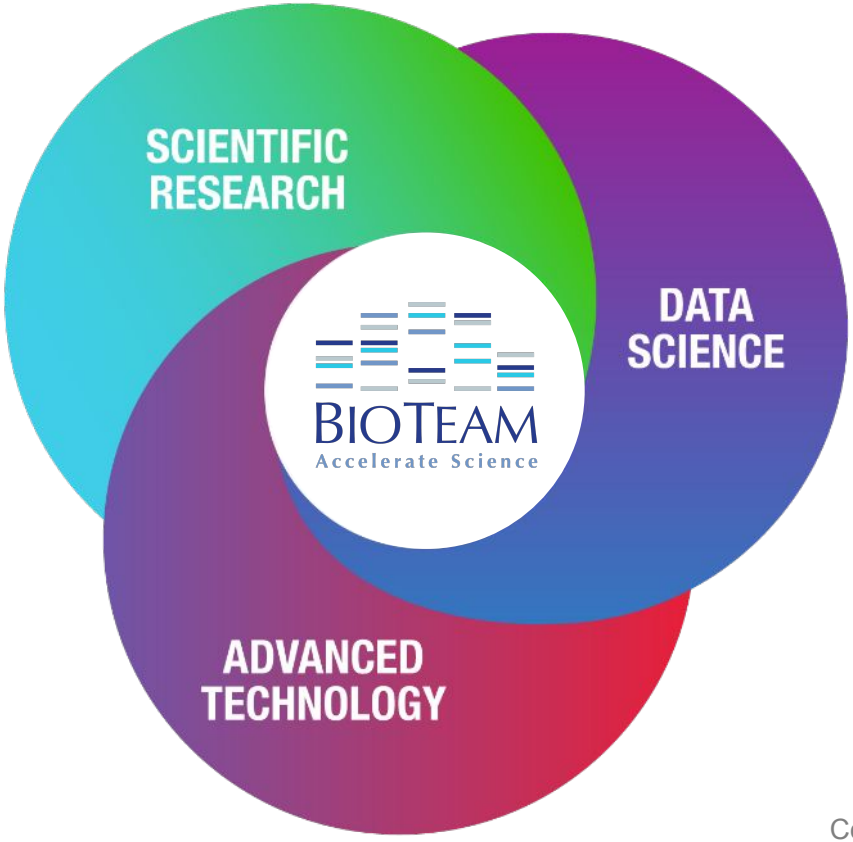
BIOTEAM
Accelerate Science

BioTeam, Inc.

These next few slides are for folks drawn by our niche technical content who have no idea who BioTeam is, how we are built, & what we tend to do

BIOTEAM
Accelerate Science

Scientific IT Consulting

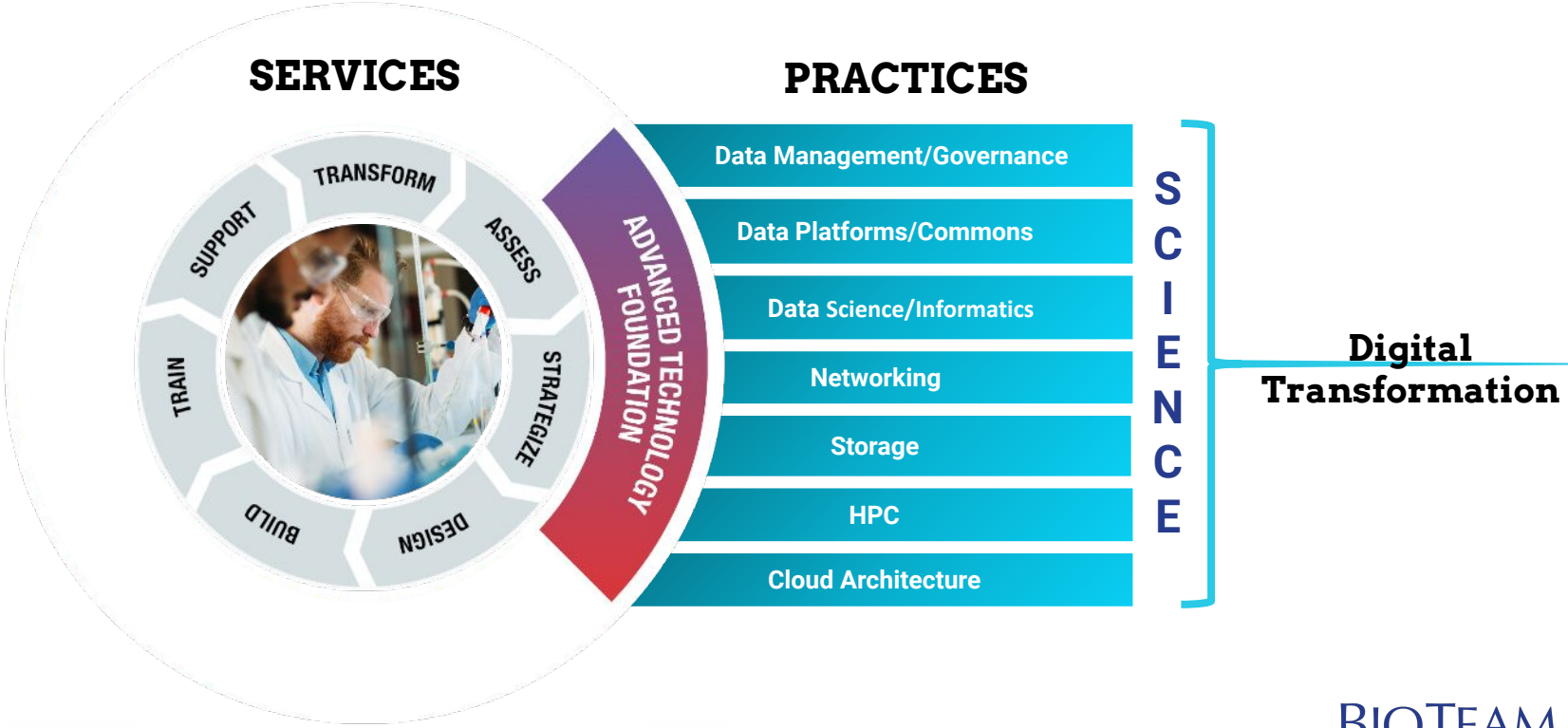


Why choose BioTeam

- Mastery of science, data, and technology integration
- Trusted and strategic partners
- Strategic thinkers, skilled at execution
- Holistic approach to solving problems
- Best practices from hands-on experience
- Creative, integrative, leading-edge, approachable strategies
- Vendor and technology neutral
- Highly collaborative teams that integrate with your team
- You hire the whole company to bear on your problems and strategies

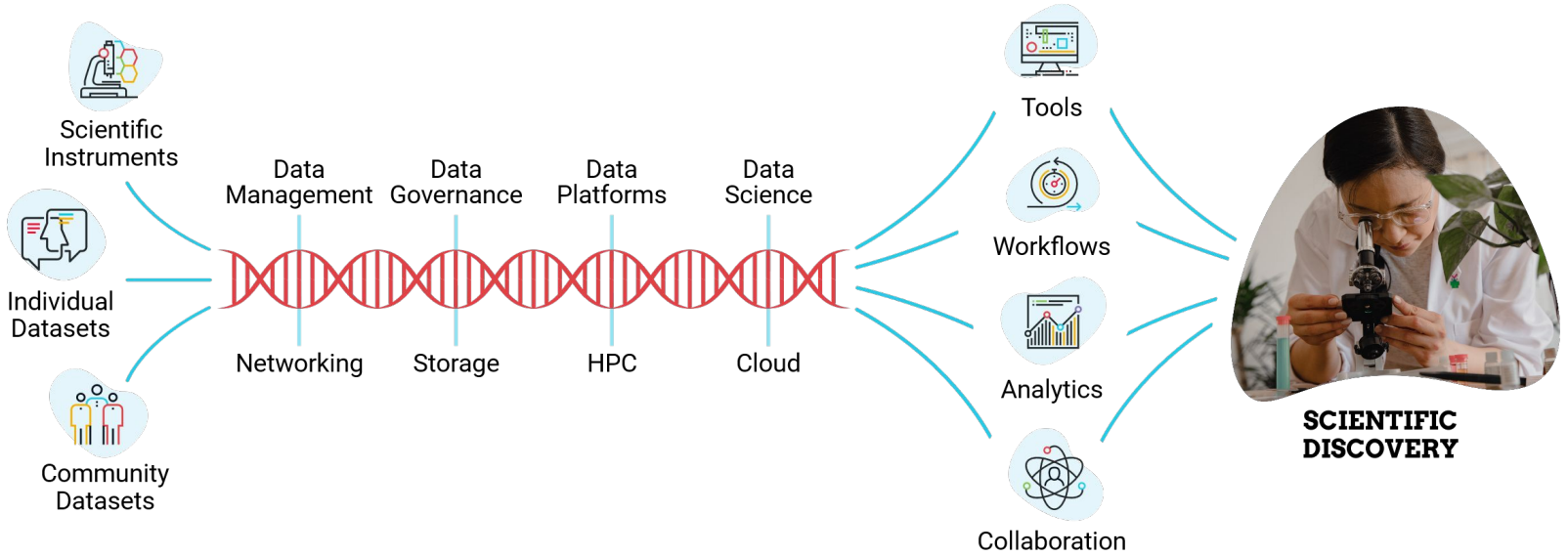
BIOTEAM
Accelerate Science

BioTeam scientific digital transformation capabilities



BIOTEAM
Accelerate Science

Our focus: Building scientific data ecosystems



BIOTEAM
Accelerate Science

WHY?

Why do we still need “Traditional HPC” on the cloud?

Talk Progress:

- **Why traditional HPC on AWS?**
- **Why ParallelCluster?**
- Tips, Tricks, & Lessons Learned
  - Operational Philosophy
  - Persistant Essentials
  - Node Configuration
  - Deployment Considerations
  - Post-deployment Acts
- Questions & Comments

BIOTEAM
Accelerate Science

Why "Traditional HPC" on AWS?

- Because cloud marketing and "evangelists" are lying to you
- In the cloud sales/marketing/evangelism world they pretend that legacy use cases and business requirements either do not exist or are simply too embarrassing to discuss
- In the rosy world of cloud marketing:
  - 100% of our applications, platforms, algorithms, and workflows are going to be re-architected and re-written from scratch using cloud-native design patterns
  - DevOps, Infrastructure-as-Code, & automated deployment is pervasive down to the level of individual scientific end-user
  - IT is fully staffed with cloud-aware engineers, support, & automation resources with a remit to directly work with scientists

Blunt Truth: "Traditional HPC" is not Going Away

In the data intensive life science informatics realm …

1. We have hundreds, if not thousands, of applications, algorithms, tools, and scripts that will NEVER be rewritten for cloud-native patterns.
2. Core tooling often requires or assumes "*Files and Folders*" view of storage - a transition to 100% native object storage will be a slow & lengthy process.
3. Scientists who invest time in learning Python & Bash for data science can instantly reuse those skills for pipelines & workflows within a traditional HPC environment.
4. Primary user/customer is a SCIENTIST working without significant cloud-aware support. Forcing scientists to learn cloud-native engineering practices means they are not actually performing the work they were recruited to do.

BIOTEAM
Accelerate Science

Why AWS ParallelCluster?

- Recreates “traditional HPC” environment on AWS while integrating cloud features like auto-scaling, Spot, & rapid reprovisioning

- Slurm becoming de facto standard across most large-scale supercomputing facilities spanning government, academia, & industry

- Easy to transition our code/scripts and workflow scripting
- Easy to transition Slurm admin and usage expertise

- Developed/maintained by AWS; free and open source
- Fantastic developer team
  - Actively maintained; new features/enhancements constantly

Note: Alternatives do exist. Schrödinger has an auto-scaling HPC product for comp chem that ships with built in Identity & VDI features. NVidia now owns Bright Cluster Manager and Altair now has PBSPro & Grid Engine -- both offer hybrid cloud or cloud-bursting capabilities that ParallelCluster does not yet support.

BIOTEAM
Accelerate Science

Tips, Tricks, & Lessons Learned

Learn from my dumb mistakes.

Talk Progress:

- Why traditional HPC on AWS?
- Why ParallelCluster?
- **Tips, Tricks, & Lessons Learned**
  - Operational Philosophy
  - Persistant Essentials
  - Node Configuration
  - Deployment Considerations
  - Post-deployment Acts
- Questions & Comments

BIOTEAM
Accelerate Science

01: Operational Philosophy

- Helps to change your mindset …
- Premise HPC
  - Shared large system w/ multiple users, workflows, & application mixes
    - Serial, parallel, batch, & interactive jobs
    - Large memory vs CPU heavy vs GPU heavy jobs
    - Latency sensitive MPI vs. storage obliterating high IO jobs
    - *"Do you know who I am? GIVE ME PRIORITY (and all the GPUs…)"*
  - Significant effort required to tune, configure, maintain, support, and troubleshoot systems that must handle many diverse concurrent users and application mixes
  - Hardware mix can only be upgraded, expanded, or altered every few years

01: Operational Philosophy

- The “Aha!” moment …
  - *I don’t have to wedge ALL users and workloads onto the same system !!!*
- Cloud lets me deploy multiple HPC clusters, each tuned for a specific application mix, use case, department, or project team
- For long time HPC folks, this is **game-changing** - especially those of us who’ve spent ***decades*** trying to configure large shared HPC environments to keep the # of angry scientists as low as possible

02: Persistent Essentials

- Lots of online info talking benefits of “disposable infra” and cloud HPC environments that are created/destroyed on-demand
- This works fine at the cloud-native cool kids table:
  - *“trigger a lambda when new data arrives; pull all data/code in from s3://; run HPC pipeline via service account user; blow it ALL away after writing results back to s3:// …”*
- **… but not in my world**
- **My HPC users are humans doing science**. They need persistent stuff that lives independently of any single ParallelCluster HPC environment

BIOTEAM
Accelerate Science

02: Persistent Essentials

Cloud HPC life is 1000x better when I have three persistent things:

1. **Identity**: A consistent source of truth for usernames, auth, & UID|GID
2. **$HOME**: A shared file system for user home directories
3. **$APPS**: A well organized, shared file system where all of the scientific software is centrally installed and maintained. Ideally, versioned and under management by something like Environment Modules.

This stuff needs to EXIST and PERSIST independently so we can plug them into our 'disposable' or 'transient' cloud HPC stacks …

BIOTEAM
Accelerate Science

02: Persistent Essentials

The solution for **IDENTITY** is "*whatever operationally works for you …*"

- Use native ParallelCluster support for Active Directory LDAP
- Create local user accounts in OS via Ansible triggered by CustomAction
- Configure auth to different LDAP via Ansible triggered by CustomAction
- Copy in /etc/passwd, /etc/shadow and /etc/group from s3 (!)
- Custom AMI image with your weird bespoke stuff baked into the OS

Note: The green methods are the most used by BioTeam

BIOTEAM
Accelerate Science

02: Persistent Essentials

The solution for **$HOME and $APPS** is "*start with AWS EFS as baseline and switch to something different when business, scientific, technical, or performance demand requires …*"

- AWS EFS is NFSv4 under the hood
- Very easy to set up including automatic backup and lifecycle tiering rules
- Native support in ParallelCluster, it "just works"

Downsides:

- Not fast enough for some use cases; EFS perf tuning or even paying for provisioned throughput is an arcane config optimization art that I don't really understand
- Infinite size of EFS share means unsupervised scientists will quickly add 700 terabytes of mysterious data with no provenance, management, organization, and no guilt about storing the exact same reference genome in 30 different locations.

  Beyond a certain scale, human effort is required to manage data.

02: Persistent Essentials

```
awsbioteam@ip-10-16-94-198:~$ pwd
/efs/home/awsbioteam
awsbioteam@ip-10-16-94-198:~$ id
uid=1748201745(awsbioteam) gid=1748200513(     ı-scicomp-ldap) groups=1748200513(     ı-scicomp-ldap),
1004(wheel)
awsbioteam@ip-10-16-94-198:~$ module avail
---------------------------------- /usr/share/modules/modulefiles ----------------------------------
dot  module-git  module-info  modules  null  use.own

------------------------------------- /efs/sw/modulefiles -------------------------------------
gromacs/intel-2022.1  python/3.10.4  r/4.2.0  schrodinger/2021-4  schrodinger/2022-2
awsbioteam@ip-10-16-94-198:~$
```

```
[centos@r   -vdi-workstation ~]$ module avail

---------------------------- /usr/share/Modules/modulefiles --------
dot         module-git  module-info modules     null        use.own

------------------------ /efs-mount/groups/compchem/modulefiles/ ---
cisTEM/1.0.0-beta
cmake/3.23.2
NAMD/alpha/3.0a12_x86-multicore-CUDA-singleNode
NAMD/stable/2.14_x86_64-multicore
NAMD/stable/2.14_x86_64-multicore-CUDA
NAMD/stable/2.14_x86_64-netlrts
python/3.10.4
[centos@mni-vdi-workstation ~]$
```

AD LDAP user with $HOME in EFS share using managed software installed in /efs/sw path

Example #2 showing TWO **versions** and FOUR "**flavors**" of the NAMD molecular dynamics app; highlighting value of tools like environment modules

BIOTEAM
Accelerate Science

Recap 01

Summarize our “lessons learned” so far …

Talk Progress:

- Why traditional HPC on AWS?
- Why ParallelCluster?
- **Tips, Tricks, & Lessons Learned**
  - **Operational Philosophy**
  - **Persistant Essentials**
  - Node Configuration
  - Deployment Considerations
  - Post-deployment Acts
- Questions & Comments

BIOTEAM
Accelerate Science

Recap 01: Lessons Learned So Far

1. Biggest mental shift for cloud-based HPC is realizing that you don't need to run single massive shared HPC stack tuned to minimize # of angry scientists. **ParallelCluster makes it reasonable to run many HPC clusters, each sized, configured, and tuned for a specific requirement.**

2. Don't be fooled by documentation or evangelism regarding ***"everything is disposable! Nuke it all and rebuild/redeploy when needed!"***

   There are certain core capabilities that should be durable and persistent in your environment so you can plug HPC clusters into them:

   - A consistent method of resolving user identities and authentication
   - Shared filesystem for home folders & centrally installed software

03: Node Configuration

- Version 3.2 of ParallelCluster when deployed with EFS and AD-LDAP integration comes as close to ***"zero touch post-deploy"*** as I've seen yet
- That said … we pretty much **ALWAYS** have to touch or tweak something in the cluster node OS before we hand over to users

```
SharedStorage:
  - MountDir: /efs
    Name: training-efs
    StorageType: Efs
    EfsSettings:
      FileSystemId: fs-04d5e7a959f570339
DirectoryService:
  DomainName: dc=use2-pcluster-dc01,dc=bioteam,dc=cloud
  DomainAddr: ldap://172.31.38.79
  LdapTlsReqCert: never
  PasswordSecretArn: arn:aws:secretsmanager:us-east-2:290725103381:secret:ADReadOnlyUser-
  DomainReadOnlyUser: ReadOnlyUser
  GenerateSshKeysForUsers: true
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: true
    ldap_search_base: ou=Users,ou=CORP,dc=use2-pcluster-dc01,dc=bioteam,dc=cloud
    override_homedir: /efs/home/%u
    override_shell:  /usr/bin/bash
Monitoring:
  DetailedMonitoring: false
  Logs:
    CloudWatch:
      Enabled: true
      RetentionInDays: 1
      DeletionPolicy: Delete
  Dashboards:
    CloudWatch:
      Enabled: true
Tags:
  - Key: LDAP
    Value: "True"
```

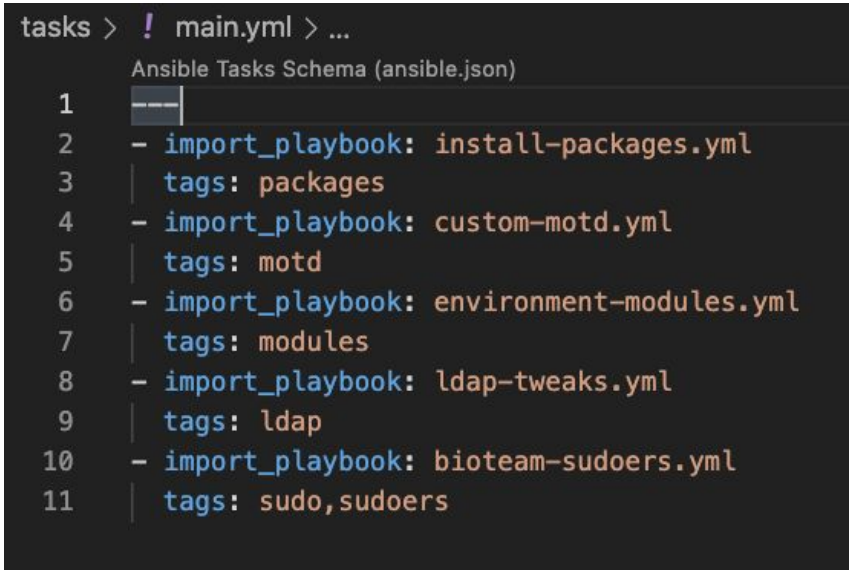
Note: This is a super unsafe unencrypted LDAP config used in a training class. DO NOT DO THIS IN PRODUCTION.

BIOTEAM
Accelerate Science

03: Node Configuration

Even with shared storage and LDAP handled by ParallelCluster, I still often need to:

1. Install packages
2. Create a login banner
3. Tell environment modules to also search the EFS mount for managed applications
4. Create a local group to fix the “GID not resolved” error when AD-LDAP is used
5. Give my LDAP users passwordless sudo access



BIOTEAM
Accelerate Science

03: Node Configuration

ParallelCluster has consistently added new features that make node configuration easier.

- **Ancient History: CfnCluster/Early-ParallelCluster era**
  - Pull region appropriate AMI ID from documentation; manually launch the image; manually apply all needed customizations; re-bundle into new AMI; cross fingers and hope cluster will launch without a fatal rollback error
- **Not that long ago**: **ParallelCluster 2.x version era**
  - ParallelCluster gained the fantastic ability to “install itself” into an AMI image you already have built. This was pretty transformative
- **Golden Age: ParallelCluster 3.x CustomActions era**
  - No more custom images! We launch the ‘native’ pcluster OS image and use `CustomActions` to apply whatever config changes we need

BIOTEAM
Accelerate Science

03: Node Configuration

No shame in making/maintaining custom AMI images for pcluster, but …

- Building, testing, and registering new AMI images is a massive time sink

However the biggest real world observable downside was:

- Custom AMI images effectively “pinned” or “froze” my clients to specific versions of ParallelCluster. We could not easily jump to the latest ParallelCluster release without a lot of time and work
- End result? Lots of folks still running “***old***” and “***it still works so why bother?***” clusters lacking the latest features/capabilities seen in the current release series
- And at the rate that ParallelCluster is adding feature enhancements you REALLY want a smooth operational way to upgrade or migrate to the latest release

03: Node Configuration

ParallelCluster **`CustomActions`**

- **`OnNodeStart`** - This hook lets you run a script as root right after the node boots up; perfect for node configuration
- **`OnNodeConfigured`** - This hook lets you run a script after the node is fully configured; Perfect for tweaking/adjusting Slurm configuration
- *Supported inside config blocks for the **HeadNode** and each Slurm **Partition/Queue** so you can do different things on "compute node" vs "head node" and even run different scripts against different Slurm Partitions*

```
  CustomActions:
    OnNodeStart:
      Script: https://assets.bioteam.net/pcluster/ubuntu-bootstrap-pcluster.sh
    OnNodeConfigured:
      Script: https://assets.bioteam.net/pcluster/mem.sh
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: cpu-q
    CustomActions:
      OnNodeStart:
        Script: https://assets.bioteam.net/pcluster/ubuntu-bootstrap-pcluster.sh
    Iam:
      AdditionalIamPolicies:
        - Policy: arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess
        - Policy: arn:aws:iam::290725103381:policy/bioteam-automation-enablement
```

Note: This screenshot is from an internal training class where our CustomAction scripts were hosted on a web server. In a real world production environment you'd likely be downloading scripts from an s3:// location secured via IAM EC2 instance role

BIOTEAM
Accelerate Science

My Favorite Node Config Method

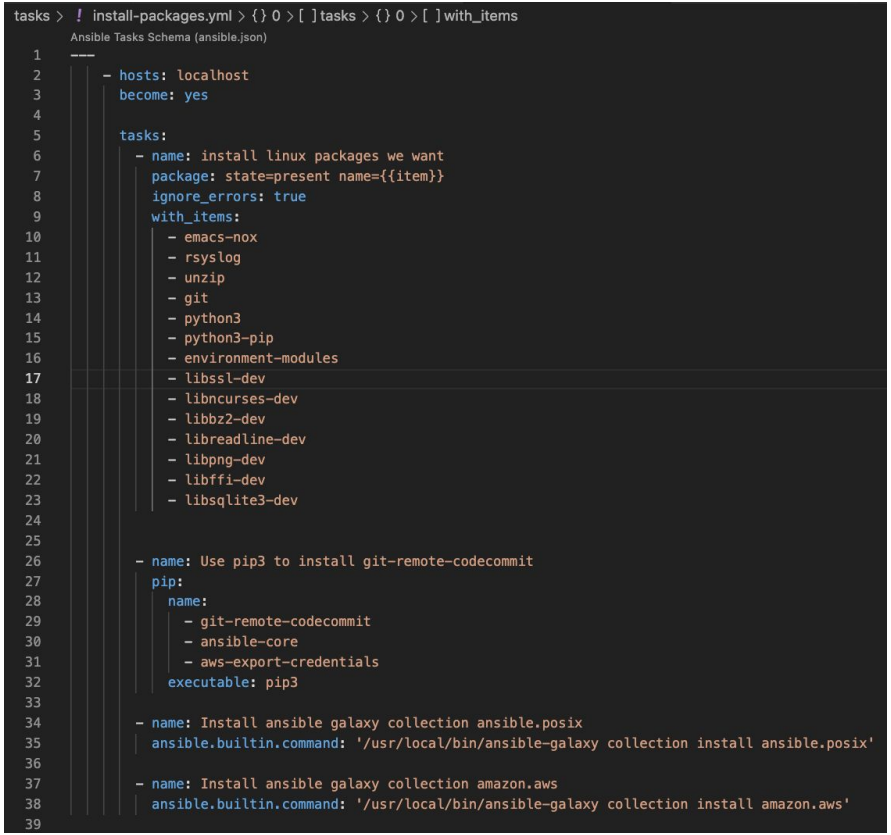
```
ansible-pull -U codecommit::us-west-2://bioteam-nodeconfig simple-playbooks/linux-common/tasks/main.yml
```

ansible-pull from playbooks hosted in AWS CodeCommit repo
(AWS CodeCommit uses git protocol)

BIOTEAM
Accelerate Science

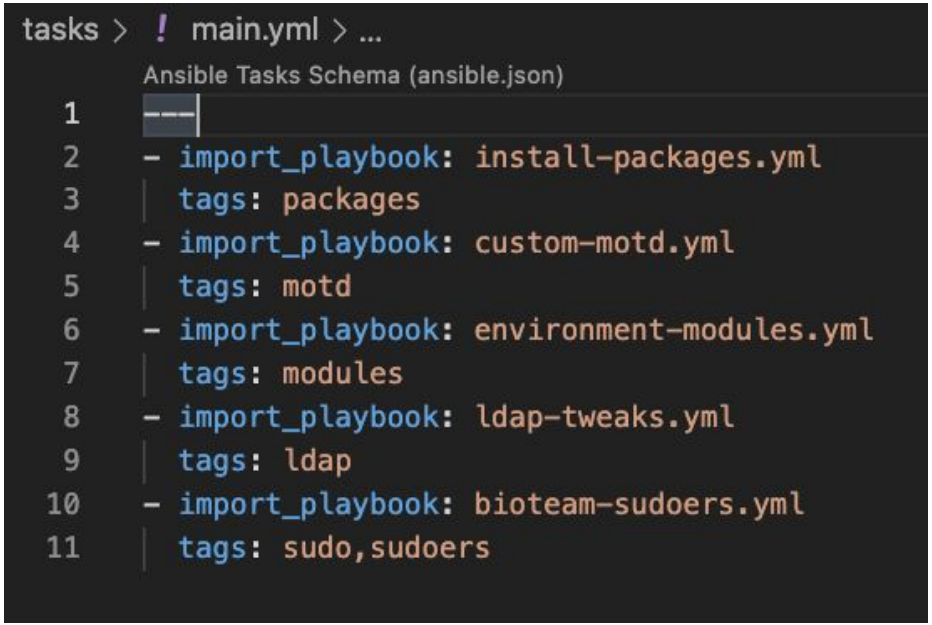
Why I Love “ansible-pull -U codecommit” for Config

1. Ansible is **super approachable,** the YAML files are easy to read/comprehend even by folks who have never seen it before
2. **HIGHLY repurposable,** which means I can use the same method across “*all Linux on AWS*” including:
   a. Ec2 Image Builder Pipeline building “trusted ec2 images” and distributing them across multiple AWS accounts
   b. Keeping all my other Linux servers and nodes in desired config state
   c. AWS ParallelCluster
3. **Idempotency** means I don’t have to rebuild/replace nodes to update them to latest config - I just rerun the git pull command!



How it Works

1. Create git repo in CodeCommit
2. Put Ansible playbooks in repo
3. Create IAM policy allowing access to repo; attach it to your EC2 IAM roles
4. Make sure nodes have the required packages (easily done with CloudInit or Pcluster CustomAction scripts)
5. Now bend the node to your will!




```
ansible-pull -U codecommit::us-west-2://usw2-sharedServices-imagebuilder-repo \
simple-playbooks/linux-nodeconfig-common/tasks/main.yml
```

BIOTEAM
Accelerate Science

IAM for Ansible Git-Pull from Codecommit

Recommendation

- This is not the only AWS permission you will want to be assigning your ParallelCluster nodes (S3, SSM, KMS, etc.)
- Consider making a consolidated IAM policy with a sufficiently generic name (“scicomp-automation-enablement”) so you can add all your special permissions into one easily assignable IAM policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CodeCommitPullFromNamedRepoOnly",
            "Effect": "Allow",
            "Action": [
                "codecommit:Describe*",
                "codecommit:Get*",
                "codecommit:List*",
                "codecommit:GitPull"
            ],
            "Resource": [
"arn:aws:codecommit:us-east-2:290725103381:ansible-playbooks"
            ]
        }
    ]
}
```

arn:aws:iam::290725103381:policy/bioteam-automation-enablement

BIOTEAM
Accelerate Science

ParallelCluster CustomAction Bootstrap Script:

```
#!/bin/bash

# Purpose: We have ansible playbooks hosted
# in a codeCommit git repo that perform common
# node configuration tasks.

# However parallelcluster nodes don't have the
# necessary tools installed to enable ansible-pull
# from codeCommit. So the purpose of this script
# is to install the minimal tools necessary to
# invoke further node config actions via ansible-pull

echo "Installing dependencies for ansible-pull"
/usr/bin/pip3 install git-remote-codecommit ansible-core

## Ansible node config via git pull from codecommit repo
echo "Calling ansible from codeCommit via ansible pull:"
/usr/local/bin/ansible-pull -U codecommit::us-east-2://ansible-playbooks bioteam-nodeconfig/linux-common/tasks/main.yml


echo "Downloading mem.sh script to set RealMemory on slurm configuration ..."
cd /opt; curl -o mem.sh https://assets.bioteam.net/pcluster/mem.sh

echo "Download myself to /opt so that admins can manually update nodes if needed"
cd /opt; curl -o ubuntu-bootstrap-pcluster.sh https://assets.bioteam.net/pcluster/ubuntu-bootstrap-pcluster.sh
```

BIOTEAM
Accelerate Science

CustomAction bootstrap script for **cross AWS account** codecommit pull

- More common these days to see “Shared Services” AWS accounts and VPCs
- Pulling a git repo from CodeCommit hosted in a different AWS account requires an extra IAM assume-role step
- Made easy via Ben Kehoe’s awesome SSO/credential utilities:
  - Twitter: @ben11kehoe
  - Github: https://github.com/benkehoe

```
#!/bin/bash

# Just to keep repos fresh
apt-get update
apt-get install unzip python3-pip

# Install AWS CLI v2
cd /tmp
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip -o awscliv2.zip
./aws/install

echo "Installing dependencies for ansible-pull"
/usr/bin/pip3 install git-remote-codecommit ansible-core

echo "Installing aws credential helper just into user space"
/usr/bin/pip3 install --user aws-export-credentials

# Set up /root/.aws/config file to support cross account
# assume role using initial credentials from EC2 metadata service
mkdir /root/.aws

/bin/cat <<EOF >/root/.aws/config

[profile cross-account-git-pull]
role_arn = arn:aws:iam::XXXXXXXX:role/sharedServices-crossAccount-codeCommit-repo-pull
optional: role_session_name = ansible-git-pull
credential_source = Ec2InstanceMetadata

EOF

# Need to assume a new IAM role to pull from code-commit in other AWS account
## Switch role and export temp session credentials as ENV vars so
## other CLI tools for ansible git pull will work

export $(/root/.local/bin/aws-export-credentials  --profile cross-account-git-pull --env)
aws sts get-caller-identity

## Now we can run ansible pull with cross account access
/usr/local/bin/ansible-pull -U codecommit::us-west-2://nodeconfig-repo \
simple-playbooks/linux-nodeconfig/tasks/main.yml
```

Closer Look at our Ansible Playbooks & Tasks

May skip for time reasons but will leave slides in for reference/review

BIOTEAM
Accelerate Science

main.yml - entry point for a playbook

tasks > ! main.yml > ...

Ansible Tasks Schema (ansible.json)

```
1  ---
2  - import_playbook: install-packages.yml
3    tags: packages
4  - import_playbook: custom-motd.yml
5    tags: motd
6  - import_playbook: environment-modules.yml
7    tags: modules
8  - import_playbook: ldap-tweaks.yml
9    tags: ldap
10 - import_playbook: bioteam-sudoers.yml
11   tags: sudo,sudoers
```

- main.yml breaks out what we want to do on each node into ‘tasks’
- By default every task in this main.yml file will be executed
- But we can tag the tasks as well so that if needed we can ansible-pull with a “--tag” argument to ONLY run (or rerun) a certain task

BIOTEAM
Accelerate Science

tasks/install-packages.yml

tasks > ! install-packages.yml > {} 0 > [ ] tasks > {} 0 > [ ] with_items

Ansible Tasks Schema (ansible.json)

```yaml
---
- hosts: localhost
  become: yes

  tasks:
    - name: install linux packages we want
      package: state=present name={{item}}
      ignore_errors: true
      with_items:
        - emacs-nox
        - rsyslog
        - unzip
        - git
        - python3
        - python3-pip
        - environment-modules
        - libssl-dev
        - libncurses-dev
        - libbz2-dev
        - libreadline-dev
        - libpng-dev
        - libffi-dev
        - libsqlite3-dev


    - name: Use pip3 to install git-remote-codecommit
      pip:
        name:
          - git-remote-codecommit
          - ansible-core
          - aws-export-credentials
        executable: pip3

    - name: Install ansible galaxy collection ansible.posix
      ansible.builtin.command: '/usr/local/bin/ansible-galaxy collection install ansible.posix'

    - name: Install ansible galaxy collection amazon.aws
      ansible.builtin.command: '/usr/local/bin/ansible-galaxy collection install amazon.aws'

```

- This is where we install OS packages we want available
- Ansible built-in package module is “OS aware” and will use the proper install commands (“yum” vs “apt”) depending on Linux distro*
  - ** The only gotcha is sometime package names differ across Linux variants so be prepared to use conditional logic in some cases*

BIOTEAM
Accelerate Science

tasks/custom-motd.yml -- template file for custom message

```yaml
- name: Custom MOTD for sytems that use the update-motd.d method
  ansible.builtin.template:
    src: ../templates/simple-motd.j2
    dest: /etc/update-motd.d/98-bioteam-custom
    owner: root
    mode: '0755'
  when:
    ansible_facts['distribution_file_variety'] == "Amazon" and ansible_facts['distribution_major_version'] == "2" or
    ansible_facts['distribution'] == "Ubuntu"
```

```
#!/bin/sh
cat << EOF

--
BioTeam Baseline Scientific Computing Server Image
  Config method: ansible-pull via playbooks hosted in repo codecommit::us-east-2://ansible-playbooks
  Last update: {{ ansible_facts['date_time']['date'] }} at {{ ansible_facts['date_time']['time'] }} {{ ansible_facts['date_time']['tz'] }}
--

EOF
```

templates/simple-motd.j2 - Ansible uses the Jinja templating engine to dynamically create files with highly customized content. In this lame MOTD template example we are just pulling in date, time and timezone info from pre-existing **ansible_facts**

BIOTEAM
Accelerate Science

custom-motd.yml -- Users see this on login

```
--
BioTeam Baseline Scientific Computing Server Image
  Config method: ansible-pull via playbooks hosted in repo codecommit::us-east-2://ansible-playbooks
  Last update: 2022-07-21 at 13:19:14 UTC
--
```

BIOTEAM
Accelerate Science

tasks/environment-modules.yml

Ansible Tasks Schema (ansible.json)

```
---
- hosts: localhost
  become: yes

  tasks:
    - name: Add our EFS modulespath to environment modules configurati
      lineinfile:
        path: /etc/environment-modules/modulespath
        line: /efs/sw/modulefiles/
```

- This is simple but important. Using ansible "*add a line to an existing file*" module, we can add an EFS file location to the list of places where "modules" will look for managed scientific software installations
- Now users who login and type "`module avail`" will see all the centrally installed and managed scientific applications hosted in our EFS share that we've mounted to our HPC cluster

BIOTEAM
Accelerate Science

tasks/ldap-tweaks.yml

Ansible Tasks Schema (ansible.json)

```yaml
---
- hosts: localhost
  become: yes

## Need this group created on all LDAP integrated hosts
## bioteam-scicomp-ldap:x:1817200513:

  tasks:
    - name: Ensure group "bioteam-scicomp-ldap" exists with correct gid
      ansible.builtin.group:
        name: bioteam-scicomp-ldap
        state: present
        gid: 1817200513
```

- This resolves an annoying error with AD-LDAP integration
- SSSD/AD integration maps all LDAP users to a default group GID
- **However this group does not actually exist and is not resolvable by default; resulting in an annoying warning error visible to end-users**
- We fix this by using Ansible to make a local group using the same GID that SSSD mapped to our AD-LDAP users

BIOTEAM
Accelerate Science

tasks/bioteam-sudoers.yml

```yaml
- hosts: localhost
  become: yes

# The group bioteam-scicomp-ldap is created in
# ldap-tweaks.yml and uses the global GID that Active
# directory gives all LDAP users. End result of this
# task here is all of our AD defined LDAP users end up
# getting passwordless sudo on the OS

  tasks:
    - name: Make sure we have a 'bioteam-scicomp-ldap' group (paranoia)
      group:
        name: bioteam-scicomp-ldap
        state: present
    - name: Allow 'bioteam-scicomp-ldap' group to have passwordless sudo
      lineinfile:
        dest: /etc/sudoers
        state: present
        regexp: '^%bioteam-scicomp-ldap'
        line: '%bioteam-scicomp-ldap ALL=(ALL) NOPASSWD: ALL'
        validate: visudo -cf %s
      ignore_errors: yes
```

- Sorry InfoSec people! You can cover your ears/eyes for this section
- Scientists use servers as “digital laboratories” not static business endpoints
- Scientists should have the ability to control/manage their “digital lab bench”
- Hence, I have no issue at all granting sudo access to all my scientific users

BIOTEAM
Accelerate Science

Recap 02

Summarize our “lessons learned” so far …

Talk Progress:

- Why traditional HPC on AWS?
- Why ParallelCluster?
- Tips, Tricks, & Lessons Learned
  - Operational Philosophy
  - Persistant Essentials
  - **Node Configuration**
  - Deployment Considerations
  - Post-deployment Acts
- Questions & Comments

BIOTEAM
Accelerate Science

Recap 02: Lessons Learned So Far

3. AWS ParallelCluster is rarely “zero touch”. There are often things you will want to change or implement on the HPC OS node images

4. Latest ParallelCluster makes it easy to install ParallelCluster into an EC2 AMI image you’ve already created; minimizing but not entirely removing the downsides associated with pcluster version-specific custom images

5. Latest ParallelCluster support for different flavors of **CustomActions** is game-changing. It is easy now to bootstrap your config changes into “native” ParallelCluster images and you can run different scripts on HeadNode vs ComputeNode or even different Slurm partitions

6. Use whatever **CustomActions** bootstrap method works for you; we showed you our favorite method involving ansible-pull and AWS CodeCommit

BIOTEAM
Accelerate Science

04: Deployment Considerations

Always install ParallelCluster into a dedicated Python VENV

- The "`pcluster`" script is version specific, you can't use it to manage/control/update any ParallelCluster stack
- If you don't maintain separate ParallelCluster instances pinned to dedicated VENVs you may lose your ability to manage or control "older" stacks; keep all your VENVs handy until you don't need them
- Name your python VENV with the version of ParallelCluster installed within it so you can easily switch versions
- Python version matters. ParallelCluster v3.2 dropped support for Python 3.6 and earlier — awkward if you are using CentOS 7 to manage your HPC fleets!

04: Deployment Considerations

Do not hoard ParallelCluster config .yaml files on your laptop

- I love deploying HPC from my laptop but …
  - Modifying a running ParallelCluster stack requires access to the .yaml config file used to launch it
  - This is operationally painful if the config file lives on someone's laptop and someone else needs to do something to the HPC stack
- **Recommended Option A**: Launch a cheap burstable EC2 node and make it your "HPC operations box" shared by all admins; put all your VENVs and config files there
- **Recommended Option B**: Push config files to S3 or use a Git repo and make sure it ALWAYS has the latest "live" version

BIOTEAM
Accelerate Science

04: Deployment Considerations

Consider launching ParallelCluster stacks with Admin-level permissions

- This one may be controversial and it's OK to disagree
- I used to spend a lot of time exhaustively crafting least-privilege IAM security policies for "pcluster-deployer" and "pcluster-stack"
  - This took time and a lot of trial and error
  - The IAM policies became pcluster version specific because each new version of ParallelCluster often required a slightly different permission mix
- If you deploy Pcluster v3.x series with Admin-like permissions, then ***ParallelCluster itself will create well scoped IAM roles and policies that are cluster-specific***.

04: Deployment Considerations

Make use of `AdditionalIamPolicies`

- Very easy to anoint your HPC nodes with additional AWS permissions
- Essential feature both for Ops as well as “getting science done”
- In example below:
  - `Ec2ReadOnlyAccess` used for slurm “mem.sh” script that uses EC2 API calls to query instance memory config so it can set proper `RealMemory` values
  - `SSMManagedInstanceCore` because I love ssm-agent being able to function
  - `bioteam-automation-enablement` - CodeCommit and other permissions

```
Iam:
  AdditionalIamPolicies:
    - Policy: arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess
    - Policy: arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
    - Policy: arn:aws:iam::290725103381:policy/bioteam-automation-enablement
```

BIOTEAM
Accelerate Science

04: Deployment Considerations

Tag your sh!t

- Tagging resources is an essential “cloud hygiene” act since forever
- Easy to do in pcluster .yaml config
- There is no excuse not to use your existing Tagging standards and policy on ParallelCluster deployments
  - *You **DO** have a Tagging policy and configured set of Cost Allocation Tags, right?*

```
Tags:
  - Key: ENV
    Value: PROD_NoAWSBackup
  - Key: PATCHGROUP
    Value: SSM_NOPATCH
  - Key: Description
    Value: This cluster does stuff for Department X
  - Key: Department
    Value: Department X
  - Key: Purpose
    Value: Scientific Computing
  - Key: TechnicalContact
    Value: Chris Dagdigian
  - Key: Business Owner
    Value: Ari Berman
  - Key: CostAllocation
      Value: ProjectX
```

BIOTEAM
Accelerate Science

04: Deployment Considerations

Consider pcluster `Monitoring`: settings

- { from memory } older pcluster versions used to create cloudwatch log groups with “never expire” settings -- potentially wasteful if you were not aware of this
- Current default is 14 days if you don’t specify otherwise
  - I choose between 1-day and 14-day retention depending on if I consider the HPC cluster to be “PROD” or “DEV/TEST
- `DeletionPolicy = Retain` by default
  - I almost always change this because by the time I’m deleting an HPC stack, I no longer have any interest in the logs

```
Monitoring:
  DetailedMonitoring: false
  Logs:
    CloudWatch:
      Enabled: true
      RetentionInDays: 1
      DeletionPolicy: Delete
  Dashboards:
    CloudWatch:
      Enabled: true
```

BIOTEAM
Accelerate Science

04: Deployment Considerations

Read the fine print on AD-LDAP Integration

- Today AD-LDAP integration is best achieved by using a REAL domain controller rather than AWS Managed AD
- There is a MASSIVE gotcha with AWS Managed AD that makes setting up encrypted LDAPS awkward
  - Current workaround is "*deploy an SSL-terminating load balancer between your cluster and AWS Managed AD*"
- Knowing this, I will always push for a real Domain Controller when LDAP integration is needed

```
SharedStorage:
  - MountDir: /efs
    Name: training-efs
    StorageType: Efs
    EfsSettings:
      FileSystemId: fs-04d5e7a959f570339
DirectoryService:
  DomainName: dc=use2-pcluster-dc01,dc=bioteam,dc=cloud
  DomainAddr: ldap://172.31.38.79
  LdapTlsReqCert: never
  PasswordSecretArn: arn:aws:secretsmanager:us-east-2:290725103381:secret:ADReadOnlyUser-
  DomainReadOnlyUser: ReadOnlyUser
  GenerateSshKeysForUsers: true
  AdditionalSssdConfigs:
    ldap_auth_disable_tls_never_use_in_production: true
    ldap_search_base: ou=Users,ou=CORP,dc=use2-pcluster-dc01,dc=bioteam,dc=cloud
    override_homedir: /efs/home/%u
    override_shell:  /usr/bin/bash
Monitoring:
  DetailedMonitoring: false
  Logs:
    CloudWatch:
      Enabled: true
      RetentionInDays: 1
      DeletionPolicy: Delete
  Dashboards:
    CloudWatch:
      Enabled: true
Tags:
  - Key: LDAP
    Value: "True"
```

Note: This is a super unsafe unencrypted LDAP config used in a training class because we used AWS Managed AD.
DO NOT DO THIS IN PRODUCTION.

BIOTEAM
Accelerate Science

Recap 03

Summarize our “lessons learned” so far …

Talk Progress:

- Why traditional HPC on AWS?
- Why ParallelCluster?
- Tips, Tricks, & Lessons Learned
  - Operational Philosophy
  - Persistant Essentials
  - Node Configuration
  - **Deployment Considerations**
  - Post-deployment Acts
- Questions & Comments

BIOTEAM
Accelerate Science

Recap 03: Lessons Learned So Far

7. Pin parallelcluster installs to named Python VENVs
8. ParallelCluster config files need to be accessible by all HPC operators
9. Ponder IAM and consider deploying with Admin permissions so pcluster can write it's own tightly scoped IAM roles and policies
10. `AdditionalIamPolicies` is your friend
11. Tag your sh!t. No excuse
12. Tune `Monitoring:` section to set proper retention and deletion values
13. Beware of the LDAPS 'gotcha' when considering use of Managed AD

05: Post-Deployment

aka “things I have not automated yet …”

- Things that I always do on every ParallelCluster deployment
  - Set up Accounting so “`seff`” and “`sacct`” utilities work
- Things that I do on a case-by-case basis
  - Configure Slurm job level memory management and enforcement
  - Configure Slurm “application license-aware scheduling”
  - Integrate Schrodinger Computational Chemistry suite so that its own internal jobcontrol system can submit “license-aware” jobs to Slurm
- Note:
  - **The ParallelCluster developers are slowly erasing the list of tasks I perform post-deploy by pushing out new releases with these features or capabilities now baked in**
    - Example:
      - ParallelCluster 3.2.0 release now has Slurm memory-based scheduling

BIOTEAM
Accelerate Science

05: Post-Deployment

You really want to configure Slurm Accounting & JobCompletionLog

- IT or HPC Ops team may not care but HPC job accounting data is an invaluable information source for end-users
- Accounting tells users how their jobs exited and provides other useful information and data
- Smarter shops can ingest and report on accounting data to determine if their HPC cluster config is optimized and if nodes are achieving high utilization (critical for expensive instance types)
- **Annoying Note:** Prior versions of Slurm allowed simple file-based accounting to be set up. The updated version of Slurm shipped with ParallelCluster 3.2 removed support for file-based accounting storage
  - You are basically forced to set up a mySQL/mariaDB server now

BIOTEAM
Accelerate Science

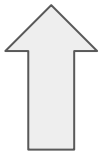
05: Post-Deployment

Why this is worth doing

- ‘seff’ is a fantastic script that uses accounting data to report on job level resource consumption
- This data is ESSENTIAL if you plan to hard enforce memory based job scheduling on the cluster
- Without functional ‘seff’ and ‘sacct’ utilities, users have very little insight into past job data and usage metrics

```
$ seff 4697

Job ID: 4697
User/Group: /scicomp-ldap-users
State: TIMEOUT (exit code 0)
Cores: 1
CPU Utilized: 00:00:00
CPU Efficiency: 0.00% of 00:01:27 core-walltime
Job Wall-clock time: 00:01:27
Memory Utilized: 101.96 MB
Memory Efficiency: 97.11% of 105.00 MB
```



This is INVALUABLE data to end-users who need to profile and understand the resources their jobs and workflow require; you need to have working Slurm accounting to use this.

BIOTEAM
Accelerate Science

05: Post-Deployment

I set up mySQL for ParallelCluster Accounting in different ways

- For heavy HPC shops with lots of work, it makes sense to use AWS RDS to set up a mySQL service that can be used by many clusters simultaneously
  - One RDS instance can run many databases so you can support multiple ParallelCluster environments with a single RDS deployment
- For lighter environments or simpler installs we just drop a mysql server on the ParallelCluster login node
- **The “right choice” is the one that best works operationally for you**

BIOTEAM
Accelerate Science

05: Post-Deployment

Other less-frequent post-deploy activities:

- Set up scripts that poll flexlm license servers and update Slurm in support of “application license-aware” job scheduling
- Compile, install and configure `slurmrestd` daemon to enable the Slurm REST API for remote job submission, control, and monitoring
- Integrate with Open XDMoD for reporting, monitoring, and service metrics - open.xdmod.org/10.0/index.html
- Future: Just got told about pcluster.cloud WebUI stack for managing and accessing ParallelCluster 3.x series clusters; looks interesting!

05: Post-Deployment

One last thing …

- Slurm Accounting “HowTo” docs often do not mention that in AWS you can’t really issue “`GRANT ALL …`” statements because RDS prohibits users from certain types of GRANTs.
- There are online URLs with database creation command examples that will fail when used on AWS RDS …
- Here are some notes from the last time I made a mySQL RDS database to store Slurm accounting info, rather than “GRANT ALL”. I had to be very specific with my GRANT commands:

```
CREATE database slurm_acct_db;

GRANT SELECT,INSERT,UPDATE,DELETE,DROP,ALTER,CREATE,CREATE ROUTINE,
ALTER ROUTINE on slurm_acct_db.* TO 'slurm'@'%' WITH GRANT OPTION;
```

BIOTEAM
Accelerate Science

Recap 04

Summarize our “lessons learned” so far …

Talk Progress:

- Why traditional HPC on AWS?
- Why ParallelCluster?
- Tips, Tricks, & Lessons Learned
  - Operational Philosophy
  - Persistant Essentials
  - Node Configuration
  - Deployment Considerations
  - **Post-deployment Acts**
- Questions & Comments

BIOTEAM
Accelerate Science

Recap 04: Lessons Learned So Far

14. Still things we need to do to Slurm “post-deploy” although the ParallelCluster dev team is working hard at shrinking this list (thanks!)

15. The most common/valuable post-deploy Slurm task is to set up accounting so utilities like `'seff'` and `'sacct'` work for your end-users.

16. Simple setup of accounting using files is no longer supported in the version of Slurm shipping with ParallelCluster 3.2 — a SQL database now seems to be required

BIOTEAM
Accelerate Science

end; thanks!

Time for Q&A & Discussion

Slides from this webinar will be available on our website