Moab/NODUS Cloud Bursting for Moab Workload Manager

Administrator Guide for Moab Workload Manager 9.1.2 and Amazon Web Services (AWS)

July 2018

Adaptive Computing Enterprises, Inc.
704 Goodlette Road North
Naples, FL 34102
+1 (239) 330-6083
www.adaptivecomputing.com

Scan to open online help

Welcome

Welcome to the *Moab/NODUS Cloud Bursting for Moab Workload Manager 9.1.2 Test Drive Quick Start*.

The Moab/NODUS Cloud Bursting aims to give you all the pieces needed to do bursting to public cloud as quickly and easily as possible. To accomplish, all the software needed for bursting comes preinstalled in some AWS images that are shared with your AWS account. These images can be launched in your AWS account to setup a virtualized on-premises cluster where bursting can occur. Following some simple instructions to set them up, you can be experimenting with the bursting capability without having to dedicate any real hardware to the cause or spend time to install the software stack.

Moab/NODUS Cloud Bursting Test Drive Quick Start

The Moab/NODUS Cloud Bursting Test Drive consists of two Amazon Machine Images preinstalled with Moab Workload Manager, Torque, and Moab Web Services. They are also preconfigured for bursting to AWS.

In this topic:

1.0.1 Moab/NODUS Cloud Bursting Test Drive AWS Images

Adaptive Computing will share the Test Drive images with your account in your requested region. Once they have been shared, log in to AWS Console and go to that region of EC2. Then click on AMIs and search for "Test Drive" under `Private Images`. You should see two images: "Test Drive w/ Viewpoint Server Node" (Server) and "Test Drive Compute Node" (Compute Node).

Server

The "Test Drive w/ Viewpoint Server Node" image contains Moab, Torque Server, Moab Web Services, Insight DB, and Viewpoint. It also has two NFS mounts, `/var/share` and `/home`. The user to connect with is `ec2-user` with sudo privileges, but other users can be created. The default hostname that is configured in Moab and Torque is `bursting-sched`.

Compute Node

The other image, "Test Drive Compute Node," is the compute node and has the Torque client installed on it. It is also configured to mount the NFS shares on the server. This image is intended to be used to simulate both “on-prem” nodes as well as cloud nodes.

1.0.2 Recommended Instance Types

When starting the server it is recommended to use a multi-process instance type—ideally, a `t2.xlarge` but the compute nodes can use a single-process `t2.micro`, if needed, although a more powerful instance type is recommended.

1.0.3 Basic Instance Configuration

To get the images able to see one another and do bursts, there are several steps needed to setup the instances and their networking. You can use the `rapid_deployment` script, which is designed to setup the Moab server and any on-prem nodes, in preparation for bursting.

1.0.4 Prerequisites

In order to set up the Test Drive, an AWS Access Key ID and Secret Access Key must be generated to launch instances. If your account is the AWS owner, the Access Key ID and Secret Access Key can be found under your account in `My Security Credentials > Access keys (access key ID and secret access key) > Create New Access Key`. Be sure to save the Access Key ID and Secret Access Key, as it will be needed later to configure NODUS.

If your account is an IAM user with administrator access, the Access key ID and Secret access key can be found under `IAM > Users`. Select your user, click on the Security Credentials tab, then click `Create Access Key`. Be sure to save the Access Key ID and Secret Access Key, as it will be needed later to configure NODUS.

If your account is an IAM user without administrator access, an AWS administrator can generate the Access Key ID and Secret Access Key for you by following the same steps as an IAM user with administrator access above.

Also, without Administrator access, certain policies must be created in IAM in order to allow the user to have the correct privileges needed to launch instances. An AWS administrator can use the following JSON files to create the policies needed for a user to create stacks and launch instances:

- `NODUS_Policies_Bursting.JSON`
- `NODUS_Policies_Stack_Build.JSON`

1.0.5 Security Groups and Ports

In order for the deployed instances to talk to one another, you must create a Security Group and specify the open ports for the Administrator and bursted instances.

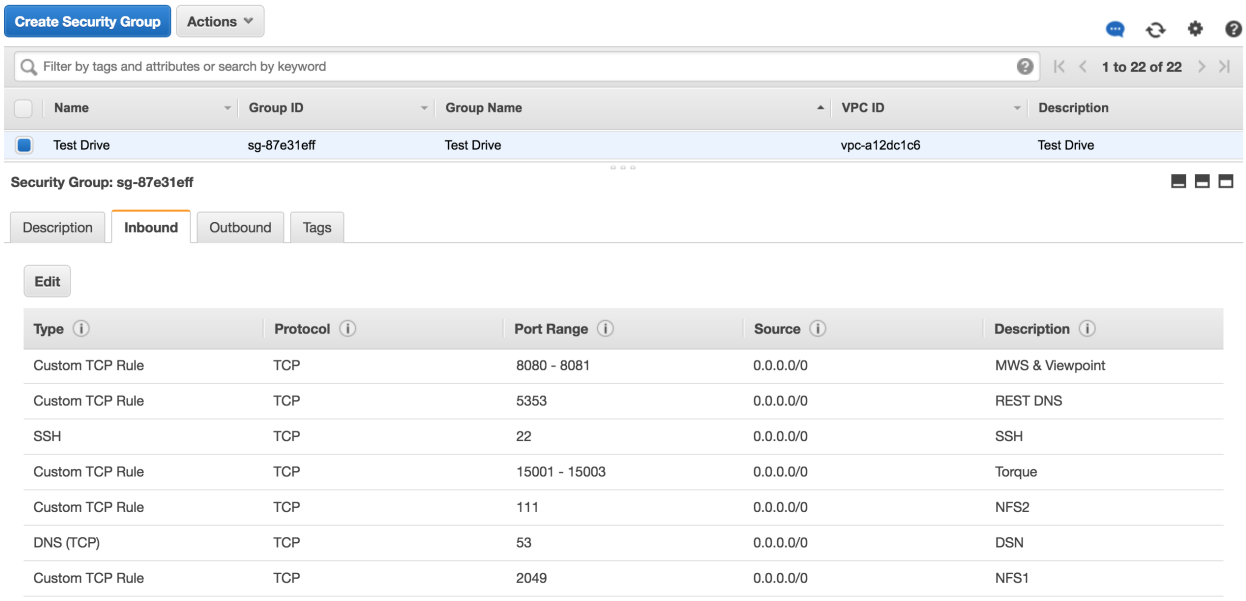
To create the Security Group and specify the open ports, do the following:

1. Go to `AWS > EC2 > Security Groups` and click `Create Security Group`.
2. Name the Security Group "Test Drive" and click `Add Rule` for each of the following components and ports:

Component	Protocol	Ports	Source
SSH	TCP	22	0.0.0.0/0
Torque	TCP	15001 15002 15003	0.0.0.0/0
MWS	TCP	8080	0.0.0.0/0
Viewpoint	TCP	8081	0.0.0.0/0
DNS	TCP	53 5353	0.0.0.0/0
NFS	TCP	111 2049	0.0.0.0/0

3. Click `Create`.

   After you have set up the Security Group, it should look similar to the following:



4. To better secure the instances, replace the source "0.0.0.0/0" for Torque, DNS (53 & 5353), and NFS (2049 & 111) to the IP range of your instances once the instances have been launched and the IP range is known (e.g. 172.31.0.0/16).

1.0.6 Create NODUS Key Pair

NODUS needs a `nodus` key pair in order to finalize the created instance. The `nodus` key pair can also be used to connect to the bursted instances. The following can be done on any machine that has the command *`ssh-keygen`* to create the key pair.

To create the NODUS key pair, perform the following steps:

1. Using the command *`ssh-keygen`*, create a new key pair, calling it `nodus`:

```
% ssh-keygen -t rsa -C "nodus" -f ~/nodus
```

   This will create public/private key pair files named `nodus.pub` and `nodus` in your home directory.

2. Rename the `nodus` private key file from `nodus` to `nodus.pem`. This is to easily identify the private key file for future reference in this guide.

3. To add the `nodus.pub` public key to AWS, do the following:

   a. In the AWS console, navigate to EC2 > Key Pairs and click “Import Key Pair.”

   b. Browse to the `nodus.pub` file created above (or paste in the file contents) and use `nodus` as the key pair name.

1.0.7 Rapid Deployment Script

The Rapid Deployment script sets up the hostname and `/etc/hosts` file on all the AWS server and "on-prem" instances so that they can resolve one another's hostname. It is meant to be run from any local Linux (or Linux-enabled) desktop. It makes remote SSH connections to the AWS instances and automates the same steps found in 1.0.8 Manual Setup - page 9, including copying the Moab licenses and restarting the services.

The Rapid Deployment script is available here.

To run the Rapid Deployment script, do the following:

1. Start the instances in AWS and select the `nodus` key. If you don't have a nodus key pair, see 1.0.6 Create NODUS Key Pair - page 8 above.
2. Once the instances are up, create a local text file and add the public and private IP addresses as well as the hostnames to the file in the following format, listing the Moab server as `bursting-sched`. This file will be passed as an argument to the Rapid Deployment script.

```
<public IP address>  <private IP address>  bursting-sched
<public IP address>  <private IP address>  <On-prem hostname1>
<public IP address>  <private IP address>  <On-prem hostname2>
. . .
. . .
```

3. Save the `moab-rlm.lic`, `moab-rlm-elastic-tracking.lic`, the `nodus.pem`, and `nodus.pub` key files (created above) to the directory where `td_rapid_deployment.sh` is located.
4. Run the `td_rapid_deployment.sh` script using the text file (created in step 2 above) as the second argument and the `nodus.pem` key (created in 1.0.6 Create NODUS Key Pair - page 8) as the third argument:

```
% ./td_rapid_deployment.sh <path to text file> nodus.pem
```

If you have problems connecting, make sure the permissions on the `nodus.pem` file are set to 600.

```
chmod 600 nodus.pem
```

If you have successfully run the Rapid Deployment script, skip to 1.0.9 Setup Verification - page 11. If there are any problems using the Rapid Deployment script, proceed to the steps in 1.0.8 Manual Setup - page 9 to configure the instances.

1.0.8 Manual Setup

Connecting to the instances is done over SSH. For each instance, do the following using the key pair that was used to start the instances:

```
% ssh -i nodus.pem ec2-user@<public IP address>
```

Perform the following steps:

1. Set the hostname. AWS always sets the initial hostname but it needs to be changed.

   On the server:

   ```
   [ec2-user ~]$ sudo hostnamectl set-hostname bursting-sched
   ```

   On each compute node:

   ```
   [ec2-user ~]$ sudo hostnamectl set-hostname <my_hostname>
   ```

   Change `<my_hostname>` to the hostname you prefer (e.g. `onprem1`).

2. Edit `/etc/hosts` on the server and each compute node to resolve hostnames:

   ```
   [ec2-user ~]$ sudo vi /etc/hosts
   ```

   Add the entries to the bottom of the file in the following format:

   ```
   <private IP address> <hostname>
   ```

   In the following example, the Moab server is called `bursting-sched` and the on-prem compute node is called `onprem1`.

   ```
   172.31.26.10 bursting-sched
   172.31.6.77 onprem1
   ```

3. Copy the Moab licenses and the `nodus` public and private key files to the server.

   ```
   $ scp -i nodus.pub nodus.pem moab-rlm.lic moab-rlm-elastic-tracking.lic ec2-
   user@<public IP address>:~
   ```

4. On the server, move the `moab-rlm.lic` file to `/opt/moab/etc/`.

   ```
   [ec2-user ~]$ sudo mv moab-rlm.lic /opt/moab/etc/
   ```

5. On the server, move the `moab-rlm-elastic-tracking.lic` file to `/opt/rlm`.

   ```
   [ec2-user ~]$ sudo mv moab-rlm-elastic-tracking.lic /opt/rlm
   ```

6. On the server, move the `nodus.pem` file to `/opt/nodus-cloud-cli`.

   ```
   [ec2-user ~]$ sudo mv nodus.pem /opt/nodus-cloud-cli
   ```

7. On the server, append the contents of the `nodus.pub` file to the end of the `/home/ec2-user/.ssh/authorized_keys` file.

   ```
   [ec2-user ~]$ cat nodus.pub >> ~/.ssh/authorized_keys
   ```

8. Restart the services on the `bursting-sched` server: Moab, Torque server, RLM Server.

   ```
   [ec2-user ~]$ sudo systemctl restart rlm.service
   [ec2-user ~]$ sudo systemctl restart trqauthd.service
   [ec2-user ~]$ sudo systemctl restart pbs_server.service
   [ec2-user ~]$ sudo systemctl restart moab.service
   ```

9. On each compute node, mount NFS shares.

```
[ec2-user ~]$ sudo mount -t nfs bursting-sched:/var/share /mnt/share
[ec2-user ~]$ sudo mount -t nfs bursting-sched:/home /home
```

10. On compute nodes, restart the services on the server.

```
[ec2-user ~]$ sudo systemctl restart pbs_mom.service
```

11. On the server, add each instance to Torque as a compute node:

```
[ec2-user ~]$ sudo /usr/local/bin/qmgr -c 'create node onprem1'
```

12. If you're planning on using the Viewpoint Web Portal, set the password for the `ec2-user`.

```
[root ~]$ passwd ec2-user
```

1.0.9 Setup Verification

To verify your setup:

1. Run the following command on the Moab server to show the nodes Torque recognizes and their state.

```
[ec2-user ~]$ pbsnodes
```

2. Run the following command on the Moab server to show the nodes Moab recognizes and their state.

```
[ec2-user ~]$ sudo /opt/moab/bin/mdiag -n
```

For additional help or inquiries, contact your sales representative.

1.0.10 Setting up Viewpoint

If you plan on using Viewpoint, there are a few steps needed in order to add the Viewpoint license.

1.0.10.A Install the Viewpoint License

Make sure your security group opens port 8081.

Do the following:

1. Using a web browser, navigate to your Viewpoint instance (`http://<bursting-sched>:8081`; where `<bursting-sched>` is the public IP address of the Moab/Viewpoint instance in AWS).
2. Log in as the `viewpoint-admin` using the password `secret1`.

3. Click the Licensed Features link on the left side of the page.
4. On the Licensed Features page, locate the Viewpoint Host ID (under the Browse button).
5. Click Browse, navigate to where you saved the Viewpoint License file, and then click Open.
6. Click Upload.

Once the license file has uploaded, the Viewpoint License information shows green check boxes for your licensed features and displays the path to your uploaded license file under the Viewpoint Host ID information.

1.0.10.B Log In to Viewpoint

To log in to Viewpoint:

1. Using a web browser, navigate to your Viewpoint instance (`http://<bursting-sched>:8081`; where `<bursting-sched>` is the IP address or name of the Moab/Viewpoint instance in AWS).
2. Log in as the `ec2-user` user using the password `secret1` if the Rapid Deployment script was used to set the password. Otherwise use the password that was set in step 12 of 1.0.8 Manual Setup - page 9.
3. Using the `ec2-user` account you can submit jobs and view cluster details.

1.0.11 Bursting

1.0.11.A Configure Bursting to Your AWS Account

In order to do bursting, the AWS account credentials must be added to the server.

Connecting to the server is done over SSH. Connect using the following command:

```
$ ssh -i nodus.pem ec2-user@<public IP for bursting-sched>
```

Add your AWS Access Key and AWS Secret Access Key to the `/opt/nodus-cloud-cli/cloud_credentials/aws/credentials.json` file in the following format:

```
{
   "access_key": "ABCDEFGHIJKLMNOPQRST",
   "secret_key": "AbcDef/gHiJkLmopxvc3/8AbCDEvpRLJ7ugjK4vle"
}
```

If you have not generated the AWS Access Key and Secret Access Key, see 1.0.4 Prerequisites - page 6.

1.0.11.B Building Stacks

Once the AWS Access keys are set, stacks (configured AMIs) are ready to be configured in NODUS. Note the Test Drive Compute Node AMI ID (in EC2 -> AMIs). When you're ready to create the stack, do the following:

1. Open the `/opt/nodus-cloud-cli/examples/stacks/ami-build/ami-build.json` file and update the AMI ID of the Compute AMI and region.

```
{
   "name": "ami-build",
   "version": "1",
   "image": {
      "source_ami": "ami-5b2f3e3b",
      "ssh_username": "ec2-user",
      "region": "us-west-1"
   },
   "bootstrap": "ami-build-bootstrap.sh",
   "tasks": []
}
```

   This specifies the region is where this image will reside. This image is meant to be used for the cloud compute node.

   To determine your region, open `https://aws.amazon.com/console/` in a web browser and click "Sign In to the Console." After you have logged in, your browser will be redirected to a new URL that contains the region ID as a prefix.

```
https://<region id>.console.aws.amazon.com
```

   For example if you are logged into the N. California region, your AWS URL will look like this:

```
https://us-west-1.console.aws.amazon.com
```

   The prefix, and thus the region ID in this case, is `us-west-1`.

2. Build the stack by running the following commands:

```
[root ~]$ cd /opt/nodus-cloud-cli
[root ~]$ nodus-run stack/build.js examples/stacks/ami-build/ami-build.json
credentials=cloud_credentials/aws/credentials.json
```

   This command will start an instance in the cloud called `Packer Builder`, which will start the AMI and set it up inside NODUS. It will then create a new AMI in AWS called `nodus-stack` and return a stack ID (i.e. `37945db6-26d8-4192-9eda-a7d5ae18de33`) that is needed in the Moab configuration.

1.0.11.C Configure/Customize Moab Backlog Bursting Behavior

In the `/opt/moab/etc/elastic.cfg` file is where the bursting triggers are found, see the trigger on the `QOSCFG[elastic]` line.

This trigger controls backlog bursting. Paste in the stack ID generated when the stack was built and the path to the key file:

```
QOSCFG[elastic] TRIGGER=EType=threshold,AType=exec,TType=elastic,Action="/opt/nodus-
cloud-cli/elastic.py -g $REQUESTGEOMETRY -f cloud -p aws -s 37945db6-26d8-4192-9eda-
a7d5ae18de33 -i t2.micro -k /opt/nodus-cloud-
cli/nodus.pem",Threshold=BACKLOGCOMPLETIONTIME>30,RearmTime=15:00,timeout=5:00
```

The `REQUESTGEOMETRY` variable controls the size of the burst. `num_instances` is the number of VMs that will be created for each burst and `time_to_live` indicates how long each VM will live.

```
<num_instances>@<time_to_live>
```

For example:

```
5@3:00:00
```

means that each burst will create 5 VM instances and each instance will live up to 3 hours.

You can also adjust the instance's size (default `t2.micro`). The `-f` flag creates a node feature and `-p` sets the instance's prefix name (default is `aws`). The `-a type=name` flag lets you specify the access control list, if the bursted nodes are meant to be for a specific user or group (e.g. `-a user=ted`).

After you're finished editing `elastic.cfg`, restart Moab:

```
[root ~]$ mschedctl -R
```

1.0.11.D Tuning Backlog Bursting

Backlog bursting happens when the `BacklogCompletionTime` in the threshold trigger is reached. You can see the `BacklogCompletionTime` using *`mdiag -T -v`*:

```
[root@bursting-sched nodus-cloud-cli]# mdiag -T -v
TrigID               Object ID              Event    TType  AType
ActionDate       State
-------------------- -------------------- -------- -------- ------ ------------------
-- -----------
2                    qos:elastic          threshol  elastic   exec
-     Blocked
Flags:          multifire,globaltrig
BlockUntil:     INFINITY  ActiveTime:  ---
Timeout:        00:05:00
Threshold:      BacklogCompletionTime > 30.00  (current value: 0.00)
Trigger Type:   elastic
RearmTime:      00:15:00
Action Data:    /opt/nodus-cloud-cli/elastic.py -g $REQUESTGEOMETRY -f cloud -p aws -s
b7b8b868-7fef-4c08-80fe-a2e30a19151f -i t2.micro -k /opt/nodus-cloud-cli/nodus.pem
NOTE:  trigger cannot launch - threshold not satisfied - requires usage 0.000000 >
30.000000
```

```
4                     node:DEFAULT              end  elastic   exec
-     Blocked
Flags:          globaltrig
BlockUntil:     INFINITY  ActiveTime:  ---
Trigger Type:   elastic
Action Data:    /opt/nodus-cloud-cli/elastic.py -r $OID
```

As jobs queue up in the idle queue, use the `BacklogCompletionTime` in the *`mdiag -T -v`* output to determine the threshold setting for a burst to occur. The `ec2-user` user ID can be used to submit jobs:

```
[ec2-user ~]$ echo sleep 30 | msub -l walltime=30
```

1.0.11.E Configure/Customize Moab OnDemand Bursting Behavior

In the `/opt/moab/etc/elastic.cfg` file there is also the trigger that does OnDemand bursting. See the trigger on the `JOBCFG[aws-t2-micro]` line. This is a job template that can be modified to create multiple job templates with the desired settings for OnDemand bursting (number of instances, instance type, stack ID, etc.).

Paste in the stack ID generated when the stack was built and the path to the key file:

```
JOBCFG[aws-t2-micro]  FLAGS=aws-cloud SELECT=TRUE
JOBCFG[aws-t2-micro]  TRIGGER=EType=create,AType=exec,Action="/opt/nodus-cloud-
cli/elastic-ondemand.py -f aws-ondemand -j $JOBID -o $OWNER -s 37945db6-26d8-4192-
9eda-a7d5ae18de33 -i t2.micro -g 2@1:00:00 -k /opt/nodus-cloud-
cli/nodus.pem",timeout=5:00,RearmTime=10:00
```

You can customize the instance type with `-i`, the number of instances and their time to live with `-g` (request geometry), and the stack ID with `-s`. You can also add more job templates with different values. For example:

```
JOBCFG[aws-t2-xlarge]  FLAGS=aws-cloud SELECT=TRUE
JOBCFG[aws-t2-xlarge]  TRIGGER=EType=create,AType=exec,Action="/opt/nodus-cloud-
cli/elastic-ondemand.py -f aws-ondemand -j $JOBID -o $OWNER -s 37945db6-26d8-4192-
9eda-a7d5ae18de33 -i t2.xlarge -g 1@1:00:00:00 -k /opt/nodus-cloud-
cli/nodus.pem",timeout=5:00,RearmTime=10:00
```

After you're finished edited the `elastic.cfg`, restart Moab:

```
[root ~]$ mschedctl -R
```

1.0.11.F OnDemand Example Job

The following can be used as an example job for OnDemand bursting:

```
[ec2-user ~]$ echo sleep 30 | msub -l walltime=30,template=aws-t2-micro,procs=2 -h
```

Make sure to submit the job with a hold (`-h`) so that Moab will not immediately try to schedule the job. The `/opt/nodus-cloud-cli/elastic-ondemand.py` bursting script will release the hold on the job once the instances are spun up.

For more in OnDemand bursting see Configuring On-Demand Elastic Computing For Use With NODUS in the *Moab/NODUS Cloud Bursting Administrator Guide*.

1.0.12 Moab Cheat Sheet

Commonly Used Moab Commands.

Command	Description
showq	Show running/idle jobs.
mdiag -n	Show nodes.
mdiag -T -v	Show triggers and trigger details.
msub	Submit Job.
canceljob	Cancel jobs.

Example Job Submissions

```
msub <job_script>
```

```
msub -l walltime=30 <job_script>
```

```
msub -l walltime=30,procs=2 <job_script>
```

Simple Job Script

```
#!/bin/bash
pbsdsh -u hostname
```

Runs the `hostname` command on all nodes in the job.

1.0.13 Troubleshooting

1.0.13.A Manual Bursting

Testing bursting can be done manually using the `/opt/nodus-cloud-cli/elastic.py` script, shown below. This script is what Moab uses to call out to NODUS. The script can also be run manually, especially to test the bursting mechanism.

```
[ec2-user ~]$ sudo su
[root ~]$ cd /opt/nodus-cloud-cli
```

```
[root ~]$ ./elastic.py -g 5@3:00:00 -f cloud -p aws -s e4816979-ceb8-4aa9-8601-
ce1a4e371af8 -i t2.micro -k /opt/nodus-cloud-cli/nodus.pem
```

This will launch five AWS instances with a TTL of three hours, using the prebuilt stack ID on a `t2.micro` instance. Once the node comes up, the AWS instances should join the cluster and show up in the results of *`pbsnodes`* and *`mdiag -n`*.

1.1 Moab/NODUS Cloud Bursting JSON Scripts

1.1.1 NODUS_Policies_Bursting.JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:RevokeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:CreateKeyPair",
                "ec2:DescribeInstanceAttribute",
                "ec2:RegisterImage",
                "ec2:CreateImage",
                "ec2:DescribeInstanceCreditSpecifications",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:ModifyInstanceCreditSpecification",
                "ec2:DescribeVpcs",
                "ec2:CreateSecurityGroup",
                "ec2:DescribeVolumes",
                "ec2:ModifyInstanceAttribute",
                "ec2:DescribeSubnets",
                "ec2:DescribeInstanceStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:CreateVolume"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances",
                "ec2:StartInstances",
```

```
                "ec2:RunInstances",
                "ec2:StopInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:launch-template/*",
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:placement-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*::image/*"
            ]
        },
        {
            "Sid": "VisualEditor4",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:instance/*"
        }
    ]
}
```

1.1.2 NODUS_Policies_Stack_Build.JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:TerminateInstances",
                "ec2:StartInstances",
                "ec2:CreateTags",
                "ec2:RunInstances",
                "ec2:StopInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:launch-template/*",
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:placement-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*::image/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:CreateKeyPair",
```

```
                "ec2:CreateSecurityGroup",
                "ec2:CreateImage",
                "ec2:DescribeKeyPairs",
                "ec2:DeleteKeyPair",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeInstanceStatus"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
                "ec2:DeleteSecurityGroup",
                "ec2:UpdateSecurityGroupRuleDescriptionsIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        }
    ]
}
```

1.1.3 Rapid Deployment Script (td_rapid_deployment.sh)

```
#!/bin/bash

# td_rapid_deployment.sh: easily setup Test Drive instances in preparation for
bursting
# HOSTS FILE FORMAT:
# <PUBLIC IP> <PRIVATE IP> <HOSTNAME>

HOSTS_FILE=$1
KEY_FILE=$2

if [ -z $KEY_FILE ]; then
  USE_KEY=""
else
  USE_KEY="-i $KEY_FILE"
fi
echo "$USE_KEY"

read -r -d '' ETC_HOSTS_PREFIX << EOM
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

18.219.74.231 app.nodusplatform.com
EOM

HOSTS=`cat $HOSTS_FILE`
PUB_IP=($(echo "$HOSTS" | awk '{print $1}'))
PRIV_IP=($(echo "$HOSTS" | awk '{print $2}'))
HOSTNAME=($(echo "$HOSTS" | awk '{print $3}'))
ETC_HOSTS=`echo "$HOSTS" | awk '{print $2 " " $3}'`
SERVERIP=`echo $HOSTS | grep bursting-sched | awk '{print $1}'`
#echo "$ETC_HOSTS"

for ((i=0;i<${#PUB_IP[@]};++i)); do
  printf "*** Configuring %s %s %s ***\n" "${HOSTNAME[i]}" "${PUB_IP[i]}" "${PRIV_IP
```

```
[i]}"
  echo "Setting hostname"
  ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo hostnamectl
set-hostname ${HOSTNAME[i]}"
  echo "Updating /etc/hosts"
  ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo su -c 'echo
\"$ETC_HOSTS_PREFIX\" > /etc/hosts'"
  ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo su -c 'echo
\"$ETC_HOSTS\" >> /etc/hosts'"
  if [ ${HOSTNAME[i]} == "bursting-sched" ]; then
    echo "Configuring Moab Server: ${HOSTNAME[i]}"
    echo "Copying up moab licenses"
    # legacy license
    #scp -o "StrictHostKeyChecking no" $USE_KEY moab.lic ec2-user@${PUB_IP[i]}:~/
    #ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mv
moab.lic /opt/moab/etc"
    # RLM license
    scp -o "StrictHostKeyChecking no" $USE_KEY moab-rlm.lic ec2-user@${PUB_IP[i]}:~/
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mv moab-
rlm.lic /opt/moab/etc"
    scp -o "StrictHostKeyChecking no" $USE_KEY moab-rlm-elastic-tracking.lic ec2-
user@${PUB_IP[i]}:~/
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mv moab-
rlm-elastic-tracking.lic /opt/rlm"
    scp -o "StrictHostKeyChecking no" $USE_KEY nodus.pem ec2-user@${PUB_IP[i]}:~/
    scp -o "StrictHostKeyChecking no" $USE_KEY nodus.pub ec2-user@${PUB_IP[i]}:~/
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mv
nodus.pem /opt/nodus-cloud-cli"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "cat nodus.pub >>
~/.ssh/authorized_keys"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo su -c 'echo
\"secret1\" | passwd \"ec2-user\" --stdin'"
    echo "Restarting services"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart rlm"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart trqauthd.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart pbs_server.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart moab.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart postgresql.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart acfileman.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart mongod.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart tomcat.service"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
restart httpd.service"
  else
    echo "Configuring Compute Node: ${HOSTNAME[i]}"
    echo "Mounting NFS share"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mount -t
nfs bursting-sched:/var/share /mnt/share"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo mount -t
nfs bursting-sched:/home /home"
    printf "Adding %s to cluster\n" "${HOSTNAME[i]}"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@$SERVERIP "sudo
/usr/local/bin/qmgr -c 'create node ${HOSTNAME[i]}'"
    echo "Restarting pbs_mom"
    ssh -o "StrictHostKeyChecking no" $USE_KEY ec2-user@${PUB_IP[i]} "sudo systemctl
```

```
restart pbs_mom.service"
  fi
done
```