Bright Cluster Manager 8.0

Cloudbursting Manual

Revision: 4aa1881

Date: Wed Sep 27 2023

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *Cloudbursting Manual* for Bright Cluster Manager 8.0.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage the cloud capabilities of Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 8.0 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for the basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Big Data Deployment Manual* describes how to deploy Big Data with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.
- The *Machine Learning Manual* describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website `https://support.brightcomputing.com`. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 13.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Introduction

In weather, a cloudburst is used to convey the idea that a sudden flood of cloud contents takes place. In cluster computing, the term *cloudbursting* conveys the idea that a flood of extra cluster capacity is made available when needed from a cloud computing services provider such as Amazon.

Bright Cluster Manager implements cloudbursting for two scenarios:

1. A "Cluster On Demand", or a "pure" cloud cluster (chapter 2). In this scenario, the entire cluster can be started up on demand from a state of non-existence. All nodes, including the head node, are instances running in a coordinated manner entirely inside the cloud computing service.

2. A "Cluster Extension", or a "hybrid" cloud cluster (chapter 3). In this scenario, the head node is kept outside the cloud. Zero or more regular nodes are also run outside the cloud. When additional capacity is required, the cluster is extended via cloudbursting to make additional nodes available from within the cloud.

Chapters 2 and 3 deal with mainly the GUI configuration of the Cluster On Demand and Cluster Extension scenarios.

Chapter 4 looks at mainly command line tools for configuration of the Cluster On Demand and Cluster Extension scenarios, considering mainly AWS.

Chapter 5 looks at Cluster Extension for Azure.

Chapter 6 discusses some miscellaneous aspects of cloudbursting.

Chapter 7 describes the concepts behind the networking of legacy and current cloud instances supported by Bright Cluster Manager on the Amazon VPC infrastructure.

2

Cluster On Demand Cloudbursting With AWS

Requirements

If the cloud provider is Amazon, then Cluster On Demand cloudbursting (the case of starting up a "pure" cloud cluster) requires:

- an Amazon account
- registration on the Bright Computing Customer Portal website at `http://customer.brightcomputing.com/Customer-Login`
- a Bright Cluster Manager product key. The key is obtained at the Customer Portal website specifically for a Cluster On Demand setup, from the `Burst!` menu. This key is later activated when the license is installed (section 2.2) on the head node. The head node and regular nodes in this case are in the cloud.

Steps

The following steps are then carried out to start up the head node and regular nodes of the cloud cluster:

- a head node instance is launched from a browser, using the Amazon management console (section 2.1)
- the head node instance is logged into via `ssh` and the cluster is configured (section 2.2)
- the regular nodes are started up from the head node using `cmsh` or Bright View to power them up (section 2.4)

These steps are now covered in more detail.

2.1 Cluster On Demand With AWS: Launching The Head Node

Launching a head node from within Amazon is covered in this section.

2.1.1 Getting To The "`Launch Instance`" Button

The Amazon management console can be logged into from `https://console.aws.amazon.com/console/` by using the e-mail address and password of the Amazon account (figure 2.1).

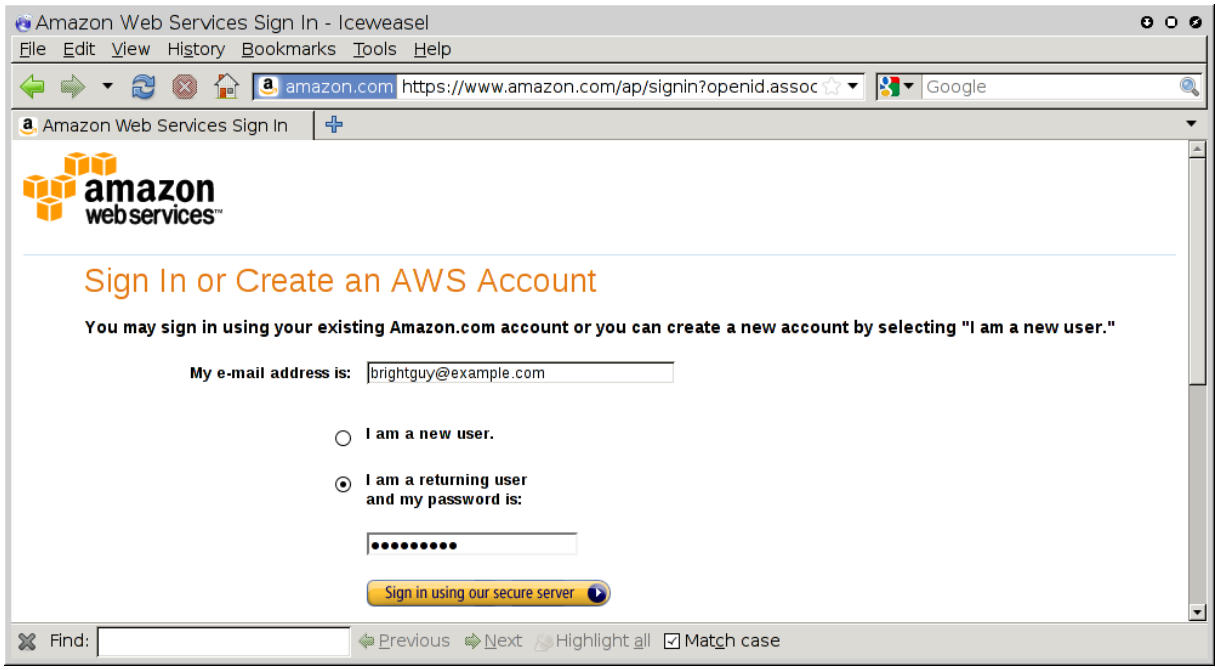

Figure 2.1: Logging Into The Amazon Management Console

By default, on login, the management console displays a list of accessible Amazon web services, including the *Elastic Compute Cloud* (`EC2`) (figure 2.2):

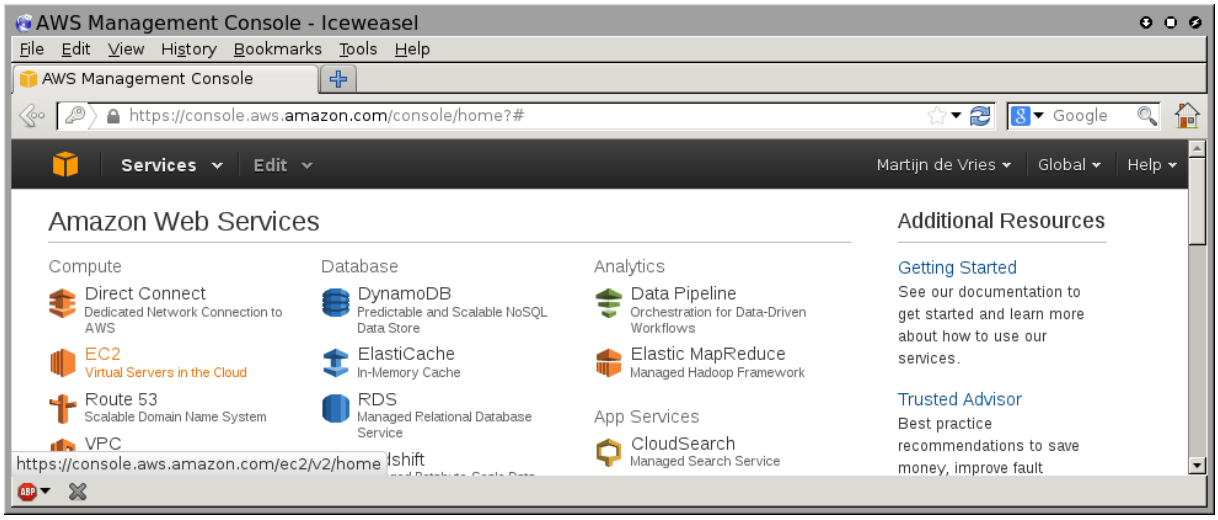

Figure 2.2: Amazon Management Console: Accessible Services

To set up the Cluster On Demand cluster, the `EC2` service within the `Compute` grouping should be clicked. This brings up the `EC2 Dashboard`, which is also the top link of a resource tree that is displayed in a `Navigation` pane (figure 2.3).

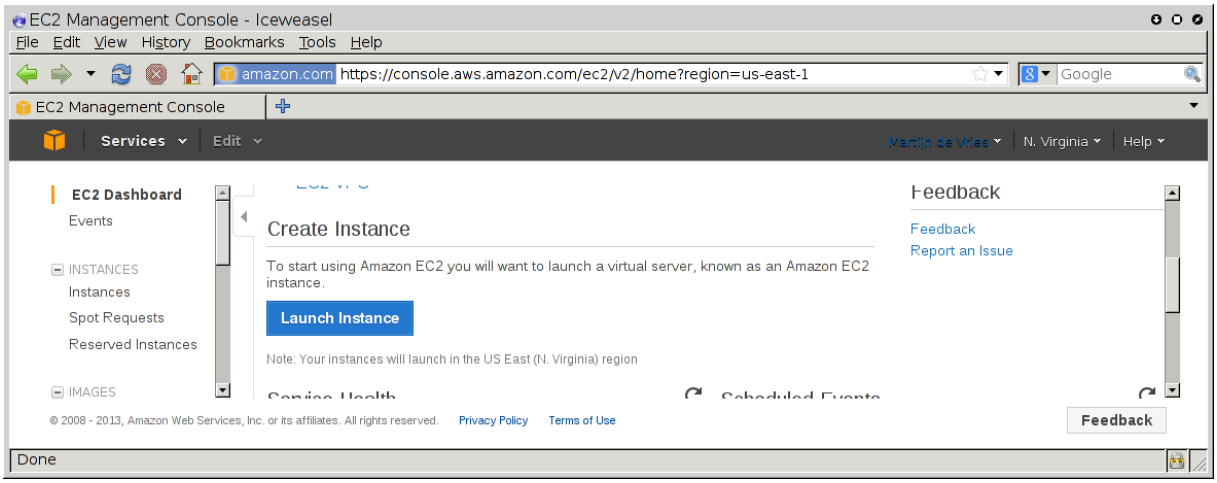

Figure 2.3: The EC2 Dashboard With The "`Launch Instance`" Button

In the main pane of the dashboard is the `Launch Instance` button. Clicking it starts up Amazon's Launch Instance Wizard. Amazon documentation for the wizard is at `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html`.

Using the wizard to launch a head node instance is described next.

2.1.2 Launching The Head Node Instance

To start a Cluster On Demand cluster, a head node instance must first be launched. This can be done as follows:

- `Step 1: Choose an Amazon Machine Image (AMI)`: The first step in the wizard offers a choice of `Select` buttons to launch an instance from an AMI image (figure 2.4).

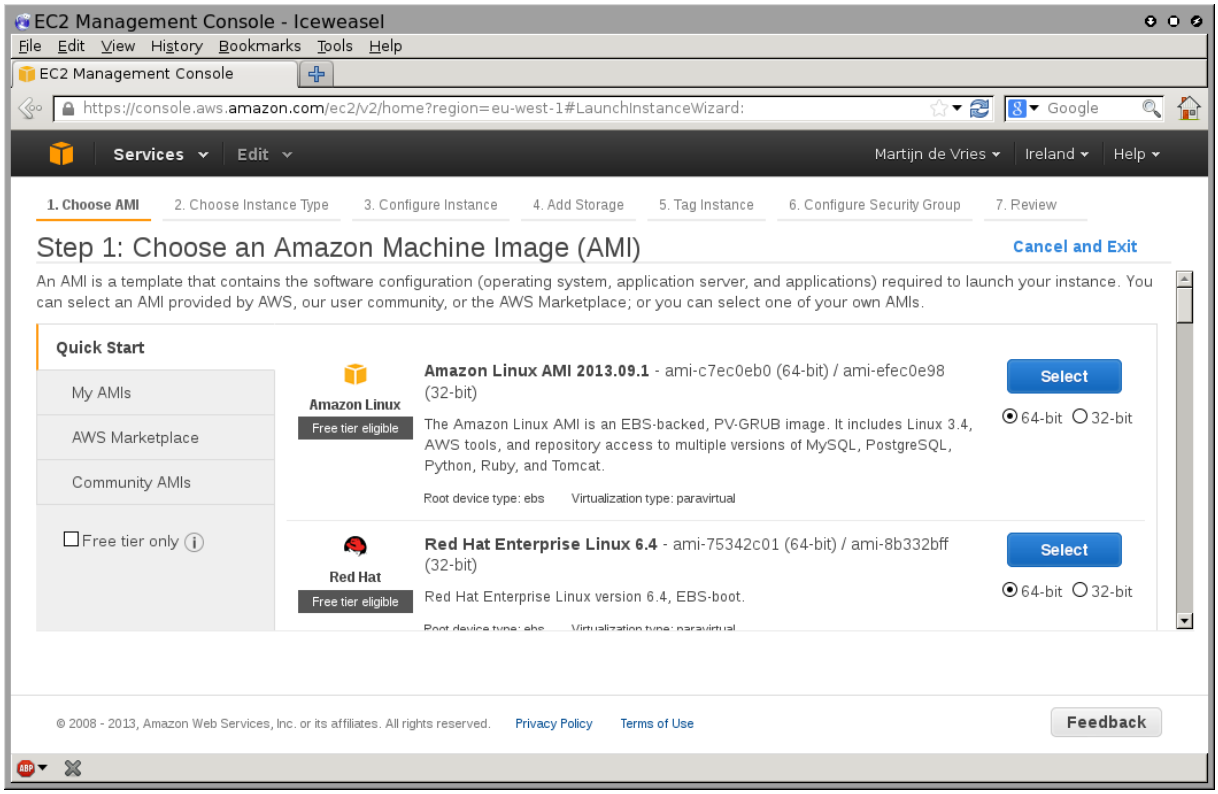

Figure 2.4: EC2: Choosing An AMI, Step 1

The default AMIs can be ignored. Clicking on the `Community AMIs` link in the left navigation pane brings up a new display of community AMIs. Entering a search text of "`brightheadnode`" then shows only the AMIs appropriate for a Bright Cluster Manager head node instance in a Cluster On Demand cluster. From Bright Cluster Manager 7.3 onward, only AMIs that use hardware virtualization (`hvm`) type AMIs are supported, due to advances in hardware technology which have made paravirtualization (`pvm`) type AMIs less popular.

Clicking on the `Select` button for the appropriate HVM head node AMI then brings up the next step in the launch wizard:

- `Step 2: Choose an Instance Type`: This displays a *t2 micro* instance by default (figure 2.5).

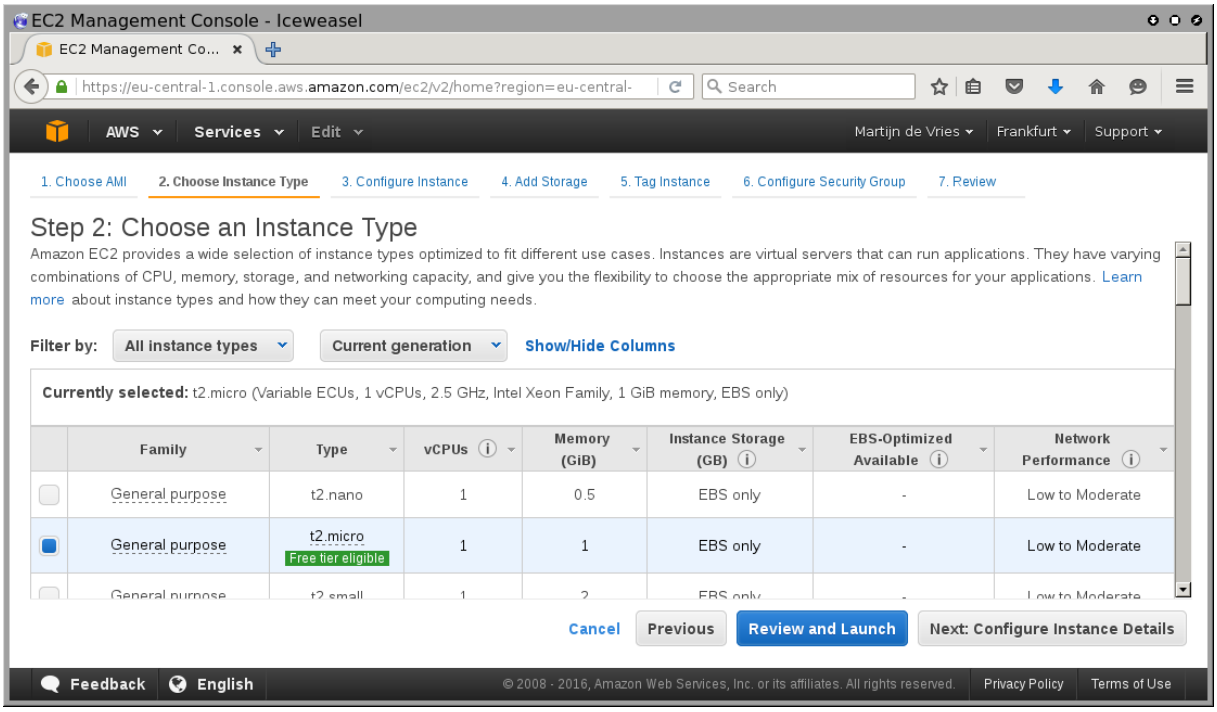

Figure 2.5: EC2: Choosing An AMI, Step 2

The `t2.micro` is one of the smallest and least powerful type of instances that a head node can run as, and is only useful for quite minor testing. It is likely to be overwhelmed when doing any significant work—even provisioning a regular cloud node from a `t2.micro` head node results in resource exhaustion. A more reasonable node type to use for testing is therefore the `t2.medium` type, which is an Amazon type that is part of their `General purpose` family.

- `Step 3: Configure Instance Details`: Among other instance options, this optional step allows the following to be set:
  - `Purchasing option`: spot instances can be requested (section 6.3.1)
  - `Network`: A VPC-defined network owned by the user must be selected
  - `Subnet`: A subnet that is associated with an *availability zone* can be chosen, if preferred. Nodes within the same availability zone can connect with low latency and high bandwidth to each other. They are also isolated from other availability zones in the same region, which reduces the risk of network outages from another zone affecting them. By default, no preference is specified for the head node, nor for the cloud nodes later. This is because spot pricing can increase as nodes from an availability zone become scarce, which may conflict with the intention of the administrator. The default availability setting for a cloud account can be set in `cmsh` from within `cloud` mode:

    Example

    ```
    [bright80->cloud[Spare Capacity]]% set defaultavailabilityzone
    ```

  - `Auto-assign Public IP`: Enables or disables public IP addresses. Enabling this allows the administrator to connect to the instance via an `ssh` client, so it should be enabled.
  - `Shutdown behavior`: this decides if the instance should be stopped (kept around in the cloud, but in a shutdown state) or terminated (removed from the cloud itself) when a shutdown command is given to the instance within the instance.

Steps 4 to 6 that follow are optional and can be skipped, by going ahead to `Step 7: Review Instance Launch`.

- `Step 4: Add Storage`: Among other storage options, this optional step allows the following options to be set:
  - `Size (GiB)`: The size of storage to be added to the instance
  - `Volume Type`: Whether the storage is Magnetic, Provisioned SSD, or General Purpose SSD
  - `Device`: A device name, chosen from `/dev/sdb` onwards, since `/dev/sda` is already taken
  - `Delete on Termination`: Whether the device is deleted when the instance terminates

  By default, the instance has a `Volume Type` called `Root`, which is a special EBS volume. It has a default `Size (GB)` of 80, which can be edited.

  For most instances other than `micro`, a certain amount of ephemeral storage is provided by Amazon for free, and can then be set for the root device in the `Storage Device Configuration` options of this screen. The EBS and ephemeral storage types are described in section 3.2.1.

- `Step 5: Add Tags`: This optional step allows the addition of metadata to an instance, via assignment of key-value pairs. A default key of `Name` is presented, and the administrator can put in a name for the instance as the associated value. The associated value can be arbitrary.

- `Step 6: Configure Security Group`: This optional step allows a *security group* to be defined. A security group is a configuration that defines how network access to the instance is allowed. By default all access to the instance is blocked, apart from SSH access.
  - Default: All SSH inbound is allowed. This means that `cmsh` can be used to control the Cluster On Demand cluster via SSH just like a regular cluster.

  It is a good idea, for security reasons, to restrict the source addresses allowed for inbound SSH. Changing it from 0.0.0.0/0, to instead be the IP address of the source for the current connection to the website should make sense for initial testing purposes. The change can be selected by choosing the `My IP` option from the `Custom` options of the `Source` column.

  Inbound connections can be defined, based on protocol, packet type, port, and source, using a CIDR specification. For example, allowing inbound connections via TCP port 8081 from anywhere (0.0.0.0/0) allows Bright View to communicate from anywhere via HTTP with the default CMDaemon back end on the head node.

  The default security group setting should be restricted by the administrator if Bright View is to be used from multiple locations to control the cluster (section 2.3).

- `Step 7: Review Instance Launch`: The configuration so far can be reviewed.

  If coming straight from step 4, then a warning is displayed, which says that the instance is reachable from any IP address on the internet. Typically, a restriction should be placed by the user on the source addresses that can reach the instance. The source address is typically changed by the user from the default IP address of `0.0.0.0/0`, to the IP address of the user.

  On clicking the `Launch` button, a pop-up dialog for "`Select an existing key pair or create a new key pair`" is displayed (figure 2.6).

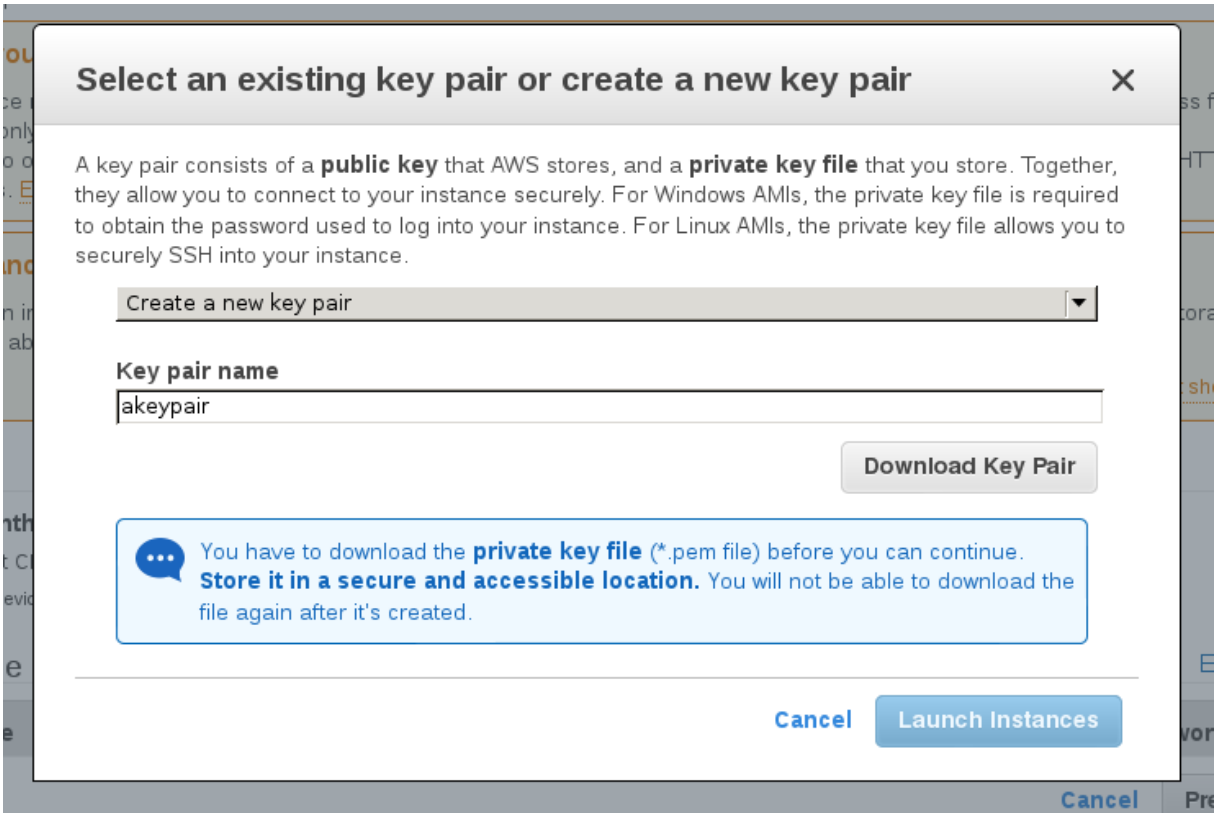

Figure 2.6: EC2: Choosing An AMI, Step 7 - Keypair generation/creation dialog

This dialog allows the creation and storage of a cryptographic key pair. It can alternatively allow an existing pair to be used from the "`Select a key pair`" selection. The private key of the key pair is used in order to allow SSH access to the head node instance when it is up and running.

After the instance is launched, the web session displays a window informing that the instance is being started up.

2.1.3 Managing A Head Node Instance With The AWS EC2 Management Console

A *newly-launched* head node instance, after it is fully up, is a fully-booted and running Linux instance, but it is not yet a fully-configured head node. That is, it is capable of running Bright Cluster Manager, but it is not yet running it, nor is it working with compute nodes at this point. The steps to make it a fully-configured head node are covered in section 2.2.

For now, the newly-launched head node instance can be watched and managed without Bright Cluster Manager in the following ways.

Status checking via instance selection from instances list:

Clicking the `Instances` menu resource item from the navigation pane opens up the "`Instances`" pane. This lists instances belonging to the account owner. An instance can be marked by ticking its checkbox. Information for the selected instance is then displayed in the lower main pane (figure 2.7).

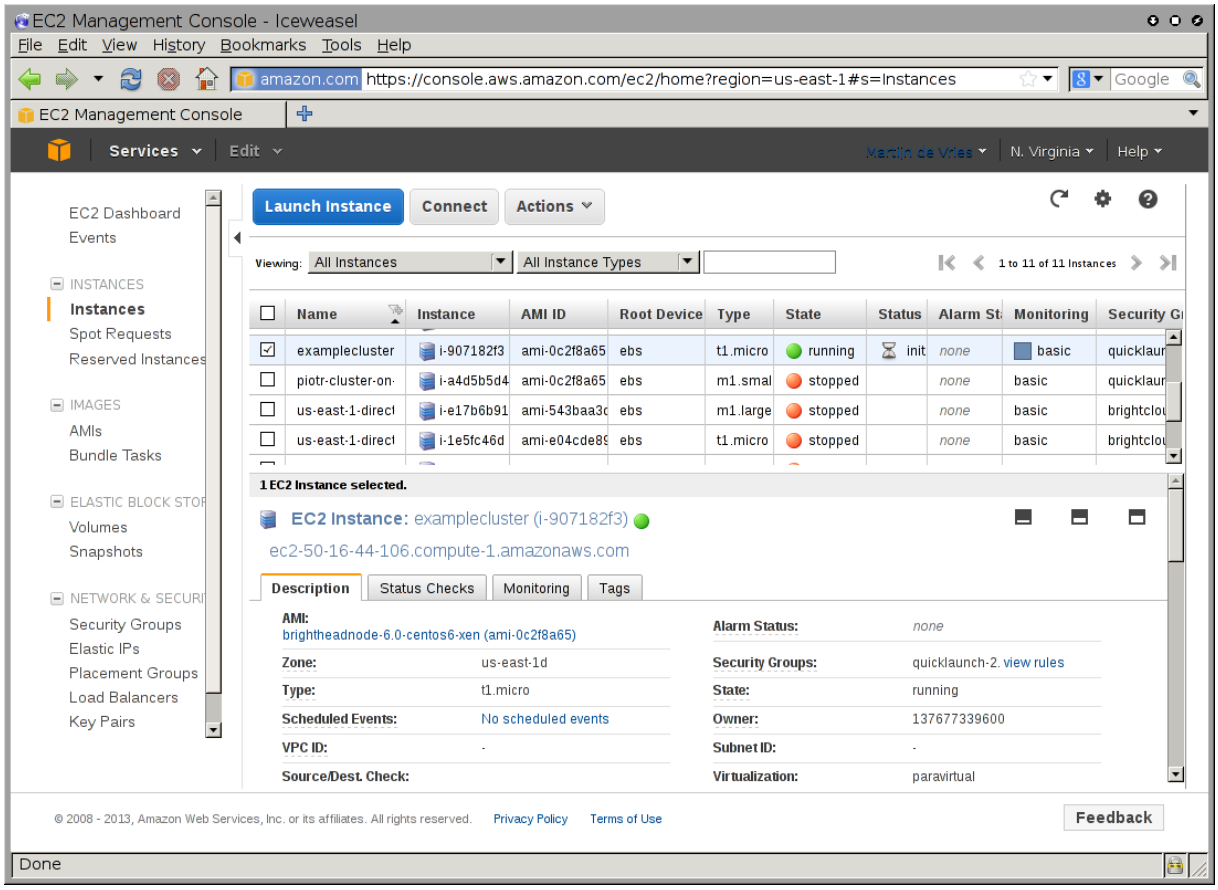

Figure 2.7: The EC2 Instances List

System (Amazon machine infrastructure) and instance (instance running under the infrastructure) reachabilities are similarly shown under the neighboring "`Status Checks`" tab (figure 2.8).

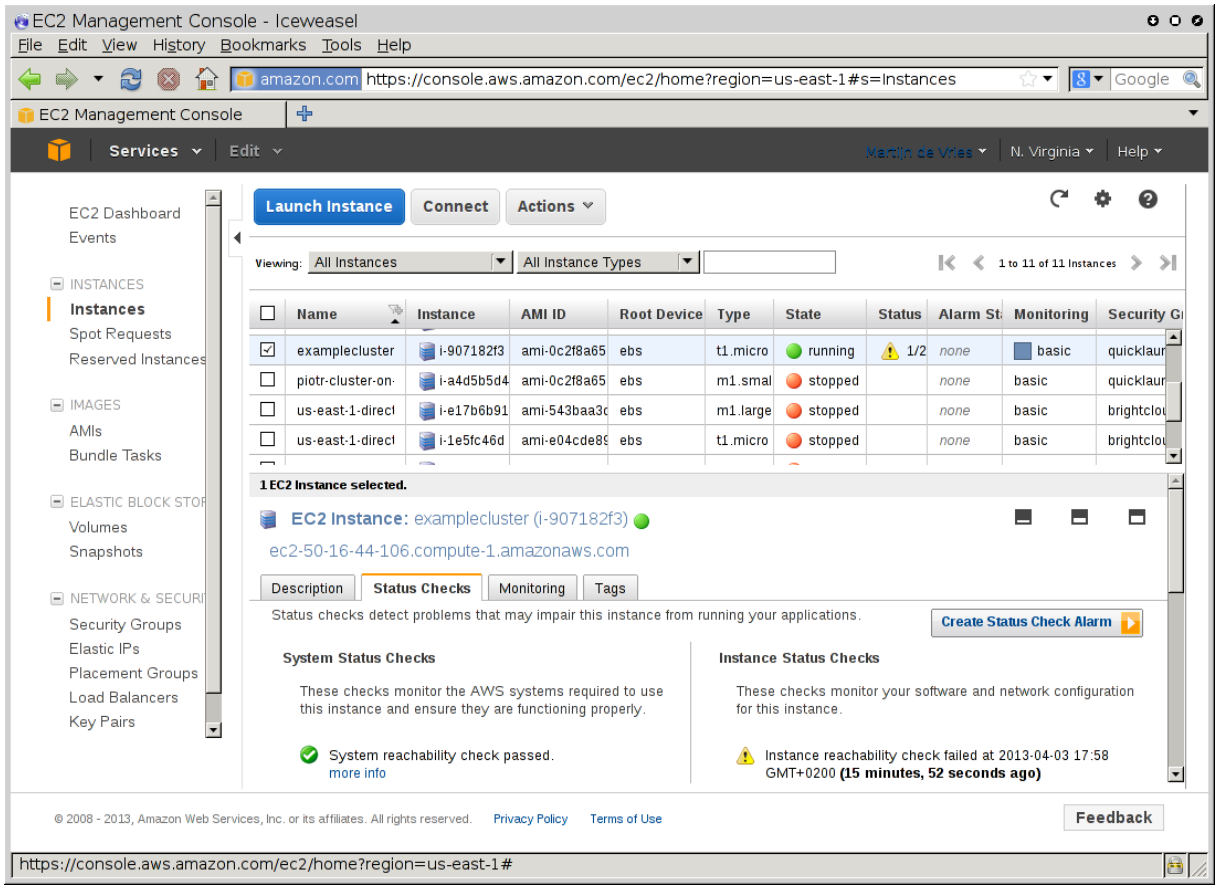

Figure 2.8: Reachability Status For An EC2 Instance

Acting on an instance from the AWS EC2 Management Console:
An instance can be marked by clicking on it. Clicking the `Actions` button near the top of the main center pane, or equivalently from a right-mouse-button click in the pane, brings up a menu of possible actions. These actions can be executed on the marked instance, and include the options to `Start`, `Stop` or `Terminate` the instance.

Connecting to an instance from the AWS EC2 Management Console:
A marked and running instance can have an SSH connection made to it. Clicking on the `Connect` button near the top of the main center pane displays a pop-up text that guides the user through the connection options for a running instance. These connection options are via:

- **a standalone SSH client**

  There is further documentation on this at:

  - `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html` for Linux clients
  - `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html` for PuTTY users

- **a browser-based Java SSH client, MindTerm**

  There is further documentation on this at:

  - `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/mindterm.html`

Most administrators should find the pop-up text enough, and the further documentation unnecessary.

The standalone SSH client help text displays instructions (figure 2.9) on how to run `ssh` from the command line to access the marked instance. If the launched head node is fully up then a login using those instructions succeeds.

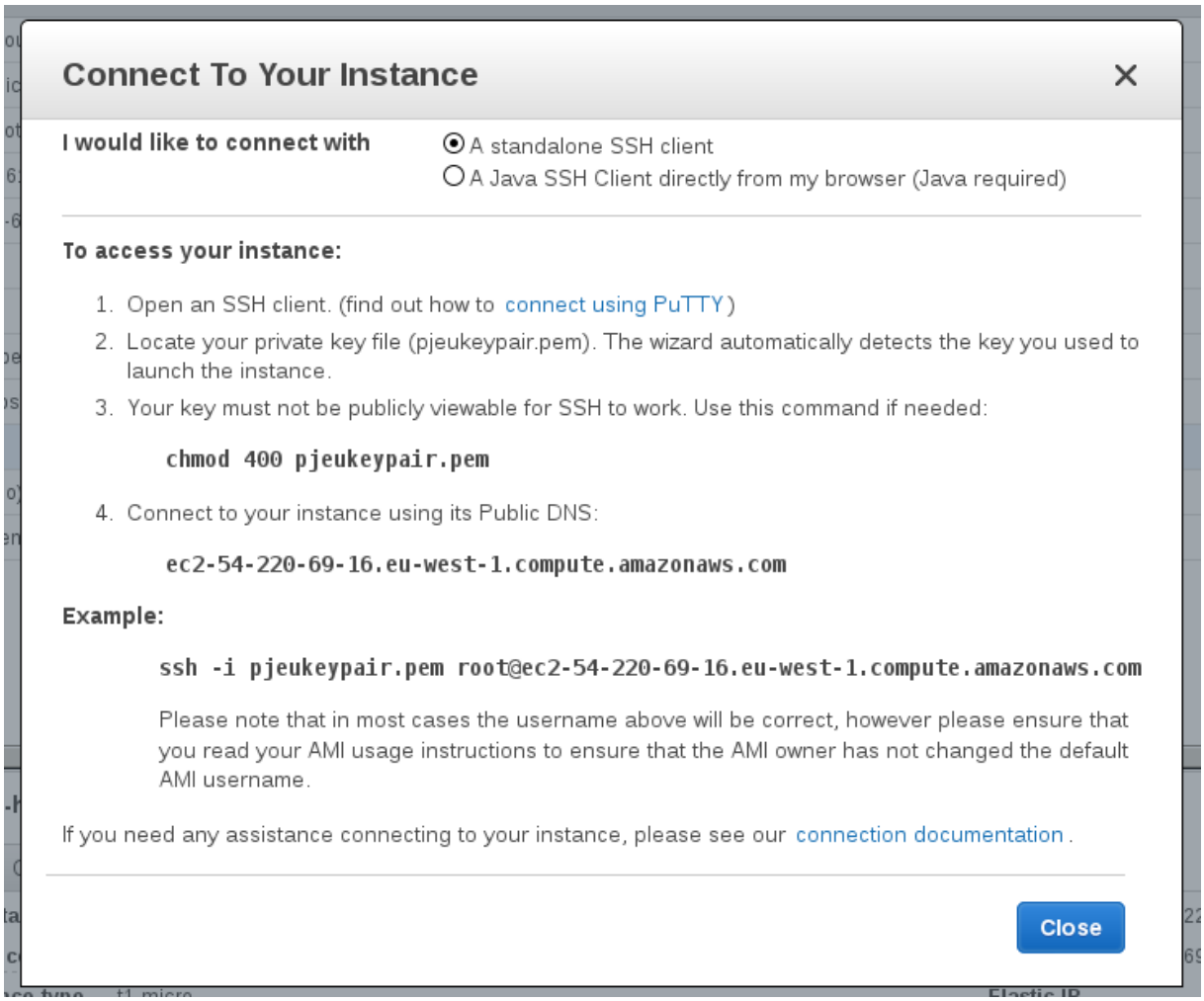

Figure 2.9: SSH Instructions To Connect To The Marked Instance

Viewing the head node console:

The head node takes about 2 minutes to start up. If, on following the instructions, an SSH connection cannot be made, then it can be worth checking the head node system log to check if the head node has started up correctly. The log is displayed on right-clicking on the "`Actions`" button, selecting the `Instance Settings` sub-menu, and selecting the "`Get System Log`" menu item (figure 2.10). A successful start of the system generates a log with a tail similar to that of figure 2.10.

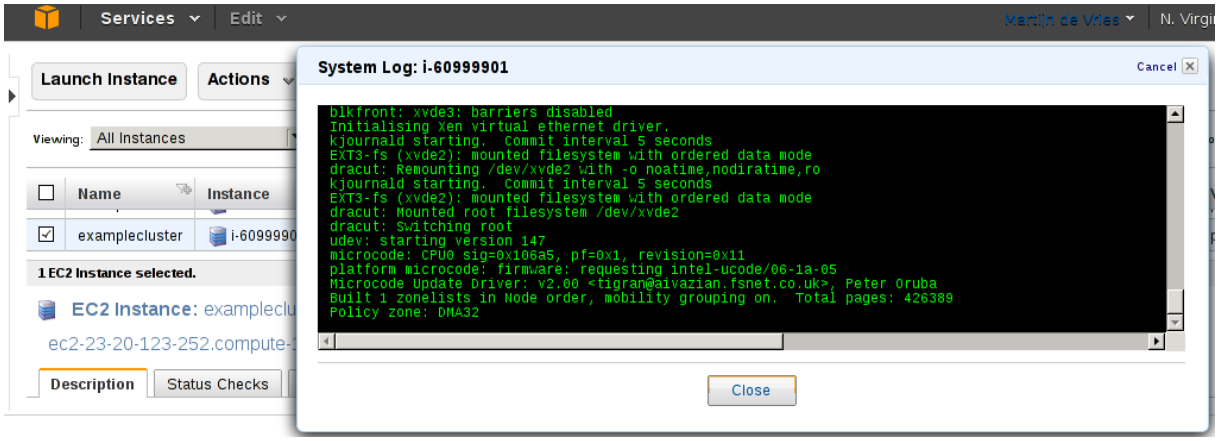

Figure 2.10: System Log Of The Checkboxed Instance

A screenshot of the instance is also possible by right-clicking on the selected instance, then following the clickpath `Instance Settings→Get Instance Screenshot`.

If the system and network are working as expected, then an SSH connection can be made to the head node to carry out the next step, which is the configuration of the head node and cluster.

2.2 Cluster On Demand With AWS: Head Node Login And Cluster Configuration

After the Amazon console manager has started up a head node instance, the head node instance and cluster must be configured. Logging into the head node via `ssh` allows this.

On logging in for the first time, the system suggests that the `cm-bright-setup` script be run:

Example

```
pj@office:~$ ssh -i pjkeypair.pem root@ec2-176-34-160-197.eu-west-1.com\
pute.amazonaws.com
The authenticity of host 'ec2-176-34-160-197.eu-west-1.compute.amazonaw\
s.com (176.34.160.197)' can't be established.
RSA key fingerprint is 66:1e:f3:77:83:f8:3f:42:c1:b7:d2:d5:56:d8:c3:58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-176-34-160-197.eu-west-1.compute.amazon\
aws.com,176.34.160.197' (RSA) to the list of known hosts.
Welcome to Bright Cluster Manager

                                          Based on Scientific Linux 5
                                          Cluster Manager ID: #999915

--------------------------------------------------------------------------

To set up your cluster, type

  cm-bright-setup

and follow the instructions

Creating DSA key for ssh
[root@headnode ~]#
```

Running the `cm-bright-setup` script goes through several screens, some of which prompt for input. At some prompts, it is hinted that typing "`I`" gives further explanation about the input.

The screens go through the following issues:

- The license agreement.
- Amazon Web Services account information. This asks for the AWS Username, AWS Account ID, Access Key ID, and Secret Access Key. These credentials are documented at `http://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html` and are needed by Bright Cluster Manager to manage the cloud node instances.
- The installation of the Bright Computing product key (formatted like 868868-797099-979199-091301-134414). This is the cloud version of the `request-license` command in section 4.3 of the *Installation Manual*, and asks for:
  - The organization information for the license. This requires input for the fields: country, state, locality, organizational unit, unit and cluster name.
  - The values to be used for the head node machine name and its administrative password (used for the root and MySQL passwords).
- Optionally, setting up the secondary drives using Amazon's EBS service.
- Optionally, setting up extra storage for `/home` using Amazon's EBS service.
- Optionally, setting up extra storage for monitoring data (recommended for more than 500 nodes).
- Setting up cloud node instance types. Amazon instance types (`http://aws.amazon.com/ec2/instance-types/`) are choices presented from node specifications consisting of memory, storage, cores, GPUs, and others. The setting up of the instance type step is looped through if necessary to allow more than one instance type to be configured.
  - Setting up
    - the number <*N*> of cloud compute nodes and
    - their base name (the `cnode` part of the name if the nodes have the names `cnode001` to `cnode`<*N*>).
  - Setting up the amount of storage for the cloud compute node instance.

    The default disk partitioning layout for nodes can be modified as described in Appendix D of the *Administrator Manual*. Using diskless nodes is also possible if the cloud compute node instance has enough RAM—about 2GB at the time of writing.

  Setting these values causes the cloud node objects to be created in the CMDaemon database. Cloud nodes are not however actually started up at this stage. Starting up must be done explicitly, and is covered in section 2.4.
- Setting up the workload manager, along with the number of slots, and if the head node is to be used for compute jobs too.

After accepting the input, the `cm-bright-setup` script runs through the configuration of the head node and the cloud nodes. Its progress is indicated by output similar to the following:

Example

```
[root@headnode ~]# cm-bright-setup
Retrieving Amazon instance information

------------------------------------------------------------------------
```

```
|                            License agreements                          |
------------------------------------------------------------------------
The end user license agreement will be shown in a new screen. To exit
this screen, type 'q'. Press any key to continue
Do you agree with the license terms and conditions? (yes, no, show): yes

------------------------------------------------------------------------
|                        Amazon account information                      |
------------------------------------------------------------------------
AWS username (I for information): exampleuser@brightcomputing.com
AWS Account ID (I for information): 313234312222
Access Key ID (I for information): OUPOUASOUDSSSAOU
Secret Access Key (I for information): Aighei8EooLi1Dae8Nio5ohl4ieXiAiaiV

Verifying Amazon credentials

------------------------------------------------------------------------
|                               Bright License             |
------------------------------------------------------------------------
Bright Product Key (I for information):
                                      423112-231432-134234-132423-134221
   Country: US
   State: CA
   Locality: San Francisco
   Organization name: Bright Computing
   Organizational Unit: Development
   Cluster Name: demo-cluster

------------------------------------------------------------------------
|                                 Head Node                            |
------------------------------------------------------------------------
Hostname: bright80
Administrative Password:
Verify password:
Do you want to create a second drive for more storage capacity?
(I for information) [YES|no] no
Extra storage for /home (I for information)? [NO|yes] n
Extra storage for monitoring data (I for information)? [NO|yes] n

------------------------------------------------------------------------
|                               Compute Nodes                          |
------------------------------------------------------------------------
Instance Type (I for information):
  m1.small
  m1.medium
  c1.medium
  m1.large
  t1.micro
  m2.xlarge
  m2.2xlarge
  m2.4xlarge
  c1.xlarge
  [t1.micro]
>
t1.micro
```

```
Node Count [2]:
2
Base name (I for information) [cnode]:
cnode
Instances of type t1.micro need to use EBS storage
Size of EBS (GB) [40]: 15
Do you want to configure more node types? [NO|yes]
no

------------------------------------------------------------------------
|                         Workload Management system                   |
------------------------------------------------------------------------
Which workload management system do you want to use? (I for information)?
  slurm
  sge
  torque
  [slurm]
>
slurm
Number of slots per node [8]:
8
Do you want to use the head node for compute jobs? [NO|yes]
no

The following information will be used to configure this head node:

  Amazon information:
    AWS username:             exampleuser@brightcomputing.com
    AWS Account ID:           313234312222
    Access Key ID:            OUPOUASOUDSSSAOU
    Secret Access Key:        Aighei8EooLi1Dae8Nio5ohl4ieXiAin5eeRoaiV
  Bright Product Key:         423112-231432-134234-132423-134221
  License information
    Country:                  US
    State:                    CA
    Locality:                 San Francisco
    Organization name:        Bright Computing
    Organizational Unit:      Development
    Cluster Name:             demo-cluster
  Hostname:                   bright80
  Second drive size:          no
  Instance Type:              t1.micro
  Node Count:                 2
  Base name:                  cnode
  Storage type:               EBS
  Size of storage:            15 GB
  Workload management system: slurm
  Number of slots:            8
  Head node for compute jobs: no

The information to configure this head node has been collected, and shown
above. The next phase will be to process this information. A new Bright
license will be installed, the Bright Cluster Manager software will be
initialized and the workload management software will be initialized
Do you want to continue? [YES|no]
```

```
yes

Starting to configure this head node
Successfully retrieved the license
Installed license
Initializing Bright Cluster Manager
Installing admin certificates
Configuring default scheduler, slurm
Set up finished
```

It is recommended that the system administrator log out and login again after the script has been run, in order to enable the new environment for the shell that the administrator is in. If the hostname was changed in the `cm-bright-setup` script, for example, the name change shows up in the shell prompt only after the re-login.

Once there is a head in the cloud, the other cloud nodes can be started up.

2.3 Cluster On Demand With AWS: Connecting To The Headnode Via `cmsh` Or Bright View

Amazon provides a security group to each instance. By default, this configures network access so that only inbound SSH connections are allowed from outside the cloud. A new security group can be configured, or an existing one modified, using the `Edit details` button in figure 2.11. Security groups can also be accessed from the navigation menu on the left side of the EC2 Management Console.

2.3.1 Cluster On Demand With AWS: Access With Bright View

The security group defined by Amazon for the head node can be modified by the administrator to allow remote connections to CMDaemon running on the head node (figure 2.11).

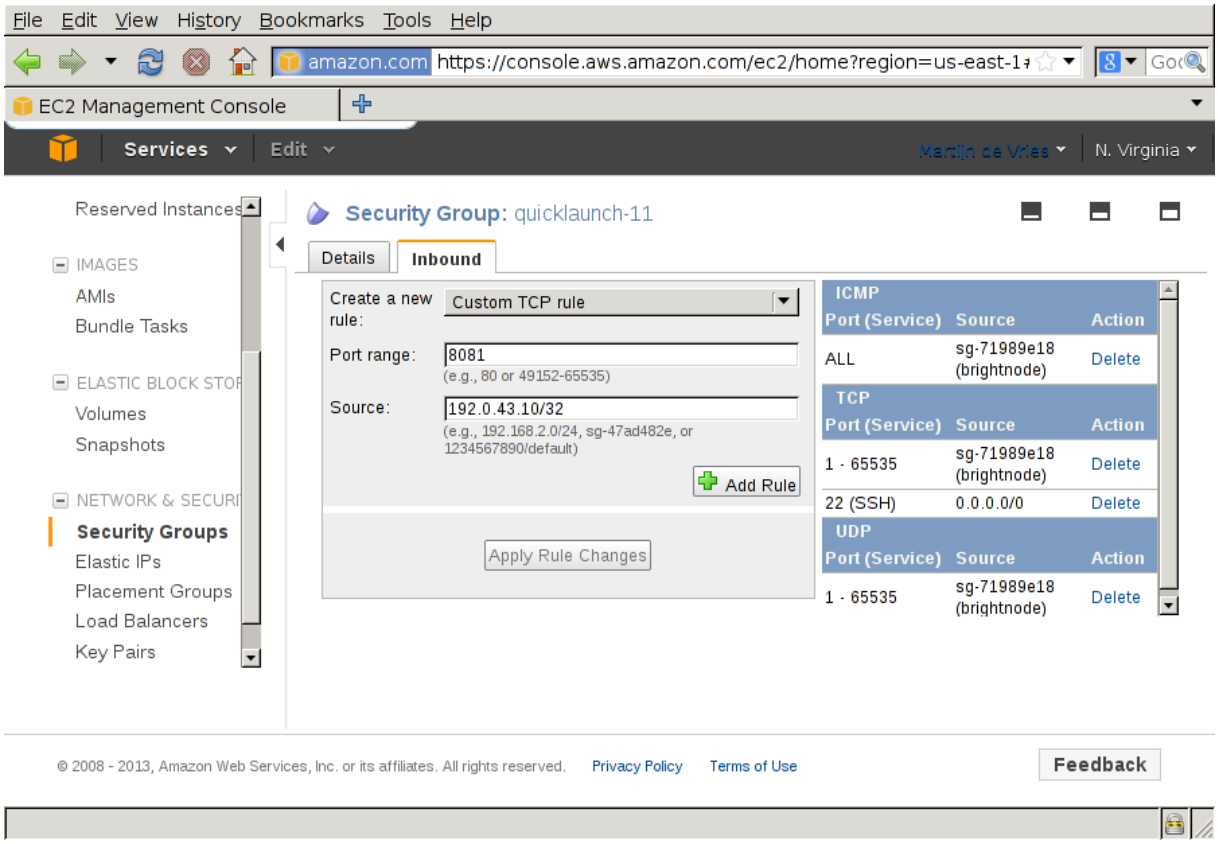

Figure 2.11: Security Group Network And Port Access Restriction

- To allow only a specific network block to access the instance, the network from which remote connections are allowed can be specified in CIDR format.
- Explicitly allowing inbound connections to port 8081 on the head node allows Bright View (section 2.4 of the *Administrator Manual*) to connect to the head node. This is because the Bright View back end, which is CMDaemon, communicates via port 8081.

2.3.2 Cluster On Demand With AWS: Access With A Local `cmsh`

The security group created by Amazon by default already allows inbound SSH connections from outside the cloud to the instance running in the cloud, even if the incoming port 8081 is blocked. Launching a `cmsh` session within an SSH connection running to the head node is therefore possible, and works well.

2.3.3 Cluster On Demand With AWS: Access With A Local-To-The-Cluster Bright View

It is possible to run an X-forwarded Bright View local-to-the-cluster session from within an `ssh -X` connection that is already running to the head node. However, it suffers from significant X-protocol lag due to the various network encapsulation layers involved. Using a local-to-the-user browser for running Bright View to access the cluster, i.e. from outside the cloud, is therefore recommended instead, for a more pleasant experience.

2.4 Cluster On Demand With AWS: Cloud Node Start-up

Cloud nodes must be explicitly started up. This is done by powering them up, assuming the associated cloud node objects exist. The cloud node objects are typically specified in the `cm-bright-setup` script—in the preceding example the cloud node objects are `cnode001` and `cnode002`.

However, more cloud node objects can be created if needed after the `cm-bright-setup` script has run. The maximum number that may be created is set by the license purchased.

Large numbers of cloud node objects can be created with Bright Cluster Manager as follows:

- In `cmsh` a large number of cloud node objects can conveniently be created with the "`foreach --clone`" command instead, as described in section 4.3.

After creation, an individual cloud node, *<cloud node>*, can be powered up from within Bright View via the clickpath:

`Cloud`→`Cloud Nodes`→*<cloud node>*↓`Power`→`On`

As with regular non-cloud nodes, cloud nodes can also be powered up from within the `device` mode of `cmsh`. The initial power status (section 4.1 of the *Administrator Manual*) of cloud nodes is `FAILED`, because they cannot be communicated with. As they start up, their power status changes to `OFF`, and then to `ON`. Some time after that they are connected to the cluster and ready for use. The device status (as opposed to the power status) remains `DOWN` until it is ready for use, at which point it switches to `UP`:

Example

```
[head1->device]% power status
cloud ................. [  FAILED ] cnode001 (Cloud instance ID not set)
cloud ................. [  FAILED ] cnode002 (Cloud instance ID not set)
No power control ...... [ UNKNOWN ] head1
[head1->device]% power on -n cnode001
cloud ................. [   ON    ] cnode001
[head1->device]% power status
cloud ................. [   OFF   ] cnode001 (pending)
cloud ................. [  FAILED ] cnode002 (Cloud instance ID not set)
No power control ...... [ UNKNOWN ] head1
[head1->device]% power on -n cnode002
cloud ................. [   ON    ] cnode002
[head1->device]% power status
cloud ................. [   ON    ] cnode001 (running)
cloud ................. [   OFF   ] cnode002 (pending)
No power control ...... [ UNKNOWN ] head1
[head1->device]% !ping -c1 cnode001
ping: unknown host cnode001
[head1->device]% status
head1 ..................... [   UP   ]
node001 ................... [   UP   ]
node002 ................... [  DOWN  ]
[head1->device]% !ping -c1 cnode001
PING cnode001.cm.cluster (10.234.226.155) 56(84) bytes of data.
64 bytes from cnode001.cm.cluster (10.234.226.155): icmp_seq=1 ttl=63 t\
ime=3.94 ms
```

Multiple cloud nodes can be powered up at a time in `cmsh` with the "`power on`" command using ranges and other options (section 4.2.2 of the *Administrator Manual*).

2.4.1 IP Addresses In The Cluster On Demand Cloud

- The IP addresses assigned to cloud nodes on powering them up are arbitrarily scattered over the 10.0.0.0/8 network
  - No pattern should therefore be relied upon in the addressing scheme of cloud nodes
- Shutting down and starting up head and regular cloud nodes can cause their IP address to change.

– However, Bright Cluster Manager managing the nodes means that a regular cloud node re-establishes its connection to the cluster when it comes up, and will have the same node name as before.

3

Cluster Extension Cloudbursting

Cluster Extension cloudbursting ("hybrid" cloudbursting) in Bright Cluster Manager is the case when a cloud service provider is used to provide nodes that are in the cloud as an extension to the number of regular nodes in a cluster. Thus, the head node in a Cluster Extension configuration is always outside the cloud, and there may be some non-cloud-extension regular nodes that are outside the cloud too.

Requirements

Cluster Extension cloudbursting requires:

- **An activated cluster license.**

  One does not simply cloudburst right away in a Cluster Extension configuration. The license must first be made active, or the attempt will fail.

  A check on the state of the license can be carried out with:

  Example

```
[root@bright80 ~]# cmsh -c "main; licenseinfo"
License Information
---------------------------- ----------------------------------------
Licensee                     /C=US/ST=NY/L=WS/O=Bright
                             Mordor/OU=Mt. Doom/CN=Bright 8.0 Cluster
Serial Number                54750
Start Time                   Mon Dec 20 00:00:00 2015
End Time                     Wed Jun 20 00:00:00 2018
Version                      7.0 and above
Edition                      Advanced
Pre-paid Nodes               100
Max Pay-per-use Nodes        1000
Max Big Data Nodes           80
Max OpenStack                70
Node Count                   4
Big Data Node Count          0
OpenStack Node Count         0
MAC Address / Cloud ID       FA:16:3F:01:52:55
```

  If activation is indeed needed, then simply running the `request-license` command with the product key should in most cases provide activation. Further details on activating the license are given in Chapter 4 of the *Installation Manual*.

- **Registration of the product key.**

The product key must also be registered on the Bright Computing Customer Portal website at `http://customer.brightcomputing.com/Customer-Login`. A Customer Portal account is needed to do this.

The product key is submitted at the Customer Portal website specifically for a Cluster Extension setup, from the `Burst!` menu. The customer portal account is then automatically associated with the license on the head node.

- **An Amazon account**, if the cloud provider is Amazon.
- **An Azure account**, if the cloud provider is Microsoft Azure.
- **An open UDP port.**

  By default, this is port 1194. It is used for the OpenVPN connection from the head node to the cloud and back. To use TCP, and/or ports other than 1194, the Bright Computing knowledgebase at `http://kb.brightcomputing.com` can be consulted using the keywords "openvpn port".

  Outbound SSH access from the head node is also useful, but not strictly required.

  By default, the Shorewall firewall as provided by Bright Cluster Manager on the head node is configured to allow all outbound connections, but other firewalls may need to be considered too.

- **A special proxy environment configuration setting**, if an HTTP proxy is used to access the AWS or Azure APIs.

  The proxy environment configuration is carried out using the `ScriptEnvironment` directive (page 602 of the *Administrator Manual*), which is a CMDaemon directive that can be set and activated (page 589 of the *Administrator Manual*).

  For example, if the proxy host is `my.proxy` and accepting connections on port `8080` with a username `my` and password `pass`, then the setting can be specified as:

```
ScriptEnvironment = { "http_proxy=http://my:pass@my.proxy:8080", \
"https_proxy=http://my:pass@my.proxy:8080", "ftp_proxy=http://my:pass@my.proxy:8080" }
```

Steps

Cluster Extension cloudbursting uses a *cloud director*. A cloud director is a specially connected cloud node used to manage regular cloud nodes, and is described more thoroughly in section 3.2. Assuming the administrator has ownership of a cloud provider account, the following steps can be followed to launch Cluster Extension cloud nodes:

1. The cloud provider is logged into from Bright View, and a cloud director is configured (section 3.1).
2. The cloud director is started up (section 3.2).
3. The cloud nodes are provisioned from the cloud director (section 3.3).

The cloud nodes then become available for general use by the cluster.

Cluster Extension Cloudbursting With A Hardware VPN

Bright Cluster Manager recommends, and provides, OpenVPN by default for Cluster Extension cloudbursting VPN connectivity. If there is a wish to use a hardware VPN, for example if there is an existing hardware VPN network already in use at the deployment site, then Bright Cluster Manager can optionally be configured to work with the hardware VPN. The configuration details can be found in the Bright Computing knowledgebase at `http://kb.brightcomputing.com` by carrying out a search on the site using the keywords "cloudbursting without openvpn".

3.1 Cluster Extension With AWS: Cloud Provider Login And Cloud Director Configuration

Cluster Extension with AWS is described in sections 3.1, 3.2, and 3.3.

Cluster Extension with Azure is described in Chapter 5.

To access the Amazon cloud service from Bright View, via the URL `https://<head node address>:8081/bright-view/`, the clickpath `Cloud→AWS→AWS Wizard` is selected. This brings up a screen introducing the `AWS Wizard`.

The wizard goes through the following stages:

1. Introduction (section 3.1.1)
2. AWS Credentials (section 3.1.2)
3. Select Regions (section 3.1.3)
4. Summary & Deployment (section 3.1.5)
5. Deploy (section 3.1.6)

3.1.1 Introduction

The first screen displayed by the wizard is the introduction screen (figure 3.1), which reminds the administrator about the prerequisites for cloudbursting.

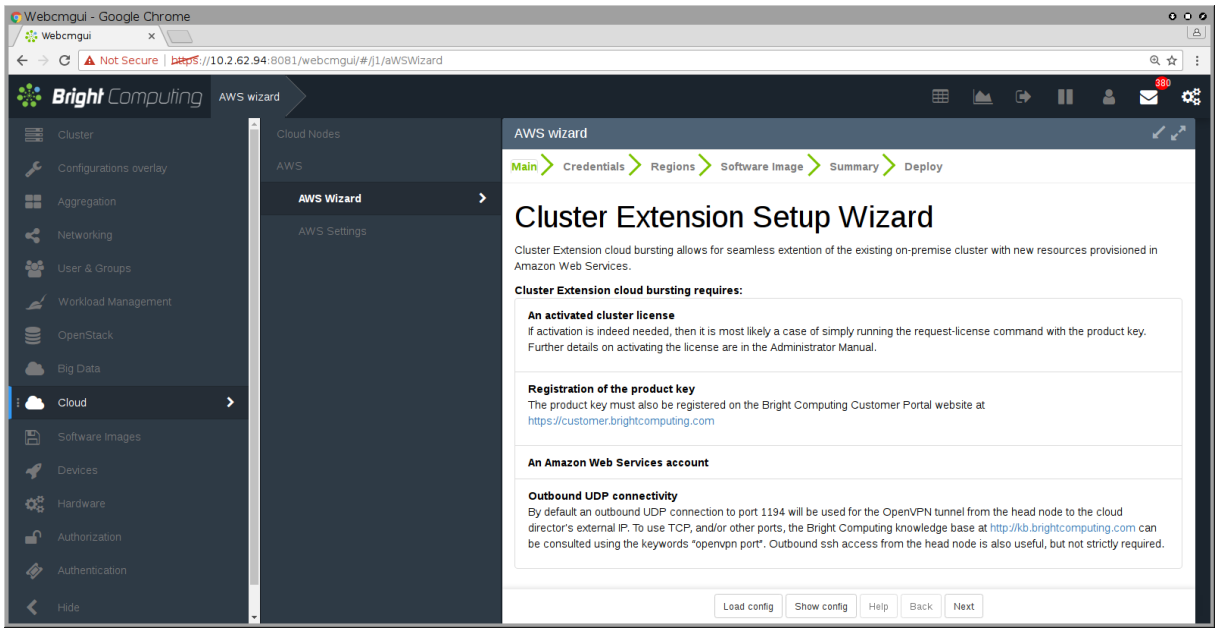

Figure 3.1: Introduction Page For Cluster Extension With Bright View

The `Show`, `Save`, and `Load` buttons allow the wizard to show, save, or load a YAML configuration text file. Saving a configuration at the start or end of a wizard run is usually convenient for an administrator.

3.1.2 AWS Credentials

The `Next` button brings up the credentials page, if the credentials for the cluster extension cloudburst are not yet known to CMDaemon (figure 3.2).

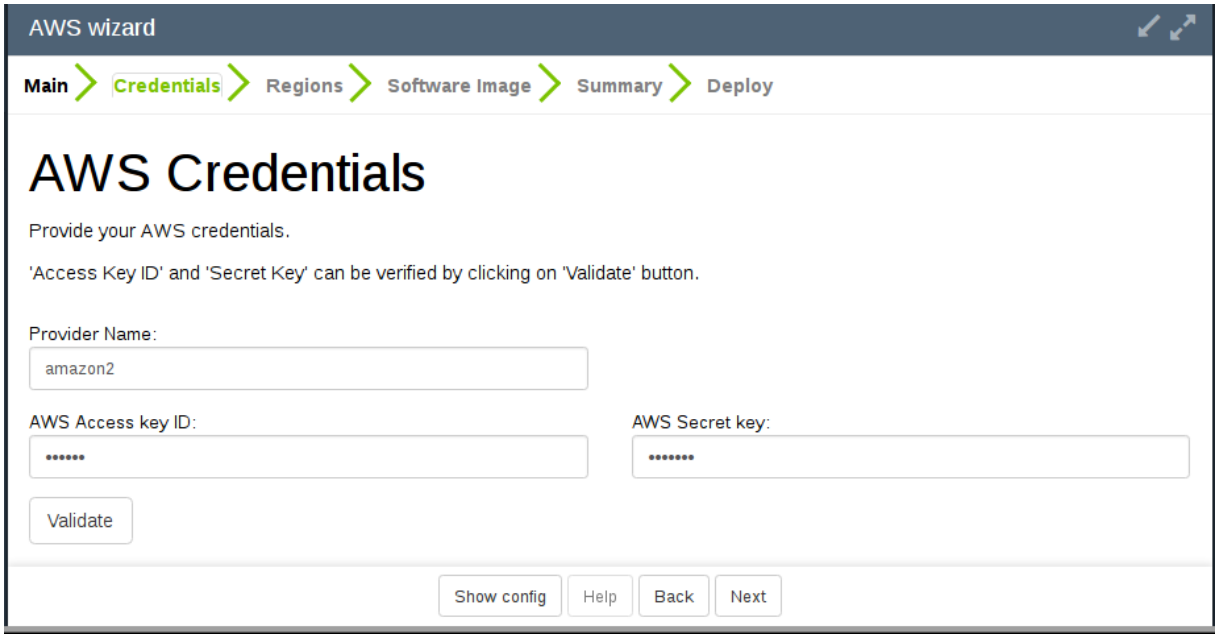

Figure 3.2: AWS Key Configuration For Cluster Extension With Bright View

This asks for:

- `Provider Name`: This is can be any user-defined value. If using Amazon, it would be sensible to just put in `amazon`
- `AWS Access key ID`: The AWS Access key. Typically a string that is a mixture of upper case letters and numbers.
- `AWS Secret key`: The AWS secret key. Typically a longer string made up of alphanumeric characters.

In the case of Amazon, the information is obtainable after signing up for Amazon Web Services (AWS) at `https://console.aws.amazon.com/iam/home?#security_credential`.

The `Validate` button validates the credentials at this point of the configuration when clicked, instead of waiting until the actual deployment. Clicking the `Next` button submits the details of this screen, and inputs for the next screen are then retrieved from Amazon.

3.1.3 Select Regions

If all goes well, the next screen (figure 3.3) displays region selection options for the Amazon cloud service.

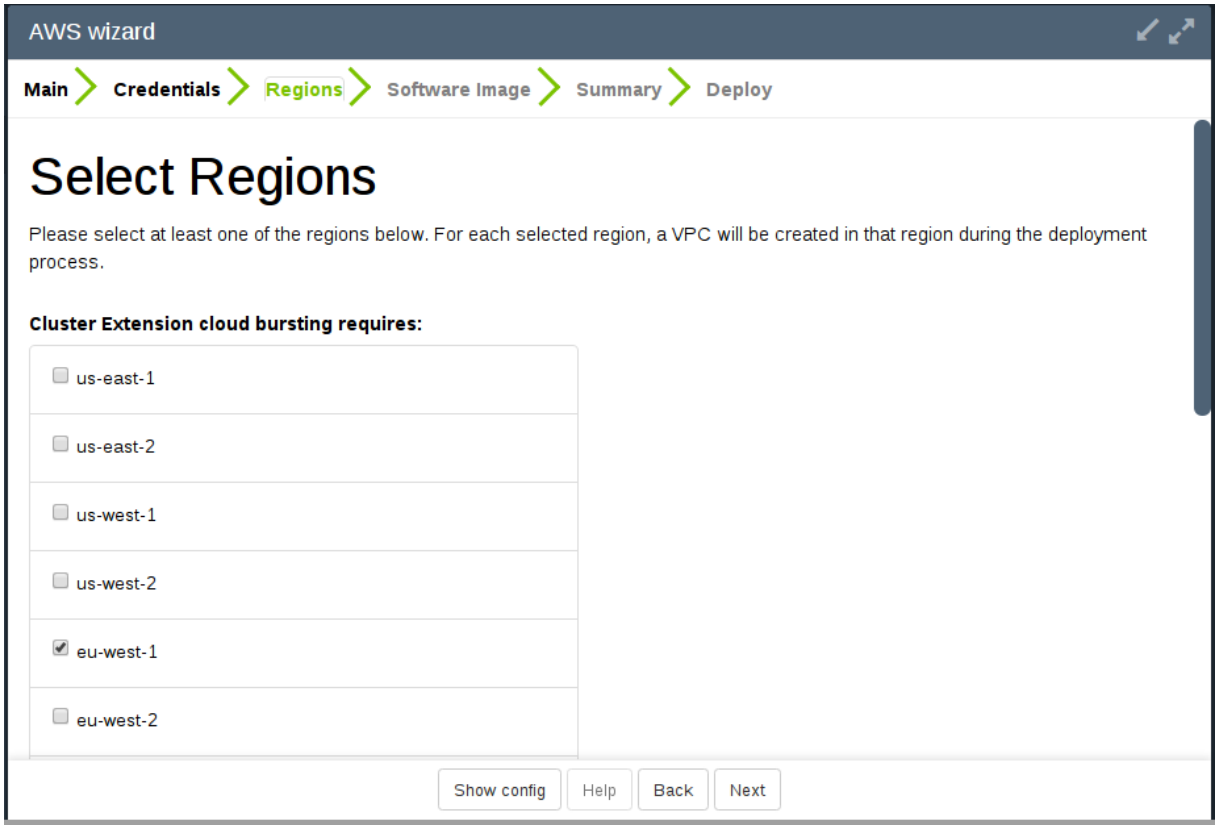

Figure 3.3: Selecting Regions For The Cluster Extension Wizard With Bright View

Regions are designated by codes, for example `us-east-1`, `eu-west-1`, and so on. The following are implied by the first two letters associated with the region:

- `us`: United States
- `eu`: Europe
- `ap`: Asia Pacific
- `sa`: South America
- `ca`: Canada

In addition, the special `us-gov` prefix designates a region that helps customers comply with US government regulations.

Similarly, European data regulations may, for example, mean that data should be processed only within an `eu` region.

Other than legislative considerations, choosing capacity from a region that is geographically closer is often sensible for avoiding lag with certain applications. On the other hand, using off-peak capacity from a geographically distant location may make more sense if it is cheaper.

Further details on regions can be found at `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html`.

After the administrator has selected the regions that are to be used in deployment, the `Next` button can be clicked, to bring up the next screen.

3.1.4 Select Software Images

The `Select Software Images` screen (figure 3.4) lets the administrator select possible cloud node software images to be placed on the cloud director node. These can then be provisioned from the director to the cloud nodes. A default cloud node image is selected from one of the possible cloud node software images.

The cloud director virtual hardware type and cloud node virtual hardware type can also be selected in this screen.

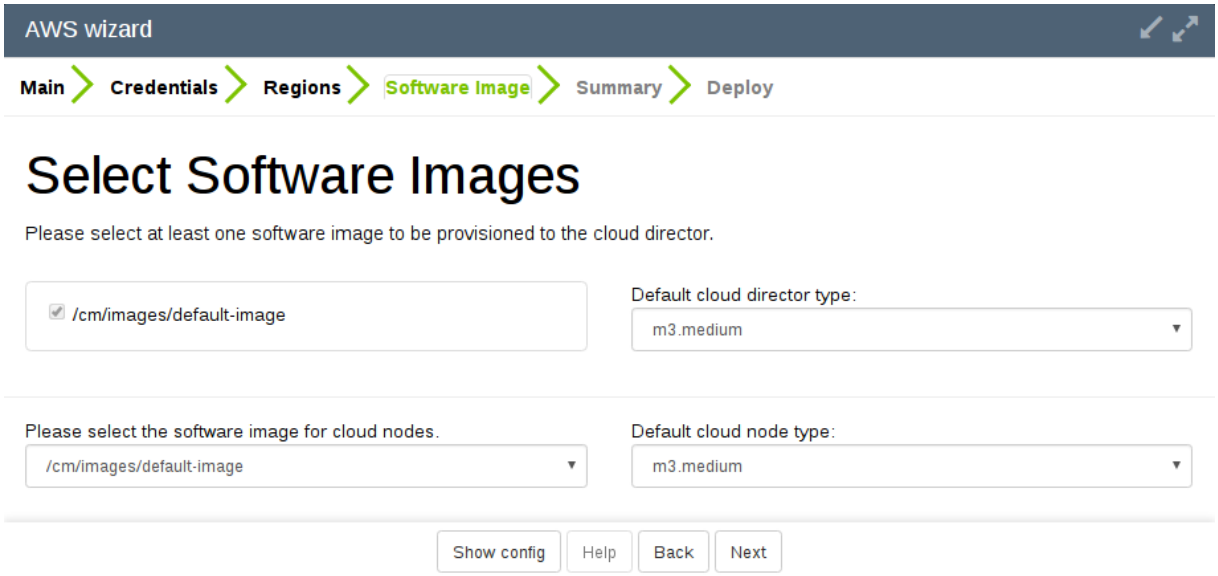

Figure 3.4: Selecting Software Images For The Cluster Extension Wizard With Bright View

3.1.5 Summary & Deployment

The next screen is a `Summary` screen (figure 3.5).

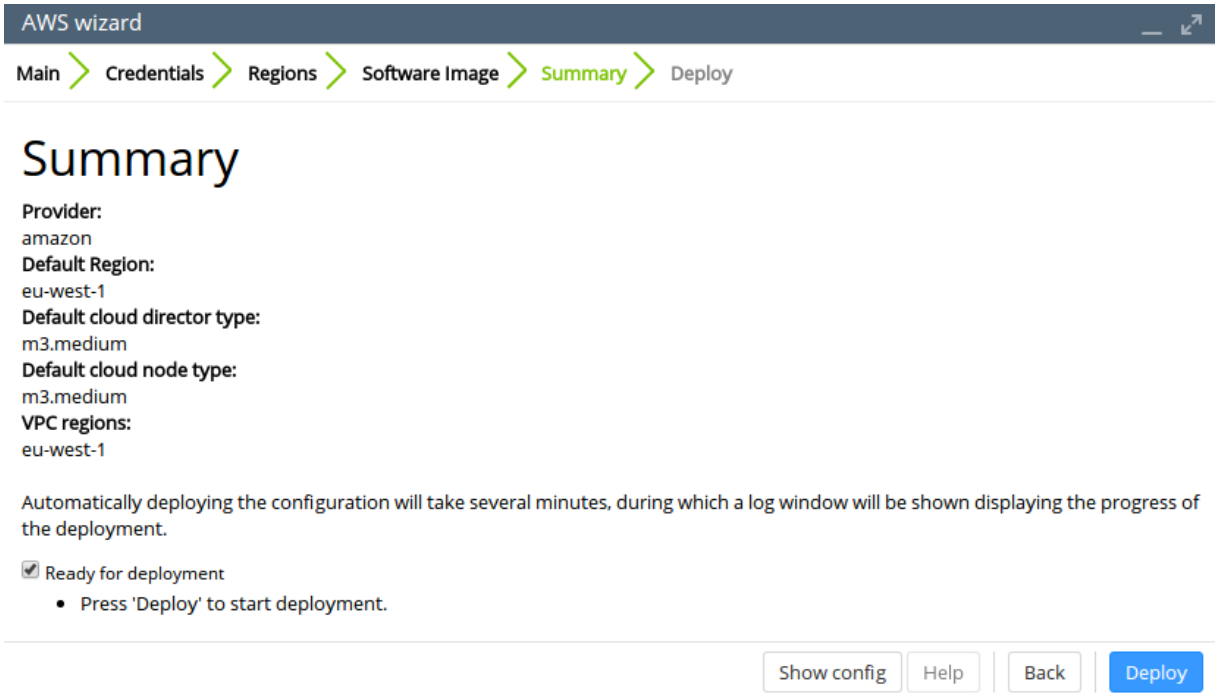

Figure 3.5: Summary Screen For The Cluster Extension Wizard With Bright View

It summarizes the selections that have been made, and lets the administrator review them. The selections can be changed, if needed, by going back to the previous screens, by directly clicking on the values. Otherwise, on clicking the `Deploy` button, the configuration is deployed.

3.1.6 Deploy

During configuration deployment, a progress meter indicates what is happening as the configuration is processed. At the end of processing, the display indicates with the `Deployed with success` message that the cluster has been extended successfully (figure 3.6).

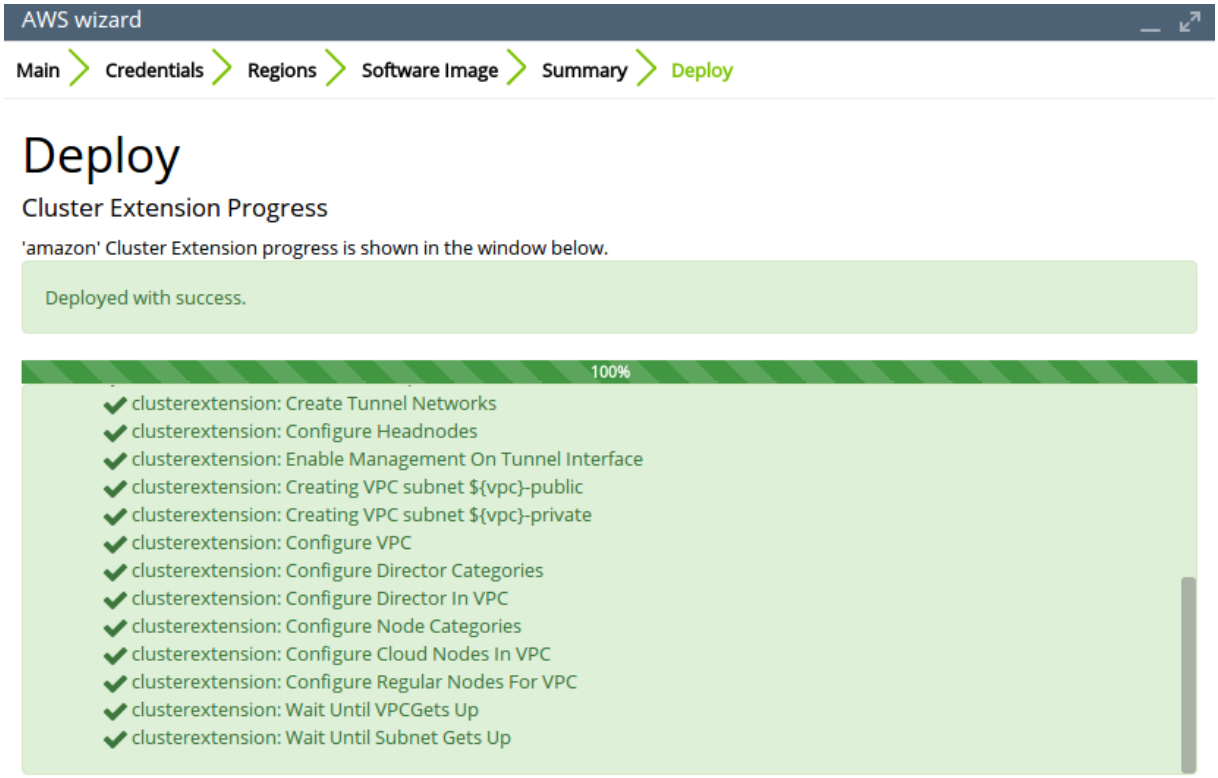

Figure 3.6: Cluster Extension Configuration Processing With Bright View

No nodes are activated yet within the cloud provider service. To start them up, the components of the cloud provider service must be started up by

- powering up the cloud directors (section 3.2)
- powering on the cloud nodes after the cloud directors are up. Often this involves creating new cloud nodes (section 3.3).

3.2 Cluster Extension With AWS: Cloud Director Startup From Scratch

The cloud director can take some time to start up the first time when it is *installing from scratch*. The bottleneck is usually due to several provisioning stages, where the bandwidth between the head node and the cloud director means that the provisioning runs typically take tens of minutes to complete. The progress of the cloud director can be followed in the Bright View event log viewer (section 12.2.11 of the *Administrator Manual*), or they can be followed in an open `cmsh` session, as the events are sent to the `cmsh` session.

The bottleneck is one of the reasons why the cloud director is put in the cloud in the first place: nodes are provisioned from a cloud director in the cloud faster than from a head node outside the cloud.

The bottleneck of provisioning from the head node to the cloud director is an issue only the first time around. The next time the cloud director powers up, and assuming persistent storage is used—as is the default—the cloud director runs through the provisioning stages much faster, and completes within a few minutes.

The reason why powering up after the first time is faster is because the image that is to power up is already in the cloud. A similar principle—of relying on data already available with the cloud provider—can be used as a technique to make first time startup even faster. The technique is to have a pre-built

image—a snapshot—of the cloud director stored already with the cloud provider. The first-time startup of a cloud director based on a snapshot restoration is discussed in section 3.4.

The remainder of this section is about starting up a cloud director from scratch—that is, a first time start, and without a pre-built image.

To recap: by default, a cloud director object is created during a run of the Cluster Extension wizard (section 3.1).

There can be only one cloud director per region. Because a cloud director also has properties specific to the region within which it directs nodes, it means that cloud directors can only be created from scratch, via cluster extension.

Once a cloud director object has been made in CMDaemon, then the cloud director is ready to be started up. In Bright View, the cloud director can be started by powering it up from its node settings, just like a regular node. For the cloud director a clickpath to power it up is:

```
Cloud→Cloud Nodes→Cloud director→↓Power→On
```

As indicated earlier on, the cloud director acts as a helper instance in the cloud. It provides some of the functions of the head node within the cloud, in order to speed up communications and ensure greater resource efficiency. Amongst the functions the cloud director provides are:

- Cloud nodes provisioning
- Exporting a copy of the shared directory `/cm/shared` to the cloud nodes so that they can mount it
- Providing routing services using an OpenVPN server. While cloud nodes within a region communicate directly with each other, cloud nodes in one region use the OpenVPN server of their cloud director to communicate with the other cloud regions and to communicate with the head node of the cluster.

Cloud directors are not regular nodes, so they have their own category, `cloud-director`, into which they are placed by default.

The cloud-related properties of the cloud director can be viewed and edited via the clickpath:

```
Cloud→Cloud Nodes→Cloud director→Edit
```

3.2.1 Setting The Cloud Director Disk Storage Device Type

Amazon provides two kinds of storage types as part of EC2:

1. **Instance storage**, using so-called ephemeral devices. Ephemeral means that the device is temporary, and means that whatever is placed on it is lost if the instance is stopped, terminated, or if the underlying disk drive fails.

   Some instances have ephemeral storage associated with the instance type. For example, at the time of writing (May 2017), the `m3.medium` type of instance has 4GB of SSD storage associated with it.

   Details on instance storage can be found at `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html#instance-store-volumes`.

2. **Elastic Block Storage (EBS) volumes:** EBS is a persistent, reliable, and highly available storage. Normally, EBS is suggested for cloud director and cloud node use. The reasons for this include:

   - it can be provided to all nodes in the same availability zone
   - unlike instance storage, EBS remains available for use when an instance using it is stopped or terminated.

- instance storage is not available for many instance types such as `t2.micro`, `t2.small`, `c4.large`.

Using The Ephemeral Device As The Drive For The Cloud Director:
Since the cloud director instance type is essential, and contains so much data, it is rare to use an ephemeral device for its storage.

However, if for some reason the administrator would like to avoid using EBS, and use the instance storage, then this can be done by removing the default EBS volume suggestion for the cloud director provided by Bright Cluster Manager. When doing this, the ephemeral device that is used as the replacement must be renamed. It must take over the name that the EBS volume device had before it was removed.

- In Bright View, this can be done in the `EC2 Storages` window, which for a cloud director <*cloud director hostname*> can be viewed and modified via the clickpath:

  `Cloud`→`Cloud Nodes`→<*cloud director hostname*>→`Edit`→`Cloud settings`→`Storage`→`Edit`

- In `cmsh`, this can be done in `device` mode, by going into the `cloudsettings` submode for the cloud director, and then going a level deeper into the `storage` submode. Within the `storage` submode, the `list` command shows the values of the storage devices associated with the cloud director. The values can be modified as required with the usual object commands. The `set` command can be used to modify the values.

  Example

```
[bright80]% device use us-east-1-director
[bright80->device[us-east-1-director]]% cloudsettings
[bright80->device[us-east-1-director]->cloudsettings]% storage
[bright80->...->cloudsettings->storage]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- ------- ----------
ebs        ebs          sdb       42GB
ephemeral  ephemeral0   sdc       0B      ephemeral0
[bright80->...->cloudsettings->storage]% remove ebs
[bright80->...->cloudsettings*->storage*]% set ephemeral0 drive sdb
[bright80->...->cloudsettings*->storage*]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- ------- ---------
ephemeral  ephemeral0   sdb       0B      ephemeral0
[bright80->...->cloudsettings*->storage*]% commit
```

3.2.2 Setting The Cloud Director Disk Size

The disk size for the cloud director can be set in Bright View using the `EC2 Storages` window (section 3.2.1).

By default, an EBS volume size of 42GB is suggested. This is as for a standard node layout (section D.3 of the *Administrator Manual*), and no use is then made of the ephemeral device.

42GB on its own is unlikely to be enough for most purposes other than running basic `hello world` tests. In actual use, the most important considerations are likely to be that the cloud director should have enough space for:

- the user home directories (under `/home/`)
- the cluster manager shared directory contents, (under `/cm/shared/`)

- the software image directories (under `/cm/images/`)

The cluster administrator should therefore properly consider the allocation of space, and decide if the disk layout should be modified. An example of how to access the disk setup XML file to modify the disk layout is given in section 3.9.3 of the *Administrator Manual*.

For the cloud director, an additional sensible option may be to place `/tmp` and the swap space on an ephemeral device, by appropriately modifying the XML layout for the cloud director.

3.2.3 Tracking Cloud Director Startup

Tracking Cloud Director Startup From The EC2 Management Console

The boot progress of the cloud director <*cloud director*> can be followed by watching the status of the instance in the Amazon EC2 management console, as illustrated in figure 2.8. The `Instance ID` that is used to identify the instance can be found

- with Bright View, within the clickpath
  `Cloud`→`Cloud Nodes`→<*cloud director*>→`Edit`→`Cloud settings`→`Instance ID`

- with `cmsh`, by running something like:

  Example

```
[bright80]% device use us-east-1-director
[bright80->device[us-east-1-director]]% get cloudid
i-f98e7441
[bright80->device[us-east-1-director]]% cloudsettings
[bright80->device[us-east-1-director]-cloudsettings]% get instanceid
i-f98e7441
```

Tracking Cloud Director Startup From Bright View

The boot progress of the cloud director can also be followed by

- watching the icon changes for the cloud node states in the clickpath `Cloud`→`Cloud Nodes`. The icons indicating the state follow the description given in section 5.5.1 of the *Administrator Manual*

Tracking Cloud Director Startup From The Bash Shell Of The Head Node

There are some further possibilities to view the progress of the cloud director after it has reached at least the initrd stage. These possibilities include:

- an SSH connection to the cloud director can be made during the pre-init, initrd stage, after the cloud director system has been set up via an rsync. This allows a login to the node-installer shell.
- an SSH connection to the cloud director can be also be made after the initrd stage has ended, after the init process runs making an SSH daemon available again. This allows a login on the cloud director when it is fully up.

During the initrd stage, the cloud director is provisioned first. The cloud node image(s) and shared directory are then provisioned on the cloud director, still within the initrd stage. To see what rsync is supplying to the cloud director, the command "`ps uww -C rsync`" can be run on the head node. Its output can then be parsed to make obvious the source and target directories currently being transferred:

Example

```
[root@bright80 ~]# ps uww -C rsync | grep -o ' /cm/.*$'
 /cm/shared/ syncer@172.21.255.251::target//cm/shared/
```

Tracking Cloud Director Startup From `cmsh`

The `provisioningstatus` command in `cmsh` can be used to view the provisioning status (some output elided):

Example

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus"
...
+ us-east-1-director
...
  Up to date images:        none
  Out of date images:       default-image
```

In the preceding output, the absence of an entry for "`Up to date images`" shows that the cloud director does not yet have an image that it can provision to the cloud nodes. After some time, the last few lines of output should change to something like:

Example

```
+ us-east-1-director
...
  Up to date images:        default-image
```

This indicates the image for the cloud nodes is now ready.

With the `-a` option, the `provisioningstatus -a` command gives details that may be helpful. For example, while the cloud director is having the default software image placed on it for provisioning purposes, the source and destination paths are `/cm/images/default-image`:

Example

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        4
Source node:          bright80
Source path:          /cm/images/default-image
Destination node:     us-east-1-director
Destination path:     /cm/images/default-image
...
```

After some time, when the shared filesystem is being provisioned, the source and destination paths should change to the `/cm/shared` directory:

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        5
Source node:          bright80
Source path:          /cm/shared
Destination node:     us-east-1-director
Destination path:     /cm/shared
...
```

After the shared directory and the cloud node software images are provisioned, the cloud director is fully up. Cloud node instances can then be powered up and provisioned from the cloud director.

3.3 Cluster Extension With AWS: Cloud Node Startup From Scratch

This section discusses the configuration of regular cloud node startup from scratch. Configuration of cloud node startup from snapshot is discussed in section 3.4. Regular cloud nodes are the cloud nodes that the cloud director starts up.

To configure the regular cloud nodes does not require a working cloud director. However to boot up the regular cloud nodes does require that the cloud director be up, and that the associated networks to the regular cloud nodes and to the head node be configured correctly.

If needed, additional cloud provisioning nodes (section 5.2 of the *Administrator Manual*) can be configured by assigning the provisioning role to cloud nodes, along with appropriate nodegroups (page 133 of the *Administrator Manual*) values, in order to create a provisioning hierarchy.

Creation and configuration of regular cloud node objects is conveniently carried out by cloning another regular cloud node from one of the default cloud nodes already created by the cluster extension wizard (section 3.1). A clickpath is:

`Cloud`→`Cloud Nodes`→<*cloud node hostname*>→↓`Clone`

Cloud node objects can also be created in `cmsh` as described in section 4.3.

The internal network for the regular cloud nodes is by default set to the VPC private network, and is somewhat similar to the internal network for regular nodes. The VPC private network can be contrasted with the VPC public network for cloud directors, which is a network that is by default assigned to cloud directors, and which floating IP addresses can connect to. Both the VPC private and VPC public networks are subnets of a cloud network. If the administrator would like to do so, the regular cloud nodes can be placed in the VPC public network and become directly accessible to the public.

If the cloud director is up, then the cloud nodes can be booted up by powering them up (section 4.2 of the *Administrator Manual*) by category, or individually.

3.4 Cluster Extension With AWS: Cloud Director And Cloud Node Startup From Snapshots

A technique that speeds up cluster deployment in the cloud is to use snapshots to start up the nodes in the cloud. Snapshots are snapshots of a shutdown state, and are stored by the cloud provider. In Amazon, they can be stored in EBS. It is cheaper to keep a machine in a stored state, rather than have it up but idling. Restoring from a snapshot is also significantly faster than starting up from scratch, due to optimizations by the cloud provider. An administrator should therefore get around to looking at using snapshots once cloudbursting is set up and the usage pattern has become clearer.

As a part of regular maintenance, snapshot configuration can be repeated whenever cloud director and cloud node files change significantly, in order to keep usage efficiency up.

3.4.1 Cloud Director Startup From Snapshots

Cloud Director Snapshot Preparation

A cloud director, for example `us-east-1-director`, can have a snapshot of its state prepared as follows by the administrator:

- The cloud director is started up from scratch (section 3.2)
- After it comes up for the first time, the administrator shuts it down cleanly. For example, with a command similar to `cmsh -c "device use us-east-1-director; shutdown"`
- After the cloud-director shutdown is complete, the administrator creates a snapshot of a cloud director using the EC2 Management Console. This can be done by selecting Elastic Block Store in the navigator column, then selecting the `Volumes` item within that menu. The volume associated with the cloud director can be identified by matching the `Attachment Information` column

value with the name `us-east-1-director` for this node, and the device to be snapshotted. In a default configuration, the device is `/dev/sdb` at the time of writing, but that may change. The `Actions` button in the main pane then provides a `Create Snapshot` item (figure 3.7).

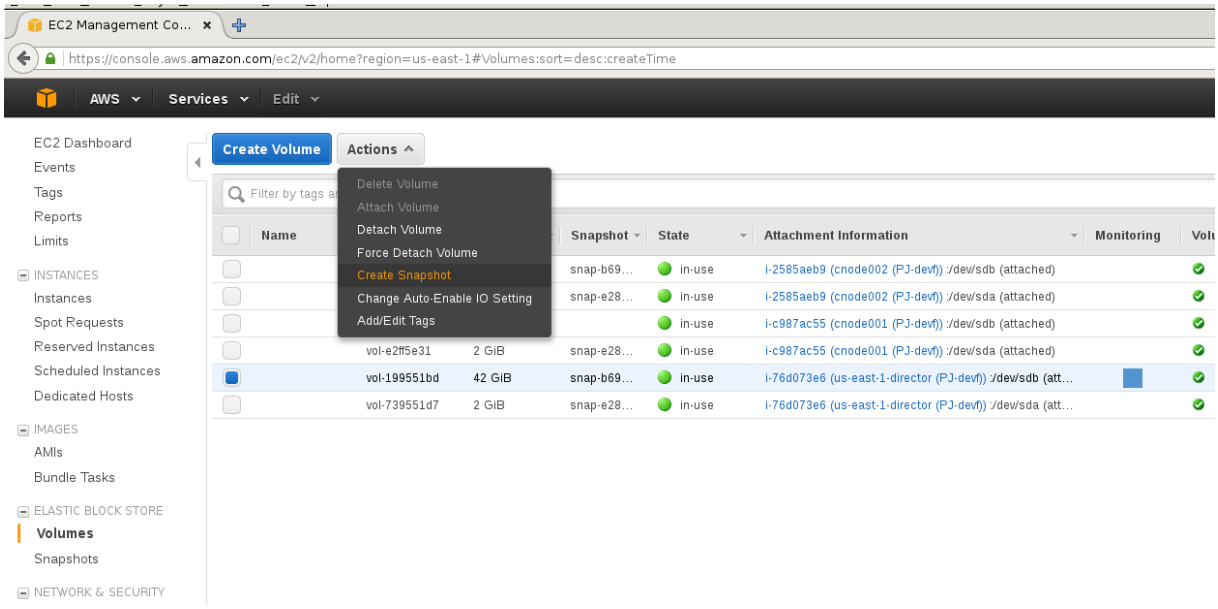

Figure 3.7: Creating A Snapshot From A Selected Volume

Using it creates a snapshot of a selected volume instance via a dialog. The snapshot ID is displayed at the end of the snapshot creation dialog, and should be noted for CMDaemon use later on, where it is saved as the value of `snapshotid`.

Created snapshots can be viewed within the `Snapshots` item of the Elastic Block Store menu.

Cloud Director Launch From Prepared Snapshot

To allow CMDaemon to launch the cloud director from the snapshot, the following procedure can be followed:

- The instance must be terminated so that the snapshot can actually be used by the instance on starting it again:

  Example

  ```
  [bright80->device[us-east-1-director]]% terminate
  us-east-1-director terminated
  ```

- The snapshot ID that was noted earlier during snapshot preparation is set in the EBS storage setting configuration of the CMDaemon database, using a session similar to:

  Example

  ```
  [root@bright80 ~]# cmsh
  [bright80]% device
  [bright80->device]% use us-east-1-director
  [bright80->device[us-east-1-director]]% cloudsettings
  [bright80->...-director]->cloudsettings]% storage
  [bright80->...-director]->cloudsettings->storage]% use ebs
  [bright80->...-director]->cloudsettings->storage[ebs]]% set snapshotid snap-2a96d0c6
  ```

The cloud director can now be powered on:

Example

```
[bright80->device[us-east-1-director]]% power on
```

The cloud director now starts up much faster than when starting up from scratch.

3.4.2 Cloud Node Startup From Snapshots

When a regular cloud node is launched from scratch (section 3.3), it uses the cloud director for provisioning, rather than a node outside the cloud, because this is faster. However, having the cloud director create an EBS volume from its storage in the cloud, and then providing the image to the cloud compute nodes still involves a lot of data I/O. On the other hand, a cloud provider such as Amazon can optimize many of these steps when creating an EBS volume from a snapshot, for example, by using copy-on-write. This means that snapshot-based provisioning is even speedier than the non-snapshot, "from scratch" method.

If the administrator wants to make a snapshot that can be used as the base for speedily launching regular cloud nodes, then the same snapshot method that is used for cloud directors (section 3.4.1) should be followed to make a snapshot for a regular cloud node.

A summary of the steps that can be followed is:

- a regular cloud node is started up from scratch (section 3.3), after the cloud director is up
- after the regular cloud node has come up, it is shut down cleanly
- a snapshot is created of the cloud node using the EC2 Management Console
- the cloud node is terminated
- the snapshot ID is set:

  Example

  ```
  [bright80->device[cnode001]->cloudsettings->storage[ebs]]% set snapshotid snap-5c9s3991
  ```

Powering on the node now launches the regular cloud node much faster than the non-snapshot method.

CMDaemon ensures that a snapshot for one cloud node can be used by other cloud nodes too, if the disk partitioning is the same. This is useful when launching cloud nodes that do not differ much from the snapshot.

It also means that even the cloud director image can be used as a snapshot to launch a regular cloud node, if the disk partitioning and other settings allow it. However, using a regular node snapshot for launch is usually much wiser, due to the extra filesystems that a cloud director has.

4

Cloudbursting Using The Command Line And cmsh

The command line and cmsh can be used to set up Cluster On Demand (section 4.1 for AWS) and Cluster Extension clusters (section 4.2 for AWS, Chapter 5 for Azure).

4.1 The cm-cluster-on-demand-aws Script For Cluster On Demand Cloud Clusters

For Cluster On Demand (pure cloud) setups, using a web browser to manage AWS is described in Chapter 2. In that chapter it is used to launch the cloud head node AMI from Amazon (section 2.1). Once an ssh connection is made by the administrator to the cloud head node, cloudbursting can be continued from command line. Thus, the cm-bright-setup script is run from the command line on the cloud head node (section 2.2), and the regular cloud nodes can then be powered up from the command line (section 2.4).

For Amazon Web Services, the cm-cluster-on-demand-aws script automates the entire process from the Bright Cluster Manager head node. This means that the administrator does not need to carry out the web browser procedure to start up a cloud head node (section 2.1), and also that the administrator does not need to run the cm-bright-setup script (section 2.2) within the cloud head node. The cm-cluster-on-demand-aws is part of Bright Cluster Manager's cluster-tools package.

4.1.1 cm-cluster-on-demand-aws Command Options Overview, Help Files

The main help text for the cm-cluster-on-demand-aws utility is:

```
[root@bright80 ~]# cm-cluster-on-demand-aws -h
usage: cm-cluster-on-demand-aws [-h] [-c C [C ...]] [-v] [-vv] [-vvv]  ...

Cluster on demand by Bright Computing

optional arguments:
  -h, --help     show this help message and exit
  -c C [C ...]   Config file path. You can specify multiple files to overlay
                 them on each other.
  -v, --verbose  Generate verbose output
  -vv            Generate very verbose log output
  -vvv           Generate extremely verbose log output

Top-level commands:

    cluster      Manage clusters
    image        Manage Images
```

```
    config       Manage runtime configuration of the tool
```

Further help for the commands is displayed using the `-h|--help` option for each command: The commands can carry out the following tasks:

- The `cluster` command: The following subcommands under this command are used to manage clusters:
  - `create`: creates a new cluster (in default region)
  - `delete`: deletes all resources in a cluster
  - `list`: list clusters (in default region)
  - `start`: start head node instances for clusters
  - `stop`: stop all instances for clusters
- The `image` command: The following subcommand under this command is used to manage an image:
  - `list`: List available Bright head node images (in default region)
- The `config` command: The following subcommands under this command are used to manage a configuration:
  - `dump`: output an example configuration
  - `dump --run-time`: display configuration that will be used after loading environment, configuration files, arguments, and overlaying, as a format compatible with a configuration file
  - `show`: display configuration that will be used after loading environment, configuration files, arguments, and overlaying, as a plain list of key-value pairs.

Yet further help is available for each subcommand with the `-h|--help` option, which shows the usage along with a large list of possible options. The help output is truncated in the text that follows.

Example

```
[root@bright80~]# cm-cluster-on-demand-aws cluster create -h
usage: cm-cluster-on-demand-aws cluster create
                                [-h]
                                [--aws-username AWS_USERNAME]
                                [--aws-account-id AWS_ACCOUNT_ID]
                                [--aws-access-key-id AWS_ACCESS_KEY_ID]
                                [--aws-secret-key AWS_SECRET_KEY]
                                [--aws-region AWS_REGION]
                                [--license-unit LICENSE_UNIT]
                                [--license-locality LICENSE_LOCALITY]
                                [--license-country LICENSE_COUNTRY]
                                [--license-product-key LICENSE_PRODUCT_KEY]
                                [--license-organization LICENSE_ORGANIZATION]
                                [--license-state LICENSE_STATE]
                                [--timezone TIMEZONE]
                                [--wlm WORKLOAD_MANAGER_NAME]
                                [--cluster-password CLUSTER_PASSWORD]
                                [--store-head-node-ip PATH_TO_FILE]
                                [--ssh-key-pair SSH_KEY_PAIR]
                                [--ssh-private-key-path SSH_PRIVATE_KEY_PATH]
                                [--ingress-ports INGRESS_PORTS]
```

```
                                [--ingress-cidr-block INGRESS_CIDR_BLOCK]
                                [--head-node-image ID]
                                [--head-node-type HEAD_NODE_TYPE]
                                [--head-node-root-volume-size SIZE_IN_GB]
                                [--head-node-root-volume-type HEAD_NODE_ROOT_VOLUME_TYPE]
                                [--nodes NODES]
                                [--node-type TYPE]
                                [--vpc-cidr-block CIDR]
                                [--public-subnet-cidr-block CIDR]
                                [--private-subnet-cidr-block CIDR]
                                [--tags TAGS]
                                [--version VERSION]
                                [--clusters CLUSTERS]
                                [--run-cm-bright-setup True,False]
                                name [name ...]

Create a new cluster in amazon VPC

positional arguments:
  name                  Name of the cluster (VPC) to create

optional arguments:
  -h, --help            show this help message and exit
  --aws-username AWS_USERNAME
                        AWS Username [$AWS_USERNAME]
  --aws-account-id AWS_ACCOUNT_ID
                        AWS Account ID. This value should be enclosed in
                        'quotes' [$AWS_ACCOUNT_ID]
  --aws-access-key-id AWS_ACCESS_KEY_ID
...
```

In the optional arguments in the preceding help text, the parameters indicated in the form of

[$*parameter*]

such as $AWS_USERNAME and $AWS_USERNAME, are environment variables that can be set for a session with, for example

```
export AWS_USERNAME=me@example.com
```

4.1.2 cm-cluster-on-demand-aws Configuration Options, YAML Configuration Files, And Environment Variables

To avoid having to specify a large number of options, a YAML configuration file can be used to provide the settings. The cm-cluster-on-demand-aws config dump command displays a configuration file template.

The top of the dumped display suggests how an administrator can set a configuration file under the /etc directory. It also suggests how a user can, in the home directory of the user, create a configuration file cm-cluster-on-demand.conf, or can create several overlayable files under a directory cm-cluster-on-demand.d, or can set up and use arbitrarily-named configuration files. Environment variables and command line flags can also be used to load configurations. A truncated example of the dump looks like:

Example

```
# To load it:
#    use '-c <path>' to load it, or
#    put it in ~/cm-cluster-on-demand.conf (it will be loaded automatically), or
```

```
#    put it in ~/cm-cluster-on-demand.d/ (it will loaded automatically and overlaid)
#
#  Tip: You can have a basic config file, and then overlay other config file on top of it
#       e.g. have ~/cm-cluster-on-demand.conf with your base settings, and then use
#       '-c' to load cluster specific settings.
#
#  Tip: If you use multiple config files, run the tool with '-v' to see
#       from which file exactly your runtime values come from
#
#  Tip: You can combine config files for different cloud providers (OpenStack and AWS )
#       in a single file
#
#  Config files load order:
#    - /etc/cm-cluster-on-demand.conf
#    - /etc/cm-cluster-on-demand.d/*
#    - ~/cm-cluster-on-demand.conf
#    - ~/cm-cluster-on-demand.d/*
#    - Environment Variables
#    - config files provided with '-c <path> [<path2>, ...]'
#    - command line flags
#
aws:
    # Licence unit
    # default: not specified
    #license_unit: not specified

    # Licence locality
    # default: not specified
    #license_locality: not specified

    # Two characters
    # default: US
    #license_country: US
...
```

The indentations are important in the template. Values should be set for the configuration options, or within the `cm-cluster-on-demand.conf` or the overlay files, for the following:

- `license_product_key` (environment variable `LICENSE_PRODUCT_KEY`)
- `cluster_password`
- `aws_username` (environment variable `AWS_USERNAME`)
- `aws_account_id` (environment variable `AWS_ACCOUNT_ID`)
- `aws_access_key_id` (environment variable `AWS_ACCESS_KEY_ID`)
- `aws_secret_key` (environment variable `AWS_SECRET_KEY`)
- `ssh_key_pair` (environment variable `SSH_KEY_PAIR`)
- `ssh_private_key_path`

Some of these values can alternatively be set as environment variables, as indicated by the text in parentheses in the preceding list.

Default values are the commented values in the configuration file. However, the defaults can be overridden by configuration settings elsewhere in the loading hierarchy shown in the top part of the file.

Configuration values can be overlaid. That means that values loaded earlier in the load order are overridden by values set later on, and if no value is set, then the default value remains. The cluster administrator may, for example, set a configuration under /etc/, which the user may override with their configuration in their home directory.

The command cm-cluster-on-demand-aws config dump --with-runtime displays

- explicitly set values
- overriding values from elsewhere in the loading hierarchy
- default values that are still at their defaults from the dumped template. The template being the one that can be obtained by executing cm-cluster-on-demand-aws config dump

In other words, the --with-runtime option displays, unsurprisingly, what one can call the actual runtime values. This can be useful to make it clear what configuration is active. The -v|--verbose option can incidentally provide further insight into what is happening with configuration lookups, when launching a cluster (section 4.1.3).

4.1.3 cm-cluster-on-demand-aws: Launching A Cluster

The cm-cluster-on-demand-aws utility can launch one, or several clusters in parallel, as follows:

Example

```
cm-cluster-on-demand-aws cluster create acluster, anothercluster
```

If the preceding command fails due to missing values in the configuration file, or missing command line configuration parameters, then cluster creation does not take place, and a message similar to the following is shown:

Example

```
The following config values are not set: license_product_key, cluster_password, aws_username,\
 aws_account_id, aws_access_key_id, aws_secret_key, ssh_key_pair, ssh_private_key_path.
You should generate and fill out the config file with
  /cm/local/apps/cluster-tools/bin/cm-cluster-on-demand-aws config dump > ~/cm-cluster-on-dem\
and.conf
```

If the cm-cluster-on-demand-aws utility is run successfully with the create command as in the preceding example, then the cm-bright-setup script is run automatically by default, as defined by default in the --run-cm-bright-setup option. The default number of regular cloud nodes that are configured by the script is set to 5. The regular cloud nodes can be started up with a power on command run from the head cloud node in the cloud. More cloud nodes can be cloned and started up as required, just like in regular non-cloud clusters.

The output from a creation run looks something like this (some output elided or truncated):

Example

```
[root@bright80 ~]# cm-cluster-on-demand-aws cluster create acluster -c anotherconf
16:23:03:     INFO: ------------------------------------------------------------------------
16:23:03:     INFO: Image name:   ami-095a387a
16:23:03:     INFO: Head nodes:   1 (m3.medium)
16:23:03:     INFO:       Nodes:   5 (m3.medium)
16:23:03:     INFO: ------------------------------------------------------------------------
```

```
16:23:04:     INFO: Setting up VPC
16:23:04:     INFO: Creating VPC for acluster in eu-west-1...
16:23:04:     INFO: Addinng internet connectivity...
16:23:04:     INFO: Creating public subnet...
16:23:05:     INFO: Creating head node (m3.medium)...
16:23:06:     INFO: Done setting up VPC for acluster.
16:23:06:     INFO: Starting head nodes...
16:23:06:     INFO: Wait for head nodes running...
16:23:23:     INFO: Starting bright setup...
16:23:23:     INFO: Waiting for ssh connection availability... ssh root@52.208.149.137...
16:23:33:     INFO: SSH connection timeout
...
16:24:30:     INFO: Retrying SSH connection attempt in 15 seconds...
16:24:45:     INFO: Connection succeeded!
16:24:45:     INFO: Wait for cloud-init to finish...
16:24:48:     INFO: Cloud-init completed!
16:24:48:     INFO: Starting cm-bright-setup...
16:25:06:     INFO: stdout Please wait...
16:25:10:     INFO: stdout Executing 23 stages
16:25:10:     INFO: stdout ################## Starting execution for 'Bright...
...
16:28:48:     INFO: stdout ## Progress: 82
16:28:48:     INFO: stdout #### stage: brightsetup: Start CMDaemon
16:28:51:     INFO: stderr Created symlink from /etc/systemd/system/multi-user.target....
16:28:52:     INFO: stdout ## Progress: 86
16:28:52:     INFO: stdout #### stage: brightsetup: Restore MOTD
16:28:52:     INFO: stdout ## Progress: 91
16:28:52:     INFO: stdout #### stage: brightsetup: Restore Bash RC
16:28:52:     INFO: stdout ## Progress: 95
16:28:52:     INFO: stdout ## Progress: 100
16:28:52:     INFO: stdout #### stage: brightsetup: Setup Periodic File...
16:28:52:     INFO: stdout tune2fs 1.42.9 (28-Dec-2013)
16:28:52:     INFO: stdout Setting interval between checks to 15552000 seconds
16:28:52:     INFO: stdout Setting time filesystem last checked to Thu Aug  4 16:28:52...
16:28:52:     INFO: stdout
16:28:52:     INFO: stdout
16:28:52:     INFO: stdout Took:     03:41 min.
16:28:52:     INFO: stdout Progress: 100/100
16:28:52:     INFO: stdout ################## Finished execution for 'Bright ...
16:28:52:     INFO: stdout
16:28:52:     INFO: stdout Bright Setup finished!
16:28:52:     INFO: stdout
16:28:52:     INFO: Settings default cloud instance type...
16:29:11:     INFO: Bright setup completed!
16:29:11:     INFO: Setup successful for clusters: acluster
16:29:11:     INFO: Script completed.
16:29:11:     INFO: ------------------------------------------------------------------
16:29:11:     INFO: Time it took:   06:07
16:29:11:     INFO:   SSH string:   ssh root@52.208.149.137 -i /root/eukeypair.pem
52.208.149.137  acluster
```

4.1.4 cm-cluster-on-demand-aws: Stopping And Terminating The Cluster

When using cm-cluster-on-demand-aws, stopping the cluster with the stop command means that the cluster is stopped, and that its objects remain in existence. Deleting the cluster with the delete command with a specific configuration means that the specified clusters are terminated, and the associated

objects removed:

Example

```
[root@bright80 ~]# cm-cluster-on-demand-aws cluster delete acluster -c anotherconf
This will destroy VPCs for acluster, continue?
Proceed? [yes/no] yes
17:11:45:     INFO: Stopping instances for VPCs on-demand acluster
17:11:45:     INFO: Listing instances...
17:11:45:     INFO: Issuing instances termination requests...
17:11:46:     INFO: Waiting until instances terminated...
17:12:31:     INFO: Destroying VPC on-demand acluster
17:12:31:     INFO: Deleting subnets...
17:12:31:     INFO: Detaching and deleting gateways...
17:12:32:     INFO: Flushing permissions...
17:12:33:     INFO: Deleting security groups...
17:12:34:     INFO: Deleting VPC...
17:12:34:     INFO: Done destroying VPC on-demand acluster
[root@bright80 ~]#
```

4.2 The cm-cluster-extension Script For Cluster Extension Clusters

4.2.1 Running The cm-cluster-extension Script On The Head Node For Cluster Extension Clusters

The cm-cluster-extension script is run from the head node. It is a part of the Bright Cluster Manager cluster-tools package. It allows cloudbursting to be carried out entirely from the command line for Cluster Extension setups. It is a command line way of carrying out the the configuration carried out by the GUI steps of section 3.1 for cloud provider login and cloud director configuration. After the script has completed its setup, then cmsh power commands can launch the required cloud nodes (sections 4.2.2 and 4.3).

The cm-cluster-extension script can be run in plain dialog mode (page 43) , or as an Ncurses dialog (page 45).

Running The cm-cluster-extension Command Line Options As A Shell Dialog

The administrator can specify command line options to cm-cluster-extension, as shown in its help text. The help text is displayed with the -h|--help option:

```
[root@bright80 ~]# cm-cluster-extension -h
Please wait...
usage: cm-cluster-extension
         [-v] [-h] [-c <config_file>]
         [--skip-modules <mod1,mod2,...>]
         [--only-these-modules <mod1,mod2,...>]
         [--dev] [--tests]
         [--undo-on-error]
         [--on-error-action debug,remotedebug,undo,abort]
         [--output-remote-execution-runner]
         [--json-output <path_to_file>]
         [--remove]  [--force]
         [--yes-i-know-this-is-dangerous-for-this-cluster <headnode_hostname>]
         [--test-networking]
         [--test-environment]
         [--test-configuration]
         [--test-everything]
```

```
common:
  Common arguments

  -v                    Verbose output
  -h, --help            Print this screen
  -c <config_file>      Load runtime configuration for modules from a YAML config file

advanced:
  Various *advanced* configuration options flags.

  --skip-modules <mod1,mod2,...>
                        Load and use all the default modules (cloudstorage, clusterextension),
                        except for these. Comma-separated list.
  --only-these-modules <mod1,mod2,...>
                        Out of all default modules, load and use only these. Comma-separated
                        list.
  --dev                 Enables additional command line arguments
  --tests               Test integrity of the utility by running Unit Tests
  --undo-on-error       Upon encountering a critical error, instead of asking the user for a
                        choice, setup will undo (revert) the deployment stages.
 --on-error-action debug,remotedebug,undo,abort
                        Upon encountering a critical error, instead of asking the user for choice,
                        setup will undo (revert) the deployment stages.
  --output-remote-execution-runner
                        Format output for CMDaemon
  --json-output <path_to_file>
                        Generate a file containing json-formatted summary of the deployment
  --play-scenario <path_to_file>
                        Reproduce wizard scenario recorded before
  --rec-scenario <path_to_file>
                        Record wizard scenario

removing cluster extension:
  Flags which can be used for removing AWS integration

  --remove              Remove definitions of all objects required for cluster extension, e.g.
                        cloud nodes, directors, cloud networks and cloud interfaces
  --force               Skip interactive confirmation for --remove
  --yes-i-know-this-is-dangerous-for-this-cluster <headnode_hostname>
                        Additional confirmation

testing cluster extension:
  Flags which can be used for troubleshooting

  --test-networking     Perform networking checks (e.g. check if API endpoints are reachable)
  --test-environment    Run environment checks (e.g. if proper RPMs are installed)
  --test-configuration  Run configuration checks, which check if cluster extension is properly
                        configured (e.g. if cloud director has correct interfaces, if cloud
                        credentials are valid, if CMDaemon can create/delete objects in the
                        cloud)
  --test-everything     Run all of the abovementioned checks.

examples:
  cm-cluster-extension                          (start interactive menu, wizard)
  cm-cluster-extension -c <config>              (configure bursting to AWS)
```

```
  cm-cluster-extension --remove                    (remove bursting)
  cm-cluster-extension --remove --force --yes-i-know-this-is-dangerous-for-this-cluster <hostname>
    (remove bursting, no confirmation)
     *WARNING* This will remove the cloud extension configuration. Any data located on your
     cloud nodes will be lost, unless you back up beforehand.
   cm-cluster-extension --test-everything

This tools looks for, and uses if found, the following environment variables:
   AWS_USERNAME, AWS_ACCOUNT_ID, AWS_ACCESS_KEY, AWS_SECRET_KEY
   AZURE_SUBSCRIPTION_ID, AZURE_TENANT_ID, AZURE_CLIENT_ID, AZURE_CLIENT_SECRET
```

It can be run with the options directly (some output skipped):

Example

```
[root@bright80 ~]# cm-cluster-extension --test-networking
Please wait...
Found an optional config file, '/root/cm-setup.conf'. Will attempt to load it.
Executing 26 stages
#################### Starting execution for 'Running networking checks'
Connecting to CMDaemon
  - cloudstorage
  - clusterextension
#### stage: clusterextension: Testing tcp connection to ec2.us-east-1.amazonaws.com:443
#### stage: clusterextension: Testing tcp connection to ec2.us-west-1.amazonaws.com:443
...
#### stage: clusterextension: Testing udp OpenVPN connectivity

Took:     00:09 min.
#################### Finished execution for 'Running networking checks', status: completed

Running networking checks finished!
```

Running The cm-cluster-extension Ncurses Dialog

The more user-friendly way to run cm-cluster-extension is to run it without options. This brings up the main screen for its Ncurses dialog (figure 4.1).

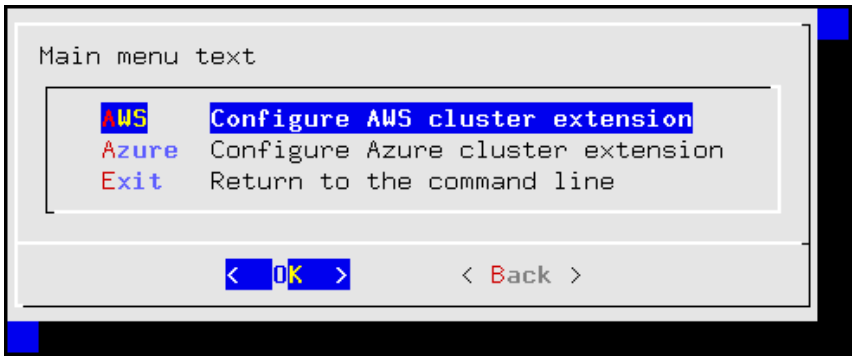

Figure 4.1: Configuration Processing With cm-cluster-extension: Main Screen

A cloud provider—Azure or AWS—can be selected from the main screen.

Cluster extension cloudbursting deployment with Azure is described in Chapter 5.

If AWS is selected, then a new AWS provider can be set if none is already set up. Testing only the network connectivity to the various AWS regions is also possible.

If a new AWS provider is selected, then a screen comes up asking for the credentials for Amazon (figure 4.2):

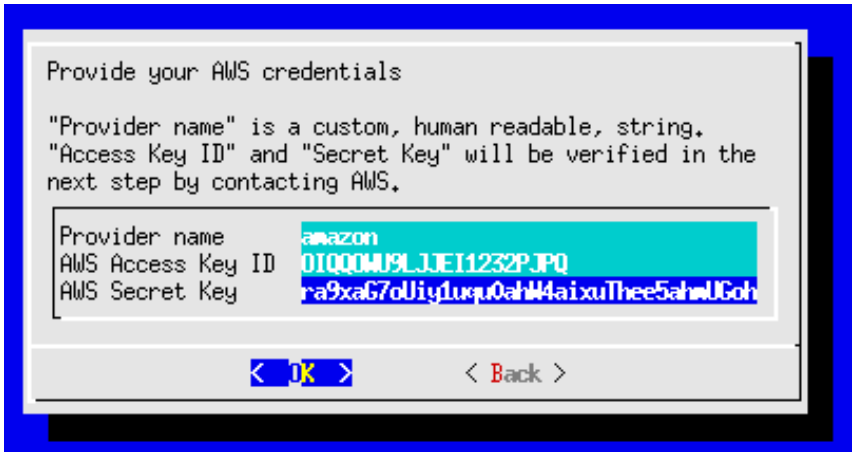

Figure 4.2: Configuration Processing With `cm-cluster-extension`: Obtaining Credentials

After checking the credentials, the initial number of cloud nodes to be set up in the cloud can be set (figure 4.3). A default of 3 is suggested.

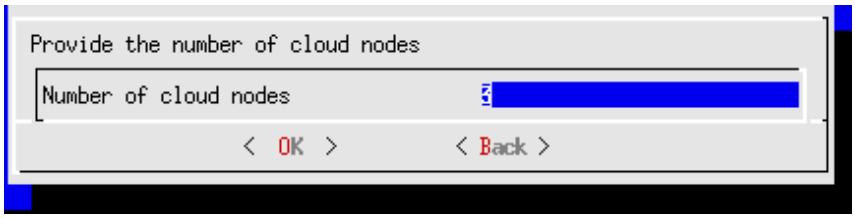

Figure 4.3: Configuration Processing With `cm-cluster-extension`: Setting The Initial Number Of Cloud Nodes

After setting an initial number of cloud nodes, the available regions into which these can be deployed are displayed (figure 4.4):

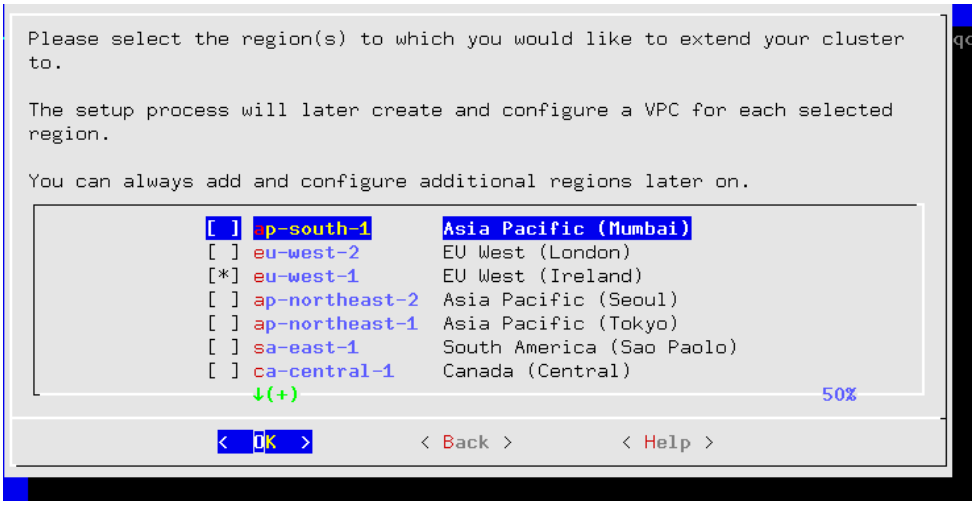

Figure 4.4: Configuration Processing With `cm-cluster-extension`: Regions Selection

After selecting one or more regions, a default instance type must be set for the regular cloud nodes.

m3.medium (3.75GB RAM, 4GB SSD, 3 EC2 compute units) is the suggested default (figure 4.5):

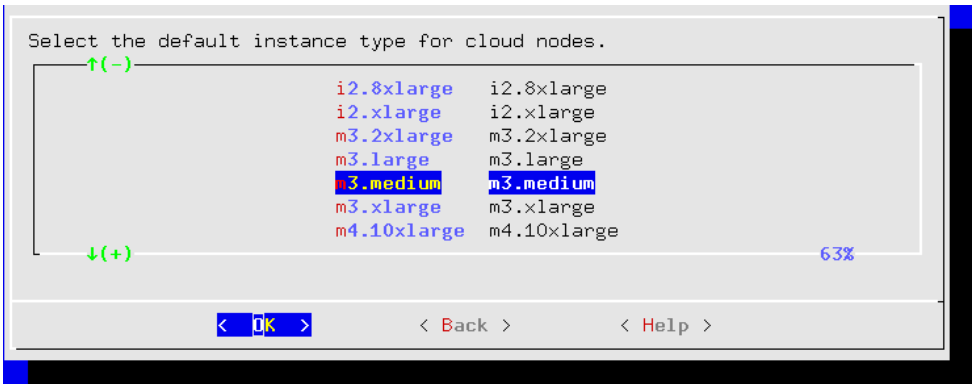

Figure 4.5: Configuration Processing With cm-cluster-extension: Default Instance Type

A similar default instance type screen asks for the cloud director node type, and m3.medium is again the suggested default.

The summary screen (figure 4.6) is a screen to let the administrator look things over before deployment:

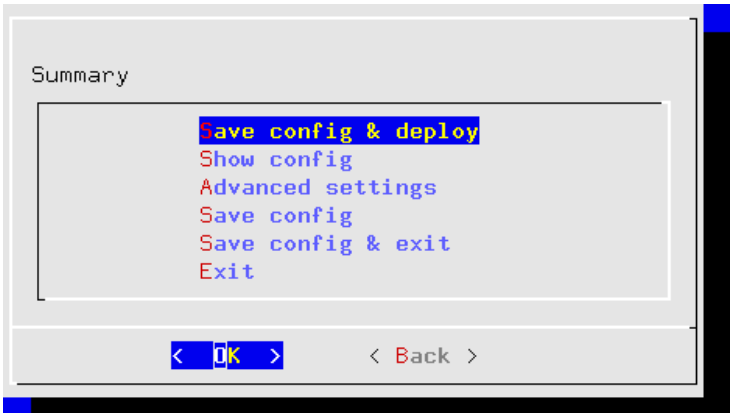

Figure 4.6: Configuration Processing With cm-cluster-extension: Summary Screen

The summary screen allows the following:

- An administrator can just go ahead, save the configuration, and deploy the cluster extension. This is usually the expected action.
- The configuration settings YAML file can be viewed. To scroll, the PageUp and PageDown keys are used.
- The advanced configuration settings screen can be accessed in addition to the standard settings. The advanced settings are usually left alone.
- The configuration file, <*configuration file*>, can be saved and the Ncurses dialog can be exited. By default, the value of <*configuration file*> is set to cm-cluster-extension.conf in the home directory of the user. On exiting the Ncurses dialog, deployment with that configuration can be carried out manually by running:

  cm-cluster-extension -c <*configuration file*>

After cm-cluster-extension has carried out a successful deployment, the cloud nodes (the cloud director and regular cloud nodes) can be launched.

4.2.2 Launching The Cloud Director For Cluster Extension Clusters

Launching the cluster in the cloud requires that the cloud director (section 3.2) and cloud nodes be powered up. This can be done using Bright View as described in sections 3.2 and 3.3. It can also be carried out in `cmsh`, for example, the cloud director `eu-west-1-director` can be powered up from device mode with:

Example

```
cmsh -c "device power on -n eu-west-1-director"
```

If the administrator is unsure of the exact cloud director name, one way it can easily be found is via tab-completion within the `device` mode of `cmsh`. Alternatively, the cloud directors for AWS can be listed with:

Example

```
cmsh -c "device; list -c aws-cloud-director"
```

As explained in section 3.2, the cloud director takes some time to power up. Its status can be followed in the notice messages sent to the `cmsh` session, or in the Bright View event viewer. The status can also be queried via the `status` command in device node. For example, a `watch` instruction such as:

```
[root@bright80 ~]# watch 'cmsh -c "device status -n eu-west-1-director"'
```

will show a series of outputs similar to:

```
eu-west-1-director ....... [ PENDING ] (Waiting for instance to start)
eu-west-1-director ....... [ PENDING ] (Waiting for instance to start)
eu-west-1-director ....... [ PENDING ] (IP assigned: 54.220.240.166)
eu-west-1-director ....... [ PENDING ] (setting up tunnel)
eu-west-1-director ....... [ INSTALLER_REBOOTING ]
eu-west-1-director ....... [ INSTALLING  ] (recreating partitions)
eu-west-1-director ....... [ INSTALLING  ] (FULL provisioning to "/")
eu-west-1-director ....... [ INSTALLING  ] (provisioning started)
eu-west-1-director [ INSTALLER_CALLINGINIT ] (switching to local root)
eu-west-1-director ....... [   UP   ]
```

4.3 Launching The Cloud Nodes

Once the cloud director is up on the cloud provider, the regular cloud nodes can also be powered up. This does however first require that the corresponding cloud node objects exist. That is, that CMDaemon must have a representation of the cloud nodes, even if they do not yet exist on the cloud provider. The objects must each have an IP address assigned to them that is consistent with that of the cloud director that manages them. That is, the network address of the cloud nodes must be what the cloud director expects.

Bright Cluster Manager's cluster extension utilities create 3 cloud node objects by default (figure 4.4, page 46). Cloning them is an easy way to extend the number of deployable cloud nodes.

With Bright View, this can be done with the `Clone` command to assign properties to the clone that match the original (section 3.3), but advance the relevant IP addresses by 1. In `cmsh`, the `clone` command works the same way.

To launch a cloud node that has an object, a command can be run as follows:

```
[bright80->device[cnode001]->interfaces]% device power on -n cnode001
```

4.3.1 Creating And Powering Up Many Nodes

For a large number of cloud nodes, the creation and assignment of IP addresses can be done with the clone option of the `foreach` command, (section 2.5.5 of the *Administrator Manual*), together with a node range specification. This is the same syntax as used to create non-cloud regular nodes with `cmsh`.

Earlier on in this section, starting from page 45, an Ncurses session was run that ended up creating

- the cloud director `eu-west-1-director` and
- regular node objects `eu-west-1-cnode001` up to `eu-west-1-cnode003`.

Continuing with the end result of that session, cloning many further regular cloud nodes can now be carried out by cloning `eu-west-1-cnode003`:

Example

```
[bright80->device]% foreach --clone eu-west-1-cnode003 -n eu-west-1-cnode0[04-12] ()
Warning: The Ethernet switch settings were not cloned, and have to be set manually
...
[bright80->device*]% commit
Successfully committed 9 Devices
[bright80->device]%
```

As a reminder, the node range option `-n eu-west-1-cnode004..eu-west-1-cnode012` would also be valid for the preceding example, and perhaps easier to comprehend, although longer.

The IP addresses are assigned to the cloud nodes via heuristics based on the value of `eu-west1-cnode003` and its cloud director.

Powering up many cloud nodes can carried out using `cmsh` with the node range option as follows:

Example

```
[bright80->device]% power on -n eu-west-1-cnode0[02-10]
```

4.4 Submitting Jobs With cmsub And Cloud Storage Nodes, For Cluster Extension Clusters

The `cmsub` command is a user command wrapper that submits job scripts to a workload manager in a Cluster Extension cluster, so that jobs are considered for running in the cloud. Its usage for an end user is covered in section 4.7 of the *User Manual*.

The `cmsub` command is available from the Bright Cluster Manager repository as part of the `cmdaemon-cmsub` package. The `cmsub` command needs the `cmsub` environment module (section 2.2 of the *Administrator Manual*) to be loaded by the end user before use. In addition, an administrator must assign the `cloudjob` profile (section 6.4 of the *Administrator Manual*) to `cmsub` users.

Example

```
[root@bright80 ~]# cmsh
[bright80]% user use henry
[bright80->user[henry]]% set profile cloudjob; commit
```

If the `cmsub` command is run by the user to submit a job, then the job is submitted to the workload manager, and the *data-aware scheduling* mechanism is initiated.

A cluster with data-aware scheduling is a cluster that ensures that it has the data needed for the cloud computing job already accessible on *cloud storage nodes*.

Cloud storage nodes are nodes that are set up by the cluster manager, before the job is executed in the cloud. Because data stored can be written and read from many cloud storage nodes for each job that

is placed in the cloud, the data throughput in the cloud becomes more efficient than if only one storage space were used.

Cloud storage nodes are powered up automatically if cmsub has been installed and configured. They must however be powered down explicitly, and this must be done before the cloud director that it depends on is powered down.

4.4.1 Installation And Configuration of cmsub For Data-aware Scheduling To The Cloud

The configuration of data-aware scheduling means configuring the cluster so that the tools that allow data-aware scheduling to work correctly are configured. The configuration that is carried out depends on the workload manager that is to be used.

If cmsub has not yet been set up, or if it needs reconfiguration, then the following steps should be carried out:

1. The cmdaemon-cmsub package is installed. It must be installed on the head node and in the software image that is to be used for compute cloud nodes and storage cloud nodes.

   Example

   ```
   [root@bright80 ~]# yum install cmdaemon-cmsub
   [...]
   [root@bright80 ~]# yum --installroot /cm/images/default-image install cmdaemon-cmsub
   [...]
   ```

2. The cm-cloud-storage-setup utility is run. Example runs are provided later, starting on page 51, but an explanatory background is given here first.

   The utility is part of the cluster-tools package, which is installed by default. The utility

   - configures cmsub properties
   - creates
     - *templates for cloud storage nodes*
     - *storage policies* for cmsub

Templates For Cloud Storage Nodes And Storage Policy

Templates for cloud storage nodes: are a cloud node definition associated with a cloud provider. Templates for cloud storage nodes, more conveniently called template nodes, provide a template that is used by the cloud storage nodes. Template nodes, being templates, are never powered on, and are therefore always in a Down state in cmsh and Bright View. Actual cloud storage nodes, on the other hand, can be powered on by the cluster manager, so that they can be used to store cloud job data.

In addition, any network interfaces associated with a template node can generally be regarded as non-functioning as far as the administrator is concerned. One feature of template nodes however is that the tunnel IP address set in the template is an offset to the network address that will be used to assign IP addresses to actual storage nodes.

A storage policy: defines other parameters for how storage for cloud jobs is handled. Its parameters include:

- Name: the name set for the policy
- Bucket Name: the S3 bucket used for cloud jobs to transfer input and output job data
- Default job output size: specifies the default free storage space that will be provisioned for the result that a job produces

- Storage node name prefix: specifies a prefix for how storage nodes are to be named. The prefix is cstorage by default. The number suffix scheme is as for regular nodes. Thus, by default, the storage nodes are cstorage001, cstorage002 and so on.
- Template for cloud nodes: the template to use as the prototype for storage nodes

Example

Configuration Of cmsub Properties With cm-cloud-storage

The cm-cloud-storage-setup is an Ncurses utility that configures cmsub properties for a cloud deployment. When run with the -h|--help option its usage is displayed:

```
[root@bright80 ~]# cm-cloud-storage-setup -h
Please wait...
usage: Cloud storage setup cm-cloud-storage-setup [-v] [-h] [-c <config_file>]
                                                  [--dev] [--tests]
                                                  [--undo-on-error]
                                                  [--on-error-action
                                                            {debug,remotedebug,undo,abort}]
                                                  [--output-remote-execution-runner]
                                                  [--json-output <path_to_file>]
                                                  [--play-scenario <path_to_file>]
                                                  [--rec-scenario <path_to_file>]
                                                  [--skip-reboot] [--remove]

optional arguments:
  --skip-reboot         Don't reboot the nodes

common:
  Common arguments

  -v                    Verbose output
  -h, --help            Print this screen
  -c <config_file>      Load runtime configuration for modules from a YAML config file

advanced:
  Various *advanced* configuration options flags.

  --dev                 Enables additional command line arguments
  --tests               Test integrity of the utility by running Unit Tests
  --undo-on-error       Upon encountering a critical error, instead of asking the user for
                        choice,setup will undo (revert) the deployment stages.
  --on-error-action {debug,remotedebug,undo,abort}
                        Upon encountering a critical error, instead of asking the user for
                        choice,setup will undo (revert) the deployment stages.
  --output-remote-execution-runner
                        Format output for CMDaemon
  --json-output <path_to_file>
                        Generate a file containing json-formatted summary of the deployment
  --play-scenario <path_to_file>
                        Reproduce wizard scenario recorded before
  --rec-scenario <path_to_file>
                        Record wizard scenario

Remove storage:
  --remove              Cleanup storage setup
```

The administrator is usually expected to run cm-cloud-storage-setup without arguments. This brings up the Ncurses-based dialog, which starts with an introductory page (figure 4.7):

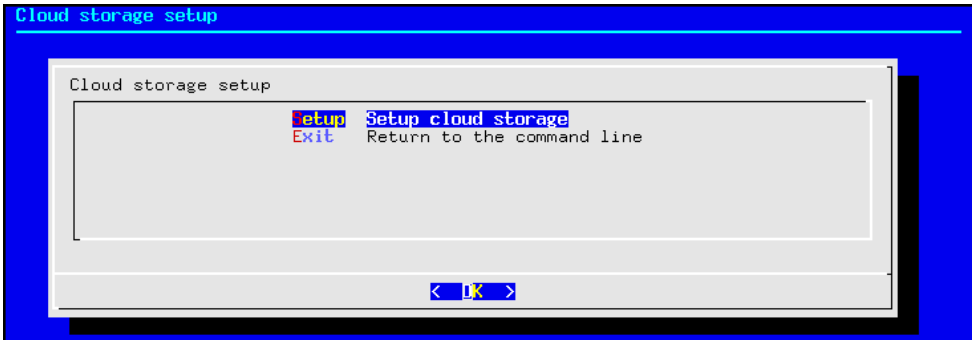

Figure 4.7: Cloud Storage Configuration Processing With cm-cloud-storage: Main Screen

Continuing on brings up the network selection screen, which sets the network in which the cloud storage is to be placed (figure 4.8):

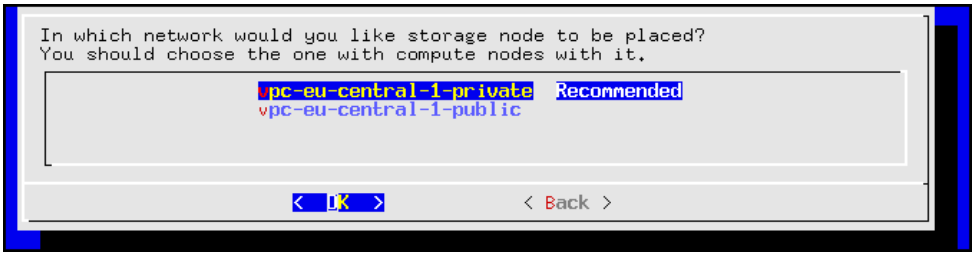

Figure 4.8: Cloud Storage Configuration Processing With cm-cloud-storage: Network Selection

After having selected the network, a category for the storage nodes is set (figure 4.9):

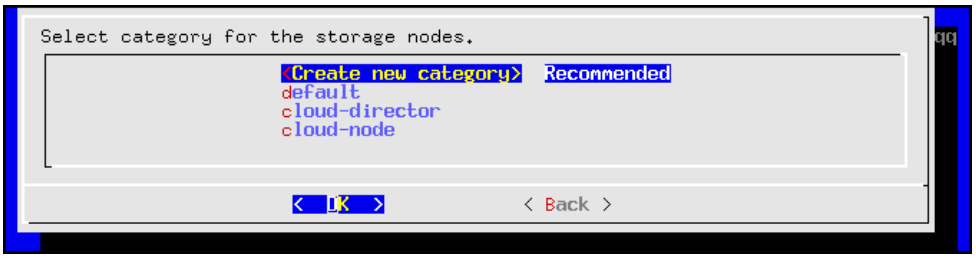

Figure 4.9: Cloud Storage Configuration Processing With cm-cloud-storage: Category Selection

It is recommended that a new category be created (figure 4.10):

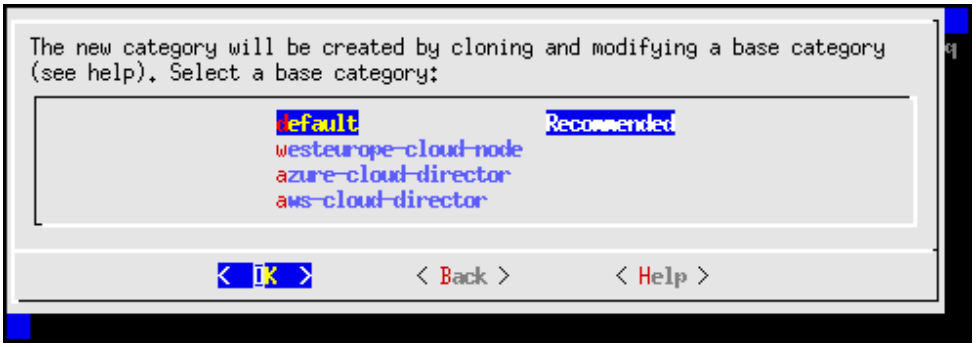

Figure 4.10: Cloud Storage Configuration Processing With cm-cloud-storage: Category Creation

A bucket name can be set (figure 4.11):

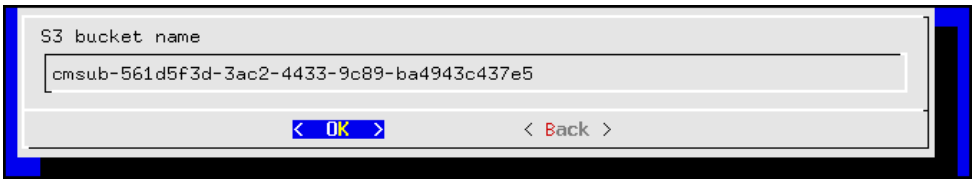

Figure 4.11: Cloud Storage Configuration Processing With cm-cloud-storage: S3 Bucket Entry

A workload manager is set for the cloud storage (figure 4.12):

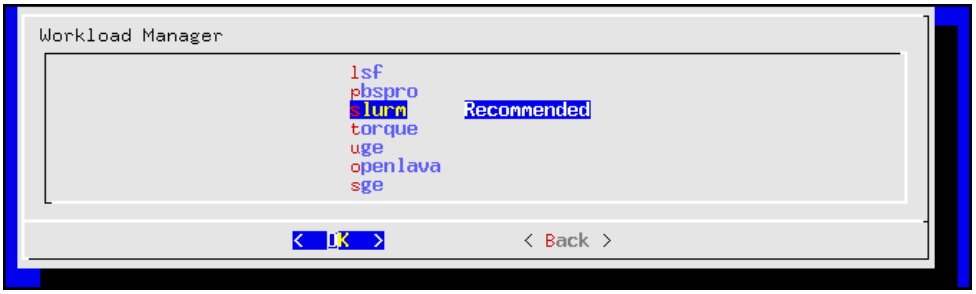

Figure 4.12: Cloud Storage Configuration Processing With cm-cloud-storage: Workload Manager Selection

It is recommended that a new queue be created for jobs that use the storage (figure 4.13):

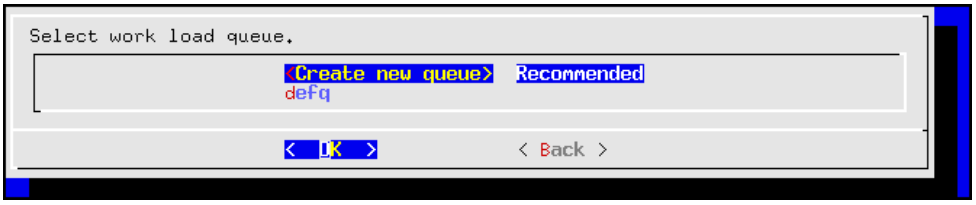

Figure 4.13: Cloud Storage Configuration Processing With cm-cloud-storage: Queue Creation

The queues can be assigned to the cloud category (figure 4.14):

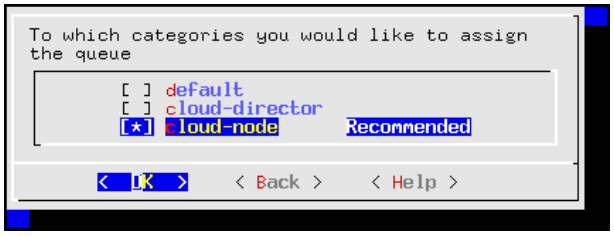

Figure 4.14: Cloud Storage Configuration Processing With cm-cloud-storage: Setting The Queue Category

The summary screen allows the configuration to be saved and to be deployed (figure 4.15):

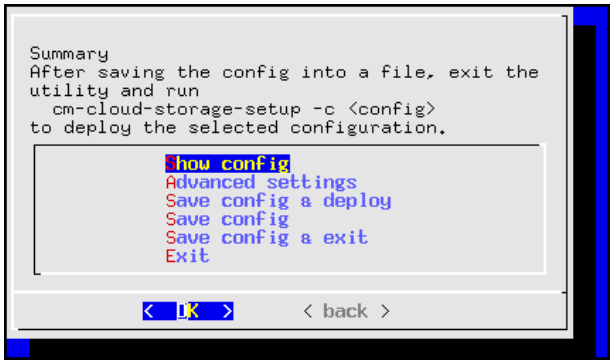

Figure 4.15: Cloud Storage Configuration Processing With cm-cloud-storage: Summary Screen

If the configuration is to be saved, then the file path should be specified (figure 4.16):

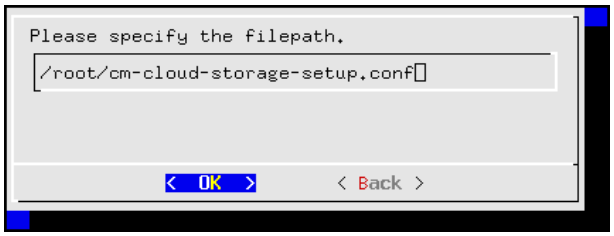

Figure 4.16: Cloud Storage Configuration Processing With cm-cloud-storage: Save & Deploy Screen

/root/cm-cloud-storage-setup.conf is the suggested default. On exit, the saved configuration can be run with:

Example

```
[root@bright80 ~]# cm-cloud-storage-setup -c cm-cloud-storage-setup.conf
```

4.5 Miscellaneous Cloud Commands

4.5.1 The cm-cloud-copy Tool

The cm-cloud-copy tool is automatically installed with the cmdaemon-cmsub package (section 4.4.1). Its purpose is to transfer data to and from AWS storage.

It is used automatically as a backend to cmsub to create and remove AWS storage containers in AWS S3, and to upload and download files and directories to those containers.

The cm-cloud-copy tool can also be used as a standalone tool by the cluster administrator directly, on loading the cm-cloud-copy module. However because its behavior may change due to development, it should only be used if the administrator has explicitly been instructed to do so by Bright

Computing engineers. At the time of writing, May 2016, its use requires that the `cm-cloud-copy` module is loaded. A short help can then be seen on running `cm-cloud-copy` without options, while more information and examples can be found in the `cm-cloud-copy(1)` man page.

5

Cluster Extension Cloudbursting With Azure

5.1 Introduction To Cluster Extension Cloudbursting With Azure

Cluster extension cloudbursting with AWS is covered in Chapter 3, where a GUI approach is described, and is also covered in section 4.2, where a text interface approach is described,

Cluster extension cloudbursting with Microsoft Azure is covered in this chapter (Chapter 5). The procedure is somewhat similar to that for AWS:

- The prerequisites to cloudburst into Azure are the same as those of cloudbursting into AWS, and are as previously described on page 21.

  In summary, the prerequisites are as follows:

  - an activated cluster license should be ensured
  - an Azure account should exist
  - Bright Cluster Manager must be registered as an Azure AD application (page 58)
  - UDP port 1194 should be open

- The techniques to cloudburst into Azure are also the same as that of cloudbursting into AWS, and are as previously described on page 22.

  In summary, the techniques are as follows:

  - a cloud director is set up in the cloud then started up
  - cloud nodes are then provisioned from the cloud director

- The deployment of cluster extension cloudbursting for Azure is carried out in a similar way to how it is done for AWS.

  In summary, the tools used to deploy cluster extension cloudbursting for Azure are as follows:

  - the Ncurses `cm-cluster-extension` utility, run from the command line (section 5.2)
  - the `Azure Wizard`, run from Bright View.

5.2 Cluster Extension Into Azure

Section 4.2.1 introduces the `cm-cluster-extension` Ncurses wizard, which is run on the head node when configuring a cluster extension for Azure or AWS. After the cluster extension configuration is completed, the cluster is capable of extending into the cloud by having cloud nodes power up into the cloud. This section (section 5.2) covers how `cm-cluster-extension` can be run for Azure, and how clouds nodes can be deployed for Azure.

There is also a Bright View browser-based wizard for cluster extension into Azure. The browser wizard is accessible via the clickpath: `Cloud→Azure→Azure Wizard`. If the browser wizard is used, then the documentation here (section 5.2) for `cm-cluster-extension` can be followed since it is a very similar procedure.

Running The `cm-cluster-extension` Command Line Options As A Shell Dialog

The `cm-cluster-extension` utility can be run as a dialog from the command line environment, as described in the section starting from page 43. Running it as an Ncurses dialog is however easier, and is described next.

Running The `cm-cluster-extension` Ncurses Dialog For Cluster Extension Into Azure

The `cm-cluster-extension` utility can be run as a more user-friendly Ncurses session within the text environment. In a session described starting from page 45, an AWS cluster extension configuration is generated and deployed. In the session that is now described here, an Azure cluster extension configuration is generated and deployed instead:

As in the AWS case, the `cm-cluster-extension` script is run without options, to bring up the Ncurses main screen (figure 5.1):

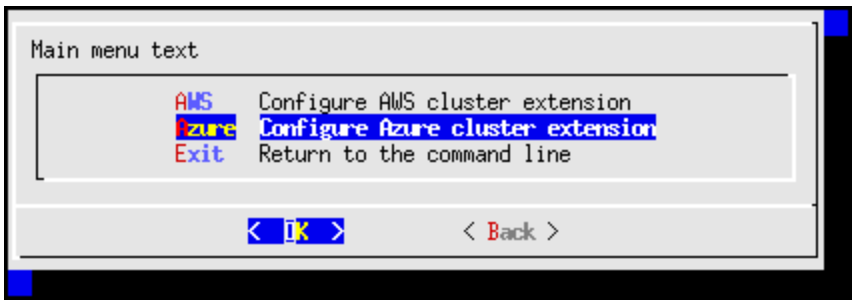

Figure 5.1: Configuration Processing With `cm-cluster-extension`: Azure Selection

The Azure option is selected in this session instead of the AWS option. The next screen is then the Azure credentials screen (figure 5.2):

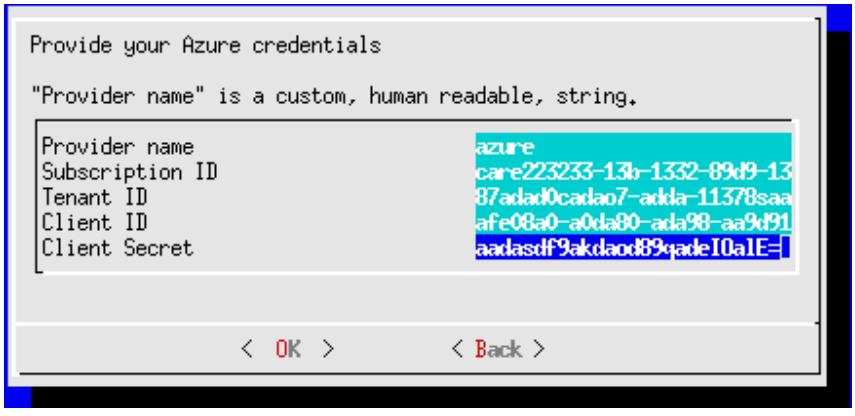

Figure 5.2: Configuration Processing With `cm-cluster-extension`: Azure Credentials

The inputs that are required here are Azure credentials, and exist for an activated Azure account which has had Bright Cluster Manager registered as an application.

How Bright Cluster Manager can be registered as an Azure application: Registration of an Azure application can be carried out with the Azure portal at `https://portal.azure.com`. The portal is

accessible with an active Azure account. After logging in, navigating to locations via clickpaths described next is possible. The steps to ensure a registration are then as follows:

- User settings should first be checked to see if users can register applications. A clickpath to view this is:

  `Azure Active Directory→User settings→Users can register applications`

  The state should be set to `yes`.

- The subscriptions should be checked for permissions. The clickpath to view subscriptions is simply:

  `Subscriptions`

  Within the subscription display the role can be viewed to determine if the account has adequate permissions to assign an AD application to a role. The account must have `Microsoft.Authorization/*/Write` access to assign an Azure AD application to a role. Assigning the `Contributor` role allows this access. Role-based access control (RBAC) is discussed at `https://docs.microsoft.com/en-us/azure/active-directory/role-based-access-control-manage-access-rest`

- The user, for example, `<fred>`, should be checked for permissions. A suitable clickpath would be:

  `Azure Active Directory→Users and groups→<fred>→Azure resources`

  The Azure resources for the user, who is assigned the subscription, should show the role and assignment value. Suitable settings would be:

  ```
  ROLE: Contributor
  ASSIGNED TO: <fred>
  ```

- `Network`, `Compute`, and `Storage` namespaces must be registered.

  A convenient way to check that this is the case, is to use the Azure CLI tool from the head node. Instructions for installing it are available at `https://docs.microsoft.com/en-us/cli/azure/install-azure-cli`.

  After installation, a list of namespaces can be seen by running:

  ```
  az provider list --query "[].{Provider:namespace, Status:registrationState}" --out table
  ```

  If the tool is run for the first time, then the tool gives the user a code and a URL. Using these, the user can authenticate the head node via a web browser.

  The required namespaces can be registered, if needed, with:

  ```
  az provider register --namespace Microsoft.Network --wait
  az provider register --namespace Microsoft.Compute --wait
  az provider register --namespace Microsoft.Storage --wait
  ```

- Create Azure Active Directory application:

  With all the permissions in place, the application can now be registered via the clickpath: `Azure Active Directory→App registrations`

  The application name and application URL values can be arbitrary since they are not actually used by `cm-cluster-extension`. The application type should be set to: `Web app / API`

- The `Client ID` and `Client Secret` values are only available to Azure admin users, or the Azure application owners. Regular users cannot obtain these values.

  If the security settings allow Azure users to define their own Azure applications, then they can in theory create their own Azure application under the subscription, and use the data for that Azure application to get a `Client ID` and `Client Secret`.

The credentials can now be picked up from the Azure portal account via the clickpaths shown in the following table:

Ncurses	Clickpath From Azure Portal Menu After Azure Portal Login
`Subscription ID`	`Subscriptions`→`SUBSCRIPTION ID`
`Tenant ID`	`Azure Active Directory`→`Properties`→`Directory ID`
`Client ID`	`Azure Active Directory`→`App registrations`→`APPLICATION ID`
`Client Secret`	`Azure Active Directory`→`App registrations`→`APPLICATION ID`→`Keys`→[*the generated key must be noted*]*

*After filling in the `DESCRIPTION` and `EXPIRES` fields at the end of this clickpath, and saving the values, the key is generated, and displayed once. The key must be noted down by the user because it cannot be retrieved.

The `Provider Name` in figure 5.2 can be set to any user-defined value. For Azure, a sensible, if unimaginative value, is simply `azure`.

After the credentials have been accepted, then Azure regions can be selected (figure 5.3):

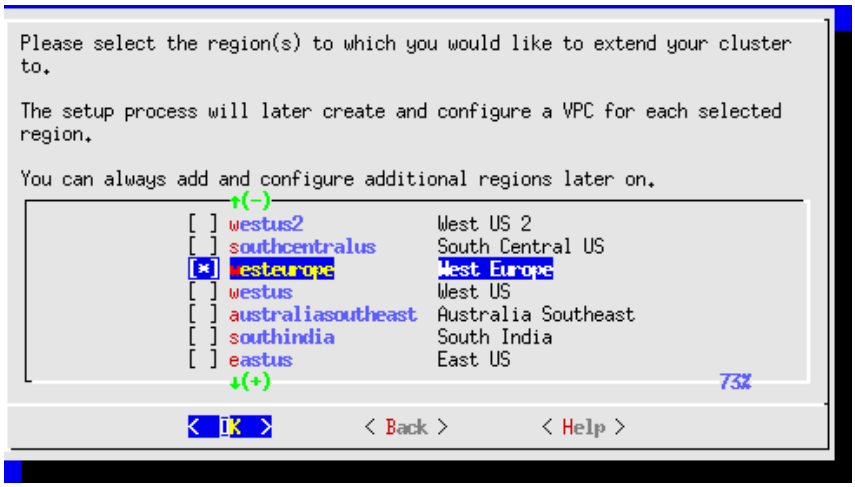

Figure 5.3: Configuration Processing With `cm-cluster-extension`: Azure Regions

Azure regions are regional data centers, and the cloud director for a region helps manage the regular cloud nodes in that region when the cluster is extended into there.

The default instance type for the regular cloud nodes is then set (figure 5.4):

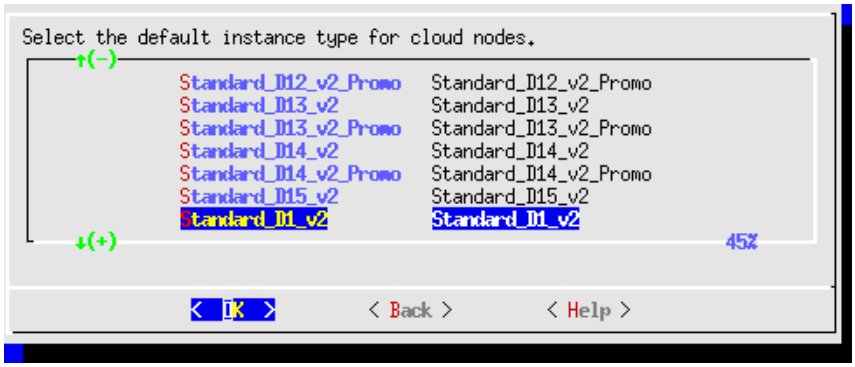

Figure 5.4: Configuration Processing With `cm-cluster-extension`: Azure Default Instance Type For Cloud Nodes

A default Azure virtual machine type that works well for general cloud node purposes is the `D1_v2` type. There are many possible machine types, and they are documented online at `https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes`. A summary listing of the supported types can be viewed in `cmsh` with:

```
[root@b80 ~]# cmsh -c "cloud use azure ; list types"
```

or via the Bright View clickpath:

`Cloud→Azure→Azure VM Sizes`

The default instance type for the cloud directors is then set (figure 5.5):

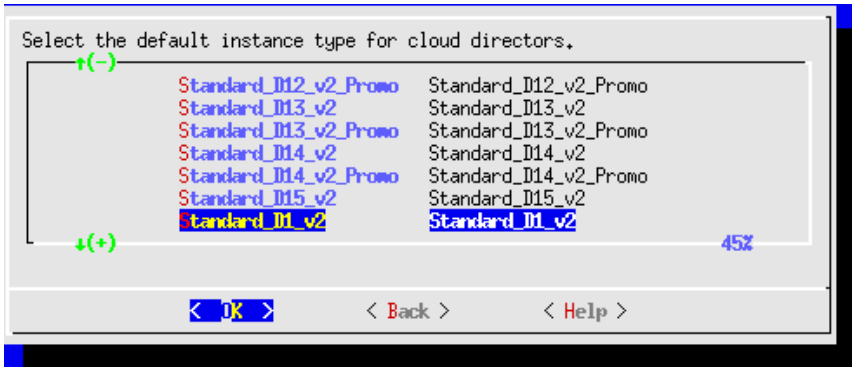

Figure 5.5: Configuration Processing With `cm-cluster-extension`: Azure Default Instance Type For Cloud Directors

The same default size of the `D1_v2` type that works for regular cloud nodes is typically adequate for cloud director nodes too, for small clusters.

The summary screen (figure 5.6) allows the administrator to look over the configuration before deployment:

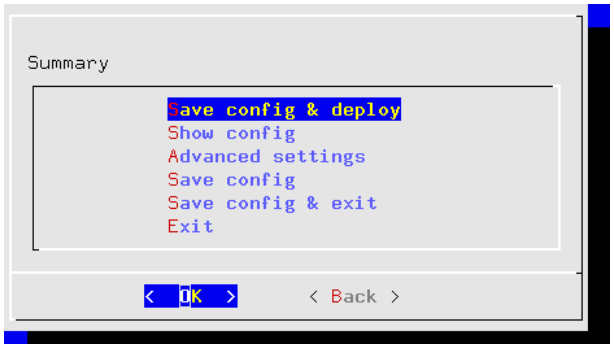

Figure 5.6: Configuration Processing With `cm-cluster-extension`: Azure Options Summary Screen

As in the AWS summary screen, the Azure summary screen allows the following:

- An administrator can just go ahead, save the configuration, and deploy the cluster extension. This is usually the expected action.
- The configuration settings YAML file can be viewed. To scroll, the PageUp and PageDown keys are used.
- An advanced configuration settings screen can be accessed in addition to the standard settings. The advanced settings are usually left alone.
- The configuration file, <*configuration file*>, is `cm-cluster-extension.conf` by default. The file can be saved, by default to the home directory of the user. On exiting the Ncurses dialog, deployment with that configuration can be carried out manually by running:

  `cm-cluster-extension -c` <*configuration file*>

Deployment Of An Azure Configuration Created With `cm-cluster-extension`

During configuration deployment, as the configuration is processed, text output indicates the progress. At the end of processing, the message

```
Azure Cloud extension configuration finished
```

indicates that the cluster has been extended successfully.

No nodes are activated yet within Azure. To start them up, the components of the cluster extension service for Azure must be started up by

- powering up the cloud directors, as introduced for AWS in section 3.2. The procedure for Azure is similar.
- powering on the cloud nodes after the cloud directors are up. This may require first creating new cloud nodes, as introduced for AWS in section 3.3. The procedure for Azure is similar.

When powering up, the cloud director can be installed from scratch (section 3.2), or from a snapshot. For example, running the `power on` command from the `device` mode of `cmsh` on a head node shows amongst others the following states (some output elided or ellipsized):

Example

```
[b80->device]% power on westeurope-director
cloud ..................... [   ON    ] westeurope-director
... [notice] b80: westeurope-director [ PENDING ] (Instance has started)
... [notice] b80: New certificate request with ID: 9
... [notice] b80: westeurope-director [     INSTALLING      ] (node installer started)
```

```
... [notice] b80: New certificate request with ID: 10 (ldaps)
... [notice] b80: westeurope-director [ INSTALLER_CALLINGINIT ] (switching to local root)
...
... [notice] b80: westeurope-director [   UP   ]
```

Example

```
[root@bright80 ~]# ps uww -C rsync | grep -o ' /cm/.*$'
 /cm/shared/ syncer@172.21.255.251::target//cm/shared/
```

Tracking Cloud Director Startup From `cmsh`

The `provisioningstatus` command in the `softwareimage` mode of `cmsh` can be used to view the provisioning status (some output elided):

Example

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus"
...
+ westeurope-director
...
  Up to date images:        none
  Out of date images:       default-image
```

In the preceding output, the absence of an entry for "`Up to date images`" shows that the cloud director does not yet have an image that it can provision to the cloud nodes. After some time, the last few lines of output should change to something like:

Example

```
+ westeurope-director
...
  Up to date images:        default-image
```

This indicates the image for the cloud nodes is now ready.

With the `-a` option, the `provisioningstatus -a` command gives details that may be helpful. For example, while the cloud director is having the default software image placed on it for provisioning purposes, the source and destination paths are `/cm/images/default-image`:

Example

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        4
Source node:          bright80
Source path:          /cm/images/default-image
Destination node:     westeurope-director
Destination path:     /cm/images/default-image
...
```

After some time, when the shared filesystem is being provisioned, then an indication of progress is shown by the `Request ID` incrementing, and the source and destination paths changing to the `/cm/shared` directory:

```
[root@bright80 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        5
Source node:          bright80
Source path:          /cm/shared
Destination node:     westeurope-director
Destination path:     /cm/shared
...
```

After the shared directory and the cloud node software images are provisioned, the cloud director is fully up. Cloud node instances can then be powered up and provisioned from the cloud director. The instances can be started up from scratch (section 3.2), or from snapshot (section 3.4).

5.3 Cluster Extension Into Azure: Cloud Node Startup From Scratch

This section discusses the configuration of regular cloud node startup from scratch.[1]

To *configure* regular cloud nodes in Azure from scratch does not require a working cloud director. However to *boot up* the regular cloud nodes does require that the cloud director be up, and that the associated networks to the regular cloud nodes and to the head node be configured correctly.

If needed, additional cloud provisioning nodes (section 5.2 of the *Administrator Manual*) can be configured by assigning the provisioning role to cloud nodes, along with appropriate nodegroups (page 133 of the *Administrator Manual*) values, in order to create a provisioning hierarchy.

Similarly to how it is done in AWS, the creation and configuration of regular cloud node objects in Azure is conveniently carried out by cloning another regular cloud node, from one of the default cloud nodes already created by the cluster extension wizard (section 5.2). A clickpath to do this cloning in Bright View is:

`Cloud`→`Cloud Nodes`→<*cloud node hostname*>→↓`Clone`

Cloud node objects can also be created in `cmsh` as described in section 4.3.

5.4 Cluster Extension Into Azure: `shutdown` *Vs* `power off`

An Azure cloud node has two stopped states:

1. `stopped`: A node running within Azure can be set to this state by running the `shutdown` command:
   - within the `device` mode of `cmsh`
   - with Bright View, using the clickpath:
     `Cloud` → `Cloud Nodes` → <*cloud node hostname*>→↓`OS`→ `Shutdown`
   - Or by clicking the `stop` button for that node within the Azure web portal, once.
2. `stopped (deallocated)`: A node running within Azure can be set to this state by running the `power off` command:
   - within the `device` mode of `cmsh`
   - with Bright View, using the clickpath:
     `Cloud`→`Cloud Nodes`→<*cloud node hostname*>→↓`Power`→`Off`
   - Or by clicking on the `stop` button within the Azure web portal, twice.

Carrying out a `power off` command is like a hard power off command, which can under some unusual conditions cause filesystem corruption. It is therefore safer to run the `shutdown` command first, wait for the node to shut down via the OS. After that, running the `power off` command ensures that the node is deallocated.

From a financial point of view when using Azure, a node that is shut down but not deallocated continues to incur costs. However, a node that is deallocated does not continue to incur costs.

[1]The configuration of cloud director and node startup from snapshot is also possible. How to do this for AWS is discussed in section 3.4. Doing this for Azure is rather more complicated and confusing. At the time of writing (August 2017), configuring this is therefore planned as a wizard-assisted option for a future version of Bright Cluster Manager, with a priority that depends on the level of interest for this feature from customers.

5.5 Submitting Jobs With cmsub And Cloud Storage Nodes, For Azure Cluster Extension Clusters

The cmsub utility, a job submission wrapper for end users for cluster extension clusters, is introduced and documented for AWS cluster extension in section 4.4. As a summary, its configuration procedure consists of:

- Making sure the cmdaemon-cmsub package is installed on the head node and the cloud image
- Ensuring the users that are to use cmsub have the cloudjob profile
- Making sure that cloud storage nodes are set up. The administrator can run cm-cloud-storage-setup to configure the cloud storage nodes.

The details of the configuration procedure for cmsub for Azure cluster extension is almost identical to that for AWS. The configuration procedure documented in section 4.4 can therefore also be followed for Azure.

How the end user can use cmsub is documented in section 4.7 of the *User Manual*.

6

Cloud Considerations And Choices With Bright Cluster Manager

6.1 Differences Between Cluster On Demand And Cluster Extension

Some explicit differences between Cluster On Demand and Cluster Extension clusters are:

Cluster On Demand	Cluster Extension
cloud nodes only in 1 region	cloud nodes can use many regions
no cloud director	uses one or more cloud directors per region
no failover head node	failover head node possible
no VPN or NetMap	VPN and NetMap
no externalnet interface on head	can have an external interface
cluster has publicly accessible IP address	cloud directors have publicly accessible IP addresses

A note about the last entry: The access to the cloud director addresses can be restricted to an administrator-defined set of IP addresses, using the "`Externally visible IP`" entry in figure 3.1 of the *Administrator Manual*.

6.2 Hardware And Software Availability

Bright Computing head node AMIs are available for the following distributions: RHEL6/7, SL6/SL7, CentOS6/CentOS7, and SLES 11/12.

AMIs with GPU computing instances are available with Amazon cloud computing services, and can be used with Bright Computing AMIs with `hvm` in the name (not `xen` in the name).

To power the system off, a `shutdown -h now` can be used, or the power commands for Bright View or `cmsh` can be executed. These commands stop the instance, without terminating it. Any associated extra drives that were created need to be removed manually, via the `Volumes` screen in the `Elastic Block Store` resource item in the navigation menu of the AWS Management Console.

6.3 Reducing Running Costs

6.3.1 Spot Pricing

The spot price field is a mechanism to take advantage of cheaper pricing made available at irregular[1] times. The mechanism allows the user to decide a threshold spot price (a price quote) in US dollars per hour for instances. Instances that run while under the threshold are called *spot instances*. Spot instances are described further at `http://aws.amazon.com/ec2/spot-instances/`.

With the pricing threshold set:

- If the set spot price threshold is above the instantaneous spot price, then the spot instances run.
- If the set spot price threshold is below the instantaneous spot price, then the spot instances are killed.
- If the set spot price threshold is `N/A`, then no conditions apply, and the instances will run on demand regardless of the instantaneous spot price.

An *on demand instance* is one that runs regardless of the price, according to the pricing at `http://aws.amazon.com/ec2/pricing/`.

A *persistent request* is one that will retry running a spot instance if the conditions allow it.

6.3.2 Storage Space Reduction

Reducing the amount of EBS disk storage used per cloud node or per cloud director is often feasible. 15 GB is usually enough for a cloud director, and 5 GB is usually enough for a cloud node with common requirements. In `cmsh` these values can be set with:

Example

```
[bright80]% device cloudsettings eu-west-1-director
[bright80->device[eu-west-1-director]->cloudsettings]% storage
[bright80->...->cloudsettings->storage]% set ebs size 15GB; commit
[bright80->...->cloudsettings->storage]% device cloudsettings cnode001
[bright80->device[cnode001]->cloudsettings]% storage
[bright80->...->cloudsettings->storage]% set ebs size 5GB; commit
```

The value for the cloud node EBS storage can also be set via Bright View, using the clickpath:

`Cloud→Cloud Nodes→Edit→Cloud settings→Storage→ebs→Edit→size`

6.4 Address Resolution In Cluster Extension Networks

6.4.1 Resolution And `globalnet`

The `globalnet` network is introduced in section 3.2.3 of the *Administrator Manual*. It allows an extra level of redirection during node resolution. The reason for the redirection is that it allows the resolution of node names across the entire cluster in a hybrid cluster, regardless of whether the node is a cloud node (cloud director node or regular cloud node) or a non-cloud node (head node, regular node or networked device). A special way of resolving nodes is needed because the Amazon IP addresses are in the 10.0.0.0/8 network space, which conflicts with some of the address spaces used by Bright Cluster Manager.

There are no IP addresses defined by `globalnet` itself. Instead, a node, with its domain defined by the `globalnet` network parameters, has its name resolved by another network to an IP address. The resolution is done by the nameserver on the head node for all nodes.

[1] irregular turns out to be random within a tight range, bound to a reserve price. Or rather, that was the case during the period 20th January–13th July, 2010 that was analyzed by Ben-Yehuda et al, `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-info.cgi/2011/CS/CS-2011-09`

6.4.2 Resolution In And Out Of The Cloud

The networks, their addresses, their types, and their domains can be listed from the `network` mode in `cmsh`:

```
[bright73->network]% list -f name:26,type:12,netmaskbits:8,baseaddress:13,domainname:24
name (key)                 type         netmaskb baseaddress   domainname
-------------------------- ------------ -------- ------------ ------------------------
us-east-1                  Tunnel       16       172.21.0.0
externalnet                External     24       192.168.100.0 brightcomputing.com
globalnet                  Global       0        0.0.0.0       cm.cluster
internalnet                Internal     16       10.141.0.0    eth.cluster
netmap                     NetMap       16       172.30.0.0
vpc-eu-central-1-private   Cloud (VPC)  17       10.42.128.0   vpc-eu-central-1.cluster
vpc-eu-central-1-public    Cloud (VPC)  24       10.42.0.0     vpc-eu-central-1.cluster
```

In a Type 1 network (section 3.3.6 of the *Installation Manual*), the head node is connected to `internalnet`. When a cloud service is configured, the head node is also "connected" to the CMDaemon-managed NetMap "network". It is useful to think of NetMap as a special network, although it is actually a network mapping from the cloud to `internalnet`. That is, it connects (maps) from the nodes in one or more cloud networks such as the `us-east-1` network provided by Amazon, to IP addresses provided by `netmap`. The mapping is set up when a cloud extension is set up. With this mapping, packets using NetMap go from the cloud, via an OpenVPN connection to the NetMap IP address. Once the packets reach the OpenVPN interface for that address, which is actually on the head node, they are forwarded via Shorewall's IPtables rules to their destination nodes on `internalnet`.

With default settings, nodes on the network `internalnet` and nodes in a cloud network such as `us-east-1` are both resolved with the help of the `cm.cluster` domain defined in `globalnet`. For a cluster with default settings and using the cloud network `us-east-1`, the resolution of the IP address of 1. a regular node and 2. a regular cloud node, takes place as follows:

1. `node001`, a regular node in the `internalnet` network, is resolved for `node001.cm.cluster` to

   (a) `10.141.0.1`, when at the head node. The cluster manager assigns this address, which is on `internalnet`. It could also be an `ibnet` address instead, such as `10.149.0.1`, if InfiniBand has been configured for the nodes instead of Ethernet.

   (b) `172.30.0.1` when at the cloud director or regular cloud node. The cluster manager assigns this address, which is a NetMap address. It helps route from the cloud to a regular node. It is not actually an IP address on the interface of the regular node, but it is convenient to think of it as being the IP address of the regular node.

2. `cnode001`, a regular cloud node in the `us-east-1` network, is resolved for `cnode001.cm.cluster` to:

   (a) 172.21.0.1 when at the head node. The cluster manager assigns this address, which is an OpenVPN tunnel address on `us-east-1`.

   (b) an IP address within 10.0.0.0/8 (10.0.0.1–10.255.255.254) when at a regular cloud node or at a cloud director. The Amazon cloud network service assigns the addresses in this network to the cloud director and regular cloud nodes after it notices the regular cloud node interface is up.

An explanation of the networks mentioned in the preceding list follows:

- The nodes within all available cloud networks (all networks such as for example, `us-east-1`, `us-west-1`, and so on) are given CMDaemon-assigned addresses in the cloud node space range

172.16.0.0–172.29.255.255. In CIDR notation that is: 172.16.0.0/12 (172.16.0.0–172.31.255.255), except for 172.31.0.0/15 (172.30.0.0–172.31.255.255).

- The network address space 172.30.0.0/16 (172.30.0.0–172.30.255.255) is taken by the CMDaemon-assigned NetMap network, explained shortly.
- Each node in a cloud network is also assigned an address in the network addressing space provided by Amazon VPC networking. The assignment of IP addresses to nodes within the 10.0.0.0/8 range is decided by Amazon via DHCP.

  The VPC networks for regular cloud nodes and cloud director nodes are subnets in this range.
- The `netmap` "network" (figure 6.1) is a helper mapping reserved for use in routing from the cloud (that is, from a cloud director or a cloud node) to a regular node. The mapping uses the 172.30.0.0/16 addressing scheme. Its routing is asymmetrical, that is, a NetMap mapping from a regular node to the cloud does not exist. Packets from a regular node to the cloud do however resolve to the `cloud` network as indicated by 2(a) in the preceding.

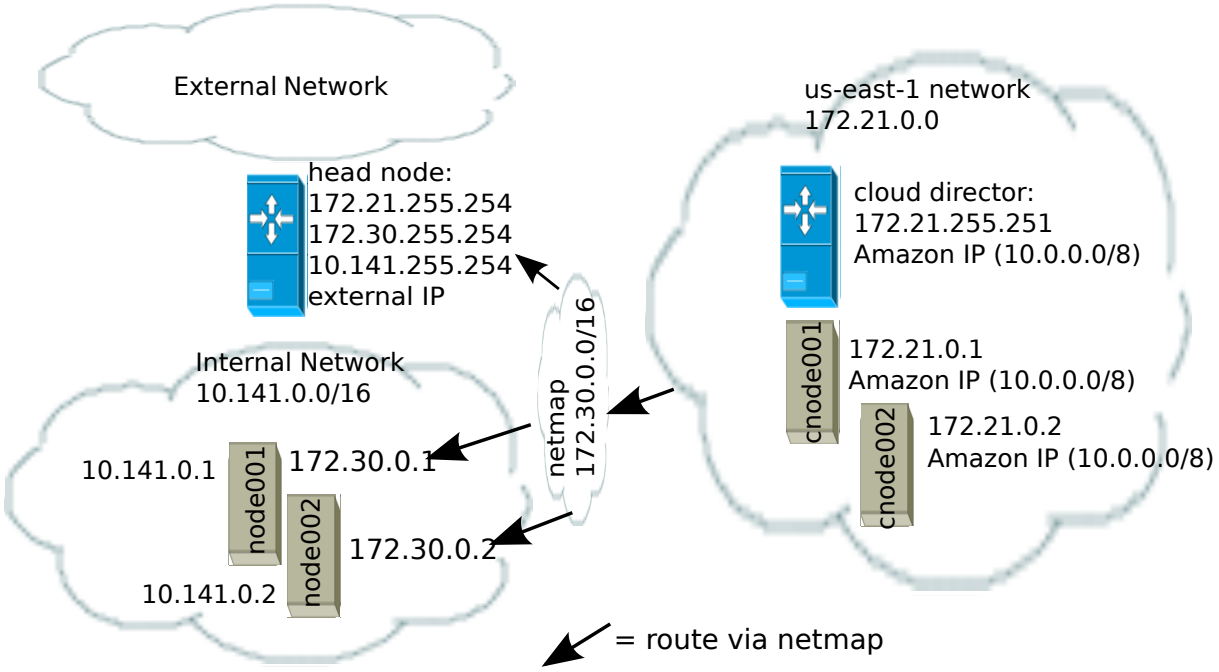

Figure 6.1: NetMap In Relation To The General Network Scheme

As pointed out in the introduction to this section (6.4), the main reason for the IP addressing network scheme used is to avoid IP address conflicts between nodes within the cloud and nodes outside the cloud.

The *difference* in resolution of the IP address for the nodes as listed in points 1 and 2 in the preceding text is primarily to get the lowest overhead route between the source and destination of the packet being routed. Thus, for example, a packet gets from the regular cloud node to the cloud director with less overhead if using the Amazon cloud IP addressing scheme (10.0.0.0/8) than if using the Bright OpenVPN addressing scheme (172.21.0.0/16). A secondary reason is convenience and reduction of networking complexity. For example, a node in the cloud may shut down and start up, and get an arbitrary Amazon IP address, but using an OpenVPN network such as `us-east-1` allows it to retain its OpenVPN address and thus stay identified instead of having the properties that have been assigned to it under Bright Cluster Manager become useless.

7

Legacy/Current Cloud Instances, And Their Network Addressing Allocations

A *virtual private cloud* is an implementation of a cluster on a virtual network in a cloud service provider. The Amazon Virtual Private Cloud (Amazon VPC) is an implementation of such a virtual private cloud. The Amazon VPC is documented at `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-vpc.html`.

Bright Cluster Manager manages VPCs so that the administrator can focus on using them productively, instead of on working out VPC configurations. However, some background on how the integration is done may be useful for administrators that have special requirements, or who are migrating from a legacy Amazon cloud.

The following VPC-related terms are explained and compared in this chapter:

- *EC2-Classic* (page 71)
- *EC2-VPC* (page 72)
- *classic cloud* (page 72)
- *defaultVPC* (page 72)
- *private cloud* (page 72)
- *custom VPC* (page 72)
- *elastic IP addresses* (page 75)

7.1 EC2-Classic And EC2-VPC

7.1.1 EC2-Classic Vs EC2-VPC Overview

So far, this manual has discussed configuring clusters within Amazon EC2. The EC2 designation actually covers two kinds of platforms:

- EC2-Classic: This is no longer supported by Bright Cluster Manager from version 7.3 onwards. It provides an environment that corresponds to a physical network. Instances in the same region exist on the same physical network and rely on explicitly configured security groups to restrict unauthorized access from other instances on the same network. A cloud instance that is created in such a network can be called a classic cloud cluster, or simply a classic cloud.

  Amazon is gradually phasing out the EC2-Classic platform.

- EC2-VPC: This platform is replacing EC2-Classic. It provides an environment corresponding to an isolated virtual network. A cloud cluster instance implemented on this virtual network is thus a virtual private cloud, or VPC, as described at the start of this chapter (Chapter 7).

  The EC2-VPC platform offers some extra features that are not available, or not as easy to configure, on the EC2-Classic platform:

  - Multiple VPCs can be configured per region
  - The inherent isolation of Amazon VPCs makes them more secure by default
  - their network properties can be customized

The isolated network design of a VPC means that instances started within a VPC cannot by default communicate with instances outside. *Elastic IP* addresses (page 75) are used to explicitly allow communication with the outside.

7.1.2 EC2-Classic Vs EC2-VPC And AWS Account Creation Date

The type of platform that can be accessed by an AWS account varies as indicated by the following table:

Account Creation Date	Typical Platform Offered
Before start of 2013	EC2-Classic only
In first half of 2013	EC2-Classic or EC2-VPC*
After first half of 2013	EC2-VPC only, in most or all regions

*Typically depends on the region accessed.

AWS accounts created after 4th December 2013 do not provide an EC2-Classic platform. However, to maintain backward compatibility for users who are migrating to EC2-VPC, and who have applications that run on the EC2-Classic platform, Amazon provides the defaultVPC instance on the EC2-VPC platform.

7.1.3 The Classic Cloud And The DefaultVPC Instances

The *classic cloud* is a cloud instance that EC2-Classic supports. This is not supported by Bright Cluster Manager from version 7.3 onwards.

The `defaultVPC` instance is a special VPC instance that emulates EC2-Classic behavior on the EC2-VPC platform. This allows legacy applications that do not support EC2-VPC to run on it. A legacy application that runs in a defaultVPC instance may be thought of as having its EC2-Classic API calls translated into EC2-VPC API calls. The defaultVPC instance is available in all regions that do not offer the EC2-Classic platform, but it is not supported by Bright Cluster Manager.

7.1.4 The Private Cloud And Custom VPC Instances

A *private cloud* (without the "virtual" in front) is the term used in the Bright Cluster Manager manuals, as well as by Amazon, and in general, for a general VPC instance.

A *custom VPC* is the term used in the manual to mean a general VPC instance, but one that is not a defaultVPC instance.

Thus, in terms of math sets:

`private clouds` = `custom VPCs` + `defaultVPCs`

In the context of Amazon VPCs, the term private cloud is often used by administrators, by convention and for convenience, to mean the more precise term of custom VPC as defined here, implicitly ignoring possible defaultVPC instances. The Bright Cluster Manager software itself also follows this convention, which is almost invariably true for Amazon VPCs nowadays, and an absolute truth as far as Bright Cluster Manager from version 7.3 onwards is concerned. So, as far as Bright Cluster Manager 7.3 onwards is concerned:

`private clouds` = `custom VPCs`

7.1.5 Cloud Cluster Terminology Summary

The cluster terminology used so far can be summarized as follows:

cluster term	platform	type and connectivity
classic cloud	EC2-Classic	classic cloud cluster that has direct connectivity to the outside
defaultVPC	EC2-VPC	a VPC that looks like it has direct connectivity to the outside because it emulates a classic cloud cluster
custom VPC	EC2-VPC	isolated VPC with no connectivity to the outside by default, and NAT gateway connectivity to the outside when made to connect
private cloud	EC2-VPC	both defaultVPC and custom VPC

7.2 Comparison Of EC2-Classic And EC2-VPC Platforms

There are several differences between EC2-Classic and EC2-VPC platforms. The most important ones are:

- Cloud nodes created inside the EC2-VPC platform do not have an external (public) IP address assigned to them by default. An exception to this is the case of nodes running in a defaultVPC instance, which emulates EC2-Classic network behaviour. Having no public IP address by default allows for a greater degree of out-of-the-box security.
- Custom VPCs are self-contained and securely isolated from the instance of other users.
- Custom VPCs are partitioned into multiple network segments, called *subnets* (section 7.3.1).
- It is possible to specify a custom base network address for the custom VPC. This is in contrast to the EC2-Classic platform, where a base network address always has the value of 10.0.0.0/8. For a defaultVPC instance the base network address takes the value of 172.30.0.0/16.

7.3 Setting Up And Creating A Custom VPC

From Bright Cluster Manager version 7.3 onwards, the EC2-classic platform is no longer available, and all nodes within the cloud provider always run within an EC2-VPC platform.

Custom VPC for Bright Cluster Manager 8.0 subnet allocation, and allocation of EIPs (External IPs) is described in sections 7.3.1–7.3.4.

7.3.1 Subnets In A Custom VPC

The components of a custom VPC include subnets, the nodes that run in them, and static IP addresses. The subnets are logical network segments within the network range of that custom VPC. Subnets can be thought of as interconnected with a central "magic" router, with Bright Cluster Manager managing the routing tables on that router. The routing ensures correct subnet communication. Inside Bright Cluster Manager, subnets are represented as a type of network (section 3.2 of the *Administrator Manual*), with a value for type set in `cmsh` to `Cloud (VPC)`, or in Bright View set to `CLOUD`.

Subnets for a custom VPC must have non-overlapping ranges. If there are multiple custom VPCs being managed by Bright Cluster Manager, then a particular subnet may be assigned to one custom VPC at the most.

Two series of valid network ranges could be:

Example

1. 10.0.0.0-10.0.31.255 (10.0.0.0/19),

10.0.32.0-10.0.63.255 (10.0.32.0/19),
10.0.64.0-10.0.95.255 (10.0.64.0/19).

2. 192.168.0.0-192.168.0.255 (192.168.0.0/24),
192.168.1.0-192.168.1.255 (192.168.1.0/24).

The `sipcalc` command (page 54 of the *Administrator Manual*) is a useful tool for calculating appropriate subnet ranges. At least one subnet must be assigned to a custom VPC before an instance can be created in that cloud. Typically two or more subnets are assigned, as shown in the custom VPC creation example in the following section.

7.3.2 Creating The Custom VPC

After subnets have been configured, a custom VPC can be created by specifying:

- the name
- the default region
- base address
- number of netmask bits

The network of the custom VPC must obviously be a superset of its subnets. Any subnets of the custom VPC must also be specified. Subnets can be added to or removed from an already-created custom VPC, but only if any cloud node instances within them are terminated first.

There are several ways to set up and create the subnets and custom VPC instance in Bright Cluster Manager:

1. by using the command line `cm-cluster-extension` utility (section 4.2),
2. by using the Bright View private cloud creation wizard (section 3.1),
3. by manually creating and configuring the `private cloud` object using `cmsh`.

Option 3 is tedious, but does show to the reader some of what the `cm-cluster-extension` utility and cloud creation wizard do. To create and configure a private cloud as in option 3, the following example sessions show how a private cloud can be built with `cmsh`. In the sessions, the subnets to be used for the custom VPC are created first, before creating the private cloud:

- **Subnet creation and cloning:** In the following example session, an arbitrary naming scheme is used for subnets, with a pattern of: `<name of custom VPC>-sn-<number>`. Here, `sn` is an arbitrary abbreviation for "subnet":

  Example

```
[bright80->network]% add vpc-0-sn-0
[bright80->network*[vpc-0-sn-0*]]% set type cloud
[bright80->network*[vpc-0-sn-0*]]% set baseaddress 10.0.0.0
[bright80->network*[vpc-0-sn-0*]]% set netmaskbits 24
[bright80->network*[vpc-0-sn-0*]]% set ec2availabilityzone eu-west-1a
[bright80->network*[vpc-0-sn-0*]]% commit
```

  Setting the `ec2availabilityzone` property is optional. It causes the subnet to be created in a specific availability zone. Leaving its value empty creates the subnet inside a randomly chosen availability zone. Having all subnets of the custom VPC inside the same availability zone is advised for better network performance. The availability zone set for the network must be one of the availability zones available for the region inside which the private cloud will be created.

  Once the first subnet has been created, it can be cloned:

Example

```
[bright80->network]% clone vpc-0-sn-0 vpc-0-sn-1
[bright80->network*[vpc-0-sn-1*]]% set baseaddress 10.0.1.0
[bright80->network*[vpc-0-sn-1*]]% commit
```

- **Custom VPC creation:** The following example session in the `privateclouds` submode of the `cloud` mode, creates a private cloud called `vpc-0`. The private cloud is actually a custom VPC according to the strict definition of a private cloud instance in the section on page 72. It is of type `ec2` and within a network that contains the two subnets specified earlier.

Example

```
[bright80->cloud[Amazon EC2]->privateclouds]%
[bright80->...->privateclouds]% add ec2privatecloud vpc-0
[bright80->...->privateclouds*[vpc-0*]]% set region eu-west-1
[bright80->...*[vpc-0*]]% set baseaddress 10.10.0.0
[bright80->...*[vpc-0*]]% set netmaskbits 16
[bright80->...*[vpc-0*]]% set subnets vpc-0-sn-0 vpc-0-sn-1
[bright80->...*[vpc-0*]]% commit
```

7.3.3 Elastic IP Addresses And Their Use In Configuring Static IP Addresses

Unlike the deprecated defaultVPC and EC2-Classic instances, a custom VPC instance does not have an externally visible (public) IP address assigned to it by Amazon by default. Without an externally visible IP address, the custom VPC cannot communicate with the internet, and it cannot even be an endpoint to an outside connection. To solve this issue, Amazon *elastic IP addresses* (EIPs) can be used to assign a public IP address to a custom VPC.

EIP addresses are the public IP addresses that Amazon provides for the AWS account. These addresses are associated with defaultVPC and EC2-Classic cloud instances by Amazon by default. These addresses can also be associated with custom VPC instances. The public addresses in the set of addresses can then be used to expose the custom VPC instance. In this manual and in Bright Cluster Manager, EIPs are referred to as "static IPs" in the cloud context. When allocating a static IP addres, the exact IP address that is allocated is a random IP address from the pool of all public IP addresses made available in the specified region by the configured cloud provider.

Automatic allocation of static IP addresses:

When a cloud director instance is started inside a custom VPC, CMDaemon automatically allocates and assigns a static IP address to it. By default, the static IP address is automatically released when the cloud director instance is terminated. This behavior can be changed in the CMDaemon cloud settings for the cloud director.

Manual allocation of static IP addresses:

It is also possible to manually allocate a static IP address to a cloud director using Bright View or `cmsh`, when the administrator decides to set it.

Allocating a static IP address in `cmsh` is done using the `staticip allocate` command, followed by the string indicating the region in which the static IP address is to be allocated. In `cmsh`, the command is issued inside a cloud provider object. A new static IP address is then made available and can be assigned to instances running within custom VPCs.

After allocation, the static IP address can be assigned and reassigned to any instance inside any custom VPC created within the region in which the IP address was allocated.

Example

```
[bright80] cloud use amazonec2
[bright80->cloud[Amazon EC2]]% staticip allocate us-west-1
Allocating Static IP. Please wait...
Successfully allocated the following static IP: 54.215.158.42
[bright80->cloud[Amazon EC2]]% staticip list
Cloud Provider   Cloud Region  Static IP      Assigned to
---------------- ------------- -------------- ---------------
Amazon EC2       us-west-1     54.215.158.42  <not assigned>
[bright80->cloud[Amazon EC2]]%
```

An allocated static IP can be released with the `staticip release` command in `cmsh`:

Example

```
[bright80->cloud[Amazon EC2]]% staticip release 54.215.158.42
Releasing static IP 54.215.158.42. Please wait...
Successfully released the static ip.
[bright80->cloud[Amazon EC2]]%
```

Once the IP address has been released, it may no longer be used for instances defined in the custom VPC.

The `staticips` command lists all allocated static IPs for all configured cloud providers.

The `staticip list` command lists static IP addresses for the currently active cloud provider object.

In Bright View, the static IPs can be managed via the clickpath `Cloud→AWS Static IPs→Allocate IP`.

7.3.4 Subnets With Static IP Addresses In A Custom VPC Recommendation

Subnets can be set up in many ways inside a custom VPC. The following is recommended:

- There must be exactly one network containing all the instances which have static IP addresses. This network should contain the cloud director. The network with the cloud director is arbitrarily referred to as the "public" network.
- There must be zero or more networks containing instances with no static IP addresses assigned to them. Such networks are arbitrarily referred to as the "private" subnets, although the "public" network without an EIP allocation is just as private. Normally the "private" network is intended to be used for regular cloud nodes. It is also possible to move the regular cloud nodes to the "public" network, which is convenient if, for example, packets to the EIP is to be routed to the regular cloud nodes via NAT.

Instances in the private subnets have no static IP addresses assigned to them, so by default they do not communicate with outside networks. To allow them to connect to the outside, the cloud director instance is automatically configured by CMDaemon as a NAT gateway for outside-bound traffic, for the instances existing inside the private subnets.

For example, the cloud director node and regular cloud nodes can be configured inside the `us-west-1` network as follows:

Example

```
[bright80->device[us-west-1-director]->interfaces]% list
Type         Network device name  IP             Network
------------ -------------------- -------------  ----------
physical     eth0 [dhcp]          0.0.0.0        vpc-us-west-1-public
tunnel       tun0 [prov]          172.18.255.251 us-west-1
```

Example

```
[bright80->device[us-west-1-cnode001]->interfaces]% list
Type         Network device name  IP           Network
------------ -------------------- ------------ --------------------
physical     eth0 [dhcp]          0.0.0.0      vpc-us-west-1-private
tunnel       tun0 [prov]          172.18.0.1   us-west-1
```