

Bright Cluster Manager 8.1

Cloudbursting Manual

Revision: 8511be6

Date: Tue May 20 2025



©2020 Bright Computing, Inc. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Bright Computing, Inc.

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. PBS Professional, PBS Pro, and Green Provisioning are trademarks of Altair Engineering, 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

Table of Contents	i
0.1 About This Manual	v
0.2 About The Manuals In General	v
0.3 Getting Administrator-Level Support	vi
0.4 Getting Professional Services	vi
1 Introduction	1
2 Cluster On Demand Cloudbursting With Azure Or AWS	3
2.1 Requirements For COD Cloudbursting	3
2.2 COD Via Bright Computing Customer Portal	4
2.3 COD Via Docker Image	9
2.3.1 COD Via Docker Image–Procedure Summary	9
2.3.2 COD Via Docker Image–Procedure Details	10
2.4 Using The AWS EC2 Management Console	18
2.4.1 Status Checking Via Instance Selection From Instances List	18
2.4.2 Acting On An Instance From The AWS EC2 Management Console	19
2.4.3 Connecting To An Instance From The AWS EC2 Management Console	19
2.4.4 Viewing The Head Node Console	19
2.4.5 Security Group Configuration To Allow Access To The Headnode Via <code>cmsh</code> Or Bright View	20
2.5 Using The Azure Dashboard	21
2.6 COD: Cloud Node Start-up	22
2.6.1 COD: IP Addresses In The Cloud	23
2.7 COD With AWS: Optimizing AWS For High Performance Computing (HPC)	23
3 Cluster Extension Cloudbursting	25
3.1 Cluster Extension With AWS: The Bright View Cluster Extension Wizard	27
3.1.1 Introduction	27
3.1.2 AWS Credentials	27
3.1.3 Select Regions	28
3.1.4 Select Software Images	30
3.1.5 Summary & Deployment	30
3.1.6 Deploy	31
3.2 Cluster Extension With AWS: Cloud Director Startup From Scratch	32
3.2.1 Setting The Cloud Director Disk Storage Device Type	33
3.2.2 Setting The Cloud Director Disk Size	34
3.2.3 Tracking Cloud Director Startup	35
3.3 Cluster Extension With AWS: Cloud Node Startup From Scratch	37
3.4 Cluster Extension With AWS: Cloud Director And Cloud Node Startup From Snapshots	37
3.4.1 Cloud Director Startup From Snapshots	37

3.4.2	Cloud Node Startup From Snapshots	39
3.5	Cluster Extension With AWS: Optimizing AWS For High Performance Computing (HPC)	39
3.5.1	Optimizing HPC Performance: EBS Volume Type	40
3.5.2	Optimizing HPC Performance: Placement Groups	40
3.5.3	Optimizing HPC Performance: Disabling Hyper-Threading	40
3.5.4	Optimizing HPC Performance: Using Elastic Network Adapter Instances	41
3.5.5	Optimizing HPC Performance: Using A Different Clock Source	41
3.5.6	Optimizing HPC Performance: Setting The Socket Buffer Sizes And TCP/IP Parameters In The Software Image	41
4	Cluster Extension Cloudbursting With AWS Using The Command Line And <code>cmsh</code>	43
4.1	The <code>cm-cluster-extension</code> Script For Cluster Extension Clusters	43
4.1.1	Running The <code>cm-cluster-extension</code> Script On The Head Node For Cluster Extension Clusters	43
4.1.2	Launching The Cloud Director For Cluster Extension Clusters	48
4.2	Launching The Cloud Nodes	48
4.2.1	Creating And Powering Up Many Nodes	49
4.3	Submitting Jobs With <code>cmsub</code> And Cloud Storage Nodes, For Cluster Extension Clusters	49
4.3.1	Installation And Configuration of <code>cmsub</code> For Data-aware Scheduling To The Cloud	50
4.4	Miscellaneous Cloud Commands	54
4.4.1	The <code>cm-cloud-copy</code> Tool	54
5	Cluster Extension Cloudbursting With Azure	57
5.1	Introduction To Cluster Extension Cloudbursting With Azure	57
5.2	Cluster Extension Into Azure	57
5.3	Cluster Extension Into Azure: Cloud Node Startup From Scratch	64
5.4	Cluster Extension Into Azure: <code>shutdown</code> Vs <code>power off</code>	64
5.5	Submitting Jobs With <code>cmsub</code> And Cloud Storage Nodes, For Azure Cluster Extension Clusters	65
6	Cloud Considerations And Choices With Bright Cluster Manager	67
6.1	Differences Between Cluster On Demand And Cluster Extension	67
6.2	Hardware And Software Availability	67
6.3	Reducing Running Costs	67
6.3.1	Spot Pricing	68
6.3.2	Storage Space Reduction	68
6.4	Address Resolution In Cluster Extension Networks	68
6.4.1	Resolution And <code>globalnet</code>	68
6.4.2	Resolution In And Out Of The Cloud	69
6.5	Internet Connectivity For Cloud Nodes	71
6.6	Passing Kernel Parameters To Cloud Nodes	71
7	Legacy/Current Cloud Instances, And Their Network Addressing Allocations	73
7.1	EC2-Classic And EC2-VPC	73
7.1.1	EC2-Classic Vs EC2-VPC Overview	73
7.1.2	EC2-Classic Vs EC2-VPC And AWS Account Creation Date	74
7.1.3	The Classic Cloud And The DefaultVPC Instances	74

7.1.4	The Private Cloud And Custom VPC Instances	74
7.1.5	Cloud Cluster Terminology Summary	75
7.2	Comparison Of EC2-Classic And EC2-VPC Platforms	75
7.3	Setting Up And Creating A Custom VPC	75
7.3.1	Subnets In A Custom VPC	75
7.3.2	Creating The Custom VPC	76
7.3.3	Elastic IP Addresses And Their Use In Configuring Static IP Addresses	77
7.3.4	Subnets With Static IP Addresses In A Custom VPC Recommendation	78

Preface

Welcome to the *Cloudbursting Manual* for Bright Cluster Manager 8.1.

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 8.1 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 *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.

There is also a feedback form available via Bright View, via the Account icon, , following the clickpath:

Account→Help→Feedback

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 Azure Or AWS

Cluster On Demand (COD) cloudbursting is when a separate cluster is started up in a cloud, with the cluster head node that manages the cluster also in that cloud. A COD cluster is regarded as an independent virtual cluster (sometimes described as a 'pure' cloud cluster), and not an extension of an existing physical cluster.

2.1 Requirements For COD Cloudbursting

COD can run in Azure (COD-Azure), in AWS (COD-AWS), or in Bright OpenStack (COD-OS). This chapter discusses COD-Azure and COD-AWS. COD-OS is discussed in Chapter 6 of the *OpenStack Deployment Manual*.

The requirements for COD-Azure and COD-AWS are:

- an account on the Bright Computing Customer Portal website at <http://customer.brightcomputing.com/Customer-Login>
- a Bright Cluster Manager product key. This must be registered at the Customer Portal via the Register Product Key menu item. This key is later activated when the license is installed (Chapter 4 of the *Installation Manual*) on the head node. The head node and regular nodes in this case are in the cloud.
- Requirements For COD-Azure: To use Azure, an Azure account subscription is needed from Microsoft. COD cloudbursting requires the following associated Azure credentials to launch:
 - tenant ID: This is available as the value of Directory ID, along the clickpath Azure Active Directory→Properties→Directory ID, as explained in <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal#get-tenant-id>
 - subscription ID. This is visible from the list of subscriptions along the clickpath All services→Subscriptions→SUBSCRIPTION ID
 - Client ID: This is the value of Application ID, available under the clickpath Azure Active Directory→App registrations→<app name>→Settings→Application ID.
 - Client Secret: This is available only once, when first generated, as the value at Azure Active Directory→App registrations→<app name>→Settings→Keys→VALUE

- Requirements For COD-AWS: To use AWS, an AWS subscription is needed from Amazon. COD cloudbursting requires the following associated AWS credentials for launch:
 - Secret Access Key: Available only once, when first generated, as described at <https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys>
 - Access Key ID: as described at <https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys>
 - AWS Account ID, as described at <https://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html>
 - AWS Username: This can be either the AWS account root user (the e-mail address of the root user), or it can be an IAM username with sufficient permissions to launch the cluster.

COD cloudbursting can be configured and started up via the Bright Computing customer portal (section 2.2), or via a Docker image (section 2.3).

2.2 COD Via Bright Computing Customer Portal

The customer portal (section 2.1) has a menu option `Cluster on Demand`. Selecting this displays an input form (figure 2.1).

Menu

Overview

Account

Account Overview

Edit Account

Edit Billing

Edit Password

Account Balance

Product

My Clusters

My Product Keys

Unlock Product Key

Register Product Key

Cluster on Demand

Burst Pricing

Support

Request Support

Support Tickets

Usage Reports

Show Usage

Bright Links

Release Notes

Manuals

Download ISO


Download CMGUI

Knowledge Base

Technical Training Videos

Training Portal

Burst! — Cluster on Demand



This page will allow you to create a cluster entirely in the cloud.

Cloud provider: ☐ Amazon Web Services ☐ Azure

Product key: ☒ New Product Key ☐ Use Existing Product Key

Name:

Email:

Organization:

Organizational unit:

Country:

State:

City:

Cluster name:

Linux distribution:

SSH access: ☒ Set SSH public key ☐ Set SSH password

SSH public key:

Submit

Figure 2.1: COD via customer portal: start screen

If this is filled in, then a COD can be launched in AWS or Azure.

Some of the options are:

- Cloud provider
- The product key. This can be the existing key, or a new one. Using the existing one for a cluster that is already in use is not normally possible.
- The cluster name. This can be arbitrary.

- The SSH access method.
 - SSH public key access is recommended, and should then be pasted into the text field. Using key access disables SSH password authentication.
 - Alternatively, a password can be set instead. This generates a somewhat random password for SSH password authentication that is not very strong. The password should normally be changed and saved, because brute force password attacks in the AWS and Azure IP address ranges are common.

Selecting AWS as an option presents some extra fields that are appropriate for launching an AWS COD (figure 2.2). How to obtain the values for the AWS keys is discussed in section 2.1. The default values that are in the form for the virtual machine types and sizes will work for small clusters. Other virtual machine types are documented at <https://aws.amazon.com/ec2/instance-types/>.

My Product Keys

Unlock Product Key

Register Product Key

Cluster on Demand

Burst Pricing

Support

Request Support

Support Tickets

Usage Reports

Show Usage

Bright Links

Release Notes

Manuals

Download ISO

Download CMGUI

Knowledge Base

Technical Training Videos

Training Portal

Cloud provider: ☒ Amazon Web Services ☐ Azure

AWS region:

AWS access key:

AWS secret key:

Product key: ☐ New Product Key ☒ Use Existing Product Key

Bright product key:

Name:

Email:

Organization:

Organizational unit:

Country:

State:

City:

Cluster name:

Bright version:

Linux distribution:

Workload management:

Head node flavor:

Head node disk type:

Head node disk size (GB):

[Read more about AWS disk types and sizes](#)

Compute node count:

Compute node flavor:

SSH access: ☐ Set SSH public key ☒ Set SSH password

SSH password:

Please make sure you save the password!

Submit

Figure 2.2: COD-AWS via customer portal: example inputs

Selecting Azure as an option presents some extra fields that are appropriate for launching an Azure COD (figure 2.3). How to obtain the values for the Azure credentials is discussed in section 2.1. The default values that are in the form for the virtual machine types and sizes will work for small clusters. Other virtual machine types are documented at <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/sizes>.

Menu

Overview

Account

Account Overview

Edit Account

Edit Billing

Edit Password

Account Balance

Product

My Clusters

My Product Keys

Unlock Product Key

Register Product Key

Cluster on Demand

Burst Pricing

Support

Request Support

Support Tickets

Usage Reports

Show Usage

Bright Links

Release Notes

Manuals

Download ISO

Download CMGUI

Knowledge Base

Technical Training Videos

Training Portal

Burst! — Cluster on Demand

Cloud provider: ☐ Amazon Web Services
☒ Azure

Azure region:

Azure client ID:

Azure client secret:

Azure tenant ID:

Azure subscription ID:

[How to retrieve Azure access credentials?](#)

Product key: ☒ New Product Key
☐ Use Existing Product Key

Name:

Email:

Organization:

Organizational unit:

Country:

State:

City:

Cluster name:

Bright version:

Linux distribution:

Workload management:

Head node flavor:

Head node disk size (GB):

Compute node count:

Compute node flavor:

SSH access: ☒ Set SSH public key
☐ Set SSH password

SSH public key:

Submit

Figure 2.3: COD-Azure via customer portal: example inputs

Form submission begins the COD installation (figure 2.4).

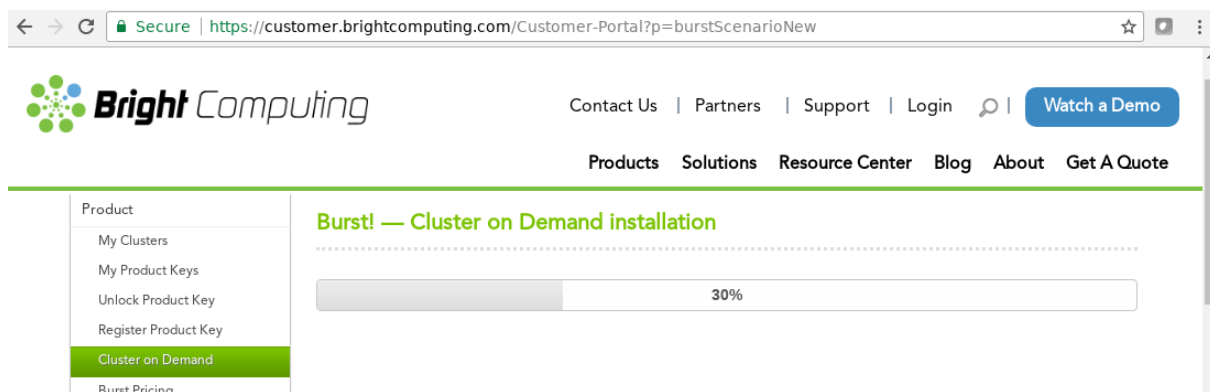


Figure 2.4: COD via customer portal: deployment in progress

When deployment is completed, an e-mail is sent with the details to the e-mail address that was entered in the form earlier. A final screen comes up if the installation was successful (figure 2.4).

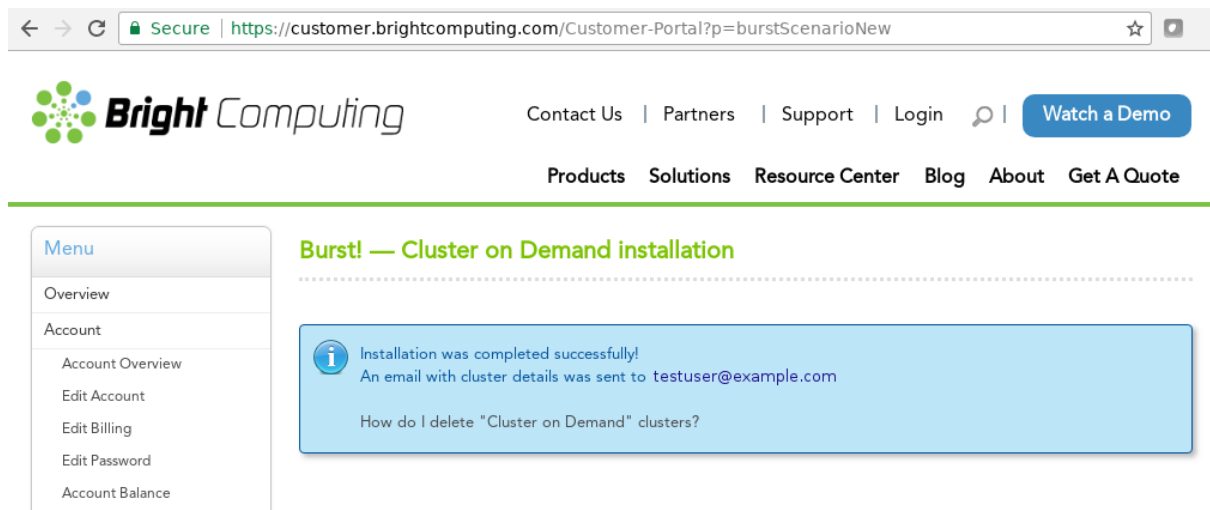


Figure 2.5: COD via customer portal: finished deploying

2.3 COD Via Docker Image

The procedure described in this section runs COD using a Docker image. The Docker instance that runs from the image can be hosted on a standalone PC that is not part of a Bright cluster, or it can be hosted on a Bright node. In this section the example uses a regular node from a Bright cluster as the Docker host.

2.3.1 COD Via Docker Image—Procedure Summary

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

- A Docker host is used to pull the Bright Cluster Manager COD (Cluster On Demand) image from the Docker registry.
- The image is run in a Docker container.
- From within the instance of the image running in the container, a cloudbursting request for a COD is carried out to a cloud service provider.

- When the request completes successfully, a cluster that is running in the cloud is ready for use.

These steps are now covered in more detail.

2.3.2 COD Via Docker Image—Procedure Details

Launching a COD from a Bright cluster can be carried out as follows:

- Docker is installed in Bright Cluster Manager with `cm-docker-setup` (section 8.1.1 of the *Administrator Manual*), or using the Bright View Docker setup wizard.
- The node that is to run the Docker image is used to pull the image to the node

Example

```
[root@bright81 ~]# ssh node001
[root@node001 ~]# module load docker
[root@node001 ~]# docker pull brightcomputing/cod:8.1
...
cac15ba059ef: Pull complete
Digest: sha256:19b263033090e1a70043989decdf3c3870d3def8c2e69b2a85ac293fd7d149ab
Status: Downloaded newer image for brightcomputing/cod:8.1
```

- The appropriate image ID can be found. For example:

Example

```
[root@node001 ~]# docker images
REPOSITORY          TAG      IMAGE ID      CREATED      SIZE
brightcomputing/cod  8.1      aeddb120e78f  5 hours ago  593 MB
```

- The image can be run in a Docker container. This is done by running the image by specifying the ID from the preceding output. The image can run interactively with, for example:

Example

```
[root@node001 ~]# docker run -it aeddb120e78f /bin/bash
[root@256ac248b583 /]#
```

- In that container an existing SSH public key should be saved, so that it can be specified as an option to the COD cluster creation command. The associated private key should be located on any machine that is to be used to access the COD via SSH.

Instead of using an existing SSH key pair, it may be convenient to generate a completely new key pair within the container, using `ssh-keygen`. In that case, the following may have to be borne in mind:

- If exiting from the container, then returning to that container is possible after restarting and attaching to the container again, so that the keys to the cluster remain accessible.
- However, if the container is to be removed, then a copy of the keys, or at least the private key, should exist outside the container, so that the cluster can still be accessed. For example the keys can be displayed with the `cat` command and then copied and pasted (some text elided):

Example

```
[root@256ac248b583 /]# cat /root/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
...
-----END RSA PRIVATE KEY-----
[root@256ac248b583 /]# cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDOXPZUImfrolTHJQT5DepyCQ...
```

It is a good idea for the administrator to plan key management, in order to minimize the spread of the private key in general, and to manage keys in a way that works well with the work flow and use.

Up to this point, the preparation for running a COD from Docker has been the same for whether Azure or AWS are to be used as the cloud provider. The next stage is to launch the COD inside the cloud service.

There are two cloud services that a COD from Docker can be launched into. These are:

- AWS (Amazon Web Services), made available by Amazon. The CLI utility in the Docker instance that launches the COD is then `cm-cod-aws` (page 15).
- Azure, made available by Microsoft. The CLI utility in the Docker instance that launches the COD is then `cm-cod-azure` (page 11).

For completeness, a non-Dockerized COD can also be launched into the Bright OpenStack private cloud service. The Bright OpenStack private cloud service is a cloud service that can be provided with the Bright OpenStack edition of Bright Cluster Manager. The launch utility for this third type of COD is `cm-cod-os`, and its use is described in detail in Chapter 6 of the *OpenStack Deployment Manual*.

Procedure When Running `cm-cod-azure`

Launching a COD inside Azure is carried by running `cm-cod-azure` from within the Docker container that is outside Azure.

Running `cm-cod-azure` from within the Docker container that is outside Azure, launches a COD within Azure.

Options For `cm-cod-azure`: The `cm-cod-azure` command options include the cluster name, node types, credentials, and so on. Running `cm-cod-azure cluster create -h` displays a list of options and explanations for the options.

Some of the options are described in table 2.1:

Option	Example Value
--azure-client-id	afe13723-c80a-68e2-9cd0-e95a657106a
--azure-client-secret	aiVohwi7OhJ6igim=
--azure-tenant-id	2c89c8dd-6b8b-5393-60f4-d876cbe188f
--azure-subscription-id	5e519b1e-aff9-e839-bc3a-c5d87e9d0d5
--azure-location	westeurope
<cluster_name>	noviceazurecluster
--node-type	Standard_D1_v2
--head-node-type	Standard_D1_v2
--version	8.1
--wlm	dont-configure
--head-node-root-volume-size	50
--nodes	3
--ssh-pub-key-path	/root/.ssh/id_rsa.pub
--license-organization	Illuminati
--license-unit	Eleusis

...continues

...continued

Option	Example Value
--license-locality	Munich
--license-state	Bavaria
--license-country	Germany
--license-product-key	12334-324234-3413413-32-324

The example values for the credentials given here, if used literally, will not work. They are just to show the rough format of the values.

Table 2.1: *cm-cod-azure options*

The first four `--azure-*` options in the preceding table require the actual Azure credentials that the cluster administrator obtains from Microsoft.

For the distinguished name values (organization, unit, and so on), the values can be arbitrary, although the administrator may find it useful to have them match the ones set in the product key license.

The `verify-license` utility shows the distinguished name values that were set in the product key license.

For example:

```
[root@bright81 ~]# verify-license /cm/local/apps/cmd/etc/cluster.pem \
/cm/local/apps/cmd/etc/cluster.key info
===== Certificate Information =====
settings for organization, unit, etc, are displayed
```

It is normally not possible to use the product key of the host cluster itself. So, normally a fresh product key must be used with the `cm-cod-azure` command.

Cluster creation run with `cm-cod-azure`: An example of a `cm-cod-azure` command that can be run is then:

Example

```
cm-cod-azure cluster create \
  --azure-location 'westeurope' \
  --azure-client-id 'afe13723-c80a-68e2-9cd0-e95a657106a' \
  --azure-client-secret 'aiVohwi7OhJ6igim=' \
  --azure-tenant-id '2c89c8dd-6b8b-5393-60f4-d876cbe188f' \
  --azure-subscription-id '5e519b1e-aff9-e839-bc3a-c5d87e9d0d5' \
  --head-node-type 'Standard_D1_v2' --head-node-root-volume-size '50' \
  --nodes '3' --node-type 'Standard_D1_v2' \
  --ssh-pub-key-path '/root/.ssh/id_rsa.pub' \
  --license-product-key '12334-324234-3413413-32-324' 'noviceazurecluster'
```

If all is well, then something similar to the following is displayed (some output elided):

```
13:22:50: INFO: Credentials are valid and have read/write authorizations
13:23:23: INFO: Cluster Create
13:23:24: INFO: -----
13:23:24: INFO: Cluster: noviceazurecluster
13:23:24: INFO: -----
13:23:24: INFO: Image: bcm-cod-image-8.1-4.vhd
13:23:24: INFO: Image date: 2018-02-27 09:01:27+00:00
13:23:24: INFO: Head nodes: 1 (Standard_D1_V2)
13:23:24: INFO: Nodes: 3 (Standard_D1_v2)
13:23:24: INFO: Region: westeurope
```

```

13:23:24:      INFO:   Key path:   /root/.ssh/id_rsa.pub
13:23:24:      INFO: -----
Press ENTER to continue and create the cluster.
Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

13:23:56:      INFO: ## Progress: 2
13:23:56:      INFO: #### stage: Creating resource group noviceazurecluster_cod_resource_group
13:23:58:      INFO: Creating storage account noviceazureclusterstorageaccount
13:24:22:      INFO: ## Progress: 5
13:24:22:      INFO: #### stage: Building deployment template
13:24:22:      INFO: generating cloud-init script
13:24:22:      INFO: ## Progress: 7
13:24:22:      INFO: #### stage: Copying head node image
13:24:22:      INFO: Copying https://brightimages.blob.core.windows.net/images/bcm-cod-image-\
8.1-4.vhd to noviceazureclusterstorageaccount/images/noviceazurecluster-head-node-os-disk.vhd
13:24:22:      INFO: ## Progress: 12.0
13:24:22:      INFO: #### stage: Server side copy
13:25:01:      INFO: ## Progress: 12.2
13:25:01:      INFO: #### stage: Server side copy
13:25:05:      INFO: ## Progress: 12.4
13:25:05:      INFO: #### stage: Server side copy
13:25:09:      INFO: ## Progress: 12.6
...
13:35:32:      INFO: #### stage: Server side copy
13:35:35:      INFO: ## Progress: 61.77
13:35:35:      INFO: #### stage: Server side copy
13:35:38:      INFO: ## Progress: 61.97
13:35:38:      INFO: #### stage: Server side copy
13:35:40:      INFO: Elapsed: 11:18 min
13:35:40:      INFO: ## Progress: 85
13:35:40:      INFO: #### stage: Creating and deploying Head node
13:39:53:      INFO: Waiting for CMDaemon to come up
...
13:41:34:      INFO: Waiting for CMDaemon to come up
13:41:34:      INFO: CMDaemon is not up yet, waiting 10 seconds...
13:41:34:      INFO: ## Progress: 100
13:41:34:      INFO: #### stage: Deployment finished successfully.
13:41:34:      INFO: -----
13:41:34:      INFO:      Cluster:      noviceazurecluster
13:41:34:      INFO: -----
13:41:34:      INFO:      Image:      bcm-cod-image-8.1-4.vhd
13:41:34:      INFO:      Image date: 2018-02-27 09:01:27+00:00
13:41:34:      INFO:      Head nodes: 1 (Standard_D1_V2)
13:41:34:      INFO:      Nodes:      3 (Standard_D1_v2)
13:41:34:      INFO:      Region:     westeurope
13:41:34:      INFO:      Key path:   /root/.ssh/id_rsa.pub
13:41:34:      INFO:      Head node ID: 3e6d2b24-2c7b-40cb-a594-fb3e66f31319
13:41:34:      INFO:      Public IP:  13.93.90.5
13:41:34:      INFO: -----
[root@743d346dca20 /]#

```

The cluster can now be logged into by using the SSH keys generated earlier, and using the public IP address shown in the last few lines of the preceding output.

Listing with `cm-cod-azure`: The clusters that are running from the container can be listed with their properties, using the credentials, as follows:

Example

```
[root@743d346dca20 /]# cm-cod-azure cluster list \
--azure-location 'westeurope' \
--azure-client-id 'afe13723-c80a-68e2-9cd0-e95a657106a' \
--azure-client-secret 'aiVohwi7OhJ6igim=' \
--azure-tenant-id '2c89c8dd-6b8b-5393-60f4-d876cbe188f' \
--azure-subscription-id '5e519b1e-aff9-e839-bc3a-c5d87e9d0d5'
```

```
+-----+-----+-----+-----+...
| Cluster Name      | Head node name      | Public IP | Location  | ...
+-----+-----+-----+-----+...
| noviceazurecluster | noviceazurecluster-head_node | 13.98.90.5 | westeurope | ...
+-----+-----+-----+-----+...
```

Removal with `cm-cod-azure` A cluster can be removed from the container with:

```
[root@743d346dca20 /]# cm-cod-azure cluster delete \
--azure-location 'westeurope' \
--azure-client-id 'afe13723-c80a-68e2-9cd0-e95a657106a' \
--azure-client-secret 'aiVohwi7OhJ6igim=' \
--azure-tenant-id '2c89c8dd-6b8b-5393-60f4-d876cbe188f' \
--azure-subscription-id '5e519b1e-aff9-e839-bc3a-c5d87e9d0d5' \
'noviceazurecluster'
```

Procedure When Running `cm-cod-aws`

Launching a COD from within AWS is carried by running `cm-cod-aws` from within the Docker container. This process is similar to that of launching within Azure (page 15).

Options For `cm-cod-aws`: The `cm-cod-aws` command takes options to create a cluster. Running `cm-cod-aws cluster create -h` displays a list of options and explanations for the options.

Some of the options are described in table 2.2:

Table 2.2: `cm-cod-aws` options

Option	Example Value
<code>--aws-region</code>	eu-central-1
<code>--aws-secret-key</code>	Oocej6eiieshahG7IiJe9Quoud0oo
<code>--aws-access-key-id</code>	AKIAJAICHAZI7OHH1EEGO
<code>--head-node-type</code>	t2.medium
<code>--node-type</code>	t2.medium
<code>--license-product-key</code>	12334-324234-3413413-32-324
<code>--cluster-password</code>	justinbieberrocks
<code>--log-cluster-password</code>	<i>does not take a value</i>
<code>--ssh-pub-key-path</code>	/root/.ssh/id_rsa.pub

...continues

...continued

Option	Example Value
--nodes	3
<cluster_name> [...]	noobawscluster

The example values for the credentials given here, if used literally, will not work. They are just to show the rough format of the values.

The --aws-secret-key and --aws-access-key-id options in the preceding table require the actual AWS credentials that the cluster administrator obtains from Amazon.

Setting a root password via the --cluster-password option is a possible way to configure a login to the cluster. Another possibility is to use ssh keys. The keys generated within the docker instance using ssh-keygen earlier on can be used for this.

The password can be seen in the logs of the command if the --log-cluster-password option is used.

If the root password is not set with --cluster-password, then a random password is set, and ssh keys can be used to login to the cluster. A root password can always be specified later using cm-change-password, after logging into the cluster via SSH key access or the root password. The default login to Bright View also uses the root password.

For the product key of cluster in AWS, it is normally not possible to use the product key of the host cluster itself. So, normally a fresh product key must be used with the cm-cod-azure command.

Cluster creation run with cm-cod-aws: An example of a cm-cod-aws command that can be run is then:

Example

```
cm-cod-aws cluster create \
  --aws-region 'eu-central' \
  --aws-secret-key 'Oocei6eiieshahG7IiJe9Quoud0oo' \
  --aws-access-key-id 'AKIAJAICHAZI7OHH1EEGO' \
  --nodes '3' --node-type 't2.medium' \
  --ssh-pub-key-path '/root/.ssh/id_rsa.pub' \
  --license-product-key '12334-324234-3413413-32-324' 'noobawscluster'
```

If all is well, then the output is similar to (some output elided):

```
12:06:57: INFO: -----
12:06:57: INFO: Cluster: noobawscluster
12:06:57: INFO: -----
12:06:57: INFO: Image: brightheadnode-8.1-centos7u4-hvm-3 (ami-71691308)
12:06:57: INFO: Image date: 2018-02-26 16:10 (27d 19h ago)
12:06:57: INFO: Head nodes: 1 (t2.medium)
12:06:57: INFO: Nodes: 5 (t2.medium)
12:06:57: INFO: Region: eu-west-1
12:06:57: INFO: ssh pub key: /root/.ssh/id_rsa.pub
12:06:57: INFO: -----
Press ENTER to continue and create the cluster.
Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

12:07:01: INFO: ## Progress: 2
12:07:01: INFO: #### stage: Setting up VPC
12:07:01: INFO: ## Progress: 5
12:07:01: INFO: #### stage: Creating VPC for noobawscluster in eu-west-1...
12:07:02: INFO: ## Progress: 10
```



```

12:07:02:      INFO: #### stage: Adding internet connectivity...
12:07:03:      INFO: ## Progress: 12
12:07:03:      INFO: #### stage: Creating public subnet...
12:07:04:      INFO: ## Progress: 16
12:07:04:      INFO: #### stage: Creating head node (t2.medium)...
12:07:04:      INFO: ## Progress: 20
12:07:04:      INFO: #### stage: Public key specified, no need to enable password authentication
12:07:04:      INFO: Scheduling cm-bright-setup.
12:07:06:      INFO: ## Progress: 22
12:07:06:      INFO: #### stage: Done setting up VPC for noobawscluster.
12:07:06:      INFO: ## Progress: 22
12:07:06:      INFO: #### stage: Starting head nodes...
12:07:06:      INFO: ## Progress: 30
12:07:06:      INFO: #### stage: Waiting for head nodes to start...
12:07:38:      INFO: ## Progress: 30
12:07:38:      INFO: #### stage: Wait for cloud-init to finish on cluster noobawscluster...
12:07:43:      INFO: ## Progress: 30
12:07:43:      INFO: #### stage: Cloud-init is not finished, waiting 10 seconds...
12:07:53:      INFO: ## Progress: 30
...
12:10:34:      INFO: #### stage: Cloud-init is not finished, waiting 10 seconds...
12:10:44:      INFO: ## Progress: 30
12:10:44:      INFO: #### stage: Wait for cloud-init to finish on cluster noobawscluster...
12:10:44:      INFO: ## Progress: 100
12:10:44:      INFO: #### stage: Deployment of cluster: noobawscluster finished successfully.
12:10:54:      INFO: Script completed.
12:10:54:      INFO: -----
12:10:54:      INFO: Time it took:    03:52
12:10:54:      INFO:   SSH string:    ssh root@52.30.93.49
52.30.93.49 noobawscluster
12:10:54:      INFO: Head node ID:    i-0120058780aa736eb
[root@256ac248b583 /]# ssh root@52.30.93.49

```

At this point, the head node running on AWS is ready for use. It can be accessed via ssh, as suggested in the last line of the preceding output.

The regular nodes running on AWS are configured for use, but are not powered on. They can be powered on using the `cmsh` or Bright View front ends, just as in a regular cluster.

From `cmsh`, the first cloud node can be powered on with, for example:

```
[noobawscluster->device]% power on cnode001
```

Listing with `cm-cod-aws`: The clusters that are running from the container can be listed with their properties, using the credentials with the region, as follows:

Example

```

[root@743d346dca20 /]# cm-cod-aws cluster list \
--aws-region 'eu-west-1' \
--aws-secret-key 'Oocei6eiieshahG7IiJe9Quoud0oo' \
--aws-access-key-id 'AKIAJAICHAZI7OHH1EEGO' \
--license-product-key '12334-324234-3413413-32-324'

+-----+-----+-----+
| Cluster Name      | Head node name      | VPC ID      | ...
+-----+-----+-----+
| noobawscluster    | noobawscluster-head_node | vpc-as89ada | ...
+-----+-----+-----+

```

Removal with `cm-cod-aws` A cluster can be removed from the container with:

```
[root@743d346dca20 /]# cm-cod-aws cluster delete \
--aws-region 'eu-west-1' \
--aws-secret-key 'OoCeI6eiieshaH7IiJe9Quoud0oo' \
--aws-access-key-id 'AKIAJAICHAZI7OHH1EEGO' \
--license-product-key '12334-324234-3413413-32-324' \
noobawscluster
```

2.4 Using The AWS EC2 Management Console

The recommended way of managing COD is using Bright View via its public IP address, or by using `cmsh` via the head node (section 2.4.5). This section (section 2.4) describes how the Amazon management console can also be used, to manage some AWS aspects of the instance.

A login is possible from <https://console.aws.amazon.com/console/> by using the e-mail address and password of the associated AWS root account, or AWS IAM user account.

The head node instance can be watched and managed without Bright Cluster Manager in the following ways.

2.4.1 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 2.6).

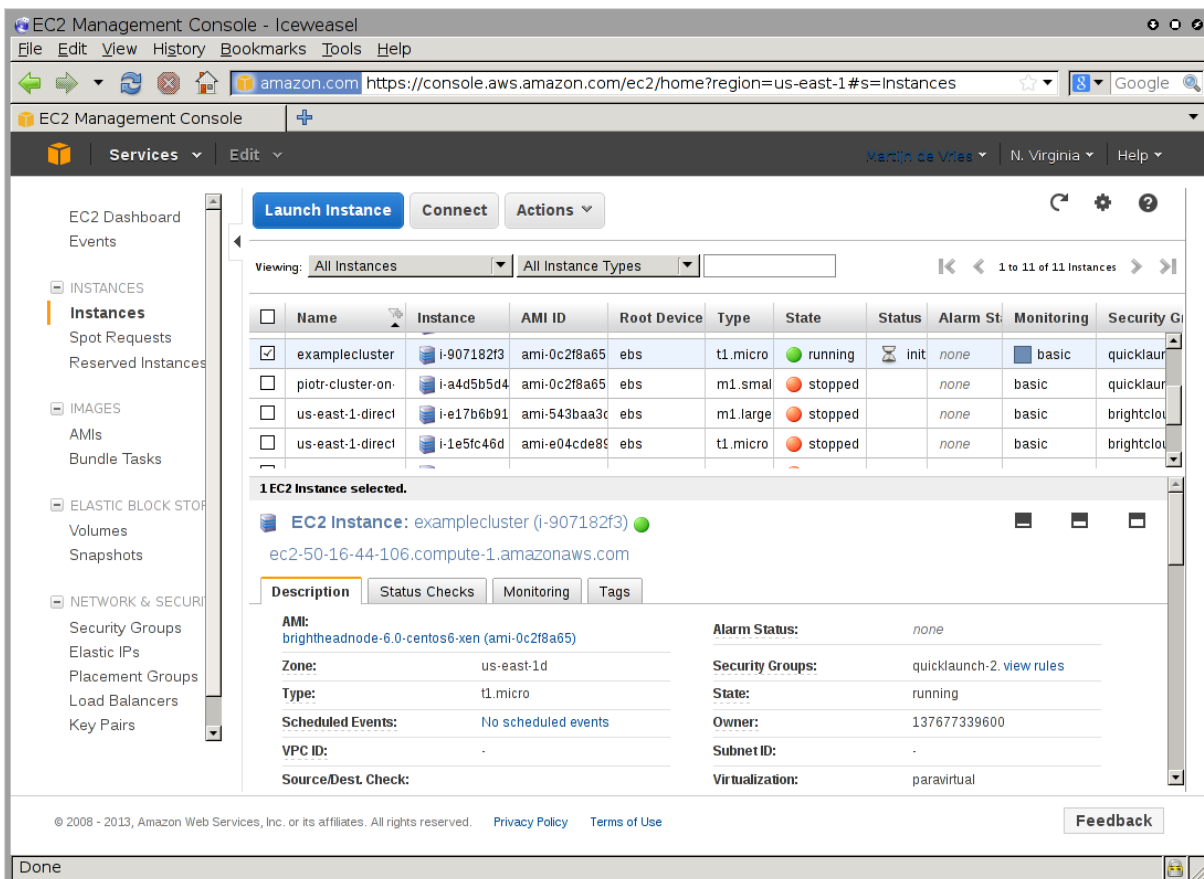


Figure 2.6: The EC2 Instances List

System (Amazon machine infrastructure) and instance (instance running under the infrastructure)

reachabilities are similarly shown under the neighboring “Status Checks” tab (figure 2.7).

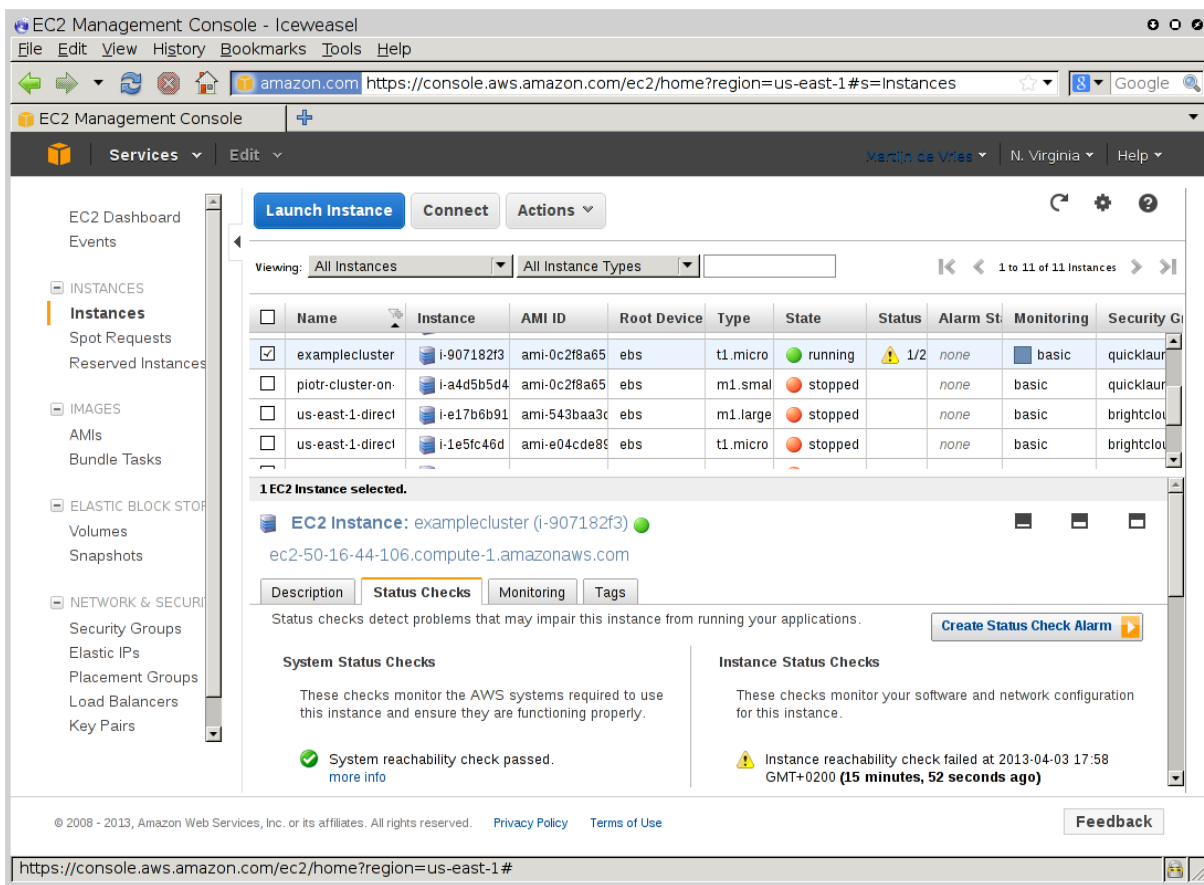


Figure 2.7: Reachability Status For An EC2 Instance

2.4.2 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.

2.4.3 Connecting To An Instance From The AWS EC2 Management Console

A marked and running instance can have an SSH connection made to it.

As in the Azure case, for most users this means running `ssh` to the public IP address as suggested at the end of the AWS COD cluster creation run (page 16). This is assuming the private ssh key that was generated by the `ssh-keygen` command in the container is used for the ssh connection, which may mean that copying it out of the container is needed..

2.4.4 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 2.8). A successful start of the system generates a log with a tail similar to that of figure 2.8.

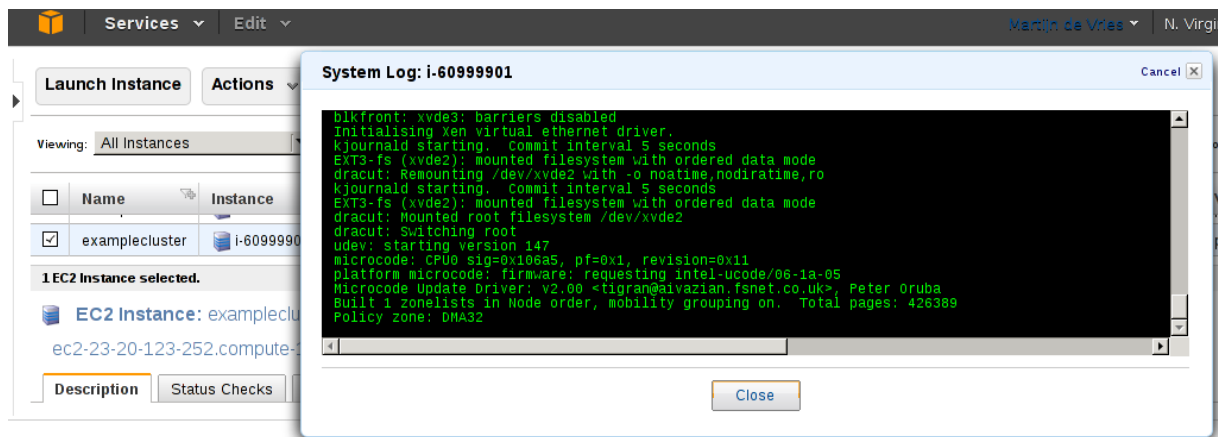


Figure 2.8: System Log Of The Checkboxed Instance

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

2.4.5 Security Group Configuration To Allow Access 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 2.9. Security groups can also be accessed from the navigation menu on the left side of the EC2 Management Console.

COD 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 2.9).

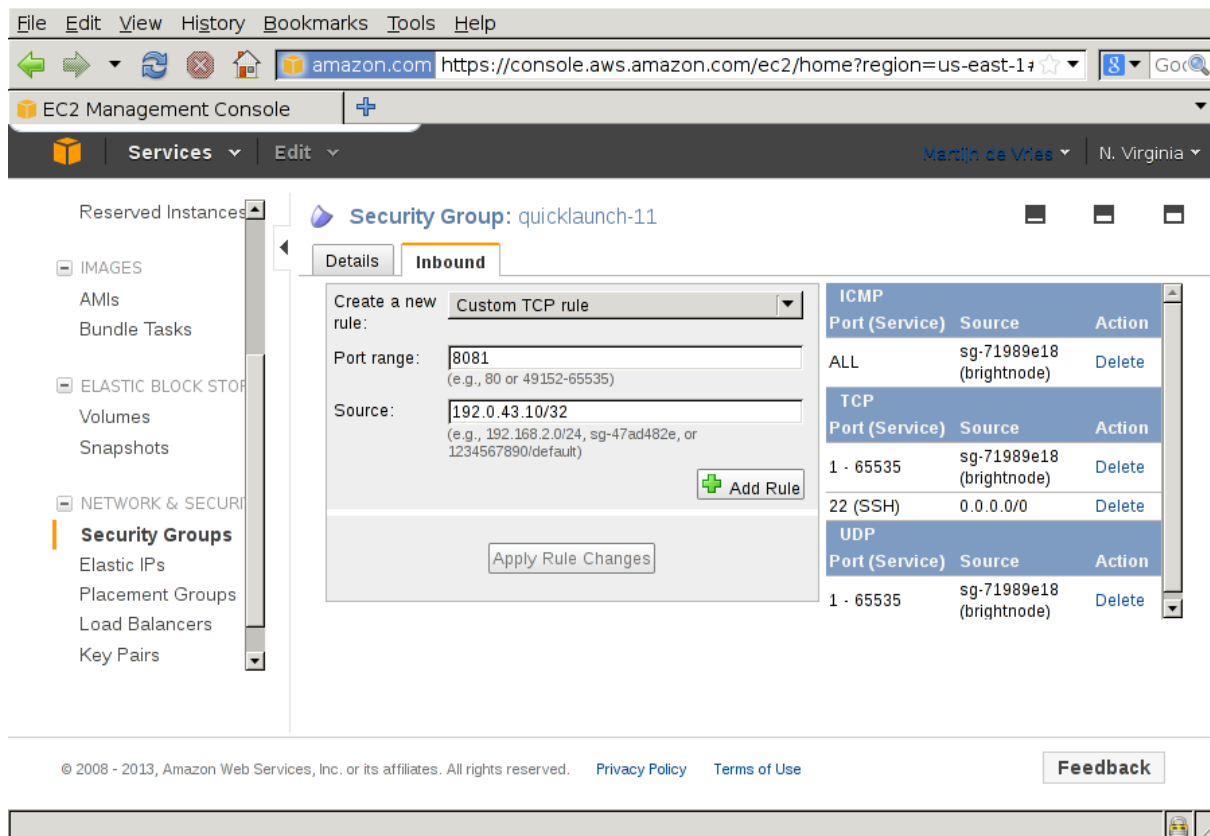


Figure 2.9: Security Group Network And Port Access Restriction

- 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.
- By default, port 8081 is open on the head node to allow 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.

COD 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 should work without lag.

2.5 Using The Azure Dashboard

For Azure, as is the case for AWS, the cluster is normally expected to be managed via Bright View or `cmsh`. Sometimes when carrying out some special configurations, there may be a need to manage the cluster objects directly via the portal at <https://portal.azure.com>. This requires an Azure login and password, which opens up the Azure dashboard, and allows the objects to be viewed and managed.

For example, the virtual machines of the COD can be viewed and managed via the `Virtual machines` resource (figure 2.10), and the virtual networks associated with the COD nodes can similarly be viewed and managed via `Virtual networks` resource.

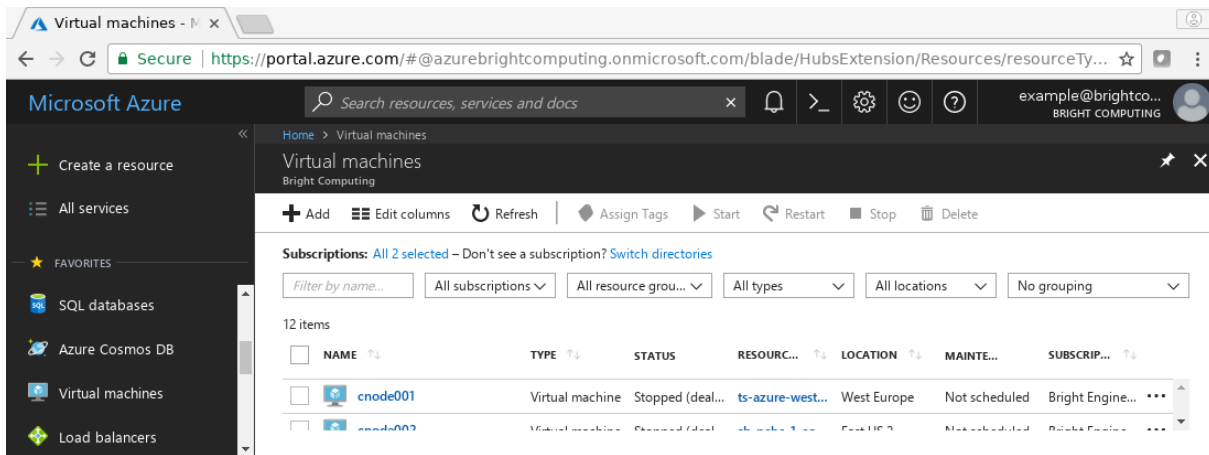


Figure 2.10: Viewing Virtual Machines In A Cluster On Demand In Azure

Some items that may be useful when accessing the portal:

- An overview of the COD head node is possible via the clickpath:
Home→Virtual machines→<head node>→Overview
- Boot process diagnostics can be viewed using the clickpath:
Home→Virtual machines→<head node>→Boot diagnostics→Serial log
- A serial console can be accessed the clickpath:
Home→Virtual machines→<head node>→Serial console
The file `/etc/ssh/sshd_config` should have the parameter `ChallengeResponseAuthentication` set to `yes`, and the service restarted, for serial console login to work.

2.6 COD: 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-cod-aws` or `cm-cod-azure` script which is run in the Docker instance. In the preceding example the cloud node objects are `cnode001` and `cnode002`.

However, more cloud node objects can be created if needed after the scripts have 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 Bright View they can conveniently be cloned via the clickpath:

Cloud→Cloud Nodes→<cloud node>↓Clone→<number>

- In `cmsh` a large number of cloud node objects can conveniently be created with the “`foreach --clone`” command instead, as described in section 4.2.

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*).

2.6.1 COD: IP Addresses In The 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.

2.7 COD With AWS: Optimizing AWS For High Performance Computing (HPC)

Optimization of cloud nodes for HPC in AWS is discussed in section 3.5.

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.

Cluster Extension cloudbursting can burst into a cloud that is running within:

- AWS (CX-AWS)
- Azure (CX-Azure)
- OpenStack (CX-OS)

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@bright81 ~]# cmsh -c "main; licenseinfo"
```

```
License Information
```

```
-----
Licensee                /C=US/ST=NY/L=WS/O=Bright
                        Mordor/OU=Mt. Doom/CN=Bright 8.1 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 629 of the *Administrator Manual*), which is a `CMDaemon` directive that can be set and activated (page 615 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: The Bright View Cluster Extension Wizard

Cluster Extension with the Bright View cluster extension wizard is described in sections 3.1, 3.2, and 3.3. Cluster Extension with Azure is described in Chapter 5.

The Amazon cloud service can be configured for Cluster Extension from Bright View. This can be done via the URL `https://<head node address>:8081/bright-view/`, and then selecting the click-path Cloud→AWS→AWS Wizard. A screen introducing the AWS Wizard is then displayed.

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.

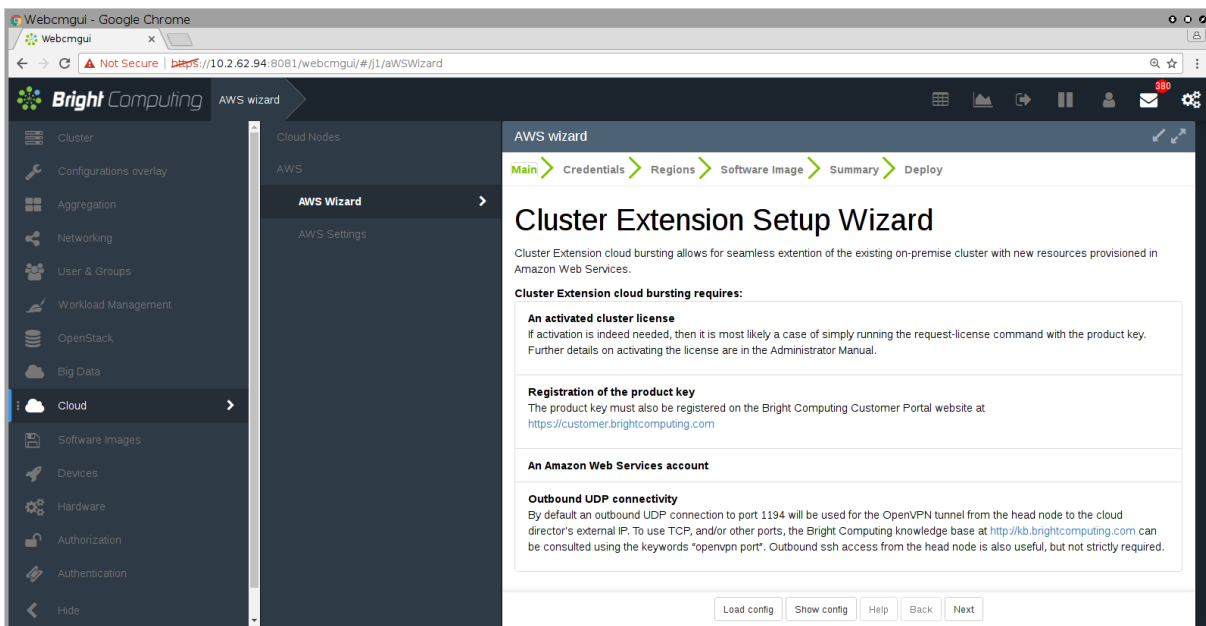


Figure 3.1: Introduction Page For Cluster Extension With Bright View

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).

AWS wizard

Main > Credentials > Regions > Software Image > Summary > Deploy

AWS Credentials

Provide your AWS credentials.

'Access Key ID' and 'Secret Key' can be verified by clicking on 'Validate' button.

Provider Name:

AWS Access key ID:

AWS Secret key:

Validate

Show config Help Back Next

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

This asks for:

- **Provider Name:** This 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.

AWS wizard

Main > Credentials > **Regions** > Software Image > Summary > Deploy

Select Regions

Please select at least one of the regions below. For each selected region, a VPC will be created in that region during the deployment process.

Cluster Extension cloud bursting requires:

- ☐ us-east-1
- ☐ us-east-2
- ☐ us-west-1
- ☐ us-west-2
- ☒ eu-west-1
- ☐ eu-west-2

Show config Help Back Next

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.

AWS wizard

Main > Credentials > Regions > Software Image > Summary > Deploy

Select Software Images

Please select at least one software image to be provisioned to the cloud director.

☒ /cm/images/default-image

Default cloud director type: m3.medium

Please select the software image for cloud nodes. /cm/images/default-image

Default cloud node type: m3.medium

Show config Help Back Next

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

3.1.5 Summary & Deployment

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

AWS wizard

Main > Credentials > Regions > Software Image > Summary > Deploy

Summary

Provider:
amazon

Default Region:
eu-west-1

Default cloud director type:
m3.medium

Default cloud node type:
m3.medium

VPC regions:
eu-west-1

Automatically deploying the configuration will take several minutes, during which a log window will be shown displaying the progress of the deployment.

☒ Ready for deployment

- Press 'Deploy' to start deployment.

Show config Help Back Deploy

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).

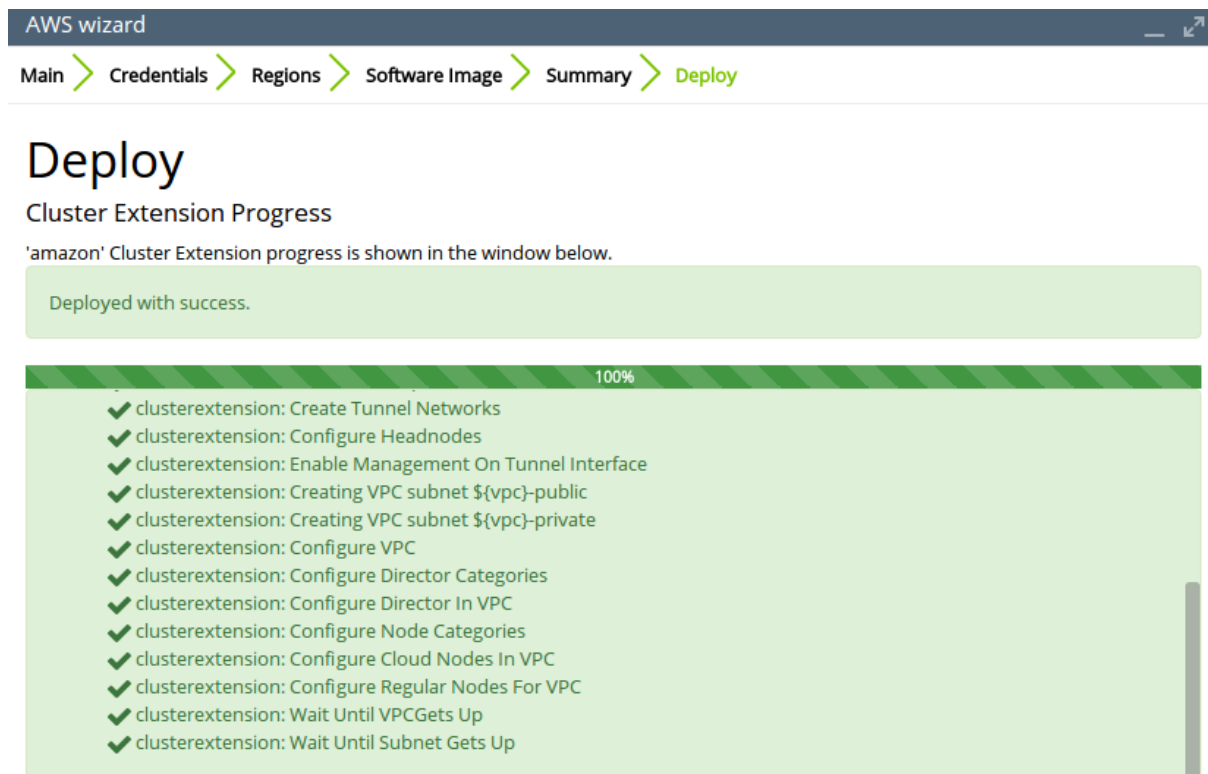


Figure 3.6: Cluster Extension Configuration Processing With Bright View

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. If the cloud director node is not visible, then a browser refresh should clear up the cache so that it becomes visible. 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

```
[bright81]% device use us-east-1-director
[bright81->device[us-east-1-director]]% cloudsettings
[bright81->device[us-east-1-director]->cloudsettings]% storage
[bright81->...->cloudsettings->storage]% list
Type      Name (key)   Drive   Size   Volume ID
-----
ebs        ebs          sdb     42GB
ephemeral  ephemeral0   sdc     0B     ephemeral0
[bright81->...->cloudsettings->storage]% remove ebs
[bright81->...->cloudsettings*->storage*]% set ephemeral0 drive sdb
[bright81->...->cloudsettings*->storage*]% list
Type      Name (key)   Drive   Size   Volume ID
-----
ephemeral  ephemeral0   sdb     0B     ephemeral0
[bright81->...->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 2.7. 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

```
[bright81]% device use us-east-1-director
[bright81->device[us-east-1-director]]% get cloudid
i-f98e7441
[bright81->device[us-east-1-director]]% cloudsettings
[bright81->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@bright81 ~]# 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@bright81 ~]# 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@bright81 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):      4
Source node:        bright81
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@bright81 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):      5
Source node:        bright81
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 137 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 4.2.

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).

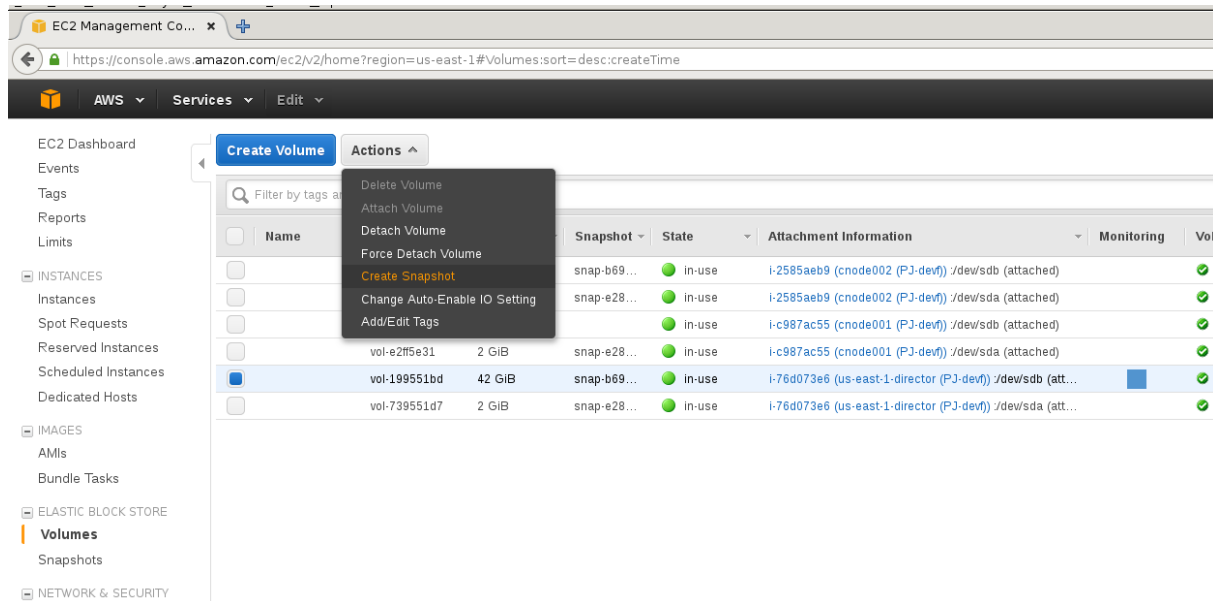


Figure 3.7: Creating A Snapshot From A Selected Volume

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

```
[bright81->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@bright81 ~]# cmsh
[bright81]% device
[bright81->device]% use us-east-1-director
[bright81->device[us-east-1-director]]% cloudsettings
[bright81->...-director]->cloudsettings]% storage
[bright81->...-director]->cloudsettings->storage]% use ebs
[bright81->...-director]->cloudsettings->storage[ebs]]% set snapshotid snap-2a96d0c6
```

The cloud director can now be powered on:

Example

```
[bright81->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

```
[bright81->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.

3.5 Cluster Extension With AWS: Optimizing AWS For High Performance Computing (HPC)

For HPC, performance is a central concern. Optimization for the jobs that are run can be carried out by the administrator considering what are the most cost-effective aspects of the system that can be improved. The considerations in this section for AWS, apply to COD instances as well as to Cluster Extension instances.

3.5.1 Optimizing HPC Performance: EBS Volume Type

The volume type used for EBS storage can be considered. AWS provides the gp2, io1, sc1, st1, and standard storage volume types. These volume types, and their associated IOPS performances, are described at <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html>.

- In `cmsh` these can be set in the storage mode:

Example

```
[bright81->device[cnode001]->cloudsettings->storage[ebs]]% set volumetype io1
```

- In Bright View, the equivalent clickpath is:

```
Cloud→Cloud Nodes→Settings→Cloud settings→Storage→EBS→Volume type
```

3.5.2 Optimizing HPC Performance: Placement Groups

The locality of the cloud nodes with respect to each other and with respect to the cluster that they extend from can be considered.

For cloud nodes in the same placement group, lag times are minimized between the nodes. For cloud nodes that are geographically near cluster that they extend from, the lag times are reduced between the cloud nodes and the cluster they extend from.

Localizing HPC cloud nodes within the same placement group is usually desirable. This can be achieved using the AWS web console to access the cluster extension AWS account. A placement group can be created for a region via the AWS web console for the instance.

The clickpath to carry this out is `Services→EC2→Services→Placement Groups→Create Placement Group`. A name should be set. The value of `Strategy` should be set to `Cluster`, to localize the nodes.

The same placement group name can then be set via `cmsh` for the not-yet-instantiated cloud nodes. Setting this means that those nodes are now started in that placement group.

Example

```
[bright81->device[cnode001]->cloudsettings]% set placementgroup indahood
```

3.5.3 Optimizing HPC Performance: Disabling Hyper-Threading

Hyper-Threading (HT) in many cases hinders HPC performance. When running cluster extension cloud nodes, HT can be left enabled on some instances and disabled on others, depending on the need.

To disable HT, the following can be inserted into the `/etc/rc.local` file for the cloud node image, and made executable with a `chmod +x`:

```
for cpunum in $(cat /sys/devices/system/cpu/cpu*/topology/thread_siblings_list | \
  cut -s -d\ - -f2- | tr ',' '\n' | sort -un)
do echo 0 > /sys/devices/system/cpu/cpu$cpunum/online; done;
```

The delimiter used in the `cut` command is `"-"` instead of a `","`.

If HT is disabled on the cloud node, then it can be confirmed by checking the output of the `lscpu --extended` command (output truncated):

Example

```
[root@cnode001 ~]# lscpu --extended
CPU NODE SOCKET CORE L1d:L1i:L2:L3 ONLINE
0 0 0 0 0:0:0:0 yes
1 - - - ::: no
...
```

In the line just before the `"..."`, the `"no"` indicates that that logical CPU (CPU1 in this case) has been disabled.

3.5.4 Optimizing HPC Performance: Using Elastic Network Adapter Instances

Enhanced networking that is needed for some instance types is possible if the AMI used has the Elastic Network Adapter (ENA) support attribute set. By default, all AMIs provided by Bright Cluster Manager since release 8.1-9 have ENA, as provided by the `ena` kernel module. If this kernel module is absent, the administrator should update the AMI.

3.5.5 Optimizing HPC Performance: Using A Different Clock Source

Xen is the default clocksource on AWS. Occasionally, some applications can benefit from using the TSC clocksource. Because of this, the clock source is generally changed as a best practice. The clock source can be changed with the following command:

Example

```
[root@cnode001 ~]# echo "tsc" > /sys/devices/system/clocksource/clocksource0/current_clocksource
```

3.5.6 Optimizing HPC Performance: Setting The Socket Buffer Sizes And TCP/IP Parameters In The Software Image

The socket and TCP/IP window values can be modified by inserting the following values into the `sysctl.conf` file in the image:

Example

```
[root@bright81 ~]# cat >> /cm/images/default-image/etc/sysctl.conf << EOF
net.core.netdev_max_backlog = 1000000

net.core.rmem_default = 124928
net.core.rmem_max     = 67108864
net.core.wmem_default = 124928
net.core.wmem_max     = 67108864

net.ipv4.tcp_keepalive_time = 1800
net.ipv4.tcp_mem            = 12184608 16246144 24369216
net.ipv4.tcp_rmem          = 4194304 8388608 67108864
net.ipv4.tcp_syn_retries   = 5
net.ipv4.tcp_wmem          = 4194304 8388608 67108864
EOF
```


4

Cluster Extension Cloudbursting With AWS Using The Command Line And `cmsh`

The command line and `cmsh` can be used to set up Cluster On Demand clusters for AWS and Azure, as discussed in Chapter 2. The command line and `cmsh` can also be used to set up Cluster Extension clusters, as are discussed in this chapter for AWS, and in Chapter 5 for Azure.

4.1 The `cm-cluster-extension` Script For Cluster Extension Clusters

4.1.1 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 4.1.2 and 4.2).

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@bright81 ~]# 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 it up beforehand.
cm-cluster-extension --test-everything
```

This tool looks for the following environment variables, and uses them if found:

```
AWS_USERNAME, AWS_ACCOUNT_ID, AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_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@bright81 ~]# 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).

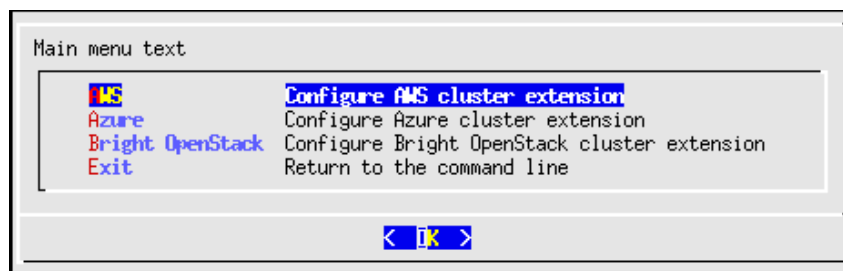


Figure 4.1: Configuration Processing With cm-cluster-extension: Main Screen

A cloud provider—Azure, AWS, or Bright OpenStack—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):

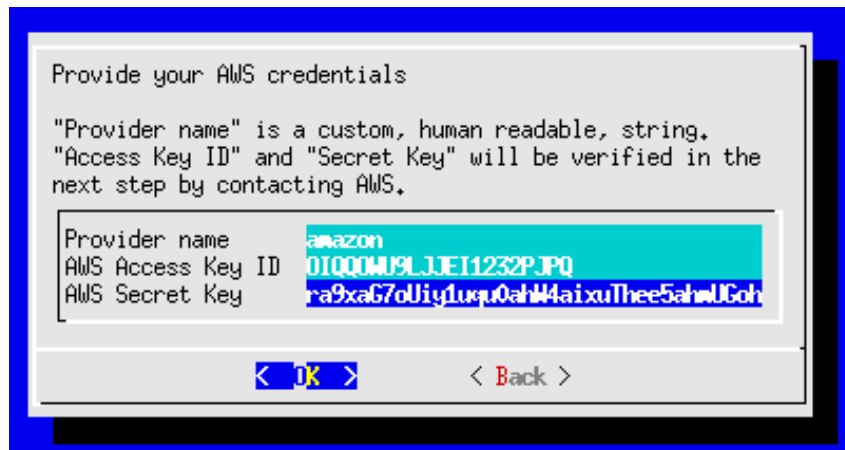


Figure 4.2: Configuration Processing With `cm-cluster-extension`: Obtaining Credentials

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.

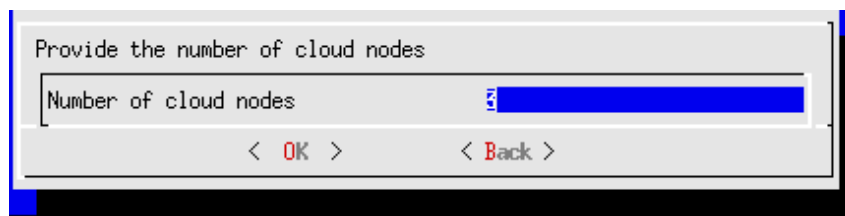


Figure 4.3: Configuration Processing With `cm-cluster-extension`: Setting The Initial Number Of Cloud Nodes

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

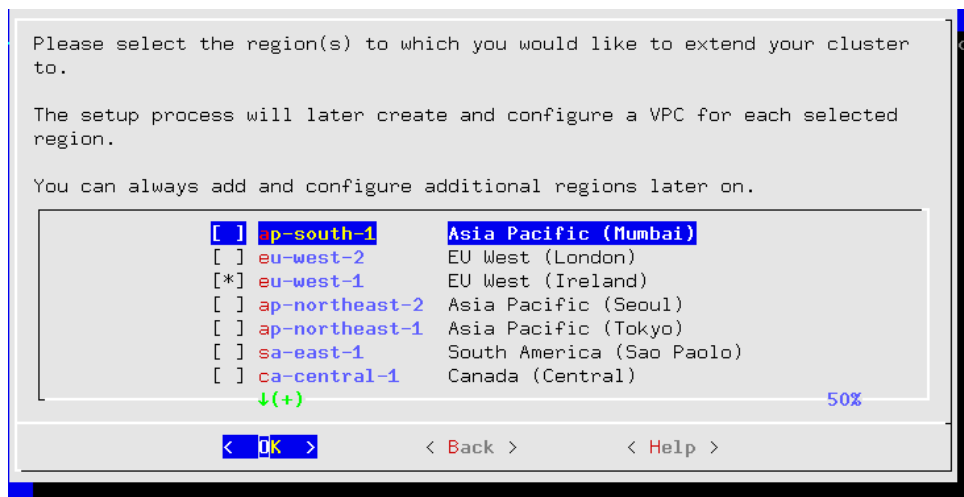


Figure 4.4: Configuration Processing With `cm-cluster-extension`: Regions Selection

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):

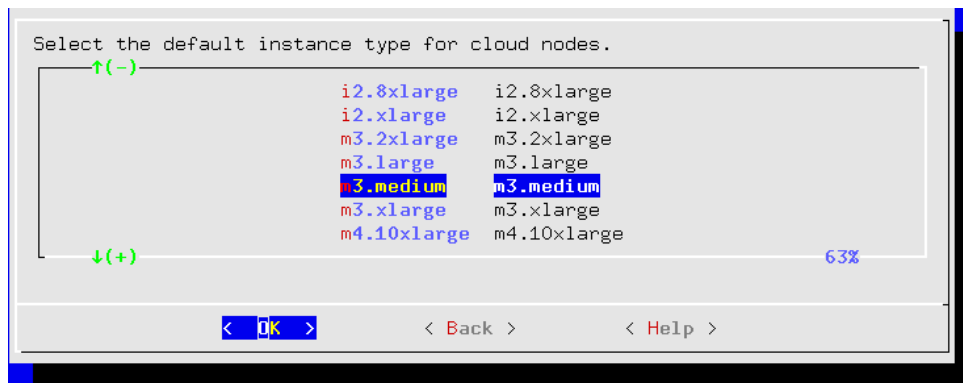


Figure 4.5: Configuration Processing With `cm-cluster-extension`: Default Instance Type

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:

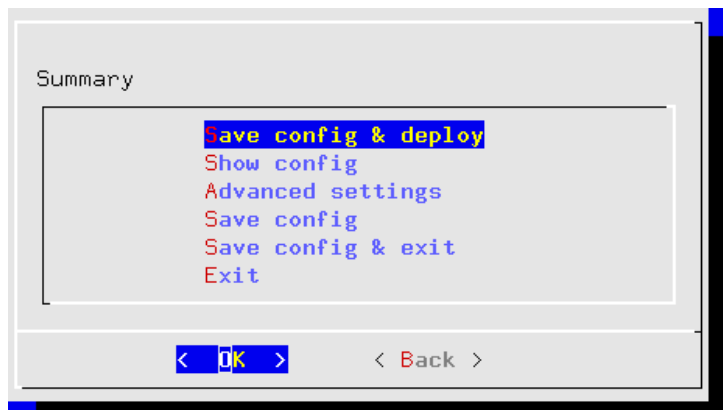


Figure 4.6: Configuration Processing With `cm-cluster-extension`: Summary Screen

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.

4.1.2 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@bright81 ~]# 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 ]
```

4.2 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:

```
[bright81->device[cnode001]->interfaces]% device power on -n cnode001
```


4.2.1 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

```
[bright81->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
...
[bright81->device*]% commit
Successfully committed 9 Devices
[bright81->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 be carried out using `cmsh` with the node range option as follows:

Example

```
[bright81->device]% power on -n eu-west-1-cnode0[02-10]
```

4.3 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@bright81 ~]# cmsh
[bright81]% user use henry
[bright81->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.

4.3.1 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@bright81 ~]# yum install cmdaemon-cmsub
[...]
```

```
[root@bright81 ~]# 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@bright81 ~]# 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):

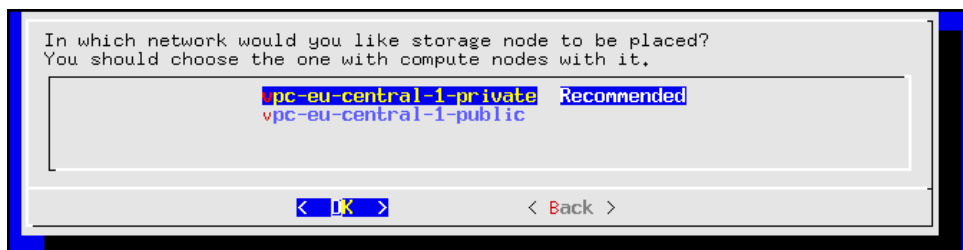


Figure 4.8: Cloud Storage Configuration Processing With `cm-cloud-storage`: Network Selection

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

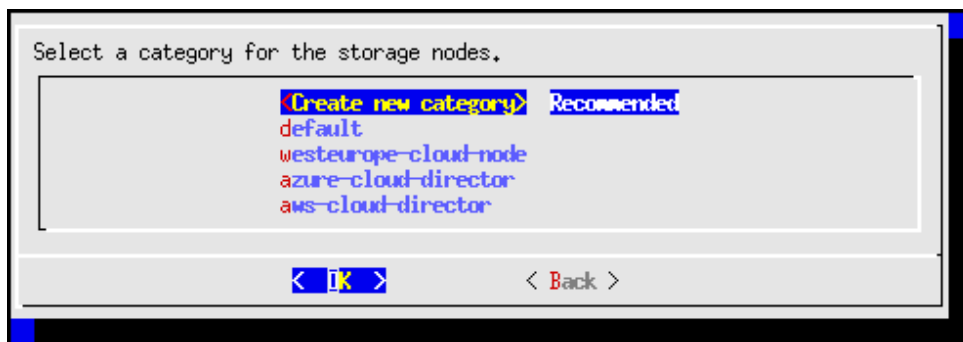
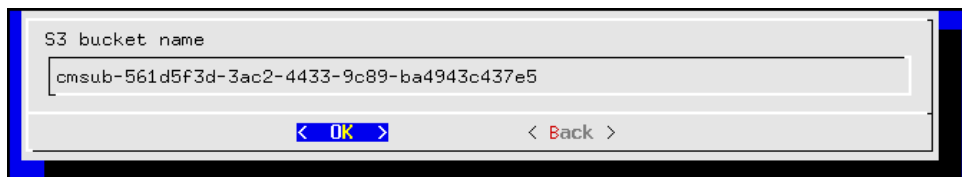


Figure 4.9: Cloud Storage Configuration Processing With `cm-cloud-storage`: Category Selection

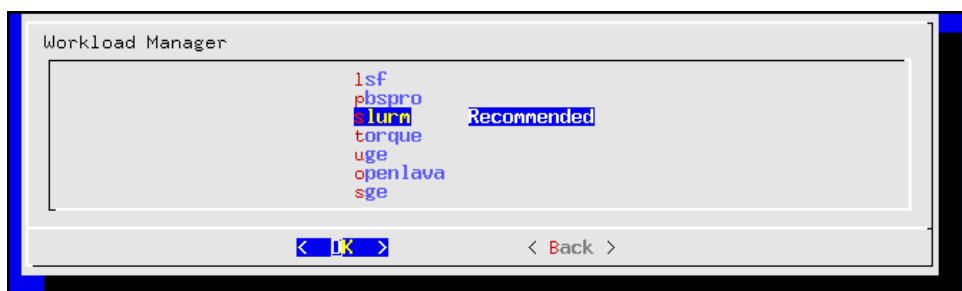
It is recommended that a new category be created (figure 4.10):

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

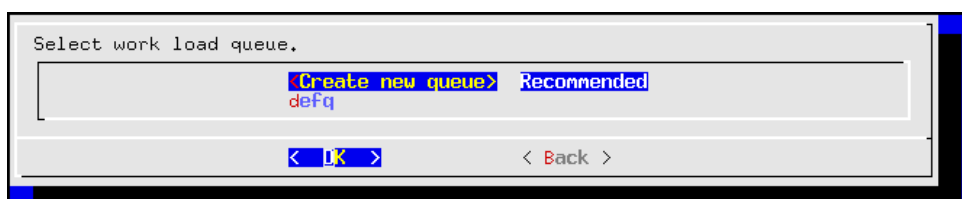
A bucket name can be set (figure 4.11):

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):

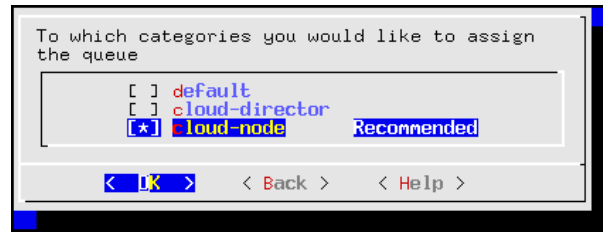


Figure 4.14: Cloud Storage Configuration Processing With `cm-cloud-storage`: Setting The Queue Category

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

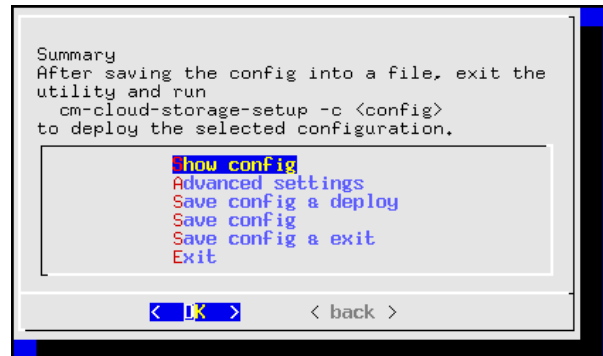


Figure 4.15: Cloud Storage Configuration Processing With `cm-cloud-storage`: Summary Screen

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@bright81 ~]# cm-cloud-storage-setup -c cm-cloud-storage-setup.conf
```

4.4 Miscellaneous Cloud Commands

4.4.1 The `cm-cloud-copy` Tool

The `cm-cloud-copy` tool is automatically installed with the `cmdaemon-cmsub` package (section 4.3.1). 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 2018, 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 4.1, 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 25.

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 26.

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 4.1.1 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):

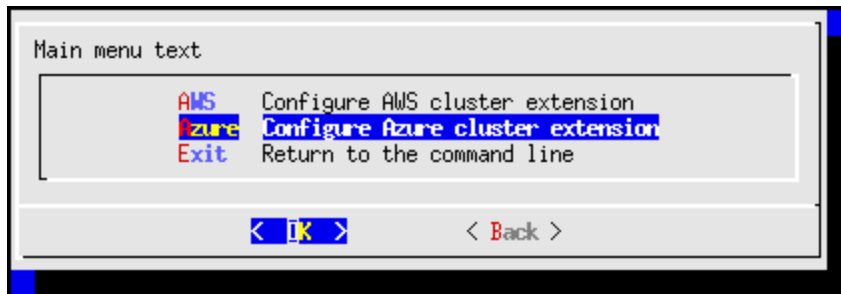


Figure 5.1: Configuration Processing With `cm-cluster-extension`: Azure Selection

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):

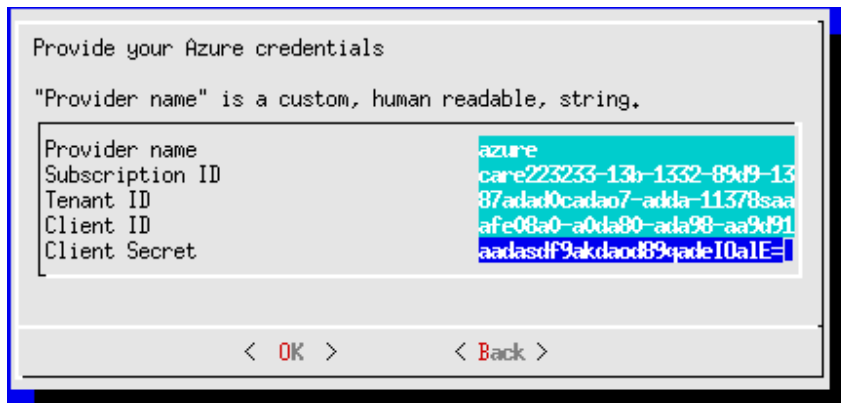


Figure 5.2: Configuration Processing With `cm-cluster-extension`: Azure Credentials

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
az provider register --namespace Microsoft.ADHybridHealthService --wait
az provider register --namespace Microsoft.Authorization --wait
az provider register --namespace Microsoft.Billing --wait
az provider register --namespace Microsoft.ClassicSubscription --wait
az provider register --namespace Microsoft.Commerce --wait
az provider register --namespace Microsoft.Consumption --wait
az provider register --namespace Microsoft.Features --wait
az provider register --namespace Microsoft.MarketplaceOrdering --wait
az provider register --namespace Microsoft.Resources --wait
az provider register --namespace Microsoft.support --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):

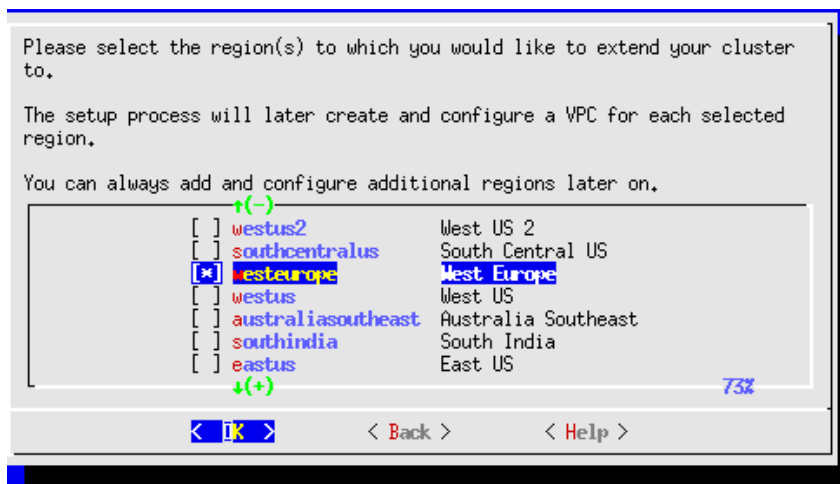


Figure 5.3: Configuration Processing With cm-cluster-extension: Azure Regions

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):

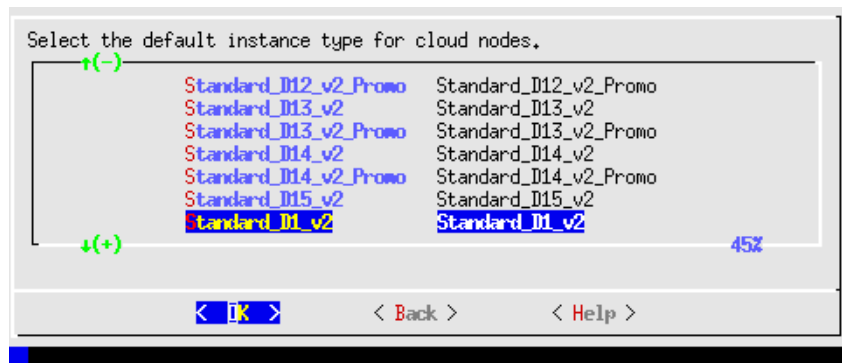


Figure 5.4: Configuration Processing With `cm-cluster-extension`: Azure Default Instance Type For Cloud Nodes

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 ; types ; list"
```

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):

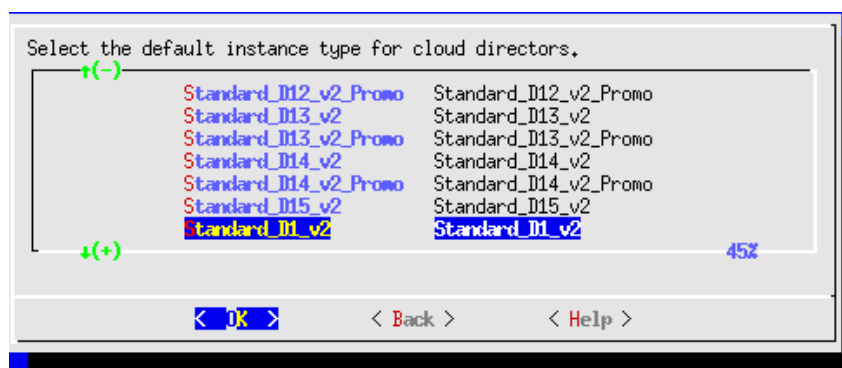


Figure 5.5: Configuration Processing With `cm-cluster-extension`: Azure Default Instance Type For Cloud Directors

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:

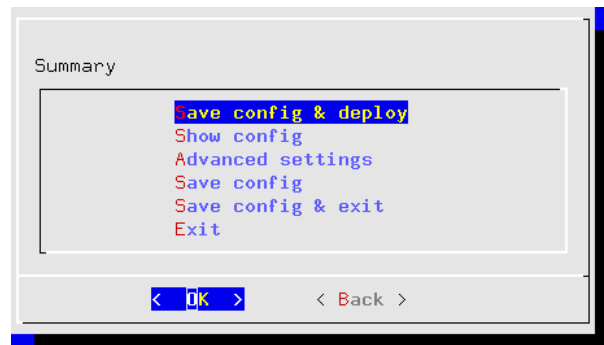


Figure 5.6: Configuration Processing With `cm-cluster-extension`: Azure Options Summary Screen

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@bright81 ~]# 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@bright81 ~]# 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@bright81 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):          4
Source node:            bright81
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@bright81 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):          5
Source node:            bright81
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 137 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 4.2.

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.

¹The 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 4.3. 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 4.3 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 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 irregular¹ 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

```
[bright81]% device cloudsettings eu-west-1-director
[bright81->device[eu-west-1-director]->cloudsettings]% storage
[bright81->...->cloudsettings->storage]% set ebs size 15GB; commit
[bright81->...->cloudsettings->storage]% device cloudsettings cnode001
[bright81->device[cnode001]->cloudsettings]% storage
[bright81->...->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.

¹irregular 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 3.3.7 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 Infini-Band 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.

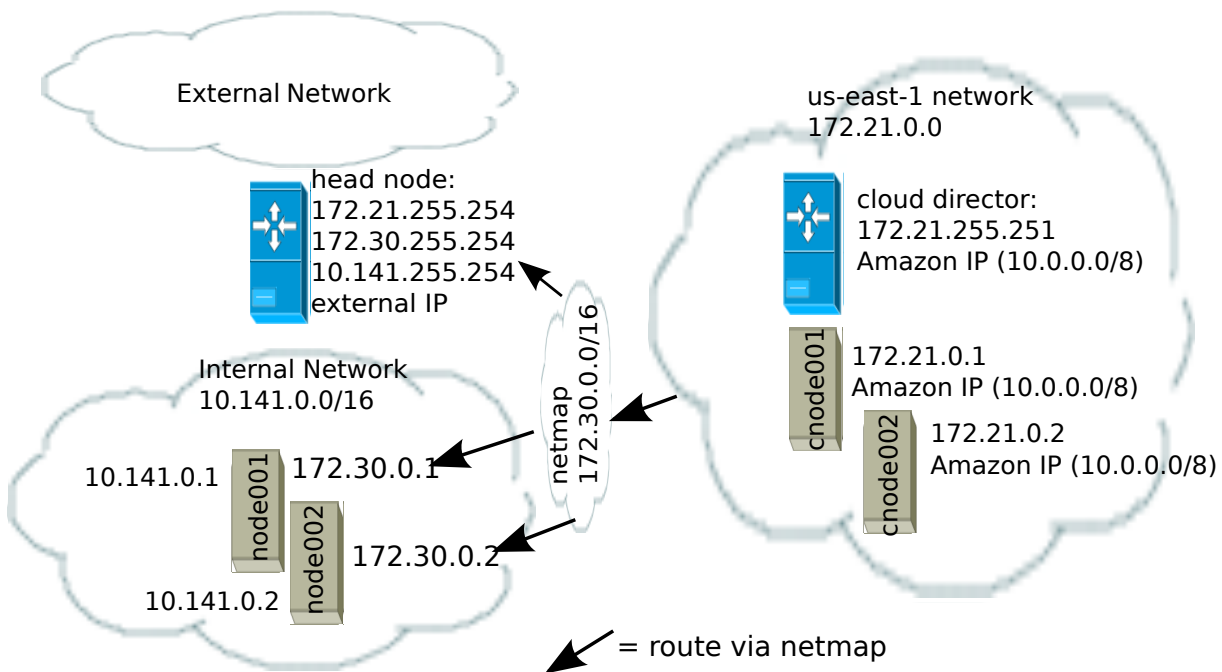


Figure 6.1: NetMap In Relation To The General Network Scheme

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.

6.5 Internet Connectivity For Cloud Nodes

If a cloudbursting setup is connected to the internet with the default settings, then only the CX-OS cloud compute nodes have no access to the internet.

Cloud compute node types in the other cloudbursting setups—CX-AWS, CX-Azure, COD-AWS, COD-Azure, and COD-OS—can all access the internet by default.

This is elaborated upon in table 6.5:

Table 6.5: Cloud compute node access to internet

Cloud node type	Internet access by default?	Details
CX-AWS	Yes	Cloud compute nodes are routed via an sNAT gateway cloud director node. They therefore do not normally require assignment of Elastic IPs. If assigning an Elastic IP to a node is required, then such a cloud node should be created on the 'public' VPC subnet (by default cloud compute nodes are created on the 'private' VPC subnet).
CX-Azure	Yes	Cloud compute nodes use the built-in NAT capabilities of Azure's gateway when accessing the internet.
CX-OS	No	<p>Cloud compute nodes are routed via a cloud director sNAT gateway. However for this to work in a OpenStack environment, the port security plugin needs to be enabled for Neutron, followed by disabling the port security setting on the cloud director ports. By default, the port security plugin is disabled for Neutron, while the port security setting is in an enabled state.</p> <p>Disabling the security setting causes the security groups to be no longer applied in the cloud director. This means anyone with Layer-3 (IP) access to the cloud director could try to establish a connection to any of the cloud director's ports. Therefore, by default for CX-OS, the port security plugin is left disabled, and the port security setting left enabled.</p>
COD-AWS	Yes	Cloud compute nodes use the head node as an sNAT gateway.
COD-Azure	Yes	Cloud compute nodes can access the internet.
COD-OS	Yes	Cloud compute nodes are routed via an sNAT gateway head node.

6.6 Passing Kernel Parameters To Cloud Nodes

If a cluster administrator configures a non-cloud cluster, then kernel parameters can be set for a particular software image used by the regular nodes. For example, in `cmsh`, if a software image `<image name>` is used, then kernel parameters such as `root=/dev/sda2 rootdelay=10 pti=auto` can be set via the navigation path:

```
cmsh→softwareimage→use <image name>→set kernelparams "root=/dev/sda2 rootdelay=10 pti=auto"
```

However, kernel parameters are not passed to cloud nodes via this mechanism at the time of writing (November 2019). If there is a need to pass kernel parameters to cloud nodes, then Bright Computing support should be contacted.

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 73)
- *EC2-VPC* (page 74)
- *classic cloud* (page 74)
- *defaultVPC* (page 74)
- *private cloud* (page 74)
- *custom VPC* (page 74)
- *elastic IP addresses* (page 77)

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 77) 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 8.1 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 55 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 4.1),
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

```
[bright81->network]% add vpc-0-sn-0
[bright81->network*[vpc-0-sn-0*]]% set type cloud
[bright81->network*[vpc-0-sn-0*]]% set baseaddress 10.0.0.0
[bright81->network*[vpc-0-sn-0*]]% set netmaskbits 24
[bright81->network*[vpc-0-sn-0*]]% set ec2availabilityzone eu-west-1a
[bright81->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

```
[bright81->network]% clone vpc-0-sn-0 vpc-0-sn-1
[bright81->network*[vpc-0-sn-1*]]% set baseaddress 10.0.1.0
[bright81->network*[vpc-0-sn-1*]]% commit
```

- **Custom VPC creation:** The following example session in the `vpc` 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 74. It is of type `ec2` and within a network that contains the two subnets specified earlier.

Example

```
[bright81->cloud[Amazon EC2]->vpcs]%
[bright81->...->vpcs]% add vpc-0
[bright81->...->vpcs*[vpc-0*]]% set region eu-west-1
[bright81->...*[vpc-0*]]% set baseaddress 10.10.0.0
[bright81->...*[vpc-0*]]% set netmaskbits 16
[bright81->...*[vpc-0*]]% set subnets vpc-0-sn-0 vpc-0-sn-1
[bright81->...*[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 address, 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

```
[bright81] cloud use amazonec2
[bright81->cloud[Amazon EC2]]% staticip allocate us-west-1
Allocating Static IP. Please wait...
Successfully allocated the following static IP: 54.215.158.42
[bright81->cloud[Amazon EC2]]% staticip list
Cloud Provider   Cloud Region   Static IP       Assigned to
-----
Amazon EC2       us-west-1      54.215.158.42  <not assigned>
[bright81->cloud[Amazon EC2]]%
```

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

Example

```
[bright81->cloud[Amazon EC2]]% staticip release 54.215.158.42
Releasing static IP 54.215.158.42. Please wait...
Successfully released the static ip.
[bright81->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

```
[bright81->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

```
[bright81->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
```