

Bright Cluster Manager 9.1

Cloudbursting Manual

Revision: 0fcda3a

Date: Tue May 20 2025



©2021 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	v
0.4 Getting Professional Services	vi
1 Introduction	1
2 Cluster On Demand Cloudbursting With Azure, AWS, VMware, Or Bright OpenStack	3
2.1 Introduction	3
2.2 Requirements For COD Cloudbursting	3
2.2.1 Credentials For Azure	4
2.2.2 Credentials For AWS	5
2.2.3 Credentials For VMware	5
2.3 COD Via Bright Computing Customer Portal (Azure And AWS)	5
2.4 COD Via The COD Command Line Tool	10
2.4.1 COD Via Docker Image	11
2.4.2 COD Via A Python Package	12
2.5 COD On Azure Via Command Line	13
2.5.1 Minimal Configuration File For COD On Azure	13
2.5.2 Cluster Creation Run With <code>cm-cod-azure</code>	13
2.5.3 Listing Clusters With <code>cm-cod-azure</code>	15
2.5.4 Cluster Removal With <code>cm-cod-azure</code>	15
2.6 COD On AWS Via Command Line	16
2.6.1 Minimal Configuration File For COD On AWS	16
2.6.2 Cluster Creation Run With <code>cm-cod-aws</code>	16
2.6.3 Listing Clusters With <code>cm-cod-aws</code>	18
2.6.4 Cluster Removal With <code>cm-cod-aws</code>	18
2.7 COD On VMware Via Command Line	18
2.7.1 Minimal Configuration File For COD On VMware	19
2.7.2 Cluster Creation Run With <code>cm-cod-vmware</code>	19
2.7.3 Listing Clusters With <code>cm-cod-vmware</code>	20
2.7.4 Cluster Removal With <code>cm-cod-vmware</code>	21
2.8 COD Client Configuration And Command Line Options	21
2.8.1 Command Line Structure	21
2.8.2 Configuration Files	23
2.9 Preparing VMware vSphere For COD	24
2.9.1 Introduction	24
2.9.2 Requirements	24
2.9.3 Networking	25
2.9.4 Accounts, Privileges, Roles And Permissions	25

2.9.5	COD Image Management	28
2.10	Using NSX-T To Manage COD VMware Networks	30
2.10.1	Requirements	30
2.10.2	Configuring NSX-T	30
2.10.3	Configuring vSphere	31
2.10.4	Configuration And Use Of The COD Client	31
2.11	Using The AWS EC2 Management Console	31
2.11.1	Status Checking Via Instance Selection From Instances List	32
2.11.2	Acting On An Instance From The AWS EC2 Management Console	33
2.11.3	Connecting To An Instance From The AWS EC2 Management Console	33
2.11.4	Viewing The Head Node Console	33
2.11.5	Security Group Configuration To Allow Access To The Head Node Via <code>cmsh</code> Or Bright View	34
2.12	Using The Azure Dashboard	35
2.13	COD: Cloud Node Start-up	36
2.13.1	COD: IP Addresses In The Cloud	37
2.14	COD With AWS: Optimizing AWS For High Performance Computing (HPC)	37
3	Cluster Extension Cloudbursting	39
3.1	Cluster Extension With AWS: The Bright View Cluster Extension Wizard	41
3.1.1	Introduction	41
3.1.2	AWS Credentials	41
3.1.3	Select Regions	42
3.1.4	Select Software Images	44
3.1.5	Summary & Deployment	44
3.1.6	Deploy	45
3.2	Cluster Extension With AWS: Cloud Director Startup From Scratch	46
3.2.1	Setting The Cloud Director Disk Storage Device Type	47
3.2.2	Setting The Cloud Director Disk Size	48
3.2.3	Tracking Cloud Director Startup	49
3.3	Cluster Extension With AWS: Cloud Node Startup From Scratch	51
3.4	Cluster Extension With AWS: Cloud Director And Cloud Node Startup From Snapshots	51
3.4.1	Cloud Director Startup From Snapshots	51
3.4.2	Cloud Node Startup From Snapshots	53
3.5	Cluster Extension With AWS: Optimizing AWS For High Performance Computing (HPC)	53
3.5.1	Optimizing HPC Performance: EBS Volume Type	54
3.5.2	Optimizing HPC Performance: Placement Groups	54
3.5.3	Optimizing HPC Performance: Disabling Hyper-Threading	54
3.5.4	Optimizing HPC Performance: Using Elastic Network Adapter Instances	55
3.5.5	Optimizing HPC Performance: Using A Different Clock Source	55
3.5.6	Optimizing HPC Performance: Setting The Socket Buffer Sizes And TCP/IP Pa- rameters In The Software Image	55
4	Cluster Extension Cloudbursting With AWS Using The Command Line And <code>cmsh</code>	57
4.1	The <code>cm-cluster-extension</code> Script For Cluster Extension Clusters	57
4.1.1	Running The <code>cm-cluster-extension</code> Script On The Head Node For Cluster Ex- tension Clusters	57

4.1.2	Launching The Cloud Director For Cluster Extension Clusters	62
4.2	Launching The Cloud Nodes	63
4.2.1	Creating And Powering Up Many Nodes	63
4.3	Submitting Jobs With cmjob And Cloud Storage Nodes, For Cluster Extension Clusters	63
4.3.1	Installation And Configuration of cmjob For Data-aware Scheduling To The Cloud	64
4.3.2	Integration Of cmjob With AWS FSx For Lustre	70
4.4	Miscellaneous Cloud Tools	75
4.4.1	Setting Exclude Lists With excludelistsnippets	75
4.4.2	The provisioningassociations Mode	76
4.5	Connecting To AWS With Direct Connect Or A Hardware VPN	78
4.5.1	Creating a VPC	78
4.5.2	Connecting The Local Network To The VPC	78
4.5.3	Configuring And Deploying The Cluster Extension	79
5	Cluster Extension Cloudbursting With Azure	83
5.1	Introduction To Cluster Extension Cloudbursting With Azure	83
5.2	Cluster Extension Into Azure	83
5.3	Cluster Extension Into Azure: Cloud Node Startup From Scratch	90
5.4	Cluster Extension Into Azure: shutdown Vs power off	90
5.5	Submitting Jobs With cmjob And Cloud Storage Nodes, For Azure Cluster Extension Clusters	91
5.5.1	Integration Of cmjob With Azure NetApp Files	91
5.6	Creating An Azure Cluster Extension Using ExpressRoute Or A Hardware VPN	96
5.6.1	Creating A Virtual Network	96
5.6.2	Connecting The Local Network To The Virtual Network	100
5.6.3	Configuring And Deploying The Cluster Extension	101
5.7	Viewing And Setting Azure Generation 1 And Generation 2 Cloud Nodes	102
5.7.1	Introduction	102
5.7.2	Implementation Of Hyper-V Generation Choice In The Cluster Manager	102
6	Cluster-On-Demand For OpenStack	103
6.1	Introduction	103
6.2	The cm-cod-os Arguments	103
6.2.1	The cm-cod-os Top Level Arguments	103
6.2.2	The cm-cod-os Context Tree	104
6.2.3	The cm-cod-os Contexts And Optional Arguments Help Text	105
6.3	The cm-cod-os Configuration Files	111
6.3.1	The cm-cod-os Configuration Files Locations	111
6.3.2	Viewing The cm-cod-os Configuration File Options	112
6.3.3	Setting The cm-cod-os Configuration File Options And Corresponding Arguments	112
6.4	The cm-cod-os Environment Variables	113
6.5	Launching A COD	113
6.5.1	Administrative Preparation Of The Host Cluster	113
6.5.2	Launching And Configuring The Nested Cluster As A User	116

7	Cluster Extension Cloudbursting With OpenStack	121
7.1	Introduction	121
7.1.1	Cluster Extension In General	121
7.1.2	Overview Of Carrying Out Cluster Extension Into OpenStack	121
7.2	Deploying A Cluster Extension Into OpenStack Via An Ncurses Dialog	121
7.3	Deploying A Cluster Extension Into An OpenStack Instance Via A Bright View Wizard . .	126
7.4	Connectivity To The Internet For OpenStack Compute Nodes	127
8	Cloud Considerations And Choices With Bright Cluster Manager	129
8.1	Differences Between Cluster On Demand And Cluster Extension	129
8.2	Hardware And Software Availability	129
8.3	Reducing Running Costs	129
8.3.1	Spot Pricing	130
8.3.2	Storage Space Reduction	130
8.4	Setting The Cloud Node Images	130
8.5	Ignoring Prepaid Node Limits With Pay-Per-Use Images	131
8.5.1	Bursting Past The Prepaid Limit In AWS	131
8.5.2	Bursting Past The Prepaid Limit In Azure	133
8.6	Address Resolution In Cluster Extension Networks	136
8.6.1	Resolution And globalnet	136
8.6.2	Resolution In And Out Of The Cloud	136
8.7	Internet Connectivity For Cloud Nodes	138
8.8	Passing Kernel Parameters To Cloud Nodes	139
8.9	Setting Up And Creating A Custom VPC	140
8.9.1	Elastic IP Addresses And Their Use In Configuring Static IP Addresses	140
8.9.2	Subnets In A Custom VPC	140
8.9.3	Creating The Custom VPC	141

Preface

Welcome to the *Cloudbursting Manual* for Bright Cluster Manager 9.1.

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 9.1 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 administration 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 *Edge Manual* describes how to deploy Bright Edge 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 Menu icon, , following the click-path:



→Help→Feedback Form

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 17.2 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 specific 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 looks at Cluster On Demand for Bright OpenStack.

Chapter 7 looks at Cluster Extension for Bright OpenStack.

Chapter 8 discusses some miscellaneous aspects of cloudbursting.

2

Cluster On Demand Cloudbursting With Azure, AWS, VMware, Or Bright OpenStack

2.1 Introduction

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. COD can run in:

- Azure (COD-Azure) (described in this Chapter)
- AWS (COD-AWS) (described in this Chapter)
- VMware vSphere (COD-VMware) (described in this Chapter)
- Bright OpenStack (COD-OS) (described in Chapter 6)

When the requirements outlined in the sections that follow have been met, COD clusters can be managed (created, deleted, etc.) in the following ways:

- Via the Bright Computing customer portal (Azure and AWS) (section 2.3)
- By using the COD command line tool via a Docker image or a Python package (all cloud platforms) (section 2.4).

Any of the preceding can be used to launch a COD head node in the cloud. Once the head node is running, then compute nodes are managed by using the various management interfaces provided by Bright Cluster Manager running on the head node (section 2.13).

2.2 Requirements For COD Cloudbursting

The high level requirements for COD are:

- an account on the Bright Computing Customer Portal website at <http://customer.brightcomputing.com/Customer-Login>, when managing COD clusters via the portal
- a Bright Cluster Manager product key. This key is later activated when the license is installed (Chapter 4 of the *Installation Manual*) on the head node.

- Credentials for the cloud environment in which the COD cluster is created. This is covered in the following sections.
- For VMware vSphere, some preparation needs to be done by the VMware administrator (section 2.9)

2.2.1 Credentials For Azure

To use Azure, an Azure account subscription is needed from Microsoft. COD cloudbursting requires the following associated Azure credentials to launch:

- tenant ID
- subscription ID
- Client ID
- Client Secret

A CLI-centric way to obtain these credentials requires logging into the Azure web portal using an account that has sufficient privileges. The Azure web bash console is then opened, and the subscription ID can then be listed for the account with:

Example

```
azure@Azure:~$ az account list -o table
Name          CloudName      SubscriptionId                               State      IsDefault
-----
anne          AzureCloud     23748c3e-507b-11e9-a994-fa163e9854eb        Enabled    False
nerds         AzureCloud     b9e22a88-507a-11e9-9352-fa163e9854eb        Enabled    False
```

A service principal (sp) is now created for role-based access control create-for-rbac in Active Directory (ad), and the remaining 3 credentials can then be seen:

Example

```
azure@Azure:~$ az ad sp create-for-rbac --name my-temp-service-principal-for-fred
Changing "my-temp-service-principal-for-fred" to a valid URI of
"http://my-temp-service-principal-for-fred", which is the required format used for service
principal names
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36

"appId": "dcf8151e-507a-11e9-a104-fa163e9854eb",    ## "Client ID"
"displayname": "my-temp-service-principal-for-fred",
"name": "http://my-temp-service-principal-for-fred",
"password": "bc9f571e-fe4c-43d5-909d-4bc66796eb41", ## "Client secret"
"tenant": "8cb88849-6e18-46d6-b0fa-551a47a31681"   ## Tenant ID
```

The newly-created service principal is added to the desired subscription as a contributor. The value of appId must be used as the value to the --assignee option. This gives the application sufficient permissions for the cluster to run:

Example

```
azure@Azure:~$ az role assignment create --assignee dcf8151e-507a-11e9-a104-fa163e9854eb --role \
Contributor --subscription b9e22a88-507a-11e9-9352-fa163e9854eb
```

```
"canDelegate": null,
"id": "/subscriptions/b9e22a88-507a-11e9-9352-fa163e9854eb/providers/Microsoft.Authorization\
/roleAssignments/1094f75c-507b-11e9-ac8f-fa163e9854eb",
"name": "1094f75c-507b-11e9-ac8f-fa163e9854eb",
"principalId": "2f899334-507b-11e9-b312-fa163e9854eb",
"roleDefinitionId": "/subscriptions/b9e22a88-507a-11e9-9352-fa163e9854eb/providers\
/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
"scope": "/subscriptions/b9e22a88-507a-11e9-9352-fa163e9854eb",
"type": "Microsoft.Authorization/roleAssignments"
```

The Microsoft documentation suggests that using the name instead of the subscription ID should also work. However in the version that was used at the time of writing (May 2019) this did not work, and an error 400 was displayed.

2.2.2 Credentials For 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.

2.2.3 Credentials For VMware

To use VMware, the following details should be obtained from the VMware administrator:

- vCenter API endpoint address
- vCenter username
- vCenter password
- vCenter resource pool

2.3 COD Via Bright Computing Customer Portal (Azure And AWS)

The customer portal (section 2.2) 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 2.2. 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 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

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 2.2. 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:

-- Select region --

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

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

Product key:

☒ New Product Key
 ☐ Use Existing Product Key

Name:

Testing Test

Email:

test@brightcomputing.com

Organization:

BrightComputing

Organizational unit:

Office

Country:

United States

State:

Not applicable

City:

San Jose

Cluster name:

myazurecod

Bright version:

8.1

Linux distribution:

CentOS 7

Workload management:

PBS Professional CE

Head node flavor:

Standard_D1_v2

Head node disk size (GB):

50

Compute node count:

2

Compute node flavor:

Standard_D1_v2

SSH access:

☒ Set SSH public key
 ☐ Set SSH password

SSH public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDAOD
UV5n/Cuf6vm7O+zetyD0j8bG5hnbsVLLYETc
QqtEJgBKqsizWJrkleyK4dgUurvWArwOuHw
0CKqh/hwCqrFMRUmxXeVi7tfkddsUnMoGg
xA9np/gB7QgQ07gEDqTfTr7tS9ZPDMQnd
QtBwLHbpWuhnoU6GIOtG/OVaViv15/TjJNIL
wxgHpZKTz5Y0JmBi4wK+1Pjvlj+fxwvmRJdG
```

Submit

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

Form submission begins the COD installation (figure 2.4).

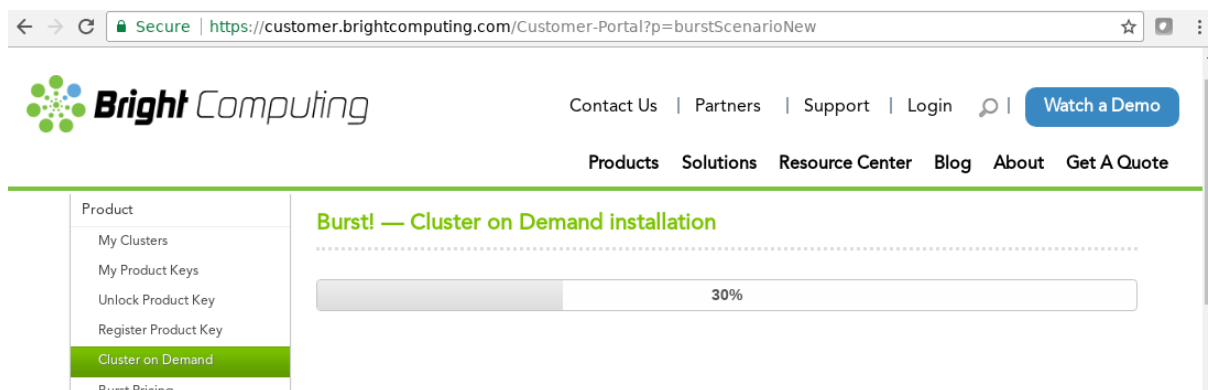


Figure 2.4: COD via customer portal: deployment in progress

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).

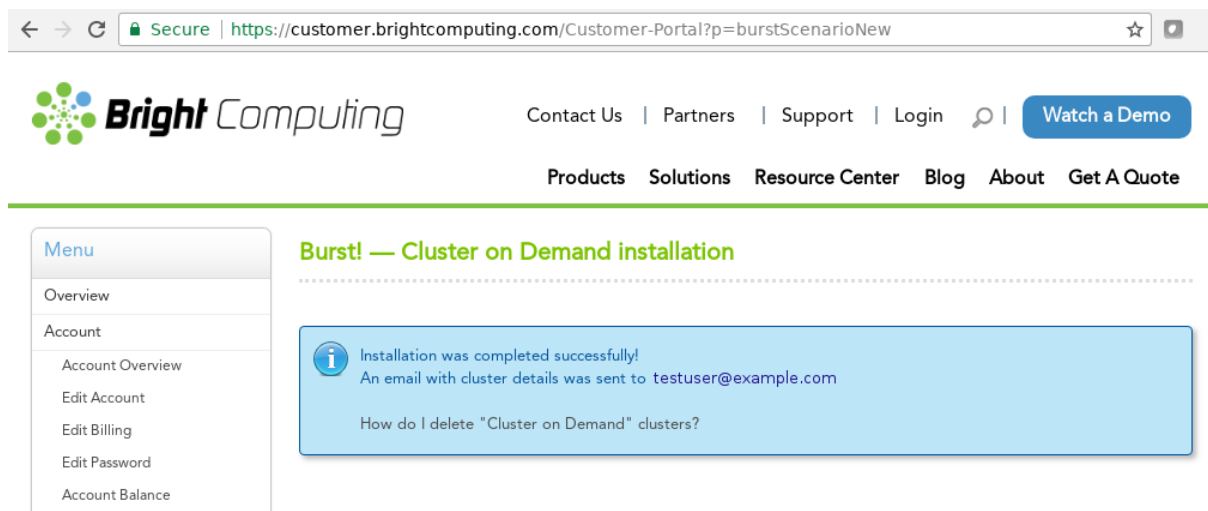


Figure 2.5: COD via customer portal: finished deploying

2.4 COD Via The COD Command Line Tool

There are two options for running the COD command line tool on a standalone PC:

- A Docker image.
- Installing a Python package in a local (virtual) environment.

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. The Python package can also be installed on a standalone PC or it can be installed on a Bright node. Section 2.4.1 describes the procedure for installing and configuring COD on Docker on a Bright node in more detail. Section 2.4.2 describes the procedure for installing and configuring the COD client using a Python package on a Bright in more detail.

Once the COD client is installed and configured, the next stage is to launch the COD inside the cloud service. There are three 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 16).
- Azure, made available by Microsoft. The CLI utility in the Docker instance that launches the COD is then `cm-cod-azure` (page 13).
- VMware vSphere. The CLI utility in the Docker instance that launches the COD is then `cm-cod-vmware` (page 18).

2.4.1 COD Via Docker Image

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.

The COD Docker image is typically run from a standalone machine, such as, for example, a laptop that is not part of a Bright cluster. However, launching a COD from a Bright cluster is also possible, and can be carried out as follows:

- Docker is installed in Bright Cluster Manager with `cm-docker-setup` (section 9.1.1 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@bright91 ~]# ssh node001
[root@node001 ~]# module load docker
[root@node001 ~]# docker pull brightcomputing/cod:latest
...
cac15ba059ef: Pull complete
Digest: sha256:19b263033090e1a70043989decdf3c3870d3def8c2e69b2a85ac293fd7d149ab
Status: Downloaded newer image for brightcomputing/cod:latest
```

- The command line COD tools installed in the container image all have several command line arguments. Instead of the user having to specify many arguments on every invocation of a tool, the options can conveniently be set in configuration files. To provide access to such configuration files it is recommended that the home directory of the user running the tool is mounted into the container instance. It is also recommended to set up the following shell aliases for each of the tools:

Example

```
alias cm-cod-aws='docker run --rm -it --network host -v /root:/root brightcomputing/cod\
:latest cm-cod-aws'
alias cm-cod-azure='docker run --rm -it --network host -v /root:/root brightcomputing\
/cod:latest cm-cod-azure'
alias cm-cod-vmware='docker run --rm -it --network host -v /root:/root brightcomputing\
/cod:latest cm-cod-vmware'
```

In the preceding aliases:

- the `--rm` option ensures that the container instance created to run a command is also cleaned up on completion
- the `-it` flags are required because some commands may present interactive prompts to the user
- the `--network host` option gives the container network access
- the argument to the `-v` flag has the form of `<source>:<destination>`, where
 - * `<source>` points to the home directory of the user invoking the command, and
 - * `<destination>` is the path where the directory is mounted inside the running container instance. The path of `<destination>` should always be `/root`.
- Once a COD cluster has been created, it is typically accessed using SSH. It is strongly recommended to use key-based authentication instead of password-based authentication. Since the home directory for users is mounted into the container, any existing SSH key pairs (stored in `~/.ssh`) can be used. If no key pair exists, then `ssh-keygen` can be used to create one. To ensure the highest level of compatibility it is recommended to use RSA as the key type and a minimum of 2048 bits.

2.4.2 COD Via A Python Package

Instead of running the COD client in a Docker container it is also possible to install and run the client directly on the host system in a Python virtual environment. The client can be run from a Bright cluster as follows:

- A virtual environment is created to install the COD client:

Example

```
[root@bright91 ~]# ssh node001
[root@node001 ~]# module load python3
[root@node001 ~]# python3 -m venv .venvs/cod
[root@node001 ~]# source .venvs/cod/bin/activate
(cod) [root@node001 ~]#
```

- Next, the required COD clients are installed. Possible clients are:
 - `cm-cluster-on-demand-aws`
 - `cm-cluster-on-demand-azure`
 - `cm-cluster-on-demand-vmware`

For example, the COD client tool for AWS can be installed as follows:

Example

```
(cod) [root@node001 ~]# pip install cm-cluster-on-demand-aws
...
Successfully installed boto3-1.7.75 botocore-1.10.84 cm-cluster-on-demand-aws-9.1 ...
```

More than one client can be installed.

- After the COD client has been installed, it can be run whenever the virtual environment is activated, without having to install it again:

Example

```
[root@node001 ~]# source .venvs/cod/bin/activate
(cod) [root@node001 ~]# cm-cod-aws
usage: cm-cod-aws [-h] [--config CONFIG] [--no-system-config] [-v]
                [--show-configuration] [--log-file LOG_FILE] [--version]
                cluster,c,cluster create,cc,cluster list,cl,cluster
                delete,cd,cr,cremove,image,i,image
                list,il,instancetype,config ...
cm-cod-aws: error: the following arguments are required: group
```

2.5 COD On Azure Via Command Line

Launching a COD inside Azure is carried out by running `cm-cod-azure` from either the Docker image or the Python package. The `cm-cod-azure` command can take configuration options from one or more configuration files, environment variables, or command line options. More details on this are given in section 2.8.

Typically some of the more static options are configured in a configuration file, while some of the more variable options are specified on the command line.

Online help is also available: for example, running `cm-cod-azure cluster create -h` displays a list of options and explanations for the `cluster create` command.

2.5.1 Minimal Configuration File For COD On Azure

It is recommended to create the following minimal configuration in `~/cm-cluster-on-demand.d/config.ini`:

Example

```
[azure.credentials]
azure_subscription_id=5e519b1e-aff9-e839-bc3a-c5d87e9d0d5
azure_tenant_id=2c89c8dd-6b8b-5393-60f4-d876cbe188f
azure_client_id=afe13723-c80a-68e2-9cd0-e95a657106a
azure_client_secret=aiVohwi70hJ6igim=
azure_location=westeurope

[cluster.create.license]
license_product_key=123456-123456-123456-123456-123456

[cluster.create.password]
ssh_pub_key_path=/root/.ssh/id_rsa.pub
```

The credentials and product key in the preceding example should be replaced with the appropriate values.

2.5.2 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 \
  --head-node-type Standard_D1_v2 --head-node-root-volume-size 50 \
  --nodes 3 --node-type Standard_D1_v2 \
  --name testcluster
```

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

Example

```
[root@cod-client ~]# cm-cod-azure cluster create \
--head-node-type Standard_D1_v2 --head-node-root-volume-size 50 \
--nodes 3 --node-type Standard_D1_v2 \
--name testcluster
09:35:01:      INFO: No custom cluster password specified. Using a random password.
Use --log-cluster-password to see it. If you are going to use the Bright View you have
to change this password by using cm-change-passwd script (change root password of head
node) or simply 'passwd root' command. If this cluster has a high-availability setup
with 2 head nodes, be sure to change the password on both head nodes.
09:35:05:      INFO: Temporary azure resources will be created in order to verify that
the API credentials have the required permissions, this can take a few minutes. You can
skip this step by specifying the following flag: '--skip-permission-verifications'
09:35:05:      INFO: Cluster Create
09:35:05:      INFO: -----
09:35:05:      INFO:          Cluster:  testcluster
09:35:05:      INFO: -----
09:35:05:      INFO:          Image name:  bcm-cod-image-9.0-8
09:35:05:      INFO:          Image date:  2020-08-21 15:10 (38d 18h ago)
09:35:05:      INFO:          Package groups:  none
09:35:05:      INFO:          Version:  9.0
09:35:05:      INFO:          Head nodes:  1 (Standard_D1_v2)
09:35:05:      INFO:          Nodes:  3 (Standard_D1_v2)
09:35:05:      INFO:          Resource Group:  testcluster_cod_resource_group
09:35:05:      INFO:          Region:  westeurope
09:35:05:      INFO: -----
09:35:05:      INFO: Press ENTER to continue and create the cluster.
09:35:05:      INFO: Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

09:35:45:      INFO: Credentials are valid and have read/write authorizations.
09:38:17:      INFO: ## Progress: 2
09:38:17:      INFO: #### stage: Creating resource group testcluster_cod_resource_group
09:38:18:      INFO: Creating storage account testclusterstorage68u3
09:38:36:      INFO: ## Progress: 7
09:38:36:      INFO: #### stage: Copying head node image from
https://brightimages.blob.core.windows.net/images/bcm-cod-image-9.0-8.vhd
09:38:36:      INFO: Copying https://brightimages.blob.core.windows.net/images/bcm-cod-image-9.0-8.vhd
to testclusterstorage68u3/images/testcluster-head-node-os-disk.vhd
09:38:37:      INFO: ## Progress: 12.0
09:38:37:      INFO: #### stage: Server side copy
09:38:59:      INFO: ## Progress: 12.13
09:38:59:      INFO: #### stage: Server side copy
09:39:03:      INFO: ## Progress: 12.26
...
09:49:40:      INFO: #### stage: Server side copy
09:49:41:      INFO: ## Progress: 51.85
09:49:41:      INFO: #### stage: Server side copy
09:49:43:      INFO: ## Progress: 51.98
09:49:43:      INFO: #### stage: Server side copy
09:49:45:      INFO: Elapsed: 11:08 min
09:49:45:      INFO: Creating Azure Image resource 'testcluster-head-node-os-disk' from blob
https://testclusterstorage68u3.blob.core.windows.net/images/testcluster-head-node-os-disk.vhd
09:50:21:      INFO: Deleting blob testclusterstorage68u3/images/testcluster-head-node-os-disk.vhd
09:50:21:      INFO: ## Progress: 5
09:50:21:      INFO: #### stage: Building deployment template
```

```

09:50:21:      INFO: generating cloud-init script
09:50:21:      INFO: ## Progress: 85
09:50:21:      INFO: #### stage: Creating and deploying Head node
09:52:22:      INFO: Head node IP: 51.145.169.45
09:52:22:      INFO: Waiting for sshd to start (ssh root@51.145.169.45).
09:52:22:      INFO: Waiting for CMDaemon to start.
09:55:05:      INFO: Waiting for CMDaemon to initialize.
09:56:07:      INFO: ## Progress: 100
09:56:07:      INFO: #### stage: Deployment finished successfully.
09:56:07:      INFO: -----
09:56:07:      INFO:          Cluster:  testcluster
09:56:07:      INFO: -----
09:56:07:      INFO:  Head node ID:  7f03d713-71ad-4eb6-88ff-045f82a41990
09:56:07:      INFO:      Public IP:  51.145.169.45
09:56:07:      INFO:      Key path:   /root/.ssh/id_rsa.pub
09:56:07:      INFO: -----
[root@cod-client ~]#

```

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.

The regular nodes running on Azure 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:

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

More details can be found in section 2.13.

2.5.3 Listing Clusters With `cm-cod-azure`

The clusters that are running can be listed with the `cluster list` command option:

Example

```

[root@cod-client ~]# cm-cod-azure cluster list
+-----+-----+-----+-----+...
| Cluster Name | Head node name      | Public IP      | Location  | ...
+-----+-----+-----+-----+...
| testcluster  | testcluster-head-node | 51.145.169.45 | westeurope | ...
+-----+-----+-----+-----+...

```

Since all of the credentials are specified in the configuration file, no further command line options are needed.

2.5.4 Cluster Removal With `cm-cod-azure`

A cluster can be removed with a command in the form of: `cluster delete <cluster name>`:

Example

```

[root@cod-client ~]# cm-cod-azure cluster delete testcluster
10:10:47:      INFO: Deleting resource group(s): testcluster_cod_resource_group
10:10:47:      INFO: Proceed? [yes/no]
y
10:10:52:      INFO: Started deleting resources for Resource Group 'testcluster_cod_resource_group'
[root@cod-client ~]#

```

2.6 COD On AWS Via Command Line

Launching a COD inside AWS is carried by running `cm-cod-aws` from either the Docker image or the Python package.

The `cm-cod-aws` command can take configuration options from one or more configuration files, environment variables, or command line options. More details on this are given in section 2.8.

Typically some of the more static options are configured in a configuration file, while some of the more variable options are specified on the command line.

Online help is also available. For example, running `cm-cod-aws cluster create -h` displays a list of options and explanations for the `cluster create` command.

2.6.1 Minimal Configuration File For COD On AWS

Creation of a minimal configuration file in `~/cm-cluster-on-demand.d/config.ini` is recommended, with the following contents:

Example

```
[aws.credentials]
aws_access_key_id=AKIAJAICHAZI7OHH1EEGO
aws_secret_key=0ocei6eiieshahG7IiJe9Quoud0oo
aws_region=eu-west-1

[cluster.create.license]
license_product_key=123456-123456-123456-123456-123456

[cluster.create.password]
ssh_pub_key_path=/root/.ssh/id_rsa.pub
```

The example credentials and product key should be replaced with the appropriate values.

2.6.2 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 \
  --head-node-type t3.medium --head-node-root-volume-size 50 \
  --nodes 3 --node-type t3.medium \
  --name testcluster
```

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

Example

```
[root@cod-client ~]# cm-cod-aws cluster create \
--head-node-type t3.medium --head-node-root-volume-size 50 \
--nodes 3 --node-type t3.medium \
--name testcluster
13:39:59:      INFO: No custom cluster password specified. Using a random password. Use
--log-cluster-password to see it. If you are going to use the Bright View you have to change this
password by using cm-change-passwd script (change root password of head node) or simply
'passwd root' command. If this cluster has a high-availability setup with 2 head nodes, be sure
to change the password on both head nodes.
13:40:01:      INFO: -----
13:40:01:      INFO:          Cluster:  testcluster
13:40:01:      INFO: -----
13:40:01:      INFO:          Image name:  brightheadnode-9.0-centos7u7-hvm-9(ami-07ef81b0e2d680131:9)
```



```

13:40:01:      INFO:      Image date:  2020-08-21 13:44 (39d 0h ago)
13:40:01:      INFO: Package groups: none
13:40:01:      INFO:      Version:  9.0
13:40:01:      INFO:      Distro:   centos7u7
13:40:01:      INFO:      Head nodes: 1 (t3.medium)
13:40:01:      INFO:      Head node IP: <auto> (set with --head-node-internal-ip)
13:40:01:      INFO:      Nodes:    3 (t3.medium)
13:40:01:      INFO:      Region:   eu-west-1
13:40:01:      INFO:      Key path:  /root/.ssh/id_rsa.pub
13:40:01:      INFO: -----
13:40:01:      INFO: Press ENTER to continue and create the cluster.
13:40:01:      INFO: Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

13:40:02:      INFO: ## Progress: 2
13:40:02:      INFO: ##### stage: Setting up VPC
13:40:02:      INFO: ## Progress: 5
13:40:02:      INFO: ##### stage: Creating VPC for on-demand testcluster in eu-west-1...
13:40:03:      INFO: ## Progress: 10
13:40:03:      INFO: ##### stage: Adding internet connectivity...
13:40:04:      INFO: ## Progress: 12
13:40:04:      INFO: ##### stage: Creating public subnet...
13:40:05:      INFO: ## Progress: 14
13:40:05:      INFO: ##### stage: Creating private subnet...
13:40:06:      INFO: Created security group for the head node: sg-057c11f4d9857a025
13:40:06:      INFO: Created security group for the compute node: sg-06249aedeffcb0d2d
13:40:07:      INFO: ## Progress: 16
13:40:07:      INFO: ##### stage: Creating head node (t3.medium)...
13:40:07:      INFO: Public key specified.
13:40:07:      INFO: Creating the head node VM instance.
13:40:08:      INFO: Created VM i-0d92887ef735f87b1.
13:40:09:      INFO: Waiting for head node to be running
13:40:25:      INFO: ## Progress: 22
13:40:25:      INFO: ##### stage: Done setting up VPC for on-demand testcluster.
13:40:25:      INFO: ## Progress: 22
13:40:25:      INFO: ##### stage: Starting the head node
13:40:25:      INFO: ## Progress: 30
13:40:25:      INFO: ##### stage: Waiting for the head node to start
13:40:25:      INFO: ## Progress: 32
13:40:25:      INFO: ##### stage: Assigning public IP to the head node (use
'--no-head-node-assign-public-ip' to skip)
13:40:27:      INFO: Elastic IP assigned: 54.217.104.57.
13:40:27:      INFO: ## Progress: 30
13:40:27:      INFO: ##### stage: Waiting for cloud-init to start (use
'--cloud-init-timeout 0' to skip)
13:40:27:      INFO: Waiting for cloud-init to finish on the cluster testcluster (by trying to
connect to 54.217.104.57:8081).
13:44:04:      INFO: ## Progress: 100
13:44:04:      INFO: ##### stage: Deployment of cluster: testcluster finished successfully.
13:44:14:      INFO: Script completed.
13:44:14:      INFO: -----
13:44:14:      INFO: Time it took:   04:12
13:44:14:      INFO:   SSH string:   'ssh root@54.217.104.57'
54.217.104.57 testcluster
13:44:14:      INFO: Head node ID:   i-0d92887ef735f87b1
[root@cod-client ~]#

```

At this point, the head node running on AWS is ready for use. It can be accessed via SSH, as suggested in the SSH string 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:

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

More details can be found at section 2.13.

2.6.3 Listing Clusters With `cm-cod-aws`

The clusters that are running can be listed with the command option `cluster list`:

Example

```
[root@cod-client ~]# cm-cod-aws cluster list
14:16:44:      INFO: Listing clusters in region eu-west-1
+-----+-----+-----+-----+
| Cluster Name | VPC ID           | Head node ID       | Public IP         | ...
+-----+-----+-----+-----+
| testcluster  | vpc-019dacc0877b | i-0d92887ef735f87b1 | 54.217.104.57    | ...
+-----+-----+-----+-----+
```

Since all of the credentials are specified in the configuration file, no further command line options are needed.

2.6.4 Cluster Removal With `cm-cod-aws`

A cluster can be removed with a command in the form of: `cluster delete <cluster name>`:

Example

```
[root@cod-client ~]# cm-cod-aws cluster delete testcluster
14:18:57:      INFO: This will destroy VPCs for testcluster, continue?
14:18:57:      INFO: Proceed? [yes/no]
y
14:19:00:      INFO: Stopping instances for VPCs on-demand testcluster
14:19:00:      INFO: Listing instances...
14:19:00:      INFO: Issuing instances termination requests...
14:19:00:      INFO: Waiting until instances terminated...
14:20:31:      INFO: Destroying VPC on-demand testcluster
14:20:31:      INFO: Deleting subnets...
14:20:32:      INFO: Deleting route tables...
14:20:32:      INFO: Detaching and deleting gateways...
14:20:33:      INFO: Flushing permissions...
14:20:34:      INFO: Deleting security groups...
14:20:35:      INFO: Deleting VPC...
14:20:35:      INFO: Done destroying VPC on-demand testcluster
[root@cod-client ~]#
```

2.7 COD On VMware Via Command Line

Launching a COD inside VMware is carried out by running `cm-cod-vmware` from either the Docker image or the Python package.

The `cm-cod-vmware` command can take configuration options from one or more configuration files, environment variables, or command line options. More details on this are given in section 2.8.

Typically some of the more static options are configured in a configuration file, while some of the more variable options are specified on the command line.

Online help is also available: for example, running `cm-cod-vmware cluster create -h` displays a list of options and explanations for the `cluster create` command.

2.7.1 Minimal Configuration File For COD On VMware

Creation of a minimal configuration file in `~/cm-cluster-on-demand.d/config.ini` is recommended, with the following contents:

Example

```
[vmware.credentials]
vmware_address = vcenter.mycompany.com
vmware_username = "bob@vsphere.local"
vmware_password = "SomeSecretWithCapsAndNumbersEtc123!"
vmware_resource_pool = "bob-cod-resource-pool"

[cluster.create.license]
license_product_key=123456-123456-123456-123456-123456

[vmware.network]
vds switch_name = DSwitch01
externalnet_vdportgroup = DSwitch01-externalnet
internalnet_vlan_ids = 400-499

[vmware.cluster.create]
ssh_pub_key_path = ~/.ssh/id_rsa.pub
```

The example credentials and product key should be replaced with the appropriate values.

In most cases the contents of this configuration file should be supplied to the COD user by the VMware administrator. Additionally, details on setting up VMware for COD are given in section 2.9.

2.7.2 Cluster Creation Run With `cm-cod-vmware`

An example of a `cm-cod-vmware` command that can be run is then:

Example

```
cm-cod-vmware cluster create \
  --head-node-number-cpus 2 --head-node-memory-size 2048 --head-node-root-volume-size 25 \
  --node-number-cpus 2 --node-memory-size 2048 --node-root-volume-size 10 \
  --name testcluster
```

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

Example

```
[root@cod-client ~]# cm-cod-vmware cluster create \
--head-node-number-cpus 2 --head-node-memory-size 2048 --head-node-root-volume-size 25 \
--node-number-cpus 2 --node-memory-size 2048 --node-root-volume-size 10 \
--name testcluster
13:25:43:      INFO: No custom cluster password specified. Using a random password. Use
--log-cluster-password to see it. If you are going to use the Bright View you have to change
this password by using cm-change-passwd script (change root password of head node) or simply
'passwd root' command. If this cluster has a high-availability setup with 2 head nodes, be
sure to change the password on both head nodes.
13:25:43:      INFO: Please wait...
```

```

13:26:00: INFO: -----
13:26:00: INFO:           Cluster:  testcluster
13:26:00: INFO: -----
13:26:00: INFO:           Image name:  bcmh-centos8u1-trunk-192(centos8u1-trunk:192)
13:26:00: INFO:           Node image:  bcmn-centos8u1-trunk-192(centos8u1-trunk:192)
13:26:00: INFO:           Image date:  2020-09-28 21:38 (1d 15h ago)
13:26:00: INFO:           Package groups:  none
13:26:00: INFO:           Version:  trunk
13:26:00: INFO:           Distro:  centos8u1
13:26:00: INFO:           Head node:  1 (2 vCPUs, 2048 MB RAM)
13:26:00: INFO:                   Root disk: 25 GiB
13:26:00: INFO:           Nodes:  5 (2 vCPUs, 2048 MB RAM)
13:26:00: INFO:                   Root disk: 10 GiB
13:26:00: INFO: -----
13:26:00: INFO: Press ENTER to continue and create the cluster.
13:26:00: INFO: Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

13:26:48: INFO: Creating cluster testcluster
13:26:48: INFO: Creating Portgroup testcluster with VLAN ID 400 on switch DSwitch01
13:26:48: INFO: Deploying OVF
13:28:18: INFO: Powering on head node
13:28:18: INFO: Node was powered on. Waiting for network
13:29:19: INFO: Waiting for sshd to start (ssh root@10.2.180.3).
13:29:19: INFO: Waiting for CMDaemon to start.
13:29:43: INFO: Waiting for CMDaemon to initialize.
13:30:43: INFO: Cluster has been created successfully.
13:30:43: INFO: -----
13:30:43: INFO:           Cluster:  testcluster
13:30:43: INFO: -----
13:30:43: INFO:           SSH string:  ssh root@10.2.180.3
13:30:43: INFO:                   IP:  10.2.180.3
13:30:43: INFO:           Key path:  /root/.ssh/id_rsa.pub
13:30:43: INFO: -----
[root@cod-client ~]#

```

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

The regular nodes running on VMware 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:

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

More details can be found in section 2.13.

2.7.3 Listing Clusters With `cm-cod-vmware`

The clusters that are running can be listed with the `cluster list` command line option:

Example

```

[root@cod-client ~]# cm-cod-vmware cluster list
14:43:31: INFO: Listing clusters...
+-----+-----+-----+-----+...
| Cluster Name | IP | Created (age) | Head status | ...
+-----+-----+-----+-----+...
| testcluster | 10.2.180.3 | 2020-09-30 13:26 (1h 16m) | poweredOn | ...
+-----+-----+-----+-----+...

```

Since all of the credentials are specified in the configuration file, no further command line options are needed.

2.7.4 Cluster Removal With `cm-cod-vmware`

A cluster can be removed with a command in the form of: `cluster delete <cluster name>`:

Example

```
[root@cod-client ~]# cm-cod-vmware cluster delete testcluster
14:45:38:      INFO: Will delete clusters: testcluster
14:45:38:      INFO: Proceed? [yes/no]
y
14:45:41:      INFO: Attempting to power off VMs: testcluster
14:45:41:      INFO: Deleting VMS: testcluster
14:45:42:      INFO: Deleting networks: testcluster
14:45:42:      INFO: Done.
[root@cod-client ~]#
```

2.8 COD Client Configuration And Command Line Options

All COD clients (`cm-cod-azure` (section 2.5), `cm-cod-aws` (section 2.6), and `cm-cod-vmware` (section 2.7)) have a similar command line structure. They can take many command line arguments. All of those arguments can also be specified in configuration files, or as environment variables. Options specified as environment variables override options specified in configuration files. Options specified on the command line override options defined as environment variables as well as options defined in configuration files.

2.8.1 Command Line Structure

The command line structure is similar for all the different COD clients. Each client offers a number of top-level commands, which each offer a number of sub-level commands. The top-level and sub-level commands are always the first two positional arguments. Each top-level as well as sub-level commands can have several optional arguments.

The syntax is indicated by:

```
cm-cod-{aws, vmware, azure} [top-level command [sub-level command]] options
```

Command Line Help

To find out which commands and options are available, the `--help` | `-h` option can be used. The following example shows how to display the top-level commands and options for the `cm-cod-aws` client:

Example

```
[root@cod-client ~]# cm-cod-aws --help
usage: cm-cod-aws [-h] [--config CONFIG] [--no-system-config] [-v]
                [--show-configuration] [--log-file LOG_FILE] [--version]
                {cluster,c,cluster create,cc,cluster list,cl,cluster
                delete,cd,cr,cremove,image,i,image
                list,il,instancetype,config} ...
```

Cluster-on-demand by Bright Computing

positional arguments:

```
{cluster,c,cluster create,cc,cluster list,cl,cluster delete,cd,cr,cremove,image,i,image list,
il,instancetype,config}
```

```

cluster (c)          Manage clusters
cluster create (cc)   Create a new cluster
cluster list (cl)     List all of the recognized clusters (VPCs)
cluster delete (cd, cr, cremove) Delete all resources in a cluster
image (i)            Manage images
image list (il)       List available Bright head node images
instancetype          Instance types
config               Configuration operations

```

optional arguments:

```

-h, --help            show this help message and exit
--config CONFIG, -c CONFIG
--no-system-config
-v, -vv, -vvv
--show-configuration
--log-file LOG_FILE
--version

```

Command Line Help At Other Levels

The `--help` option can also be used to get the available options for a specific top-level command. For example:

```
cm-cod-aws cluster --help
```

The `--help` option can also be used to get the available options for a specific sub-level command.

For example:

```
cm-cod-aws cluster create --help.
```

Command Line Parameter Explanations With `--explain`

More detailed information about a specific parameter can be obtained using the `--explain` flag.

The following example shows how to find out more about the `--nodes` parameter of the `cluster create` command of `cm-cod-aws`. The last parameter is specified as `nodes`, not `--nodes`.

Example

```

[root@cod-client ~]# cm-cod-aws cluster create --explain nodes
NAME
  nodes (cm-cod-aws cluster create --nodes <value> | -n <value>)

DESCRIPTION
  The amount of cloud nodes to configure for a cluster. The nodes are not powered on automatically.

  Its default value is 5.

NAMESPACES AND COMMANDS
  It is used in the following namespaces and commands

  -[cluster.create.nodes:nodes]
    '-[cluster.create:nodes]
      '-[aws.cluster.create:nodes]      (command: cm-cod-aws cluster create)

HOW TO CONFIGURE
  On the Command Line:
    cm-cod-aws cluster create --nodes 5 [...]

```

```
cm-cod-aws cluster create -n 5 [..]
```

As Environment Variable:

```
COD_NODES=5 cm-cod-aws cluster create [..]
```

In a Config File:

```
[cluster.create.nodes]  
nodes=5
```

The detailed information includes what the default for the parameter is (if applicable), how to specify the parameter as an environment variable, and how to include it in a configuration file.

2.8.2 Configuration Files

By default, configuration files for setting up the COD are searched for in the following locations, and in the following sequence. Settings found earlier in the sequence are overwritten by settings later on in the sequence.

- /etc/cm-cluster-on-demand.ini
- /etc/cm-cluster-on-demand.conf
- /etc/cm-cluster-on-demand.d/*
- ~/cm-cluster-on-demand.ini
- ~/cm-cluster-on-demand.conf
- ~/cm-cluster-on-demand.d/*
- a file location specified on the command line. For example, a file mycodsettings can be accessed using the --config option of the COD client:

Example

```
[fred@bright91 ~]$ cm-cod-vmware --config mycodsettings
```

Typically, the administrator sets up configuration options in one of the first 3 locations, and the regular user modifies the options or adds other options in one of the last 4 locations. The configuration files are formatted as .ini files.

Viewing The Configuration File Options

A dump of the existing configuration can be viewed using the `config dump` command, for example:

```
cm-cod-vmware config dump
```

To check what options have been applied, and their sequence, the log to STDOUT can be viewed if the `-v|--verbose` option has been applied.

A list of configuration options for a command can be seen with the `--show-configuration` option, for example (output elided):

Example

```
[fred@bright91 ~]$ cm-cod-azure cluster create --show-configuration
```

option	value
accept_eula	False (default)
access_validation	True (default)
advanced_help	False (default)
ask_to_confirm_cluster_creation	True (default)
azure_client_id	'12345678-1234-1234-1234-123456789012' (file: /etc/.../config.ini)
azure_client_secret	XXXXXXXXXXXX (file: /etc/.../config.ini) # use ... to uncover
azure_location	'westeurope' (file: /etc/cm-cluster-on-demand.d/config.ini)
azure_subscription_id	'12345678-1234-1234-1234-123456789012' (file: /etc/.../config.ini)
azure_tenant_id	'12345678-1234-1234-1234-123456789012' (file: /etc/.../config.ini)
cluster_password	None (default)

...

The table also indicates from which file a specific setting was taken.

Arguments to the `cm-cod-{aws,azure,vmware}` commands override the equivalent configuration file settings. This means that the configuration file settings of a working configuration can be used as a default template, and modifications to the template can conveniently be carried out via command line.

Setting Configuration File Options And Corresponding Arguments

The configuration files are formatted as `.ini` files, which can contain multiple sections. The `--explain` flag can be used to find out how a certain command line parameter could be defined in a configuration file. Thus in the earlier example, in the case of the `cm-cod-aws cluster create --node` parameters, defining it as a configuration file would end up with a file looking like this:

```
[cluster.create.nodes]
nodes=5
```

2.9 Preparing VMware vSphere For COD

2.9.1 Introduction

Cluster On Demand (COD) can be used to deploy virtual Bright clusters inside a VMware vSphere environment. The VMware vSphere environment is typically local. Before a vSphere environment can be used by the COD client software, the administrator needs to ensure several requirements are met. In addition, several preparation tasks must be completed. The requirements and tasks are described in the following sections.

2.9.2 Requirements

COD for VMware has been validated against VMware vSphere 6.7 Enterprise Plus. In order to run COD clusters on a vSphere environment the following requirements have to be met:

- A range of VLAN IDs should be configured in the physical switches. These are used for the internal networks of the virtual clusters.
- A virtual distributed switch must be used to host one distributed port group per virtual cluster, to serve as the internal network for that cluster.
- All virtual head nodes are connected to a pre-defined distributed port group which should provide DHCP leases and internet access.
- VMware's DRS should be enabled so that virtual nodes can automatically be distributed across the hypervisors.
- For each COD user a resource pool should be created, in order to contain the VMs of that user.

2.9.3 Networking

Each COD cluster consists of a single head node VM and a number of compute node VMs. As is usual in a Bright cluster, the head node VM has two network connections. The first connection is for the COD cluster's internal network, which connects the head node VM to the compute node VMs. The second connection is for the external connection, which is used to access the COD cluster through, for example, SSH or Bright View.

Currently, Bright Cluster Manager supports two different methods for deploying the two virtual networks:

1. vSphere method: In a vSphere deployment the head node VM is connected to two distributed port groups. These two port groups are referred to as the *internal port group* and the *external port group*.
2. NSX-T method: In an NSX-T deployment the head node VM is connected to two segments.

More details on configuring Bright for VMware on NSX-T are given in section 2.10.

When the COD client is used to create a COD cluster using vSphere networking, it expects the external port group to already exist. The internal port group is created as part of the cluster creation process. This means that the vSphere administrator should first create a virtual distributed switch which should span all the hypervisors that are to be hosting COD VMs. On that switch, the external port group should be created. The virtual head nodes should be able to obtain a DHCP lease on this port group and it should provide IP connectivity to the internet, as well as to the client machine running the COD client. Typically this means the external port group is connected to the local corporate/campus network. Outbound internet access is required in order to activate the Bright license during cluster creation.

The name of the virtual distributed switch as well as the name of the external port group should be included in the COD client configuration file as follows:

Example

```
[vmware.network]
vds switch_name = DSwitch01
externalnet_vdportgroup = DSwitch01-externalnet
```

The internal port groups, one per COD cluster, are created by the COD client as part of the cluster creation process. Each internal port group requires a unique VLAN ID. Since vSphere networking is not able to dynamically allocate VLAN IDs, a range of VLAN IDs need to be allocated. These VLAN IDs should be provisioned in the physical switches, on the ports connected to the hypervisors that are to run the COD VMs.

If multiple users should be able to create COD clusters, then the VLAN ID range should be manually divided between those users. Each user should set their allowed VLAN ID range in their COD configuration as shown in the example that follows.

Example

```
[vmware.network]
internalnet_vlan_ids = 400-499
```

The range is inclusive, so that, for example, 10-19 creates ten clusters.

2.9.4 Accounts, Privileges, Roles And Permissions

From a security point of view, there are several ways in which (groups of) users are involved in using COD for VMware. For the purpose of this documentation a few roles are defined here. These roles should not be confused with actual roles defined in vSphere, which are covered later. Also, some of the following roles might not be needed, or may overlap, depending on how an organization is using COD for VMware.

- **VMware Administrator:** The VMware administrator has full administrative access to the vSphere environment. The VMware administrator sets up the networking infrastructure required for COD (described in the previous section) as well as the COD images available to the COD Users (described in the next section).
- **COD User:** The COD User can use the COD client to create, list and remove their own COD clusters.
- **COD Cluster Administrator:** The COD cluster administrator has full root access to the virtual COD cluster. The CMDaemon running on the COD clusters head node VM has a set of VMware credentials it uses to manage (create, power on/off, delete) compute node VMs. So by extension, the COD cluster administrator has access to those VMware credentials as well.
- **End User:** The End User has non-privileged access to the COD clusters. End users are typically the ones running the actual workloads on the COD clusters. They do not have access to any VMware credentials.

Administrator Only

In the most simple form COD could be used by just the VMware administrator. In this case the VMware administrator also takes on the roles of the COD user and the COD cluster administrator. The VMware administrator uses the COD client to manage (create, list, delete) COD clusters and provides end users access to the virtual clusters to run their workloads.

In this scenario no specific VMware privileges or permissions need to be set up. By default the COD client injects the VMware credentials used to create the COD clusters into the CMDaemon that runs on the virtual head nodes. This implies that anyone with root access to the head nodes effectively has administrative access to the vSphere environment.

Multiple COD Users

A more common use case is where there are multiple COD users who each manage their own set of COD clusters. In this scenario a resource pool has to be created for each COD User. The COD Users each have their own set of VMware credentials. The permissions for the COD Users are set up in such a way that they can only manage the COD clusters in their own resource pool.

Again, by default the COD users credentials used to create the COD cluster are injected into the CMDaemon running on the COD clusters head node VM. So, anyone with root level access to those head nodes has access to the vSphere environment with the privileges granted to the COD User.

Although the COD users are limited to manage only their own VMs, the vSphere permission model is not capable of restricting the COD users to just their own networking resources. As part of the COD cluster creation, the COD client needs to create the internal port group (covered earlier) for the COD cluster. Because of this the COD users need permissions to manage port groups. Due to limitations in vSphere's permission model the COD users are able to create port groups with VLAN IDs they should not be using. They can also remove port groups that belong to other COD users' clusters.

If configured correctly, the COD client prevents COD users from doing such operations. But those restrictions cannot be enforced in vSphere. This implies that there has to be a trust between the VMware administrator and the COD users, as well as between COD users, to not abuse their networking access.

The first step for this scenario is to set up a user group called `COD-Users`, which should contain all the COD users.

Next, the following roles should be defined in the vSphere UI, with the privileges assigned accordingly:

Role name	Privileges	Action	Subaction
COD-Content-Library-Read	Content Library	Download files	
COD-Assign-Tags	vSphere Tagging	Assign/Unassign vSphere Tag	
COD-Manage-Networks	<div> <div></div> <div>dvPort group</div> <div></div> </div> <div> <div></div> <div>Network</div> <div></div> </div>	<div>Create</div> <div>Delete</div> <div>Assign network</div>	
COD-Allocate-Datastore-Space	Datastore	Allocate space	
COD-Manage-VMs	<div> <div></div> <div>Resource vApp</div> <div></div> </div> <div> <div></div> <div>Virtual machine</div> <div></div> </div>	<div>Assign VM to resource pool</div> <div>Import</div> <div>Change Configuration</div> <div>Edit Inventory</div> <div>Interaction</div>	<div>Add existing disk</div> <div>Add new disk</div> <div>Change CPU count</div> <div>Change Memory</div> <div>Change Settings</div> <div>Extend virtual disk</div> <div>Remove</div> <div>Configure CD media</div> <div>Power off</div> <div>Power on</div> <div>Reset</div>

Finally, the permissions need to be set up as detailed in the table below:

Resource	Role	User/Group	Propagate to children?
Global Permissions	COD-Content-Library-Read	COD-Users	yes
vCenter	COD-Assign-Tags	COD-Users	no
Network Folder*	COD-Manage-Networks	COD-Users	yes
Datastore	COD-Allocate-Datastore-Space	COD-Users	no
Resource pool	COD-Manage-VMs	user	yes

* The COD-Manage-Networks role should be assigned to a network folder which contains the distributed switch that hosts the COD networks. It is very important to use a network folder, even if it contains just a single switch. Assigning the role directly to the switch does not work.

Using Alternative vSphere Credentials For Head Node CMDaemon

By default the COD client injects the vSphere credentials used to create the COD cluster into the CMDaemon that runs on the virtual clusters head node. The CMDaemon makes use of those credentials to manage the compute nodes of the virtual cluster.

As described in the previous section, there are some limitations in the vSphere permissions model

when it comes to networking. It is not possible to grant the rights to create/delete port groups, which is needed to create COD clusters, but at the same time limit to which port groups VMs can be attached.

Since someone with root access to the virtual head node has access to the vSphere credentials stored in CMDaemon, that user can use those credentials to create a VM and attach it to another port group.

To mitigate this, it is possible to inject alternative credentials into the head node CMDaemon. To set up this mode of operation in the vSphere client UI, it is recommended to first create an additional user group called COD-CMDaemon. For each virtual cluster that requires isolation, a separate vSphere user should be created and added to the COD-CMDaemon group.

Next, the administrator needs to create a role that allows the new user to attach new VMs to the internal net. The administrator must also make a small modification to the COD-Manage-Networks role, adding the Modify Permission privilege. It should now look like:

Role name	Privileges	Action
COD-Assign-Network	Network	Assign network
COD-Manage-Networks	Permissions	Modify permission
	dvPort group	Create
		Delete
	Network	Assign network

The permissions for the new COD-CMDaemon group and users should be configured as detailed in the following table:

Resource	Role	User/Group	Propagate to children?
Global Permissions	COD-Content-Library-Read	COD-CMDaemon	yes
Datastore	COD-Allocate-Datastore-Space	COD-CMDaemon	no
Resource pool	COD-Manage-VMs	CMDaemon user	yes

This is in addition to the permissions defined in the previous section.

When creating the cluster, the alternative credentials for CMDaemon have to be passed to the COD client. And the client should be instructed to set up permissions that assign the COD-Assign-Network role to those credentials, on the clusters internal port group. As with any configuration option these can be put in a configuration file or passed via the command line or as environment variables. The example that follows uses the command line options:

Example

```
[alice@bright91 ~]$ cm-cod-vmware cluster create \
--cmd-vmware-username eng_cluster1_cmdaemon@vsphere.local \
--cmd-vmware-password somethingLong \
--internalnet-role COD-Assign-Network
...
```

2.9.5 COD Image Management

COD clusters are deployed using images, which are stored in the content library on the local vSphere environment. So, before any COD cluster can be created, the vSphere administrator needs to create a content library, named Bright COD Images by default.

After creating the content library, one or more COD images should be installed. The images are made available in the Bright Computing online image repository. The default repository is located on the West Coast of the United States. An alternative mirror is located in Europe. Since the images are of a considerable size, selecting the closest mirror may improve download speed. The creation time of a virtual cluster does not depend on which mirror is selected.

No extra options are required to use the default US mirror. However, to use the European mirror, the following parameter should be appended to the example commands:

```
--root-manifest https://support.brightcomputing.com/imagerepo/repo-eu.yaml
```

To list which images are available from the Bright Computing online image repository, the administrator can run the `image repo-list` command (some output removed for clarity):

Example

```
[root@cod-client ~]# cm-cod-vmware image repo-list [--root-manifest URL]
+-----+-----+-----+-----+-----+-----+-----+
| ImageID:Revision | Head(GB) | Node(GB) | Distro   | BCM Version | Created | Public |
+-----+-----+-----+-----+-----+-----+-----+
| centos7u8-9.1:1  | 6.0      | 1.57     | centos7u8 | 9.1         | 1w ago | True   |
| centos8u2-9.1:1  | 6.0      | 1.57     | centos8u2 | 9.1         | 1w ago | True   |
+-----+-----+-----+-----+-----+-----+-----+
```

To install a particular image, it can be selected via image ID, distribution and/or version. For example, to install the latest CentOS 8 update 2 image:

Example

```
[root@cod-client ~]# cm-cod-vmware image install --distro centos8u2 \
--content-library-name "Bright COD Images" [--root-manifest URL]
+-----+-----+-----+-----+-----+-----+-----+
| Image name          | Image type | Size(GB) | Distro   | BCM Version | Created   |
+-----+-----+-----+-----+-----+-----+-----+
| bcmh-centos8u2-9.1-1 | Head node  | 6.0      | centos8u2 | 9.1         | 1w 15h ago |
| bcmn-centos8u2-9.1-1 | Compute node | 1.57     | centos8u2 | 9.1         | 1w 15h ago |
+-----+-----+-----+-----+-----+-----+-----+
12:57:58: INFO: About to install these images
12:57:58: INFO: Proceed? [yes/no]
y
12:58:03: INFO: Creating image bcmn-centos8u2-9.1-1 with URL https://cod-images.s3-us-west-
1.amazonaws.com/bcmn-centos8u2-9.1-1.ova
13:01:38: INFO: Tagging image
13:02:06: INFO: Creating image bcmh-centos8u2-9.1-1 with URL https://cod-images.s3-us-west-
1.amazonaws.com/bcmh-centos8u2-9.1-1.ova
13:15:48: INFO: Tagging image
```

Images installed in the local content library can be listed with the `image list` command (some output removed):

Example

```
[root@cod-client ~]# cm-cod-vmware image list
+-----+-----+-----+-----+-----+-----+-----+
| ImageID:Revision | Head(GB) | Node(GB) | Distro   | BCM Version | Created   | Public |
+-----+-----+-----+-----+-----+-----+-----+
| centos8u2-9.1:1  | 6.0      | 1.57     | centos8u2 | 9.1         | 1w 15h ago | True   |
+-----+-----+-----+-----+-----+-----+-----+
```

By default the `image list` command displays only the latest revision of each distribution/version combination. To include older revisions, the `--all-revisions` flag can be added:

Example

```
[root@cod-client ~]# cm-cod-vmware image list --all-revisions
```

ImageID:Revision	Head(GB)	Node(GB)	Distro	BCM Version	Created	Public
centos8u2-9.1:1	6.0	1.57	centos8u2	9.1	1w 15h ago	True
centos8u2-9.1:2	6.0	1.57	centos8u2	9.1	1d 15h ago	True

To remove an image, it is recommended to specify the image ID for the image:

Example

```
[root@cod-client ~]# cm-cod-vmware image delete centos8u2-9.1:1
13:45:30:      INFO: Finding images...
```

Image name	Image type	Size(GB)	Distro	BCM Version	Created
bcmh-centos8u2-9.1-1	Head node	6.0	centos8u2	9.1	1w 15h ago
bcmn-centos8u2-9.1-1	Compute node	1.57	centos8u2	9.1	1w 15h ago

```
13:45:34:      INFO: About to delete these images
13:45:34:      INFO: Proceed? [yes/no]
y
13:45:51:      INFO: Done
```

2.10 Using NSX-T To Manage COD VMware Networks

Each COD cluster consists of a single head node VM and a number of compute node VMs. As is usual in a Bright cluster, the head node VM has two network connections.

- The first connection is for the COD cluster's internal network, which connects the head node VM to the compute node VMs.
- The second connection is for the external connection, which is used to access the COD cluster through, for example, SSH or Bright View.

In an NSX-T deployment the internal and external networks are created using NSX-T segments. The following subsections discuss how to configure NSX-T, vSphere, and the COD client for NSX-T networking.

2.10.1 Requirements

Bright for VMware has been validated against VMware vSphere 6.7 Enterprise Plus and VMware NSX-T Data Center 3.1. In order to run COD clusters in a vSphere environment using NSX-T networking, the following requirements must be met:

- A vSphere environment, as discussed in section 2.9.
- An NSX-T Data Center deployment containing an overlay transport zone that contains the hypervisors from the vSphere environment.
- An edge deployment, to provide external connectivity to the COD cluster head nodes.

2.10.2 Configuring NSX-T

When COD VMware is used to deploy clusters with NSX-T networking, the COD client expects the external network segment to exist in the overlay transport zone. The external network segment is used to access the head node of COD clusters. This means that the network segment needs to be part of

a routed network. More information on how to create and configure routed segments is available in the VMware NSX-T Data Center Administration Guide, available from <https://docs.vmware.com/en/VMware-NSX-T-Data-Center/index.html>

2.10.3 Configuring vSphere

Configuring vSphere is covered in section 2.9. If all COD VMware clusters are to be deployed using NSX-T networks then section 2.9.3 may be skipped. However, in case of either a mix of vSphere and NSX-T clusters, or a combination of vSphere port groups and NSX-T segments in a single COD cluster, both vSphere and NSX-T networking must be configured.

If there are multiple COD users defined, as described in section 2.9.4, then COD-Users need to have the COD-Assign-Network role set up on the data center level. This permission is needed so that the COD users can connect the VMs to the segments which are deployed in the data center. This means that in addition to the roles and permissions in 2.9.4, the permissions should be set up as detailed in the following table:

Resource	Role	User/Group	Propagate to children?
Datacenter	COD-Assign-Network	COD-Users	yes

2.10.4 Configuration And Use Of The COD Client

Before deploying a cluster with NSX-T networking, the COD client must be reconfigured. The recommended approach is to add this to the COD client configuration file as follows:

Example

```
[vmware.network]
externalnet_network = externalnet-segment
transport_zone = aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaaa

[vmware.credentials]
nsx_address = nsx-manager.mycompany.com
nsx_username = admin
nsx_password = AdminPassword!
```

The configuration is added to the already existing configuration as discussed in sections 2.7.1 and 2.9.3. The transport zone ID in the NSX-T manager can be found via the clickpath System→Fabric→Transport Zones which displays the overlay transport zone.

A cluster with NSX-T segments for the internal and external network can be created by adding the following additional command line options to the `cm-cod-vmware` command:

Example

```
[alice@bright91 ~]$ cm-cod-vmware cluster create \
--internalnet-network-backend nsx \
--externalnet-network-backend nsx
...
```

2.11 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 2.11.5). This section (section 2.11) 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.

2.11.1 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).

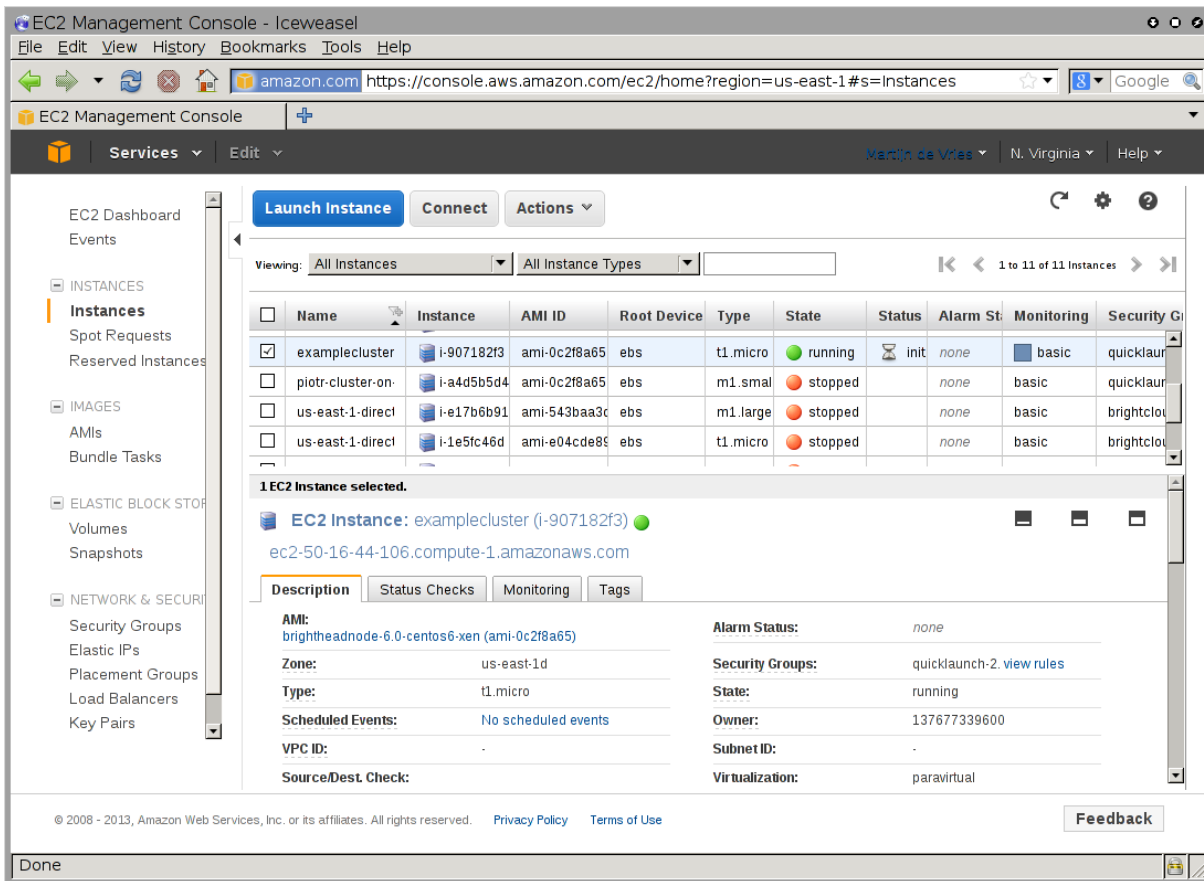


Figure 2.6: The EC2 Instances List

System (Amazon machine infrastructure) and instance (instance running under the infrastructure) reachabilities are similarly shown under the neighboring “Status Checks” tab (figure 2.7).

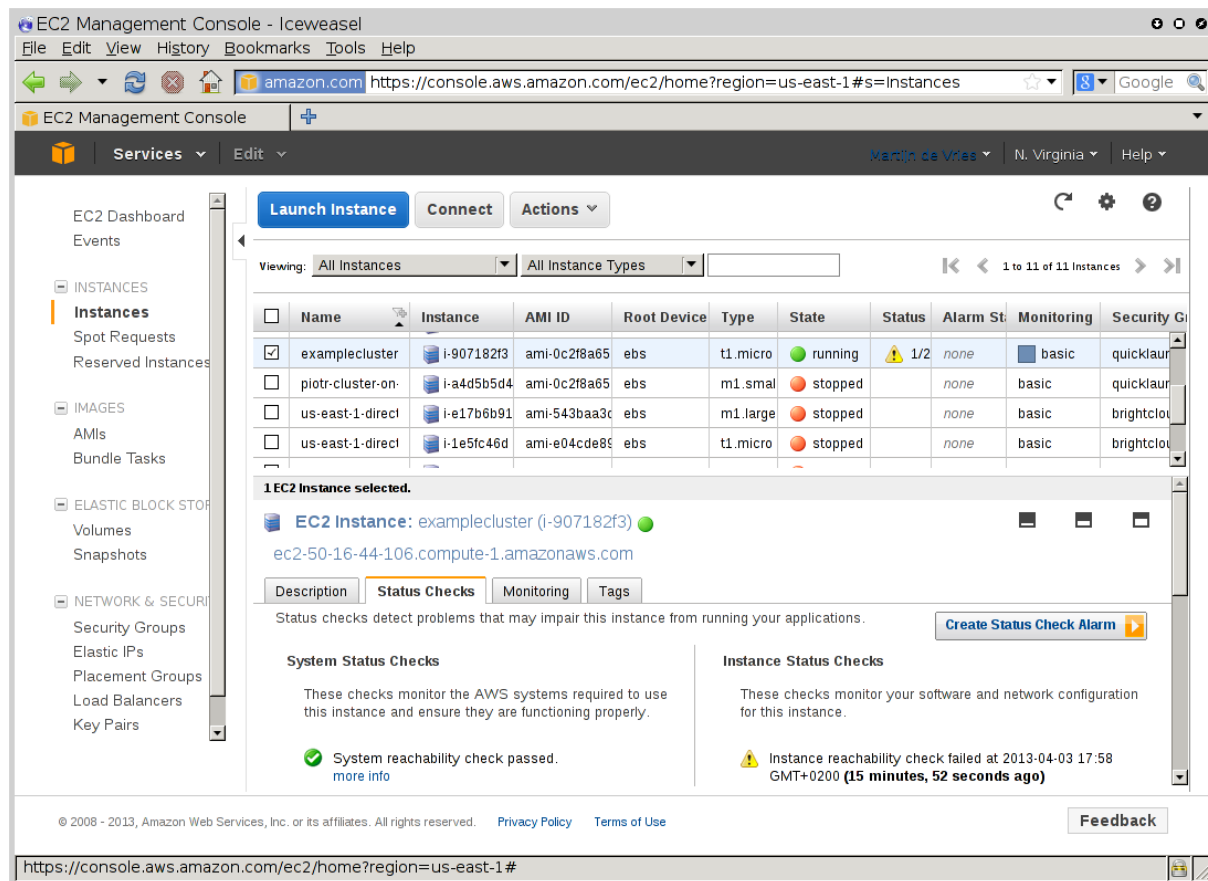


Figure 2.7: Reachability Status For An EC2 Instance

2.11.2 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.

2.11.3 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 (section 2.6.2). 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.

2.11.4 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.

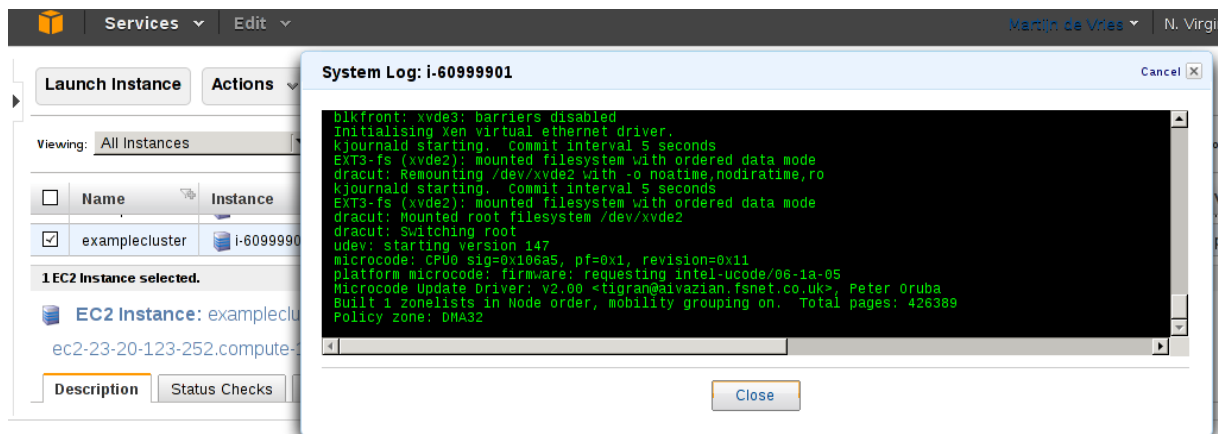


Figure 2.8: System Log Of The Checkboxed Instance

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

2.11.5 Security Group Configuration To Allow Access To The Head Node 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).

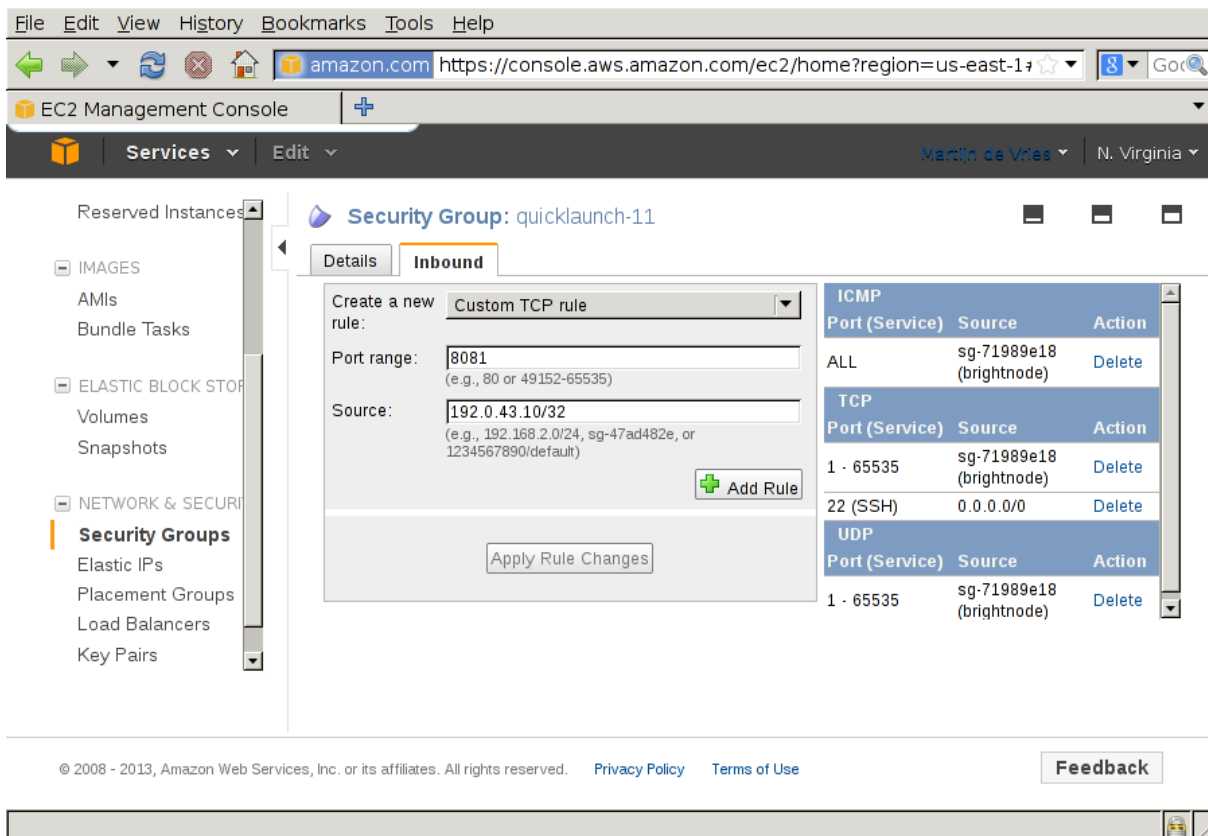


Figure 2.9: Security Group Network And Port Access Restriction

- 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.

2.12 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.

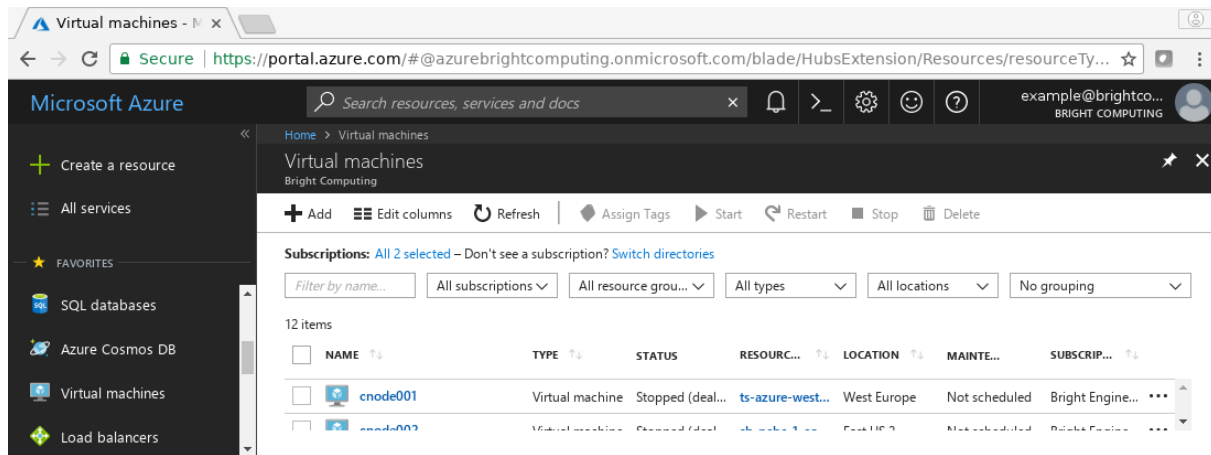


Figure 2.10: Viewing Virtual Machines In A Cluster On Demand In Azure

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.

2.13 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:

Devices→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:

Devices→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 4.2.3 of the *Administrator Manual*).

2.13.1 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.

2.14 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). This is described in Chapters 3 and 4 of the *Cloudbursting Manual*.
- Azure (CX-Azure). This is described in Chapter 5 of the *Cloudbursting Manual*.
- OpenStack (CX-OS). This is described in Chapter 7.

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@bright91 ~]# cmsh -c "main; licenseinfo"
```

```
License Information
```

```
-----
Licensee                /C=US/ST=NY/L=WS/O=Bright
                        Mordor/OU=Mt. Doom/CN=Bright 9.1 Cluster
Serial Number           1210883
Start Time               Mon Oct 16 01:00:00 2017
End Time                 Fri Dec 31 23:59:00 2038
Version                  7.0 and above
Edition                  Advanced
Type                     Commercial
Pre-paid Nodes           100
Pay-per-use Nodes        yes
Data Science Nodes       80
Node Count                6
Allow edge sites         Yes
Accelerator Node Count    0
MAC Address / Cloud ID   FA:16:3E:DE:A4:35
```

The value of End Time, Pre-paid Nodes, and Pay-per-use Nodes in the preceding license should be checked. Pre-paid nodes are nodes that may be started without incurring any additional charges. Pay-per-use nodes (section 8.5) are nodes that incur additional charges when started. Both pre-paid and pay-per-use can be on-premises or off-premises. That is, both kinds can be used for cluster extension cloudbursting.

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*.

- **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 795 of the *Administrator Manual*), which is a `CMDaemon` directive that can be set and activated (page 777 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”.

Cluster Extension Cloudbursting Logging

All AWS logging goes to the CMDaemon logs in `/var/log/cmdaemon`. The CLOUD tag in the log is used to indicate cloud-related operations.

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 clickpath 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.

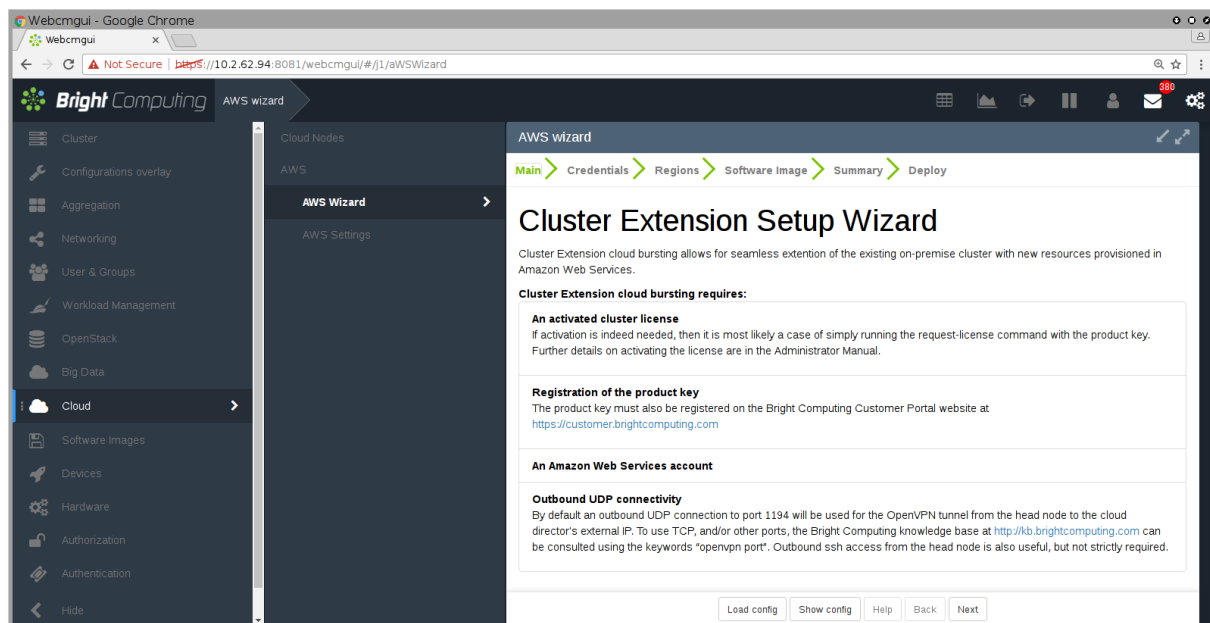


Figure 3.1: Introduction Page For Cluster Extension With Bright View

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:
amazon2

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 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).

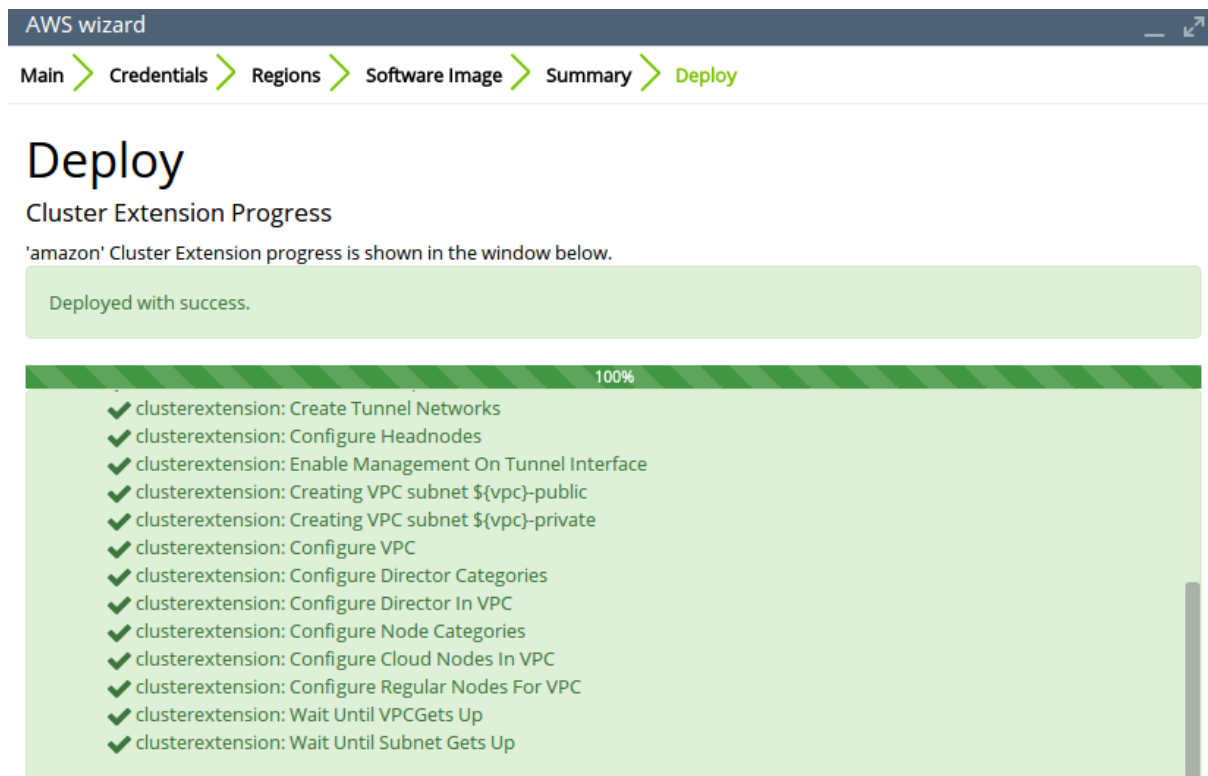


Figure 3.6: Cluster Extension Configuration Processing With Bright View

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 13.2.11 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:

Devices→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:

Devices→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:

```
Devices→Cloud Nodes→<cloud director hostname>→Edit→Settings→Cloud settings→
STORAGE→Storage→<storage type>→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

```
[bright91]% device use us-east-1-director
[bright91->device[us-east-1-director]]% cloudsettings
[bright91->device[us-east-1-director]->cloudsettings]% storage
[bright91->...->cloudsettings->storage]% list
Type          Name (key)    Drive    Size    Volume ID
-----
ebs            ebs          sdb      42GB
ephemeral      ephemeral0    sdc      0B      ephemeral0
[bright91->...->cloudsettings->storage]% remove ebs
[bright91->...->cloudsettings*->storage*]% set ephemeral0 drive sdb
[bright91->...->cloudsettings*->storage*]% list
Type          Name (key)    Drive    Size    Volume ID
-----
ephemeral      ephemeral0    sdb      0B      ephemeral0
[bright91->...->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
Devices→Cloud Nodes→<cloud director>→Edit→Settings→Cloud settings→Instance ID
- with cmlsh, by running something like:

Example

```
[bright91]% device use us-east-1-director
[bright91->device[us-east-1-director]]% get cloudid
i-f98e7441
[bright91->device[us-east-1-director]]% cloudsettings
[bright91->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 Devices→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@bright91 ~]# 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@bright91 ~]# 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@bright91 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):      4
Source node:        bright91
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@bright91 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):      5
Source node:        bright91
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 163 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:

Device→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).

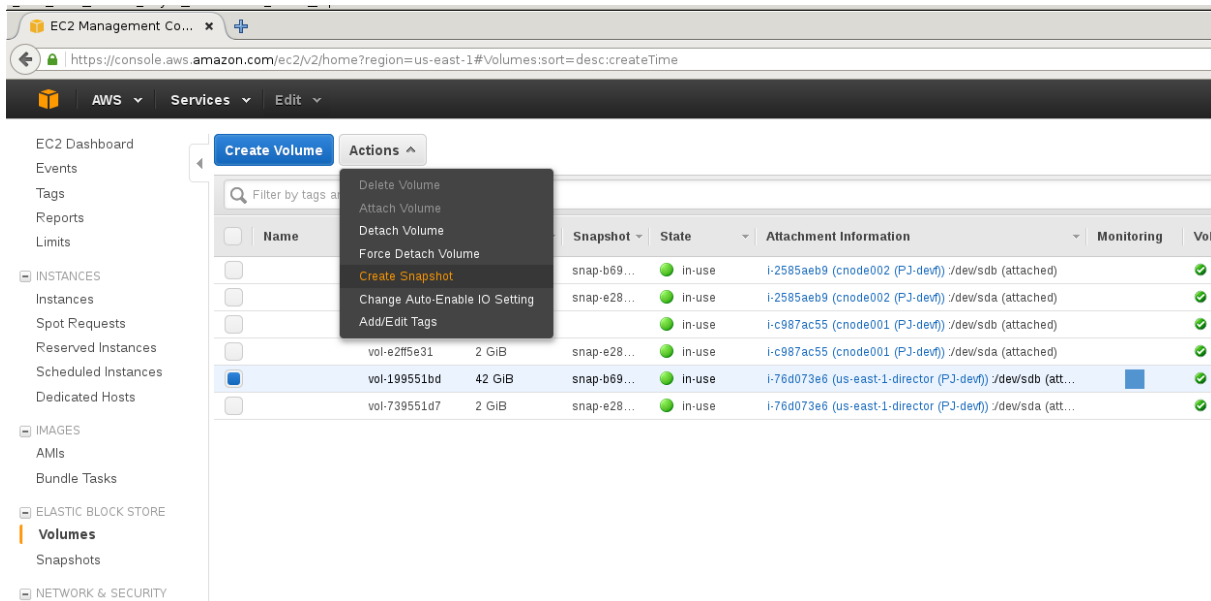


Figure 3.7: Creating A Snapshot From A Selected Volume

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

```
[bright91->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@bright91 ~]# cmsh
[bright91]% device
[bright91->device]% use us-east-1-director
[bright91->device[us-east-1-director]]% cloudsettings
[bright91->...-director]->cloudsettings% storage
[bright91->...-director]->cloudsettings->storage% use ebs
[bright91->...-director]->cloudsettings->storage[ebs]% set snapshotid snap-2a96d0c6
```

The cloud director can now be powered on:

Example

```
[bright91->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

```
[bright91->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

```
[bright91->device[cnode001]->cloudsettings->storage[ebs]]% set volumetype io1
```

- In Bright View, the equivalent clickpath is:

```
Devices->Cloud Nodes-><cloud node hostname>->Edit->Settings->Cloud settings->STORAGE->
Storage->ebs->CREATION TIME->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

```
[bright91->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@bright91 ~]# 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 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 57), or as an `Ncurses` dialog (page 59).

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` or `--help` option:

```
[root@bright91 ~]# 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]
        [--on-error-action {debug,remotedebug,undo,abort}]
        [--output-remote-execution-runner]
        [--json] [--no-distro-checks]
        [--min-reboot-timeout <reboot_timeout_seconds>]
        [--remove]
        [--terminate-instances]
        [--remove-fsx-instances]
        [--force]
```

```

[--yes-i-know-this-is-dangerous-for-this-cluster <headnode_hostname>]
[--test-networking]
[--test-environment]
[--test-configuration]
[--test-everything]
[--enable-external-network-connectivity]

```

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 plugins (cloudstorage, clusterextension),
    except for these. Comma-separated list.
--only-these-modules <mod1,mod2,...>
    Out of all default plugins, load and use only these. Comma-separated
    list.
--dev            Enables additional command line arguments
--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           Use json formatting for log lines printed to stdout
--no-distro-checks
    Disable distribution checks based on ds.json
--min-reboot-timeout <reboot_timeout_seconds>
    How long to wait for nodes to finish reboot (default and minimum allowed:
    300 seconds).

```

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
--terminate-instances
    Terminate all non-terminated VMs.
--remove-fsx-instances
    Remove FSX instances.
--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.

```

cluster extension to Bright OpenStack:
Options specific to this cloud type.

```

--enable-external-network-connectivity
    Enable external network connectivity for cloud nodes. Note that this
    disables security groups for the cloud director, which may be a security
    risk. Requires Neutron Port Security extension driver.

```

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@bright91 ~]# 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

Cluster extension cloudbursting deployment can be carried out by selecting AWS (described in this chapter) or Azure (Chapter 5) from the main screen.

After AWS is selected, 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):

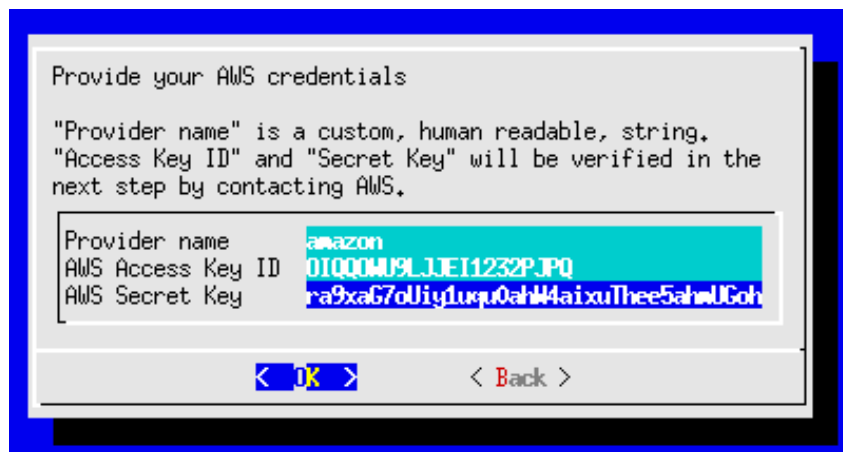


Figure 4.2: Configuration Processing With cm-cluster-extension: Obtaining Credentials

To paste the credentials from the clipboard, the cluster administrator may find it helpful to know that a paste to Ncurses can usually be carried out with a <shift><right click>.

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.

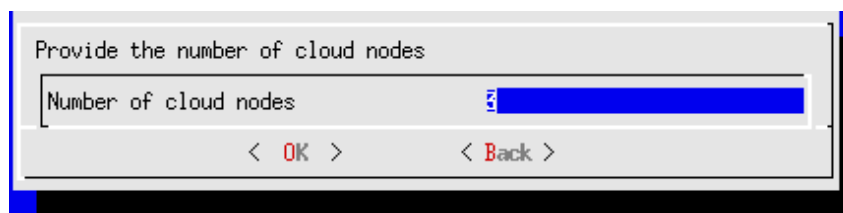


Figure 4.3: Configuration Processing With cm-cluster-extension: Setting The Initial Number Of Cloud Nodes

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

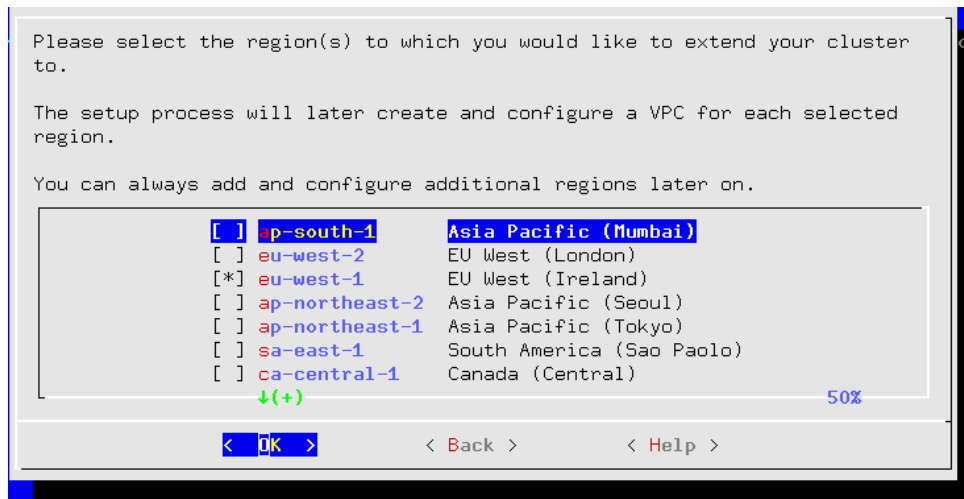


Figure 4.4: Configuration Processing With cm-cluster-extension: Regions Selection

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):

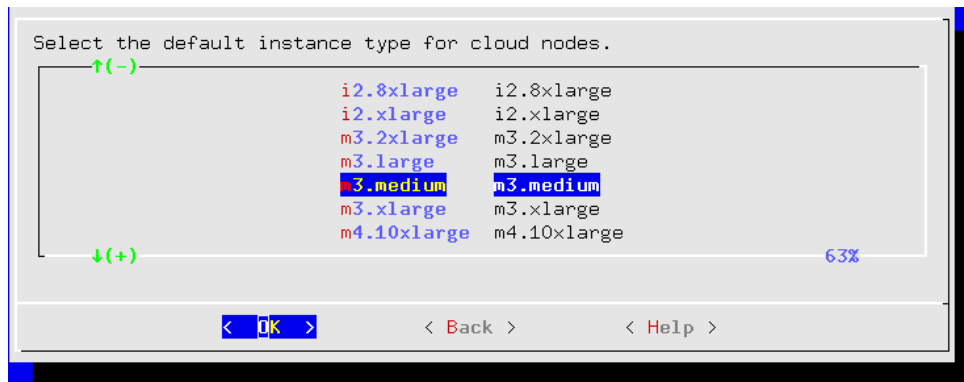


Figure 4.5: Configuration Processing With cm-cluster-extension: Default Instance Type

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:

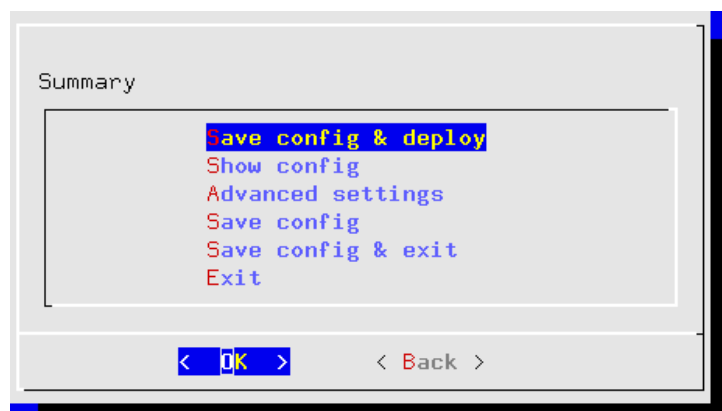


Figure 4.6: Configuration Processing With cm-cluster-extension: Summary Screen

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@bright91 ~]# 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 61). 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:

```
[bright91->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 59, 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

```
[bright91->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
...
[bright91->device*]% commit
Successfully committed 9 Devices
[bright91->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

```
[bright91->device]% power on -n eu-west-1-cnode0[02-10]
```

4.3 Submitting Jobs With `cmjob` And Cloud Storage Nodes, For Cluster Extension Clusters

The `cmjob` 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 `cmjob` command is available from the Bright Cluster Manager repository as part of the `cmdaemon-cmjob` package. The `cmjob` command needs the `cmjob` 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 `cmjob` users.

Example

```
[root@bright91 ~]# cmsh
[bright91]% user use henry
[bright91->user[henry]]% set profile cloudjob; commit
```

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

1. The `cmdaemon-cmjob` 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@bright91 ~]# yum install cmdaemon-cmjob
[...]
[root@bright91 ~]# yum --installroot /cm/images/default-image install cmdaemon-cmjob
[...]
```

2. The `cm-cloud-storage-setup` utility is run. Example runs are provided later, starting on page 65, 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 `cmjob` properties
- creates
 - *templates for cloud storage nodes*
 - *storage policies for cmjob*

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 `cmjob` Properties With `cm-cloud-storage`

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

```
[root@bright91 ~]# cm-cloud-storage-setup -h
Please wait...
usage: Cloud storage setup cm-cloud-storage-setup [-v] [-h] [-c <config_file>]
                                         [--dev]
                                         [--on-error-action
                                         {debug,remotedebug,undo,abort}]
                                         [--output-remote-execution-runner]
                                         [--json]
                                         [--no-distro-checks]
                                         [--min-reboot-timeout <reboot_timeout_seconds>]
                                         [--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
--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               Use json formatting for log lines printed to stdout
--no-distro-checks
--min-reboot-timeout <reboot_timeout_seconds>
                    How long to wait for nodes to finish reboot (default and minimum allowed:
                    300 seconds).
```

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):

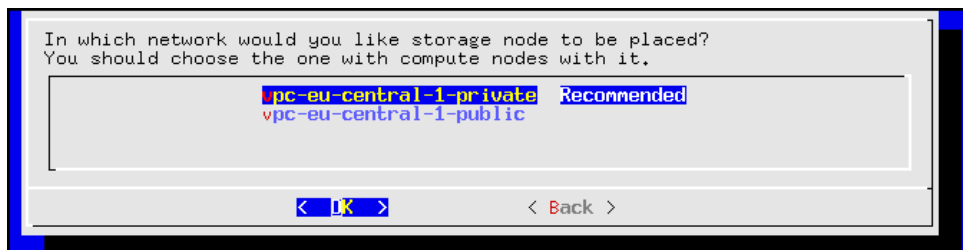


Figure 4.8: Cloud Storage Configuration Processing With `cm-cloud-storage`: Network Selection

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

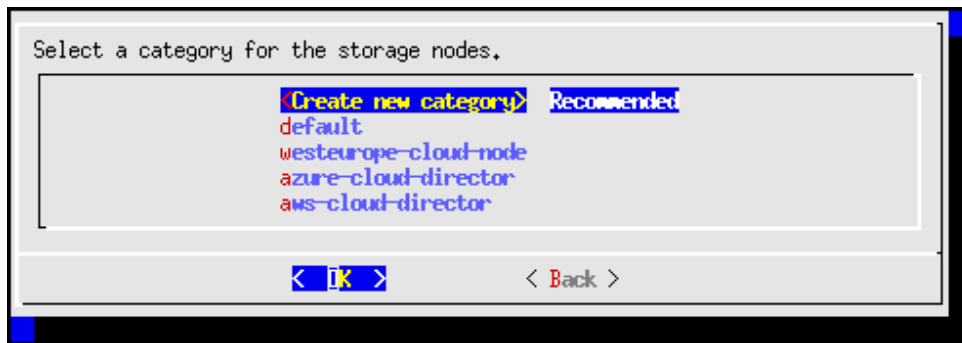


Figure 4.9: Cloud Storage Configuration Processing With cm-cloud-storage: Category Selection

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

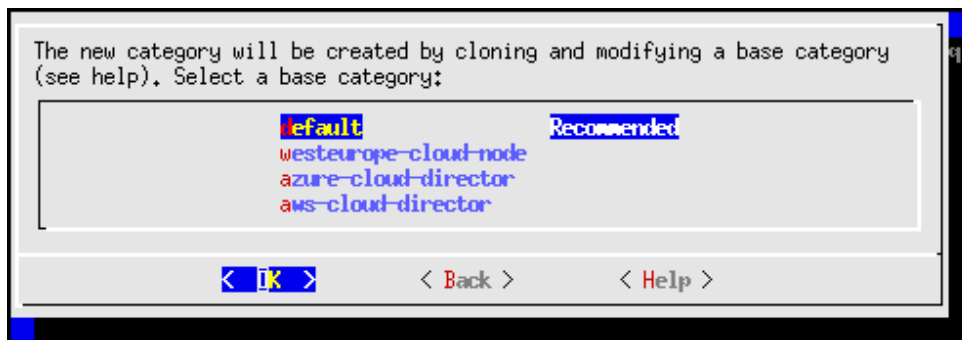


Figure 4.10: Cloud Storage Configuration Processing With cm-cloud-storage: Category Creation

A bucket name can be set (figure 4.11):

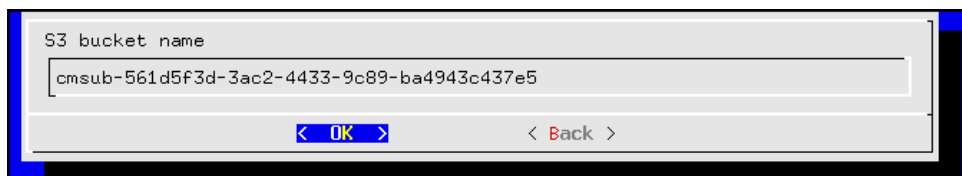


Figure 4.11: Cloud Storage Configuration Processing With cm-cloud-storage: S3 Bucket Entry

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):

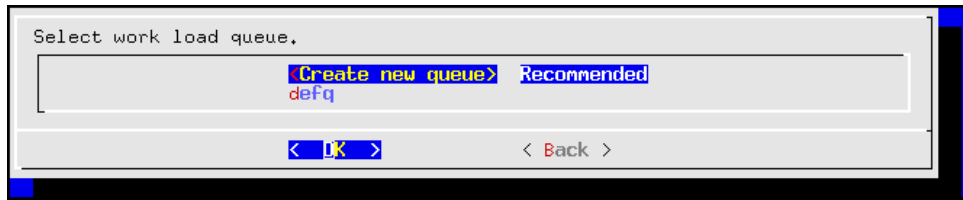


Figure 4.13: Cloud Storage Configuration Processing With `cm-cloud-storage`: Queue Creation

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

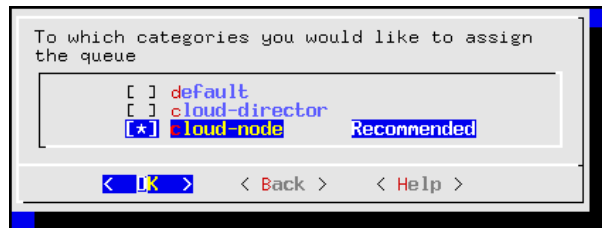


Figure 4.14: Cloud Storage Configuration Processing With `cm-cloud-storage`: Setting The Queue Category

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

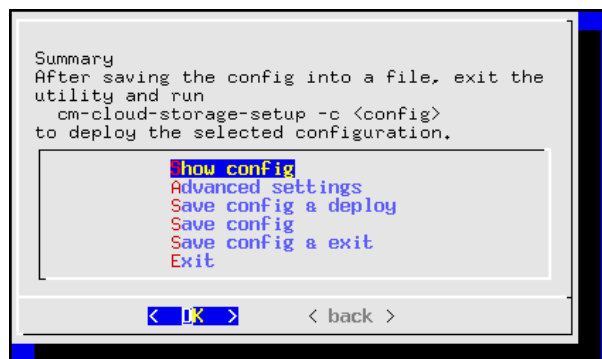


Figure 4.15: Cloud Storage Configuration Processing With `cm-cloud-storage`: Summary Screen

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@bright91 ~]# cm-cloud-storage-setup -c cm-cloud-storage-setup.conf
```

Configuring cmjob Properties With cmsh

The storage node policy settings can be modified via CMDaemon using cmsh:

```
[root@bright91 ~]# cmsh
[bright91]% cmjob
[bright91->cmjob[cmjob]]% storagenodepolicies
[bright91->cmjob[cmjob]->storagenodepolicies]% list
Name (key)          Template for cloud nodes
-----
us-east-1-policy    us-east-1-storage-template
[bright91->cmjob[cmjob]->storagenodepolicies]% use us-east-1-policy
[bright91->cmjob[cmjob]->storagenodepolicies[us-east-1-policy]]% show
Parameter          Value
-----
Default job output size    30.0GiB
Exported directories      home
Intermediate storage      <submode>
Max download time         2h
Max jobs per node         10
Max storage nodes         5
Max upload time           2h
Minimum Storage volume size 30.0GiB
Name                     us-east-1-policy
Revision
Storage node idle time limit 15m
Storage node name prefix   us-east-1-cstorage
Storage volume filesystem  ext3
Template for cloud nodes   us-east-1-storage-template
Tunnel IP start address    0.0.100.0
[bright91->cmjob[cmjob]->storagenodepolicies[us-east-1-policy]]% intermediatestorage
[bright91->cmjob[cmjob]->storagenodepolicies[us-east-1-policy]->intermediatestorage]% show
Parameter          Value
-----
Bucket name        bright911b86bdc32219494f9a21
Max object lifetime 4w 2d
Region              us-east-1
Revision
Type                AWSIntermediateStorage
SecGroupFSx         sg-0f1fe4fc6d3d49e66
Default FSx Instance Capacity (GiB) 1200
Max FSx Instance Capacity (GiB) 7200
Max FSx Instance Count Per User 5
[bright91->cmjob[cmjob]->storagenodepolicies[us-east-1-policy]->intermediatestorage]%
```

By default the S3 bucket storage expires after 30 days.

Cloud jobs running under cmjob are listed under the cloudjobs submode:

```
[root@bright91 ~]# cmsh
[bright91]% cmjob
[bright91->cmjob[cmjob]]% cloudjobs
[bright91->cmjob[cmjob]->cloudjobs]% list
Name (key)  jobStatus          storageNode          user
-----
slurm-35    Error: Job failed. Exit code: 0. Status: FAILED  us-east-1-cstorage001  fred
slurm-36    Error: Failed to acquire storage node: Failed t+  us-east-1-cstorage001  fred
slurm-37    Running...
```

4.3.2 Integration Of `cmjob` With AWS FSx For Lustre

FSx for Lustre is a high-performance highly scalable distributed file system that is available as a cloud service within some AWS regions. A Bright cluster that has burst into such a region can use FSx for Lustre to create a shared storage volume that can be read from, and can be written to, by many compute nodes simultaneously.

On a cluster that has burst into such a region, `cmjob` can provide workload managers with an FSx for Lustre instance to store input and output data.

Sequential workload manager jobs can then continue to use the same FSx for Lustre instance. This means that the output of one job can immediately become the input of the next one without having to transmit the data to and from an S3 bucket, or to and from the head node.

Additionally, workload manager jobs that run on multiple compute nodes in parallel can obtain their shared input from the same FSx for Lustre instance.

Providing shared storage to cloud compute nodes via FSx for Lustre is an alternative to providing it via cloud storage nodes (section 4.3). The storage node approach is not sufficiently scalable when there are more than several dozen compute nodes working simultaneously with the same storage node. FSx for Lustre is significantly more scalable and far more performant than storage nodes—although at a higher cost.

Bright Cluster Manager currently offers three different methods of integrating `cmjob` with FSx for Lustre:

1. *On-demand FSx*: An FSx file system is created on a per-job basis and is deleted automatically when the job completes.
2. *User-managed FSx*: A user creates an FSx instance and then submits one or more jobs that use that same FSx instance. These jobs can execute in sequence or simultaneously. When the last job has finished, the user is responsible for deleting the FSx instance.
3. *Admin-managed FSx*: An administrator creates an FSx instance and shares it with one or more users. These users are able to use the instance in the same way as with User-managed FSx, but they cannot delete the instance or see each other's files.

Prerequisites

To be able to use the integration of `cmjob` with FSx for Lustre, the following requirements need to be met:

- The cluster must be bursting into an AWS region that supports FSx for Lustre. AWS does not offer this feature in all regions. The AWS documentation at <https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/> describes the regions that support FSx for Lustre.
- AWS needs to offer Lustre installation packages for the operating system and kernel that is used to extend into AWS. While the necessary Lustre package could perhaps be obtained directly from the Lustre project, it cannot be guaranteed that these packages are compatible with the FSx for Lustre file systems that are offered by AWS. The AWS packages are built and tested to work only with specific kernel versions.
- At the time of writing (May 2020), the supported operating systems are CentOS and Redhat 7.5, 7.6, 7.7. A list of supported operating systems as well as the packages for installing the Lustre client for such systems can be found at <https://docs.aws.amazon.com/fsx/latest/LustreGuide/install-lustre-client.html>.

Enabling `cmjob` integration with FSx for Lustre:

The integration of `cmjob` with FSx for Lustre can be accomplished in two ways, either:

- through the cm-cloud-storage-setup wizard (section 4.3.1)
- or
- manually

The cm-cloud-storage-setup wizard can be used if the preceding prerequisites have been met. In any case, during the wizard interaction, the administrator is asked to choose if FSx support should be enabled. If it is chosen, then:

- If FSx support is not available, then the administrator is informed that one of the requirements was not met.
- If FSx support is available, then no further action is required from the administrator.

After the deployment has finished, all users belonging to the cloudjob profile are able to create user-managed and on-demand FSx volumes through cmjob.

Enabling cmjob integration manually: The manual integration approach should be followed when cmjob has already been configured on a cluster. That is, if the cm-cloud-storage-setup tool has already been run but the cluster has been running without FSx support so far.

Two separate actions need to be executed to manually add FSx for Lustre integration:

1. Installation of the required packages to the images

The AWS documentation should mention the packages that are required to install the Lustre Client. Often these two packages are needed: kmod-lustre-client and lustre-client. Care must be taken to match the package version to the Linux kernel version of the images. The packages must be installed to the images under /cm/images that are used by the director and by all compute nodes within that cluster extension.

Example

At the time of writing (May 2020), for a CentOS 7.7 system, with a 3.10 kernel, the instructions at <https://docs.aws.amazon.com/fsx/latest/LustreGuide/install-lustre-client.html> suggested the following steps:

```
[root@bright91 ~]# cm-chroot-sw-img /cm/images/<image used by director and compute nodes>
...
[root@bright91 /]# wget https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/\
fsx-rpm-public-key.asc -O /tmp/fsx-rpm-public-key.asc
...
[root@bright91 /]# wget https://fsx-lustre-client-repo.s3.amazonaws.com/el/7/\
fsx-lustre-client.repo -O /etc/yum.repos.d/aws-fsx.repo
[root@bright91 /]# rpm --import /tmp/fsx-rpm-public-key.asc
...
[root@bright91 /]# vi /etc/yum.repos.d/aws-fsx.repo

    change references from el/7/ to el/7.7/ and exit out of editor

[root@bright91 /]# yum clean all
...
[root@bright91 /]# yum install -y kmod-lustre-client lustre-client
...
[root@bright91 /]# exit
```

2. Addition of the FSx tokens to the user profiles

Profiles (section 6.4 of the *Administrator Manual*) can be set for FSx users. Every user that should be able to create FSx for Lustre instances should have one or both of the following FSx tokens appended to their profile:

- ON_DEMAND_FSX_TOKEN
- USER_MANAGED_FSX_TOKEN.

Example

A user, fred is already using cluster extension with AWS, but without FSx. Now the user would like to launch and manage FSx storage on demand.

The existing profile of fred can be cloned in profile mode. The tokens needed can be appended to the new profile, and the modified profile can then be set for fred. The following `cmsh` session suggests how this can be carried out:

```
[bright91->user[fred]]% get profile
cloudjob
[bright91->user[fred]]% profile
[bright91->profile]% clone cloudjob cloudjobfsx
[bright91->profile*[cloudjobfsx*]]% get tokens
SUBMIT_CLOUD_JOB_DESCRIPTION_TOKEN
GET_CLOUD_JOB_DESCRIPTION_TOKEN
[bright91->profile*[cloudjobfsx*]]% append tokens ON_DEMAND_FSX_TOKEN USER_MANAGED_FSX_TOKEN
[bright91->profile*[cloudjobfsx*]]% get tokens
SUBMIT_CLOUD_JOB_DESCRIPTION_TOKEN
GET_CLOUD_JOB_DESCRIPTION_TOKEN
ON_DEMAND_FSX_TOKEN
USER_MANAGED_FSX_TOKEN
[bright91->profile*[cloudjobfsx*]]% commit
[bright91->profile[cloudjobfsx*]]% user use fred
[bright91->user[fred]]% set profile cloudjobfsx
[bright91->user*[fred*]]% commit
```

When the two preceding actions have been carried out, the director and all compute nodes should be rebooted. A user should then be able to create and submit jobs with a user-managed FSx instance, or submit jobs with an OnDemand FSx instance, depending on which tokens were assigned to the profile of the user.

Configuration

Through `cmsh`, an administrator can set profile tokens (section 6.4 of the *Administrator Manual*) to control how users can work with FSx for Lustre. An administrator can also set quotas that control how many FSx instances a user can have simultaneously, and how large these FSx instances can be. The aim of this is to help manage costs.

Profile tokens: The following profile tokens are available for user management in FSx for Lustre:

- ON_DEMAND_FSX_TOKEN: allows a user to use the `--on-demand-fsx` flag when submitting jobs with `cmjob`.
- USER_MANAGED_FSX_TOKEN: allows a user to create, manage and delete FSx instances.
- DELETE_ANY_FSX_INSTANCE_TOKEN: allows a user to delete FSx instances that are owned by other users. Normally reserved for root only.

- `LIST_ALL_FSX_INSTANCES_TOKEN`: allows a user to see FSx instances owned by other users that are not shared with that user. Normally reserved for root only.
- `SHARE_FSX_INSTANCE_TOKEN`: allows a user to share FSx instances with other users. Only the owner is allowed to share it. Normally reserved for root only.

Settings: The following `cmsh` navigation path:

```
cmsh→cmjob→storagenodepolicies→use <policy>→intermediatestorage
```

leads to these storage parameters:

- Max FSx Instance Capacity (GiB)
- Max FSx Instance Count Per User

which control quota limits for FSx instances.

Each quota affects both on-demand and user-managed FSx instances. So if a user has already reached the maximum number of user-managed FSx instances, then the user cannot submit a job that uses an OnDemand FSx instance.

The same navigation path also leads to the storage parameter:

- Default FSx Instance Capacity (GiB)

which controls the default capacity when a user or admin creates a user-managed or admin-managed FSx instance.

Usage

As listed earlier, the three methods that Bright Cluster Manager provides for `cmjob` integration with FSx for Lustre are:

1. on-demand FSx,
2. user-managed FSx, and
3. admin-managed FSx.

How to run these methods using `cmjob` is covered in the next sections.

Using on-demand FSx: When using `cmjob` to submit a job, a user can now choose whether to use EBS-backed storage nodes or an FSx for Lustre file system to host the input data and output data. For a user there is little difference between how two shared storage options are used, besides their performance.

The steps are:

1. an EBS volume or FSx for Lustre file system is created as shared storage,
2. the input data is copied from the head node to this shared storage
3. all compute nodes chosen by the workload manager mount this shared storage, and then run their jobs,
4. storing their output data on the same shared storage.
5. the output data is uploaded from the shared storage to the head node
6. the EBS volume or FSx for Lustre file system is deleted.

Submitting a job to run with storage nodes is the default.

If a user wants a job to use FSx for Lustre, then the user must set the `--on-demand-fsx` flag when submitting the job:

Example

```
cmjob submit --on-demand-fsx job.sh
```

The capacity of the on-demand FSx instance is determined by doubling the size of the input data. This calculation takes into account both the data uploaded from the head node, and labeled data downloaded from an S3 bucket. If the user thinks that the calculated capacity is not large enough to hold the output data (the output includes temporary data), then the capacity can be increased with the `--expected-output-size` option. The specified size is combined with the size of the input data and rounded up to the nearest valid FSx volume size. The option also works for EBS storage.

Using user-managed FSx: `cmjob` with a user-managed FSx file system operates differently from an on-demand system. With user-managed FSx, the user is responsible for creating and deleting the FSx file system.

In some cases this can be more expensive than using the on-demand approach, especially if the user forgets to delete an FSx instance after no longer needing it.

However with user-managed FSx a user can run multiple concurrent jobs with the same FSx file system. It is also possible to run multiple jobs in sequence, each using the output of the previous job as input, without having the overhead of copying files over from the FSx file system to an S3 bucket, and vice versa. Depending on the size of the workload, this option may be cheaper, or more desirable, than on-demand FSx.

To create an FSx instance, the user executes `cmjob` with the `create-fs-instance` option as follows:

Example

```
cmjob create-fsx-instance <name> --capacity <capacity>
```

`<name>` can be a sequence of alphanumeric characters and dashes, and `<capacity>` is the total size of the FSx file system in gigabytes and is ideally set to a multiple of 1200, to prevent rejection by AWS. If not specified, the default value is used. The name does not have to be unique.

To delete an FSx instance, the user can run `cmjob` with the `delete-fsx-instance` option:

Example

```
cmjob delete-fsx-instance <name or FSx ID>
```

If the name is not unique, then the user can always specify the FSx ID. The ID can be found using the `list-fsx-instances` option. This lists all existing FSx instances, their names, unique IDs and statuses:

Example

```
cmjob list-fsx-instances
```

Once the FSx instance has reached a status of `Available` (which can take a few minutes), it is possible to submit a job with this instance.

Example

```
cmjob submit job.sh --fsx-instance <name or FSx ID>
```

Using admin-managed FSx: Using `cmjob` with an admin-managed file system is very similar to using it with a user-managed file system. The main difference is that the admin is responsible for creating and deleting the file system, and that the file system of the admin is shared with specific users. To share the file system, the `share-fsx-instance` option can be run by the admin, together with the `--with` option, as follows:

Example

```
cmjob share-fsx-instance <name or FSx ID> --with <user names>
```

Users do not need FSx-specific tokens to submit a job with an FSx instance. Using this approach for user-managed FSx instances gives the cluster administrator more control over the creation and deletion of FSx instances. By managing profiles, the administrator is free to divide users into a group of users who are allowed to create and delete their own instances, and a group of users who are only allowed to submit jobs with instances that are shared with them.

To stop sharing an FSx instance, the admin can run the `--stop-sharing` option:

Example

```
cmjob share-fsx-instance <name or FSx ID> --stop-sharing
```

Tracking FSx costs: All FSx instances created via `cmjob` have a `BCM Owner` tag assigned to them. This tag allows the cluster administrator to track the FSx costs on a per-user basis using AWS's cost management capabilities.

4.4 Miscellaneous Cloud Tools

4.4.1 Setting Exclude Lists With `excludelistsnippets`

The `excludelistsnippets` submode allows extra exclude list entries to be created and configured for a provisioned file system.

These extra exclude list snippets are added to the regular exclude lists described in section 5.6 of the *Administrator Manual*. The addition of these exclude list snippets to the regular lists is done by setting a mode parameter to the snippets. The mode parameter sets the exclude list that is associated with an exclude list snippet, as indicated by the following table:

Mode	Exclude list
sync	<code>excludelistsyncinstall</code> (active by default)
full	<code>excludelistfullinstall</code>
update	<code>excludelistupdate</code> (active by default)
grab	<code>excludelistgrab</code>
grab new	<code>excludelistgrabnew</code>

An exclude list snippet with an associated mode parameter behaves as if its entries are run along with the regular exclude list that is associated with it.

For example, to exclude the files `/useless/test1` and `/useless/test2`, a snippet called `test` may be created as follows:

Example

```
[bright91->fspart]% use /cm/shared
[bright91->fspart[/cm/shared]]% excludelistsnippets
[bright91->fspart[/cm/shared]->excludelistsnippets]%
```

```
[bright91->fspart[/cm/shared]->excludelistsnippets]% add test
[bright91->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set modefull yes
[bright91->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set excludelist
[bright91->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set excludelist /useless/test1
[bright91->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% append excludelist /useless/test2
[bright91->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% show
```

Parameter	Value
-----	-----
Lines	2
Name	test
Revision	
Exclude list	/useless/test1,/useless/test2
Disabled	no
No new files	no
Mode sync	yes
Mode full	yes
Mode update	yes
Mode grab	no
Mode grab new	no

When the test snippet is used, its corresponding `.rsync` file has the following configuration:

```
[root@bright91 cmd]# grep useless /var/spool/cmd/*shared.rsync

- /useless/test1
- /useless/test2
```

4.4.2 The provisioningassociations Mode

The provisioningassociations mode is not strictly restricted to cloud use, because it can also be used outside the cloud.

It allows direct access to provisioning associations via `cmsh`. The provisioningassociations mode can be used to set properties for the following file systems:

1. `/cm/shared`: as used by the cloud director
2. `/cm/shared`: as used by the edge director
3. `/tftpboot`: as used by the boot role
4. `/cm/node-installer`: as used by the boot role
5. `<image>`: for nodes with a provisioning role: for the image-to-image rsync to other provisioning nodes

The exclude lists for standard image-to-node sync to a regular (non-provisioning) node cannot be altered this way, and should be done in the normal category exclude list way (section 5.6 of the *Administrator Manual*), or via `excludelistsnippets` (section 4.4.1).

The trigger command in `fspart` mode (page 169 of the *Administrator Manual*) triggers an update for a non-image partition. It is not needed for images under `/cm/image`, such as item 5 in the exclude list items that the provisioningassociations mode can be used for (page 76).

Setting Properties Of Edge Director FSParts

The provisioning associations properties can be accessed for an edge director as follows:

Example

```
[bright91]% device use edge-director
[bright91->device[edge-director]]% roles
[bright91->device[edge-director]->roles]% use edgedirector
[bright91->device[edge-director]->roles[edgedirector]]%
provisioningassociations
[bright91->device[edge-director]->roles[edgedirector]->provisioningassociations]% list
FSPart (key)      Sync point      Disabled
-----
/cm/shared                no
[bright91->device[edge-director]->roles[edgedirector]->provisioningassociations]% show /cm/shared
Parameter                      Value
-----
Revision
Sync point
Type                      FSPartBasicAssociation
FSPart                      /cm/shared
On shared storage          no
Disabled                   no
Backup directory
Is root                     no
```

Setting sync for a non-shared filesystem for an edge director: Edge directors can be configured with high availability (HA) (section 2.1.1 of the *Edge Manual*) during the `cm-edge-setup` run (section 2.1.1 of the *Edge Manual*).

- The administrator can set up `/cm/shared` on the edge directors to be shared via a network attached storage (NAS) that is within the edge site network. This would be similar to how HA head nodes are by default configured with NFS (Chapter 18 of the *Administrator Manual*).

For trying it out, it could even be the NFS from the head node that is the NAS, even though this would probably be a bad design choice in a production environment, because edge sites normally need to be able to function with autonomy from the head node.

- Alternatively, the administrator can set up `/cm/shared` on the edge directors to be non-shared, and regularly sync from the active director to the passive director instead.

If the cluster administrator wants `/cm/shared` on the edge directors not to be shared, then syncing that filesystem part between the edge directors needs to be carried out in some way. If the provisioning association parameter `revision` is set with a key-value setting of `ha=yes` for the FSPart (page 169 of the *Administrator Manual*) of `/cm/shared`, then CMDaemon takes care of syncing from the active edge director to the passive edge director.

For example, if directors called `my-director` and `my-director2` have been set up during the `cm-edge-setup` HA configuration run, then the settings that instruct CMDaemon to update `/cm/shared` on the passive edge director can be set up as follows in `cmsh`:

Example

```
[root@bright91 ~]# cmsh
[bright91]% device use my-director
[bright91->device[my-director]]% roles
[bright91->device[my-director]->roles]% use edgedirector
[bright91->device[my-director]->roles[edgedirector]]% provisioningassociations
[bright91->device[my-director]->roles[edgedirector]->provisioningassociations]% list
FSPart (key)      Sync point      Disabled
-----
```

```

/cm/shared                                no
[bright91->device[my-director]->ro...sioningassociations]% use /cm/shared
[bright91->device[my-director]->ro...sioningassociations[/cm/shared]]% set revision ha=yes
[bright91->device*[my-director*]->...sioningassociations*[/cm/shared*]]% commit
[bright91->device[my-director]->ro...sioningassociations[/cm/shared]]% device use my-director2
[bright91->device[my-director2]->ro...sioningassociations[/cm/shared]]% set revision ha=yes
[bright91->device*[my-director2*]->...sioningassociations*[/cm/shared*]]% commit
[bright91->device[my-director2]->ro...sioningassociations[/cm/shared]]%

```

4.5 Connecting To AWS With Direct Connect Or A Hardware VPN

For simple configurations, Bright Cluster Manager recommends, and provides, OpenVPN by default for cluster extension cloudbursting VPN connectivity. If there is a wish to use Direct Connect or a hardware VPN (for example, an IPSec tunnel), then Bright Cluster Manager can be configured to work with those.

The AWS cluster extension always runs in an AWS Virtual Private Cloud (VPC). In the default deployment scenario, the head node communicates with the cloud nodes using an OpenVPN connection between the head node and the cloud director. In the case of a Direct Connect connection or a hardware VPN, the head node can be configured to communicate directly with the cloud director and cloud nodes.

Setting up Direct Connect or a VPN for a cluster extension can be carried out according to these three steps:

1. VPC creation (section 4.5.1). This step can be skipped if an existing VPC is to be used.
2. Connecting the local network to the VPC (section 4.5.2). The connection can be with, for example, Direct Connect, or a hardware VPN. This step can be skipped if the local network is already connected to an existing VPC via a Direct Connect or a hardware VPN connection.
3. Configuring and deploying the cluster extension (section 4.5.3).

4.5.1 Creating a VPC

The Amazon Virtual Private Cloud (VPC) is a logically isolated section of the AWS cloud. A VPC allows complete control over the networking environment for cloud resources used by the cluster extension. The VPC enables the cloud director and cloud nodes to securely communicate with each other and can be extended to the local network of the cluster. Documentation for Amazon VPC can be found at:

<https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>

A new VPC can be created and configured for the cluster extension as follows:

1. After logging in to the AWS management console, <https://console.aws.amazon.com/console/>, the following clickpath can be followed:
Services→VPC→VPC Dashboard→VPCs→Create VPC.
2. An IPv4 CIDR block can be set to a desired range. This range should not conflict with the address space used on-premises. A possible range could be, for example: 10.42.0.0/16
3. Once the VPC range is set, a subnet can be set via the navigation options under:
VIRTUAL PRIVATE CLOUD→Subnets→Create subnet.
4. The VPC created in step 2 is selected as the VPC during the Create subnet dialog. An IPv4 CIDR block that is a subnet of the VPC range must also be defined in the Create subnet dialog. A possible subnet range would be, for example: 10.42.0.0/24

4.5.2 Connecting The Local Network To The VPC

Amazon offers two options to connect the local network to the VPC: Direct Connect and VPN.

Connecting Via Direct Connect

To connect the local network to the VPC via Direct Connect requires a private virtual interface. Amazon's instructions to use AWS Direct Connect to access a VPC are at:

<https://docs.aws.amazon.com/directconnect/latest/UserGuide/Welcome.html>

Connecting Via A Site-To-Site Connection Using A VPN

Connecting an on-premises network to the virtual network using a VPN requires an *Amazon Site-to-Site connection*. Amazon's instructions for configuring a site-to-site VPN connection are at:

<https://docs.aws.amazon.com/vpn/latest/s2svpn/SetUpVPNConnections.html>

The site-to-site connection consists of 4 components:

1. a customer gateway (routes traffic from VPC to local network)
2. a target gateway (routes traffic from local network to VPC)
3. a security group
4. a connection

The target gateway can either be a virtual private gateway or a transit gateway, depending on the configuration of the other AWS resources.

If the customer gateway device does not support BGP, then the site-to-site connection needs to be configured for static routing. The default Static IP Prefixes are: 10.141.0.0/16, 192.168.200.0/24.

4.5.3 Configuring And Deploying The Cluster Extension

Once IP connectivity from on-prem to the AWS virtual network is running, the final step is creating the cluster extension using the newly created site-to-site connection.

Getting Through Shorewall

First, the firewall rules on the head node must be adjusted to accept traffic from the subnet. The file `/etc/shorewall/rules` can be edited so that the `net` section allows packets from the CIDR subnet. A quick and dirty way to do it is to append to the file with:

```
[root@bright91 ~]# echo "ACCEPT net:<subnet CIDR> fw - -" >> /etc/shorewall/rules
[root@bright91 ~]# shorewall reload
```

Cluster Extension Advanced Settings Configuration

A cluster extension can now be created using the `cm-cluster-extension` utility in Ncurses mode as explained in section 4.1.1, page 59.

However, on reaching the summary screen (figure 4.6, page 61), the cluster administrator should not immediately select the `Save config & deploy` option, but should first go to the `Advanced settings` option, which opens up an `Advanced plugin configuration` screen (figure 4.17):

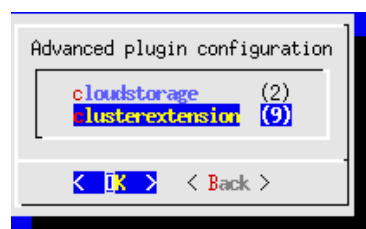


Figure 4.17: Cluster Extension Configuration Processing: Advanced Plugins Screen

The `clusterextension` option should be selected, which opens up an advanced options settings screen (figure 4.18):

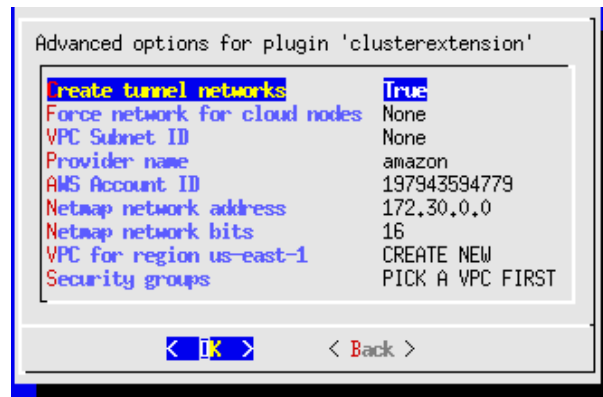


Figure 4.18: Cluster Extension Configuration Processing: Advanced Plugins For 'clusterextension' Options Screen

In this screen:

- the `Create tunnel networks` option should be set to `False`
- the `VPC for region <region>` should be set to the VPC created for the cluster extension. The cluster extension is what is being connected to via Direct Connect or via a site-to-site connection
- The `Security groups` to use should be set for the cloud nodes and the cloud director.

Custom Security Groups Configuration

For the security group it is important to realize that the default security groups, as created by the cluster extension utility, do not accept traffic from the IPSec tunnel. It is recommended to create a custom security group, and to use that for both the director and the nodes. To create this security group:

1. From the AWS management console, the following clickpath is followed:

VPC Dashboard→Security Groups→Create security group

2. The cluster extension VPC is selected
3. A new inbound rule is added, to accept all traffic from the cluster and the VPC. For example: 192.168.200.0/24 and 10.42.0.0/16.

The Nurses advanced settings screens can be backed out of by selecting `Back` twice. The option `Save config & deploy` can then be selected to create the cluster extension.

Cloud Node Certificate Autosigning

By default Bright Cluster Manager does not issue certificates for nodes on the external network. This means that for cloud nodes the certificates need to be issued manually, once for every new cloud node.

Alternatively, to automatically sign certificate requests by cloud nodes, `autosign` can be enabled by the administrator for external networks. Autosigning may be a security concern, as this allows anyone on the external network to request a node certificate. Autosign can be enabled on `externalnet` in `cmsh` as follows:

```
[root@bright91~]# cmsh
[bright91]% network
[bright91->network]% set externalnet allowautosign always
[bright91->network]% commit
```


The cluster extension can now be deployed as explained in section 4.1.2.

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 39.

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 84)
 - 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 40.

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 57. 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 59, 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):

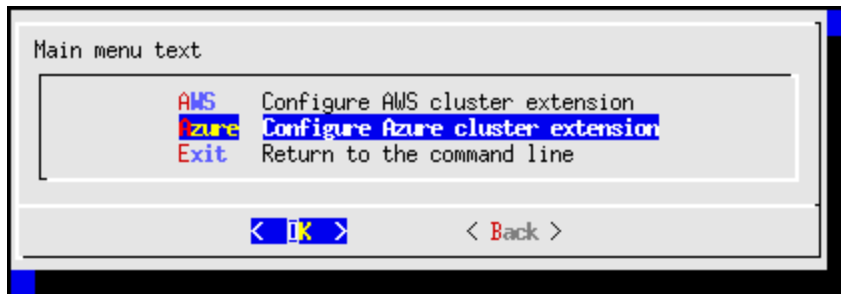


Figure 5.1: Configuration Processing With `cm-cluster-extension`: Azure Selection

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):

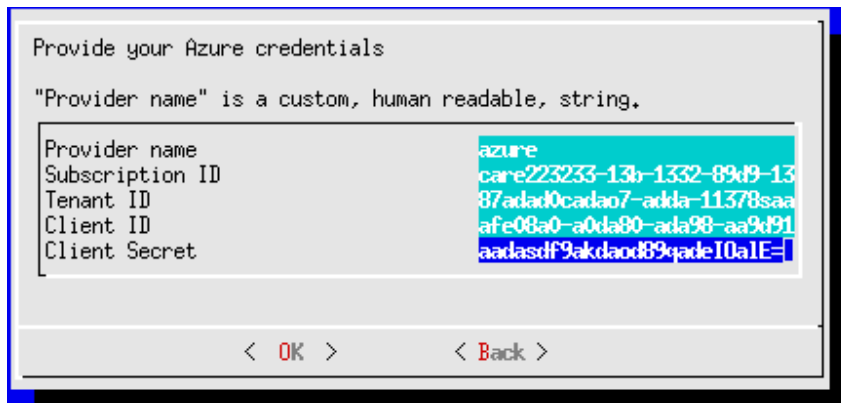


Figure 5.2: Configuration Processing With `cm-cluster-extension`: Azure Credentials

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):

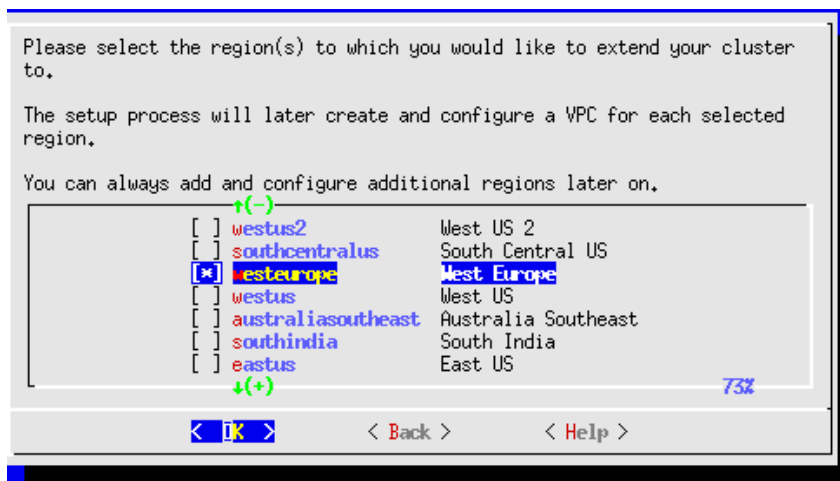


Figure 5.3: Configuration Processing With cm-cluster-extension: Azure Regions

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):

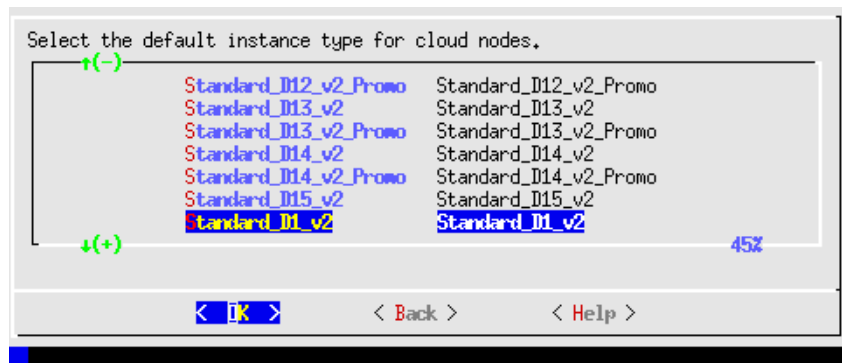


Figure 5.4: Configuration Processing With cm-cluster-extension: Azure Default Instance Type For Cloud Nodes

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):

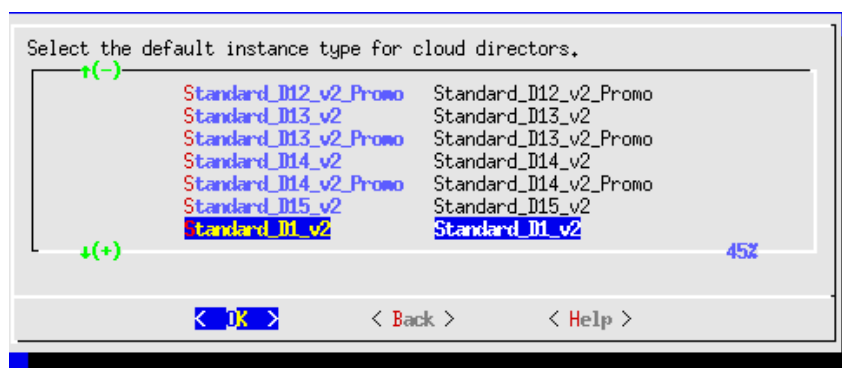


Figure 5.5: Configuration Processing With cm-cluster-extension: Azure Default Instance Type For Cloud Directors

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:

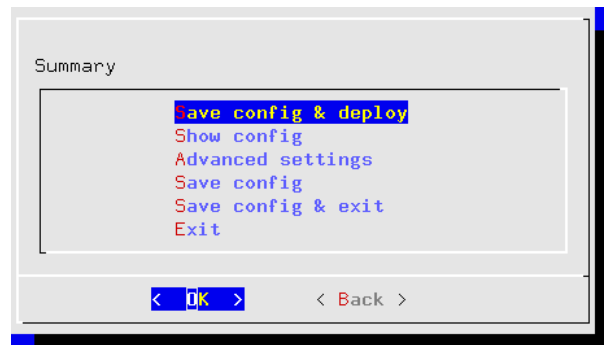


Figure 5.6: Configuration Processing With `cm-cluster-extension`: Azure Options Summary Screen

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

```
[bright91->device]% power on westeurope-director
cloud ..... [ PENDING ] westeurope-director
... [notice] bright91: New certificate request with ID: 9
... [notice] bright91: westeurope-director [ INSTALLER REBOOTING ]
... [notice] bright91: westeurope-director [   INSTALLING   ] (node installer started)
```



```
... [notice] bright91: New certificate request with ID: 10 (ldaps)
... [notice] bright91: westeurope-director [ INSTALLER_CALLINGINIT ] (switching to local root)
... [notice] bright91: westeurope-director [ UP ]
```

Example

```
[root@bright91 ~]# ps uww -C rsync | grep -o ' /cm/.*$'
/cm/shared/ syncer@172.21.255.251::target//cm/shared/
```

Tracking Cloud Node Startup From cmsh

The `provisioningstatus` command in the `softwareimage` mode of `cmsh` can be used to view the provisioning status of cloud directors from the head node, or of cloud nodes from the cloud director (some output elided):

Example

```
[root@bright91 ~]# cmsh -c "softwareimage provisioningstatus"
...
+ westeurope-director
...
Up to date images:      none
Out of date images:    default-image
Nodegroups:            westeurope-director-dependents
```

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
Nodegroups:            westeurope-director-dependents
```

This indicates the image for the cloud nodes is now ready. The image used for a device is defined by the value of `image` under `cloudsettings` for the cloud director, or cloud node:

Example

```
[root@bright91 ~]# cmsh -c "device use westeurope-director; cloudsettings; get image"
marketplace-free-node-installer-image
```

By default, this value is the Azure marketplace image for the current Bright Cluster Manager version.

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@bright91 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):          4
Source node:            bright91
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@bright91 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):      5
Source node:        bright91
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).

Cluster Extension Cloudbursting Logging

All Azure logging goes to the CMDaemon logs in /var/log/cmdaemon, where the CLOUD tag is used to indicate cloud-related operations.

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 163 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:

```
Devices→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:
Devices → 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:

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.

- within the device mode of `cmsh`
- with Bright View, using the clickpath:
Devices→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.

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

The `cmjob` 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-cmjob` package is installed on the head node and the cloud image
- Ensuring the users that are to use `cmjob` 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 `cmjob` 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 `cmjob` is documented in section 4.7 of the *User Manual*.

5.5.1 Integration Of `cmjob` With Azure NetApp Files

Azure NetApp Files (ANF) is a high-performance highly scalable network file service that is available as a cloud service within most Azure regions. A Bright cluster that has burst into such a region can use ANF to create a shared storage volume that can be read from, and can be written to, by many compute nodes simultaneously.

On a cluster that has burst into such a region, `cmjob` can provide workload managers with an ANF volume to store input and output data.

Sequential workload manager jobs can use the same ANF volume. This means that the output of one job can immediately become the input of the next one without having to transmit the data to and from Azure cloud storage, or to and from the head node.

Additionally, workload manager jobs that run on multiple compute nodes in parallel can obtain their shared input from the same ANF volume.

Providing shared storage to cloud compute nodes via ANF is an alternative to providing it via cloud storage nodes (section 4.3). The storage node approach is not sufficiently scalable when there are more than several dozen compute nodes working simultaneously with the same storage node. ANF is significantly more scalable and far more performant, although at a higher cost.

Bright Cluster Manager currently offer three different methods of integrating `cmjob` with Azure NetApp Files:

1. *On-Demand ANF*: An ANF file system is created on a per-job basis and is deleted automatically when the job completes.

2. *User-managed ANF*: A user creates an ANF volume and then submits one or more jobs that use that same ANF volume. These jobs can execute in sequence or simultaneously. When the last job has finished, the user is responsible for deleting the ANF volume.
3. *Admin-managed ANF*: An administrator creates an ANF volume and shares it with one or more users. These users are able to use the volume in the same way as with user-managed ANF, but they cannot delete the volume or see each other's files.

Prerequisite

To be able to use the integration of cmjob with ANF NetApp Files, the cluster must be bursting into an Azure region that supports Azure NetApp Files. Azure does not offer this feature in all regions. The Azure documentation should be checked to see if a region offers ANF for Azure.

Enabling cmjob Integration With Azure NetApp

The integration of cmjob with ANF can be accomplished in two ways, either:

- through the cm-cloud-storage-setup wizard (section 4.3.1)
- or
- manually

During the cm-cloud-storage-setup wizard run, when the guidance for the Azure extension is being carried out, the administrator is asked if ANF support should be enabled. If the administrator agrees to this, then ANF support is set up. After the deployment has finished, all users that have the cloudjob profile can then create user-managed and on-demand ANF volumes through cmjob.

Enabling cmjob integration manually: The manual integration approach should be followed when cmjob has already been configured on a cluster. That is, if the cm-cloud-storage-setup tool has been run, but the cluster has been running without ANF support so far.

To add cmjob integration with ANF manually, the following manual action is all that is needed:

Every user that should be able to create ANF volumes should have one or both of these tokens set on their profile (profiles are explained in section 6.4 of the *Administrator Manual*):

- ON_DEMAND_ANF_TOKEN
- USER_MANAGED_ANF_TOKEN.

Example

A user, fred is already using cluster extension with Azure, but without ANF. Now the user would like to be able to launch and manage ANF storage on demand.

To allow fred to do that, the steps that are carried out could be as follows: The existing profile of fred can be cloned in profile mode. The tokens needed can be appended to the new profile, and the modified profile can then be set for fred.

The following cmsh session shows these steps:

```
[bright91->user[fred]]% get profile
cloudjob
[bright91->user[fred]]% profile
[bright91->profile]% clone cloudjob cloudjobanf
[bright91->profile*[cloudjobanf*]]% get tokens
SUBMIT_CLOUD_JOB_DESCRIPTION_TOKEN
GET_CLOUD_JOB_DESCRIPTION_TOKEN
[bright91->profile*[cloudjobanf*]]% append tokens ON_DEMAND_ANF_TOKEN USER_MANAGED_ANF_TOKEN
[bright91->profile*[cloudjobanf*]]% get tokens
```

```

SUBMIT_CLOUD_JOB_DESCRIPTION_TOKEN
GET_CLOUD_JOB_DESCRIPTION_TOKEN
ON_DEMAND_ANF_TOKEN
USER_MANAGED_ANF_TOKEN
[bright91->profile*[cloudjobanf*]]% commit
[bright91->profile[cloudjobanf]]% user use fred
[bright91->user[fred]]% set profile cloudjobanf
[bright91->user*[fred*]]% commit

```

No directors need to be updated or rebooted when the necessary user profiles have been updated. All users with profiles with those ANF tokens should then be able to create and submit jobs with a user-managed ANF volume, or submit jobs with an on-demand ANF volume, depending on which tokens were assigned to the profile of the user.

Configuration

Through `cmsh`, an administrator can use profile tokens (section 6.4 of the *Administrator Manual*) to control how users can work with ANF. An administrator can also set quotas that control how many ANF volumes a user can have simultaneously and how large these ANF volumes can be. The aim of this is to help manage costs.

Profile tokens:

- `ON_DEMAND_ANF_TOKEN`: allows a user to use the `--on-demand-anf` flag when submitting jobs with `cmjob`.
- `USER_MANAGED_ANF_TOKEN`: allows a user to create, manage and delete ANF volumes.
- `DELETE_ANY_ANF_VOLUME_TOKEN`: allows a user to delete ANF volumes that are owned by other users. Normally reserved for root only.
- `LIST_ALL_ANF_VOLUMES_TOKEN`: allows a user to see ANF volumes owned by other users. Normally reserved for root only.
- `SHARE_ANF_VOLUME_TOKEN`: allows a user to share ANF volumes with other users. Only the owner is allowed to share it. Normally reserved for root only.

Settings: The following `cmsh` navigation path

```
cmsh->cmjob->storagenodepolicies->use <policy>->intermediatestorage
```

leads to these storage parameters:

- Max ANF Volume Capacity (TiB)
- Max ANF Volume Count Per User

which control quota limits for ANF volumes for the number and size of ANF volumes, for when a user or an administrator creates a user-managed or administrator-managed ANF volume.

The count is combined. If a user has already reached the maximum number of user-managed ANF volumes, then the user cannot submit a job that uses an on-demand ANF volume.

Also in the same navigation path is the storage parameter:

- Default ANF Volume Capacity (TiB)

which controls the default capacity when a user or admin creates a user-managed or admin-managed ANF volume.

Usage

As listed earlier, the three methods that Bright Cluster Manager provides for `cmjob` integration with ANF are:

1. on-demand ANF,
2. user-managed ANF, and
3. admin-managed ANF.

How to run these methods using `cmjob` is covered in the next sections.

Using on-demand ANF: When using `cmjob` to submit a job, a user can choose whether to use an Azure storage volume or an ANF file system to host the input data and output data. For a user there is little difference between the ways in which the two shared storage options are used, beside performance.

The steps are:

1. an Azure storage volume or an ANF file system is created as shared storage,
2. the input data is copied from the head node to this shared storage
3. all compute nodes chosen by the workload manager mount this shared storage, and then run their jobs,
4. storing their output data on the same shared storage.
5. the output data is uploaded from the shared storage to the head node
6. the Azure volume or ANF file system is deleted.

Submitting a job to run with Azure storage nodes is the default.

If a user wants a job to use ANF, then the user must set the `--on-demand-anf` flag when submitting the job:

Example

```
cmjob submit --on-demand-anf job.sh
```

The capacity of the OnDemand ANF volume is determined by doubling the size of the input data. This calculation takes into account both the data uploaded from the head node, and labeled data downloaded from the Azure cloud storage. If the user thinks that the calculated capacity is not large enough to hold the output data (the output includes temporary data), then the user can increase the capacity with the `--expected-output-size` parameter. The specified size is combined with the size of the input data and rounded up to the nearest valid ANF volume size.

Using user-managed ANF `cmjob` with a user-managed ANF file system operates differently from an on-demand system. With user-managed ANF, the user is responsible for creating and deleting the ANF file system.

In some cases, this can be more expensive than using the on-demand approach, especially if the user forgets to delete the ANF volume after no longer needing it.

However, with user-managed ANF a user can run multiple concurrent jobs with the same ANF file system. It is also possible to run multiple jobs in sequence, each using the output of the previous job as input, without having the overhead of the files being copied from the ANF file system to Azure cloud storage, and vice versa. Depending on the size of the workload, this option may be cheaper, or more desirable, than on-demand ANF.

To create an ANF volume, the user can execute `cmjob` with the `create-anf-volume` option:

Example

```
cmjob create-anf-volume <name or ID> --capacity <capacity>
```

Here:

- `<name or ID>` can be a sequence of alphanumeric characters and dashes. It does not have to be unique.
- `<capacity>` is the total size of the ANF file system in terabytes. If not specified, the default value is used. The default value can be managed via:
`cmsh→cmjob→storagenodepolicies→use <policy>→intermediatestorage.`

To delete an ANF volume, a user can run `cmjob` with the `delete-anf-volume` option:

Example

```
cmjob delete-anf-volume <name or ID>
```

To obtain a list of all existing ANF volumes with their names and statuses, the user can run `cmjob` with the `list-anf-volumes` option:

Example

```
cmjob list-anf-volumes
```

Once the ANF volume has reached a status of `AVAILABLE` (which can take a few minutes), it is possible to submit a job with this volume name or ID with the `--anf-volume` option.

Example

```
cmjob submit job.sh --anf-volume <name or ID>
```

Using admin-managed ANF Using `cmjob` with an admin-managed file system is very similar to using it with a user-managed file system. The main difference is that the administrator is responsible for creating and deleting the file system, and that the file system of the administrator is shared with specific users. To share the file system, the `share-anf-volume` option can be run by the administrator, together with the `--with` option, as follows:

Example

```
cmjob share-anf-volume <name or ID> --with <user names>
```

With admin-managed ANF, users do not need any ANF-specific tokens to submit a job with an ANF volume. Using admin-managed ANF volumes instead of user-managed ANF volumes gives the administrator more control over the creation and deletion of ANF volumes. By managing profiles, the administrator is free to divide users into a group of users that are allowed to create and delete their own volumes, and a group of users that are only allowed to submit jobs with volumes that are shared with them.

To stop sharing the ANF volume, the admin can run the `--stop-sharing` option:

Example

```
cmjob share-anf-volume <name or ID> --stop-sharing
```

Tracking NetApp costs: All Azure NetApp Files capacity pools created via `cmjob` have a `BCM Owner` tag assigned to them. This tag allows the cluster administrator to track the NetApp costs on a per-user basis using Azure's cost management capabilities.

5.6 Creating An Azure Cluster Extension Using ExpressRoute Or A Hardware VPN

For simple configurations, Bright Cluster Manager recommends, and provides, OpenVPN by default for cluster extension cloudbursting VPN connectivity. If there is a wish to use Direct Connect or a hardware VPN (for example, an IPSec tunnel), then Bright Cluster Manager can be configured to work with those.

The Azure cluster extension always runs in an Azure Virtual Network. In the default deployment scenario, the head node communicates with the cloud nodes using an OpenVPN connection between the head node and the cloud director. In the case of an ExpressRoute connection or a hardware VPN, the head node can be configured to communicate directly with the cloud director and cloud nodes.

Setting up ExpressRoute or a VPN for a cluster extension can be carried out according to these three steps:

1. Virtual network creation (section 5.6.1). This step can be skipped if an existing virtual network is to be used.
2. Connecting the local network to the virtual network (section 5.6.2). The connection can be with, for example, ExpressRoute, or a hardware VPN. This step can be skipped if the local network is already connected to an existing virtual network via an ExpressRoute or a hardware VPN connection.
3. Configuring and deploying the cluster extension (section 5.6.3).

5.6.1 Creating A Virtual Network

The Azure virtual network is the fundamental building block for building private networks in Azure. A virtual network allows the cloud director and cloud nodes to communicate with each other securely, and can be extended to the on-premises networks. Documentation for creating a virtual network can be found at:

<https://docs.microsoft.com/en-us/azure/virtual-network/quick-create-portal>

A new virtual network can be created and configured for the cluster extension as follows:

1. After logging in to the Azure portal, <https://portal.azure.com>, from the home page, <https://portal.azure.com/#home>, the following clickpath can be followed:

+ Create a resource→Azure Marketplace→Networking→Virtual network

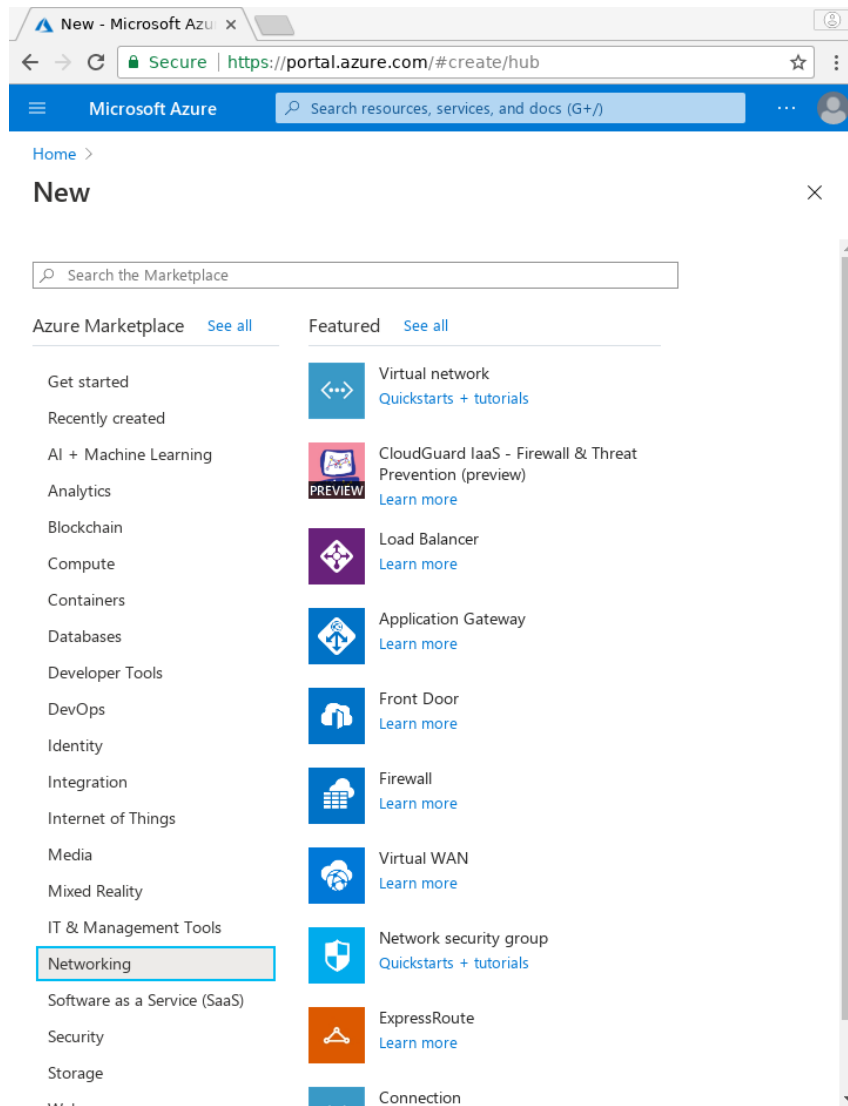


Figure 5.7: Azure Marketplace Networking Screen

2. A resource group should be selected or a new resource group should be created for the virtual network in the Basics tab:

The screenshot shows the 'Create virtual network' wizard in the Azure portal, specifically the 'Basics' tab. The browser address bar shows the URL: <https://portal.azure.com/?quickstart=true#create/Microsoft.VirtualNetwork-ARM>. The page title is 'Create virtual network'. Below the title, there are tabs for 'Basics', 'IP Addresses', 'Security', 'Tags', and 'Review + create'. The 'Basics' tab is selected. The main content area explains that Azure Virtual Network (VNet) is the fundamental building block for a private network in Azure. It then presents two sections: 'Project details' and 'Instance details'. In the 'Project details' section, the 'Subscription' is set to 'Free Trial' and the 'Resource group' is empty, with a 'Create new' link below it. In the 'Instance details' section, the 'Name' field is empty and the 'Region' is set to '(US) East US'. At the bottom, there are three buttons: 'Review + create' (blue), '< Previous' (disabled), and 'Next : IP Addresses >' (disabled). A link 'Download a template for automation' is also present.

Create virtual network

Basics IP Addresses Security Tags Review + create

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation. [Learn more about virtual network](#)

Project details

Subscription *

Resource group *
[Create new](#)

Instance details

Name *

Region *

[Review + create](#) [< Previous](#) [Next : IP Addresses >](#) [Download a template for automation](#)

Figure 5.8: Creating A Resource Group In The Basics Tab

3. An IPv4 CIDR block can be set to a desired range via the IP Addresses tab, reached by clicking on the Next : IP Addresses button:

The screenshot shows the 'Create virtual network' wizard in the Azure portal, specifically the 'IP Addresses' tab. The browser address bar shows the URL: <https://portal.azure.com/?quickstart=true#create/Microsoft.VirtualNetwork-ARM>. The page has a blue header with the Microsoft Azure logo and a search bar. Below the header, there's a breadcrumb trail: Home > New >. The main title is 'Create virtual network' with a close button (X). The tabs are 'Basics', 'IP Addresses' (selected), 'Security', 'Tags', and 'Review + create'. A descriptive text states: 'The virtual network's address space, specified as one or more address prefixes in CIDR notation (e.g. 192.168.1.0/24)'. Under 'IPv4 address space', there's a text input field containing '10.0.0.0/16' and a button to clear the field. Below this is a checkbox labeled 'Add IPv6 address space' with a help icon. Another descriptive text states: 'The subnet's address range in CIDR notation (e.g. 192.168.1.0/24). It must be contained by the address space of the virtual network.' Below this, there's a '+ Add subnet' button and a 'Remove subnet' button. A table lists the subnets:

Subnet name	Subnet address range
default	10.0.0.0/24

At the bottom, there are three buttons: 'Review + create' (blue), '< Previous' (disabled), and 'Next : Security >' (disabled). There is also a link: 'Download a template for automation'.

Figure 5.9: Creating IP Address Ranges In The IP Addresses Tab

This range should not conflict with the address space used on-premises. A possible range could be, for example, 10.77.0.0/16

4. Once the virtual network range is set, a subnet can be entered via the + Add subnet option, which opens up a dialog.

Add subnet ✕

Subnet name *
 ✓

Subnet address range * ⓘ
 ✓
 10.77.0.0 - 10.77.0.255 (251 + 5 Azure reserved addresses)

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

Services ⓘ
 ▼

Add **Cancel**

Figure 5.10: Creating IP Address Ranges In The IP Addresses Tab: Subnet Addition

A possible subnet could be, for example, 10.77.0.0/24. The Add button then adds it.

5. The clickpath:

Review + create → Create

goes on to deploy the virtual network.

5.6.2 Connecting The Local Network To The Virtual Network

Azure offers two virtual network gateway types to connect the local network to the virtual network: ExpressRoute and VPN.

Connecting Via ExpressRoute

To connect the local network to the virtual network via ExpressRoute requires using an *ExpressRoute circuit connection*. Azure's instructions for this are at:

<https://docs.microsoft.com/en-us/azure/expressroute/expressroute-howto-linkvnet-portal-resource-manager>

Connectng Via A Site-To-Site Connection Using A VPN

Connecting an on-premises network to the virtual network using a VPN requires an Azure site-to-site connection. Azure's instructions for configuring a site-to-site IPsec VPN connection are at:

<https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-resource-manager-portal>

In Azure this consists of three components:

1. a virtual network gateway (routes traffic from local network to virtual network)
2. a local network gateway (routes traffic from virtual network to local network)
3. a connection

The local network gateway needs to define the address ranges of the local network. The default address ranges for the local network are: 10.141.0.0/16, 10.2.0.0/16, and 192.168.200.0/24.

The virtual network gateway requires a gateway subnet. The address range for the gateway subnet should be a subnet of the virtual network that does not overlap with the subnet address range for the cluster extension, for example: 10.77.1.0/24.

5.6.3 Configuring And Deploying The Cluster Extension

Once IP connectivity from on-prem to the Azure virtual network is running, the final step is creating the cluster extension using the newly created site-to-site connection.

Getting Through Shorewall

First, the firewall rules on the head node must be adjusted to accept traffic from the virtual network. The file `/etc/shorewall/rules` can be edited so that the `net` section allows packets from the CIDR subnet. A quick-and-dirty way to do it is to append to the file with:

```
[root@bright91 ~]# echo "ACCEPT net:<subnet CIDR> fw - -" >> /etc/shorewall/rules
[root@bright91 ~]# shorewall reload
```

Cluster Extension Advanced Settings Configuration

A cluster extension can now be created using the `cm-cluster-extension` utility in Ncurses mode as explained in section 5.2.

However, on reaching the summary screen (figure 5.6, page 88), the cluster administrator should not immediately select the `Save config & deploy` option, but should first go to the `Advanced settings` option, which opens up an `Advanced plugin configuration` screen (figure 5.11):

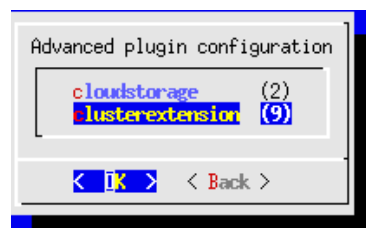


Figure 5.11: Cluster Extension Configuration Processing: Advanced Plugins Screen

The `clusterextension` option should be selected, which opens up an advanced options settings screen for Azure (figure 5.12):

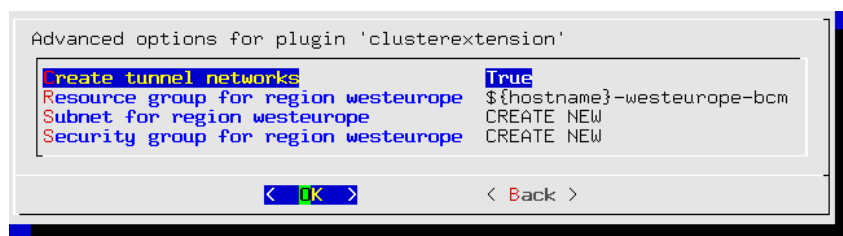


Figure 5.12: Cluster Extension Configuration Processing: Advanced Plugins For 'clusterextension', Azure Options Screen

In this screen:

- the `Create tunnel networks` option should be set to `False`
- the `Subnet for region <region>` should be set to the subnet that has been created for the cluster extension. The cluster extension is what is being connected to via ExpressRoute or via site-to-site connection

The Ncurses advanced settings screens can be backed out of by selecting `Back` twice. The option `Save config & deploy` can then be selected to create the cluster extension.

Cloud Node Certificate Autosigning

By default Bright Cluster Manager does not issue certificates for nodes on the external network. This means that for cloud nodes the certificates need to be issued manually, once for every new cloud node.

Alternatively, to automatically sign certificate requests by cloud nodes, autosign can be enabled by the administrator for external networks. Autosigning may be a security concern, as this allows anyone on the external network to request a node certificate. Autosign can be enabled on `externalnet` in `cmsh` as follows:

```
[root@bright91~]# cmsh
[bright91]% network
[bright91->network]% set externalnet allowautosign always
[bright91->network]% commit
```

The cluster extension can now be deployed as explained on page 88.

5.7 Viewing And Setting Azure Generation 1 And Generation 2 Cloud Nodes

5.7.1 Introduction

Azure introduced Generation 2 (Gen2) VMs in 2019. These boast a number of new features, including increased memory, Intel Software Guard Extensions (Intel SGX), and virtualized persistent memory (vPMEM). Gen2 VMs use the new UEFI-based boot architecture rather than the BIOS-based architecture used by Gen1 VMs. Compared to Gen1 VMs, Gen2 VMs might have improved boot and installation times.

It is not possible to change the generation once a VM has been created. The generation to be used must therefore be specified before deploying a cloud instance.

More information on generation 2 VMs can be found at the feature documentation at <https://docs.microsoft.com/en-us/azure/virtual-machines/generation-2>.

5.7.2 Implementation Of Hyper-V Generation Choice In The Cluster Manager

To use a Gen2 Hyper-V in 9.1, the user needs to:

- select a VM size supporting Gen2.
- specify the Hyper-V generation that is to be used. Some VM sizes support both V1 and V2.

To define the generations used by Azure cloud nodes the administrator needs to set parameters within the `AzureHyperVGenMap` directive. The `AzureHyperVGenMap` directive (page 806 of the *Administrator Manual*) is in turn a parameter of the `AdvancedConfig` directive (page 794 of the *Administrator Manual*).

6

Cluster-On-Demand For OpenStack

6.1 Introduction

If Bright OpenStack is running on a cluster, then ordinary users of the cluster can run a Cluster On Demand (COD) within it. A COD is a complete cluster that is managed by Bright Cluster Manager, which means that a regular user can become a cluster administrator of a virtual cluster running under the hosting cluster. This kind of setup is a case of running a cluster within a cloud service, with the cloud in this case being the Bright OpenStack. Analogous setups where the cloud service is provided by AWS, Azure, or VMware, can also be managed with Bright Cluster Manager (Chapter 2).

In Bright jargon these setups are conveniently called COD-OS, COD-AWS, and COD-Azure.

Bright provides a client, `cm-cod-os`, to launch a COD within a Bright OpenStack cluster. The client can run on the head node of the cluster itself, or from a remote location.

Some possible uses for a Bright OpenStack COD are:

- a staging environment to test the production configuration of a software running on a Bright cluster
- a way to do batch job processing using whatever workload manager the user would like, instead of being limited to one due to the possibility of the workload managers interfering with each other.
- a way to run virtual Bright clusters to try things out on, so that a user can become familiar with administrating Bright cluster, without breaking a production system.

The `cm-cod-os` client is provided by the `cm-cluster-on-demand-openstack` package:

Example

```
[root@bright91 ~]# yum install cm-cluster-on-demand-openstack
```

The `cm-cod-os` client is a Python script that can be run with configuration files (section 6.3) and expects some environment variable settings. If arguments are used by the script, then the arguments override the corresponding configuration file values.

The `cm-cod-os` client is usually run by an ordinary user, and not the administrator, of the host cluster. The ordinary user then typically becomes the owner and administrator of the COD. The COD is thus a virtual cluster that is hosted by the host cluster. This is called a nested cluster.

6.2 The `cm-cod-os` Arguments

6.2.1 The `cm-cod-os` Top Level Arguments

Options to `cm-cod-os` can be viewed with the help option, `-h` | `--help`:

Example

```
[fred@bright91 ~]$
usage: cm-cod-os [-h] [--config CONFIG] [--no-system-config] [-v]
               [--show-configuration]
               cluster,c,cluster create,cc,cluster list,cl,cluster
               delete,cd,cremove,node,n,image,i,image list,il,image
               delete,id,vnc,v,vnc list,vl,flavor,config ...
```

Cluster-on-demand by Bright Computing

positional arguments:

```
cluster,c,cluster create,cc,cluster list,cl,cluster delete,cd,cremove,node,n,image,i,\
image list,il,image delete,id,vnc,v,vnc list,vl,flavor,config
cluster (c)          Cluster operations
cluster create (cc)   Create cluster
cluster list (cl)     List clusters
cluster delete (cd,cremove) Delete all resources in a cluster
node (n)             Node operations
image (i)            Image subcommands
image list (il)       Lists cluster images
image delete (id)     Deletes image from glance
vnc (v)              VNC subcommands
vnc list (vl)         List VNC ports
flavor               Flavor subcommands
config              Configuration operations
```

optional arguments:

```
-h, --help          show this help message and exit
--config CONFIG, -c CONFIG
--no-system-config
-v, -vv, -vvv
--show-configuration
```

6.2.2 The cm-cod-os Context Tree

cm-cod-os has a hierarchy of options. Some special positional subcommand options allow particular hierarchies to be accessed. The context of such a special positional subcommand option decides the hierarchy available.

In Bright terminology, such a “special positional subcommand option” is therefore more conveniently called a “context”. If the context is one level deeper in the hierarchy, then it is called a “sub-context” for precision. However, in more loose usage, the word “context” is generally just assumed to include the idea of “subcontext”.

The syntax is indicated by:

```
cm-cod-os [context [subcontext]] options
```

The cm-cod-os tree has its context and subcontext branches organized as follows:

```
cod
+ cluster
  +create
  +list
  +delete
  +description
```



```

    +shelve
    +show
    +start
    +stop
    +tag

+ config
    +dump
+ node
    +create
+ image
    +list
    +stats
    +repo-list
    +install
    +download
    +repo-download
    +delete
    +show
    +update
    +usage
+ vnc
    +list
+ flavor
    +list

```

6.2.3 The cm-cod-os Contexts And Optional Arguments Help Text

Most contexts and subcontexts, like at the top level of cm-cod-os, have around 10 contexts and argument options or less. For example, the cm-cod-os→cluster help text has the following contexts and options (some output ellipsized):

Example

```
[fred@bright91 ~]$ cm-cod-os cluster -h
usage: cm-cod-os cluster [-h]
```

```

    {create,c,list,l,delete,d,remove,description,shelve,show,start,stop,tag}
    ...

```

positional arguments:

```

{create,c,list,l,delete,d,remove,description,shelve,show,start,stop,tag}
  create (c)          Create cluster
  list (l)            List clusters
  delete (d,remove)   Delete all resources in a cluster
  description         View/change cluster description
  shelve             This command shelves all instances in a cluster. It
                    performs a clean shutdown first.
  show              This command shows details for a (list of) cluster(s)
                    and all related objects.
  start            Start/unshelve cluster(s)
  stop            Stop cluster(s)
  tag            Set tags on a heat stack to turn it into a cluster

```

optional arguments:

```
-h, --help    show this help message and exit
```

The cm-cod-os cluster create Options Help Text

Probably the most used path in the cm-cod-os command hierarchy for a user is cm-cod-os→cluster→create. This can be used to create a cluster in many different ways, so naturally it has the most options.

Example

```
[fred@bright91 ~]$ cm-cod-os cluster create -h
usage: cm-cod-os cluster create [options]
```

Create cluster

cluster create parameters:

```
-n NODES, --nodes NODES
    Number of compute nodes to be created according to one
    of the following formats (several space separated
    groups can be specified): node_number
    node_number:node_flavor node_number:node_template_path
    node_number:node_template_path:node_flavor You find an
    example of what node templates look like in:
    /cm/local/apps/python3/lib/python3.7/site-
    packages/clusterondemandopenstack/static/cod-os-node-
    template.yml.

--wlm {dont-configure, sge, pbspro-ce, pbspro, slurm}
    Workload Manager of choice. This can also be configured
    later. Note that this list is a subset of the WLM
    systems supported across different versions of Bright.
    Therefore, some WLM systems are only configurable later
    on, after the cluster has been created. Defaults to
    'dont-configure'. dont-configure - do not configure any
    (can be configured later).

--copy-file SRC_PATH[:DST_PATH] [SRC_PATH_2[:DST_PATH_2] ... ]
    Colon separated source path and destination path. If
    only the source path is specified, it will be used as
    the destination path. Note: -Using a tilde (~) will be
    expanded to the user home. Note: The source path can
    be an HTTP URL. Note: The keyword {COD_BRIGHT_VERSION},
    if present, will be replaced with the value of COD
    Bright Versions. Example1: /etc/file.conf Example2:
    /home/user/custom-file.conf:/etc/file.conf
    Example3: http://localhost/path1:/destination/path1
    /source/path2:/destination/path2.

--copy-file-with-env COPY_FILE_WITH_ENV
    Same as --copy-file but replaces instances of
    ${ENV_VAR} inside of the file being copied with the
    content of the environment variable 'ENV_VAR' as well
    as {COD_BRIGHT_VERSION} by the bright version in the
    file path.

--name NAME
    (default: auto) Name of the cluster to createBy default
    the name is generated from version, distro, label.

--timezone TIMEZONE
    Timezone of the cluster.

--store-head-node-ip PATH_TO_FILE
    Once the cluster has been created, store the IP of the
    head node in a file. Useful for automation.
```

```

--ask-to-confirm-cluster-creation {TRUE,FALSE}
    Ask for confirmation when creating a new cluster. Use
    -y to skip this question.
--run-cm-bright-setup {TRUE,FALSE}
    Whether or not to initialize the cluster by running cm-
    bright-setup (activate license, etc).
-m HEAD_NODE_TYPE, --head-node-type HEAD_NODE_TYPE
    Flavor for the head node. Use flavor name from 'cm-cod-
    os flavor list'.
--secondary-head-node-type SECONDARY_HEAD_NODE_TYPE
    Flavor for the secondary head node. Use flavor name
    from 'cm-cod-os flavor list'.
--nas-node-type NAS_NODE_TYPE
    Flavor for the NAS node. This node is created in HA
    clusters to provide shared storage, based on NFS, to
    both head nodes. Use flavor name from 'cm-cod-os flavor
    list'.
--ha
    Setup HA for the cluster head nodes.
--head-node-az HEAD_NODE_AVAILABILITY_ZONE, --head-node-availability-zone
    HEAD_NODE_AVAILABILITY_ZONE
    Name of the availability zone to create the head node
    on. If not specified, OpenStack's scheduler will
    decide. This argument can also be used to force the
    head node onto a specific hypervisor. To do so, specify
    "<availability_zone>:<hypervisor>", e.g.
    "default:hyper01".
--ssh-key-pair SSH_KEY_PAIR
    Name of the key pair used to access the head node.
--internal-cidr INTERNAL_CIDR
    CIDR of the cluster's internal network.
--failover-cidr FAILOVER_CIDR
    CIDR of the cluster's failover network.
--internal-mtu NUMBER
    MTU of the cluster's internal network.
--ingress-icmp INGRESS_ICMP
    CIDR from which to allow ingress ICMP traffic to the
    head node. Specify 'None' to disable ICMP all together.
--wait-ssh SECONDS
    Wait up to that many seconds for SSH to come up.
--wait-cmdaemon SECONDS
    Wait up to that many seconds for CMDaemon to come up.
--prebs COMMAND
    Command(s) executed by cloud-init before cm-bright-
    setup (before CMDaemon starts). Useful for package
    update. Multiple arguments are allowed.
--postbs COMMAND
    Command(s) executed by cloud-init post cm-bright-setup
    (Once CMDaemon starts).
--append-to-root-bashrc ENV=VAR
    Lines to append to the /root/.bashrc file on the head
    node.
--admin-email ADMIN_EMAIL
    Admin email address to set in CMDaemon.
--inbound-rule INBOUND_RULE
    One or several inbound traffic rules for the cluster's
    head node in the following format:
    [src_cidr[:src_port],]dst_port[:protocol]. Where port
    can be a single port or a dash separated range and

```

```

        supported protocols are: tcp, udp. A wildcard value
        will be assumed for every optional non-provided
        parameter (e.g. all ports, all protocols, all IPs)
        Examples: '80' '21:udp' '11.0.0.0/24,20-23:TCP'
        '12.0.0.0/32:6000-6500,443'.

--send-email-first-boot
        Send an email on the first boot to the cluster
        administrator.

-d, --dry-run
        Dry run - do not actually create the cluster. Useful
        with --template.

--description DESCRIPTION
        Cluster description.

-t OUTPUT_FILE, --template OUTPUT_FILE
        Generate resulting heat template to the file. Use '-'
        as the file name to output the template to stdout.
        Useful with --dry-run.

-y, --yes
        Do not ask for confirmation when creating a new
        cluster.

--wait-for-nodes SECONDS
        Wait for up to that many seconds for the compute nodes
        to come up.

--power-control
        Enable support for power control.
        (76 additional parameters can be displayed with the --advanced-help argument)

image selection parameters:
--version VERSION
        Bright Cluster Manager version.
--distro DISTRO
        Linux distribution name.
--package-groups PACKAGE_GROUPS
        Package group.
--image IMAGE_SPEC
        Single image selector statement. See cm-cod-os image
        list --help. Overrides filter arguments such as
        --version, --distro, etc.
--head-node-image UUID|IMAGE-NAME|IMAGE-SET
        Single image selector statement for the head node
        image. Can either be a Glance image UUID, the name of
        that Glance image or the name of the image set.
        Overrides the head node image selected by --image and
        all other image filter arguments.
--node-image NODE_IMAGE
        Single image selector statement for node image (as in
        advanced mode) '--node-image none' will force the
        cluster to not use a node image at all. Overrides the
        head node image selected by --image and all other image
        filter arguments.
--tags TAGS
        Single image selector statement. See cm-cod-os image
        list --help.
--ha-tags HA_TAGS
        Overrides --tags, but only when --ha is used.
--status STATUS
        Glance status of the image.
--cmd-revision-min NUMBER
        Minimum CMDaemon revision required.
--cmd-revision-max NUMBER
        Maximum CMDaemon revision.
--revision NUMBER
        Select clusters with specified revision.
        (9 additional parameters can be displayed with the --advanced-help argument)

```

root login method to the head node:

```
--log-cluster-password
    Log cluster password to the screen and log files. This
    option is mandatory if no custom password, nor SSH
    keypairs, were specified.

--cluster-password CLUSTER_PASSWORD
    The root user password to the cluster. If not
    specified, a random one will be generated (use --log-
    cluster-password to see it). This is also the root user
    SQL password on the head node. Upon cluster creation
    the password is stored in the
    /cm/local/apps/cmd/etc/cmd.conf on the head node.

--ssh-password-authentication {TRUE,FALSE}
    If set to true, it will be possible to SSH to the head
    node using a password. This option should NOT be used
    in untrusted environments as it exposes the head node
    to brute force login attacks.

--access-validation
    Causes the cluster creation process to abort early if
    it won't be able to guarantee that the SSH access to
    the cluster will be possible. Disabling it is useful
    when e.g. the public SSH key is being delivered to the
    image in some other way, than the usual command line
    argument. Note, that access validation does not attempt
    to actually connect to the cluster. Instead, it merely
    tries to predict whether the cluster will be accessible
    to the user, given the specified argument combination.

--ssh-pub-key-path PATH_TO_FILE
    Path to the public key.

    (4 additional parameters can be displayed with the --advanced-help argument)
```

node volume parameters:

```
--node-root-volume-size SIZE_IN_GB
    Root volume size in GB.

--node-root-volume-type NODE_ROOT_VOLUME_TYPE
    Compute node root disk volume type. Allows for
    specifying a special volume type with different Quality
    of Service policy (more IOPS etc).

--head-node-root-volume-size SIZE_IN_GB
    Head node root disk size in GB. Should be bigger than
    the Image size.

--head-node-root-volume-type HEAD_NODE_ROOT_VOLUME_TYPE
    Head node root disk volume type. Allows for specifying
    a special volume type with different Quality of Service
    policy (more IOPS etc).

--head-node-extra-volume-type HEAD_NODE_EXTRA_VOLUME_TYPE
    Head node extra disk volume type. Allows for specifying
    a special volume type with different Quality of Service
    policy (more IOPS etc).

--head-node-extra-volume-size SIZE_IN_GB
    Second volume for the extra.

    (1 additional parameters can be displayed with the --advanced-help argument)
```

Bright Cluster Manager licensing information:

```
--license-unit LICENSE_UNIT
    License unit.
--license-locality LICENSE_LOCALITY
    License locality.
--license-country LICENSE_COUNTRY
    Two characters.
--license-product-key LICENSE_PRODUCT_KEY
    Bright Cluster Manager Product Key.
--license-organization LICENSE_ORGANIZATION
    Name of your organization.
--license-state LICENSE_STATE
    Name of your state or province.
(3 additional parameters can be displayed with the --advanced-help argument)
```

SSH local configuration:

```
--update-ssh-config    Updates the contents of ~/.ssh/config, this is used to
                        maintain the local ssh configuration access with ssh,
                        scp and related tools.
--ssh-config-alias-prefix SSH_CONFIG_ALIAS_PREFIX
                        Prefix to be used when populating host entries for the
                        cluster head nodes in the COD section of ~/.ssh/config.
                        Default is ''.
(1 additional parameters can be displayed with the --advanced-help argument)
```

common parameters:

```
-h, --help            Show this message and exit.
--advanced-help       Don't omit advanced configuration parameters from the
                        help output. Implies --help.
--explain EXPLAIN     Show detailed information about the immediately
                        following parameter. Which can be a name, a regular
                        expression, ENV VAR or another flag.
-v, -vv, -vvv, --verbose
                        Verbosity level.
(11 additional parameters can be displayed with the --advanced-help argument)
```

configuration parameters:

```
-c CONFIG, --config CONFIG
    Extra config files.
(8 additional parameters can be displayed with the --advanced-help argument)
```

node definition parameters:

```
--node-disk-setup-path PATH_TO_XML
    Path to the XML file with the disk setup which is to be
    used for the nodes.
--node-disk-setup NODE_DISK_SETUP

--node-az DEFAULT_NODE_AVAILABILITY_ZONE, --default-node-az DEFAULT_NODE_AVAILABILITY_ZONE,
--node-availability-zone DEFAULT_NODE_AVAILABILITY_ZONE, --default-node-
```

```
availability-zone DEFAULT_NODE_AVAILABILITY_ZONE
Default name of the availability zone to create the
compute nodes on. Will not override the availability
zone that is specified in the node template. If not
specified, OpenStack's scheduler will decide. This
argument can also be used to force the compute node
onto a specific hypervisor. To do so, specify
"<availability_zone>:<hypervisor>", e.g.
"default:hyper01".
```

(4 additional parameters can be displayed with the --advanced-help argument)

openstack authentication parameters:

```
--os-auth-url URL      OpenStack Authentication URL. This should typically be
                        the public Keystone API endpoint (v3), e.g. 'http://my-
                        example-cloud:5000/v3'. This can also be set with
                        OS_AUTH_URL environment variable.

--os-username USERNAME Username of the OpenStack user which is to be used for
                        authentication with keystone (--os-auth-url). The user
                        should have a OpenStack role assigned in the project
                        specified with --os-project-name. This can also be set
                        with OS_USERNAME environment variable.

--os-password PASSWORD OpenStack password for the OpenStack user specified
                        with --os-username. This can also be set with
                        OS_PASSWORD environment variable. Avoid setting the
                        password via the command line, it's unsafe. If no
                        password is provided beforehand, the user will be
                        presented with an interactive prompt, and asked to
                        provide it at runtime.

--os-project-name PROJECT_NAME The name of the OpenStack project (a.k.a. tenant). The
                                OpenStack user (--os-username) which is used for
                                authentication should have a OpenStack role assigned in
                                this project. This can also be set with OS_PROJECT_NAME
                                environment variable.
```

(6 additional parameters can be displayed with the --advanced-help argument)

Some sections were omitted

6 sections with a total of 19 parameters can be displayed with the --advanced-help argument

6.3 The cm-cod-os Configuration Files

6.3.1 The cm-cod-os Configuration Files Locations

By default, configuration files for setting up the COD are searched for in the following locations, and in the following sequence. Settings found earlier in the sequence are overwritten by settings later on in the sequence.

- /etc/cm-cluster-on-demand.ini
- /etc/cm-cluster-on-demand.conf
- /etc/cm-cluster-on-demand.d/*
- ~/cm-cluster-on-demand.ini

- `~/cm-cluster-on-demand.conf`
- `~/cm-cluster-on-demand.d/*`
- a file location specified on the command line. For example, a file `mycodsettings` can be accessed using the `--config` option of `cm-cod-os`:

Example

```
[fred@bright91 ~]$ cm-cod-os --config mycodsettings
```

Typically, the administrator sets up configuration options in one of the first 3 locations, and the regular user modifies the options or adds other options in one of the last 4 locations.

6.3.2 Viewing The `cm-cod-os` Configuration File Options

A dump of the existing configuration can be viewed using `cm-cod-os config dump`

To check what options have been applied, and their sequence, the log to STDOUT can be viewed if the `-v|--verbose` option has been applied.

A list of configuration options for the `cm-cod-os cluster create` command can be seen with the `--show-configuration` option (output truncated):

Example

```
[fred@bright91 ~]$ cm-cod-os cluster create --show-configuration
+-----+-----+
| option                                | value                                |
+-----+-----+
| advanced_help                        | False (default)                    |
| append_to_root_bashrc                | [] (default)                       |
| ask_to_confirm_cluster_creation      | True (default)                     |
| cluster_password                     | None (default)                     |
| cmd_revision_max                     | None (default)                     |
| ...
```

Arguments to `cm-cod-os` override the equivalent configuration file settings. This means that the configuration file settings of a working configuration can be used as a default template, and modifications to the template can conveniently be carried out via command line.

6.3.3 Setting The `cm-cod-os` Configuration File Options And Corresponding Arguments

The configuration options can be placed under sections that are associated with the corresponding `cm-cod-os` contexts and subcontexts of the tree in section 6.2.2.

For example, the path to the subcontext `cm-cod-os cluster create` has a large number of possible options (the options listed starting on page 106). The configuration file for the options can then have a section that begins with:

```
[openstack.cluster.create]
```

The section and options that can be placed in a configuration file can be worked out from the help text output of the `cm-cod-os` for the associated context or subcontext.

For example, the help text for `cm-cod-os cluster create -h` has the excerpt:

```
--ask-to-confirm-cluster-creation {TRUE,FALSE}
    Ask for confirmation when creating a new cluster. Use
    -y to skip this question.
```


The configuration option for this option then takes the form:

```
openstack.cluster.create.ask_to_confirm_cluster_creation
```

The options are placed in a key=value format under the associated section.

Thus, the option that can be set to ask for confirmation before creating a new cluster is then `ask_to_confirm_cluster_creation`. It can be placed under the section `[openstack.cluster.create]` in key=value format as follows:

```
[openstack.cluster.create]
ask_to_confirm_cluster_creation=yes
```

6.4 The cm-cod-os Environment Variables

The environment in which `cm-cod-os` runs also provides the script with information via OpenStack environment variables.

These OpenStack variables are typically exported in the `.bashrc` or `.openstackrc` file for the COD owner using the Bright OpenStack cluster. More on the export of the OpenStack variables is given in the section with the heading **Background Note: Automated Writing Out Of The `.openstackrc`* Files**, in Chapter 5 of the Bright 9.0 OpenStack Deployment manual. The environment variables, which are prefixed with `OS_`, then typically exist in the environment of the OpenStack COD owner.

Example

```
[fred@bright91 ~]$ grep OS_ .bashrc
export OS_AUTH_URL="http://master:5000/v3"
export OS_PROJECT_NAME="${USER}-project"
export OS_USERNAME="${USER}"
export OS_TENANT_NAME="${USER}-project"
export OS_PROJECT_DOMAIN_ID="9b9d86bb35934072b7c2a5c73ce75d43"
export OS_USER_DOMAIN_ID="9b9d86bb35934072b7c2a5c73ce75d43"
export OS_IDENTITY_API_VERSION=3
export OS_CACERT="/etc/keystone/ssl/certs/ca.pem"
export OS_INITIALS=$COD_PREFIX
[fred@bright91 ~]$
```

So, when the `cm-cod-os` script runs, it works for the COD owner with the local Bright OpenStack cluster by default.

To have `cm-cod-os` work with other clusters requires appropriate changes in these environment variables, as well as in the `cm-cod-os` configuration options.

6.5 Launching A COD

This section consists of example sessions, to show how the material in the preceding sections of this Chapter can be used to launch a nested COD. Some administrative preparation is first carried out on the host cluster (subsection 6.5.1). Once the host cluster is ready, the nested cluster can be launched and configured (subsection 6.5.2).

6.5.1 Administrative Preparation Of The Host Cluster

It is assumed that the host cluster is configured with Bright OpenStack already. For example, with Bright View OpenStack wizard (section 3.1 of the Bright 9.0 OpenStack Deployment manual), or with `cm-openstack-setup` (section 3.2 of the Bright 9.0 OpenStack Deployment manual).

Installing cm-cod-os

On the head node, the cm-cod-os utility should be present. If it is missing, then it should be installed with:

Example

```
[root@bright91 ~]# yum install cm-cluster-on-demand-openstack
```

Setting The OpenStack Port Security Extension Driver

The OpenStack port security plugin (<https://wiki.openstack.org/wiki/Neutron/ML2PortSecurityExtensionDriver>) should be enabled to allow the toggling of packet filtering for the hosted devices, so that the hosted nodes can be served DHCP leases.

```
[root@bright91 ~]# cmsh
[bright91]% openstack
[bright91->openstack[default]]% settings
[bright91->openstack[default]->settings]% networking
[...settings->networking]% set enableml2portsecurityplugin yes
[...settings*->networking*]% commit
```

Viewing The Existing Flavors

If the OpenStack credentials and environment are in place, then the standard flavor list provided by Bright OpenStack can be seen by using the OpenStack CLI client. The root user can see these with:

Example

```
[root@bright91 ~]# source .openstackrc; source .openstackrc_password
[root@bright91 ~]# openstack flavor list
```

ID	Name	RAM	Disk	Ephemeral	VCPUs	Is Public
0f57527b-4fb2-452d-9b52-13c2	m1.large	8192	20	80	4	True
5309e46d-5c2d-47a0-9ad8-dae9	m1.xtiny	256	1	0	1	True
9b711a53-282c-4901-8bef-20ac	m1.xlarge	16384	40	160	8	True
ccbad1b4-bdb2-40fe-b948-1819	m1.tiny	512	5	5	1	True
e6408d86-a9ca-4694-9066-f8a8	m1.medium	4096	20	40	2	True
ebf16a3f-c3ee-41e7-952d-2643	m1.xsmall	1024	10	10	1	True
f66090cd-b020-49a2-808c-393b	m1.small	2048	10	20	2	True

Alternatively, without having to explicitly source the credentials and the environment, the Bright Cluster Manager cmsh equivalent can be run:

Example

```
cmsh -c "openstack; flavors; list"
```

Adding COD Flavors

Some arbitrary COD flavors can be defined by the hosting administrator according to the possible requirements. A convenient set could be:

Example

```
[root@bright91 ~]# openstack flavor create --ram 1024 --vcpus 1 cod.xsmall
[root@bright91 ~]# openstack flavor create --ram 2048 --vcpus 2 cod.small
[root@bright91 ~]# openstack flavor create --ram 4096 --vcpus 2 cod.medium
[root@bright91 ~]# openstack flavor create --ram 8192 --vcpus 4 cod.large
[root@bright91 ~]# openstack flavor create --ram 16384 --vcpus 8 cod.xlarge
```

Alternatively, if using `cmsh`, then the flavors can be set within the `flavors` submode of `openstack` mode. For example, the preceding `cod.xsmall` flavor can be set with:

Example

```
cmsh -c "openstack flavors; add cod.xsmall; set ram 1GiB; set vcpus 1; commit"
```

Creating A Volume Type

A volume is a block storage that can be used for persistent storage and attached to instances. A simple volume type with default properties can be created by the root user, and given the unimaginative name of `default`, with:

Example

```
[root@bright91 ~]# openstack volume type create default
+-----+
| Field      | Value                                     |
+-----+
| description | None                                     |
| id          | 6486cff8-c2bf-4a3d-a32e-3a67b94ca564    |
| is_public   | True                                     |
| name        | default                                  |
+-----+
```

If the user has access to `cmsh`, then another way to set this would be:

```
cmsh -c "openstack; volumetypes; add default; commit"
```

At this stage in the session, the following points may help orient the reader:

OpenStack client commands typically have `cmsh` or Bright View equivalents: As the preceding examples illustrate, the more useful of the OpenStack client commands can be carried out with a `cmsh` or Bright View equivalent. Whatever is used is a matter of preference and convenience.

Non-root methods to carry out the OpenStack client commands: So far in this section (6.5.1) the preparation to launch a COD has been carried out as a root user. However a non-root user can run these tasks too, with the right credentials.

If using the OpenStack client, then the appropriate credentials for a user are created when an OpenStack user with the same name as the Bright user is created. This is described in more detail in section 5.1.2 of the Bright 9.0 OpenStack Deployment manual. Thus, for example, if a non-root user who is the Bright user `fred` is created as an OpenStack user too, then `fred` becomes able to view the flavor list too, as follows:

Example

```
[fred@bright91 ~]$ source .openstackrc; source .openstackrc_password
[fred@bright91 ~]$ openstack flavor list
```

OpenStack flavors are a superset of COD flavors. To view only COD flavors—flavors that are used by COD instances only—the following command can be run, as root or as the regular user:

Example

```
cod flavor list
```

It should be noted that the `cmsh` or Bright View equivalents for commands in this section (6.5.1) can always be run by a non-root user. However, to be able to run them, the non-root user must have sufficient privileges. Such privileges can be set by the host cluster administrator modifying the profile settings (section 6.4 of the *Administrator Manual*) for the non-root user.

Generally, the preparation in this section (6.5.1) is done as a root user. In the following section (6.5.2), the non-root end user gets to launch and configure the cluster.

6.5.2 Launching And Configuring The Nested Cluster As A User

Typically, the administrator is expected to have configured a global default configuration file already (section 6.3).

If the user would like to generate and modify one for themselves, then the steps in this section can be followed. The steps here are also useful for an administrator who is setting up and trying out a global default configuration file for users.

Generating And Modifying A Configuration File For `cm-cod-os`

A `.ini` configuration file can be generated for the user with

```
[fred@bright91 ~]$ cm-cod-os config dump > ~/cm-cod-os.ini
```

To launch the nested cluster, the settings that must be modified within this file are the following:

- `license_product_key=<the product key>`
- `cluster_password=<a password>`
- `floating_ip_network_uuid=<network UUID>`

If the Bright flat external net is used for floating IP addresses, then the network UUID can be obtained from the OpenStack client:

Example

```
[fred@bright91 ~]$ openstack network show bright-external-flat-externalnet -f value -c id
9a41d19c-7d04-441e-85e8-ee4f15c3cb7f
```

The following settings in the `.ini` file are assumed to have been defined, but can be modified:

- `node_boot_image=<image name>`
By default, the boot image name is `iPXE-plain-eth0`.
- `head_node_type=<image name>`
By default, the head node image name is `cod.medium`.
- `default_node_type=<image name>`
By default, the regular node image name is `cod.xsmall`.
- `internal_mtu=<MTU size of the hosting internal network>`

By default, the hosted network MTU size is set to a standard size of 1500. This allows hosted nodes to PXE boot because PXE booting does not accept MTU options in the DHCP server. For VXLAN-based network isolation this means that the hosting internal network must have a larger MTU value than 1500 to accommodate the hosted network MTU size.

If the MTU value of the hosting internal network must remain at a value of 1500, then VLAN-based network isolation can be used instead.

The arguments to the preceding options take the form `--license-product-key`, `--cluster-password`, `--floating-ip-network-uuid`, and so on, as listed within the section on the `cm-cod-os cluster create` options help text (page 106).

Viewing And Picking Up An Image To Be Used For `cm-cod-os`

The image IDs that are available from the Bright Computing repositories, and their properties can be listed:

Example

```
[fred@bright91 ~]$ cm-cod-os image repo-list
```

ImageID:Revision	Head(GB)	Node(GB)	Distro	CMD Rev.	BCM Version
centos7u5-8.1:9	3.72	1.4	centos7u5	131439	8.1
centos7u2-8.0:7	3.65	1.25	centos7u2	127928	8.0
centos7u2-7.3:14	3.38	1.04	centos7u2	35931	7.3

A suitable image ID value can then be chosen. Each image ID has one head image and one regular node image associated with it. These images are then both installed:

Example

```
[fred@bright91 ~]$ cm-cod-os image install --is-public yes <ImageID:Revision>
```

The installation can take some time (minutes between each stage). A session run displays output similar to the following:

Example

```
[fred@bright91 ~]$ cm-cod-os image install centos7u5-8.1:9
```

ImageID:Revision	Head(GB)	Node(GB)	Distro	CMD Rev.	BCM Version
centos7u5-8.1:9	3.72	1.4	centos7u5	131439	8.1

```
About to install these images
Proceed? [yes/no] yes
INFO: Downloading bcmn-centos7u5-8.1-9 to /home/fred/bcmn-centos7u5-8.1-9.img.gz...
INFO: Creating manifest file /home/fred/bcmn-centos7u5-8.1-9.img.gz.manifest
INFO: Downloading https://s3-eu-west-1.amazonaws.com/cod-os-images.support.brightcomputing.com/
      bcmn-centos7u5-8.1-9.img.gz
INFO: Checking MD5 sum of /home/fred/bcmn-centos7u5-8.1-9.img.gz
INFO: Uploading /home/fred/bcmn-centos7u5-8.1-9.img.gz to glance...
INFO: Upload operation finished successfully.
INFO: Downloading bcmh-centos7u5-8.1-9 to /home/fred/bcmh-centos7u5-8.1-9.img.gz...
INFO: Creating manifest file /home/fred/bcmh-centos7u5-8.1-9.img.gz.manifest
INFO: Downloading https://s3-eu-west-1.amazonaws.com/cod-os-images.support.brightcomputing.com/
      bcmh-centos7u5-8.1-9.img.gz
INFO: Checking MD5 sum of /home/fred/bcmh-centos7u5-8.1-9.img.gz
INFO: Uploading /home/fred/bcmh-centos7u5-8.1-9.img.gz to glance...
INFO: Upload operation finished successfully.
[fred@bright91 ~]$
```

The image ID for the installed images installed by `cm-cod-os` locally can then be seen with:

Example

```
[fred@bright91 ~]$ cm-cod-os image list
+-----+-----+-----+-----+-----+-----+
| ImageID:Revision | Head(GB) | Node(GB) | Distro   | CMD Rev. | BCM Version |
+-----+-----+-----+-----+-----+-----+
| centos7u5-8.1:9   | 12.46    | 3.93     | centos7u5 | 131439    | 8.1         |
+-----+-----+-----+-----+-----+-----+
```

A more extensive images list can be seen with:

Example

```
[fred@bright91 ~]$ openstack image list
+-----+-----+-----+-----+-----+
| ID                               | Name                                           | Status |
+-----+-----+-----+-----+-----+
| 43797ed5-0cdf-40f6-bbcd-1bdb11bfb74 | Bright-COD-headnode-bootloader               | active |
| 71568b33-1ef7-4099-aa10-f9690e52aaf1 | Bright-Managed-VM-iPXE-eth0                 | active |
| 61856733-8f1f-4df7-a68b-d5ca27873f85 | Bright-Managed-VM-iPXE-eth1                 | active |
| c34cbb10-3bfc-4d5f-b320-d446f7ded267 | bcmh-centos7u5-8.1-9                         | active |
| ca519e84-5fac-4c03-8050-299400e822d2 | bcmn-centos7u5-8.1-9                         | active |
| 953f9337-d5a1-4170-9edd-21d9149bbe4a | iPXE-plain-eth0                             | active |
| 0d94b9a9-3d06-4a6a-b446-bc3cea7dfdd9 | iPXE-plain-eth1                             | active |
+-----+-----+-----+-----+-----+
[fred@bright91 ~]$
```

Using cm-cod-os

The user can now start creating clusters. A basic command using the .ini configuration file would be:

```
[fred@bright91 ~]$ cm-cod-os -c cm-cod-os.ini cluster create
11:15:09: INFO: Please wait...
11:15:10: INFO: -----
11:15:10: INFO: Cluster: c-09-05-8.1-c7u5
11:15:10: INFO: -----
11:15:10: INFO: Image name: bcmh-centos7u5-8.1-9(centos7u5-8.1:9)
11:15:10: INFO: Node image: bcmn-centos7u5-8.1-9(centos7u5-8.1:9)
11:15:10: INFO: Image date: 2018-08-17 10:48 (19d 0h ago)
11:15:10: INFO: Package groups: none
11:15:10: INFO: CMD rev.: 131439
11:15:10: INFO: Version: 8.1
11:15:10: INFO: Distro: centos7u5
11:15:10: INFO: Head node: 1 cod.medium: 4096 RAM, 2 vCPUs
11:15:10: INFO: Nodes: 5 cod.xsmall: 1024 RAM, 1 vCPUs
11:15:10: INFO: Disk: 10 GiB (type: default)
11:15:10: INFO: -----
Press ENTER to continue and create the cluster.
Press ctrl+c (or type 'a') to abort. Type 'i' for more info.
```

Ideally, if the administrator of the host cluster has set things up conveniently for the end user, the command needs no configuration file, and is run simply as `cm-cod-os cluster create`.

After a cluster has been created, it can be listed with:

```
[fred@bright91 ~]$ cm-cod-os cluster list
+-----+-----+-----+-----+-----+
| Cluster (stack) name | IP           | Head Status | Head Image           | CMD rev. |
+-----+-----+-----+-----+-----+
| c-09-05-b8.1-c7u5   | 192.168.200.8 | ACTIVE      | bcmh-centos7u5-8.1-9 | 131439    |
+-----+-----+-----+-----+-----+
```

Inactive clusters can additionally be listed by appending the `-a|--all` option to the list subcontext, as: `cm-cod-os cluster list -a`

Cluster removal is possible by specifying the cluster name to be deleted with the `delete|d|remove` option:

Example

```
[fred@bright91 ~]$ cm-cod-os cluster delete c-09-05-b8.1-c7u5
a list of resources that are to be removed is shown
You are about to delete all of the above resources.
Proceed? [yes/no] yes
...
14:18:25:      INFO: Done. Stack 'c-09-05-b81-c7u5' deleted.
```

A less drastic way to conserve resources than the `delete` option, can be to use the `shelve` option. Shelving a cluster means that it is made inactive and stored. It can then be resumed with the `start` option.

The help options for the COD contexts can be seen by appending `--help` to them

Example

```
[fred@bright91 ~]$ cm-cod-os cluster --help
```

Example

```
cm-cod-os cluster create --help
```


Cluster Extension Cloudbursting With OpenStack

7.1 Introduction

7.1.1 Cluster Extension In General

If Bright OpenStack is running on a cluster (the host cluster) that a user can access and run OpenStack instances on, then the user can become an administrator of a local OpenStack Bright cluster instance (the client cluster) running within the host cluster.

Users, instead of using up resources on their own local client cluster, may sometimes wish to extend into a second, remote, Bright OpenStack, cluster. This second cluster can then be regarded as a cloud service host that provides Bright OpenStack services.

This kind of setup is a case of running a cluster within a cloud service.

Thus, for this case, the cloud service is provided by Bright OpenStack. Analogous setups where the cloud service is provided by AWS, or Azure, can also be managed with Bright Cluster Manager (Chapter 3). In Bright jargon these cluster extension setups are conveniently abbreviated as:

- CX-OS, for cluster extension into OpenStack
- CX-AWS, for cluster extension into Amazon Web Services
- CX-Azure, for cluster extension into Microsoft Azure

This chapter explains cluster extension into OpenStack.

7.1.2 Overview Of Carrying Out Cluster Extension Into OpenStack

Bright provides a client, `cm-cluster-extension`, to set up and launch a CX-OS. It is run from the head node of a client cluster. Bursting from Bright Cluster Manager 9.1 is possible into a Bright Cluster Manager host cluster of version 9.0 that is running Bright OpenStack.

The `cm-cluster-extension` client is provided by default with Bright Cluster Manager as part of the `cm-setup` package. Its CLI options are covered in Chapter 4.1.

Running it without CLI options is recommended, and brings up an Ncurses dialog, which provides an easier option to deploy a CX-OS (section 7.2).

Another option to deploy CX-OS is via a Bright View wizard (section 7.3).

7.2 Deploying A Cluster Extension Into OpenStack Via An Ncurses Dialog

The sequence of screens that follows shows the Ncurses-based procedure for CX-OS deployment. It has some parallels with the Ncurses-based procedure for CX-AWS deployment (page 59).

Figure 7.1 shows the introductory window after running `cm-cluster-extension` as root:

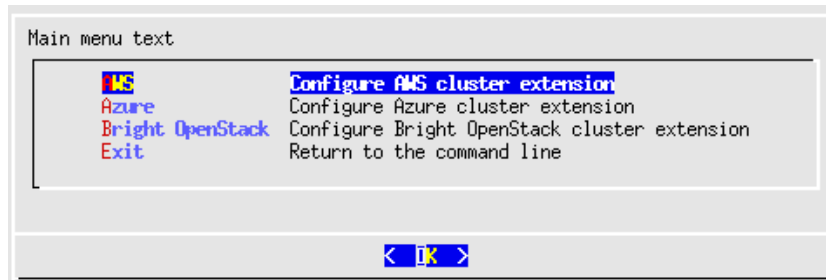


Figure 7.1: Ncurses Dialog

The type of cloud provider can be chosen. For this chapter, it means selecting Bright OpenStack as the cloud provider.

The provider instance can be added in figure 7.2. Previously added providers can also be removed in this screen (figure 7.2):



Figure 7.2: Ncurses Dialog: Add/Remove Provider

Credentials to the OpenStack instance can be configured in the screen in figure 7.3:

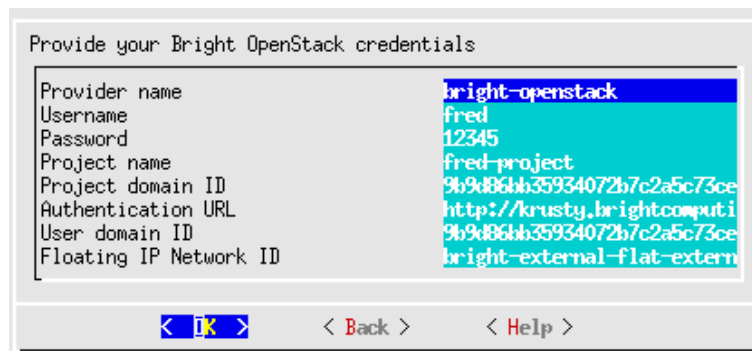


Figure 7.3: Ncurses Dialog: Add Bright OpenStack Credentials

The associated values of Authentication URL and Floating IP Network ID, in figure 7.3 are not credentials *per se*. These values should be obtained from the administrator of the Bright OpenStack cloud provider into which the bursting is being done.

A node-installer image source location should be set. This can be an image published by Bright, which if used causes the setup to download a latest node-installer image from the Bright image repository hosted on Amazon's S3 service. Another possibility is to select Image is already in OpenStack, which uses an image that is on the OpenStack host, if images have already set up there by the tenant (figure 7.4), or if set up there by another tenant and made public:

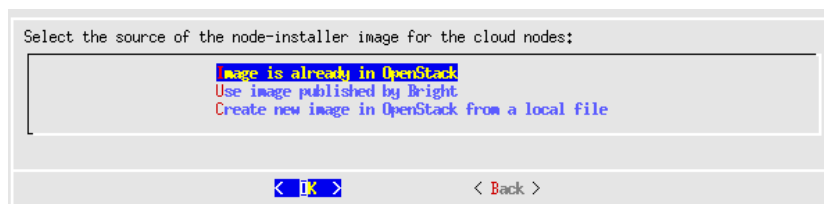


Figure 7.4: Ncurses Dialog: Select Node-installer Image Source

The node-installer image can be set (figure 7.5):

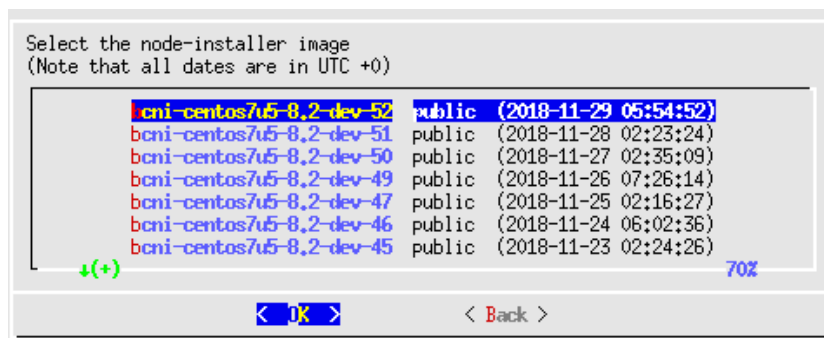


Figure 7.5: Ncurses Dialog: Select Node-installer Image

The number of cloud compute nodes available from the OpenStack host can be set (figure 7.6):

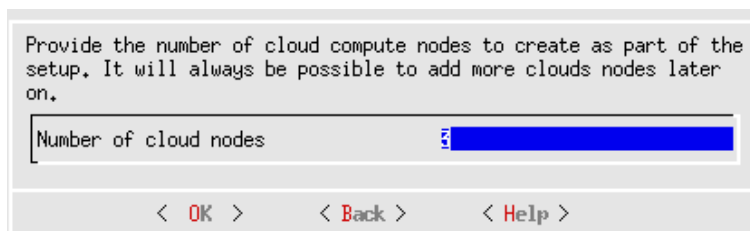


Figure 7.6: Ncurses Dialog: Setting The Number Of Cloud Compute Nodes

The cloud region (the host or hosts into which the extension takes place) can be set (figure 7.7):

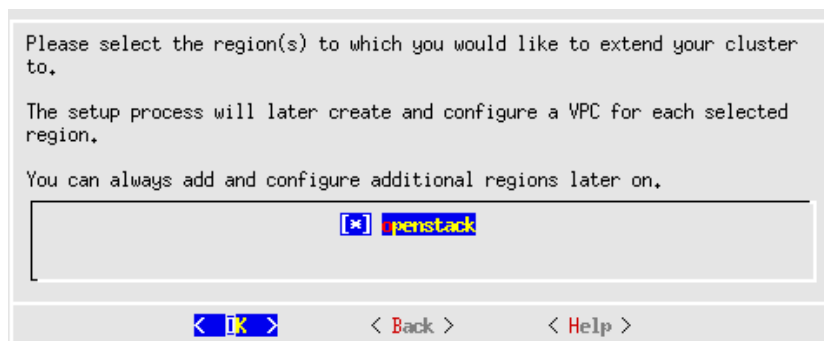


Figure 7.7: Ncurses Dialog: Cloud Region Selection

The default cloud director type can be set (figure 7.8):

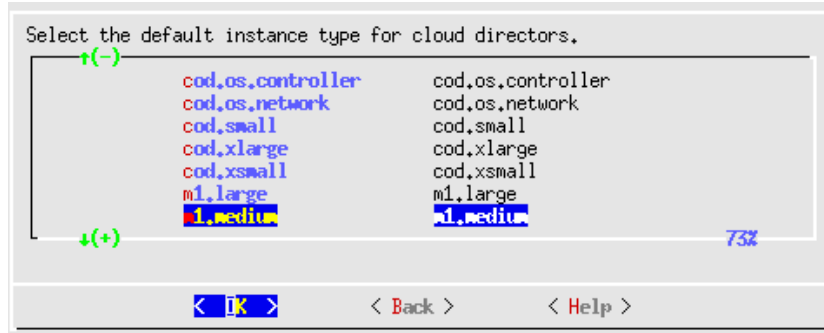


Figure 7.8: Ncurses Dialog: Default Cloud Director Instance Type

The default cloud node instance type can be set (figure 7.9):

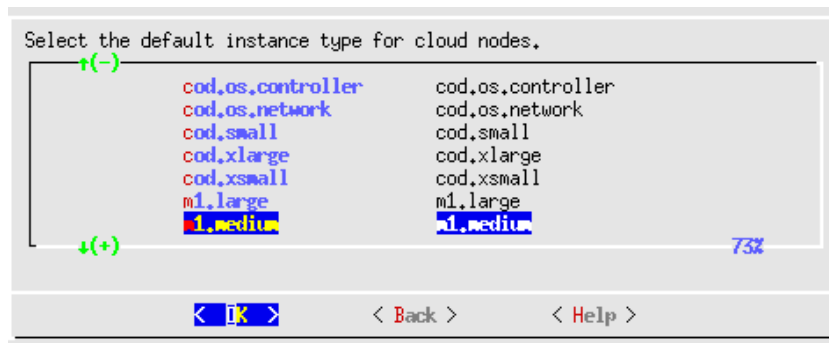


Figure 7.9: Ncurses Dialog: Default Cloud Node Instance Type

A summary of the configuration that has been done is displayed before committing (figure 7.10):



Figure 7.10: Ncurses Dialog: Summary

The configuration can be stored into a file (figure 7.11), so that it can be reused later:



Figure 7.11: Ncurses Dialog: Setting Configuration File Location

The session then finishes with output showing the progress as the configuration is carried out.

Example

```
Executing 18 stages
##### Starting execution for 'Cluster Extension'
- cloudstorage
- clusterextension
## Progress: 0
#### stage: clusterextension: Fill Out Aliases In Config
## Progress: 5
#### stage: clusterextension: Check Subnet Configuration
Connecting to CMDaemon
## Progress: 11
#### stage: clusterextension: Create Provider
## Progress: 16
#### stage: clusterextension: Wait For Provider Data
## Progress: 33
#### stage: clusterextension: Create Netmap Network
NETMAP network 'netmap' already exists. Skipping
## Progress: 38
#### stage: clusterextension: Create Tunnel Networks
## Progress: 44
#### stage: clusterextension: Create Cloud Network
## Progress: 50
#### stage: clusterextension: Configure Headnodes
Configuring tunnel interfaces for headnode
Configuring roles for headnode
Adding CloudGatewayRole to bright91
## Progress: 55
#### stage: clusterextension: Configure Regular Nodes For VPC
## Progress: 61
#### stage: clusterextension: Enable Management On Tunnel Interface
## Progress: 66
#### stage: clusterextension: bright-openstack-cloud-director
## Progress: 72
#### stage: clusterextension: Create Extension
Heat Stack 4b5d2e21-8d65-412a-8a60-6eb457772d62 is being created...
## Progress: 77
#### stage: clusterextension: Create Security Groups
Creating security group 'bright91-bright-openstack-openstack-director-sg'
Creating security group 'bright91-bright-openstack-openstack-cnode-sg'
## Progress: 83
#### stage: clusterextension: Create Director
Adding FSExport for '/cm/shared' to hosts '10.42.0.0/16' to cloud director openstack-director
Adding FSExport for '/home' to hosts '10.42.0.0/16' to cloud director openstack-director
## Progress: 88
#### stage: clusterextension: Configure Node Categories
## Progress: 94
#### stage: clusterextension: Create Cloud Node
Processing node 'openstack-cnode001'
Processing node 'openstack-cnode002'
Processing node 'openstack-cnode003'
## Progress: 100

Took:      00:05 min.
```

Progress: 100/100

Finished execution for 'Cluster Extension', status: completed

Cluster Extension finished!

After the cloud director and associated cloud compute nodes are configured, they can be powered up (some output elided):

Example

```
[root@bright91 ~]# cmsb
[bright91]# device
[bright91->device]# list
```

Type	Hostname (key)	Category	Ip	Status
CloudNode	openstack-cnode001	openstack-cloud-node	172.16.0.1	[DOWN]
CloudNode	openstack-cnode002	openstack-cloud-node	172.16.0.2	[DOWN]
CloudNode	openstack-cnode003	openstack-cloud-node	172.16.0.3	[DOWN]
CloudNode	openstack-director	bright-openstack-cloud-director	172.16.255.251	[DOWN]

```
...
[bright91->device]# power on openstack-director
cloud ..... [ PENDING ] openstack-director (Powering on)
Thu Nov 29 18:04:55 2018 [notice] bright91: openstack-director [ PENDING ] (Instance has started)
...
Thu Nov 29 18:10:33 2018 [notice] bright91: openstack-director [ UP ]
[bright91->device]# power on -n openstack-cnode00[1-3]
cloud ..... [ PENDING ] openstack-cnode001 (Powering on)
cloud ..... [ PENDING ] openstack-cnode002 (Powering on)
cloud ..... [ PENDING ] openstack-cnode003 (Powering on)
...
Fri Nov 30 12:55:37 2018 [notice] bright91: openstack-cnode002 [ PENDING ] (Instance has started)
Fri Nov 30 12:55:37 2018 [notice] bright91: openstack-cnode001 [ PENDING ] (Instance has started)
Fri Nov 30 12:55:37 2018 [notice] bright91: openstack-cnode003 [ PENDING ] (Instance has started)
...
Fri Nov 30 13:00:33 2018 [notice] bright91: openstack-cnode003 [ UP ]
Fri Nov 30 13:00:34 2018 [notice] bright91: openstack-cnode002 [ UP ]
Fri Nov 30 13:01:04 2018 [notice] bright91: openstack-cnode001 [ UP ]
```

7.3 Deploying A Cluster Extension Into An OpenStack Instance Via A Bright View Wizard

Bright View can be used to deploy a cluster extension into OpenStack via a wizard that can be run from the clickpath:

Cloud→OpenStack→OpenStack Wizard

This then brings up the following display:

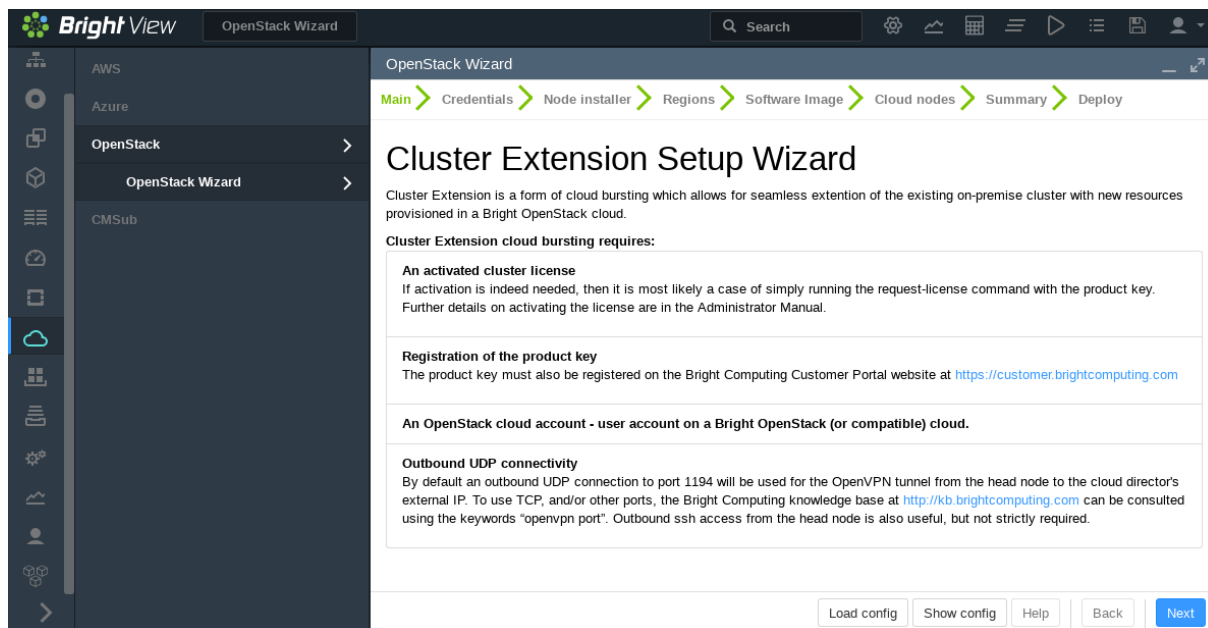


Figure 7.12: Bright View Wizard: Cluster Extension Into OpenStack Wizard Main Page

The procedure from here on is sufficiently similar to the Ncurses installation in section 7.2 that further elaboration here is not needed.

7.4 Connectivity To The Internet For OpenStack Compute Nodes

OpenStack cluster extension 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.

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 ports.

Therefore, by default for CX-OS, the port security plugin is left disabled, and the port security setting left enabled. To enable it, `cm-cluster-extension` can be run as follows:

```
cm-cluster-extension --enable-external-network-connectivity
```

For the option to work, the OpenStack instance which the cluster is being extended into should also have its Neutron ML2 port security plugin enabled:

Example

```
[clustername]% openstack; settings; networking
[clustername->openstack[default]->settings->networking]% set enableml2portsecurityplugin yes
[clustername->openstack*[default*]->settings*->networking*]% commit
```


8

Cloud Considerations And Choices With Bright Cluster Manager

8.1 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*.

8.2 Hardware And Software Availability

Bright Computing head node AMIs are available for the following distributions: RHEL7 and CentOS7.

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.

8.3 Reducing Running Costs

8.3.1 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.

8.3.2 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

```
[bright91]% device cloudsettings eu-west-1-director
[bright91->device[eu-west-1-director]->cloudsettings]% storage
[bright91->...->cloudsettings->storage]% set ebs size 15GB; commit
[bright91->...->cloudsettings->storage]% device cloudsettings cnode001
[bright91->device[cnode001]->cloudsettings]% storage
[bright91->...->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:

Devices→Cloud Nodes→Edit→Settings→Cloud settings→STORAGE→Storage→ebs→Edit→size

8.4 Setting The Cloud Node Images

The default node-installer image can be set within the cloud provider instance, for example `myam`, via selection of a predefined Amazon Machine Image (AMI):

Example

```
[bright91->cloud[myam]]% set defaultami [TAB] [TAB]
brightinstaller-005  brightinstaller-651  brightinstaller-657  brightinstaller-663  brightinstaller-669  bright
brightinstaller-006  brightinstaller-652  brightinstaller-658  brightinstaller-664  brightinstaller-670  bright
...
```

Some more details on the available images can be found by listing them:

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>

Example

```
[bright91->cloud[myam]]% images; list; exit
```

Name (key)	AMI	Paid
brightinstaller-005	ami-08e65d664f8b7d3ea (af-south-1), ami-0f482e508673790fd (ap-e+ no	
...		
brightinstaller-676	ami-0d88cb048b106a3f4 (eu-central-1), ami-0256fb6a7e380ea5d (eu+ no	
brightinstaller-678	ami-0bf32c8797a6049ba (eu-central-1), ami-01a5b4f547ec693aa (eu+ no	
marketplace-node-installer-image	ami-021a254f49625e15f (af-south-1), ami-0a6cc36cee998b556 (ap-e+ yes	

```
[bright91->cloud[amazon]]%
```

Setting a default AMI means that cluster extension cloud nodes, by default, take that AMI as their image when they boot up:

Example

```
[bright91->device[eu-west-1-cnode001]->cloudsettings]% show
```

Parameter	Value
Availability zone	eu-west-1b
Provider	myam
...	
AMI	latest:brightinstaller-678 (myam)
...	

The inherited default value can be overridden by the device by setting a value for the AMI within device mode, within the cloudsettings submode:

Example

```
[bright91->device[eu-west-1-cnode001]->cloudsettings]% set ami ami-08e65d664f8b7d3ea; commit
```

8.5 Ignoring Prepaid Node Limits With Pay-Per-Use Images

By default a user is restricted by the cloud licence from running more VMs in the cloud than a limit set within the license. This is a wise default, because as a general principle cluster administrators wish to keep costs within a planned budget.

However, there are times when a cluster administrator may wish to ignore this restriction. Ignoring the restriction is possible—at a price.

A use case for this might be a cluster that uses only a regular amount of cloudbursting most of the time, but on certain rare days it may need to burst beyond this regular amount.

Another use case might be an IT department that runs a cluster, suddenly finds itself with a large amount of unused budget, and it turns out that it is sensible to use the budget up on bursting runs.

How to allow bursting past the prepaid limit is described in the following sections for AWS (section 8.5.1) and for Azure (section 8.5.2).

8.5.1 Bursting Past The Prepaid Limit In AWS

Node-installer Image Types For AWS Cloud Nodes

Bright Cluster Manager version 9.0 introduced a way to set the node-installer image type for AWS cloud nodes. The possible cloud node node-installer image types provided by Bright Cluster Manager are:

- prepaid image type: a free node-installer image type, used to power on all nodes within the license limit. The license limit is the maximum number of images that can be run, and is the value of Pre-paid nodes in the output of the `licenseinfo` command (section 4.1.2 of the *Installation Manual*).
- pay-per-use image type: a paid node-installer image type, used to power on all nodes beyond the license limit. Using this as the node-installer image incurs additional hourly costs for the licensing part as well as for the AWS infrastructure.

Default Policy For Node-installer Image Type For AWS Cloud Nodes

A default policy to set the cloud node node-installer image type can be defined at cloud provider level using `usepaidmarketplaceamis`.

Example

```
[root@bright91 ~]# cmsh
[bright91]% cloud
[bright91->cloud[amazon]]% show
Parameter                                Value
-----
Access key ID                            *****
Account ID                              197943594779
Billing access key ID                     < not set >
Cloud job tagging                         no
Default AMI                              latest:brightinstaller-42
Default director type                     m3.medium
Default region                           eu-west-1
Default type                             m3.medium
Name                                      amazon
Revision
Secret access key                         *****
Secret billing access key                 < not set >
Tags                                      COD_PREFIX=PJ,BCM Bursting=extension,Name=bright91
Type                                      ec2
Use paid Market place AMIs               NEVER
Username                                 krusty-clusters
VPCs                                     <1 in submode>
[bright91->cloud[amazon]]% set usepaidmarketplaceamis <TAB><TAB>
always      as_needed  never
[bright91->cloud[amazon]]% set usepaidmarketplaceamis as_needed
[bright91->cloud*[amazon*]]% commit
```

There are 3 possible values for `usepaidmarketplaceamis`:

1. `never`: paid marketplace images are never used. Any attempt to power on any extra cloud nodes beyond the license limit fails.
2. `always`: The paid marketplace image is always used when powering on cloud nodes.
3. `as_needed`: The free image is used as a default so long as the prepaid node count limit has not been exceeded. Beyond the prepaid node count limit, all additional cloud nodes use the paid marketplace image.

It should be noted that once a cloud node is up, and using a paid marketplace image, then it continues to run as a paid node, even if the node count goes under the prepaid node count limit after some time. The node running a paid marketplace image only stops running as a paid node when it is terminated.

Indication that the node is a paid marketplace image: In `cmsh`, if one of the cloud nodes is using a paid marketplace image, then it is displayed tagged with the text `additional cost`, as seen in the following for `eu-west-1-cnnode002`:

Example

```
[bright91->device[eu-west-1-cnnode002]->cloudsettings]% ds
eu-west-1-cnnode001 ..... [ UP    ]
eu-west-1-cnnode002 ..... [ PENDING ] (Instance has started), additional cost
eu-west-1-cnnode003 ..... [ DOWN  ]
eu-west-1-director ..... [ UP    ]
node001 ..... [ UP    ]
node002 ..... [ UP    ]
bright91 ..... [ UP    ]
```

In Bright View, following the clickpath `Devices`, a cloud node running a paid marketplace image is tagged with a \$ sign (figure 8.1):

<input type="checkbox"/> ab-c-10-07-t-c7u5	↑	Head node	FA:16:3E:3E:D4:1B	Edit ▼
<input type="checkbox"/> eu-west-1-cnnode001	↓	Cloud node	00:00:00:00:00:00	Edit ▼
<input type="checkbox"/> eu-west-1-cnnode002 \$	⌘	Cloud node	00:00:00:00:00:00	Edit ▼
<input type="checkbox"/> eu-west-1-cnnode003	↓	Cloud node	00:00:00:00:00:00	Edit ▼
<input type="checkbox"/> eu-west-1-cstorage001	↓	Cloud node	00:00:00:00:00:00	Edit ▼
<input type="checkbox"/> eu-west-1-cstorage002	↓	Cloud node	00:00:00:00:00:00	Edit ▼
<input type="checkbox"/> eu-west-1-director	↑	Cloud node	00:00:00:00:00:00	Edit ▼

Figure 8.1: A cloud node, `eu-west-1-cnnode002`, seen running a marketplace image

First-time Use Of A Marketplace Node-installer Image For AWS Cloud Nodes

Before a marketplace image can be used for the first time, the terms of the offer must be accepted, and subscribed to.

If this is not done, then, if the administrator tries to power on a cloud node using a marketplace image via `cmsh`, an error message similar to the following appears:

Example

```
[device[eu-west-1-director]]% power on

cloud ..... [ FAILED ] eu-west-1-director
                (An error occurred (OptInRequired) when calling the RunInstances
                operation: In order to use this AWS Marketplace product you need to
                accept terms and subscribe. To do so please visit
                https://aws.amazon.com/marketplace/pp?sku=<some string>)
```

The URL that appears in the error message can be visited. At that URL the terms can be accepted and a subscription started. Then the attempt to power on a cloud node can be made once more.

8.5.2 Bursting Past The Prepaid Limit In Azure

Node-installer Image Types For Azure Cloud Nodes

In Azure, as for AWS, the possible cloud node node-installer image types provided by Bright Cluster Manager are:

- prepaid image type: a free node-installer image type, used to power on all nodes within the license limit. The license limit is the maximum number of images that can be run, and is the value of Pre-paid nodes in the output of the `licenseinfo` command (section 4.1.2 of the *Installation Manual*).
- pay-per-use image type: a paid node-installer image type, used to power on all nodes beyond the license limit. Using this as the node-installer image incurs additional hourly costs for the licensing part as well as for the Azure infrastructure.

Default Policy For Node-installer Image Type For Azure Cloud Nodes

In Azure a default policy to set the cloud node node-installer image type can be defined at cloud provider level using `usepaidmarketplaceimages`.

Example

```
[root@bright91 ~]# cmsh
[bright91]% cloud
[bright91->cloud[azure]]% show
```

Parameter	Value
Client ID	afe23456-06b9-4241-80a9-17ba71ccb911
Client secret	*****
Default Cloud Director VM Size	Standard_D2_v3
Default Location	germanywestcentral
Default VM Size	Standard_D2_v3
Default node-installer image	https://brightimages.blob.core.windows.net/images/
Extensions	<1 in submode>
Name	azure
Revision	
Subscription ID	2b8fad2b-aaf1-425a-bf45-36cfd495107e
Tags	COD_PREFIX=PJ,BCM Bursting=extension
Tenant ID	8cb88849-6e18-46d6-b0fa-551a47a31681
Type	azure
Use paid Marketplace images	NEVER

```

[bright91->cloud[azure]]% set usepaidmarketplaceimages <TAB><TAB>
always      as_needed never
[bright91->cloud[azure]]% set usepaidmarketplaceimages as_needed
[bright91->cloud*[azure*]]% commit

```

In an Azure cloud, just as for the AWS case for `usepaidmarketplaceamis`, there are 3 possible values for `usepaidmarketplaceimages` at the cloud provider level:

1. `never`: paid marketplace images are never used. Any attempt to power on any extra cloud nodes beyond the license limit fails.
2. `always`: The paid marketplace image is always used when powering on cloud nodes.
3. `as_needed`: The free image is used as a default so long as the prepaid node count limit has not been exceeded. Beyond the prepaid node count limit, all additional cloud nodes use the paid marketplace image.

Setting A Node-installer Image For Azure Cloud Nodes

It is possible to set an Azure cloud node, for example `germanywestcentral-cnnode001`, to follow a different policy by specifying it in the `cloudsettings` submode within the device mode:

Example

```
[root@bright91 ~]# cmsh
[bright91]% device
[bright91->device]% use <TAB><TAB>
germanywestcentral-cnode001  germanywestcentral-cnode003  master      node002
germanywestcentral-cnode002  germanywestcentral-director  node001     bright91
[bright91->device]% use germanywestcentral-cnode001
[bright91->device[germanywestcentral-cnode001]]% cloudsettings
[bright91->device[germanywestcentral-cnode001]->cloudsettings]% get usepaidmarketplaceimages
Never (azure)
[bright91->device[germanywestcentral-cnode001]->cloudsettings]% set usepaidmarketplaceimages <TAB><TAB>
always          as needed          follow provider  never
[bright91->device[germanywestcentral-cnode001]->cloudsettings]% set usepaidmarketplaceimages always
[bright91->device*[germanywestcentral-cnode001*]->cloudsettings*]% commit
```

This overrides the policy set at the cloud provider level, until the instance is terminated.

First-time Use Of A Marketplace Image Provided By Bright For Azure Cloud Nodes

Before using the Azure marketplace image provided by Bright for the first time, the legal terms of the subscription must be agreed to. This is done via the Azure CLI.

The Azure CLI must therefore be installed.

The agreement process is then carried out by running the following command:

```
$ az vm image terms accept --urn <publisher>:<offer>:<sku>:<version>
```

There are 4 items of information that need to be supplied to the command:

1. *publisher*: brightcomputing
2. *offer*:

In order to obtain this item of information, the available offers published by the publisher brightcomputing should be listed first:

Example

```
$ az vm image list-offers --location westus --publisher brightcomputing --output table
```

Location	Name
westus	bcmni-azure-9-0

From the list, the item corresponding to the version of the product in use should be used. Here the version is the one corresponding to Bright Cluster Manager version 9.0, and the offer is bcmni-azure-9-0.

3. *sku*: In a similar way to the previous step, the SKU is listed, and the corresponding version selected:

Example

```
$ az vm image list-skus --location westus --publisher brightcomputing --offer bcmni-azure-9-0 --output table
```

Location	Name
westus	bcm-ni-azure-9-0

Here the SKU is bcm-ni-azure-9-0.

4. *version*: latest

The agreement command in this case would then be:

Example

```
$ az vm image accept-terms --urn brightcomputing:bcmni-azure-9-0:bcm-ni-azure-9-0:latest
```

8.6 Address Resolution In Cluster Extension Networks

8.6.1 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.

8.6.2 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 3.3.9 of the *Installation Manual*), the head node is connected to `internalnet`. When a cloud service is configured, the head node is also “connected” to the CMDaemon-managed NetMap “network”. It is useful to think of NetMap as a special network, although it is actually a network mapping from the cloud to `internalnet`. That is, it connects (maps) from the nodes in one or more cloud networks such as the `us-east-1` network provided by Amazon, to IP addresses provided by `netmap`. The mapping is set up when a cloud extension is set up. With this mapping, packets using NetMap go from the cloud, via an OpenVPN connection to the NetMap IP address. Once the packets reach the OpenVPN interface for that address, which is actually on the head node, they are forwarded via Shorewall’s IPtables rules to their destination nodes on `internalnet`.

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

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

- (a) 10.141.0.1, when at the head node. The cluster manager assigns this address, which is on `internalnet`. It could also be an `ibnet` address instead, such as 10.149.0.1, if InfiniBand has been configured for the nodes instead of Ethernet.
 - (b) 172.30.0.1 when at the cloud director or regular cloud node. The cluster manager assigns this address, which is a NetMap address. It helps route from the cloud to a regular node. It is not actually an IP address on the interface of the regular node, but it is convenient to think of it as being the IP address of the regular node.
2. `cnode001`, a regular cloud node in the `us-east-1` network, is resolved for `cnode001.cm.cluster` to:
- (a) 172.21.0.1 when at the head node. The cluster manager assigns this address, which is an OpenVPN tunnel address on `us-east-1`.
 - (b) an IP address within 10.0.0.0/8 (10.0.0.1–10.255.255.254) when at a regular cloud node or at a cloud director. The Amazon cloud network service assigns the addresses in this network to the cloud director and regular cloud nodes after it notices the regular cloud node interface is up.

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

- The nodes within all available cloud networks (all networks such as for example, `us-east-1`, `us-west-1`, and so on) are given CMDaemon-assigned addresses in the cloud node space range 172.16.0.0–172.29.255.255. In CIDR notation that is: 172.16.0.0/12 (172.16.0.0–172.31.255.255), except for 172.31.0.0/15 (172.30.0.0–172.31.255.255).
- The network address space 172.30.0.0/16 (172.30.0.0–172.30.255.255) is taken by the CMDaemon-assigned NetMap network, explained shortly.
- Each node in a cloud network is also assigned an address in the network addressing space provided by Amazon VPC networking. The assignment of IP addresses to nodes within the 10.0.0.0/8 range is decided by Amazon via DHCP.

The VPC networks for regular cloud nodes and cloud director nodes are subnets in this range.

- The `netmap` “network” (figure 8.2) 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.

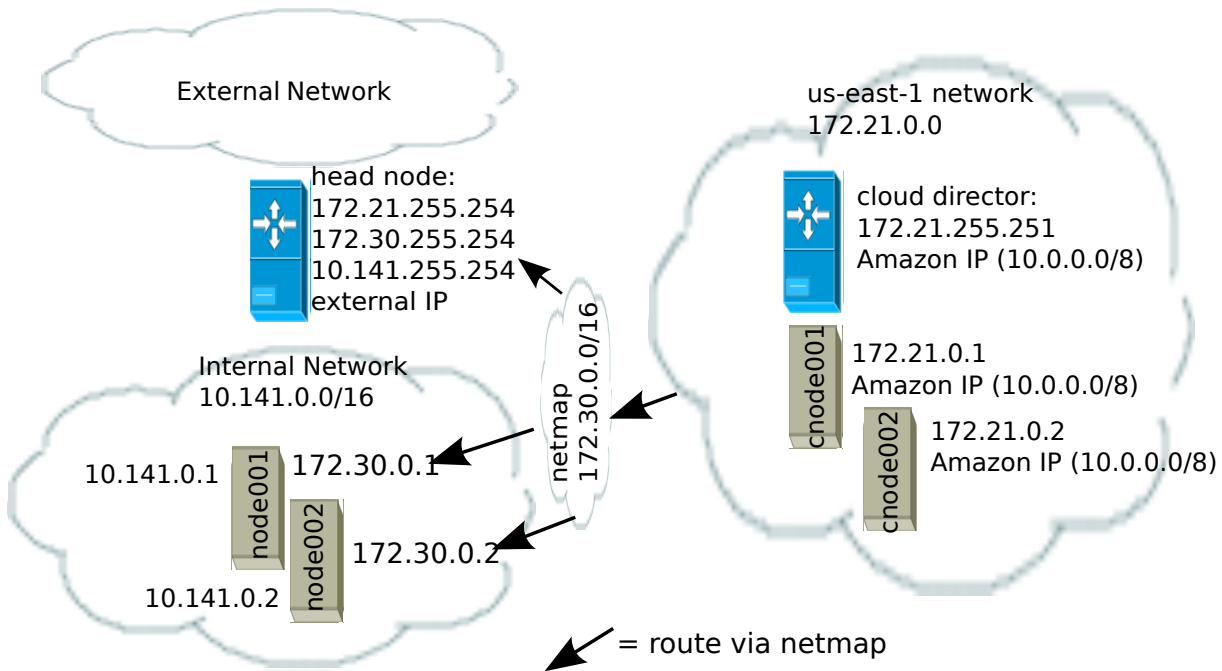


Figure 8.2: NetMap In Relation To The General Network Scheme

As pointed out in the introduction to this section (8.6), 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.

8.7 Internet Connectivity For Cloud Nodes

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

This is elaborated upon in table 8.7:

Table 8.7: Cloud compute node access to internet

Cloud node type	Internet access by default?	Details
CX-AWS	Yes	<p>Cloud compute nodes are routed via an sNAT gateway cloud director node. They therefore do not normally require assignment of Elastic IPs (section 8.9.1). 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).</p> <p>To allocate a public IP to a node, the cloud setting <code>allocatepublicip</code> is set to <code>yes</code> before creating the node. By default, cloud directors are configured to be allocated public IP addresses:</p> <p>Example</p> <pre>[bright91->device[us-east-1-director]->cloudsettings]% get allocatepublicip yes</pre>
CX-Azure	Yes	<p>Cloud compute nodes use the built-in NAT capabilities of Azure's gateway when accessing the internet.</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	<p>Cloud compute nodes use the head node as an sNAT gateway. If the head node is overloaded with network traffic, then an AWS NAT gateway can be added to the VPC, and configure that device to be the default gateway for a private subnet by modifying the routing table of that subnet.</p>
COD-Azure	Yes	<p>Cloud compute nodes can access the internet via TCP/UDP using the Azure Load Balancer. ICMP packets are silently discarded. ICMP traffic can only be allowed by associating a public IP address to the compute node. Microsoft advises its users to use an alternative to ping for connectivity testing that works using TCP packets, and tries to connect to a specific port.</p>
COD-OS	Yes	<p>Cloud compute nodes are routed via an sNAT gateway head node.</p>

8.8 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.

8.9 Setting Up And Creating A Custom VPC

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

Custom VPC for Bright Cluster Manager 9.1 subnet allocation, and allocation of EIPs (External IPs, public IP addresses) is described in sections 8.9.1–8.9.3.

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

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 can 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 “public IPs” in the cloud context. When allocating a public 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.

8.9.2 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 70 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.

8.9.3 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 Advanced settings options in the clusterextension plugin options, in 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

```
[bright91->network]% add vpc-0-sn-0
[bright91->network*[vpc-0-sn-0*]]% set type cloud
[bright91->network*[vpc-0-sn-0*]]% set baseaddress 10.0.0.0
[bright91->network*[vpc-0-sn-0*]]% set netmaskbits 24
[bright91->network*[vpc-0-sn-0*]]% commit
```

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

Example

```
[bright91->network]% clone vpc-0-sn-0 vpc-0-sn-1
[bright91->network*[vpc-0-sn-1*]]% set baseaddress 10.0.1.0
[bright91->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, and belongs to a network that contains the two subnets specified earlier.

Example

```
[bright91->cloud[Amazon EC2]->vpcs]%  
[bright91->...->vpcs]% add vpc-0  
[bright91->...->vpcs*[vpc-0*]]% set region eu-west-1  
[bright91->...*[vpc-0*]]% set baseaddress 10.10.0.0  
[bright91->...*[vpc-0*]]% set netmaskbits 16  
[bright91->...*[vpc-0*]]% set subnets vpc-0-sn-0 vpc-0-sn-1  
[bright91->...*[vpc-0*]]% commit
```