NVIDIA Base Command Manager 11

Cloudbursting Manual

Revision: d7928f6

Date: Wed Oct 21 2026

©2026 NVIDIA Corporation & affiliates. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of NVIDIA Corporation.

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 SUSE LLC. NVIDIA, CUDA, GPUDirect, HPC SDK, NVIDIA DGX, NVIDIA Nsight, and NVLink are registered trademarks of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. Green Provisioning is a trademark 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. NVIDIA Corporation shall not be liable for technical or editorial errors or omissions which may occur in this document. NVIDIA Corporation shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to NVIDIA Corporation

The NVIDIA Base Command Manager product principally consists of free software that is licensed by the Linux authors free of charge. NVIDIA Corporation shall have no liability nor will NVIDIA Corporation provide any warranty for the NVIDIA Base Command 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 NVIDIA Base Command 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 NVIDIA Base Command 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 . . . 3
0.1 About This Manual . . . 7
0.2 About The Manuals In General . . . 7
0.3 Getting Administrator-Level Support . . . 7
0.4 Getting Professional Services . . . 8

1 Introduction **9**

2 Cluster On Demand Cloudbursting With Azure, AWS, OCI, Or GCP **11**
2.1 Introduction . . . 11
2.2 Requirements For COD Cloudbursting . . . 11
2.2.1 Credentials For Azure . . . 12
2.2.2 Credentials For AWS . . . 13
2.2.3 Credentials For OCI . . . 13
2.2.4 Credentials For GCP . . . 14
2.2.5 Mandatory Inbound Rules . . . 14
2.3 COD Using The COD Command Line Tool . . . 14
2.3.1 COD Using A Docker Image . . . 15
2.3.2 COD Using A Python Package . . . 16
2.4 COD On Azure Using The Command Line . . . 18
2.4.1 Minimal Configuration File For COD On Azure . . . 18
2.4.2 Cluster Creation Run With `cm-cod-azure` . . . 18
2.4.3 Listing Clusters With `cm-cod-azure` . . . 20
2.4.4 Cluster Removal With `cm-cod-azure` . . . 21
2.5 COD On AWS Using The Command Line . . . 21
2.5.1 Minimal Configuration File For COD On AWS . . . 21
2.5.2 Cluster Creation Run With `cm-cod-aws` . . . 22
2.5.3 Listing Clusters With `cm-cod-aws` . . . 23
2.5.4 Cluster Removal With `cm-cod-aws` . . . 23
2.5.5 ARM64 Images Support In COD . . . 24
2.6 COD On OCI Using The Command Line . . . 24
2.6.1 Minimal Configuration File For COD On OCI . . . 25
2.6.2 Cluster Creation Run With `cm-cod-oci` . . . 25
2.6.3 Listing Clusters With `cm-cod-oci` . . . 27
2.6.4 Cluster Removal With `cm-cod-oci` . . . 27
2.7 COD On GCP Using The Command Line . . . 27
2.7.1 Minimal Configuration File For COD On GCP . . . 28
2.7.2 Cluster Creation Run With `cm-cod-gcp` . . . 29
2.7.3 Cluster Removal With `cm-cod-gcp` . . . 30
2.8 COD Client Configuration And Command Line Options . . . 30
2.8.1 Command Line Structure . . . 30

2.8.2 Configuration Files . . . 32
2.9 Using The AWS EC2 Management Console . . . 33
2.9.1 Status Checking Using Instance Selection From Instances List . . . 33
2.9.2 Acting On An Instance From The AWS EC2 Management Console . . . 35
2.9.3 Connecting To An Instance From The AWS EC2 Management Console . . . 35
2.9.4 Viewing The Head Node Console . . . 35
2.9.5 Security Group Configuration To Allow Access To The Head Node Using `cmsh` Or Base View . . . 36
2.10 Using The Azure Dashboard . . . 37
2.11 Using the OCI Console Dashboards . . . 38
2.12 COD: Cloud Node Start-up . . . 39
2.12.1 COD: IP Addresses In The Cloud . . . 40
2.13 COD With AWS: Optimizing AWS For High Performance Computing (HPC) . . . 40
2.14 COD And High Availability . . . 40
2.14.1 Introduction . . . 40
2.14.2 Deployment . . . 40
2.14.3 COD HA Checks And Recovery . . . 43
2.14.4 Recovery Of The Passive COD Head . . . 44
2.14.5 COD HA Deletion . . . 45

3 Cluster Extension Cloudbursting 47
3.1 Cluster Extension With AWS: The Base View Cluster Extension Wizard . . . 49
3.1.1 Introduction . . . 49
3.1.2 AWS Credentials . . . 50
3.1.3 Select Regions . . . 50
3.1.4 Select Availability Zones . . . 52
3.1.5 Select Software Images . . . 52
3.1.6 Select Cloud Types . . . 53
3.1.7 Summary & Deployment . . . 53
3.1.8 Deploy . . . 54
3.2 Cluster Extension With AWS: Cloud Director Startup From Scratch . . . 55
3.2.1 Setting The Cloud Director Disk Storage Device Type . . . 56
3.2.2 Setting The Cloud Director Disk Size . . . 57
3.2.3 Tracking Cloud Director Startup . . . 58
3.3 Cluster Extension With AWS: Cloud Node Startup From Scratch . . . 60
3.4 Cluster Extension With AWS: Cloud Director And Cloud Node Startup From Snapshots . 60
3.4.1 Cloud Director Startup From Snapshots . . . 60
3.4.2 Cloud Node Startup From Snapshots . . . 62
3.5 Cluster Extension With AWS: Optimizing AWS For High Performance Computing (HPC) 62
3.5.1 Optimizing HPC Performance: EBS Volume Type . . . 63
3.5.2 Optimizing HPC Performance: Placement Groups . . . 63
3.5.3 Optimizing HPC Performance: Disabling Hyper-Threading . . . 63
3.5.4 Optimizing HPC Performance: Using Elastic Network Adapter Instances . . . 64
3.5.5 Optimizing HPC Performance: Using A Different Clock Source . . . 64
3.5.6 Optimizing HPC Performance: Setting The Socket Buffer Sizes And TCP/IP Parameters In The Sofware Image . . . 64

4 Cluster Extension Cloudbursting With AWS Using The Command Line And `cmsh` 65
4.1 The `cm-cluster-extension` Script For Cluster Extension Clusters . . . 65
4.1.1 Running The `cm-cluster-extension` Script On The Head Node For Cluster Extension Clusters . . . 65
4.1.2 Launching The Cloud Director For Cluster Extension Clusters . . . 73
4.2 Launching The Cloud Nodes . . . 73
4.2.1 Creating And Powering Up Many Nodes . . . 74
4.3 Miscellaneous Cloud Tools . . . 74
4.3.1 Setting Exclude Lists With `excludelistsnippets` . . . 74
4.4 Connecting To AWS With Direct Connect Or A Hardware VPN . . . 76
4.4.1 Creating a VPC . . . 76
4.4.2 Connecting The Local Network To The VPC . . . 77
4.4.3 Configuring And Deploying The Cluster Extension . . . 77

5 Cluster Extension Cloudbursting With Azure 81
5.1 Introduction To Cluster Extension Cloudbursting With Azure . . . 81
5.2 Cluster Extension Into Azure . . . 81
5.3 Cluster Extension Into Azure: Cloud Node Startup From Scratch . . . 91
5.4 Cluster Extension Into Azure: `shutdown` *Vs* `power off` . . . 91
5.5 Creating An Azure Cluster Extension Using ExpressRoute Or A Hardware VPN . . . 92
5.5.1 Creating A Virtual Network . . . 92
5.5.2 Connecting The Local Network To The Virtual Network . . . 96
5.5.3 Configuring And Deploying The Cluster Extension . . . 97
5.6 Viewing And Setting Azure Generation 1 And Generation 2 Cloud Nodes . . . 98
5.6.1 Introduction . . . 98
5.6.2 Implementation Of Hyper-V Generation Choice In BCM . . . 98
5.7 Running NVIDIA A100 GPUs On Cloud Nodes . . . 100

6 Cloud Considerations And Choices With NVIDIA Base Command Manager 101
6.1 Differences Between Cluster On Demand And Cluster Extension . . . 101
6.2 Hardware And Software Availability . . . 101
6.3 Reducing Running Costs . . . 101
6.3.1 Spot Pricing . . . 102
6.3.2 Storage Space Reduction . . . 103
6.4 Setting The Cloud Node Images . . . 104
6.5 Cloud Provider Default Settings . . . 105
6.5.1 Default Settings For AWS Cloud Nodes . . . 105
6.5.2 Default Settings For Azure Cloud Nodes . . . 105
6.5.3 Default Settings For OCI Cloud Nodes . . . 106
6.6 Address Resolution In Cluster Extension Networks . . . 106
6.6.1 Resolution And `globalnet` . . . 106
6.6.2 Resolution In And Out Of The Cloud . . . 106
6.7 Internet Connectivity For Cloud Nodes . . . 108
6.8 Passing Kernel Parameters To Cloud Nodes . . . 109
6.9 Setting Up And Creating A Custom VPC . . . 110
6.9.1 Elastic IP Addresses And Their Use In Configuring Static IP Addresses . . . 110
6.9.2 Subnets In A Custom VPC . . . 110

6.9.3 Creating The Custom VPC 111

Preface

Welcome to the *Cloudbursting Manual* for NVIDIA Base Command Manager 11.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage the cloud capabilities of NVIDIA Base Command 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 NVIDIA Base Command Manager 11 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `https://docs.nvidia.com/base-command-manager`.

- 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 BCM.
- The *Edge Manual* describes how to deploy BCM Edge with BCM.
- The *Containerization Manual* describes how to manage containers with BCM.
- The *NVIDIA Mission Control Manual* describes NVIDIA Mission Control capabilities and integration with BCM.

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 BCM environment and the addition of new hardware and/or applications. The manuals also regularly incorporate feedback from administrators and users, who can submit comments, suggestions or corrections via the website

`https://enterprise-support.nvidia.com/s/create-case`

Section 14.2 of the *Administrator Manual* has more details on submitting an issue.

0.3 Getting Administrator-Level Support

Support for BCM subscriptions from version 10 onwards is available via the NVIDIA Enterprise Support page at:

`https://www.nvidia.com/en-us/support/enterprise/`

Section 14.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

The BCM support team normally differentiates between

- regular support (customer has a question or problem that requires an answer or resolution), and
- professional services (customer asks for the team to do something or asks the team to provide some service).

Professional services can be provided via the NVIDIA Enterprise Services page at:
`https://www.nvidia.com/en-us/support/enterprise/services/`

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.

NVIDIA Base Command Manager implements cloudbursting for two scenarios:

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

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

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

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

Chapter 5 looks at Cluster Extension for Azure.

Chapter 6 discusses some miscellaneous aspects of cloudbursting.

2

Cluster On Demand Cloudbursting With Azure, AWS, OCI, Or GCP

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. This chapter describes how COD can run within the following cloud service providers:

- Azure (COD-Azure)
- AWS (COD-AWS)
- Oracle Cloud Infrastructure (COD-OCI)
- Google Cloud Platform (COD-GCP)

If the credential requirements for the cloud service providers are met (section 2.2), then COD clusters can be managed (created, deleted, and so on) on all cloud platforms by using the COD command line interface tool for that cloud provider. The COD CLI tool is provided by a Docker image or a Python package (section 2.3). After a COD head node has been launched in the cloud, then compute nodes are managed by using the various management interfaces provided by NVIDIA Base Command Manager running on the head node (section 2.12).

2.2 Requirements For COD Cloudbursting

The high level requirements for COD are:

- a BCM 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 COD cluster, as described in:
  - Azure (section 2.2.1)
  - AWS (section 2.2.2)
  - OCI (section 2.2.3)
  - GCP (section 2.2.4)

2.2.1 Credentials For Azure

To use COD on Azure, an Azure account subscription is needed from Microsoft. COD-Azure requires the following associated Azure credentials:

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

These credentials allow the COD command line tool (section 2.3) to launch a COD with Azure.

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 COD AWS, an AWS subscription is needed from Amazon. COD-AWS requires the following associated AWS credentials:

- 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.

These credentials allow the COD command line tool (section 2.3) to launch a COD with AWS.

2.2.3 Credentials For OCI

To use COD on OCI, an OCI account subscription is needed from Oracle. COD-OCI requires the following OCI credentials:

- API signing Key: This key can be either an uploaded public key from cod machine, or a downloaded private key, as described at `https://docs.oracle.com/en-us/iaas/Content/API/Concepts/apisigningkey.htm#two`, in the section `How to Generate an API Signing Key`. After it is added, a configuration file is displayed.
- User's OCID: This shows up in the configuration file with a format such as:
  `user=ocid1.user.oc1..`<*unique ID*>
  where <*unique ID*> may have a format such as:

  `aaaaaaaax5552rxvs6ar4qm7kur6jq2kmhrvzmk6coqwq6b7vxocltr64ba`
- Fingerprint: This shows up in the configuration file with a format such as:
  `fingerprint=12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef`
- Tenancy's OCID: This shows up in the configuration file with a format such as:
  `tenancy=ocid1.tenancy.oc1..`<*unique ID*>
- Compartment ID: In the OCI navigation menu, the compartment ID can be found using the navigation path:
  `Identity & Security` > `Identity` > `Compartments`
  Further information on finding the OCID of a compartment can be found at:

  `https://docs.oracle.com/en-us/iaas/Content/GSG/Tasks/contactingsupport_topic-Finding_the_OCID_of_a_Compartment.htm`.

To use Oracle's `oci-cli` utility, the configuration file has to be placed in `~/.oci/config`. For the COD client, the credentials should be added to the COD configuration file as described in 2.6.1.

These credentials allow the COD command line tool (section 2.3) to launch a COD with OCI.

2.2.4 Credentials For GCP

To use COD on GCP, a GCP account (`https://cloud.google.com`) is needed. COD-GCP requires the following GCP credentials:

- `project_id`: The GCP project ID
- `head_node_service_account` The head node service account ID
- `ssh_pub_key_path`: The public SSH key of the user.
- `license_product_key`: The product key of the BCM COD cluster.
- `cod_prefix`: An arbitrary prefix used to associate the cluster with the user.
- `head_node_service_account`: The Google service account.
- `inbound_rule`= IP address, port, and protocol specification for Ingress.
- `ingress_icmp`= Ingress IP ping address.

These credentials allow the COD command line tool (section 2.3) to launch a COD with GCP (section 2.7.2).

2.2.5 Mandatory Inbound Rules

By default, a COD cluster is no longer created open to the world. Unless an inbound rule is configured for the cluster, cluster creation fails validation. The user **must** specify one of the following for the cluster:

- `inbound_rule` (and, for AWS, `inbound_network_acl_rule`) opening port 22 (SSH), or
- `inbound_rule` (and, for AWS, `inbound_network_acl_rule`) opening port 8081 (web management)

At least one of these two ports must be open. The port does not have to be open specifically to the IP address of the machine that runs the COD client. For example, creating a cluster on behalf of another user is a valid use case.

The value `{CLIENT_IP}` can be used in `inbound_rule` or `inbound_network_acl_rule` instead of a hard-coded IP address, and resolves to the public IP address of the machine running the COD client.

Examples for each cloud are given in the minimal configuration file examples in sections 2.4.1, 2.5.1, 2.6.1, and 2.7.1.

2.3 COD Using The COD Command Line Tool

There are two options for running the COD command line tool:

- The COD command line tool can be run from a Docker image:
  - The Docker instance that runs from the image can be hosted on a standalone PC that is not part of a BCM cluster. It can also be hosted by a BCM node.
  - Section 2.3.1 describes the procedure for installing and configuring COD on Docker on a BCM node in more detail.
- The COD command line tool can be run after installing a Python package in a local (virtual) environment:

- The Python package can be installed on a standalone PC. It can also be installed on a BCM node.
- Section 2.3.2 describes the procedure for installing and configuring the COD client using a Python package on a BCM node in more detail.

Once the COD client is installed and configured, the next stage is to launch the COD inside the cloud service. The following table shows the 4 cloud services that can run a COD launched from Docker, along with the utility that launches the COD:

Cloud Service	CLI utility in Docker instance that launches the COD
`Azure`	`cm-cod-azure` (page 18)
`AWS`	`cm-cod-aws` (page 21)
`OCI`	`cm-cod-oci` (page 24)
`GCP`	`cm-cod-gcp` (page 27)

2.3.1 COD Using A Docker Image

The following steps are an overview of how to start up the head node and regular nodes of the COD:

- On a standalone machine or on a BCM node, a Docker host is used to pull a BCM 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 BCM cluster. However, launching a COD Docker image from BCM is possible. In more detail, the procedure can be as follows:

- Typically Docker on BCM is installed by the cluster administrator using `cm-docker-setup` or using the Base View Docker setup wizard (section 2.1 of the *Containerization Manual*). Installing the distribution version of Docker should not be done, since it leaves out some integrations with BCM.
- After installation, the administrator should start the Docker service with:
  `systemctl start docker.service`.
- To allow a regular user to run Docker, the user can be added to the `docker` system group. For example, if BCM user management (section 6.2 of the *Administrator Manual*) has created a user `joe`, then the cluster admiinistrator can add the user to the Docker system group with:

  Example

  ```
  root@basecm11:~# usermod -a -G docker joe                #just a regular joe
  ```

- User `joe` can then load the environment module (section 2.3 of the *User Manual*) for Docker, and pull a COD client:

  Example

```
joe@basecm11:~$ module load docker
joe@basecm11:~$ docker pull  brightcomputing/cod:latest
latest: Pulling from brightcomputing/cod
...
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 COD client, 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 COD client is mounted into the container instance. The following shell aliases are recommended for each COD client:

Example

```
alias cm-cod-aws='docker run --rm -it --network host -v ~:/root brightcomputing/cod cm-cod-aws'

alias cm-cod-azure='docker run --rm -it --network host -v ~:/root brightcomputing/cod cm-cod-azure'

alias cm-cod-oci='docker run --rm -it --network host -v ~:/root brightcomputing/cod cm-cod-oci'

alias cm-cod-gcp='docker run --rm -it --network host -v ~/.config/gcloud:/root/.config/gcloud \
-v ~/.config/cm-cluster-on-demand/:/root/.config/cm-cluster-on-demand/ -v ~/.ssh:/root/.ssh \
-e LOGNAME=$LOGNAME \
brightcomputing/cod cm-cod-gcp'
```

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`.
- for the `cm-cod-gcp` alias, the extra GCP mounts are needed for authentication becuase the GCP boot procedure differs from the others, and requires `gcloud`.

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.

2.3.2 COD Using 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. A minimum Python version of 3.12 is required for this. Loading a module as the default for users (section 2.2.3 of the *Administrator Manual*) for the Python version may be helpful.

The client can be run from BCM as follows:

- A virtual environment is created to install the COD client. For example, on a regular node:

  Example

```
joe@basecm11:~$ ssh node001
joe@node001:~$ python -V
Python 3.10.12                                  <---version not high enough
joe@node001:~$ module load python312
Python 3.12.10                                  <---can be set for loading automatically
joe@node001:~$ python -m venv .venvs/cod
joe@node001:~$ source .venvs/cod/bin/activate
(cod) python@node001:~$
```

- Next, the required COD clients are installed. Possible packages that can be installed, and their associated executables, are:

Pip package	Executable
cm-cluster-on-demand-aws	cm-cod-aws
cm-cluster-on-demand-azure	cm-cod-azure
cm-cluster-on-demand-oci	cm-cod-oci
cm-cluster-on-demand-gcp	cm-cod-gcp

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

  Example

```
(cod) joe@node001:~$ pip install cm-cluster-on-demand-aws
...
```

  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@basecm11:~# python -V
Python 3.10.12
root@basecm11:~# su joe
Loading gcc/14.2.0
  Loading requirement: gmp/6.3.0 mpfr/4.2.1 mpc/1.3.1
joe@basecm11:/root$ cd
joe@basecm11:~$ python -V
Python 3.10.12
joe@basecm11:~$ source .venvs/cod/bin/activate
(cod) joe@basecm11:~$ python -V
Python 3.12.10
(cod) joe@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
```

2.4 COD On Azure Using The 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.

2.4.1 Minimal Configuration File For COD On Azure

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

The credentials and product key in the following example are dummy credentials, and must be replaced with the appropriate values for the Azure instance used by the cluster administrator. The value for `ssh_pub_key_path` is the key for connecting to the head node, which can be generated using `ssh-keygen -b 4096`.

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=aiVohwi7OhJ6igim=
azure_location=westeurope

[azure.cluster.create]
inbound_rule={CLIENT_IP},22:tcp

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

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

The `inbound_rule` setting shown opens SSH access to the machine running the COD client (section 2.2.5).

The credentials allow the COD command line tool (section 2.3) to launch a COD with Azure (section 2.4.2).

2.4.2 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
```

The BCM Azure integration makes use of several node-installer images which are published in the Azure marketplace. In order to use those images, Azure requires the administrator to accept the legal terms for those images. The code automatically detects if the terms were already accepted in the past, in which case it proceeds and creates the cluster. If the terms still need to be accepted, then several links to the legal terms are presented, along with a prompt to accept them. The output looks similar to the following (some output ellipsized):

Example

```
To use Azure VM images, you need to accept the following terms:
- License (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://download.microsoft.com/download/F/D/8/FD8BA8F2...
- Privacy Policy (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://www.brightcomputing.com/privacy-policy
- Marketplace Terms (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://mpcprodsa.blob.core.windows.net/market...
- License (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://download.microsoft.com/download/F/D/8/FD8BA...
- Privacy Policy (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://www.brightcomputing.com/privacy-policy
- Marketplace Terms (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://mpcprodsa.blob.core.windows.net/mark...
Accept (y/n)?
```

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 testclusterstoragerg68u3
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 testclusterstoragerg68u3/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://testclusterstoragerg68u3.blob.core.windows.net/images/testcluster-head-node-os-disk.vhd
09:50:21:     INFO: Deleting blob testclusterstoragerg68u3/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 Base 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 2.12.

2.4.3 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.

2.4.4 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 ~]#
```

2.5 COD On AWS Using The 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.

2.5.1 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=Oocei6eiieshahG7IiJe9QuoudOoo
aws_region=eu-west-1

[aws.cluster.create]
inbound_rule={CLIENT_IP},22:tcp
inbound_network_acl_rule={CLIENT_IP},22:tcp,ALLOW,100

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

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

The `inbound_rule` and `inbound_network_acl_rule` settings shown open SSH access to the machine running the COD client (section 2.2.5).

The example credentials and product key should be replaced with the appropriate values. The value for `ssh_pub_key_path` is the key for connecting to the head node, which can be generated using `ssh-keygen -b 4096`.

The credentials allow the COD command line tool (section 2.3) to launch a COD with AWS (section 2.5.2).

2.5.2 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 using 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 Base 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 2.12.

2.5.3 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-019dacc0877bfd71a | i-0d92887ef735f87b1 | 54.217.104.57 | ...
+----------------+-----------------------+---------------------+---------------+-...
```

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

2.5.4 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 ~]#
```

2.5.5 ARM64 Images Support In COD

COD supports `arch64` images. This allows workloads to run on ARM-based infrastructure.

A cluster with `arch64` images can be created with the `--arch=aarch64` CLI parameter. An instance type that supports `aarch64` images must be chosen for this to work.

Example

```
cm-cod-aws cc --name arm64-cluster --arch=aarch64 --head-node-type t4g.medium ...[other options]...
```

The text

...[*other options*]...

in the preceding example should be replaced with additional parameters required for the specific cluster configuration.

ARM64 Images Support Limitations

For COD:

- `aarch64` image support is available only for COD-AWS, COD-OCI, and COD-GCP. Support for other COD providers may be added in future releases.
- Mixed-architecture clusters are not supported. The head node and all compute nodes must use the same architecture-—either all `x86_64` or all `aarch64`.

2.6 COD On OCI Using The Command Line

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

The `cm-cod-oci` 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-oci cluster create -h` displays a list of options and explanations for the `cluster create` command.

2.6.1 Minimal Configuration File For COD On OCI

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

Example

```
[oci.credentials]
  oci_user=ocid1.user.oc1..<unique_ID>
  oci_fingerprint=12:34:56:78:90:ab:cd:ef:12:34:56:78:90:ab:cd:ef
  oci_tenancy=ocid1.tenancy.oc1..<unique_ID>
  oci_region=us-sanjose-1
  oci_key_file=~/.oci/oci_api_key.pem
[oci.cluster.common]
  oci_compartment_id = "ocid1.compartment.oc1..<unique_ID>
[oci.cluster.create]
  inbound_rule={CLIENT_IP},22:tcp
[cluster.create.password]
  ssh_pub_key_path=/root/.ssh/id_rsa.pub
```

The `inbound_rule` setting shown opens SSH access to the machine running the COD client (section 2.2.5).

The credentials should be replaced with the OCI credentials from 2.2. The value for `ssh_pub_key_path` is the key for connecting to the head node, which can be generated using `ssh-keygen -b 4096`.

The credentials allow the COD command line tool (section 2.3) to launch a COD with OCI (section 2.6.2).

2.6.2 Cluster Creation Run With `cm-cod-oci`

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

Example

```
cm-cod-oci cluster create \
  --head-node-number-cpus 2 --head-node-root-volume-size 50  \
  --head-node-shape VM.Standard.E4.Flex --nodes 2 \
  --node-shape VM.Standard3.Flex --node-number-cpus 2 --node-root-volume-size 50 \
  --name testcluster
```

Note: Bare metal shapes with no local disks, such as `BM.Standard3.64`, are not supported yet.

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

Example

```
[root@cod-client ~]# cm-cod-oci cluster create \
  --head-node-number-cpus 2 --head-node-root-volume-size 50  \
  --head-node-shape VM.Standard.E4.Flex --nodes 2 \
  --node-shape VM.Standard3.Flex --node-number-cpus 2 --node-root-volume-size 50 \
  --name testcluster
05:35:54:     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 us-
ing 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.
05:35:56:     INFO: Oracle Cloud Infrastructure (OCI) requires acceptance of terms & conditions ass-
ociated with relevant images whenever a new version of these images is made available. The following
terms & conditions / end user license agreements must be accepted in order to use this software:
```

```
Head node terms:
(already accepted)

Compute node terms:
- https://objectstorage.eu-marseille-1.oraclecloud.com/n/axogvvqxyiae/b/community-application
/o/eula.html
- https://cloudmarketplace.oracle.com/marketplace/content?contentId=50511634&render=inline
- https://www.oracle.com/legal/privacy/privacy-policy.html

Do you grant permission for Bright Cluster Manager to accept the required end user license
agreements and terms & conditions on your behalf when creating machine instances in OCI?
05:35:56:     INFO: Proceed? [yes/no]
yes
05:35:58:     INFO: ----------------------------------------------------------------------
05:35:58:     INFO:               Cluster:  testcluster
05:35:58:     INFO: ----------------------------------------------------------------------
05:35:58:     INFO:            Image name:  bcmh-ubuntu2004-10.0-1021(ubuntu2004-10.0:1021)
05:35:58:     INFO:            Image date:  2023-05-14 07:09 (22h 26m ago)
05:35:58:     INFO:        Package groups:  none
05:35:58:     INFO:               Version:  10.0
05:35:58:     INFO:                Distro:  ubuntu2004
05:35:58:     INFO:             Head node:  1 (VM.Standard.E4.Flex)
05:35:58:     INFO:                 Nodes:  2 (VM.Standard3.Flex)
05:35:58:     INFO:                Region:  us-sanjose-1
05:35:58:     INFO:           Compartment:  bright
05:35:58:     INFO:              Key path:  /root/.ssh/id_rsa.pub
05:35:58:     INFO:  Cloud authentication:  Head node will inherit COD credentials for cloud
authentication
05:35:58:     INFO: ---------------------------------------------------------------------
05:35:58:     INFO: Press ENTER to continue and create the cluster.
05:35:58:     INFO: Press ctrl+c (or type 'a') to abort. Type 'i' for more info.

05:36:00:     INFO: No availability domain configured; selected 'exbx:US-SANJOSE-1-AD-1'
from those in the configured compartment
05:36:01:     INFO: Creating VCN
05:36:05:     INFO: Creating subnet
05:36:07:     INFO: Creating network security groups
05:36:09:     INFO: Creating head node
05:37:18:     INFO: Waiting for sshd to start (ssh root@155.248.204.211).
05:38:26:     INFO: Waiting for CMDaemon to start.
05:39:28:     INFO: Waiting for CMDaemon to initialize.
05:41:38:     INFO: ---------------------------------------------------------------------
05:41:38:     INFO:       Cluster:  testcluster
05:41:38:     INFO: ---------------------------------------------------------------------
05:41:38:     INFO:  Head node ID:  ocid1.instance.oc1.us-sanjose-1.anzwuljricxcl6acdujsuhn
3yw6uze3h3c44jgbgitwwlqceqpdqkebzij3q
05:41:38:     INFO:     Public IP:  155.248.204.211
05:41:38:     INFO:      Key path:  /root/.ssh/id_rsa.pub
05:41:38:     INFO: ---------------------------------------------------------------------
```

At this point, the head node running on OCI is ready for use. It can be accessed using SSH, as suggested in the SSH string line of the preceding output. The regular nodes running on OCI are configured for use, but are not powered on. They can be powered on using the `cmsh` or Base View front ends, just as in a regular cluster.

In cmsh, the first cloud node can be powered on with:

Example

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

More details can be found in section 2.12.

2.6.3 Listing Clusters With cm-cod-oci

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

Example

```
[root@cod-client ~]# cm-cod-oci cluster list
05:47:06:     INFO: Listing clusters in region us-sanjose-1
+-------------+-------------------------+----------------------+---------+---------------
| Cluster Name| VCN Name                | Head node name       | State   | Shape
+-------------+-------------------------+----------------------+---------+---------------
| testcluster | on-demand testcluster-01 | testcluster-head-node | RUNNING | VM.Standard.E4
+-------------+-------------------------+----------------------+---------+---------------
```

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-oci

A cluster testcluster can be removed with a command such as:

Example

```
[root@cod-client ~]# cm-cod-oci cluster delete -- filters testcluster

05:50:45:     INFO: This will delete all OCI resources tagged as 'testcluster'; continue?
05:50:45:     INFO: Proceed? [yes/no]
yes
05:50:49:     INFO: Cluster testcluster: No cluster networks 'BCM_Cluster=testcluster'; skipping
05:50:49:     INFO: Cluster testcluster: No instance pools found matching 'BCM_Cluster=testcluster';
skipping
05:50:49:     INFO: Cluster testcluster: Terminating testcluster-head-node (instance state:
RUNNING)...
05:51:54:     INFO: Cluster testcluster: Terminated 1 instance in 64.48 seconds
05:51:54:     INFO: Cluster testcluster: No volumes found matching 'BCM_Cluster=testcluster';
skipping
05:51:55:     INFO: Cluster testcluster: Deleting subnet 'testcluster-01'
05:51:58:     INFO: Cluster testcluster: Deleting route table 'Default Route Table for on-demand
testcluster-01'
05:52:00:     INFO: Cluster testcluster: Deleting internet gateway 'testcluster-01'
05:52:02:     INFO: Cluster testcluster: No Instance Configurations found matching 'BCM_Cluster=test-
cluster'; skipping
05:52:02:     INFO: Cluster testcluster: Deleting VCN 'on-demand testcluster-01'
[root@cod-client ~]#
```

2.7 COD On GCP Using The Command Line

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

The cm-cod-gcp 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-gcp cluster create -h` displays a list of options and explanations for the `cluster create` command.

The `gcloud` Application

The `gcloud` application is a CLI application provided by Google to run its GCP services. The GCP services can be integrated with BCM by using a `gcp.ini` file to set initial values. This file can be read by `gcloud` and by the `cm-cod-gcp` client. The application can be installed on a head node, or it can be installed within a regular node image.

The cluster administrator can set the credentials for COD in GCP in `gcp.ini`

How to set up `gcloud` and have it work with BCM using credentials is described next.

Installing `gcloud`: The `gcloud` application is distributed by Google from `https://cloud.google.com/sdk/docs/install`.

A general Linux tarball is available at that URL. This can be picked up, unpacked, and installed by a non-root user:

Example

```
joe@head:~$ curl -O https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/
google-cloud-cli-linux-x86_64.tar.gz
...
joe@head:~$ tar xvzf google-cloud-cli-linux-x86_64.tar.gz
...
joe@head:~$ cd google-cloud-sdk
```

A directory `~/google-cloud-sdk` is created by the tarball extraction. The `gcloud` application is found in the `bin` directory inside that directory. An installation procedure is run in the main directory with `install.sh` script:

```
joe@head:~/google-cloud-sdk$ ./install.sh
...
Welcome to the Google Cloud CLI!
...
Modify profile to update your $PATH and enable shell command completion?

Do you want to continue (Y/n)?Y
...
```

If the defaults are used during the preceding run, then the user's `.bashrc` file is slightly modified to allow the `gcloud` binary to be included in `$PATH`. The `gcloud` binary can then be used if the user, `joe` in this case, logs into the system again.

2.7.1 Minimal Configuration File For COD On GCP

The `gcp.ini` file should be created by the user within the directory `~/.config/cm-cluster-on-demand/`

Example

```
joe@head:~/.config/cm-cluster-on-demand$ cat gcp.ini
[gcp.common]
project_id = "nv-bcmcodgcp-20240216"

[gcp.cluster.create]
head_node_service_account = ci-202@nv-bcmcodgcp-20240216.iam.gserviceaccount.com
```

```
head_node_type = e2-medium
head_node_zone = europe-west4-c
node_type = e2-medium
image_storage_location = eu
inbound_rule="87.233.215.97/32,22:tcp",
             "87.233.215.97/32,8081:tcp",
             "82.168.163.140/32,22:tcp",
             "87.213.230.195/32,22:tcp"
ingress_icmp = 87.233.215.97/32

[cluster.create]
#ssh_password_authentication = true
#log_cluster_password = true

[cluster.create.password]
ssh_pub_key_path = ~/.ssh/id_ecdsa.pub

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

[cluster.prefix]
cod_prefix = joe
```

The project ID and service account are specific to the GCP account.

The machine types available for GCP are listed at `https://cloud.google.com/compute/docs/machine-resource`.

The commented-out password authentication and logging lines are not needed if certificate-based SSH authentication is used, with `ssh_pub_key_path`.

The inbound rules are Ingress rules, and are specific for the end user, which means that they should be replaced by appropriate rules for the user. The format is a CIDR IP address, with a port number and a protocol.

The value `{CLIENT_IP}` can be used instead of a hardcoded IP address, and at least one of port 22 or port 8081 must be opened for cluster creation to succeed (section 2.2.5).

The ICMP rule is to allow pings at that server IP address.

A value for `cod_prefix` is not needed, but is almost always useful.

Getting The Authentication Token With `gcp.ini`

The `gcp.ini` file is read by running an authentication session with a `gcloud` authentication command as follows:

```
joe@head:~$ gcloud auth application-default login
```

The user is prompted to go to a long URL that uses the Google Auth Library. The library should be allowed access to what it requests.

A verification code is provided by the library. The code should be entered in the `gcloud` authentication session.

This then sets up an authentication token that is valid for 24 hours. After the token expires, then the `gcloud` authentication command can be run again to allow authentication again.

Authentication allows the COD command line tool (section 2.3) to launch a COD with GCP (section 2.7.2)

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

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

Example

```
joe@head:~$ cm-cod-gcp cluster create \
    --head-node-root-volume-size 50  \
    --head-node-type e2-medium --nodes 2 \
    --node-type e2-medium --node-root-volume-size 50 \
    --name testcluster
```

2.7.3 Cluster Removal With cm-cod-gcp

A cluster testcluster can be removed with a command such as:

Example

```
joe@head:~$ cm-cod-gcp cluster delete -- filters testcluster

05:50:45:     INFO: This will delete all GCP resources tagged as 'testcluster'; continue?
05:50:45:     INFO: Proceed? [yes/no]
...
```

2.8 COD Client Configuration And Command Line Options

All COD clients (cm-cod-azure (section 2.4), cm-cod-aws (section 2.5), and cm-cod-oci (section 2.6) cm-cod-gcp (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,azure,oci,gcp} [*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/*`
- 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@basecm11 ~]$ cm-cod-azure --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-azure 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@basecm11 ~]$ 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            | XXXXXXXXXXXXXXX (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}` 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 using the 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 Using The AWS EC2 Management Console

The recommended way of managing COD is using Base View with its public IP address, or by using `cmsh` with the head node (section 2.9.5). This section (section 2.9) 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 BCM in the following ways.

2.9.1 Status Checking Using 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 2.1).

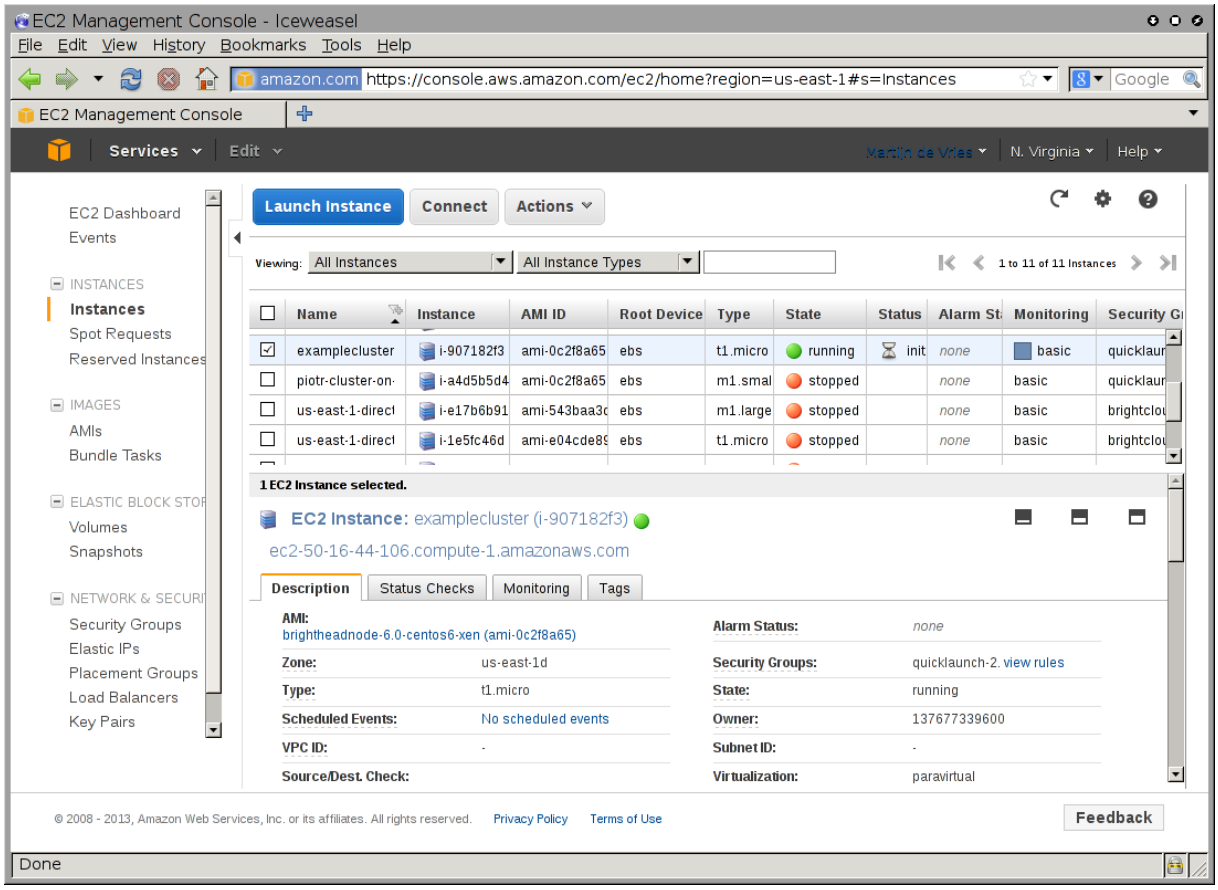

Figure 2.1: The EC2 Instances List

System (Amazon machine infrastructure) and instance (instance running under the infrastructure) reachabilities are similarly shown under the neighboring "`Status Checks`" tab (figure 2.2).

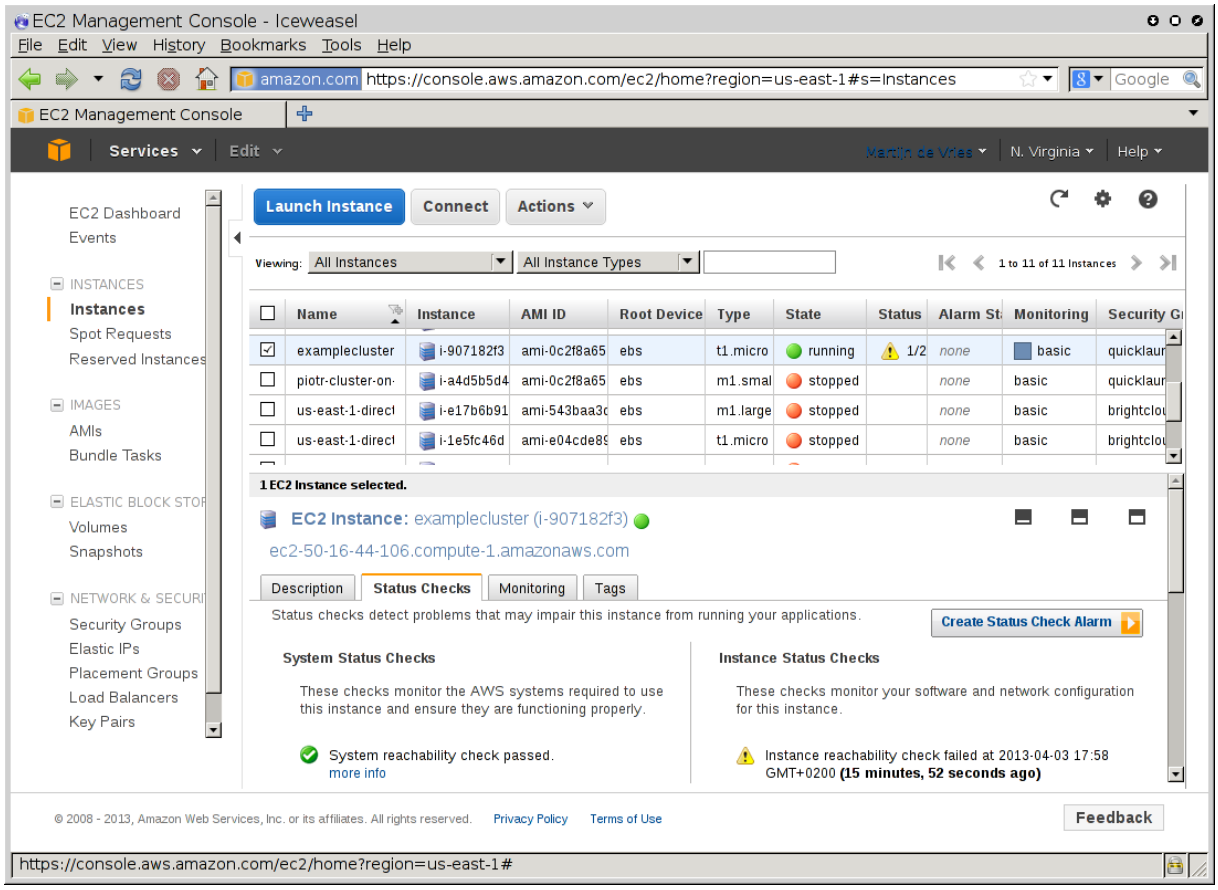

Figure 2.2: Reachability Status For An EC2 Instance

2.9.2 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.

2.9.3 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 2.5.2). 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.

2.9.4 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 2.3). A successful start of the system generates a log with a tail similar to that of figure 2.3.

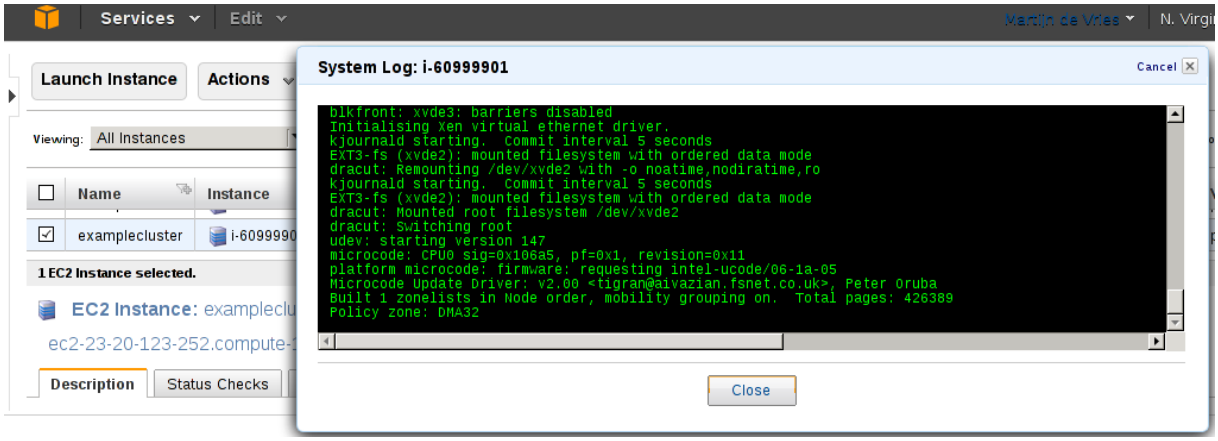

Figure 2.3: System Log Of The Checkboxed Instance

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

2.9.5 Security Group Configuration To Allow Access To The Head Node Using `cmsh` Or Base View

Amazon provides a security group to each instance, which controls network access according to the `inbound_rule` setting configured at cluster creation (section 2.2.5); the `inbound_network_acl_rule` setting configures the subnet's network ACL separately. A new security group can be configured, or an existing one modified, using the `Edit details` button in figure 2.4. Security groups can also be accessed from the navigation menu on the left side of the EC2 Management Console.

COD With AWS: Access With Base 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 2.4).

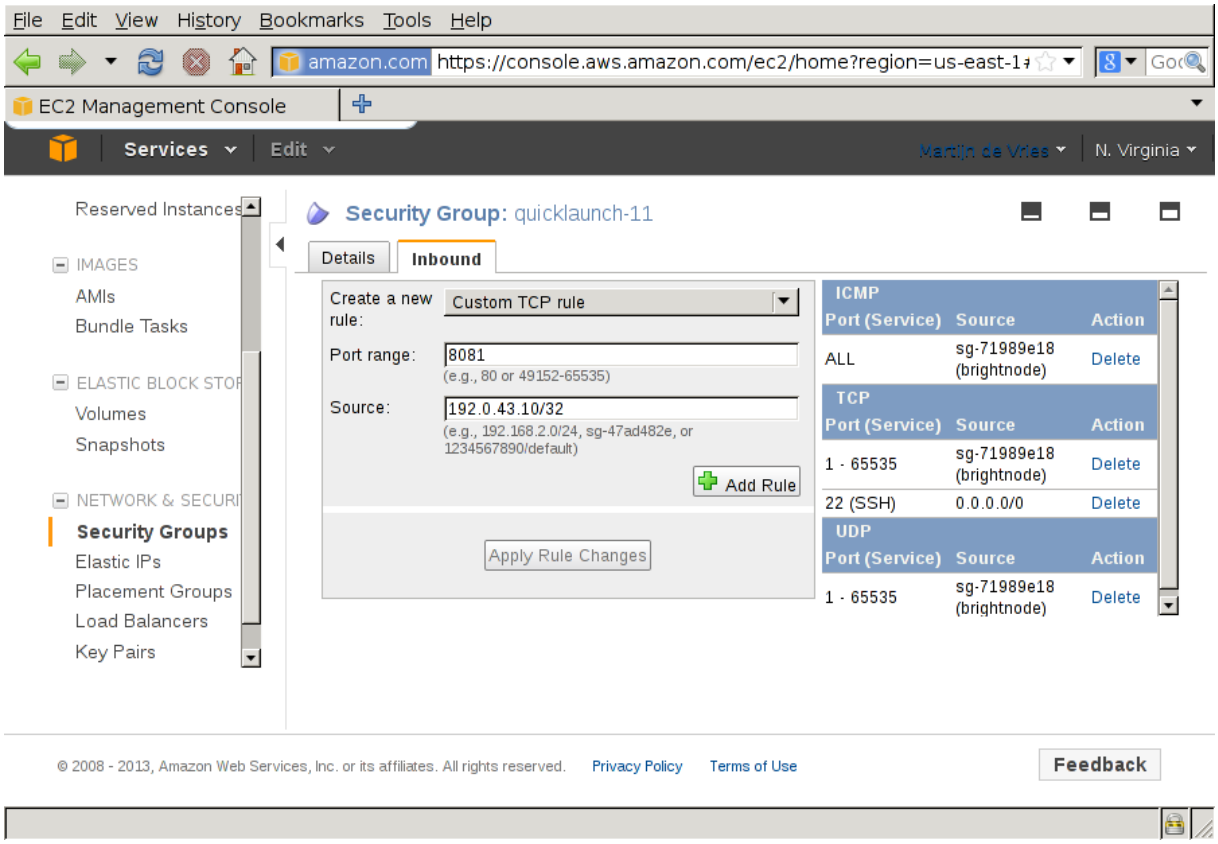

Figure 2.4: Security Group Network And Port Access Restriction

- 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.
- If the cluster was created with an inbound rule opening port 8081, that allows Base View (section 2.4 of the *Administrator Manual*) to connect to the head node. This is because the Base View back end, which is CMDaemon, communicatesusing port 8081.

COD With AWS: Access With A Local `cmsh`:
If the cluster was created with an inbound rule opening port 22, the security group created by Amazon 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.

2.10 Using The Azure Dashboard

For Azure, as is the case for AWS, the cluster is normally expected to be managed using Base View or `cmsh`. Sometimes when carrying out some special configurations, there may be a need to manage the cluster objects directly using 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 using the `Virtual machines` resource (figure 2.5), and the virtual networks associated with the COD nodes can similarly be viewed and managed using the `Virtual networks` resource.

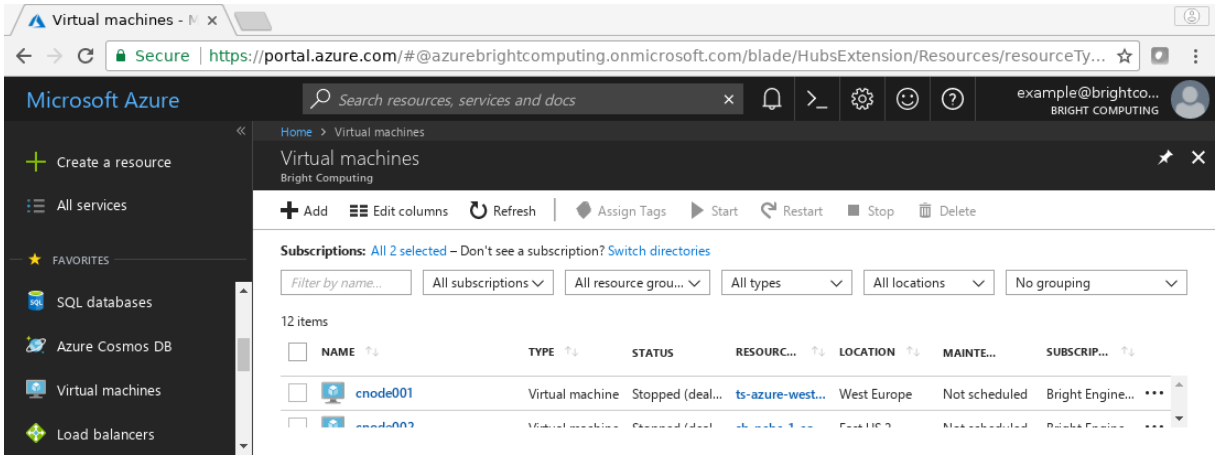

Figure 2.5: Viewing Virtual Machines In A Cluster On Demand In Azure

Some items that may be useful when accessing the portal:

- An overview of the COD head node is possible using the navigation path:

  `Home` > `Virtual machines` > <*head node*> > `Overview`

- Boot process diagnostics can be viewed using the navigation path:

  `Home` > `Virtual machines` > <*head node*> > `Boot diagnostics` > `Serial log`

- A serial console can be accessed the navigation path:

  `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.

2.11 Using the OCI Console Dashboards

For OCI, the cluster is expected to be managed using `cmsh`. Sometimes when the cluster are corrupted or it needs special configurations, cluster objects can be directly controlled using the portal at `https://www.oracle.com/cloud/`. This requires cloud account, or SSO to sign in.

For example, all the instances can be viewed using `Left Navigation` > `Compute` > `Instances` (figure 2.6)

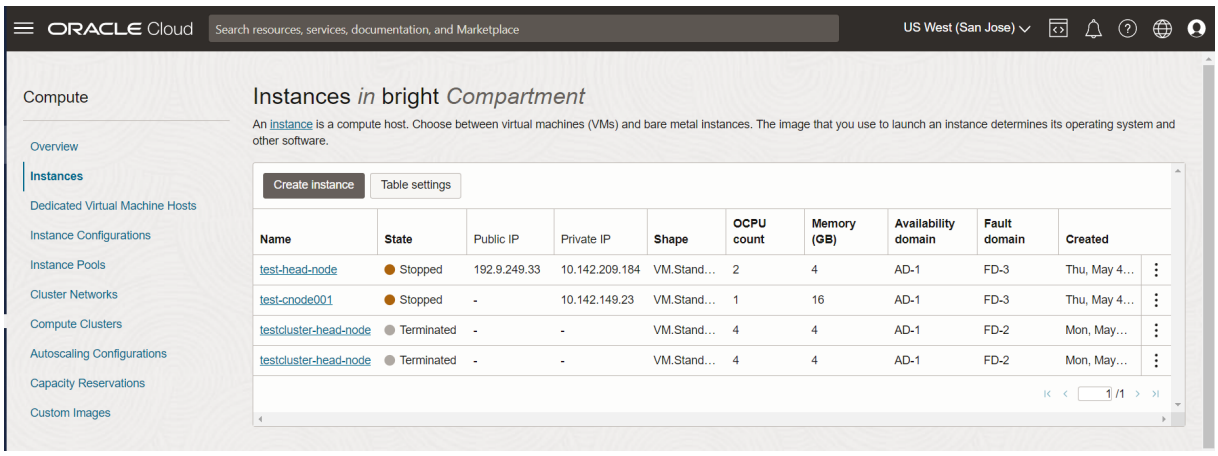

Figure 2.6: Viewing All Instances In A Cluster On Demand In OCI

2.12 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 BCM as follows:

- In Base View they can conveniently be cloned using the navigation path:

  `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 Base View using the navigation path:

`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*).

2.12.1 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, BCM 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.

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

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

2.14 COD And High Availability

2.14.1 Introduction

A COD head node, like a physical on-premises head node, can crash or somehow become non-operational. The possible reasons for this are pretty much the same in the case of a COD head node or a physical head node. The solution to reduce downtime due to these reasons is also the same: that is, to provide high availability (HA).

HA has always been available as a feature for BCM running on on-premises clusters (Chapter 15 of the *Administrator Manual*). Since 9.2, HA is also available for edge directors (section 2.1.1 of the *Edge Manual*) and for COD (this section 2.14).

The cluster administrator should have some familiarity with how HA for physical head nodes is implemented (Chapter 15 of the *Administrator Manual*) before implementing it on a COD. HA on COD is implemented very similarly to on a physical cluster, with a passive COD head node that takes over from a failing active COD head node.

COD with HA is deployed in a similar manner for AWS, Azure, and OCI.

The COD head nodes use shared storage for `cm/shared` and `/home`, just like the physical head nodes do. For COD HA head nodes, the shared storage needs to be a special filesystem in the cloud, just like the physical head HA nodes use a special filesystem.

Two shared IP addresses are also allocated—public and private. These shared IP addresses always point to the active head node, and are analogous to the external and internal IP address in the case of the physical head HA nodes (figure 15.1 of the *Administrator Manual*).

2.14.2 Deployment

COD HA Deployment Overview

A COD HA cluster is created by first creating a standard (that is: non-HA) COD cluster using `cm-cod-aws` (section 2.3), `cm-doc-azure` (section 2.4), or `cm-cod-oci` (section 2.6).

For AWS the following may be run:

Example

```
[root@basecm11 ~]# cm-cod-aws cluster create --version HEAD --name basecm11-ha
```

The procedures for Azure (with `cm-cod-azure`) and OCI (with `cm-cod-oci`) are very similar.

The standard COD cluster, which here has arbitrarily been given the name `basecm11-ha` in advance, can now be converted to an HA COD cluster.

For AWS, after the deployment of the converted cluster is complete, the original head node automatically gets the suffix `-a`, and the new head node that is brought up gets suffix `-b`.

Configuration For Deployment

The conversion is done by running the `cm-cloud-ha-setup` wizard, whch starts up a TUI for configuring the HA deployment. The `Deploy` item in the main menu starts the configuration (figure 2.7):

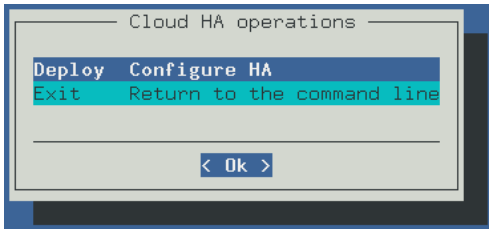

Figure 2.7: Cluster On Demand TUI Wizard: Deploy Selection

The deployment process has the cluster administrator:

- provide the product key, so that the MAC address of the secondary COD head node can be added to the license (figure 2.8):

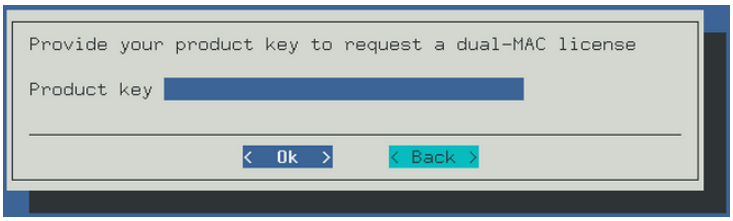

Figure 2.8: Cluster On Demand TUI wizard: deploy selection

- decide using using a screen on whether to auto-allocate a private IP address to be shared between the head nodes, or whether to specify the private address (figure 2.9):

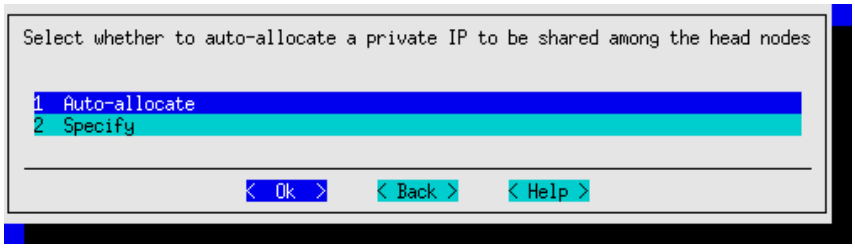

Figure 2.9: Cluster On Demand TUI wizard: decide on how private IP address is a created

- if specifying the private address is chosen, then the private IP address can be set (figure 2.10):

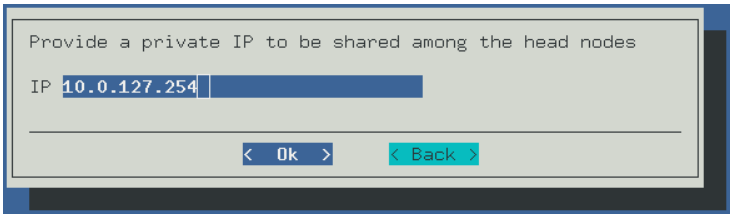

Figure 2.10: Cluster On Demand TUI wizard: private IP address allocation

- decide on where to deploy shared storage (figure 2.11):

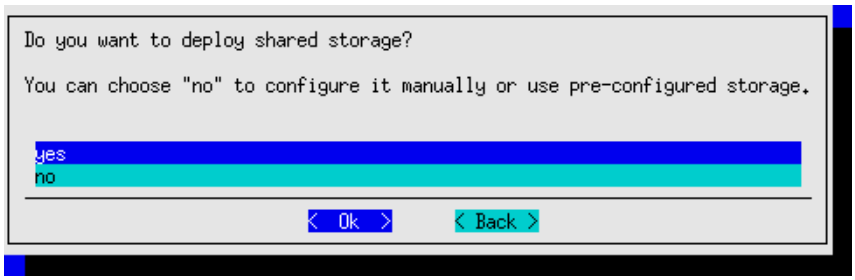

Figure 2.11: Cluster On Demand TUI wizard: decide on where the shared storage should be

  - If `yes` is selected, then the shared storage is created by the wizard creating an AWS EFS instance and sharing the `/home` and `/cm/shared` partitions among the cluster nodes using NFS
  - If `no` is selected, then the shared storage should be specified manually, or a preconfigured storage can be used

- decide on how to set up a NAT gateway (figure 2.12):

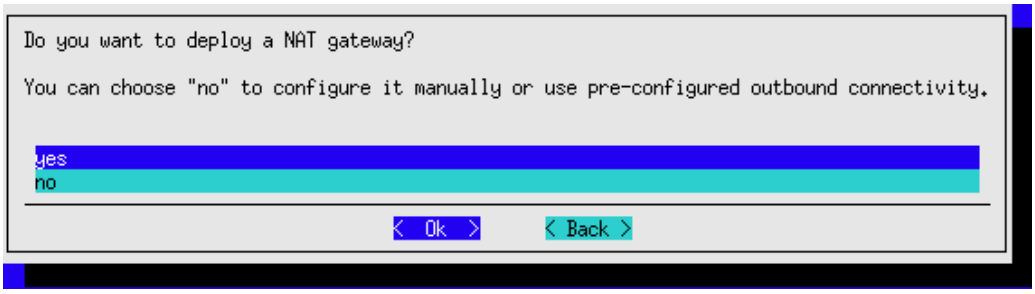

Figure 2.12: Cluster On Demand TUI wizard: decide on how to set up a NAT gateway

- decide on setting up the shared IP address as public or private (figure 2.13):

Figure 2.13: Cluster On Demand TUI wizard: decide on how to set up the shared IP address

- save the configuration and deploy the conversion (figure 2.14):

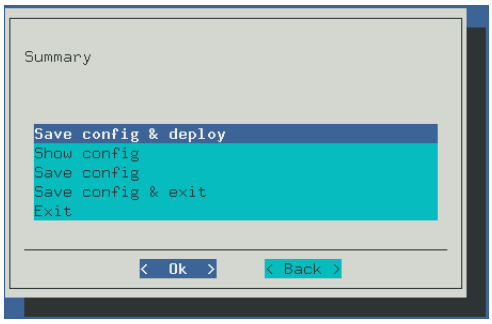

Figure 2.14: Cluster On Demand TUI wizard: saving the configuration, as part of deployment

Conversion To HA

The deployment that converts the cluster to HA then takes place by bringing up a second HA node in the cloud. Deployment typically takes about 50 minutes. The progress can be followed on the console, and in the logs `/var/log/cmha.log` and `/var/log/failoversetup.log`.

If the deployment is not yet complete, then the `cmha setup` command indicates that with, for example:

```
root@basecm11-ha-a:~# cmha status
No HA status available.
Unable to get passive head node, is failover setup complete?
```

or later on, during the `Finalizing HA stage`, with:

```
root@basecm11-ha-a:~# cmha status
Unable to connect to cluster
Internal error, unable to connect to cluster: [Errno 111] Connection refused
Internal error, unable to connect at local cluster: [Errno 111] Connection refused
```

2.14.3 COD HA Checks And Recovery

COD HA Head Status Check

Once the deployment using `cm-cloud-ha-setup` is complete, the HA status of the cluster can be checked:

```
[root@basecm11-ha-b ~]# cmha status
Node Status: running in active mode

basecm11-ha-b* -> basecm11-ha-a
  mysql          [  OK  ]
  ping           [  OK  ]
  status         [  OK  ]

basecm11-ha-a -> basecm11-ha-b*
  mysql          [  OK  ]
  ping           [  OK  ]
  status         [  OK  ]

[root@basecm11-ha-b ~]#
```

COD Failover Session

If the active head node fails, then the passive head node is activated automatically by reassigning the shared public and private IP addresses to the passive head node, and marking the passive head node as active.

This process is called failover and can be triggered automatically or manually (section 15.1.7 of the *Administrator Manual*).

A manual failover can be carried out with `cmha makeactive`:

```
[root@basecm11-ha-b ~]# cmha makeactive

==========================================================================
This is the passive head node. Please confirm that this node should become
the active head node. After this operation is complete, the HA status of
the head nodes will be as follows:

basecm11-ha-b will become active head node (current state: passive)
basecm11-ha-a will become passive head node (current state: active)
==========================================================================

Continue(c)/Exit(e)? c

Initiating failover............................ [  OK  ]

basecm11-ha-b is now active head node, makeactive successful

[root@basecm11-ha-b ~]# cmha status
Node Status: running in active mode

basecm11-ha-b* -> basecm11-ha-a
  mysql          [  OK  ]
  ping           [  OK  ]
  status         [  OK  ]

basecm11-ha-a -> basecm11-ha-b*
  mysql          [  OK  ]
  ping           [  OK  ]
  status         [  OK  ]

[root@basecm11-ha-b ~]#
```

Managing HA and manual failover is described in more detail in section 15.4 of the *Administrator Manual*.

2.14.4 Recovery Of The Passive COD Head

Unlike with physical HA head nodes, the recovery of a passive head node in the cloud is automated.

Recovery can be carried out by running `cm-cloud-ha-setup` wizard, and selecting the `Recovery` item in the main menu (figure 2.15):

Figure 2.15: Cluster On Demand TUI Wizard: Recovery Selection

The product key is needed to license the new head node (figure 2.16):

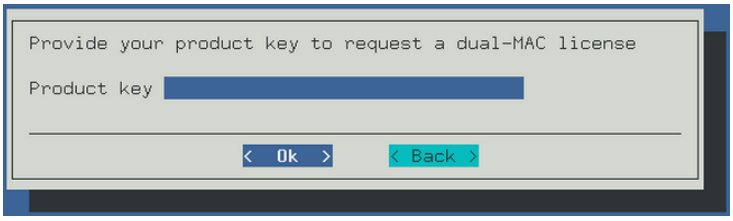

Figure 2.16: Cluster On Demand TUI Wizard: Product Key Selection

The recovery process warns that all resources associated with the old head node are to be deleted (virtual machine, disks, IP addresses) (figure 2.17):

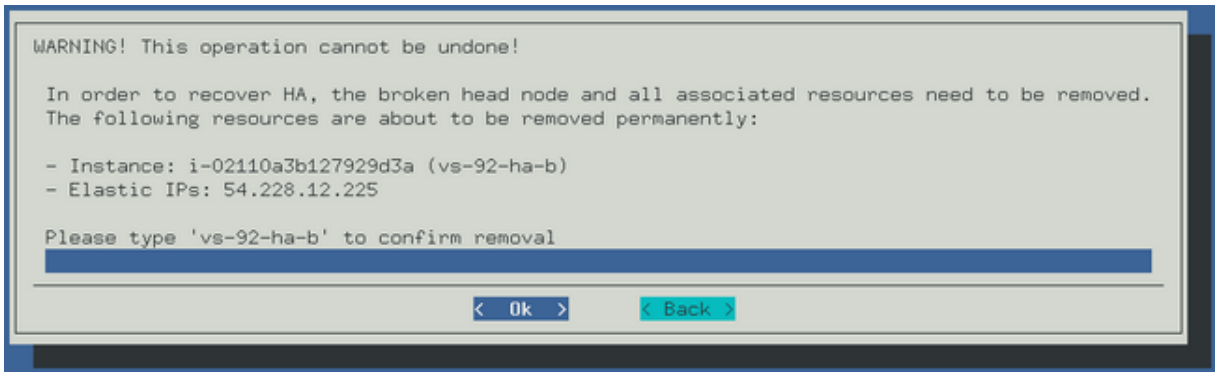

Figure 2.17: Cluster On Demand TUI Wizard: Cleanup

It then clones the active head node, configuring the clone to become the new passive head node, and the cluster then resumes normal HA operation.

When recovery is completed, the new passive head node elastic IP for the cluster called <*cluster_name*> can be obtained by running:

`cm-cod-aws cluster list [<cluster_name>]`

The `cm-cod-aws` command in the preceding text can be replaced ty `cm-cod-azure` or `cm-cod-oci` in order to get a similar listing for COD Azure or COD OCI clusters.

The output from `cmha status` can be used to confirm recovery is complete.

2.14.5 COD HA Deletion

The COD clusters for AWS listing might show something like (some output truncated):

```
[root@basecm11 ~]# cm-cod-aws cluster list
20:10:08:     INFO: Listing clusters in region eu-west-1
+--------------+----------------------+------------------------+--------------------+---------+-----------
| Cluster Name | VPC ID               | Head node ID           | Cluster IP         | State   | Type
+--------------+----------------------+------------------------+--------------------+---------+-----------
| basecm11-ha  | vpc-01d547e708c9eec9c | i-0224f8b38fe0b8658 (A) | 54.155.161.116 (A)  | running | t3.medium
|              |                      | i-0f906293daa687d20 (B) | 52.213.160.8 (B)    |         |
|              |                      |                         | 52.211.148.112 (HA) |         |
+--------------+----------------------+------------------------+--------------------+---------+-----------
```

The COD cluster can be specified for removal with the `delete` option. For example, with AWS:

Example

```
[root@basecm11 ~]# cm-cod-aws cluster delete basecm11-ha
20:11:43:  WARNING: Some clusters have a Cloud HA EFS storage. It will also be deleted
20:11:43:     INFO: This will destroy resources associated with clusters: 'basecm11-ha'.
```

```
              Would you like to continue?
20:11:43:     INFO: Proceed? [yes/no]
yes
20:11:53:     INFO: Deleting EFS fs-0dee13aa1d1a2d337...
20:12:16:     INFO: Finding instances for cluster 'basecm11-ha'...
20:12:16:     INFO: Issuing termination requests for 2 instances for cluster 'basecm11-ha'...
20:12:17:     INFO: Waiting until instances of cluster 'basecm11-ha' are terminated...
...
20:16:40:     INFO: Done destroying cluster 'basecm11-ha' resources in VPC 'on-demand basecm11-ha'
```

3

Cluster Extension Cloudbursting

Cluster Extension cloudbursting ("hybrid" cloudbursting) in NVIDIA Base Command 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*.

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@basecm11 ~]# cmsh -c "main; licenseinfo"
License Information
--------------------------------  ----------------------------------------
Licensee                          /C=US/ST=NY/L=WS/O=BCM
                                  Mordor/OU=Mt. Doom/CN=BCM HEAD Cluster
Serial number                     1768388
Start time                        Mon Oct 16 01:00:00 2022
End time                          Fri Dec 31 23:59:00 2038
Version                           10.0 and above
Edition                           Advanced
Type                              Commercial
Licensed nodes                    6 / 100
Licensed accelerators             0 / 0
Accelerators details              6 * 0 gpu
Allow edge sites                  Yes
MAC Address / Cloud ID            FA:16:3E:91:14:86
```

  The value of `End time`, and `Type` in the preceding license should be checked.

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 Knowledge Articles in the NVIDIA Enterprise Support Portal at `https://enterprise-support.nvidia.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 BCM 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 926 of the *Administrator Manual*), which is a CMDaemon directive that can be set and activated (page 905 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 Base 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

BCM 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 BCM can optionally be configured to work with the hardware VPN using the cluster extension configuration wizard. The wizard can also guide the cluster administrator on VPN configuration for AWS (section 4.4) or Azure (section 5.5).

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 Base View Cluster Extension Wizard

Cluster Extension with the Base 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 Base View. This can be done via Base View (`https://<`*head node address*`>:8081/base-view/`), and then selecting the navigation path `Cloud` > `AWS` > `AWS Wizard`. A screen introducing the `AWS Wizard` is then displayed.

The wizard goes through the following stages:

1. Introduction (section 3.1.1)
2. AWS Credentials (section 3.1.2)
3. Select Regions (section 3.1.3)
4. Summary & Deployment (section 3.1.7)
5. Deploy (section 3.1.8)

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.

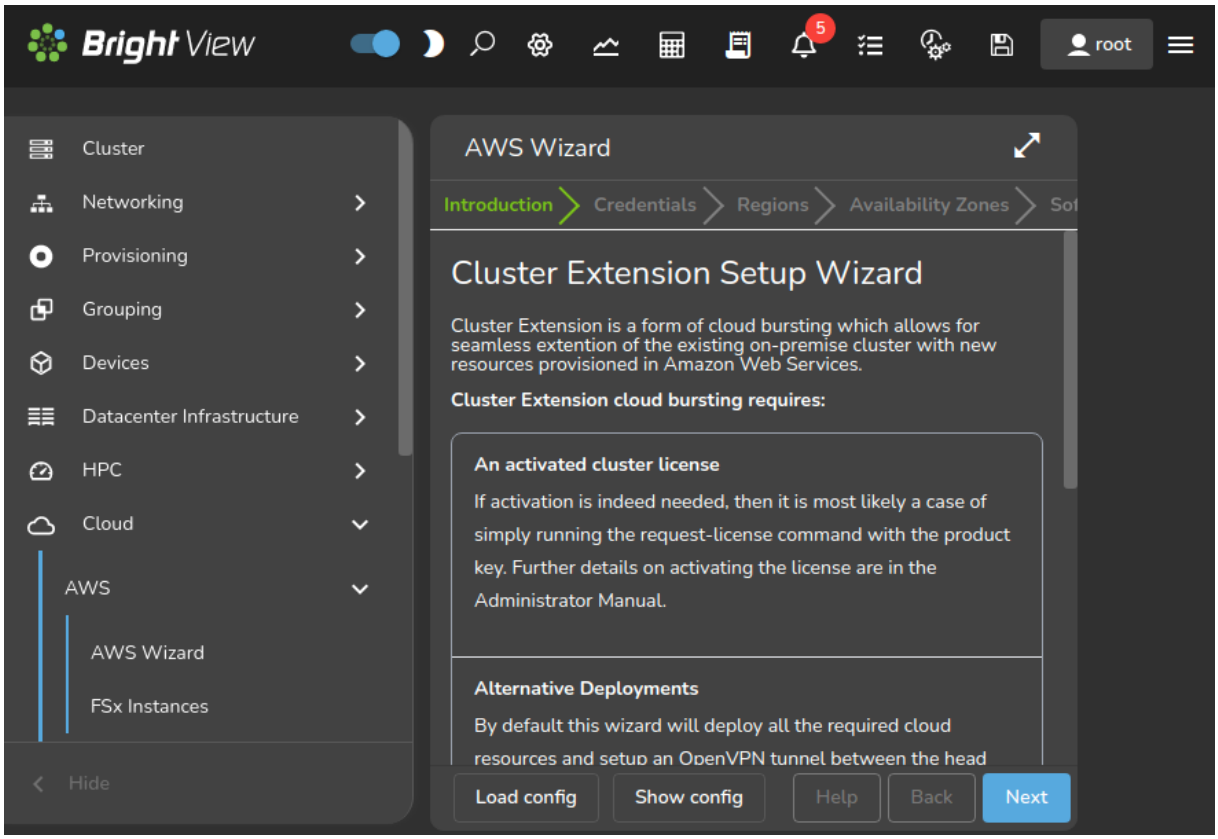

Figure 3.1: Introduction Page For Cluster Extension With Base View

During the wizard session:

- The `Load config` button allows the wizard to load a previously-saved YAML configuration text file from a file location accessible by the browser.

- The `Show config` button allows the wizard to show a YAML configuration text file. The file can be saved to a location by the browser, by using a `Save` button that is available when the configuration is shown.

Saving the configuration at the start and end of a wizard run is usually convenient for an administrator who wishes to record the changes that are being made.

3.1.2 AWS Credentials

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

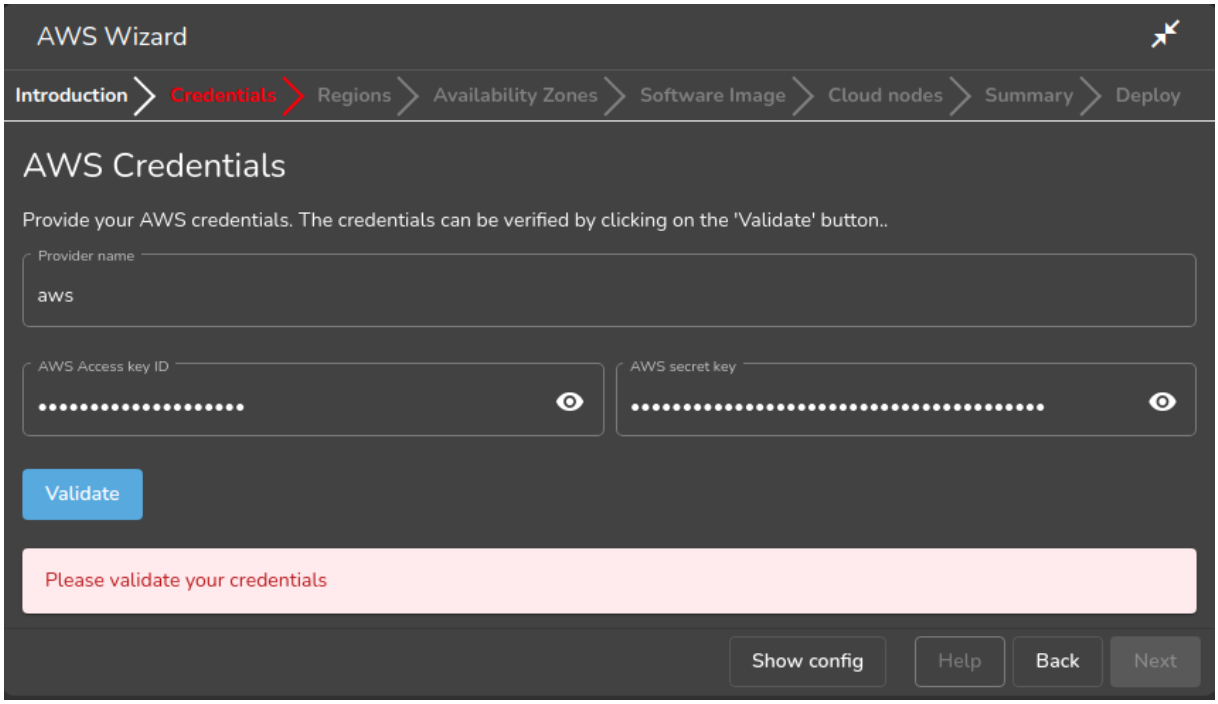

Figure 3.2: AWS Key Configuration For Cluster Extension With Base View

This asks for:

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

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

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

3.1.3 Select Regions

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

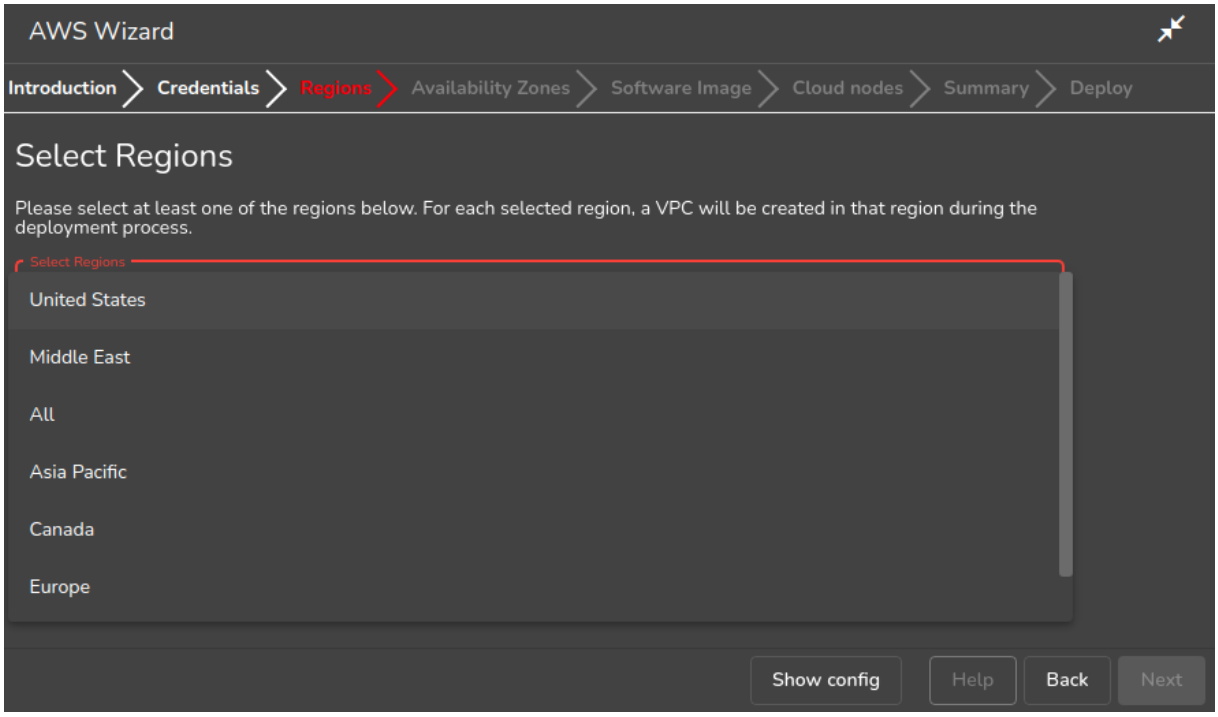

Figure 3.3: Selecting Regions With The Cluster Extension Wizard With Base View

The regions are designated by codes, for example `us-east-1`, `eu-west-1`, and so on. The first two letters associated with the region imply a major region as follows:

- `af`: Africa
- `ap`: Asia Pacific
- `ca`: Canada
- `cn`: China
- `eu`: Europe
- `me`: Middle East
- `sa`: South America
- `us`: United States

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 Availability Zones

The `Select Availability Zones` screen (figure 3.4) allows subnets in the region to be specified for the private and public subnets of the cluster extension.

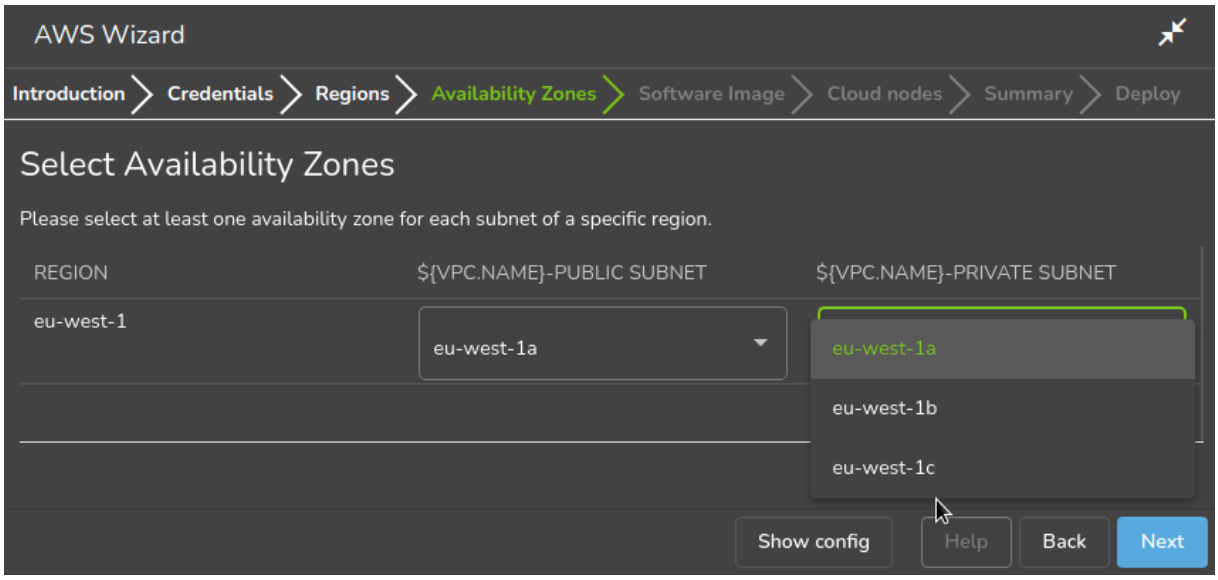

Figure 3.4: Selecting Availability Zones With The Cluster Extension Wizard With Base View

3.1.5 Select Software Images

Next, the `Select Software Images` screen (figure 3.5) 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.

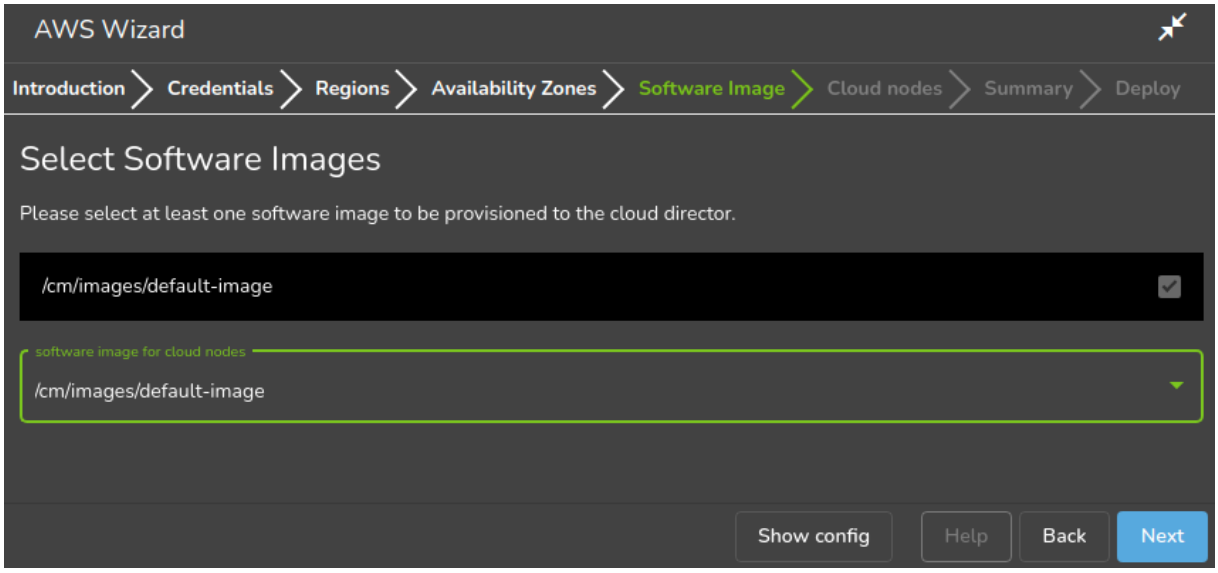

Figure 3.5: Selecting Images For The Cloud Director Cloud Nodes With The Cluster Extension Wizard With Base View

3.1.6 Select Cloud Types

Next, the default cloud director type and default cloud node type can be selected (figure 3.6) . The type specifies the virtual hardware type.

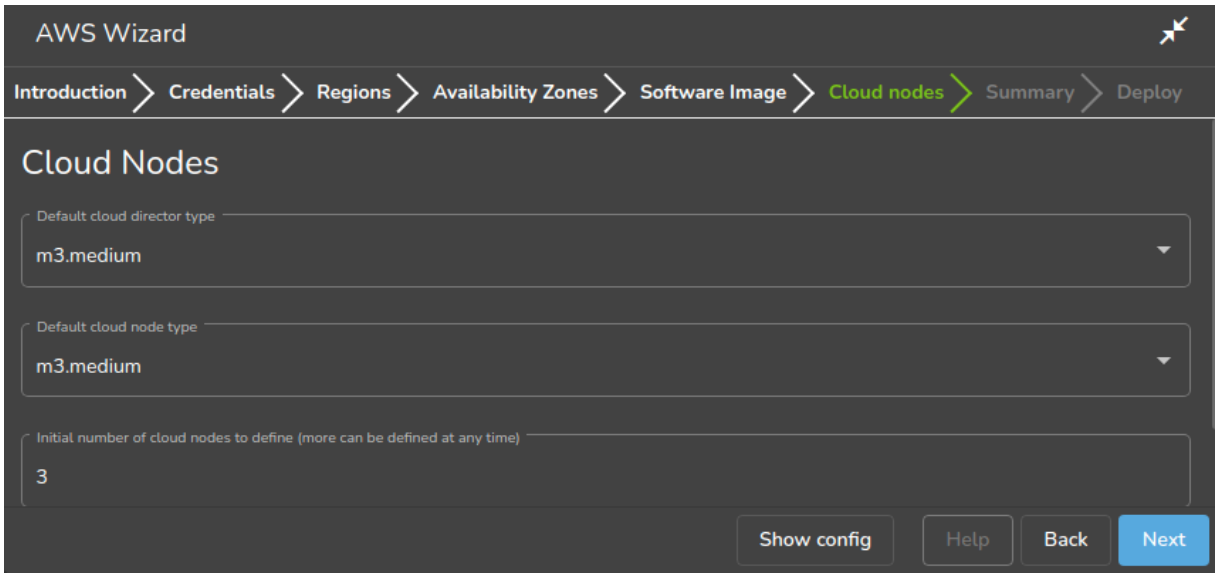

Figure 3.6: Selecting Cloud Types With The Cluster Extension Wizard With Base View

3.1.7 Summary & Deployment

The next screen is a `Summary` screen (figure 3.7).

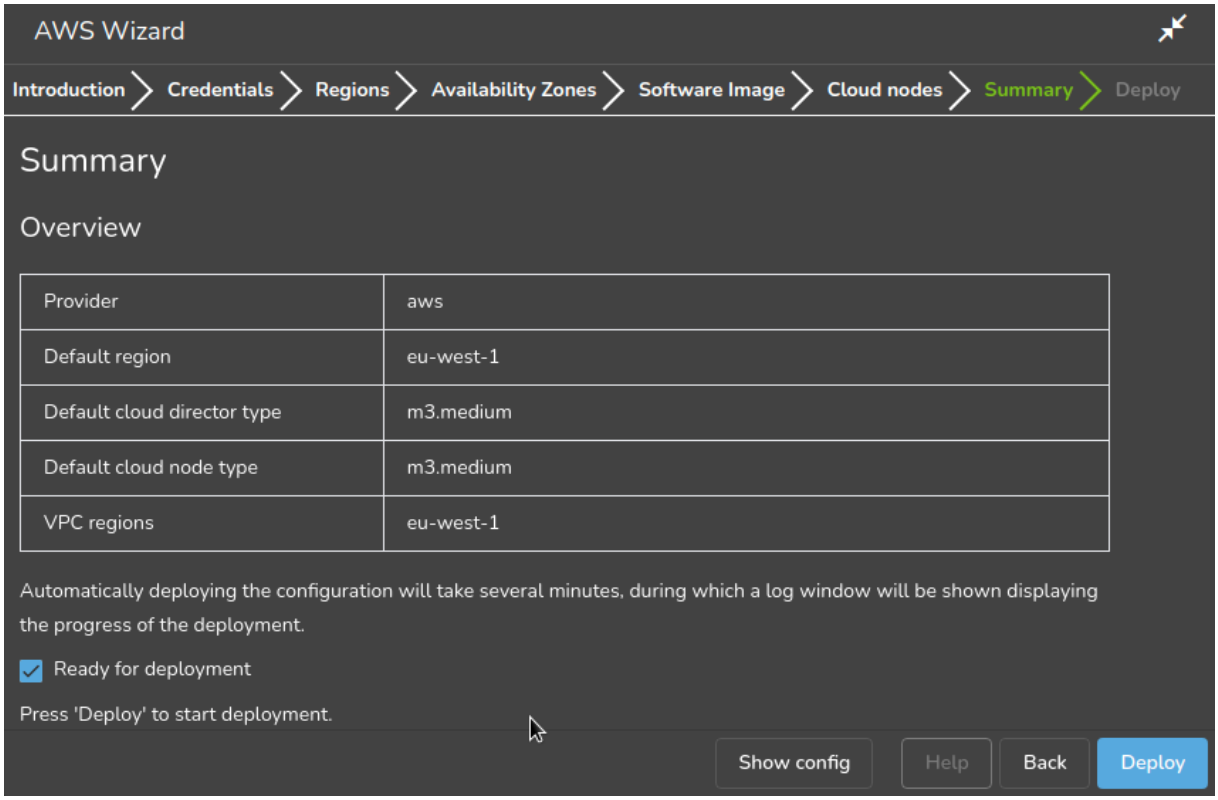

Figure 3.7: Summary Screen For The Cluster Extension Wizard With Base View

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.

3.1.8 Deploy

During configuration deployment, a progress meter indicates what is happening as the configuration is processed. Logs showing the more detailed steps can also be viewed. At the end of processing, the display indicates with the `Deployed with success` message that the cluster has been extended successfully (figure 3.8).

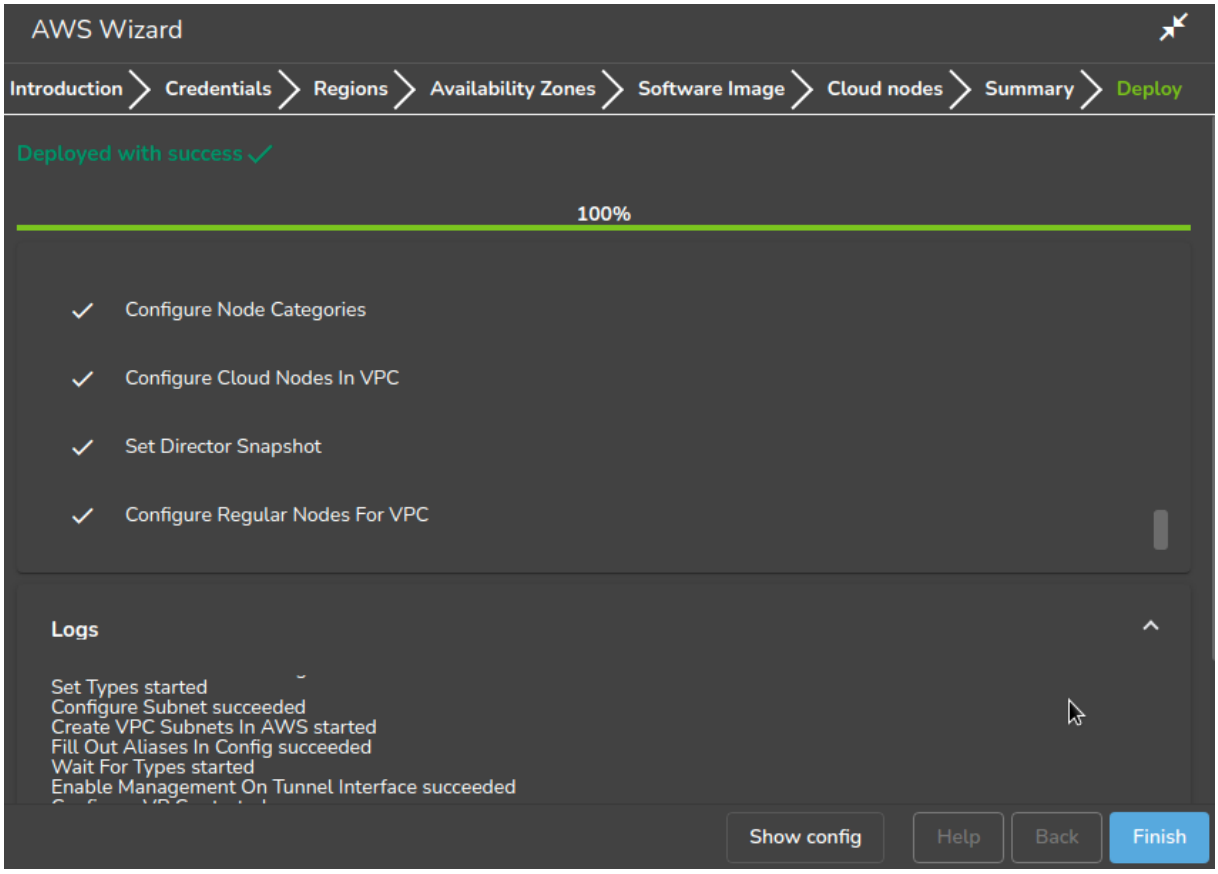

Figure 3.8: Cluster Extension Configuration Processing With Base View

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 Base View event log viewer (section 10.2.11 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 active per region for a cluster. 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 Base 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 navigation path 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 navigation path:

`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 BCM. 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 Base 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 navigation path:

  `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

```
[basecm11]% device use us-east-1-director
[basecm11->device[us-east-1-director]]% cloudsettings
[basecm11->device[us-east-1-director]->cloudsettings]% storage
[basecm11->...->cloudsettings->storage]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- ------- ----------
ebs        ebs          sdb       42GB
ephemeral  ephemeral0   sdc       0B      ephemeral0
[basecm11->...->cloudsettings->storage]% remove ebs
[basecm11->...->cloudsettings*->storage*]% set ephemeral0 drive sdb
[basecm11->...->cloudsettings*->storage*]% list
Type       Name (key)   Drive     Size    Volume ID
---------- ------------ --------- ------- ---------
ephemeral  ephemeral0   sdb       0B      ephemeral0
[basecm11->...->cloudsettings*->storage*]% commit
```

3.2.2 Setting The Cloud Director Disk Size

The disk size for the cloud director can be set in Base 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 3.13.3 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 2.2. The Instance ID that is used to identify the instance can be found

- with Base View, within the navigation path
  Devices > Cloud Nodes > <*cloud director*> > Edit > Settings > Cloud settings > Instance ID

- with cmsh, by running something like:

  Example

```
[basecm11]% device use us-east-1-director
[basecm11->device[us-east-1-director]]% get cloudid
i-f98e7441
[basecm11->device[us-east-1-director]]% cloudsettings
[basecm11->device[us-east-1-director]->cloudsettings]% get instanceid
i-f98e7441
```

Tracking Cloud Director Startup From Base View

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

- watching the icon changes for the cloud node states in the navigation path 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@basecm11 ~]# 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@basecm11 ~]# 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@basecm11 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        4
Source node:          basecm11
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@basecm11 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        5
Source node:          basecm11
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 245 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 navigation path is:

`Devices` > `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 3.9).

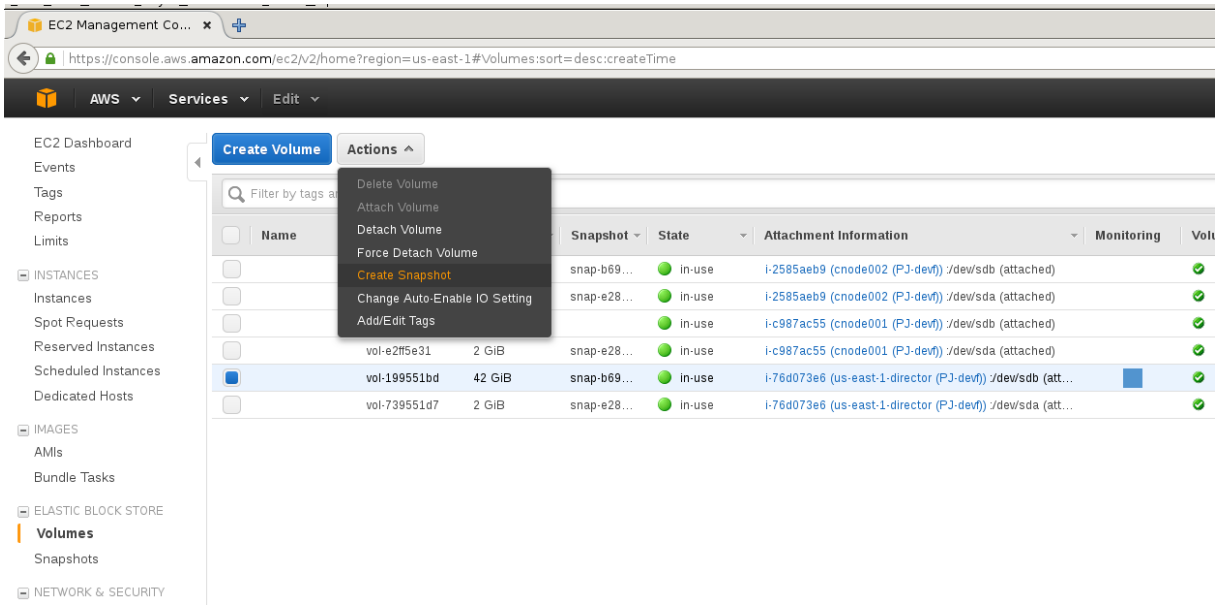

Figure 3.9: Creating A Snapshot From A Selected Volume

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

  ```
  [basecm11->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@basecm11 ~]# cmsh
  [basecm11]% device
  [basecm11->device]% use us-east-1-director
  [basecm11->device[us-east-1-director]]% cloudsettings
  [basecm11->...-director]->cloudsettings]% storage
  [basecm11->...-director]->cloudsettings->storage]% use ebs
  [basecm11->...-director]->cloudsettings->storage[ebs]]% set snapshotid snap-2a96d0c6
  ```

The cloud director can now be powered on:

Example

```
[basecm11->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

  ```
  [basecm11->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

  ```
  [basecm11->device[cnode001]->cloudsettings->storage[ebs]]% set volumetype io1
  ```

- In Base View, the equivalent navigation path 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 navigation path 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

```
[basecm11->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 (Amazon Machine Instance) used has the Elastic Network Adapter (ENA) support attribute set. By default, all AMIs provided by NVIDIA Base Command 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/cl*/cl*/current_clocksource
```

3.5.6 Optimizing HPC Performance: Setting The Socket Buffer Sizes And TCP/IP Parameters In The Sofware 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@basecm11 ~]# 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 BCM 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 CLI mode (page 65) , or as a TUI dialog (page 67).

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

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

```
[root@basecm11 ~]# cm-cluster-extension -h

usage: Cluster Extension cm-cluster-extension [-c <config_file>] [--remove] [--remove-provider <provider_name>]
                                              [--remove-region [<provider_name>.]<region_name>]
                                              [--terminate-instances] [--remove-fsx-instances]
                                              [--yes-i-really-mean-it] [--test-networking] [--test-environment]
                                              [--test-configuration] [--test-everything]
                                              [--enable-external-network-connectivity]
                                              [--azure-cloud-name AZURE_CLOUD_NAME] [-v] [--no-distro-checks]
                                              [--json] [--output-remote-execution-runner]
                                              [--on-error-action debug,remotedebug,undo,abort] [--skip-packages]
                                              [--min-reboot-timeout <reboot_timeout_seconds>] [--dev] [-h]


optional arguments:
  -h, --help            Print this screen
```

```
common:
  Common arguments

  -c <config_file>      Load runtime configuration for plugins from a YAML config file

removing cluster extension:
  Flags which can be used for removing CX

  --remove              Remove definitions of all objects required for cluster extension, e.g. cloud nodes,
                        directors, cloud networks and cloud interfaces
  --remove-provider <provider_name>
                        Remove specified CX provider
  --remove-region [<provider_name>.]<region_name>
                        Remove specified CX region
  --terminate-instances
                        Terminate all non-terminated VMs.
  --remove-fsx-instances
                        Remove FSX instances.
  --yes-i-really-mean-it
                        Required for additional safety

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 above mentioned checks.


cluster extension to Azure:
  Options specific to this cloud type.

  --azure-cloud-name AZURE_CLOUD_NAME
                        Used for deploying to a non-public Azure Cloud. E.g.: AzureUSGovernment

advanced:
  Various *advanced* configuration options flags.

  -v, --verbose         Verbose output
  --no-distro-checks    Disable distribution checks based on ds.json
  --json                Use json formatting for log lines printed to stdout
  --output-remote-execution-runner
                        Format output for CMDaemon
  --on-error-action debug,remotedebug,undo,abort
                        Upon encountering a critical error, instead of asking the user for choice, setup
                        will do selected action.
  --skip-packages       Skip the any stages which install packages. Requires packages to be already
                        installed.
  --min-reboot-timeout <reboot_timeout_seconds>
                        How long to wait for nodes to finish reboot (default and minimum allowed: 300
                        seconds).
  --dev                 Enables additional command line arguments
```

```
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 --yes-i-really-mean-it
    (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, and uses if found, the following environment variables:
   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@basecm11 ~]# 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 TUI

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

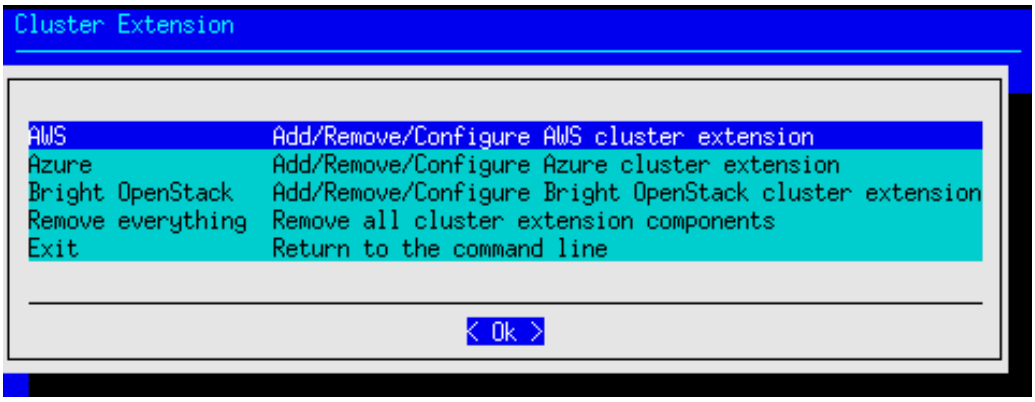

Figure 4.1: Configuration Processing With cm-cluster-extension: Main Screen

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. If an AWS region has been configured previously, then its configuration can be removed. Testing only the network connectivity to the various AWS regions is also possible (figure 4.2).

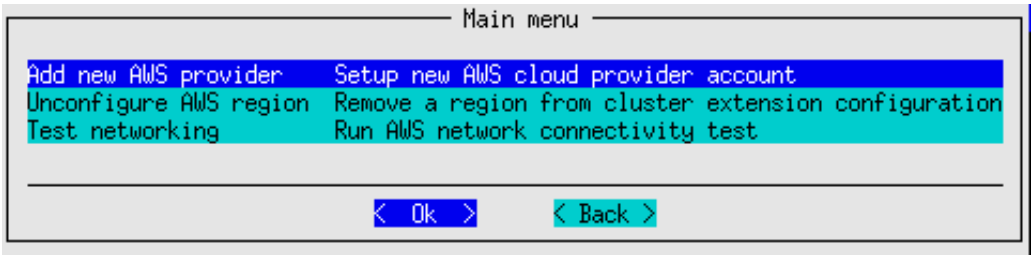

Figure 4.2: Configuration Processing With cm-cluster-extension: Add, Unconfigure, Or Test

If a new AWS provider is selected, then a screen comes up asking for the credentials for Amazon (figure 4.3):

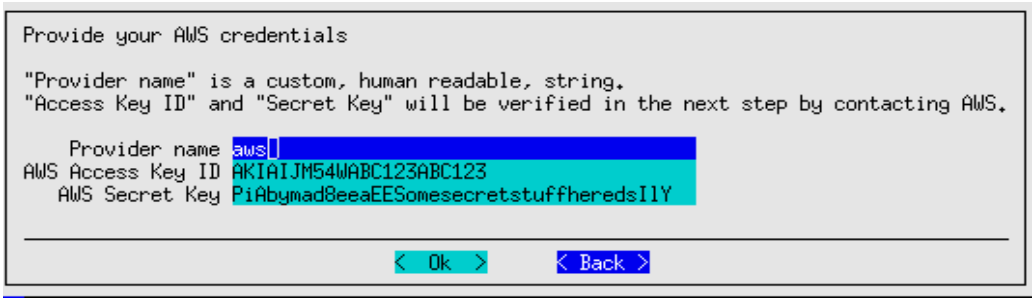

Figure 4.3: Configuration Processing With cm-cluster-extension: Obtaining Credentials

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

After checking and accepting the credentials, the administrator is asked to choose a setup type for the wizard. This can be a default setup, or it can be an advanced setup (figure 4.4):

Figure 4.4: Configuration Processing With cm-cluster-extension: Default Setup Or Advanced Setup

The default setup is recommended for most use cases. The default setup configures an OpenVPN network connection to the cluster extension. The advanced setup allows the network connection to the cluster extension to use Direct Connect for AWS, or another hardware-based VPN.

After a default or advanced setup has been chosen, the initial number of cloud nodes to be set up in the cloud can be set (figure 4.5). A default of 3 is suggested.

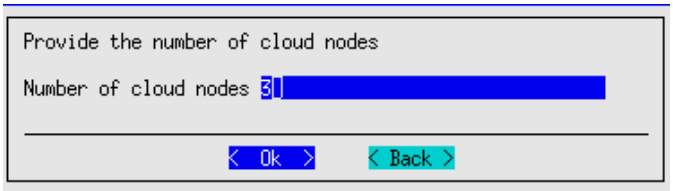

Figure 4.5: Configuration Processing With cm-cluster-extension: Setting The Initial Number Of Cloud Nodes

After setting an initial number of cloud nodes, the major regions into which these can be deployed are displayed (figure 4.6):

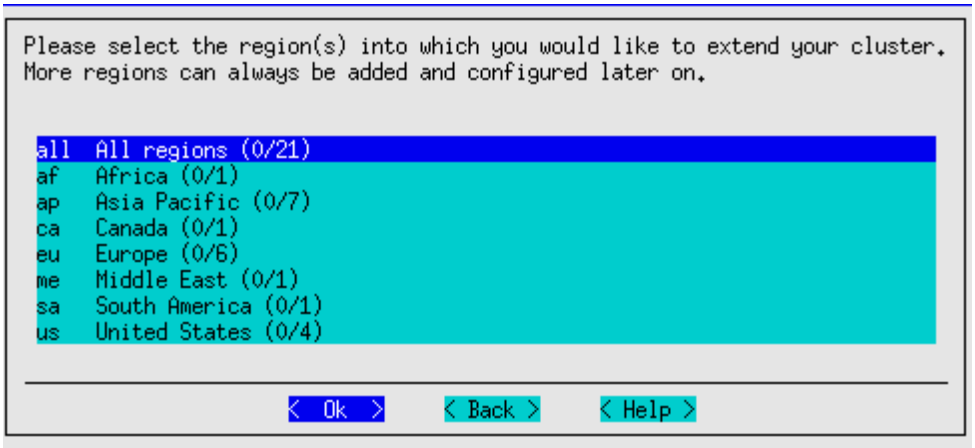

Figure 4.6: Configuration Processing With cm-cluster-extension: Major Regions Selection

After selecting a major region, the available regions under it into which the cloud nodes can be deployed are displayed (figure 4.7):

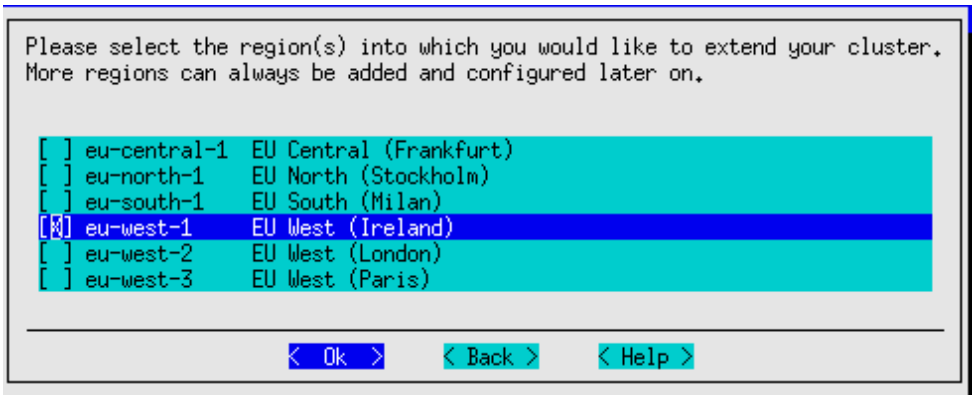

Figure 4.7: Configuration Processing With cm-cluster-extension: Regions Selection

After selecting one or more regions, the default region into which the cloud nodes can be deployed can be set (figure 4.8):

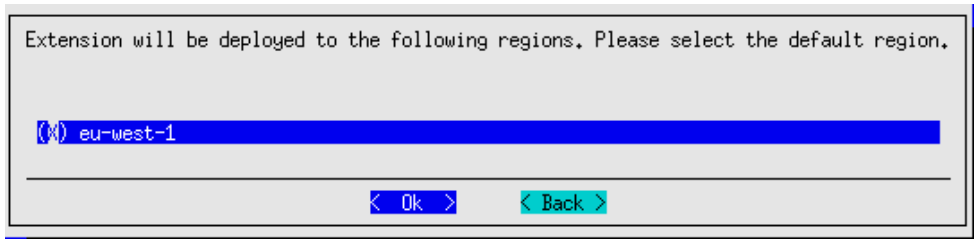

Figure 4.8: Configuration Processing With `cm-cluster-extension`: Default Region Selection

After setting the default region, the availability zone must be set for the public subnets (figure 4.9):

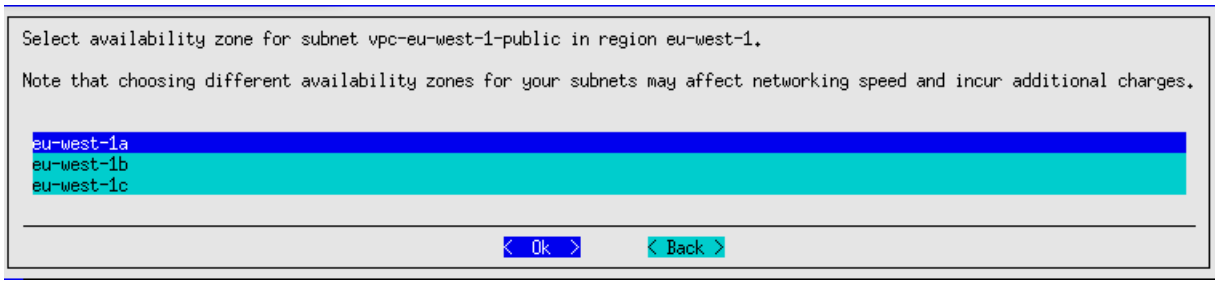

Figure 4.9: Configuration Processing With `cm-cluster-extension`: Availability Zone Selection For Subnets

A similar screen is displayed for the private subnets.

After setting the availability zones for the public and private subnets, a default instance type must be set for the regular cloud nodes. The major instance type is first set. For example, the `m3` type is a common major instance type that can be set (figure 4.10):

```
Please select the instance type family for cloud nodes

                        21 elements above ...
g4ad     g4ad instances (5)
g4dn     g4dn instances (7)
g5       g5 instances (8)
h1       h1 instances (4)
i2       i2 instances (4)
i3       i3 instances (7)
i3en     i3en instances (8)
im4gn    im4gn instances (6)
inf1     inf1 instances (4)
is4gen   is4gen instances (6)
m1       m1 instances (4)
m2       m2 instances (3)
m3       m3 instances (4)
m4       m4 instances (6)
m5       m5 instances (9)
m5a      m5a instances (8)
m5ad     m5ad instances (8)
m5d      m5d instances (9)
m5dn     m5dn instances (9)
m5n      m5n instances (8)
m5zn     m5zn instances (7)
m6a      m6a instances (10)
                        33 elements below ...

                 < Ok >      < Back >      < Help >
```

Figure 4.10: Configuration Processing With cm-cluster-extension: Major Instance Type

After selecting the major instance type, the specific default instance type can be set for the regular cloud nodes. The m3.medium type, (3.75GB RAM, 4GB SSD, 3 EC2 compute units) is a common default that can be used: (figure 4.11):

```
Select the default instance type for cloud nodes.

m3.medium   1 cores/3.75 GB
m3.large    2 cores/7.5 GB
m3.xlarge   4 cores/15.0 GB
m3.2xlarge  8 cores/30.0 GB

                 < Ok >      < Back >      < Help >
```

Figure 4.11: Configuration Processing With cm-cluster-extension: Default Instance Type

After setting the default instance type for the regular cloud nodes, screens (similar to figures 4.10 and 4.11) are displayed to guide the administrator into setting a default instance type screen for the cloud director node type.

After having set the default instance types for the regular and cloud director nodes. the VPN connection to the cluster extension is configured.

If the advanced setup (figure 4.4) was selected earlier in the wizard, then the administrator can select the type of VPN connection to the cluster extension (figure 4.12):

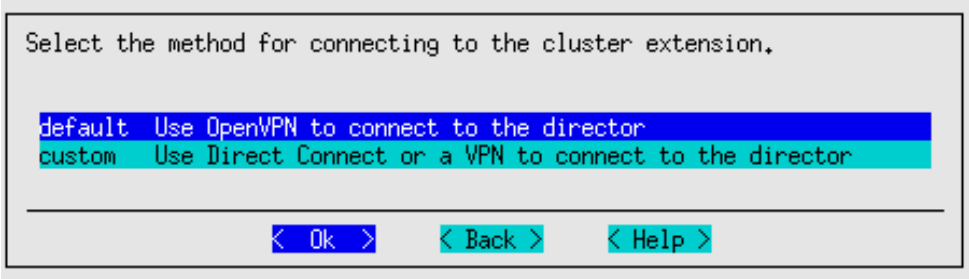

Figure 4.12: Configuration Processing With `cm-cluster-extension`: Custom VPN Configuration

- This can be the OpenVPN option that is automatically configured if the `advanced` setup is not chosen.
- With the `advanced` setup it is also possible to configure
  - Direct Connect (for AWS)
  - or another hardware-based VPN

As part of VPN configuration, the wizard guides the administrator through the process of configuring the cluster extension so that the extension:

- creates a new VPC for each cluster region. Alternatively, an existing VPC can be set.
- creates subnets for its private and public networks per region by default. Alternatively, existing subnets can be used instead, but the cloud director must be able to contact the on-premises head node.
- creates a security group for each region by default. Alternatively, existing security groups can be used instead.

After the VPN connection is configured, the summary screen (figure 4.13) is displayed. This lets the administrator look things over before deployment:

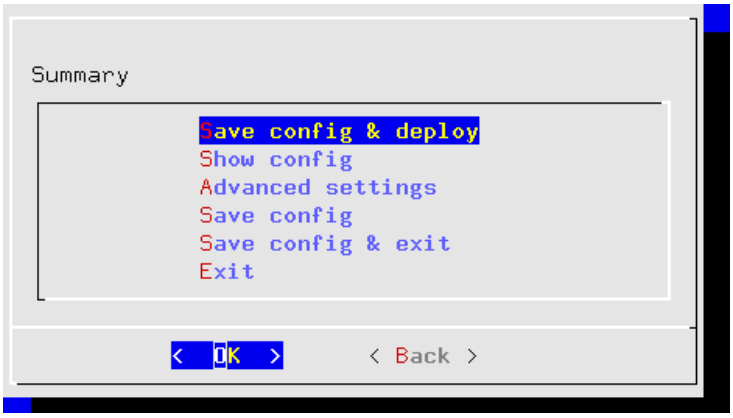

Figure 4.13: Configuration Processing With `cm-cluster-extension`: Summary Screen

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 TUI 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 TUI, 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 Base 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 Base View event viewer. The status can also be queried via the `status` command in device node. For example, a `watch` instruction such as:

```
[root@basecm11 ~]# 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.

BCM's cluster extension utilities create 3 cloud node objects by default (figure 4.7, page 69). Cloning them is an easy way to extend the number of deployable cloud nodes.

With Base 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:

```
[basecm11->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 67, a TUI 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

```
[basecm11->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
...
[basecm11->device*]% commit
Successfully committed 9 Devices
[basecm11->device]%
```

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

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

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

Example

```
[basecm11->device]% power on -n eu-west-1-cnode0[02-10]
```

4.3 Miscellaneous Cloud Tools

4.3.1 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	excludelistsyncinstall (active by default)
full	excludelistfullinstall
update	excludelistupdate (active by default)
grab	excludelistgrab
grab new	excludelistgrabnew

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

```
[basecm11->fspart]% use /cm/shared
[basecm11->fspart[/cm/shared]]% excludelistsnippets
[basecm11->fspart[/cm/shared]->excludelistsnippets]%
[basecm11->fspart[/cm/shared]->excludelistsnippets]% add test
[basecm11->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set modefull yes
[basecm11->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set excludelist
[basecm11->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% set excludelist /useless/test1
[basecm11->fspart*[/cm/shared*]->excludelistsnippets*[test*]]% append excludelist /useless/test2
[basecm11->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@basecm11 cmd]# grep useless /var/spool/cmd/*shared.rsync

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

Setting Filesystem Synchronization For Edge Director Filesystem Partitions

The edge director role allows the sync/cm/shared property to decide if the /cm/shared filesystem partition should be synchronized.

Example

```
[basecm11]% device use edge-director
[basecm11->device[edge-director]]% roles
[basecm11->device[edge-director]->roles]% use edgedirector
```

```
[basecm11->device[edge-director]->roles[edgedirector]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Name                             edgedirector
Revision
Type                             EdgeDirectorRole
Add services                     yes
Node power operations            yes
Director power operations        no
Add named service                yes
Add slapd service                yes
Add ntpd service                 yes
Open TCP ports on head node
Open UDP ports on head node
Externally visible IP            0.0.0.0
Externally visible head node IP  0.0.0.0
Sync /cm/shared                  yes
Disable automatic exports        no
Create home directories          never
```

4.4 Connecting To AWS With Direct Connect Or A Hardware VPN

For simple configurations, BCM 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 BCM 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 4.4.1). This step can be skipped if an existing VPC is to be used.
2. Connecting the local network to the VPC (section 4.4.2). 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 4.4.3).

4.4.1 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 navigation path 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 options under the navigation path:

   `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`

4.4.2 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`.

4.4.3 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 on the head node must be adjusted to accept traffic from the cloud subnet. Using the `openports` mode of the `firewall` role in `cmsh`, one rule is added per port that must be reachable from the peer subnet, restricting the source address to that subnet where possible:

```
[root@basecm11 ~]# cmsh
[basecm11]% device use basecm11; roles; use firewall; openports
[basecm11->device[basecm11]->roles[firewall]->openports]% add ACCEPT net <port> tcp fw <subnet CIDR>
[basecm11->device[basecm11]->roles[firewall]->openports]% commit
```

The `add` command is repeated for each required port and protocol. The `shorewall` service is restarted automatically after `commit`.

Cluster Extension Advanced Settings Configuration

A cluster extension can now be created using the `cm-cluster-extension` utility in TUI mode as explained in section 4.1.1, page 67.

However, on reaching the summary screen (figure 4.13, page 72), 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 4.14):

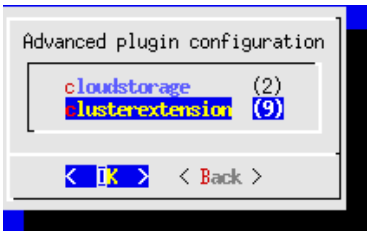

Figure 4.14: Cluster Extension Configuration Processing: Advanced Plugins Screen

The `clusterextension` option should be selected, which opens up an advanced options settings screen (figure 4.15):

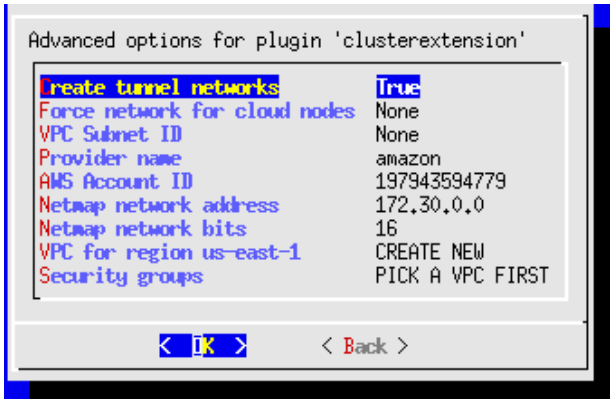

Figure 4.15: Cluster Extension Configuration Processing: Advanced Plugins For 'clusterextension' Options Screen

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 navigation path is used:

   `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 TUI 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 BCM 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@basecm11~]# cmsh
[basecm11]% network
[basecm11->network]% set externalnet allowautosign always
[basecm11->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 47.

  In summary, the prerequisites are as follows:

  - an activated cluster license should be ensured
  - an Azure account should exist
  - BCM must be registered as an Azure AD application (page 83)
  - 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 48.

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

5.2 Cluster Extension Into Azure

Section 4.1.1 introduces the `cm-cluster-extension` TUI 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 Base View browser-based wizard for cluster extension into Azure. The browser wizard is accessible via the navigation path: `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 65. Running it as a TUI session is however easier, and is described next.

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

The `cm-cluster-extension` utility can be run as a more user-friendly TUI session within the text environment. In a session described starting from page 67, 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 TUI main screen (figure 5.1):

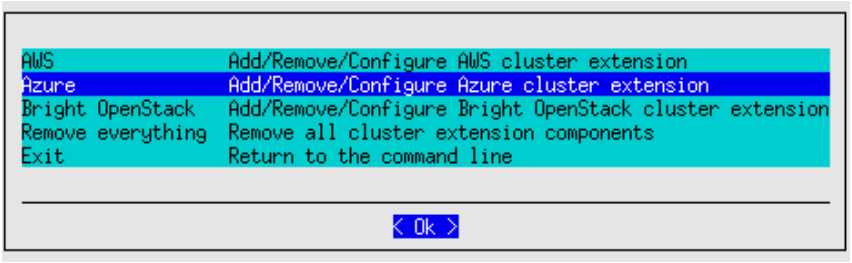

Figure 5.1: Configuration Processing With `cm-cluster-extension`: Azure Selection

After Azure is selected, a new Azure provider can be set if none is already set up. If an Azure region has been configured previously, then its configuration can be removed. Testing only the network connectivity to the various Azure regions is also possible (figure 5.2).

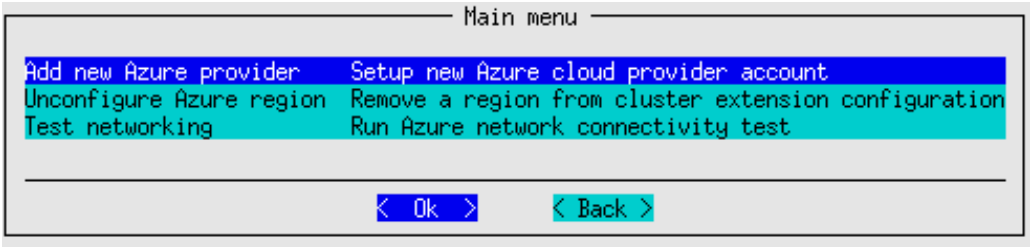

Figure 5.2: Configuration Processing With `cm-cluster-extension`: Add, Unconfigure, Or Test

If a new Azure provider is selected, then a screen comes up asking for the credentials for Azure (figure 5.3):

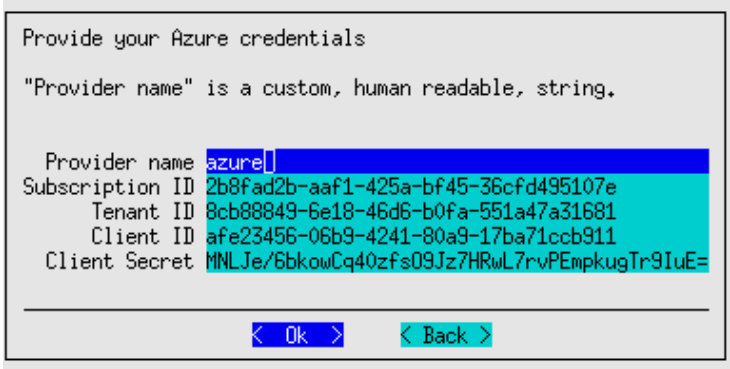

Figure 5.3: Configuration Processing With `cm-cluster-extension`: Azure Credentials

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

How BCM 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 the navigation paths 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 navigation path 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 navigation path 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 navigation path 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 navigation path: `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 navigation paths shown in the following table:

TUI	navigation path 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 navigation path, 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 5.3 can be set to any user-defined value. For Azure, a sensible, if unimaginative value, is simply `azure`.

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

After checking and accepting the credentials, legal terms may need to be accepted. The BCM Azure integration makes use of several node-installer images which are published in the Azure marketplace.

In order to use those images, Azure requires the administrator to accept the legal terms for those images. The script automatically detects if the terms were already accepted in the past, in which case it proceeds to the next step. If the terms still need to be accepted, then several links to the legal terms are presented, along with a prompt to accept them. The output looks similar to the following (some output ellipsized):

Example

```
#### stage: cx_azure: Ask Terms Acceptance
To use Azure VM images, you need to accept the following terms:
- License (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://download.microsoft.com/download/F/D/8/FD8BA8F...
- Privacy Policy (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://www.brightcomputing.com/privacy-policy
- Marketplace Terms (bcmni-azure-9-2:bcm-ni-azure-9-2-v1): https://mpcprodsa.blob.core.windows.net/market...
- License (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://download.microsoft.com/download/F/D/8/FD8BA...
- Privacy Policy (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://www.brightcomputing.com/privacy-policy
- Marketplace Terms (bcmni-azure-9-2-free:bcm-ni-azure-9-2-v2): https://mpcprodsa.blob.core.windows.net/mar...
Accept (y/n)?
```

After checking and accepting the legal terms, the administrator is asked to choose a setup type for the wizard. This can be a `default` setup, or it can be an `advanced` setup (figure 5.4):

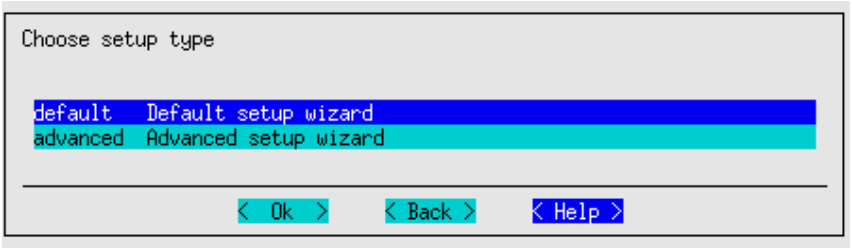

Figure 5.4: Configuration Processing With `cm-cluster-extension`: Default Setup Or Advanced Setup

The `default` setup is recommended for most use cases. The `default` setup configures an OpenVPN network connection to the cluster extension. The `advanced` setup allows the network connection to the cluster extension to use ExpressRoute (for Azure). or another hardware-based VPN.

After a `default` or `advanced` setup has been chosen, the initial number of cloud nodes to be set up in the cloud can be set (figure 5.5). A default of 3 is suggested.

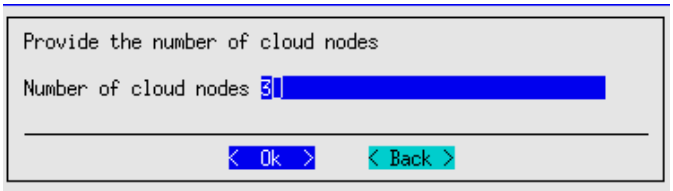

Figure 5.5: Configuration Processing With `cm-cluster-extension`: Setting The Initial Number Of Cloud Nodes

After setting an initial number of cloud nodes, the major regions into which these can be deployed are displayed (figure 5.6):

```
Please select the region(s) into which you would like to extend your cluster.
More regions can always be added and configured later on.

all          All regions (0/45)
africa       Africa (0/2)
asia         Asia (0/2)
australia    Australia (0/4)
brazil       Brazil (0/2)
canada       Canada (0/2)
europe       Europe (0/2)
france       France (0/2)
germany      Germany (0/2)
india        India (0/5)
japan        Japan (0/2)
korea        Korea (0/2)
norway       Norway (0/2)
sweden       Sweden (0/1)
switzerland  Switzerland (0/2)
uae          United Arab Emirates (0/2)
uk           United Kingdom (0/2)
us           United States (0/9)

                    < Ok >      < Back >      < Help >
```

Figure 5.6: Configuration Processing With `cm-cluster-extension`: Major Regions Selection

After selecting a major region, the available regions under it into which the cloud nodes can be deployed are displayed (figure 5.7):

```
Please select the region(s) into which you would like to extend your cluster.
More regions can always be added and configured later on.

[ ] northeurope  North Europe
[X] westeurope   West Europe

                    < Ok >      < Back >      < Help >
```

Figure 5.7: Configuration Processing With `cm-cluster-extension`: Regions Selection

After selecting one or more regions, the default region into which the cloud nodes can be deployed can be set (figure 5.8):

```
Extension will be deployed to the following regions. Please select the default region.

(X) westeurope

                    < Ok >      < Back >
```

Figure 5.8: Configuration Processing With `cm-cluster-extension`: Default Region Selection

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.

After setting the regions up, a default instance type must be set for the regular cloud nodes. The

instance type family is first set. For example, the Dv3 type is an instance family type that can be set (figure 5.9):

```
Please select the instance type family for cloud nodes

                        18 elements above ...
standardDFamily                 D-Series (8)
standardDSFamily                DS-Series (8)
standardDSv2Family              DSv2-Series (17)
standardDSv2PromoFamily         DSv2Promo-Series (8)
standardDSv3Family              DSv3-Series (7)
standardDSv4Family              DSv4-Series (7)
standardDSv5Family              DSv5-Series (8)
standardDv2Family               Dv2-Series (10)
standardDv2PromoFamily          Dv2Promo-Series (8)
standardDv3Family               Dv3-Series (7)
standardDv4Family               Dv4-Series (7)
standardDv5Family               Dv5-Series (8)
                        55 elements below ...

                  < Ok >      < Back >      < Help >
```

Figure 5.9: Configuration Processing With `cm-cluster-extension`: Instance Type Family

Instance types with Azure: After selecting the instance type family, the specific default instance type can be set for the regular cloud nodes. The `Standard_D2_v3` type, (2 cores, 8GB RAM) is a common default that can be used (figure 5.10):

```
Select the default instance type for cloud nodes.

Standard_D2_v3   2 cores/8.0 GB
Standard_D4_v3   4 cores/16.0 GB
Standard_D8_v3   8 cores/32.0 GB
Standard_D16_v3  16 cores/64.0 GB
Standard_D32_v3  32 cores/128.0 GB
Standard_D48_v3  48 cores/192.0 GB
Standard_D64_v3  64 cores/256.0 GB

                  < Ok >      < Back >      < Help >
```

Figure 5.10: Configuration Processing With `cm-cluster-extension`: Default Instance Type

Azure intance types are documented online at

- `https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes` and
- `https://docs.microsoft.com/en-us/azure/virtual-machines/vm-naming-conventions`.

Eventually, when the wizard has finished deployment of the Azure instance, 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 Base View navigation path:

`Cloud > Azure > Azure VM Sizes`

After setting the default instance type for the regular cloud nodes, screens (similar to figures 5.9 and 5.10) are displayed to guide the administrator into setting a default instance type screen for the cloud director node type. The same default instance type suggested for regular cloud nodes earlier, `Standard_D2_v3`, is typically adequate for cloud director nodes too, for small clusters.

Accelerated networking with Azure: NICs use hardware queues (ring buffers) to send and receive packets. Some NICs allow individual queues to be directly mapped from the physical host to a VM. The standard for doing this is called single root I/O virtualization (SR-IOV). This frees the host kernel from handling those packets, since, for example, network interrupts are delivered directly to the VM kernel. SR-IOV typically results in significantly lower latencies and ping times for the VM, similar to that of a bare metal machine.

On Azure, *Accelerated Networking* enables SR-IOV, if the VM type supports it. The VM types that support Accelerated Networking can be found using the

```
az vm list-skus
```

command from the Azure CLI, as described at `https://learn.microsoft.com/en-us/azure/virtual-network/accelerated-networking-overview`.

VPN configuration with Azure: After having set the default instance types for the regular and cloud director nodes. the VPN connection to the cluster extension is configured.

If the `advanced` setup (figure 5.4) was selected earlier in the wizard, then the administrator can select the type of VPN connection to the cluster extension (figure 5.11):

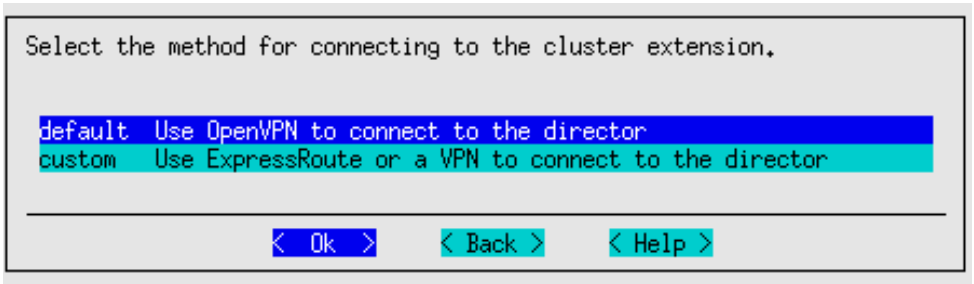

Figure 5.11: Configuration Processing With `cm-cluster-extension`: Custom VPN Configuration

- This can be the OpenVPN option that is automatically configured if the `advanced` setup is not chosen.
- With the `advanced` setup it is also possible to configure
  - ExpressRoute (for Azure)
  - or another hardware-based VPN

As part of VPN configuration, the wizard guides the administrator through the process of configuring the cluster extension so that the extension:

- creates a resource group for each region by default. Alternatively, existing resource groups can be used instead.
- creates a subnet per region by default. Alternatively, existing subnets can be used instead, but the cloud director must be able to contact the on-premises head node.
- creates a storage account for each region by default. Alternatively, existing storage accounts can be used per region instead.

- creates a security group for each region by default. Alternatively, existing security groups can be used instead.

After the VPN connection is configured, the summary screen (figure 4.13) is displayed. This lets the administrator look things over before deployment:

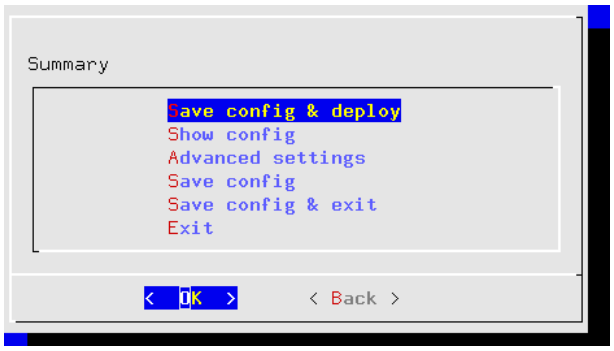

Figure 5.12: Configuration Processing With `cm-cluster-extension`: Azure Options Summary Screen

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.
- 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*>, can be saved and the TUI 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 TUI, 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

```
[basecm11->device]% power on westeurope-director
cloud ..................... [ PENDING ] westeurope-director
... [notice] basecm11: New certificate request with ID: 9
... [notice] basecm11: westeurope-director [ INSTALLER REBOOTING ]
... [notice] basecm11: westeurope-director [     INSTALLING      ] (node installer started)
... [notice] basecm11: New certificate request with ID: 10 (ldaps)
... [notice] basecm11: westeurope-director [ INSTALLER_CALLINGINIT ] (switching to local root)
... [notice] basecm11: westeurope-director [   UP   ]
```

Example

```
[root@basecm11 ~]# 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@basecm11 ~]# 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@basecm11 ~]# 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 BCM 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@basecm11 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        4
Source node:          basecm11
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@basecm11 ~]# cmsh -c "softwareimage provisioningstatus -a"
Request ID(s):        5
Source node:          basecm11
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.

NOTE: Cloud director and node startup from snapshot is also possible. Section 3.4 describes it for AWS. For Azure it is more involved, and is not covered here.

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 245 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 navigation path to do this cloning in Base 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 Base View, using the navigation path:
  `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:

   - within the `device` mode of `cmsh`
   - with Base View, using the navigation path:
     `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 Creating An Azure Cluster Extension Using ExpressRoute Or A Hardware VPN

For simple configurations, BCM 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 BCM 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 5.5.1). This step can be skipped if an existing virtual network is to be used.

2. Connecting the local network to the virtual network (section 5.5.2). 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 5.5.3).

5.5.1 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 navigation path can be followed:

   `+ Create a resource` > `Azure Marketplace` > `Networking` > `Virtual network`

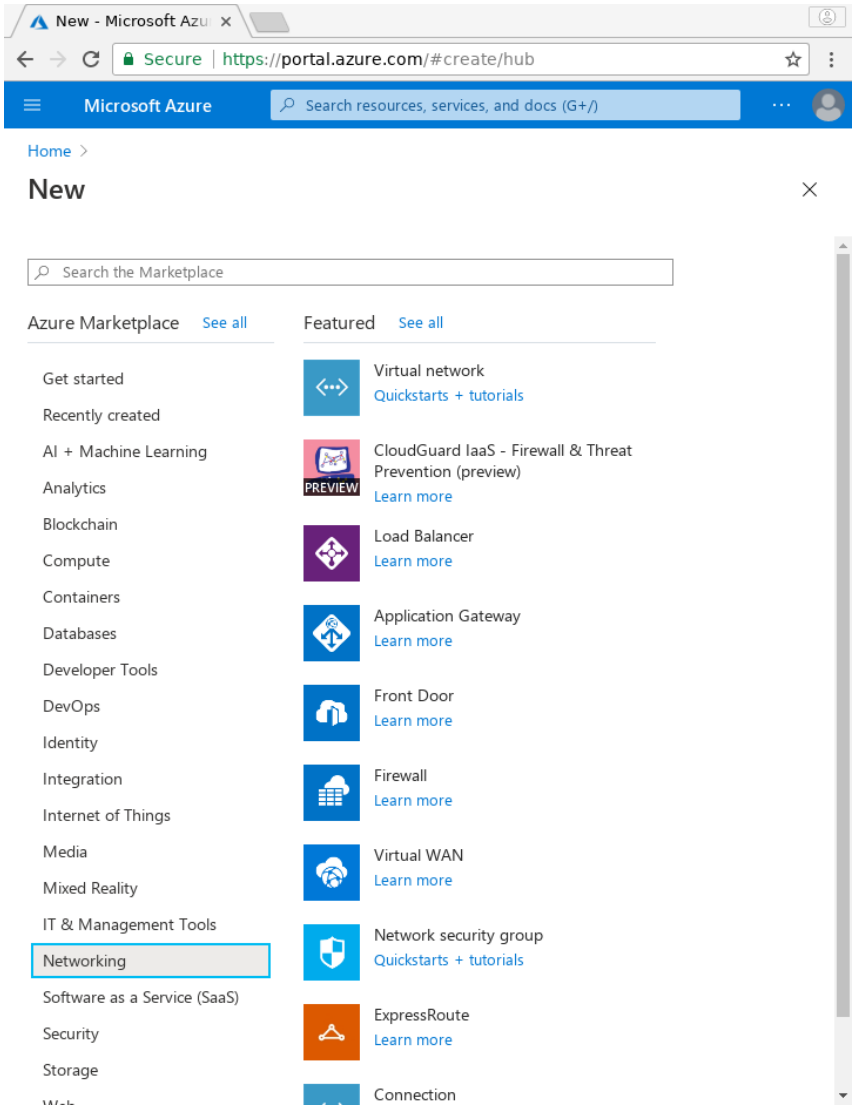

Figure 5.13: Azure Marketplace Networking Screen

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

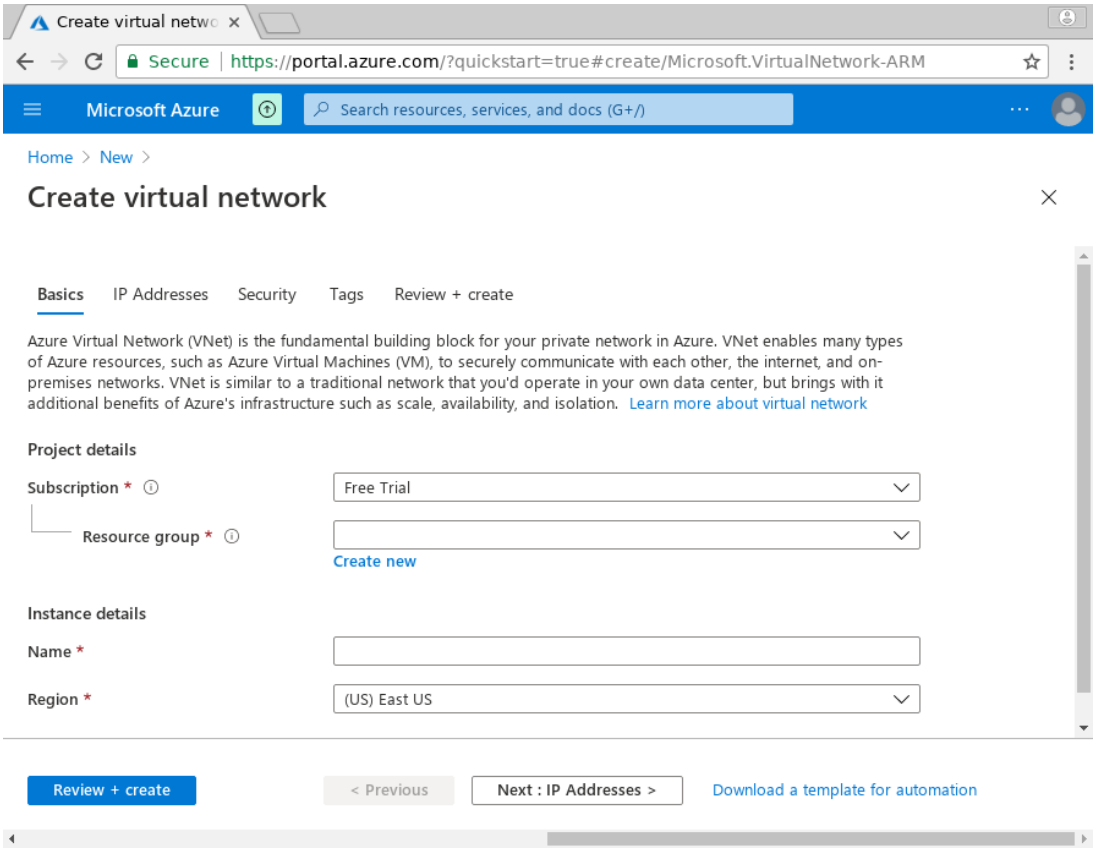

Figure 5.14: Creating A Resource Group In The `Basics` Tab

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:

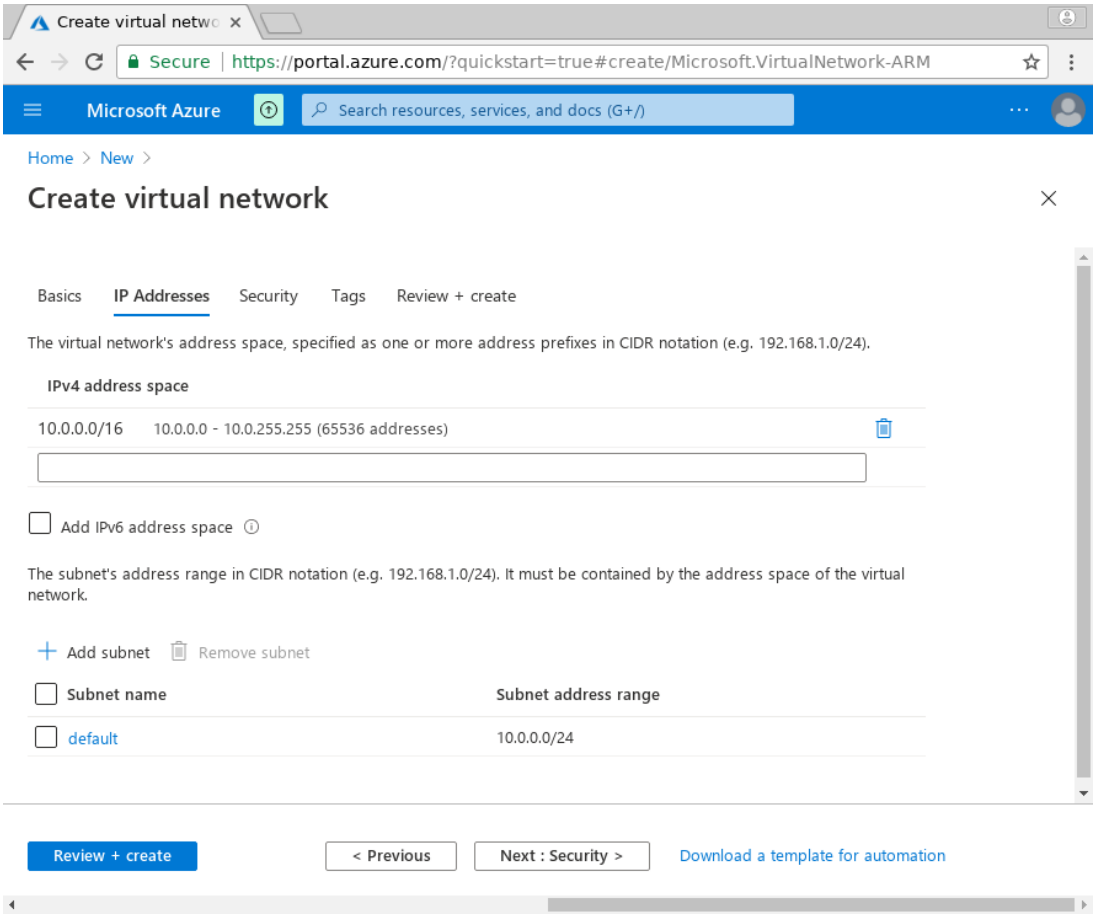

Figure 5.15: Creating IP Address Ranges In The `IP Addresses` Tab

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.

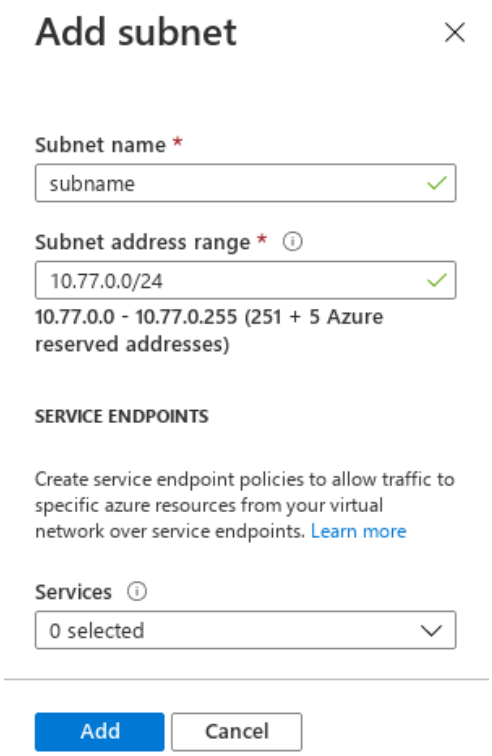

Figure 5.16: Creating IP Address Ranges In The `IP Addresses` Tab: Subnet Addition

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

5. The navigation path:

   `Review + create` > `Create`

   goes on to deploy the virtual network.

5.5.2 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`

Connecting 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`.

5.5.3 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 on the head node must be adjusted to accept traffic from the virtual network. Using the `openports` mode of the `firewall` role in `cmsh`, one rule must be added per port that must be reachable from the peer subnet, restricting the source address to that subnet where possible:

```
[root@basecm11 ~]# cmsh
[basecm11]% device use basecm11; roles; use firewall; openports
[basecm11->device[basecm11]->roles[firewall]->openports]% add ACCEPT net <port> tcp fw <subnet CIDR>
[basecm11->device[basecm11]->roles[firewall]->openports]% commit
```

The `add` command must be repeated for each required port and protocol. The `shorewall` service is restarted automatically after `commit`.

Cluster Extension Advanced Settings Configuration

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

However, on reaching the summary screen (figure 5.12, page 89), 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 5.17):

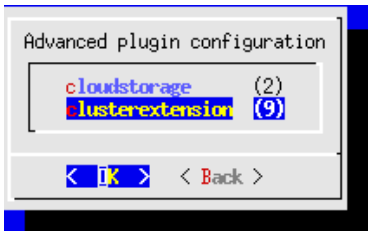

Figure 5.17: Cluster Extension Configuration Processing: Advanced Plugins Screen

The `clusterextension` option should be selected, which opens up an advanced options settings screen for Azure (figure 5.18):

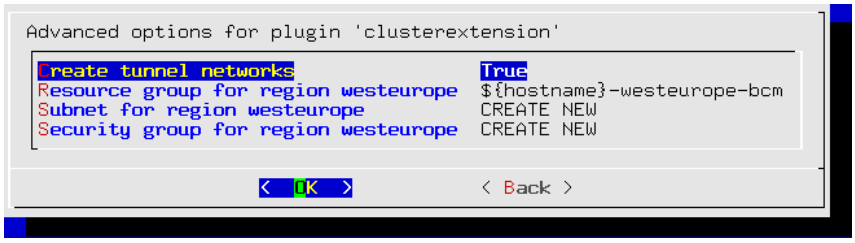

Figure 5.18: Cluster Extension Configuration Processing: Advanced Plugins For 'clusterextension', Azure Options Screen

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 TUI 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 BCM 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@basecm11~]# cmsh
[basecm11]% network
[basecm11->network]% set externalnet allowautosign always
[basecm11->network]% commit
```

The cluster extension can now be deployed as explained on page 89.

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

5.6.1 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`.

5.6.2 Implementation Of Hyper-V Generation Choice In BCM

Additional fields were introduced in NVIDIA Base Command Manager 9.2 to allow the generation to be specified, for the Azure VMSize, CloudProvider, and CloudSettings entities.

Also introduced in 9.2 were new "hybrid" node-installer images, which support Gen1 (BIOS-based) and Gen2 (UEFI-based) VMs. The images are partitioned with GPT and contain an ESP partition for UEFI boot. The old, BIOS-only images cannot be used for Gen2 VMs.

In the Azure Marketplace BCM provides separate SKUs for Gen1 and Gen2 images, although internally they are the same.

If a cloud node instantiated from a non-Marketplace node-installer image VHD, then CMDaemon first creates a generation-specific Azure Image resource out of the VHD in the cluster extension resource group. That image resource is then used to create the VM.

Some Azure VM sizes support both Gen1 and Gen2, while others are generation-specific.

To see which VM size supports which generation of Hyper-V, cloud types can be listed in `cmsh` with the `hypervgenerations` parameter:

Example

```
[basecm11->cloud[azure]]% types
[basecm11->cloud[azure]->types]% list -f name,hypervgenerations
name (key)           hypervgenerations
-------------------- --------------------
Basic_A0             V1
Basic_A1             V1
Basic_A2             V1
Basic_A3             V1
Basic_A4             V1
Standard_A0          V1
Standard_A1          V1
...
Standard_B12ms       V1,V2
Standard_B16ms       V1,V2
Standard_B1ls        V1,V2
Standard_B1ms        V1,V2
```

Formats for the list command are discussed on page 61 of the *Administrator Manual*.

Gen1 or Gen2 availability for a cloud is indicated by the tag `V1` or `V2` respectively.

For historical reasons, and because of the many VM sizes available, BCM currently (September 2022) creates Gen1 VMs by default. This is not expected to remain the case.

Switching to Gen2 from Gen1 is a matter of using a VM size that supports `V2`, and then changing the Hyper-V generation parameter.

For the cloud provider the change can be carried out as follows, and all cloud nodes then become Gen2:

Example

```
[basecm11->cloud[azure]]% set defaulthyper-vgeneration v2
[basecm11->cloud*[azure*]]% commit
[basecm11->cloud[azure]]%
```

Alternatively, each VM can have its Hyper-V generation setting changed within the `cloudsettings` submode:

Example

```
[basecm11->device[westeurope-director]->cloudsettings]% get hyper-vgeneration
V1 (azure)
[basecm11->device[westeurope-director]->cloudsettings]% set hyper-vgeneration v2; commit
[basecm11->device[westeurope-director]->cloudsettings]%
```

5.7 Running NVIDIA A100 GPUs On Cloud Nodes

GPU hardware detection by the Linux kernel may be very delayed, or even be unsuccessful, on cloud nodes that run NVIDIA A100 GPUs. A100 hardware is made available on some instance types such as, for example, `Standard_NC96ads_A100_v4`. To work around this issue, the `pci-hyperv` and `pci-hyperv-intf` kernel modules must be loaded to the kernel used by cloud node images.

In a default Azure setup, this can typically be carried out by adding the kernel modules to the existing kernel modules used by the cloud node category:

Example

```
[basecm11->category]% use centralus-cloud-node
[basecm11->category[centralus-cloud-node]]% kernelmodules
[basecm11->category[centralus-cloud-node]->kernelmodules]% add pci-hyperv
[basecm11->category*[centralus-cloud-node*]->kernelmodules*[pci-hyperv*]]% add pci-hyperv-intf
[basecm11->category*[centralus-cloud-node*]->kernelmodules*[pci-hyperv-intf*]]% commit
[basecm11->category[centralus-cloud-node]->kernelmodules[pci-hyperv-intf]]%
[basecm11->category[centralus-cloud-node]->kernelmodules[pci-hyperv-intf]]% exit
[basecm11->category[centralus-cloud-node]->kernelmodules]% list
Module (key)             Parameters
------------------------ ----------------------------------------------
pci-hyperv
pci-hyperv-intf
[basecm11->category[centralus-cloud-node]->kernelmodules]%
... [notice] basecm11: Initial ramdisk for category centralus-cloud-node based on image default-image
is being generated
... [notice] basecm11: Initial ramdisk for category centralus-cloud-node based on image default-image
was generated successfully
```

Booting the cloud nodes with these kernel modules configured allows the GPUs to be detected correctly.

6

Cloud Considerations And Choices With NVIDIA Base Command Manager

6.1 Differences Between Cluster On Demand And Cluster Extension

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

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

A note about the last entry: The access to the cloud director addresses can be restricted to an administrator-defined set of IP addresses, using the "`Externally visible IP`" entry in figure 3.1, page 94, of the *Administrator Manual*, after scrolling down in the screen to reach that entry.

6.2 Hardware And Software Availability

BCM 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 BCM 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 Base View or `cmsh` can be executed. These commands stop the instance, without terminating it. Any associated extra drives that were created need to be removed manually, via the `Volumes` screen in the `Elastic Block Store` resource item in the navigation menu of the AWS Management Console.

6.3 Reducing Running Costs

6.3.1 Spot Pricing

Spot prices are volatile prices that apply to the nodes within an AWS cluster extension. Tracking their values helps the user to take advantage of cheaper pricing made available at irregular times. The user can 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/on-demand/`.

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

Getting An On-Demand Price

The `prices` command is used to query AWS for On-Demand Prices.

One, or both, of the following items must be specified

- A region, with the `-r|--region` option.
- A type of cloud node instance, with the `-t|--type` option.

For example, if the cloud provider instance for AWS is `myaws`, then a `prices` query can be run as follows, from within `cloud` mode (some output elided):

Example

```
[basecm11->cloud]% prices myaws  -t m5.large
Region                      Instance Type Price         Currency
--------------------------- ------------- ------------- --------
AWS GovCloud (US-East)      m5.large      0.1210000000  USD
AWS GovCloud (US-West)      m5.large      0.1210000000  USD
Africa (Cape Town)          m5.large      0.1270000000  USD
...
US West (Los Angeles)       m5.large      0.1150000000  USD
US West (N. California)     m5.large      0.1120000000  USD
US West (Oregon)            m5.large      0.0960000000  USD
```

The preceding query is for all regions, and asks for a price for all cloud node instances of type `m5.large`. Filtering further for a particular region can be done with the `-r` option:

```
[basecm11->cloud]% prices myaws -r eu-west-1 -t m5.large
Region                Instance Type Price         Currency
--------------------- ------------- ------------- --------
EU (Ireland)          m5.large      0.1070000000  USD
```

Checking the price for a particular region for all types of cloud node instances can be carried out as follows (hundreds of lines of output omitted):

```
[basecm11->cloud]% prices myaws -r eu-west-1
Region               Instance Type      Price           Currency
-------------------- ------------------ --------------- --------
EU (Ireland)         a1.2xlarge         0.2304000000    USD
EU (Ireland)         a1.4xlarge         0.4608000000    USD
EU (Ireland)         a1.large           0.0576000000    USD
...
```

Getting A Spot Price

The `spotprices` command is used to query AWS for Spot Prices.

A region or availability zone must be specified with the `-r|--region` or the `-z|--az` options. Additionally, an extra filter can be used to specify the cloud node instance type with the `-t|--type` option.

For example, if the cloud provider instance for AWS is `myaws`, then a `spotprices` query can be run as follows, from within `cloud` mode (some hundreds of lines of output elided):

Example

```
[basecm11->cloud]% spotprices myaws -z eu-west-1a
Availability Zone    Instance Type   Price per Hour Timestamp
-------------------- --------------- -------------- --------------------
eu-west-1a           c1.medium       0.014800       2022-02-16 18:02:30
eu-west-1a           c1.xlarge       0.070300       2022-02-17 17:44:25
eu-west-1a           c3.2xlarge      0.256700       2022-02-17 10:46:21
...
```

An example of additionally filtering for the `m5.large` node instance type is:

Example

```
[basecm11->cloud[myaws]]% spotprices myaws -r eu-west-1 -t m5.large
Availability Zone    Instance Type Price per Hour Timestamp
-------------------- ------------- -------------- --------------------
eu-west-1a           m5.large      0.036800       2022-02-17 18:17:06
eu-west-1b           m5.large      0.036200       2022-02-17 09:55:08
eu-west-1c           m5.large      0.036100       2022-02-17 18:38:26
```

6.3.2 Storage Space Reduction

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

Example

```
[basecm11]% device cloudsettings eu-west-1-director
[basecm11->device[eu-west-1-director]->cloudsettings]% storage
[basecm11->...->cloudsettings->storage]% set ebs size 15GB; commit
[basecm11->...->cloudsettings->storage]% device cloudsettings cnode001
[basecm11->device[cnode001]->cloudsettings]% storage
[basecm11->...->cloudsettings->storage]% set ebs size 5GB; commit
```

The value for the cloud node EBS storage can also be set via Base View, using the navigation path:

`Devices` > `Cloud Nodes` > `Edit` > `Settings` > `Cloud settings` > `STORAGE` > `Storage` > `ebs` > `Edit` > `size`

6.4 Setting The Cloud Node Images

The default cloud node image can be set within the VPC instance of `cmsh`:

Example

```
[basecm11->device[eu-west-1-cnode001]->cloudsettings]% cloud use myaws
[basecm11->cloud[myaws]]% vpcs
[basecm11->cloud[myaws]->vpcs]% use vpc-eu-west-1
[basecm11->cloud[myaws]->vpcs[vpc-eu-west-1]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Name                             vpc-eu-west-1
Revision
Default AMI                      latest
VPC ID                           vpc-0e4565afe04528047
...
region                           eu-west-1
...
```

The value of `defaultami` is set by specifying the AMI ID.

Example

```
[basecm11->cloud[myaws]->vpcs[vpc-eu-west-1]]% set defaultami ami-0ce80021a5acafcd0; commit
```

A list of possible images is queried with the `images` command:

Example

```
[basecm11->cloud[myaws]->vpcs[vpc-eu-west-1]]% images
Name                 Id                   API Hash                         Region
-------------------- -------------------- -------------------------------- ----------
brightinstaller-002  ami-0defc32f9b2756e40                                  eu-west-1
brightinstaller-100  ami-0ce80021a5acafcd0  5cd0910d8814d98fc331f7af72975c8a eu-west-1
...
```

Setting the default AMI to the special value `latest` means that the latest AMI is used.

Setting a default AMI means that cluster extension cloud nodes by default take that default AMI as their image:

Example

```
[basecm11->device[eu-west-1-cnode001]->cloudsettings]% show
Parameter                                      Value
---------------------------------------------- ----------------------------------------------
Availability zone                              eu-west-1a
Provider                                       myaws
...
Use kernel and initrd from the software image  yes
Region                                         eu-west-1
AMI                                            latest (vpc-eu-west-1)
...
```

The inherited default value can be overridden within the `device` modei, within the `cloudsettings` submode:

Example

```
[basecm11->device[eu-west-1-cnode001]->cloudsettings]% set ami ami-0defc32f9b2756e40; commit
```

6.5 Cloud Provider Default Settings

6.5.1 Default Settings For AWS Cloud Nodes

The cloud provider level in cmsh for AWS has some defaults that can be managed:

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% cloud
[basecm11->cloud[amazon]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Directors
Name                             amazon
Nodes
Revision
Type                             ec2
API Region Name
Access key ID                    < not set >
Secret access key                < not set >
IAM role name
Default region
Default type
Default director type
Image Owners                     137677339600,197943594779
Tags
VPCs                             <0 in submode>
Cloud job tagging                no
```

6.5.2 Default Settings For Azure Cloud Nodes

The cloud provider level in cmsh for Azure also has some defaults that can be managed:

Example

```
[root@basecm11 ~]# cmsh
[basecm11]% cloud
[basecm11->cloud[azure]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Directors
Name                             azure
Nodes
Revision
Type                             azure
Subscription ID
Client ID
Client secret                    < not set >
Tenant ID
Default node-installer image
Cloud Name                       AzureCloud
Default Location
Default Cloud Director VM Size
Default VM Size
Default Hyper-V generation       V1
Free image type                  Marketplace
Tags
Extensions                       <0 in submode>
```

6.5.3 Default Settings For OCI Cloud Nodes

The cloud provider level in cmsh for OCI also has some defaults that can be managed:

Example

```
[basecm11->cloud[oci]]% show
Parameter                        Value
-------------------------------- ------------------------------------------------
Directors
Name                             oci
Nodes
Revision
Type                             oci
Default node-installer image ID
Default compartment ID
Default Region
Default VM Shape
API Region Name
SecGroupN
Tags
Auth User
Auth Key Content                 <OB>
Auth Fingerprint
Auth Tenancy
Images compartment ID
Images manifest base URL
```

6.6 Address Resolution In Cluster Extension Networks

6.6.1 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 BCM.

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.

6.6.2 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 6.1) is a helper mapping reserved for use in routing from the cloud (that is, from a cloud director or a cloud node) to a regular node. The mapping uses the 172.30.0.0/16 addressing scheme. Its routing is asymmetrical, that is, a NetMap mapping from a regular node to the cloud does not exist. Packets from a regular node to the cloud do however resolve to the `cloud` network as indicated by 2(a) in the preceding.

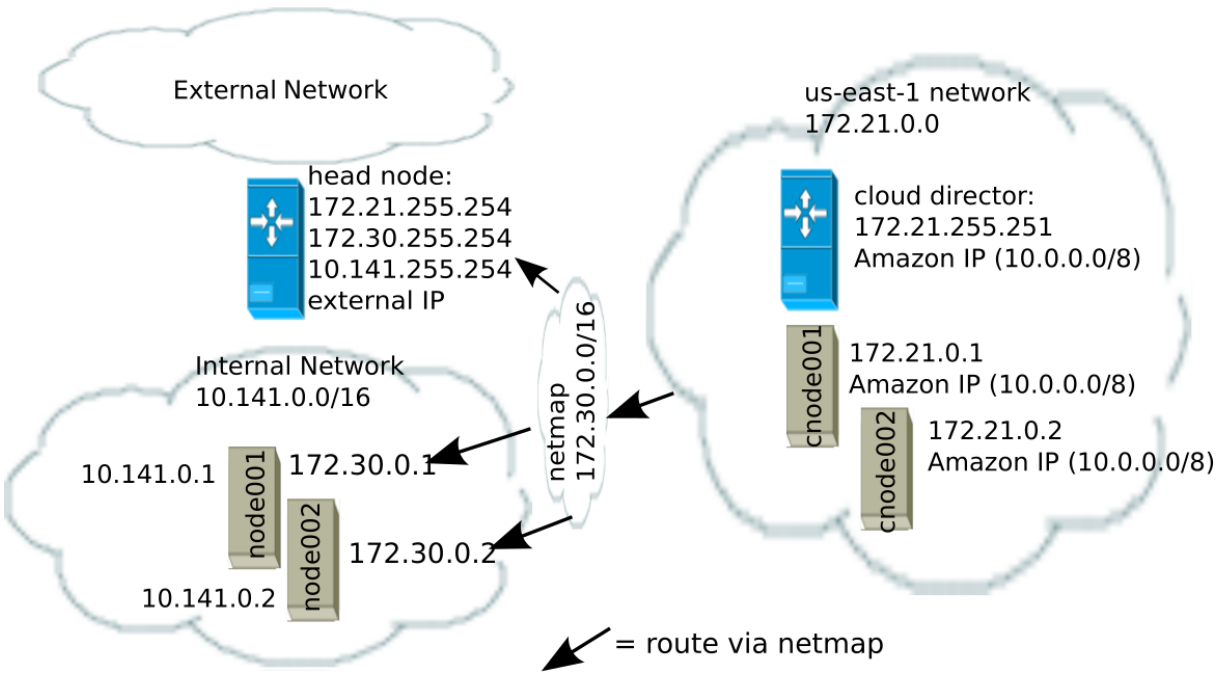

Figure 6.1: NetMap In Relation To The General Network Scheme

As pointed out in the introduction to this section (6.6), 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 BCM 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 BCM become useless.

6.7 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 6.7:

Table 6.7: Cloud compute node access to internet

Cloud node type	Internet access by default?	Details

CX-AWS	Yes	Cloud compute nodes are routed via an sNAT gateway cloud director node. They therefore do not normally require assignment of Elastic IPs (section 6.9.1). 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).
		To allocate a public IP to a node, the cloud setting `allocatepublicip` is set to `yes` before creating the node. By default, cloud directors are configured to be allocated public IP addresses:
		Example
		`[basecm11->device[us-east-1-director]->cloudsettings]% get allocatepublicip`
		`yes`

CX-Azure	Yes	Cloud compute nodes use the built-in NAT capabilities of Azure's gateway when accessing the internet. ICMP from cloud compute nodes to the internet does not work. This is because the default Azure gateway does not support it. The Load Balancer is a TCP or UDP product for load balancing and port forwarding for these specific IP protocols. Load balancing rules and inbound NAT rules are supported for TCP and UDP and not supported for other IP protocols including ICMP. If ICMP connectivity to the internet is needed then a public IP must be assigned to the cloud compute node.
COD-AWS	Yes	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. For HA with COD-AWS, the administrator may wish to create a private cluster with dedicated network connectivity. This can be achieved by disabling creation of a NAT gateway and disabling use of a public shared IP address during setup via TUI (page 41).
COD-Azure	Yes	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.

6.8 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 BCM support should be contacted.

6.9 Setting Up And Creating A Custom VPC

From NVIDIA Base Command Manager version 7.3 onward, the Amazon EC2-classic platform is no longer available, and all nodes run via BCM within Amazon always run within an EC2-VPC platform.

Custom VPC for NVIDIA Base Command Manager 11 subnet allocation, and allocation of EIPs (External IPs, public IP addresses) is described in sections 6.9.1–6.9.3.

6.9.1 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 BCM, 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.

6.9.2 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 BCM managing the routing tables on that router. The routing ensures correct subnet communication. Inside BCM, 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 Base View set to `CLOUD`.

Subnets for a custom VPC must have non-overlapping ranges. If there are multiple custom VPCs being managed by BCM, 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 107 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.

6.9.3 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 BCM:

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 Base 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

  ```
  [basecm11->network]% add vpc-0-sn-0
  [basecm11->network*[vpc-0-sn-0*]]% set type cloud
  [basecm11->network*[vpc-0-sn-0*]]% set baseaddress 10.0.0.0
  [basecm11->network*[vpc-0-sn-0*]]% set netmaskbits 24
  [basecm11->network*[vpc-0-sn-0*]]% commit
  ```

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

  Example

  ```
  [basecm11->network]% clone vpc-0-sn-0 vpc-0-sn-1
  [basecm11->network*[vpc-0-sn-1*]]% set baseaddress 10.0.1.0
  [basecm11->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

  ```
  [basecm11->cloud[Amazon EC2]->vpcs]%
  [basecm11->...->vpcs]% add vpc-0
  [basecm11->...->vpcs*[vpc-0*]]% set region eu-west-1
  [basecm11->...*[vpc-0*]]% set baseaddress 10.10.0.0
  [basecm11->...*[vpc-0*]]% set netmaskbits 16
  [basecm11->...*[vpc-0*]]% set subnets vpc-0-sn-0 vpc-0-sn-1
  [basecm11->...*[vpc-0*]]% commit
  ```