Advanced Cluster Options

Date of Publish: 2019-05-28

https://docs.hortonworks.com/

Contents

Custom images

In addition to providing default images, Cloudbreak allows you to create custom base images. Refer to this section if you would like to create custom images for Cloudbreak-managed clusters.

Default images are available for each supported cloud provider and region. The following table lists the default base images available:

Cloud provider	Default image
AWS	Amazon Linux
AWS	Amazon Linux 2
Azure	CentOS 7
GCP	CentOS 7
OpenStack	CentOS 7

Since these default images may not fit the requirements of some users (for example when user requirements include custom OS hardening, custom libraries, custom tooling, and so on), you can use your own custom base images.

In order to use your own custom base images you must:

1. Build your custom image(s).
2. Prepare the custom image catalog JSON file and save it in a location accessible to the Cloudbreak instance.
3. Register your custom image catalog in Cloudbreak.
4. If using a different operating system than that included on default images, set the CB_ENABLED_LINUX_TYPES variable in Profile.
5. Select a custom image when creating a cluster.

Note:

Only base images can be created and registered as custom images. Do not create or register prewarmed images as custom images.

Build custom images

Refer to the https://github.com/hortonworks/cloudbreak-images repository for information on how to build custom images.

This repository includes instructions and scripts to help you build custom images. Once you have the images, refer to the documentation below for information on how to create an image catalog and register it with Cloudbreak.

Related Information

https://github.com/hortonworks/cloudbreak-images

Prepare image catalog

Once you've built the custom images, prepare your custom image catalog JSON file.

Steps

1. Prepare your custom image catalog JSON file. Use the information included in this section to create a valid image catalog.

2. Once your image catalog JSON file is ready, save it in a location accessible via HTTP/HTTPS.

Structure of the image catalog

Use this information to create a valid image catalog.

The image catalog JSON file includes the following two high-level sections:

- images: Contains information about the created images. The burned images are stored in the base-images section.
- versions: Contains the cloudbreak entry, which includes mapping between Cloudbreak versions and the image identifiers of burned images available for these Cloudbreak versions.

Note:

After adding your image(s) to the images section, make sure to also update the versions section.

The images section

The burned images are stored in the base-images sub-section of images. The base-images section stores one or more image “records”. Every image “record” must contain the date, description, images, os, os_type, and uuid fields.

Parameter	Description
date	Date for your image catalog entry.
description	Description for your image catalog entry.
images	The image sets by cloud provider. An image set must store the virtual machine image IDs by the related region of the provider (AWS, Azure) or contain one default image for all regions (GCP, OpenStack). The virtual machine image IDs come from the result of the image burning process and must be an existing identifier of a virtual machine image on the related provider side. For the providers which use global rather than per-region images, the region should be replaced with default .
os	The operating system used in the image.
os_type	The type of operating system which will be used to determine the default Ambari and HDP/HDF repositories to use. Set os_type to “redhat6” for amazonlinux or centos6 images. Set os_type to “redhat7” for centos7 or rhel7 images.
uuid	The uuid field must be a unique identifier within the file. You can generate it or select it manually. The utility uuidgen available from your command line is a convenient way to generate a unique ID.
package-versions	The package versions used for Salt (salt) and Salt Bootstrap (salt-bootstrap).

The versions section

The versions section includes a single “cloudbreak” entry, which maps the uuids to a specific Cloudbreak version:

Parameter	Description
images	Image uuid, same as the one that you specified in the base-images section.
versions	The Cloudbreak version(s) for which you would like to use the images.

Example image catalog

Use this example to create a valid image catalog.

Here is an example image catalog JSON file that includes two sets of custom base images:

- A custom base image for AWS:
  - That is using Amazon Linux operating system
  - That will use the Redhat 6 repos as default Ambari and HDP repositories during cluster create
  - Has a unique ID of “44b140a4-bd0b-457d-b174-e988bee3ca47”

  - Is available for Cloudbreak 2.8.0
- A custom base image for Azure, Google, and OpenStack:
  - That is using CentOS 7 operating system
  - That will use the Redhat 7 repos as default Ambari and HDP repositories during cluster create
  - Has a unique ID of “f6e778fc-7f17-4535-9021-515351df3692”
  - Is available to Cloudbreak 2.8.0

You can also download it from here.


```
{
  "images": {
    "base-images": [
      {
        "date": "2017-10-13",
        "description": "Cloudbreak official base image",
        "images": {
          "aws": {
            "ap-northeast-1": "ami-78e9311e",
            "ap-northeast-2": "ami-84b613ea",
            "ap-southeast-1": "ami-75226716",
            "ap-southeast-2": "ami-92ce23f0",
            "eu-central-1": "ami-d95be5b6",
            "eu-west-1": "ami-46429e3f",
            "sa-east-1": "ami-86d5abea",
            "us-east-1": "ami-51a2742b",
            "us-west-1": "ami-21ccfe41",
            "us-west-2": "ami-2a1cdc52"
          }
        },
        "os": "amazonlinux",
        "os_type": "redhat6",
        "uuid": "44b140a4-bd0b-457d-b174-e988bee3ca47",
        "package-versions": {
          "salt": "2017.7.5",
          "salt-bootstrap": "0.13.0-2018-05-03T07:39:07"
        }
      },
      {
        "date": "2017-10-13",
        "description": "Cloudbreak official base image",
        "images": {
          "azure": {
            "Australia East": "https://
hwxaustraliaeast.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "Australia South East": "https://
hwxaustralisoutheast.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "Brazil South": "https://
sequenceiqbrazilsouth2.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "Canada Central": "https://
sequenceiqcanadacentral.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "Canada East": "https://
sequenceiqcanadaeast.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "Central India": "https://hwxcentralindia.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "Central US": "https://
sequenceiqcentralus2.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "East Asia": "https://sequenceiqeastasia2.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "East US": "https://sequenceiqeastus12.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
```

```
            "East US 2": "https://sequenceiqeastus22.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "Japan East": "https://
sequenceiqjapaneast2.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "Japan West": "https://
sequenceiqjapanwest2.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "Korea Central": "https://hwxkoreacentral.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "Korea South": "https://hwxkoreasouth.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "North Central US": "https://
sequenceiqorthcentralus2.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "North Europe": "https://
sequenceiqnortheurope2.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "South Central US": "https://
sequenceiqouthcentralus2.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "South India": "https://hwxsouthindia.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "Southeast Asia": "https://
sequenceiqsoutheastasia2.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "UK South": "https://hwxsouthuk.blob.core.windows.net/images/
hdc-hdp--1710161226.vhd",
            "UK West": "https://hwxwestuk.blob.core.windows.net/images/hdc-
hdp--1710161226.vhd",
            "West Central US": "https://
hwxwestcentralus.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "West Europe": "https://
sequenceiqwesteurope2.blob.core.windows.net/images/hdc-hdp--1710161226.vhd",
            "West India": "https://hwxwestindia.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "West US": "https://sequenceiqwestus2.blob.core.windows.net/
images/hdc-hdp--1710161226.vhd",
            "West US 2": "https://hwxwestus2.blob.core.windows.net/images/
hdc-hdp--1710161226.vhd"
          },
          "gcp": {
            "default": "sequenceiqimage/hdc-hdp--1710161226.tar.gz"
          },
          "openstack": {
            "default": "hdc-hdp--1710161226"
          }
        },
        "os": "centos7",
        "os_type": "redhat7",
        "uuid": "f6e778fc-7f17-4535-9021-515351df3691",
        "package-versions": {
          "salt": "2017.7.5",
          "salt-bootstrap": "0.13.0-2018-05-03T07:39:07"
        }
      }
    ]
},
  "versions": {
    "cloudbreak": [
      {
        "images": [
          "44b140a4-bd0b-457d-b174-e988bee3ca47",
          "f6e778fc-7f17-4535-9021-515351df3692"
        ],
        "versions": [
```

```
            "2.8.0"
          ]
        }
      ]
    }
  }
```

Related Information

Example image catalog

Register image catalog

Once you've created your image catalog JSON file, register it with your Cloudbreak instance.

You can do this by using one of the following:

- Cloudbreak web UI
- Cloudbreak CLI
- By editing the Profile file

Note:

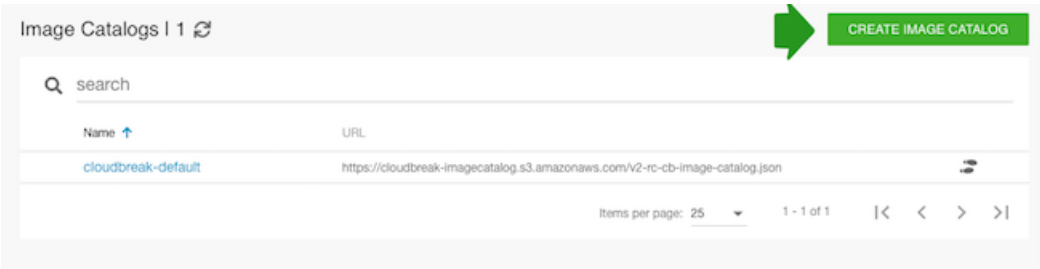
The content type of your image catalog file should be “application/json” for Cloudbreak to be able to process it.

Register image catalog in the UI

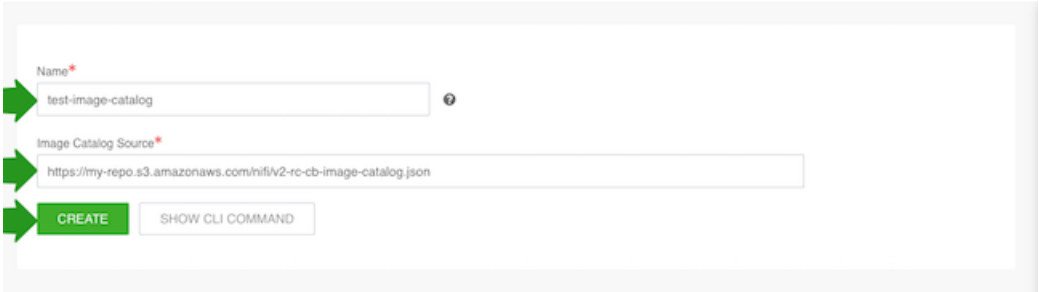
Use these steps to register your custom image catalog in the Cloudbreak web UI.

Steps

1. In the Cloudbreak UI, select External Sources > Image Catalogs from the navigation menu.
2. Click Create Image Catalog:



3. Enter name for your image catalog and the URL to the location where it is stored.
4. Click Create:



After performing these steps, the image catalog will be available and automatically selected as the default entry in the image catalog drop-down list in the create cluster wizard.

Register image catalog in the CLI

To register your custom image catalog using the CLI, use the cb imagecatalog create command. For more information, refer to CLI documentation.

Register image catalog in the Profile

As an alternative to using the web UI or CLI, it is possible to place the catalog file to the Cloudbreak deployer`s etc directory and then set CB_IMAGE_CATALOG_URL variable in your Profile to IMAGE_CATALOG_FILE_NAME.JSON.

Steps

1. On the Cloudbreak machine, switch to the root user by using sudo su
2. Save the image catalog file on your Cloudbreak machine in the /var/lib/cloudbreak-deployment/etc directory.
3. Edit the Profile file located in /var/lib/cloudbreak-deployment by adding export CB_IMAGE_CATALOG_URL to the file and set it to the name of your JSON file which declares your custom images. For example:

```
export CB_IMAGE_CATALOG_URL=custom-image-catalog.json
```

4. Save the Profile file.
5. Restart Cloudbreak by using:

```
cbd restart
```

Set CB_ENABLED_LINUX_TYPES

If using a different operating system than that included on the default images, you must set the CB_ENABLED_LINUX_TYPES variable in Profile.

Note:

If you are using CentOS 6, CentOS 7, Amazon Linux, or Amazon Linux 2, you don't need to perform this step. If your custom images include any other operating system, you must perform this step.

Steps

1. On the Cloudbreak machine, switch to the root user by using sudo su
2. Edit the Profile file located in /var/lib/cloudbreak-deployment by adding export CB_ENABLED_LINUX_TYPES and set it to include all operating systems that you would like to use on your custom images. For example:

```
CB_ENABLED_LINUX_TYPES=redhat6,redhat7,centos6,centos7,amazonlinux
```

   Possible values are: centos6, centos7, redhat6, redhat7, debian9, ubuntu16, ubuntu18, sles12, amazonlinux, amazonlinux2.
3. Save the Profile file.
4. Restart Cloudbreak by using:

```
cbd restart
```

Create clusters with custom images

Once you have registered your image catalog, you can use your custom image(s) when creating a cluster.

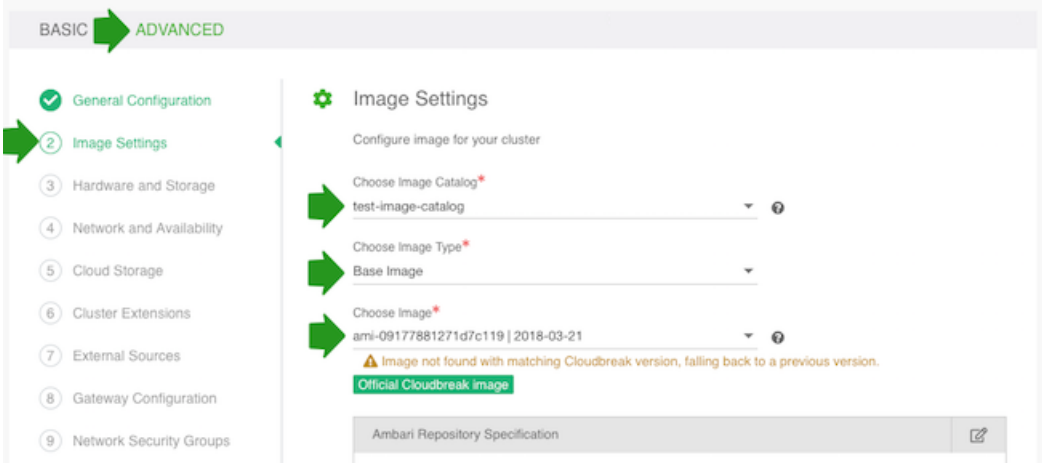
You can do this either with the web UI or CLI.

Select a custom image in Cloudbreak web UI

Perform these steps in the advanced General Configuration section of the create cluster wizard.

Steps

1. In the create cluster wizard, make sure that you are using the advanced wizard version.
2. Navigate to the Image Settings section of the wizard.
3. Under Choose Image Catalog, select your custom image catalog.
4. Under Choose Image Type, select “Base Image”.
5. Under Choose Image, select the provider-specific image that you would like to use. The “os” that you specified in the image catalog will be displayed in the selection and the content of the “description” will be displayed in green:



6. You can leave the default entries for the Ambari and HDP/HDF repositories, or you can customize to point to specific versions of Ambari and HDP/HDF that you want to use for the cluster.

Select a custom image in the CLI

To use the custom image when creating a cluster via CLI, perform these steps.

Steps

1. Obtain the image ID. For example:

```
cb imagecatalog images aws --imagecatalog custom-catalog
[
  {
 "Date": "2017-10-13",
 "Description": "Cloudbreak official base image",
 "Version": "2.5.1.0",
 "ImageID": "44b140a4-bd0b-457d-b174-e988bee3ca47"
  },
  {
 "Date": "2017-11-16",
 "Description": "Official Cloudbreak image",
 "Version": "2.5.1.0",
 "ImageID": "3c7598a4-ebd6-4a02-5638-882f5c7f7add"
  }
]
```

2. When preparing a CLI JSON template for your cluster, set the “ImageCatalog” parameter to the image catalog that you would like to use, and set the “ImageId” parameter to the uuid of the image from that catalog that you would like to use. For example:

```
...
  "name": "aszegedi-cli-ci",
```

```
  "network": {
 "subnetCIDR": "10.0.0.0/16"
  },
  "orchestrator": {
 "type": "SALT"
  },
  "parameters": {
 "instanceProfileStrategy": "CREATE"
  },
  "region": "eu-west-1",
  "stackAuthentication": {
 "publicKeyId": "seq-master"
  },
  "userDefinedTags": {
 "owner": "aszegedi"
  },
  "imageCatalog": "custom-catalog",
  "imageId": "3c7598a4-ebd6-4a02-5638-882f5c7f7add"
}
```

Custom blueprints

In addition to providing default blueprints, Cloudbreak allows you to bring your own custom blueprints. Refer to this section if you would like to use custom blueprints for Cloudbreak-managed clusters.

We recommend that you review the default blueprints to check if they meet your requirements. You can do this by selecting Blueprints from the navigation pane in the Cloudbreak web UI.

Related Information

Default cluster configurations

Creating blueprints

A blueprint exported from a running Ambari cluster can be reused in Cloudbreak after slight modifications.

Ambari blueprints are specified in JSON format. When a blueprint is exported, it includes some hardcoded configurations such as domain names, memory configurations, and so on, that are not applicable to Cloudbreak-managed clusters. There is no automatic way to modify an exported blueprint and make it instantly usable in Cloudbreak, the modifications have to be done manually.

In general, the blueprint should include the following elements:

```
"Blueprints": {
   "blueprint_name": "hdp-small-default",
   "stack_name": "HDP",
   "stack_version": "2.6"
 },
 "settings": [],
 "configurations": [],
 "host_groups": [
 {
     "name": "master",
     "configurations": [],
     "components": []
   },
   {
     "name": "worker",
     "configurations": [],
     "components": [ ]
```

```
    },
    {
      "name": "compute",
      "configurations": [],
      "components": []
    }
  ]
}
```

For correct blueprint layout and other information about Ambari blueprints, refer to the Ambari cwiki page. You can also use the default blueprints provided in the Cloudbreak web UI as a model.

After you provide the blueprint to Cloudbreak, the host groups in the JSON will be mapped to a set of instances when starting the cluster, and the specified services and components will be installed on the corresponding nodes. It is not necessary to define a complete configuration in the blueprint. If a configuration is missing, Ambari will use a default value.

Blueprint name

Cloudbreak requires you to define an additional element in the blueprint called “blueprint_name”. This should be a unique name within Cloudbreak's list of blueprints. For example:

```
"Blueprints": {
   "blueprint_name": "hdp-small-default",
   "stack_name": "HDP",
   "stack_version": "2.6"
 },
 "settings": [],
 "configurations": [],
 "host_groups": [
 ...
```

The “blueprint_name” is not included in the Ambari export.

Blueprints for Ambari 2.6.1 or newer

Ambari 2.6.1 or newer cannot install the mysqlconnector, as the connector is released under version 2 of the GNU General Public License. Therefore, when creating a blueprint for Ambari 2.6.1 or newer you should not include the MYSQL_SERVER component for Hive Metastore in your blueprint. Instead, you have two options:

- Configure an external RDBMS instance for Hive Metastore and include the JDBC connection information in your blueprint. If you choose to use an external database that is not PostgreSQL (such as Oracle, mysql) you must also set up Ambari with the appropriate connector; to do this, create a pre-ambari-start recipe and pass it when creating a cluster.
- If a remote Hive RDBMS is not provided, Cloudbreak installs a Postgres instance and configures it for Hive Metastore during the cluster launch.

For information on how to configure an external database and pass your external database connection parameters, refer to Ambari cwiki.

If you still include MYSQL_SERVER in your blueprint, then depending on your chosen operating system, MariaDB or MySQL Server will be installed.

Related Information

Apache cwiki: Blueprints

Creating dynamic blueprints

Cloudbreak allows you to create dynamic blueprints, which include templating.

The values of the variables specified in the blueprint are dynamically replaced in the cluster creation phase, picking up the parameter values that you provided in the Cloudbreak web UI or CLI. Cloudbreak supports mustache kind of templating with "{{{ }}}" syntax.

Production cluster configurations typically include certain configuration parameters, such as those related to external database (for Hive, Ranger, etc) and LDAP/AD, forcing you to create multiple versions of the same blueprint to handle different component configurations for these external systems. Dynamic blueprints solve this problem by offering the ability to manage external sources (such as RDBMS and LDAP/AD) outside of your blueprint. They merely use the blueprint as a template and Cloudbreak injects the actual configurations into your blueprint. This simplifies the reuse of cluster configurations for external sources (RDBMS and LDAP/AD) and simplifies the blueprints themselves.

Note:

You cannot use functions in the blueprint file; only variable injection is supported.

External authentication source

When using an external authentication (LDAP/AD) source for your cluster, the following variables can be specified in your blueprint for replacement:

Variable	Description	Example
ldap.connectionURL	the URL of the LDAP (host:port)	ldap://10.1.1.1:389
ldap.bindDn	The root Distinguished Name to search in the directory for users	CN=Administrator,CN=Users,DC=ad,DC=hdc,DC=com
ldap.bindPassword	The root Distinguished Name password	Password1234!
ldap.directoryType	The directory of type	LDAP or ACTIVE_DIRECTORY
ldap.userSearchBase	User search base	CN=Users,DC=ad,DC=hdc,DC=com
ldap.userNameAttribute	Username attribute	cn
ldap.userObjectClass	Object class for users	person
ldap.groupSearchBase	Group search base	OU=Groups,DC=ad,DC=hdc,DC=com
ldap.groupNameAttribute	Group attribute	cb
ldap.groupObjectClass	Group object class	group
ldap.groupMemberAttribute	Attribute for membership	member
ldap.domain	Your domain	example.com

Related Information

External authentication source for clusters

External database

When using external databases (RDBMS) for your cluster components, the following variables can be specified in your blueprint for replacement:

Variable	Description	Example
rds.[type].connectionString	The jdbc url to the RDBMS	jdbc:postgresql://db.test:5432/test
rds.[type].connectionDriver	The connection driver	org.postgresql.Driver
rds.[type].connectionUserName	The user name to the database	admin
rds.[type].connectionPassword	The password for the connection	Password1234!
rds.[type].subprotocol	Parsed from jdbc url	postgres
rds.[type].databaseEngine	Capital database name	POSTGRES

Upload blueprints

Once you have your blueprint ready, upload it to Cloudbreak and then select it during cluster creation.

Upload blueprints from the web UI

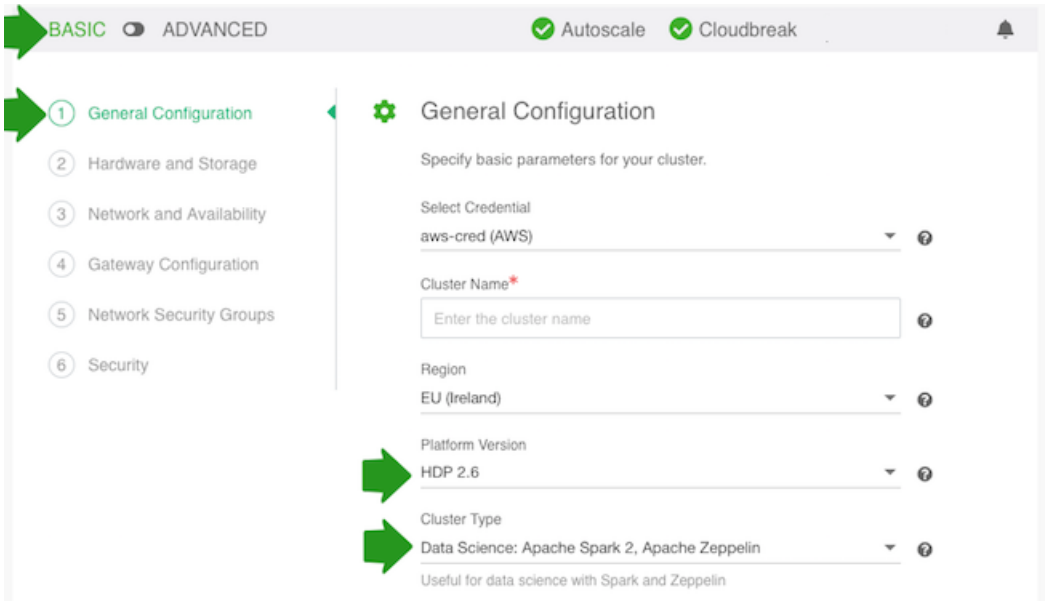
Follow these steps to upload a blueprint from the Cloudbreak web UI.

Steps

1. In the Cloudbreak web UI, select Blueprints from the navigation pane.
2. To add your own blueprint, click Create Blueprint and enter the following parameters:

Parameter	Value
Name	Enter a name for your blueprint.
Description	(Optional) Enter a description for your blueprint.
Blueprint Source	Select one of: • Text: Paste blueprint in JSON format. • File: Upload a file that contains the blueprint. • URL: Specify the URL for your blueprint.

3. To use the uploaded blueprints, select it when creating a cluster. The option is available on the General Configuration page. First select the Platform Version and then select your chosen blueprint under Cluster Type:



Upload blueprint from the CLI

To upload a custom blueprint from the CLI, use the cb blueprint create command. To use the uploaded blueprints, generate a valid JSON template and then create a cluster with cb cluster create.

Related Information

Creating a cluster

Recipes

Cloudbreak allows you to create and run scripts (called "recipes") that perform specific tasks on your cluster nodes. Refer to this section if you would like to create and use recipes.

Although Cloudbreak lets you provision clusters in the cloud based on custom Ambari blueprints, Cloudbreak provisioning options don’t consider all possible use cases. For that reason, recipes (custom scripts) can be used. A recipe is a script that runs on all nodes of a selected node group at a specific time. You can use recipes for tasks such as installing additional software or performing advanced cluster configuration. For example, you can use a recipe to put a JAR file on the Hadoop classpath.

Available recipe execution times are:

- Before Ambari server start
- After Ambari server start
- After cluster installation
- Before cluster termination

You can upload your recipes to Cloudbreak via the web UI or CLI. Then, when creating a cluster, you can optionally attach one or more “recipes” and they will be executed on a specific host group at a specified time.

Writing recipes

Refer to these guidelines when creating your recipes.

When using recipes, consider the following guidelines:

- Cloudbreak supports running bash and python scripts as recipes. We recommend using scripts with Shebang character sequence, for example:

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/sh
#!/usr/bin/bash
#!/usr/bin/env sh
#!/usr/bin/env bash
#!/bin/sh -x
#!/usr/bin/python
#!/usr/bin/env python
```

- The scripts are executed as root. The recipe output is written to /var/log/recipes on each node on which it was executed.
- Supported parameters can be specified as variables by using mustache kind of templating with "{{{ }}}" syntax. Once specified in a recipe, these variables are dynamically replaced when the recipe is executed, generating the actual values that you provided to Cloudbreak as part of cluster creation process.

  For example, if your cluster includes an external LDAP and your recipe includes {{{ldap.connectionURL}}}, as demonstrated in the following example

```
#!/bin/bash -e

main() {
 ping {{{ ldap.connectionURL }}}
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

then, when this recipe runs, the {{{ldap.connectionURL}}} is replaced with the actual connection URL specified as part of cluster creation process, as demonstrated in the following example:

```
#!/bin/bash -e

main() {
 ping 192.168.59.103
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

- Recipe logs can be found at /var/log/recipes/${RECIPE_TYPE}/${RECIPE_NAME}.log
- The scripts are executed on all nodes of the host groups that you select (such as “master”, “worker”, “compute”).
- In order to be executed, your script must be in a network location which is accessible from the Cloudbreak and cluster instances VPC.
- Make sure to follow Linux best practices when creating your scripts. For example, don’t forget to script “Yes” auto-answers where needed.
- Do not execute yum update –y as it may update other components on the instances (such as salt) – which can create unintended or unstable behavior.

Example Python script

```
#!/usr/bin/python
print("An example of a python script")
import sys
print(sys.version_info)
```

Example bash script for yum proxy settings

```
#!/bin/bash
cat >> /etc/yum.conf
<<ENDOF
proxy=http://10.0.0.133:3128
ENDOF
```

Example recipe including variables

Original recipe:

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u {{{ general.userName }}}:{{{ general.password }}}
 -H "X-Requested-By: ambari" -X POST -d '{"RequestInfo":
{"command":"RESTART","context":"Restart all components required
 ATLAS","operation_level":
{"level":"SERVICE","cluster_name":"{{{ general.clusterName }}}","service_name":"ATLAS"}
resource_filters":[{"hosts_predicate":"HostRoles/
stale_configs=false&HostRoles/cluster_name={{{ general.clusterName }}}"}]}'
 http://$(hostname -f):8080/api/v1/clusters/{{{ general.clusterName }}}/
requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Generated recipe (to illustrate how the variables from the original recipe were replaced by Cloudbreak):

```
#!/bin/bash -e

function setupAtlasServer() {
    curl -iv -u admin:admin123 -H "X-Requested-By: ambari" -X POST -d
 '{"RequestInfo":{"command":"RESTART","context":"Restart all components
 required ATLAS","operation_level":{"level":"SERVICE","cluster_name":"super-
cluster","service_name":"ATLAS"}},"Requests/resource_filters":
[{"hosts_predicate":"HostRoles/stale_configs=false&HostRoles/
cluster_name=super-cluster"}]}' http://$(hostname -f):8080/api/v1/clusters/
super-cluster/requests
}

main() {
    setupAtlasServer
}

[[ "$0" == "$BASH_SOURCE" ]] && main "$@"
```

Recipe parameters

The following supported parameters can be specified as variables in recipes by using mustache kind of templating with "{{{ }}}" syntax.

The parameter keys listed below follow the following general conventions:

- { } indicates that the parameter key has multiple supported values, which are provided in this documentation. For example {fileSystemType} can be one of the following: s3, adls, adls_gen_2, wasb, or gcs.
- [index] indicates that the parameter includes an index value for example sharedService.datalakeComponents.[index] can be "sharedService.datalakeComponents.[0]", "sharedService.datalakeComponents.[1]", and so on. There is no easy way to find out what the index will be, but you may still be able to use these parameters (for example by creating a condition to filter them).

Custom properties

Any custom property specified in the blueprint can be used as a recipe parameter. Refer to Custom properties documentation.

General

The general parameter group includes parameters related to general cluster configuration.

Parameter key	Description	Example key	Example value
general.email	Email of the Cloudbreak user.	general.email	cloudbreak@hortonworks.com
general.gatewayInstanceMetadataPresented	Flag indicating if gateway instance metadata is present.	general.gatewayInstanceMetadataPresented	true
general.instanceGroupsPresented	Flag indicates that instance groups are presented.	general.instanceGroupsPresented	true
general.clusterName	Name of cluster.	general.clusterName	testcluster
general.stackName	Name of stack.	general.stackName	teststack
general.uuid	UUID of cluster.	general.uuid	9aab7fdb-8940-454b-bc0a-62f04bce6519
general.userName	Ambari username.	general.userName	admin
general.password	Ambari password.	general.password	admin1234

Parameter key	Description	Example key	Example value
general.executorType	Type of execution. Possible values: DEFAULT or CONTAINER.	general.executorType	DEFAULT
general.ambariIp	Ambari IP.	general.ambariIp	127.0.0.1
general.orchestratorType	Type of cluster orchestration. Possible values: HOST or CONTAINER.	general.orchestratorType	HOST
general.containerExecutor	Flag indicates that the cluster is running containers.	general.containerExecutor	false
general.nodeCount	Number of nodes.	general.nodeCount	5
general.primaryGatewayInstanceDiscoveryFQDN	FQDN of primary gateway instance.	general.primaryGatewayInstanceDiscoveryFQDN	ip-10-0-88-28.example.com
general.kafkaReplicationFactor	Number indicating the Kafka replication factor (3 or 1).	general.kafkaReplicationFactor	1

Attached cluster

The following parameters are only used with clusters attached to a data lake.

Parameter key	Description	Example key	Example value
REMOTE_CLUSTER_NAME	Name of data lake cluster to which the workload cluster is attached.	REMOTE_CLUSTER_NAME	testclusterdatalake
remoteClusterName	Name of data lake cluster to which the workload cluster is attached.	remoteClusterName	testclusterdatalake
remote.cluster.name	Name of data lake cluster to which the workload cluster is attached.	remote.cluster.name	testclusterdatalake
cluster_name	Cluster name.	cluster_name	testcluster
cluster.name	Cluster name.	cluster.name	testcluster
ranger.audit.solr.zookeepers	Ranger Audit URL.	ranger.audit.solr.zookeepers	ip-10-0-137-205.eu-west-1.compute.internal:2181/infra-solr
atlas.rest.address	Atlas component REST address.	atlas.rest.address	http://ip-10-0-137-205.eu-west-1.compute.internal:21000
atlas.kafka.bootstrap.servers	Bootstrap server URL for Atlas Kafka.	atlas.kafka.bootstrap.servers	ip-10-0-137-205.eu-west-1.compute.internal:6667
ranger_admin_username	Username of Ranger admin.	ranger_admin_username	amb_ranger_admin
policymgr_external_url	Load balancer URL of Ranger.	policymgr_external_url	http://ip-10-0-137-205.eu-west-1.compute.internal:6080

Blueprint

The bleuprint parameter group includes parameters related to blueprint configuration.

Parameter key	Description	Example key	Example value
blueprint.blueprintText	Blueprint text in JSON format.	blueprint.blueprintText	
blueprint.version	Version of blueprint.	blueprint.version	3.2
blueprint.type	Type of blueprint.	blueprint.type	HDF
blueprint.components.[index]	Components in the blueprint.	blueprint.components.[0]	TEZ_CLIENT

Blueprint components

The components represented in a blueprint can be used as a recipe parameters. Possible uses of these values are to list the blueprint's components or check if this list contains a specific component.

Parameter key	Description	Example key	Example value
components.[index]	The components represented in a blueprint.	components.[0]	TEZ_CLIENT

Cloud storage

The fileSystemConfigs parameter group includes parameters related to cloud storage configuration.

When forming the parameter keys, the {fileSystemType} should be replaced with an actual cloud storage type such as "s3", "adls", "adls_gen_2", "wasb", or "gcs".

Parameter key	Description	Example key	Example value
File system common configurations			
fileSystemConfigs.{fileSystemType}.storageContainer	Name of container in Azure storage account (Cloudbreak + stackId).	fileSystemConfigs.s3.storageContainer	cloudbreak123
fileSystemConfigs.{fileSystemType}.type	Type of filesystem.	fileSystemConfigs.s3.type	S3
fileSystemConfigs.{fileSystemType}.defaultFs	Flag to indicate if the file system is the default filesystem.	fileSystemConfigs.s3.defaultFs	false
fileSystemConfigs.{fileSystemType}.locations.[index].configFile	Configuration file used to configure the filesystem.	fileSystemConfigs.s3.locations.[0].configFile	hbase-site
fileSystemConfigs.{fileSystemType}.locations.[index].property	Property key of filesystem path in defined config.	fileSystemConfigs.s3.locations.[0].property	hbase.rootdir
fileSystemConfigs.{fileSystemType}.locations.[index].value	Value of filesystem path in defined config.	fileSystemConfigs.s3.locations.[0].value	s3a://ahorvathtestranger/testrecipe2/apps/hbase/data
Amazon S3 configurations			
fileSystemConfigs.s3.instanceProfile	ARN of related instance profile in AWS	fileSystemConfigs.s3.instanceProfile	arn:aws:iam::980678866538:instance-profile/CloudbreakRole
WASB configurations			
fileSystemConfigs.wasb.accountKey	Access key of the corresponding Azure storage account.	fileSystemConfigs.wasb.accountKey	81a9blll-bebf-436f-a333-f67b29880f1z
fileSystemConfigs.wasb.accountName	Name of the corresponding Azure storage account.	fileSystemConfigs.wasb.accountName	teststorageaccount
fileSystemConfigs.wasb.secure	Flag indicating that the file system is secure.	fileSystemConfigs.wasb.secure	true
fileSystemConfigs.wasb.resourceGroupName	Name of the corresponding Azure resource group.	fileSystemConfigs.wasb.resourceGroupName	testresourcegroup
fileSystemConfigs.wasb.storageContainerName	Name of container in Azure storage account.	fileSystemConfigs.wasb.storageContainerName	testcontainer
ADLS Gen1 configurations			
fileSystemConfigs.adls.accountName	Name of the corresponding Azure storage account.	fileSystemConfigs.adls.accountName	teststorageaccount
fileSystemConfigs.adls.clientId	The corresponding Azure client ID.	fileSystemConfigs.adls.clientId	a9a9a88e-28dc-4851-ad3d-182a08c44666

Parameter key	Description	Example key	Example value
fileSystemConfigs.adls.tenantId	Tenant ID of Azure account.	fileSystemConfigs.adls.tenantId	d85131e4-1763-42d6-b9c7-b6bad64b3a51
fileSystemConfigs.adls.resourceGroupName	Name of the corresponding Azure resource group.	fileSystemConfigs.adls.resourceGroupName	myresourcegroup
ADLS Gen2 configurations			
fileSystemConfigs.adls_gen_2.accountName	Name of the corresponding Azure storage account.	fileSystemConfigs.adls_gen_2.accountName	mystorageaccount
fileSystemConfigs.adls_gen_2.accountKey	Access key of the corresponding Azure storage account.	fileSystemConfigs.adls_gen_2.accountKey	8f8f9lll-bebf-436f-a333-f67b29880f1z
fileSystemConfigs.adls_gen_2.storageContainerName	Container name in Azure storage account.	fileSystemConfigs.adls_gen_2.storageContainerName	mycontainer
GCS configurations			
fileSystemConfigs.gcs.serviceAccountEmail	Email id of the user's GCS account.	fileSystemConfigs.gcs.serviceAccountEmail	mail@gmail.com

External authentication source

The ldap parameter group includes parameters related to external authentication source configuration.

Parameter key	Description	Example key	Example value
ldap.bindDn	LDAP Bind DN.	ldap.bindDn	Admin2@AD.HWX.COM
ldap.bindPassword	Root Distinguished Name (Bind DN) password.	ldap.bindPassword	Admin1234
ldap.directoryType	Directory type. Possible values: LDAP or ACTIVE_DIRECTORY.	ldap.directoryType	ACTIVE_DIRECTORY
ldap.userSearchBase	LDAP user search base. This defines the location in the directory from which the LDAP search begins.	ldap.userSearchBase	OU=Users,OU=AD,DC=AD,DC=HWX,DC=COM
ldap.userNameAttribute	The attribute for which to conduct a search on the user base.	ldap.userNameAttribute	sAMAccountName
ldap.userObjectClass	Directory object class for users.	ldap.userObjectClass	person
ldap.groupSearchBase	LDAP group search base. This defines the location in the directory from which the LDAP search begins.	ldap.groupSearchBase	OU=Users,OU=AD,DC=AD,DC=HWX,DC=COM
ldap.groupNameAttribute	The attribute for which to conduct search on groups.	ldap.groupNameAttribute	cn
ldap.groupObjectClass	The directory object class for groups.	ldap.groupObjectClass	group
ldap.groupMemberAttribute	The attribute on the group object class that represents members.	ldap.groupMemberAttribute	member
ldap.domain	Domain of LDAP.	ldap.domain	ad.hwx.com
ldap.protocol	Protocol used by the LDAP: LDAP or LDAPS.	ldap.protocol	ldap
ldap.adminGroup	Name of the admin group.	ldap.adminGroup	cloudbreak
ldap.userDnPattern	LDAP User DN Pattern, which is used to bind an LDAP user.	ldap.userDnPattern	CN={0},OU=Users,OU=AD,DC=AD,DC=HWX,DC
ldap.connectionURL	Full connection URL of the authentication source.	ldap.connectionURL	ldap://hwxmsad-bd87e95aa9775a71.elb.eu-west-1.amazonaws.com:389

Parameter key	Description	Example key	Example value
ldap.host	Host of the authentication source (without protocol).	ldap.host	hwxmsad-bd87e95aa9775a71.elb.eu-west-1.amazonaws.com
ldap.port	Port of the authentication source.	ldap.port	389

External database

The rds parameter group includes parameters related to external database configuration.

When forming the parameter keys, the {rdsType} should be replaced with the actual database type such as "ambari", "beacon", "druid", "hive", "oozie", "ranger", "superset", or some other user-defined type.

Parameter key	Description	Example key	Example value
rds.{rdsType}.connectionURL	JDBC connection URL.	rds.hive.connectionURL	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.connectionDriver	JDBC driver used for connection.	rds.hive.connectionDriver	org.postgresql.Driver
rds.{rdsType}.connectionUserName	Username used for the JDBC connection.	rds.hive.connectionUserName	testuser
rds.{rdsType}.connectionPassword	Password used for the JDBC connection.	rds.hive.connectionPassword	TestPssword123
rds.{rdsType}.databaseName	Target database of the JDBC connection.	rds.hive.databaseName	myhivedb
rds.{rdsType}.host	Host of the JDBC connection.	rds.hive.host	mydbhost
rds.{rdsType}.hostWithPortWithJdbc	Host of JDBC connection with port and JDBC prefix.	rds.hive.hostWithPortWithJdbc	Value is specified in the following format: jdbc:postgresql://host:port
rds.{rdsType}.subprotocol	Sub-protocol from the JDBC URL.	rds.hive.subprotocol	postgresql
rds.{rdsType}.connectionString	URL of JDBC the connection. In case of Ranger, this does not contain the port.	rds.hive.connectionString	Value is specified in the following format: jdbc:postgresql://host:port/database
rds.{rdsType}.databaseVendor	Database vendor.	rds.hive.databaseVendor	POSTGRES
rds.{rdsType}.withoutJDBCPrefix	URL of the JDBC connection without JDBC prefix.	rds.hive.withoutJDBCPrefix	Value is specified in the following format: host:port/database

Gateway

The gateway parameter group includes parameters related to gateway configuration.

Parameter key	Description	Example key	Example value
gateway.gatewayType	Type of gateway. Possible values: CENTRAL/INDIVIDUAL.	gateway.gatewayType	CENTRAL
gateway.path	Base path of gateway (typically this is the name of the cluster).	gateway.path	test
gateway.ssoType	Type of SSO. Possible values: SSO_PROVIDER/NONE.	gateway.ssoType	SSO_PROVIDER
gateway.ssoConfigured	Flag indicating if SSO is provided.	gateway.ssoConfigured	true
gateway.ssoProvider	Path to the SSO provider.	gateway.ssoProvider	/test/sso/api/v1/websso
gateway.signKey	Base64 encoded signing key.	gateway.signKey	
gateway.signPub	Signing certificate (x509 format).	gateway.signPub	

Parameter key	Description	Example key	Example value
gateway.signCert	Public SSH key used for signing (standard public key format).	gateway.signCert	
gateway.gatewayTopologies.{topologyName}	List of exposed services in a specific topology. Value is specified in JSON format.	gateway.gatewayTopologies.dp-proxy	{"services":["AMBARI","SPARK2HISTORYUI","LIVYSERVE

HDF

The hdf parameter group includes parameters related to HDF configuration.

Parameter key	Description	Example key	Example value
hdf.nodeEntities	NiFi node entities content (needed in nifi-ambari-ssl-config configuration).	hdf.nodeEntities	<property name="Node Identity 1">CN=ip-10-0-85-196.eu-west-1.compute.internal, OU=NIFI</property>
hdf.registryNodeEntities	NiFi registry node entities content (needed in nifi-registry-ambari-ssl-config configuration).	hdf.registryNodeEntities	<property name="NiFi Identity 1">CN=ip-10-0-85-196.eu-west-1.compute.internal, OU=NIFI</property>
hdf.nodeUserEntities	NiFi node user entities content.	hdf.nodeUserEntities	<property name="Initial User Identity 1">CN=ip-10-0-85-196.eu-west-1.compute.internal, OU=NIFI</property>
hdf.proxyHosts	List of proxy hosts (needed in nifi-properties configuration).	hdf.proxyHosts	34.244.122.193:9091

Shared services

The sharedService parameter group includes parameters related to data lake configuration.

Parameter key	Description	Example key	Example value
sharedService.rangerAdminPassword	Admin password of the Ranger component.	sharedService.rangerAdminPassword	Admin1234
sharedService.attachedCluster	Flag indicating that the cluster is attached to a data lake cluster.	sharedService.attachedCluster	true
sharedService.datalakeCluster	Flag indicating that the cluster is a data lake cluster.	sharedService.datalakeCluster	true
sharedService.rangerAdminPort	Admin port of the Ranger component.	sharedService.rangerAdminPort	6080
sharedService.datalakeAmbariIp	Ambari IP of data lake cluster.	sharedService.datalakeAmbariIp	127.0.0.1
sharedService.datalakeAmbariFqdn	Ambari FQDN of data lake cluster (or the IP if FQDN cannot be found).	sharedService.datalakeAmbariFqdn	ip-10-0-88-28.example.com
sharedService.datalakeComponents[index]	Data lake component list.	sharedService.datalakeComponents[0]	METRICS_COLLECTOR

Stack Version

Parameter key	Description	Example key	Example value
stack_version	Stack (HDP or HDF) version.		3.2

Related Information

Custom properties

Add recipes

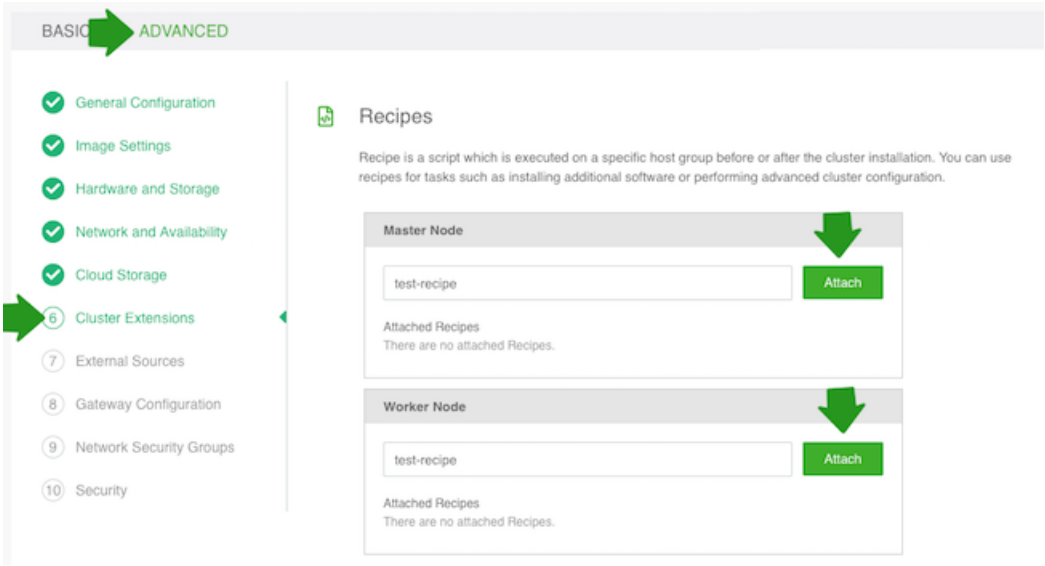
In order to use your recipe for clusters, you must first register it with Cloudbreak.

Steps

1. Place your script in a network location accessible from Cloudbreak and cluster instances virtual network.
2. Select External Sources > Recipes from the navigation menu.
3. Click on Create Recipe.
4. Provide the following:

Parameter	Value
Name	Enter a name for your recipe.
Description	(Optional) Enter a description for your recipe.
Execution Type	Select one of the following options: • pre-ambari-start: The script will be executed prior to Ambari server start. • post-ambari-start: The script will be executed after Ambari server start but prior to cluster installation. • post-cluster-install: The script will be executed after cluster deployment. • pre-termination: The script will be executed before cluster termination.
Script	Select one of: • Script: Paste the script. • File: Point to a file on your machine that contains the recipe. • URL: Specify the URL for your recipe.

5. When creating a cluster, you can select and attach previously added recipes on the advanced Cluster Extensions page of the create cluster wizard:



Related Information

Creating a cluster

Reusable recipes

The following section includes recipes for running common tasks.

Install mysql connector recipe

This recipe can be used to manually install and register the ‘mysql-connector-java.jar’.

Starting from Ambari version 2.6, if you have ‘MYSQL_SERVER’ component in your blueprint, you have to manually install and register the ‘mysql-connector-java.jar’. If you would like to automate this process in Cloudbreak:

- Review the recipe content to ensure that the version of the connector provided in the recipe is as desired; if it is not adjust the version.
- Apply the recipe as “pre-ambari-start”.

The recipe content is:

```
#!/bin/bash

download_mysql_jdbc_driver() {
  wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-
java-5.1.39.tar.gz -P /tmp
  tar xzf /tmp/mysql-connector-java-5.1.39.tar.gz -C /tmp/
  cp /tmp/mysql-connector-java-5.1.39/mysql-connector-java-5.1.39-bin.jar /
opt/jdbc-drivers/mysql-connector-java.jar
}

main() {
  download_mysql_jdbc_driver
}

main
```

Related Information

Using Hive with MySQL/MariaDB (Ambari)

Management packs

Cloudbreak supports using management packs, allowing you to register them in Cloudbreak web UI and CLI and then select to install them as part of cluster creation. Refer to this section if you would like to use management packs with Cloudbreak.

Management packs allow you to deploy a range of services to your Ambari-managed cluster. You can use a management pack to deploy a specific component or service, such as HDP Search, or to deploy an entire platform, such as HDF.

For general information on management packs, refer to Apache cwiki.

Related Information

Apache cwiki: Management Packs

Creating a cluster

Add management pack

In order to have a management stack installed for a specific cluster, you must register it with Cloudbreak by using the following steps.

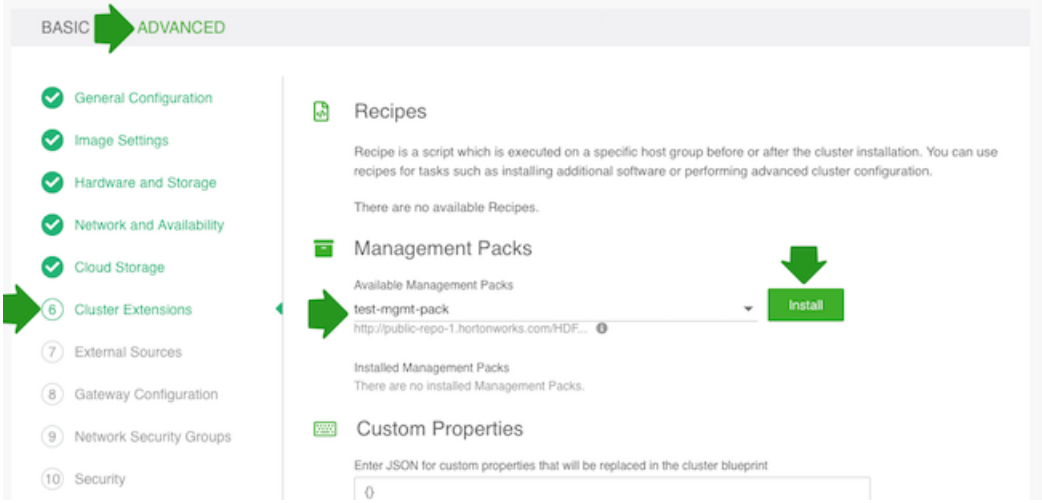
Steps

1. Obtain the URL for the management pack tarball file that you want to register in Cloudbreak. The tarball must be available in a location accessible to clusters created by Cloudbreak.
2. In Cloudbreak web UI, select External Sources > Management Packs from the navigation menu.
3. Click on Register Management Pack.

4. Provide the following:

Parameter	Value
Name	Enter a name for your management pack.
Description	(Optional) Enter a description.
Management pack URL	Provide the URL to the location where the management pack tarball file is available.
Remove all existing Ambari stack definitions prior to installing this Management Pack (“mpack –purge”).	Checking this option allows you to purge any existing stack definition and should be included only when installing a stack management pack. Do not select this when installing an add-on service management pack.

5. When creating a cluster, on the advanced Cluster Extensions page of the create cluster wizard, you can select one or more previously registered management packs. After selecting, click Install to use the management pack for the cluster:



Kerberos security

When creating a cluster via Cloudbreak, you can optionally enable Kerberos security in that cluster and provide your Kerberos configuration details. Cloudbreak will automatically extend your blueprint configuration with the defined properties. Refer to this section if you would like to use Kerberos security with Cloudbreak-managed clusters.

Kerberos overview

Kerberos is a third party authentication mechanism, in which users and services that users wish to access Hadoop rely on a third party - the Kerberos server - to authenticate each to the other.

The Kerberos server itself is known as the Key Distribution Center, or KDC. At a high level, the KDC has three parts:

- A database of the users and services (known as principals) and their respective Kerberos passwords
- An Authentication Server (AS) which performs the initial authentication and issues a Ticket Granting Ticket (TGT)
- A Ticket Granting Server (TGS) that issues subsequent service tickets based on the initial TGT

A user principal requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal’s Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS. Service tickets are what allow the principal to access various services.

Since cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a keytab, which contains the resource principal authentication credentials. The set of hosts, users, and services over which the Kerberos server has control is called a realm.

The following table explains the Kerberos related terminology:

Term	Description
Key Distribution Center, or KDC	The trusted source for authentication in a Kerberos-enabled environment.
Kerberos KDC Server	The machine, or server, that serves as the Key Distribution Center (KDC).
Kerberos Client	Any machine in the cluster that authenticates against the KDC.
Principal	The unique name of a user or service that authenticates against the KDC.
Keytab	A file that includes one or more principals and their keys.
Realm	The Kerberos network that includes a KDC and a number of clients.

Enabling Kerberos

The option to enable Kerberos is available in the advanced Security section of the create cluster wizard. You have the following options for enabling Kerberos in a Cloudbreak-managed cluster:

Option	Description	Environment
Use existing KDC	Allows you to leverage an existing MIT KDC or Active Directory for enabling Kerberos with the cluster. You can either provide the required parameters and Cloudbreak will generate the descriptors on your behalf, or provide the exact Ambari Kerberos descriptors to be injected into your blueprint in JSON format.	Suitable for production
Use test KDC	Installs a new MIT KDC on the master node and configures the cluster to leverage that KDC.	Suitable for evaluation and testing only, not suitable for production

Related Information

Creating a cluster

Using existing KDC

To use an existing KDC, in the advanced Security section of the create cluster wizard select Enable Kerberos Security. By default, Use Existing KDC option is selected.

You must provide the following information about your MIT KDC or Active Directory. Based on these parameters, kerberos-env and krb5-conf JSON descriptors for Ambari are generated and injected into your Blueprint:

Note:

Before proceeding with the configuration, you must confirm that you met the requirements by checking the boxes next to all requirements listed. The configuration options are displayed only after you have confirmed all the requirements by checking every box.

Parameter	Description
Kerberos Admin Principal	The admin principal in your existing MIT KDC or AD.
Kerberos Admin Password	The admin principal password in your existing MIT KDC or AD.
MIT KDC or Active Directory	Select MIT KDC or Active Directory.

Use basic configuration

Parameter	Required if using…	Description
Kerberos Url	MIT, AD	IP address or FQDN for the KDC host. Optionally a port number may be included. Example: “kdc.example1.com:88” or “kdc.example1.com”
Kerberos Admin URL	MIT, AD	(Optional) IP address or FQDN for the KDC admin host. Optionally a port number may be included. Example: “kdc.example2.com:88” or “kdc.example2.com”
Kerberos Realm	MIT, AD	The default realm to use when creating service principals. Example: “EXAMPLE.COM”
Kerberos AD Ldap Url	AD	The URL to the Active Directory LDAP Interface. This value must indicate a secure channel using LDAPS since it is required for creating and updating passwords for Active Directory accounts. Example: “ldaps://ad.example.com:636”
Kerberos AD Container DN	AD	The distinguished name (DN) of the container used store service principals. Example: “OU=hadoop,DC=example,DC=com”
Use TCP Connection	Optional	By default, Kerberos uses UDP. Checkmark this box to use TCP instead.

Use advanced configuration

Checking the Use Custom Configuration option allows you to provide the actual Ambari Kerberos descriptors to be injected into your blueprint (instead of Cloudbreak generating the descriptors on your behalf). This is the most powerful option which gives you full control of the Ambari Kerberos options that are available. You must provide:

- Kerberos-env JSON Descriptor (required)
- krb5-conf JSON Descriptor (optional)

To learn more about the Ambari Kerberos JSON descriptors, refer to Apache cwiki.

Related Information

Apache cwiki: Kerberos Configurations

Using test KDC

To use a test KDC, in the advanced Security section of the create cluster wizard select Enable Kerberos Security and then select Use Test KDC.

Note:

Using the Test KDC is for evaluation and testing purposes only, and cannot be used for production clusters. To enable Kerberos for production use, you must use the Use Existing KDC option.

You must provide the following parameters for your new test KDC:

Parameter	Description
Kerberos Master Key	The master key for the KDC database.
Kerberos Admin Username	The admin principal to create that can administer the KDC.
Kerberos Admin Password	The admin principal password.
Confirm Kerberos Admin Password	The admin principal password.

When using the test KDC option:

- Cloudbreak installs an MIT KDC instance on the Ambari server node.
- Kerberos clients are installed on all cluster nodes, and the krb5.conf is configured to use the MIT KDC.
- The cluster is configured for Kerberos to use the MIT KDC. Very basic Ambari KSON Kerberos descriptors are generated and used accordingly.

Example kerberos-env JSON descriptor file:

```
{
      "kerberos-env" : {
        "properties" : {
          "kdc_type" : "mit-kdc",
          "kdc_hosts" : "ip-10-0-121-81.ec2.internal",
          "realm" : "EC2.INTERNAL",
          "encryption_types" : "aes des3-cbc-sha1 rc4 des-cbc-md5",
          "ldap_url" : "",
          "admin_server_host" : "ip-10-0-121-81.ec2.internal",
          "container_dn" : ""
        }
      }
    }
```

Example krb5-conf JSON descriptor file:

```
{
      "krb5-conf" : {
        "properties" : {
          "domains" : ".ec2.internal",
          "manage_krb5_conf" : "true"
        }
      }
    }
```

To learn more about the Ambari Kerberos JSON descriptors, refer to Apache cwiki.

Related Information

Apache cwiki: Kerberos Configurations

EBS encryption on AWS

Cloudbreak allows you to configure encryption for Amazon Elastic Block Store (EBS) volumes used by the cluster's VM instances to store data. Refer to this section if you would like to encrypt EBS volumes used for clusters running on AWS.

Amazon's Key Management System (KMS) or external KMS generated keys can be used.

Since an encryption key must be specified for each host group, it is possible to either have one encryption key for multiple host groups or to have a separate encryption key for each host group. Once enabled, encryption is configured for the following disk types:

- Block devices
- Root devices

Once the encryption is configured for a given host group, it is automatically applied to any new devices added as a result of cluster scaling.

Overview of configuring EBS encryption

In order to configure EBS encryption:

- Your Cloudbreak credential must have the minimum access permissions.

- Your encryption key must fulfill the following criteria:
  - It must be located in the same region where you would like to create clusters with encrypted volumes.
  - Your IAM user (if using role-based credential) or IAM role (if using key-based credential) must be assigned to the encryption key as both key administrator and key user.
  - The AWSServiceRoleForAutoScaling built-in role must be assigned to the encryption key as both key administrator and key user.
- When creating a cluster, you must explicitly select an existing encryption key for each host group on which you would like to configure EBS volume encryption.

These requirements are described in the sections listed below.

Related Information

Amazon EBS Encryption (AWS)

Permissions for using EBS encryption

If planning to use encryption, ensure that the IAM role (if using role-based credential) or IAM user (if using key-based credential) that you are using for the Cloudbreak credential has the following permissions.

EC2 permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage",
      "ec2:CreateSnapshot",
      "ec2:DeleteSnapshot",
      "ec2:DescribeSnapshots",
      "ec2:CreateVolume",
      "ec2:DeleteVolume",
      "ec2:DescribeVolumes",
      "ec2:DeregisterImage",
    ],
    "Resource": "*"
  }
}
```

KMS permissions

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:ListKeys",
      "kms:ListKeyPolicies",
      "kms:ListAliases"
    ],
    "Resource": "*"
  }
}
```

Related Information

Create CredentialRole

Prerequisites for key-based authentication

Encryption key requirements

If planning to use encryption, ensure that your encryption key can be used with Cloudbreak or if you need to create a new encryption key.

Ensuring that an existing encryption key can be used with Cloudbreak

If you have an existing encryption key that you would like to use with Cloudbreak, make sure that the following are attached as both key administrator and key user:

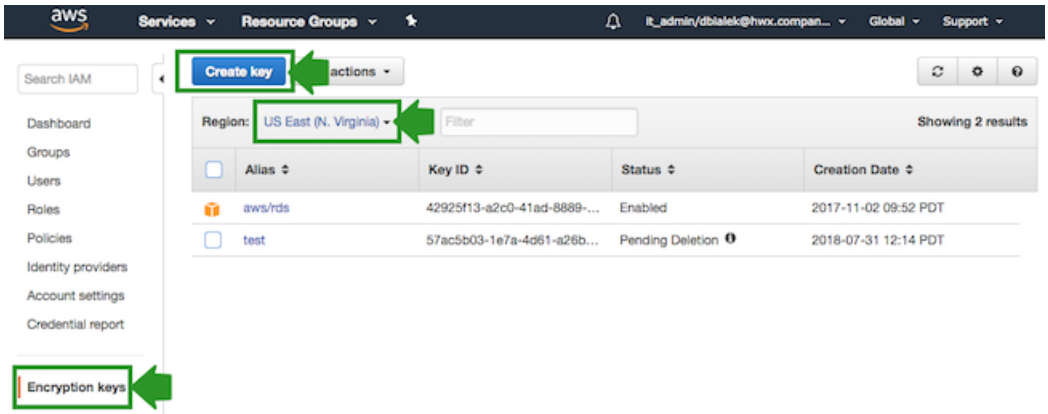
- The AWSServiceRoleForAutoScaling built-in role.
- Your IAM role or IAM user used for the Cloudbreak credential.

To check that these are attached, navigate to the IAM console > Encryption keys, select your encryption key, and scroll to Key Administrators and then Key Users.

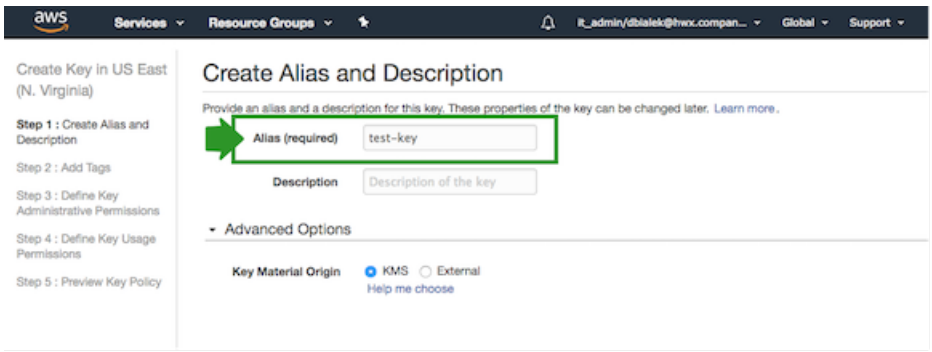
Create a new encryption key on AWS

To create a new encryption key, follow these steps:

1. On AWS, navigate to the IAM console.
2. Select Encryption keys.
3. From the Region dropdown, select the region in which you would like to create and use the encryption key.
4. Click Create key:

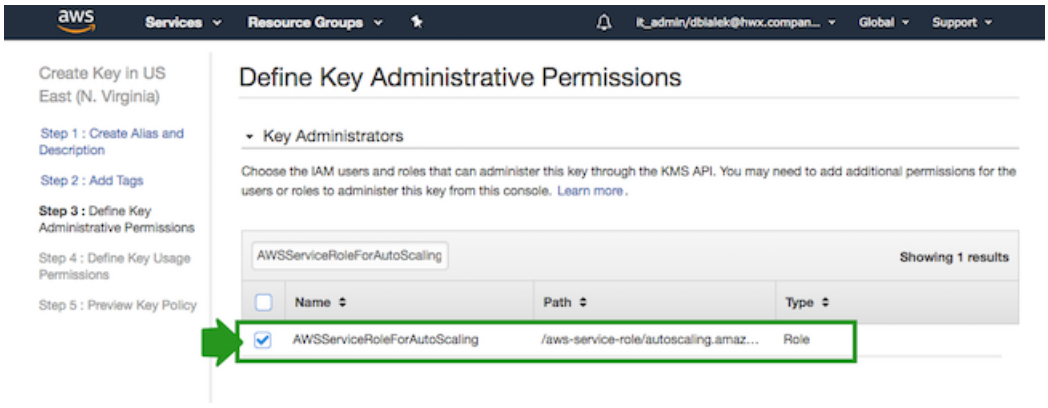


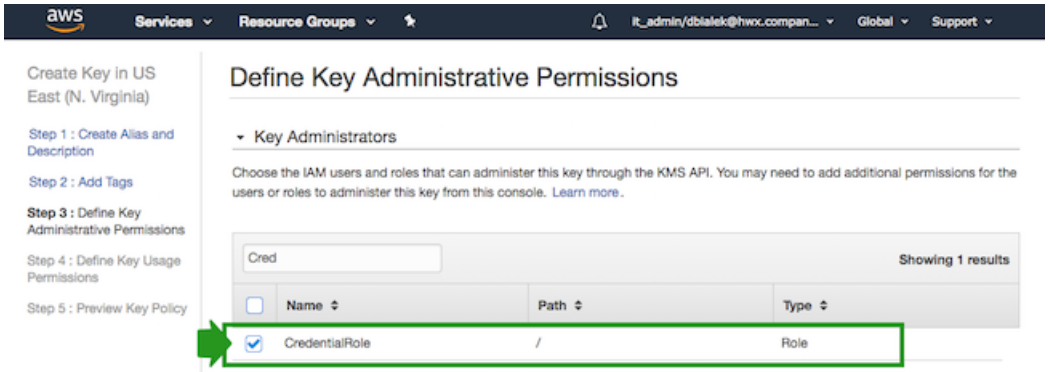
5. In Step 1: Create Alias and Description:
   a. Enter an Alias for your key.
   b. Expand Advanced Options and under Key Material Origin, select “KMS” or “External”.



6. In Step 3: Define Key Administrative Permissions, select the following:

a. AWSDerviceRoleForAutoScaling built-in role.
b. Your IAM user (if using role-based credential) or IAM role (if using key-based credential).





7. In Step 4: Define Key Usage Permissions, select the same items as in the previous steps.
8. Navigate to the last page of the wizard and then click Finish to create an encryption key.

Create a cluster with encrypted EBS volumes

EBS encryption can be configured on the Hardware and Storage page of the advanced create cluster wizard.

The Encryption configuration option is available per host group. The default setting is Encryption: Not encrypted. To enable encryption for a given host group:

1. Under Instance Type you can see “Encryption Supported” next to all instance types for which encryption is supported. Ensure that encryption is supported for the instance type that you would like to use.
2. Click on the

icon next to the chosen host group.
3. Under Encryption key, select the encryption key that you would like to use:

- To use the default encryption key, select "Default" from the dropdown.
- To use a custom key, select it from the dropdown.

Note that when encryption option is selected, the cluster creation process takes a few minutes longer than usual.

Once the cluster is running, you can confirm that encryption is enabled by navigating to cluster details > Hardware tab. The Key ID of the encryption key is also displayed with a link redirecting you to the AWS IAM console. Furthermore, if in the EC2 console on AWS you navigate to details of the block devices or root devices, you can see that the device is marked as “Encrypted” and the “KMS Key ARN” is listed.

Related Information

Creating a cluster

Disk encryption on GCP

Cloudbreak supports encryption options available on Google Cloud's Compute Engine. Refer to this section if you would like to encrypt key encryption keys used for cluster storage on Google Cloud.

As stated in Protecting resources with Cloud KMS Keys in Google Cloud documentation, "By default, Compute Engine encrypts customer content at rest. Compute Engine handles and manages this encryption for you without any additional actions on your part. However, if you want to control and manage this encryption yourself, you can use key encryption keys. Key encryption keys do not directly encrypt your data but are used to encrypt the data encryption keys that encrypt your data."

Google Cloud's Compute Engine offers two options for these key encryption keys:

- Using the Cloud Key Management Service to create and manage encryption keys, known as "customer-managed encryption keys" (CMEK).
- Creating and managing your own encryption keys, known as "customer-supplied encryption keys" (CSEK).

When Cloudbreak provisions resources in Compute Engine on your behalf, Compute Engine applies data encryption as usual and you have an option to configure one of these two methods to encrypt the encryption keys that are used for data encryption.

Since an encryption option must be specified for each host group, it is possible to either have one encryption key for multiple host groups or to have a separate encryption key for each host group. Once the encryption is configured for a given host group, it is automatically applied to any new devices added as a result of cluster scaling.

Overview of configuring key encryption

In order to configure encryption key encryption by using a KMS key (CMEK) or a custom key (CSEK):

- You must enable all required APIs and permissions as described in Google Cloud documentation.
- Your encryption key must be in the same project and location where you would like to create clusters.
- The service account used for the Cloudbreak credential must have the minimum permissions.
- When creating a cluster, you must explicitly select an existing encryption option for each host group on which you would like to configure disk encryption.

These requirements are described in detail in the following sections.

Related Information

Protecting resources with Cloud KMS Keys (GCP)

Encryption key requirements

If planning to use encryption, ensure that your Google Cloud configuration meets the following requirements.

When fulfilling Google Cloud's prerequisites (as described in Protecting resources with cloud KMS keys) and creating encryption keys (as described in Creating key rings and keys) on Google Cloud, ensure that you do the following:

- Compute Engine and Cloud KMS must be in the same Google Cloud Platform project (not in two different projects). Furthermore, this must be the same project where you are planning to launch clusters.
- Set up API access for Compute Engine.
- Enable the Cloud KMS API.
- Create the key rings and keys as described in Creating key rings and keys in Google Cloud documentation. Note that your encryption keys must be in the same location (or "region") where you are planning to launch clusters.

- Assign the Cloud KMS CryptoKey Encrypter/Decrypter role to the Compute Engine system service account (service-[PROJECT_NUMBER]@compute-system.iam.gserviceaccount.com).

Related Information

Protecting resources with Cloud KMS Keys (GCP)
Creating key rings and keys (GCP)
Encrypting disks with customer-supplied encryption keys (GCP)

Permissions required for key encryption

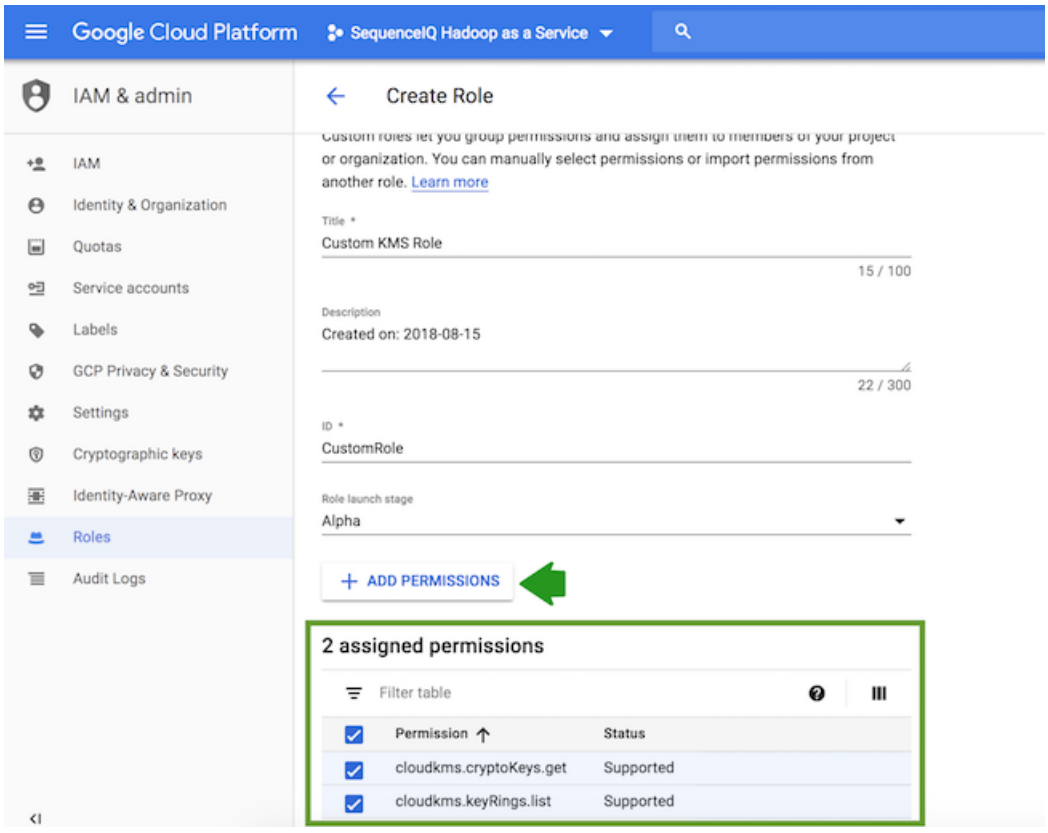
If planning to use encryption key encryption, ensure that you configure the following permissions.

Cloudbreak credential service account's permissions

If you would like to use a KMS key (CMEK) or a custom key (CSEK), you must:

1. Create a new custom role with the following permissions:
   a. cloudkms.cryptoKeys.get
   b. cloudkms.keyRings.list

   To create a custom role, navigate to IAM & admin > Roles and click on +Create role. Next, click on +Add permissions, select and add the required permissions, and click on Create:

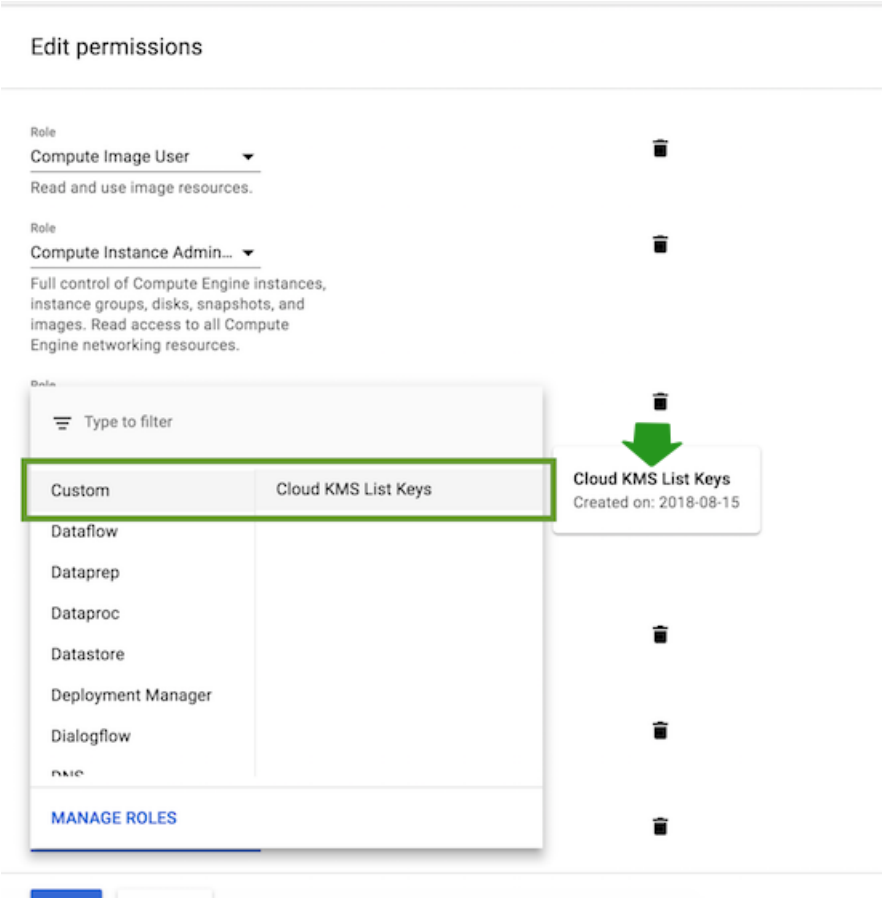


Note:

For testing purposes, it is also possible to use the built-in Cloud KMS Admin role.

2. Assign this role to your Cloudbreak credential's service account.

   To assign this role, on Google Cloud portal, navigate to IAM & admin > IAM, edit the service account that you created for Cloudbreak credential, click on +Add another role, add the newly created custom role and click on Save. You should find this role under "Custom":



Compute Engine system service account's permissions

Assign the Cloud KMS CryptoKey Encrypter/Decrypter role to the Compute Engine system service account as described in Protecting resources with Cloud KMS Keys in Google Cloud documentation. This is required by Google Cloud.

Create a cluster with key encryption

GCP disk encryption can be configured on the Hardware and Storage page of the advanced create cluster wizard.

You can configure it per host group by clicking on the

icon next to the chosen host group. Under Encryption Key you can choose between default encryption, KMS encryption key, or user-provided custom encryption key.

Encryption type	Description	How to configure
Default	The Default encryption option is selected by default because Compute Engine encrypts all customer content at rest. There is no option to turn disk encryption off.	You do not need to do anything.
Select an existing KMS key (CMEK)	You can optionally select a previously created KMS key. If such as a key exists in the selected region, you can select it from the list. The format will be <key-ring-name>/<key>.	From the Encryption Key dropdown, select an existing key.

Encryption type	Description	How to configure
Enter an existing custom key (CSEK)	You can optionally provide a custom key. In this case, you must provide a key (max. 255 characters long) and the method used to send this key to Google: either RAW unencrypted format, or RSA encrypted format. Either way, an SHA-256 hashed version of the provided key will be sent, because GCP expects 256 bytes long blocks.	1. Select “Provide Custom Key” from the Encryption Key dropdown. 2. Under Encryption Method, select RSA or RAW. 3. Under Custom Encryption Key, paste the encryption key.

Once the cluster is running, you can confirm that encryption is enabled by navigating to cluster details > Hardware tab. On Google Cloud, you can navigate to Compute Engine > disks > click on disk name, and you should see the Encryption key ID listed.

Related Information

Creating a cluster

External databases for cluster components

Cloudbreak allows you to register an existing RDBMS instance as an external source to be used for a database for certain services. After you register the RDBMS with Cloudbreak, you can use it for multiple clusters. Refer to this section if you would like to use external databases for cluster components (such as Ambari, Hive, and so on).

The general steps for configuring an external database are:

1. Review the supported databases and then create the external database prior to registering it with Cloudbreak.
2. Review the available options to find out which type to use.
3. Create a template blueprint.
4. Register an existing database in the Cloudbreak web UI or CLI.

Once registered, the database will now show up in the list of available databases when creating a cluster under advanced External Sources > Configure Authentication. Create a cluster by using the blueprint and by attaching the database.

Supported databases

Review the supported database types and versions to ensure that you are using one that is supported.

Only supported database types and versions can be used as external databases.

- If you would like to use an external database for one of the components that support it, you may use the database types and versions defined in the Support Matrix.
- For more information on whether a default database is provided by default and for steps for configuring an external database, refer to the component-specific documentation:

Component	Documentation link
Ambari	Refer to Using Existing Databases - Ambari or to documentation for the specific version that you would like to use.
Druid	Refer to Configuring Druid and Superset Metadata Stores in MySQL or to documentation for the specific version that you would like to use.
Hive	Refer to Using Existing Databases - Hive or to documentation for the specific version that you would like to use.
Oozie	Refer to Using Existing Databases - Oozie or to documentation for the specific version that you would like to use.

Component	Documentation link
Ranger	Refer to Using Existing Databases - Ranger or to documentation for the specific version that you would like to use.
Registry	Only a MySQL database can be used. Other database types are not supported.
Superset	Refer to Configuring Druid and Superset Metadata Stores in MySQL or to documentation for the specific version that you would like to use.
Other	Refer to the component-specific documentation.

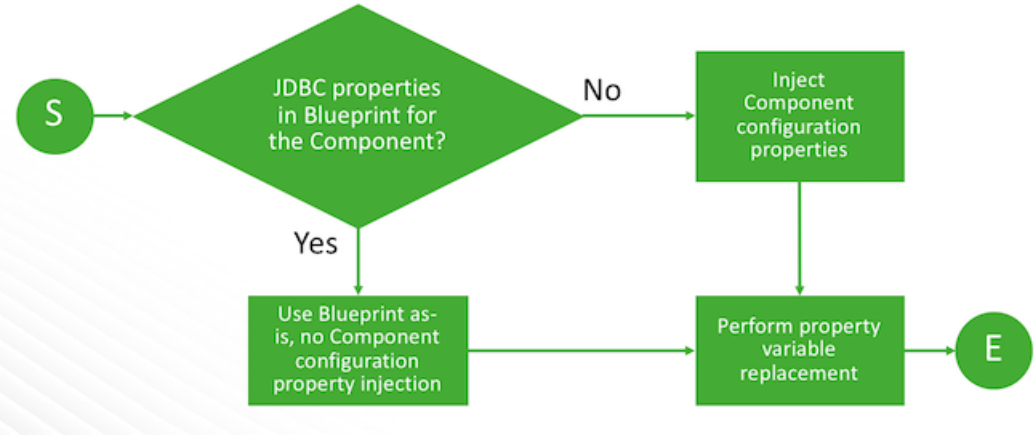
External database options

Review the following options to find out whether to use a built-in or other external database type.

When using an external database for cluster services and components, Cloudbreak supports selected built-in types and allows for specifying other types. Cloudbreak supports the following external database options:

Option	Description	Blueprint requirements	Steps	Example
Built-in types	Cloudbreak includes a few built-in types: Ambari, Beacon, Druid, Hive, Oozie, Ranger, and Superset.	Use a standard blueprint which does not include any JDBC parameters. Cloudbreak automatically injects the JDBC property variables into the blueprint.	Simply register the database in the UI. After that, you can attach the database config to your clusters.	Refer to Example 1
Other types	In addition to the built-in types, Cloudbreak allows you to specify custom types. In the UI, this corresponds to the UI option is called “Other” > “Enter the type”.	You must provide a custom dynamic blueprint which includes RDBMS-specific variables. Refer to Creating a template blueprint.	Prepare your custom blueprint first. Next, register the database in the UI. After that, you can attach the database config to your clusters.	Refer to Example 2

During cluster create, Cloudbreak checks whether the JDBC properties are present in the blueprint:



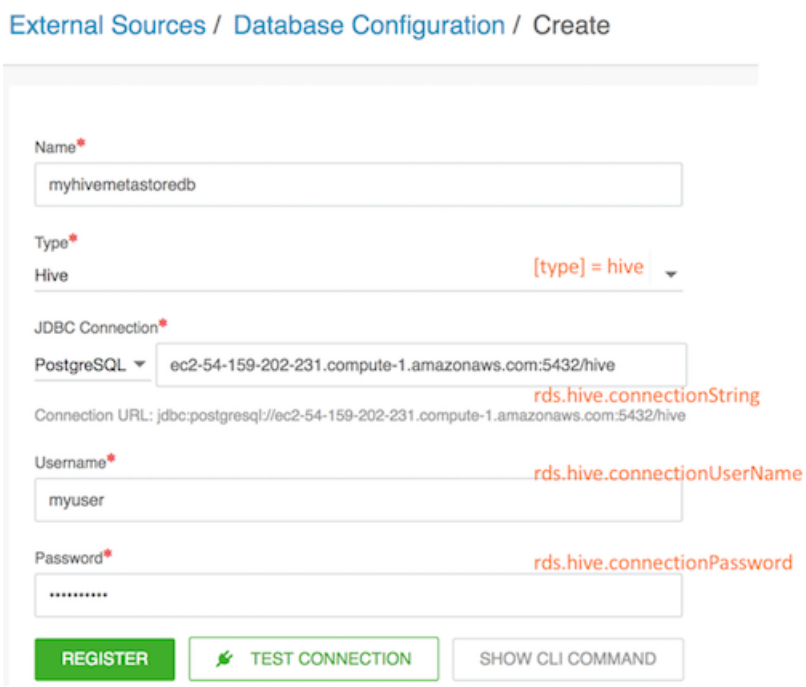
Related Information

Preparing the blueprint for LDAP/AD
Register an authentication source
Creating a template blueprint for RDMBS

Example 1: Built-in type Hive

In this scenario, you start up with a standard blueprint, and Cloudbreak injects the JDBC properties into the blueprint.

1. Register an existing external database of “Hive” type (built-in type):



Property variable	Example value
rds.hive.connectionString	jdbc:postgresql:// ec2-54-159-202-231.compute-1.amazonaws.com:5432/hive
rds.hive.connectionDriver	org.postgresql.Driver
rds.hive.connectionUserName	myuser
rds.hive.connectionPassword	Hadoop123!
rds.hive.fancyName	PostgreSQL
rds.hive.databaseType	postgres

2. Create a cluster by using a standard blueprint (i.e. one without JDBC related variables) and by attaching the external Hive database configuration.
3. Upon cluster create, Hive JDBC properties will be injected into the blueprint according to the following template:

```
...
"hive-site": {
"properties": {
  "javax.jdo.option.ConnectionURL": "{{{ rds.hive.connectionString }}}",
  "javax.jdo.option.ConnectionDriverName":
 "{{{ rds.hive.connectionDriver }}}",
  "javax.jdo.option.ConnectionUserName":
 "{{{ rds.hive.connectionUserName }}}",
  "javax.jdo.option.ConnectionPassword":
 "{{{ rds.hive.connectionPassword }}}"
}
  },
  "hive-env" : {
"properties" : {
  "hive_database" : "Existing {{{ rds.hive.fancyName}}} Database",
  "hive_database_type" : "{{{ rds.hive.databaseType }}}"
}
}
...
```

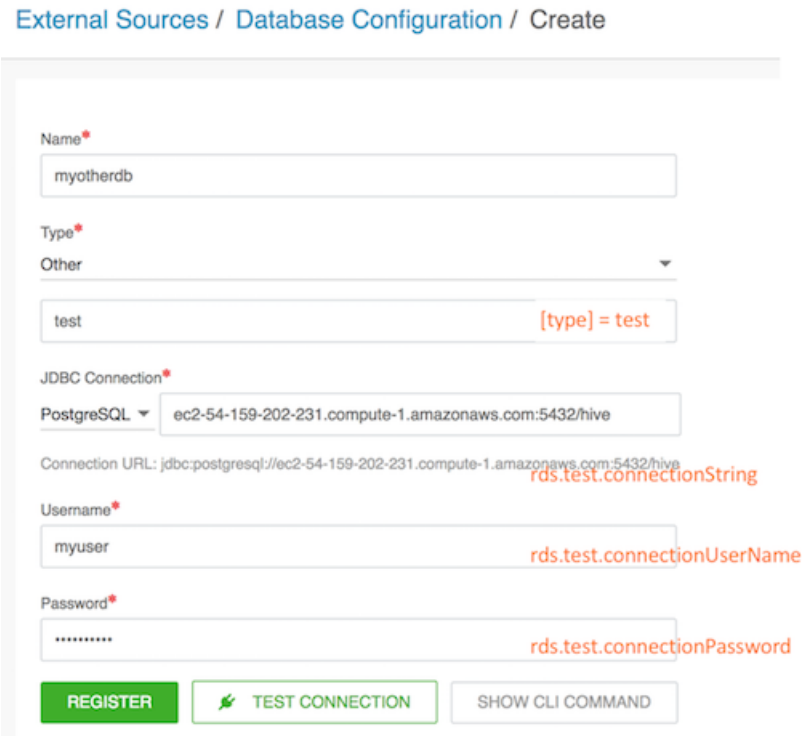
Example 2: Other type

In this scenario, you start up with a special blueprint including JDBC property variables, and Cloudbreak replaces JDBC-related property variables in the blueprint.

1. Prepare a blueprint blueprint that includes property variables. Use mustache template syntax. For example:

```
...
"test-site": {
  "properties": {
   "javax.jdo.option.ConnectionURL":"{{{rds.test.connectionString}}}"
  }
...
```

2. Register an existing external database of some “Other” type. For example:



a.

Property variable	Example value
rds.hive.connectionString	jdbc:postgresql:// ec2-54-159-202-231.compute-1.amazonaws.com:5432/hive
rds.hive.connectionDriver	org.postgresql.Driver
rds.hive.connectionUserName	myuser
rds.hive.connectionPassword	Hadoop123!
rds.hive.subprotocol	postgres
rds.hive.databaseEngine	POSTGRES

3. Create a cluster by using your custom blueprint and by attaching the external database configuration.
4. Upon cluster create, Cloudbreak replaces JDBC-related property variables in the blueprint.

Related Information

https://mustache.github.io/

Creating a template blueprint for RDMBS

In order to use an external RDBMS for some component other than the built-in components, you must include JDBC property variables in your blueprint.

See Example 2: Other type for an example configuration.

Related Information

Example 2: Other type

Register an external database

Create the external RDBMS instance and database and then register it with Cloudbreak.

Once you have the database instance running, you can:

1. Register it in Cloudbreak web UI or CLI.
2. Once registered, the database will now show up in the list of available databases when creating a cluster under advanced External Sources > Configure Databases. You can use it with one or more clusters.

Prerequisites

If you are planning to use an external MySQL or Oracle database, you must download the JDBC connector's JAR file and place it in a location available to the cluster host on which Ambari is installed. The steps below require that you provide the URL to the JDBC connector's JAR file.

Note:

If you are using your own custom image, you may place the JDBC connector's JAR file directly on the machine as part of the image burning process.

Steps

1. From the navigation pane, select External Sources > Database Configurations.
2. Select Register Database Configuration.
3. Provide the following information:

Parameter	Description
Name	Enter the name to use when registering this database to Cloudbreak. This is not the database name.
Type	Select the service for which you would like to use the external database. If you selected "Other", you must provide a special blueprint.
JDBC Connection	Select the database type and enter the JDBC connection string (HOST:PORT/DB_NAME).
Connector's JAR URL	(MySQL and Oracle only) Provide a URL to the JDBC connector's JAR file. The JAR file must be hosted in a location accessible to the cluster host on which Ambari is installed. At cluster creation time, Cloudbreak places the JAR file in the /opts/jdbc-drivers directory. You do not need to provide the "Connector's JAR URL if you are using a custom image and the JAR file was either manually placed on the VM as part of custom image burning or it was placed there by using a recipe.
Username	Enter the JDBC connection username.
Password	Enter the JDBC connection password.

4. Click Test Connection to validate and test the RDS connection information.

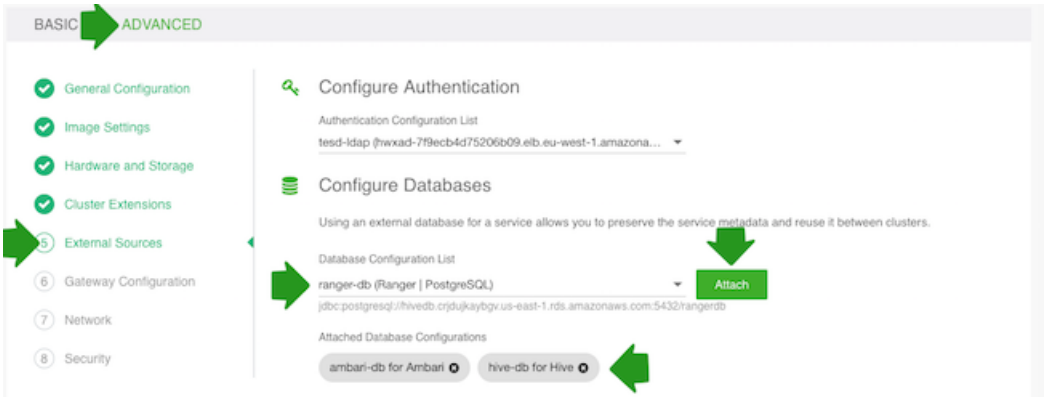
Note:

The Test Connection* option:

- Does not work when an external authentication source uses LDAPS with a self-signed certificate.
- Might not work if Cloudbreak instance cannot reach the LDAP server instance.

In these cases, ignore the error and proceed with cluster installation.

5. Once your settings are validated and working, click REGISTER to save the configuration.
6. The database will now show up on the list of available databases when creating a cluster under advanced External Sources > Configure Databases. You can select it and click Attach each time you would like to use it for a cluster:



Related Information

Creating a cluster

External authentication source for clusters

Cloudbreak allows you to register an existing LDAP/AD instance as an external source and use it for multiple clusters. Refer to this section if you would like to use LDAP for Cloudbreak-managed clusters.

You must create the LDAP/AD prior to registering it with Cloudbreak. Once you have it ready, the general steps are:

1. Prepare a cluster blueprint as described in the instructions for preparing a blueprint for LDAP/AD.
2. Register an existing LDAP in the Cloudbreak web UI or CLI.

Once registered, the LDAP will now show up in the list of available authentication sources when creating a cluster under advanced External Sources > Configure Authentication. Create a cluster by using the blueprint and by attaching the authentication source. Cloudbreak automatically injects the LDAP property variables into the blueprint.

Preparing the blueprint for LDAP/AD

In order to use LDAP/AD for your cluster, you must provide a suitable cluster blueprint.

The blueprint must fulfill the following requirements:

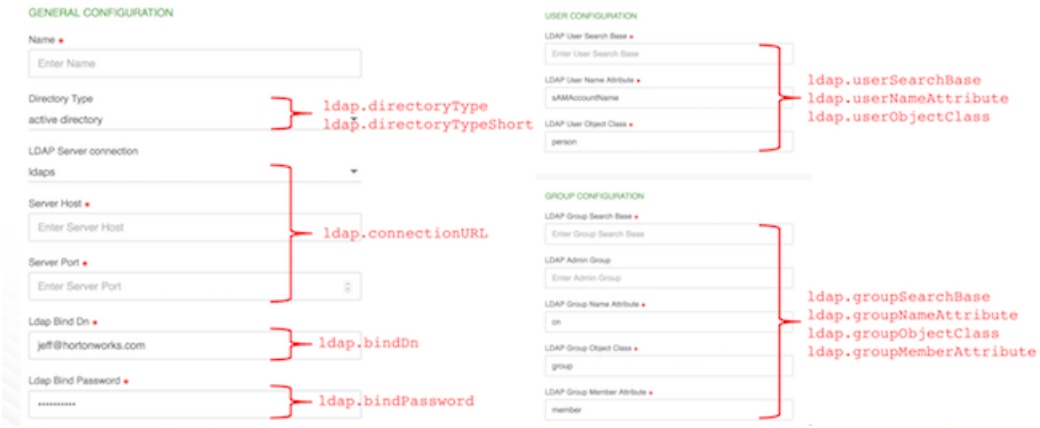
- The blueprint must include one or more of the following supported components: Atlas, Hadoop, Hive LLAP, Ranger Admin, Ranger UserSync.
- The blueprint should not include any LDAP properties. Before injecting the properties, Cloudbreak checks if LDAP related properties already exist in the blueprint. If they exist, they are not injected.

During cluster creation the following properties will be injected in the blueprint:

- ldap.connectionURL
- ldap.domain
- ldap.bindDn
- ldap.bindPassword
- ldap.userSearchBase

- ldap.userObjectClass
- ldap.userNameAttribute
- ldap.groupSearchBase
- ldap.groupObjectClass
- ldap.groupNameAttribute
- ldap.groupMemberAttribute
- ldap.directoryType
- ldap.directoryTypeShort

Their values will be the values that you provided to Cloudbreak:



Register an authentication source

Cloudbreak allows you to register an existing LDAP/AD instance and use it for multiple clusters. You must create the LDAP/AD prior to registering it with Cloudbreak.

Once you have it ready, you can:

1. Register an existing LDAP in Cloudbreak web UI or CLI.
2. Use it as an authentication source for your clusters. Once registered, the LDAP will now show up in the list of available authentication sources when creating a cluster under advanced External Sources > Configure Authentication.

Steps

1. From the navigation pane, select External Sources > Authentication Configurations.
2. Select Register Authentication Source.
3. Provide the following parameters related to your existing LDAP/AD:

   GENERAL CONFIGURATION

Parameter	Description	Example
Name	Enter a name for your LDAP.	cb-ldap
Directory Type	Choose whether your directory is LDAP or Active Directory.	LDAP
LDAP Server Connection	Select LDAP or LDAPS.	LDAP
Server Host	Enter the hostname or IP address for the LDAP or AD server.	10.0.3.128
Server Port	Enter the LDAP server port.	389
LDAP Domain	(Optional) Enter your LDAP domain if applicable.	ad.mytestldap.com

Parameter	Description	Example
LDAP Bind DN	Enter the LDAP Bind DN.	CN=Administrator,CN=Users,DC=ad,DC=hdc,DC=com
LDAP Bind Password	Enter the LDAP Bind DN password.	MyPassword1234!

USER CONFIGURATION

Parameter	Description	Example
LDAP User Search Base	Enter your LDAP user search base. This defines the location in the directory from which the LDAP search begins.	CN=Users,DC=ad,DC=hdc,DC=com
LDAP User Name Attribute	Enter the attribute for which to conduct a search on the user base.	HDCaccountName
LDAP User Dn Pattern	Enter LDAP User DN Pattern, which is used to bind an LDAP user.	CN={0},CN=Users,DC=ad,DC=hdc,DC=com
LDAP User Object Class	Enter the directory object class filter for users.	person

GROUP CONFIGURATION

Parameter	Description	Example
LDAP Group Search Base	Enter your LDAP group search base. This defines the location in the directory from which the LDAP search begins.	CN=Users,DC=ad,DC=hdc,DC=com
LDAP Admin Group	(Optional) Enter your LDAP admin group if applicable.	hdc
LDAP Group Name Attribute	Enter the attribute for which to conduct a search on groups.	cn
LDAP Group Object Class	Enter the directory object class filter for groups.	group
LDAP Group Member Attribute	Enter the attribute on the group object class that represents members.	member

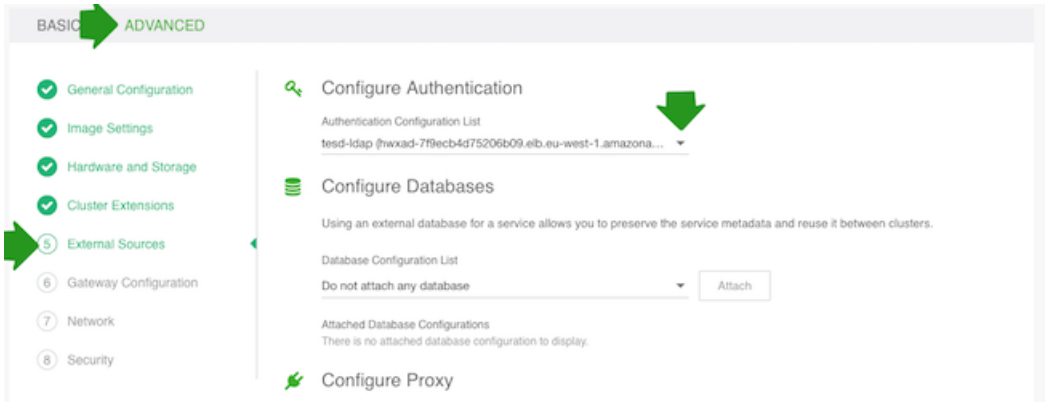
4. Click Test Connection to verify that the connection information that you entered is correct.

Note: The Test Connection option:

- Does not work when an external authentication source uses LDAPS with a self-signed certificate.
- Might not work if Cloudbreak instance cannot reach the LDAP server instance.

In these cases, ignore the error and proceed with cluster installation.

5. Click REGISTER.
6. The LDAP will now show up on the list of available authentication sources when creating a cluster under advanced External Sources > Configure Authentication. It can be reused with multiple clusters. Just select it if you would like to use it for a given cluster:



Related Information

Creating a cluster

Custom internal hostnames for cluster hosts

Cloudbreak maintains an internal DNS server and allows you to specify a custom hostname and a custom domain name.

Cloudbreak maintains an internal DNS server (using Unbound) with entries for all hosts in the cluster. In this setup, the internal DNS is configured to allow internal communication between the nodes of the same cluster. Queries that the internal DNS cannot resolve are automatically forwarded to the cloud provider's nameserver. The DNS server is distributed in the sense that each of the VMs runs one server with correct entries, so there is no single point of failure.

By default, the internal FQDNs are inherited from the cloud provider's DNS server. You can optionally change the hostname prefix and the domain by specifying them as parameters in your cluster CLI JSON. If you would like to specify a custom hostname and a custom domain name, add the following entries to the CLI JSON at the top level of the JSON, replacing the [$DOMAIN_NAME] and [$HOST_NAME_PATTERN] with actual values:

```
"customDomain": {
    "customDomain": "[$DOMAIN_NAME]",
    "customHostname": "[$HOST_NAME_PATTERN]"
  }
```

For example:

```
 "customDomain": {
    "customDomain": "hortonworks.local",
    "customHostname": "test"
  }
```

When these parameters are provided in the CLI JSON, VM FQDNs are generated based on the values specified. For the values included in the example, these FQDNs are "test0.hortonworks.local", "test1.hortonworks.local", "test2.hortonworks.local", and so on.

Custom hostnames based on DNS on AWS

By default, when Cloudbreak provisions cloud provider resources, your cloud provider assigns hostnames for your cluster nodes. Optionally, instead of using default hostnames, you can configure Cloudbreak to use custom hostnames based on DNS. To do that, follow the steps for configuring reverse Domain Name System (DNS) described below.

When the cluster node machines are provisioned, they try to make a reverse DNS lookup (by querying of the DNS to determine the domain name associated with a specific IP address); if the reverse DNS lookup returns a valid value, then that value is set as hostname. This is why reverse DNS setup is required for using custom hostnames.

On AWS you have the following two options:

- Use Route53 as DNS provider.
- Set up your own DNS server in your VPC

Both options require you to have an existing VPC and attach a custom DHCP options set to it.

Note:

The instructions provided in this section describe how to perform the steps in Amazon web consoles, but the steps can also be performed (and automated) in the AWS CLI.

Configure DNS using Route53

Follow these general steps to configure reverse DNS using Route53.

Step 1: Create a new VPC or use your existing VPC

1. You can create a new VPC from the Amazon VCP console (for example by using Start VPC Wizard):
   - CIDR block example: 10.1.0.0/16
   - Subnet's CIDR example: 10.1.1.0/28
2. Make sure to:
   - Enable DNS resolution for the VPC. You can do this by selecting the VPC, selecting Actions > Edit DNS resolution and choosing Yes.
   - Enable DNS hostnames for the VPC. You can do this by selecting the VPC, selecting Actions > Edit DNS hostnames and choosing Yes.

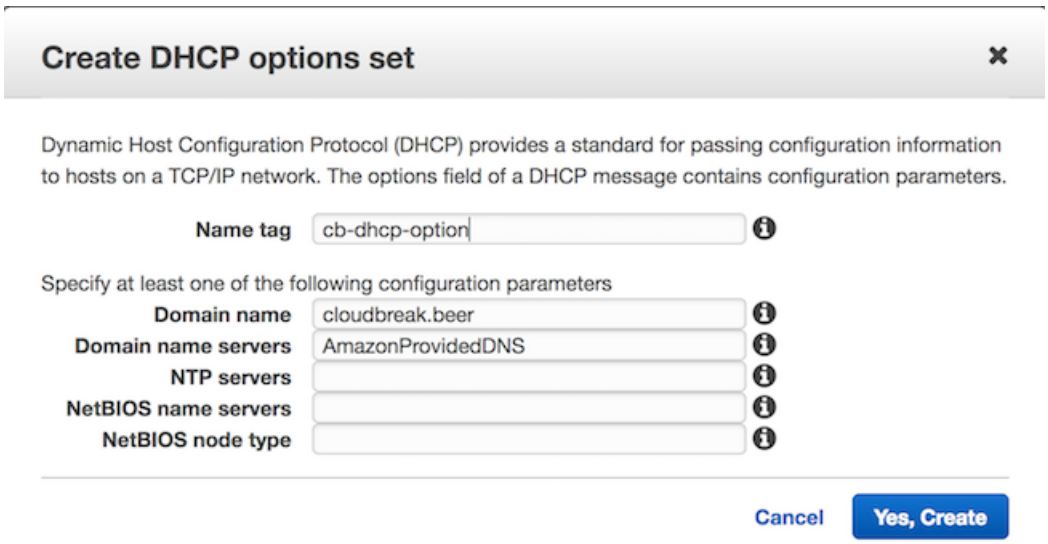
Note:

Optionally, you may want to set up an Internet Gateway for the VPC and add a default route to the routing table for the Internet Gateway. Additionally, you may want to enable the Auto-assign Public IP option. This way Cloudbreak would reach the cluster from outside of the VPC and the cluster would have internet access.

Step 2: Create a DHCP options set:

Perform this step from the Amazon VPC console. Select DHCP Options Sets from the left pane and click on Create a DHCP options set. Make sure to:

- Set the Domain name to a preferred domain, for example cloudbreak.beer
- Set the Domain name servers to AmazonProvidedDNS



For detailed steps, refer to AWS documentation.

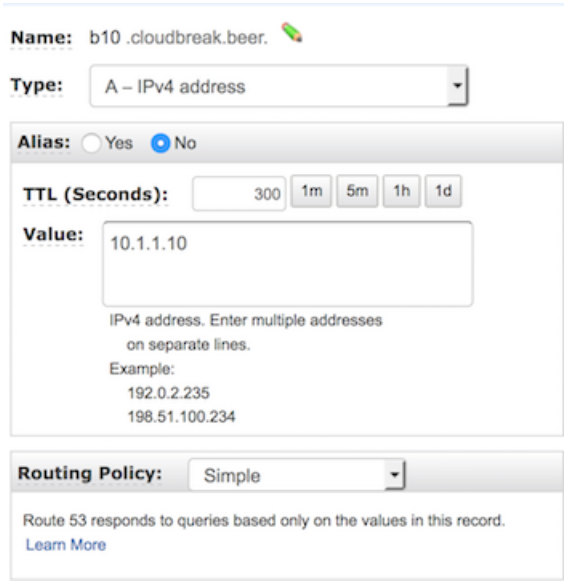
Step 3: Assign the newly created DHCP options set to your VPC

1. From the Amazon VPC console, select Your VPCs from the left pane.
2. Select the VPC created earlier.
3. Click on Actions > Edit DHCP Options Set.
4. Select the newly created DHCP option set.

Step 4: Configure your domain at Route53

Perform these steps from the Amazon Route53 console. For general steps, refer to AWS documentation.

1. Select Hosted zones from the left pane.
2. Create a hosted zone by clicking on Create Hosted Zone. Make sure to:
   - Use the same domain name as used previously with the DHCP options set (In the example this was cloudbreak.beer).
   - Set the Type to Private Hosted Zone for Amazon VPC.
   - Select the VPC ID of the VPC to which you previously assigned the DHCP option.
3. Add records for your hosted zone:
   - Select the hosted zone and choose Go to Record Sets
   - Click Create Record Set to create a record set. You must perform this step for every available IP, so that each IP can have a custom name (If you used the subnet example listed above, these IPs will be in the range of 10.1.1.4-14):
     - Type: select A
     - Name: for example b10
     - Value: for example 10.1.1.10



4. After performing this step for each IP, you should end up with an many records as IPs. For example:

Name	Type	Value
cloudbreak.beer.	NS	ns-1536.awsdns-00.co.uk. ns-0.awsdns-00.com. ns-1024.awsdns-00.org. ns-512.awsdns-00.net.
cloudbreak.beer.	SOA	ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
b10.cloudbreak.beer.	A	10.1.1.10
b11.cloudbreak.beer.	A	10.1.1.11
b12.cloudbreak.beer.	A	10.1.1.12
b13.cloudbreak.beer.	A	10.1.1.13
b14.cloudbreak.beer.	A	10.1.1.14
b4.cloudbreak.beer.	A	10.1.1.4
b5.cloudbreak.beer.	A	10.1.1.5
b6.cloudbreak.beer.	A	10.1.1.6
b7.cloudbreak.beer.	A	10.1.1.7
b8.cloudbreak.beer.	A	10.1.1.8
b9.cloudbreak.beer.	A	10.1.1.9

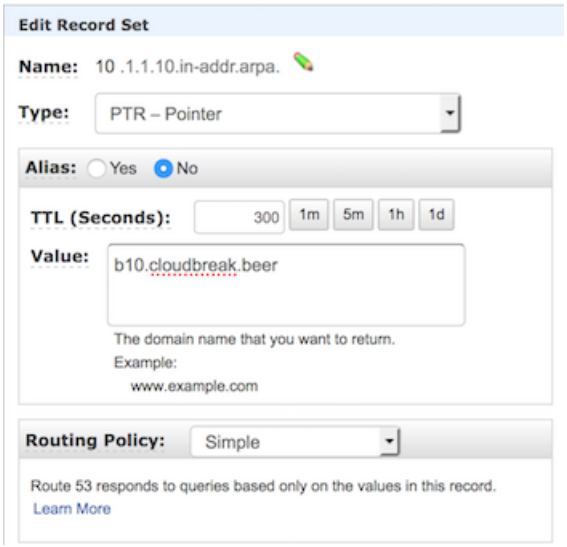
Step 5: Create another hosted zone for reverse DNS lookup

Perform these steps from the Amazon Route53 console.

1. Select Hosted zones from the left pane.
2. Create a hosted zone by clicking on Create Hosted Zone. Make sure to:
   - For example, if you used the subnet example listed above, its Domain name should look like this (as reverse DNS lookups use the special domain in-addr.arpa):

     ```
     1.1.10.in-addr.arpa.
     ```

   - Set the Type to Private Hosted Zone for Amazon VPC.
   - Select the VPC ID to which you previously assigned the DHCP option set.
3. Add records for every created domain:
   - Type: select PTR
   - Name: This determines the first part of the IP, for example 10
   - Value: Enter the domain name that you set in the previous step, for example, b10



4. After performing this step for each domain, you should end up with as many records as IPs. For example:

Name	Type	Value
1.1.10.in-addr.arpa.	NS	ns-1536.awsdns-00.co.uk. ns-0.awsdns-00.com. ns-1024.awsdns-00.org. ns-512.awsdns-00.net.
1.1.10.in-addr.arpa.	SOA	ns-1536.awsdns-00.co.uk. awsdns-hostmaster.amazon.com. 1 7200 900 1209600 86400
10.1.1.10.in-addr.arpa.	PTR	b10.cloudbreak.beer
11.1.1.10.in-addr.arpa.	PTR	b11.cloudbreak.beer
12.1.1.10.in-addr.arpa.	PTR	b12.cloudbreak.beer
13.1.1.10.in-addr.arpa.	PTR	b13.cloudbreak.beer
14.1.1.10.in-addr.arpa.	PTR	b14.cloudbreak.beer
4.1.1.10.in-addr.arpa.	PTR	b4.cloudbreak.beer
5.1.1.10.in-addr.arpa.	PTR	b5.cloudbreak.beer
6.1.1.10.in-addr.arpa.	PTR	b6.cloudbreak.beer
7.1.1.10.in-addr.arpa.	PTR	b7.cloudbreak.beer
8.1.1.10.in-addr.arpa.	PTR	b8.cloudbreak.beer
9.1.1.10.in-addr.arpa.	PTR	b9.cloudbreak.beer

Step 6: Create the cluster in the VPC configured in the earlier steps and you will have the same hostnames set as the domain names.

Note:

Since you don't have control the order over the IP addresses leased to the machines, the names may not be in order.

Configure DNS using custom DNS server

Follow these general steps to configure reverse DNS using a custom DNS server.

Step 1: Create a new VPC or use your existing VPC

1. You can create a new VPC from the Amazon VCP console (for example by using Start VPC Wizard):
   - CIDR block example: 10.1.0.0/16
   - Subnet's CIDR example: 10.1.1.0/28
2. Make sure to:

- Enable DNS resolution for the VPC. You can do this by selecting the VPC, selecting Actions > Edit DNS resolution and choosing Yes.
- Enable DNS hostnames for the VPC. You can do this by selecting the VPC, selecting Actions > Edit DNS hostnames and choosing Yes.

Note:

Optionally, you may want to set up an Internet Gateway for the VPC and add a default route to the routing table for the Internet Gateway. Additionally, you may want to enable the Auto-assign Public IP option. This way Cloudbreak would reach the cluster from outside of the VPC and the cluster would have internet access.

Step2: Set up DNS server in your VPC/subnet

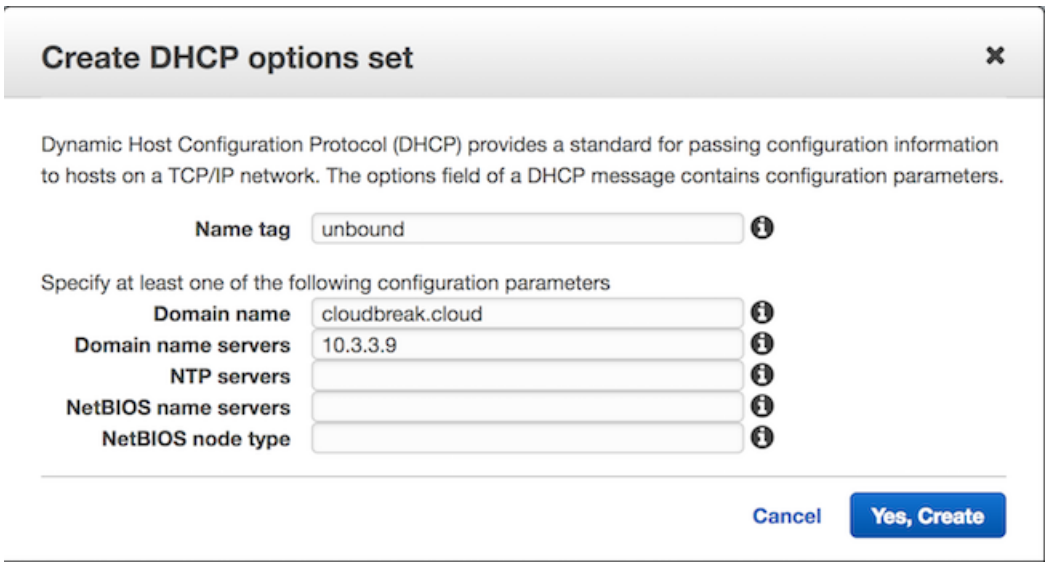
- In the configuration ensure that you have DNS records and reverse DNS pointers for all IP address (for example 10.3.3.4-14)
- Example unbound configuration:

```
[root@ip-10-3-3-9 conf.d]# cat 00-cloudbreak.cloud.conf
   server:
      local-zone: "cloudbreak.cloud." static
      local-data: "aww1.cloudbreak.cloud. IN A 10.3.3.4"
      local-data-ptr: "10.3.3.4 aww1.cloudbreak.cloud."
      local-data: "aww2.cloudbreak.cloud. IN A 10.3.3.5"
      local-data-ptr: "10.3.3.5 aww2.cloudbreak.cloud."
      local-data: "aww3.cloudbreak.cloud. IN A 10.3.3.6"
      local-data-ptr: "10.3.3.6 aww3.cloudbreak.cloud."
      local-data: "aww4.cloudbreak.cloud. IN A 10.3.3.7"
      local-data-ptr: "10.3.3.7 aww4.cloudbreak.cloud."
      local-data: "aww5.cloudbreak.cloud. IN A 10.3.3.8"
      local-data-ptr: "10.3.3.8 aww5.cloudbreak.cloud."
      local-data: "aww6.cloudbreak.cloud. IN A 10.3.3.9"
      local-data-ptr: "10.3.3.9 aww6.cloudbreak.cloud."
      local-data: "aww7.cloudbreak.cloud. IN A 10.3.3.10"
      local-data-ptr: "10.3.3.10 aww7.cloudbreak.cloud."
      local-data: "aww8.cloudbreak.cloud. IN A 10.3.3.11"
      local-data-ptr: "10.3.3.11 aww8.cloudbreak.cloud."
      local-data: "aww9.cloudbreak.cloud. IN A 10.3.3.12"
      local-data-ptr: "10.3.3.12 aww9.cloudbreak.cloud."
      local-data: "aww10.cloudbreak.cloud. IN A 10.3.3.13"
      local-data-ptr: "10.3.3.13 aww10.cloudbreak.cloud."
      local-data: "aww11.cloudbreak.cloud. IN A 10.3.3.14"
      local-data-ptr: "10.3.3.14 aww11.cloudbreak.cloud."
```

Step 3: Create a DHCP options set

Perform this step from the Amazon VPC console. Select DHCP Options Sets from the left pane and click on Create a DHCP options set. Make sure to:

- Set the Domain name to your preferred domain, for example cloudbreak.cloud
- Set Domain name servers to the previously created DNS server
- Optionally, set a Name tag



For detailed steps, refer to AWS documentation.

Step 4: Assign the newly created DHCP options set to your VPC

1. From the Amazon VPC console, select Your VPCs from the left pane.
2. Select the VPC created earlier.
3. Click on Actions > Edit DHCP Options Set.
4. Select the newly created DHCP option set.

Step 5: Create the cluster in the VPC configured in the preceding steps and you will have the same hostnames set as the domain names.

Note:

Since you don't have control the order over the IP addresses leased to the machines, the names may not be in order.

Related Information

Create a DHCP Options Set