EDB Ark™

Getting Started Guide

Version 2.0

September 13, 2016

EDB Ark Getting Started Guide, Version 2.0
by EnterpriseDB® Corporation

EnterpriseDB Corporation, 34 Crosby Drive Suite 100, Bedford, MA 01730, USA
T +1 781 357 3390 **F** +1 978 589 5701 **E** info@enterprisedb.com **www**.enterprisedb.com

Table of Contents

1 Introduction ... 5
1.1 What's New ... 7
1.2 Typographical Conventions Used in this Guide ... 7
2 EDB Ark - Overview ... 8
2.1 The Benefits of using EDB Ark ... 8
2.2 Architecture Overview ... 10
2.3 EDB Ark ... 13
3 Using the Ark console ... 14
3.1 The Dashboard Tab ... 17
3.2 The Clusters Tab ... 18
3.2.1 The Details Panel ... 22
3.2.2 The Monitoring Panel ... 26
3.2.3 The Events Panel ... 27
3.3 The Backups Tab ... 28
3.4 The User Tab ... 29
4 Creating a Server Cluster ... 31
4.1 Creating a New Server Cluster ... 32
4.1.1 Perform OS and Software Update ... 36
4.2 Creating a Cluster that Enables Point-In-Time Recovery ... 38
4.3 Creating a Developer Sandbox ... 40
4.4 Modifying a Cluster's Administrative Settings ... 42
5 Connecting an Application to an EDB Ark Cluster ... 44
5.1 Using iptables Rules ... 46
6 Managing Backups and Recovery ... 47
6.1 Performing a Base Backup for Point-In-Time Recovery ... 48
6.2 Reviewing Stored Backups ... 49
6.3 Restoring a Cluster from Backup ... 51
7 Automatic Failover ... 53
8 Manual Scaling ... 55
8.1 Manually Adding Replicas and Storage ... 55
8.2 Manually Removing a Replica ... 58
8.3 Manually Changing the Server Class ... 59

9 Automatic Scaling ... 62
9.1 Adjusting the Automatic Scaling Thresholds ... 62
10 Load Balancing ... 64
11 Customizing Your Cluster ... 67
11.1 Adding an Extension to a New Cluster ... 68
12 Database Management ... 69
12.1 Connecting to the Cluster ... 70
12.1.1 Using ssh to Access a Server ... 70
12.1.2 Connecting with the Postgres Enterprise Manager Client ... 72
12.1.3 Connecting to EDB Ark with psql or edb-psql ... 75
12.2 Moving an Existing Database into a New Cluster ... 79
12.3 Manually Modifying Configuration Files ... 85
12.3.1 Best Practices for Modifying Configuration Files ... 86
12.4 Controlling the Server ... 87
12.4.1 Controlling a Service on CentOS or RHEL 7.x ... 87
12.4.2 Controlling a Service on CentOS or RHEL 6.x ... 88
12.4.3 Using the pg_ctl Utility ... 89
12.5 Updating Packages on the EDB Ark Cluster ... 91
12.5.1 Performing a Major Version Upgrade ... 93
13 Troubleshooting ... 94
13.1 Frequently Asked Questions ... 95
13.2 The EDB Ark Email Notification System ... 97
14 EDB Ark API Support ... 98
14.1 Resources ... 98
14.1.1 /admin/logs ... 99
14.1.2 /admin/wall ... 99
14.1.3 /clusters ... 100
14.1.4 /dbengines ... 101
14.1.5 /dbengines/*engine_id* ... 103
14.1.6 /options/backup-windows ... 104
14.1.7 /options/ip-pools/*tenant_name* ... 104
14.1.8 /options/server-classes/*tenant_name*/?engineId=*id* ... 105
14.1.9 /options/types ... 106
14.1.10 /options/versions/*type* ... 106

14.1.11 /options/vpcids/*tenant_name* 106
14.1.12 /owners 107
14.1.13 /owners/*tenant_name*/backups 107
14.1.14 /owners/*tenant_name*/backups/*backup_id* 109
14.1.15 /owners/*tenant_name*/clusters 111
14.1.16 /owners/*tenant_name*/clusters/*cluster_name* 113
14.1.17 /owners/*tenant_name*/clusters/*cluster_name*/events 117
14.1.18 /owners/*tenant_name*/clusters/*cluster_name*/statistics?start=*start_time*&end=*end_time* 118
14.1.19 /serverimages 119
14.1.20 /serverimages/*image_id* 119
14.1.21 /tokens 120
14.1.22 /users 121
14.1.23 /users/*user_id* 122
14.1.24 /users/*user_id*/notifications 123
14.2 Response Codes 124

1 Introduction

EDB Ark automatically provisions PostgreSQL or EDB Postgres Advanced Server databases in single instances, high-availability clusters, or application development sandboxes. EDB Ark allows service providers and organizations to offer elastic and highly scalable database-as-a-service (DBaaS) environments while freeing DBAs and application developers from the rigors of setting up and administering modern and robust database environments.

In minutes, EDB Ark configures a cluster of database machines with:

- Streaming replication
- Connection pooling
- Load balancing
- Automatic failover (transaction or recovery time preferred)
- Secure data encryption
- Rotating user-scheduled backups
- Point-in-time recovery
- Elastic storage
- Elastic scale out

EDB Ark's automatic scaling of storage resources and scale out of read replicas when a database cluster reaches user-defined thresholds is especially worth noting - this functionality provides unattended, around-the-clock responsiveness to unpredictable load demands on your database infrastructure.

This document will demonstrate how to use the EDB Ark interface successfully in your cloud-based database management activities:

- **EDB Ark - Overview** – Section 2 provides information about EDB Ark functionality and architecture.

- **Using the Ark Console** – Section 3 introduces you to the EDB Ark graphical user interface, and provides an overview of the functionality offered by the user interface controls.

- **Creating a New Server Cluster –** Section 4 walks you through how to create a server cluster, and how to create a developer sandbox.

- **Connecting an Application** – Section 5 describes how to locate connection information for your server nodes, so your client applications can access your cluster.

- **Managing Backups and Recovery -** Section 6 describes how to backup or restore a database hosted on EDB Ark.

- **Automatic Failover** – Section 7 discusses EDB Ark failover functionality.

- **Manual Scaling** – Section 8 describes how to manually scale up your database cluster by adding replica nodes or memory.

- **Automatic Scaling –** Section 9 discusses how to set the automatic scale up thresholds for your database.

- **Load Balancing –** Section 10 discusses how to use load balancing to optimize client performance.

- **Customizing Your Cluster -** Section 11 discusses some of the ways you can customize your Ark cluster.

- **Database Management** – Section 12 provides information about performing administrative tasks on an Ark cluster.

- **Troubleshooting –** Section 13 provides helpful troubleshooting resources, and detailed information about how to recover from a console failure.

- **API Reference –** Section 14 provides reference information about EDB Ark's JSON compatible API.

This document provides an introduction to EDB Ark and is written to acquaint you with the process of configuring and using the product's core features; it is not a comprehensive guide to using EnterpriseDB database products. Depending on your operating environment, there may be differences in EDB Ark features and functions.

For more information about using EnterpriseDB products, please visit the EnterpriseDB website at:

http://www.enterprisedb.com/documentation

This document uses *Postgres* to mean either the PostgreSQL or EDB Postgres Advanced Server database.

1.1 What's New

The following features have been added to EDB Ark for release 2.0:

- The Ark console can provision and manage clusters that reside in an OpenStack environment.
- The Ark console enables yum-based provisioning of RPM packages across all nodes of a cluster.
- EDB Ark supports a JSON-compatible API.

1.2 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the proceeding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

2 EDB Ark - Overview

EDB Ark simplifies the process of provisioning robust Postgres deployments, while taking advantage of the benefits of cloud computing. When used with Advanced Server, EDB Ark also provides a platform with compatibility with the Oracle database, offering dramatic cost savings and competitive advantages.

2.1 The Benefits of using EDB Ark

EDB Ark solves common challenges faced by businesses that need more agility, velocity, and thrift in deploying and using relational, ACID-compliant databases:

- **Develop / Test / Deploy.** Quickly set up and take down Postgres databases with standard configurations to support software development and testing activities, then deploy applications to the database or cluster – all at a pace dramatically quicker than physical provisioning.

- **Workload Portability.** The same Postgres database trusted in the datacenter also runs in a cloud cluster with scalability and high-availability.

- **Enterprise-class power.** Postgres was designed to solve critical business challenges requiring reliable, high-performance, ACID-compliant database processing. As the only open source database meeting those requirements, it offers an extremely attractive alternative to more expensive options.

EDB Ark includes the following functionality:

- **Scale computing resources up and out.** EDB Ark automatically scales up storage capacity, and provides a simple button to scale your server class up when data processing loads and usage characteristics require a change in the underlying virtual machine resources.

- **Automatic Connection Pooling and Load Balancing.** EDB Ark maintains a cluster of database nodes, automatically scaling out replicas based on increasing user demand. The integrated connection pooling manager increases database read performance by distributing requests across all cluster members.

- **Self-Healing Failover.** EDB Ark automatically replaces downed database nodes, preserving the continuity and performance of the cluster. Users can choose to replace the master with a new master (preserving all committed transactions) or with a promoted replica (for faster recovery time).

- **Automatic Online backup.** EDB Ark uses user-directed rotating backups to protect your data from loss due to mishaps

- **Supports data encryption.** EDB Ark offers data encryption that is easy to activate, protects data at rest, and is transparent to connecting clients.

- **Cost-saving Compatibility with the Oracle Database.** Using a database that is compatible with Oracle is a reliable, fast and cost-effective way to support Oracle applications in public and private clouds.

- **Web-based interface.** EDB Ark provides easy to use point-and-click cluster lifecycle management from start to finish from your favorite web browser.

- **Database Cloning.** EDB Ark allows you to quickly and easily create developer 'sandboxes' based on real production data, saving System Administrators setup, configuration and data load time.

- **Professional Postgres Support.** EnterpriseDB provides support from Postgres experts who work with top Postgres open source developers.

- **JSON Compatible API Support.** EDB Ark supports a JSON-compatible API.

2.2 Architecture Overview

The Ark console is designed to help you easily create and manage high-availability database clusters from a web browser.

Traditionally, the expression *cluster* refers to a single instance of Postgres managing multiple databases; an EDB Ark *database server cluster* is a collection of high-availability Postgres server instances that reside in a cloud or on a traditional network.

When you create a new cluster (a group of replicated database servers), EDB Ark initializes one or more Postgres instances (virtual machines) according to your specifications. EDB Ark uses Postgres streaming replication to synchronize replicas in the cluster, and pgpool-II to implement load balancing and connection pooling among all active instances. Figure 2.1 provides a general overview of the EDB Ark architecture.

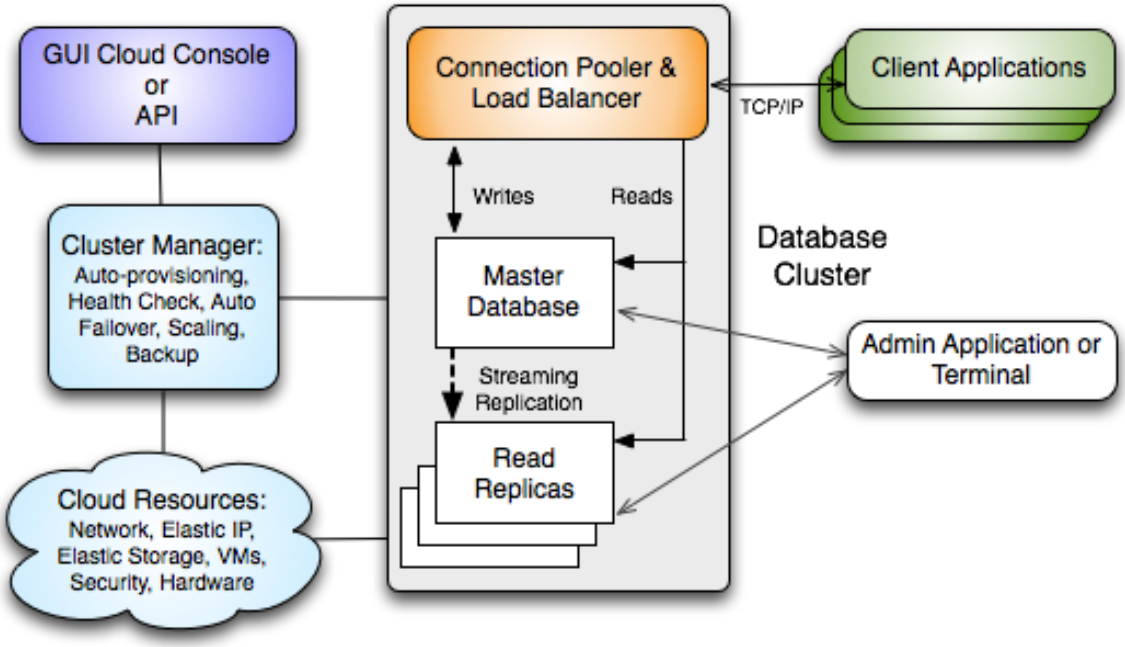


Figure 2.1 - An overview of the EDB Ark architecture.

The master node of the cluster contains a host operating system with a running instance of Postgres, along with the load balancer. Database modifications are automatically routed to the master node; any modifications to the master node are subsequently propagated to each replica using Postgres streaming replication.

EDB Ark installs Postgres on each replica node in a read-only hot-standby role that automatically duplicates all data found on the master node, and all changes made to that data. In hot-standby mode, the data is available to service user queries providing read scalability to the cluster (see Figure 2.2). In addition, any schema changes made to the

master are also replicated to the replica nodes, making development and deployment of application changes easy and seamless without interruption to normal operations.

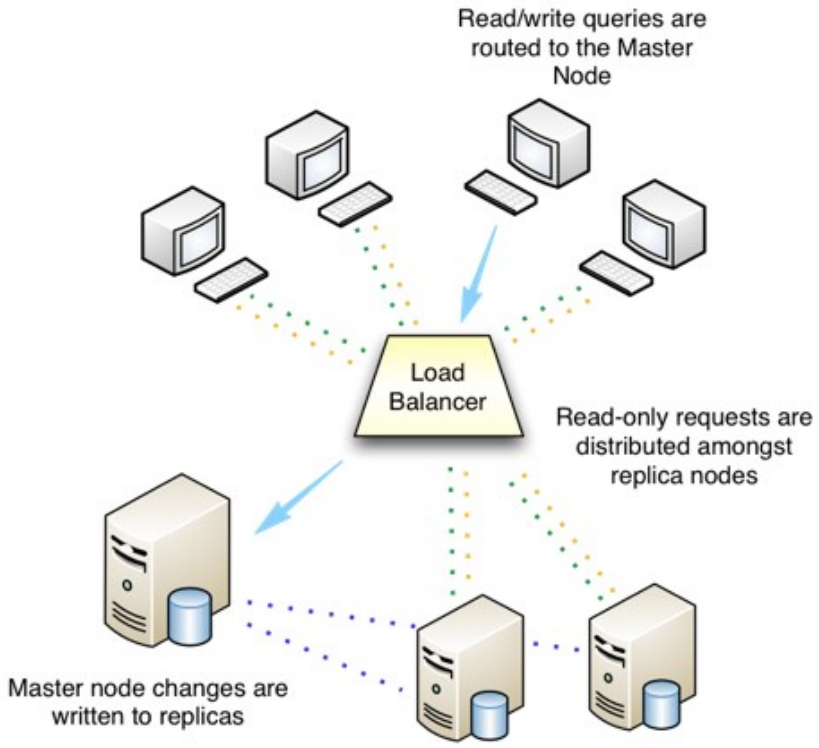


Figure 2.2 - EDB Ark performs automatic load balancing.

Replicas provide balanced user support as needed - if any instance in the cluster goes offline, the cluster's load is re-balanced among the remaining servers while the instance is automatically replaced.

When used in the default healing configuration, in the event of a failure of the master node, a new node is spun up and the disk-storage for the old master node is attached to it, providing zero data loss when the new master becomes available. While the replica nodes are standing by, they are read-only resources, load balancing client queries without a risk of compromising data integrity. Optionally, users can choose to promote an existing replica to master status when healing is required, providing a faster recovery time at the expense of losing any transactions committed on the master, but not yet replicated when the master went down.

EDB Ark automatically archives data at regular intervals; you can specify a convenient backup window and how many backups to retain when creating a database cluster. EDB Ark also offers backup on demand - simply click the `Backup` icon to save a copy of the instance. Automatic backups are retained according to your specifications; on-demand backups are retained until you delete them. Each backup is a complete copy of the cluster; you can use a backup to restore a cluster.

EDB Ark makes it easy to *scale* a database cluster:

- To increase read performance, you can add read replicas to the cluster (manually or automatically).
- To handle expanding data requirements you can increase the amount of storage available (manually or automatically).
- To increase the RAM or CPU processing power of the cluster's underlying virtual machine, you can manually scale a cluster into a more appropriate server class.

2.3 EDB Ark

A cloud (shown in Figure 2.3) is a collection of virtual machines; each virtual machine runs a separate copy of an operating system and an installation of Postgres.

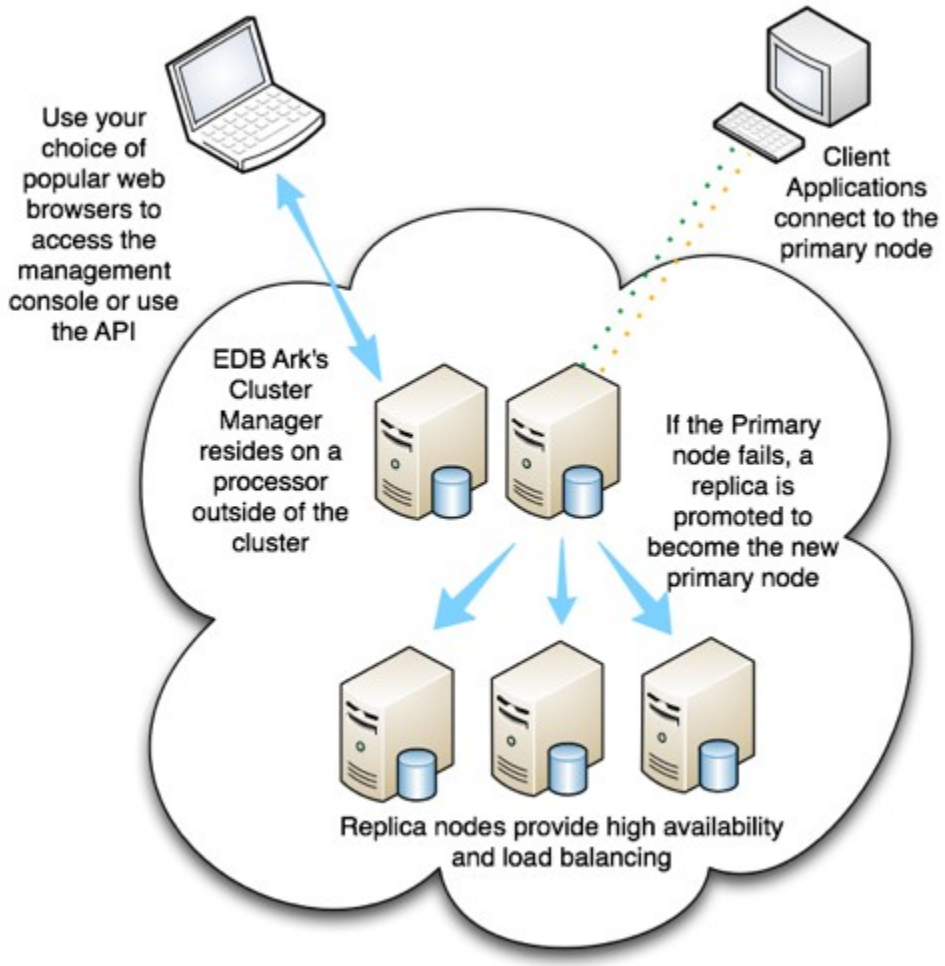


Figure 2.3 - Using EDB Ark in a Private Cloud.

You can specify different combinations of CPU speed, RAM, and disk space to suit your needs when provisioning an EDB Ark cluster. EDB Ark makes it easy to scale up to a more capable cluster, or scale down as your requirements change.

3 Using the Ark console

Use the IP address provided by your system administrator in your web browser of choice to access the Ark console. When you navigate to the address, the Ark console will prompt you to `Log in` (as shown in Figure 3.1).

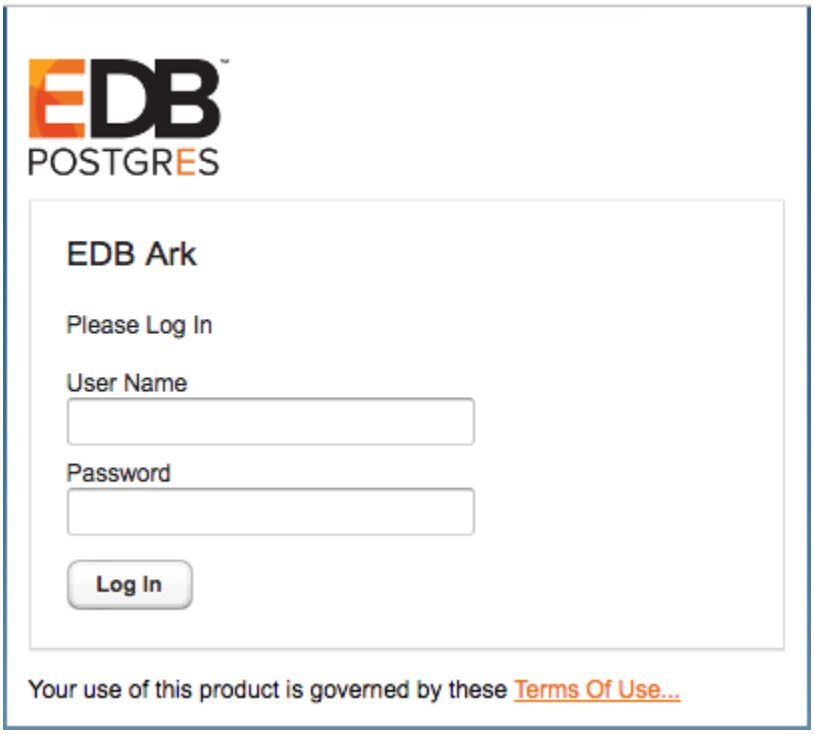


Figure 3.1 - The EDB Ark Log in dialog.

Use your OpenStack credentials to connect to the Ark console. When connecting for the first time, the `User` tab will open as shown in Figure 3.2.

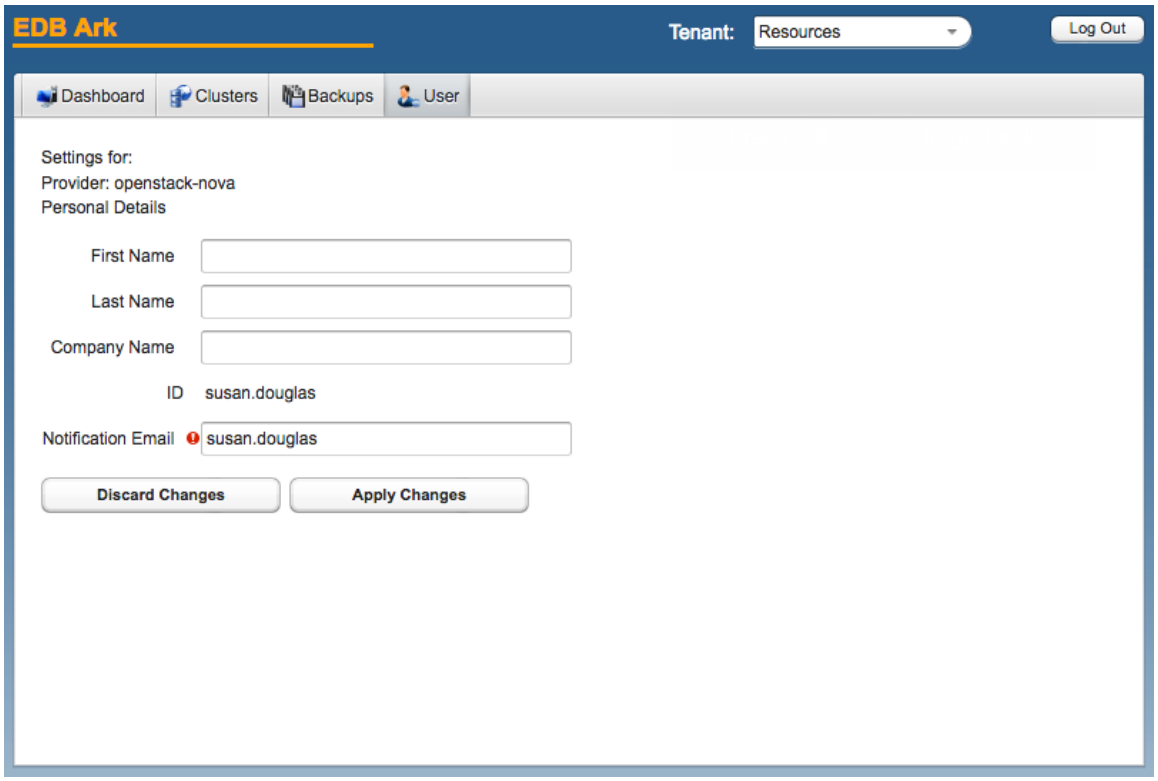


Figure 3.2 - The EDB Ark User tab.

Use the fields on the `User` tab to provide user information:

- Use the `First Name` field to specify the first name of the connecting user.
- Use the `Last Name` field to specify the last name of the connecting user.
- Use the `Company Name` field to specify the company name.
- The `ID` field displays the OpenStack identifier of the connected user.
- Use the `Notification Email` field to specify the default email address that will be used for cluster notifications unless an alternate address is provided.

  You can optionally provide an alternate email address when a cluster is created (on the `Create a new Server Cluster` dialog), or modify a cluster's notification email address on the `Administrative Settings` dialog.

After providing user information, click `Apply Changes` to save the information before exiting the `User` tab. A popup will prompt you to enter your user password to confirm the changes to the `Notification Email` field (see Figure 3.3).

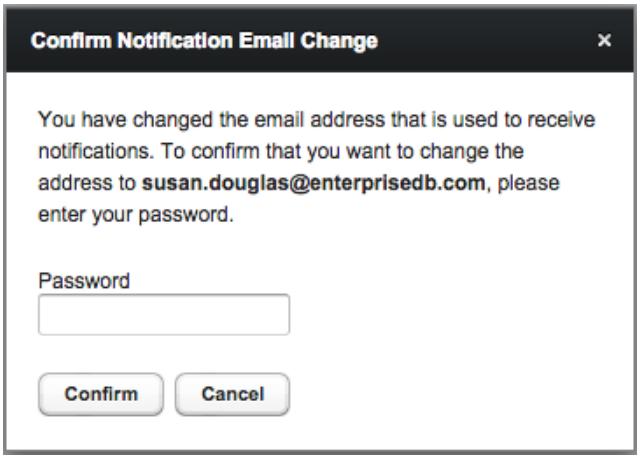


Figure 3.3 – Enter a password to confirm an email change.

Enter your password and select `Confirm` to continue. After providing user information, you can navigate to either the `Dashboard` tab (shown in Figure 3.4) or the `Clusters` tab (see Section 3.2) to launch a cluster.

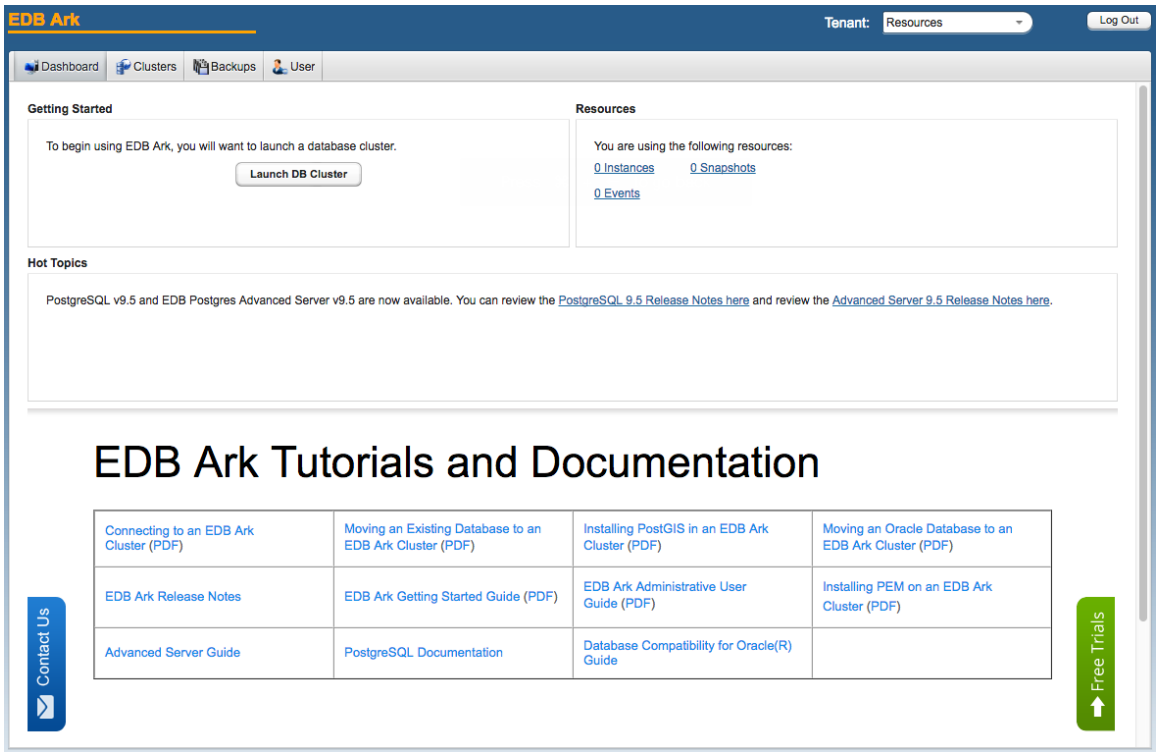


Figure 3.4 - The Dashboard tab of the Ark console.

3.1 The Dashboard Tab

The `Dashboard` tab (shown in Figure 3.5) provides an overview of the EDB Ark service status, resources, useful information links and a quick-start `Launch DB Cluster` button.

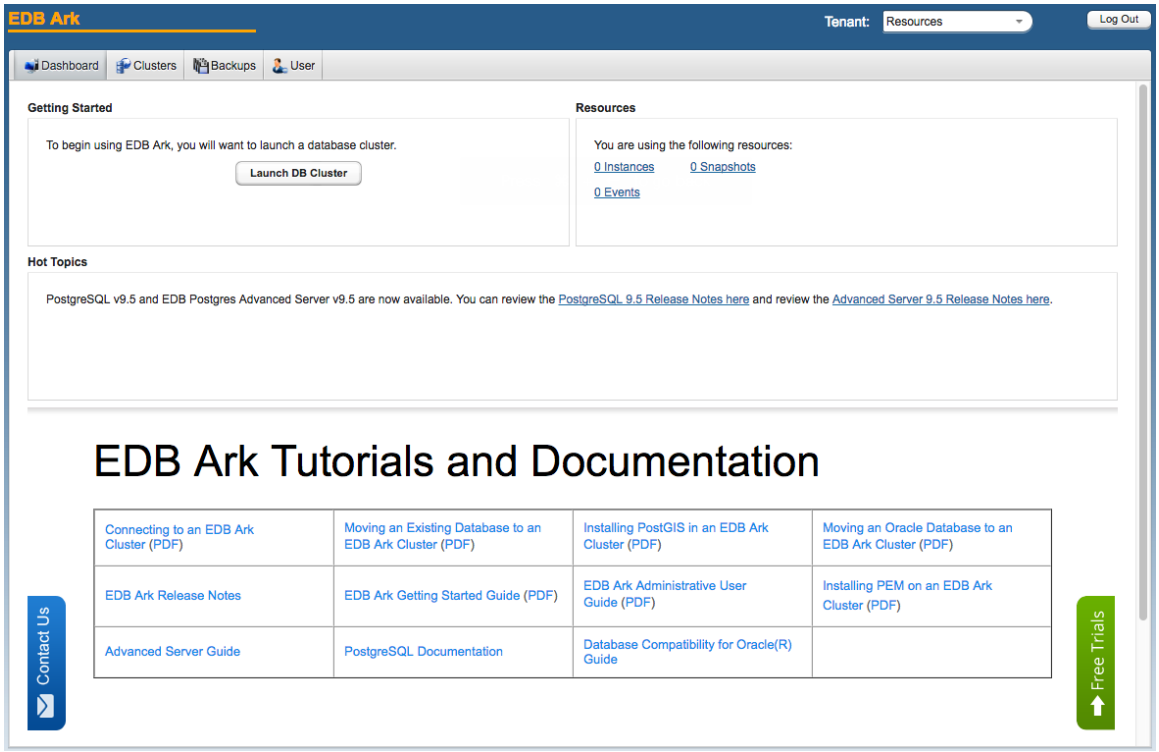


Figure 3.5 - The Dashboard tab.

To launch a cluster from the `Dashboard` tab, use the `Tenant` drop-down listbox to select the tenant in which the cluster will be created. Then, use the `Launch DB Cluster` button located in the `Getting Started` panel to open the `Create New Cluster` dialog and define the cluster attributes. For more information about defining a cluster, see Section 4.

The `Resources` panel contains an overview of the activity shown on the other tabs of the Ark console; click a link to navigate to the listed resource. For example, click the `Events` link to navigate to the `Clusters` tab to review the event logs.

The `Hot Topics` panel provides a link to the EDB Ark website.

Use the links in the `EDB Ark Tutorials and Documentation` section to access EDB Ark and Postgres documentation.

3.2 The Clusters Tab

Use the `Clusters` tab (shown in Figure 3.6) to create, monitor and manage active clusters that reside in the cluster.

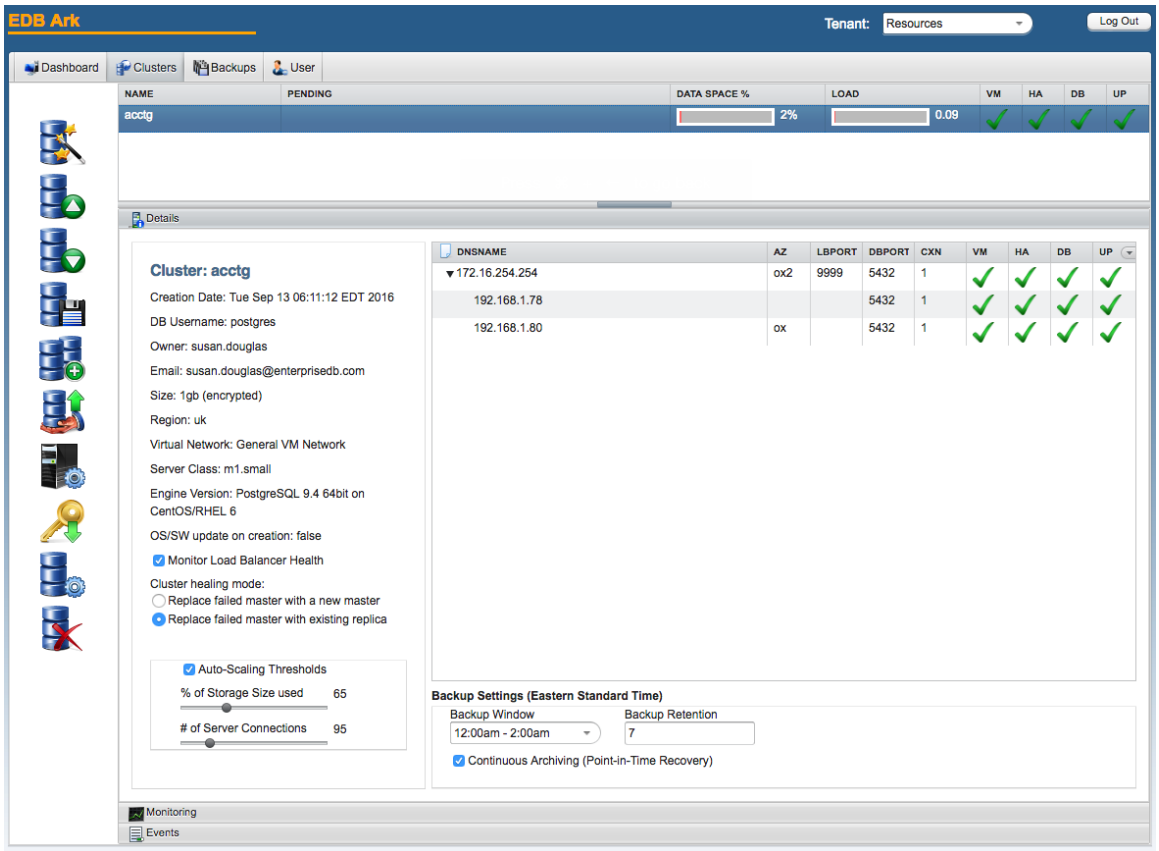


Figure 3.6 - The Clusters tab.

Indicators in the columns to the right of a cluster name display the current health of the cluster. Click on a column name to sort the contents of the column; click a second time to reverse the sort-order.

- The `VM` column displays the state of the virtual machine on which the cluster resides.
- The `HA` column displays the state of the high-availability cluster.
- The `DB` column displays the state of the database server.
- The `UP` column displays the current status of the packages installed on the cluster. Periodically, the cluster manager performs a check to see if the packages are up to date.

Status indicators provide quick visual feedback about each feature:

 A green checkmark indicates that an object is healthy.

 A yellow alert symbol calls attention to an object that requires attention.

 A red error symbol signifies that an object is not available.

 A busy-indicator signals that the cluster is processing a request.

 A question mark indicates that the state of the resource is unknown.

Use the icons along the left side of the `Clusters` tab to create new clusters or manage existing clusters:

 Use the `Add Cluster` icon to create a new Postgres cluster. For more information, see Section 4.1.

 Select the `Scale Up` icon to manually add one or more replicas to the current cluster, or add additional storage to the current cluster servers. For information about manually adding replica servers or storage, see Section 8.

 Use the `Scale Down` icon to remove one or more specified replicas from the cluster. For more information about using the `Scale Down` icon, see Section 8.2.

 Select the `Backup` icon to take a backup of the highlighted cluster (a single backup of the cluster data, and a backup of the cluster configuration files). For more information, see Section 3.3.

 Select the `Clone` icon to copy the master node of the selected database into a *clone* of the original master node. Use this feature to create a developer sandbox that is an exact duplicate of a working server; for more information about creating a clone, see Section 4.3.

When you clone a database, only the master node is recreated in the new cluster. For information about manually adding replica servers to the new cluster, see Section 8, Manual Scaling.

 You can use the `Upgrade` icon to invoke a `yum update` command on each node of your cluster, updating any installed packages to the most recent version available through your specified repositories. After performing a `yum update`, the cluster nodes will be rebooted (initiating any kernel updates required). For more information, see Section 4.1.1.

 Use the `Scale Machine Type` icon to change the size of the virtual machine for the selected cluster. EDB Ark will copy the cluster into a new cluster of a different server class (i.e. RAM and CPU), and optionally re-assign the IP address of the existing cluster to the new cluster. For more information about using the `Scale Machine Type` dialog, see Section 8.1.

 Use the `Download SSH Key` icon to download the SSH key associated with the selected cluster. Each cluster has a unique key that you can use to access nodes in that cluster. When you download a key, a popup will inform you of the steps required to connect to your cluster with SSH (see Figure 3.7).

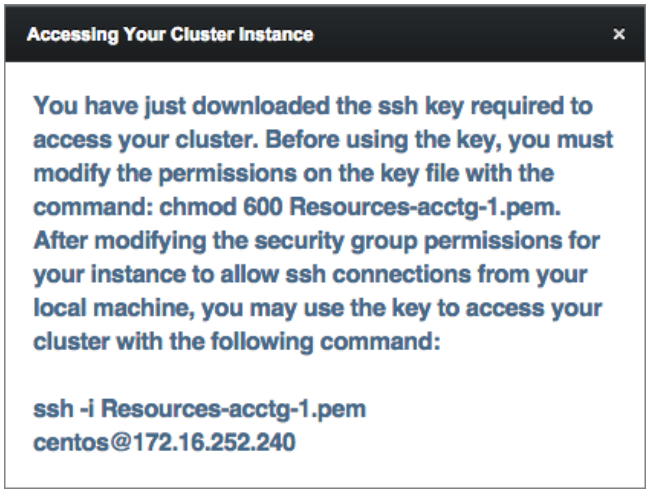


Figure 3.7 – Accessing Your Cluster Instance.

The popup displays the tenant name, the cluster name, the name that you should use when connecting to the cluster, and the IP address to which you should connect. For more information about using SSH to connect to a cluster, see Section 12.1.1.

Use the `Cluster Administrative Settings` icon to access a popup dialog that allows you to view or modify the ownership and notification email address of the currently selected cluster. For more information about the `Administrative Settings` dialog, see Section 4.4.

Use the `Delete Cluster` icon to delete the currently selected cluster. A popup dialog will ask you to confirm your decision to terminate a cluster; once terminated, a cluster may only be restored from a backup.

By default, the box next to `Release elastic IP address` is checked. Deselect this option if you wish to retain the IP address for re-use with other clusters. If you release the IP address, it will be made available for use by other clusters.

When you terminate an active cluster, backups are not deleted. Backups (including user data) are retained until they are selected and deleted from the `Backups` tab.

The panels located at the bottom of the `Clusters` tab provide easy access to helpful statistical usage and activity information about the currently selected cluster. Three navigation bars control the display; click a panel name on the navigation bar to access one of the following panels:

- Select the `Details` bar to view information about the state of the selected cluster.
- Select the `Monitoring` bar to view usage statistics for the selected cluster.
- Select the `Events` bar to review event logs describing activities on the selected cluster.

3.2.1 The Details Panel

Click the `Details` navigation bar to open the `Details` panel (shown in Figure 3.8).

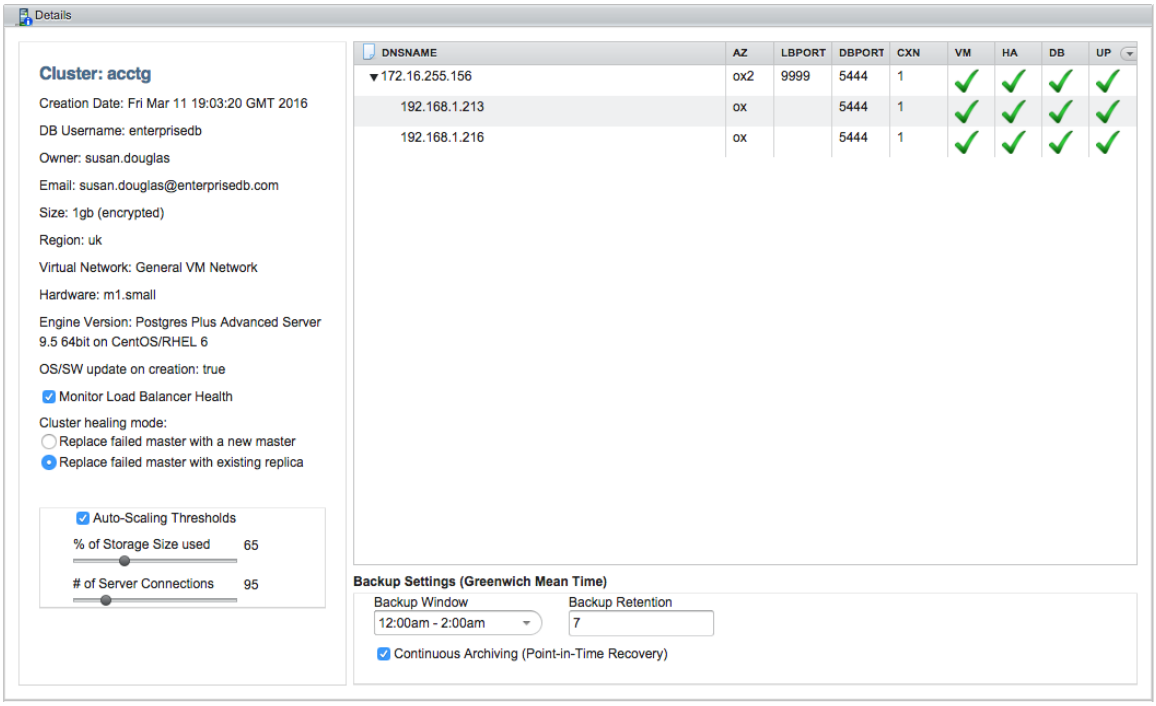


Figure 3.8 - The Details panel on the Clusters tab.

The left pane of the `Details` panel displays information about the currently selected cluster:

- The name of the selected cluster
- The date and time that the cluster was created
- The name of the database superuser for the cluster
- The name of the cluster owner
- The email address to which notifications about the cluster will be sent
- The size of the cluster
- If the cluster is encrypted
- The region in which the cluster resides
- The virtual network in which the cluster resides
- The cluster's hardware type
- The Postgres engine version that resides on the server
- If the cluster is configured to update when provisioned

You can use controls on the `Details` panel to specify:

- If Load Balancer health should be monitored
- Failover preferences for the cluster
- Auto-scaling thresholds for the cluster
- Backup preferences for the cluster
- If Continuous Archiving should be enabled for the cluster

When you modify the settings on the `Details` panel, EDB Ark displays a `New value saved` notice, confirming that the change has been saved.

Monitoring Load Balancer Health

By default, EDB Ark monitors the health of the load balancer to ensure that service is not interrupted. If the load balancer (pgpool) should fail while monitoring is enabled, PgPool will be automatically restarted. If the load balancer cannot be automatically restarted, EDB Ark will display a warning sign in the `DB` column next to the cluster name, and send a notification email to the cluster user.

Deselect the `Monitor Load Balancer Health` checkbox to indicate that you do not wish for load balancer health to be monitored and automatically restarted if an interruption in service is detected.

Selecting a Cluster Healing Mode

Use the `Cluster healing mode` radio buttons to specify the type of failover that should be employed:

- Select the `Replace failed master with a new master` radio button to specify that the cluster manager should create a new master to replace a failed master node.

  When replacing a failed master node with a new master node, the data volumes from the failed instance are attached to the new master node, preserving data integrity, while the replicas continue serving client queries.

- Select the `Replace failed master with existing replica` radio button to specify that the cluster manager should promote a replica node to be the new master node for the cluster.

  When replacing a failed master node with an existing replica, a replica node is marked for promotion to master node, while the other replica nodes are re-configured to replicate data from the new master node. Since replica nodes use asynchronous replication, any data that was committed to the old master node, but not pushed to the replica prior to the node failure will be lost.

Please note that replacing a failed master node with a new master node can take a bit longer than promoting a replica node to the role of master, but it does have the advantage

of guaranteeing that no committed data will be lost. If recovery time for your cluster is more important than preserving any non-replicated transactions, then select `Replace failed master with existing replica` as the healing mode.

Adjusting Auto-Scaling Thresholds

Use the `Auto-Scaling Thresholds` controls on the `Details` panel to adjust the threshold at which EDB Ark automatically scales up cluster resources. For more information about using the controls, see Section 9.1, *Adjusting the Automatic Scaling Thresholds*.

Modifying Backup Settings

Use the fields in the `Backup Settings` box to change your backup preferences for the selected cluster:

- Use the `Backup Window` drop-down listbox to select an optimal time to process cluster backups; specify a time when the number of clients accessing the database is minimal.

- Use the `Backup Retention` field to specify the number of backups that should be stored for the selected cluster.

- Select the checkbox next to `Continuous Archiving (Point-in-Time Recovery)` to enable point-in-time recovery for a cluster. When enabled, a base backup is automatically performed that can to be used to restore to a specific point in time. All subsequent automatic scheduled backups will also support point-in-time recovery. Note that if you deselect this option, the cluster (and subsequent automatic backups) will be re-configured to not include support for point-in-time recovery.

  When point-in-time recovery is enabled, the value specified in the `Backup Retention` field determines the duration of the point-in-time recovery backup window. For example, if you specify a value of `7`, the backup window will be `7` calendar days long. When the backup retention threshold is reached, the oldest base backup is removed, as well as any WAL files required to perform a recovery with that backup.

Reviewing Cluster Connection and Status Information

The `DNSNAME` table (located on the right side of the `Details` panel) contains a status overview and connection information for the selected cluster. If you have created replicas, the secondary server nodes are listed below the master node in the tree control; expand the tree control to view the status of the replication nodes.

Status indicators on the `Clusters` tab provide quick visual feedback about the status of your cluster:

 A green checkmark indicates that an object is healthy.

 A yellow alert symbol calls attention to an object that requires processing.

 A red error symbol signifies that an object is not available.

 A question mark indicates that the state of the resource is unknown.

Use the drop-down tab in the upper-right corner of the `DNSNAME` pane to select the columns that will be displayed in the pane:

- The `AZ` column displays the Availability Zone in which the cluster resides.
- The `LBPORT` column displays the port number to which a client application should connect to utilize load balancing.
- The `DBPORT` column displays the default listener port for the Advanced Server or PostgreSQL server.
- The `CXN` column displays the current number of connections to the node
- The `VM` column displays the state of the virtual machine on which the cluster resides.
- The `HA` column displays the state of the high-availability cluster.
- The `DB` column displays the state of the database server.
- The `UP` column displays the current status of the packages installed on the cluster. Periodically, the cluster manager performs a check to see if the packages are up to date. If an update becomes available, the `UP` column will display:

   A yellow alert symbol if the update is non-critical.

   A red error symbol if the update is a critical (security) alert.

  If alerted to an out-of-date package, you can use the `Upgrade` icon to invoke a `yum update` to update the package on all of the nodes on your cluster.

3.2.2 The Monitoring Panel

The `Monitoring` panel displays graphs that allow you to review statistical usage information about the amount of storage and the CPU load for the selected cluster (see Figure *3.9*).

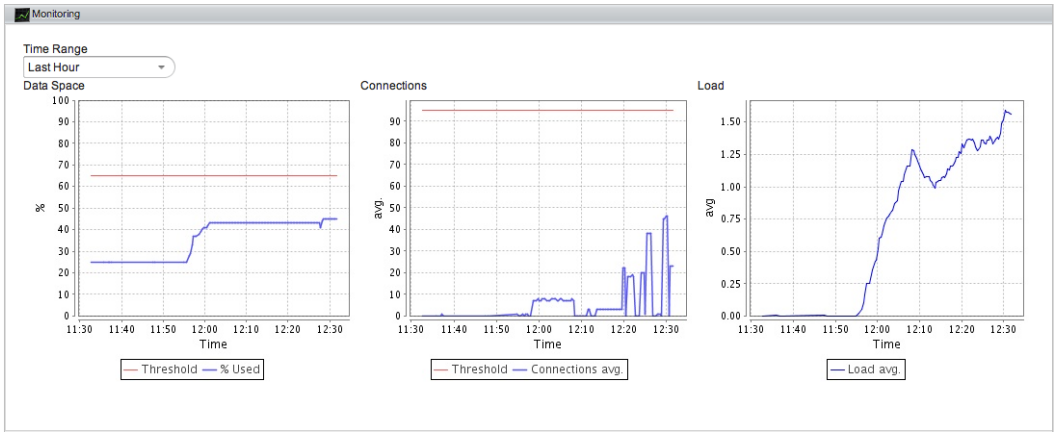


Figure 3.9 - The Monitoring panel displays usage information.

Use the `Time Range` drop-down listbox to modify the time period that the charted information on the `Monitoring` panel spans.

The graphs on the `Monitoring` panel display resource usage information:

- The `Data Space` chart displays the amount of allocated data space used by the selected cluster. The red line denotes the threshold specified by the `Data Space Threshold` slider on the `Details` panel (the threshold at which the cluster will be scaled-up). The blue line indicates the amount of the data space that is currently in use.

- The `Connections` chart displays a graph of the number of connections to the cluster during the selected time range. The red line denotes the threshold specified by the `Connections` slider on the `Details` panel.

- The `CPU Load` chart displays the processing load placed on the CPU by connecting clients. The value displayed is the actual load average as read from the program, `/proc/loadavg`. The chart shows the number of jobs in the run queue (state `R`) or waiting for disk I/O (state `D`), averaged over 15 minute periods.

3.2.3 The Events Panel

The `Events` panel (shown in Figure 3.10) displays an event log containing a history of selected events for the connected user.

Events

ID	TIME	MESSAGE
12,751	Mon May 04 10:36:41 BST 2015	Creation of cluster acctg started.
12,752	Mon May 04 10:36:59 BST 2015	Mapping Elastic IP Address 172.16.255.21
12,753	Mon May 04 10:37:04 BST 2015	Provisioning master: 172.16.255.21
12,757	Mon May 04 11:08:13 BST 2015	Connecting 172.16.255.21
12,758	Mon May 04 11:09:33 BST 2015	Creating Data Volume
12,759	Mon May 04 11:09:48 BST 2015	Configuring encryption...
12,760	Mon May 04 11:09:54 BST 2015	Creating node certificates
12,761	Mon May 04 11:10:09 BST 2015	Starting server...
12,762	Mon May 04 11:10:19 BST 2015	Creation of master server in cluster acctg completed.
12,763	Mon May 04 11:10:19 BST 2015	Performing (PITR) Base backup
12,764	Mon May 04 11:10:19 BST 2015	(PITR) Base Backup of cluster acctg started.
12,765	Mon May 04 11:10:19 BST 2015	Creating 2 replica(s)...
12,766	Mon May 04 11:10:23 BST 2015	Backup in progress
12,767	Mon May 04 11:10:24 BST 2015	Saving snapshots of attached volumes
12,768	Mon May 04 11:10:32 BST 2015	Database Server changed to state RUNNING on 172.16.255.21:
12,769	Mon May 04 11:10:33 BST 2015	The OS/SW status on node 172.16.255.21 of cluster 42a0-8653-8d3e2ae89367-acctg is now OK
12,770	Mon May 04 11:11:19 BST 2015	Backup of cluster acctg started.
12,771	Mon May 04 11:11:22 BST 2015	Backup in progress
12,772	Mon May 04 11:11:24 BST 2015	Saving snapshots of attached volumes
12,773	Mon May 04 11:11:26 BST 2015	Backup of cluster acctg completed.
12,774	Mon May 04 11:11:26 BST 2015	Addition of replica to cluster acctg started.

Figure 3.10 - The Events panel displays server activity.

Highlight a cluster name to display only events for that cluster; if you do not select a cluster, the `Events` panel will display the collected events for the connected user.

- Click a column heading to sort the logged activity by the selected column; click again to reverse the sort order.

- Use a mouse to select multiple rows from the event log for copy and paste operations.

3.3 The Backups Tab

Use the `Backups` tab (shown in Figure 3.11) to manage cluster backups; the tab displays a list of the available backups.

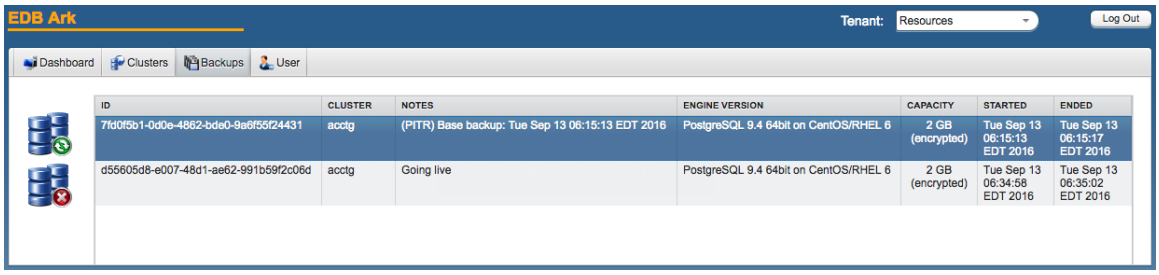


Figure 3.11 - The Backups tab of the Ark console.

A backup captures and stores the status and condition of a cluster at a specific point-in-time. Click a column heading to sort the column contents; click again to reverse the sort order.

If the comment in the `NOTES` column for a specific cluster includes `PITR`, point in time recovery is enabled on the cluster. When point in time recovery is enabled, the backup can be used to restore your cluster to a state at any given time since the backup was taken.

Use the icons on the left side of the `Backups` tab to restore or delete backups:


Highlight a backup in the list, and click the `Recover Backup` icon to open a dialog that allows you to restore a cluster from the selected backup. Specify a name for the cluster, and click the `Recover` button to continue. A popup confirms that the cluster is being restored; close the popup and navigate to the `Clusters` tab to monitor the restoration process.


Highlight one or more backups in the list and click the `Delete Backup` icon to delete the selected backups. A popup will ask you to confirm that you wish to delete the selected backups before the backups are actually deleted.

3.4 The User Tab

Fields on the `User` tab (shown in Figure 3.12) allow you to view or modify information about the current user.

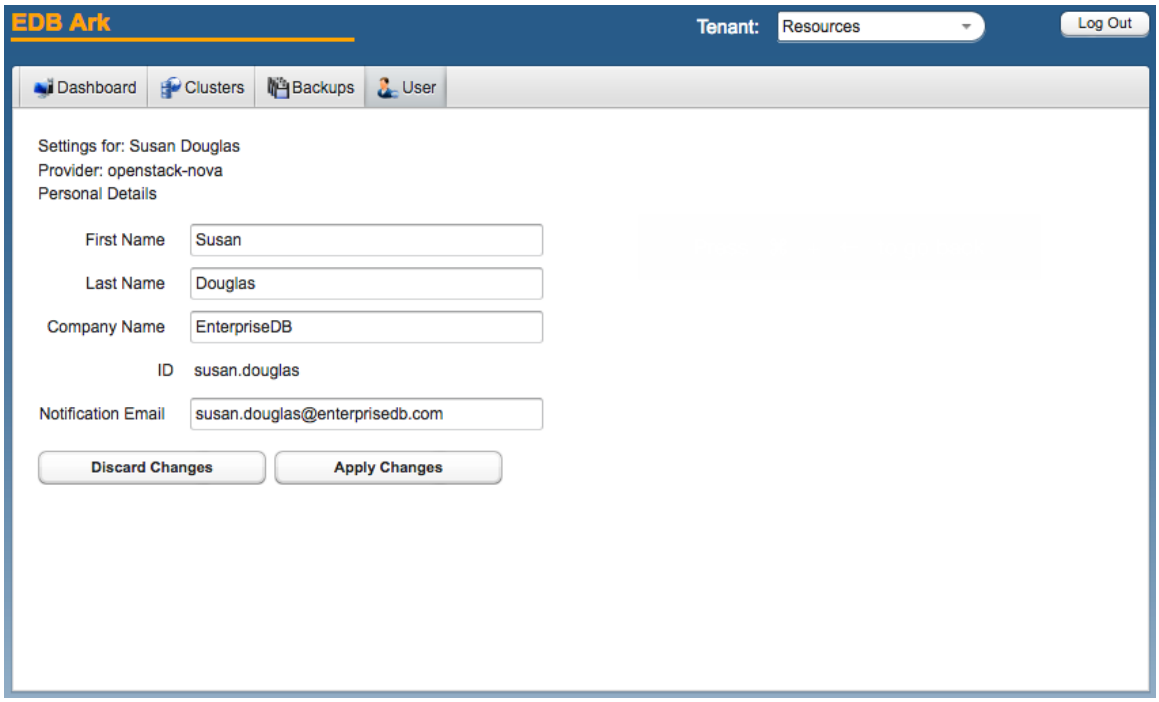


Figure 3.12 - The User tab of the Ark console.

To change the `First Name`, `Last Name`, or `Company Name` of the registered user, modify the corresponding fields and click the `Apply Changes` button. A popup will confirm that the changes have been applied (see Figure 3.13).

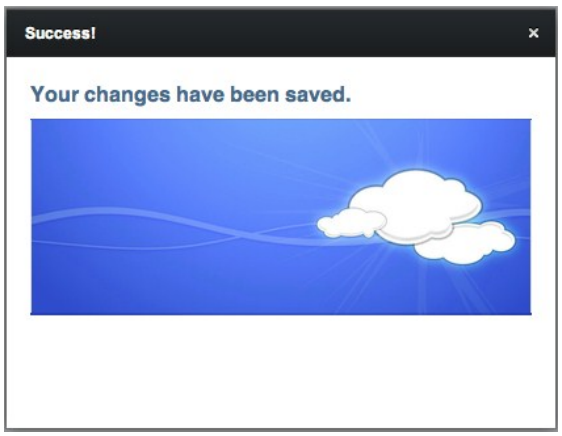


Figure 3.13 - EDB Ark confirms modifications to user information.

The `Notification Email` field on the `User` tab displays the default email address that will be used for cluster notifications unless an alternate address is provided. You can optionally:

- provide an alternate email address when a cluster is created (on the `Create a new Server Cluster` dialog).
- modify a cluster's notification email address on the `Administrative Settings` dialog.

To change the default notification email address, enter a new address in the `Notification Email` field, and click the `Apply Changes` button. A popup dialog will open, prompting you to enter your password to confirm the change of address (see Figure 3.14).

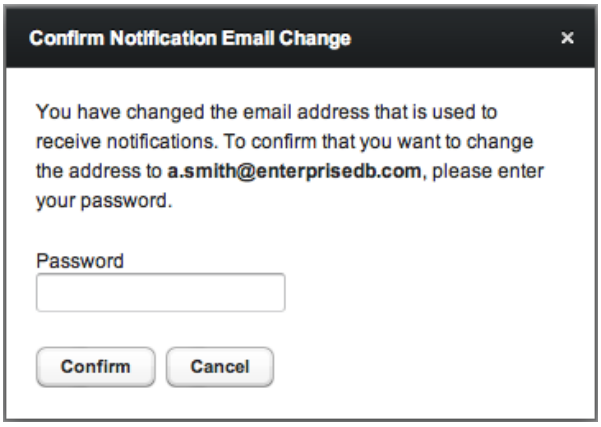


Figure 3.14 - Confirming a change in the notification email address.

Enter your password, and click `Confirm` to modify the address, or click `Cancel` to exit the popup without applying the change.

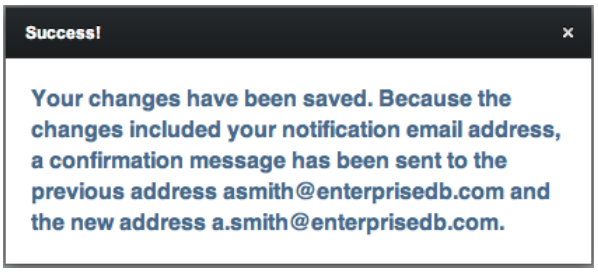


Figure 3.15 - The notification email address has been modified.

If you elect to change the notification email address, EDB Ark will send a confirmation email to both the old notification address and the new notification address (see Figure 3.15).

4 Creating a Server Cluster

There are several ways to create a new EDB Ark server cluster. You can:

- Use the `Launch DB Cluster` button (located on the `Dashboard Tab`) to open the `Create a new Server Cluster` dialog and define a new cluster.
- Click the `New Server` button (located on the `Clusters` tab) to open the `Create a new Server Cluster` dialog and define a new cluster.
- Clone a new server cluster from an existing cluster.
- Restore an existing cluster definition from backup.

The sections that follow detail several of the methods for creating a new cluster.

4.1 Creating a New Server Cluster

Before you can connect to Postgres from a client application, you must create a server cluster. Use the `Launch DB Instance` button (located in the upper left panel of the `Dashboard Tab`) or click the `Add Server` button on the `Clusters` tab to open the `Create a New Server Cluster` dialog, as shown in Figure 4.1.

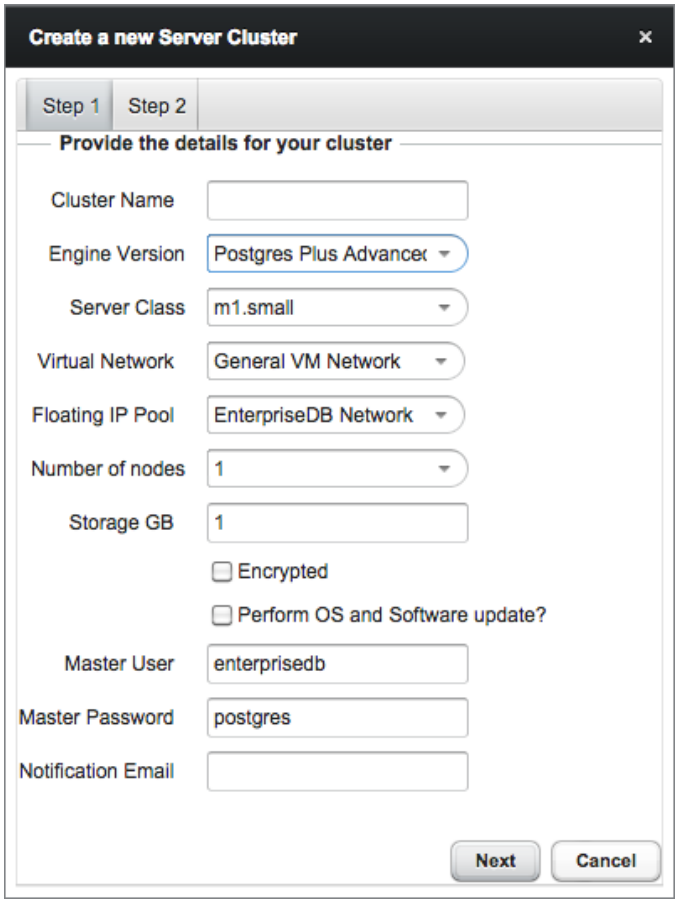


Figure 4.1 - Specify information about the new cluster on the Step 1 tab.

Use fields on the `Create a New Server Cluster` dialog to specify information about the new cluster:

- Specify a name for the new server cluster in the `Cluster Name` field.

  Warning: EDB Ark uses the name specified in the `Cluster Name` field to identify the cluster when performing management functions. The cluster name is also part of the `Instance Name` on the OpenStack console; you must not modify the name in the OpenStack management console. Changing the cluster name in the OpenStack console can break key EDB Ark features (i.e. failover).

- Use the drop-down listbox in the `Engine Version` field to select the version of the Postgres engine that you wish to use.

- Use the drop-down listbox in the `Server Class` field to specify the size of each cluster node. The server class determines the size and type (compute power and RAM) of each node within the cluster.

  You can adjust the amount of storage used by the cluster, or number of replicas in the cluster as your resource demands change. For example, you can start with a `m1.small` instance, and later, easily upgrade to a more capable `c1.medium` instance as your performance requirements dictate.

- Use the drop-down listbox in the `Virtual Network` field to specify the identity of the network in which the cluster should reside.

- Use the drop-down listbox in the `Floating IP Pool` field to select the address pool in which the cluster should reside.

- Use the drop-down listbox in the `Number of nodes` field to specify the number of server nodes that you wish to create. The name specified in the `Cluster Name` field will apply to the master node; each additional node will act as a replication server for the master node.

- Use the `Storage GB` field to specify the initial size of the data space (in Gigabytes).

- Check the box next to `Encrypted` to indicate that the cluster should be encrypted. EDB Ark uses the `aes-xts-plain` (512-bit) cipher suite to provide an encryption environment that is both secure and transparent to connecting clients. When encryption is enabled, everything residing on the cluster is encrypted except for the root filesystem.

- Check the box next to `Perform OS and Software` update to specify that a software update should be performed whenever the cluster is provisioned. For more information, see Section 4.1.1.

- Enter the name of the database superuser in the `Master User` field.

- Enter the password associated with the database superuser in the `Master Password` field.

- The default notification email associated with the user that is creating the cluster is provided in the `Notification Email` field; you can optionally provide an alternate email for cluster notifications.

Click the `Next` button to continue to the `Step 2` tab (shown in Figure 4.2).

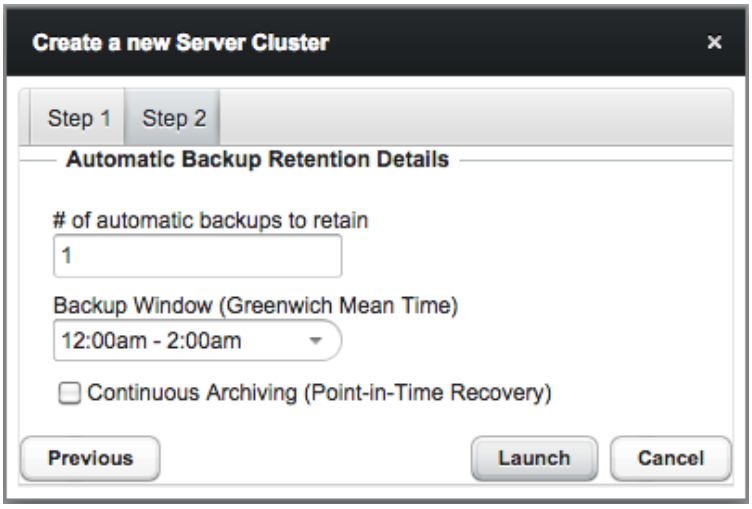


Figure 4.2 - Specify backup information on the `Step 2` tab.

Use the fields on the `Step 2` tab to specify additional database information:

- Use the `# of automatic backups to retain` field to specify the number of server backups stored. When the specified number of server backups is reached, EDB Ark will delete the oldest backup to make room for a new backup.

  When point-in-time recovery (PITR) is enabled, the value specified in the `# of automatic backups to retain` setting determines the duration of the PITR backup window. For example, if you specify a value of `7`, the PITR backup window will be `7` calendar days long.

- Use the `Backup Window` field to specify a time that it is convenient to backup the server (you may wish to schedule backups to occur when the CPU load is the lightest).

- Check the box next to `Continuous Archiving (Point-in-Time Recovery)` to enable point-in-time recovery for the cluster. When enabled, a base backup is automatically performed that can to be used to restore to a specific point in time. All subsequent automatic scheduled backups will also support point-in-time recovery. Note that if you deselect this option, the cluster (and subsequent automatic backups) will be re-configured to not include support for point-in-time recovery.

Use the `Previous` button or select a tab to return to the `Step 1` tab to review or update information; when you have completed the `Create a New Server` dialog, click `Launch` to create the database cluster.

A popup dialog confirms that EDB Ark is creating a new cluster (see Figure 4.3); click the `X` in the upper-right corner of the popup to close the popup.

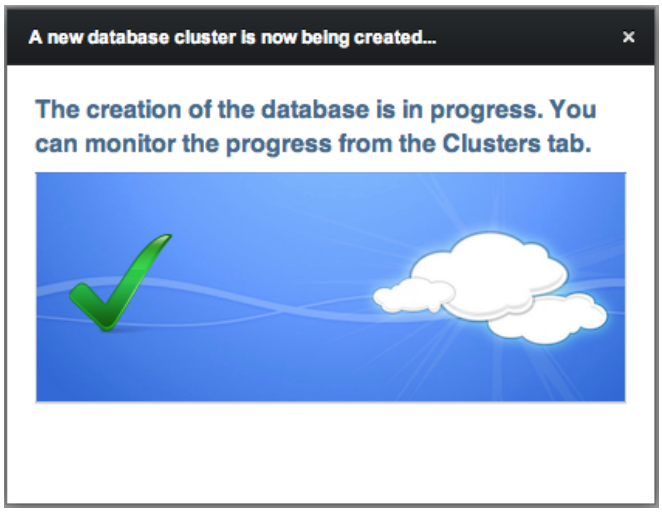


Figure 4.3 - A popup confirms that the new cluster is being created.

Navigate to the `Clusters` tab of the Ark console to monitor the creation of the cluster.

4.1.1 Perform OS and Software Update

Check the box next to `Perform OS and Software update` when defining a cluster (see Figure 4.4) to instruct EDB Ark to perform a `yum update` whenever the cluster is provisioned. The `yum update` command will update all of the packages that reside on the cluster that have an update available. Provisioning occurs when a cluster is scaled up, scaled down, or during a failover.

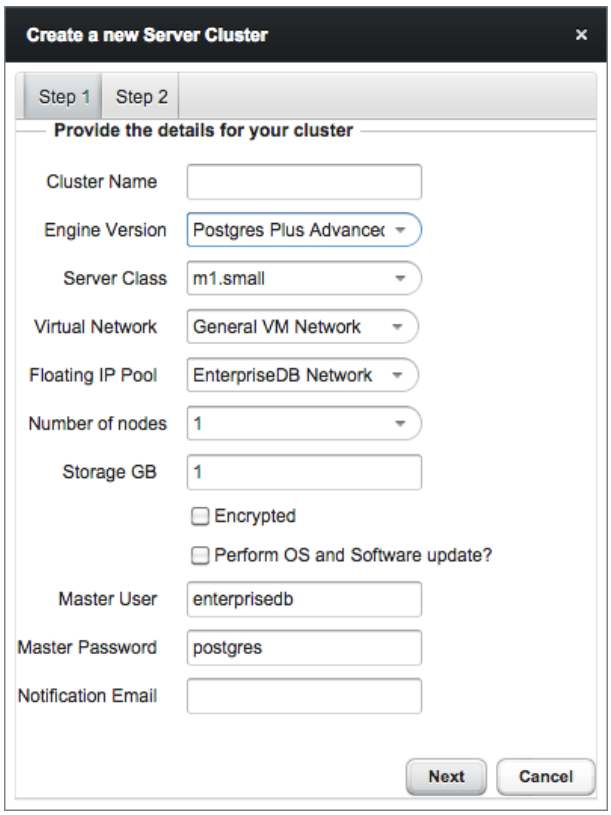


Figure 4.4 – The new Server Cluster dialog.

When you check the box next to `Perform OS and Software update ?`, EDB Ark will warn you that enabling this functionality can significantly slow down cluster operations (see Figure 4.5).

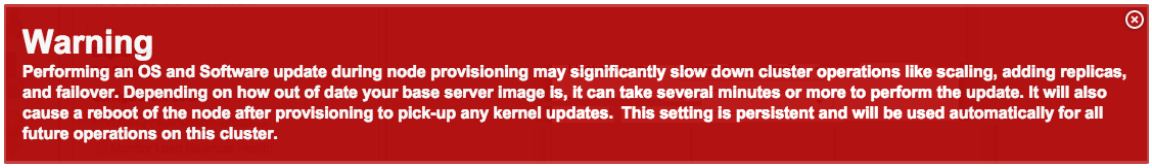


Figure 4.5 – The software update warning.

Updating packages may slow down cluster maintenance operations; an update can easily take 10 minutes or more, and will initiate a reboot of the node (updating the operating system kernel). This setting is persistent; if you enable software updates for a cluster, you cannot directly disable software updates for that cluster at a later time.

4.2 Creating a Cluster that Enables Point-In-Time Recovery

EDB Ark supports point-in-time recovery (PITR). To create a cluster that implements PITR, use the fields on the `Step 1` tab of the `Create a new Server Cluster` dialog to specify information about the cluster (see Figure 4.6).

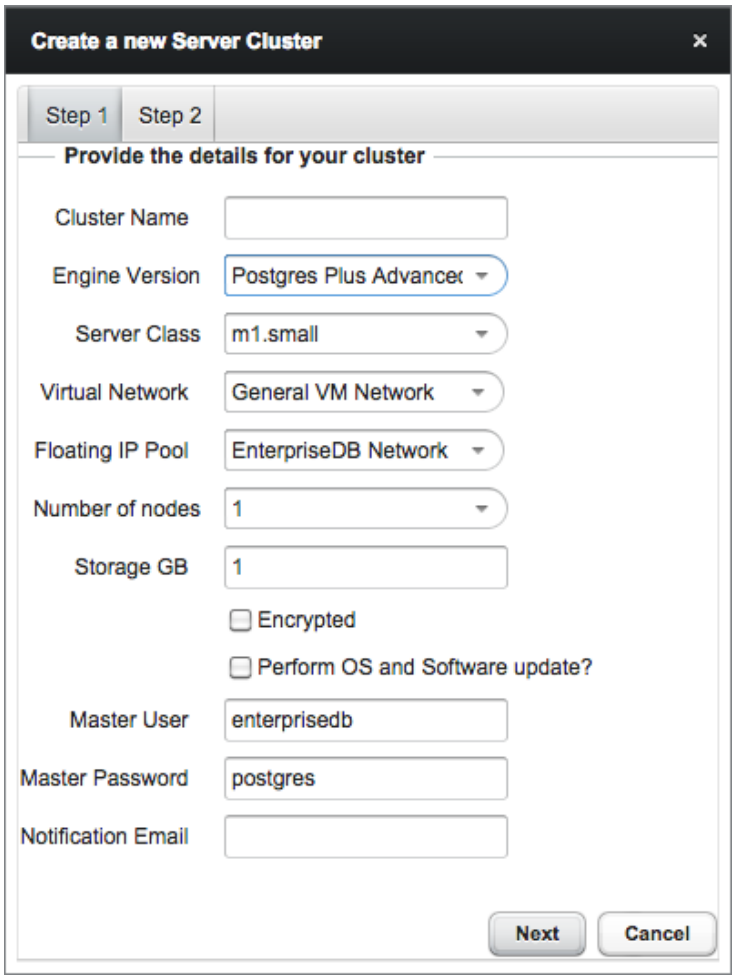
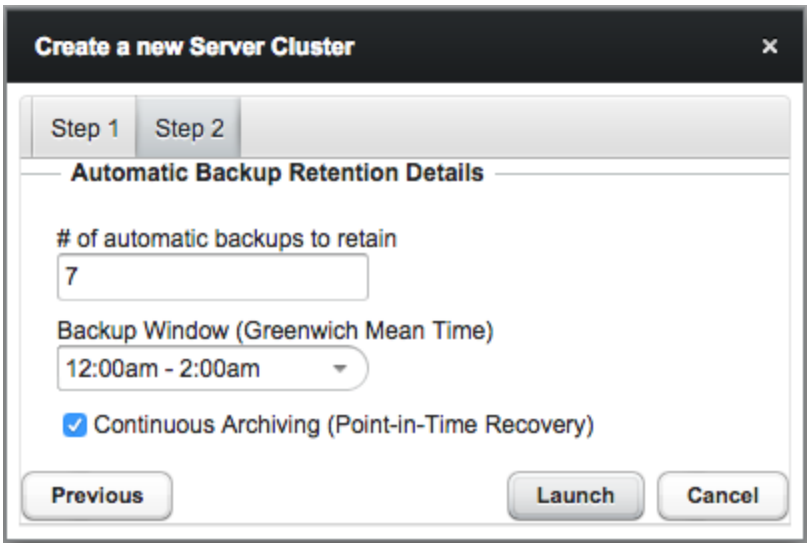


Figure 4.6 – Name the cluster and select cluster options.

After specifying options on the `Step 1` tab, select `Next` to continue to the `Step 2` tab (shown in Figure 4.7).



4.7 - Implementing PITR on a new cluster

On the `Step 2` tab, check the box next to `Continuous Archiving (Point-in-Time Recovery)` to specify that the cluster should be created with point-in-time recovery enabled.

When enabled, a base backup is automatically performed; the base backup and all subsequent scheduled backups will support point-in-time recovery. Note that if you do not select this option, the cluster (and subsequent automatic backups) will be configured to not include support for point-in-time recovery.

When point-in-time recovery is enabled, the value specified in the `Backup Retention` field determines the duration of the point-in-time recovery backup window. For example, if you specify a value of `7`, the backup window will be `7` calendar days long. When the backup retention threshold is reached, the oldest base backup is removed, as well as any WAL files required to perform a recovery with that backup.

4.3 Creating a Developer Sandbox

With a few simple steps, you can create a developer sandbox that is an exact duplicate of the original master node:

1. Navigate to the `Clusters` tab.

2. Highlight the name of the cluster you wish to clone into the sandbox.

3. Click the `Clone` icon located on the left side of the window.

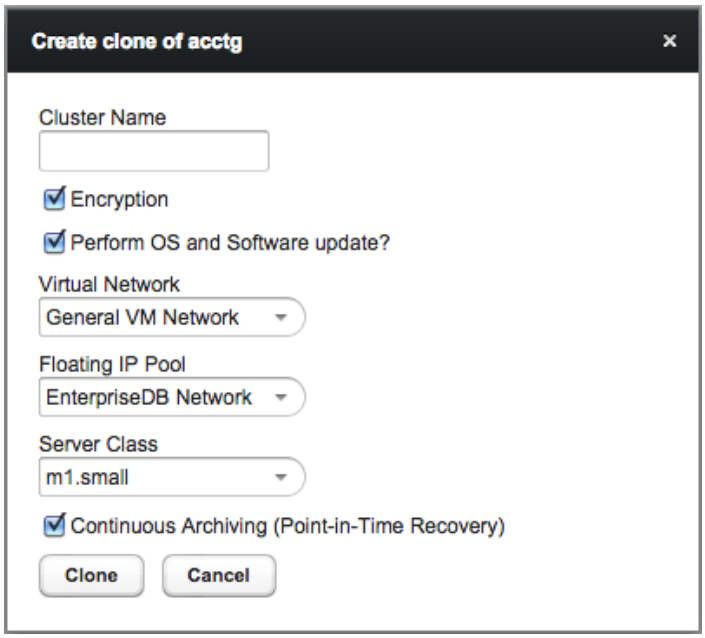


Figure 4.8 - Creating a clone of a database.

When the `Create clone...` dialog (shown in Figure 4.8) opens:

- Provide a name for the clone in the `Cluster Name` field.

- Check the box next to `Encryption` if you would like the clone to be created in an encrypted cluster.

- Check the box next to `Perform OS and Software update`? if you would like the server to perform a software update each time the clone is provisioned. For more information, see Section 4.1.1.

- Use the `Virtual Network` drop-down list box to specify a network name.

- Use the `Floating IP Pool` drop-down to specify the name of the IP pool that the cluster will use.

- Use the `Server Class` drop-down listbox to specify the initial size of the new cluster.

- Check the box next to `Continuous Archiving (Point-in-Time Recovery)` to enable point-in-time recovery on the clone.

When you've completed the dialog, click the `Clone` button to create the sandbox.

When you clone a database, only the master node is recreated in the new cluster; for information about manually adding replica servers to the new cluster, see Section 8, *Manual Scaling*.

4.4 Modifying a Cluster's Administrative Settings

Fields on the `Administrative Settings` dialog display the current owner and the email address to which notification emails about the state of the cluster are sent.

To modify the owner of a cluster or the email address associated with a cluster, highlight the name of a cluster on the `Clusters` tab, and click the `Administrative Settings` icon. The dialog shown in Figure 4.9 opens.

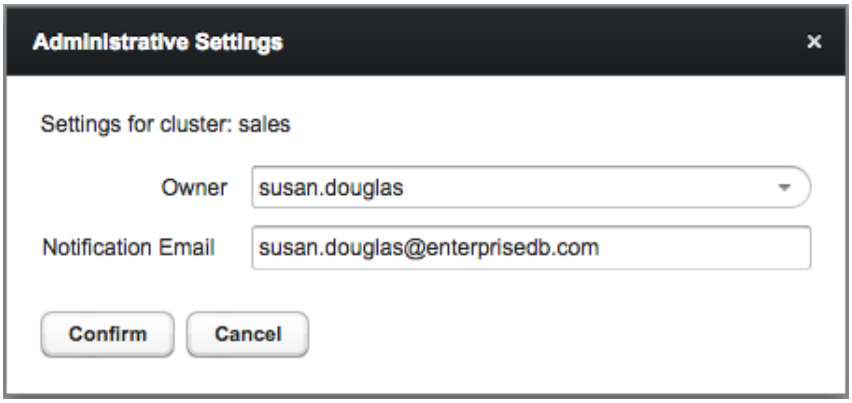


Figure 4.9 – The Administrative Settings dialog.

Use the fields on the dialog to modify the administrative settings for the cluster:

- Use the drop-down listbox in the `Owner` field to select a new cluster owner; please note that only those users with permissions to access the tenant on which the cluster resides are included in the list.

- Use the `Notification Email` field to specify the address to which you wish notices about the state of the cluster to be sent.

After modifying the owner or notification email for the cluster, click the `Confirm` button; a dialog will open, prompting you to provide the password associated with the connected session (see Figure 4.10).

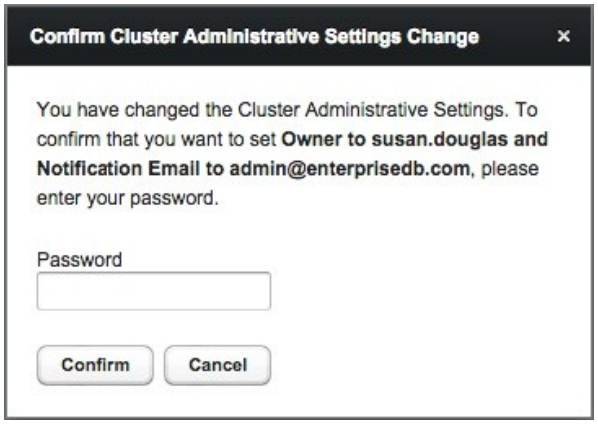


Figure 4.10 – Provide a password to confirm changes.

Provide a password in the `Password` field and click `Confirm` to save your changes and exit, or `Cancel` to exit without saving the changes.

5 Connecting an Application to an EDB Ark Cluster

Before connecting to a database that resides on OpenStack, you must configure the security rules on the OpenStack server to make a port available to accept the connection. After opening a port for the connection, the client can connect by providing the IP address and port of the server, and the credentials associated with the role defined when the server cluster was created.

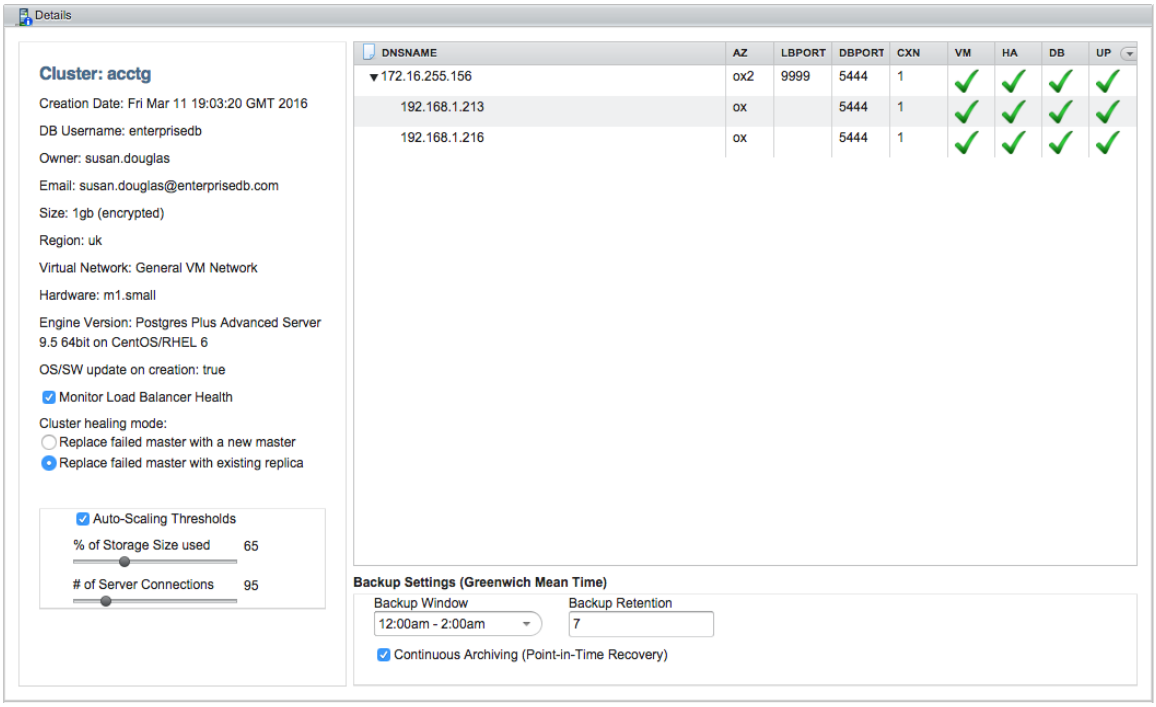


Figure 5.1- The Details panel on the Clusters tab.

If you have defined a cluster with two or more servers, client applications should always connect to the load balancing port of the master server (the first DNS name listed in the `Details` panel). This will ensure that read requests are distributed efficiently across the cluster replicas to maximize performance, while write requests are directed only to the cluster master. Replica server nodes are listed below the master node in the tree view.

- The `DNSNAME` column displays the address of the node; a connecting client should use this address when connecting to a specific server (see Figure 5.1).

- The `LBPORT` column displays the port number to which a client application should connect to utilize load balancing.

Since only the master node of a multi-server cluster operates in read/write mode, all write queries will be directed to the master node, while any read-only queries may be directed to a replica node.

- The `DBPORT` column displays the default listener port for the Advanced Server or PostgreSQL server. To connect directly to the database listener port, modify the cluster's security group (in the OpenStack console) to allow connections from your client.

Use the authentication information (`Master User` and `Master Password`) provided on the `Create a New Server Cluster` dialog to establish the initial connection as the database superuser, using a client application (such as EDB-PSQL or Postgres Enterprise Manager). Please note that connecting with this identity grants you superuser privileges on the server; you should not share this connection information with un-trusted users.

After connecting as the database superuser, you should create lesser-privileged user roles with which non-administrative users will connect.

For detailed information about connecting to an EDB Ark cluster with `ssh`, `edb-psql` (or `psql`), or the Postgres Enterprise Manager client (including pgAdmin), please see Section 12.1, *Connecting to the Cluster*.

5.1 Using iptables Rules

If you are using iptables rules to manage security in an OpenStack image or on the host of the Ark console, please note that you must *not* modify the iptables rules provided by EDB Ark.

If you are using iptables on the host of the Ark console, do not modify the following rules:

```
iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 –j
     REDIRECT --to-port 8080
iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 443 –j
     REDIRECT --to-port 8181
iptables -I INPUT 1 -p tcp --dport 4848 -j ACCEPT
iptables -I INPUT 1 -p tcp --dport 8181 -j ACCEPT
iptables -I INPUT 1 -p tcp --dport 8080 -j ACCEPT
/sbin/service iptables save
```

These rules:

- redirect `http` and `https` traffic on ports `80` and `443` to the default GlassFish ports (`8080` and `8181`).
- allow inbound traffic to the default GlassFish administration port.
- allow inbound traffic on `8080` and `8181`.
- save the configuration (to preserve the behaviors when the system reboots).

If you are using iptables on an Advanced Server cluster, do not modify the following rules:

```
iptables -I INPUT 1 -p tcp --dport 7800:7802 -j ACCEPT
iptables -I INPUT 1 -p tcp --dport 5444 -j ACCEPT
/sbin/service iptables save
```

If you are using iptables on a PostgreSQL cluster, do not modify the following rules:

```
iptables -I INPUT 1 -p tcp --dport 7800:7802 -j ACCEPT
iptables -I INPUT 1 -p tcp --dport 5432 -j ACCEPT
/sbin/service iptables save
```

The rules:

- allow inbound traffic from the Ark console on ports 7800 and 7802.
- allow inbound traffic on the database listener ports.
- save the configuration (to preserve the behaviors when the system reboots).

6 Managing Backups and Recovery

When you use the Ark console to take a backup, EDB Ark makes a copy of the contents of the PostgreSQL `PGDATA` directory. The `PGDATA` directory contains the data and the meta-data required to construct an exact copy of the Postgres data cluster (the data and the database objects that reside within that Postgres instance).

To capture a backup of a cluster, navigate to the `Clusters` tab, highlight a name in the cluster list, and click the `Backup` icon.

Figure 6.1 - The Backup Data? dialog.

You can include a reference note about the backup that can be viewed on the `Backups` tab by adding a message to the `Optional notes` field on the `Backup Data?` dialog before clicking the `Backup` button (see Figure 6.1).

When you click the `Backup` button, EDB Ark will perform the backup. While EDB Ark performs the backup, the `PENDING` column of the selected cluster (on the `Clusters` tab) will display the message, `Backup in progress`.

6.1 Performing a Base Backup for Point-In-Time Recovery

When point-in-time recovery is enabled, a base backup is automatically performed that can to be used to restore to a specific point in time. All subsequent automatic scheduled backups will also support point-in-time recovery. Note that if you deselect this option, the cluster (and subsequent automatic backups) will be re-configured to not include support for point-in-time recovery.

When point-in-time recovery is enabled, the value specified in the `Backup Retention` field of the `Create cluster` dialog determines the duration of the point-in-time recovery backup window. For example, if you specify a value of `7`, the backup window will be `7` calendar days long. When the backup retention threshold is reached, the oldest base backup is removed, as well as any WAL files required to perform a recovery with that backup.

Please note that you cannot perform a base backup on a cluster while the database is in recovery and not accepting connections. If you attempt to perform a base backup during recovery, the backup will fail (the failure will be noted on the `Events` panel of the `Clusters` tab). You should instead wait until the database recovery is complete to enable point-in-time recovery for the cluster.

Point-in-time recovery is enabled on the `Details` panel of the `Clusters` tab. If a base backup fails, you can trigger EDB Ark to perform a base backup by disabling point-in-time recovery, and then (after waiting a few minutes) re-enable point-in-time recovery.

6.2 Reviewing Stored Backups

Navigate to the `Backups` tab (shown in Figure 6.2) to review a list of stored cluster backups.

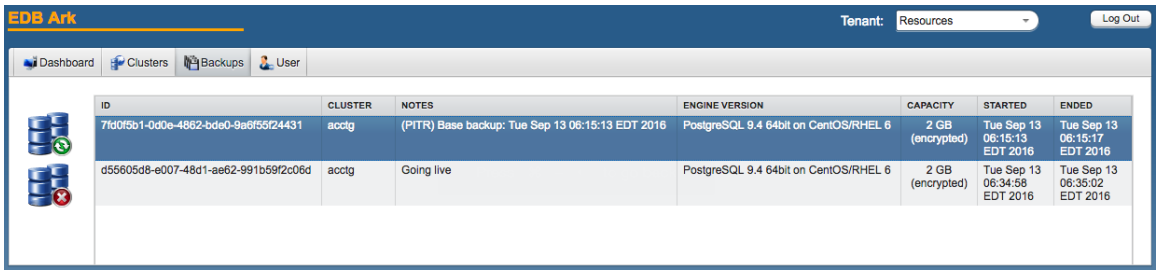


Figure 6.2 - The Backups tab of the Ark console.

A backup captures and stores the status and condition of a cluster at a specific point-in-time.

- The `ID` column contains a unique backup identifier.
- The `CLUSTER` column displays the name of the cluster that was the target of the backup.
- The `NOTES` column displays an informational note (provided by either the user or the system at the time of backup). Those messages that include (PITR) can be restored to a specific point-in-time within the backup window.
- The `ENGINE VERSION` column contains a description of the Postgres version that the saved cluster is using.
- The `CAPACITY` column contains the storage capacity of the cluster at the time that the backup was taken.
- The `STARTED` column displays the date and time that the backup was initiated.
- The `ENDED` column displays the data and time that the backup completed.

You can use the icons on the left side of the `Backups` tab to restore or delete the selected backup:


Highlight a backup in the list, and click the `Recover Backup` icon to open a dialog that allows you to restore a cluster from the selected backup.

 Highlight one or more backups in the list and click the `Delete Backup` icon to delete the selected backups. A popup will ask you to confirm that you wish to delete the selected backups before the backups are actually deleted.

6.3 Restoring a Cluster from Backup

You can restore a cluster, recovering the state of a cluster at the time that a selected snapshot was taken, or use the restoration process to create a developer sandbox. To restore a cluster, navigate to the `Backups` tab, and highlight the backup to be restored in the onscreen list.


Click the `Recover Backup` icon, located on the left side of the window.

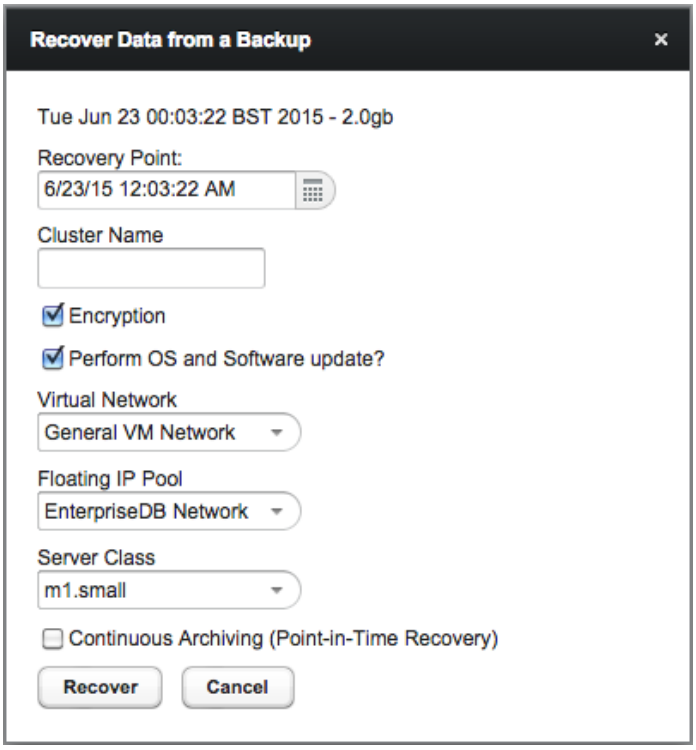


Figure 6.3 - The Recover Data from a Backup dialog.

When the `Recover Data from a Backup` dialog (shown in Figure 6.3) opens:

- If applicable, use the calendar selector in the `Recovery Point` field to specify the recovery target (the date and time that the database was in the state in which you wish the new cluster to start). The `Recovery Point` field is only displayed for backups that were taken with point-in-time recovery implemented; you cannot perform a point-in-time recovery with a backup unless point-in-time recovery is enabled for the cluster when the backup was taken.

- Specify a name for the new cluster in the `Cluster Name` field.

- Check the box next to `Encryption` to specify that the new cluster should reside in an encrypted cluster. Please note that you can restore a non-encrypted backup into an encrypted cluster.

- Check the box next to `Perform OS and Software update` to instruct EDB Ark to perform a `yum update` whenever the cluster is provisioned.

- Use the `Virtual Network` drop-down listbox to select a virtual network.

- Use the `Floating IP Pool` drop-down listbox to specify the name of the IP pool that the cluster will use.

- Use the `Server Class` drop-down listbox to specify the server class of the new cluster.

- Check the box next to `Continuous Archiving (Point-In-Time Recovery)` to indicate that the new cluster should implement point-in-time recovery. Please note that to restore into a cluster with point-in-time recovery enabled, the backup from which you are restoring must have had point-in-time recovery implemented when the backup was taken. The checkbox will not be available if point-in-time recovery was not implemented when the backup was taken.

Click the `Recover` button to continue, or the `Cancel` button to exit without starting the recovery process. A popup confirms that the cluster is being restored (see Figure 6.4); close the popup and navigate to the `Clusters` tab to monitor the restoration process.

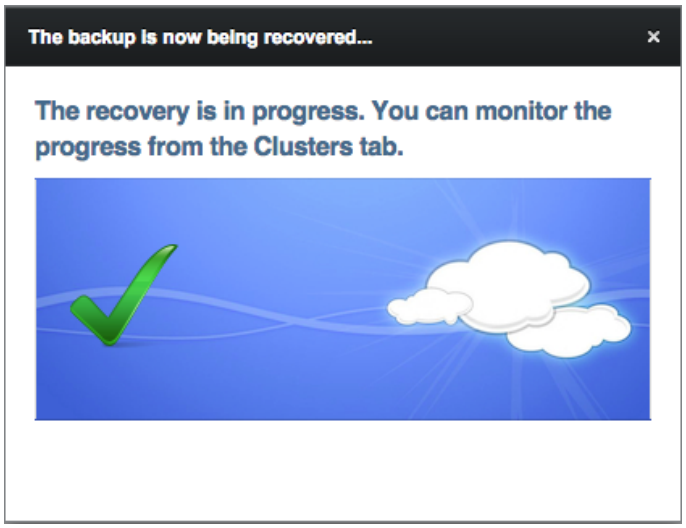


Figure 6.4 – The recovery is in progress.

Please note: when you restore a backup, the server configuration will match the original configuration, but the server addresses will change.

7 Automatic Failover

The EDB Ark cluster manager constantly monitors the state of each cluster. Each cluster is composed of a single master Postgres instance that operates in read-write mode (performing all writes to the database) and one or more replica Postgres instances. Replica nodes are read-only, automatically duplicating all data found on the master node, and all changes made to that data.

If a replica fails, EDB Ark automatically spins up a new replica instance and attaches it to the master database. The cluster continues operating during the replacement process, with the master servicing writes and reads, and the remaining replicas servicing reads. Overall read performance may degrade for a short period of time until the cluster is returned to full strength.

If a master failover occurs, the server will enforce one of two behaviors, specified by the `Cluster healing mode` radio buttons, located on the `Details` panel:

- Select the `Replace failed master with a new master` radio button to specify that the cluster manager should create a new master to replace a failed master node.

  When replacing a failed master node with a new master node, the data volumes from the failed instance are attached to the new master node, preserving all transactions that were committed on the master.

- Select the `Replace failed master with existing replica` radio button to specify that the cluster manager should promote a replica node to be the new master node for the cluster. Choose this option when speed of recovery is important, and your application can tolerate the loss of some transactions. This is the default behavior.

  When replacing a failed master node with an existing replica, a replica node is marked for promotion to master node, while the other replica nodes are re-configured to replicate data from the new master node. Since replica nodes use asynchronous replication, any data that was committed to the old master node, but not yet pushed to the replica prior to the node failure will be lost.

  If you opt to promote a replica to replace the master node, a replacement replica will also be added to the cluster during the failover process, returning the cluster to full strength. This self-healing property is at the heart of providing high availability to cluster users.

Please note that replacing a failed master node with a new master node can take a bit longer than promoting a replica node to the role of master, but it does have the advantage of guaranteeing that no committed data will be lost.

Triggering a Failover

EDB Ark will initiate a failover when OpenStack reports that an instance is terminated or stopped. Missing heartbeats from the EDB Ark instance do not trigger a failover; a missing heartbeat may merely be a sign of interrupted communication, rather than a complete instance failure.

By design, EDB Ark does **not** perform a failover when the Postgres server is stopped, because the server stop or restart may be intentional:

- A user may intentionally restart the server when performing maintenance or tuning. For example, a server restart is required when updating server configuration parameters; this restart will not invoke failover.

- If a user intentionally kills the `postmaster` process, the server will not failover; the `postmaster` process is responsible for restarting the server.

- The Postgres server may intentionally perform a server restart. For example, when a backend server process crashes (or is intentionally killed by a user), the Postgres server automatically invokes a restart.

When a failover is complete, EDB Ark does not delete the original master instance of the database; you can use the preserved master instance to perform any post-mortem activities that may be required. If you do not wish to utilize the preserved instance, you should use the OpenStack management console to delete the instance.

Please note: A cluster will not fail-over to an existing replica, if the name of that replica has been changed (in the OpenStack management console) from the original EDB Ark generated name; when you change a cluster name, the cluster manager is unable to identify an instance as part of the cluster.

8 Manual Scaling

The Ark console makes it simple to add replicas and storage to an existing cluster, or to upgrade to a larger server class (i.e. vertical scaling).

- Adding additional replicas to your database cluster increases the CPU power available to handle additional client requests or applications, increasing the number of client connections that can be serviced. When the scale up is complete, each additional replica automatically assumes a share of the read-only workload from incoming queries.

- Adding additional storage to the cluster increases the amount of data that can be stored by the database servers. When you add additional storage to the cluster, each member of the cluster gets the additional storage amount.

- Vertically scaling to a larger server class increases the processing capabilities of your cluster, allowing the server to process customer requests with greater speed.

You can also downsize a cluster by selectively removing a replica. You can machine scale to a smaller machine type to reduce resource usage (cpu/memory) and/or cost.

8.1 Manually Adding Replicas and Storage

EDB Ark's `Scale Up` dialog makes it simple to manually add additional replicas to a cluster if you find that server resources are strained. The dialog also allows you to increase the amount of storage available to a cluster.

If you specify that EDB Ark should add both storage and replicas, EDB Ark will process the request for additional storage *before* adding replicas to the cluster. All of the nodes on the cluster will be of the newly specified storage size.

 To add a replica or storage space to a cluster, navigate to the `Clusters` tab, highlight a cluster name, and select the `Scale Up` icon. The `Scale Up` dialog opens as shown in Figure 8.1.

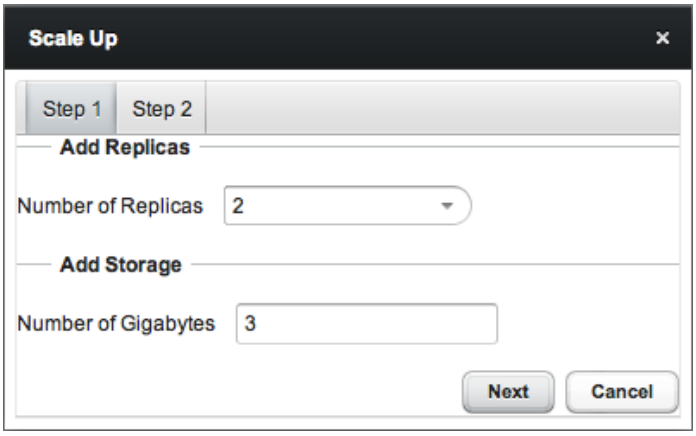


Figure 8.1 - The Scale Up dialog.

Use the drop-down listboxes on the `Step 1` tab to specify:

- The number of replicas to add to the cluster.
- The amount of storage memory (in Gigabytes) that will be added to each server in the cluster.

When you've completed the dialog, click `Next` to continue to the `Step 2` tab (shown in Figure 8.2).

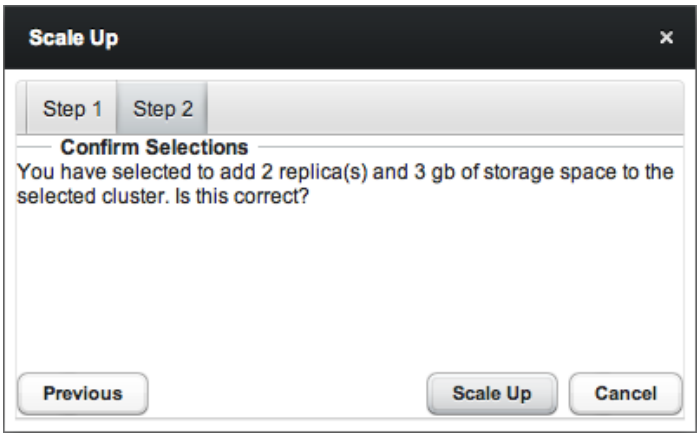


Figure 8.2 - The Scale Up dialog.

Use the `Previous` button to return to the `Step 1` tab to modify specified values. Use the `Scale Up` button to confirm that you wish to add the specified number of replication servers or the specified amount of memory to the cluster. Use the `Cancel` button, or simply close the dialog to exit without modifying the cluster.

Figure 8.3 - Scaling up is in progress.

EDB Ark will confirm that replicas or memory are being added to the cluster (as shown in Figure 8.3).

8.2 Manually Removing a Replica

EDB Ark's `Scale Down` dialog makes it simple to manually remove one or more unneeded replicas from a cluster.


To delete a replica, navigate to the `Clusters` tab, and click the `Scale Down` icon (shown above). The `Scale Down` dialog opens as shown in Figure 8.4.

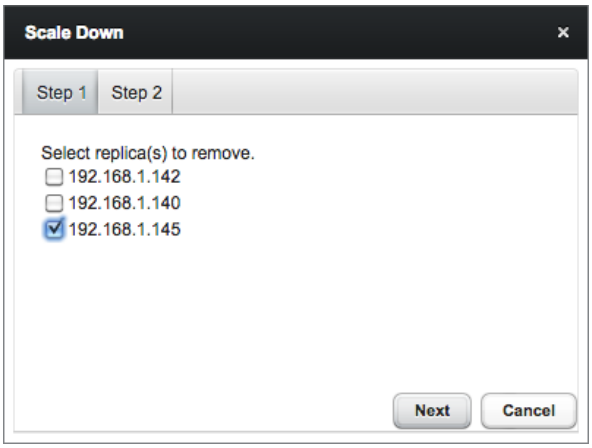


Figure 8.4 - The Scale Down dialog.

Check the box to the left of the name of a replica, and click `Next` to proceed to the `Step 2` tab of the dialog (shown in Figure 8.5).

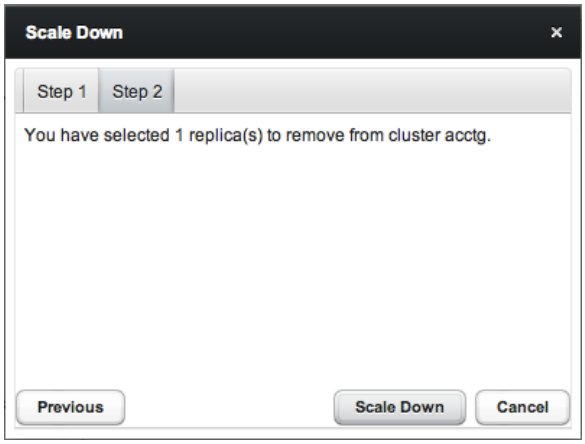


Figure 8.5 - The Step 2 tab of the Scale Down dialog.

Click `Scale Down` to confirm that you wish to remove the replica, or `Previous` to return to the `Step 1` tab. Use the `Cancel` button, or simply close the dialog to exit without modifying the cluster.

8.3 Manually Changing the Server Class

When your RAM processing needs, CPU power, or other circumstances warrant a larger virtual machine for your application, you can vertically scale to a larger server class. You can either:

- Use the `Scale Machine Type` dialog to copy the cluster into a larger server class.

  When you use the `Scale Machine Type` dialog to move your cluster into a larger server class, you must provide a new name for the upgraded cluster. You can also use the dialog to specify that EDB Ark should re-assign the IP address of the cluster, so the upgrade will be transparent to connecting clients.

  Please note: you may wish to postpone the IP address reassignment to perform configuration tasks or test the new server size.

- Use the `pg_dump` and `pg_restore` utilities to move the cluster into a larger server class.

  To move to a larger server class, use the `pg_dump` utility to make a backup of the cluster. After backing up the cluster, create a new instance with the larger server class, and use `pg_restore` to restore the cluster on the new instance. For more information about using `pg_dump` and `pg_restore`, see Section 12.2 *Moving an Existing Database into a New Cluster*.

You can also downsize a cluster by selectively removing a replica. You can machine scale to a smaller machine type to reduce resource usage (cpu/memory) and/or cost.

When you vertically scale your cluster with the `Scale Machine Type` dialog, EDB Ark will copy the existing cluster into a new cluster of a different server class, and optionally re-assign the IP address of the existing cluster to the new cluster.

 To open the `Scale Machine Type` dialog, navigate to the `Clusters` tab, and select the `Scale Machine Type` icon.

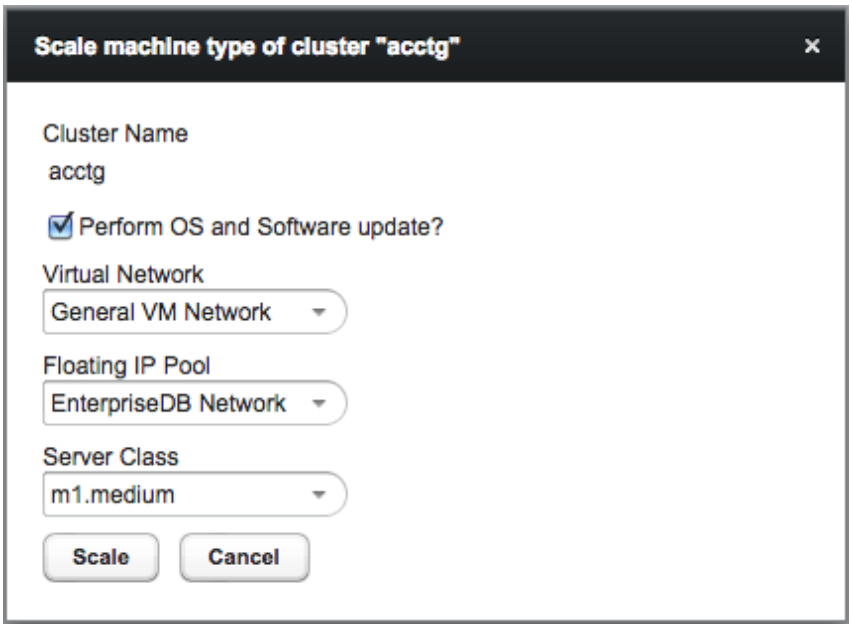


Figure 8.6 - The Scale Machine Type dialog.

Use the fields on the `Scale Machine Type` dialog (shown in Figure 8.6) to specify details about the new cluster:

- Check the box next to `Perform OS and Software update` to instruct EDB Ark to perform a `yum update` whenever the cluster is provisioned.
- Use the `Virtual Network` drop-down listbox to select a network (if applicable).
- Use the `Floating IP Pool` drop-down listbox to specify the name of the IP pool that the cluster will use.
- Use the `Server Class` drop-down listbox to specify the size of the new cluster.

When you click the `Scale` button to start scaling the cluster, EDB Ark will confirm that the scaling is in progress (see Figure 8.7).

Figure 8.7 - The Scale Machine Type dialog.

Before creating the new cluster and (optionally) re-assigning the IP address, EDB Ark will perform a backup of the original cluster. During the process, status indicators in the `PENDING` column of the `Clusters` tab will keep you informed as EDB Ark backs up the original cluster, and initializes the new cluster.

9 Automatic Scaling

When auto-scaling in enabled, EDB Ark monitors the server storage and connection resources in use, and automatically adds additional resources when usage exceeds a user specified percent. Controls on the `Details` panel of the `Clusters` tab makes it easy to adjust the threshold at which EDB Ark automatically scales up resources.

- When the `Data Space Threshold` is reached, EDB Ark adds additional storage space.
- When the `Connection Threshold` is reached, EDB Ark adds replica nodes.

Adding additional replicas to your database cluster increases the number of client connections and queries that each cluster can handle, while maintaining a high-level of overall performance. Each additional replica automatically assumes a share of the read-only workload from incoming queries.

9.1 Adjusting the Automatic Scaling Thresholds

Use the `Auto-Scaling Thresholds` controls (located on the `Details` panel) to adjust the threshold at which EDB Ark automatically scales up cluster resources. To access the `Details` panel, navigate to the `Clusters` tab, and highlight the name of a cluster. Click the `Details` navigation bar on the `Clusters` tab to open the `Details` panel for the cluster (shown in Figure 9.1).

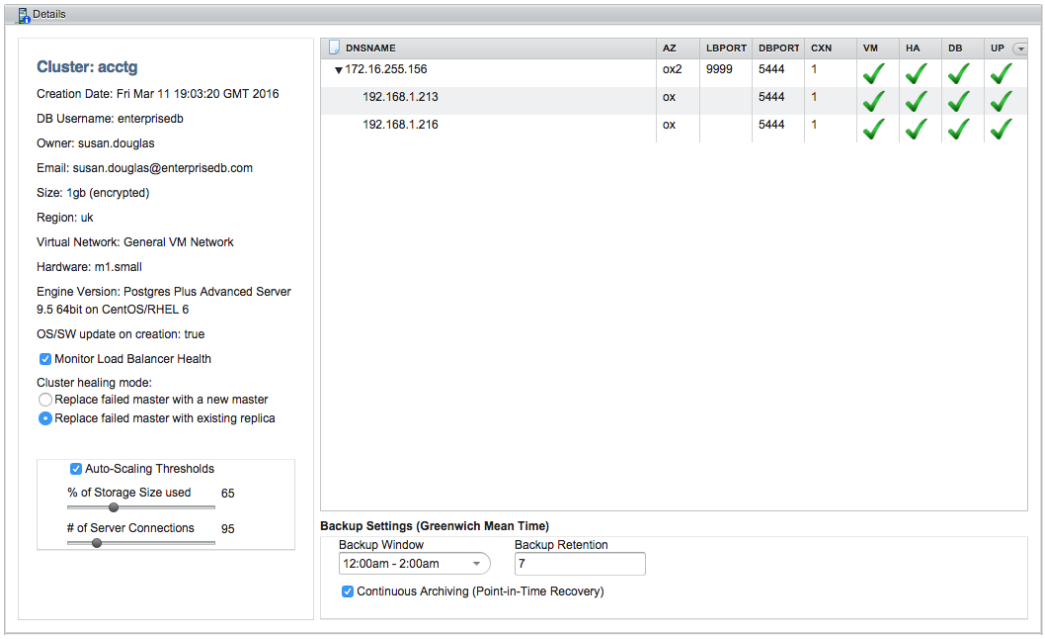


Figure 9.1 - The Details panel on the Clusters tab.

Adjust the `Auto-Scaling Thresholds` sliders to increase or decrease the thresholds at which automatic scaling is invoked. When you modify the values, EDB Ark will display a `New Value Saved` notice, alerting you that your changes have been saved.

Auto-scaling is enabled by default; when auto-scaling is enabled, EDB Ark will automatically increase your data space by 50% when the disk usage exceeds the value specified by the `Data Space Threshold` slider. To disable auto-scaling, un-check the `Auto-Scaling` checkbox.

10 Load Balancing

EDB Ark uses pgPool functionality to implement automatic load balancing. Load balancing increases system performance by distributing client queries to replica nodes, while routing database modifications to the master node. Any modifications to the master node are subsequently propagated to each replica using Postgres streaming replication.

Utilizing Load Balancing

By default, load balancing is enabled on an EDB Ark cluster. To utilize load balancing, you should direct client applications to connect to the load balancing port (by default, `9999`). A cluster's load balancing port number is displayed in the `LBPORT` column on the `Details` pane of the `Clusters` tab of the Ark console.

pgPool may direct the following statement types to *either* a primary or a standby node:

- `SELECT` statements (not listed below)
- `COPY TO`
- `DECLARE`
- `FETCH`
- `CLOSE`
- `SHOW`
- `SET`
- `DISCARD`
- `DEALLOCATE ALL`

When deciding which node a query should be routed to, pgPool checks the transaction log number; if the transaction log number on the standby server is lower than the log number on the master, pgPool routes the statement to the master node. This helps to ensure that the data returned by the query is the most recent available.

In some cases, specific clauses within a query statement will signal pgPool to direct a statement to the master node. In other cases, the transaction type, or order of commands within a transaction can direct a statement to the master node. By default, the following transaction types will always be executed on the master node:

- `SELECT INTO`, `SELECT FOR UPDATE` or `SELECT FOR SHARE` statements
- `SELECT` statements within `SERIALIZABLE` transactions
- `SELECT` statements that follow an `INSERT` statement
- `SET SESSION CHARACTERISTICS AS TRANSACTION… READ WRITE` statements
- `SET transaction_read_only = off` statements
- `EXPLAIN` and `EXPLAIN ANALYZE SELECT` statements
- `START TRANSACTION… READ WRITE` statements

- LOCK commands that are stricter than ROW EXCLUSIVE MODE
- Transactions that start with a BEGIN statement
- The nextval() and setval() sequence functions
- Large objects creation commands

Please Note: If your application uses JDBC, and the autocommit option is set to false, the JDBC driver will include a BEGIN and COMMIT statement with each SELECT statement. To enable load balancing when using the JDBC driver, your application must include a call to setAutoCommit(true).

pgPool directs the following non-query statement types to the master node only:

- INSERT
- UPDATE
- DELETE
- COPY FROM
- TRUNCATE
- CREATE
- DROP
- ALTER
- COMMENT
- PREPARE TRANSACTION
- COMMIT PREPARED
- ROLLBACK PREPARED
- LISTEN
- UNLISTEN
- NOTIFY
- VACUUM

Selectively Enforcing Load Balancing

pgPool does not enforce load balancing for SELECT statements with a leading white space or leading comment. For example, the following statement would be directed to the master node:

```
/*Ignore load balancing*/ SELECT * FROM emp;
```

To enforce load balancing of SELECT statements with leading white space or comments, modify the pgpool.conf file, and set the ignore_leading_white_space parameter to true.

You can also use the black_list and white_list parameters (located in the pgpool.conf file) to instruct pgPool to direct specific statements or functions to the master node. This is useful for cases where a SELECT statement (normally directed to a

replica) calls a function that in turn might modify the database, and so should be directed to the master.

Monitoring Load Balancer Health

By default, EDB Ark monitors the health of the load balancer to ensure that service is not interrupted. If the load balancer (pgpool) should fail while monitoring is enabled, PgPool will be automatically restarted. If the load balancer cannot be automatically restarted, EDB Ark will display a warning sign next to the cluster name on the `Details` panel and send a notification email to the cluster user.

Deselect the `Monitor Load Balancer Health` checkbox (located on the `Details` panel of the `Clusters` tab) to indicate that you do not wish for load balancer health to be monitored and automatically restarted if an interruption in service is detected.

11 Customizing Your Cluster

EDB Ark creates fully-functioning, cloud-based, high-availability database clusters of various sizes complete with replication, load balancing, connection pooling, backup and failover capabilities. An EDB Ark cluster can be defined in minutes without any special database knowledge or skills. This characteristic is greatly appreciated by application developers who want to create robust, data-intensive applications quickly, and who may not have the time, inclination, or skills to otherwise achieve the same results. This type of black box setup was designed to dramatically increase the productivity of developers, DBAs, and system administrators alike.

However, there are many users who, while enjoying the black box benefits described above, prefer to take a more hands-on approach to managing their databases. EDB Ark was also designed with these users in mind.

You can also use supporting components to extend the functionality of your EDB Ark cluster; the following sections provide an overview of how to add an extension to a new or existing cluster.

> The EDB Ark Administrator's console provides an easy way to install and maintain the latest server-related packages. Talk to your system administrator about automatically including supporting components for your cluster when provisioning the database engine.

11.1 Adding an Extension to a New Cluster

Supporting components and utilities can extend the functionality of your Postgres cluster. For example, you may want to consider adding EDB Postgres Enterprise Manager for management, monitoring, and statistical analysis functionality, or PostGIS, to provide support for spatial data types and functions.

An administrative user can use the `Optional Node Packages` field on the `Add Engine` or `Edit Engine` dialog to modify a database engine definition, providing the names of optional rpm packages that will be installed (from the specified repository) during provisioning. All engines created with that definition will contain the new component; the component will be provisioned on each replica as well as on the master node. As each rpm is installed, yum will satisfy the dependencies for the new component.

Packages added via the `Optional Node Packages` field on the master node of the cluster will be provisioned on any standby nodes that are subsequently created. If the package requires manual configuration steps, you will be required to repeat those steps on each node of the cluster; package configurations will not be propagated to standby nodes. If you add a node through cluster operations (such as failover, scaling, or restoring a node from backup), any packages on the new node will also require manual configuration.

For information about modifying a database engine to add a supporting component, see the *EDB Ark Administrative User's Guide*.

12 Database Management

The sections that follow detail some of the tasks that are performed outside of the Ark console's graphical interface:

- Moving an existing database into an EDB Ark cluster
- Connecting an administrative client to a Postgres Server
- Manually modifying configuration parameters
- Stopping and starting the server

Please note: To perform the tasks described in this section, your system must have permission to connect to the cluster on port 22; an OpenStack administrator must modify the security group of the cluster to permit connections.

12.1 Connecting to the Cluster

The following sections will walk you through the process of connecting to a node of an EDB Ark cluster using some of the client applications that are distributed with Advanced Server and PostgreSQL.

12.1.1 Using ssh to Access a Server

EDB Ark creates an `ssh` key when you create a new cluster; each cluster has a unique key. Before connecting to a Postgres instance that resides on the cloud via an `ssh` encrypted connection, you must download the `ssh` key, and adjust the privileges on the key file. Please note that you can only download the ssh key for a cluster that you own.


To download your private key, navigate to the `Clusters` tab, and click the `Download SSH Key` icon. The `Accessing Your Cluster Instance` popup opens (see Figure 12.1).

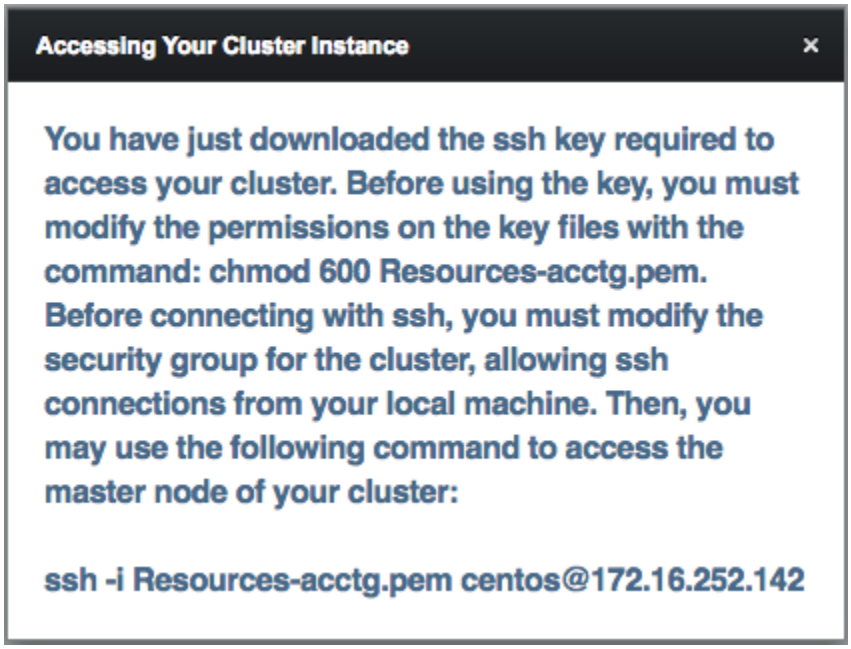


Figure 12.1 – Accessing Your Cluster Instance.

The popup displays the tenant name, the cluster name, the name that you should use when connecting to the cluster, and the IP address to which you should connect.

Before using the private key, you must modify the permissions on the keyfile. Use the following command to restrict file permissions:

```
chmod 0600 ssh_key_file.pem
```


Where *ssh_key_file.pem* specifies the complete path and name of the EDB Ark ssh private key file.

After modifying the key file permissions, you can use ssh to connect to the cluster (see Figure 13.7). Open a terminal window, and specify the location of the ssh key and the connection information provided on the Accessing Your Cluster Instance popup:

```
ssh -i ssh_key_file.pem user_name@host_address
```

Where:

ssh_key_file.pem specifies the complete path and name of the EDB Ark ssh private key file.

user_name specifies the user account that is connecting to the cluster.

host_address specifies the host name of the node to which you wish to connect. You can find the host name in the DNSNAME column, on the Details panel of the Clusters tab in the Ark console.

After connecting via ssh, you can:

- Stop, start, or restart the Postgres server.
- Download and install Postgres extensions.
- Use the PostgreSQL Client Applications.
- Invoke PostgreSQL Server Applications.

  Please note: Postgres Server applications must be invoked by the Postgres cluster owner (identified when creating an EDB Ark cluster as the Master User). If you are using a PostgreSQL server, the default user name is postgres; if you are using Advanced Server, the default user name is enterprisedb. To change your identity after connecting via ssh, use the su command:

  ```
  # sudo su database_user_name
  ```


12.1.2 Connecting with the Postgres Enterprise Manager Client

The Postgres Enterprise Manager (PEM) client provides a powerful graphical interface that you can use to create and manage database objects (and privileges) on a local Postgres installation, or on an EDB Ark cluster node. The PEM client is installed (by default) with the Advanced Server graphical installer; the PEM client is also available for PostgreSQL users. For detailed information about installing the PEM client, visit:

http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager

The PEM client should be installed on and invoked from a local workstation; you must define a separate server connection for each node of an EDB Ark cluster that you wish to manage with the PEM client. Before connecting with the PEM client, an administrator must ensure that the OpenStack service provider allows connections from the host of the PEM client.

To access an EDB Ark cluster with the PEM client, open the client and select `Add Server` from the `File` menu. The `New Server Registration` dialog opens (as shown in Figure 12.2).

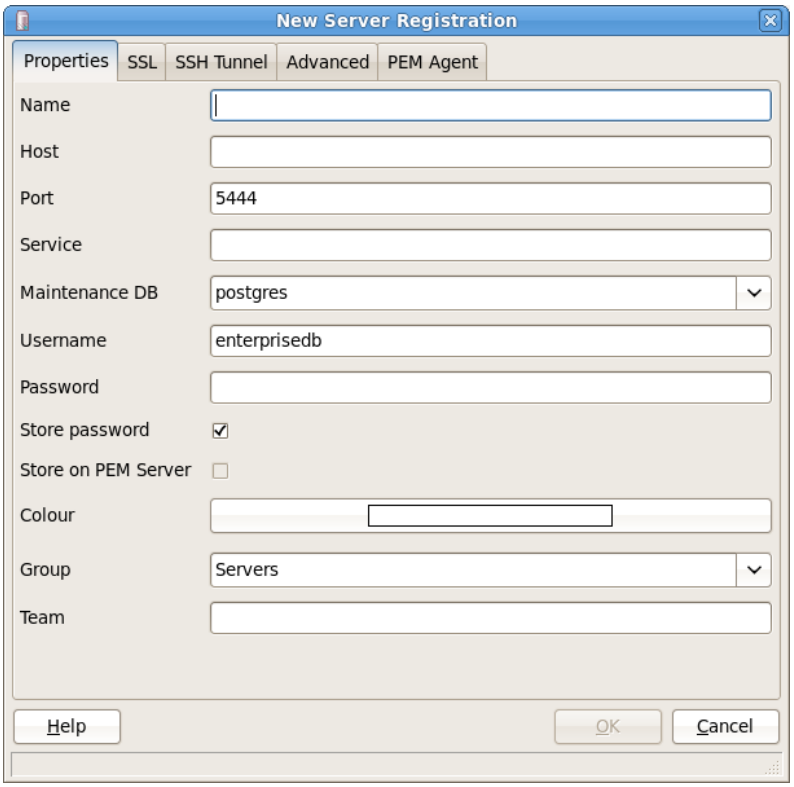


Figure 12.2 - Connecting to the EDB Ark host.

Provide information about the connection in the `New Server Registration` dialog:

- Specify the name of the EDB Ark cluster in the `Name` field.

- Provide the IP address or host name of the master node of the cluster in the `Host` field. You can find the IP address in the `DNSNAME` column on the `Details` panel for the cluster on the Ark console.

- Specify the `Port` through which you wish to connect to the server.

  By default, the only port open for connections from clients (residing outside of the cluster) is port `9999`, on the master node. Port `9999` is the load balancing port, and is best used when performing queries.

  If you are modifying a database or invoking administrative functions, you should connect to the master node's listener port, identified in the `DBPORT` column, on the `Details` panel of the `Clusters` tab. Before connecting to the server's listener port, an OpenStack administrator must modify the cluster's security group to allow connections from the connecting client.

- Select a maintenance database using the drop-down listbox in the `Maintenance DB` field. Select `edb` if you are connecting to an Advanced Server database, or `postgres` if you are connecting to a PostgreSQL database.

- Specify the role name that the PEM client should use when connecting, in the `Username` field.

- Provide the password associated with that role, in the `Password` field.

Click `OK` to connect to EDB Ark; once connected, the server will appear in the tree control in the PEM `Object browser` (shown in Figure 12.3).

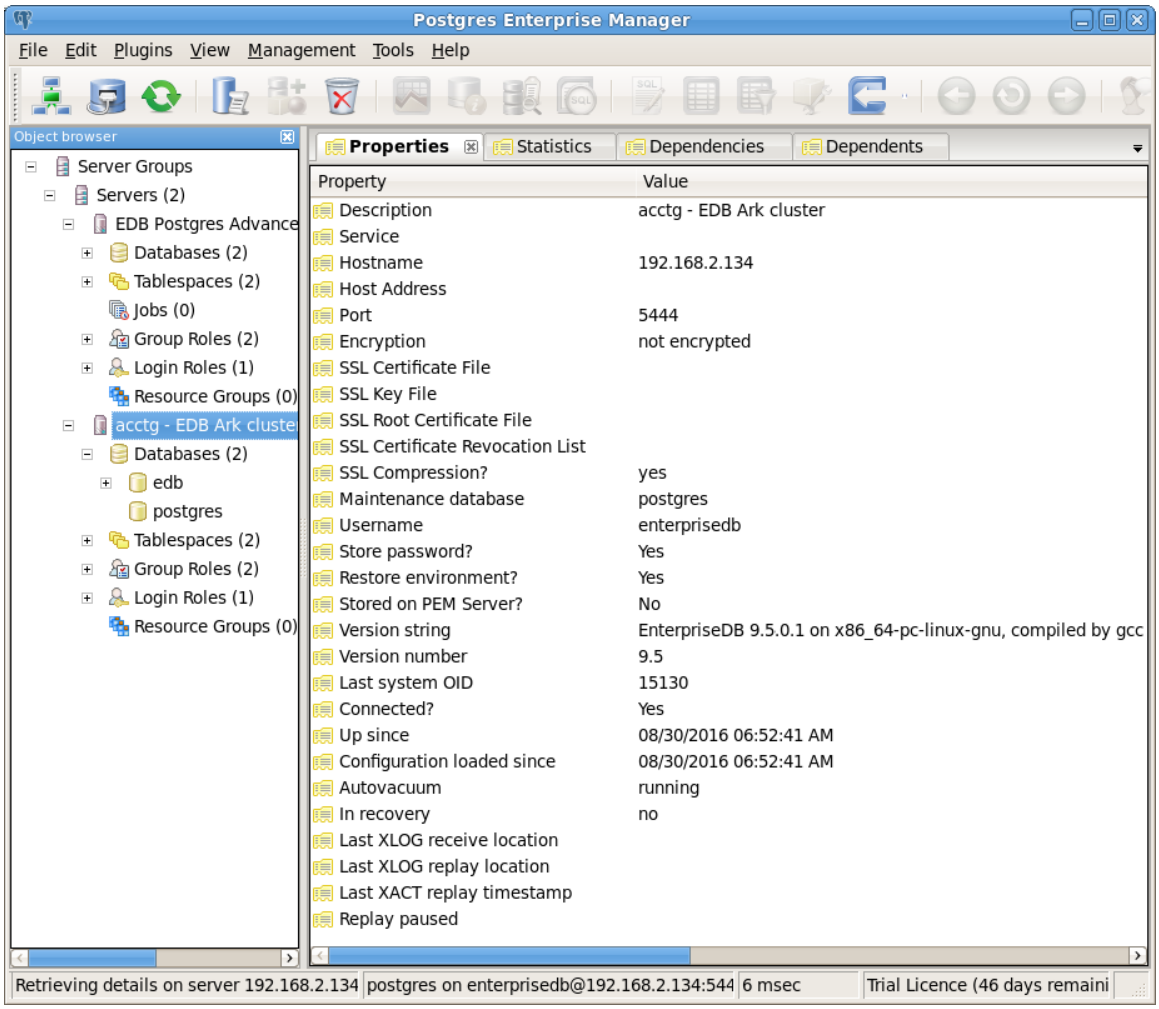


Figure 12.3 - The PEM client window, showing local and EDB Ark servers.

Now, you are ready to use the point-and-click functionality of the PEM client to create and manage database objects that reside on the node of your EDB Ark cluster to which you have connected.

The PEM client offers context-driven help; click the `Help` button provided on each dialog to access online documentation with information about the current dialog.

12.1.3 Connecting to EDB Ark with psql or edb-psql

After connecting to a server hosted on EDB Ark with the Advanced Server `edb-psql` client or PostgreSQL `psql` client, you can invoke SQL commands or use meta-commands to:

- Execute queries
- Insert, update, and delete data
- Create and manage database objects (tables, indexes, views, etc.)
- Create user roles and manage privileges
- Review object and role attributes
- Invoke scripts containing complex (or simple) commands

By default, an EDB Ark cluster is only open to connections via port `9999` on the master node. Port `9999` is a good choice if you are connecting for the purpose of querying the database, but if you are modifying database objects, or performing administrative functions, you should connect directly to the server's listener port.

Some administrative functions, if executed over port `9999`, may be directed to the incorrect node of a multi-node cluster where they may not have the intended effect, or may return an invalid value.

The listener port number is displayed in the `DBPORT` column of the `Details` panel of the `Clusters` tab.

Before connecting to the server's listener port, an OpenStack administrator must modify the security group to allow connections from the host of your client application.

Connecting with edb-psql (or psql) From a Local Workstation

After installing Advanced Server or PostgreSQL on a local workstation, you can use `psql` to perform administrative tasks on an EDB Ark cluster.

To open the `edb-psql` client on an Advanced Server workstation, navigate through the `Applications` (or `Start`) menu to the `Postgres Plus Advanced Server` menu; then, open the `Run SQL Command Line` menu, and select `EDB-PSQL`.

To open a `psql` client on a PostgreSQL workstation, navigate through the `Applications` (or `Start`) menu to the `PostgreSQL` menu, and select `SQL Shell (psql)`.

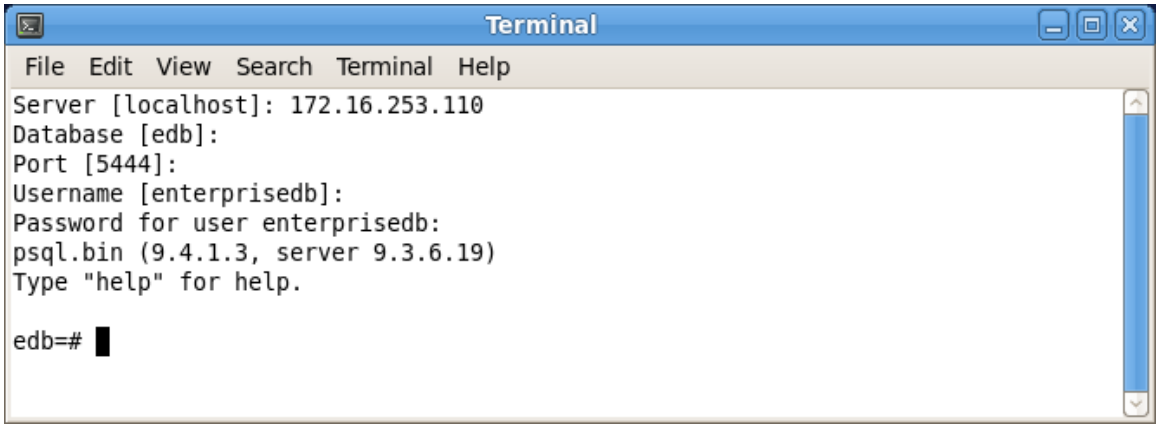


Figure 12.4 - The EDB-PSQL command line utility.

Provide connection information for the server to which you are connecting:

- When prompted for a `Server`, enter the IP address or DNS name of the EDB Ark server. The IP address is displayed in the `DNSNAME` column on the `Details` panel of the `Clusters` tab of the Ark console.

- When prompted for a `Database`, enter the name of the database to which you wish to connect. By default, an Advanced Server cluster is created with a database named `edb`; a PostgreSQL cluster is created with a database named `postgres`.

- When prompted for a `Port`, enter the port on which the server is listening. For database queries, you can use port `9999`; if you are modifying database objects or performing administrative functions, you should use the server's listener port (`5444` for an Advanced Server cluster, `5432` for a PostgreSQL cluster).

- When prompted for a `Username`, enter the role you wish to use when connecting to the server. The name of the database superuser is specified in the `Master User` field when defining an EDB Ark server cluster. By default, the Advanced Server database superuser is `enterprisedb`. The default superuser of a PostgreSQL database is `postgres`.

- When prompted for a `Password`, enter the password associated with that role. The database superuser's password is specified in the `Master Password` field when defining an EDB Ark server cluster.

After connecting, the `edb-psql` (or `psql`) prompt will display the name of the database to which you are connected (as shown in Figure 12.4).

Invoking edb-psql (or psql) on an EDB Ark Server

To use a copy of the psql client that resides on the EDB Ark host, first connect to the cluster using `ssh`:

```
ssh -i/path/ssh_key root@host_name
```

After connecting to the host, assume the identity of the database superuser (or a user with sufficient privileges to invoke the client). On an Advanced Server host, use the command:

```
sudo su enterprisedb
```

On a PostgreSQL host, use the command:

```
sudo su postgres
```

Then, invoke the psql client. On an Advanced Server host, use the command:

```
/usr/bin/edb-psql -d edb
```

On a PostgreSQL host, use the command:

```
/usr/bin/psql -d edb
```

Include the `-d` option to specify the name of the database to which you wish to connect. The session opens as shown in Figure 12.5.

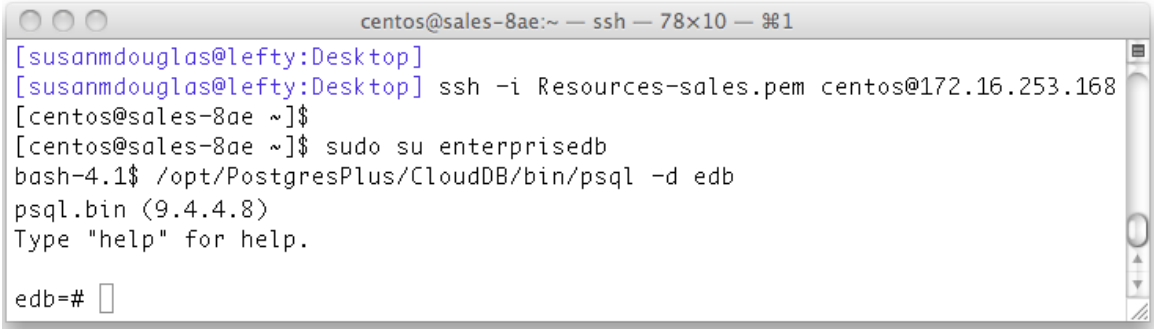


Figure 12.5 - A psql session on the EDB Ark host.

To exit the `psql` client, enter `\q`.

For information about using `psql` and the `psql` meta-commands, please see the Postgres documentation at:

http://www.enterprisedb.com/docs/en/9.4/pg/app-psql.html

For more information about the Postgres SQL commands, please see the Postgres documentation at:

http://www.enterprisedb.com/docs/en/9.4/pg/sql-commands.html

12.2 Moving an Existing Database into a New Cluster

You can use the Postgres `pg_dump` utility to migrate an existing Postgres database (schema, data, and associated database objects) into an EDB Ark cluster.

`pg_dump` creates an archive that contains the commands needed to re-create and populate your existing database. After moving the archive to the EDB Ark, use `pg_restore` to uncompress and play the SQL commands contained in the archive. The following section will walk you through the process of moving a database to EDB Ark using `pg_dump`.

You can also use the `pg_dumpall` utility to move an entire *Postgres* cluster (data, schema information, and roles) to EDB Ark; for detailed information about using `pg_dumpall`, please see the Postgres documentation at:

http://www.postgresql.org/docs/9.5/static/app-pg-dumpall.html

The following example assumes that you have provisioned an EDB Ark cluster and opened a port for SSH connections.

Step One – Navigate into the directory that contains pg_dump

Open a terminal window on the system that contains your Postgres source database, assume the identity of the Postgres superuser, and navigate into the `bin` directory that resides under your Postgres installation directory.

On Advanced Server the path to the `bin` directory is:

```
/usr/ppas-9.x/bin
```

On PostgreSQL, the path to the directory is:

```
/usr/pgsql-9.x/bin
```

Step Two - Create the pg_dump Archive

Use the `pg_dump` utility to create an archive that contains the commands required to recreate a database. When invoking `pg_dump`, include the `-Ft` flag to instruct `pg_dump` to format the output as a `tar` file, and the `-U` flag to specify the name of the database superuser (see Figure 12.6):

```
pg_dump -Ft -U db_superuser db_name > archive_name.tar
```



```
susan@localhost:/home/susan
File  Edit  View  Search  Terminal  Help
-bash-4.1$
-bash-4.1$ ./pg_dump -Ft -U enterprisedb hr >/tmp/hr.tar
Password:
-bash-4.1$
```

Figure 12.6 - Creating the pg_dump archive.

Where:

> *db_superuser* is the name of a Postgres database superuser.
>
> *db_name* is the name of the database that you wish to move to EDB Ark.
>
> *archive_name.tar* is the complete path and name of the archive. Please note that you must have permission to write a file to the location specified.

If prompted, enter the password associated with the database superuser.

Step Three - Move the Archive to EDB Ark

Use the `scp` command to copy the archive to the master server in the EDB Ark cluster; include the `-i` option to specify the location of your `ssh` key (see Figure 12.7):

```
scp -i ssh_key_file file_name user_name@host_name:target
```

```
susan@localhost:/home/susan/Desktop
File  Edit  View  Search  Terminal  Help
[root@localhost Desktop]#
[root@localhost Desktop]# scp -i f890ae43-584c-4e67-b237-18e05254eb86-acctg-0.pem
/tmp/hr.tar centos@172.16.253.110:/tmp
hr.tar                                        100% 8704     8.5KB/s   00:00
[root@localhost Desktop]#
```

Figure 12.7 - Moving the archive to EDB Ark.

Where:

> *ssh_key_file* specifies the pathname of the EDB Ark `ssh` private key file.
>
> *file_name* specifies the archive name.
>
> *user_name* specifies the name used to connect to the master node of the EDB Ark cluster.

host_name specifies the host name of the master node of the EDB Ark cluster; the host name is located on the `Details` panel of the `Clusters` tab in the Ark console.

target specifies the name of the target directory on the EDB Ark host. Including `:/tmp/` at the end of this command directs `scp` to copy the file to the `tmp` directory

Step Four - Connect to EDB Ark with ssh

Use `ssh` to connect to your EDB Ark cluster master node. Provide the user identity of the operating system superuser, and the location of the `ssh` key (on your local host) in the command (see Figure 12.8):

```
ssh -i/path/ssh_key.pem root@host_name
```

Where:

path specifies the location of your EDB Ark `ssh` certificate on the system from which you are connecting.

ssh_key.pem specifies the name of the EDB Ark `ssh` private key file.

host_name specifies the host name of the master node of the EDB Ark cluster; the host name is located on the `Details` panel of the `Clusters` tab in the Ark console.

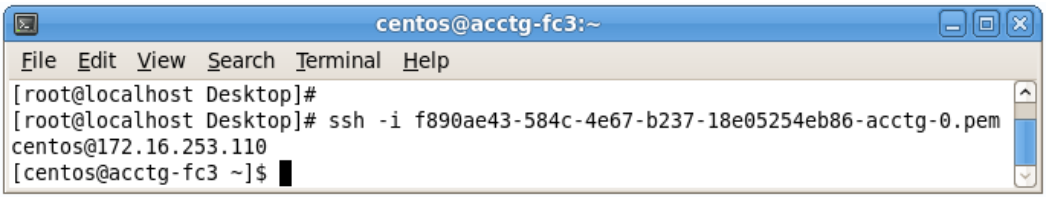


Figure 12.8 - Connecting to EDB Ark with ssh.

Step Five – Navigate into the bin directory on the Ark Host

After connecting, assume the identity of the database superuser and navigate into the directory on the Ark host that contains the `pg_restore` utility (see Figure 12.9). On an Advanced Server host:

```
cd /opt/PostgresPlus/CloudDB/bin
```

On a PostgreSQL host:


```
cd /opt/PostgreSQL/CloudDB/bin
```

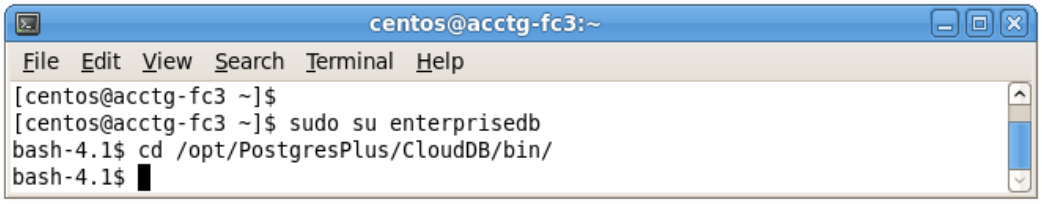


Figure 12.9 – Navigate into the bin directory.

Step Six - Invoke pg_restore on the master server in the EDB Ark cluster

Before invoking the `pg_restore` utility, you must create the target database in the master server; you can use the `createdb` client utility at the command line to create the target:

```
createdb -U db_superuser database_name
```

Where:

`db_superuser specifies` the name of the database superuser. On an Advanced Server cluster, the default is `enterprisedb`; on a PostgreSQL cluster, the default is `postgres`.

`database_name` specifies the name of the database on EDB Ark.

Then, invoke the `pg_restore` utility:

```
pg_restore -Ft -U db_superuser /path/archive_name.tar -d
target_db_name
```

Where:

`db_superuser specifies` the name of the database superuser. On an Advanced Server cluster, the default is `enterprisedb`; on a PostgreSQL cluster, the default is `postgres`.

`path` is the pathname to the archive on the Ark.

`archive_name.tar` is the name of the archived database.

`target_db_name` is the name of the target database on the Ark.

Include:

the `-Ft` flag to specify that the file is an archive

the `-U` flag to specify the name of a database superuser.

the `-d` *`target_db_name`* flag to specify the name of the target database

```
centos@acctg-fc3:~
File  Edit  View  Search  Terminal  Help
bash-4.1$
bash-4.1$ createdb -U enterprisedb hr
bash-4.1$ pg_restore -Ft -U enterprisedb /tmp/hr.tar -d hr
bash-4.1$
```

Figure 12.10 - Restoring the database.

Step Seven - Confirm that the Move was Successful

After performing the restore, you can use the `psql` client (or Advanced Server's `edb-psql`) to connect to the EDB Ark and confirm that the database has been transferred (see Figure 12.11):

```
psql -U database_superuser -d target_db_name
```

Where:

`db_superuser specifies` the name of the database superuser. On an Advanced Server cluster, the default is `enterprisedb`; on a PostgreSQL cluster, the default is `postgres`.

`target_db_name` is the name of the target database.

Use the `\dt` command to view a list of database objects in the current database:


```
centos@acctg-fc3:~
File  Edit  View  Search  Terminal  Help
bash-4.1$
bash-4.1$ ./psql -d hr
psql.bin (9.3.6.19)
Type "help" for help.

hr=# \d
                List of relations
 Schema |    Name    |   Type   |    Owner
--------+------------+----------+--------------
 public | dept       | table    | enterprisedb
 public | emp        | table    | enterprisedb
 public | jobhist    | table    | enterprisedb
 public | next_empno | sequence | enterprisedb
 public | salesemp   | view     | enterprisedb
(5 rows)

hr=#
```

Figure 12.11 - Confirming that the move was successful.

To exit the `psql` client, enter `\q`; to exit the `ssh` session, type `exit` and `Return`.

For more information about using the `psql` client, please see the tutorial, *Connecting to an EDB Ark*. You can access the tutorial through the `Dashboard` tab of the Ark console.

For more information about using PostgreSQL utilities to move an existing database into EDB Ark, please see the documentation at:

http://www.postgresql.org/docs/9.5/static/backup-dump.html

12.3 Manually Modifying Configuration Files

Many of the features of a Postgres server may be influenced by settings specified in configuration files:

- The `postgresql.conf` file determines Postgres server behavior as it pertains to auditing, authentication, file locations, resource usage, query planning, statistic gathering, error handling and more.

- The `pgpool.conf` file determines the behavior of EDB Ark as it pertains to load balancing.

- The `pg_hba.conf` file controls the type of authentication that should be used when a client application connects to an EDB Ark service. By default, the `pg_hba.conf` file is configured to require clients to provide a valid md5-encrypted password.

- The `pg_ident.conf` file contains user mappings for external authentication methods (like LDAP or GSSAPI). Each entry within the `pg_ident.conf` file maps an external user name to his corresponding Postgres user name.

To modify configuration file settings, you must `ssh` to the server, and manually edit the file. By default, on an Advanced Server cluster, configuration files reside in:

```
/etc/ppas-pgpool
```

on a PostgreSQL cluster, configuration files reside in:

```
/etc/pgpool-II-9x
```

To modify a configuration file:

1. `ssh` to the node of the cluster that contains the file you wish to modify. For information about using `ssh` to connect to the server, see Section 12.1.1, *Using ssh to Access a Server*.

2. Use your choice of editor to modify the files.

3. Reload or restart the server. For detailed information about reloading the server, see Section 12.4, *Controlling the Server*.

When you add or remove nodes from a cluster, EDB Ark takes a backup of your `pg_hba.conf` and `pgpool.conf` configuration files. Configuration file backups are appended with the date that the backup was taken and a unique identifier; for example, `pg_hba.conf.20140319-140903` identifies a backup of the `pg_hba.conf` file.

When modifying a configuration file, you should make changes only to those files that *are not* appended with a timestamp and identifier.

12.3.1 Best Practices for Modifying Configuration Files

Please note that changing parameter settings can have unintended consequences, ranging from degraded performance to system crashes. Consequently, we recommend that only an advanced user who accepts these risks, and has experience with both Postgres and cloud environments modify parameter settings.

There are several ways that you can minimize the risks involved when making parameter changes:

- Always make a snapshot backup of your data before making parameter changes. For information about taking a backup, refer to Section 6, *Managing Backups and Recovery*.

- Always setup a test cluster to test parameter changes, to ensure they have the intended effect before deploying them to your production environment. Make the test environment mirror the final target environment as much as possible - this is easy to accomplish by restoring a production backup into a similar size cluster as the original. For more details, see Section 4.3, *Creating a Developer Sandbox*.

- Only change one parameter at a time (or as few as possible when dealing with interdependent settings) and monitor its effect until you are comfortable with the result.

- Make parameter changes on a *copy* of the existing configuration that is in use for the master or replicas. That way, if the parameter changes prove detrimental it will be easy for you to re-apply the original settings. If you are making changes to configuration files through `ssh`, make a backup of the configuration files before making any changes, so they can be easily restored.

When adjusting parameters, be mindful of that fact that the master node in the cluster processes both read and write requests, while the replica nodes in the cluster accept only read requests. You can tune the master node and the replica nodes independently to quickly have an impact (either positive or negative) on your write or read performance.

For more information about modifying Postgres server parameters, please visit:

http://www.postgresql.org/docs/9.5/static/runtime-config.html

12.4 Controlling the Server

You can use your platform-specific service controller or `pg_ctl` to control the EDB Ark service. It is important to note that `pg_ctl` does not alert the Linux service controller to changes in the status of a server.

- The name of the Advanced Server service is `ppas-9.x`.
- The name of the PostgreSQL service is `postgresql-9.x`.

Where *x* specifies the version.

12.4.1 Controlling a Service on CentOS or RHEL 7.x

If your cluster resides on version 7.x of RHEL and CentOS, you can use the `systemctl` command to control the service. The `systemctl` command must be in your search path and must be invoked with superuser privileges. To use the command, open a command line, and enter:

```
systemctl action service_name
```

Where:

service_name

service_name specifies the name of the service.

action

action specifies the action taken by the service command. Specify:

- `start` to start the service.
- `stop` to stop the service.
- `restart` to stop and then start the service.
- `status` to discover the current status of the service.

12.4.2 Controlling a Service on CentOS or RHEL 6.x

On version 6.x of RHEL or CentOS Linux, you can control a service at the command line with the `service` command. The Linux `service` controller mechanism allows you to start and stop the server gracefully. Using the `service` command to change the status of a service allows the service controller to keep track of the server status (the `pg_ctl` command does not alert the service controller to changes in the status of a server).

The command must be in your search path and must be invoked with superuser privileges. Open a command line, and issue the command:

```
service service_name action
```

The Linux `service` command invokes a script (with the same name as the service). If your Linux distribution does not support the `service` command, you can call the script directly by entering:

```
/etc/init.d/service_name action
```

Where:

`service_name`

`service_name` specifies the name of the service.

`action`

`action` specifies the action taken by the service command. Specify:

- `start` to start the service.
- `stop` to stop the service.
- `condstop` to stop the service without displaying a notice if the server is already stopped.
- `restart` to stop and then start the service.
- `condrestart` to restart the service without displaying a notice if the server is already stopped.
- `try-restart` to restart the service without displaying a notice if the server is already stopped.
- `status` to discover the current status of the service.

12.4.3 Using the pg_ctl Utility

You can use the pg_ctl utility to control the server; after connecting to the server with `ssh`, use the `su -` command to assume the identity of the database superuser:

```
sudo su - superuser_name
```

Where *`superuser_name`* is the name of the user that created the cluster.

Then, navigate into the directory that contains the `pg_ctl` utility.

On an Advanced Server cluster, `pg_ctl` resides in:

```
/usr/ppas-9.x/bin
```

On PostgreSQL, `pg_ctl` resides in::

```
/usr/pgsql-9.x/bin
```

Where `x` specifies the installation version.

Then, invoke the `pg_ctl` utility. The syntax is:

```
pg_ctl -D path_to_data_directory action
```

Where:

`path_to_data_directory`

`path_to_data_directory` specifies the path to the database `data` directory. By default, the data directory is located in:

On Advanced Server:

```
/opt/PostgresPlus/CloudDB/data
```

On PostgreSQL:

```
/opt/PostgreSQL/CloudDB/data
```

`action`

`action` specifies the action taken by the `pg_ctl` utility. Specify:

- `start` to start the service.
- `stop` to stop the service.

- `restart` to stop and then start the service.
- `reload` to reload the Postgres configuration changes without stopping the server.
- `status` to discover the current status of the service.

For more information about the options available for the `pg_ctl` utility, please see the Postgres core documentation, available from EnterpriseDB at:

http://www.postgresql.org/docs/9.5/static/app-pg-ctl.html

12.5 Updating Packages on the EDB Ark Cluster

When an update becomes available for a package installed on your cluster, the Ark console will display an alert symbol in the `UP` column of the `Details` panel for the cluster, and in the `UP` column of the `DNSNAME` table adjacent to the node that requires an update (see Figure 12.12):

DNSNAME	AZ	LBPORT	DBPORT	CXN	VM	HA	DB	UP
▼172.16.255.29	ox2	9999	5444	1	✓	✓	✓	⚠

Figure 12.12 – The DNSNAME table.

The column displays:


A green checkmark if all of the packages on your cluster are up-to-date.


A yellow alert symbol if non-critical updates are available.


A red error symbol if critical security updates are available.


A grey question mark if the package status is undetermined.

The overall cluster status (displayed in the top section of the `Clusters` tab) is based on the values of the nodes within the cluster.

- If all of the nodes within the cluster are up-to-date, the `UP` column displays a green checkmark.
- If one or more nodes require a non-critical update, the `UP` column displays a yellow alert symbol.
- If one or more nodes require a critical update, the `UP` column for the cluster displays a red error symbol.
- If one or more nodes have an unknown package status, the `UP` column for the cluster displays a grey checkmark.


You can use the `Upgrade` icon (located on the `Clusters` tab) to perform a `yum update` on each node within the cluster. Note that if the `yum update` command fails during the upgrade process, EDB Ark will terminate the process and `yum update` will not be run on any remaining nodes, leaving the cluster partially upgraded.

The `yum update` command will update all installed packages to the most recent version available of the same release (i.e., if you are running a 9.2 server, yum will update your database server to the most recent version of 9.2).

The upgrade does not facilitate upgrading the server to a newer *major* release of Postgres. For example, to migrate between server version 9.3 and 9.4, you must perform a manual upgrade. For more information about performing a major version upgrade, please see Section 12.5.1, *Performing a Major Version Upgrade*.

Before performing the upgrade, EDB Ark will perform a backup. During the upgrade process, all clients will be disconnected from the server. The upgraded server will retain the IP address used by the original server. When the upgrade has completed, clients may once again connect.

After performing a `yum update`, the node will be rebooted, initiating any kernel updates required. When the update completes, EDB Ark will send an email notification that contains a list of the updated packages.

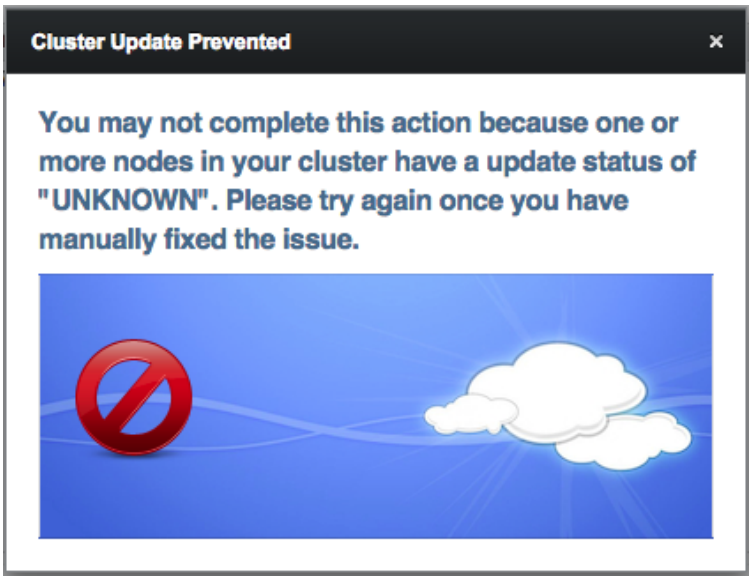


Figure 12.13 – Cluster Update Prevented error.

If one or more nodes in your cluster are currently displaying an unknown status, EDB Ark will display the error message shown in Figure 12.13. You must correct the problem that is causing the unknown status before EDB Ark can perform an automatic upgrade.

12.5.1 Performing a Major Version Upgrade

You can use the Postgres `pg_dump` and `pg_restore` utilities to upgrade a running cluster to a new major version of Postgres. To upgrade, use the `pg_dump` utility to make a copy your database, create a cluster that is running the upgraded version of Postgres, and then use `pg_restore` to install the database into the new cluster.

Please note that during the upgrade process your database will be offline.

The steps for upgrading to a new server version are:

1. Halt all client activity against the database, to ensure that the database is in a stable state and that no transactions are lost.

2. Invoke `pg_dump`, backing up the old cluster.

3. Create a new cluster, selecting the new major version for the database server in the `Create a New Server Cluster` dialog.

4. Perform a `pg_restore` on the new master using the backup created with `pg_dump`.

   For detailed information about using `pg_dump` and `pg_restore` to move an existing database into a new cluster, see Section 12.2.

5. Reassign the IP address from the old cluster to the new cluster.

13 Troubleshooting

This section provides helpful troubleshooting information; if you still have unanswered questions after reviewing this section, you can also find solutions through EnterpriseDB:

If you have purchased support, you can log a support ticket:

in the Customer Portal: http://www.enterprisedb.com/support

via email: mailto:support@enterprisedb.com

or by phone: +1-732-331-1320 or 1-800-235-5891 (US Only)

If you have not purchased support, and would like to, view your support options at:

http://www.enterprisedb.com/cloud-database/support

You are always welcome to log an issue via email; when time permits, our customer support experts will respond to inquiries from customers that have not purchased support.

You can also find free help on a wide variety of topics in the EnterpriseDB User Forums, at:

http://forums.enterprisedb.com/forums/show/21.page

Postgres documentation and helpful tutorials are available from the EDB Ark bookshelf, located on the `Dashboard` tab of the management console.

13.1 Frequently Asked Questions

Question: How do I use the pgAdmin client to connect to my EDB Ark cluster?

You can use the same steps to connect a pgAdmin client to an EDB Ark cluster as when connecting to a cluster with a Postgres Enterprise Manager client. For detailed information about using the Postgres Enterprise Manager graphical client or pgAdmin to connect to a cluster, see Section 12.1.2, *Connecting with the Postgres Enterprise Manager Client*.

Problem: Logging into the Console sometimes takes a long time.

This can be attributed to delays in the connection time to the backend server. When you log in, the Console Manager must pass your credentials to the server to log in; any delays at the service provider may slow your connection time.

Problem: I am attempting to connect to my cluster, but don't know my default database name.

- The name of the default database in an Advanced Server cluster is `edb`.
- The name of the default database in a PostgreSQL cluster is `postgres`.

Problem: unable to connect to the load balancing port (9999).

If you are having difficulty connecting to the load balancing port, you should:

- Make sure you are connecting to the master server's DNS name, rather than a replica's DNS name; the load balancer resides on the master node of an EDB Ark cluster.
- Make sure that your client application is providing an MD-5 encrypted password when attempting to connect to the load balancing port. The `username:password-md5` combination is stored in `pgpool_passwd.conf`, and is automatically updated when a user changes password, or when a new user is created.

Problem: pgpool keeps issuing the following error:
make_persistent_db_connection: s_do_auth failed.

pgpool attempts to connect to each node to perform replication lag checking. This happens unconditionally if pgpool is configured in a master-slave mode and streaming replication is being used (which is the case for EDB Ark). The pgpool community has been alerted to this behavior; please ignore these messages.

Question: How do I stop the Postgres server on a cluster node without triggering a failover process?

To safely stop a Postgres server without triggering failover, you can use either the `service` command or the `pg_ctl` utility. For more information, see Section 12.4.

Problem: I am attempting to connect to my Advanced Server database with the psql client, and am getting the error:
(03/23/2012 13:36:53)-> psql --host=192.0.43.10 -p 9999 -U enterprisedb
Password for user enterprisedb: psql: FATAL: database "postgres" does not exist

The `psql` client expects the default database to be named `postgres`; the `edb-psql` client expects the default database to be named `edb`. If you attempt to connect to an Advanced Server cluster with the `psql` client without specifying the name of the database to which it should connect, the client will fail to connect.

You can include the `-d` or `--dbname` flag, followed by the database name when invoking either client to specify the database to which the client should connect.

Question: I'm trying to drop a database from a cluster, but I am getting an error that there are open sessions. There are no clients connected. How can I terminate any leftover backend sessions?

It may be that pgpool is retaining a connection to the database. You can use the `pg_cancel_backend()` or `pg_terminate_backend()` functions to selectively close connections to the database you wish to drop.

Question: Why do I have to restart pgPool before it will recognize new users that I've added to the database server?

pgPool does not check for new Postgres users. EDB Ark has a periodic update process that updates the user list every 20 seconds; if the update process identifies a new user, it sends a `reload` signal to the pgPool process. After the reload, pgPool will allow new users to login.

Instead of reloading, simply waiting for 20 seconds between the `CREATE USER` statement and the `CREATEDB` statement should solve the problem.

Question: Why are scheduled backups not working?

If you invoke the `pg_start_backup()` function before performing a manual backup your database, you must remember to invoke the `pg_stop_backup()` function when the backup has completed, or EDB Ark scheduled backups will fail.

13.2 The EDB Ark Email Notification System

EDB Ark invokes an email notification system that will alert you if your cluster changes or encounters a problem. Email notifications are sent to the address used to log in to the management console.

EDB Ark will send an email:

- When a new cluster is created.
- If a server stops (or is terminated).
- When a replica is added to a cluster.
- When memory is scaled up.
- When failover is invoked on a master or a replica.
- If a backup fails.
- If the password associated with your user account changes.

The `Notification Email` field (on the `User` tab) allows you to change the notification email associated with your user account; for more information, see Section 3.4, *The User Tab*.

14 EDB Ark API Support

EDB Ark provides JSON-compatible support for the API described in this chapter.

EDB Ark uses token-based authentication. For information about retrieving a token, see Section 14.1.21. After receiving a token string, you should include the string in the `X-AuthInfo-Token` header when calling any resource except `/tokens`.

14.1 Resources

When calling a resource, prefix the resource name with the URI:

```
https://<ppcd_host_address>/api/v2.0
```

EDB Ark supports the request types shown below for the resources listed:

Resource Name	GET	POST	PUT	DELETE
/admin/logs	GET			
/admin/wall	GET		PUT	DELETE
/clusters	GET			
/dbengines	GET	POST		
/dbengines/*engine_id*	GET		PUT	DELETE
/options/backup-windows	GET			
/options/server-classes/*tenant_name/*?engineId={*id*}	GET			
/options/types	GET			
/options/version/*type*	GET			
/options/vpcids/ *tenant_name*	GET			
/owners	GET			
/owners/*tenant_name*/backups	GET	POST		
/owners/*tenant_name*/backups/*backup_id*	GET			DELETE
/owners/ *tenant_name* /clusters	GET	POST		
/owners/ *tenant_name* /clusters/*cluster_name*	GET		PUT	DELETE
/owners/ *tenant_name* /clusters/*cluster_name*/events	GET			
/owners/ *tenant_name* /clusters/*cluster_name*/statistics?start=*start_time*&end=*end_time*	GET			
/serverimages	GET	POST		
/serverimages/*image_id*	GET		PUT	DELETE
/token		POST		DELETE
/users	GET			
/users/*user_id*	GET		PUT	
/users/*user_id*/notifications	GET			

Each call to a resource will return a response code. For a complete list of response codes, see Section 14.2.

14.1.1 /admin/logs

Use the `/admin/logs` resource to download the server log files. You must be an administrator to use this resource. The following is an example of a GET call to `/admin/logs`:

```
curl -H "X-Auth-Token: ostoken"
https://ppcd_host_address/api/v2.0/admin/logs -o logs.zip
```

The file containing the console logs will be saved to the location specified by the calling application.

A successful call to this resource will return a resource code of `200`.

14.1.2 /admin/wall

Use the `/admin/wall` resource to manage the information displayed on the console wall. You must be an administrator to use this resource.

A call to GET this resource could return the following:

```
{
    "wallMessage":" The console will be unavailable Sunday
morning due to scheduled maintenance."
}
```

To update the wall message, pass a new value for the `wallMessage` field with a PUT request.

Use a DELETE request to remove the console wall message.

A successful call to this resource will return a resource code of `200`.

14.1.3 /clusters

An administrator can use the `/clusters` resource to retrieve a list of all clusters; the output will include information about the master instance of each cluster. If there is only one cluster, a GET request returns information about the single instance:

```
{
    "instance": {
       "autoScale": "true",
       "availabilityZone": "ox",
       "backupRetention": "1",
       "backupWindow": "12:00am - 2:00am",
        ...
       }
}
```

If there are multiple clusters, a GET request returns a list that contains information about each instance:

```
{
    "instance": [
    {
       "autoScale": "true",
       "availabilityZone": "ox",
       "backupRetention": "1",
       "backupWindow": "12:00am - 2:00am",
        ...
    },
    {
    "instance": {
       "autoScale": "true",
       "availabilityZone": "ox",
       "backupRetention": "1",
       "backupWindow": "12:00am - 2:00am",
        ...
    },
]
}
```

See Section 14.1.16 for a complete listing of the information returned by `/clusters`.

A successful call to this resource will return a resource code of `200`.

14.1.4 /dbengines

The /dbengines resource allows you to retrieve information about database engines or create a database engine. A GET request returns a list of the currently defined database engines:

```
{
    "dbEngine":[
      {
          "engineId": "PG_94",
          "eol": "false",
          "id": "1",
           ...
       },
       {
          "engineId": "PPAS_95",
          "eol": "false",
          "id": "2",
           ...
       }
    ]
}
```

If the GET request is issued using a token retrieved by an administrative user, the list will include database engines that have been disabled (`eol = true`). If the list is retrieved by a non-administrative user, disabled engines will be omitted. For a complete list of the information returned by /dbengines, please see Section 14.1.5.

An administrator may use /dbengines to create a new database engine. When using a POST request to create a new database engine, pass the following fields:

```
{
    "engineId": "PPAS_95",
    "eol": "false",
    "name": "EDB Postgres Advanced Server 9.5 64bit",
    "optionalPkgs": "",
    "repos": {
    "url":http://user:pwd@yum.enterprisedb.com/9.5/
redhat/rhel-$releasever-$basearch
    },
    "requiredPkgs": "ppas95-server ppas-pgpool-34
     ppas95-pgpool34-extensions",
    "serverImage": {
       "id": "3"
    },
    "type": "advanced_server",
    "version": "9.5"
}
```


When passing in repository information, only the repository URL is required.

When passing in information about the server image, pass in only the `id` number of the server image (returned by the `/serverimages` resource); the server image must already exist in the Ark console, or the request will fail.

A successful call to this resource will return a resource code of `204`.

14.1.5 /dbengines/*engine_id*

The /`dbengines`/*`engine_id`* resource allows you to retrieve information about a specific database engine, modify a database engine definition, or delete an engine. Pass the *`engine_id`* with a GET request to retrieve the following information:


```
{
    "engineId": "PG_94",
    "eol": "false",
    "id": "1",
    "name": "PostgreSQL 9.4 64bit",
    "optionalPkgs": "",
    "repos": {
        "id": "30",
        "url": http://yum.postgresql.org/9.4/redhat
        /rhel-6-x86_64/pgdg-redhat94-9.4-1.noarch.rpm
             },
    "requiredPkgs": "postgresql94-server.x86_64
     pgpool-II-94.x86_64 postgresql94-jdbc.x86_64",
    "serverImage": {
        "id": "2",
        "imageId": "a8ed57dd-9a34-40ca-977b-ce3af9ad3745",
        "initialUser": "centos",
        "serverDescription": "CentOS 6.6",
        "serverId": "centos_6.6"
                    },
    "type": "postgres",
    "version": "9.4"
}
```


`engine_id` is the unique identifier of a database engine, provided in the `id` field.

Provide the *`engine_id`* with a PUT request to update the definition of a specific database engine. Please note that you cannot change the following attributes:

- `engineId`
- `type`
- `version`

Provide the *`engine_id`* with a DELETE request to remove the definition of a database engine.

You must be an administrator to delete or modify a database engine. A successful call to this resource will return a resource code of `204`.

14.1.6 /options/backup-windows

Use the `/options/backup-windows` resource to retrieve a list of backup windows; a GET request to the resource returns information in the following format:

```
{
    "backupWindows": [
        "12:00am - 2:00am",
        "2:00am - 4:00am",
        "4:00am - 6:00am",
        "6:00am - 8:00am",
        "8:00am - 10:00am",
        "10:00am - 12:00pm",
        "12:00pm - 2:00pm",
        "2:00pm - 4:00pm",
        "4:00pm - 6:00pm",
        "6:00pm - 8:00pm",
        "8:00pm - 10:00pm",
        "10:00pm - 12:00am"
  ]
}
```

A successful call to this resource will return a resource code of `200`.

14.1.7 /options/ip-pools/*tenant_name*

Pass in the name of the tenant when calling this resource to retrieve a list of IP pools available to the tenant:

```
{
    "ipPools": [
        "Sales East",
        "Mgmt"
  ]
}
```

A successful call to this resource will return a resource code of `200`.

14.1.8 /options/server-classes/*tenant_name/*?engineId=*id*

Pass in the name of the tenant when calling this resource to retrieve a list of server classes available to the tenant; optionally, include the name of an engine to filter the result set. A call to this resource returns a list of server classes in the form:

```
{
    "serverClasses": [
        "m1.small",
        "m1.medium",
        "m1.large",
        "m1.xlarge",
        "d1.large",
        "m1.tiny",
        "d1.small",
        "d1.xlarge",
        "d1.tiny",
        "d1.medium"
  ]
}
```

tenant_name is the unique identifier of the tenant.

id is an integer value that represents the database engine; use the /dbengines resource to retrieve a list of engine ids. The *id* parameter is optional. If supplied, the specified engine will be used to filter the list of available server classes meeting the minimum requirements of the backing VM image.

GET - Gets a list of the available server classes for the given tenant (which must match the tenant to which the authentication token is scoped).

A successful call to this resource will return a resource code of 200.

14.1.9 /options/types

Use the `/options/types` API to return a list of available database types in the form:

```
{
    "types": ["postgres","ppas"]
}
```

GET - returns a list of the available database types.

A successful call to this resource will return a resource code of `200`.

14.1.10 /options/versions/*type*

Use the `/options/version/`*`type`* API to return a list of database versions available for the specified type:

```
{
    "versions": ["9.4","9.5"]
}
```

A successful call to this resource will return a resource code of `200`.

14.1.11 /options/vpcids/*tenant_name*

Pass in the name of the tenant when calling this resource to retrieve a list of virtual network IDs available to the tenant:

```
{
    "vpcids": "General VM Network"
}
```

A successful call to this resource will return a resource code of `200`.

14.1.12 /owners

Use the `/owners` resource to retrieve a list of tenants that may be accessed by the user that retrieved the security token. A GET request to the resource returns:

```
{
    "owners": [
        "admin",
        "acctg",
        "sales"
  ]
}
```

The user specified by the `service.account.id` property (in the `ppcd.properties` file) has access to all of the tenants that the console understands.

A successful call to this resource will return a resource code of `200`.

14.1.13 /owners/*tenant_name*/backups

Use the `/owners/`*`tenant_name`*`/backups` resource to retrieve a list with information about the current cluster backups.

```
{
  "backup": [
      {
      "backupType": "Manual",
      "capacity": "2",
      "clusterUuid":
      "0e6d9b08-19f1-4d15-8b80-96b186a7dcf0",
      "ended": "2016-01-18T23:35:05.497Z ",
      "started": "2016-01-18T18:15:06.497Z ",
      "tenant": "Resources",
      "yumUpdate": "true"
       ...
       },
       {
      "backupType": "Manual",
      "capacity": "2",
      "clusterUuid":
      "0e6d9b08-19f1-4d15-8b80-96b186a7dcf0",
      "ended": "2016-01-18T23:35:05.497Z ",
      "started": "2016-01-18T18:15:06.497Z ",
      "tenant": "Resources",
      "yumUpdate": "true"
       ...
      }
```



```
    ]
  }
```

For a complete list of the information returned for each backup, see Section 14.1.14.

tenant_name

tenant_name is the name of the tenant in which the cluster resides.

Provide the name of the tenant in which a cluster resides with a GET request to retrieve information about all backups for the specified tenant. If there are no backups, the server returns response code `204`.

Use a POST request to create a backup for a cluster. When creating a backup, only the cluster identifier is required; the cluster identifier is passed in as `clusterUuid`:

```
{
    "clusterUuid": "0e6d9b08-19f1-4d15-8b80-96b186a7dcf0",
    "notes": "This backup was taken right before closing."
}
```

The `notes` field is optional. A successful call to this resource returns response code `202`.

14.1.14 /owners/*tenant_name*/backups/*backup_id*

The `/owners/tenant_name/backups/backup_id` resource allows you to retrieve information about a cluster backup or delete a specific backup. A GET request to the resource returns:


```
{
    "backup": {
        "backupType": "Manual",
        "capacity": "2",
        "clusterUuid":
        "0e6d9b08-19f1-4d15-8b80-96b186a7dcf0",
        "continuousArchiving": "false",
        "dbEngine": {
            "engineId": "PG_94",
            "eol": "false",
            "id": "1",
            "name": "PostgreSQL 9.4 64bit",
            "optionalPkgs": "",
            "repos": {
                "id": "30",
                "url": http://yum.postgresql.org/9.4/redhat/
                 rhel-6-x86_64/pgdg-redhat94-9.4-1.noarch.rpm
            },
            "requiredPkgs": "postgresql94-server.x86_64
             pgpool-II-94.x86_64 postgresql94-jdbc.x86_64",
            "serverImage": {
                "id": "2",
                "imageId":
                "a8ed57dd-9a34-40ca-977b-ce3af9ad3745",
                "initialUser": "centos",
                "serverDescription": "CentOS 6.6",
                "serverId": "centos_6.6"
            },
            "type": "postgres",
            "version": "9.4"
        },
        "encrypted": "false",
        "encryptionKey": "",
        "ended": "2016-01-18T23:35:05.497Z ",
        "engineVersion": "PostgreSQL 9.4 64bit",
        "id": "6f9cc175-2f30-45e9-8a40-50c144117162",
        "masterUser": "postgres",
        "notes": "",
        "owner": "some.user",
        "signature": "upgradecluster",
        "started": "2016-01-18T18:15:06.497Z ",
        "tenant": "Resources",
        "yumUpdate": "true"
    }
}
```


tenant_name

tenant_name is the name of the tenant in which the cluster resides.

backup_id

backup_id is the unique identifier of the backup provided in the `id` field.

Provide the name of the tenant in which a cluster resides and a backup identifier when calling /`owners`/*tenant_name*/`backups`/*backup_id* to retrieve information about a specific backup.

When sending a DELETE request, provide the name of the tenant in which a cluster resides and a backup identifier. If the DELETE is successful, this resource returns code `202`.

14.1.15 /owners/*tenant_name*/clusters

Use the `/owners/tenant_name/clusters` resource to retrieve cluster details about all of the clusters that reside within the specified tenant or to create a new cluster.

```
{
  "instance":[
  {
      "backupRetention": "2",
      "backupWindow": "4:00pm - 6:00pm",
      "clusterName": "acctg",
      "continuousArchiving": "false",
      "dbEngine": {
          "id": "1"
      },
      "encrypted": "false",
      "hardware": "m1.small",
      "ipPool": "Sales East"
       ...
  },
  {
      "backupRetention": "2",
      "backupWindow": "4:00pm - 6:00pm",
      "clusterName": "admin",
      "continuousArchiving": "false",
      "dbEngine": {
          "id": "2"
      },
      "encrypted": "false",
      "hardware": "m1.small",
      "ipPool": "Mgmt",
       ...
      }
  ]
}
```

tenant_name

tenant_name is the name of the tenant.

Pass the tenant name with a GET request to retrieve a list of all the clusters in the tenant. The list will include all the master instances of the clusters. Calls to this resource return a response code of `204` if there are no clusters for that tenant, or `404` if the tenant does not exist.

Pass the cluster details with a POST request to create a new cluster. When you create a new cluster, the resource responds with an HTTP header that contains an URL that

represents the location of the new cluster. A successful call to this resource returns a `201`.

To use a POST request to clone a cluster, pass in the following fields.

```
{
    "clusterName": "new_cluster_name",
    "continuousArchiving": "false",
    "encrypted": "false",
    "fromCluster": source_cluster_id,
    "hardware": "m1.small",
    "ipPool": "IP_pool_name",
    "vpcid": "VM_Network",
    "yumUpdate": "true"
}
```

Note that `fromCluster` refers to the `clusterUuid` of the source cluster, and is not a value stored after this call.

To create a cluster from a backup, pass in the same fields, but specify `fromBackup` instead of `fromCluster`, and pass in the backup identifier:

```
"frombackup": "backup_id"
```

A successful call to this resource will return a resource code of `200`.

14.1.16 /owners/*tenant_name*/clusters/*cluster_name*

When calling `/owners/tenant_name /clusters/cluster_name`, pass in the name of the tenant in which the cluster resides and the name of the cluster; the resource will return the following information about the specified cluster:


```
{
    "instance": {
        "autoScale": "true",
        "availabilityZone": "ox",
        "backupRetention": "1",
        "backupWindow": "12:00am - 2:00am",
        "caState": "",
        "clusterKey": "-----BEGIN RSA PRIVATE KEY-----
         ...
         -----END RSA PRIVATE KEY-----",
        "clusterKeyName": "Resources-onenode",
        "clusterName": "onenode",
        "clusterState": "2",
        "clusterUuid":
        "230b3b81-2dd7-4b21-a221-79cfff2b22e7",
        "connectionThreshold": "95",
        "connections": "1",
        "continuousArchiving": "false",
        "cpuLoad": "9",
        "dataThreshold": "95",
        "creationTime": "2015-10-27T13:06:41.798Z",
        "dataThreshold": "65",
        "dbEngine": {
            "engineId": "PG_94",
            "eol": "false",
            "id": "1",
            "name": "PostgreSQL 9.4 64bit",
            "optionalPkgs": "",
            "repos": {
                "id": "30",
                "url": "http://yum.postgresql.org/9.4/
                 redhat/rhel-6-x86_64/pgdg-redhat94-9.4-
                 1.noarch.rpm"
            },
            "requiredPkgs": "postgresql94-server.x86_64
             pgpool-II-94.x86_64 postgresql94-jdbc.x86_64",
            "serverImage": {
                "id": "2",
                "imageId":
                "a8ed57dd-9a34-40ca-977b-ce3af9ad3745",
                "initialUser": "centos",
                "serverDescription": "CentOS 6.6",
                "serverId": "centos_6.6"
            },
            "type": "postgres",
```



```
                "version": "9.4"
            },
            "dbName": "postgres",
            "dbPort": "5432",
            "dbState": "2",
            "dnsName": "172.16.252.27",
            "encrypted": "false",
            "engineVersion": "PostgreSQL 9.4 64bit",
            "freeDataSpace": "1882272",
            "hardware": "m1.small",
            "id": "ed68c50c-26ca-45d1-bded-c07334161dce",
            "imageId": "a8ed57dd-9a34-40ca-977b-ce3af9ad3745",
            "instanceState": "RUNNING",
            "iops": "0",
            "ipPool": "Sales East",
            "lbPort": "9999",
            "masterUser": "postgres",
            "monitoringLB": "true",
            "notificationEmail": "first.last@enterprisedb.com",
            "numberOfNodes": "1",
            "optimized": "false",
            "owner": "first.last",
            "pendingModifications": "",
            "port": "22",
            "primaryFailoverToReplica": "false",
            "privateIp": "192.168.1.16",
            "profile": "m1.small",
            "publicIp": "172.16.252.27",
            "readonly": "false",
            "region": "uk",
            "securityGroup": "jclouds-Resources-onenode",
            "storage": "1.0",
            "tenant": "Resources",
            "usedDataSpace": "40484",
            "versionNum": "020000",
            "vpcid": "General VM Network",
            "yumStatus": "2",
            "yumUpdate": "true",
            "zone": "ox"
        }
    }
```


tenant_name

tenant_name is the name of the tenant in which the cluster resides.

cluster_name

cluster_name is the name of the cluster.

Pass the tenant name and the cluster name when using a GET request. The information will include the master and standbys of the cluster. A GET request may include the following values for the specified cluster:

Parameter	Description
pendingModifications	A user-readable value such as Initializing or Backup in Progress. An empty value means there are no pending modifications.
dbState	dbState may be 0 (stopped), 1 (starting), or 2 (running).
clusterState	clusterState may be 0 (stopped), 1 (starting), or 2 (running).

Please note: the values returned for the `dbState` and `clusterState` fields correspond with the information displayed in the `DB` and `HA` columns in the Ark console.

Use this resource with a PUT request to change cluster settings; when you change a cluster, specify the relevant keyword and the new value. A successful call to this resource returns `202` (accepted) for asynchronous events, for synchronous events, `204`.

You can use the properties listed below with a PUT request to update cluster settings.

Keyword	Description	Example
numberOfNodes	Scale up (add replicas). The number must be greater than the current number of nodes in the cluster.	`{"numberOfNodes":"4"}`
removeNode	Remove replica(s) from a cluster.	`{"removeNode" : "id1"}` or `{"removeNode" : ["id1", "id2"]}`
primaryFailoverToReplica	Change primary failover type. The value must be 'true' or 'false,' ignoring case.	`{"primaryFailoverToReplica":"false"}`
autoScale	Turn auto scale on or off.	`{"autoScale" : "false"}`
backupRetention	Set the backup retention.	`{"backupRetention" : "4"}`
backupWindow	Set the backup window:	`{"backupWindow" : "10:00am - 12:00pm"}`
connectionThreshold	Set the connection threshold.	`{"connectionThreshold" : "60"}`
dataThreshold	Set the cpu threshold.	`{"dataThreshold" : "80"}`
upgrade	Perform update on cluster. Value can be `true` or false; passing in false does nothing and a 204 is returned.	`{"upgrade" : "true"}`
monitoringLB	Turn load balancer monitoring on/off.	`{"monitoringLB" : "false"}`
notificationEmail	Change the notification email.	`{"notificationEmail" : "name@example.com"}`
owner	Change the cluster owner.	`{"owner" : "id"}`
continuousArchiving	Turn on continuous archiving. If the value passed in is already what the cluster is using, a 204 is returned. Otherwise a 202 is returned while the cluster is changed in the background.	`{"continuousArchiving" : "true"}`
storage	Add storage to the cluster. The number passed in is the new	`{"storage" : "5"}`

	total, not the amount to be added (in GB).	

serverClass	Machine scale a cluster.	`{`
		`"serverClass" : "m1.medium",`
		`"yumUpdate": "true",`
		`"vpcid": " General VM Network ",`
		`"ipPool": " Sales East "`
		`}`
		Please note that `yumUpdate` is optional, and will default to `false`.

Use a DELETE request with this resource to terminate a cluster. A successful call to this resource returns response code `202`.

14.1.17 /owners/*tenant_name*/clusters/*cluster_name*/events

Use the `/owners/`*`tenant_name`*`/clusters/`*`cluster_name`*`/events` resource to retrieve a list of events for a specific cluster. The information returned about each cluster may include:


```
{
    "event": [
        {
            "clocktime": "2016-01-18T23:35:05.497Z ",
            "description":
            "Creation of cluster acctg started.",
            "id": "12251",
            "owner": "Resources",
            "source":
            "fc6c56f0-c2c5-480d-8775-15249c70e1f4"
        },
        {
            "clocktime": "2016-01-18T23:35:05.497Z ",
            "description": "Load Balancer Port Notification
             cfa1bdb7-6357-4384-b48c-b38620b51939",
            "id": "12261",
            "owner": "Resources",
            "source":
            "fc6c56f0-c2c5-480d-8775-15249c70e1f4"
        }
    ]
}
```


`tenant_name`

`tenant_name` is the name of the tenant.

`cluster_name`

`cluster_name` is the name of the cluster.

Pass the tenant name and cluster name when using a GET request to retrieve a list of events for a specific cluster. An event has a numeric ID, a timestamp, and a message such as "`Creation of cluster acctg started.`"

A successful call to this resource will return a resource code of `200`.

14.1.18 /owners/*tenant_name*/clusters/*cluster_name*/statistics?start=*start_time*&end=*end_time*

Use this resource to retrieve statistics about the specified cluster for the given time period. The resource returns:


```
{
    "nodeStatistics": [
        {
            "nodeId":
            "aa27d8e0-6325-41c6-82dd-39868a66bd1c",
            "cpuload": "11",
            "freemem": "1882276",
            "usedmem": "40480",
            "connections": "1",
            "opspersecond": "1",
            "timestamp": "2016-01-18T23:35:05.497Z"
        },
        {
            "nodeId":
            "aa27d8e0-6325-41c6-82dd-39868a66bd1c",
            "cpuload": "10",
            "freemem": "1882276",
            "usedmem": "40480",
            "connections": "1",
            "opspersecond": "1",
            "timestamp": "2016-01-18T23:35:05.497Z "
        }
    ]
}
```


tenant_name

tenant_name is the name of the tenant.

cluster_name

cluster_name is the name of the cluster.

start_time

start_time is the time at which the report will start; specify the time in an ISO_8601 format, or as the number of milliseconds since January 1, 1970.

end_time

end_time is thetime at which the report ends; enter the time in an ISO_8601 format, or as the number of milliseconds since January 1, 1970. This parameter is optional.

Pass the tenant name, the cluster name, and the start and end times of the report with a GET request to retrieve statistics about a specific cluster over the given time period. Note that statistics are only kept for the last 14 days.

A successful call to this resource will return a resource code of `200`.

14.1.19 /serverimages

You can use the `/serverimages` resource to retrieve information about all currently defined server images, or to create a new server image:

```
{
    "id": "1",
    "imageId": "ccec7685-09d1-4bc4-8f30-4b2bf0f54bc7",
    "initialUser": "centos",
    "serverDescription": "Centos6.5",
    "serverId": "CENTOS6_5"
}
```

Use a GET request with this resource to retrieve a list of server images.

Use a POST request with this resource to create a new server image. When creating a new server image, omit the `id` field; the server will assign a unique identifier. You must specify a `serverId` field. The `serverID` field is displayed to the user in the `Server Type` field of the Ark console.

Only an administrative user may create a new server image. A successful POST returns response code `204`.

14.1.20 /serverimages/*image_id*

Use the `/serverimages/`*`image_id`* resource to retrieve information about a specific server image, modify a server image, or delete a server image.

```
{
    "id": "1",
    "imageId": "ccec7685-09d1-4bc4-8f30-4b2bf0f54bc7",
    "initialUser": "centos",
    "serverDescription": "Centos6.5",
    "serverId": "CENTOS6_5"
}
```


image_id

image_id is the unique identifier of the server image provided in the `id` field.

Pass in the image identifier of a server image along with a GET request to retrieve information about a specific server image.

Pass in the image identifier of a server image with the PUT request to update a server image.

Pass in the image identifier with a DELETE request to delete a server image.

You must be an administrative user to update or delete a server image. A successful call to update or delete this resource returns response code `204`.

14.1.21 /tokens

The EDB Ark API uses token-based authentication. All calls to the EDB Ark API require a valid token be passed in with the X-Auth-Token header.

You can use the `/tokens` resource to retrieve a token. The following example uses curl to demonstrate obtaining a token resource:

```
curl -i -H "Content-Type: application/json" -d \
'{"name":"alice","password":"1safepwd","tenant":"acctg"}' \
https://<host>/api/v2.0/tokens

HTTP/1.1 201 Created
Server: Admin
X-Subject-Token: 014khlia0abddk4xboyhy4bsygr9dt27ycyp1sdv
Content-Type: application/json
Transfer-Encoding: chunked
Date: Mon, 21 Sep 2015 17:25:42 GMT

{"expiresAt":"2015-09-21T18:55:42.582+01:00",
"issuedAt":"2015-09-21T18:25:42.582+01:00"}
```

The returned token is the string of random values returned in the `X-Subject-Token` field.

Use a POST request with a `/token` resource to retrieve a token used for token-based authentication.

When deleting a token, pass the token to be deleted in the `X-Subject-Token` header along with the normal `X-Auth-Token` header. A successful call returns response code `204`.

14.1.22 /users

Use a GET request with the `/users` resource to retrieve information about all currently registered users:

```
{
  "user":[
  {
    "id": "first.last",
    "region": "uk",
    "serviceprovider": "openstack-nova",
    "firstname": "First",
    "lastname": "Last",
    "email": "first.last@enterprisedb.com",
    "companyName": "EDB",
    "serviceProviderEndpoint": "",
    "creationTime": "2015-10-27T13:06:41.798Z",
    "lastLogin": "2016-02-23T19:38:25.369Z",
    "numLogins": "100",
    "enabled": "true",
    "numNodes": "65",
    "activationTime": "2015-10-27T13:06:41.798Z",
    "walArchiveContainer": "EDBArk"
  },
  {
    "id": "first.last",
    "region": "uk",
     ...
  }
       ]
}
```

This resource can be called only by an administrative user.

Please note: the `numNodes` field specifies the cumulative number of nodes created by the user; the nodes may or may not be currently running.

A successful call to this resource will return a resource code of `200`.

14.1.23 /users/*user_id*

Use the /`users`/*`user_id`* resource to retrieve or modifyinformation about a specific user:

```
{
    "id": "first.last",
    "region": "uk",
    "serviceprovider": "openstack-nova",
    "firstname": "First",
    "lastname": "Last",
    "email": "first.last@enterprisedb.com",
    "companyName": "EDB",
    "serviceProviderEndpoint": "",
    "creationTime": "2015-10-27T13:06:41.798Z",
    "lastLogin": "2016-02-23T19:38:25.369Z",
    "numLogins": "100",
    "enabled": "true",
    "numNodes": "65",
    "activationTime": "2015-10-27T13:06:41.798Z",
    "walArchiveContainer": "EDBArk"
}
```

`user_id`

> *`user_id`* is the identity of a registered EDB Ark user; the *`user_id`* takes the form of *`first_name.last_name`*.

Pass the *`user_id`* with a GET request to retrieve information about a specific user. The `email` field will be returned only if the user has set a notification email value.

Use a PUT request to update user information. You can modify:

- user's first or last name
- user's last name
- company name
- email address

Other changes will be ignored. Successful calls to this resource return a response code of `204`.

14.1.24 /users/*user_id*/notifications

Use the `/users/`*`user_id`*`/notifications` resource to retrieve a notification for a specific user. A call to `/users/`*`id`*`/notifications` returns a message in the format:

```
{"message":"The service provider was unable to create the
requested instance at this time"}
```

`user_id`

> *`user_id`* is the identity of a registered EDB Ark user; the *`user_id`* takes the form of *`first_name.last_name`*.

Provide the *`user_id`* of a registered EDB Ark user with a GET request to retrieve a notification message for the user (if available). If there is no notification for the specified user, the server returns response code `204`.

14.2 Response Codes

The API will return the response codes listed in the table below. For more information about HTTP response codes, please visit:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Code	Means	Description
200	OK	Returned when the request has completed successfully.
201	Created	A resource has been created; the location header contains the new URI.
202	Accepted	Returned when the request was accepted and the task is continuing asynchronously.
204	No Content	Usually sent in response to a PUT or DELETE call that is changing or deleting a resource. Also returned if there is no available information.
400	Bad Request	The request cannot be understood by the server.
401	Unauthorized	Request does not have valid auth token, a request for a token doesn't have the right info, the user isn't authorized for that information, etc.
404	Not Found	Response when a URI does not point to a valid location.
405	Method Not Allowed	Returned when the HTTP method is not allowed.
409	Conflict	The request would create a conflict with the current state of the resource.
500	Server side errors	500 means that a problem has occurred on the server side – check your server log for details.

