IBM Cloudant
Version 1 Release 0

Installing and Maintaining IBM Cloudant Data Layer Local Edition

IBM Cloudant
Version 1 Release 0

Installing and Maintaining IBM Cloudant Data Layer Local Edition

Note
Before using this information and the product it supports, read the information in "Notices" on page 93.

Version 1 Release 0

This edition applies to version 1, release 0, modification 0 of IBM Cloudant Data Layer Local Edition and to all subsequent releases and modifications until otherwise indicated in new editions.

| Changes in this edition are marked by the revision tag shown alongside this paragraph.

Contents

Chapter 1. IBM Cloudant Data Layer Local Edition Overview

The IBM® Cloudant® Data Layer Local Edition (Cloudant Local) version 1.0.0.5 includes the same robust, flexible features that were originally available only in the Cloudant Database-as-a-Service (DBaaS) offering.

Cloudant DBaaS was the first data management solution to use the availability, elasticity, and reach of the cloud to create a global data delivery network (DDN). Cloudant enables applications to scale larger and remain available to users wherever they are. Now, those features are available to implement in your location.

You can perform the following tasks after you install Cloudant Local.

- Distribute readable and writable copies of data to multiple locations or devices.
- Synchronize data continuously through filtered, multi-master replication.
- Store data of any structure as self-describing JSON documents. Moreover, users can read from and write to the closest available data source.
- Integrate with a RESTful API.

Chapter 2. What's New in Cloudant Data Layer Local Edition

Read about the new features in Cloudant Data Layer Local Edition release version 1.0.0.5.

Expanded OS Platform Support

Cloudant Local expanded its support to include more operating system platforms and versions as you can see in the following updated list.

- *x86_64 architecture*

  Debian-derived Linux distributions
  - Ubuntu Server 12.04 (precise) and 14.04 (trusty)

  Red Hat-derived Linux distributions
  - Red Hat Enterprise Linux Server 6.x and 7.x
  - Community ENTerprise Operating System (CentOS) 6.x and 7.x
  - Oracle Enterprise Linux Operating System (OEL) 6.x and 7.x

  SUSE Linux distributions
  - SUSE Linux Enterprise Server (SLES) 12
- *IBM System z® s390x architecture*

  Linux on IBM System z
  - Linux on IBM System z - Red Hat Enterprise Linux 7.x
  - SUSE Linux Enterprise Server (SLES) 12

Easier installation and configuration with Cluster Admin and Support Tool

The Cluster Admin and Support Tool (CAST) is available for all platforms. You can use the tool to manage the installation, configuration, and various administration activities on a Cloudant Local cluster and its individual nodes.

MustGather captures diagnostics

The MustGather tool collects system data, including Cloudant cluster data, logs, and other configuration information, making it easier to collect diagnostic information. The tool consolidates the information into a compressed file that can be sent to IBM Support.

LDAP integration

You can configure Cloudant Local to use the LDAP service for access control. You can delegate authentication and authorization services to LDAP when you access a Cloudant Local instance.

Standardized query output format

In order to standardize the response format of queries, you can make the following configuration changes.

- You can change the response format for geospatial queries to match the standard response format used for other types of queries. This change can be set up by using the following configuration variables in the `local.ini` file.
  - **Section** *hastings*
  - **Key** *default_format*
  - **Possible values** *view*, *legacy_local*

- **Default** *legacy_local*

  Note: The default value *legacy_local* provides the format used in all the previous versions. Use *view* for the standardized format.

- You can change the format of the **update_seq**, **seq**, and **last_seq** because the parameters used in the database information responses, **_changes responses** and **_changes queries**, can be changed. You can use the new string format by setting the following configuration variables in the `local.ini` file.
  - **Section** *update_seq*
  - **Key** *format*
  - **Possible values** *array*, *string*
  - **Default** *array*

  Note: The default value *array* provides the format used in all the previous versions. Use *string* for the standardized string output format.

Chapter 3. Cloudant Local hardware and software requirements

Before you install Cloudant Local, confirm that your system meets the following requirements. The requirements include cluster server requirements, supported platforms, hardware requirements, single node requirements, and user requirements for installing the product.

Cluster server requirements

A minimum of five servers are recommended to create a fully functional Cloudant cluster that ensures 24 x 7 availability.

Database Nodes

- Cloudant Local must be installed on at least three database nodes for replication purposes.
- (Optional) Cloudant Local can be installed on as many other nodes as needed to scale for your business requirements.

Load Balancers

- One load balancer must be installed on a server that is separate from the servers that are used in the Cloudant cluster.
- It is a good practice to install two load balancers in case the primary load balancer fails. If a second load balancer is installed, you must install it on a separate server to ensure uninterrupted service.

  The nodes in the Cloudant cluster and the server where your load balancer is installed must use the same operating system. For example, if the nodes in the cluster are operating in a CentOS environment, the load balancer must be installed on a CentOS server.

Supported architecture and platforms

You can install Cloudant Local on the following platforms.

x86_64 architecture

Debian-derived Linux distributions

- Ubuntu Server 12.04 (precise)
- Ubuntu Server 14.04 (trusty)

Red Hat-derived Linux distributions

- Red Hat Enterprise Linux Server 6.x
- Red Hat Enterprise Linux Server 7.x
- Community ENTerprise Operating System (CentOS) 6.x
- Community ENTerprise Operating System (CentOS) 7.x
- Oracle Enterprise Linux Operating System (OEL) 6.x
- Oracle Enterprise Linux Operating System (OEL) 7.x

SUSE Linux distributions

- SUSE Linux Enterprise Server (SLES) 12

IBM System z s390x architecture

Linux on IBM System z

- Linux on IBM System z - Red Hat Enterprise Linux 7.x
- SUSE Linux Enterprise Server (SLES) 12

Hardware requirements

Cloudant Local hardware requirements vary based on various factors for the database and load balancers.

x86_64 hardware requirements

The following hardware requirements apply to the x86_64 architecture.

Database Nodes

- The minimum requirements include four cores and eight threads, such as Xeon E3-1270 V2, 8 GB of RAM and 1-gigabit network.
- For larger implementations, the minimum requirements include 12 cores and 24 threads, such as dual Xeon E5 2620, 64 GB of RAM, and local SSD drives to meet data volume requirements of your usage and a 1-gigabit network.

Disk space requirements for the data vary based on how much data you want to store. At minimum, the standard per-node storage setup is as follows.

- If you use spinning disks, 4 x 600 GB 15k SAS drives in RAID 0 can provide about 2.2 TB of usable storage.
- If you use solid-state drives (SSDs), 2 x 800 GB SSDs in RAID 0 can provide about 1.5 TB of usable storage.
- An ext4 file system that is mounted with the noatime option is recommended for performance reasons.

Note: A storage area network (SAN) is not recommended. If you use a centralized location to store all the database files, such as a SAN, you might lose high-availability. If the SAN goes down, all of your database nodes are unavailable. However, if all of your database nodes use directly attached storage, you can lose two-thirds of your system and remain operational.

For the operating system and Cloudant binaries, the disk that is allocated must be 10 GB in RAID 1.

Load Balancer Nodes

- The minimum requirements are dual-core processor and 4 GB RAM, 500 GB local hard disk drive, and a 1-gigabit network.
- For larger implementations, the minimum requirements include a quad-core processor and 8 GB RAM, 1 TB local hard disk drive, and a 1-gigabit network.

Linux on IBM System z

Confirm that your environment meets the following hardware requirements.

Models

- z13
- zEnterprise - zBC12 and zEC12
- zEnterprise - z114 and z196

Processors and Memory

- Database Nodes
  - 2 IFLs in SMT mode

- 8 GB memory for each node
- Load Balancer Nodes
  - 2 IFLs
  - 4 GB memory for each node

Requirements for a single-node implementation

In a single-node implementation, a database node and a load balancer node are installed on a single node. This setup is only recommended for trial testing and development.

If you use a single-node implementation for this purpose, the requirements include a single-core VM with 0.5 GB of RAM is sufficient to run the product.

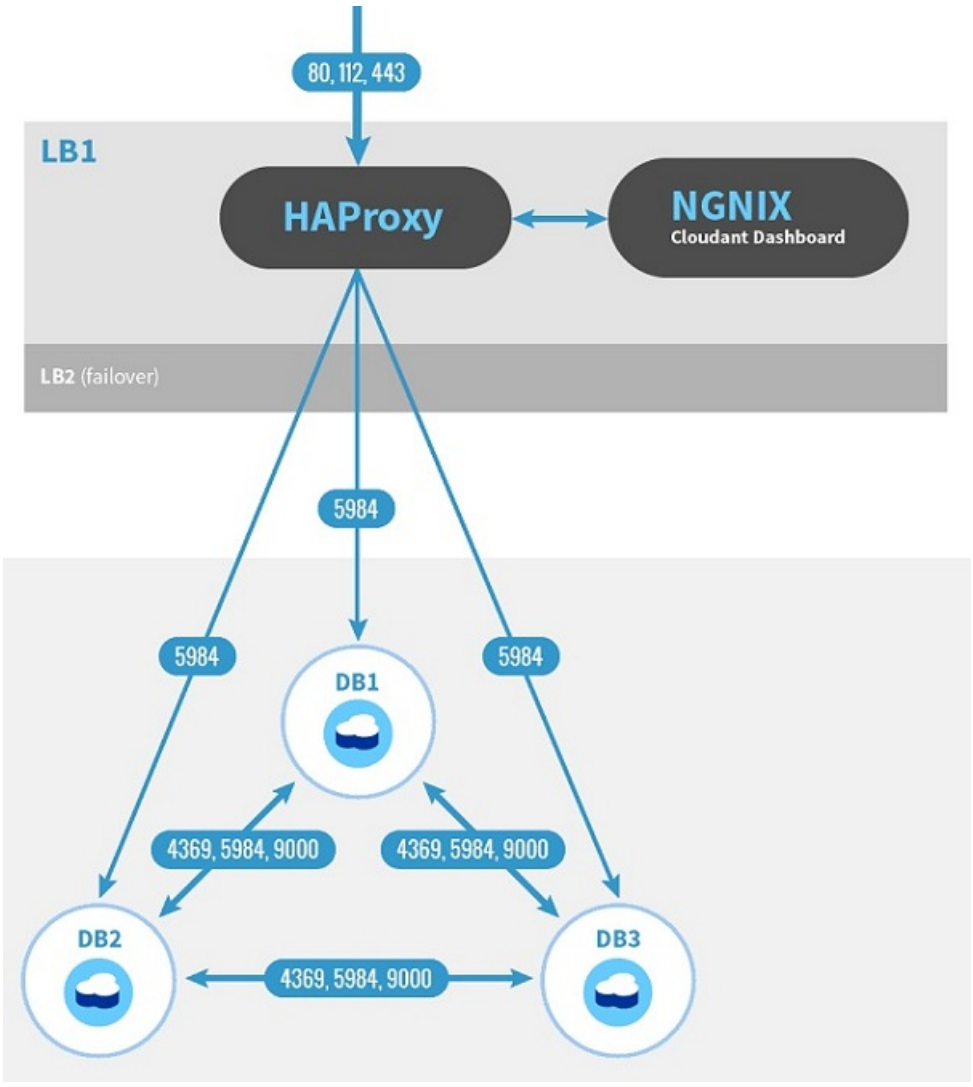
A single-node implementation is not recommended in a production environment. However, if you use a single-node implementation in a production environment, the requirements are the same as for a database node.

Cloudant Local architecture and ports

The following diagram shows the architecture of a Cloudant Local implementation.

- Three database nodes that are identified as `DB1`, `DB2`, and `DB3`.
- Two load balancers that are identified as `LB1` and `LB2`.

In the diagram, the lines and accompanying text identify the communication paths and ports that are used by the components of Cloudant Local.



What to do next

After you confirm that your system meets all relevant requirements for implementing Cloudant Local, see Chapter 4, “Planning your Cloudant Local installation,” on page 9.

Chapter 4. Planning your Cloudant Local installation

Use this information to plan your implementation and verify that your nodes and load balancers are properly configured before you install Cloudant Local.

Factors to consider when you are planning where to install Cloudant Local

Like most applications, Cloudant Local consists of two major sections: the application and the data it manages. After installation, the application does not change often, but the data is likely to change and grow over time. Therefore, it is helpful to plan your installation carefully before you install the product.

In particular, consider the following aspects.

- Any organizational policies that you might have regarding the installation of software and data. For example, you might have a policy against installing databases on the root partition of a system.
- Current® and expected storage requirements. For example, while the target system might have enough storage space for the initial installation, subsequent use and data growth might exceed the available space.
- Simplification of maintenance and backup. For example, by keeping the application software and data locations separate, it is often easier to perform backup and maintenance tasks.

If you are installing Cloudant Local onto a dedicated system, you are likely to have more control over where data is stored and how future capacity is assured. If you are installing Cloudant Local onto an existing or shared system, it is even more important to consider your installation plan carefully because you might not have full control over available space and future storage space.

When you consider these factors in your plan, you are better able to identify installation locations and any storage partition requirements.

Cloudant Local default Extract and Install locations

When you install Cloudant Local, the installation wizard creates `${HOME}/cloudant` by default, and extracts the files there. You can change the name of the ${HOME} directory if choose and *cloudant* is appended.

The Cloudant Local packages create the Cloudant software and database directories. The software directory contains the Cloudant binaries, scripts, configuration files, and other executable files. The database directory is used to store your data files. By default, the packages create the Cloudant Local software and database directories in the following locations.

- The software is installed in a directory named `/opt/cloudant`.
- The database is installed in a directory named `/srv/cloudant`.
- (Optional) You can move the database location by running `cast database relocate`. Display the help text by running the help command, `cast database relocate --help`.
- Ensure that the new directory location is owned by `cloudant:cloudant` and is applied to all database nodes with the same path and name.

If needed, you can move these directories to another drive or partition by creating at least one mount point for the Cloudant software and database directories.

- If you want the Cloudant software and database directories on the same mount point, create one mount point for the two directories.
- If you want the Cloudant software and database directories on separate mount points, create two mount points, one for the software directory and another for the database directory.

If you want to install Cloudant Local on a mount point or move the product to a mount point after it is installed, follow these instructions to install Cloudant Local on a partition other than your root partition.

1. Create the following Cloudant directories on at least one mount point.
   - Make a Cloudant software directory named */your mount point*/opt/cloudant.
   - Make a Cloudant database directory named */your mount point*/srv/cloudant.

   You can put both directories on one mount point, or you can put the two directories on separate mount points. For example, if your *mount point* for both directories is /export/local, use the following commands to create the Cloudant directories.

   ```
   mkdir /export/local/opt/cloudant
   mkdir /export/local/srv/cloudant
   ```

   The first command makes the Cloudant software directory on the mount point. The second command makes the Cloudant database directory on the mount point.
2. Create symbolic links (symlinks) to the mount point for each directory.

   For example, if your mount point for both directories is /export/local, use the following commands to create the symbolic links.

   ```
   ln -s /export/local/opt/cloudant /opt/cloudant/ -s /export/local/srv/cloudant /srv/cloudant/
   ```

   The first command creates the symbolic link for the Cloudant software directory. The second command creates the symbolic link for the Cloudant database directory.
3. Install Cloudant Local on each database or load balancer node in your implementation. See "Installing a cluster" on page 13.

Verify that your nodes are properly configured

Before you install Cloudant Local on any database or load balancer node, you must verify that all nodes in your cluster are named and configured properly with static IP addresses. When Cloudant is installed, it is configured to start automatically. If your nodes are not named and configured correctly, the wrong node names are included in the memberships for your Cloudant cluster.

Run the **hostname -f** command on each node in your cluster to verify that the node is named and configured correctly. The **hostname -f** command returns a Fully Qualified Domain Name (FQDN), such as db1.domain.com. If the displayed hostname is incorrect, do one of the following steps.

- If your operating system is Red Hat or CentOS, do the following steps.
  - Append the following line to the /etc/hosts file on each node: <node IP address> <hostname>. For <node IP address>, specify the external IP address for the node, not the loop-back interface IP address. For <hostname>, specify the appropriate hostname, such as the one in the following example.

    ```
    107.170.185.247               db2.millay-centos65-cloudant-local.com.
    ```

  - Confirm that the correct hostname is specified in the /etc/sysconfig/network file. Use the **hostname -f** command to confirm that the name is correct.
  - If the hostname is correct, restart the server for the new entry to take effect. If not, correct the invalid hostname.
- If your operating system is Debian or Ubuntu, confirm that your DNS entries are set up correctly, and then do the following steps.
  - In the /etc/hosts file, edit the line that starts with 127.0.1.1. For example, 127.0.1.1 db1.clustername.domain.net db1. Change this entry as needed to suit your environment.
  - Edit the /etc/hostname file and change the content of that file to reflect the short-form hostname in the /etc/hosts file, not the fully qualified name. In the previous step, for example, the short-form hostname is db1.
  - Run the **sudo hostname --file /etc/hostname** command to update the operating system hostname, based on the name that is specified in the /etc/hostname file.

– Confirm that the correct hostname is specified with the **hostname -f** command. If the specified hostname is incorrect, correct it and repeat the previous step.

Attention: You can configure your database nodes for either local logging or remote logging to a separate `syslog` logging server. The default setting is local, but that setting is recommended only in test environments. In a production environment, you must use remote logging to ensure optimum system performance. If you want to set up logging, see Chapter 10, "Configuring logging," on page 33.

Load balancer prerequisites

Before you install Cloudant Local, verify that you meet the following prerequisites.

1. You must install and configure a load balancer on a separate server from the ones that are used in the Cloudant cluster. To ensure uninterrupted service, you can install a second (failover) load balancer server.

   The nodes in the Cloudant cluster and the server on which your load balancer is installed must use the same operating system. For example, if the nodes in the cluster are operating in a CentOS environment, the load balancer must be installed on a CentOS server.
2. If you are deploying multiple load balancers in a failover configuration, confirm that all load balancers are on the same Layer 2 LAN segment. This step is required for VRRP-based failover to function correctly.
3. Confirm that DNS entries exist for both the public and private IP addresses for each Load Balancer node.

   Record the following information for your primary load balancer (Load Balancer 1) for future reference.
   - Public DNS Name
   - Public IP
   - Private DNS Name
   - Private IP

   Note: If you use only a single IP address per node, complete only the "Public" lines for your primary and secondary load balancers. The "Public" IP addresses for these database nodes do not have to be public IP spaces. They are public only in the sense that they provide the actual database service. In general, this network would be a separate network segment from the "private" side of the servers, where all node-to-node and out-of-band communication takes place.

   If you use a failover load balancer, record the same information for your second load balancer (Load Balancer 2).
   - Public DNS Name
   - Public IP
   - Private DNS Name
   - Private IP
4. Confirm that the virtual IP address is available for the load balancers to share. This address is the IP address that clients use to access the cluster. Record this IP address for future reference.
   - Virtual IP address

What to do next

After you develop an installation plan and verify that your nodes and load balancers are properly configured, you can "Installing a cluster" on page 13.

Chapter 5. Installing Cloudant Local

Use these instructions to install Cloudant Local on a database and a load balancer node in either a multi-node cluster or a single-node implementation.

Installing a cluster

To install and configure a Cloudant Local cluster, you use the Cluster Admin and Support Tool (CAST) which manages the installation, configuration, and support of a Cloudant Local cluster and its individual nodes.

Prerequisites

- You must be the root user, or have sudo privileges, to install Cloudant Local. If you are not the root user, prefix all commands with the sudo command.
- If security restrictions prevent you from accessing third-party repositories, you must download and install the required third-party packages before you start the installation.
- Install EPEL on Red Hat Enterprise Linux versions 6 and 7 and CentOS 6.5 only. The Cloudant Local installation uses packages from EPEL.
  - If the packages are not installed on your system, run this command to install and enable EPEL.

    ```
    sudo yum -y install epel-release
    ```
  - When the installation is complete, you can remove EPEL with this command.

    ```
    sudo yum erase epel-release
    ```

These instructions describe how to extract and install CAST and install Cloudant Local.

Extracting Cloudant Local installation packages

To install (or uninstall) Cloudant Local, you must be a root user or use the `sudo` command.

Important: If you select a different home directory, the installer automatically adds the `cloudant` directory to your home directory.
Extract the Cloudant Local installation packages by using the following steps.

1. Download the `tar` file for the OS version you want to install.
2. Extract the contents of the `tar` file.

   ```
   cd cloudant-installer
   tar -xvf xxxxxxx.tar.gz
   ```

   The *xxxxxxx*`.tar.gz` parameter is the Cloudant Local part number.

   Note: The Cloudant Local installation kit creates a `cloudant-installer` subdirectory and extracts the installation files into it.
3. Extract the Cloudant Local packages and install CAST.

a. To extract the Cloudant Local packages by using an interactive wizard, run the following command.

```
./install.bin
```

b. To extract the Cloudant Local packages silently, and implicitly accept the IBM License Agreement, run the following command.

```
./install.bin -i silent -f production.properties | development.properties
```

The properties file you specify determines whether this instance of Cloudant Local is used for production or development purposes.

Installing the first database node

Install and configure the first database node in the cluster.

1. Run the CAST installation command to install the first database node of the cluster.

```
cast system install -p dbadmin_password –db
Options:
-db –dbnode     Flag to install a database node.
-p –password    (Optional) The database admin password.
If you do not provide a password, the password defaults to pass.
```

2. Run the export command to export configuration values to a file.

```
cast cluster export cluster_dbnode.yaml
```

Important: For each additional database node in the cluster, you must use the same configuration file. This file is required since the user passwords and cookie values must be the same on each host to join the nodes together into a cluster.

3. Copy the `cluster_dbnode.yaml` file to a place that is accessible when you install more database nodes.

Installing additional database nodes

For each additional database node you add within the cluster, you must perform the following step.

```
cast system install -c cluster_dbnode.yaml -db
Options:
-db –dbnode    Flag to install a database node.
-c –config     The cluster configuration file exported from the first node.
– maintenance  Start the system in maintenance mode.
```

You added the additional database nodes. Next, you must join the cluster.

Joining nodes in a cluster

After you install and configure all the database nodes, you must join them together to form the cluster.

1. Log in to the first database node.
2. Run the `cast cluster add` command for each node in the cluster except for the database node from which you are running the command.

```
cast cluster add FQDN1 FQDN2
```

For example, if your environment includes three database nodes, the Fully Qualified Domain Name (FQDN) might be db1.vm, db2.vm, or db3.vm. In this case, if you are on the db1.vm node, you must add both database nodes, db2.vm and db3.vm by running the following command.

```
cast cluster add db2.vm db3.vm
```

3. Verify the status of each cluster by running the following command.

Each node in the cluster displays an online or offline status. An offline status indicates that the node did not join the cluster correctly. If a node did not join the cluster correctly, remove the node and repeat the installation and configuration steps.

```
cast cluster status
```

4. If you need to remove a node from the cluster, run the following command.

```
cast cluster remove FQDN1 FQDN2
```

This command is an example.

```
cluster remove db2.vm db3.vm
```

After you join the cluster the first time, you must initialize it.

Initializing the cluster

During a new installation and after all the database nodes are joined to the cluster, you must initialize the cluster.

Run the following command from one of the database nodes to initialize the cluster.

```
cast cluster init
```

Now, you must install a load balancer.

Installing load balancer nodes

Install and configure each load balancer node in the cluster by following these steps.

1. Install a load balancer node by running the following command.

```
cast system install -lb

Options:
-lb –lbnode     Flag to install a load balancer node.
```

2. Copy the load balancer node configuration file. `/opt/cloudant/cast/samples/lbnode.yaml`.
3. Update the hostname and IP address values for each database node in the cluster under the Nodes section of the configuration file. Replace the hostname with the FQDN.

```
hostname: xxx.xx
ipaddress: xxx
```

4. Change the default admin password from `pass` as shown in the sample.

```
# Sample CAST input file for configuring a load balancer in a three-node database cluster.
          lbnode:
          users:
          http_stats: pass
          nodes:
          - hostname: db1.vm
          ipaddress: 192.168.211.101
          - hostname: db2.vm
          ipaddress: 192.168.211.102
          - hostname: db3.vm
          ipaddress: 192.168.211.103
```

5. Save the configuration file and update the node with the new values by running the following command.

```
cast node config lbnode.yaml
```

After you run the `cast node config` command, the load balancer node is configured. If you have completed the installation of the database nodes and initialized the cluster, verify that you can “Start the Cloudant Local Dashboard” on page 16.

Installing a single node

If you want to install Cloudant Local in a single-node implementation instead of a cluster, you must install the load balancer and database nodes to run on the same node. In most cases, a stand-alone node

is installed for development or demonstration purposes and uses the default configuration settings. This implementation is not recommended for a production environment.

Install a single node by running the following command.

```
cast system install -p dbadmin_password -lb -db
```

```
Options:
-db –dbnode    Flag to install a database node.
-lb –lbnode    Flag to install a load balancer node.
-p –password   (Optional) The database admin password.
If you do not provide a password, the password defaults to pass.
```

Start the Cloudant Local Dashboard

After you install and configure the database and load balancer nodes, verify that Cloudant Local installed successfully by logging in to the Cloudant Local Dashboard on each load balancer.

1. Start and stop the dashboard by using the following commands.

   ```
   cast node start
   cast node stop
   ```

2. Enter the following URL in a web browser to access the Dashboard.

   ```
   https://your_load_balancer_IP_address/dashboard.html
   ```

   In this URL, `your_load_balancer_IP_address` is the fully qualified hostname of your load balancer.

3. Include the `/dashboard.html` part of the URL to display the dashboard. If you omit the `/dashboard.html` part, the standard Cloudant welcome message is displayed instead of the dashboard.

   ```
   {
               couchdb: "Welcome",
               version: "1.0.0.5-local",
               vendor: {
               name: "Cloudant, an IBM Company",
               version: "1.0.0.5",
               variant: "local"
               },
               features: [
               "geo"
               ] }
   ```

4. Enter your database administration credentials at the Cloudant Dashboard.

   If you successfully authenticate to the Cloudant Local Dashboard, your install is complete.

What to do next

After you complete the basic Cloudant Local installation, you can continue configuring other areas of Cloudant Local, such as logging, SSL communication, and database-level security.

- Configure the location where you want to store remote and local logging information. See Chapter 10, “Configuring logging,” on page 33.
- Configure SSL communication. See Chapter 8, “Configuring SSL,” on page 21.
- Configure database-level security. See Chapter 11, “(Optional) Configuring database-level security,” on page 39.

Chapter 6. Managing Cloudant Local services

Use these instructions to stop, start, or restart Cloudant Local with the `cast node` commands. You can also display the status of various processes and test the Cloudant Local API.

You can start, stop, and restart your Cloudant Local services with the `cast node` command by specifying the correct option.

- Start all Cloudant Local.

  `cast node start`

- Stop all Cloudant Local services.

  `cast node stop`

- Restart all Cloudant Local services. Use the restart command to restart your services after you change s configuration file.

  `cast node restart`

- (Optional) Display the syntax help text for the CAST command.

  `cast system install --help`

Chapter 7. Using the Mustgather tool for Diagnostics

The MustGather tool is a lightweight diagnostic tool that collects system data, including Cloudant cluster data, logs, and other data that is useful for diagnosing issues. You consolidate your collected system data into a zip file to send to IBM Support.

Overview of Cloudant Local MustGather tool

The MustGather tool creates a snapshot of the on-premise Cloudant clusters that can be used to diagnose issues more quickly. The MustGather tool is available on all database and load balancer nodes. You can run the tool from any node. You specify the nodes within the cluster in the `mustgather.ini` file or from the command line. The MustGather tool collects the relevant files from different nodes and consolidates them into one compressed file that you send to IBM Support to help them diagnose issues with Cloudant Local.

Modify the `mustgather.ini` file

The `mustgather.ini` file is configured when you run the MustGather tool. However, if you need to, you can update certain information in the `mustgather.ini` file after you run the MustGather tool.

The following example of the `mustgather.ini` file is in the `/opt/cloudant/etc` directory.

```
# Properties file
#
# Enter a comma-separated list of the cluster nodes in the NODES field.
#
# This example shows how to specify a cluster that includes three databases and one load balancer node:
#
# NODES: lb.your_locale.your_company.com,
#  db1.your_locale.your_company.com,
#  db2.your_locale.your_company.com,
#  db3.your_locale.your_company.com

[cluster]
nodes:
os_messages: /var/log/messages*
haproxy_cfg: /etc/haproxy/haproxy.cfg
haproxy_logs: /var/log/haproxy*
nginx_cfg: /etc/nginx/nginx.conf
nginx_dashboard_cfg: /etc/nginx/conf.d/cloudant-dashboard.conf
nginx_logs: /var/log/nginx*
cloudant_properties_cfg: /opt/cloudant/etc
cloudant_version: /root/cloudant/version.txt
cloudant_logs: /var/log/cloudant
```

Note: To collect log files from different nodes, port 22 must be open.

Edit the `mustgather.ini` file

To edit the `mustgather.ini` file, open the `mustgather.ini` file and update the following information on the node where you want to run the tool.

- Update the nodes in the `[cluster]` section of the file with the information for your cluster nodes. When you specify multiple nodes in the cluster, enter a comma-separated node name. The node name can either be the hostname or the IP address. If you specify a hostname, make sure to update the `hosts` file as you can see in the example.

  ```
  nodes: db1.cloudant-local.com,db2.cloudant-local.com,db3.cloudant-local.com,lb1.cloudant-local.com
  nodes: 192.168.1.100,192.168.1.101,192.168.1.102,192.168.1.103
  ```

- (Optional) Update the other information in the `[cluster]` section.

Run the MustGather tool

If you have an issue with Cloudant Local, run the MustGather tool to collect your system data and send it to IBM Support.

1. Run the MustGather tool by using the following command.

   Important: To run the Mustgather tool, the user must have root privileges.

   ```
   mustgather --help
   usage: mustgather [-h] [-n NODES] [-o OUTPUT_DIR]

   mustgather - Gathers diagnostic data for a Cloudant cluster
   optional arguments:
   -h, --help     Show this help message and exit
   -n NODES       Comma-separated list of hostnames for the cluster nodes that you can configure as
   the list of nodes for mustgather to use
   -o OUTPUT_DIR  Directory where the output tar file is created
   ```

2. Run the MustGather tool with option `-n` or option `-o`.
   a. If the `mustgather.ini` file was not updated to specify the nodes, the first time you run the MustGather tool you must specify the `-n` option. For any subsequent run, you do not need to specify the `-n` option unless the node information changes from the previous run.
   b. If the `-o` option is specified, the `mustgather.tar` file is saved to the output directory. If the `-o` option is not specified, the `mustgather.tar` file is saved to the current directory.

      After you run the tool, send the `tar` file to IBM support.

Chapter 8. Configuring SSL

Use these instructions to configure Secure Sockets Layer (SSL) for your Cloudant Local installation, especially if you access the Cloudant database across an untrusted network. If your network is trusted, configuring SSL/HTTPS is optional, and you can use HTTP requests (that is, port 80), instead.

SSL is a standard security technology for establishing an encrypted link between a server and a client, such as between a website and a browser. SSL is used to ensure that sensitive information is transmitted securely over the internet. SSL uses a cryptographic system that uses two keys to encrypt data: a public key that is available to anyone, and a secret key that is known only to the recipient of the message.

Configuring SSL on your load balancers

Use these instructions to configure Secure Sockets Layer (SSL) for Cloudant Local on each load balancer in your cluster. If you use SSL and you use two load balancers (that is, a primary and failover load balancer), you must configure SSL on both load balancers.

Follow these steps to configure SSL for Cloudant Local on each load balancer in your cluster.

Generate an RSA private key

RSA is a public-key cryptosystem, and is widely used for secure data transmission.

RSA involves a public key and a private key. The public key is available to anyone and is used to encrypt messages, while the private key is required to decrypt the messages. RSA also is used to sign a Certificate Signing Request or CSR.

Use the **`openssl`** toolkit to generate an RSA private key in the `/etc/haproxy` directory. The key is 1024 bit and is stored in PEM format, which is readable as ASCII text.

Note: PEM is a container format that can include just the public certificate, or it can include an entire certificate chain, including a public key, private key, and root certificates. PEM was originally developed to secure email, and the PEM acronym was derived from the phrase `Privacy Enhanced Email`.

Use this command to generate an RSA key.

```
openssl genrsa -out rsa.key 1024
```

This example shows how this command is used to generate an RSA key.

```
[root@lb1centos haproxy]# openssl genrsa -out rsa.key 1024
Generating RSA private key, 1024 bit long modulus
.++++++
...................++++++
e is 65537 (0x10001)
[root@lb1centos haproxy]#
```

Generate a CSR

Use the RSA private key that you generate to create a certificate signing request (CSR).

In a production environment, the CSR is sent to a certificate authority (CA), who supplies a signed certificate to reassure users that the certificate is valid. In this information, the CSR is self-signed because the certificate is for internal use only.

The **openssl** toolkit is used to generate the CSR.

Use this command to generate a CSR.

```
openssl req -new -key rsa.key -out rsa.csr
```

During the CSR generation process, you are prompted for several pieces of information, such as your company name, email address, and a "challenge password". When prompted for a Common Name, enter the domain name for the system where the certificate is to be installed, for example a load balancer. Reply to all prompts as needed.

This example shows how this command is used to generate a CSR.

```
[root@lb1centos haproxy]# openssl req -new -key rsa.key -out rsa.csr
You are about to be asked to enter information that incorporates
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
Some fields have a default value,
if you enter '.', the field is left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:New Jersey
Locality Name (eg, City) [Default City]:Princeton
Organization Name (eg, company) [Default Company Ltd]:IBM
Organizational Unit Name (eg, section) []:Cloudant
Common Name (eg, your name or your server's hostname) []:lb1centos.princeton.usnj.ibm.com
Email Address []:silvagni at us dot ibm dot com

Enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
[root@lb1centos haproxy]#
```

Generate a self-signed certificate

If required, create a self-signed certificate for your internal use.

When a self-signed certificate is used for a website, an error message is displayed in a browser when a user attempts to connect to the site. The message warns users that the certificate for the specified website is signed by an unknown and untrusted certificate authority. In a production environment, a CSR usually is sent to a certificate authority (CA) for certification. After the CA confirms that the site is valid, the CA issues a signed certificate to reassure users that the certificate for the site is valid.

Use this command to generate a self-signed certificate that is valid for 365 days, and that is suitable for testing.

```
openssl x509 -req -days 365 -in rsa.csr -signkey rsa.key -out rsa.crt
```

This example shows how this command is used to generate a self-signed certificate.

```
[root@lb1centos haproxy]# openssl x509 -req -days 365 -in rsa.csr -signkey rsa.key -out rsa.crt
Signature ok
subject=/C=US/ST=New Jersey/L=Princeton/O=IBM/OU=Cloudant/CN=lb1centos.princeton.usnj.ibm.com/
emailAddress=silvagni at us dot ibm dot com
Getting Private key
[root@lb1centos haproxy]#
```

Combine the RSA certificate and key

The RSA certificate and key must be combined into a single file for use in HAProxy 1.5 and higher.

Combine the RSA certificate (`/etc/haproxy/rsa.crt`) and RSA key (`/etc/haproxy/rsa.key`) into a new single file (`/etc/haproxy/certificate.pem`) for use in HAProxy. Put the certificate in the `certificate.pem` file first, with the key as the final portion of the file.

Use this command to combine the files in the correct order.

```
cat /etc/haproxy/rsa.crt /etc/haproxy/rsa.key > /etc/haproxy/certificate.pem
```

Secure the RSA key and certificate

The RSA key and certificate must be protected by using appropriate access permissions.

Ensure that the RSA key and certificate are owned by root, belong to group root, and have permissions set to 0400.

```
chown root:root /etc/haproxy/rsa.* /etc/haproxy/certificate.pem
chmod 0400 /etc/haproxy/rsa.* /etc/haproxy/certificate.pem
```

This example shows how these commands are used to ensure that the RSA key and certificate are owned by root, group root and permissions are set to 0400.

```
[root@lb1centos haproxy]# chown root:root /etc/haproxy/rsa.* /etc/haproxy/certificate.pem
[root@lb1centos haproxy]# chmod 0400 /etc/haproxy/rsa.* /etc/haproxy/certificate.pem
[root@lb1centos haproxy]# ls -la
total 44
drwxr-xr-x.  2 root root 4096 Sep 29 15:35 .
drwxr-xr-x. 89 root root 4096 Sep 29 15:35 ..
-r--------.  1 root root 1884 Sep 29 15:26 certificate.pem
-rw-r--r--.  1 root root 8552 Sep 25 02:23 haproxy.cfg
-rwxrwx---.  1 root root 7471 Sep 19 01:35 haproxy-cloudant.cfg
-r--------.  1 root root  997 Sep 29 15:25 rsa.crt
-r--------.  1 root root  781 Sep 29 15:24 rsa.csr
-r--------.  1 root root  887 Sep 29 15:23 rsa.key
[root@lb1centos haproxy]#
```

Configure HAProxy for SSL connections

The RSA key and certificate must be protected by using appropriate access permissions.

HAProxy 1.5 includes native support for SSL connections. The `/etc/haproxy/haproxy.cfg` file that is delivered with Cloudant Local includes commented-out configuration lines that you must uncomment to enable SSL. Follow these instructions to configure HAProxy for SSL connections.

As root, open the following file on each load balancer.

```
/etc/haproxy/haproxy.cfg
```

Locate the following two lines in the `haproxy.cfg` file that are commented out, and uncomment them by removing the # at the start of the line.

```
#bind :443 ssl crt /etc/haproxy/certificate.pem
#redirect scheme https if dashboard_assets !{ ssl_fc }
```

After you update the modifying the `haproxy.cfg` file, you must restart HAProxy.

Validate your SSL connection

Check that your SSL connection is working correctly.

Connect to the load balancer with a web browser by using the https protocol. This example shows how to connect to a load balancer with the https protocol.

```
https://lb1centos.princeton.usnj.ibm.com/dashboard.html
```

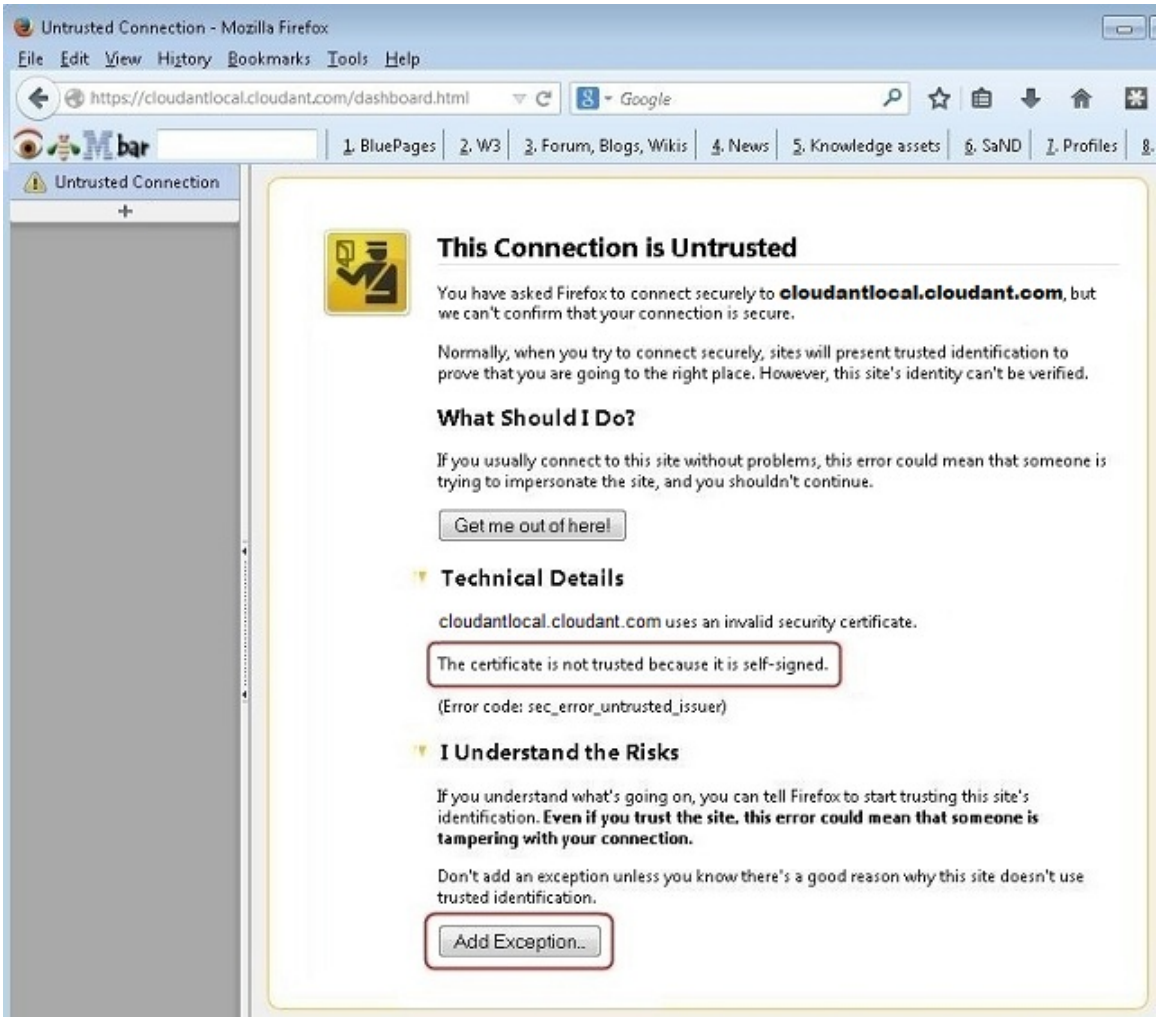
If your SSL connection is working correctly, the Cloudant Local dashboard appears in your browser.

Note: If you are using self-signed security certificates, your browser warns you that the SSL certificates are not from a recognized certificate authority.

Display and confirm your untrusted certificate

If you use a certificate that was validated by a certificate authority (CA), this step is not needed.

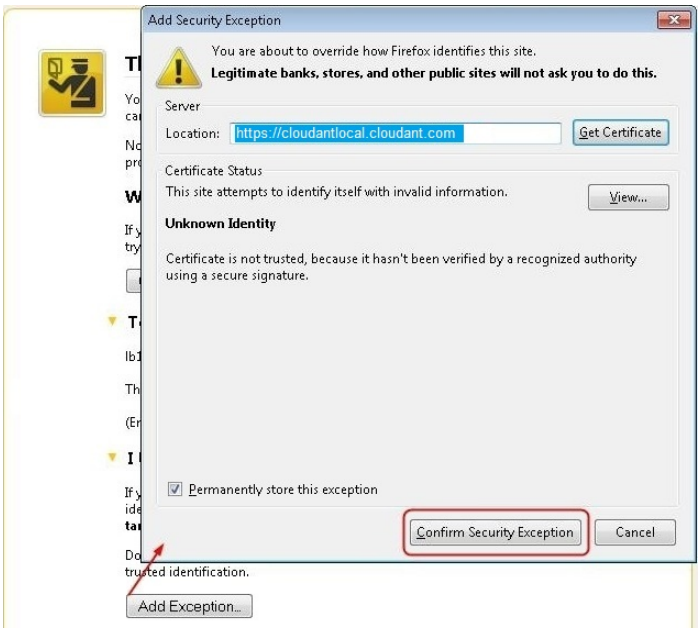
If you use a self-signed certificate, you must confirm your certificate when the following untrusted connection message is displayed.



Do the following steps to accept and confirm the untrusted certificate.

1. Click **Add Exception** on the This Connection is Untrusted message.

2. When an Add Security Exception window is displayed, do the following steps.
   - Select the **Permanently store this exception** check box so the exception is stored for future use.
   - Click **Confirm Security Exception** to confirm the exception.



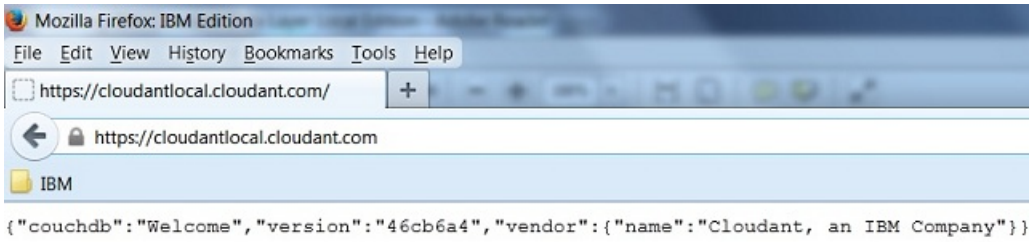
Note: Every user who accesses the load balancer from a separate machine must accept and confirm the self-signed certificate.

View the Load Balancer

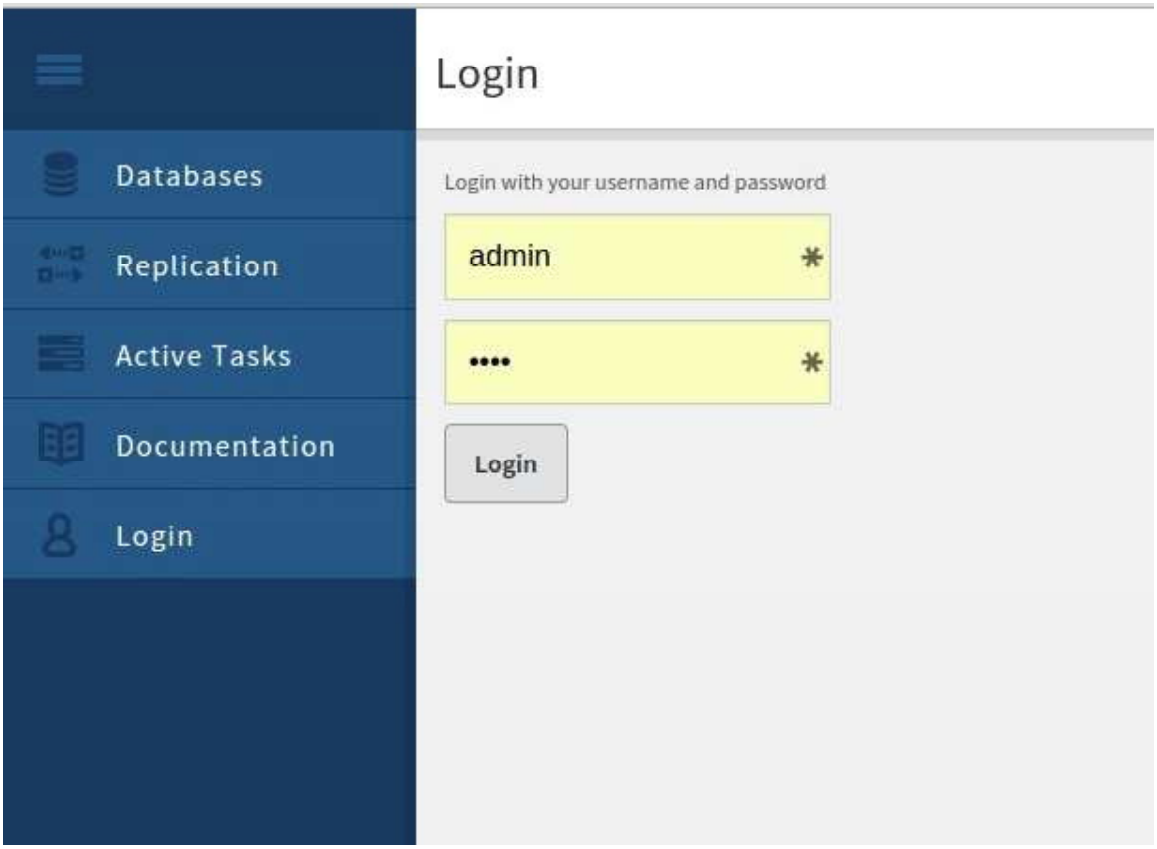
Check that you can view the Load Balancer correctly.

If you are using a properly signed certificate or you accepted an "untrusted" certificate, one of the following screens is displayed.

If you specified `https://cloudantlocal.cloudant.com` or a similar URL, a load balancer welcome message is displayed, similar to the example shown.



If you specified a URL that includes the Dashboard component, `/dashboard.html`, the Cloudant Dashboard (or the Login screen for the dashboard) is displayed. This example shows the user specified `dashboard.html` as part of the URL.



Connect load balancer and database nodes by using SSL

If you are using SSL to connect your Cloudant Local load balancer and database nodes, make these load balancer configuration changes.

If you use SSL for communication between load balancer and database nodes, the final step is to enable the secure communication on each load balancer node.

Do this using the following steps.

1. Copy the `ca.pem` file to each load balancer node. For more information about creating the `ca.pem` file, see "Generate the certificate authority file for a database node" on page 27.
2. In the `haproxy.cfg` file, make the following changes.
   a. Find the section label.
   ```
   ######
   # NOTE: Specify the appropriate hostnames and IP addresses below.
   ######
   ```
   b. Ensure all the database nodes are listed.
   c. For each server, change the port from `5984` to `6984`.
   d. Add the following text to the end of each "server" line.
   ```
   ssl verify required ca-file /<file_location>/ca.pem
   ```
3. Save and close the `haproxy.cfg` file.
4. Restart **`haproxy`**.

Configuring SSL on your database nodes

Use these instructions to configure Secure Sockets Layer (SSL) for Cloudant Local on each database node in your cluster. This method enables SSL communication between load balancers and database nodes.

Follow these steps to configure SSL for Cloudant Local on each database node in your cluster:

- "Generate the certificate authority file for a database node"
- "Generate the server certificate file for a database node"
- "Copy the SSL security files to the database node" on page 28
- "Enable SSL security on the database node" on page 28

Generate the certificate authority file for a database node

If you are using self-signed certificates, generate a certificate authority file for the database nodes.

If you are using certificates that are provided by a third party, you can proceed directly to "Copy the SSL security files to the database node" on page 28.

To use self-signed certificates on database nodes, you must generate a certificate authority (CA) file. Only one CA file is required. The same file is used by all the database nodes and all the load balancer nodes within your cluster. The file can be generated on any server.

To generate a self-signed CA file, use the following commands:

```
openssl genrsa -out privkey.key
openssl req -new -x509 -key privkey.key -out ca.pem
```

The resulting `ca.pem` file is used on each database node and each load balancer node.

Note: To use the `ca.pem` file on a load balancer node, follow the instructions in "Connect load balancer and database nodes by using SSL" on page 26.

Generate the server certificate file for a database node

If you are using self-signed certificates, generate a unique server certificate file for each database node.

If you are using certificates provided by a third party, you can proceed directly to "Copy the SSL security files to the database node" on page 28.

Each database node requires its own server certificate file to identify itself. A unique name is required for each server certificate. In the instructions that follow, replace each instance of the phrase `serverX` with whatever name you choose. Any name can be used, as long as it is unique amongst the servers in your cluster.

To generate server certificate file, use the following commands.

```
openssl genrsa -out serverX.key
openssl req -new -key serverX.key -out serverX.req
openssl x509 -req -in serverX.req -CA ca.pem -CAkey privkey.key -set_serial 01 -out serverX.pem
```

The `ca.pem` file was generated in "Generate the certificate authority file for a database node."

Note: The `serverX.key` file is the 'secret' necessary to unlock self-generated certificates. Be careful to protect this file with appropriate access controls and security permissions.

Copy the SSL security files to the database node

To enable SSL security, the prerequisite security certificate files must be available on the database node.

Copy the following files to any convenient location within the file system of the respective database node.

```
ca.pem
serverX.key
serverX.pem
```

Note: The `serverX.key` file is the 'secret' necessary to unlock self-generated certificates. Be careful to protect this file with appropriate access controls and security permissions.

The `ca.pem` file is the certificate authority (CA) file, generated in the "Generate the certificate authority file for a database node" on page 27 step, or provided by a third party. The same `ca.pem` file is used by all the database nodes.

The `serverX.key` and `serverX.pem` files are unique and specific to each database node. They were generated in "Generate the server certificate file for a database node" on page 27, or provided by a third party.

Enable SSL security on the database node

Configure the database node to use the SSL security files.

Update the `/opt/cloudant/etc/local.ini` file on each database node.

1. Enable the **`https`** daemon by modifying the **`httpsd`** line in the **`[Daemons]`** section.

   ```
   [daemons]
   httpsd = {chttpd, start_link, [https]}
   ```

2. Provide links to the `ca.pem`, `serverX.key` and `serverX.pem` files.

   ```
   [ssl]
   cacert_file = /<filelocation>/ca.pem
   cert_file = /<filelocation>/serverX.pem
   key_file = /<filelocation>/serverX.key
   ```

3. Save and close the `/opt/cloudant/etc/local.ini` file.
4. Restart Cloudant on the database node.

Note: For more information about configuring Cloudant Local for SSL-based secure connections, see Secure Socket Level Options.

Connect load balancer and database nodes by using SSL

If you are using SSL to connect your Cloudant Local load balancer and database nodes, make these load balancer configuration changes.

If you use SSL for communication between load balancer and database nodes, the final step is to enable the secure communication on each load balancer node.

Do this using the following steps.

1. Copy the `ca.pem` file to each load balancer node. For more information about creating the `ca.pem` file, see "Generate the certificate authority file for a database node" on page 27.
2. In the `haproxy.cfg` file, make the following changes.
   a. Find the section label.

      ```
      ######
      # NOTE: Specify the appropriate hostnames and IP addresses below.
      ######
      ```

   b. Ensure all the database nodes are listed.

c. For each server, change the port from `5984` to `6984`.
d. Add the following text to the end of each "server" line.
   ```
   ssl verify required ca-file /<file_location>/ca.pem
   ```
3. Save and close the `haproxy.cfg` file.
4. Restart **haproxy**.

Chapter 9. Determining cluster health with the Metrics application

The Cloudant Metrics web application displays statistics and data about the health of your Cloudant Local cluster.

Cloudant Metrics

You can use the Metrics application to spot potential problems in your Cloudant cluster before they become serious issues and adversely affect the performance of your cluster. A glossary describes the various metrics available on the Metrics web application.

Displaying the Metrics application

This example shows the default URL for the Metrics web application.

protocol`://`*loadbalancer.company.com*`/metrics_app/statistics/index.html`

Replace the following variables in the default URL for the Metrics web application.

- Replace *protocol* with `http` or `https` if you configured Cloudant Local to use Secure Sockets Layer (SSL). For more information about SSL, see Chapter 8, "Configuring SSL," on page 21.
- Replace *loadbalancer.company.com* with your load balancer hostname, such as `cloudantlocal.cloudant.com`.
- If needed, replace *metrics_app* with the `METRICS_DBNAME` you entered in the `metrics.ini` file and append `_app` to it. The default value for that field is **`metrics_app`**. Therefore, if you did not change that default value, enter the literal **`metrics_app`** as part of the URL.

This example URL shows the Metrics web application URL with the user specified to display the Metrics application.**`https://cloudantlocal.cloudant.com/metrics_app/statistics/index.html`**

The first time you log in to the Metrics application the Metrics login page is displayed. The example shows the data that was displayed on the Metrics application for **Database Read/Write Rate** and **Document Read/Write Rate** in graph form.

Chapter 10. Configuring logging

Use these instructions to configure logging for your Cloudant Local installation.

Overview

Several components within a Cloudant Local installation generate log files. These log files are valuable for monitoring performance, troubleshooting, and other administrative tasks. In particular, the database node and load balancer components of your Cloudant Local cluster generate log information.

By default, a Cloudant Local installation configures local logging for these components. The effect is that all the log information is stored on the same system as the active component.

For database nodes, the default of local logging is suitable only for test or non-production environments.

Note: For a production environment, you must configure the database node for remote logging. Remote logging helps ensure optimum system performance because local logging adversely affects performance. Remote logging also makes it easier to combine logs from all database nodes.

For the load balancer node, you can keep the default logging value: local logging. For heavy load, or to log to a separate logging server, the load balancer logs can be configured for remote logging.

As an option for managing the logs of your Cloudant Local cluster, you might choose to use the load balancer node as the central logging server for your cluster. In other words, all the database nodes could be configured to use the load balancer server as the remote logging server. The load balancer itself would continue to use the default local logging configuration. This option has the advantage that all the key logs from the cluster nodes are available on the load balancer server.

Summary of logs and locations

Table 1. Database nodes

Log type and purpose	**Configuration file**	**Local logging default log file location**	**Remote logging default `rsyslog` *facility***

Cloudant database core logs
 	
Contains data about events such as runtime errors, warnings, or crashes that were encountered by the Cloudant database core.
	`/opt/cloudant/etc/sys.config`	`/var/log/cloudant/cloudant.log`
		
		`/var/log/cloudant/cloudant-crash.log`
			`local2.*`
			
			`/var/log/cloudant/cloudant.log`

Clouseau Search Service log
 	
Contains information after the search indexes are built and committed. Also includes service start or stop status, and any errors encountered.
	`/opt/cloudant/etc/log4j.properties`	`/var/log/cloudant/clouseau.log`
		Note: Uses **`rsyslog`** *facility* `local5` for local logging.
			`local5.*`
			
			`/var/log/cloudant/clouseau.log`

Table 1. Database nodes (continued)

Log type and purpose	Configuration file	Local logging default log file location	Remote logging default **rsyslog** *facility*

Metrics Service log
 	
Contains information about service start or stop status, and any errors in the Cloudant Metrics data gathering service.
	`/opt/cloudant/etc/metrics.ini`	`/var/log/cloudant/metrics.log`
		Note: Uses **rsyslog** *facility* `local3` for local logging.
			`local3.*`
			
			`/var/log/cloudant/metrics.log`

Apache CouchDB Service log Contains information about service start or stop status, and any errors in the Cloudant Apache CouchDB.	`/opt/cloudant/etc/local.ini`	`/var/log/cloudant/cloudant-svlogd/current`	None.

Load balancer node

Log type and purpose	Configuration file	Local logging default log file location	Remote logging default **rsyslog** *facility*

HAProxy logs
 	
Contains information about service start or stop status, and runtime errors. Can be extended to record access and other request information.
	`/etc/haproxy/haproxy.cfg`	`/var/log/haproxy.log`
		Note: Uses **rsyslog** *facility* `local4` for local logging.
			`local4.*`
			
			`/var/log/haproxy.log`

NGINX logs
 	
Contains information about service start or stop status, any errors, and access details.
	`/etc/nginx/nginx.conf`	`/var/log/nginx/access.log`
		
		`/var/log/nginx/error.log`
			None.

All nodes

Log type and purpose	Configuration file	Local logging default log file location	Remote logging default **rsyslog** *facility*

System logs
 	
General system and security logs on the server.
		On Debian and Ubuntu:
		
		`/var/log/auth.log`
		
		`/var/log/syslog` On Red Hat-derived Linux distributions only:
		
		`/var/log/secure`
		
		`/var/log/messages`
			None.

Configuring the Remote Logging server to use `rsyslog`

Before configuring remote logging, the remote logging server must be set up and confirmed to be operational. The remote logging server can be any separate enterprise logging server that used by operations for centralized logging.

Alternatively, you might choose to use the Cloudant load balancer server as the remote log server to centralize the logs from all the nodes.

Confirm or enable a remote logging server by following these steps.

1. Check whether the **syslog** daemon is running on the **syslog** server.

   ```
   [root@lb1 tmp]# ps -ef | grep syslog
   root      6306     1  0 11:15 ?        00:00:00 /sbin/rsyslogd -i /var/run/syslogd.pid -c 5

   [root@lb1 tmp]# service rsyslog status
   rsyslogd (pid  6306) is running...
   ```

   If the service is not running, start **rsyslog**.

   ```
   service rsyslog start
   ```

   Note: If **rsyslog** is not already installed, install it on the remote log server and then start it. To install **rsyslog** on Debian or Ubuntu, use **apt-get**. To install **rsyslog** on Red Hat-derived Linux™ distributions only, use **yum**.

2. Check the configuration in `/etc/rsyslog.conf`.

   Ensure that the `ModLoad` and `UDPServerRun` entries are not commented out.

   ```
   # Provides UDP syslog reception
   $ModLoad imudp
   $UDPServerRun 514
   ```

3. Create a file `cloudant.conf` in the `/etc/rsyslog.d` directory and add the following lines.

   ```
   local2.*   /var/log/cloudant/cloudant.log
   local3.*   /var/log/cloudant/metrics.log
   local5.*   /var/log/cloudant/clouseau.log
   ```

   This configuration ensures that **rsyslog** receives the log messages according to the different *facility* configurations.

   Note: A sample of the `cloudant.conf` file is available in `/etc/rsyslog.d` on any of the database nodes.

4. For the HAproxy, create a file `haproxy.conf` in the `/etc/rsyslog.d` directory. The file must contain the following line.

   ```
   local4.* /var/log/haproxy.log
   ```

   This configuration ensures that **rsyslog** receives the log messages according to the different *facility* configurations.

   Note: A sample of the `haproxy.conf` file is available in `/etc/rsyslog.d` on the load balancer nodes.

5. Restart **syslog** with the following command.

   ```
   service rsyslog restart
   ```

6. Check that the **syslog** daemon is running, as described in step 1.

Troubleshooting a remote logging configuration

If the source is configured to send log messages to a remote logging server, but the messages are not seen in the logs, perform the following checks.

- Ensure that the **syslog** *facility* property is configured on the source by checking the source log configuration file. Similarly, check the configuration on the remote logging server by inspecting the `/etc/rsyslog.conf` or `/etc/rsyslog.d/*.conf` files to see where the *facility* writes the log.

  Ensure that the *facility* values are not in conflict with the different types of logs.

  You can use different *facility* values from `local2` through to `local7` inclusive. For more information about *facility* values, see RFC 3164.

  The following table lists the values used by Cloudant Local.

facility	**Purpose**
`local2`	Cloudant database logs
`local3`	Metrics logs
`local4`	HAProxy logs
`local5`	Clouseau logs

- Change the logging level on the source nodes temporarily to level `info` to generate more logging activity. After you complete your verification checks, remember to change the logging level back to the default or your preferred level.
- Confirm that the UDP port that is used for remote logging is open on the remote syslog server. The default port is 514. You can modify the port used by changing the value of the `UDPServerRun` entry in the `/etc/rsyslog.conf` file. Confirm that the port value is the same port that the source nodes are using to send log messages.
- Verify that log messages are being sent by the source node, and received by the log server, through the configured port. To check, use a network monitoring tool such as **tcpdump**. Use the tool on the source and log servers, and monitor or`sniff` the logging port.

  For example, by using **tcpdump** to monitor outgoing traffic on the default logging port 514, you would expect to see results similar to the following example.

```
[root@db2 tmp]# sudo tcpdump -n -s 1500 -X port 514

14:33:01.995812 IP 104.131.89.154.58467 > 104.131.35.26.syslog: SYSLOG local2.notice, length: 221
0x0000:  4500 00f9 0000 4000 4011 ec39 6883 599a  E.....@.@..9h.Y.
0x0010:  6883 231a e463 0202 00e5 4eb1 3c31 3439  h.#..c....N.<149
.....
0x0090:  5d20 636c 6f75 6461 6e74 4064 6232 2e61  ].cloudant@db2.a
0x00a0:  6e75 6a2e 6365 6e74 6f73 2e63 6c6f 7564  nuj.centos.cloud
0x00b0:  616e 742d 6c6f 6361 6c2e 636f 6d20 3c30  ant-local.com.<0
0x00d0:  6120 3130 342e 3133 312e 3335 2e32 3620  a.104.131.35.26.
0x00e0:  756e 6465 6669 6e65 6420 4745 5420 2f20  undefined.GET./.
0x00f0:  3230 3020 6f6b 2033 0a                   200.ok.3.
```

  In this example, the database node logs are being monitored, by watching for traffic that is associated with the *facility* `local2`.

  If you are monitoring the incoming traffic on the log server at the same time, you would expect to see results similar to the following example.

```
[root@lb1 tmp]# sudo tcpdump -n -s 1500 -X port 514
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 1500 bytes
14:33:02.193469 IP 104.131.89.154.58467 > 104.131.35.26.syslog: SYSLOG local2.notice, length: 221
0x0000:  4500 00f9 0000 4000 3f11 ed39 6883 599a  E.....@.?..9h.Y.
0x0010:  6883 231a e463 0202 00e5 777d 3c31 3439  h.#..c....w}<149
......
0x0070:  2032 3031 342d 3130 2d30 3320 3134 3a33  .2014-10-03.14:3
0x0090:  5d20 636c 6f75 6461 6e74 4064 6232 2e61  ].cloudant@db2.a
0x00a0:  6e75 6a2e 6365 6e74 6f73 2e63 6c6f 7564  nuj.centos.cloud
0x00b0:  616e 742d 6c6f 6361 6c2e 636f 6d20 3c30  ant-local.com.<0
0x00e0:  756e 6465 6669 6e65 6420 4745 5420 2f20  undefined.GET./.
0x00f0:  3230 3020 6f6b 2033 0a                   200.ok.3.
```

Other enterprise services might also be sending messages to the log server through the same port. When you verify incoming traffic on the logging server, check which node or server the message came from.

- You might need to edit `/etc/rsyslog.conf` and disable any other default rules that could cause log duplication. Log duplication occurs when log messages are written to multiple files.
- Automatic log rotation might not be configured. You can enable automatic log rotation by using **`logrotate`**. More information about **`logrotate`** is available at http://linux.die.net/man/8/logrotate. An alternative option is to use output channel scripts, as described at http://www.rsyslog.com/doc/log_rotation_fix_size.html.

Configuring local system logs

In addition to the Cloudant Local database node and load balancer node logs, several local system logs must be enabled and captured. After you enable these logs, you can see security audit and system level information.

Debian and Ubuntu use the following logs.

- `/var/log/auth.log`
- `/var/log/syslog`

Red Hat-derived Linux distributions only use the following logs.

- `/var/log/secure`
- `/var/log/messages`

If the logs are not enabled on your OS platform by default, refer to platform-spescific information for details about configuring them.

Chapter 11. (Optional) Configuring database-level security

Use these instructions to configure database-level security for Cloudant Local. This step is not required to configure Cloudant Local.

Cloudant Local uses the CouchDB security model for database-level security. For more information about that security model, see the following documents on the CouchDB website.

- http://docs.couchdb.org/en/latest/intro/security.html
- http://docs.couchdb.org/en/latest/api/database/security.html
- http://guide.couchdb.org/draft/security.html

If global read/write access is unacceptable and user- and role-based authorization to individual databases is required, do the following steps.

1. Create a `_users` database.
2. Create a document in the `_users` database for each non-administrator account that requires access to your databases.
3. For each database that you create, add a `_security` document that identifies who is allowed to access that database as a member or reader. For more information about the syntax that is used in that document and the API call, see http://docs.couchdb.org/en/latest/api/database/security.html#put--db-_security.
4. Use this command to validate the security settings on each database.

   curl -u username:password https://hostname.customer.com/dbname

   Replace the following variables in this command.

   - Replace **hostname.customer.com** with the hostname of the configured cluster load balancer.
   - Replace **dbname** with the name of the database.
   - To test anonymous access, omit the **-u** flag.
   - Replace **username:password** with the user ID and password for any non-administrator account that you created.

If you are not familiar with the CouchDB model for database-level security, the following example illustrates how database-level security works in CouchDB.

Database-level security example

The following example shows how database-level security works in CouchDB.

- Confirm that the **admin** user is identified in the `local.ini` file.
- Create two sample databases, **db1** and **db2**.
- Create two non-admin users, **member** and **outsider**.
- Configure database security to limit access to the **db1** database, but not the **db2** database.
- Grant read permission to the **member** user for the **db1** database, but not the **outsider** user. (By default, the **admin** user also has read and write access to all databases. In CouchDB and Cloudant, an admin user is an administrator, a super user, or root who is allowed to do anything to a CouchDB installation.)
- Run a number of checks to validate that security settings were properly applied.

Follow these steps to configure database-level security. For reference purposes, the steps are written from the perspective that you, the reader, are doing the steps.

1. Confirm that the **admin** user is specified in the `local.ini` file in the `[admins]` section.

2. Use these commands to create the **db1** and **db2** databases.

   ```
   $ curl -X PUT -u admin:password https://loadbalancer.example.com/db1
   $ curl -X PUT -u admin:password https://loadbalancer.example.com/db2
   ```

   If you use these commands, replace the following variables in these commands and all subsequent sample commands.

   - Replace **admin:password** with the user ID and password for your admin user.
   - Replace **loadbalancer.example.com** with the hostname of your configured cluster load balancer.
   - Replace **db1** with the name of your database.

   You receive an {"ok":true} message after the databases are created.

3. Acting as an unauthenticated user, use these commands to create a test document in the **db2** database.

   ```
   $ curl -X POST http://loadbalancer.example.com/db2 \
   > -H "Content-Type: application/json" \
   > -d "{\"likes\": \"turtles\"}"
   ```

   You receive an {"ok":true} message after the document is created, and the message includes the document ID and revision number for the document.

   ```
   {"ok":true,"id":"171806ad7968475970bf5450e91a5259","rev":"1-
   e6accc814683c1cadf6c74b492570c42"}
   ```

   The unauthorized user was allowed to add the test document because no database-level security is in place for **db2**. By default, CouchDB and Cloudant Local allow anonymous reads and writes, which is colloquially called "admin party" mode. Many CouchDB users choose to use this approach and put their databases behind a firewall without configuring any additional database-level security.

4. Use your administrator credentials to create a **_users** database and two test users that are named **member** and **outsider**. Replace all variables in these examples as described in step 2.

   - Use this command to create the **_users** database.

     ```
     $curl -X PUT http://loadbalancer.example.com/_users -u admin:password
     ```

     You receive an {"ok":true} message after the database is created.

   - Use this command to create the **member** user.

     ```
     $ curl -X PUT http://loadbalancer.example.com/_users/org.couchdb.user:member \
     > -H "Accept: application/json" \
     > -H "Content-Type: application/json" \
     > -d '{"name": "member", "password": "f1wu8tvp5gGe", "type": "user"}' \
     > -u admin:password
     ```

     The password for the **member** user is specified on the fourth line.

     You receive an {"ok":true} message after the document is created, and the message includes the document ID and revision number for that user document as seen in this example.

     ```
     {"ok":true,"id":"org.couchdb.user:member","rev":"1-d9bdb39bac9288b154cdf5cc4d643ce9"}
     ```

   - Use this command to create the **outsider** user.

     ```
     $ curl -X PUT http://loadbalancer.example.com/_users/org.couchdb.user:outsider \
     > -H "Accept: application/json" \
     > -H "Content-Type: application/json" \
     > -d '{"name": "outsider", "password": "ncGfv9bcDBn0", "type": "user"}' \
     > -u admin:password
     ```

     The password for the **outsider** user is specified on the fourth line.

     For example, the message {"ok":true} displays after the document is created, and the message includes the document ID and revision number for that user document.

     ```
     {"ok":true,"id":"org.couchdb.user:outsider","rev":"1-7d2b68aca9a2634fee51c49e4d7d39ca"}
     ```

     Both users are now listed as valid users in the **_users** database, and both users were given permission to create JSON documents. However, neither user was granted authority to add documents to a database that includes database-level security. You can add database-level security by adding the **_security** object, as described in the next step.

5. Use the following commands to set up security on just one of your test databases. Replace all variables in this example as described in step 2.

```
$ curl -X PUT http://loadbalancer.example.com/db1/_security \
> -u admin:password \
> -H "Content-Type: application/json" \
> -d "{"admins": {"names": ["admin"], "roles": []}, "members": {"names": ["member"], "roles": []}}"
```

You receive an {"ok":true} message after the security is set up.

In the preceding example, security is set up for the **db1** database only. Moreover, only the **member** user is granted authority to create JSON documents in that database, other than the **admin** user, who always has authority to do so.

6. Use the following commands to confirm that access to the **db1** database is restricted to authorized users only. Replace all variables in these examples as described in step 2 on page 40.
   - Use this command to access the **db1** database as an unauthorized user, that is, a user who is not in the **_users** database.

     ```
     $ curl http://loadbalancer.example.com/db1
     ```

     Since the user is not authorized to access the **db1** database, you receive the following error message.

     ```
     {"error":"unauthorized","reason":"You are not authorized to access this db."}
     ```

   - Use this command to access the **db1** database as the **outsider** user.

     ```
     $ curl http://loadbalancer.example.com/db1 -u outsider:ncGfv9bcDBn0
     ```

     Since the **outsider** user also is not authorized to access the **db1** database, you receive the following error message.

     ```
     {"error":"unauthorized","reason":"You are not authorized to access this db."}
     ```

   - Use this command to access the **db1** database as the **member** user.

     ```
     $ curl http://loadbalancer.example.com/db1 -u member:f1wu8tvp5gGe
     ```

     Since the **member** user is authorized to access the database, you receive a message like the one in this example.

     ```
     {"db_name":"db1","update_seq":[10,"g1AAAAIDeJzLYWBg4MhgTmFQT87JL01JzCtxSEky1INxc
     vKTE3PgPL281JIcoAamRIYk-f___2clMpKsNUkBSCbZk6vbAaQ7nlzdCSDd9WTqzmMBkgwNQApowPysR
     GYyTVgAMWE_-W44ADHhPvkmPICYAAqHLABhzbHo"],"sizes":{"file":107887,"external":76,"
     active":1942},"purge_seq":0,"other":{"data_size":76},"doc_del_count":1,"doc_coun
     t":0,"disk_size":107887,"disk_format_version":6,"data_size":1942,"compact_runnin
     g":false,"instance_start_time":"0"}
     ```

   - Use this command to access the **db1** database as the **admin** user.

     ```
     $ curl http://loadbalancer.example.com/db1 -u admin:password
     ```

     Since the **admin** user is authorized to access any database, including **db1**, you receive a message like the one in this example.

     ```
     {"db_name":"db1","update_seq":[11,"g1AAAAIDeJzLYWBg4MhgTmFQT87JL01JzCtxSEky1INxc
     vKTE3PgPL281JIcoAamRIYk-f___2clMpKsNUkBSCbZk6vbAaQ7nlzdCSDd9WTqzmMBkgwNQApowPysR
     BYyTVgAMWE_-W44ADHhPvkmPICYAAqHLABilbHp"],"sizes":{"file":128367,"external":175,
     "active":2282},"purge_seq":0,"other":{"data_size":175},"doc_del_count":1,"doc_co
     unt":1,"disk_size":128367,"disk_format_version":6,"data_size":2282,"compact_runn
     ing":false,"instance_start_time":"0"}
     ```

7. Use the following commands to confirm that only authorized users can add a JSON document to the **db1** database. Replace all variables in these examples as described in step 2.
   - Use this command to add a document to the **db1** database as the **outsider** user.

     ```
     $ curl -X PUT http://loadbalancer.example.com/db1/foo \
      -H "Accept: application/json" \
      -H "Content-Type: application/json" \
      -d '{"bar": "baz"}' \
      -u outsider:ncGfv9bcDBn0
     ```

     Since the **outsider** user is not authorized to access the database, you receive a message like the one in this example.

```
{"error":"unauthorized","reason":"You are not authorized to access this db."}
```

- Use this command to add a document to the **db1** database as the **member** user.

```
$ curl -X PUT http://loadbalancer.example.com/db1/foo \
 -H "Accept: application/json" \
 -H "Content-Type: application/json" \
 -d '{"bar": "baz"}' \
 -u member:f1wu8tvp5gGe
```

  Since the **member** user is authorized to add documents to the database, you receive a {"ok":true} message like the one in this example.

```
{"ok":true,"id":"foo","rev":"1-c86e975fffb4a635eed6d1dfc92afded"}
```

  The message includes the following information about the document: its document ID, which is foo, and its revision number.

- Use this command to add a document to the **db1** database as the **admin** user.

```
$ curl -X PUT http://loadbalancer.example.com/db1/foo2 \
 -H "Accept: application/json" \
 -H "Content-Type: application/json" \
 -d '{"bar": "baz"}' \
 -u admin:password
```

  Since the **admin** user is authorized to add documents to the database, you receive an {"ok":true} message like the one in this example.

```
{"ok":true,"id":"foo2","rev":"1-c86e975fffb4a635eed6d1dfc92afded"}
```

  The message includes the following information about the document.

  – The document ID, which is foo2
  – The revision number

8. Use the following commands to confirm that only authorized users can create a design document. Design documents are similar to materialized views or indexes in a relational database management system (RDBMS). In Couch DB and Cloudant, only admin-level users can create design functions like views, shows, and others, as illustrated in this step. Replace all variables in these examples as described in step 2.

   - Use this command to create a design document in the **db1** database as the **outsider** user.

```
$ curl -X PUT http://loadbalancer.example.com/db1/_design/all \
 -H "Accept: application/json" \
 -H "Content-Type: application/json" \
 -d '{"language":"javascript", \
  "views":{"all":"mapin ":function(doc){emit(doc._id, 1)};","reduce": "_count"}}' \
 -u outsider:ncGfv9bcDBn0
```

     Since the **outsider** user is not authorized to create a design document, you receive a message like the one in this example.

```
{"error":"unauthorized","reason":"You are not authorized to access this db."}
```

   - Use this command to create a design document in the **db1** database as the **member** user.

```
$ curl -X PUT http://loadbalancer.example.com/db1/_design/all \
 -H "Accept: application/json" \
 -H "Content-Type: application/json" \
 -d '{"language":"javascript", \
  "views":{"all":{"map":"function(doc){emit(doc._id, 1)};","reduce":"_count"}}}' \
 -u member:f1wu8tvp5gGe
```

     Since the **member** user is not authorized to create a design document, you receive a message like the one in this example.

```
{"error":"unauthorized","reason":"You are not a db or server admin."}
```

     As the message indicates, you must be an admin-level user to create a design document.

   - Use this command to create a design document in the **db1** database as the **admin** user.

```
$ curl -X PUT http://loadbalancer.example.com/db1/_design/all \
 -H "Accept: application/json" -H "Content-Type: application/json" \
 -d '{"language":"javascript", \
  "views":{"all":{"map":"function(doc){emit(doc._id,1)};","reduce":"_count"}}}'\
 -u admin:password
```

Since the **admin** user is authorized to create a design document, you receive a message like the one in this example.

```
{"ok":true,"id":"_design/all","rev":"1-5c0878a3c1cabf82004ed85113fa59c6"}
```

The message includes the following information about the design document.

- The document ID, which is `_design/all`
- The revision number

Chapter 12. Authenticating with LDAP

Use these instructions to configure Lightweight Directory Access Protocol (LDAP) authentication and authorization for Cloudant Local.

Cloudant Local has its own security model with a self-contained authentication and authorization system. The authentication and authorization system relies on `ini` files and user databases to store user data, including user names (uids), passwords, and user roles for accessing resources in the database. If your organization uses an LDAP service for access control, you can delegate authentication and authorization services to LDAP when you access a Cloudant Local database. The `ldap_auth` configuration permits you to use LDAP with Cloudant Local.

At a high level, when `ldap_auth` is configured, it replaces the default basic and cookie authentication and authorization. It ignores any admin users configured in the `ini` files and user databases. Instead, `ldap_auth` validates user credentials by using the configured LDAP service. After you authenticate your credentials with the LDAP service, `ldap_auth` determines the user's roles, which determine what the user is authorized to access, based on the LDAP groups where the user is a member.

Important: If you want to use LDAP and the Metrics application together, you must be aware that the Metrics application requires specific authentication credentials and roles to exist in the LDAP database to work properly.

The Metrics application stores its authentication credentials in the `/opt/cloudant/etc/metrics.ini` file. The Metrics application's default authentication credentials include username `admin` and password `pass`. You must create a user in your LDAP database that has both `_admin` and `server_admin` roles and whose credentials match the credentials set in the `metrics.ini` file. See Chapter 9, "Determining cluster health with the Metrics application," on page 31.

Configuring LDAP authentication

You configure LDAP authentication and authorization in the `ini` configuration file under the `ldap_auth` section. See the following example of the configuration information in the `local.ini` file in `/opt/cloudant/etc`.

```
[ldap_auth]
use_ssl = false
servers = 10.35.186.185
searcher_dn = cn=Manager,dc=example,dc=com
searcher_password = secretsanta
group_base_dn = dc=example,dc=com
```

Note: You must configure at least one of the available parameters inside the `ldap_auth` section of an `ini` configuration file. If you fail to explicitly configure at least one `ldap_auth` parameter, the system uses the default basic and cookie authentication and authorization handlers instead, and Cloudant Local does not use `ldap_auth` as its auth handler.
Modify the parameters and their associated default values (in parentheses) in the following list.

`servers* (127.0.0.1)`
: Supports one or more LDAP servers, which default to a single host, but might also be a comma-separated list.

  Note: All servers must use the same port, which is a limitation of the underlying eldap library.

`port (389)`
: LDAP server port for decrypted communication.

ssl_port (636)
LDAP server port for encrypted communication.

use_ssl (true)
If *true*, use TLS to encrypt traffic to LDAP servers.

timeout (5000)
Milliseconds to wait for a response from an LDAP server before an error occurs.

user_base_dn* (ou=users,dc=example,dc=com)
Defines a directory location to use during a search for users.

user_classes (person)
Defines which objectClasses indicate a particular entry as a user during search.

user_uid_attribute (uid)
Defines which attribute maps to a user name.

group_base_dn* (ou=groups,dc=example,dc=com)
Defines a directory location to use during a search for groups.

group_classes (posixGroup)
Defines which objectClasses indicate a particular entry as a group during search.

group_member_attribute (memberUid)
Defines which group attribute maps to user **uid**.

group_role_attribute (description)
Defines which group attribute maps to a particular role.

searcher_dn* (uid=ldapsearch,ou=users,dc=example,dc=com)
Defines the distinguished name (DN) to use while ldap_auth searches for users and groups.

searcher_password* (secret)
Defines the password for the **searcher_dn** parameter.

user_bind_dns ([ ])
Defines one or more base DNs into which the authenticating user's user name can be inserted as the **user_uid_attribute** and used to bind to it directly.

*The default values for this parameter are only useful for testing. Change the parameter's value to match your environment.

Setting up `ldap_auth` to work with Cloudant Local

Cloudant Local LDAP authentication, `ldap_auth`, implements basic and cookie-based authentication handlers to obtain the roles associated with a particular user. Using `ldap_auth` with basic authentication requires no client-side changes. A properly configured database automatically attempts to authenticate via LDAP by using the supplied credentials. For cookie authentication, **POST** credentials to the `_session` endpoint to obtain an **AuthSession** cookie, which contains a signed hash of its contents: name, time issued, and authorized roles. Subsequent requests that use that cookie automatically use the roles in the cookie until it expires. Modify the following configuration parameters in the `[couch_httpd_auth]` section.

timeout (600)
Seconds until the **AuthSession** cookie expires.

secret The shared secret that is used to sign the **AuthSession** cookie that is created when the cluster is provisioned.

Important: LDAP uses **AuthSession** as the default auth. If you alternately log in by using default auth and `ldap_auth`, you must clear the cookies between sessions to avoid a conflict.

Optimizing LDAP Authentication and Authorization

The `ldap_auth` function uses two different modes of operation that depend on whether the **`user_bind_dns`** parameter is defined. Both modes return a list of roles associated with a user upon successful authentication.

By default, `ldap_auth` opens a connection to an LDAP server and binds the connection handle with the searcher DN and password. It then uses that connection to search for a user record with the authenticating user name. If a matching user DN is found, it opens a second connection to the server and attempts to bind that connection by using the user name and password credentials. If the bind is successful, it closes that connection and uses the original "searcher" bound connection to search for groups that contain the **`group_member_attribute`** parameter that matches the user's user name. Those groups have a **`group_role_attribute`** parameter that indicates the actual roles for the user.

If you know in advance that all your users' DNs can be constructed by inserting their user names into the following pattern, **`$uid_attribute=$username,$user_bind_dn (for example, uid=jay,ou=users,dc=example,dc=com)`**, you can configure multiple **`user_bind_dns`** parameters, and the `ldap_auth` does not use the searcher DN or the **`user_base_dn`**. Instead, `ldap_auth` binds directly to the constructed user DNs. If the bind is successful, the bound connection is used again to make the same group search described earlier. This technique can eliminate extra network round trips.

Defining LDAP groups and roles

The method for defining groups and roles varies based on your LDAP server. However, before Cloudant Local and LDAP function together, you must define groups for each Cloudant role. Each role defines the tasks users in that group can perform. Some of the available roles are defined in the following list.

_reader
: Read from a database.

_writer
: Update a database.

After the groups are defined, you must define the users that can perform the tasks in each role.

Chapter 13. Uninstalling Cloudant Local

Use these instructions to uninstall Cloudant Local with the `cast system uninstall` command.

Uninstalling Cloudant Local

When you decide to uninstall Cloudant Local, you must use the same user ID that you used when you first installed Cloudant Local. For example, if you installed as the root user, you must uninstall as the root user.

To uninstall Cloudant Local, follow these instructions.

1. Run the following commands to uninstall Cloudant Local services and the CAST tool.

   ```
   cast system uninstall
   ```

2. Run this command to uninstall the installation wizard.

   ```
   {install-dir}/uninstall/uninstall.bin
   ```

   You have uninstalled Cloudant Local.

Chapter 14. Cloudant Local maintenance tasks

The following tasks1 can help you maintain your Cloudant Local cluster.

Checking the health of a cluster with Weatherreport

Use these instructions to use the Weatherreport utility to check the health of your Cloudant cluster. Weatherreport is a command-line application that provides information about the status of a `dbcore` node or cluster. It is useful in troubleshooting cluster issues, such as increased latencies, low disk space, or node failures.

Running Weatherreport

To use the Weatherreport utility, follow these steps.

1. As root or `cloudant` OS user, update that user profile file with the following PATH or set the PATH in your current SSH session by running the following command.

   `export PATH=$PATH:/opt/cloudant/bin`

2. SSH into a `dbcore` node and use this command to run Weatherreport.

   `/opt/cloudant/bin/weatherreport`

By default, `weatherreport` carries out checks for the local node. If you run it with the `--all-nodes` option, the whole cluster is checked.

Weatherreport checks

The following list shows what `weatherreport` checks when you run it.

```
$ /opt/cloudant/bin/weatherreport --list
Weatherreport check list

custodian            Shard safety/liveness checks
disk                 Data directory permissions and atime
internal_replication Check the number of pending internal replication jobs
ioq                  Check the total number of active IOQ requests
mem3_sync            Check there is a registered mem3_sync process
membership           Cluster membership validity
memory_use           Measure memory usage
message_queues       Check for processes with large mailboxes
node_stats           Check useful erlang statistics for diagnostics
nodes_connected      Cluster node liveness
process_calls        Check for large numbers of processes with the same current/initial call
process_memory       Check for processes with large mailboxes
safe_to_rebuild      Check whether the node can safely be taken out of service
search               Check the local search node is responsive
tcp_queues           Measure the length of tcp queues in the kernel
```

To execute only one of the checks, you can run Weatherreport with the name of the check as the first parameter. For instance, here is how to check for memory use issues only on all nodes.

```
$ /opt/cloudant/bin/weatherreport --all-nodes memory_use
```

A list of the command-line options and their meaning are available by using this command.

```
$ /opt/cloudant/bin/weatherreport --help
```

The `memory_use` check

What it checks.

You can check the amount of free RAM on a `dbcore` node with `memory_use`.

What an error for this check means.

A node is running out of RAM. The `oom` might stop `dbcore` soon, which can cause externally visible errors.

How to fix it.

It is worth attending to this problem quickly. You can check the memory graphs in the metrics application to see how urgent it is and whether memory use is increasing quickly or slowly.

Fast increase.

If it is a sharp increase, a process might be getting a long message queue backup. That probably means you can stop the process. However, before you do that, use `process_info` to find out more about the process. To find processes that are using too much memory within `dbcore`, log in to the node and start a `remsh` session. Find the processes that are using the most memory.

```
# Size is in bytes
(dbcore@db6.bigpark002.cloudant.net)1> recon:proc_count(memory, 10).
[{<0.101.0>,16365344,
  [lager_event,
   {current_function,{gen_event,fetch_msg,5}},
   {initial_call,{proc_lib,init_p,5}}]},
 {<0.267.0>,13090216,
  [rexi_server,
   {current_function,{gen_server,loop,6}},
   {initial_call,{proc_lib,init_p,5}}]},
 {<0.14538.4675>,9210944,
  [{current_function,{gen,do_call,4}},
   {initial_call,{erlang,apply,2}}]},
 {<0.25.0>,2917752,
  [file_server_2,
   {current_function,{gen_server,loop,6}},
   {initial_call,{proc_lib,init_p,5}}]},
....
```

The second parameter to `proc_count` is the number of processes to show, in this case 10. So here `<0.101.0>` is using 16 MB of memory, which is typically nothing to worry about. It is possible for individual processes to use several gigabytes of memory, though. As a rule, if the top few processes are using an order of magnitude more memory, it is worth looking into whether you can stop those processes.

Use `process_info` to find out what process is running.

```
(dbcore@db1.slqs002.cloudant.net)3> process_info(pid(0,9900,0), [current_funct
ion,message_queue_len,initial_call]).
[{current_function,{couch_key_tree,merge_at,3}}, {message_queue_len,2270}, {in
itial_call,{proc_lib,init_p,5}}]
```

Now, use your judgment or contact support to see whether it is safe to stop the process.

Slower increase.

It might be that "garbage" is building up. Check the different types of memory for the node.

```
> recon:node_stats_print(1,1).
{[{process_count,1362},
  {run_queue,0},
  {error_logger_queue_len,0},
  {memory_total,168628304},
  {memory_procs,83944688},
  {memory_atoms,486930},
  {memory_bin,19389792},
```

```
  {memory_ets,4410896}],
 [{bytes_in,76984},
  {bytes_out,275},
  {gc_count,25},
  {gc_words_reclaimed,17015},
  {reductions,11983462}]}
ok
```

Run a garbage collection from a `remsh` on the node.

```
[erlang:garbage_collect(Pid) || Pid <- processes()]
```

If that does not change the situation, check whether the lines in the provided memory information add up to the memory the OS is reporting for `beam.smp` by using top on the node. If a significant discrepancy exists, the node is probably leaking memory, and if the server is close to running out, restart `cloudant` with this command.

```
sv restart cloudant
```

The `message_queues` checks

This check covers a group of checks.

The `couch_db_updater` message queue check

What it checks.

This check monitors the message queues of the various `couch_db_updater` processes. These processes manage access to a logical database; one `couch_db_updater` process exists for each open database shard and one for each open view shard. In general, `couch_db_updater` processes do not exhibit the same pathological failure modes that `couch_file` processes do. Regardless, a failing or backed-up `couch_db_updater` process is likely to cause externally visible problems and must be acted upon.

How to confirm the failure.

SSH to the node where the failure occurred and create the following function in a `remsh`.

```
FindTiredPids = fun(Mod, N) ->
    {monitors, M} = process_info(whereis(couch_stats_process_tracker), monitors),
    Candidates = [Pid || {process, Pid} <- M],
    lists:foldl(
        fun(Pid, Acc) ->
            case process_info(Pid, [message_queue_len, dictionary]) of
                undefined -> PI = [];
                PI -> ok
            end,
            case proplists:get_value('$initial_call', proplists:get_value(dictionary, PI, [])) of
                {Mod, init, 1} ->
                    case proplists:get_value(message_queue_len, PI) of
                        Len when Len > N -> [Pid|Acc];
                        _ -> Acc
                    end;
                _  ->
                    Acc
            end
        end,
        [],
        Candidates
    )
end.
```

Next, get a list of PIDs.

```
> Pids = FindTiredPids(couch_db_updater, 1000).
[<0.31017.4673>,<0.28837.3289>]
```

This command returns a list of PIDs with long > *1000* message queues.

How to remediate the failure.

It is worthwhile to be patient in remediating this type of failure because it is possible for the processes to recover on their own. If you observe a process with a message queue that is spiraling out of control (for example, the mailbox has more than 100,000 messages), stop the process immediately. If not, wait a few minutes and see whether the system recovers.

If the process does not recover on its own, run this `remsh` command to stop all backed-up CouchDB updaters.

```
> [exit(Pid, kill) || Pid <- global_changes_serverPids].
[true,true]
```

How to verify the remediation.

Confirm that the list of backed-up CouchDB updaters is empty.

```
> FindTiredPids(couch_db_updater, 1000).
[]
```

The `couch_file` message queue check

What it checks.

This check monitors the message queues of the various `couch_file` processes. These processes manage access to the underlying file system; one `couch_file` process exists for each open database shard and one for each open view shard.

Under heavy load, a `couch_file` process might be unable to process messages as fast as it receives them. When this problem occurs, you must stop the process because the process might not recover.

How to confirm the failure.

SSH to the node where the failure occurred and create the following function in a `remsh`.

```
FindTiredPids = fun(Mod, N) ->
    {monitors, M} = process_info(whereis(couch_stats_process_tracker), monitors),
    Candidates = [Pid || {process, Pid} <- M],
    lists:foldl(
        fun(Pid, Acc) ->
            case process_info(Pid, [message_queue_len, dictionary]) of
                undefined -> PI = [];
                PI -> ok
            end,
            case proplists:get_value('$initial_call', proplists:get_value(dictionary, PI, [])) of
                {Mod, init, 1} ->
                    case proplists:get_value(message_queue_len, PI) of
                        Len when Len > N -> [Pid|Acc];
                        _ -> Acc
                    end;
                _  ->
                    Acc
            end
        end,
        [],
        Candidates
    )
end.
```

Next, get a list of PIDs with long > *1000* message queues).

```
> Pids = FindTiredPids(couch_file, 1000).
[<0.31017.4673>,<0.28837.3289>]
```

How to remediate the failure.

In `remsh`, run this command to stop all processes in the list.

```
> [exit(Pid, kill) || Pid <- Pids].
[true,true]
```

How to verify the remediation.

Use the same `remsh` function that you used to identify the tired processes. This command returns an empty list.

```
> FindTiredPids(couch_file, 1000).
[]
```

When to escalate the page.

If you are paged multiple times for a cluster, that is, the issue is recurring, escalate the page.

The `couch_server` message queue check

What it checks.

This function checks whether the `couch_server` message queues are growing larger, which indicates that they are backing up.

What it means when it fails.

The `couch_server` is on the critical path for many RPC calls. The overall effect of a backed-up `couch_server` is dramatically increased latency on a subset of requests. For example, requests that are on a critical path.

How to fix it.

First, use the metrics application and check whether the incident is ongoing or a spike. If it is a spike that is subsiding, no further action is required. If it is not decreasing, restart `couch_server` with this command.

```
exit(whereis(couch_server), kill). src
```

The `ddoc_cache_opener` message queue check

What it checks.

The `ddoc_cache_opener` message queue is backing up. Use this command from a `remsh` to monitor the queue directly.

```
process_info(whereis(ddoc_cache_opener), message_queue_len).
```

What it means when it fails.

If it continues to back up, the ability of the server to handle HTTP requests might be effected.

How to fix it.

If the `message_queue` size is not recovering on its own, restart the `ddoc_cache_opener` process, as follows.

```
exit(whereis(ddoc_cache_opener), kill).
```

If this approach does not resolve the problem, contact support.

The `global_changes_server` message queue check

What it checks.

This command check monitors the number of messages in the `global_changes_server` message queue.

How to fix it.

Check the logs for entries that mention `erlang.message_queues.global_changes_server`. If the total does not drop to zero, an ongoing problem exists. If the number of messages is decreasing, but peaking often, more action might be needed.

It is possible to reduce the pressure on the `global_changes` server by increasing the `global_changes/max_event_delay` and `global_changes/max_write_delay` parameters. The default value is `500ms`, although values as high as `10000ms` might be useful.

In a `remsh`, run the following to increase the intervals (in this case to `5000ms`).

```
> rpc:multicall(config, set, [global_changes, max_event_delay, 5000,
                              Alleviate global_changes_server message queue pain]).
> rpc:multicall(config, set, [global_changes, max_write_delay, 5000,
                            Alleviate global_changes_server message queue pain]).
```

It is also possible that the `global_changes_server` message queue is a side effect of another issue. Watch for the values in the logs to return to zero. If the message queue size is not decreasing, contact support.

The `mem3_shards` message queue check

What it checks.

The length of the `mem3_shards` message queue. The `mem3_shards` process acts as a cache of the `/dbs` database.

What it means when it fails.

As it is a cache, the node probably still functions correctly, but slower, as it reads from the disk more often.

How to fix it.

You can watch the increase on the node by using the following process.

```
process_info(whereis(mem3_shards), message_queue_len)
```

This process shows a speedy increase in the message queue length, such as 1000 per second.

First, put the node into `maintenance_mode` with this command.

```
config:set("cloudant", "maintenance_mode", "true")
```

Next, see whether the message queue length decreases. If it does decrease, wait for it to go to zero and bring the node back in with this command.

```
config:set("cloudant", "maintenance_mode", "false")
```

Now, check whether the queue starts increasing or stays at zero. If it starts to increase again, the final fix is to stop the process.

```
exit(whereis(mem3_shards), kill).
```

If it is still increasing, call support.

The `rexi_server` message queue check

What it checks.

This check monitors the number of messages in the various `rexi_server` message queues. Depending on cluster configuration, this process might be a single Erlang process or a set of processes, one for each node in the cluster.

These processes manage all inter-node communication. If inter-node communication is not processing their mail boxes correctly, that communication is delayed.

How to confirm the failure.

SSH to the node where the failure occurred and create the following function in a `remsh`.

```
ShowRexiServerMailboxes = fun() ->
    Names = case config:get("rexi", "server_per_node") of
        "true" ->
            [element(1, C) || C <- supervisor:which_children(rexi_server_sup)];
        _ ->
            [rexi_server]
    end,
    Mailboxes = lists:map(
        fun(Name) ->
            Pid = whereis(Name),
            {message_queue_len, Len} = process_info(Pid, message_queue_len),
            {Name, Len}
        end,
        Names
    ),
    lists:sort(fun(A, B) -> element(2, A) =< element(2, B) end, Mailboxes)
end,
```

Calling the function gives you a list of mailboxes and their sizes.

```
> ShowRexiServerMailboxes().
[{rexi_server,0}]
```

This function returns a list of {*registered_process_name, mailbox_size*} tuples; the sum of the message queue length values exceeds the alert threshold.

How to fix it.

As indicated previously, you can configure `rexi` communication patterns in two ways. You probably need to switch over to the newer *server-per-node* configuration.

In a `remsh` on the relevant node, verify that the server-per-node configuration is disabled.

```
> config:get("rexi", "server_per_node", "false").
```

This function returns the string “false”. If the function returns “true”, contact support.

If the server-per-node configuration is disabled, enable it in a `remsh`.

```
> F = fun() -> supervisor:restart_child(rexi_sup, rexi_server_mon), \
    supervisor:restart_child(rexi_sup, rexi_server_sup) end.
> rpc:multicall(erlang, apply, [F, []]).
> timer:sleep(60000).
> rpc:multicall(config, set, ["rexi", "server_per_node", "true"]).
```

Calling `restart_child` might return `{error,running}`, or something similar, which is expected. They are restarted as a fail-safe measure, and attempting to do so multiple times does not have a negative impact.

How to verify that it is fixed.

Run `ShowRexiServerMailboxes().` again and see whether the size is decreasing. If the cluster is already using the server-per-node configuration, but the problem persists, contact support.

The `custodian` check

What it checks.

This function checks the number of shard replicas that are currently reachable for all shard ranges for any database is equal to the default N, usually 3.

How it is checked.

A program that is called Custodian runs on every database node in the cluster and inspects the `dbs` database. Documents in the `dbs` database are referred to as shard maps or partition tables, and contain a mapping of database shards to nodes.

Custodian raises an alarm if one of the following situations occur.

- Some shards have less than or more than the appropriate number of replicas that are listed in the shard table.
- The node to which a shard is mapped is unavailable.

What it means when it fails.

If `n < N`, it means that a database has shards that are under protected, with fewer than N replicas available in one or more shards ranges in one or more databases. If `n > N`, it means that some shard ranges in some databases have too many replicas. This issue is not as serious as having too few, but must be addressed.

Often, this warning is a proxy for a disabled database node (such as a `dbcore` restart or hardware failure), as this disabled database node renders a good number of shard replicas unreachable. If it is not caused by a disabled node, it might be caused by one of the following issues.

- Someone deliberately created a database with less than N replicas.
- A shard moves failed, a shard disappeared, or an extra copy was created, or the shard map was not properly updated.
- Some other shard-map issue.
- One or more nodes are not connected to all cluster nodes.
- One or more shards have an N value greater than the default N.

How to fix it.

Make sure that the problem is not caused by a disabled server, unreachable, or not connected to all other nodes. If that is not the case, contact support.

The `disk` check

What it checks.

This function checks whether the `/` and `/srv` file systems are writable.

How it is checked.

This check is done by writing a temporary file to `/tmp` and `/srv/sensu` and making sure that it succeeds. A failure of any kind or a 15-second timeout causes the check to fail.

What it means when it breaks.

The file system has most likely become read-only. A read-only file system can cause `dbcore` to run in a degraded mode that affects the rest of the cluster (if it is the `/srv` file system that is read-only). If the `/` file system is read-only, a number of other problems can occur. At the least, chef does not run.

If a disk error occurs, this disk error can cause a file system to be remounted as read-only.

If the disk is mounted read/write and enough space is available, the file system used the pool of free `inodes`.

How to fix it.

To check the mount status, use mount. In the following example, `rw` indicates mounted read/write, `ro` read-only.

```
$ mount
            /dev/sda1 on / type ext4 (rw)
            ...
            /dev/sdc1 on /srv type ext4 (rw,noatime)
```

If `/srv` is read-only, stop `dbcore`: `sudo sv stop cloudant` to keep it from affecting the rest of the cluster. You also might investigate what is causing the check to fail.

- Was it remounted as read-only (possibly due to a disk error)? Run **`mount`** and check for `ro`.
- Is there another reason the file cannot be written? Check for the existence of `/srv/` and that it is writable.

- Check the load on the server (by using uptime or top) to see whether a high load is causing file system operations to time out.

You must restart the node. When it starts, it might work or you might get disk errors. If you get disk errors, you probably need to replace the drive. Then, do a standard node replacement. If the disks are mounted correctly, run `df -i` and look at the `IFree` column to check the available `inodes`. If the number of `inodes` is less than 1000, remedy the immediate situation by disposing of unneeded and orphaned files. Additionally, the file system might be configured incorrectly, in which case `sudo tune2fs -m 0 <partition>` can help. However, neither of these options is a permanent solution.

The `ioq` check

What it checks.

This check monitors the number of requests that are waiting to be processed by IOQ.

What it means when it fails.

The IOQ has numerous pending requests, but it is not necessarily stuck. To check, look at the volume of IOQ requests being processed in the metrics application.

How to fix it.

Verify that the situation is not caused by a sudden "spike" in activity. Look for a relatively slow growth in pending requests. If you see that, do the following steps.

Verify that the IOQ is not saturated with traffic. If the number of IOQ reads and writes shown in the metrics application is more than 20k, IOQ is probably saturated with traffic. A quick fix to this problem is to bypass interactive IOQ traffic.

```
> config:set("ioq.bypass", "interactive", "true").
```

After this bypass is enabled, the pending request count trends downward.

If the IOQ is not saturated, something else is happening. The best immediate action is to put the node in maintenance mode.

```
> config:set("cloudant", "maintenance_mode", "true").
```

Next, wait 30 approximately seconds, then restart `dbcore`.

```
$ sudo sv restart dbcore
```

If you get to this point, contact support to investigate why IOQ failed.

The `search` check

What it checks.

This functions checks whether `clouseau` is running on the node. Clouseau acts a wrapper around the Lucene library that does the generation, updating and querying of search indexes at the shard level. If `clouseau` is not running, the node cannot serve search requests.

How to fix it.

Try disconnecting `clouseau`; it automatically reconnects.

```
disconnect_node('clouseau@127.0.0.1').
```

Next, run Weatherreport search to see whether the problem is fixed. You might need to repeat this cycle a few times. If it is still not working, try this solution instead.

```
sudo sv restart clouseau
sudo sv restart clouseau
weatherreport search
```

Replacing a node

Use these instructions to replace a Cloudant database or load balancer node.

Select the instructions to replace a database or load balancer node.

- Replace a database node
- Replace a load balancer node

Replace a database node

1. Run the following `safe_to_rebuild` command on the node to determine whether you can decommission the node.

   /opt/cloudant/bin/weatherreport safe_to_rebuild

   - If you can rebuild or replace the node, no diagnostic messages are displayed in response to this command.
   - If it is not safe, one or more shard ranges are reduced to one or zero live copies. In that case, an error message (for one live copy) or a critical message (for zero live copies) is displayed with the shard ranges affected.

     If any messages are returned, make sure that the shards have enough copies in live nodes before you replace the node.

   Note: Weatherreport is a command-line application that provides information about the status of a `cloudant` node or cluster. It is useful in troubleshooting cluster issues, such as increased latencies, low disk space, or node failures. For more information about Weatherreport, see "Checking the health of a cluster with Weatherreport" on page 51.
2. If the node is running, stop any services that are running on the node and shut it down.
3. Repair or replace the node.
4. Install Cloudant Local on the database node, as described in "Installing additional database nodes" on page 14.
5. Run `cast system install` and pass the `--maintenance` flag as well as the original `cluster_dbnode.yaml` file used to setup the node initially.

   ```
   cast system install --maintenance -db -c cluster_dbnode.yaml
   ```

   - (Optional) If you do not have the original `cluster_dbnode.yaml` file, you can generate a new one by running the `cast export cluster` command from an existing node.

     ```
     cast cluster export cluster_dbnode.yaml
     ```
6. Confirm that the node is in maintenance mode by using the following command.

   ```
   cast node maintenance
   ```

   If the node is in maintenance mode, the response shows the status of the nodes.

   ```
   The node is IN maintenance mode.
   ```
7. In maintenance mode, the node starts the process to bring itself back into a healthy state and ready for traffic. To monitor the process, check the log files for the node to determine the number of `pending_changes` in the internal replication and indexing processes as shown in the following example.

   ```
   <0.3333.0> - mem3_sync shards/00000000-3fffffff/dbname.1407441053 cloudant@db1.clu
   ster001.cloudant.net {pending_changes,62}
   ```

   Check the number of pending internal replication tasks by running the following command in a terminal on one of the database nodes.

   /opt/cloudant/bin/weatherreport -a -d info internal_replication

   If the internal replication is complete, the output from this command indicates that the number of pending internal replication jobs is 0 or close to 0 for the database node. Additionally, the active tasks

section of the metrics application shows the number of indexing processes. After the node reaches the point where only a few indexing tasks are running, take the node out of maintenance mode and put it in production mode.

Replace a load balancer node

1. Stop traffic to the load balancer node before you remove it. How you do that varies based on your setup.

   If you are running a failover load balancer, confirm that the failover load balancer is operating correctly, and you can access the cluster with that load balancer.

   Next, make any required changes, such as changes to DNS settings, to reroute all traffic through the failover load balancer.
2. If the node is running, stop any services that are running on the node and shut it down.
3. Repair or replace the node.
4. Install Cloudant Local on the load balancer node, as described in Chapter 5, "Installing Cloudant Local," on page 13.
5. Configure the `lbnode.yaml` file with your current cluster configuration when you run the `cast system install` commands.
6. Verify that the replacement load balancer is operational. You can do that by directly accessing the load balancer and confirming that you get a valid response.

   Next, start sending traffic to the load balancer again.

   Note: Configuration changes and restarts to the load balancer can affect traffic. How these changes are made depends on your setup. If you are running a failover load balancer, confirm that it is running correctly and you can access the cluster through it. Then, make any required changes, such as DNS settings, to reroute all traffic through the failover or replacement load balancer.

Adding a node to a cluster

Use these instructions to add a node to a cluster and rebalance the shards.

Step 1. Provision the new node

Use the following steps to configure the node by using the modified configuration process.

1. Install Cloudant Local on the database node that you are adding, as described in "Installing additional database nodes" on page 14.
2. Run the `cast system install` command by passing the cluster configuration from your current `dbnode.yaml` file.
3. Put the node in maintenance mode after installation.

   ```
   cast node maintenance --true
   ```

Step 2. Confirm that the node is in maintenance mode

The node must be in maintenance mode so that it does not cause any errors when it is brought into the cluster.

1. Confirm that the node is in maintenance mode by running the following command.

   ```
   cast node maintenance
   ```

   The response shows the status of the node.

   ```
   The node is IN maintenance mode.
   ```

Step 3. Add the new node to the nodes database on one of the existing nodes

1. SSH into one of the existing database nodes and add the new node to its `nodes` database.

   ```
   cast cluster add {new_node_hostname}
   ```

2. Verify that the node was successfully added to the `nodes` database.

   ```
   cast cluster status
   ```

Step 4. Update the load balancer configuration

The load balancers must know to direct traffic to the new node.

1. For each load balancer, add the new node to the list of nodes in the `lbnode.yaml` file.
2. Restart `haproxy` after you save the changes.
3. Verify that your load balancer can correctly access the new node.

   Note: Configuration changes and restarts to the load balancer can affect traffic. How these changes are made depends on your setup. If you are running a failover load balancer, confirm that it is running correctly and you can access the cluster through it. Then, make the required changes, such as DNS settings, to reroute all traffic through the failover load balancer.

4. Edit the `lbnode.yaml` file that was created during initial cluster installation to add the new database node.
5. Run the `cast node config` command as seen in the example below.

   ```
   cast node config /opt/cloudant/cast/samples/lbnode.yaml
   ```

Step 5. Create a rebalancing plan

You must rebalance the cluster by create a rebalancing plan.

1. Run the rebalancing process from any of your load balancers if you are using more than one load balancer.
2. If you do not have a configuration file for the rebalancer, create a file that is named `rebal.ini` and save it in the home directory for the user on the load balancer. See the format of the configuration file. Ensure that you remove the line breaks from the comment lines that start with the pound sign (#).

   ```
   [cloudant]
   # System user which runs the db process
   db_user = cloudant
   # Server-wide database admin credentials. These have to be identical for every node of the cluster.
   rebal_user =
   rebal_password =
   # Default filename for rebalance plans.
   rebal_plan = rebalance_plan
   # Mapping between public and private hostnames for nodes. This is only needed if
   # the hostnames have a private and a public interface and the load balancer does
   # not have access to the private interface. If the private hostname can be used,
   # these fields can be left blank. Otherwise, the private hostname extension
   # is replaced with the public hostname extension to arrive at the public hostname.
   # If you are unsure, try leaving these empty.

   private_host_ext =
   public_host_ext =
   [rebal]
   # Set to true to disable prompts on failure and just continue
   batch_mode = false
   ```

3. Confirm that the admin credentials for *rebal_user* and *rebal_password* in the configuration file are correct by logging in to one of the database nodes with SSH, and then run the curl command.

   ```
   curl http://{admin:password}@localhost:5986/_active_tasks
   ```

4. Replace *admin:password* in the preceding command with the credentials for the database administrator. If you specify the wrong credentials, you receive an error message.

Setting up remote access for the rebalancer

The rebalancer runs on the load balancer and needs SSH access to the database nodes to run the operations that are required to move shards between database nodes. The Cloudant Local installer creates a `cloudantrebal` user account on every database node in your cluster for the rebalancer. During installation, this account is configured with the permissions and path that are required to do the operations that the rebalancer needs to do with SSH. The `cloudantrebal` account eliminates some of the work an operator must do to set up a cluster rebalance.

Follow these instructions to enable SSH access for the rebalancer.

1. On the load balancer, generate a pair of public and private keys for SSH access to the rebalancer. This step can be done as the root user. See the following example.

   ```
   > cd ~/.ssh
   > ssh-keygen -t rsa
   ```

   Enter a name for the key file, or leave it as the default. Record the passphrase that you enter for the key creation. The result is to produce two key files in the `~/.ssh` folder.

   - A private key file with the default name `id_rsa`.
   - A public key file with the default name `id_rsa.pub`.

2. Add the public and private keys generated in step 1 to every database node in your cluster.

   The controlling node is defined as the node where you run **`rebal`** (in this case your load balancer). The new node is the database node that you just added to the cluster. You must set up the keys since **`rebal`** uses SSH to communicate between the controlling node and the database nodes (existing and new), as well as between nodes when `rsync` shards from existing database nodes to the new node.

   a. Create an authorized_keys file in `/opt/cloudantrebal/.ssh`.

   b. Add the public key to the `/opt/cloudantrebal/.ssh/authorized_keys` file on all the database nodes, including the one you are adding.

   c. Copy the public and private keys to `/opt/cloudantrebal/.ssh`.

   d. Ensure that the `owner:group` is `cloudantrebal:Cloudant`.

      The private key has permission, `0600`, and the public key has permission, `0644`. See the following example.

      ```
      cloudantrebal@db1:~/.ssh$ ls -la
      total 20
      drwxr-xr-x 2 cloudantrebal cloudant 4096 Mar 27 23:57 .
      drwxr-xr-x 3 cloudantrebal cloudant 4096 Mar 27 23:57 ..
      -rw-r--r-- 1 cloudantrebal cloudant  395 Mar 27 23:57 authorized_keys
      -rw------- 1 cloudantrebal cloudant 1675 Mar 27 23:57 id_rsa
      -rw-r--r-- 1 cloudantrebal cloudant  395 Mar 27 23:57 id_rsa.pub
      ```

   e. Test that you can SSH from your existing database nodes to the new node by using the following command.

      ```
      cloudantrebal@db1:~/.ssh$ ssh new-db-node.example.com
      ```

3. On the load balancer, create the following entry in the `~/.ssh/config` file for the account that you are using to run the rebalancer.

   ```
   Host *.yourcluster.yourdomain.com
   User cloudantrebal
   ```

4. If you use `ssh-agent` to manage the SSH credentials for the rebalancer, ensure that the private key used in step 1 is specified by providing the filename of the private key to `ssh-add`. See the following example.

   ```
   > eval $(ssh-agent)
   > ssh-add ~/.ssh/id_rsa
   ```

Enter the passphrase of the key when prompted. These steps save the passphrase so that you are not prompted for it again in the current **ssh** session. You run the commands again if you open a new **ssh** session on the load balancer, for example, when you run the rebalance shards scripts in subsequent steps.

5. Confirm that the database nodes are accessible with SSH from the load balancer, such as by configuring `ssh-agent`. You can verify the access by trying to SSH into all database nodes from the load balancer.

Create a rebalancing plan

After the rebalancer is configured and the `ssh-agent` is set up, you can start the rebalancing process by creating a rebalancing plan.

1. Create a working directory to hold the plan and log files and change to that directory by specifying these commands.

   ```
   mkdir rebaldir
   cd rebaldir
   ```

2. Create the plan by running `rebal plan expand` with one of the database nodes as an argument, as shown in the following example.

   ```
   rebal plan expand 'db1.cluster001.example.com'
   ```

3. Check the output from this command for any error messages similar to this one.

   ```
   Retrying SSH connection
   ```

4. If you receive a similar error message, check the `ssh-agent` configurations and confirm that you can SSH to all database nodes, without specifying a password.

   Your current directory now contains a `rebalance_plan` file that includes a list of shard moves.

Step 6. Put the new node into production mode

You must take the new node out of maintenance mode so it can serve traffic for the moved shards, while the rebalancing is in progress.

1. Run the `cast node` command to take the node out of maintenance mode.

   ```
   cast node maintenance --false
   ```

2. Run the following command on the new database node to verify the status of that node.

   ```
   cast node maintenance
   ```

   If the node is functioning normally in production mode, you receive this response.

   ```
   The node is OUT of maintenance mode.
   ```

Step 7. Run rebalancing

Before you continue, confirm that no other shard moves are running on the same cluster. Moving multiple shards of the same database concurrently can lead to data loss.

1. Run your rebalancing plan, and replace *rebalance_plan_filename* with the filename for your rebalance plan.

   Note: The filename was specified earlier when you edited the `rebal.ini` file while creating a rebalancing plan. The command runs the shard moves identified in the plan. Shards moves are batched by database name with each batch run in a series. Multiple batches are run concurrently, with batches of 20 as the default. Progress and errors are logged to the `rebalance_plan.log`.

   ```
   rebal run rebalance_plan_filename
   ```

2. Follow the progress by using the **tail** command.

   ```
   tail -f rebalance_plan.log
   ```

   As the command progresses, 2 log files are created for each database, `dbname.out`, and `dbname.err`. The files store standard output and error messages.

What to do if an error occurs

If an error occurs, do the following steps.

- Check the log files for the presence of **sudo** or **tty** errors. If they are present, do the following tasks for the `/etc/sudoers` file on all the database nodes.
  1. Search for a line that contains.

     ```
     Defaults requiretty
     ```

     Modify the line to read.

     ```
     Defaults !requiretty
     ```
  2. Add the following lines to the file.

     ```
     cloudantrebal        ALL=(ALL) NOPASSWD: ALL
     %cloudant            ALL=(ALL) NOPASSWD: ALL
     ```
- Check whether it is a problem with the configuration of `rebal` or `ssh-agent`. Confirm that `ssh-agent` is set up correctly and that the `.rebal` file contains the right credentials.
- Check whether multiple `erl_call: failed to connect to node` messages in the `dbname.out` files exist. If multiple messages exist, you might need to reduce the concurrency of the rebalancing process.

  ```
  rebal run --concurrency 5 rebalance_plan_filename
  ```

  This command reduces the number of concurrent shard moves to 5 from the default of 20.
- It might be a temporary issue, such as an unreachable or overloaded node, leading to a timeout. Run the rebalancing process again with the **rebal run** command and the appropriate rebalance_plan_ filename. Rebalancing in this way does not cause a shard to have too few copies, but it might result in shards with more than the expected number of copies.
- If you see `rsync` errors in the log file, verify that `rsync` is installed and operational on all database nodes. Otherwise, install `rsync` package on all the nodes. For example, on RedHat platforms, use the `yum install rsync` command to install `rsync`.

If you checked the configuration and tried to rerun the shard moves, but the problem persists, contact support.

Step 8. Verify the new node

Confirm that the new node is set up correctly.

1. Log in to the new node with SSH and run Weatherreport.

   ```
   /opt/cloudant/bin/weatherreport
   ```
2. If errors are reported during the run, see the information for Weatherreport checks, or call support.
3. After the run finishes, check the rebalancing logs that are described in the Run rebalancing step for any errors.
4. Verify whether the shards for different databases are now distributed across the recently added node. The following command checks the shard map for a database.

   ```
   curl -X GET http://localhost:5986/dbs/default%2Fdatabase_name | python -m json.tool | less
   ```

   Look at the `by_node` key in the output from this command to determine whether the shards are spread evenly across the cluster nodes.

   Note: Weatherreport is a command-line application that provides information about the status of a `dbcore` node or cluster. Weatherreport is useful in troubleshooting cluster issues, such as increased latencies, low disk space, or node failures. For more information about Weatherreport, see "Checking the health of a cluster with Weatherreport" on page 51.

Upgrading your Cloudant Local installation

To upgrade Cloudant Local, you take each database and load balancer node offline one at a time and run the upgrade. After the upgrade, you bring the node back online and repeat the process on the next node in the cluster. This way Cloudant Local stays online during the upgrade.

Overview for upgrading a database node and a load balancer node

Follow these steps to upgrade from a Cloudant Local version 1.0.0.3 installation to Cloudant Local version 1.0.0.5.

- Upgrade the Cloudant Local binary code.
- Upgrade the internal data storage format that is used by Cloudant Local including changes that affect format or directory usage.

You are not always required to upgrade the data storage format since it depends on the capabilities you use. When a format upgrade is required, the upgrade is an automatic, one-way process.

Important: After the upgrade is complete, you cannot roll back to an older version.

You can upgrade only one node at a time, which can eliminate downtime. Upgrading a node is similar to replacing a node to a cluster.

Follow the basic upgrade steps for a database node.

1. Put the database node that you want to upgrade in maintenance mode.
2. Uninstall the existing Cloudant Local version binaries.
3. Install the new Cloudant Local packages.
4. Start the upgraded node in maintenance mode.
5. Check the status of the node.
6. Bring the node out of maintenance mode.

Follow the basic upgrade steps for a load balancer node.

1. Confirm that load balancer failover operates correctly if one load balancer node is being upgraded.
2. Uninstall the existing Cloudant Local version binaries.
3. Install the new Cloudant Local packages and start the upgraded node.
4. Check the status of the node and confirm that traffic is being serviced through the upgraded node.

Upgrading a database node

Follow these steps to upgrade each Cloudant Local database node.

1. Put the database node that you want to upgrade in maintenance mode.
   a. Run this command from your current installation and put the database node in maintenance mode, **/opt/cloudant/sbin/cloudant.sh -m**.
   b. Verify that the database node is in maintenance mode, **curl localhost:5984/_up**. You see this response. `{"status":"maintenance_mode"}`
   c. Verify that this node is not available for traffic from your load balancer URL, `http://load_balancer/_haproxy`. The node status must be stopped for maintenance.
2. Uninstall the existing Cloudant Local version binaries.
   a. Uninstall the current version by using the **uninstall** command, **cd /root/Cloudant/ ./uninstall.sh -q -r**. This command also stops the services.
   b. Check for Cloudant processes, **ps -ef | grep clo**.
   c. Stop any Cloudant processes that are running.

```
kill -9 (pid)
```

d. Rename the following directories to proceed with the new version. You might need to refer to these files in the future. See the following examples that are renamed with an `_old` extension.

```
mv /opt/cloudant/ /opt/cloudant_old
mv /var/log/cloudant /var/log/cloudant_old
mv /etc/sv/cloudant /etc/sv/cloudant_old
mv /etc/sv/clouseau /etc/sv/clouseau_old
mv /etc/sv/cloudant-local-metrics /etc/sv/cloudant-local-metrics_old
```

3. Install the database node by using the steps in "Installing the first database node" on page 14.

   Note: Ensure that when you run the `cast system install` command you start the node with a maintenance mode flag and pass the current cluster configuration to your `dbnode.yaml` file.

   a. Find the current values of database node cluster credentials and IDs from the `local.ini`, `vm.args`, and `default.ini` files on other nodes or from the backed-up directories, such as `/opt/cloudant_old/etc`.

   b. Configure these values as-is in the `dbnode.yaml` file. Use encrypted or hashed values for all fields except for metrics fields that are configured with plain-text credentials.

   c. After you configure the `dbnode.yaml` file, you can distribute and use it for other database node upgrades. For example, if you configure the values in `/root/cloudant-installer/dbnode.yaml`, you can run the following commands.

```
cd /root/cloudant-installer
cast system install --maintenance -db -c /root/cloudant-installer/dbnode.yaml
```

4. Check status of the database node update.

   a. Verify the status of the database node.

```
curl -X GET http://localhost:5984
```

   The response shows the new version.

```
{"couchdb":"Welcome","version":"1.0.0.5-local","vendor":{"name":"Cloudant,
an IBM Company","version":"1.0.0.5","variant":"local"},"features":["geo"]}
```

   b. Verify that the database node is in maintenance mode and is not receiving traffic on the load balancer.

```
cast node maintenance
```

   The response shows the status of the node.

```
The node is IN maintenance mode.
```

   c. Verify the status of the upgraded node as a member of the cluster.

```
cast cluster status
```

   The response lists all the cluster nodes.

   d. Check the health of the database node by using Weatherreport, "Checking the health of a cluster with Weatherreport" on page 51.

5. Bring the database node out of maintenance mode and check its availability.

   a. Run the following command to bring the database node out of maintenance mode.

```
cast node maintenance --false
```

   The response shows the status of the node.

```
The node is OUT of maintenance mode.
```

   b. Verify that the database node is up and available for traffic from your load balancer.

```
http://{load_balancer}/_haproxy
```

   If the upgrade was successful, the response shows that the node is up and available.

Upgrading a load balancer node

Follow these steps to upgrade each Cloudant Local load balancer node.

1. Verify that the load balancer failover works correctly when one of two or more load balancers are taken offline during an upgrade.
2. Uninstall the current version of Cloudant Local.

   ```
   cd /root/Cloudant/
   ./uninstall.sh -q -r
   ```

3. Install the new Cloudant Local packages and start the upgraded node, "Installing load balancer nodes" on page 15.
   a. Find the load balancer node's current credentials, cluster nodes, hostname, and IP addresses in the current version of the `/etc/haproxy/haproxy.cfg` file.
   b. Configure the values in the `lbnode.yaml` file. If you configured the values in the `/root/cloudant-installer/lbnode.yaml` file, run the **`cast system install`** command that uses this format.

      ```
      cast system install -lb -c /root/cloudant-installer/lbnode.yaml
      ```

   c. Pass the load balancer configuration information from your current `lbnode.yaml` file when you run **`cast system install`**.
4. Verify that load balancer is reachable through its URL and that all cluster nodes are listed and available to receive traffic. Test the load balancer with this URL, `http://{upgrade node load balancer}/_haproxy`.

Configuring Failover Load Balancers

Configure load balancing to eliminate downtime if a load balancer fails to respond.

A load balancer node assists with distributing Cloudant Local requests to enable rapid response. It is possible for a load balancer node to become unavailable, for example, as part of a system administration activity. This task describes how to configure extra or replacement node balancers for a Cloudant Local system.

Assume that each load balancer has its own unique and public IP address. A single, separate IP address is used as the actual front-end address where all requests are directed from outside the Cloudant Local system.

Table 2. Example Load Balancer configuration

IP Address	System
10.10.50.5	Front-end system
10.10.50.6	Load Balancer 0
10.10.50.7	Load Balancer 1

1. Perform the load balancer installation task for each of the load balancer systems (Load Balancer 0 and Load Balancer 1). See Chapter 5, "Installing Cloudant Local," on page 13.
2. On each load balancer node, as the root user, modify the `haproxy.cfg` file to identify the front-end address and change the following line from

   ```
   listen dbfarm 0.0.0.0:80
   ```

   to

   ```
   listen dbfarm 10.10.50.5:80
   ```

   where the new IP address is that of the front-end system.
3. Install the **`keepalived`** application on each load balancer node as the root user. For CentOS or RHEL, use the command.

```
yum install keepalived
```

For Debian or Ubuntu, use the command.

```
apt-get install keepalived
```

4. Create a `/etc/keepalived/keepalived.conf` file on each load balancer based on the following lines.

```
! Configuration File for keepalived

global_defs {
   notification_email {
     failover@domain.com
   }
   notification_email_from lb@domain.com
   smtp_server localhost
   smtp_connect_timeout 30
}

vrrp_instance VI_1 {
    state MASTER
    interface eth1
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
 10.10.50.5
    }
}
```

Ensure that each load balancer has a unique `priority` value. For example, Load Balancer 0 might be set to `priority 100` while Load Balancer 1 might be set to `priority 101`.

5. Set the `virtual_ipaddress` IP to the IP address for the front-end server.
6. Configure **`keepalived`** so that it runs automatically when the server starts. For CentOS or RHEL, use the commands.

```
chkconfig keepalived on
service keepalived start
```

For Debian or Ubuntu, use the command.

```
service keepalived start
```

7. Check that the front-end IP address can be accessed successfully by going to the IP address in a browser.
8. Check that load balancing is working correctly. Check that the access is successful.
   a. Disconnect the highest priority load balancer.
   b. Revisit the front-end IP address.
9. Reconnect the highest priority load balancer. Disconnect the lower priority load balancer. Again, revisit the front-end IP address. Check that the access is successful.

Resolving disk space issues

Use these instructions to identify and resolve disk space issues.

Check Metrics

First, check metrics to get background information on the cluster, such as the current request load and the status of compaction tasks.

Assess the space on the disk.

Log in to the remote server.

```
ssh
db1
```

Use **du -h -s /srv/*** to get a basic outline of how disk space is distributed. The output tells you whether database or view data is taking up the most space.

For more detailed information, use `durep` to monitor disk usage.

```
durep -x -hs 5000M /srv | less
```

In this example, the following command is returned.

```
50.5G [################              ]  53.56% prod_features.1342146305.couch
```

The output shows that the `prod_features` shards are large.

Compacting specific, oversized shards

Sometimes, specific shards seem much bigger than needed. If so, you can compact just that shard.

```
durep -x -hs 5000M /srv | less
```

This command puts the biggest shards at the top of the list. If the shard is large, such as 80%+ of the space, you can manually compact it.

```
$ pwd
/srv/db/shards/f5555546-ffffffff/foo
$ ls -lah bar.1332615163.couch
-rw-r--r-- 1 dbcore dbcore 944G Sep 30 09:01 bar.1332615163.couch
```

In `remsh`, you see the following information.

```
{ok, Db} = couch_db:open(<<"shards/f5555546-ffffffff/foo/bar.1332615163">>,
    [{user_ctx, {user_ctx, null, [<<"_admin">>], undefined}}]).
couch_db:start_compact(Db).
```

You can watch progress by using `_active_tasks` for the cluster and this `jq` pipeline.

```
jq '.[] | select(contains({node: "db1", type: "database_compaction",
  database: "shards/f5555546-ffffffff"}))'
```

If compaction has failed before, `compact.data` and `compact.meta` files are saved to `/srv/db/shards/`, alongside the shard. You can see that the `.compact` files are orphaned.

- by the date
- by doing **sudo lsof dbname.1332615163.couch.compact.data**

This command returns nothing if a compaction process that uses the file no longer exists (that is, no process has an active file-handle to that file).

Compaction

Smoosh is the auto-compactor for the databases. If Smoosh is having issues, it can result in compactions not happening or compacting inappropriate shards (for example, shards that do not reclaim much space). For more information, see the guide for compaction.

Overview of disks and file system layout on a DBCore node

Read this information to learn about disks and file system layout on a DBCore node.

Layout of DBCore files

Cloudant Local stores files on a single partition. The partition includes the following files.

- CouchDB files for each shard of a database
- View database files
- Lucene search index files
- Geo index files

The following example shows how a database node might look.

Database files

```
/srv/cloudant/db/dbs.couch        <---  node-local couch DB file of all dbs
/srv/cloudant/db/nodes.couch      <---  node-local couch DB file of all nodes in the cluster
/srv/cloudant/db/cluster001/   <--- cluster-based user
  users.couch               <--- node-local couch DB file of users on this cluster
/srv/cloudant/db/shards/          <---- shards directory
  00000000-0aaaaaa9/         <--- one dir per shard range
    cluster001/              <--- all dbs for this user
      stats.1348021480.couch     <--- shard of cluster wide stats db
      users.1348018682.couch     <--- older version of sharded users DB (note timestamp in filename)
      users.1348020303.couch     <--- latest version of sharded users DB
    userfoo/   <--- all dbs for user userfoo
      testdb1.1351859018.couch   <--- sharded version testdb1 db for user userfoo
      testdb2.1348081688.couch
      testdb3.1348081700.couch
      testdb4.1348394170.couch
      testdb4.1348394460.couch
      ..
      testdb303.1348081734.couch
  0aaaaaaa-15555553
  ...
  eaaaaa9c-f5555545
  f5555546-ffffffff
```

View Index files

```
/srv/cloudant/view_index/.shards/ <--- where view index shards are stored (not different from db files)
  00000000-0aaaaaa9/
  0aaaaaaa-15555553/              <--- per shard directory
    cluster001/                   <--- cluster user
      stats.1348021480_design           <--- stats db design doc
      users.1348020303_design           <--- users db design doc
    userfoo/                 <--- user directory
      testdb1.1351859018_design    <--- design docs for db testdb1
      testdb2.1348081688_design
      testdb3.1348081700_design
      testdb4.1348394460_design
      ...
      testdb303.1348081734_design
  ...
  f5555546-ffffffff
```

Search files

```
/srv/cloudant/search_index/shards/     <--- search index shards
  00000000-0aaaaaa9/
  0aaaaaaa-15555553/          <--- per shard directory
    userfoo/            <--- user directory
      testdb1.1350387023      <--- per user, per shard DB search index
      testdb2.1348081700
      testdb3.1354707402
```

```
    testdb4.1348081723
    testdb4.1351155794
...
f5555546-ffffffff/
```

Geo Index files

Geo index files are stored in `/srv/cloudant/geo_index/`.

Troubleshooting elevated request latencies

Use these instructions to troubleshoot elevated request latencies.

Latencies on any cluster vary in response to various internal and external factors. In this context, 'elevated' means 'elevated beyond an acceptable tolerance'. What level of latencies is acceptable to you, depends on your application. Elevated latencies are difficult to debug. While a predefined solution is impossible, the following paragraphs offer some guidance.

Is there a wider context?

The increased latency might be related to ongoing work. Determine whether any other operations work is ongoing and be aware of the following issues.

- Cluster expansions or contractions
- Node replacements
- Planned maintenance

Has anything changed?

It is unusual for latencies to increase of their own accord. The following examples show changes that can affect request latency.

Workload
: Determine whether a correlation exists between request volume and latency. Request volume alone does not give a clear picture of the workload. It is worth breaking down the traffic by HTTP request type and request size when you are looking for changes. You can derive this information from the logs that are generated for each node. Also, look for changes in the number of `active_tasks` ongoing on the cluster. For example, a correlating increase in `changes_pending` for an indexer job might indicate that a new design document was uploaded and the cluster is undertaking an extra indexing workload. You can find information about indexing tasks in the logs and through the `/_active_tasks` endpoint. If the workload profile changed, you might be able to manage the increased latency by using IOQ priorities.

Cluster configuration
: Determine whether any changes were made recently to the cluster configuration. You can get a list of recent configuration changes for a cluster from the logs. If any changes were made that correlate with increased latency, determine the context in which those changes were made to determine whether they can be reverted.

Hardware failure
: This problem might be the result of the full or partial failure of the servers, disks, networking components, or related failures. Consider the following questions when you are diagnosing the problem. Do any cluster nodes have disk errors? Are disk operation times or volumes anomalous on one or more cluster nodes? Have any cluster nodes recently become unavailable?

Are elevated latencies isolated to a particular account, database, or request type?

Take a closer at these items to find the cause of the problem.

- Logs show that most requests go to one or a few databases
- Have the same request type (such as the same HTTP method)
- Have the same backend name

Latencies for a particular database

If latencies are elevated only for a particular database, confirm whether a corresponding change in the traffic profile for that database exists.

Check that the database is evenly balanced across the cluster by inspecting the shard map.

```
ssh db1.cluster001.cloudant.com curl \
 -X GET http://localhost:5986/dbs/backend_name%2Fdatabase_name | python -m json.tool | less
```

Look at the `by_node` key and determine whether the shards are spread evenly across the cluster. If the shards are not evenly distributed, rebalancing the shards might improve the situation. For more information, see the documentation about rebalancing a cluster.

Latencies for view requests

If latencies are elevated only for a specific view (or view `group/design_doc`), check whether the affected requests are using `stale=ok`. If they are using `stale=ok`, check the distribution of `ushards` for that view by running the following command in `remsh`.

```
> mem3:ushards("db_name").
```

If the `ushards` are not evenly distributed (meaning every node has the same number of `ushards`), you can change the order of the nodes for each range under `by_range`. If you need assistance, contact support.

If they are not using `stale=ok`, check for ongoing view builds for the related design documents. If index builds are ongoing, which might be a result of a new design document version or a large volume of data ingress, increased latency for `stale=false` queries is expected. You might be able to speed up the index builds by increasing the IOQ priority or setting an IOQ bypass. However, this approach can cause other changes in cluster performance, so proceed carefully. For example, change one thing at a time and monitor the situation closely.

To change IOQ priority for views, do the following steps in a `remsh`.

```
> rpc:multicall(config, set, ["ioq", "views", "1", "Speed up index builds"]).
```

Set an IOQ bypass.

```
> rpc:multicall(config, set, ["ioq.bypass", "view_update", "true", "Speed up index builds"]).
```

Are the elevated latencies converging on a specific value?

If the request times for affected requests are converging around specific values, an internal timeout might be triggering. Typical timeout values are `5000ms`, `60000ms`, and `3600000ms`, though check `/opt/cloudant/etc/local.ini` to see whether any custom timeout values are set.

If a timeout is triggering, that might help you determine which part of the system is slowing down the requests.

If you can determine that one node is responsible for the timeouts, put that node into maintenance mode to see whether performance improves. However, first do the appropriate checks to determine whether maintenance mode is safe.

Set maintenance mode in `remsh`.

```
> config:set("cloudant", "maintenance_mode", "true").
```

Do any recorded metrics correlate with latency increases?

IOQ latency

Increased IOQ latencies (check the metrics application) often correlate with increased HTTP request latencies. However, this metric alone is not enough to determine whether requests are IOQ bound or IO bound. If `disk.sdb.disk_time` is not elevated, the request might be genuinely IOQ bound, in which case you might achieve some gains by tuning IOQ. Consider increasing IOQ/concurrency.

```
> rpc:multicall(config, set, ["ioq", "concurrency", "50", "Attempt to reduce IOQ latency"]).
```

Also, consider doing the following steps.

Increase IOQ priority for a particular request class.

```
> rpc:multicall(config, set, ["ioq", "writes", "1", "Attempt to reduce IOQ latency"]).
```

Decrease IOQ priority for a competing request class, but first verify in metrics whether another class of request is competing.

```
> rpc:multicall(config, set, ["ioq", "views", "0.1", "Attempt to reduce IOQ latency"]).
```

Set an IOQ bypass for a particular request class.

```
> rpc:multicall(config, set, \
  ["ioq.bypass", "interactive", "true", "Attempt to reduce IOQ latency"]).
```

Load average and the Erlang run queue

These metrics are analogous. Load is the system load average, that is, the number of processes in a runnable state that are waiting to run or displayed by top or uptime. Erlang run queue length is the number of processes in the Erlang VM in a runnable state (available in the metrics application).

If correlating increases exist here, the system might be overloaded. Check whether the metrics are elevated across the cluster or whether the problem is limited to one node. If the problem is local to a subset of nodes, investigate those nodes and check the shard distribution on the cluster.

If load average is growing, but the Erlang run queue is not, check the affected nodes for other processes. `collectd` or the JVM might be using too many resources, leading to poor performance of the Erlang VM. If no obvious signs of a specific problem exist, the cluster is balanced and the increases are not limited to a single node, the cluster might be overloaded. Verify that a correlating increase in requests exists and consider adding more nodes to the cluster.

Tuning parameters for common replication cases

Use these instructions to tune parameters for common replication cases.

The following parameters impact the performance of a replication task.

`worker_processes`
Number of concurrent processes for this replication.

`worker_batch_size`
Number of documents each worker processes in a single batch.

`http_connections`
Maximum total number of HTTP connections.

`connection_timeout`
Number of milliseconds until an HTTP connection times out.

The optimal values for these parameters depend on several factors.

- Document size
- Number/size of attachments

- Write load of the database
- Number of replication tasks that are running concurrently
- Total load of the cluster

The following scenarios and suggestions include the values to use.

High Throughput, Small Documents

If you want to optimize for high throughput and most documents in the database are small (about 1 KB), try to keep `worker_processes` to less than the number of cores on the mediating server. The `worker_batch_size` must not be larger than 500 to avoid errors (such as dropped connections) that would lead to the replicator to retry a large write request. The `connection_timeout` value is increased from the default of 30000 for the same reason.

```
{
  "source": "...",
  "target": "...",
  "worker_processes": 20,
  "worker_batch_size": 500,
  "http_connections": 300,
  "connection_timeout": 60000
}
```

High Throughput with Attachments

A database that contains documents with large or variable numbers of attachments requires different treatment. Increase worker count (`worker_processes`) and decrease the amount of work that is done by each worker (`worker_batch_size`) to avoid the case where all workers are tied up replicating one slow attachment each.

```
{
  "source": "...",
  "target": "...",
  "worker_processes": 100,
  "worker_batch_size": 20,
  "http_connections": 150,
  "connection_timeout": 60000
}
```

Many Concurrent Replications

If you want to run many replications concurrently, each replication job must not use too many resources, so set `worker_processes` to 1.

```
{
  "source": "...",
  "target": "...",
  "worker_processes": 1,
  "worker_batch_size": 100,
  "http_connections": 1,
  "connection_timeout": 60000
}
```

Background Replication

To keep performance impact on the source or target server to a minimum, you can use these settings, although these settings might not deliver much throughput. The `connection_timeout` is increased because the server might be heavily loaded and pushing this replication into the background.

```
{
  "source": "...",
  "target": "...",
  "worker_processes": 1,
  "worker_batch_size": 100,
  "http_connections": 5,
  "connection_timeout": 300000
}
```

Understanding and tuning the Cloudant Local automatic compactor

Read these instructions to understand how the Cloudant Local automatic compactor works and how you can tune it. The automatic compactor for Cloudant databases is an application that is called Smoosh. Smoosh automatically selects and processes the compacting of database shards on each node.

Smoosh Channels

Smoosh uses the concept of channels. A channel is essentially a queue of pending compactions. Database and view compactions use separate channels. Each channel is assigned a configuration that defines whether a compaction ends up in the queue for the channel and how compactions are prioritized within that queue. Smoosh takes each channel and works through the compactions queued in each in priority order. Each channel is processed concurrently, so the priority levels only matter within each channel. Finally, each channel has an assigned number of active compactions, which defines how many compactions happen in parallel for that channel. For example, a cluster with many database writes but few views might require more active compactions to the database channels. A channel is local to a `dbcore` node, so each node maintains and processes an independent set of compactions.

Channel configuration options

Channel types

Each channel has a basic type for the algorithm it uses to select pending compactions for its queue and how it prioritizes them. The following list describes the two queue types.

Ratio Uses the ratio `total_bytes / user_bytes` as its driving calculation. The result X must be greater than some configurable value Y for a compaction to be added to the queue. Compactions are then prioritized for higher values of X.

slack Uses the `total_bytes - user_bytes` as its driving calculation. The result X must be greater than some configurable value Y for a compaction to be added to the queue. Compactions are prioritized for higher values of X.

In both cases, Y is set with the `min_priority` configuration variable. The calculation of X is described in "Priority calculation". Both algorithms use the following two measures.

user_bytes
The amount of data the user has in the file. It does not include storage, such as old revisions or on-disk `btree` structure.

total_bytes
The size of the file on disk.

Channel type is set with the priority configuration setting.

Further configuration options

Beyond its basic type, you can apply several other configuration options to a queue. All options must be set as strings. For all settings and defaults, see the Smoosh configuration section.

Priority calculation

The algorithm type and certain configuration options feed into the priority calculation. The priority is calculated when a compaction is enqueued. Since each channel has a different

configuration, each channel ends up with a different priority value. The enqueue code checks each channel in turn to see whether the compaction passes its configured priority threshold (`min_priority`). After a channel is found that can accept the compaction, the compaction is added to the queue for that channel and the enqueue process stops. Therefore, the ordering of channels has a bearing in what channel a compaction ends up in.

Background Detail

To get metadata about a shard, you can query `$host:5986/shards%2F$shard%2f$dbname`, where `$host` is the database node on which the shard is `$shard` and `$dbname` is the name of the database. To get a list of all shards of a database, query the `/$dbname/_shards` endpoint.

```
$ curl -X GET 'http://localhost:5986/shards%2F$shard%2Fdefault%2F$database.$timestamp'
```

You can find the string to replace the `$timestamp` value in this directory, `/srv/cloudant/db/shards/`. Look for filenames that start with the database name and copy the number that follows the database name. For example, the following files are in a database named `authorization2` under the directory, `/srv/cloudant/`.

```
./db/shards/20000000-3fffffff/default/authorization2.1434466035.couch
./db/shards/c0000000-dfffffff/default/authorization2.1434466035.couch
./db/shards/a0000000-bfffffff/default/authorization2.1434466035.couch
./db/shards/e0000000-ffffffff/default/authorization2.1434466035.couch
./db/shards/80000000-9fffffff/default/authorization2.1434466035.couch
./db/shards/00000000-1fffffff/default/authorization2.1434466035.couch
./db/shards/40000000-5fffffff/default/authorization2.1434466035.couch
./db/shards/60000000-7fffffff/default/authorization2.1434466035.couchss
```

You can use the following query to collect shard metadata for this example.

```
 curl -X GET
'http://localhost:5986shards%2F00000000-1fffffff%2Fdefault%2Fauthorization2.1434466035'
```

This command returns a JSON structure like the one shown in the following example, where the `data_size` field corresponds to `user_bytes` in the priority calculation for ratio or slack channels.

```
{
    "committed_update_seq": 4,
    "compact_running": false,
    "compacted_seq": 0,
    "data_size": 1115,
    "db_name": "shards/00000000-1fffffff/exampleDB",
    "disk_format_version": 6,
    "disk_size": 2740400,
    "doc_count": 0,
    "doc_del_count": 0,
    "instance_start_time": "1412077801084432",
    "other": {
        "data_size": 0
    },
    "purge_seq": 0,
    "sizes": {
        "active": 1115,
        "external": 0,
        "file": 2740400
    },
    "update_seq": 4,
    "uuid": "a1995bbab83dee7072b69b640942a81b"
}
```

The `user_bytes` are called `data_size` in `db_info` blocks. It is the total of all bytes that are used to store docs and their attachments. Since `.couch` files are append only, every update adds data to the file. When you update a `btree`, a new leaf node is written and all the nodes back up the root. In this update, old data is never overwritten and these parts of the file are no longer live, including old `btree` nodes and document bodies. Compaction takes this file and writes a new file that contains only live data. `disk_size` is the number of bytes in the file as reported by `ls -al filename`.

Defining a channel

Defining a channel is done through normal `dbcore` configuration, with some convention as to the parameter names. Channel configuration is defined by using `smoosh.channel_name` top-level configuration options. Defining a channel involves setting the options that you want for the channel, then bringing it into the sets of active channels for Smoosh by adding it to either `db_channels` or `view_channels`. This process means that Smoosh channels can be defined either for a single node or globally across a cluster, by setting the configuration either globally or locally. In the following example, a new global channel is set up. It is important to choose good channel names. The following conventional channel names are defined by default if no other channel names are set.

`ratio_dbs`
: A ratio channel for databases that usually uses the default settings.

`slack_dbs`
: A slack channel for databases that usually uses the default settings.

`ratio_views`
: A ratio channel for views that usually uses the default settings.

`slack_views`
: A slack channel for views that usually uses the default settings.

This standard definition channel name must be added often.

`big_dbs`
: A ratio channel for enqueuing only large database shards. The term large is workload-specific.

Channels have certain defaults for their configuration, which are defined in the Smoosh configuration section. It is only necessary to set up how this channel differs from those defaults. In the following example, the `min_size` and concurrency settings are set, and keep the priority as the default to ratio, along with the other defaults.

```
# Define the new channel
(dbcore@db1.cluster001.cloudant.net)3> \
  rpc:multicall(config, set, ["smoosh.big_dbs", "min_size", "20000000000"]).
{[ok,ok,ok],[]}
(dbcore@db1.cluster001.cloudant.net)3> \
  rpc:multicall(config, set, ["smoosh.big_dbs", "concurrency", "2"]).
{[ok,ok,ok],[]}

# Add the channel to the db_channels set -- note we need to get the original
# value first so we can add the new one to the existing list!
(dbcore@db1.cluster001.cloudant.net)5> \
  rpc:multicall(config, get, ["smoosh", "db_channels"]).
{[{'dbcore@db1.cluster001.cloudant.net',"ratio_dbs"},
{'dbcore@db3.cluster001.cloudant.net',"ratio_dbs"},
{'dbcore@db2.cluster001.cloudant.net',"ratio_dbs"}],
[]}
(dbcore@db1.cluster001.cloudant.net)6> \
  rpc:multicall(config, set, ["smoosh", "db_channels", "ratio_dbs,big_dbs"]).
{[ok,ok,ok],[]}
```

Viewing active channels

```
(dbcore@db3.cluster001.cloudant.net)3> \
  rpc:multicall(config, get, ["smoosh", "db_channels"]).
{[{'dbcore@db3.cluster001.cloudant.net',"ratio_dbs,big_dbs"},
  {'dbcore@db1.cluster001.cloudant.net',"ratio_dbs,big_dbs"},
  {'dbcore@db2.cluster001.cloudant.net',"ratio_dbs,big_dbs"}],
 []}
(dbcore@db3.cluster001.cloudant.net)4> \
  rpc:multicall(config, get, ["smoosh", "view_channels"]).
```

```
{[{'dbcore@db3.cluster001.cloudant.net',"ratio_views"},
  {'dbcore@db1.cluster001.cloudant.net',"ratio_views"},
  {'dbcore@db2.cluster001.cloudant.net',"ratio_views"}],
 []}
```

Removing a channel

```
# Remove it from the active set
(dbcore@db1.cluster001.cloudant.net)5> \
  rpc:multicall(config, get, ["smoosh", "db_channels"]).
{[{'dbcore@db1.cluster001.cloudant.net',"ratio_dbs,big_dbs"},
{'dbcore@db3.cluster001.cloudant.net',"ratio_dbs,big_dbs"},
{'dbcore@db2.cluster001.cloudant.net',"ratio_dbs,big_dbs"}],
[]}
(dbcore@db1.cluster001.cloudant.net)6> \
  rpc:multicall(config, set, ["smoosh", "db_channels", "ratio_dbs"]).
{[ok,ok,ok],[]}

# Delete the config -- you need to do each value
(dbcore@db1.cluster001.cloudant.net)3> \
  rpc:multicall(config, delete, ["smoosh.big_dbs", "concurrency"]).
{[ok,ok,ok],[]}
(dbcore@db1.cluster001.cloudant.net)3> \
  rpc:multicall(config, delete, ["smoosh.big_dbs", "min_size"]).
{[ok,ok,ok],[]}
```

Getting channel configuration

```
(dbcore@db1.cluster001.cloudant.net)1> \
  rpc:multicall(config, get, ["smoosh.big_dbs", "concurrency"]).
{[{'dbcore@db3.cluster001.cloudant.net',"2"},
  {'dbcore@db1.cluster001.cloudant.net',"2"},
  {'dbcore@db2.cluster001.cloudant.net',"2"}],
 []}
```

Setting channel configuration

Set the channel configuration in the same way as defining a channel by setting the new value.

```
dbcore@db1.cluster001.cloudant.net)2> \
  rpc:multicall(config, set, ["smoosh.ratio_dbs", "concurrency", "1"]).
{[ok,ok,ok],[]}
```

It sometimes takes time to take effect on disk usage.

Standard operating procedures

Operators must do a few standard things when they respond to issues. In addition to the following items, it is useful in some circumstances to define new channels with certain properties (`big_dbs` is a common one) if Smoosh is not selecting and prioritizing compactions that well.

Checking the status of Smoosh

You can see the queued items for each channel by going into `remsh` on a node and running the `smoosh:status` command.

```
> smoosh:status().
{ok,[{"ratio_dbs",
      [{active,1},
       {starting,0},
       {waiting,[{size,522},
         {min,{5.001569007970237,{1378,394651,323864}}},
         {max,{981756.5441159063,{1380,370286,655752}}}]}]},
     {"slack_views",
      [{active,1},
       {starting,0},
       {waiting,[{size,819},
         {min,{16839814,{1375,978920,326458}}},
         {max,{1541336279,{1380,370205,709896}}}]}]},
     {"slack_dbs",
      [{active,1},
       {starting,0},
       {waiting,[{size,286},
```

```
        {min,{19004944,{1380,295245,887295}}},
        {max,{48770817098,{1380,370185,876596}}}]}]},
    {"ratio_views",
     [{active,1},
      {starting,0},
      {waiting,[{size,639},
        {min,{5.0126340031149335,{1380,186581,445489}}},
        {max,{10275.555632057285,{1380,370411,421477}}}]}]}]}
```

The results give you the node-local status for each queue. You can see more information under each channel, such as what is shown in the following list.

Active Number of current compactions in the channel.

Starting
Number of compactions that are beginning.

Waiting
Number of queued compactions.

Min and max give an idea of the effectiveness of the queued jobs. The values for these items depend on whether the queue is ratio or slack.

For ratio queues, the default minimum for Smoosh to enqueue a compaction is 5. In the previous example, you can guess that 981,756 is high. However, this database might be small, so it does not necessarily mean useful compactions from the point of view of reclaiming disk space.

For this example, many queued compactions are waiting, but it is unknown which compaction would be most effective to run to reclaim disk space. It is worth noting that the waiting queue sizes are only meaningful relative to other factors on the cluster, such as database number and size.

Smoosh IOQ priority

This setting is a global setting that affects all channels. Increasing it allows each active compaction to proceed faster as the compaction work is of a higher priority relative to other jobs. Decreasing it has the inverse effect. By this point, you know whether Smoosh is backing up. If it is falling behind (large queues), try increasing compaction priority. The IOQ priority for Smoosh is controlled through the IOQ compaction queue.

```
> rpc:multicall(config, get, ["ioq", "compaction"]).
{[{'dbcore@db1.cluster001.cloudant.net',undefined},
  {'dbcore@db2.cluster001.cloudant.net',undefined},
  {'dbcore@db3.cluster001.cloudant.net',undefined}],
 []}
```

Priority by convention runs 0 - 1, though the priority can be any positive number. The default for compaction is 0.01. If it looks like Smoosh has too much work that is not getting through, you can increase the priority. However, be careful that this order does not adversely impact request performance.

```
rpc:multicall(config, set, ["ioq", "compaction", "0.5"]).
{[ok,ok,ok],[]}
```

In general, adjusting the Smoosh IOQ priority is a temporary measure. For some clusters, a change from the default might be required to help Smoosh keep up with particular workloads.

Granting specific channels more workers

Giving Smoosh a higher concurrency for each channel can allow a backlog in that channel to be processed. Again, some clusters run best when specific channels have more workers. When you assess disk space, you must know whether the biggest offenders are database or view files. From your assessment, you can infer whether it is worth giving a specific Smoosh channel a higher concurrency. The current setting can be seen for a channel like so.

```
> rpc:multicall(config, get, ["smoosh.ratio_dbs", "concurrency"]).
{[{'dbcore@db1.cluster001.cloudant.net',undefined},
  {'dbcore@db2.cluster001.cloudant.net',undefined},
  {'dbcore@db3.cluster001.cloudant.net',undefined}],
 []}
```

`undefined` means that the default is used.

If you know that databases are the major user of disk space, you might want to increase a `_dbs channel`. Experience shows `ratio_dbs` are often best, but you must evaluate the databases based on their status. You can increase the `ratio_dbs` setting by using the following command.

```
> rpc:multicall(config, set, ["smoosh.ratio_dbs", "concurrency", "2"]).
{[ok,ok,ok],[]}
```

Suspending Smoosh

If Smoosh is causing issues, you can suspend its operation. Suspending Smoosh differs from stop (smoosh) or setting the concurrency for all channels to zero. Suspending Smoosh pauses ongoing compactions and maintains the channel queues intact. For example, if the compactions for a node are causing disk space issues, you can suspend Smoosh while you determine which channel is causing the problem. For example, a `big_dbs` channel might be creating huge compaction-in-progress files if there is not much in the shard to compact. Therefore, it is useful to use when you are testing to see whether Smoosh is causing a problem.

```
# suspend
smoosh:suspend().

# resume a suspended smoosh
smoosh:resume().
```

Suspend is implemented by calling `erlang:suspend_process(Pid, [unless_suspending])` for each compaction process in each channel. `resume_process` is called for resume.

Restarting Smoosh

Restarting Smoosh is a long shot and is a brute force approach in the hope that when Smoosh rescans the databases that it makes the right decisions. If this step is required, contact `support@cloudant.com`, since doing the step might indicate a problem with Smoosh.

Configuring Smoosh

Use these instructions to configure Smoosh, the Cloudant auto-compaction daemon. Smoosh is notified when databases and views are updated, and it might then elect to enqueue them for compaction.

Top-Level Settings

The following settings are the main settings that you use with Smoosh.

`db_channels`

A comma-separated list of channel names for databases.

`view_channels`

A comma-separated list of channel names for views.

`staleness`

The number of minutes that the (expensive) priority calculation can be stale before it is recalculated. The default value is 5 minutes.

Sometimes it is necessary to use the following items.

`cleanup_index_files`

Whether Smoosh cleans up the files for indexes that were deleted, it defaults to false and must not be changed unless the cluster is running low on disk space, and only after you consider the ramifications.

wait_secs
: The time a channel waits for compactions to allow time to observe the system and decide what to compact first. Rarely changed from the default. The default value is 30 seconds.

Channel Settings

A channel has several important settings that control runtime behavior.

capacity
: The maximum number of items the channel can hold (lowest priority item is removed to make room for new items). Defaults to 9999.

concurrency
: The maximum number of jobs that can run concurrently. Defaults to 1.

max_priority
: The item must have a priority lower than this setting to be enqueued. Defaults to infinity.

max_size
: The item must be no larger than the value specified in length to be enqueued. Defaults to infinity.

min_priority
: The item must have a priority at least as high as this setting to be enqueued. Defaults to 5.0 for ratio and 16 MB for slack.

min_size
: The item must be at least the value that is specified in length to be enqueued. Defaults to 1 MB (1048576 bytes).

priority
: The method that is used to calculate priority. Can be ratio (calculated as `disk_size/data_size`) or slack (calculated as `disk_size-data_size`). Defaults to ratio.

Compaction Scheduling Algorithm

Smoosh decides whether to compact a database or view by evaluating the item against the selection criteria of each channel in the order they are configured. By default, two channels for databases (`ratio_dbs` and `slack_dbs`), and two channels for views (`ratio_views` and `slack_views`) exist, Smoosh the new item to the first channel that accepts it. If no channel accepts, the item is not enqueued for compaction.

Example config commands

Change the set of database channels.

```
config:set("smoosh", "db_channels", "small_dbs,medium_dbs,large_dbs").
```

Change the set of database channels on all live nodes in the cluster.

```
rpc:multicall(config, set, ["smoosh", "db_channels", "small_dbs,medium_dbs,large_dbs"]).
```

Change the concurrency of the `ratio_dbs` database channel to 2.

```
config:set("smoosh.ratio_dbs", "concurrency", "2").
```

Change it on all live nodes in the cluster.

```
rpc:multicall(config, set, ["smoosh.ratio_dbs", "concurrency", "2"]).
```

Example API commands

```
smoosh:status()
```

This command prints the state of each channel, how many jobs are currently running, and how many jobs are enqueued, including the lowest and highest priority of those enqueued items. The idea is to provide, at a glance, sufficient insight into Smoosh that an operator can assess whether Smoosh is adequately targeting the reclaimable space in the cluster. In general, a working correctly status output has items in the `ratio_dbs` and `ratio_views` channels. Due to the default settings, `slack_dbs` and `slack_views` contain items.

```
smoosh:enqueue_all_dbs(), smoosh:enqueue_all_views()
```

You usually do not need to use these functions, since Smoosh finds the best compaction candidates on its own. However, if you experience disk space issues, these functions might help. If they do, that indicates a bug or configuration issue with Smoosh. Check your configuration and contact support.

Chapter 15. A guide to IOQ for operators

Read this information to learn how the Cloudant IO queue (IOQ) works and how you can configure and tune it.

IOQ handles the prioritization of IO operations in the database. It has the following main responsibilities.

1. Providing configurable prioritization of interactive requests and background tasks such as compaction and internal replication.
2. Providing equal prioritization for interactive requests by backend or database.

From an operational perspective, the first point is of most interest as it provides a set of levers that can be pulled to change the behavior of the cluster in favor of particular workloads or operational concerns.

Basic overview

From an operational point-of-view, IOQ carries out the following fundamental operations.

1. Enqueuing requests into one of a number of available channels.
2. Selecting and submitting a request from the available channels according to configured priorities.

IOQ categorizes IO requests by class and by priority. The class of a request dictates the channel into which it is enqueued and the priority influences the probability that a request is dequeued and executed. The following table lists the IOQ classes and the corresponding priorities.

Note: Mapping of IOQ classes to class priorities is not 1:1.

IOQ class	IOQ priority	Description
interactive	reads, writes	IO requests related to requests made by users through the http layer.
`db_update`	writes	Interactive IO requests that are database write operations.
`view_update`	views	IO requests related to view index builds.
`db_compact`	compaction	IO requests related to database compactions.
`view_compact`	compaction	IO requests related to view compactions.
`internal_repl`	replication	IO requests related to internal replication, that is, replication between nodes in a cluster.
low	low	IO requests related to requests made by users through the http layer where the `x-cloudant-priority` header is set to `low`.
other	undefined	IO requests that do not fit any of the previous classes, including search IO requests.

Internals

To understand the relationship between the IOQ classes and the IOQ priorities, it is useful to understand the channels into which IO requests are enqueued. IOQ uses the following channels.

- Compaction
- Internal replication
- Low

- Customer

The Customer channel is effectively a meta-channel where each item in the queue represents a backend / database name combination that consists of a more channels.

- Interactive
- DB update
- View update

Requests are enqueued according to the following scheme.

- Requests with class `internal_repl`, low, `db_compact`, or `view_compact` are enqueued into Internal replication, Low, or Compaction channels.
- Requests with class interactive, `db_update` or `view_update` are enqueued into the Interactive, DB update, or View update channel of the relevant Customer channel for the backend/database combination.
- Requests with class other are enqueued into the Interactive queue of a Customer channel that is reserved for other IOQ requests.

Requests are submitted by using the following processes.

- The next item is selected from either the Compaction, Internal replication, Low, or Customer channel according to the configured priorities (compaction, replication, low, and customer).
- If the item is obtained from the Compaction, Internal replication, or Low channels then the request is submitted for execution.
- If the item is obtained from the Customer channel, the request is selected from either the Interactive, DB update, or View update channel according to the configured priorities (reads, writes, and views).

Configuration

Unless there is prior knowledge of the IOQ configuration that is required to support the intended workload of a cluster on a given hardware specification, it is a best practice that the IOQ is initially left with the default configuration values. As more becomes known about the behavior of a cluster under load, the IOQ settings can be tuned to provide optimal performance for the production workload. Tuning IOQ is not the answer to all performance problems, and there are a finite number of gains to be had (possibly zero). You must also consider the total load on the cluster, the capabilities of the underlying hardware, and the usage patterns and design of the applications that sit on top of the data layer.

Priorities

The default IOQ configuration that gives interactive reads/writes and view builds a high priority (1.0) and the background requests a much lower priority (0.001 for compaction and 0.0001 for replication and low). You can set the priorities to other values with the `config` app in a `remsh` session, as shown here.

```
config:set("ioq", "views", "0.5", "FBXXXXX reduce views IOQ priority").
```

To return to the default value, delete the configuration value.

```
config:delete("ioq", "views", "FBXXXXX revert to default priority").
```

Ensuing sections describe situations where tuning IOQ priorities might be appropriate.

Internal replication backlog

If cluster nodes are frequently exhibiting an internal replication backlog, consider increasing the replication priority. You can confirm a backlog by checking the logs for `erlang.internal_replication_jobs`.

If this value is consistently elevated by more than a few hundred changes, try increasing the replication IOQ priority.

```
config:set("ioq", "replication", "0.5", \
"FBXXXXX speed up internal replication").
```

If this is effective, you see a change in the rate at which the metric decreases. It is worth experimenting with values as high as 1.0. However, keep an eye on HTTP request latencies to make sure there is no adverse impact on other aspects of cluster performance.

Compactions not completing quickly enough

If disk usage is rising on cluster nodes and there is a corresponding backlog in compaction work, consider increasing the compaction priority. Check the volume of pending changes for ongoing compaction in the logs by looking for `changes_pending.*compaction`.

Increase the priority for compaction.

```
config:set("ioq", "compaction", "0.5", \
"FBXXXXX speed up compaction").
```

Now, monitor the `changes_pending` metrics to see whether the rate at which changes are being processed has increased. Experiment with values as high as 1.0, if necessary, and keep an eye on cluster performance.

Interactive requests and views competing for IO resource

Metrics might show that read/write performance worsens when views are building or conversely that view build performance slows when read/write load increases. If the performance requirements of the cluster are such that a particular type of request is more critical to the application it supports, consider reducing the other IOQ priorities. Here is an example.

```
config:set("ioq", "views", "0.1", \
 "FBXXXXX attempt to improve read/write performance").
```

Concurrency

Concurrency defines the total number of concurrent IO operations that are allowed by IOQ. The default value is 20, but you might want to increase this number in the following circumstances.

1. The number of active requests or IOQ latency are consistently elevated.
2. Disk usage is significantly less than 100%.

If performance is impacted by requests that are waiting in the queues, consider increasing IOQ concurrency (sensible values to try are 30, 50, and 100) and observing the resulting effect. However, be aware that increasing this value beyond a certain point can result in the disks being overloaded and overall performance degradation. The exact point depends on the cluster workload and hardware, so it is important to monitor the cluster when you are making changes.

Bypasses

In extreme cases, it is possible that IOQ is the bottleneck for certain request classes. If so, you can bypass IOQ for that request class. For example, for interactive requests.

```
config:set("ioq.bypass", "interactive", "true", \
 "FBXXXXX attempt to improve interactive performance").
```

Bypasses are set for IOQ classes, not IOQ priorities. Therefore, if you want to bypass all compaction requests, set a bypass for `db_compact` and `view_compact`. When you consider setting an IOQ bypass, be aware of the following possibilities.

- Other request classes continue to be routed through IOQ and are not be able to compete with the bypassed requests. Monitor the cluster carefully to determine whether overall performance is acceptable. In particular, keep an eye on compaction (unless it is being bypassed) because, if the rate of compaction slows too much, the disk might start filling up.
- The bypass effectively shifts the bottleneck to another part of the system, which is typically evident in `couch_file` and `couch_db_updater` message queue backups.
- Disk performance might also become saturated, which might lead to performance degradation.

Generally, it is a good idea to avoid IOQ bypasses.

Chapter 16. Appendix: Install Cloudant Local Offline

The instructions in this appendix describe how to install Cloudant Local offline.

An online system has internet access and full access to the package repositories, while an offline system has no access to the internet. You must download both the third-party packages and Python dependencies, bundle the packages and dependencies, and transfer and install them on the offline system.

Prerequisites

Before you install Cloudant Local offline, you must meet the following prerequisites.

1. You must be a root user or have sudo privileges.
2. Complete the steps to extract the Cloudant Local installation packages.

Before you can install Cloudant Local offline, you must run a download-only or dry-run installation. You run a dry-run installation from a system that has internet access and runs the same version of the operating system as the target system. The new system is called the online system.

Install Third-party Package Dependencies

Cloudant Local depends on several third-party packages. Before you install Cloudant Local, the third-party packages must be installed on the offline system. You can use the dry-run installation option to bundle and transfer the dependencies to your offline system. The dry-run installation option is supported by the operating system package managers that Cloudant Local uses.

You must configure access to the following system types on the online system.

- Operating system-specific package repositories (EPEL, and so on)
- Python Package Index (PyPI) site

Install Red Hat Dependencies

For Red Hat and CentOS platforms, you must bundle the dependencies and transfer them to the offline system.

1. Log in to your Red Hat online system.
2. Bundle the dependencies into a compressed (tar) file by running the following commands.

```
yum install epel-release
yum install createrepo
mkdir cloudant-depss
yum install -y —-downloadonly —downloaddir=cloudant-deps cloudant-dbnode cloudant-lbnode
createrepo cloudant-deps
tar -czvf cloudant-deps.tar.gz cloudant-deps
```

3. Copy the file to the offline system.

```
cp cloudant-deps.tar.gz offline.system.com:/tmp
```

4. Log in to the offline system.
5. Set up the repository for the Cloudant dependencies.

```
cd /tmp
tar -xvf cloudant-deps.tar.gz

echo "[cloudant-deps]" > /etc/yum.repos.d/cloudant-deps.repo
echo "name=IBM, Cloudant Dependencies" >> /etc/yum.repos.d/cloudant-deps.repo
echo "baseurl=file:///tmp/cloudant-deps >> /etc/yum.repos.d/cloudant-deps.repo
echo "enabled = 1" >> /etc/yum.repos.d/cloudant-deps.repo
```

```
echo "gpgcheck = 0" >> /etc/yum.repos.d/cloudant-deps.repo

yum clean all
yum install -y python-pip
yum install -y python-devel
```

Install SUSE Dependencies

You must bundle the SUSE dependencies and transfer them to the offline system.

1. Log in to your SUSE online system.
2. Download and bundle the dependencies into a compressed (tar) file by running the following commands.

```
zypper -n install createrepo
mkdir cloudant-deps
zypper -n –pkg-cache-dir cloudant-deps install –download-only cloudant-dbnode cloudant-lbnode
createrepo cloudant-deps
tar -czvf cloudant-deps.tar.gz cloudant-deps
```

3. Copy the compressed (tar) file to the offline system.

```
scp cloudant-deps.tar.gz offline.system.com:/tmp
```

4. Log in to the offline system.
5. Set up the repository for the Cloudant dependencies.

```
cd /tmp
tar -xvf cloudant-deps.tar.gz
zypper -n addrepo –no-gpgcheck /tmp/cloudant-deps cloudant-deps
zypper -n install python-pip
zypper -n install python-devel
```

Install Ubuntu System Dependencies

Download and transfer the apt-offline package to the offline system.

1. Log in to the Ubuntu online system.

```
apt-get download apt-offline
scp apt-offline* offline.system.com:/tmp
```

2. Log in to the offline system.

```
cd /tmp
dpkg -i apt-offline*
apt-offline set cloudant-deps.sig –install-packages cloudant-dbnode cloudant-lbnode python-pip
```

3. Copy the required dependency list back to the online system.

```
scp cloudant-deps.sig online.system.com:/tmp
```

4. Log in to the online system.

```
cd /tmp
apt-offline get cloudant-deps.sig –no-checksum -d cloudant-deps
rm cloudant-deps/cloudant-*
tar -czvf cloudant-deps.tar.gz cloudant-deps
```

5. Copy the dependencies back to the offline system.

```
scp cloudant-deps.tar.gz offline.system.com:/tmp
```

6. Log in to the offline system and install the cloudant dependencies.

```
cd /tmp
tar -xvf cloudant-deps.tar.gz
apt-offline install cloudant-deps
```

Install Python Dependencies

After you transfer the third-party dependencies, you must transfer the Python dependencies from the online system.

Important: You must apply these steps to all the platforms in your environment.

1. Create a text file named `requirements.txt`.
2. Add the Python module to the `requirements.txt` file.

```
click
importlib
jinja2
pyyaml
setuptools
fabric
requests == 2.7.0
argparse >= 1.2.1
colorama >= 0.2.5
paramiko >= 1.7.7.1
retrying >= 1.2.3
semantic_version == 2.4.1
```

3. Download and bundle the modules into a compressed (tar) file.

```
mkdir cloudant-python-deps
pip install -d cloudant-python-deps -r requirements.txt
tar -czvf cloudant-python-deps.tar.gz cloudant-python-deps
```

4. Copy the files to the offline system.

```
scp cloudant-python-deps.tar.gz offline.system.com:/tmp
scp requirements.txt offline.system.com:/tmp/requirements.txt
```

5. Log in to the offline system.
6. Install the Python modules.

```
cd /tmp
tar -xvf cloudant-python-deps.tar.gz
pip install –no-index –find-linux="cloudant-python-deps" -r requirements.txt
```

Install Cloudant CAST

After you install the third-party repositories and Python modules on the offline system, you must install the CAST tool.

1. Install the Cloudant CAST tool by running the command for your environment.
   a. Install the CAST tool on a Red Hat or CentOS system.

```
yum install -y cloudant-cast
cast –help
```

   b. Install the CAST tool on a SUSE system.

```
zypper -n install cloudant-cast
cast –help
```

   c. Install the CAST tool on an Ubuntu system.

```
apt-get install cloudant-cast
cast –help
```

2. After you install the CAST tool, you can continue the installation by installing a database node or a load balancer node.
   a. Install the first database node.
   b. Install load balancer nodes.
3. Verify that the installation was successful by launching the Cloudant dashboard that uses the load balancer node URL. http://*loadbalancer_url*/dashboard.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator
Director of Engineering, Information Management (Office 16)
111 Campus Drive
Princeton, NJ 08540
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at 'Copyright and trademark information' at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Printed in USA