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 91.

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

IBM Cloudant Data Layer Local Edition Overview

The IBM® Cloudant Data Layer Local Edition (Cloudant Local) includes the same robust and flexible features that were originally available only in the Cloudant Database-as-a-Service (DBaaS) offering. Refer to this information to learn about some of the benefits of using Cloudant Local, and read the basic steps for installing the product.

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.

After you install Cloudant Local at your location, you can

- Distribute readable, 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.

Basic steps to implement Cloudant Local

The following basic steps are used to implement Cloudant Local:

1. Verify that your system meets the hardware and software requirements for Cloudant Local.
2. Plan your installation and verify that your environment is properly configured to install the product.
3. Extract and Install Cloudant Local.
4. Configure Cloudant Local.
5. In a production environment, configure your database nodes for remote logging.
6. If needed, Configure SSL on your load balancers. If you access the Cloudant database across an untrusted network, configuring SSL/HTTPS is recommended. If your network is trusted, configuring SSL/HTTPS is optional, and you can use HTTP requests (that is, port 80), instead.

Note: Cloudant Local uses the CouchDB security model for database-level security. If you want to configure database-level security for Cloudant Local, see "Configuring database-level security" on page 49. This step is optional, and is not required to configure Cloudant Local.

Installing Cloudant Local on zLinux

The information in this chapter is specific to the IBM Cloudant Data Layer Local Edition Version 1.0.0.4 for zLinux. The new information detailed here is for zLinux users only.

Hardware and Software Requirements

Before you install Cloudant Local, confirm that your system meets the following 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

Supported Platform
Linux on IBM System z - Red Hat Enterprise Linux 7.0

Overview of Cloudant Local installation for zLinux

The Cluster Admin and Support Tool (CAST) manages the installation, configuration, and support of a Cloudant Local cluster and its individual nodes. These instructions describe how to extract and install CAST and install Cloudant Local. During the Cloudant Local installation, CAST performs the following installation and configuration tasks:

- Verifies that system requirements are met before starting the installation.
- Installs the software for the type of load balancer and database node types that you specify.
- Opens the firewall ports.
- Sets the configuration files for all the services to default values or values you provide.
- Initializes the database.
- Creates security documents.
- Starts the system.

If you know the configuration values you want to use prior to installation, you can provide a node configuration file containing those values during installation using the `-c` option. The configuration file for a database node is discussed here. The configuration file for a load balancer node is discussed here.

To preserve your Extra Packages for Enterprise Linux (EPEL) user information, use the `-e` option.

The `check` option provided in CAST verifies that your system meets the prerequisites before starting the Cloudant Local installation. For other options, see the help text provided with the `cast system install` command, described here.

Important:

If security restrictions prevent you from accessing third-party repositories, you can download the required third-party packages beforehand and install them in your environment.

If using this approach, you must download and install the required third-party packages before starting the installation.

Use the `check` option provided in CAST to confirm that your system has the necessary pre-requisites. Do this by running the following command as the root user to check the database node requirements:

```
cast system check -db
```

To check the load balancer node requirements, run the following command:

```
cast system check -lb
```

Install CAST

Before you install Cloudant Local, you must install CAST on each database node and each load balancer node. You might do this by installing CAST first onto all of the nodes, and then install Cloudant Local on each of those nodes. Alternatively, you might install CAST followed immediately by installing Cloudant Local, one node at a time.

Regardless of the install sequence you choose, the steps for installing CAST on a node are as follows:

1. Extract the Cloudant Local installation packages.
   a. Download the `tar` file specifically designed for installation on zLinux.
   b. Create an empty directory where you can expand (untar) the Cloudant Local installation packages. For example, `mkdir /root/installer`.

      Move the downloaded `tar` file to the newly created directory.
   c. Unpackage the installation media within the empty directory, using the following command:

      ```
      tar –zxvf xxxxxxx.tar.gz
      ```

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

      After unpackaging the media, the directory contains the files required to install the product, including the `install.bin` file that extracts the build packages.
   d. Extract the Cloudant Local build packages using the wizard or the silent install option.
      - To extract the Cloudant Local packages using an interactive wizard, run the following command:

        ```
        ./install.bin
        ```
      - 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 will be used for production or development purposes.
2. Install Python Pip.

   You can install python-pip from the standard repository for your operating system or use the one provided by Cloudant Local. To install python-pip provided by Cloudant Local, run the following command:

   ```
   rpm -U {install-dir}/repo/{OS-Ver}/{OS-Arch}/python-pip*.rpm
   ```

   In this command:

   rpm The package manager for RPM-based Linux distributions.

   install-dir
   The base installation directory. The default is `/root/cloudant`.

   OS-Ver
   The major operating system version, for example 6, 7, or 7Server.

   OS-Arch
   The operating system architecture, for example x86_84 or s390x.

   To install python-pip on an RPM-based Linux running on the s390x architecture, you would use a command similar to the following:

   ```
   rpm -U {install-dir}/repo/7Server/s390x/python-pip*.rpm
   ```

   For example:

   ```
   rpm -U /root/cloudant/repo/7Server/s390x/python-pip*.rpm
   ```
3. Install CAST. The command is similar to that used to install python-pip.

a. Run the following command to install CAST:

```
rpm -U {install-dir}/repo/{OS-Ver}/{OS-Arch}/cloudant-cast*.rpm
```

To install CAST on an RPM-based Linux running on the s390x architecture, you would use a command similar to the following:

```
rpm -U {install-dir}/repo/7Server/s390x/cloudant-cast*.rpm
```

For example:

```
rpm -U /root/cloudant/repo/7Server/s390x/cloudant-cast*.rpm
```

CAST is now installed on your system.

b. (Optional) You can display the syntax help text for the CAST command, as follows:

```
cast system install --help
```

The help text is similar to the following:

```
Usage: cast system install [OPTIONS] REPO_DIR

Installs, configures, and starts a database/load balancer node.

Options:
-db, --dbnode       Installs a database node
-lb, --lbnode       Installs a load balancer node
-c, -config PATH    Override defaults from a node configuration file
--epel              Enables the EPEL repository
--nocheck           Bypass running the system check command
--help              Show this message and exit
```

Install Cloudant Data Layer Local Edition

Before you install Cloudant Local on a node, ensure that you successfully installed CAST.

Important: You must be the root user, or have sudo privledges, to install Cloudant Local. If you are not the root user, prefix all commands with the sudo command.

Install Database Nodes

This section covers the minimum steps needed to install and setup a Cloudant Local cluster using CAST.

1. Install the first database node.

   Important: Perform these steps on the first database node in the cluster.

   a. Run the following three commands to install the first database node:

   ```
   cast system install {install-dir}/repo
   yum install -y cloudant-dbnode
   cast system install -db {install-dir}/repo
   ```

   For example:

   ```
   cast system install /root/cloudant/repo
   yum install -y cloudant-dbnode
   cast system install -db /root/cloudant/repo
   ```

   b. Make a copy of the database node configuration file:/opt/cloudant/cast/samples/dbnode.yaml. For example:

   ```
   cp /opt/cloudant/cast/samples/dbnode.yaml /opt/cloudant/cast/samples/dbnode.yaml.original
   ```

   c. Edit the file /opt/cloudant/cast/samples/dbnode.yaml.

   The only fields that need to be changed are the passwords for the admin and cloudant database users. Change the default password to one that is unique to your installation. This password is required when executing CAST commands.

Note: You can set the value for the cookie and uuid fields instead of having one automatically generated, but the quorum values should not be changed for a cluster installation.
Here is a sample of the dbnode.yaml file before editing:

```
# Sample CAST input file to configure a database node.
# openssl - Keyword to indicate that a field's value should be generated automatically.
# The value is created using the command (openssl rand -hex 16) internally.
dbnode:
    users:
        admin: pass
        cloudant: pass
    cookie: openssl
    uuid: openssl

    quorum:
        q: 8
        r: 2
        w: 2
        n: 3
```

d. Save the dbnode.yaml file.

e. Update the database node to use the new values, by running the following command:

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

You must specify the full pathname if the dbnode.yaml file is not in the current directory.

f. Create a cluster configuration file.

Each database node of the cluster must use the same configuration file. This is required because the user passwords and cookie values must be the same on each host in order to join the nodes together into a cluster. You create the configuration file on the first database node of the cluster only.

To export the configuration values, run the following command on the first node of the cluster:

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

g. Copy the exported cluster_dbnode.yaml configuration file to a place that is accessible by the other database nodes.

2. Install additional database nodes.

a. For each additional database node in the cluster, run the following three install commands:

```
cast system install {install-dir}/repo
yum install -y cloudant-dbnode
cast system install -db {install-dir}/repo
```

For example:

```
cast system install /opt/cloudant/repo
yum install -y cloudant-dbnode
cast system install -db /opt/cloudant/repo
```

b. Retrieve the cluster_dbnode.yaml file you created with the *first* database node. Use this file as input to the following command:

```
cast node config cluster_dbnode.yaml
```

Install a Load Balancer Node

For each load balancer node you install, you must perform the following steps:

1. Run the install command:

```
cast install -lb {install-dir}/repo
```

2. Make a copy of the load balancer node configuration file /opt/cloudant/cast/samples/lbnode.yaml. For example:

```
cp /opt/cloudant/cast/samples/lbnode.yaml /opt/cloudant/cast/samples/lbnode.yaml.original
```

3. Edit the `lbnode.yaml` file. Add the hostname and ipaddress for each database node in the cluster under the nodes section.

   Important: You must change the admin password in the `lbnode.yaml` file. The password must be entered in clear text, **not** the encrypted form that appears in the `cluster_dbnode.yaml` file generated during the installation of the first database node. Ensure that the clear text admin password matches the password you specified when you configured the database nodes.

```
# Sample CAST input file for configuring a load balancer for a 3 node database cluster.
lbnode:
   users:
       admin: 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
```

4. Save the `lbnode.yaml` file.
5. Update the load balancer node to use the new values, by running the following command:

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

   You must specify the full pathname if the `lbnode.yaml` file is not in the current directory.

Join Nodes in a Cluster

After the database nodes are installed and configured, you must join them together to form the cluster.

1. Add a node.

   From any database node, run the following command for each database node in the cluster, *excluding* the one you are on:

```
cast cluster add FQDN
```

   For example, if you have three database nodes, their Fully Qualified Domain Names (FQDN) might be `db1.vm`, `db2.vm` and `db3.vm`. If you are currently on the `db1.vm` node, you must add both `db2.vm` and `db3.vm` using the following commands:

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

2. To make sure the nodes of the cluster are joined properly, run the `cast cluster show` command.

   Each node you added displays a status of online or offline. An offline status indicates that the node did not join the cluster properly. If a cluster you added displays an offline status, remove the node from the cluster, then install and configure the node again.
3. To remove a node from the cluster permanently, run the command:

```
cast cluster delete cloudant@<FQDN>
```

   For example:

```
cast cluster delete cloudant@db2.vm
```

Install Cloudant Local in a Single Node

A standalone installation is a single-node configured to run as a load balancer and a database node. Typically, a standalone node is used for development or demonstration purposes, so no additional configuration is needed beyond the defaults. The default admin password is `pass`.

1. Run the following command to install Cloudant Local as a single node:

```
cast system install -lb -db {install-dir}/repo
```

2. If you want to configure SSL on a single node instance, see “Configuring SSL” on page 26.

Open the Cloudant Dashboard

You can verify that Cloudant Local installed successfully by logging into the Cloudant Local Dashboard.

1. To access the Dashboard, enter the following URL in a web browser:

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

2. After authentication, the Databases page opens.

   If the installation was successful, you will see the following databases that were automatically created during installation:

   - _replicator
   - _users
   - metrics
   - metrics_app

   You have successfully installed Cloudant Local for zLinux.

Uninstall Cloudant Local

To uninstall Cloudant Local, follow these instructions.

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

```
cast system uninstall
yum -y erase cloudant-cast
```

2. Run this command to uninstall the installation wizard.

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

   For example:

```
/opt/cloudant/uninstall/uninstall.bin
```

   You have now uninstalled Cloudant Local for zLinux.

Cloudant Local hardware and software requirements

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

Cluster machine requirements

A minimum of five machines 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.
  - However, you can install Cloudant Local on as many other nodes as needed to scale for your business requirements.
- Load Balancers:
  - At least one load balancer must be installed on a separate machine from the machines that are used in the Cloudant cluster.
  - A second load balancer is highly recommended in case the primary load balancer fails. If a second load balancer is installed, you must install it on a separate machine to ensure uninterrupted service.

Supported platforms

You can install Cloudant Local on the following platforms:

- Debian-derived Linux distributions:
  - Debian 6.0.10
  - Ubuntu Server 12.04.4
- Red Hat-derived Linux distributions:
  - Red Hat Enterprise Linux Server 6.5
  - Community ENTerprise Operating System (CentOS) 6.5
  - Oracle Enterprise Linux Operating System (x86_64) 6.5
  - Linux on IBM System z - Red Hat Enterprise Linux 7.0

The nodes in the Cloudant cluster and the machine 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 machine.

Hardware requirements

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

Database Nodes

- The minimum requirements are 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 are 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 setups are as follows:

- For spinning disks: 4 x 600 GB 15k SAS drives in RAID 0, which provides about 2.2 TB of usable storage.
- For solid-state drives (SSDs): 2 x 800 GB SSDs in RAID 0, which provides about 1.5 TB of usable storage.
- An ext4 filesystem mounted with the noatime option is recommended for performance reasons.

Note: A storage area network (SAN) is not recommended. If you use a centralized place 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 are a quad core processor and 8 GB RAM, 1 TB local hard disk drive, and a 1-gigabit network.

Prerequisite packages

Cloudant Local requires some packages to be installed already. If your system does not have access to public repositories, you might need to download and install these prerequisite packages before starting to install Cloudant Local.

The list of prerequisite packages is provided in a text file, available in the root of the installation `.tar.gz` file. Several text files are provided; choose the file that matches the node type and operating system for your system:

Table 1. Files listing prerequisite packages

File name	Node type and operating system
`cloud_prereq_db_deb.txt`	Database node, on Debian
`cloud_prereq_lb_deb.txt`	Load Balancer node, on Debian
`cloud_prereq_db_ubu.txt`	Database node, on Ubuntu
`cloud_prereq_lb_ubu.txt`	Load Balancer node, on Ubuntu
`cloud_prereq_db_rh.txt`	Database node, on RedHat
`cloud_prereq_lb_rh.txt`	Load Balancer node, on RedHat
`cloud_prereq_db_oel.txt`	Database node, on Oracle Enterprise Linux
`cloud_prereq_lb_oel.txt`	Load Balancer node, on Oracle Enterprise Linux

Requirements for a single-node implementation

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

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

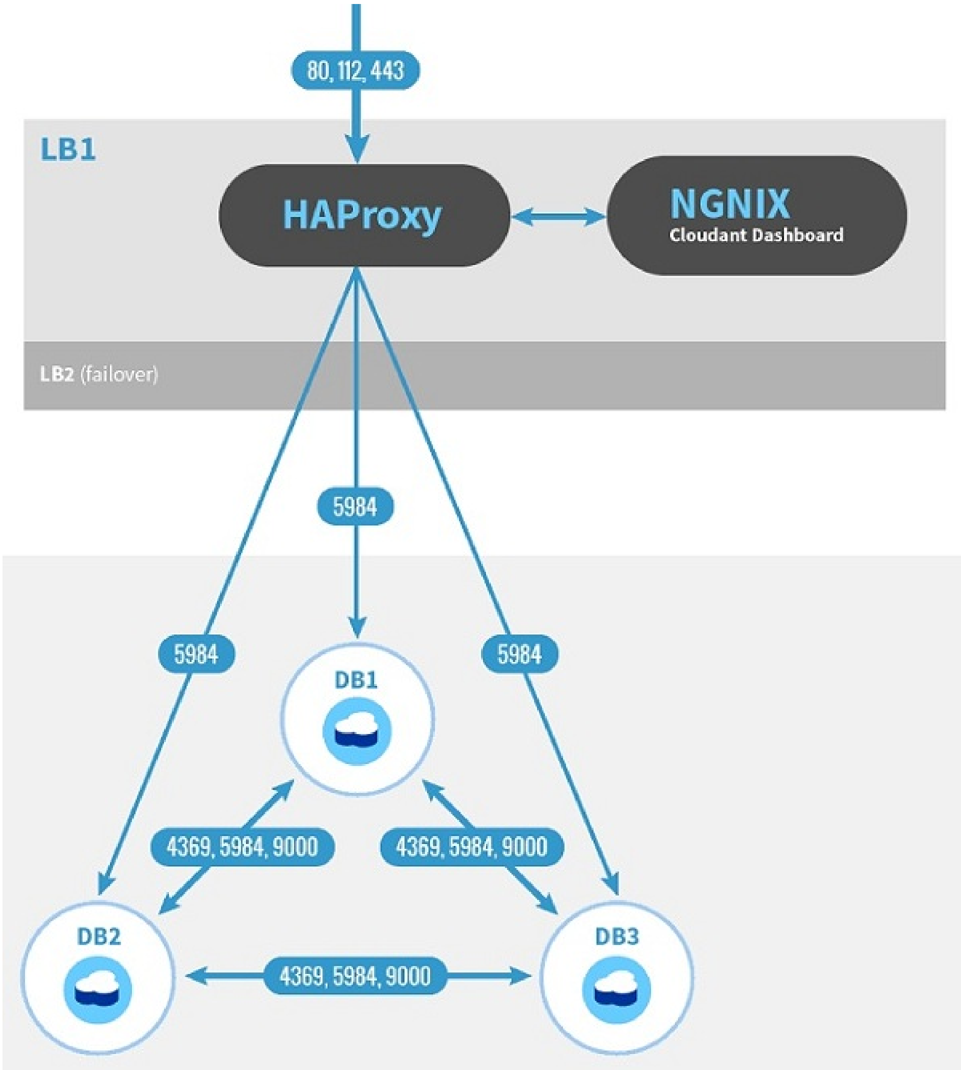
A single-node implementation is not recommended in a production environment. However, if you use a single-node implementation in a production environment, the requirements are the same as for a database node.

Cloudant Local architecture and ports

The following diagram shows the architecture of a Cloudant Local implementation that includes:

- three database nodes, which are identified as `DB1`, `DB2`, and `DB3`
- two load balancers, which 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.



Extract, Install, and Uninstall user requirements

Any user can extract the Cloudant Local packages, but you must be a root user or use the `sudo` command to install (or uninstall) Cloudant Local. If the root user or the `sudo` command is not used to do the extraction, the installation packages are extracted in the `(home)/Cloudant/repo` directory for that user, or the directory that was specified during the extract process. For more information, see "Extracting and Installing Cloudant Local" on page 13.

Moreover, you must use the same user ID that you used to install Cloudant Local to uninstall the product. For example, if you installed as root, you must uninstall as root. For more information about how to uninstall Cloudant Local, see "Uninstalling Cloudant Local" on page 53.

What to do next

After you confirm that your system meets all relevant requirements for implementing Cloudant Local, you can "Planning your Cloudant Local installation" on page 11.

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 and future storage space.

Note: In Red Hat and CentOS only, a Cloudant Local installation requires certain third-party packages. These packages are normally available if the Extra Packages for Enterprise Linux (EPEL) repository is installed. If EPEL access is not available, you must manually install the pre-requisite packages. Other alternatives include accessing the EPEL repository temporarily, or accessing the EPEL packages from an internal company-approved site. For more information about accessing EPEL, see the **`-E`** and **`-e`** options in "Extracting and Installing Cloudant Local" on page 13.

Having considered 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 and using Mount Points

The default extract location for Cloudant Local is `~/Cloudant`. However, this location is not the installation location for the product. It is simply a staging area where the Cloudant installation packages are placed until they are installed.

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`.

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, see the instructions in "Extracting and Installing Cloudant Local" 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 will be 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 host name 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 host name. Here is an example:

    `107.170.185.247 db2.millay-centos65-cloudant-local.com`.
  - Confirm that the correct host name is specified in the `/etc/sysconfig/network` file. Use the **hostname -f** command to confirm that the name is correct.
  - If the host name is correct, restart the machine for the new entry to take effect. If not, correct the invalid host name.
- 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`. Here is an 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 host name in the `/etc/hosts` file, not the fully qualified name. In the previous step, for example, the short-form host name is `db1`.
  - Run the **sudo hostname --file /etc/hostname** command to update the operating system host name, based on the name that is specified in the `/etc/hostname` file.
  - Confirm that the correct host name is specified with the **hostname -f** command. If the specified host name 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. For more information, see "Configuring Cloudant Local" on page 20.

Load balancer prerequisites

1. You must install and configure a load balancer on a separate machine from the ones that are used in the Cloudant cluster. A second (failover) load balancer is recommended to ensure uninterrupted service.

   The nodes in the Cloudant cluster and the machine 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 machine.
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.

   Note: If you indicate that you use two load balancers when you specify your installation properties in the `configure.ini` file, the `configure.sh` script will install **Keepalived** on your primary and secondary load balancers. **Keepalived** is a routing software that provides facilities for load balancing and high-availability to Linux system and Linux based infrastructures. For more information about the `configure.ini` file and the `configure.sh` script, see "Configuring Cloudant Local" on page 20.

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 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 there is a virtual IP address available for the load balancers to share. This is the IP address that clients will use to access the cluster. Record this IP address for future reference:
   - Virtual IP address

Note: You can configure your load balancers for either local logging or remote logging to a separate `syslog` logging server. For more information, see "Configuring Cloudant Local" on page 20.

What to do next

After you develop an installation plan and verify that your nodes and load balancers are properly configured, you can "Extracting and Installing Cloudant Local."

Extracting and Installing Cloudant Local

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

Important: If you want to install Cloudant Local on a mount point, follow the instructions in "Installing Cloudant Local on a mount point if the product is not installed" on page 14. If you do not want to install the product on a mount point, proceed as described in "Extract and Install Cloudant Local" on page 14. If Cloudant Local is already installed and you want to move the product to a mount point, follow the instructions in "Moving Cloudant Local to a mount point after the product is installed" on page 19.

Overview of Cloudant Local installation for a multi-node environment

Installing Cloudant Local on a multi-node environment requires you to perform the following steps:
1. Ensure that all nodes have the correct name, by using the **`hostname -f`** command.
2. Ensure that all nodes have a static IP address, by using the **`ifconfig`** command.
3. Modify the `configure.ini` file in the root directory of the installation media on the first node only.

If you have a `configure.ini` file from a previous release, use it instead of the new `configure.ini` file. See "Configuring Cloudant Local" on page 20 for more information.

As a best practice, before you run `quiet_install.sh`, modify the `configure.ini` (or use a `configure.ini` from version 1.0.0.2) and pass it as a parameter.

4. Run the `quiet_install.sh` script to install Cloudant Local on all nodes using the -c parameter and the `configure.ini` file you modified in the previous step.
5. Run the `~/Cloudant/repo/configure.sh` script on the first node to finish the configuration and join the nodes.

The remainder of these instructions provides specific details for installing Cloudant Local on an individual node. Remember you must follow the steps described in this overview to install Cloudant Local in a multi-node environment.

Installing Cloudant Local on a mount point if the product is not installed

If Cloudant Local is not currently 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`, you would use these 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`, you would use these commands to create the symbolic links:
   - **`ln -s /export/local/opt/cloudant /opt/cloudant/`**
   - **`ln -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, as described in "Extract and Install Cloudant Local."

Extract and Install Cloudant Local

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

Important: Any user can extract the Cloudant Local packages, but you must be a root user or use the `sudo` command to install Cloudant Local on a database or load balancer node. If the root user or the `sudo` command is not used to do the extraction, the installation packages are extracted in the `(home)/Cloudant/repo` directory for that user, or the directory that was specified during the extract process.

1. Extract the Cloudant Local installation packages or kit from the installation media:

- If you downloaded the installation media as a disk image (an `.iso` file) or you received physical media (such as a DVD), use the `quiet_install.sh` script in the root directory of the installation media to extract and install the Cloudant Local packages, as described in subsequent steps.
- If you downloaded the installation packages as a `tar` file, you must first expand the installation media:
  a. Create an empty directory to expand the Cloudant Local installation packages.
  b. Use this command to expand the installation media:

     tar –zxvf cloudant_linux_*pgk_v.v.v.v***_amd64_***yyyymmdd_hhmm***.tar.gz**

     The variables in the preceding command are as follows:
     - *pgk* is one of the following entries, based on your Linux operating system:
       - **rpm** for Red Hat-derived Linux distributions:
       - **deb** for Ubuntu or Debian
     - *v.v.v.v* is the Cloudant Local version number, such as 1.0.0.2.
     - *yyyymmdd* is the four-digit year, two-digit month, and two-digit day of the build, such as 20150117 for January 17, 2015.
     - *hhmm* is the two-digit hour and two-digit minute of the build, such as 0918 for 9:18.

2. Change to the directory in which you placed the Cloudant Local installation packages or kit. That directory includes the `quiet_install.sh` file that is used to install the product.
3. Use this command to display the syntax help text for the **`quiet_install.sh`** command:

   ```
   ./quiet_install.sh -h
   ```

   Here is the help text:

   ```
   Usage:
   quiet_install.sh [option] [download-folder]
   [-h]|[-v][-x[-o [-w download_folder_name]]]

   license  node-type [instance-type]
   -a -d|-l|-s  [-p|-n]

   [epel-option] [package-folder]
   [-E]|[-e epel-url] [-f folder_name]

   [configuration]
   [-c file [-m]]

   Where:
   option is one or more of these optional parameters:
   -h = Display this help information and exit
   -v = Run in verbose mode
   -x = Extract the packages but do not install them
   -o = Only download the Cloudant Local prerequisite packages
   (Requires -x)

   download-folder is this optional parameter-value pair:
   -w folder_name = Folder to download prerequisite packages (Requires -x -o)
   (Default is /root/Cloudant/download)

   license is this required option:
   -a = Accept the IBM License Agreement for Cloudant Local

   node-type is one of these required parameters:
   -d = Install a Database Node
   -l = Install a Load Balancer Node
   -s = Install a Single Node implementation

   instance-type is one of these optional parameters:
   -p = Install a Production instance (Default)
   -n = Install a Non-production instance
   ```

```
epel-option is one of these optional parameters for Red Hat-derived Linux distributions only:
-E = Activate the EPEL (Extra Packages for Enterprise Linux)
     repository only for the duration of this install:
     http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
-e epel_url = Activate the specified EPEL repository

package-folder is this optional parameter-value pair:
-f folder_name = Folder to extract packages
(Default is /root/Cloudant)

configuration is this optional parameter-value pair:
-c file = Upon a successful installation, immediately run
the configure.sh script with the specified
configure.ini file

-m = Leave Cloudant in maintenance mode (requires -c)
```

Before you run the **./quiet_install.sh** command, read all the following information about this command.

- You must specify at least the following two parameters with the **./quiet_install.sh** command:
  - The **-a** parameter to accept the IBM license agreement.
  - One of the node-type parameters: **-d**, **-l**, or **-s**.

  For example, to install Cloudant Local on a database node, you must specify at least these two parameters: **-a -d**.
- The **-c** option allows you to specify an alternate location for the **configure.ini** file. This is useful when you want to:
  - Install from a read-only ISO image, and cannot change the **configure.ini** file.
  - Keep or save the modified configure.ini file in a safe location.
  - Keep the installation image unchanged.

    When you specify the **-c** option for the quiet_install.sh script, the script automatically calls the configure.sh script once the install is completed. This allows you to install and configure in one command. If any error is detected during the install step, the configure step is not run.
- The option, instance-type, and package-folder parameters are optional.
- The default instance-type is **-p** for a production instance. Use this default value to install Cloudant Local in a production environment.

  If you want to install Cloudant Local in a non-production environment or test environment, you must specify the **-n** parameter.
- By default, the extract packages are placed in the Cloudant subdirectory in the home directory of the logged on user (that is, ~/Cloudant). This location is not the installation location for the product. It is simply a staging area where the Cloudant installation packages are placed until they are installed.

  If you want the extract packages to be put in another location, specify the **-f** parameter, followed by the fully qualified path to the folder in which to place the packages.
- Use the **-v** option only if you want to be prompted before certain prerequisite items are installed.

  Important: If you use this option, you must type y for Yes in response to every prompt. If you type n for No in response to any prompt, the product is not installed.
- [Red Hat-derived Linux distributions only] The **-E** option activates temporarily the EPEL repository: http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm The repository is activated only for the duration of the installation.

  Note: Before using this option, check for compliance with your organizational security requirements, and ensure that you are authorized to proceed.
- [Red Hat-derived Linux distributions only] If the default EPEL repository is not available or accessible at your installation, you can use an alternate repository by specifying the **-e** option with

a valid URL. This might be appropriate if the network used by your organization prevents you from accessing external repositories, but you are permitted to access the EPEL packages from an internal company-approved site. In that case, you can use the **-e** option to access an internal copy of the EPEL package by specifying **-e** with the appropriate internal URL.

- [Red Hat-derived Linux distributions only] If you do not specify one of the **-E** or **-e** options, the installation assumes you have already manually installed the EPEL pre-requisite packages. If you have not previously installed the pre-requisites, an error message is displayed, similar to the following example:

```
[root@examplehost exampledir]# ./quiet_install.sh -a -s
Neither -e or -E was specified, the EPEL repository will not be activated.
Loaded plugins: fastestmirror
Cleaning repos: base extras updates
Cleaning up Everything
Cleaning up list of fastest mirrors
Package rsyslog-5.8.10-10.el6_6.x86_64 already installed and latest version
Package libicu-4.2.1-9.1.el6_2.x86_64 already installed and latest version
Package python-2.6.6-52.el6.x86_64 already installed and latest version
Package curl-7.19.7-40.el6_6.4.x86_64 already installed and latest version
Package rsync-3.0.6-12.el6.x86_64 already installed and latest version
Error: Nothing to do
Install for Python Pip libraries failed or was canceled by user.
Function:installDBRPMS
Command: yum -y -q install  python-pip
Package: python-pip
RC:      1
This package is usually found in an EPEL (Extra Packages for Enterprise Linux) repository:
You can temporarily activate the default EPEL repository by using the '-E' option,
provide an alternate EPEL repository by using the '-e epel_url' option, or
manually install the package before continuing.
```

Tip:

If you receive an error message during installation of the Cloudant Local packages, you do not need to rerun **quiet_install.sh**. Instead, use the **install.sh** command to restart the installation process in verbose mode, or use the **install.sh -q** command to restart in quiet mode. Restarting with these commands eliminates the need to repeat steps that were successfully completed. Use **install.sh -h** to display the help text for the installation command. The script is in the directory in which you placed the Cloudant Local installation packages. The only time that you need to rerun **quiet_install.sh** is if an error occurs when the Cloudant Local packages are being extracted before installation.

The **install.sh** command also works with the **-E** or **-e** options, if you need to restart the installation process with access to an EPEL repository, as shown in the following example:

```
[root@examplehost exampledir]# /root/Cloudant/repo/install.sh -q -E
Loaded plugins: fastestmirror
Cleaning repos: base extras updates
Cleaning up Everything
Cleaning up list of fastest mirrors
Loaded plugins: fastestmirror
Cleaning repos: base extras updates
Cleaning up Everything
Retrieving http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
warning: /var/tmp/rpm-tmp.fef06V: Header V3 RSA/SHA256 Signature, key ID 0608b895: NOKEY
Preparing...                ########################################### [100%]
   1:epel-release           ########################################### [100%]
Loaded plugins: fastestmirror
Cleaning repos: base epel extras updates
Cleaning up Everything

To complete the installation using an EPEL repository, run one of these forms of the command:
/root/Cloudant/repo/install.sh -q -E
/root/Cloudant/repo/install.sh -q -e epel_url
```

```
To complete the installation without using an EPEL repository, run the command.
/root/Cloudant/repo/install.sh -q.
Exiting
[root@examplehost exampledir]#
```

4. Specify one of the following options with the **quiet_install.sh** command to install the product:
   - If you want to install Cloudant Local on a database node in a multi-node cluster, use this command:

     ./quiet_install.sh -a -d
   - If you want to install Cloudant Local on a load balancer in a multi-node cluster, use this command:

     ./quiet_install.sh -a -l

     A second (failover) load balancer is recommended to ensure uninterrupted service in a multi-node cluster.
   - If you want to install both a database node and a load balancer on a single node, use this command:

     ./quiet_install.sh -a -s

     A single-node implementation can be used for trial testing or development.
   - If you want to extract the Cloudant Local packages and install the product later, include the **-x** option. For example, to extract the packages and scripts for a database node only, you would use the command: **./quiet_install.sh -x -a -d**

     When you are ready to install the product, use the command: **~/Cloudant/repo/install.sh -q**. You can display the help text for `install.sh` with this command: **~/Cloudant/repo/install.sh -h**.

Note: If you need to uninstall Cloudant Local, follow the instructions in "Uninstalling Cloudant Local" on page 53.

Install Cloudant Local

It is only necessary to use the `install.sh` command in the following instances:
- The `quiet_install.sh` fails to install all of the packages.
- You specify `-x` to extract the packages instead of installing them.
- You do not run `quiet_install.sh`.

When you are ready to install the product, use the **install.sh**command as discussed in the following instructions.
1. Change to the directory where you placed the Cloudant Local installation packages or kit. That directory includes the **install.sh** file that is used to install the product.
2. Use this command to display the syntax help text for the **install.sh** command:

   ```
   ./install.sh -h
   ```

   Here is the help text:

   ```
   Usage:
   install.sh [option] [download-folder] [epel-option]
   [-h]|[-q][-o [-w download_folder_name]] [-E]|[-e epel_url]

   Where:
   option is one or more of these optional parameters:
   -h = Display this help information and exit
   -q = Run the install in quiet mode to suppress most messages
   -o = Only download the Cloudant Local prerequisite packages

   download-folder is this optional parameter-value pair:
   -w folder_name = Folder to download prerequisite packages (Requires -o)
   (Default is /Users/robs/Cloudant/download)

   epel-option is one of these optional parameters for Red Hat-derived Linux distributions only:
   ```

```
-E = Activate the EPEL (Extra Packages for Enterprise Linux)
repository only for the duration of this install:
http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
-e epel_url = Activate the specified EPEL repository
```

Note: The parameters for the `install.sh` script are also included in the `quiet_install.sh` script. If a parameter exists in the `install.sh` script and in the `quiet_install.sh` script, the parameter has the same meaning in both scripts. The `quiet_install.sh` script calls the `install.sh` script and passes the parameters to it.

Moving Cloudant Local to a mount point after the product is installed

If Cloudant Local is already installed on the root partition of your system, follow these instructions to move the Cloudant software and database to one or two mount points.

1. You can stop or kill `cloudant`, `clouseau`, and `cloudant-local-metrics` by running the following command:

   `opt/Cloudant/sbin/cloudant.sh –`*`k`*

   The `–k` option at the end of the command means kill or stop the processes.
2. Create the following Cloudant directories on ats 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`, you would use these 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.
3. Move your previously installed `/opt/cloudant` and `/srv/cloudant` directories to the appropriate directories on your mount points.

   This step moves each directory from the root partition on your system to the appropriate directory on your mount point. For example, if your mount point for both directories is `/export/local`, you would use these commands to move the files:
   - **`mv /opt/cloudant /export/local/opt/cloudant`**
   - **`mv /srv/cloudant /export/local/srv/cloudant`**

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

   For example, if your mount point for both directories is `/export/local`, you would use these commands to create the symbolic links:
   - **`ln -s /export/local/opt/cloudant /opt/cloudant/`**
   - **`ln -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.
5. You can start `cloudant`, `clouseau`, and `cloudant-local-metrics` by running the following command:

   `opt/Cloudant/sbin/cloudant.sh –`*`s`*

   The `–s` option at the end of the command means start the processes.

What to do next

After you extract and install Cloudant Local on each database and load balancer node in your implementation, you must configure Cloudant Local.

Configuring Cloudant Local

Use these instructions to configure Cloudant Local on a database or load balancer node in either a multi-node cluster or a single-node implementation. You must configure Cloudant Local on each database and load balancer node in your implementation.

Modify the `configure.ini` file

If you supplied the `configure.ini` file as part of the `-c` parameter to the `quiet_install.sh` script, your nodes are already configured as part of the installation step. To complete the installation, rerun the `configure.sh` script on the first database node.

You must edit the `configure.ini` file to specify the installation properties for your implementation, such as the host names and IP addresses for your database nodes. The default directory for the `configure.ini` file is `~/Cloudant/repo`, unless you specified another directory during the extract process.

Follow the instructions in the `configure.ini` file to specify the appropriate values for your implementation. The file includes instructions and examples for each installation variable you can specify in a multi-node or single-node implementation.

After you identify your installation properties in the `configure.ini` file, you can "Run the configure.sh script to configure your nodes." The `configure.sh` script uses your installation properties to configure:

- a single-node implementation where both a Cloudant database and a load balancer are installed on a single node, or
- all of the database and load balancer nodes in a multi-node cluster.

Important: After you configure your first node, copy the `configure.ini` file to each database and load balancer node in your cluster. You must copy the file after you configure your first node for two reasons:

- The `configure.ini` file is modified by the `configure.sh` script.
- The same modified file must be used to configure every other node in your cluster.

Run the `configure.sh` script to configure your nodes

Follow these instructions to use the `configure.sh` script to configure your database and load balancer nodes, based on the installation properties you specified in the `configure.ini` file. If needed, you can rerun this script at any time to update your nodes to reflect any changes in your installation properties in the `configure.ini` file.

Note: The `configure.sh` script configures your system firewall to allow communications on ports 4369, 5984, and 9000 on each database node in your cluster. The script also enables access to ports 80, 112, and 443 on each load balancer. The `configure.sh` script enables these ports based on the port setting in the `enableports.sh` script. If you want to change the default port number, see "Changing the default port number" on page 22.

1. Change to the directory in which the `configure.sh` script is installed. The default directory is `~/Cloudant/repo`, unless you specified another directory during the extract process.
2. Use this command to display the syntax help text for the `configure.sh` command:

   `./configure.sh -h`

   Here is the help text:

```
Usage:
configure.sh  [option]  [command]
              [-h | -q] [-D] [-m] [-c file]

Where:
 option is one of these optional parameters:
  -h       = Display this help information and exit
  -q       = Run in quiet mode

 command is this optional parameter:
  -D       = Delete the Cloudant databases
  -m       = Leave Cloudant in maintenance mode

 configuration is this optional parameter-value pair:
 -c file  = Use the specified configure.ini file
```

Note: When upgrading Cloudant Local from version 1.0.0.2 to a newer release, it is necessary to use maintenance mode. This is applied automatically when required by the `remove.sh` and `configure.sh` scripts. They invoke the `cloudant.sh` script with the required `-m` or `-w` parameters.

The `-c` option allows you to specify an alternate location for the `configure.ini` file. This is useful when you want to:

- Install from a read-only ISO image, and cannot change the `configure.ini` file.
- Keep or save the modified `configure.ini` file in a safe location.
- Keep the installation image unchanged.

3. Change to the directory in which the `enableports.sh` script is installed. The default directory is `~/Cloudant/repo`, unless you specified another directory during the extract process.
4. Use this command to display the syntax help text for the **`enableports.sh`** command:

   `./enableports.sh -h`

   Here is the help text:

   ```
   Usage:
   enableports.sh  [option]  [command]
   [-h | -q]

   Where:
   option is one or more of these optional parameters:
   -h = Display this help information and exit
   -q = Run in quiet mode to suppress prompts
   ```

   If another script calls the `enableports.sh` script, the calling script either specifies or implies the `-q` parameter. For example, when the `quiet_install.sh` script calls the `enableports.sh` script, the `-q` parameter is implied.
5. Specify one of the following options with the **`configure.sh`** command:
   - If you want to configure your database and load balancer nodes in manual mode, use this command:

     `./configure.sh`

     Important: Use manual mode to configure your first node. The reason is to verify the installation properties in your `configure.ini` file. The prompts allow you to review your installation properties and identify any errors that might result in an incorrect configuration. If you see an error in a prompt, exit the configuration process as described on the prompt, and then fix the error in the `configure.ini` file. After the error is corrected, repeat steps 2-4.

     After you configure your first node, you can use the next command to configure your other nodes in quiet mode, or you can continue to use manual mode.
   - If you want to configure your database and load balancer nodes in quiet mode, use this command:

     `./configure.sh -q`

In quiet mode, prompts are not displayed during configuration processing, so you do not have an opportunity to review the installation properties in your `configure.ini` file before your nodes are configured. Therefore, quiet mode is not recommended to configure your first node, although you can use it to configure the other database and load balancer nodes in your cluster.

- If you want to delete the Cloudant Local databases from your system, use this command:

  ```
  ./configure.sh -D
  ```

  Note: Use this command with extreme caution because it deletes all of your Cloudant Local databases.

6. Repeat these steps on each database and load balancer node in your implementation.

   Remember: After you configure your first node, copy the `configure.ini` file to each database and load balancer node in your cluster.

7. Once the installation and configuration is completed on all nodes, re-run `configure.sh` on the first database node. This step joins all the nodes to create a cluster of the database nodes. It also creates the metrics database that is utilized by the Metrics application.

Changing the default port number

If you elect to change the default ports configured in the `enableports.sh` script, you must make the changes described in this section.

Note: You must perform each change on every node in your cluster. Cloudant Local must be stopped on all the nodes while the changes are being performed.
The commands to stop and start Cloudant Local using the `cloudant.sh` script are described in "Managing Cloudant Local services" on page 25.

1. Change the default port number for HTTP communication.
   - Edit the file `/opt/cloudant/etc/local.ini` to include a chttpd section where port is set to the new port number.
   - For example, assume the new port number for HTTP communication is 12345, and edit the file to include the following parameters:

     ```
     [chttpd]
     port = 12345
     ```

2. Change the default port number for Erlang node-to-node communication.
   - Edit the file `/opt/cloudant/etc/vm.args` to set the `inet_dist_listen_min` and `inet_dist_listen_max` values to the new port number.
   - For example, assume these values are currently set to 9000, and the new port number for Erlang node-to-node communication is 9999. Edit the file to replace the existing port number:

     ```
     -kernel inet_dist_listen_min 9000
                         -kernel inet_dist_listen_max 9000
     ```

     with the new port number:

     ```
     -kernel inet_dist_listen_min 9999
                         -kernel inet_dist_listen_max 9999
     ```

3. Change the port number for the Erlang Port Mapper Daemon (epmd).
   a. Edit the following files to include the command to set the `ERL_EPMD_PORT` environment variable to the new port number for the epmd:

      ```
      /etc/sv/cloudant/run
                              /etc/sv/clouseau/run
                              /opt/cloudant/bin/erl_call_cloudant
      ```

      For example, assume the new port number for the epmd is 4370, and add this line to the end of each of the files mentioned above:

      ```
      ERL_EPMD_PORT=4370
      ```

See the following example:

```
/etc/sv/cloudant/run ERL_EPMD_PORT=4370
                                /etc/sv/clouseau/run ERL_EPMD_PORT=4370
                                /opt/cloudant/bin/erl_call_cloudant ERL_EPMD_PORT=4370
```

b. In the file /opt/cloudant/etc/clouseau.policy, change the port number in the line beginning permission java.net.SocketPermission to the new port number for the epmd.

   For example, assume the current port number for the epmd is 4369 and the new port number is 4370, and change the line:

```
permission java.net.SocketPermission "127.0.0.1:4369", "connect,resolve"
```

   to:

```
permission java.net.SocketPermission "127.0.0.1:4370", "connect,resolve"
```

c. Run **pkill epmd** to stop the epmd process. Next time you start Cloudant Local, the epmd process will start automatically. If you change the port number for the epmd, you must set the environment variable ERL_EPMD_PORT to the new port number before you run **weatherreport** or **remsh**. Otherwise, the commands will fail to run.

Configuring remote and local logging

You can configure where log information is stored.

The nodes in your Cloudant Local cluster can be configured for either local logging, or remote logging to a separate **syslog** logging server. By default, your installation configures local logging for all nodes. If you want to keep local logging, you do not need to make any other logging configuration changes.

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 and might require much local storage space. Remote logging also makes it easier to combine logs from several nodes.

For details about setting up remote logging for nodes within the cluster, including log locations and configuration, see "Configuring logging" on page 43.

Launch Cloudant Local Dashboard

After you install and configure the database and load balancer nodes, you can check that the installation succeeded by launching the Cloudant Local Dashboard on each load balancer. Do this by:

1. Checking that the dashboard is running on the load balancer node.
2. Visting the dashboard URL using a web browser.

You can confirm that the dashboard is running by checking its status on the node. Do this using the command:

```
sudo sv status nginx
```

You can start the dashboard using the command:

```
sudo sv start nginx
```

Should you need to do so, you can stop the dashboard using the command:

```
sudo sv stop nginx
```

To access the Dashboard, enter the following URL in a web browser:

```
http://<loadbalancer.domain.com>/dashboard.html
```

In this URL, `<loadbalancer.domain.com>` is the fully qualified hostname of your load balancer.

You must 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: "0.1.0-local",
    vendor: {
        name: "Cloudant, an IBM Company"
    }
}
```

When you see the login display for Cloudant Dashboard, enter the database administration credentials that were configured in the `configure.ini` file.

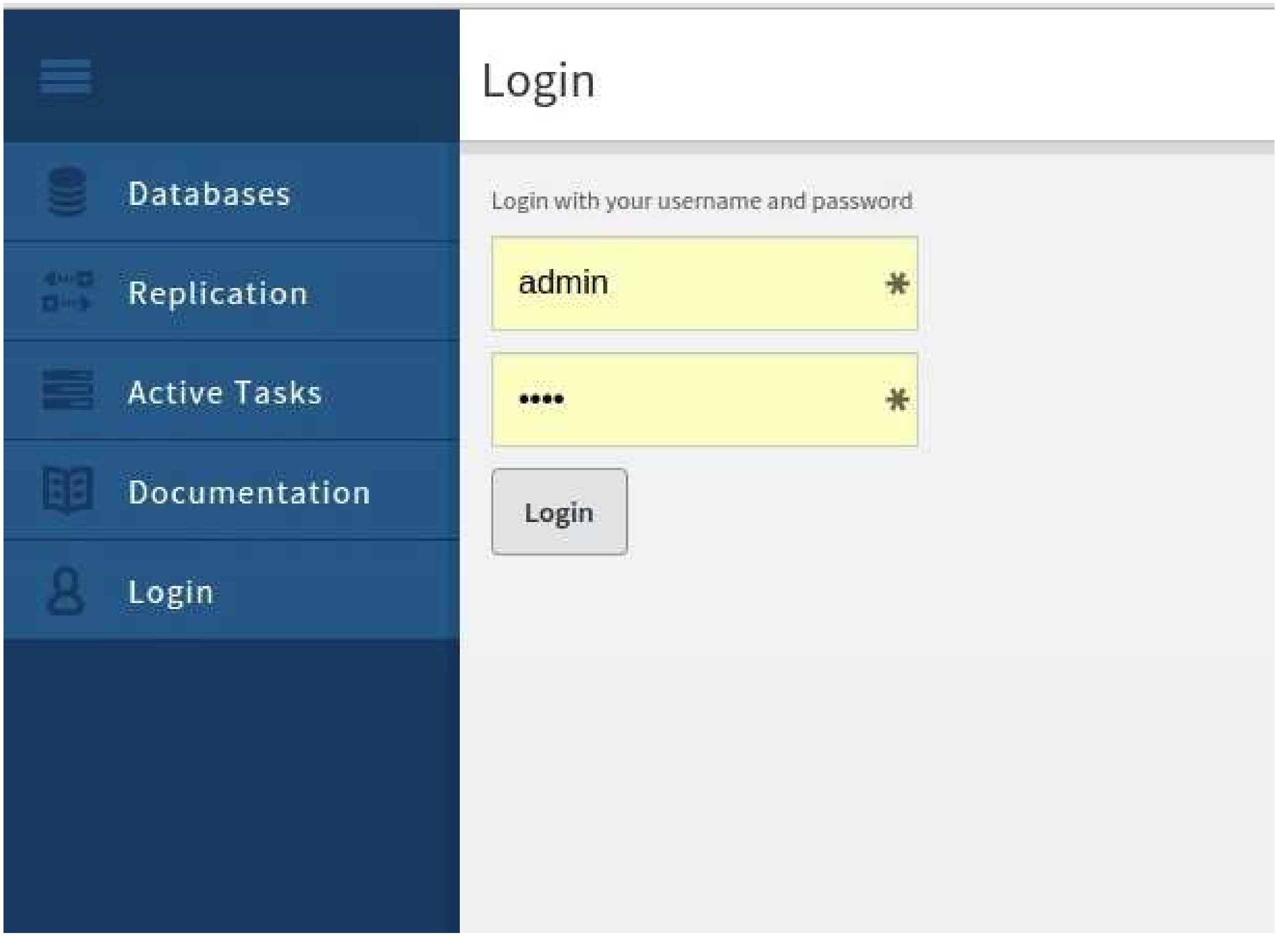


Figure 1. Login display for Cloudant Dashboard

What to do next

After you configure Cloudant Local, do the following steps:

- Configure SSL communication, as described in "Configuring SSL" on page 26. If you access the Cloudant database across an untrusted network, configuring SSL/HTTPS is advisable. If your network is trusted, configuring SSL/HTTPS might be considered optional so that you could use HTTP requests instead of HTTPS.

- Configure database-level security, as described in "Configuring database-level security" on page 49. Cloudant Local uses the CouchDB security model for database-level security. You should set up database-level security on your databases.
- View cluster health, as described in "Determining cluster health with the Metrics application" on page 35. The Cloudant Metrics web application displays statistics and data about the health of your Cloudant Local cluster. Use the Metrics application to identify potential problems in your Cloudant cluster.
- Review the information about various common maintenance activities and troubleshooting common issues, as described in "Cloudant Local maintenance tasks" on page 55.

Managing Cloudant Local services

Use these instructions to stop, start, or restart Cloudant Local with the `cloudant.sh` script. You also can use the script to display the status of various processes and to test the Cloudant Local API.

Follow these instructions to use the `cloudant.sh` script.

1. Change to the directory where the `cloudant.sh` script is installed. The default directory is `~/Cloudant/repo`, unless you specified another directory during the extract process. (The script is also placed in the `/opt/cloudant/sbin` directory.)
2. Use this command to display the syntax help text for the `cloudant.sh` command:

   `./cloudant.sh -h`

   Here is the help text:

   ```
   Usage:
   cloudant.sh option
   -h | -d | -t | -u | -k | -v | -r | -s | -m | -w

   Where:
   option is one of these parameters:
   -h = Display this help information and exit
   -d = Display the Cloudant process IDs
   -t = Test the Cloudant URLs
   -u = Determine if Cloudant is running
   -v = Display the Cloudant version
   -k = Kill (stop) all Cloudant services
   -r = Restart all Cloudant services
   -s = Start all Cloudant services
   -m = Put Cloudant in maintenace mode
   -w = Wake Cloudant from maintenace mode
   ```

   Prior to running certain commands, the **`cloudant.sh`** script validates that the appropriate tag files were saved by the **`install.sh`** script. If the tag files are not found, an error occurs.
3. Specify one of the following options with the `cloudant.sh` command:
   - If you want to start all Cloudant Local services, use this command:

     `./cloudant.sh -s`

     Use this command to start these services: `cloudant`, `clouseau`, and `cloudant-local-metrics`.
   - If you want to "kill" or stop all Cloudant Local services, use this command:

     `./cloudant.sh -k`

     Use this command to stop these services: `cloudant`, `clouseau`, and `cloudant-local-metrics`.
   - If you want to restart all Cloudant Local services, use this command:

     `./cloudant.sh -r`

     Use this command to restart your services after you change a configuration file.
   - If you want to display the status of the processes that are used by your Cloudant Local services, use this command:

     `./cloudant.sh -d`

The output from this command lists the processes, such as `beam.smp`, which is the primary Cloudant DB service.

- If you want to test the Cloudant Local API, use this command:

  ./cloudant.sh -t

  Use this command to confirm that Cloudant Local is properly configured and responds to a basic API request.

Note: When upgrading Cloudant Local from version 1.0.0.2 to a newer release, it is necessary to use maintenance mode. This is applied automatically when required by the `remove.sh` and `configure.sh` scripts. They invoke the `cloudant.sh` script with the required `-m` or `-w` parameters.

Configuring SSL

Use these instructions to configure Secure Sockets Layer (SSL) for your Cloudant Local installation. If you access the Cloudant database across an untrusted network, configuring SSL/HTTPS is recommended. 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"
- "Generate a CSR" on page 27
- "Generate a self-signed certificate" on page 27, if needed
- "Combine the RSA certificate and key" on page 28
- "Secure the RSA key and certificate" on page 28
- "Configure HAProxy for SSL connections" on page 28
- "Validate your SSL connection" on page 29
- "Display and confirm your untrusted certificate" on page 29, if needed
- "View the Load Balancer" on page 31
- "Connect load balancer and database nodes using SSL" on page 32

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
```

Here is an example of 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 usually is sent to a Certificate Authority (CA), who then issues 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.

Here is an example of 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 will be incorporated
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
For some fields there will be a default value,
If you enter '.', the field will be 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

Please 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 which is suitable for testing:

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

Here is an example of 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(s) 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 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. Do this by using the following commands:

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

Here is an example of 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 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, using the https protocol. Here is an example of how to connect to a load balancer with the https protocol:

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

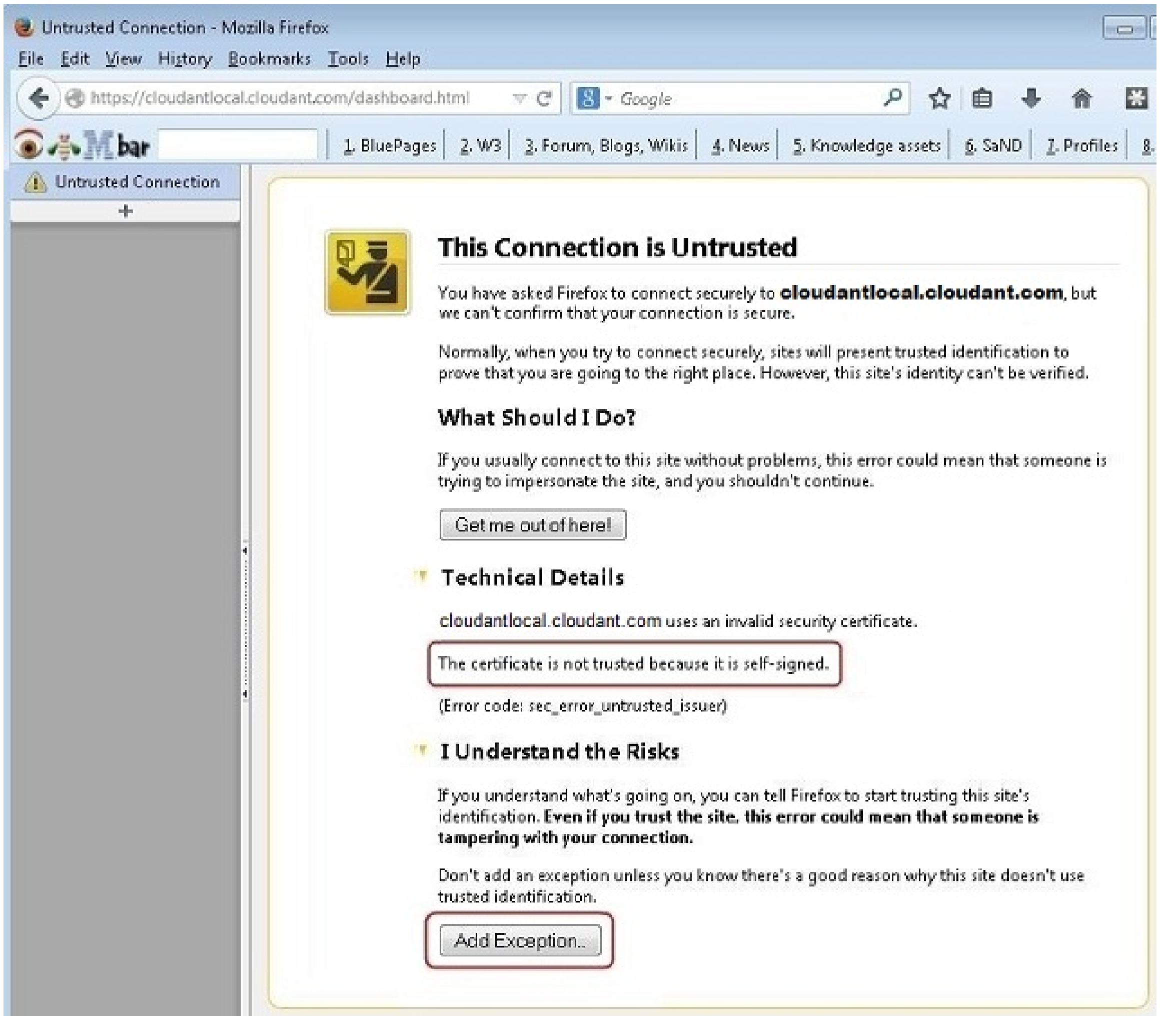
If your SSL connection is working correctly, the Cloudant Local dashboard appears in your browser.

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

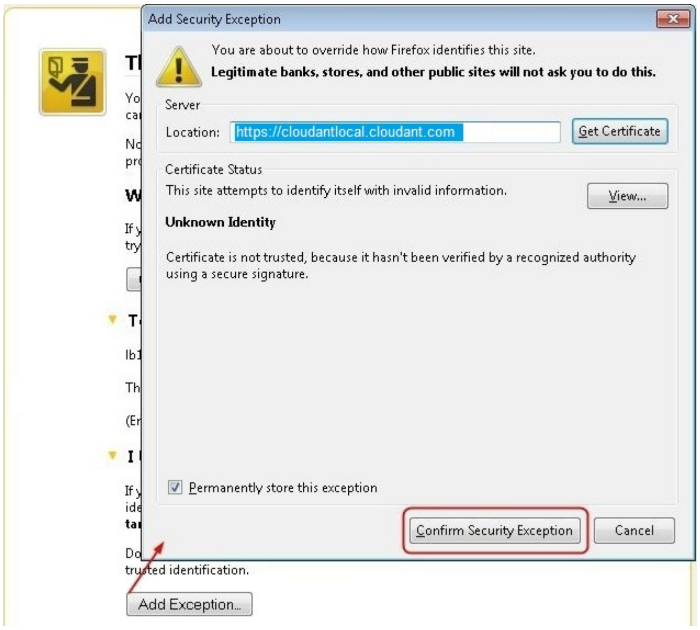
If you use a certificate that was validated by a Certificate Authority, this step is not needed.

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



Do the following 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 will be stored for future use.
   - Click **Confirm Security Exception** to confirm the exception.



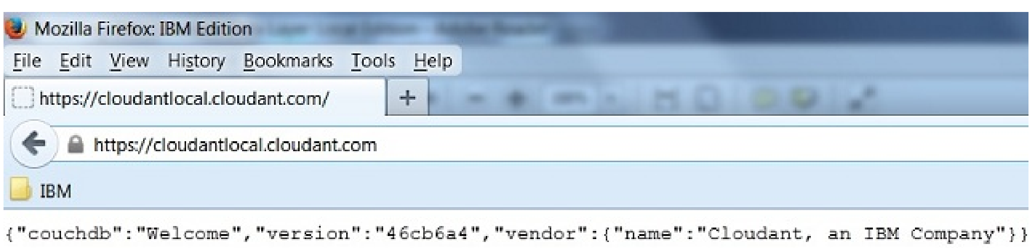
Note: Every user who accesses the load balancer from a separate machine must accept and confirm the self-signed certificate.

View the Load Balancer

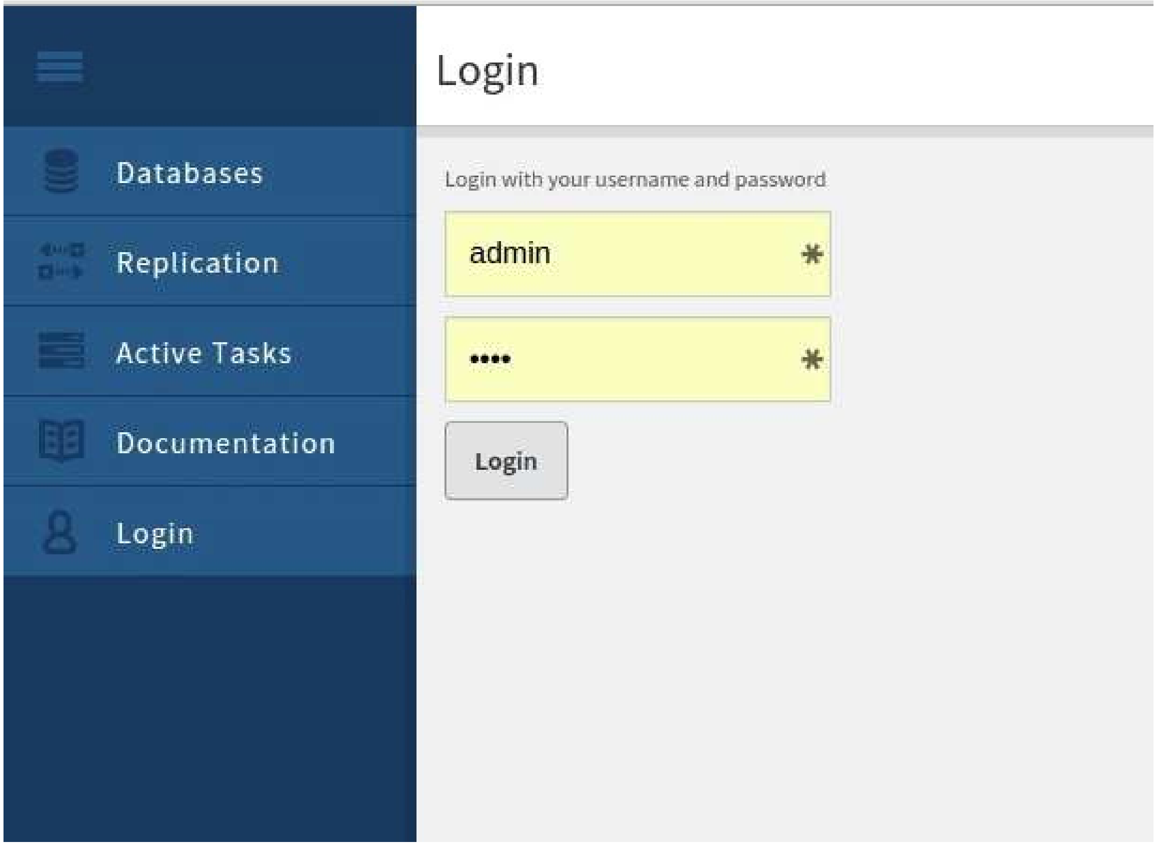
Check that you can view the Load Balancer correctly.

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. An example in which the user specified `dashboard.html` as part of the URL is as follows:



Connect load balancer and database nodes 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 have chosen to 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 on creating the `ca.pem` file, see "Generate the certificate authority file for a database node" on page 33.
2. In the `haproxy.cfg` file, make the following changes:
   a. Find the section labelled as follows:
   ```
   ######
   # NOTE: Specify the appropriate host names 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 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"
- "Enable SSL security on the database node" on page 34

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 provided by a third party, you can proceed directly to "Copy the SSL security files to the database node."

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 machine.

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 using SSL" on page 32.

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."

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 may 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 pre-requisite security certificate files must be available on the database node.

Copy the following files to any convenient location within the filesystem 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 33 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 33, 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. Do this by performing the following steps:

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, as follows:
```
[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 generic information on configuring Cloudant Local for SSL-based secure connections, see the information on Secure Socket Level Options.

Connect load balancer and database nodes 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 have chosen to 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 on creating the ca.pem file, see "Generate the certificate authority file for a database node" on page 33.
2. In the haproxy.cfg file, make the following changes:
   a. Find the section labelled as follows:
   ```
   ######
   # NOTE: Specify the appropriate host names 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**.

Enabling SSL inter-node encryption

Use these instructions to enable Secure Sockets Layer (SSL) encryption between database nodes in your cluster.

Follow these steps as the root user, for each node, to enable SSL inter-node encryption:

1. Check that you have a valid Certificate Authority (CA) certificate: `ca.pem` For more information about this, see "Generate the certificate authority file for a database node" on page 33.
2. Check that you have a valid certificate files for each database node: `serverX.key` and `serverX.pem` For more information about this, see "Generate the server certificate file for a database node" on page 33.
3. Copy the CA certificate file (`ca.pem`), and the respective database node certificate file (`serverX.pem`) and database key file (`serverX.key`) to the corresponding database node.
4. Edit the `/opt/cloudant/etc/vm.args` file on each database node, adding the following lines:
```
# SSL config
-proto_dist inet_tls
-ssl_dist_opt server_certfile "/path/to/serverX.pem"
-ssl_dist_opt server_keyfile "/path/to/serverX.key"
-ssl_dist_opt server_cacertfile "/path/to/ca.pem"
-ssl_dist_opt client_cacertfile "/path/to/ca.pem"
-ssl_dist_opt server_secure_renegotiate true client_secure_renegotiate true
```
5. Save and close the `/opt/cloudant/etc/vm.args` file.
6. Edit the `/opt/cloudant/bin/remsh` file on each database node, adding the following code to the end of the last line:
```
-proto_dist inet_tls -ssl_dist_opt client_cacertfile "/path/to/ca.pem"
```
Note: If you prefer, simply add a '\' character to the end of the last line, then paste the required code as the new final line of the file.
7. Save and close the `/opt/cloudant/bin/remsh` file.
8. Restart Cloudant by running the following command:
```
sv restart cloudant
```

Once all the nodes have restarted, check that the cluster is running as expected.

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.

Modify the metrics.ini file

Configuration of the `metrics.ini` file is done during configuration of Cloudant Local, as described in "Configuring Cloudant Local" on page 20. However, you can update certain information in the `metrics.ini` file, if needed.

Here is an example of the `metrics.ini` file, which is in the `/opt/cloudant/etc` directory.

```
[dbcore]
USERID: admin
PASSWORD: password

[metricsDB]
URL: http://localhost:5984 ;<protocol>://<host>:<port>
METRICS_DBNAME: metrics ;Only lowercase letters (a-z), digits (0-9), and any of
                         the characters _, $, (, ), +, -, and / are allowed. The
                         database name must begin with a letter.
```

```
USERID: admin
PASSWORD: password
METRICS_INTERVAL: 1     ;Metrics collection interval. Default: 1 minute.
                         Valid values are from 1 to 60 minutes

[logging_syslog]
LOGGER_HOST: localhost
LOGGER_PORT: 514
LOGGER_FACILITY: local3
LOGGER_LEVEL: INFO ;Valid values are INFO, DEBUG, ERROR, CRITICAL, WARNING
```

To edit this file, open the `metrics.ini` file and update the following information on each database node in your cluster:

- Update the `USERID` and `PASSWORD` in the **[dbcore]** section of the file with the appropriate credentials for your installation.

  Note: The `USERID` and `PASSWORD` in the **[dbcore]** section of the file must match the user ID and unencrypted password of the Cloudant administrator in the `local.ini` file. Do not use the encrypted password. The `local.ini` file is in `/opt/cloudant/etc/local.ini`.
- Update the `URL` and `METRICS_DBNAME` in the **[metricsDB]** section to identify your target database.
  - For `URL`, specify the URL to the location in which you want the metrics database created. Your target database can be on the local Cloudant database or a remote Cloudant database.
  - For `METRICS_DBNAME`, specify the name for your metrics database. The default name is `metrics`. If the name is changed, the database must be created in advance by the user on the target database that is specified in the URL. If you change that default value, you must specify your replacement entry in the URL that is used to display the Metrics application. For more information, see "Displaying the Metrics application" on page 37.
- Update the `USERID` and `PASSWORD` in the **[metricsDB]** section with the appropriate credentials for your installation.
- Update the `METRICS_INTERVAL` in the **[metricsDB]** section to an appropriate value. The metrics application polls at specified intervals, and collects statistics from the dbnode. This interval is specified in minutes as the `METRICS_INTERVAL` value. By default, the collection interval time is 1 minute.

  The `USERID` and `PASSWORD` in the **[metricsDB]** section of the file must match the user ID and unencrypted password of the Cloudant administrator for the cluster that holds the metrics database. Do not use the encrypted password. The `local.ini` file is in `/opt/cloudant/etc/local.ini`. Here are two examples:
  - If the metrics DB is to be stored in the same cluster as the current machine, the `USERID` and `PASSWORD` in the **[metricsDB]** section must match the `USERID` and `PASSWORD` in the **[dbcore]** section of the `metrics.ini` file.
  - If the metrics DB is to be stored in a remote cluster, such as on `cloudant.com`, the `USERID` and `PASSWORD` in the **[metricsDB]** section must match the user ID and password of your `cloudant.com` account.
- If needed, update the logging information in the **[logging_syslog]** section:
  - For `LOGGER_HOST`, specify the remote logger host name for remote logging, or use the default value of `localhost` for local logging.
  - For `LOGGER_PORT`, specify the remote logger port number for remote logging, or use the default value of port 514 for local logging.
  - For `LOGGER_FACILITY`, specify the logging facility to which the log is written.
  - For `LOGGER_LEVEL`, specify the logging level for which you want to receive messages, such as `INFO`, `WARNING`, `ERROR`, or `CRITICAL`.

    For more information about logging, including log locations and configuration, see "Configuring logging" on page 43.

Important: The information that is entered in the `metrics.ini` file must be the same on every database node in your cluster.

Starting, stopping, and restarting the Metrics service

After you update the `metrics.ini` file, you must run the following command to restart the `cloudant-local-metrics` service on all the database nodes in the cluster:

```
sudo sv restart cloudant-local-metrics
```

To start and stop the Metrics application, use these commands:

```
sudo sv start cloudant-local-metrics
```

```
sudo sv stop cloudant-local-metrics
```

Displaying the Metrics application

Here is the default URL for the Metrics web application, including variables:

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

Replace the following variables in the preceding URL:

- Replace **`<protocol>`** with `http` or `https` if you configured Cloudant Local to use Secure Sockets Layer (SSL). For more information about SSL, see "Configuring SSL" on page 26.
- Replace **`<loadbalancer.company.com>`** with your load balancer host name, 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.

Here is an example of the Metrics web application. In this example, the user specified **`https://cloudantlocal.cloudant.com/metrics_app/statistics/index.html`** to display the Metrics application. If you have not logged 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.

Glossary of Metrics terminology

Refer to this glossary for descriptions of the various metrics that are displayed on the Cloudant Metrics web application.

The glossary describes the metrics from an operational point of view and identifies common issues that you can recognize with these metrics.

Unless otherwise noted, frequency measurements, such as the number of requests or the number of messages, are given per minute.

HTTP metrics

HTTPD Request Methods

These request methods display the number of requests in a given category, such as the number of `GET`, `PUT`, or `POST` requests. The total number of requests is also available.

COPY Displays the number of `HTTP COPY` requests. These requests are used to copy database documents.

DELETE
Displays the number of `HTTP DELETE` requests. These requests are used to delete documents, databases, or indexes.

GET Displays the number of `HTTP GET` requests. `GET` requests are used to retrieve information from the database without modifying it. Examples include reading a database, querying a map-reduce index, doing a full-text search, retrieving meta-information, or a list of changes.

HEAD
Displays the number of `HTTP HEAD` requests. `HEAD` requests are used only to determine whether information that was previously retrieved was updated.

POST Displays the number of `HTTP POST` requests. `POST` requests have various uses, such as creating documents, searching, or starting a replication.

PUT Displays the number of `HTTP PUT` requests. `PUT` requests are primarily used to update or create documents or databases.

HTTPD Status Codes

These request methods display metrics for 200, 300, 400, and 500 series status codes.

HTTPD 200 Series Status Codes
Displays metrics for 200 series status codes, which indicate that the requested operation completed successfully. Under normal circumstances, most status codes fall into this range.

HTTPD 300 Series Status Codes
Displays metrics for 300 series status codes, which indicate that further action is required by the client to fulfill the request. However, they do not indicate that an error occurred.

HTTPD 400 Series Status Codes
Displays metrics for 400 series status codes, which indicate an error occurred in servicing the request. For example, the request might be malformed or authentication might be missing or incorrect. Messages of this type usually indicate that a problem occurred in the application that made the request. Correct the problem in the application, and then repeat the request. In some cases, you can resolve an error by repeating the request, as in the case of a 409 error that was caused by a document update conflict.

HTTPD 500 Series Status Codes
Displays metrics for 500 series status codes, which indicate a problem with the cluster. If the

number of responses in this range increases, use **Weatherreport** to investigate whether there is an issue with the cluster. For more information about **Weatherreport**, see "Checking the health of a cluster with Weatherreport" on page 55.

CouchDB

Database read/write rate
: Displays the number of database reads and writes:
- **database_reads** identifies the number of times a document was read from a database.
- **database_writes** identifies the number of times a database was changed.

Document read/write rate
: Displays the number of document inserts, document writes, and local document writes:
- **document_inserts** identifies the number of times a document was inserted.
- **document_write** identifies the number of times a document was changed. These metrics give you an idea of the request profile of your application. They can help you optimize your application, such as reduce the number of changes in favor of inserts.
- **local_document_writes** is the number of local documents to which users have written.

Document map-doc / view-emit
: Displays the number of documents that were
- passed to a view map function (**map_doc**)
- emitted by map functions (**emits**)

These metrics correlate to view activity and indicate how well view builds are running.

IO Queues

Displays metrics about the IO queue for the cluster. You can use this information to pinpoint performance issues, since it identifies where most of the strain on a cluster is coming from. An example of this is whether view builds are using more IO resources than interactive requests, or vice versa.

IOQ Processing Rates by Class
: Displays the volume of IO queue requests for each request class. IOQ prioritizes IO requests according to configurable priorities.

IOQ Latency
: Displays the time delay or latency in processing requests through the IO queue.

IOQ Processing Rates by Type
: Displays the volume of IO queue requests for each request type.

IO Queue Types and Descriptions

interactive
: The volume of IO queue interactive requests, which are requests to IOQ that resulted from HTTP requests.

db_update
: The volume of IO queue `db_update` requests, which are requests to update a database.

db_compact
: The volume of IO queue `db_compact` requests, which are requests that resulted from database compaction jobs.

view_update
: The volume of IO queue `view_update` requests, which are requests to update view data.

view_compact
The volume of IO queue `view_compact` requests, which are requests that resulted from view compaction jobs.

internal_repl
The volume of IO queue `internal_repl` requests, which are requests from internal replication jobs, that is, replication between nodes in one cluster.

latency
IO queue latency information.

reads IO queue reads.

writes IO queue writes.

Message queues

Message Queue For Servers

Displays metrics about the length of the message queues for processes that are running on the Erlang VM on a node. Elevated and steadily increasing message queues indicate a potential problem. Any serious issues are reported by **`Weatherreport`**. For more information about **`Weatherreport`**, see "Checking the health of a cluster with Weatherreport" on page 55.

Descriptions for Message Queues Server

couch_event_server
The couch event server handles the `database _changes` feeds.

couch_server
The couch server.

ioq_server
The IOQ server handles requests to IOQ.

rexi_server
The `rexi` server handles RPC calls between nodes in a cluster.

Message Queue For Couch DB Updater

Displays distributions of message queue lengths for the Couch DB Updater, such as the 50th percentile, 99th percentile, and maximum, and minimum. Since only one or two Couch DB Updater and Couch File processes are affected, the most relevant metrics are usually the 99th percentile and maximum.

Message Queue For Couch

Displays distributions of message queue lengths for the Couch File, such as the 50th percentile, 99th percentile, and maximum, and minimum. Since only one or two Couch DB Updater and Couch File processes are affected, the most relevant metrics are usually the 99th percentile and maximum.

Erlang

Erlang Memory

Displays memory usage inside the `Erlang` VMs for various categories.

Memory Types and Descriptions

atom The total amount of memory that is allocated for atoms.

atom_used
The total amount of memory that is used for atoms.

binary The total amount of memory that is allocated for binaries.

code The total amount of memory that is allocated for code.

ets The total amount of memory that is allocated for `ets` tables.

processes
The total amount of memory that is allocated by the processes.

processes_used
The total amount of memory that is allocated by processes that are used.

other The total amount of memory that is used for everything else.

Erlang Statistics

Displays metrics for internal replication jobs, process count, run queue, and uptime.

Internal_Replication_jobs
Displays the total number of replication jobs on a node; each job is a set of changes that must be "pushed" to another node. This metric is essential in determining how consistent a cluster is. This information also is important when you bring a node back into the cluster because some application designs might be negatively impacted if a node is active, but not up-to-date.

Process_Count
Displays the total number of processes that are running on the `Erlang` VM.

Run_Queue
Displays the total number of processes that are in a runnable state, but are not running. This information is analogous to OS load average.

Uptime
Display a counter that is incremented every second that the `Erlang` VM is active. This information is useful in identifying when a node is rebooted.

Active tasks

Displays the number background jobs that are running and how many changes are pending on each job. You can use this information to correlate changes in internal cluster workload with changes in performance. For example, if someone uploads a new design document that creates a backlog of indexing work, you might see a corresponding drop in performance for interactive requests.

Database Compaction
Displays database compaction metrics, including `changes_pending`, `total_changes`, and `total_tasks`. Database compaction reduces disk space usage by removing unused and old data from the database files.

Indexer
Displays indexer metrics, including `changes_pending`, `total_changes`, and `total_tasks`. Indexer tasks are responsible for building map/reduce views.

Replication
Displays replication metrics, including `changes_pending` and `total_tasks`. Replication synchronizes two copies of the same database, which allows users to have low latency access to data no matter where they are.

Search Indexer
Displays search Indexer metrics, including `changes_pending`, `total_changes`, and `total_tasks`. Search indexer tasks are responsible for building the search indexes that are required to respond to search queries for the Lucene-based search.

View Compaction
Displays view compaction metrics, including `changes_pending`, `total_changes`, and `total_tasks`. View compaction reduces disk space usage by removing unused and old data from view index files.

Clouseau

Clouseau is a service that is used by `cloudant` to perform the `cloudant` search. The Cloudant search capability is built on Lucene and allows you to do more ad hoc queries over your data than can be done with primary and secondary indexes.

Performance of the Clouseau service is affected by available memory, in particular the maximum heap size. For information about modifying the heap size, see the information later in this topic.

In the following material, JVM refers to the Java Virtual Machine.

Index Opening Latency
The time that is taken for indexes to open, expressed as percentiles.

Index Opening Throughput
The number of search indexes opened per second. Once a search index is open, it is cached for future queries. If there is sufficient memory, index opening throughput is likely to be low or zero, even though search throughput might be high.

Search Latency
The time that is taken for individual search queries to complete, expressed as percentiles.

Search Throughput
The total number of search queries per second.

Commit Latency
The length of time that is taken to commit a search index to disk.

Garbage Collection: Count and Time
Garbage collection is the process of freeing space in the heap for allocation of new objects. There are two garbage collection categories: Garbage Collection Count and Garbage Collection Time. Each of these categories includes these items:

- **ParNew** is the garbage collection of newer objects in the JVM.
- **ConcurrentMarkSweep** is the garbage collection of older objects. JVM is paused while `ConcurrentMarkSweep` is running, which can have performance implications.

Garbage Collection Count
The total number of garbage collection calls that occurred.

Garbage Collection Time
The total duration of a garbage collection, expressed in milliseconds.

Heap Memory
Heap memory is the memory that is allocated to the JVM by the operating system:

- **committed** identifies the committed amount of heap memory that was used by the JVM.
- **initial** identifies the initial amount of heap memory that was used by the JVM.
- **max** identifies the maximum amount of heap memory that was used by the JVM.

If you are seeing performance issues that you think are related to the heap, for example garbage collection is taking a long time, you might try modifying the maximum heap size. Do this by:

1. Locate the line in `/etc/sv/clouseau/run` that sets the maximum heap size:
```
-Xmx2G \
```
2. Increase the value, for example from 2G to 4G.

3. Restart the service:
   ```
   sv restart clouseau
   ```
- **used** identifies the heap memory that was used by the JVM.

Non-Heap Memory
Non-heap memory is all of the memory that the JVM allocates for purposes other than the heap:
- **committed** identifies the committed amount of non-heap memory that was used by the JVM.
- **initial** identifies the initial amount of non-heap memory that was used by the JVM.
- **max** identifies the maximum amount of non-heap memory that was used by the JVM.
- **used** identifies the non-heap memory that was used by the JVM.

Open Files
The number of file descriptors opened by the JVM.

Uptime
The amount of time that the JVM has been running.

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, logging can remain as the default: 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

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`

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/syslogOn Red Hat-derived Linux™ distributions only: /var/log/secure /var/log/messages	None.

Configuring the Remote Logging server to use rsyslog

Before configuring any 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 is 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.

To confirm or enable a remote logging server:

1. Check if 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** with the following command:

   ```
   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. 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 on page 45.

Configuring a database node for remote or local logging

By default, local logging is enabled on Cloudant Local database nodes. If you intend to continue the use of local logging, then no further configuration changes are needed.

Logging can be switched between local and remote configurations by using the script configureLog.sh provided on each database node in the ~/Cloudant/repo directory. The script configures the logging for three Cloudant Local database node components:

- Cloudant dbcore
- Clouseau
- Metrics

To display all the options that are provided by the script, use the following command:
```
~/Cloudant/repo/configureLog.sh --help
```

Enabling remote logging for the database node by using the configureLog.sh script.

To switch the database node to use remote logging, run the configureLog.sh script, supplying the remote hostname and the logging port number. Do this using the following command:
```
~/Cloudant/repo/configureLog.sh --hostname <hostname> --port <port>
```

In this command, *<hostname>* is the remote server host, and *<port>* is the remote log server port.

Note: The configureLog.sh script must be run on every database node.

To verify logging on the remote logging server, check whether logs from database nodes are being written to the locations configured in /etc/rsyslog.d/cloudant.conf. The log file contains the messages from all database nodes that are configured for remote logging. To help you identify which messages belong to which node, the log messages include the database node hostnames and the identity value in the log message lines.

Note: On the database nodes, the Apache CouchDB service log is locally written under /var/log/cloudant/cloudant-svlogd/current. It is not available for remote logging. Log levels and other configuration changes for this log can be made by using /opt/cloudant/etc/default.ini and /opt/cloudant/etc/local.ini. The Apache CouchDB service log contains basic information about Couch DB service status, such as whether it is started or stopped.

Enabling local logging for the database node by using the configureLog.sh script.

If logging on the database node is set to remote logging, and you want to switch to use local logging, run the configureLog.sh script, supplying *localhost* as the remote hostname and 514 as the default **rsyslog** logging port number. Do this by using the following command:
```
~/Cloudant/repo/configureLog.sh --hostname localhost --port 514
```

Note: The configureLog.sh script must be run on every database node.

Configuring a load balancer node for remote or local logging

By default, local logging is configured for the HAProxy log using **syslog**.

To change the logging to use a remote server, or to extend the captured log information, use the following steps:

1. Edit the `haproxy.cfg` file on the load balancer. In the `'global'` section, search for a line similar to the following example:
   ```
   log 127.0.0.1 local4 warning
   ```
   This configuration sends log information at the `'warning'` level of detail to the **rsyslog** *facility* local4 running on the local system `localhost`, which has the IP address 127.0.0.1

   If you want to change the level of log detail, for example from `'warning'` to `'info'` or `'debug'`, simply make the corresponding change in the `haproxy.cfg` file. For example, to get details about individual requests, change the level of log detail to `'info'` or `'debug'`

   To send logging messages to a remote server, use the fully qualified domain name or IP address of that server instead of the `127.0.0.1` value. For example, to send log messages to remote log server `myremoteserver.com` at *facility* local4, use a configuration line similar to:
   ```
   log myremoteserver.com local4 info
   ```
2. On the logging server, configure **rsyslog** to log to a file. This configuration is always required for the logging server, regardless of whether it is local or remote. Ensure that there is a suitable configuration file in `/etc/rsyslog.d` on the logging server. For example, you might create a file `/etc/rsyslog.d/haproxy.conf`, containing the following configuration:
   ```
   local4.* /var/log/haproxy.log
   ```
   This configuration stores all log messages that are received through the `local4` *facility* to the file `/var/log/haproxy.log` on the logging server.
3. Restart **rsyslog** on the logging server:
   ```
   service rsyslog restart
   ```
4. Restart **haproxy** on the load balancer:
   ```
   sv restart haproxy
   ```
5. Check the logs on the logging server. If the logging level is `'info'`, the log should now contain details for each request. The information is similar to the following example:
   ```
   Dec 29 16:59:57 localhost haproxy[32241]: 216.58.49.100:50673 [29/Dec/2014:16:59:57.408]\
    dbfarm dbfarm/db1 1/0/2/10/13 200 757 - - ---- 1/1/0/0/0 0/0 "GET /loadtest HTTP/1.1"
   Dec 29 16:59:58 localhost haproxy[32241]: 216.58.49.100:50673 [29/Dec/2014:16:59:57.421]\
    dbfarm dbfarm/db2 570/0/1/9/581 200 757 - - ---- 1/1/0/0/0 0/0 "GET /loadtest HTTP/1.1"
   Dec 29 16:59:58 localhost haproxy[32241]: 216.58.49.100:50673 [29/Dec/2014:16:59:58.001]\
    dbfarm dbfarm/db3 555/0/1/10/566 200 757 - - ---- 1/1/0/0/0 0/0 "GET /loadtest HTTP/1.1"
   Dec 29 17:00:09 localhost haproxy[32241]: 216.58.49.100:50677 [29/Dec/2014:17:00:09.570]\
    dbfarm dbfarm/db1 0/0/1/4/6 200 331 - - ---- 1/1/0/0/0 0/0 "GET / HTTP/1.1"
   ```

For more information on configuring **rsyslog**, see http://www.percona.com/blog/2014/10/03/haproxy-give-me-some-logs-on-centos-6-5/.

For more information on HAProxy logging, see http://cbonte.github.io/haproxy-dconv/configuration-1.5.html#8.

Note: On load balancer nodes, the NGINX log files are stored locally under `/var/log/nginx/access.log` and `/var/log/nginx/error.log`. These files contain status information about the NGINX service. The information is used by the Cloudant Local dashboard. It is not possible to store these logs remotely, for reasons that are discussed at http://nginx.org/en/docs/syslog.html. The logging levels for NGINX logs are configured in the `/etc/nginx/nginx.conf` file.

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 for 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 values that are used by Cloudant Local are:

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. Remember to change the logging level back to the default or your preferred level after completing your verification checks.
- 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:

```
[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 is where 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 here: http://linuxers.org/howto/howto-use-logrotate-manage-log-files. An alternative option is to use output channel scripts, as described here: 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. These logs enable you to get security audit and system level information. If the logs are not enabled on your OS platform by default, refer to platform-specific information for details about configuring them.

For Debian and Ubuntu, the logs are:

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

For Red Hat-derived Linux™ distributions only, the logs are:

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

Configuring database-level security

Use these instructions to configure database-level security for Cloudant Local. This is an optional step that 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 host name 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. In the example, we do the following steps to illustrate this information:
- Confirm that our **admin** user is identified in our `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 our security settings were properly applied.

Here are the steps we did to accomplish these tasks. For reference purposes, the steps are written from the perspective that you, the reader, are doing the steps.
1. Confirm that your **admin** user is specified in your `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 host name of your configured cluster load balancer.
   - Replace **db1** with the name of your database.

   You will 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 will receive an `{"ok":true}` message after the document is created, and the message will include the document id and revision number for the document. Here is an example:
   ```
   {"ok":true,"id":"171806ad7968475970bf5450e91a5259","rev":"1-
   e6accc814683c1cadf6c74b492570c42"}
   ```
   The unauthorized user was allowed to add the test document because there is no database-level security 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 will 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 will receive an {"ok":true} message after the document is created, and the message will include the document id and revision number for that user document. Here is an 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.

You will receive an {"ok":true} message after the document is created, and the message will include the document id and revision number for that user document. Here is an example:

```
{"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 on page 50.

```
$ 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 will 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 50.

- 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 will 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 will 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 will receive a message like this one:

```
{"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 will receive a message like this one:


```
{"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 will receive a message like this one:

     ```
     {"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 will receive an `{"ok":true}` message like this one:

     ```
     {"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 will receive an `{"ok":true}` message like this one:

     ```
     {"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 will receive a message like this one:

  ```
  {"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 will receive a message like this one:

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

  As the message indicates, you must be an admin-level user to write 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 will receive a message like this one:

  ```
  {"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

Uninstalling Cloudant Local

Use these instructions to uninstall Cloudant Local with the `uninstall.sh` script in quiet mode or manual mode.

The Cloudant Local `uninstall.sh` script is used to uninstall the product in both manual and quiet (or silent) mode.

- In quiet mode, you run the `uninstall.sh` script with two parameters to do the uninstall. Quiet mode is also referred to as silent mode.
- In manual mode, you run the `uninstall.sh` script without any parameters and respond to a few simple prompts to complete the uninstall.

Important: You must use the same user ID that you used to install Cloudant Local to uninstall the product. For example, if you installed as root, you must uninstall as root.

Uninstall Cloudant Local in Quiet Mode

You can use the `uninstall.sh` script to remove the extracted Cloudant installation packages, and optionally uninstall the Cloudant packages from your operating system.

Follow these instructions to uninstall Cloudant Local in quiet (or silent) mode.

1. Change to the directory in which the `uninstall.sh` script is installed. The default directory is `~/Cloudant/`, unless you specified another directory during the extract process.
2. Use this command to display the syntax help text for the **`uninstall.sh`** command:

   `./uninstall.sh -h`

   Here is the help text:

```
Usage:
uninstall.sh  [option]     [type]
              [-h]|[-s|-q  [-l|-r]]

Where:
 option is one of these optional parameters:
  -h     = Display this help text and exit
  -q     = Run the uninstall in quiet or silent mode to suppress most messages
  -s     = Same as -q

 type is one of these required parameters if you use -q or -s:
  -l     = Leave Cloudant running and only delete the extracted installation
           packages, so that Cloudant is still available for your use
  -r     = Stop Cloudant and remove the packages from the operation system,
           so that both the product and packages are removed from your system
```

3. Use one of the following commands to uninstall Cloudant Local in quiet mode:
   - If you want to remove Cloudant Local and its installation packages from your operating system, use this command:

     `./uninstall.sh -q -r`

     Use this command only if you do not want to use Cloudant Local anymore, because this command will remove both the product and its installation packages.
   - If you want to remove the installation packages from your system, but you want to continue using Cloudant Local, use this command:

     `./uninstall.sh -q -l`

     Use this command to remove the previous installation packages from your system before you install a new Cloudant Local release or fix pack.
4. When the uninstall is completed, an appropriate completion message is displayed.

Uninstall Cloudant Local in Manual Mode

Follow these instructions to uninstall Cloudant Local in manual mode.

1. Change to the directory in which the `uninstall.sh` script is installed. The default directory is `~/Cloudant/`, unless you specified another directory during the extract process.
2. Use this command to run the uninstall script in manual mode:

   `./uninstall.sh`
3. You will receive the following prompt:

   `Do you also want to remove the installed Cloudant packages from the operating system? (y/n):`

   Do one of the following in response to this prompt:
   - If you want to remove Cloudant Local and its installation packages, type **`y`** and press Enter.

Choose this option only if you do not want to use Cloudant Local anymore, because this option will remove both the product and its installation packages.

- If you want to upgrade to a new Cloudant version or fix pack, type **n** and press Enter to remove the previous installation packages from your system.

  Choose this option if you want to continue to use Cloudant Local, and you want to install a new release or fix pack.
- Type **quit** and press Enter to exit the uninstall process.

4. When the Remove Cloudant Packages prompt is displayed, press Enter to proceed with the removal.

   Note: You can type **quit** and press Enter at any time to cancel and exit the uninstall wizard.
5. When the Removal Options prompt is displayed, type **1** and press Enter to remove all extracted Cloudant packages and files from the installation directory.
6. When the uninstall is completed, an appropriate completion message is displayed.

Cloudant Local maintenance tasks

Use the following tasks to 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, do the following steps:

1. As root or the `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

A list of the checks that are carried out is available with:

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

Currently, this list includes the following items:

```
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 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 with:

```
$ /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. It is likely that oom killer will kill 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, it is probably a process getting a long message queue backing up. That probably means you can kill the process. however, before you do that, use process_info to find out more about the process. To find processes that are using a lot of memory within dbcore, log in to the node and start a remsh session. Find the processes that are consuming 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 kill those processes.

Use process_info to find out what the process is:

```
(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 kill 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
```

To run a GC, from a remsh on the node:

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

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

sudo sv restart cloudant

The message_queues checks

This check covers a group of checks, as follows.

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; there is one couch_db_updater process 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 should 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 like this:

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

This command will return a list of PIDs with long (> 1000) message queues. To kill all processes in this list, you can run:

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), kill 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 kill 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 now empty, as follows:

```
> 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; there is one couch_file process 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, the process usually enters a "death spiral" of sorts, and will not recover, so you must kill the process.

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) like this:

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

How to remediate the failure.
In remsh, run this command to kill 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 should return an empty list:

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

When to escalate the page.
If you were paged multiple times for a given cluster, that is, the issue is recurring, escalate the page.

The couch_server message queue check

What it checks.
This checks whether the couch_server message queues are getting large, 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 affect of a backed up couch_server is dramatically increased latency on a subset of requests. That is, those for which it is 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 starting to decrease, 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 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, there is an ongoing problem. If the number of messages is decreasing, but peaking often, additional 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", "Alle
viate global_changes_server message queue pain"]).
> rpc:multicall(config, set, ["global_changes", "max_write_delay", "5000", "Alle
viate 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 the values in the logs that were mentioned previously. They should go back 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 should still be functioning correctly, but slower, as it reads from the disk more often.

How to fix it.

You can watch the increase on the node, as follows:

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

This probably will show a speedy increase in the message queue length, such as 1000 per second.

First, try to 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 last resort fix is to kill 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 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 they are having trouble processing their mail boxes, that communication will be 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 in boxes and their sizes:

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

This should return a list of {`registered_process_name`, `mailbox_size`} tuples; the sum of the message queue length values will exceed the alert threshold.

How to fix it.

As indicated previously, there are two ways to configure `rexi` communication patterns. 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 should return the string “false”. If it 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 just as a fail-safe measure, and attempting to do so multiple times will 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 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.

If Custodian finds that some of the shards have fewer or more than the appropriate number of replicas listed in the shard table or that the node to which a shard is mapped is unavailable, it raises an alarm.

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 should be addressed.

Often, this warning is a proxy for a database node being down (such as a `dbcore` reboot or hardware failure), as this will have rendered a good number of shard replicas unreachable. If it is not caused by a node being down, it might be caused by one of the following issues:

- Someone deliberately created a database with less than N replicas.
- A shard move failed, and a shard has gone missing (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 server being down, unreachable, or not connected to all other nodes. If that is not the case, contact support.

The `disk` check

What it 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. This is bad and 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 / is read-only, this can cause a number of other problems. At the least, chef will not be running.

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

If the disk is mounted read/write and there is space, the file system might have exhausted 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 could not be written? Check for the existence of /srv/ and that it is writable.
- Check the load on the server (using uptime or top) to see whether a high load is causing file-system operations to time out.

You likely have to reboot 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 provide 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 generally more than 20k, IOQ is probably saturated with traffic. A quick fix to this problem is to bypass interactive IOQ traffic, as follows:

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

After this bypass is enabled, the pending request count should trend 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 reboot `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 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 will automatically reconnect:

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

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

Follow these instructions to:
- 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 will be displayed in response to this command.
   - If it is not safe, one or more shard ranges will be 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) will be displayed, along 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 55.
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 "Extracting and Installing Cloudant Local" on page 13.

5. Configure the node as described in “Configuring Cloudant Local” on page 20, but with the following modifications to the configuration process:
   a. Update the `configure.ini` file. Confirm that the `configure.ini` file on this node has correct and complete information about the cluster. Check the names and IP addresses of the database and load balancer nodes, username and passwords, encrypted passwords and the cookie value. If required, refer to or copy the last updated version of the `configure.ini` file that might exist on the other database nodes in the directory `/root/Cloudant/repo`. Ensure the file is accurate and complete.
   b. Run the `configure.sh` script with a *–m* argument to configure and start the node in maintenance mode:

      ```
      configure.sh -m
      ```

      The node being replaced must be started in maintenance mode. The *–m* argument in the command configures the node based on the `configure.ini` file, and starts the node in maintenance mode.

      Note: After copying the original `local.ini` file back to the `/opt/cloudant/etc/` directory, you must add the following configuration variable settings to the `local.ini` file.

      ```
      [chttpd_auth]
      authentication_db = default/_users
      ```

      You must use a database name `_users` as your authentication database. If you use a different database name, you must retain the `default/` prefix as in the example above.
   c. Confirm that the node is in maintenance mode. Use the following command to confirm the node is in maintenance mode:

      ```
      curl localhost:5984/_up
      ```

      If the node is in maintenance mode, you receive the following response:

      ```
      {
          "status": "maintenance_mode"
      }
      ```

      You can also manually place the node in maintenance mode by adding the line

      ```
      maintenance_mode = true
      ```

      in the **[couchdb]** section of the `/opt/cloudant/etc/local.ini` file, then restarting Cloudant.
6. 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. Here is an example of what to you are looking for:

   ```
   <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 will indicate 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 "Extracting and Installing Cloudant Local" on page 13.
5. Configure the replaced node as described in "Configuring Cloudant Local" on page 20, but with the following modifications to the configuration process:
   a. Update the `configure.ini` file. Confirm that the `configure.ini` file has correct and complete information about the cluster. Check the names and IP addresses of the database and load balancer nodes, username and passwords, encrypted passwords and the cookie value. If required, refer to or copy the last updated version of the `configure.ini` file that might exist on the other database nodes in the directory `/root/Cloudant/repo`. Ensure the file is accurate and complete.
   b. Run the `configure.sh` script to configure and start the load balancer:
   ```
   configure.sh
   ```
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

Install Cloudant Local on the database node that you are adding, as described in "Extracting and Installing Cloudant Local" on page 13.

Configure the node as described in "Configuring Cloudant Local" on page 20, but with the following modifications to the configuration process:

1. Update the `configure.ini` file. Confirm that the `configure.ini` file on this node has correct and complete information about the cluster. Check the names and IP addresses of the database and load balancer nodes, username and passwords, encrypted passwords and the cookie value. If required, refer to or copy the last updated version of the `configure.ini` file that might exist on the other database nodes in the directory `/root/Cloudant/repo`. Ensure that the file is accurate and complete.
2. Add the name and IP address of this new node to the database node section in the `configure.ini` file.
3. Run the `configure.sh` script with a *–m* argument to configure and start the node in maintenance mode:
   ```
   configure.sh -m
   ```

   The node to be added must be started in maintenance mode. The *–m* argument in the command configures the node based on the `configure.ini` file, and starts the node in maintenance mode.

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. Confirm that the node is in maintenance mode. Use the following command to confirm that the node is in maintenance mode:

```
curl localhost:5984/_up
```

If the node is in maintenance mode, you receive the following response:

```
{
    "status": "maintenance_mode"
}
```

You can also manually place the node in maintenance mode by adding the line:

```
maintenance_mode = true
```

in the **[couchdb]** section of the `/opt/cloudant/etc/local.ini` file, then restarting Cloudant.

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

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

`curl 'localhost:5986/nodes/cloudant@db4.cluster001.cloudant.net' -X PUT -d '{}'`

Use this command to verify that the node was successfully added to the `nodes` database:

`curl -u admin:password http://localhost:5986/nodes/_all_docs?include_docs=true`

If the node was added to the `nodes` database, the output from this command includes the new node in the resulting nodes list. The list also includes the document ID specified for each database node in the cluster.

Use this command to verify that membership for the new node was replicated to every database node in your cluster:

curl -u *admin:password* **http://***hostname***:5984/_membership**

Replace *admin:password* in the preceding command with the credentials for the database administrator. Also, replace *hostname* with the fully qualified domain name for the database node.

Run the preceding curl command for every node to confirm that membership for the new node is included in the output for the other nodes in your cluster.

Step 4: Update the load balancer configuration

The load balancers must know to direct traffic to the new node. For each load balancer, add the new node to the list of nodes in the `/etc/haproxy/haproxy.cfg` file. You must restart haproxy after the changes.

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 any required changes, such as DNS settings, to reroute all traffic through the failover load balancer.

Step 5: Create a rebalancing plan

The next step is to rebalance the cluster.

If you are using more than one load balancer, you can run the rebalancing process from any one of your load balancers. If you do not have a configuration file for the rebalancer, create a file named `rebal.ini` and save it in the home directory for the user on the load balancer. The format of the configuration file is as follows:

```
[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 host names have a private and a public interface and the load balancer does
# not have access to the private interface. If the private host name can be used,
# these fields can be left blank. Otherwise, the private host name extension
# is replaced with the public host name extension to arrive at the public host
# name. 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
```

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 this curl command:

curl http://*admin:password***@localhost:5986/_active_tasks**

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 execute 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. For 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. On every database node in your cluster, create an `authorized_keys` file in `/opt/cloudantrebal/.ssh`. Then, add the public key generated in step 1 on page 68 to that file.
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 on page 68 is specified by providing the file name of the private key to `ssh-add`. For 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 would 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.

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. Next, 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
```
   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.
4. 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.

Use the following command in `remsh` to put the node in production mode:
```
config:set("couchdb", "maintenance_mode", "false").
```

Note: Pay close attention to the '`.`' character at the end of the command.

Run the following command on the new database node to verify the status of that node:
```
curl localhost:5984/_up
```

If the node is functioning normally in production mode, you receive the response:
```
{"status":"ok"}
```

Step 7: Run rebalancing

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

Use the following command to run your rebalancing plan:

```
rebal run rebalance_plan_filename
```

Replace *rebalance_plan_filename* with the file name for your rebalance plan. The file name was specified earlier when you edited the `rebal.ini` file while creating a rebalancing plan.

The command executes the shard moves identified in the plan. Shards moves are batched by database name with each batch being executed in series. Multiple batches are executed concurrently, with batches of 20 as the default. Progress and errors are logged to the `rebalance_plan.log`. You can 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, respectively.

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
     ```
- It might be 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.
- If you see many `erl_call: failed to connect to node` messages in the `dbname.out` files, 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. Try running 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 checked the configuration and tried to rerun the shard moves, but the problem persists, contact support.

Step 8: Verify the new node

To confirm that the new node is set up correctly, log in to it with SSH and run Weatherreport with this command:

```
/opt/cloudant/bin/weatherreport
```

If errors are reported during the run, see the information for Weatherreport checks, or call support.

After the run finishes, check the rebalancing logs that are described in the Run rebalancing step for any errors.

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%2F<database_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 55.

Upgrading your Cloudant Local installation

Use this information to understand the upgrade tasks for an existing Cloudant Local installation.

Overview

There are two aspects to upgrading an existing Cloudant Local installation:
- The Cloudant Local binary code is upgraded.
- The internal data storage format used by Cloudant Local is upgraded to take account of changes affecting format or directory usage.

Upgrading the data storage format is not always required, as it depends on what capabilities you are utilizing. When a format upgrade is required, it is automatic and is a 'one-way' process. Once completed, it is not possible to rollback to an older version.

Each node within the installation can be upgraded one at a time. This means that there should be no need for any downtime.

Upgrading a node is very similar to replacing a node to a cluster. The basic upgrade steps are as follows:
1. Stop any Cloudant Local services running on the node.
2. Install the new Cloudant Local packages on the node.
3. Configure the node using maintenance mode.
4. Check status of the node.
5. Bring the node out of maintenance mode.

Step 1: Stopping Cloudant Local services running on the node

To do this, use the `cloudant.sh -k` command described in "Managing Cloudant Local services" on page 25.

Step 2: Install the new Cloudant Local packages on the node

To do this, follow the instructions in "Extract and Install Cloudant Local" on page 14. You are effectively performing a fresh install of the newer version. When you run the `quiet_install.sh` script, the installation process detects the presence of an existing Cloudant Local installation on the node, and modifies the install accordingly.

Step 3: Ensure that the correct configuration is present in the `configure.ini` file

To do this, follow the instructions in Update the `configure.ini` file.

Important: If you are reusing an existing metrics database from a previous version, the indexes used by the metrics application need to be rebuilt as part of the upgrade process. Metrics will be unavailable during that time. Creating a new metrics database is recommended as a best practice for this release. In the `configure.ini` file, update the `_METRICS_DBNAME` field from `metrics` to a new database name, for example:

```
_METRICS_DBNAME=metrics_v3
```

You can use the new metrics application (1.0.0.3) to view the statistics from both the original and existing metrics databases as well as from your new metrics database. The metrics user interface is available at `http://`*`loadbalancer.company.com/metrics-db`*`_app/statistics/index.html`. For example, if the metrics database name is metrics_v3, you point to the following URL:

```
http://loadbalancer.company.com/metrics_v3_app/statistics/index.html
```

On the metrics login page, specify the metrics database whose metrics you want to view. In this example, you can specify and point to your original database `metrics`, or specify and point to your new database `metrics_v3`.

Step 4: Configure the node using maintenance mode

To do this, follow the instructions in "Configuring Cloudant Local" on page 20. The `configure.sh` script automatically places the installation into maintenance mode as part of the configuration process.

Step 5: Check the status of the node

To do this, use the `cloudant.sh -t` command described in "Managing Cloudant Local services" on page 25. Use this command to test that the installation is properly configured, and can respond to a basic API request.

Step 6: Bring the node out of maintenance mode

To do this, use the `cloudant.sh -w` command described in "Managing Cloudant Local services" on page 25.

Configuring Failover Load Balancers

You can configure the way in which load balancing operates in the event of a load balancer failing 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 additional or replacement node balancers for a Cloudant Local system.

We 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

Table 2. Example Load Balancer configuration (continued)

IP Address	System
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). Do this by using the `quiet_install.sh` command, with the `-l` option to install a load balancer node, as described here.
2. On each load balancer node, as the root user modify the `haproxy.cfg` file to identify the front end address. Do this by changing the line:
```
listen dbfarm 0.0.0.0:80
```
to read:
```
listen dbfarm 10.10.50.5:80
```
where the new IP address is that of the "Front end" system.
3. On each load balancer node, as the root user install the **keepalived** application. For CentOS or RHEL, use the command:
```
yum install keepalived
```
For Debian or Ubuntu, use the command:
```
apt-get install keepalived
```
4. On each load balancer node, create a `/etc/keepalived/keepalived.conf` file. The file should be 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` .

Set the `virtual_ipaddress` IP to be that of the front end server.
5. 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
```

6. Check that the “front end” IP address can be accessed successfully. Do this by going to the IP address in a browser.
7. Check that load balancing is working correctly. Do this by disconnecting the highest priority load balancer, then revisit the front end IP address. Check that the access is successful.

   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. This will tell you whether db 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 command returned:

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

So, we know 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 there is a large shard (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`:

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

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

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

If compaction failed previously, there will be a `compact.data` and `compact.meta` file in `/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 will return nothing if there is no longer a compaction process using the file (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 inappropriate shards being compacted (for example, those 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. These files include:

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

Let's look at a database node:

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 are examples of changes that can affect request latency:

Workload

Determine whether there is a correlation between request volume and latency. Request volume alone does not give a clear picture of the workload. It is worth breaking the traffic down 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?

If the logs show that most requests go to one or a few databases or have the same request type (such as the same HTTP method) or backend name, take a closer look at those items to find the cause of the problem.

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

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"]).
```

Also, to 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.

To 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 for execution, 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 there are correlating increases 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 there are no obvious signs of a specific problem, the cluster is balanced and the increases are not limited to a single node, the cluster might be overloaded. Verify that there is a correlating increase in requests and consider adding more nodes to the cluster.

Tuning parameters for common replication cases

Use these instructions to tune parameters for common replication cases.

Here are the parameters that 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

Here are some scenarios and suggestions about what 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` should 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 should 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. There are separate sets of channels for database and view compactions. 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 a given 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 two queue types are as follows:

- **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 `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 operate on two main measures:

- **user_bytes**: the amount of data the user has in the file. It does not include storage overhead: old revisions, on-disk `btree` structure, and so on.

- **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"
}
```

user_bytes is 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 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. There are some conventional ones:

- ratio_dbs: a ratio channel for dbs, usually using the default settings.
- slack_dbs: a slack channel for dbs, usually using the default settings.
- ratio_views: a ratio channel for views, usually using the default settings.
- slack_views: a slack channel for views, usually using the default settings.

These four are defined by default if there are no others set. And some standard definitions for ones that often must be added:

- big_dbs: a ratio channel for enqueuing only large database shards. What large means 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, we set the min_size and concurrency settings, and allow the priority to 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

The same as defining a channel, you need to set 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

There are a few standard things that operators often must do 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 using:

```
> 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}}}]}]}]}
```

This gives you the node-local status for each queue. Under each channel, there is some information about the channel, such as the following items:

- active: number of current compactions in the channel.
- starting: number of compactions starting-up.
- 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, we can guess that 981,756 is very high. However, this might be a small database, so it does not necessarily mean useful compactions from the point of view of reclaiming disk space.

For this example, we can see that there are many queued compactions, but we do not know which 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 db number and size.

Smoosh IOQ priority

This 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 a lot of work that it is not getting through, priority can be increased. However, be careful that this does not adversely impact request performance.

```
rpc:multicall(config, set, ["ioq", "compaction", "0.5"]).
{[ok,ok,ok],[]}
```

In general, this 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 a given channel can allow a backlog in that channel to catch up. Again, some clusters run best when specific channels have more workers. From assessing disk space, you should know whether the biggest offenders are db or view files. From this, 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` is often best, but evaluate this based on the current status. If you want to increase the `ratio_dbs` setting:

```
> rpc:multicall(config, set, ["smoosh.ratio_dbs", "concurrency", "2"]).
{[ok,ok,ok],[]}
```

Suspending Smoosh

If Smoosh is causing issues, you can suspend its operation. This differs from either application:stop(smoosh). or setting the concurrency for all channels to zero because it both 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 step might indicate that there is a bug in 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

There are the main settings with which one interacts:

- **`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 for before it is recalculated. Defaults to 5.

Sometimes it is necessary to use the following items:

- **`cleanup_index_files`**: Whether Smoosh cleans up the files for indexes that were deleted. Defaults to false and probably should 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 before starting compactions to allow time to observe the system and make a decision about what to compact first. Hardly ever changed from the default. Default 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 this many bytes 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 this many bytes 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, there are two channels for databases (`ratio_dbs` and `slack_dbs`), and two channels for views (`ratio_views` and `slack_views`) Smoosh enqueues the new item to the first channel that accepts it. If none accept it, 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 they are currently running, and how many jobs are enqueued (as well as 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 healthy status output has items in the `ratio_dbs` and `ratio_views` channels. Owing to the default settings, `slack_dbs` and `slack_views` probably will have items in them.

```
smoosh:enqueue_all_dbs(), smoosh:enqueue_all_views()
```

These functions do what they say but you usually do not need to call them, since Smoosh will find the best compaction candidates on its own. However, if you experience disk space issues, they might help. If they do, that indicates a bug or configuration issue with Smoosh. Check your configuration and contact support, if needed.

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 two 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, point 1 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 two 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 that the 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 four channels:

- Compaction
- Internal replication
- Low

- Customer

The Customer channel is effectively a meta-channel where each item in the queue represents a backend / dbname combination that consists of a further three 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 respectively.
- 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 as follows:

- 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 recommended that 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. Note that tuning IOQ is not the answer to all performance problems and there are a finite number of gains to be had (possibly zero). You should 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

IOQ ships with a default 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 follows:

```
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 will 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 if

1. The number of active requests or IOQ latency are consistently elevated.
2. Disk utilization 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`. The following warnings should be heeded when you consider setting an IOQ bypass:

- Other request classes will continue to be routed through IOQ and will 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.

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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

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