Couchbase

# Architecture and Administration Basics

## Security

# Security Pillars in Couchbase

| Authentication | Authorization | Crypto | Auditing | Operations |
|---|---|---|---|---|
| App/Data: SASL AuthN<br>Admin: Local or LDAP or<br>PAM Authentication | Local Admin User<br>Local Read-Only Admin<br>RBAC for Admins<br>RBAC for Applications (since 5.0) | TLS admin access<br>TLS client-server access<br>Secure XDCR<br>X.509 certificates for TLS<br>Data-at-rest Encryption*<br>Field-level Encryption (since 5.5)<br>Secret Management | Admin auditing<br>API request auditing (since 5.5)<br>N1QL auditing (since 5.5) | Security management via UI/CLI/REST |

* Via third-party partners

# 1 Authentication

# Authentication

## Internal

- Username/password
  - Users and passwords stored in Couchbase
- Certificate-based authentication
  - Client certificate signed using the same cluster CA that was used to sign the node certificates
  - Username encoded in one of the fields of the client certificate

## External

- Couchbase stores only user names
- Password is validated by an external system
  - LDAP server
  - Pluggable Authenication Modules (PAM)
- Authentication method is configured over **saslauthd**

# Pluggable Authentication Modules (PAM)

- Allows UNIX local accounts to authenticate as Couchbase administrators
- Pluggable authentication architecture that is policy driven

## Centralized Management

Centralized and synchronize administrator account management using UNIX user management services

## Security Policy Enforcement

Allows configuration of strong security policies such as strong password requirements

# Demo: Setting Up PAM Authentication

- Setting up PAM-based authentication, creating an external user

  https://docs.couchbase.com/server/6.0/manage/manage-security/configure-pam.html

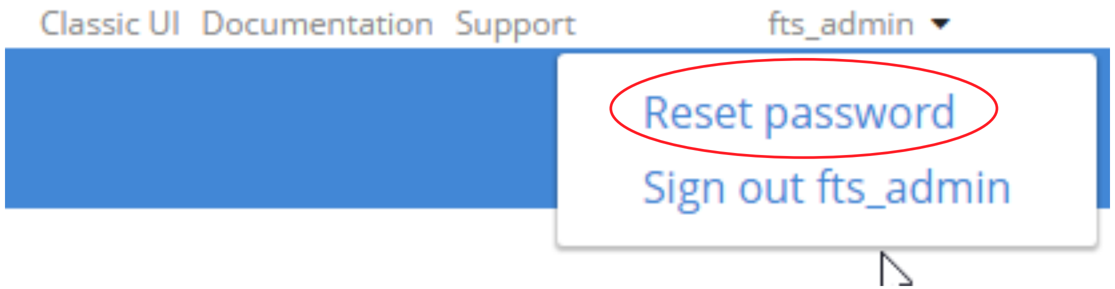
# Password Policy and Rotation

Default Policy

```
{
 "enforceDigits": false,
 "enforceLowercase": false,
 "enforceSpecialChars": false,
 "enforceUppercase": false,
 "minLength": 6
}
```

## Policy and Rotation

- Simple password policy rules enforced when initially set or rotated
- Policy can be set using REST or CLI: couchbase-cli setting-password-policy
- Password can be reset using UI, REST or CLI



2

# Role-Based Access Control (RBAC)

# Role-Based Access Control (RBAC) for Administrators

Role-Based Access Control (RBAC) allows you to specify what each admin can access in couchbase through role membership

## Regulatory Compliance

A strong demand for applications to meet standards recommended by regulatory authorities

## Segregation of Admin Duties

Every admin does not have all the privileges. Depending on the job duties, admins can hold only those privileges that are required.

## Security Privilege Separation

Only the full-admin has the privilege to manage security, and his/her actions can be audited just like other administrators.

# Role-Based Access Control (RBAC) for Applications

- Meet regulatory compliance requirements for data users and applications
- Simplified access control management for data and admin users across the cluster

## Regulatory Compliance

A strong demand for applications to meet standards recommended by regulatory authorities

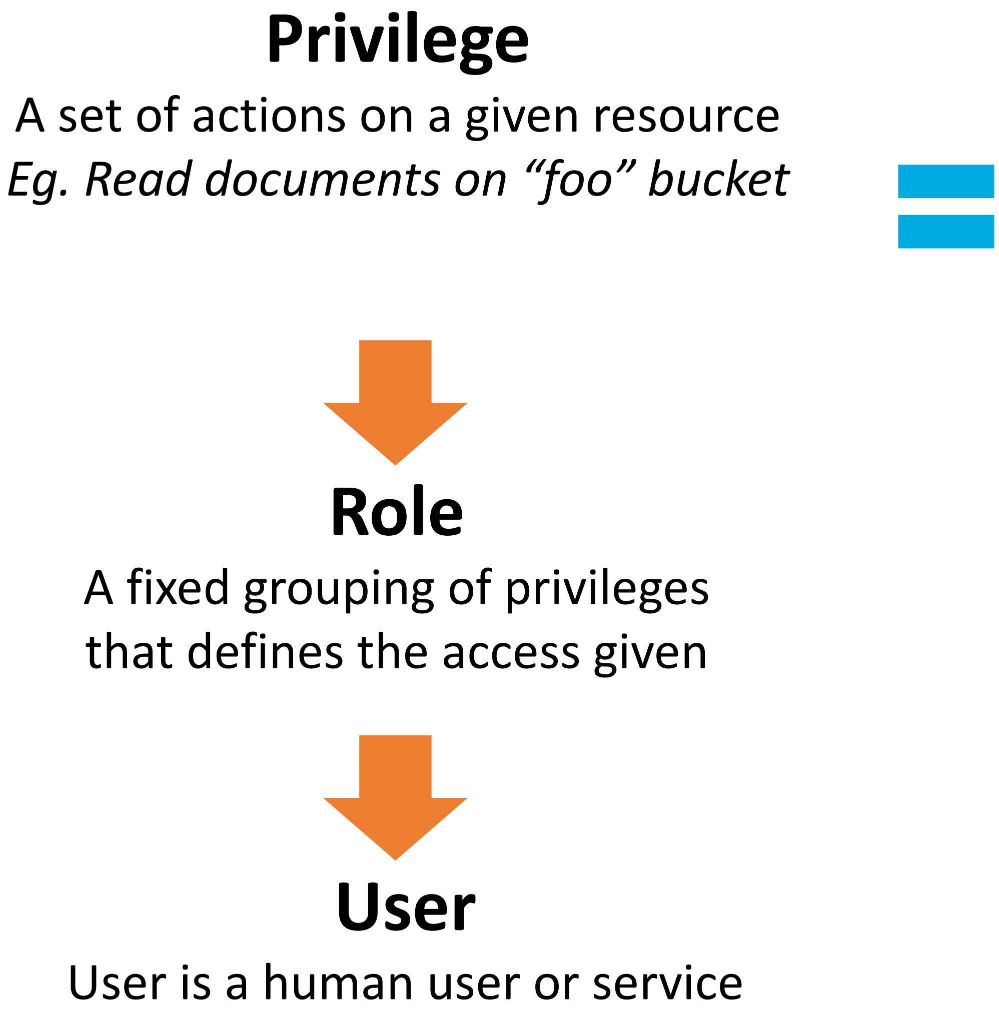
## Segregation of User Duties

Depending on the job duties, users can hold only those privileges that are required

## Locking Down Services

Depending on what the service is needed for, only those roles can be assigned

# RBAC Security Model



## Privilege

A set of actions on a given resource
*Eg. Read documents on “foo” bucket*

## Role

A fixed grouping of privileges that defines the access given

## User

User is a human user or service

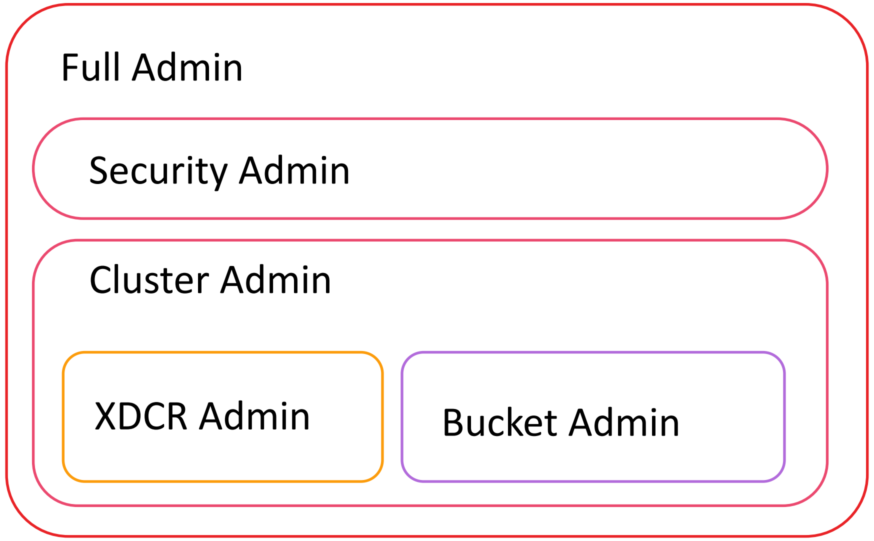
**Action:** an operation *eg. read, write, read metadata*

**Resource**: some system object that an action can be performed on. *eg. bucket, index, etc.*

- NIST Model
- Scalable users accounts
- Fixed out-of-the-box data roles in 5.0
- 1:N User-to-role mapping
- Roles can be applied for specific buckets / across all buckets [*]

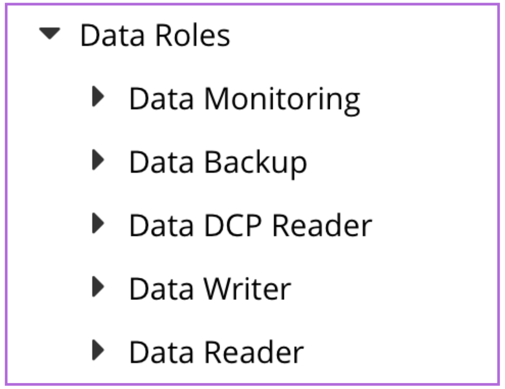
# Administrator Roles

- Roles pre-defined with permissions for specific resources
  - Full admin: full access to cluster and data
  - Security Admin: user management
  - Cluster Admin: all kind of cluster and bucket configuration
  - XDCR Admin: create and manage XDCR
  - Bucket Admin: create and manage buckets
  - Read-Only Admin: read-only access to cluster configuration, monitoring statistics
- All admin roles, except Full Admin, do not provide access to data



# Roles for Data Service

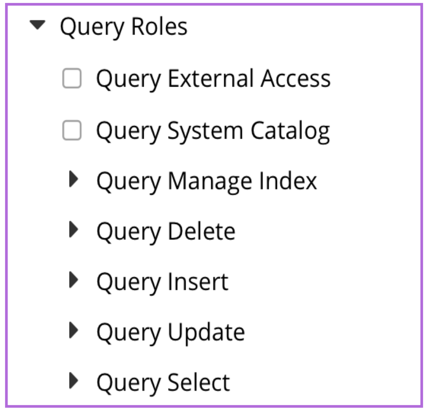
| | |
|---|---|
| Data Reader | • Read data from bucket |
| Data Writer | • Write data to bucket |
| Data DCP Reader | • Can read the DCP stream from bucket |
| Data Backup | • Can backup/restore the bucket |
| Data Monitoring | • Can monitor statistics for bucket |



- Data Roles
  - Data Monitoring
  - Data Backup
  - Data DCP Reader
  - Data Writer
  - Data Reader

# Roles for Query Service

| | |
|---|---|
| Query Select | Can execute SELECT N1QL statement for bucket |
| Query Update | Can execute UPDATE N1QL statement for bucket |
| Query Insert | Can execute INSERT N1QL statement for bucket |
| Query Delete | Can execute DELETE N1QL statement for bucket |
| Query Manage Index | Can execute index management statements for bucket |
| Query System Catalog | Can query system tables for bucket |
| Query External Access | Can execute N1QL CURL statement |



- ▾ Query Roles
  - ☐ Query External Access
  - ☐ Query System Catalog
  - ▸ Query Manage Index
  - ▸ Query Delete
  - ▸ Query Insert
  - ▸ Query Update
  - ▸ Query Select

# Bucket Roles

**Application Access**

- Full Read/Write access over the bucket for compatibility for pre-5.0 authentication

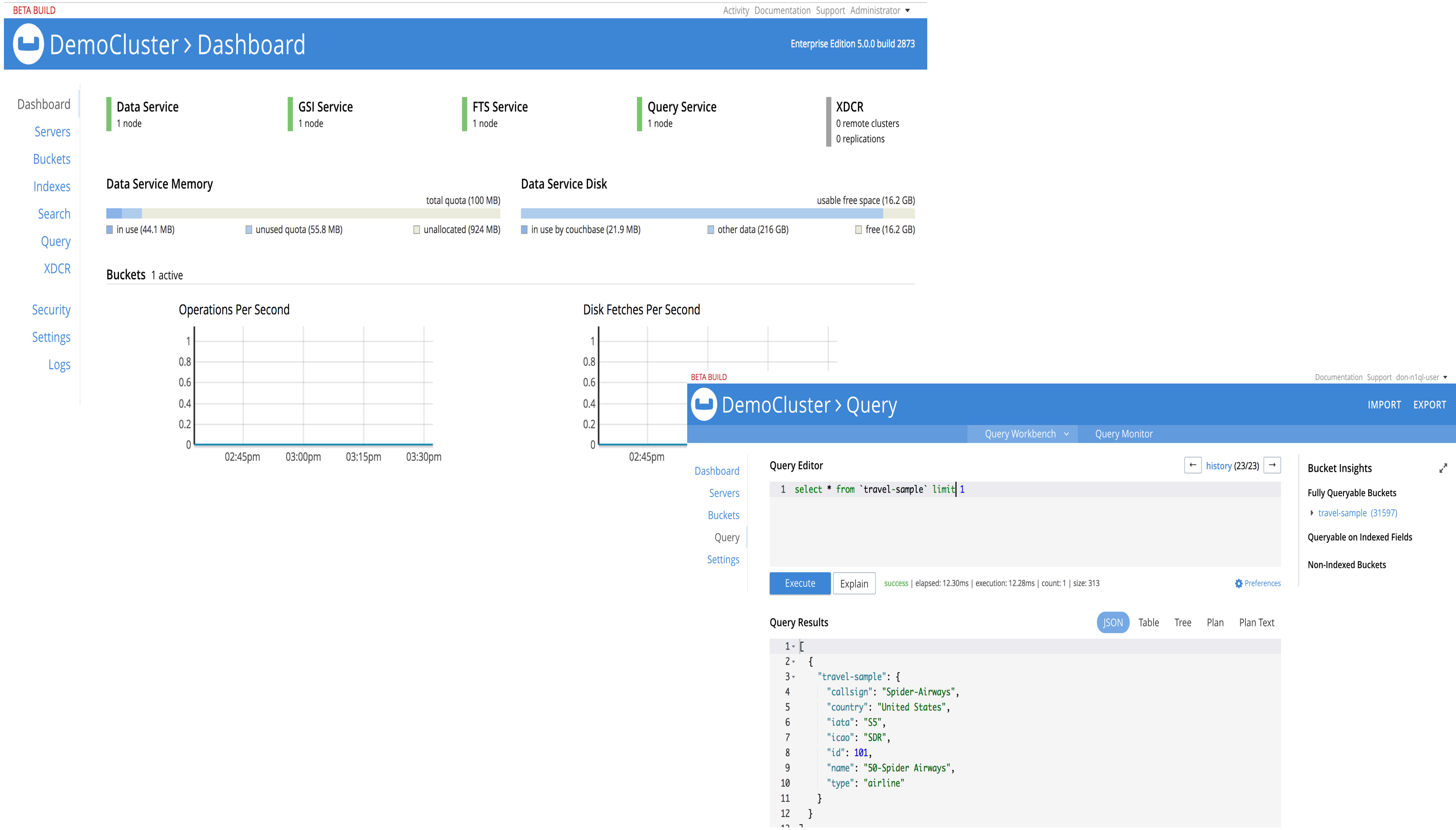
**Bucket Admin**

- Full Read/Write access over the bucket, and ability to change bucket settings

# Web Console For Administrators and Developers

## Who gets to log into web console ?

1. Administrators (Any administrator role)
2. Developers (Users who have one ore more query role)



Data Service
1 node

GSI Service
1 node

FTS Service
1 node

Query Service
1 node

XDCR
0 remote clusters
0 replications

Data Service Memory

total quota (100 MB)

in use (44.1 MB) unused quota (55.8 MB) unallocated (924 MB)

Data Service Disk

usable free space (16.2 GB)

in use by couchbase (21.9 MB) other data (216 GB) free (16.2 GB)

Buckets 1 active

Query Editor

← history (23/23) →

```
select * from `travel-sample` limit 1
```

success | elapsed: 12.30ms | execution: 12.28ms | count: 1 | size: 313

Bucket Insights

Fully Queryable Buckets

- travel-sample (31597)

Queryable on Indexed Fields

Non-Indexed Buckets

Query Results

```
[
  {
    "travel-sample": {
      "callsign": "Spider-Airways",
      "country": "United States",
      "iata": "S5",
      "icao": "SDR",
      "id": 101,
      "name": "50-Spider Airways",
      "type": "airline"
    }
  }
```

# Role Assignment – Using REST and CLI

## Using REST

```
curl -X PUT http://localhost:8091/settings/rbac/users/local/don-data-user
 -u Administrator:password -d "roles=data_reader[travel-sample]" -d
"password=donpassword”
```

## Using CLI

```
./couchbase-cli user-manage --set --rbac-username don-n1ql-user --rbac-
password donpassword --auth-domain local --roles "data_reader[*],
query_select[*]" -c http://localhost:8091 -u Administrator -p password
```

# GRANT /REVOKE statements in N1QL for RBAC

## GRANT ROLE

GRANT ROLE data_reader(`*`) to don

## REVOKE ROLE

REVOKE ROLE data_reader(`*`) from don

# System tables for RBAC

## system:applicable_roles (provides user-role mappings)

```
SELECT * FROM system:applicable_roles
WHERE bucket_name="travel-sample"
```

## system:user_info (provides full user information)

```
SELECT * FROM system:user_info
```

# 3 Encryption

# Encryption

## On-the-wire Encryption

- TLS between client and server
- TLS between datacenters using secure XDCR
- X.509 CA Certificates for trusted encryption between client and server
- Field-level Encryption

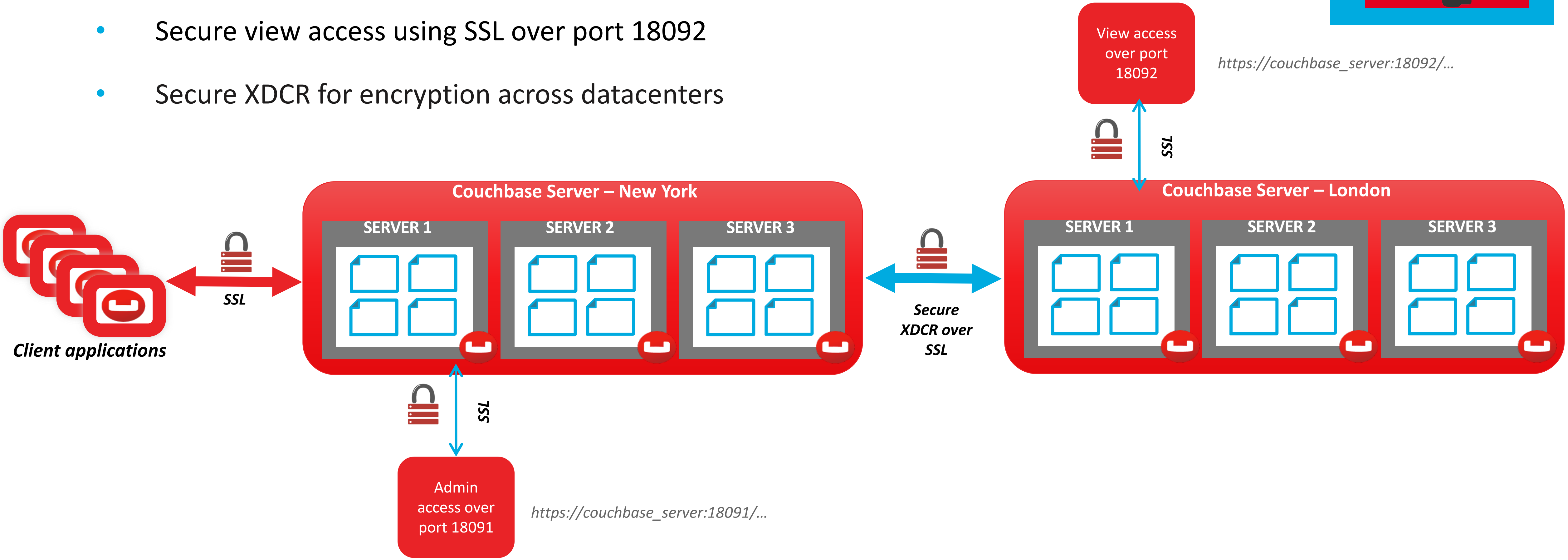
## On-Disk Encryption

- Volume and application level encryption through our trusted 3rd partners (LUKS, Vormetric, Protegrity,  SafeNet)
- FIPS 140-2 compliant
- Field-level Encryption

# Couchbase encryption overview (In Motion)

ENCRYPTION

- **Data-in-motion encryption**
  - Client-server communication can be encrypted using SSL
  - Secure admin access using SSL over port 18091
  - Secure view access using SSL over port 18092
  - Secure XDCR for encryption across datacenters

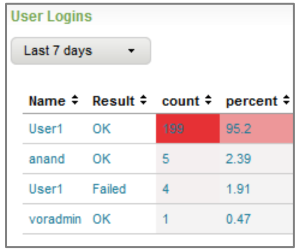


View access over port 18092
https://couchbase_server:18092/…
SSL
Couchbase Server – New York
SERVER 1
SERVER 2
SERVER 3
Couchbase Server – London
SERVER 1
SERVER 2
SERVER 3
SSL
Client applications
Secure XDCR over SSL
SSL
Admin access over port 18091
https://couchbase_server:18091/…

# Couchbase encryption overview

ENCRYPTION

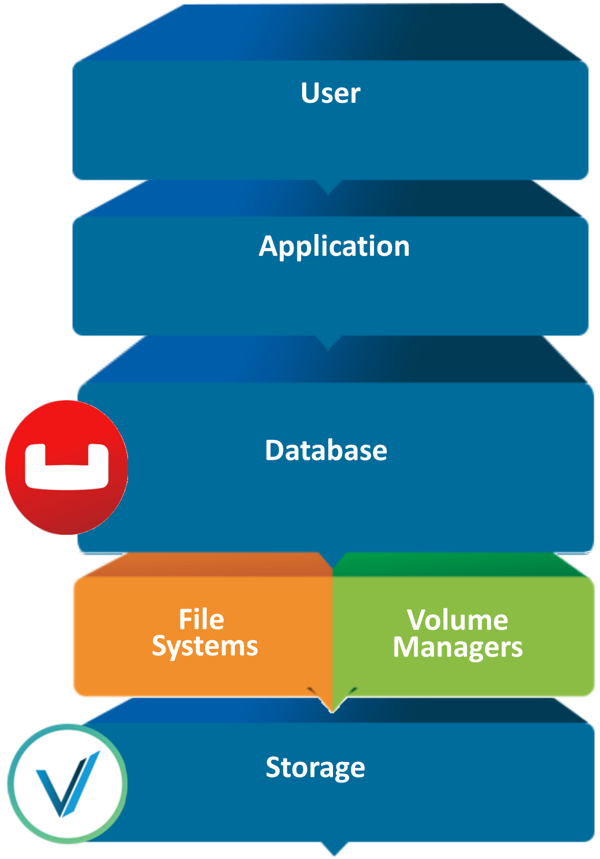
- Transparent data-at-rest encryption solution



User Logins

Last 7 days

| Name | Result | count | percent |
|---|---|---|---|
| User1 | OK | 199 | 95.2 |
| anand | OK | 5 | 2.39 |
| User1 | Failed | 4 | 1.91 |
| voradmin | OK | 1 | 0.47 |



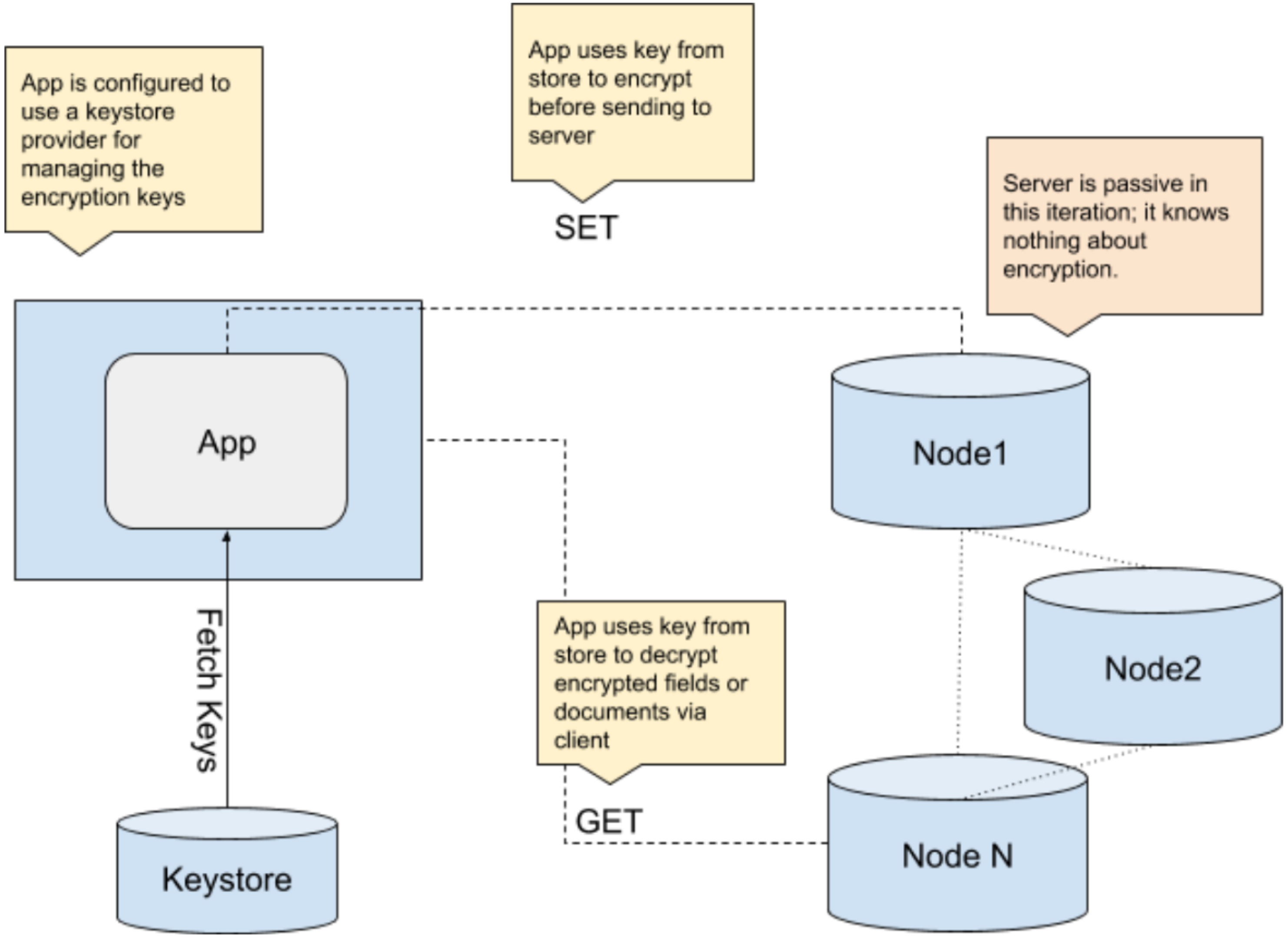
**Vormetric**
**Data Security Manager**
on Enterprise premise or in cloud
virtual or physical appliance

**Secure Personally Identifiable Information**

- User profile information
- Login Credentials
- IP Addresses

- Centrally manage keys and policy
- Virtual and physical appliance
- High-availability with cluster
- Multi-tenant and strong separation of duties
- Proven 10,000+ device and key management scale
- Web, CLI, API Interfaces
- FIPS 140-2 certified

# Field Level Encryption (since CB5.5)



App is configured to use a keystore provider for managing the encryption keys

App uses key from store to encrypt before sending to server

Server is passive in this iteration; it knows nothing about encryption.

App uses key from store to decrypt encrypted fields or documents via client

# Field Level Encryption: Example

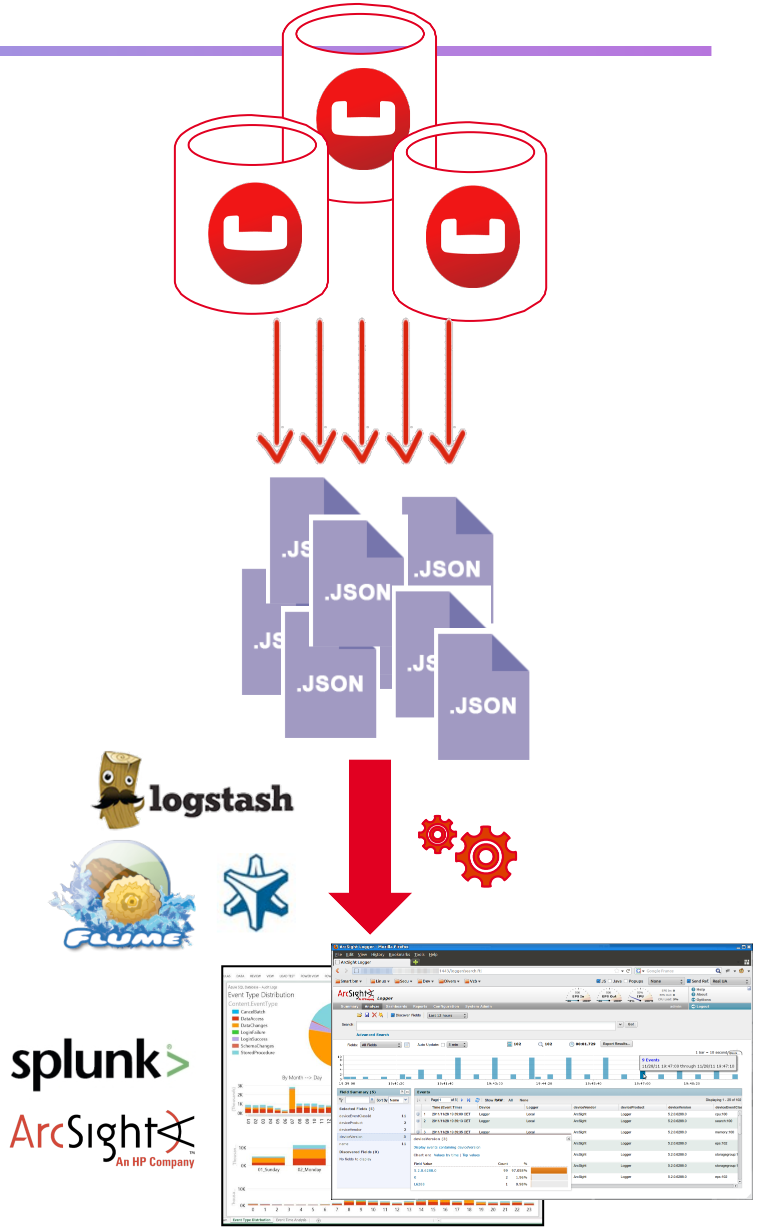
```
{
"message":"The old grey goose jumped over
the wrickety gate.",
“recipient”: “jeffry.morris@couchbase.com”
}
```

```
{
  "__crypt_message": {
    "alg": "RSA-2048-OAEP-SHA1",
    "kid": "MyPublicKeyName",
    "ciphertext":
"iX2MXbUlief8Xxk4DYysivEsUXeoiFBLkm4/EC7E9vRnGikDOiuaWl
lLTJU/oNKeVNlWPzfN6r/uLEpttp+BLC0DswdxLkA3ONeO85TDdHaHm
rJ3dJQ7qgDFe35K6MbTEPXE98f1wL2vOL70xJxW+3KsgdcYYYqg8VNw
2U9eKVC2lv4DS19l/r+6l+O8EGvBaa0FidezgF7CzgdXpGmG20cA0D8
yCmmGoW8oq7KWoq0PNaKsb9JOYfOYi13bxpPOIbyI003qLb5b7y1qVm
s8KDZ0+nk7Xnn5OYFmBHQDyJ39nuibEMKNMlA2ZNlCvfFqE1dU3iqqZ
YyS7OTukFBO2g=="
  },
“recipient”: “jeffry.morris@couchbase.com”
}
```

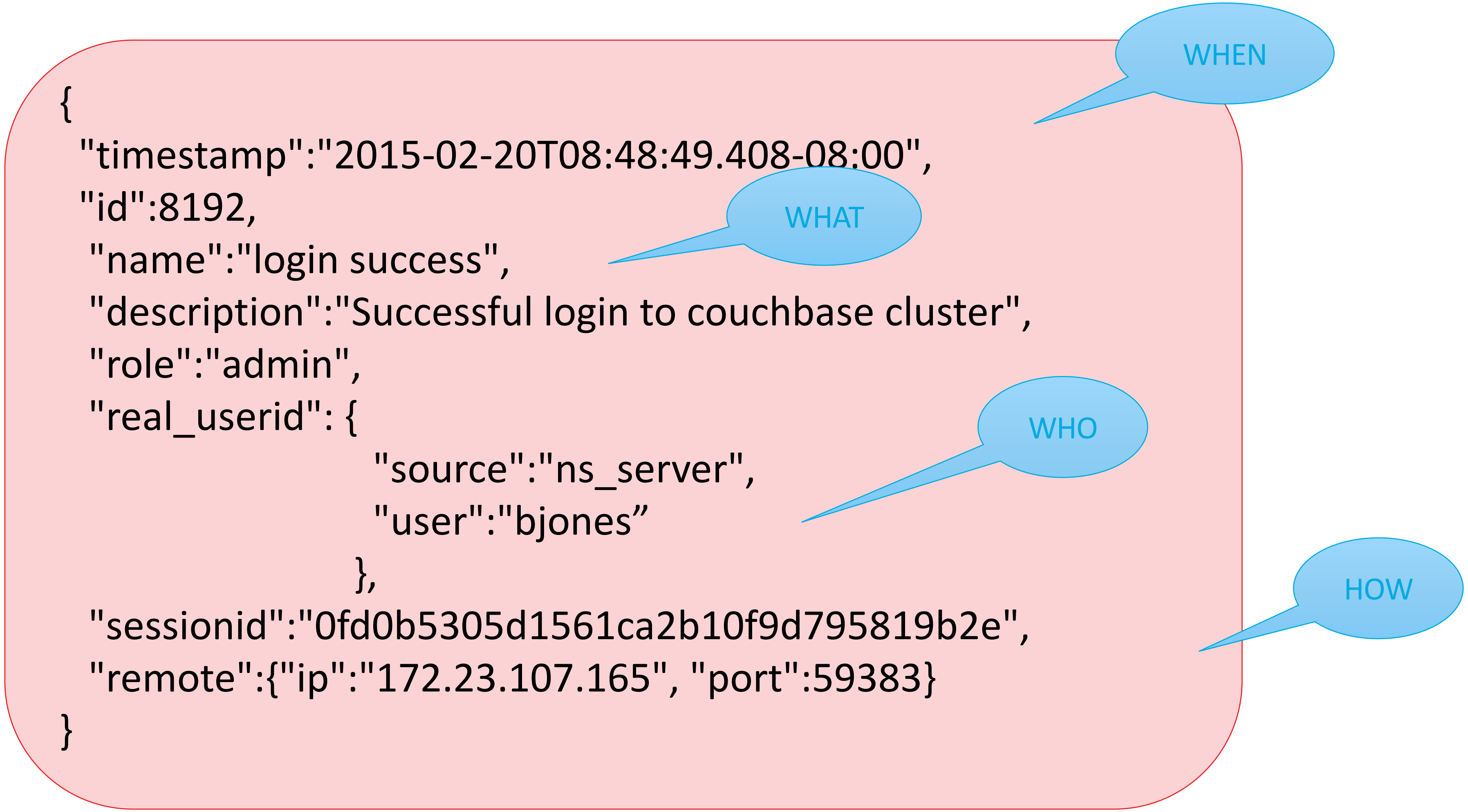
# 4 Auditing

# Admin Auditing in Couchbase

- **Rich audit events**
  - Over 25+ different, detailed admin audit events
  - Auditing for tools including backup
- **Configurable auditing**
  - Configurable file target
  - Support for time based log rotation and audit filtering
- **Easy integration**
  - JSON format allows for easy integration with downstream systems using flume, logstash, and syslogd



# Auditing a successful login



```
{
 "timestamp":"2015-02-20T08:48:49.408-08:00",
 "id":8192,
  "name":"login success",
  "description":"Successful login to couchbase cluster",
  "role":"admin",
  "real_userid": {
                        "source":"ns_server",
                        "user":"bjones”
                      },
  "sessionid":"0fd0b5305d1561ca2b10f9d795819b2e",
  "remote":{"ip":"172.23.107.165", "port":59383}
}
```

# Thank you

Couchbase