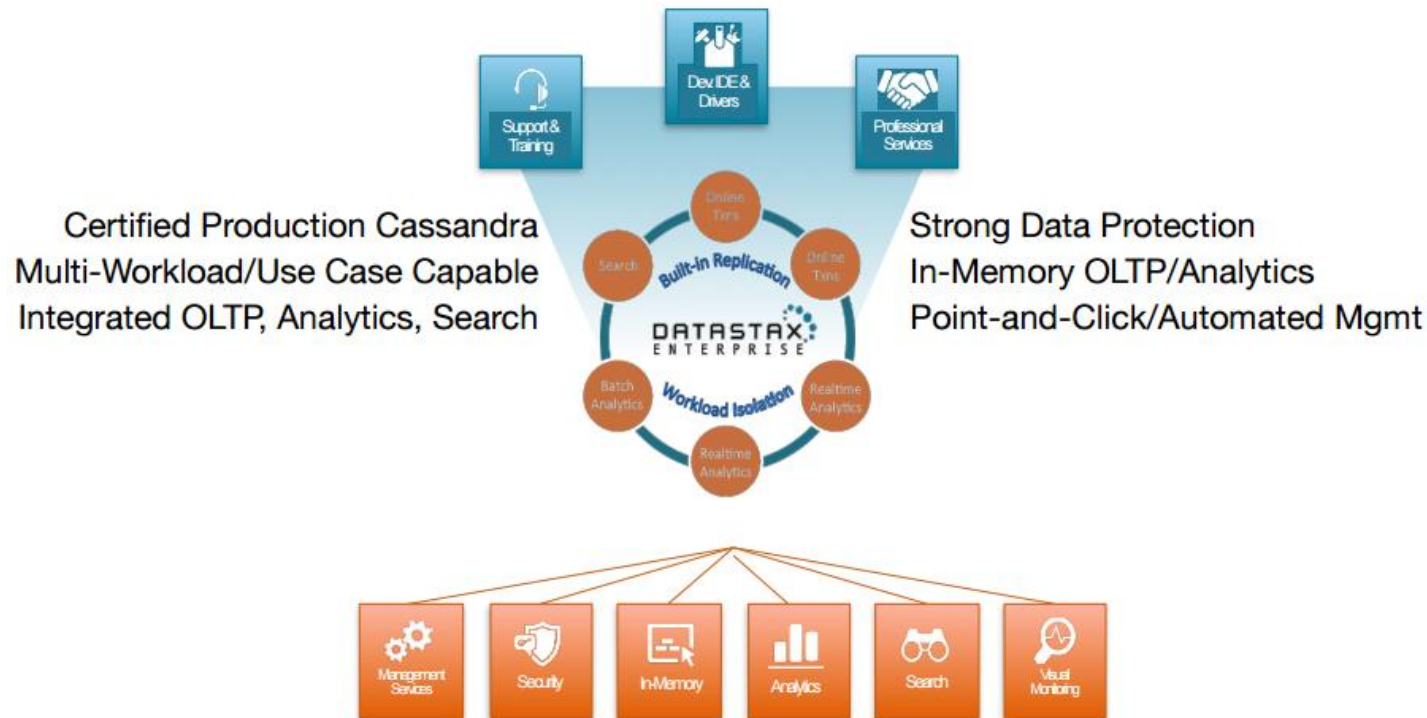


- Thingworx provides an option to choose a persistence provider
- Apache Cassandra designed to handle data workloads across multiple data centers with no single point of failure
- Customer industries:
 - Financial
 - eCommerce
 - Marketing
 - Recommendation Engines
 - Health Sciences
 - Fraud Detection
 - Sensor Data
 - Online Games

WHAT IS DATASTAX ENTERPRISE?

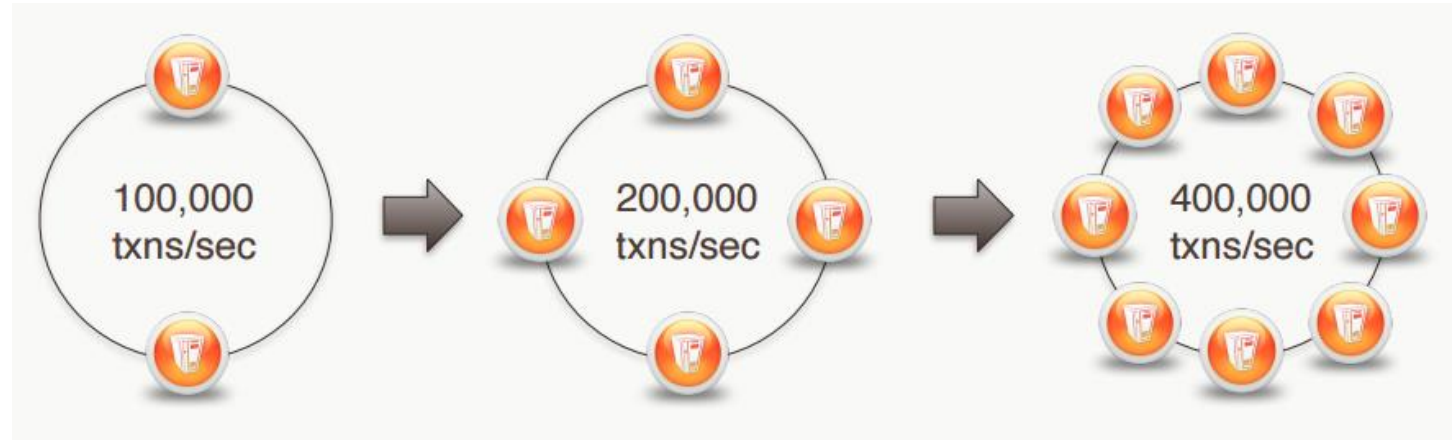
- DataStax delivers Apache Cassandra in a database platform purpose-built for the performance and availability demands of Web, Mobile, and IOT applications, giving enterprises a secure always-on database that remains operationally simple when scaled in a single datacenter or across multiple datacenters and clouds.



- CQL (Cassandra Query Language) is the primary API
- Clients that use the native driver also have access to various policies that enable the client to intelligently route requests as required.
- DataStax drivers: Java, Python, C#, C++, Ruby(much more to come and in the community)
- This includes:
 - Load Balancing
 - Data Centre Aware
 - Latency Aware
 - Token Aware
 - Reconnection policies
 - Retry policies
 - Downgrading Consistency
 - Plus others.. • <http://www.datastax.com/download/clientdrivers>

WHY USE DSE?

- Masterless architecture.
 - No Single Point of failure
 - No Hadoop complexity – every node is built the same
- Continuous availability.
- Multi-data center and cloud availability zone support.
- Flexible data model.
- Linear scale performance.
- Operationally simple.
- CQL – SQL-like language.



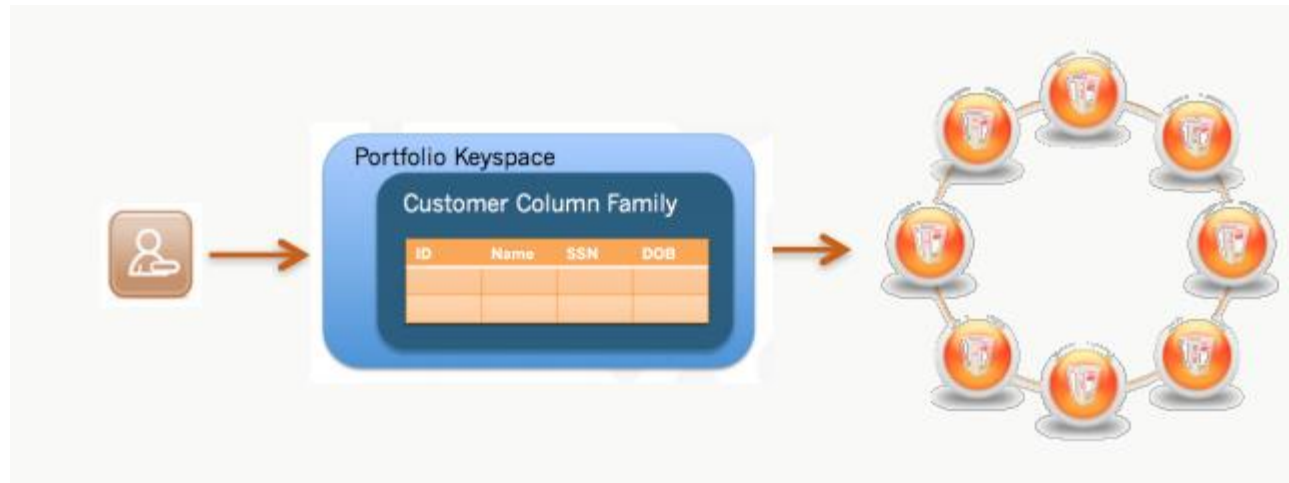
WHY USE DSE?

- **Elastic scalability**
- **Always on architecture**
- **Fast linear-scale performance**
- **Flexible data storage**
- **Easy data distribution**
- **Operational simplicity**
- **Transaction support**
- **Requires no new equipment**

CASSANDRA TERMINOLOGY

- Node (Vnode)
- Cluster
- Datacenter
- Partition
- Gossip
- Snitch
- Replication Factor
- Consistency

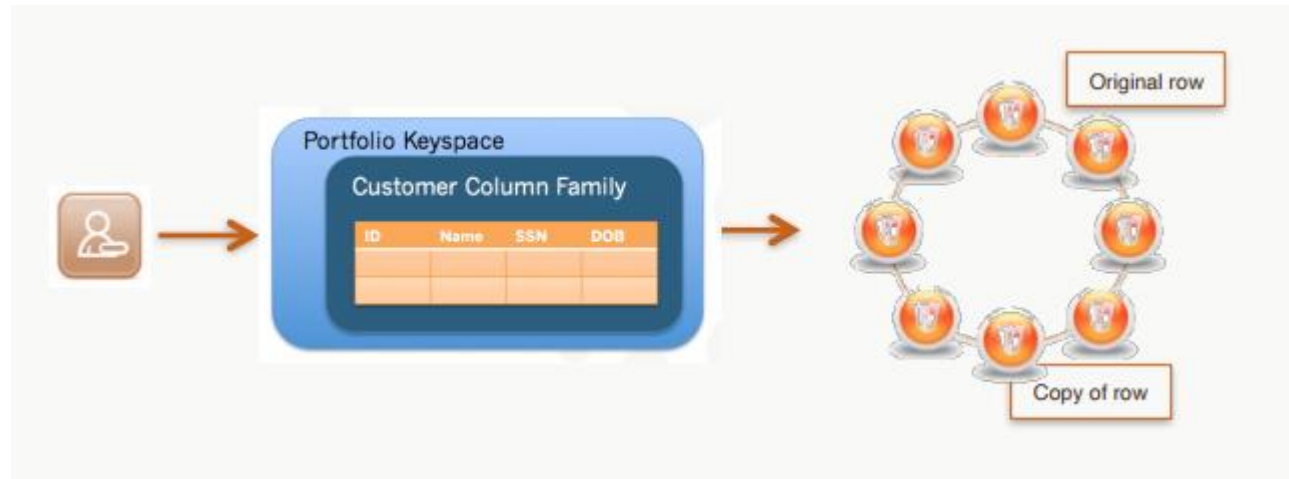
- Node: A computer server running Cassandra
- Cluster: A group of Cassandra nodes working together
 - Data is evenly distributed around the nodes
- Datacenter: A set of Cassandra clusters, logical, physical, cloud-based
- Partitioning: Random – default and recommended
 - Ordered Partitioning – stores column family row keys in sorted order across the nodes in a db cluster



- Each node “owns” a set of tokens
- A node’s token range is manually configured or randomly assigned when the node joins the cluster
- A partition key is defined for each table
- The partitioner applies a hash function to convert a partition key to a token. This determines which node(s) store that piece of data.
- By default, Cassandra uses the Murmur3Partitioner
 - MurmurHash function. This hashing function creates a 64-bit hash value of the row key. The possible range of hash values is from -263 to +263.

REPLICATION

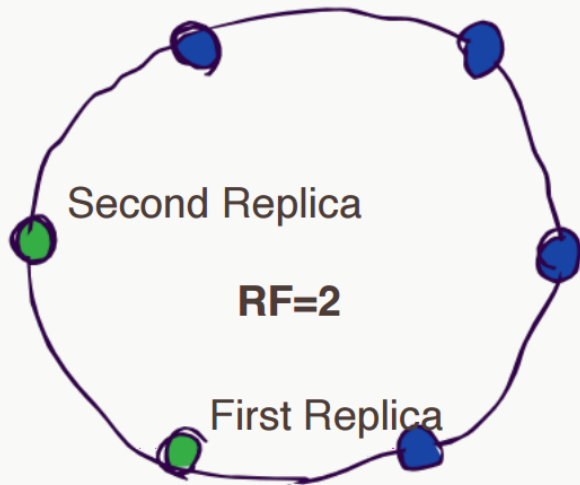
- Replication is controlled by what is called the replication factor. A replication factor of 1 means there is only one copy of a row in a cluster. A replication factor of 2 means there are two copies of a row stored in a cluster
- Replication is controlled at the keyspace level in Cassandra



REPLICATION

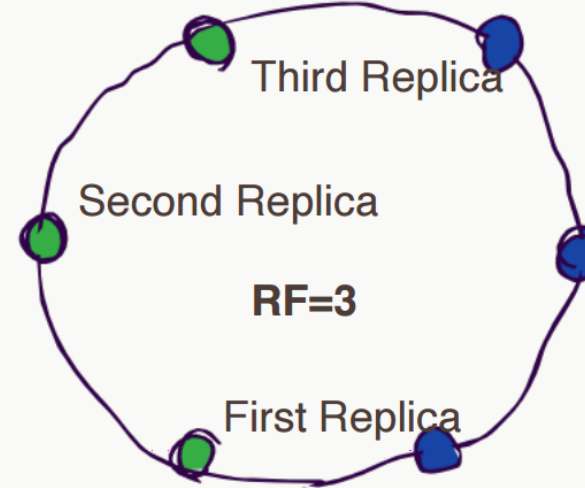
- Simple strategy

Simple Topology – Single Datacenter

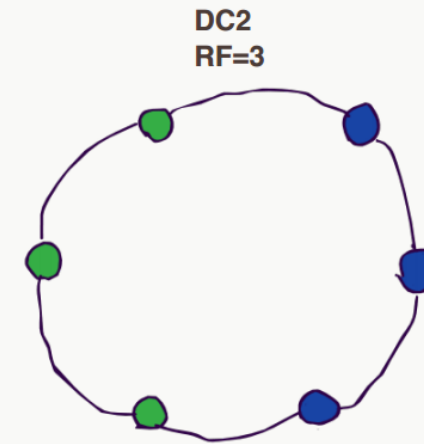
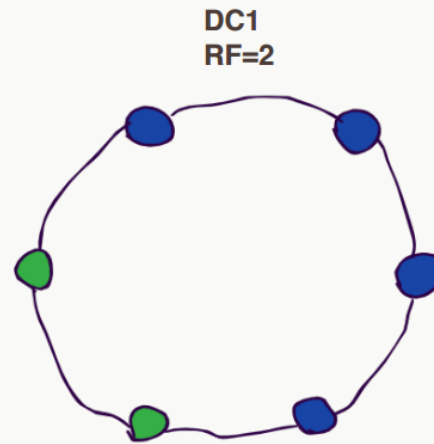


```
{ 'class' : 'SimpleStrategy', 'replication_factor' : 2 };
```

Simple Topology – Single Datacenter



```
{ 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
```



```
CREATE KEYSPACE Test  
WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 3 };
```

- A snitch determines which data centers and racks are written to and read from.
 - Snitches inform Cassandra about the network topology so that requests are routed efficiently and allows Cassandra to distribute replicas by grouping machines into data centers and racks.
- Cassandra does its best not to have more than one replica on the same rack.

SNITCH TYPE

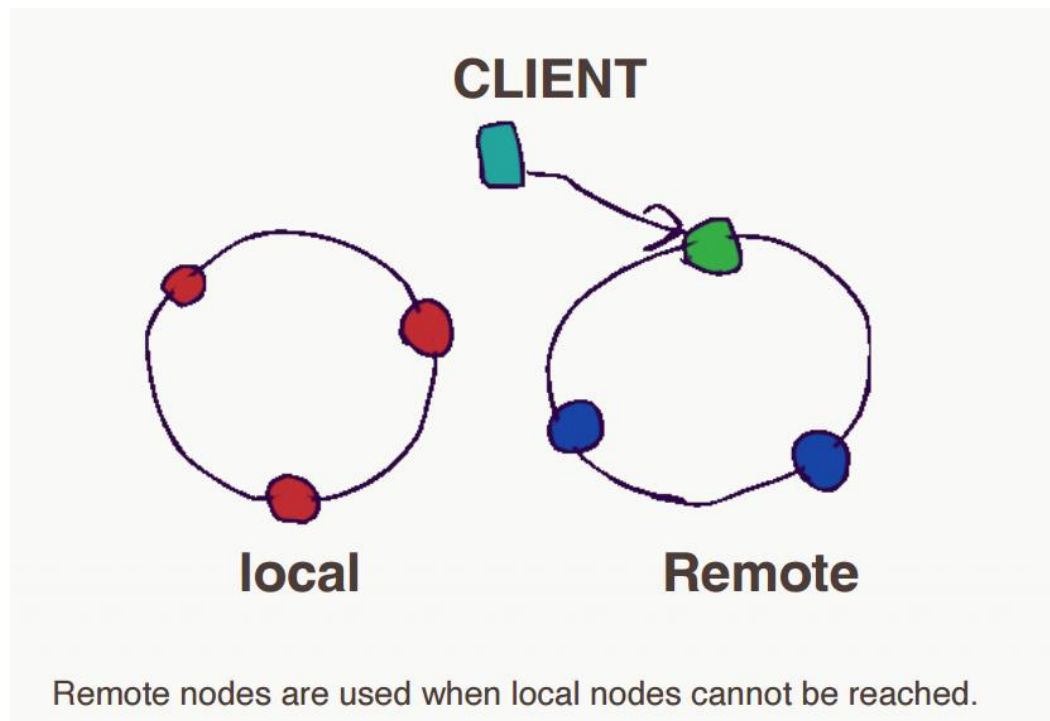
- SimpleSnitch
 - Single-data center deployments (or single-zone in public clouds)
- RackInferringSnitch
 - Determines the location of nodes by rack and data center corresponding to the IP addresses
- PropertyFileSnitch
 - User-defined description of the network details
 - cassandra-topology.properties file
- GossipingPropertyFileSnitch
 - Defines a local node's data center and rack
 - Uses gossip for propagating this information to other nodes
 - cassandra-rackdc.properties
- Amazon Snitches
 - EC2Snitch
 - EC2MultiRegionSnitch

GOSSIP = INTERNODE COMMUNICATIONS

- Gossip is a peer-to-peer communication protocol in which nodes periodically (every second) exchange information about themselves and about other nodes they know about.
- Cassandra uses gossip to discover location and state information about the other nodes participating in a Cassandra cluster

LOAD BALANCING

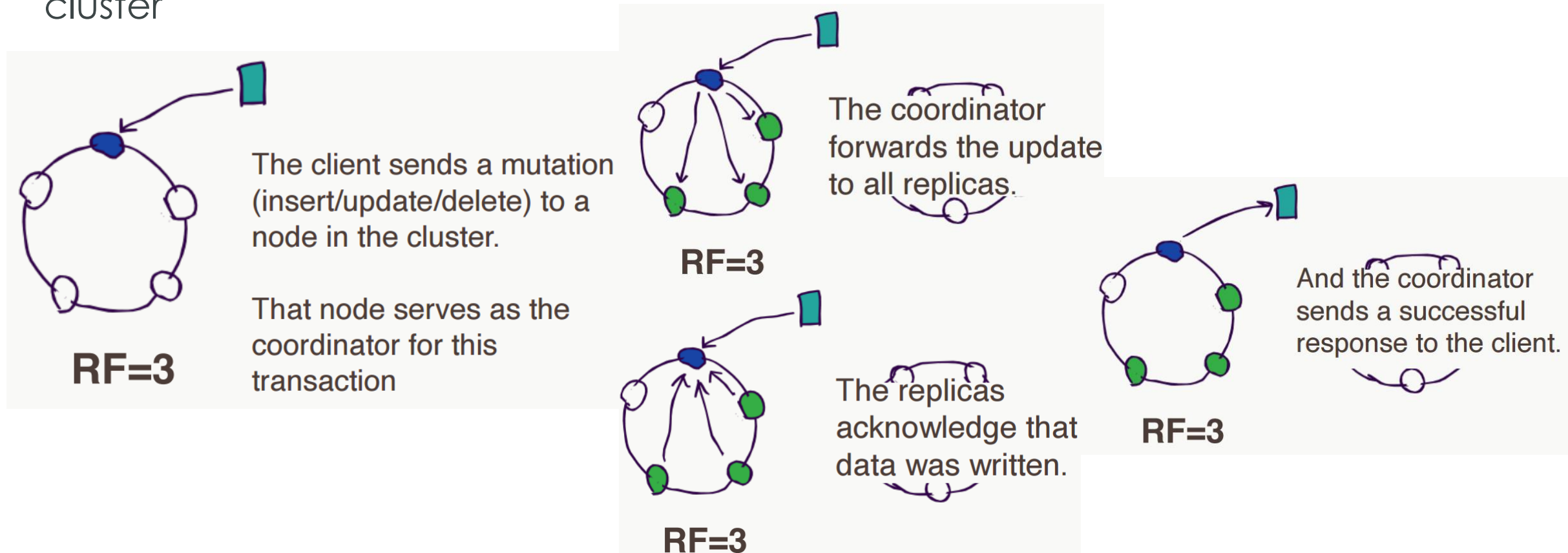
- Each node handles client requests, but the balancing policy is configurable
- Round Robin
- DC-Aware Round Robin
- Token-Aware



- Instead of each node owning a single token range, Vnodes divide each node into many ranges (256).
- Vnodes simplify many tasks in Cassandra:
 - You no longer have to calculate and assign tokens to each node.
 - Rebalancing a cluster is no longer necessary when adding or removing nodes.
 - Rebuilding a dead node is faster because it involves every other node in the cluster.
 - Improves the use of heterogeneous machines in a cluster. You can assign a proportional number of vnodes to smaller and larger machines.

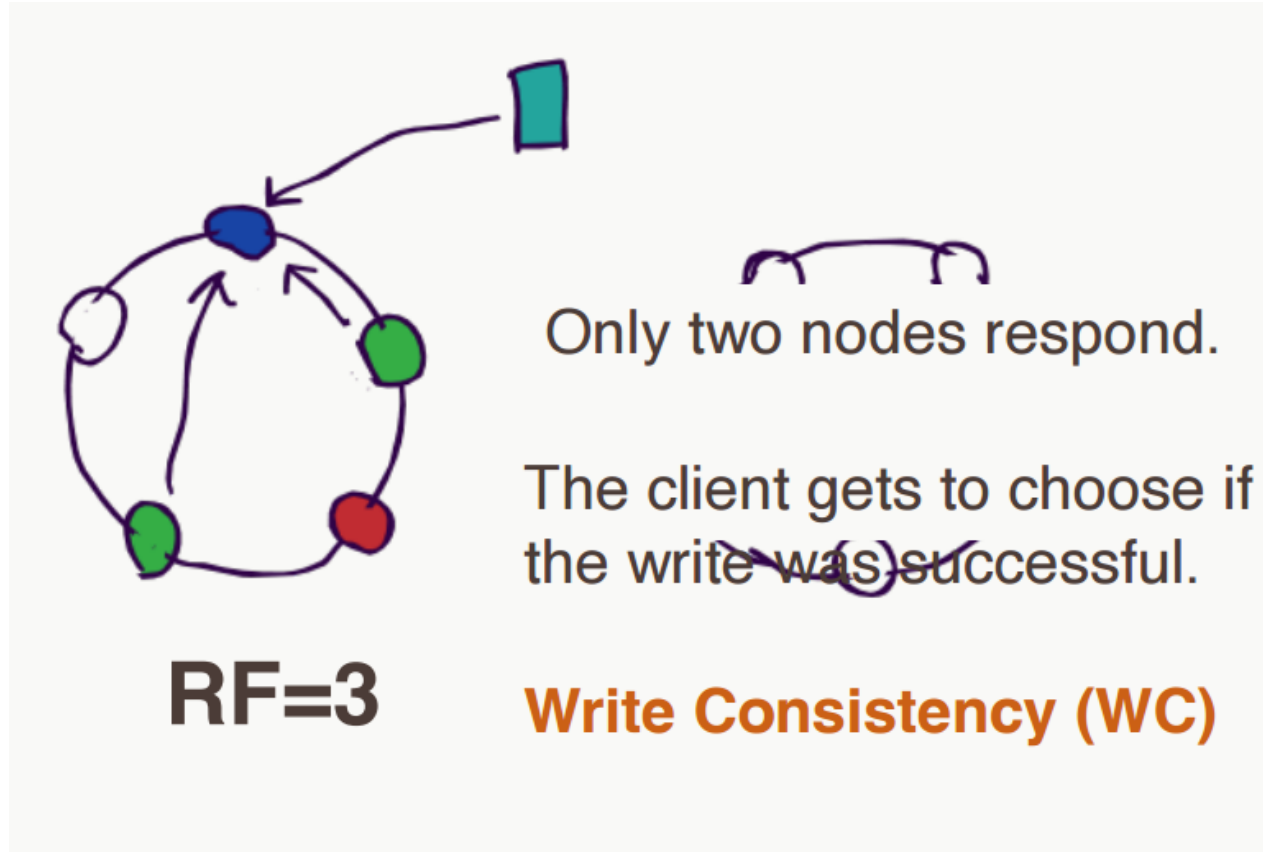
READING AND WRITING TO CASSANDRA NODES

- Cassandra has a 'location independence' architecture, which allows any user to connect to any node in any data center and read/write the data they need
- All writes being partitioned and replicated for them automatically throughout the cluster



WHAT IF A NODE IS DOWN?

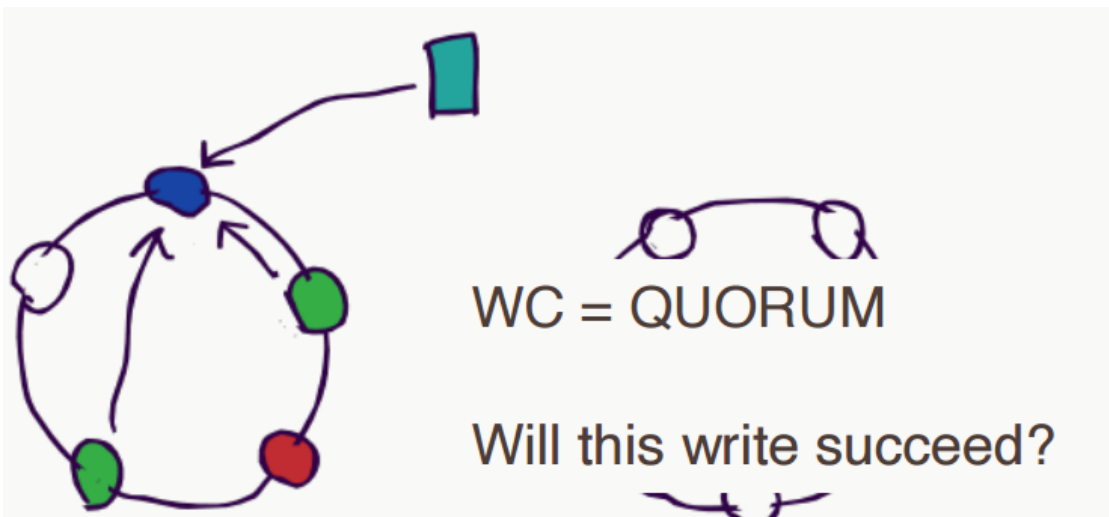
-



- Choose between strong and eventual consistency (one to all responding) depending on the need
- Can be done on a per-operation basis, and for both reads and writes
- Handles multi-data center operations
- Consistency:
 - ANY
 - Returns data from any of the replica.
 - QUORUM
 - Returns the most recent data from the majority of replicas.
 - LOCAL QUORUM
 - Returns the most recent data from the majority of local replicas.
 - ALL
 - Returns the most recent data from all replicas.
- Read on consistency levels and Quorum
[:http://docs.datastax.com/en/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html](http://docs.datastax.com/en/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html)

QUORUM MEANS $> 50\%$

What if two nodes are down?



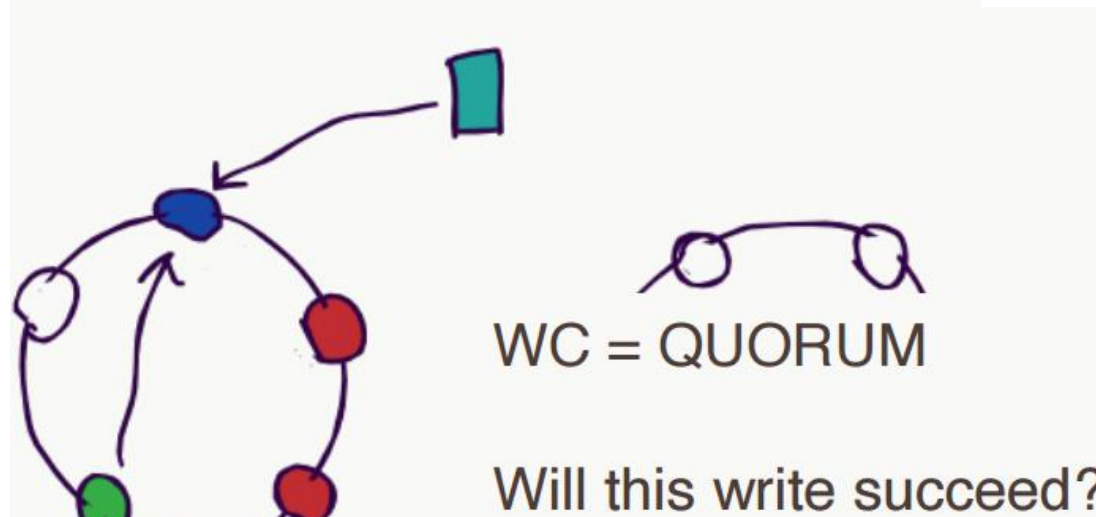
WC = QUORUM

Will this write succeed?

RF=3

YES!!

A majority of replicas received the mutation.



WC = QUORUM

Will this write succeed?

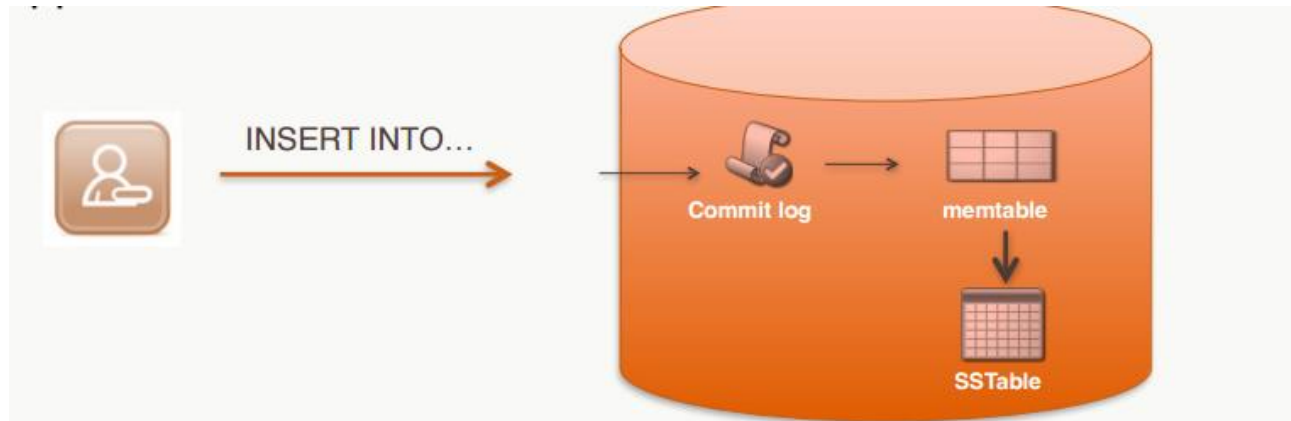
RF=3

NO

Failed to write a majority of replicas.

RAPID READ PROTECTION AND WRITES

- Cassandra performs only as many requests as necessary to meet the requested Consistency Level. Cassandra routes requests to the most-responsive replicas.
- If a replica doesn't respond quickly, Cassandra will try another node. This is known as an "eager retry"
- Data is first written to a commit log for durability. then written to a memtable in memory. Once the memtable becomes full, it is flushed to a sorted strings table(SSTable) Writes are atomic at the row level: all columns are written or updated, or none are. Note:RDBMS-styled transactions are not supported



LIMIT DATA PER NODE?

- Depends on the *rate of operations*.
- If replication factor is above 1 and consistency level is not ALL, other replicas will be able to respond quickly to read requests

MODELING A CLUSTER- NON GOALS

- Minimize the Number of Writes
- Minimize Data Duplication

MODELING A CLUSTER – BASIC GOALS

- Spread data evenly around the cluster
- Minimize the Number of Partitions Read
 - Why is this important?
 - Some notes on data limitations <http://wiki.apache.org/cassandra/LargeDataSetConsiderations>

FEATURES



Internal Authentication

Manages login IDs and passwords inside the database



Object Permission Management

controls who has access to what and who can do what in the database



Client to Node Encryption

protects data in flight to and from a database cluster

BENEFITS

- + Ensures only authorized users can access a database system using internal validation
- + Simple to implement and easy to understand
- + No learning curve from the relational world

- + Provides granular based control over who can add/change/delete/read data
- + Uses familiar GRANT/REVOKE from relational systems
- + No learning curve

- + Ensures data cannot be captured/stolen in route to a server
- + Data is safe both in flight from/to a database and on the database; complete coverage is ensured

FEATURES



External Authentication
uses external security software systems to control security



Transparent Data Encryption
encrypts data at rest



Data Auditing
provides trail of who did and looked at what/when

BENEFITS

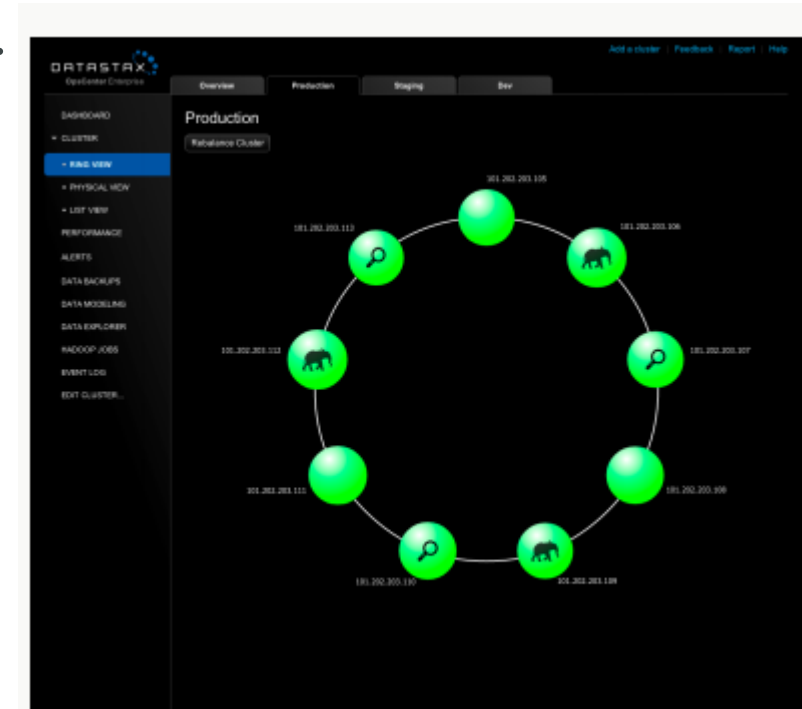
- + Only authorized users have access to a database system using external validation
- + Uses most trusted external security systems (Kerberos, LDAP, AD), mainstays in government and finance
- + Single sign on to all data domains

- + Protects sensitive data at rest from theft and from being read at the file system level
- + No changes needed at application level

- + Supplies admins with an audit trail of all accesses and changes
- + Granular control to audit only what's needed
- + Uses log4j interface to ensure performance and efficient audit operations



- Visual, browser-based user interface negates need to install client software
- Administration tasks carried out in point-and-click fashion
- Allows for visual rebalance of data across a cluster when new nodes are added
- Contains proactive alerts that warn of impending issues.
- Built-in external notification abilities
- Visually perform and schedule backup operations



A new, 10-node DSE cluster with OpsCenter running on AWS in **3 minutes...**

1

2

3

Done

Welcome to DataStax OpsCenter

Select one of the following options:

Create New Cluster

or

Use Existing Cluster

Create a new DataStax Enterprise or Cassandra cluster in EC2 or on existing hardware.

Manage an existing DataStax Enterprise or Cassandra cluster with OpsCenter.

Create Cluster

Cloud Local

Package

DataStax Community 1.1.2

Nodes (newline delimited)

```
ec2-50-18-12-217.us-west-1.compute.amazonaws.com
ec2-184-169-236-208.us-west-1.compute.amazonaws.com
ec2-184-169-254-173.us-west-1.compute.amazonaws.com
ec2-184-169-192-72.us-west-1.compute.amazonaws.com
ec2-204-236-145-114.us-west-1.compute.amazonaws.com
```

Node Credentials (sudo)

username

Password

Private SSH Key (optional)

Cluster name

Test Cluster

Build Cluster

View Options

Cancel

DATASTAX
OpsCenter Enterprise

BUILD

DASHBOARD

CLUSTER

PERFORMANCE

ALERTS

SCHEMA

DATA BACKUPS

DATA EXPLORER

EVENT LOG

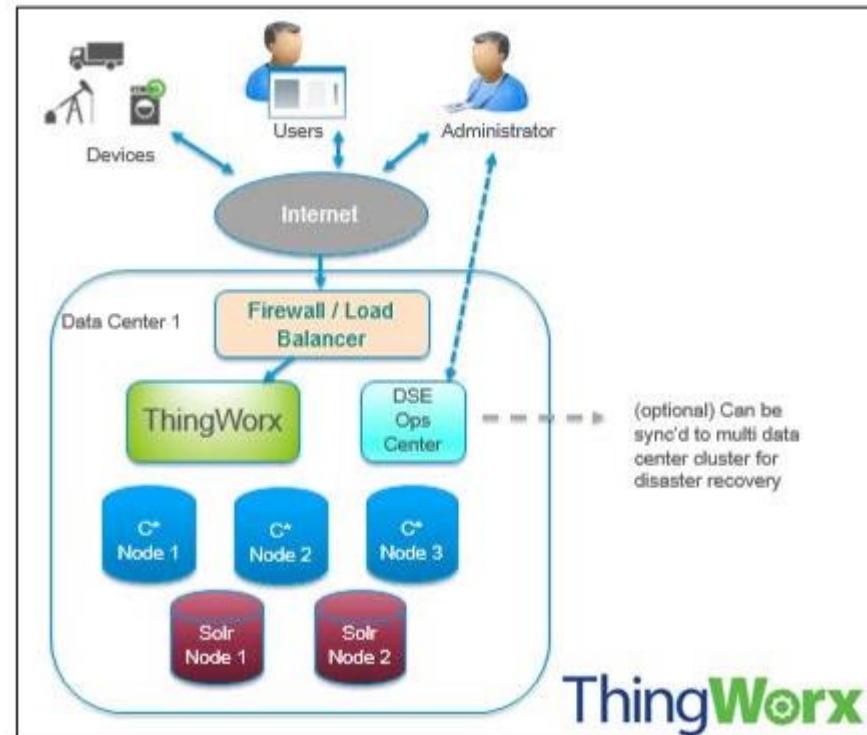
EOF CLUSTER

Build in Progress

Agent Connection Successful

Address	Stages	Status
ec2-184-169-254-173.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-50-18-12-217.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-169-192-72.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-169-243-17.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-169-236-208.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-204-236-145-114.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-72-4-106.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-169-258-112.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-50-18-72-176.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful
ec2-184-169-204-255.us-west-1.compute.amazonaws.com	●●●●●	Agent Connection Successful

- NOTE: ThingWorx 6.0 or later is required for this feature.
- DSE and ThingWorx Starting Landscape
 - The starting landscape for a typical implementation of DSE and ThingWorx is shown below.



HIGH-LEVEL PROCESS STEPS FOR DSE IMPLEMENTATION



The high level steps to implement DSE are as follows and are described in detail in our documentation:

1. Determine if DSE is the right solution for your data. Refer to the sizing and planning sections.
2. Register and install DSE. This is all performed independently of the ThingWorx Platform.
3. Import the DSE persistence provider extension into ThingWorx. NOTE: Contact ThingWorx Technical Support to obtain this extension.
4. Create a persistence provider instance in ThingWorx that will connect the Cassandra data store.
5. Configure the settings for the persistence provider in ThingWorx.
6. If necessary, migrate entities and data.
7. Monitor and maintain your DSE implementation. Best practices for creating a successful maintenance plan are described in documentation/will be mentioned further.

BENEFITS OF USING DSE AS THE PERSISTENCE PROVIDER



- Higher rate of ingestion of data
- Can have more than one data repository for runtime data
- Elastic scaling properties.
- Separates data processes from Platform processes
- Cloud-friendly architecture

- Start with understanding its architecture and specifically, the differences compared to regular Relational Databases.
- Free online courses offered by DataStax Academy:
 - <https://academy.datastax.com/courses/understanding-cassandra-architecture>
 - <https://academy.datastax.com/courses/installing-and-configuring-cassandra>
- The following section will guide you through some of the specifics:
 - http://datastax.com/documentation/cassandra/2.0/cassandra/architecture/architecturePlanningAbout_c.html

- **Disk Sizing**

- 18 to 20 bytes of disk space per message after compression to store integer property.
- RF3, it will require total 60 bytes to store a single message.
- Inserting 1 million messages per second, db will use about 60mb every second.
- With the TTL of 30 days, the total number comes to around 150TB (60mb * 86400sec * 30days)
- Have to provision double the disk space (300TB) for Cassandra compactions to work efficiently.

DATA DISTRIBUTION/TESTING SETTINGS

- 1 million things with 1 property each
- Value Stream source buckets count 1000
- Value Stream property buckets count 1000
- Cassandra cluster size 30 nodes
- Total throughput 300,000 requests/sec with 1 day TTL and 0 gc_grace_period

- Results:
- Node with highest load was receiving 12k requests/sec
- Node with lowest load was receiving 9k requests/sec

Sample Standard Deviation, s	0.85354170857426
Variance (Sample Standard), s^2	0.72853344827586
Population Standard Deviation, σ	0.83919544803341
Variance (Population Standard), σ^2	0.704249
Total Numbers, N	30
Sum:	308.67
Mean (Average):	10.289
Standard Error of the Mean ($SE_{\bar{x}}$):	0.15583468251921

Confidence Interval Approximations, If sampling distribution of the mean follows normal distribution

Confidence Level	Range
68.3%, $SE_{\bar{x}}$	10.133165317481 - 10.444834682519
90%, $1.645SE_{\bar{x}}$	10.032651947256 - 10.545348052744
95%, $1.960SE_{\bar{x}}$	9.9835640222624 - 10.594435977738
99%, $2.576SE_{\bar{x}}$	9.8875698578305 - 10.690430142169
99.9%, $3.291SE_{\bar{x}}$	9.7761480598293 - 10.801851940171
99.99%, $3.891SE_{\bar{x}}$	9.6826472503178 - 10.895352749682
99.999%, $4.417SE_{\bar{x}}$	9.6006782073127 - 10.977321792687
99.9999%, $4.892SE_{\bar{x}}$	9.526656733116 - 11.051343266884

THINGWORX DSE DEPLOYMENT

+ New Entity | Import/Export | Monitoring | Help | Learning Connector | New Comp

IMPORT

- From File
- From Thingworx Storage
- Source Control Entities

EXPORT

- To File
- To ThingworxStorage
- Source Control Entities

EXTENSIONS

- Import
- Manage

Type	Modified
Thing	2017-08-03
Thing	2017-08-03
Mashup	2017-08-03
Thing	2017-08-03
Thing Template	2017-08-03
Model Tag Vocabulary	2017-08-03

Import Extensions

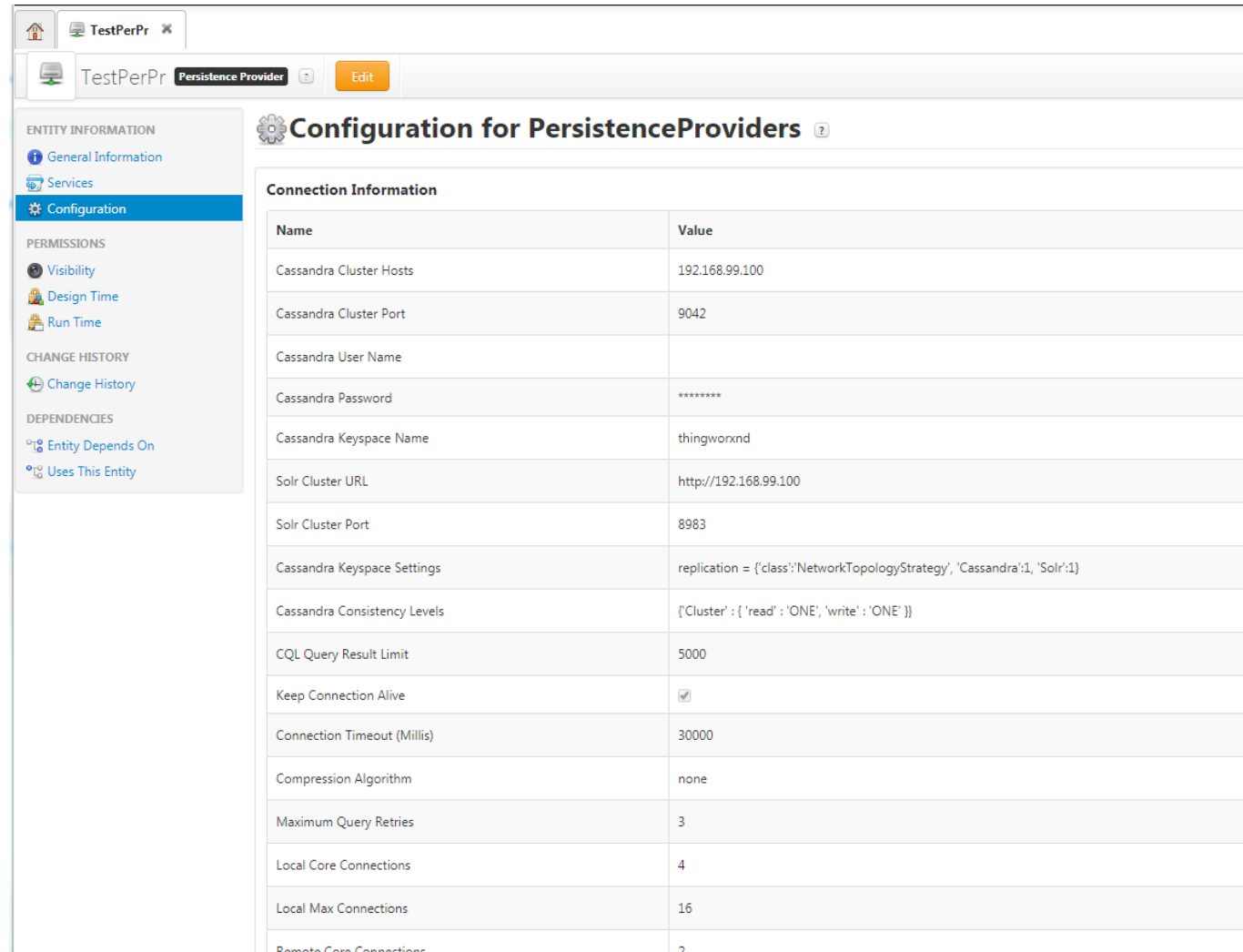
Choose File DsePersistenceProvider_ExtensionPackage.zip

Validation Results

- ✓ DsePersistenceProvider_ExtensionPackage:1.1.2

Installation Results

- ✓ DsePersistenceProvider_ExtensionPackage:1.1.2



The screenshot shows a web application interface for configuring a persistence provider. The browser tab is titled 'TestPerPr Persistence Provider' with an 'Edit' button. The main heading is 'Configuration for PersistenceProviders'. A left-hand navigation menu includes sections for 'ENTITY INFORMATION' (General Information, Services, Configuration), 'PERMISSIONS' (Visibility, Design Time, Run Time), 'CHANGE HISTORY' (Change History), and 'DEPENDENCIES' (Entity Depends On, Uses This Entity). The 'Configuration' section is active, displaying a table of configuration parameters.

Name	Value
Cassandra Cluster Hosts	192.168.99.100
Cassandra Cluster Port	9042
Cassandra User Name	
Cassandra Password	*****
Cassandra Keyspace Name	thingworxnd
Solr Cluster URL	http://192.168.99.100
Solr Cluster Port	8983
Cassandra Keyspace Settings	replication = { 'class': 'NetworkTopologyStrategy', 'Cassandra': 1, 'Solr': 1 }
Cassandra Consistency Levels	{ 'Cluster' : { 'read' : 'ONE', 'write' : 'ONE' } }
CQL Query Result Limit	5000
Keep Connection Alive	<input checked="" type="checkbox"/>
Connection Timeout (Millis)	30000
Compression Algorithm	none
Maximum Query Retries	3
Local Core Connections	4
Local Max Connections	16
Remote Core Connections	2

SELECT PERSISTENCE PROVIDER

The screenshot shows a software interface with a search results dialog box. The dialog box is titled "Search Results" and contains the following elements:

- Actions:** A button labeled "+ Persistence Provider".
- Filters:** "All 2" (selected), "Recent 1".
- TYPES:** "PersistenceProviders 2".
- Results:** A list of providers: "TestPerPr" (selected) and "ThingworxPersistenceProvider".

The background interface shows a "New Thing" form with a "Persistence Provider" field. A search bar at the bottom of this field is labeled "Search Persistence Providers".

ACCESS OPSCENTER



localhost:8888/opscenter/index.html

Starting with version 6.0, OpsCenter will only be compatible with DataStax Enterprise (DSE) clusters. For more information, please see [OpsCenter Policy Changes](#). [Dismiss](#)

OpsCenter 5.2.4 5.2.5, 6.1.2 available X NEW CLUSTER ALERTS 0 SETTINGS HELP

Test Cluster: Dashboard

Cassandra 2.2.8 All agents connected Cluster Actions

Default

2017-08-08 11:10 AM to 2017-08-08 11:30 AM Update Current

Graph Scale 20m Hour Day Week Month Add Graph Add Widget

Cluster Health

Datacenter datacenter1

1 0 0 0

Storage Capacity

Used: 122 GB Free: 28 GB Total: 150 GB
1 of 1 nodes

Write Requests

Cluster

8/sec
6/sec
4/sec
2/sec
0/sec

11:10AM 11:20AM

Cluster (Total)

Write Request Latency

Cluster

NO DATA

11:10AM 11:15AM 11:20AM 11:25AM

OS: Disk Utilization

Cluster

NO DATA

11:10AM 11:15AM 11:20AM 11:25AM

OS: Load

Cluster

0.20
0.15
0.10
0.05
0

11:10AM 11:15AM 11:20AM 11:25AM

Cluster (Avg)

TEST CASSANDRA ON WINDOWS WITHOUT COMMUNITY EDITION



- Docker toolbox
- Note IP address when docker starts up, can also "inspect" later
- `docker run --name some-cassandra -d cassandra:tag`
- `docker pull cassandra`
`run --name cassandra -p 9042:9042 -p 9160:9160 -d cassandra`
- https://hub.docker.com/_/cassandra/