

STORE & ACCESS YOUR DATA SECURELY WITH RED HAT JBOSS DATA GRID

Tristan Tarrant / ttarrant@redhat.com / [@tristantarrant](https://twitter.com/tristantarrant)

Divya Mehra / dmehra@redhat.com

About Tristan



Infinispan Project Lead

Open Source hacker since 1993

@ Red Hat since 2011

About Divya



JBoss Data Grid Product Manager
@ Red Hat since 2012

AGENDA

- JBoss Data Grid overview
- Cluster authentication & authorization
- Remote authentication
- Cache authorization
- Auditing accesses
- Directory integration
- Other security aspects
- Roadmap

Red Hat JBoss Data Grid Overview



RED HAT JBOSS DATA GRID

A distributed, in-memory NoSQL datastore

HIGH PERFORMANCE AND SCALABILITY

- In-memory access to large data-sets
- High availability, easy scale out

POLYGLOT

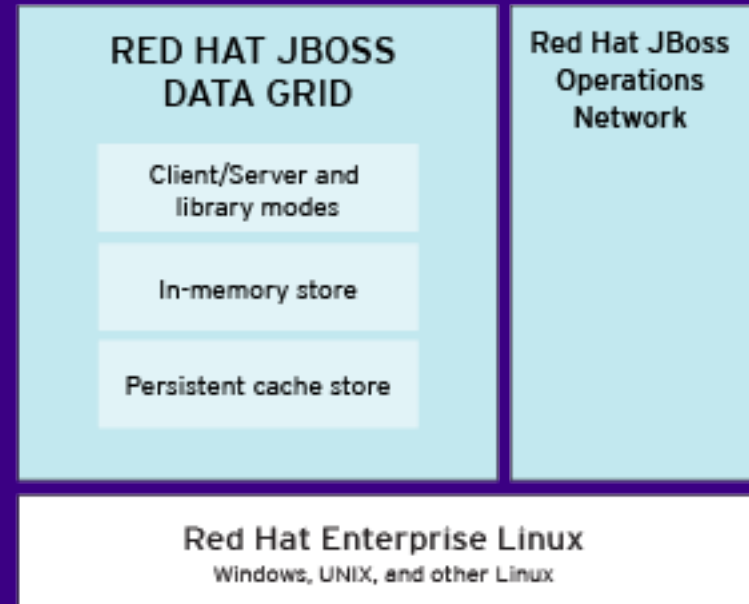
- Java, C++, .NET Hot Rod clients
- REST and memcached protocols available for other languages

CERTIFIED INTEGRATION WITH OTHER JBOSS PRODUCTS

- JBoss EAP, JBoss Fuse, JBoss Data Virtualization, JBoss Web Server

FULLY OPEN SOURCE

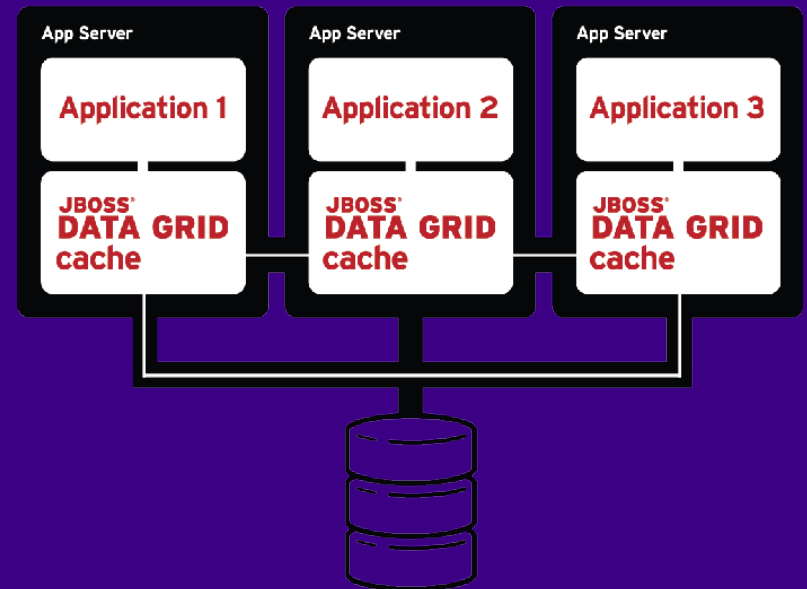
- Based on popular Infinispan project



DEPLOYMENT MODES

Library mode: Embedded Cache

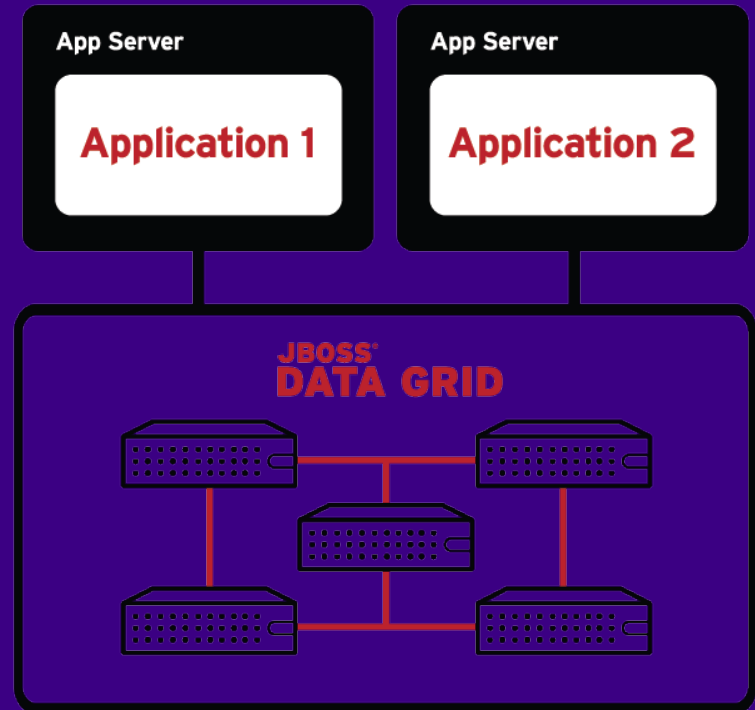
- Clustered JDG caches share heap with applications
 - Data grid scales with the application tier
- Application accesses a cache entry, regardless of whether it is present on locally or on a remote node



DEPLOYMENT MODES

Client/Server mode: Remote Cache

- Applications communicate with JDG server via protocols
 - Hot Rod
 - REST
 - Memcached
- Application accesses a cache entry, regardless of whether it is present on locally or on a remote node



CLIENT AND SERVER

Multiple access protocols

Protocol	Format	Type	Smart?	Balancing / failover
REST	text	any	no	external
Memcached	text	any	no	pre-defined
Hot Rod	binary	Java/C++/C#	yes	auto/dynamic

Hot Rod: Native TCP client/server protocol with rich functionality

- Hashing and topology aware
- Failover during topology changes
- Smart request routing in partitioned or distributed server clusters

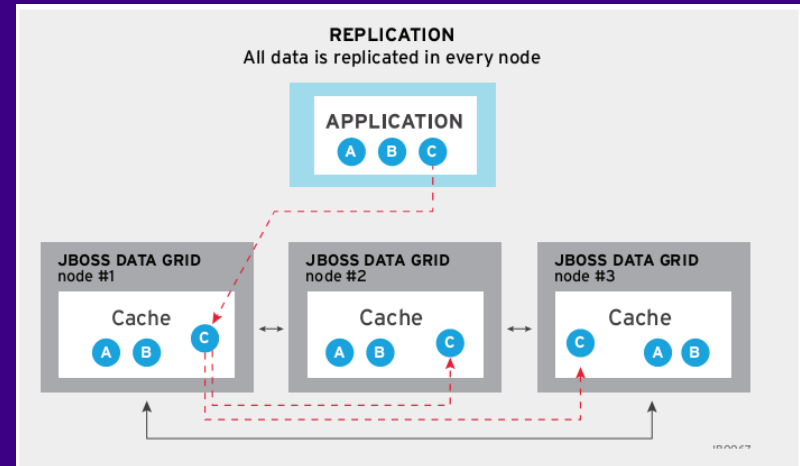
ARCHITECTURE

Replicated cache

- Replicate the (key/value) entry to each node of cluster
- Local reads
- Writes become slower with increasing number of nodes
- Data limited to a single JVM heap size

Ideal for:

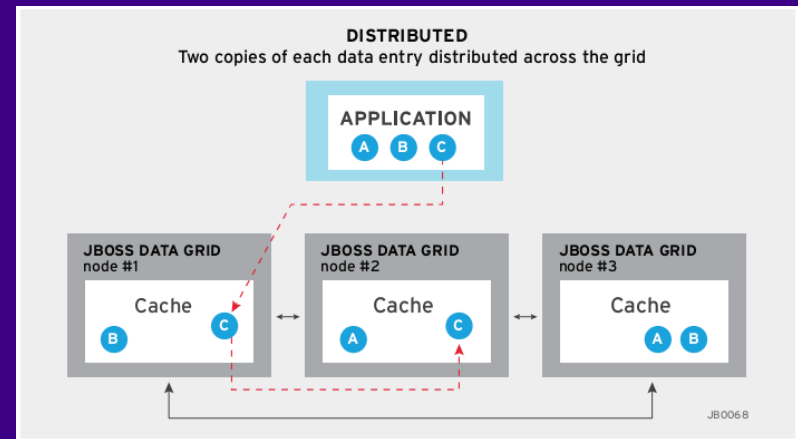
- Small, fixed datasets
- Highest read performance (local reads)



ARCHITECTURE

Distributed cache

- High performance + high scalability
- Typically maintain 2 or 3 copies of each entry on separate nodes
- Server hinting allows nodes on separate physical machines



SECURITY IN JDG

Design considerations

- Use case: Store personally identifiable information or other sensitive data in Jboss Data Grid
- Works in both deployment modes
 - Embedded
 - Client-Server (Hot Rod)
- Integrates with Directory service for identity information
 - MS ActiveDirectory
 - LDAP Server

SECURITY IN JDG

Design considerations

- Use popular protocols
 - X.509 certificates
 - TLS/SSL for encryption
 - Kerberos
- Use standard Java security frameworks/libraries
 - Simple Authentication and Security Layer (SASL)
 - Java Authentication and Authorization Service (JAAS)
 - Java Secure Sockets Extension (JSSE)
 - Java Cryptographic Architecture (JCA)



CLUSTER AUTHENTICATION AND AUTHORIZATION

JDG AND THE NETWORK

- Clustering network [LAN]
- Remote client protocols (HotRod, REST, Memcached) [LAN, WAN, Internet]
- Cross-site replication [WAN, Internet]

JDG CLUSTERING

The transport

- Based on JGroups
- Multiprotocol: UDP + TCP
- Discovery: Multicast + Unicast + S3 + Google + etc
- Geographical replication
- Configurable protocol stack

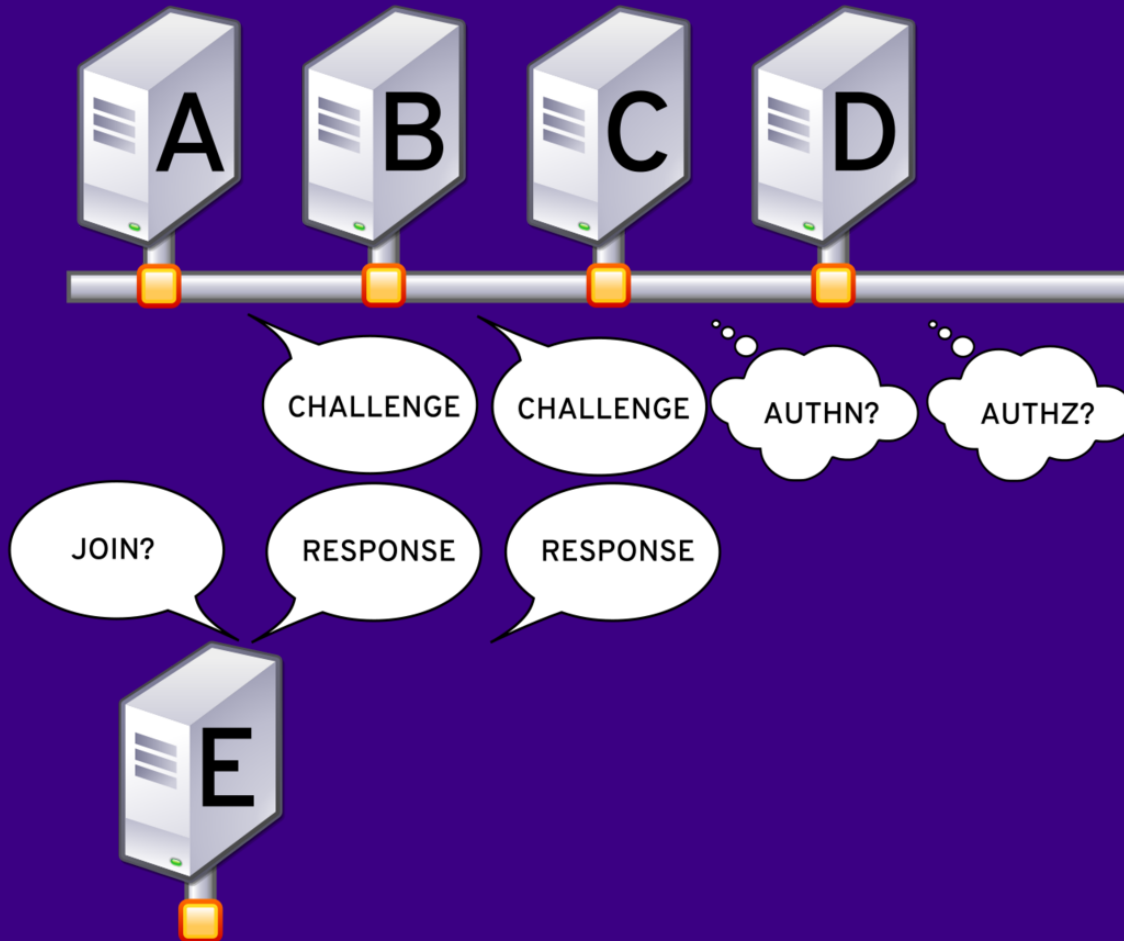
JGROUPS PROTOCOL STACK

- UDP or TCP Network protocol
- PING Discovery protocol
- MERGE Partition handling
- FD Failure detection
- UNICAST and NAKACK Message reliability
- SASL and AUTH Authentication
- GMS Membership
- ENCRYPT Encryption
- FC Flow control
- FRAG Fragmentation of large messages

JGROUPS AUTHENTICATION USING SASL

- Happens before group membership (GMS)
- Listens for JOIN/MERGE requests
- Leverages SASL challenge/response mechs (RFC-4222)
 - PLAIN
 - DIGEST-MD5
 - GSSAPI aka Kerberos
- User-provided *javax.security.auth.CallbackHandler*

SASL Cluster authentication



JAAS CALLBACKHANDLERS: BLACK MAGIC ?

- Deceptively trivial: just one method

```
void handle(Callback[] callbacks);
```

- Callbacks will be different depending on context
- May be invoked multiple times, depending on mechanism
- Need to maintain state between invocations
- Balancing act when interacting with JAAS

MAKING THINGS SIMPLE

Two solutions out of the box:

- SimpleAuthorizingCallbackHandler
 - Supports property files of nodes/passwords
 - Optionally performs role-based access control
- SaslClientCallbackHandler
 - Supports simple username/password
 - Supports GSSAPI (i.e. Kerberos tokens)

CLUSTER AUTHENTICATION IN JDG SERVER

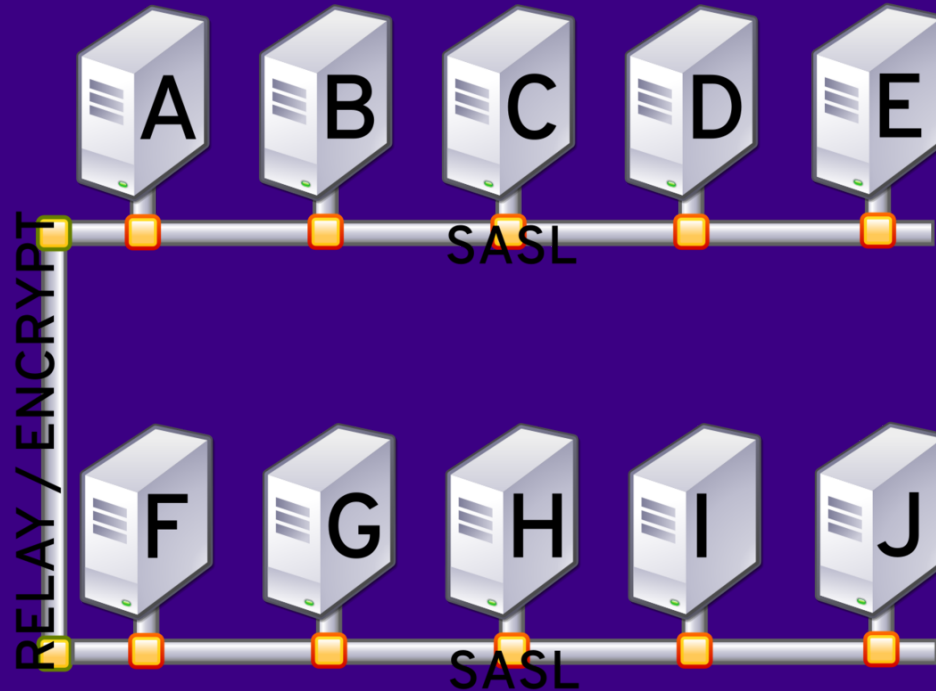
- Similar to embedded/library mode
- Integrated with server Security Realms
- Leverage EAP-style RBAC, GSSAPI, LDAP
- Can retrieve secrets from the vault

ENCRYPTING CLUSTER NODE TRAFFIC

- Why not use SASL's QOP for integrity and confidentiality ?
- The "burden" of asymmetry
- The ENCRYPT protocol
 - Option 1: Shared key store
 - Option 2: Dynamically generated key

WHAT ABOUT CROSS-SITE REPLICATION ?

- Relay transports are like normal transports
- Just add the required security protocols (i.e. ENCRYPT)
- Different strategies for different segments



THE BEST SECURITY IS "PHYSICAL"

Don't forget about network separation

- Dedicated network for cluster traffic
 - Physically separate
 - PVLAN
- Possibly with channel bonding for redundancy



CACHE

AUTHORIZATION

AUTHORIZATION

- Coarse-grained authorization permissions
- Cache- and CacheManager-level
- Roles and permissions
- The JDK's SecurityManager

CACHE PERMISSIONS

- READ (get, contains)
- WRITE (put, replace, remove, evict)
- BULK_READ (getAll, size, keySet, entrySet, values, query)
- BULK_WRITE (putAll, clear)
- ALL_READ (combines READ and BULK_READ)
- ALL_WRITE (combines WRITE and BULK_WRITE)
- LIFECYCLE (start, stop)
- LISTEN (addListener)
- EXEC (map/reduce, distributed executor)
- ADMIN (everything else)
- ALL (everything)

CACHEMANAGER PERMISSIONS

- CONFIGURATION (define configuration)
- LISTEN (addListener)
- LIFECYCLE (start, stop)
- ALL (everything)

COMBINING PERMISSIONS: ROLES

- Roles are sets of permissions (ACL)
- Classes of users vs Classes of operations
- Roles are global (i.e. CacheManager)
- How do I map users to roles ?

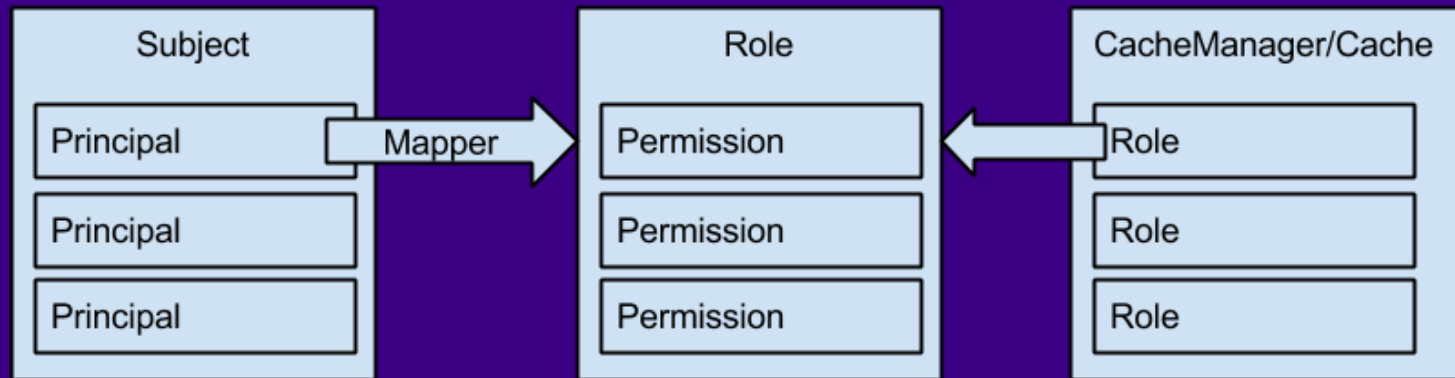
ROLE MAPPING

- The JAAS Subject
- A collection of Principals (user, groups, etc)
- Convert Principal to Roles

```
interface PrincipalRoleMapper {  
    Set<String> principalToRoles(Principal prin  
}
```

- Out-of-the-box mappers:
 - IdentityRoleMapper
 - CommonNameRoleMapper
 - ClusterRoleMapper

FROM SUBJECT TO PRINCIPAL TO ROLE TO PERMISSION



THE SECURITYMANAGER

- Controls who can execute certain parts of the code
- Coarse-grained, based on the code-source (i.e. jar) and a policy file
- Fine grained, based on the `AccessControlContext`, i.e. *Subject.doAs*
- Heavy impact on performance

LIGHTWEIGHT SECURITY

- Does NOT need a SecurityManager
- Replaces the AccessControlContext with a ThreadLocal solution
- Not a generic solution: only serves JDG
- Performs much better

ACCESS CONTROL IN LIBRARY MODE

```
final CacheManager cacheManager = ...
Subject.doAs(subject, new PrivilegedAction<Void>() {
    Void run() {
        Cache cache = cacheManager.getCache();
        cache.put("key", "value");
        return null;
    }
});
```

- Obtain a Subject (Container, PicketBox, Shiro, etc)
- Wrap calls in *Subject.doAs(subject, ...)*;

ACCESS CONTROL IN JBOSS DATA GRID SERVER

- Integrated with server Security realms
- Configured per-endpoint
- One container, multiple endpoints, different security

```
<hotrod-connector
  socket-binding="hotrod"
  cache-container="local">
  <authentication
    security-realm="ApplicationRealm">
      <sasl
server-name="localhost" mechanisms="PLAIN"/>
    </authentication>
  </hotrod-connector>
```

REMOTE CLIENT AUTHENTICATION



REMOTE CLIENT SECURITY

HotRod

- Encryption via SSL/TLS server certificates
- Authentication based on SASL or client certificates
- PLAIN, DIGEST-MD5, GSSAPI, etc
- Clients must provide their own CallbackHandler

AUDITING

The background is a solid dark blue. A wavy, horizontal line in shades of orange and yellow flows from the left side towards the right, curving slightly downwards. On the left side, there is a cluster of small, glowing yellow dots of varying sizes, some of which are positioned above the wavy line.

AUDITING

- When, Who, What, Where
- Simple pluggable interface

```
interface AuditLogger {  
    void audit(Subject subject,  
               AuditContext context,  
               String contextName,  
               AuthorizationPermission perm,  
               AuditResponse resp);  
}
```

- Default implementation as a Logger

DEMO

The background is a solid dark blue. A wavy, horizontal line in shades of orange and yellow flows from the left side towards the right, curving slightly downwards. On the left side, near the start of this wave, there is a cluster of small, bright yellow dots that appear to be trailing off or dispersing.

DEMO



Cluster Authentication

```
<stack name="tcp">
  <transport type="TCP" socket-binding="jgroups-tcp"/>
  ...
  <sasl security-realm="ApplicationRealm" mech="DIGEST-MD5">
    <property name="client_password">
      ${VAULT::node0_md5::passwd_hash::1}
    </property>
  </sasl>
</stack>
```

```
<vault>
  <vault-option name="KEYSTORE_URL"
    value="${jboss.server.config.dir}/vault/vault.keystore"/>
  <vault-option name="KEYSTORE_PASSWORD" value="MASK-AI3ZRwVO1Pd"/>
  <vault-option name="KEYSTORE_ALIAS" value="ispn-vault"/>
  <vault-option name="SALT" value="12345678"/>
  <vault-option name="ITERATION_COUNT" value="23"/>
  <vault-option name="ENC_FILE_DIR"
    value="${jboss.server.config.dir}/vault/" />
</vault>
```

Cluster Authorization

```
<security-realm name="ApplicationRealm">
  <authentication>
    <properties path="application-users.properties"
               relative-to="jboss.server.config.dir"/>
  </authentication>
  <authorization>
    <properties path="application-roles.properties"
               relative-to="jboss.server.config.dir"/>
  </authorization>
</security-realm>
```

```
# Node roles
thunder=clustered
lightning=clustered
```

Cache Container Authorization

```
<cache-container name="clustered"
                 default-cache="default"
                 statistics="true">
  <security>
    <authorization>
      <identity-role-mapper/>
      <role name="admin"
            permissions="ALL"/>
      <role name="reader"
            permissions="READ BULK_READ"/>
      <role name="writer"
            permissions="WRITE BULK_WRITE"/>
      <role name="supervisor"
            permissions="ALL_READ ALL_WRITE"/>
    </authorization>
  </security>
</cache-container>
```

Cache Authorization

```
<distributed-cache name="default" ...>  
  <security>  
    <authorization  
      roles="admin reader writer supervisor"  
      enabled="true"/>  
    </security>  
</distributed-cache>
```

HotRod Endpoint Authentication

```
<hotrod-connector socket-binding="hotrod"
  cache-container="clustered">
  <authentication security-realm="LdapRealm">
    <sasl server-context-name="hotrod-service"
      server-name="clustered"
      mechanisms="GSSAPI"
      qop="auth"
      strength="high medium low" />
  </authentication>
</hotrod-connector>
```

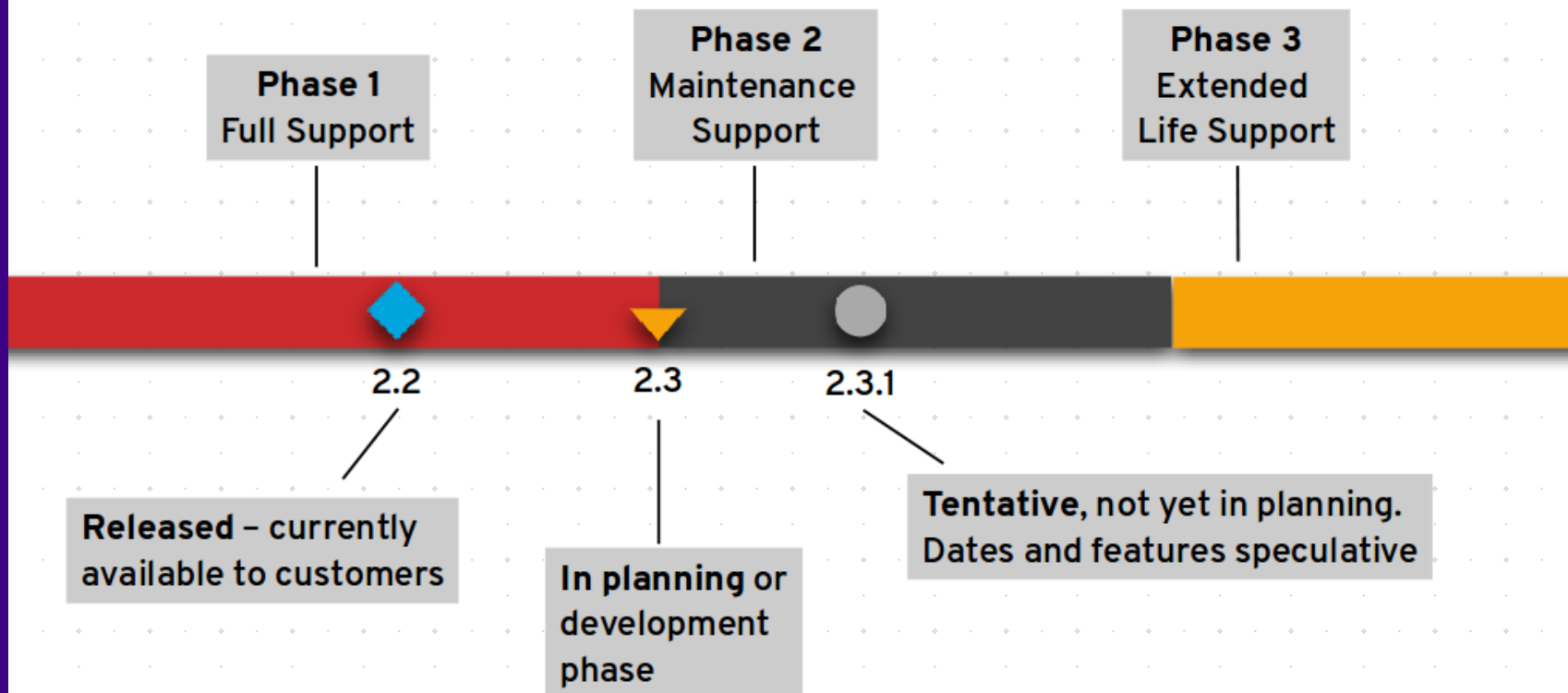
```
<security-domain name="hotrod-service" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="storeKey" value="true"/>
      <module-option name="useKeyTab" value="true"/>
      <module-option name="refreshKrb5Config" value="true"/>
      <module-option name="principal"
        value="hotrod/clustered@ISHKUR.NET"/>
      <module-option name="keyTab"
        value="{jboss.server.config.dir}/hotrod_clustered."
      >
      <module-option name="doNotPrompt" value="true"/>
    </login-module>
  </authentication>
</security-domain>
```

HotRod Endpoint Authorization

```
<security-realm name="LdapRealm">
  <authorization>
    <ldap connection="ldap_connection">
      <username-to-dn>
        <username-filter base-dn="cn=Users,dc=ishkur,dc=net"
          recursive="false"
          attribute="cn"
          user-dn-attribute="dn" />
      </username-to-dn>
      <group-search group-name="SIMPLE"
        iterative="true"
        group-dn-attribute="dn"
        group-name-attribute="cn">
        <principal-to-group group-attribute="memberOf" />
      </group-search>
    </ldap>
  </authorization>
</security-realm>
```


THE FUTURE

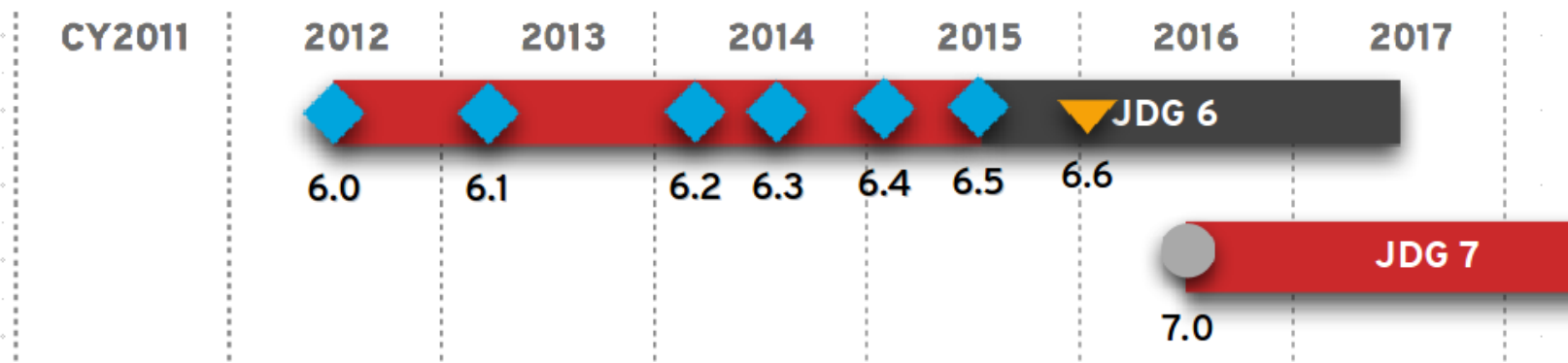
ROADMAP KEY



Usual caveats apply – features and release dates can and will change

JBOSS DATA GRID ROADMAP

Subject to Change



6.3 (July 2014)

- Security – Authentication/Authorization
- .NET Hot Rod client – Tech Preview
- Support in Karaf and Weblogic
- EAP 6 modules

6.4 (Jan 2015)

- JBoss Fuse - Camel component
- Remote Querying – Full Support
- Handling network partitions
- Clustered listeners
- Remote listeners – Tech Preview

6.5 (June 2015)

- Remote listeners – Full Support
- Near caching on Java Hot Rod client
- JSR-107 support
- Deploy Custom cache store on JDG Server
- .NET Hot Rod client – Full support
- JDG as Lucene Directory

6.6 (Target Q1 CY16)

- Continuous Queries

On the Horizon

Data Grid 7.0

In-memory analytics

- JDG as (standalone) in-memory store for Apache Hadoop and Spark
 - Leverage tools from Hadoop ecosystem – Hive, Pig
 - In-memory store for recent data
 - Faster processing
 - Historic data can be persisted to disk-based Hadoop/Spark store (Cassandra, Red Hat Storage)
- Enhanced Management and Monitoring
 - Intuitive visualization and operations at the cluster, cache, or node level

THANKS / Q&A

<http://www.redhat.com/en/technologies/jboss-middleware/data-grid>

<http://infinispan.org>

@infinispan + @tristantarrant

HAVE MORE QUESTIONS ?

Speak one-on-one with Red Hat Product Security experts in
Customer Central

Hall A, First floor