

Eike Stepper

stepper@esc-net.de
<http://www.esc-net.de>
<http://thegordian.blogspot.com>

Berlin, Germany

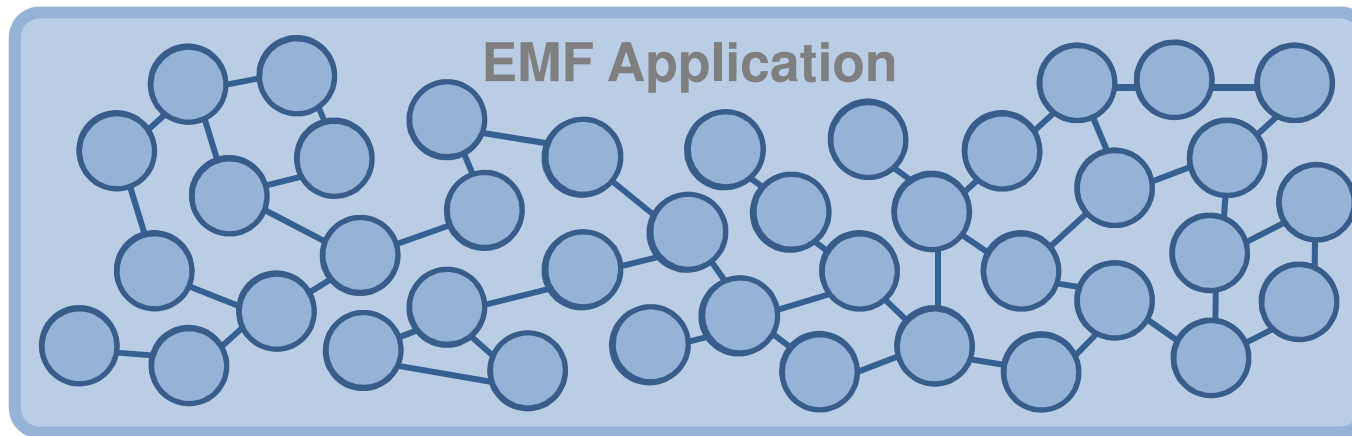


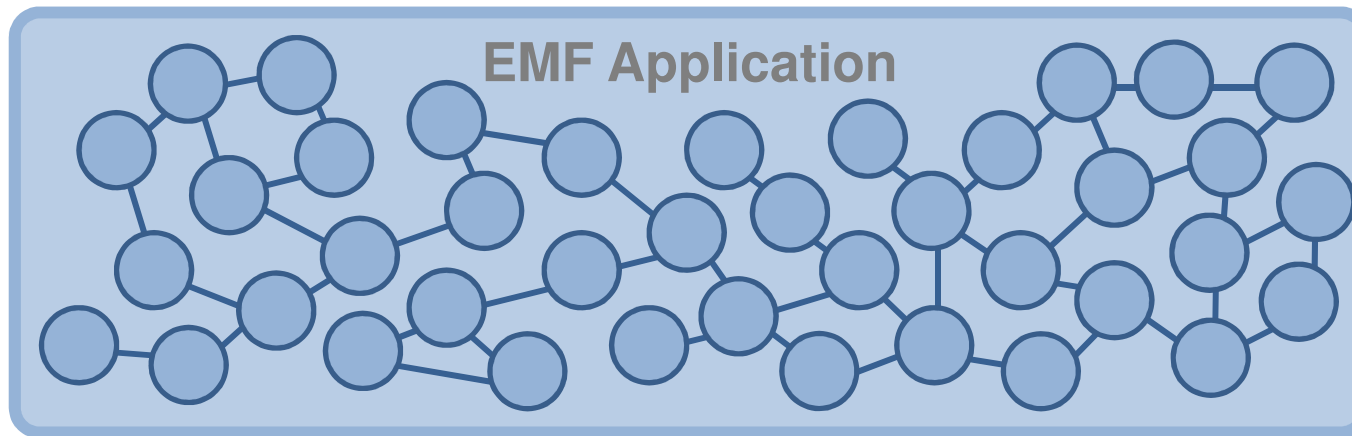
Scale, Share and Store your Models with CDO

EclipseCon Talk, March 24, 2010

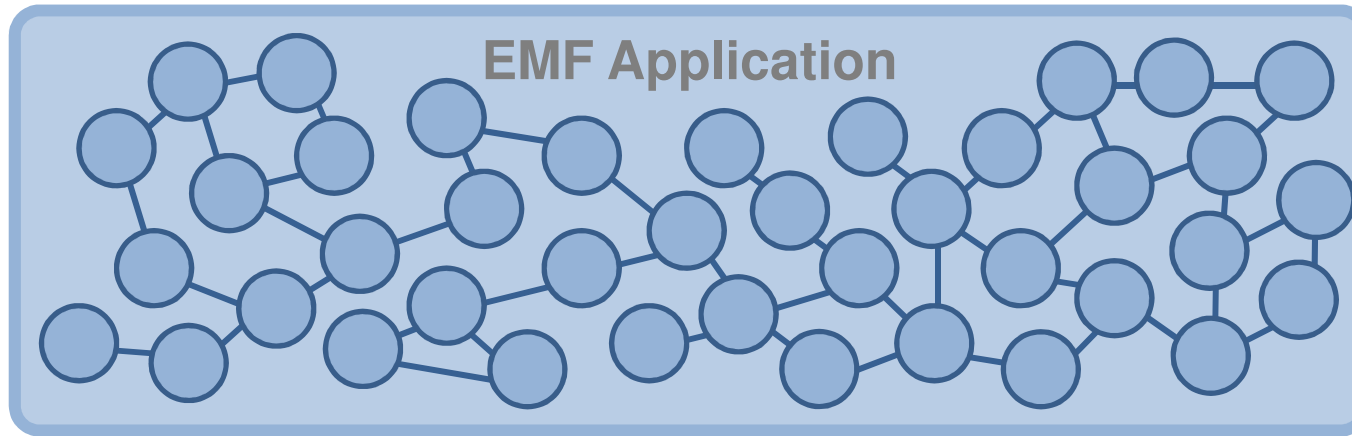
Wednesday, 14:30, 50 minutes | Stevens Creek

7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16



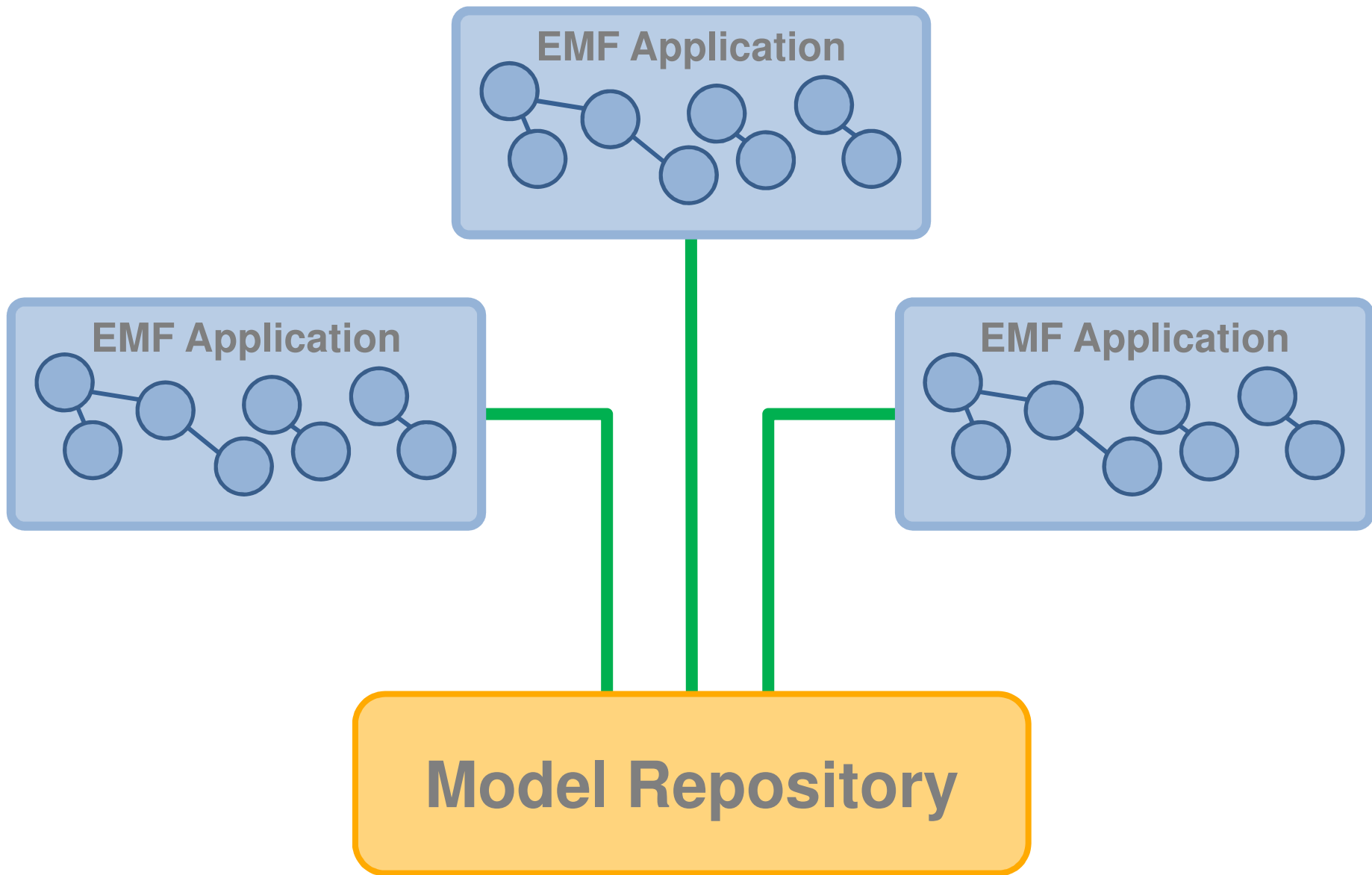


- **Huge models require lots of smaller files**
- **Partitioning must be done at design time**
- **Saving changes is not transactional safe**
- **Loading single objects is still impossible**
- **Garbage collection of objects is impossible**
- **Conflicts must be resolved in text form**
- **No change notifications to other clients**

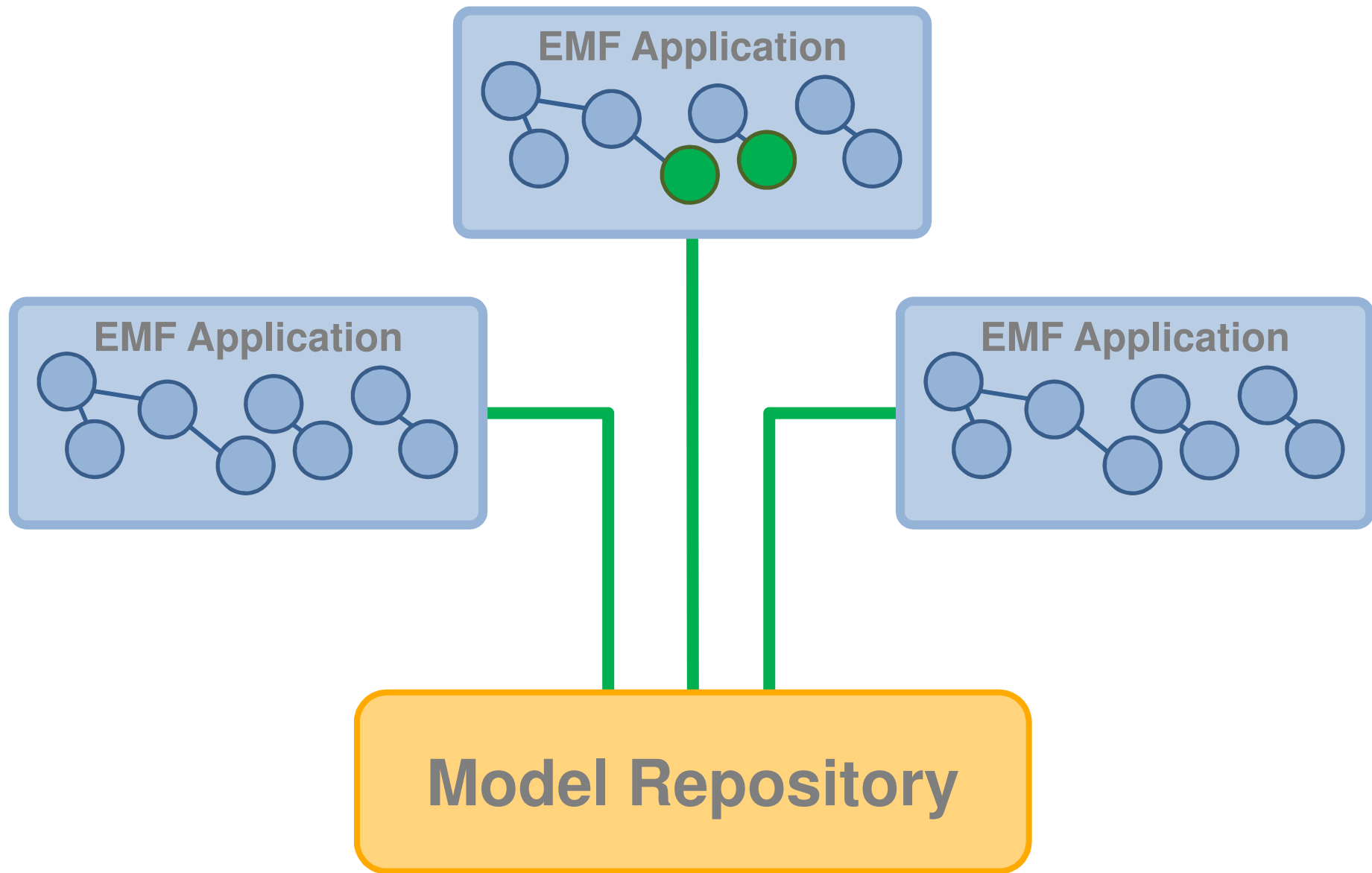


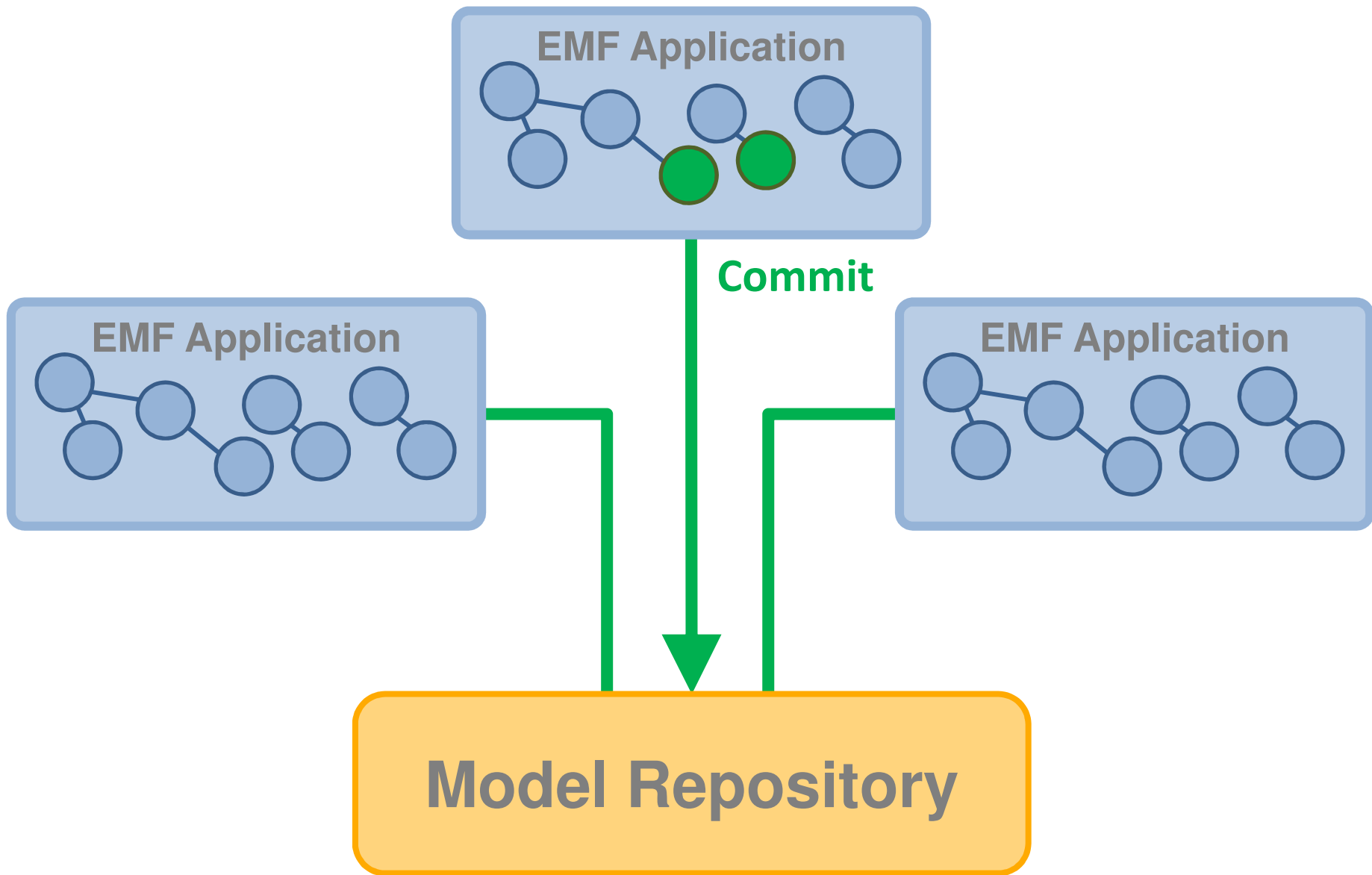
Does not scale well

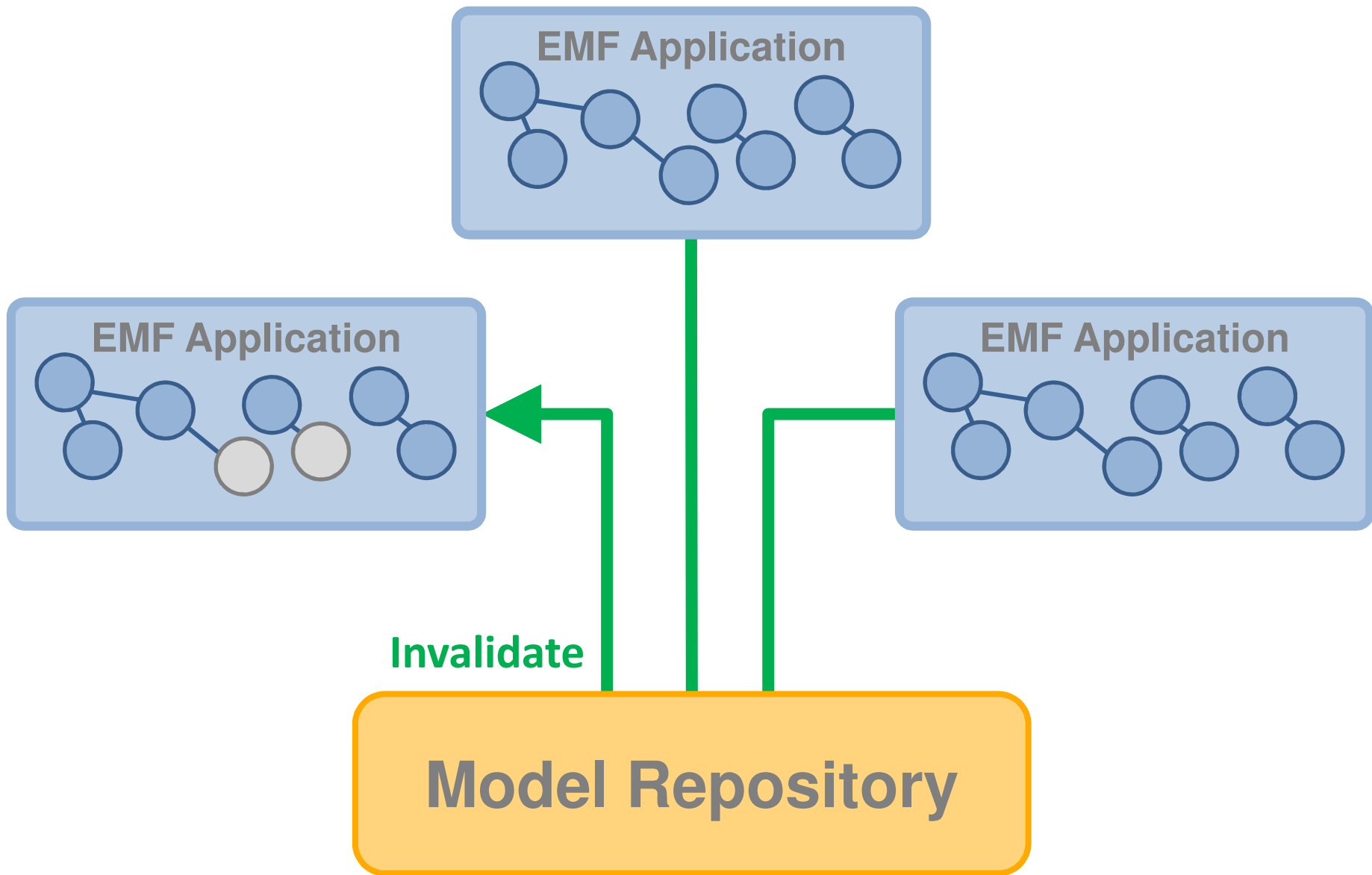
Not suitable for multi-user

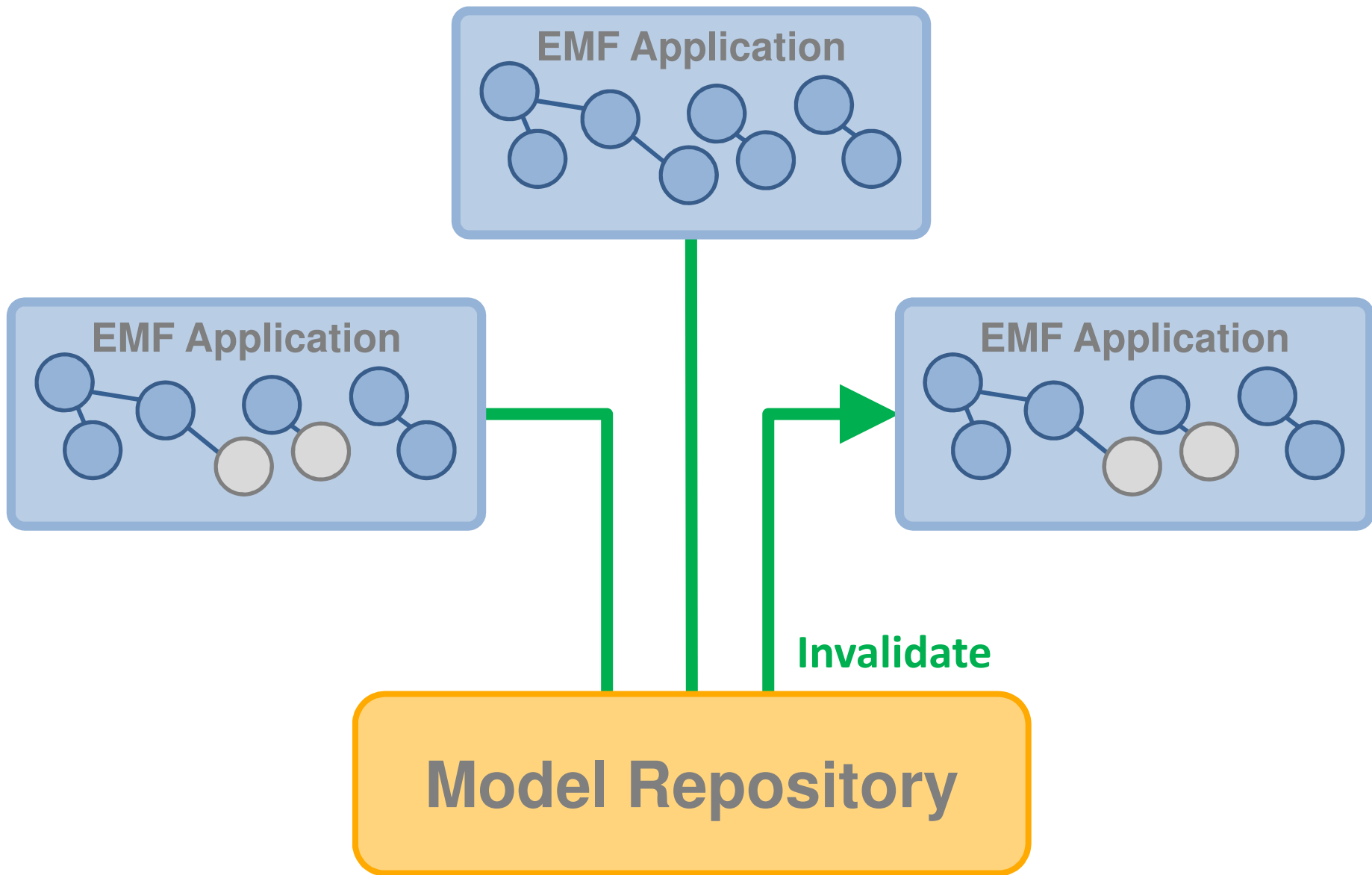


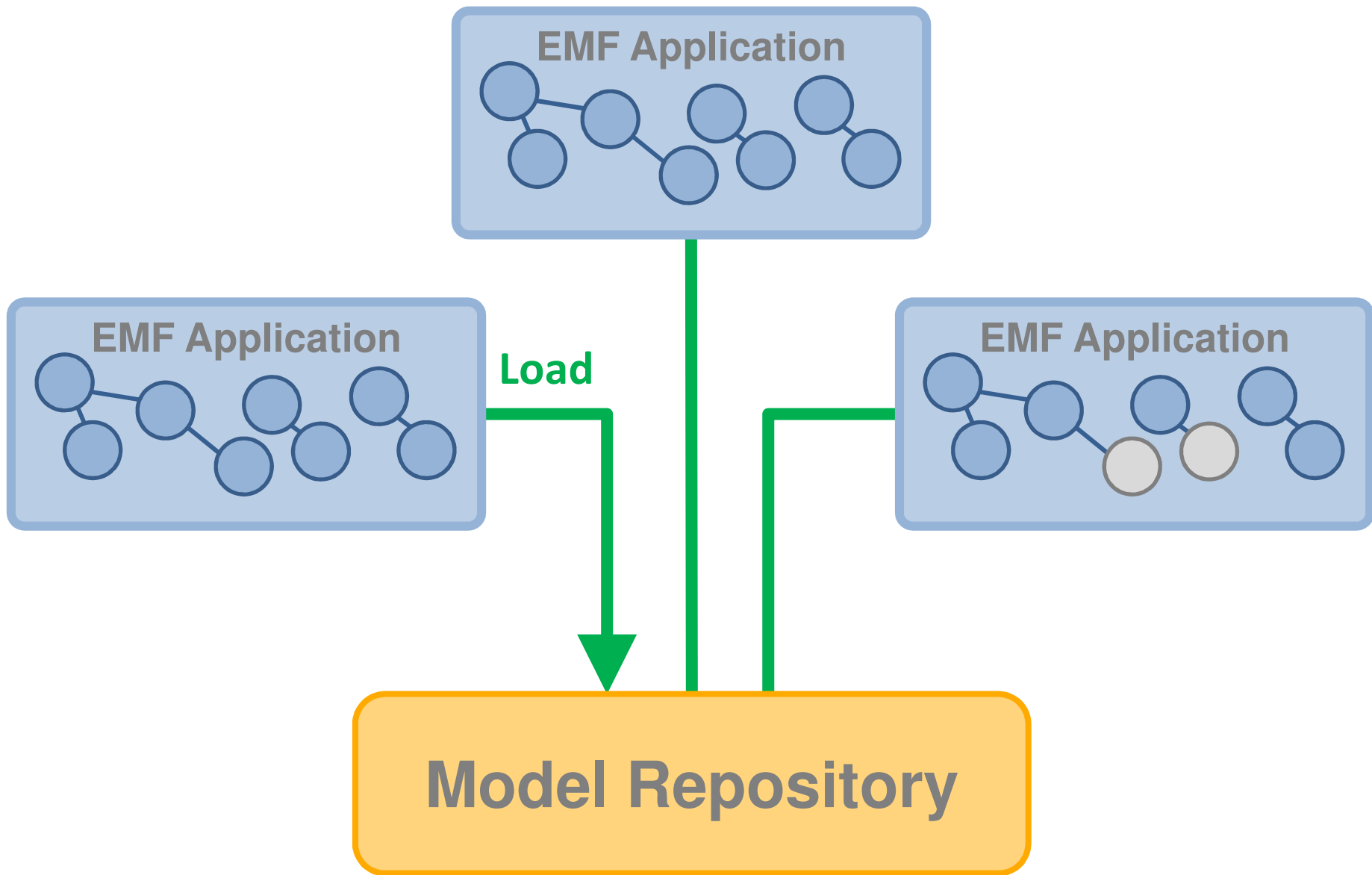
Modify

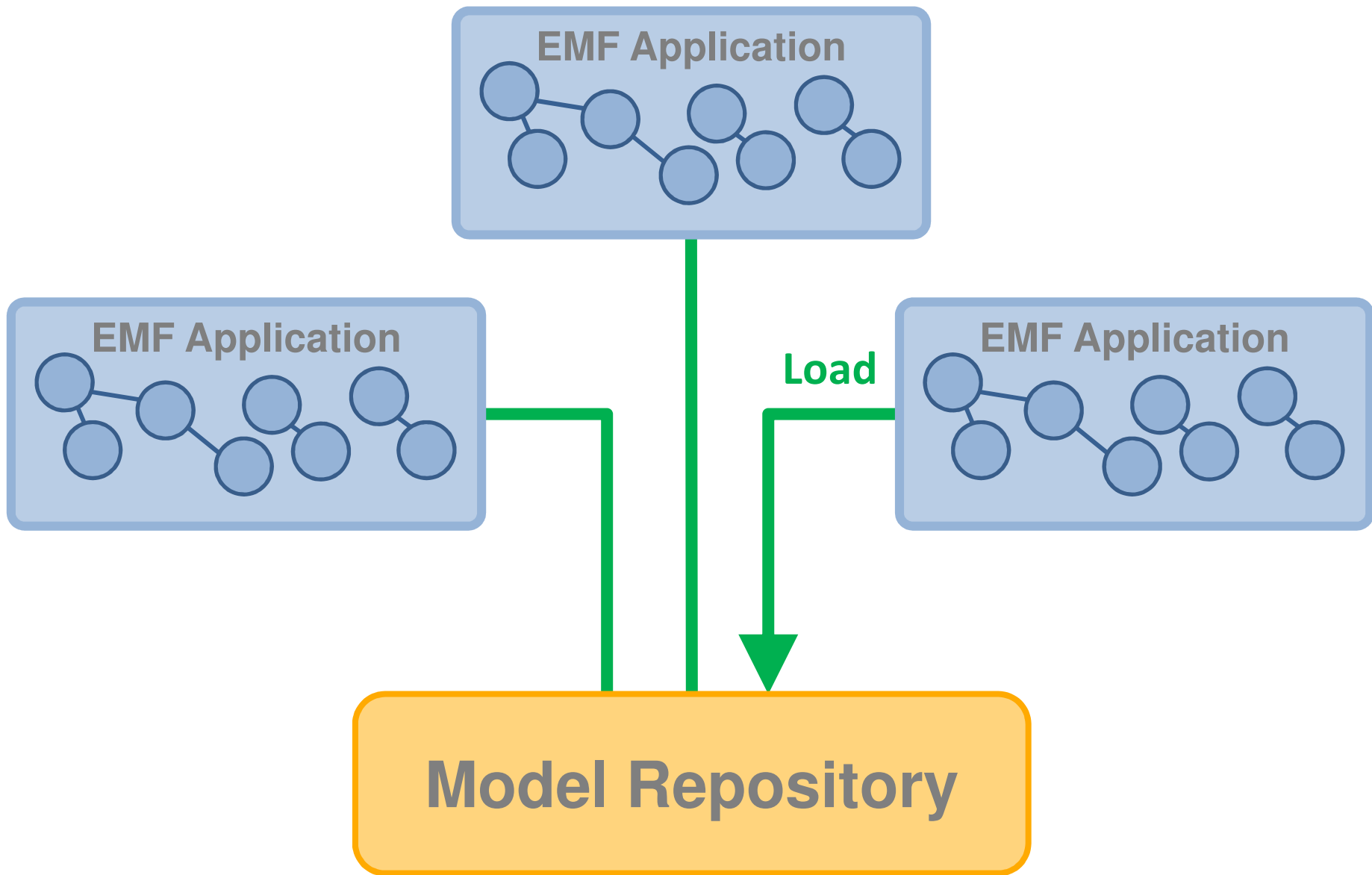


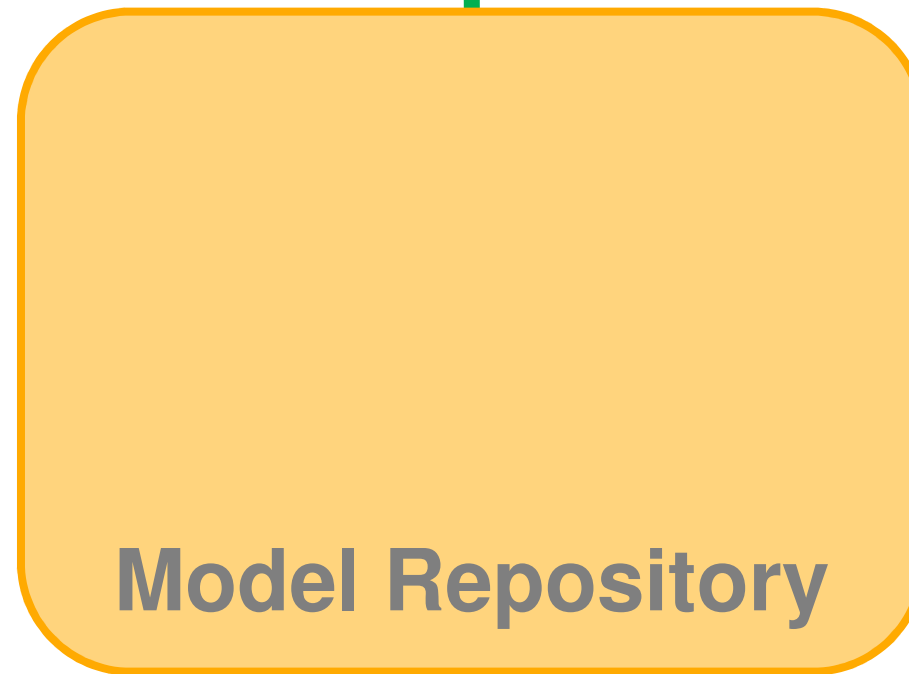
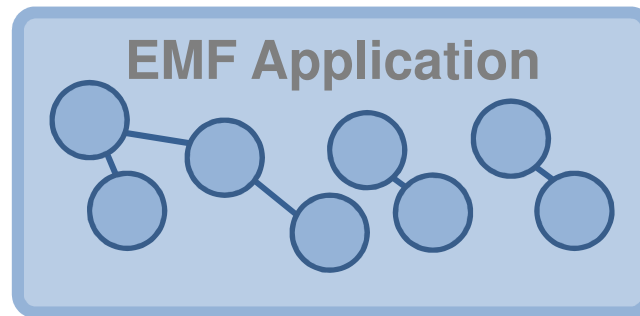


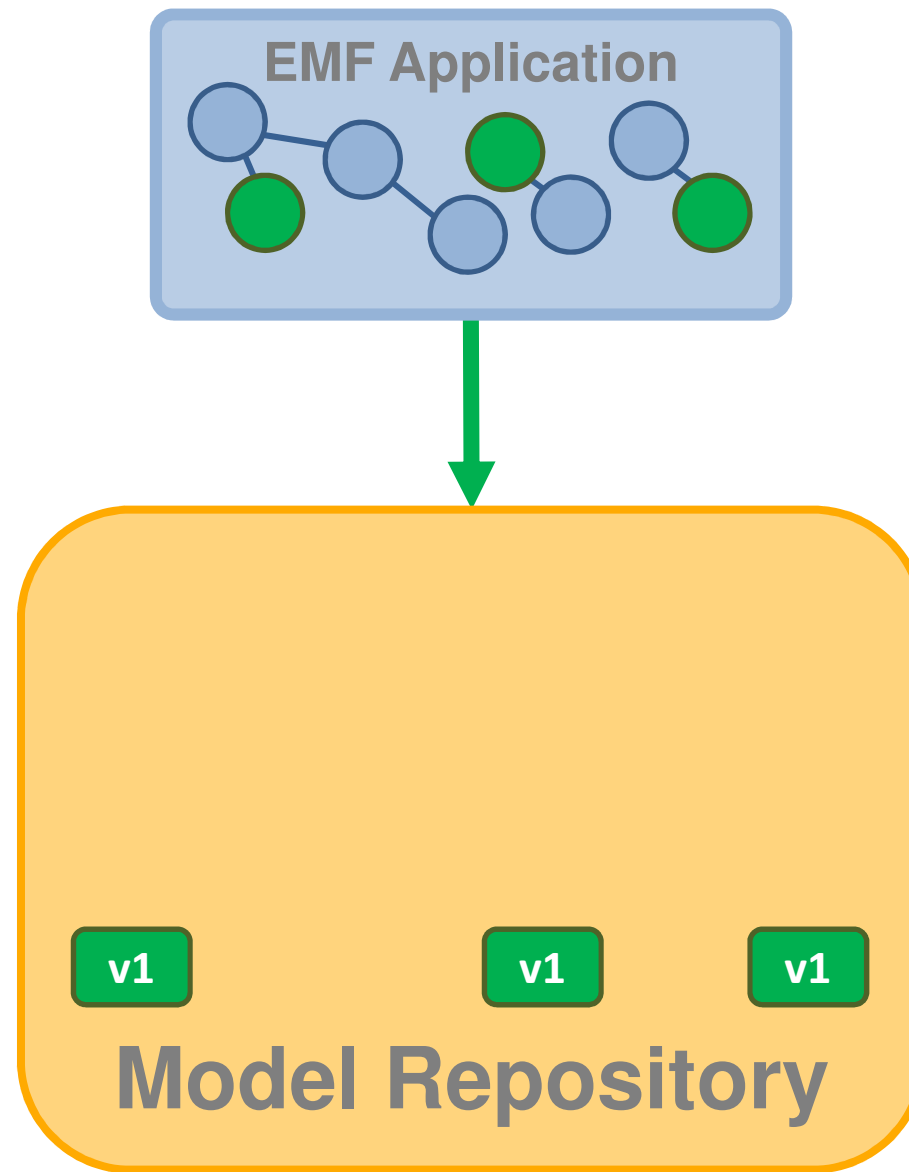


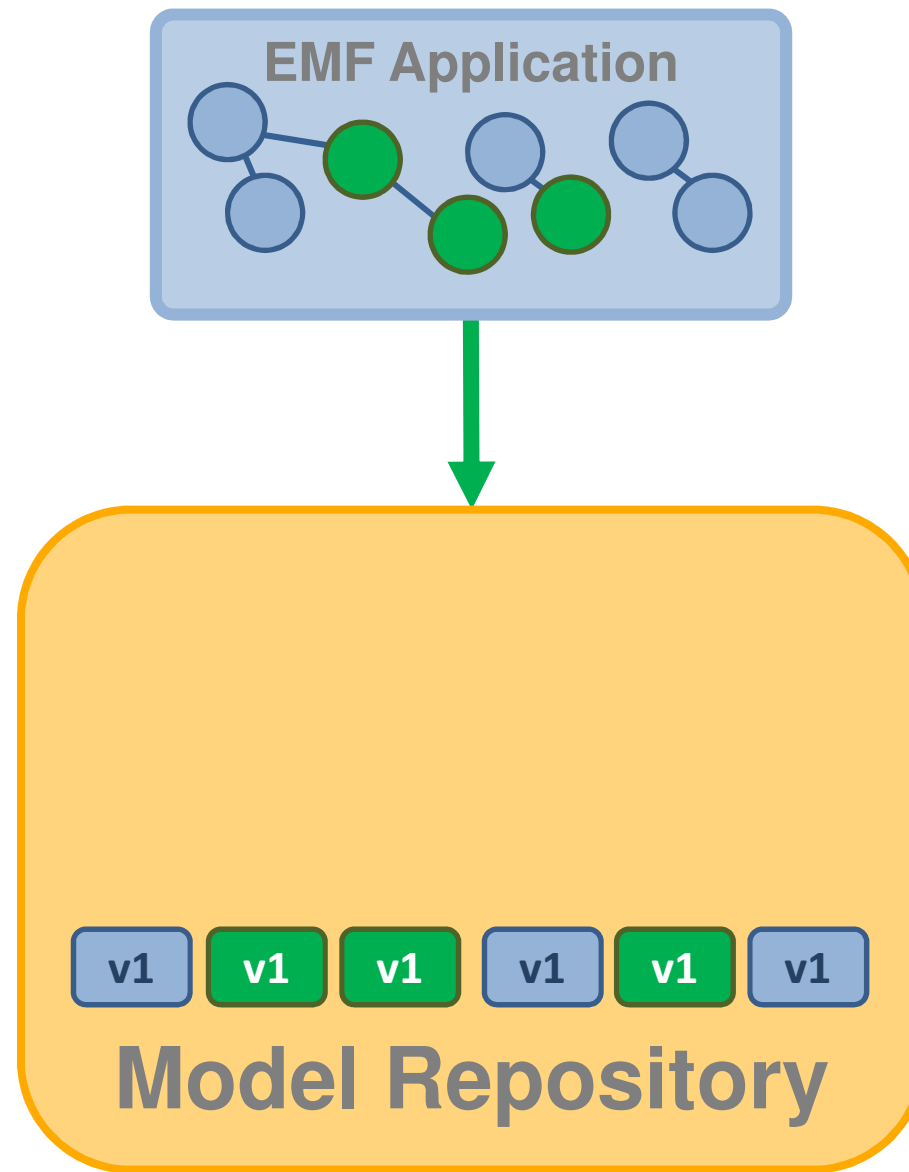


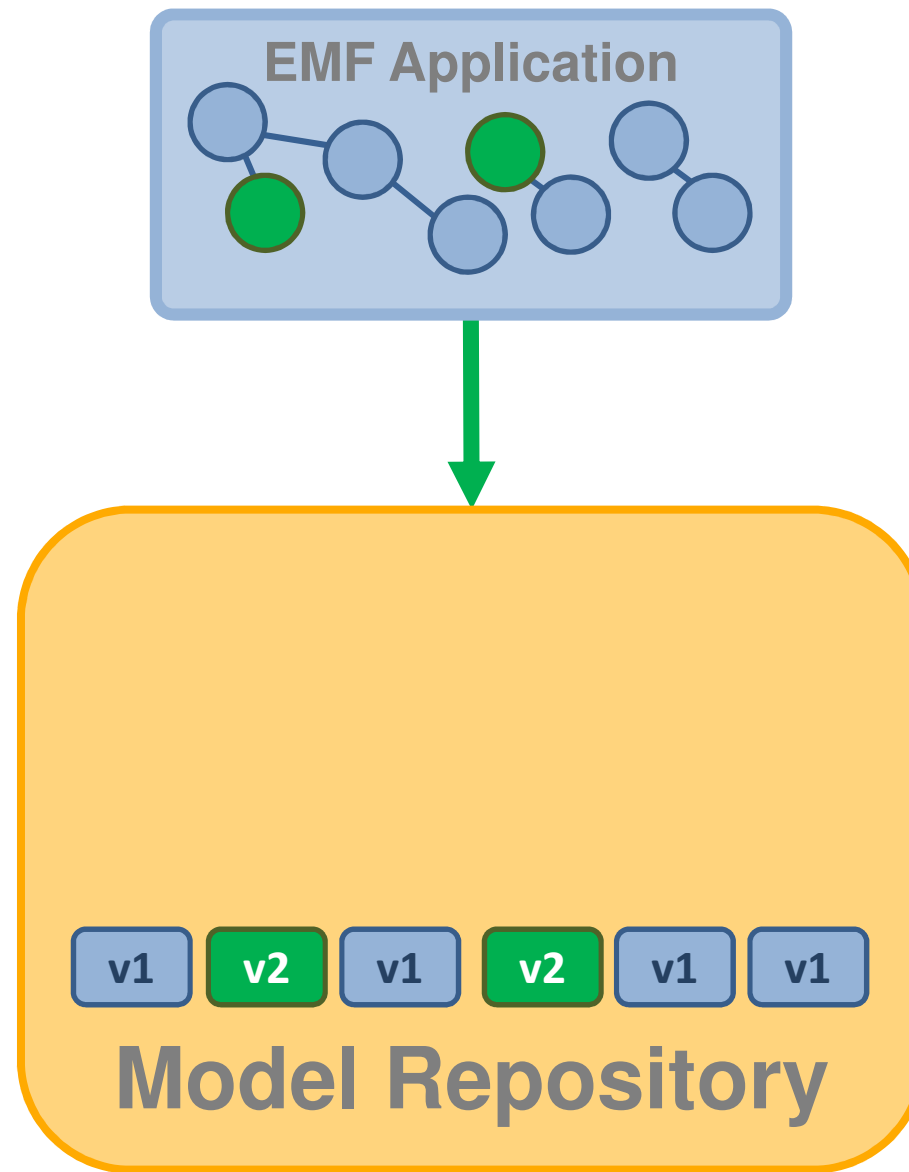


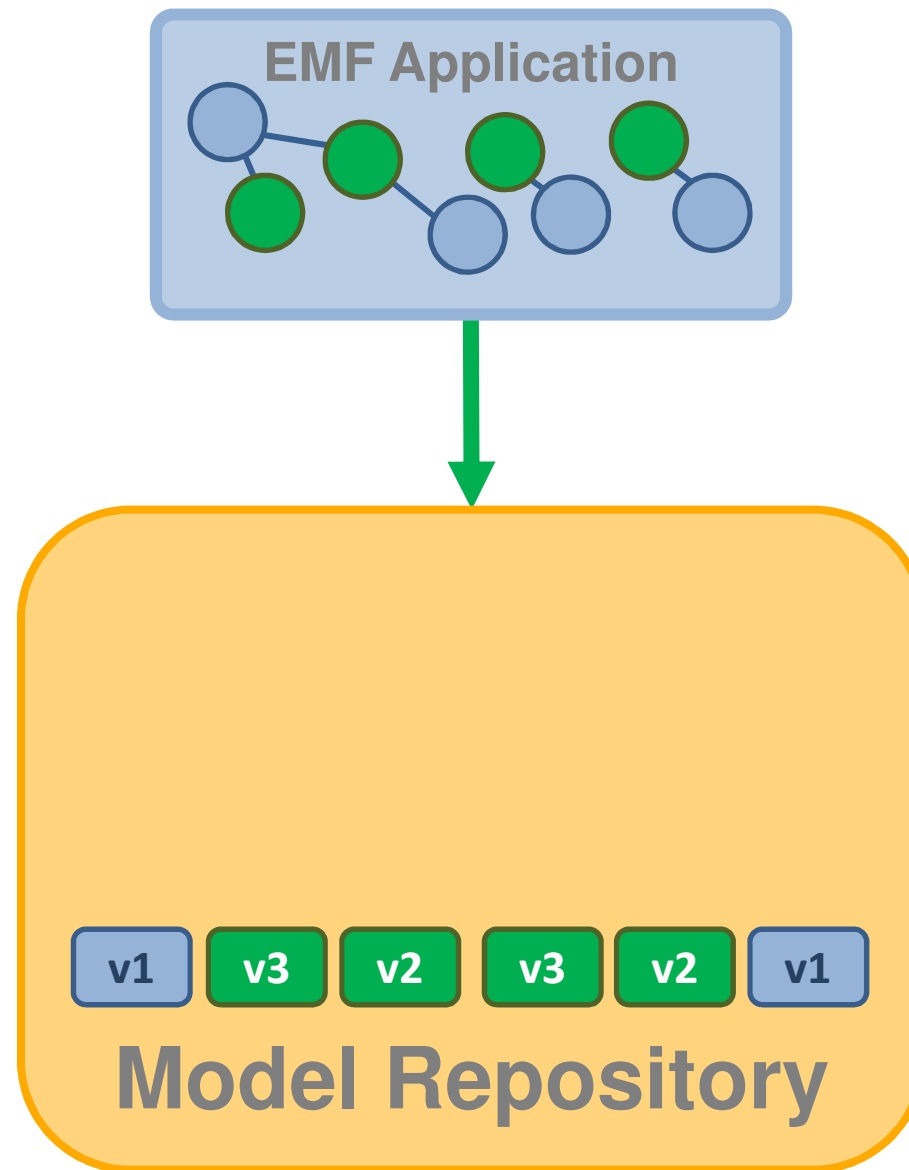


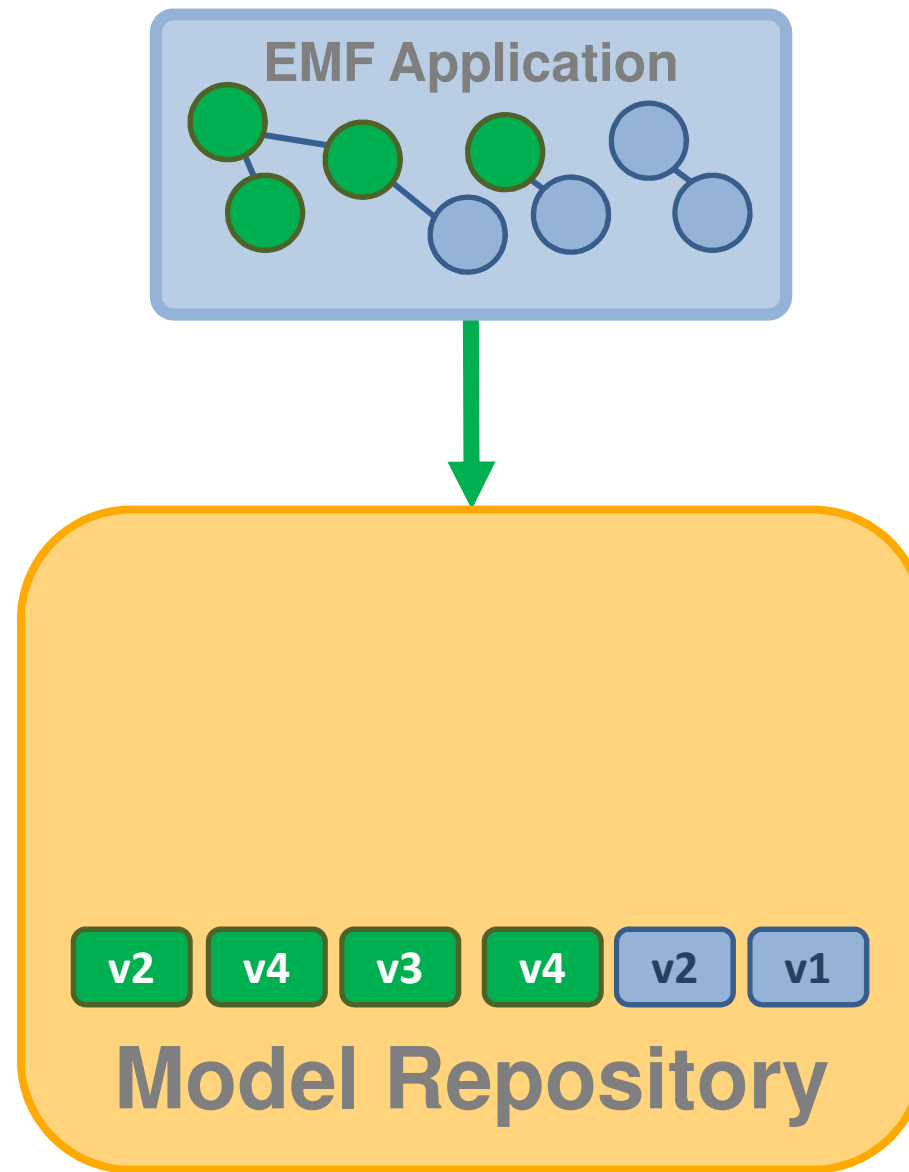


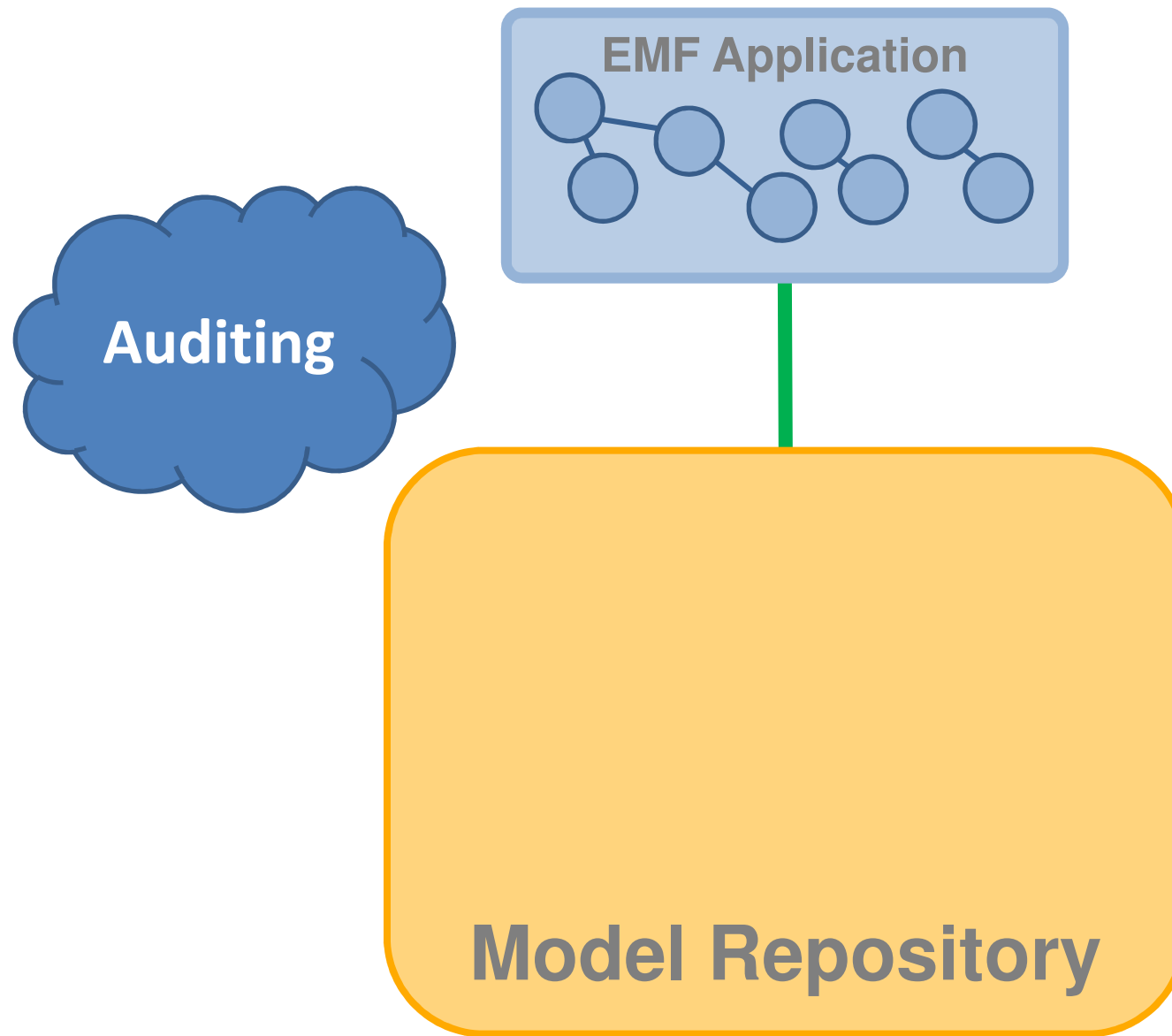


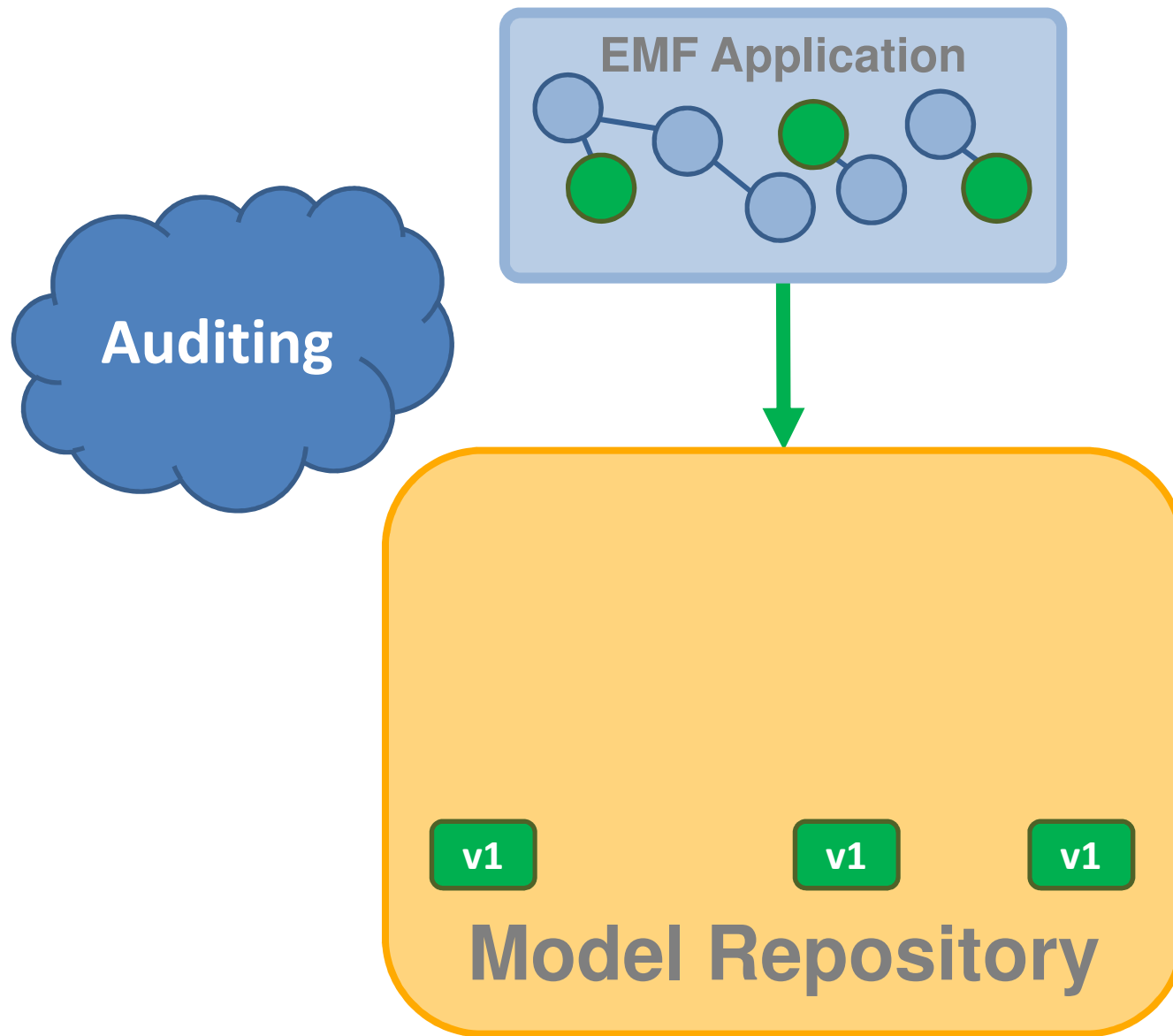


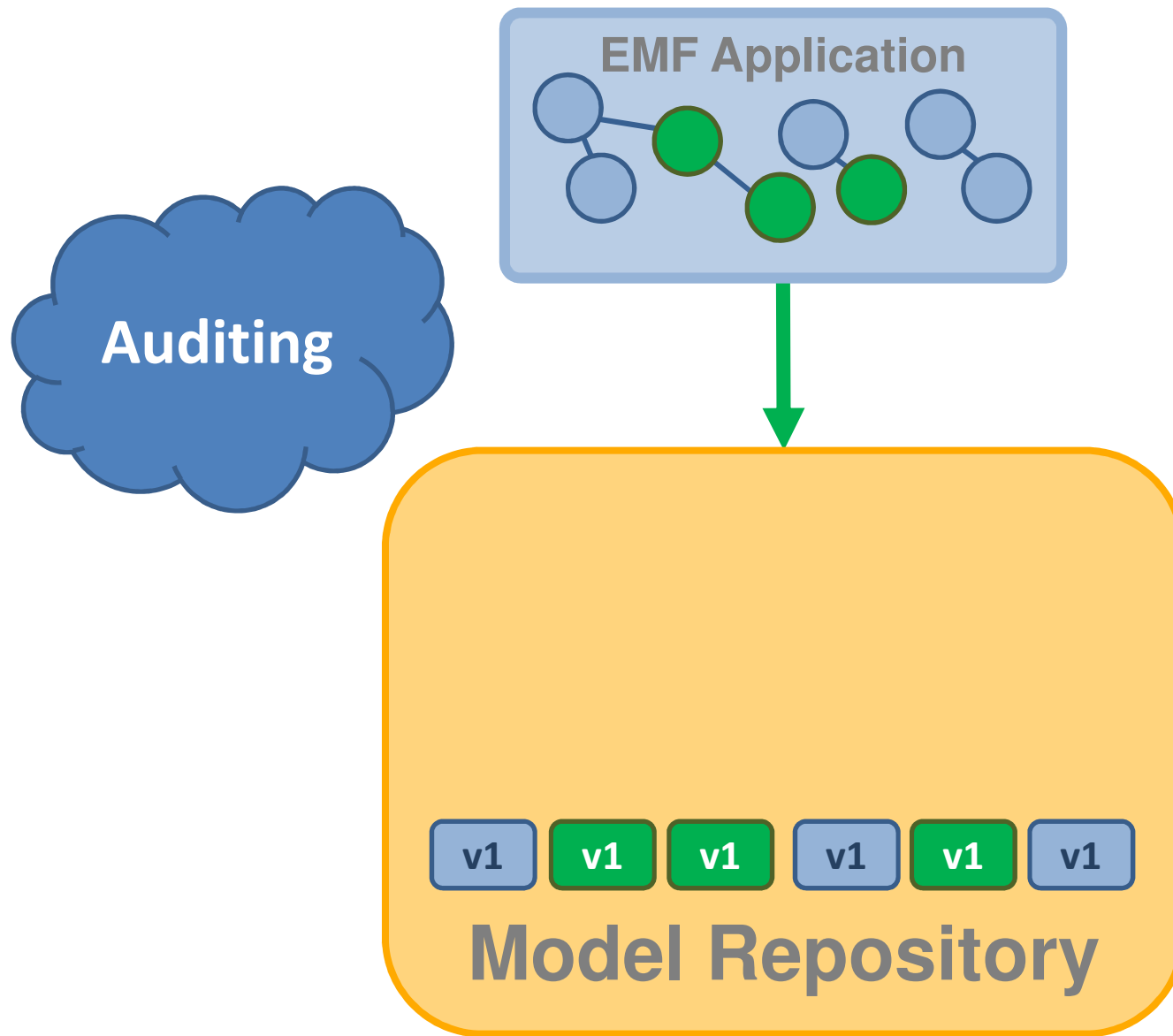


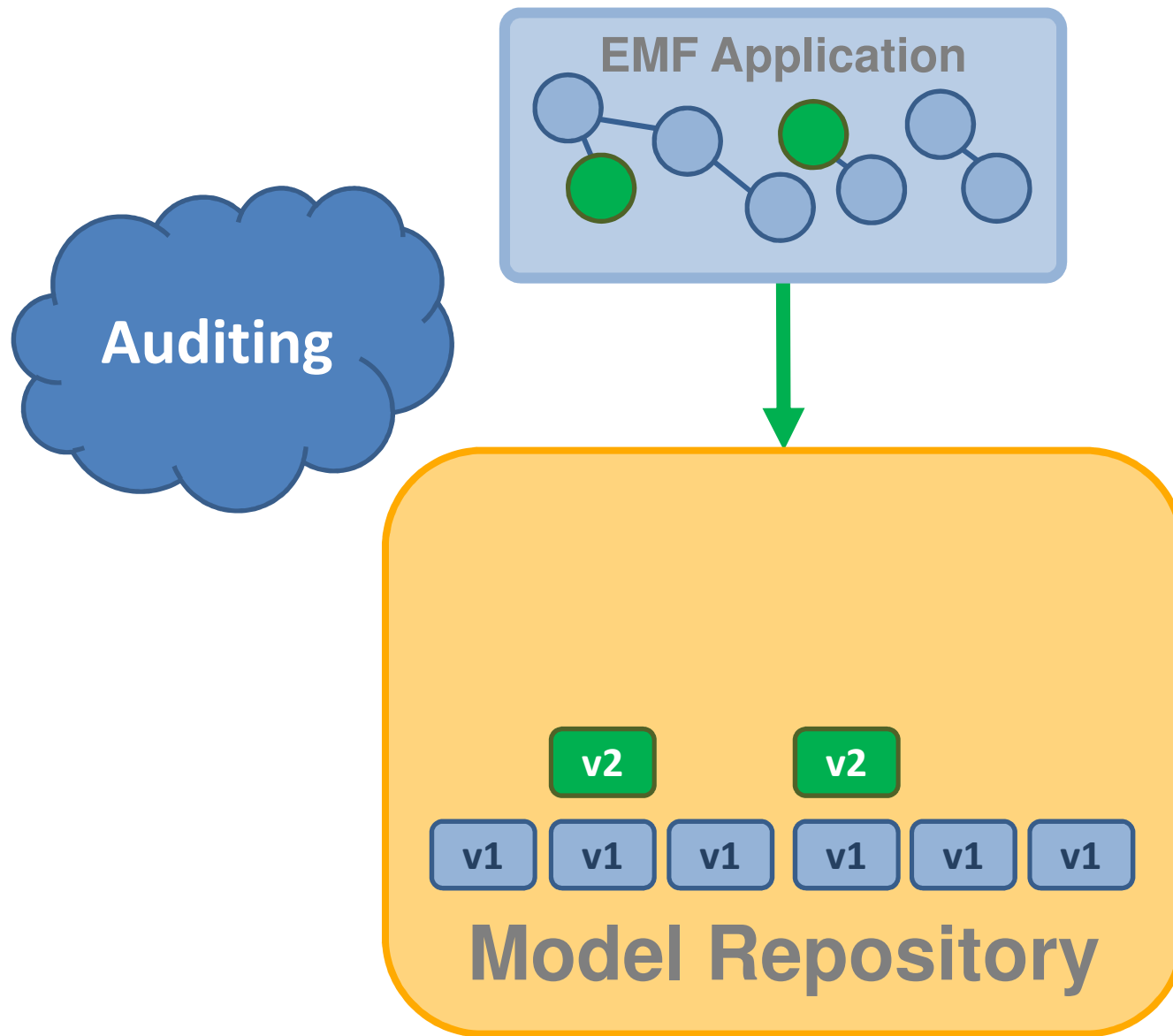


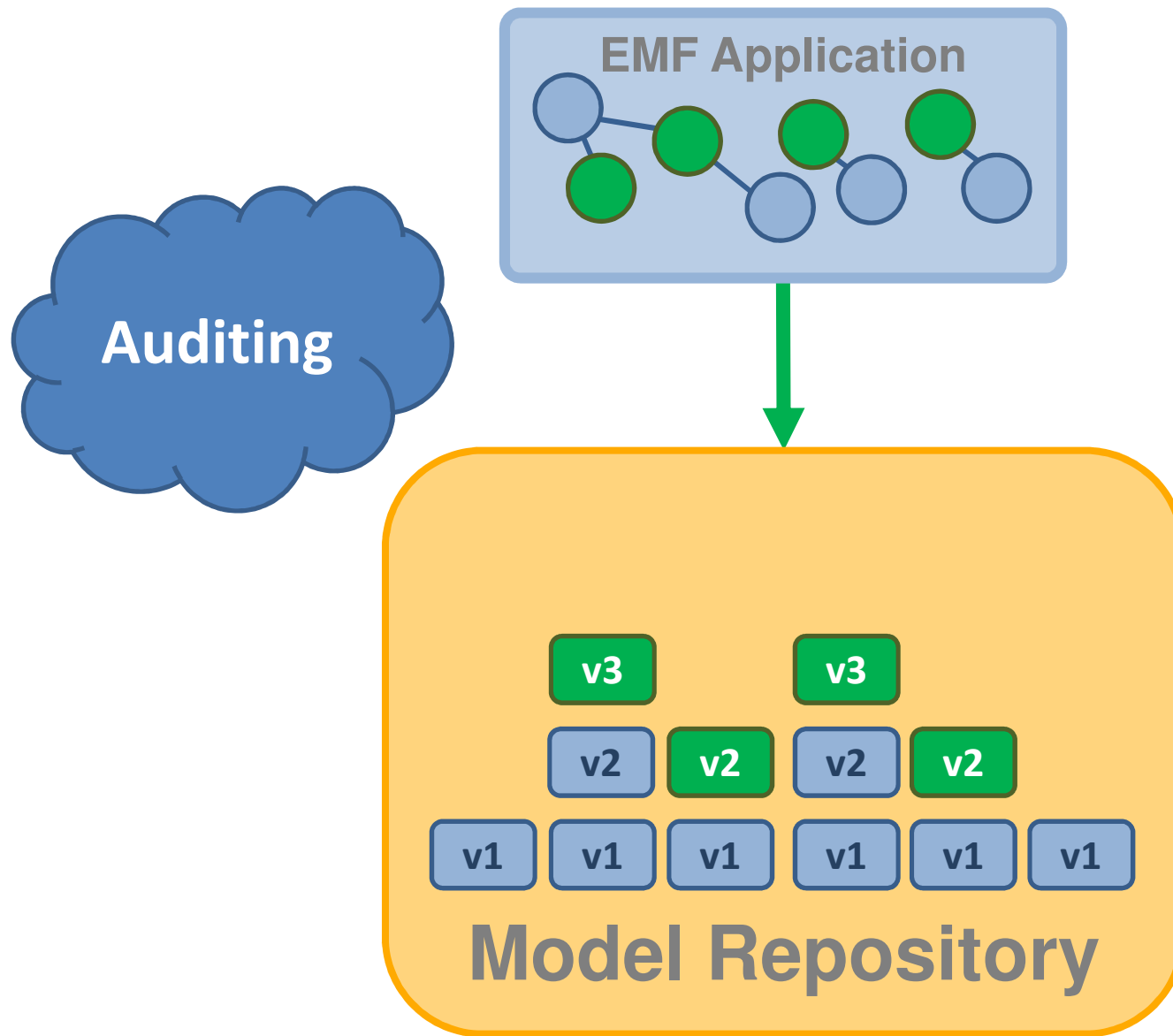


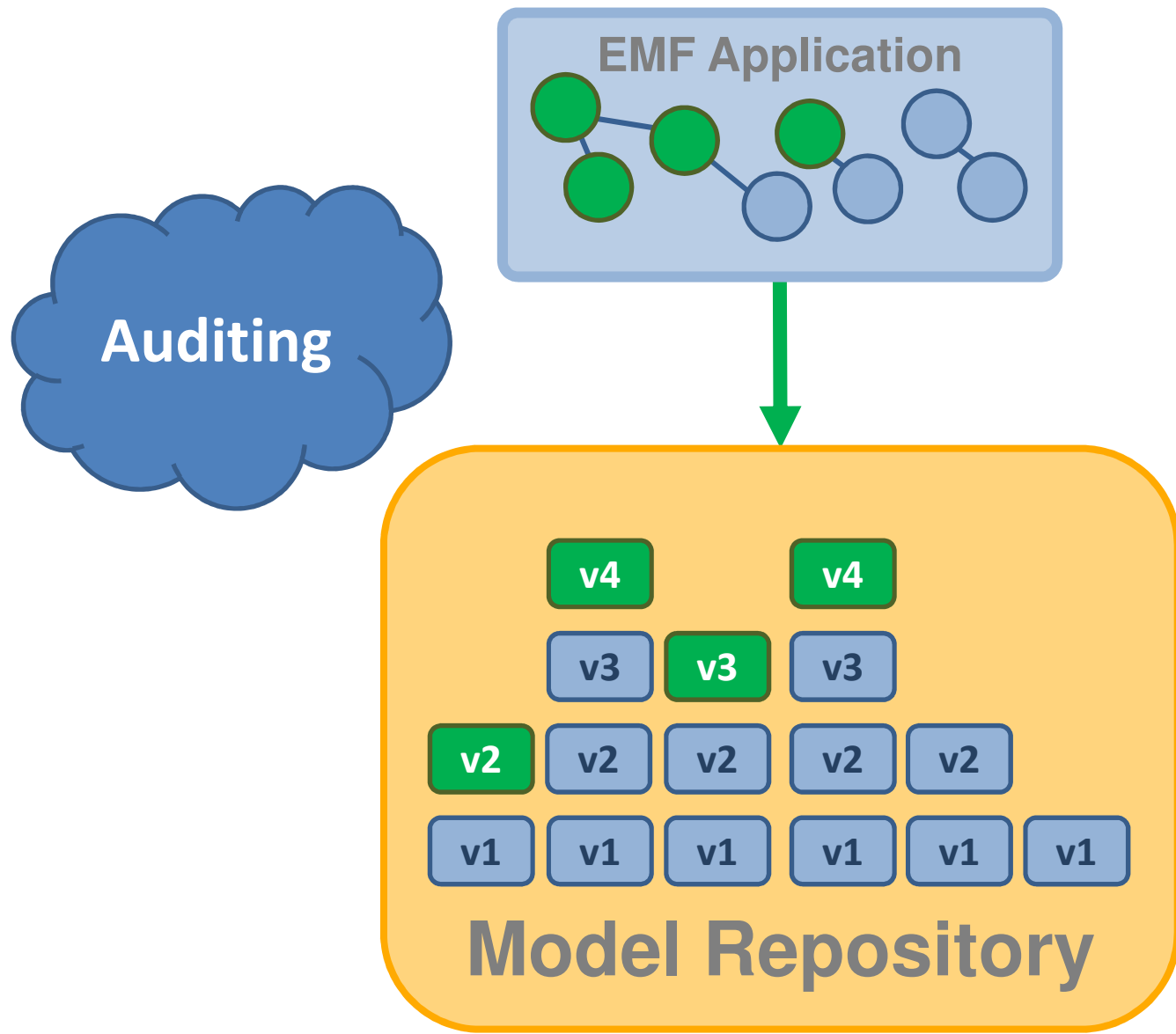


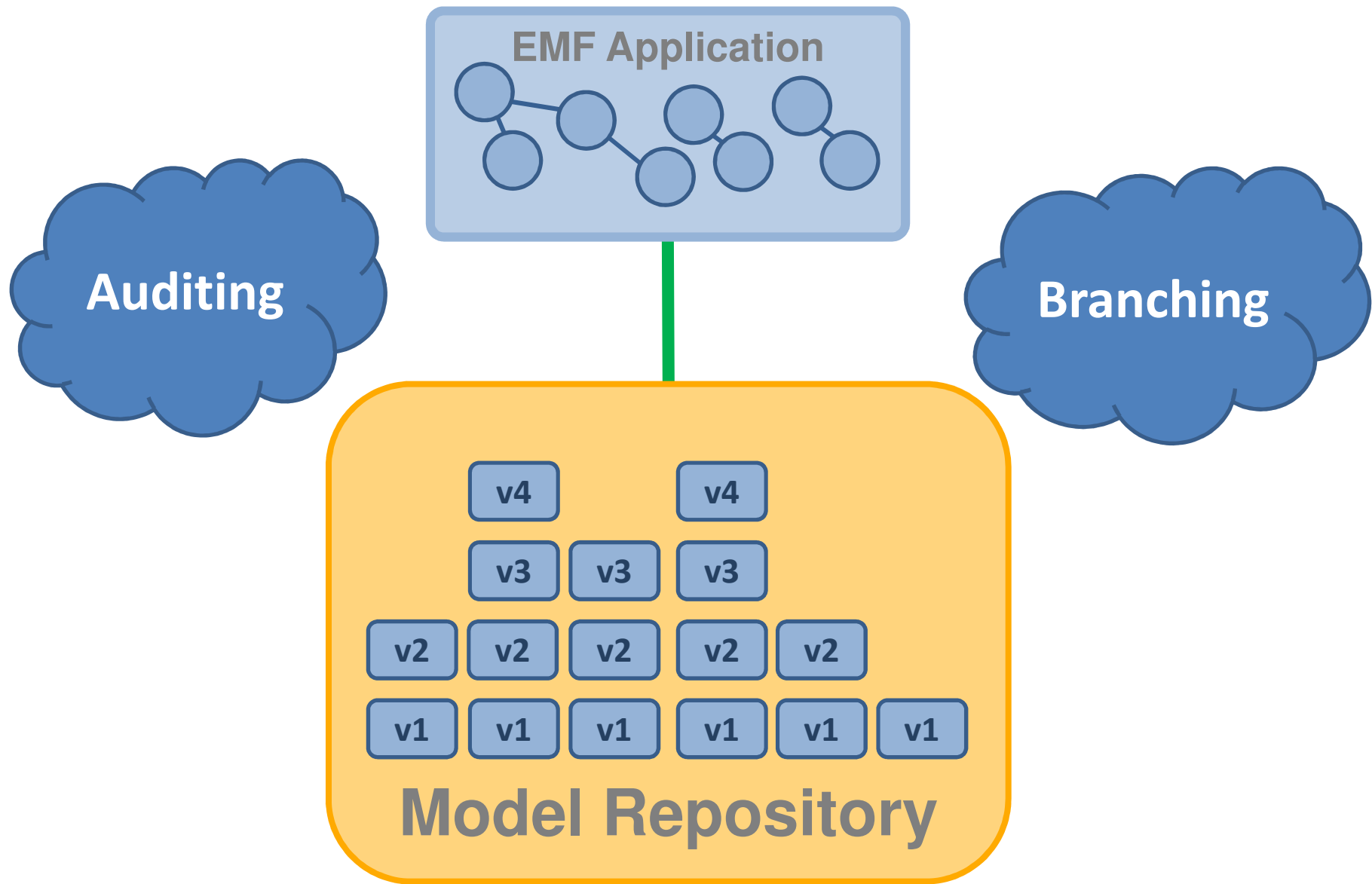


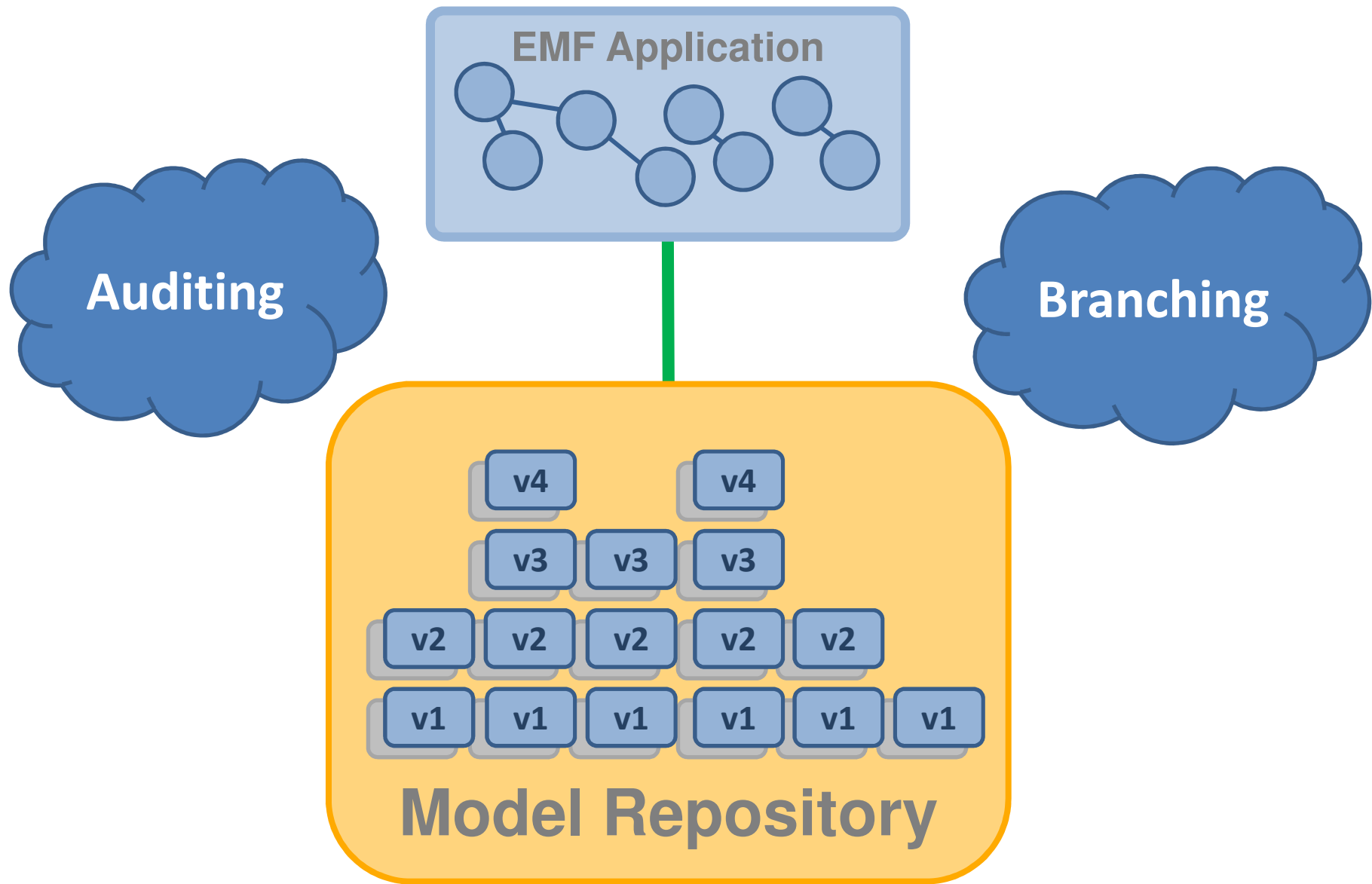


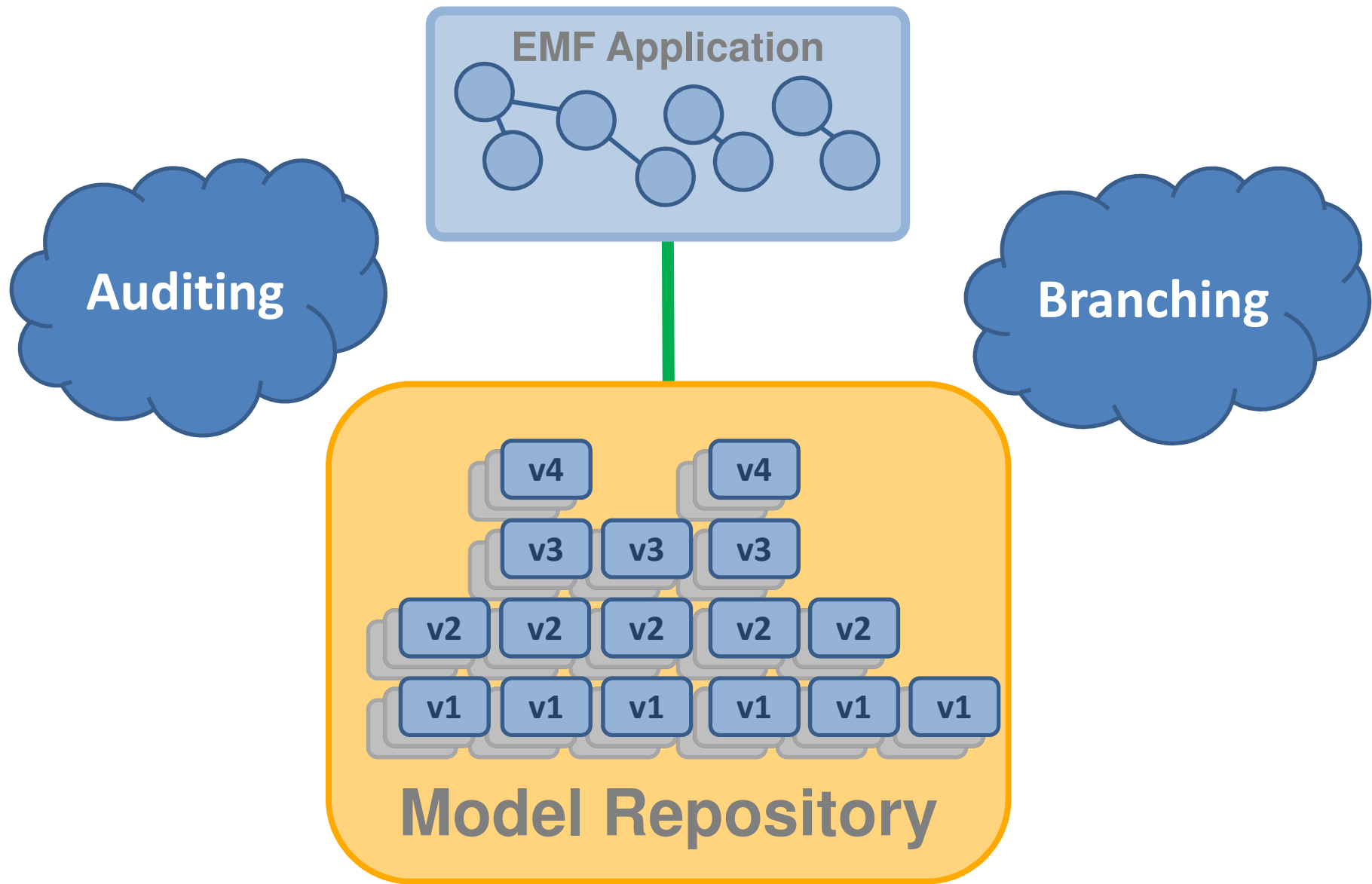


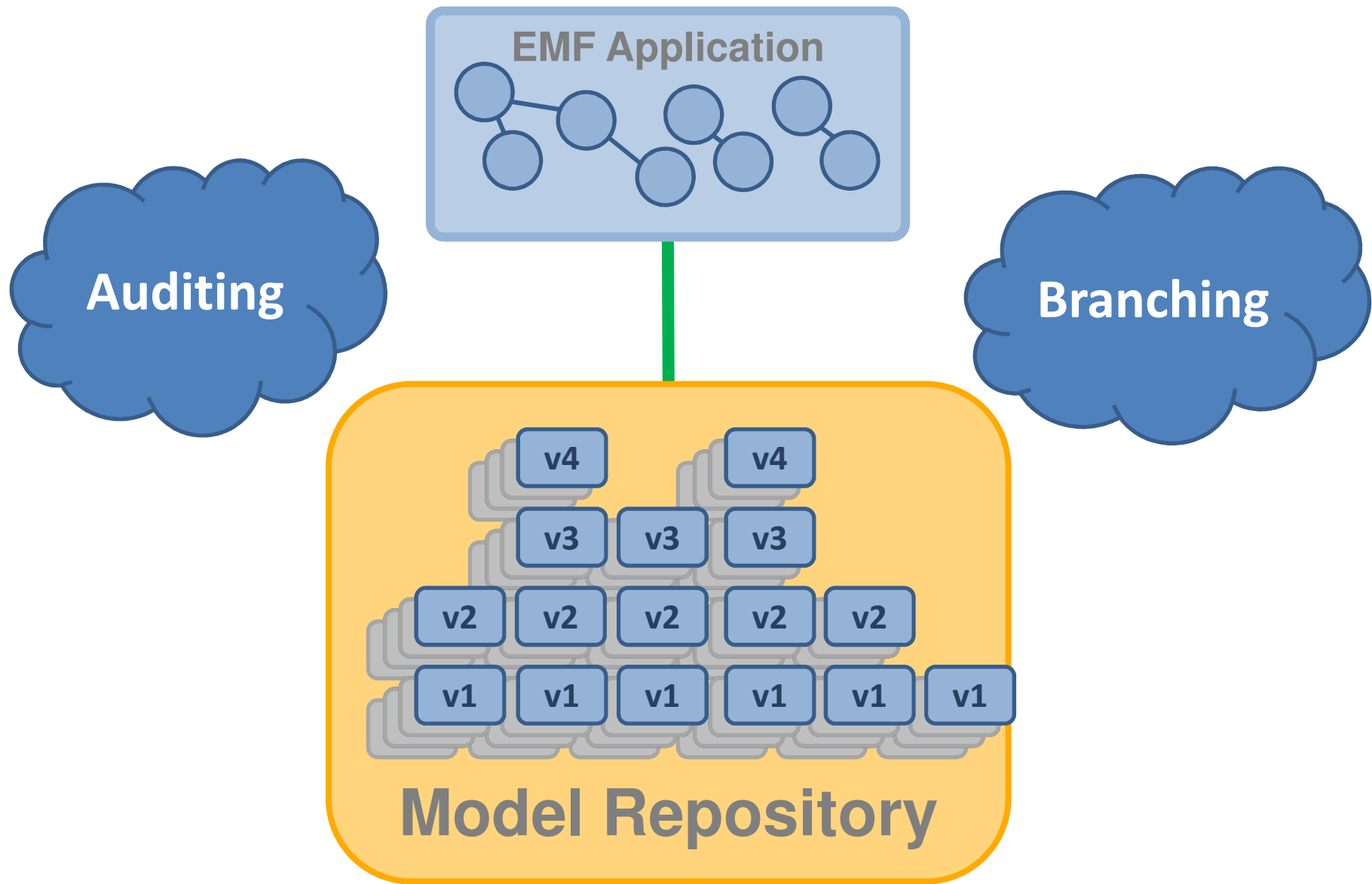












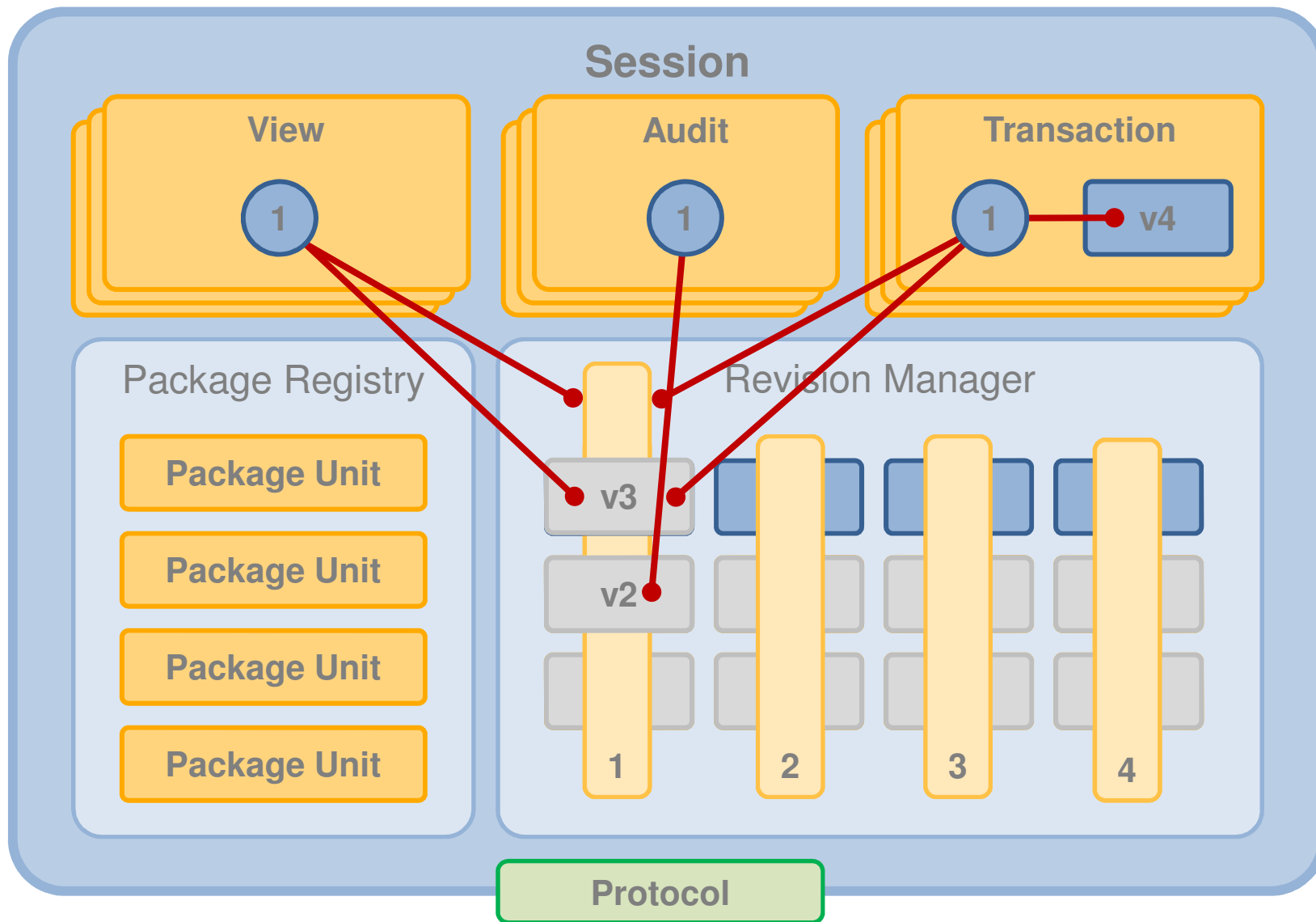
CDORevision

EClass eClass
CDOID id
CDOBranch branch
int version
long created
long revised

Revision Data

CDOID resourceID
CDOID containerID
int containerFeature
Object[] values

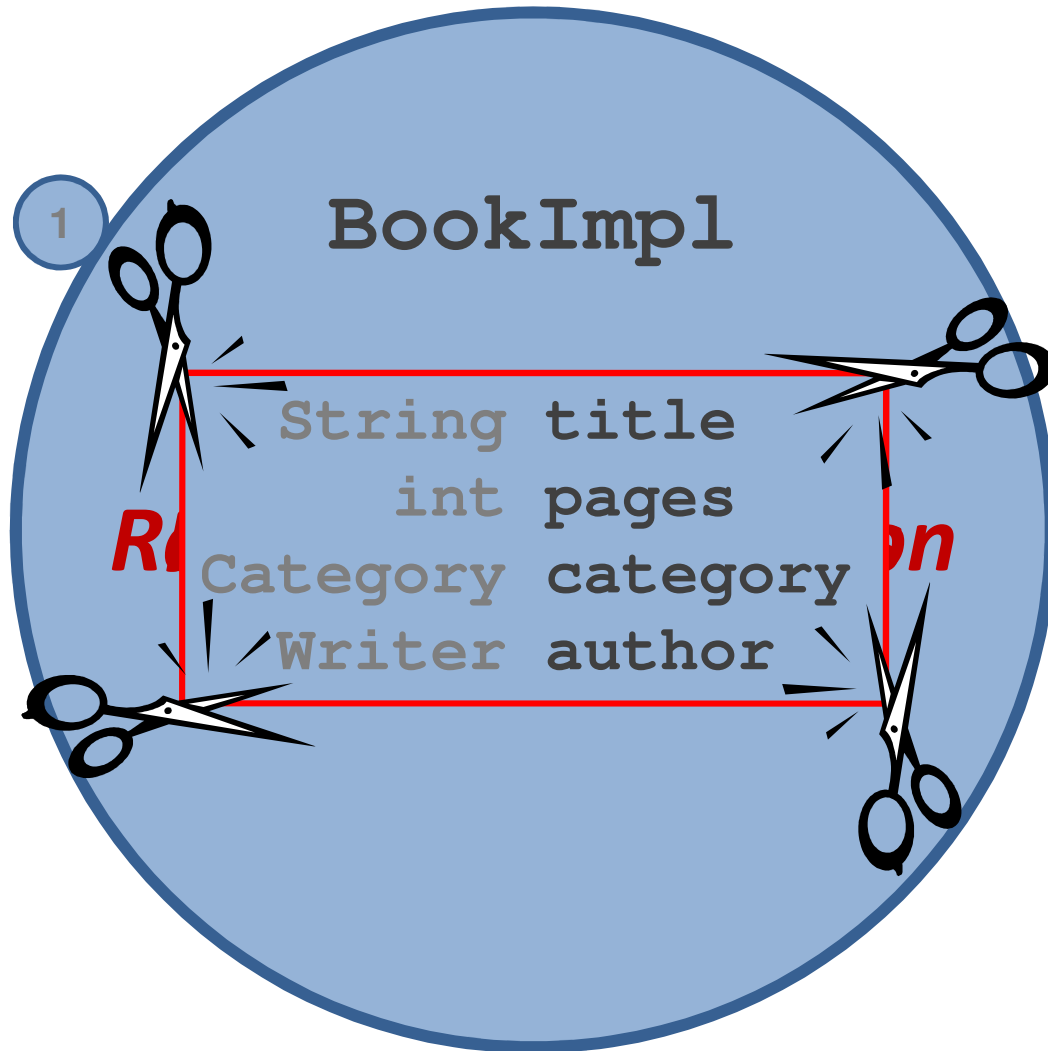


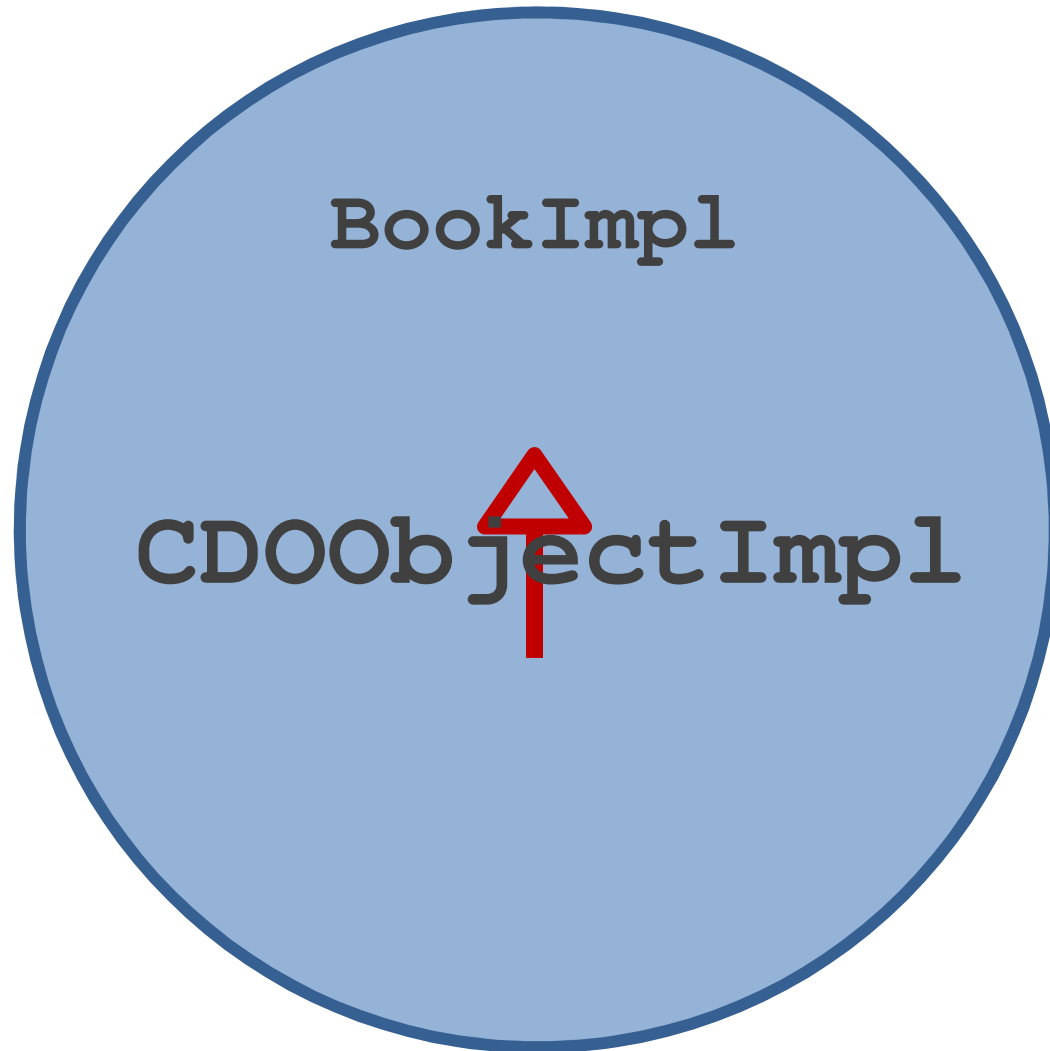


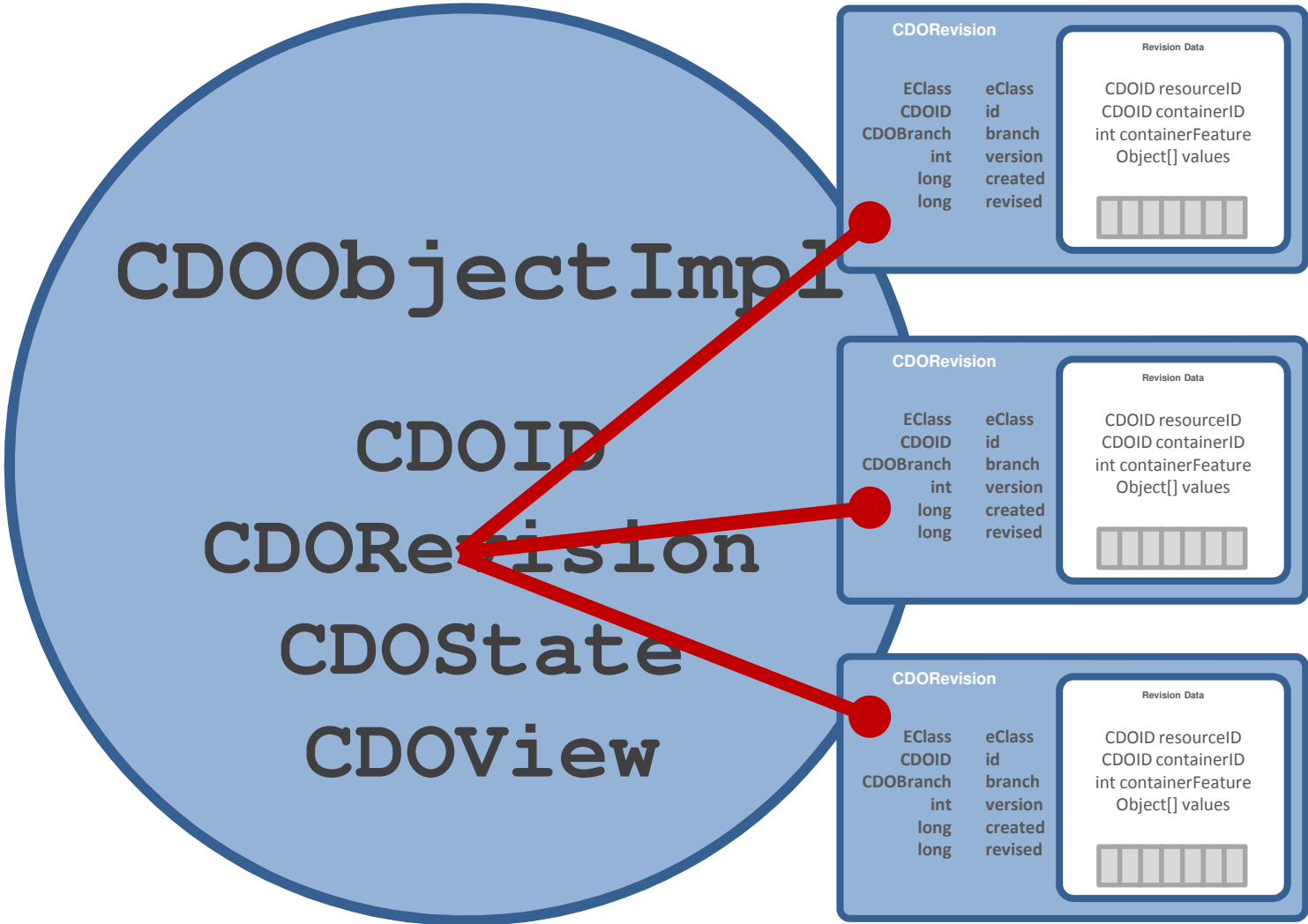


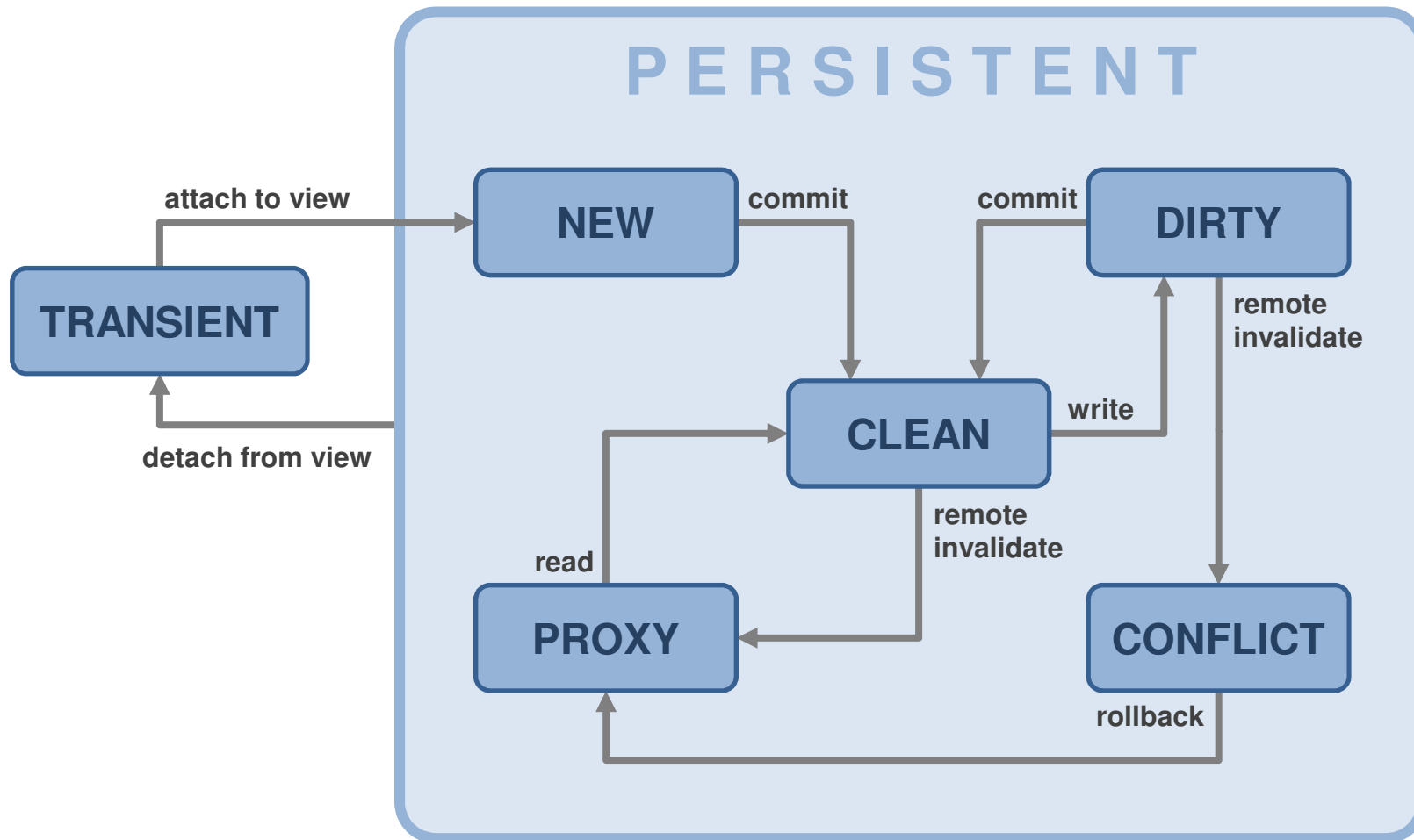
Technical Challenges:

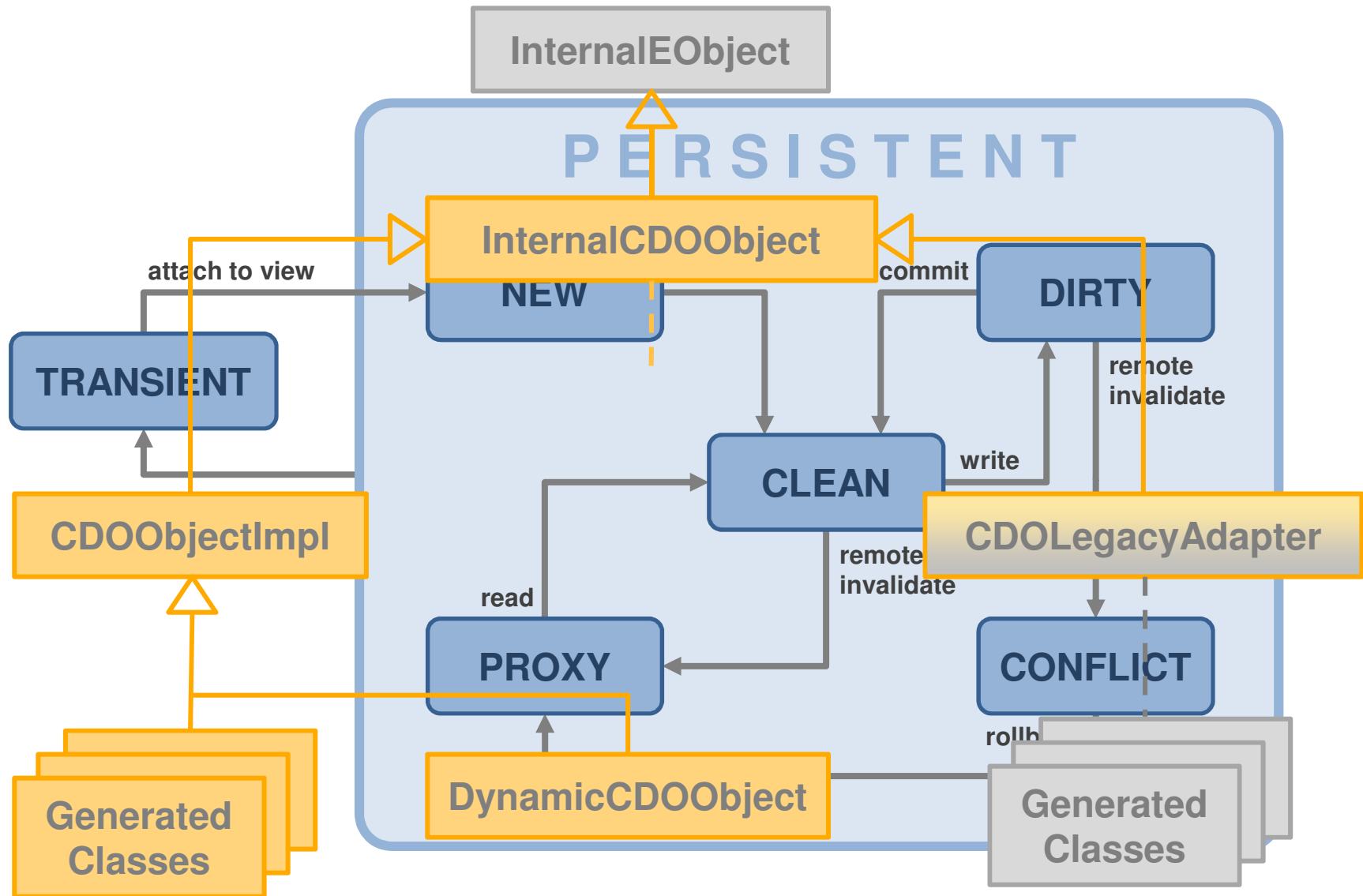
- **Transfer revisions over the network**
- **Swap revisions on remote invalidation**
- **Swap revisions when changing view time**
- **Swap revisions when changing view branch**
- **Make objects reclaimable by GC**

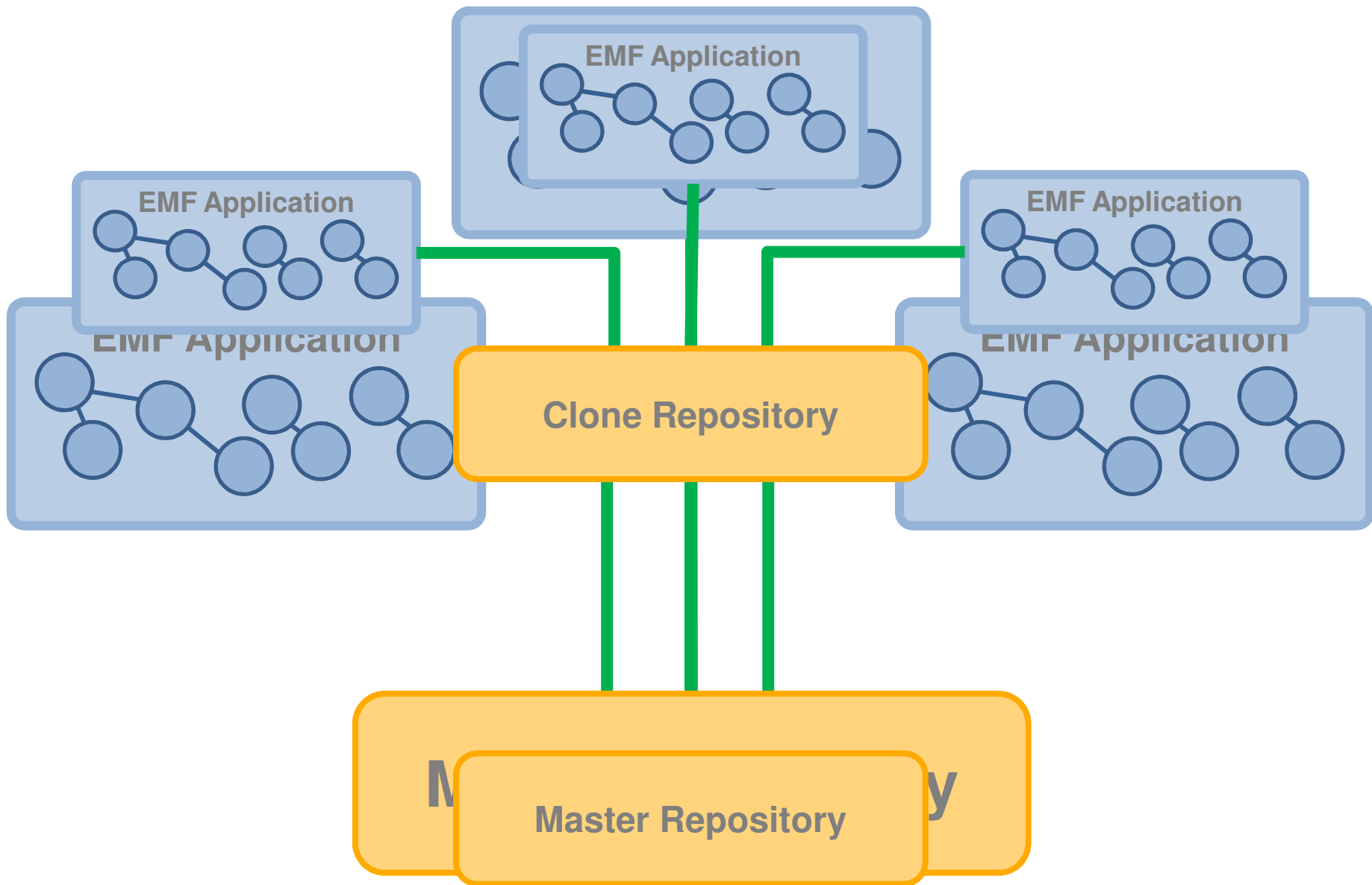


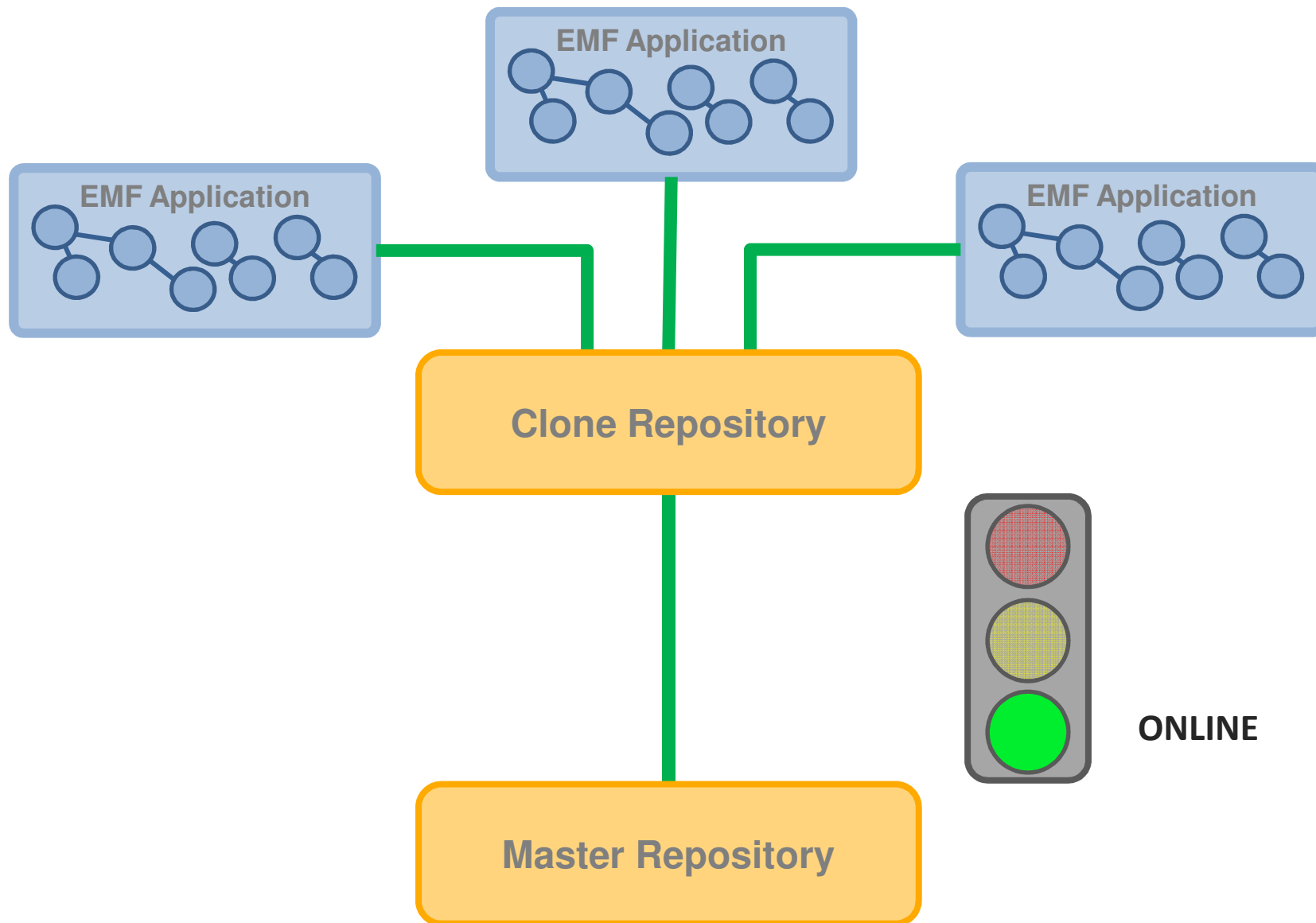


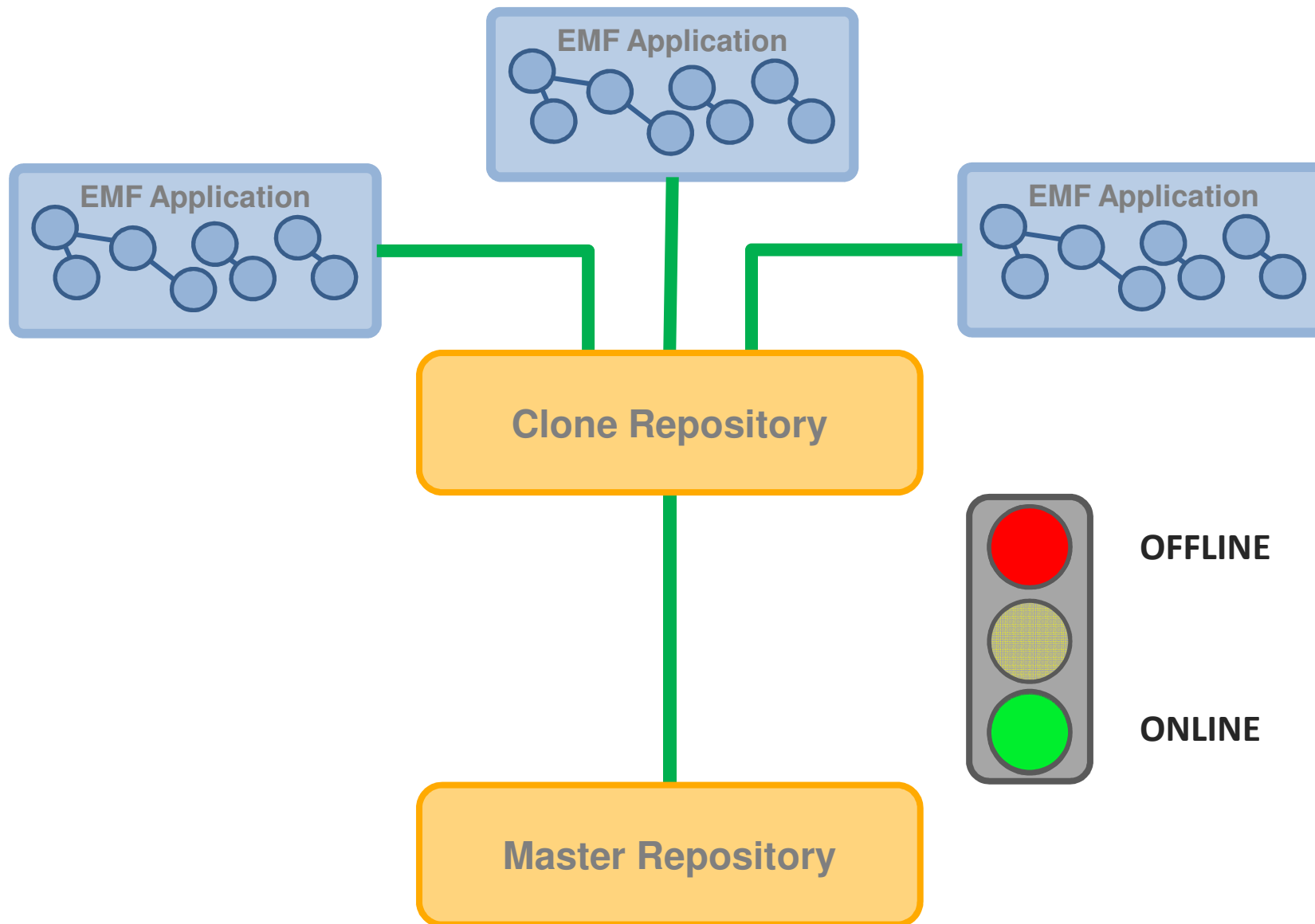


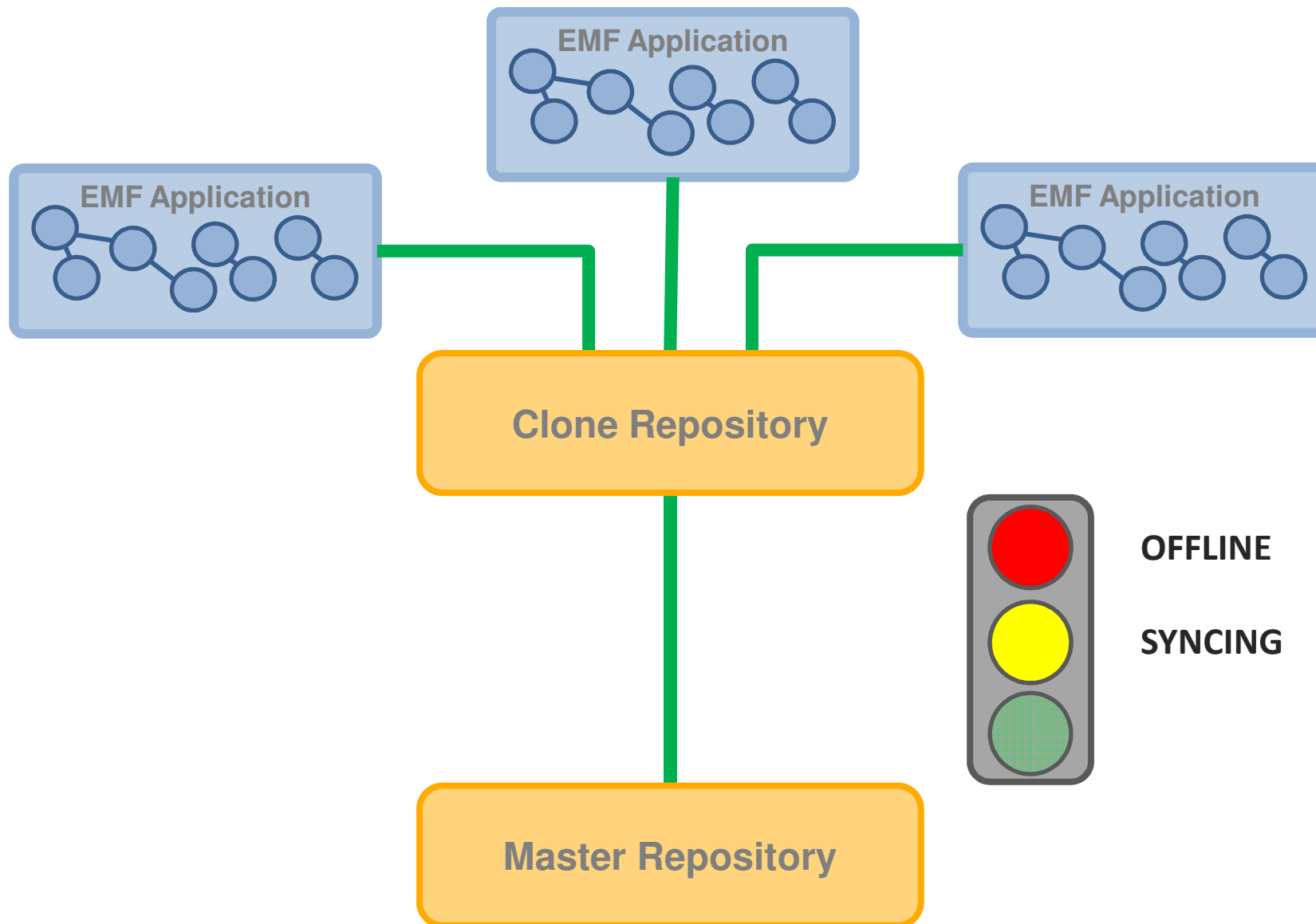


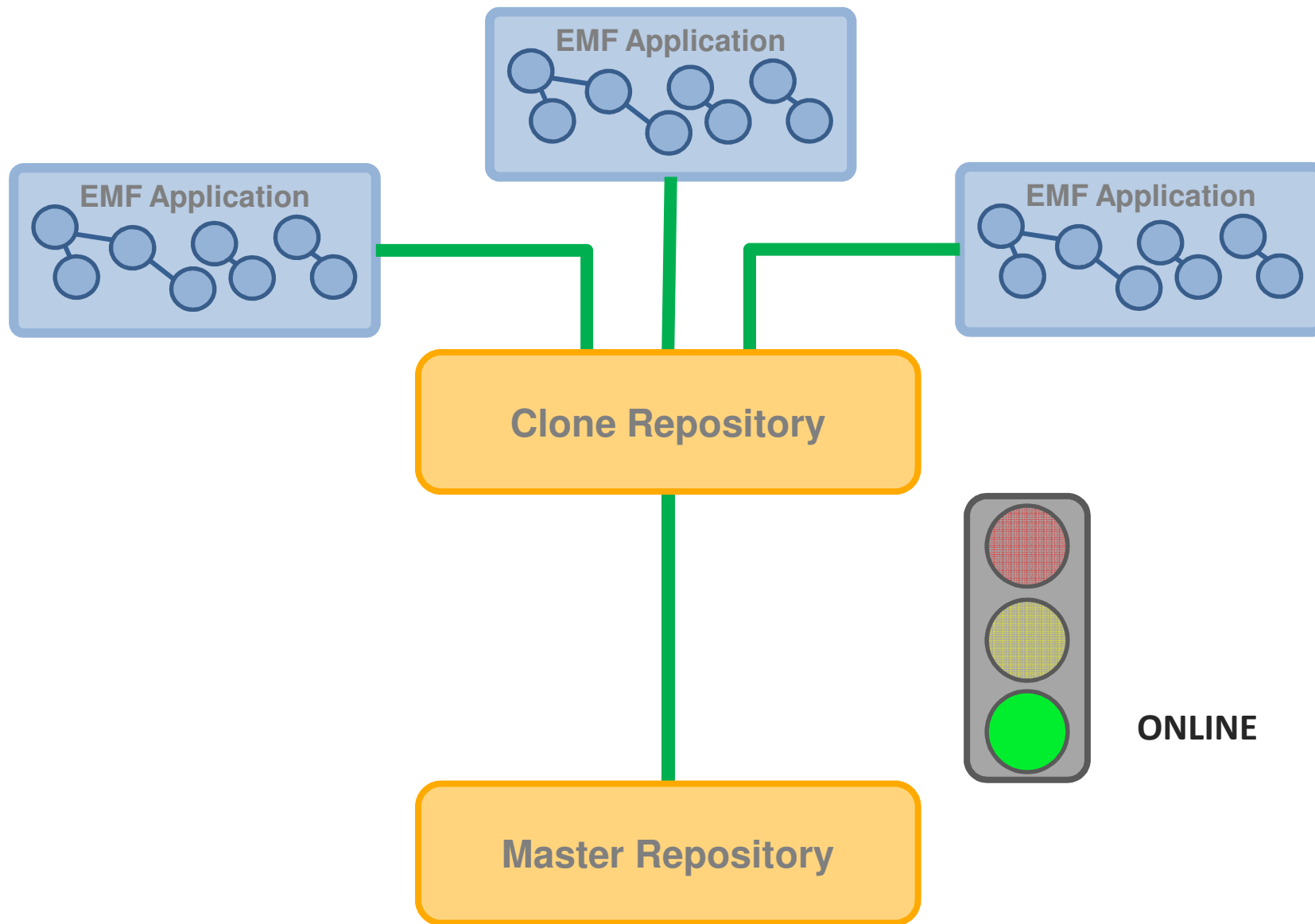


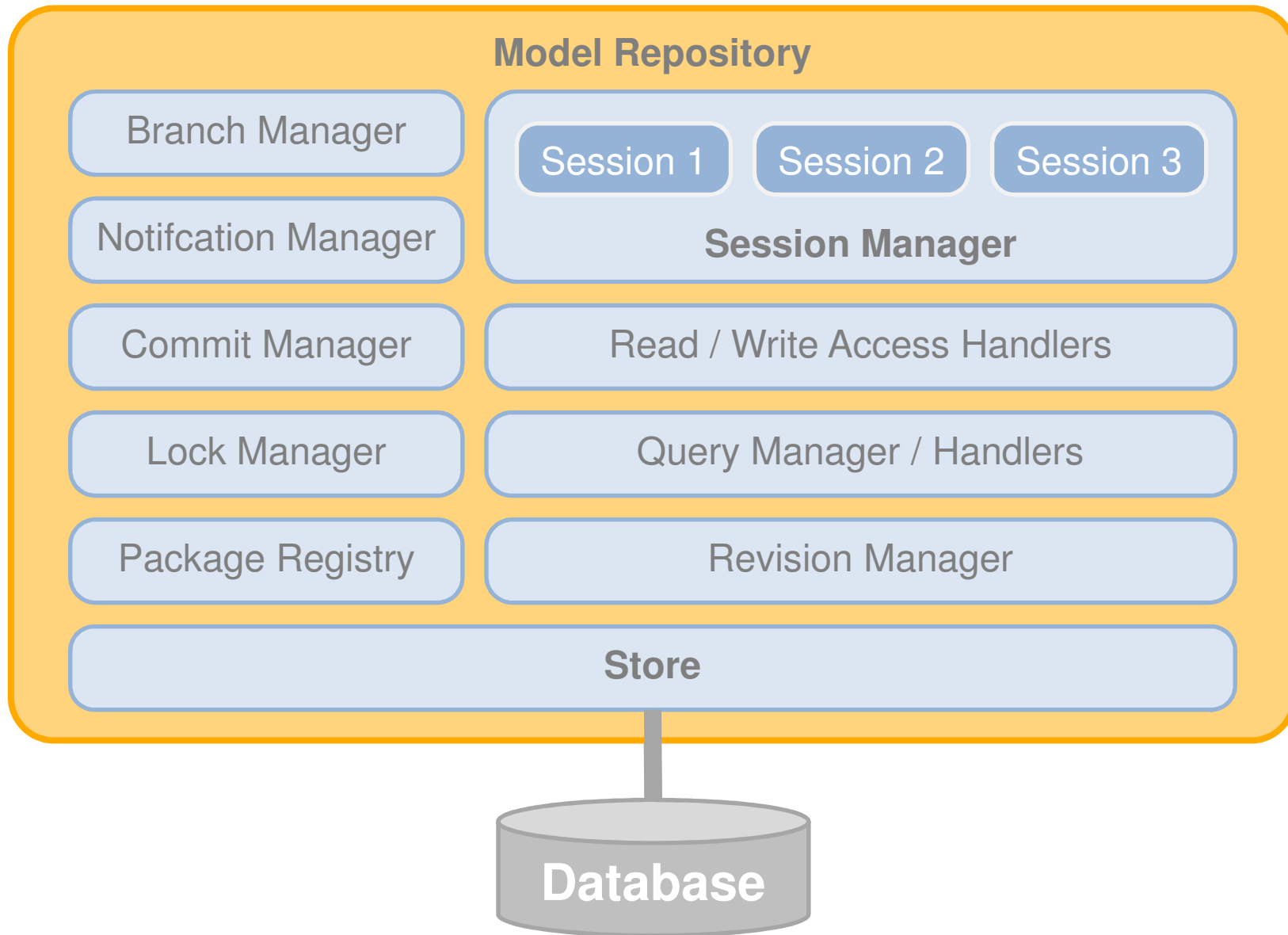


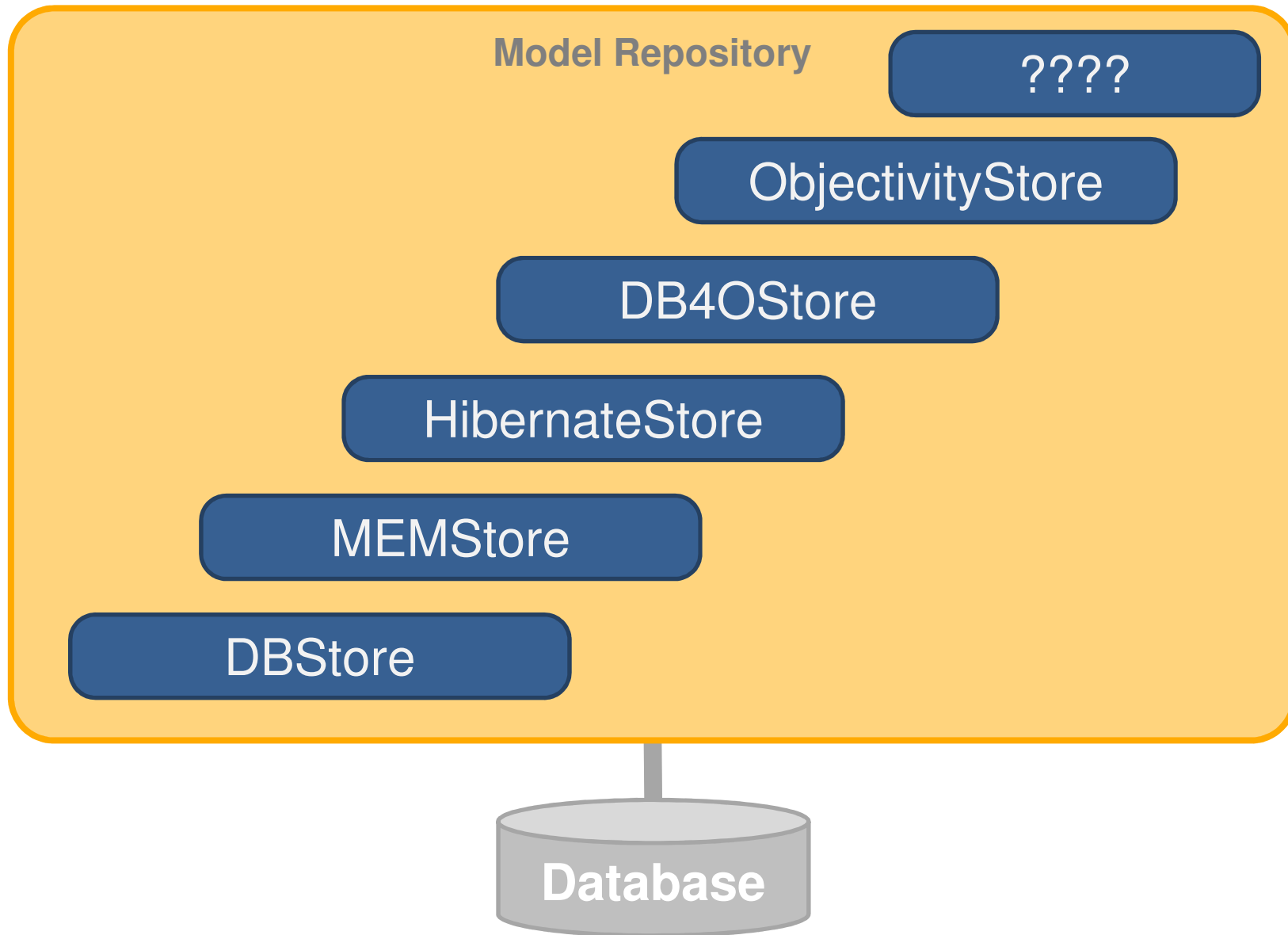












CDO Core Features

Distribution

- **Various ways to set up an IRepository**
 - XML config file, programmatically, Spring, ...
 - OSGi, stand-alone, ...
 - All components customizable
- **Various ways to open a CDOSession**
 - Net4j: TCP, HTTP, embedded, ...
 - CDO: embedded
 - Other transports possible
- **Offline mode coming soon**
 - Cloned and sync'ed repository, normal sessions

Persistence

- **Pluggable storage backend adapters (IStores)**
 - DBStore (CDO's own O/R mapper)
 - HibernateStore / Teneo
 - ObjectivityStore
 - DB4OStore
 - MEMStore
- **Changing the store type does not affect client applications!**

Resources

- **A CDOResource is an EObject**
- **A repository contains CDOResourceNodes**
 - CDOResourceFolders
 - CDOResources
- **The resource tree is**
 - Navigable through EMF
 - Queryable through CDO

Versioning

- **CDO supports record temporality**
 - Must be supported by IStore
 - Can be configured per IRepository
- **CDO supports branching**
 - Must be supported by IStore
 - Can be configured per IRepository
- **A CDOView provides consistent graphs**
 - From a particular branch
 - From a particular point in time

Scalability

- **Lazy loading at object granule**
- **Lazy loading without container object**
- **Partial collection loading, chunking**
- **Adaptive prefetching**
- **Manual prefetching**
- **Automatic unloading at object granule**

Queries

- **CDO includes a generic query framework**
 - Supports any query language
 - Supports named parameters
 - Supports synchronous execution
 - Supports asynchronous execution
- **Query language handlers can be**
 - plugged into an IRepository (OCL?, EMF-Q?, ...)
 - implemented by an IStore (SQL, HQL, custom, ...)

Transactionality

- **Strong transactional safety at model-level**
- **Multiple transactions per session**
- **Multiple save points per transaction**
- **Rollback to any save point**
- **Commit with progress monitoring**
- **Hooks for custom transaction handlers**
- **Conflict detection and fail-early-transactions**
- **Pluggable conflict resolvers**
- **Explicit read/write locking on object granule**
- **XA transactions to multiple repositories**

Collaboration

- **Passive Updates**
 - Asynchronous commit notifications
 - Invalidation of objects, lazy reload if needed
 - Can be switched off per session
- **Change subscriptions**
 - Asynchronous change delta delivery
 - Registration with repository per object
 - Automated through pluggable adapter policies
- **Remote session manager**
 - Notifies about state of other sessions
 - Supports sending/receiving of arbitrary messages

Integration

- **Integrates with EMF at the model level, not at the edit- or UI-level.**
- **Uninvasive to the .ecore file.**
- **Best results with regenerated models (native)**
- **Regeneration not needed (legacy)**
- **Dynamic models supported**
- **Multiple repositories per ResourceSet**
- **External references**

Dawn – Rise of Graphical Collaboration

The image displays two side-by-side windows illustrating graphical collaboration in UML modeling.

The left window is a web browser showing a URL: `http://localhost:8080/DawnServer/showDiagram.do?projectname=my.classdi`. The browser interface includes a menu bar with "Lesezeichen", "Extras", and "Hilfe", and a navigation bar with "Home", "Projects", "Users", "Roles", and "myProjects". The main content area shows a UML class diagram with the following elements:

- MyInterface** (Interface): `public printName():void`
- MySubClass** (Class): Inherits from **MyInterface**.
- MyClass** (Class): Inherits from **MyInterface**. Attributes: `private name:String`; Operation: `public getName():String`.
- MyAssociation** (Association): Connected to **MyClass**.
- MyComposition** (Composition): Connected to **MyClass**.
- MyAggregation** (Aggregation): Connected to **MyClass**.

The right window is the Eclipse IDE, titled "Java - ClassDiagram/my.classdiagram_diagram - Eclipse SDK". The IDE interface includes a menu bar (File, Edit, Diagram, Navigate, Search, Project, Tomcat, Run, Window, Help), a toolbar, and a text editor showing "Tahoma" font, size "9", with bold and italic options. The main editor area displays the same UML class diagram as the browser. The left sidebar shows a project tree with "ClassDiagram", "src", "JRE System", "my.classdi", "Dawn", and "UML". The right sidebar contains a "Palette" with "Node" (Class, Interface, AnAttribute, AnOperation) and "Connecti..." (inherits, implements, association, aggregation, composition). The bottom status bar shows "Server Online" and "Aclass MySubClass".

Scale, Share and Store your Models with CDO

© 2010 by Eike Stepper, Berlin, Germany. Made available under the EPL v1.0

Dawn – Rise of Graphical Collaboration

- Conflict handling
 - Dawn provides detection and handling mechanisms for conflicts
 - It will build on the CDO conflict mechanisms and provide flexible and intuitive UI to handle conflicts
 - Conflicts are displayed inside the diagram editor. Conflicts that cannot be visualized inside the editor will be show in a special view (Dawn Conflict View)
- Locking
 - Dawn will support locking on different hierarchy levels in the GMF diagram
 - Locked objects are marked with special visualisations
- WebViewer/WebEditor
 - Dawn provides a web viewer to view changes in the diagram while they are processed in Eclipse
 - It also will support changing the diagram (adding/deleting/manipulating) in a browser
 - Allows editing GMF-diagrams on mobile devices even if no Java platform is installed

Dawn – Rise of Graphical Collaboration

- Do not change existing code
 - A dynamic design and a flexible generator will make it possible to “collaborate” existing GMF editors even if the source is
 - Existing editor do not need to modified
- Firewall transparency mode
 - Allows to operate from within restricted networks
 - This mode will use a web-based protocol on CDO
- Network independence (Offline Mode)
 - Using one of the latest CDO features (offline support) Dawn will allow modifying GMF diagrams without a repository connection.
- Authentication/Authorization
 - Providing access rights on diagram level will allow to protect your model data
 - Additionally the use of the diagram (show, modify, view) will be restrictable. Locking behaviour can also be influenced.