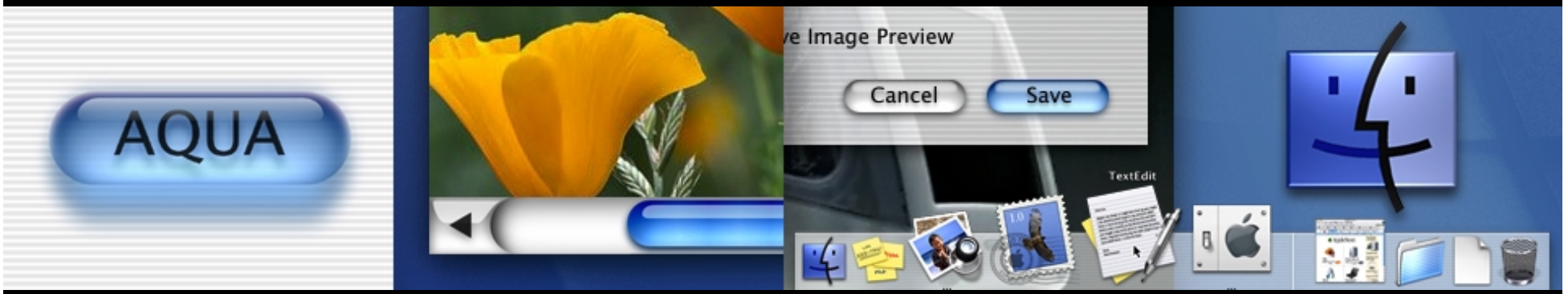
Session 412

# WebObjects: Caching and Synchronization

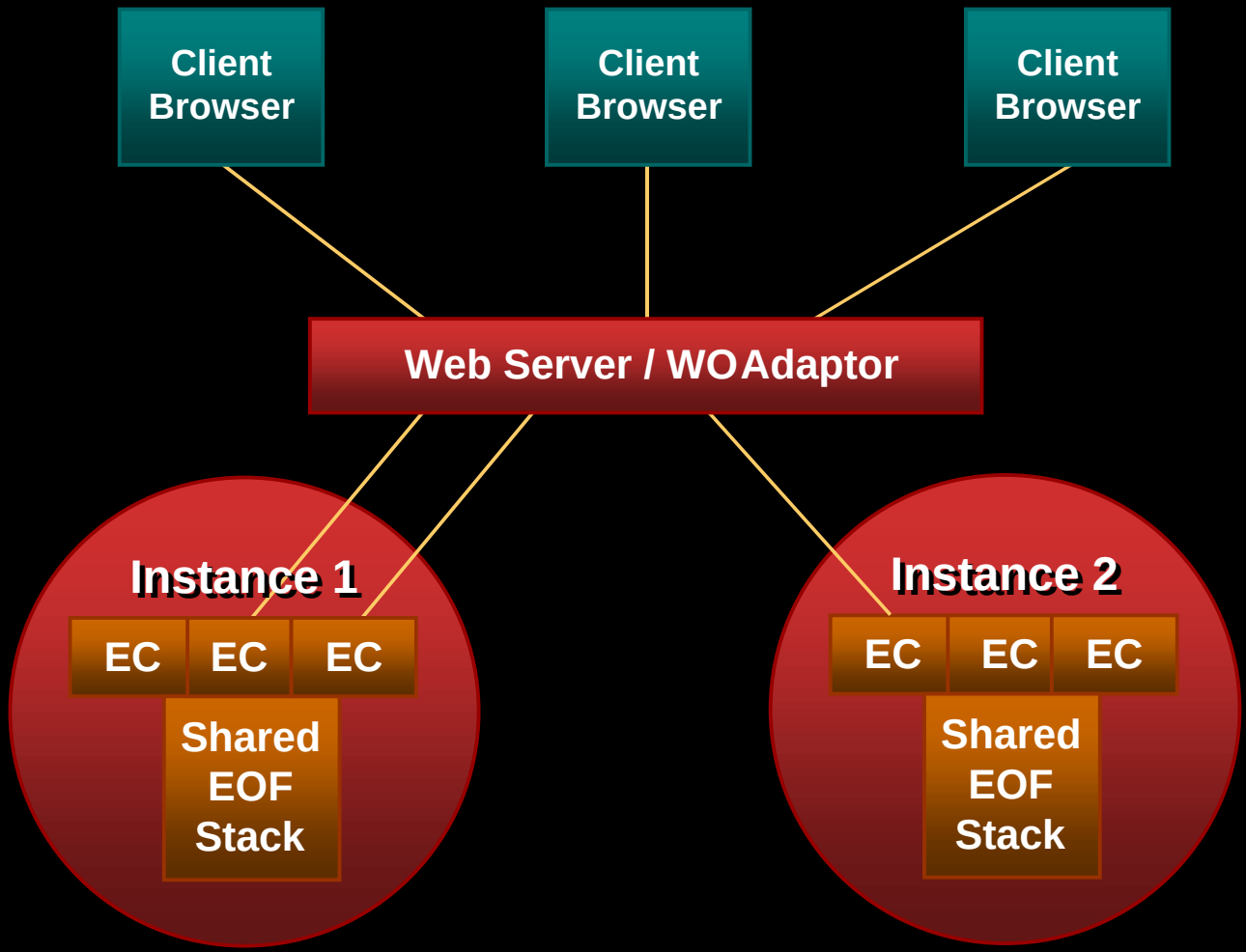



**Daniel Abrams**
**Senior Consulting Engineer, Apple iServices**

# Introduction: The Issues

- Default WebObjects/EOF deployment scenario
- Fetching and snapshotting
- Committing and receiving changes
- Coordinating updates

# Default Deployment

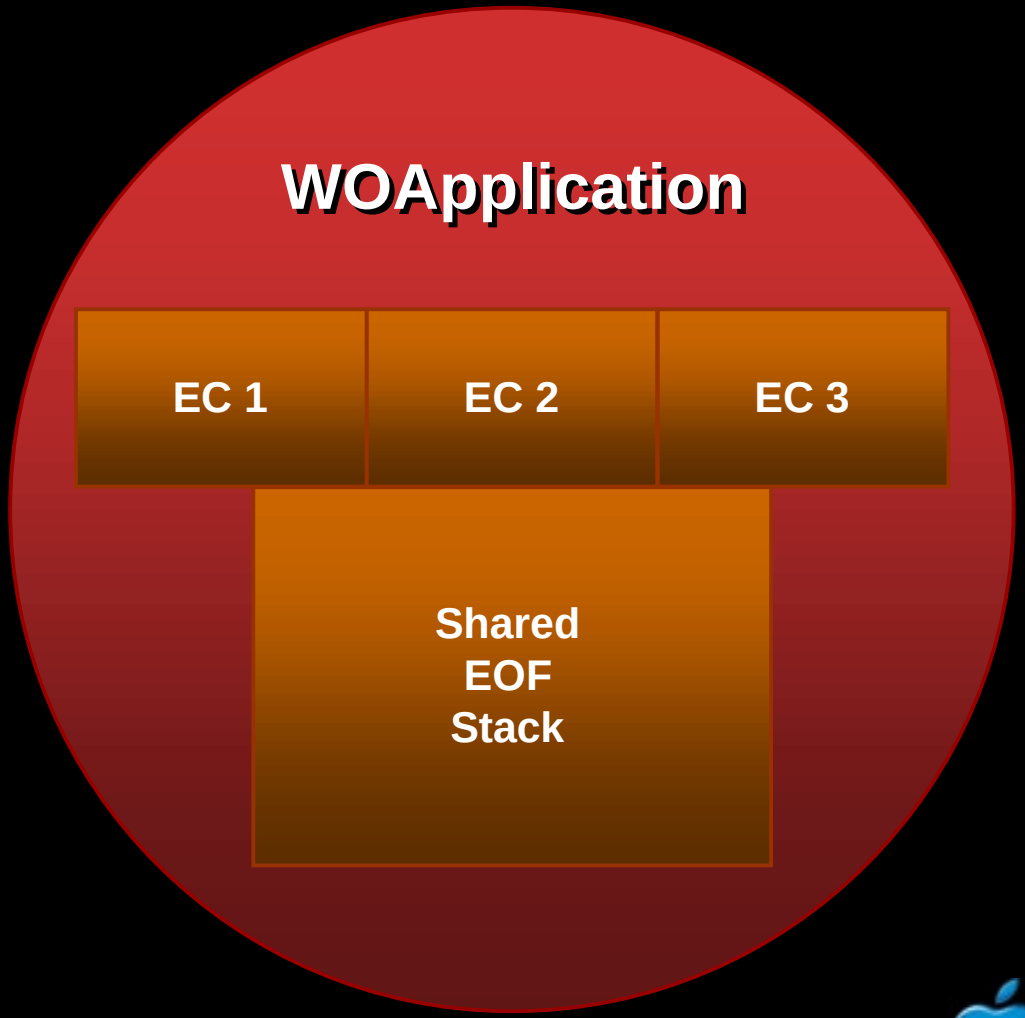


# Understanding Snapshots

- Key to all caching and synchronization
- Master repository for coordinating data retrieval and updates

# How Snapshots Work

By default editingContexts share a common set of snapshots.



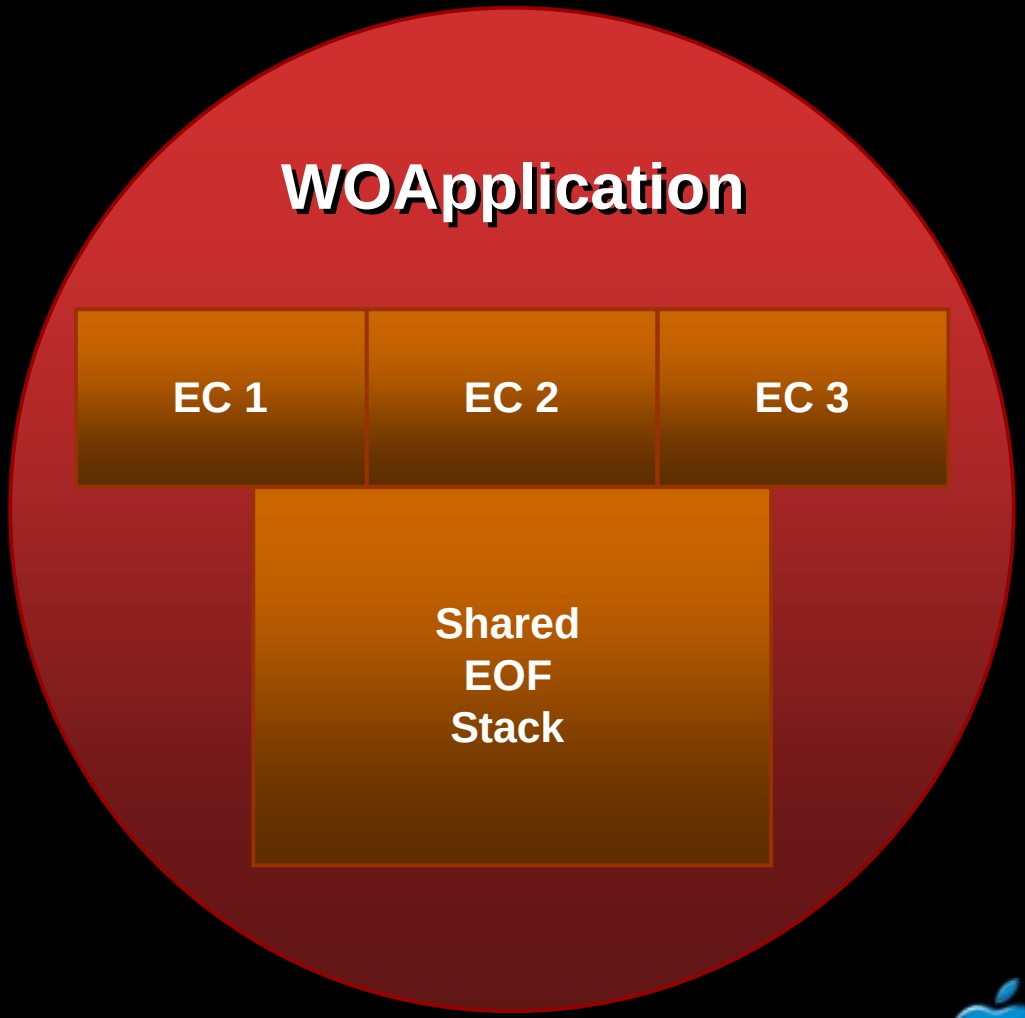

# Fetching

- My users complain that they aren't seeing the latest data
- Why do I see the application hit the database, but still see older data?

# Simple Fetch: New Snapshot

EC 1 Fetch:

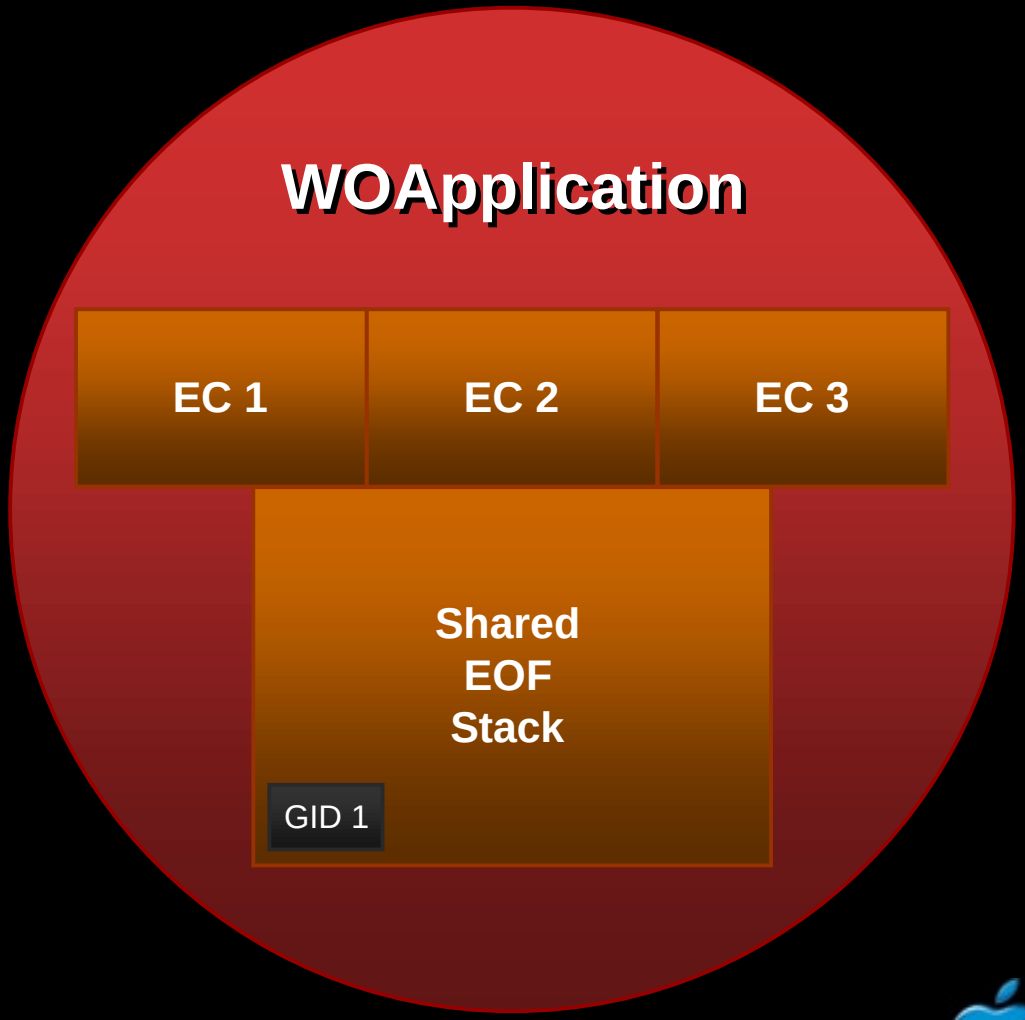
**Query Database**
Snapshot GID 1
Object in EC1




# Simple Fetch: New Snapshot

EC 1 Fetch:

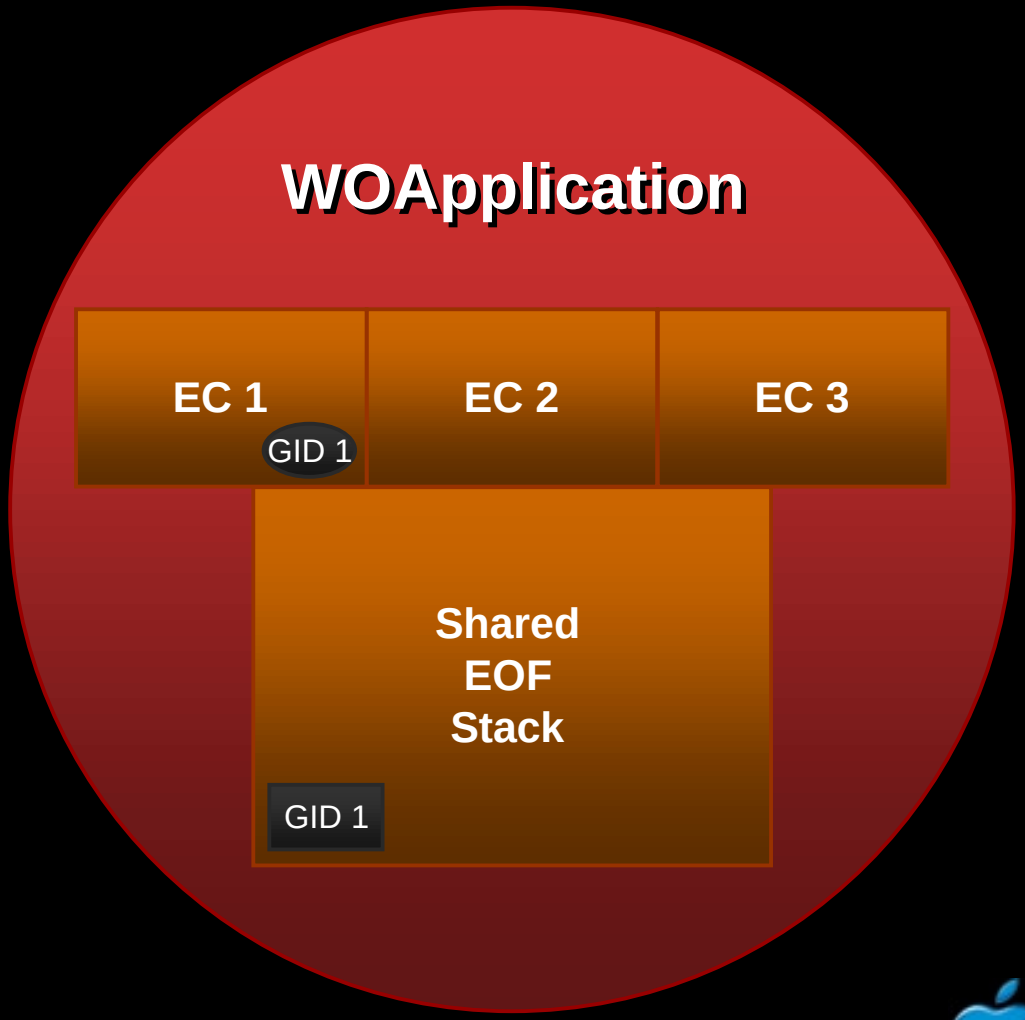
Query Database
**Snapshot GID 1**
Object in EC1




# Simple Fetch: New Snapshot

EC 1 Fetch:

Query Database
Snapshot GID 1
**Object in EC1**

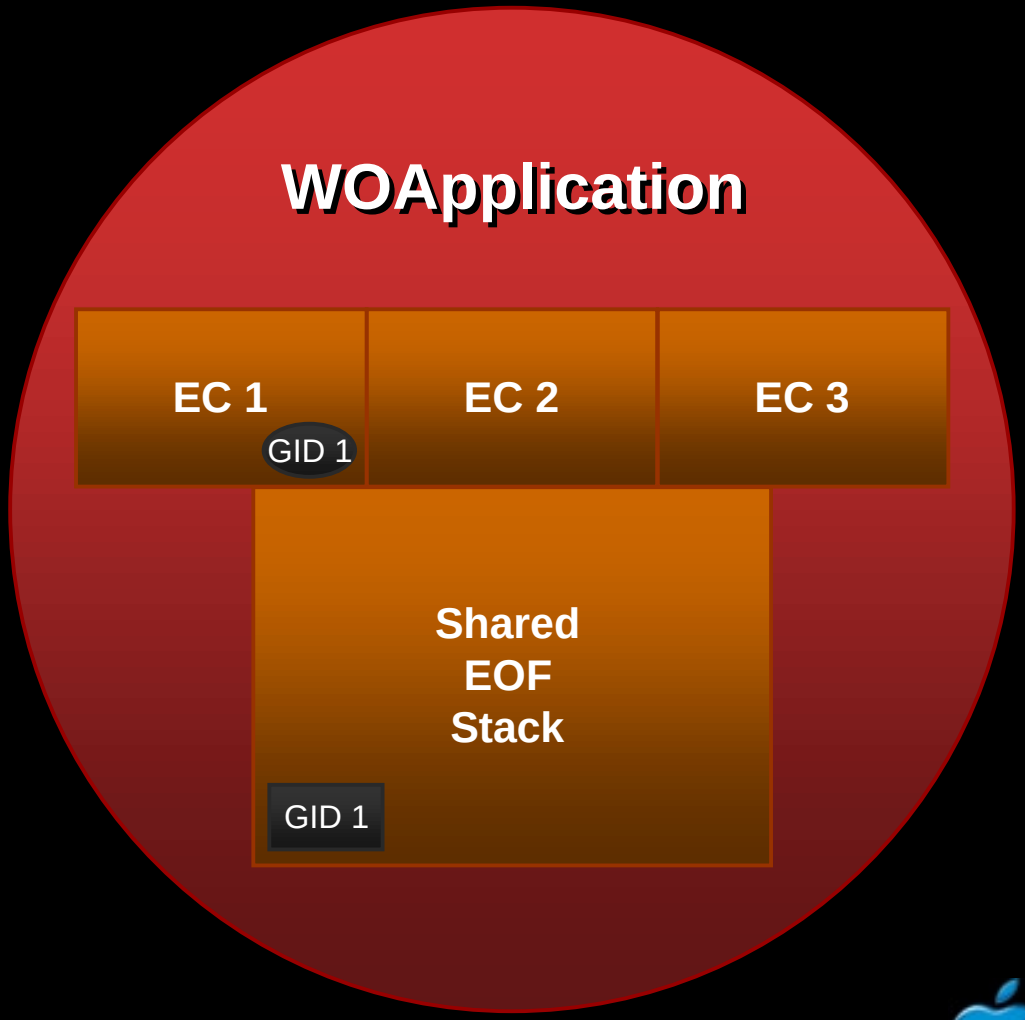


# Simple Fetch: Existing Snapshot

EC 2 Fetch:

**Query Database**
**Ignore Updates**

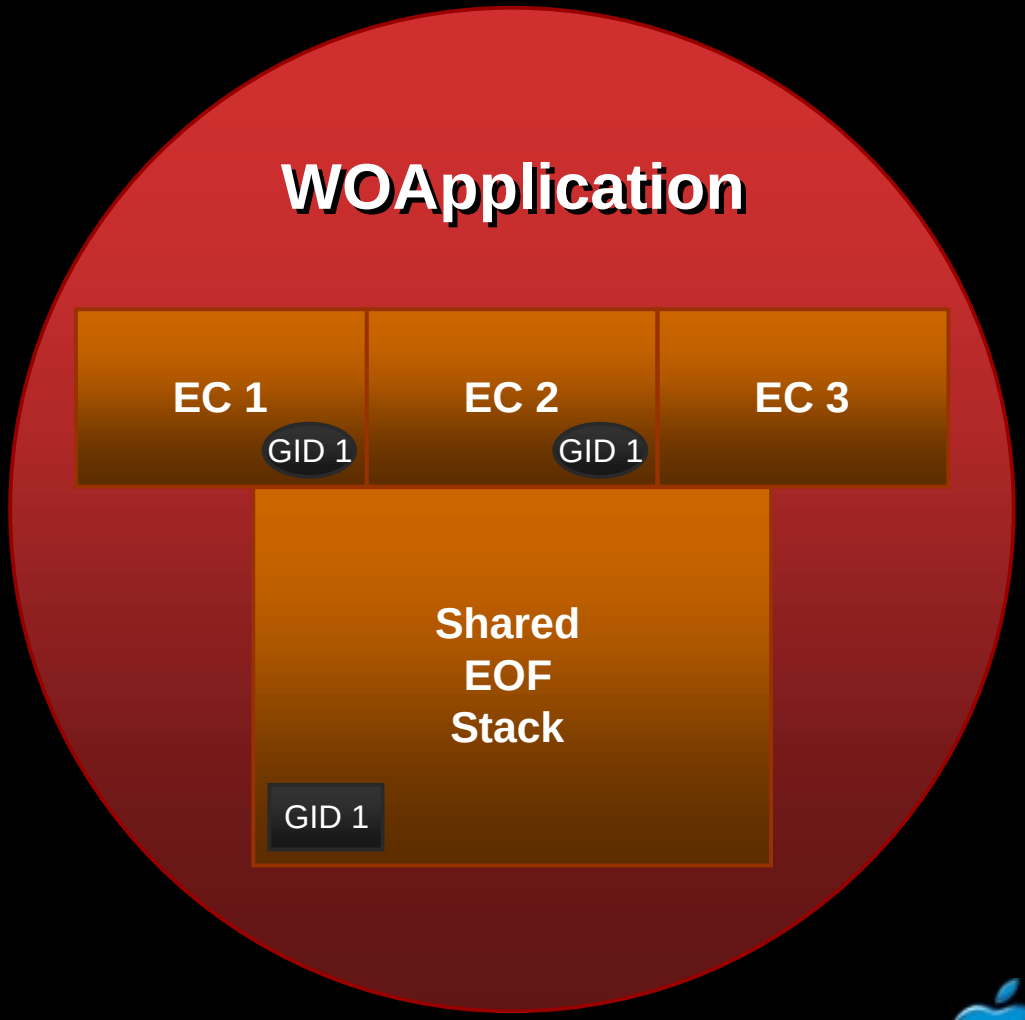
Object in EC2: created from snapshot




# Simple Fetch: Existing Snapshot

EC 2 Fetch:

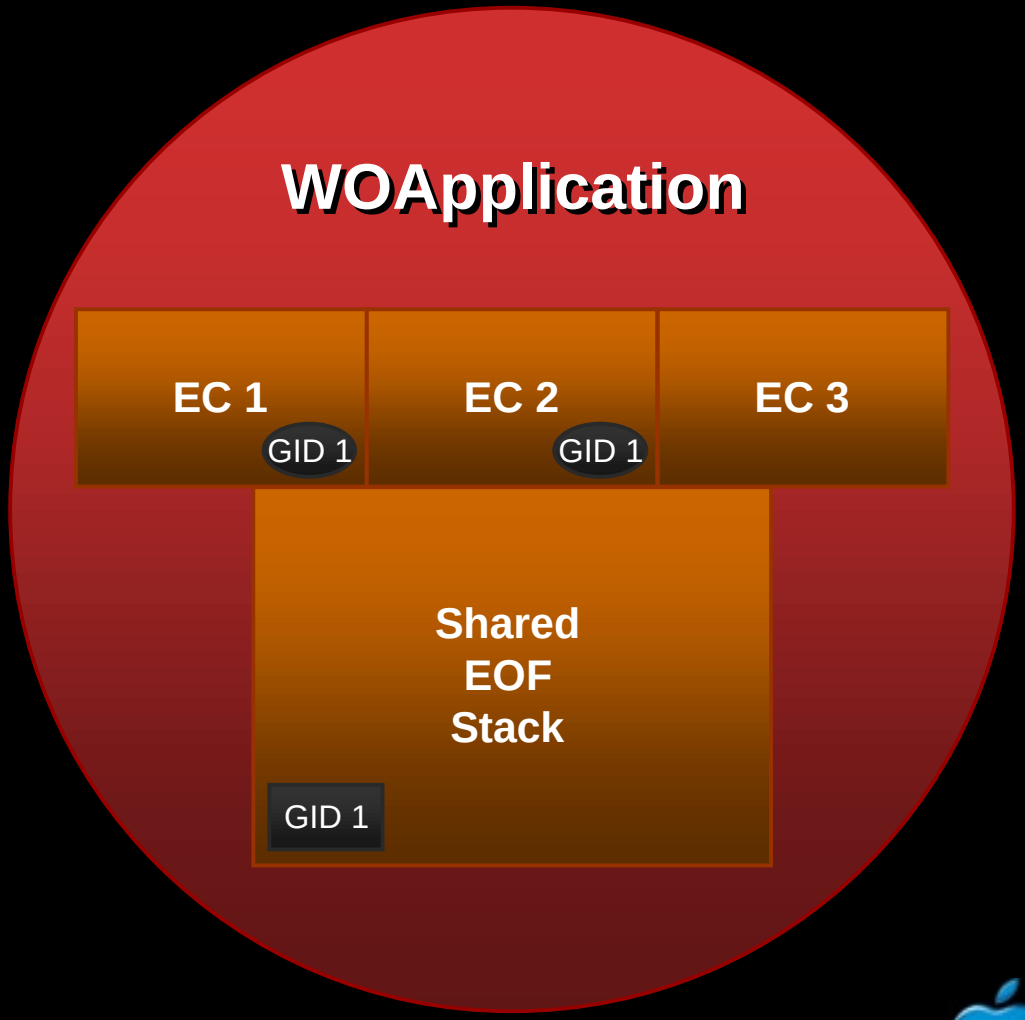
Query Database
Ignore Updates
**Object in EC2:** **created from snapshot**



# Fetch With Refresh

EC 3 Fetch/Refresh:

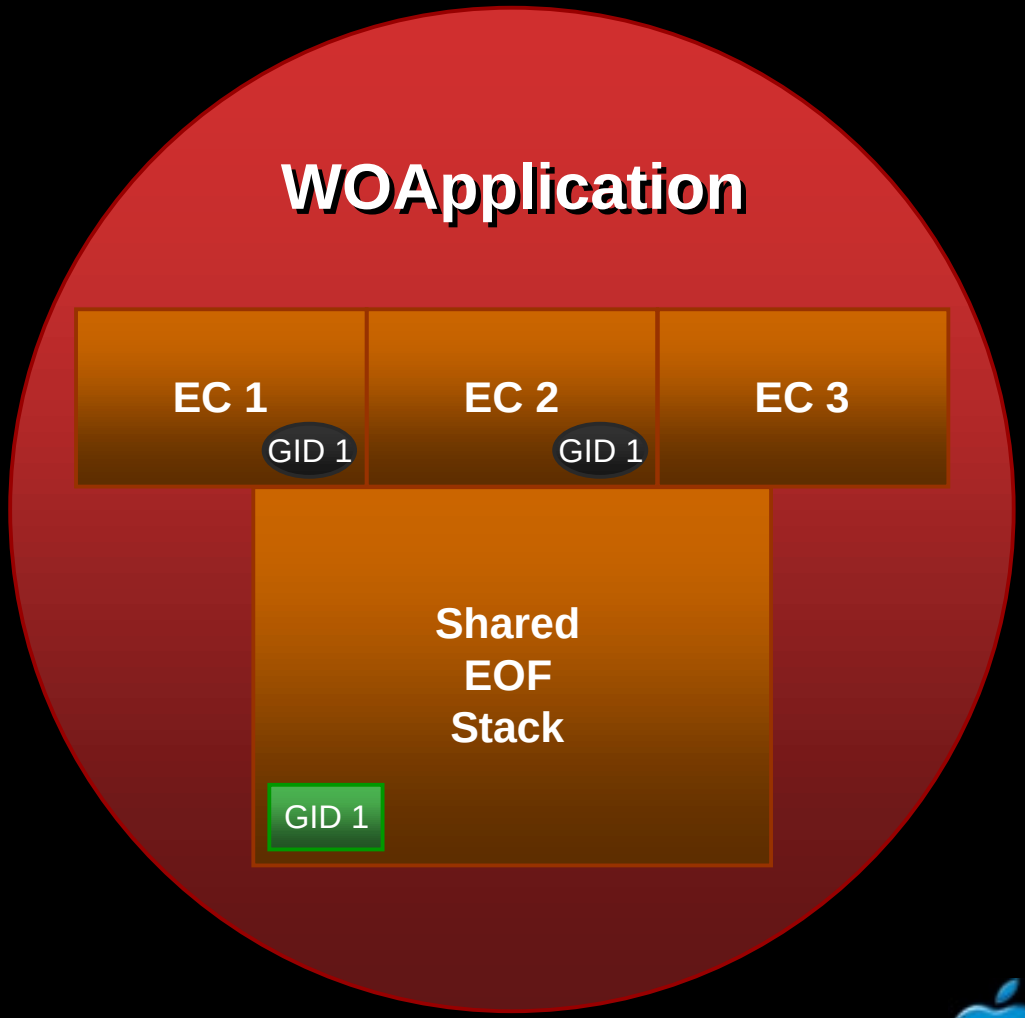
**Query Database**
Update Snapshot
Broadcast Changes
Object in EC3




# Fetch With Refresh

EC 3 Fetch/Refresh:

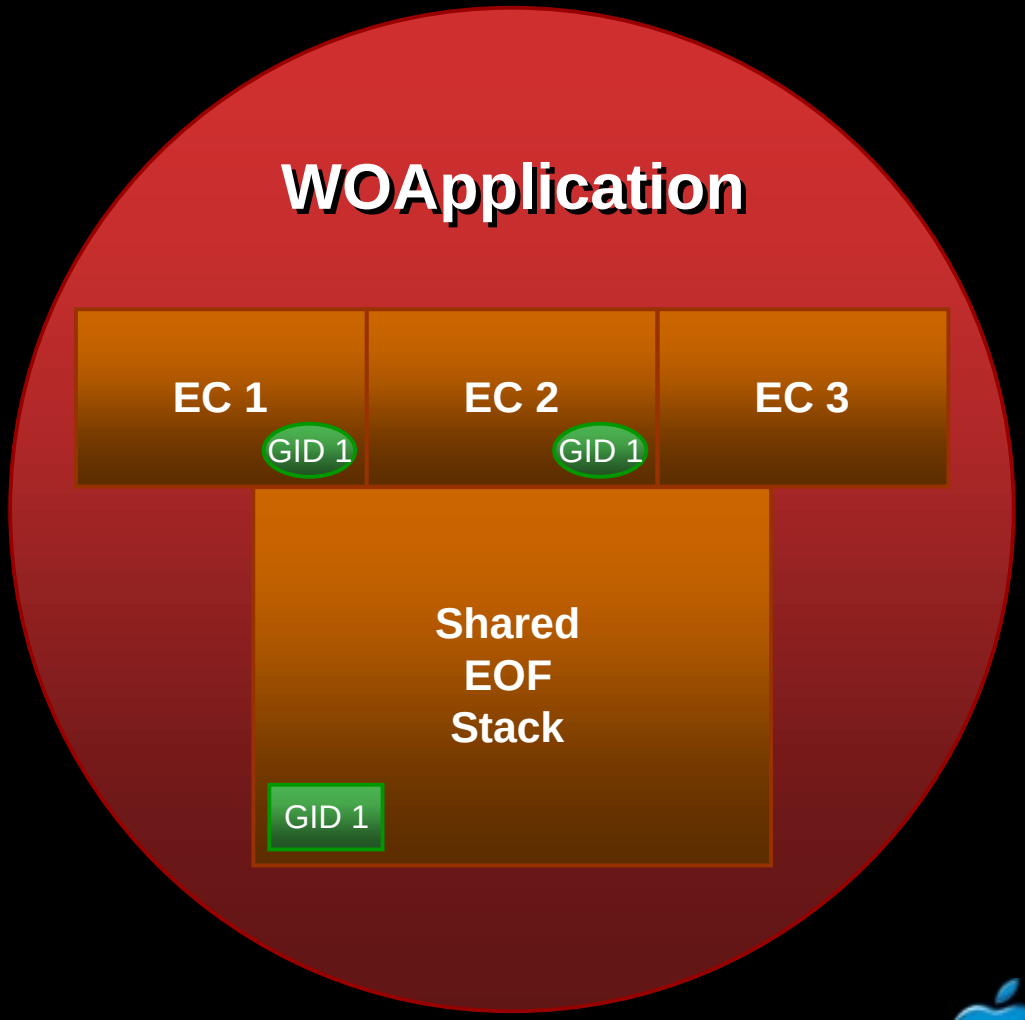
Query Database
**Update Snapshot**
Broadcast Changes
Object in EC3




# Fetch With Refresh

EC 3 Fetch/Refresh:

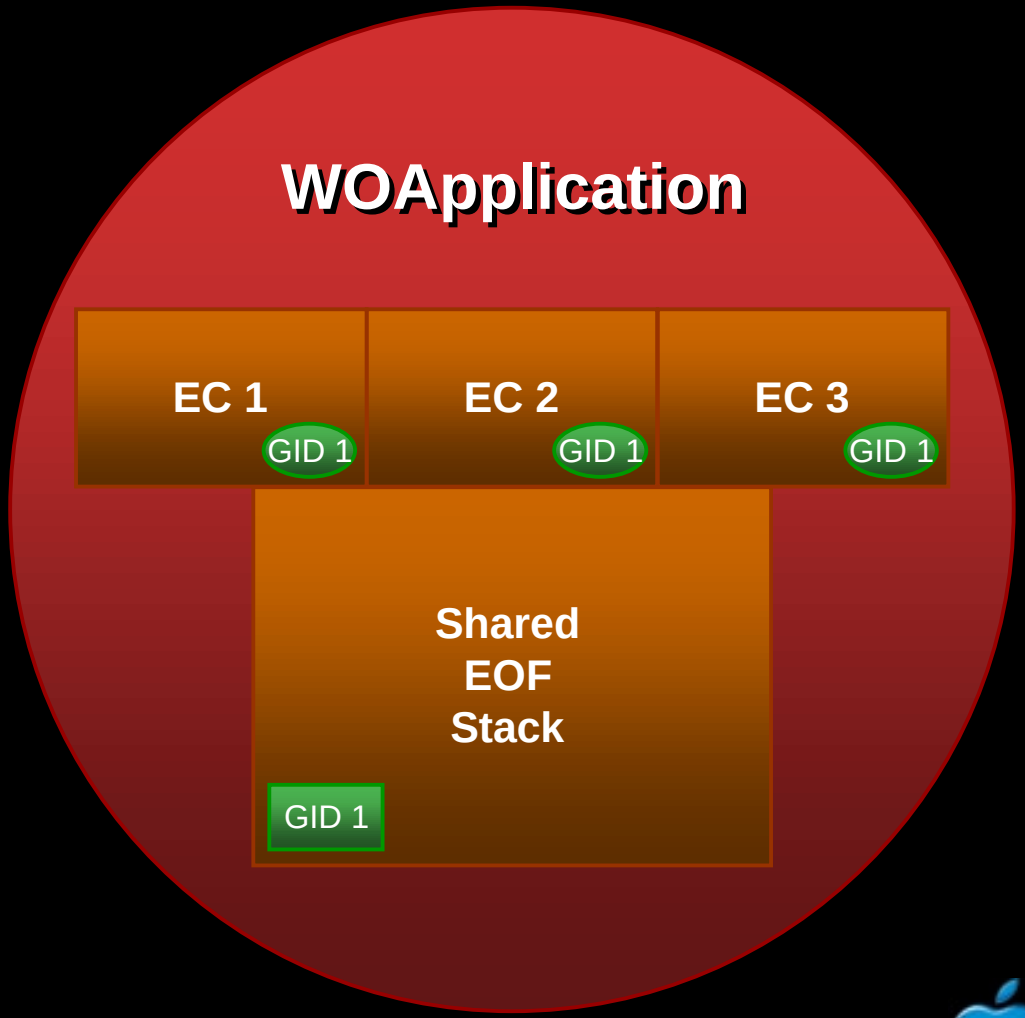
Query Database
Update Snapshot
**Broadcast Changes**
Object in EC3




# Fetch With Refresh

EC 3 Fetch/Refresh:

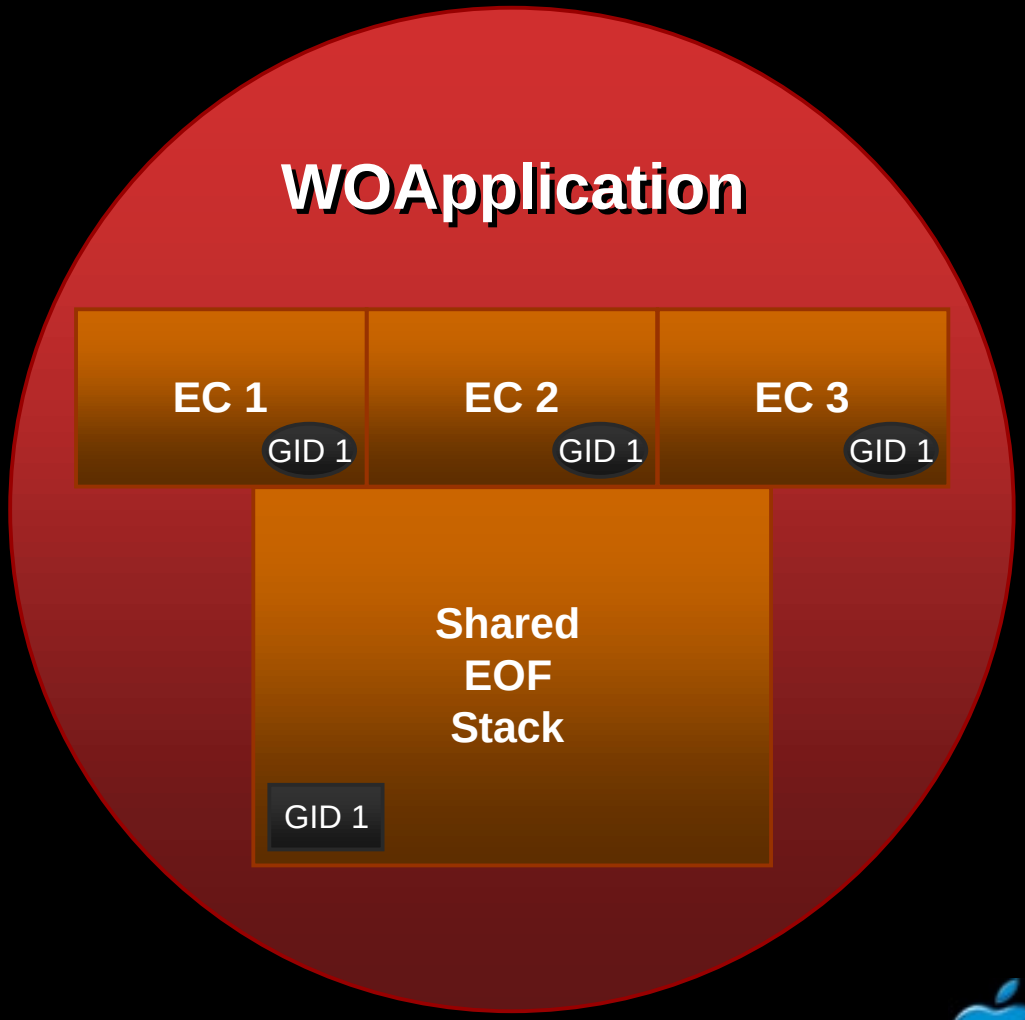
Query Database
Update Snapshot
Broadcast Changes
**Object in EC3**




# New in 4.5: TimeStamps

EC 3 Fetch:

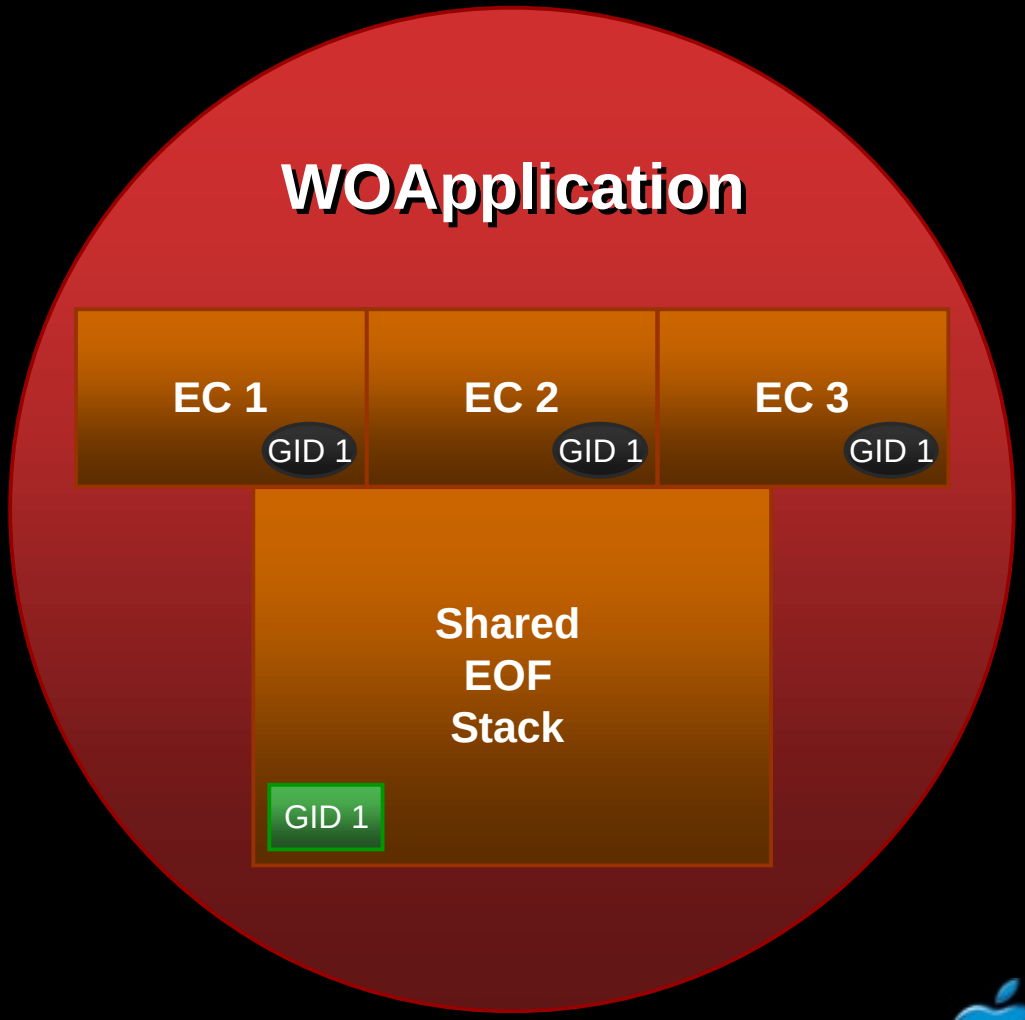
**Query Database**
**Check TimeStamp**
Update Snapshot
Broadcast Changes
Object in EC3




# New in 4.5: TimeStamps

EC 3 Fetch:

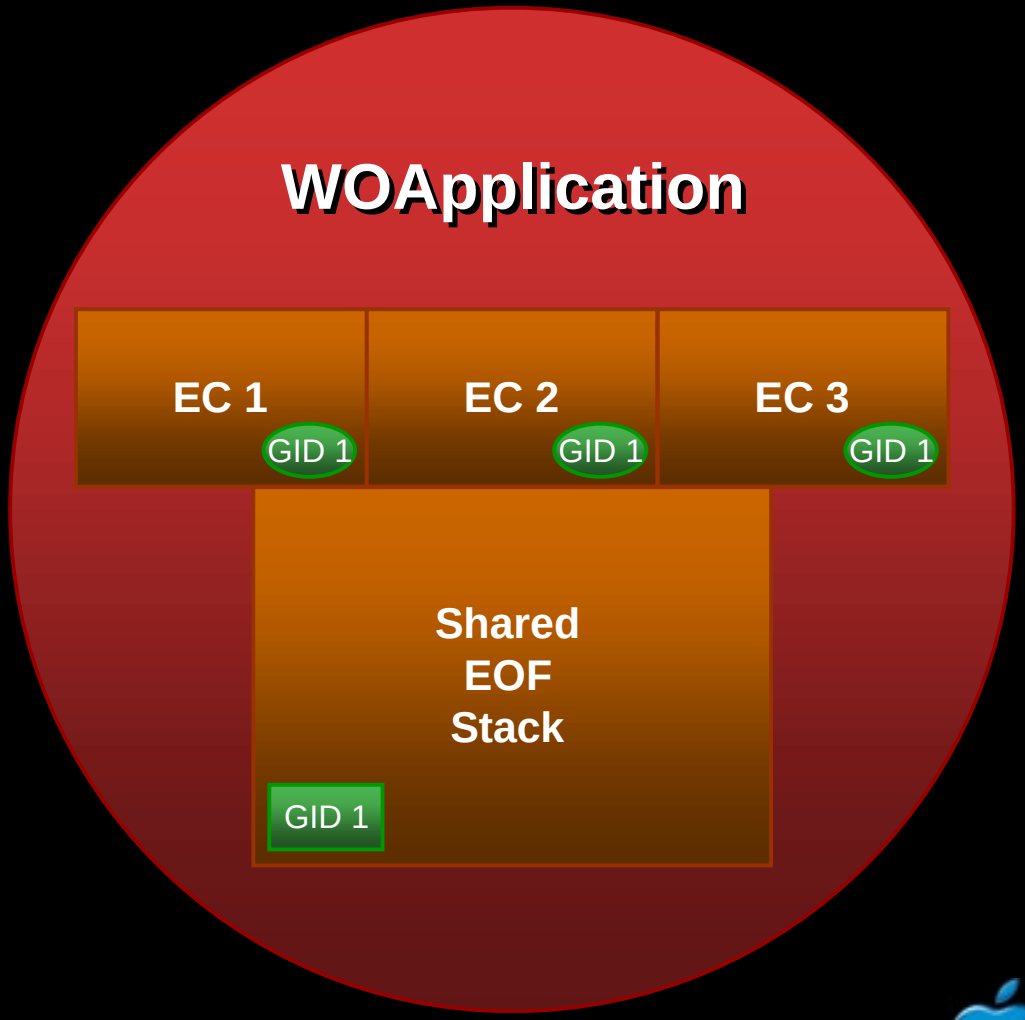
Query Database
**Update Snapshot**
Broadcast Changes
Object in EC3



# New in 4.5: TimeStamps

EC 3 Fetch:

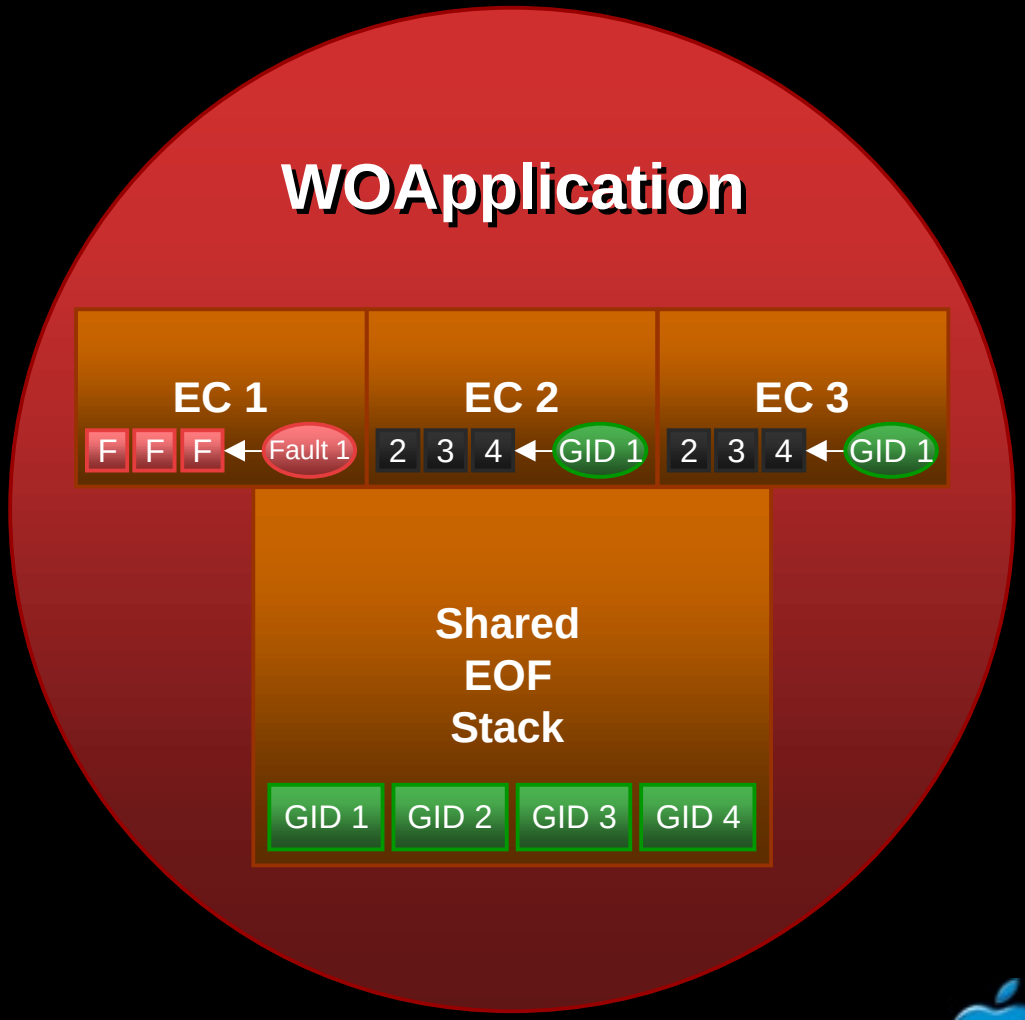
Query Database
Update Snapshot
**Broadcast Changes**



# Invalidating Objects Individually

EC 1 Invalidates Object:

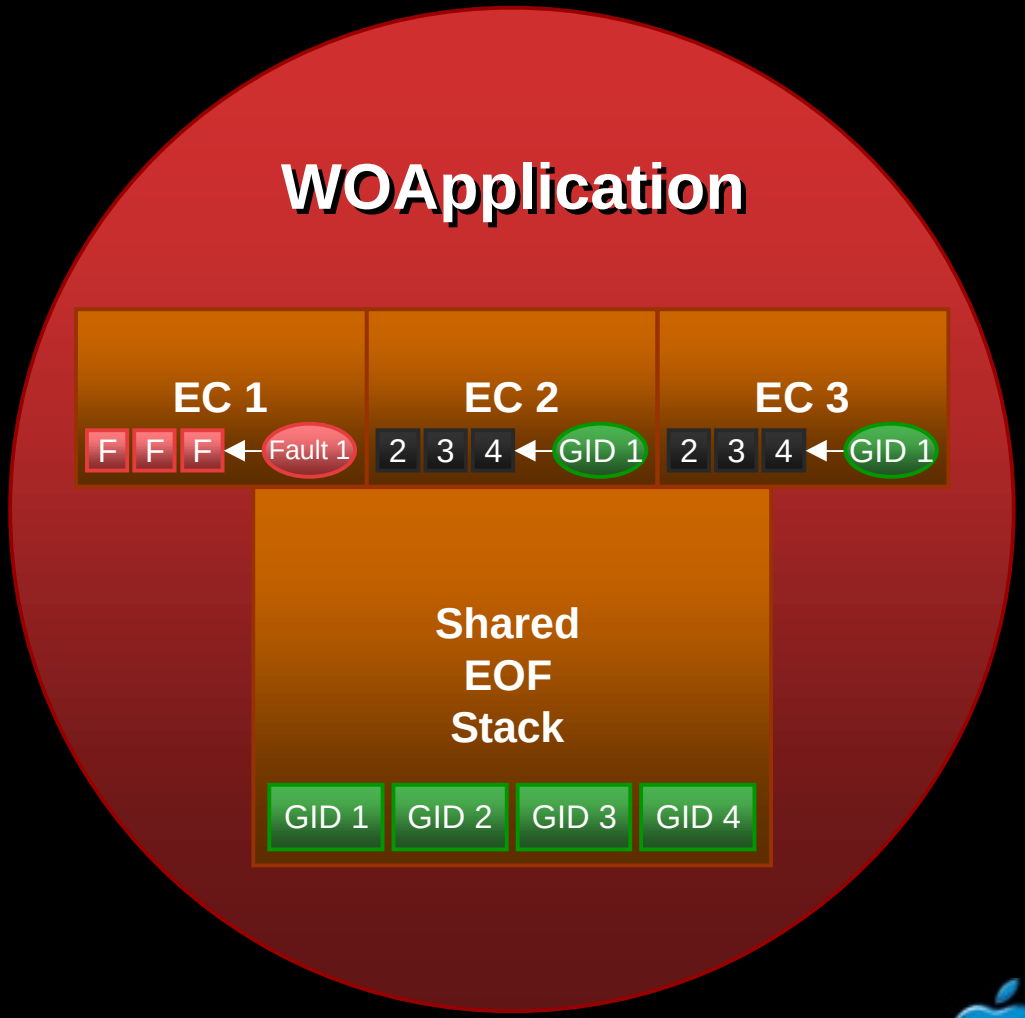
**Refault Object**
**Refault To-Many Relationships**
**Preserve To-One**



# Invalidating Objects Individually

EC 1 Trips Fault/Relationship:

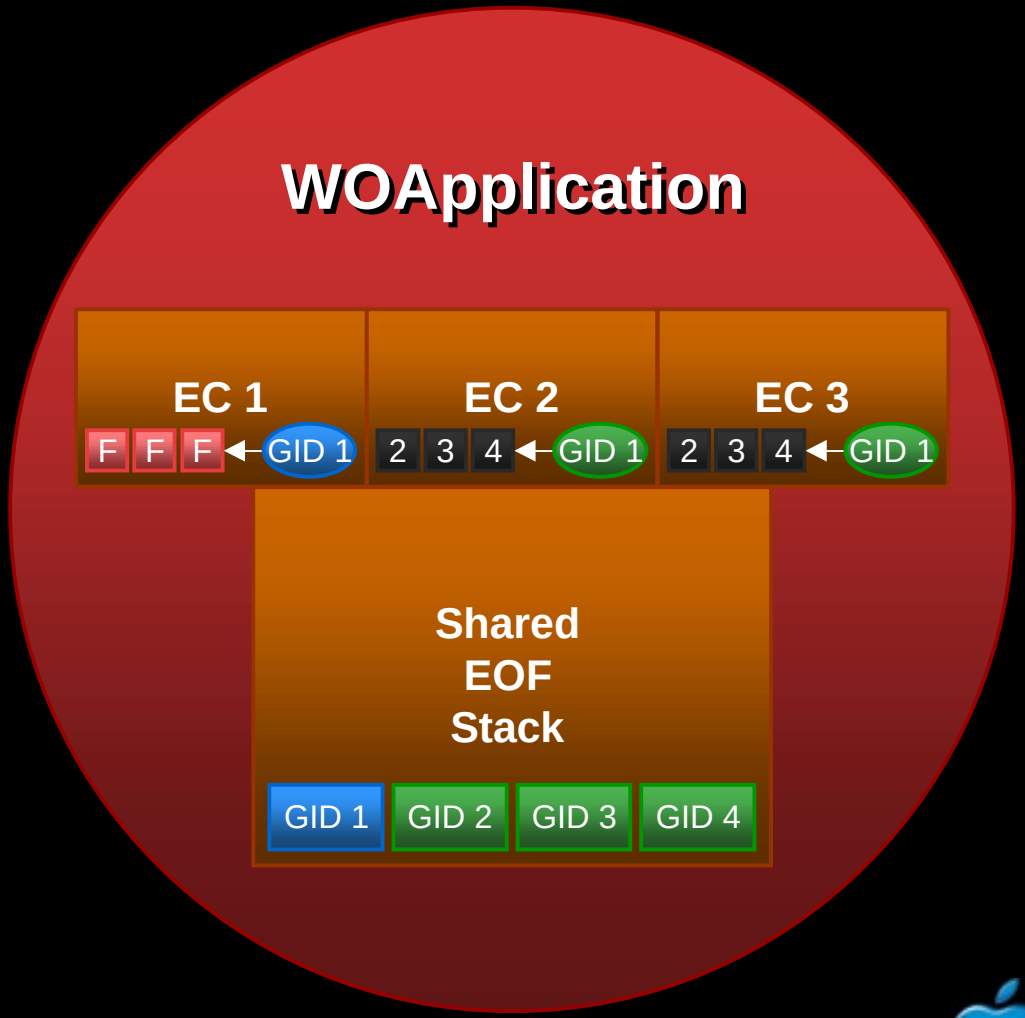
**Query Database:** Object 1
Update Snapshot
Updates in EC 1
Broadcast Changes
Query DB: Relationship
Discard Changes
Relationship in EC 1




# Invalidating Objects Individually

EC 1 Trips Fault/Relationship:

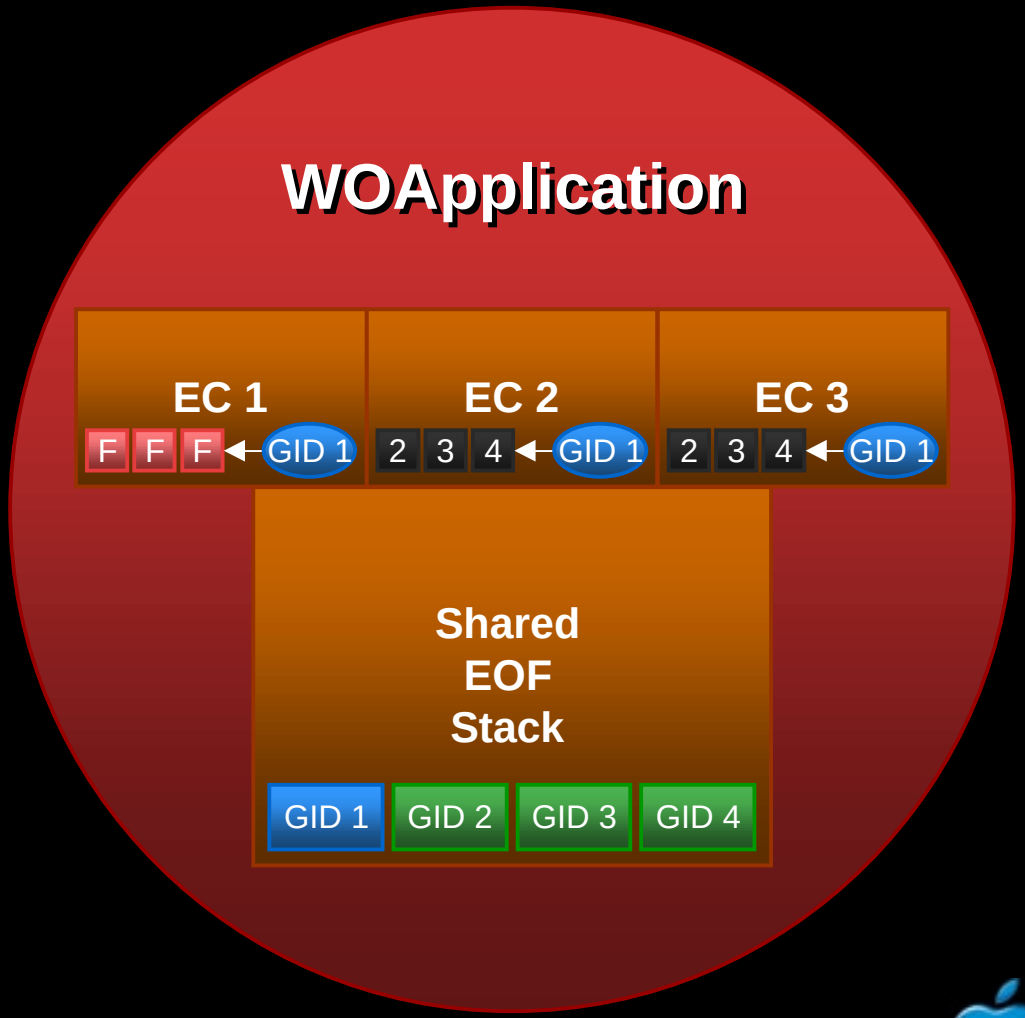
Query Database: Object 1
**Update Snapshot**
**Updates in EC 1**
Broadcast Changes
Query DB: Relationship
Discard Changes
Relationship in EC 1



# Invalidating Objects Individually

EC 1 Trips Fault/Relationship:

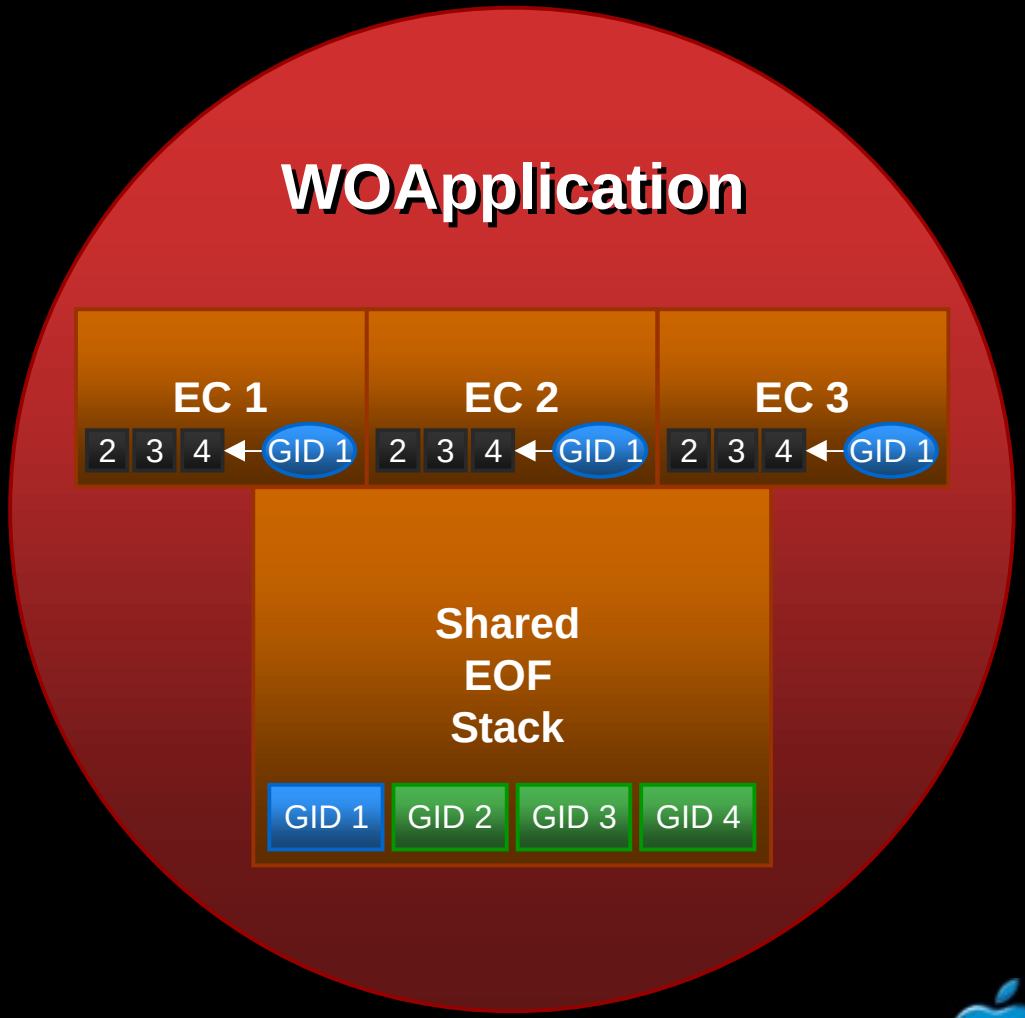
Query Database: Object 1
Update Snapshot
Updates in EC 1
**Broadcast Changes**
Query DB: Relationship
Discard Changes
Relationship in EC 1



# Invalidating Objects Individually

EC 1 Trips Fault/Relationship:

Query Database: Object 1
Update Snapshot
Updates in EC 1
Broadcast Changes
**Query DB: Relationship**
**Discard Changes**
**Relationship in EC** 1

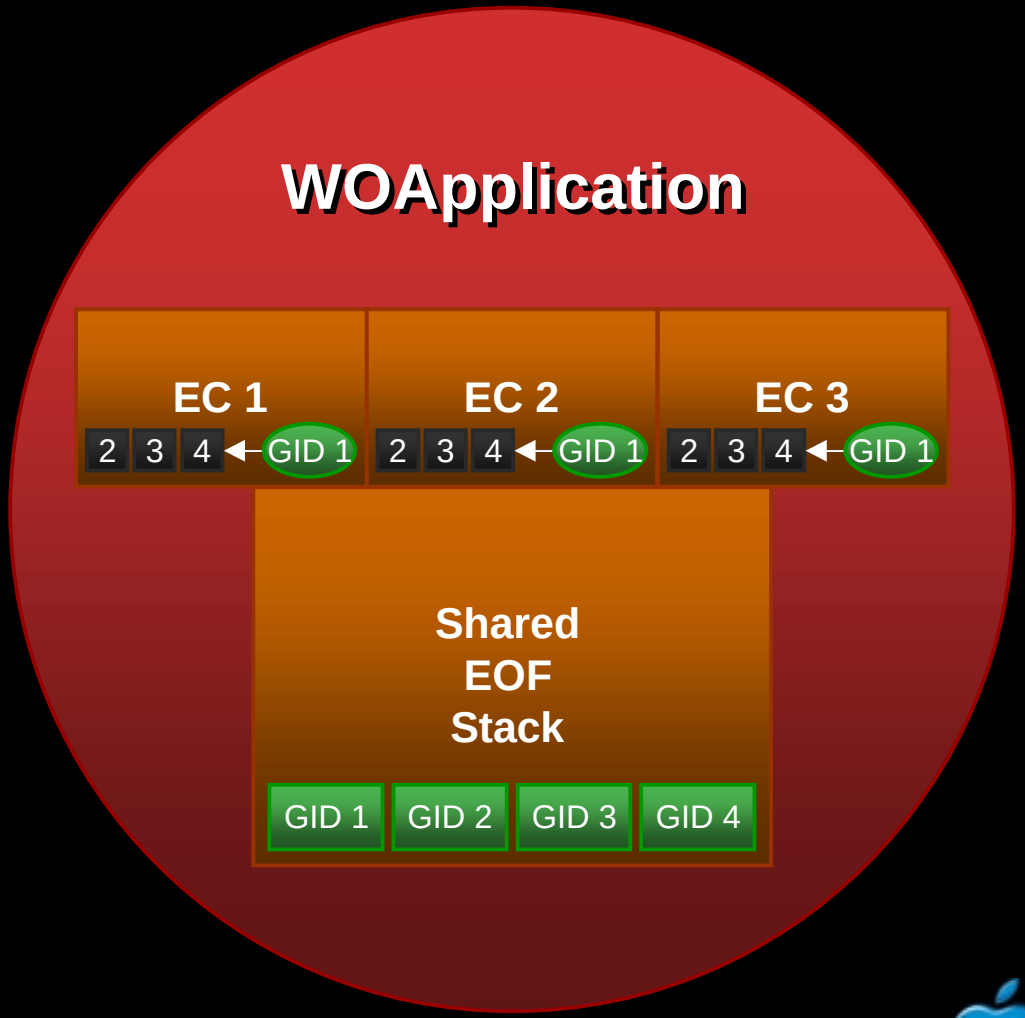


# Invalidating All Objects

invalidateAllObjects

Every Object is Refaulted
Every Relationship Refaulted
For every tripped fault:
new query
update snapshot
broadcast changes
including to-many

MORE EXPENSEIVE THEN ORIGINAL QUERIES



# Coordinating Changes

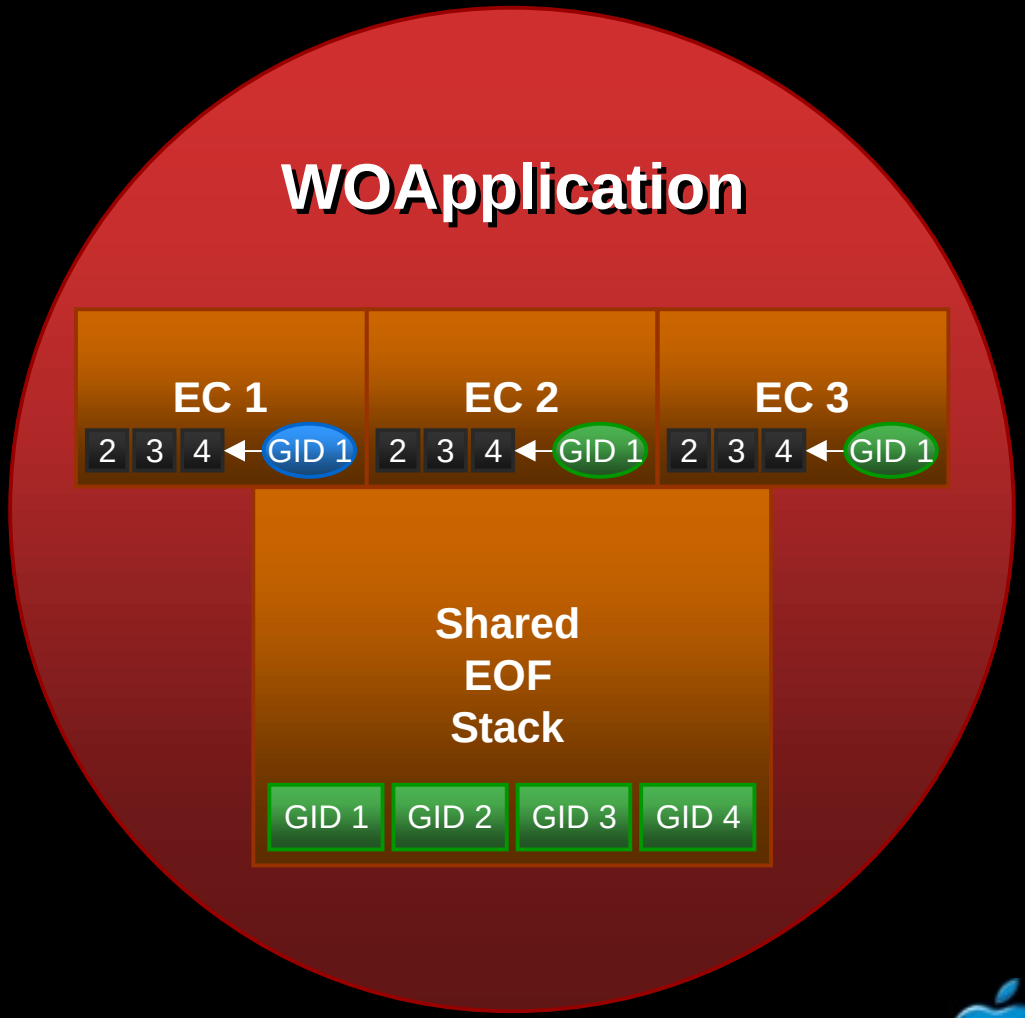
- Understanding locking behavior
- Sometimes users see locking failures, sometimes they overwrite changes, why?
- My relationships are changing

# Committing Changes:

## Single Instance

EC 1 Modifies and Commits Change

**Object 1 modified**
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

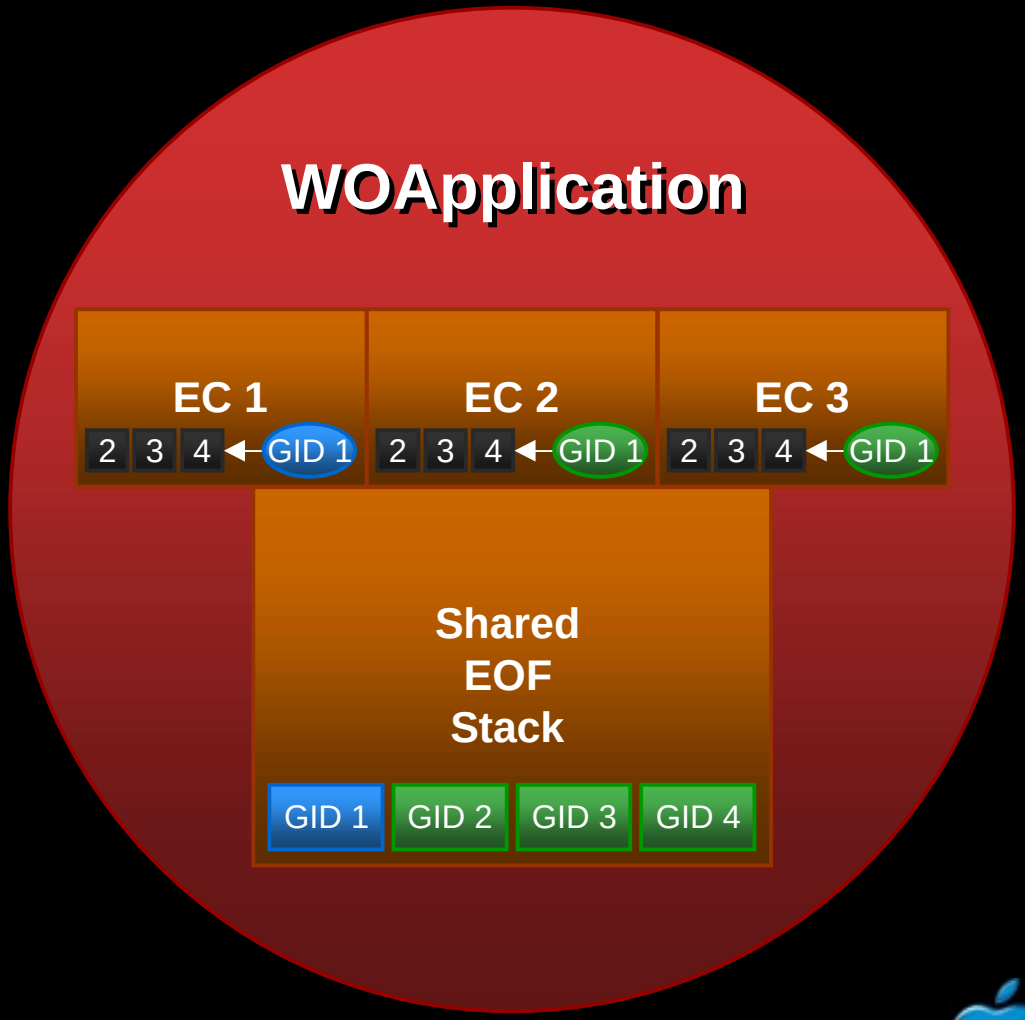



# Committing Changes:

## Single Instance

EC 1 Modifies and Commits Change

Object 1 modified
**Database updated:**
lock against snapshot
**Snapshot updated**
Broadcast Changes

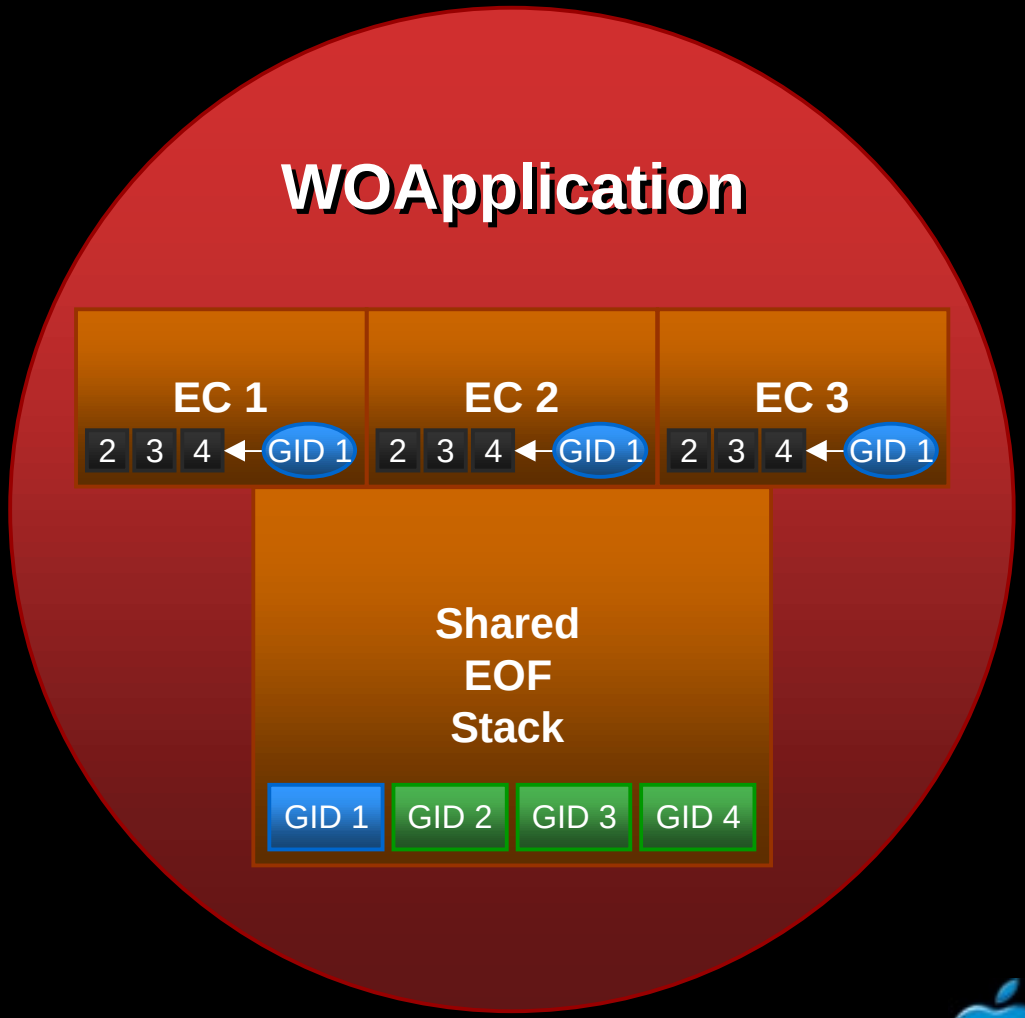


# Committing Changes:

## Single Instance

EC 1 Modifies and Commits Change

Object 1 modified
Database updated:
lock against snapshot
Snapshot updated
**Broadcast Changes**




# Committing Changes:

## Single Instance

EC 1 and EC 2 Modify and Commit Change

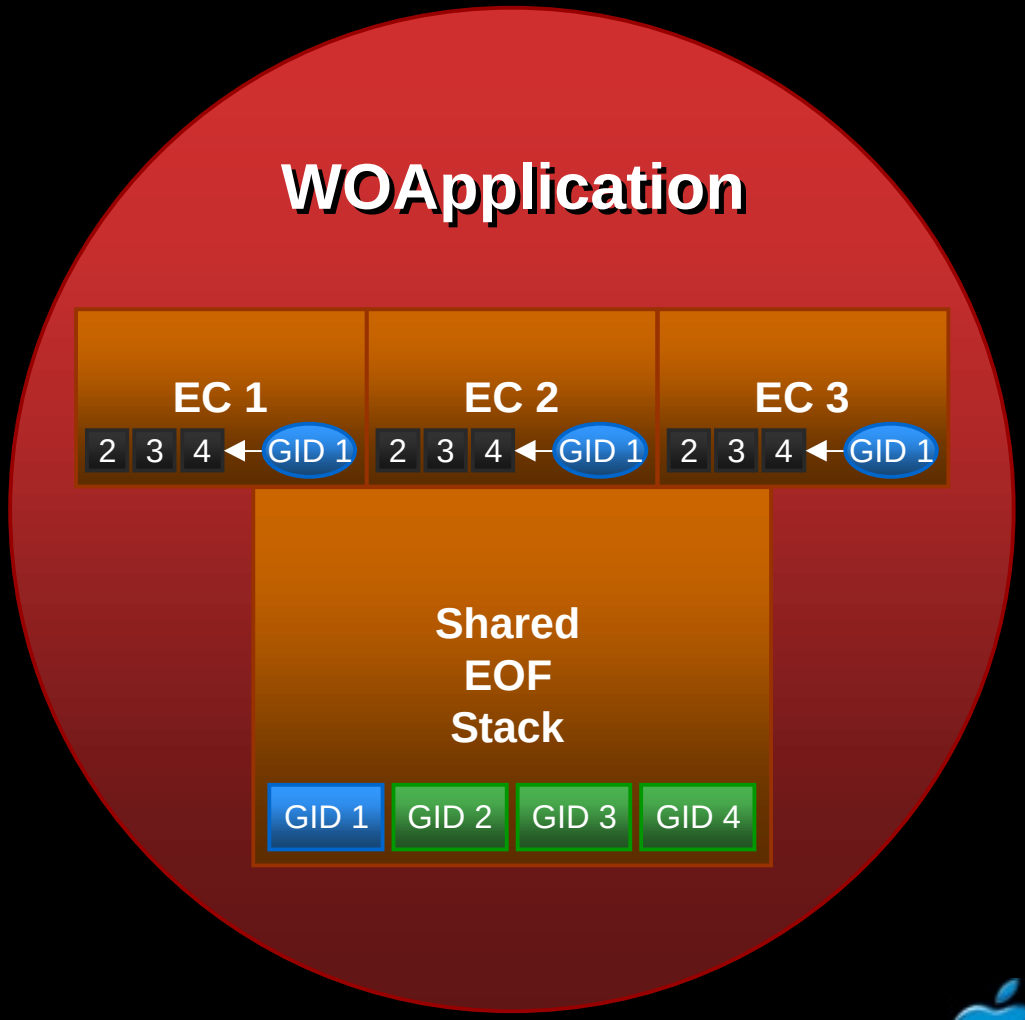
EC1 and EC2 modify

EC 1 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

EC 2 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



# Committing Changes:

## Single Instance

EC 1 and EC 2 Modify and Commit Change

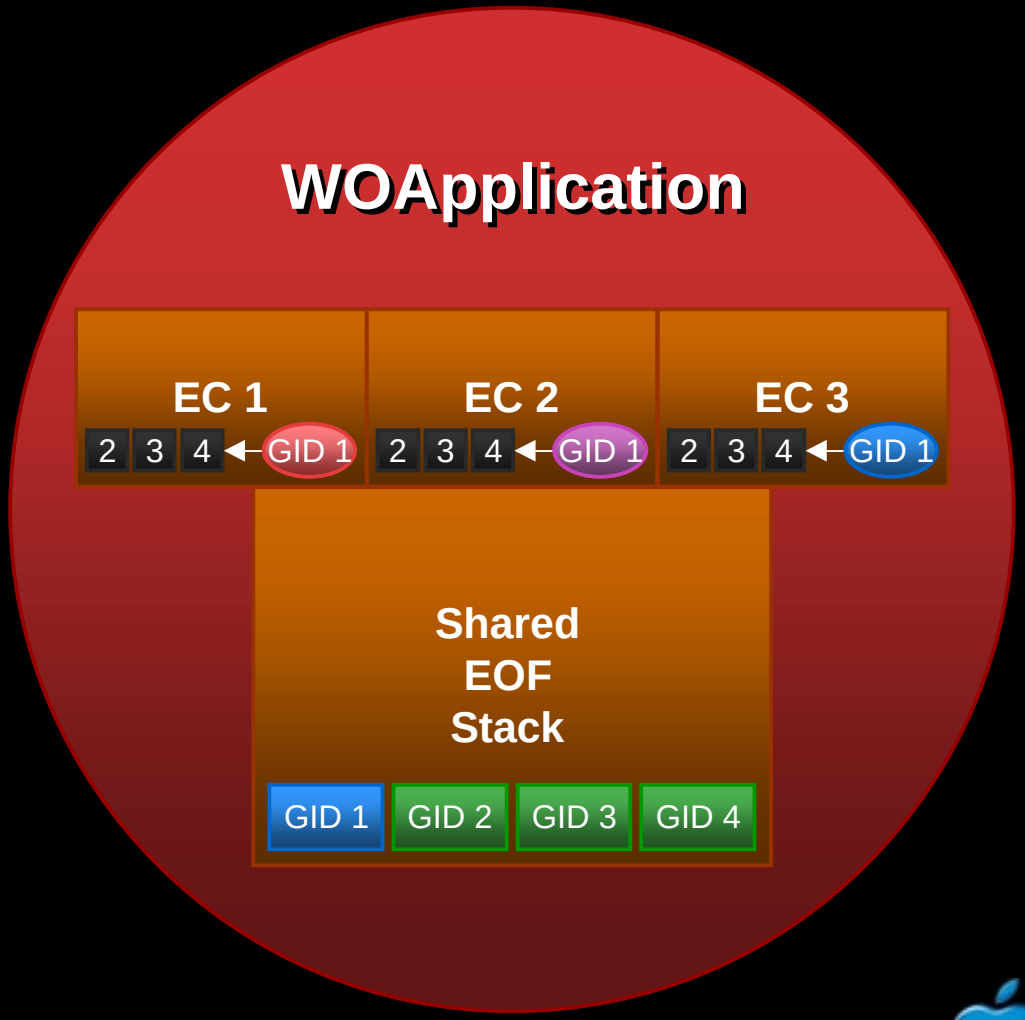
**EC1 and EC2 modify**

EC 1 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

EC 2 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



# Committing Changes:

## Single Instance

EC 1 and EC 2 Modify and Commit Change

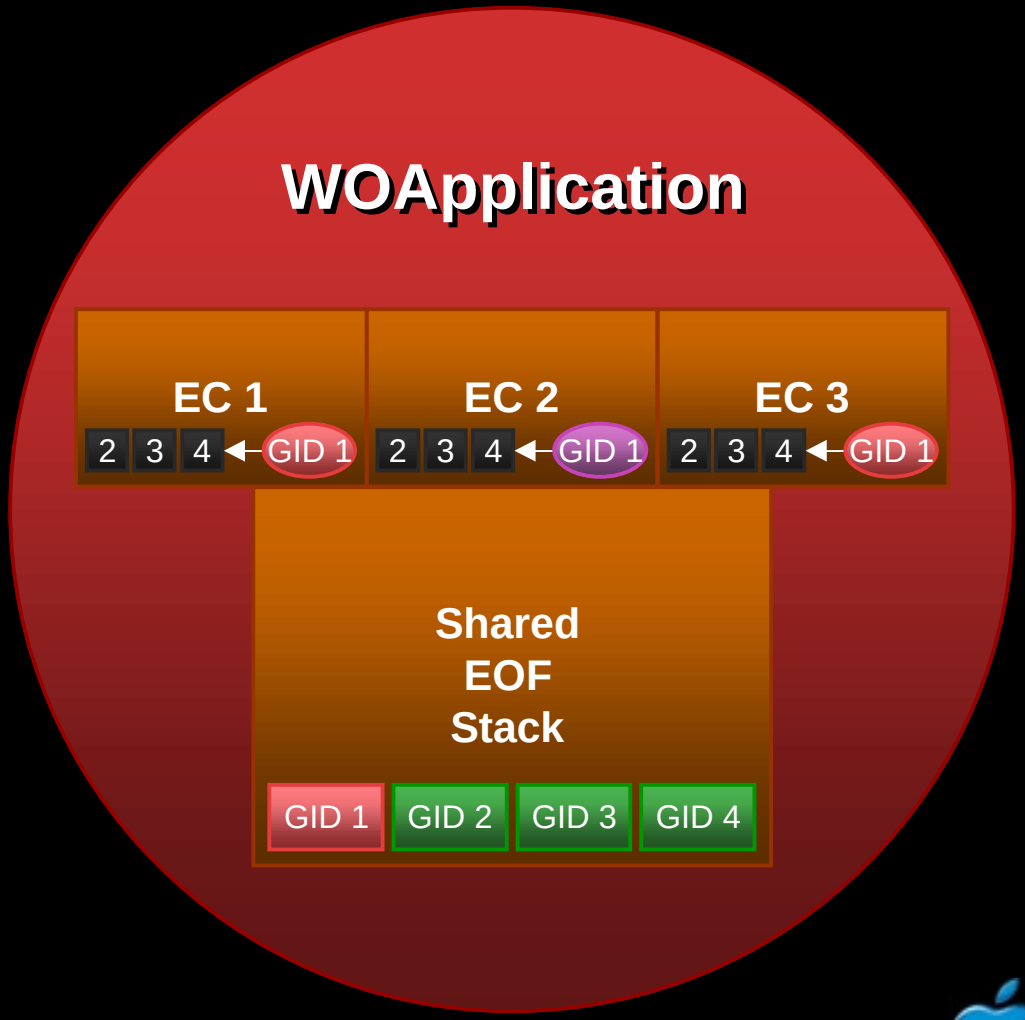
EC1 and EC2 modify

**EC 1 commits:**

**Database updated:**
**lock against snapshot**
**Snapshot updated**
**Broadcast Changes**

EC 2 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes




# Committing Changes:

## Single Instance

EC 1 and EC 2 Modify and Commit Change

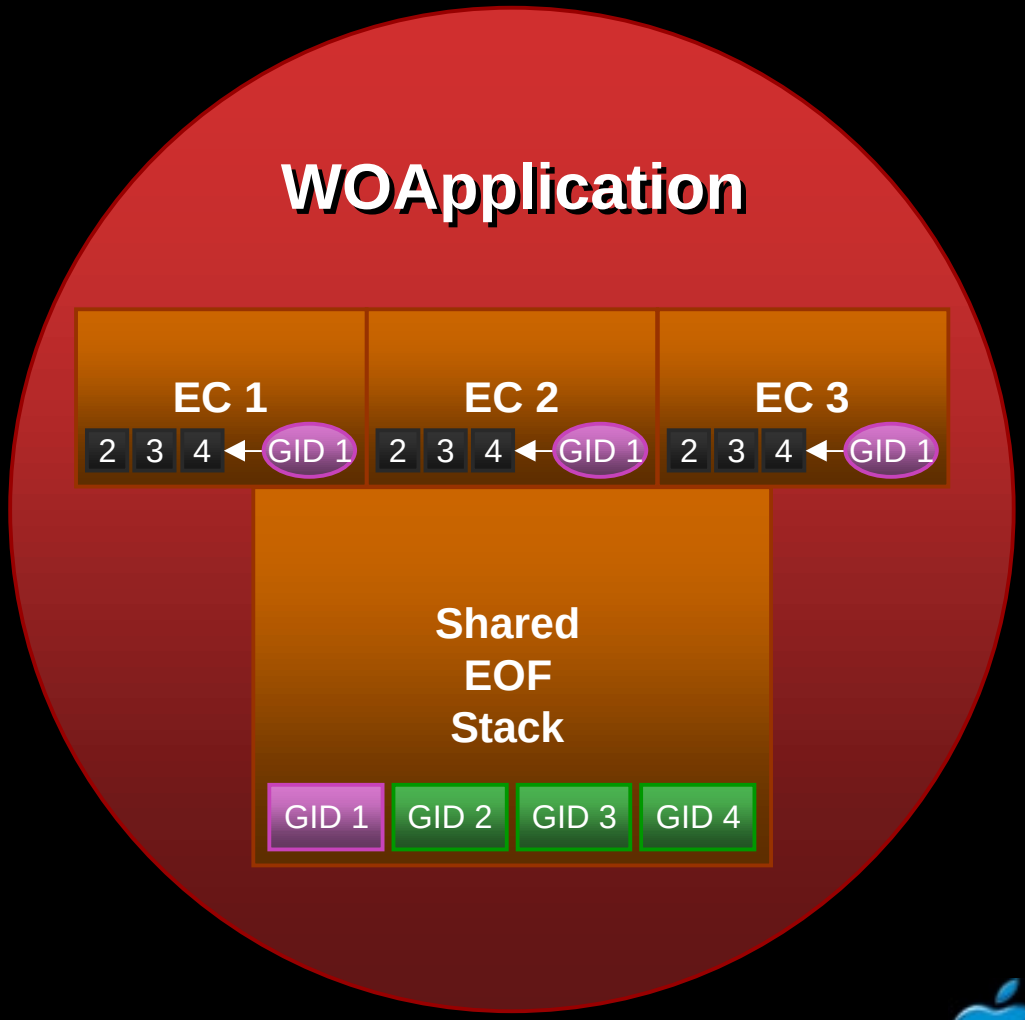
EC1 and EC2 modify

EC 1 commits:

Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

**EC 2 commits:**

**Database updated:**
**lock against snapshot**
**Snapshot updated**
**Broadcast Changes**



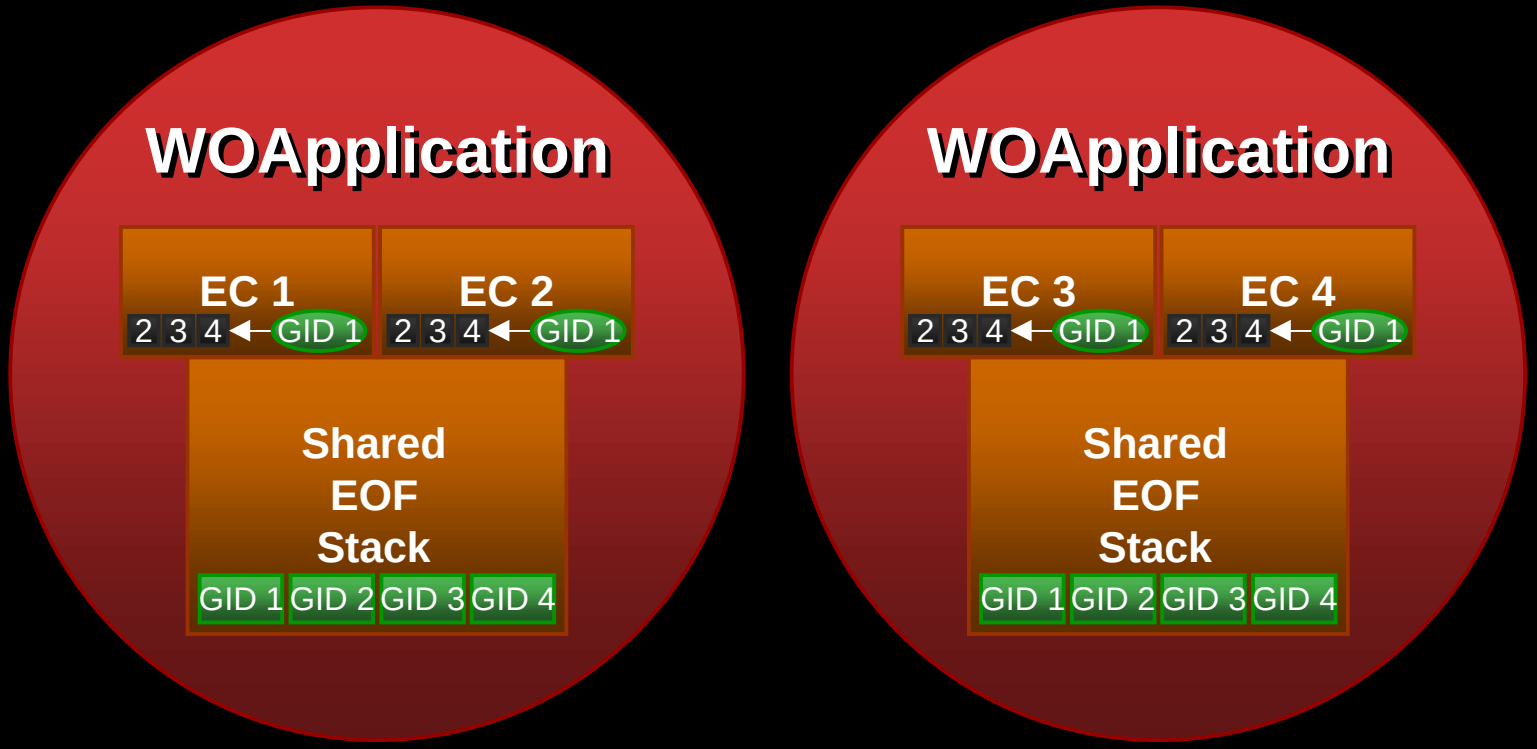
# Committing Changes:

## Multiple Instance

EC 1 and EC 3
Modify and
Commit Change

EC 1 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

EC 3 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



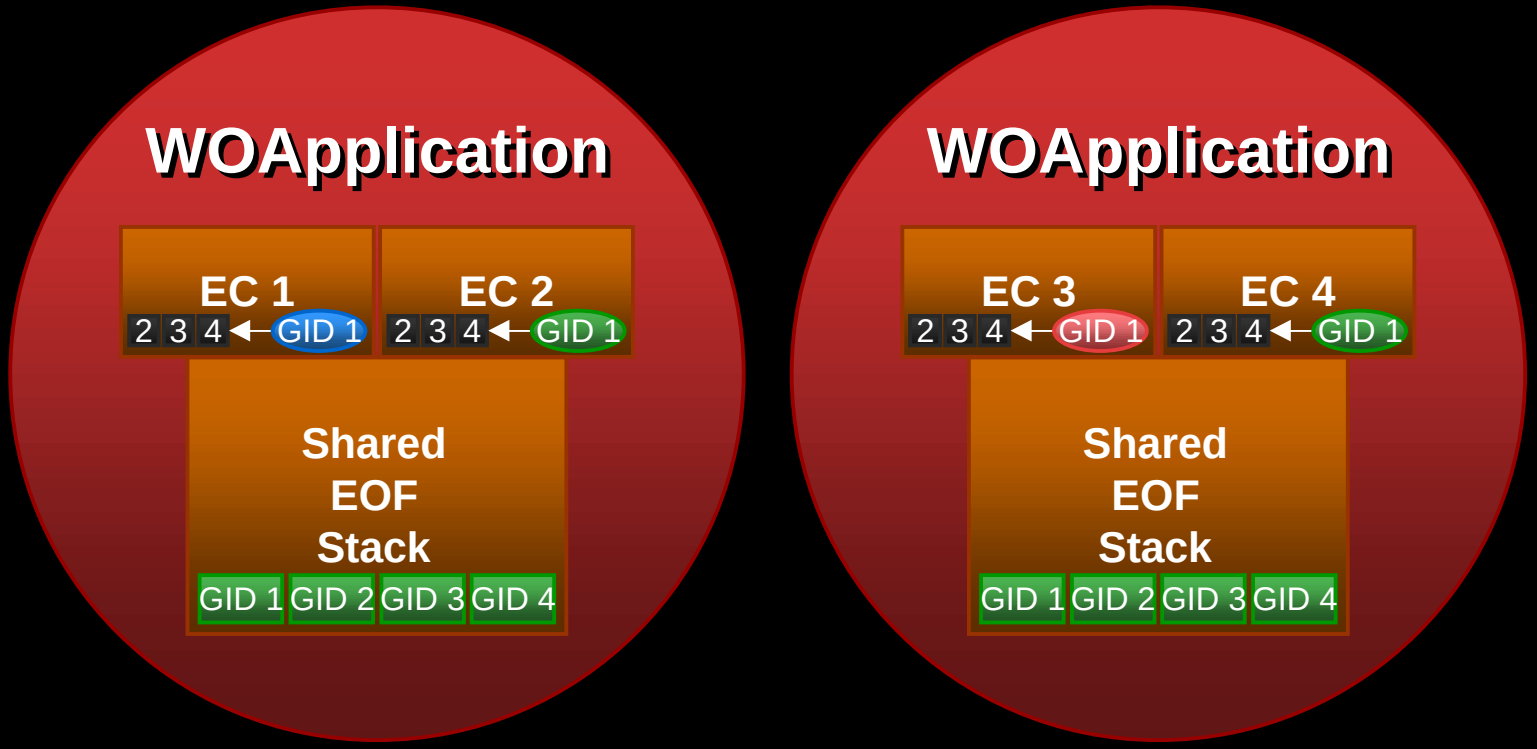
# Committing Changes:

## Multiple Instance

**EC 1 and EC 2**
**Modify** and
Commit Change

EC 1 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

EC 3 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



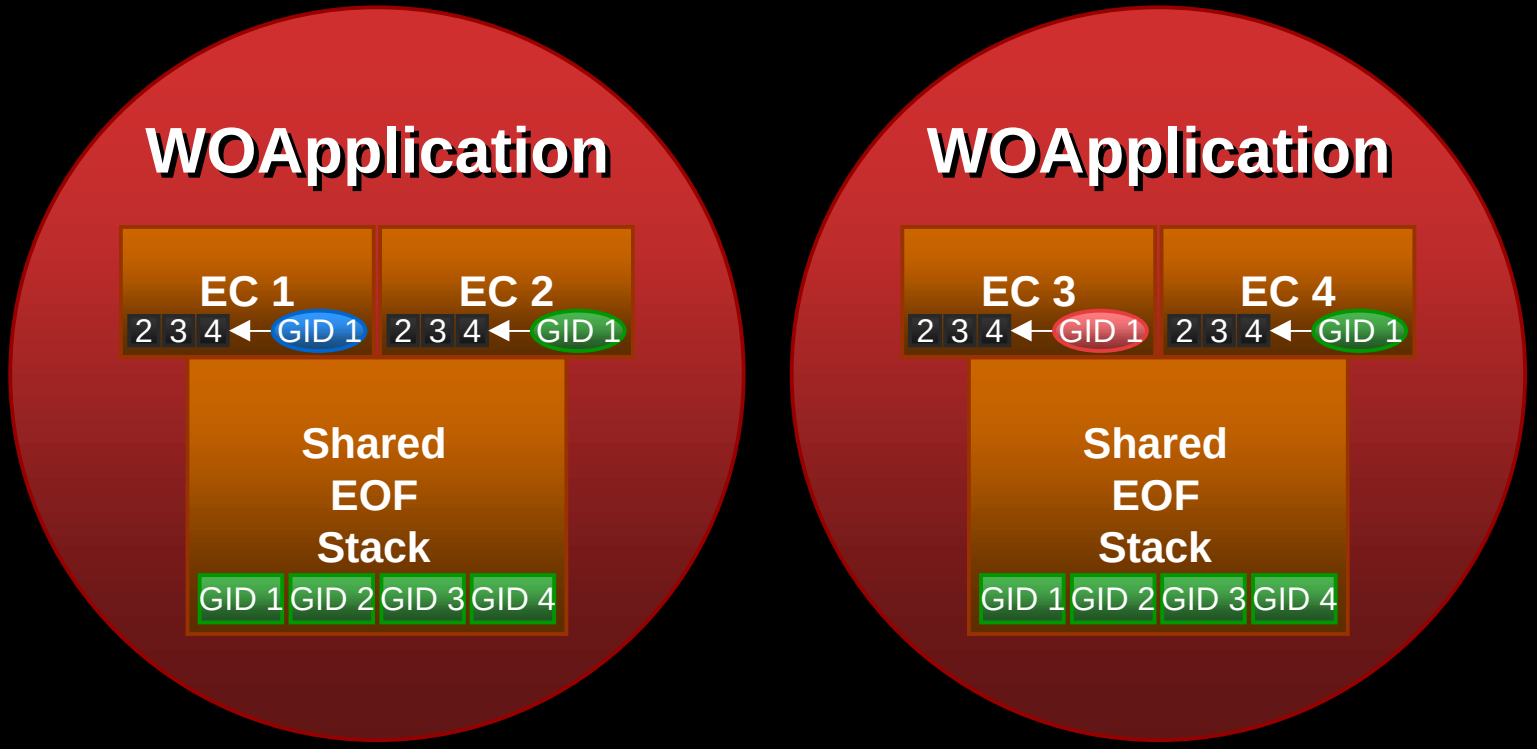
# Committing Changes:

## Multiple Instance

EC 1 and EC 2
Modify and
Commit Change

**EC 1 commits:**
**Database updated:**
lock against snapshot
Snapshot updated
Broadcast Changes

EC 3 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



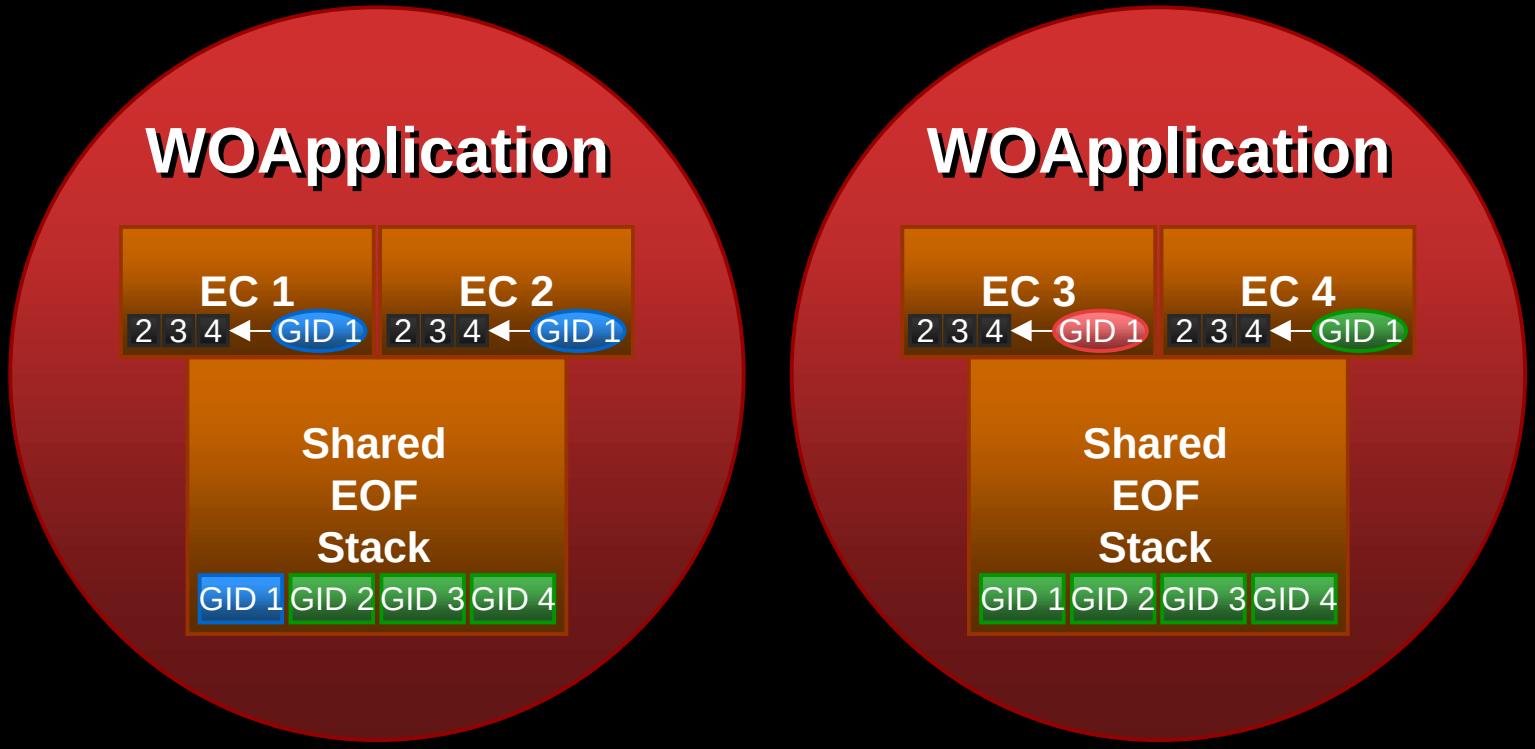
# Committing Changes:

## Multiple Instance

EC 1 and EC 2
Modify and
Commit Change

**EC 1 commits:**
Database updated:
lock against snapshot
**Snapshot updated**
**Broadcast Changes**

EC 3 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes



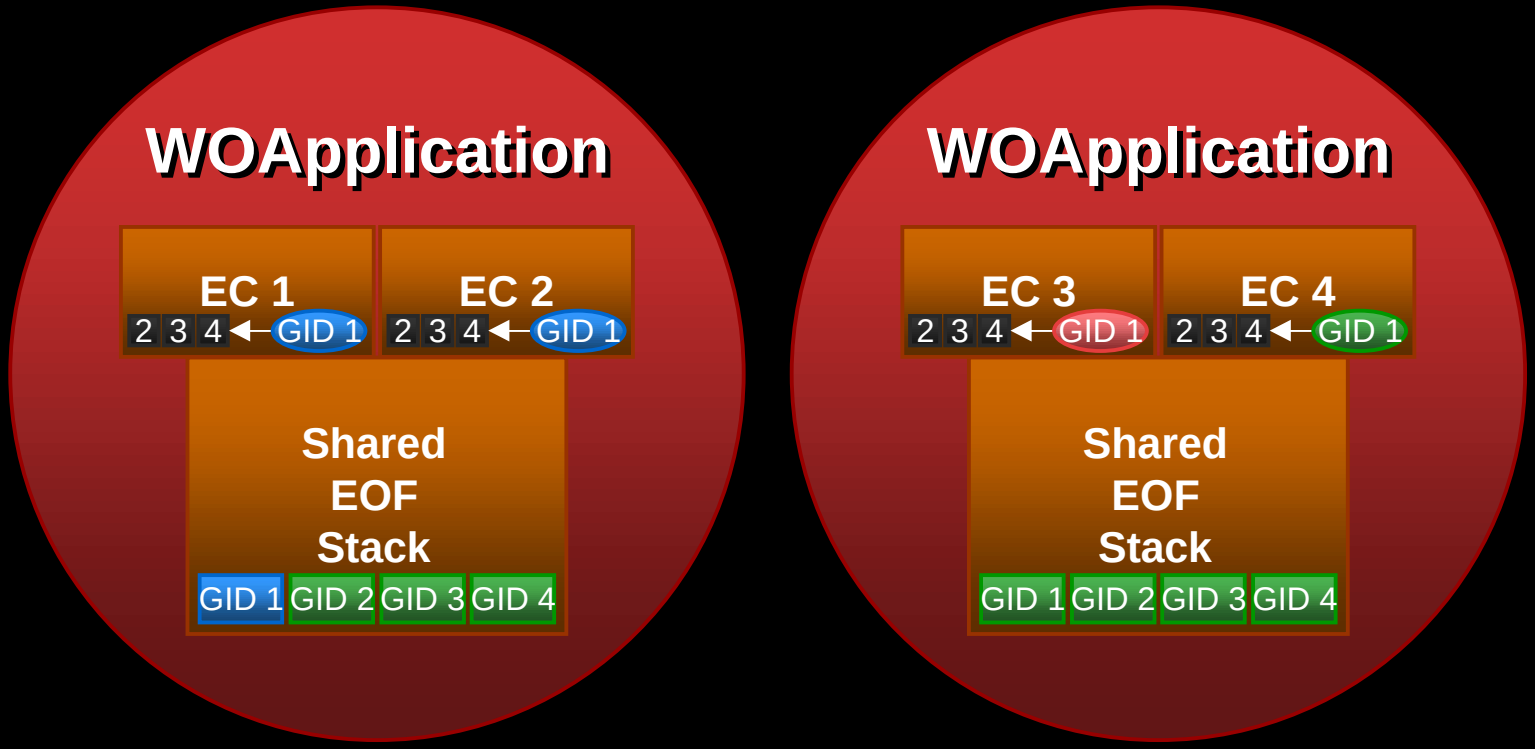
# Committing Changes:

## Multiple Instance

EC 1 and EC 2
Modify and
Commit Change

EC 1 commits:
Database updated:
lock against snapshot
Snapshot updated
Broadcast Changes

**EC 3 commits:**
**Database updated:**
**lock against snapshot**
**Locking Failure!!**



# For More Information

**http://www.apple.com/webobjects**

Visit the WebObjects lab downstairs!
Everyday from 11:00 a.m.–2:00 p.m.

Try out your WebObjects 4.5 Evaluation CD!

Session 412

# Q&A

# Who to Contact

**Toni Trujillo Vian**
Director, WebObjects Engineering
**wofeedback@group.apple.com**

**Ernest Prabhakar**
Product Line Manager, WebObjects
**webobjects@group.apple.com**

# Useful Delegates

- public abstract boolean **editingContextShouldMergeChangesForObject** (EOEditingContext *anEditingContext*, EOEnterpriseObject *object*)

- public abstract void **editingContextDidMergeChanges** (EOEditingContext *anEditingContext*)

# Refreshing to Many

- Construct a method that accepts relationship name, an instance of an EO
- Construct fetch spec, with key value qualifiers for each EO in the relationship
- setRefresh to true, and fetch objects
- Beware of usual side effects of refreshing snapshot

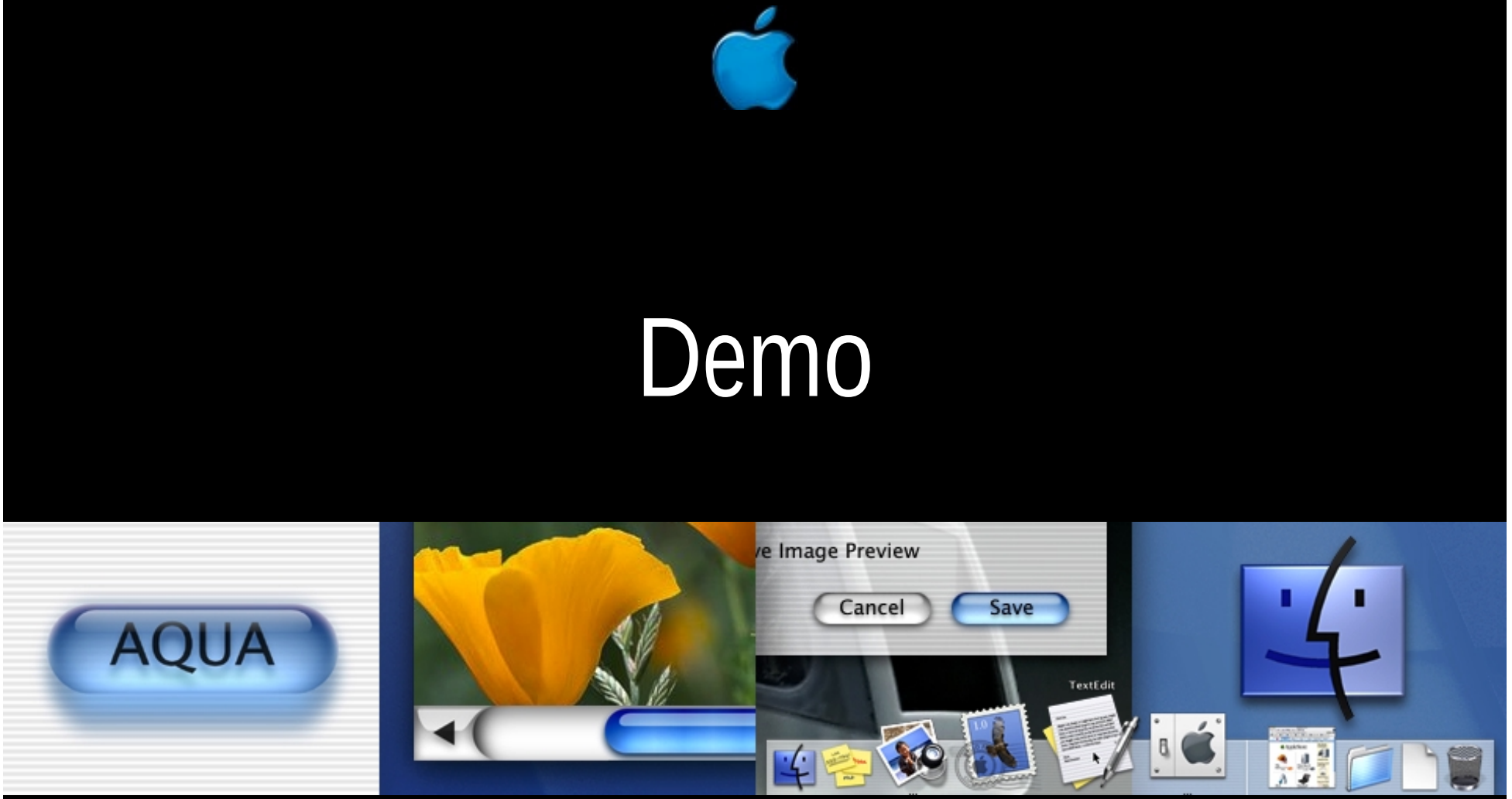
# Proposed Changes

- Eliminate broadcasting notification on changes
- Maintain revision numbers on snapshots and EOs
- Always refresh snapshot and object in fetching EC on fetches, updating revision number
- Use revision numbering to check for optimistic lock failures
- Give user easy toggle between no locking (last one wins), and optimistic locking exceptions

WWDC

Worldwide Developers Conference 2000

Think different.




**Daniel Abrams**
**Consulting Engineer**