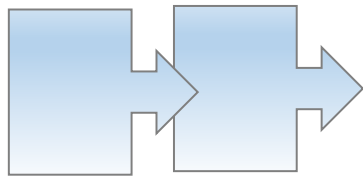


# Consistency Options

(Demo and Discussion)

Open-Source Meeting 06/20



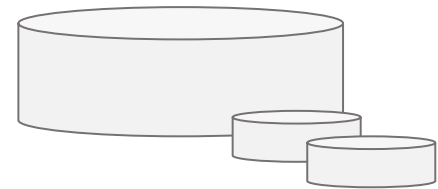
# Topics

“Session” Consistency Demo

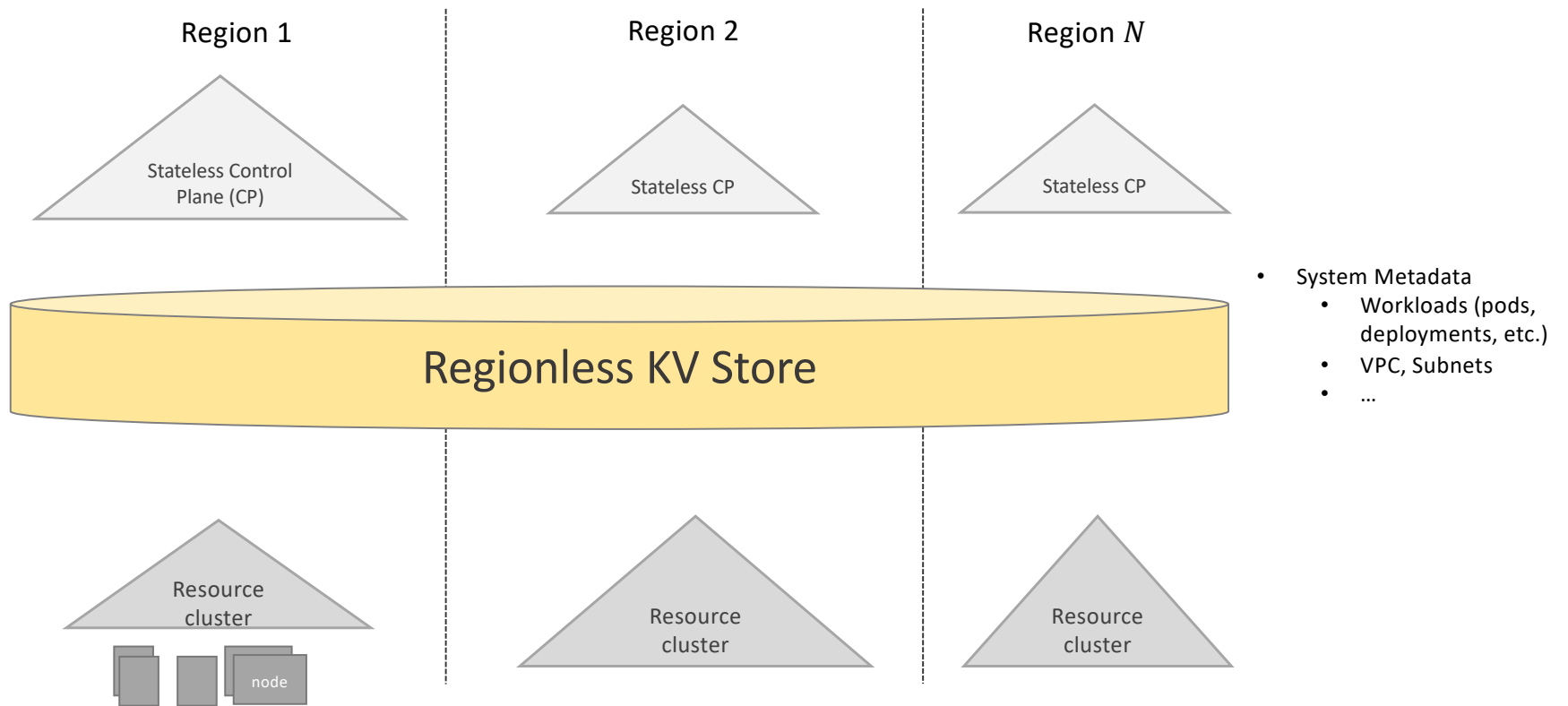


Regionless Store R&D Update

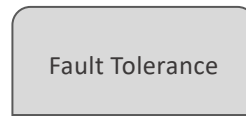
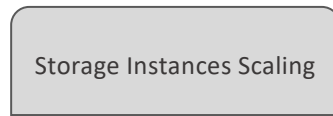
# 1. Project Update



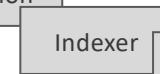
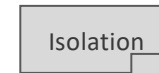
# Regionless Storage User Scenario



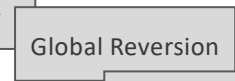
# R&D



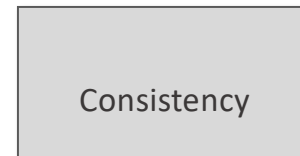
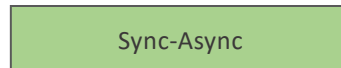
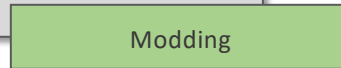
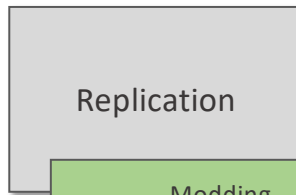
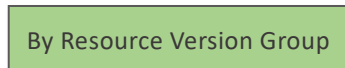
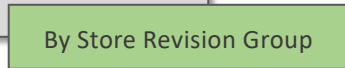
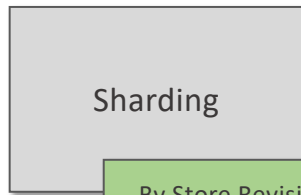
Owner: Hongwei Chen



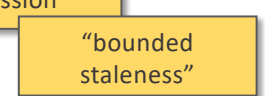
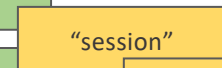
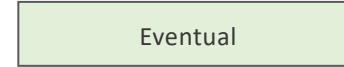
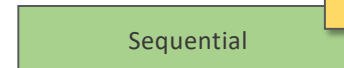
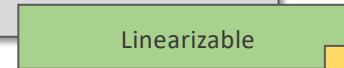
Owner: Jun Shao



Owner: Hongwei Chen

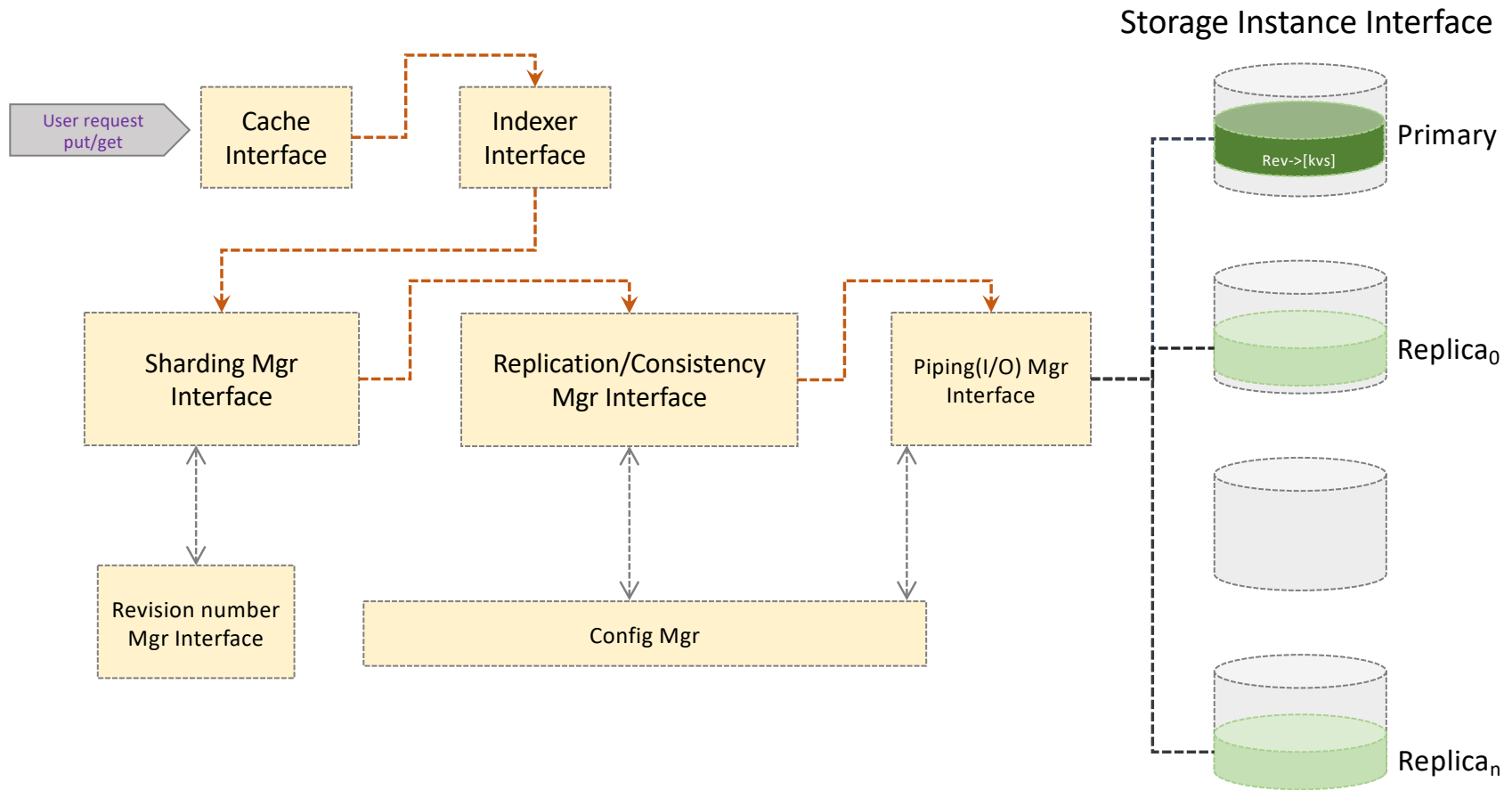


Owner: Ke Xu



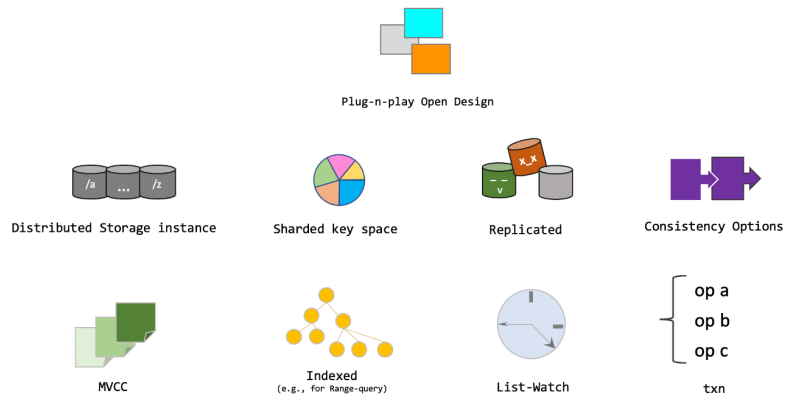
Component Interface

# Architecture (at interface level)



# Key Features POC

## Highlighted Features



- Distributed storage nodes.
- Sharded key space.
- Replicated for HA and faster read.
- MVCC versioning.
- Indexed DB for fast query.
- Supporting put, get, range query and watch.
- Supporting txn KV access (e.g. write one revision with a set of key-value pairs).

Implementation for each component based on an interface design.

### Languages



\*This repo is private now but will be made public shortly

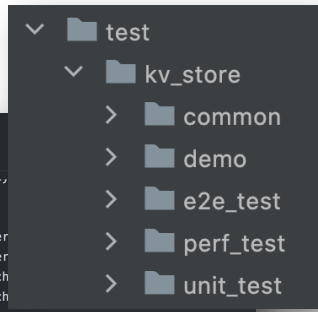
### Regionless\_KV\_POC\_Bot

succeeded 2 hours ago in 1m 47s

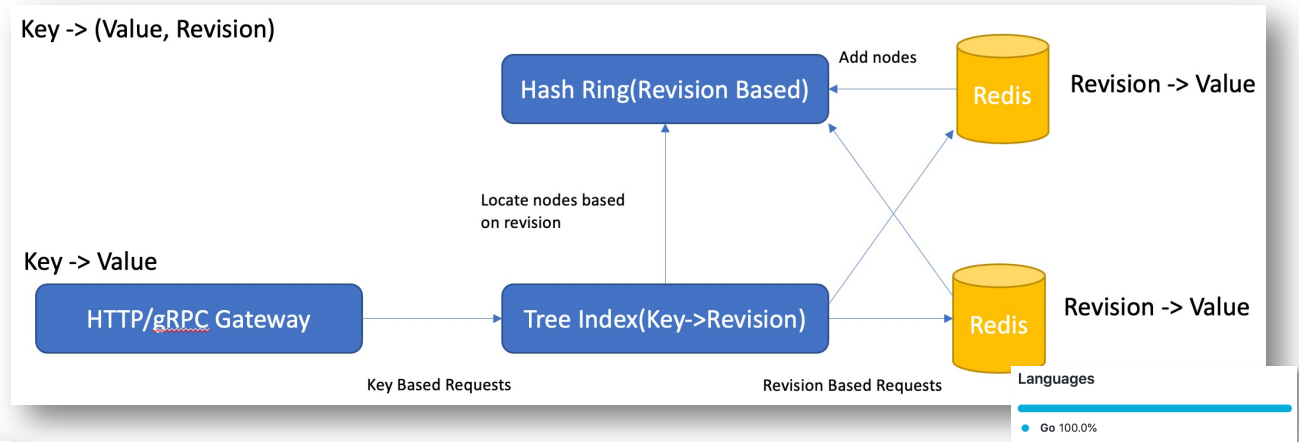
```

37 2: [ RUN ] e2e.range_query
38 2: [ OK ] e2e.range_query (2 ms)
39 2: [ RUN ] e2e.range_watch_single_watcher
40 2: [ OK ] e2e.range_watch_single_watcher
41 2: [ RUN ] e2e.range_watch_multiple_watcher
42 2: [ OK ] e2e.range_watch_multiple_watcher
43 2: [ RUN ] e2e.range_watch_prefix
44 2: [ OK ] e2e.range_watch_prefix (2 ms)
45 2: [ RUN ] e2e.async_read
46 2: [LOG]: in e2e.async_read: async read not ready, will retry after 1 sec
47 2: [ OK ] e2e.async_read (1000 ms)
48 2: [-----] 12 tests from e2e (1018 ms total)
49 2:
50 2: [-----] 1 test from play_ground
51 2: [ RUN ] play_ground.put_get
52 2: [LOG]: get returns rev[1]: [hello : world]
53 2: [ OK ] play_ground.put_get (0 ms)
54 2: [-----] 1 test from play_ground (0 ms total)
55 2:
56 2: [-----] 4 tests from perf
57 2: [ RUN ] perf.read_same_key_bucket_sharding
58 2: [ OK ] perf.read_same_key_bucket_sharding (9 ms)
59 2: [ RUN ] perf.read_same_key_mod_sharding
60 2: [ OK ] perf.read_same_key_mod_sharding (5 ms)
61 2: [ RUN ] perf.read_same_key_ver_bucket_vs_mod_sharding
62 2: [ OK ] perf.read_same_key_ver_bucket_vs_mod_sharding (59 ms)
63 2: [ RUN ] perf.read_same_key_grouping_ver_bucket_vs_mod_sharding
64 2: [ OK ] perf.read_same_key_grouping_ver_bucket_vs_mod_sharding (69 ms)
65 2: [-----] 4 tests from perf (142 ms total)
66 2:
67 2: [-----] 12 tests from unit_test
68 2: [ RUN ] unit_test.btree_indexer_find_version_different_key
69 2: [ OK ] unit_test.btree_indexer_find_version_different_key (0 ms)
70 2: [ RUN ] unit_test.btree_indexer_find_version_same_key
71 2: [ OK ] unit_test.btree_indexer_find_version_same_key (0 ms)
72 2: [ RUN ] unit_test.config_storage_grouping
73 2: [ OK ] unit_test.config_storage_grouping (0 ms)
74 2: [ RUN ] unit_test.cache_able_to_put_get

```



# Storage and Performance Validation



## I/O Perf Test

### go-ycsb

go-ycsb is a Go port of [YCSB](#). It fully supports all YCSB CRUD benchmarks with Go.

### Why another Go YCSB?

- We want to build a standard benchmark tool in Go.
- We are not familiar with Java.

Owner: *Hongwei Chen*

```
• Create
ubuntu@ip-172-31-9-140:~$ curl -X POST -k http://localhost:8090/kv -d '{"key": "hello", "value": "world"}'
The key value pair (hello,world) has been saved as revision 2 at 127.0.0.1:6379
ubuntu@ip-172-31-9-140:~$ redis-cli -h 127.0.0.1 -p 6379
127.0.0.1:6379> get 2
"world"

• Update
ubuntu@ip-172-31-9-140:~$ curl -X POST -k http://localhost:8090/kv -d '{"key": "hello", "value": "believe"}'
The key value pair (hello,believe) has been saved as revision 4 at 127.0.0.1:6379
ubuntu@ip-172-31-9-140:~$ redis-cli -h 127.0.0.1 -p 6379
127.0.0.1:6379> get 4
"believe"

• View
ubuntu@ip-172-31-9-140:~$ curl -X GET -k http://localhost:8090/kv?key=hello
The value is believe with the revision 4 at 127.0.0.1:6379

• Delete
ubuntu@ip-172-31-9-140:~$ curl -X DELETE -k http://localhost:8090/kv?key=hello
The key [hello] has been removed at 127.0.0.1:6379
```

Owner: *Jun Shao*



## 2. Demo

Regionless KV store architecture for “session” consistency

**“Naturally”**

*“as an application or without major changes”*

# “Strong consistency” vs “Session Consistency”



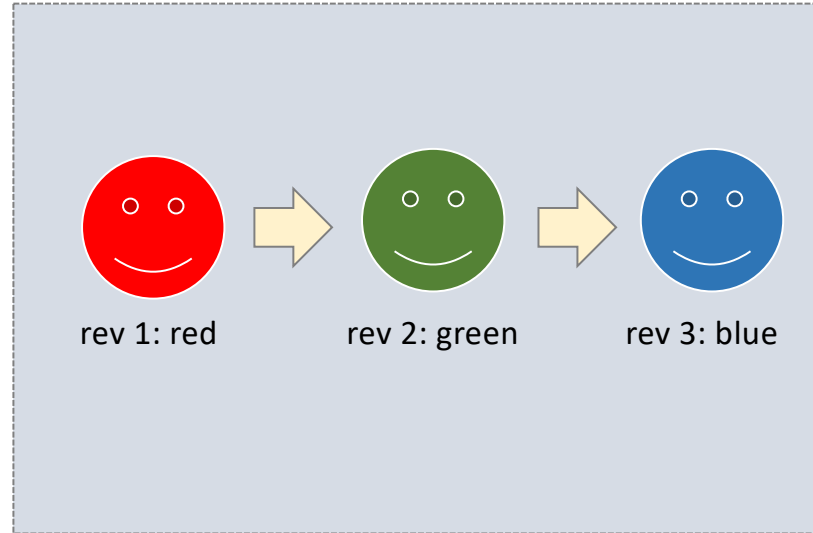
Strong consistency





KV Store

Game Session: **RUNNING**



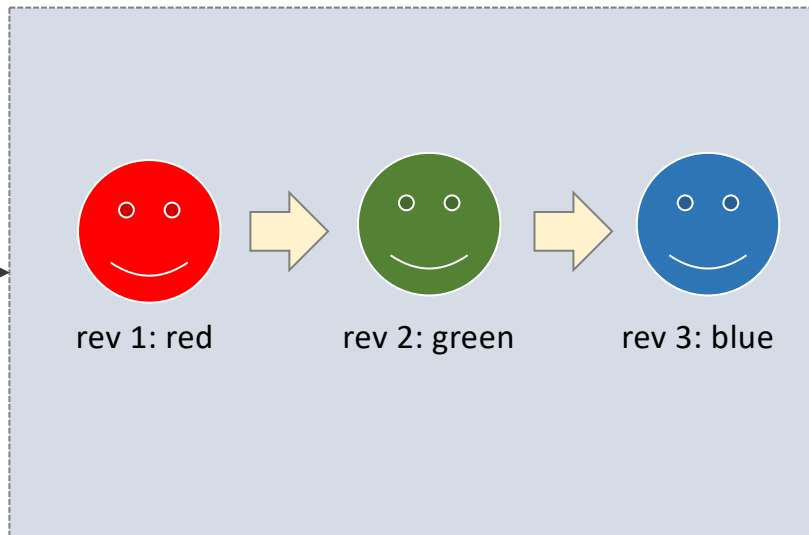
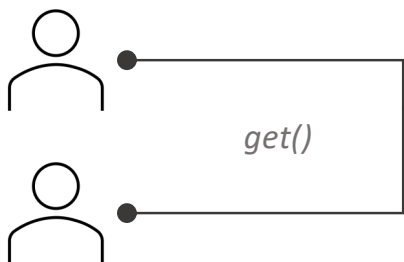
# "Session"



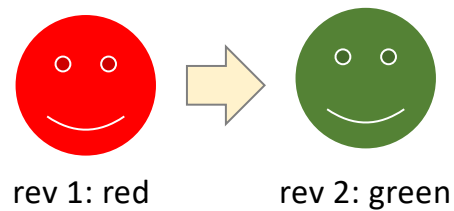
KV Store

Game Session: **RUNNING**

In-session readers



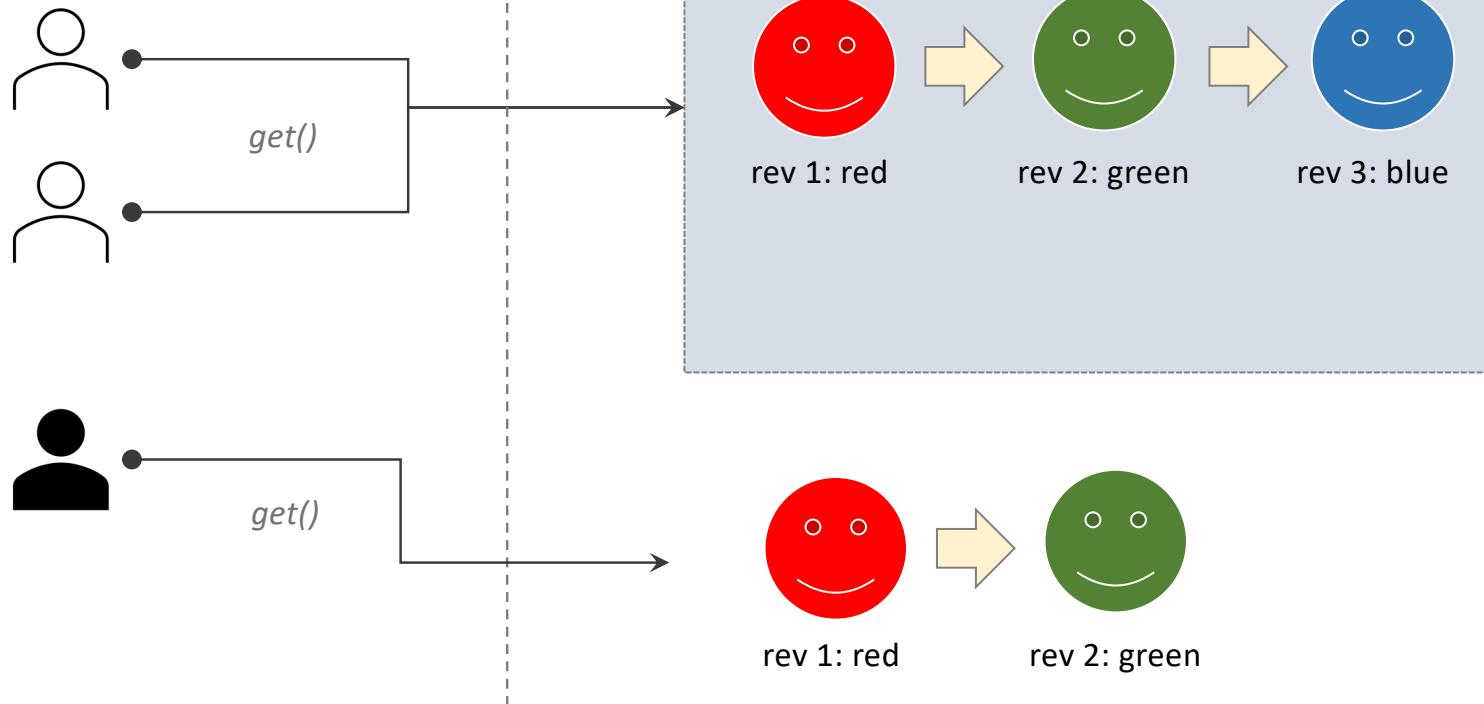
Out-of-session readers



# "Session"

KV Store

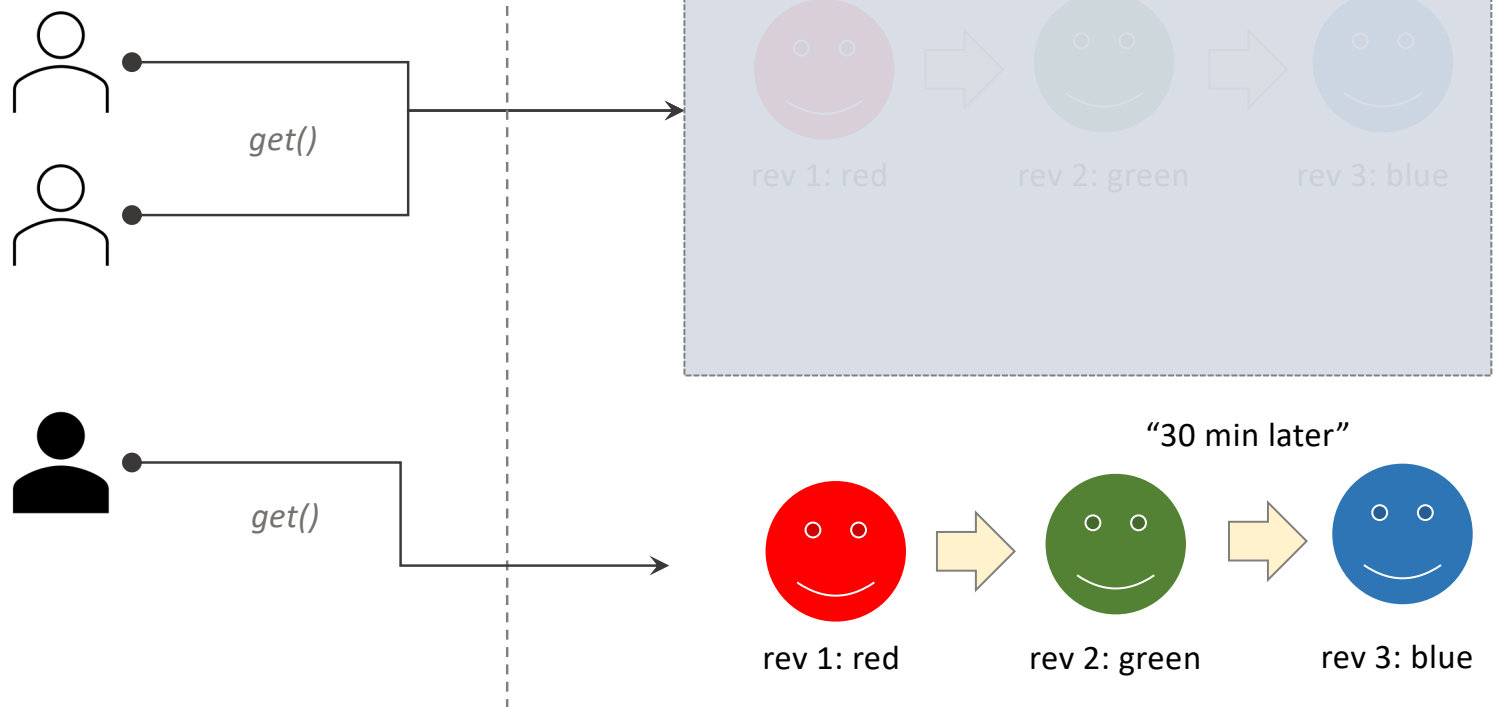
Game Session: **OVER**




“Session”

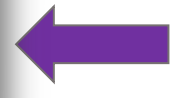
KV Store

Game Session: **OVER**



 Session credential

```
// setup session credential
std::pair<std::string, std::string> session_challenge = {
    u1: store.fetch_session_challenge(),
    u2: "thunder"
};
bool done_reading = store.in_session_get( session_challenge_answer: session_challenge,
                                         key,
                                         starting_revision: start_rev,
                                         &: found_values);
```

 In-session *get()*  
(read from async replica)

Out-of-session *get()*  
(read from sync replica)



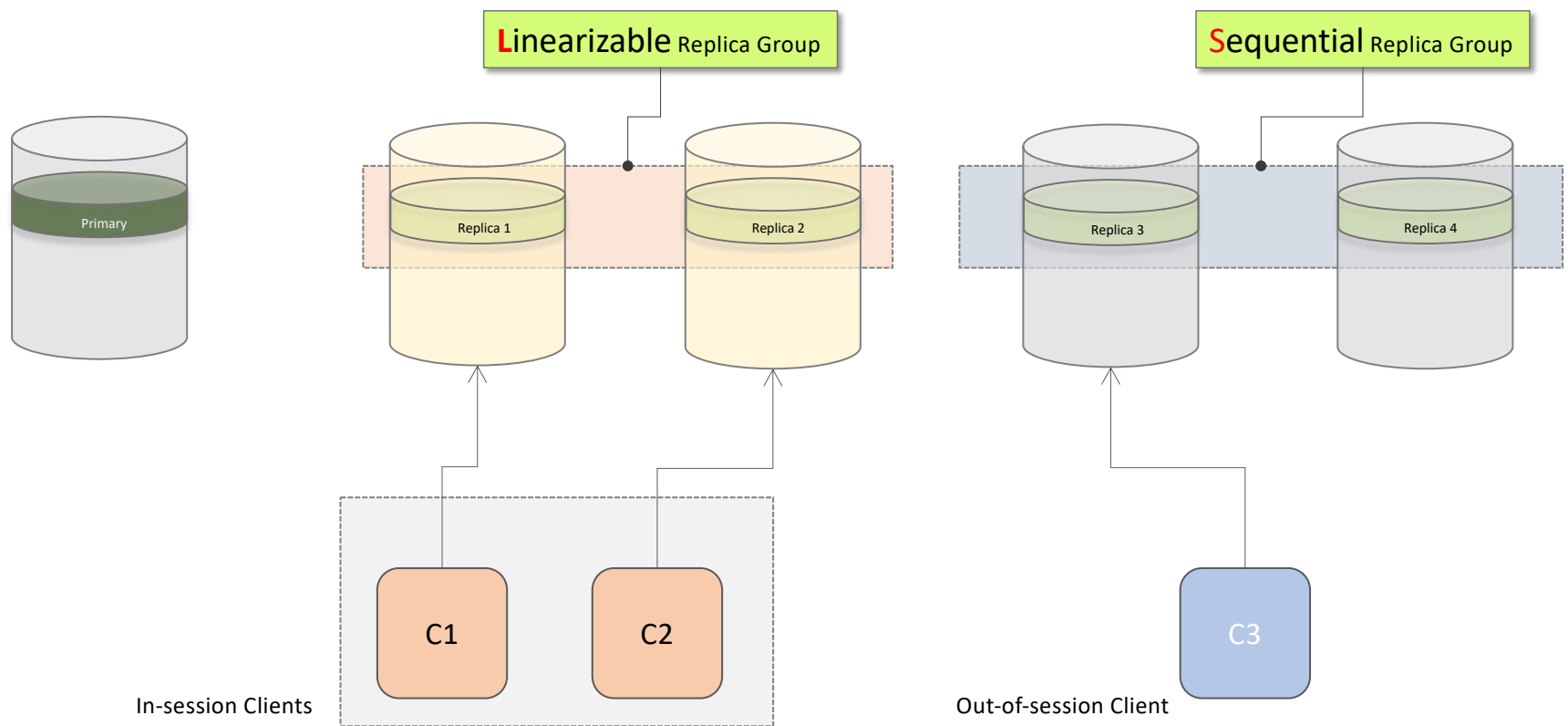
```
store.get(key,
          starting_revision: start_rev,
          &: found_values,
          consistency: SEQUENTIAL_CONSISTENCY);
```

\*Read-uncommitted Isolation for demo purpose



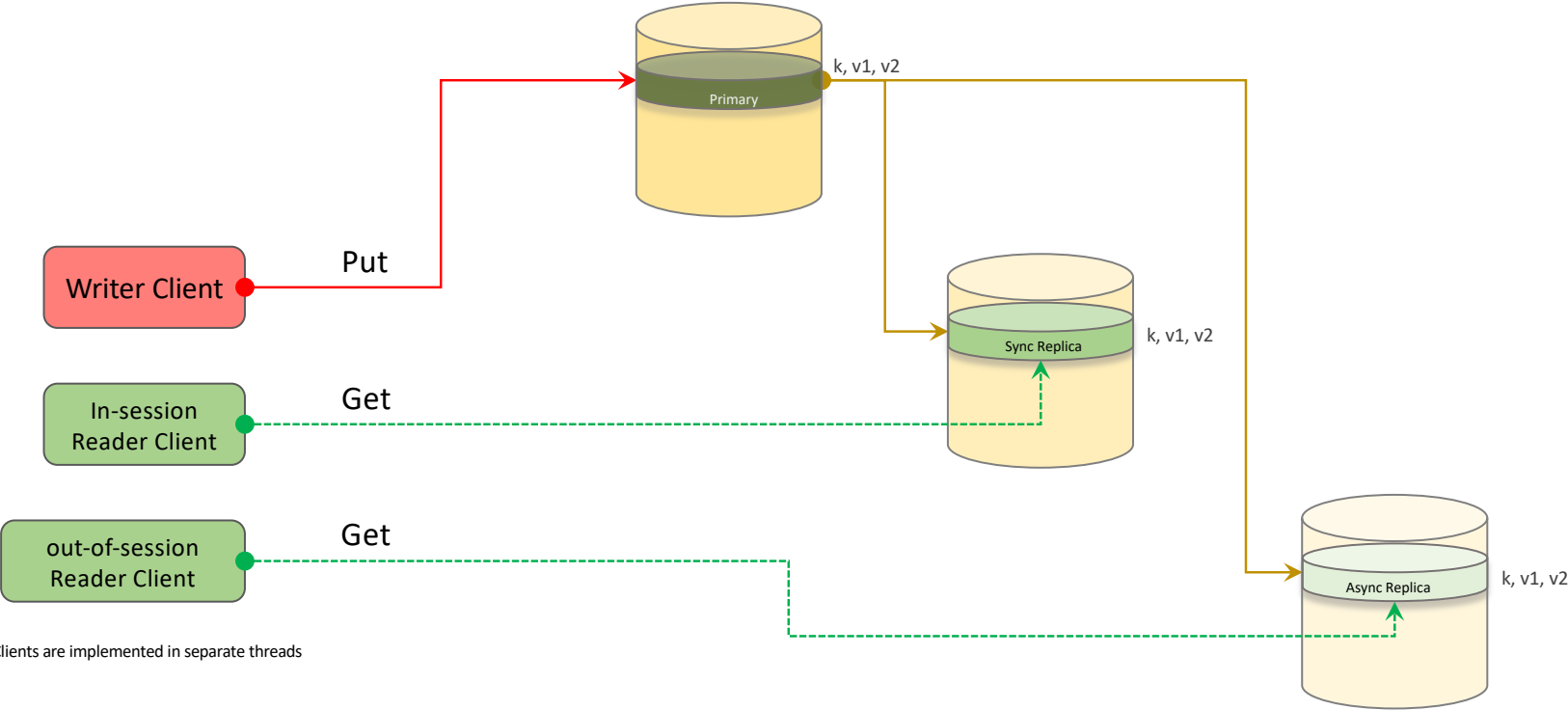


# Replication Consistency by Group



**Time: T 1**

- 1.1 Writer: {k1, v1}, {k2, v2} → primary
- 1.2 {k1, v1}, {k2, v2} → all replicas
- 1.3 Both clients can read {k, v1}, {k, v2}



\*Clients are implemented in separate threads

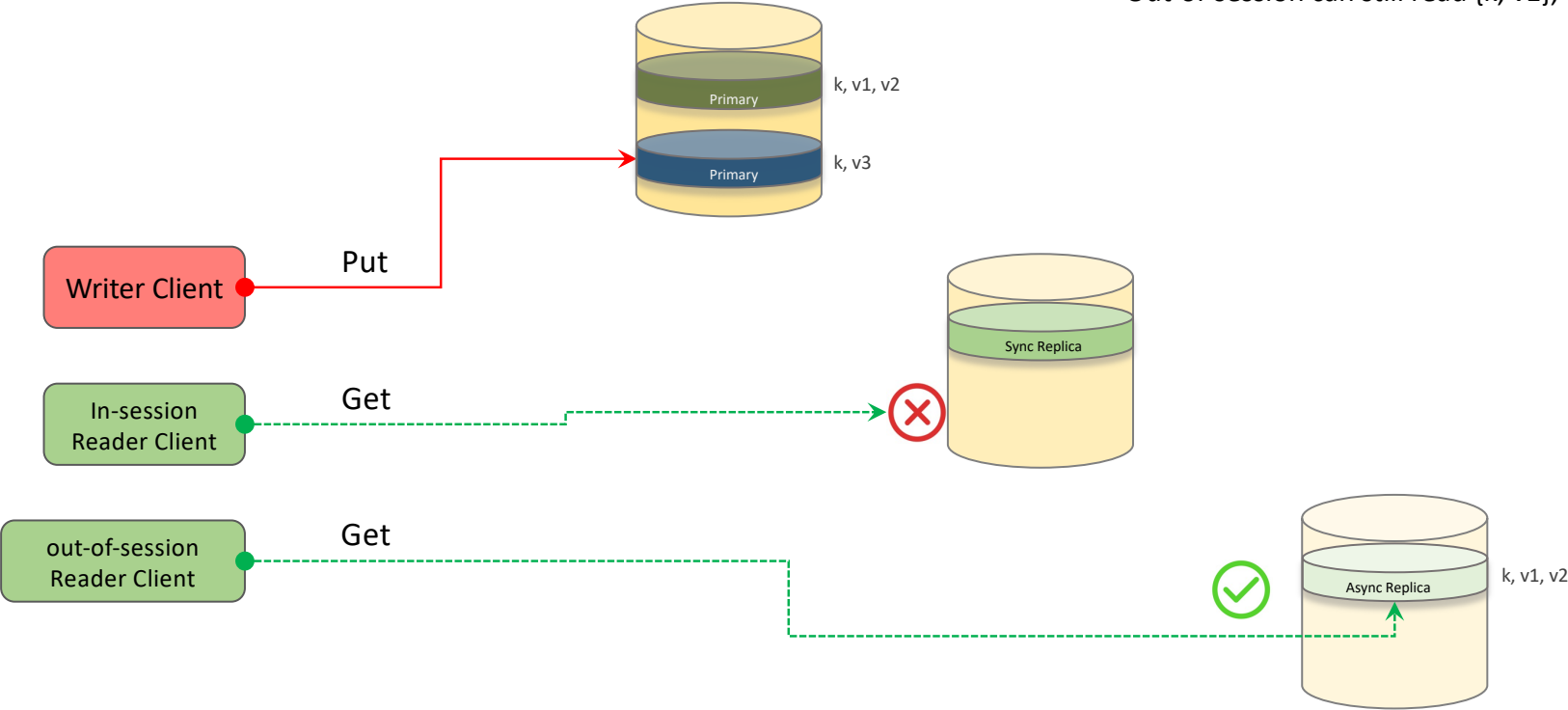
\*Read-uncommitted Isolation for demo purpose

**Time T 2:**

2.1 {k, v3} → primary

2.2 Readers

- In-session clients blocked on reading
- Out-of-session can still read {k, v1}, {k, v2}



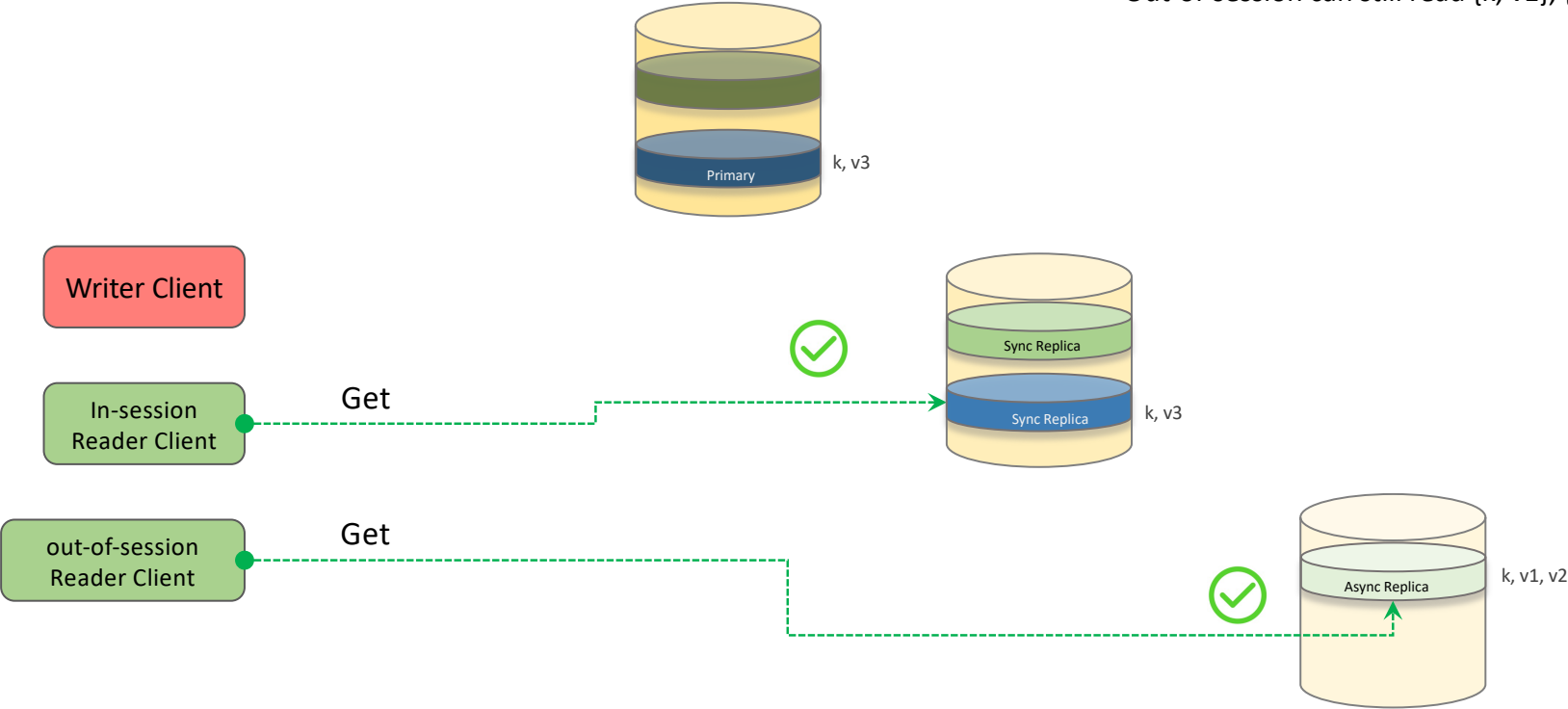
\*Read-uncommitted Isolation for demo purpose

**Time T 3:**

3.1 {k, v3} → sync replica

3.2 Readers

- In-session clients can now read up to the 3<sup>rd</sup> kv
- Out-of-session can still read {k, v1}, {k, v2}



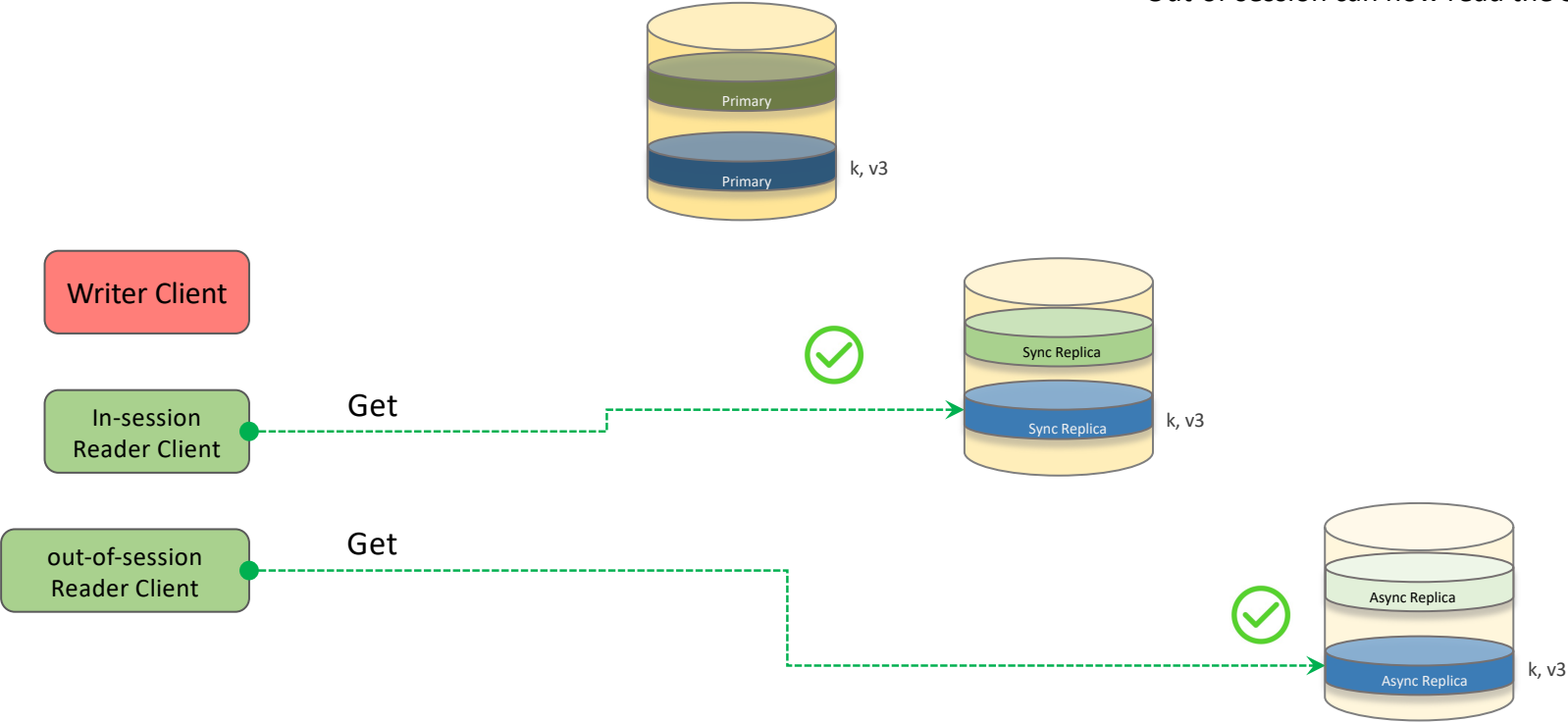
\*Read-uncommitted Isolation for demo purpose

**Time T 4:**

4.1 The 3<sup>rd</sup> kv propagated to async replica

4.2 Readers

- In-session clients can now read the 3<sup>rd</sup> kv
- Out-of-session can now read the 3<sup>rd</sup> kv



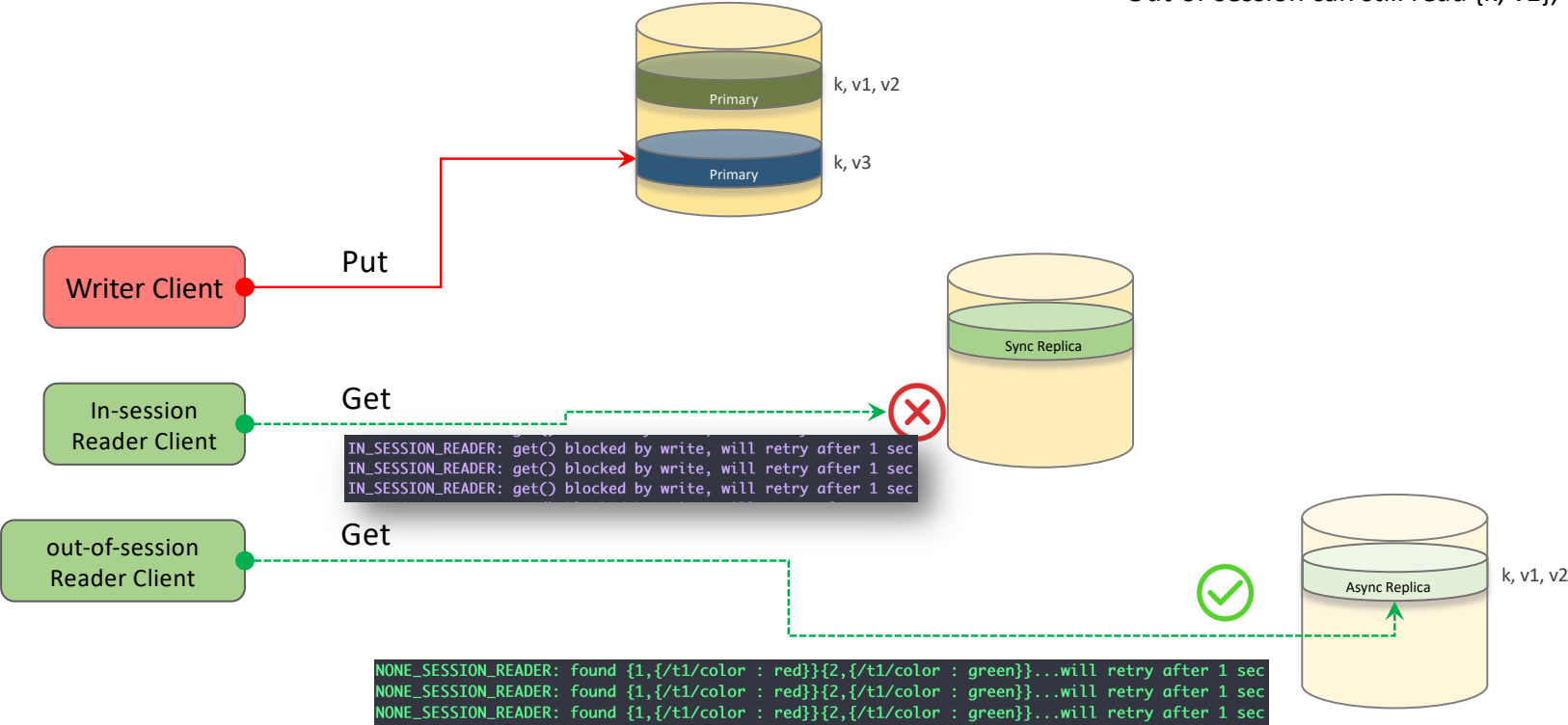
\*Read-uncommitted Isolation for demo purpose

**Time T 2:**

2.1 {k, v3} → primary

2.2 Readers

- In-session clients blocked on reading
- Out-of-session can still read {k, v1}, {k, v2}



\*Read-uncommitted Isolation for demo purpose



# Demo Summary



1. MVCC for versioned key-value pairs
2. In-session put() provides strong consistency (linearizable)
3. Out-of-session put() provides sequential consistency (linearizable)
  - Can be further extended to bounded sequential
4. Regionless store “natural” design