

API Types and Evolution

Pedro Félix

March 2020

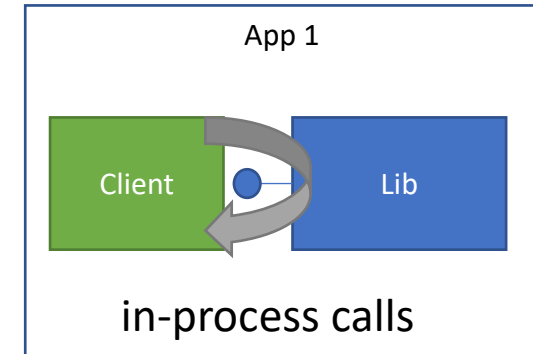
Summary

- In-process APIs versus remote APIs
- Comparing the evolution and versioning of in-process and remote APIs
- Remote APIs types
 - Client scenarios
 - Exposed functionality

In-process vs. remote APIs

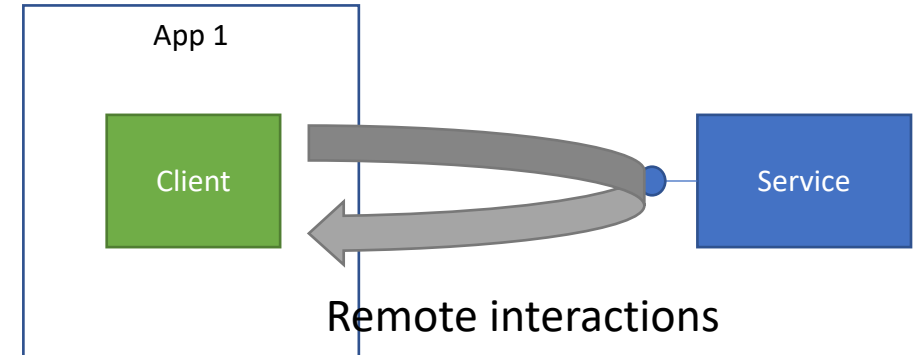
In process APIs

- Same security boundary
 - No authentication, no authorization, no confidentiality or integrity protection
- No I/O, synchronous calls
- Failure modes
 - Deterministic errors
 - Typically due to unsatisfied pre-requisites
 - Catastrophic errors typically impact the complete application
- Same type system
 - Object sharing between client and API provider (i.e. Library)
- Return values or exceptions



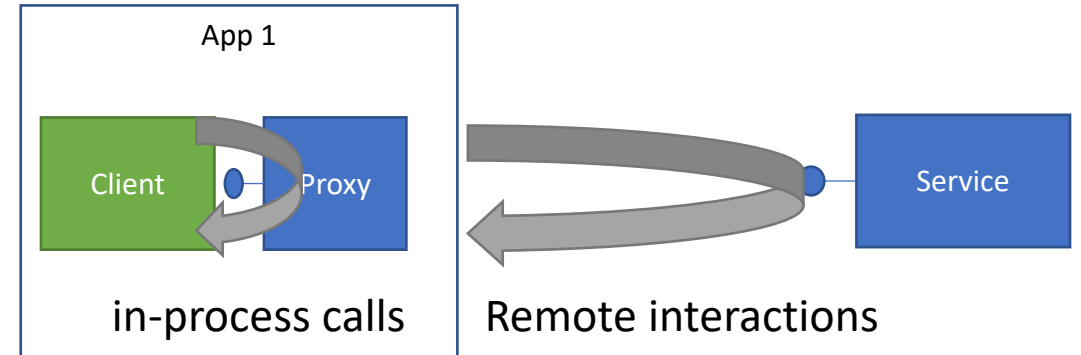
Remote APIs

- Different security boundaries
 - Authentication, authorization,
 - Confidentiality and integrity protection
- I/O, asynchronous calls
- Failure modes
 - Indeterministic errors
 - Transient errors
 - Communication errors
- Different type systems
 - No object sharing
 - Representation based on agreed formats
- **Message exchanges, not calls.**
- Non-success messages and communication errors
 - HTTP origin-servers don't throw exceptions
 - They responds with non-success messages or fail to communicate



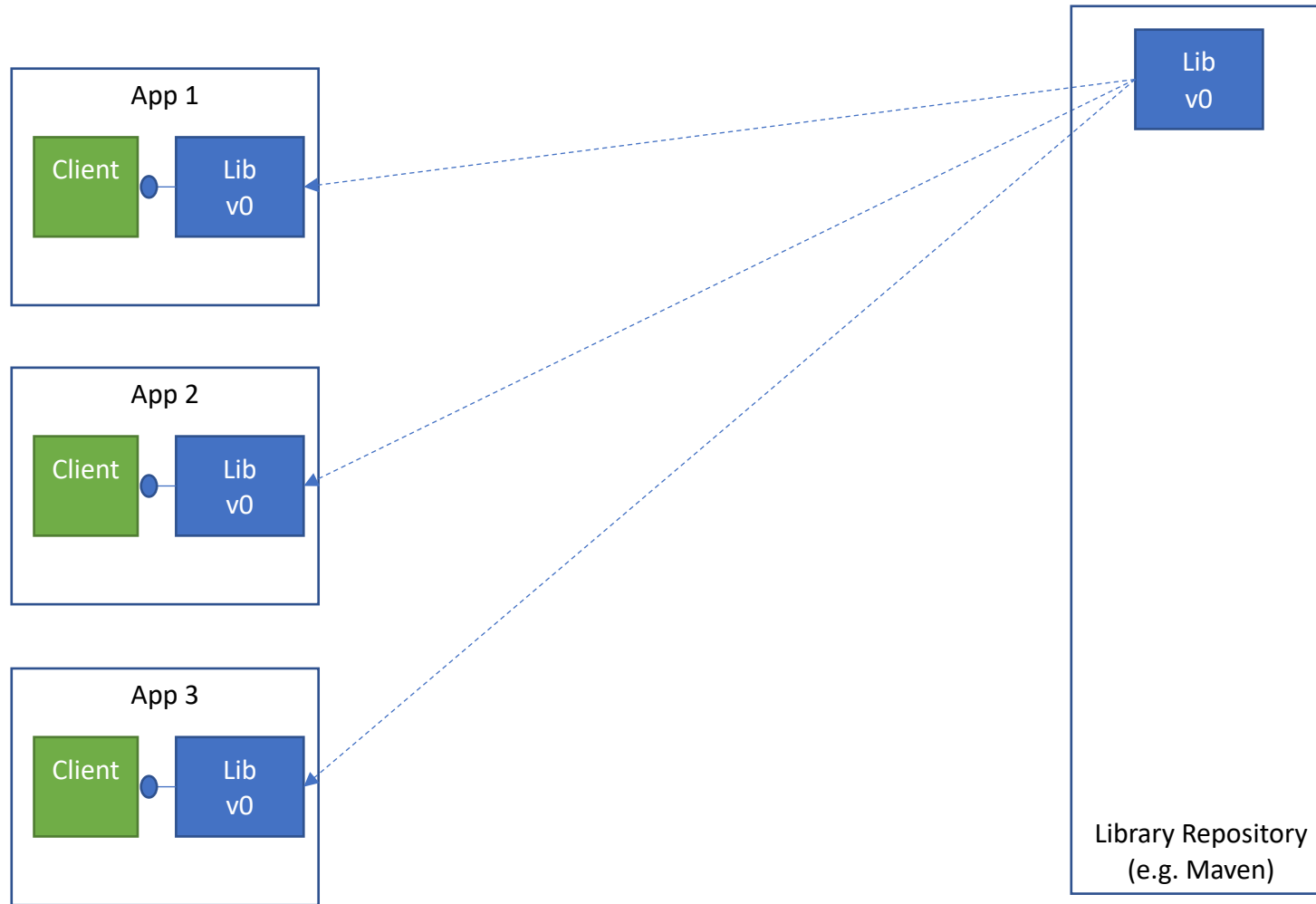
Remote APIs

- Proxies
 - Local-representant of a remote service
 - As-if the service was in-process
- Leaky abstraction
 - The remote characteristic cannot be abstracted away
 - Due to the reasons presented before
- In the end, the client app is still interacting with a remote service

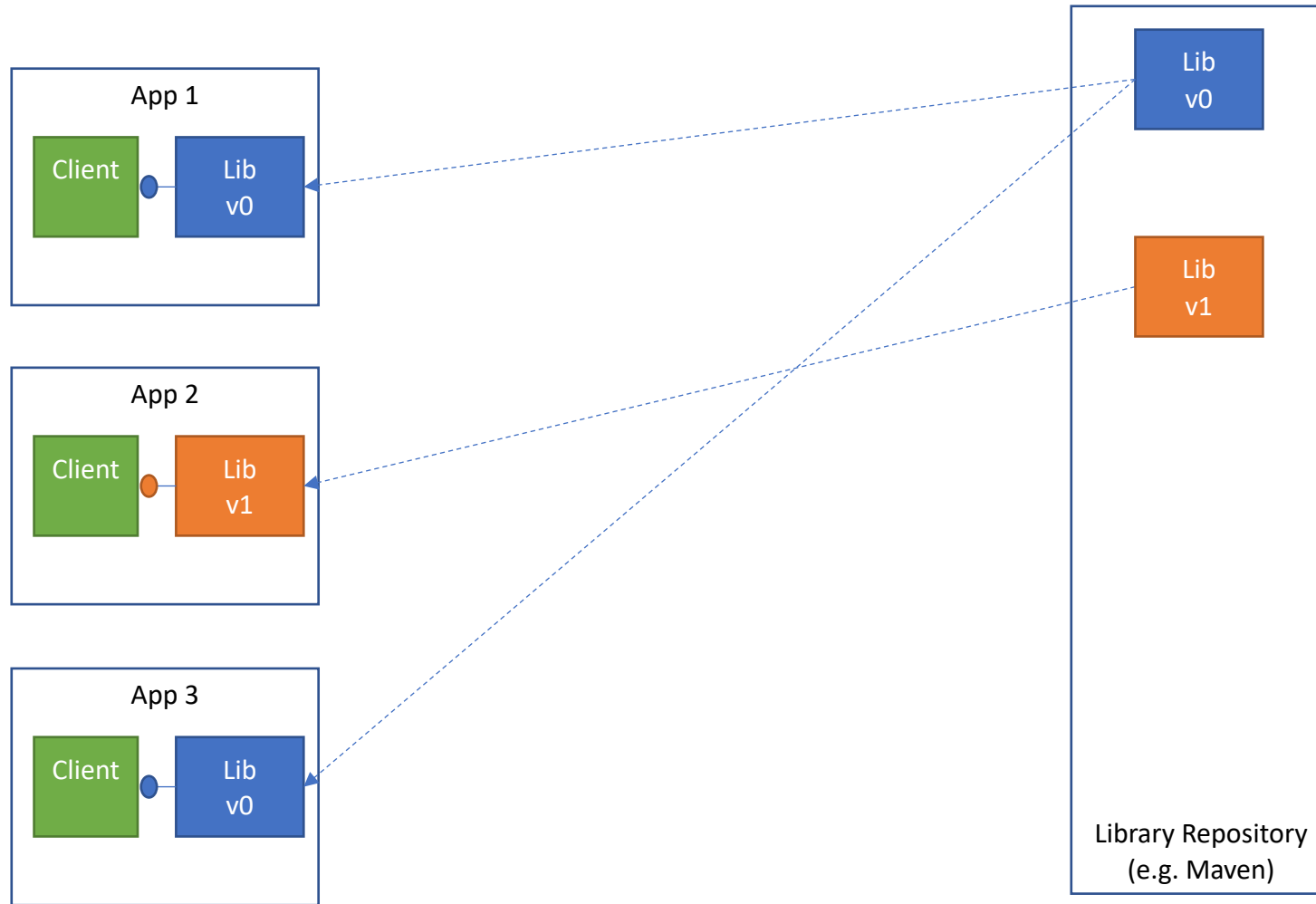


Evolution and versioning

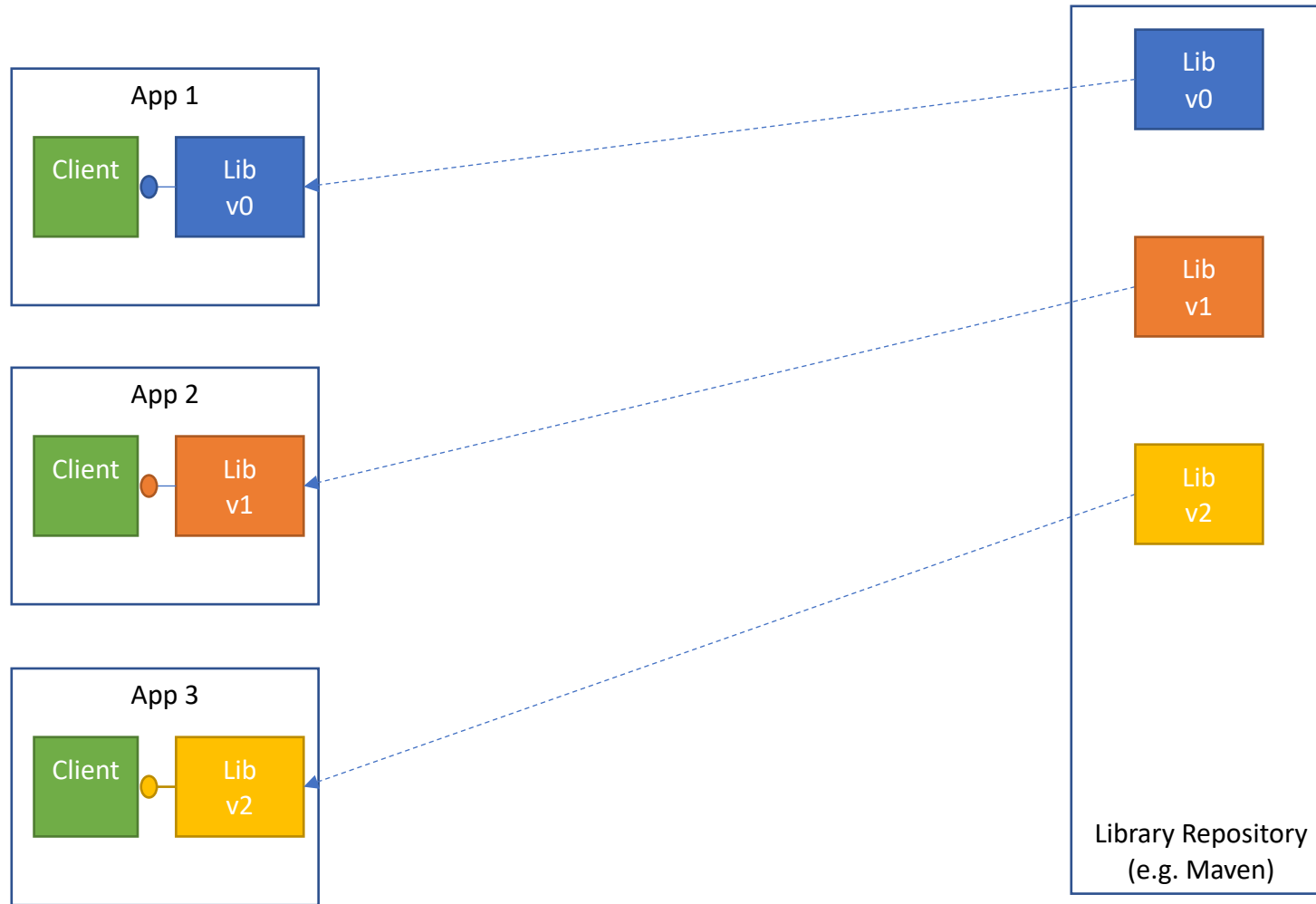
Versioning



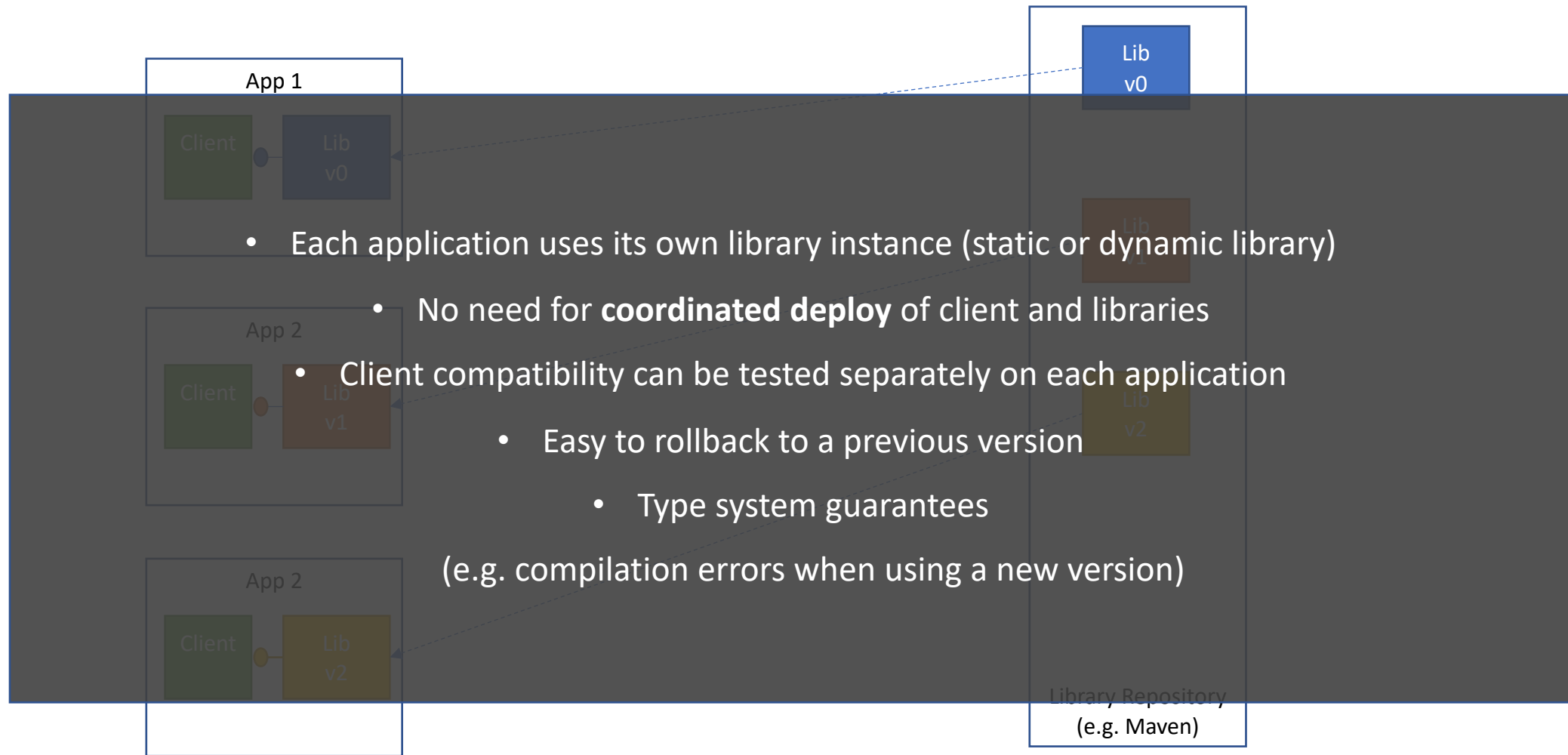
Versioning



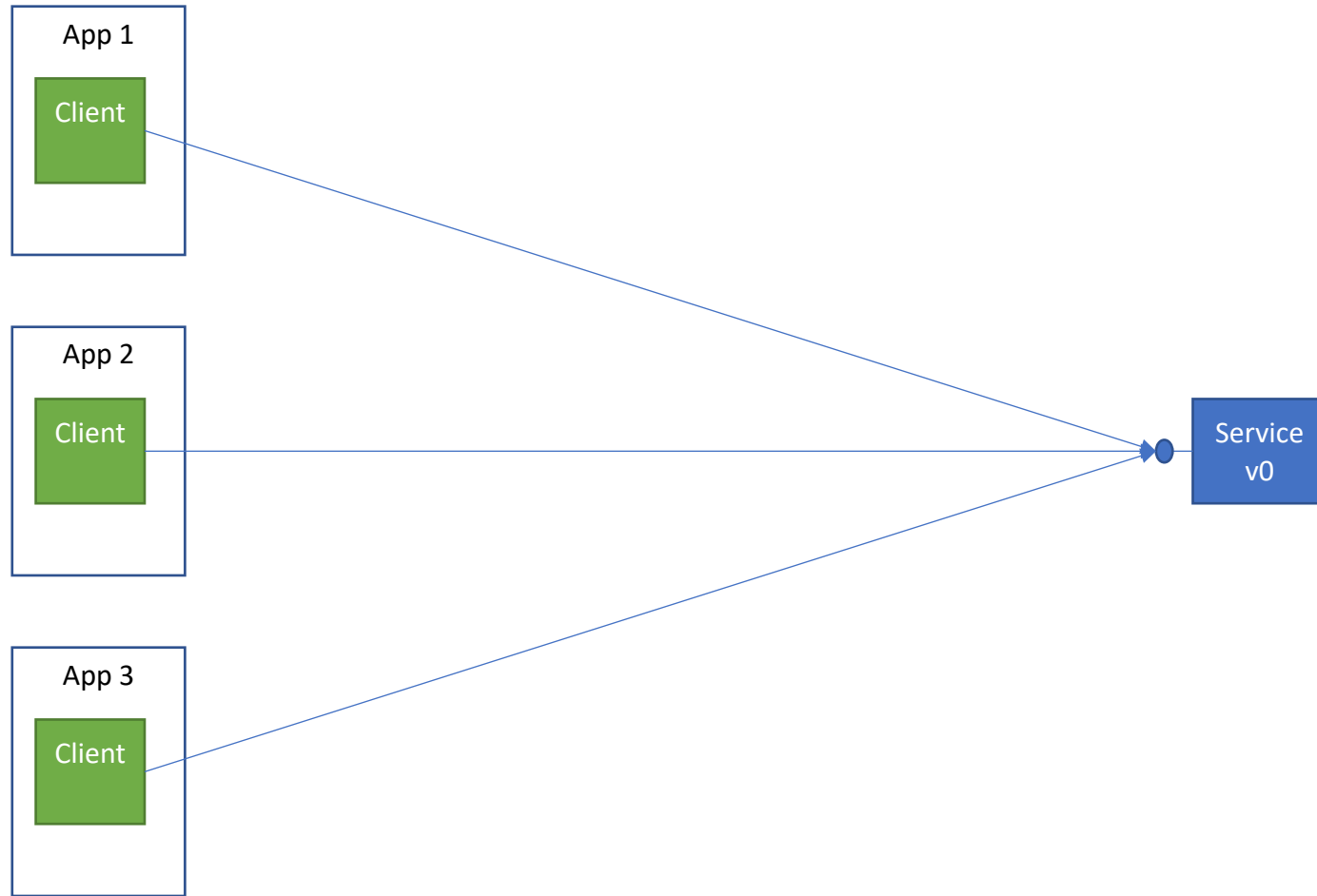
Versioning



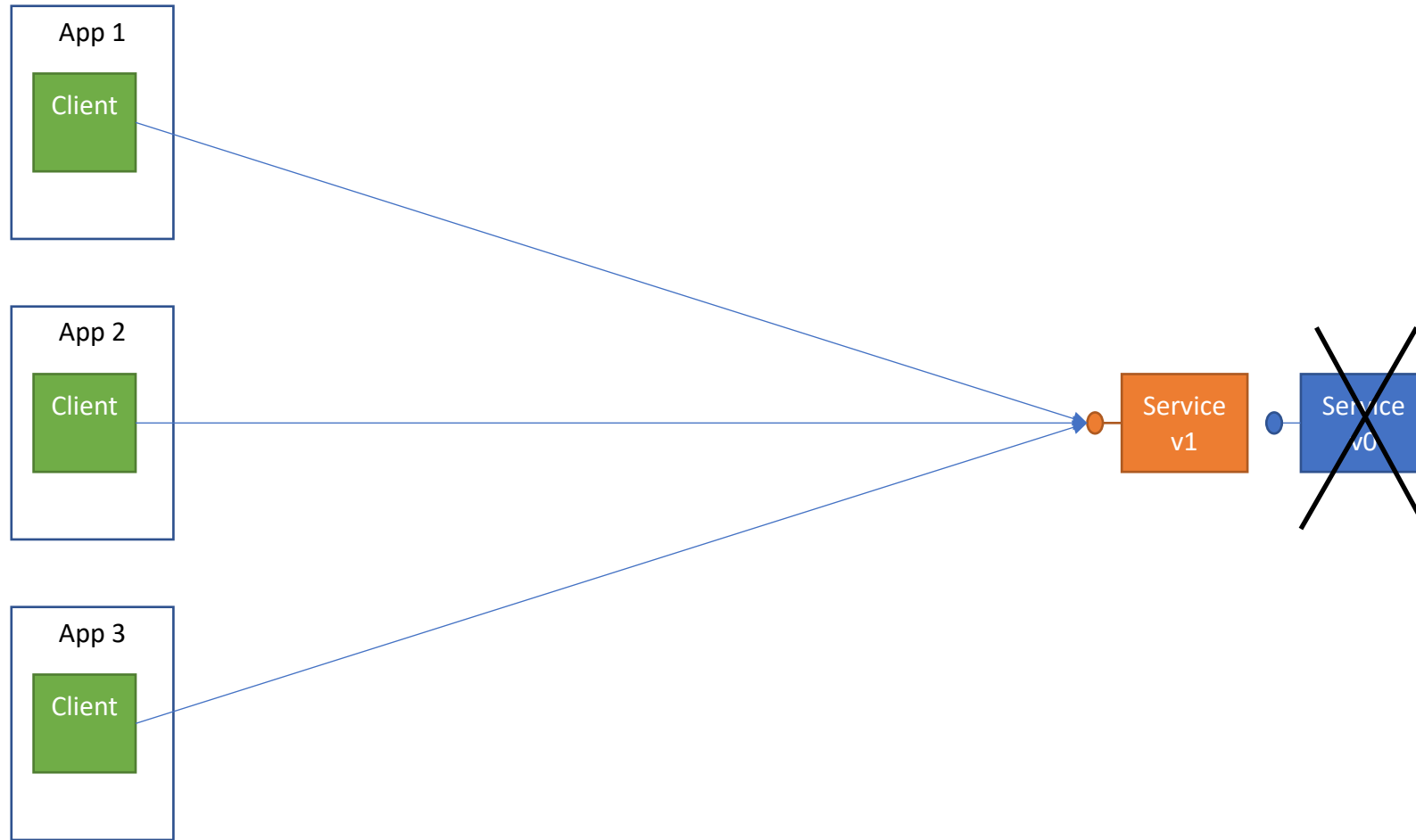
Versioning



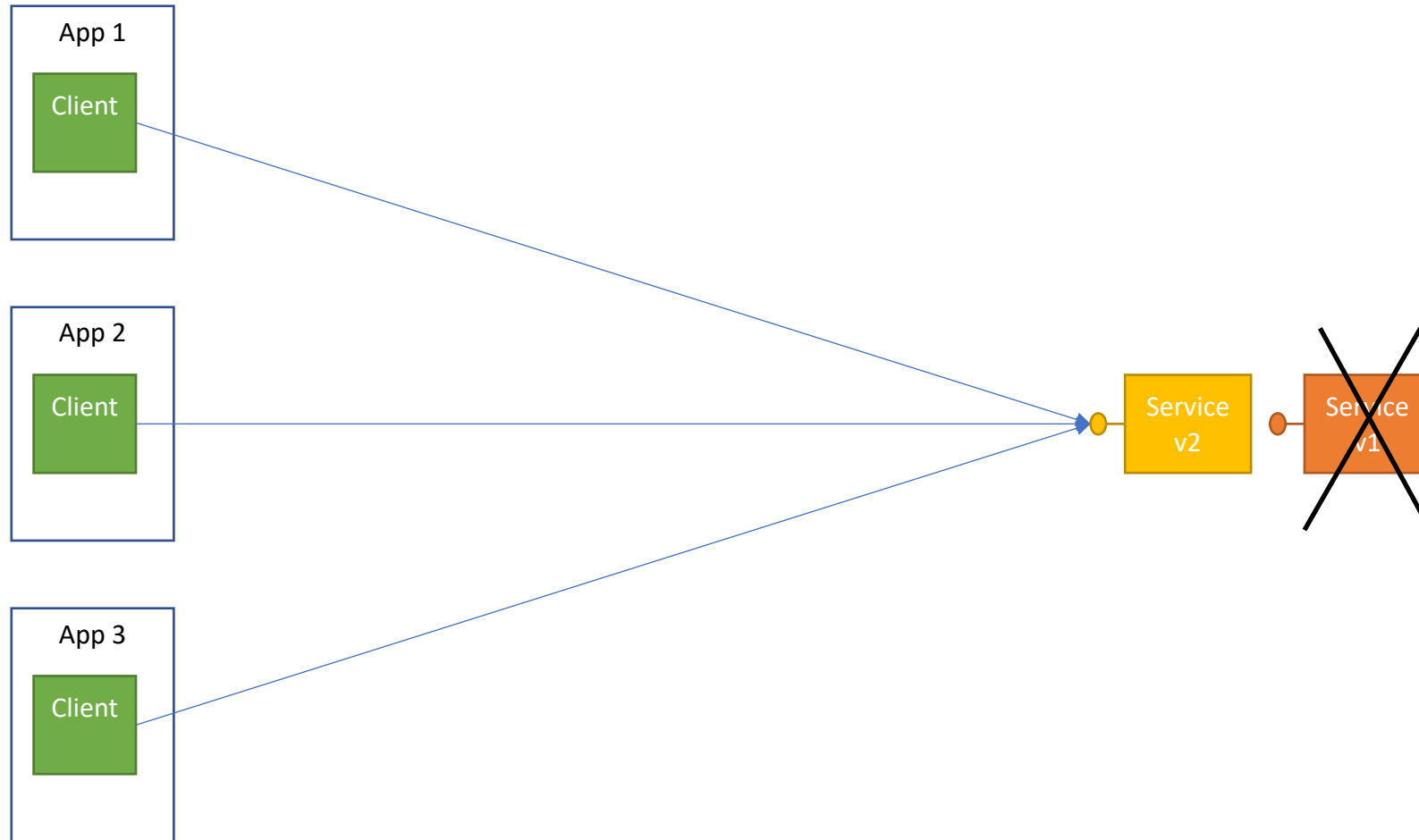
Versioning



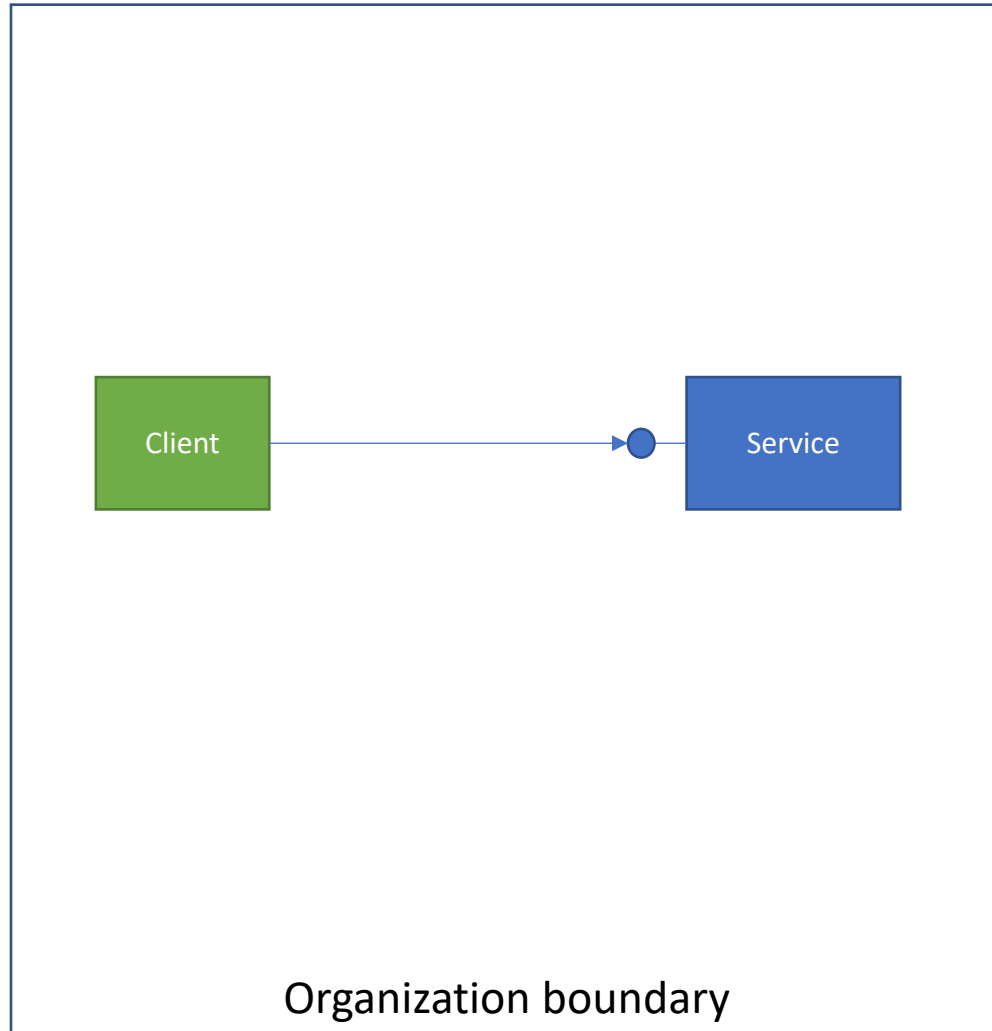
Versioning



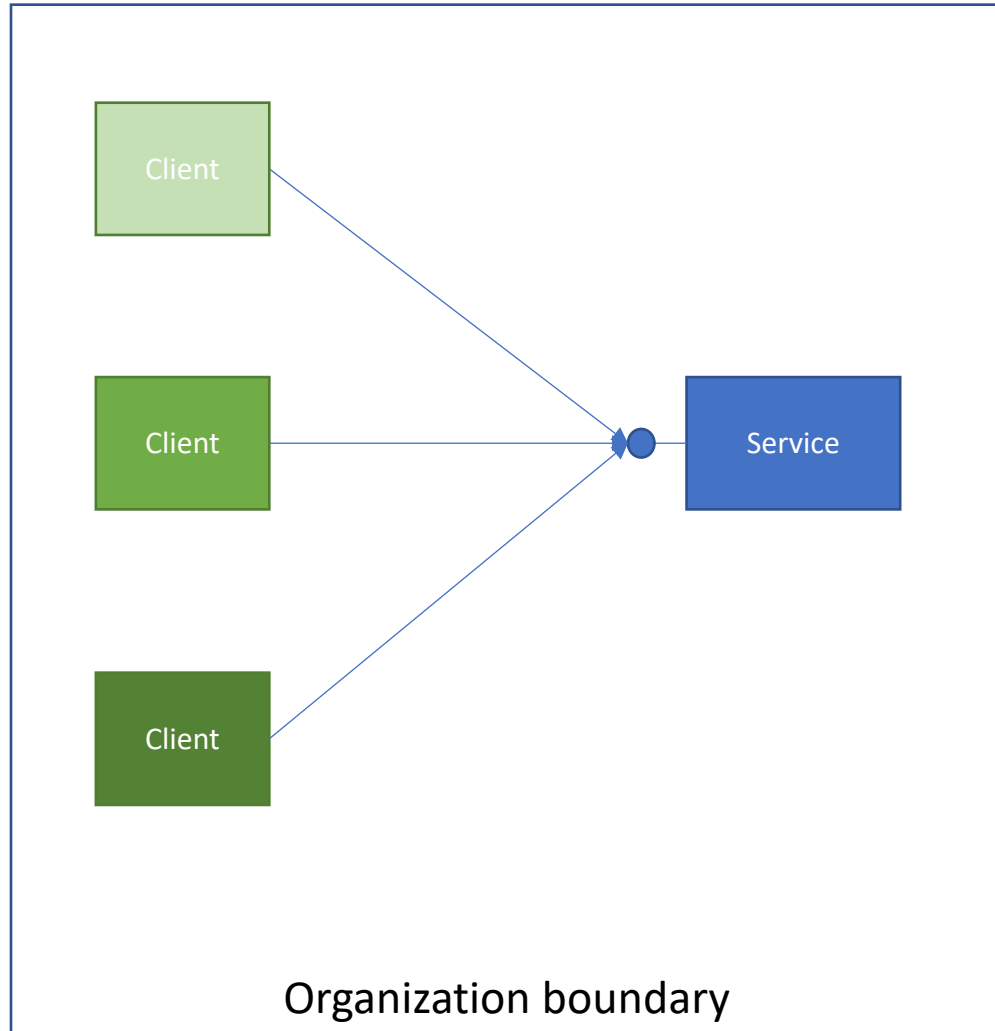
Versioning



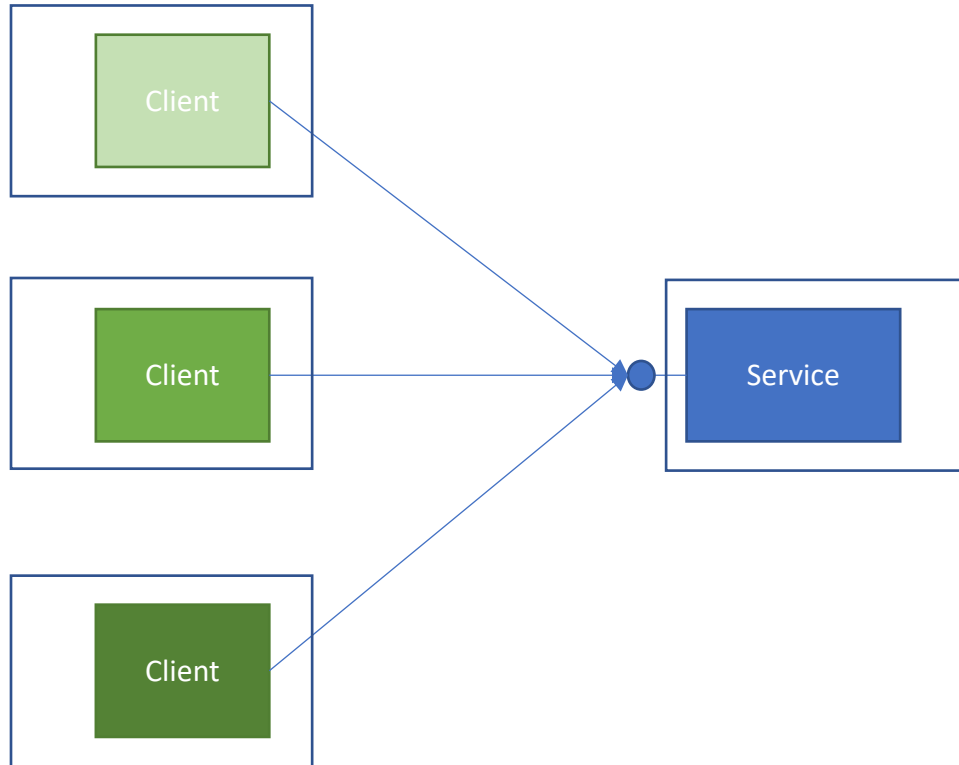
Client scenarios



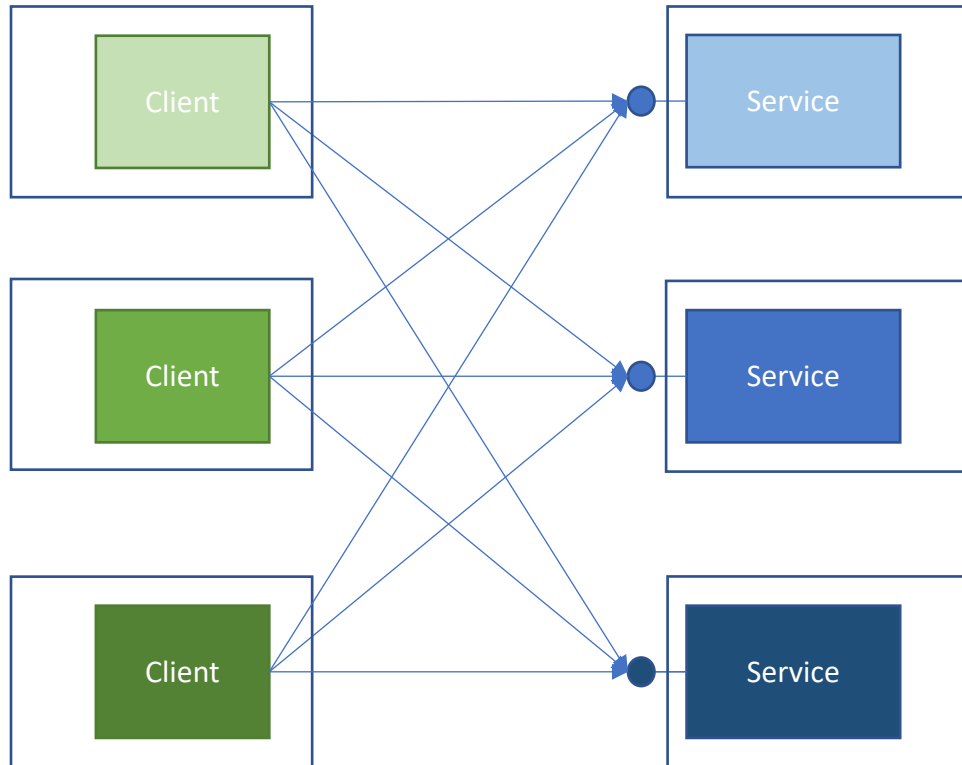
- Single client
- Client and service on the same organization boundary
 - Simpler API contract coordination
 - Simpler coordinated deploy
 - Depends on the client lifecycle
Web app is not the same as iOS app
- Short time span
 - E.g. TV show contests system
 - E.g. Elections systems



- Multiple clients make coordinated evolution harder
- Client and service on the same organization boundary



- Multiple clients in distinct organizations makes coordinated evolution even harder



- API with multiple consumers
- And multiple providers
- API is a specification
 - E.g. OAuth 2.0
 - E.g. Git remote protocol
- Specification evolution implies coordination between consumers and providers
- Long time spans
 - E.g. Decades
- See <https://www.crummy.com/writing/speaking/2015-RESTFest/>

API functionality



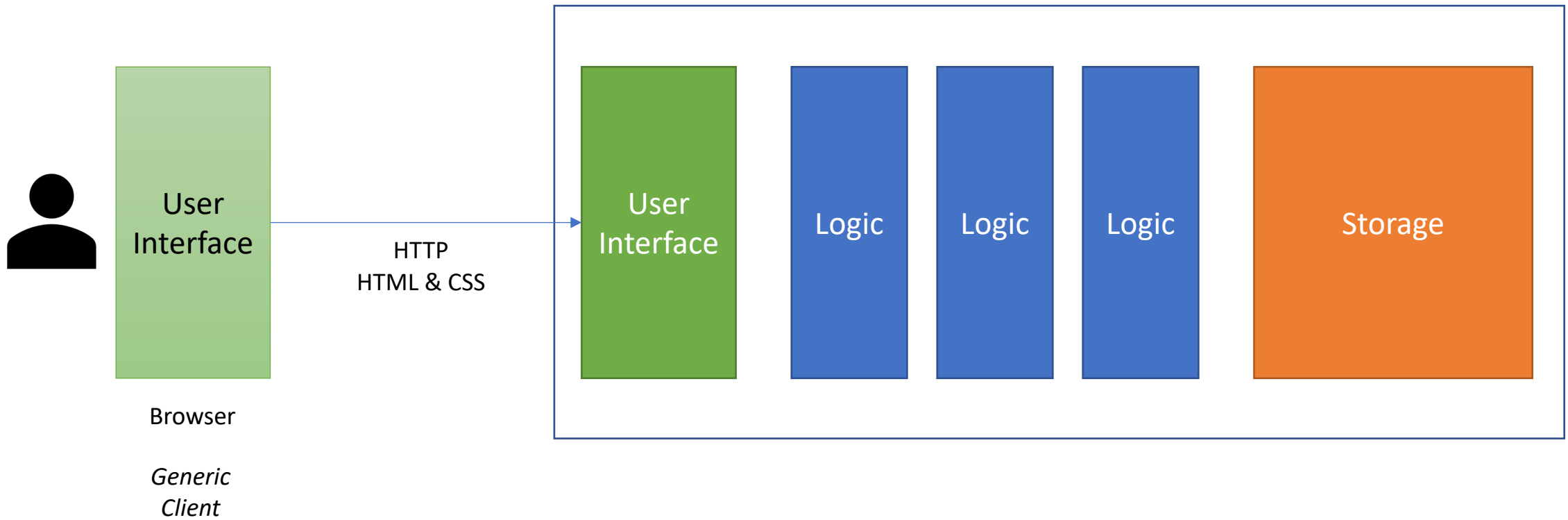
User
Interface

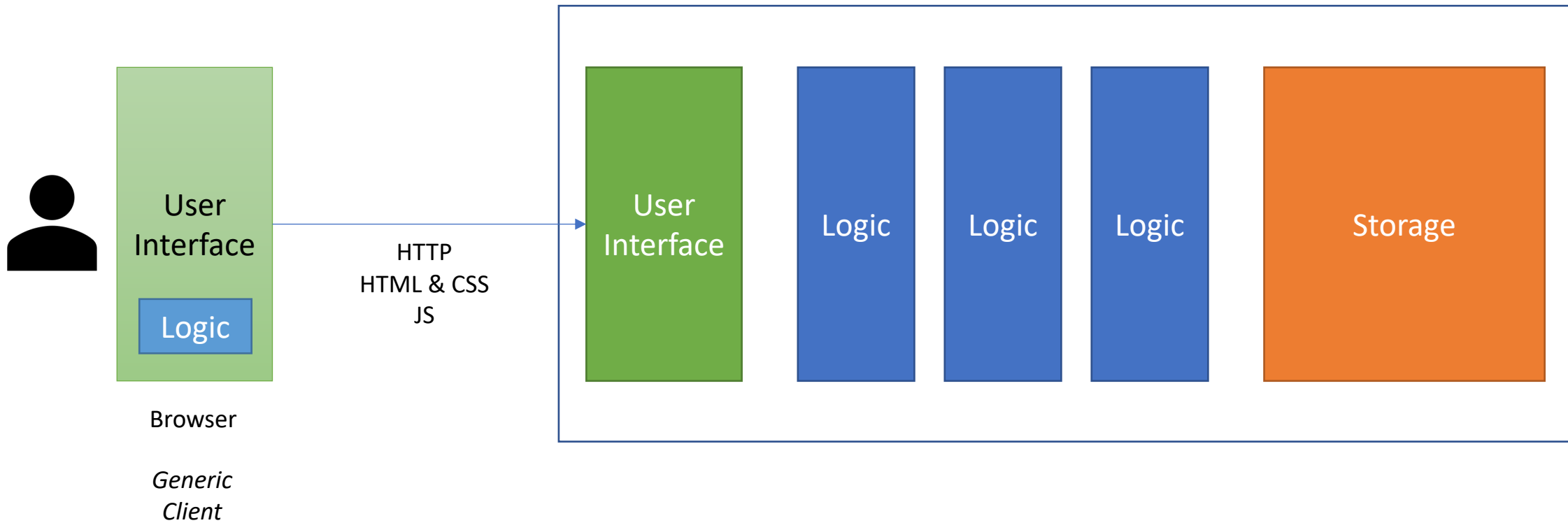
Logic

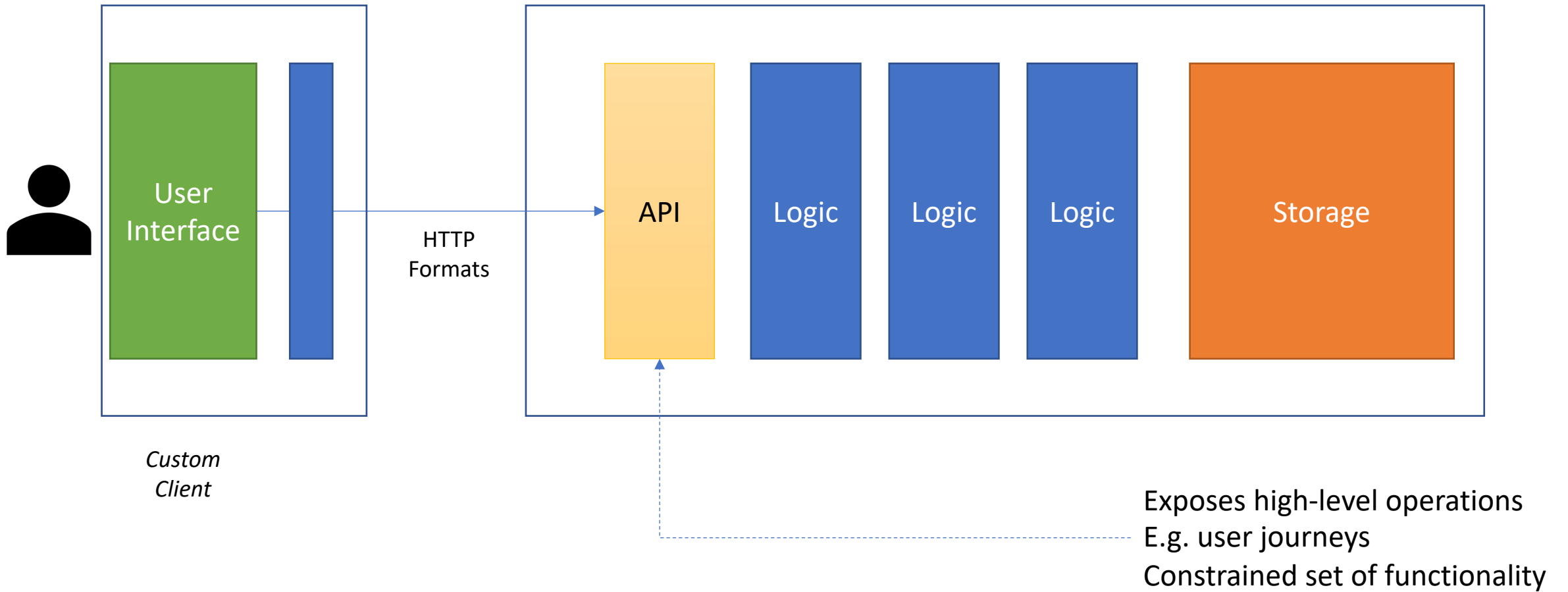
Logic

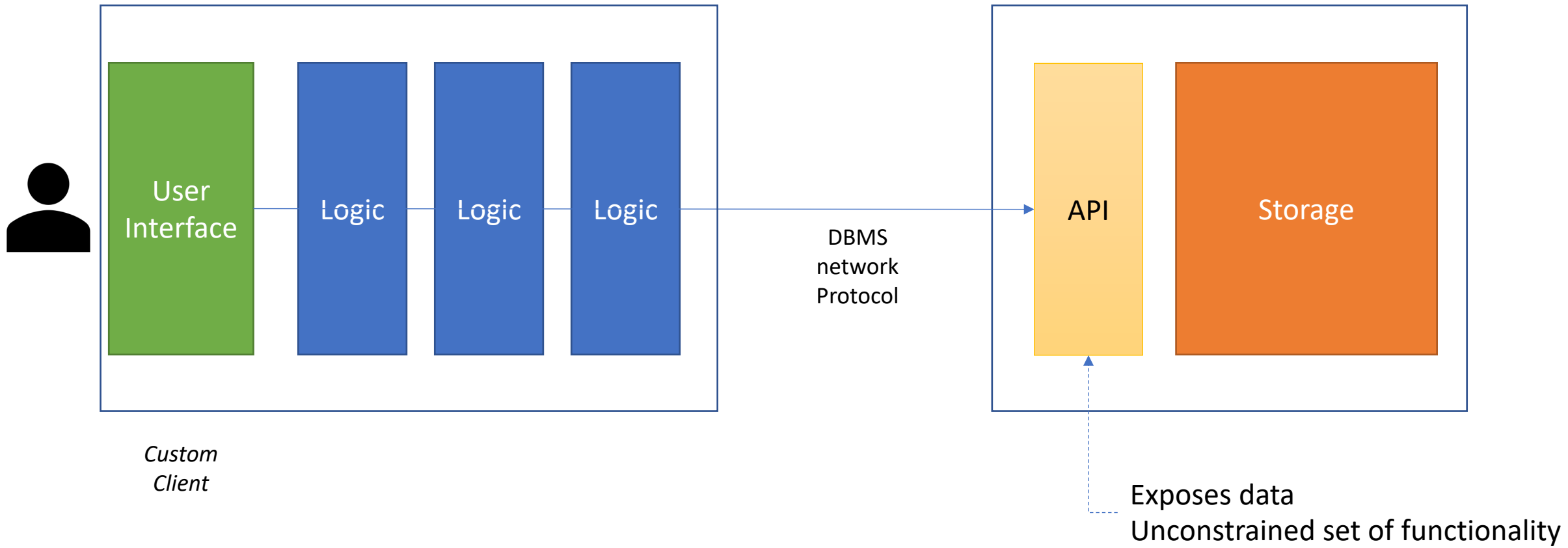
Logic

Storage









Conclusion

- Remote APIs are not the same thing as in-process APIs.
- Versioning techniques for in-process APIs are not easily extended to remote APIs.
- Remote APIs can be used in different contexts.
 - Ranging from short-lived APIs with a single client, run by the same organization...
 - ... to long-lived APIs with multiple providers and multiple clients.
- When designing APIs all these aspects need to be taken into consideration.
- There isn't a "one size fits all" approach