

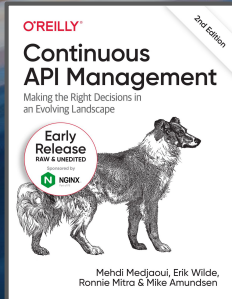
Service and API Migration at Speed

Mike Amundsen

@mamund

[linkedin.com/in/mamund](https://www.linkedin.com/in/mamund)

[youtube.com/mamund](https://www.youtube.com/mamund)





Mike Amundsen
@mamund

The
Pragmatic
Programmers

Design and Build Great Web APIs

Robust, Reliable, and Resilient



Mike Amundsen
edited by Katharine Dvorak

O'REILLY®

Continuous API Management

Making the Right Decisions in
an Evolving Landscape

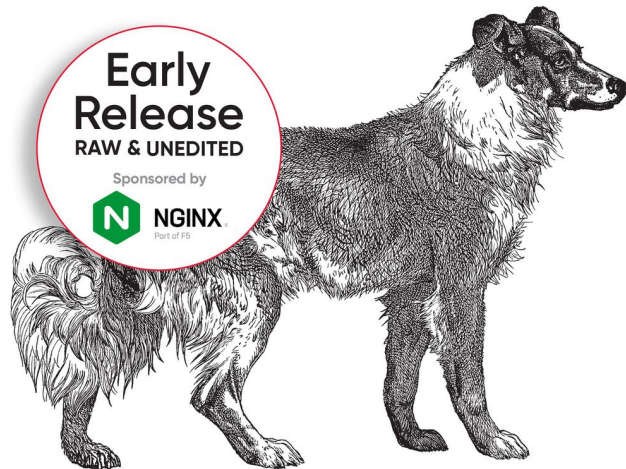
2nd Edition

Early
Release
RAW & UNEDITED

Sponsored by



NGINX
Part of FS



Mehdi Medjaoui, Erik Wilde,
Ronnie Mitra & Mike Amundsen



Let's get started...

A Look Ahead

- Why?
- Unlocking Business Value
- Basic Principles
- Stabilizing Interfaces
- Transforming Implementations
- Adding Functionality
- Rinse and Repeat







Why?



Reasons to migrate to microservices

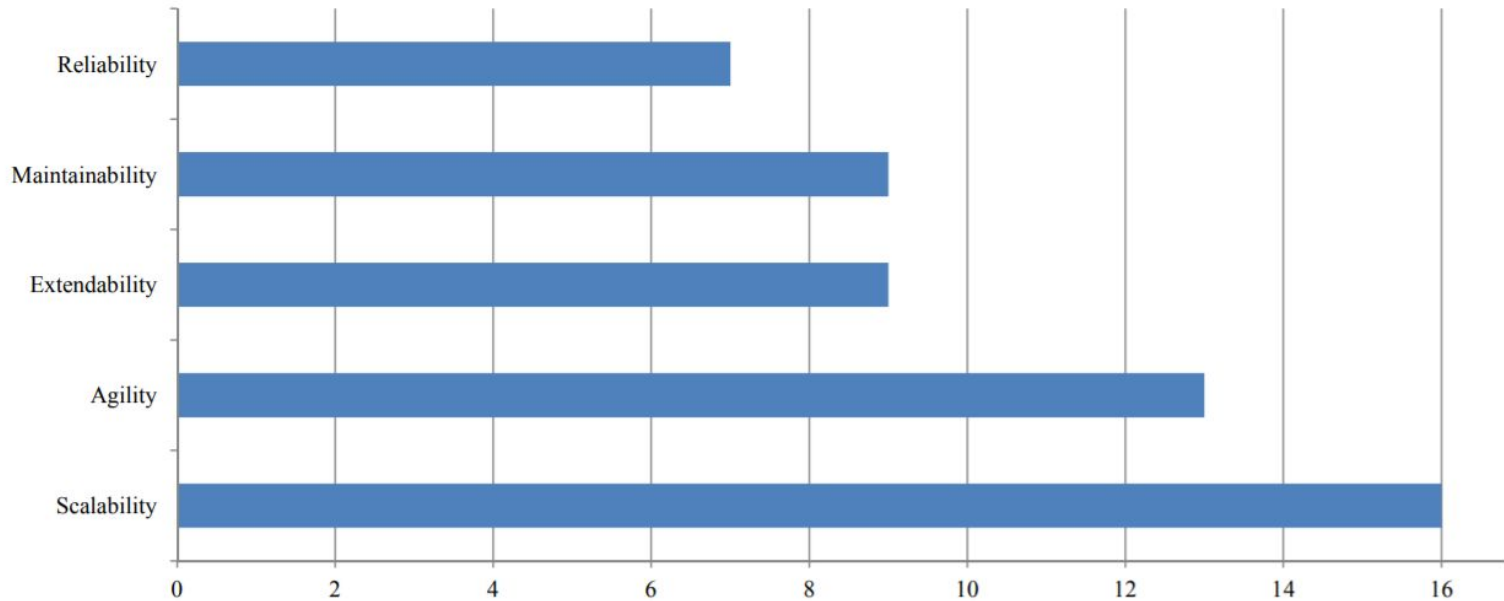
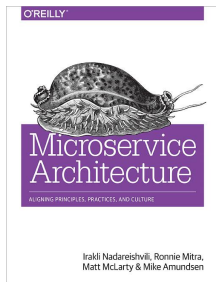
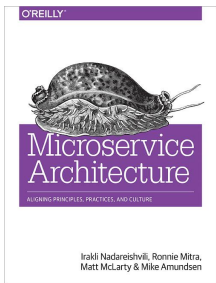


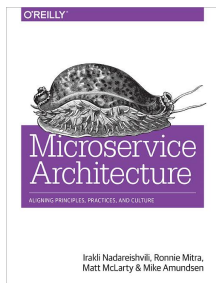
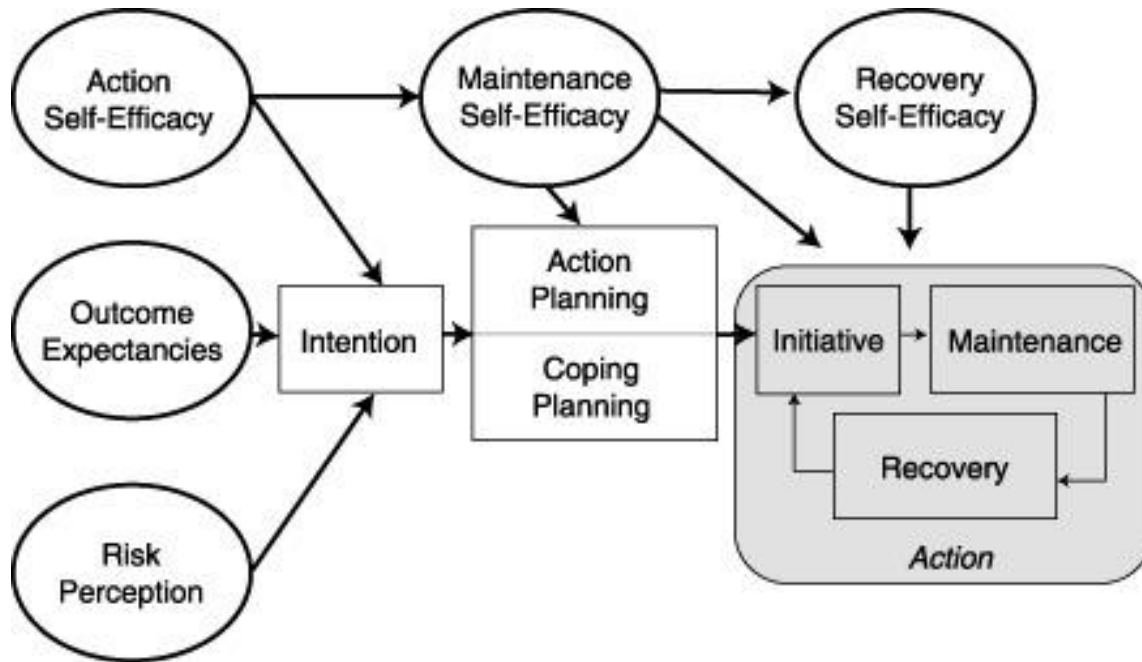
Fig. 2. Participants' primary intention for using a microservice architecture

<https://www.researchgate.net/publication/328216639>



Irakli Nadareishvili, Ronnie Mitra,
Matt McLarty & Mike Amundsen



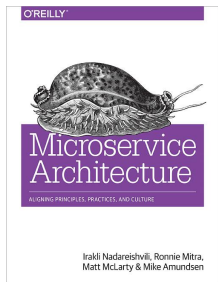
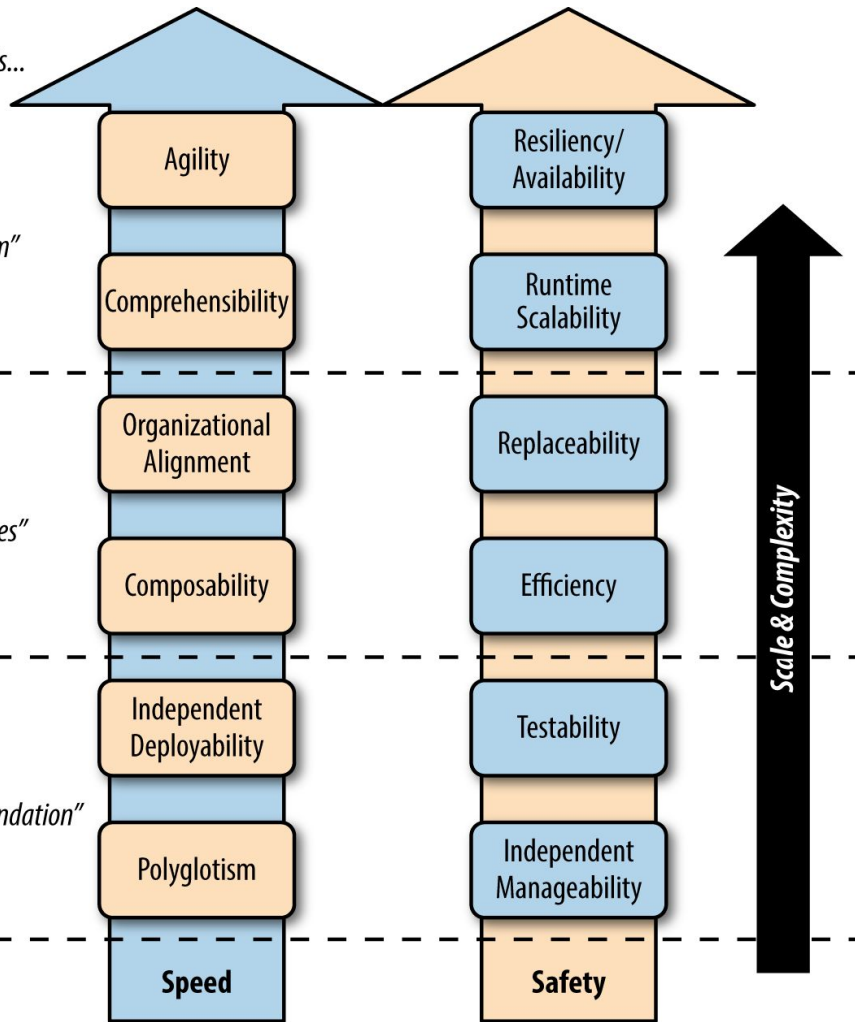


A microservice architecture that is...

Systematized
"Design the system"

Cohesive
"Design the services"

Modularized
"Establish the foundation"





O'REILLY



Microservice Architecture

A CONCISE, PRACTICAL, REAL-WORLD GUIDE

Irakli Nadareishvili, Ronnie Mitra,
Matt McLary & Mike Amundsen



FLY CALIFORNIA

VAULT DOOR

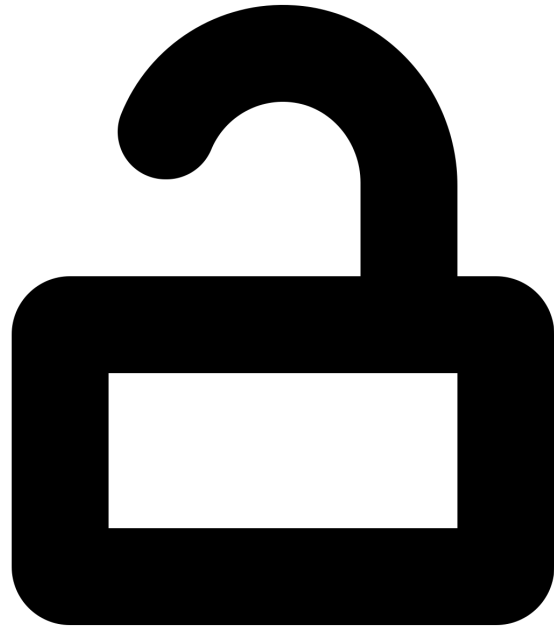
WEIGHT: 22 1/2 Tons
THICKNESS: 22 Inches
STEEL: 11 Layers of Special
Cutting and Drill Resistant
LOCKS: 4 Hamilton Watch
Movements for Time Locks







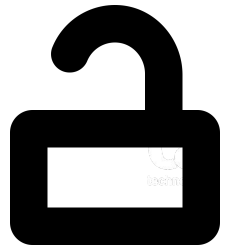
Balancing Speed and Safety at Scale





Unlocking Business Value

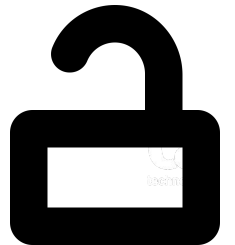
Where is everything?



Where is everything?



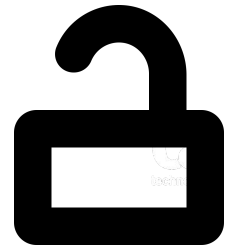
By Pascal from Heidelberg, Germany - The Mess, CC BY 2.0, <https://commons.wikimedia.org/w/index.php?curid=37981790>



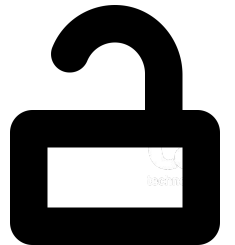
Where is everything?

"Data and services are stuck inside isolated applications within the enterprise."

-- Tung and Biltz, Accenture



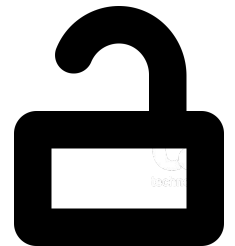
Why does it cost so much to get at it?



Why does it cost so much to get at it?



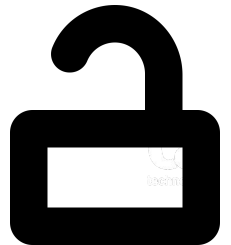
By DOJ - US Department of Justice photo, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=6419733>



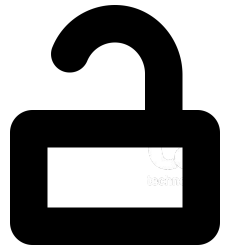
Why does it cost so much to get at it?

"It is about renovating at the core, as opposed to getting rid of the core."

-- Hung LeHong, Gartner



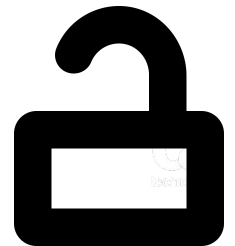
How can I reduce cost/risk?



How can I reduce cost/risk?

		COMPLEXITY				
		C1	C2	C3	C4	C5
SIZE	S1	100	250	400	550	700
	S2	175	325	475	625	775
	S3	250	400	550	700	850
	S4	325	475	625	775	625
	S5	400	550	700	850	1000

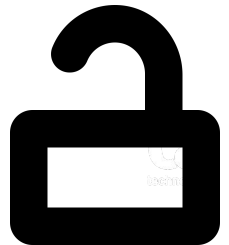
<https://www.infoq.com/articles/standish-chaos-2015>



How can I reduce cost/risk?

"Lower the risk of change through tools and culture."

-- John Allspaw, Etsy





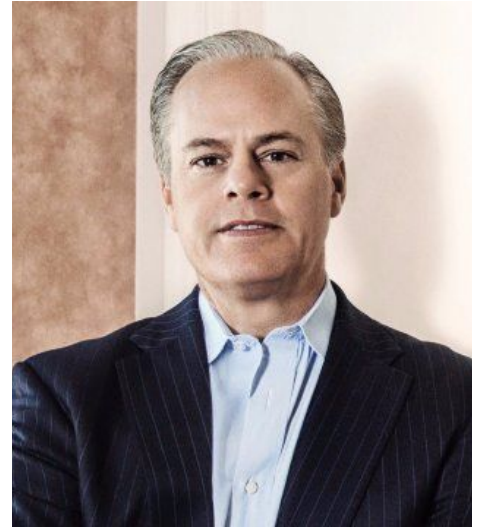
Changes Ahead





"Built for Change."

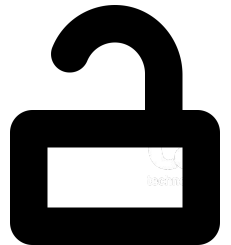
-- Mike Gregoire, CEO, CA Technologies



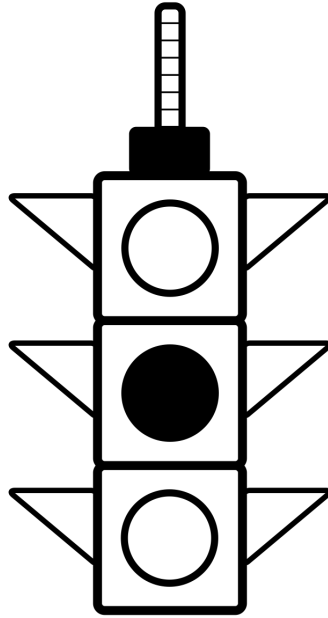
How can we do it?

Give your system the STAR treatment

- **Stabilize**
- **Transform**
- **Add**
- **Repeat**



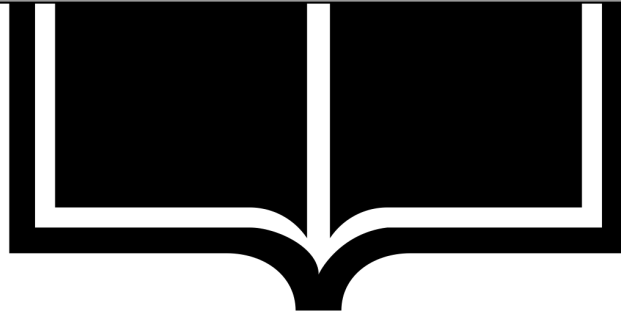
But first...







Basic Principles



The elephant in the room: one bite at a time.



The elephant in the room: one bite at a time.



By Bit Boy - Flickr: The Elephant in the Room, CC BY 2.0, <https://commons.wikimedia.org/w/index.php?curid=20972528>



The elephant in the room: one bite at a time.

"Whenever you do a transition, do the smallest thing that teaches you the most and do that over and over again."

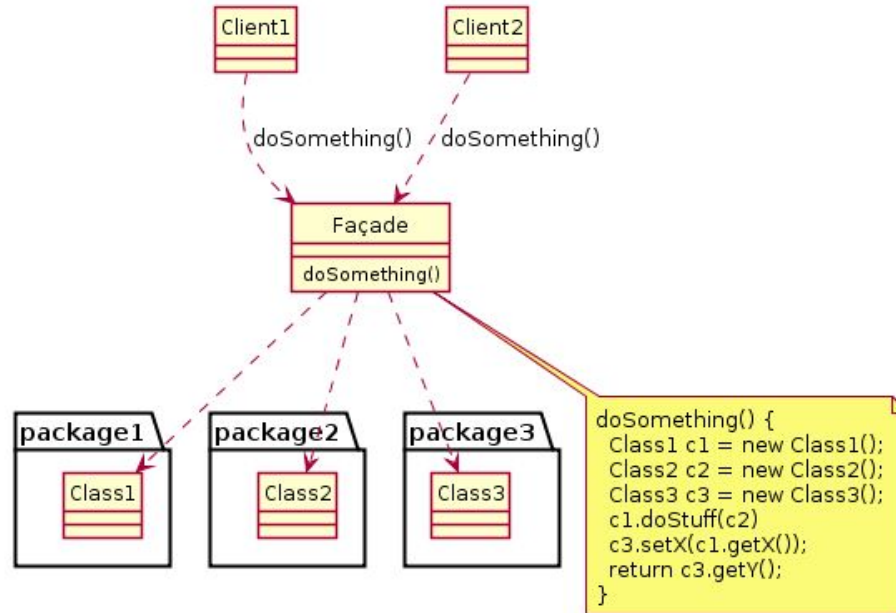
-- Adrian Cockcroft, Netflix



Employ facades, stranglers, and refactoring



Employ facades, stranglers, and refactoring



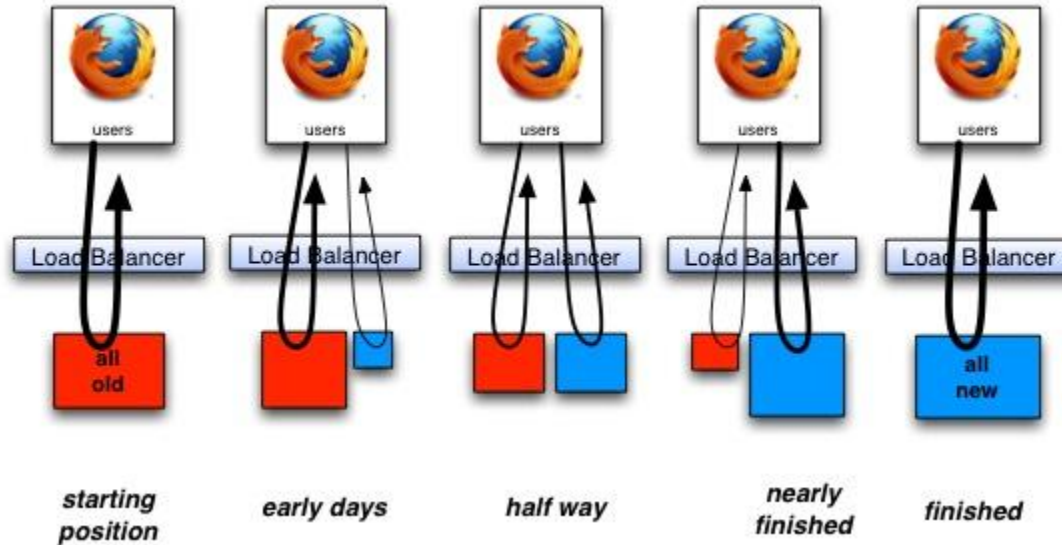
Employ facades, stranglers, and refactoring

*"The **facade** design pattern is used to define a simplified interface to a more complex subsystem."*

-- Richard Carr, BlackWasp



Employ facades, stranglers, and refactoring



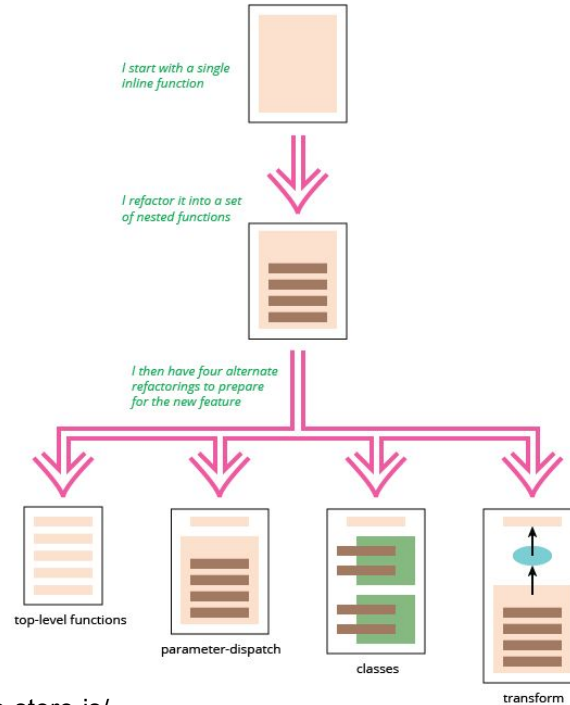
Employ facades, stranglers, and refactoring

"Strangulation of a legacy solution is a safe way to phase one thing out for something better."

-- Paul Hammant, Thoughtworks



Employ facades, stranglers, and refactoring



<https://martinfowler.com/articles/refactoring-video-store-js/>



Employ facades, stranglers, and refactoring

*"When you **refactor** you are improving the design of the code after it has been written."*

-- Martin Fowler, Thoughtworks



APIs are forever, code is not.



APIs are forever, code is not.

Not Found

The requested URL /oldpage.html was not found on this server.

Apache/2.2.3 (CentOS) Server at www.example.com Port 80



APIs are forever, code is not.

"We knew that designing APIs was a very important task as we'd only have one chance to get it right."

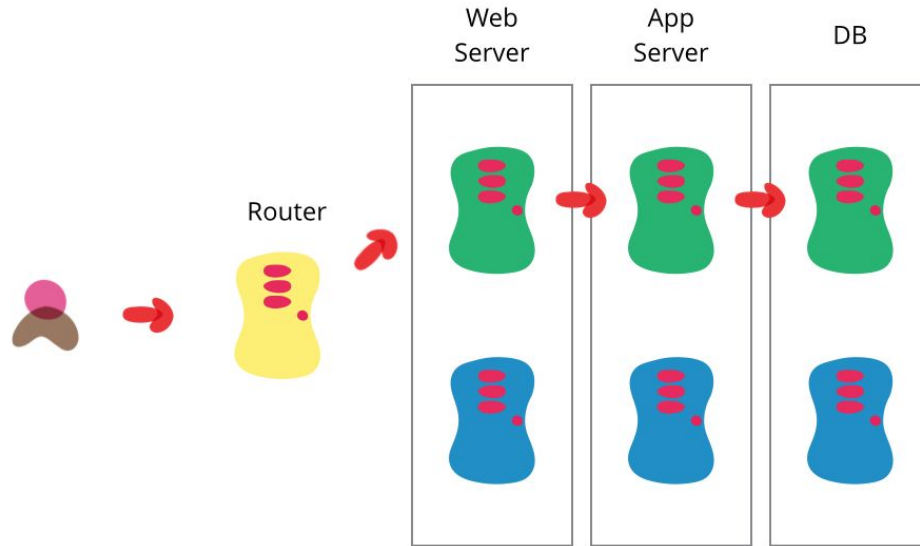
-- Werner Vogels, Amazon



Continuous change and instant reversibility



Continuous change and instant reversibility



<https://martinfowler.com/bliki/BlueGreenDeployment.html>



Continuous change and instant reversibility

"Blue-green deployment gives you a rapid way to rollback - if anything goes wrong."

-- Martin Fowler, Thoughtworks

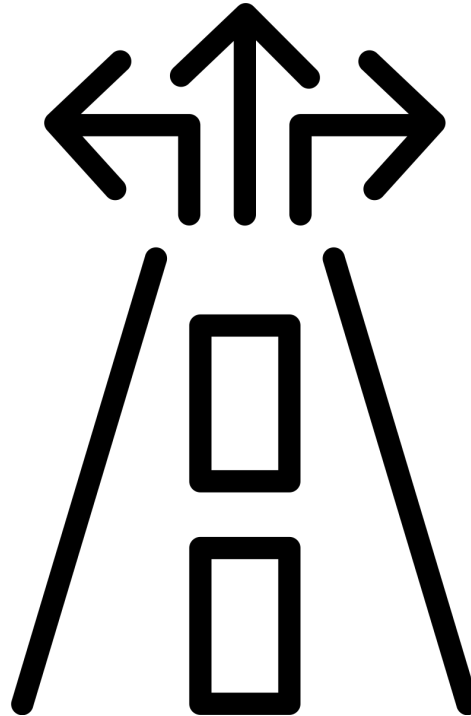


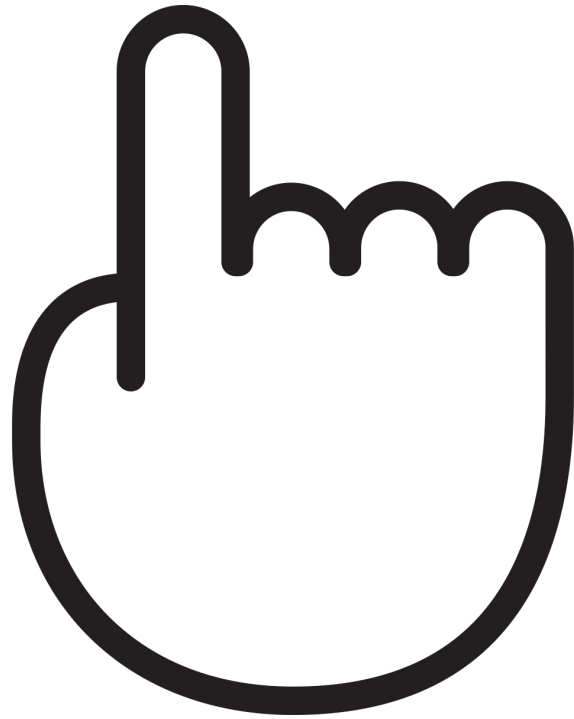
Basic Principles

- Take one bite at a time.
- Employ facades, stranglers, and refactoring
- APIs are forever, code is not
- Continuous change and instant reversibility



So, what's the roadmap?



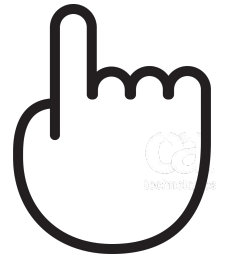




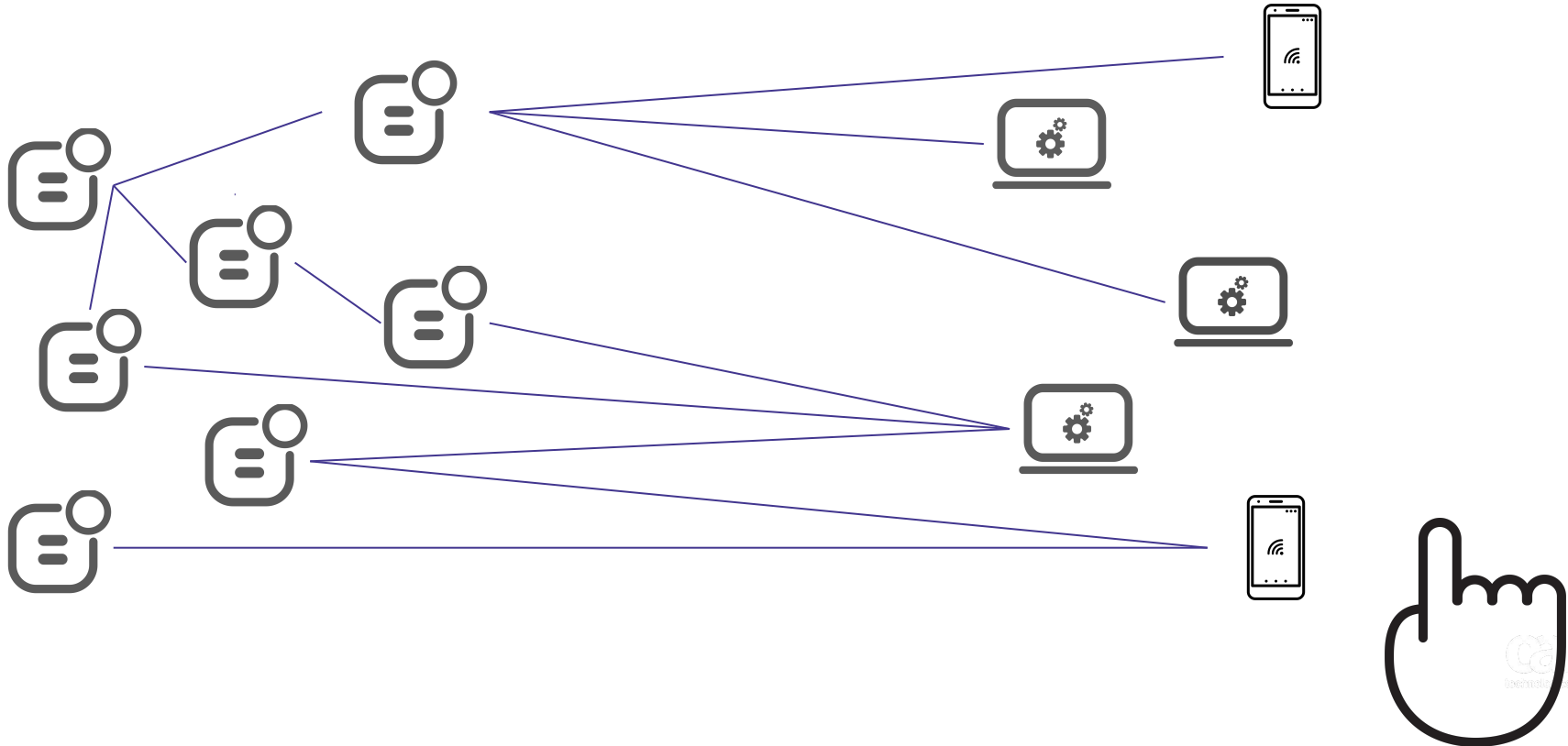
Step 1: Stabilize the Interface

Step 1: Stabilize the Interface

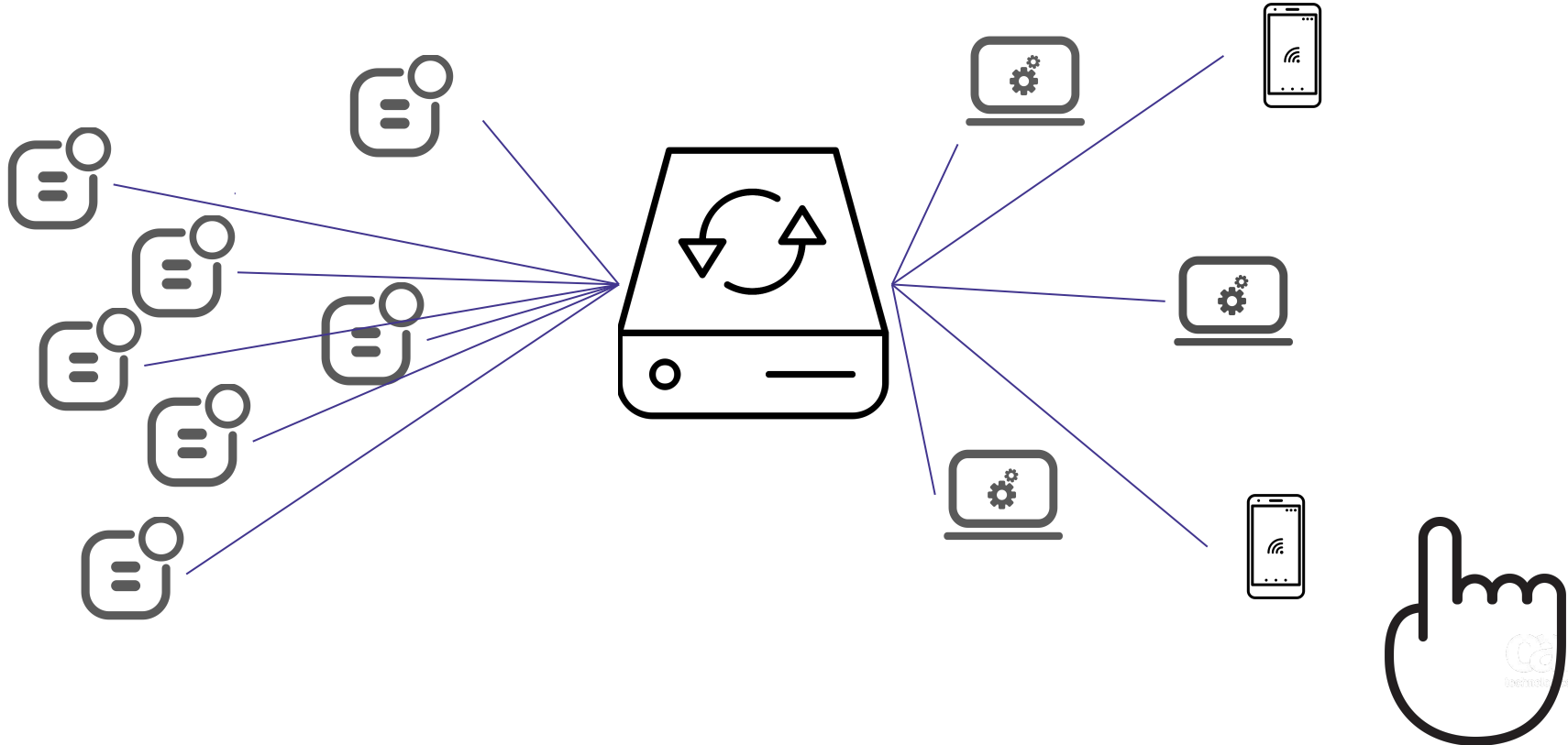
All API consumers talk to a proxy



All API consumers talk to a proxy

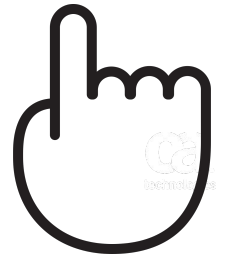


All API consumers talk to a proxy

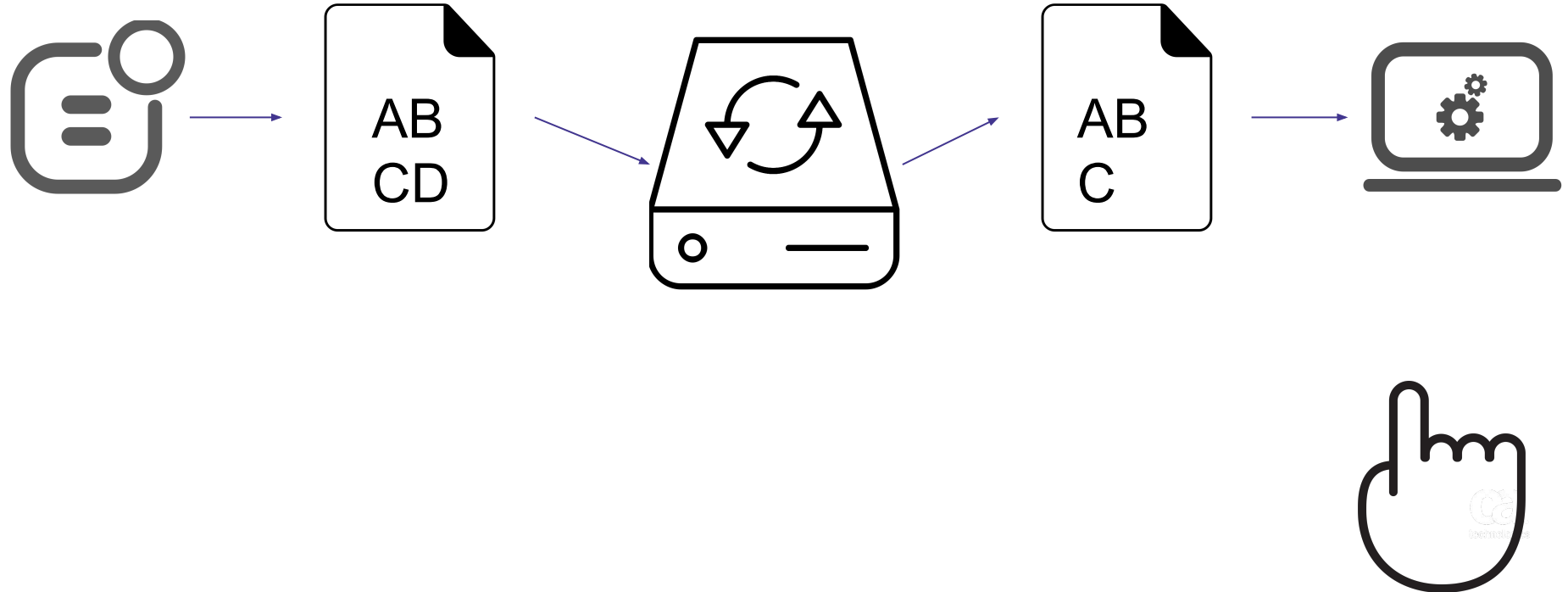


Step 1: Stabilize the Interface

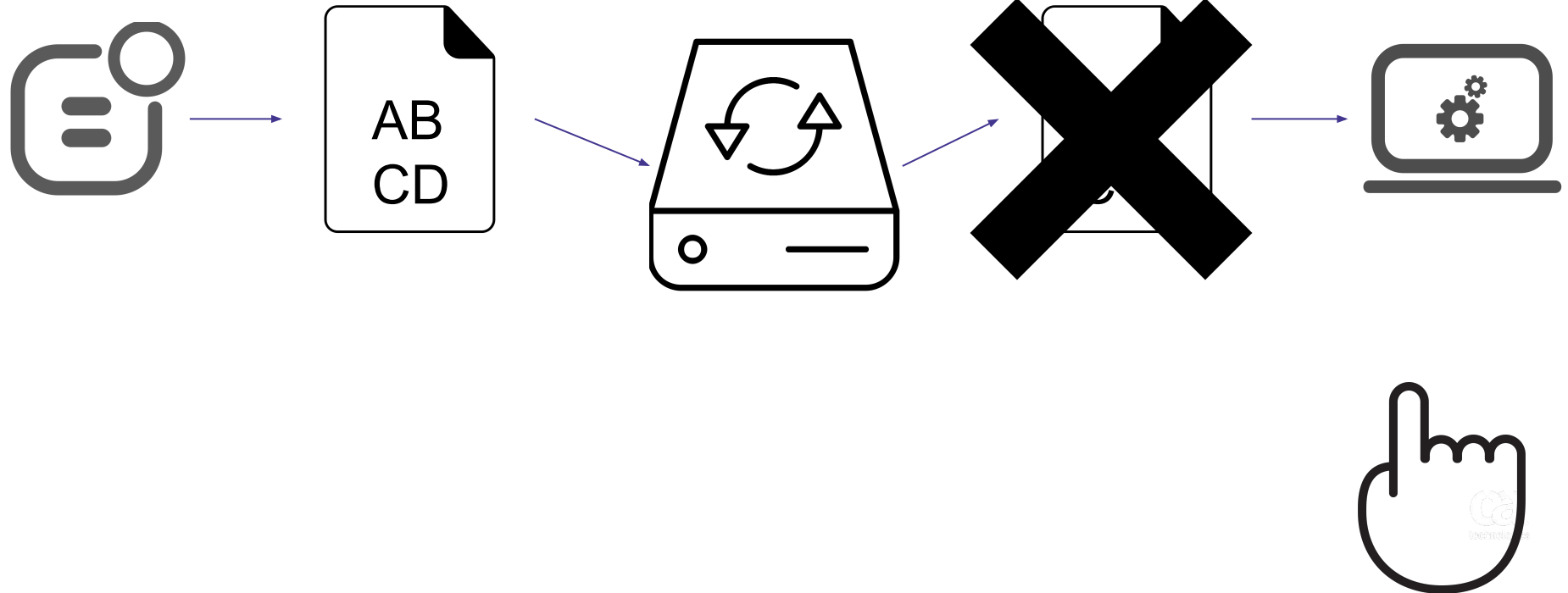
The stabilizer proxy is be pass-through only



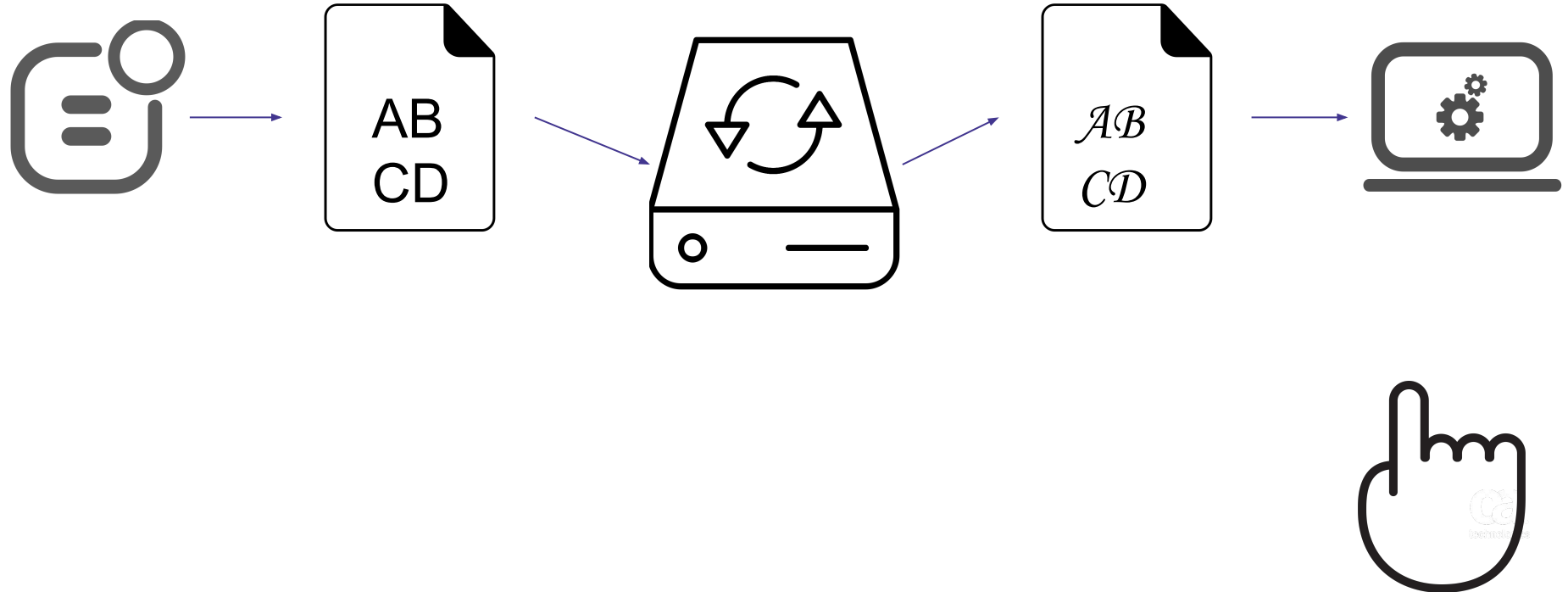
The stabilizer proxy is be pass-through only



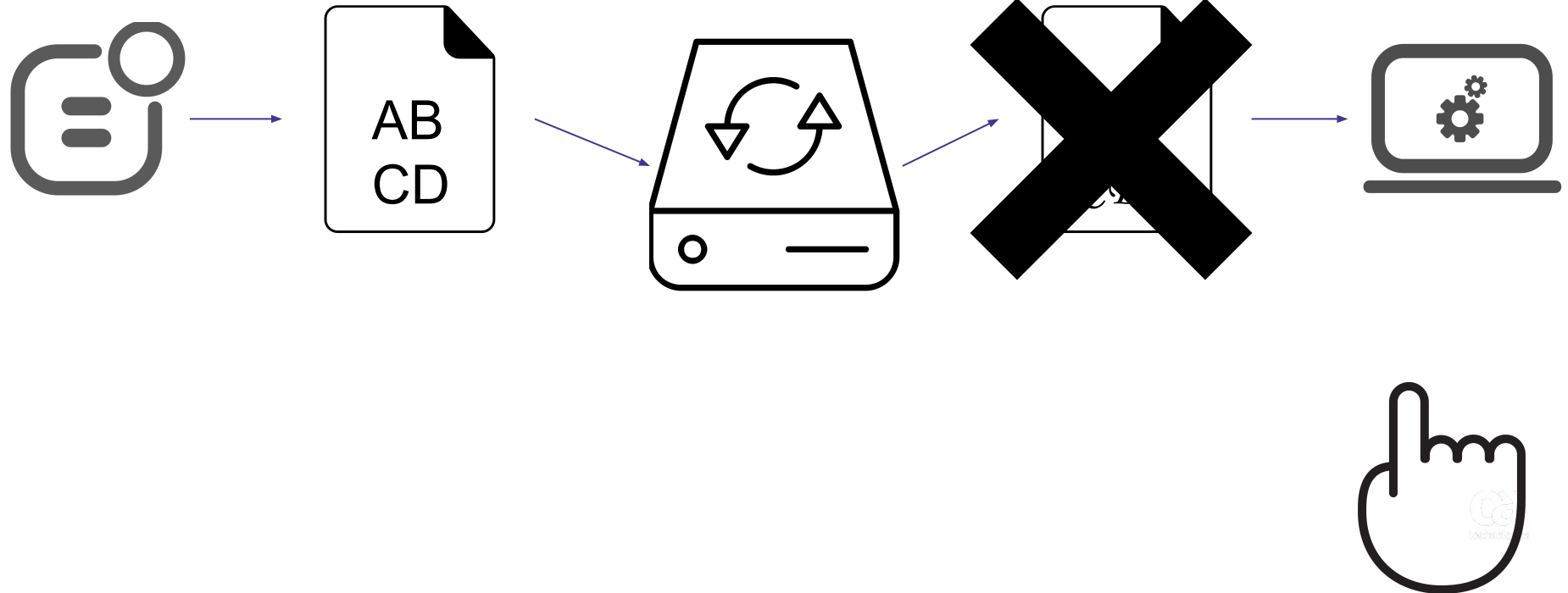
The stabilizer proxy is be pass-through only



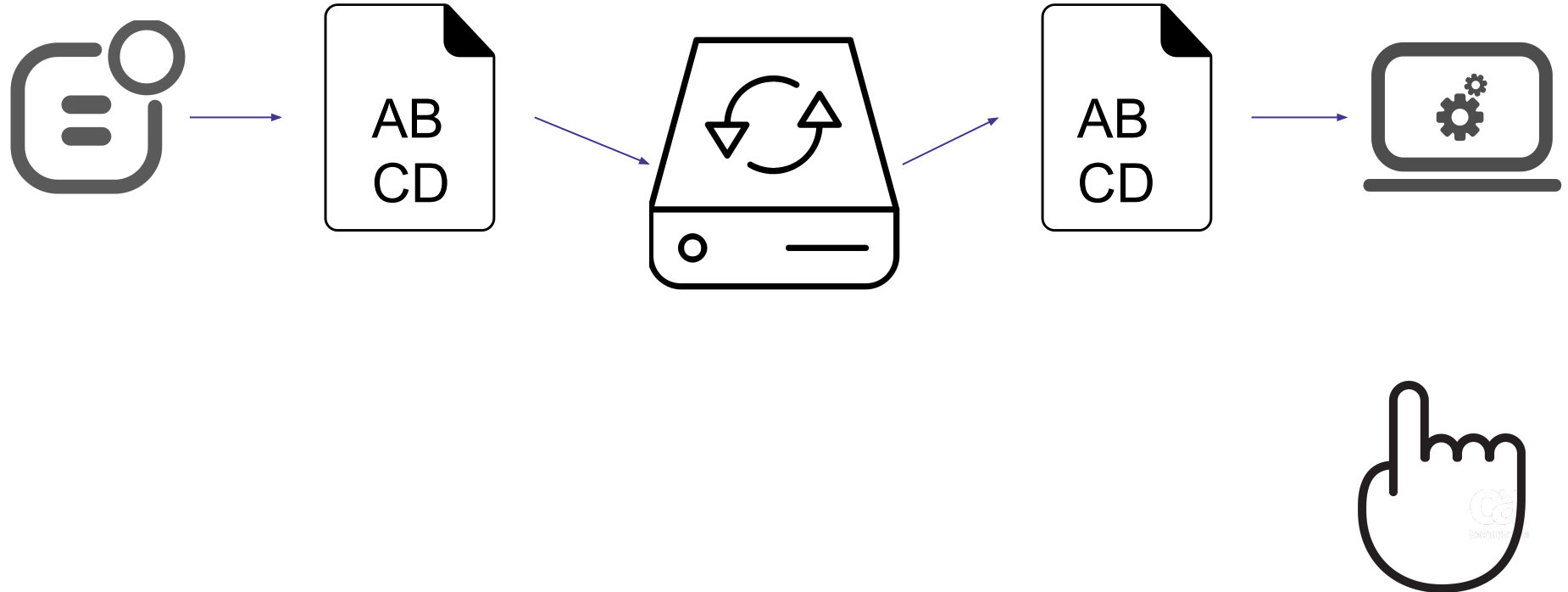
The stabilizer proxy is be pass-through only



The stabilizer proxy is be pass-through only

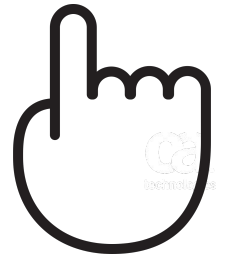


The stabilizer proxy is be pass-through only

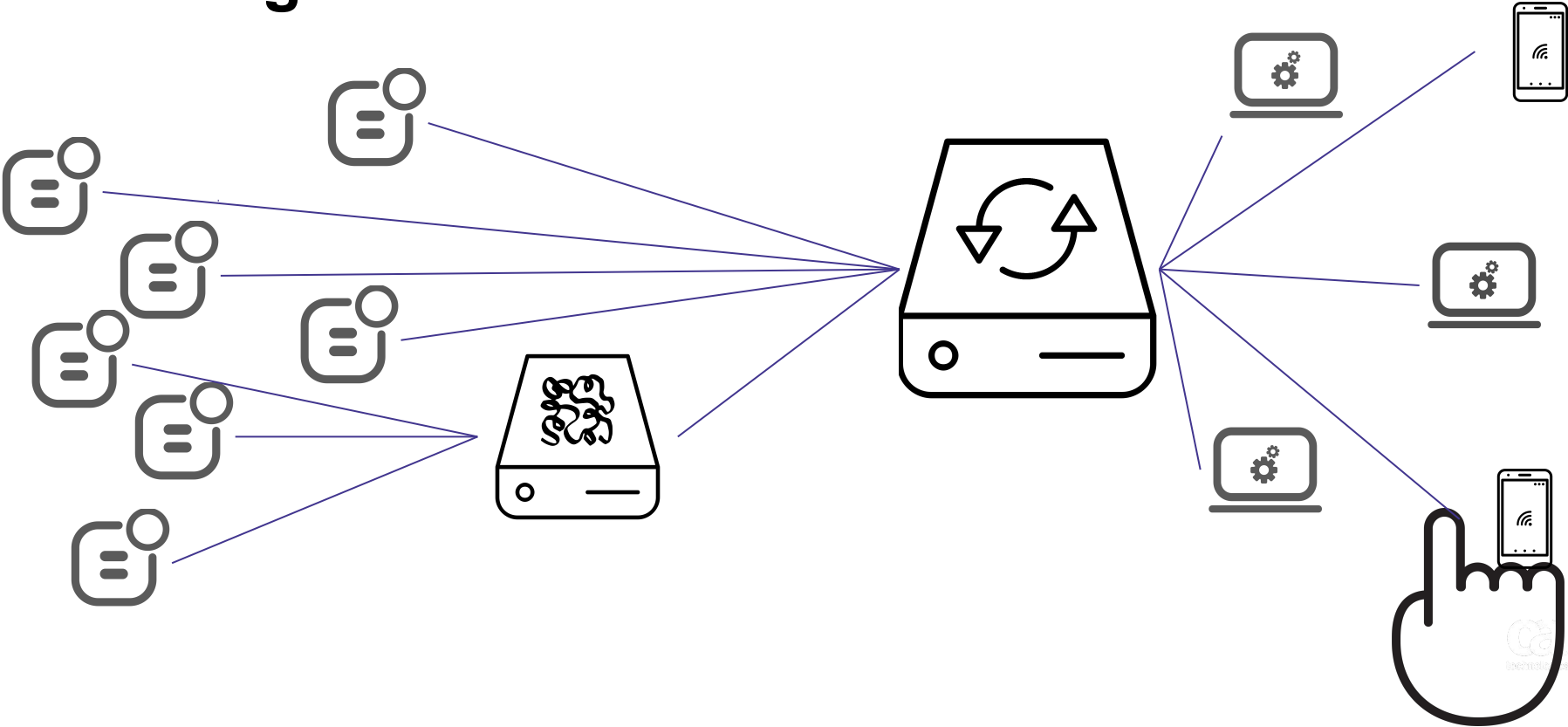


Step 1: Stabilize the Interface

ESBs, external services go behind the Stabilizer

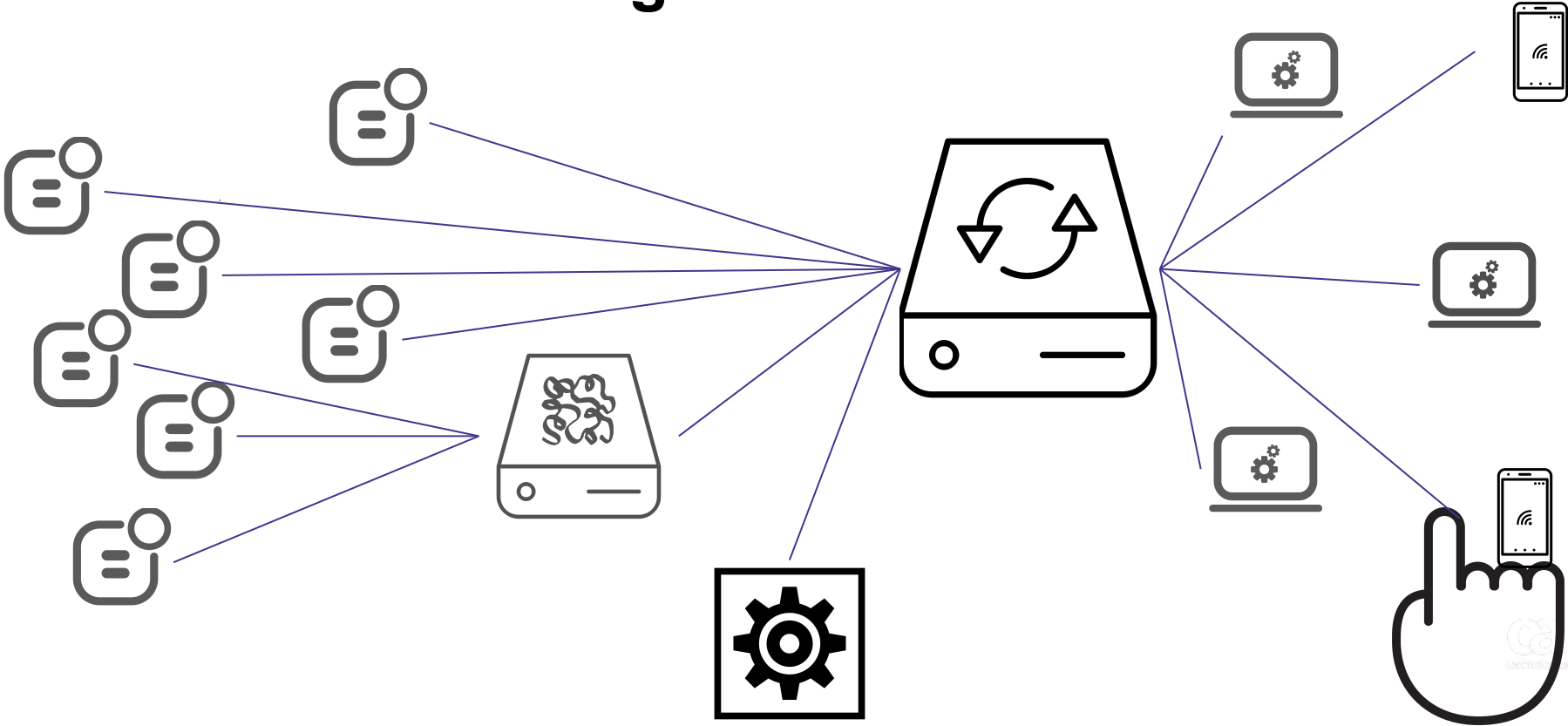


ESBs go behind the Stabilizer



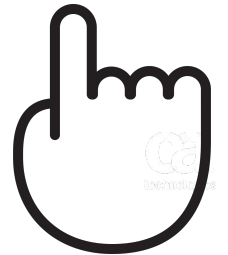
Step 1: Stabilize the Interface

External services go behind the Stabilizer

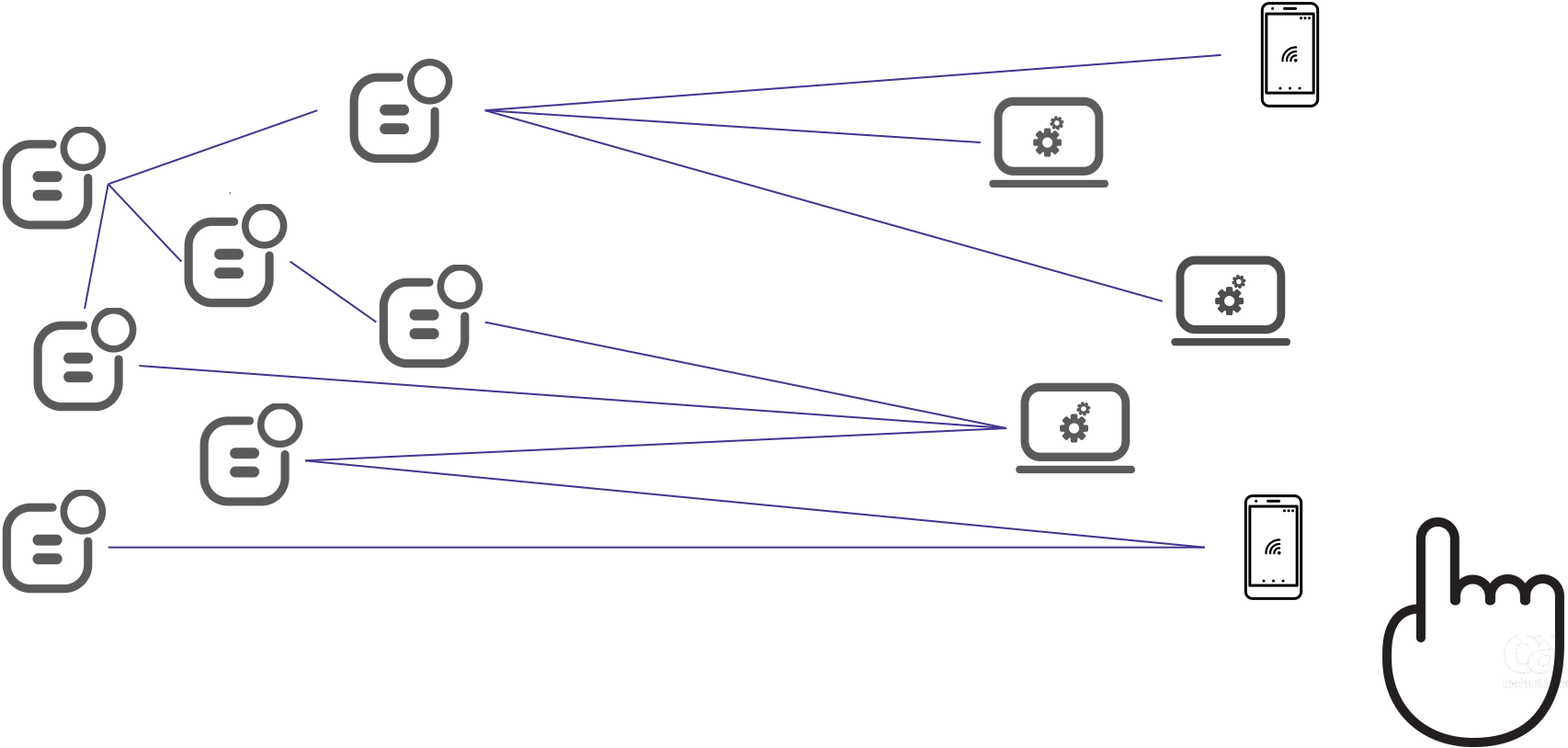


Step 1: Stabilize the Interface

"Proxy Marching"

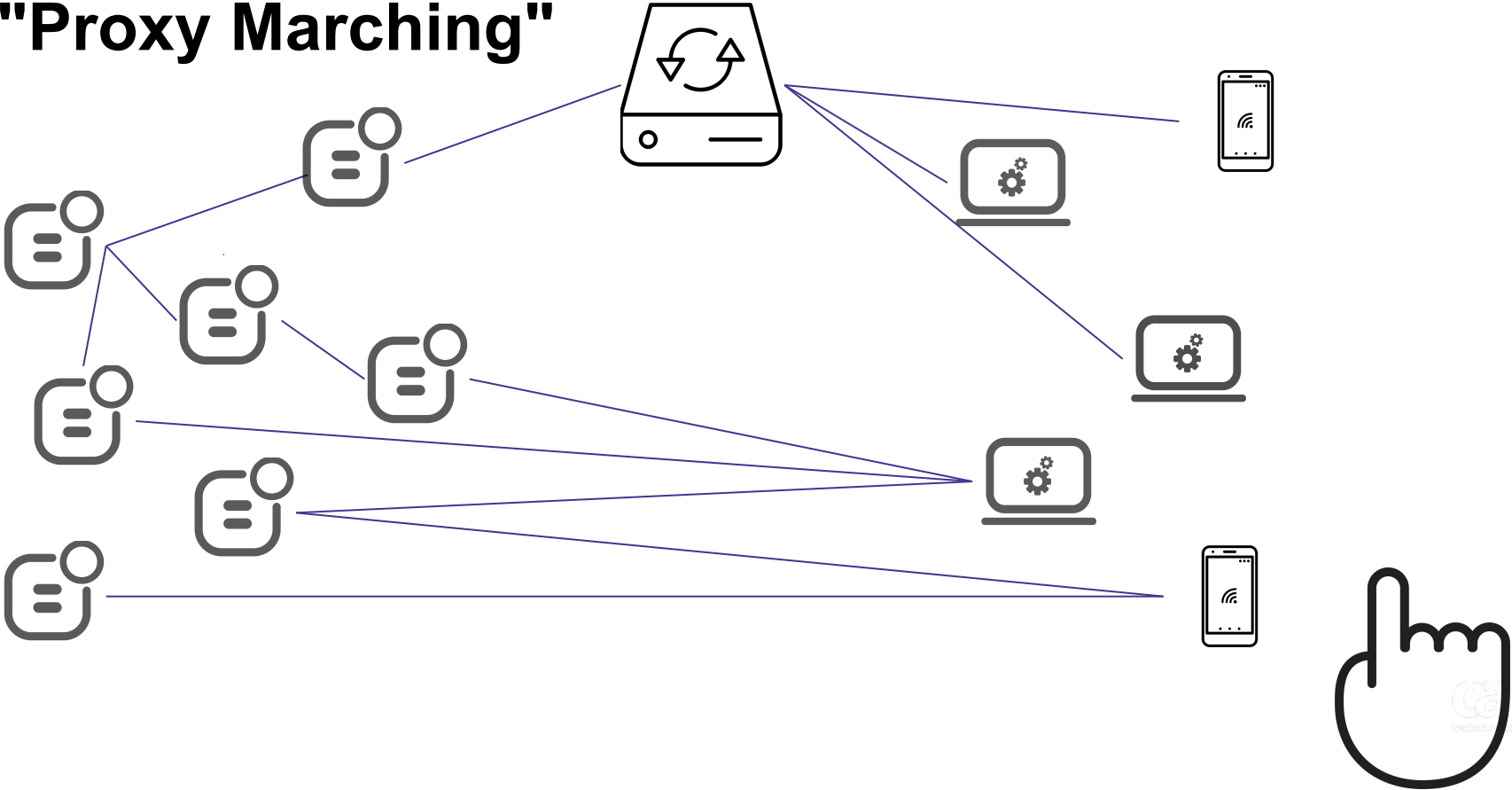


"Proxy Marching"



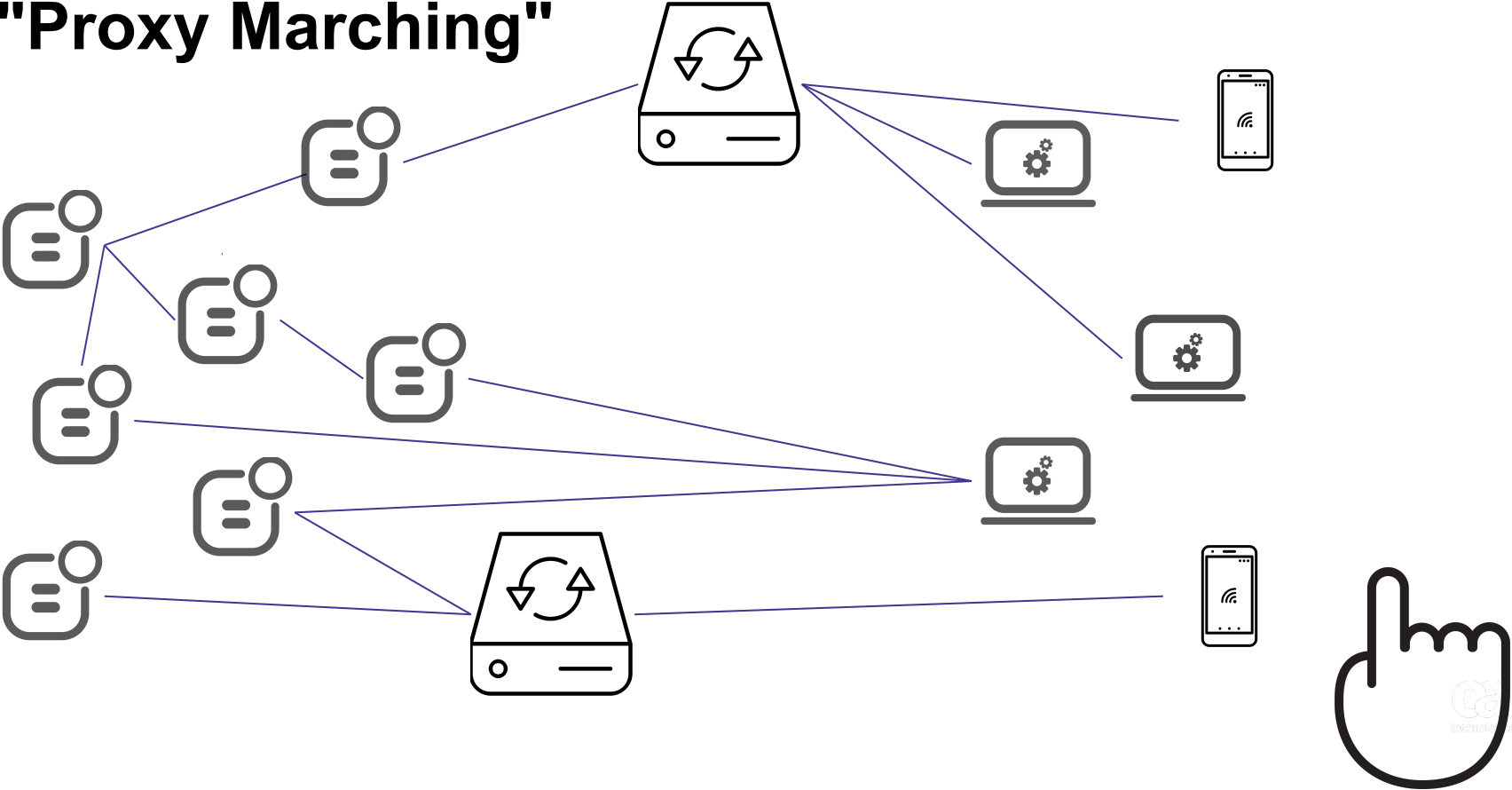
Step 1: Stabilize the Interface

"Proxy Marching"



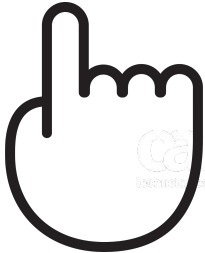
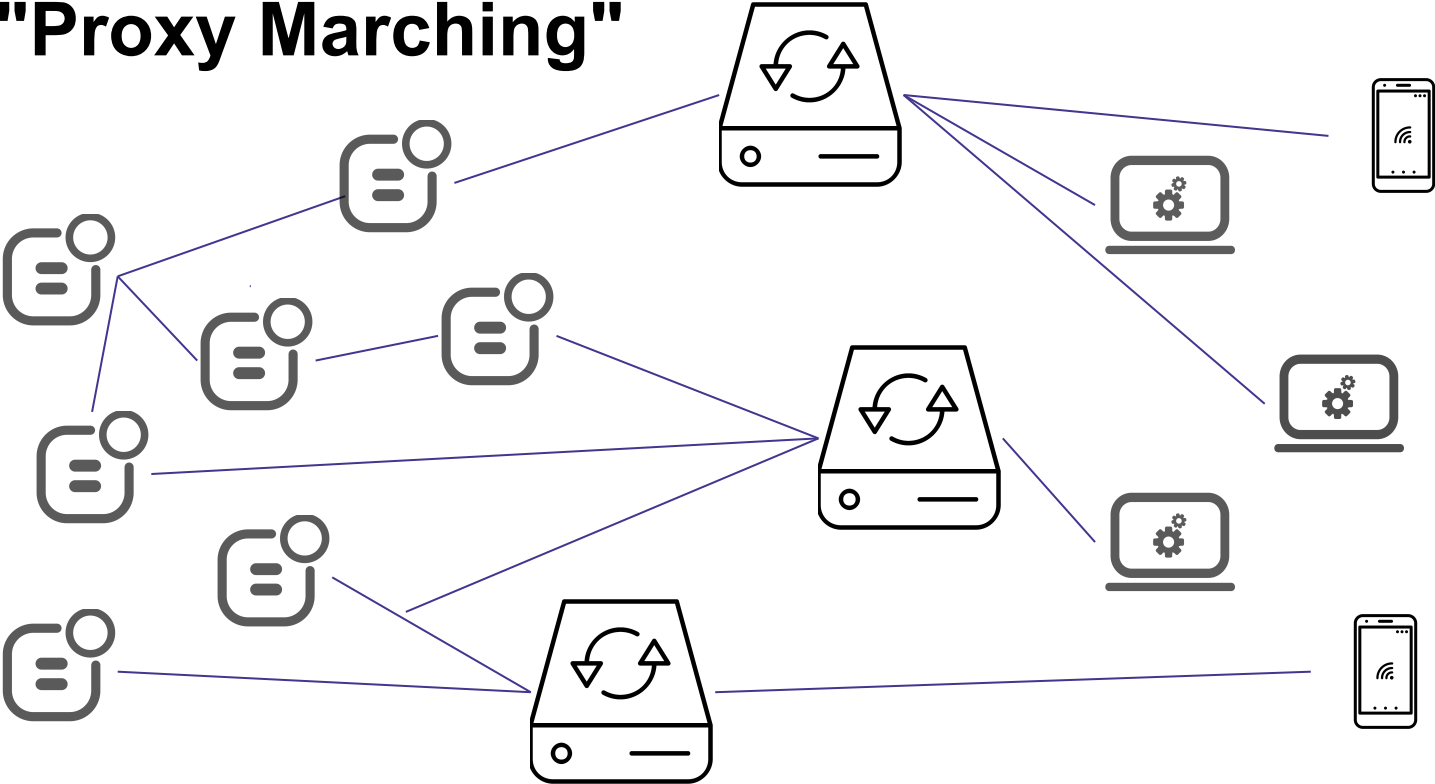
Step 1: Stabilize the Interface

"Proxy Marching"



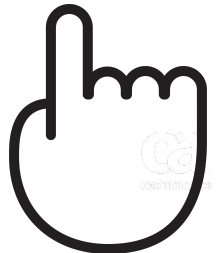
Step 1: Stabilize the Interface

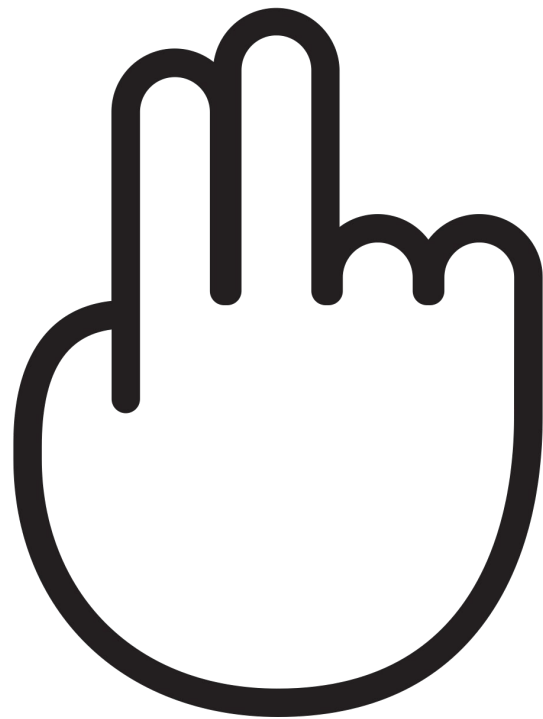
"Proxy Marching"



Stabilize the Interface

- All API consumers talk to a proxy
- The proxy **MUST** be pass-through only
- Keep ESBs & external services *behind* the proxy
- Employ a "Proxy March"



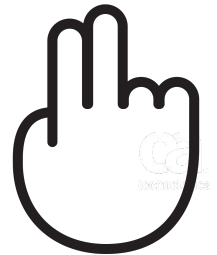




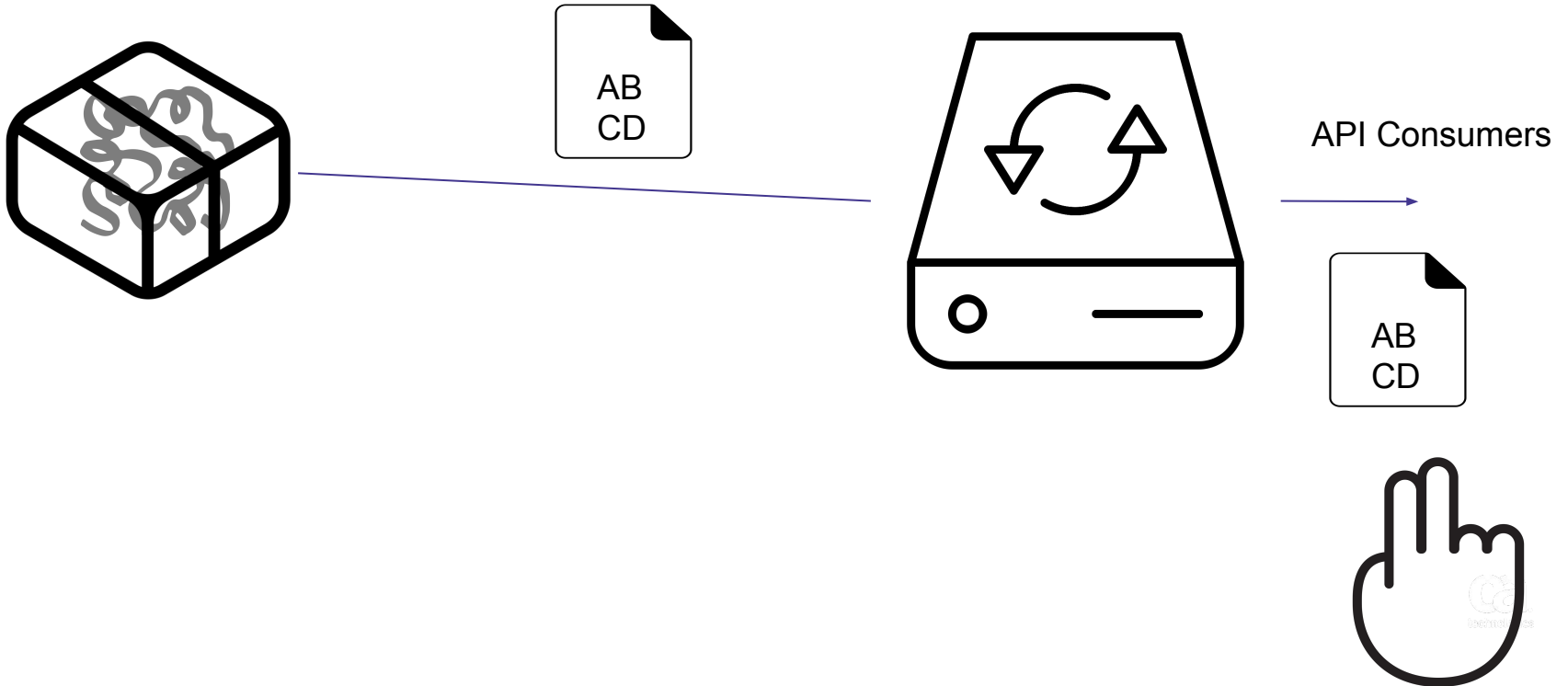
Step 2: Transform the Implementation

Step 2: Transform the Implementation

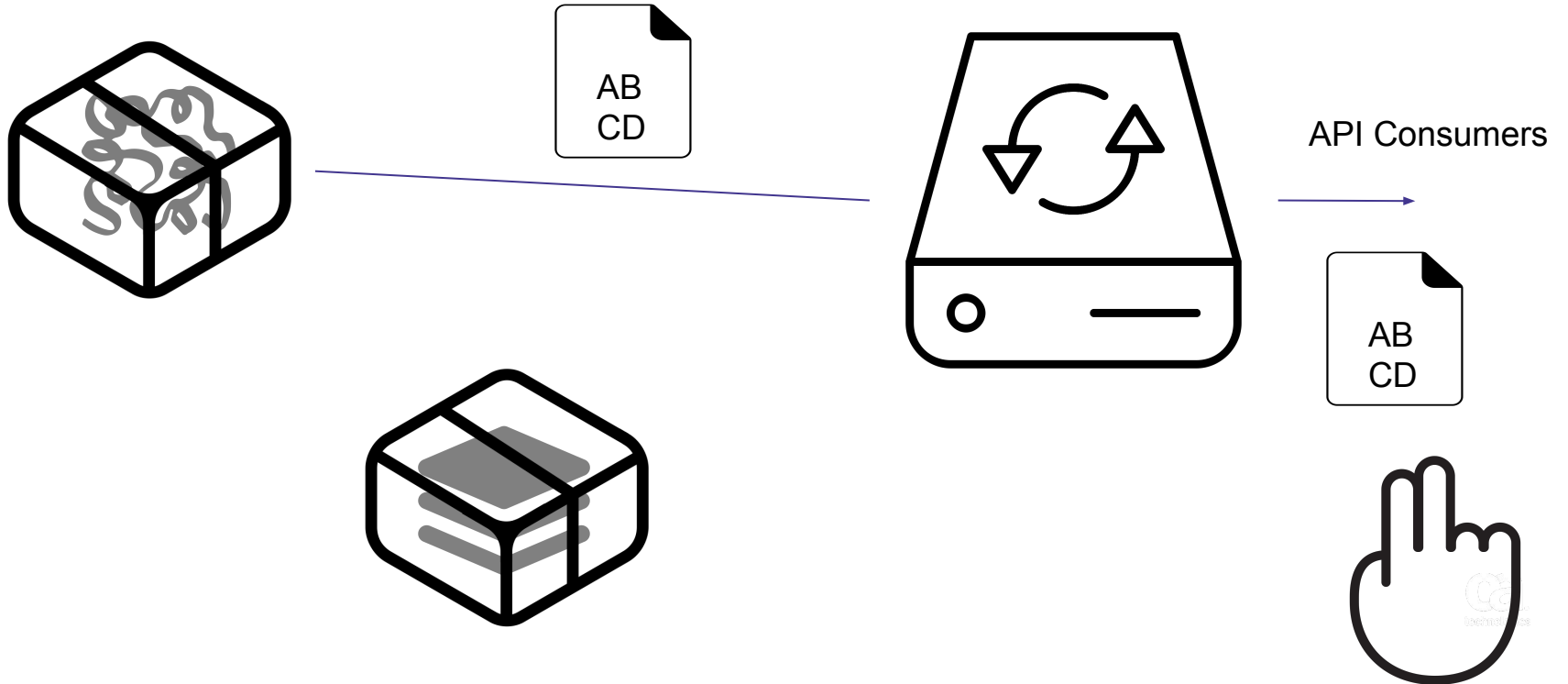
Refactor existing components



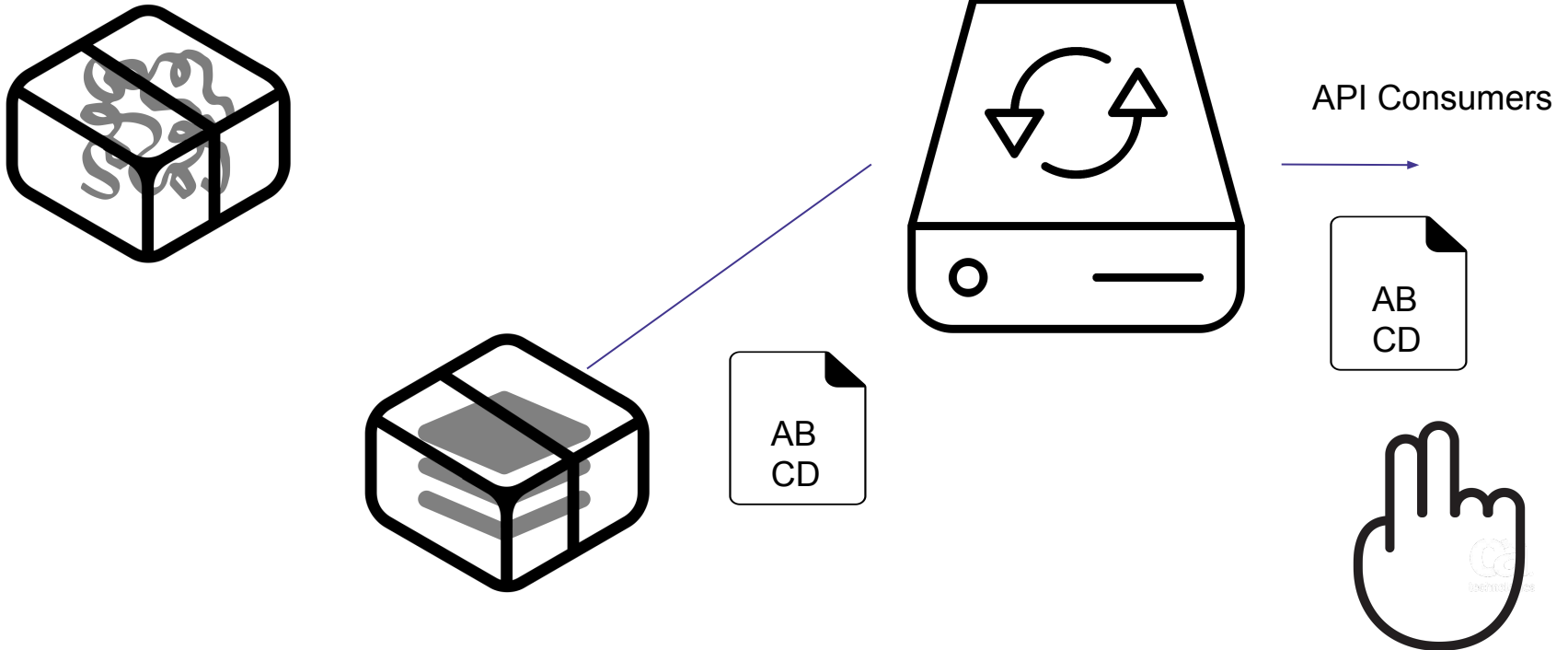
Refactor existing components



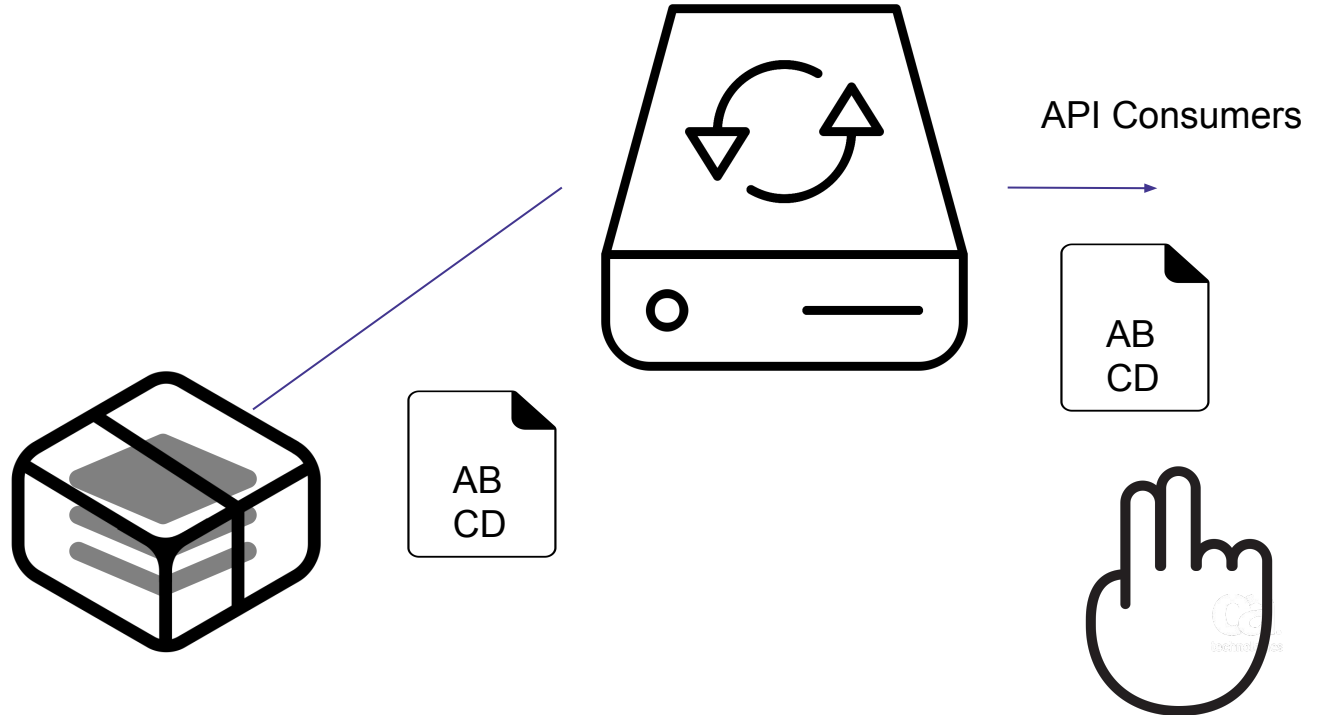
Refactor existing components



Refactor existing components

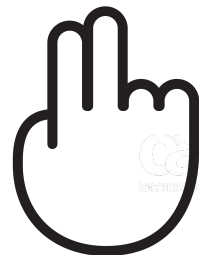


Refactor existing components



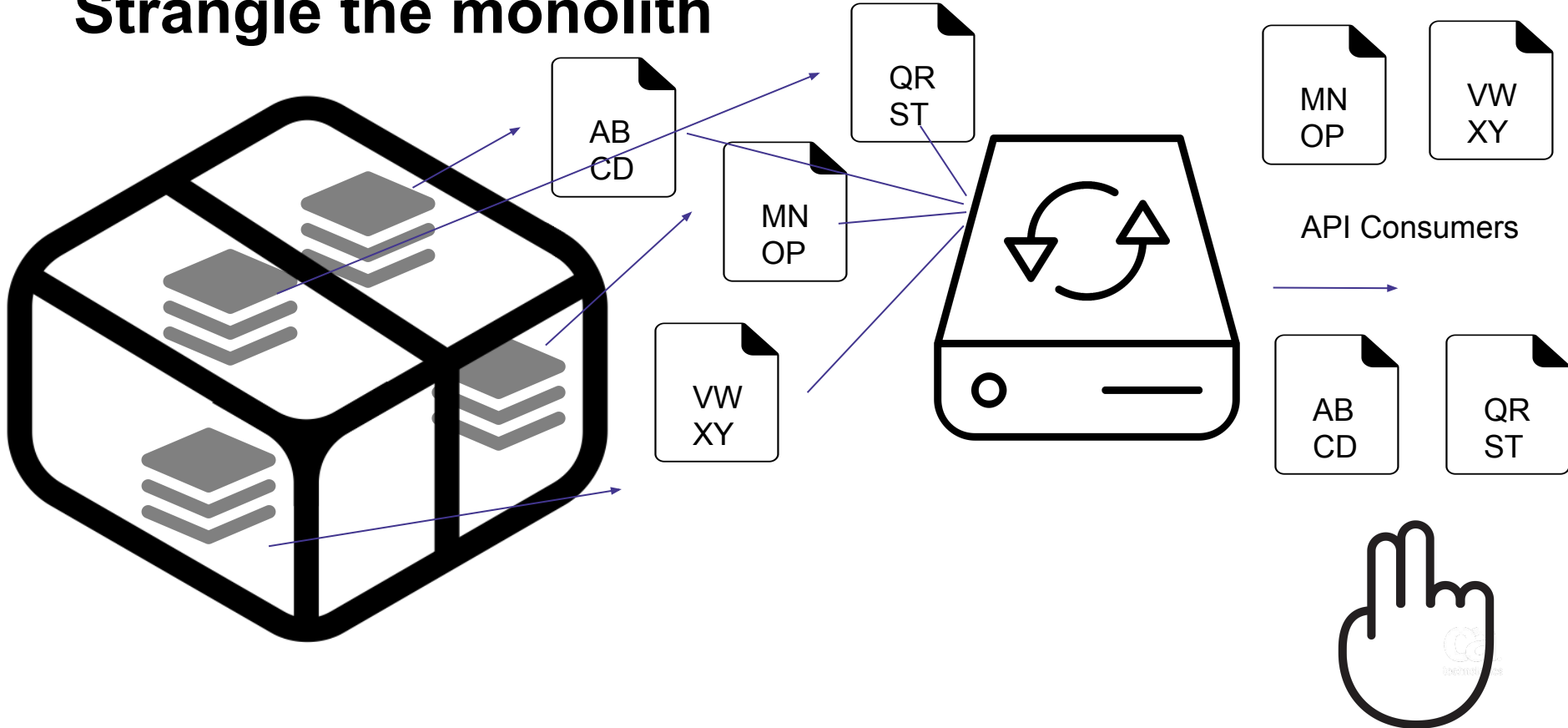
Step 2: Transform the Implementation

Strangle the monolith



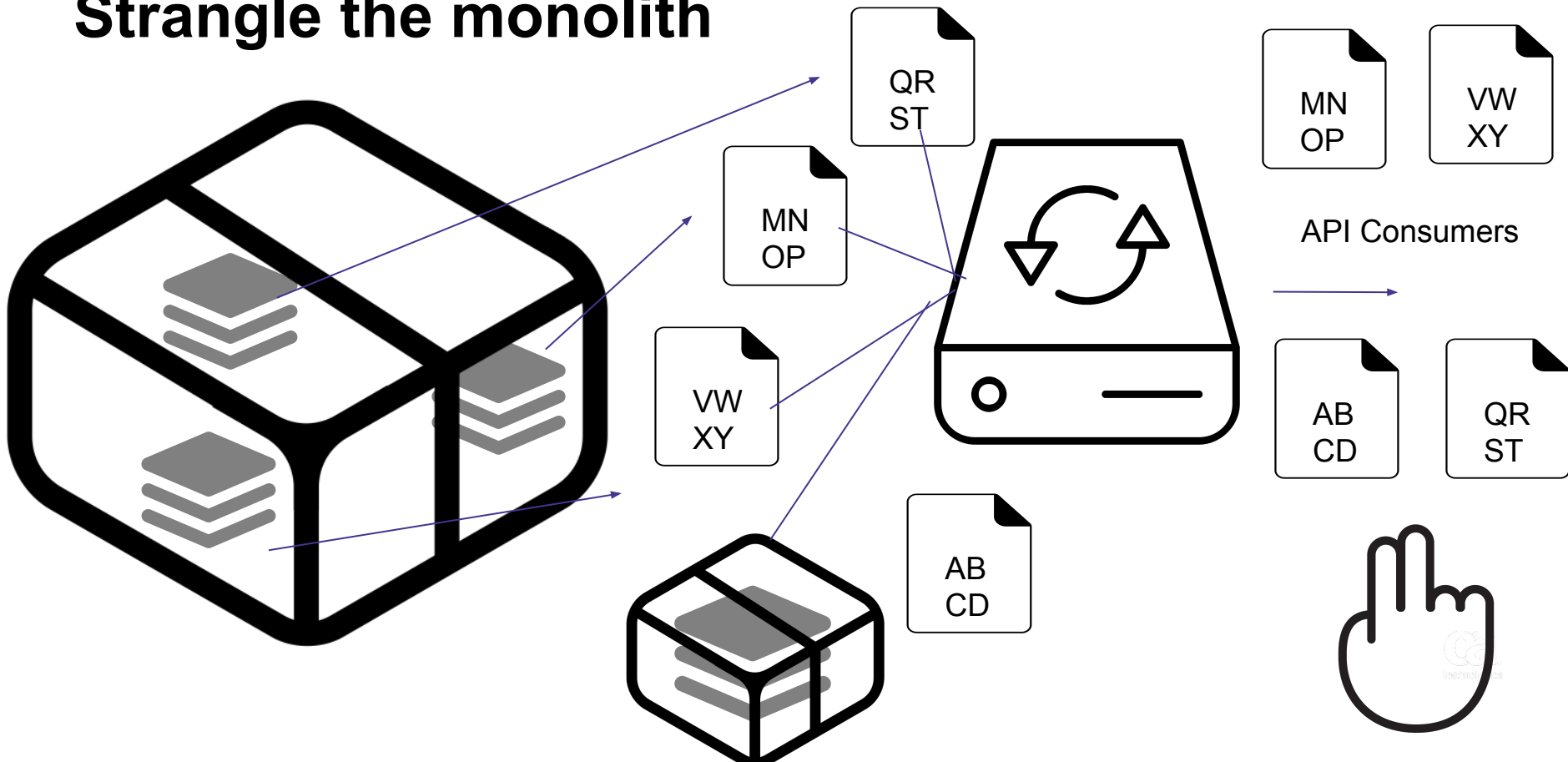
Step 2: Transform the Implementation

Strangle the monolith



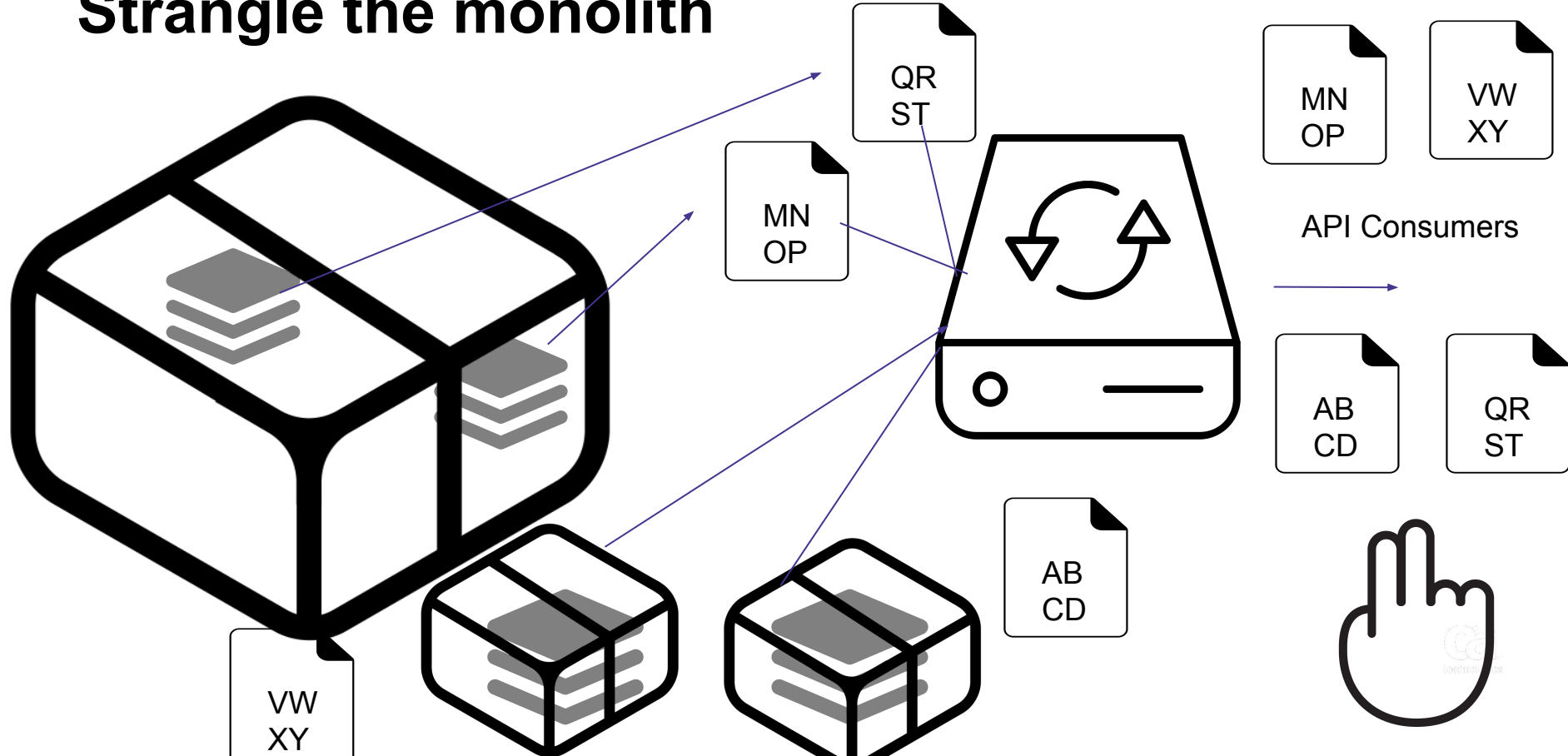
Step 2: Transform the Implementation

Strangle the monolith



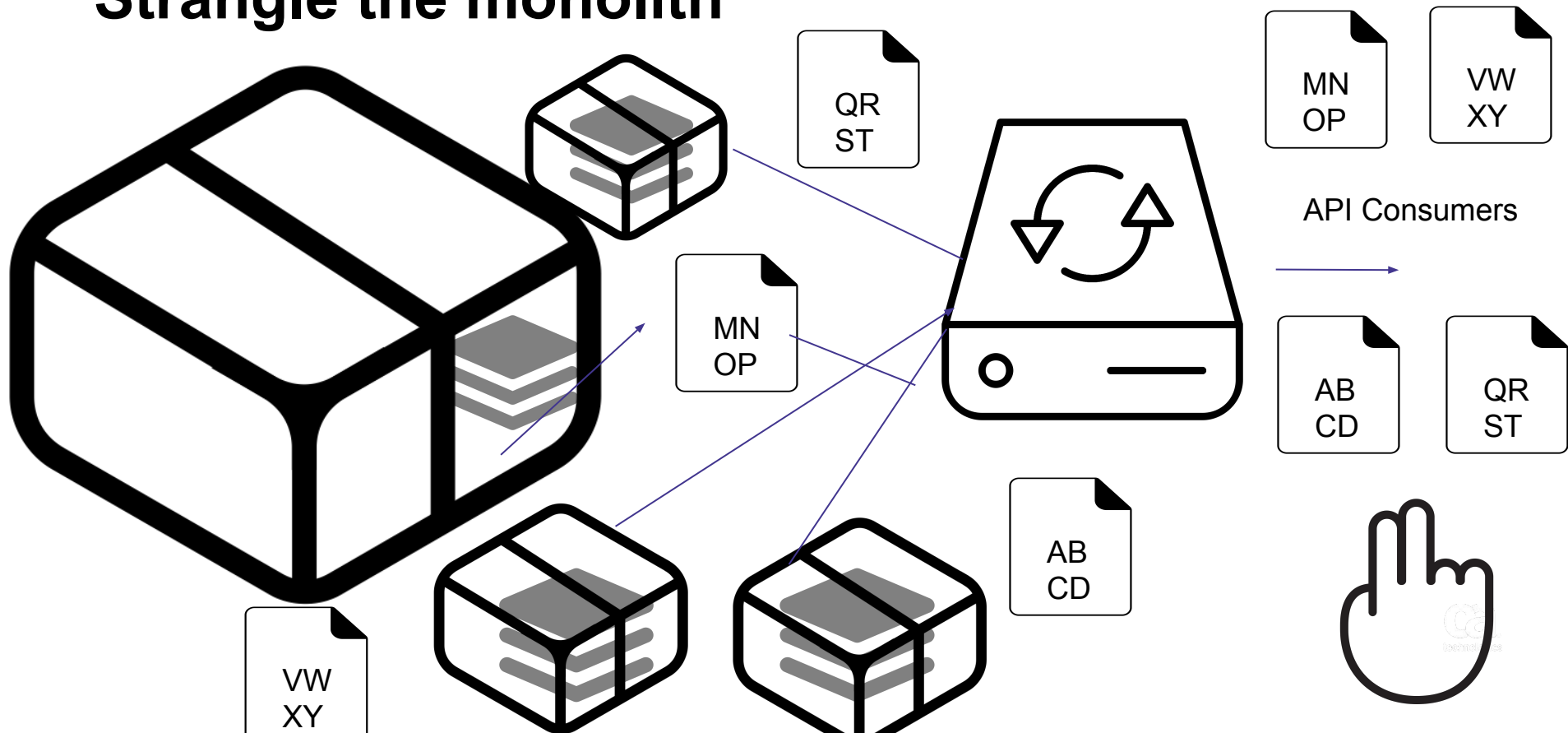
Step 2: Transform the Implementation

Strangle the monolith



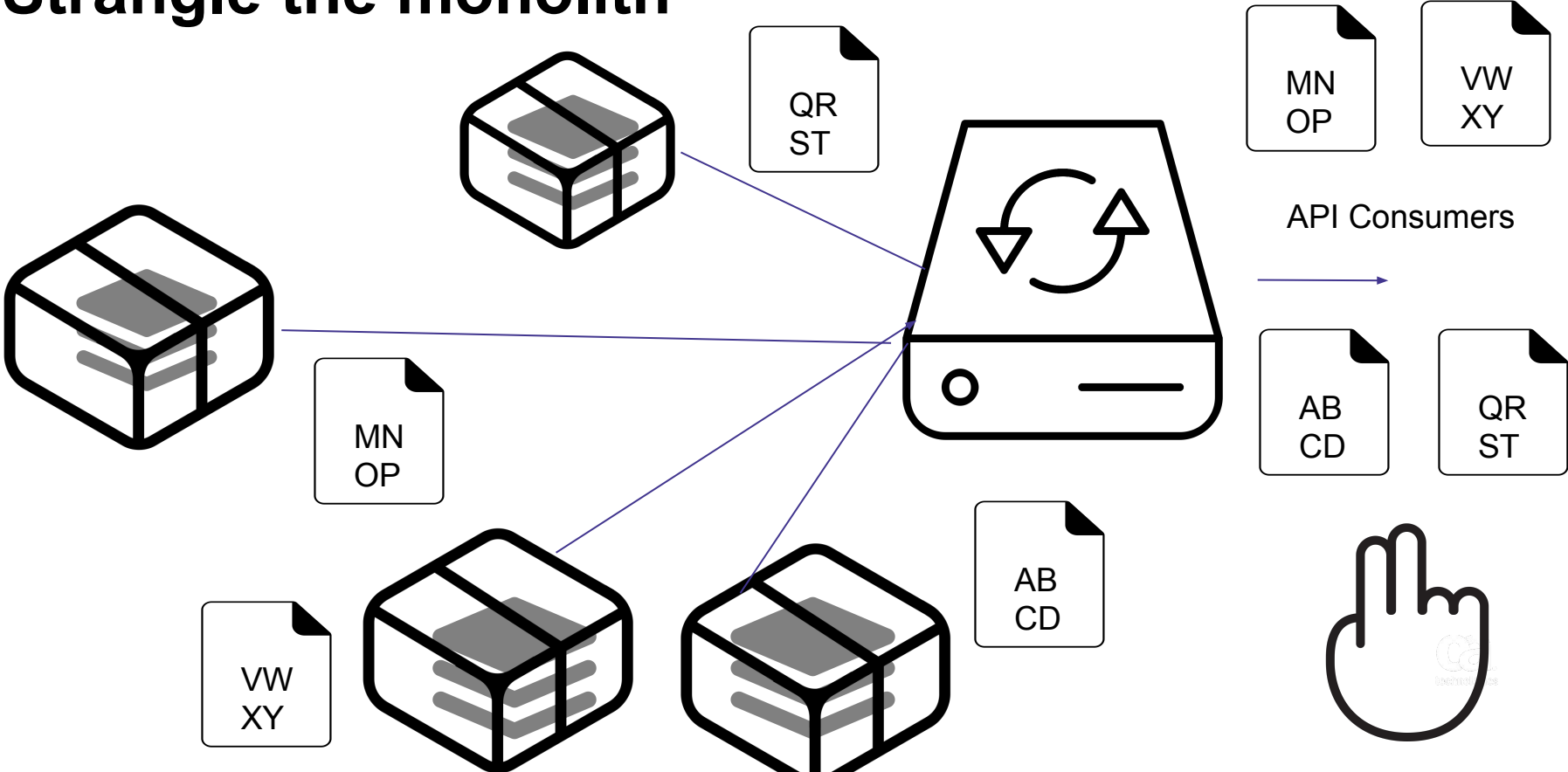
Step 2: Transform the Implementation

Strangle the monolith



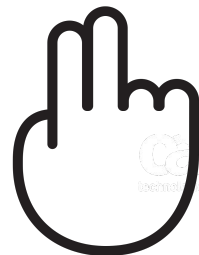
Step 2: Transform the Implementation

Strangle the monolith



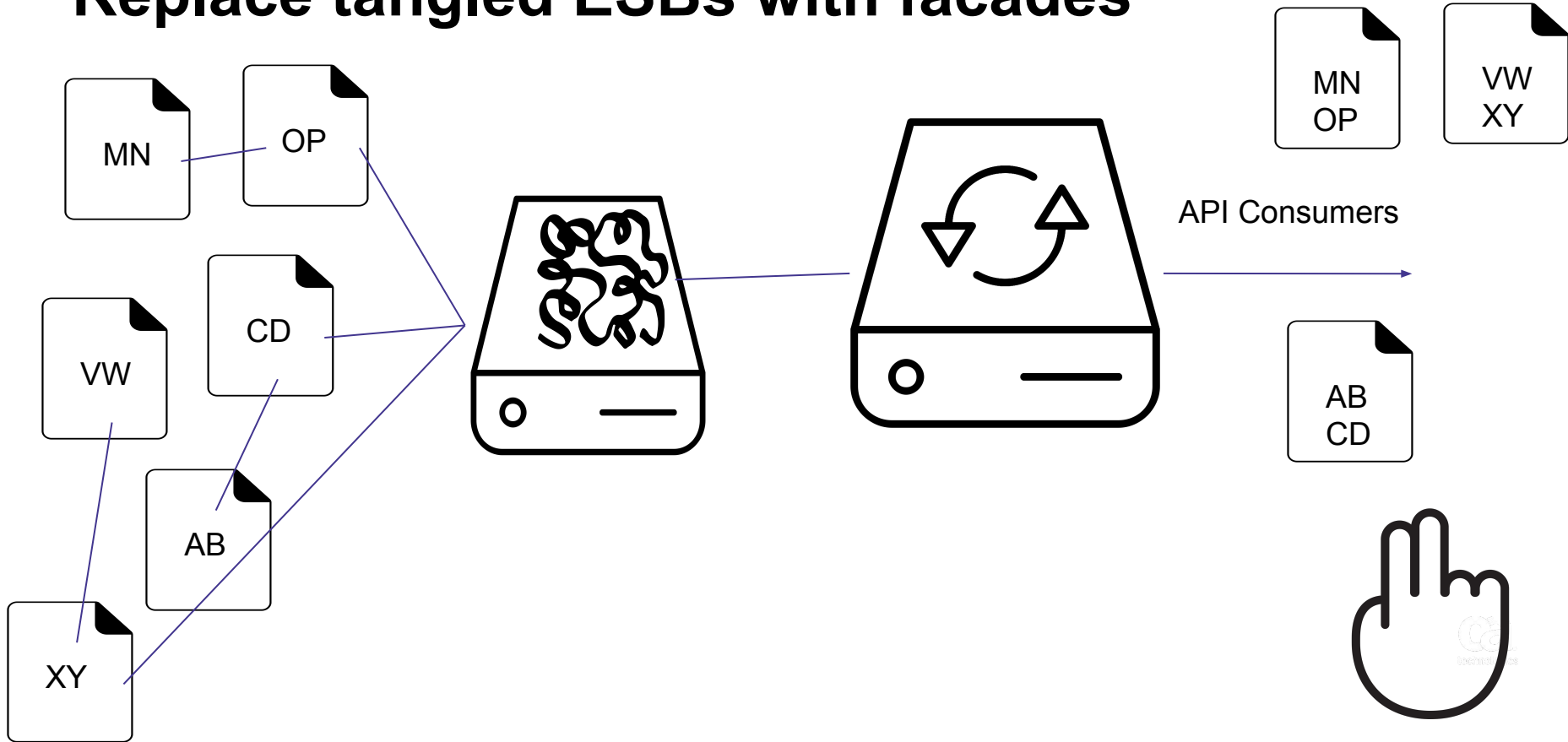
Step 2: Transform the Implementation

Replace tangled ESBs with facades



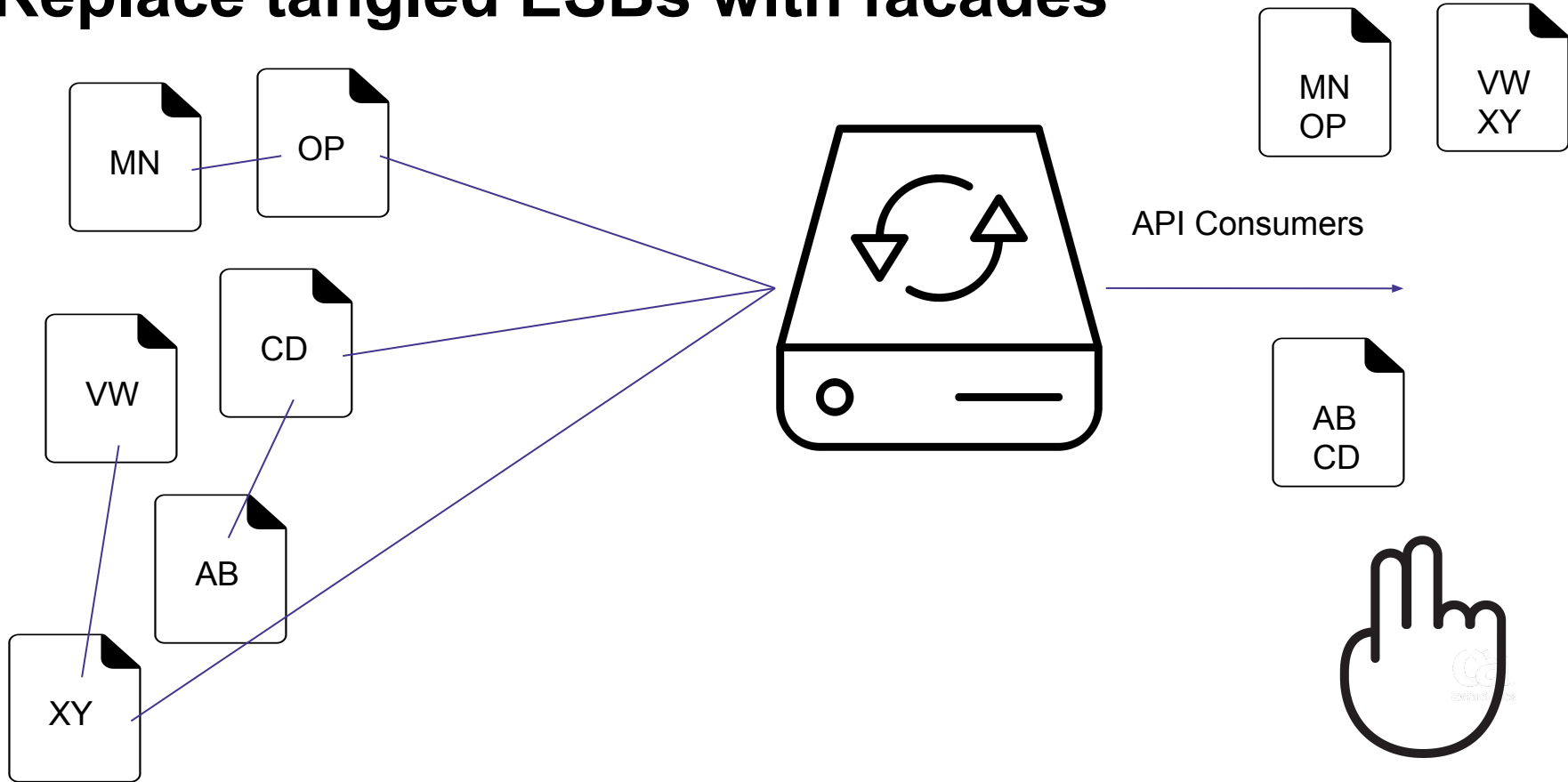
Step 2: Transform the Implementation

Replace tangled ESBs with facades



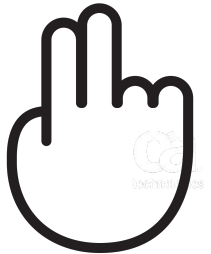
Step 2: Transform the Implementation

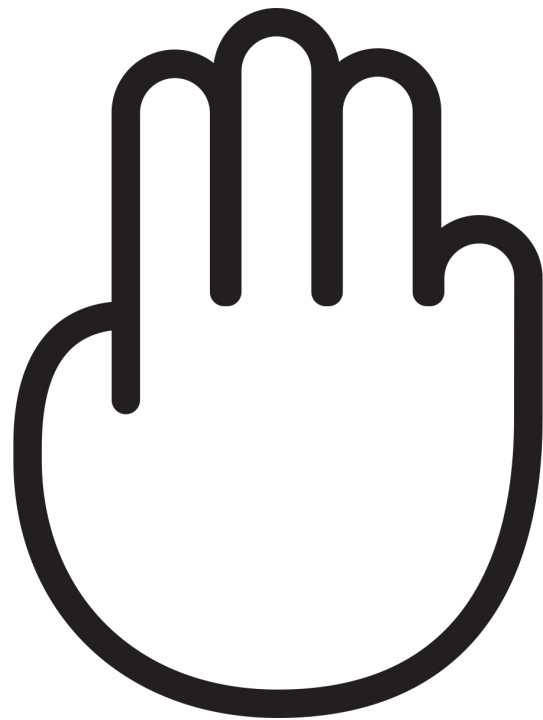
Replace tangled ESBs with facades



Transform the Implementation

- Refactor existing components
- Strangle the monolith
- Replace tangled ESBs



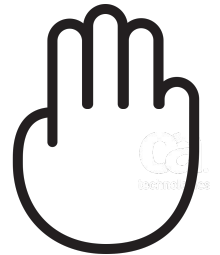




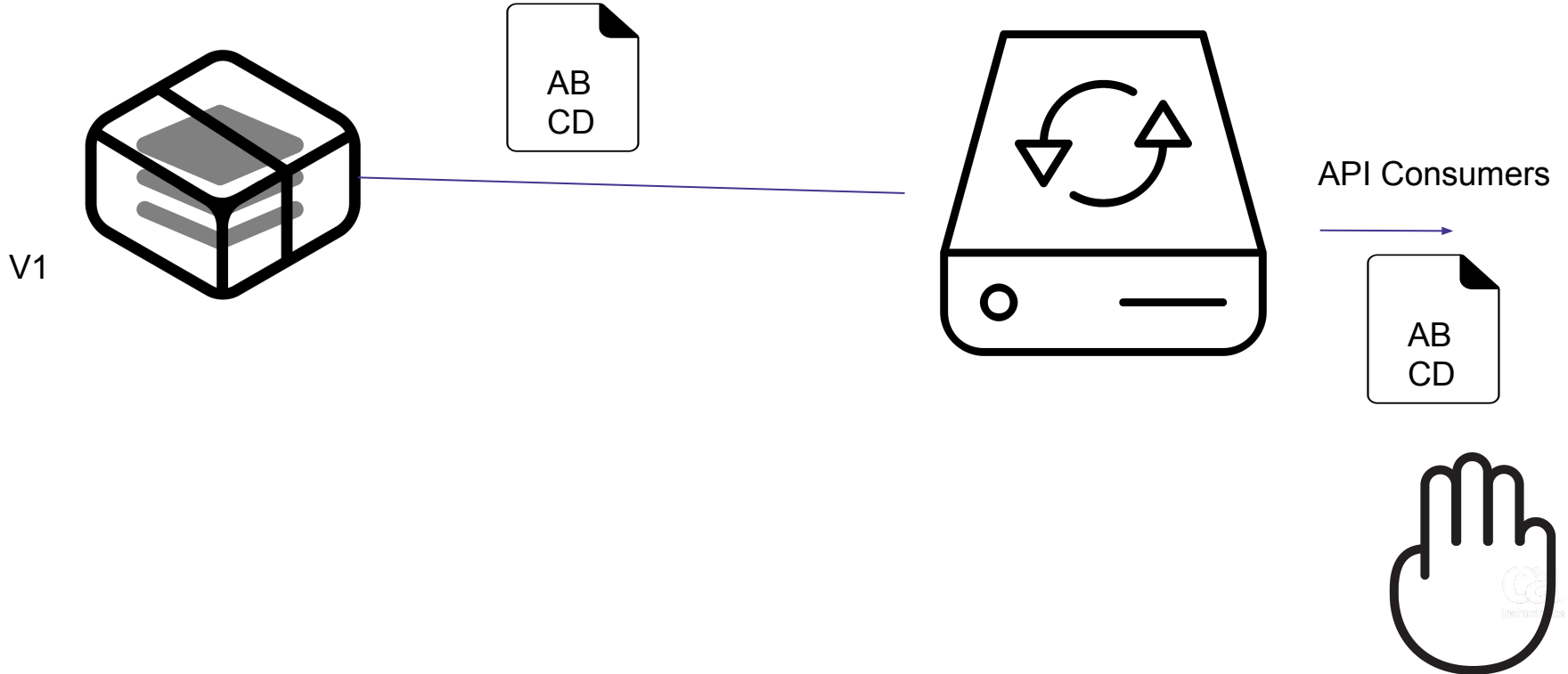
Step 3: Add Functionality

Step 3: Add Functionality

Update functionality via side-by-side components

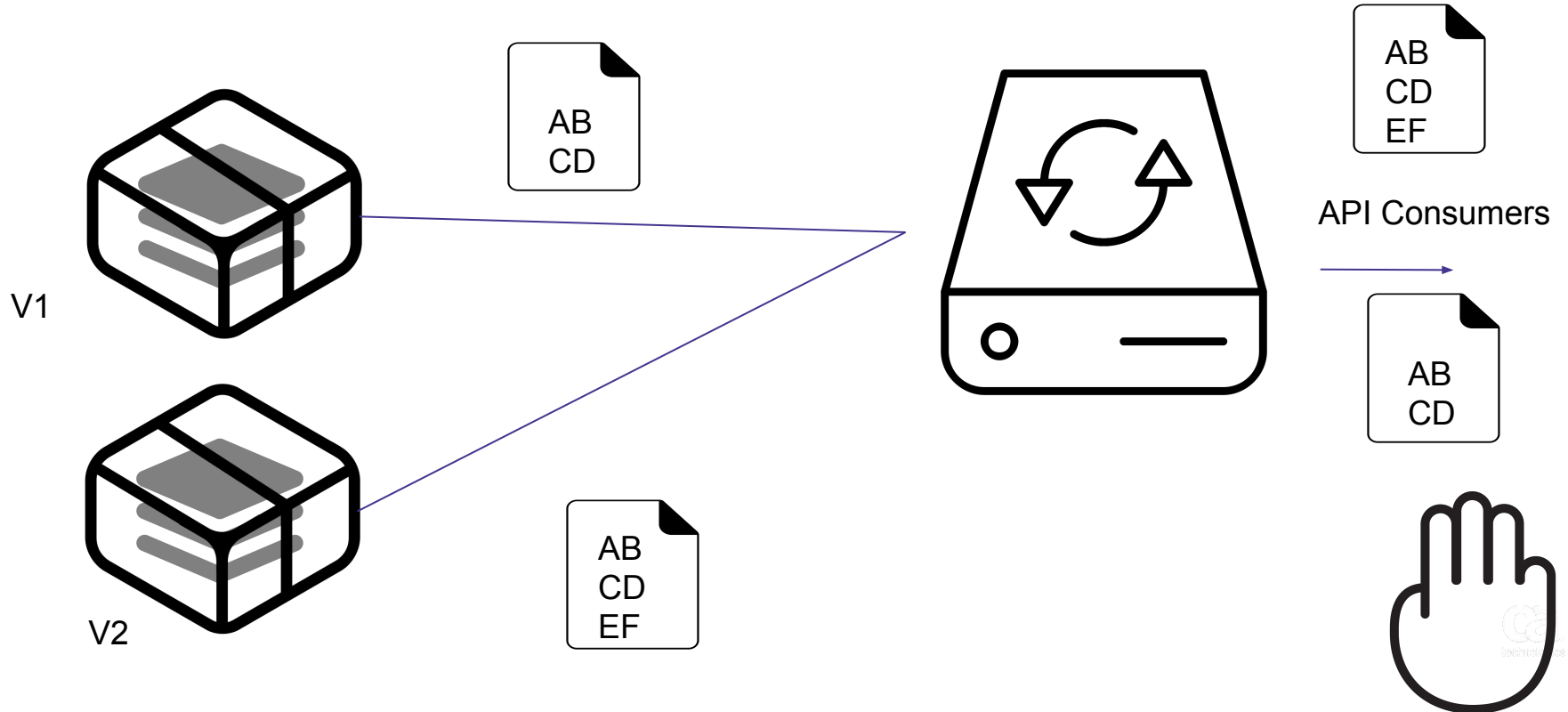


Update functionality via side-by-side components



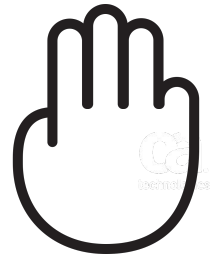
Step 3: Add Functionality

Update functionality via side-by-side components

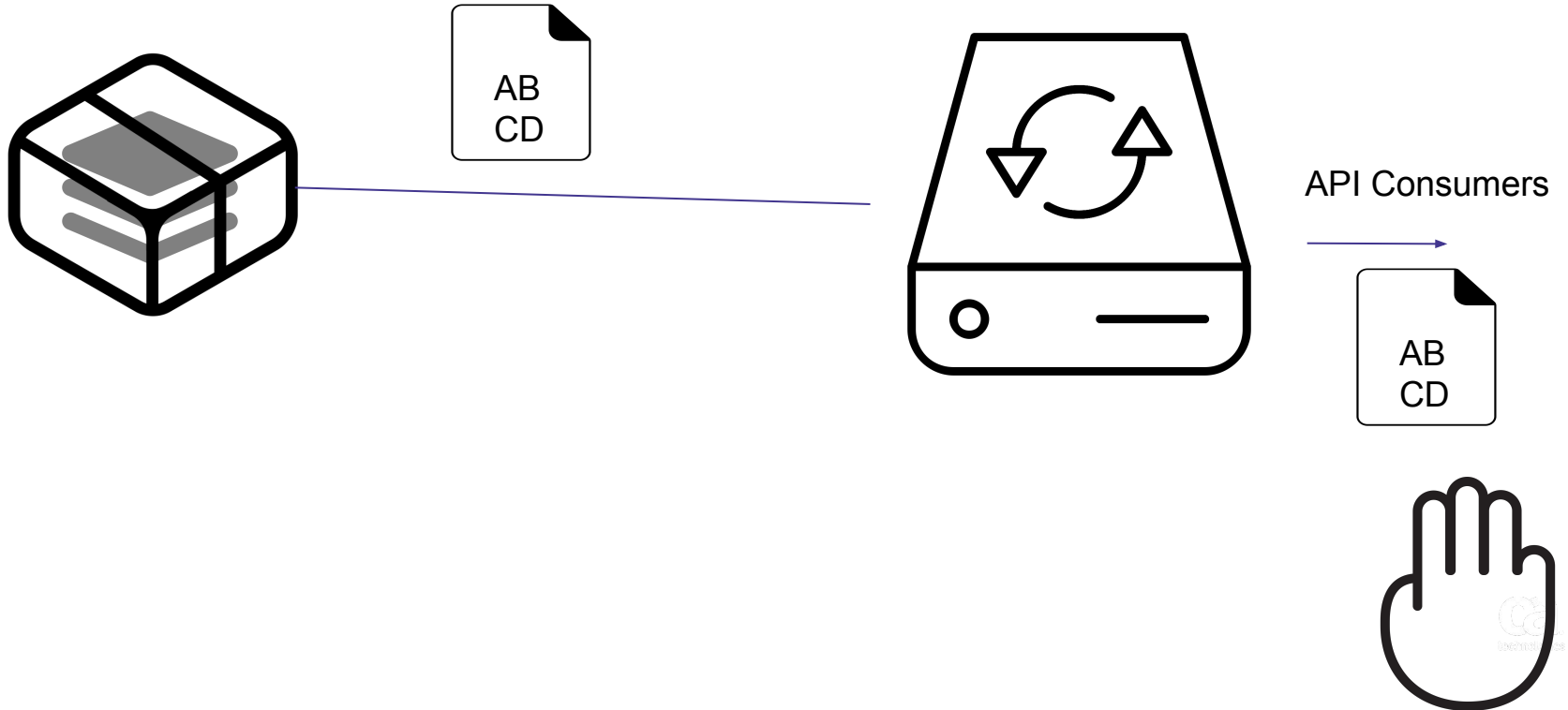


Step 3: Add Functionality

Introduce new functionality via new components

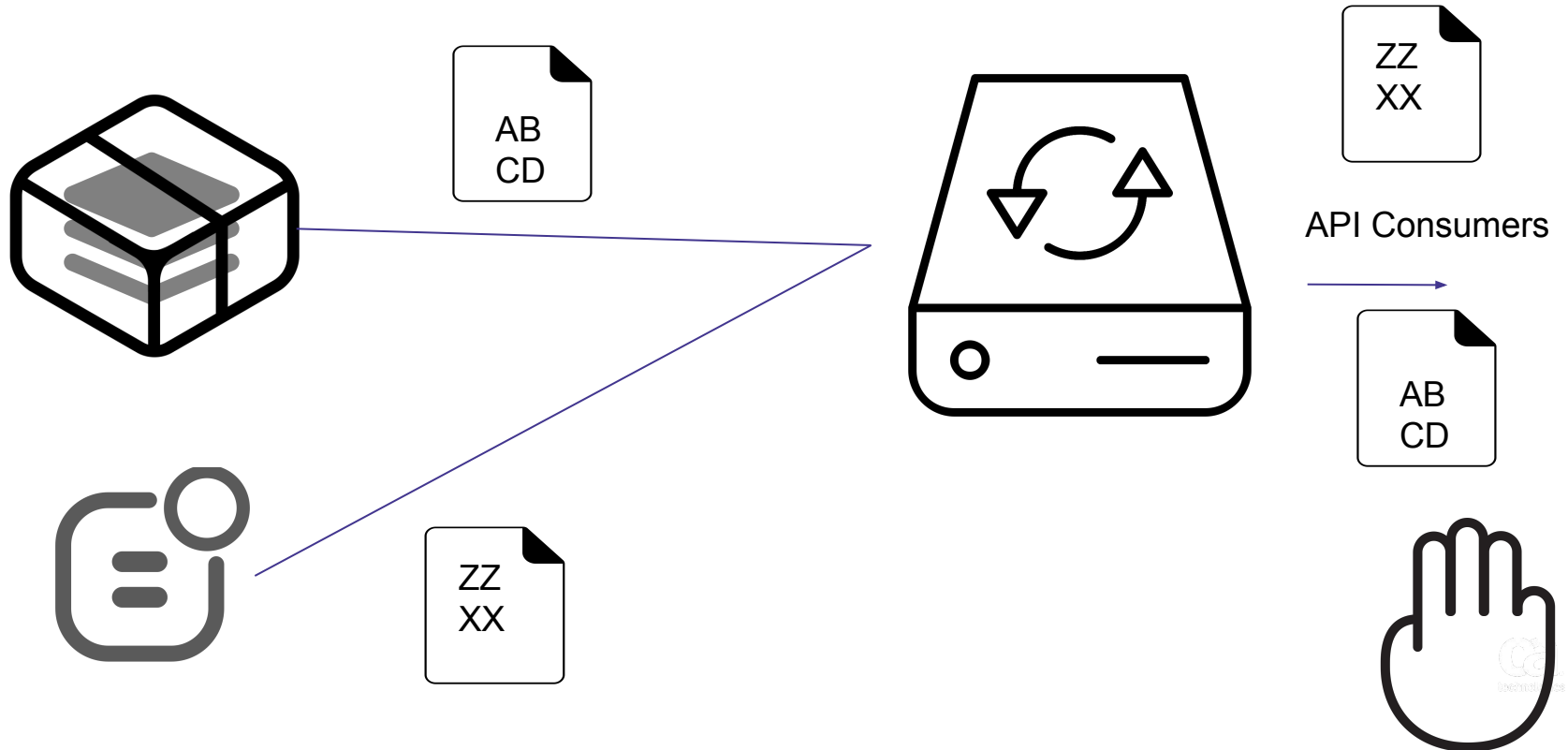


Introduce new functionality via new components

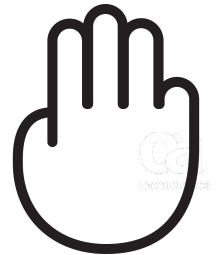


Step 3: Add Functionality

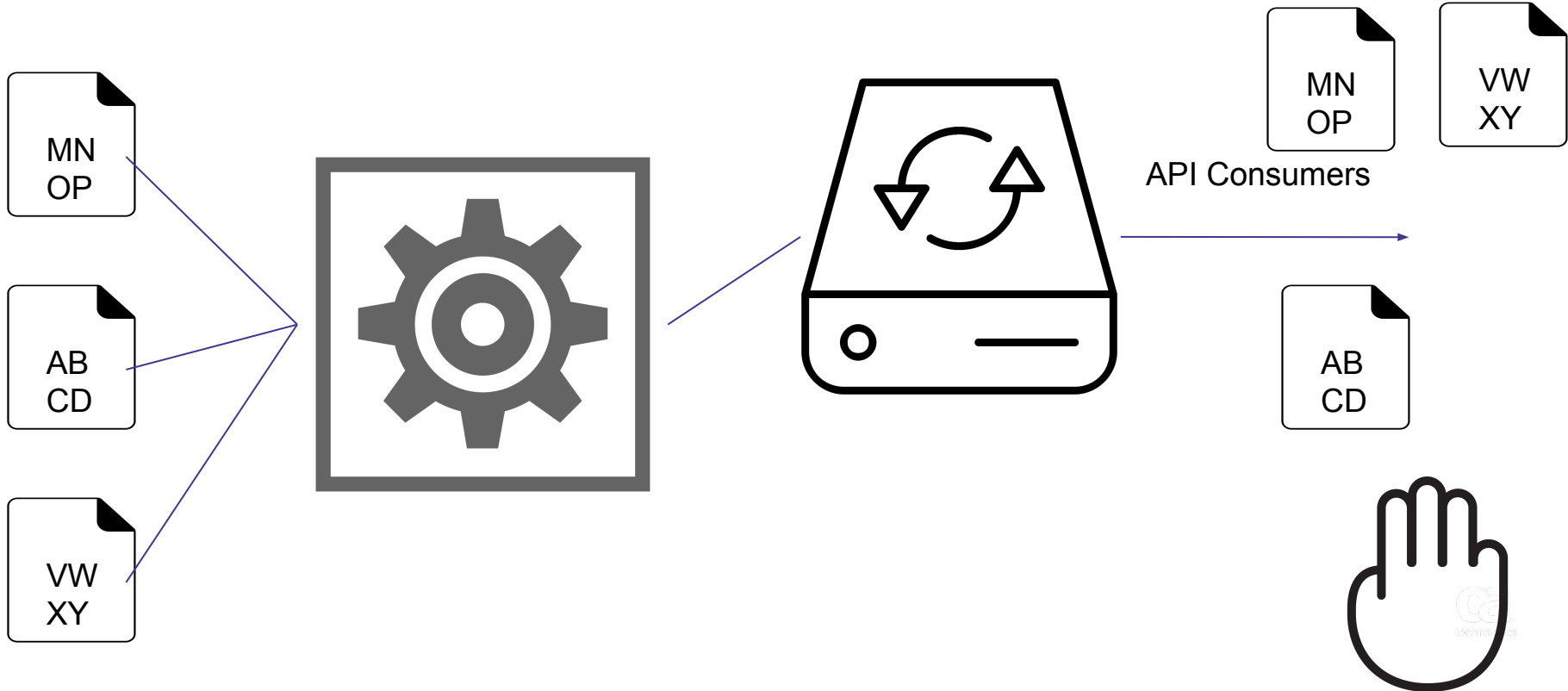
Introduce new functionality via new components



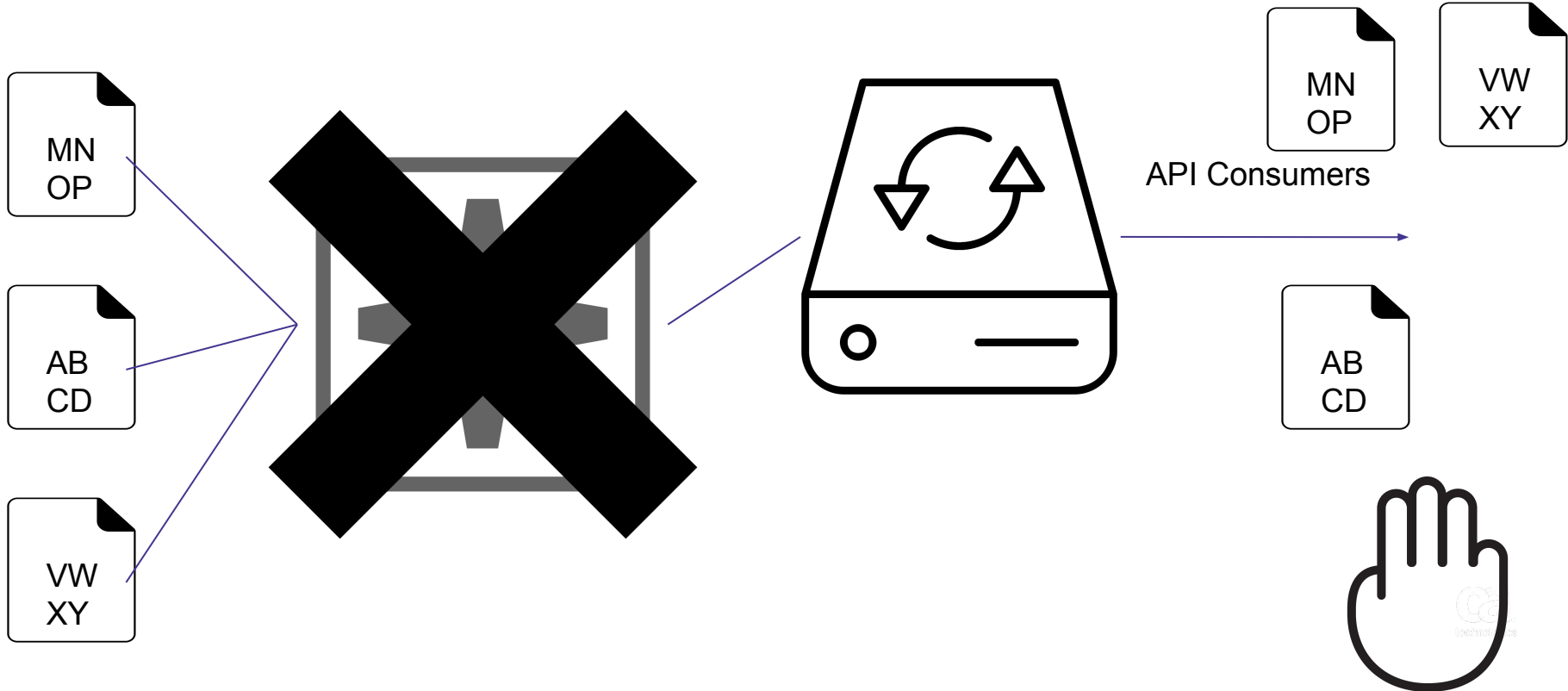
Acquire new functionality via 3rd party facades



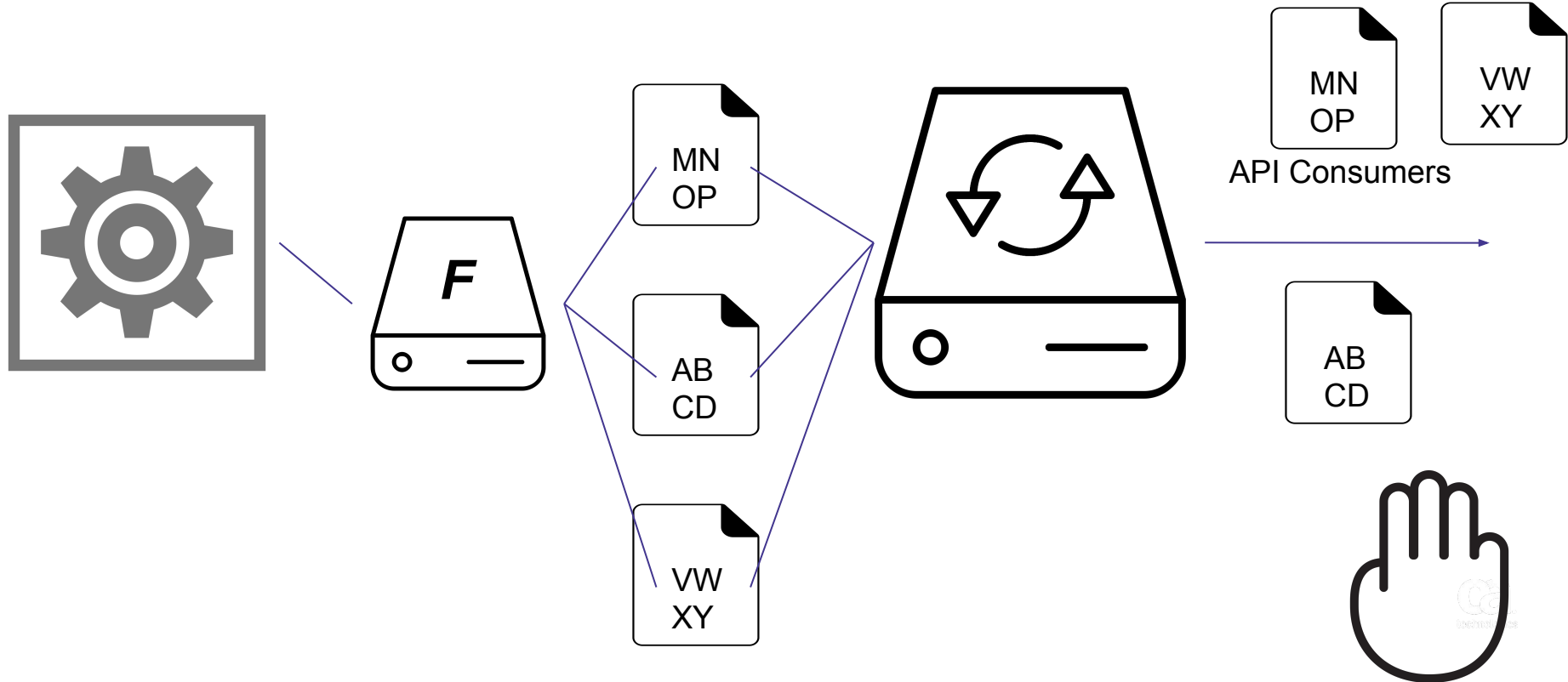
Acquire new functionality via 3rd party facades



Acquire new functionality via 3rd party facades

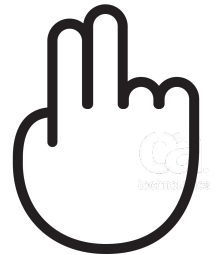


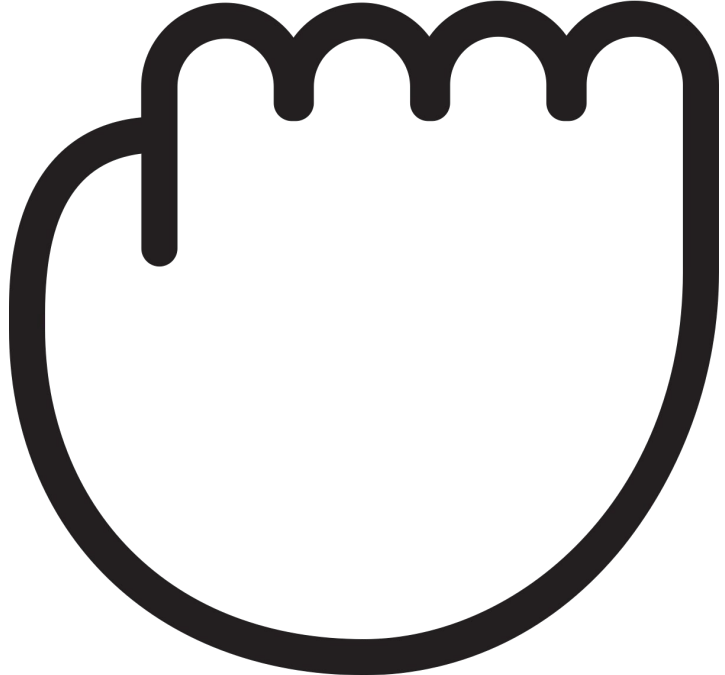
Acquire new functionality via 3rd party facades



Add Functionality

- Side-by-side updates
- New components
- External services facades



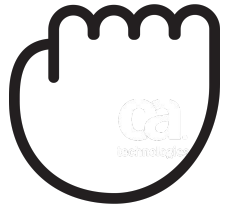




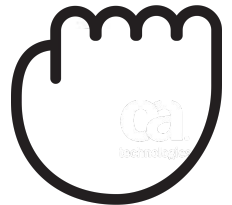
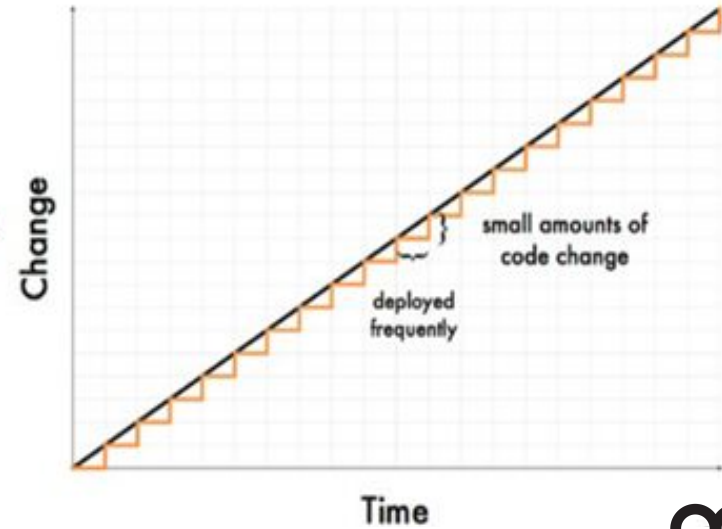
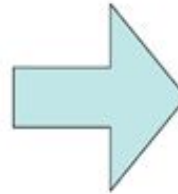
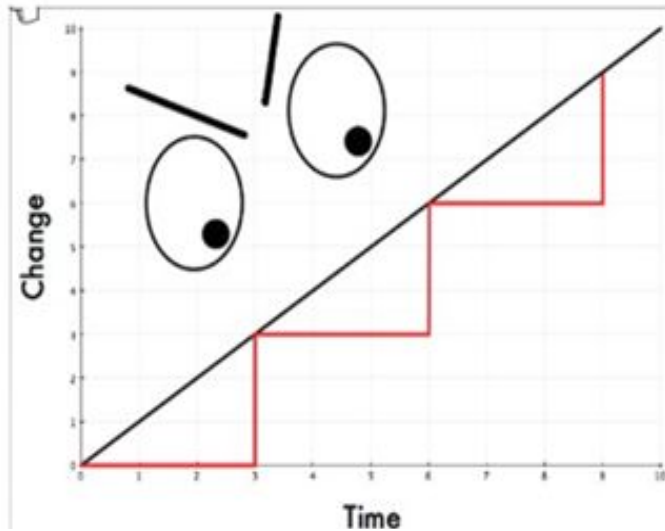
Step 4: Rinse and Repeat

Step 4: Rinse and Repeat

All changes are incremental



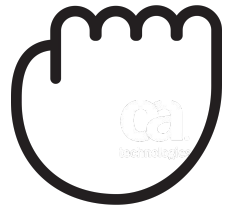
All changes are incremental



All changes are incremental

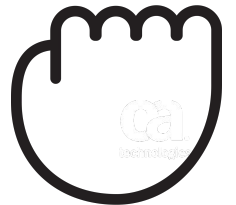
*"Incremental change may just be **the next big thing** this decade."*

-- Sandeep Kishore, HCL Technologies

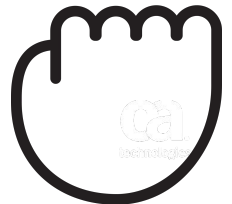
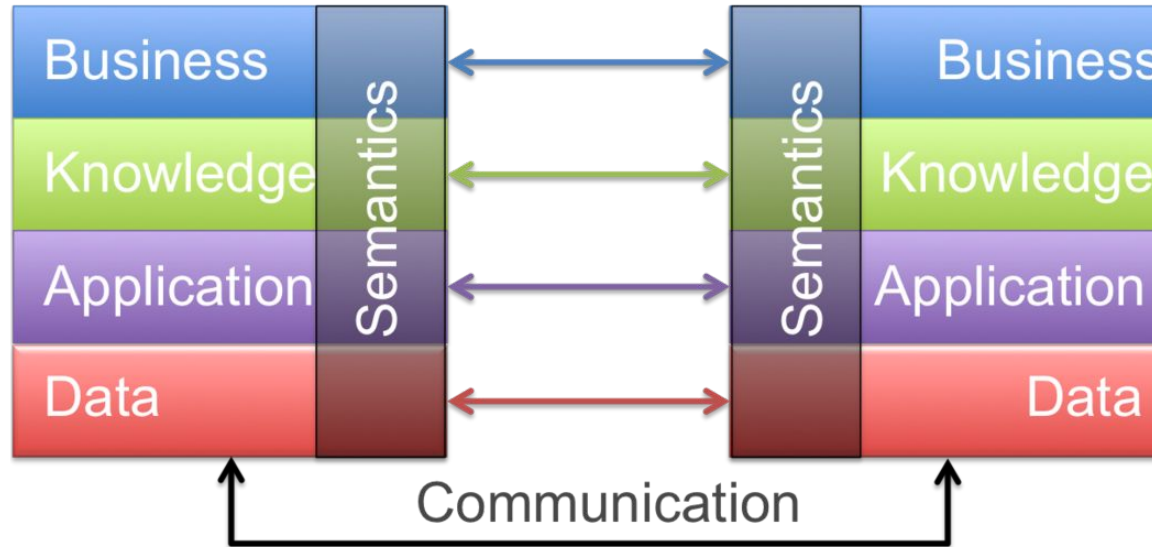


Step 4: Rinse and Repeat

Aim for loose interop, not tight integration



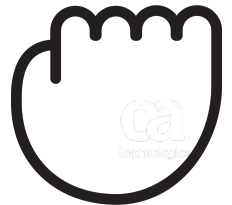
Aim for loose interop, not tight integration



Aim for loose interop, not tight integration

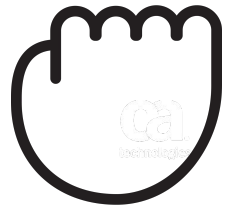
"Interoperation is peer to peer. Integration is where a system is subsumed within another."

-- Michael Platt, Microsoft

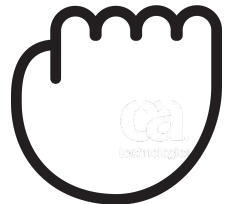
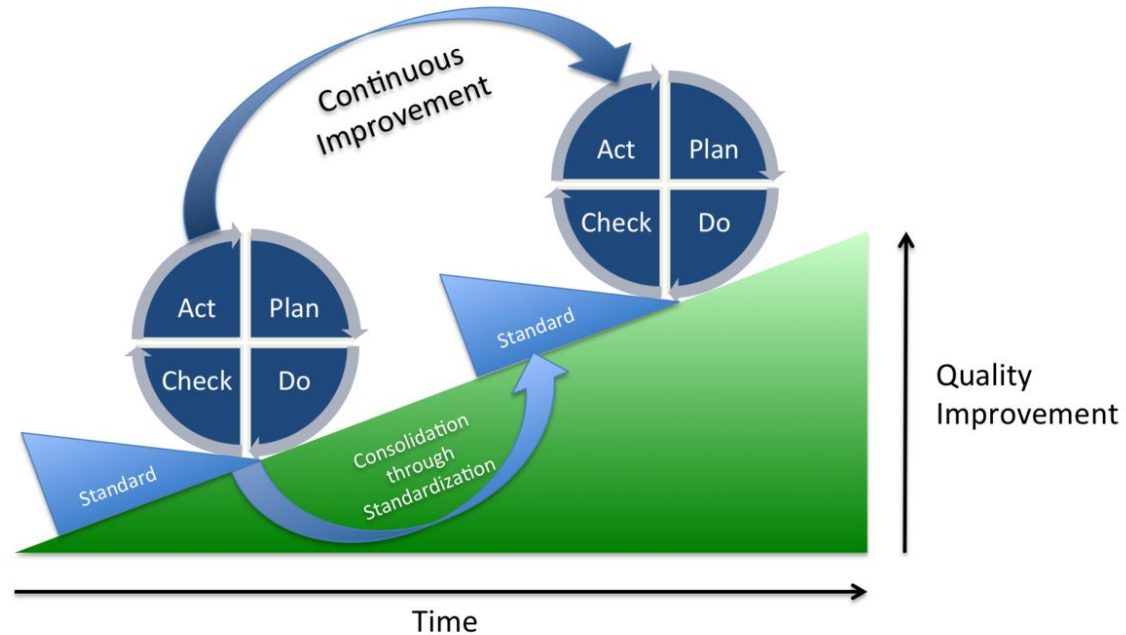


Step 4: Rinse and Repeat

Support continuous improvement



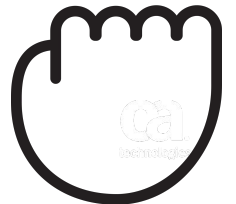
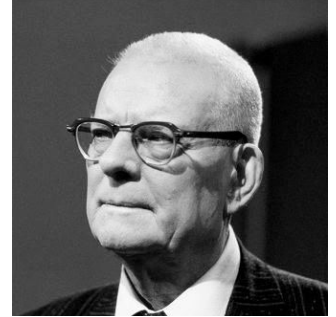
Support continuous improvement



Support continuous improvement

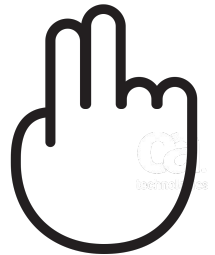
"Management's job is to improve the system."

-- W. Edwards Deming



Rinse and Repeat

- Make only incremental changes
- Aim for peer-to-peer interoperability
- Support continuous improvement



So...

The Quick Summary



The Quick Summary

- Focus on Unlocking Value



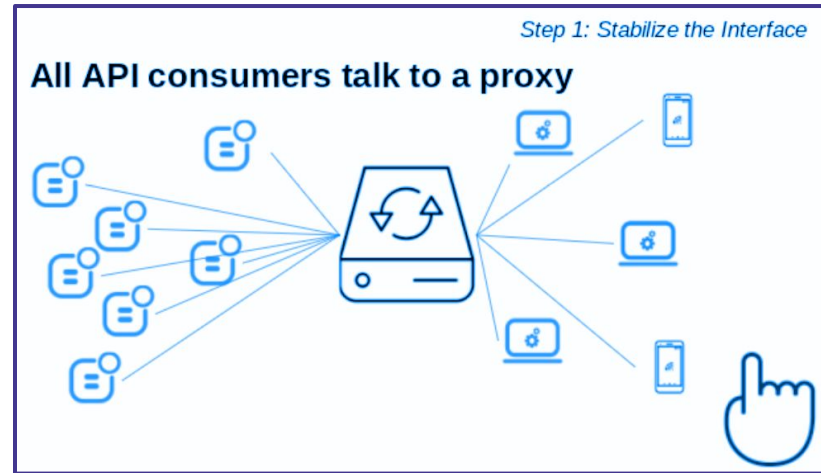
The Quick Summary

- Focus on Unlocking Value
- Change One Thing



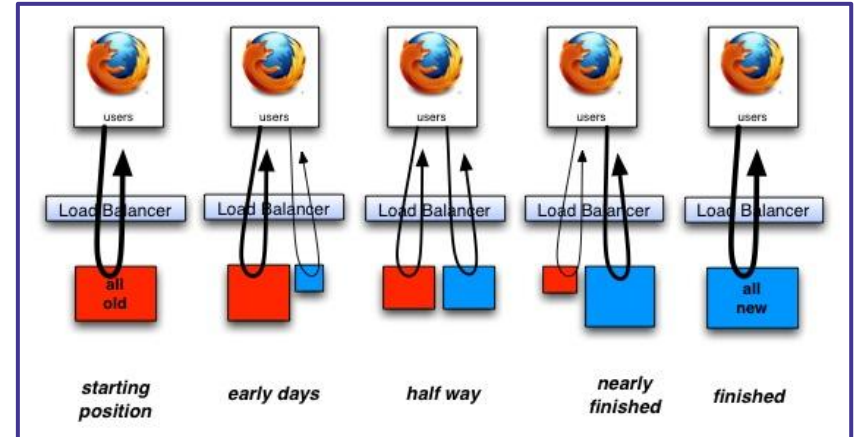
The Quick Summary

- Focus on Unlocking Value
- Change One Thing
- Stabilize the Interface



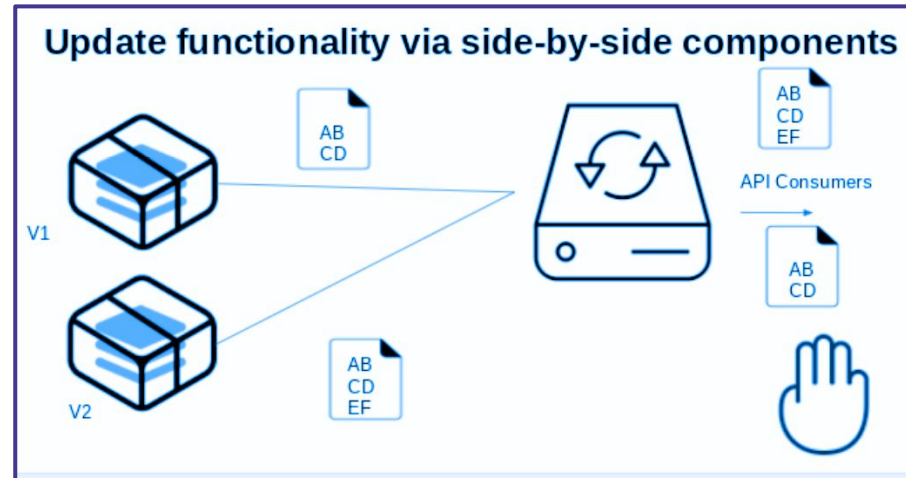
The Quick Summary

- Focus on Unlocking Value
- Change One Thing
- Stabilize the Interface
- Transform the Implementation



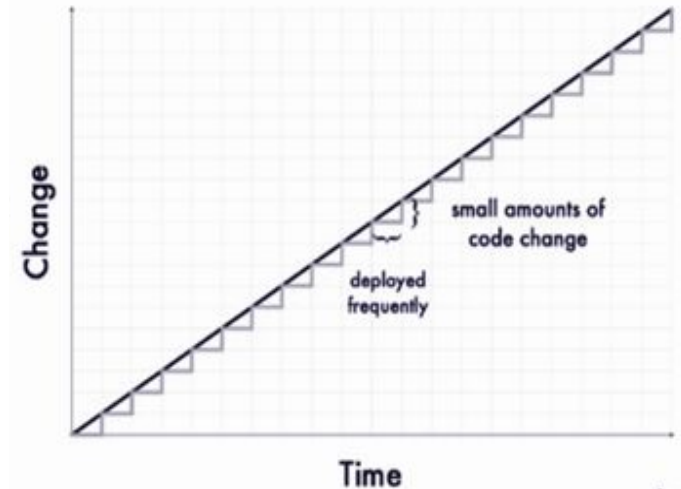
The Quick Summary

- Focus on Unlocking Value
- Change One Thing
- Stabilize the Interface
- Transform the Implementation
- Add Functionality



The Quick Summary

- Focus on Unlocking Value
- Change One Thing
- Stabilize the Interface
- Transform the Implementation
- Add Functionality
- Rinse and Repeat





Service and API Migration at Speed

Mike Amundsen

@mamund

[linkedin.com/in/mamund](https://www.linkedin.com/in/mamund)

[youtube.com/mamund](https://www.youtube.com/mamund)

