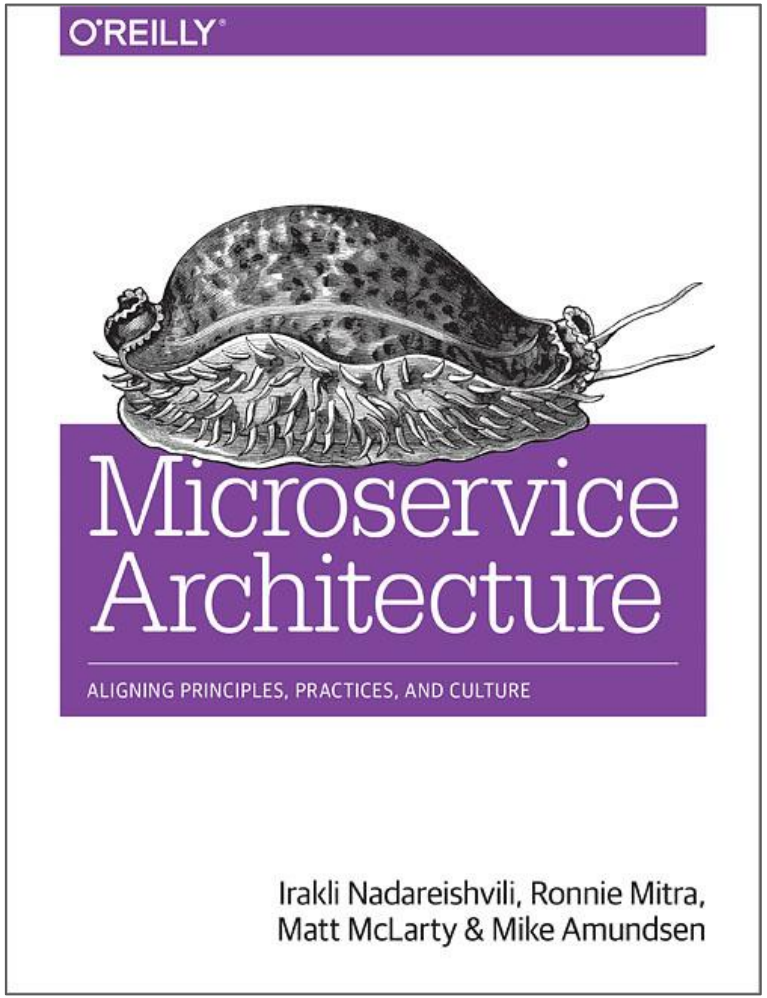
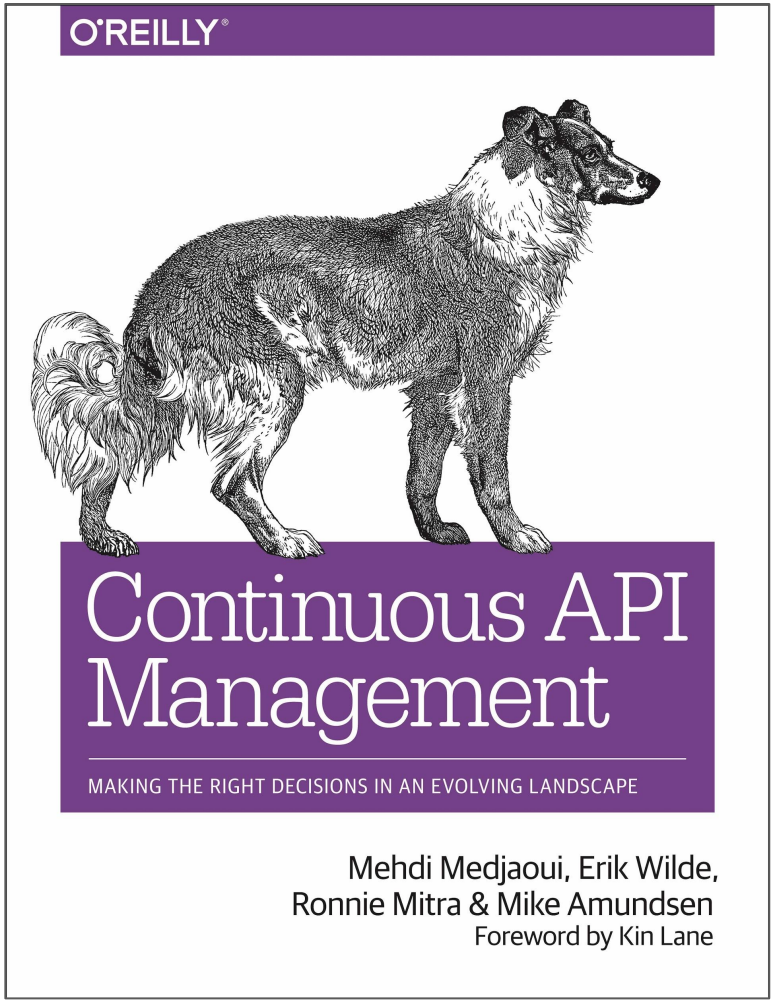
Service and API Migration at Speed

Mike Amundsen
@mamund
training.amundsen.com


Mike Amundsen
@mamund


O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Irakli Nadareishvili, Ronnie Mitra,
Matt McLarty & Mike Amundsen


O'REILLY
Continuous API Management
MAKING THE RIGHT DECISIONS IN AN EVOLVING LANDSCAPE
Mehdi Medjaoui, Erik Wilde,
Ronnie Mitra & Mike Amundsen
Foreword by Kin Lane

A Look Ahead

- Why?
- Unlocking Business Value
- Services in a Nutshell
- Basic Principles
- The **S*T*A*R** Model

What's in a name?

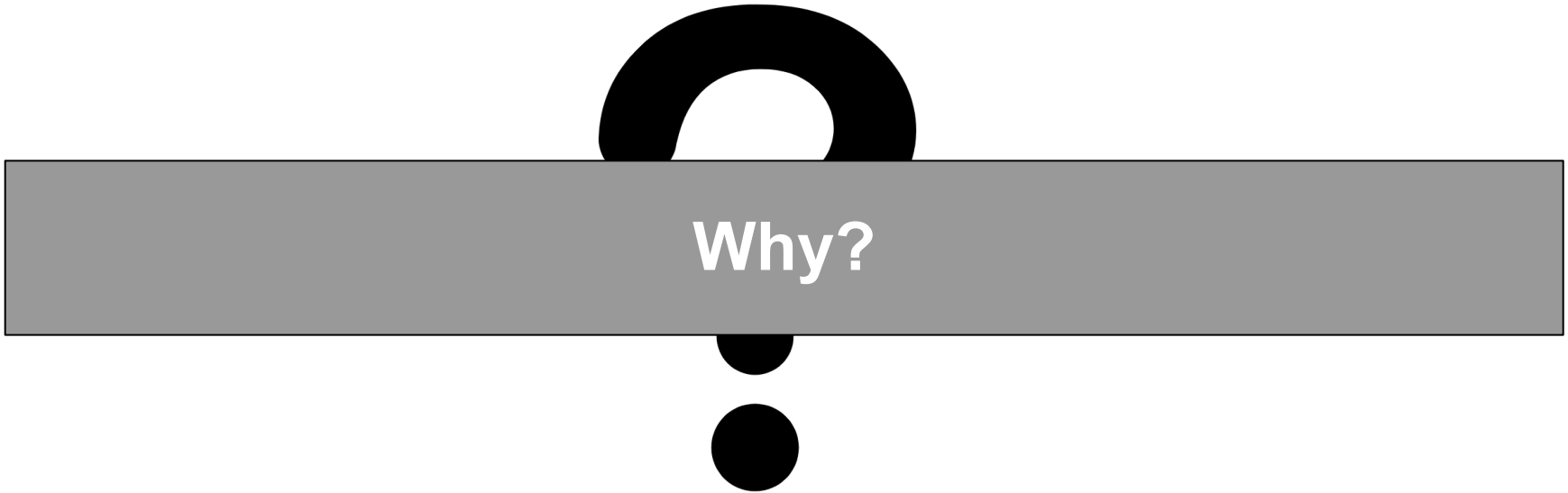
Naming things

- SOA
- Microservices
- Self-Contained Systems
- Right-sized Services

"We concluded that a service-oriented architecture would give us the level of isolation that would allow us to build many software components rapidly and independently."

-- Werner Vogels, 2006

https://queue.acm.org/detail.cfm?id=1142065


Why?

Reasons to migrate to microservices

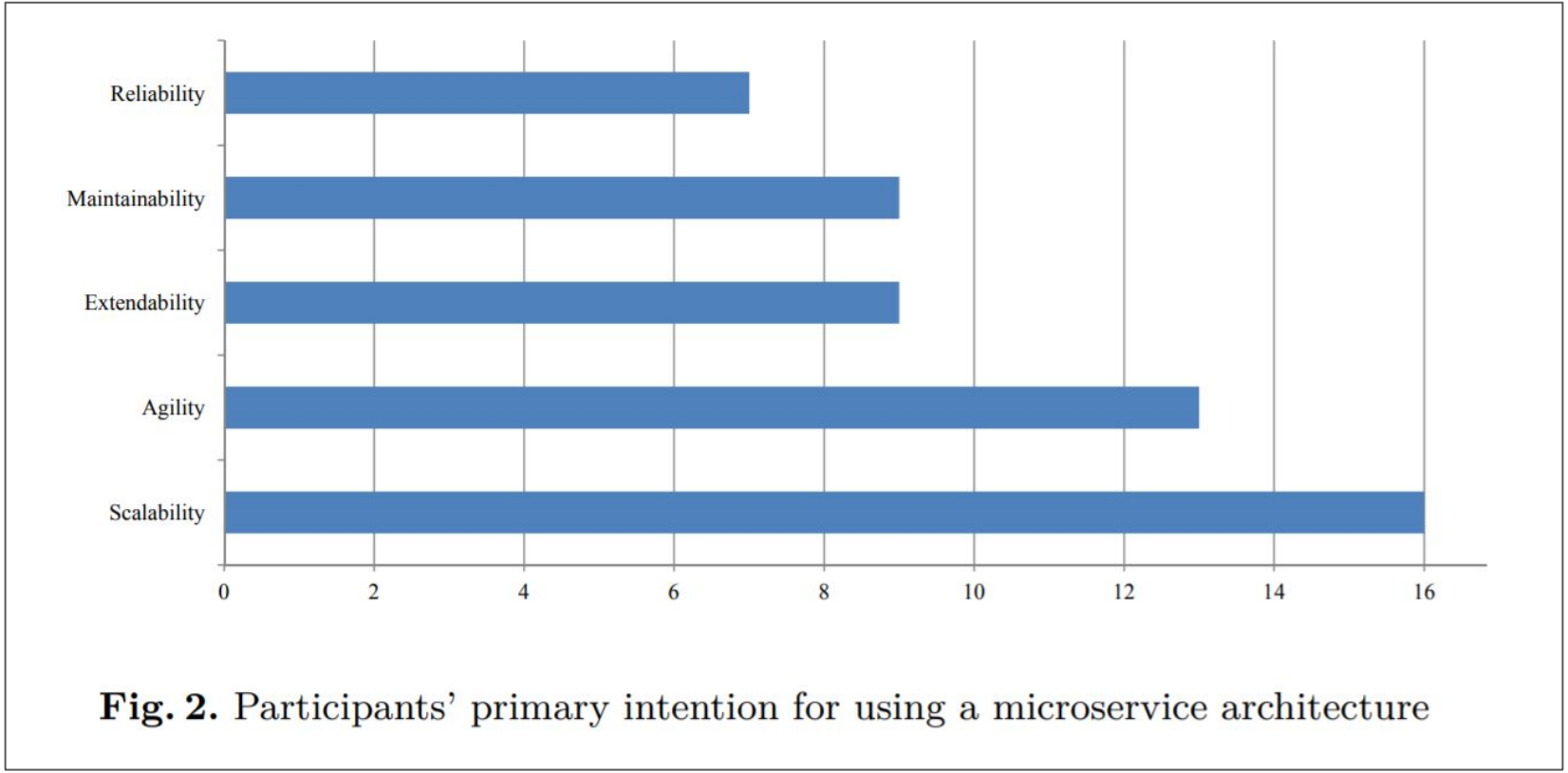
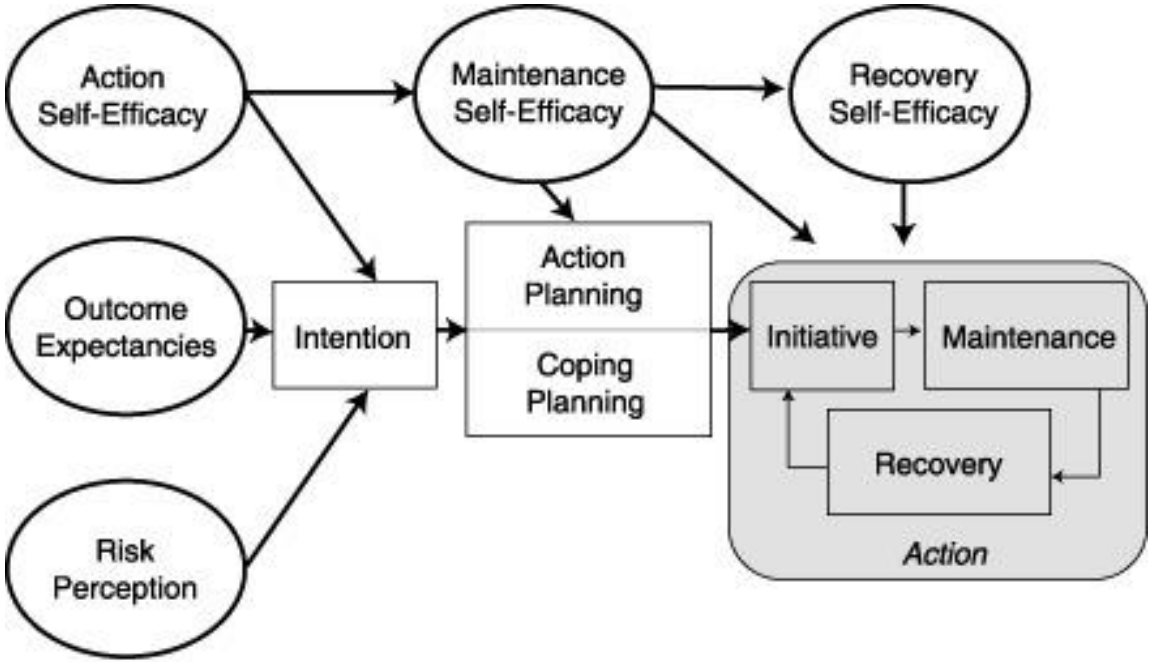
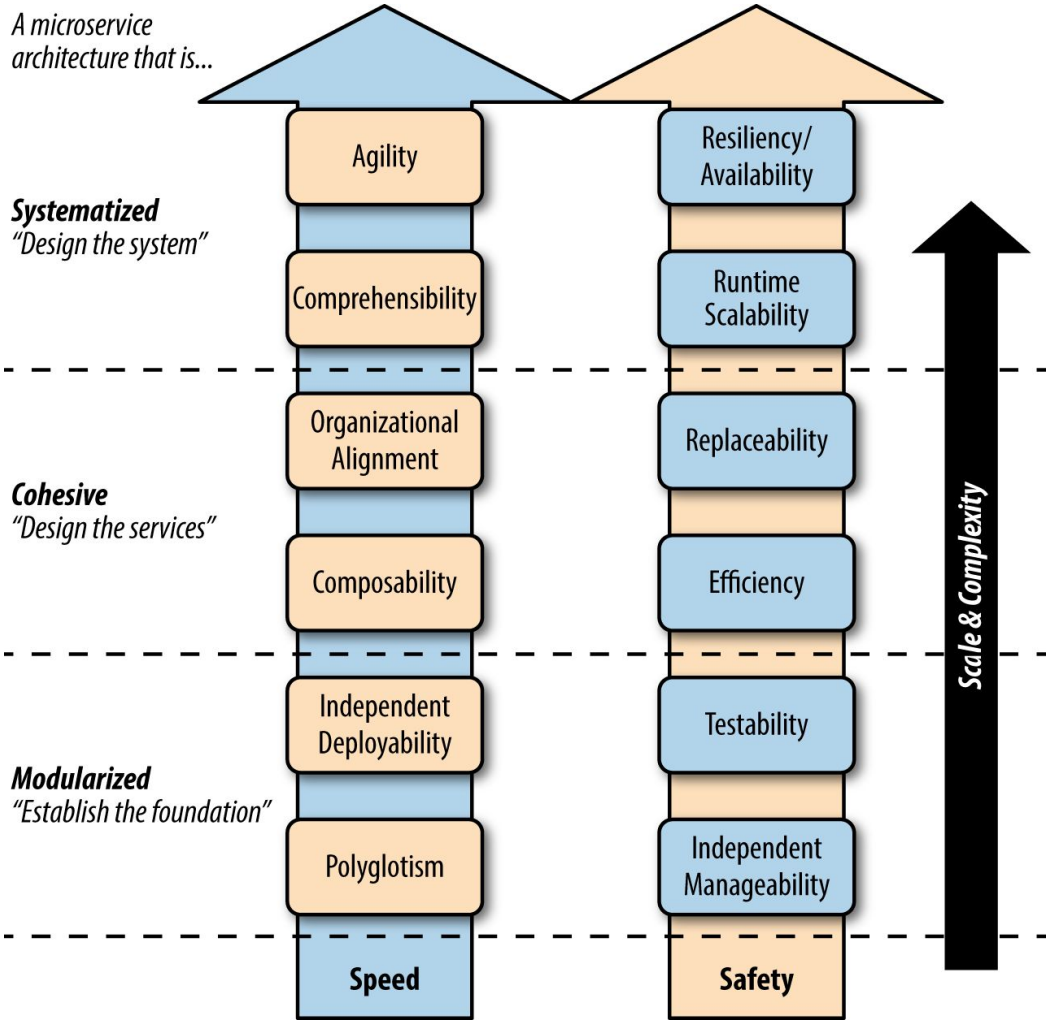


Fig. 2. Participants' primary intention for using a microservice architecture

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


TOPEAK


Action Self-Efficacy
Maintenance Self-Efficacy
Recovery Self-Efficacy
Outcome Expectancies
Intention
Action Planning
Coping Planning
Initiative
Maintenance
Recovery
Action
Risk Perception


A microservice architecture that is...
Systematized
"Design the system"
Cohesive
"Design the services"
Modularized
"Establish the foundation"
Agility
Comprehensibility
Organizational Alignment
Composability
Independent Deployability
Polyglotism
Speed
Resiliency/ Availability
Runtime Scalability
Replaceability
Efficiency
Testability
Independent Manageability
Safety
Scale & Complexity

"If you go back to 2001, the Amazon.com retail website was a large architectural monolith … and it took a long time for a code change to go from check-in, to be running in production."

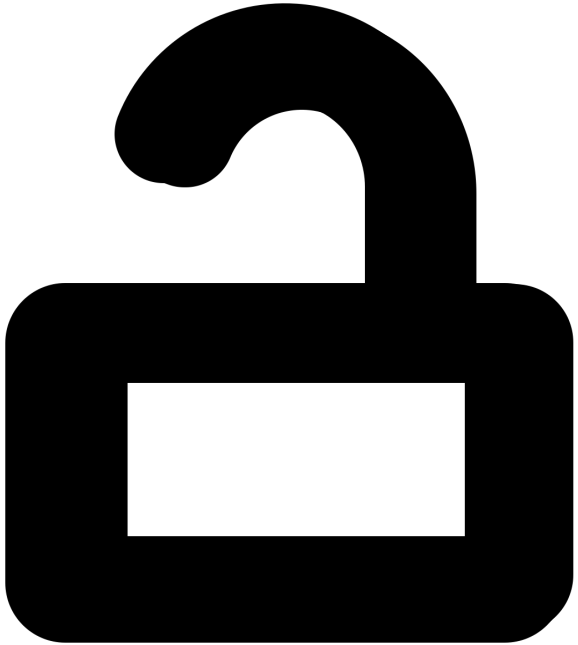
-- Rob Brigham, Amazon, 2015

https://thenewstack.io/led-amazon-microservices-architecture/


FLY CALIFORNIA

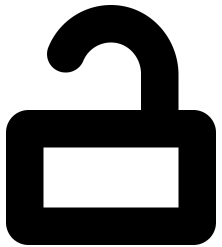

VAULT DOOR
WEIGHT: 22 1/2 Tons
THICKNESS: 22 Inches
STEEL: 11 Layers of Special Cutting and Drill Resistant
LOCKS: 4 Hamilton Watch Movements for Time Locks

Balancing Speed and Safety at Scale

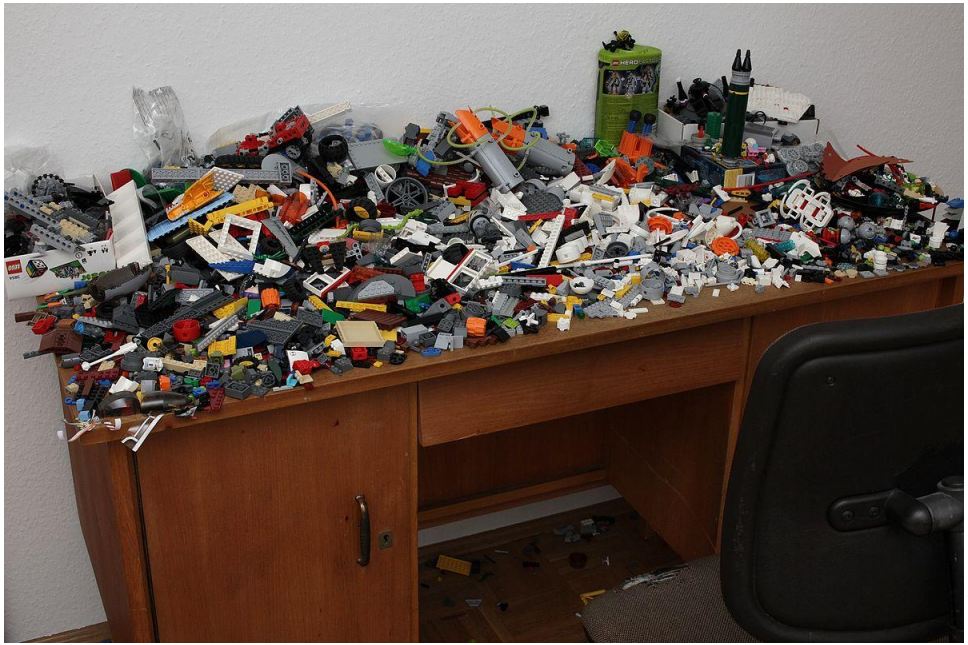


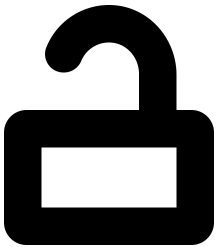

Unlocking Business Value

Where is everything?



Where is everything?



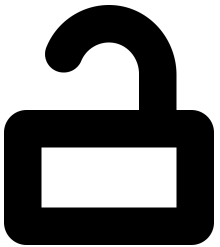


Where is everything?

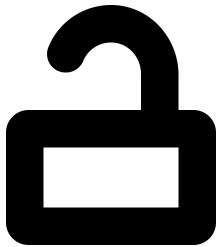
"Data and services are stuck inside isolated applications within the enterprise."

-- Tung and Biltz, Accenture

http://www.computerweekly.com/feature/APIs-can-be-strategic-tools-to-unlock-business-value

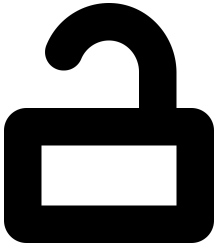


Why does it cost so much to get at it?



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

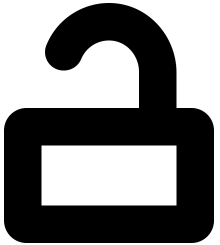


Why does it cost so much to get at it?

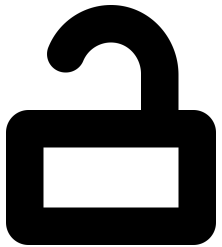
"It is about renovating at the core, as opposed to getting rid of the core."

-- Hung LeHong, Gartner

http://www.zdnet.com/article/eight-obstacles-to-overcome-in-your-digital-transformation-journey/



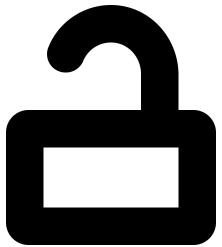
How can I reduce cost/risk?



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

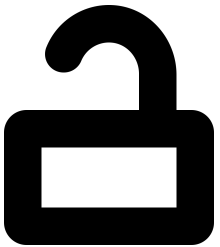


How can I reduce cost/risk?

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

-- John Allspaw, Etsy

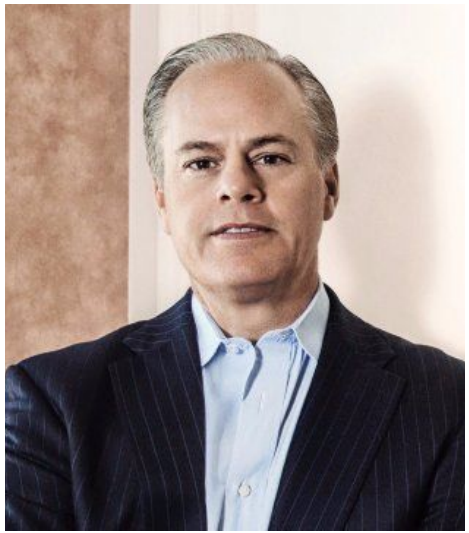
https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr/16-Dev_and_Ops/16




Changes Ahead

"Built for Change."

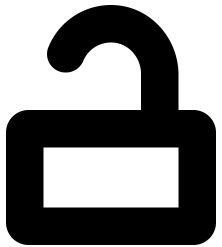
-- Mike Gregoire, CEO, CA Technologies



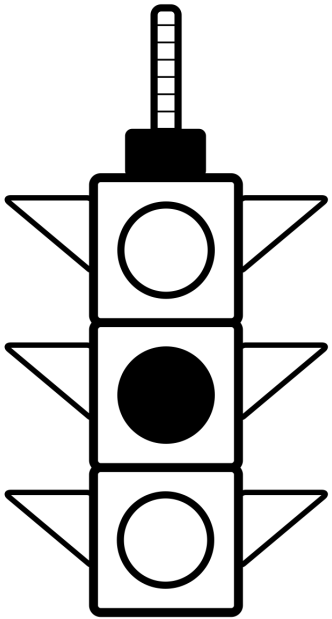
How can we do it?

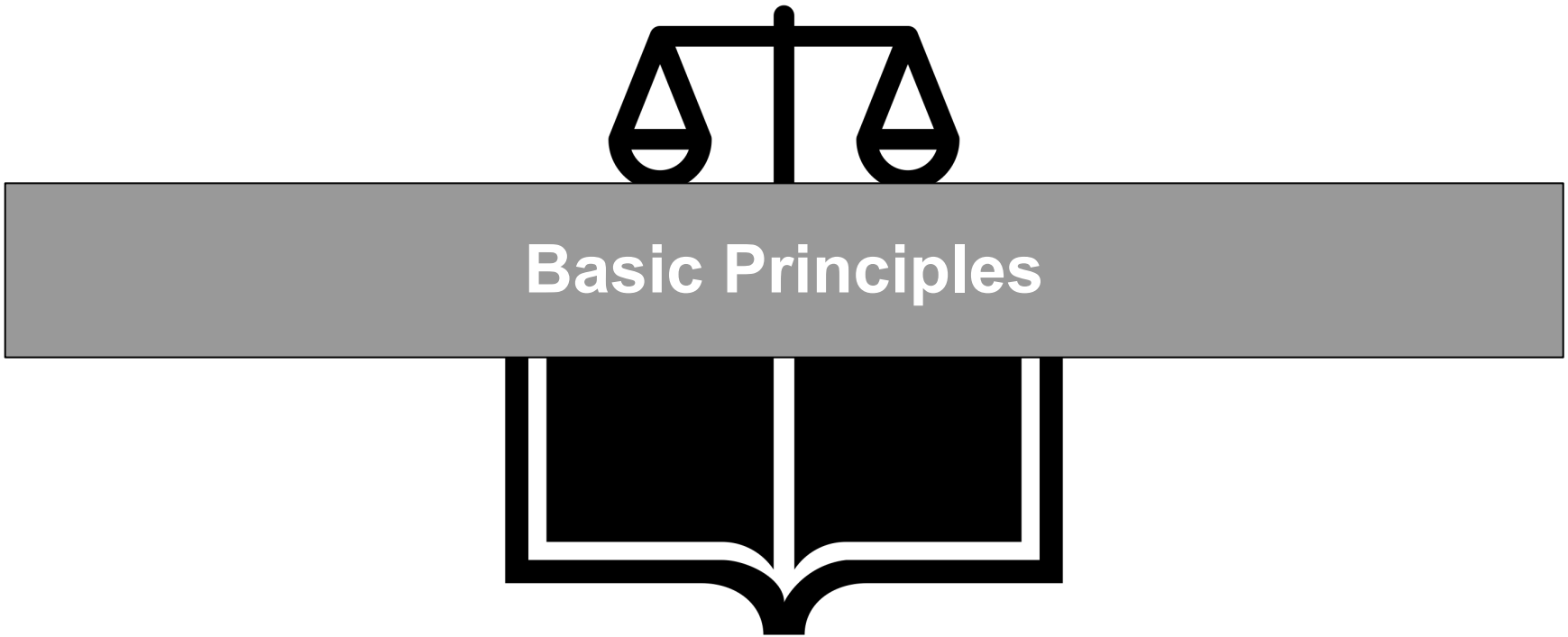
Give your system the STAR treatment

- **S**tabilize
- **T**ransform
- **A**dd
- **R**epeat



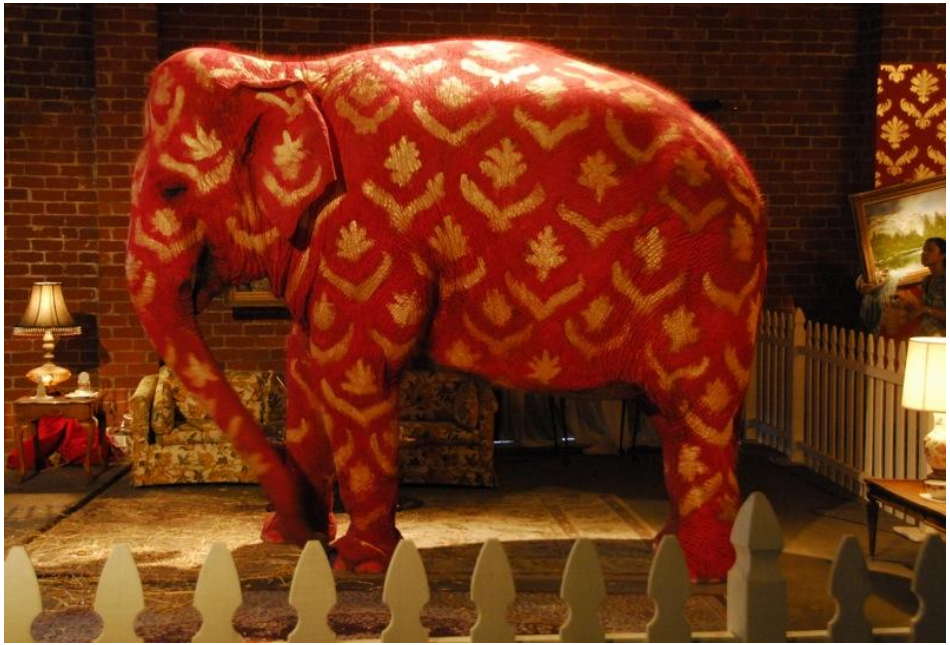
But first...




Basic Principles

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

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

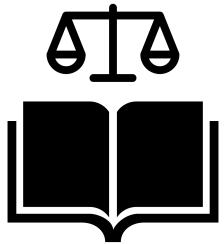


The elephant in the room: one bite at a time.

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

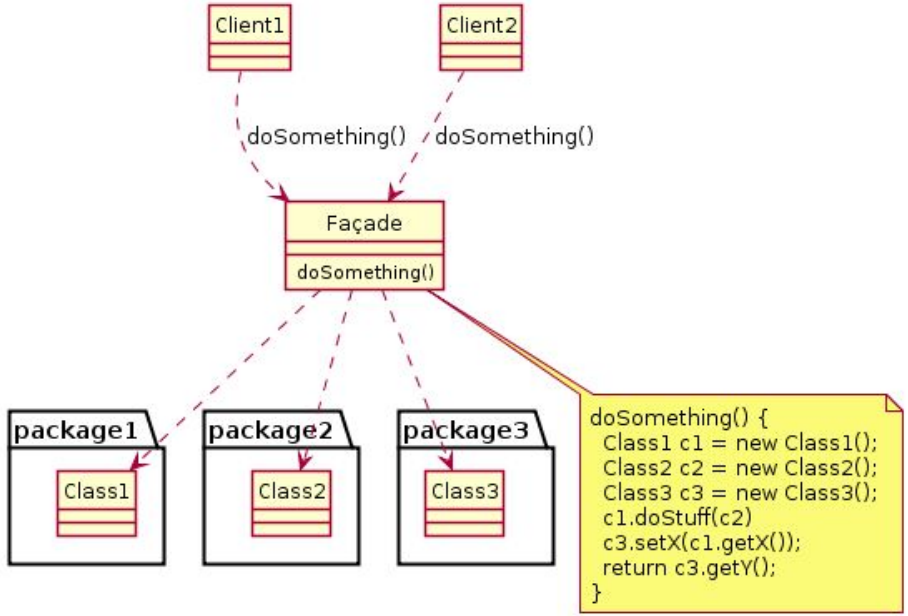
-- Adrian Cockcroft, Netflix

https://medium.com/s-c-a-l-e/talking-microservices-with-the-man-who-made-netflix-s-cloud-famous-1032689afed3

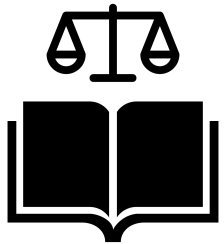


Employ facades, stranglers, and refactoring

Employ facades, stranglers, and refactoring



https://upload.wikimedia.org/wikipedia/en/5/57/Example_of_Facade_design_pattern_in_UML.png

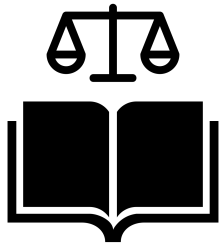


Employ facades, stranglers, and refactoring

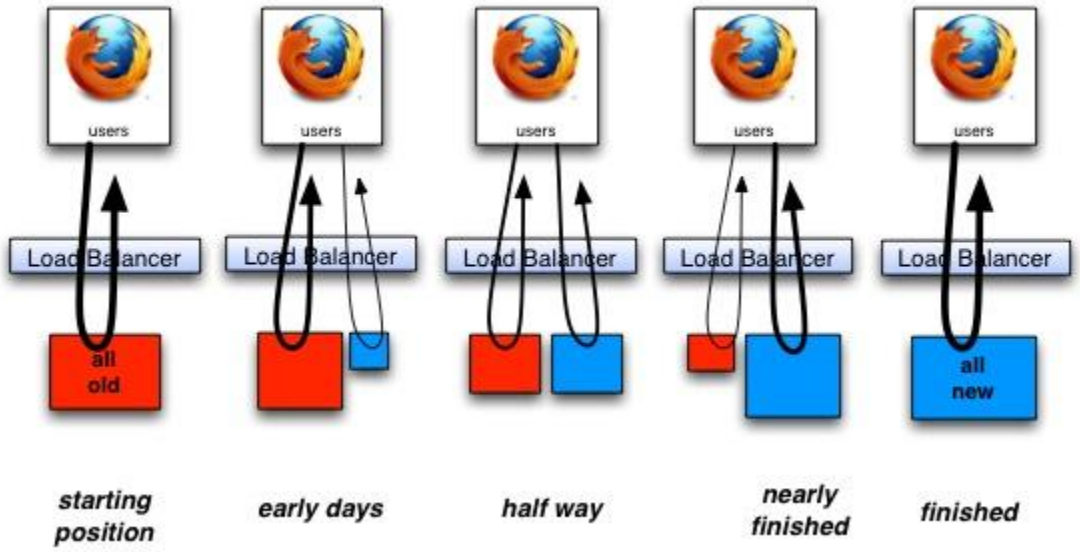
*"The **facade** design pattern is used to define a simplified interface to a more complex subsystem."*

-- Richard Carr, BlackWasp

http://www.blackwasp.co.uk/facade.aspx



Employ facades, stranglers, and refactoring



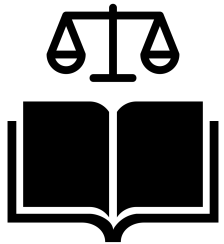
https://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/

Employ facades, stranglers, and refactoring

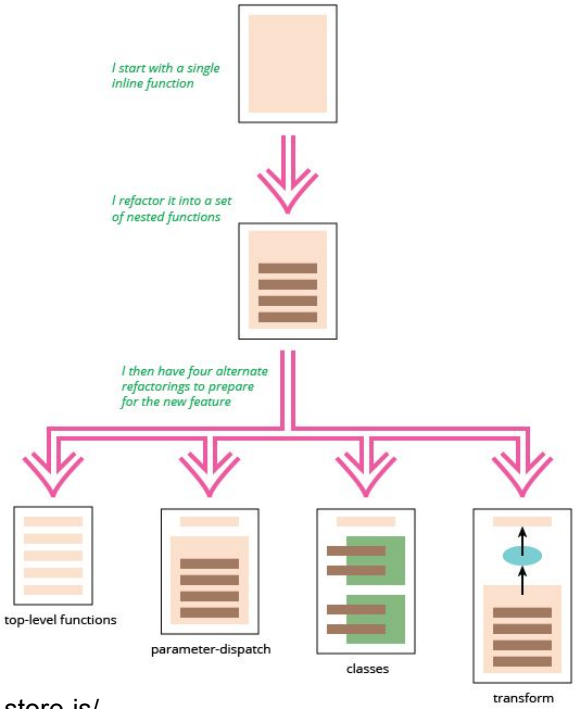
*"**Strangulation** of a legacy solution is a safe way to phase one thing out for something better."*

-- Paul Hammant, Thoughtworks

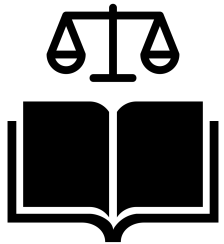
https://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/



Employ facades, stranglers, and refactoring



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

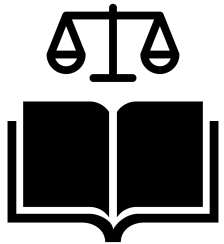


Employ facades, stranglers, and refactoring

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

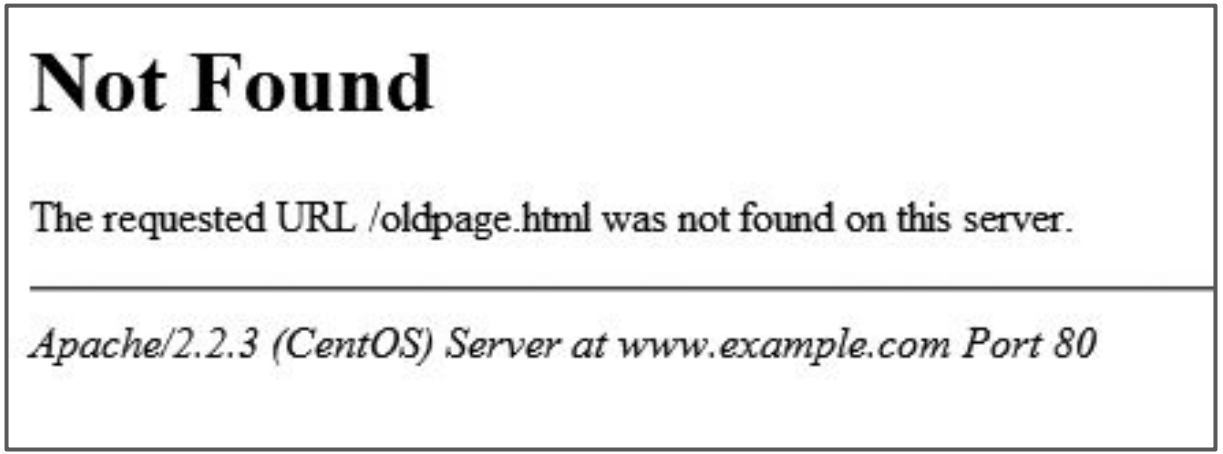
-- Martin Fowler, Thoughtworks

https://martinfowler.com/books/refactoring.html

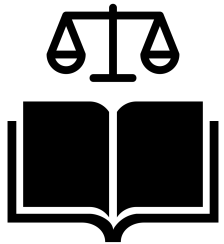


APIs are forever, code is not.

APIs are forever, code is not.



https://upload.wikimedia.org/wikipedia/commons/5/5f/404_not_found.png

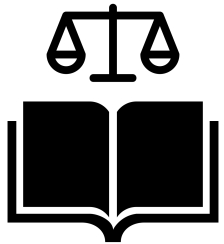


APIs are forever, code is not.

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

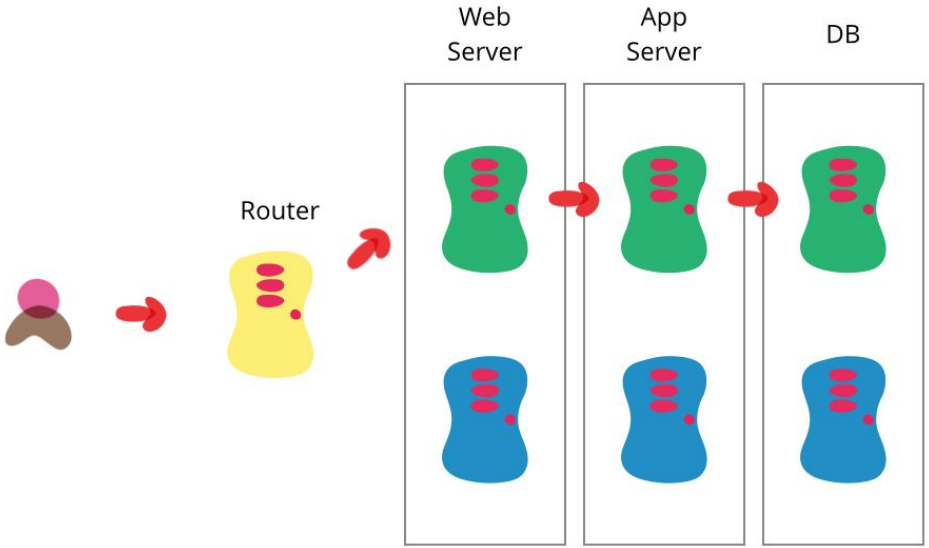
-- Werner Vogels, Amazon

http://www.allthingsdistributed.com/2016/03/10-lessons-from-10-years-of-aws.html

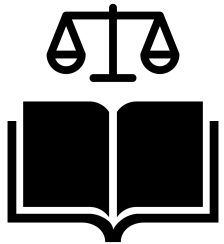


Continuous change and instant reversibility

Continuous change and instant reversibility



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

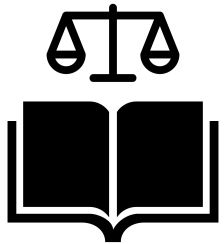


Continuous change and instant reversibility

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

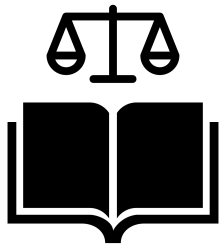
-- Martin Fowler, Thoughtworks

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

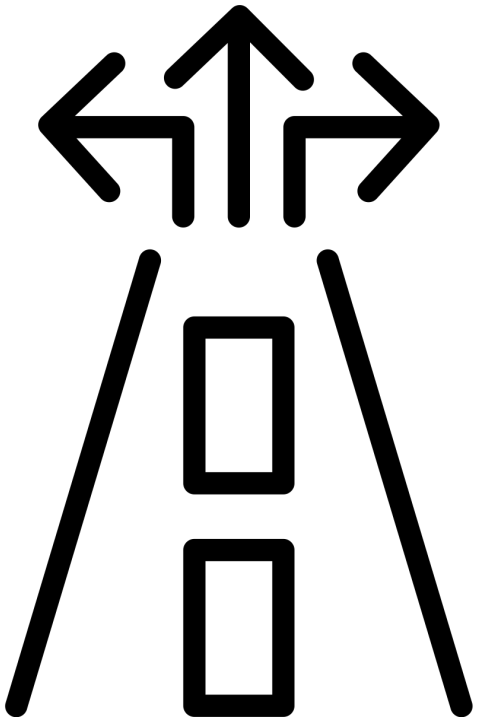


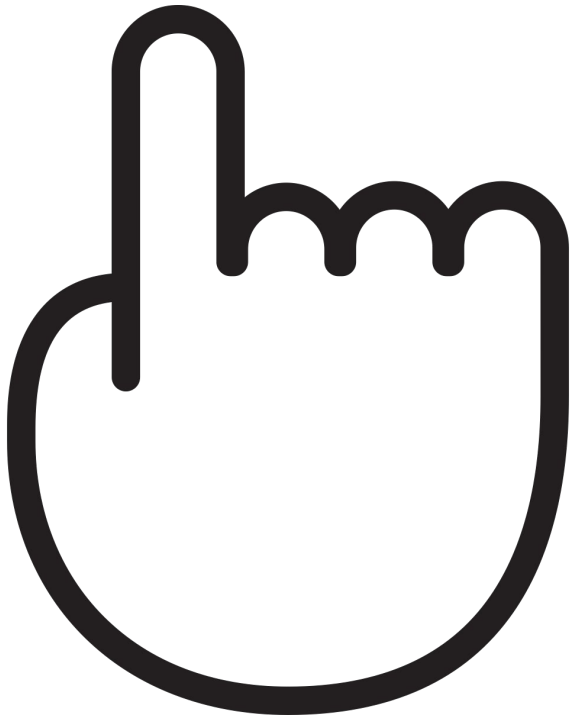
Basic Principles

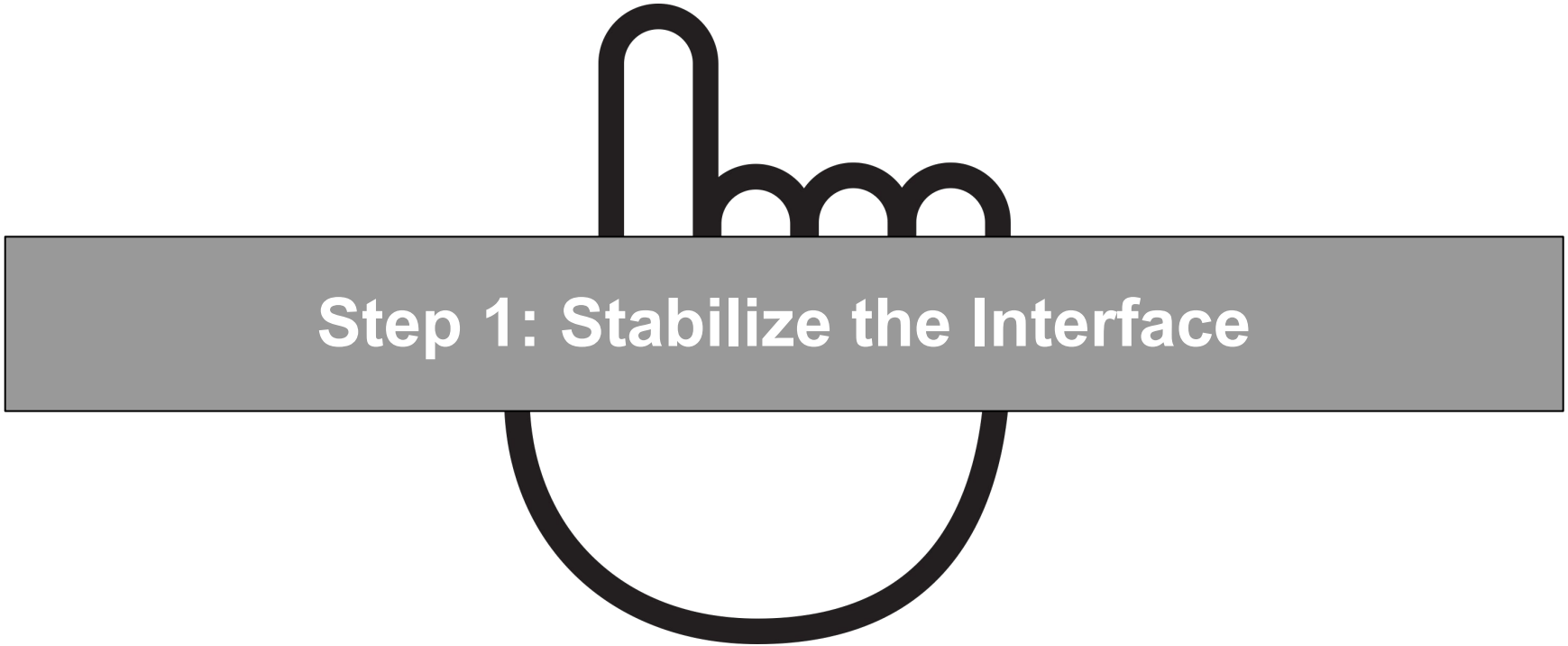
- Take one bite at a time.
- Employ facades, stranglers, and refactoring
- APIs are forever, code is not
- Continuous change and instant reversibility



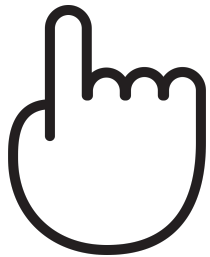
So, what's the roadmap?



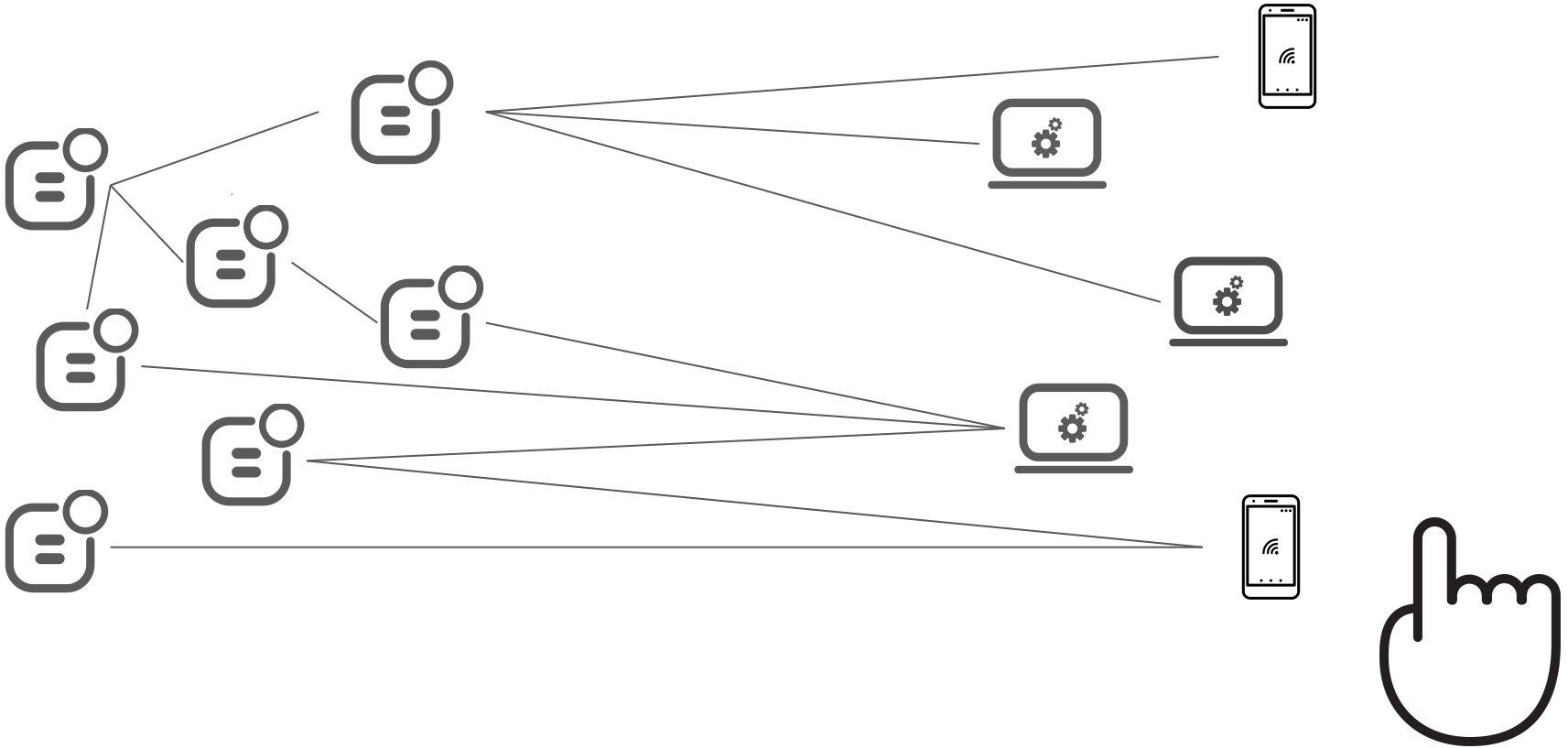



Step 1: Stabilize the Interface

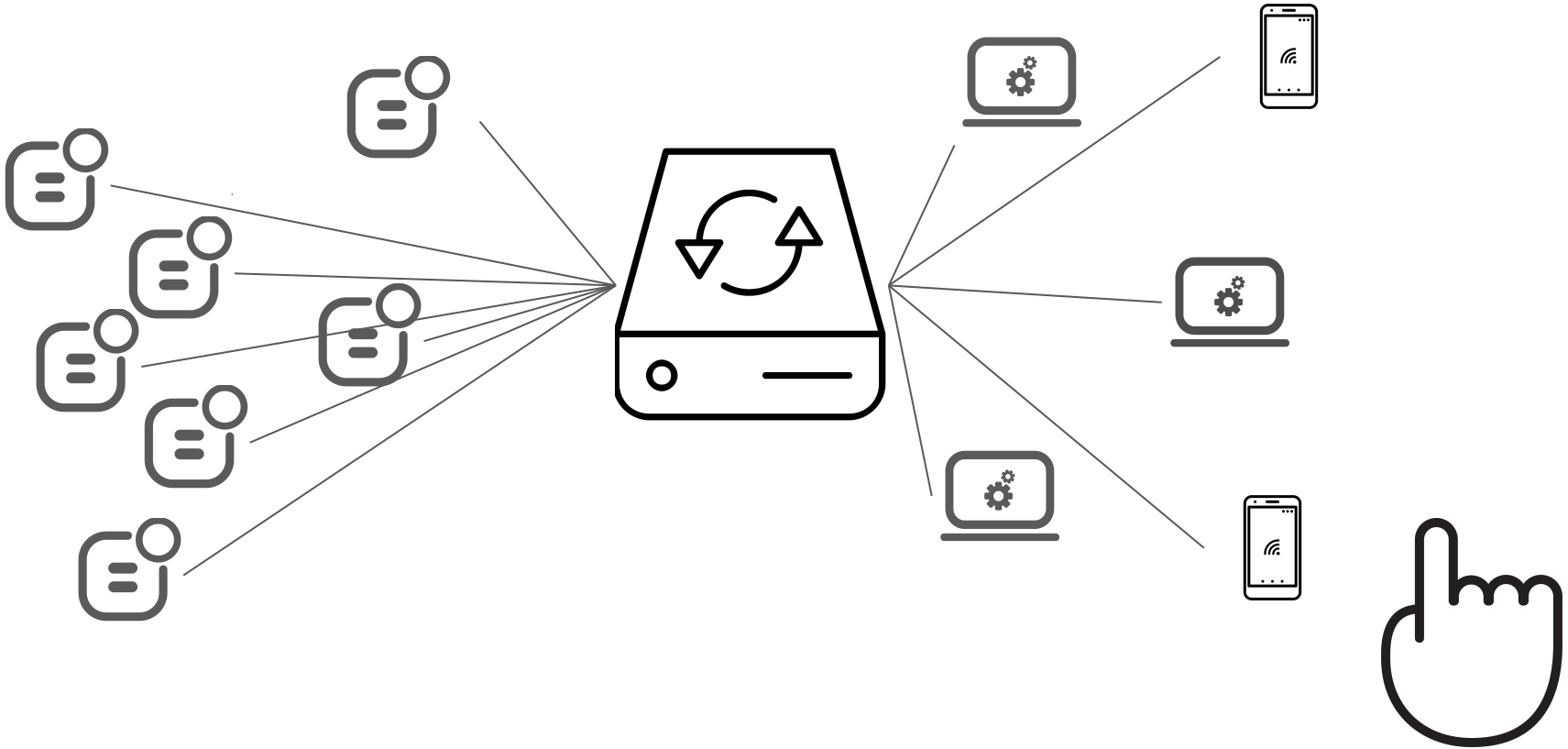
All API consumers talk to a proxy



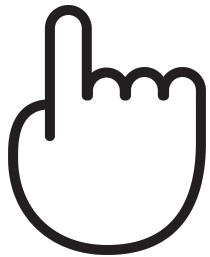
All API consumers talk to a proxy



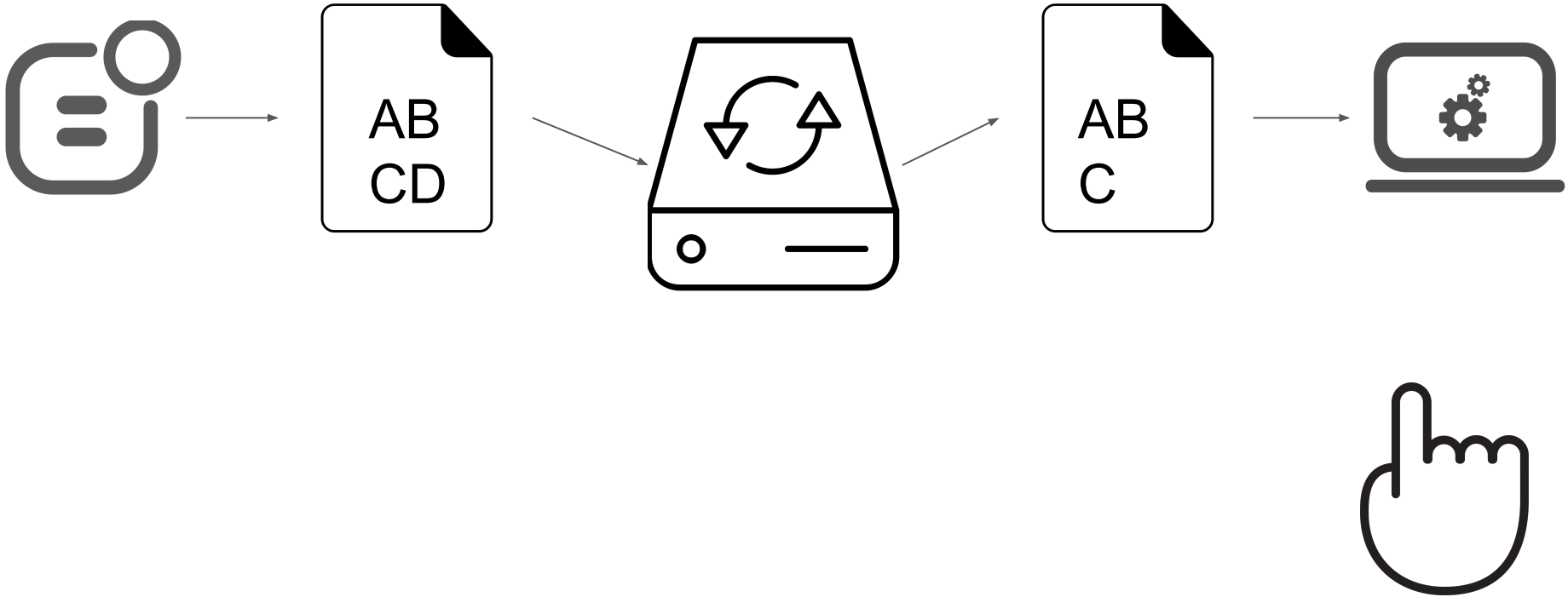
All API consumers talk to a proxy



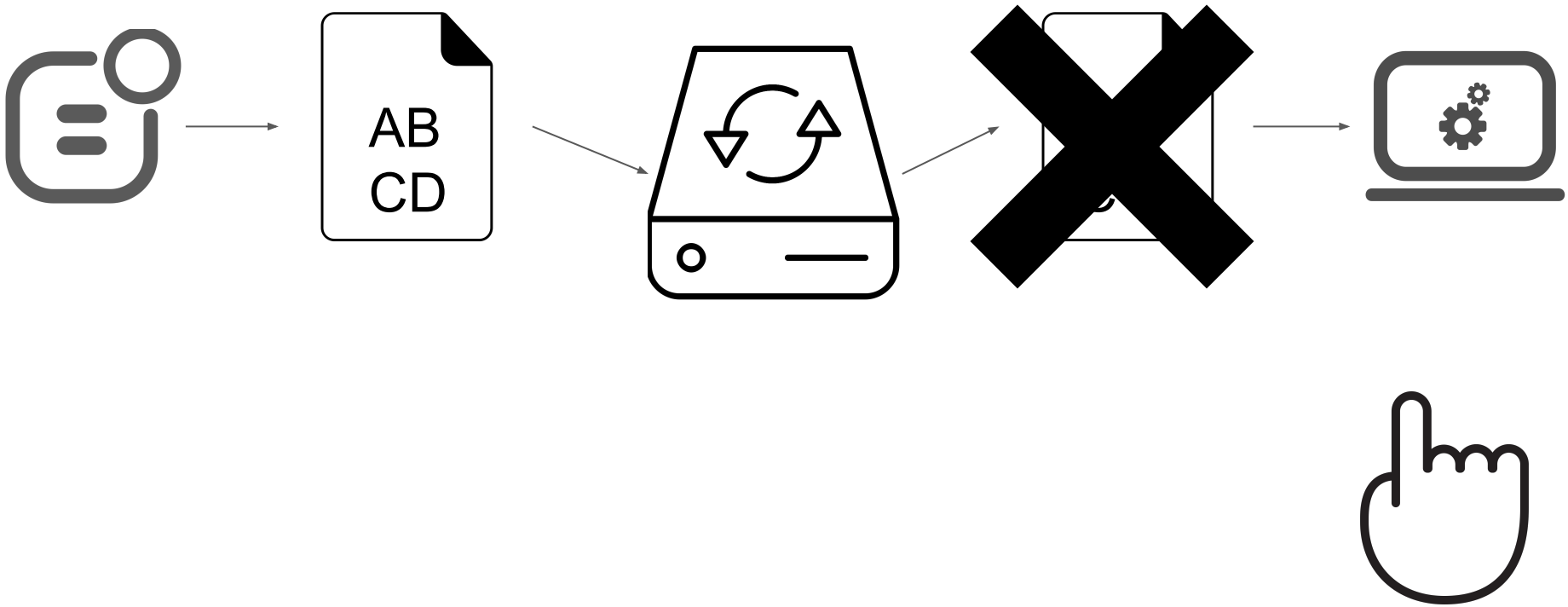
The stabilizer proxy is be pass-through only



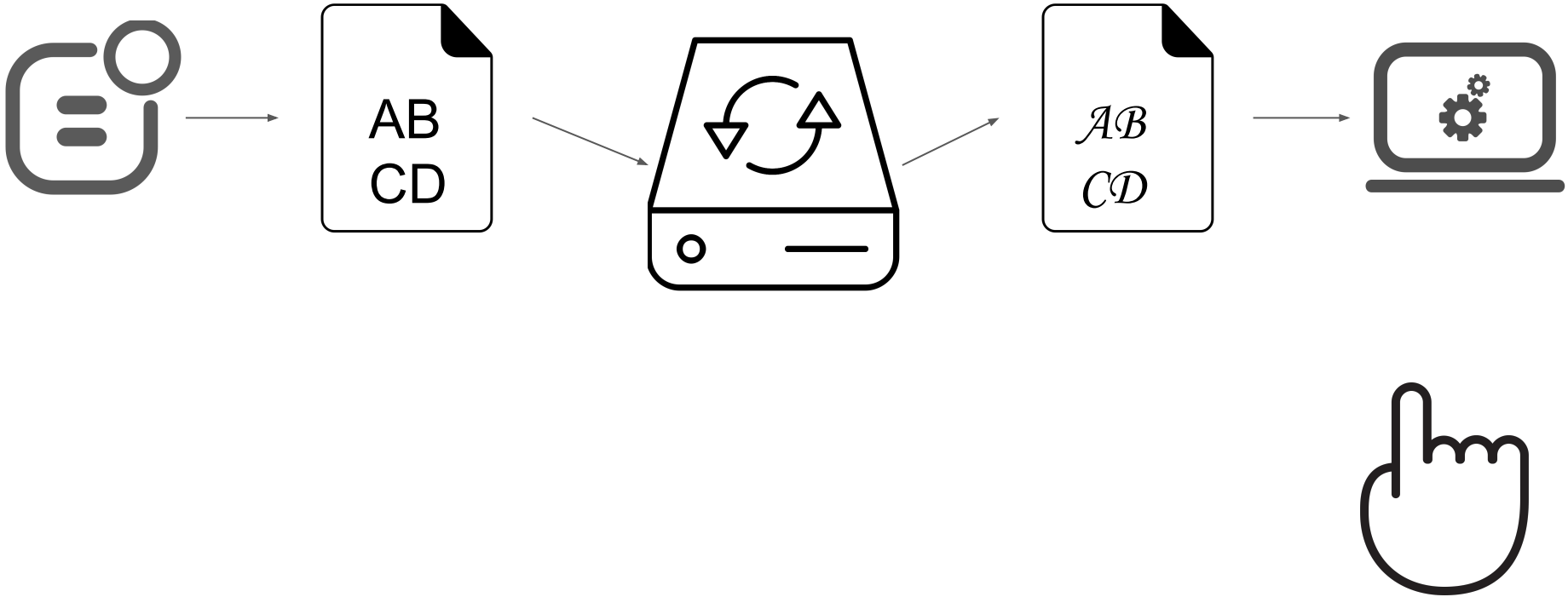
The stabilizer proxy is be pass-through only



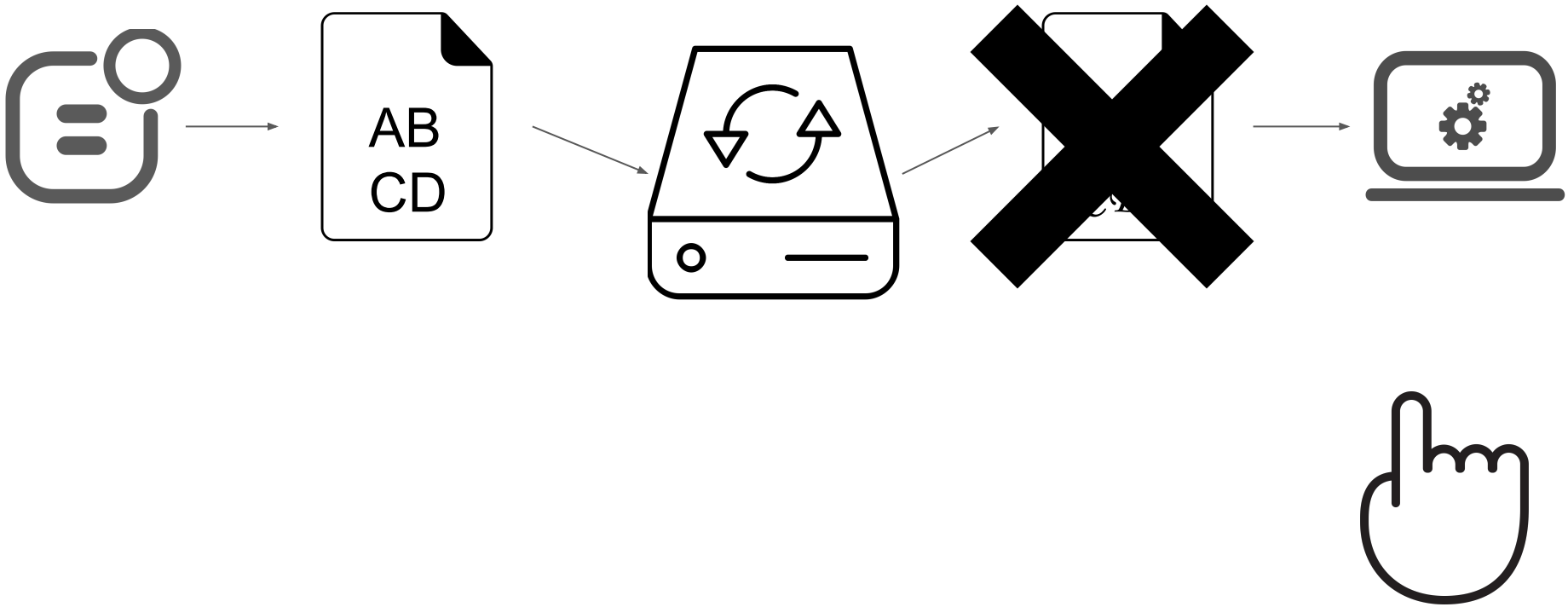
The stabilizer proxy is be pass-through only



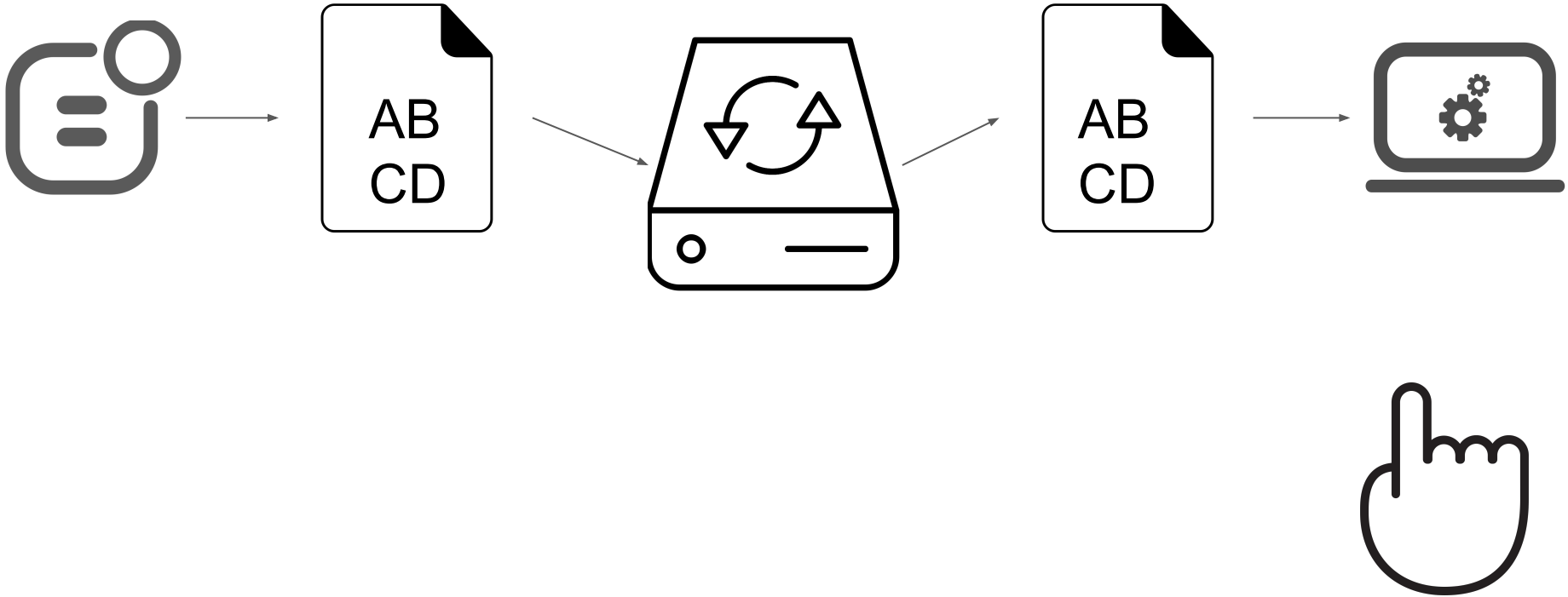
The stabilizer proxy is be pass-through only



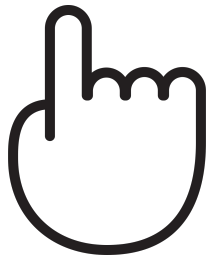
The stabilizer proxy is be pass-through only



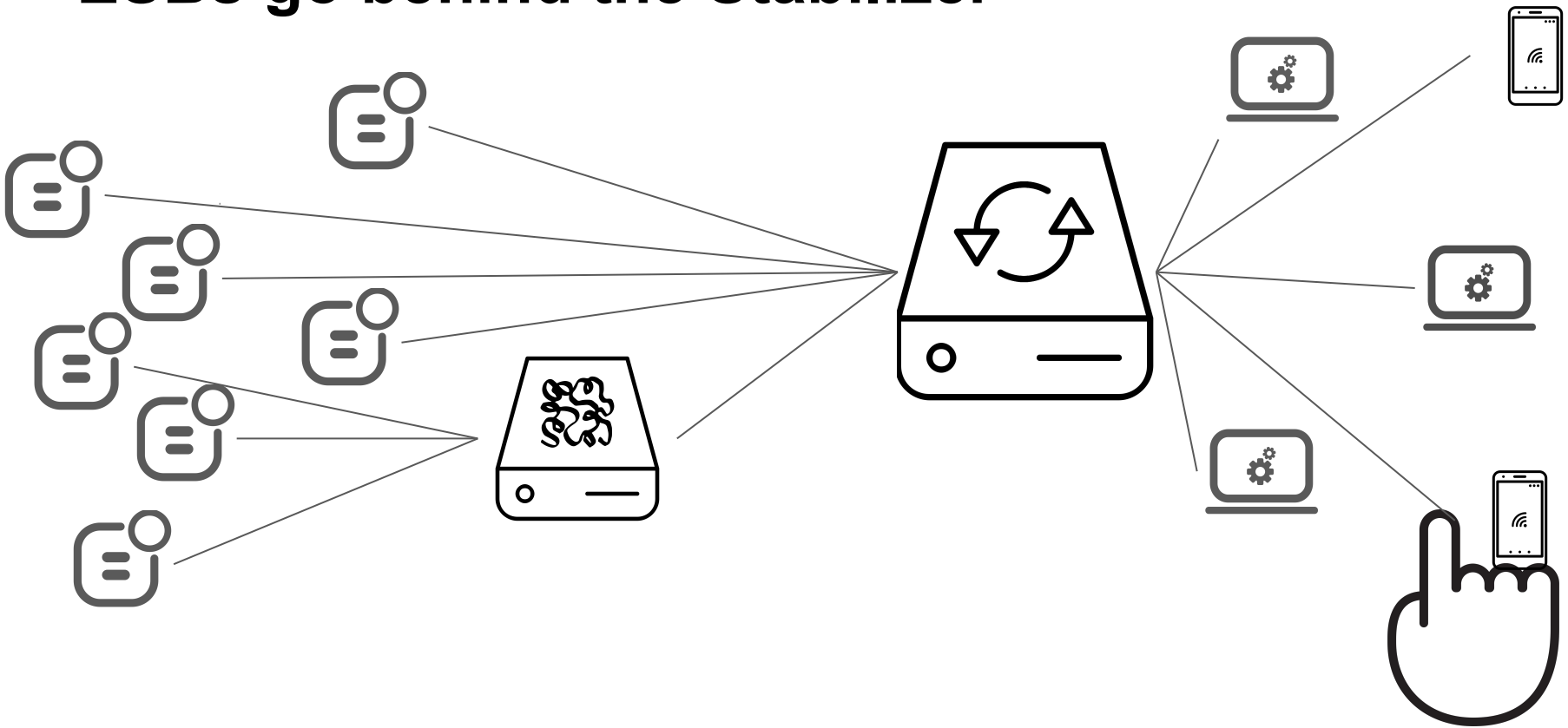
The stabilizer proxy is be pass-through only



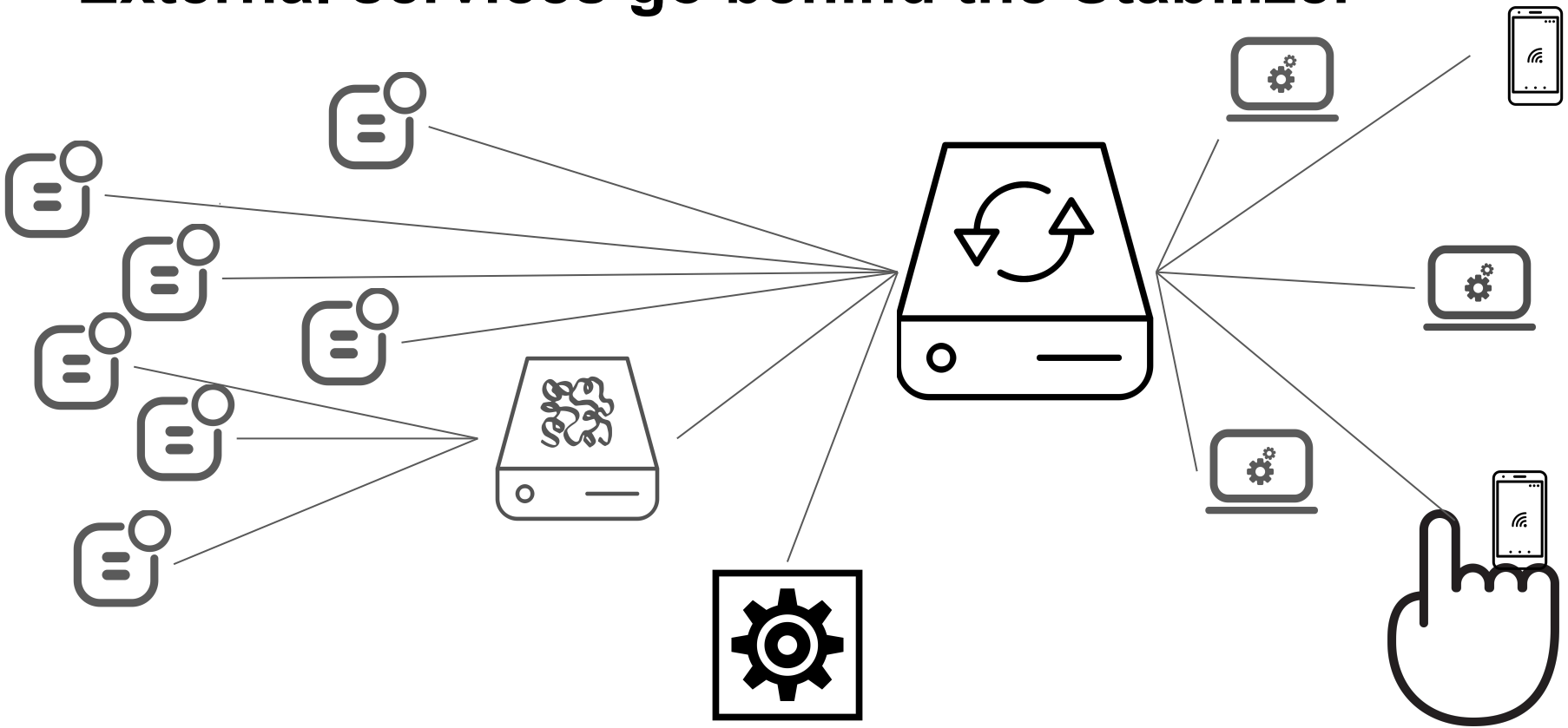
ESBs, external services go behind the Stabilizer



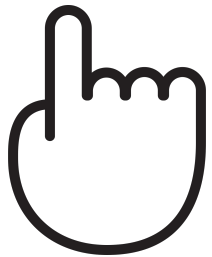
ESBs go behind the Stabilizer



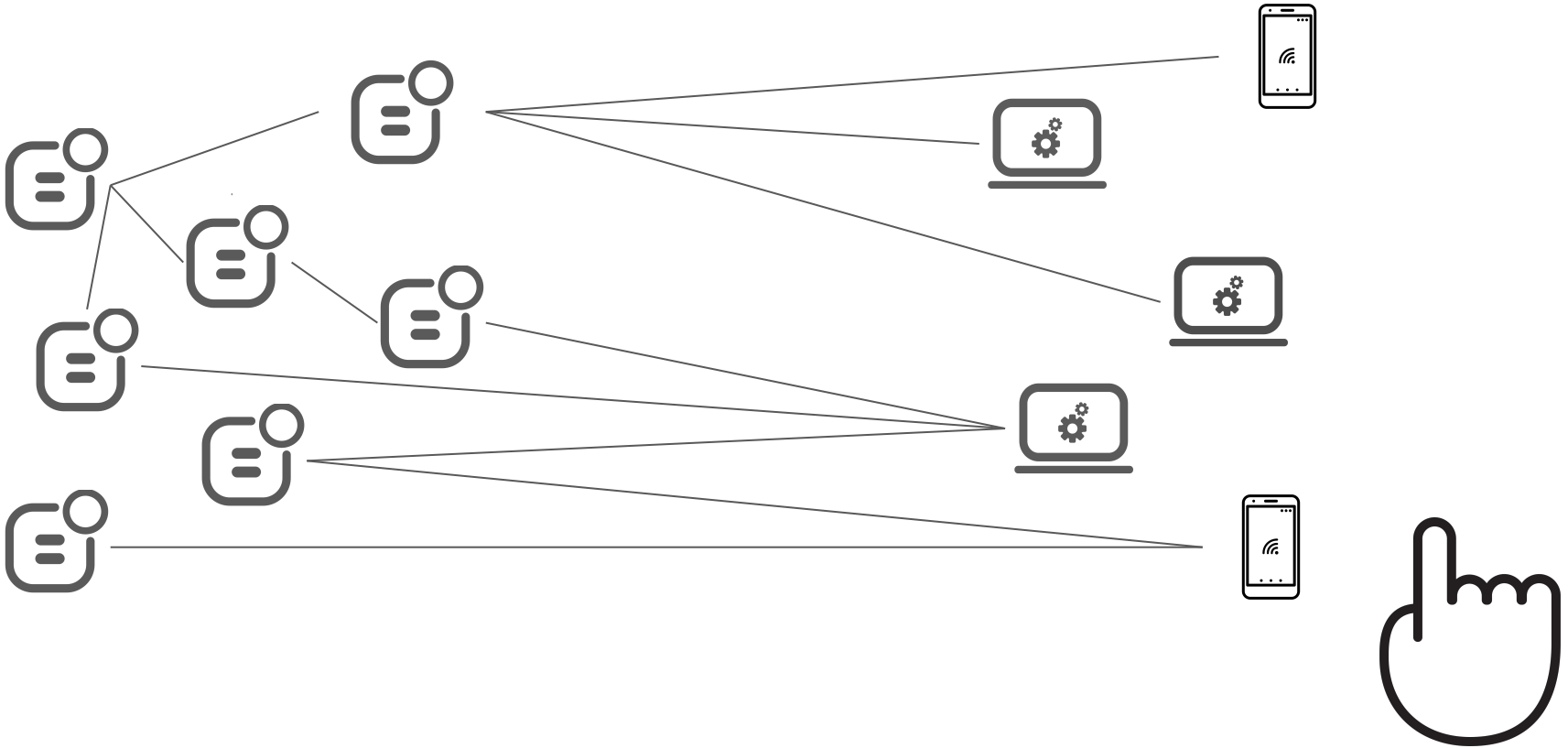
External services go behind the Stabilizer



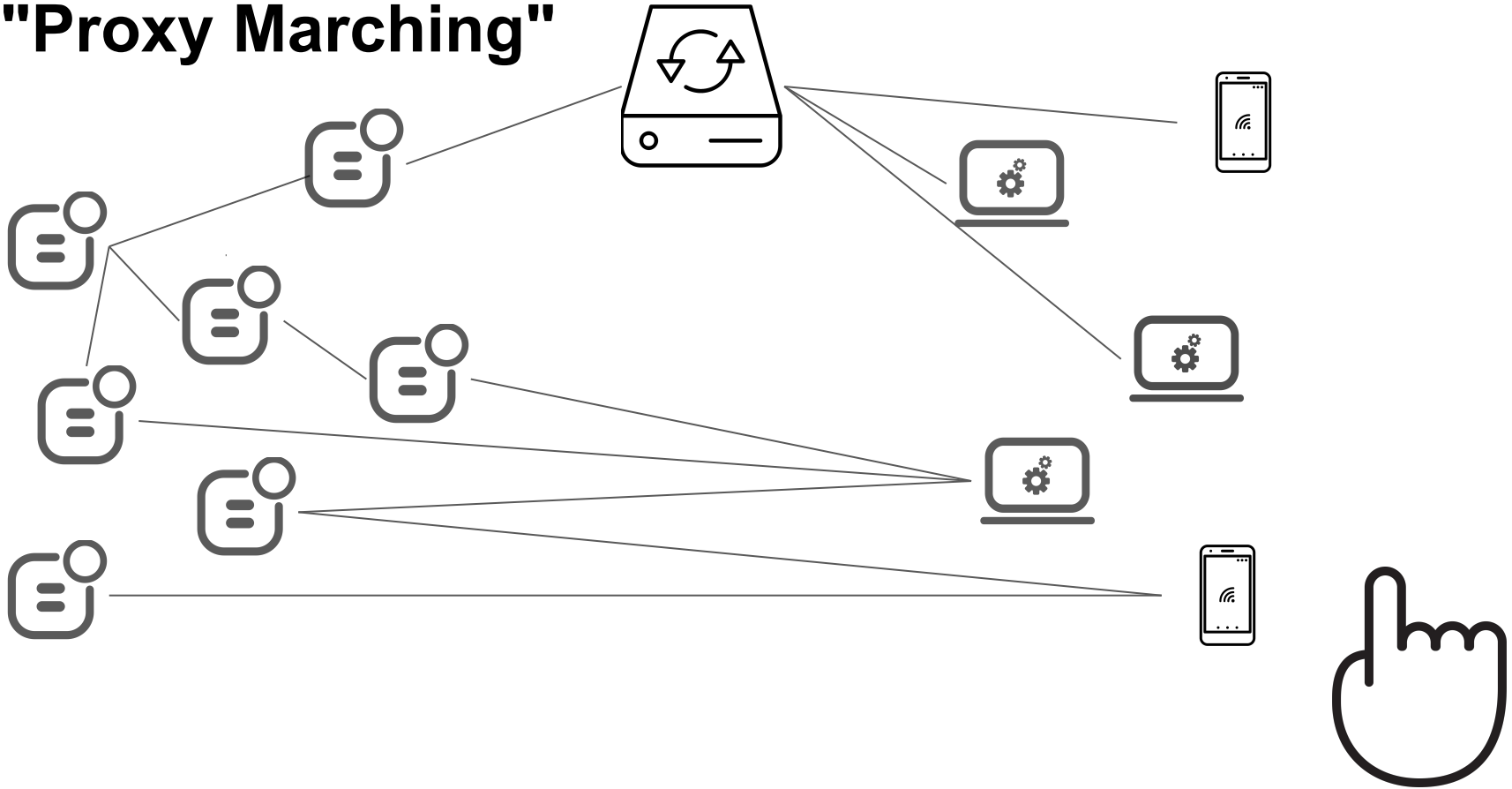
"Proxy Marching"



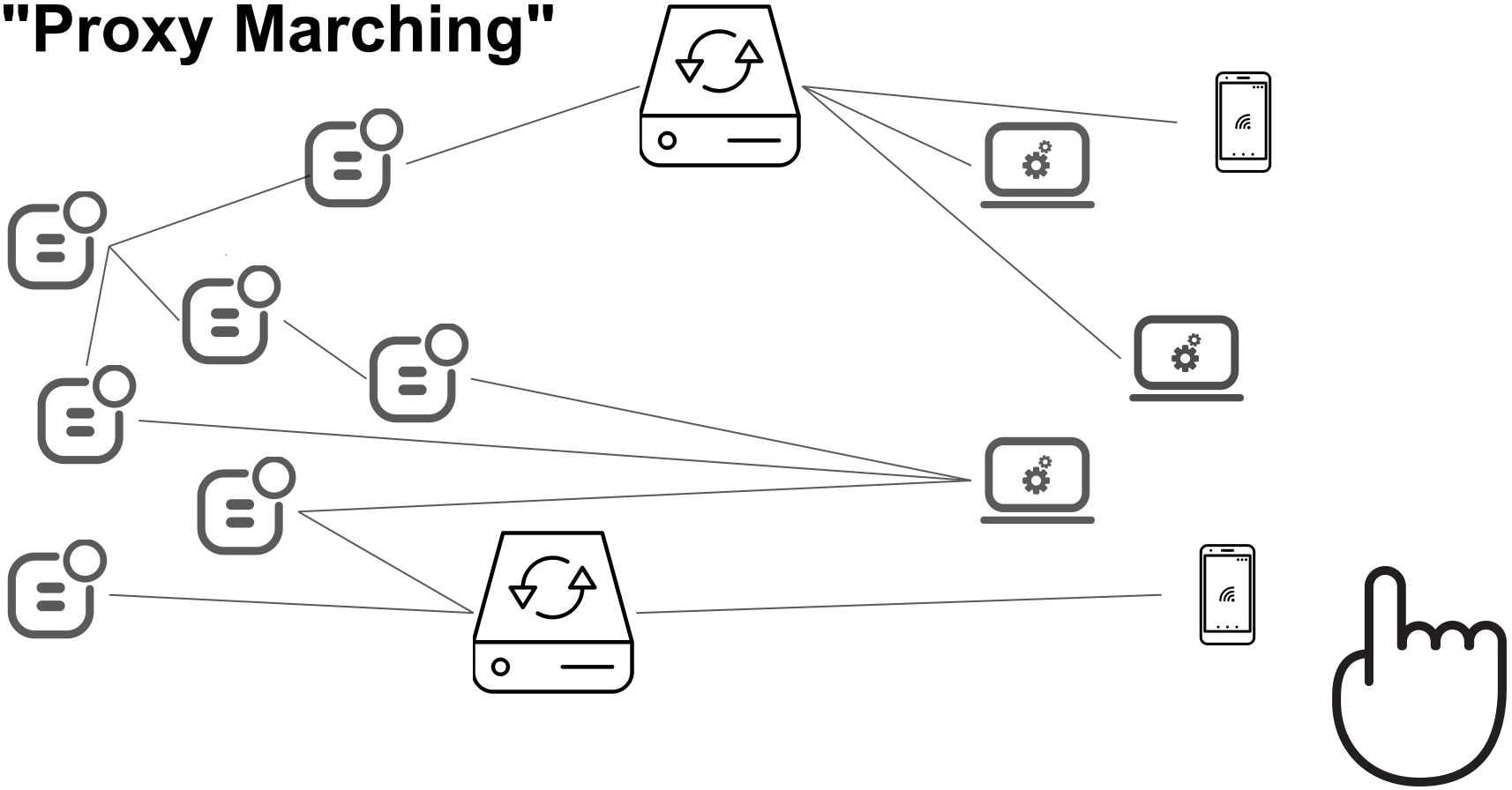
"Proxy Marching"



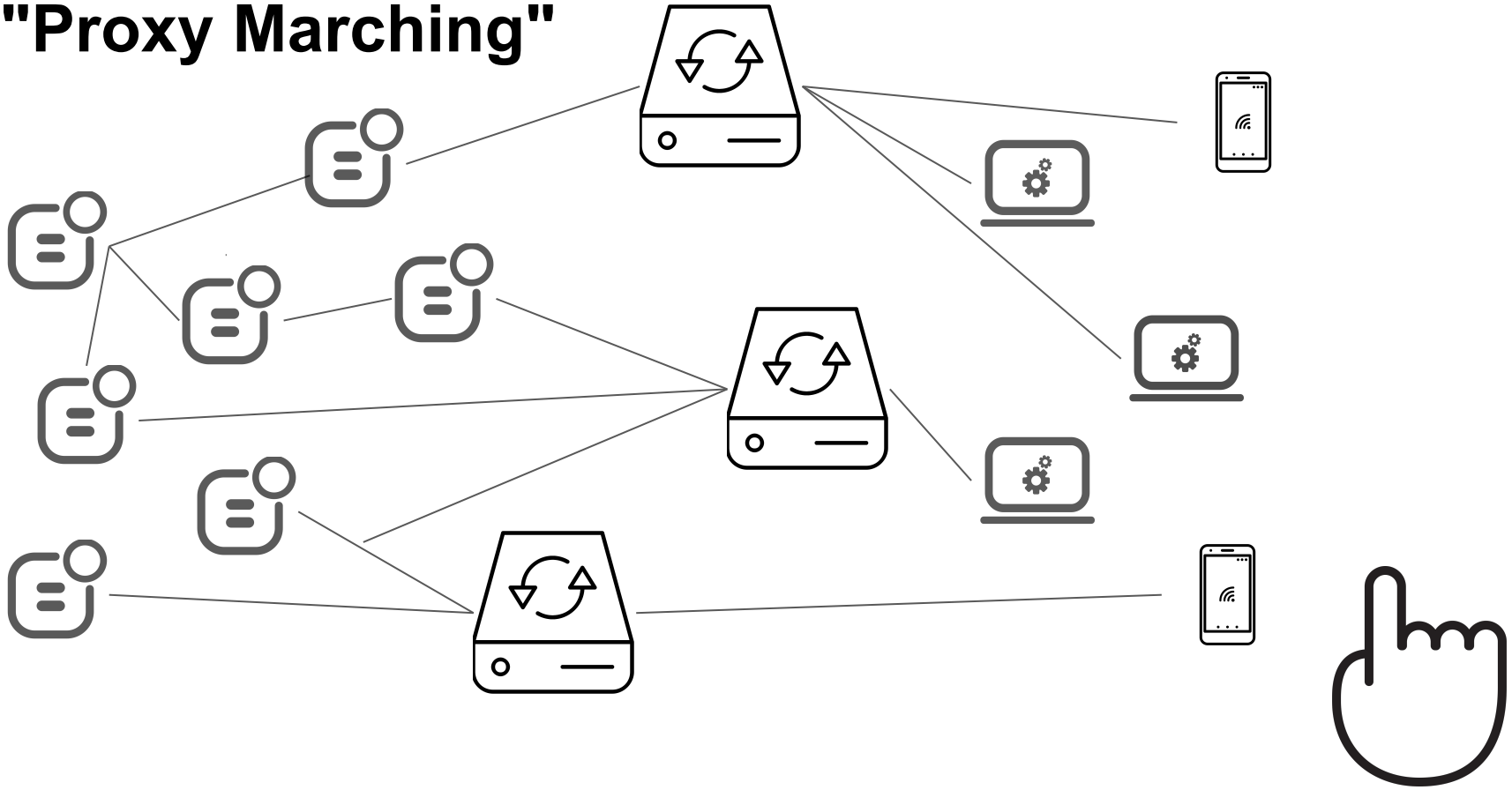
"Proxy Marching"



"Proxy Marching"

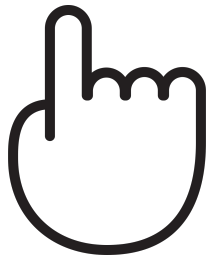


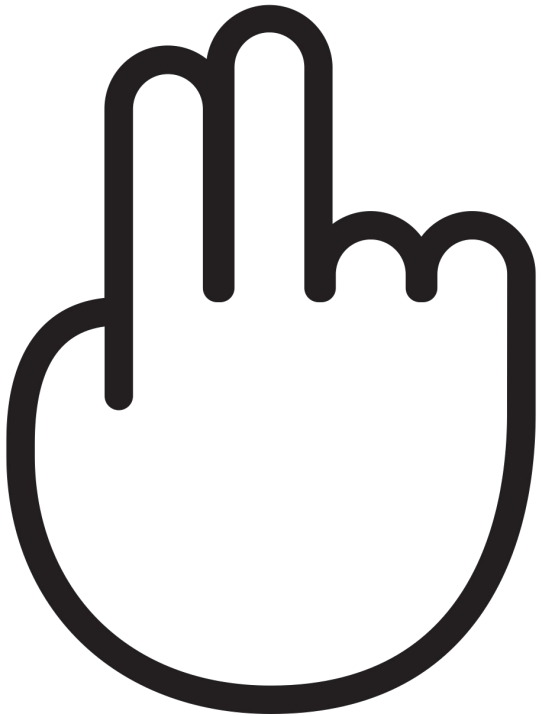
"Proxy Marching"



Stabilize the Interface

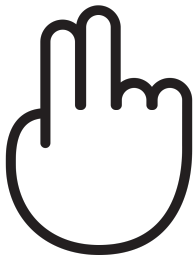
- All API consumers talk to a proxy
- The proxy MUST be pass-through only
- Keep ESBs & external services *behind* the proxy
- Employ a "Proxy March"



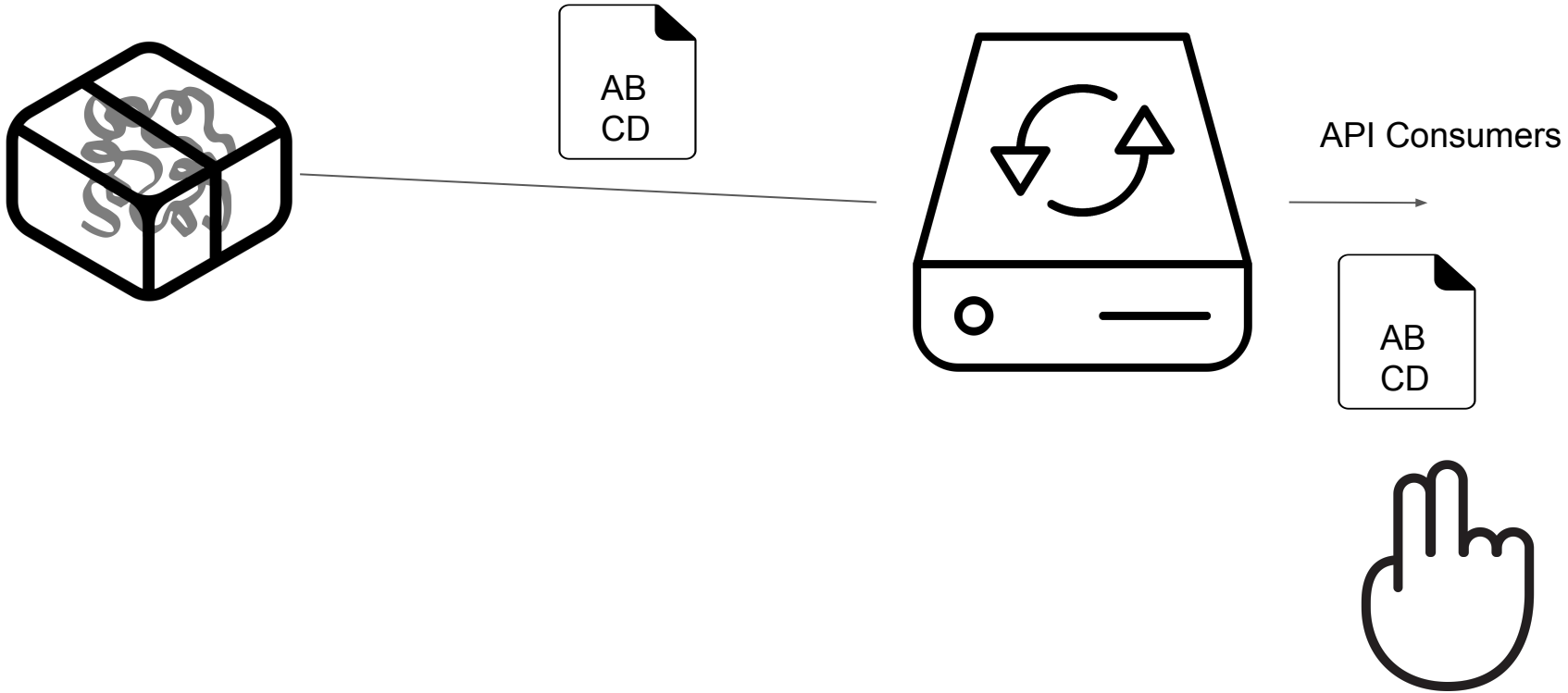


Step 2: Transform the Implementation

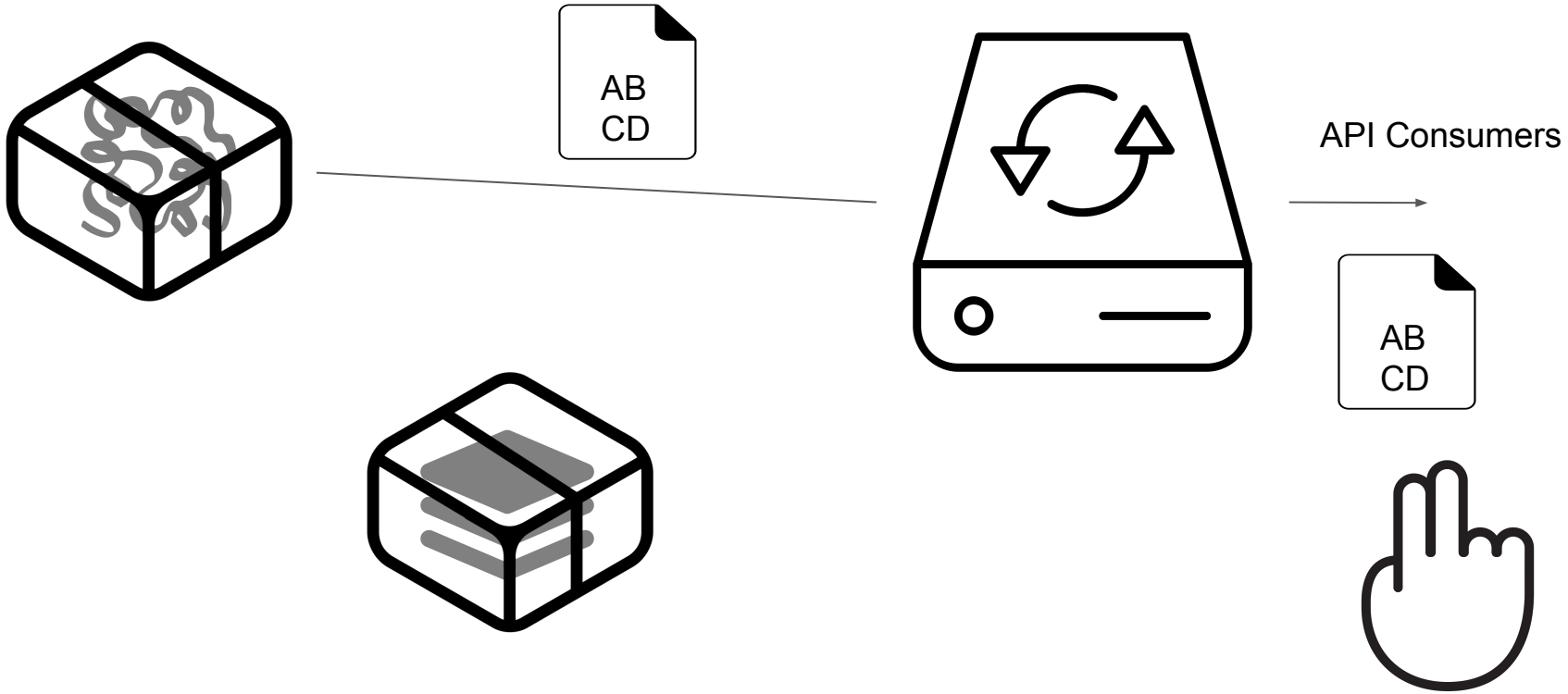
Refactor existing components



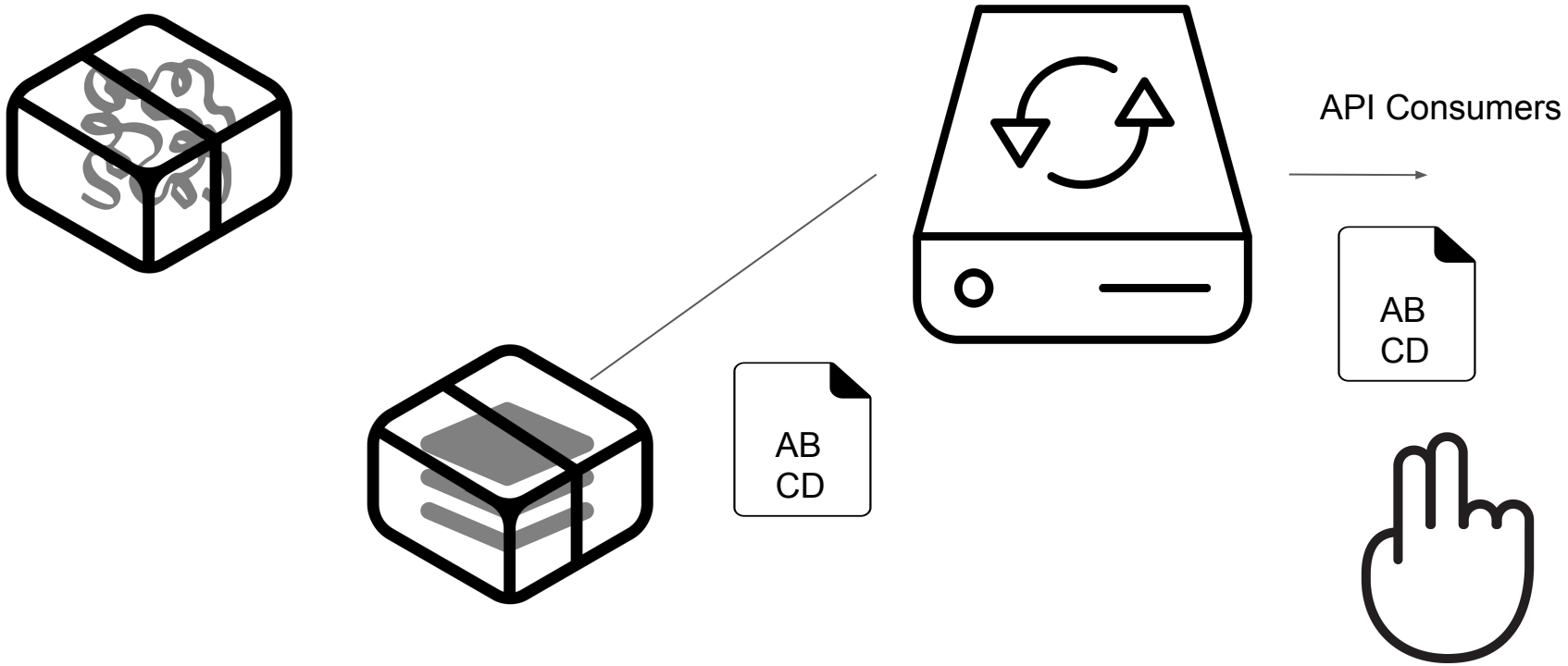
Refactor existing components



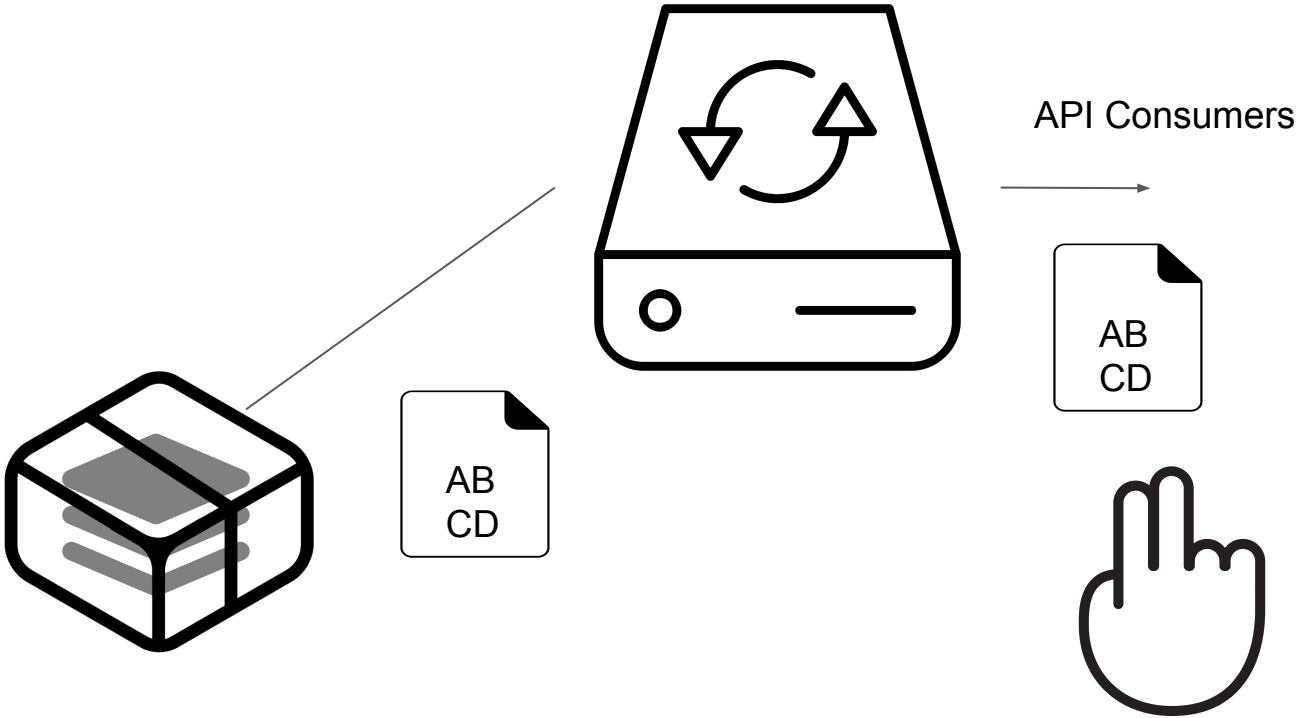
Refactor existing components



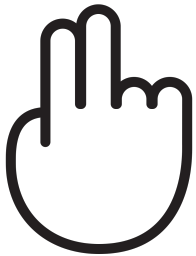
Refactor existing components



Refactor existing components

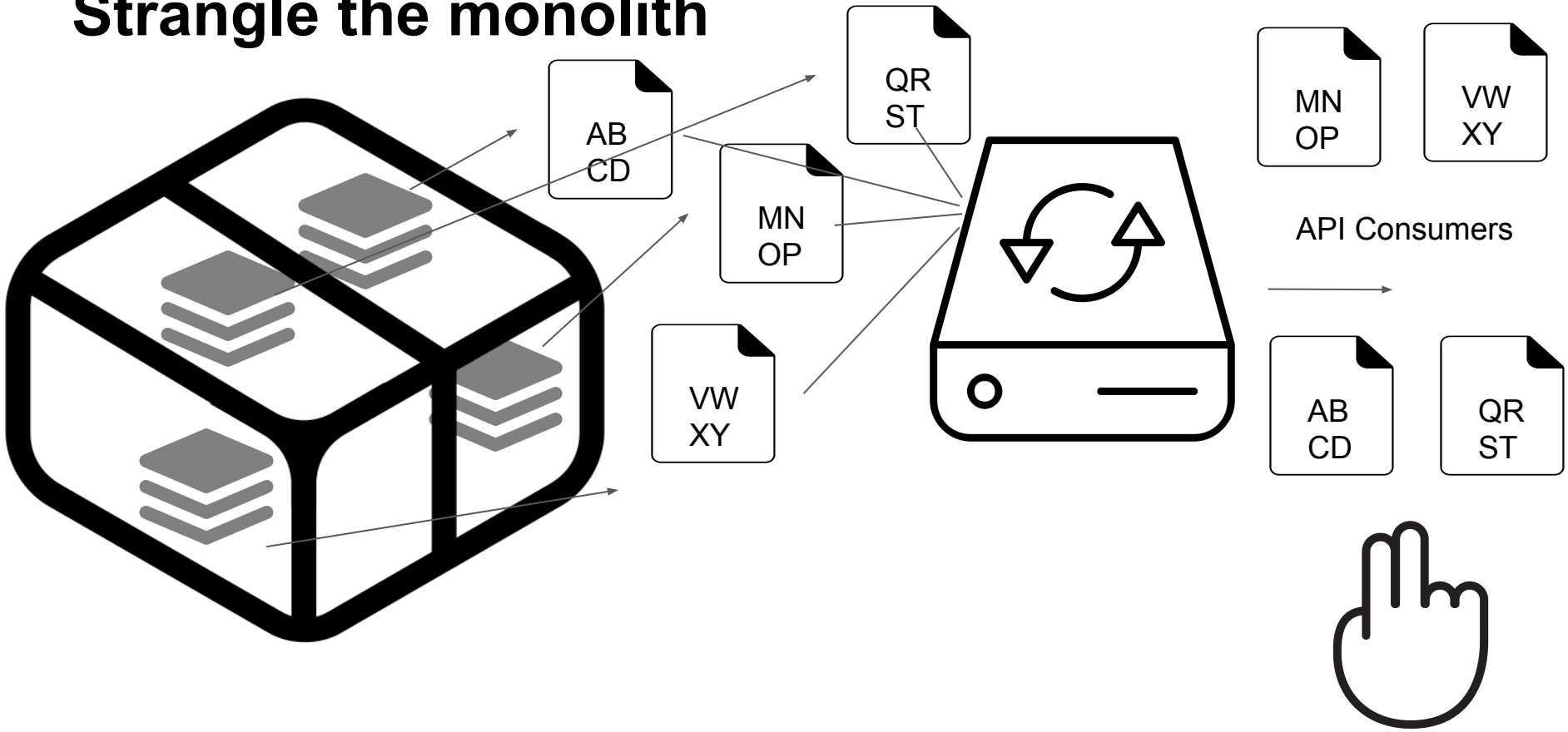


Strangle the monolith

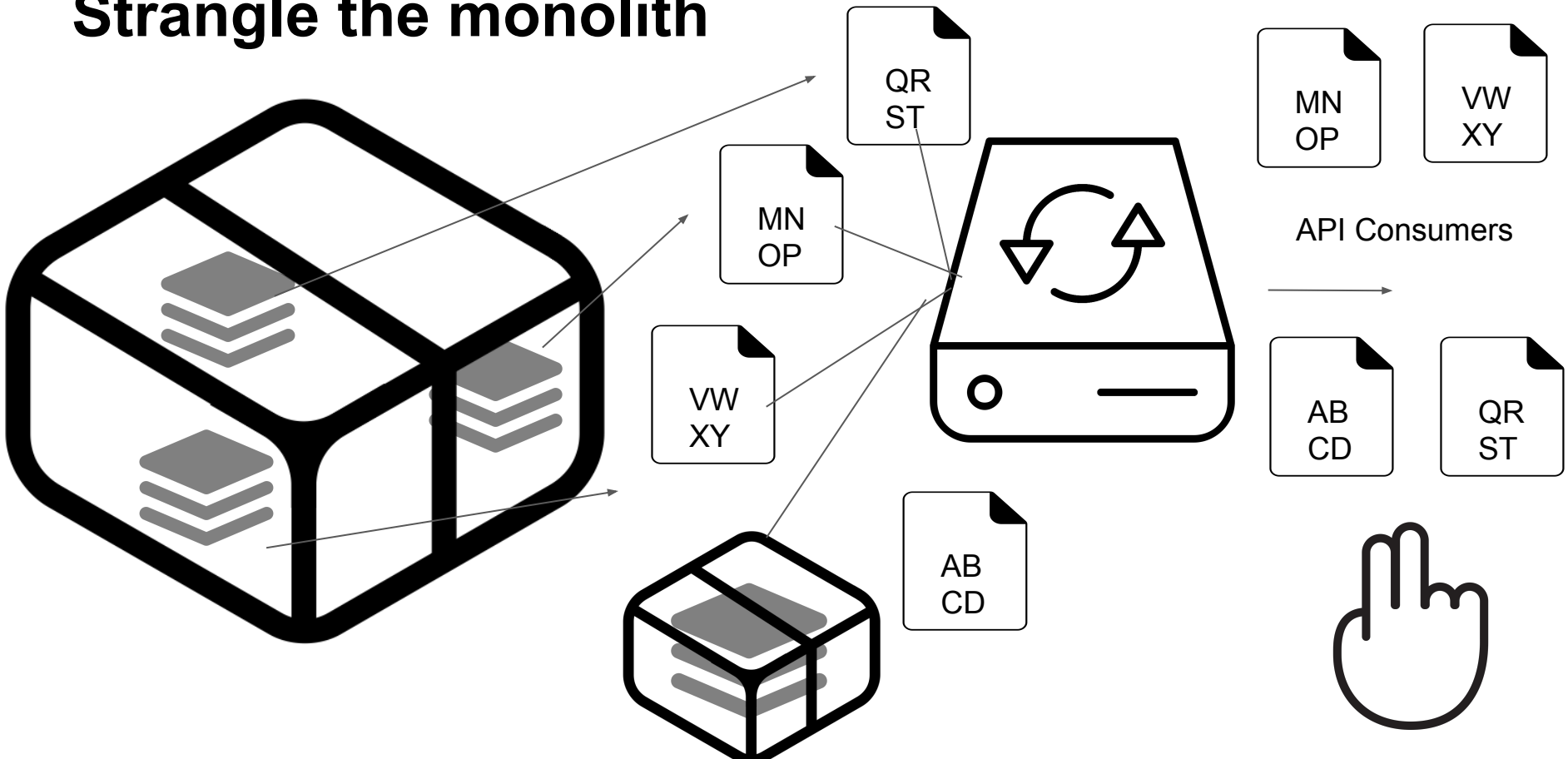


Step 2: Transform the Implementation

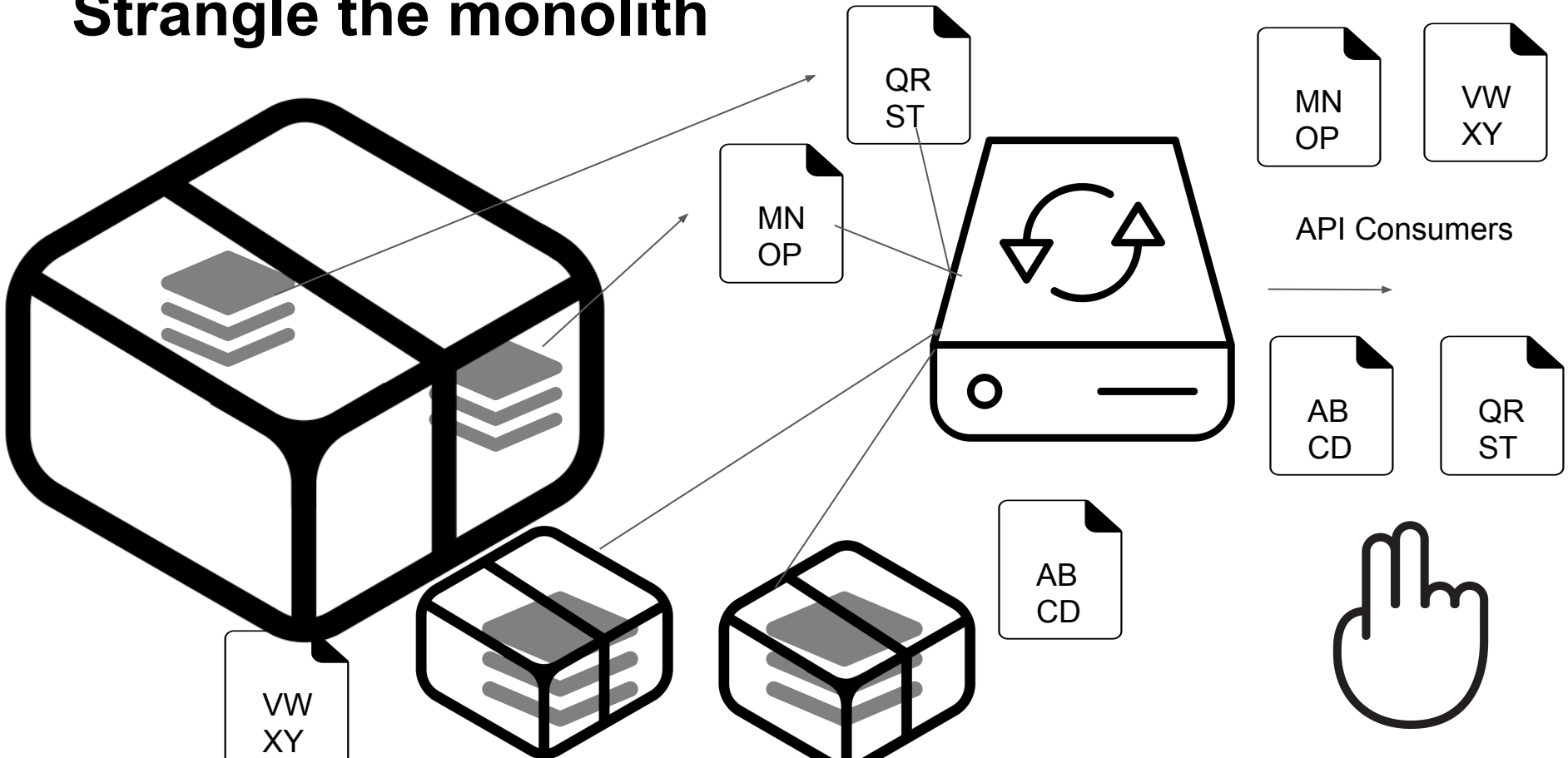
Strangle the monolith



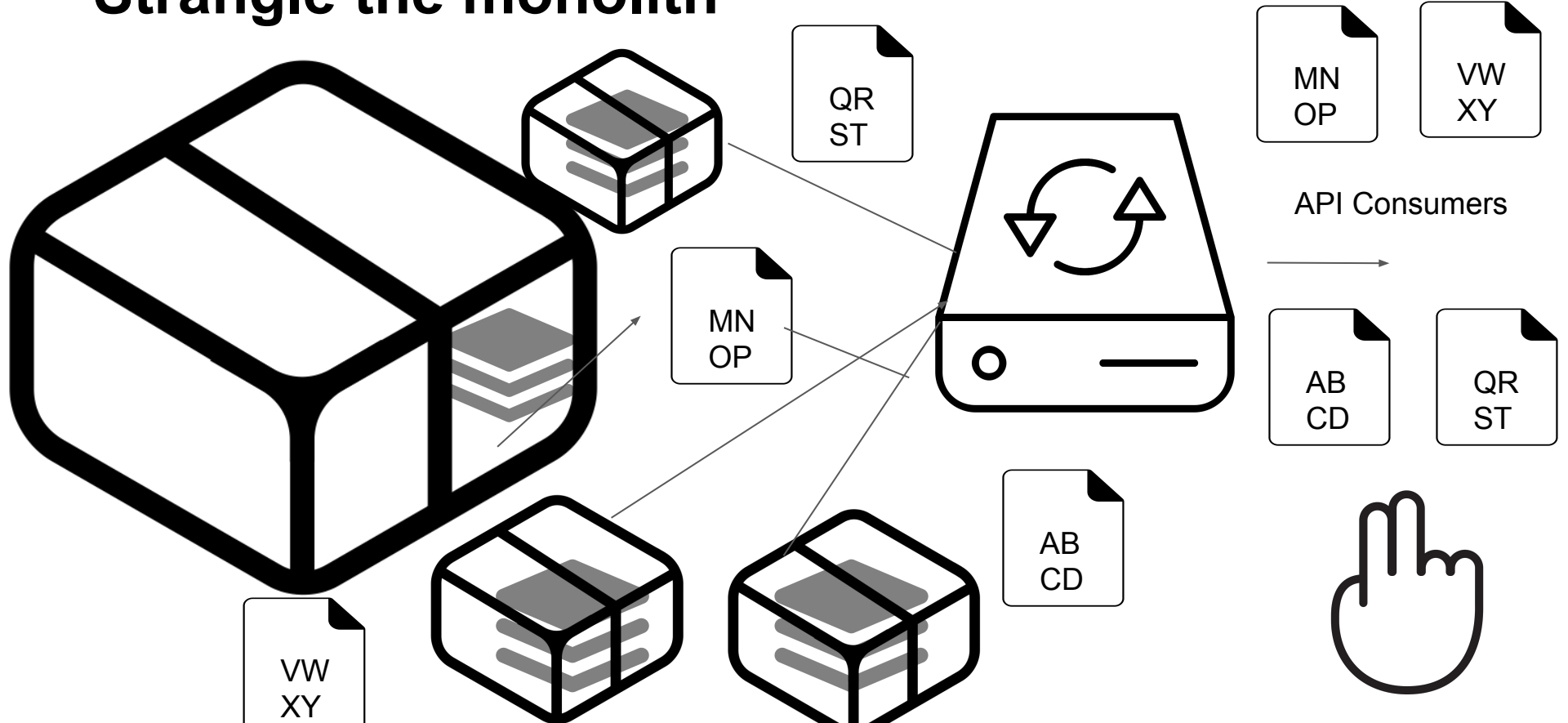
Strangle the monolith



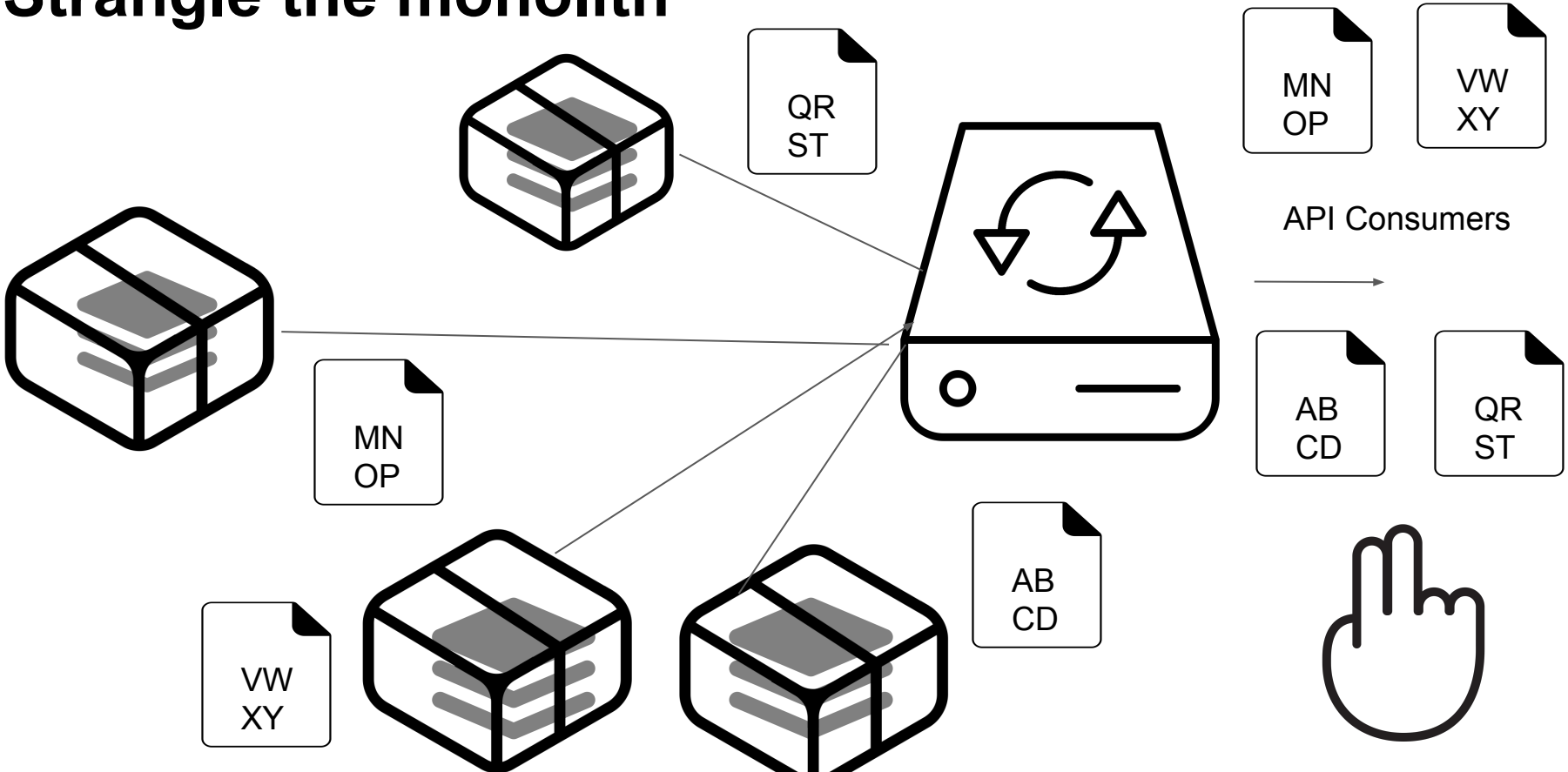
Strangle the monolith



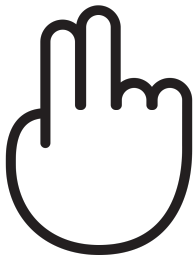
Strangle the monolith



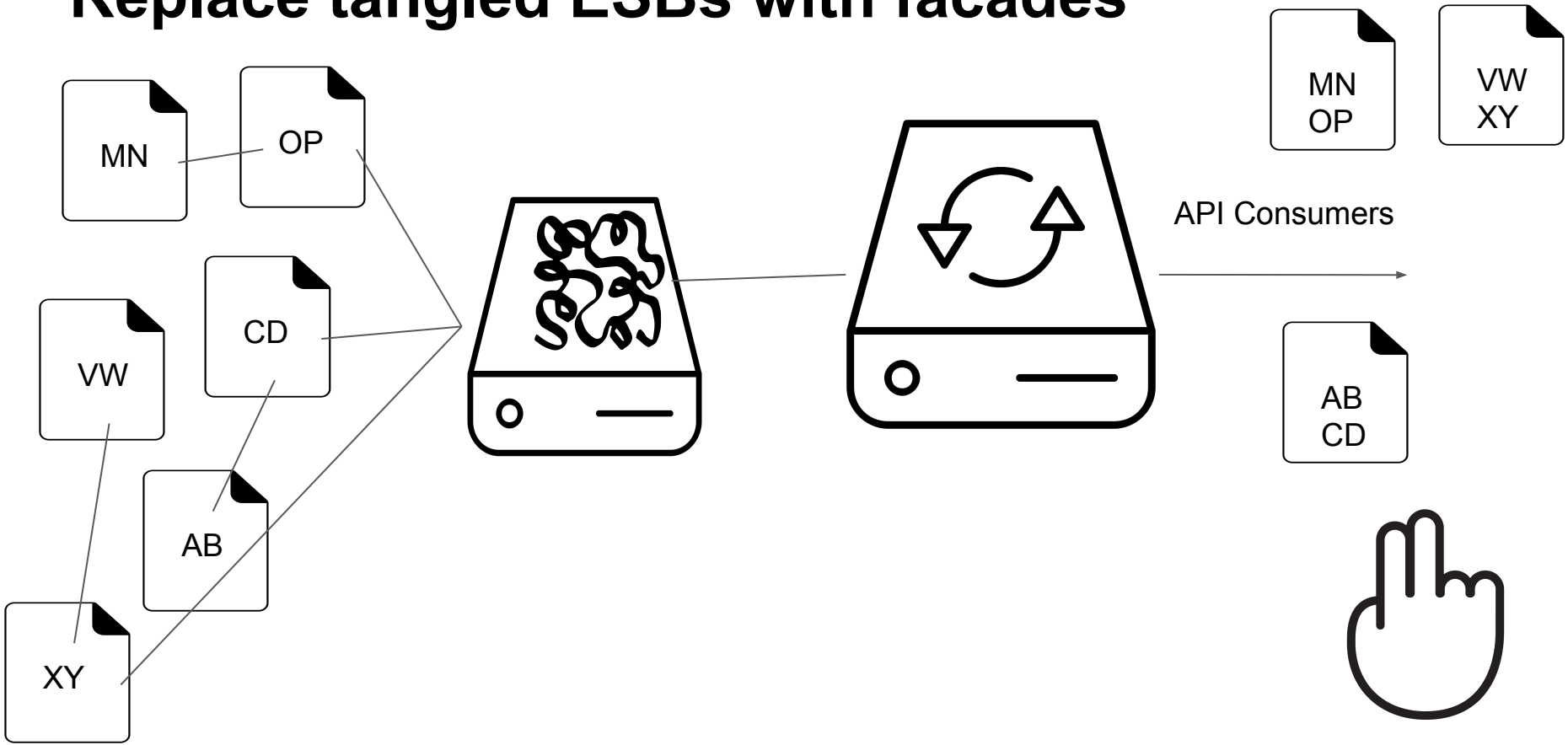
Strangle the monolith



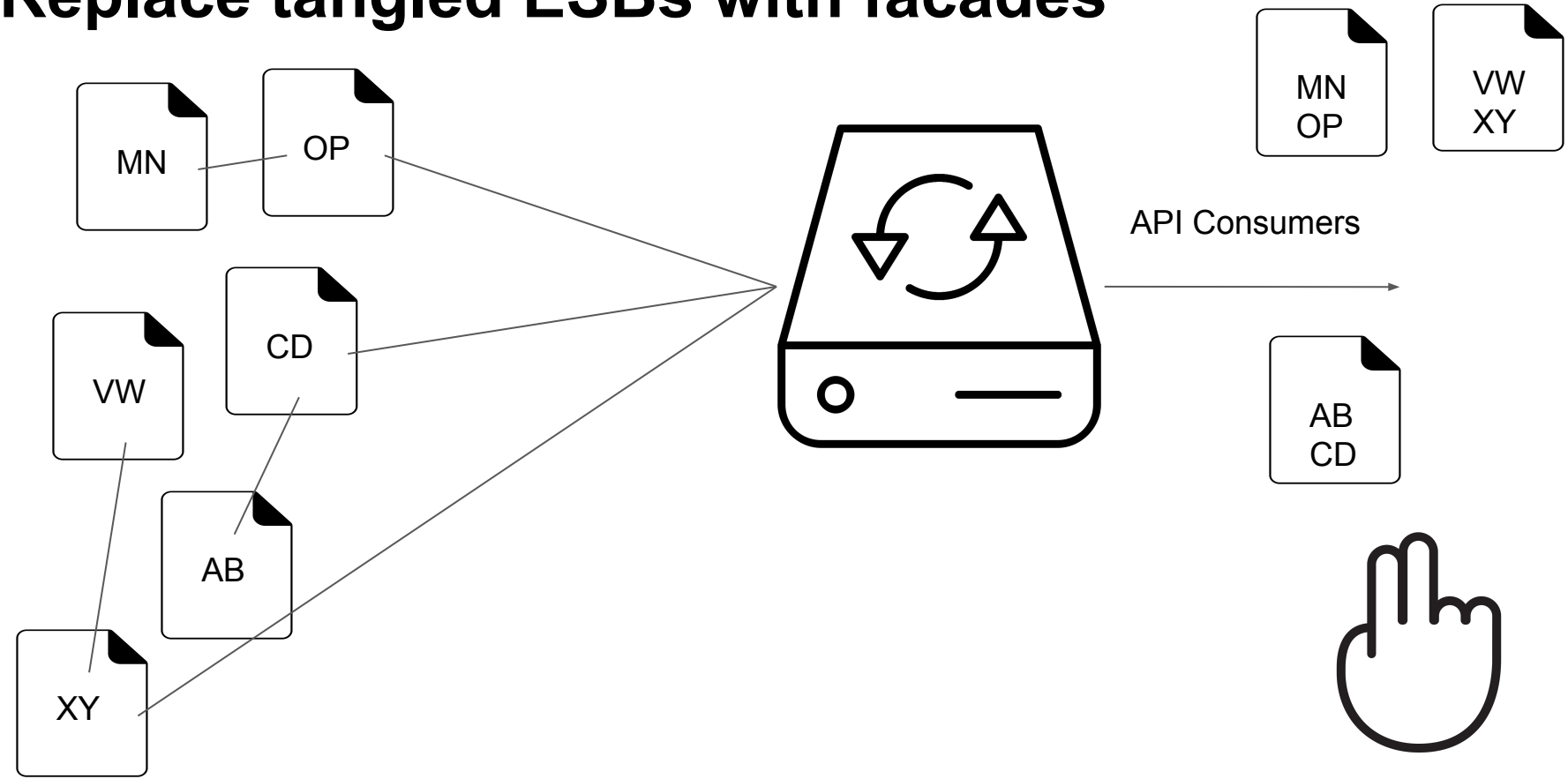
Replace tangled ESBs with facades



Replace tangled ESBs with facades

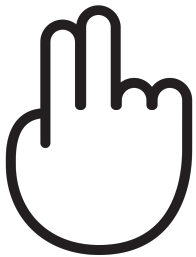


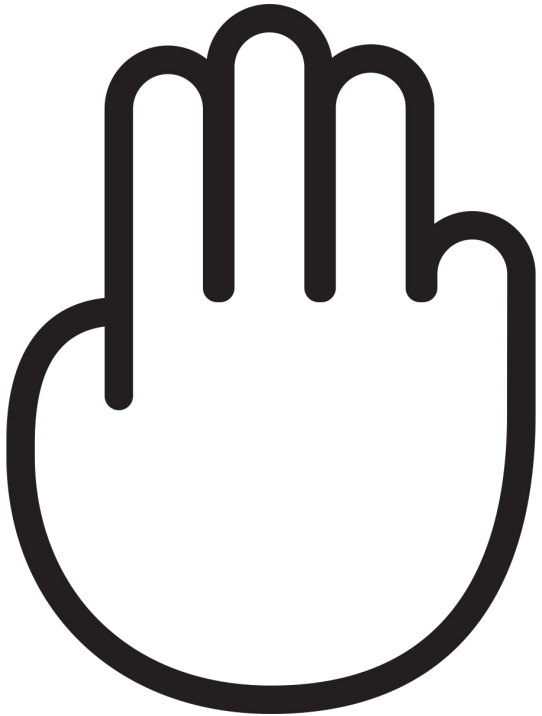
Replace tangled ESBs with facades

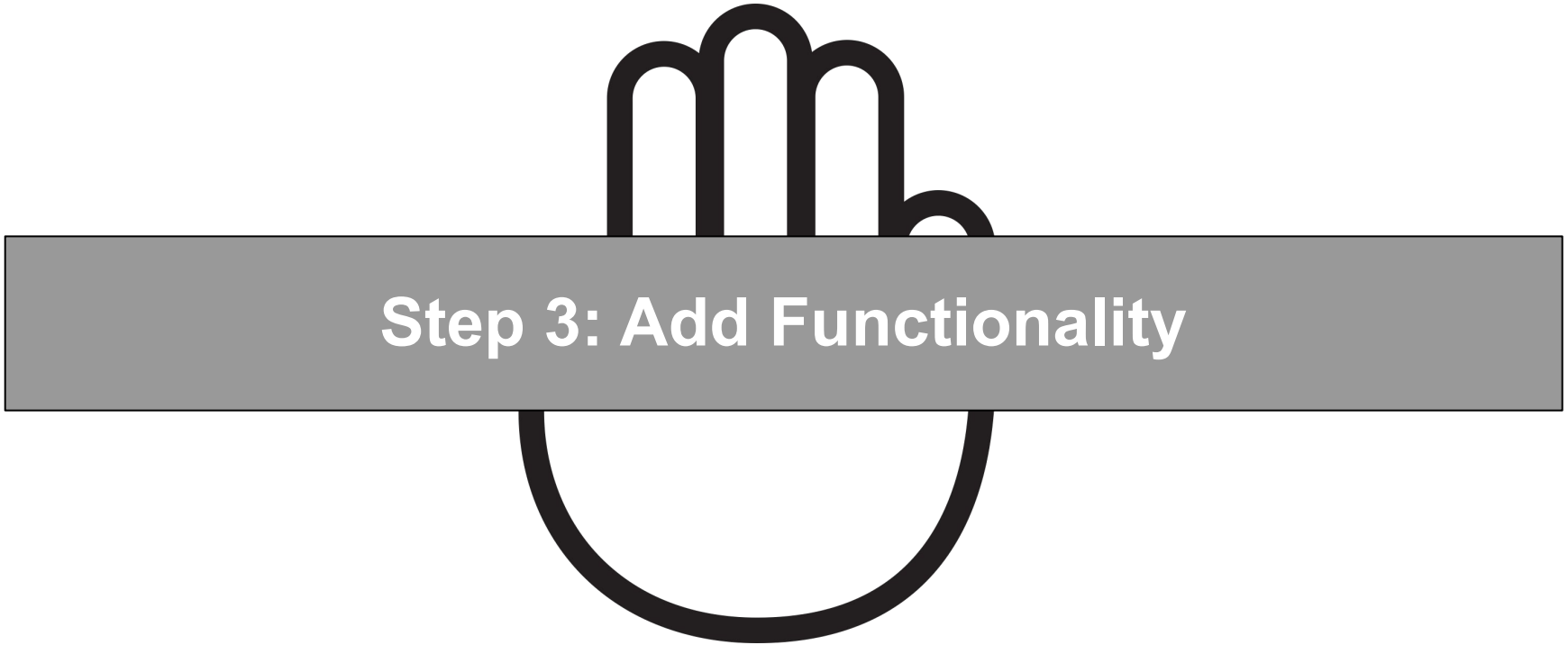


Transform the Implementation

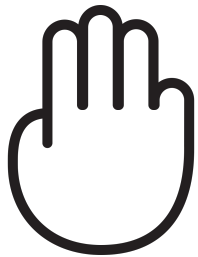
- Refactor existing components
- Strangle the monolith
- Replace tangled ESBs



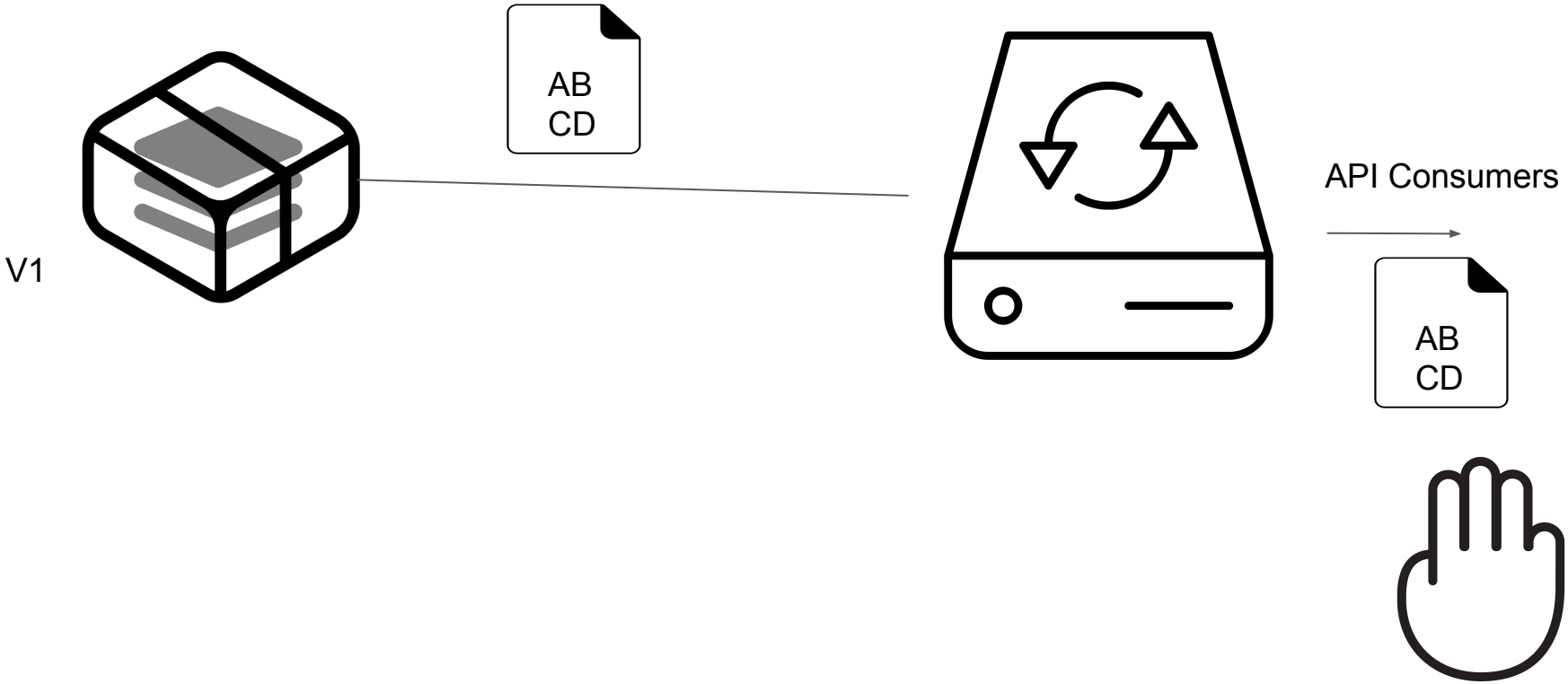



Step 3: Add Functionality

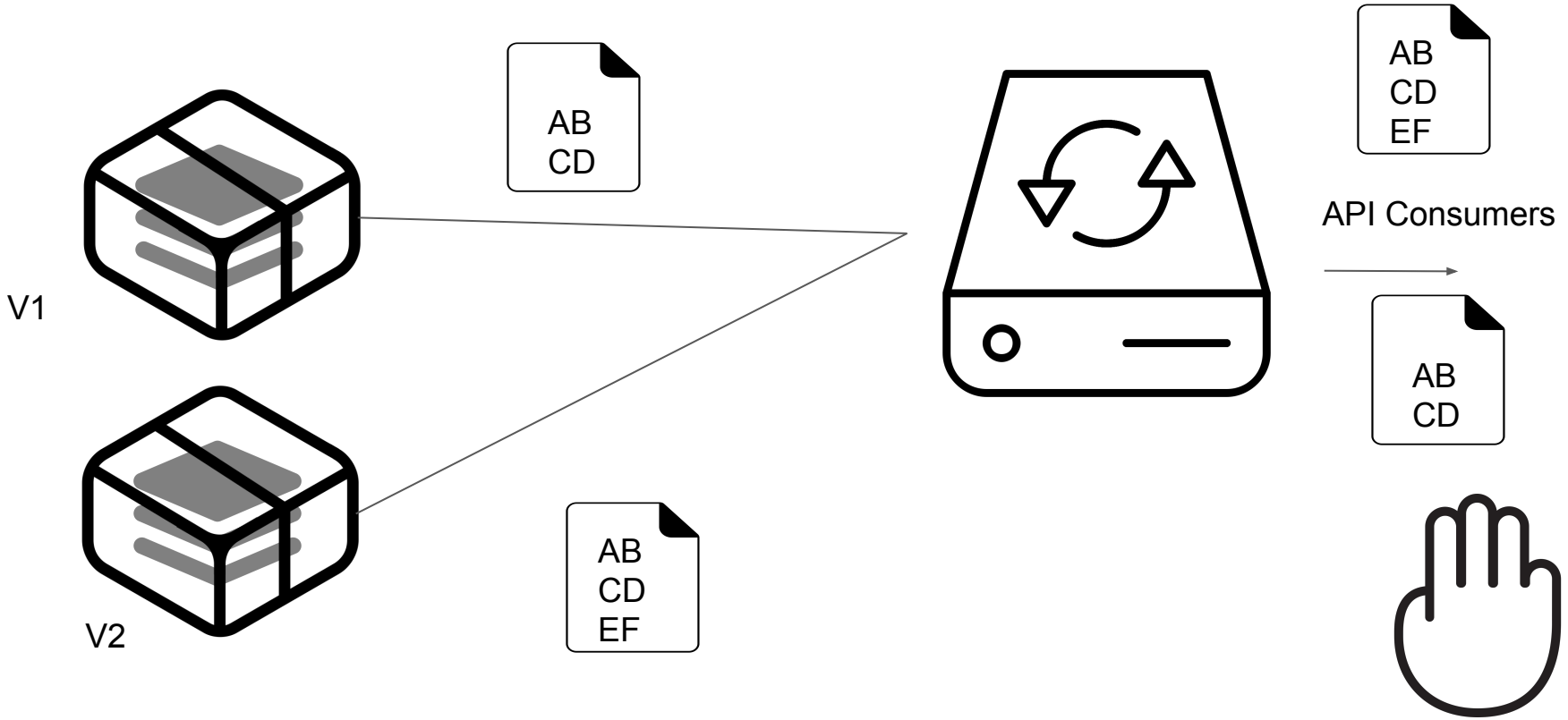
Update functionality via side-by-side components



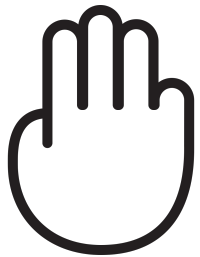
Update functionality via side-by-side components



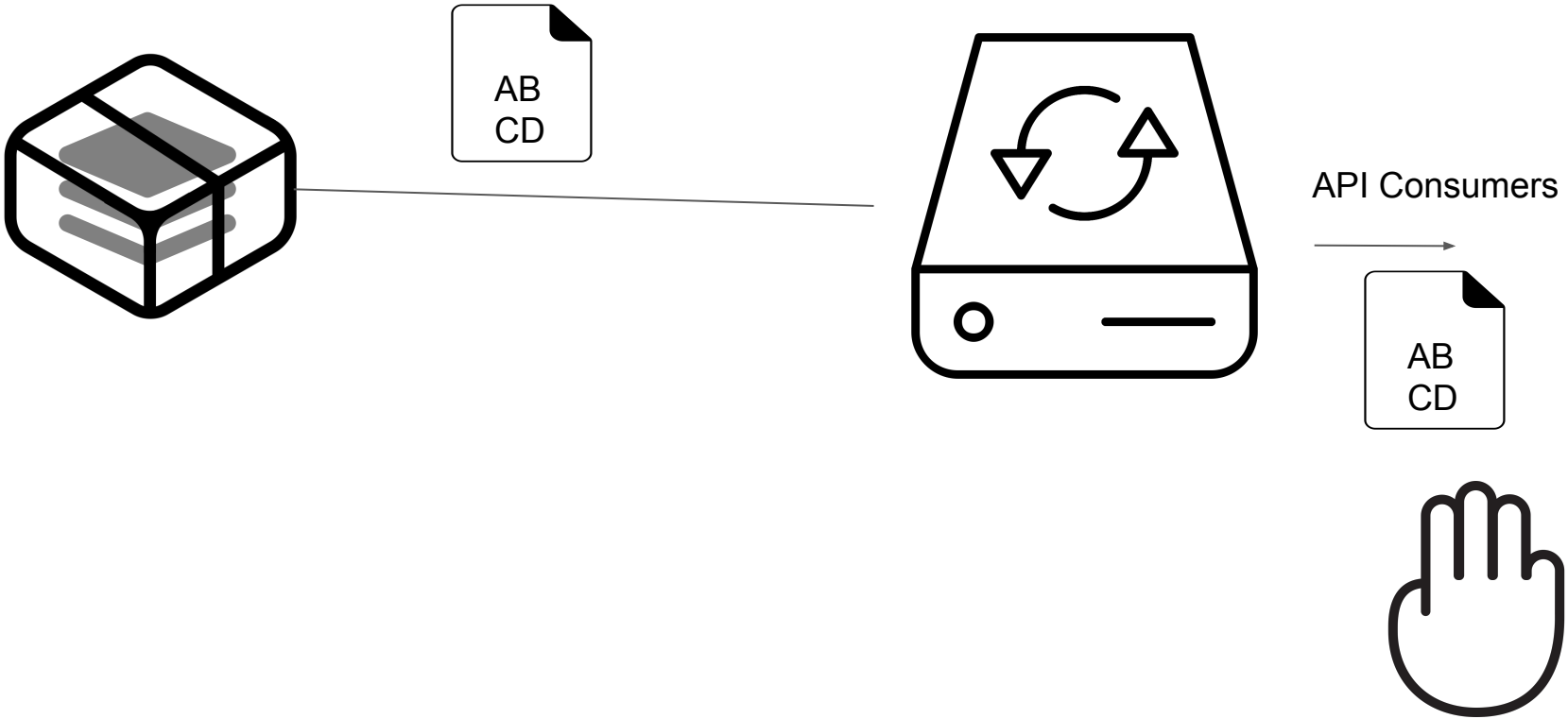
Update functionality via side-by-side components



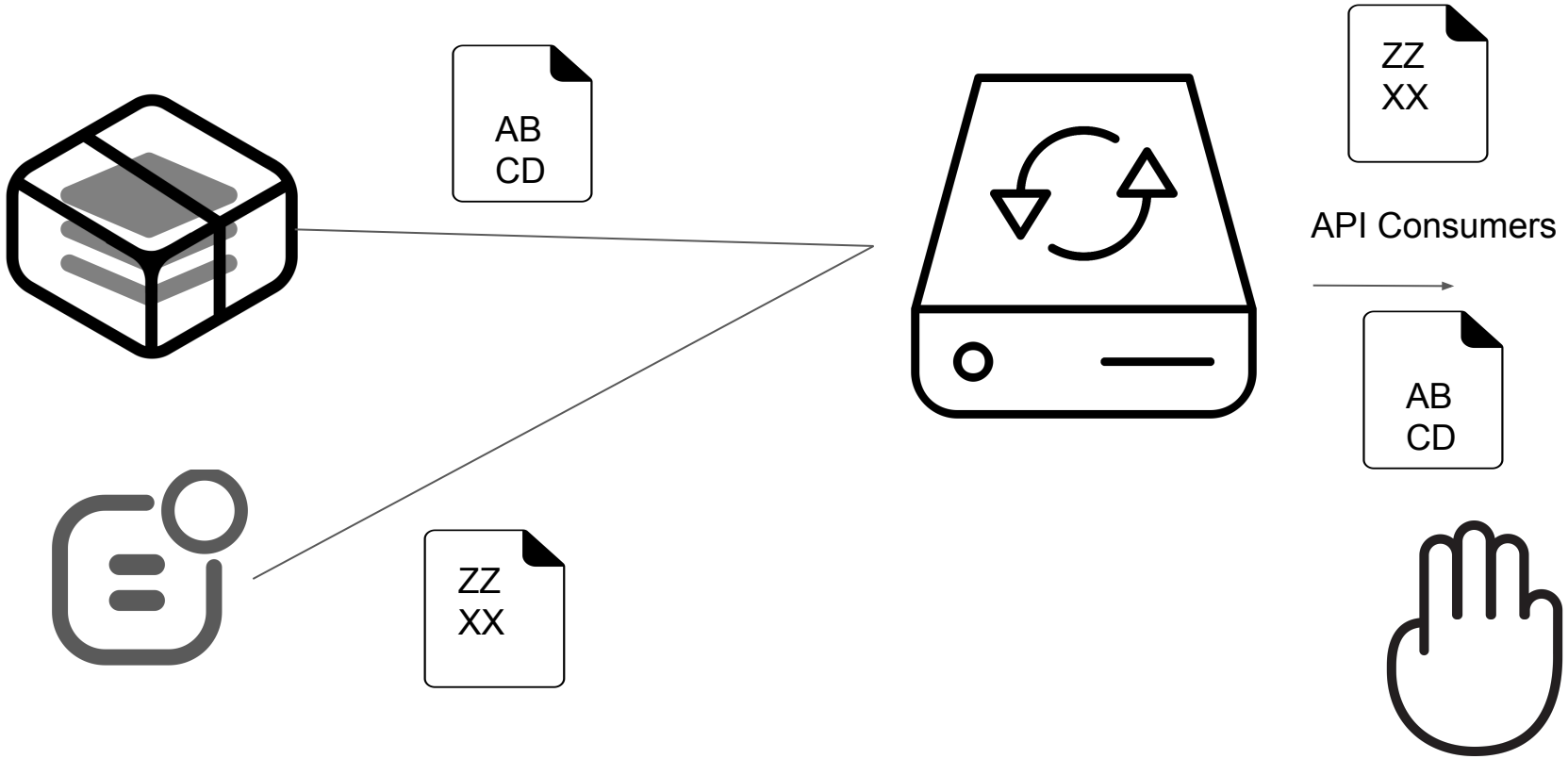
Introduce new functionality via new components



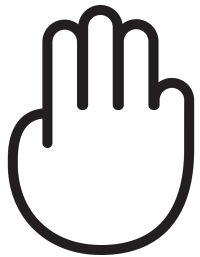
Introduce new functionality via new components



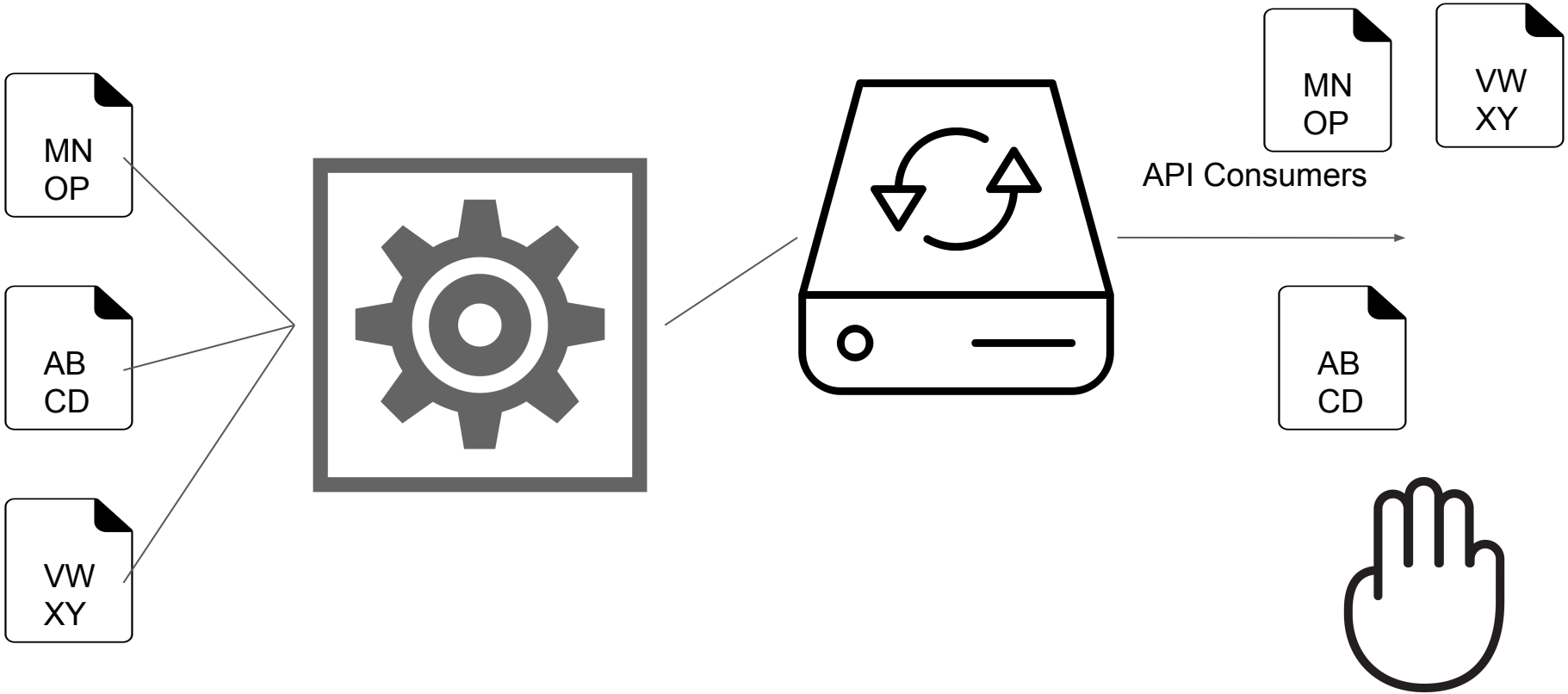
Introduce new functionality via new components



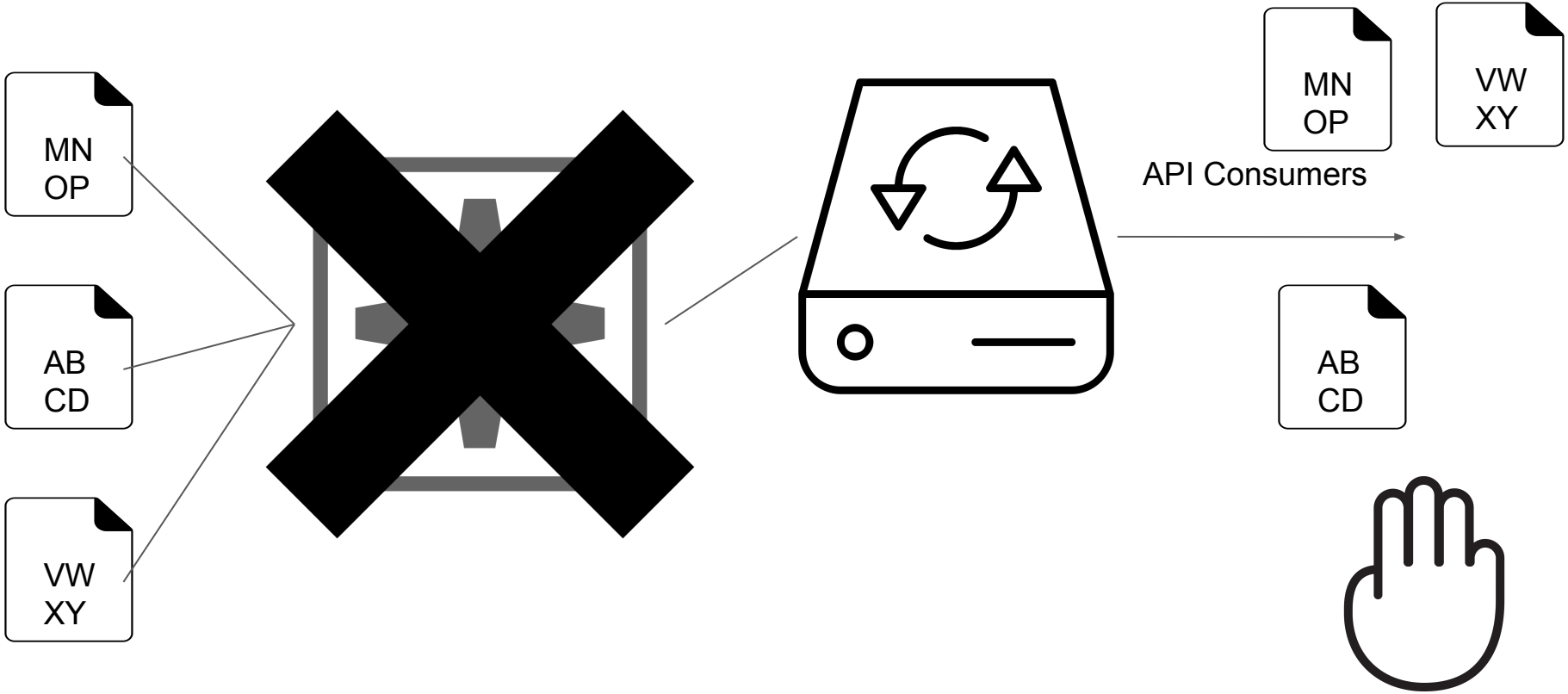
Acquire new functionality via 3rd party facades



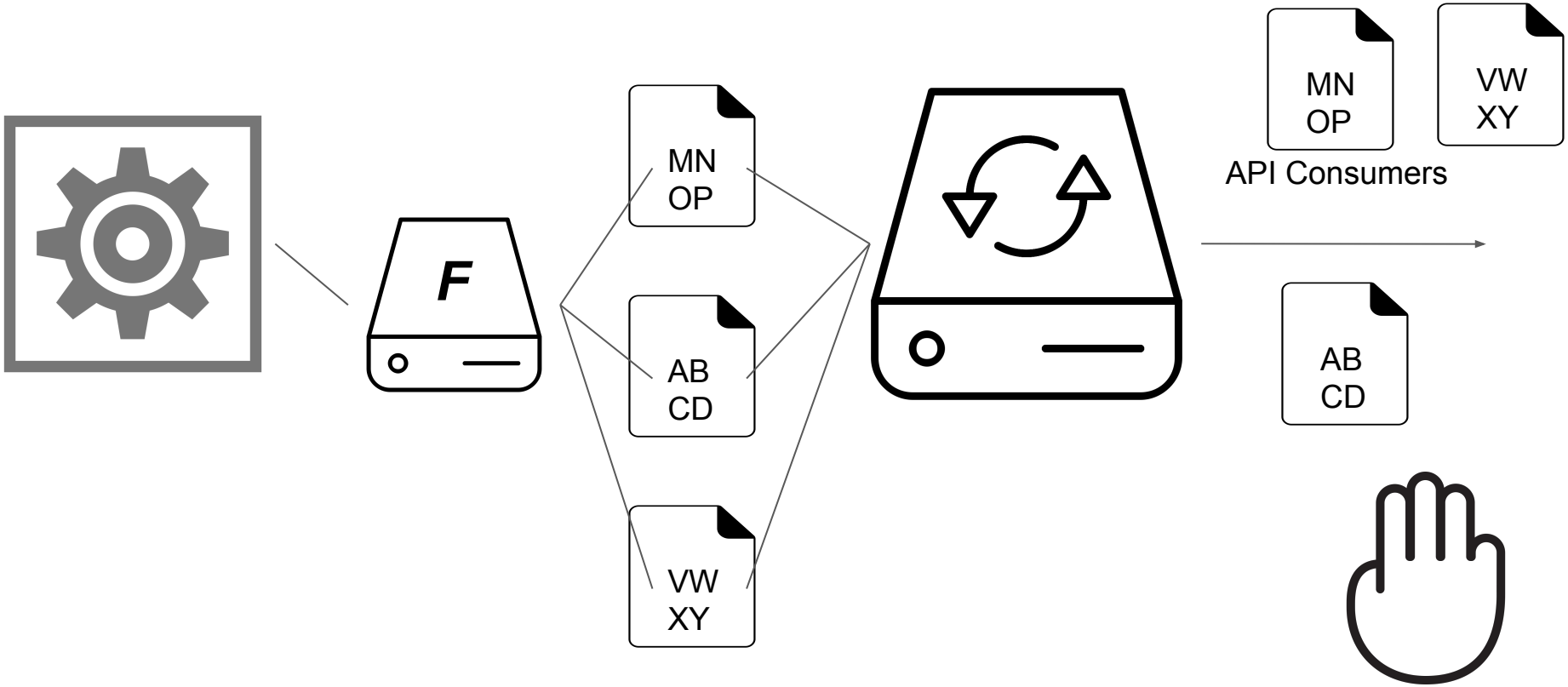
Acquire new functionality via 3rd party facades



Acquire new functionality via 3rd party facades

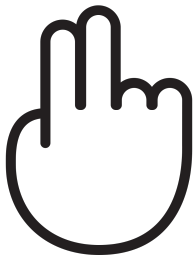


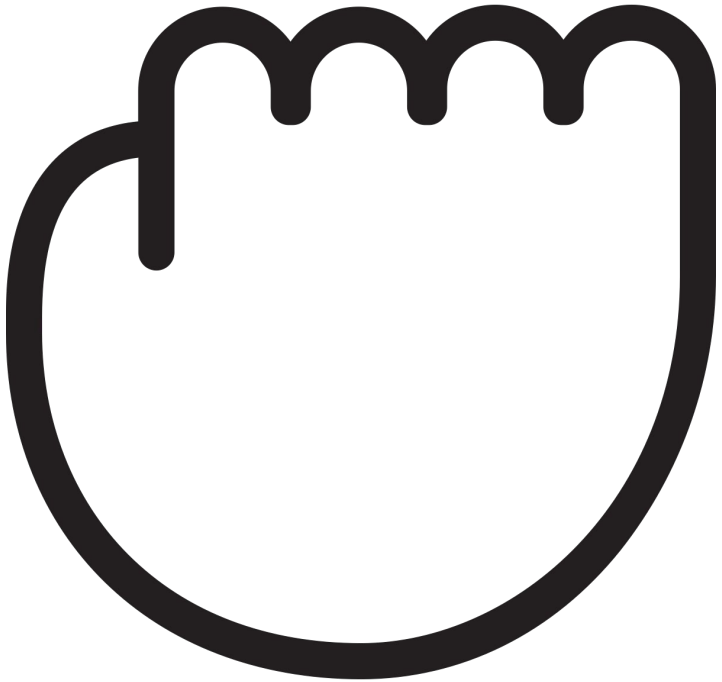
Acquire new functionality via 3rd party facades

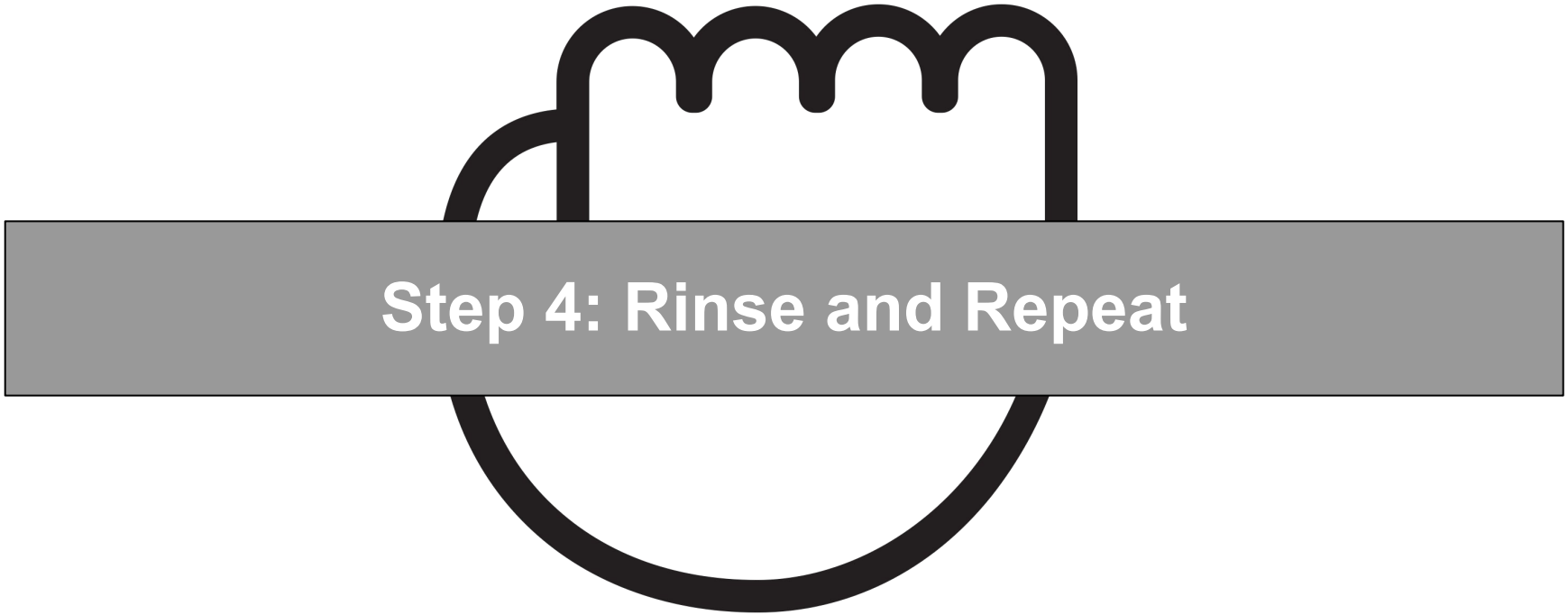


Add Functionality

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

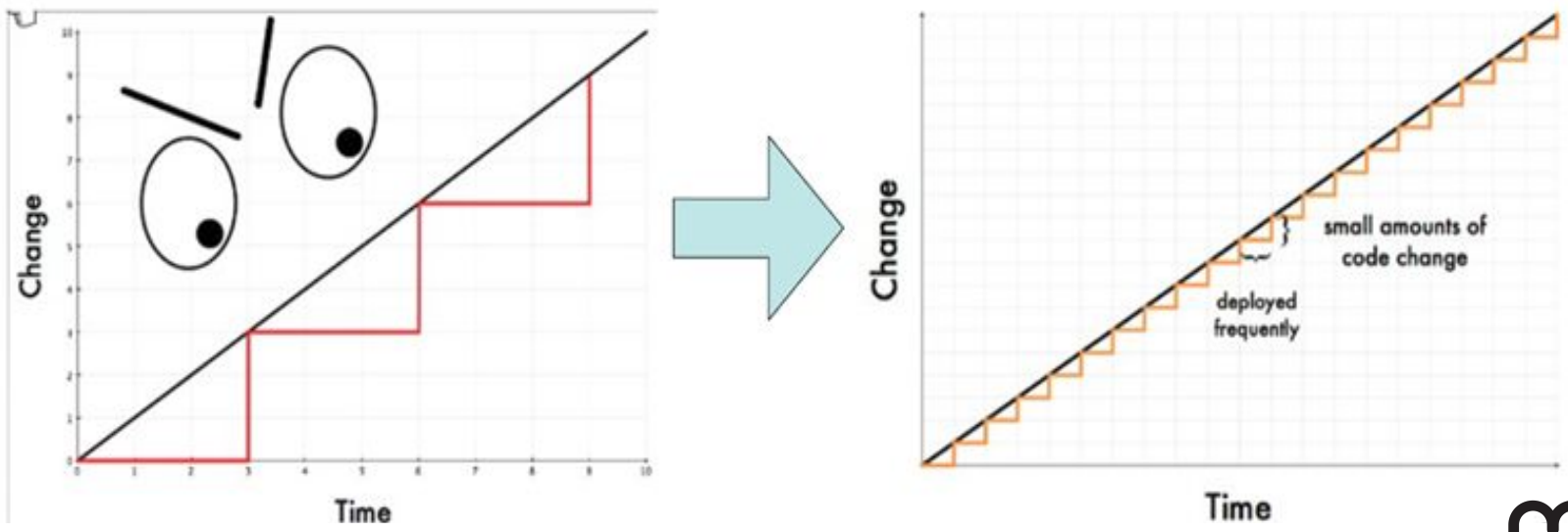





Step 4: Rinse and Repeat

All changes are incremental

All changes are incremental



https://www.slideshare.net/jallspaw/ops-metametrics-the-currency-you-pay-for-change-4608108

All changes are incremental

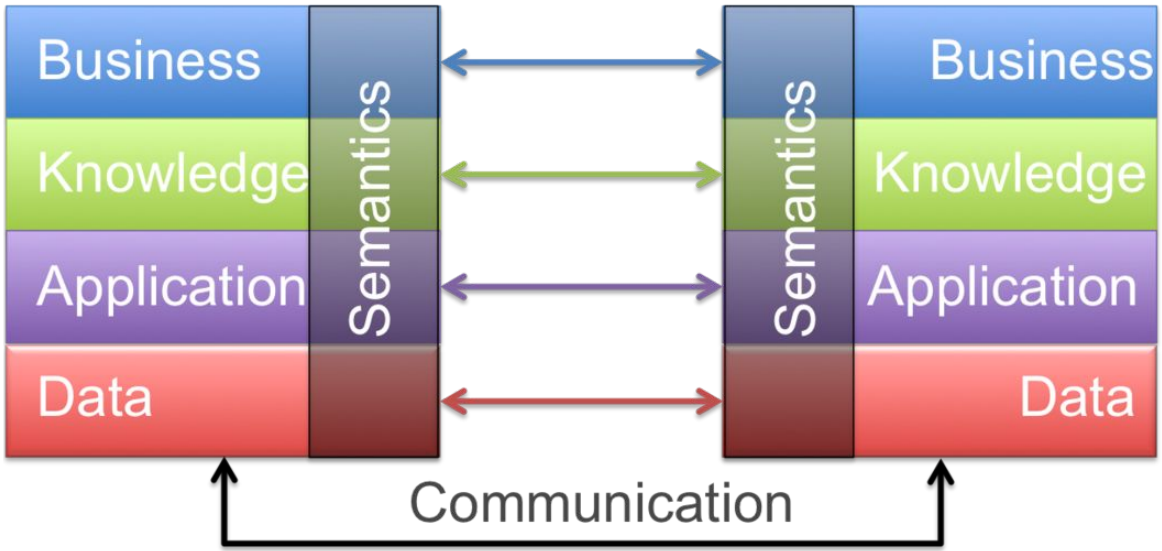
*"Incremental change may just be **the next big thing** this decade."*

-- Sandeep Kishore, HCL Technologies

https://www.wired.com/insights/2013/11/the-power-of-incremental-innovation/

Aim for loose interop, not tight integration

Aim for loose interop, not tight integration



Aim for loose interop, not tight integration

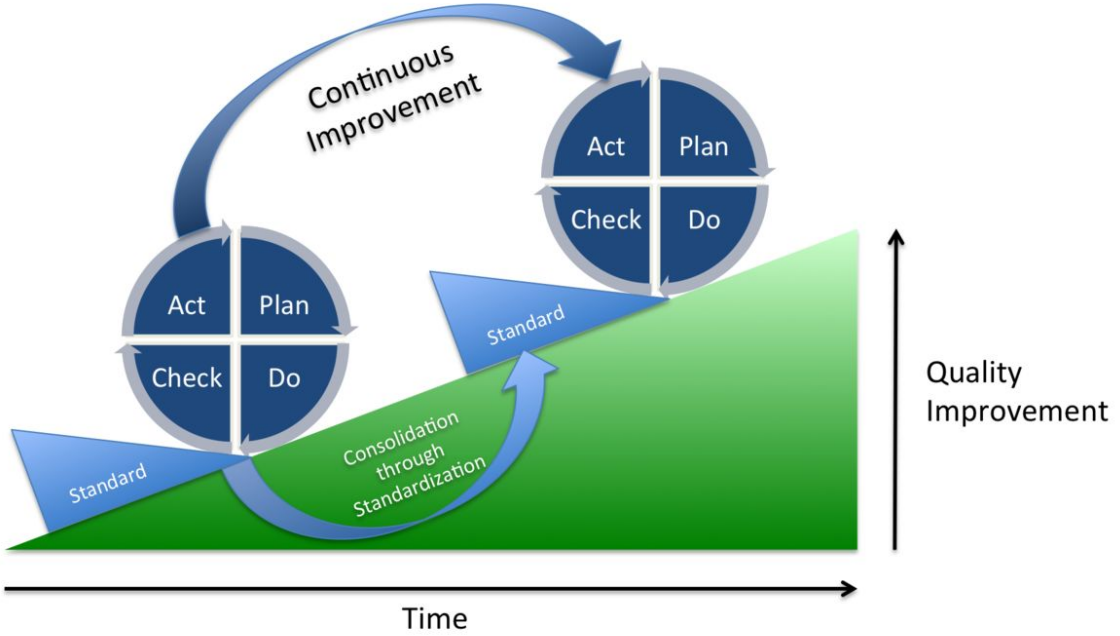
"Interoperation is peer to peer. Integration is where a system is subsumed within another."

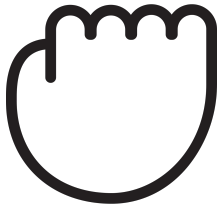
-- Michael Platt, Microsoft

https://blogs.technet.microsoft.com/michael_platt/2005/08/30/integration-and-interoperability/

Support continuous improvement

Support continuous improvement

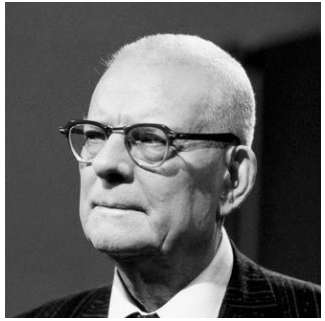




Support continuous improvement

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

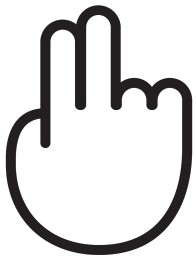
-- W. Edwards Deming



https://deming.org/management-system/pdsacycle

Rinse and Repeat

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

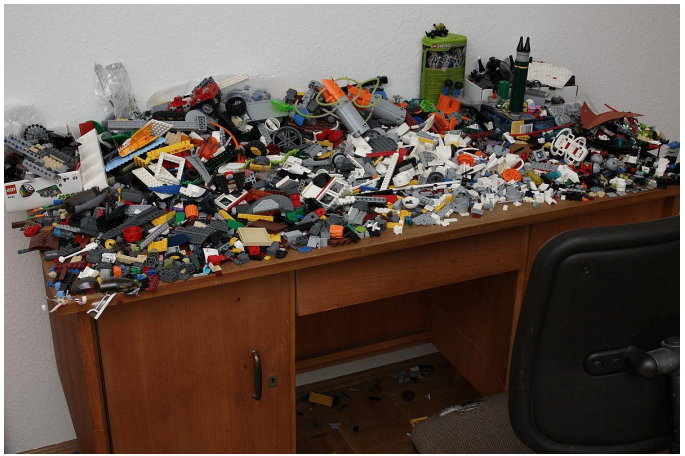


So...

The Quick Summary

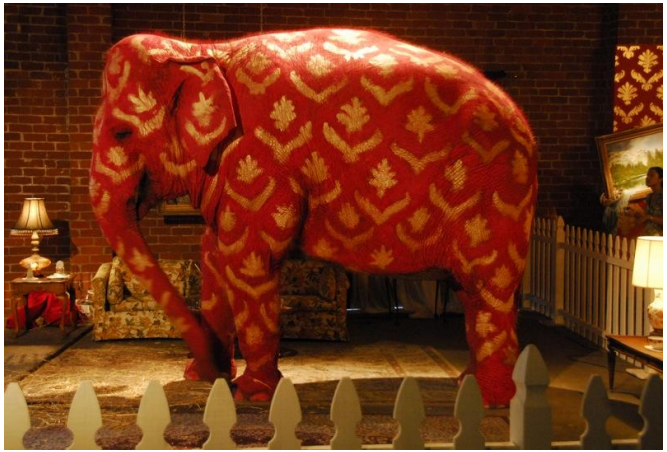
The Quick Summary

- Focus on Unlocking Value



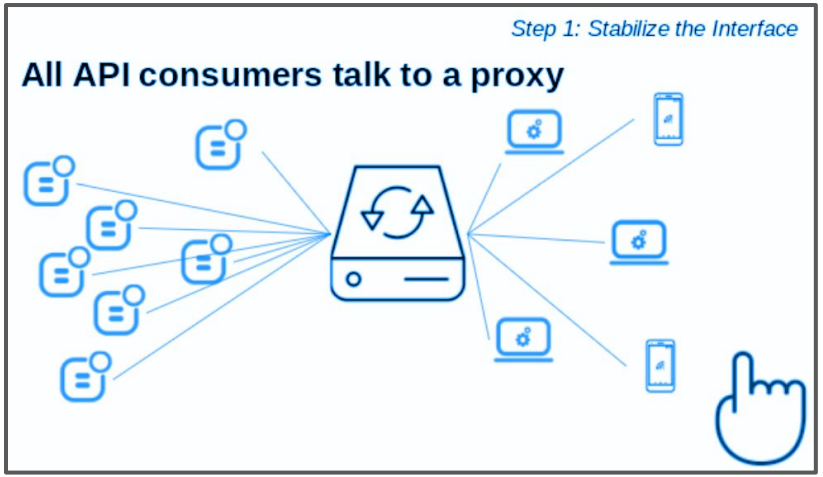
The Quick Summary

- Focus on Unlocking Value
- Change One Thing



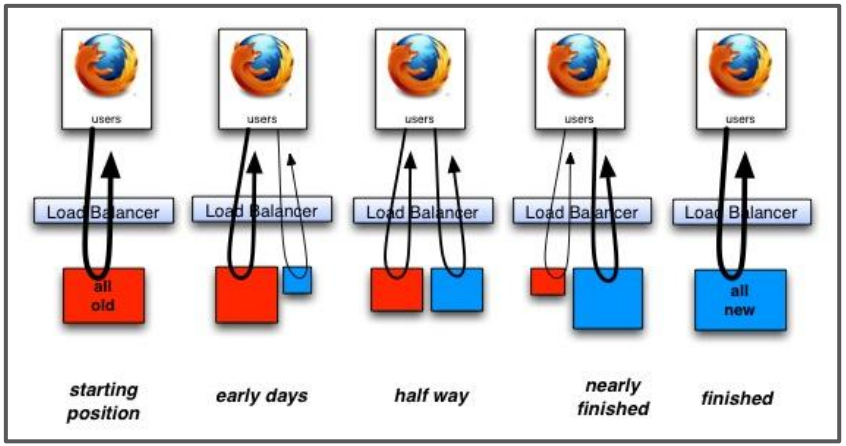
The Quick Summary

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



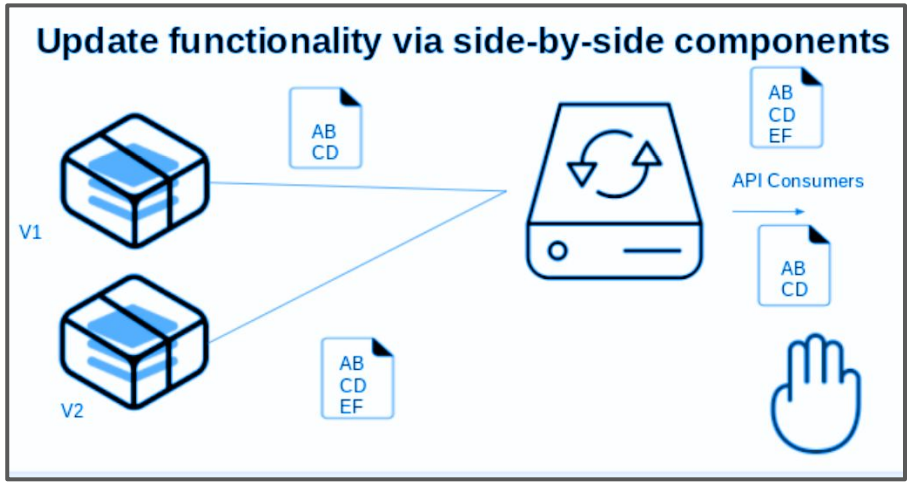
The Quick Summary

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



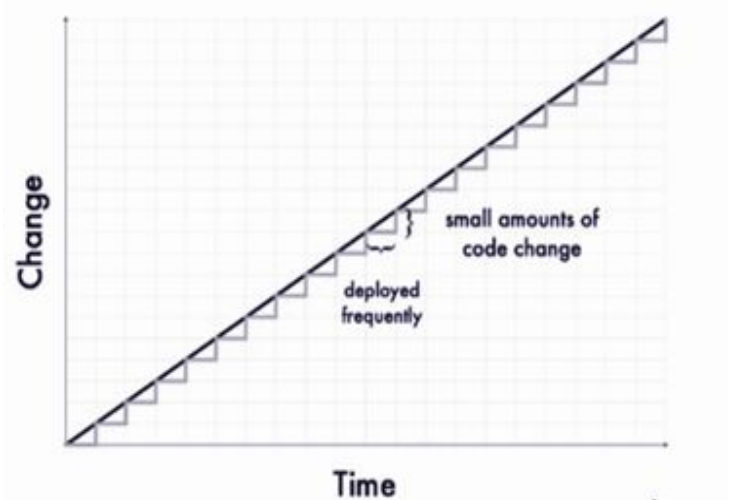
The Quick Summary

- 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




STRATEGY
PLAN
NOW

Service and API Migration at Speed

Mike Amundsen
@mamund
training.amundsen.com