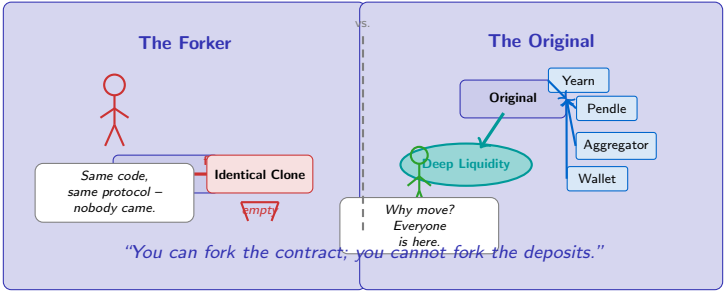
Composability Business Models

Every primitive is forkable — but the liquidity sitting on top is not

Digital Finance


The Forker
vs.
The Original
Same code,
same protocol –
nobody came.
Identical Clone
empty
Original
Yearn
Pendle
Aggregator
Wallet
Deep Liquidity
Why move?
Everyone
is here.
"You can fork the contract; you cannot fork the deposits."

Why Does the Forkable Primitive Earn Less Than the Aggregator That Sits on Top of It?

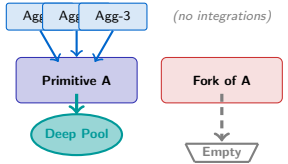
(In business-model language, a *moat* = a competitive advantage that rivals cannot easily copy.)

The Forkable-Moat Friction

A composable stack is a ladder of primitives, each one consuming the output of the layer below. Anyone can copy the source code of a lower primitive and ship a perfect functional clone. Yet the clones sit empty while the original keeps capturing fees.

The friction the BM exploits: switching costs in a composable ecosystem do not live in the contract. They live in the deposits that already point at the contract — and in the dozens of upstream integrations that hard-code its address.

- Contract code is open, auditable and replicable in an afternoon.
- Liquidity, integrations and oracle feeds take years to accrue and cannot be copy-pasted.
- A new entrant must persuade depositors AND every aggregator above them to redirect plumbing simultaneously.
- The primitive that draws first liquidity often draws all of it.



Value Proposition: a depositor home that everything points at

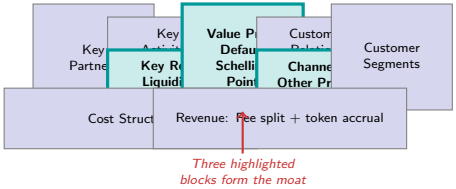
BMC anchor: Value Proposition. The proposition is not the code – it is the position as a default Schelling point for everyone above and below.

What Does the Business Model Canvas Reveal When Applied to a Forkable Primitive?

The Composability BMC Pattern
Osterwalder's Business Model Canvas, applied to a composable primitive, exposes a striking inversion: the most valuable blocks are not the ones the protocol controls. They are the ones it attracts.

- **Value Proposition:** Be the default integration target – the address every upstream integrator hard-codes.
- **Key Resources:** Deposited liquidity, oracle feeds, governance token distribution. None of these are written into the source code.
- **Channels:** Other protocols. The primitive does not market to end users; aggregators do that.
- **Revenue Streams:** Fee splits with integrators, plus governance-token accrual from protocol-owned activity.

The pattern: each forkable primitive lives or dies on the three non-code blocks. Code is necessary; liquidity and integration are the moat.



BMC reveals an inversion: the three blocks the protocol does NOT control by code (liquidity, default-position, integration-channel) form the only durable moat.

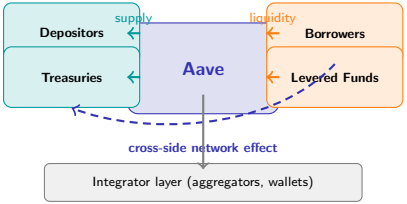
How Does Aave Turn a Forkable Lending Pool into a Multi-Sided Default?

A *two-sided platform* = a service that sells access to one user group by first attracting another; *cross-side network effects* mean the value on each side grows with participation on the other.

The Aave Case

Aave, the open-source lending protocol launched in Switzerland and deployed to Ethereum mainnet with smart contracts maintained by a Swiss non-profit and a United-Kingdom-registered company, is, structurally, a small set of pooled-lending contracts. Anyone has forked them. Many such forks exist on every major chain. Yet none has displaced Aave on the chains where it landed first.

- **Multi-sided platform:** depositors on one side, borrowers on the other, integrators (yield aggregators, wallets, structured products) underneath as a third side.
- **Cross-side network effect:** more depositors lower the borrow rate, attracting more borrowers; more borrowers raise the deposit yield, attracting more depositors; deeper liquidity attracts more integrators, who route still more flow back into the pool.
- **Chicken-and-egg solution:** Aave bootstrapped the depositor side first with governance-token incentives, knowing borrowers and integrators would follow the deepest pool.
- **Result:** the integrator layer hard-codes Aave's address. Every fork must persuade those integrators to redirect, one by one.



Platform economics anchor: Aave is a three-sided platform (depositors, borrowers, integrators). The integrator layer is the layer that locks every fork out.

How Does Yearn Start as a Yield Picker and End as the Whole Aggregator Stack?

Unbundling = pulling one service out of a historical bundle and offering it alone; *rebundling* = stacking adjacent services onto that foothold once trust is established. Clayton Christensen (Harvard Business School) argued disruptors start narrow, earn trust, then expand upward.

Christensen's Cycle Applied to Yearn

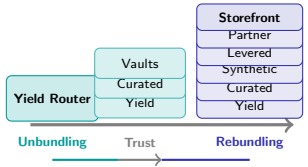
Yearn Finance, the Ethereum-mainnet yield-aggregator protocol originated by a pseudonymous developer in the United States and governed by a distributed token-holder community, illustrates the cycle on an open stack.

Phase One — Unbundling: Yearn launched as a single function – pick the highest-yielding lending pool today, deposit there, repeat. Every other piece (custody, accounting, the pools themselves) was outsourced to lower primitives.

Phase Two — Trust Earned: Depositors learned that the optimiser worked. The protocol attracted recurring deposits without needing to acquire each one through marketing.

Phase Three — Rebundling: Curated strategy vaults, synthetic-asset vaults, leverage vaults, partner-chain vaults – each one a higher-margin product layered on top of the original yield-router. The single-purpose primitive became a strategy storefront.

The irony of composability: rebundling takes more code but produces more sticky deposits, because each new strategy creates one more non-portable reason to stay.



Christensen anchor: in a composable stack, rebundling is sticky because every added strategy is a non-portable switching cost on top of the original wedge.

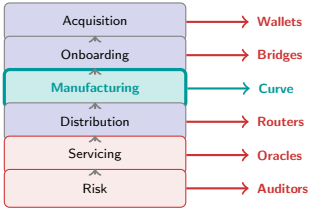
Where Does Curve Insert Itself in the Composability Value Chain – and Why That Link?

Value chain = the ordered sequence of activities a service passes through. Evans and Wurster (BCG) argued that when information is cheap, each link can be split off to a specialist — the chain *deconstructs* into independent layers.

The Composability Value Chain

A composable stack has six links, each contestable, each with a different fee profile.

- **Acquisition** – wallet onboarding, social referral
- **Onboarding** – KYC-light or KYC-free entry rails
- **Manufacturing** – the core primitive (lending, swap, options) that produces a financial output
- **Distribution** – the routing layer that picks where to send capital this block
- **Servicing** – price oracles, keepers, liquidation bots
- **Risk** – audit, insurance, governance hardening



Curve Finance, the Ethereum-mainnet stable-asset automated-market-maker protocol originated by a developer based in Switzerland and deployed on open infrastructure, owns the Manufacturing link for stable-asset swaps. Its specialised invariant produces lower slippage on like-priced pairs than any generic AMM. Because every yield aggregator routes through that invariant, the link captures fees that an end-user-facing front-end would never see. Owning the link rather than the screen is the value-chain bet of a composability primitive.

Evans-Wurster anchor: in a composable stack, the Manufacturing link captures the deepest fee because every higher link must route through it.

Is Pendle's Yield-Stripping Niche a Durable Moat or a Temporary Arbitrage on Curve?

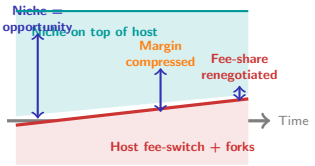
Regulatory arbitrage = a firm earns profit specifically because it faces a lighter rulebook than its competitors, not because it is better at the underlying business. The advantage lasts only as long as the rulebook gap does.

The Composability Arbitrage Tension

Pendle Finance, the yield-stripping protocol launched on Ethereum mainnet by a team based in Singapore, splits a yield-bearing token into a principal piece and a yield piece, then trades each separately. The protocol does not own the underlying yield engine – it sits on top of Aave, Curve, EigenLayer and others, taking a slice each time a user separates or recombines.

- **Regulatory arbitrage:** the trade resembles an interest-rate swap but is shipped as a token, escaping much of the conventional derivatives perimeter for now.
- **Composability arbitrage:** the protocol relies on Curve's invariant for pricing the principal token. If Curve flips on a fee-switch or the host primitive changes its yield mechanics, Pendle's economics shift overnight.
- **Integration revenue-sharing:** the moat forms only if Pendle captures durable fee-share agreements with its hosts before generic forks do the same trick.

The tension: composability arbitrage is even more transient than regulatory arbitrage. The host can flip a flag, fork the consumer, or ship a competing primitive in a single governance vote.



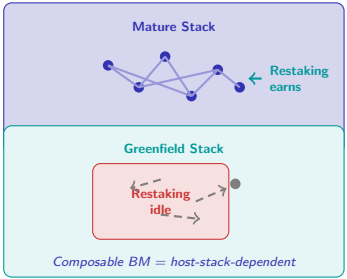
Arbitrage anchor: a composability niche is a moat only if integration revenue-share lockups are signed before the host ships its own version.

Why Does EigenLayer's Restaking Primitive Earn More on a Mature Stack Than a Greenfield One?

The EigenLayer Lesson

EigenLayer, the Ethereum-mainnet restaking protocol developed by a research group based in the United States and shipped as open smart-contract infrastructure, recycles the security of an existing stake – letting the same deposit secure a primary chain and additional services (oracles, sidechains, data availability). The model only earns when the underlying stake is large, mature and trusted. On a young chain with thin stake, the same protocol is structurally a non-starter.

- Restaking value depends on the depth and reputation of the base-layer stake – a non-portable resource.
- Mature stacks already host the integrators, oracle providers and slashing standards that the restaking layer needs.
- Greenfield stacks lack the integrator layer; the restaking primitive has nothing to plug into.
- The lesson: composable BMs are infrastructure-sensitive in a way that no balance-sheet business is. The same code earns nothing in the wrong context.



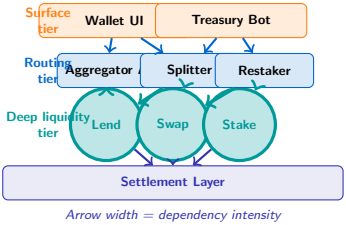
Context-dependency anchor: composable BMs are stack-sensitive in a way balance-sheet BMs are not. Code is portable; the host stack is not.

Which Five Tests Predict Whether a Composable Primitive Becomes a Lasting Moat?

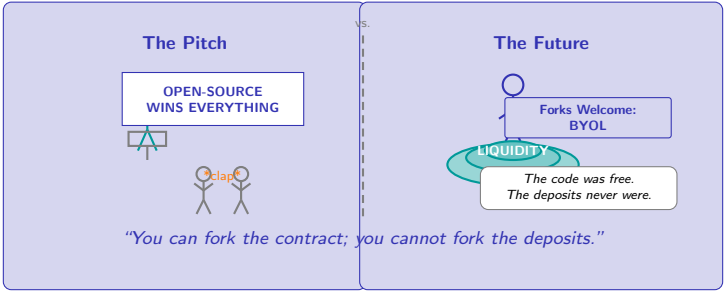
The Five-Test Synthesis for Composability

1. **Friction test:** Does the primitive solve a real composability friction (low slippage, atomic settlement, capital efficiency)? Real friction means upstream protocols wire to it voluntarily.
2. **Platform test:** Is there a third side – integrators – whose presence raises the cost of any fork? A pure two-sided pool can be drained by a token incentive; a three-sided platform cannot.
3. **Rebundling test:** Can the protocol layer adjacent products on top of its primitive (curated vaults, structured products, branded integrations) once the wedge captures liquidity?
4. **Infrastructure test:** Is the host stack mature enough to provide the integrator layer? A primitive on a greenfield chain has no aggregator above it.
5. **Arbitrage test:** Will the host primitive ship a competing version (fee-switch flip, in-house fork, governance capture)? A consumer-of-host BM has the shortest arbitrage half-life.

A primitive that passes at least three of these tests builds a liquidity-and-integration moat that no fork can replicate. Most forks pass the friction test only.



Five-test synthesis: friction + platform + rebundling + infrastructure + arbitrage. Passing three or more = a primitive a fork cannot dislodge.



"You can fork the contract; you cannot fork the deposits."