Bridge + Vesting + Module/IPFS Requirements (Comprehensive)

0) Objectives

- Enable COM → MOD migration with conviction filtering:
- Longer vesting = higher multiplier.
- Discourage short-term holders.
- Bootstrap liquidity for MOD while avoiding immediate sell pressure.
- Maintain predictable inflation and emissions schedule.
- Provide a one-time, trust-minimized migration path.
- Reuse/port existing **module registry + IPFS integration**.
- Start from **Substrate solo-chain template** with minimal but complete runtime.

---

1) Scope

- **In Scope**
- One-time bridge from COM (legacy) → MOD (new chain).
- Snapshot + Merkle root claims (no live generic cross-chain messaging).
- Vesting schedules with convex duration multiplier.
- Treasury receives unclaimed allocation with max duration vesting.
- Module registry pallet with IPFS integration for off-chain storage.
- Governance control over parameters, pause switch, and upgrades.
- **Out of Scope**
- Generic live token bridge (beyond snapshot claim).
- Perpetual re-locking, staking, or multi-chain interoperability (future work).
- Legacy COM → Base back-bridge.

---

2) High-Level Flow

1. Snapshot COM balances at block `S_height`.
2. Build Merkle tree `(address, balance, snapshot_block, chainId, salt)`.
3. Publish Merkle root + snapshot metadata on MOD (`pallet-bridge`).
4. User interacts with a minimal claims contract on Base (EVM) to choose `T_days` and emit an attestation/event.

5. Off-chain relayer/FE submits the Merkle proof, attestation data, and `T_days` to MOD `pallet-bridge::claim`.
6. MOD verifies proof and parameters, computes entitlement, and mints a vesting schedule on MOD.
7. Treasury receives unclaimed allocation vested at `T_max`.

2.1 Interaction Model (Base EVM claims contract)

- Base contract serves as a UX anchor and source of user-attested `T_days`; no generic cross-chain messaging.
- MOD remains the source of truth; only claims recorded on MOD are effective.
- Off-chain relayer watches Base events and calls MOD `claim` with the user's Merkle proof and `T_days`.
- Security: include `chainId`, `snapshot_block`, and a salt in the leaf to prevent replay on other forks; verify Base chain ID and contract address in relayer config.

3) Tokenomics Math

- **Base ratio:** `R = MOD per COM`. Example: `R = 0.1`.
- **Duration multiplier:**

```
f(T) = (T / T_max)^k, k > 1
```

- `f(0) = 0`
- `f(T_max) = 1`
- **Effective entitlement:**

`E = E_base * f(T)` * **Vesting unlock:**

- Linear: `unlock(t) = E * t / T`
- Optional back-loaded: `unlock(t) = E * (t/T)^q, q > 1`

3.1 Eligibility and bounds (snapshot assumptions)

- **Minimum balance to be eligible**: at or above the chain's existential deposit (ED). Balances below ED are filtered out in the snapshot pipeline and do not produce a claim.
- **Maximum balance**: no explicit per-account cap; the effective maximum is simply the largest balance present at the snapshot.
- **Proportionality**: `E_base` scales linearly with the snapshot balance; multipliers apply uniformly via `f(T)`.

4) Pallet Requirements

4.1 Bridge Pallet (`pallet-bridge`)

- **Storage:**
- `MerkleRoot`
- `SnapshotBlock`, `SnapshotTime`
- `BaseRatio`
- `Claimed: map<AccountId, bool>`
- `Params: { TMin, TMax, K, UnlockShape }`
- `Paused: bool`
- **Calls:**
- `claim(proof, leaf, T_days)`
- `set_params(...)` (governance)
- `pause()` / `unpause()` (emergency)
- **Events:**
- `Claimed(account, base, T_days, effective)`
- `ParamsUpdated`
- `Paused`, `Unpaused`

4.2 Vesting Pallet (`pallet-vesting` or fork)

- Supports per-account schedules.
- Configurable unlock shape (linear or back-loaded).
- Prevents transfer of locked funds.
- Events for unlocks.

4.3 Treasury Pallet (`pallet-treasury`)

- Custody of unclaimed allocation.
- Assigned default max-duration vesting schedule.

4.4 Module Registry Pallet (`pallet-module-registry`)

- **Storage:** mapping of module IDs → IPFS CIDs, metadata, tags.
- **Calls:** register, update, retire, transfer ownership.
- **Events:** `Registered`, `Updated`, `Retired`, `OwnershipTransferred`.
- **Validation:** enforce CIDv1, deterministic size limits.

4.5 Optional IPFS Pinning Pallet (`pallet-ipfs-pin`)

- Queues pin/unpin requests.
- Off-chain worker or infra picks up events to manage IPFS cluster.

4.6 Governance

- **pallet-democracy / pallet-referenda**
- **pallet-collective**
- **pallet-sudo** (bootstrap only, removed later)

4.7 Core FRAME Pallets

- `frame-system`
- `pallet-timestamp`
- `pallet-balances`
- `pallet-assets` (if multi-asset needed)
- `pallet-utility` (batch, multisig)
- `pallet-identity` (optional, module authorship)

---

5) Snapshot Pipeline (Off-chain)

- Extract balances at `S_height`.
- Filter dust below existential deposit.
- Exclude known burn/system accounts.
- Build canonical JSON and Merkle tree.

- Public artifacts:

- JSON snapshot (IPFS pinned)

- Merkle root
- Totals report

---

6) Claim UX

- One transaction claim:

- User inputs `T`.

- Display:
    - Base entitlement `E_base`
    - Multiplier `f(T)`
    - Effective entitlement `E`

  - Daily unlock
  - Warn claims are one-time and final.

7) Anti-abuse & Safety

- **Pause switch** in bridge and vesting pallets.
- **Replay protection** (chainId + snapshot block).
- **Reentrancy safe** (state set before mint/vest).
- **Dust handling** via ED filter.
- **Dispute window** before claims open, to contest snapshot.

8) Edge Cases

- **Already claimed**: reject via `Claimed` map; idempotent checks in `pallet-bridge::claim`.
- **Wrong chain / replay**: Merkle leaf includes `chainId` and `snapshot_block`; proofs from other chains/forks fail.
- **Lost keys**: default policy is no manual remediation; if governance chooses, a narrow exception process can be proposed and logged on-chain.
- **Dust balances**: below-ED balances excluded at snapshot; no claim produced; amounts can be aggregated to treasury per published rule.
- **Double-submission from Base**: Base contract emits events; only MOD-side claim changes state. Relayer must be idempotent; MOD enforces single claim per account.
- **Inconsistent totals**: deployment halts if the sum of leaves does not match the published snapshot report; governance must re-publish root.
- **Throughput limits**: optional rate-limiting (e.g., max N claims per block) to protect RPC during open.

9) Governance Parameters

- Adjustable by governance:

- `R`, `T_min`, `T_max`, `k`, `unlock_shape`

- `snapshot_block`, `merkle_root` (only pre-open or via time-lock)
- `treasury_account`

10) Observability

- Indexer tracks:

- `Claimed` events
- Treasury vesting trajectory
- Chosen `T` distribution
- Public dashboard:
- Claimed vs unclaimed
- Weighted average `T`
- Emissions per day

---

11) Test Plan

- **Unit tests:**
- Merkle verification correctness.
- Multiplier math matrix.
- Vesting math linear/back-loaded.
- Pause behavior.
- **Integration:**
- Genesis config loads snapshot root.
- Claim + vesting end-to-end.
- Treasury schedule initialized.
- **Property/Fuzz:**
- Proof verification.
- Claim replay.
- Unlock trajectory.

---

12) Deliverables & Milestones

1. **M0 — Bootstrap**

- Solo-chain template + FRAME pallets
- Add balances/treasury/governance 2. **M1 — Bridge Pallet**
- Merkle proof claim
- Params + pause 3. **M2 — Vesting**
- Linear unlock schedules
- Treasury schedule 4. **M3 — Module Registry**
- Port pallet + IPFS integration

- Off-chain infra for pinning 5. **M4 — Hardening**
- Fuzz, audit, doc 6. **M5 — Testnet**
- Public faucet
- Canary Merkle root
- Governance dry-run

---

12) Deployment Timeline Goals (indicative)

These goals are for planning only and subject to governance approval:

- Bridge claims and vesting MVP on testnet as early as feasible.
- Main enablement targeted shortly after successful testnet and review.

---

13) Owners & Responsibilities (initial)

- **Ziggy**: SDK/relayer integration; submit MOD claims from Base events; support testnet.
- **Bako**: Author and finalize bridge requirements/parameters; coordinate snapshot pipeline.
- **Huck**: Front-end flows for Base claim UX and MOD vesting visualization.
- **Fam**: Module system V1 (remote execution) and cross-component coordination.

Notes: ownership is indicative; final assignment subject to approval.