# **M0 — Environment & Scaffolding (≤100 steps, critical-path)**

## **Pre-flight & repository hygiene**

1. Confirm you have admin on the GitHub repo and Actions enabled for the default branch.
2. Create a SECURITY.md with a security contact and responsible disclosure guidance.
3. Add a permissive LICENSE (e.g., MIT) and ensure compatibility with Uniswap/OZ licenses.
4. Create a top-level CODEOWNERS file (contracts, TS core, CLI each owned by two reviewers).
5. Enable branch protection on main: require PR, require status checks, require 2 reviews.
6. Enable “Require signed commits” (or at least verified) on protected branches.
7. Turn on GitHub secret scanning and Dependabot security updates for the repo.

## **Toolchain pinning & package manager**

1. Add .nvmrc (Node LTS you target; e.g., v20.x).
2. Add .tool-versions (asdf) or engines in root package.json to pin Node.
3. Adopt pnpm workspaces; add pnpm-workspace.yaml with packages/\* globs.
4. In root package.json, set "packageManager": "pnpm@<exact>" for reproducibility.
5. Add a bootstrap script: "setup": "pnpm i -w".

## **Monorepo structure & ESM setup**

1. Create packages: packages/core-exec, packages/adapters-evm, packages/cli.
2. In each package.json, set "type": "module" (ESM), "private": true.
3. Add root tsconfig.base.json (strict, module: "ESNext", moduleResolution: "bundler", skipLibCheck:false).
4. Add per-package tsconfig.json that extends the base, sets rootDir/src, outDir/dist.
5. Configure paths in tsconfig.base.json for local imports (@core-exec/\*, etc.).
6. Add vitest (or jest if preferred) to root devDependencies; add test scripts per package.

## **Formatting, linting, and hooks**

1. Add .editorconfig for consistent whitespace, LF endings.
2. Add prettier with a root config; add format scripts (pnpm -w format).
3. Add eslint with TS plugin, @typescript-eslint/parser, ESM rules.
4. Add solhint config for Solidity style checks.
5. Add lefthook or husky for Git hooks; install dev dependency.
6. Configure pre-commit hook: run prettier -c, eslint, and solhint on staged files.
7. Configure pre-push hook: run pnpm -w typecheck and the fastest unit tests.

## **Solidity toolchain (Foundry) & remappings**

1. Install Foundry (forge/cast/anvil) locally; record version in foundryup --version.
2. Add foundry.toml (pin solc version; enable optimizer with target runs, e.g., 200–800).
3. Add remappings for OpenZeppelin and Uniswap v3 interfaces/periphery.
4. Create contracts/lib/ to host vendored interfaces if needed (document source commit).
5. Add forge-std as dev dependency; update remappings accordingly.

## **Common contracts scaffolding**

1. Create contracts/src/common/Errors.sol with custom errors used everywhere.
2. Create contracts/src/common/Events.sol for shared event definitions.
3. Create contracts/src/common/Types.sol (on-chain mirrors of StepParams, etc.).
4. Ensure SPDX identifiers and pragma match repo policy (e.g., pragma solidity ^0.8.24;).

## **Environment configuration & secrets safety**

1. Create .env.example with placeholders: RPC\_URL, FORK\_URL, PRIVATE\_KEY\_DEPLOYER, CHAIN\_ID, WETH\_ADDRESS, UNIV3\_ROUTER, UNIV3\_QUOTER, GUARDIAN\_ADDRESS, OWNER\_ADDRESS.
2. Add .env to .gitignore.
3. Add scripts/dev/validate-env.ts to assert .env completeness (fail with actionable messages).
4. Root script "env:check": "tsx scripts/dev/validate-env.ts".

## **Network addresses & config artifacts**

1. Create config/addresses.json (per-network entries, keys: chainId, weth, univ3Router, univ3Quoter, executor, vault, etc.).
2. Create scripts/dev/fill-addresses.ts to merge env-provided addresses into config/addresses.json.
3. Root script "addr:sync": "tsx scripts/dev/fill-addresses.ts".

## **Local chain orchestration**

1. Create scripts/dev/anvil.sh to start anvil with deterministic --chain-id, --block-time, and fixed mnemonic.
2. Print funded dev accounts on boot; write them to .cache/anvil-accounts.json.
3. Root script "anvil:up": "bash scripts/dev/anvil.sh", "anvil:kill": "pkill -f anvil || true".

## **Minimal mocks & fixtures (for truly repeatable M0)**

1. Add contracts/src/mocks/MockERC20.sol (mintable, 18 decimals).
2. Add contracts/src/mocks/MockRouter.sol (no-op approvals, deterministic swap math) as a fallback if Uniswap v3 is not deployed locally.
3. Add scripts/dev/mint.ts to mint MockERC20 to the deployer and test users on anvil.
4. Root script "dev:mint": "tsx scripts/dev/mint.ts".

## **Deployment scaffolding (no logic yet)**

1. Create contracts/script/Deploy.s.sol that deploys: PauseGuardian, RouteRegistry, PolicyGuards, SettlementVault, BundleExecutor (skeletons).
2. Create contracts/script/Configure.s.sol placeholder (will fill in M1–M4); for now just logs addresses.
3. Root script "deploy:local": "forge script contracts/script/Deploy.s.sol --rpc-url http://127.0.0.1:8545 --broadcast --legacy --slow" (adjust flags for your setup).
4. Persist outputs to deployments/local.json via --json and a small scripts/dev/save-deploy.ts.

## **Core package wiring (TypeScript)**

1. In packages/core-exec/src/BundleTypes.ts, add initial empty exports and placeholder types to satisfy imports.
2. In packages/core-exec/src/providers.ts, implement a minimal RPC client factory reading RPC\_URL.
3. Add packages/core-exec/src/abi/ directory (empty for now; CI won’t fail).

## **ESM interoperability**

1. Verify all TS packages compile with "module": "ESNext" and "target": "ES2022".
2. Ensure no require() remains; migrate commonjs imports to ESM syntax.
3. Add "exports" fields to each package.json to expose ./dist/index.js.

## **Commands & developer UX**

1. Root script "build": "pnpm -r --parallel build", each package has build: tsc -b.
2. Root script "clean": "git clean -xfd -e .env -e .env.local".
3. Root script "typecheck": "pnpm -r typecheck".
4. Root script "lint": "pnpm -r lint" and "format": "pnpm -r format".

## **CI — fast feedback loop**

1. Create .github/workflows/ci.yml with matrix (Node 18.x, 20.x).
2. CI steps: checkout → setup pnpm (exact) → setup Node (respect .nvmrc) → cache pnpm → pnpm -w i.
3. CI job 1: pnpm -w lint + pnpm -w format -c.
4. CI job 2: pnpm -w typecheck for TS workspaces.
5. CI job 3: forge build for Solidity.
6. CI job 4: spin anvil service, run forge test -vvv (for now allow zero tests).
7. Upload build artifacts (dist/, out/) on success for debugging.

## **CI — security & compliance (baseline)**

1. Add .github/workflows/deps.yml to run Dependabot or pnpm audit --prod (non-blocking).
2. Add gitleaks (or GH secret scanning native) as an informational job.
3. Add license check (e.g., license-checker) to fail on forbidden licenses.

## **Determinism & caching**

1. Configure pnpm store path cache in CI to speed installs.
2. Cache ~/.foundry and target/out across CI runs keyed by lockfiles and foundry.toml.

## **Developer documentation (M0 only)**

1. Add docs/dev/env-setup.md (Node, pnpm, Foundry install instructions with versions).
2. Add docs/dev/workflow.md with the canonical “M0 local loop”: pnpm setup → pnpm anvil:up → pnpm deploy:local.
3. Add README Quickstart section that links to both docs and the M0 commands.

## **Address & artifact verification**

1. Add scripts/dev/print-addresses.ts to pretty-print config/addresses.json and deployments/local.json.
2. Root script "addr:print": "tsx scripts/dev/print-addresses.ts".

## **Health checks & smoke test**

1. Add scripts/dev/healthcheck.ts: validates Node/pnpm/foundry versions, RPC reachable, .env complete.
2. Root script "dev:health": "tsx scripts/dev/healthcheck.ts".
3. Add scripts/dev/smoke.ts: boots anvil (if not running), deploys skeletons, mints mock, prints summaries; exit non-zero on any failure.
4. Root script "dev:smoke": "tsx scripts/dev/smoke.ts".

## **GitHub Actions permissions & provenance**

1. In workflow YAML, set permissions: contents: read, packages: read minimal; add id-token: write if you plan OIDC later.
2. Set concurrency group for CI (concurrency: group: ${{ github.ref }}-ci, cancel-in-progress: true) to avoid queue pileups.

## **Secrets management & local keys**

1. Document that .env dev keys are local only; production keys will live in GH Environments (later milestones).
2. Add .env.local support (ignored) for developer-specific overrides.

## **Logging baseline & debug ergonomics**

1. Add a tiny logger utility in packages/core-exec (just console wrappers) with levels from LOG\_LEVEL env.
2. Root script "log:levels": "node -e \"console.log('LOG\_LEVEL=debug|info|warn|error')\"" (just an educational hint).

## **Verification of remappings & compilation**

1. Add a trivial contract (e.g., contracts/src/hello/Hello.sol) to ensure Foundry compiles; test with forge test “hello world”.
2. Ensure forge test succeeds on CI with zero flakes.

## **Consistent decimals & bigints in TS**

1. Add a packages/core-exec/src/units.ts helper: parseUnits, formatUnits using viem/ethers v6; test it.
2. Set TS exactOptionalPropertyTypes and noUncheckedIndexedAccess in tsconfig.base.json.

## **Error handling policy in scripts**

1. All TS scripts must process.exitCode = 1 on failure and print actionable hints.
2. Wrap RPC calls with retry/backoff; export a small withRetry util.

## **Makefile convenience (optional but helpful)**

1. Add Makefile targets: make anvil, make build, make deploy-local, make smoke.
2. Ensure make smoke chains the exact root scripts so developers have a one-liner.

## **Reproducible local liquidity (fixture stub)**

1. Add scripts/dev/seed-liquidity.ts stub that later (M2) will deploy/poke Uniswap v3 pools; for M0 it logs “not configured” so agents don’t stall.

## **Workspace validation**

1. Add a root script "doctor": "pnpm env:check && pnpm dev:health && pnpm -w typecheck && forge build" to quickly validate the whole dev surface.
2. Run pnpm doctor locally and on CI; fix any red flags before starting M1.