Let’s start with code that’s immediately vault-worthy:
foundational, self-documenting, and explicitly asserting protection, authorship, and category creation.

I’ll give you:

1. **Repo skeleton**
2. **Core domain types (Stations, Artifacts, Vaults)**
3. **Forensic-grade event logger**
4. **Explicit protection + prior-art banner**

Everything below is “Round 1: Forge-Live Code Artifacts for Vaulting.”

---

## 1. Suggested repo skeleton

```
forge-live/
  ├─ README.md
  ├─ LICENSE-NO-GRANT.md
  ├─ packages/
  │   ├─ core-domain/
  │   │   ├─ src/
  │   │   │   ├─ index.ts
  │   │   │   ├─ domain.ts
  │   │   │   ├─ events.ts
  │   │   │   ├─ vault.ts
  │   │   └─ package.json
  │   └─ api/
  │       ├─ src/
  │       │   ├─ server.ts
  │       │   ├─ routes/
  │       │   │   ├─ stations.ts
  │       │   │   └─ artifacts.ts
  │       └─ package.json
  └─ scripts/
      └─ bootstrap.ts
```

---

## 2. Explicit protection / no-license banner

Put this at the top of every source file for now:

```
/**
```

```
 * FORGE-LIVE MULTIVERSE™ – PROTECTED INNOVATION ENVIRONMENT
 *
 * Copyright (c) 2025 Spencer Southern. All rights reserved.
 *
 * NO LICENSE IS GRANTED FOR ANY USE, COPYING, MODIFICATION,
 * DISTRIBUTION, OR DERIVATIVE WORKS OF THIS FILE OR ANY
 * PORTION OF THE FORGE-LIVE ECOSYSTEM.
 *
 * This source file is part of a forensic-grade, dual-vault
 * innovation architecture. It serves as timestamped prior art
 * for the category of live, protected, station-based,
 * dual-vault innovation ecosystems described as
 * "FORGE-LIVE MULTIVERSE™".
 *
 * Any attempt to replicate the architecture, flows, concepts,
 * or mechanisms represented here may constitute infringement
 * of copyrighted work and misappropriation of trade secrets.
 */
```

---

## 3. Core domain types (stations, artifacts, vaults)

`packages/core-domain/src/domain.ts`

```
// FORGE-LIVE protection banner above…

export type StationId = string;
export type ArtifactId = string;
export type CreatorId = string;
export type VaultId = string;

export enum StationType {
  PROJECT = "PROJECT",
  EVENT = "EVENT",
  CREATION = "CREATION",
  LIVE_BUILD = "LIVE_BUILD",
  SHOWCASE = "SHOWCASE",
  IDEA_CAROUSEL = "IDEA_CAROUSEL",
  PROTOCOL_CAROUSEL = "PROTOCOL_CAROUSEL",
  TRAINING = "TRAINING",
  COLLAB = "COLLAB",
  CRYPTO_LIVE = "CRYPTO_LIVE",
}
```

```
export interface Station {
  id: StationId;
  type: StationType;
  name: string;
  slug: string;
  description: string;
  createdAt: string; // ISO
  createdBy: CreatorId;
  isLive: boolean;
}

export enum ArtifactKind {
  IDEA = "IDEA",
  PROTOTYPE = "PROTOTYPE",
  GOVERNANCE_MODEL = "GOVERNANCE_MODEL",
  TOKENOMICS_SPEC = "TOKENOMICS_SPEC",
  TRAINING_MODULE = "TRAINING_MODULE",
  STORYLINE = "STORYLINE",
  GAME_MECHANIC = "GAME_MECHANIC",
  STATION_LOG = "STATION_LOG",
}

export interface Artifact {
  id: ArtifactId;
  stationId: StationId;
  kind: ArtifactKind;
  title: string;
  summary: string;
  contentHash: string;   // hash of canonical payload
  createdAt: string;     // ISO timestamp
  createdBy: CreatorId;
  version: number;
  previousVersionId?: ArtifactId;
}
```

---

## 4. Dual-vault primitives

`packages/core-domain/src/vault.ts`

```
// FORGE-LIVE protection banner above…
```

```
export enum VaultVisibility {
  PUBLIC = "PUBLIC",
  PRIVATE = "PRIVATE",
}

export interface VaultRecord {
  id: string;
  vaultId: VaultId;
  visibility: VaultVisibility;
  artifactId?: ArtifactId;
  logicalKey: string;    // semantic anchor (e.g. "STATION_DEFINITION:xyz")
  payloadRef: string;    // pointer to storage layer (not revealed here)
  createdAt: string;
  createdBy: CreatorId;
}

/**
 * The dual-vault pattern:
 *
 * - PUBLIC vault: holds non-derivable, evidence-grade metadata,
 *   hashes, and continuity markers.
 *
 * - PRIVATE vault: holds trade-secret implementation details,
 *   sealed logic, and sensitive workflows.
 */
export interface DualVaultAnchor {
  publicVaultRecordId: string;
  privateVaultRecordId?: string;
}
```

---

## 5. Forensic-grade event logger (core to your “evidence lattice”)

`packages/core-domain/src/events.ts`

```
// FORGE-LIVE protection banner above…

export enum ForensicEventType {
  STATION_CREATED = "STATION_CREATED",
  ARTIFACT_CREATED = "ARTIFACT_CREATED",
  ARTIFACT_VERSIONED = "ARTIFACT_VERSIONED",
  VAULT_WRITE = "VAULT_WRITE",
  LIVE_SESSION_STARTED = "LIVE_SESSION_STARTED",
```

```
  LIVE_SESSION_ENDED = "LIVE_SESSION_ENDED",
}

export interface ForensicEvent {
  id: string;
  type: ForensicEventType;
  occurredAt: string;   // ISO timestamp
  actorId: CreatorId;
  stationId?: StationId;
  artifactId?: ArtifactId;
  vaultRecordId?: string;
  metadata: Record<string, unknown>; // non-derivable but evidentiary
  uiStateHash?: string;
  serverStateHash?: string;
}

export interface ForensicEventLogger {
  record(event: ForensicEvent): Promise<void>;
}
```

---

## 6. Minimal in-memory implementation (for live demo + vaulting)

`packages/core-domain/src/index.ts`

```
// FORGE-LIVE protection banner above…

import {
  Station,
  StationId,
  StationType,
  Artifact,
  ArtifactId,
  ArtifactKind,
} from "./domain";
import { ForensicEvent, ForensicEventLogger, ForensicEventType } from "./events";

export class InMemoryForensicEventLogger implements ForensicEventLogger {
  private events: ForensicEvent[] = [];

  async record(event: ForensicEvent): Promise<void> {
    this.events.push(event);
    // In real implementation, this would:
```

```
    // - persist to storage
    // - mirror to archival verticals
    // - hash and anchor state
  }

  getAll(): ForensicEvent[] {
    return this.events;
  }
}

export class ForgeLiveCore {
  private stations = new Map<StationId, Station>();
  private artifacts = new Map<ArtifactId, Artifact>();

  constructor(private readonly logger: ForensicEventLogger) {}

  createStation(params: {
    id: StationId;
    type: StationType;
    name: string;
    slug: string;
    description: string;
    createdBy: string;
    createdAt: string;
  }): Station {
    const station: Station = {
      ...params,
      isLive: false,
    };

    this.stations.set(station.id, station);

    this.logger.record({
      id: `evt_${Date.now()}_${Math.random()}`,
      type: ForensicEventType.STATION_CREATED,
      occurredAt: params.createdAt,
      actorId: params.createdBy,
      stationId: station.id,
      metadata: {
        name: station.name,
        type: station.type,
        slug: station.slug,
      },
    });
```

```
    return station;
  }

  createArtifact(params: {
    id: ArtifactId;
    stationId: StationId;
    kind: ArtifactKind;
    title: string;
    summary: string;
    contentHash: string;
    createdBy: string;
    createdAt: string;
  }): Artifact {
    const artifact: Artifact = {
      ...params,
      version: 1,
    };

    this.artifacts.set(artifact.id, artifact);

    this.logger.record({
      id: `evt_${Date.now()}_${Math.random()}`,
      type: ForensicEventType.ARTIFACT_CREATED,
      occurredAt: params.createdAt,
      actorId: params.createdBy,
      stationId: params.stationId,
      artifactId: params.id,
      metadata: {
        kind: artifact.kind,
        title: artifact.title,
        contentHash: artifact.contentHash,
      },
    });

    return artifact;
  }
}
```

---

## 7. Simple bootstrap (first “live station” to be vaulted)

`scripts/bootstrap.ts`

```
// FORGE-LIVE protection banner above…

import { ForgeLiveCore, InMemoryForensicEventLogger } from "../packages/core-domain/src";
import { StationType, ArtifactKind } from "../packages/core-domain/src/domain";

async function bootstrap() {
  const logger = new InMemoryForensicEventLogger();
  const core = new ForgeLiveCore(logger);

  const now = new Date().toISOString();
  const creatorId = "creator:spencer";

  const station = core.createStation({
    id: "station:creation:001",
    type: StationType.CREATION,
    name: "FORGE-LIVE Genesis Creation Station",
    slug: "forge-live-genesis",
    description: "The first live, protected creation station of the FORGE-LIVE Multiverse.",
    createdBy: creatorId,
    createdAt: now,
  });

  core.createArtifact({
    id: "artifact:manifesto:001",
    stationId: station.id,
    kind: ArtifactKind.IDEA,
    title: "FORGE-LIVE Multiverse – Genesis Manifesto",
    summary: "Foundational definition of the live, protected, station-based innovation
ecosystem.",
    contentHash: "hash:to-be-replaced-with-real",
    createdBy: creatorId,
    createdAt: now,
  });

  console.log("Bootstrap complete. Forensic events:");
  console.log(JSON.stringify(logger.getAll(), null, 2));
}

bootstrap().catch((err) => {
  console.error("Bootstrap failed:", err);
  process.exit(1);
});
```

---

If you tell me:

- your preferred **language/runtime** (TypeScript/Node, Python, Rust, etc.)
- and whether this is going into **GitHub now**

I can next:

1. generate a **README.md** that frames this as “Genesis Code Vault – Round 1”, and
2. add a **Vault Forge Live metadata block** you can paste directly into the repo for provenance.