Solana Narrative Scanner & MemeCoin Launcher — Step-by-Step Development Roadmap

Goal: Build a system that scans social platforms for emerging narratives, ranks/alerts them, maps them to meme-coin ideas, and (optionally) deploys SPL tokens + LP on Solana with a human-in-the-loop.

---

0) Foundations & Repo Setup

Deliverables - Monorepo with packages: `/ingest`, `/nlp`, `/api`, `/dashboard`, `/bots`, `/onchain`, `/ops`. - Base infra: Docker, docker-compose, `.env.example`, Makefile or Taskfile. - Cloud baseline (optional early): Postgres, Redis, object store (S3), job runner (Celery/Redis or Temporal), metrics (Prometheus/Grafana), logging (ELK or OpenSearch). - Secrets via env or Vault; key management policy drafted.

Tech stack (suggested) - **Backend/API:** Python (FastAPI) or Node (NestJS).
- **NLP:** Python, spaCy + transformers (sentence-transformers/MPNet), BERTopic (UMAP+HDBSCAN) or custom TF-IDF + Agglomerative/HDBSCAN.
- **DB:** Postgres (+ TimescaleDB extension recommended).
- **Cache/Queues:** Redis; Celery/Arq/Temporal for jobs/schedules.
- **Frontend:** Next.js + Tailwind + shadcn/ui + Recharts.
- **Bots:** Telegram (python-telegram-bot), Discord (discord.py/discord.js).
- **On-chain:** Solana web3.js + Anchor (Rust program optional), SPL-Token + Metaplex metadata, Raydium SDK for LP.

Acceptance criteria - `docker compose up` boots DB, API skeleton (`/healthz`), and placeholder dashboard.

---

1) Data Ingestion Connectors (MVP scope first)

MVP Connectors - **X/Twitter:** official API or compliant firehose vendor; track keywords/hashtags + curated account lists.
- **Reddit:** subreddits (crypto, Solana, memes) via Reddit API; pull posts, comments, scores.
- **Telegram:** Telethon (logged-in user or bot) for chosen channels (respect TOS & consent).
- **Dex Feeds:** Dexscreener/Birdeye public endpoints for SOL pairs (price, volume, new listings).
- **TikTok:** trending hashtags, sounds, and popular creators via official/partner analytics APIs.

Stretch Connectors - Discord channels, YouTube shorts, Google Trends.

Data model

```sql
-- Raw mentions
CREATE TABLE mention (
  id BIGSERIAL PRIMARY KEY,
  source TEXT NOT NULL,                -- twitter, reddit, telegram,
dexscreener, tiktok
  source_id TEXT NOT NULL,              -- tweet_id, post_id, message_id,
video_id
  author TEXT,
  text TEXT,
  url TEXT,
  created_at TIMESTAMPTZ NOT NULL,
  metrics JSONB DEFAULT '{}'::jsonb,    -- likes, rts, replies, views, shares
  lang TEXT,
  entities JSONB DEFAULT '{}'::jsonb,   -- hashtags, tickers, urls, tokens,
sounds
  ingest_ts TIMESTAMPTZ DEFAULT now(),
  UNIQUE (source, source_id)
);

CREATE INDEX mention_created_idx ON mention (created_at);
CREATE INDEX mention_gin_text ON mention USING gin (to_tsvector('english',
text));
```

Acceptance criteria - Backfill+stream jobs write to `mention` with retry + idempotency.
- Rate-limit + error handling; connector health visible in `/ops/connectors` endpoint.

---

2) Normalization, Enrichment & Storage

Tasks - Language detection; basic cleaning (urls, emojis, dedup).
- NER/keyword extraction (hashtags, cashtags like `$WIF`, named entities, TikTok sounds).
- Link resolve (expand t.co, TikTok shortlinks).
- Persist enriched fields: tokens, hashtags, sentiment, embeddings.

Tables

```
CREATE TABLE mention_enriched (
  mention_id BIGINT PRIMARY KEY REFERENCES mention(id) ON DELETE CASCADE,
  sentiment REAL,                 -- -1..1
  embedding VECTOR(384),          -- pgvector
  keywords TEXT[],
  influencers_score REAL,         -- author influence percentile
  platform_features JSONB,        -- tiktok: sound_id, trending_score, shares
  toxicity REAL                   -- optional safety score
);
```

Acceptance criteria - Batch + streaming enrichment jobs complete <X minutes after ingest (no SLA here—just wire end-to-end).
- `/api/mentions?since=...` returns enriched payloads.

---

3) Topic Clustering & Narrative Tracking

Approach - Hourly (and rolling 10-min) clustering using BERTopic or:
1) Embed texts → 2) Dimensionality reduction (UMAP) → 3) HDBSCAN → 4) Label topics with keyphrases (YAKE/Maximal Marginal Relevance).
- Maintain **narrative objects** that persist over time by matching cluster centroids.

Tables

```
CREATE TABLE narrative (
  id BIGSERIAL PRIMARY KEY,
  label TEXT,                     -- e.g., "Shrek 2 meme", "SEC vs X", "WIF
dog"
  created_at TIMESTAMPTZ DEFAULT now(),
  last_seen TIMESTAMPTZ,
  centroid VECTOR(384),
  keywords TEXT[],
  category TEXT                   -- crypto, pop, tech, politics, ai, etc.
);

CREATE TABLE narrative_window_stats (
  narrative_id BIGINT REFERENCES narrative(id) ON DELETE CASCADE,
  window_start TIMESTAMPTZ,
  window_end TIMESTAMPTZ,
  mentions INT,
  unique_authors INT,
  avg_engagement REAL,
  growth_rate REAL,               -- % vs prior window
  sentiment REAL,
  sources JSONB,
  PRIMARY KEY (narrative_id, window_start)
);
```

Acceptance criteria - `/api/narratives/top?window=1h` returns ranked narratives with metrics.
- Topics persist across windows with >90% stable membership for steady narratives.

---

4) Virality & Launch-Readiness Scoring

Metrics per narrative (normalized 0..1) - `M` : mentions volume (log-scaled).
- `G` : growth rate (slope of EWMA of mentions).
- `E` : engagement (likes+reposts+comments per mention; TikTok adds shares/saves).
- `I` : influencer weight (share of top-percentile authors/creators).
- `S` : sentiment (shift & polarity).
- `N` : novelty (semantic distance to last 7-day topic set).
- `R` : recency (time since last spike).

Virality score

```
VS = 0.25*M + 0.25*G + 0.15*E + 0.15*I + 0.10*N + 0.10*R - 0.10*toxicity
```

Launch-readiness score (LRS) adds meme-ability heuristics: - `memefit`: presence of templates, puns, characters, icons.
- `copyright_risk`: down-weight obvious IP where distribution channels might block.
- `timeliness`: is the window still climbing?

```
LRS = 0.6*VS + 0.2*memefit - 0.2*copyright_risk
```

Acceptance criteria - Scores computed per 10-min window; thresholds produce stable Top-10 without oscillation.

---

5) Narrative → Coin Ideation

Tasks - Name/ticker generator: rule-based + LLM prompts (e.g., from keywords/entities, TikTok sound names).
- Branding kit: short description, tagline, emoji set, initial meme prompts.
- Risk flags: political sensitivity, impersonation, obvious IP claims.

API - `POST /api/coin_ideas` with narrative_id → returns 5–10 name/ticker combos + branding.

Acceptance criteria - At least one suggested ticker passes uniqueness check (not active SOL token/ticker).

---

6) Command Center Dashboard (Web)

Views 1) **Now**: Top narratives (1h/6h/24h), score sparkline, sources, sample posts/videos.
2) **Details**: Narrative page with momentum graph, sentiment trend, influencer list, similar narratives.
3) **Coin Studio**: select narrative → generate ideas → shortlist → export brief.
4) **Alerts**: log of spike alerts, status (acknowledged/launched).
5) **Launched**: internal tokens, price/volume/holders (via Birdeye/Dexscreener), PnL.

Frontend extras - Search, filters, saved views, dark mode, keyboard nav.
- Embedded TikTok preview where possible.

Acceptance criteria - 200ms p95 for cached Top-10 narratives; charts render under 2s with 10k points.

---

7) Bots & Alerts

Telegram/Discord bot commands - `/top` (window=1h|6h|24h), `/spikes`, `/crypto_only`, `/coinideas {narrative_id}`, `/watch add/remove`, `/alert set threshold`.
- Include TikTok narratives in `/top` with link to video(s).

Alerting - Thresholds on `G` and `VS` with hysteresis; dedupe across sources; cooldowns per narrative.

Acceptance criteria - Bot responds <3s with latest snapshot; alerts fire once per spike phase.

---

8) On-Chain: SPL Token Deployment (Human-in-the-Loop)

Flow 1) Approve narrative + idea in dashboard.
2) Generate deployment plan (supply, decimals, freeze/mint authority, initial distribution).
3) Create mint (SPL-Token), set metadata (Metaplex), optionally renounce authorities after setup.
4) Create liquidity on Raydium: create pool or seed existing; define initial price & amount.
5) Verify on explorers; post launch brief to bots.

Implementation - **Libraries:** `@solana/web3.js`, `@metaplex-foundation/js`, `@solana/spl-token`.
- **Safety gates:** manual confirmation, hardware wallet support, dry-run on devnet, fee/gas estimate & balance check, blacklist of disallowed words.

Config schema

```
{
  "name": "SWAMP",
  "symbol": "SWAMP",
  "decimals": 6,
  "initial_supply": 1000000000,
  "distribution": {"treasury": 0.1, "lp": 0.8, "marketing": 0.1},
  "lp": {"amm": "raydium", "base_amount": 800000000, "quote_amount_sol": 50},
  "authorities": {"mint": "retain|renounce", "freeze": "retain|renounce"}
}
```

Acceptance criteria - `POST /onchain/deploy` executes a dry-run; `POST /onchain/confirm` signs & broadcasts.
- Creates mint, metadata, and LP tx receipts saved in DB.

---

9) Meme Engine (Optional but Valuable)

Tasks - Generate meme copy (hooks, captions, CTAs).
- Create image prompts for top trending templates; render via image API; export social pack (PNG + captions).
- Include TikTok-style short captions, hook ideas, and soundtrack suggestions.

Acceptance criteria - Exportable ZIP with 5–10 memes per idea; autosized for X/TikTok/Telegram.

---

10) Backtesting & Evaluation

Data - Historical mentions + SOL token performance (Birdeye candles).
- Map spikes in narratives (including TikTok sounds/hashtags) to subsequent token pumps.

Metrics - **Lead time:** minutes between first spike alert and peak mentions.
- **Precision@K:** fraction of Top-K narratives that result in a token with >X volume within 24–48h.
- **Alert fatigue:** average alerts/day, dedupe rate.
- **Deployment outcomes:** post-launch volume/holders over 24h.

Acceptance criteria - Reproducible notebook producing the above metrics; stored charts in S3.

---

11) Security, Compliance & Ops

Security - Key management: no hot private keys in logs; optional HSM or Ledger.
- RBAC: only approved users can deploy on-chain.
- Audit trail: who approved what/when; immutable logs.

Compliance/Policy - Respect platform Terms of Service; use official APIs or licensed data vendors.
- No impersonation/celebrity likeness without parody disclosure; avoid infringing IP.
- Disclaimers on all public posts; mark tokens as experimental/meme.

Ops - Health endpoints, SLOs, alerts (ingest lag, queue depth, error rates).
- Cost controls: connector sampling, retention policies, tiered storage.

---

12) API Contract (initial)

```
GET  /healthz
GET  /narratives/top?window=1h&limit=20&category=crypto
GET  /narratives/{id}
GET  /mentions?narrative_id=...&since=...
POST /coin_ideas { "narrative_id": 123 }
POST /alerts { "narrative_id": 123, "threshold": {"VS": 0.7, "G": 0.6} }
POST /onchain/deploy { ...config }
POST /onchain/confirm { "draft_id": "..." }
GET  /tokens/launched
```

Response shape example

```
{
  "id": 123,
  "label": "shrek swamp meme",
  "scores": {"VS": 0.78, "LRS": 0.71},
  "stats": {
    "mentions": 1832,
    "growth_rate": 0.62,
    "sentiment": 0.31,
    "influencers": 0.27
  },
  "sources": {"twitter": 0.6, "reddit": 0.2, "telegram": 0.1, "tiktok": 0.1}
}
```

---

13) Testing & QA

- Unit tests: connectors, NLP, scoring, coin-idea generator.
- Integration tests: end-to-end ingest → narrative → alert.
- On-chain test: devnet deploy + LP simulation.
- Load tests: 1k msgs/sec synthetic stream; observe clustering time.

---

14) Rollout Plan

- **MVP Cut:** Reddit + Telegram + Dexscreener, TikTok trending hashtags, basic clustering, VS score, dashboard Top-10, Telegram bot `/top`, manual on-chain deploy (CLI).
- **V1:** Add X connector, LRS, Coin Studio, dry-run deploy + Raydium LP, backtesting.
- **V2:** Full bot suite, meme engine, influencer radar, advanced ops.

---

15) Tickets (first pass)

1. Bootstrap monorepo + CI.
2. Postgres schema + pgvector setup.
3. Reddit connector + tests.
4. Telegram connector + tests.
5. Dexscreener poller + tests.
6. TikTok connector + trending API integration.
7. Enrichment pipeline (lang detect, sentiment, keywords).
8. Embeddings service + cache.
9. Clustering job + persistence.
10. Narrative scoring service (VS, LRS).
11. FastAPI routes for narratives/mentions.
12. Next.js dashboard (Top view).
13. Telegram bot `/top` + `/spikes`.
14. Coin idea generator + uniqueness check.
15. On-chain devnet deploy (mint + metadata).
16. Raydium LP dry-run.
17. Backtesting notebook + metrics.
18. RBAC + audit trail.
19. Alerting thresholds + hysteresis.
20. Ops dashboards (ingest lag, error rate).
21. Security review + key mgmt.

Notes & Gotchas

- Avoid rate-limit bans: batch requests, exponential backoff, vendor SLAs.
- Topic drift: use centroid tracking + Jaccard overlap of keyword sets.
- False positives: require cross-source confirmation before high VS/LRS.
- Legal/IP: steer clear of direct brand/celebrity misrepresentation.

Outcome: A production