Parlay King — Automated Pipeline Run Reference

June 21, 2026 | Successful 3:30 AM ADT Run | V3-18 Gold Standard

Document Purpose: This is the definitive step-by-step reference of a successful pipeline run. Use this document to diagnose and repair the pipeline if it ever breaks. Every step is documented with exact timestamps, log messages, expected outputs, and failure indicators.

Pipeline Architecture Overview

The automated system runs in **two stages** triggered by two separate cron jobs:

Stage	Cron Time	Script	Purpose
Stage 1	2:00 AM ADT	`stage1_wrapper.sh` → `v3_history_prefetch.py`	Pre-fetch 90-day team history, odds pre-cache, Pinnacle validator
Stage 2	3:30 AM ADT	`cron_wrapper.sh` → `v3_pipeline.py`	Main pipeline: scrape games, Groq AI analysis, write picks.json
Fallback	6:30 AM ADT	`cron_wrapper.sh` (fallback guard)	Re-runs Stage 2 if 3:30 AM run failed to produce today's picks

---

STAGE 1 — Pre-Fetch (2:00 AM ADT)

Step 1A — F13 Odds Pre-Cache (02:30 ADT)

Script: `v3_history_prefetch.py` (no flag) **What it does:** Queries the Odds API for all 49 soccer leagues and caches today's odds into `f13_cache.json`. This is the "F13 Pre-Fetch" — it runs before the main pipeline so odds are available instantly when the pipeline runs at 3:30 AM.

Expected log output (stage1_prelim.log):

```
2026-06-21 02:30:35 [F13-PREFETCH] ═══ F13 PRE-FETCH START (FULL COVERAGE
2026-06-21 02:30:35 [F13-PREFETCH] Target date: 2026-06-21 | Time: 03:30 A
2026-06-21 02:30:47 [F13-PREFETCH]   ✅ [Brazil Serie B] Cuiabá vs Avai —
2026-06-21 02:30:56 [F13-PREFETCH]   ✅ [FIFA World Cup] Saudi Arabia vs S
2026-06-21 02:30:56 [F13-PREFETCH]   ✅ [FIFA World Cup] Iran vs Belgium —
2026-06-21 02:30:56 [F13-PREFETCH]   ✅ [FIFA World Cup] Cape Verde vs Uru
2026-06-21 02:30:56 [F13-PREFETCH]   ✅ [FIFA World Cup] Egypt vs New Zeal
```

Normal failures (non-fatal): `All retry attempts failed for soccer_scotland_premiership` — leagues with no active games return 404/empty. This is expected and non-fatal. **Output file:** `/home/soccsbur/scripts/f13_cache.json`

Step 1B — History Prefetch — Domestic Teams (02:30 ADT)

Script: `v3_history_prefetch.py` (no flag, runs inside main() after F13) **What it does:** For every league that has games today, fetches 90 days of match history for each team and computes weighted statistics. Writes to `history_cache.json`.

Expected log output (stage1_prelim.log):

```
2026-06-21 02:31:23 [prefetch] INFO    Brazil Serie B: 8 teams today — sca
2026-06-21 02:31:47 [prefetch] INFO      Avaí (AVA): 12 games | Over2.5(w)
2026-06-21 02:31:47 [prefetch] INFO      Cuiabá (CUI): 12 games | Over2.5(
2026-06-21 02:31:51 [prefetch] INFO    WNBA: 6 teams today — scanning 90 d
2026-06-21 02:32:14 [prefetch] INFO      Los Angeles Sparks (LA): 17 games
```

Output file: `/home/soccsbur/scripts/history_cache.json` **Key structure:** `{ "teams": { "soccer/bra.2:AVA": {...}, "soccer/bra.2:CUI": {...} }, "intl_teams": {...}, "meta": {...} }`

Failure indicator: If `history_cache.json` is empty or has 0 teams — the pipeline will run in BLIND_START mode (uses league averages instead of real team history). Picks will still be generated but with lower accuracy.

Step 1C — Pinnacle Validator (02:31 ADT)

Script: `pinnacle_validator.py` (called by stage1_wrapper.sh) **What it does:** Fetches Pinnacle sharp-money lines for all active leagues and flags games where the public odds exceed

the Pinnacle ceiling by more than +130. These games are marked as “ceiling blocks” and the pipeline will avoid picking them.

Expected log output:

```
2026-06-21T02:31:57Z [PINNACLE] INFO: pinnacle_lines.json written: 20 game
2026-06-21T02:31:57Z [PINNACLE] INFO: shadow_log.json: appended 20 Pinnacl
2026-06-21T02:31:57Z [PINNACLE] INFO: ═══ Pinnacle Validator COMPLETE ═══
```

Output file: `/home/soccsbur/scripts/pinnacle_lines.json`

Step 1D — International Sidecar (02:32 ADT)

Script: `v3_history_prefetch.py --intl` → `v3_international.py` **What it does:** Fetches today’s international games (FIFA World Cup, Copa America, etc.) from ESPN and runs a separate Groq analysis pass. Writes results to `picks_intl.json`.

Expected log output (v3_international.log):

```
2026-06-21 02:32:xx [INTL] FIFA World Cup games found: 4
2026-06-21 02:32:xx [INTL] picks_intl.json written: X picks
```

Output file: `/home/soccsbur/scripts/picks_intl.json`

STAGE 2 — Main Pipeline (3:30 AM ADT)

Step 2A — Mutex Lock Check

Script: `cron_wrapper.sh` **What it does:** Checks for `/tmp/cron_pipeline.lck`. If the lock file exists and the PID inside is still running, exits immediately to prevent double-runs. If stale, removes it and continues.

Expected cron_debug.log:

```
Sun Jun 21 03:30:34 ADT 2026 FALLBACK GUARD: picks.json date=2026-06-20, t
```

(Note: On June 21 the 3:30 AM cron did not fire — the fallback guard at 6:30 AM triggered the run instead. The pipeline ran at 06:30 UTC = 03:30 ADT via the fallback.)

Failure indicator: If you see `MUTEX: already running (PID XXXXX)` — a stale lock file is blocking the run. SSH to server and run `rm /tmp/cron_pipeline.lck`.

Step 2B — V3-18 Pipeline START

Script: `v3_pipeline.py` **What it does:** Loads environment variables, Groq API keys, and history cache. Logs the pipeline version and Halifax date.

Expected log output (v3_pipeline.log):

```
2026-06-21 02:31:57,728  INFO  ═══ V3-15 Pipeline START -- 2026-06-21 (Hal
2026-06-21 02:31:57,728  INFO  Groq keys loaded: 3
```

(Note: Log says V3-15 because the log file is shared — the actual script running is v3_pipeline.py which is V3-17/V3-18. The version number in the log reflects the logger name, not the script version.)

Failure indicator: If no START line appears — the script crashed on import. Check for `IndentationError`, `SyntaxError`, or missing `.env` file.

Step 2C — Game Scraping (Odds API Primary + TSDB Supplement)

Script: `v3_pipeline.py` → `scrape_all_games()` **What it does:**

1. Queries Odds API for all leagues in `ODDS_SOCCER_LEAGUES` (49 leagues)
2. Calls TSDB Premium `eventsday` for supplementary games not found by Odds API
3. Deduplicates by team name pair
4. Fetches WNBA and NBA games from ESPN

Expected log output:

```
2026-06-21 02:32:01,732  INFO    Soccer [Brazil Serie B]: 4 games
2026-06-21 02:32:05,317  INFO    Soccer [FIFA World Cup]: 4 games
2026-06-21 02:32:05,538  INFO    Soccer [Sweden Superettan]: 5 games
2026-06-21 02:32:08,002  INFO  [TSDB] Added 37 supplementary soccer games
2026-06-21 02:32:08,364  INFO    NBA [WNBA]: 1 games
2026-06-21 02:32:08,507  INFO    NBA [NCAA]: 1 games
2026-06-21 02:32:08,507  INFO  Total scraped — Soccer: 50, MLS: 0, NBA: 1
```

Failure indicator: If `Total scraped — Soccer: 0` — Odds API key is exhausted or TSDB key is invalid. Check `.env` for `THE_ODDS_API_KEY` and `TSDB_API_KEY`.

Step 2D — Odds Enrichment

Script: `v3_pipeline.py` → `fetch_odds_dual()` **What it does:** For each scraped game, fetches the best available odds from the Odds API across all bookmakers. Injects `home_odds`, `away_odds`, `draw_odds`, and `best_market` into each game object.

Expected log output:

```
2026-06-21 02:32:23,841  INFO  [ODDS] PRIMARY — fetched 167 total soccer e
2026-06-21 02:32:23,842  INFO  [ODDS] Enriched 8/50 soccer games with real
```

Note: Only $^{8}/_{50}$ games enriched is normal — most TSDB supplementary games are from leagues the Odds API doesn't cover. The 4 Brazil Serie B games and 4 FIFA World Cup games get full odds enrichment.

Step 2E — Groq Pass 1 (AI Analysis)

Script: `v3_pipeline.py` → `run_groq_batch()` **What it does:** Sends all 50 soccer games to Groq LLM with the V3-18 prompt. Groq evaluates each game across 10 factors (F1–F14) and returns picks with confidence scores. Only picks ≥68% confidence survive Pass 1.

Expected log output:

```
2026-06-21 02:32:24,182  INFO  Sending 50 soccer games to Groq (Pass 1)...
2026-06-21 02:32:24,184  INFO  Groq [SOCCER]: trying PRIMARY key...
2026-06-21 02:32:28,260  INFO  Groq: response received (3402 chars)
2026-06-21 02:32:28,261  INFO  Groq [SOCCER]: PRIMARY succeeded -- 4 picks
2026-06-21 02:32:28,261  INFO  Soccer Pass 1 survivors (≥68%): 4
```

Failure indicator: If `Groq [SOCCER]: PRIMARY failed` — pipeline tries backup keys automatically. If all 3 keys fail, soccer picks will be 0. Check Groq API key quota at `console.groq.com`.

Step 2F — Groq Pass 2 (Dual-Brain Confirmation)

Script: `v3_pipeline.py` → dual-brain confirmation loop **What it does:** Each Pass 1 survivor is sent to a second Groq call for independent confirmation. A pick must be confirmed by both brains to proceed. This eliminates false positives.

Expected log output:

```
2026-06-21 02:32:31,961  INFO  Groq: response received (2283 chars)
2026-06-21 02:32:31,962  INFO  [PASS2] SOCCER: 4/4 picks confirmed by dual
2026-06-21 02:32:31,962  INFO  [V3-17] Soccer: 4/4 picks passed F13+F14+10
2026-06-21 02:32:31,962  INFO  Soccer picks after dual-brain confirmation:
```

Step 2G — NBA/WNBA Analysis

Script: `v3_pipeline.py` → `run_groq_batch()` (NBA mode) **What it does:** Same Pass 1 + Pass 2 process for NBA/WNBA games.

Expected log output:

```
2026-06-21 02:32:31,962  INFO  Sending 1 NBA games to Groq (Pass 1)...
2026-06-21 02:32:36,380  INFO  Groq [NBA]: PRIMARY succeeded -- 2 picks
2026-06-21 02:32:36,380  INFO  NBA Pass 1 survivors (≥68%): 2
2026-06-21 02:32:39,354  INFO  [PASS2] NBA: 1/2 picks confirmed by dual-br
2026-06-21 02:32:39,354  INFO  [V3-17] NBA: 1/1 picks passed F13+F14+10-fa
```

Step 2H — Guardrail Validation

Script: `v3_pipeline.py` → guardrail check **What it does:** Validates that the pipeline has produced enough picks to publish. Requires minimum 3 soccer picks and 1 NBA pick (if NBA games are available). If guardrail fails, the pipeline aborts and does NOT overwrite picks.json.

Expected log output:

```
2026-06-21 02:32:39,354  INFO  Guardrail: 0 international picks available
2026-06-21 02:32:39,354  INFO  Guardrail -- Soccer: 4 domestic + 0 intl =
2026-06-21 02:32:39,355  INFO  GUARDRAIL PASSED: Total=5 picks (Soccer=4 N
```

Failure indicator: `GUARDRAIL FAILED` — pipeline will NOT write picks.json. The previous day's picks remain live. This is intentional — better to show yesterday's picks than publish 0 picks.

Step 2I — Combiner / Picks Writer

Script: `v3_combiner.py` (called by cron_wrapper.sh after pipeline exits) **What it does:** Merges domestic soccer picks + international picks, applies F11 urgency gate to NBA picks, rebuilds `three_leg_conservative` (top 3 soccer), `nba_parlay`, and `mls_parlay` structures, injects Kelly units, and atomically writes `picks.json` to both `/public_html/picks.json` and `/public_html/data/picks.json`.

Expected log output:

```
2026-06-21 02:32:44,708 [COMBINER] Merged: 4 main soccer + 0 intl = 4 tota
2026-06-21 02:32:44,708 [COMBINER] Hybrid order preserved: safety_anchor 9
2026-06-21 02:32:44,708 [COMBINER] [F11 GATE] Rejected 1 NBA pick(s) with
2026-06-21 02:32:44,709 [COMBINER] FIX GAP4: Rebuilt nba_parlay with 0 leg
2026-06-21 02:32:44,710 [COMBINER] Atomic write: /home/soccsbur/public_htm
2026-06-21 02:32:44,713 [COMBINER] history_log.csv: appended 4 picks for 2
2026-06-21 02:32:44,713 [COMBINER] Merge complete -- 4 soccer (0 intl, 0 a
```

Step 2J — Stage 2 F13 Personnel Check

Script: `stage2_f13_check.py` **What it does:** Post-publish safety check. Re-reads picks.json and verifies that no published pick has a key player injury or suspension that was reported after the pipeline ran. Removes affected picks if found.

Expected log output:

```
2026-06-21 02:32:44,760 [STAGE2-F13] F13 check: 0 global picks checked | 0
2026-06-21 02:32:44,760 [STAGE2-F13] No picks removed — picks.json unchang
2026-06-21 02:32:44,761 [STAGE2-F13] Pulse log updated: stage=STAGE2_F13 s
```

Step 2K — Kelly Optimizer

Script: `kelly_optimizer.py` **What it does:** Calculates optimal bet sizing for each pick using the Kelly Criterion (25% fractional Kelly). Injects `kelly_unit` and `tier` into each pick in picks.json.

Expected log output:

```
2026-06-21 02:33:17 [KELLY] Kelly Fraction: 25% | Max Unit: 1.0u | Min Uni
2026-06-21 02:33:17 [KELLY]   Spain vs Saudi Arabia | conf=90% | odds=-125
2026-06-21 02:33:17 [KELLY]   Uruguay vs Cape Verde | conf=85% | odds=-235
2026-06-21 02:33:17 [KELLY]   Avai vs Cuiabá | conf=70% | odds=+132 | unit
2026-06-21 02:33:17 [KELLY]   Los Angeles Sparks vs New York Liberty | con
2026-06-21 02:33:17 [KELLY] kelly_units section injected into picks.json
```

Note: Spain @ -1250 odds gets `unit=0.0u | PASS` — Kelly correctly identifies that at -1250 odds the EV edge is too small to warrant a bet even at 90% confidence.

Step 2L — Cards Engine

Script: `cards_engine.py` **What it does:** Analyzes cards/corners data for each pick and generates supplementary cards market picks. Currently runs in `DRY_RUN=True` mode — no changes to picks.json.

Expected log output:

```
2026-06-21 02:33:18 [CARDS_ENGINE] V3-18 DYNAMIC CARDS ENGINE — DRY_RUN=Tr
2026-06-21 02:33:18 [CARDS_ENGINE] SUMMARY: 0/3 games generated a card pic
2026-06-21 02:33:18 [CARDS_ENGINE] DRY_RUN=True — NO changes made to picks
```

Note: Cards Engine skips World Cup games due to missing team IDs in the ESPN format. This is expected.

Step 2M — Meta Tag Writer

Script: `update_meta_tags.py` **What it does:** Updates `index.html` Open Graph meta tags, Twitter card tags, and page title with today's picks for SEO and social sharing.

Expected log output:

```
2026-06-21 02:33:18 [META_WRITER] Found 4 picks for 2026-06-21
2026-06-21 02:33:18 [META_WRITER] New og:title:  NBA & Soccer Parlay Picks
2026-06-21 02:33:18 [META_WRITER] index.html updated successfully (10 meta
```

Step 2N — Expert Analysis Generator

Script: `daily_expert_analysis.py` **What it does:** Reads the highest-confidence pick from picks.json and generates a full expert analysis article using Groq. Writes to `expert_analysis.json` which the site displays in the "Expert Deep Analysis" section.

Expected log output:

```
[INFO] Best pick: Saudi Arabia @ Spain | conf=90.0% | pick=Spain or Draw (
[INFO] Groq success — 1230 tokens used
[DONE] Expert analysis published: Saudi Arabia @ Spain — Expert Deep Analy
```

Output file: `/home/soccsbur/public_html/expert_analysis.json`

Step 2O — Fallback Guard (6:30 AM ADT)

Script: `cron_wrapper.sh` fallback guard **What it does:** Checks if picks.json was updated today. If not (meaning the 3:30 AM run failed), re-runs the full pipeline immediately.

Expected log output (success — no re-run needed):

```
Sun Jun 21 06:30:30 ADT 2026 FALLBACK GUARD: picks.json already generated
```

Expected log output (failure — re-run triggered):

```
Sun Jun 21 06:30:34 ADT 2026 FALLBACK GUARD: picks.json date=2026-06-20, t
```

Final Output — picks.json Structure

After a successful run, `picks.json` contains:

```
{
  "date": "2026-06-21",
  "three_leg_conservative": {
    "legs": [
      { "match": "Saudi Arabia @ Spain", "pick": "Spain or Draw (1X)", "co
      { "match": "Cape Verde @ Uruguay", "pick": "Uruguay or Draw (1X)", "
      { "match": "Cuiabá @ Avai", "pick": "Avai ML", "confidence": 70, "le
    ]
  },
  "nba_parlay": {
    "legs": [
      { "match": "New York Liberty @ Los Angeles Sparks", "pick": "Over +1
    ]
  },
  "kelly_units": { ... },
  "expert_analysis": { ... }
}
```

Quick Diagnostic Checklist

If the pipeline fails, check these in order:

Check	Command / Location	Expected
1. Mutex lock stuck	`ls /tmp/cron_pipeline.lck`	File should NOT exist after 6:30 AM
2. picks.json date	`cat /public_html/picks.json | grep date`	Should be today's date
3. Groq keys working	`cron_debug.log` → search "Groq [SOCCER]"	Should say "PRIMARY succeeded"
4. Odds API quota	`cron_debug.log` → search "ODDS API"	Should not say "quota exceeded"
5. History cache populated	`scripts/history_cache.json` → check `teams` key	Should have 200+ team entries
6. Guardrail passed	`cron_debug.log` → search "GUARDRAIL"	Should say "GUARDRAIL PASSED"
7. picks.json written	`cron_debug.log` → search "Atomic write"	Should say "Atomic write: …/picks.json"
8. Expert analysis updated	`public_html/expert_analysis.json` → check date	Should be today's date

Key File Locations on Server

File	Path	Purpose
Main pipeline	`/home/soccsbur/scripts/v3_pipeline.py`	Core AI pick generation
Cron wrapper	`/home/soccsbur/scripts/cron_wrapper.sh`	Stage 2 orchestrator
Stage 1 wrapper	`/home/soccsbur/scripts/stage1_wrapper.sh`	Stage 1 orchestrator
History prefetch	`/home/soccsbur/scripts/v3_history_prefetch.py`	Team history cache builder
Combiner	`/home/soccsbur/scripts/v3_combiner.py`	Picks merger and writer
Safety filter	`/home/soccsbur/scripts/safety_filter.py`	Pick validation
Expert analysis	`/home/soccsbur/scripts/daily_expert_analysis.py`	Article generator
Pipeline log	`/home/soccsbur/scripts/v3_pipeline.log`	Main run log
Cron debug log	`/home/soccsbur/scripts/cron_debug.log`	Full orchestration log
Stage 1 log	`/home/soccsbur/scripts/stage1_prelim.log`	Prefetch log
History cache	`/home/soccsbur/scripts/history_cache.json`	90-day team stats
F13 cache	`/home/soccsbur/scripts/f13_cache.json`	Pre-cached odds

File	Path	Purpose
Picks output	`/home/soccsbur/public_html/picks.json`	Live site picks
Expert analysis	`/home/soccsbur/public_html/expert_analysis.json`	Live site analysis

Document generated: June 21, 2026 | Based on actual log data from the successful 3:30 AM ADT run