AWS Recovery Point Copy – Domain Summary (Current Iteration)

Primary Actor

- AWS Administrator with access to two AWS accounts (source with ~1M recovery points across ~12 vaults; destination replicated account).

Domain Overview

- Recovery points span 8 AWS resource types (EBS, RDS, Aurora, S3, EFS…).
- Goal: replicate vault structure to another account, same region, filtering recovery points by APMID.
- Metadata enrichment via external CSV keyed by resourceArn.

Functional Requirements

1. Inventory & Metadata Enrichment

- Enumerate all recovery points with complete pagination.
- Join external CSV metadata using resourceArn.
- Use metadata (APMID and others) to tag or filter recovery points before copying.

2. Copy Orchestration

- Replicate vault structure like-for-like across accounts.
- Exclude recovery points whose APMID does not match allowed sets.
- Maintain region parity; no cross-region copies.
- Apply required compliance flags from source vaults to destination vaults.

3. Permissions & Compliance Validation

- Module to pre-check all required IAM permissions.
- Module to validate compliance settings match production expectations.
- Test environment must include representative dummy data covering all resource types.

4. Token Lifecycle Management

- External tool manages auth token in ~/.aws/credentials.
- On any expired token error:
- Destroy AWS clients.
- Reload credentials and retry.
- After 3 consecutive auth failures:
- Pause execution.
- Notify operator to refresh token.

- Resume on keypress.

5. Long-Running Reliability

- Job may run for hours or days.
- Operator may be idle (sleep, gone from console).
- Program must pause safely on token lapses.
- Ctrl-C triggers graceful shutdown with state saved.

6. Stateful Persistence & Resumability

- Persist all progress (inventory, copy queue, completed items).
- On startup: default to resume mode.
- Allow operator to regenerate or reset portions of state if needed.

7. Scheduling Constraints (Change Windows)

- Operator provides a maximum runtime window.
- Program must exit gracefully when time limit reached.
- State must support picking up exactly where it left off in the next window.
- Program should provide data to forecast total time needed for completion.

8. Performance & Parallelism

- Multiple worker threads should improve throughput.
- Safe concurrency around token refresh, queues, and state writes.
- Mechanisms for avoiding double-processing or missed items.

9. Operator Feedback

- Continuous console output showing progress, rates, remaining items.

Test Environment Strategy (80/20 Approach)

- Goal: create minimal viable test environment that reflects:
- All resource types.
- Vault structure diversity.
- Metadata enrichment scenarios (valid/missing APMID, mismatches).
- Permission and compliance edge cases.
- Focus on:
- Generating dummy recovery points per resource type.

- Creating representative vaults with compliance flags.
- Producing a realistic CSV metadata file.
- Ensuring IAM roles cover all required actions.
- Avoid recreating full production scale; instead aim for representative shape and behavior.