Pandacea Protocol - MVP Development - Sprint 1 Plan (v1.1)

Sprint:	1 (MVP Bootstrap)
Duration:	2 Weeks
Status:	**To Do**
Lead Engineer:	Asa Conant
Related Documents:	SDD (v1.1), API Spec (v1.1), Engineering Playbook (v1.1)

1. Sprint Goal

The goal of Sprint 1 is to establish the foundational infrastructure for the protocol and build the "scaffolding" for the core components. By the end of this sprint, we will have a running internal testnet and the initial, verifiable interaction between a Spender and an Earner agent.

2. Key Deliverables

1. A running internal testnet environment on the Polygon Mumbai testnet with the initial LeaseAgreement contract deployed.
2. CI/CD pipelines in GitHub Actions that automatically test and lint all core repositories (Smart Contracts, Agent Backend, Builder SDK).
3. A basic MyData Agent (Earner) backend, discoverable on the P2P network, that responds to API requests for its data products.
4. A basic Builder SDK (Spender) that can discover the Earner agent and successfully retrieve its list of data products.

3. Sprint Backlog

Sprint Strategy & Dependencies

- **Week 1 Priority:** The highest priority is to complete AGENT-1, AGENT-2, and INFRA-4.
- **Handoff:** Once the MyData Agent is deployed to the internal testnet, its static Peer ID and URL will be pinned in the team's engineering channel. This will unblock the SDK developer (SDK-2) and the integration test developer

(INTEGRATION-1).

Track 1: DevOps & Infrastructure

- **[INFRA-1] Setup Smart Contract CI:**
  - **Description:** Create the GitHub Actions workflow to compile, lint (Solhint), and run tests (Foundry) for the smart contracts repository on every pull request.
- **[INFRA-2] Create Local Dev Environment:**
  - **Description:** Develop and document a docker-compose.yml file that spins up a complete local development environment, including a local blockchain node (Anvil), an IPFS node, and placeholders for the agent backends.
- **[INFRA-3] Deploy to Testnet:**
  - **Description:** Create and document the deployment scripts (using Foundry) to deploy the v1.0 smart contracts to the Polygon Mumbai testnet.
- **[INFRA-4] Setup Agent Backend CI:**
  - **Description:** Create the GitHub Actions workflow to lint (golangci-lint) and run unit tests (go test) for the Go agent backend repository.
- **[INFRA-5] Setup Builder SDK CI:**
  - **Description:** Create the GitHub Actions workflow to lint (flake8) and run unit tests (pytest) for the Python SDK repository.

Track 2: Smart Contracts (Solidity)

- **[SC-1] Implement LeaseAgreement v1:**
  - **Description:** Implement the initial LeaseAgreement.sol contract with the createLease and approveLease functions. Include basic stubs for interactions with other protocol contracts.
  - **Acceptance Criteria:** The contract is implemented as per the SDD and compiles without errors.
- **[SC-2] Unit Test LeaseAgreement:**
  - **Description:** Write comprehensive unit tests for the LeaseAgreement contract using Foundry, covering all success and failure cases.
  - **Acceptance Criteria:** Test coverage for the contract exceeds 80%.

Track 3: Agent Backend (Go)

- **[AGENT-1] Setup P2P Node:**
  - **Description:** Create the basic Go application for the MyData Agent that initializes a libp2p host, joins the KAD-DHT, and can be discovered by other peers.
  - **Acceptance Criteria:** The agent starts, logs its Peer ID, and can be successfully pinged by another libp2p node.

- **[AGENT-2] Implement Product Discovery Endpoint:**
    - **Description:** Implement the GET /api/v1/products endpoint on the agent.
    - **Acceptance Criteria:** The endpoint returns a 200 OK with a hardcoded list of mock DataProduct objects that conform to the API specification.
- **[AGENT-3] Implement Basic Lease Request Endpoint:**
    - **Description:** Implement the POST /api/v1/leases endpoint.
    - **Acceptance Criteria:**
        1. The endpoint validates that the request body contains a valid productId.
        2. On success, it returns a 202 Accepted response with a dummy leaseProposalId.
        3. If validation fails, it returns a standard 400 Bad Request error object.

Track 4: Builder SDK (Python)

- **[SDK-1] Project Scaffolding:**
    - **Description:** Set up the Python project for the SDK using Poetry for dependency management and pytest for testing.
- **[SDK-2] Implement Agent Discovery Function:**
    - **Description:** Implement the spender_agent.discover() function.
    - **Acceptance Criteria:** The function takes an Earner's Peer ID as input, connects to its agent, successfully calls the GET /api/v1/products endpoint, and returns the parsed list of data products.

Track 5: Integration & Verification

- **[INTEGRATION-1] E2E Discovery Test:**
    - **Description:** Create an integration test that uses the Builder SDK to connect to a running instance of the MyData Agent on the internal testnet and verify the end-to-end discovery flow.
    - **Acceptance Criteria:**
        1. The test script successfully initializes the SDK.
        2. It connects to the deployed MyData Agent using its pinned Peer ID.
        3. It successfully retrieves and validates the structure of the mock data products.
        4. The test is added to a new integration test suite in the CI pipeline.