

Chain Graduation & Deployment PRD

1. Feature Introduction

Overview

The Chain Graduation & Deployment system is the automated infrastructure that transitions virtual chains from the launchpad's bonding curve phase to fully operational Nested Chains on Canopy. It orchestrates the complex process of chain deployment, validator bootstrapping, liquidity migration, and code upgrades while ensuring zero downtime and maintaining user positions throughout the transition.

Problem Statement

Currently, there's no seamless mechanism to transition from price discovery to actual blockchain deployment. Projects face significant technical hurdles in deploying infrastructure, bootstrapping validators, migrating liquidity, and implementing custom functionality. This gap between virtual launch and operational chain creates friction, delays, and potential failure points that can doom otherwise viable projects.

2. Principles & Objectives

Goals

- Seamless Transition: Zero-downtime migration from virtual to operational chain
- Automated Deployment: One-click chain deployment upon reaching graduation threshold
- Liquidity Preservation: Automatic migration of all liquidity to AMM pools
- Template Compliance: Ensure all chains deploy with validated, working configurations
- Developer Empowerment: Enable custom code deployment post-graduation

Success Metrics (KPIs)

- Graduation success rate > 99.9%
- Time from threshold to operational chain < 5 minutes
- Zero liquidity loss during migration
- Post-graduation chain survival rate (30 days) > 80%

Non-Goals

- Supporting non-template based deployments initially
- Manual intervention in graduation process

- Custom graduation thresholds per project

3. Users

Primary Personas

1. Chain Developer/Founder

- Background: Project team that launched on the launchpad
- Technical Level: Varies (low to high)
- Needs: Smooth deployment, retained liquidity, custom code activation
- Pain Points: Technical complexity, validator recruitment, post-launch management

2. Virtual Pool Participants

- Background: Early supporters who bought during bonding curve
- Technical Level: Low to medium
- Needs: Preserved token positions, continued liquidity, seamless transition
- Pain Points: Lost funds during migration, broken liquidity, unclear process

3. Potential Validators

- Background: Node operators looking for new chains to validate
- Technical Level: High
- Needs: Clear onboarding, economic incentives, technical documentation
- Pain Points: Unclear requirements, complex setup, unprofitable chains

Key User Stories

1. As a chain developer, I want my virtual chain to automatically deploy when it reaches the graduation threshold so that I can focus on building my application rather than infrastructure.
2. As a virtual pool participant, I want my tokens and liquidity to seamlessly transition to the deployed chain so that I maintain my position, earn fees, and can continue trading.
3. As a potential validator, I want clear information and incentives to join newly graduated chains so that I can profitably participate in chain security.

4. Requirements

P0 - Must Have

Graduation Trigger System

- Monitor market cap thresholds in real-time
- Validate graduation eligibility (template compliance, minimum liquidity)

- Lock virtual pool upon threshold achievement
- Initiate automated deployment sequence
- Notify all stakeholders of graduation status (launcher, holders, anyone who owned the token at one point during the curve)

Chain Deployment Engine

- Deploy template-based chain configuration
- Initialize genesis state with token distribution (as determined by bonding curve)
- Set up network endpoints and RPC nodes

Liquidity Migration

- Calculate final bonding curve state
- Mint equivalent tokens on new chain
- Create CNPY/Token AMM pool
- Transfer all liquidity to AMM
- Issue LP “tokens” to participants

Validator Bootstrapping

- Deploy initial validator node (Foundation-operated)
- Distribute validator incentives/rewards
- Monitor validator joining and stake
- Ensure minimum validator set before activation (1 or greater)

Code Upgrade System

- Connect to developer's GitHub repository
- Schedule upgrade at specified block height
- Execute chain upgrade

P1 - Great to Have

Advanced Deployment Features

- Multi-validator initial deployment (paid feature?)

Developer Tools

- Chain management dashboard
- Performance monitoring tools
- Upgrade testing environment
- Debug and logging access

Participant Experience

- Detailed graduation notifications
- Position tracking through transition

- Automatic wallet configuration
- Trading continuity guarantees
- Historical data preservation

P2 - Nice to Have

Validator Incentives

- Automated validator rewards program
- Stake matching for early validators
- Performance-based bonuses
- Validator ranking system
- Delegation recommendations

Analytics & Monitoring

- Graduation success metrics
- Chain health monitoring
- Validator performance tracking
- Economic security analysis
- Growth trajectory projections

P3 - If We Have Time

Advanced Features

- Custom graduation triggers
- Gradual deployment options
- Beta/testnet deployment first
- Governance-controlled graduation
- Emergency pause mechanisms
- Rollback capabilities

5. Technical Architecture

System Components

Graduation Controller

- Threshold monitoring service
- Eligibility validation engine
- State snapshot system
- Deployment orchestrator
- Event notification service

Deployment Infrastructure

- Node provisioning service
- Network configuration manager
- DNS and endpoint setup

Migration Engine

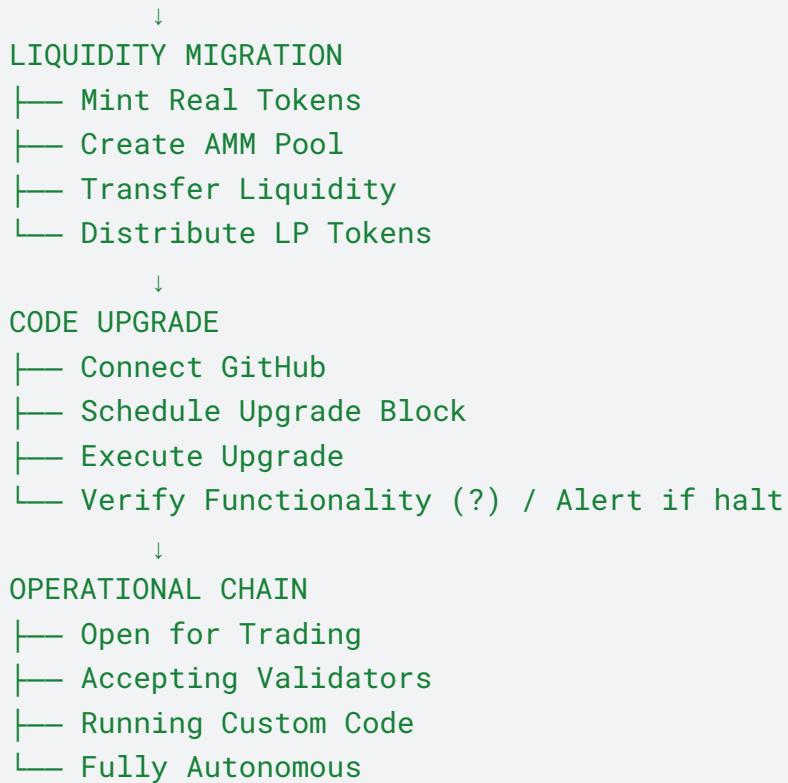
- Token minting service
- Liquidity calculation engine
- AMM pool creator
- LP token distributor
- State verification system

Upgrade Manager

- GitHub integration service
- Code validation system
- Upgrade scheduler
- Consensus coordinator
- Rollback controller (if possible)

State Transition Flow

```
None
VIRTUAL CHAIN STATE
├─ Bonding Curve Active
├─ Virtual Token Supply: X
├─ CNPY in Pool: Y
└─ Participants: List [Address]
    ↓
GRADUATION TRIGGERED
├─ Lock Virtual Pool
├─ Snapshot Final State
├─ Calculate Token Distribution
└─ Prepare Genesis State
    ↓
CHAIN DEPLOYMENT
├─ Deploy Infrastructure
├─ Initialize Validator(s)
├─ Create Genesis Block
└─ Start Network
```



Integration Requirements

Launchpad Integration

- Graduation threshold monitoring
- Virtual pool state access
- Participant data transfer
- Token economics preservation

AMM Integration

- Automatic pool creation
- Liquidity injection
- LP token minting
- Trading pair activation

Explorer Integration

- Chain registration
- Network monitoring
- Validator tracking

- Performance metrics

Wallet Integration

- Network auto-detection
- Token addition
- RPC endpoint configuration
- Transaction routing

6. User Experience Design

Developer Experience

1. Pre-Graduation
 - Monitor bonding curve progress
 - Prepare GitHub repository
 - Configure upgrade parameters
 - Plan validator recruitment
2. During Graduation
 - Receive notification of threshold
 - Monitor deployment progress
 - Verify chain initialization
 - Confirm liquidity migration
3. Post-Graduation
 - Access management dashboard
 - Monitor chain health
 - Execute planned upgrade
 - Engage with validators

Participant Experience

1. Notification Phase
 - Alert of impending graduation
 - Position summary
 - Expected timeline
 - Action requirements (none)
2. Migration Phase
 - View migration progress
 - See liquidity locking
 - Track token conversion
 - Monitor LP token distribution
3. Completion Phase
 - Access new tokens
 - Trade on AMM
 - View LP positions

- Continue normal operations

Validator Onboarding Flow

1. Discover graduated chain
2. Review chain specifications
3. Review economic opportunity (project docs?)
4. Download validator package
5. Deploy validator node
6. Register on-chain
7. Begin validation
8. Monitor performance

7. Metrics and Analytics

Deployment Metrics

- Graduation Rate: Virtual chains reaching threshold
- Deployment Time: Threshold to operational
- Success Rate: Successful vs failed deployments
- Infrastructure Cost: Per-chain deployment cost
- Resource Utilization: Computing resources per chain

Migration Metrics

- Liquidity Preservation: Value retained through migration
- Participant Retention: Active users post-graduation
- Trading Continuity: Time to first post-graduation trade
- LP Distribution: Successful LP token claims
- Slippage Impact: Price impact of migration

Chain Health Metrics

- Validator Recruitment: Time to minimum validator set
- Stake Distribution: Decentralization metrics
- Transaction Activity: Post-graduation usage
- Upgrade Success: Custom code activation rate
- 30-Day Survival: Chains still active after month

8. Open Questions / Next Steps

Pending Decisions

- Graduation threshold amount (\$50,000 suggested)
- Minimum validator requirements

- Initial validator incentive structure (default Canopy economics?)
- Emergency pause conditions
- Rollback criteria and procedures

Considered but Deferred

- Manual Graduation Override: Admin-triggered graduation
- Security: Custom root deployment
- Forced Graduation: Community vote for graduation
- Dynamic Thresholds: Market-adjusted graduation levels
- Rescue Mechanisms: Recovery for failed upgrades

Next Steps

1. Finalize graduation threshold parameters
2. Design validator incentive program
3. Build deployment automation infrastructure
4. Create migration testing framework
5. Develop validator onboarding materials
6. Establish monitoring and alerting systems
7. Plan disaster recovery procedures
8. Create graduation simulation environment