- Pre-audit Assessment for Readiness: Graph Horizon and Subgraph Services
  - Scope Details
    - Subgraph Services
    - Horizon
    - Main Contracts
  - Test Information
    - Subgraph Services
    - Horizon
    - Main Contracts
  - Validation of Prior Critical & High Issues
    - OpenZeppelin - The Graph Horizon Audit (September 5, 2024)
    - Trust Security - The Graph Horizon Upgrade Audit (December 17, 2024)
  - Automated Scanning Results
    - Boolean Literals Used in Complex Expressions or as Conditionals
    - Floating Pragma
    - Incomplete Docstrings
    - Lack of Storage Gap in Upgradeable Contracts
    - Missing Docstrings
    - Require Statement Does Not Check for Any Conditions
    - Require Statement With Multiple Conditions
    - Missing Error Message in Revert Statement
    - Unsafe ABI Encoding
    - Use `calldata` Instead of `memory`
    - Duplicate Imports
    - File and Contract Names Mismatch
    - Todo Comments in the Code
    - Inconsistent Use of Named Returns in a Function
    - Prefix Increment Operator `++i` Can Save Gas in Loops
    - Indecisive Licenses
    - Lack of Indexed Event Parameters
    - Lack of Security Contact
    - Modifiers Could Be Used
    - Non-explicit Imports Are Used
    - Redundant Getter Functions
    - Redundant Return Statement
    - Use Custom Errors
    - Unnecessary Casts
    - Unsafe Casts
    - Unused Functions With Internal or Private Visibility
    - Unused Imports
    - Unused Modifiers
    - Unused Named Return Variables
    - Variables Could Be `constant`

Pre-audit Assessment for Readiness: Graph Horizon and Subgraph Services

Link to the repo: Graph Protocol

Report commit: 7bac074

Scope Details

Audit of pull request #944 which concerns the below contracts.

Subgraph Services

9 contracts, 3 libraries, 2 interfaces:

```
subgraph-service
  contracts
      DisputeManager.sol
      DisputeManagerStorage.sol
      SubgraphService.sol
      SubgraphServiceStorage.sol
      interfaces
          IDisputeManager.sol
          ISubgraphService.sol
      libraries
          Allocation.sol
          Attestation.sol
          LegacyAllocation.sol
      utilities
           AllocationManager.sol
           AllocationManagerStorage.sol
           AttestationManager.sol
           AttestationManagerStorage.sol
           Directory.sol
```

Horizon

20 contracts, 7 libraries, 15 interfaces:

```
horizon
  contracts
      data-service
          DataService.sol
          DataServiceStorage.sol
          extensions
              DataServiceFees.sol
              DataServiceFeesStorage.sol
```

```
        DataServicePausable.sol
        DataServicePausableUpgradeable.sol
        DataServiceRescuable.sol
    interfaces
        IDataService.sol
        IDataServiceFees.sol
        IDataServicePausable.sol
        IDataServiceRescuable.sol
    libraries
        ProvisionTracker.sol
    utilities
         ProvisionManager.sol
         ProvisionManagerStorage.sol
interfaces
    IAuthorizable.sol
    IGraphPayments.sol
    IGraphProxyAdmin.sol
    IGraphTallyCollector.sol
    IHorizonStaking.sol
    IPaymentsCollector.sol
    IPaymentsEscrow.sol
    internal
         IHorizonStakingBase.sol
         IHorizonStakingExtension.sol
         IHorizonStakingMain.sol
         IHorizonStakingTypes.sol
libraries
    Denominations.sol
    LibFixedMath.sol
    LinkedList.sol
    MathUtils.sol
    PPMMath.sol
    UintRange.sol
payments
    GraphPayments.sol
    PaymentsEscrow.sol
    collectors
         GraphTallyCollector.sol
staking
    HorizonStaking.sol
    HorizonStakingBase.sol
    HorizonStakingExtension.sol
    HorizonStakingStorage.sol
    libraries
        ExponentialRebates.sol
    utilities
```

```
            Managed.sol
        utilities
            Authorizable.sol
            GraphDirectory.sol
```

Main Contracts

10 contract, 6 interfaces (selected based on PR changes, but auditors might need to reference other contracts within `contracts/contracts`):

```
contracts
   contracts
       curation
           Curation.sol
           CurationStorage.sol
           ICuration.sol
       governance
           Controller.sol
           Governed.sol
           IController.sol
           IManaged.sol
           Managed.sol
           Pausable.sol
       l2
           curation
               IL2Curation.sol
               L2Curation.sol
       rewards
           IRewardsIssuer.sol
           IRewardsManager.sol
           RewardsManager.sol
           RewardsManagerStorage.sol
       staking
           Staking.sol
```

Test Information

Complete testing suite.

Subgraph Services

Great coverage. No failing tests. 144 total Foundry tests:

```
Ran 35 test suites in 120.93s (882.84s CPU time): 144 tests passed, 0 failed, 0 skipped (144
| File                                        | % Lines           | % Statements       | % Br
|---------------------------------------------|-------------------|--------------------|-----
| contracts/DisputeManager.sol                | 98.56% (137/139)  | 98.20% (164/167)   | 92.5
```

| contracts/SubgraphService.sol | 94.90% (93/98) | 95.65% (110/115) | 94.5
| contracts/libraries/Allocation.sol | 100.00% (30/30) | 100.00% (42/42) | 66.6
| contracts/libraries/Attestation.sol | 100.00% (15/15) | 100.00% (19/19) | 50.0
| contracts/libraries/LegacyAllocation.sol | 66.67% (6/9) | 58.33% (7/12) | 62.5
| contracts/utilities/AllocationManager.sol | 97.22% (70/72) | 95.51% (85/89) | 95.0
| contracts/utilities/AttestationManager.sol | 100.00% (7/7) | 100.00% (9/9) | 100.
| contracts/utilities/Directory.sol | 87.50% (7/8) | 90.00% (9/10) | 50.0
| test/SubgraphBaseTest.t.sol | 98.51% (66/67) | 98.67% (74/75) | 100.
| test/disputeManager/DisputeManager.t.sol | 99.20% (249/251) | 98.68% (299/303) | 78.9
| test/mocks/MockCuration.sol | 100.00% (1/1) | 100.00% (1/1) | 100.
| test/mocks/MockEpochManager.sol | 50.00% (6/12) | 36.00% (9/25) | 100.
| test/mocks/MockGRTToken.sol | 66.67% (2/3) | 66.67% (2/3) | 100.
| test/mocks/MockRewardsManager.sol | 100.00% (12/12) | 100.00% (15/15) | 100.
| test/shared/HorizonStakingShared.t.sol | 100.00% (38/38) | 100.00% (41/41) | 100.
| test/shared/SubgraphServiceShared.t.sol | 97.22% (70/72) | 97.70% (85/87) | 100.
| test/subgraphService/SubgraphService.t.sol | 100.00% (187/187) | 100.00% (235/235) | 100.
| test/utils/Utils.sol | 100.00% (2/2) | 100.00% (2/2) | 100.
| Total | 97.56% (998/1023) | 96.64% (1208/1250) | 87.5

Horizon

Great coverage. No failing tests. 315 total Foundry tests:

Ran 46 test suites in 145.00s (1808.56s CPU time): 315 tests passed, 0 failed, 0 skipped (31

File	% Lines
contracts/data-service/DataService.sol	100.00% (5/5)
contracts/data-service/extensions/DataServiceFees.sol	100.00% (26/26)
contracts/data-service/extensions/DataServicePausable.sol	100.00% (6/6)
contracts/data-service/extensions/DataServicePausableUpgradeable.sol	16.67% (1/6)
contracts/data-service/extensions/DataServiceRescuable.sol	0.00% (0/7)
contracts/data-service/libraries/ProvisionTracker.sol	70.00% (7/10)
contracts/data-service/utilities/ProvisionManager.sol	88.37% (38/43)
contracts/libraries/Denominations.sol	0.00% (0/1)
contracts/libraries/LibFixedMath.sol	89.66% (78/87)
contracts/libraries/LinkedList.sol	100.00% (21/21)
contracts/libraries/MathUtils.sol	100.00% (3/3)
contracts/libraries/PPMMath.sol	100.00% (5/5)
contracts/libraries/UintRange.sol	100.00% (2/2)
contracts/mocks/ControllerMock.sol	0.00% (0/15)
contracts/mocks/CurationMock.sol	100.00% (4/4)
contracts/mocks/EpochManagerMock.sol	50.00% (6/12)
contracts/mocks/MockGRTToken.sol	66.67% (2/3)
contracts/mocks/RewardsManagerMock.sol	100.00% (4/4)
contracts/payments/GraphPayments.sol	100.00% (21/21)
contracts/payments/PaymentsEscrow.sol	100.00% (37/37)

```
| contracts/payments/collectors/GraphTallyCollector.sol                    | 96.97% (32/33)
| contracts/staking/HorizonStaking.sol                                     | 98.99% (293/296)
| contracts/staking/HorizonStakingBase.sol                                 | 85.92% (61/71)
| contracts/staking/HorizonStakingExtension.sol                            | 94.12% (112/119)
| contracts/staking/libraries/ExponentialRebates.sol                       | 100.00% (9/9)
| contracts/staking/utilities/Managed.sol                                  | 66.67% (2/3)
| contracts/utilities/Authorizable.sol                                     | 100.00% (23/23)
| contracts/utilities/GraphDirectory.sol                                   | 100.00% (26/26)
| test/GraphBase.t.sol                                                     | 100.00% (83/83)
| test/data-service/implementations/DataServiceBase.sol                    | 100.00% (8/8)
| test/data-service/implementations/DataServiceBaseUpgradeable.sol         | 100.00% (3/3)
| test/data-service/implementations/DataServiceImpFees.sol                 | 33.33% (2/6)
| test/data-service/implementations/DataServiceImpPausable.sol             | 100.00% (4/4)
| test/data-service/implementations/DataServiceImpPausableUpgradeable.sol  | 100.00% (4/4)
| test/data-service/implementations/DataServiceOverride.sol                | 100.00% (4/4)
| test/data-service/libraries/ProvisionTracker.t.sol                       | 100.00% (2/2)
| test/escrow/GraphEscrow.t.sol                                            | 96.77% (60/62)
| test/libraries/ListImplementation.sol                                    | 100.00% (11/11)
| test/payments/GraphPayments.t.sol                                        | 100.00% (1/1)
| test/payments/graph-tally-collector/GraphTallyCollector.t.sol            | 100.00% (51/51)
| test/shared/horizon-staking/HorizonStakingShared.t.sol                   | 99.08% (863/871)
| test/shared/payments-escrow/PaymentsEscrowShared.t.sol                   | 100.00% (18/18)
| test/staking/HorizonStaking.t.sol                                        | 93.94% (31/33)
| test/utilities/Authorizable.t.sol                                        | 100.00% (32/32)
| test/utilities/GraphDirectoryImplementation.sol                          | 81.82% (9/11)
| test/utils/Bounder.t.sol                                                 | 100.00% (8/8)
| test/utils/Utils.sol                                                     | 100.00% (2/2)
| Total                                                                    | 95.64% (2020/211
```

Main Contracts

Great coverage. No failing tests. 724 total Hardhat tests:

```
724 passing (15m)
-----------------------------------------|----------|----------|----------|----------|-----
File                                     |  % Stmts | % Branch |  % Funcs |  % Lines |Uncov
-----------------------------------------|----------|----------|----------|----------|-----
 base/                                   |      100 |      100 |      100 |      100 |
  IMulticall.sol                         |      100 |      100 |      100 |      100 |
  Multicall.sol                          |      100 |      100 |      100 |      100 |
 curation/                               |      100 |    90.91 |      100 |      100 |
  Curation.sol                           |      100 |    90.91 |      100 |      100 |
  CurationStorage.sol                    |      100 |      100 |      100 |      100 |
  GraphCurationToken.sol                 |      100 |      100 |      100 |      100 |
  ICuration.sol                          |      100 |      100 |      100 |      100 |
  IGraphCurationToken.sol                |      100 |      100 |      100 |      100 |
```

discovery/	99.59	92.39	100	99.59	
GNS.sol	99.36	94.23	100	99.37	
GNSStorage.sol	100	100	100	100	
IGNS.sol	100	100	100	100	
IServiceRegistry.sol	100	100	100	100	
ISubgraphNFT.sol	100	100	100	100	
ISubgraphNFTDescriptor.sol	100	100	100	100	
L1GNS.sol	100	93.75	100	100	
L1GNSStorage.sol	100	100	100	100	
ServiceRegistry.sol	100	100	100	100	
ServiceRegistryStorage.sol	100	100	100	100	
SubgraphNFT.sol	100	78.57	100	100	
SubgraphNFTDescriptor.sol	100	100	100	100	
discovery/erc1056/	0	0	0	0	
EthereumDIDRegistry.sol	0	0	0	0	... 2
IEthereumDIDRegistry.sol	100	100	100	100	
disputes/	99.17	86	100	99.19	
DisputeManager.sol	99.17	86	100	99.19	
DisputeManagerStorage.sol	100	100	100	100	
IDisputeManager.sol	100	100	100	100	
epochs/	77.78	50	81.82	77.78	
EpochManager.sol	77.78	50	81.82	77.78	...
EpochManagerStorage.sol	100	100	100	100	
IEpochManager.sol	100	100	100	100	
gateway/	100	76.83	100	100	
BridgeEscrow.sol	100	100	100	100	
GraphTokenGateway.sol	100	87.5	100	100	
ICallhookReceiver.sol	100	100	100	100	
L1GraphTokenGateway.sol	100	75.68	100	100	
governance/	97.56	89.47	100	97.73	
Controller.sol	100	100	100	100	
Governed.sol	100	83.33	100	100	
IController.sol	100	100	100	100	
IManaged.sol	100	100	100	100	
Managed.sol	100	93.75	100	100	
Pausable.sol	86.67	75	100	86.67	
l2/curation/	100	87.5	100	100	
IL2Curation.sol	100	100	100	100	
L2Curation.sol	100	87.5	100	100	
l2/discovery/	100	96.15	100	100	
IL2GNS.sol	100	100	100	100	
L2GNS.sol	100	96.15	100	100	
L2GNSStorage.sol	100	100	100	100	
l2/gateway/	100	76.47	100	100	
L2GraphTokenGateway.sol	100	76.47	100	100	
l2/staking/	100	100	100	100	

IL2Staking.sol	100	100	100	100	
IL2StakingBase.sol	100	100	100	100	
IL2StakingTypes.sol	100	100	100	100	
L2Staking.sol	100	100	100	100	
l2/token/	100	80	100	100	
GraphTokenUpgradeable.sol	100	91.67	100	100	
L2GraphToken.sol	100	62.5	100	100	
libraries/	95.24	50	100	97.67	
Base58Encoder.sol	96.3	50	100	100	
HexStrings.sol	93.33	50	100	93.75	
payments/	92.59	85	87.5	92.59	
AllocationExchange.sol	92.59	85	87.5	92.59	
rewards/	96.69	92	93.1	96.83	
IRewardsIssuer.sol	100	100	100	100	
IRewardsManager.sol	100	100	100	100	
RewardsManager.sol	94.94	86.36	90.48	95	126,1
RewardsManagerStorage.sol	100	100	100	100	
SubgraphAvailabilityManager.sol	100	96.43	100	100	
staking/	98.49	96.81	96.97	98.51	
IL1GraphTokenLockTransferTool.sol	100	100	100	100	
IL1Staking.sol	100	100	100	100	
IL1StakingBase.sol	100	100	100	100	
IStaking.sol	100	100	100	100	
IStakingBase.sol	100	100	100	100	
IStakingData.sol	100	100	100	100	
IStakingExtension.sol	100	100	100	100	
L1Staking.sol	100	100	100	100	
L1StakingStorage.sol	100	100	100	100	
Staking.sol	97.71	95.92	96.08	97.72	... 9
StakingExtension.sol	99.12	95.45	97.44	99.12	
StakingStorage.sol	100	100	100	100	
staking/libs/	60.1	47.83	77.14	60.1	
Exponential.sol	100	100	100	100	
LibFixedMath.sol	50	38.16	57.89	50	... 3
MathUtils.sol	100	100	100	100	
Stakes.sol	100	90	100	100	
tests/	100	60	100	100	
CallhookReceiverMock.sol	100	100	100	100	
GovernedMock.sol	100	100	100	100	
L1GraphTokenLockTransferToolBadMock.sol	100	50	100	100	
L1GraphTokenLockTransferToolMock.sol	100	50	100	100	
LegacyGNSMock.sol	100	100	100	100	
tests/ens/	100	100	100	100	
IENS.sol	100	100	100	100	
IPublicResolver.sol	100	100	100	100	
ITestRegistrar.sol	100	100	100	100	

```
 token/                                  |      100 |      100 |      100 |      100 |
  GraphToken.sol                         |      100 |      100 |      100 |      100 |
  IGraphToken.sol                        |      100 |      100 |      100 |      100 |
 upgrades/                               |    98.28 |    65.38 |    96.97 |    97.22 |
  GraphProxy.sol                         |      100 |    71.43 |      100 |      100 |
  GraphProxyAdmin.sol                    |      100 |       50 |      100 |      100 |
  GraphProxyStorage.sol                  |    92.31 |        0 |    85.71 |       90 |
  GraphUpgradeable.sol                   |      100 |      100 |      100 |      100 |
  IGraphProxy.sol                        |      100 |      100 |      100 |      100 |
 utils/                                  |      100 |       90 |      100 |      100 |
  TokenUtils.sol                         |      100 |       90 |      100 |      100 |
-----------------------------------------|----------|----------|----------|----------|-----
All files                                |    92.38 |    82.88 |    93.06 |    92.48 |
-----------------------------------------|----------|----------|----------|----------|-----
```

Validation of Prior Critical & High Issues

These will validated later.

OpenZeppelin - The Graph Horizon Audit (September 5, 2024)

Link

Trust Security - The Graph Horizon Upgrade Audit (December 17, 2024)

Link

Automated Scanning Results

These are curated issues from the Inspector scan. See closed issues in the audit repository for false positives.

Boolean Literals Used in Complex Expressions or as Conditionals

Boolean literals in code have only a few legitimate uses. Other uses (in complex expressions, as conditionals) indicate either an error or, most likely, the presence of faulty code.

Throughout the codebase, there are multiple potentially misused Boolean literals.

- Boolean literal `True` within the contract `HorizonStakingExtension` in `HorizonStakingExtension.sol`.
- Boolean literal `True` within the contract `Staking` in `Staking.sol`.
- Boolean literal `True` within the contract `StakingExtension` in `StakingExtension.sol`.

To avoid misuse of a Boolean literal, ensure it aligns with intended behavior and is well documented.

Floating Pragma

> Auditor's comment: It seems to be for backwards compatibility. However, I'm still concerned with `^0.7.6 || 0.8.27`. Arithmetic operations have different behaviours in these versions. So any contract with this pragma directive must be confirmed to not have any arithmetic operation.

Pragma directives should be fixed to clearly identify the Solidity version with which the contracts will be compiled.

Throughout the codebase, there are multiple floating pragma directives:

- `Controller.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `Curation.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Curation.sol` has the `abicoder v2` floating pragma directive.
- `CurationStorage.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Exponential.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Governed.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `GraphUpgradeable.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IController.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `ICuration.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IEpochManager.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IGNS.sol` has the `solidity ^0.7.6` floating pragma directive.
- `IGraphCurationToken.sol` has the `solidity ^0.7.6` floating pragma directive.
- `IGraphProxy.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IGraphToken.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.

- `IL2Curation.sol` has the `solidity ^0.7.6` floating pragma directive.
- `IManaged.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IMulticall.sol` has the `solidity ^0.7.6` floating pragma directive.
- `IMulticall.sol` has the `abicoder v2` floating pragma directive.
- `IRewardsIssuer.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IRewardsManager.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `IStaking.sol` has the `solidity >=0.6.12 <0.8.0` floating pragma directive.
- `IStaking.sol` has the `abicoder v2` floating pragma directive.
- `IStakingBase.sol` has the `solidity >=0.6.12 <0.8.0` floating pragma directive.
- `IStakingBase.sol` has the `abicoder v2` floating pragma directive.
- `IStakingData.sol` has the `solidity >=0.6.12 <0.8.0` floating pragma directive.
- `IStakingExtension.sol` has the `solidity >=0.6.12 <0.8.0` floating pragma directive.
- `IStakingExtension.sol` has the `abicoder v2` floating pragma directive.
- `ITokenGateway.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `L2Curation.sol` has the `solidity ^0.7.6` floating pragma directive.
- `L2Curation.sol` has the `abicoder v2` floating pragma directive.
- `LibFixedMath.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Managed.sol` has the `solidity ^0.7.6` floating pragma directive.
- `MathUtils.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Multicall.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Multicall.sol` has the `abicoder v2` floating pragma directive.
- `Pausable.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `RewardsManager.sol` has the `solidity ^0.7.6` floating pragma directive.
- `RewardsManager.sol` has the `abicoder v2` floating pragma directive.

- `RewardsManagerStorage.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.
- `Stakes.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Stakes.sol` has the `abicoder v2` floating pragma directive.
- `Staking.sol` has the `solidity ^0.7.6` floating pragma directive.
- `Staking.sol` has the `abicoder v2` floating pragma directive.
- `StakingExtension.sol` has the `solidity ^0.7.6` floating pragma directive.
- `StakingExtension.sol` has the `abicoder v2` floating pragma directive.
- `StakingStorage.sol` has the `solidity ^0.7.6` floating pragma directive.
- `TokenUtils.sol` has the `solidity ^0.7.6 || 0.8.27` floating pragma directive.

Consider using fixed pragma directives.

Incomplete Docstrings

Throughout the codebase, multiple instances of incomplete docstrings were identified:

- In `Authorizable.sol`, the `authorizeSigner` function For example:
  - The `signer`, `proofDeadline`, `proof` parameters are not documented.
- In `Authorizable.sol`, the `thawSigner` function For example:
  - The `signer` parameter is not documented.
- In `Authorizable.sol`, the `cancelThawSigner` function For example:
  - The `signer` parameter is not documented.
- In `Authorizable.sol`, the `revokeAuthorizedSigner` function For example:
  - The `signer` parameter is not documented.
- In `Authorizable.sol`, the `getThawEnd` function For example:
  - The `signer` parameter is not documented.
  - Not all return values are documented.
- In `Authorizable.sol`, the `isAuthorized` function For example:
  - The `authorizer`, `signer` parameters are not documented.
  - Not all return values are documented.
- In `Controller.sol`, the `SetContractProxy` event For example:

  - The `id`, `contractAddress` parameters are not documented.
- In `Controller.sol`, the `getGovernor` function For example:
  - Not all return values are documented.
- In `Curation.sol`, the `Signalled` event For example:
  - The `curator`, `subgraphDeploymentID`, `tokens`, `signal`, `curationTax` parameters are not documented.
- In `Curation.sol`, the `Burned` event For example:
  - The `curator`, `subgraphDeploymentID`, `tokens`, `signal` parameters are not documented.
- In `Curation.sol`, the `Collected` event For example:
  - The `subgraphDeploymentID`, `tokens` parameters are not documented.
- In `DataService.sol`, the `getThawingPeriodRange` function For example:
  - Not all return values are documented.
- In `DataService.sol`, the `getVerifierCutRange` function For example:
  - Not all return values are documented.
- In `DataService.sol`, the `getProvisionTokensRange` function For example:
  - Not all return values are documented.
- In `DataService.sol`, the `getDelegationRatio` function For example:
  - Not all return values are documented.
- In `DataServiceFees.sol`, the `releaseStake` function For example:
  - The `numClaimsToRelease` parameter is not documented.
- In `DataServiceRescuable.sol`, the `rescueGRT` function For example:
  - The `to`, `tokens` parameters are not documented.
- In `DataServiceRescuable.sol`, the `rescueETH` function For example:
  - The `to`, `tokens` parameters are not documented.
- In `DisputeManager.sol`, the `createIndexingDispute` function For example:
  - Not all return values are documented.
- In `DisputeManager.sol`, the `createQueryDispute` function For example:
  - Not all return values are documented.

- In `DisputeManager.sol`, the `createQueryDisputeConflict` function For example:
  - Not all return values are documented.
- In `DisputeManager.sol`, the `getStakeSnapshot` function For example:
  - Not all return values are documented.
- In `DisputeManager.sol`, the `areConflictingAttestations` function For example:
  - Not all return values are documented.
- In `DisputeManager.sol`, the `isDisputeCreated` function For example:
  - Not all return values are documented.
- In `Governed.sol`, the `NewPendingOwnership` event For example:
  - The `from`, `to` parameters are not documented.
- In `Governed.sol`, the `NewOwnership` event For example:
  - The `from`, `to` parameters are not documented.
- In `GraphPayments.sol`, the `collect` function For example:
  - The `paymentType`, `receiver`, `tokens`, `dataService`, `dataServiceCut` parameters are not documented.
- In `GraphTallyCollector.sol`, the `collect` function For example:
  - The `paymentType`, `data` parameters are not documented.
  - Not all return values are documented.
- In `GraphTallyCollector.sol`, the `recoverRAVSigner` function For example:
  - The `signedRAV` parameter is not documented.
  - Not all return values are documented.
- In `GraphTallyCollector.sol`, the `encodeRAV` function For example:
  - The `rav` parameter is not documented.
  - Not all return values are documented.
- In `HorizonStaking.sol`, the `stake` function For example:
  - The `tokens` parameter is not documented.
- In `HorizonStaking.sol`, the `stakeTo` function For example:
  - The `serviceProvider`, `tokens` parameters are not documented.
- In `HorizonStaking.sol`, the `stakeToProvision` function For example:

  - The `serviceProvider`, `verifier`, `tokens` parameters are not documented.

- In `HorizonStaking.sol`, the `unstake` function For example:
  - The `tokens` parameter is not documented.

- In `HorizonStaking.sol`, the `provision` function For example:
  - The `serviceProvider`, `verifier`, `tokens`, `maxVerifierCut`, `thawingPeriod` parameters are not documented.

- In `HorizonStaking.sol`, the `addToProvision` function For example:
  - The `serviceProvider`, `verifier`, `tokens` parameters are not documented.

- In `HorizonStaking.sol`, the `thaw` function For example:
  - The `serviceProvider`, `verifier`, `tokens` parameters are not documented.
  - Not all return values are documented.

- In `HorizonStaking.sol`, the `deprovision` function For example:
  - The `serviceProvider`, `verifier`, `nThawRequests` parameters are not documented.

- In `HorizonStaking.sol`, the `reprovision` function For example:
  - The `serviceProvider`, `oldVerifier`, `newVerifier`, `nThawRequests` parameters are not documented.

- In `HorizonStaking.sol`, the `setProvisionParameters` function For example:
  - The `serviceProvider`, `verifier`, `newMaxVerifierCut`, `newThawingPeriod` parameters are not documented.

- In `HorizonStaking.sol`, the `acceptProvisionParameters` function For example:
  - The `serviceProvider` parameter is not documented.

- In `HorizonStaking.sol`, the `delegate` function For example:
  - The `serviceProvider`, `verifier`, `tokens`, `minSharesOut` parameters are not documented.

- In `HorizonStaking.sol`, the `addToDelegationPool` function For example:
  - The `serviceProvider`, `verifier`, `tokens` parameters are not documented.

- In `HorizonStaking.sol`, the `undelegate` function For example:

  - The `serviceProvider`, `verifier`, `shares` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStaking.sol`, the `withdrawDelegated` function For example:
  - The `serviceProvider`, `verifier`, `nThawRequests` parameters are not documented.
- In `HorizonStaking.sol`, the `redelegate` function For example:
  - The `oldServiceProvider`, `oldVerifier`, `newServiceProvider`, `newVerifier`, `minSharesForNewProvider`, `nThawRequests` parameters are not documented.
- In `HorizonStaking.sol`, the `setDelegationFeeCut` function For example:
  - The `serviceProvider`, `verifier`, `paymentType`, `feeCut` parameters are not documented.
- In `HorizonStaking.sol`, the `delegate` function For example:
  - The `serviceProvider`, `tokens` parameters are not documented.
- In `HorizonStaking.sol`, the `undelegate` function For example:
  - The `serviceProvider`, `shares` parameters are not documented.
- In `HorizonStaking.sol`, the `withdrawDelegated` function For example:
  - The `serviceProvider`, `None` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStaking.sol`, the `slash` function For example:
  - The `serviceProvider`, `tokens`, `tokensVerifier`, `verifierDestination` parameters are not documented.
- In `HorizonStaking.sol`, the `provisionLocked` function For example:
  - The `serviceProvider`, `verifier`, `tokens`, `maxVerifierCut`, `thawingPeriod` parameters are not documented.
- In `HorizonStaking.sol`, the `setOperatorLocked` function For example:
  - The `verifier`, `operator`, `allowed` parameters are not documented.
- In `HorizonStaking.sol`, the `setAllowedLockedVerifier` function For example:
  - The `verifier`, `allowed` parameters are not documented.
- In `HorizonStaking.sol`, the `setMaxThawingPeriod` function For example:
  - The `maxThawingPeriod` parameter is not documented.

- In `HorizonStaking.sol`, the `setOperator` function For example:
  - The `verifier`, `operator`, `allowed` parameters are not documented.
- In `HorizonStaking.sol`, the `isAuthorized` function For example:
  - The `serviceProvider`, `verifier`, `operator` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getServiceProvider` function For example:
  - The `serviceProvider` parameter is not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getStake` function For example:
  - The `serviceProvider` parameter is not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getIdleStake` function For example:
  - The `serviceProvider` parameter is not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getDelegationPool` function For example:
  - The `serviceProvider`, `verifier` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getDelegation` function For example:
  - The `serviceProvider`, `verifier`, `delegator` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getDelegationFeeCut` function For example:
  - The `serviceProvider`, `verifier`, `paymentType` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getProvision` function For example:
  - The `serviceProvider`, `verifier` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getTokensAvailable` function For example:
  - The `serviceProvider`, `verifier`, `delegationRatio` parameters are not documented.

  - Not all return values are documented.

- In `HorizonStakingBase.sol`, the `getProviderTokensAvailable` function For example:
  - The `serviceProvider`, `verifier` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getDelegatedTokensAvailable` function For example:
  - The `serviceProvider`, `verifier` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getThawRequest` function For example:
  - The `requestType`, `thawRequestId` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getThawRequestList` function For example:
  - The `requestType`, `serviceProvider`, `verifier`, `owner` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getThawedTokens` function For example:
  - The `requestType`, `serviceProvider`, `verifier`, `owner` parameters are not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `getMaxThawingPeriod` function For example:
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `isAllowedLockedVerifier` function For example:
  - The `verifier` parameter is not documented.
  - Not all return values are documented.
- In `HorizonStakingBase.sol`, the `isDelegationSlashingEnabled` function For example:
  - Not all return values are documented.
- In `HorizonStakingExtension.sol`, the `getAllocationData` function For example:
  - Not all return values are documented.
- In `IAuthorizable.sol`, the `getThawEnd` function For example:

  - The `signer` parameter is not documented.
  - Not all return values are documented.

- In `IAuthorizable.sol`, the `isAuthorized` function For example:
  - The `authorizer`, `signer` parameters are not documented.
  - Not all return values are documented.

- In `IDataServiceRescuable.sol`, the `RescuerSet` event For example:
  - The `account`, `allowed` parameters are not documented.

- In `IDisputeManager.sol`, the `QueryDisputeCreated` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens`, `subgraphDeploymentId`, `attestation`, `stakeSnapshot` parameters are not documented.

- In `IDisputeManager.sol`, the `IndexingDisputeCreated` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens`, `allocationId`, `poi`, `stakeSnapshot` parameters are not documented.

- In `IDisputeManager.sol`, the `DisputeAccepted` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens` parameters are not documented.

- In `IDisputeManager.sol`, the `DisputeRejected` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens` parameters are not documented.

- In `IDisputeManager.sol`, the `DisputeDrawn` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens` parameters are not documented.

- In `IDisputeManager.sol`, the `DisputeLinked` event For example:
  - The `disputeId1`, `disputeId2` parameters are not documented.

- In `IDisputeManager.sol`, the `DisputeCancelled` event For example:
  - The `disputeId`, `indexer`, `fisherman`, `tokens` parameters are not documented.

- In `IGNS.sol`, the `tokensToNSignal` function For example:
  - Not all return values are documented.

- In `IGNS.sol`, the `nSignalToTokens` function For example:
  - Not all return values are documented.

- In `IGraphTallyCollector.sol`, the `collect` function For example:
  - Not all return values are documented.

- In `IHorizonStakingBase.sol`, the `getServiceProvider` function For example:
  - Not all return values are documented.
- In `IHorizonStakingBase.sol`, the `getMaxThawingPeriod` function For example:
  - Not all return values are documented.
- In `IHorizonStakingBase.sol`, the `isDelegationSlashingEnabled` function For example:
  - Not all return values are documented.
- In `IHorizonStakingExtension.sol`, the `AllocationClosed` event For example:
  - The `indexer`, `subgraphDeploymentID`, `epoch`, `tokens`, `allocationID`, `sender`, `poi`, `isPublic` parameters are not documented.
- In `IHorizonStakingExtension.sol`, the `RebateCollected` event For example:
  - The `assetHolder`, `indexer`, `subgraphDeploymentID`, `allocationID`, `epoch`, `tokens`, `protocolTax`, `curationFees`, `queryFees`, `queryRebates`, `delegationRewards` parameters are not documented.
- In `IHorizonStakingExtension.sol`, the `StakeSlashed` event For example:
  - The `indexer`, `tokens`, `reward`, `beneficiary` parameters are not documented.
- In `IHorizonStakingMain.sol`, the `withdrawDelegated` function For example:
  - The `None` parameter is not documented.
  - Not all return values are documented.
- In `IPaymentsCollector.sol`, the `collect` function For example:
  - Not all return values are documented.
- In `IPaymentsEscrow.sol`, the `getBalance` function For example:
  - Not all return values are documented.
- In `IStakingBase.sol`, the `StakeDeposited` event For example:
  - The `indexer`, `tokens` parameters are not documented.
- In `IStakingBase.sol`, the `StakeLocked` event For example:
  - The `indexer`, `tokens`, `until` parameters are not documented.

- In `IStakingBase.sol`, the `StakeWithdrawn` event For example:
  - The `indexer`, `tokens` parameters are not documented.
- In `IStakingBase.sol`, the `AllocationCreated` event For example:
  - The `indexer`, `subgraphDeploymentID`, `epoch`, `tokens`, `allocationID`, `metadata` parameters are not documented.
- In `IStakingBase.sol`, the `AllocationClosed` event For example:
  - The `indexer`, `subgraphDeploymentID`, `epoch`, `tokens`, `allocationID`, `sender`, `poi`, `isPublic` parameters are not documented.
- In `IStakingBase.sol`, the `RebateCollected` event For example:
  - The `assetHolder`, `indexer`, `subgraphDeploymentID`, `allocationID`, `epoch`, `tokens`, `protocolTax`, `curationFees`, `queryFees`, `queryRebates`, `delegationRewards` parameters are not documented.
- In `IStakingBase.sol`, the `DelegationParametersUpdated` event For example:
  - The `indexer`, `indexingRewardCut`, `queryFeeCut`, `__DEPRECATED_cooldownBlocks` parameters are not documented.
- In `IStakingBase.sol`, the `SetOperator` event For example:
  - The `indexer`, `operator`, `allowed` parameters are not documented.
- In `IStakingBase.sol`, the `SetRewardsDestination` event For example:
  - The `indexer`, `destination` parameters are not documented.
- In `IStakingBase.sol`, the `ExtensionImplementationSet` event For example:
  - The `extensionImpl` parameter is not documented.
- In `IStakingBase.sol`, the `setDelegationParameters` function For example:
  - The `None` parameter is not documented.
- In `IStakingBase.sol`, the `getAllocationData` function For example:
  - The `_allocationID` parameter is not documented.
  - Not all return values are documented.
- In `IStakingBase.sol`, the `isActiveAllocation` function For example:
  - The `_allocationID` parameter is not documented.
  - Not all return values are documented.
- In `IStakingExtension.sol`, the `StakeDelegated` event For example:

  - The `indexer`, `delegator`, `tokens`, `shares` parameters are not documented.

- In `IStakingExtension.sol`, the `StakeDelegatedLocked` event For example:
  - The `indexer`, `delegator`, `tokens`, `shares`, `until` parameters are not documented.
- In `IStakingExtension.sol`, the `StakeDelegatedWithdrawn` event For example:
  - The `indexer`, `delegator`, `tokens` parameters are not documented.
- In `IStakingExtension.sol`, the `StakeSlashed` event For example:
  - The `indexer`, `tokens`, `reward`, `beneficiary` parameters are not documented.
- In `IStakingExtension.sol`, the `SlasherUpdate` event For example:
  - The `caller`, `slasher`, `allowed` parameters are not documented.
- In `IStakingExtension.sol`, the `withdrawDelegated` function For example:
  - Not all return values are documented.
- In `ISubgraphService.sol`, the `getAllocation` function For example:
  - Not all return values are documented.
- In `ISubgraphService.sol`, the `getLegacyAllocation` function For example:
  - Not all return values are documented.
- In `ISubgraphService.sol`, the `encodeAllocationProof` function For example:
  - Not all return values are documented.
- In `L2Curation.sol`, the `Signalled` event For example:
  - The `curator`, `subgraphDeploymentID`, `tokens`, `signal`, `curationTax` parameters are not documented.
- In `L2Curation.sol`, the `Burned` event For example:
  - The `curator`, `subgraphDeploymentID`, `tokens`, `signal` parameters are not documented.
- In `L2Curation.sol`, the `Collected` event For example:
  - The `subgraphDeploymentID`, `tokens` parameters are not documented.

- In `L2Curation.sol`, the `SubgraphServiceSet` event For example:
  - The `newSubgraphService` parameter is not documented.
- In `L2Curation.sol`, the `setDefaultReserveRatio` function For example:
  - The `None` parameter is not documented.
- In `L2Curation.sol`, the `mint` function For example:
  - Not all return values are documented.
- In `Managed.sol`, the `ParameterUpdated` event For example:
  - The `param` parameter is not documented.
- In `Managed.sol`, the `SetController` event For example:
  - The `controller` parameter is not documented.
- In `Managed.sol`, the `ContractSynced` event For example:
  - The `nameHash`, `contractAddress` parameters are not documented.
- In `Pausable.sol`, the `PartialPauseChanged` event For example:
  - The `isPaused` parameter is not documented.
- In `Pausable.sol`, the `PauseChanged` event For example:
  - The `isPaused` parameter is not documented.
- In `Pausable.sol`, the `NewPauseGuardian` event For example:
  - The `oldPauseGuardian`, `pauseGuardian` parameters are not documented.
- In `PaymentsEscrow.sol`, the `deposit` function For example:
  - The `collector`, `receiver`, `tokens` parameters are not documented.
- In `PaymentsEscrow.sol`, the `depositTo` function For example:
  - The `payer`, `collector`, `receiver`, `tokens` parameters are not documented.
- In `PaymentsEscrow.sol`, the `thaw` function For example:
  - The `collector`, `receiver`, `tokens` parameters are not documented.
- In `PaymentsEscrow.sol`, the `cancelThaw` function For example:
  - The `collector`, `receiver` parameters are not documented.
- In `PaymentsEscrow.sol`, the `withdraw` function For example:
  - The `collector`, `receiver` parameters are not documented.
- In `PaymentsEscrow.sol`, the `collect` function For example:

  - The `paymentType`, `payer`, `receiver`, `tokens`, `dataService`, `dataServiceCut` parameters are not documented.

- In `PaymentsEscrow.sol`, the `getBalance` function For example:
  - The `payer`, `collector`, `receiver` parameters are not documented.
  - Not all return values are documented.

- In `RewardsManager.sol`, the `HorizonRewardsAssigned` event For example:
  - The `indexer`, `allocationID`, `amount` parameters are not documented.

- In `RewardsManager.sol`, the `RewardsDenied` event For example:
  - The `indexer`, `allocationID` parameters are not documented.

- In `RewardsManager.sol`, the `RewardsDenylistUpdated` event For example:
  - The `subgraphDeploymentID`, `sinceBlock` parameters are not documented.

- In `RewardsManager.sol`, the `SubgraphServiceSet` event For example:
  - The `oldSubgraphService`, `newSubgraphService` parameters are not documented.

- In `RewardsManager.sol`, the `initialize` function For example:
  - The `_controller` parameter is not documented.

- In `RewardsManager.sol`, the `getRewards` function For example:
  - The `_rewardsIssuer` parameter is not documented.

- In `Staking.sol`, the `getAllocationData` function For example:
  - The `_allocationID` parameter is not documented.
  - Not all return values are documented.

- In `Staking.sol`, the `isActiveAllocation` function For example:
  - The `_allocationID` parameter is not documented.
  - Not all return values are documented.

- In `Staking.sol`, the `setDelegationParameters` function For example:
  - The `None` parameter is not documented.

- In `StakingExtension.sol`, the `initialize` function For example:
  - The `None` parameter is not documented.

- In `StakingExtension.sol`, the `withdrawDelegated` function For example:

  - Not all return values are documented.

- In `SubgraphService.sol`, the `acceptProvisionPendingParameters` function For example:
  - The `None` parameter is not documented.
- In `SubgraphService.sol`, the `collect` function For example:
  - Not all return values are documented.
- In `SubgraphService.sol`, the `closeStaleAllocation` function For example:
  - The `allocationId` parameter is not documented.
- In `SubgraphService.sol`, the `resizeAllocation` function For example:
  - The `indexer`, `allocationId`, `tokens` parameters are not documented.
- In `SubgraphService.sol`, the `migrateLegacyAllocation` function For example:
  - The `indexer`, `allocationId`, `subgraphDeploymentID` parameters are not documented.
- In `SubgraphService.sol`, the `setPauseGuardian` function For example:
  - The `pauseGuardian`, `allowed` parameters are not documented.
- In `SubgraphService.sol`, the `setRewardsDestination` function For example:
  - The `rewardsDestination` parameter is not documented.
- In `SubgraphService.sol`, the `setMinimumProvisionTokens` function For example:
  - The `minimumProvisionTokens` parameter is not documented.
- In `SubgraphService.sol`, the `setDelegationRatio` function For example:
  - The `delegationRatio` parameter is not documented.
- In `SubgraphService.sol`, the `setStakeToFeesRatio` function For example:
  - The `stakeToFeesRatio_` parameter is not documented.
- In `SubgraphService.sol`, the `setMaxPOIStaleness` function For example:
  - The `maxPOIStaleness` parameter is not documented.
- In `SubgraphService.sol`, the `setCurationCut` function For example:

  - The `curationCut` parameter is not documented.

- In `SubgraphService.sol`, the `getAllocation` function For example:
  - The `allocationId` parameter is not documented.
  - Not all return values are documented.
- In `SubgraphService.sol`, the `getLegacyAllocation` function For example:
  - The `allocationId` parameter is not documented.
  - Not all return values are documented.
- In `SubgraphService.sol`, the `getDisputeManager` function For example:
  - Not all return values are documented.
- In `SubgraphService.sol`, the `getGraphTallyCollector` function For example:
  - Not all return values are documented.
- In `SubgraphService.sol`, the `getCuration` function For example:
  - Not all return values are documented.
- In `SubgraphService.sol`, the `encodeAllocationProof` function For example:
  - The `indexer`, `allocationId` parameters are not documented.
  - Not all return values are documented.
- In `SubgraphService.sol`, the `isOverAllocated` function For example:
  - The `indexer` parameter is not documented.
  - Not all return values are documented.

Consider thoroughly documenting all functions/events (and their parameters or return values) that are part of a contract's public API. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Lack of Storage Gap in Upgradeable Contracts

Throughout the codebase, there are several contracts that are inherited by upgradeable contracts that do not have a storage gap:

- The `CurationV1Storage` contract.
- The `CurationV2Storage` contract.
- The `CurationV3Storage` contract.

To allow the addition of new state variables without compromising storage compatibility with existing deployments, consider leaving a storage gap at the end of each contract.

Missing Docstrings

Throughout the codebase, multiple instances of missing docstrings were identified:

- In `AllocationManagerStorage.sol`, the `AllocationManagerV1Storage` abstract contract
- In `AttestationManagerStorage.sol`, the `AttestationManagerV1Storage` abstract contract
- In `CurationStorage.sol`, the `subgraphService` state variable
- In `DataServiceFeesStorage.sol`, the `feesProvisionTracker` state variable
- In `DataServiceStorage.sol`, the `DataServiceV1Storage` abstract contract
- In `DisputeManager.sol`, the `MAX_FISHERMAN_REWARD_CUT` state variable
- In `DisputeManager.sol`, the `MIN_DISPUTE_DEPOSIT` state variable
- In `DisputeManagerStorage.sol`, the `DisputeManagerV1Storage` abstract contract
- In `GraphTallyCollector.sol`, the `collect` function
- In `HorizonStaking.sol`, the `fallback` function
- In `HorizonStakingExtension.sol`, the `legacySlash` function
- In `HorizonStakingExtension.sol`, the `__DEPRECATED_getThawingPeriod` function
- In `IController.sol`, the `IController` interface
- In `IController.sol`, the `getGovernor` function
- In `IController.sol`, the `setContractProxy` function
- In `IController.sol`, the `unsetContractProxy` function
- In `IController.sol`, the `updateController` function
- In `IController.sol`, the `getContractProxy` function
- In `IController.sol`, the `setPartialPaused` function
- In `IController.sol`, the `setPaused` function
- In `IController.sol`, the `setPauseGuardian` function
- In `IController.sol`, the `paused` function
- In `IController.sol`, the `partialPaused` function
- In `IDisputeManager.sol`, the `IDisputeManager` interface

- In `IDisputeManager.sol`, the `setDisputePeriod` function
- In `IDisputeManager.sol`, the `setArbitrator` function
- In `IDisputeManager.sol`, the `setDisputeDeposit` function
- In `IDisputeManager.sol`, the `setFishermanRewardCut` function
- In `IDisputeManager.sol`, the `setMaxSlashingCut` function
- In `IDisputeManager.sol`, the `createQueryDispute` function
- In `IDisputeManager.sol`, the `createQueryDisputeConflict` function
- In `IDisputeManager.sol`, the `createIndexingDispute` function
- In `IDisputeManager.sol`, the `acceptDispute` function
- In `IDisputeManager.sol`, the `acceptDisputeConflict` function
- In `IDisputeManager.sol`, the `rejectDispute` function
- In `IDisputeManager.sol`, the `drawDispute` function
- In `IDisputeManager.sol`, the `cancelDispute` function
- In `IDisputeManager.sol`, the `setSubgraphService` function
- In `IDisputeManager.sol`, the `getVerifierCut` function
- In `IDisputeManager.sol`, the `getDisputePeriod` function
- In `IDisputeManager.sol`, the `isDisputeCreated` function
- In `IDisputeManager.sol`, the `encodeReceipt` function
- In `IDisputeManager.sol`, the `getAttestationIndexer` function
- In `IDisputeManager.sol`, the `getStakeSnapshot` function
- In `IDisputeManager.sol`, the `areConflictingAttestations` function
- In `IEpochManager.sol`, the `IEpochManager` interface
- In `IEpochManager.sol`, the `setEpochLength` function
- In `IEpochManager.sol`, the `runEpoch` function
- In `IEpochManager.sol`, the `isCurrentEpochRun` function
- In `IEpochManager.sol`, the `blockNum` function
- In `IEpochManager.sol`, the `blockHash` function
- In `IEpochManager.sol`, the `currentEpoch` function
- In `IEpochManager.sol`, the `currentEpochBlock` function
- In `IEpochManager.sol`, the `currentEpochBlockSinceStart` function
- In `IEpochManager.sol`, the `epochsSince` function

- In `IEpochManager.sol`, the `epochsSinceUpdate` function
- In `IGraphCurationToken.sol`, the `IGraphCurationToken` interface
- In `IGraphCurationToken.sol`, the `initialize` function
- In `IGraphCurationToken.sol`, the `burnFrom` function
- In `IGraphCurationToken.sol`, the `mint` function
- In `IGraphProxy.sol`, the `IGraphProxy` interface
- In `IGraphProxy.sol`, the `admin` function
- In `IGraphProxy.sol`, the `setAdmin` function
- In `IGraphProxy.sol`, the `implementation` function
- In `IGraphProxy.sol`, the `pendingImplementation` function
- In `IGraphProxy.sol`, the `upgradeTo` function
- In `IGraphProxy.sol`, the `acceptUpgrade` function
- In `IGraphProxy.sol`, the `acceptUpgradeAndCall` function
- In `IGraphToken.sol`, the `IGraphToken` interface
- In `IGraphToken.sol`, the `burn` function
- In `IGraphToken.sol`, the `burnFrom` function
- In `IGraphToken.sol`, the `mint` function
- In `IGraphToken.sol`, the `addMinter` function
- In `IGraphToken.sol`, the `removeMinter` function
- In `IGraphToken.sol`, the `renounceMinter` function
- In `IGraphToken.sol`, the `isMinter` function
- In `IGraphToken.sol`, the `permit` function
- In `IGraphToken.sol`, the `increaseAllowance` function
- In `IGraphToken.sol`, the `decreaseAllowance` function
- In `IHorizonStakingExtension.sol`, the `__DEPRECATED_getThawingPeriod` function
- In `IRewardsIssuer.sol`, the `IRewardsIssuer` interface
- In `IRewardsManager.sol`, the `IRewardsManager` interface
- In `IRewardsManager.sol`, the `setIssuancePerBlock` function
- In `IRewardsManager.sol`, the `setMinimumSubgraphSignal` function
- In `IRewardsManager.sol`, the `setSubgraphService` function

- In `IRewardsManager.sol`, the `setSubgraphAvailabilityOracle` function
- In `IRewardsManager.sol`, the `setDenied` function
- In `IRewardsManager.sol`, the `isDenied` function
- In `IRewardsManager.sol`, the `getNewRewardsPerSignal` function
- In `IRewardsManager.sol`, the `getAccRewardsPerSignal` function
- In `IRewardsManager.sol`, the `getAccRewardsForSubgraph` function
- In `IRewardsManager.sol`, the `getAccRewardsPerAllocatedToken` function
- In `IRewardsManager.sol`, the `getRewards` function
- In `IRewardsManager.sol`, the `calcRewards` function
- In `IRewardsManager.sol`, the `updateAccRewardsPerSignal` function
- In `IRewardsManager.sol`, the `takeRewards` function
- In `IRewardsManager.sol`, the `onSubgraphSignalUpdate` function
- In `IRewardsManager.sol`, the `onSubgraphAllocationUpdate` function
- In `ITokenGateway.sol`, the `ITokenGateway` interface
- In `ITokenGateway.sol`, the `outboundTransfer` function
- In `ITokenGateway.sol`, the `finalizeInboundTransfer` function
- In `Pausable.sol`, the `Pausable` abstract contract
- In `RewardsManager.sol`, the `setSubgraphService` function
- In `RewardsManagerStorage.sol`, the `RewardsManagerV1Storage` contract
- In `RewardsManagerStorage.sol`, the `accRewardsPerSignal` state variable
- In `RewardsManagerStorage.sol`, the `accRewardsPerSignalLastBlockUpdated` state variable
- In `RewardsManagerStorage.sol`, the `subgraphAvailabilityOracle` state variable
- In `RewardsManagerStorage.sol`, the `subgraphs` state variable
- In `RewardsManagerStorage.sol`, the `denylist` state variable
- In `RewardsManagerStorage.sol`, the `RewardsManagerV2Storage` contract

- In `RewardsManagerStorage.sol`, the `minimumSubgraphSignal` state variable
- In `RewardsManagerStorage.sol`, the `RewardsManagerV3Storage` contract
- In `RewardsManagerStorage.sol`, the `RewardsManagerV4Storage` contract
- In `RewardsManagerStorage.sol`, the `issuancePerBlock` state variable
- In `RewardsManagerStorage.sol`, the `RewardsManagerV5Storage` contract
- In `RewardsManagerStorage.sol`, the `subgraphService` state variable
- In `Staking.sol`, the `fallback` function
- In `StakingStorage.sol`, the `StakingV1Storage` contract
- In `SubgraphServiceStorage.sol`, the `SubgraphServiceV1Storage` abstract contract

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Require Statement Does Not Check for Any Conditions

> Auditor's comment: This seems like a high risk DOS issue as the function always reverts.

In Solidity, using `revert()` is recommended when no conditions are being checked.

In the file `BancorFormula.sol`, the line `495` uses `require(False)` where `revert()` would be more appropriate.

- The `require(False)` in the line `495`.

Consider replacing instances of `require(False)` with `revert()` for better clarity and maintainability of the codebase.

Require Statement With Multiple Conditions

Throughout the codebase, there are `require` statements that require multiple conditions to be satisfied.

- The `require( _tokensSlash != 0 && _tokensSlash <= maxTokensSlash, DisputeManagerInvalidTokensSlash(_tokensSlash, maxTokensSlash) )` in `DisputeManager.sol`.

- The `require( oldPendingGovernor != address(0) && msg.sender == oldPendingGovernor, "Caller must be pending governor" )` in `Governed.sol`.
- The `require(shares != 0 && shares >= _minSharesOut, HorizonStakingSlippageProtection(sh _minSharesOut))` in `HorizonStaking.sol`.

To simplify the codebase and raise the most helpful error messages for failing `require` statements, consider having a single require statement per condition.

Missing Error Message in Revert Statement

> Auditor's comment: The finding seems like a false positive. However revert message handling is incorrect. It doesn't account for `Panic(uint256)` and custom error messages. Especially for custom error messages with length greater than `0x64` it might result in reverts during decoding.

It is recommended to provide informative error messages in `revert` statements. This practice improves code clarity and assists with troubleshooting when a requirement is not satisfied.

Within `Multicall.sol`, there is a `revert` statement in line 24 that lacks an error message.

- The `revert` statement on line 24.

To improve the clarity and maintainability of the code, consider adding informative error messages to all `revert` statements.

Unsafe ABI Encoding

It is not an uncommon practice to use `abi.encodeWithSignature` or `abi.encodeWithSelector` to generate calldata for a low-level call. However, the first option is not typo-safe and the second option is not type-safe. The result is that both of these methods are error-prone and should be considered unsafe.

Throughout the codebase, there are multiples uses of unsafe ABI encodings:

- The use of `abi.encodeWithSelector` within the `HorizonStaking.sol`
- The use of `abi.encodeWithSelector` within the `Staking.sol`

Use `calldata` Instead of `memory`

When dealing with the parameters of `external` functions, it is more gas-efficient to read their arguments directly from `calldata` instead of storing them to `memory`. `calldata` is a read-only region of memory that contains the arguments of incoming `external` function calls. This makes using `calldata` as the data location for such parameters cheaper and more efficient compared to `memory`.

Thus, using `calldata` in such situations will generally save gas and improve the performance of a smart contract.

Throughout the codebase, multiple instances where function parameters should use `calldata` instead of `memory` were identified:

- In `DisputeManager.sol`, the `receipt` parameter
- In `DisputeManager.sol`, the `attestation1` parameter
- In `DisputeManager.sol`, the `attestation2` parameter
- In `GraphTallyCollector.sol`, the `data` parameter
- In `GraphTallyCollector.sol`, the `data` parameter

Consider using `calldata` as the data location for the parameters of `external` functions to optimize gas usage.

Duplicate Imports

Throughout the codebase, there are duplicate imports.

- Import `import "@openzeppelin/contracts/math/SafeMath.sol";` in `RewardsManager.sol`.
- Import `import "./IRewardsManager.sol";` in `RewardsManager.sol`.
- Import `import { IRewardsIssuer } from "./IRewardsIssuer.sol";` imports duplicated alias `IRewardsIssuer` in `RewardsManager.sol`.
- Import `import "@openzeppelin/contracts/math/SafeMath.sol";` in `Stakes.sol`.

Consider removing duplicate imports to improve the overall clarity and readability of the codebase.

File and Contract Names Mismatch

Throughout the codebase, there are multiple file names containing contracts with different names.

- The `Exponential.sol` file name does not match the `LibExponential` contract name.

To make the codebase easier to understand for developers and reviewers, consider renaming the files to match the contract names.

Todo Comments in the Code

During development, having well described TODO/Fixme comments will make the process of tracking and solving them easier. Without this information these comments might age and important information for the security of the system

might be forgotten by the time it is released to production. These comments should be tracked in the project's issue backlog and resolved before the system is deployed.

Throughout the codebase, multiple instances of TODO/Fixme comments were found.

- The `TODO` comment in line 59 of `GraphDirectory.sol`.
- The `TODO` comment in line 435 of `HorizonStaking.sol`.
- The `TODO` comment in line 668 of `HorizonStaking.sol`.
- The `TODO` comment in line 723 of `HorizonStaking.sol`.
- The `TODO` comment in line 253 of `HorizonStakingBase.sol`.
- The `TODO` comment in line 268 of `HorizonStakingBase.sol`.
- The `TODO` comment in line 22 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 63 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 77 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 225 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 235 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 246 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 260 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 288 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 316 of `HorizonStakingExtension.sol`.
- The `TODO` comment in line 11 of `HorizonStakingStorage.sol`.
- The `TODO` comment in line 19 of `IHorizonStakingBase.sol`.
- The `TODO` comment in line 5 of `IHorizonStakingTypes.sol`.
- The `TODO` comment in line 69 of `LegacyAllocation.sol`.
- The `TODO` comment in line 7 of `Managed.sol`.

Consider removing all instances of TODO/Fixme comments and instead tracking them in the issues backlog. Alternatively, consider linking each inline TODO/Fixme to the corresponding issues backlog entry.

Inconsistent Use of Named Returns in a Function

To improve the readability of the function, use the same return style.

Throughout the codebase, there are multiple instances where functions have inconsistent usage of named returns.

- In `_deprovision function`.
- In `_getThawingPeriodRange function`.
- In `_getThawingPeriodRange function`.
- In `_getVerifierCutRange function`.
- In `_getVerifierCutRange function`.
- In `_collectQueryFees function`.

Consider being consistent with the use of named returns throughout the functions.

Prefix Increment Operator `++i` Can Save Gas in Loops

Throughout the codebase, there are multiple opportunities where this optimization could be applied.

- The i++ in `Multicall.sol`.
- The i++ in `RewardsManager.sol`.

Consider using the prefix increment operator `++i` instead of the post-increment operator `i++` in order to save gas. This optimization skips storing the value before the incremental operation, as the return value of the expression is ignored.

Indecisive Licenses

Throughout the codebase, there are multiple files having indecisive SPDX licenses.

- The `GPL-3.0-or-later` SPDX license in `Allocation.sol`.
- The `GPL-3.0-or-later` SPDX license in `AllocationManager.sol`.
- The `GPL-3.0-or-later` SPDX license in `AllocationManagerStorage.sol`.
- The `GPL-3.0-or-later` SPDX license in `Attestation.sol`.
- The `GPL-3.0-or-later` SPDX license in `AttestationManager.sol`.
- The `GPL-3.0-or-later` SPDX license in `AttestationManagerStorage.sol`.
- The `GPL-3.0-or-later` SPDX license in `Authorizable.sol`.
- The `GPL-2.0-or-later` SPDX license in `Controller.sol`.
- The `GPL-2.0-or-later` SPDX license in `Curation.sol`.
- The `GPL-2.0-or-later` SPDX license in `CurationStorage.sol`.
- The `GPL-3.0-or-later` SPDX license in `DataService.sol`.
- The `GPL-3.0-or-later` SPDX license in `DataServiceFees.sol`.

- The GPL-3.0-or-later SPDX license in DataServiceFeesStorage.sol.
- The GPL-3.0-or-later SPDX license in DataServicePausable.sol.
- The GPL-3.0-or-later SPDX license in DataServicePausableUpgradeable.sol.
- The GPL-3.0-or-later SPDX license in DataServiceRescuable.sol.
- The GPL-3.0-or-later SPDX license in DataServiceStorage.sol.
- The GPL-3.0-or-later SPDX license in Denominations.sol.
- The GPL-3.0-or-later SPDX license in Directory.sol.
- The GPL-2.0-or-later SPDX license in DisputeManager.sol.
- The GPL-2.0-or-later SPDX license in DisputeManagerStorage.sol.
- The GPL-2.0-or-later SPDX license in Exponential.sol.
- The GPL-2.0-or-later SPDX license in ExponentialRebates.sol.
- The GPL-2.0-or-later SPDX license in Governed.sol.
- The GPL-2.0-or-later SPDX license in GraphDirectory.sol.
- The GPL-3.0-or-later SPDX license in GraphPayments.sol.
- The GPL-3.0-or-later SPDX license in GraphTallyCollector.sol.
- The GPL-2.0-or-later SPDX license in GraphUpgradeable.sol.
- The GPL-2.0-or-later SPDX license in HorizonStaking.sol.
- The GPL-2.0-or-later SPDX license in HorizonStakingBase.sol.
- The GPL-2.0-or-later SPDX license in HorizonStakingExtension.sol.
- The GPL-2.0-or-later SPDX license in HorizonStakingStorage.sol.
- The GPL-3.0-or-later SPDX license in IAuthorizable.sol.
- The GPL-2.0-or-later SPDX license in IController.sol.
- The GPL-2.0-or-later SPDX license in ICuration.sol.
- The GPL-3.0-or-later SPDX license in IDataService.sol.
- The GPL-3.0-or-later SPDX license in IDataServiceFees.sol.
- The GPL-3.0-or-later SPDX license in IDataServicePausable.sol.
- The GPL-3.0-or-later SPDX license in IDataServiceRescuable.sol.
- The GPL-2.0-or-later SPDX license in IDisputeManager.sol.
- The GPL-2.0-or-later SPDX license in IEpochManager.sol.
- The GPL-2.0-or-later SPDX license in IGNS.sol.
- The GPL-2.0-or-later SPDX license in IGraphCurationToken.sol.

- The GPL-3.0-or-later SPDX license in IGraphPayments.sol.
- The GPL-2.0-or-later SPDX license in IGraphProxy.sol.
- The GPL-2.0-or-later SPDX license in IGraphProxyAdmin.sol.
- The GPL-3.0-or-later SPDX license in IGraphTallyCollector.sol.
- The GPL-2.0-or-later SPDX license in IGraphToken.sol.
- The GPL-2.0-or-later SPDX license in IHorizonStaking.sol.
- The GPL-2.0-or-later SPDX license in IHorizonStakingBase.sol.
- The GPL-2.0-or-later SPDX license in IHorizonStakingExtension.sol.
- The GPL-2.0-or-later SPDX license in IHorizonStakingMain.sol.
- The GPL-2.0-or-later SPDX license in IHorizonStakingTypes.sol.
- The GPL-2.0-or-later SPDX license in IL2Curation.sol.
- The GPL-2.0-or-later SPDX license in IManaged.sol.
- The GPL-2.0-or-later SPDX license in IMulticall.sol.
- The GPL-2.0-or-later SPDX license in IPaymentsCollector.sol.
- The GPL-3.0-or-later SPDX license in IPaymentsEscrow.sol.
- The GPL-2.0-or-later SPDX license in IRewardsIssuer.sol.
- The GPL-2.0-or-later SPDX license in IRewardsManager.sol.
- The GPL-2.0-or-later SPDX license in IStaking.sol.
- The GPL-2.0-or-later SPDX license in IStakingBase.sol.
- The GPL-2.0-or-later SPDX license in IStakingData.sol.
- The GPL-2.0-or-later SPDX license in IStakingExtension.sol.
- The GPL-3.0-or-later SPDX license in ISubgraphService.sol.
- The GPL-2.0-or-later SPDX license in L2Curation.sol.
- The GPL-3.0-or-later SPDX license in LegacyAllocation.sol.
- The GPL-2.0-or-later SPDX license in LinkedList.sol.
- The GPL-2.0-or-later SPDX license in Managed.sol.
- The GPL-2.0-or-later SPDX license in Managed.sol.
- The GPL-2.0-or-later SPDX license in MathUtils.sol.
- The GPL-2.0-or-later SPDX license in MathUtils.sol.
- The GPL-2.0-or-later SPDX license in Multicall.sol.
- The GPL-3.0-or-later SPDX license in PPMMath.sol.

- The `GPL-2.0-or-later` SPDX license in `Pausable.sol`.
- The `GPL-3.0-or-later` SPDX license in `PaymentsEscrow.sol`.
- The `GPL-3.0-or-later` SPDX license in `ProvisionManager.sol`.
- The `GPL-3.0-or-later` SPDX license in `ProvisionManagerStorage.sol`.
- The `GPL-3.0-or-later` SPDX license in `ProvisionTracker.sol`.
- The `GPL-2.0-or-later` SPDX license in `RewardsManager.sol`.
- The `GPL-2.0-or-later` SPDX license in `RewardsManagerStorage.sol`.
- The `GPL-2.0-or-later` SPDX license in `Stakes.sol`.
- The `GPL-2.0-or-later` SPDX license in `Staking.sol`.
- The `GPL-2.0-or-later` SPDX license in `StakingExtension.sol`.
- The `GPL-2.0-or-later` SPDX license in `StakingStorage.sol`.
- The `GPL-3.0-or-later` SPDX license in `SubgraphService.sol`.
- The `GPL-3.0-or-later` SPDX license in `SubgraphServiceStorage.sol`.
- The `GPL-2.0-or-later` SPDX license in `TokenUtils.sol`.
- The `GPL-3.0-or-later` SPDX license in `UintRange.sol`.

Consider specifying only one license to prevent possible licensing ambiguity.

Lack of Indexed Event Parameters

Throughout the codebase, several events do not have indexed parameters:

- The `MaxPOIStalenessSet` event of `AllocationManager.sol`.
- The `SubgraphServiceDirectoryInitialized` event of `Directory.sol`.
- The `DisputePeriodSet` event of `IDisputeManager.sol`.
- The `DisputeDepositSet` event of `IDisputeManager.sol`.
- The `MaxSlashingCutSet` event of `IDisputeManager.sol`.
- The `FishermanRewardCutSet` event of `IDisputeManager.sol`.
- The `MaxThawingPeriodSet` event of `IHorizonStakingMain.sol`.
- The `StakeToFeesRatioSet` event of `ISubgraphService.sol`.
- The `CurationCutSet` event of `ISubgraphService.sol`.
- The `ParameterUpdated` event of `Managed.sol`.
- The `SetController` event of `Managed.sol`.
- The `PartialPauseChanged` event of `Pausable.sol`.

- The `PauseChanged` event of `Pausable.sol`.
- The `ProvisionTokensRangeSet` event of `ProvisionManager.sol`.
- The `DelegationRatioSet` event of `ProvisionManager.sol`.
- The `VerifierCutRangeSet` event of `ProvisionManager.sol`.
- The `ThawingPeriodRangeSet` event of `ProvisionManager.sol`.

To improve the ability of off-chain services to search and filter for specific events, consider indexing event parameters.

Lack of Security Contact

Providing a specific security contact (such as an email or ENS name) within a smart contract significantly simplifies the process for individuals to communicate if they identify a vulnerability in the code. This practice is quite beneficial as it permits the code owners to dictate the communication channel for vulnerability disclosure, eliminating the risk of miscommunication or failure to report due to a lack of knowledge on how to do so. In addition, if the contract incorporates third-party libraries and a bug surfaces in those, it becomes easier for their maintainers to contact the appropriate person about the problem and provide mitigation instructions.

Throughout the codebase, there are contracts that do not have a security contact:

- The `Allocation` library.
- The `AllocationManager` abstract contract.
- The `AllocationManagerV1Storage` abstract contract.
- The `Attestation` library.
- The `AttestationManager` abstract contract.
- The `AttestationManagerV1Storage` abstract contract.
- The `Controller` contract.
- The `Curation` contract.
- The `CurationV1Storage` abstract contract.
- The `CurationV2Storage` abstract contract.
- The `CurationV3Storage` abstract contract.
- The `DataService` abstract contract.
- The `DataServiceFees` abstract contract.
- The `DataServiceFeesV1Storage` abstract contract.

- The `DataServicePausable` abstract contract.
- The `DataServicePausableUpgradeable` abstract contract.
- The `DataServiceRescuable` abstract contract.
- The `DataServiceV1Storage` abstract contract.
- The `Denominations` library.
- The `Directory` abstract contract.
- The `DisputeManagerV1Storage` abstract contract.
- The `LibExponential` library.
- The `ExponentialRebates` library.
- The `Governed` abstract contract.
- The `GraphDirectory` abstract contract.
- The `GraphUpgradeable` abstract contract.
- The `HorizonStakingBase` abstract contract.
- The `HorizonStakingV1Storage` abstract contract.
- The `IAuthorizable` interface.
- The `IController` interface.
- The `ICuration` interface.
- The `IDataService` interface.
- The `IDataServiceFees` interface.
- The `IDataServicePausable` interface.
- The `IDataServiceRescuable` interface.
- The `IDisputeManager` interface.
- The `IEpochManager` interface.
- The `IGNS` interface.
- The `IGraphCurationToken` interface.
- The `IGraphPayments` interface.
- The `IGraphProxy` interface.
- The `IGraphProxyAdmin` interface.
- The `IGraphTallyCollector` interface.
- The `IGraphToken` interface.
- The `IHorizonStaking` interface.

- The `IHorizonStakingBase` interface.
- The `IHorizonStakingExtension` interface.
- The `IHorizonStakingMain` interface.
- The `IHorizonStakingTypes` interface.
- The `IL2Curation` interface.
- The `IManaged` interface.
- The `IMulticall` interface.
- The `IPaymentsCollector` interface.
- The `IPaymentsEscrow` interface.
- The `IRewardsIssuer` interface.
- The `IRewardsManager` interface.
- The `IStaking` interface.
- The `IStakingBase` interface.
- The `IStakingData` interface.
- The `IStakingExtension` interface.
- The `ISubgraphService` interface.
- The `ITokenGateway` interface.
- The `L2Curation` contract.
- The `LegacyAllocation` library.
- The `LibFixedMath` library.
- The `LibFixedMath` library.
- The `LinkedList` library.
- The `Managed` abstract contract.
- The `Managed` abstract contract.
- The `MathUtils` library.
- The `MathUtils` library.
- The `Multicall` abstract contract.
- The `PPMMath` library.
- The `Pausable` abstract contract.
- The `ProvisionManager` abstract contract.
- The `ProvisionManagerV1Storage` abstract contract.

- The `ProvisionTracker` library.
- The `RewardsManager` contract.
- The `RewardsManagerV1Storage` contract.
- The `RewardsManagerV2Storage` contract.
- The `RewardsManagerV3Storage` contract.
- The `RewardsManagerV4Storage` contract.
- The `RewardsManagerV5Storage` contract.
- The `Stakes` library.
- The `Staking` abstract contract.
- The `StakingExtension` contract.
- The `StakingV1Storage` contract.
- The `StakingV2Storage` contract.
- The `StakingV3Storage` contract.
- The `StakingV4Storage` contract.
- The `SubgraphServiceV1Storage` abstract contract.
- The `TokenUtils` library.
- The `UintRange` library.

Consider adding a NatSpec comment containing a security contact above each contract definition. Using the `@custom:security-contact` convention is recommended as it has been adopted by the OpenZeppelin Wizard and the ethereum-lists.

Modifiers Could Be Used

Throughout the codebase, there are modifiers that could be used:

- The `register` function could use the `onlyRegisteredIndexer` modifier in `SubgraphService.sol`.

Consider using any modifier that could be used, to avoid code duplication.

Non-explicit Imports Are Used

The use of non-explicit imports in the codebase can decrease code clarity and may create naming conflicts between locally-defined and imported variables. This is particularly relevant when multiple contracts exist within the same Solidity file or when inheritance chains are long.

Throughout the codebase, global imports are being used:

- The import "@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol"; import in `IGraphCurationToken.sol`.
- The import "@openzeppelin/contracts/token/ERC20/IERC20.sol"; import in `IGraphToken.sol`.
- The import "@openzeppelin/contracts/math/SafeMath.sol"; import in `MathUtils.sol`.
- The import "./IMulticall.sol"; import in `Multicall.sol`.
- The import "@openzeppelin/contracts/math/SafeMath.sol"; import in `RewardsManager.sol`.
- The import "../upgrades/GraphUpgradeable.sol"; import in `RewardsManager.sol`.
- The import "../staking/libs/MathUtils.sol"; import in `RewardsManager.sol`.
- The import "./RewardsManagerStorage.sol"; import in `RewardsManager.sol`.
- The import "./IRewardsManager.sol"; import in `RewardsManager.sol`.
- The import "./IRewardsManager.sol"; import in `RewardsManagerStorage.sol`.
- The import "../governance/Managed.sol"; import in `RewardsManagerStorage.sol`.
- The import "@openzeppelin/contracts/math/SafeMath.sol"; import in `Stakes.sol`.
- The import "./MathUtils.sol"; import in `Stakes.sol`.
- The import "../token/IGraphToken.sol"; import in `TokenUtils.sol`.

Following the principle that clearer code is better code, consider using the named import syntax *(`import {A, B, C} from "X"`)* to explicitly declare which contracts are being imported.

Redundant Getter Functions

When state variables use public visibility in a contract, a getter method for the variable is automatically included.

Throughout the codebase, there are various redundant getter functions.

- Within the `Controller` contract in `Controller.sol`, the `getGovernor` function is redundant because the `governor` state variable already has a getter.
- Within the `DisputeManager` contract in `DisputeManager.sol`, the `getVerifierCut` function is redundant because the `fishermanRewardCut` state variable already has a getter.
- Within the `DisputeManager` contract in `DisputeManager.sol`, the `getDisputePeriod` function is redundant because the `disputePeriod` state variable already has a getter.

To improve the overall clarity, intent, and readability of the codebase, consider removing the redundant getter functions.

Redundant Return Statement

To improve the readability of the contract, it is recommended to remove redundant return statements from functions that have named returns.

Throughout the codebase, there are multiple instances of redundant return statements:

- The `return tokensThawed_;` statement in `HorizonStaking.sol`.
- The `return _getIdleStake(serviceProvider);` statement in `HorizonStakingBase.sol`.
- The `return f / FIXED_1;` statement in `LibFixedMath.sol`.
- The `return f / FIXED_1;` statement in `LibFixedMath.sol`.
- The `return (minimumProvisionTokens, maximumProvisionTokens);` statement in `ProvisionManager.sol`.
- The `return (minimumThawingPeriod, maximumThawingPeriod);` statement in `ProvisionManager.sol`.
- The `return (minimumVerifierCut, maximumVerifierCut);` statement in `ProvisionManager.sol`.
- The `return (_disputeManager().getDisputePeriod(), DEFAULT_MAX_THAWING_PERIOD);` statement in `SubgraphService.sol`.
- The `return (_disputeManager().getVerifierCut(), DEFAULT_MAX_VERIFIER_CUT);` statement in `SubgraphService.sol`.
- The `return tokensCollected;` statement in `SubgraphService.sol`.

Consider removing the redundant return statement in functions with named returns to improve the readability of the contract.

Use Custom Errors

> Auditor's comment: Multicall contract expects `Error(string)` revert messages, so using custom errors might cause an issue there.

Since Solidity version 0.8.4, custom errors provide a cleaner and more cost-efficient way to explain to users why an operation failed.

Throughout the codebase, instances of revert and/or require messages were found:

- In the `Controller.sol` file, the `require(msg.sender == governor || msg.sender == pauseGuardian, "Only Governor or Guardian can call")` statement.

- In the `Controller.sol` file, the `require(_contractAddress != address(0), "Contract address must be set")` statement.
- In the `Controller.sol` file, the `require(_controller != address(0), "Controller must be set")` statement.
- In the `Controller.sol` file, the `require(_newPauseGuardian != address(0), "PauseGuardian must be set")` statement.
- In the `Governed.sol` file, the `require(msg.sender == governor, "Only Governor can call")` statement.
- In the `Governed.sol` file, the `require(_newGovernor != address(0), "Governor must be set")` statement.
- In the `Governed.sol` file, the `require( oldPendingGovernor != address(0) && msg.sender == oldPendingGovernor, "Caller must be pending governor" )` statement.
- In the `GraphUpgradeable.sol` file, the `require(msg.sender == _proxy.admin(), "Caller must be the proxy admin")` statement.
- In the `GraphUpgradeable.sol` file, the `require(msg.sender == _implementation(), "Only implementation")` statement.
- In the `HorizonStaking.sol` file, the `require(tokensToWithdraw > 0, "!tokens")` statement.
- In the `HorizonStaking.sol` file, the `require(success, "Delegatecall: legacySlash failed")` statement.
- In the `HorizonStakingBase.sol` file, the `revert("RECEIVE_ETH_NOT_ALLOWED")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(msg.sender == address(_graphTokenGateway()), "ONLY_GATEWAY")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(__DEPRECATED_slashers[msg.sender] == true, "!slasher")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(allocationID != address(0), "!alloc")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(allocState != AllocationState.Null, "!collect")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(tokens > 0, "!tokens")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(tokens >= reward, "rewards>slash")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(indexerStake.tokensStaked > 0, "!stake")` statement.

- In the `HorizonStakingExtension.sol` file, the `require(tokens <= indexerStake.tokensStaked, "slash>stake")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(beneficiary != address(0), "!beneficiary")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(allocState == AllocationState.Active, "!active")` statement.
- In the `HorizonStakingExtension.sol` file, the `require(isIndexerOrOperator, "!auth")` statement.
- In the `LibFixedMath.sol` file, the `revert("out-of-bounds")` statement.
- In the `LibFixedMath.sol` file, the `revert("out-of-bounds")` statement.
- In the `LibFixedMath.sol` file, the `revert("out-of-bounds")` statement.
- In the `LibFixedMath.sol` file, the `revert("overflow")` statement.
- In the `LibFixedMath.sol` file, the `revert("overflow")` statement.
- In the `LibFixedMath.sol` file, the `revert("overflow")` statement.
- In the `LibFixedMath.sol` file, the `revert("overflow")` statement.
- In the `TokenUtils.sol` file, the `require(_graphToken.transferFrom(_from, address(this), _amount), "!transfer")` statement.
- In the `TokenUtils.sol` file, the `require(_graphToken.transfer(_to, _amount), "!transfer")` statement.

For conciseness and gas savings, consider replacing require and revert messages with custom errors.

Unnecessary Casts

Throughout the codebase, multiple instances of unnecessary casts were identified:

- The `address(subgraphAvailabilityOracle)` cast in the `RewardsManager` contract
- The `address(subgraphAvailabilityOracle)` cast in the `RewardsManager` contract
- The `uint256(_percentage)` cast in the `StakingExtension` contract

To improve the overall clarity and intent of the codebase, consider removing any unnecessary casts.

Unsafe Casts

Throughout the codebase, there are unsafe casts.

- The `int32` cast in the `LibExponential` contract.
- The `int32` cast in the `LibExponential` contract.
- The `int32` cast in the `LibExponential` contract.
- The `int32` cast in the `LibExponential` contract.
- The `int256` cast in the `LibExponential` contract.
- The `int256` cast in the `LibExponential` contract.
- The `int32` cast in the `ExponentialRebates` contract.
- The `int32` cast in the `ExponentialRebates` contract.
- The `int32` cast in the `ExponentialRebates` contract.
- The `int32` cast in the `ExponentialRebates` contract.
- The `int256` cast in the `ExponentialRebates` contract.
- The `int256` cast in the `ExponentialRebates` contract.
- The `int256` cast in the `ExponentialRebates` contract.
- The `uint160` cast in the `SubgraphService` contract.

To avoid unexpected behavior of the codebase, ensure that the casts are used well.

Unused Functions With Internal or Private Visibility

Throughout the codebase, there are unused functions.

- The `_getVerifierCutRange` function in `SubgraphService.sol`.

To improve the overall clarity, intentionality, and readability of the codebase, consider using or removing any currently unused functions.

Unused Imports

Throughout the codebase, there are imports that are unused and could be removed.

- The import `import { ICuration } from "./ICuration.sol";` imports unused alias `ICuration` in `Curation.sol`.
- The import `import { IStakingBase } from "../staking/IStakingBase.sol";` imports unused alias `IStakingBase` in `Managed.sol`.

Consider removing unused imports to improve the overall clarity and readability of the codebase.

Unused Modifiers

Throughout the codebase, there are unused modifiers:

- The `onlyL2Gateway` modifier in `HorizonStakingExtension.sol`.

To improve the overall clarity, intentionality, and readability of the codebase, consider either using or removing any unused modifier.

Unused Named Return Variables

Named return variables are a way to declare variables that are meant to be used within a function's body for the purpose of being returned as that function's output. They are an alternative to explicit in-line `return` statements.

Throughout the codebase, multiple instances of unused named return variables were identified:

- In `HorizonStaking.sol`, the `tokensThawed` return variable of the `_deprovision` function
- In `HorizonStakingBase.sol`, the `tokens` return variable of the `getIdleStake` function
- In `LibFixedMath.sol`, the `n` return variable of the `toInteger` function
- In `LibFixedMath.sol`, the `r` return variable of the `exp` function
- In `LibFixedMath.sol`, the `n` return variable of the `toInteger` function
- In `LibFixedMath.sol`, the `r` return variable of the `ln` function
- In `LibFixedMath.sol`, the `c` return variable of the `_mul` function
- In `LibFixedMath.sol`, the `r` return variable of the `exp` function
- In `LibFixedMath.sol`, the `c` return variable of the `_mul` function
- In `ProvisionManager.sol`, the `min` return variable of the `_getProvisionTokensRange` function
- In `ProvisionManager.sol`, the `max` return variable of the `_getProvisionTokensRange` function
- In `ProvisionManager.sol`, the `min` return variable of the `_getThawingPeriodRange` function
- In `ProvisionManager.sol`, the `max` return variable of the `_getThawingPeriodRange` function
- In `ProvisionManager.sol`, the `min` return variable of the `_getVerifierCutRange` function
- In `ProvisionManager.sol`, the `max` return variable of the `_getVerifierCutRange` function

- In `SubgraphService.sol`, the `min` return variable of the `_getThawingPeriodRange` function
- In `SubgraphService.sol`, the `max` return variable of the `_getThawingPeriodRange` function
- In `SubgraphService.sol`, the `min` return variable of the `_getVerifierCutRange` function
- In `SubgraphService.sol`, the `max` return variable of the `_getVerifierCutRange` function
- In `SubgraphService.sol`, the `feesCollected` return variable of the `_collectQueryFees` function

Consider either using or removing any unused named return variables.

Variables Could Be `constant`

If a variable is only ever assigned a value when it is declared, then it could be declared as `constant`.

Throughout the codebase, there are variables that could be `constant`.

- The `fixedReserveRatio` state variable variable in `L2Curation.sol`.
- The `CURATION` state variable variable in `Managed.sol`.
- The `EPOCH_MANAGER` state variable variable in `Managed.sol`.
- The `REWARDS_MANAGER` state variable variable in `Managed.sol`.
- The `STAKING` state variable variable in `Managed.sol`.
- The `GRAPH_TOKEN` state variable variable in `Managed.sol`.
- The `GRAPH_TOKEN_GATEWAY` state variable variable in `Managed.sol`.
- The `GNS` state variable variable in `Managed.sol`.

To better convey the intended use of variables and to potentially save gas, consider adding the `constant` keyword to variables that are only set when they are declared.