1. Make contracts upgradeable

Modified contracts:

packages/contracts/contracts/ActivePool.sol — View file @ d8fdb7e5

```
@@ -4,7 +4,7 @@ pragma solidity 0.6.11;

 import './Interfaces/IActivePool.sol';
 import "./Dependencies/SafeMath.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
 import "./Dependencies/CheckContract.sol";
 import "./Dependencies/console.sol";

@@ -15,7 +15,7 @@ import "./Dependencies/console.sol";
  * Stability Pool, the Default Pool, or both, depending on the liquidation conditions.
  *
  */
-contract ActivePool is Ownable, CheckContract, IActivePool {
+contract ActivePool is OwnableUpgradeable, CheckContract, IActivePool {
     using SafeMath for uint256;

     string constant public NAME = "ActivePool";
@@ -36,6 +36,10 @@ contract ActivePool is Ownable, CheckContract, IActivePool {

     // --- Contract setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses(
         address _borrowerOperationsAddress,
         address _troveManagerAddress,
@@ -60,7 +64,7 @@ contract ActivePool is Ownable, CheckContract, IActivePool {
         emit StabilityPoolAddressChanged(_stabilityPoolAddress);
         emit DefaultPoolAddressChanged(_defaultPoolAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // --- Getters for public variables. Required by IPool interface ---
```

packages/contracts/contracts/BorrowerOperations.sol — View file @ d8fdb7e5

```
@@ -9,11 +9,11 @@ import "./Interfaces/ICollSurplusPool.sol";
 import "./Interfaces/ISortedTroves.sol";
 import "./Interfaces/ILQTYStaking.sol";
 import "./Dependencies/LiquityBase.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
 import "./Dependencies/CheckContract.sol";
 import "./Dependencies/console.sol";

-contract BorrowerOperations is LiquityBase, Ownable, CheckContract, IBorrowerOperations {
+contract BorrowerOperations is LiquityBase, OwnableUpgradeable, CheckContract, IBorrowerOperations {
     string constant public NAME = "BorrowerOperations";

     // --- Connected contract declarations ---
@@ -95,6 +95,10 @@ contract BorrowerOperations is LiquityBase, Ownable, CheckContract, IBorrowerOpe

     // --- Dependency setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses(
         address _troveManagerAddress,
         address _activePoolAddress,
@@ -148,7 +152,7 @@ contract BorrowerOperations is LiquityBase, Ownable, CheckContract, IBorrowerOpe
         emit LUSDTokenAddressChanged(_lusdTokenAddress);
         emit LQTYStakingAddressChanged(_lqtyStakingAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // --- Borrower Trove Operations ---
```

packages/contracts/contracts/CollSurplusPool.sol

View file @ d8fdb7e5

```
@@ -4,12 +4,12 @@ pragma solidity 0.6.11;

 import "./Interfaces/ICollSurplusPool.sol";
 import "./Dependencies/SafeMath.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
 import "./Dependencies/CheckContract.sol";
 import "./Dependencies/console.sol";


-contract CollSurplusPool is Ownable, CheckContract, ICollSurplusPool {
+contract CollSurplusPool is OwnableUpgradeable, CheckContract, ICollSurplusPool {
     using SafeMath for uint256;

     string constant public NAME = "CollSurplusPool";
@@ -34,6 +34,10 @@ contract CollSurplusPool is Ownable, CheckContract, ICollSurplusPool {

     // --- Contract setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses(
         address _borrowerOperationsAddress,
         address _troveManagerAddress,
@@ -55,7 +59,7 @@ contract CollSurplusPool is Ownable, CheckContract, ICollSurplusPool {
         emit TroveManagerAddressChanged(_troveManagerAddress);
         emit ActivePoolAddressChanged(_activePoolAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     /* Returns the ETH state variable at ActivePool address.
```

packages/contracts/contracts/DefaultPool.sol

View file @ d8fdb7e5

```
@@ -4,7 +4,7 @@ pragma solidity 0.6.11;

 import './Interfaces/IDefaultPool.sol';
 import "./Dependencies/SafeMath.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
 import "./Dependencies/CheckContract.sol";
 import "./Dependencies/console.sol";

@@ -15,7 +15,7 @@ import "./Dependencies/console.sol";
  * from the Default Pool to the Active Pool.
  */
-contract DefaultPool is Ownable, CheckContract, IDefaultPool {
+contract DefaultPool is OwnableUpgradeable, CheckContract, IDefaultPool {
     using SafeMath for uint256;

     string constant public NAME = "DefaultPool";
@@ -31,6 +31,10 @@ contract DefaultPool is Ownable, CheckContract, IDefaultPool {

     // --- Dependency setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses(
         address _troveManagerAddress,
         address _activePoolAddress
@@ -47,7 +51,7 @@ contract DefaultPool is Ownable, CheckContract, IDefaultPool {
         emit TroveManagerAddressChanged(_troveManagerAddress);
         emit ActivePoolAddressChanged(_activePoolAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // --- Getters for public variables. Required by IPool interface ---
```

packages/contracts/contracts/HintHelpers.sol — View file @ 59e8e2ba

```
@@ -5,10 +5,10 @@ pragma solidity 0.6.11;
 import "./Interfaces/ITroveManager.sol";
 import "./Interfaces/ISortedTroves.sol";
 import "./Dependencies/LiquityBase.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
 import "./Dependencies/CheckContract.sol";

-contract HintHelpers is LiquityBase, Ownable, CheckContract {
+contract HintHelpers is LiquityBase, OwnableUpgradeable, CheckContract {
     string constant public NAME = "HintHelpers";

     ISortedTroves public sortedTroves;
@@ -21,6 +21,10 @@ contract HintHelpers is LiquityBase, Ownable, CheckContract {

     // --- Dependency setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses(
         address _sortedTrovesAddress,
         address _troveManagerAddress
@@ -37,7 +41,7 @@ contract HintHelpers is LiquityBase, Ownable, CheckContract {
         emit SortedTrovesAddressChanged(_sortedTrovesAddress);
         emit TroveManagerAddressChanged(_troveManagerAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // --- Functions ---
```

packages/contracts/contracts/LQTY/LQTYStaking.sol — View file @ 59e8e2ba

```
@@ -4,7 +4,7 @@ pragma solidity 0.6.11;

 import "../Dependencies/BaseMath.sol";
 import "../Dependencies/SafeMath.sol";
-import "../Dependencies/Ownable.sol";
+import "../Dependencies/OwnableUpgradeable.sol";
 import "../Dependencies/CheckContract.sol";
 import "../Dependencies/console.sol";
 import "../Interfaces/ILQTYToken.sol";
@@ -12,7 +12,7 @@ import "../Interfaces/ILQTYStaking.sol";
 import "../Dependencies/LiquityMath.sol";
 import "../Interfaces/ILUSDToken.sol";

-contract LQTYStaking is ILQTYStaking, Ownable, CheckContract, BaseMath {
+contract LQTYStaking is ILQTYStaking, OwnableUpgradeable, CheckContract, BaseMath {
     using SafeMath for uint;

     // --- Data ---
@@ -57,6 +57,10 @@ contract LQTYStaking is ILQTYStaking, Ownable, CheckContract, BaseMath {

     // --- Functions ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setAddresses
     (
         address _lqtyTokenAddress,
@@ -87,7 +91,7 @@ contract LQTYStaking is ILQTYStaking, Ownable, CheckContract, BaseMath {
         emit BorrowerOperationsAddressSet(_borrowerOperationsAddress);
         emit ActivePoolAddressSet(_activePoolAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // If caller has a pre-existing stake, send any accumulated ETH and LUSD gains to them.
```

packages/contracts/contracts/LPRewards/Unipool.sol

View file @ 1cb68ffc

```
@@ -4,7 +4,7 @@ pragma solidity 0.6.11;

 import "../Dependencies/LiquityMath.sol";
 import "../Dependencies/SafeMath.sol";
-import "../Dependencies/Ownable.sol";
+import "../Dependencies/OwnableUpgradeable.sol";
 import "../Interfaces/ILQTYToken.sol";
 import "./Dependencies/SafeERC20.sol";
@@ -71,14 +71,14 @@ contract LPTokenWrapper is ILPTokenWrapper {
  * either LQTY token contract is deployed, and therefore LQTY tokens are minted to Unipool contract,
  * or first liquidity provider stakes UNIv2 LP tokens into it.
  */
-contract Unipool is LPTokenWrapper, Ownable, CheckContract, IUnipool {
+contract Unipool is LPTokenWrapper, OwnableUpgradeable, CheckContract, IUnipool {
     string constant public NAME = "Unipool";

     uint256 public duration;
     ILQTYToken public lqtyToken;

-    uint256 public periodFinish = 0;
-    uint256 public rewardRate = 0;
+    uint256 public periodFinish;
+    uint256 public rewardRate;
     uint256 public lastUpdateTime;
     uint256 public rewardPerTokenStored;
     mapping(address => uint256) public userRewardPerTokenPaid;
@@ -92,6 +92,12 @@ contract Unipool is LPTokenWrapper, Ownable, CheckContract, IUnipool {
     event RewardPaid(address indexed user, uint256 reward);

     // initialization function
+    function initialize() public initializer {
+        __Ownable_init();
+        periodFinish = 0;
+        rewardRate = 0;
+    }
+
     function setParams(
         address _lqtyTokenAddress,
         address _uniTokenAddress,
```


```
    function initialize() public initializer {
        __Ownable_init();
        periodFinish = 0;
        rewardRate = 0;
    }

    function setParams(
        address _lqtyTokenAddress,
        address _uniTokenAddress,
```


```
@@ -113,7 +119,7 @@ contract Unipool is LPTokenWrapper, Ownable, CheckContract, IUnipool {
         emit LQTYTokenAddressChanged(_lqtyTokenAddress);
         emit UniTokenAddressChanged(_uniTokenAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     // Returns current timestamp if the rewards program has not finished yet, end time otherwise
```

packages/contracts/contracts/LQTY/CommunityIssuance.sol

View file @ 1cb68ffc

```
@@ -6,12 +6,12 @@ import "../Interfaces/ILQTYToken.sol";
 import "../Interfaces/ICommunityIssuance.sol";
 import "../Dependencies/BaseMath.sol";
 import "../Dependencies/LiquityMath.sol";
-import "../Dependencies/Ownable.sol";
+import "../Dependencies/OwnableUpgradeable.sol";
 import "../Dependencies/CheckContract.sol";
 import "../Dependencies/SafeMath.sol";


-contract CommunityIssuance is ICommunityIssuance, Ownable, CheckContract, BaseMath {
+contract CommunityIssuance is ICommunityIssuance, OwnableUpgradeable, CheckContract, BaseMath {
     using SafeMath for uint;

     // --- Data ---
```

```
@@ -59,7 +59,8 @@ contract CommunityIssuance is ICommunityIssuance, Ownable, CheckContract, BaseMa

    // --- Functions ---

-    constructor() public {
+    function initialize() public initializer {
+        __Ownable_init();
         deploymentTime = block.timestamp;
     }

@@ -79,13 +80,15 @@ contract CommunityIssuance is ICommunityIssuance, Ownable, CheckContract, BaseMa
         stabilityPoolAddress = _stabilityPoolAddress;

         // When LQTYToken deployed, it should have transferred CommunityIssuance's LQTY entitlement
+        LQTYSupplyCap = lqtyToken.getCommunityIssuanceEntitlement();
+        assert(LQTYSupplyCap > 0);
         uint LQTYBalance = lqtyToken.balanceOf(address(this));
-        assert(LQTYBalance >= LQTYSupplyCap);
+        assert(LQTYBalance == LQTYSupplyCap);

         emit LQTYTokenAddressSet(_lqtyTokenAddress);
         emit StabilityPoolAddressSet(_stabilityPoolAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     function issueLQTY() external override returns (uint) {
```

packages/contracts/contracts/LQTY/LQTYToken.sol — View file @ 1cb68ffc

```
-contract LQTYToken is CheckContract, ILQTYToken {
+contract LQTYToken is CheckContract, ILQTYToken, Initializable {
     using SafeMath for uint256;

     // --- ERC20 Data ---

-    string constant internal _NAME = "LQTY";
-    string constant internal _SYMBOL = "LQTY";
+    string constant internal _NAME = "PIGGY";
+    string constant internal _SYMBOL = "PIGGY";
     string constant internal _VERSION = "1";
     uint8 constant internal  _DECIMALS = 18;

@@ -70,30 +77,32 @@ contract LQTYToken is CheckContract, ILQTYToken {

     // Cache the domain separator as an immutable value, but also store the chain id that it corresponds to, in order to
     // invalidate the cached domain separator if the chain id changes.
-    bytes32 private immutable _CACHED_DOMAIN_SEPARATOR;
-    uint256 private immutable _CACHED_CHAIN_ID;
+    bytes32 private _CACHED_DOMAIN_SEPARATOR;
+    uint256 private _CACHED_CHAIN_ID;

-    bytes32 private immutable _HASHED_NAME;
-    bytes32 private immutable _HASHED_VERSION;
+    bytes32 private _HASHED_NAME;
+    bytes32 private _HASHED_VERSION;

     mapping (address => uint256) private _nonces;
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
85  92      // uint for use with SafeMath
86      -   uint internal _1_MILLION = 1e24;    // 1e6 * 1e18 = 1e24
    93  +   uint internal constant _1_MILLION = 1e24;    // 1e6 * 1e18 = 1e24
87  94
88      -   uint internal immutable deploymentStartTime;
89      -   address public immutable multisigAddress;
    95  +   uint internal deploymentStartTime;
    96  +   address public teamAddress;
    97  +   address public investorAddress;
90  98
91      -   address public immutable communityIssuanceAddress;
92      -   address public immutable lqtyStakingAddress;
    99  +   address public communityIssuanceAddress;
    100 +   address public lqtyStakingAddress;
93  101
94      -   uint internal immutable lpRewardsEntitlement;
    102 +   uint internal lpRewardsEntitlement;
    103 +   uint internal communityIssuanceEntitlement;
95  104
96      -   ILockupContractFactory public immutable lockupContractFactory;
    105 +   ILockupContractFactory public lockupContractFactory;
97  106
98  107     // --- Events ---
99  108
... ...   @@ -103,22 +112,27 @@ contract LQTYToken is CheckContract, ILQTYToken {
103 112
104 113     // --- Functions ---
105 114
106     -   constructor
    115 +   function initialize
107 116     (
108 117         address _communityIssuanceAddress,
109 118         address _lqtyStakingAddress,
110 119         address _lockupFactoryAddress,
111     -       address _bountyAddress,
112 120         address _lpRewardsAddress,
113     -       address _multisigAddress
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
104 113     // --- Functions ---
105 114
106     -   constructor
    115 +   function initialize
107 116     (
108 117         address _communityIssuanceAddress,
109 118         address _lqtyStakingAddress,
110 119         address _lockupFactoryAddress,
111     -       address _bountyAddress,
112 120         address _lpRewardsAddress,
113     -       address _multisigAddress
    121 +       address _contributorMiningAddress,
    122 +       address _treasuryAddress,
    123 +       address _airdropAddress,
    124 +       address _teamAddress,
    125 +       address _investorAddress
114 126     )
115     -       public
    127 +       public
    128 +       initializer
116 129     {
117 130         checkContract(_communityIssuanceAddress);
118 131         checkContract(_lqtyStakingAddress);
119 132         checkContract(_lockupFactoryAddress);
120 133
121     -       multisigAddress = _multisigAddress;
    134 +       teamAddress = _teamAddress;
    135 +       investorAddress = _investorAddress;
122 136         deploymentStartTime  = block.timestamp;
123 137
124 138         communityIssuanceAddress = _communityIssuanceAddress;
... ...   @@ -135,23 +149,21 @@ contract LQTYToken is CheckContract, ILQTYToken {
```

packages/contracts/contracts/LQTY/LockupContractFactory.sol — View file @ 1cb68ffc

```
@@ -4,7 +4,7 @@ pragma solidity 0.6.11;

 import "../Dependencies/CheckContract.sol";
 import "../Dependencies/SafeMath.sol";
-import "../Dependencies/Ownable.sol";
+import "../Dependencies/OwnableUpgradeable.sol";
 import "../Interfaces/ILockupContractFactory.sol";
 import "./LockupContract.sol";
 import "../Dependencies/console.sol";
@@ -16,21 +16,21 @@ import "../Dependencies/console.sol";
 * This registry is checked by LQTYToken when the Liquity deployer attempts to transfer LQTY tokens. During the first year
 * since system deployment, the Liquity deployer is only allowed to transfer LQTY to valid LockupContracts that have been
 * deployed by and recorded in the LockupContractFactory. This ensures the deployer's LQTY can't be traded or staked in the
-* first year, and can only be sent to a verified LockupContract which unlocks at least one year after system deployment.
+* first half year, and can only be sent to a verified LockupContract which unlocks at least half year after system deployment.
 *
 * LockupContracts can of course be deployed directly, but only those deployed through and recorded in the LockupContractFactory
 * will be considered "valid" by LQTYToken. This is a convenient way to verify that the target address is a genuine
 * LockupContract.
 */

-contract LockupContractFactory is ILockupContractFactory, Ownable, CheckContract {
+contract LockupContractFactory is ILockupContractFactory, OwnableUpgradeable, CheckContract {
     using SafeMath for uint;

     // --- Data ---
     string constant public NAME = "LockupContractFactory";

-    uint constant public SECONDS_IN_ONE_YEAR = 31536000;
-
     address public lqtyTokenAddress;

     mapping (address => address) public lockupContractToDeployer;
@@ -42,13 +40,17 @@ contract LockupContractFactory is ILockupContractFactory, Ownable, CheckContract

     // --- Functions ---

+    function initialize() public initializer {
```

packages/contracts/contracts/LQTY/LockupContractFactory.sol — View file @ 1cb68ffc

```
@@ -42,13 +40,17 @@ contract LockupContractFactory is ILockupContractFactory, Ownable, CheckContract

     // --- Functions ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
     function setLQTYTokenAddress(address _lqtyTokenAddress) external override onlyOwner {
         checkContract(_lqtyTokenAddress);

         lqtyTokenAddress = _lqtyTokenAddress;
         emit LQTYTokenAddressSet(_lqtyTokenAddress);

-        _renounceOwnership();
+        renounceOwnership();
     }

     function deployLockupContract(address _beneficiary, uint _unlockTime) external override {
```

packages/contracts/contracts/LUSDToken.sol — View file @ 1cb68ffc

```
@@ -6,6 +6,7 @@ import "./Interfaces/ILUSDToken.sol";
 import "./Dependencies/SafeMath.sol";
 import "./Dependencies/CheckContract.sol";
 import "./Dependencies/console.sol";
+import "./Dependencies/Initializable.sol";

 /*
 *
@@ -24,12 +25,12 @@ import "./Dependencies/console.sol";
 * 2) sendToPool() and returnFromPool(): functions callable only Liquity core contracts, which move LUSD tokens between Liquity <-> user.
 */

-contract LUSDToken is CheckContract, ILUSDToken {
+contract LUSDToken is CheckContract, ILUSDToken, Initializable {
     using SafeMath for uint256;

     uint256 private _totalSupply;
```

packages/contracts/contracts/LUSDToken.sol

View file @ 1cb68ffc

```
45      -    bytes32 private immutable _CACHED_DOMAIN_SEPARATOR;
46      -    uint256 private immutable _CACHED_CHAIN_ID;
    46  +    bytes32 private _CACHED_DOMAIN_SEPARATOR;
    47  +    uint256 private _CACHED_CHAIN_ID;
47  48
48      -    bytes32 private immutable _HASHED_NAME;
49      -    bytes32 private immutable _HASHED_VERSION;
    49  +    bytes32 private _HASHED_NAME;
    50  +    bytes32 private _HASHED_VERSION;
50  51
51  52      mapping (address => uint256) private _nonces;
52  53
... ...   @@ -55,22 +56,23 @@ contract LUSDToken is CheckContract, ILUSDToken {
55  56      mapping (address => mapping (address => uint256)) private _allowances;
56  57
57  58      // --- Addresses ---
58      -    address public immutable troveManagerAddress;
59      -    address public immutable stabilityPoolAddress;
60      -    address public immutable borrowerOperationsAddress;
    59  +    address public troveManagerAddress;
    60  +    address public stabilityPoolAddress;
    61  +    address public borrowerOperationsAddress;
61  62
62  63      // --- Events ---
63  64      event TroveManagerAddressChanged(address _troveManagerAddress);
64  65      event StabilityPoolAddressChanged(address _newStabilityPoolAddress);
65  66      event BorrowerOperationsAddressChanged(address _newBorrowerOperationsAddress);
66  67
67      -    constructor
    68  +    function initialize
68  69      (
69  70          address _troveManagerAddress,
70  71          address _stabilityPoolAddress,
71  72          address _borrowerOperationsAddress
72  73      )
73      -        public
    74  +        public
    75  +        initializer
74  76      {
75  77          checkContract(_troveManagerAddress);
```

packages/contracts/contracts/MultiTroveGetter.sol

View file @ 1cb68ffc

```
... ...   @@ -5,9 +5,10 @@ pragma experimental ABIEncoderV2;
5   5
6   6   import "./TroveManager.sol";
7   7   import "./SortedTroves.sol";
    8   + import "./Dependencies/Initializable.sol";
8   9
9   10  /*  Helper contract for grabbing Trove data for the front end. Not part of the core Liquity system. */
10      - contract MultiTroveGetter {
    11  + contract MultiTroveGetter is Initializable {
11  12      struct CombinedTroveData {
12  13          address owner;
13  14
... ...   @@ -22,7 +23,7 @@ contract MultiTroveGetter {
22  23      TroveManager public troveManager; // XXX Troves missing from ITroveManager?
23  24      ISortedTroves public sortedTroves;
24  25
25      -    constructor(TroveManager _troveManager, ISortedTroves _sortedTroves) public {
    26  +    function initialize(TroveManager _troveManager, ISortedTroves _sortedTroves) public initializer {
26  27          troveManager = _troveManager;
27  28          sortedTroves = _sortedTroves;
28  29      }
... ...
```

packages/contracts/contracts/PriceFeed.sol

View file @ 1cb68ffc

```
@@ -6,7 +6,7 @@ import "./Interfaces/IPriceFeed.sol";
import "./Interfaces/ITellorCaller.sol";
import "./Dependencies/AggregatorV3Interface.sol";
import "./Dependencies/SafeMath.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
import "./Dependencies/CheckContract.sol";
import "./Dependencies/BaseMath.sol";
import "./Dependencies/LiquityMath.sol";
@@ -20,7 +20,7 @@ import "./Dependencies/console.sol";
* switching oracles based on oracle failures, timeouts, and conditions for returning to the primary
* Chainlink oracle.
*/
-contract PriceFeed is Ownable, CheckContract, BaseMath, IPriceFeed {
+contract PriceFeed is OwnableUpgradeable, CheckContract, BaseMath, IPriceFeed {
    using SafeMath for uint256;

    string constant public NAME = "PriceFeed";
@@ -83,6 +83,10 @@ contract PriceFeed is Ownable, CheckContract, BaseMath, IPriceFeed {
    event PriceFeedStatusChanged(Status newStatus);

    // --- Dependency setters ---
+
+    function initialize() public initializer {
+        __Ownable_init();
+    }

    function setAddresses(
        address _priceAggregatorAddress,
@@ -109,7 +113,7 @@ contract PriceFeed is Ownable, CheckContract, BaseMath, IPriceFeed {

        _storeChainlinkPrice(chainlinkResponse);

-        _renounceOwnership();
+        renounceOwnership();
    }
```

packages/contracts/contracts/SortedTroves.sol

View file @ 1cb68ffc

```
@@ -6,7 +6,7 @@ import "./Interfaces/ISortedTroves.sol";
import "./Interfaces/ITroveManager.sol";
import "./Interfaces/IBorrowerOperations.sol";
import "./Dependencies/SafeMath.sol";
-import "./Dependencies/Ownable.sol";
+import "./Dependencies/OwnableUpgradeable.sol";
import "./Dependencies/CheckContract.sol";
import "./Dependencies/console.sol";

@@ -43,7 +43,7 @@ import "./Dependencies/console.sol";
*
* - Public functions with parameters have been made internal to save gas, and given an external wrapper function for external access
*/
-contract SortedTroves is Ownable, CheckContract, ISortedTroves {
+contract SortedTroves is OwnableUpgradeable, CheckContract, ISortedTroves {
    using SafeMath for uint256;

    string constant public NAME = "SortedTroves";
@@ -77,6 +77,10 @@ contract SortedTroves is Ownable, CheckContract, ISortedTroves {

    // --- Dependency setters ---

+    function initialize() public initializer {
+        __Ownable_init();
+    }
+
    function setParams(uint256 _size, address _troveManagerAddress, address _borrowerOperationsAddress) external override onlyOwner {
        require(_size > 0, "SortedTroves: Size can't be zero");
        checkContract(_troveManagerAddress);
@@ -90,7 +94,7 @@ contract SortedTroves is Ownable, CheckContract, ISortedTroves {
        emit TroveManagerAddressChanged(_troveManagerAddress);
        emit BorrowerOperationsAddressChanged(_borrowerOperationsAddress);

-        _renounceOwnership();
+        renounceOwnership();
    }

    /*
```

packages/contracts/contracts/StabilityPool.sol

View file @ 1cb68ffc

```
...  ...  @@ -12,7 +12,7 @@ import "./Interfaces/ICommunityIssuance.sol";
12   12   import "./Dependencies/LiquityBase.sol";
13   13   import "./Dependencies/SafeMath.sol";
14   14   import "./Dependencies/LiquitySafeMath128.sol";
15        - import "./Dependencies/Ownable.sol";
     15   + import "./Dependencies/OwnableUpgradeable.sol";
16   16   import "./Dependencies/CheckContract.sol";
17   17   import "./Dependencies/console.sol";
18   18
...  ...  @@ -145,7 +145,7 @@ import "./Dependencies/console.sol";
145  145   * The product P (and snapshot P_t) is re-used, as the ratio P/P_t tracks a deposit's depletion due to liquidations.
146  146   *
147  147   */
148       - contract StabilityPool is LiquityBase, Ownable, CheckContract, IStabilityPool {
     148  + contract StabilityPool is LiquityBase, OwnableUpgradeable, CheckContract, IStabilityPool {
149  149       using LiquitySafeMath128 for uint128;
150  150
151  151       string constant public NAME = "StabilityPool";
...  ...  @@ -199,7 +199,7 @@ contract StabilityPool is LiquityBase, Ownable, CheckContract, IStabilityPool {
199  199       * During its lifetime, a deposit's value evolves from d_t to d_t * P / P_t , where P_t
200  200       * is the snapshot of P taken at the instant the deposit was made. 18-digit decimal.
201  201       */
202       -     uint public P = DECIMAL_PRECISION;
     202  +     uint public P;
203  203
204  204       uint public constant SCALE_FACTOR = 1e9;
205  205
...  ...  @@ -269,6 +269,11 @@ contract StabilityPool is LiquityBase, Ownable, CheckContract, IStabilityPool {
269  269
270  270       // --- Contract setters ---
271  271
     272  +     function initialize() public initializer {
     273  +         __Ownable_init();
     274  +         P = DECIMAL_PRECISION;
     275  +     }
     276  +
272  277       function setAddresses(
273  278           address _borrowerOperationsAddress,
274  279           address _troveManagerAddress,
...  ...  @@ -306,7 +311,7 @@ contract StabilityPool is LiquityBase, Ownable, CheckContract, IStabilityPool {
```

packages/contracts/contracts/StabilityPool.sol

View file @ 1cb68ffc

```
     272  +     function initialize() public initializer {
     273  +         __Ownable_init();
     274  +         P = DECIMAL_PRECISION;
     275  +     }
     276  +
272  277       function setAddresses(
273  278           address _borrowerOperationsAddress,
274  279           address _troveManagerAddress,
...  ...  @@ -306,7 +311,7 @@ contract StabilityPool is LiquityBase, Ownable, CheckContract, IStabilityPool {
306  311           emit PriceFeedAddressChanged(_priceFeedAddress);
307  312           emit CommunityIssuanceAddressChanged(_communityIssuanceAddress);
308  313
309       -         _renounceOwnership();
     314  +         renounceOwnership();
310  315       }
311  316
312  317       // --- Getters for public variables. Required by IPool interface ---
```

packages/contracts/contracts/TroveManager.sol

View file @ 1cb68ffc

```
...  ...  @@ -10,11 +10,11 @@ import "./Interfaces/ISortedTroves.sol";
10   10   import "./Interfaces/ILQTYToken.sol";
11   11   import "./Interfaces/ILQTYStaking.sol";
12   12   import "./Dependencies/LiquityBase.sol";
13        - import "./Dependencies/Ownable.sol";
     13   + import "./Dependencies/OwnableUpgradeable.sol";
14   14   import "./Dependencies/CheckContract.sol";
15   15   import "./Dependencies/console.sol";
16   16
17        - contract TroveManager is LiquityBase, Ownable, CheckContract, ITroveManager {
     17   + contract TroveManager is LiquityBase, OwnableUpgradeable, CheckContract, ITroveManager {
18   18       string constant public NAME = "TroveManager";
19   19
20   20       // --- Connected contract declarations ---
...  ...  @@ -231,6 +231,10 @@ contract TroveManager is LiquityBase, Ownable, CheckContract, ITroveManager {
231  231
232  232       // --- Dependency setter ---
233  233
     234  +    function initialize() public initializer {
     235  +        __Ownable_init();
     236  +    }
     237  +
234  238       function setAddresses(
235  239           address _borrowerOperationsAddress,
236  240           address _activePoolAddress,
...  ...  @@ -284,7 +288,7 @@ contract TroveManager is LiquityBase, Ownable, CheckContract, ITroveManager {
284  288           emit LQTYTokenAddressChanged(_lqtyTokenAddress);
285  289           emit LQTYStakingAddressChanged(_lqtyStakingAddress);
286  290
287       -        _renounceOwnership();
     291  +        renounceOwnership();
288  292       }
```

New Contracts:

packages/contracts/contracts/Dependencies/ContextUpgradeable.sol 0 → 100644

View file @ d8fdb7e5

```
1   + // SPDX-License-Identifier: MIT
2   +
3   + pragma solidity 0.6.11;
4   + import "./Initializable.sol";
5   +
6   + /*
7   +  * Based on OpenZeppelin's ContextUpgradeable contract:
8   +  * https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/v3.3.0/contracts/GSN/ContextUpgradeable.sol
9   +  *
10  +  * @dev Provides information about the current execution context, including the
11  +  * sender of the transaction and its data. While these are generally available
12  +  * via msg.sender and msg.data, they should not be accessed in such a direct
13  +  * manner, since when dealing with GSN meta-transactions the account sending and
14  +  * paying for execution may not be the actual sender (as far as an application
15  +  * is concerned).
16  +  *
17  +  * This contract is only required for intermediate, library-like contracts.
18  +  */
19  + abstract contract ContextUpgradeable is Initializable {
20  +     function __Context_init() internal initializer {
21  +         __Context_init_unchained();
22  +     }
23  +
24  +     function __Context_init_unchained() internal initializer {
25  +     }
26  +     function _msgSender() internal view virtual returns (address payable) {
27  +         return msg.sender;
28  +     }
29  +
30  +     function _msgData() internal view virtual returns (bytes memory) {
31  +         this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
32  +         return msg.data;
33  +     }
34  +     uint256[50] private __gap;
35  + }
     \ No newline at end of file
```

packages/contracts/contracts/Dependencies/Initializable.sol 0 → 100644

View file @ 1cb68ffc

```
+ // SPDX-License-Identifier: MIT
+
+ // solhint-disable-next-line compiler-version
+ pragma solidity 0.6.11;
+
+
+ /**
+  * Based on OpenZeppelin's Initializable contract:
+  * https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/v3.3.0/contracts/proxy/Initializable.sol
+  *
+  * @dev This is a base contract to aid in writing upgradeable contracts, or any kind of contract that will be deployed
+  * behind a proxy. Since a proxied contract can't have a constructor, it's common to move constructor logic to an
+  * external initializer function, usually called `initialize`. It then becomes necessary to protect this initializer
+  * function so it can only be called once. The {initializer} modifier provided by this contract will have this effect.
+  *
+  * TIP: To avoid leaving the proxy in an uninitialized state, the initializer function should be called as early as
+  * possible by providing the encoded function call as the `_data` argument to {UpgradeableProxy-constructor}.
+  *
+  * CAUTION: When used with inheritance, manual care must be taken to not invoke a parent initializer twice, or to ensure
+  * that all initializers are idempotent. This is not verified automatically as constructors are by Solidity.
+  */
+ abstract contract Initializable {
+
+     /**
+      * @dev Indicates that the contract has been initialized.
+      */
+     bool private _initialized;
+
+     /**
+      * @dev Indicates that the contract is in the process of being initialized.
+      */
+     bool private _initializing;
+
+     /**
+      * @dev Modifier to protect an initializer function from being invoked twice.
+      */
```

```
+     /**
+      * @dev Modifier to protect an initializer function from being invoked twice.
+      */
+     modifier initializer() {
+         require(_initializing || _isConstructor() || !_initialized, "Initializable: contract is already initialized");
+
+         bool isTopLevelCall = !_initializing;
+         if (isTopLevelCall) {
+             _initializing = true;
+             _initialized = true;
+         }
+
+         _;
+
+         if (isTopLevelCall) {
+             _initializing = false;
+         }
+     }
+
+     /// @dev Returns true if and only if the function is running in the constructor
+     function _isConstructor() private view returns (bool) {
+         // extcodesize checks the size of the code stored in an address, and
+         // address returns the current address. Since the code is still not
+         // deployed when running a constructor, any checks on its code size will
+         // yield zero, making it an effective way to detect if a contract is
+         // under construction or not.
+         address self = address(this);
+         uint256 cs;
+         // solhint-disable-next-line no-inline-assembly
+         assembly { cs := extcodesize(self) }
+         return cs == 0;
+     }
+ }
\ No newline at end of file
```

packages/contracts/contracts/Dependencies/OwnableUpgradeable.sol 0 → 100644 View file @ 1cb68ffc

```
+ // SPDX-License-Identifier: MIT
+
+ pragma solidity 0.6.11;
+
+ import "./ContextUpgradeable.sol";
+ import "./Initializable.sol";
+ /**
+  * Based on OpenZeppelin's Ownable contract:
+  * https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/v3.3.0/contracts/access/OwnableUpgradeable.sol
+  *
+  * @dev Contract module which provides a basic access control mechanism, where
+  * there is an account (an owner) that can be granted exclusive access to
+  * specific functions.
+  *
+  * By default, the owner account will be the one that deploys the contract. This
+  * can later be changed with {transferOwnership}.
+  *
+  * This module is used through inheritance. It will make available the modifier
+  * `onlyOwner`, which can be applied to your functions to restrict their use to
+  * the owner.
+  */
+ abstract contract OwnableUpgradeable is Initializable, ContextUpgradeable {
+     address private _owner;
+
+     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
+
+     /**
+      * @dev Initializes the contract setting the deployer as the initial owner.
+      */
+     function __Ownable_init() internal initializer {
+         __Context_init_unchained();
+         __Ownable_init_unchained();
+     }
+
+     function __Ownable_init_unchained() internal initializer {
+         address msgSender = _msgSender();
+         _owner = msgSender;
+         emit OwnershipTransferred(address(0), msgSender);
+     }
+
```

packages/contracts/contracts/Dependencies/OwnableUpgradeable.sol 0 → 100644 View file @ 1cb68ffc

```
+
+     /**
+      * @dev Returns the address of the current owner.
+      */
+     function owner() public view returns (address) {
+         return _owner;
+     }
+
+     /**
+      * @dev Throws if called by any account other than the owner.
+      */
+     modifier onlyOwner() {
+         require(_owner == _msgSender(), "Ownable: caller is not the owner");
+         _;
+     }
+
+     /**
+      * @dev Leaves the contract without owner. It will not be possible to call
+      * `onlyOwner` functions anymore. Can only be called by the current owner.
+      *
+      * NOTE: Renouncing ownership will leave the contract without an owner,
+      * thereby removing any functionality that is only available to the owner.
+      */
+     function renounceOwnership() public virtual onlyOwner {
+         emit OwnershipTransferred(_owner, address(0));
+         _owner = address(0);
+     }
+
+     /**
+      * @dev Transfers ownership of the contract to a new account (`newOwner`).
+      * Can only be called by the current owner.
+      */
+     function transferOwnership(address newOwner) public virtual onlyOwner {
+         require(newOwner != address(0), "Ownable: new owner is the zero address");
+         emit OwnershipTransferred(_owner, newOwner);
+         _owner = newOwner;
+     }
+     uint256[49] private __gap;
+ }
\ No newline at end of file
```

2. Modified `_NAME` (LUSD Stablecoin -> PUSD Stablecoin) and `_SYMBOL` (LUSD -> PUSD) in LUSDToken.sol.

packages/contracts/contracts/LUSDToken.sol — View file @ 1cb68ffc

```
29  30      using SafeMath for uint256;
30  31
31  32      uint256 private _totalSupply;
32      -   string constant internal _NAME = "LUSD Stablecoin";
33      -   string constant internal _SYMBOL = "LUSD";
    32  +   string constant internal _NAME = "PUSD Stablecoin";
    33  +   string constant internal _SYMBOL = "PUSD";
33  34      string constant internal _VERSION = "1";
34  35      uint8 constant internal _DECIMALS = 18;
35  36
```

3. Modified `_NAME` (LQTY -> PIGGY) and `_SYMBOL` (LQTY -> PIGGY) in LQTYToken.sol

packages/contracts/contracts/LQTY/LQTYToken.sol — View file @ 1cb68ffc

```
54  61
55      -   string constant internal _NAME = "LQTY";
56      -   string constant internal _SYMBOL = "LQTY";
    62  +   string constant internal _NAME = "PIGGY";
    63  +   string constant internal _SYMBOL = "PIGGY";
57  64      string constant internal _VERSION = "1";
58  65      uint8 constant internal  _DECIMALS = 18;
59  66
... ... @@ -70,30 +77,32 @@ contract LQTYToken is CheckContract, ILQTYToken {
70  77
```

4. Modified `LUSD_GAS_COMPENSATION` from 200 to 20 in LiquityBase.sol, lowered preserved amount to 20.

Modified `MIN_NET_DEBT` from 1800 to 180 in LiquityBase.sol, lowered net debt to 180. Borrowers need to maintain their total debt to at least 180 + 20 =200

packages/contracts/contracts/Dependencies/LiquityBase.sol — View file @ d8fdb7e5

```
... ... @@ -25,10 +25,11 @@ contract LiquityBase is BaseMath, ILiquityBase {
25  25      uint constant public CCR = 150000000000000000; // 150%
26  26
27  27      // Amount of LUSD to be locked in gas pool on opening troves
28      -   uint constant public LUSD_GAS_COMPENSATION = 200e18;
    28  +   uint constant public LUSD_GAS_COMPENSATION = 20e18;
29  29
30  30      // Minimum amount of net LUSD debt a trove must have
31      -   uint constant public MIN_NET_DEBT = 1800e18;
    31  +   uint constant public MIN_NET_DEBT = 180e18;
    32  +
32  33      // uint constant public MIN_NET_DEBT = 0;
33  34
34  35      uint constant public PERCENT_DIVISOR = 200; // dividing by 200 yields 0.5%
```

5. Adjust LQTY allocations

Liquity:

- 2 million unlocked tokens are minted at deployment to bug bounties / hackathons
- Stability Pool preserves 32 million rewards which follow a yearly halving schedule
- LUSD/ETH LP mining reward pool (1 + 1/3) million, released in 6 weeks
- multisign address preserves (64 + 2/3) million which will be locked for a year

Piggy:

- 10 million tokens are minted to contributor mining address
- 3 million goes to Liquity treasury multisig address
- 2 million goes to Liquity airdrop multisig address
- Stability Pool preseves 47.5 million rewards which follow a yearly halving schedule
- 7.5 million goes to LP mining reward pool

- 10 million tokens are minted to investor multisig address, and will be locked for half year
- 20 million tokens are minted to team multisig address, and will be locked for half year

packages/contracts/contracts/LQTY/CommunityIssuance.sol

View file @ 1cb68ffc

```
42  42      *
43      -   * Set to 32M (slightly less than 1/3) of total LQTY supply.
    43  +   * Set to 47.5M of total LQTY supply.
44  44      */
45      -   uint constant public LQTYSupplyCap = 32e24; // 32 million
    45  +   uint public LQTYSupplyCap; // 47.5 million
46  46
47  47      ILQTYToken public lqtyToken;
48  48
49  49      address public stabilityPoolAddress;
50  50
51  51      uint public totalLQTYIssued;
52      -   uint public immutable deploymentTime;
    52  +   uint public deploymentTime;
53  53
54  54      // --- Events ---
55  55
... ... @@ -59,7 +59,8 @@ contract CommunityIssuance is ICommunityIssuance, Ownable, CheckContract, BaseMa
59  59
60  60      // --- Functions ---
61  61
62      -   constructor() public {
    62  +   function initialize() public initializer {
    63  +       __Ownable_init();
63  64          deploymentTime = block.timestamp;
64  65      }
65  66
... ... @@ -79,13 +80,15 @@ contract CommunityIssuance is ICommunityIssuance, Ownable, CheckContract, BaseMa
79  80          stabilityPoolAddress = _stabilityPoolAddress;
80  81
81  82          // When LQTYToken deployed, it should have transferred CommunityIssuance's LQTY entitlement
    83  +       LQTYSupplyCap = lqtyToken.getCommunityIssuanceEntitlement();
    84  +       assert(LQTYSupplyCap > 0);
82  85          uint LQTYBalance = lqtyToken.balanceOf(address(this));
83      -       assert(LQTYBalance >= LQTYSupplyCap);
    86  +       assert(LQTYBalance == LQTYSupplyCap);
84  87
85  88          emit LQTYTokenAddressSet(_lqtyTokenAddress);
86  89          emit StabilityPoolAddressSet(_stabilityPoolAddress);
87  90
```

packages/contracts/contracts/Interfaces/ILQTYToken.sol

View file @ 1cb68ffc

```
... ... @@ -20,4 +20,6 @@ interface ILQTYToken is IERC20, IERC2612 {
20  20      function getDeploymentStartTime() external view returns (uint256);
21  21
22  22      function getLpRewardsEntitlement() external view returns (uint256);
    23  +
    24  +   function getCommunityIssuanceEntitlement() external view returns (uint256);
23  25  }
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
@@ -135,23 +149,21 @@ contract LQTYToken is CheckContract, ILQTYToken {

        // --- Initial LQTY allocations ---

-       uint bountyEntitlement = _1_MILLION.mul(2); // Allocate 2 million for bounties/hackathons
-       _mint(_bountyAddress, bountyEntitlement);
+       _mint(_contributorMiningAddress, _1_MILLION.mul(10)); // Allocate 10 million for contributor mining

-       uint depositorsAndFrontEndsEntitlement = _1_MILLION.mul(32); // Allocate 32 million to the algorithmic issuance schedule
-       _mint(_communityIssuanceAddress, depositorsAndFrontEndsEntitlement);
+       communityIssuanceEntitlement = _1_MILLION.mul(475).div(10); // Allocate 47.5 million to the algorithmic issuance schedule
+       _mint(_communityIssuanceAddress, communityIssuanceEntitlement);

-       uint _lpRewardsEntitlement = _1_MILLION.mul(4).div(3);  // Allocate 1.33 million for LP rewards
-       lpRewardsEntitlement = _lpRewardsEntitlement;
-       _mint(_lpRewardsAddress, _lpRewardsEntitlement);
-
-       // Allocate the remainder to the LQTY Multisig: (100 - 2 - 32 - 1.33) million = 64.66 million
-       uint multisigEntitlement = _1_MILLION.mul(100)
-           .sub(bountyEntitlement)
-           .sub(depositorsAndFrontEndsEntitlement)
-           .sub(_lpRewardsEntitlement);
+       lpRewardsEntitlement = _1_MILLION.mul(75).div(10); // Allocate 7.5 million for LP rewards
+       _mint(_lpRewardsAddress, lpRewardsEntitlement);

-       _mint(_multisigAddress, multisigEntitlement);
+       _mint(_treasuryAddress, _1_MILLION.mul(3)); // Allocate 3 million for treasury
+
+       _mint(_airdropAddress, _1_MILLION.mul(2)); // Allocate 2 million for airdrop
+
+       _mint(_investorAddress, _1_MILLION.mul(10)); // Allocate 10 million for investor
+
+       _mint(_teamAddress, _1_MILLION.mul(20)); // Allocate the remainder 20 million to the team
    }

    // --- External functions ---
@@ -172,9 +184,13 @@ contract LQTYToken is CheckContract, ILQTYToken {
        return lpRewardsEntitlement;
    }
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
@@ -187,6 +206,14 @@ contract LQTYToken is CheckContract, ILQTYToken {
+   function getCommunityIssuanceEntitlement() external view override returns (uint256) {
+       return communityIssuanceEntitlement;
+   }
+
    function transfer(address recipient, uint256 amount) external override returns (bool) {
-       // Restrict the multisig's transfers in first year
-       if (_callerIsMultisig() && _isFirstYear()) {
+       // Restrict the multisig's transfers in first half year
+       if (_callerHasLockPeriod() && _isHalfYear()) {
            _requireRecipientIsRegisteredLC(recipient);
        }

@@ -190,14 +206,14 @@ contract LQTYToken is CheckContract, ILQTYToken {
    }

    function approve(address spender, uint256 amount) external override returns (bool) {
-       if (_isFirstYear()) { _requireCallerIsNotMultisig(); }
+       if (_isHalfYear()) { _requireCallerHasNoLockPeriod(); }

        _approve(msg.sender, spender, amount);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) external override returns (bool) {
-       if (_isFirstYear()) { _requireSenderIsNotMultisig(sender); }
+       if (_isHalfYear()) { _requireSenderHasNoLockPeriod(sender); }

        _requireValidRecipient(recipient);

@@ -207,14 +223,14 @@ contract LQTYToken is CheckContract, ILQTYToken {
    }

    function increaseAllowance(address spender, uint256 addedValue) external override returns (bool) {
-       if (_isFirstYear()) { _requireCallerIsNotMultisig(); }
+       if (_isHalfYear()) { _requireCallerHasNoLockPeriod(); }

        _approve(msg.sender, spender, _allowances[msg.sender][spender].add(addedValue));
        return true;
    }
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
216 232     function decreaseAllowance(address spender, uint256 subtractedValue) external override returns (bool) {
217     -       if (_isFirstYear()) { _requireCallerIsNotMultisig(); }
    233 +       if (_isHalfYear()) { _requireCallerHasNoLockPeriod(); }
218 234
219 235           _approve(msg.sender, spender, _allowances[msg.sender][spender].sub(subtractedValue, "ERC20: decreased allowance below zero"));
220 236           return true;
... ...   @@ -222,7 +238,7 @@ contract LQTYToken is CheckContract, ILQTYToken {
222 238
223 239       function sendToLQTYStaking(address _sender, uint256 _amount) external override {
224 240           _requireCallerIsLQTYStaking();
225     -       if (_isFirstYear()) { _requireSenderIsNotMultisig(_sender); }  // Prevent the multisig from staking LQTY
    241 +       if (_isHalfYear()) { _requireSenderHasNoLockPeriod(_sender); }  // Prevent the team & investor from staking LQTY
226 242           _transfer(_sender, lqtyStakingAddress, _amount);
227 243       }
228 244
... ...   @@ -302,12 +318,12 @@ contract LQTYToken is CheckContract, ILQTYToken {
302 318
303 319       // --- Helper functions ---
304 320
305     -   function _callerIsMultisig() internal view returns (bool) {
306     -       return (msg.sender == multisigAddress);
    321 +   function _callerHasLockPeriod() internal view returns (bool) {
    322 +       return (msg.sender == teamAddress || msg.sender == investorAddress);
307 323       }
308 324
309     -   function _isFirstYear() internal view returns (bool) {
310     -       return (block.timestamp.sub(deploymentStartTime) < ONE_YEAR_IN_SECONDS);
    325 +   function _isHalfYear() internal view returns (bool) {
    326 +       return (block.timestamp.sub(deploymentStartTime) < HALF_YEAR_IN_SECONDS);
311 327       }
312 328
313 329       // --- 'require' functions ---
... ...   @@ -330,12 +346,12 @@ contract LQTYToken is CheckContract, ILQTYToken {
330 346           "LQTYToken: recipient must be a LockupContract registered in the Factory");
331 347       }
332 348
333     -   function _requireSenderIsNotMultisig(address _sender) internal view {
334     -       require(_sender != multisigAddress, "LQTYToken: sender must not be the multisig");
```

packages/contracts/contracts/LQTY/LQTYToken.sol

View file @ 1cb68ffc

```
332 348
333     -   function _requireSenderIsNotMultisig(address _sender) internal view {
334     -       require(_sender != multisigAddress, "LQTYToken: sender must not be the multisig");
    349 +   function _requireSenderHasNoLockPeriod(address _sender) internal view {
    350 +       require(_sender != teamAddress && _sender != investorAddress, "LQTYToken: sender must not be the multisig");
335 351       }
336 352
337     -   function _requireCallerIsNotMultisig() internal view {
338     -       require(!_callerIsMultisig(), "LQTYToken: caller must not be the multisig");
    353 +   function _requireCallerHasNoLockPeriod() internal view {
    354 +       require(!_callerHasLockPeriod(), "LQTYToken: caller must not be the multisig");
339 355       }
340 356
341 357       function _requireCallerIsLQTYStaking() internal view {
... ...
```

packages/contracts/contracts/LQTY/LockupContract.sol — View file @ 1cb68ffc

```
@@ -6,14 +6,14 @@ import "../Dependencies/SafeMath.sol";
 import "../Interfaces/ILQTYToken.sol";

 /*
- * The lockup contract architecture utilizes a single LockupContract, with an unlockTime. The unlockTime is passed as an argument
- * to the LockupContract's constructor. The contract's balance can be withdrawn by the beneficiary when block.timestamp > unlockTime.
- * At construction, the contract checks that unlockTime is at least one year later than the Liquity system's deployment time.
+ * The lockup contract architecture utilizes a single LockupContract, with an unlockTime. The unlockTime is passed as an argument
+ * to the LockupContract's constructor. The contract's balance can be withdrawn by the beneficiary when block.timestamp > unlockTime.
+ * At construction, the contract checks that unlockTime is at least half year later than the Liquity system's deployment time.

- * Within the first year from deployment, the deployer of the LQTYToken (Liquity AG's address) may transfer LQTY only to valid
+ * Within the half year from deployment, the deployer of the LQTYToken (Liquity AG's address) may transfer LQTY only to valid
  * LockupContracts, and no other addresses (this is enforced in LQTYToken.sol's transfer() function).
  *
- * The above two restrictions ensure that until one year after system deployment, LQTY tokens originating from Liquity AG cannot
+ * The above two restrictions ensure that until half year after system deployment, LQTY tokens originating from Liquity AG cannot
  * enter circulating supply and cannot be staked to earn system revenue.
  */
 contract LockupContract {
@@ -22,7 +22,7 @@ contract LockupContract {
     // --- Data ---
     string constant public NAME = "LockupContract";

-    uint constant public SECONDS_IN_ONE_YEAR = 31536000;
+    uint constant public SECONDS_IN_HALF_YEAR = 15552000;

     address public immutable beneficiary;

@@ -38,23 +38,23 @@ contract LockupContract {

     // --- Functions ---

-    constructor
+    constructor
     (
-        address _lqtyTokenAddress,
-        address _beneficiary,
```

packages/contracts/contracts/LQTY/LockupContract.sol — View file @ 1cb68ffc

```
+    constructor
     (
-        address _lqtyTokenAddress,
-        address _beneficiary,
+        address _lqtyTokenAddress,
+        address _beneficiary,
         uint _unlockTime
     )
-        public
+        public
     {
         lqtyToken = ILQTYToken(_lqtyTokenAddress);

         /*
-        * Set the unlock time to a chosen instant in the future, as long as it is at least 1 year after
-        * the system was deployed
+        * Set the unlock time to a chosen instant in the future, as long as it is at least half year after
+        * the system was deployed
         */
-        _requireUnlockTimeIsAtLeastOneYearAfterSystemDeployment(_unlockTime);
+        _requireUnlockTimeIsAtLeastHalfYearAfterSystemDeployment(_unlockTime);
         unlockTime = _unlockTime;
-
+

         beneficiary =  _beneficiary;
         emit LockupContractCreated(_beneficiary, _unlockTime);
     }
@@ -79,8 +79,8 @@ contract LockupContract {
         require(block.timestamp >= unlockTime, "LockupContract: The lockup duration must have passed");
     }

-    function _requireUnlockTimeIsAtLeastOneYearAfterSystemDeployment(uint _unlockTime) internal view {
+    function _requireUnlockTimeIsAtLeastHalfYearAfterSystemDeployment(uint _unlockTime) internal view {
         uint systemDeploymentTime = lqtyToken.getDeploymentStartTime();
-        require(_unlockTime >= systemDeploymentTime.add(SECONDS_IN_ONE_YEAR), "LockupContract: unlock time must be at least one year after system deployment");
+        require(_unlockTime >= systemDeploymentTime.add(SECONDS_IN_HALF_YEAR), "LockupContract: unlock time must be at least half year after system deployment");
     }
 }
```

6. Disable Tellor since it's not availble on BSC

packages/contracts/contracts/PriceFeed.sol

View file @ 1cb68ffc

```
114 118
115 119     // --- Functions ---
... ... @@ -130,7 +134,7 @@ contract PriceFeed is Ownable, CheckContract, BaseMath, IPriceFeed {
130 134         // Get current and previous price data from Chainlink, and current price data from Tellor
131 135         ChainlinkResponse memory chainlinkResponse = _getCurrentChainlinkResponse();
132 136         ChainlinkResponse memory prevChainlinkResponse = _getPrevChainlinkResponse(chainlinkResponse.roundId, chainlinkResponse.decimals);
133     -       TellorResponse memory tellorResponse = _getCurrentTellorResponse();
    137 +       TellorResponse memory tellorResponse;
134 138
135 139         // --- CASE 1: System fetched last price from Chainlink  ---
136 140         if (status == Status.chainlinkWorking) {
... ...
```

7. Fix a bug which has been reported through Bug Bounty program and accepted by Liquity team

packages/contracts/contracts/TroveManager.sol

View file @ 1cb68ffc

```
... ... @@ -719,7 +723,7 @@ contract TroveManager is LiquityBase, Ownable, CheckContract, ITroveManager {
719 723                 // Update aggregate trackers
720 724                 vars.remainingLUSDInStabPool = vars.remainingLUSDInStabPool.sub(singleLiquidation.debtToOffset);
721 725                 vars.entireSystemDebt = vars.entireSystemDebt.sub(singleLiquidation.debtToOffset);
722     -               vars.entireSystemColl = vars.entireSystemColl.sub(singleLiquidation.collToSendToSP);
    726 +               vars.entireSystemColl = vars.entireSystemColl.sub(singleLiquidation.collToSendToSP).sub(singleLiquidation.collSurplus);
723 727
724 728                 // Add liquidation values to their respective running totals
725 729                 totals = _addLiquidationValuesToTotals(totals, singleLiquidation);
... ...
```