Dapper Dino contracts deployment steps :-

1. Deploy **RewardsPool** contract and it doesn't take any arguments.

2. Then deploy the **UtilityManager** contract. It also doesn’t take any arguments.

3. Then deploy **FossilToken** contract by passing the following arguments in the constructor :

```
    constructor(
        string memory _name,
        string memory _symbol,
        uint256 _initialSupply,
        address _utilityManager
    )
```

Arguments are :

“Fossil”, “FOS”, 1000000000000000000000000000, utilityManagerContractAddress

NOTE : take utilityManagerContract address from point 2 deployment.

4. Then deploy the **DinoToken** contract.

```
constructor(string memory baseURI) ERC721("Dapper Dinos", "DINO")  { }
```

Arguments are :

“https://nft-url/token/”

NOTE : It takes an argument as the url of the NFT.

5. Then deploy the **DinoPool** contract.

```
constructor(
        string memory _name,
        string memory _symbol,
        address _depositToken,
        address _rewardToken,
        uint256 _maxBonus,
        uint256 _maxLockDuration,
        address _utilityManager,
        address _rewardsPool,
 address _dinoToken
        )  TimeLockPool(_name,  _symbol,  _depositToken,  _rewardToken,
_maxBonus, _maxLockDuration, _utilityManager, _rewardsPool, _dinoToken
) { }
```

Arguments are :

“Staked Dino Token”, “SDT”, dinoTokenContractAddress, fossilTokenContractAddress, 1000000000000000000, 600, utilityManagerContractAddress, rewardsPoolContractAddress, dinoTokenContractAddress

NOTE :

a. Take dinoTokenContractAddressaddress from point 4 deployment.
b. Take fossilTokenContractAddress from point 3 deployment.
c. 600 is the maximum staking time period in seconds (10 minutes).
d. Take utilityManagerContractAddress from point 2 deployment.
e. Take rewardsPoolContractAddress from point 1 deployment.
f. Take rewardsPoolContractAddress from point 4 deployment.

6. Then deploy the **LiquidityMiningManager** contract.

```
        constructor(address  _reward,  address  _rewardSource,  address
_utilityManager) { }
```

Arguments are :

fossilTokenContractAddress, deployerWalletAddress, utilityManagerContractAddress

NOTE :

a. Take fossilTokenContractAddress from point 3 deployment.
b. In _rewardSource, pass your wallet address as you are the deployer and will hold rewards in your wallet.
c. Take utilityManagerContractAddress from point 2 deployment.

===

Till here deployments of all contracts are done, but need to call some function to initialise values. So to do that follow the below instructions and call the method in the same order as mentioned below :

1. Call **setContractAddresses**( ) method of **RewardsPool** contract.

```
         function   setContractAddresses(address   _dinoPool,   address
_rewardToken)
```

Arguments are :
dinoPoolContractAddress, fossilTokenContractAddress

2. Call **setContractAddress**( ) method of **UtilityManager** contract.

```
function setContractAddress( address _fossilToken, address _dinoPool,
     address _liquidityMiningManager
   )
```

Arguments are :
fossilTokenContractAddress, dinoPoolContractAddress, liquidityMiningManagerContractAddress

3. Call **updatePendingRewards**( ) method of **UtilityManager** contract.

```
function updatePendingRewards(uint256 _amount, uint256 _operation)
```

Arguments are :
totalSuppy of FoslToken, 1

NOTE :
a. To get totalSupply of FossilToken, call the **totalSupply**( ) method of **FossilToken** contract.
b. And 1 means to add the totalSupply in pending rewards.

4. Call **approve**( ) method of **FossilToken** contract.

```
function approve(address spender, uint256 amount)
```

Arguments are :
liquidityMiningManagerContractAddress, totalSupply of FossilToken

NOTE :
a. To get totalSupply of FossilToken, call the **totalSupply**( ) method of **FossilToken** contract.

5. Call **grantRole**( ) method of **LiquidityMiningManager** contract.

```
function grantRole(bytes32 role, address account)
```

Arguments are :
GOV_ROLE (bytes32) , userWalletAddress

NOTE :
a. To get **GOV_ROLE** bytes32 data, call **GOV_ROLE**( ) method of **LiquidityMiningManager** contract.

6. Call **grantRole**( ) method of **LiquidityMiningManager** contract.

```
function grantRole(bytes32 role, address account)
```

Arguments are :
REWARD_DISTRIBUTOR_ROLE(bytes32) , userWalletAddress

NOTE :
a. To get **REWARD_DISTRIBUTOR_ROLE** bytes32 data, call **REWARD_DISTRIBUTOR_ROLE**( ) method of **LiquidityMiningManager** contract.

7. Call **addPool**( ) method of **LiquidityMiningManager** contract

```
function addPool(address _poolContract, uint256 _weight).
```

Arguments are :
dinoPoolContractAddress, 100

NOTE :
a. Weight of the pool is here 100%.

7. Call **setRewardPerSecond**( ) method of **LiquidityMiningManager** contract.

```
function setRewardPerSecond(uint256 _rewardPerSecond)
```

Arguments are :
5000000000000000

NOTE :
a. Here reward per second is set to 5000000000000000 wei.

==========XX Contract deployment & initialization ends here XX============