

# day88-sol-truffle-petshop

≡ 태그	
📅 날짜	@2023년 2월 1일

폴더 생성

- mkdir pet-shop
- cd pet-shop
- truffle unbox pet-shop

트러플 박스 - 기본예제들

unbox - 예제들을 불러오는 명령어

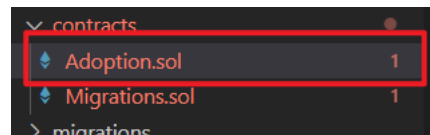
```
✓ Preparing to download
✓ Downloading
✓ Cleaning up temporary files
- Setting up boxnpm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile
npm WARN deprecated set-value@2.0.0: Critical bug fixed in v3.0.1, please upgrade to the latest version.
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated set-value@0.4.3: Critical bug fixed in v3.0.1, please upgrade to the latest version.
npm WARN deprecated mixin-deep@1.3.1: Critical bug fixed in v2.0.1, please upgrade to the latest version.
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated source-map-url@0.4.0: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated chokidar@2.0.4: Chokidar 2 does not receive security updates since 2019. Upgrade to chokidar 3 with
15x fewer dependencies
npm WARN deprecated source-map-resolve@0.5.2: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated axios@0.17.1: Critical security vulnerability fixed in v0.21.1. For more information, see https://gi
thub.com/axios/axios/pull/3410
✓ Setting up box

Unbox successful. Sweet!

Commands:
  Compile:      truffle compile
  Migrate:      truffle migrate
  Test contracts: truffle test
  Run dev server: npm run dev

PS C:\#TRUFFLE\pet-shop>
```

스마트계약 작성을 위해 contract 폴더로 이동



파일 생성

```
pragma solidity ^0.5.0;

contract Adoption{

    // Solidity 에는 주소라는 고유 유형이 있다.
    // 주소는 20바이트 값으로 저장되는 이더리움 주소이다.
    address[16] public adopters;

    //반려동물 입양 기능함수
    function adopt(uint petId) public returns(uint){
```

```

//우리 배열 petId의 범위에 있는지 확인 하고 있다.
// adoptersSolidity의 배열은 0부터 인덱싱되므로 id값은 0에서 15사이여야한다.
//이 require()명령문을 사용하여 id가 범위 내에 있는지 확인한다.
require(petId >=0 && petId <=15);
//id가 범위 내에 있으면 호출한 주소를 adopters배열에 추가한다.
//이 기능을 호출한 사람 또는 스마트 컨트랙트의 주소로 표시된다.
//msg.sender

adopters[petId] = msg.sender;
//마지막으로 petID제공된 것을 확인으로 반환.
return petId;
}

// adopters가 이미 선언되었으므로 간단히 반환할 수 있다.
// 반환 유형이(이 경우의 유형 adopters)로 지정해야한다.
//address[16] memory. memory변수의 데이터 위치를 제공한다.
function getAdopters() public view returns (address[16] memory){
    // 함수 선언의 view 키워드는 함수가 계약 상태를 수정하지 않음을 의미한다.
    return adopters;
}
}

```

- truffle compile

```

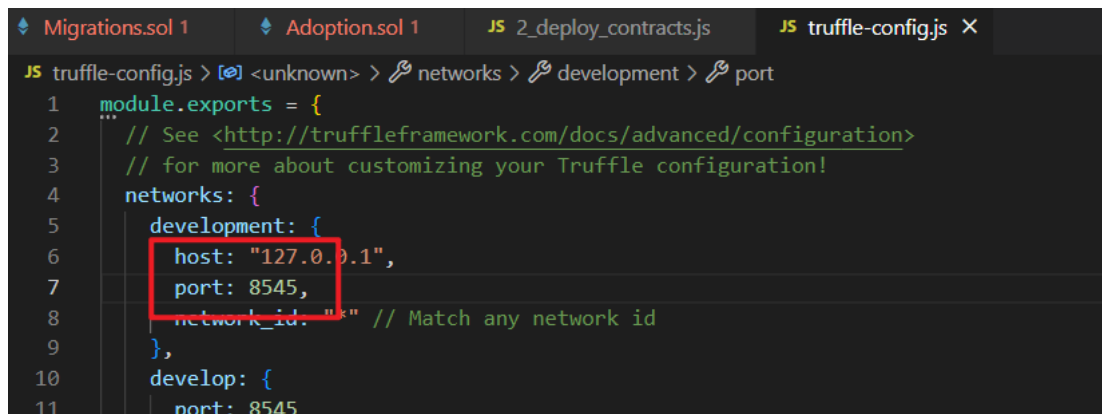
=====
> Compiling .\contracts\Adoption.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\TRUFFLE\pet-shop\build\contracts
> Compiled successfully using:
    - solc: 0.5.12+commit.7709ece9.Emscripten.clang

C:\TRUFFLE\pet-shop>

```

- truffle migrate
- truffle migrate --network development

에러 발생



```

JS truffle-config.js > [?] <unknown> > networks > development > port
1  module.exports = {
2    ...
3    // See <http://truffleframework.com/docs/advanced/configuration>
4    // for more about customizing your Truffle configuration!
5    networks: {
6      development: {
7        host: "127.0.0.1",
8        port: 8545,
9        network_id: "*" // Match any network id
10     },
11     develop: {
12       port: 8545
13     }
14   }
15 }

```

포트번호로 8545로 변경

- truffle migrate --network development

```

1_initial_migration.js
=====

Deploying 'Migrations'
-----
> transaction hash: 0xc18936132ca741cf92cc3d50a0dc5c275c750aa12db6890f6b21fc50bc48e5d1
> Blocks: 0 Seconds: 0
> contract address: 0xcED6877fb970426F2b86d2Fa425488CF13B91032
> block number: 4
> block timestamp: 1675216899
> account: 0x0Ca6d82dB9A10337156fD3b8729D9ABece114760
> balance: 99.97676482
> gas used: 191943
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00383886 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.00383886 ETH

2_deploy_contracts.js
=====

Deploying 'Adoption'
-----
> transaction hash: 0xc4b77a4f8b3e8c5ac8d1c57587af1c8b5c52ebd662eba907f507908da1c30d58
> Blocks: 0 Seconds: 0
> contract address: 0x21AF0E8652dDF293b42F50b977a81091B31cAF3e
> block number: 6
> block timestamp: 1675216900
> account: 0x0Ca6d82dB9A10337156fD3b8729D9ABece114760
> balance: 99.97184152

```

```

2_deploy_contracts.js
=====

Deploying 'Adoption'
-----
> transaction hash: 0xc4b77a4f8b3e8c5ac8d1c57587af1c8b5c52ebd662eba907f507908da1c30d58
> Blocks: 0 Seconds: 0
> contract address: 0x21AF0E8652dDF293b42F50b977a81091B31cAF3e
> block number: 6
> block timestamp: 1675216900
> account: 0x0Ca6d82dB9A10337156fD3b8729D9ABece114760
> balance: 99.97184152
> gas used: 203827
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00407654 ETH

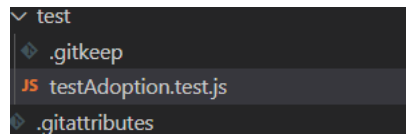
> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.00407654 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0.0079154 ETH

```

C:\TRUFFLE\pet-shop>

컨트랙트 주소 0x21AF0E8652dDF293b42F50b977a81091B31cAF3e



테스트 파일 생성

testAdoption.test.js

```
const Adoption = artifacts.require("Adoption");

contract("Adoption", (accounts) => {
  let adoption;
  let expectedPetId;

  before(async() => {
    adoption = await Adoption.deployed();
  });

  describe("adopting a pet and retrieving account addresses", async() => {
    before("adopt a pet using accounts[0]", async() => {
      await adoption.adopt(8, {from: accounts[0]});
      expectedAdopter = accounts[0];
    })
    it("can fetch the address of an owner by pet id", async() => {
      const adopter = await adoption.adopters(8);
      assert.equal(adopter, expectedAdopter, "The owner of the adopted pet should be the first account.");
    });
  })
})
```

/src/js/app.js

```
initWeb3: async function() {
  /*
   * 웹페이지 열리고 web3 설정
   */
  if(window.ethereum) {
    App.web3Provider = window.ethereum;
    try {
      await window.ethereum.enable();
    } catch (error) {
      console.error("User denied account access");
    }
  }

  return App.initContract();
},
```

app.js

web3 부분 변경

```
App = {
  web3Provider: null,
  contracts: {},

  init: async function() {
    // Load pets.
    $.getJSON('../pets.json', function(data) {
      var petsRow = $('#petsRow');
      var petTemplate = $('#petTemplate');
```

```

        for (i = 0; i < data.length; i++) {
            petTemplate.find('.panel-title').text(data[i].name);
            petTemplate.find('img').attr('src', data[i].picture);
            petTemplate.find('.pet-breed').text(data[i].breed);
            petTemplate.find('.pet-age').text(data[i].age);
            petTemplate.find('.pet-location').text(data[i].location);
            petTemplate.find('.btn-adopt').attr('data-id', data[i].id);

            petsRow.append(petTemplate.html());
        }
    });

    return await App.initWeb3();
},

initWeb3: async function() {
    /*
     * 웹페이지 열고 web3설정
     */

    if(window.ethereum) {
        App.web3Provider = window.ethereum;
        try{
            await window.ethereum.enable();
        }catch(error){
            console.error("User denied account access");
        }
    }

    return App.initContract();
},

initContract: function() {
    /*
     * Replace me...
     */

    return App.bindEvents();
},

bindEvents: function() {
    $(document).on('click', '.btn-adopt', App.handleAdopt);
},

markAdopted: function() {
    /*
     * Replace me...
     */
},

handleAdopt: function(event) {
    event.preventDefault();

    var petId = parseInt($(event.target).data('id'));

    /*
     * Replace me...
     */
}

};

$(function() {
    $(window).load(function() {
        App.init();
    });
});

```

## app.js

```

handleAdopt: function(event) {
    event.preventDefault();

    var petId = parseInt($(event.target).data('id'));

    /*
     web3를 사용하여 사용자 계정을 가져옵니다. 오류 확인 후 콜백에서 첫 번째 계정을 선택합니다.
     여기에서 위에서 수행한 대로 배포된 계약을 가져오고 인스턴스를 에 저장합니다 adoptionInstance.
     하지만 이번에는 호출 대신 트랜잭션 을 보낼 것 입니다.
     트랜잭션에는 "보낸 사람" 주소가 필요하며 관련 비용이 있습니다.
     에테르로 지불되는 이 비용을 가스 라고 합니다.
     가스 비용은 계산을 수행하거나 스마트 계약에 데이터를 저장하는 데 드는 비용입니다.
    */
}

```

이전 adopt()에 account 트랜잭션을 보낸 결과는 트랜잭션 개체입니다.  
오류가 없으면 markAdopted()함수를 호출하여 새로 저장된 데이터와 UI를 동기화합니다.

```
*/

var adoptionInstance;

web3.eth.getAccounts(function(error, accounts) {
  if (error) {
    console.log(error);
  }

  var account = accounts[0];

  App.contracts.Adoption.deployed().then(function(instance) {
    adoptionInstance = instance;

    // Execute adopt as a transaction by sending account
    return adoptionInstance.adopt(petId, {from: account});
  }).then(function(result) {
    return App.markAdopted();
  }).catch(function(err) {
    console.log(err.message);
  });
});








};
```

ACCOUNTS
BLOCKS
TRANSACTIONS
CONTRACTS
EVENTS
LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 12
GAS PRICE 20000000000
GAS LIMIT 6721975
HARDFORK MUIRGLACIER
NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:8545
MINING STATUS AUTOMINING
WORKSPACE QUICKSTART
SAVE SWITCH

MNEMONIC ? buddy able record dry account diary dress man light filter morning north
HD PATH m/44'/60'/0'/0/account\_index

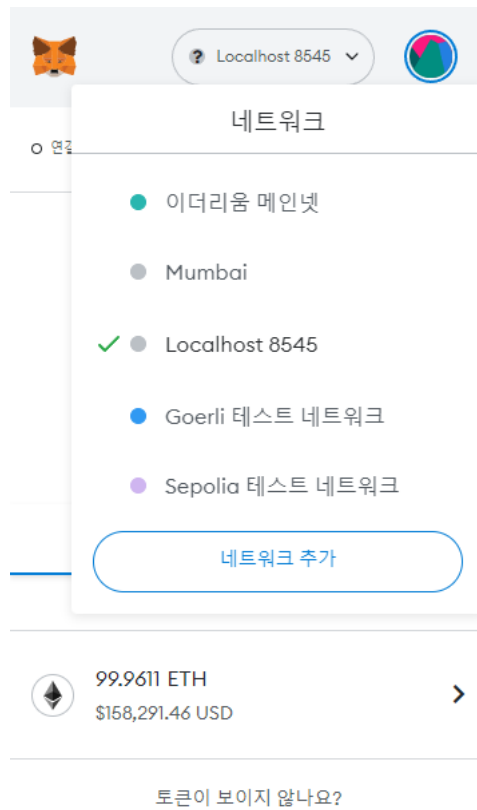
ADDRESS 0x0Ca6d82dB9A10337156fD3b8729D9ABece114760	BALANCE 99.96 ETH	TX COUNT 12	INDEX 0	
ADDRESS 0x0BBC78edEADe58B1513E1Bbd67253F3b8d1ab2eD	BALANCE 100.00 ETH	TX COUNT 0	INDEX 1	
ADDRESS 0x92d9835FbDa7B9E5a3aF331bC23726Ee14f66D52	BALANCE 100.00 ETH	TX COUNT 0	INDEX 2	
ADDRESS 0xa625078337B5B2c70D712E5d992EB259Cb216c64	BALANCE 100.00 ETH	TX COUNT 0	INDEX 3	
ADDRESS 0x5a3C3EAC5E5FCE289881fe9635E6149047Ca790c	BALANCE 100.00 ETH	TX COUNT 0	INDEX 4	
ADDRESS 0xE9868dd9C160762ce3aB350AB16Fb6e2BBe431Fc	BALANCE 100.00 ETH	TX COUNT 0	INDEX 5	
ADDRESS 0x014a2dfe74DDc6990E1a6d5A6309A1990Ef08Ad7	BALANCE 100.00 ETH	TX COUNT 0	INDEX 6	

#### PRIVATE KEY

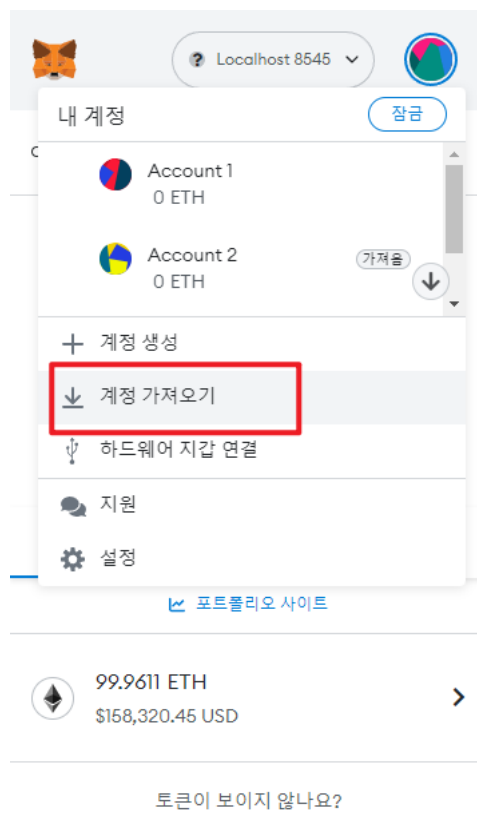
ac1a7d1c2ef31605650a3897f413c8f1f5f9e589346ddba274d84927b1e1c270

Do not use this private key on a public blockchain; use it for development purposes only!

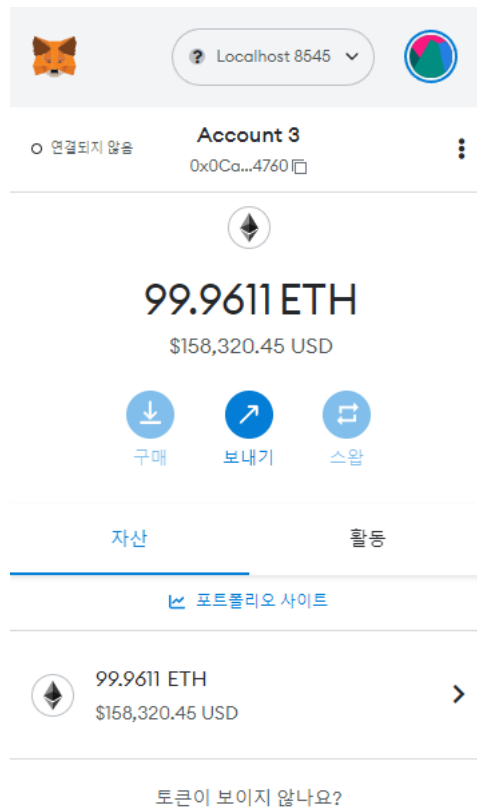
DONE



로컬호스트8545



프라이빗 키를 넣으면



완료

실행 해보기

- npm run dev

에러

app.js (initWeb3 부분 수정)

```
App = {
  web3Provider: null,
  contracts: {},

  init: async function() {
    // Load pets.
    $.getJSON('../pets.json', function(data) {
      var petsRow = $('#petsRow');
      var petTemplate = $('#petTemplate');

      for (i = 0; i < data.length; i++) {
        petTemplate.find('.panel-title').text(data[i].name);
        petTemplate.find('img').attr('src', data[i].picture);
        petTemplate.find('.pet-breed').text(data[i].breed);
        petTemplate.find('.pet-age').text(data[i].age);
        petTemplate.find('.pet-location').text(data[i].location);
        petTemplate.find('.btn-adopt').attr('data-id', data[i].id);

        petsRow.append(petTemplate.html());
      }
    });

    return await App.initWeb3();
  },

  initWeb3: async function() {
    /*
     * 웹페이지 열리고 web3설정
     */
  }
}
```



```

if(window.ethereum) {
  App.web3Provider = window.ethereum;
  try{
    await window.ethereum.enable();
  }catch(error){
    console.error("User denied account access");
  }
}

else if(window.web3){
  App.web3Provider = window.web3.currentProvider;
}
else{
  App.web3Provider = new web3.providers.HttpProvider('http://localhost:8545');
}
web3 = new Web3(App.web3Provider);

return App.initContract();
},

initContract: function() {
  /*
   * 이제 web3를 통해 Ethereum과 상호 작용할 수 있으므로 web3가
   스마트 계약을 찾을 위치와 작동 방식을 알 수 있도록 스마트 계약을
   인스턴스화해야 합니다.
   Truffle에는 이를 도와주는 @truffle/contract.
   마이그레이션과 동기화된 계약에 대한 정보를 유지하므로
   계약의 배포된 주소를 수동으로 변경할 필요가 없습니다.
   */

  $.getJSON('Adoption.json', function(data){
    // Get the necessary contract artifact file and instantiate it with @truffle/contract
    var AdoptionArtifact = data;
    App.contracts.Adoption = TruffleContract(AdoptionArtifact);

    // Set the provider for our contract
    App.contracts.Adoption.setProvider(App.web3Provider);

    // Use our contract to retrieve and mark the adopted pets
    return App.markAdopted
  })

  return App.bindEvents();
},

bindEvents: function() {
  $(document).on('click', '.btn-adopt', App.handleAdopt);
},

markAdopted: function() {
  /*
   * 배포된 Adoption계약에 액세스한 다음 getAdopters() 해당 인스턴스를 호출합니다.

   먼저 스마트 계약 호출 외부에서 변수를 선언하여 adoptionInstance 초기 검색 후 인스턴스에 액세스할 수 있습니다.

   call() 을 사용 하면 전체 트랜잭션을 보내지 않고도 블록체인에서 데이터를 읽을 수 있습니다. 즉, 이더를 사용할 필요가 없습니다.

   를 호출한 후 getAdopters() 모든 애완 동물에 대해 주소가 저장되어 있는지 확인하면서 모든 항목을 반복합니다. 배열에는 주소 유형이 포함되어 있으므로 Ethereum

   해당 주소가 있는 a petId를 찾으면 채택 버튼을 비활성화하고 버튼 텍스트를 "성공"으로 변경하여 사용자가 피드백을 받을 수 있도록 합니다.

   모든 오류는 콘솔에 기록됩니다.
   */

  var adoptionInstance;

  App.contracts.Adoption.deployed().then(function(instance){
    adoptionInstance = instance;

    return adoptionInstance.getAdopters.call();

  }).then(function(adopters){
    for(i = 0; i < adopters.length; i++){
      if(adopters[i] !== '0x0000000000000000000000000000000000000000000000000000000000000000'){
        $(''.panel-pet').eq(i).find('button').text('Success').attr('disabled', true);
      }
    }
  }).catch(function(err){
    console.log(err.message);
  });
},

handleAdopt: function(event) {
  event.preventDefault();

```

```

var petId = parseInt($(event.target).data('id'));

/*
web3를 사용하여 사용자 계정을 가져옵니다. 오류 확인 후 콜백에서 첫 번째 계정을 선택합니다.
여기에서 위에서 수행한 대로 배포된 계약을 가져오고 인스턴스를 에 저장합니다 adoptionInstance.
하지만 이번에는 호출 대신 트랜잭션 을 보낼 것 입니다. 트랜잭션에는 "보낸 사람" 주소가 필요하며 관련 비용이 있습니다.
에테르로 지불되는 이 비용을 가스 라고 합니다. 가스 비용은 계산을 수행하거나 스마트 계약에 데이터를 저장하는 데 드는 비용입니다.
이전 adopt()에 account 트랜잭션을 보낸 결과는 트랜잭션 개체입니다.
오류가 없으면 markAdopted()함수를 호출하여 새로 저장된 데이터와 UI를 동기화합니다.
*/

var adoptionInstance;

web3.eth.getAccounts(function(error, accounts) {
  if (error) {
    console.log(error);
  }

  var account = accounts[0];

  App.contracts.Adoption.deployed().then(function(instance) {
    adoptionInstance = instance;

    // Execute adopt as a transaction by sending account
    return adoptionInstance.adopt(petId, {from: account});
  }).then(function(result) {
    return App.markAdopted();
  }).catch(function(err) {
    console.log(err.message);
  });
});

};

$(function() {
  $(window).load(function() {
    App.init();
  });
});

```

