Solidity Contracts :

- Mocha & Chai
- Contract Lifecycle
- Truffle Development & Deployment

Discount Coupon Link to UDEMY course:

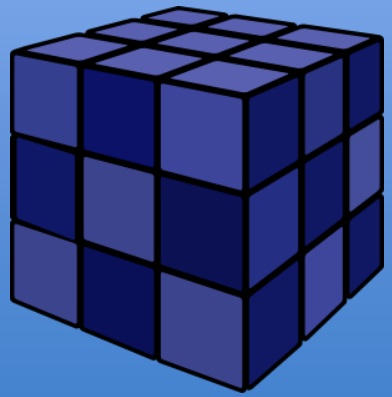
https://www.udemy.com/ethereum-dapp/?couponCode=ETHDAPP101

This deck is part of a online course on "Ethereum: Design and Development of Decentralized Apps.

raj@acloudfan.com

@acloudfan

http://ACloudFan.com



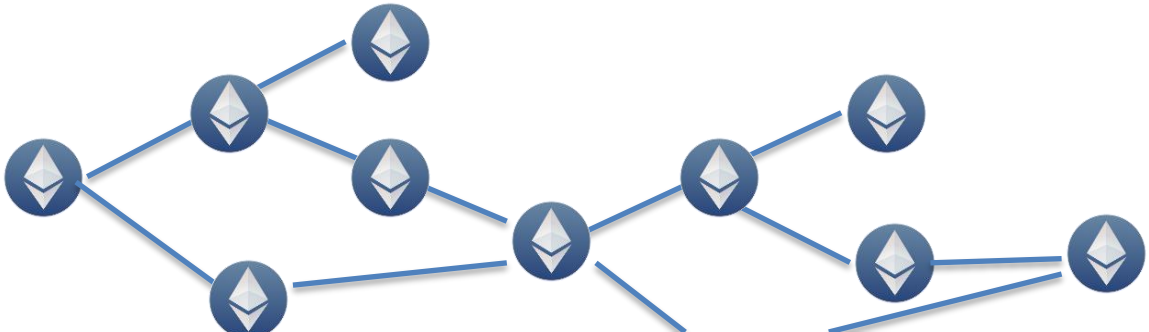
Local Development

>> Ethereum Simulator

Local Development Machine

Compiled Contract

Ethereum Simulator



Truffle

- Provides CLI for developer tasks
- Test driven development

- Local execution of contracts within Truffle environment
- Javascript objects as contract abstraction (*Eth Pudding library*)

```
> npm install –g truffle
```

"Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing *simple* and *fun*"

https://mochajs.org/

describe(…)

- Grouping of test cases
- Allows nesting

it(…)

- Test case
- Uses assertion lib *Chai*

before() beforeEach() after() afterEach()

- Hooks

"Chai is a *Test/Behavior Driven Development (TDD, BDD)* assertion library for node and the browser that can be delightfully paired with any javascript testing framework"

https://chaijs.org/

- Multiple assertion styles

Assert

```
var assert = chai.assert;

assert.typeOf(foo, 'string');
assert.equal(foo, 'bar');
assert.lengthOf(foo, 3)
assert.property(tea, 'flavors');
assert.lengthOf(tea.flavors, 3);
```

Classical assert style

Expect

```
var expect = chai.expect;

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.length(3);
expect(tea).to.have.property('flavors')
  .with.length(3);
```

BDD style

Should

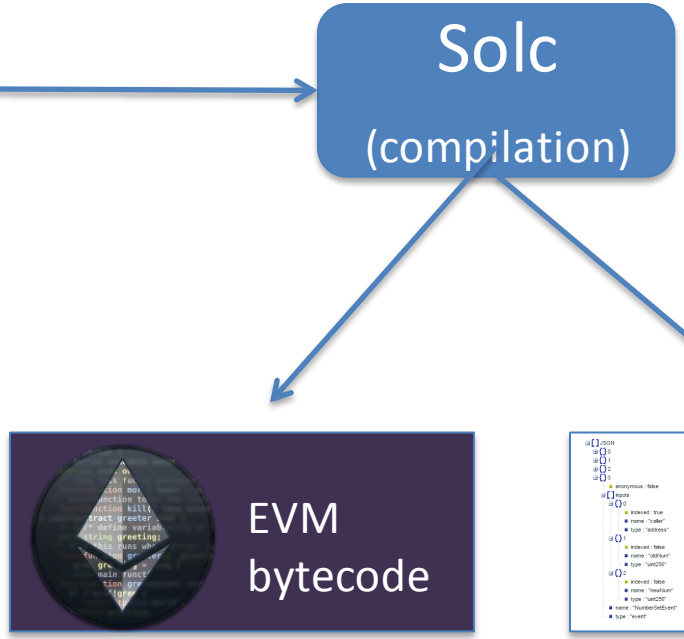
```
chai.should();

foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.length(3);
tea.should.have.property('flavors')
  .with.length(3);
```

BDD style

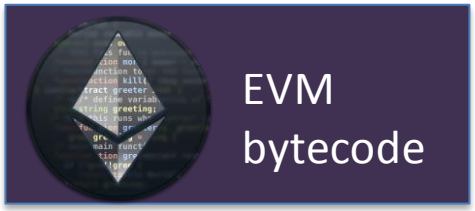
Smart Contract Lifecycle

Compilation

```
pragma solidity ^0.4.6;
contract MyContract {

  uint   num;

  event NumberSetEvent(address indexed caller,
          bytes32 indexed oldNum, bytes32 indexed newNum);

   function getNum()  returns (uint n) {
      return num;
   }
   function setNum(uint n) {
     uint old = num;
     num=n;
     NumberSetEvent(msg.sender,bytes32(old),bytes32(num));
   }
   // constructor
   function MyContract(uint x){num=x;}
}
```



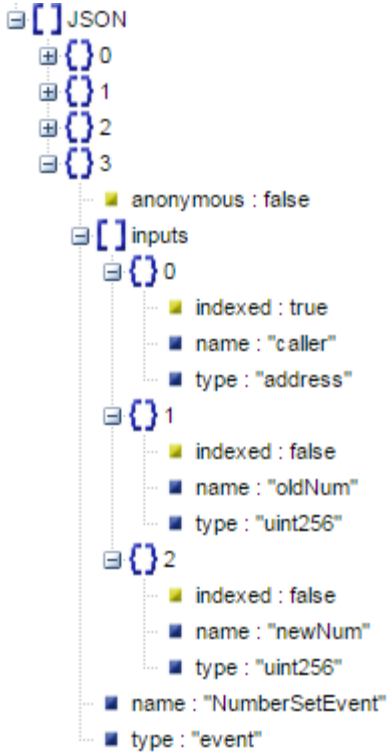
"Application Binary Interface"



- EVM Bytecode | Op codes
  - 140+ op codes

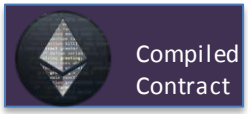
0x00	STOP	Halts execution
0x01	ADD	Addition operation
0x02	MUL	Multiplication operation
0x03	SUB	Subtraction operation
0x04	DIV	Integer division operation
0x05	SDIV	Signed integer
0x06	MOD	Modulo
0x07	SMOD	Signed modulo
0x08	ADDMOD	Modulo
0x09	MULMOD	Modulo
0x0a	EXP	Exponential operation
0x0b	SIGNEXTEND	Extend length of two's complement signed integer

abiDefinition



- Contains function & event *metadata*
- Needed for contract invocation & deployment

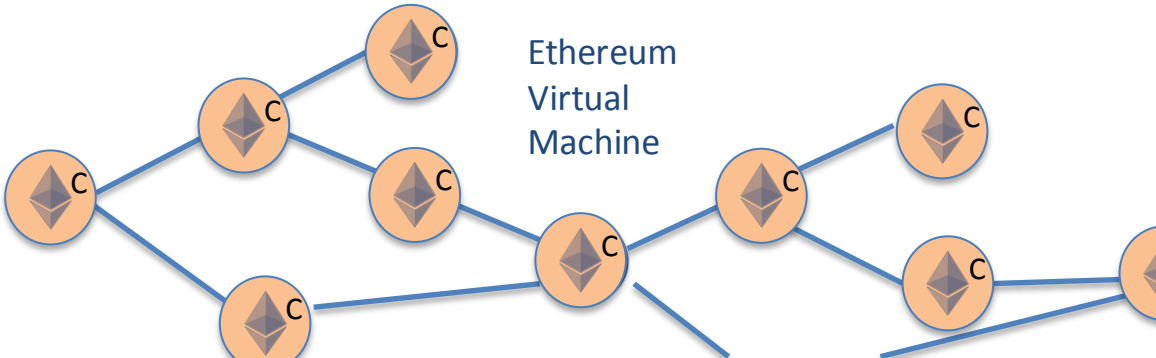
Contract Lifecycle



1. Deployed

2. Invoked

3. Self-Destruct

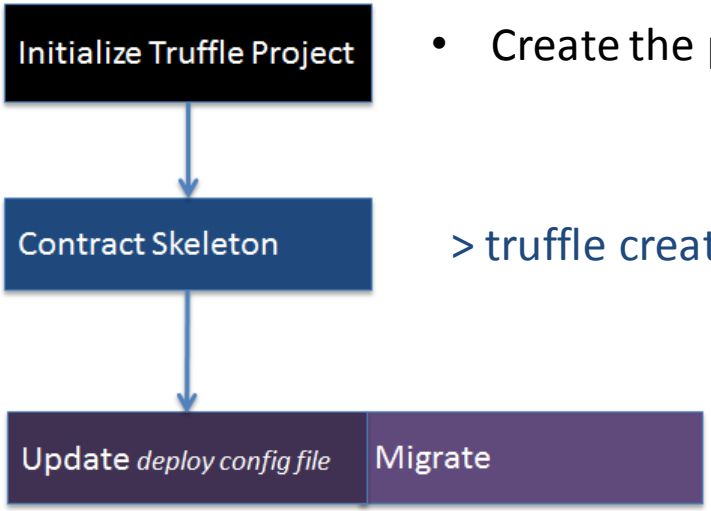


Truffle Contract Development, Deployment

https://github.com/acloudfan/Blockchain-Course-Calculator

/contracts/Calculator.sol
/test/calculator.js

Truffle Project Setup



- Create the project folder > truffle init

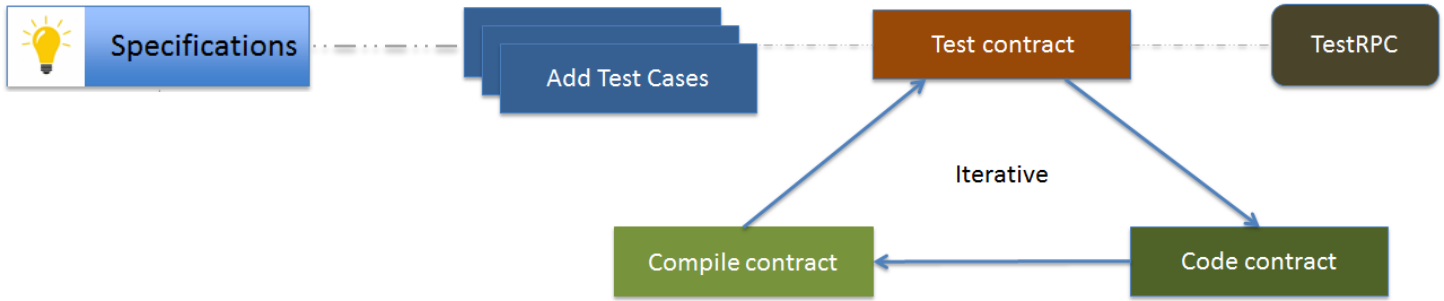
> truffle create contract *Calculator*

- Update the file 2_deploy_contracts > truffle migrate

Truffle : Coding & Testing

> truffle create test calculator

> truffle test ./test/calculator.js

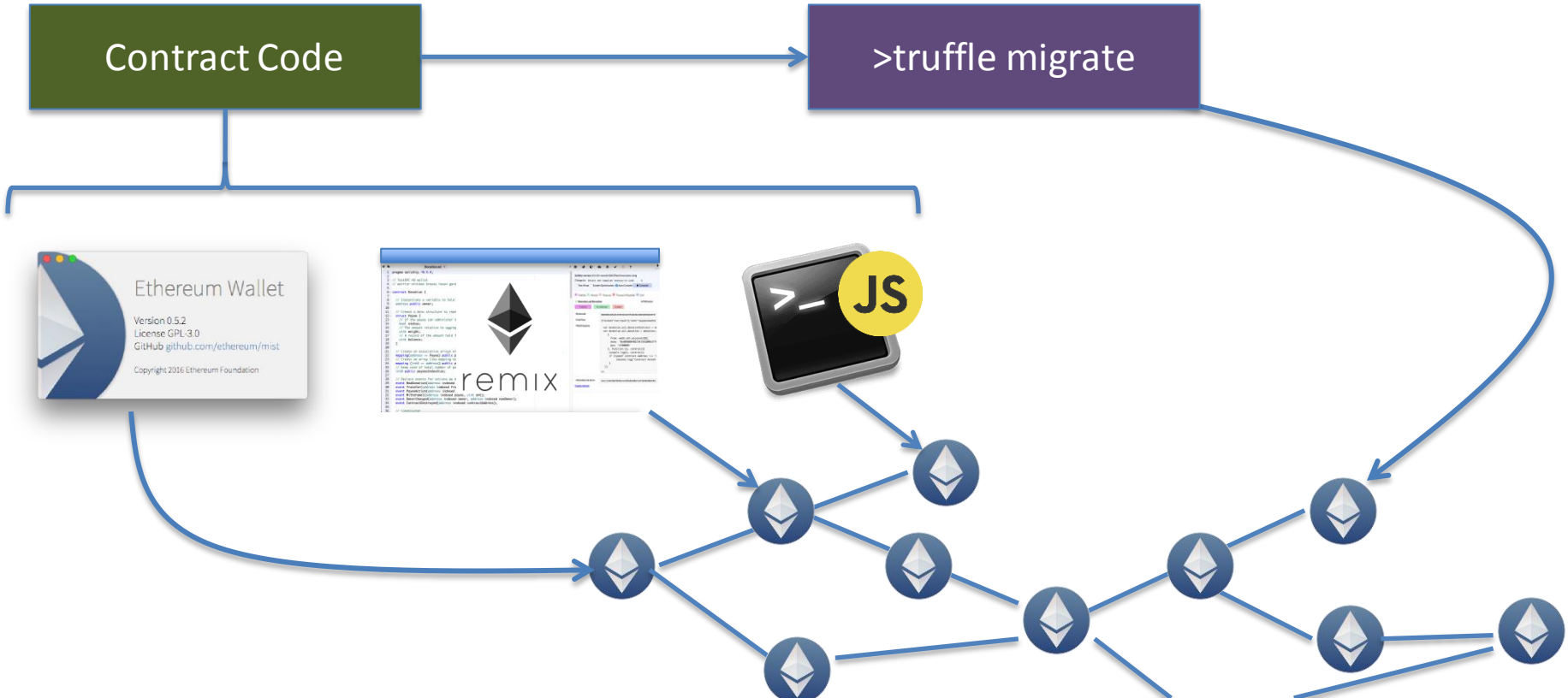


> truffle compile

> truffle migrate

> truffle create contract Calculator

Deployment


Contract Code
>truffle migrate
Ethereum Wallet
Version 0.5.2
License GPL-3.0
GitHub github.com/ethereum/mist
Copyright 2016 Ethereum Foundation
remix
JS

> truffle migrate --options

- `--reset` : Run all migrations from the beginning, instead of running from the last completed migration.
- `-f number` : Run contracts from a specific migration.
- `--network name` : Specify the network to use, saving artifacts specific to that network.
- `--compile-all` : Compile all contracts instead of intelligently choosing.
- `--verbose-rpc` : Log communication between Truffle and the RPC.

> truffle create migration

> truffle migrate --network QA

truffle migrate --network

```
module.exports = {
  networks: {
    development: {
      host: "localhost",
      port: 8545,
      network_id: "*" // Match any network id
    },

    QA: {
      host: "localhost",
      port: 8545,
      network_id: "3"   // ROPSTEN
      // Options - gas, gasPrice, from
    },

    PRODUCTION: {
      host: "localhost",
      port: 8545,
      network_id: "1"   // LIVE
      // Options - gas, gasPrice, from
    }
  }
};
```

TestRPC

> truffle migrate

ROPSTEN

> truffle migrate --network QA

> truffle migrate --network PRODUCTION

LIVE

- Provides a CLI for creating a dapp project

#2 Setup contract

- **2.1** File name = contract name
- **2.2** Setup test cases
- **2.3** Code the contract
- **2.4** Update deployment file

#3 > truffle compile

#4 > truffle test

#5 > truffle migrate

https://truffle.readthedocs.io/en/latest/advanced/configuration/