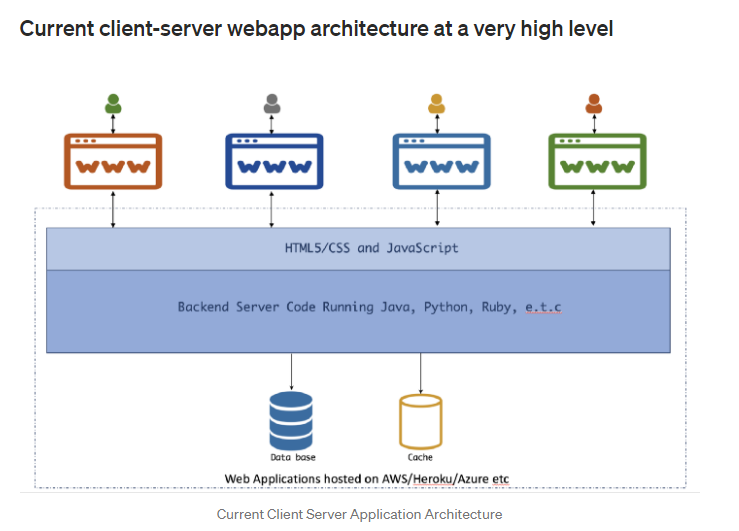
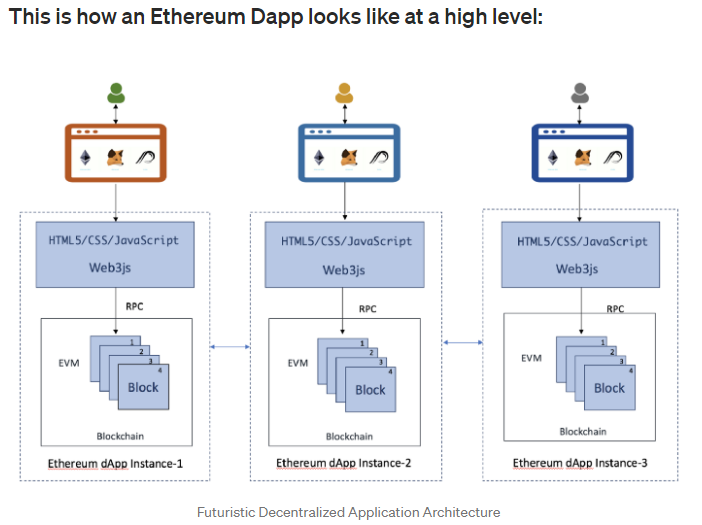
# Developing a DApp. How to access Ethereum.



**Client-server Environment**

* The web application is hosted on a hosting provider like AWS, Heroku or any other VPS.
* The client interacts with this one central application.
* Clients can be a browser, widget, curl command or other API consuming services.
* When a client makes a request via a web browser to the server, the server code implemented interacts with the database and/or cache for CRUD operation and serves the client.



**Dapp Environment**

* If one observes every client (browser) communicates with its own instance of the application.
* There is no central server to which all clients connect to.
* This means every person who wants to interact with a dapp (Decentralized Application) will need a full copy of the blockchain running on their computer/phone etc.
* That means, before you can use an application, you have to download the entire blockchain and then start using the application.

This might sound ridiculous at first but it has the advantage of not relying on a single central server which might disappear tomorrow.

# Getting Started

## Basic tools we need –

Node.js is an open source server environment.

Node.js allows you to run JavaScript on the server.

*Node Package Manager (npm)* — NPM is a package manager for Node.js packages or modules.

*Ganache* — Ganache (earlier TestRPC) is a personal blockchain for Ethereum development that developers can use to deploy contracts, develop applications, and run tests. It runs 100% locally and simulates a full Ethereum client but is much much faster making it ideal for testing and development.

It comes in two flavor — Ganache (GUI) & Ganache-CLI.

Ganache CLI, part of the Truffle suite of Ethereum development tools, is the command line version of [Ganache](https://github.com/trufflesuite/ganache), your personal blockchain for Ethereum development.

ganache-cli is written in JavaScript and distributed as a Node.js package via npm. Make sure you have Node.js (>= v8) installed.

*Truffle* — Truffle is a development environment, testing framework, and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM). Basically, it helps you deploy your contracts to the blockchain, swap old contracts for new ones (called a migration), and hook your front-end up to your deployed contracts

Deploying smart contracts with Truffle and Ganache

To deploy contract we need to have development environment truffle and ethereum client ganache.

1. Truffle
2. Ganache

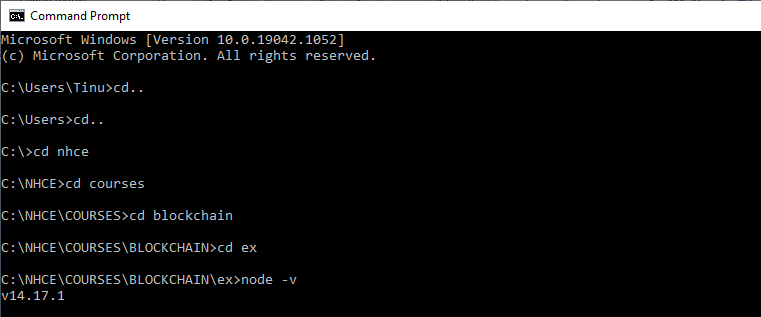
# Once npm is installed using nodejs, we can use it to install ganache-cli, a Node.js Ethereum client that mimics a real Ethereum node and truffle which is the development and testing environment

Truffle is the framework we'll use to help us easily manage and deploy our Solidity codes. Ganache runs a local instance of the Ethereum network for our testing purposes.

**Step1: (i)Download Nodejs**

(ii)Cut paste into a folder (ex)and mention the path till ex when installing Nodejs.

(iii) open cmd and check for version of Nodejs/npm, use the command [node -v or npm -v]



**Step 2: Install truffle globally on your machine [can use same cmd]**

Command: npm install -g truffle

[In case of error try to do same in another cmd or try to install truffle and ganache together using the command: npm install -g ganache-cli truffle or npm install -g ganache(spacehere)-cli truffle ]

**Step 3: Install ganache cli**

Command: npm install -g ganache-cli or ganache(space here)-cli

**Step 4: Once installed make the ganache listen by the command**

Command:ganache-cli

# This will start up the client and generate 10 accounts for you to use, each with 100 ETH. You should see something like this:



once the ganache is listening, can initialize truffle in another cmd by using

**Step 5: Initialize Truffle**

Command: truffle init [To create a new project in truffle framework]

**Step 6: Create Contract**

Inside the contracts folder, write the contract in solidity and save it will “Cname.sol” [all files]

**Step 7: compile the contract in truffle**

Command: truffle compile

**Step 8: Deploy contract in truffle framework**

**Step 8.1 First need to migrate the contract**

Migration happens in these 3 steps

1. Import the smart contract from contracts folder [artifacts.require]
2. Define a function that implements migration step
3. Export this function so that truffle can use it.

Create a file named “2\_deploy\_contracts.js” [all files type] and write the following code

**var calc = artifacts.require("Calculator");**

**module.exports = function(deployer) {**

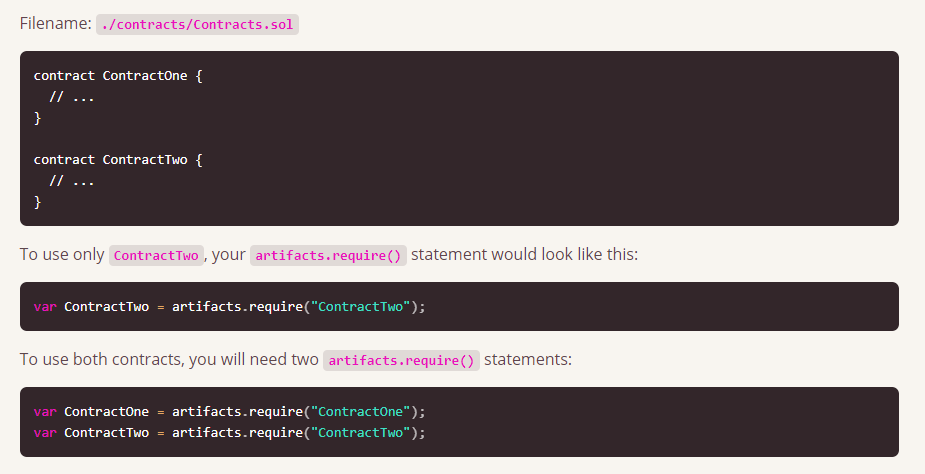
**deployer.deploy(calc);**

**// Additional contracts can be deployed here**

**};**

**artifacts.require() method**

At the beginning of the migration, we tell Truffle which contracts we'd like to interact with via the artifacts.require() method. The name specified should match **the name of the contract definition** within that source file. Do not pass the name of the source file, as files can contain more than one contract.



### module.exports

All migrations must export a function via the module.exports syntax. The function exported by each migration should accept a deployer object as its first parameter.

**Step 9: Modify the file truffle\_config.js , paste the below contents**

module.exports={

networks:{

development:{

host:"localhost",

network\_id:'\*',

port:8545

}}}

The project is developed in the local host node, we have only 1 node here.

**Step 10: Truffle migration**

Command: truffle migrate

**To test using truffle**

Step 1: Copy the test program for the smart contract (name\_test.sol) in the test folder (all files)

Command : truffle test.

<https://medium.com/coinmonks/getting-started-with-ethereum-and-building-basic-dapp-ebb681fb3748>

<https://medium.com/coinmonks/get-started-with-building-ethereum-dapps-and-smart-contracts-d86b9f7bd1c>

<https://medium.com/coinmonks/creating-deploying-a-smart-contract-using-truffle-framework-ganache-cli-part-2-f2dcf400fbde>