# Angular JS

Open source

<https://angularjs.org/>

https://www.tutorialspoint.com/angularjs/angularjs\_environment.htm

## General Features

The general features of AngularJS are as follows −

* AngularJS is a efficient framework that can create Rich Internet Applications (RIA).
* AngularJS provides developers an options to write client side applications using JavaScript in a clean Model View Controller (MVC) way.
* Applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

Overall, AngularJS is a framework to build large scale, high-performance, and easyto-maintain web applications.

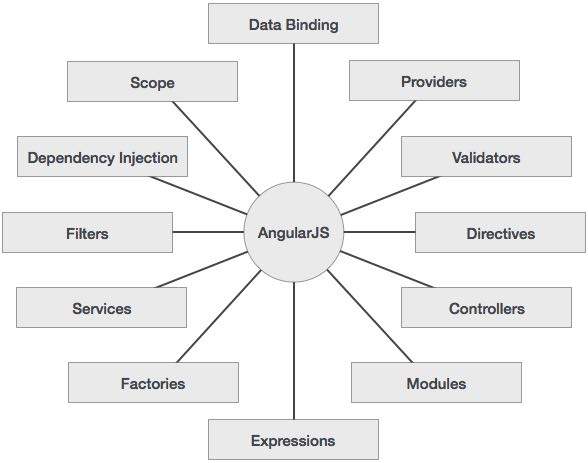
## Core Features

The core features of AngularJS are as follows −

* **Data-binding** − It is the automatic synchronization of data between model and view components.
* **Scope** − These are objects that refer to the model. They act as a glue between controller and view.
* **Controller** − These are JavaScript functions bound to a particular scope.
* **Services** − AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.
* **Filters** − These select a subset of items from an array and returns a new array.
* **Directives** − Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel, etc.
* **Templates** − These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using *partials*.
* **Routing** − It is concept of switching views.
* **Model View Whatever** − MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.
* **Deep Linking** − Deep linking allows to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.
* **Dependency Injection** − AngularJS has a built-in dependency injection subsystem that helps the developer to create, understand, and test the applications easily.

## Concepts

The following diagram depicts some important parts of AngularJS which we will discuss in detail in the subsequent chapters.



## Advantages of AngularJS

The advantages of AngularJS are −

* It provides the capability to create Single Page Application in a very clean and maintainable way.
* It provides data binding capability to HTML. Thus, it gives user a rich and responsive experience.
* AngularJS code is unit testable.
* AngularJS uses dependency injection and make use of separation of concerns.
* AngularJS provides reusable components.
* With AngularJS, the developers can achieve more functionality with short code.
* In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

On the top of everything, AngularJS applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

## Disadvantages of AngularJS

Though AngularJS comes with a lot of merits, here are some points of concern −

* **Not Secure** − Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
* **Not degradable** − If the user of your application disables JavaScript, then nothing would be visible, except the basic page.

## AngularJS Directives

The AngularJS framework can be divided into three major parts −

* **ng-app** − This directive defines and links an AngularJS application to HTML.
* **ng-model** − This directive binds the values of AngularJS application data to HTML input controls.
* **ng-bind** − This directive binds the AngularJS application data to HTML tags.

**Install Angular JS**

1. **Download angular.min.js and add <script> to your html file**

**(working in Chrome but not in Firefox)**

## Prerequisites

Before you begin, make sure your development environment includes Node.js® and an npm package manager.

### Node.js

Angular requires Node.js version 8.x or 10.x.

* To check your version, run node -v in a terminal/console window.
* To get Node.js, go to [nodejs.org](https://nodejs.org/).

### npm package manager

Angular, the Angular CLI, and Angular apps depend on features and functionality provided by libraries that are available as [npm packages](https://docs.npmjs.com/getting-started/what-is-npm). To download and install npm packages, you must have an npm package manager.

This Quick Start uses the [npm client](https://docs.npmjs.com/cli/install) command line interface, which is installed with Node.js by default.

To check that you have the npm client installed, run npm -v in a terminal/console window.

## Step 1: Install the Angular CLI

You use the Angular CLI to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.

Install the Angular CLI globally.

To install the CLI using npm, open a terminal/console window and enter the following command:

content\_copynpm install -g @angular/cli

## Step 2: Create a workspace and initial application

You develop apps in the context of an Angular [**workspace**](https://angular.io/guide/glossary#workspace). A workspace contains the files for one or more [**projects**](https://angular.io/guide/glossary/#project). A project is the set of files that comprise an app, a library, or end-to-end (e2e) tests.

To create a new workspace and initial app project:

1. Run the CLI command ng new and provide the name my-app, as shown here:

content\_copyng new my-app

1. The ng new command prompts you for information about features to include in the initial app project. Accept the defaults by pressing the Enter or Return key.

The Angular CLI installs the necessary Angular npm packages and other dependencies. This can take a few minutes.

It also creates the following workspace and starter project files:

* A new workspace, with a root folder named my-app
* An initial skeleton app project, also called my-app (in the src subfolder)
* An end-to-end test project (in the e2e subfolder)
* Related configuration files

The initial app project contains a simple Welcome app, ready to run.

## Step 3: Serve the application

Angular includes a server, so that you can easily build and serve your app locally.

1. Go to the workspace folder (my-app).
2. Launch the server by using the CLI command ng serve, with the --open option.

content\_copycd my-app

ng serve --open

The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files.

The --open (or just -o) option automatically opens your browser to [http](https://angular.io/api/common/http)://localhost:4200/.

Your app greets you with a message:

## Step 4: Edit your first Angular component

[Components](https://angular.io/guide/glossary#component) are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on that input.

As part of the initial app, the CLI created the first Angular component for you. It is the root component, and it is named app-root.

1. Open ./src/app/app.component.ts.
2. Change the title property from 'my-app' to 'My First Angular App'.

src/app/app.component.ts

content\_copy@[Component](https://angular.io/api/core/Component)({

selector: 'app-root',

templateUrl: './app.component.html',

[styleUrls](https://angular.io/api/core/Component#styleUrls): ['./app.component.css']

})

export class AppComponent {

title = 'My First Angular App!';

}

The browser reloads automatically with the revised title. That's nice, but it could look better.

1. Open ./src/app/app.component.css and give the component some style.

src/app/app.component.css

content\_copyh1 {

color: #369;

font-family: Arial, Helvetica, sans-serif;

font-size: 250%;

}

### Add Some Control

#### Data Binding

Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is awesome because it eliminates DOM manipulation from the list of things you have to worry about.

#### Controller

Controllers are the behavior behind the DOM elements. AngularJS lets you express the behavior in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes.

# D3 JS

Data Driven Documentation java script

### Scalable Vector Graphics (SVG)

We can draw other shapes in SVG such as − Line, Circle, Ellipse, Text and Path.

d3.**select**("div.myclass").**append**("span").**text**("from D3.js");

<**svg** width = "500" height = "500">

<**rect** x = "0" y = "0" width = "300" height = "200"></**rect**>

</**svg**>

<!DOCTYPE html>

<html>

<head>

<script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>

</head>

<body>

<div class = "myclass">

Hello World!

</div>

<script>

d3.select("div.myclass").append("span").text("from D3.js");

</script>

</body>

</html>

### The html() Method

The html() method is used to set the html content of the selected / appended elements.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>  </head>  <body>  <div class = "myclass">  Hello World! sdfasdfadsfs  </div>    <script>  d3.select**(".myclass").**html("Hello World! <span>from D3.js</span>");  </script>  </body>  </html>  **output :** Hello World! from D3.js |

|  |  |
| --- | --- |
| <!DOCTYPE html>  <html>  <head>  <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>  </head>  <body>  <div class = "myclass">  Hello World! Good Morning  </div>    <script>  d3.select**("#myclass").**html("Hello World! <span>from D3.js</span>");  </script>  </body>  </html>  **output :** Hello World! Good Morning |  |

### The attr() Method

The attr() method is used to add or update the attribute of the selected elements. Create a webpage “select\_and\_modify.html” and add the following code.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>  </head>  <body>  <div class = "myclass">  Hello World!  </div>    <script>  d3.select(".myclass").attr("style", "color: red");  </script>  </body>  </html>  Output : Hello World! |

### The style() Method

The style() method is used to set the style property of the selected elements. Create a webpage “select\_and\_style.html” and add the following code.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"></script>  </head>  <body>  <div class = "myclass">  Hello World!  </div>    <script>  d3.select(".myclass").style("color", "red");  </script>  </body>  </html>  Output : Hello World! |

### The classed() Method

The classed() method is exclusively used to set the “class” attribute of an HTML element. Since, a single HTML element can have multiple classes; we need to be careful while assigning a class to an HTML element. This method knows how to handle one or many classes on an element, and it will be performant.

* **Add class** − To add a class, the second parameter of the classed method must be set to true. It is defined below −

d3.select(".myclass").classed("myanotherclass", true);

* **Remove class** − To remove a class, the second parameter of the classed method must be set to false. It is defined below −

d3.select(".myclass").classed("myanotherclass", false);

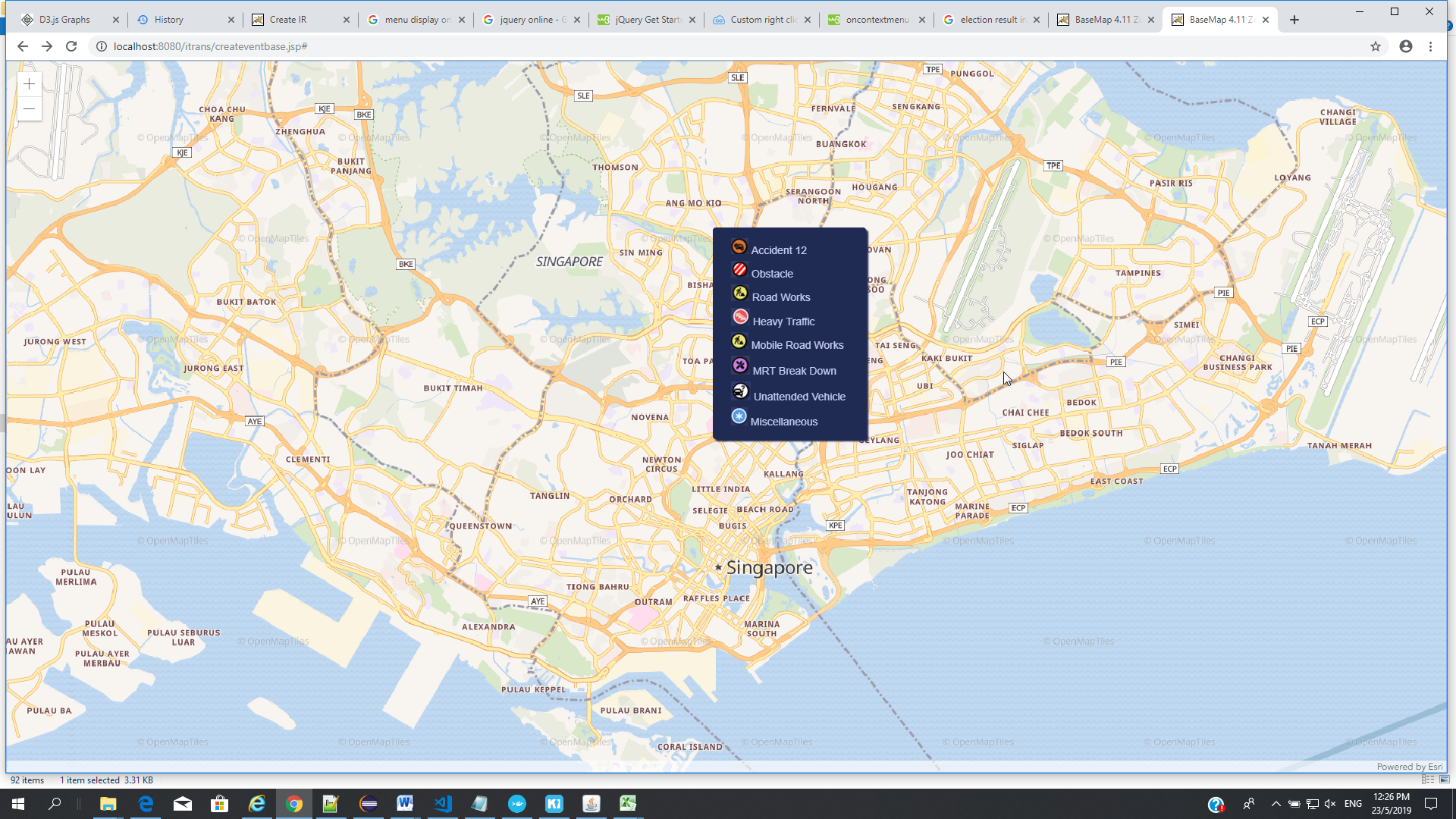
* **Check class** − To check for the existence of a class, just leave off the second parameter and pass the class name you are querying. This will return true, if it exists, false, if it does not.

d3.select(".myclass").classed("myanotherclass");

## The selectAll() Method

The selectAll() method is used to select multiple elements in the HTML document. The select method selects the first element, but the selectAll method selects all the elements that match the specific selector string. In case the selection matches none, then it returns an empty selection. We can chain all the appending modifying methods, **append(), html(), text(), attr(), style(), classed(),** etc., in the selectAll() method as well. In this case, the methods will affect all the matching elements. Let us understand by creating a new webpage “select\_multiple.html” and add the following script –

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <script type = "text/javascript" src = "https://d3js.org/d3.v4.min.js"> </script>  </head>  <body>  <h2 class = "myclass">Message</h2>  <div class = "myclass">  Hello World!  </div>    <script>  d3.selectAll(".myclass").attr("style", "color: red");  </script>  </body>  </html> |

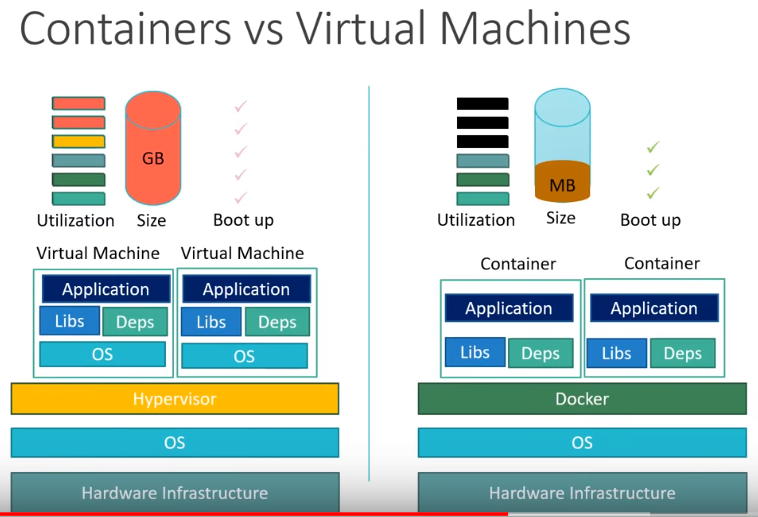
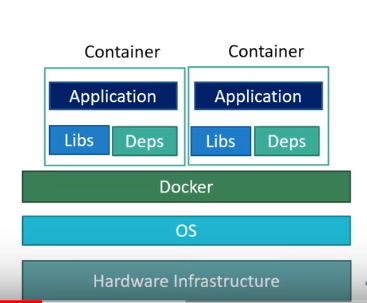
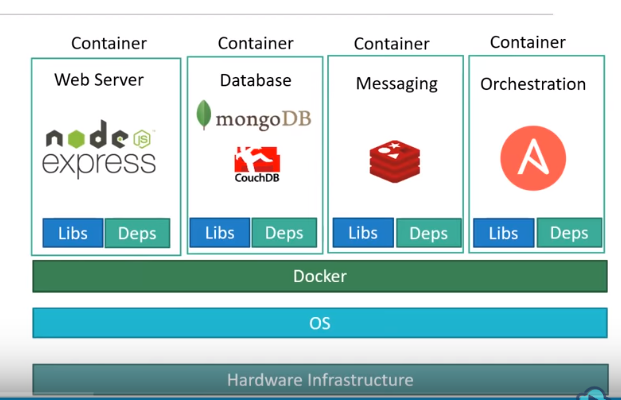
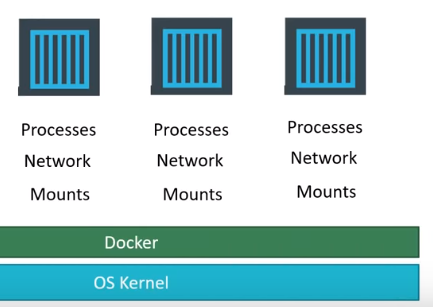


# Dockers

It solves

Dependency / Software version

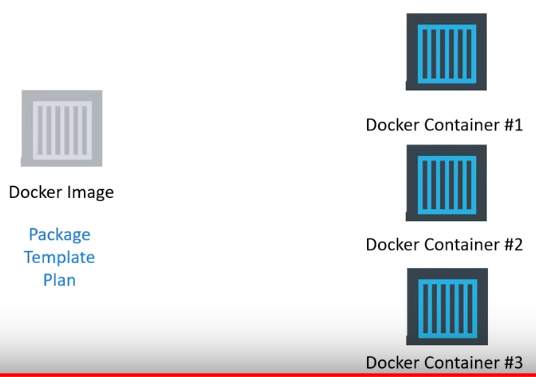
Compatibility issue



**Create the instance using Docker run command for your software requirement**

**If you want Node js**

|  |  |
| --- | --- |
| **Docker run mongodb**  **Docker run node.js**  **Docker run node.js** |  |



Application

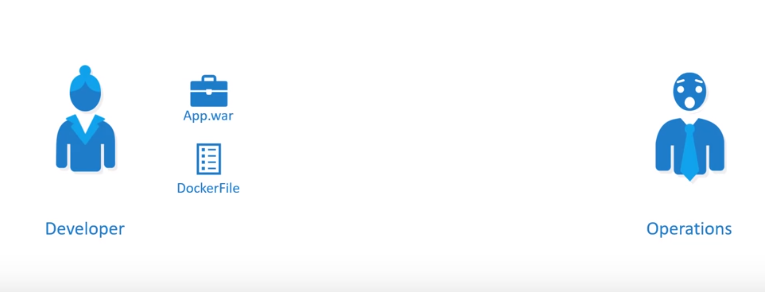
DB

Node.js

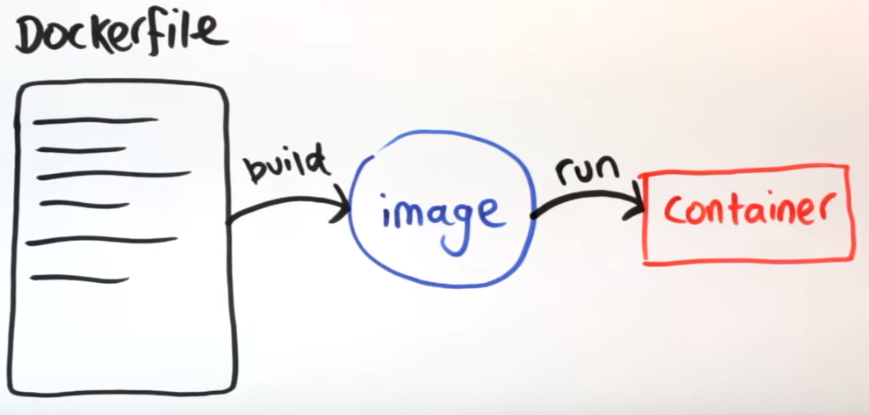
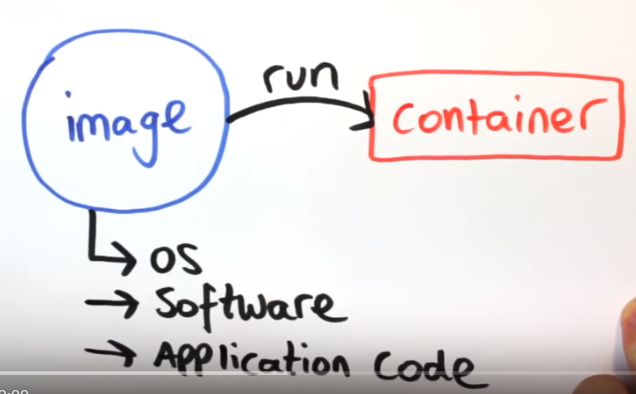
Library

Dependency

1. Developer build docker image
2. Operator can use the image
3. To deploy the application



In production no need to installation software



**Docker file is a text file**

1. Source code .war file
2. Software list
3. Dependency files etc.,