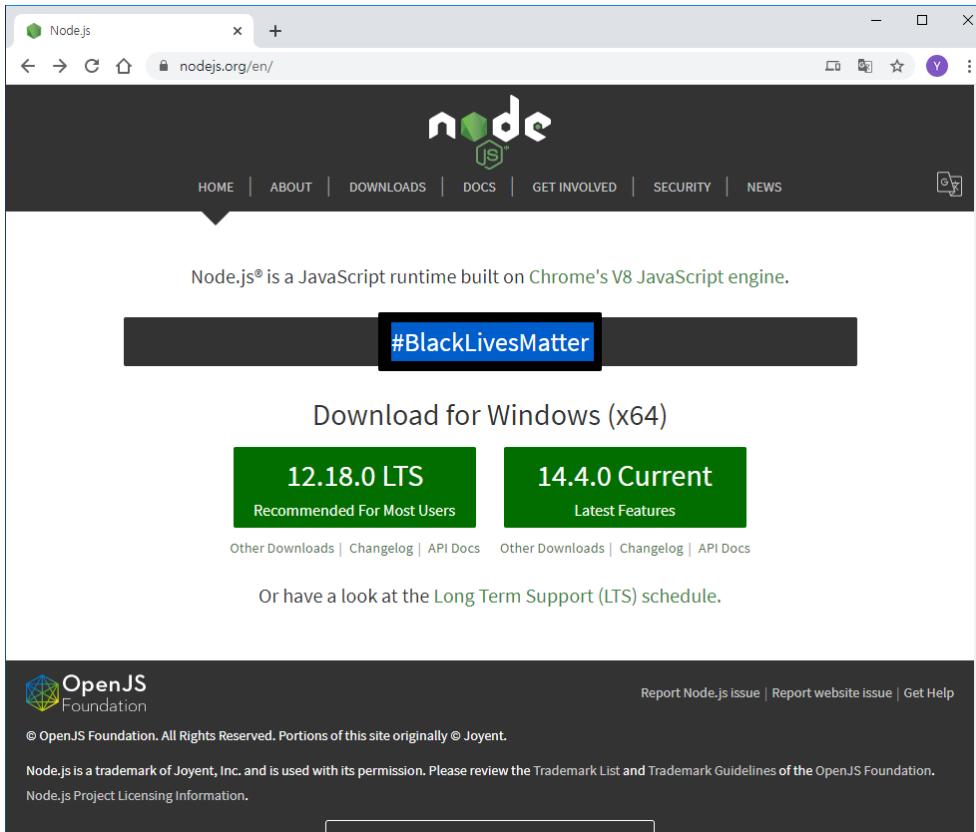


Tutorials for Map-UI

0. 설치 가이드

A. Node.js를 설치한다.

- <https://nodejs.org>에서 LTS 버전을 설치한다.

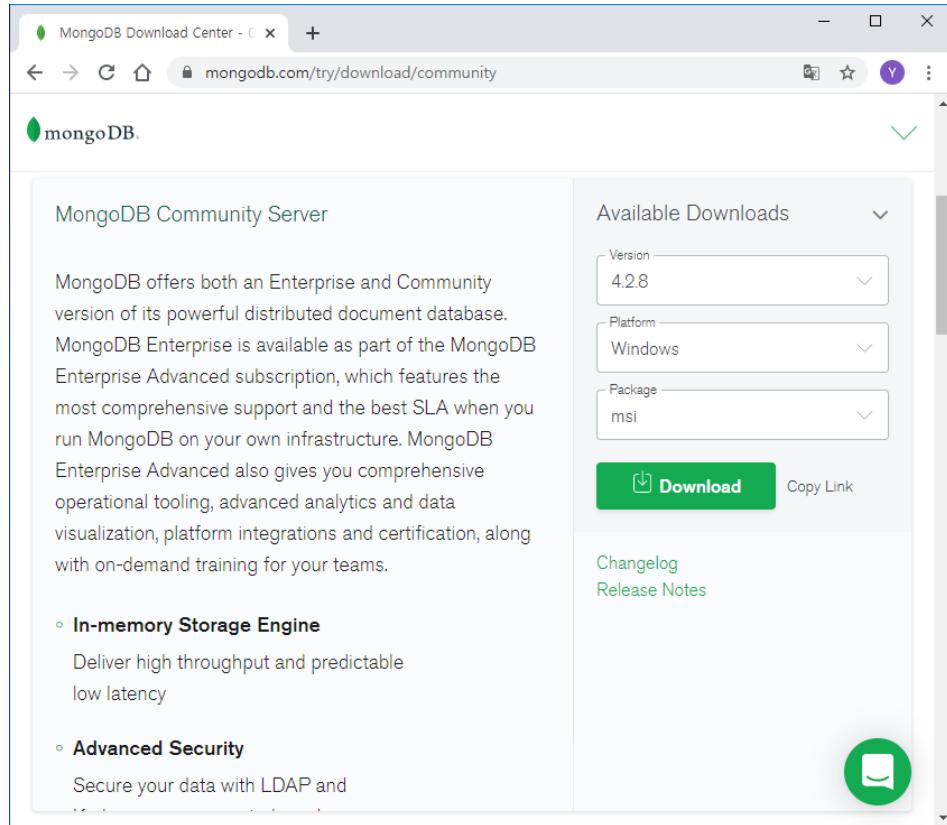


```
C:\W> npm install http-server -g
```

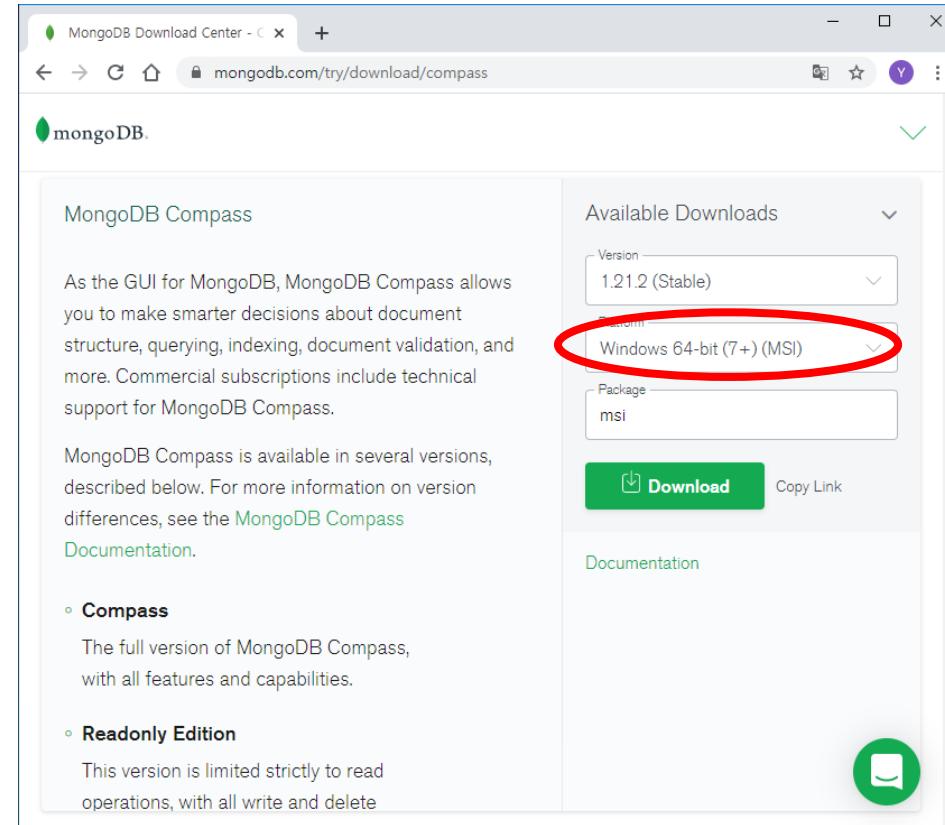
<https://nodejs.org/>

0. 설치 가이드

B. MongoDB Community Server와 Compass를 설치한다.



<https://www.mongodb.com/try/download/community>



<https://www.mongodb.com/try/download/compass>

Community version의 compass에는 기능적인 제약이 있다고 함.
MongoDB 설치할 때는 compass를 설치하지 말고
별도로 다운 받아서 설치하는 것이 좋음.

0. 설치 가이드

C. Minikube를 설치한다.

- <https://kubernetes.io/ko/docs/tasks/tools/install-minikube/>

1) Minikube가 동작하기 위해서 가상화 서비스가 필요하다.

Windows 10 Enterprise / Professional에서는 Hyper-V를 사용하면 되고,

Windows 10 Home 등에서는 Virtual Box를 설치한다(<https://www.virtualbox.org/wiki/Downloads>).

2) Minikube를 설치한다.

(<https://github.com/kubernetes/minikube/releases/latest/download/minikube-installer.exe>)

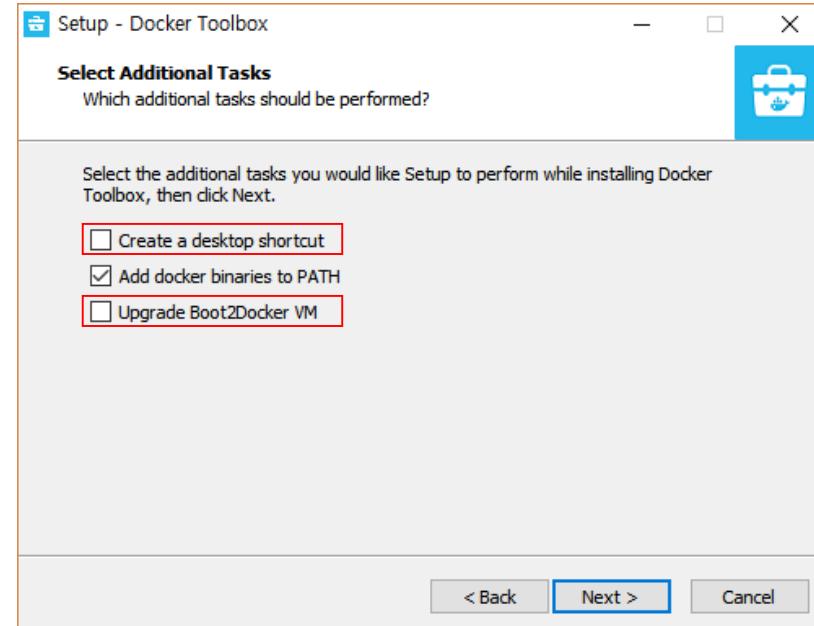
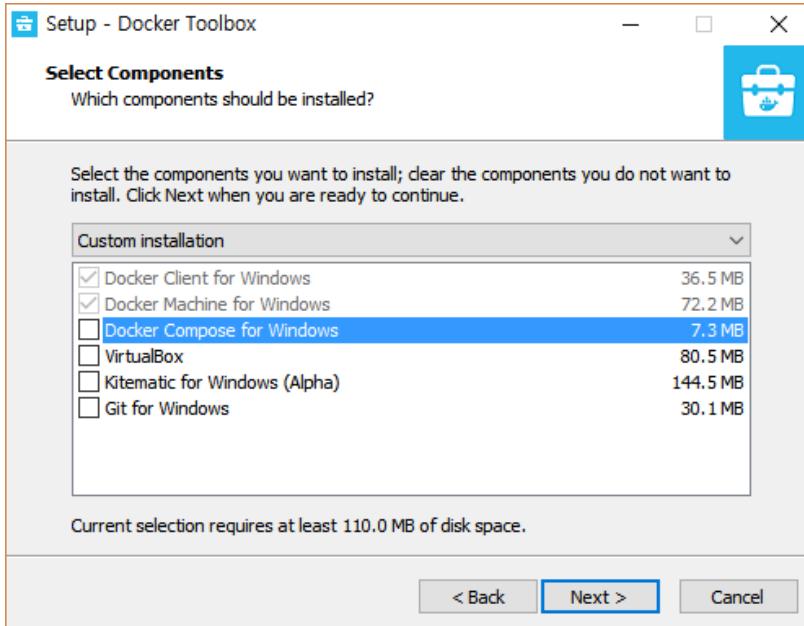
3) Kubectl을 설치한다.

(<https://kubernetes.io/ko/docs/tasks/tools/install-kubectl/>)

0. 설치 가이드

D. Docker Toolbox를 설치한다.

- <https://download.docker.com/win/stable/DockerToolbox.exe>
- Docker Engine과 Client만 설치함. Boot2Docker VM도 하지 않음.
Docker Quickstart Terminal 도 사용하지 않음.



1. Google Maps API 가이드

■ Google Map 보기(tutorials/google-maps/index1.html)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Google Maps Tutorial</title>
    <style type="text/css">
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }

      /* Makes the sample page fill the window. */
      html,
      body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
```

```
<script>
  function initMap() {
    // The location of seoul station
    var seoul = { lng: 126.9706673, lat: 37.5547787 };
    var options = {
      center: seoul,
      zoom: 15,
    };
    var map = new google.maps.Map(document.getElementById('map'), options);
  }
</script>
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callback=initMap">
</script>
</head>
<body>
  <div id="map"></div>
</body>
</html>
```

1. Google Maps API 가이드

■ Marker 보이기(tutorials/google-maps/index2.html)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Google Maps Tutorial</title>
    <style type="text/css">
      /* Always set the map height explicitly to define the size of the div
       * element that contains the map. */
      #map {
        height: 100%;
      }

      /* Makes the sample page fill the window. */
      html,
      body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>

<script>
  function initMap() {
    // The location of seoul station
    var seoul = { lng: 126.9706673, lat: 37.5547787 };
    var options = {
      center: seoul,
      zoom: 15,
    };
    var map = new google.maps.Map(document.getElementById('map'), options);
    var marker = new google.maps.Marker({ position: seoul, map: map });
  }
</script>
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callback=initMap">
</script>

```

1. Google Maps API 가이드

- 여러 개의 Marker 보이기(tutorials/google-maps/index3.html)

```
<script>
  function initMap() {
    // The location of seoul station
    var seoul = { lng: 126.9706673, lat: 37.5547787 };
    var options = {
      center: seoul,
      zoom: 15,
    };
    var map = new google.maps.Map(document.getElementById('map'), options);
//    var marker = new google.maps.Marker({ position: seoul, map: map });

    var locations = [
      { lng: 126.97081750370484, lat: 37.55506788676746 },
      { lng: 126.97714751697998, lat: 37.56547786297515 },
      { lng: 126.98303263797399, lat: 37.57004847771681 },
    ];

    var markers = [];
    for (var i = 0; i < locations.length; i++){
      markers[i] = new google.maps.Marker({ position: locations[i], map: map });
    }
  }
</script>
```

1. Google Maps API 가이드

■ 클러스터링(tutorials/google-maps/index4.html)

```
<script>
  function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });

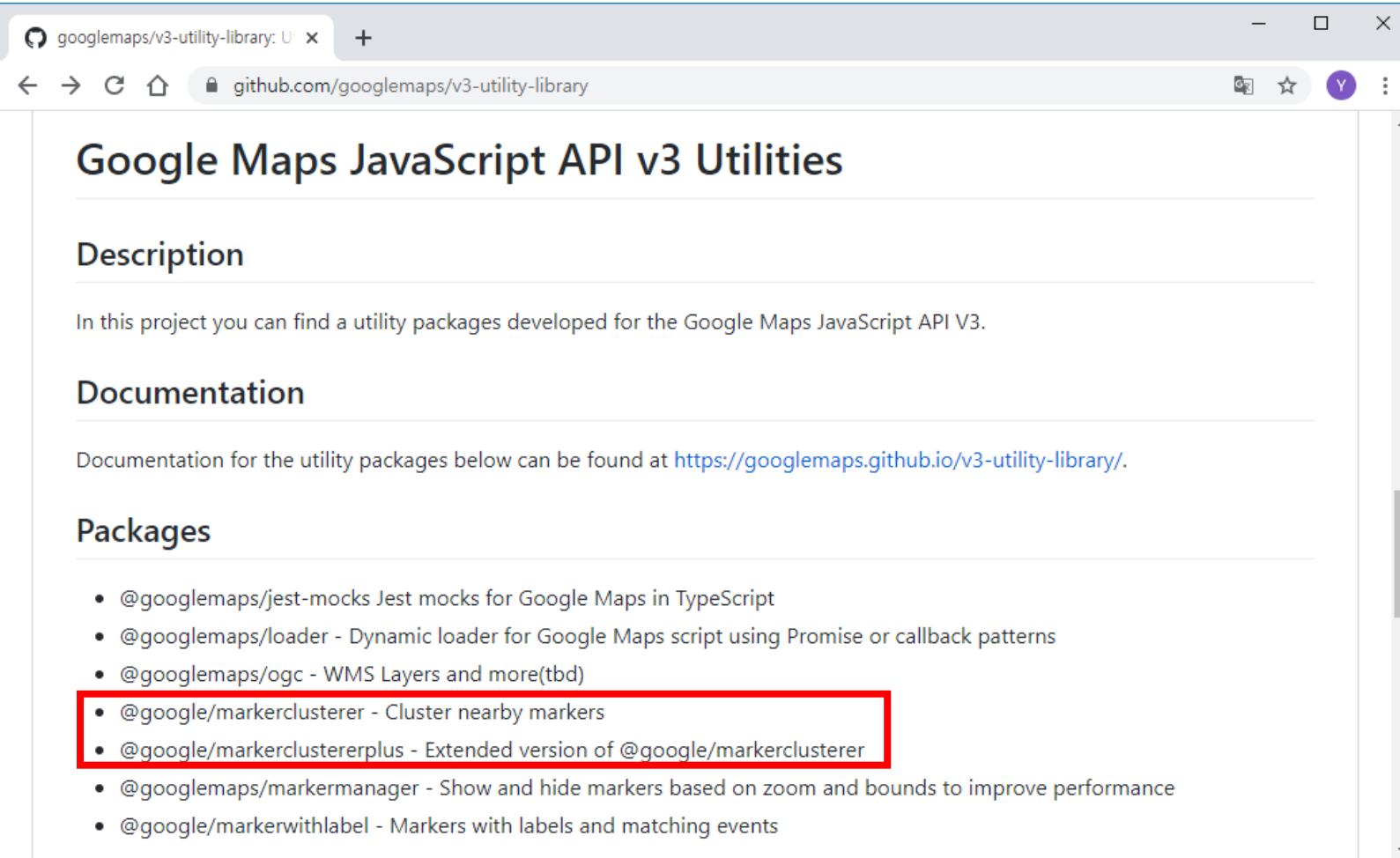
    var locations = [
      { lng: 126.97081750370484, lat: 37.55506788676746 },
      { lng: 126.97714751697998, lat: 37.56547786297515 },
      { lng: 126.98303263797399, lat: 37.57004847771681 },
    ];

    var markers = locations.map(function(location, i){
      return new google.maps.Marker({ position: location, map: map });
    });

    // Add a marker clusterer to manage the markers.
    var markerCluster = new MarkerClusterer(map, markers,
      { imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m' });
  }
</script>
<script src="https://unpkg.com/@google/markerclustererplus@4.0.1/dist/markerclustererplus.min.js" />
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callback=initMap />
```

1. Google Maps API 가이드

■ 클러스터링(tutorials/google-maps/index4.html)



The screenshot shows a browser window with the URL github.com/googlemaps/v3-utility-library. The page title is "Google Maps JavaScript API v3 Utilities". It contains sections for "Description", "Documentation", and "Packages". The "Packages" section lists several npm packages, with two of them, "@google/markerclusterer" and "@google/markerclustererplus", highlighted by a red rectangular box.

Description

In this project you can find a utility packages developed for the Google Maps JavaScript API V3.

Documentation

Documentation for the utility packages below can be found at <https://googlemaps.github.io/v3-utility-library/>.

Packages

- [@googlemaps/jest-mocks](#) Jest mocks for Google Maps in TypeScript
- [@googlemaps/loader](#) - Dynamic loader for Google Maps script using Promise or callback patterns
- [@googlemaps/ogc](#) - WMS Layers and more(tbd)
- [@google/markerclusterer](#) - Cluster nearby markers
- [@google/markerclustererplus](#) - Extended version of @google/markerclusterer
- [@googlemaps/markermanager](#) - Show and hide markers based on zoom and bounds to improve performance
- [@google/maps-markerwithlabel](#) - Markers with labels and matching events

1. Google Maps API 가이드

■ 마우스 이벤트(tutorials/google-maps/index5.html)

```
<script>
  function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });

    var locations = [
      { lng: 126.97081750370484, lat: 37.55506788676746 },
      { lng: 126.97714751697998, lat: 37.56547786297515 },
      { lng: 126.98303263797399, lat: 37.57004847771681 },
    ];

    var markers = locations.map(function(location, i){
      var marker = new google.maps.Marker({ position: location, map: map });
      marker.addListener('click', function(e){
        var position = marker.getPosition();
        infowindow.setPosition(position);
        infowindow.setContent("{ lng: " + position.lng() + ", lat: " + position.lat() + "}");
        infowindow.open(map);
      });

      return marker;
    });
    const infowindow = new google.maps.InfoWindow();
  }
</script>
```

1. Google Maps API 가이드

■ Google Maps Data Layer(tutorials/google-maps/index6.html)

```
<script>
function initMap() {
  var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });

  // GeoJSON format data
  var geodata = {
    "type": "FeatureCollection",
    "features": [
      { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.97081750370484, 37.55506788676746 ] },
        "properties": { "name": "서울역" } },
      { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.97714751697998, 37.56547786297515 ] },
        "properties": { "name": "시청역" } },
      { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.98303263797399, 37.5700484771681 ] },
        "properties": { "name": "종각역" } }
    ];
  };

  map.data.addGeoJson(geodata);
}
</script>
```

1. Google Maps API 가이드

▪ GeoJSON - <https://en.wikipedia.org/wiki/GeoJSON>

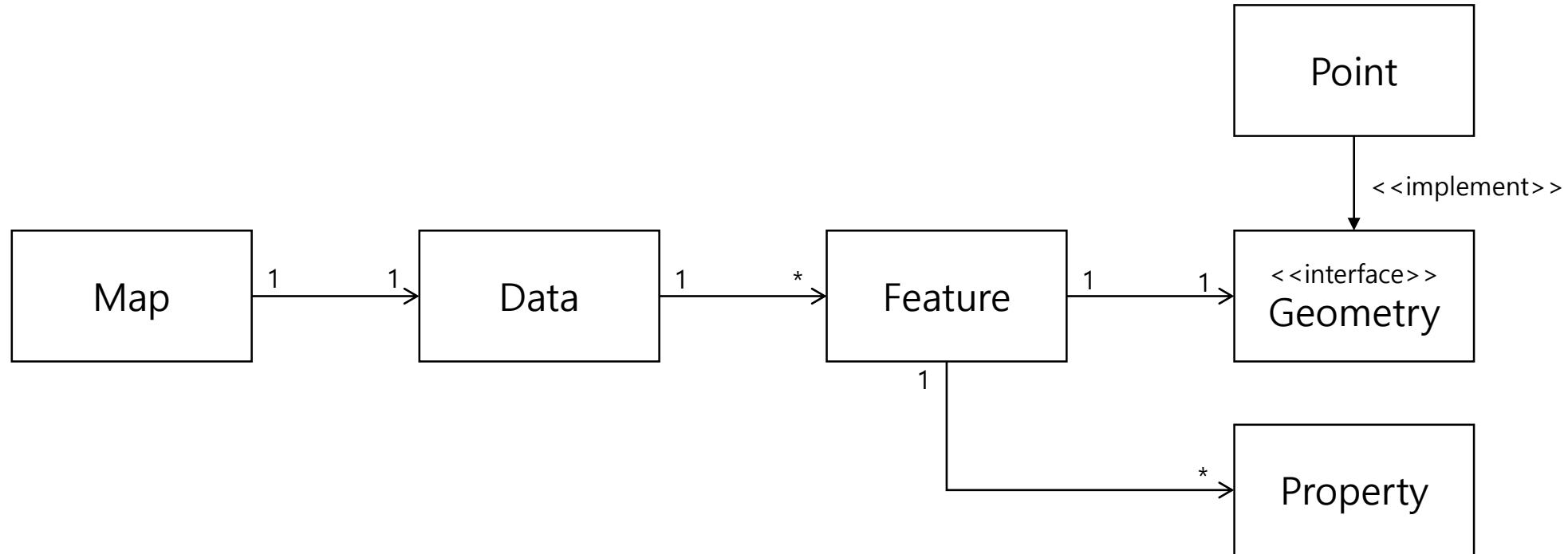
an open standard format designed for representing simple geographical features, along with their non-spatial attributes.

The features include points (therefore addresses and locations), line strings (therefore streets, highways and boundaries), polygons (countries, provinces, tracts of land), and multi-part collections of these types.

```
{  
  "type": "FeatureCollection",  
  "features": [  
    { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.97081750370484, 37.55506788676746 ] },  
      "properties": { "name": "서울역" } },  
    { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.97714751697998, 37.56547786297515 ] },  
      "properties": { "name": "시청역" } },  
    { "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.98303263797399, 37.57004847771681 ] },  
      "properties": { "name": "종각역" } },  
  ]  
}
```

1. Google Maps API 가이드

■ Google Maps Data Layer



```
{ "type": "Feature", "geometry": { "type": "Point", "coordinates": [ 126.97081750370484, 37.55506788676746 ] },  
  "properties": { "name": "서울역" } },
```

1. Google Maps API 가이드

■ Styling data(tutorials/google-maps/index7.html)

```
<script>
  function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });

    // GeoJSON format data
    var geodata = { ... };

    map.data.addGeoJson(geodata);
    map.data.setStyle(function(feature){
      return {
        icon: {
          url: "/icons/subway-24px.svg",
          anchor: { x: 12, y: 12 },
          labelOrigin: { x:12, y: 30 },
        },
        label: {
          color: "#FF0000",
          text: feature.getProperty("name"),
        },
        title: feature.getProperty("name"),    // rollover text
      };
    });
  }
</script>
```

1. Google Maps API 가이드

■ 마우스 이벤트(tutorials/google-maps/index8.html)

```
<script>
  function initMap() {
    var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });

    // GeoJSON format data
    var geodata = { ... };

    map.data.addGeoJson(geodata);

    const infowindow = new google.maps.InfoWindow();
    map.data.addListener('click', function(e){
      const feature = e.feature;
      const name = feature.getProperty("name");
      const position = e.latLng;

      infowindow.setPosition(position);
      infowindow.setContent("<div align=center><img src=/icons/subway-24px.svg /><br><b>" +
                           + name + "</b><br>(" + position.lng() + "," + position.lat() + ")" + "</div>");
      infowindow.open(map);
    });
  }
</script>
```

1. Google Maps API 가이드

- idle 이벤트(tutorials/google-maps/index9.html)

```
<script>
function initMap() {
  var map = new google.maps.Map(document.getElementById('map'), { center: { lng: 126.9706673, lat: 37.5547787 }, zoom: 15 });
  var geodata = { ... };

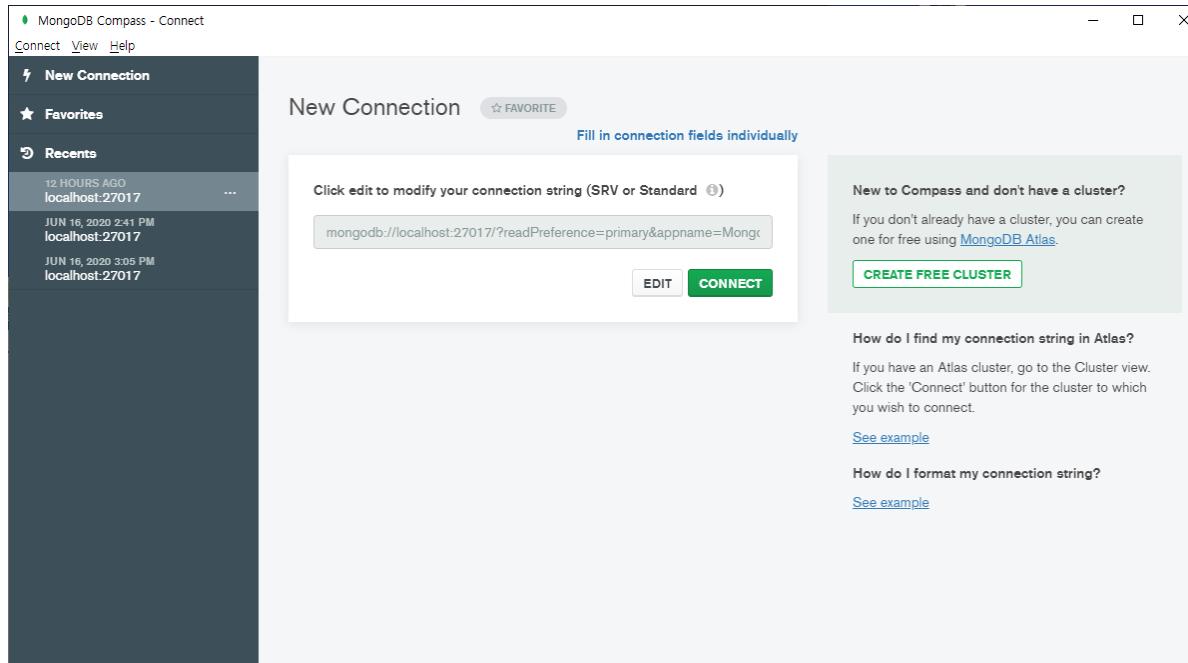
  map.addListener('idle', function(){
    // remove all features
    map.data.forEach(function(feature){
      map.data.remove(feature);
    });

    var count = 0;
    const bounds = map.getBounds();
    const features = geodata.features;
    for(var i = 0; i < features.length; i++){
      const coordinates = features[i].geometry.coordinates;
      if (bounds.contains(new google.maps.LatLng({ lng: coordinates[0], lat: coordinates[1] }))){
        map.data.addGeoJson(features[i]); count++;
      }
    }
    console.log(count + " data added!!!");
  });
}
</script>
```

2. MongoDB 가이드

- MongoDB compass (GUI client)

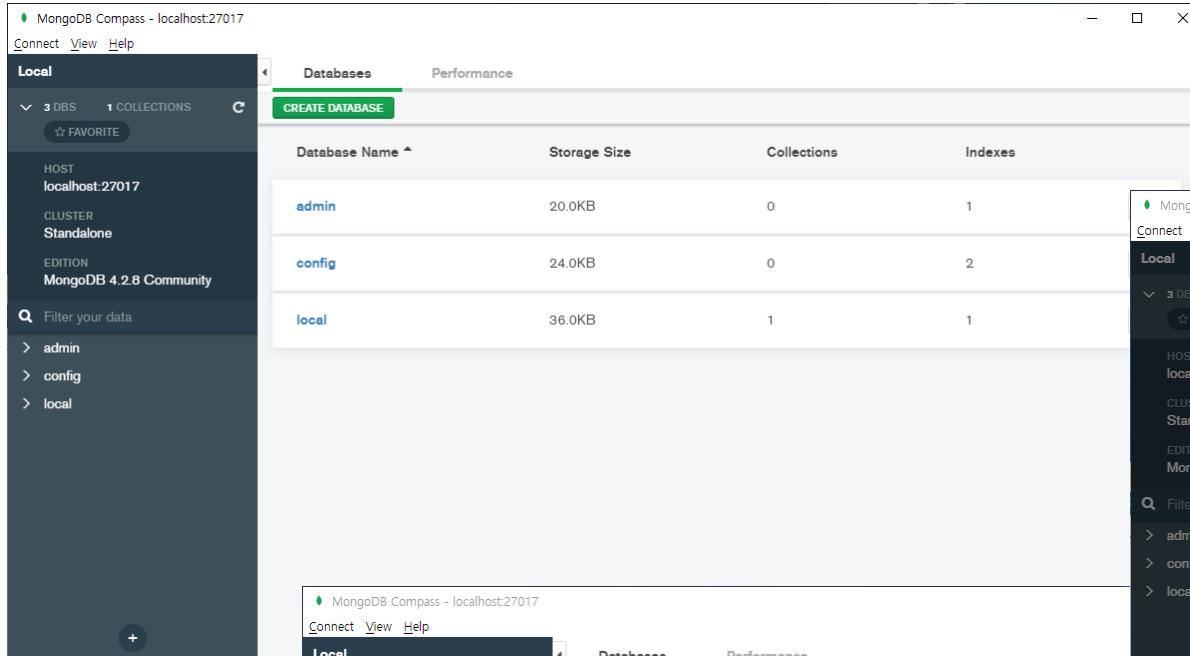
- MongoDB를 설치하면 서버가 실행된다. (`mongodb://localhost:27017`)
- CONNECT를 눌러서 서버에 연결한다.



`mongodb://localhost:27017` 가 디폴트이므로 입력하지 않고 CONNECT를 눌러도 된다.

2. MongoDB 가이드

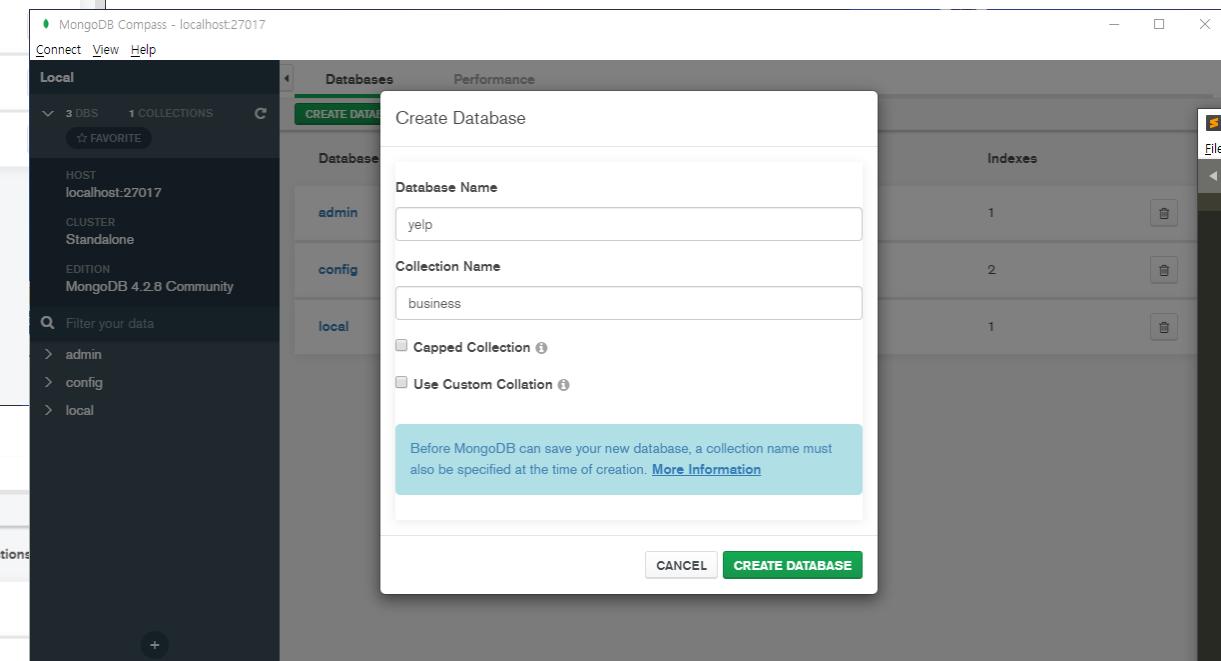
- yelp DB와 business collection을 생성한다.



The screenshot shows the MongoDB Compass interface with the 'Databases' tab selected. On the left, the sidebar shows 'Local' with 'HOST localhost:27017', 'CLUSTER Standalone', and 'EDITION MongoDB 4.2.8 Community'. Below the sidebar, there are three database entries: 'admin' (Storage Size 20.0KB, Collections 0, Indexes 1), 'config' (Storage Size 24.0KB, Collections 0, Indexes 2), and 'local' (Storage Size 36.0KB, Collections 1, Indexes 1). A green 'CREATE DATABASE' button is visible at the top right of the main panel.

1. CREATE DATABASE 버튼을 누른다.

2. DATABASE 이름(yelp)과 Collection 이름(business)를 입력/생성한다.



The screenshot shows the 'Create Database' dialog box from MongoDB Compass. It has fields for 'Database Name' (set to 'yelp') and 'Collection Name' (set to 'business'). There are two unchecked checkboxes: 'Capped Collection' and 'Use Custom Collection'. A note at the bottom states: 'Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)'. At the bottom right are 'CANCEL' and 'CREATE DATABASE' buttons.

3. yelp DB가 생성되었음을 확인할 수 있다.

2. MongoDB 가이드

- business collection에 business.json을 로드한다.

This collection has no data
It only takes a few seconds to import data from a JSON or CSV file
Import Data

1. Import Data 버튼을 누른다.

2. json 파일을 선택하고, IMPORT 버튼을 누른다.

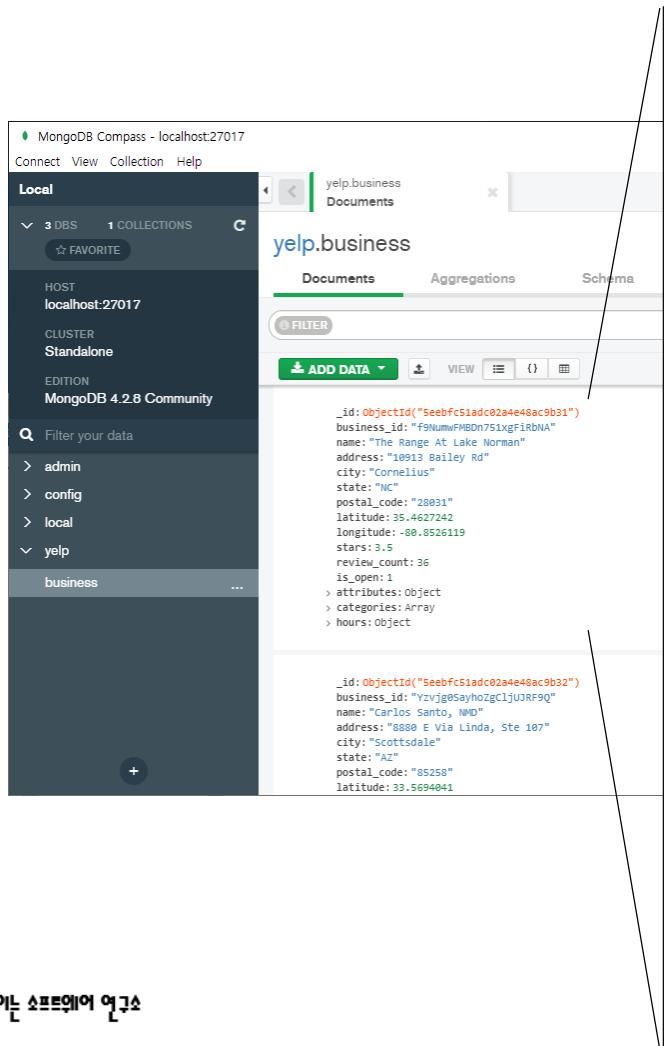
3. business.json을 로드한 결과

```
_id:ObjectId("Seebfc5iad02a4e48ac9b31")
business_id:"f9numPMBn751xgFirBNA"
name:"The Range At Lake Norman"
address:"10913 Bailey Rd"
city:"Cornelius"
state:"NC"
postal_code:"28031"
latitude:35.4627242
longitude:-80.8526119
stars:3.5
review_count:36
is_open:1
> attributes:Object
> categories:Array
> hours:Object

_id:ObjectId("Seebfc5iad02a4e48ac9b32")
business_id:"Yzvjg0sayhozgClju3RFQ"
name:"Carlos Santo, MD"
address:"8880 E Via Linda, Ste 107"
city:"Scottsdale"
state:"AZ"
postal_code:"85258"
latitude:33.5694041
```

2. MongoDB 가이드

■ business collection



The screenshot shows the MongoDB Compass interface connected to 'localhost:27017'. The left sidebar lists databases (admin, config, local, yelp) and collections (business). The main area shows the 'yelp.business' collection with two documents listed. The top document is expanded to show its full JSON structure.

```
_id: ObjectId("5eebf51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
attributes: Object
  BusinessAcceptsCreditCards: "True"
  BikeParking: "True"
  GoodForKids: "False"
  BusinessParking: "{ 'garage': False, 'street': False, 'validated': False, 'lot': True, 'v...'}"
  ByAppointmentOnly: "False"
  RestaurantsPriceRange2: "3"
categories: Array
  0: "Active Life"
  1: "Gun/Rifle Ranges"
  2: "Guns & Ammo"
  3: "Shopping"
hours: Object
```

2. MongoDB 가이드

■ business collection 검색하기

The screenshot shows the MongoDB Compass interface for the 'yelp.business' collection. At the top, it displays statistics: DOCUMENTS 209.4k, TOTAL SIZE 124.3MB, AVG. SIZE 622B, INDEXES 1, TOTAL SIZE 1.8MB, and AVG. SIZE 1.8MB. Below this, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. The 'Documents' tab is selected.

In the search bar, the query `{ name: "The Range At Lake Norman" }` is entered. Below the search bar, there are buttons for 'OPTIONS', 'FIND', 'RESET', and three dots. The results section shows one document with the following fields:

```
_id: ObjectId("5eefc51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
> attributes: Object
> categories: Array
> hours: Object
```

Below the document details, there are icons for edit, delete, copy, and refresh. The status message 'Displaying documents 1 - 1 of 1' is shown at the bottom of the results area.

2. MongoDB 가이드

■ Text search (<https://docs.mongodb.com/manual/text-search/>)

- text index를 생성한다. db.business.createIndex({ name: "text", categories: "text" });

```
D:\#>mongo
MongoDB shell version v4.2.8
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b3d6a87d-0a93-49c0-a483-9b3d5afb3aba") }
MongoDB server version: 4.2.8
Server has startup warnings:
2020-06-16T14:38:04.436+0900 | CONTROL  [initandlisten]
2020-06-16T14:38:04.437+0900 | CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-06-16T14:38:04.437+0900 | CONTROL  [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-06-16T14:38:04.437+0900 | CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> use yelp
switched to db yelp
> db.business.createIndex( { name: "text", categories: "text" } );
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
```

2. MongoDB 가이드

■ Text search (<https://docs.mongodb.com/manual/text-search/>)

- text index를 생성한다. db.business.createIndex({ name: "text", categories: "text" });

The screenshot shows the MongoDB Compass interface connected to a local host at port 27017. The database selected is 'yelp'. In the 'Indexes' tab of the 'yelp.business' collection, a new index is being created. The index is named 'name_text_categories_text' and is defined with the fields '_fts' and 'text' under the 'fts' field type, and '_ftsx' under the 'ftsx' field type. The 'Type' is set to 'TEXT'. The index has a size of 12.5 MB and is marked as 'COMPOUND'. A red box highlights this index entry. The interface also shows other existing indexes: '_id_ (REGULAR, 1.9 MB, UNIQUE)' and '_fts (text, _ftsx) (TEXT, 12.5 MB, COMPOUND)'. The overall document count is 209.4k, and there are 2 indexes.

TEXT Type의 인덱스가
만들어졌다.

2. MongoDB 가이드

■ Text search (<https://docs.mongodb.com/manual/text-search/>)

- text index를 생성한다. db.business.createIndex({ name: "text", categories: "text" });
- text index를 사용하여 검색한다. db.business.find({ \$text : { \$search : "Norman" } });

The screenshot shows the MongoDB Compass interface connected to a local database at localhost:27017. The collection selected is 'yelp.business'. The 'Documents' tab is active, displaying 209.4k documents. A red box highlights the search query in the filter bar: { \$text : { \$search : "Norman" } }. Below the table, a document is expanded to show its fields, with 'name' also highlighted in red.

_id	business_id	name	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open
ObjectId("5eebfcc86adc02a4e48afa0c7")	"p37gJHffidefQwEvXlhfbQ"	"Merle Norman"	"2887 N Green Valley Pkwy"	"Henderson"	"NV"	"89014"	36.071284	-115.083303	0	23	0

2. MongoDB 가이드

- Query an Array (<https://docs.mongodb.com/manual/tutorial/query-arrays/>)

```
_id: ObjectId("5eebfcc51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
attributes: Object
  BusinessAcceptsCreditCards: "True"
  BikeParking: "True"
  GoodForKids: "False"
  BusinessParking: {"garage": False, 'street': False, 'validated': False, 'lot': True}
  ByAppointmentOnly: "False"
  RestaurantsPriceRange2: "3"
categories: Array
  0: "Active Life"
  1: "Gun/Rifle Ranges"
  2: "Guns & Ammo"
  3: "Shopping"
hours: Object
```

```
db.business.find({ categories: ["Shopping"] });
```

The screenshot shows the MongoDB Compass interface. The top navigation bar includes 'MongoDB Compass - localhost:27017/yelp.business', 'Connect', 'View', 'Collection', and 'Help'. Below the navigation is a sidebar with 'Local' selected, showing '3 DBS' and '1 COLLECTIONS'. Under 'yelp' is the 'business' collection. The main area is titled 'yelp.business' with tabs for 'Documents', 'Aggregations', and 'Schema'. A filter bar at the bottom of the main area contains the query '{ categories: ["Shopping"] }'. The results table shows one document:

Document
<pre>_id: ObjectId("5eebfcc52adc02a4e48aca3fd") business_id: "XUWTq3w8_sPtqsg9wKZ8kw" name: "The Home Goods Store" address: "Rea Road At Blakeney Shopping Mall" city: "Charlotte" state: "NC" postal_code: "28263" latitude: 35.036134 longitude: -80.807372 stars: 0 review_count: 4 is_open: 1 attributes: Object categories: Array 0: "Shopping"</pre>

2. MongoDB 가이드

- Query an Array (<https://docs.mongodb.com/manual/tutorial/query-arrays/>)

```
_id: ObjectId("5eebf51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
attributes: Object
  BusinessAcceptsCreditCards: "True"
  BikeParking: "True"
  GoodForKids: "False"
  BusinessParking: "{ 'garage': False, 'street': False, 'validated': False, 'lot': True }"
  ByAppointmentOnly: "False"
  RestaurantsPriceRange2: "3"
categories: Array
  0: "Active Life"
  1: "Gun/Rifle Ranges"
  2: "Guns & Ammo"
  3: "Shopping"
hours: Object
```

```
db.business.find({ categories: "Shopping" } );
```

The screenshot shows the MongoDB Compass interface. On the left, the database structure is visible with 3 DBs and 1 Collection. The 'yelp.business' database is selected. The 'business' collection is highlighted. A filter bar at the top right shows the query: { categories: "Shopping" }. The results pane displays the document from the code block above.

```
_id: ObjectId("5eebf51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
attributes: Object
  BusinessAcceptsCreditCards: "True"
  BikeParking: "True"
  GoodForKids: "False"
  BusinessParking: "{ 'garage': False, 'street': False, 'validated': False, 'lot': True }"
  ByAppointmentOnly: "False"
  RestaurantsPriceRange2: "3"
categories: Array
  0: "Active Life"
  1: "Gun/Rifle Ranges"
  2: "Guns & Ammo"
  3: "Shopping"
hours: Object
```

2. MongoDB 가이드

- Query an Array (<https://docs.mongodb.com/manual/tutorial/query-arrays/>)

```
_id: ObjectId("5eebf51adc02a4e48ac9b31")
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
address: "10913 Bailey Rd"
city: "Cornelius"
state: "NC"
postal_code: "28031"
latitude: 35.4627242
longitude: -80.8526119
stars: 3.5
review_count: 36
is_open: 1
attributes: Object
  BusinessAcceptsCreditCards: "True"
  BikeParking: "True"
  GoodForKids: "False"
  BusinessParking: {"garage": False, 'street': False, 'validated': False, 'lot': True}
  ByAppointmentOnly: "False"
  RestaurantsPriceRange2: "3"
categories: Array
  0: "Active Life"
  1: "Gun/Rifle Ranges"
  2: "Guns & Ammo"
  3: "Shopping"
hours: Object
```

```
db.business.find({ categories: { $all: ["Korean", "Restaurants"] } });
```

The screenshot shows the MongoDB Compass interface with the 'yelp.business' collection selected. The 'Documents' tab is active. A red box highlights the query text: `db.business.find({ categories: { $all: ["Korean", "Restaurants"] } })`. Below the query, the results are displayed as a list of documents. One document is shown in full:

```
_id: ObjectId("5eebf51adc02a4e48ac9c46")
business_id: "d9A5lhFHOBvEw-BxSYnWOA"
name: "Song Cook's Corean Chilli"
address: "681 Bloor St W"
city: "Toronto"
state: "ON"
postal_code: "M6G 1L3"
latitude: 43.6637457
longitude: -79.4164519
stars: 0
review_count: 4
is_open: 0
attributes: Object
  RestaurantsPriceRange2: "1"
  BusinessParking: {"garage": False, 'street': False, 'validated': False, 'lot': True}
  GoodForKids: "True"
  BikeParking: "True"
categories: Array
  0: "Restaurants"
  1: "Korean"
```

2. MongoDB 가이드

- Query an Array (<https://docs.mongodb.com/manual/tutorial/query-arrays/>)

```
db.business.find({ categories: { $all: ["Korean", "Restaurants"] } }).sort( { stars: -1 } ).limit(10);
```

The screenshot shows the MongoDB Compass interface for the `yelp.business` collection. The `Documents` tab is selected. At the top, it displays `DOCUMENTS 209.4k`, `TOTAL SIZE 124.3MB`, and `AVG. SIZE 622B`. To the right, it shows `INDEXES 2`, `TOTAL SIZE 13.8MB`, and `AVG. SIZE 6.9MB`.

The query builder section contains the following filters:

- `FILTER { categories: { $all: ["Korean", "Restaurants"] } }`
- `PROJECT`
- `SORT { stars: -1 }`
- `COLLATION`

Below the filters, there are buttons for `OPTIONS`, `FIND`, `RESET`, and three dots. On the right side of the query builder, there are buttons for `MAXTIMEMS 5000`, `SKIP 0`, and `LIMIT 10`.

At the bottom left, there are buttons for `ADD DATA`, `VIEW`, and three icons. At the bottom right, it says `Displaying documents 1 - 10 of 10` with navigation arrows, and a `REFRESH` button.

The document list below shows the first document in the results:

```
_id: ObjectId("5eefc51adc02a4e48ac9e1b")
business_id: "7HbKKqXtzUjf6uvSHZ8wyw"
name: "Let's Meat Kbbq"
address: "1400 S Church St, Ste B"
city: "Charlotte"
state: "NC"
postal_code: "28203"
latitude: 35.2190478215
longitude: -80.8576791734
stars: 4.5
review_count: 354
is_open: 1
> attributes: Object
< categories: Array
  0: "Restaurants"
  1: "Barbeque"
  2: "Korean"
  3: "Buffets"
```

2. MongoDB 가이드

■ 위치 기반 검색(<https://docs.mongodb.com/manual/geospatial-queries/>)

To specify GeoJSON data, use an embedded document with:

- a field named **type** that specifies the GeoJSON object type and
 - a field named **coordinates** that specifies the object's coordinates.
- If specifying latitude and longitude coordinates, list the **longitude** first and then **latitude**:
- Valid longitude values are between **-180** and **180**, both inclusive.
 - Valid latitude values are between **-90** and **90**, both inclusive.

```
<field>: { type: <GeoJSON type> , coordinates: <coordinates> }
```

For example, to specify a GeoJSON Point:

```
location: {  
    type: "Point",  
    coordinates: [-73.856077, 40.848447]  
}
```

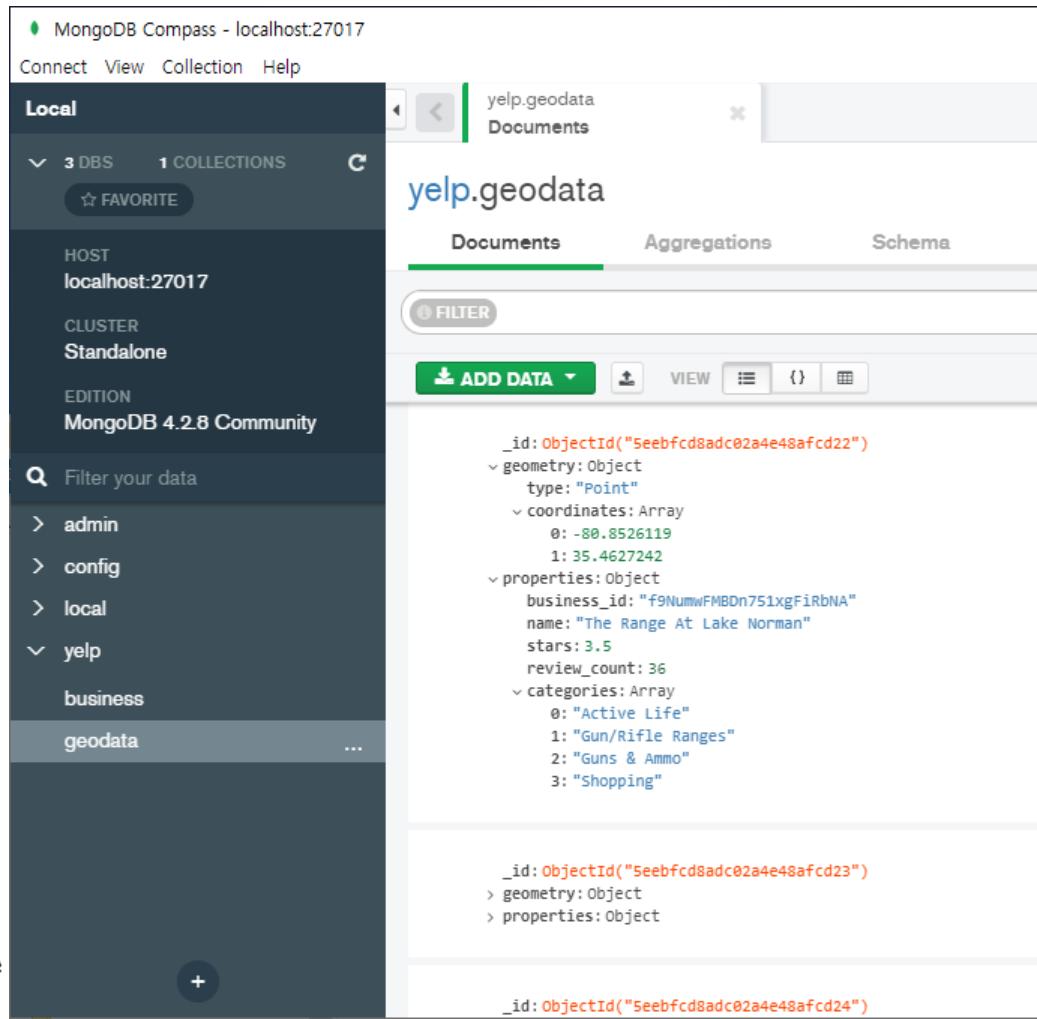
2. MongoDB 가이드

■ geodata collection 만들기

```
db.business.find().forEach(function(biz){  
    db.geodata.insert({ "geometry": { "type": "Point", "coordinates": [ biz.longitude, biz.latitude ] },  
        "properties": { "business_id": biz.business_id,  
            "name": biz.name,  
            "stars": biz.stars,  
            "review_count": biz.review_count,  
            "categories": biz.categories } });  
});
```

2. MongoDB 가이드

■ geodata collection 만들기



MongoDB Compass - localhost:27017

Connect View Collection Help

Local

3 DBS 1 COLLECTIONS

FAVORITE

HOST localhost:27017

CLUSTER Standalone

EDITION MongoDB 4.2.8 Community

Filter your data

admin

config

local

yelp

business

geodata

yelp.geodata

Documents Aggregations Schema

ADD DATA FILTER VIEW

`_id: ObjectId("5eebfcd8adc02a4e48afcd22")
geometry: Object
type: "Point"
coordinates: Array
0: -80.8526119
1: 35.4627242
properties: Object
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
stars: 3.5
review_count: 36
categories: Array
0: "Active Life"
1: "Gun/Rifle Ranges"
2: "Guns & Ammo"
3: "Shopping"`

`_id: ObjectId("5eebfcd8adc02a4e48afcd23")
geometry: Object
properties: Object`

`_id: ObjectId("5eebfcd8adc02a4e48afcd24")`

```
_id: ObjectId("5eebfcd8adc02a4e48afcd22")
geometry: Object
type: "Point"
coordinates: Array
0: -80.8526119
1: 35.4627242
properties: Object
business_id: "f9NumwFMBDn751xgFiRbNA"
name: "The Range At Lake Norman"
stars: 3.5
review_count: 36
categories: Array
0: "Active Life"
1: "Gun/Rifle Ranges"
2: "Guns & Ammo"
3: "Shopping"
```

2. MongoDB 가이드

■ 2dsphere 인덱스 생성하기

The screenshot illustrates the process of creating a 2dsphere index in MongoDB. It shows two main windows: a 'Create Index' dialog box and a 'yelp.geodata' collection details page.

Create Index Dialog:

- Name and Definition:** The 'geometry' field is selected as the index key type, and '2dsphere' is chosen as the index type.
- ADD ANOTHER FIELD:** A button to add more fields to the index.
- CREATE INDEX:** A green button to finalize the index creation.

yelp.geodata Collection Details:

- Documents:** 209.4k documents.
- Indexes:** 2 indexes.
- Schema:** Shows the '_id' field as a regular index.
- Indexes:** Shows the 'geometry' field as a geospatial index (2dsphere).

A red arrow points from the 'CREATE INDEX' button in the dialog to the newly created 'geometry' index entry in the collection's 'Indexes' table.

2. MongoDB 가이드

- 위치 기반 검색(<https://docs.mongodb.com/manual/geospatial-queries/>)

```
db.geodata.find({ geometry: { $geoWithin: { $box: [ [ -80.5, 35], [ -80., 35.5] ] } } });
```

The screenshot shows the MongoDB Compass interface with the database 'yelp' selected. The 'geodata' collection is open. The 'Documents' tab is active. At the top, it displays 'DOCUMENTS 209.4k', 'TOTAL SIZE 62.5MB', 'AVG. SIZE 313B', 'INDEXES 2', 'TOTAL SIZE 4.3MB', and 'AVG. SIZE 2.2MB'. Below this, there are tabs for 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. A 'FILTER' bar contains the query: '{ geometry: { \$geoWithin: { \$box: [[-80.5, 35], [-80., 35.5]] } } }'. Buttons for 'OPTIONS', 'FIND', 'RESET', and three dots are available. Below the filter, there are buttons for 'ADD DATA', 'VIEW', and document navigation. The main area shows the first 20 documents of 23 found, displaying their _id, geometry (a point at approximately -80.46, 35.26), properties (business_id: T7MRk8fxSBJD_c5qhHDe7w, name: "Rocky River Vineyards", stars: 4.5, review_count: 34), and categories (Venues & Event Spaces, Local Flavor, Food, Event Planning & Services, Wine Tasting Room, Arts & Entertainment, Party & Event Planning, Wineries). On the right, there are edit, delete, and copy icons for each document.

2. MongoDB 가이드

- 위치 기반 검색(<https://docs.mongodb.com/manual/geospatial-queries/>)

```
db.geodata.find({ geometry: { $near: { $geometry: { type: "Point", coordinates: [-80.8, 35.5] }, $minDistance: 0, $maxDistance: 1000 } }});
```

The screenshot shows the MongoDB Compass interface with the database 'yelp' selected and the collection 'geodata'. The 'Documents' tab is active. At the top, it displays 'DOCUMENTS 209.4k' and 'TOTAL SIZE 62.5MB'. Below the header, there's a search bar with the query '{geometry:{\$near:{\$geometry:{ type:"Point", coordinates: [-80.8, 35.5] }, \$maxDistance: 1000 }}}'. To the right of the search bar are buttons for 'OPTIONS', 'FIND', 'RESET', and three dots. The main area shows two document results:

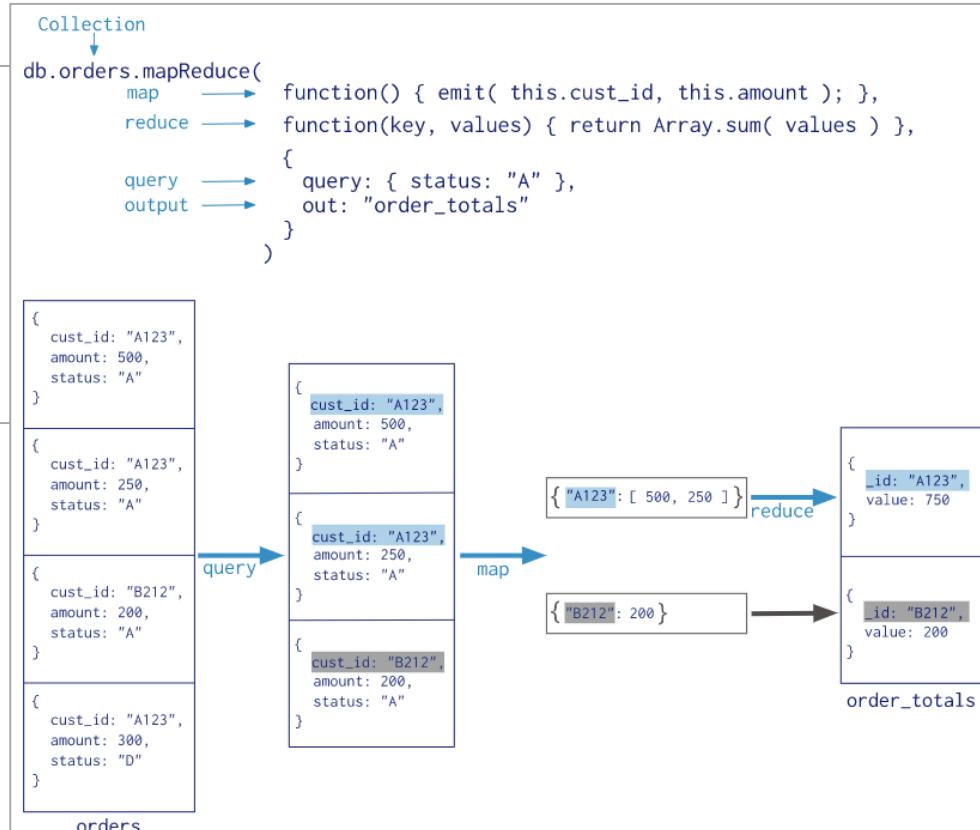
```
_id: ObjectId("5eebfce0adc02a4e48b0848e")
geometry: Object
  type: "Point"
  coordinates: Array
    0: -80.7960876
    1: 35.4949288
properties: Object

_id: ObjectId("5eebfccf0adc02a4e48b20d16")
geometry: Object
  type: "Point"
  coordinates: Array
    0: -80.7973093
    1: 35.5079098
properties: Object
```

2. MongoDB 가이드

- CRUD Operations (<https://docs.mongodb.com/manual/crud/>)
- Query Operators (<https://docs.mongodb.com/manual/reference/operator/query/>)
- Aggregation (<https://docs.mongodb.com/manual/aggregation/>)

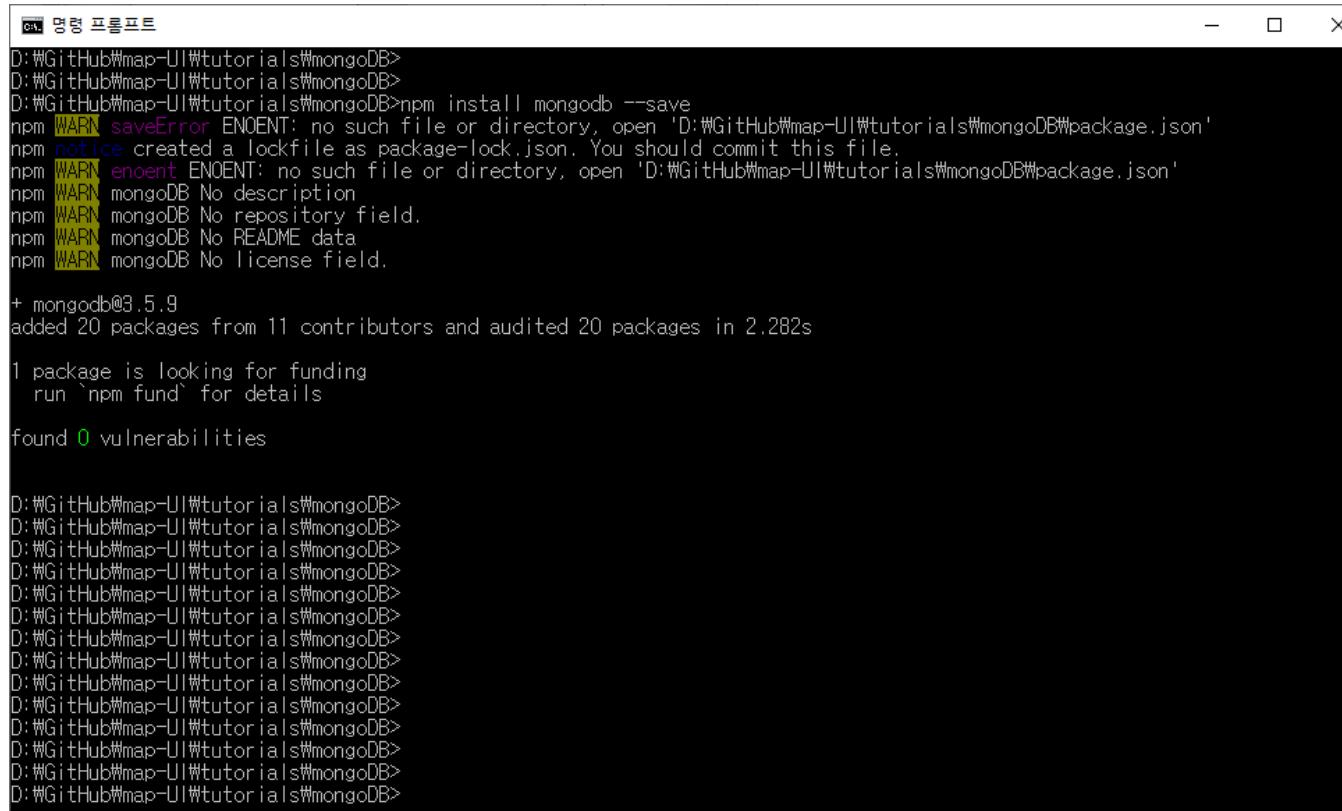
```
db.orders.aggregate([
  { $match: { status: "A" } },
  { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
```



3. MongoDB + Node.js 가이드

▪ MongoDB + Node.js 가이드

(<http://mongodb.github.io/node-mongodb-native/3.4/quick-start/quick-start/>)



```
D:\#GitHub\map-UI\tutorials\mongoDB>
D:\#GitHub\map-UI\tutorials\mongoDB>
D:\#GitHub\map-UI\tutorials\mongoDB>npm install mongodb --save
npm WARN saveError ENOENT: no such file or directory, open 'D:\#GitHub\map-UI\tutorials\mongoDB\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file;
npm warn enoent ENOENT: no such file or directory, open 'D:\#GitHub\map-UI\tutorials\mongoDB\package.json'
npm warn mongoDB No description
npm warn mongoDB No repository field.
npm warn mongoDB No README data
npm warn mongoDB No license field.

+ mongodb@3.5.9
added 20 packages from 11 contributors and audited 20 packages in 2.282s
1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\#GitHub\map-UI\tutorials\mongoDB>
```

D:\#> npm install mongodb --save

package-lock.json
파일이 생성됨.

3. MongoDB + Node.js 가이드

■ 간단한 Query(tutorials/mongoDB/db1.js)

```
const MongoClient = require('mongodb').MongoClient;

const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);

client.connect(function(err) {
  console.log("Connected successfully to server");

  const db = client.db('yelp');
  const collection = db.collection('business');

  collection.find({ categories: { $all: ["Korean", "Restaurants"] } }).sort({ stars: -1 }).limit(5).toArray(function(err, docs){
    console.log("Found the following records");
    console.log(docs);

    client.close();
  });
});
```

3. MongoDB + Node.js 가이드

■ 간단한 Query(tutorials/mongoDB/db2.js)

```
const MongoClient = require('mongodb').MongoClient;

const url = 'mongodb://localhost:27017';
const client = new MongoClient(url);

client.connect(function(err) {
  console.log("Connected successfully to server");

  const db = client.db('yelp');
  const collection = db.collection('geodata');

  collection.find({ geometry: { $near: { $geometry: { type: "Point", coordinates: [-80.8, 35.5] },
                                             $minDistance: 0, $maxDistance: 1000 } }}).toArray(function(err, docs){
    console.log("Found the following records");
    console.log(docs);

    client.close();
  });
});
```

3. MongoDB + Node.js 가이드

■ Node.js web server(tutorials/mongoDB/web.js)

```
var http = require('http')

var app = http.createServer(function(request, response){
    response.writeHead(200, { 'Content-Type': 'text/plain' })
    response.write('Hello World!!!!')
    response.end()
})

app.listen(8080)
```

3. MongoDB + Node.js 가이드

■ Node.js web server(tutorials/mongoDB/db3.js)

```
const MongoClient = require('mongodb').MongoClient
const client = new MongoClient('mongodb://localhost:27017')

client.connect(function(err) {
  console.log("Connected successfully to DB server")

  const db = client.db('yelp')
  const collection = db.collection('geodata')

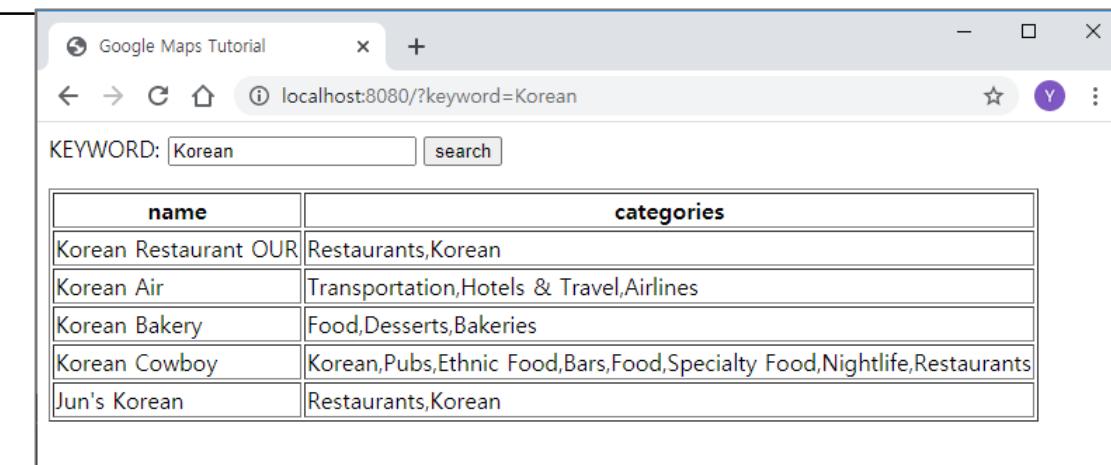
  require('http').createServer(function(request, response){
    collection.find({ geometry: { $near: { $geometry: { type: 'Point', coordinates: [-80.8, 35.5] },
      $minDistance: 0, $maxDistance: 1000 } }}).toArray(function(err, docs){
      response.writeHead(200, { 'Content-Type': 'text/plain' })
      response.write(JSON.stringify(docs))
      response.end()
    });
  }).listen(8080)
});
```

3. MongoDB + Node.js 가이드

■ Node.js web server(tutorials/mongoDB/index.html)

```
<html>
  <head>
    <title>Google Maps Tutorial</title>
  </head>
  <body>
    <form action="/">
      KEYWORD: <input type="text" name="keyword" />
      <input type="submit" value="search" />
    </form>

    <table border=1>
      <thead><tr><th>name</th><th>categories</th></tr></thead>
      <tr><td>Korean Restaurant OUR</td><td>Restaurants,Korean</td></tr>
      <tr><td>Korean Air</td><td>Transportation,Hotels & Travel,Airlines</td></tr>
      <tr><td>Korean Bakery</td><td>Food,Desserts,Bakeries</td></tr>
      <tr><td>Korean Cowboy</td><td>Korean,Pubs,Ethnic Food,Bars,Food,Specialty Food,Nightlife,Restaurants</td></tr>
      <tr><td>Jun's Korean</td><td>Restaurants,Korean</td></tr>
    </table>
  </body>
</html>
```



A screenshot of a web browser window titled "Google Maps Tutorial". The address bar shows "localhost:8080/?keyword=Korean". Below the address bar is a search form with a text input containing "Korean" and a "search" button. To the right of the form is a table with five rows of search results. The table has two columns: "name" and "categories".

name	categories
Korean Restaurant OUR	Restaurants,Korean
Korean Air	Transportation,Hotels & Travel,Airlines
Korean Bakery	Food,Desserts,Bakeries
Korean Cowboy	Korean,Pubs,Ethnic Food,Bars,Food,Specialty Food,Nightlife,Restaurants
Jun's Korean	Restaurants,Korean

3. MongoDB + Node.js 가이드

■ Node.js web server(tutorials/mongoDB/db4.js)

```
const MongoClient = require('mongodb').MongoClient
const client = new MongoClient('mongodb://localhost:27017')

client.connect(function(err) {
  const db = client.db('yelp')
  const collection = db.collection('geodata')

  require('http').createServer(function(request, response){
    var url = request.url
    if(url == '/favicon.ico'){
      return response.writeHead(404);
    }

    response.writeHead(200, { 'Content-Type': 'text/html' })
    response.write('
<html>
<head><title>Google Maps Tutorial</title></head>
<body>
<form action="/">
  KEYWORD: <input type="text" name="keyword">
  <input type="submit" value="search" />
</form>
')
  })
})
```

db.geodata.createIndex({ "properties.name": "text", "properties.category": "text" }) // text index를 생성함.

3. MongoDB + Node.js 가이드

▪ Node.js web server(tutorials/mongoDB/db4.js)

```
///?keyword=KEYWORD
var n = url.indexOf('?');
var keyword = (n < 0) ? '' : url.substr(n + 9)
console.log(keyword)

if (keyword != ''){
    collection.find({ $text: { $search: keyword }}).limit(5).toArray(function(err, docs){
        response.write('<table border=1>')
        response.write('<thead><tr><th>name</th><th>categories</th></tr></thead>')
        for(var i = 0; i < docs.length; i++){
            response.write('<tr><td>' + docs[i].properties.name + '</td>')
            response.write('<td>' + docs[i].properties.categories + '</td></tr>')
        }
        response.write('</table></body></html>')
        response.end()
    })
}
else {
    response.write('</body></html>')
    response.end()
}
}).listen(8080)
})
```

db.geodata.createIndex({ "properties.name": "text", "properties.category": "text" }) // text index를 생성함.

4. Express.js + Mongoose

- Express.js : 빠르고 개방적인 간결한 웹 프레임워크 (<https://expressjs.com>)

```
$ npm install express --save
```

```
const express = require('express')

const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening at http://localhost:${port}`))
```

4. Express.js + Mongoose

- Express.js : 빠르고 개방적인 간결한 웹 프레임워크 (<https://expressjs.com>)
라우팅 (<https://expressjs.com/ko/guide/routing.html>)

```
const express = require('express')

const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Main Page!'))

app.get('/search', (req, res) => res.send('Search page!!!'))
app.post('/search', (req, res) => res.send('Search result page!!!'))

app.listen(port, () => console.log(`Example app listening at http://localhost:${port}`))
```

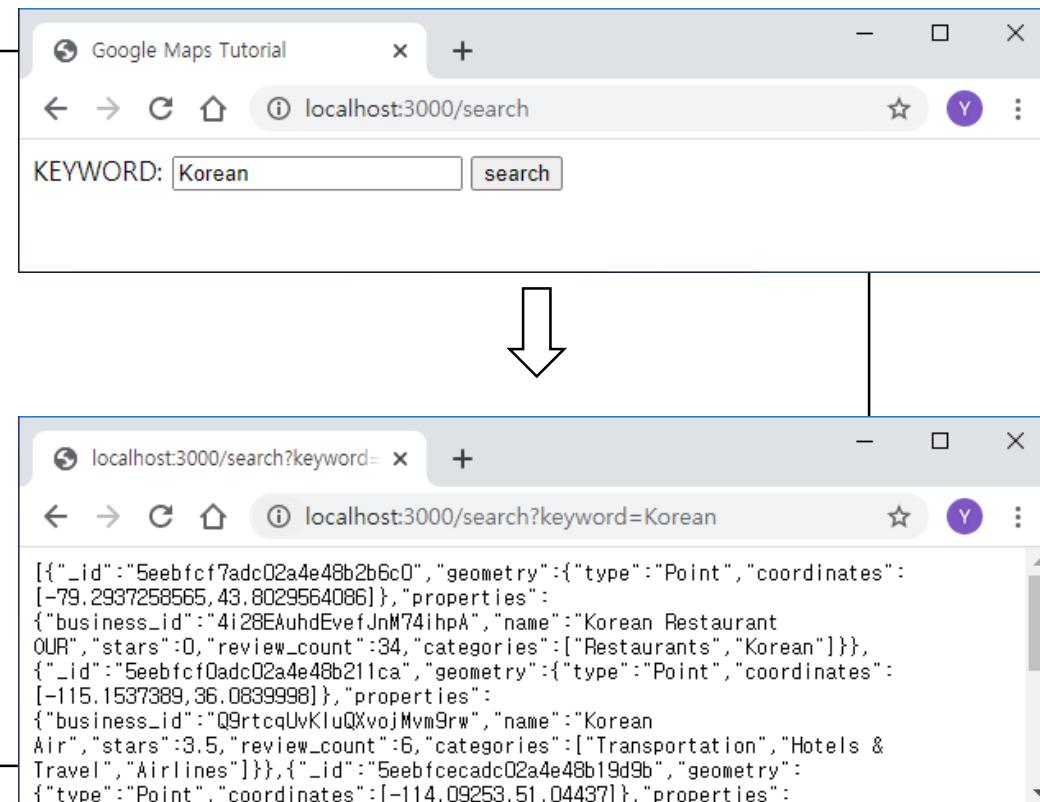
4. Express.js + Mongoose

- Express.js : 빠르고 개방적인 간결한 웹 프레임워크 (<https://expressjs.com>)
라우팅 (<https://expressjs.com/ko/guide/routing.html>)
- Express.js server(tutorials/express/db5.js)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/search', function(request, response){
  const keyword = request.query.keyword
  console.log(keyword)

  if (keyword == undefined){
    response.sendFile(__dirname + '/search.html')
  }
  else {
    collection.find({ $text: { $search: keyword }})
      .limit(5).toArray(function(err, docs){
        response.json(docs)
      })
  }
})
```



4. Express.js + Mongoose

- Express.js : 빠르고 개방적인 간결한 웹 프레임워크 (<https://expressjs.com>)
라우팅 (<https://expressjs.com/ko/guide/routing.html>)
라우터 모듈

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Birds home page!'))

var birds = require('./birds') ——————→
app.use('/birds', birds)

app.listen(port)
```

./birds.js

```
var express = require('express');
var router = express.Router();

// define the home page route
router.get('/', function(req, res) {
  res.send('Birds home page');
});

// define the about route
router.get('/about', function(req, res) {
  res.send('About birds');
});

module.exports = router;
```

4. Express.js + Mongoose

- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose1.js)

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true, useUnifiedTopology: true});

// 1) defining schema
const userSchema = new mongoose.Schema({
  name: String, age: Number,
});

// 2) creating model (collection)
const User = mongoose.model('users', userSchema);

// 3) instantiating document
const kim = new User({ name: 'Kim', age: 45 });
console.log(kim); // 'Silence'

// 4) updating document
kim.save();
```

4. Express.js + Mongoose

- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose2.js)

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true, useUnifiedTopology: true});

// 1) defining schema
const userSchema = new mongoose.Schema({
  name: String, age: Number,
});

// 2) creating model (collection)
const User = mongoose.model('users', userSchema);

// 3) finding document
const query = User.find().where('age').gt(40).lt(60);
query.exec(function(error, result){
  console.log(result.length);      // result is array
  result.forEach(function(user){
    console.log(user.name + '(' + user.age + ')');
  });
});
```

4. Express.js + Mongoose

- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose3.js)

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test', {useNewUrlParser: true, useUnifiedTopology: true});

// 1) defining schema
const userSchema = new mongoose.Schema({
  name: String, age: Number,
});

// 1-A) adding static
userSchema.statics.findByName = function(name){
  return this.find({name: name});
};

// 1-B) adding instance method
userSchema.methods.print = function(){
  console.log(this.name + '(' + this.age + ')');
};

// 1-C) adding query helper
userSchema.query.byName = function(name){
  return this.where({name: name});
};

// NOTE: they must be added to the schema before compiling it with mongoose.model()

// 2) creating model (collection)
const User = mongoose.model('users', userSchema);

// 3) finding document
User.find().byName('Kim').exec(function(error, result){
  result.forEach(function(user){
    user.print();
  })
})

User.find().byName('Kim').exec(function(error, result){
  result.forEach(function(user){
    user.print();
  })
})
```

4. Express.js + Mongoose

- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose4.js, models/user.js)

mongoose4.js

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test',
    {useNewUrlParser: true, useUnifiedTopology: true});

const User = require('./models/user');

User.findByName('Kim').exec(function(error, result){
    result.forEach(function(user){
        user.print();
    })
})
```

models/user.js

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
});

userSchema.statics.findByName = function(name){
    return this.find({name: name});
};

userSchema.methods.print = function(){
    console.log(this.name + '(' + this.age + ')');
};

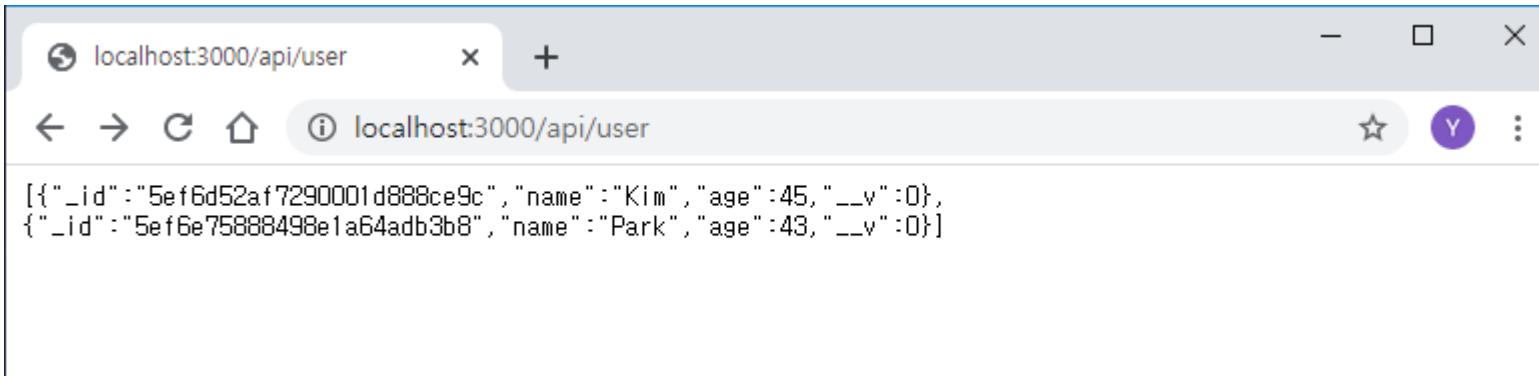
userSchema.query.byName = function(name){
    return this.where({name: name});
};

module.exports = mongoose.model('users', userSchema);
```

4. Express.js + Mongoose

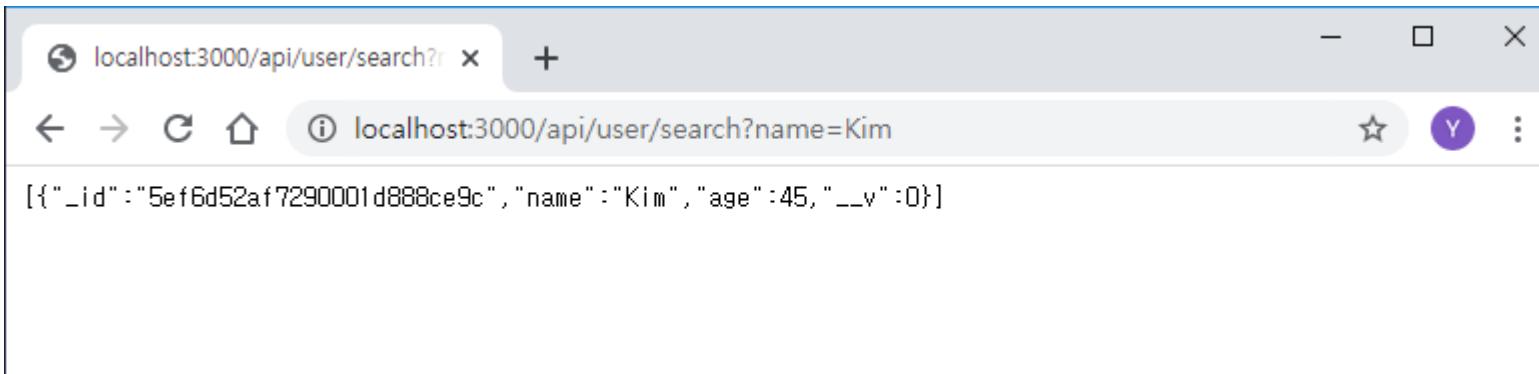
- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose5.js, models/user.js, api/user.js)

GET: /api/user



```
[{"_id": "5ef6d52af7290001d888ce9c", "name": "Kim", "age": 45, "__v": 0}, {"_id": "5ef6e75888498e1a64adb3b8", "name": "Park", "age": 43, "__v": 0}]
```

GET: /api/user/search?name=NAME



```
[{"_id": "5ef6d52af7290001d888ce9c", "name": "Kim", "age": 45, "__v": 0}]
```

4. Express.js + Mongoose

- mongoose : elegant MongoDB object modeling (<https://mongoosejs.com/>)
(tutorials/express/mongoose5.js, models/user.js, api/user.js)

mongoose5.js

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test',
    {useNewUrlParser: true, useUnifiedTopology: true});

const express = require('express');
const app = express();
const port = 3000;

app.use('/api/user', require('./api/user'));

app.listen(port);
```

api/user.js

```
const express = require('express');
const router = express.Router();

const User = require('../models/user');

router.get('/', function(request, response) {
    User.find(function(error, result){
        if (error) return response.status(500).json(error);
        return response.json(result);
    });
});

router.get('/search', function(request, response) {
    User.findByName(request.query.name).exec(function(error, result){
        if (error) return response.status(500).json(error);
        return response.json(result);
    });
});

module.exports = router;
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/models/business.js)

```
const mongoose = require('mongoose');

const businessSchema = new mongoose.Schema({
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    address: String,
    city: String,
    state: String,
    postal_code: String,
    latitude: Number,
    longitude: Number,
    stars: Number,
    review_count: Number,
    is_open: Boolean,
    attributes: Object,
    categories: [String],
    hours: {
        Monday: String,
        Tuesday: String,
        Wednesday: String,
        Thursday: String,
        Friday: String,
        Saturday: String,
        Sunday: String,
    }
},
{
    collection: 'business' // collection name
);
businessSchema.index({name: 'text', categories: 'text'});

businessSchema.statics.findById = function(ID){
    return this.find({business_id: ID});
};

businessSchema.statics.findByKeyword = function(keyword){
    return this.find({$text: {$search: keyword}});
};

module.exports = mongoose.model('business', businessSchema);
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/api/business.js)

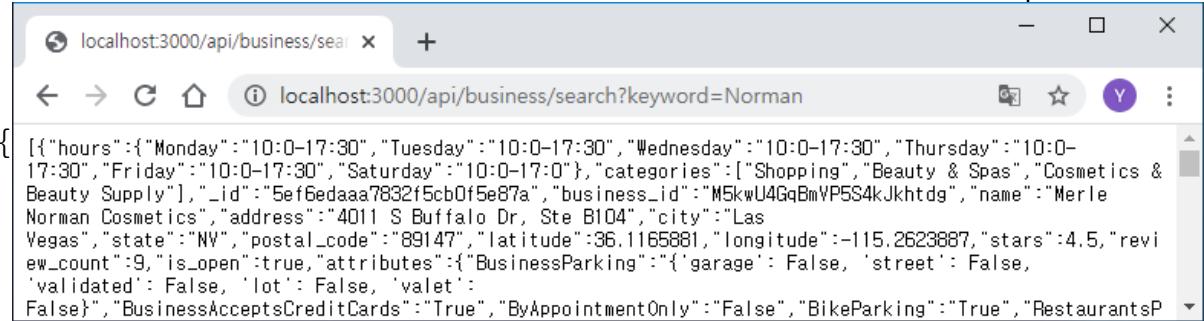
```
const express = require('express');
const router = express.Router();

const Business = require('../models/business' );

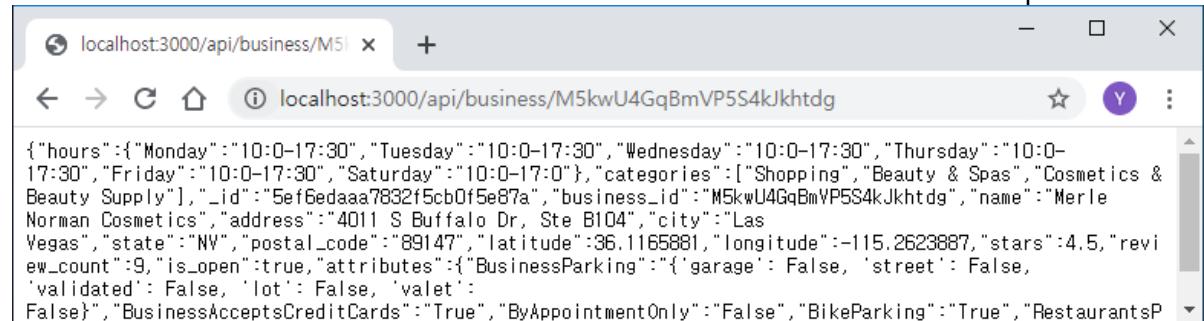
router.get('/search', function(request, response) {
  Business.findByKeyword(request.query.keyword)
    .sort({stars: -1}).limit(10).exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});

router.get('/:id', function(request, response) {
  Business.findById(request.params.id).exec(function(error, result){
    if (error) return response.status(500).json(error);
    return response.json(result[0]);
  });
}

module.exports = router;
```



A screenshot of a browser window displaying the URL `localhost:3000/api/business/search?keyword=Norman`. The page content shows a JSON array of business results, with the first item being a detailed object for "Merle Norman Cosmetics". The object includes fields like hours, categories, address, city, state, postal code, latitude, longitude, stars, review count, and various attributes related to parking and credit card acceptance.



A screenshot of a browser window displaying the URL `localhost:3000/api/business/M5kwU4GqBmVP5S4kJkhtdg`. The page content shows a single business object with the same detailed information as the search results, including hours, categories, address, city, state, postal code, latitude, longitude, stars, review count, and various attributes related to parking and credit card acceptance.

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/models/geodata.js)

```
const mongoose = require('mongoose');

const geodataSchema = new mongoose.Schema({
  geometry: {
    type: { type: String },
    coordinates: [Number]
  },
  properties: {
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    stars: Number,
    review_count: Number,
    categories: [String],
  }
},
{
  collection: 'geodata'          // collection name
}
);

mongoose.set('debug', true);
```

```
const mongoose = require('mongoose');
const geodataSchema = new mongoose.Schema({
  geometry: {
    type: String,
    coordinates: [Number]
  },

```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/models/geodata.js)

```
const mongoose = require('mongoose');

const geodataSchema = new mongoose.Schema({
  geometry: {
    type: { type: String },
    coordinates: [Number]
  },
  properties: {
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    stars: Number,
    review_count: Number,
    categories: [String],
  }
},
{
  collection: 'geodata'          // collection name
}
);
```

```
const mongoose = require('mongoose');

const geodataSchema = new mongoose.Schema({
  geometry: {
    type: String,
    coordinates: [Number]
  },
});
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/models/geodata.js)

```
const mongoose = require('mongoose');

const pointSchema = new mongoose.Schema({
  type: { type: String, enum: ['Point'], required: true },
  coordinates: { type: [Number], required: true }
});

const geodataSchema = new mongoose.Schema({
  geometry: pointSchema,
  properties: {
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    stars: Number,
    review_count: Number,
    categories: [String],
  },
  {
    collection: 'geodata' // collection name
  }
);

geodataSchema.index({ 'properties.name': 'text', 'properties.categories': 'text' });

geodataSchema.statics.findById = function(ID){
  return this.find().where('properties.business_id', ID);
};

geodataSchema.statics.findByKeyword = function(keyword){
  return this.find({$text: {$search: keyword}});
};
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/models/geodata.js)

```
const mongoose = require('mongoose');

const pointSchema = new mongoose.Schema({
  type: { type: String, enum: ['Point'], required: true },
  coordinates: { type: [Number], required: true }
});

const geodataSchema = new mongoose.Schema({
  geometry: pointSchema,
  properties: {
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    stars: Number,
    review_count: Number,
    categories: [String],
  },
  {
    collection: 'geodata' // collection name
  }
);

geodataSchema.index({geometry: '2dsphere'});

geodataSchema.statics.findInBox = function(lowerLeft, upperRight){
  return this.find().where('geometry').box(lowerLeft, upperRight);
};

geodataSchema.statics.findNear = function(center, maxDistance){
  return this.find().where('geometry')
    .near({center: center, maxDistance: maxDistance, spherical: true});
};

module.exports = mongoose.model('geodata', geodataSchema);
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/api/geodata.js)

```
const express = require('express');
const router = express.Router();

const geodata = require('../models/geodata');

router.get('/within', function(request, response) {
  geodata.findInBox([Number(request.query.left), Number(request.query.lower)],
    [Number(request.query.right), Number(request.query.upper)])
    .exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});

router.get('/near', function(request, response) {
  geodata.findNear([Number(request.query.longitude), Number(request.query.latitude)],
    Number(request.query.maxDistance))
    .limit(10).exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});
```

4. Express.js + Mongoose

■ Yelp Business(tutorials/express/api/geodata.js)

```
router.get('/search', function(request, response) {
  console.log(request.query.keyword);
  geodata.findByKeyword(request.query.keyword)
    .sort({stars: -1}).limit(10).exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});

router.get('/:id', function(request, response) {
  geodata.findById(request.params.id).exec(function(error, result){
    if (error) return response.status(500).json(error);
    return response.json(result[0]);
  });
}

module.exports = router;
```

4. Express.js + Mongoose

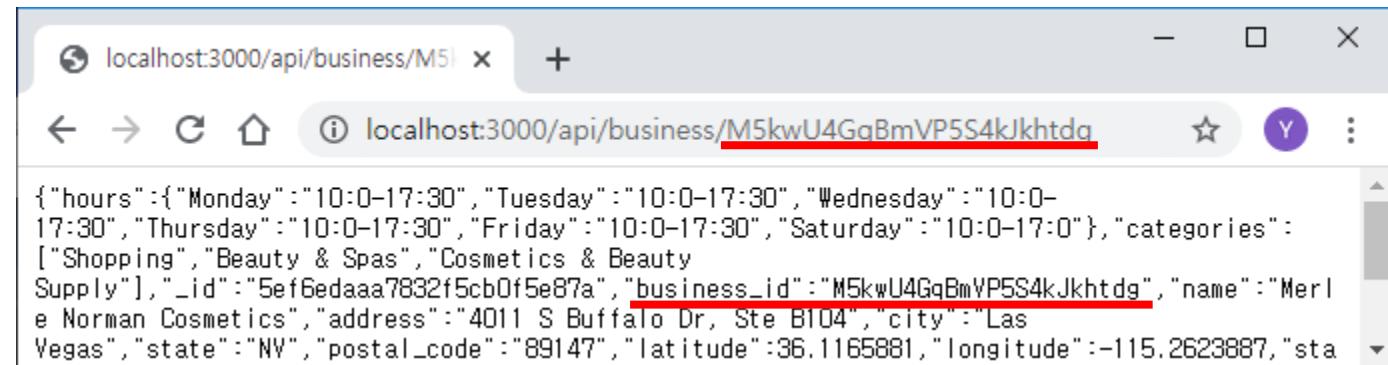
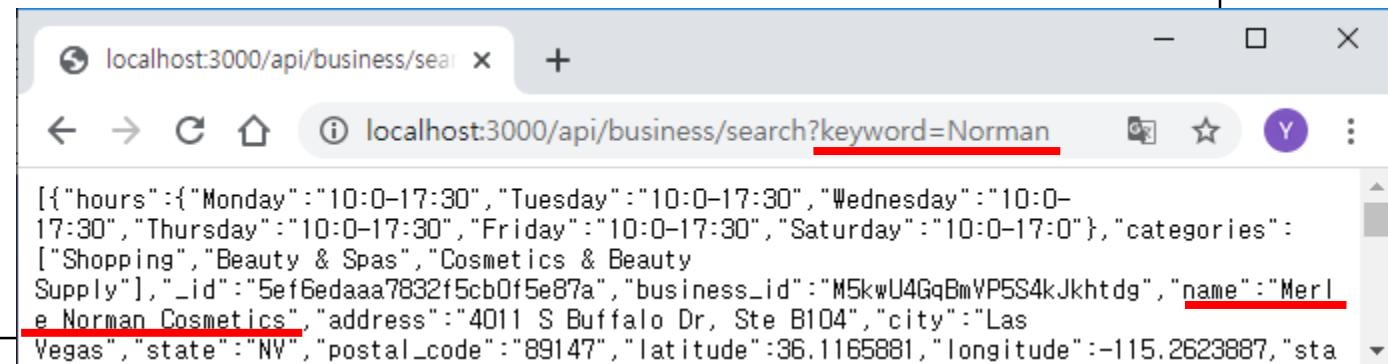
■ Express.js server(tutorials/express/db6.js)

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/yelp',
  {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true});
mongoose.set('debug', true);

const express = require('express');
const app = express();
const port = 3000;

app.use('/api/business', require('./api/business'));
app.use('/api/geodata', require('./api/geodata'));

app.listen(port);
```



4. Express.js + Mongoose

■ Express.js server(tutorials/express/db6.js)

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/yelp',
    {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true});
mongoose.set('debug', true);

const express = require('express');
const app = express();
const port = 3000;

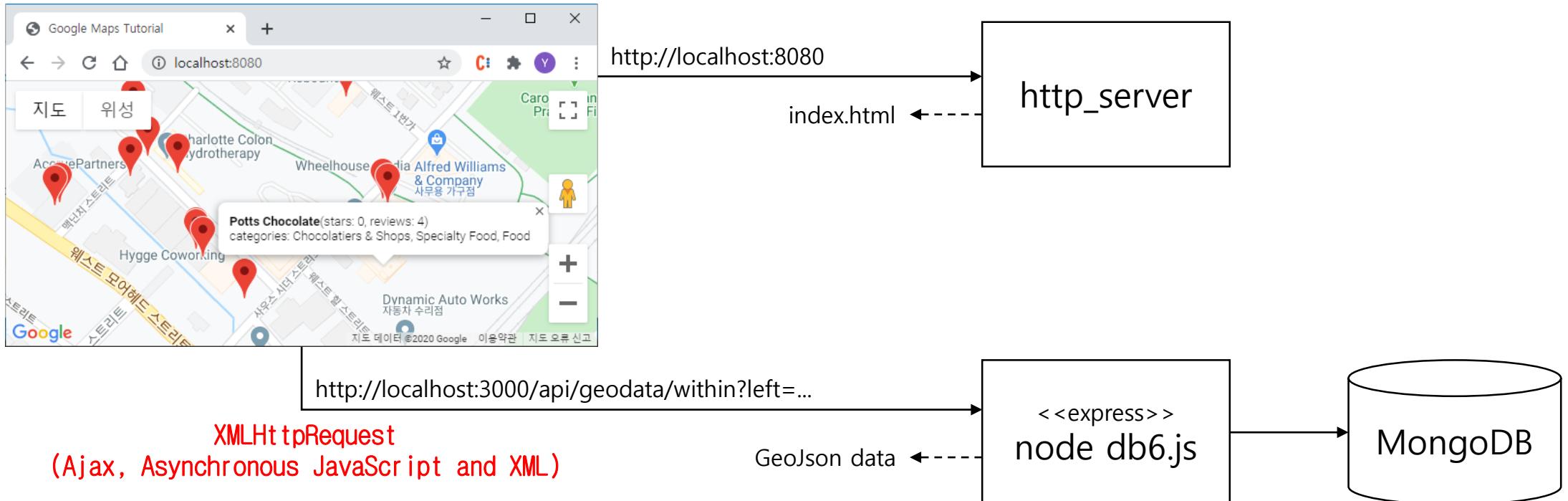
app.use('/api/business', require('./api/business'));
app.use('/api/geodata', require('./api/geodata'));

app.listen(port);
```



4. Express.js + Mongoose

- Integrating Maps with Mongo DB(tutorials/express/index.html)



4. Express.js + Mongoose

- Integrating Maps with Mongo DB(tutorials/express/index.html)

```
<!DOCTYPE html>
<html>
  <head>
    <title>Google Maps Tutorial</title>
    <style type="text/css">
      #map {
        height: 100%;
      }

      html,
      body {
        height: 100%;
        margin: 0;
        padding: 0;
      }
    </style>

    <script>
      function initMap() {
        var options = {
          center: { lat: 35.25, lng: -80.85 },
          zoom: 16,
        };
        var map = new google.maps.Map(document.getElementById('map'), options);
      }
    </script>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>
```

4. Express.js + Mongoose

▪ Integrating Maps with Mongo DB(tutorials/express/index.html)

```
map.addListener('idle', function(){
  // remove all
  map.data.forEach(function(feature){ map.data.remove(feature); });

  const bounds = map.getBounds();
  const sw = bounds.getSouthWest();
  const ne = bounds.getNorthEast();

  const request = new XMLHttpRequest();
  const url = 'http://localhost:3000/api/geodata/within?' + 'left=' + sw.lng() + '&lower=' + sw.lat()
    + '&right=' + ne.lng() + '&upper=' + ne.lat();
  request.open('GET', url);
  request.responseType = 'json';
  request.send();

  request.onload = function() {
    const geodata = request.response;
    for(var i = 0; i < geodata.length; i++){
      const latLng = new google.maps.LatLng({ lng: geodata[i].geometry.coordinates[0],
                                              lat: geodata[i].geometry.coordinates[1] });
      map.data.add(new google.maps.Data.Feature({ geometry: latLng, properties: geodata[i].properties } ));
    }
  }
});
```

4. Express.js + Mongoose

■ Integrating Maps with Mongo DB(tutorials/express/index.html)

```
const infowindow = new google.maps.InfoWindow();
map.data.addListener('click', function(e){
    const feature = e.feature;
    const name = feature.getProperty("name");
    const stars = feature.getProperty("stars");
    const review_count = feature.getProperty("review_count");
    const categories = feature.getProperty("categories");

    let content = '<div><b>' + name + '</b>(stars: ' + stars + ', reviews: ' + review_count + ')<br>categories: ';
    content += categories[0];
    for(var i = 1; i < categories.length; i++)
        content += ', ' + categories[i];
    content += '</div>';

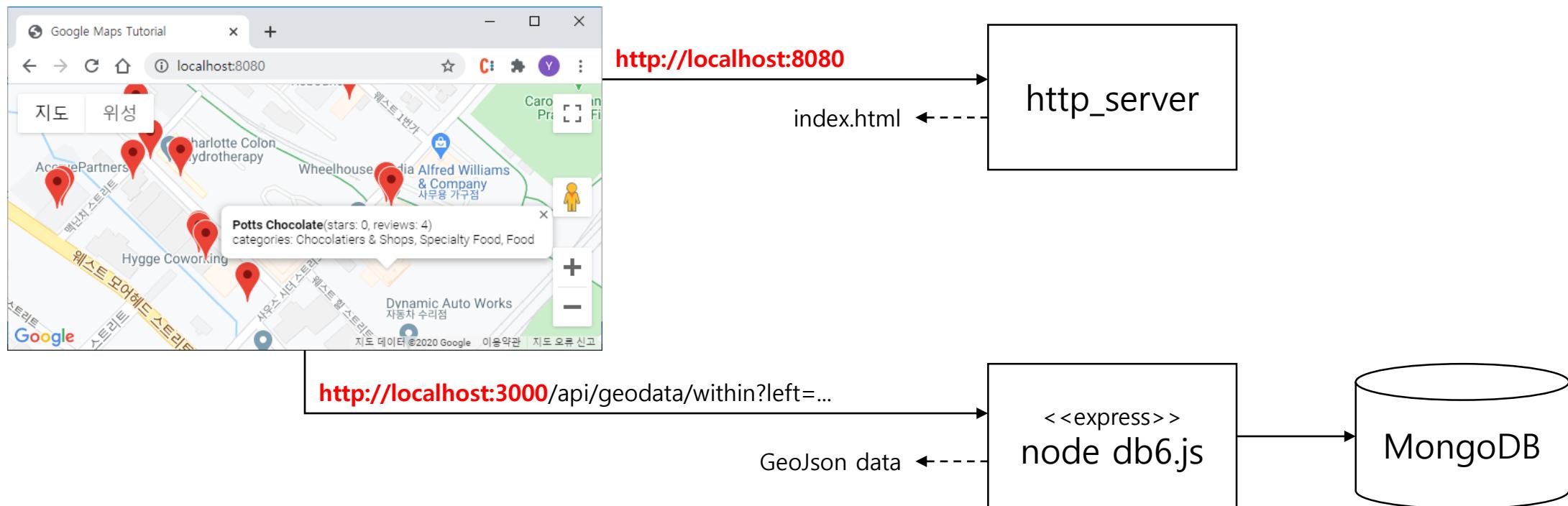
    infowindow.setPosition(e.latLng);
    infowindow.setContent(content);
    infowindow.open(map);
});

</script>
```

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&callback=initMap&libraries=geometry">
</script>
</head>
<body>
<div id="map"></div>
</body>
</html>
```

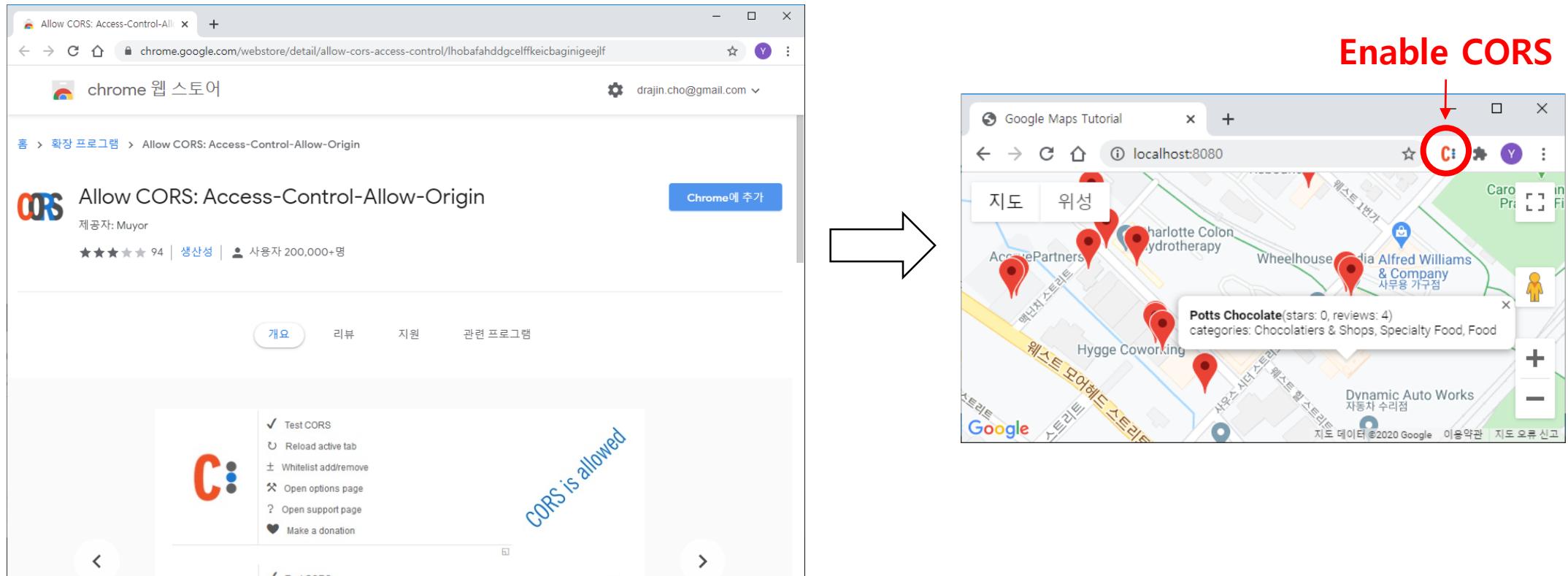
4. Express.js + Mongoose

- Integrating Maps with Mongo DB(tutorials/express/index.html)
- Ajax에서 교차 출처 리소스 공유(Cross-origin Resource Sharing, CORS)는 허용되지 않는다.



4. Express.js + Mongoose

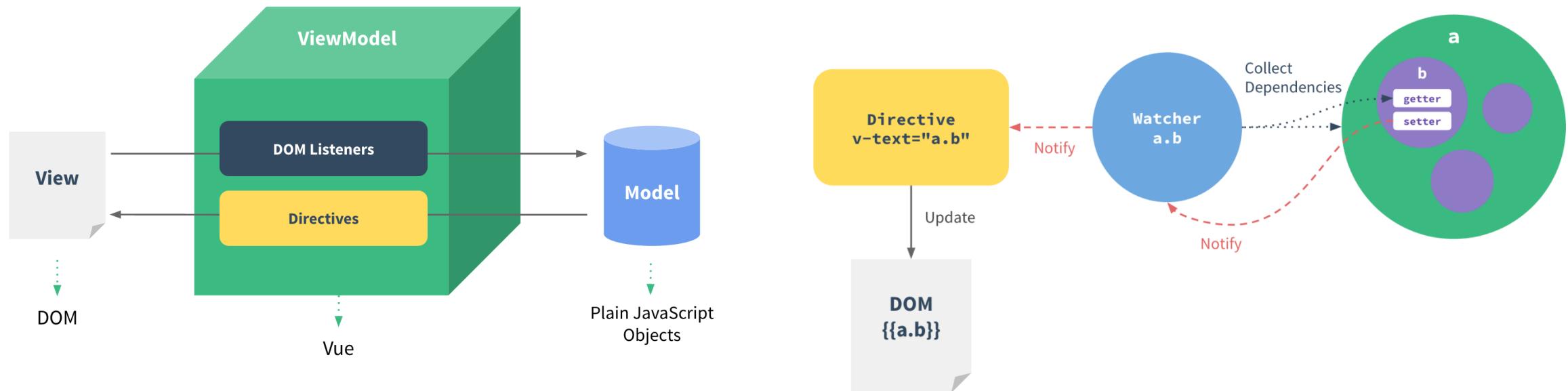
- Integrating Maps with Mongo DB(tutorials/express/index.html)
- Ajax에서 교차 출처 리소스 공유(Cross-origin Resource Sharing, CORS)는 허용되지 않는다. → Chrome Plugins



https://ko.wikipedia.org/wiki/교차_출처_리소스_공유

5. Vue.js

- Vue.js is a library for building interactive web interfaces.
- Vue.js is focused on the View Model layer of the MVVM pattern.



<https://012.vuejs.org/guide/>

5. Vue.js

■ Hello World (tutorials/vue/index1.html)

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <title>Vue.js Tutorial</title>
  </head>
  <body>
    <div id="app">
      {{ message }}
    </div>
    <script>
      var app = new Vue({
        el: '#app',
        data: {
          message: 'Hello, Vue!'
        }
      })
    </script>
  </body>
</html>
```

```
<html>
  <head>
    <title>Vue.js Tutorial</title>
  </head>
  <body>
    <div id="app">
      Hello, Vue!
    </div>
  </body>
</html>
```

5. Vue.js

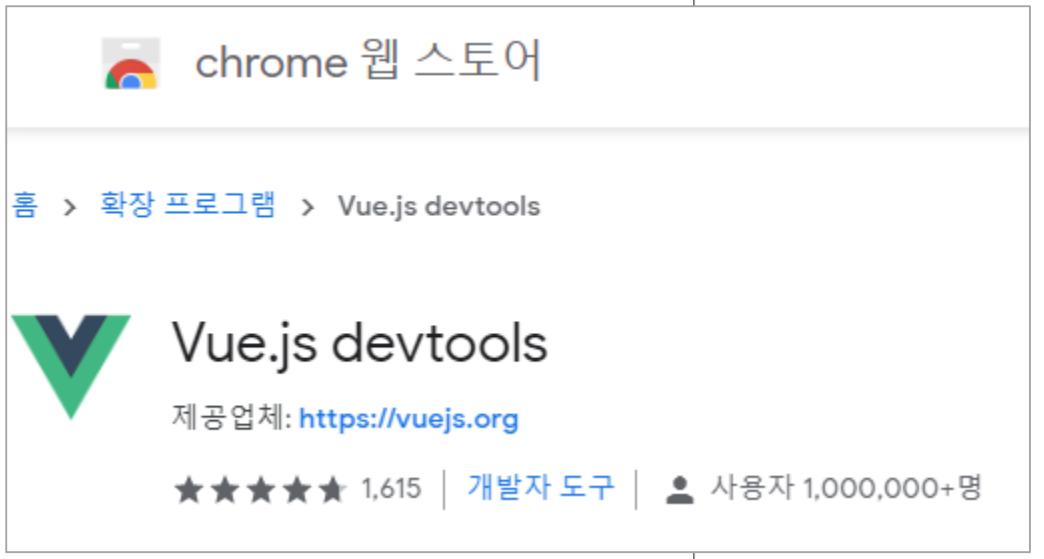
▪ Hello World (tutorials/vue/index2.html)

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <title>Vue.js Tutorial</title>
  </head>
  <body>
    <div id="app">
      <ol>
        <li v-for="todo in todos"> {{ todo.text }} </li>
      </ol>
    </div>
    <script>
      var app = new Vue({
        el: '#app',
        data: {
          todos: [
            { text: 'JavaScript 배우기' },
            { text: 'Vue 배우기' },
            { text: '무언가 멋진 것을 만들기' }
          ]
        }
      })
    </script>
  </body>
</html>
```

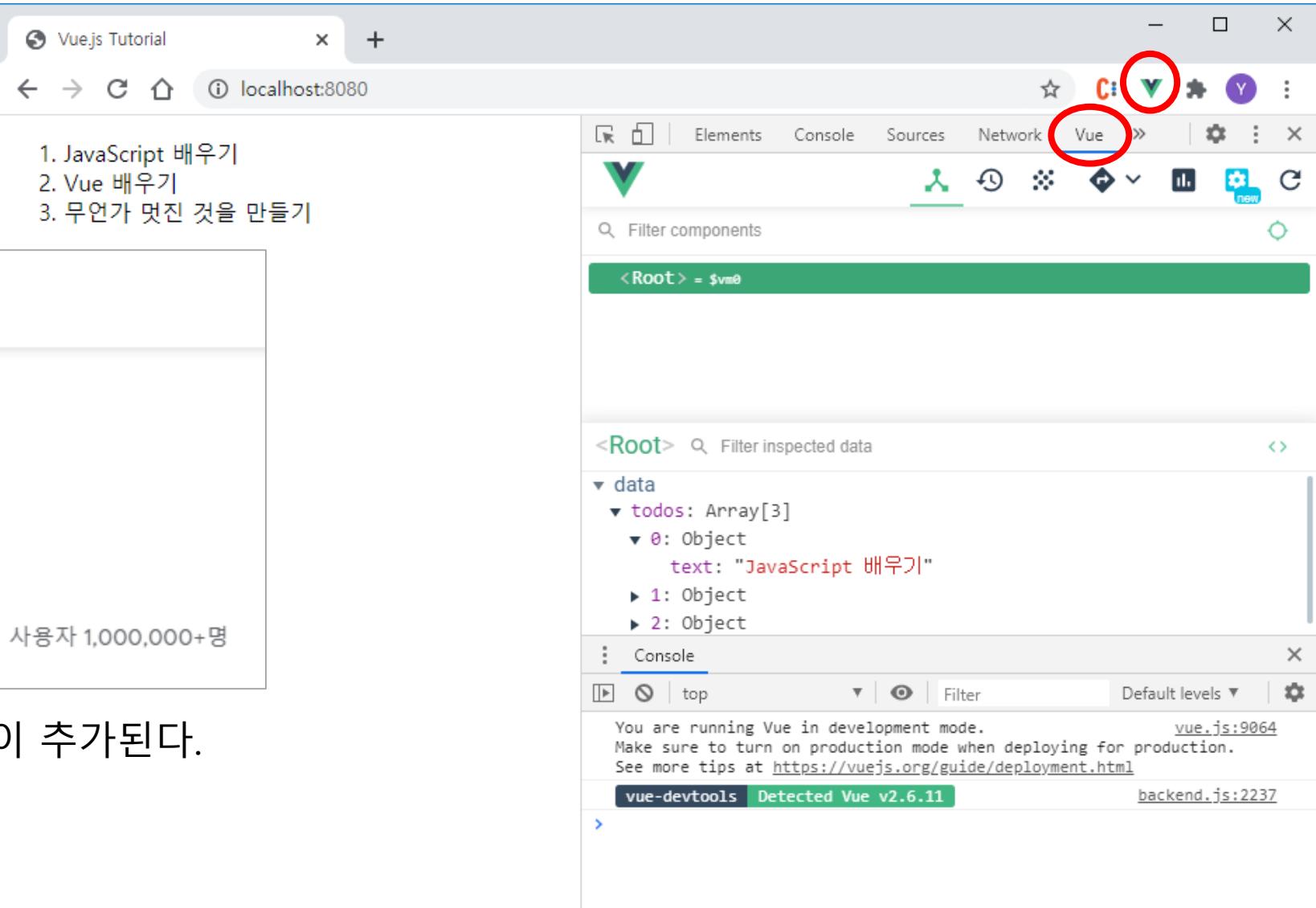
```
<html>
  <head>
    <title>Vue.js Tutorial</title>
  </head>
  <body>
    <div id="app">
      <ol>
        <li> JavaScript 배우기 </li>
        <li> Vue 배우기 </li>
        <li> 무언가 멋진 것을 만들기 </li>
      </ol>
    </div>
  </body>
</html>
```

5. Vue.js

■ Vue.js devtools



Chrome 개발자 도구에 Vue 탭이 추가된다.



The screenshot shows the Chrome DevTools interface with the 'Vue' tab selected. The left sidebar displays a tree view of component data, showing an array of todos with their text content. The bottom console panel shows a message about development mode and deployment tips.

```
<Root> = $vm0
<Root> Filter inspected data
data
todos: Array[3]
0: Object
text: "JavaScript 배우기"
1: Object
2: Object
Console
You are running Vue in development mode. vue.js:9064
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
vue-devtools Detected Vue v2.6.11
backend.js:2237
```

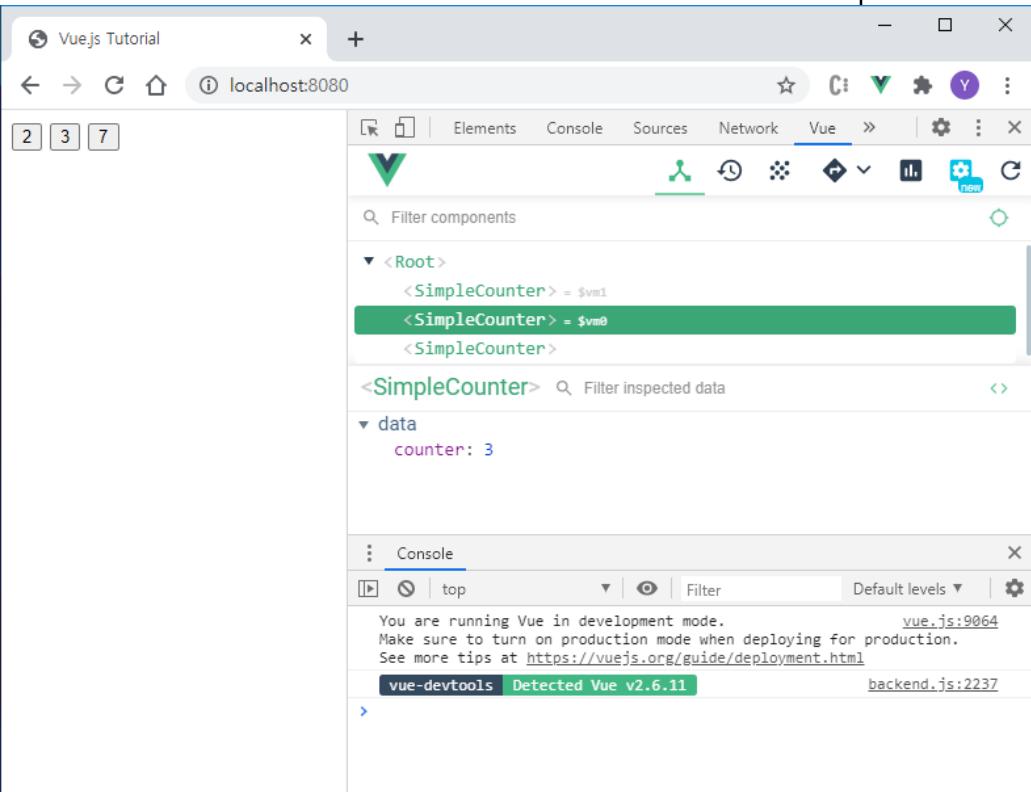
5. Vue.js

<https://kr.vuejs.org/v2/guide/components.html>

■ Vue Component ([tutorials/vue/index3.html](https://tutorials.vue/index3.html))

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    <title>Vue.js Tutorial</title>
  </head>
  <body>
    <div id="app">
      <simple-counter></simple-counter>
      <simple-counter></simple-counter>
      <simple-counter></simple-counter>
    </div>
    <script>
      Vue.component('simple-counter', {
        template: '<button v-on:click="counter += 1">{{ counter }}</button>',
        data: function() {
          return { counter: 0 }
        },
      })

      var app = new Vue({ el: '#app' })
    </script>
  </body>
</html>
```



5. Vue.js

- Vue CLI – standard tool for Vue.js Development

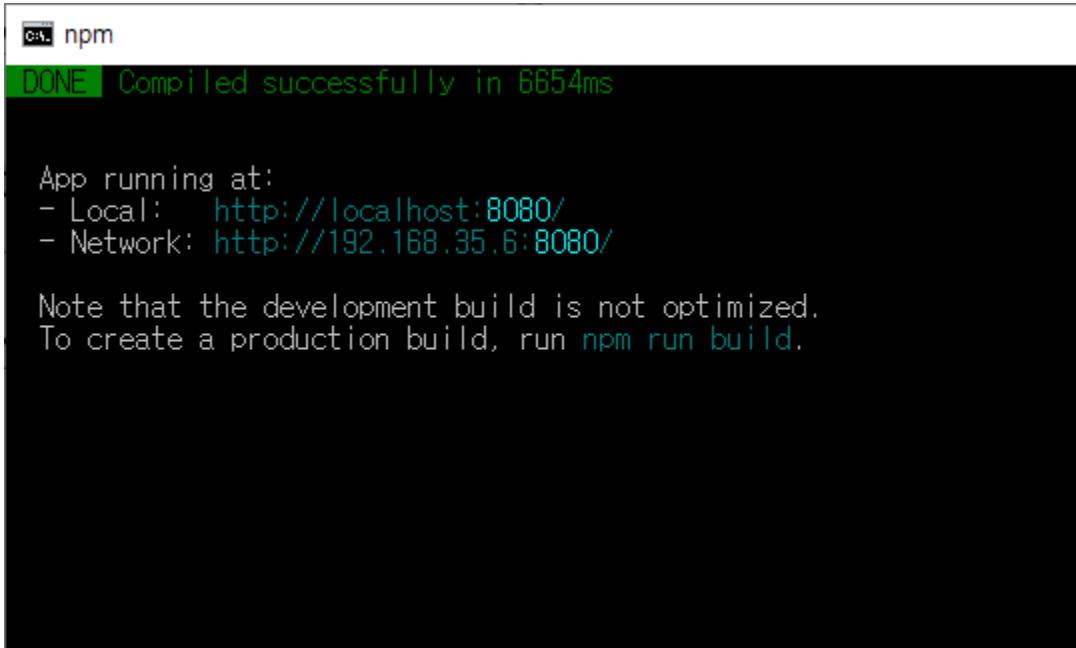
<https://cli.vuejs.org/>

```
C:\> npm install @vue/cli -g
```

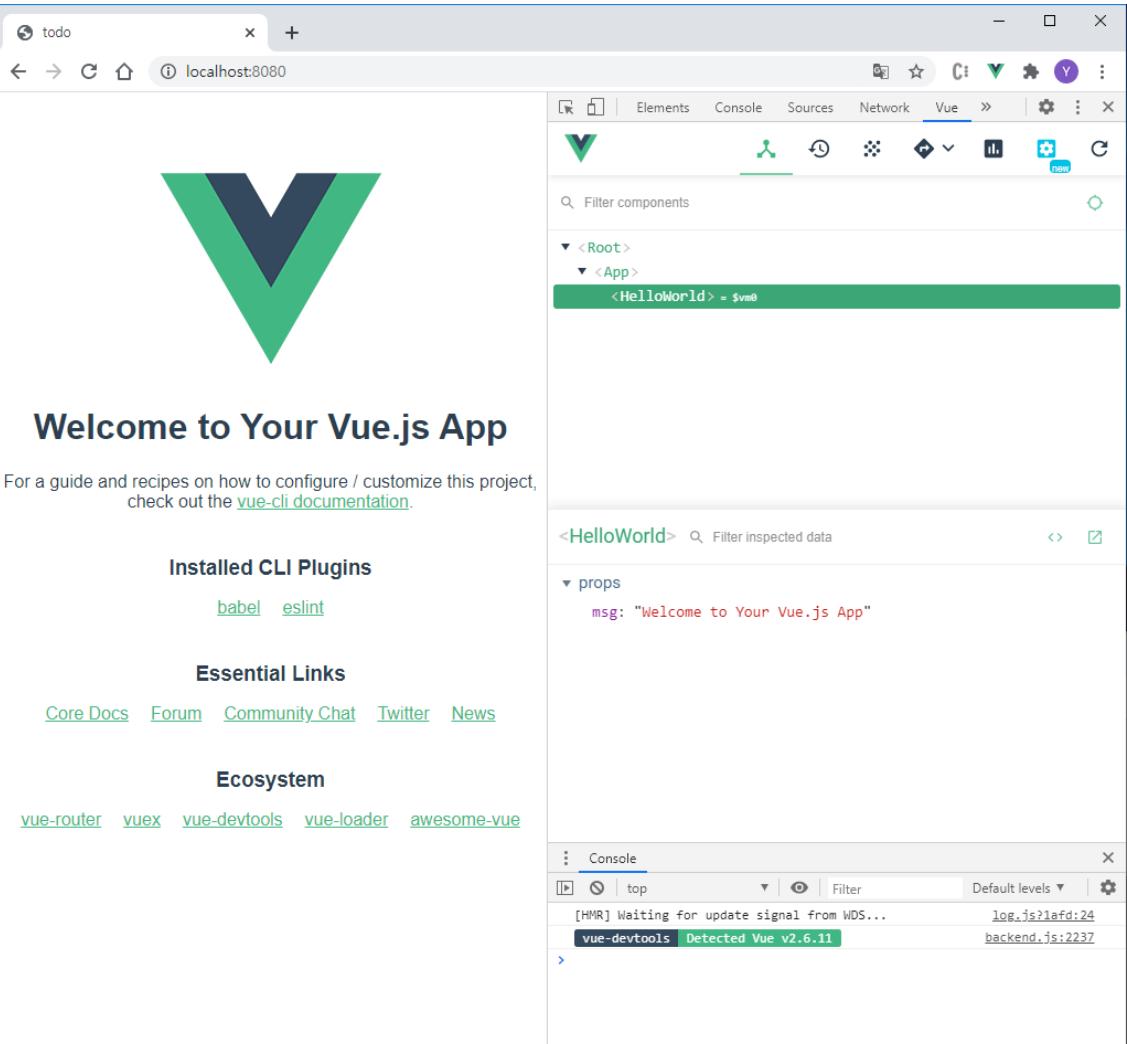
```
C:\> vue create todo
```

```
C:\> cd todo
```

```
C:\todo> npm run serve
```



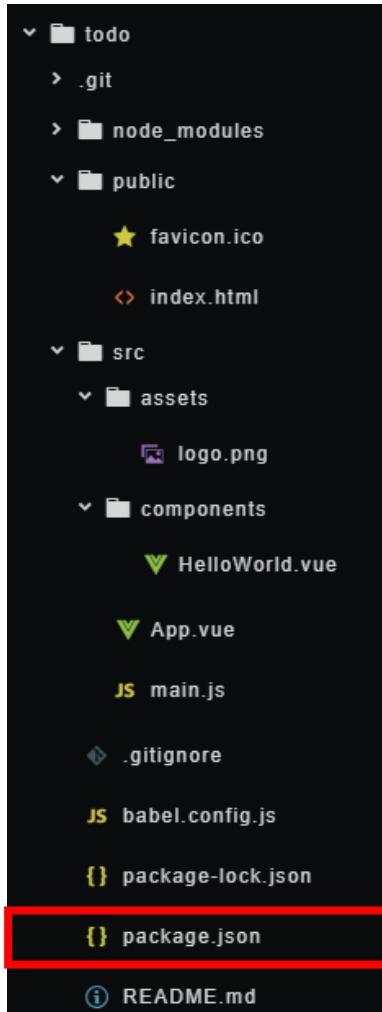
```
npm
DONE Compiled successfully in 6654ms
App running at:
- Local:  http://localhost:8080/
- Network: http://192.168.35.6:8080/
Note that the development build is not optimized.
To create a production build, run npm run build.
```



5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>

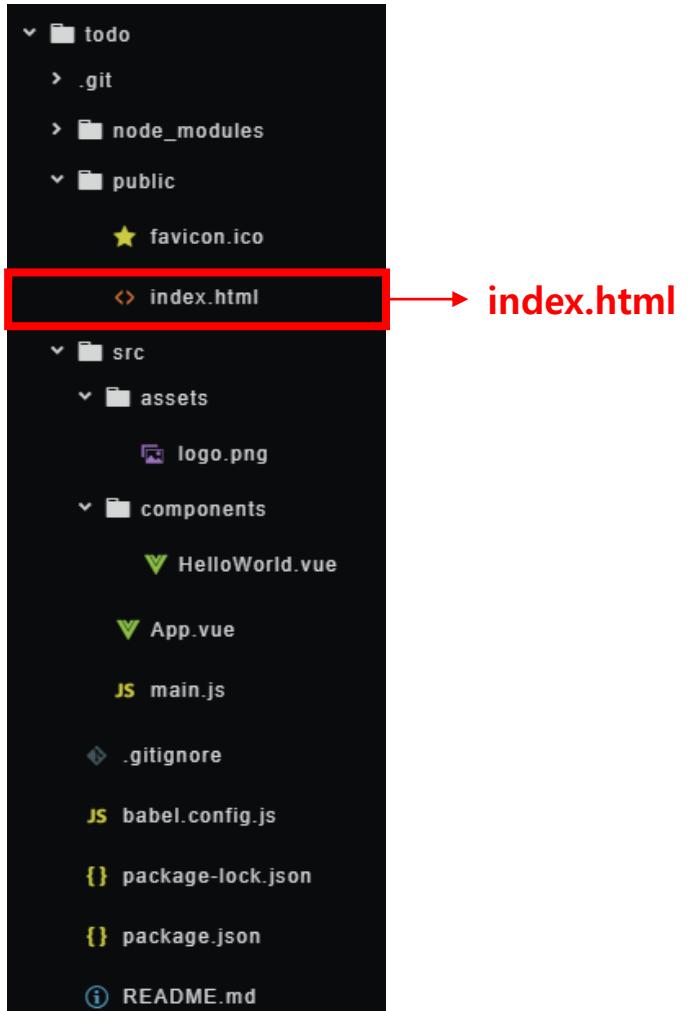


```
{  
  "name": "todo",  
  "version": "0.1.0",  
  "private": true,  
  "scripts": {  
    "serve": "vue-cli-service serve",  
    "build": "vue-cli-service build",  
    "lint": "vue-cli-service lint"  
  },  
  "dependencies": {  
    "core-js": "^3.6.5",  
    "vue": "^2.6.11"  
  },  
  "devDependencies": {  
    "@vue/cli-plugin-babel": "~4.4.0",  
    "@vue/cli-plugin-eslint": "~4.4.0",  
    "@vue/cli-service": "~4.4.0",  
    "babel-eslint": "^10.1.0",  
    "eslint": "^6.7.2",  
    "eslint-plugin-vue": "^6.2.2",  
    "vue-template-compiler": "^2.6.11"  
  },  
  "eslintConfig": { ... },  
  "browserslist": [ "> 1%", "last 2 versions", "not dead" ]  
}
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<link rel="icon" href="<%= BASE_URL %>favicon.ico">
<title><%= htmlWebpackPlugin.options.title %></title>
</head>
<body>
<div id="app"></div>
<!-- built files will be auto injected -->
</body>
</html>
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <link rel="icon" href="<%= BASE_URL %>favicon.ico">
  <title><%= htmlWebpackPlugin.options.title %></title>
</head>
<body>
  <div id="app"></div>
  <!-- built files will be auto injected -->
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,initial-scale=1.0">
  <link rel="icon" href="/favicon.ico">
  <title>todo</title>
  <link href="/js/app.js" rel="preload" as="script">
  <link href="/js/chunk-vendors.js" rel="preload" as="script"></head>
<body>
  <div id="app"></div>
  <!-- built files will be auto injected -->
  <script type="text/javascript" src="/js/chunk-vendors.js"></script>
  <script type="text/javascript" src="/js/app.js"></script></body>
</html>
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>

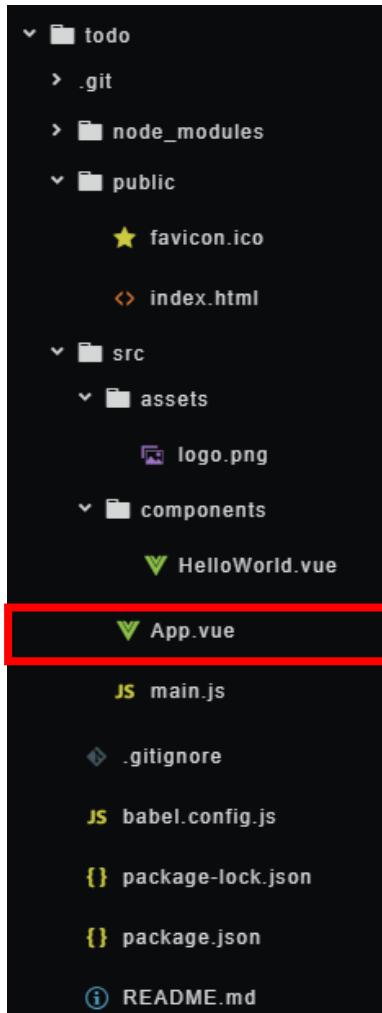


```
import Vue from 'vue'  
import App from './App.vue'  
  
Vue.config.productionTip = false  
  
new Vue({  
  render: h => h(App),  
}).$mount('#app')
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>



App.vue(main component)

```
<template>
<div id="app">
  
  <HelloWorld msg="Welcome to Your Vue.js App"/>
</div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

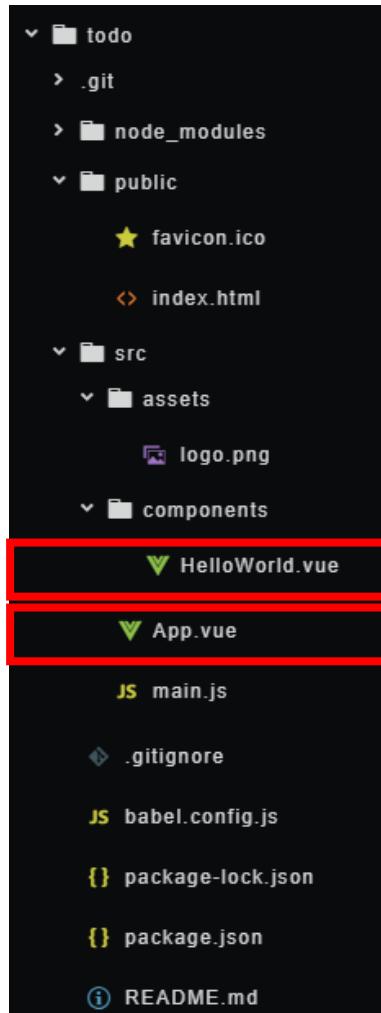
export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

<https://cli.vuejs.org/>



→ **HelloWorld.vue**

→ **App.vue(main component)**

```
<template>
<div id="app">
  
  <HelloWorld msg="Welcome to Your Vue.js App"/>
</div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

```
<template>
<div class="hello">
  <h1>{{ msg }}</h1>
</div>
</template>

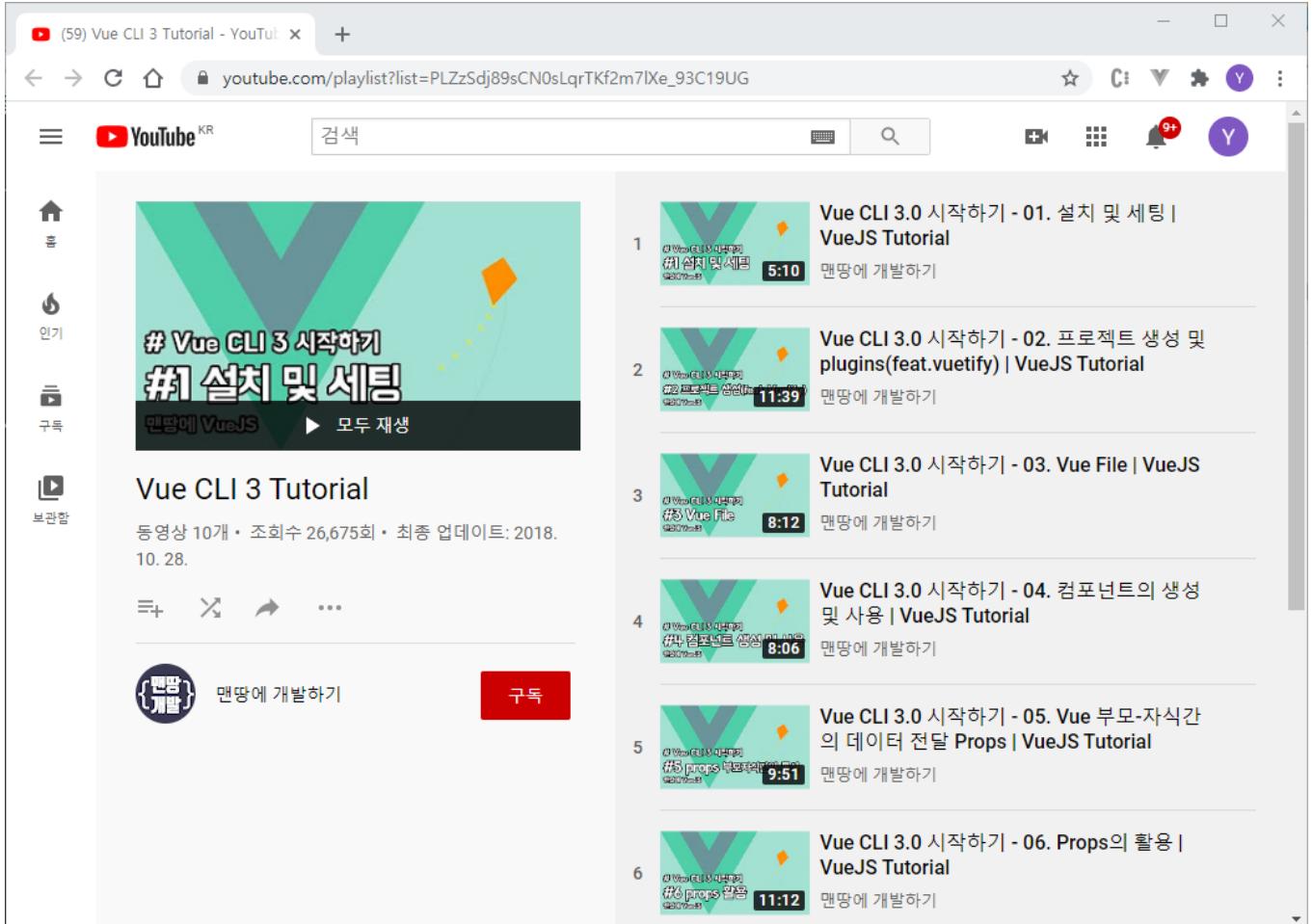
<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>
```

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

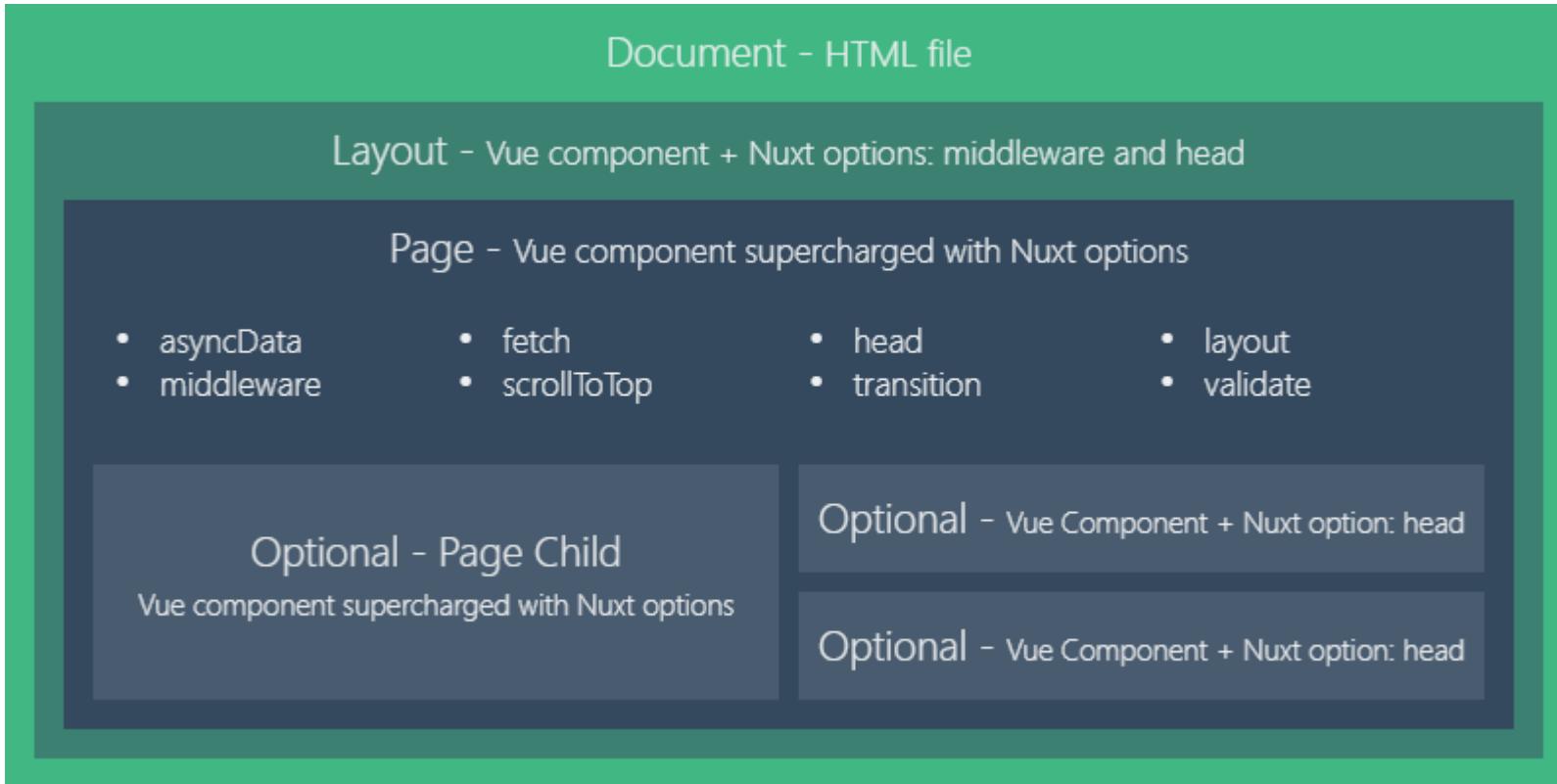
<https://cli.vuejs.org/>

https://www.youtube.com/playlist?list=PLZzSdj89sCN0sLqrTKf2m7lXe_93C19UG



6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework

```
C:\> npx create-nuxt-app nuxt
```

```
create-nuxt-app v3.1.0
```

```
★ Generating Nuxt.js project in nuxt
```

```
? Project name: nuxt
```

```
? Programming language: JavaScript
```

```
? Package manager: Npm
```

```
? UI framework: (Press <space> to select, <a> to toggle all, <i> to invert selection)
```

```
? Nuxt.js modules: (Press <space> to select, <a> to toggle all, <i> to invert selection)
```

```
? Linting tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)
```

```
? Testing framework: None
```

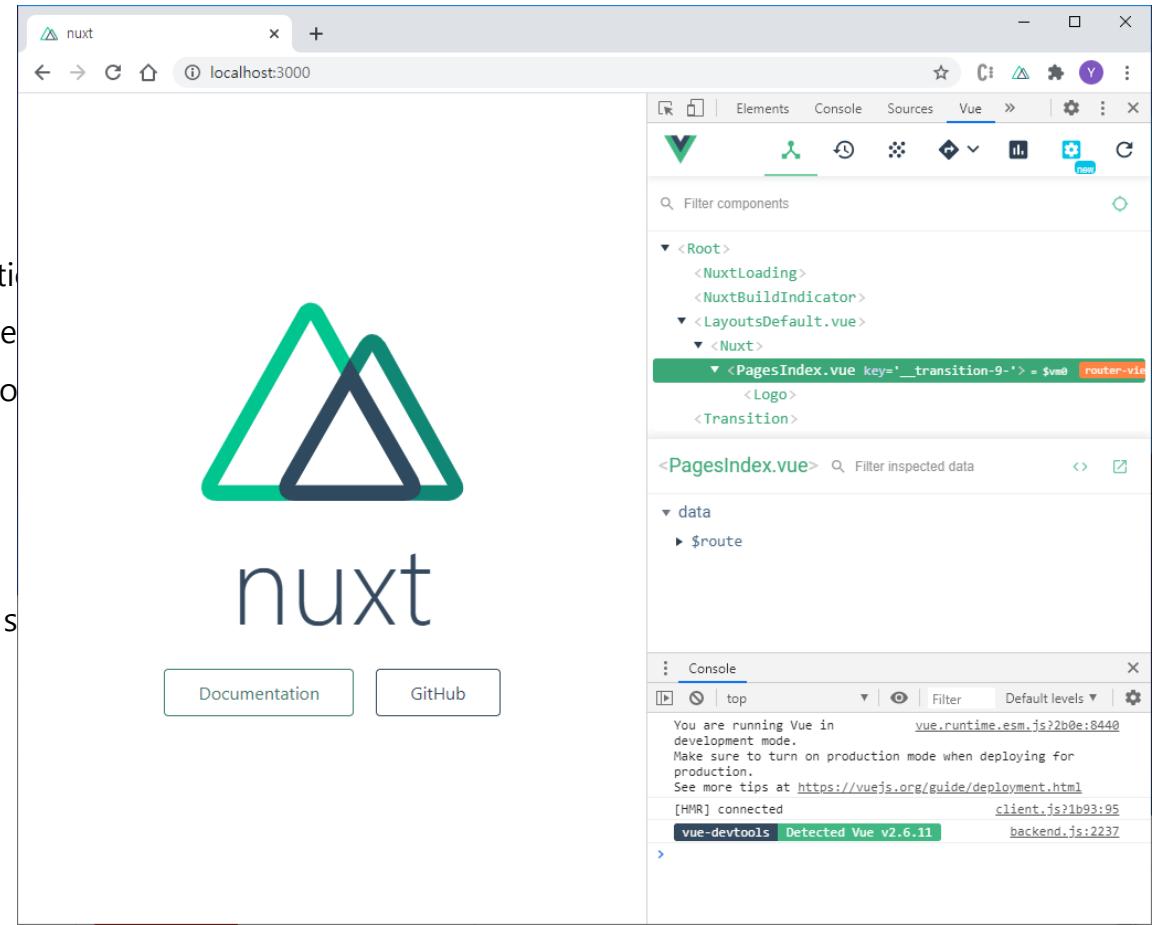
```
? Rendering mode: Universal (SSR / SSG)
```

```
? Deployment target: Server (Node.js hosting)
```

```
? Development tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)
```

```
C:\> cd nuxt
```

```
C:\nuxt> npm run dev
```



6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework

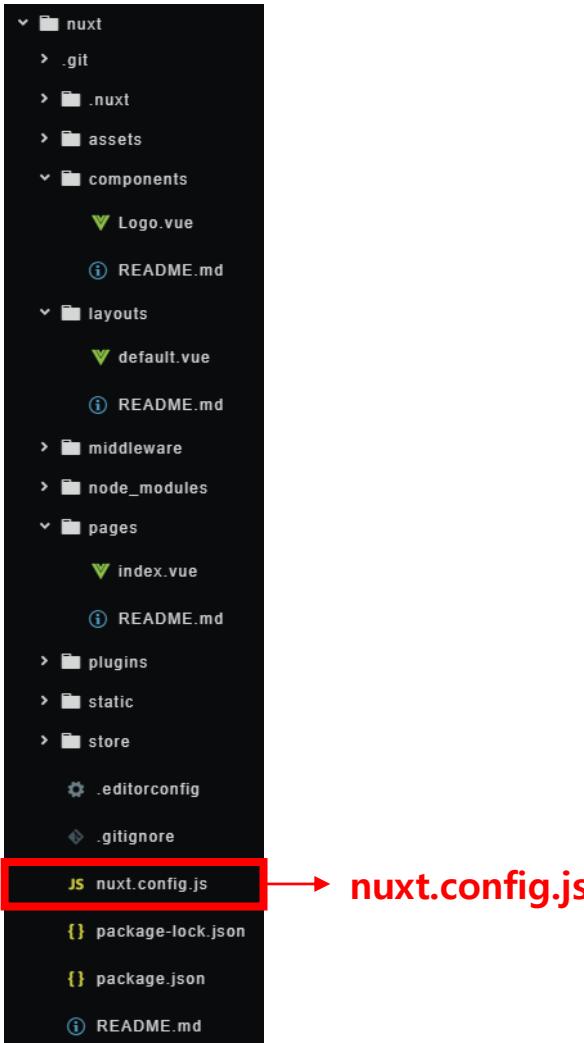


```
{  
  "name": "nuxt",  
  "version": "1.0.0",  
  "private": true,  
  "scripts": {  
    "dev": "nuxt",  
    "build": "nuxt build",  
    "start": "nuxt start",  
    "export": "nuxt export",  
    "serve": "nuxt serve"  
},  
  "dependencies": {  
    "nuxt": "^2.13.0"  
},  
  "devDependencies": {}  

```

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



```
export default {  
  /*  
   * Nuxt rendering mode  
   * See https://nuxtjs.org/api/configuration-mode  
   */  
  mode: 'universal',  
  /*  
   * Nuxt target  
   * See https://nuxtjs.org/api/configuration-target  
   */  
  target: 'server',  
  /*  
   * Headers of the page  
   * See https://nuxtjs.org/api/configuration-head  
   */  
  head: {  
    title: process.env.npm_package_name || '',  
    meta: [  
      { charset: 'utf-8' },  
      { name: 'viewport', content: 'width=device-width, initial-scale=1' },  
      { hid: 'description', name: 'description', content: process.env.npm_package_description || '' }  
    ],  
    link: [  
      { rel: 'icon', type: 'image/x-icon', href: '/favicon.ico' }  
    ]  
  },  
};
```

6. Vue.js / Nuxt.js

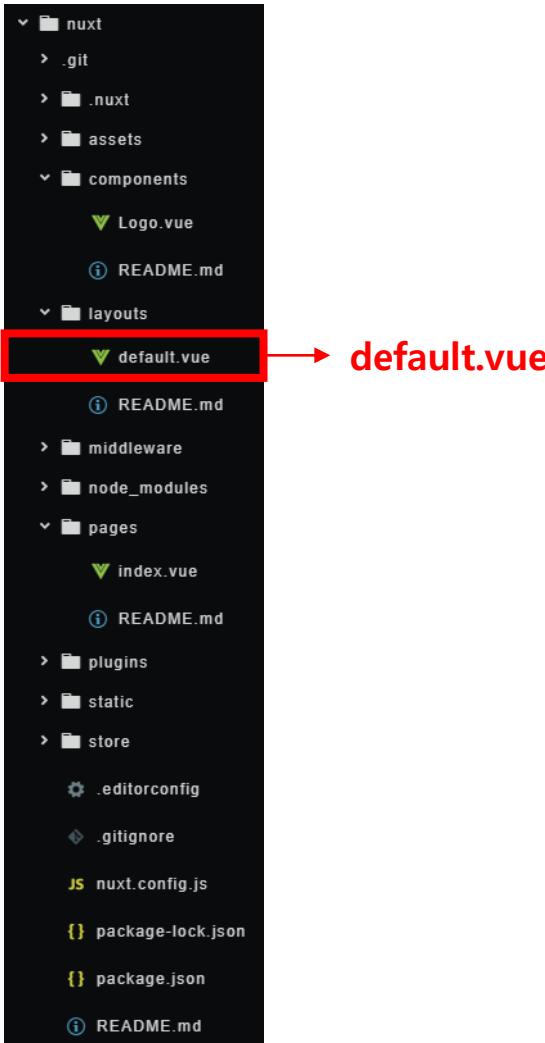
-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



```
/*
 ** Global CSS
 */
css: [
],
/*
** Plugins to load before mounting the App
** https://nuxtjs.org/guide/plugins
*/
plugins: [
],
/*
** Auto import components
** See https://nuxtjs.org/api/configuration-components
*/
components: true,
/*
** Nuxt.js dev-modules
*/
buildModules: [
],
/*
** Nuxt.js modules
*/
modules: [
],
/*
** Build configuration
** See https://nuxtjs.org/api/configuration-build/
*/
build: {
}
```

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



```
<template>
```

```
</div>
```

```
<Nuxt />
```

```
</div>
```

```
</template>
```

```
<style>
```

```
...
```

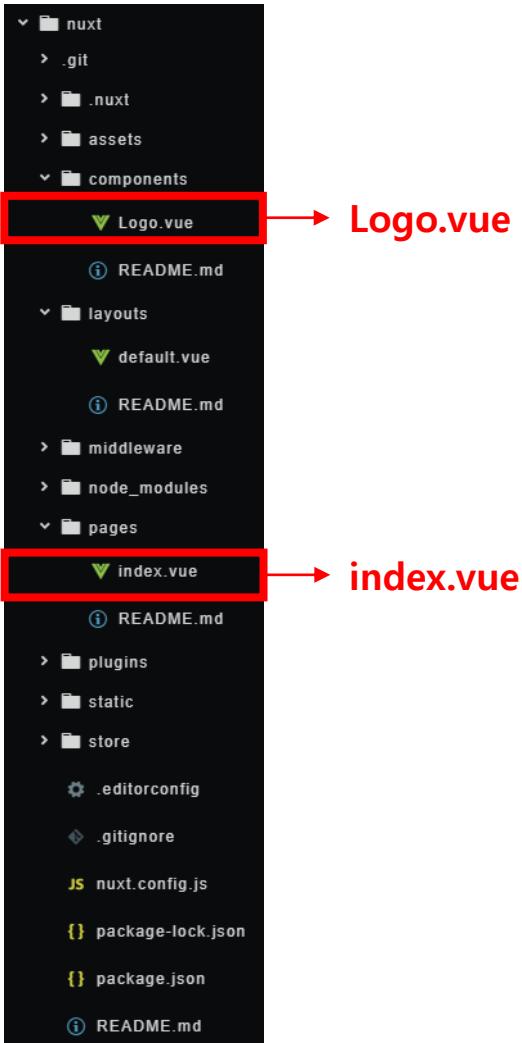
```
</style>
```

모든 페이지에 사용되는 layout에 대한 정의.

각 페이지는 <Nuxt />에 렌더링된다.

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



```
<template>
<div class="container">
  <div>
    <Logo />
    <h1 class="title">
      nuxt
    </h1>
  </div>
</div>
</template>

<script>
  export default {}
</script>

<style>
...
</style>
```

각각 페이지는 /pages 폴더에
/ → /pages/index.vue
/a → /pages/a/index.vue
로 라우팅 된다.
공용 컴포넌트는 /components 폴더에.

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework
- 라우팅(<https://ko.nuxtjs.org/guide/routing>)

```
pages/
--| user/
----| index.vue
----| one.vue
--| index.vue
```

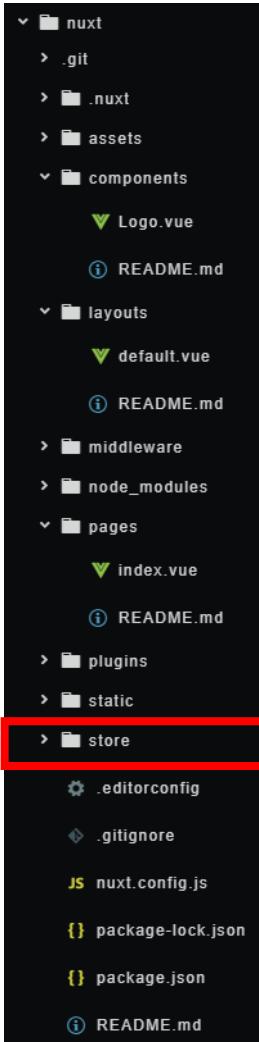
/	→ pages/index.vue
/user	→ pages/user/index.vue
/user/one	→ pages/user/one.vue

```
pages/
--| _slug/
----| comments.vue
----| index.vue
--| users/
----| _id.vue
--| index.vue
```

/	→ pages/index.vue
/:slug	→ pages/_slug/index.vue
/:slug/comments	→ pages/_slug/comments.vue
/users/:id	→ pages/users/_id.vue

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



→ Vuex 공유 저장소

/store/todos.js

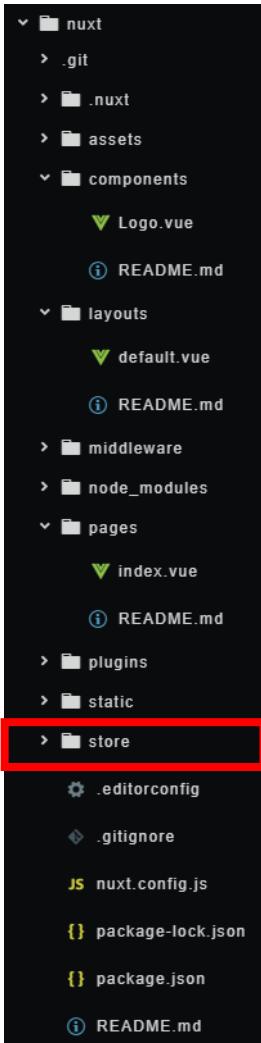
```
export const state = () => ({  
  list: []  
})  
  
export const mutations = {  
  add (state, text) {  
    state.list.push(text)  
  }  
}
```

/pages/todo.vue

```
<template>  
<ul>  
  <li v-for="todo in todos">{{todo}}</li>  
  <li><input @keyup.enter="addTodo"  
         placeholder="What needs to be done?"></li>  
</ul>  
</template>  
  
<script>  
  export default {  
    computed: {  
      todos() {  
        return this.$store.state.todos.list  
      }  
    },  
    methods: {  
      addTodo(e){  
        this.$store.commit('todos/add', e.target.value)  
        e.target.value = ''  
      }  
    }  
  }  
</script>
```

6. Vue.js / Nuxt.js

-  **NUXTJS** (<https://nuxtjs.org/>) – The Intuitive Vue Framework



→ Vuex 공유 저장소

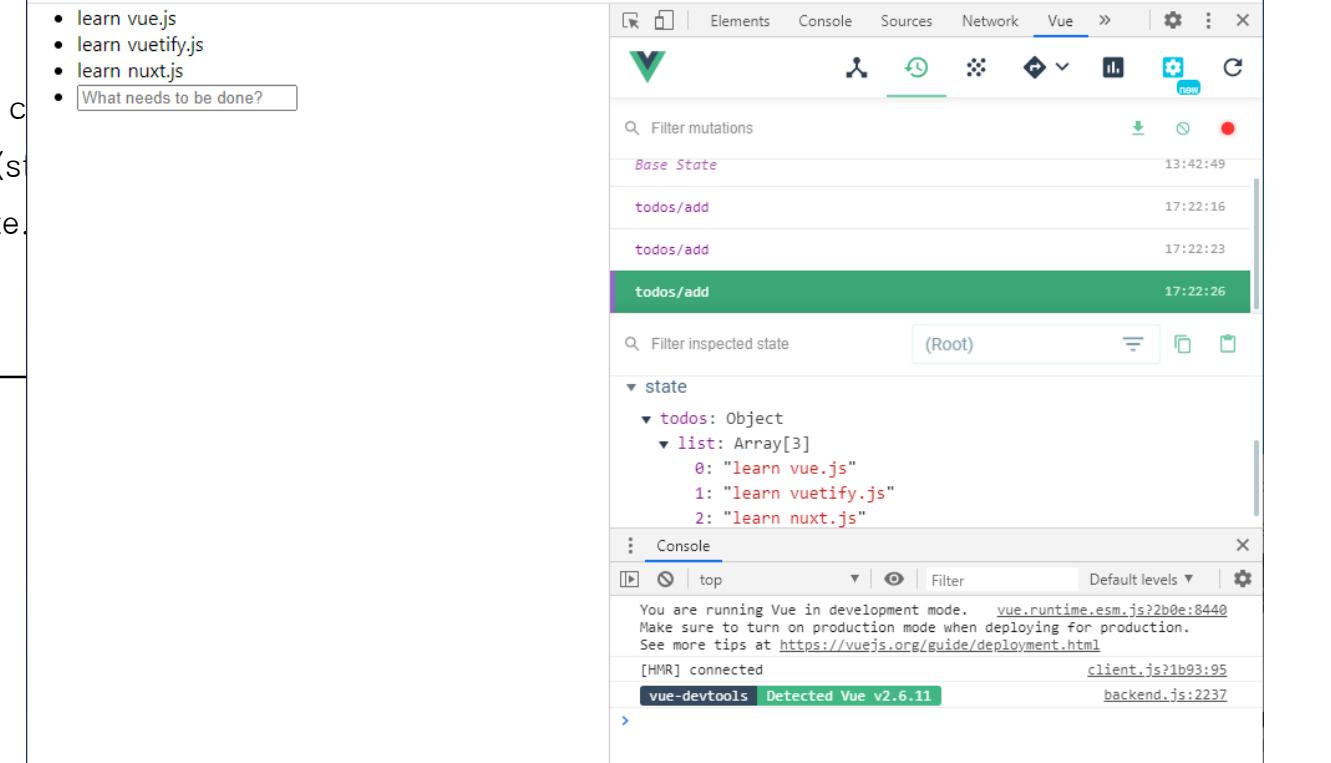
/store/todos.js

```
export const state = () => ({  
  list: []  
})  
  
export const mutations = {  
  add(state, todo) {  
    state.list.push(todo)  
  }  
}  
  
export const actions = {  
  add({ commit }, todo) {  
    commit('add', todo)  
  }  
}
```

localhost:3000/todo

• learn vue.js
• learn vuetify.js
• learn nuxt.js
• What needs to be done?

/pages/todo.vue



The screenshot shows the Vue DevTools interface integrated into a browser's developer tools. The left sidebar shows the Vuex store structure with 'todos' as the state key. The main area displays a list of mutations that have been triggered. The mutations log shows three 'todos/add' events, each adding a new item to the 'list' array. The timeline on the right shows these mutations occurring at specific timestamps. The bottom section of the devtools shows the current state of the 'todos' object, which contains an array of three items: 'learn vue.js', 'learn vuetify.js', and 'learn nuxt.js'.

Vue DevTools

mutations

todos/add

todos/add

todos/add

state

todos: Object

list: Array[3]

0: "learn vue.js"
1: "learn vuetify.js"
2: "learn nuxt.js"

Console

Default levels

You are running Vue in development mode. vue.runtime.esm.js?2b0e:8440
Make sure to turn on production mode when deploying for production.
See more tips at <https://vuejs.org/guide/deployment.html>

[HMR] connected

client.js?1b93:95

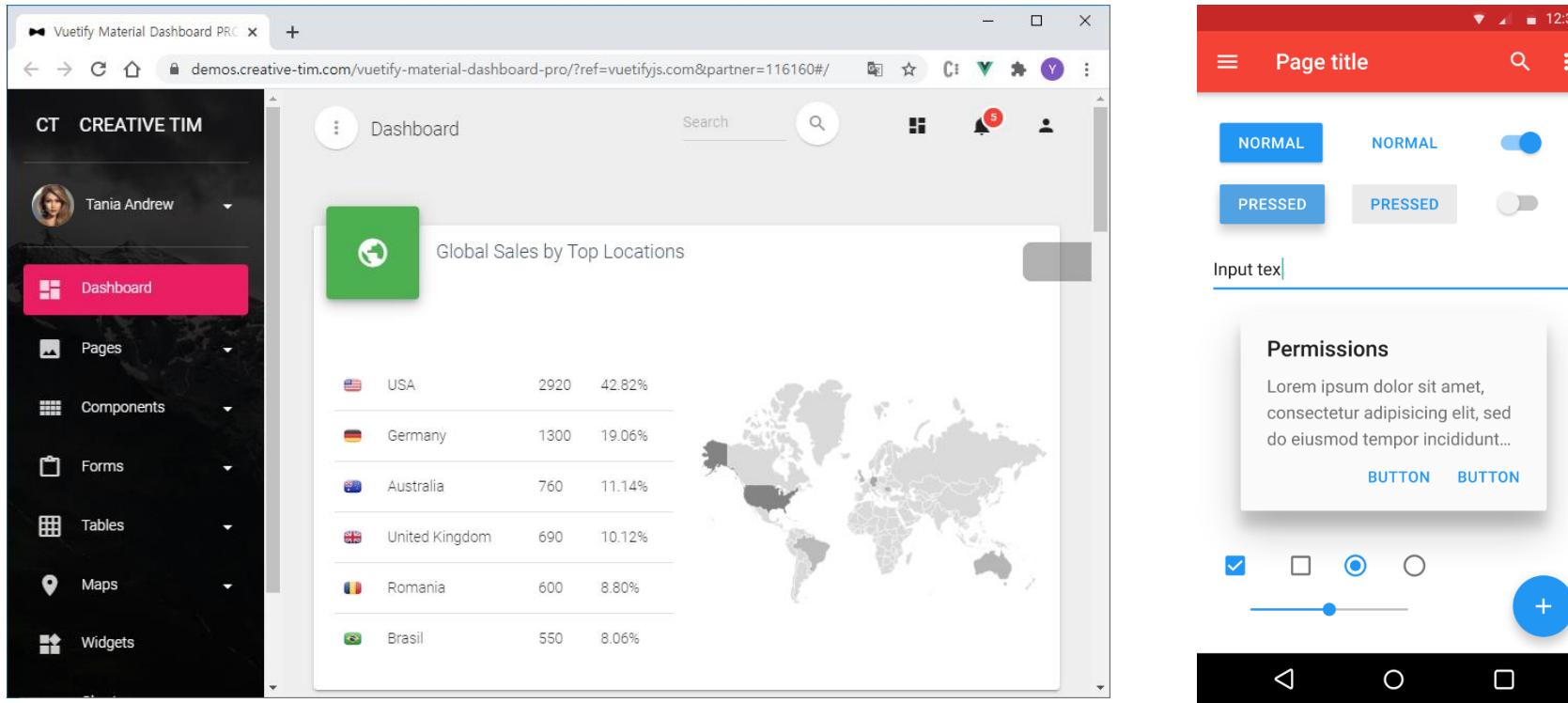
vue-devtools Detected Vue v2.6.11

backend.js:2237

6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

머티리얼 디자인(Material Design, 코드명: Quantum Paper)이란 플랫 디자인의 장점을 살리면서도 빛에 따른 종이의 그림자 효과를 이용하여 입체감을 살리는 디자인 방식을 말한다. 2014년 구글이 안드로이드 스마트폰에 적용하면서 널리 퍼지기 시작했다. 플랫 디자인과 마찬가지로 최소한의 요소만을 사용하여 대상의 본질을 표현하는 디자인 기법인 미니멀리즘(minimalism)을 추구한다. (https://ko.wikipedia.org/wiki/머티리얼_디자인)
- <https://material.io/>



6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

C:\> **npx create-nuxt-app vuetify**

create-nuxt-app v3.1.0

★ Generating Nuxt.js project in vuetify

? Project name: vuetify

? Programming language: JavaScript

? Package manager: Npm

? UI framework: **Vuetify.js**

? Nuxt.js modules: Axios, Progressive Web App (PWA), Content

? Linting tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)

? Testing framework: None

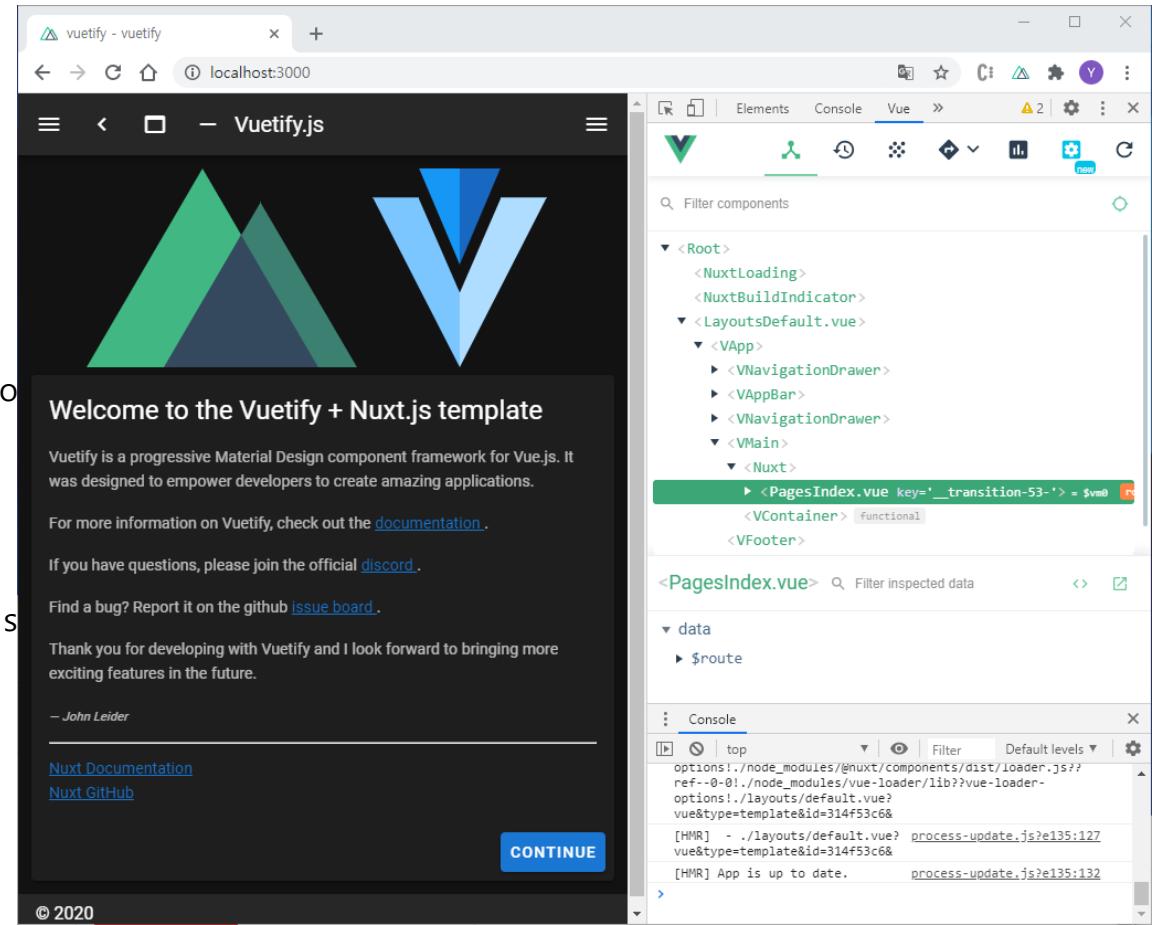
? Rendering mode: Universal (SSR / SSG)

? Deployment target: Server (Node.js hosting)

? Development tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)

C:\> cd vuetify

C:\vuetify> **npm run dev**



6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

C:\> **npx create-nuxt-app vuetify**

create-nuxt-app v3.1.0

★ Generating Nuxt.js project in vuetify

? Project name: vuetify

? Programming language: JavaScript

? Package manager: Npm

? UI framework: **Vuetify.js**

? Nuxt.js modules: Axios, Progressive Web App (PWA), Content

? Linting tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)

? Testing framework: None

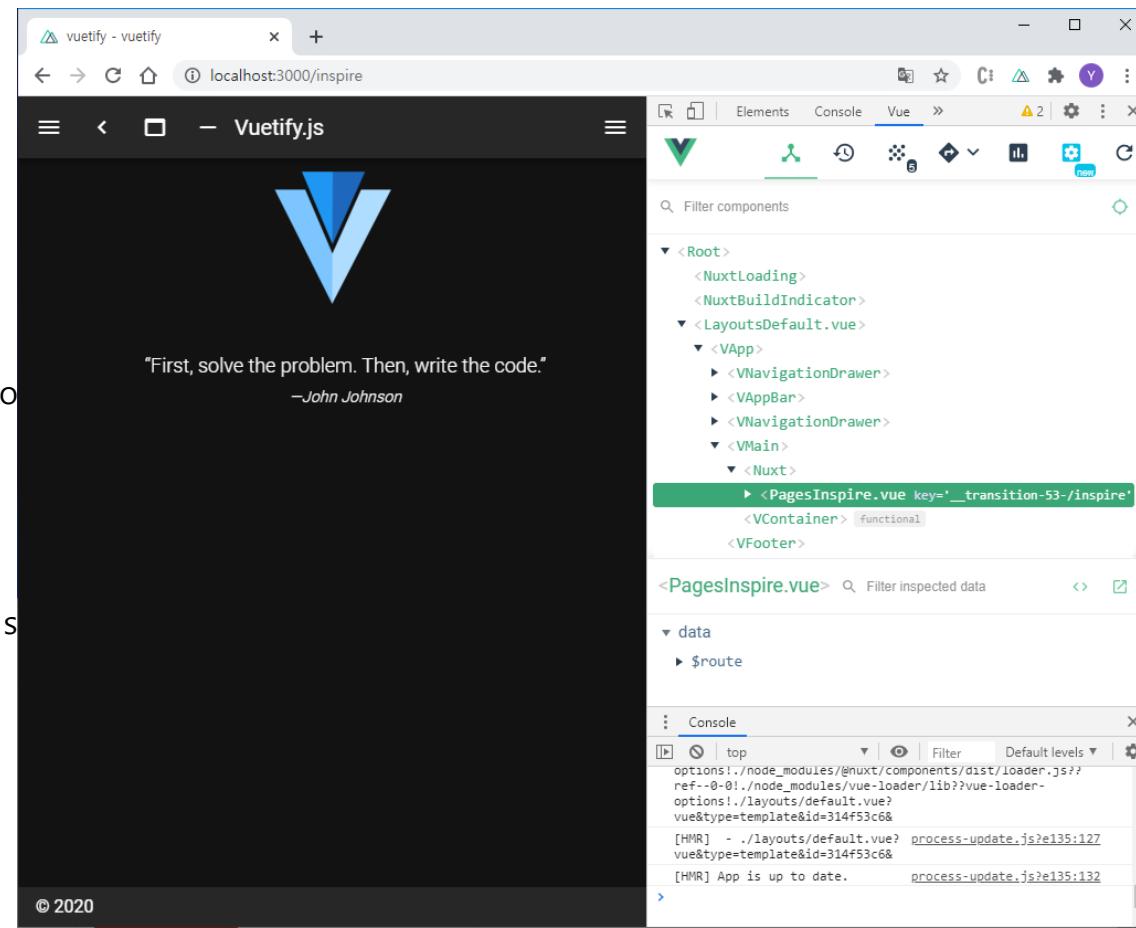
? Rendering mode: Universal (SSR / SSG)

? Deployment target: Server (Node.js hosting)

? Development tools: (Press <space> to select, <a> to toggle all, <i> to invert selection)

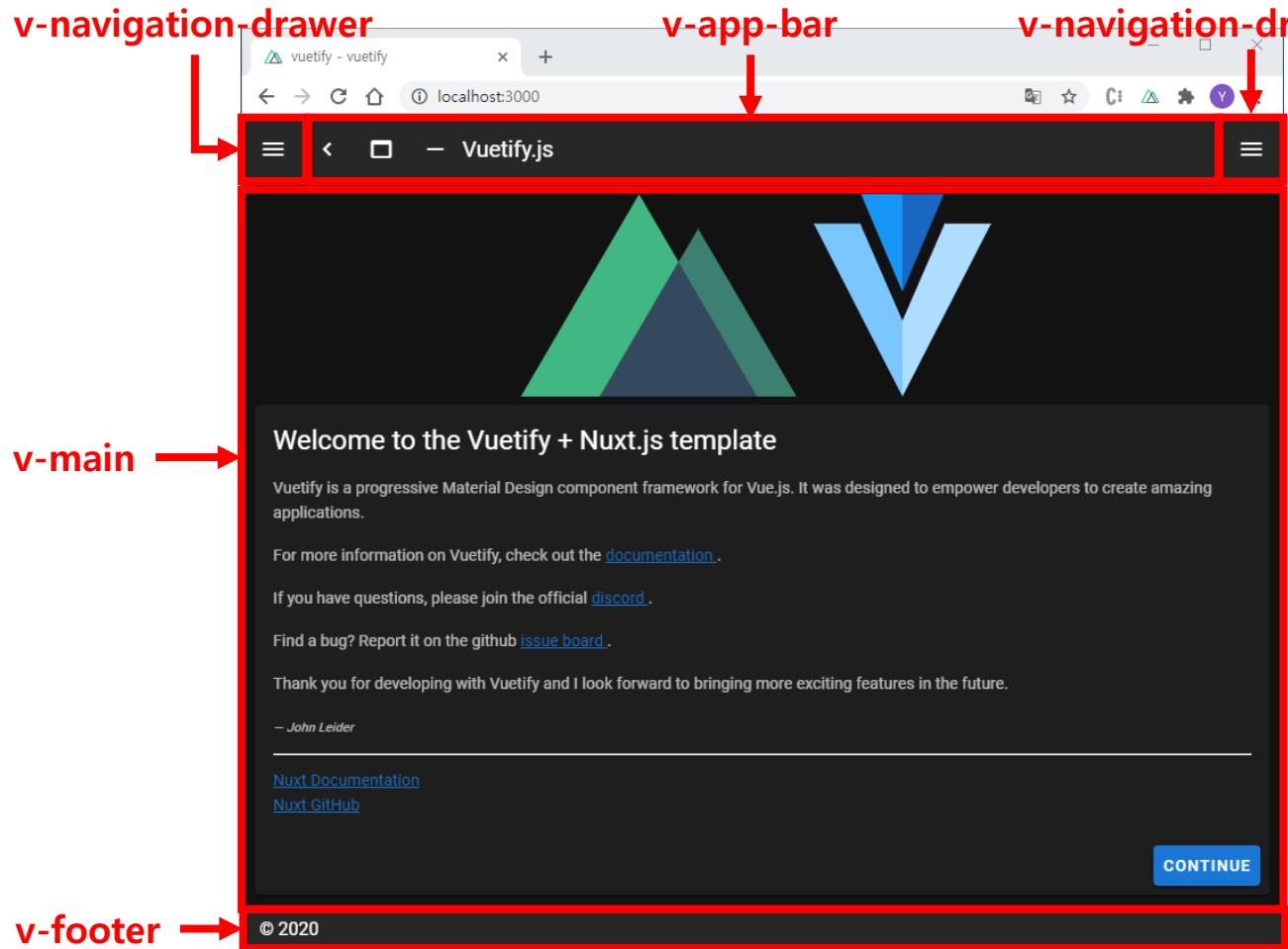
C:\> cd vuetify

C:\vuetify> **npm run dev**



6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework



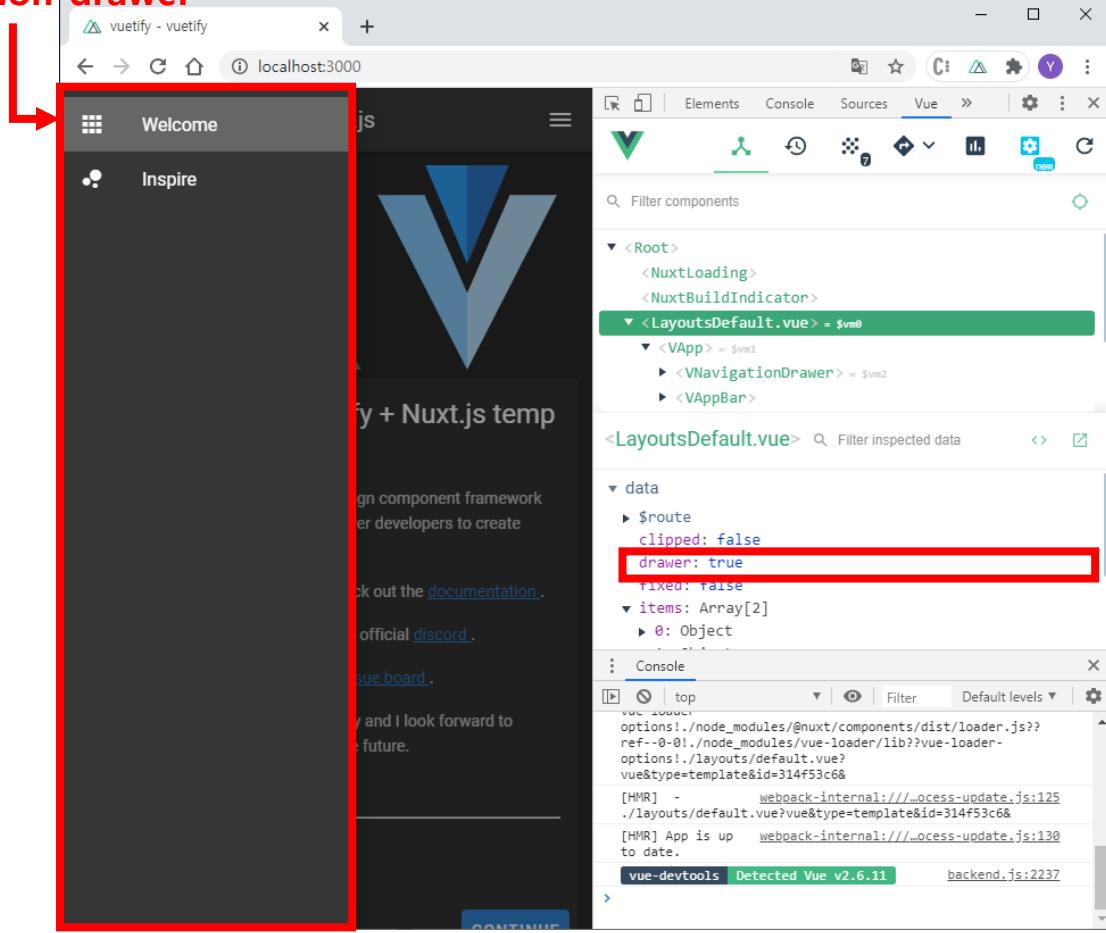
/layouts/default.vue

```
<template>
  <v-app dark>
    <v-navigation-drawer>...</v-navigation-drawer>
    <v-app-bar>... </v-app-bar>
    <v-main>
      <v-container>
        <nuxt />
      </v-container>
    </v-main>
    <v-navigation-drawer>...</v-navigation-drawer>
    <v-footer>
      <span>&copy; {{ new Date().getFullYear() }}</span>
    </v-footer>
  </v-app>
</template>
```

6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

v-navigation-drawer



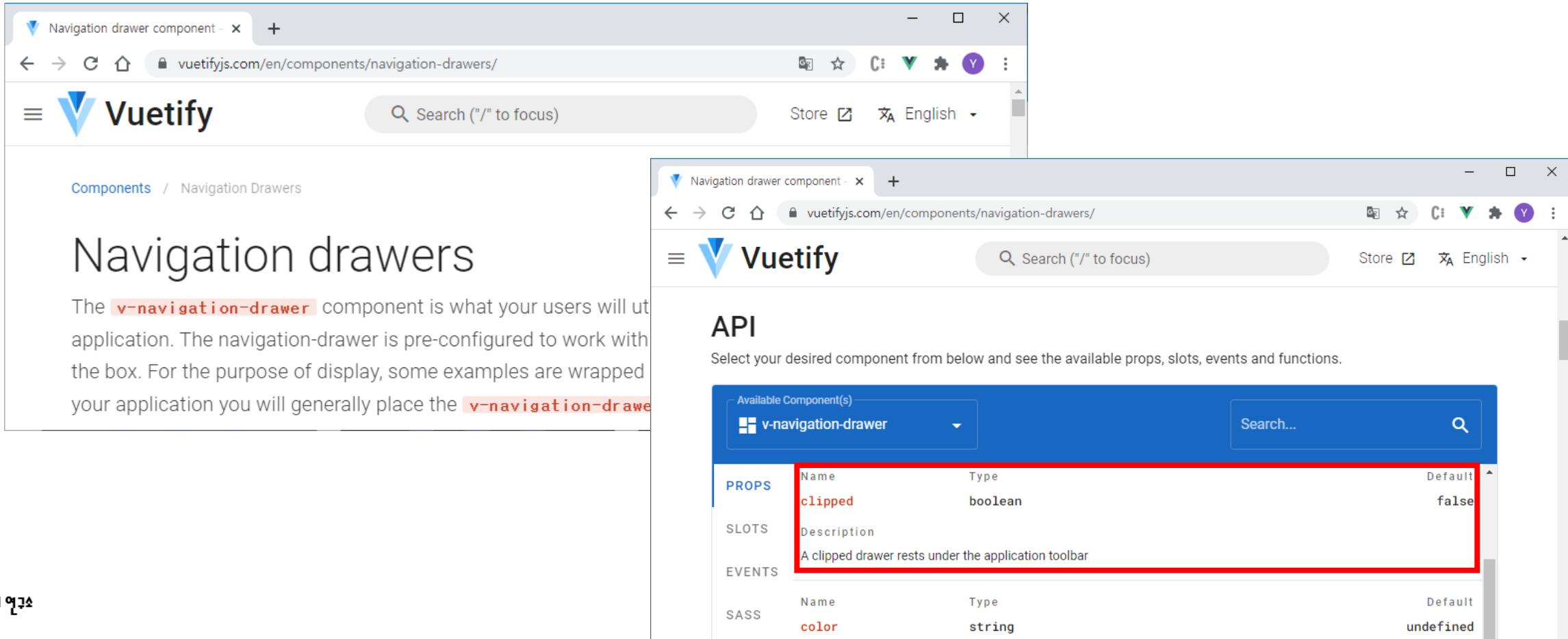
/layouts/default.vue

```
<template>
<v-app dark>
  <v-navigation-drawer v-model="drawer"
    :mini-variant="miniVariant"
    :clipped="clipped" fixed app>
    <v-list>
      <v-list-item v-for="(item, i) in items"
        :key="i"
        :to="item.to" router exact>
        <v-list-item-action>
          <v-icon>{{ item.icon }}</v-icon>
        </v-list-item-action>
        <v-list-item-content>
          <v-list-item-title v-text="item.title" />
        </v-list-item-content>
      </v-list-item>
    </v-list>
  </v-navigation-drawer>
...
<script>
export default {
  data () {
    return {
      drawer: false,
      clipped: false,
      items: [
        {
          icon: 'mdi-apps',
          title: 'Welcome',
          to: '/'
        },
        {
          icon: 'mdi-chart-bubble',
          title: 'Inspire',
          to: '/inspire'
        }
      ],
      miniVariant: false,
    }
  }
}
</script>
```

6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

<https://vuetifyjs.com/en/components/navigation-drawers/>



The screenshot shows two side-by-side browser windows displaying the Vuetify website.

Left Window (Navigation drawers component):

- Page title: Navigation drawer component
- URL: <https://vuetifyjs.com/en/components/navigation-drawers/>
- Content:
 - Vuetify logo and navigation bar.
 - Section: Components / Navigation Drawers
 - Section: Navigation drawers
 - Description: The `v-navigation-drawer` component is what your users will ultimately interact with in your application. The navigation-drawer is pre-configured to work with the rest of the Vuetify components out of the box. For the purpose of display, some examples are wrapped in a `div`. In your application you will generally place the `v-navigation-drawer` component inside a `div`.

Right Window (API documentation for v-navigation-drawer):

- Page title: Navigation drawer component
- URL: <https://vuetifyjs.com/en/components/navigation-drawers/>
- Content:
 - Vuetify logo and navigation bar.
 - Section: API
 - Description: Select your desired component from below and see the available props, slots, events and functions.
 - Available Component(s): `v-navigation-drawer`
 - Props table:

Name	Type	Default
<code>clipped</code>	boolean	false
 - Slots table:

Name	Description
<code>clipped</code>	A clipped drawer rests under the application toolbar
 - Events table:

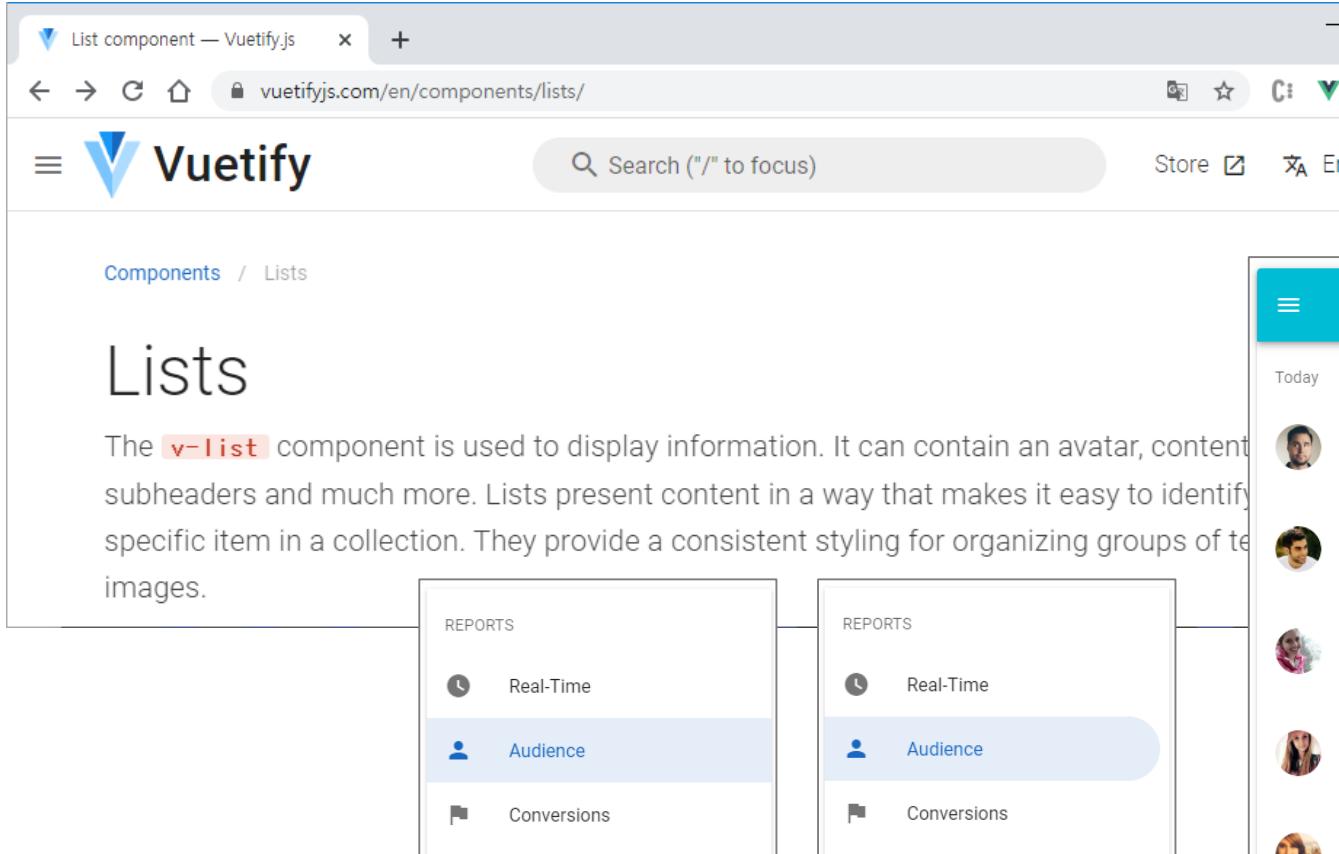
Name	Description
<code>color</code>	Color of the drawer's header
 - SASS table:

Name	Description
<code>color</code>	Color of the drawer's header

6. Vue.js / Nuxt.js / Vuetify.js

-  **Vuetify** (<https://vuetifyjs.com/>) – **Material Design** Component Framework

<https://vuetifyjs.com/en/components/lists/>



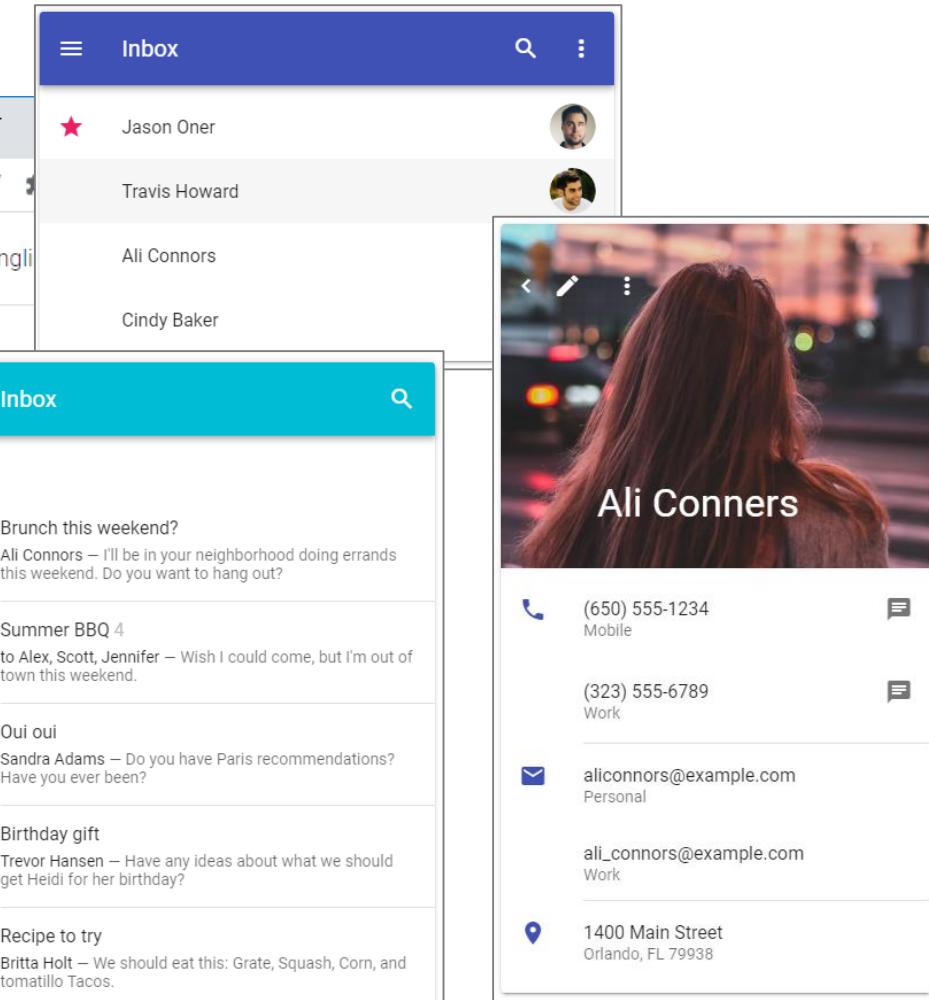
The `v-list` component is used to display information. It can contain an avatar, content, subheaders and much more. Lists present content in a way that makes it easy to identify specific item in a collection. They provide a consistent styling for organizing groups of text, images.

REPORTS

- Real-Time
- Audience
- Conversions

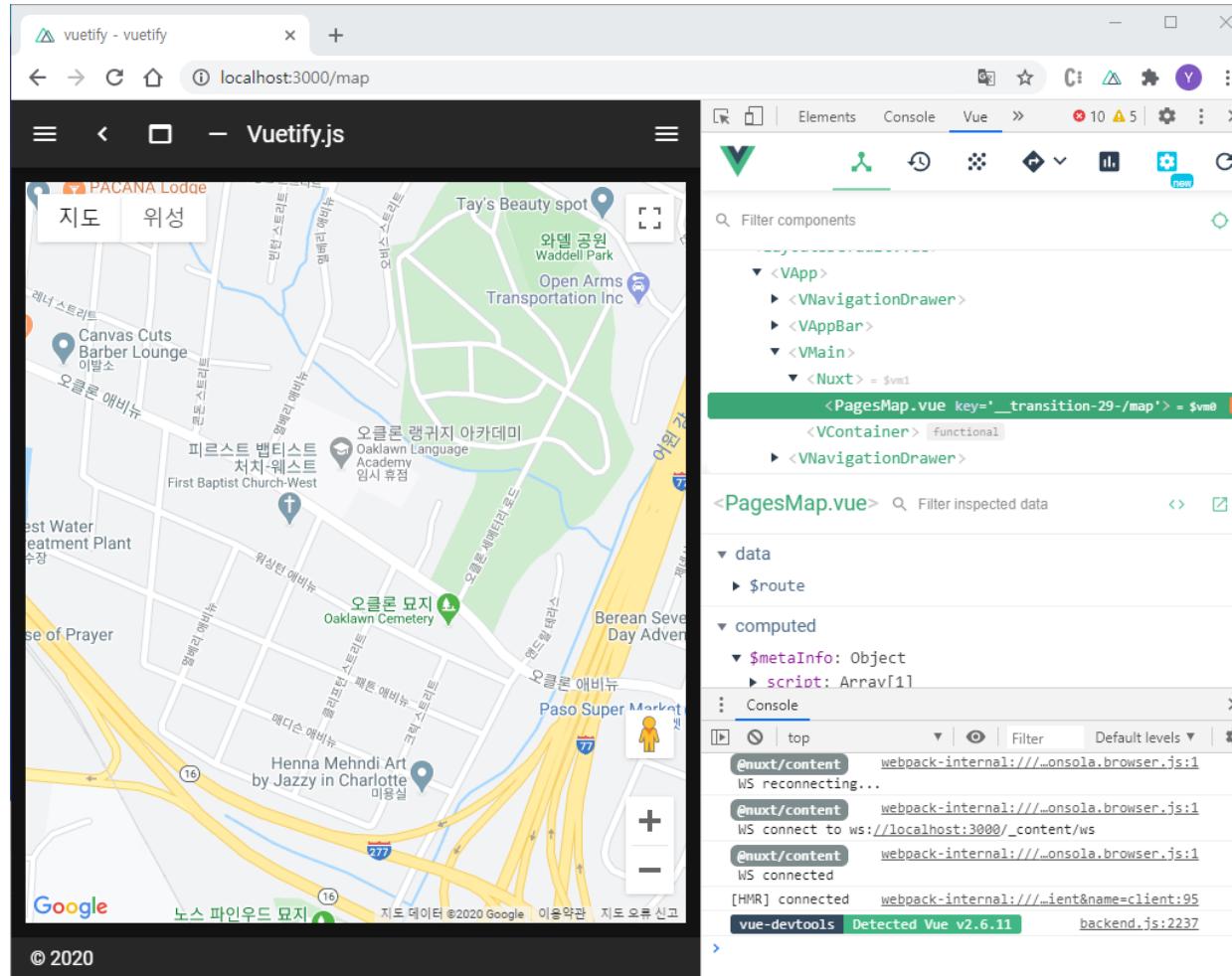
REPORTS

- Real-Time
- Audience
- Conversions



6. Vue.js / Nuxt.js / Vuetify.js

- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)



6. Vue.js / Nuxt.js / Vuetify.js

- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)

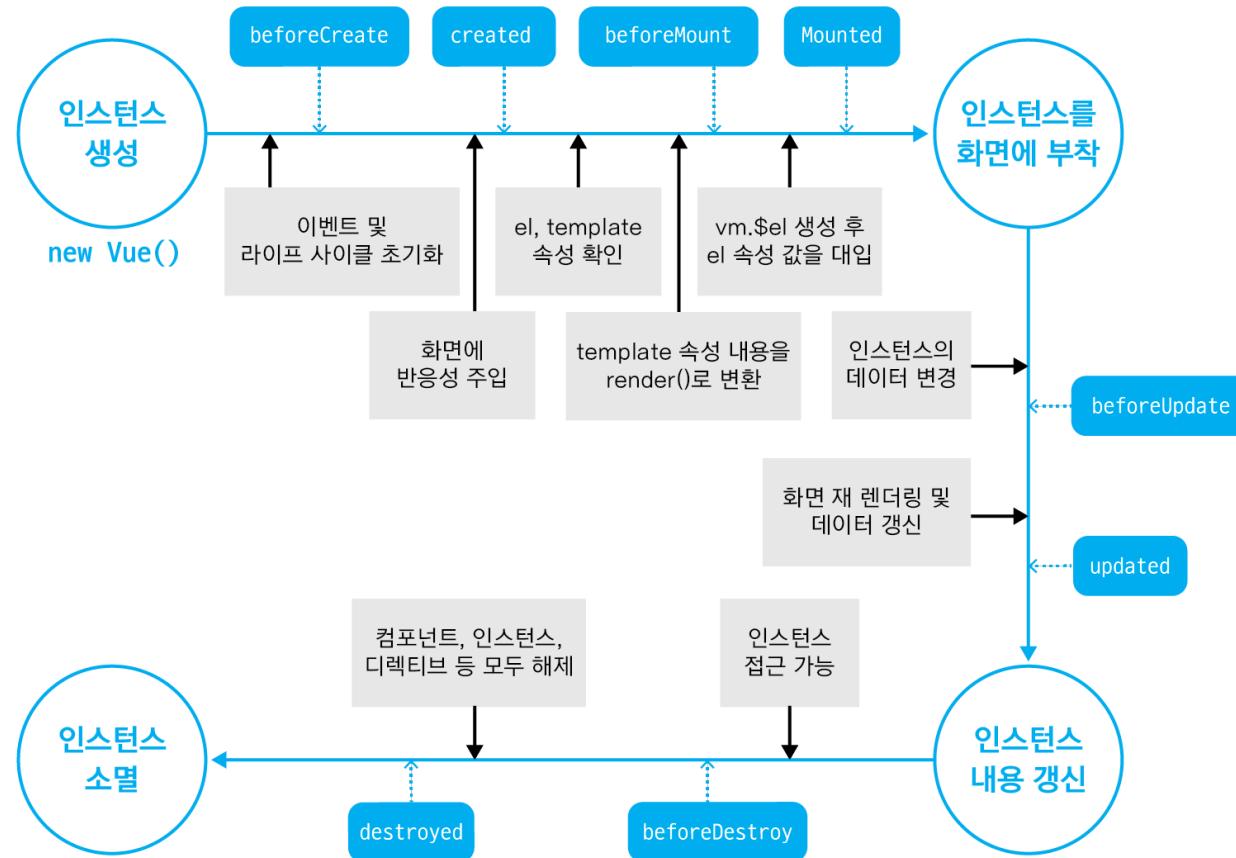
```
<template>
  <div id="map"></div>
</template>

<style>
#map {
  width: 100%;
  height: 100%;
}
</style>
```

```
<script>
export default {
  head() {
    return {
      script: [
        { src: 'https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&libraries=geometry' }
      ]
    };
  },
  mounted(){
    const map = new google.maps.Map(document.getElementById('map'), {
      center: { lat: 35.25, lng: -80.85 },
      zoom: 16
    });
  }
}
</script>
```

6. Vue.js / Nuxt.js / Vuetify.js

- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)



6. Vue.js / Nuxt.js / Vuetify.js

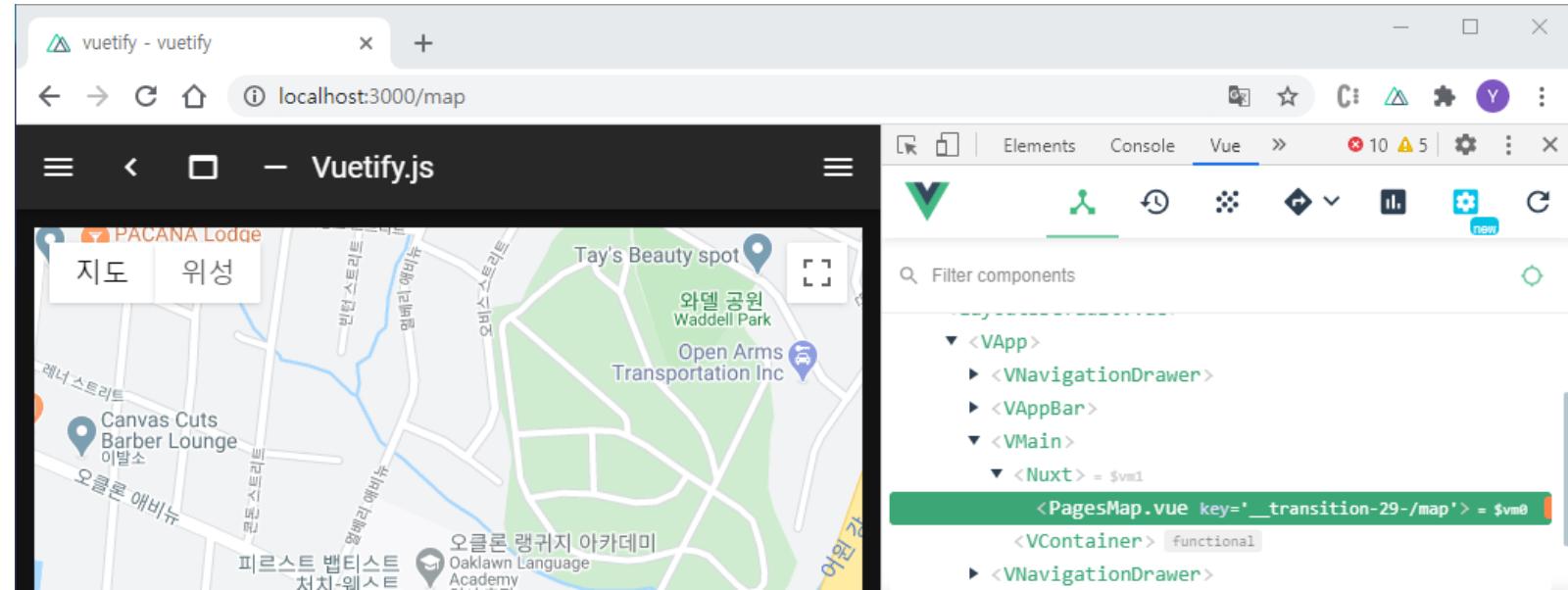
- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)

default.vue

```
<template>
  <v-app dark>
    <v-main>
      <v-container fluid fill-height>
        <nuxt />
      </v-container>
    </v-main>
  ...
</template>
```

오른쪽 생성된 Vue code를 보면
VMain의 child로 Nuxt와 VContainer가 있음.
→ VContainer를 page로 옮기는 것이 좋을 듯.

1. v-content → v-main으로 수정함(deprecated).
2. v-container에 fluid fill-height option을 설정함.



6. Vue.js / Nuxt.js / Vuetify.js

- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)

default.vue

```
<script>
export default {
  data () {
    return {
      items: [
        {
          icon: 'mdi-apps',
          title: 'Welcome',
          to: '/'
        },
        {
          icon: 'mdi-chart-bubble',
          title: 'Inspire',
          to: '/inspire'
        },
        {
          icon: 'mdi-map-search',
          title: 'Google Map',
          to: '/map'
        }
      ]
    }
  }
}
```

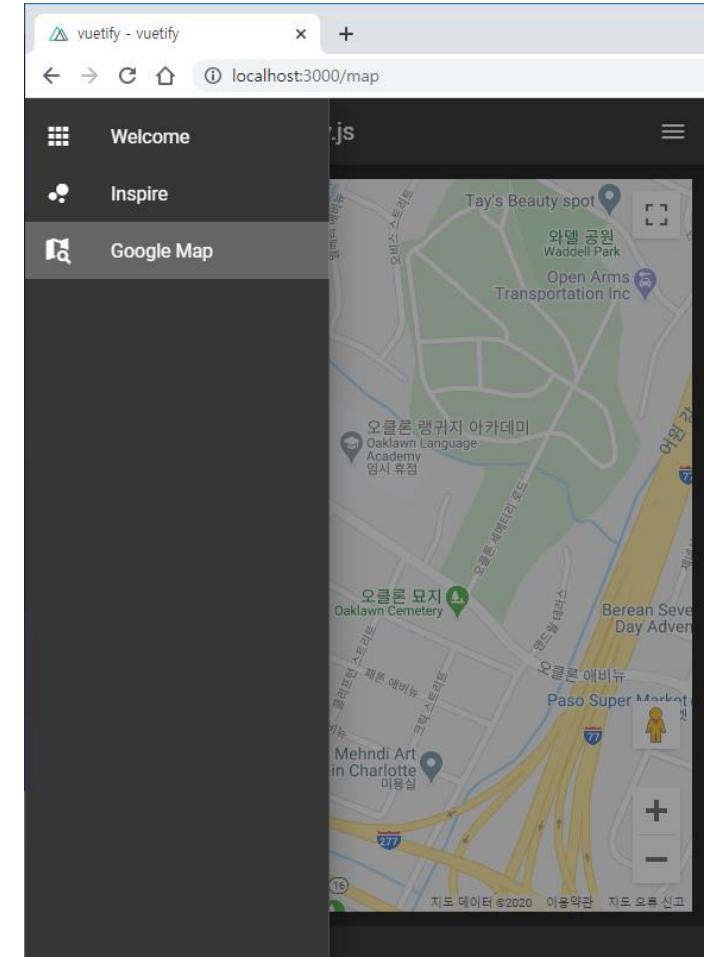
Drawer menu에 map 추가하기

mdi ← material design icon

Vuetify가 @mdi/js를 이용하는 듯.

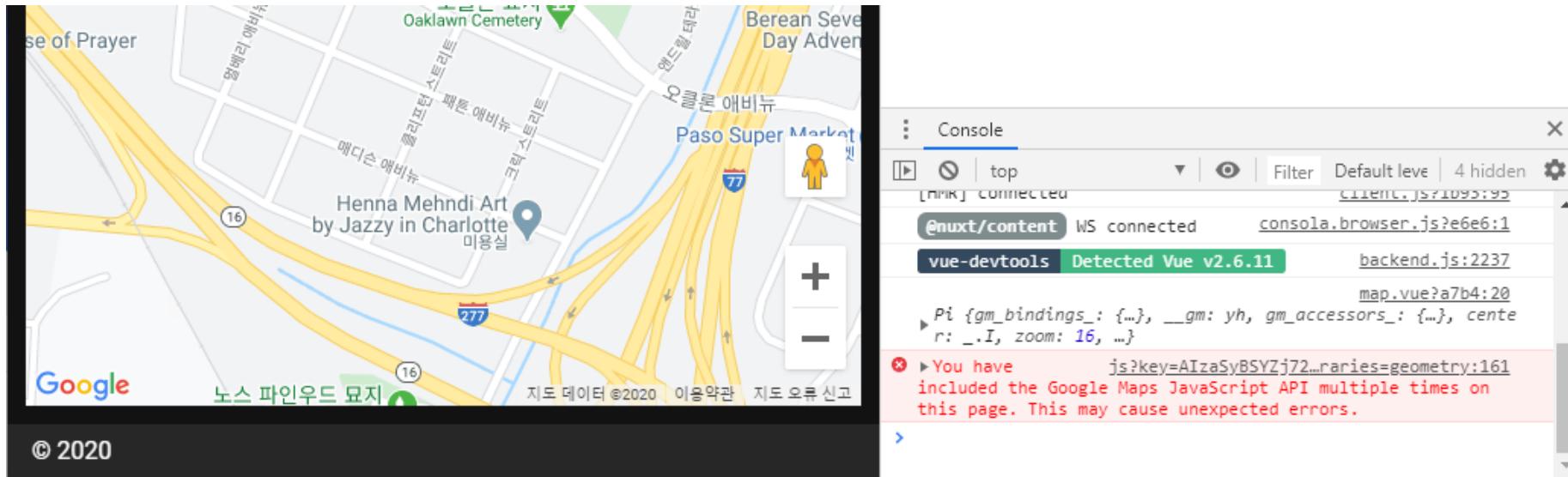
(<https://www.npmjs.com/package/@mdi/js>)

<https://materialdesignicons.com>에서 icon을
검색할 수 있음.



6. Vue.js / Nuxt.js / Vuetify.js

- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)



페이지를 이동하다 보면, map.vue가 다시 로드 되면서 Google Maps API가 반복적으로 로드 됨.

→ Google Maps API 로드를 nuxt.config.js로 옮김.

6. Vue.js / Nuxt.js / Vuetify.js

- On idle, update markers ([tutorials/vue/vuetify/pages/map.vue](#))

```
<script>
export default {
  mounted(){
    const map = new google.maps.Map(document.getElementById('map'), { center: { lat: 35.25, lng: -80.85 }, zoom: 16 });
    map.addListener('idle', function(){
      map.data.forEach(function(feature){ map.data.remove(feature); });

      const bounds = map.getBounds();
      const sw = bounds.getSouthWest(), ne = bounds.getNorthEast();

      const request = new XMLHttpRequest();
      const url = 'http://localhost:3000/api/geodata/within?' + 'left=' + sw.lng() + '&lower=' + sw.lat() + '&right=' + ne.lng() + '&upper=' + ne.lat();
      request.open('GET', url);
      request.responseType = 'json';
      request.send();

      request.onload = function() {
        const geodata = request.response;
        for(var i = 0; i < geodata.length; i++){
          const latLng = new google.maps.LatLng({ lng: geodata[i].geometry.coordinates[0], lat: geodata[i].geometry.coordinates[1] });
          map.data.add(new google.maps.Data.Feature({ geometry: latLng, properties: geodata[i].properties } ));
        }
      }
    });
  }
}
</script>
```

7. Kubernetes

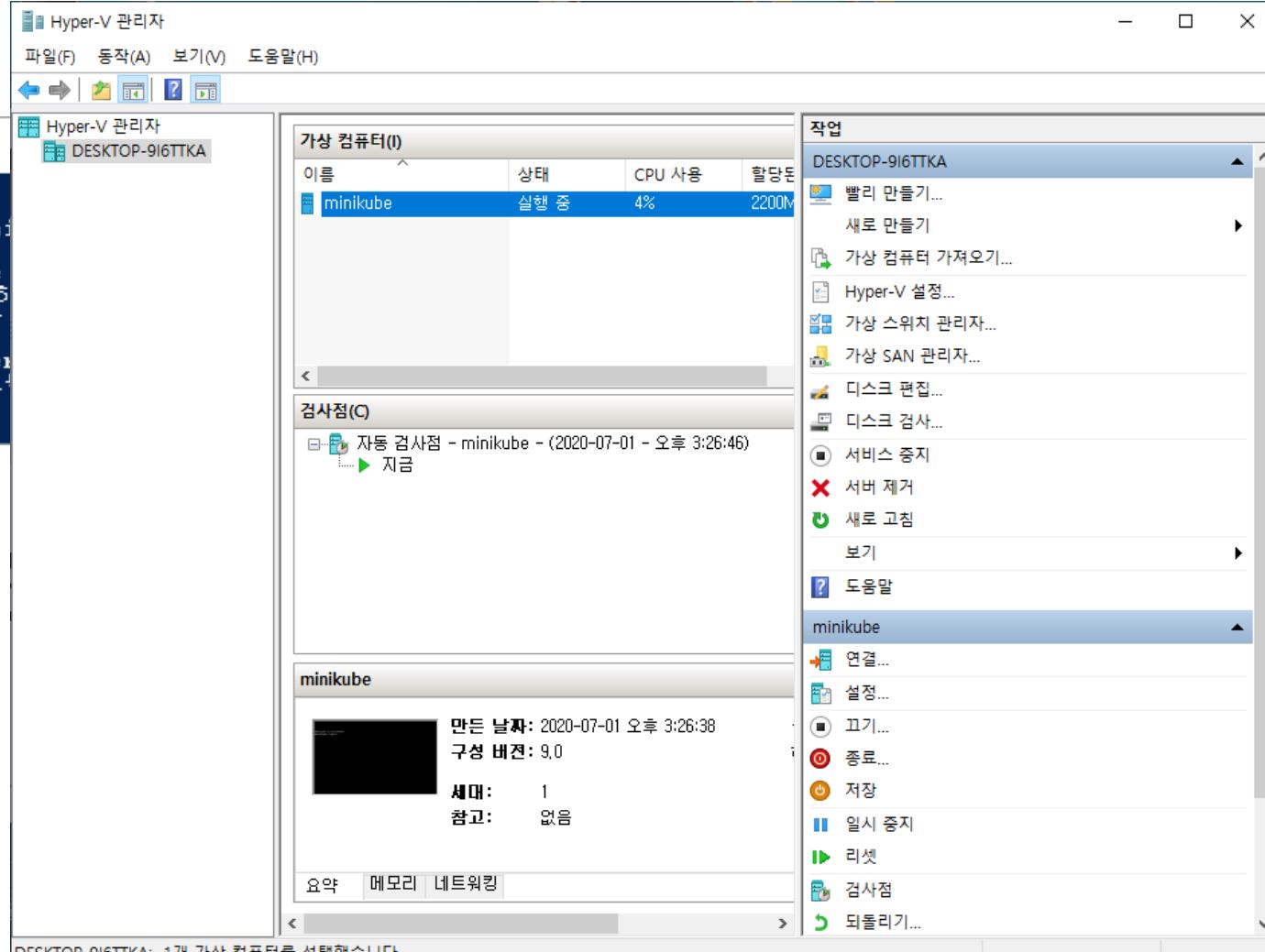
<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ Minikube 시작하기

- 1) 관리자 모드로 PowerShell을 열어서, minikube start 명령으로 minikube를 시작한다.

```
관리자: Windows PowerShell
PS C:\> minikube start
* Microsoft Windows 10 Pro 10.0.18363 Build 18363 위의 minikube VM을 찾을 수 없습니다.
* 기존 프로필에 기반하여 hyper-v 드라이버를 사용하는 중
* Starting control plane node minikube in cluster minikube
* hyper-v VM (CPUs=2, Memory=2200MB, Disk=20000MB)를 생성합니다.
* 쿠버네티스 v1.18.3 및 Docker 19.03.8 런타입으로 설치합니다.
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner
* 끝났습니다! 이제 kubectl 및 "minikube"를 사용할 수 있습니다.
PS C:\>
```

Hyper-V 관리자에서 minikube 가상 머신을 확인한다.



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ Minikube 시작하기

- 2) minikube dashboard 명령으로 Kubernetes 서비스가 활성화되었음을 확인한다.

```
관리자: Windows PowerShell
PS C:\> minikube start
* Microsoft Windows 10 Pro 10.0.18363 Build 18363 위의 minikube 시작합니다.
* 기존 프로필에 기반하여 hyperv 드라이버를 사용하는 중입니다.
* Starting control plane node minikube in cluster minikube...
* hyperv VM (CPUs=2, Memory=2200MB, Disk=20000MB)를 생성합니다.
* 쿠버네티스 v1.18.3 및 Docker 19.03.8 런타임으로 설치합니다.
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner...
* 출발했습니다! 이제 kubectl 및 "minikube"를 사용할 수 있습니다.
PS C:\>
PS C:\> minikube dashboard
* 대시보드를 활성화하는 중...
* Verifying dashboard health...
* 프록시를 시작하는 중...
* Verifying proxy health...
* Opening http://127.0.0.1:62152/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/#/
    Your default browser...
```

Kubernetes Dashboard를 확인한다.

The screenshot shows the Kubernetes Dashboard interface. On the left, there's a sidebar with navigation links: 클러스터 (Cluster), 클러스터 툴 (Cluster Tools), 네임스페이스 (Namespaces), 노드 (Nodes), 퍼시스턴트 볼륨 (Persistent Volumes), 스토리지 클래스 (Storage Classes), 네임스페이스 (Namespaces), default (Default), 개요 (Overview), 워크로드 (Workloads), 크론 잡 (Cron Jobs), 데몬 세트 (Daemon Sets), 디플로이먼트 (Deployments), and 잡 (Jobs). The main content area is titled 'Overview' and includes sections for '클러스터' (Cluster), '디스크버리 및 로드 밸런싱' (Discovery and Load Balancing), '서비스' (Services), '컨피그 및 스토리지' (Config and Storage), and '시크릿' (Secrets). The '서비스' section lists one service named 'kubernetes' with details: 네임스페이스 (Namespace): default, 레이블 (Labels): component: apiserver, provider: kubernetes, 클러스터 IP (Cluster IP): 10.96.0.1, 내부 엔드포인트 (Internal Endpoints): kubernetes:443, 외부 엔드포인트 (External Endpoints): kubernetes:443, 생성 시간 (Creation Time): 11 minutes ago. The '시크릿' section lists one secret named 'default-token-4tnhn' with details: 네임스페이스 (Namespace): default, 타입 (Type): kubernetes.io/service-account-token, 생성 시간 (Creation Time): 11 minutes ago.

이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 kubernetes:443	kubernetes:443 kubernetes:443	11 minutes ago

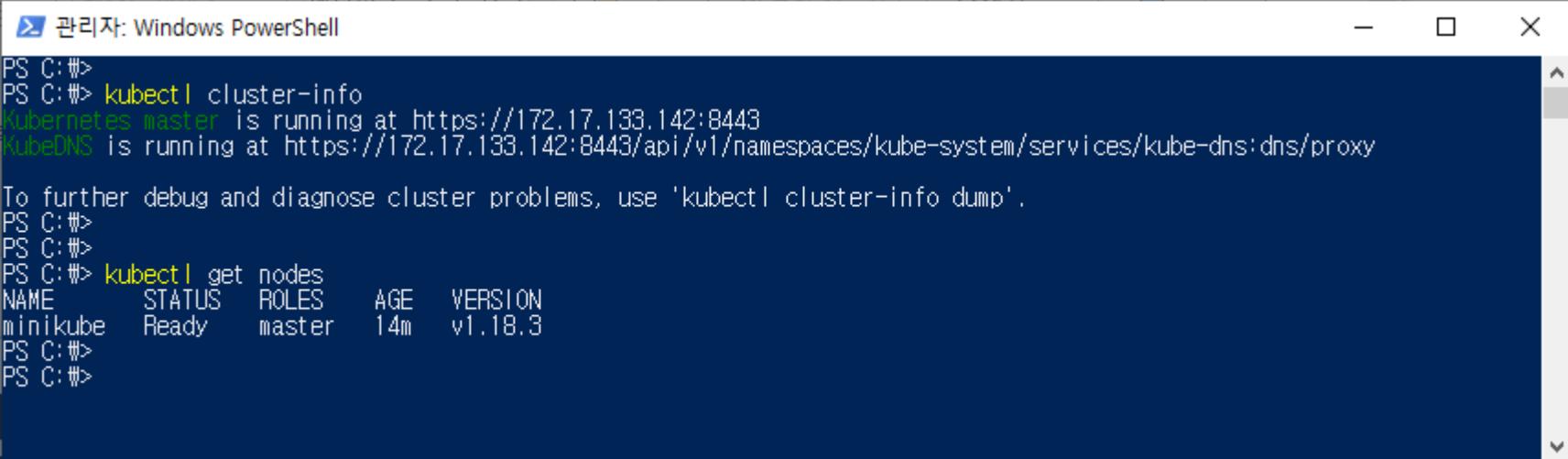
이름	네임스페이스	레이블	타입	생성 시간
default-token-4tnhn	default	-	kubernetes.io/service-account-token	11 minutes ago

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ Minikube 시작하기

3) kubectl 명령으로 cluster와 node 정보를 확인한다.



```
PS C:\#> kubectl cluster-info
Kubernetes master is running at https://172.17.133.142:8443
KubeDNS is running at https://172.17.133.142:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
PS C:\#>
PS C:\#>
PS C:\#> kubectl get nodes
NAME     STATUS   ROLES    AGE    VERSION
minikube Ready    master   14m    v1.18.3
PS C:\#>
PS C:\#>
```

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

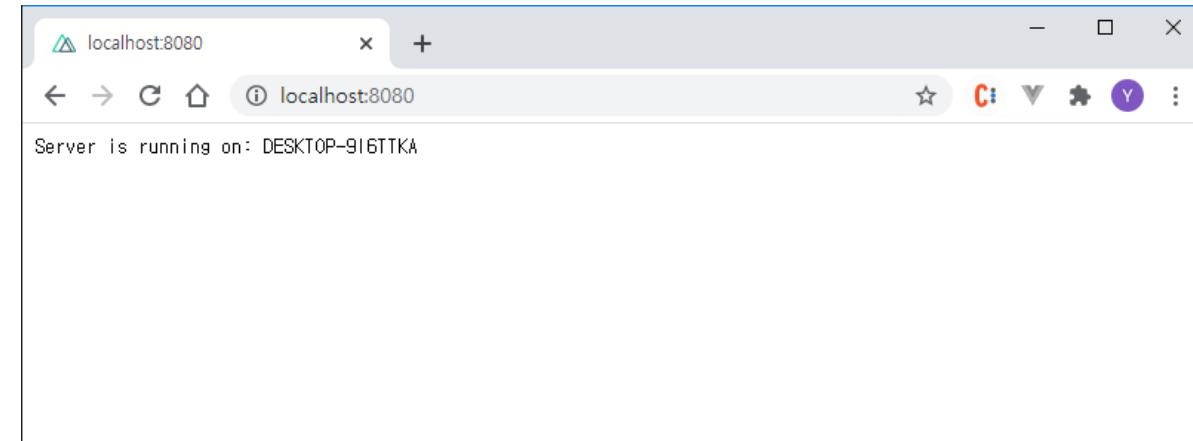
■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

1) hostname을 출력하는 웹 서버

```
var http = require('http');
var os = require('os');

var host = os.hostname();
var handleRequest = function(request, response) {
    console.log('Received request for URL: ' + request.url);
    response.writeHead(200);
    response.end('Server is running on: ' + host);
};

var www = http.createServer(handleRequest);
www.listen(8080);
```

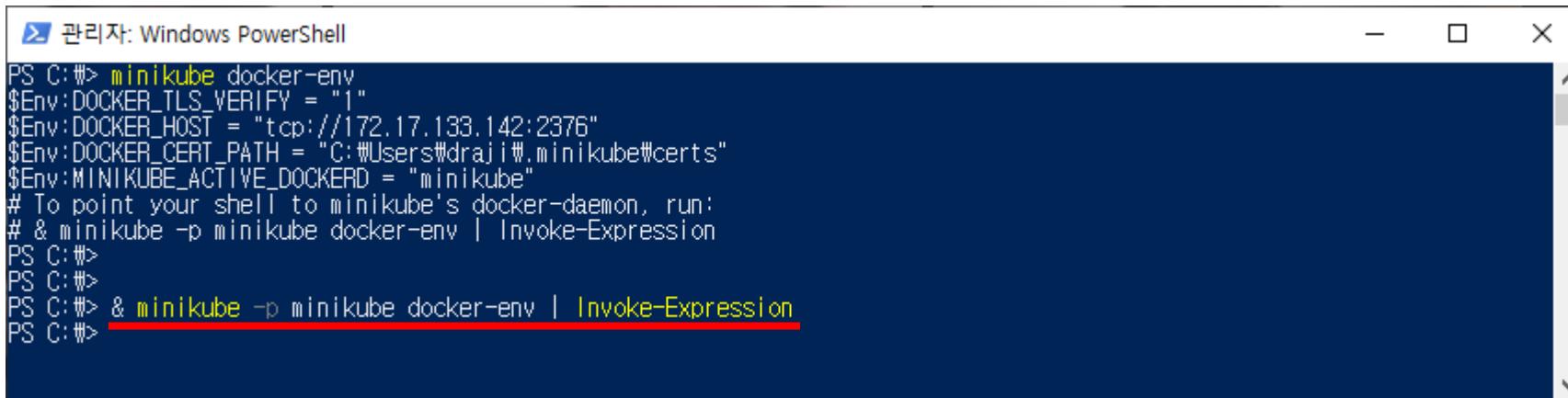


7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

2) Docker 환경 설정



```
PS C:\> minikube docker-env
$Env:DOCKER_TLS_VERIFY = "1"
$Env:DOCKER_HOST = "tcp://172.17.133.142:2376"
$Env:DOCKER_CERT_PATH = "C:\Users\draji\minikube\certs"
$Env:MINIKUBE_ACTIVE_DOCKERD = "minikube"
# To point your shell to minikube's docker-daemon, run:
# & minikube -p minikube docker-env | Invoke-Expression
PS C:\>
PS C:\>
PS C:\> & minikube -p minikube docker-env | Invoke-Expression
PS C:\>
```

minikube docker-env 명령으로 docker 환경 설정 방법을 알 수 있다.

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

3) Docker 이미지 만들기

dockerfile1

```
FROM node:12.18.0  
EXPOSE 8080  
COPY server1.js .  
CMD node server1.js
```

docker build -t hostname:v1 -f dockerfile1 . 명령으로 이미지를 빌드한다.

docker images 명령으로 이미지를 확인한다.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hostname	v1	09b402b313d7	About a minute ago	918MB
node	12.18.0	b31738bd2a38	3 weeks ago	918MB
k8s.gcr.io/kube-proxy	v1.18.3	3439b7546f29	5 weeks ago	117MB
k8s.gcr.io/kube-apiserver	v1.18.3	7e28efa976bd	5 weeks ago	173MB
k8s.gcr.io/kube-controller-manager	v1.18.3	da26705ccb4b	5 weeks ago	162MB
k8s.gcr.io/kube-scheduler	v1.18.3	76216c34ed0c	5 weeks ago	95.3MB
kubernetesui/dashboard	v2.0.0	8b32422733b3	2 months ago	222MB
k8s.gcr.io/pause	3.2	80d28bedfe5d	4 months ago	683kB
k8s.gcr.io/coredns	1.6.7	67da37a9a360	5 months ago	43.8MB
k8s.gcr.io/etcdd	3.4.3-0	303ce5db0e90	8 months ago	288MB
kubernetesui/metrics-scrapers	v1.0.2	3b08661dc379	8 months ago	40.1MB
gcr.io/k8s-minikube/storage-provisioner	v1.8.1	4689081edb10	2 years ago	80.8MB

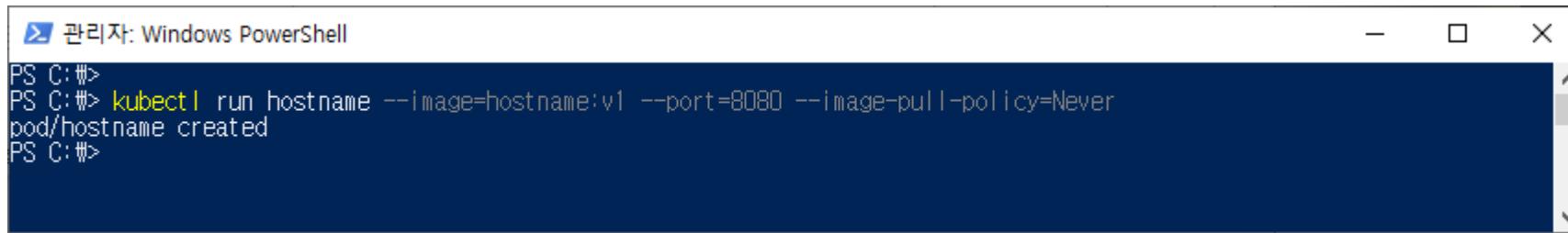
7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

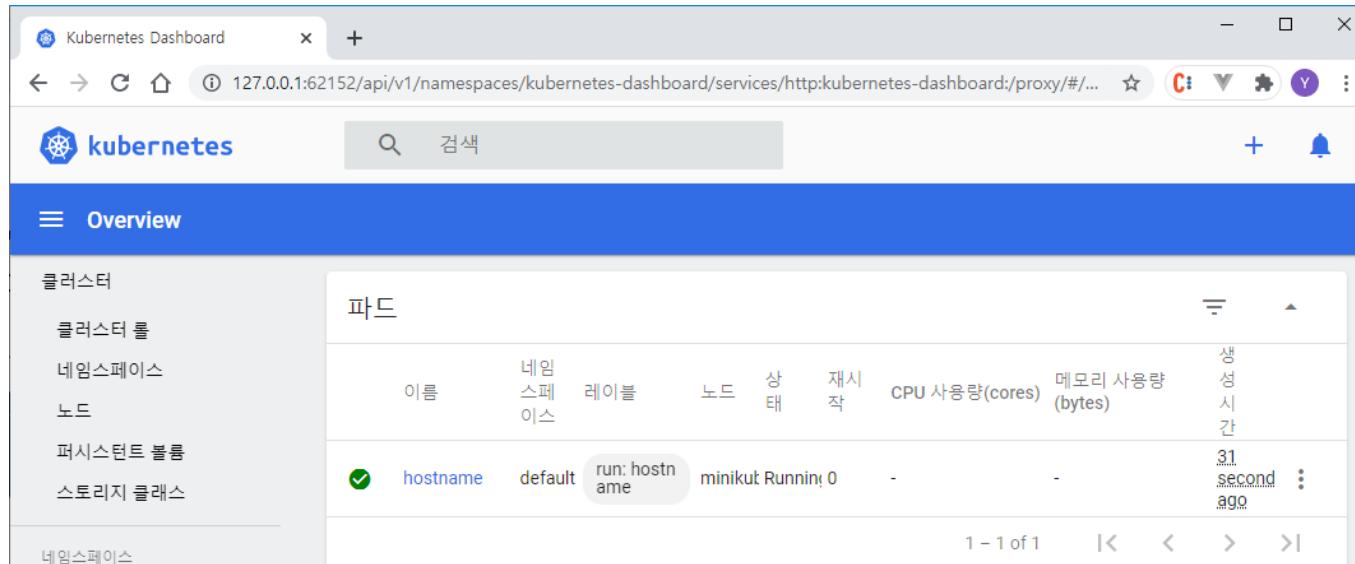
4) POD 생성하기

kubectl run 명령으로 POD를 생성한다.



```
PS C:\#>
PS C:\#> kubectl run hostname --image=hostname:v1 --port=8080 --image-pull-policy=Never
pod/hostname created
PS C:\#>
```

dashboard에서 생성된 POD를 확인할 수 있다.



The screenshot shows the Kubernetes Dashboard's Overview page. On the left, there is a sidebar with navigation links: 클러스터, 클러스터 룰, 네임스페이스, 노드, 퍼시스턴트 블룸, and 스토리지 클래스. The main area has a title "파드" (Pods). A table lists one pod: "hostname" (Status: Running, Node: minikube, Image: run:hostname, CPU: 0 cores, Memory: 31 bytes, Created: 31 seconds ago).

이름	네임스페이스	레이블	노드	상태	재시작	CPU 사용량(cores)	메모리 사용량(bytes)	생성시간
hostname	default	run:hostname	minikube	Running	0	-	-	31 second ago

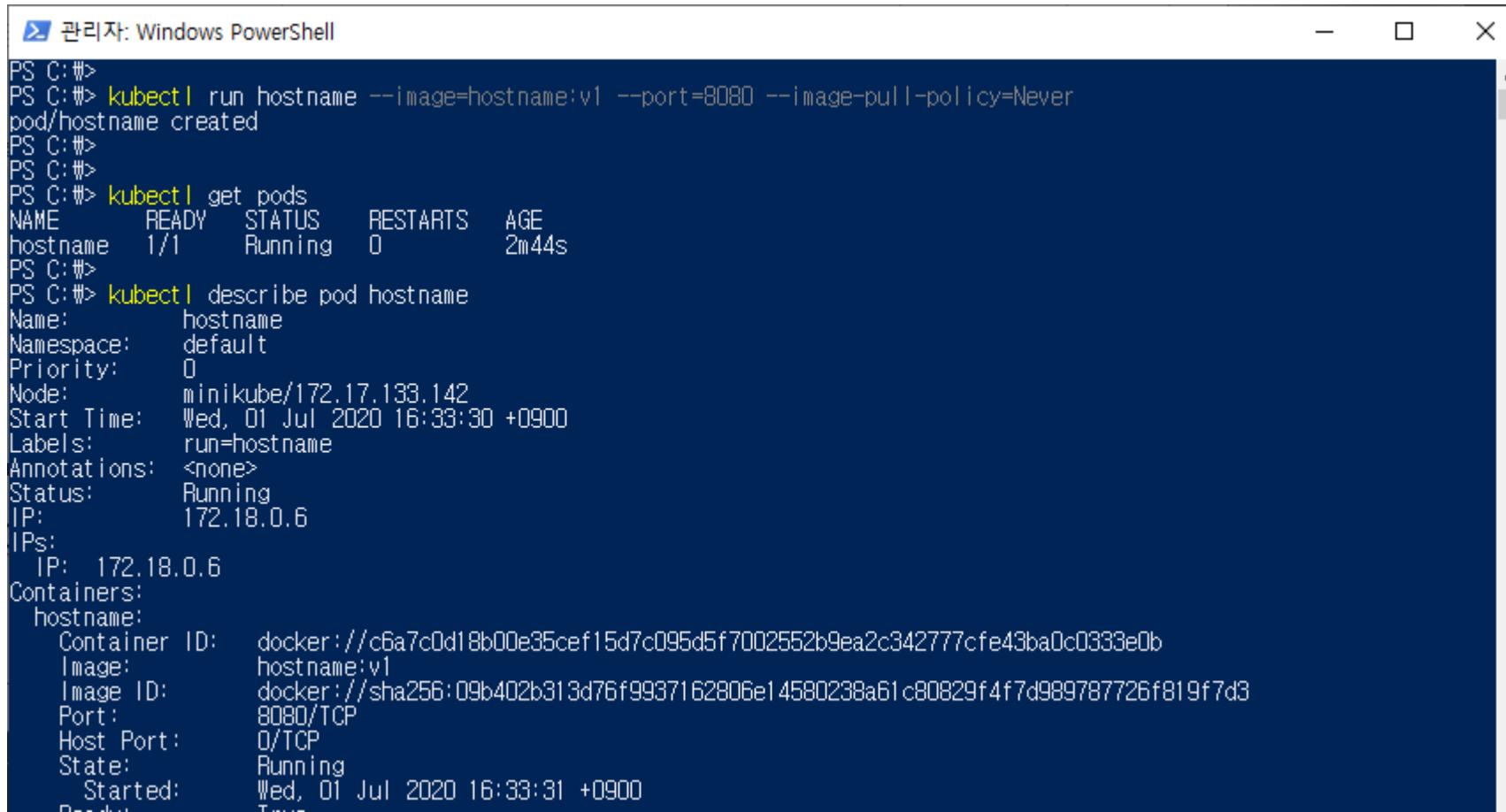
7. Kubernetes

<https://kubernetes.io/ko/docs/concepts/workloads/pods/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

4) POD 생성하기

kubectl get pods, kubectl describe pod {POD Name}으로 pod 정보를 확인할 수 있다.



```
PS C:\#> kubectl run hostname --image=hostname:v1 --port=8080 --image-pull-policy=Never
pod/hostname created
PS C:\#>
PS C:\#>
PS C:\#> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
hostname  1/1     Running   0          2m44s
PS C:\#>
PS C:\#> kubectl describe pod hostname
Name:           hostname
Namespace:      default
Priority:       0
Node:          minikube/172.17.133.142
Start Time:    Wed, 01 Jul 2020 16:33:30 +0900
Labels:         run=hostname
Annotations:   <none>
Status:        Running
IP:            172.18.0.6
IPs:
  IP: 172.18.0.6
Containers:
  hostname:
    Container ID:  docker://c6a7c0d18b00e35cef15d7c095d5f7002552b9ea2c342777cf43ba0c0333e0b
    Image:         hostname:v1
    Image ID:     docker://sha256:09b402b313d76f9937162806e14580238a61c80829f4f7d989787726f819f7d3
    Port:          8080/TCP
    Host Port:    0/TCP
    State:        Running
    Started:     Wed, 01 Jul 2020 16:33:31 +0900
    Ready:        True
```

7. Kubernetes

<https://kubernetes.io/ko/docs/concepts/services-networking/>

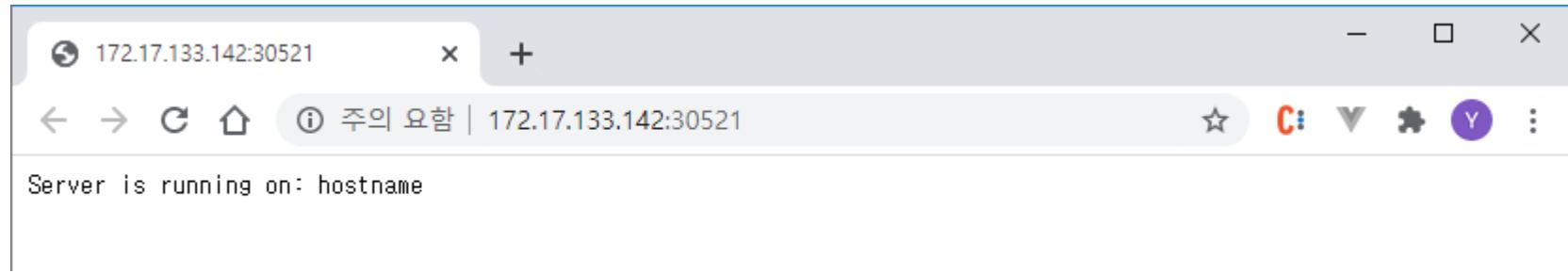
■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

5) 서비스를 생성해서 POD를 노출하기

kubectl expose pod hostname --type=LoadBalancer 명령으로 pod를 노출시키는 service를 생성한다.

minikube service hostname 명령으로 서비스 연결을 테스트할 수 있다.

```
관리자: Windows PowerShell
PS C:\#> kubectl expose pod hostname --type=LoadBalancer
service/hostname exposed
PS C:\#>
PS C:\#> minikube service hostname
|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|
| default   | hostname | 8080 | http://172.17.133.142:30521 |
|-----|
* Opening service default/hostname in default browser...
PS C:\#>
```



7. Kubernetes

<https://kubernetes.io/ko/docs/concepts/services-networking/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

5) 서비스를 생성해서 POD를 노출하기

kubectl expose pod hostname --type=LoadBalancer 명령으로 pod를 노출시키는 service를 생성한다.

minikube service hostname 명령으로 서비스 연결을 테스트할 수 있다.

dashboard에서 확인할 수 있습니다.

The screenshot shows the Kubernetes Dashboard interface. The top navigation bar includes a back button, forward button, refresh button, and a search bar with the placeholder '검색' (Search). The main header has the 'kubernetes' logo and a search icon. Below the header, a blue navigation bar displays the title 'Overview'. On the left side, there is a sidebar with the following menu items: '클러스터' (Cluster), '클러스터 툴' (Cluster Tools), '네임스페이스' (Namespace), '노드' (Node), '퍼시스턴트 볼륨' (Persistent Volume), and '스토리지 클래스' (Storage Classes). The main content area is titled '서비스' (Services) and contains a table with the following data:

이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간
hostname	default	run: hostname	10.99.169.42	hostname:8080	hostname:30521	17 minutes ago

7. Kubernetes

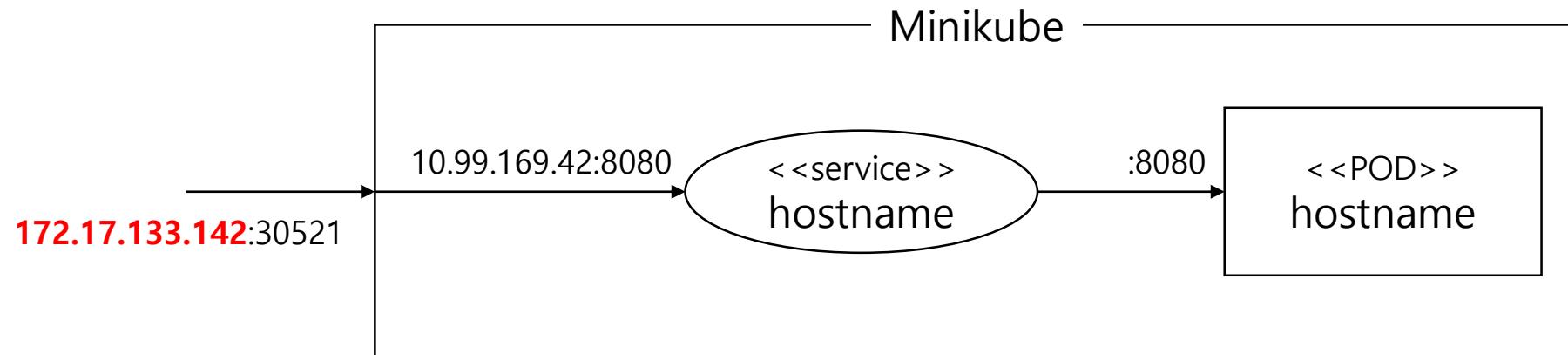
<https://kubernetes.io/ko/docs/concepts/services-networking/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

5) 서비스를 생성해서 POD를 노출하기

kubectl expose pod hostname --type=LoadBalancer 명령으로 pod를 노출시키는 service를 생성한다.

minikube service hostname 명령으로 서비스 연결을 테스트할 수 있다.



서비스	이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간	⋮
	hostname	default	run: hostname	10.99.169.42	hostname:8080 TCP	hostname:30521 TCP	17 minutes ago	⋮

7. Kubernetes

<https://kubernetes.io/ko/docs/concepts/workloads/controllers/deployment/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

6) Deployment 컨트롤러 생성하기

kubectl delete pods,services --all 명령으로 생성된 POD와 서비스를 모두 제거할 수 있습니다.

kubectl create deployment hostname --image=hostname:v1 명령으로 deployment 컨트롤러를 생성한다.

The screenshot displays two windows side-by-side. On the left is a Windows PowerShell window titled '관리자: Windows PowerShell' showing the command: `kubectl create deployment hostname --image=hostname:v1`. The output shows: `deployment.apps/hostname created`. On the right is a screenshot of the 'Kubernetes Dashboard' browser interface. The dashboard shows three main sections: Overview, Deployments, and Pods. In the Overview section, there is a summary of cluster resources. The Deployments section lists one deployment named 'hostname' with details: Name: hostname, Namespace: default, Labels: app: hostname, Pods: 1 / 1, Created: a minute ago, Image: hostname:v1. The Pods section lists one pod named 'hostname-748d5d5d49-bzzjn' with details: Name: hostname-748d5d5d49-bzzjn, Namespace: default, Labels: app: hostname, Pod template hash: 748d5d5d49, Node: minikube, Status: Running, CPU usage: 0, Memory usage: a minute ago. The Pods table also includes columns for Name, Namespace, Labels, Node, Status, Start Time, CPU Usage (cores), Memory Usage (bytes), and Creation Time.

dashboard에서 deployment, pod, replica set이 생성됨을 확인할 수 있다.

Deployment는 POD와 replica set을 관리한다.

Replica set은 POD의 복제를 관리한다.

Deployment 또는 replica set에서 scale을 변경해 보면, POD 개수가 조정됨을 알 수 있다.

7. Kubernetes

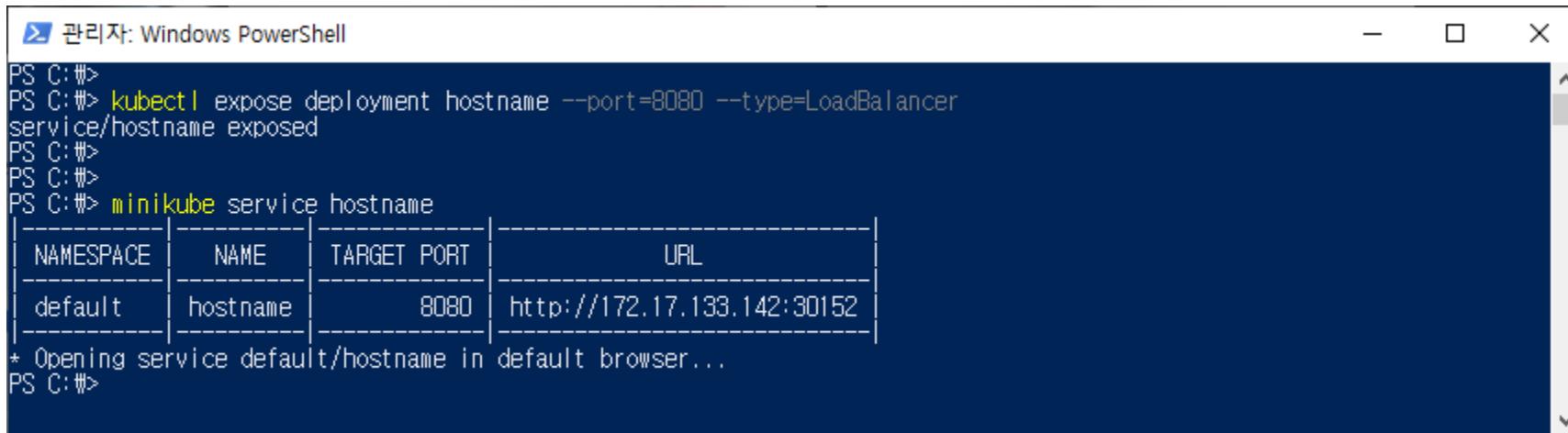
<https://kubernetes.io/ko/docs/concepts/services-networking/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

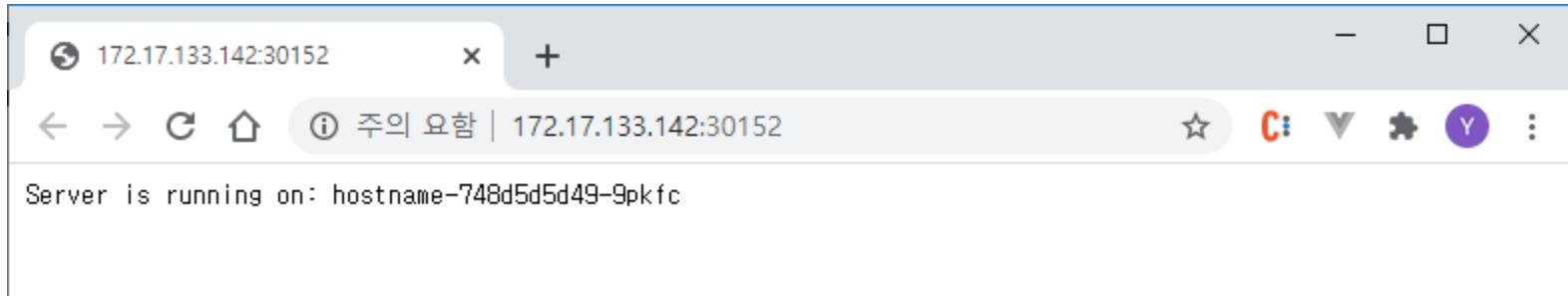
7) 서비스를 생성해서 deployment를 노출하기

kubectl expose deployment hostname --port=8080 --type=LoadBalancer 명령으로 service를 생성한다.

minikube service hostname 명령으로 서비스 연결을 테스트할 수 있다.



```
PS C:\#> kubectl expose deployment hostname --port=8080 --type=LoadBalancer
service/hostname exposed
PS C:\#>
PS C:\#>
PS C:\#> minikube service hostname
|-----+-----+-----+-----|
| NAMESPACE | NAME | TARGET PORT | URL           |
|-----+-----+-----+-----|
| default   | hostname | 8080 | http://172.17.133.142:30152 |
|-----+-----+-----+-----|
* Opening service default/hostname in default browser...
PS C:\#>
```



7. Kubernetes

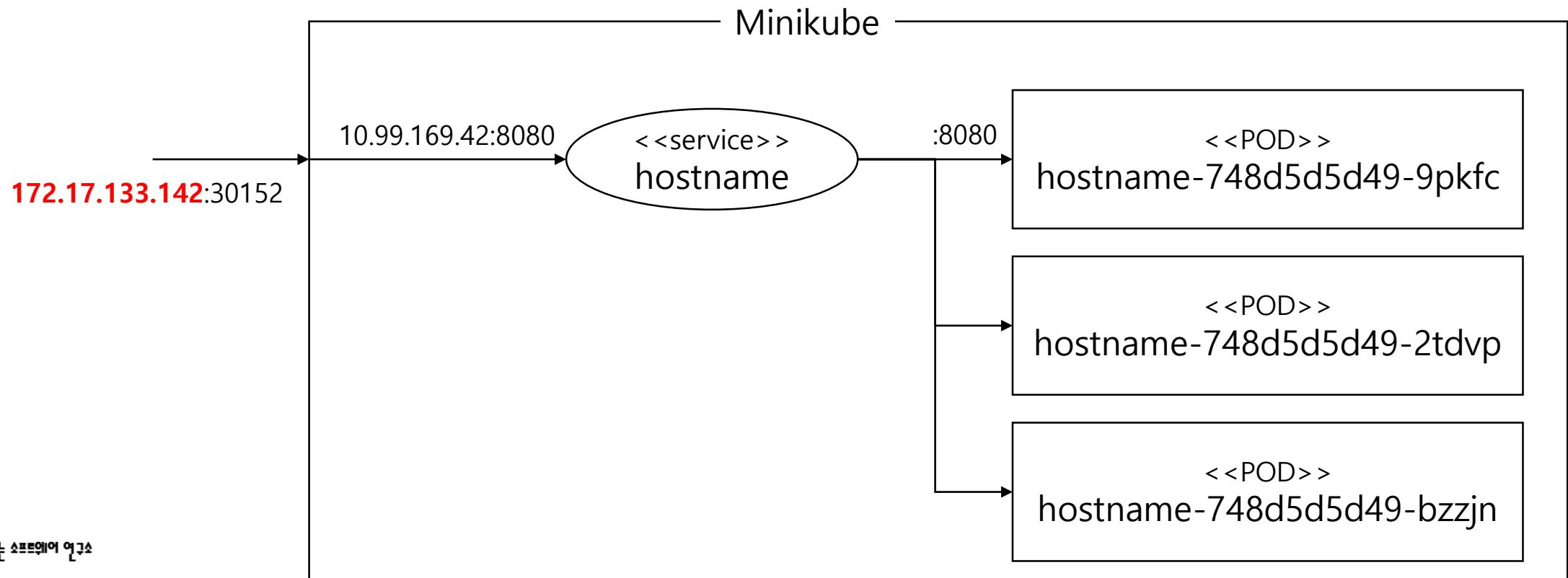
<https://kubernetes.io/ko/docs/concepts/services-networking/>

■ 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

7) 서비스를 생성해서 deployment를 노출하기

kubectl expose deployment hostname --port=8080 --type=LoadBalancer 명령으로 service를 생성한다.

minikube service hostname 명령으로 서비스 연결을 테스트할 수 있다.



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

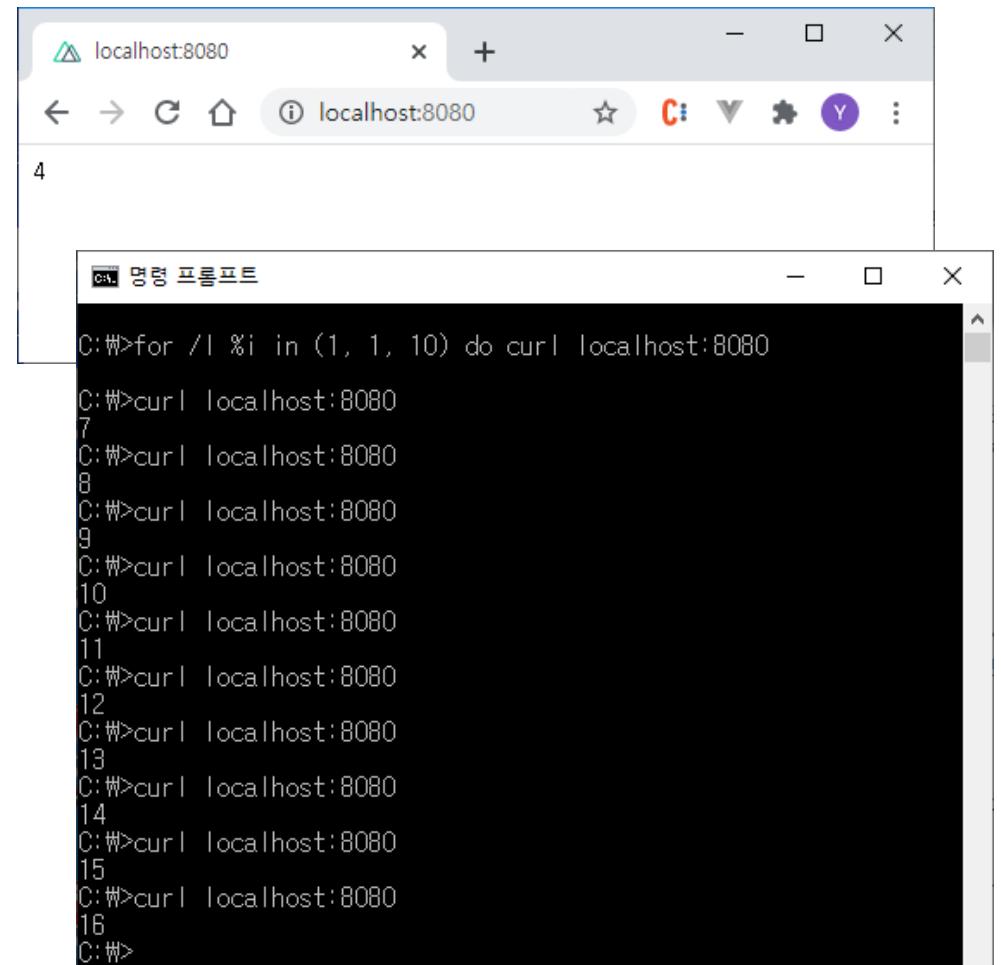
■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

- 1) 방문할 때마다 count를 증가시킴

```
var count = 0;

var http = require('http');
http.createServer(function(request, response) {
    count = count + 1;

    response.writeHead(200);
    response.end(count.toString());
}).listen(8080);
```



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

- 2) 방문할 때마다 count를 증가시킴 – count.txt에 저장함

```
var count = 0;

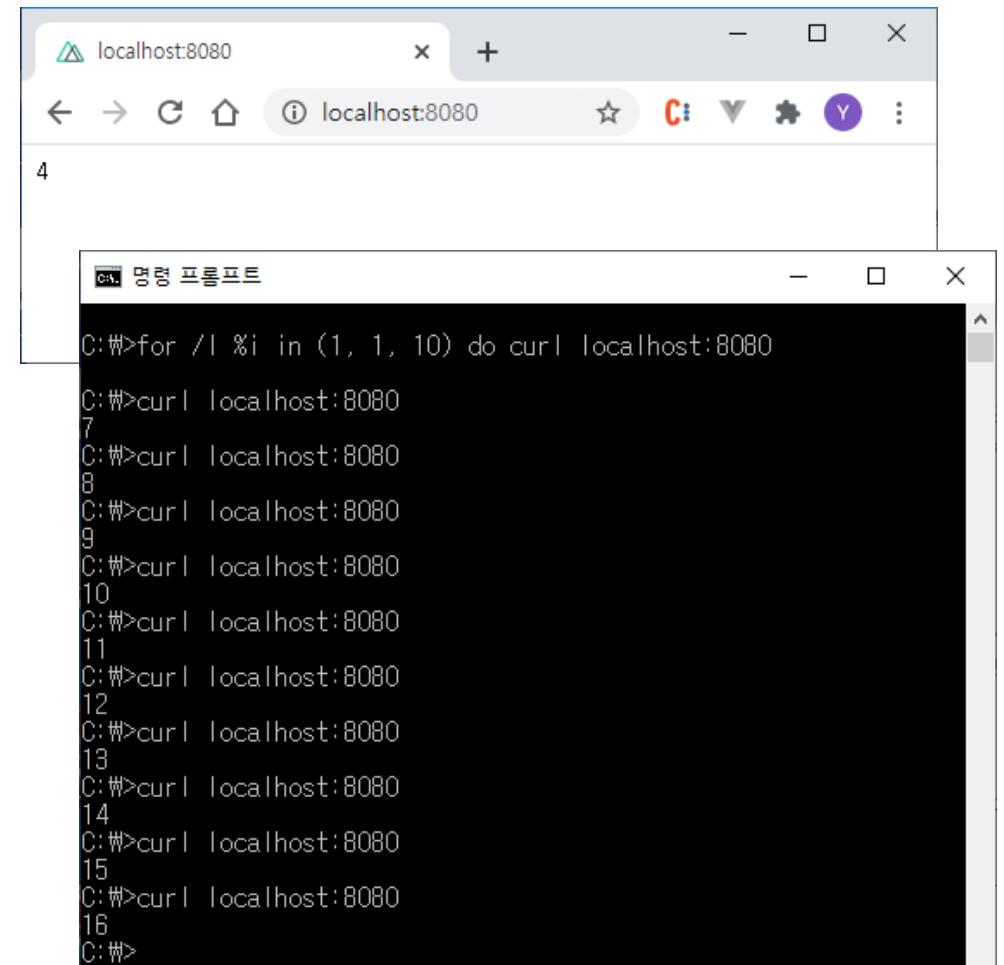
var fs = require('fs');

fs.readFile('./count.txt', 'utf8', function(err, data){
  if ( !err ) count = parseInt(data);
});

var http = require('http');

http.createServer(function(request, response) {
  count = count + 1;
  fs.writeFileSync('./count.txt', count);

  response.writeHead(200);
  response.end(count.toString());
}).listen(8080);
```



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

3) Docker image 빌드

dockerfile2

```
FROM node:12.18.0  
EXPOSE 8080  
COPY server2.js .  
CMD node server2.js
```

docker build -t count:v2 -f dockerfile2 . 명령으로 이미지를 빌드한다.

docker images 명령으로 이미지를 확인한다.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
count	v2	7eefbd1db665	26 seconds ago	918MB
count	v1	5c6b7547775f	22 minutes ago	918MB
hostname	v1	09b402b313d7	41 hours ago	918MB
node	12.18.0	b31738bd2a38	3 weeks ago	918MB
k8s.gcr.io/kube-proxy	v1.18.3	3439b7546f29	6 weeks ago	117MB
k8s.gcr.io/kube-controller-manager	v1.18.3	da26705ccb4b	6 weeks ago	162MB
k8s.gcr.io/kube-apiserver	v1.18.3	7e28efa976bd	6 weeks ago	173MB
k8s.gcr.io/kube-scheduler	v1.18.3	76216c34ed0c	6 weeks ago	95.3MB
kubernetesui/dashboard	v2.0.0	8b32422733b3	2 months ago	222MB
k8s.gcr.io/pause	3.2	80d28bedfe5d	4 months ago	683kB
k8s.gcr.io/coredns	1.6.7	67da37a9a360	5 months ago	43.8MB
k8s.gcr.io/etcd	3.4.3-0	303ce5db0e90	8 months ago	288MB
kubernetesui/metrics-scrapers	v1.0.2	3b08661dc379	8 months ago	40.1MB
gcr.io/k8s-minikube/storage-provisioner	v1.8.1	4689081edb10	2 years ago	80.8MB

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

- 4) Deployment 컨트롤러를 생성한다.

server2_deployment.yaml

```
apiVersion: apps/v1  
  
kind: Deployment  
metadata:  
  name: count  
  labels:  
    app: count  
  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: count  
  template:  
    metadata:  
      labels:  
        app: count  
    spec:  
      containers:  
        - name: count  
          image: 'count:v2'
```

kubectl apply -f server2_deployment.yaml 명령으로 Deployment 컨트롤러를 생성한다.
(kubectl create deployment count --image=count:v2와 동일함)

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

- 5) 서비스를 생성한다.

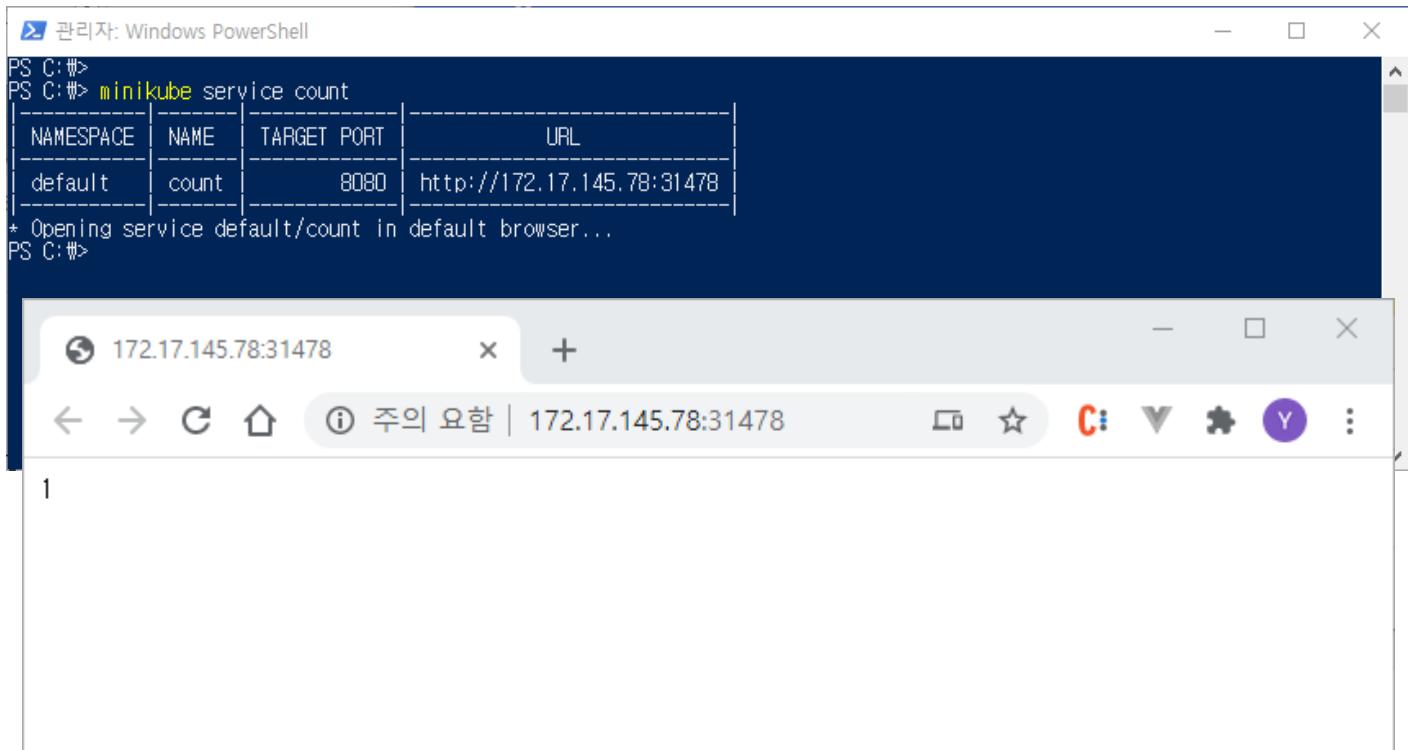
server2_service.yaml

```
apiVersion: v1

kind: Service
metadata:
  name: count
  labels:
    app: count

spec:
  ports:
    - port: 8080
  selector:
    app: count
  type: LoadBalancer
```

kubectl apply -f server2_service.yaml 명령으로 서비스를 생성한다.
(kubectl expose deployment count --port=8080 --type=LoadBalancer와 동일함)



```
PS C:\#> minikube service count
|-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----|
| default   | count | 8080        | http://172.17.145.78:31478 |
* Opening service default/count in default browser...
PS C:\#>
```

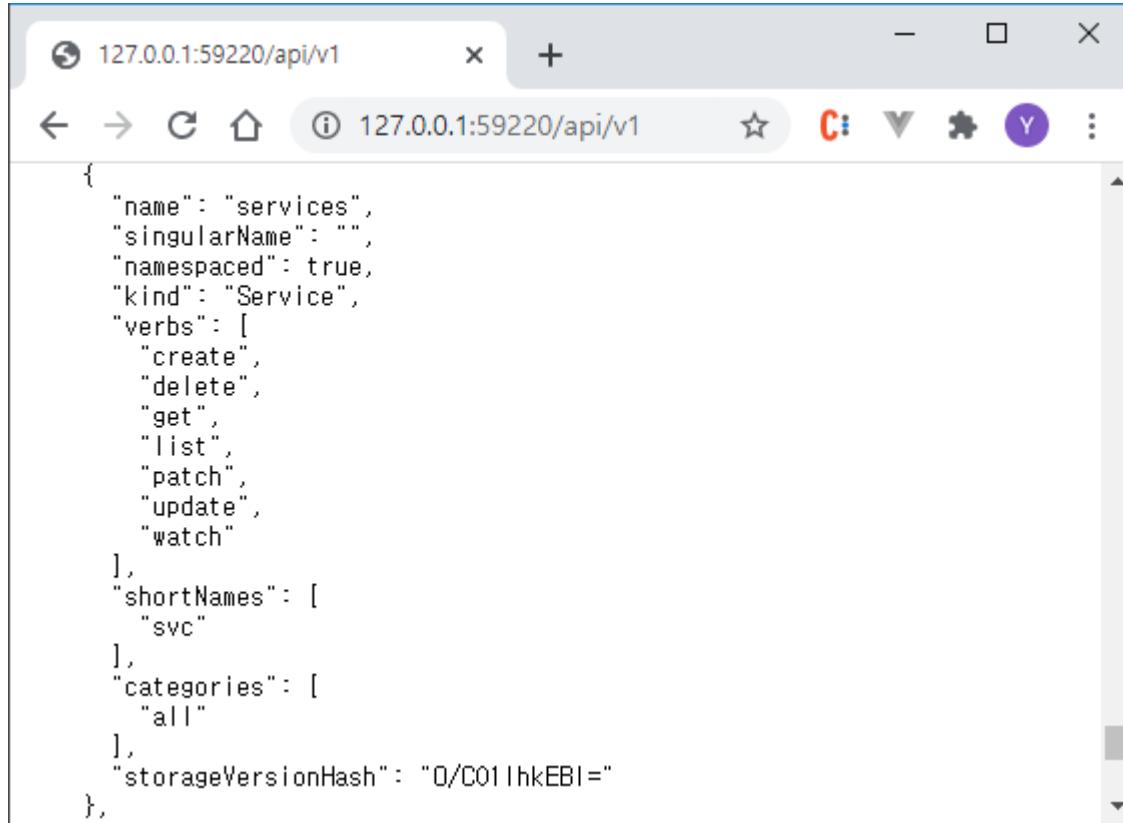
172.17.145.78:31478

1

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ Kubernetes API

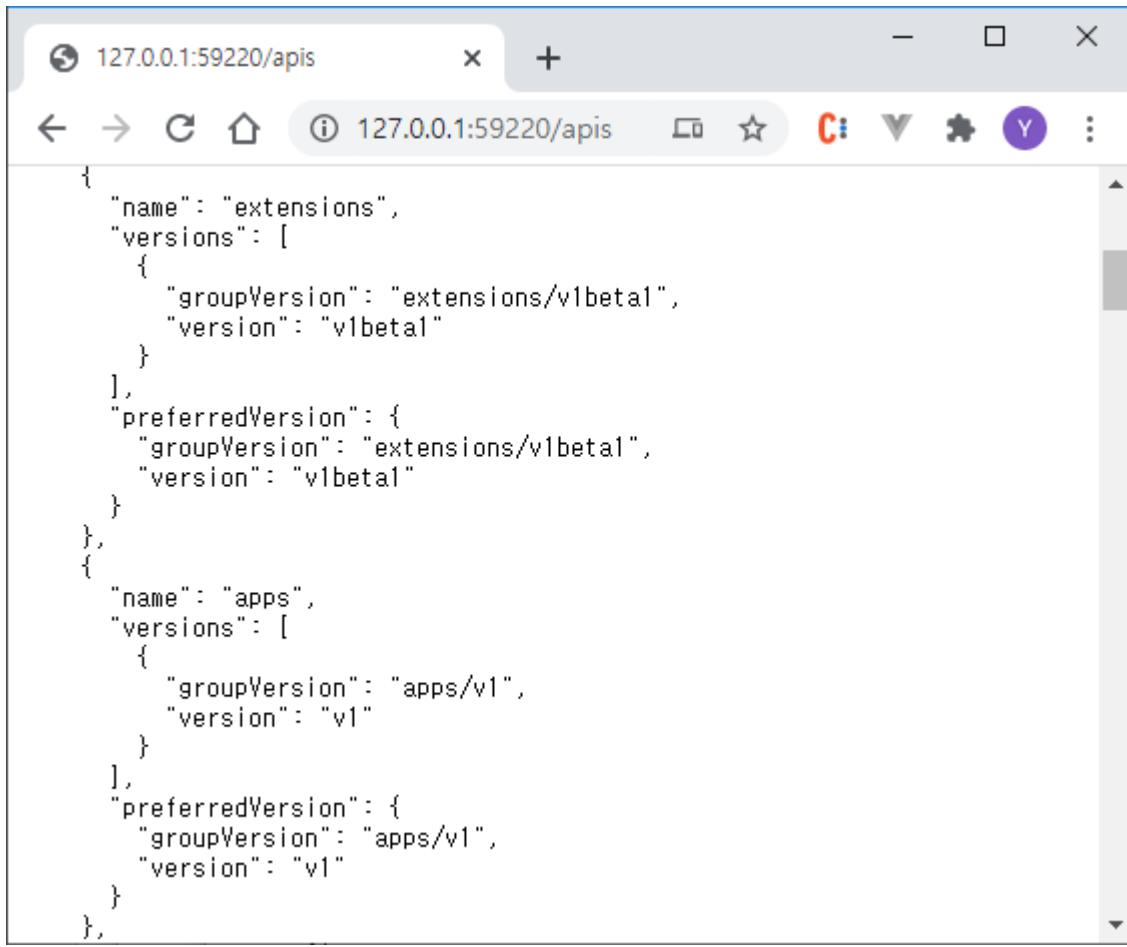


The screenshot shows a browser window with the URL `127.0.0.1:59220/api/v1` in the address bar. The page content is a JSON object representing the 'services' resource. The JSON structure includes fields such as 'name', 'singularName', 'namespaced', 'kind', 'verbs', 'shortNames', 'categories', and 'storageVersionHash'. The 'verbs' field lists actions like 'create', 'delete', 'get', 'list', 'patch', 'update', and 'watch'. The 'shortNames' field contains 'svc'. The 'categories' field contains 'all'. The 'storageVersionHash' field is set to '0/C01lhkEBI='.

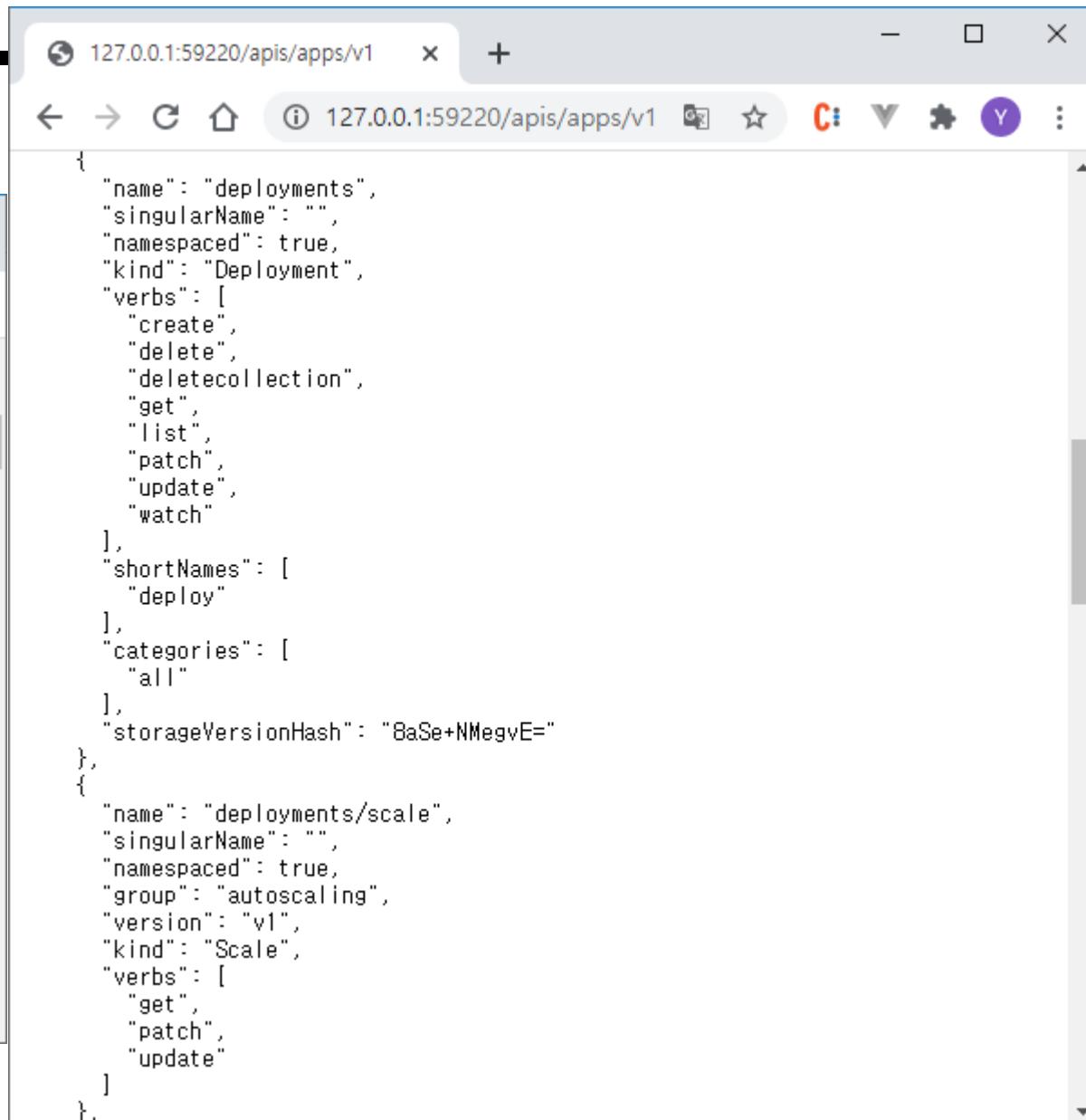
```
{
  "name": "services",
  "singularName": "",
  "namespaced": true,
  "kind": "Service",
  "verbs": [
    "create",
    "delete",
    "get",
    "list",
    "patch",
    "update",
    "watch"
  ],
  "shortNames": [
    "svc"
  ],
  "categories": [
    "all"
  ],
  "storageVersionHash": "0/C01lhkEBI="
}
```

7. Kubernetes

■ Kubernetes API



```
{  
  "name": "extensions",  
  "versions": [  
    {  
      "groupVersion": "extensions/v1beta1",  
      "version": "v1beta1"  
    }  
  ],  
  "preferredVersion": {  
    "groupVersion": "extensions/v1beta1",  
    "version": "v1beta1"  
  },  
  {  
    "name": "apps",  
    "versions": [  
      {  
        "groupVersion": "apps/v1",  
        "version": "v1"  
      }  
    ],  
    "preferredVersion": {  
      "groupVersion": "apps/v1",  
      "version": "v1"  
    }  
  },  
  ...  
}
```



```
{  
  "name": "deployments",  
  "singularName": "",  
  "namespaced": true,  
  "kind": "Deployment",  
  "verbs": [  
    "create",  
    "delete",  
    "deletecollection",  
    "get",  
    "list",  
    "patch",  
    "update",  
    "watch"  
  ],  
  "shortNames": [  
    "deploy"  
  ],  
  "categories": [  
    "all"  
  ],  
  "storageVersionHash": "8aSe+NMegvE=",  
  {  
    "name": "deployments/scale",  
    "singularName": "",  
    "namespaced": true,  
    "group": "autoscaling",  
    "version": "v1",  
    "kind": "Scale",  
    "verbs": [  
      "get",  
      "patch",  
      "update"  
    ]  
  }  
}
```

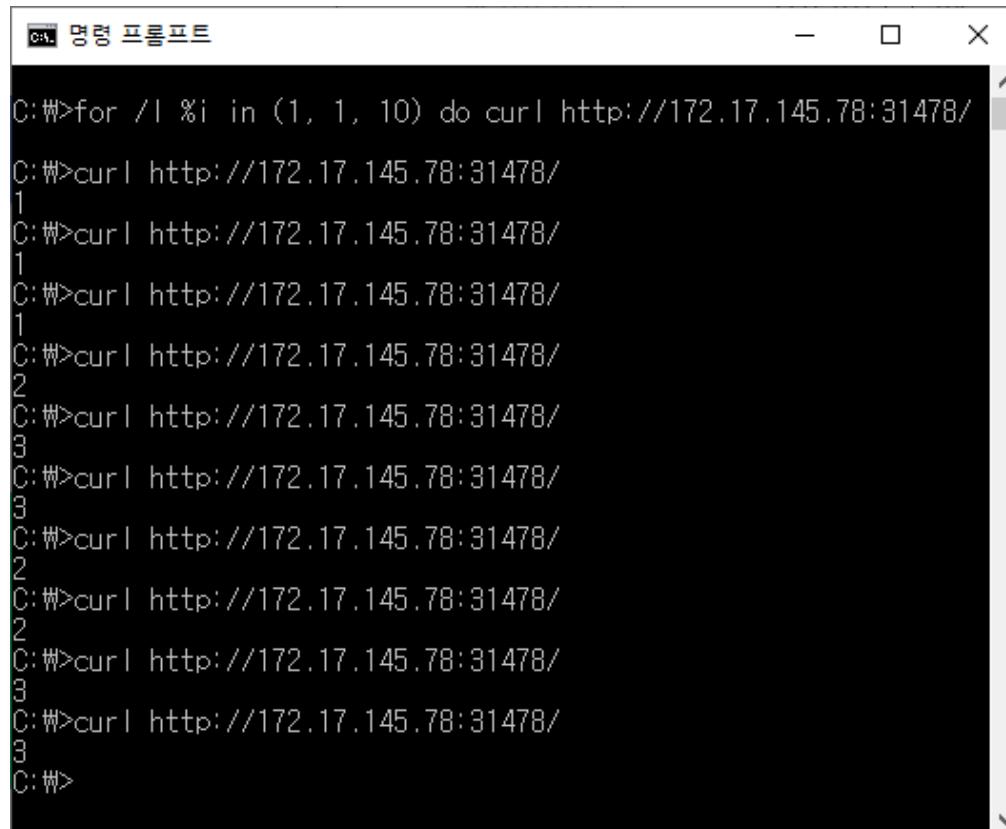
7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

5) 서비스를 생성한다.

A) POD의 개수를 늘리고 서비스에 연결해 본다.



A screenshot of a Windows Command Prompt window titled "명령 프롬프트". The window contains the following command and its execution:

```
C:\#>for /l %i in (1, 1, 10) do curl http://172.17.145.78:31478/  
C:\#>curl http://172.17.145.78:31478/  
1  
C:\#>curl http://172.17.145.78:31478/  
1  
C:\#>curl http://172.17.145.78:31478/  
1  
C:\#>curl http://172.17.145.78:31478/  
2  
C:\#>curl http://172.17.145.78:31478/  
3  
C:\#>curl http://172.17.145.78:31478/  
3  
C:\#>curl http://172.17.145.78:31478/  
2  
C:\#>curl http://172.17.145.78:31478/  
2  
C:\#>curl http://172.17.145.78:31478/  
3  
C:\#>curl http://172.17.145.78:31478/  
3  
C:\#>
```

7. Kubernetes

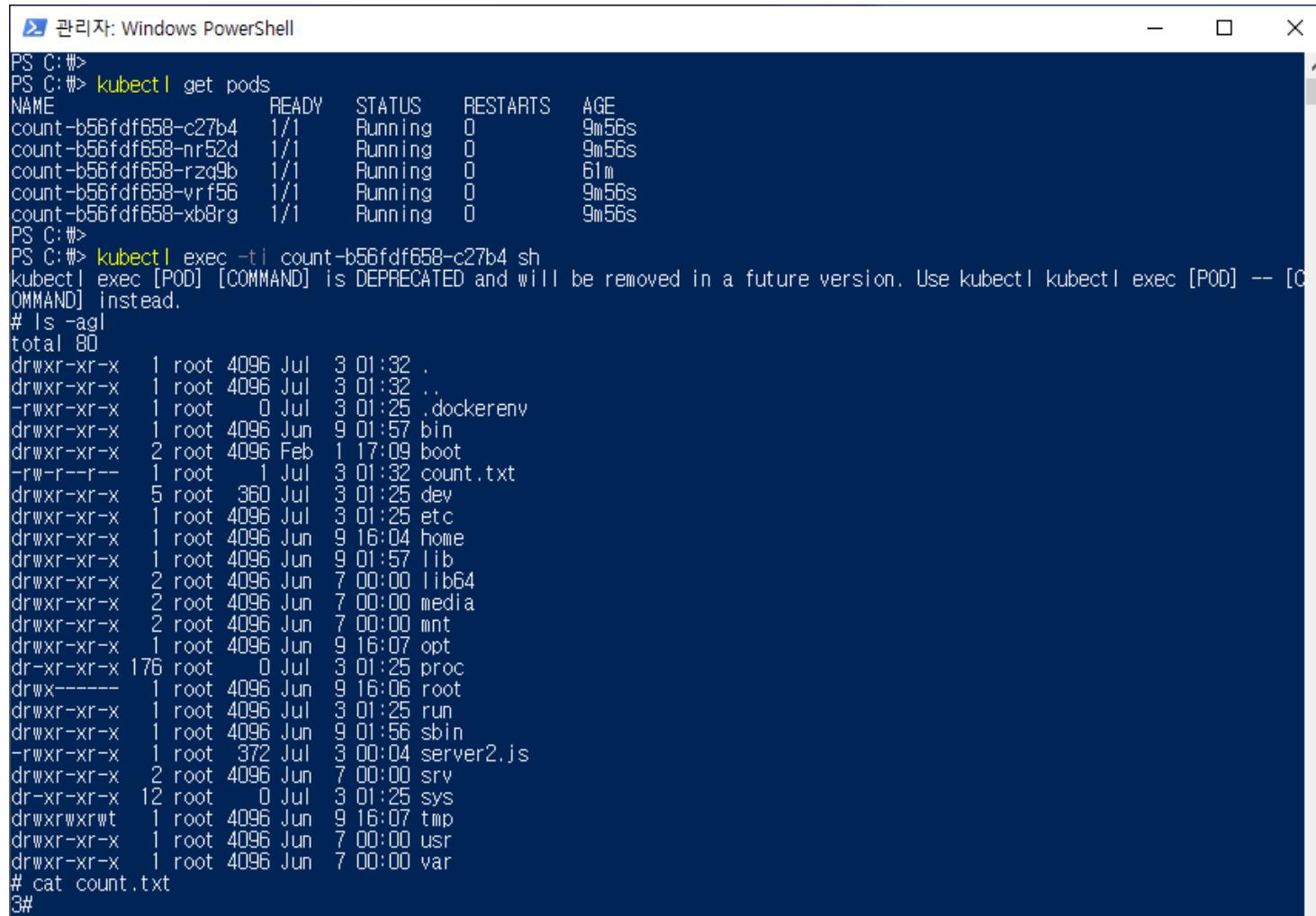
<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

5) 서비스를 생성한다.

B) POD에 shell로 연결해 본다.

kubectl exec -ti {POD이름} sh 명령으로
POD에 연결한다.



A screenshot of a Windows PowerShell window titled "관리자: Windows PowerShell". The window shows the output of several Kubernetes commands. It starts with "kubectl get pods" which lists five pods named 'count' with various status codes. Then it shows "kubectl exec -ti count-b56fdf658-c27b4 sh", followed by a warning about the command being deprecated. The user then runs "ls -al" to list the files in the pod's directory, which includes 'count.txt', 'server2.js', and other standard Linux system files like proc, dev, etc. Finally, the user runs "cat count.txt".

```
PS C:\#>
PS C:\#> kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
count-b56fdf658-c27b4  1/1     Running   0          9m56s
count-b56fdf658-nr52d  1/1     Running   0          9m56s
count-b56fdf658-rzq9b  1/1     Running   0          61m
count-b56fdf658-vrf56  1/1     Running   0          9m56s
count-b56fdf658-xb8rg  1/1     Running   0          9m56s
PS C:\#>
PS C:\#> kubectl exec -ti count-b56fdf658-c27b4 sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl kubectl exec [POD] -- [COMMAND] instead.
# ls -al
total 80
drwxr-xr-x  1 root 4096 Jul  3 01:32 .
drwxr-xr-x  1 root 4096 Jul  3 01:32 ..
-rwxr-xr-x  1 root    0 Jul  3 01:25 .dockerenv
drwxr-xr-x  1 root 4096 Jun  9 01:57 bin
drwxr-xr-x  2 root 4096 Feb  1 17:09 boot
-rw-r--r--  1 root    1 Jul  3 01:32 count.txt
drwxr-xr-x  5 root 360 Jul  3 01:25 dev
drwxr-xr-x  1 root 4096 Jul  3 01:25 etc
drwxr-xr-x  1 root 4096 Jun  9 16:04 home
drwxr-xr-x  1 root 4096 Jun  9 01:57 lib
drwxr-xr-x  2 root 4096 Jun  7 00:00 lib64
drwxr-xr-x  2 root 4096 Jun  7 00:00 media
drwxr-xr-x  2 root 4096 Jun  7 00:00 mnt
drwxr-xr-x  1 root 4096 Jun  9 16:07 opt
dr-xr-xr-x  176 root 0 Jul  3 01:25 proc
drwxr----- 1 root 4096 Jun  9 16:06 root
drwxr-xr-x  1 root 4096 Jul  3 01:25 run
drwxr-xr-x  1 root 4096 Jun  9 01:56 sbin
-rwxr-xr-x  1 root 372 Jul  3 00:04 server2.js
drwxr-xr-x  2 root 4096 Jun  7 00:00 srv
dr-xr-xr-x  12 root 0 Jul  3 01:25 sys
drwxrwxrwt  1 root 4096 Jun  9 16:07 tmp
drwxr-xr-x  1 root 4096 Jun  7 00:00 usr
drwxr-xr-x  1 root 4096 Jun  7 00:00 var
# cat count.txt
```

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

5) 서비스를 생성한다.

B) POD에 shell로 연결해 본다.

kubectl exec -ti {POD이름} sh 명령으로
POD에 연결한다.

또는 dashboard에서 "Exec" 명령으로
POD의 shell에 연결할 수 있다.

파드								
이름	네임스페이스	레이블	노드	상태	재시작	CPU 사용량(cores)	메모리 사용량(bytes)	생성 시간
count-b56fdf658-c27b4	default	app: count pod-template-hash: b56fdf658	minikube	Running	0	-	-	17 minutes ago
count-b56fdf658-nr52d	default	app: count pod-template-hash: b56fdf658	minikube	Running	0	-	-	17 minutes ago
count-b56fdf658-rzq9b	default	app: count pod-template-hash: b56fdf658	minikube	Running	0	-	-	17 minutes ago

Kubernetes Dashboard

kubernetes

Workloads > Pods > count-b56fdf658-c27b4 > Shell

클러스터

클러스터 룰

네임스페이스

노드

퍼시스턴트 볼륨

스토리지 클래스

네임스페이스

default

개요

워크로드

크론 잡

데몬 세트

디플로이먼트

잡

파드

레플리카 세트

count-b56fdf658-c27b4의 count

```
root@count-b56fdf658-c27b4:/# ls -al
total 80
drwxr-xr-x  1 root 4096 Jul  3 01:32 .
drwxr-xr-x  1 root 4096 Jul  3 01:32 ..
-rw-r--r--  1 root    0 Jul  3 01:25 .dockerenv
drwxr-xr-x  1 root 4096 Jun  9 01:57 bin
drwxr-xr-x  2 root 4096 Feb  1 17:09 boot
-rw-r--r--  1 root   1 Jul  3 01:32 count.txt
drwxr-xr-x  5 root 360 Jul  3 01:25 dev
drwxr-xr-x  1 root 4096 Jul  3 01:25 etc
drwxr-xr-x  1 root 4096 Jun  9 16:04 home
drwxr-xr-x  1 root 4096 Jun  9 01:57 lib
drwxr-xr-x  2 root 4096 Jun  7 00:00 lib64
drwxr-xr-x  2 root 4096 Jun  7 00:00 media
drwxr-xr-x  2 root 4096 Jun  7 00:00 mnt
drwxr-xr-x  1 root 4096 Jun  9 16:07 opt
dr-xr-xr-x 176 root   0 Jul  3 01:25 proc
drwxr-xr-x  1 root 4096 Jul  3 01:40 root
drwxr-xr-x  1 root 4096 Jul  3 01:25 run
drwxr-xr-x  1 root 4096 Jun  9 01:56 sbin
-rw-r--r--  1 root 372 Jul  3 00:04 server2.js
drwxr-xr-x  2 root 4096 Jun  7 00:00 srv
dr-xr-xr-x 12 root   0 Jul  3 01:25 sys
drwxrwxrwt  1 root 4096 Jun  9 16:07 tmp
drwxr-xr-x  1 root 4096 Jun  7 00:00 usr
drwxr-xr-x  1 root 4096 Jun  7 00:00 var
root@count-b56fdf658-c27b4:/# cat count.txt
3root@count-b56fdf658-c27b4:/# 
```

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

6) Persistent Volume을 생성한다.

server2_deployment.yaml

```
---  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: count-pv-claim  
  labels:  
    app: count  
  
spec:  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 1Ki
```

```
kind: Deployment  
spec:  
  template:  
    spec:  
      containers:  
        - name: count  
          image: 'count:v3'  
          volumeMounts:  
            - name: count-data  
              mountPath: /data  
          volumes:  
            - name: count-data  
              persistentVolumeClaim:  
                claimName: count-pv-claim
```

```
var count = 0;  
  
var fs = require('fs');  
var http = require('http');  
http.createServer(function(request, response) {  
  fs.readFile('/data/count.txt', 'utf8', function(err, data){  
    if ( !err ) count = parseInt(data);  
  
    count = count + 1;  
    fs.writeFileSync('/data/count.txt', count);  
  
    response.writeHead(200);  
    response.end(count.toString());  
  });  
}).listen(8080);
```

docker build -t count:v3 -f dockerfile2 . 명령으로 이미지를 빌드한다.

kubectl apply -f server2_deployment.yaml 명령으로 Deployment 컨트롤러를 수정한다.

server2.js

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

6) Persistent Volume을 생성한다.

A) Shell로 연결해서 Persistent Volume을 확인한다.

The image shows two terminal sessions on a host node named 'count-54b88c7844-b5d7h'. The left terminal displays the output of the 'mount' command, which lists various file systems mounted on the host. The right terminal shows the output of 'cd data' followed by 'ls -al', which lists the contents of a directory named 'data' on the Persistent Volume. A red box highlights the line '/dev/sda1 on /data type ext4 (rw,relatime)' in the 'mount' output, indicating the PV's location.

```
root@count-54b88c7844-b5d7h:~# mount
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/1/SJ32AYGX32K34HQYJCMXPPLWQC:/var/lib/docker/overlay2/1/ZQ3P030ZU5VJ0B50LOPALKQ/MqGZKJYSK35UTMUYI4TJZWN:/var/lib/docker/overlay2/1/Y6AZV2SV7HGFK752650AW54C:/var/lib/docker/overlay2/1/TSYFJdocke/overlay2/1/ZK54MGMECWTON4ESV4DLTKGF03:/var/lib/docker/overlay2/1/Ah44CL3IFDF4RGP7XGPQGLJUT:/var/lib/docker/HEWLLV:/var/lib/docker/overlay2//LG5ETMVLYHBWQAKNIOUGRLK7,upperdir=~/var/lib/docker/overlay2/aef2fa411adbc0aa4ecce0f708049c94f,proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,size=65536k,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
sysfs on /sys type sysfs (ro,nosuid,nodev,noexec,relatime)
tmpfs on /sys/fs/group type tmpfs (ro,nosuid,nodev,noexec,relatime,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (ro,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/systemd/timedate)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/devices type cgroup (ro,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
mqueue on /dev/mqueue type mqueue (ro,nosuid,nodev,noexec,relatime)
/dev/sda1 on /data type ext4 (rw,relatime)
/dev/sda1 on /dev/centosvg1 type ext4 (rw,relatime)
/dev/sda1 on /etc/resolv.conf type ext4 (rw,relatime)
/dev/sda1 on /etc/hostname type ext4 (rw,relatime)
/dev/sda1 on /etc/hosts type ext4 (rw,relatime)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
tmpfs on /run/secrets/kubernetes.io/serviceaccount type tmpfs (ro,relatime)
proc on /proc/sound type proc (ro,relatime)
proc on /proc/bus type proc (ro,relatime)
proc on /proc/fs type proc (ro,relatime)
proc on /proc/irq type proc (ro,relatime)
proc on /proc/sys type proc (ro,relatime)
proc on /proc/sysrq-trigger type proc (ro,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
tmpfs on /sys/firmware type tmpfs (ro,relatime)
root@count-54b88c7844-b5d7h:~#
```

```
count-54b88c7844-b5d7h의 count
```

```
root@count-54b88c7844-b5d7h:~# cd data
root@count-54b88c7844-b5d7h:/data# ls -al
total 12
drwxrwxrwx 2 root 4096 Jul 3 02:00 .
drwxr-xr-x 1 root 4096 Jul 3 02:00 ..
-rw-r--r-- 1 root 2 Jul 3 02:02 count.txt
root@count-54b88c7844-b5d7h:/data# more count.txt
13
root@count-54b88c7844-b5d7h:/data#
```

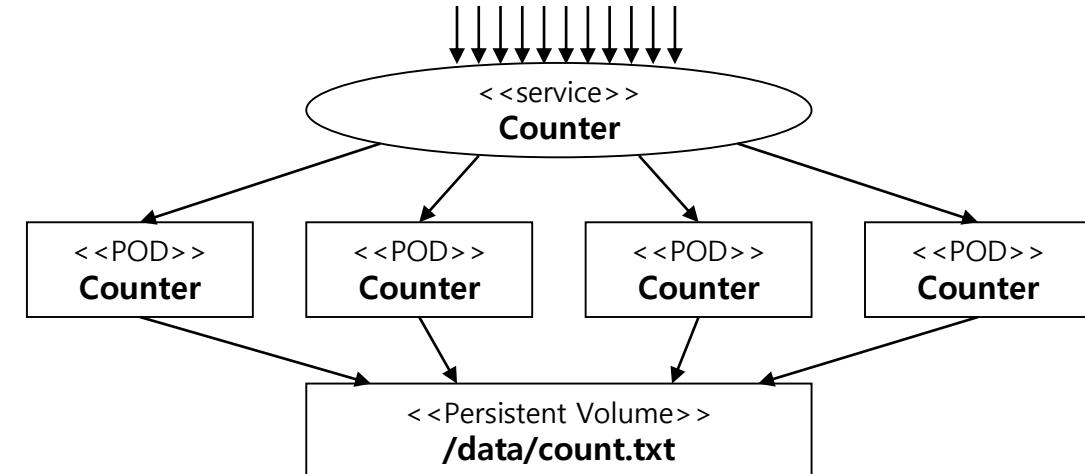
7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

- 6) Persistent Volume을 생성한다.
- B) POD의 개수를 늘리고 서비스에 연결해 본다.

```
C:\#>for /l %i in (1, 1, 10) do curl http://172.17.145.78:31478/  
C:\#>curl http://172.17.145.78:31478/  
22  
C:\#>curl http://172.17.145.78:31478/  
23  
C:\#>curl http://172.17.145.78:31478/  
24  
C:\#>curl http://172.17.145.78:31478/  
25  
C:\#>curl http://172.17.145.78:31478/  
26  
C:\#>curl http://172.17.145.78:31478/  
27  
C:\#>curl http://172.17.145.78:31478/  
28  
C:\#>curl http://172.17.145.78:31478/  
29  
C:\#>curl http://172.17.145.78:31478/  
30  
C:\#>curl http://172.17.145.78:31478/  
31  
C:\#>
```



read count
increase count
write count
가 ATOMIC 해야 한다.

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

- 1) Redis DB 서비스를 생성한다.

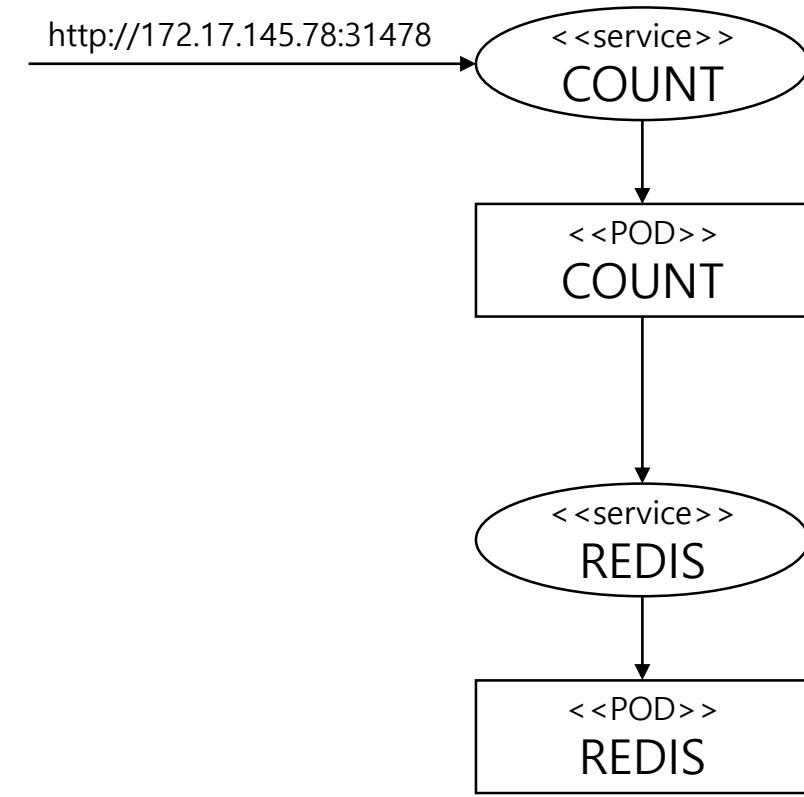
redis_deployment.yaml

```
apiVersion: apps/v1  
  
kind: Deployment  
metadata:  
  name: redis  
  labels:  
    app: redis  
  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: redis  
  template:  
    metadata:  
      labels:  
        app: redis  
    spec:  
      containers:  
        - name: redis  
          image: redis
```

redis_service.yaml

```
apiVersion: v1  
  
kind: Service  
metadata:  
  name: redis  
  labels:  
    app: redis  
  
spec:  
  ports:  
    - port: 6379  
      targetPort: 6379  
  selector:  
    app: redis
```

http://172.17.145.78:31478



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

2) Redis DB를 사용한다.

server3.js

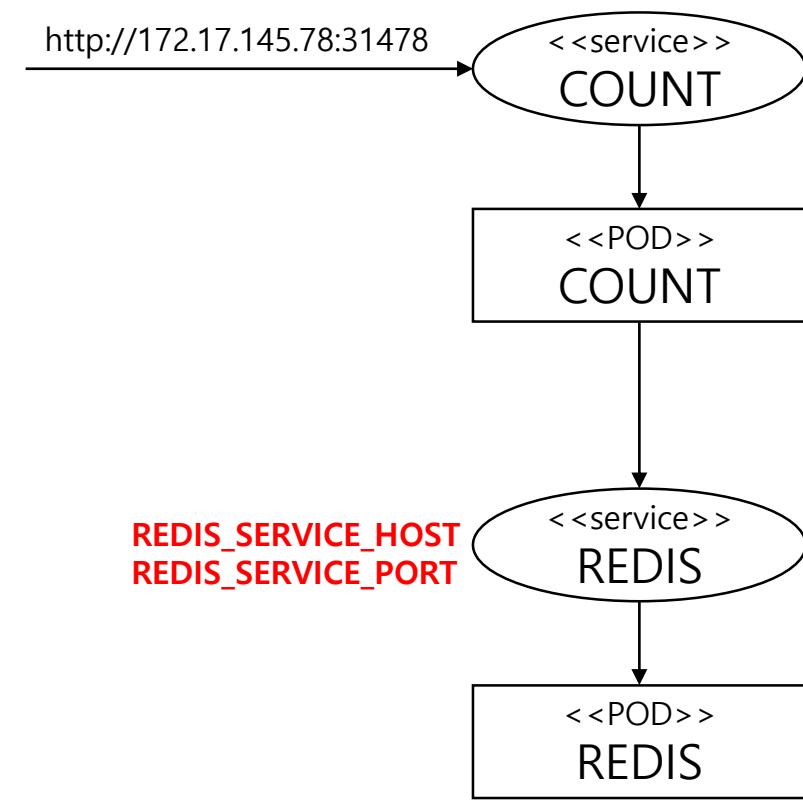
```
var redis = require('redis').createClient(process.env.REDIS_SERVICE_PORT,
                                         process.env.REDIS_SERVICE_HOST);

var http = require('http');

http.createServer(function(request, response) {
  redis.get('count', function(err, data){
    var count = 0;
    if ( !err && data != null) count = parseInt(data);

    count = count + 1;
    redis.set('count', count);

    response.writeHead(200);
    response.end(count.toString());
  });
}).listen(8080);
```



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

2) Redis DB를 사용한다.

package.json

```
{  
  "name": "count",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "start": "node server3.js"  
  },  
  "dependencies": {  
    "redis": "latest"  
  }  
}
```

dockerfile3

```
FROM node:12.18.0  
  
WORKDIR /src  
COPY package.json .  
RUN npm install  
  
COPY server3.js .  
  
EXPOSE 8080  
CMD npm start
```

docker build -t count:v4 -f dockerfile3 . 명령으로 이미지를 빌드한다.
kubectl set image deployments/count count=count:v4 으로
이미지를 update한다.

파드	이름	네임스페이스	레이블	노드	상태	재시작	CPU 사용량(cores)	메모리 사용량(bytes)	생성 시간
	count-54b88c7844-7979h	default	app: count pod-template-hash: 54b88c7844	minikube	Terminated: Error	0	-	-	an hour ago
	count-54b88c7844-btwmt	default	app: count pod-template-hash: 54b88c7844	minikube	Terminated: Error	0	-	-	an hour ago
	count-585c774c7-4p9mg	default	app: count pod-template-hash: 585c774c7	minikube	Running	0	-	-	39 seconds ago
	count-585c774c7-hcwld	default	app: count pod-template-hash: 585c774c7	minikube	Running	0	-	-	39 seconds ago
	count-585c774c7-jmb7w	default	app: count pod-template-hash: 585c774c7	minikube	Running	0	-	-	37 seconds ago
	count-585c774c7-x4vp	default	app: count pod-template-hash: 585c774c7	minikube	Running	0	-	-	37 seconds ago
	count-585c774c7-z7mij	default	app: count pod-template-hash: 585c774c7	minikube	Running	0	-	-	39 seconds ago
	redis-85d47694f4-d4xgp	default	app: redis pod-template-hash: 85d47694f4	minikube	Running	0	-	-	36 minutes ago

새로운 POD 들이 생성되고, 기존 POD들이 종료된다.
계속 서비스는 제공된다.

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

2) Redis DB를 사용한다.

Kubernetes Dashboard

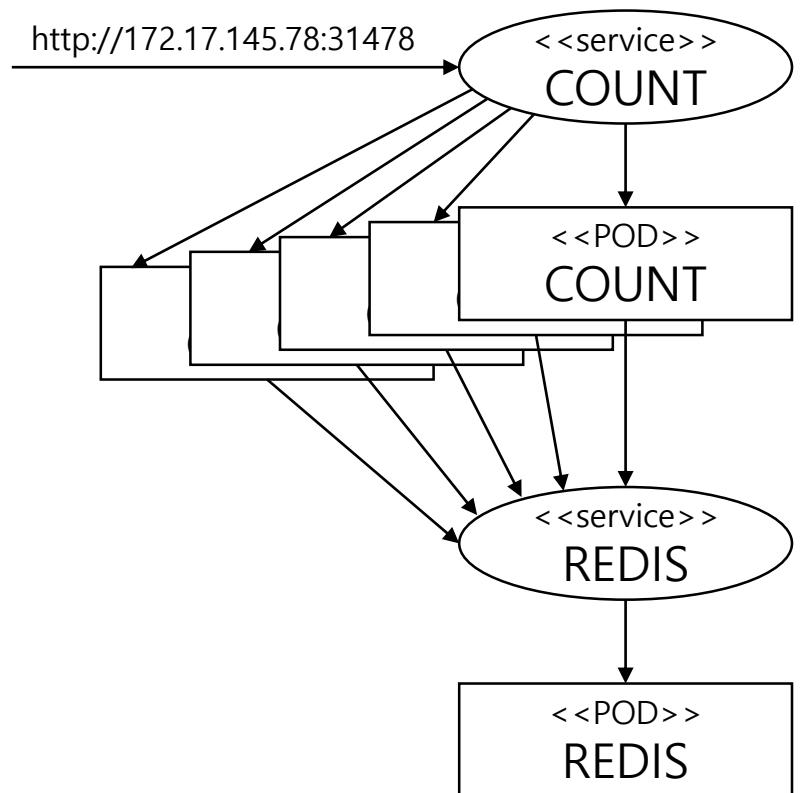
Overview

디플로이먼트

이름	네임스페이스	레이블	파드	생성 시간	이미지
count	default	app: count	5 / 5	an hour ago	count:v4
redis	default	app: redis	1 / 1	40 minutes ago	redis

파드

이름	네임스페이스	레이블	노드	상태	재시작	CPU 사용량(cores)	메모리 사용량(bytes)	생성 시간
count-585c774c7-4p9mg	default	app: count pod-template-hash: 58 5c774c7	minikube	Running	0	-	-	4 minutes ago
count-585c774c7-hcwkl	default	app: count pod-template-hash: 58 5c774c7	minikube	Running	0	-	-	4 minutes ago
count-585c774c7-jmb7w	default	app: count pod-template-hash: 58 5c774c7	minikube	Running	0	-	-	4 minutes ago
count-585c774c7-x4vip	default	app: count pod-template-hash: 58 5c774c7	minikube	Running	0	-	-	4 minutes ago
count-585c774c7-z7hvj	default	app: count pod-template-hash: 58 5c774c7	minikube	Running	0	-	-	4 minutes ago
redis-85d47694f4-d4xgp	default	app: redis pod-template-hash: 85 d47694f4	minikube	Running	0	-	-	40 minutes ago



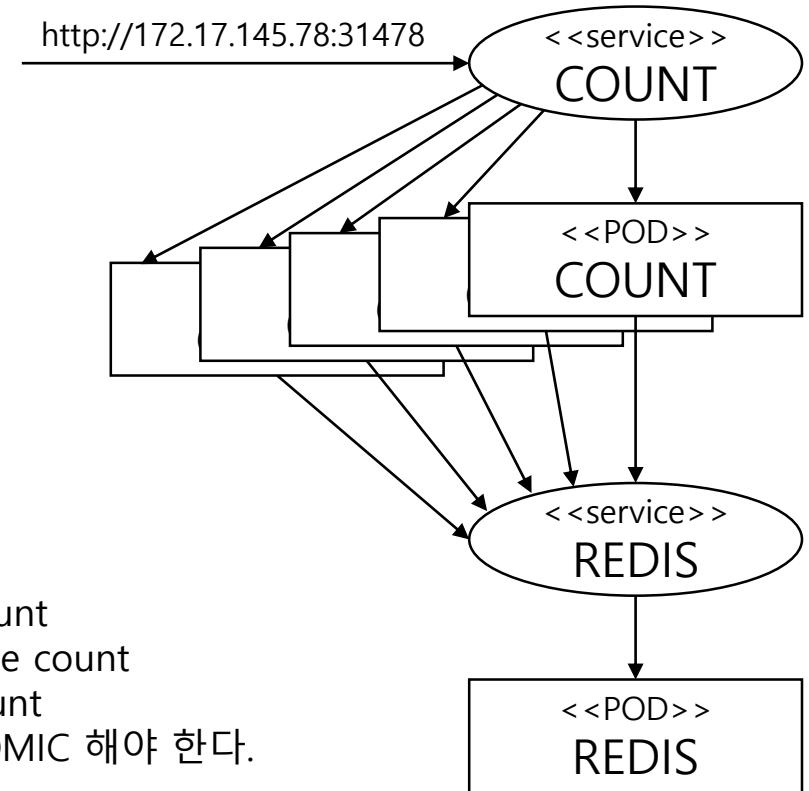
7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

2) Redis DB를 사용한다.

```
C:\₩>for /l %i in (1, 1, 10) do curl http://172.17.145.78:31478/  
C:\₩>curl http://172.17.145.78:31478/  
1  
C:\₩>curl http://172.17.145.78:31478/  
2  
C:\₩>curl http://172.17.145.78:31478/  
3  
C:\₩>curl http://172.17.145.78:31478/  
4  
C:\₩>curl http://172.17.145.78:31478/  
5  
C:\₩>curl http://172.17.145.78:31478/  
6  
C:\₩>curl http://172.17.145.78:31478/  
7  
C:\₩>curl http://172.17.145.78:31478/  
8  
C:\₩>curl http://172.17.145.78:31478/  
9  
C:\₩>curl http://172.17.145.78:31478/  
10  
C:\₩>
```



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

2) Redis DB를 사용한다.

server4.js

```
var redis = require('redis').createClient(process.env.REDIS_SERVICE_PORT,
                                         process.env.REDIS_SERVICE_HOST);

var http = require('http');

http.createServer(function(request, response) {
  redis.incr('count', function(err, data){
    var count = 0;
    if ( !err && data != null) count = parseInt(data);

    response.writeHead(200);
    response.end(count.toString());
  });
}).listen(8080);
```

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

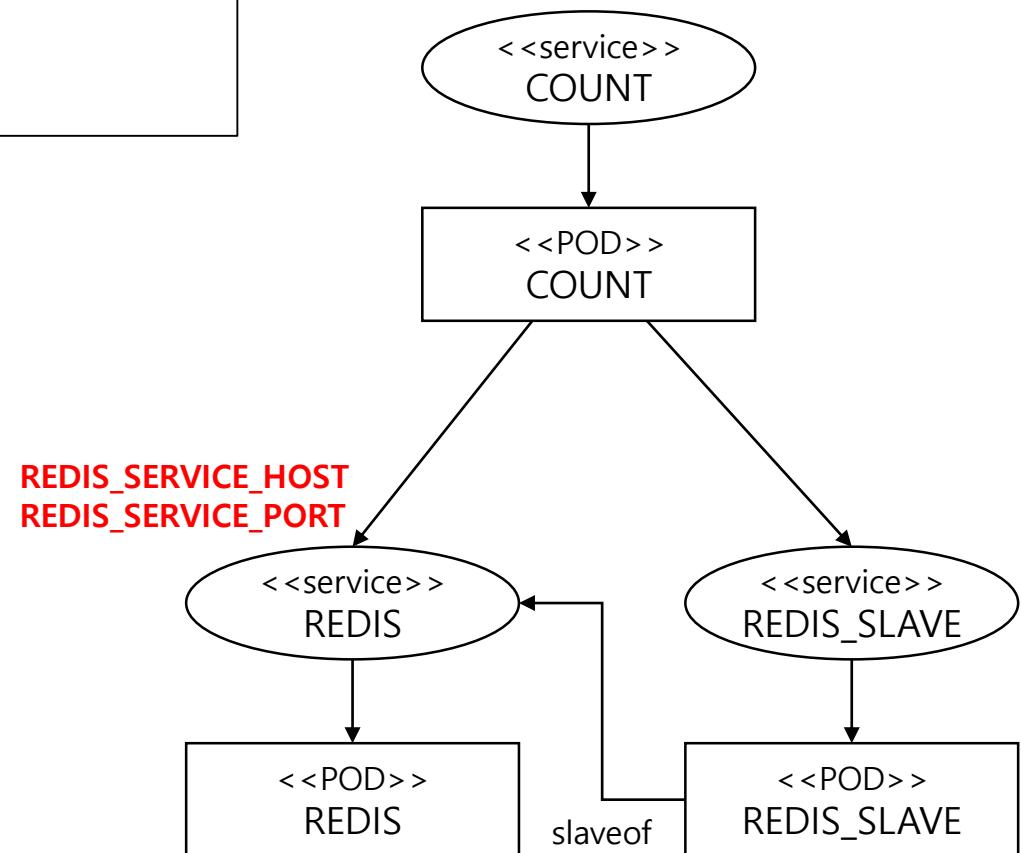
■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

3) Redis master-slave DB를 구성해 본다.

dockerfile_redis_slave

```
FROM redis:latest  
CMD redis-server --slaveof ${REDIS_SERVICE_HOST} ${REDIS_SERVICE_PORT}
```

docker build -t redis-slave:v1 -f dockerfile_redis_slave . 명령으로
이미지를 빌드한다.



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

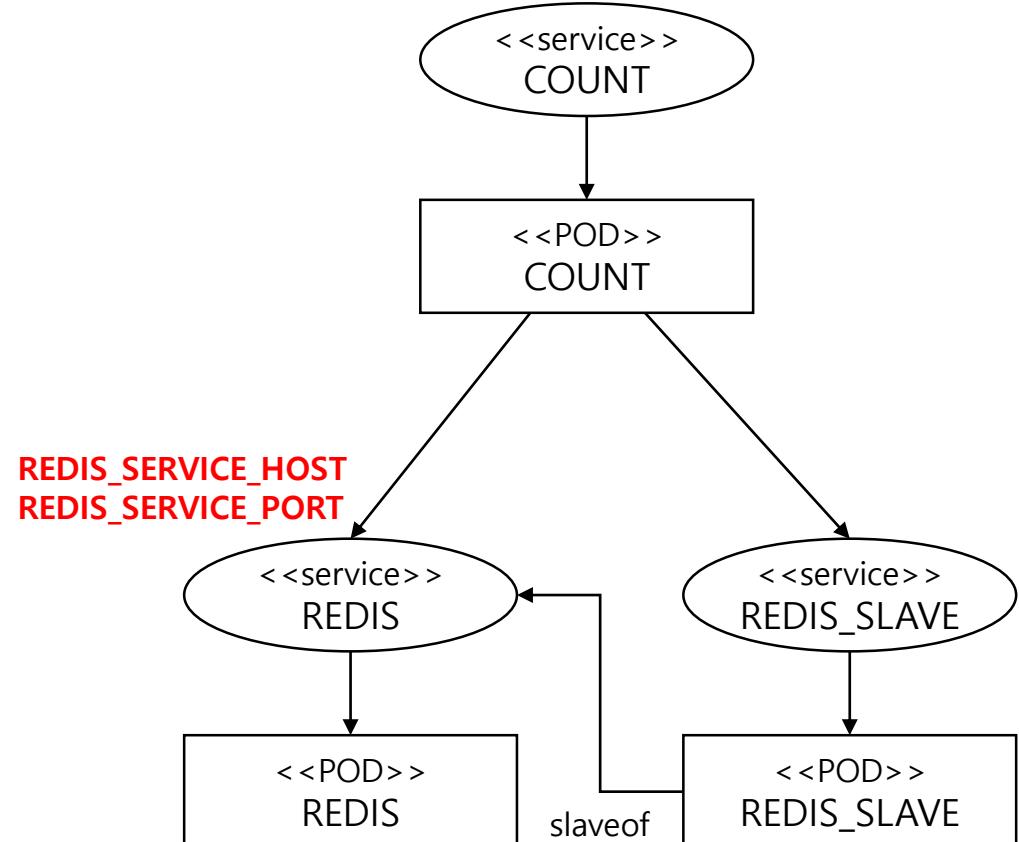
3) Redis master-slave DB를 구성해 본다.

redis_slave_deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-slave
  labels:
    app: redis-slave
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis-slave
  template:
    metadata:
      labels:
        app: redis-slave
    spec:
      containers:
        - name: redis-slave
          image: redis-slave:v1
```

redis_slave_service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: redis-slave
  labels:
    app: redis-slave
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    app: redis-slave
```



7. Kubernetes

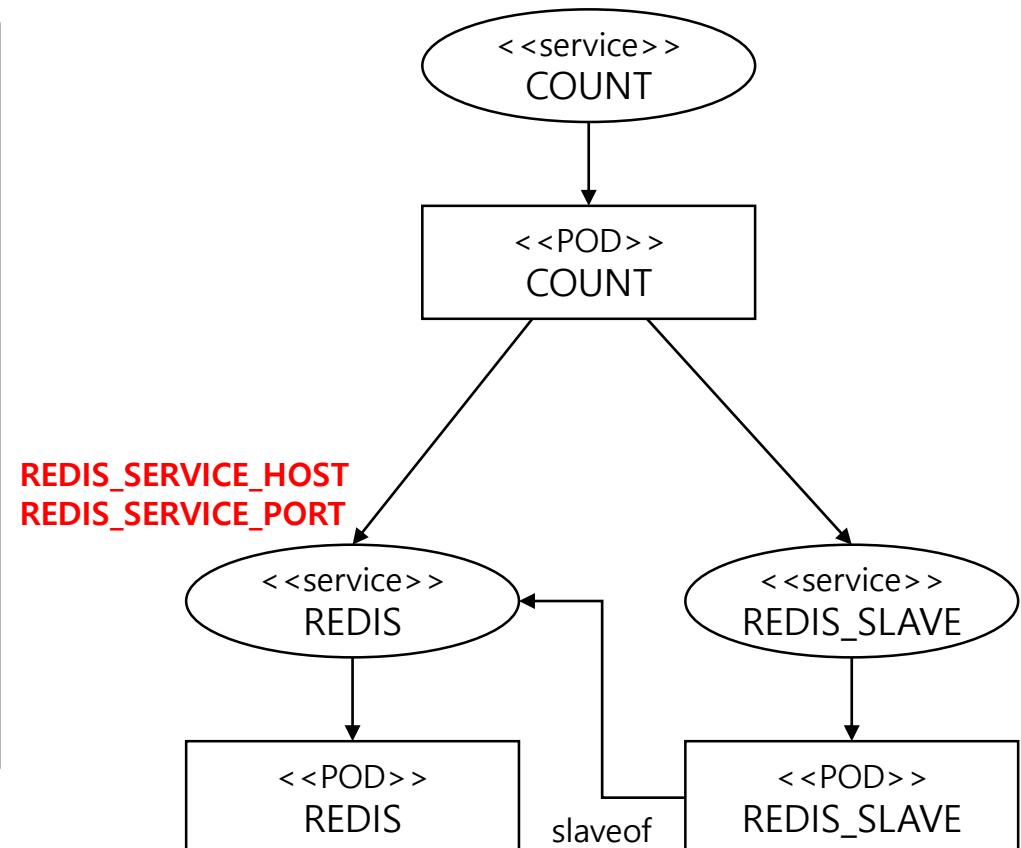
<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

- 3) Redis master-slave DB를 구성해 본다.

redis-slave POD에서 redis-cli로 master DB와 SYNC되었는지를 확인한다.

```
redis-slave-8f6bff989-v87vt의 redis-slave    ▾ 에 셀 인
root@redis-slave-8f6bff989-v87vt:/data# redis-cli
127.0.0.1:6379> get count
"30"
127.0.0.1:6379>
```



7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

3) Redis master-slave DB를 구성해 본다.

server5.js

```
var master = require('redis').createClient(process.env.REDIS_SERVICE_PORT,
                                         process.env.REDIS_SERVICE_HOST);
var slave = require('redis').createClient(process.env.REDIS_SLAVE_SERVICE_PORT,
                                         process.env.REDIS_SLAVE_SERVICE_HOST);

var http = require('http');
http.createServer(function(request, response) {
  slave.get('count', function(err, data){
    var count = 0;
    if ( !err && data != null) count = parseInt(data);

    count = count + 1;
    master.set('count', count);

    response.writeHead(200);
    response.end(count.toString());
  });
}).listen(8080);
```

docker build -t count:v5 -f dockerfile5 . 명령으로 이미지를 빌드한다.
kubectl set image deployments/count count=count:v5 으로
이미지를 update한다.

7. Kubernetes

<https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/>

■ 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

3) Redis master-slave DB를 구성해 본다.

디플로이먼트						
이름	네임스페이스	레이블	파드	생성 시간	이미지	⋮
✓ count	default	app: count	5 / 5	3 hours ago	count:v5	⋮
✓ redis	default	app: redis	1 / 1	2 hours ago	redis	⋮
✓ redis-slave	default	app: redis-slave	5 / 5	36 minutes ago	redis-slave:v1	⋮

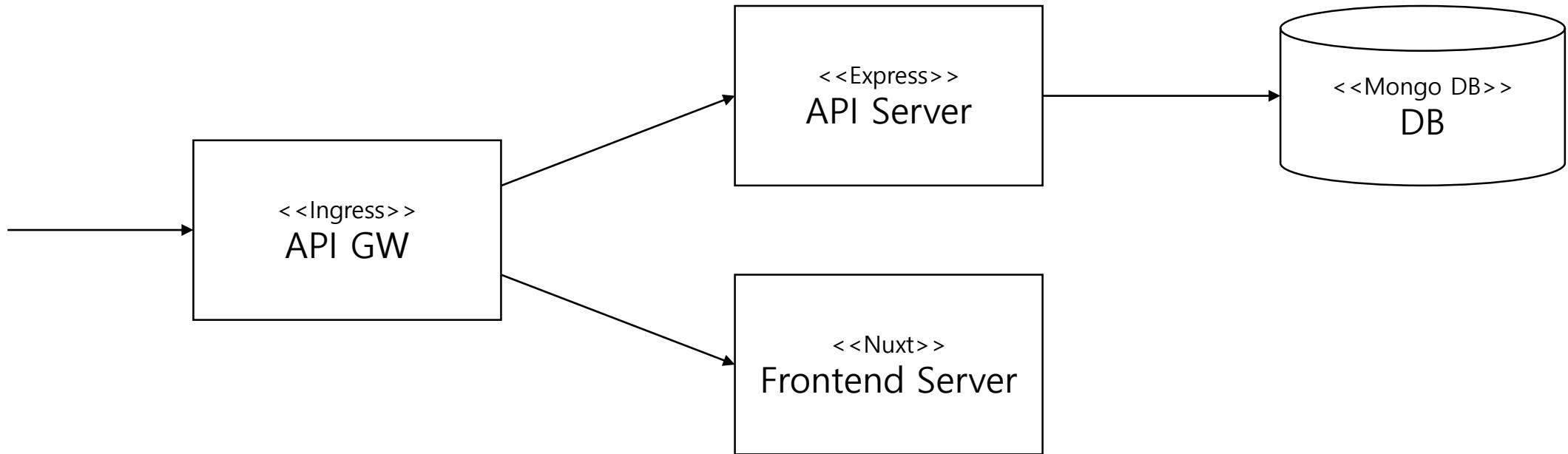
redis-slave POD도 5개로 증설한다.

curl 또는 브라우저로 동작을 확인한다.

8. Map UI

<https://github.com/bosornd/map-ui>

■ 서비스 구조



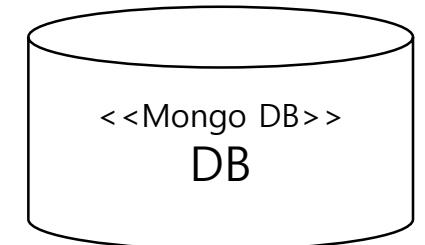
8. Map UI

<https://github.com/bosornd/map-ui>

1) Mongo DB (db/db_deployment.yaml)

```
apiVersion: apps/v1  
  
kind: Deployment  
metadata:  
  name: mongo-db  
  labels:  
    db: mongo-db  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      db: mongo-db  
  template:  
    metadata:  
      labels:  
        db: mongo-db  
    spec:  
      containers:  
        - name: mongo-db  
          image: mongo  
          volumeMounts:  
            - name: mongo-db-storage  
              mountPath: /data/db  
            - name: host  
              mountPath: /host
```

```
volumes:  
  - name: mongo-db-storage  
    persistentVolumeClaim:  
      claimName: db-pvc  
  - name: host  
    hostPath:  
      path: /dataset  
  
---  
apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
  name: db-pvc  
  labels:  
    db: mongo-db  
spec:  
  accessModes:  
    - ReadWriteOnce  
  resources:  
    requests:  
      storage: 10Gi
```



8. Map UI

<https://github.com/bosornd/map-ui>

1) Mongo DB (db/db_deployment.yaml)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-db
  labels:
    db: mongo-db
spec:
  replicas: 1
  selector:
    matchLabels:
      db: mongo-db
  template:
    metadata:
      labels:
        db: mongo-db
    spec:
      containers:
        - name: mongo-db
          image: mongo
          volumeMounts:
            - name: mongo-db-storage
              mountPath: /data/db
            - name: host
              mountPath: /host
```

```
volumes:
  - name: mongo-db-storage
    persistentVolumeClaim:
      claimName: db-pvc
  - name: host
    hostPath:
      path: /dataset
```

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: db-pvc
  labels:
    db: mongo-db
```

Data를 import하기 위해서, host를 mount 한다.

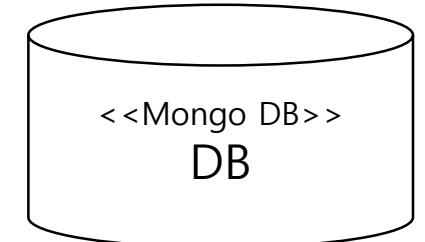
Hyper-V에서는 다음과 같이 공유한다.

1) 폴더를 공유한다.

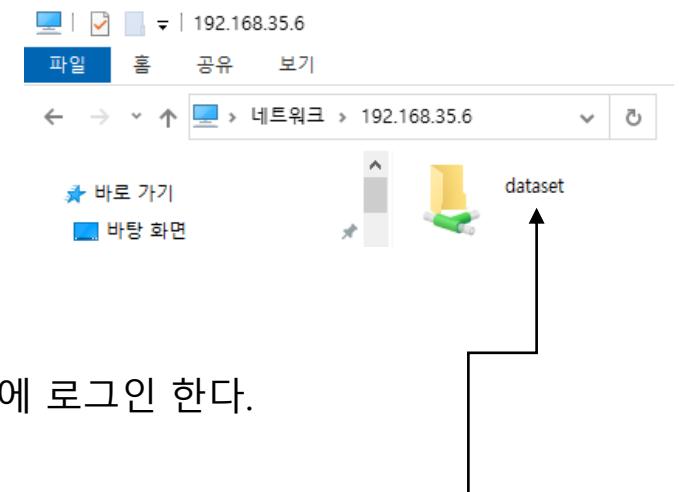
2) minikube ssh 명령으로 minikube 가상 머신에 로그인 한다.

```
$ sudo mkdir /dataset
```

```
$ sudo mount -t cifs -o "user={ID},password={PWD}" //192.168.35.6/dataset /dataset
```



Persistent Volume을 생성해서,
/data/db에 마운트 한다.



8. Map UI

<https://github.com/bosornd/map-ui>

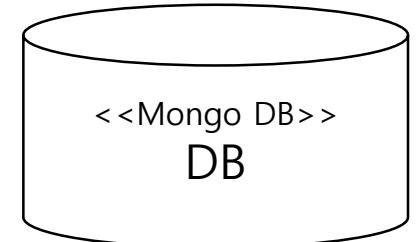
1) Mongo DB (db/db_deployment.yaml)

kubectl apply -f db_deployment.yaml 명령으로 Deployment와 Service를 생한다.

Mongo DB POD에 Shell로 접속한다.

- host 폴더를 확인한다.
- mongoimport로 데이터를 import한다.

\$ **mongoimport --db=yelp --collection=business --jsonArray --file=yelp_academic_dataset_business.json**



```
mongo-db-67b477f6c8-grr95의 mongo-db 에 셀 인
root@mongo-db-67b477f6c8-grr95:/# cd /host
root@mongo-db-67b477f6c8-grr95:/host# ls -agl
total 11207644
drwxr-xr-x 2 root      4096 Jul  3 07:16 .
drwxr-xr-x 1 root      4096 Jul  3 13:46 ..
drwxr-xr-x 2 root      0 Apr 19 05:55 photos
-rw xr-xr-x 1 root 146321327 Jun 18 11:13 yelp_academic_dataset_business.json
-rw xr-xr-x 1 root 449663480 Dec 13 2019 yelp_academic_dataset_checkin.json
-rw xr-xr-x 1 root 89091484 Jun 18 14:30 yelp_academic_dataset_geodata.json
-rw xr-xr-x 1 root 25615185 Jan  7 22:12 yelp_academic_dataset_photo.json
-rw xr-xr-x 1 root 6325565224 Dec 13 2019 yelp_academic_dataset_review.json
-rw xr-xr-x 1 root 263489322 Dec 13 2019 yelp_academic_dataset_tip.json
-rw xr-xr-x 1 root 4176861589 Jun 16 09:43 yelp_academic_dataset_user.json
root@mongo-db-67b477f6c8-grr95:/host# mongoimport --db=yelp --collection=business --jsonArray --file=yelp_academic_dataset_business.json
2020-07-03T13:48:58.500+0000    connected to: mongodb://localhost/
2020-07-03T13:49:01.500+0000    [####.....] yelp.business      25.4MB/140MB (18.2%)
2020-07-03T13:49:04.539+0000    [#####.....] yelp.business      49.7MB/140MB (35.6%)
2020-07-03T13:49:07.500+0000    [#####.....] yelp.business      75.3MB/140MB (53.9%)
2020-07-03T13:49:10.500+0000    [#####.....] yelp.business      99.7MB/140MB (71.5%)
2020-07-03T13:49:13.501+0000    [#####.....] yelp.business     127MB/140MB (90.7%)
2020-07-03T13:49:14.990+0000    [#####.....] yelp.business     140MB/140MB (100.0%)
2020-07-03T13:49:14.991+0000    209393 document(s) imported successfully. 0 document(s) failed to import.
root@mongo-db-67b477f6c8-grr95:/host#
```

8. Map UI

<https://github.com/bosornd/map-ui>

1) Mongo DB (db/db_deployment.yaml)

kubectl apply -f db_deployment.yaml 명령으로 Deployment와 Service를 생성한다.

Mongo DB POD의 Shell에 접속한다.

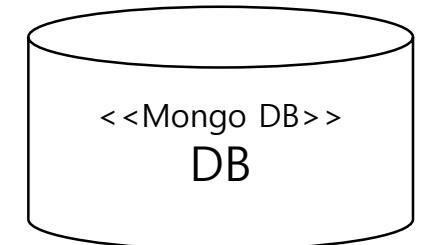
- host 폴더를 확인한다.
- mongoimport로 데이터를 import한다.
- business collection으로부터 geodata를 생성한다.

\$ **mongo** ← mongo client로 DB 서버에 연결한다.

connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

...

```
> use yelp
> db.business.find().forEach(function(biz){
  db.geodata.insert({ "geometry": { "type": "Point", "coordinates": [ biz.longitude, biz.latitude ] },
    "properties": { "business_id": biz.business_id, "name": biz.name, "stars": biz.stars,
      "review_count": biz.review_count, "categories": biz.categories } });
});
```



8. Map UI

<https://github.com/bosornd/map-ui>

2) API Server (db/models/business.js)

```
const mongoose = require('mongoose');
```

```
const businessSchema = new mongoose.Schema({
  business_id: { type: String, unique: true, index: true, required: true },
```

```
  name: String,
```

```
  address: String,
```

```
  city: String,
```

```
  state: String,
```

```
  postal_code: String,
```

```
  latitude: Number,
```

```
  longitude: Number,
```

```
  stars: Number,
```

```
  review_count: Number,
```

```
  is_open: Boolean,
```

```
  attributes: Object,
```

```
  categories: [String],
```

```
  hours: {
```

```
    Monday: String,
```

```
    Tuesday: String,
```

```
    Wednesday: String,
```

```
    Thursday: String,
```

```
    Friday: String,
```

```
    Saturday: String,
```

```
    Sunday: String,
```

```
}
```

```
,
```

```
  {
    collection: 'business'          // collection name
  }
);
```

```
businessSchema.index({name: 'text', categories: 'text'});
```

```
businessSchema.statics.findById = function(ID){
  return this.find({business_id: ID});
};
```

```
businessSchema.statics.findByKeyword = function(keyword){
  return this.find({$text: {$search: keyword}});
};
```

```
module.exports = mongoose.model('business', businessSchema);
```

<<Express>>
API Server

8. Map UI

2) API Server (db/models/geodata.js)

```
const mongoose = require('mongoose');

const pointSchema = new mongoose.Schema({
  type: { type: String, enum: ['Point'], required: true },
  coordinates: { type: [Number], required: true }
});

const geodataSchema = new mongoose.Schema({
  geometry: pointSchema,
  properties: {
    business_id: { type: String, unique: true, index: true, required: true },
    name: String,
    stars: Number,
    review_count: Number,
    categories: [String],
  },
  {
    collection: 'geodata' // collection name
  }
);

geodataSchema.index({geometry: '2dsphere'});

geodataSchema.statics.findInBox = function(lowerLeft, upperRight){
  return this.find().where('geometry').box(lowerLeft, upperRight);
};

geodataSchema.statics.findNear = function(center, maxDistance){
  return this.find().where('geometry')
    .near({center: center, maxDistance: maxDistance, spherical: true});
};

module.exports = mongoose.model('geodata', geodataSchema);
```

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/apis/business.js)

```
const express = require('express');
const router = express.Router();

const Business = require('../models/business');

router.get('/search', function(request, response) {
    Business.findByKeyword(request.query.keyword)
        .sort({stars: -1}).limit(10).exec(function(error, result){
            if (error) return response.status(500).json(error);
            return response.json(result);
        });
});

router.get('/:id', function(request, response) {
    Business.findById(request.params.id).exec(function(error, result){
        if (error) return response.status(500).json(error);
        return response.json(result[0]);
    });
});

module.exports = router;
```

<<Express>>
API Server

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/apis/geodata.js)

```
const express = require('express');
const router = express.Router();

const geodata = require('../models/geodata');

router.get('/within', function(request, response) {
  geodata.findInBox([Number(request.query.left), Number(request.query.lower)],
    [Number(request.query.right), Number(request.query.upper)])
    .exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});

router.get('/near', function(request, response) {
  geodata.findNear([Number(request.query.longitude), Number(request.query.latitude)],
    Number(request.query.maxDistance))
    .limit(10).exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});
```

<<Express>>
API Server

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/apis/geodata.js)

```
router.get('/search', function(request, response) {
  console.log(request.query.keyword);
  geodata.findByKeyword(request.query.keyword)
    .sort({stars: -1}).limit(10).exec(function(error, result){
      if (error) return response.status(500).json(error);
      return response.json(result);
    });
});

router.get('/:id', function(request, response) {
  geodata.findById(request.params.id).exec(function(error, result){
    if (error) return response.status(500).json(error);
    return response.json(result[0]);
  });
}

module.exports = router;
```

<<Express>>
API Server

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/server.js)

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://' + process.env.MONGO_DB_SERVICE_HOST + '/yelp',
    {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true});
mongoose.set('debug', true);

const express = require('express');
const app = express();
const port = 8080;

app.use('/api/business', require('./apis/business'));
app.use('/api/geodata', require('./apis/geodata'));

app.listen(port);
```

<<Express>>
API Server

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/package.json)

```
{  
  "name": "server",  
  "version": "0.0.1",  
  "description": "DB API SERVER",  
  "author": "Yong Jin, Cho <drajin.cho@bosornd.com>",  
  "scripts": {  
    "start": "node server.js"  
  },  
  "dependencies": {  
    "express": "latest",  
    "mongoose": "latest"  
  }  
}
```

<<Express>>
API Server

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/dockerfile)

```
FROM node:latest
```

```
WORKDIR /src
```

```
COPY . .
```

```
RUN npm install
```

```
EXPOSE 8080
```

```
CMD npm start
```

<<Express>>
API Server

docker build -t api-server:v1 . 명령으로 이미지를 빌드한다.

docker images 명령으로 이미지를 확인한다.

8. Map UI

<https://github.com/bosornd/map-ui>

3) API Server (db/api_deployment.yaml)

kubectl apply -f api_deployment.yaml 명령으로 Deployment와 Service를 생성한다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-server
  labels:
    svc: api-server
spec:
  selector:
    matchLabels:
      svc: api-server
  template:
    metadata:
      labels:
        svc: api-server
    spec:
      containers:
        - name: api-server
          image: 'api-server:v1'
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: api-server
  labels:
    svc: api-server
spec:
  ports:
    - port: 8080
  selector:
    svc: api-server
```

<<Express>>
API Server

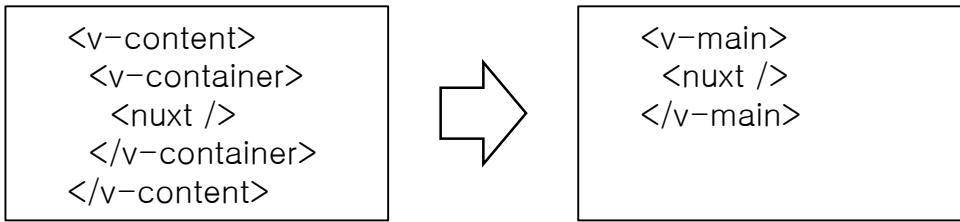
8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

npx create-nuxt-app frontend로 프로젝트를 생성한다.

layouts/default.vue nuxt 부분을 다음과 같이 수정한다.



<<Nuxt>>
Frontend Server

nuxt.config.js 에 Google Maps API 소스를 추가한다.

```
head: {  
  ...  
  script: [  
    { src: 'https://maps.googleapis.com/maps/api/js?key=YOUR_KEY&libraries=geometry' }  
  ]  
},
```

8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

pages/index.vue를 다음과 같이 수정한다.

```
<template>
  <v-container fluid fill-height>
    <div id="map"></div>
  </v-container>
</template>

<style>
  #map {
    width: 100%;
    height: 100%;
  }
</style>

<script>
export default {
  mounted(){
    const map = new google.maps.Map(document.getElementById('map'), {
      center: { lat: 35.25, lng: -80.85 },
      zoom: 16
    });
  }
}
</script>
```

<<Nuxt>>
Frontend Server

8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

pages/index.vue를 다음과 같이 수정한다.

```
map.addListener('idle', function(){
  // remove all
  map.data.forEach(function(feature){
    map.data.remove(feature);
  });

  const bounds = map.getBounds();
  const sw = bounds.getSouthWest(), ne = bounds.getNorthEast();

  const request = new XMLHttpRequest();
  const url = 'http://minikube/api/geodata/within?' + 'left=' + sw.lng() + '&lower=' + sw.lat() + '&right=' + ne.lng() + '&upper=' + ne.lat();
  request.open('GET', url);
  request.responseType = 'json';
  request.send();

  request.onload = function() {
    const geodata = request.response;
    for(var i = 0; i < geodata.length; i++){
      const latLng = new google.maps.LatLng({ lng: geodata[i].geometry.coordinates[0], lat: geodata[i].geometry.coordinates[1] });
      map.data.add(new google.maps.Data.Feature({ geometry: latLng, properties: geodata[i].properties } ));
    }
  }
});
```

<< Nuxt >>
Frontend Server

8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

pages/index.vue를 다음과 같이 수정한다.

```
const infowindow = new google.maps.InfoWindow();

map.data.addListener('click', function(e){
    const feature = e.feature;
    const name = feature.getProperty("name");
    const stars = feature.getProperty("stars");
    const review_count = feature.getProperty("review_count");
    const categories = feature.getProperty("categories");

    let content = '<div style="color:black"><b>' + name + '</b>(stars: ' + stars + ', reviews: ' + review_count + ')<br>categories: ';
    content += categories[0];
    for(var i = 1; i < categories.length; i++)
        content += ', ' + categories[i];
    content += '</div>';

    infowindow.setPosition(e.latLng);
    infowindow.setContent(content);
    infowindow.open(map);
});

</script>
```

<<Nuxt>>
Frontend Server

8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

package.json 은 다음과 같다.

```
{  
  "name": "frontend",  
  "version": "1.0.0",  
  "private": true,  
  "config": {  
    "nuxt": {  
      "host": "0.0.0.0", ──────────────────> 설정안하면 localhost만 접근 가능함.  
      "port": "8080" ─────────────────> 포트  
    }  
  },  
  "scripts": {  
    "dev": "nuxt", ─────────────────> 개발 버전 실행  
    "build": "nuxt build", ─────────────────> 서비스 버전 빌드  
    "start": "nuxt start", ─────────────────> 서비스 버전 실행  
    "export": "nuxt export",  
    "serve": "nuxt serve"  
  },  
  "dependencies": {  
    "nuxt": "^2.13.0"  
  },  
  "devDependencies": {  
    "@nuxtjs/vuetify": "^1.11.2"  
  }  
}
```

<< Nuxt >>
Frontend Server

dockerfile 은 다음과 같다.

```
FROM node:latest  
  
WORKDIR /src  
COPY . .  
RUN npm install  
RUN npm run build  
  
EXPOSE 8080  
CMD npm run start
```

docker build -t frontend-server:v1 . 명령으로 이미지를 생성한다.

8. Map UI

<https://github.com/bosornd/map-ui>

4) Frontend Server (frontend/frontend_deployment.yaml)

kubectl apply -f frontend_deployment.yaml 명령으로 Deployment와 Service를 생성한다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-server
  labels:
    app: frontend-server
spec:
  selector:
    matchLabels:
      app: frontend-server
  template:
    metadata:
      labels:
        app: frontend-server
    spec:
      containers:
        - name: frontend-server
          image: 'frontend-server:v1'
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-server
  labels:
    app: frontend-server
spec:
  ports:
    - port: 8080
  selector:
    app: frontend-server
```

<<Nuxt>>
Frontend Server

8. Map UI

<https://kubernetes.io/ko/docs/concepts/services-networking/ingress/>
<https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/>

5) Ingress 설정하기 (frontend/ingress_deployment.yaml)

인그레스(Ingress)

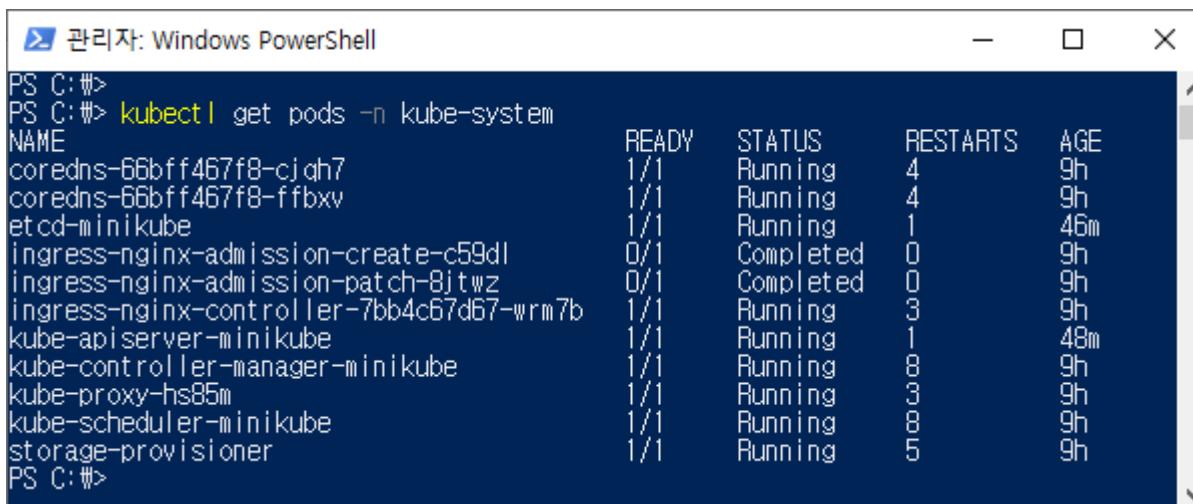
FEATURE STATE: Kubernetes v1.1 [beta]

클러스터 내의 서비스에 대한 외부 접근을 관리하는 API 오브젝트이며, 일반적으로 HTTP를 관리함.

인그레스는 부하 분산, SSL 종료, 명칭 기반의 가상 호스팅을 제공할 수 있다.

minikube addons enable ingress 명령으로 인그레스를 활성화 한다.

kubectl get pods -n kube-system 명령으로 NginX Ingress Controller가 동작함을 확인한다.



```
PS C:\#> kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS   AGE
coredns-66bff467f8-cjqh7          1/1     Running   4          9h
coredns-66bff467f8-ffbxy          1/1     Running   4          9h
etcd-minikube                      1/1     Running   1          46m
ingress-nginx-admission-create-c59dl 0/1     Completed  0          9h
ingress-nginx-admission-patch-8jtzw 0/1     Completed  0          9h
ingress-nginx-controller-7bb4c67d67-wrm7b 1/1     Running   3          9h
kube-apiserver-minikube            1/1     Running   1          48m
kube-controller-manager-minikube   1/1     Running   8          9h
kube-proxy-hs85m                   1/1     Running   3          9h
kube-scheduler-minikube           1/1     Running   8          9h
storage-provisioner                1/1     Running   5          9h
PS C:\#>
```

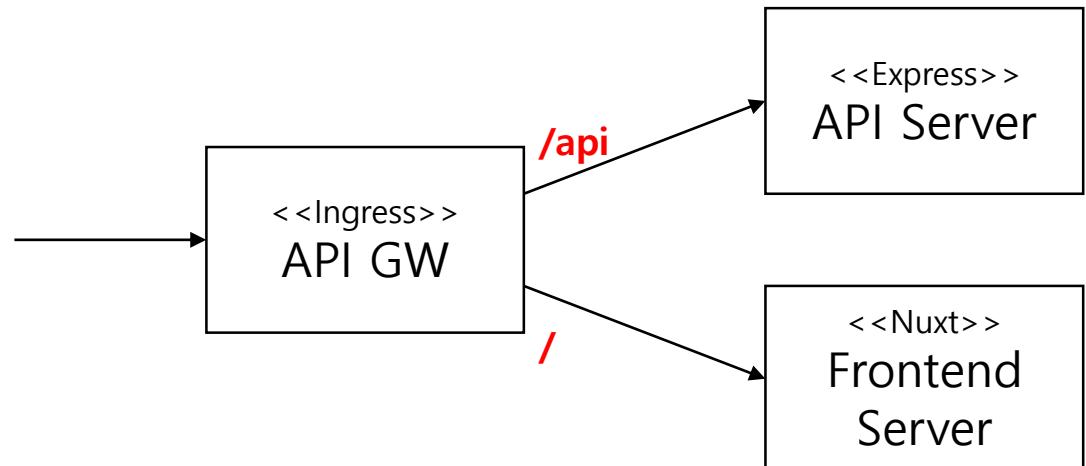
8. Map UI

<https://kubernetes.io/ko/docs/concepts/services-networking/ingress/>
<https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/>

5) Ingress 설정하기 (frontend/ingress_deployment.yaml)

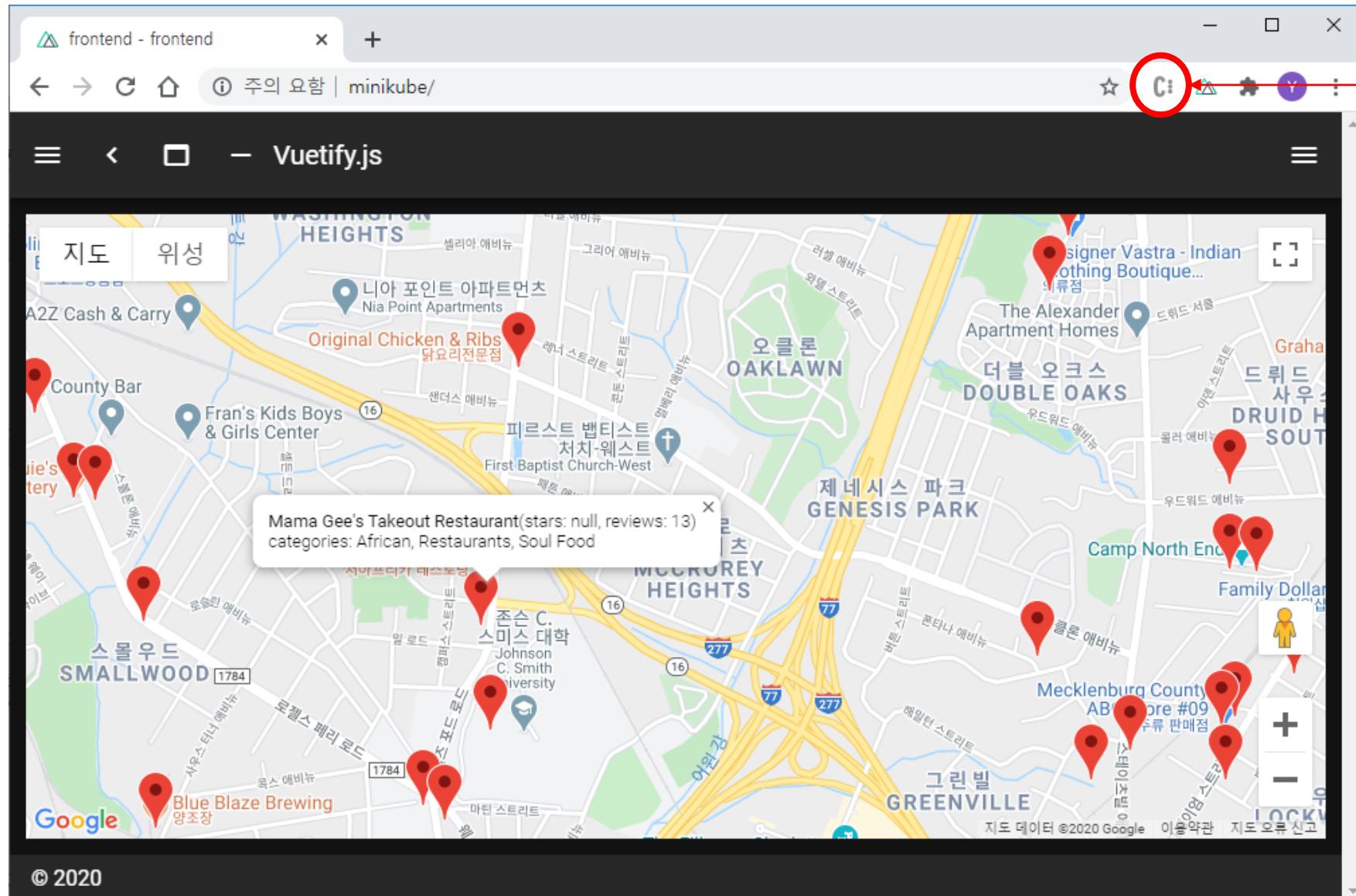
kubectl apply -f ingress_deployment.yaml로 인그레스 정책을 설정한다.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-api-gateway
spec:
  rules:
  - http:
    paths:
    - path: /
      backend:
        serviceName: frontend-server
        servicePort: 8080
    - path: /api
      backend:
        serviceName: api-server
        servicePort: 8080
```



8. Map UI

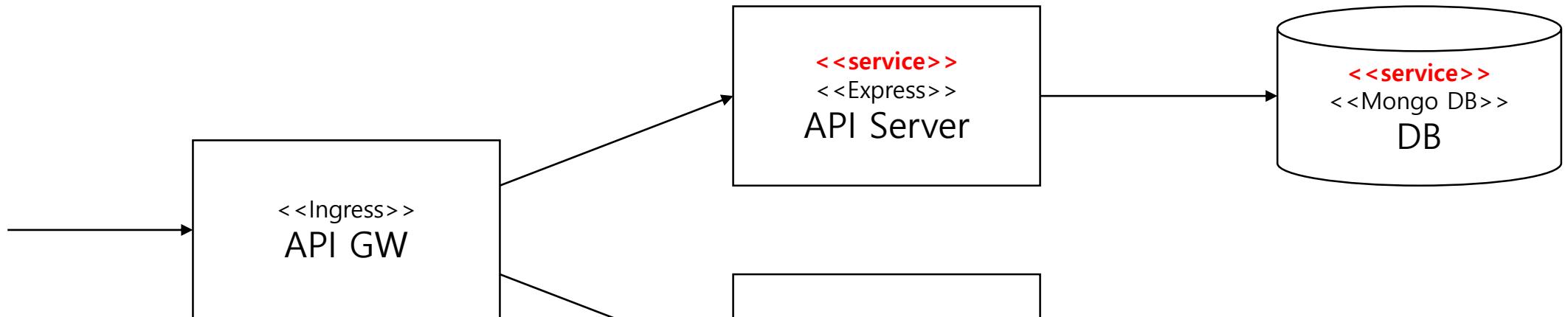
<https://github.com/bosornd/map-ui>



CORS 문제가 없음.

8. Map UI

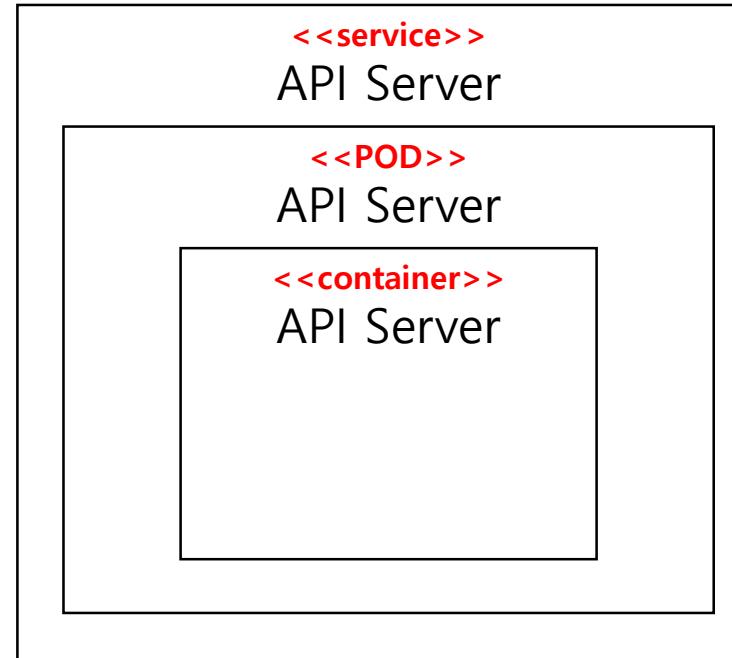
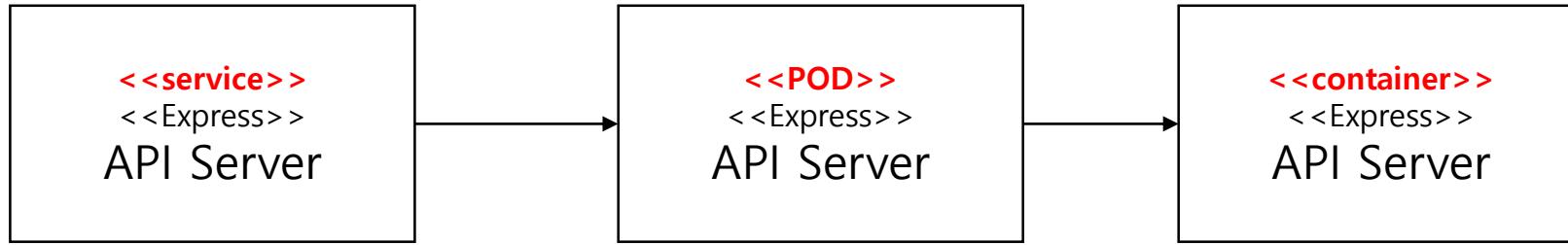
<https://github.com/bosornd/map-ui>



서비스						
이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간
api-server	default	svc: api-server	10.104.61.72	api-server:8080 TCP api-server:0 TCP	-	10 minutes ago
frontend-server	default	app: frontend-server	10.102.146.244	frontend-server:8080 TCP frontend-server:0 TCP	-	8 minutes ago
kubernetes	default	component: apiserver provider: kubernetes	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	9 hours ago
mongo-db	default	db: mongo-db	10.111.183.28	mongo-db:27017 TCP mongo-db:0 TCP	-	9 hours ago

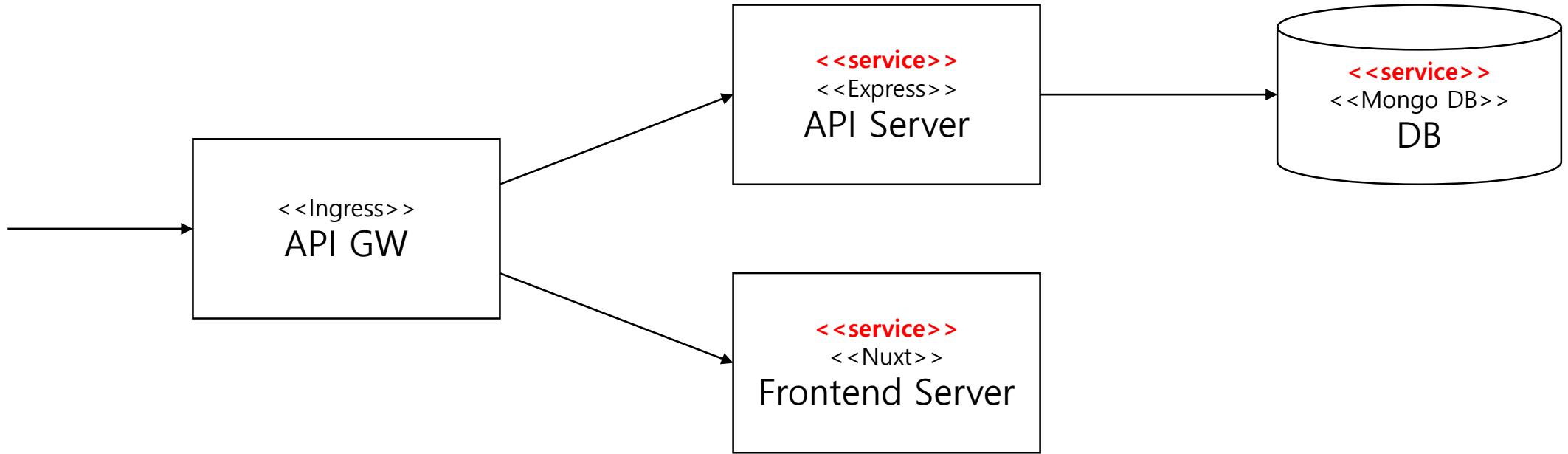
8. Map UI

<https://github.com/bosornd/map-ui>



8. Map UI

<https://github.com/bosornd/map-ui>

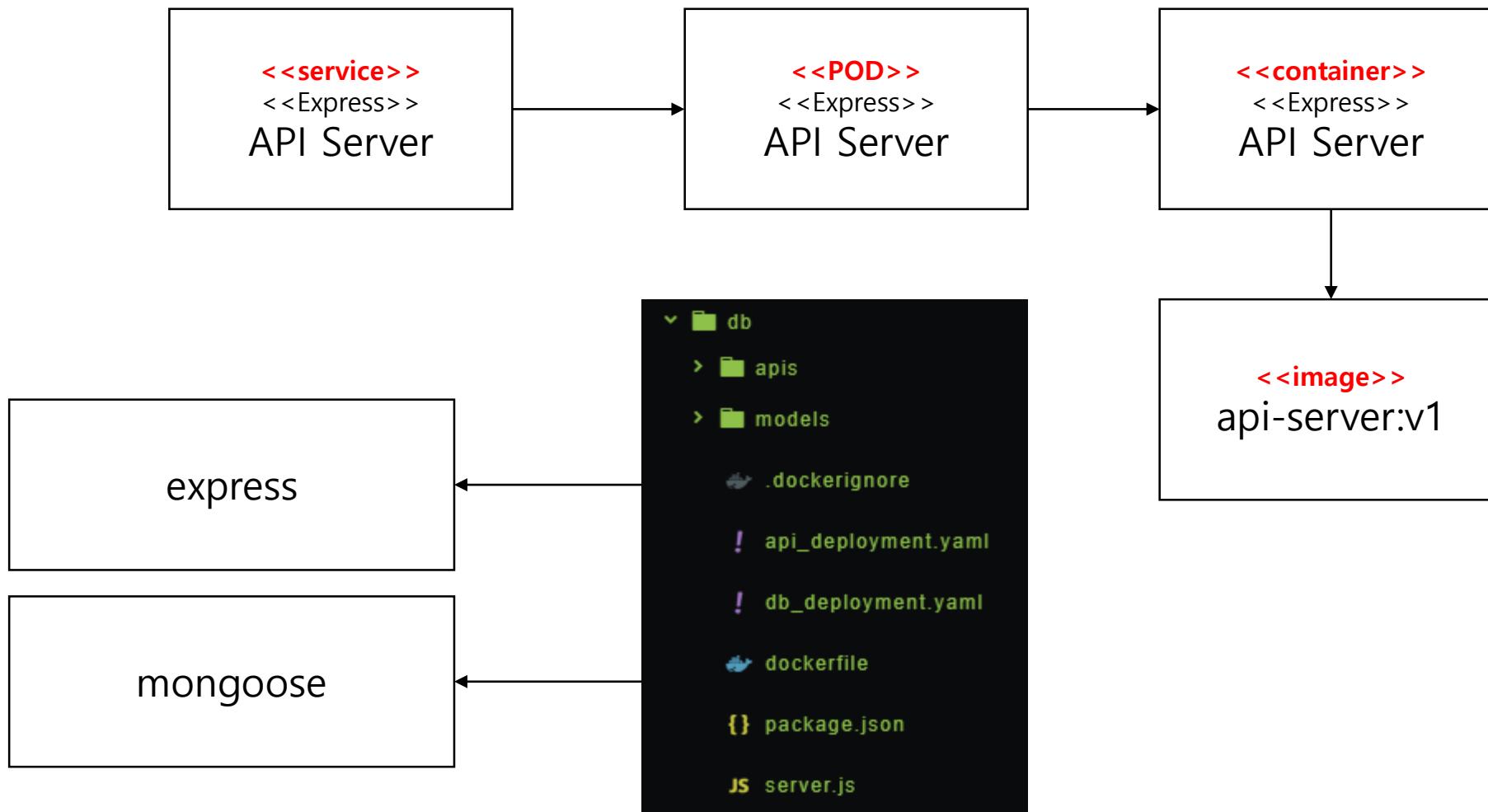


```
PS C:\#> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
frontend-server	v1	bc44e9c9b04b	42 minutes ago	1.13GB
api-server	v1	b3fe14288fa5	10 hours ago	956MB
node	latest	37ad18cd8bd1	3 days ago	943MB
mongo	latest	2b2cc1f48aed	2 weeks ago	388MB
k8s.gcr.io/kube-proxy	v1.18.3	3439b7546f29	6 weeks ago	117MB
k8s.gcr.io/kube-scheduler	v1.18.3	76216c34ed0c	6 weeks ago	95.3MB
k8s.gcr.io/kube-controller-manager	v1.18.3	da26705ccb4b	6 weeks ago	162MB
k8s.gcr.io/kube-apiserver	v1.18.3	7e28efa976bd	6 weeks ago	173MB
quay.io/kubernetes-ingress-controller/nginx-ingress-controller	0.32.0	70144d369cb2	2 months ago	327MB
kubernetesui/dashboard	v2.0.0	8b32422733b3	2 months ago	222MB
k8s.gcr.io/pause	3.2	80d28bedfe5d	4 months ago	683kB
k8s.gcr.io/coredns	1.6.7	67da37a9a360	5 months ago	43.8MB
jetttech/kube-webhook-certgen	v1.2.0	b7f5a2787829	8 months ago	49MB

8. Map UI

<https://github.com/bosornd/map-ui>



`==` VS. `== =`

- `==`, `!=`은 비교 전에 자료형을 일치시키고 비교함.
 - `== =`, `!== =`은 자료형도 동일해야 함.
-
- `1 == '1' → true`
 - `1 == = '1' → false`
 - `0 == false → true`
 - `0 == = false → false`

null vs. undefined

- null ← 의도적으로 비어 있음을 표현하기 위한 값
 - undefined ← 값이 할당되어 있지 않음
-
- $\text{null} \neq 0 \rightarrow \text{true}$
 - $\text{undefined} \neq 0 \rightarrow \text{true}$
 - $\text{null} == \text{undefined} \rightarrow \text{true}$
 - $\text{null} !== \text{undefined} \rightarrow \text{true}$

```
var a = null, b
      // → null
b      // → undefined
a == b // → true
a === b // → false
```

변수의 범위

- var ← function scope
- let ← block scope
- const ← block scope / constant