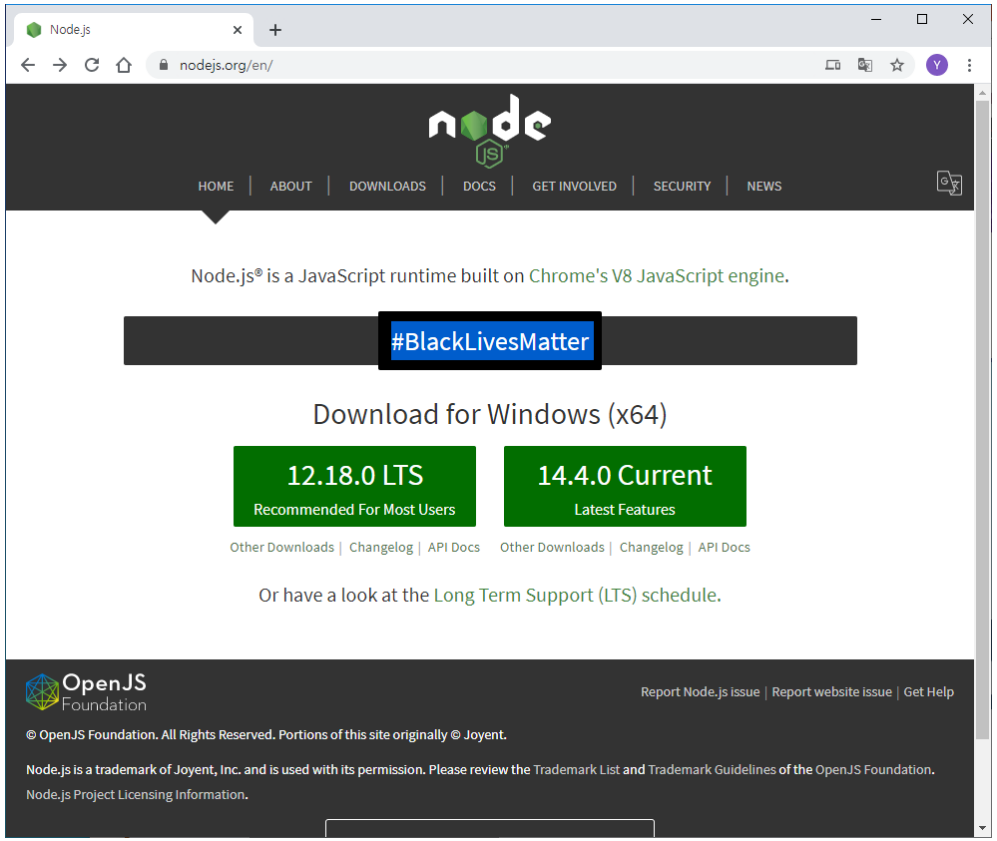
Tutorials for Map-UI

0. 설치 가이드

A. Node.js를 설치한다.

- https://nodejs.org 에서 LTS 버전을 설치한다.

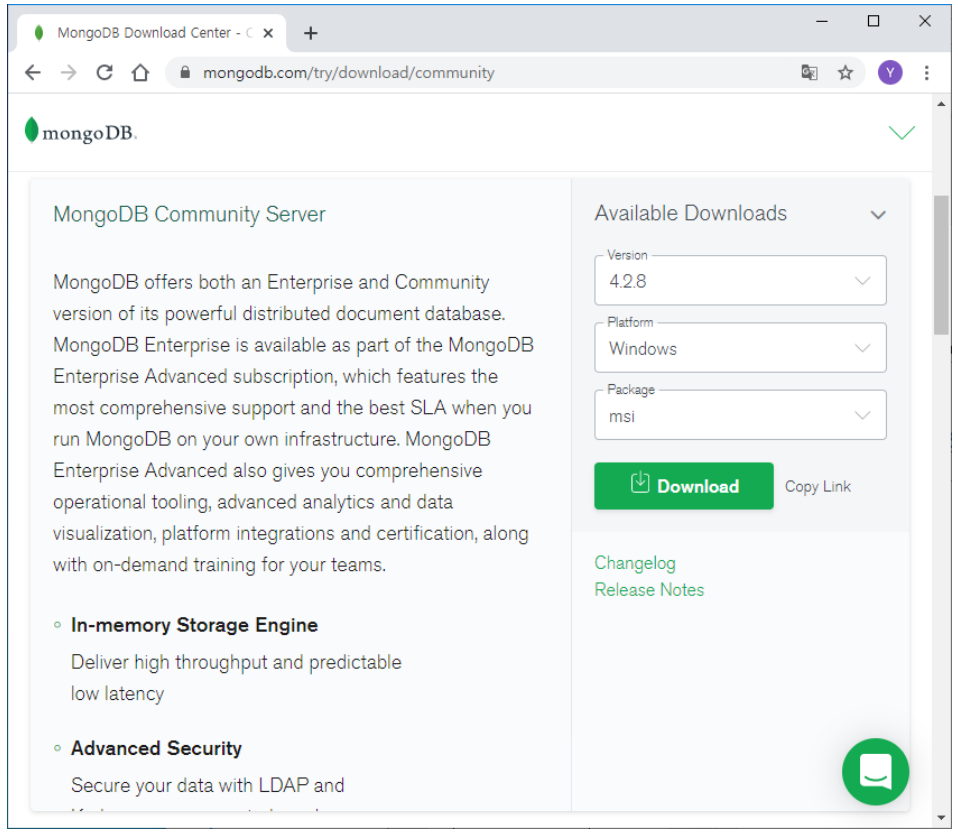


https://nodejs.org/

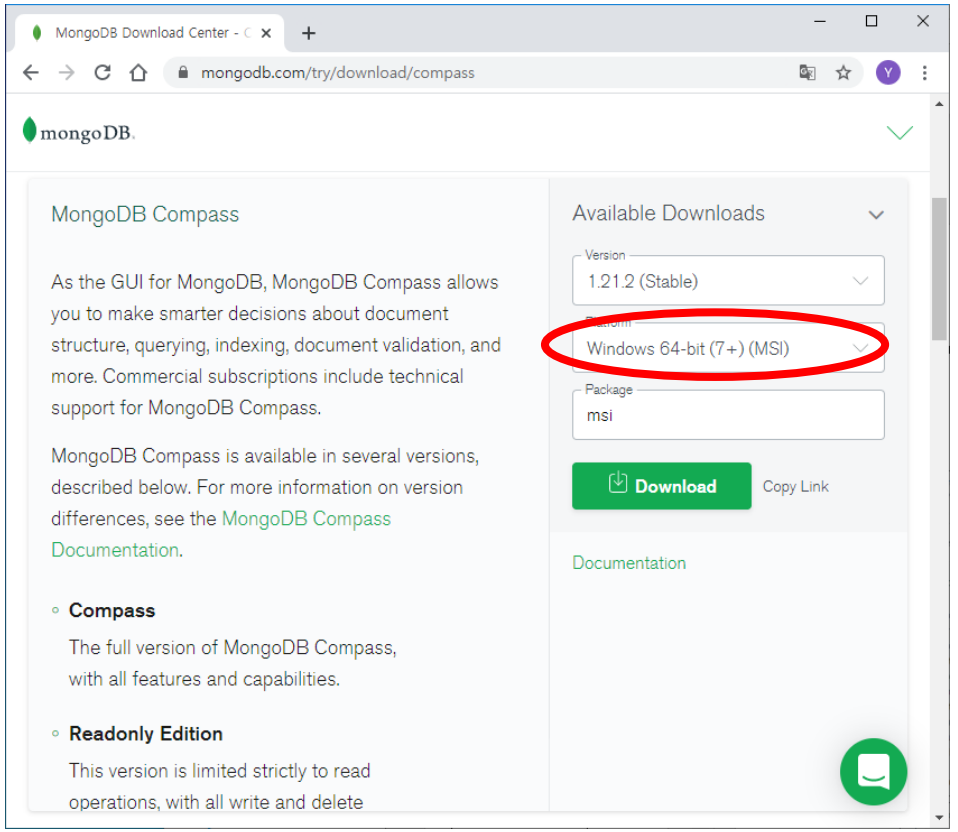
C:₩> npm install http-server -g

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)

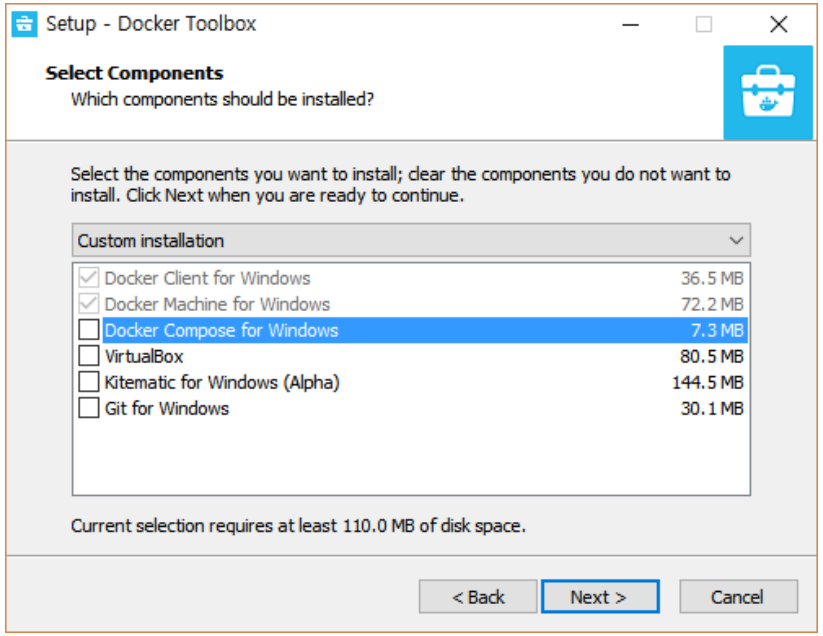
3) Kubectl을 설치한다.

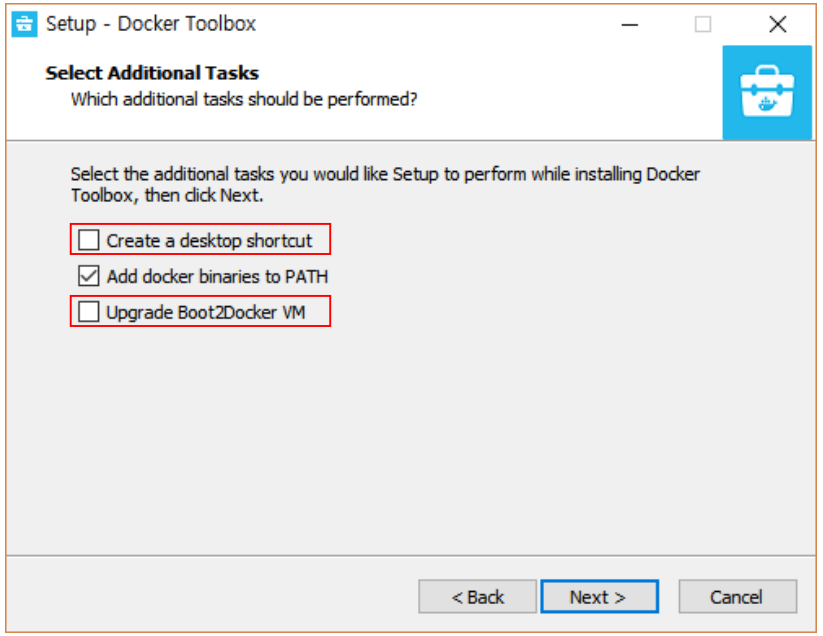
(https://kubernetes.io/ko/docs/tasks/tools/install-kubectl/)

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 보이기(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>
```

```
    <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>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>
```

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>
```

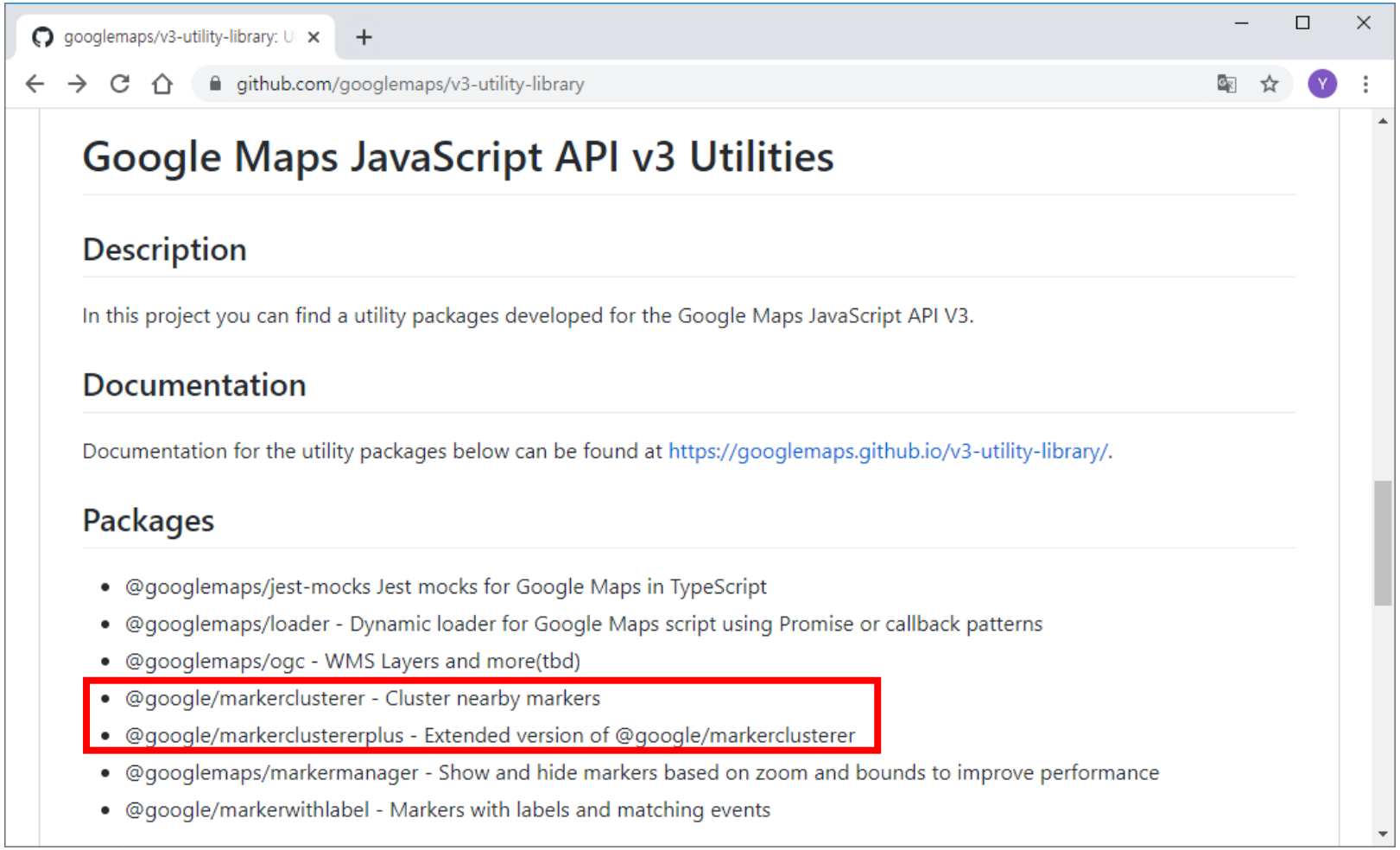
1. Google Maps API 가이드

- 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)



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, 37.57004847771681 ] },
                             "properties": { "name": "종각역" } },
      ]
    };

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

1. Google Maps API 가이드

- GeoJSON - https://en.wikipedia.org/wiki/GeoJSON

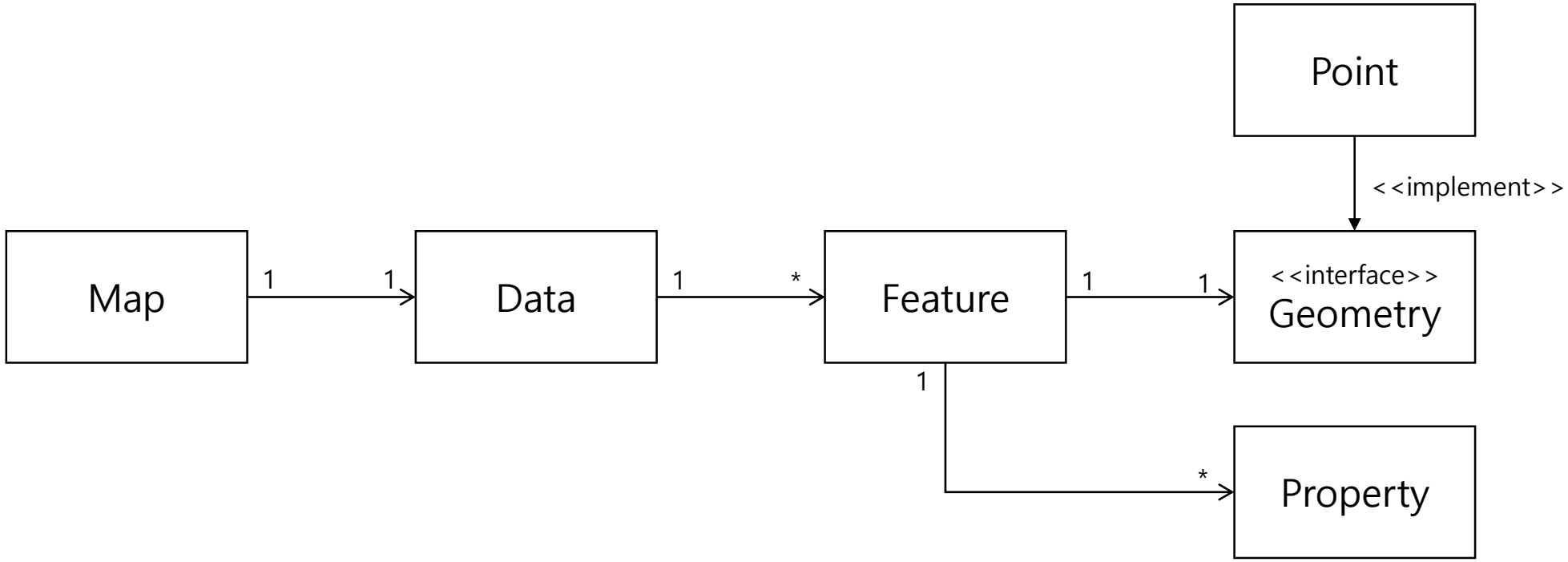
  an open standard format designed for representing simple geographical features, along with their non-spatial attributes.

  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": [126.970817503704 84, 37.555067886767 46] },
"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>
```

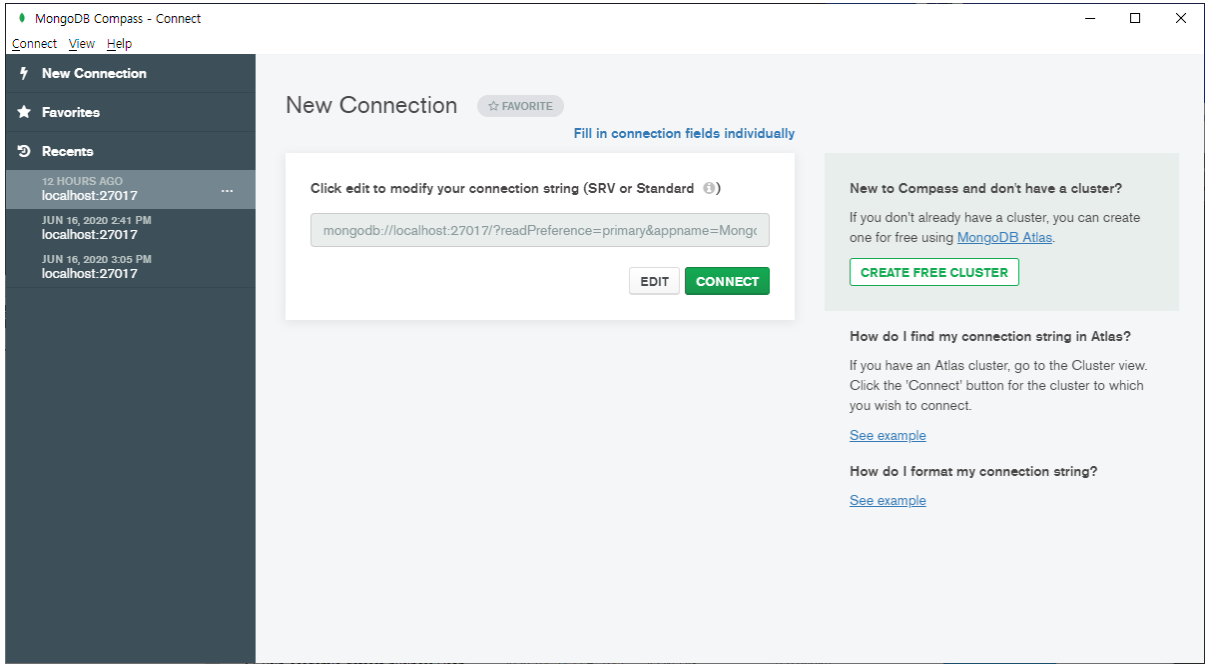
1. Google Maps API 가이드

- 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

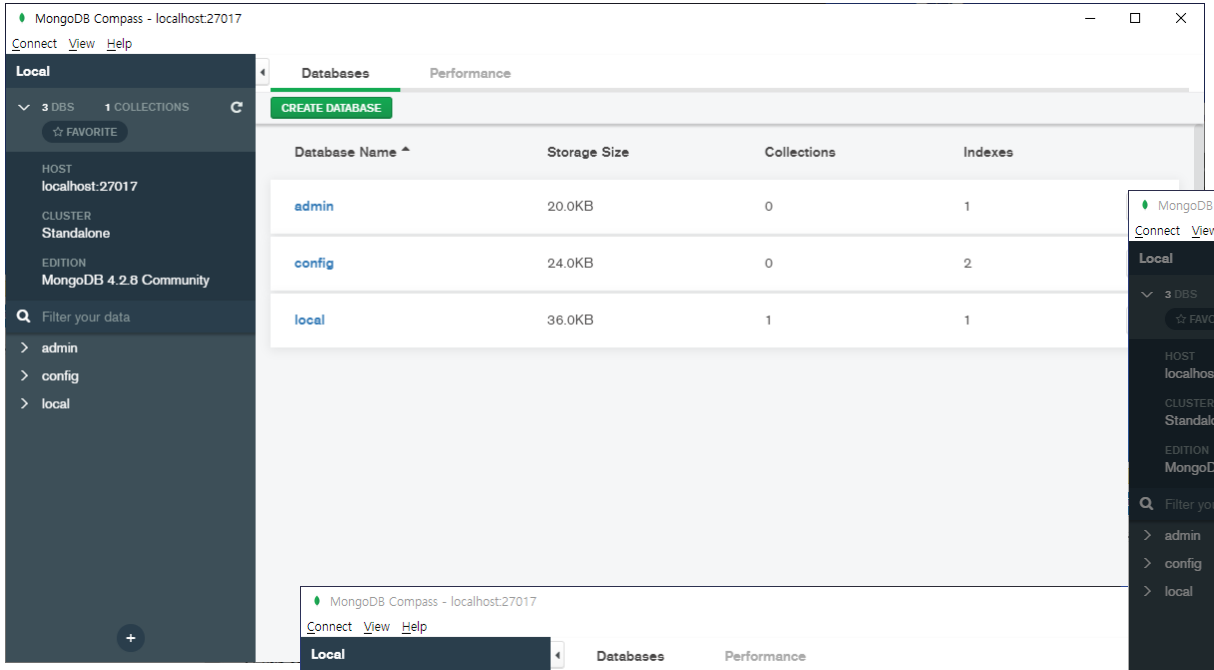
- MongoDB compass (GUI client)
  - MongoDB를 설치하면 서버가 실행된다. (mongodb://localhost:27017)
  - CONNECT를 눌러서 서버에 연결한다.



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

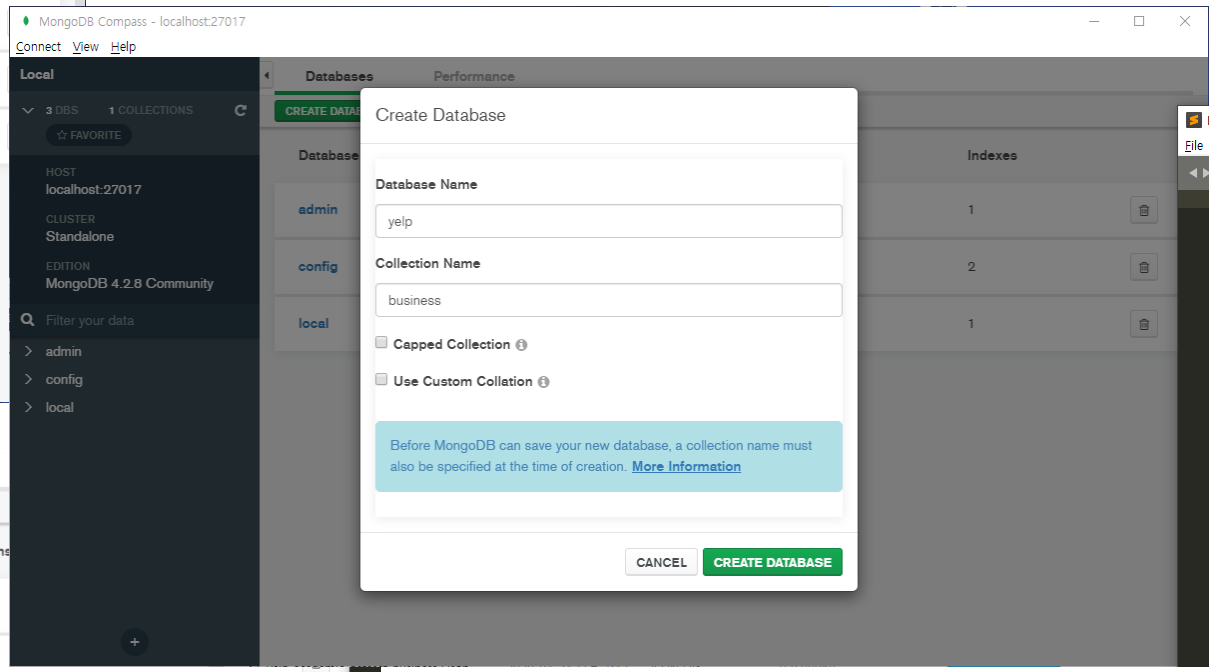
2. MongoDB 가이드

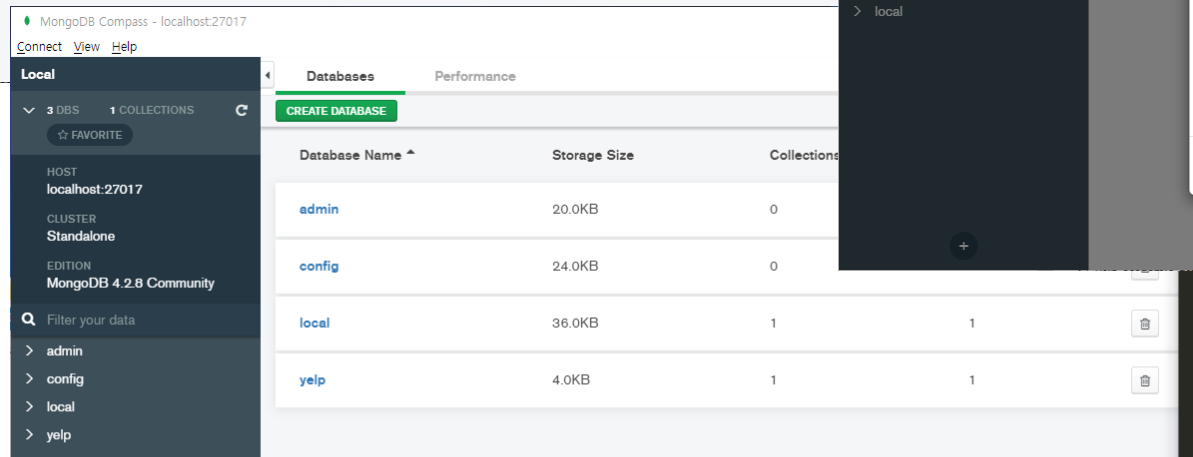
- yelp DB와 business collection을 생성한다.



1. CREATE DATABASE 버튼을 누른다.

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

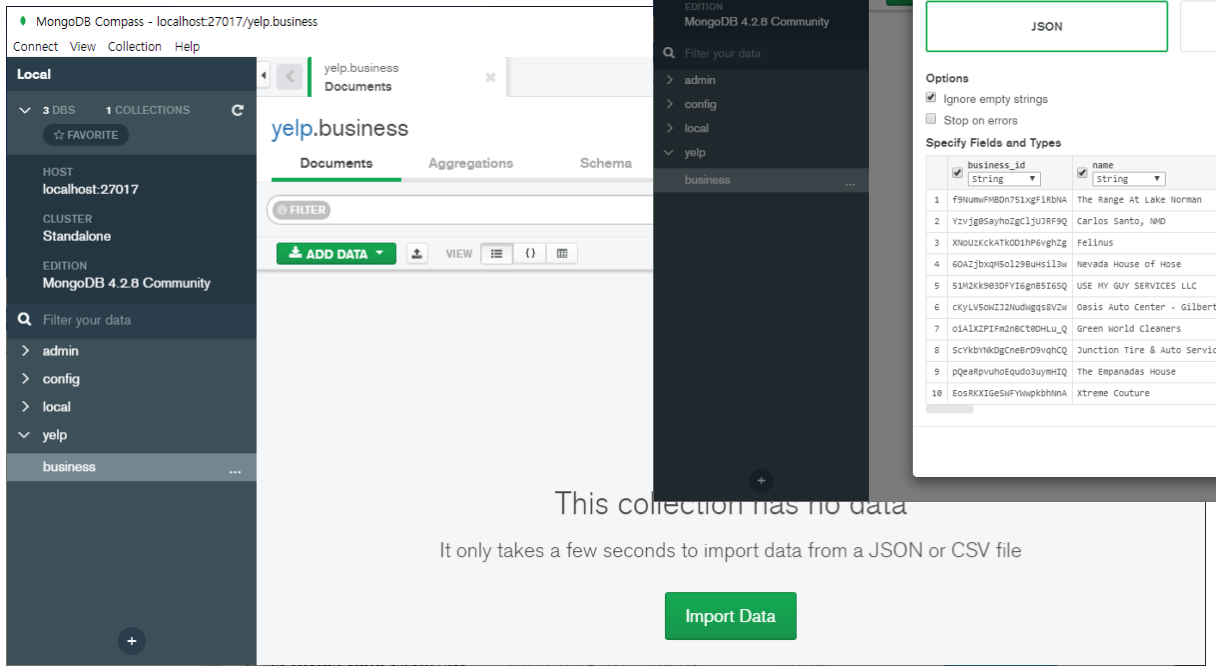




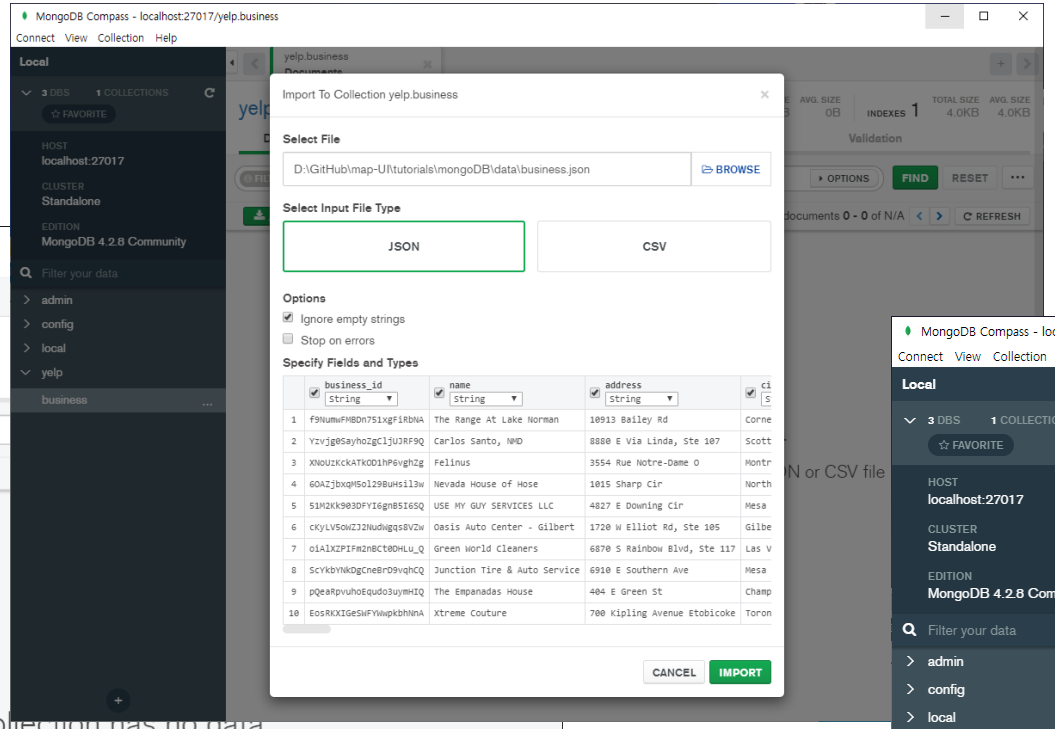
3. yelp DB가 생성되었음을 확인할 수 있다.

2. MongoDB 가이드

- business collection에 business.json을 로드한다.

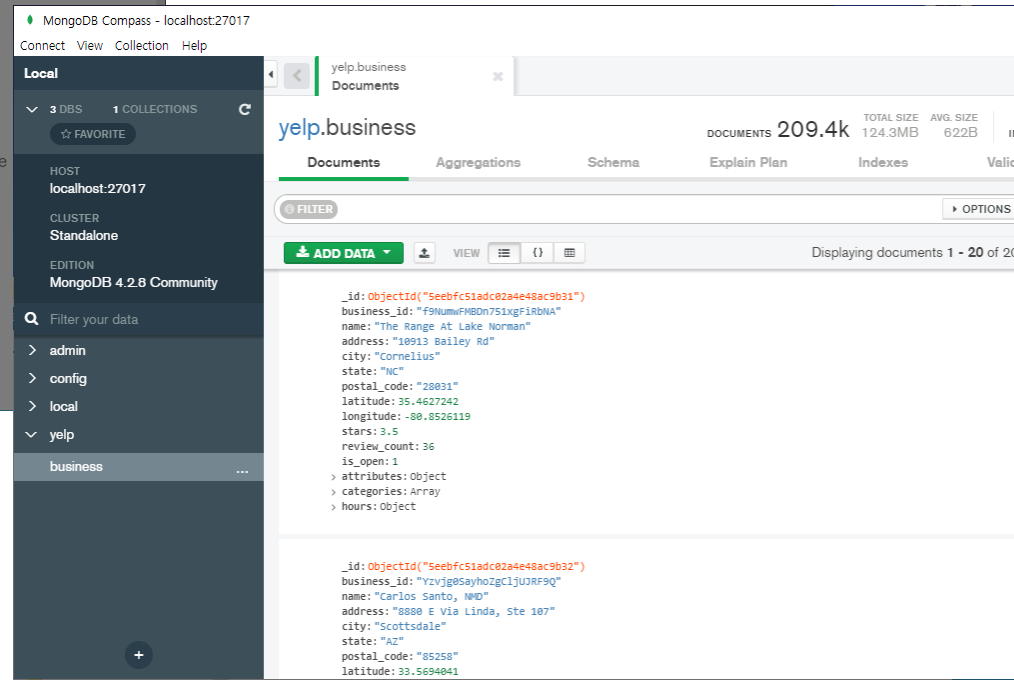


1. Import Data 버튼을 누른다.



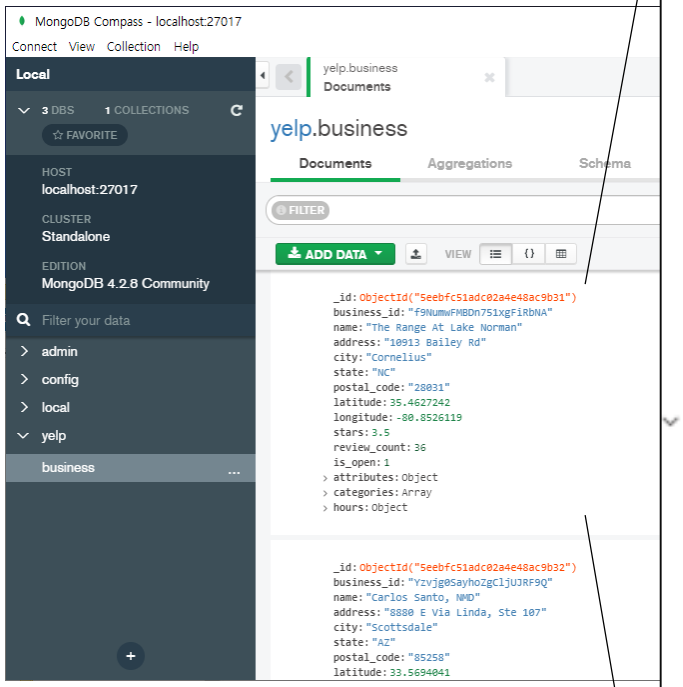
2. json 파일을 선택하고, IMPORT 버튼을 누른다.

3. business.json을 로드한 결과



2. MongoDB 가이드

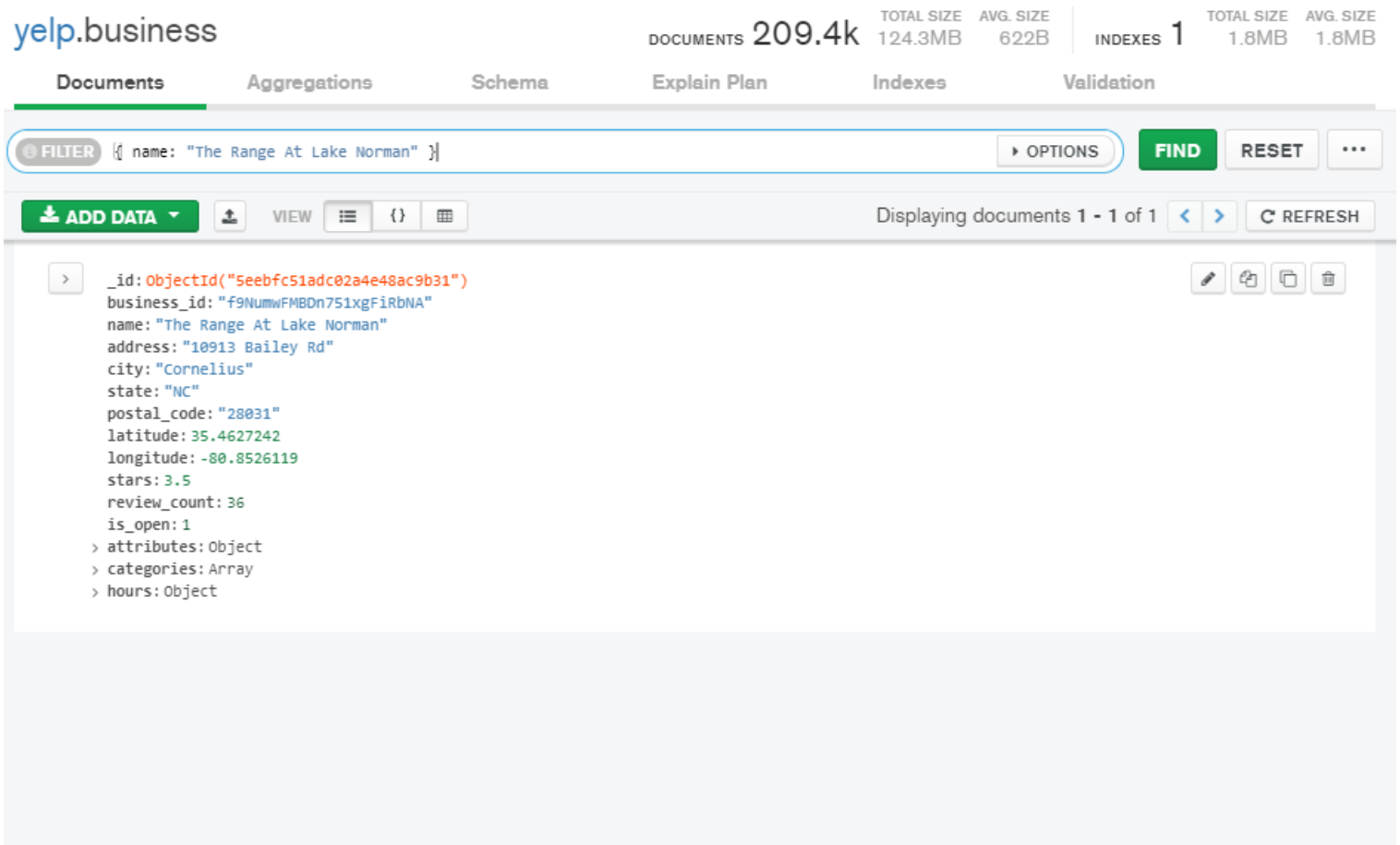
- business collection




```
_id: ObjectId("5eebfc51adc02a4e48ac9b31")
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
```

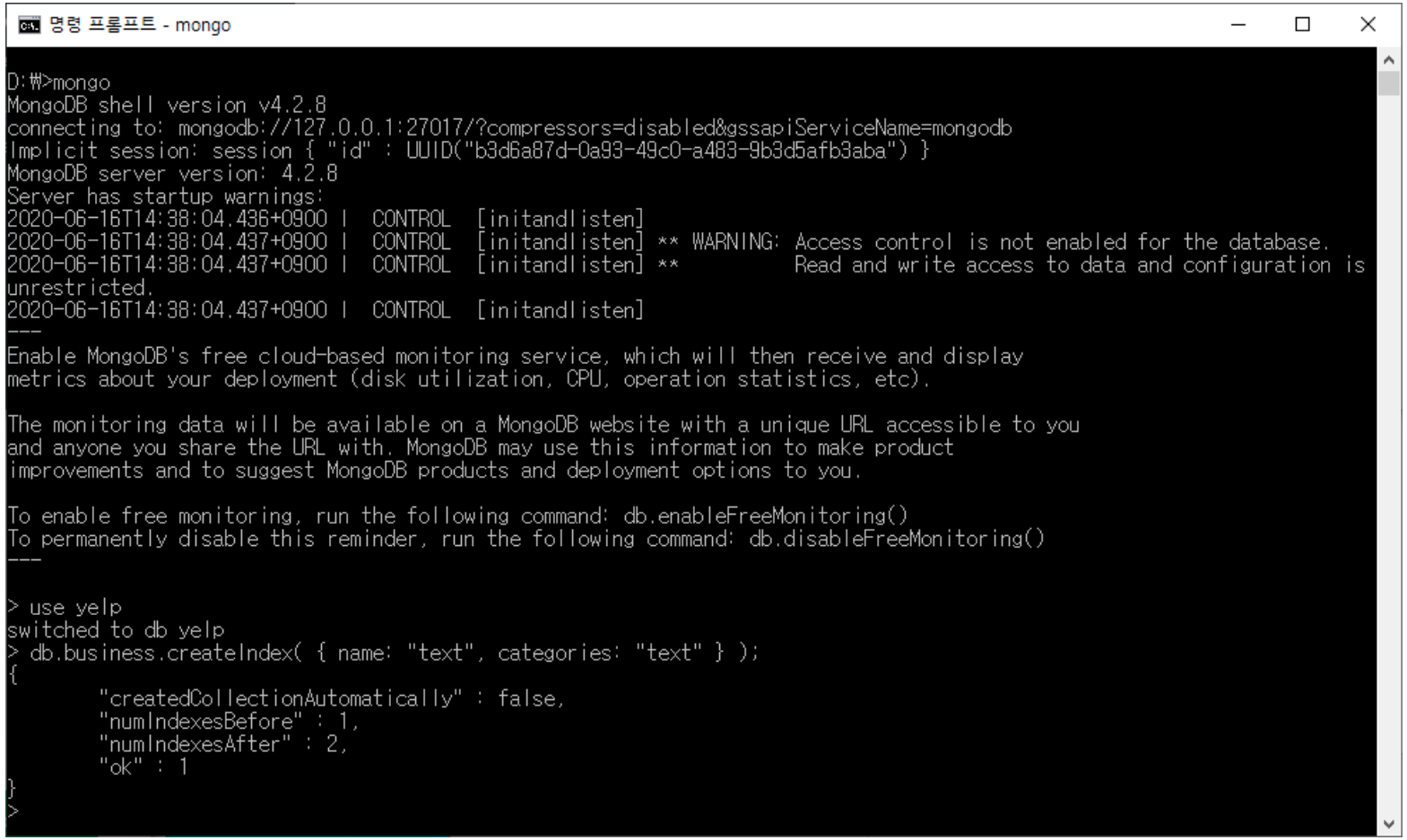
2. MongoDB 가이드

- business collection 검색하기



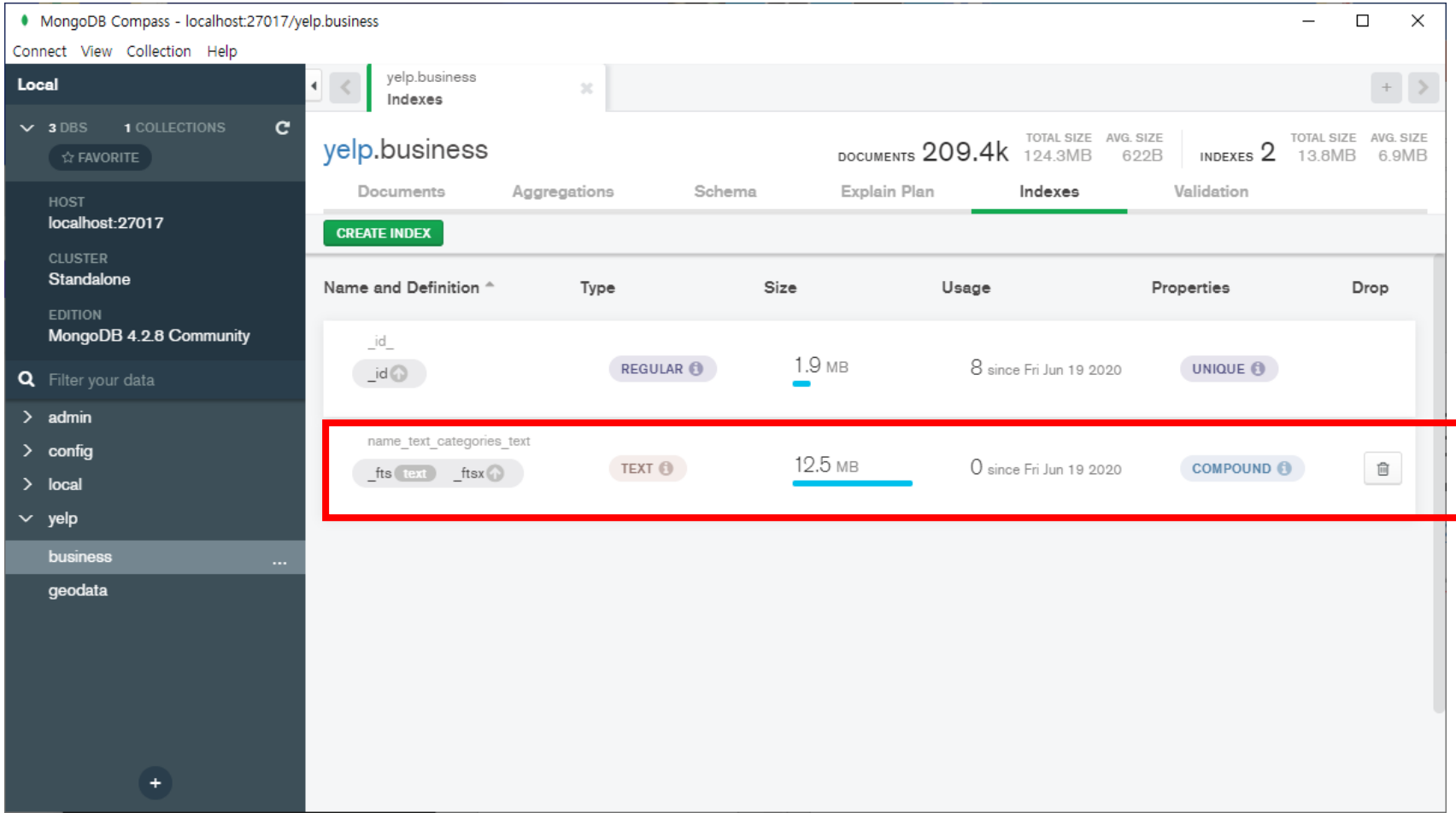
2. MongoDB 가이드

- Text search (https://docs.mongodb.com/manual/text-search/)
  - text index를 생성한다. db.business.createIndex({ name: "text", categories: "text" });



2. MongoDB 가이드

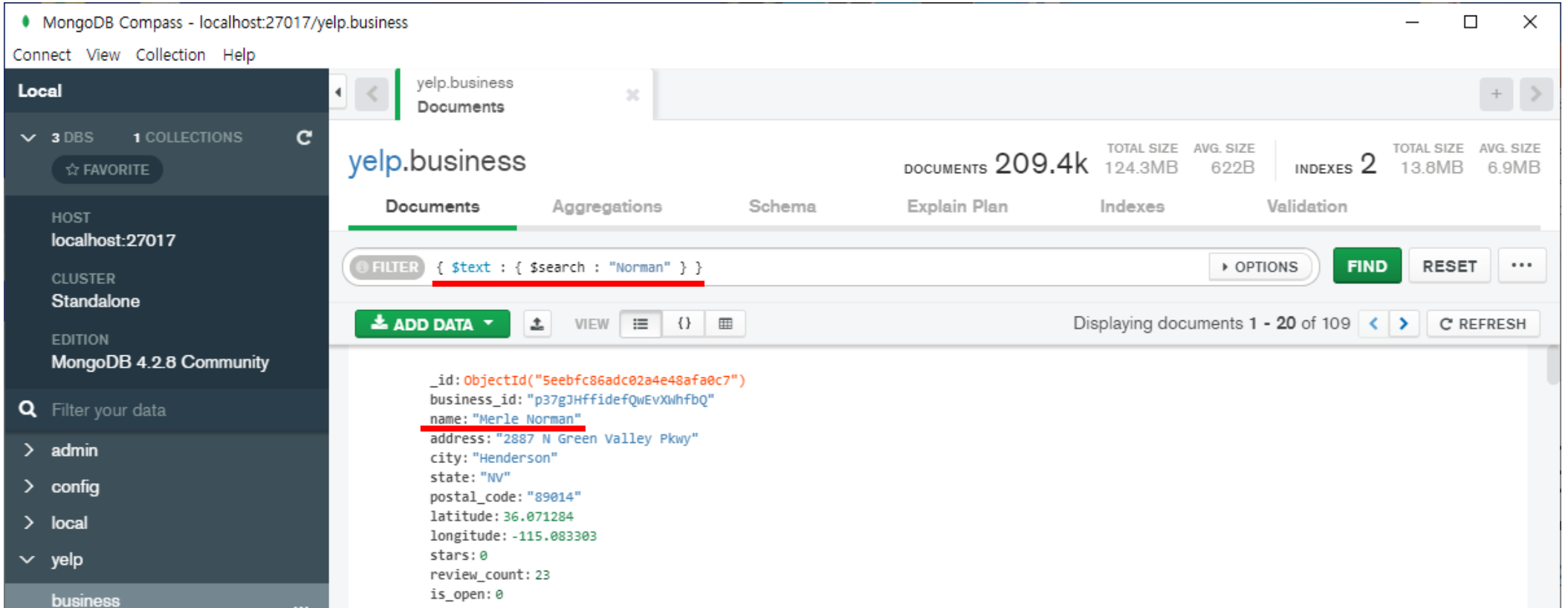
- Text search (https://docs.mongodb.com/manual/text-search/)
  - text index를 생성한다. db.business.createIndex({ name: "text", categories: "text" });



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" } });

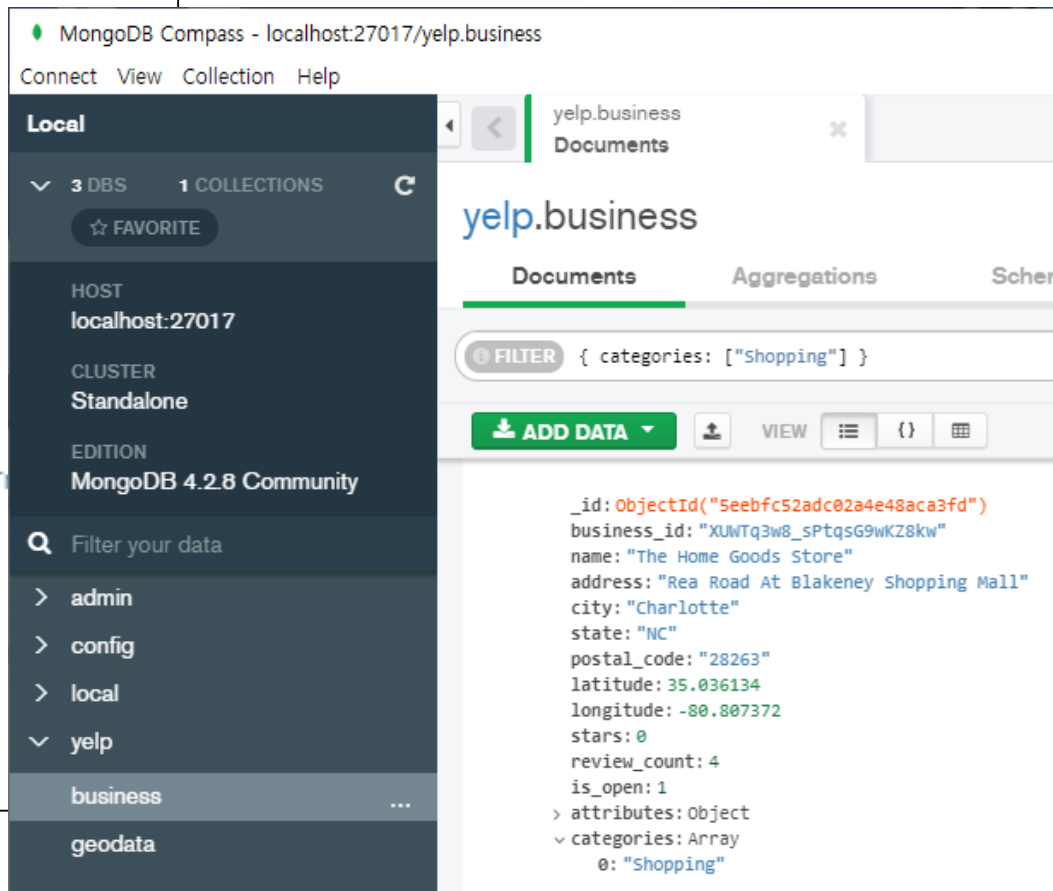


2. MongoDB 가이드

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

```
_id: ObjectId("5eebfc51adc02a4e48ac9b31")
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': T
   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"]** });

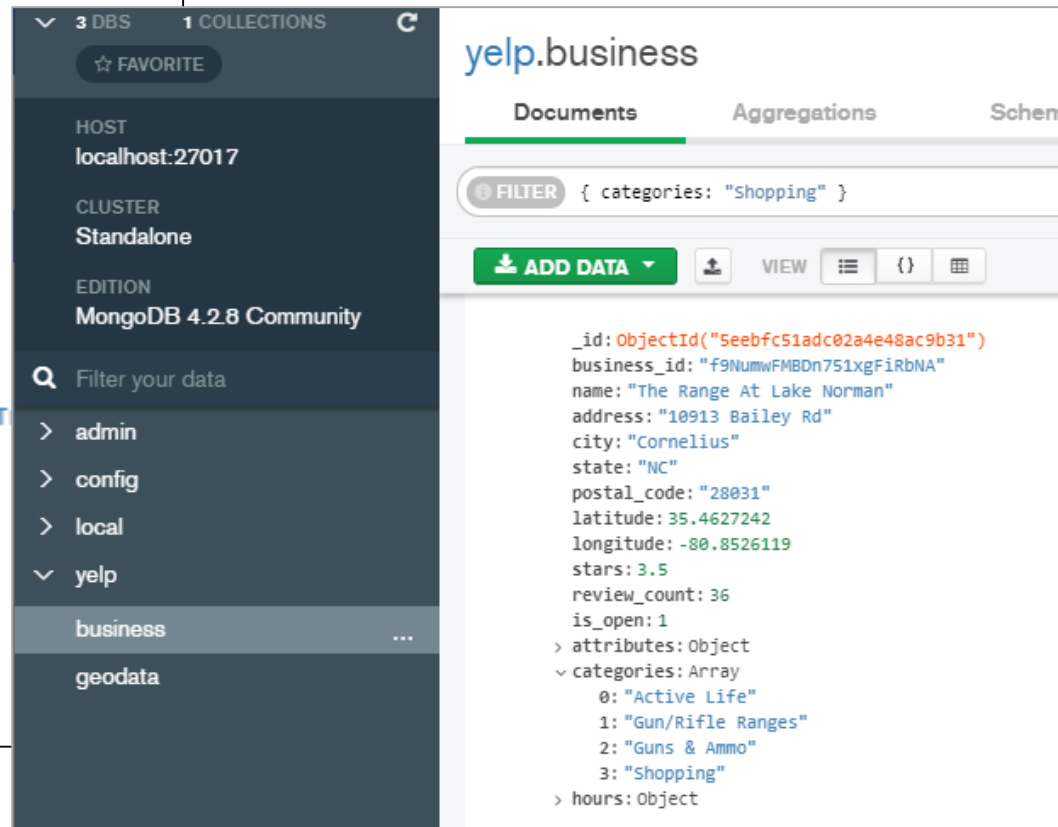


2. MongoDB 가이드

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

```
_id: ObjectId("5eebfc51adc02a4e48ac9b31")
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': T
    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" } );
```

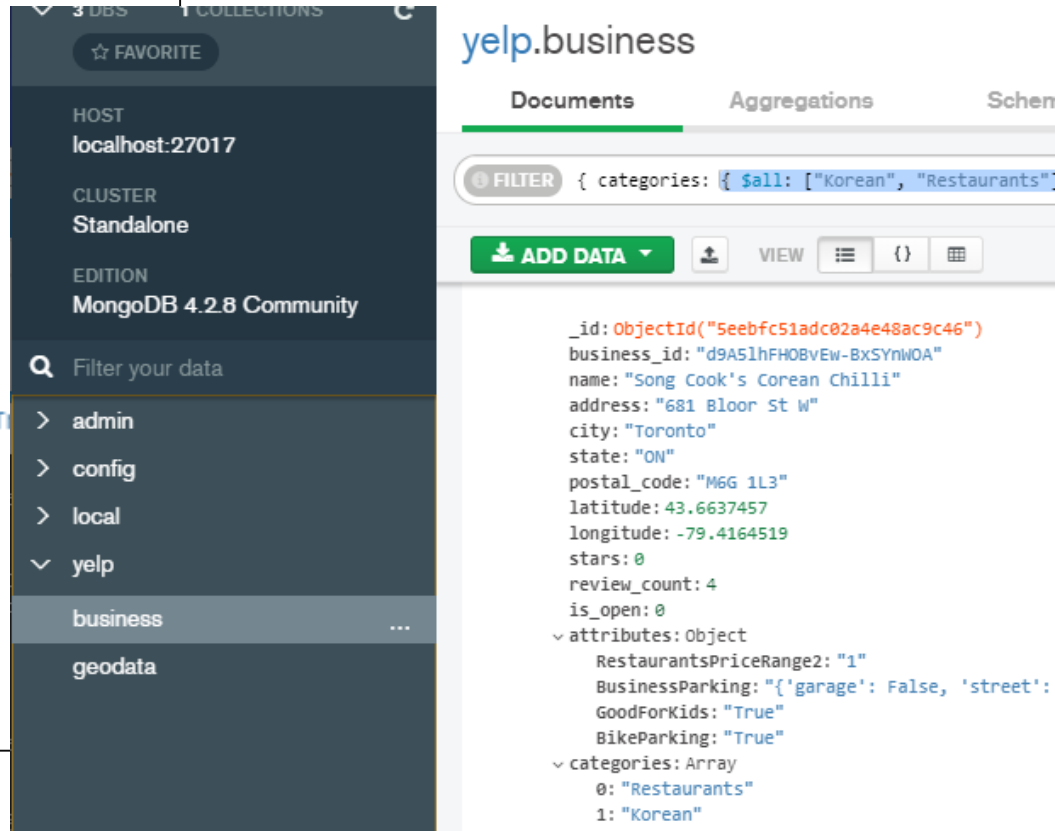


2. MongoDB 가이드

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

```
_id: ObjectId("5eebfc51adc02a4e48ac9b31")
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': T
    ByAppointmentOnly: "False"
    RestaurantsPriceRange2: "3"
categories: Array
    0: "Active Life"
    1: "Gun/Rifle Ranges"
    2: "Guns & Ammo"
    3: "Shopping"
hours: Object
```

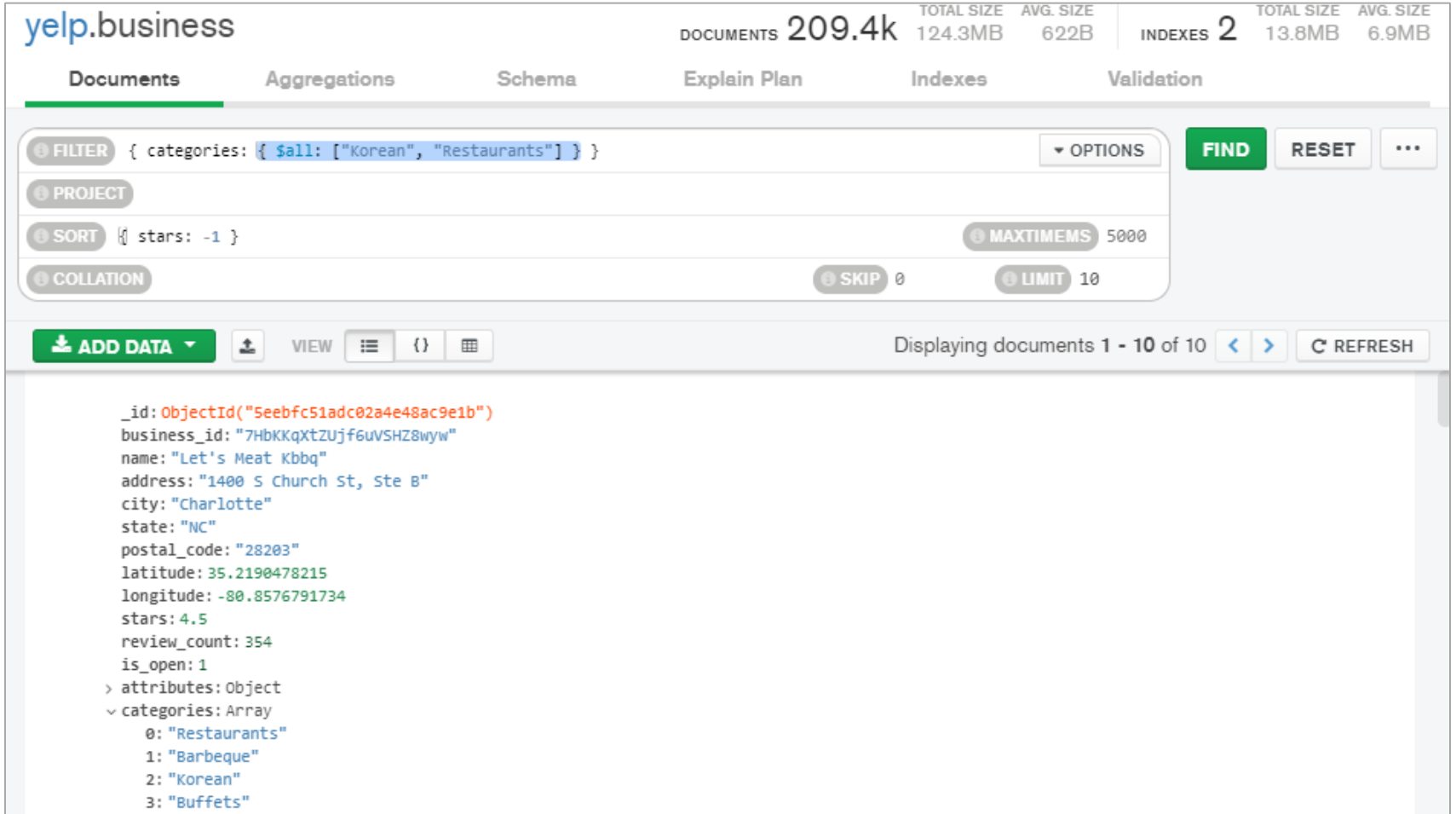
```
db.business.find({ categories: { $all: ["Korean", "Restaurants"] } });
```



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)**;



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]
}
```

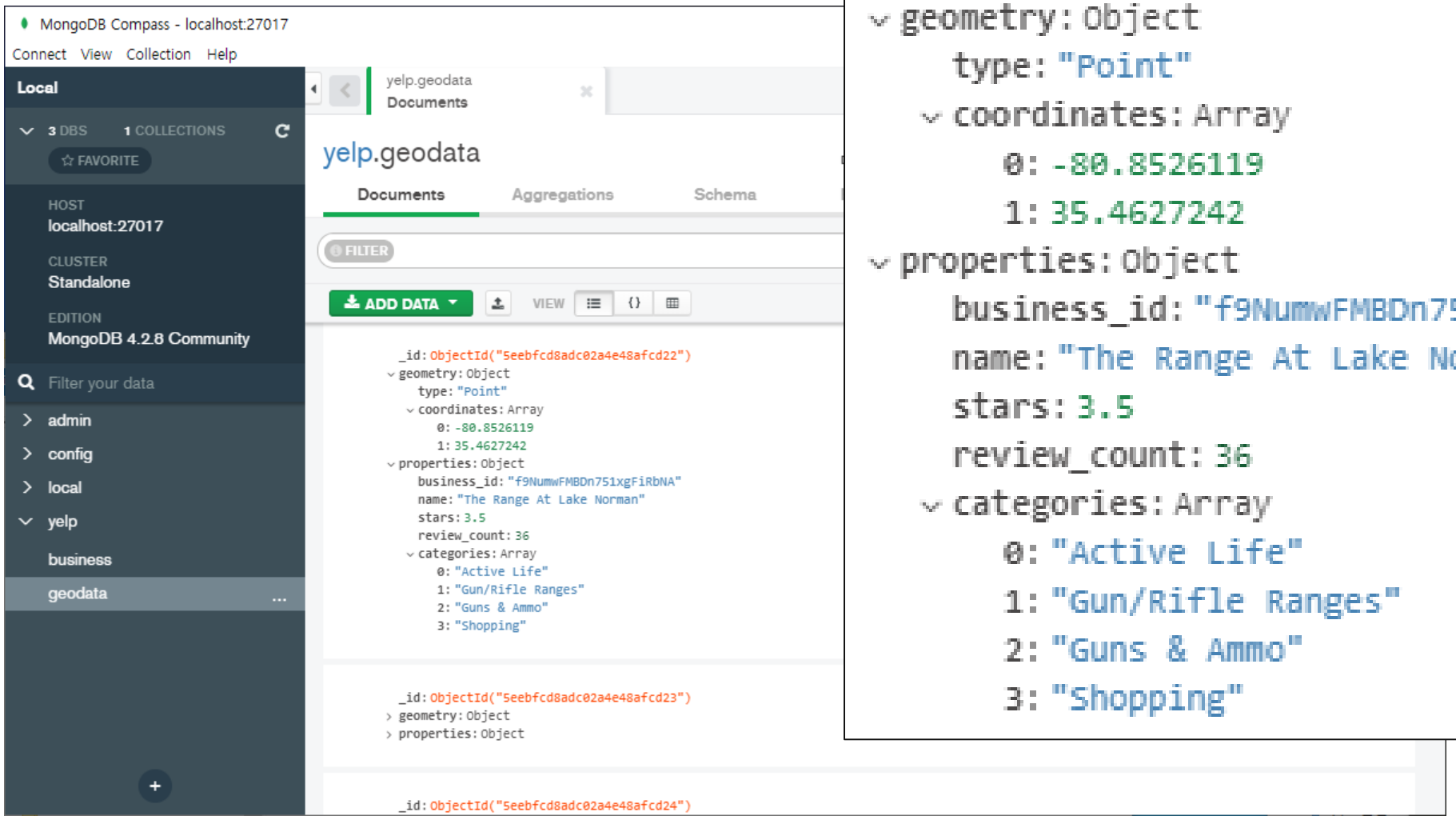
2. MongoDB 가이드

- 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

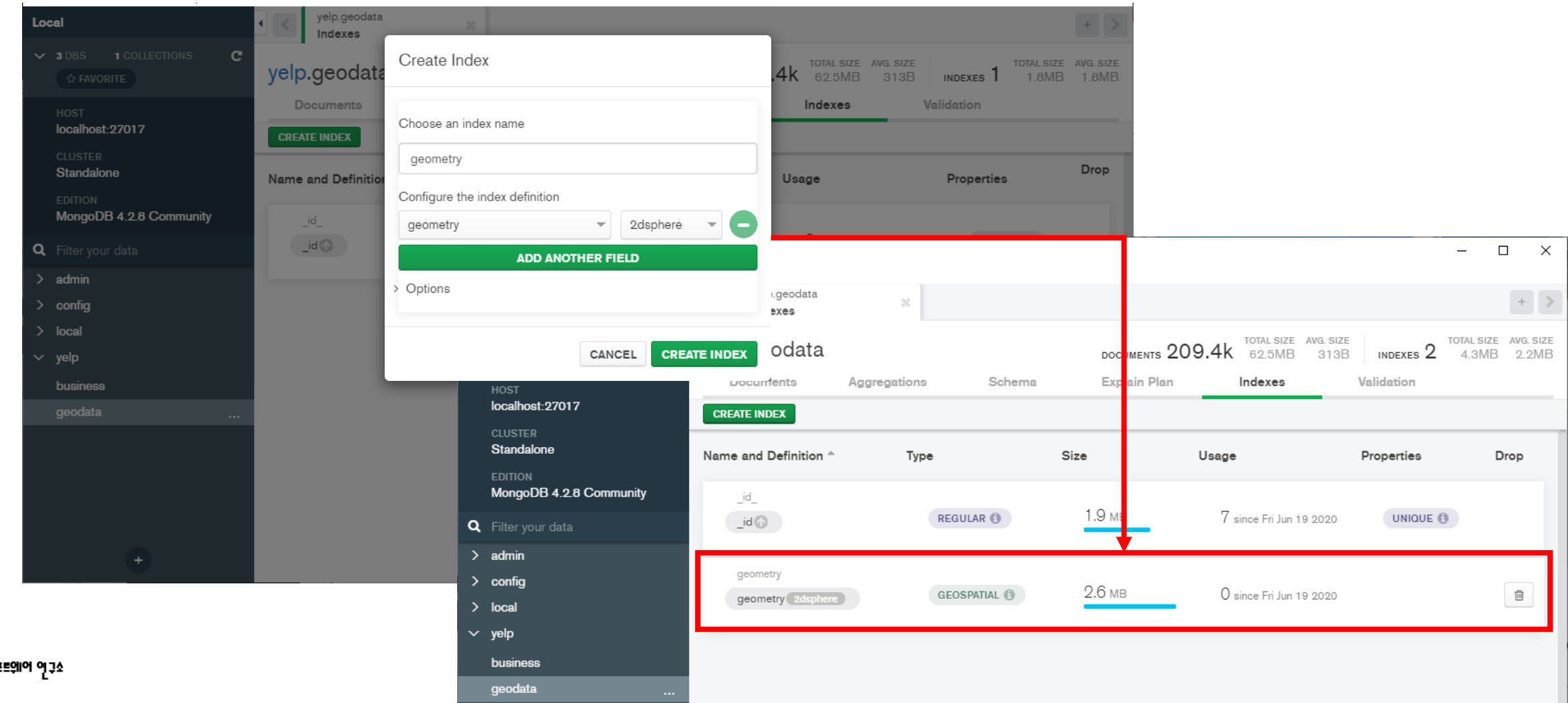
- geodata collection 만들기



_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

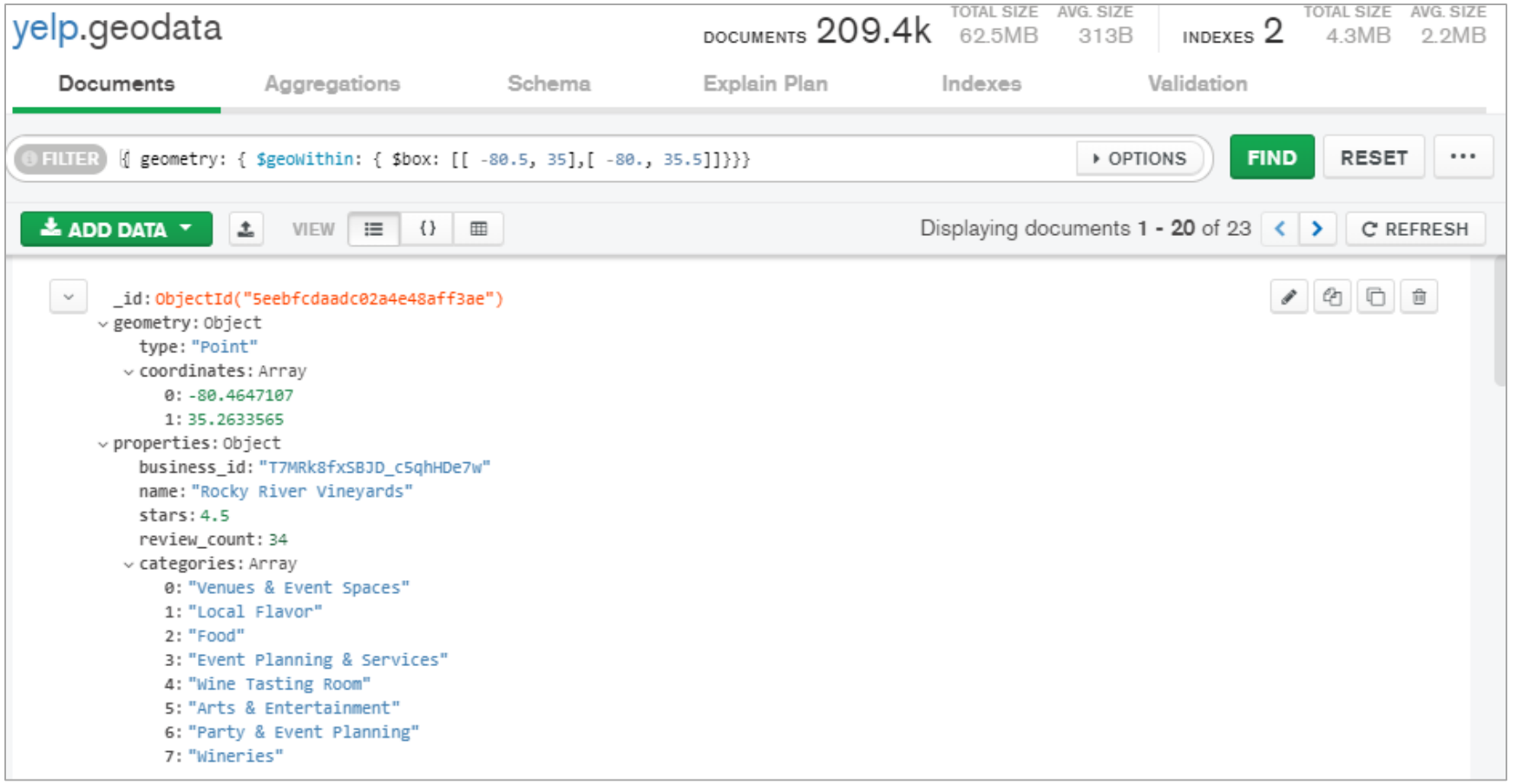
- 2dsphere 인덱스 생성하기



2. MongoDB 가이드

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

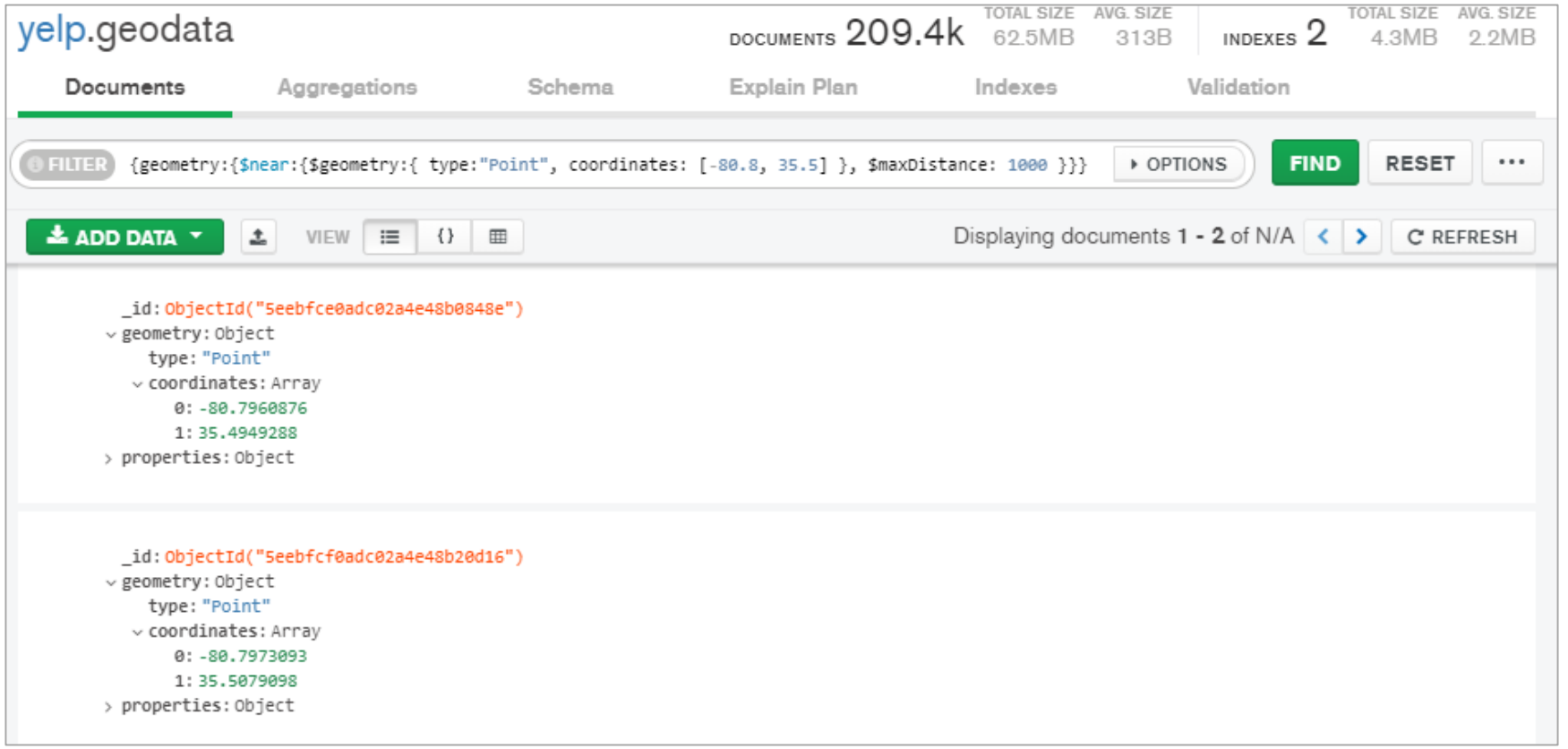
```
db.geodata.find({ geometry: { $geoWithin: { $box: [[ -80.5, 35],[ -80., 35.5]]}}});
```



2. MongoDB 가이드

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

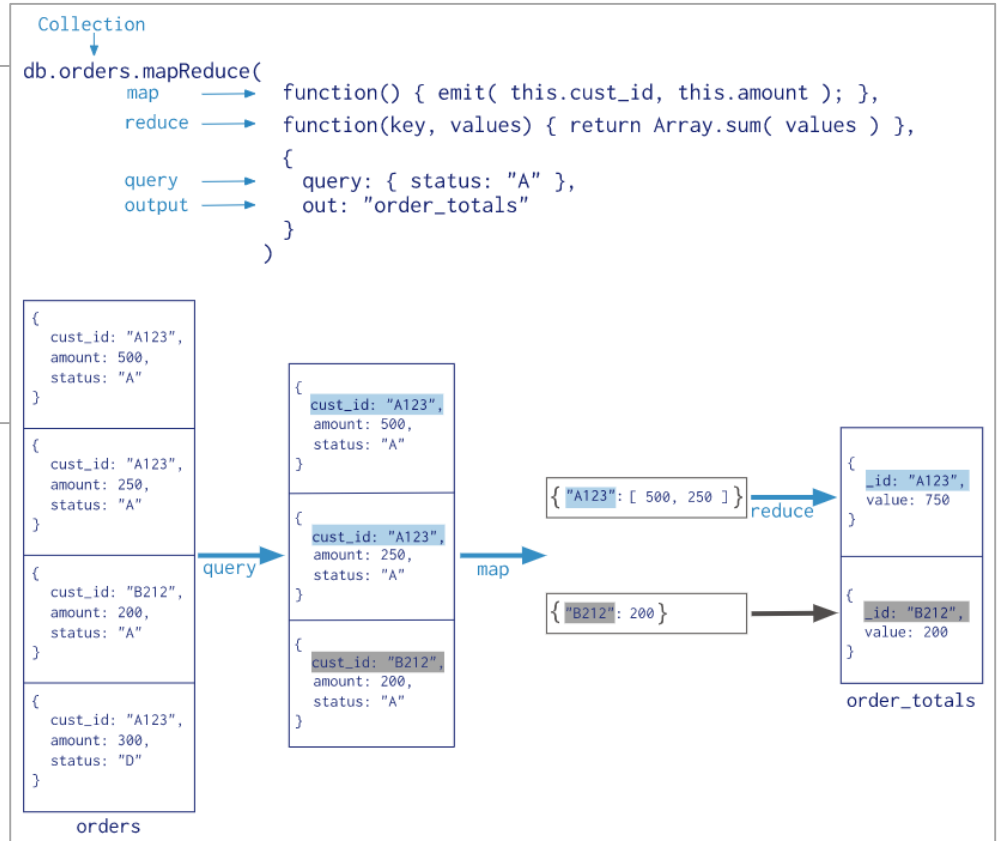
```
db.geodata.find({ geometry: { $near: { $geometry: { type: "Point", coordinates: [-80.8, 35.5] },
                                       $minDistance: 0, $maxDistance: 1000 }}});
```



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/)

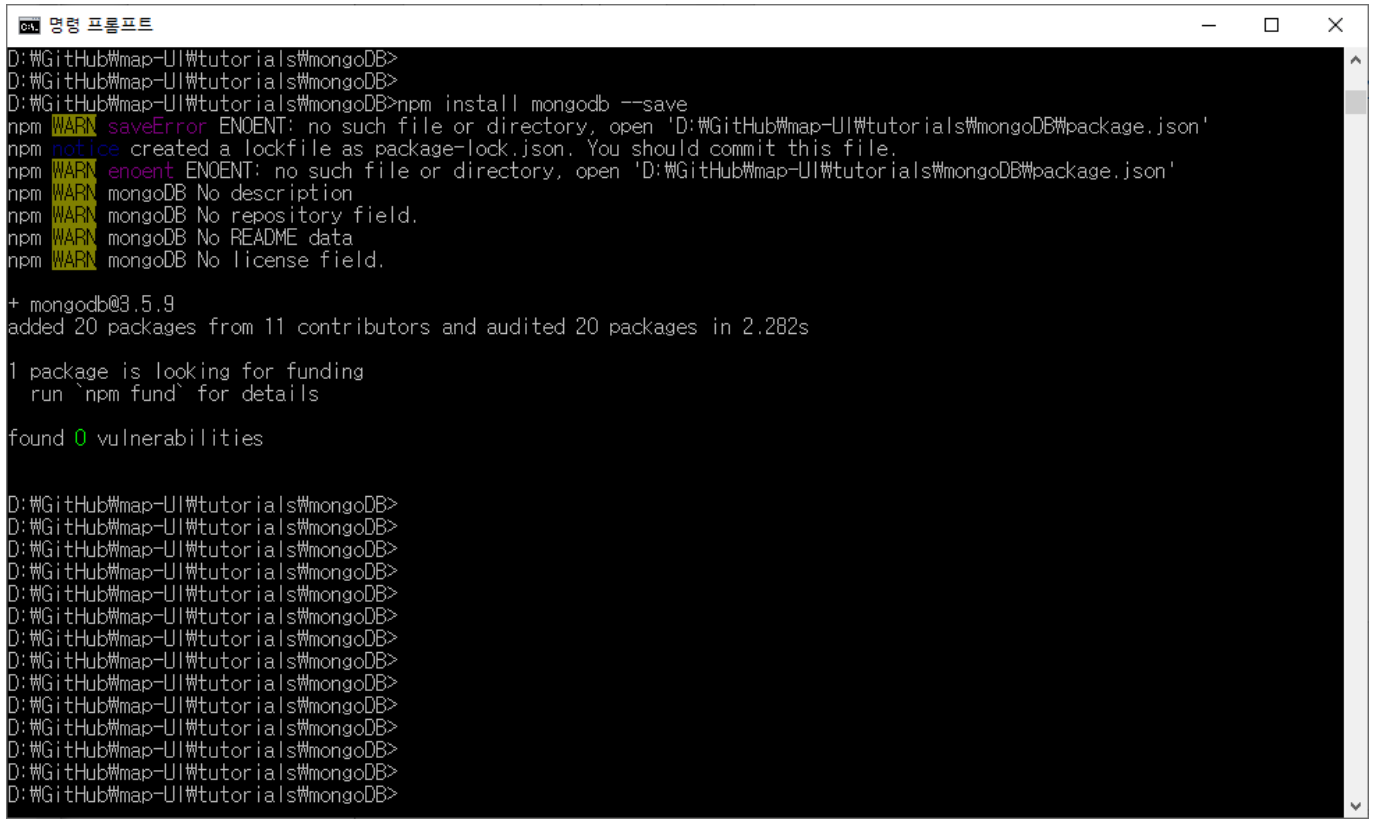
```
db.orders.aggregate([
    { $match: { status: "A" } },
    { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
```



3. MongoDB + Node.js 가이드

- MongoDB + Node.js 가이드

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



D:₩> 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

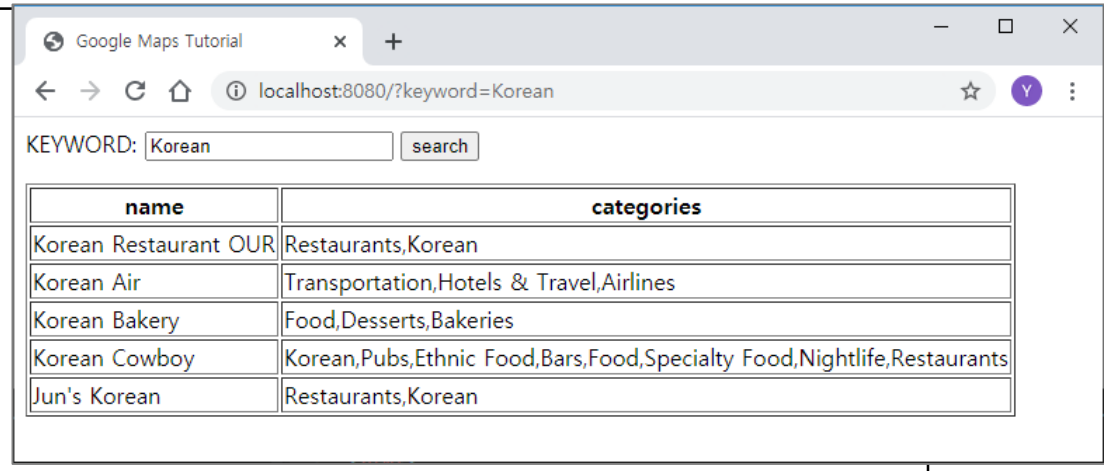
- Node.js web server(tutorials/mongoDB/db3.js)

```
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>
```



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

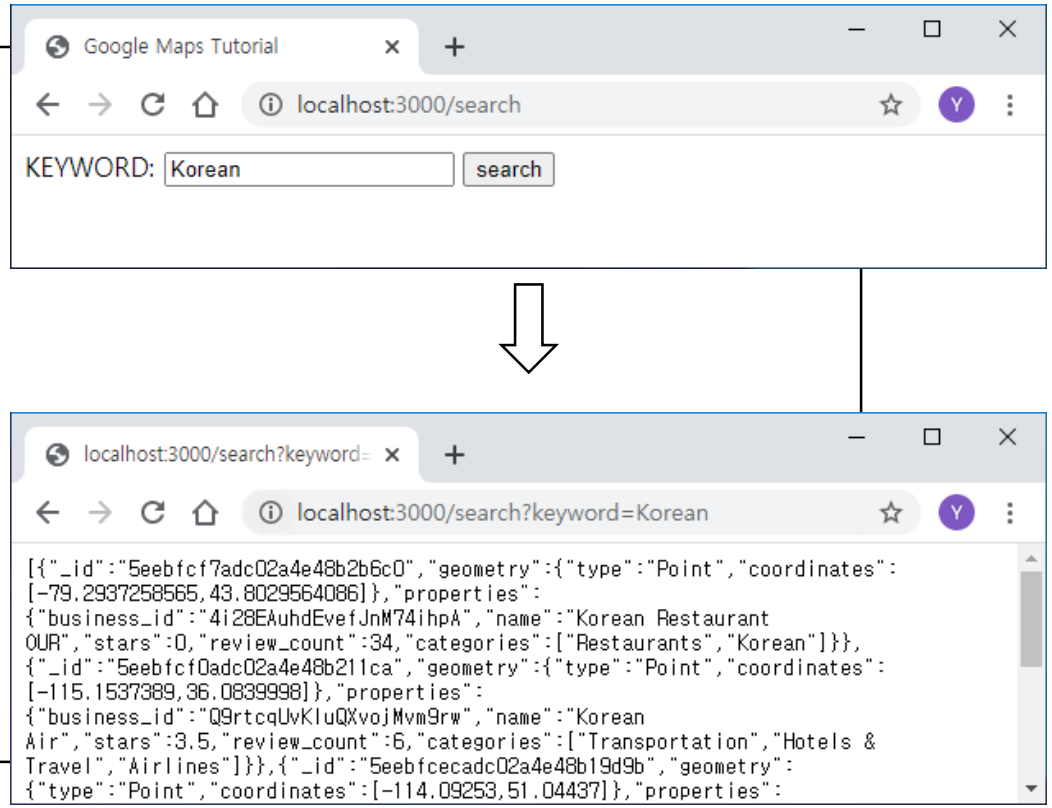
- Express.js : 빠르고 개방적인 간결한 웹 프레임워크 (https://expressjs.com)
  라우팅 (https://expressjs.com/ko/guide/routing.html)

```
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.findByName('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();
  })
})
```

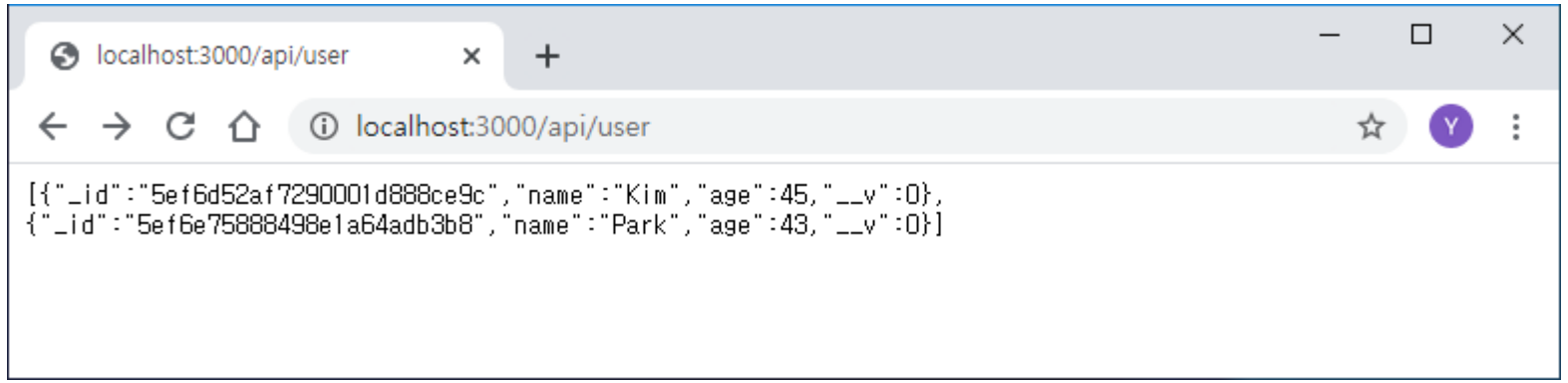
models/user.js

```
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);
```

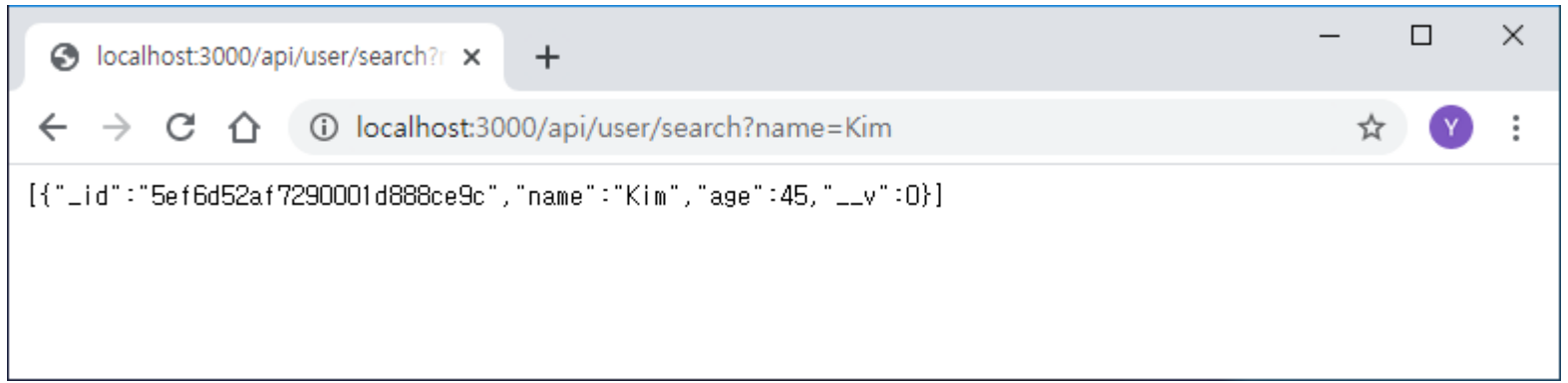
4. Express.js + Mongoose

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

GET: /api/user



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



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

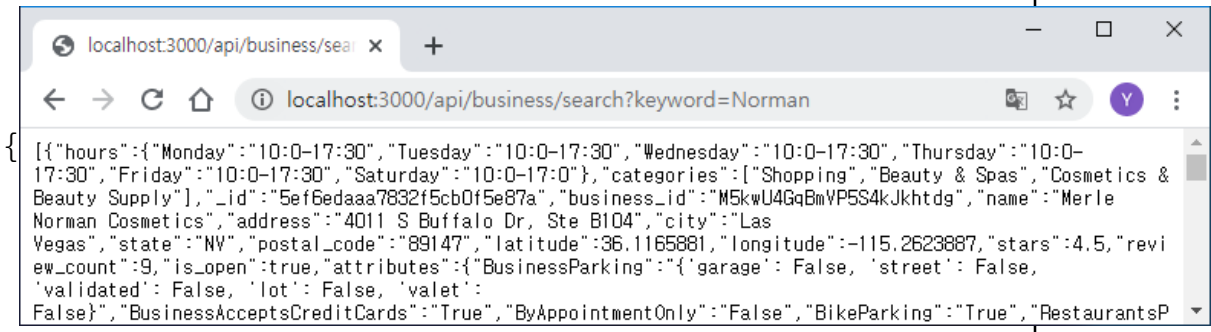
- Yelp Business(tutorials/express/models/business.js)

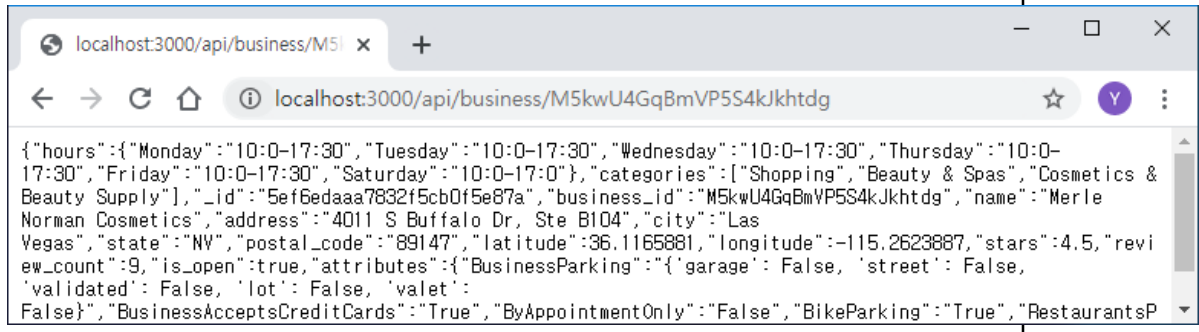
```
const mongoose = require('mongoose' );

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;
```





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

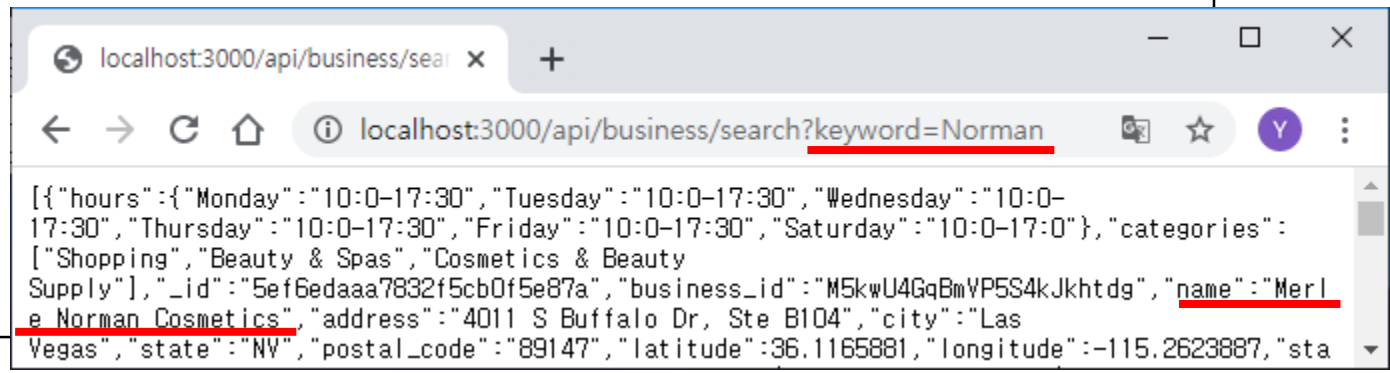
- Yelp Business(tutorials/express/api/geodata.js)

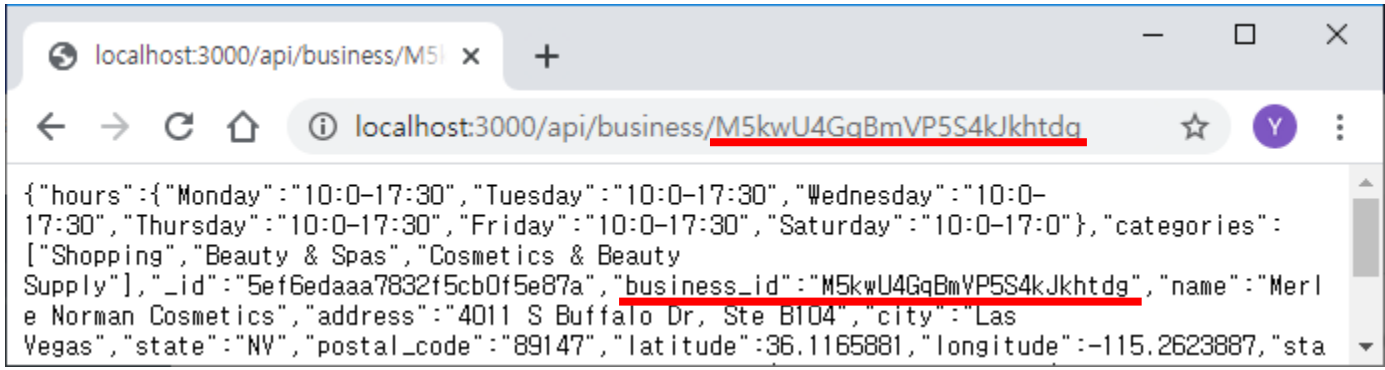
```
router.get('/search', function(request, response) {
  console.log(request.query.keyword);
  geodata.findByKeyword(request.query.keyword)
          .sort({stars: -1}).limit(10).exec(function(error, result){
    if (error) return response.status(500).json(error);
    return response.json(result);
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

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);
```

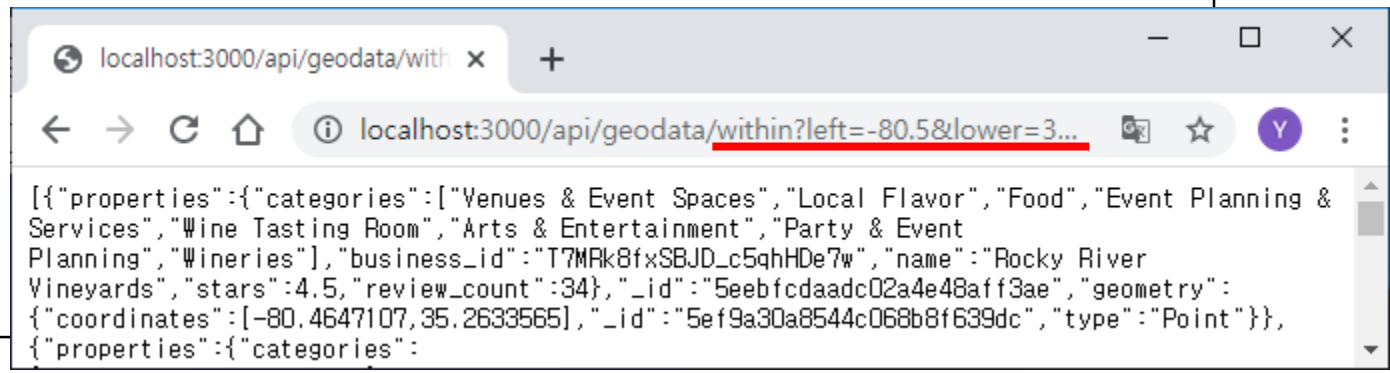


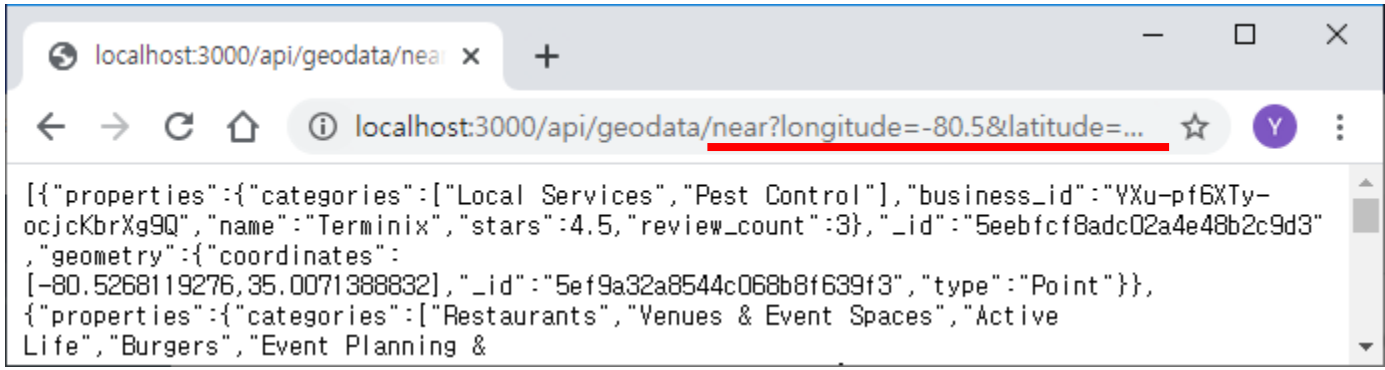


4. Express.js + Mongoose

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

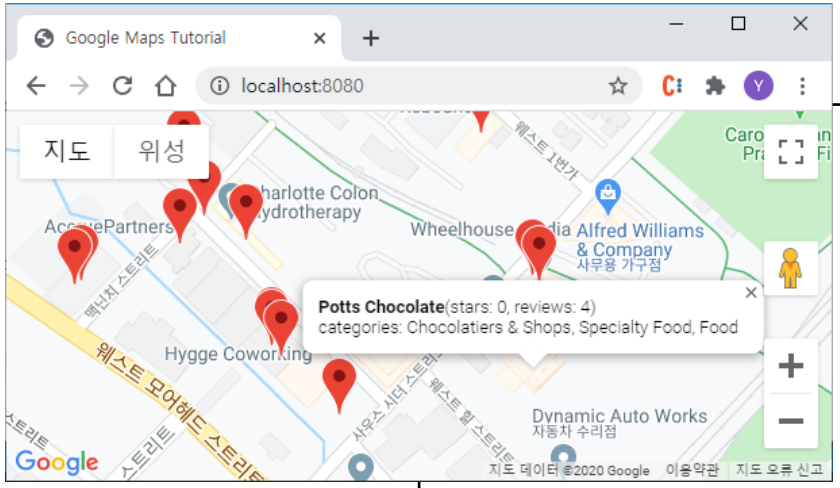
```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/yelp',
            {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true});
mongoose.set('debug', true);

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)



http://localhost:8080

index.html

http_server

http://localhost:3000/api/geodata/within?left=...

XMLHttpRequest
(Ajax, Asynchronous JavaScript and XML)

GeoJson data

<<express>>
node db6.js

MongoDB

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

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);
```

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

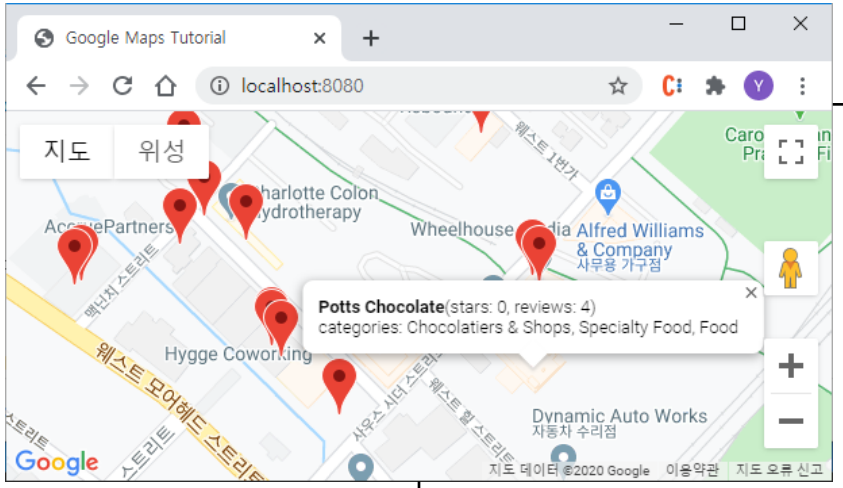
- Integrating Maps with Mongo DB(tutorials/express/index.html)

```
    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.



http://localhost:8080

index.html

http_server

http://localhost:3000/api/geodata/within?left=...

GeoJson data

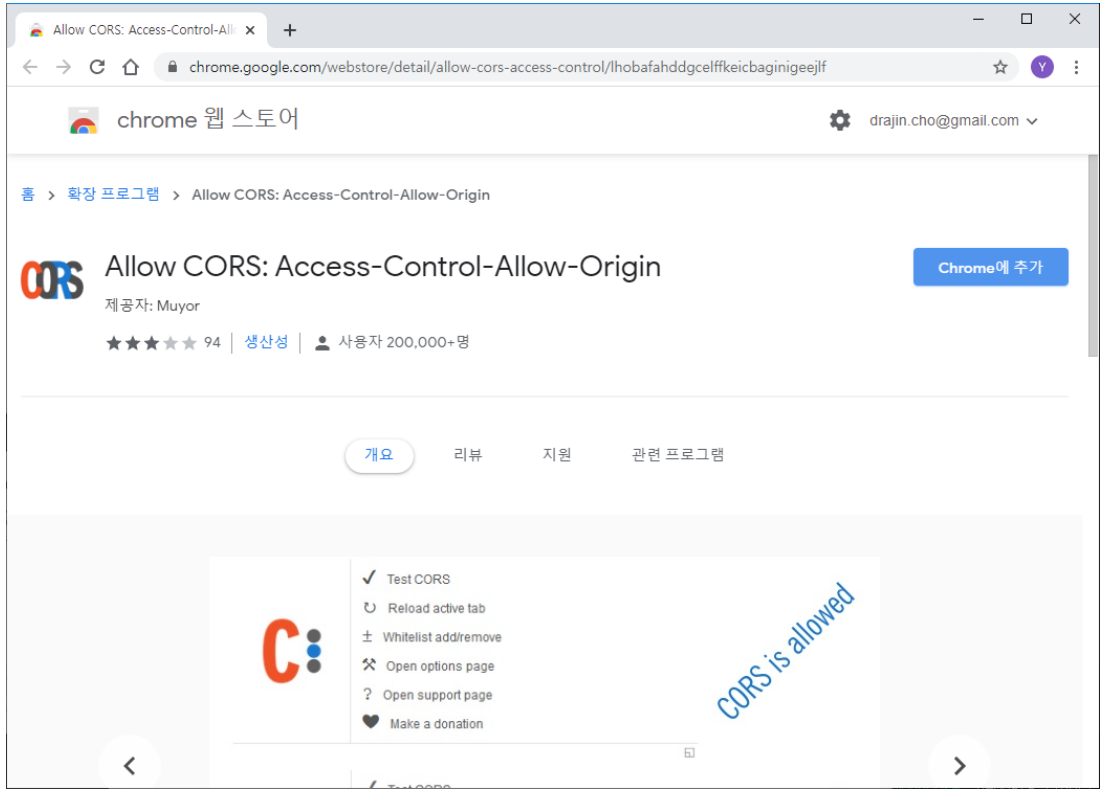
<<express>>
node db6.js

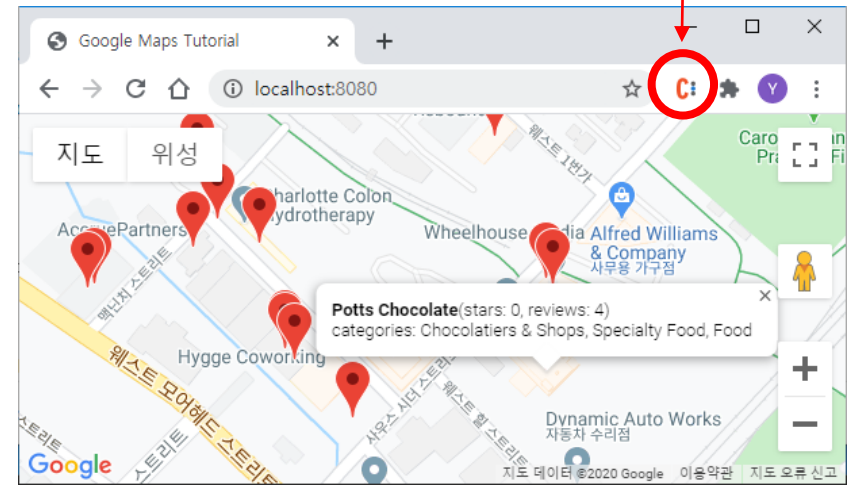
MongoDB

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

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

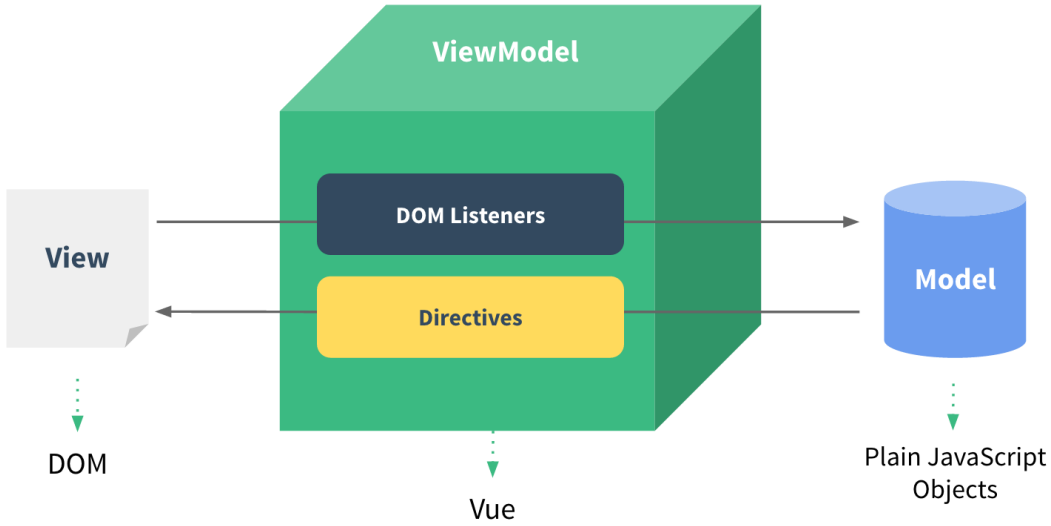


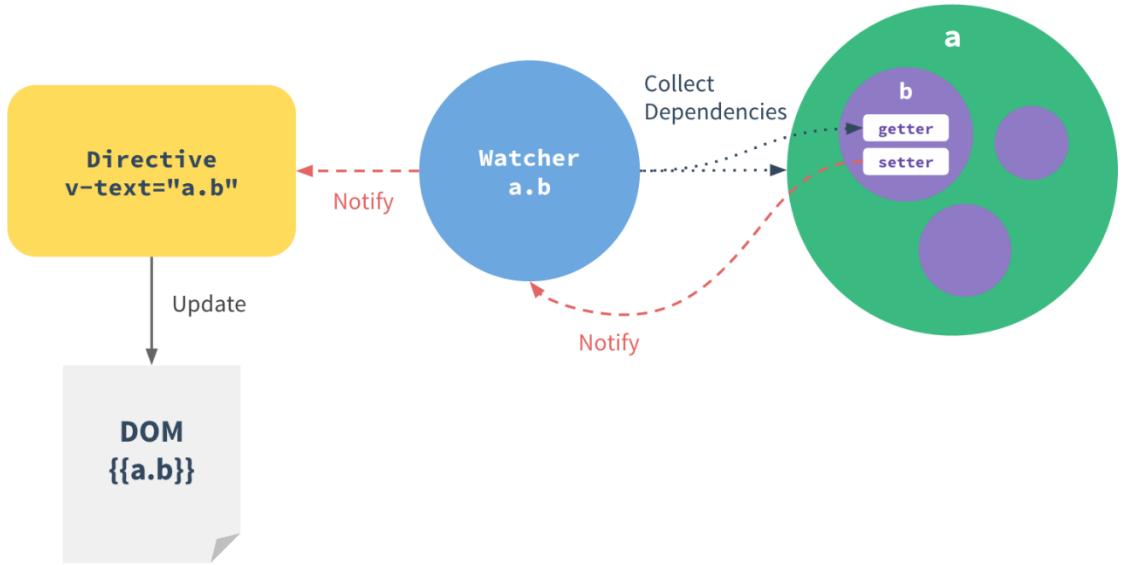


https://ko.wikipedia.org/wiki/교차_출처_리소스_공유

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/

프로그레시브
JavaScript 프레임워크
https://vuejs.org/

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

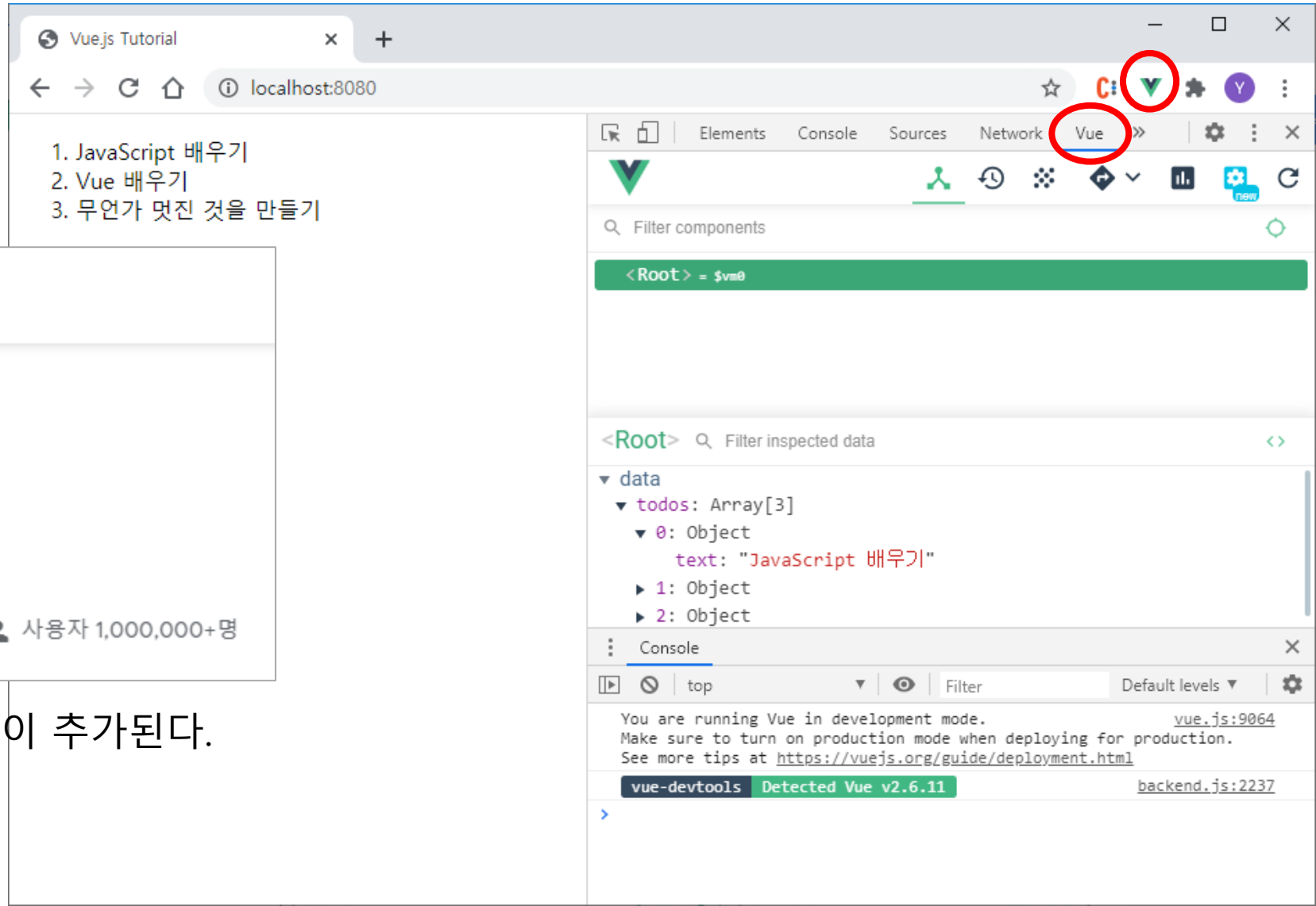
- Hello World (tutorials/vue/index2.html)

```
<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



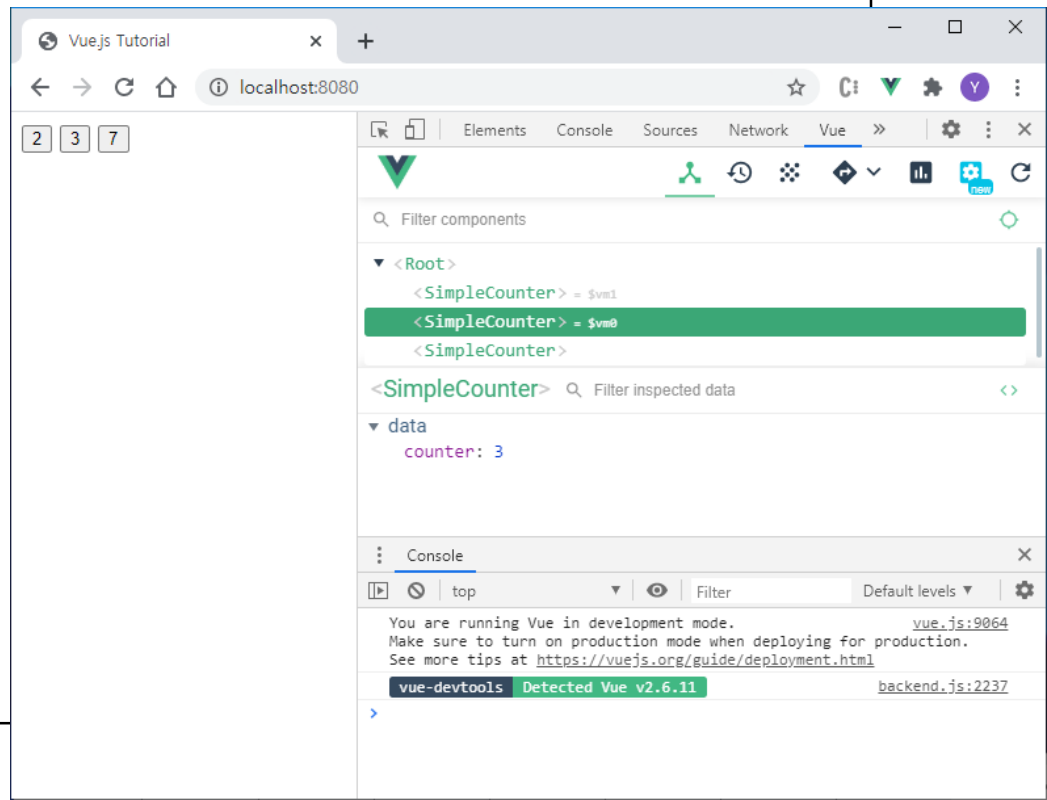
Chrome 개발자 도구에 Vue 탭이 추가된다.

5. Vue.js

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

- Vue Component (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>
```



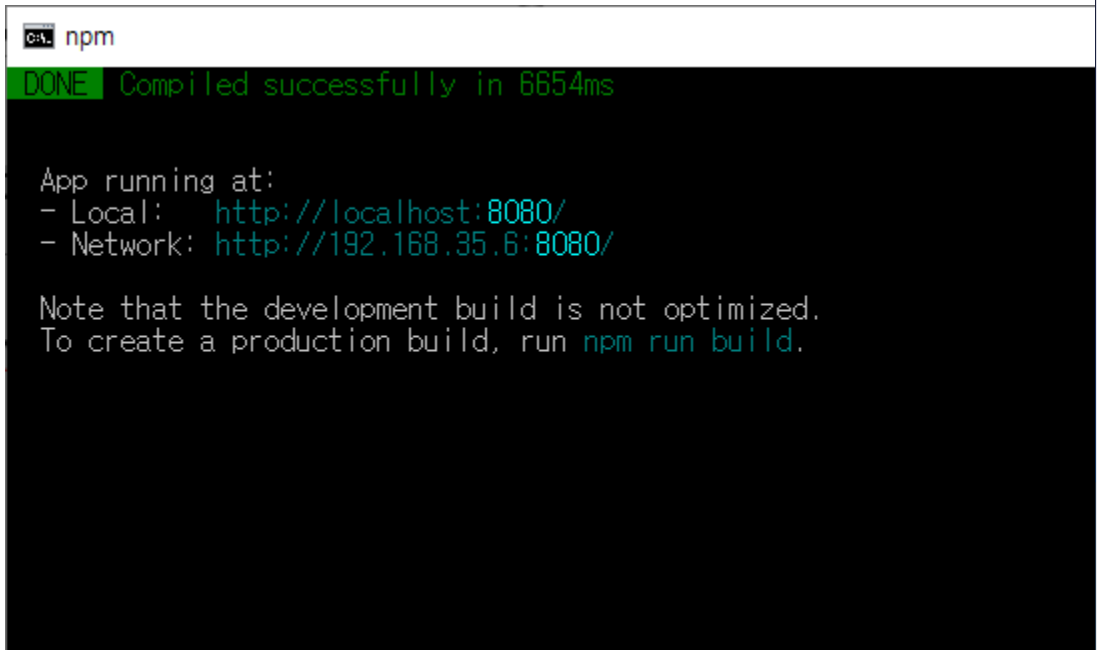
프로그레시브 JavaScript 프레임워크
https://vuejs.org/

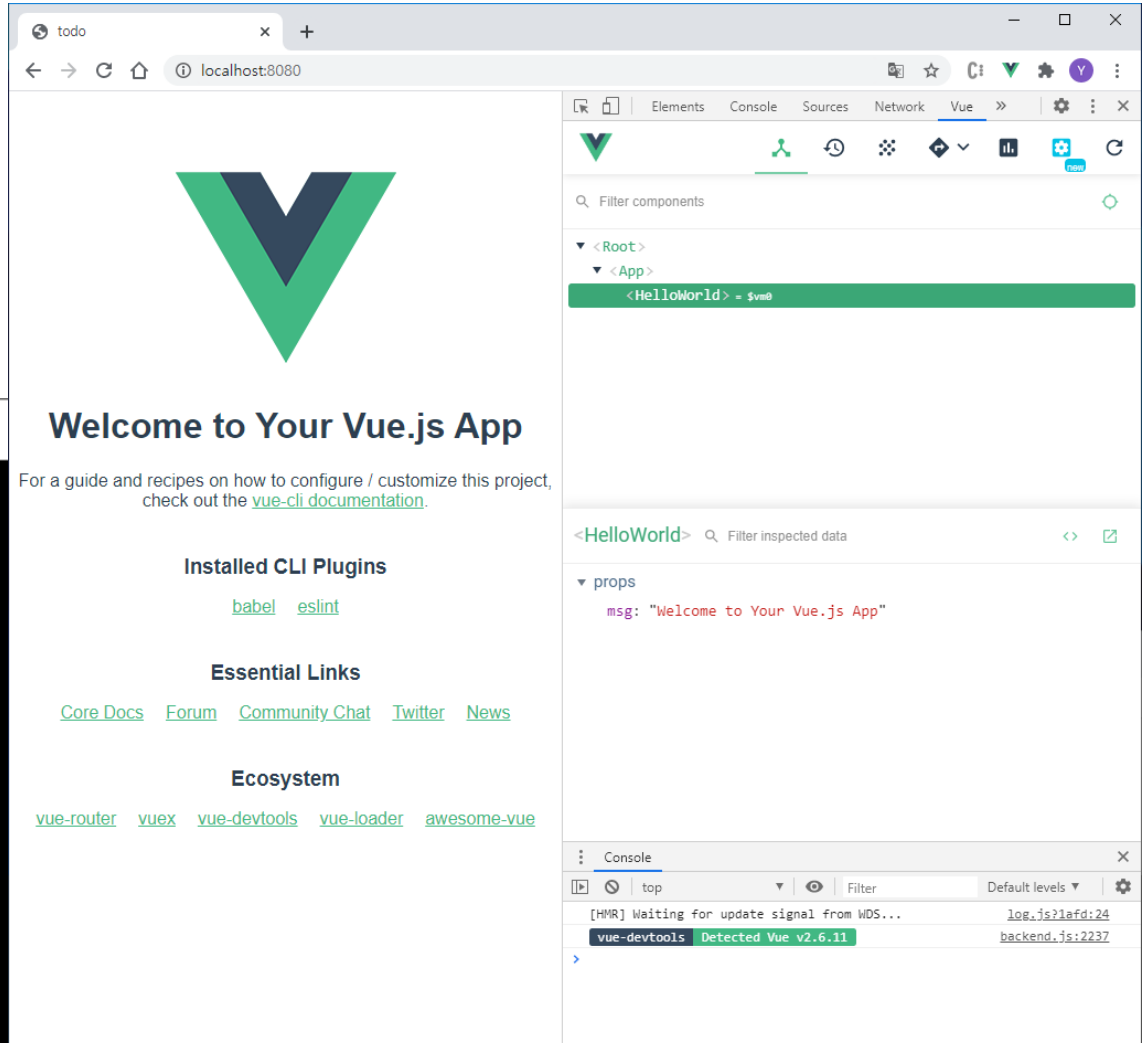
5. Vue.js

- Vue CLI – standard tool for Vue.js Development

https://cli.vuejs.org/

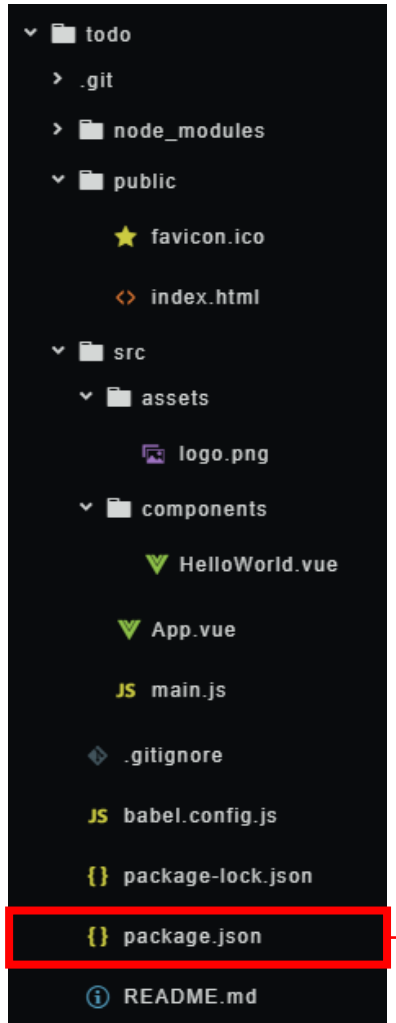
```
C:₩> npm install @vue/cli –g
C:₩> vue create todo
C:₩> cd todo
C:₩todo> npm run serve
```





5. Vue.js

- Vue CLI – standard tool for Vue.js Development https://cli.vuejs.org/

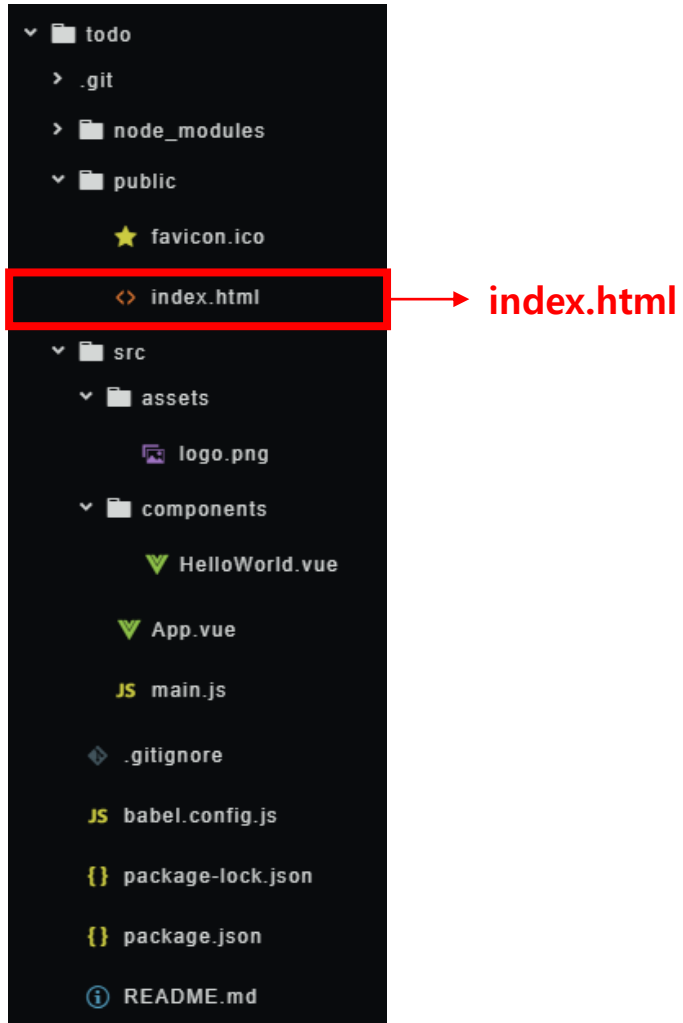


→ Package에 대한 정의

```
{
  "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/

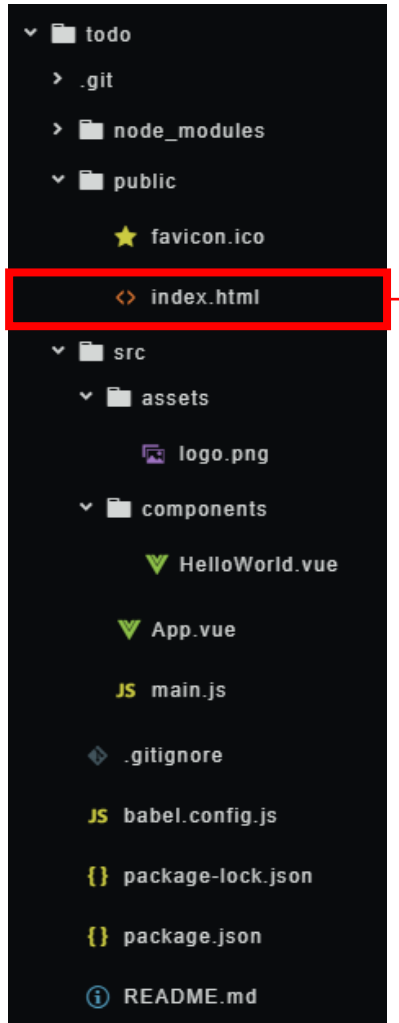


index.html

```
<!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="<%= 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/



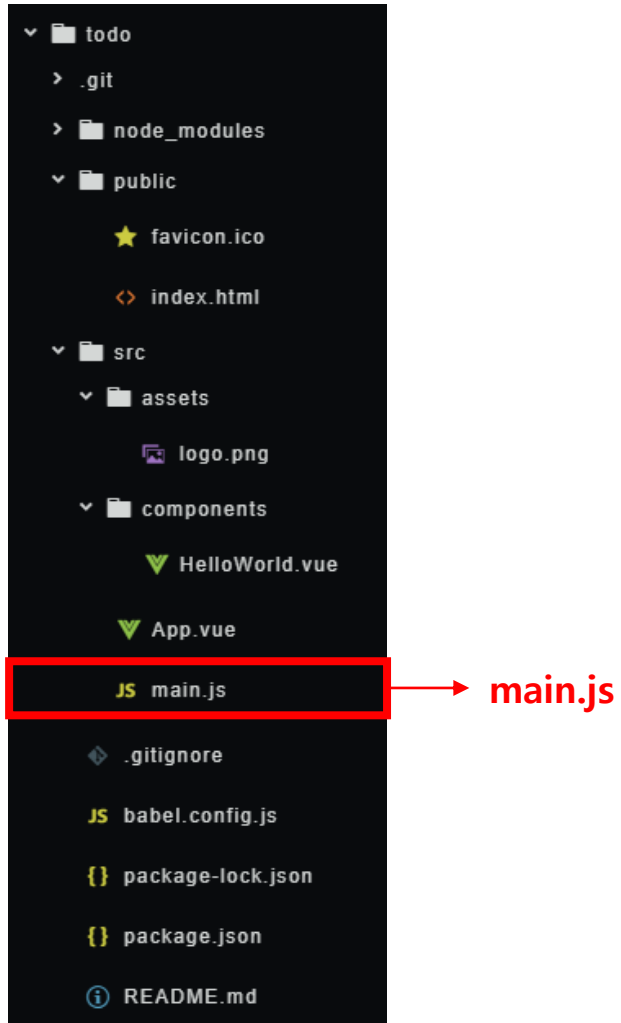
index.html

```
<!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="<%= 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/

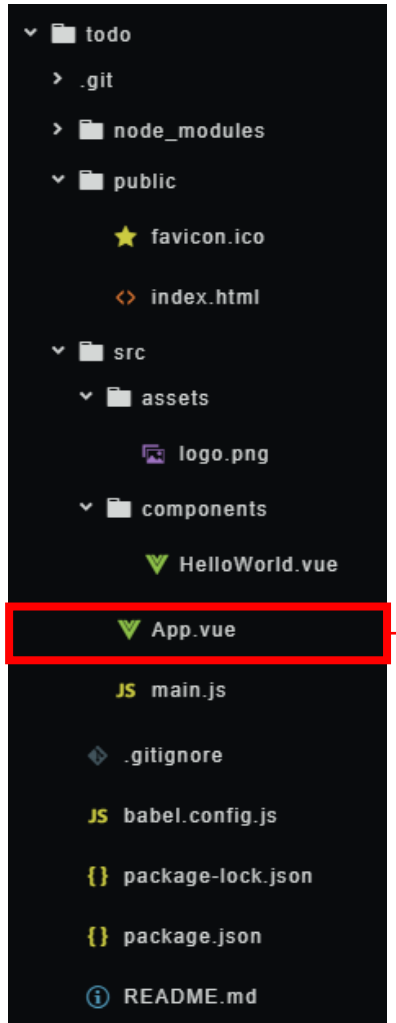


main.js

```
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/

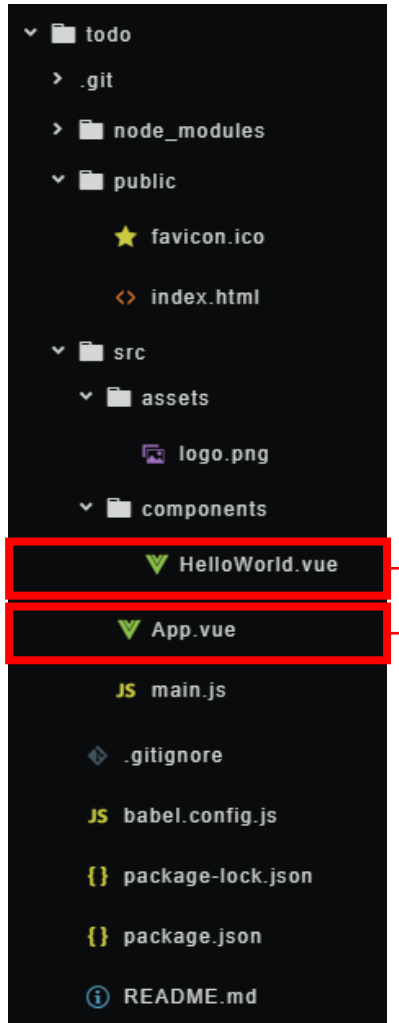


App.vue(main component)

```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <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">
    <img alt="Vue logo" src="./assets/logo.png">
    <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, Helv
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

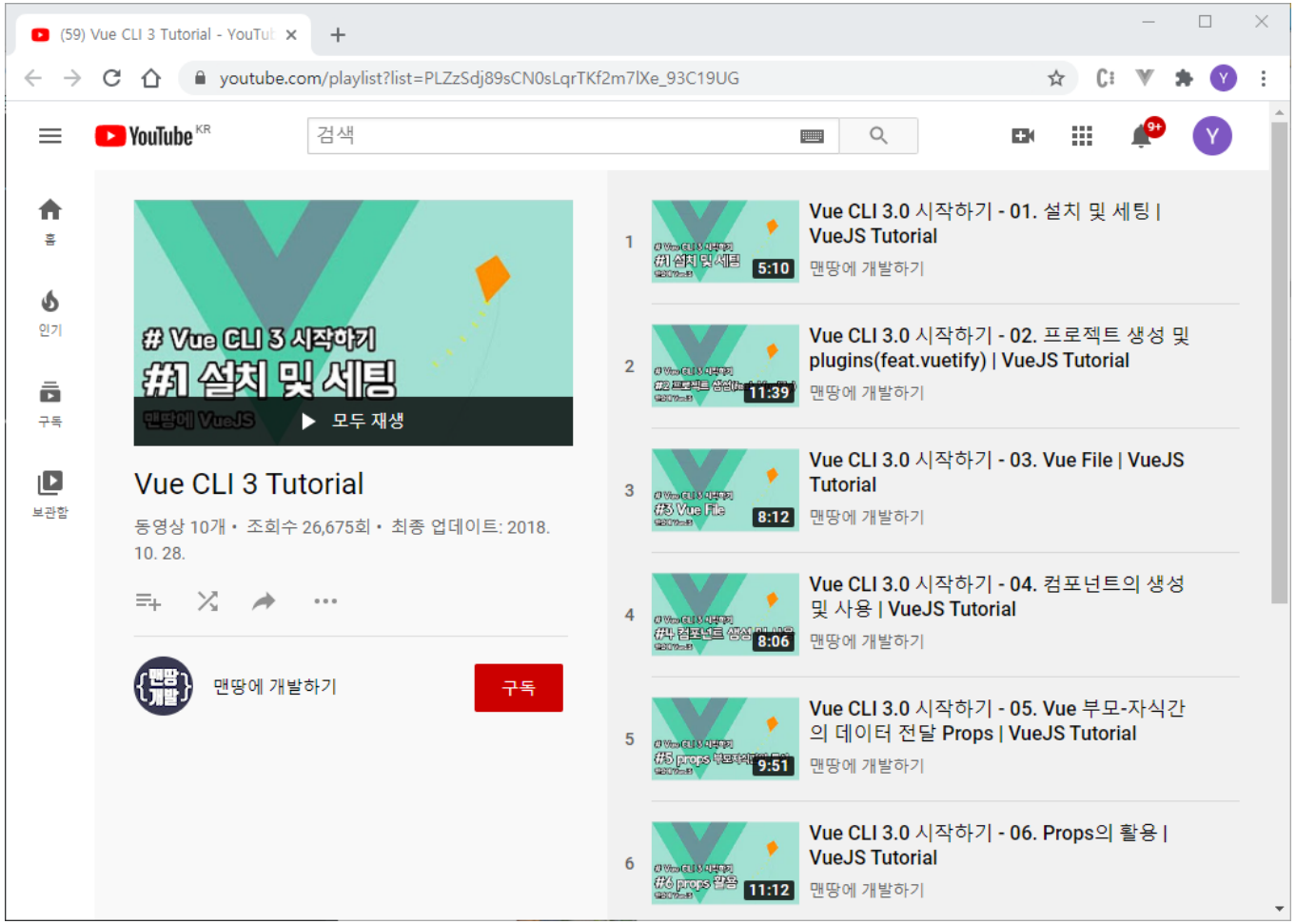
```
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
  </div>
</template>

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

5. Vue.js

- Vue CLI – standard tool for Vue.js Development

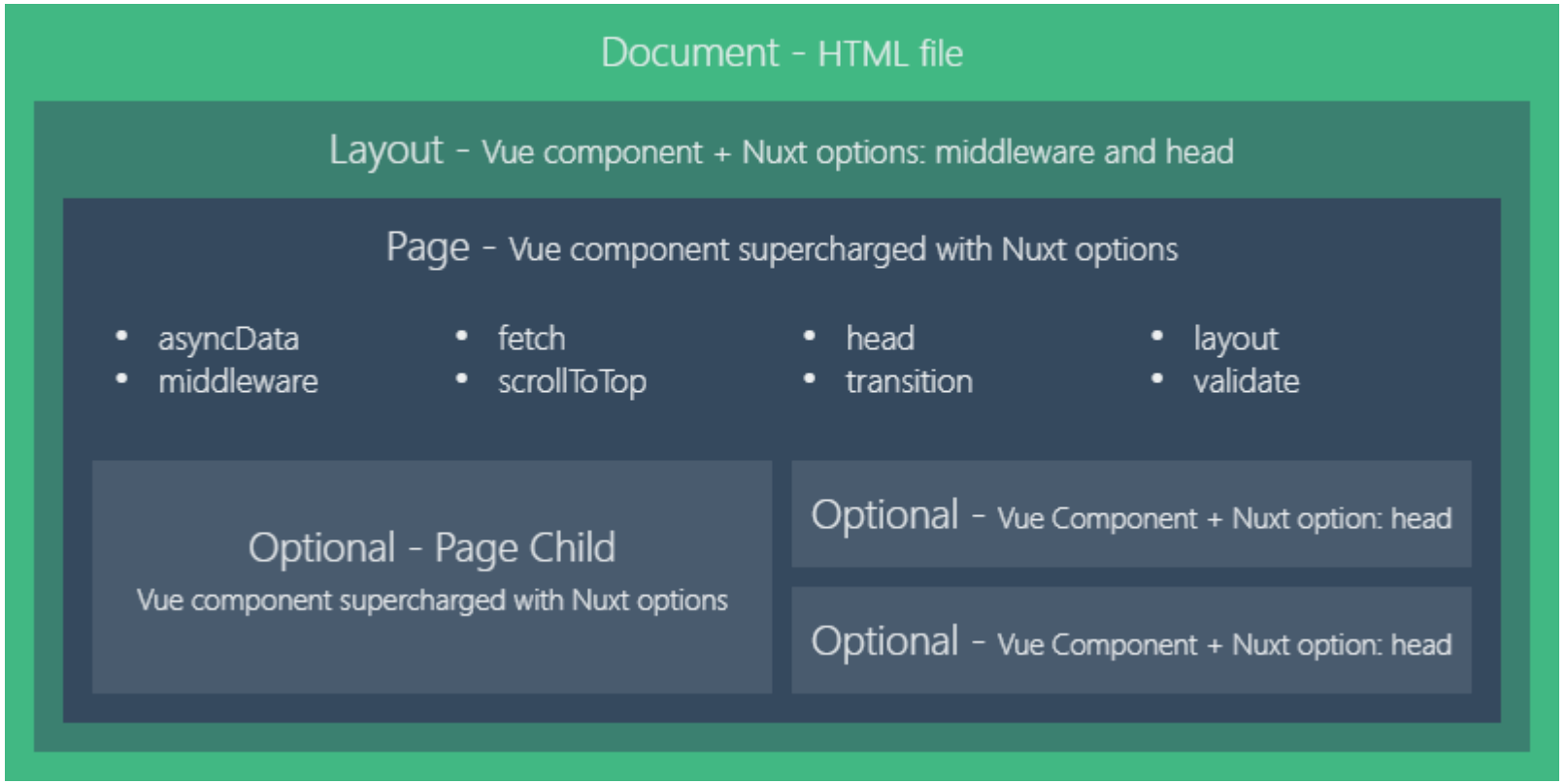
https://cli.vuejs.org/

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



6. Vue.js / Nuxt.js

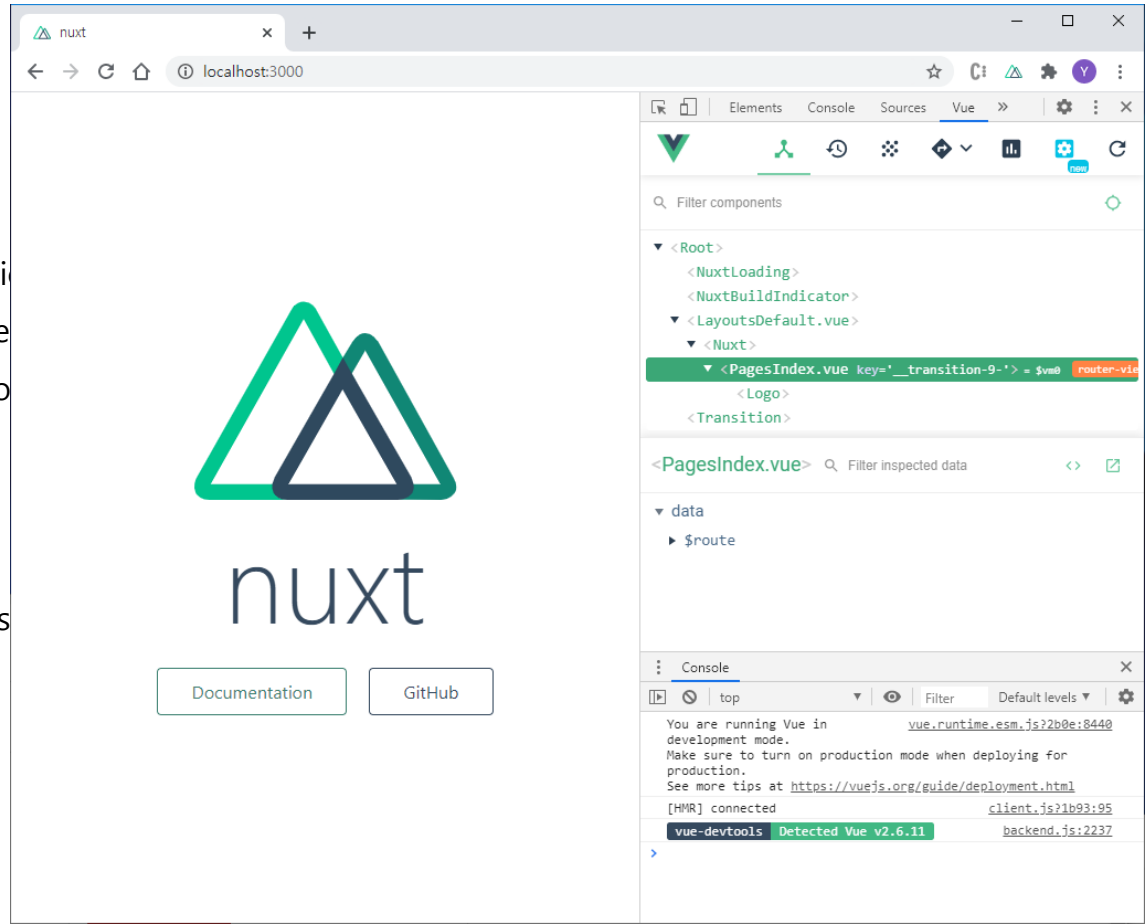
- NUXTJS (https://nuxtjs.org/) – The Intuitive Vue Framework



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 selecti
? Nuxt.js modules: (Press <space> to select, <a> to toggle all, <i> to invert sele
? Linting tools: (Press <space> to select, <a> to toggle all, <i> to invert selectio
? 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 s

C:₩>cd nuxt
C:₩nuxt> npm run dev
```



6. Vue.js / Nuxt.js

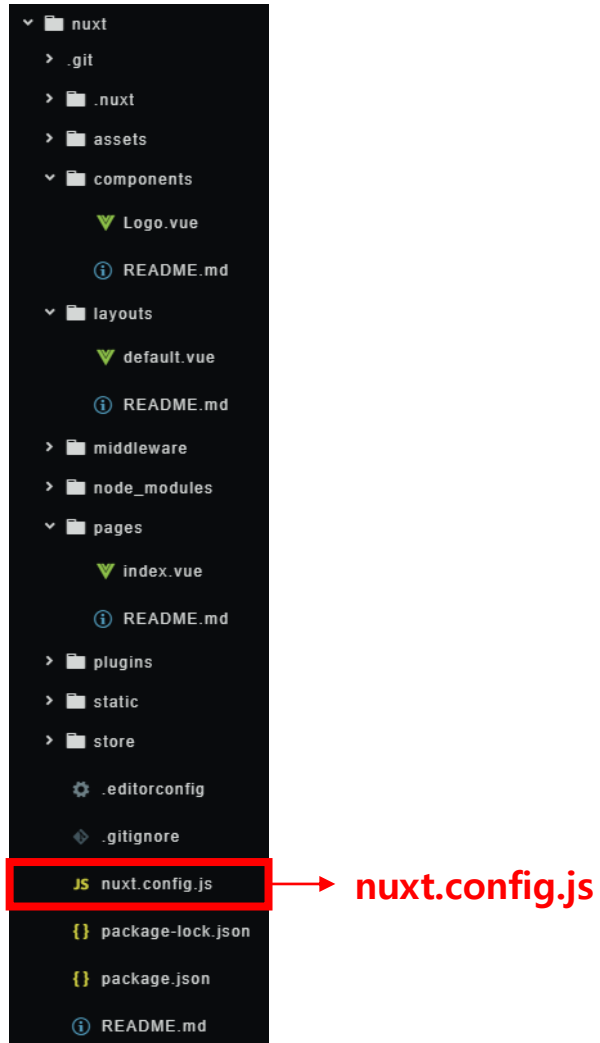
- NUXTJS (https://nuxtjs.org/) – The Intuitive Vue Framework

package.json

```
{
  "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



nuxt.config.js

```
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

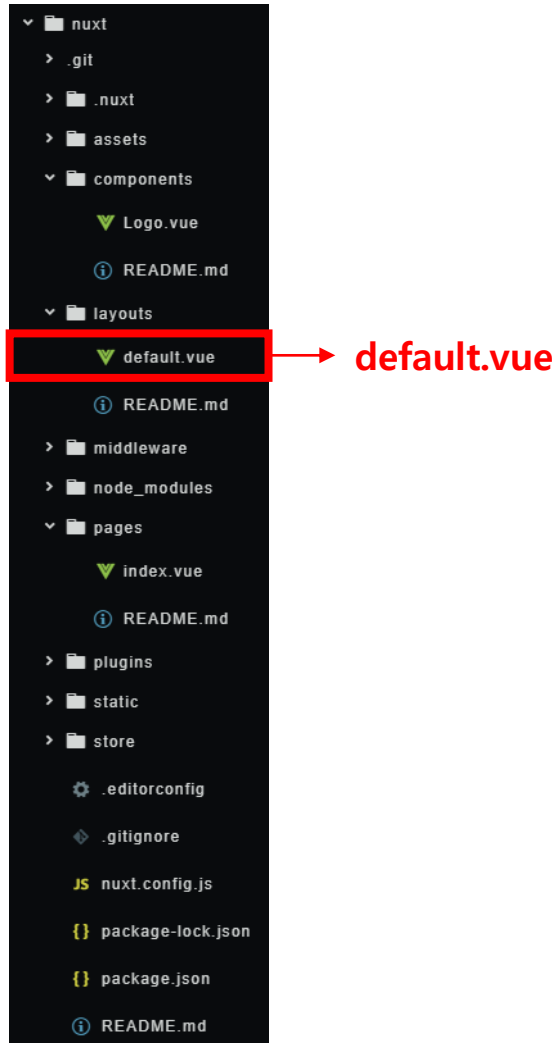
nuxt.config.js

```
/*
 ** 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



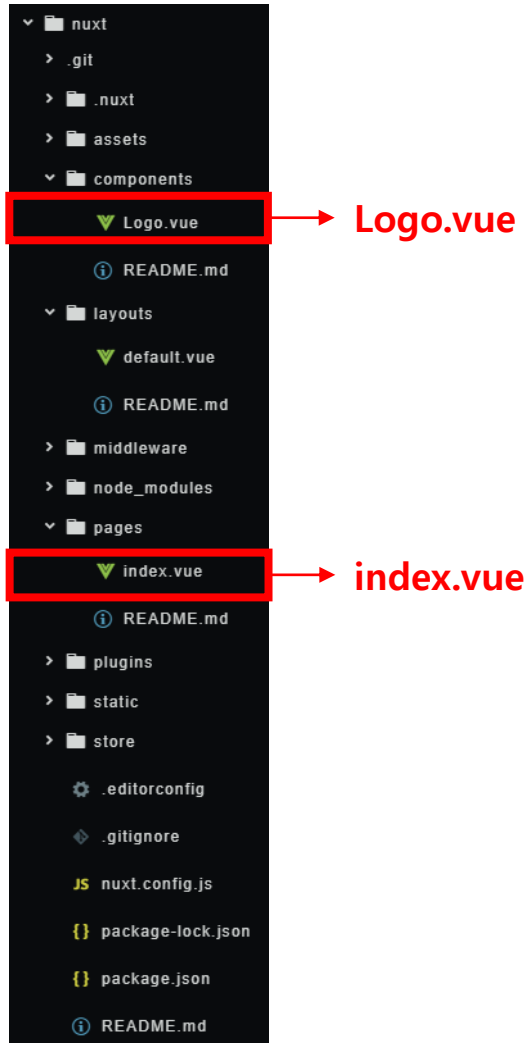

```
<template>
  <div>
    <Nuxt />
  </div>
</template>

<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

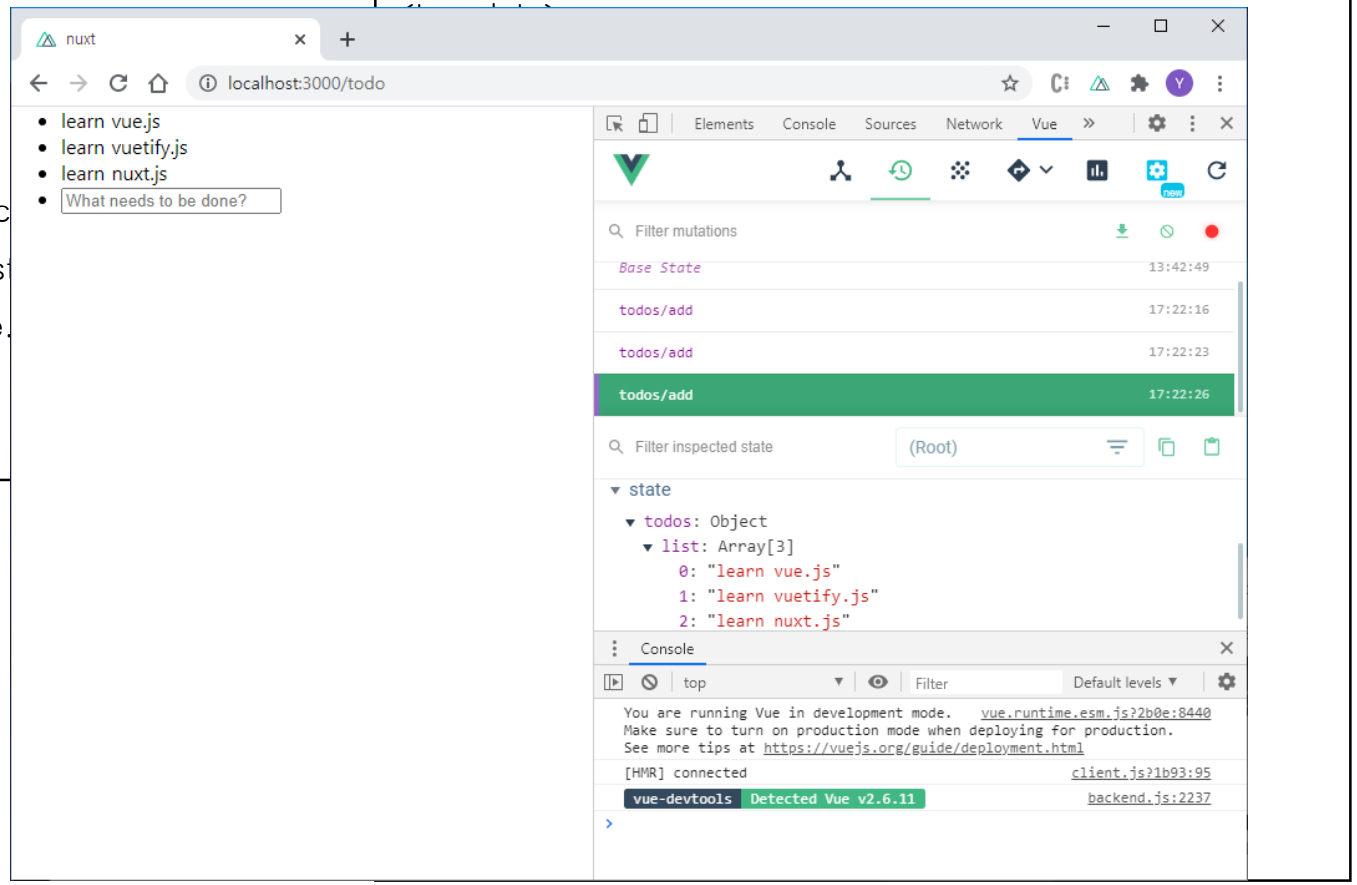
- NUXTJS (https://nuxtjs.org/) – The Intuitive Vue Framework

Vuex 공유 저장소

/store/todos.js

```
export const state = () => ({
  list: []
})

export c
  add (st
    state.
  }
}
```

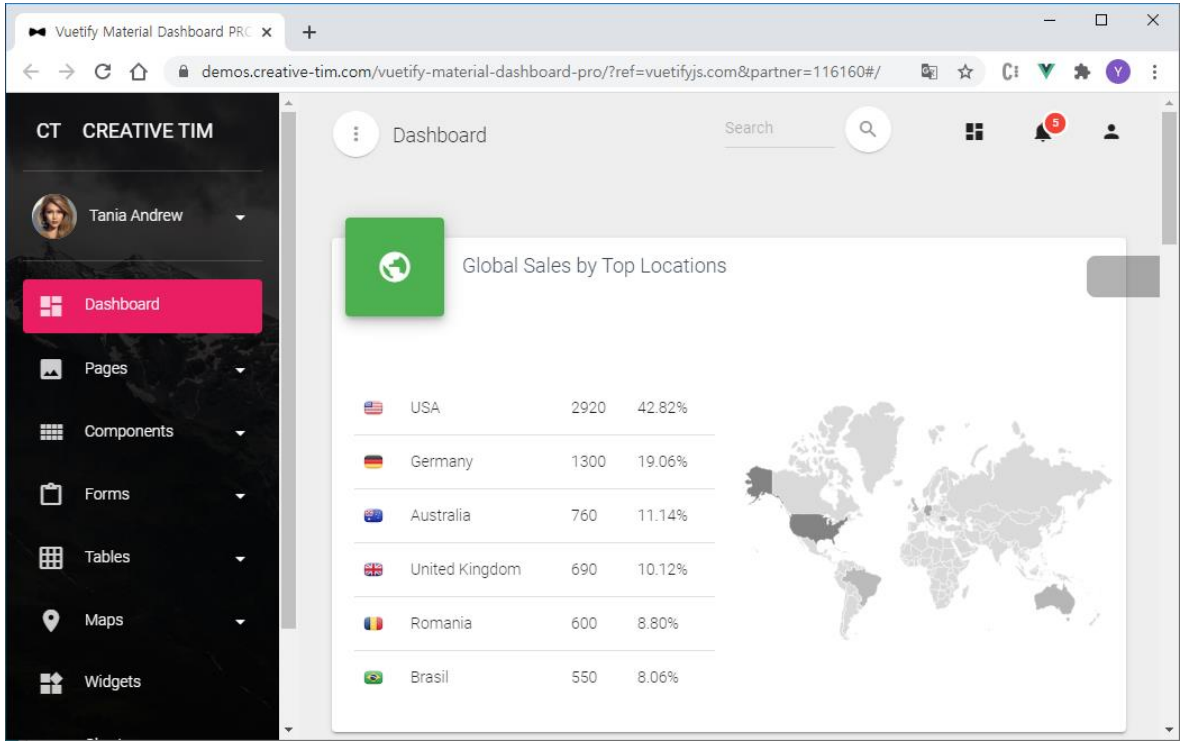
/pages/todo.vue

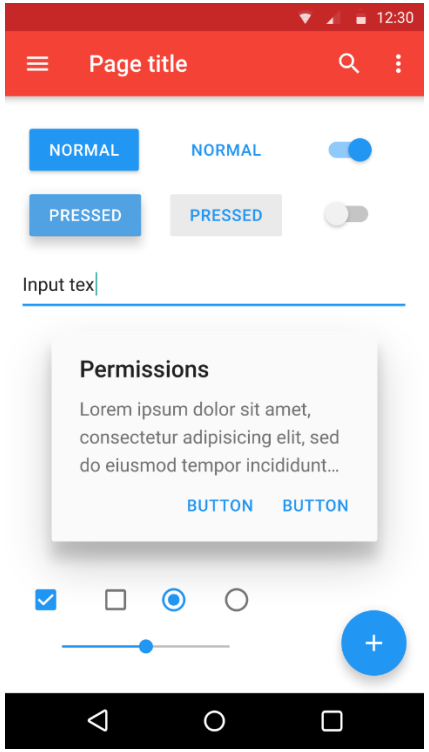


6. Vue.js / Nuxt.js / Vuetify.js

- **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 selectio

? Testing framework: None

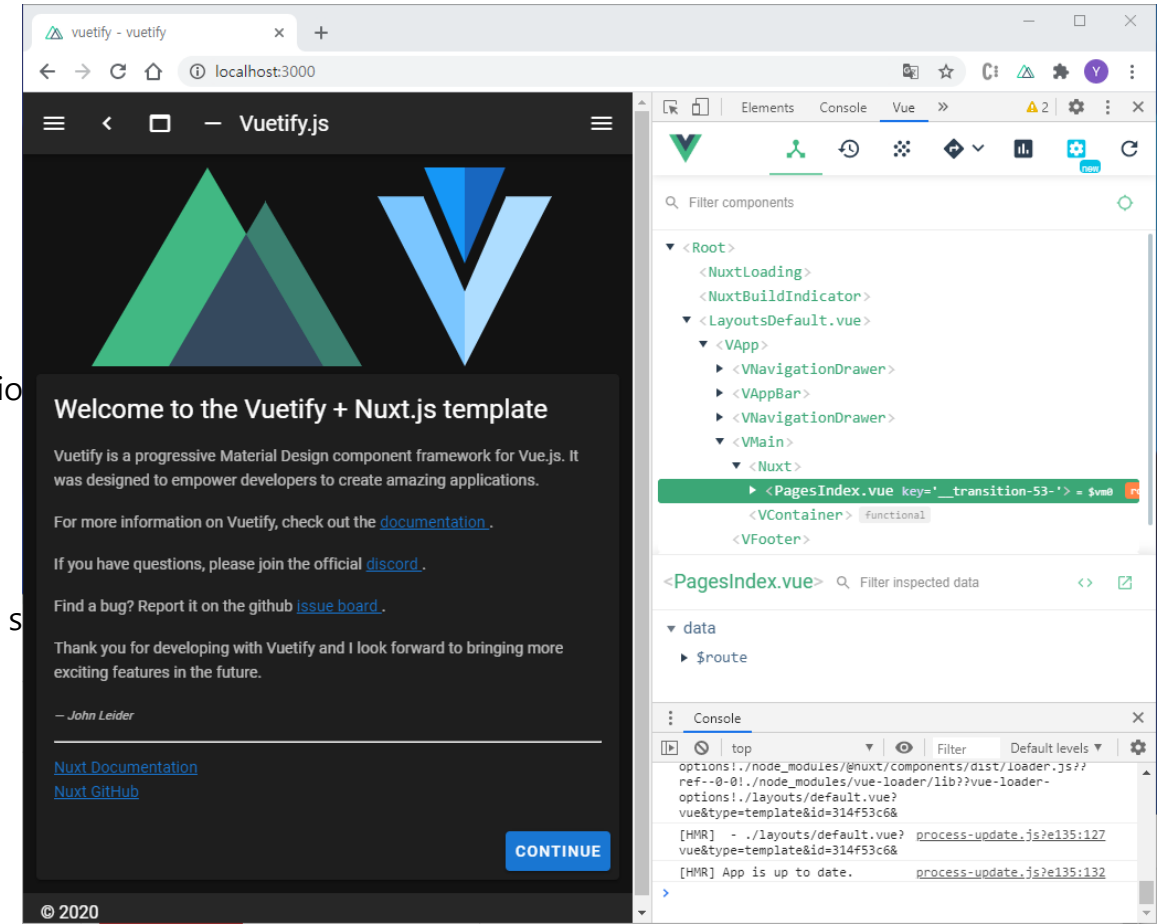
? Rendering mode: Universal (SSR / SSG)

? Deployment target: Server (Node.js hosting)

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

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 selectio

? Testing framework: None

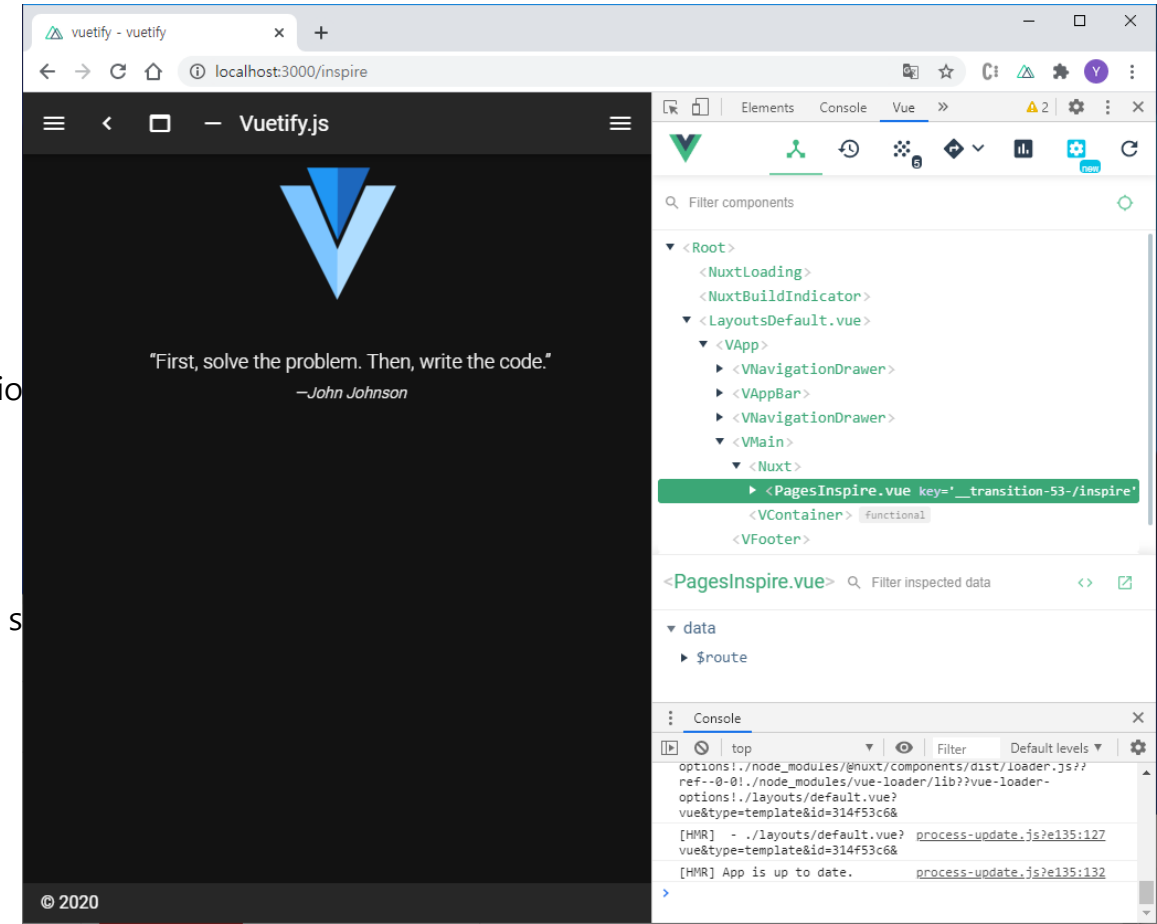
? Rendering mode: Universal (SSR / SSG)

? Deployment target: Server (Node.js hosting)

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

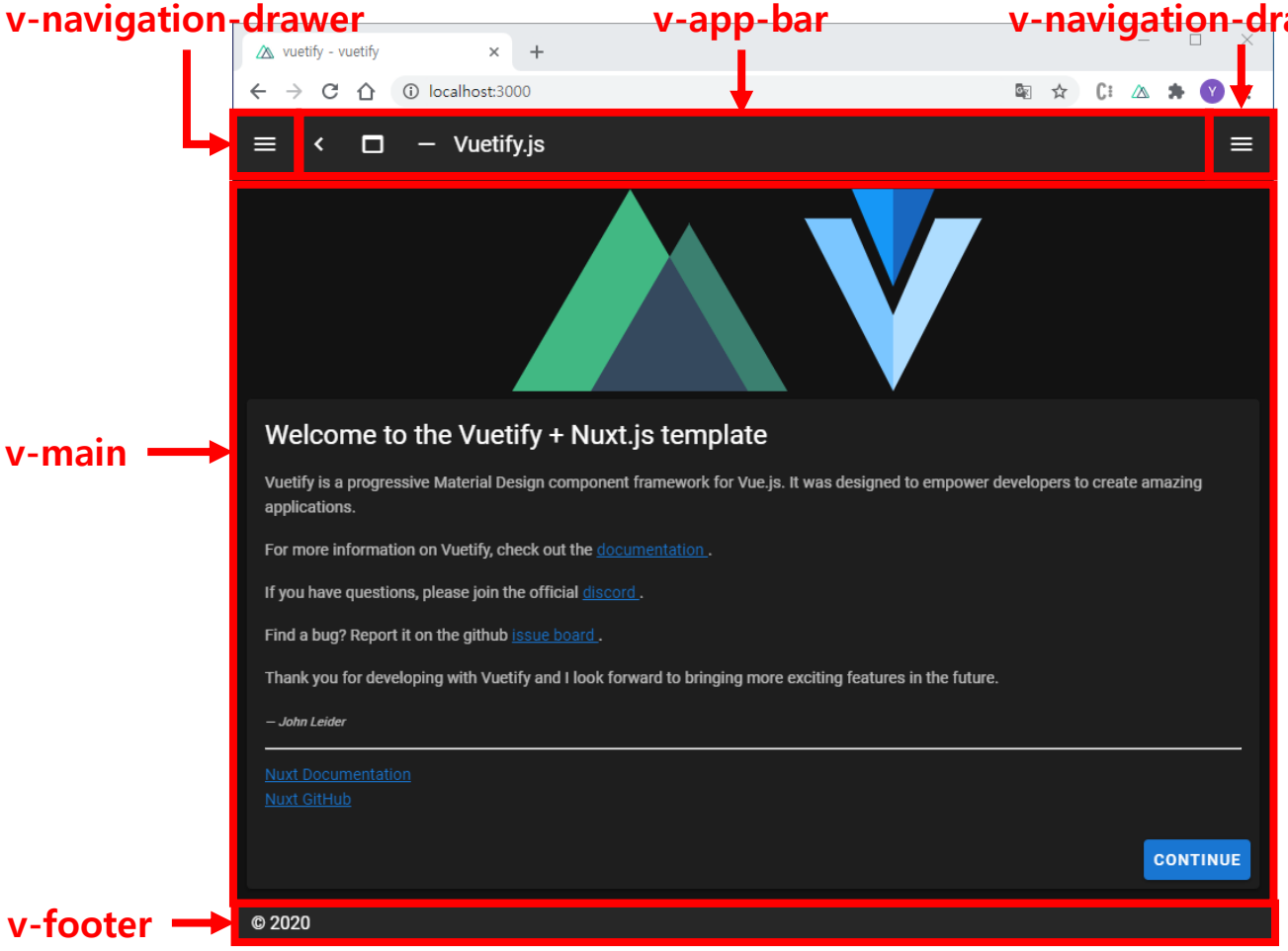
C:₩>cd vuetify

C:₩vuetify> **npm run dev**



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

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



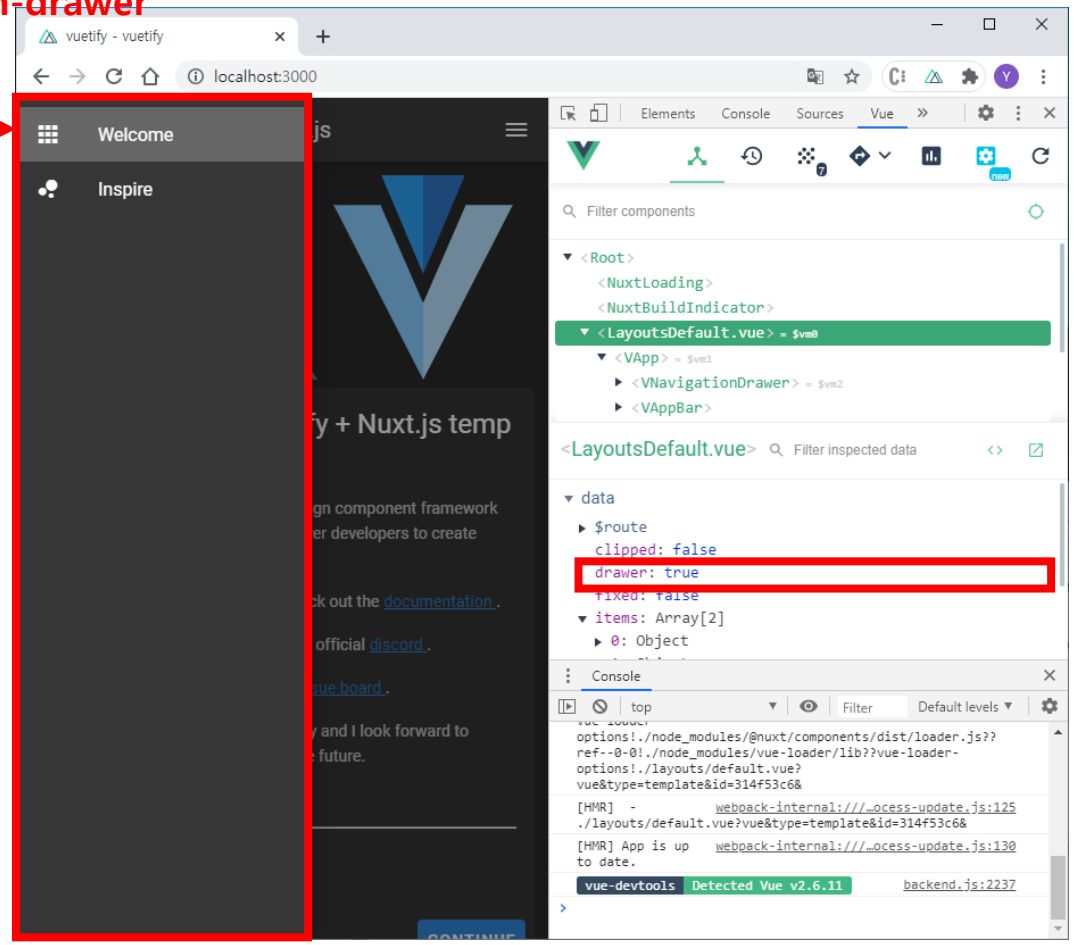
/layouts/default.vue

```
<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

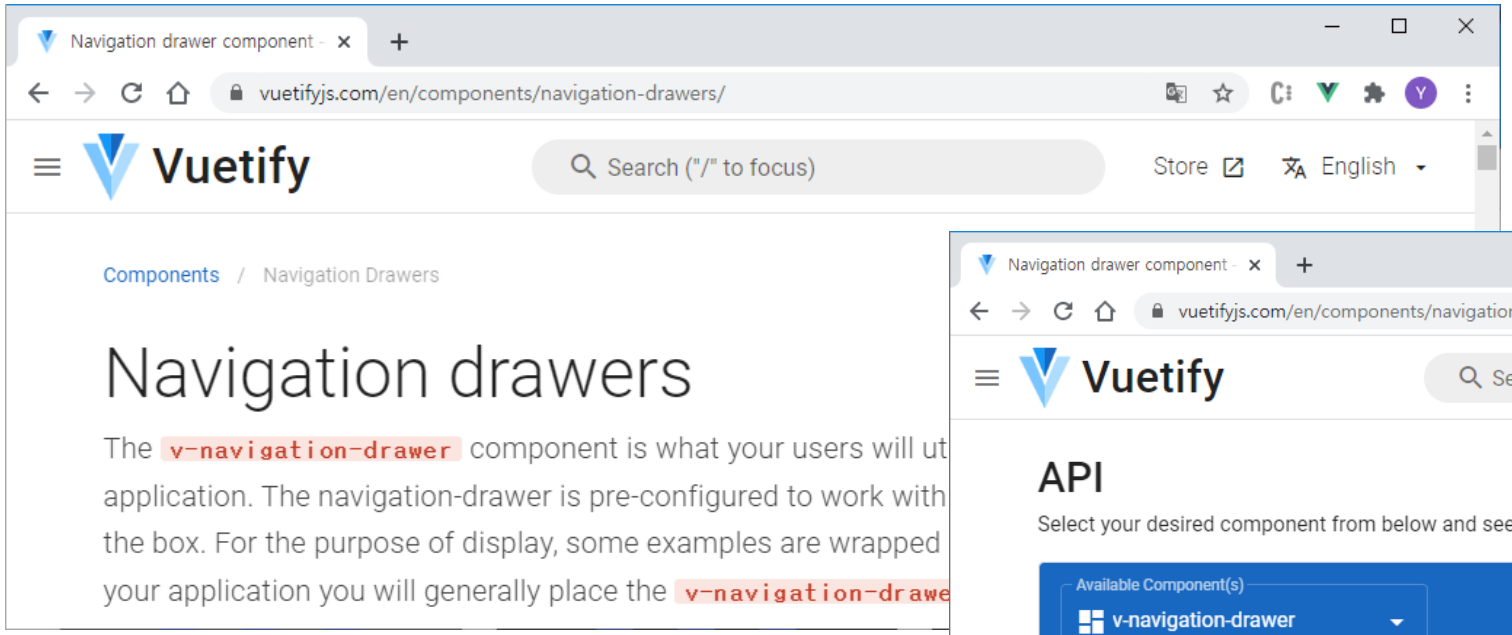
```
<template>
  <v-app dark>
    <v-navigation-drawer  v-model="drawer"
      :mini-variant="miniVariant"
      :clipped="clipped" fixed app>
      <v-list>
        <v-list-item v-for="(item, i) in items"
          :key="i"
          :to="item.to" router exact>
          <v-list-item-action>
            <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>
...
```

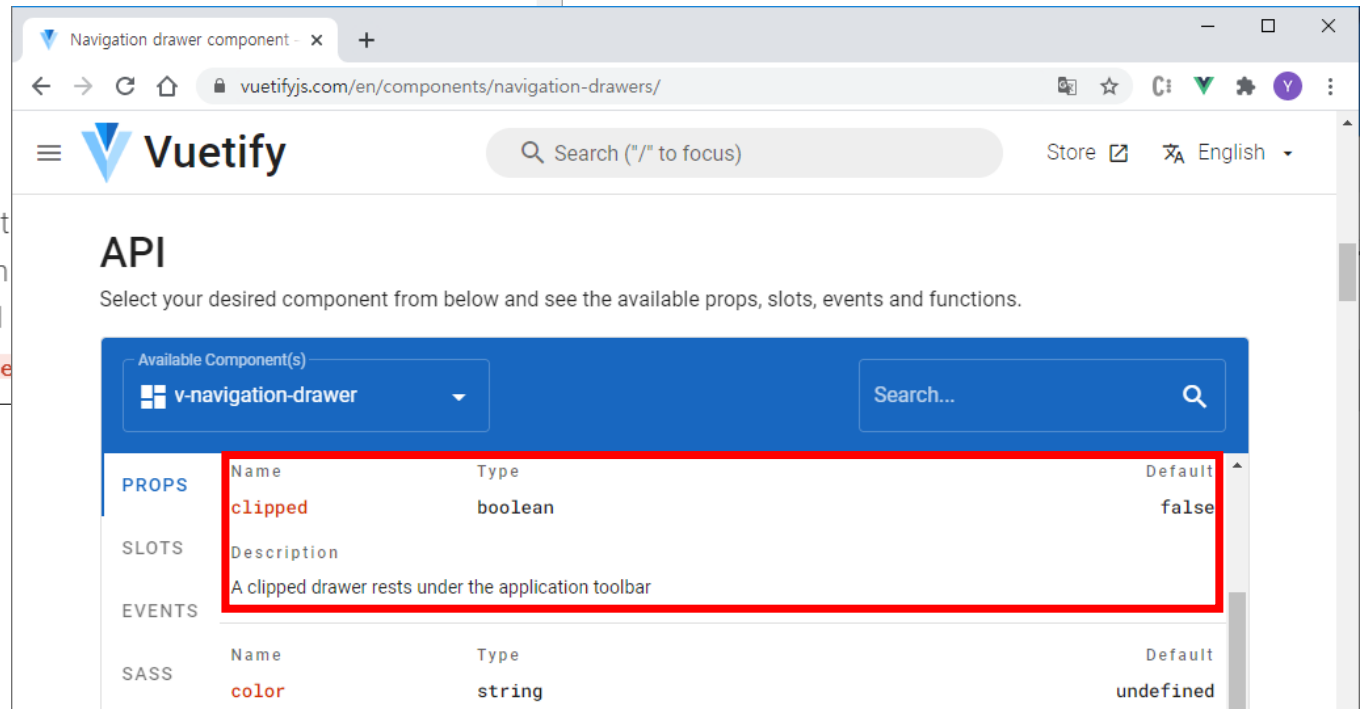
```
<script>
export default {
  data () {
    return {
      drawer: false,
      clipped: false,
      items: [
        {
          icon: 'mdi-apps',
          title: 'Welcome',
          to: '/'
        },
        {
          icon: 'mdi-chart-bubb
          title: 'Inspire',
          to: '/inspire'
        }
      ],
      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/

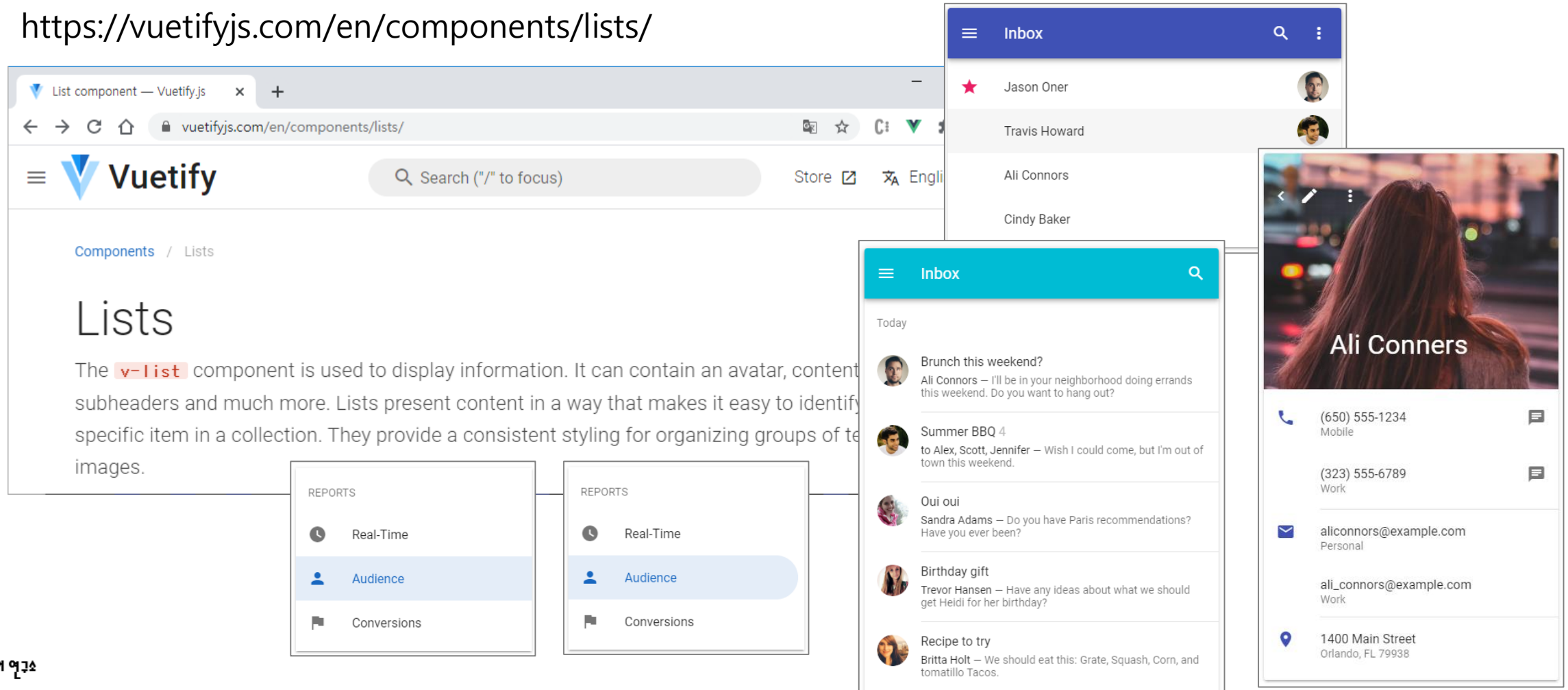




6. Vue.js / Nuxt.js / Vuetify.js

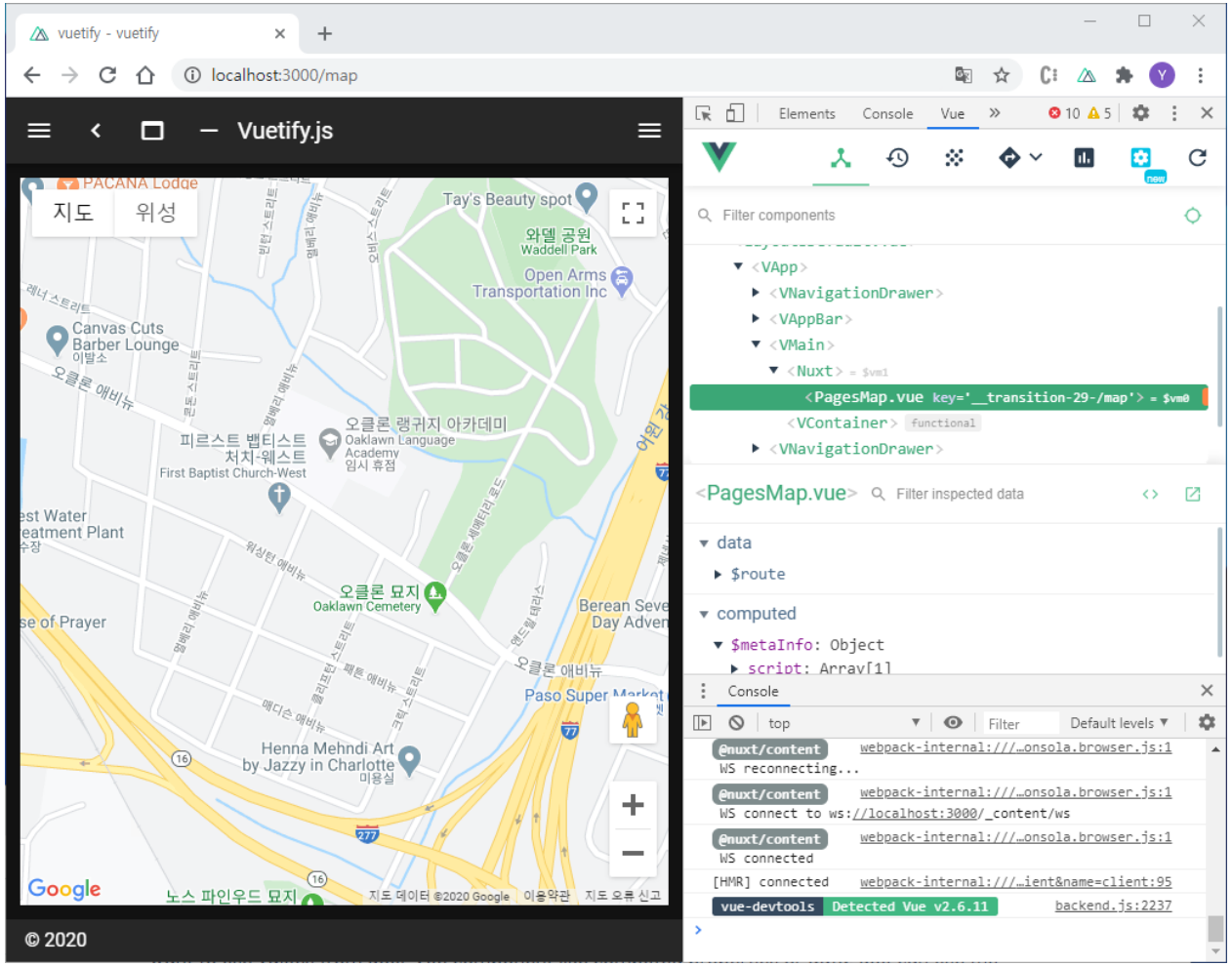
- **Vuetify** (https://vuetifyjs.com/) – **Material Design** Component Framework

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



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

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



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

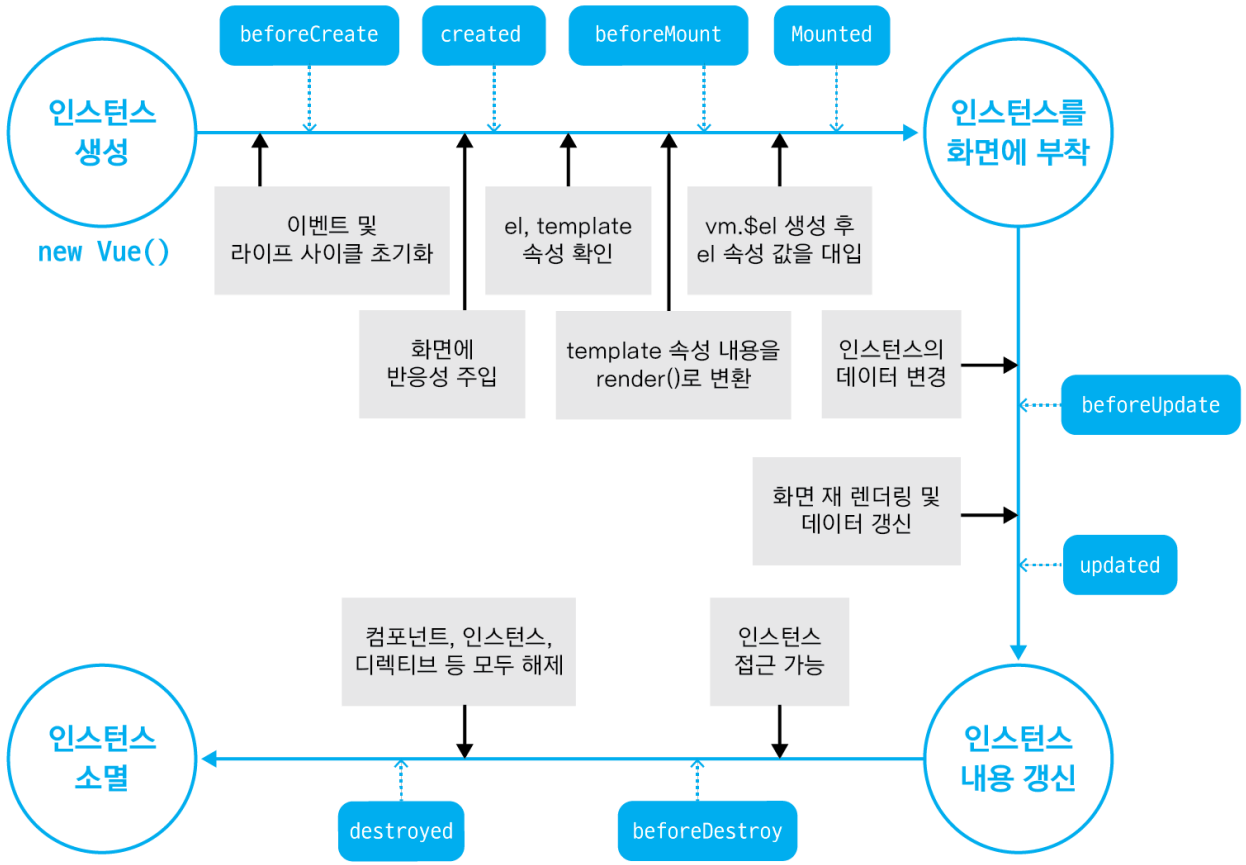
- Integrating Google Maps (tutorials/vue/vuetify/pages/map.vue)

```
<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)



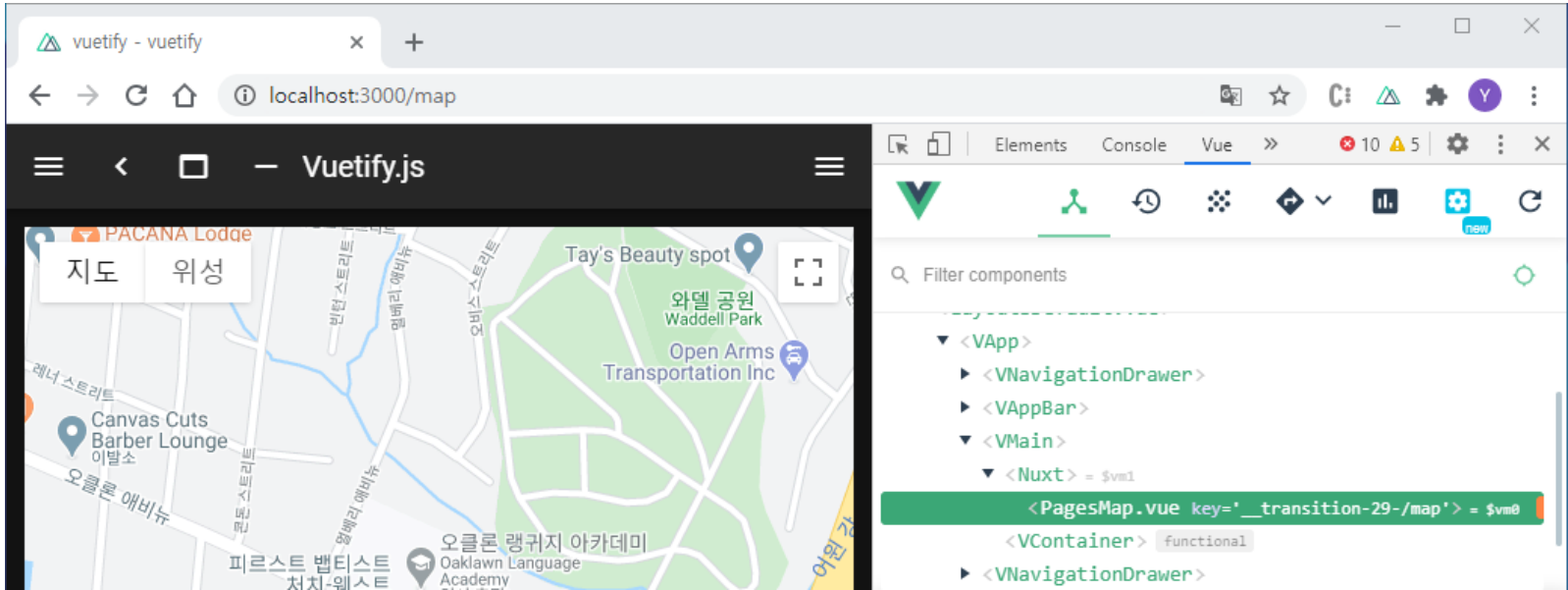
6. Vue.js / Nuxt.js / Vuetify.js

- 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>
...
```

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



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

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'
        }
      ],
```

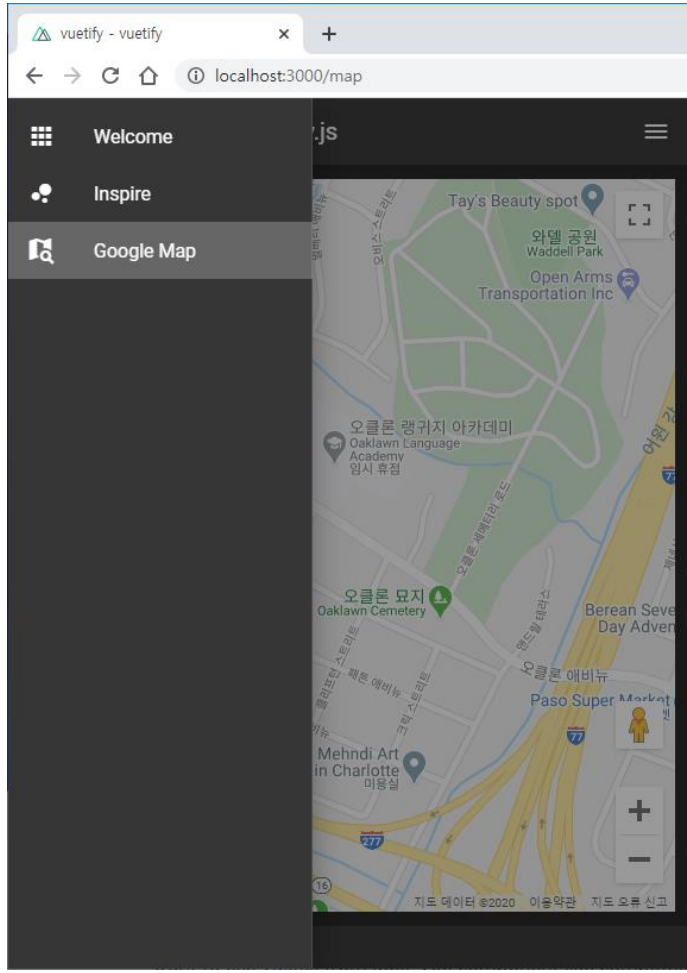
Drawer menu에 map 추가하기

mdi ← material design icon

Vuetify가 @mdi/js를 이용하는 듯.

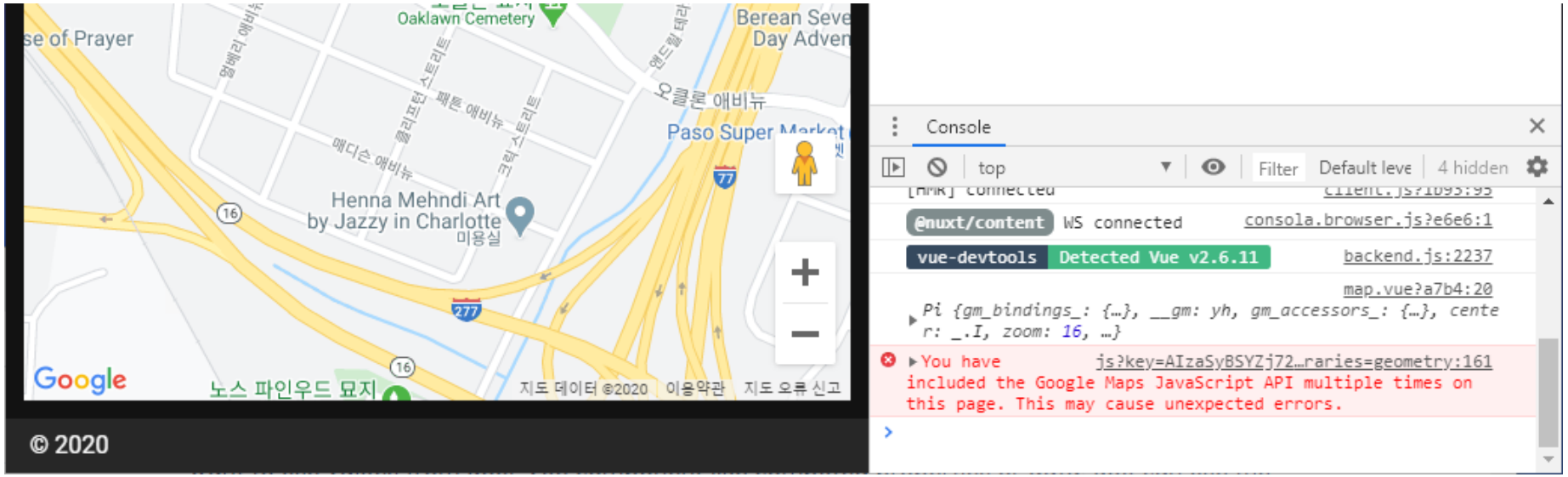
(https://www.npmjs.com/package/@mdi/js)

https://materialdesignicons.com에서 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)

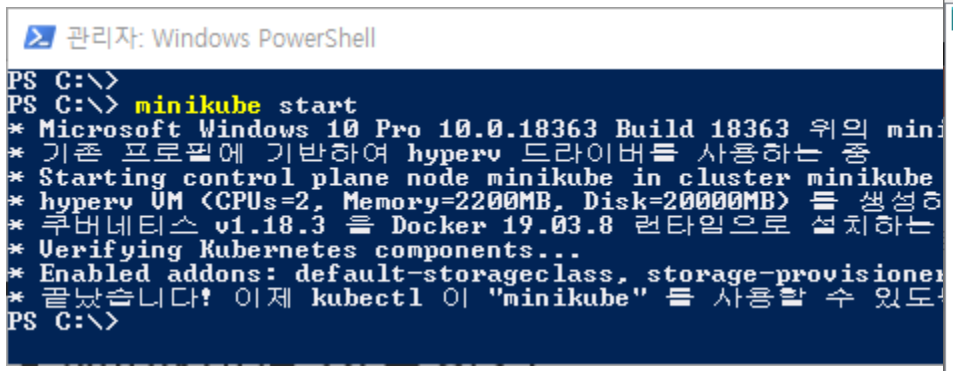
```
<script>
export default {
  mounted(){
    const map =  new google.maps.Map(document.getElementById('map'), { center: { lat: 35.25, lng: -80.85 }, zoom: 16 });
    map.addListener('idle', function(){
      map.data.forEach(function(feature){ map.data.remove(feature); });

      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

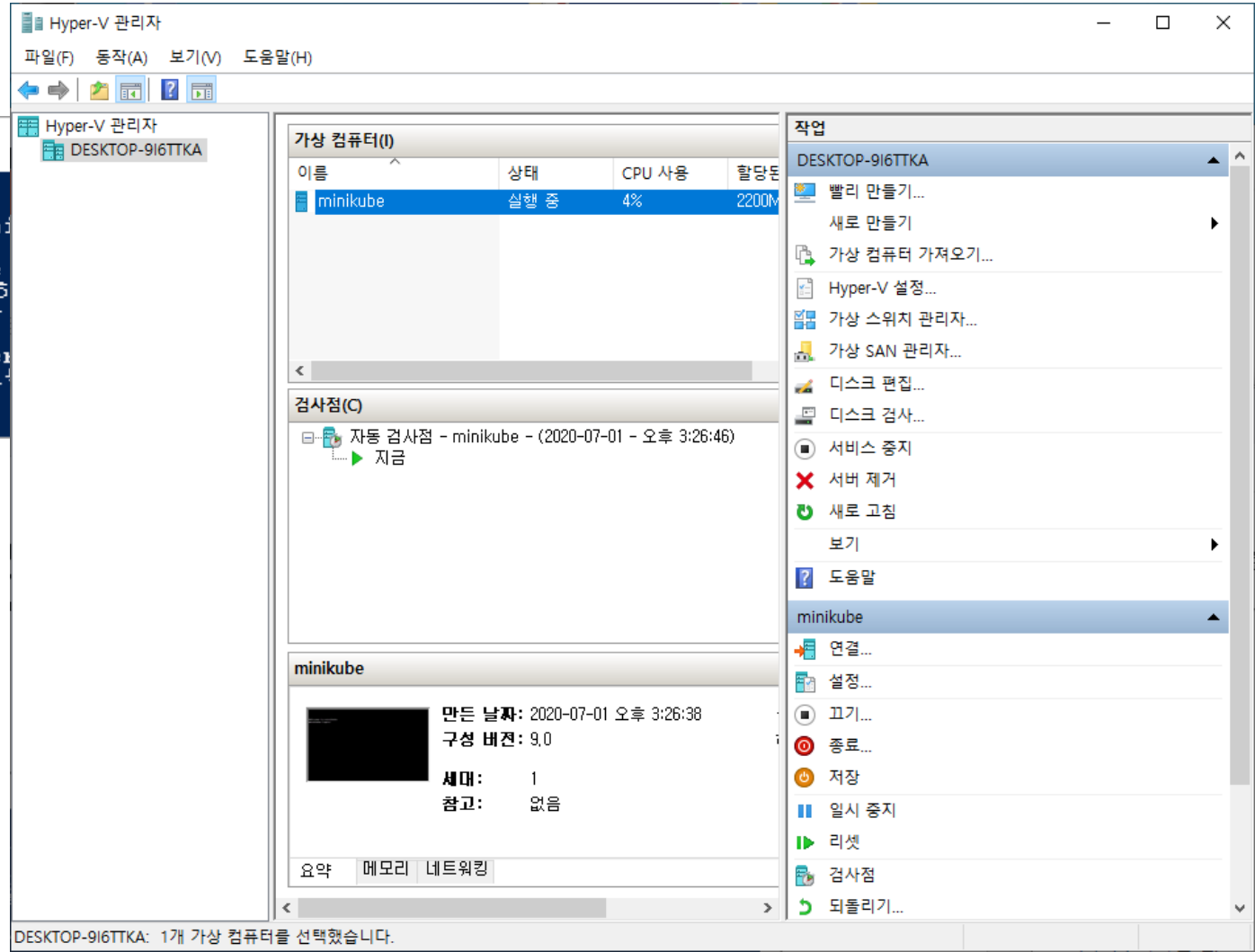
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

- Minikube 시작하기

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



Hyper-V 관리자에서 minikube 가상 머신을 확인한다.



7. Kubernetes

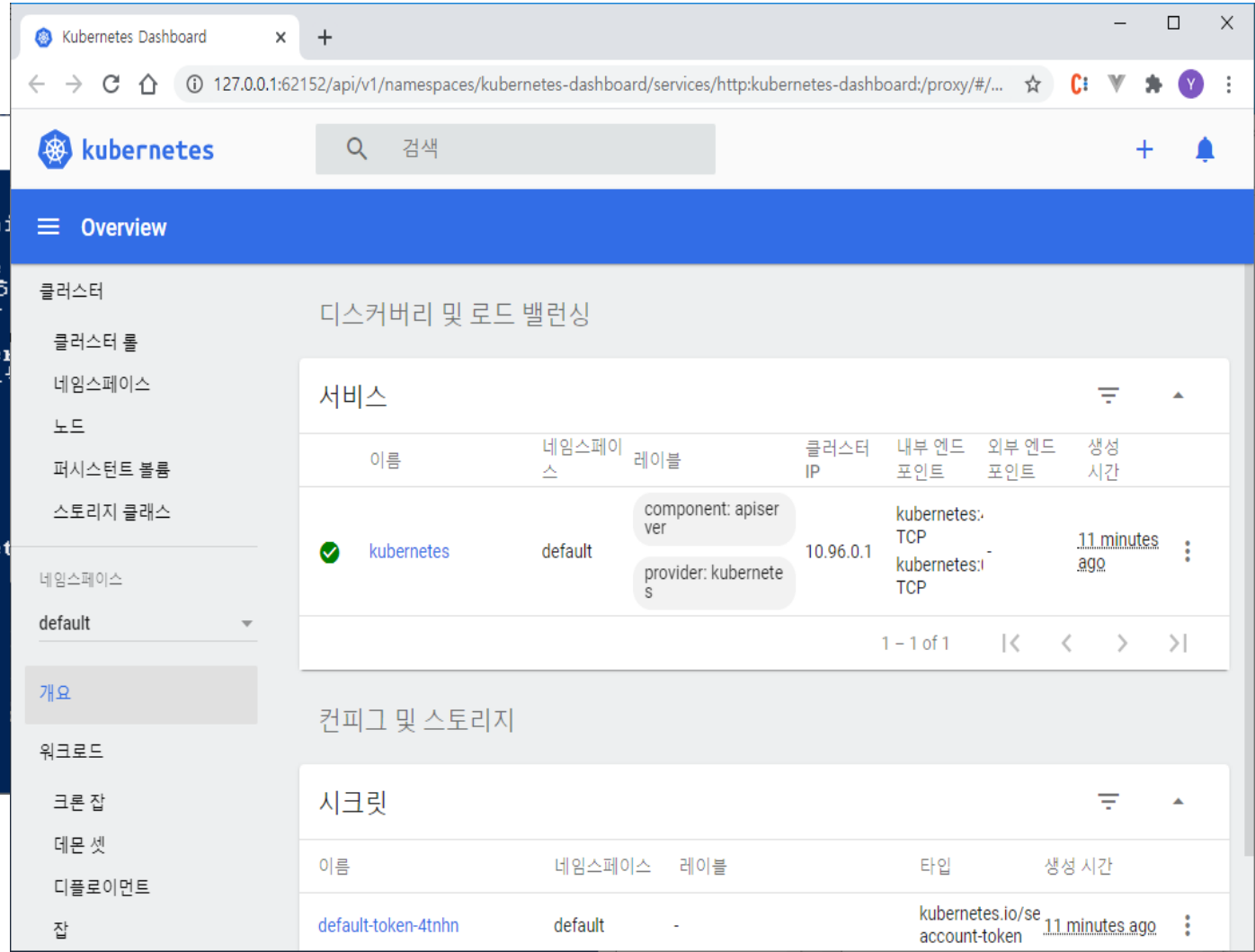
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

- Minikube 시작하기

2) minikube dashboard 명령으로 Kubernetes 서비스가 활성화 되었음을 확인한다.

```
관리자: Windows PowerShell
PS C:\>
PS C:\> minikube start
* Microsoft Windows 10 Pro 10.0.18363 Build 18363 위의 mini
* 기존 프로필에 기반하여 hyperv 드라이버를 사용하는 중
* Starting control plane node minikube in cluster minikube
* hyperv VM (CPUs=2, Memory=2200MB, Disk=20000MB) 를 생성하
* 쿠버네티스 v1.18.3 를 Docker 19.03.8 런타임으로 설치하는
* Verifying Kubernetes components...
* Enabled addons: default-storageclass, storage-provisioner
* 끝났습니다! 이제 kubectl 이 "minikube" 를 사용할 수 있도
PS C:\>
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/kubernet
ur default browser...
```

Kubernetes Dashboard를 확인한다.

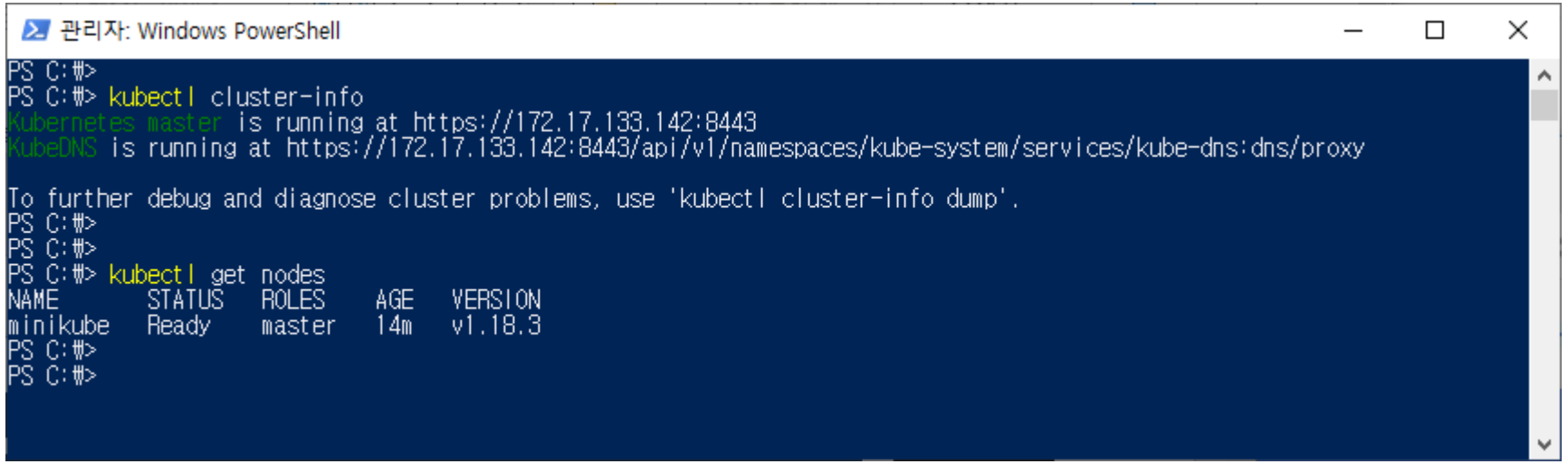


7. Kubernetes

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

- Minikube 시작하기

3) kubectl 명령으로 cluster와 node 정보를 확인한다.



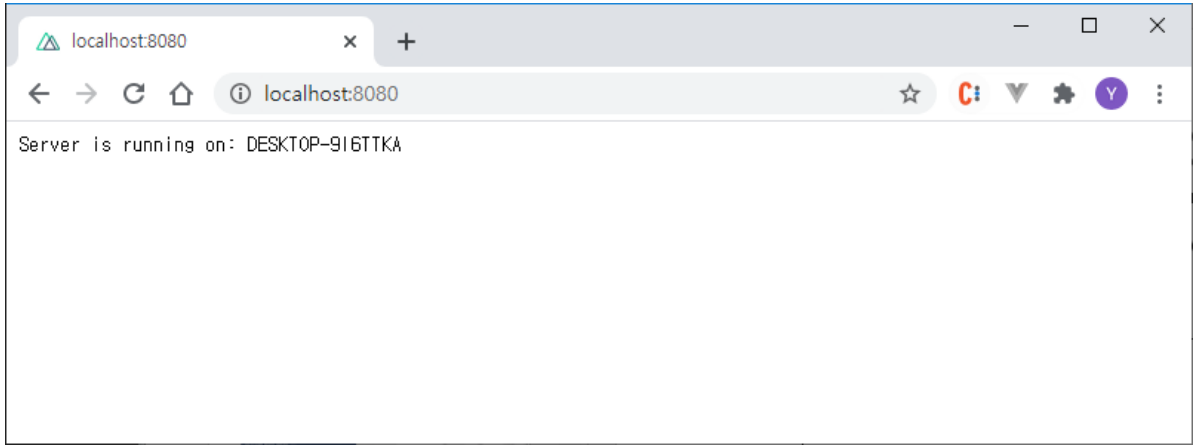
7. Kubernetes

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);
```

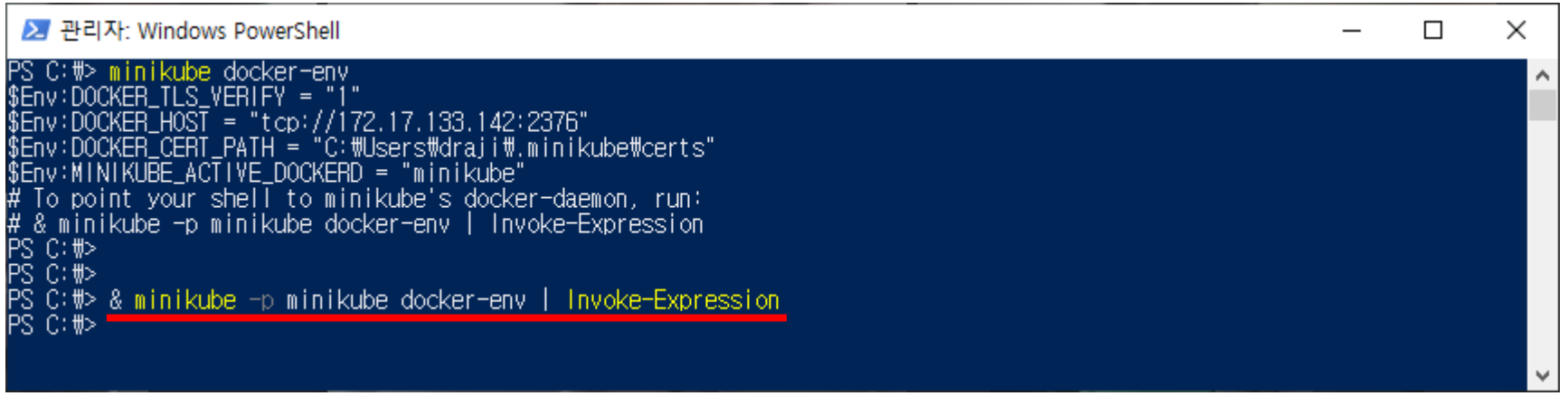


7. Kubernetes

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

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

2) Docker 환경 설정



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)

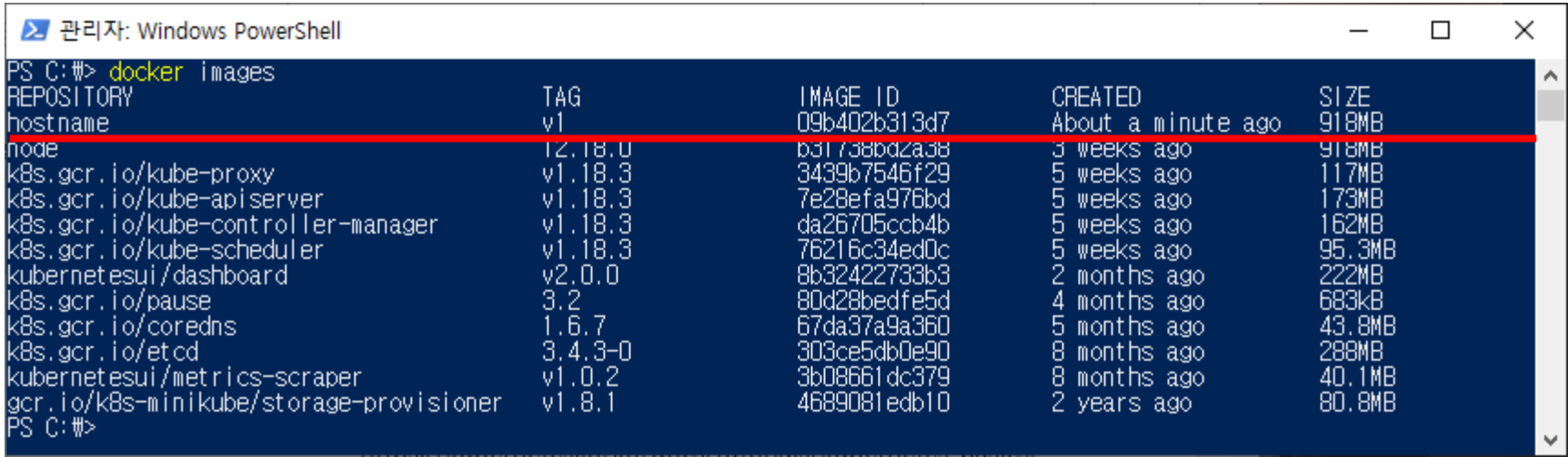
3) Docker 이미지 만들기

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.



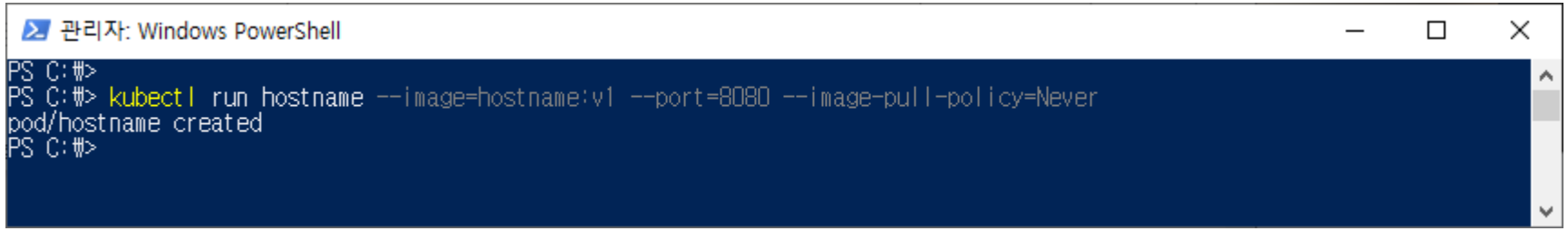
7. Kubernetes

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

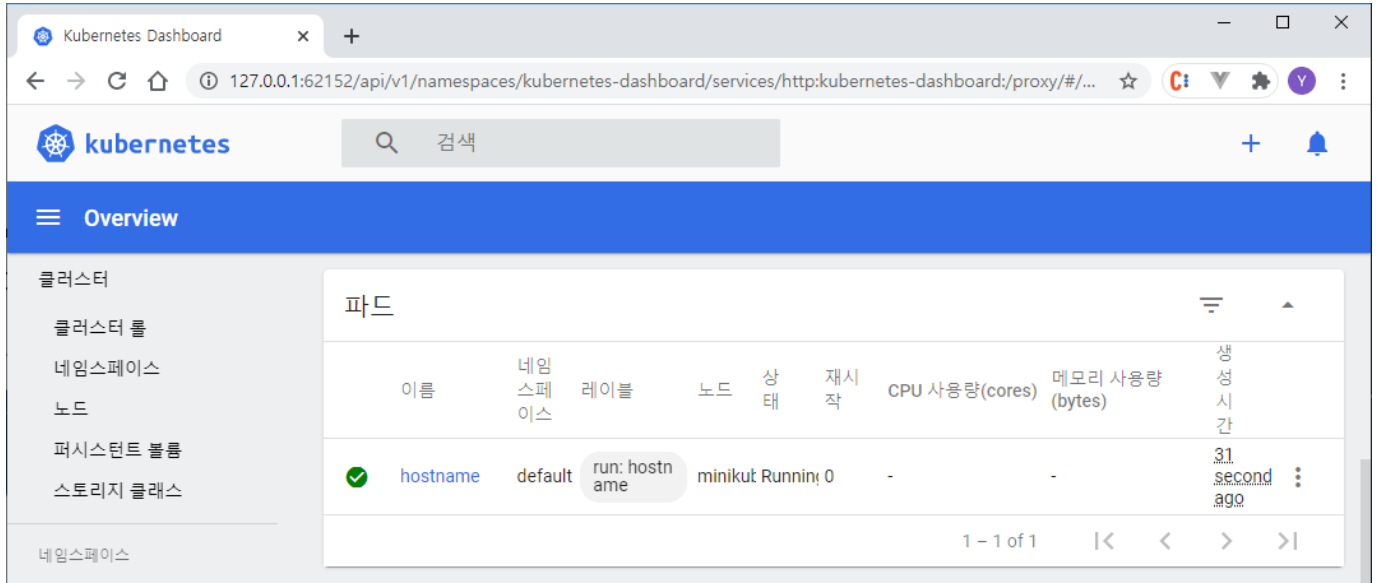
- 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

4) POD 생성하기

kubectl run 명령으로 POD를 생성한다.



dashboard에서 생성된 POD를 확인할 수 있다.



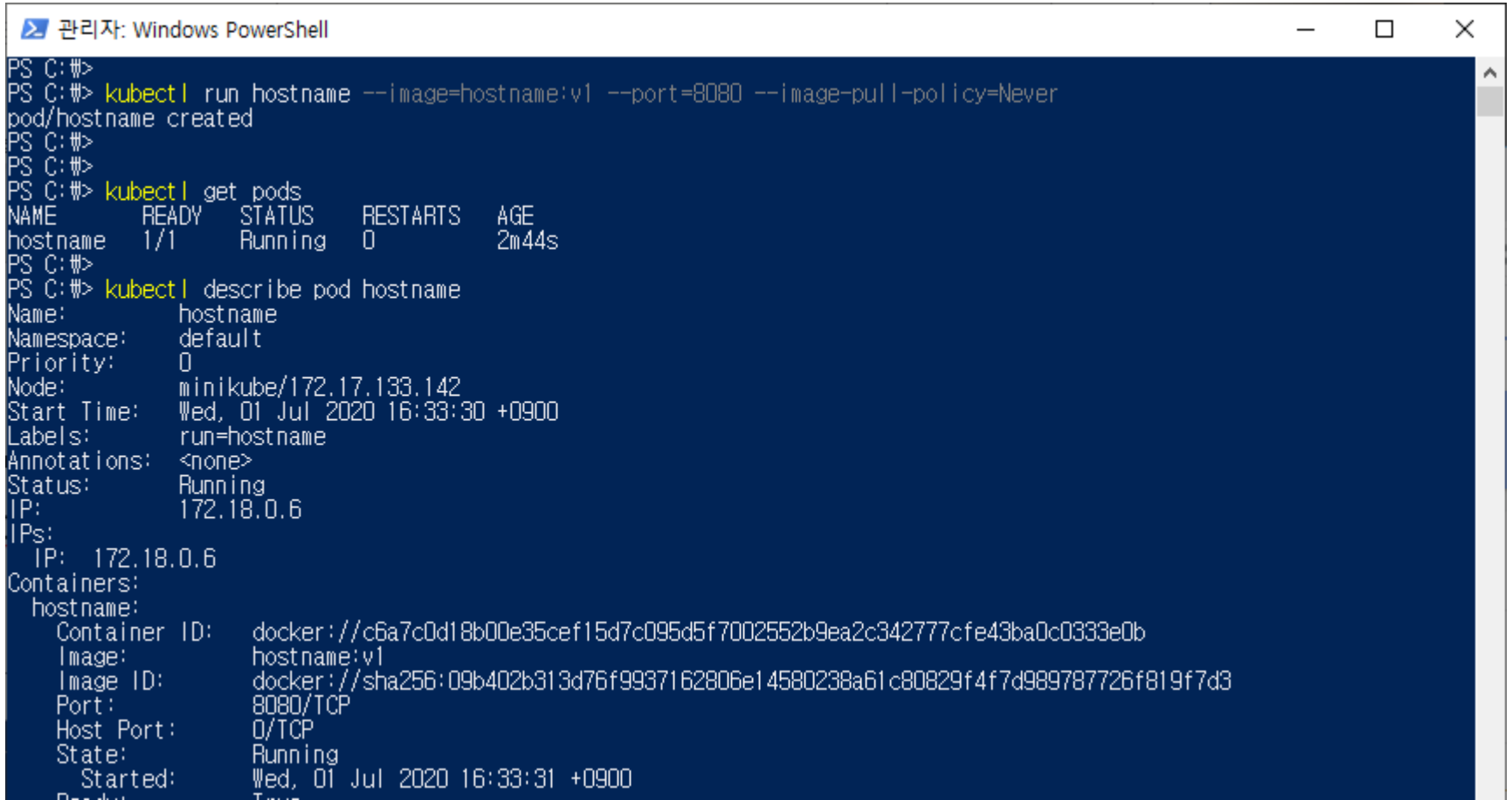
7. Kubernetes

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.



7. Kubernetes

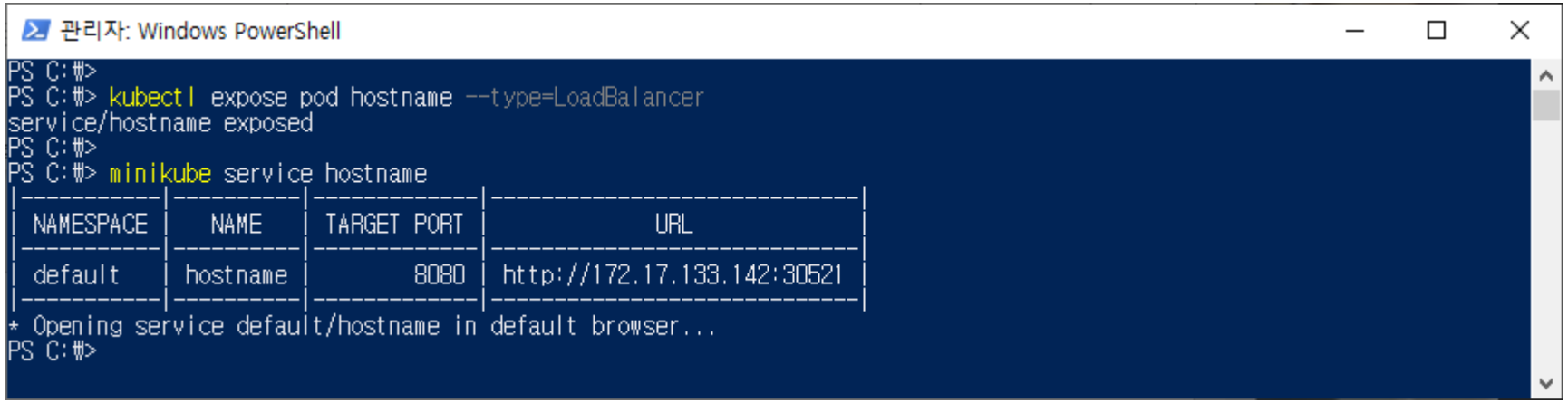
https://kubernetes.io/ko/docs/concepts/services-networking/

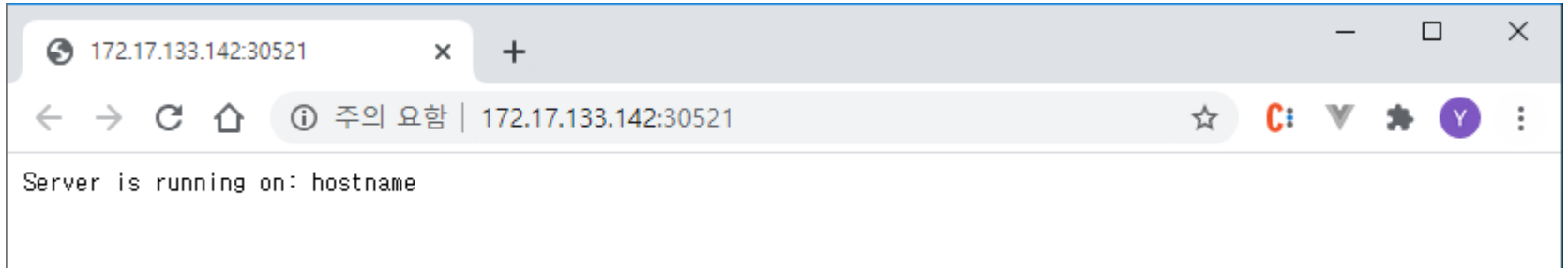
- 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

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.





7. Kubernetes

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

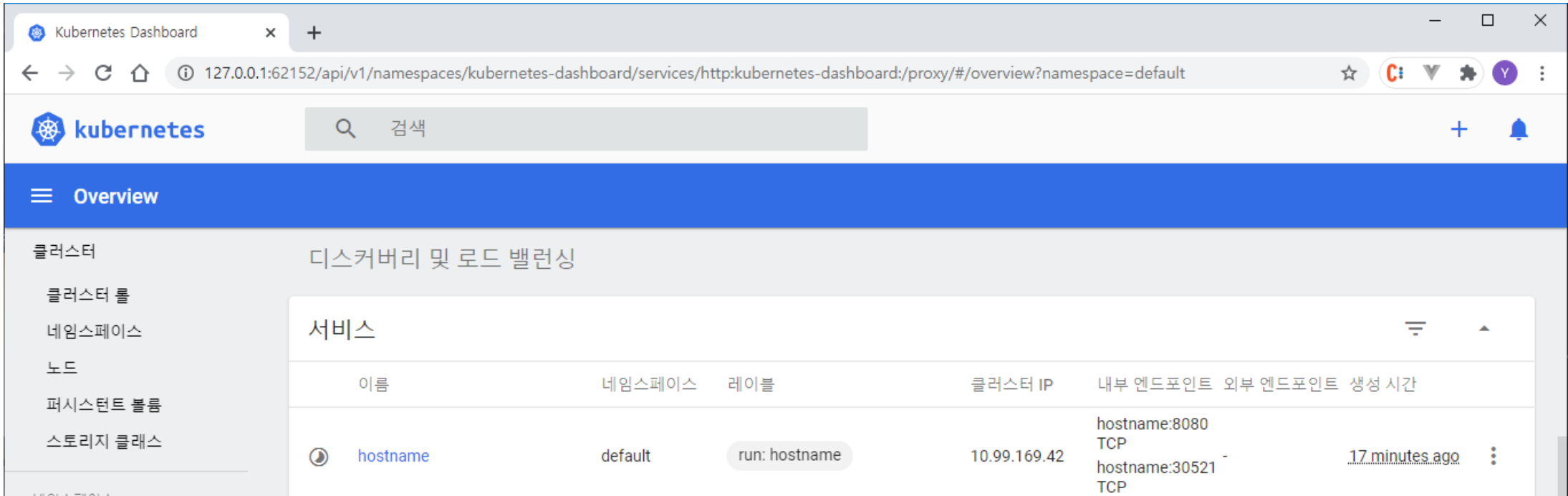
- 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

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.



7. Kubernetes

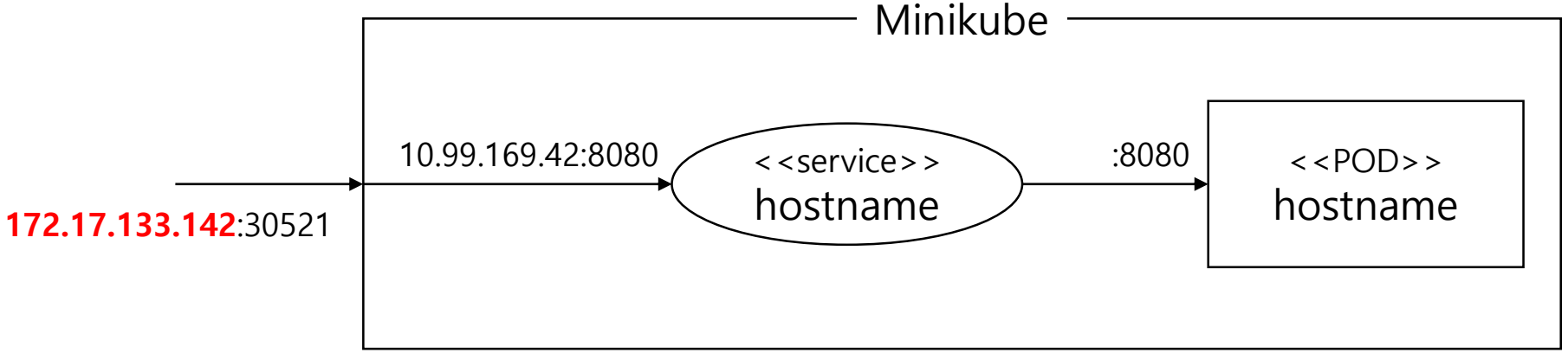
https://kubernetes.io/ko/docs/concepts/services-networking/

- 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

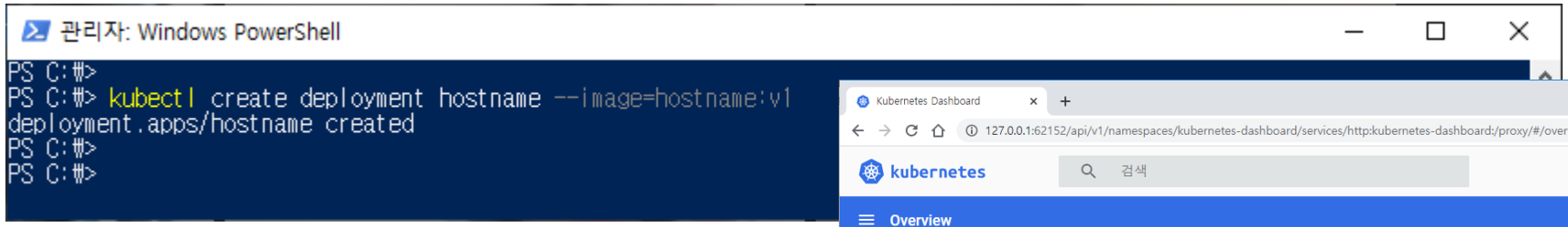
https://kubernetes.io/ko/docs/concepts/workloads/controllers/deployment/

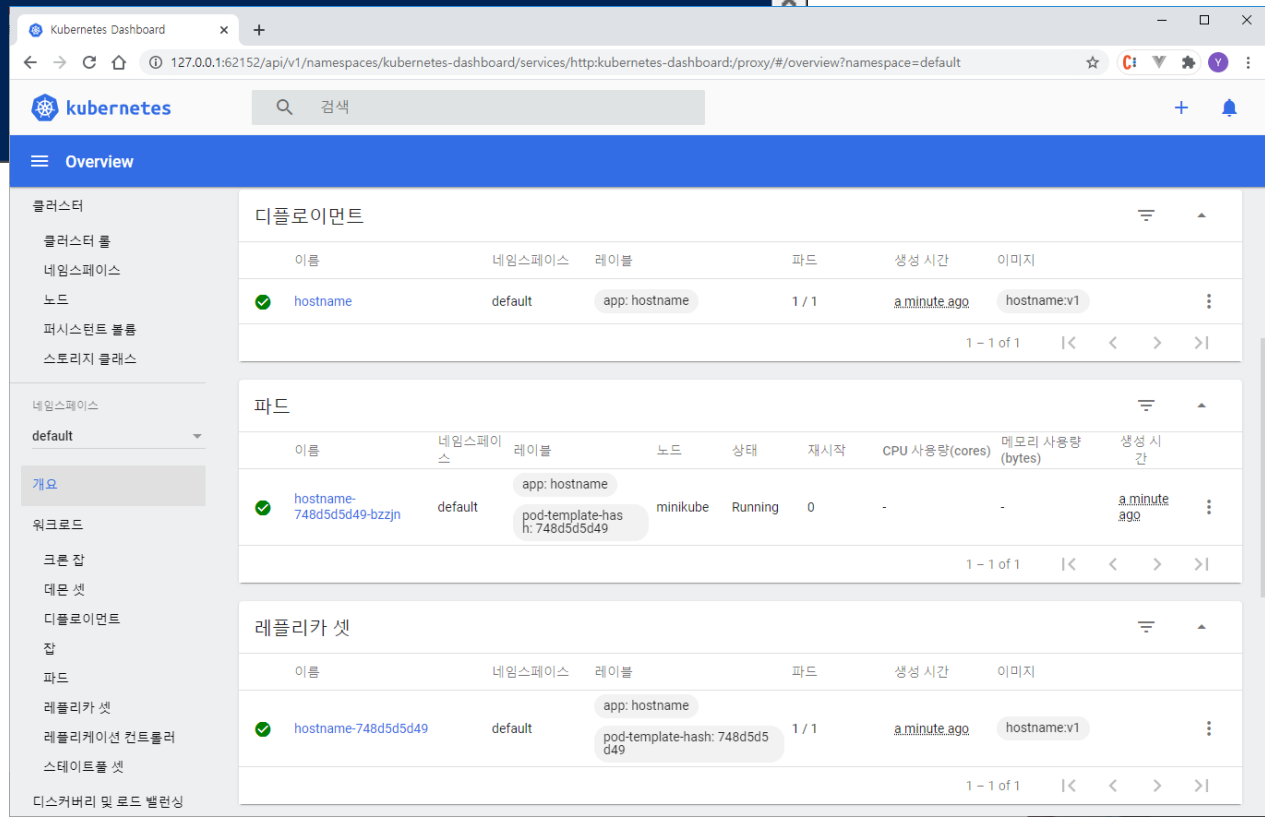
- 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

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.





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

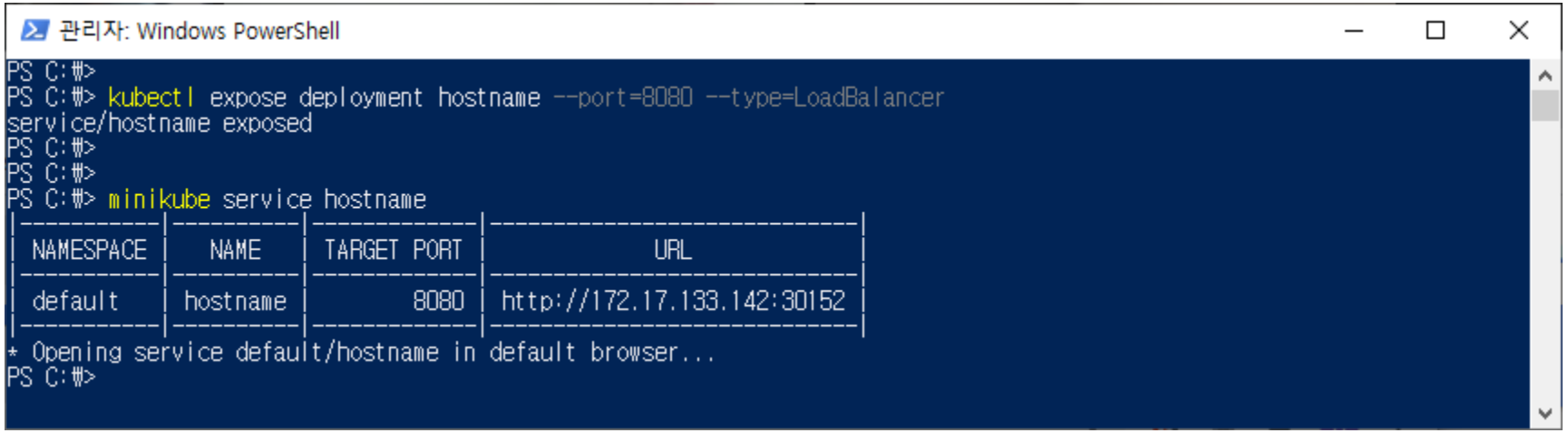
https://kubernetes.io/ko/docs/concepts/services-networking/

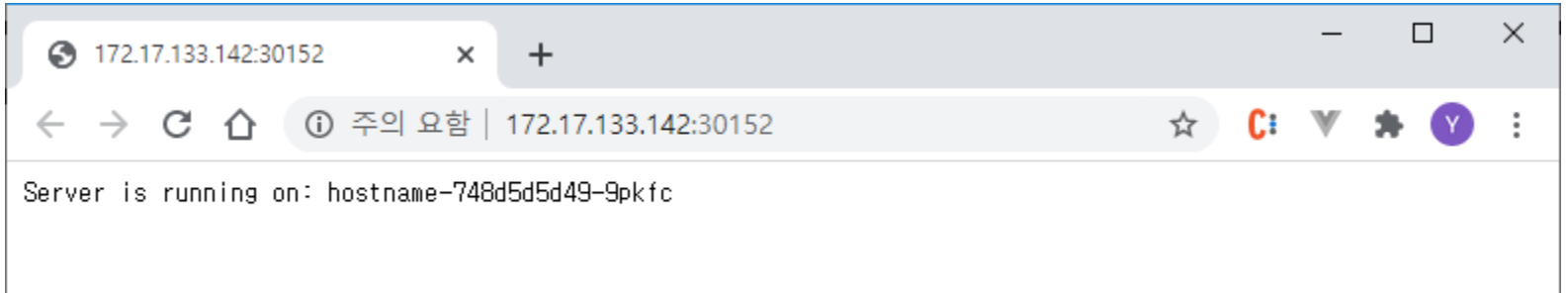
- 간단한 예제 - 1 (tutorials/kubernetes/server1.js)

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.





7. Kubernetes

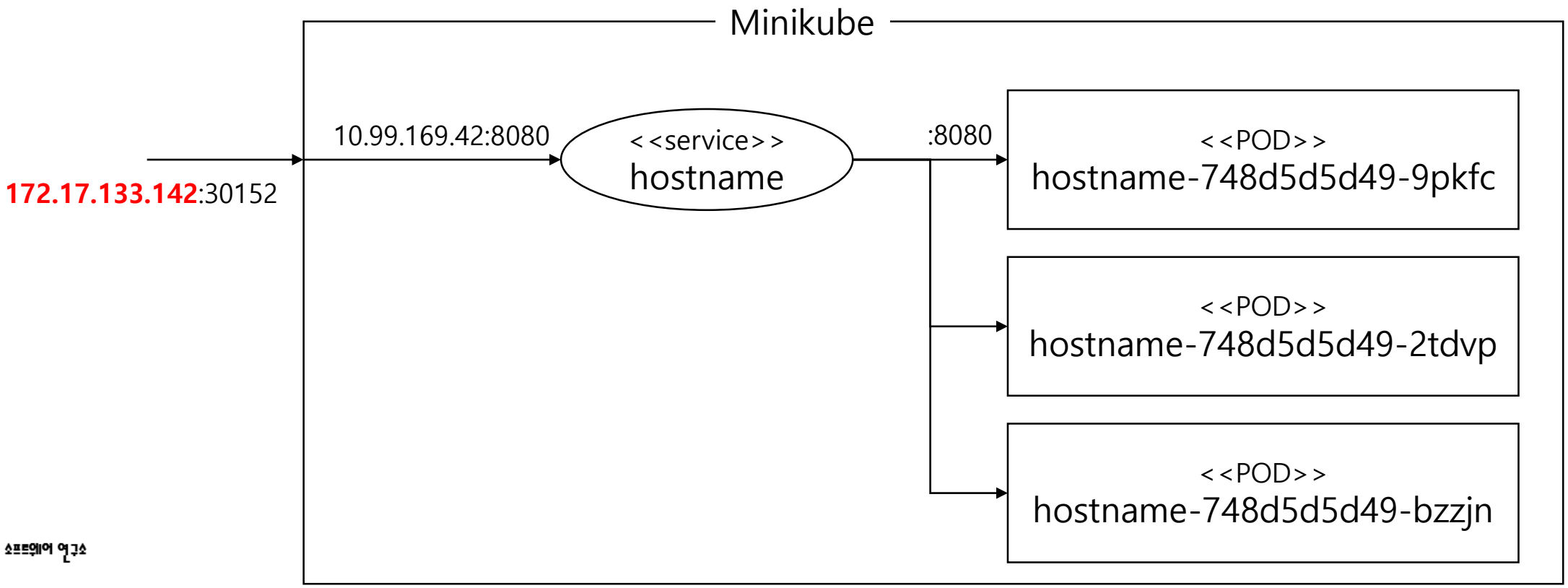
https://kubernetes.io/ko/docs/concepts/services-networking/

- 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.



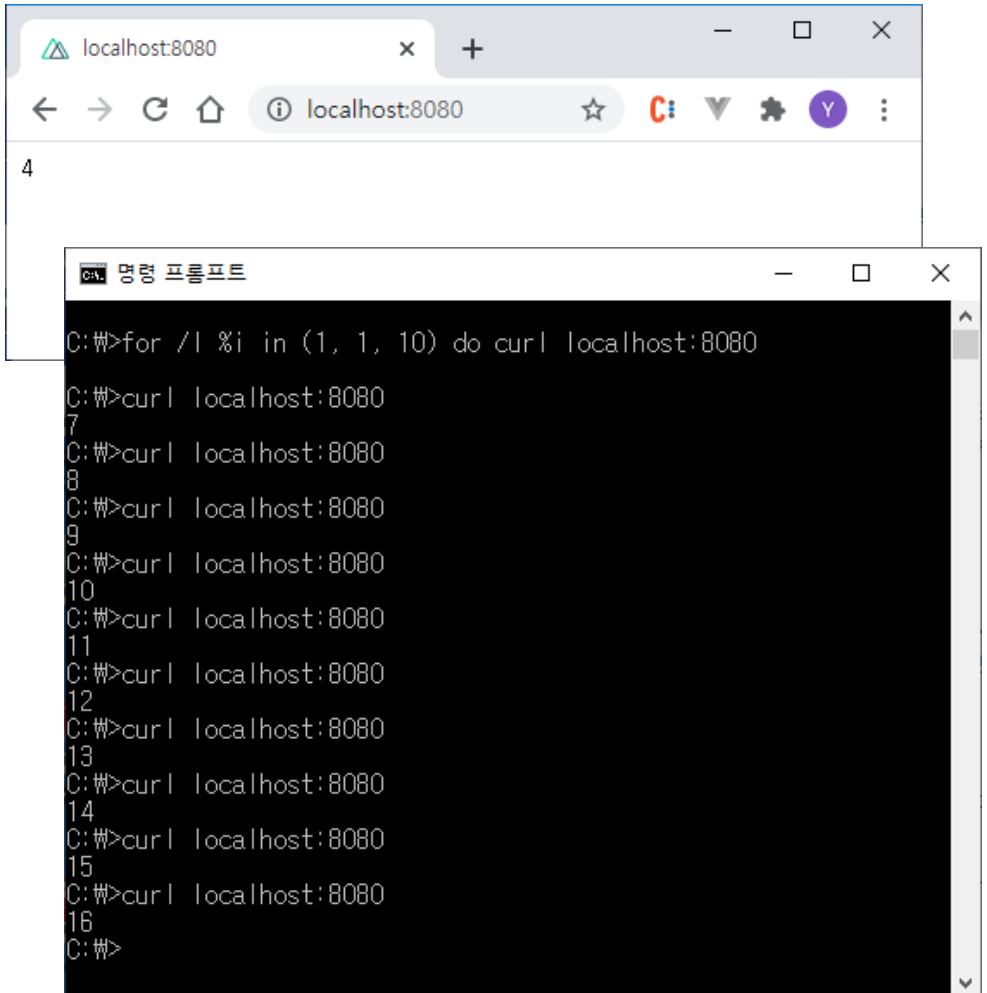
7. Kubernetes

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);
```



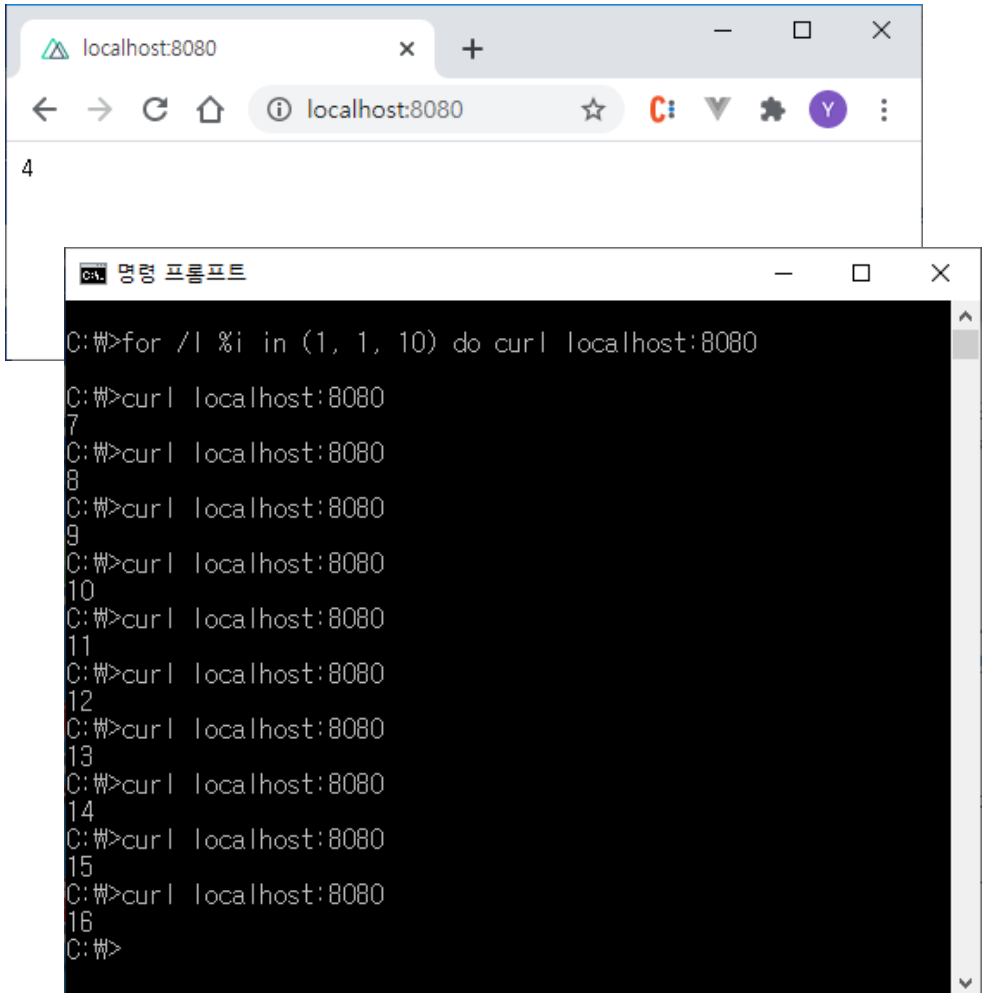
7. Kubernetes

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)

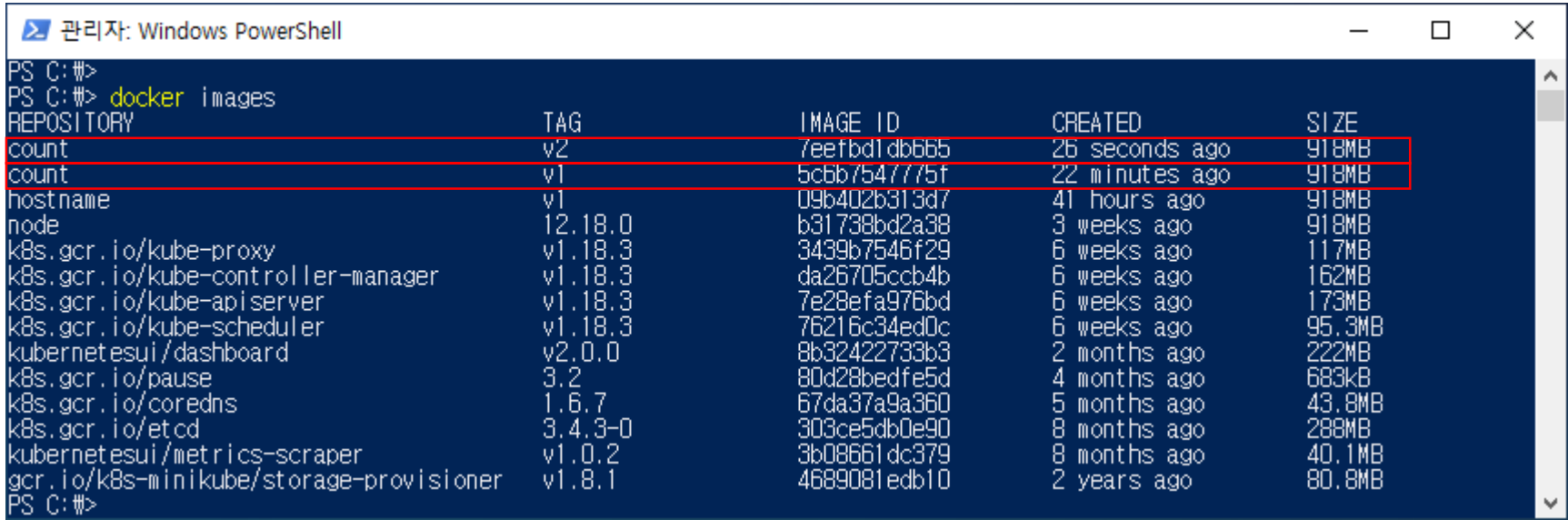
3) Docker image 빌드

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.



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)

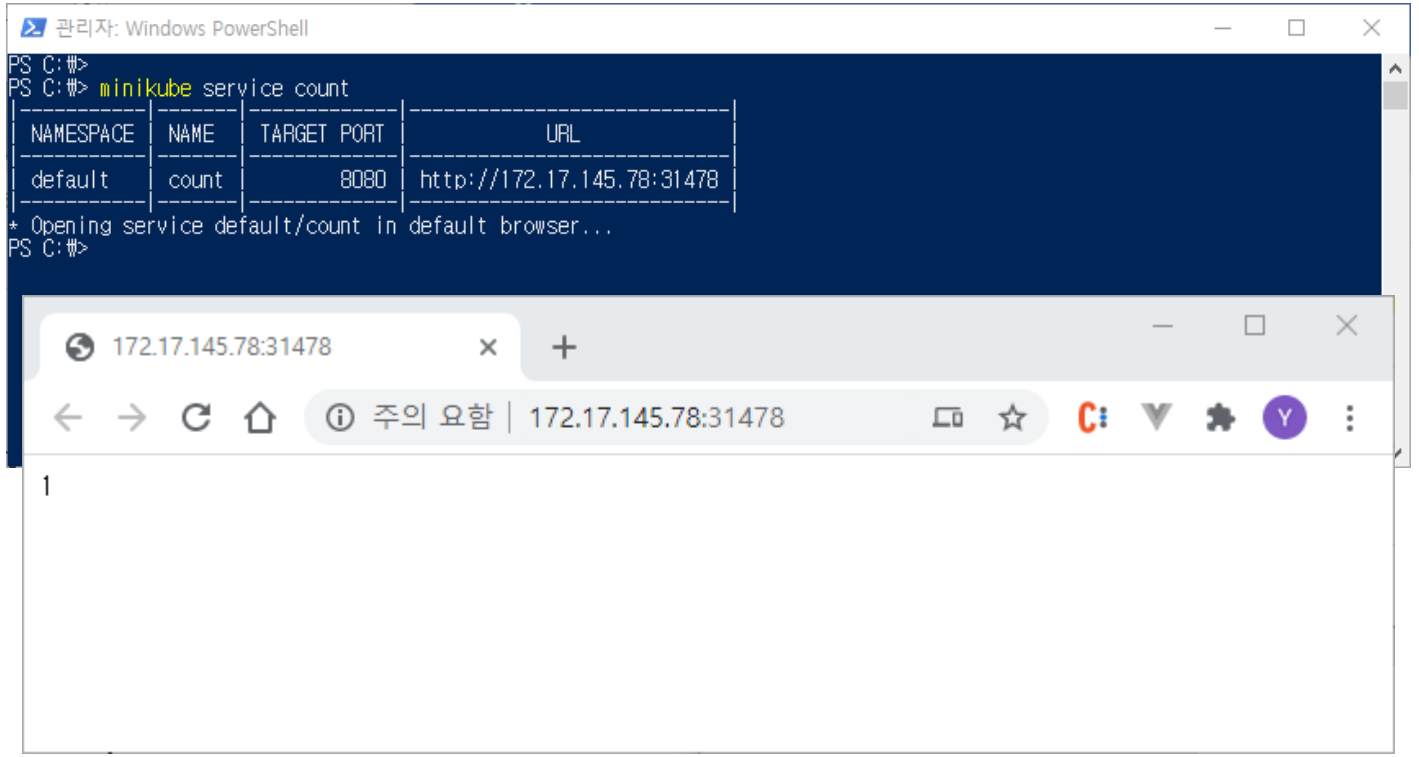
5) 서비스를 생성한다.

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.

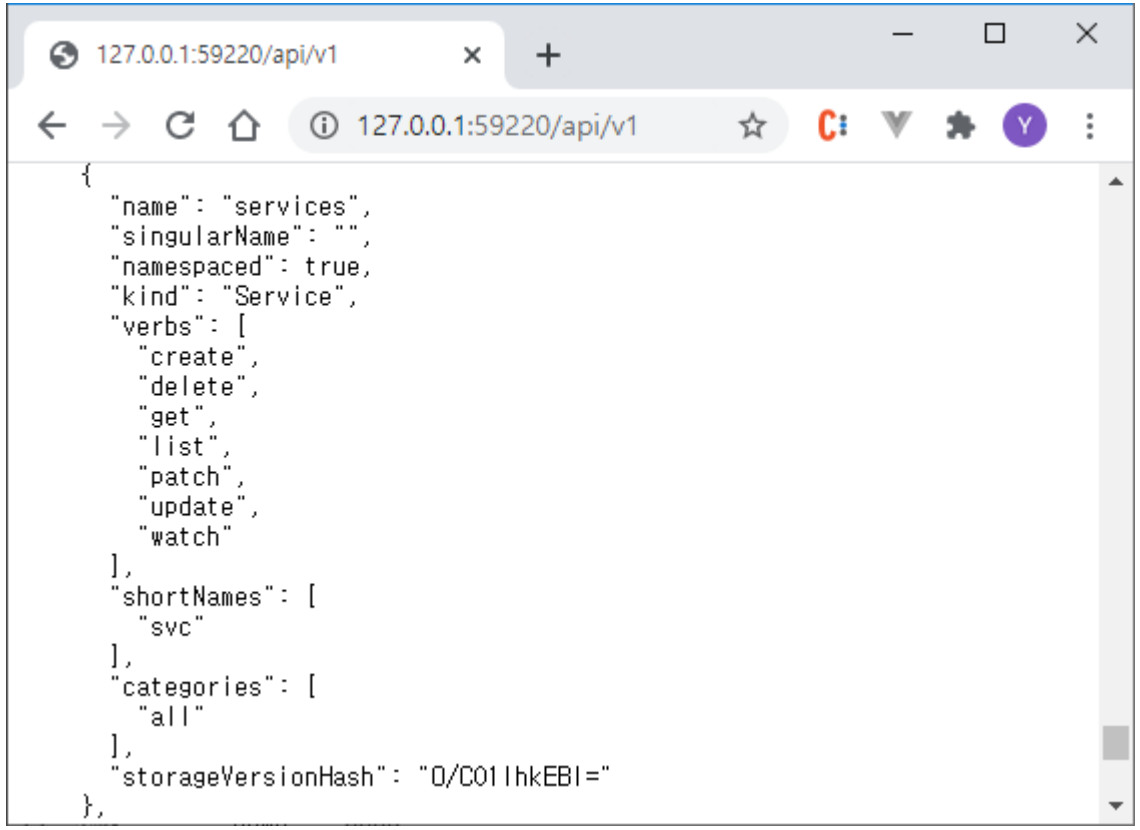
(kubectl expose deployment count --port=8080 --type=LoadBalancer와 동일함)



7. Kubernetes

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

- Kubernetes API



7. Kubernetes

- Kubernetes API

127.0.0.1:59220/apis


```
{
  "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"
  }
},
```


127.0.0.1:59220/apis/apps/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"
  ]
},
```

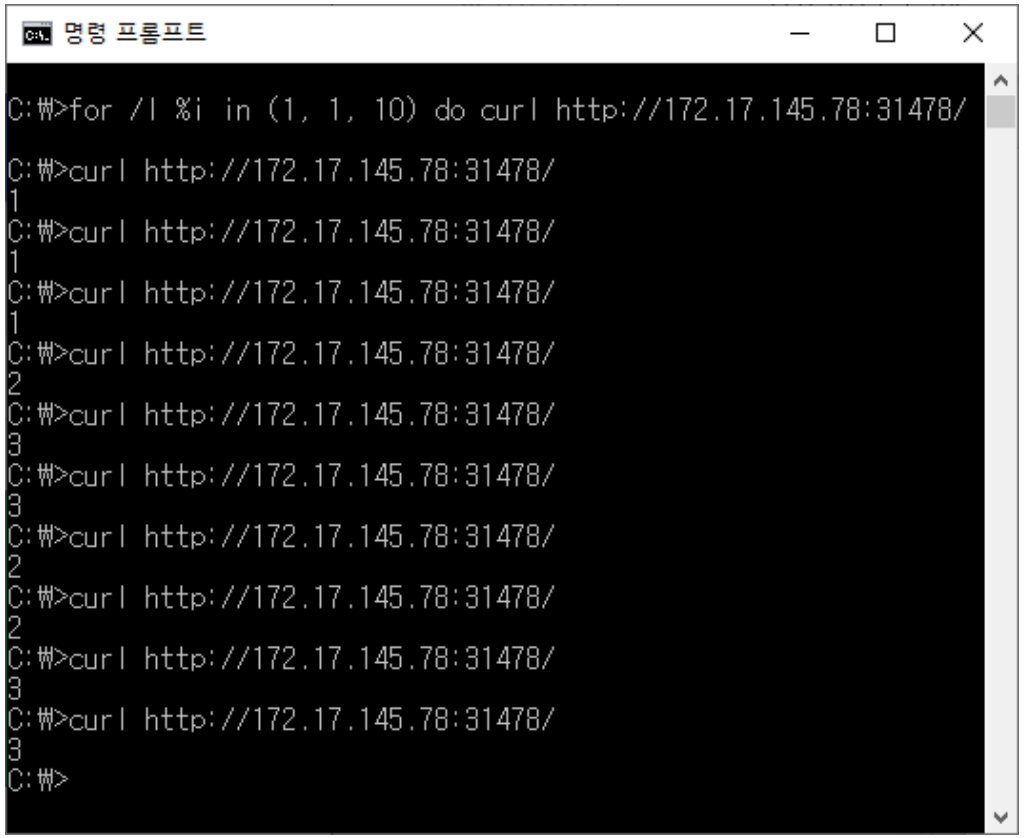
7. Kubernetes

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.



7. Kubernetes

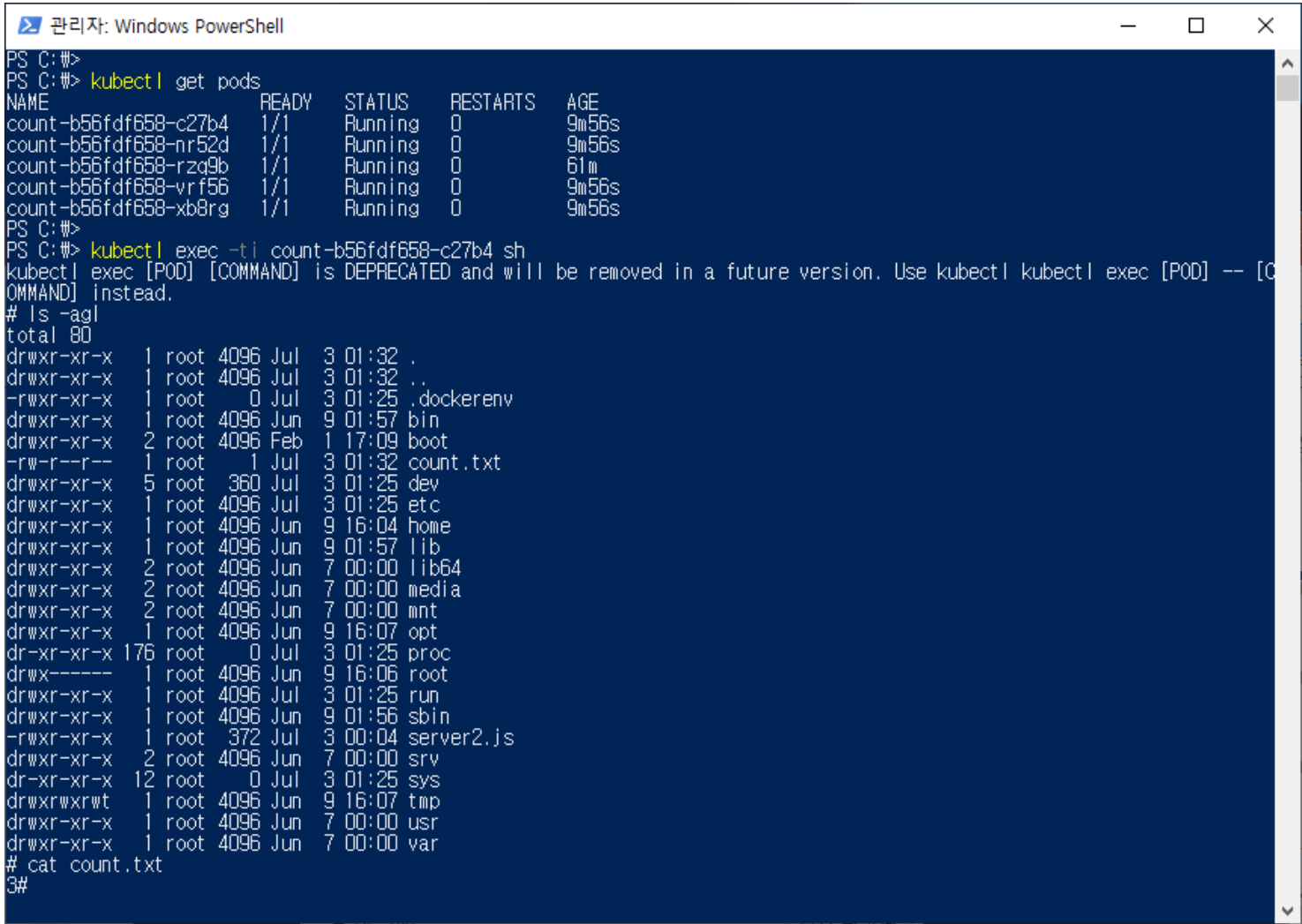
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

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

5) 서비스를 생성한다.

B) POD에 shell로 연결해 본다.

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



7. Kubernetes

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

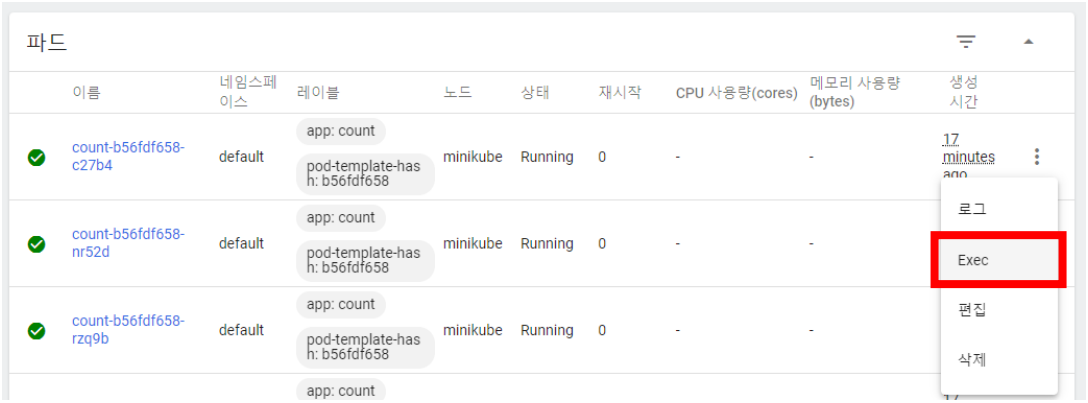
- 간단한 예제 - 2 (tutorials/kubernetes/server2.js)

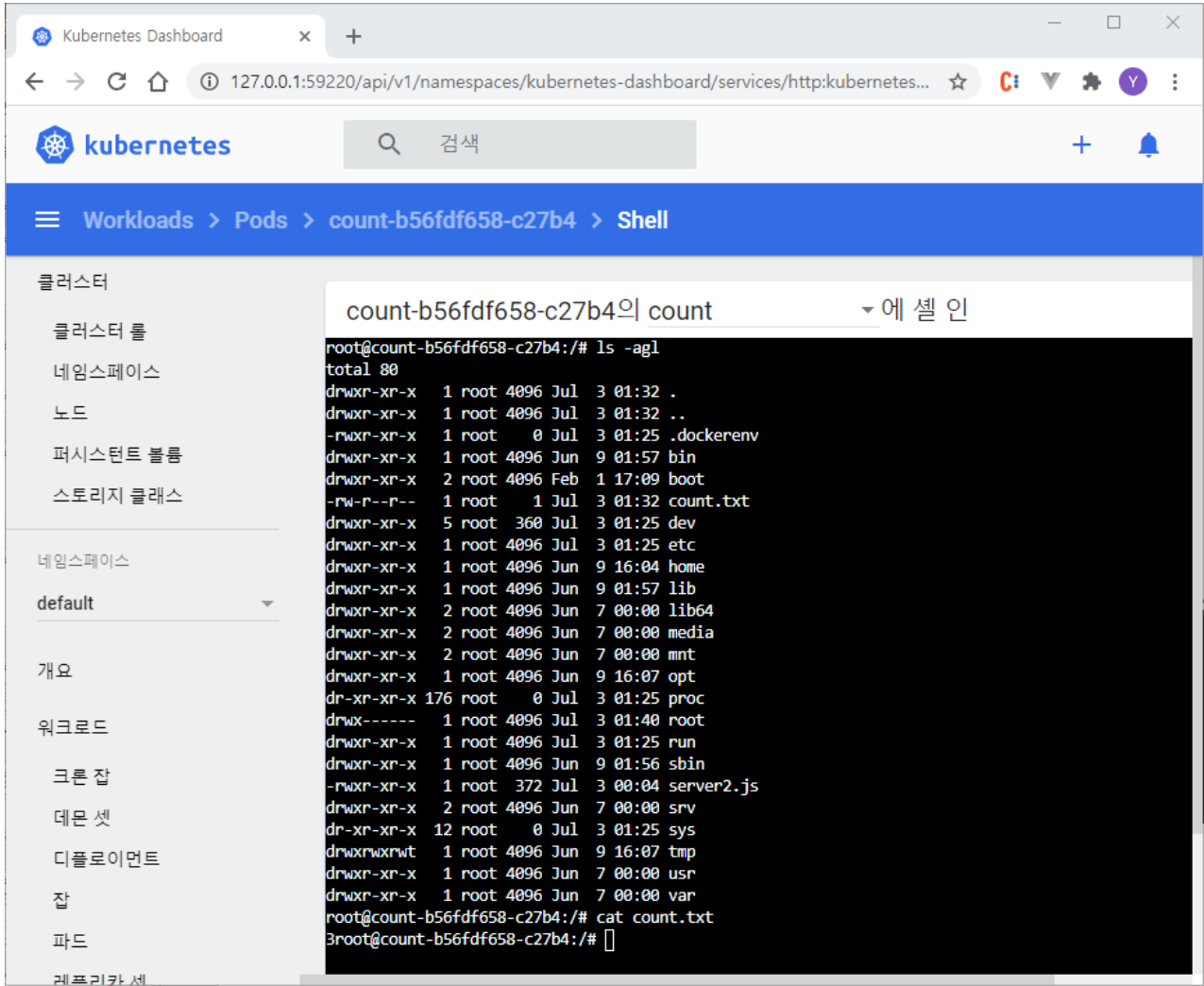
5) 서비스를 생성한다.

B) POD에 shell로 연결해 본다.

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

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





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
```

server2.js

```
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.

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.

count-54b88c7844-b5d7h의 count 에 셸 인

```
root@count-54b88c7844-b5d7h:/# mount
overlay on / type overlay (rw,relatime,lowerdir=/var/lib/docker/overlay2/l/SJ32AYGX32K34HQYJCMXPPLWQC:/var/lib/dock
N2DMEHLRVT:/var/lib/docker/overlay2/l/4EUSBOIQ72NVD3ZSP2MFR6DOXI:/var/lib/docker/overlay2/l/ZQ3PO3OZU5VJOB5OLOPALK
/QWWGZKJYSK3SUTMYIU4TJJZW7N:/var/lib/docker/overlay2/l/Y6AZV2SSV7HCGFK75Z65OAW540:/var/lib/docker/overlay2/l/TSFYF7
docker/overlay2/l/ZKS4MGMECWTQN4E5V4DLTKGFD3:/var/lib/docker/overlay2/l/AWH4CL3IFDFD4RGP7XGPQGLJUT:/var/lib/docker/
HEWLCLV:/var/lib/docker/overlay2/l/LGI5ETMVYLHBMWQ4KNIOUGRLK7,upperdir=/var/lib/docker/overlay2/ae2fa411adbc0aa4ecec
f708049c94f1f61/diff,workdir=/var/lib/docker/overlay2/ae2fa411adbc0aa4ecec92c54b5999bbc9dcffd6e16703ae0f708049c94f1
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/cgroup 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/
temd)
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 (rw,nosuid,nodev,noexec,relatime)
/dev/sda1 on /data 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/asound 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 에 셸 인

```
root@count-54b88c7844-b5d7h:/# cd data
root@count-54b88c7844-b5d7h:/data# ls -agl
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#
```

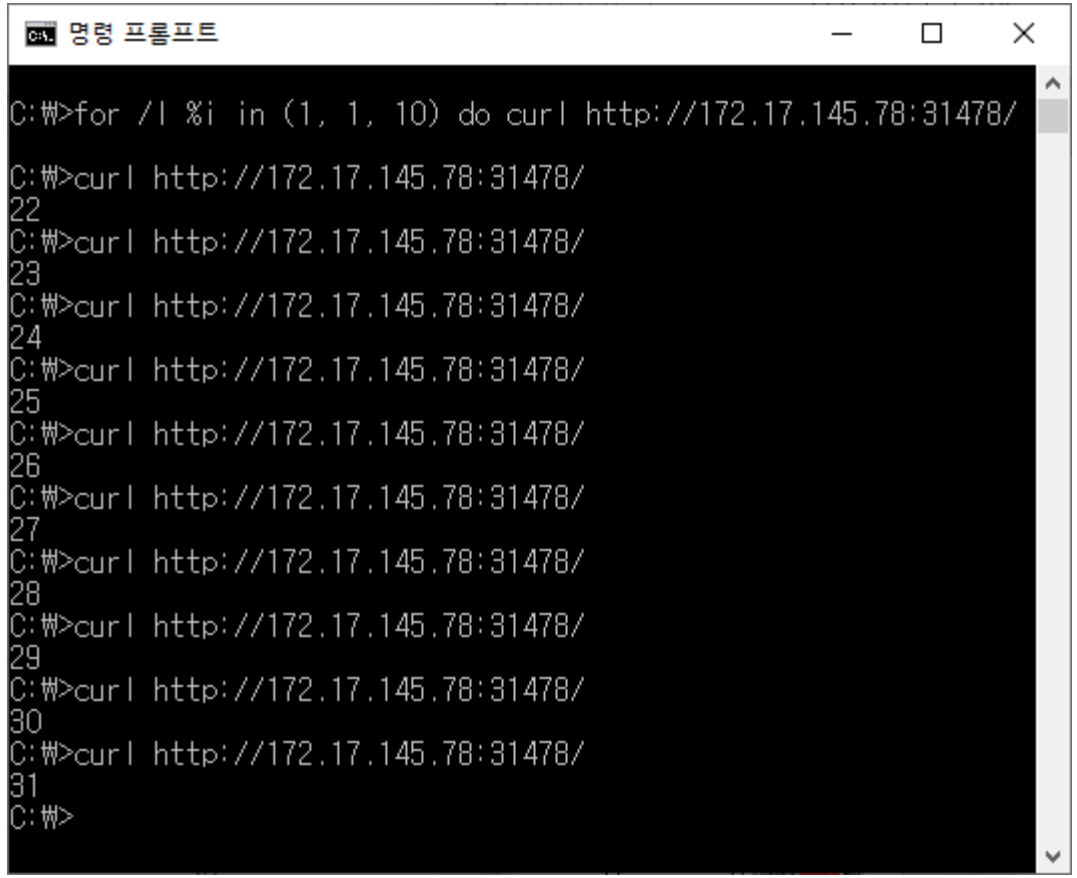
7. Kubernetes

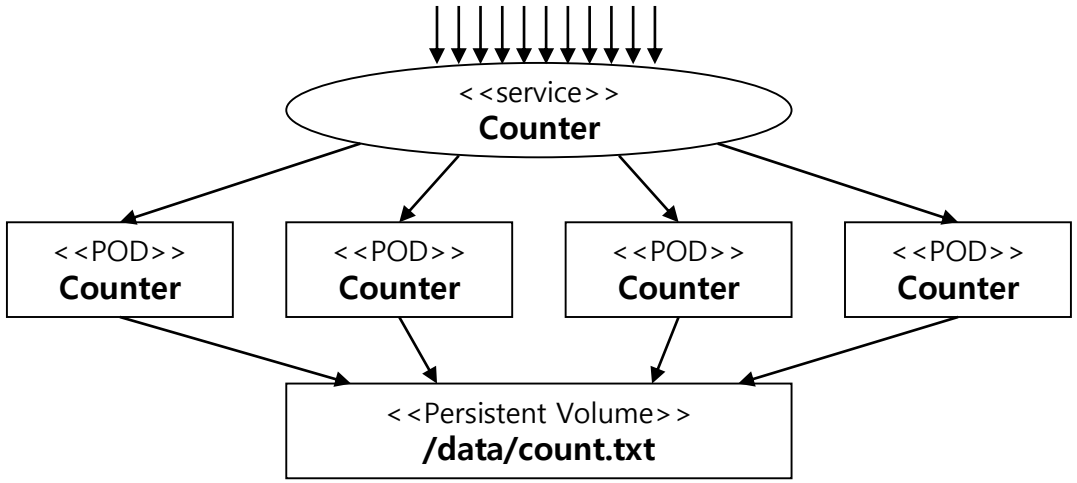
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

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

6) Persistent Volume을 생성한다.

B) POD의 개수를 늘리고 서비스에 연결해 본다.





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)

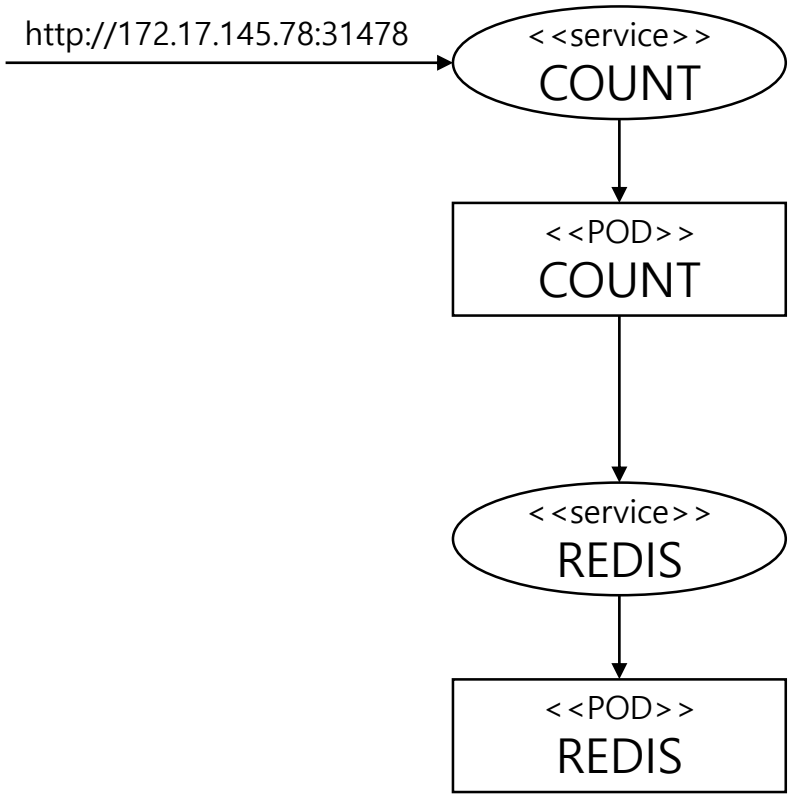
1) Redis DB 서비스를 생성한다.

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
```



7. Kubernetes

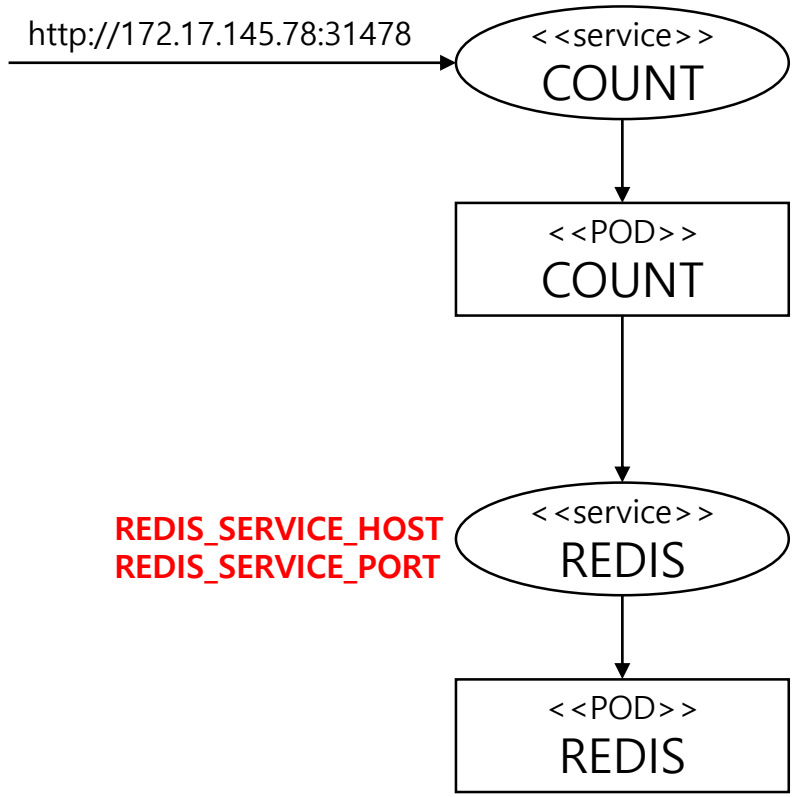
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

- 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

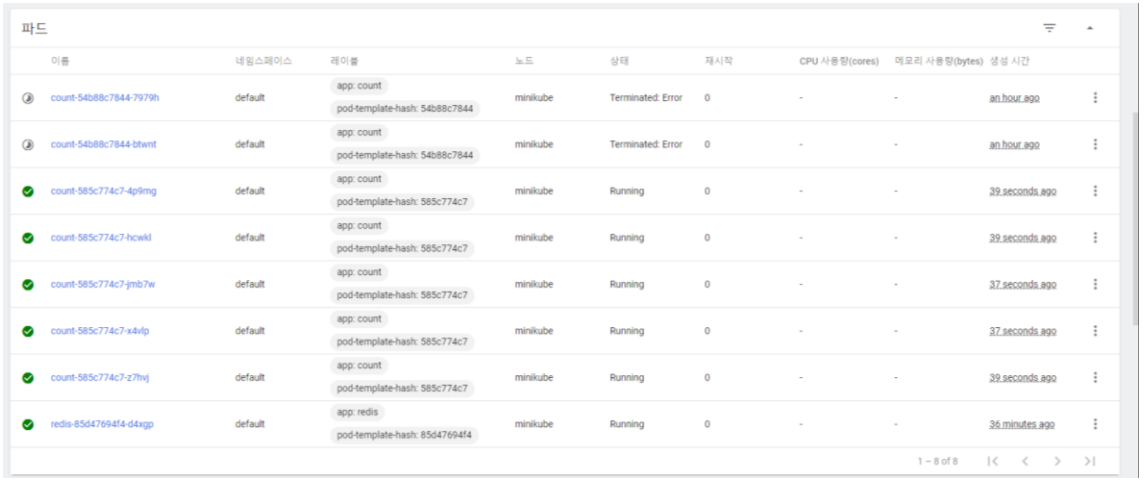
```
{
  "name": "count",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node server3.js"
  },
  "dependencies": {
    "redis": "latest"
  }
}
```

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.



새로운 POD 들이 생성되고, 기존 POD들이 종료된다.

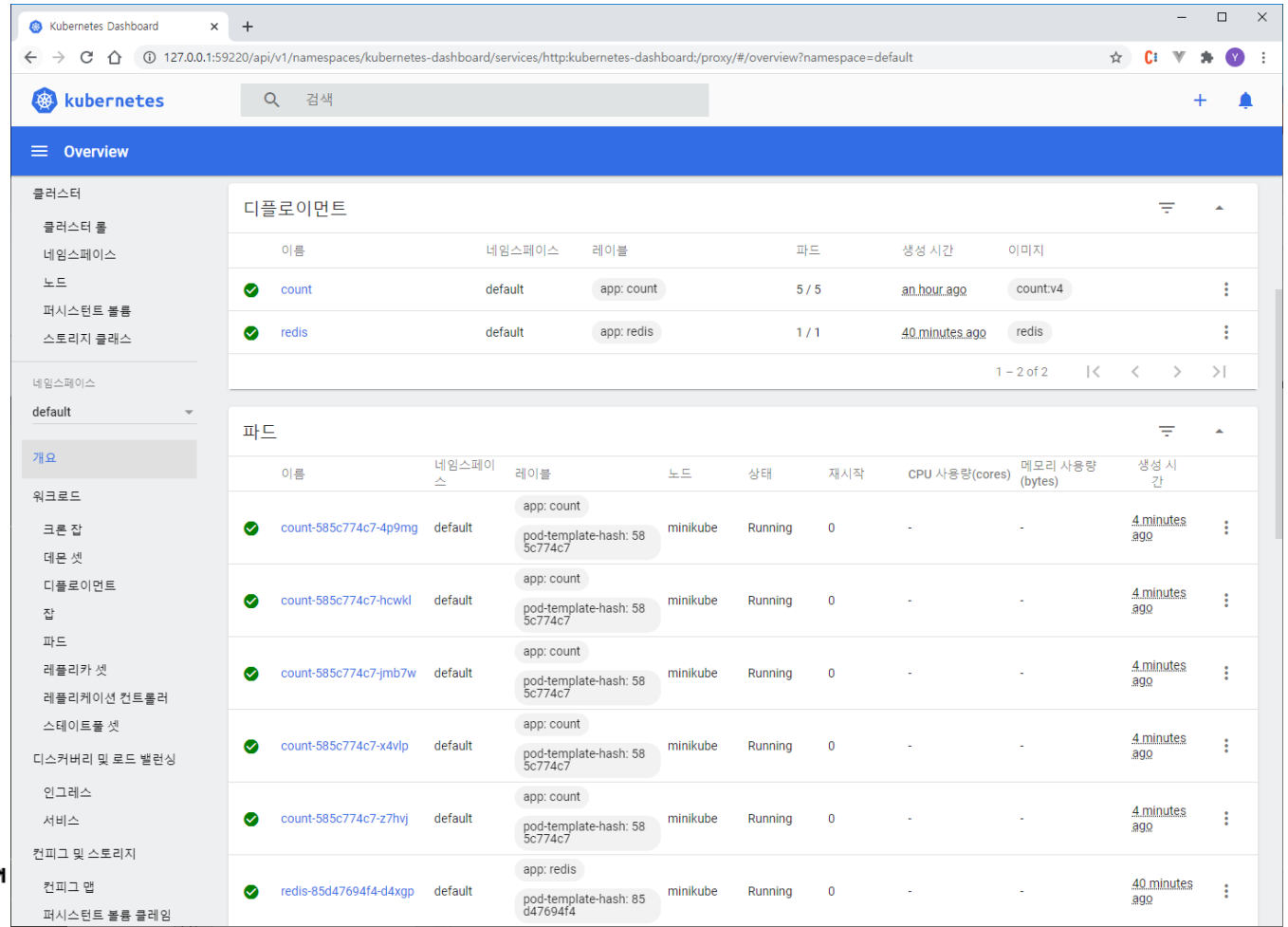
계속 서비스는 제공된다.

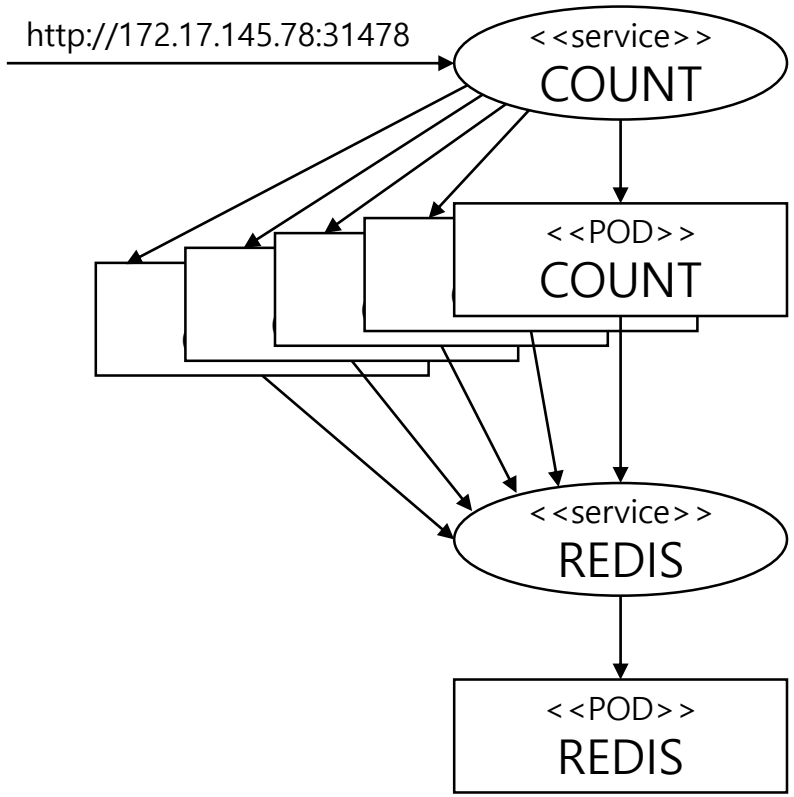
7. Kubernetes

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

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

2) Redis DB를 사용한다.



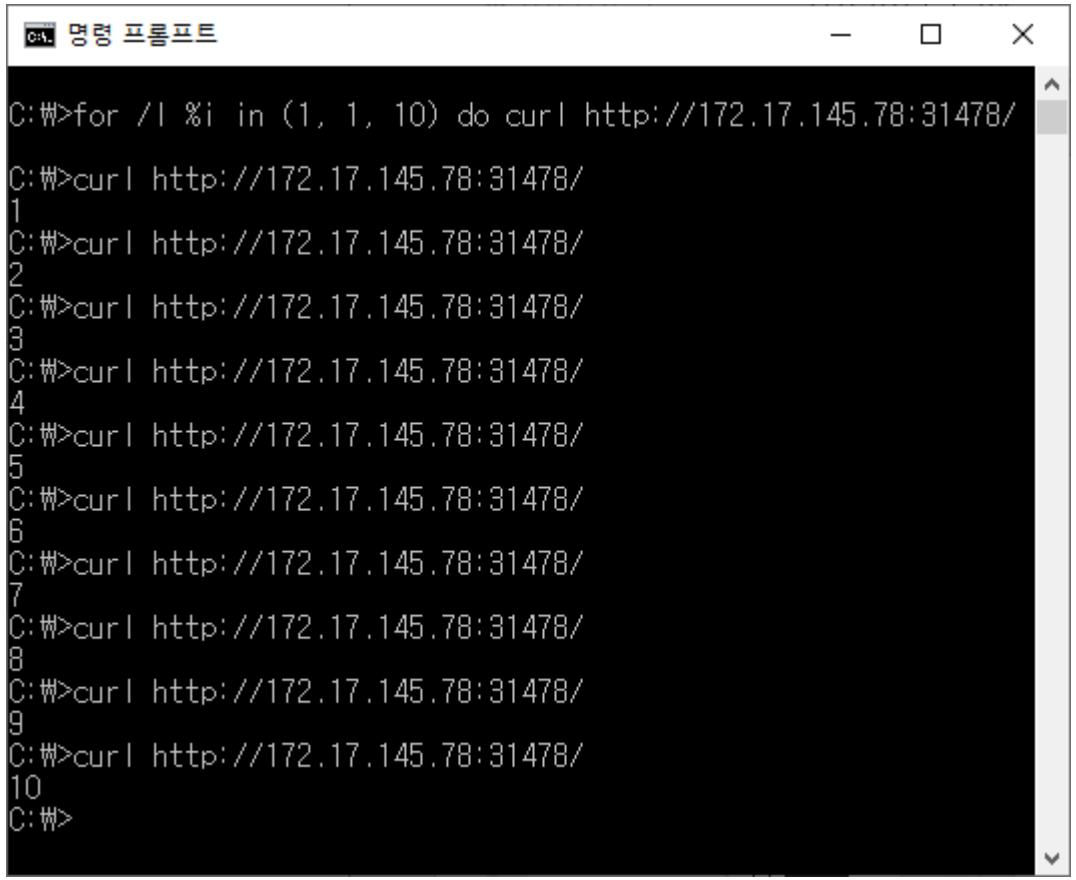


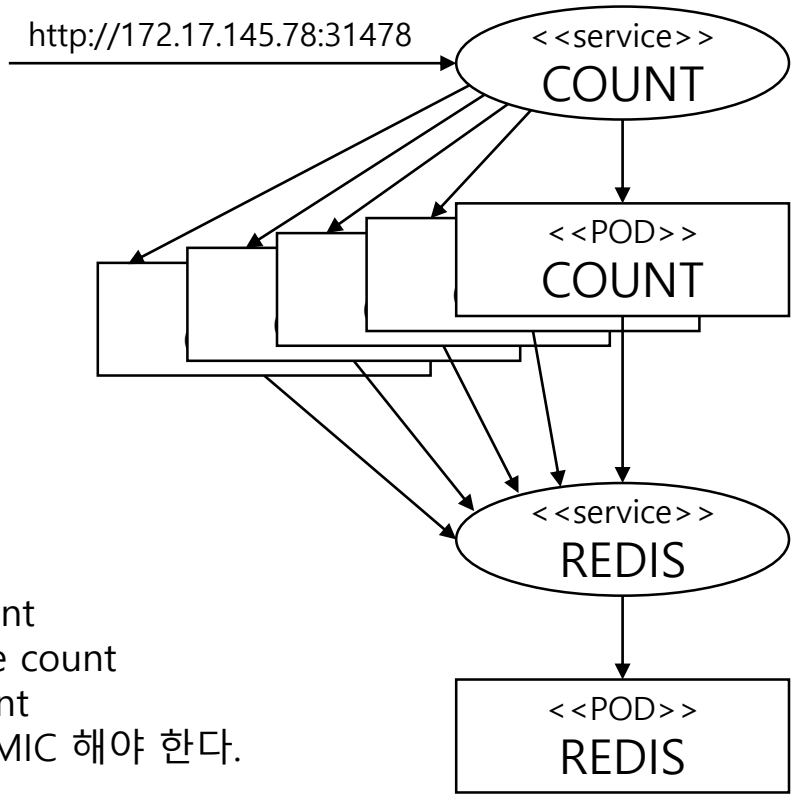
7. Kubernetes

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

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

  2) Redis DB를 사용한다.





get count
increase count
set count
가 ATOMIC 해야 한다.

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/

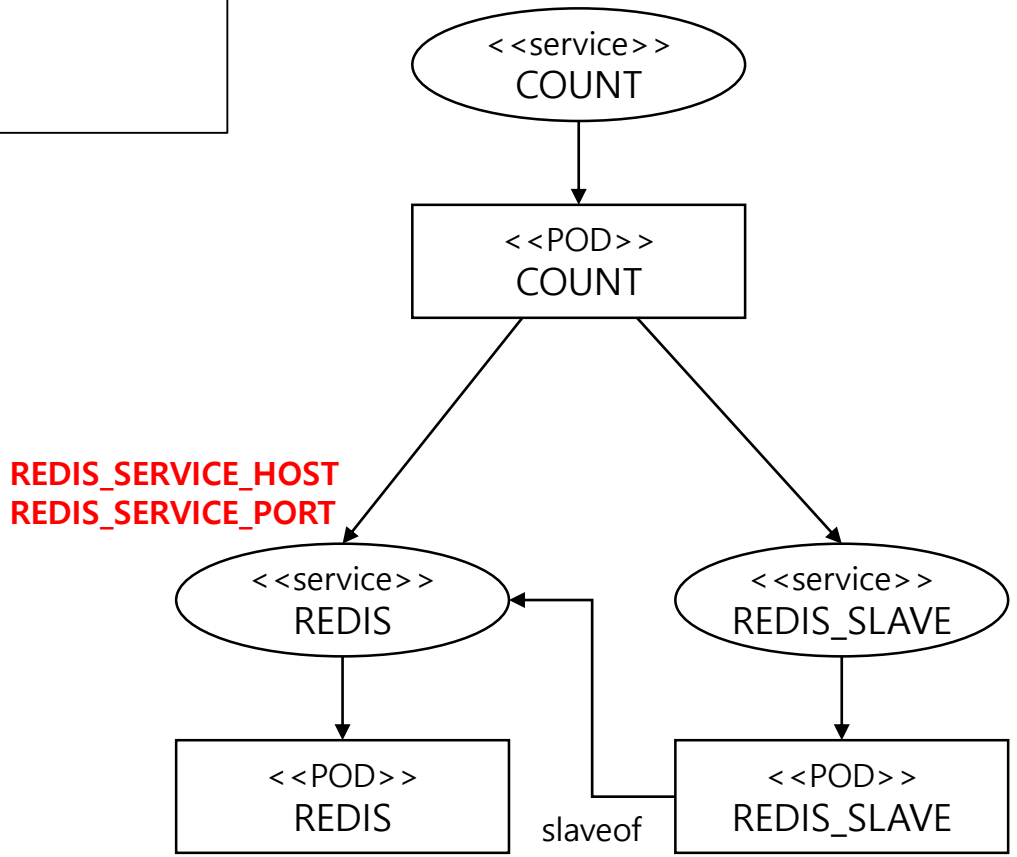
- 간단한 예제 - 3 (tutorials/kubernetes/server3.js)

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)

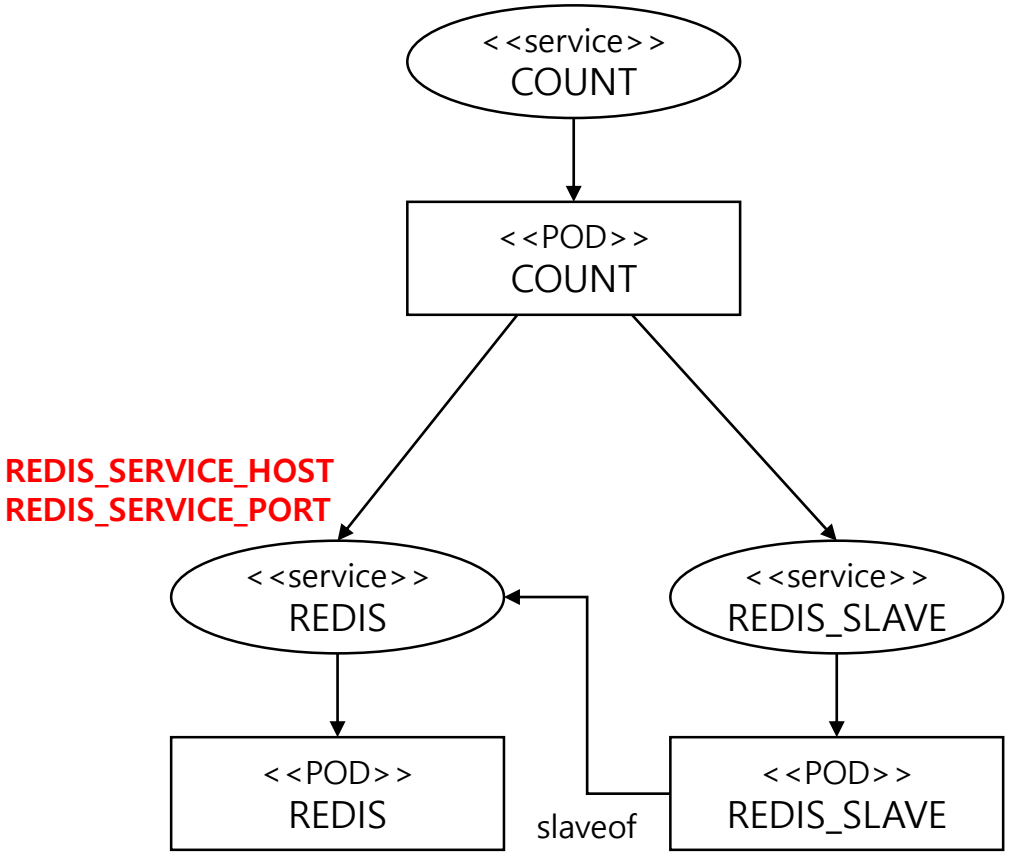
3) Redis master-slave DB를 구성해 본다.

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
```



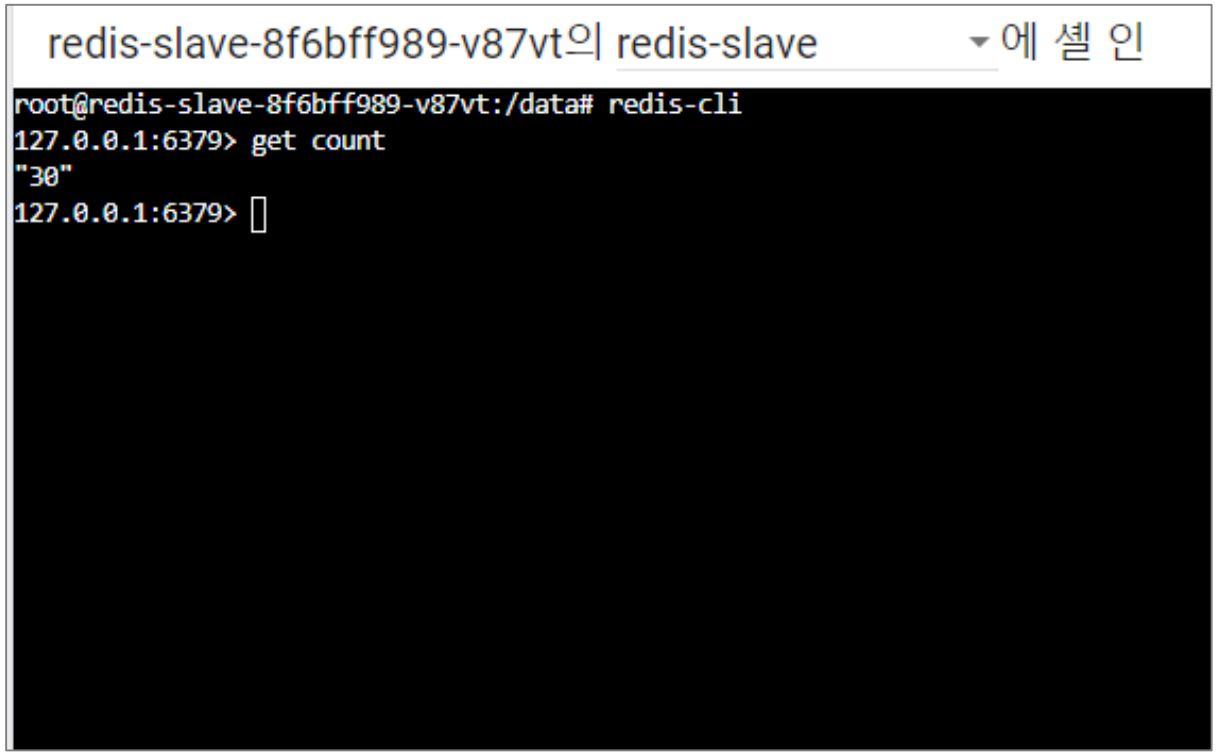
7. Kubernetes

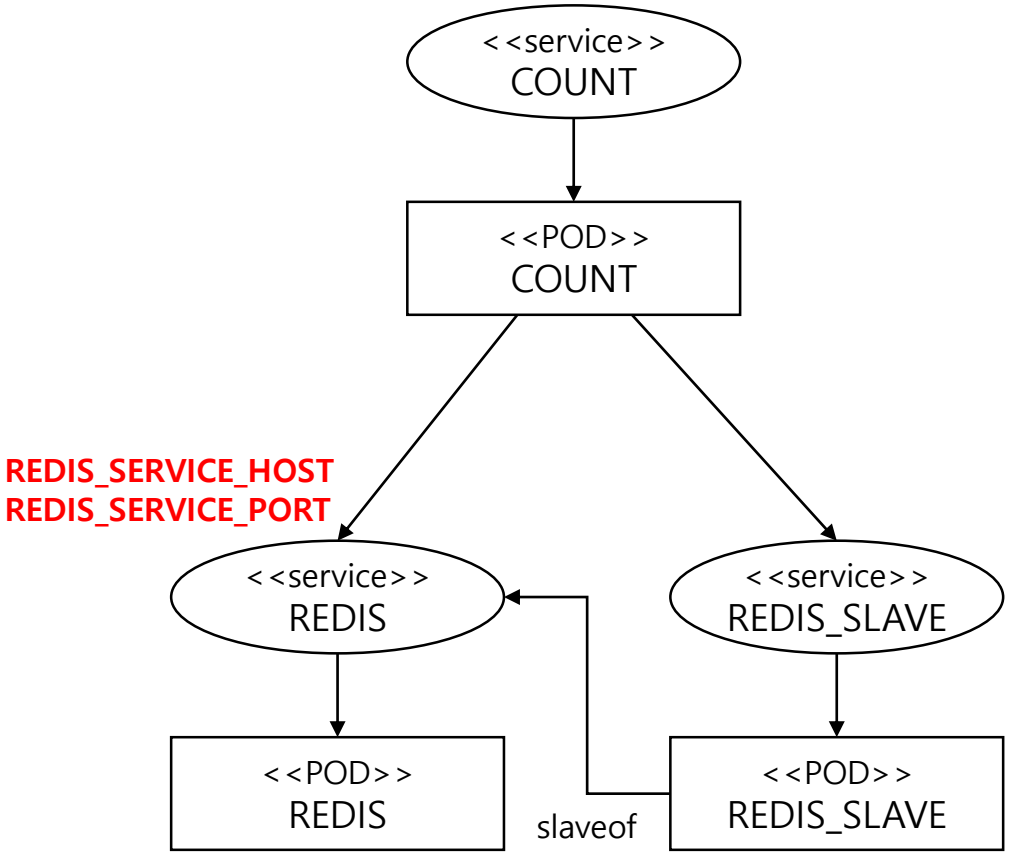
https://kubernetes.io/ko/docs/tutorials/kubernetes-basics/

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

3) Redis master-slave DB를 구성해 본다.

redis-slave POD에서 redis-cli로 master DB와 SYCN되었는 지를 확인한다.





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.

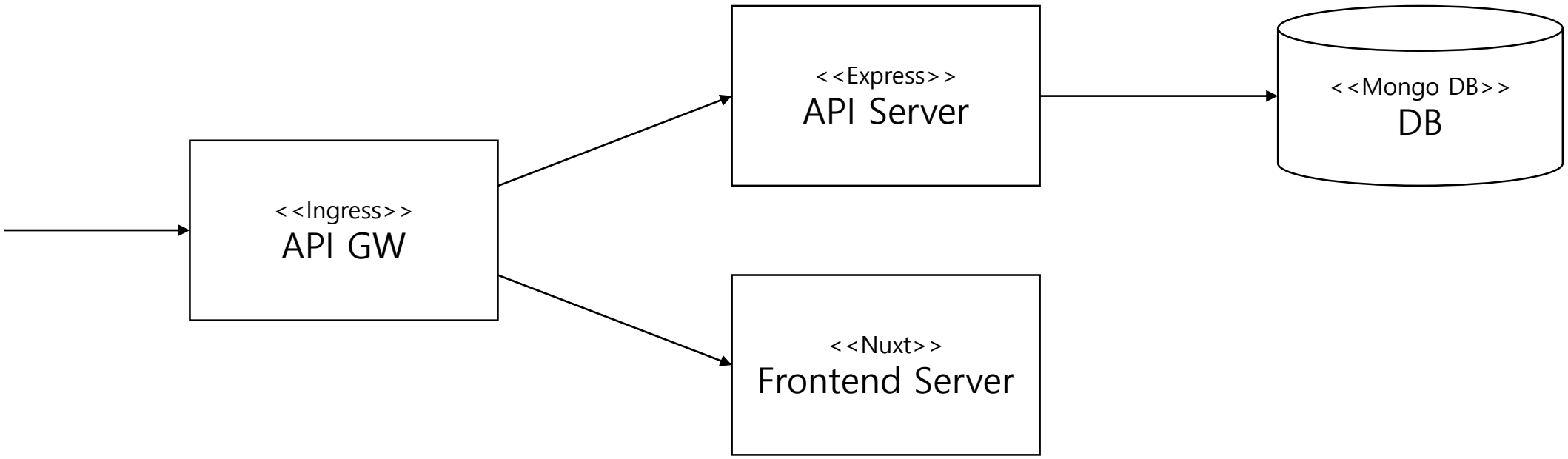
redis-slave POD도 5개로 증설한다.

curl 또는 브라우저로 동작을 확인한다.

8. Map UI

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

- 서비스 구조

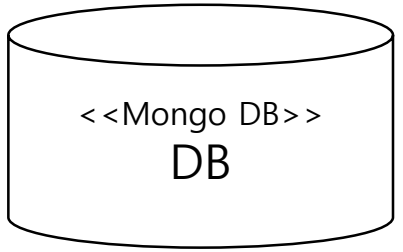


8. Map UI

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
```



Persistent Volume을 생성해서,

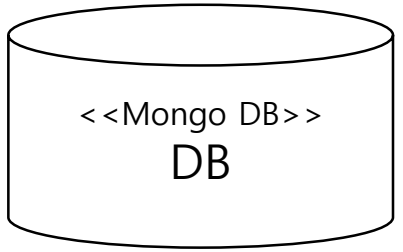
/data/db에 마운트 한다.

8. Map UI

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

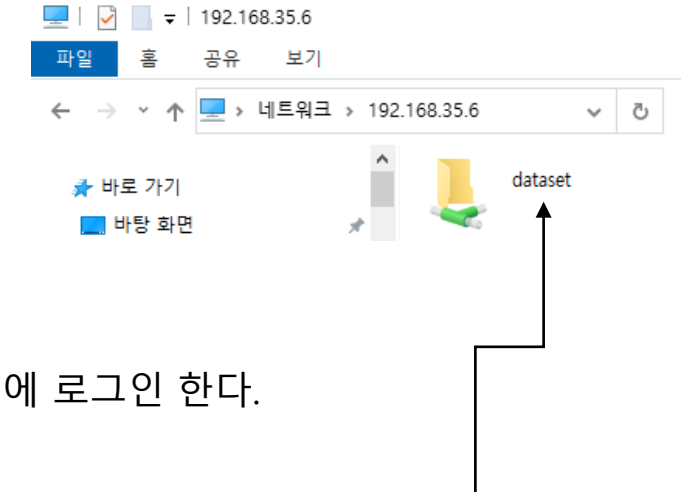
1) Mongo DB (db/db_deployment.yaml)

```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
```



Persistent Volume을 생성해서,

/data/db에 마운트 한다.



Data를 import하기 위해서, host를 mount 한다.

Hyper-V에서는 다음과 같이 공유한다.

1) 폴더를 공유한다.

2) minikube ssh 명령으로 minikube 가상 머신에 로그인 한다.

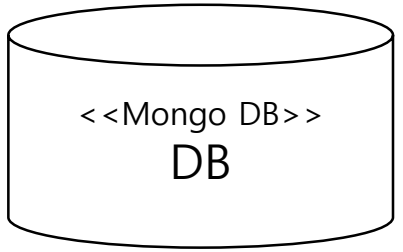
$ sudo mkdir **/dataset**

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

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**

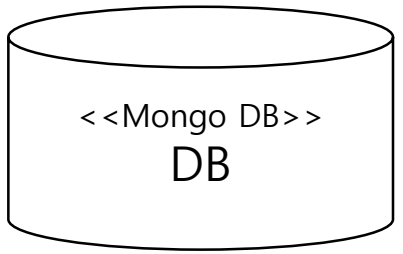
mongo-db-67b477f6c8-grr95의 mongo-db 에 셸 인

```
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
-rwxr-xr-x 1 root  146321327 Jun 18 11:13 yelp_academic_dataset_business.json
-rwxr-xr-x 1 root  449663480 Dec 13  2019 yelp_academic_dataset_checkin.json
-rwxr-xr-x 1 root   89091484 Jun 18 14:30 yelp_academic_dataset_geodata.json
-rwxr-xr-x 1 root   25615185 Jan  7 22:12 yelp_academic_dataset_photo.json
-rwxr-xr-x 1 root 6325565224 Dec 13  2019 yelp_academic_dataset_review.json
-rwxr-xr-x 1 root  263489322 Dec 13  2019 yelp_academic_dataset_tip.json
-rwxr-xr-x 1 root 4176861509 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_busine
ss.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 생한다.

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)

<<Express>>
API Server

```
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);
```

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)

<<Express>>
API Server

```
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;
```

8. Map UI

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

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

<<Express>>
API Server

```
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);
  });
});
```

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)

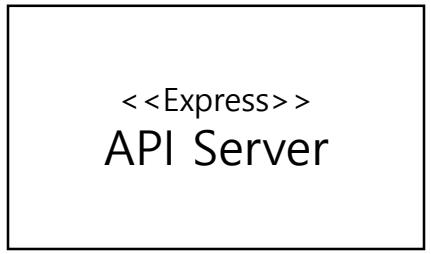
```
{
  "name": "server",
  "version": "0.0.1",
  "description": "DB API SERVER",
  "author": "Yong Jin, Cho <drajin.cho@bosornd.com>",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "latest",
    "mongoose": "latest"
  }
}
```

<<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
```

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.

<<Express>>
API Server

```yaml
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'
```

```yaml
---
apiVersion: v1

kind: Service
metadata:
  name: api-server
  labels:
    svc: api-server

spec:
  ports:
    - port: 8080
  selector:
    svc: api-server
```

8. Map UI

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

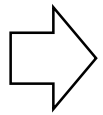
4) Frontend Server (frontend/frontend_deployment.yaml)

<<Nuxt>>
Frontend Server

npx create-nuxt-app frontend로 프로젝트를 생성한다.

layouts/default.vue nuxt 부분을 다음과 같이 수정한다.

```
<v-content>
  <v-container>
    <nuxt />
  </v-container>
</v-content>
```



```
<v-main>
  <nuxt />
</v-main>
```

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.

<<Nuxt>>
Frontend Server

```
<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
    });
```

8. Map UI

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

4) Frontend Server (frontend/frontend_deployment.yaml)

pages/index.vue를 다음과 같이 수정한다.

<<Nuxt>>
Frontend Server

```
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 } ));
    }
  }
});
```


minikube ip를 hosts에 등록한다(C:₩Windows₩System32₩drivers₩etc₩hosts).

8. Map UI

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

4) Frontend Server (frontend/frontend_deployment.yaml)

pages/index.vue를 다음과 같이 수정한다.

<<Nuxt>>
Frontend Server

```
    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>
```

8. Map UI

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

4) Frontend Server (frontend/frontend_deployment.yaml)

<<Nuxt>>
Frontend Server

package.json 은 다음과 같다.

```
{
  "name": "frontend",
  "version": "1.0.0",
  "private": true,
  "config": {
    "nuxt": {
      "host": "0.0.0.0",
      "port": "8080"
    }
  },
  "scripts": {
    "dev": "nuxt",
    "build": "nuxt build",
    "start": "nuxt start",
    "export": "nuxt export",
    "serve": "nuxt serve"
  },
  "dependencies": {
    "nuxt": "^2.13.0"
  },
  "devDependencies": {
    "@nuxtjs/vuetify": "^1.11.2"
  }
}
```

"host": "0.0.0.0", → 설정안하면 localhost만 접근 가능함.
"port": "8080" → 포트

"dev": "nuxt", → 개발 버전 실행
"build": "nuxt build", → 서비스 버전 빌드
"start": "nuxt start", → 서비스 버전 실행

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.

<<Nuxt>>
Frontend Server

```yaml
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'
```

```yaml
---
apiVersion: v1

kind: Service
metadata:
  name: frontend-server
  labels:
    app: frontend-server

spec:
  ports:
    - port: 8080
  selector:
    app: 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)

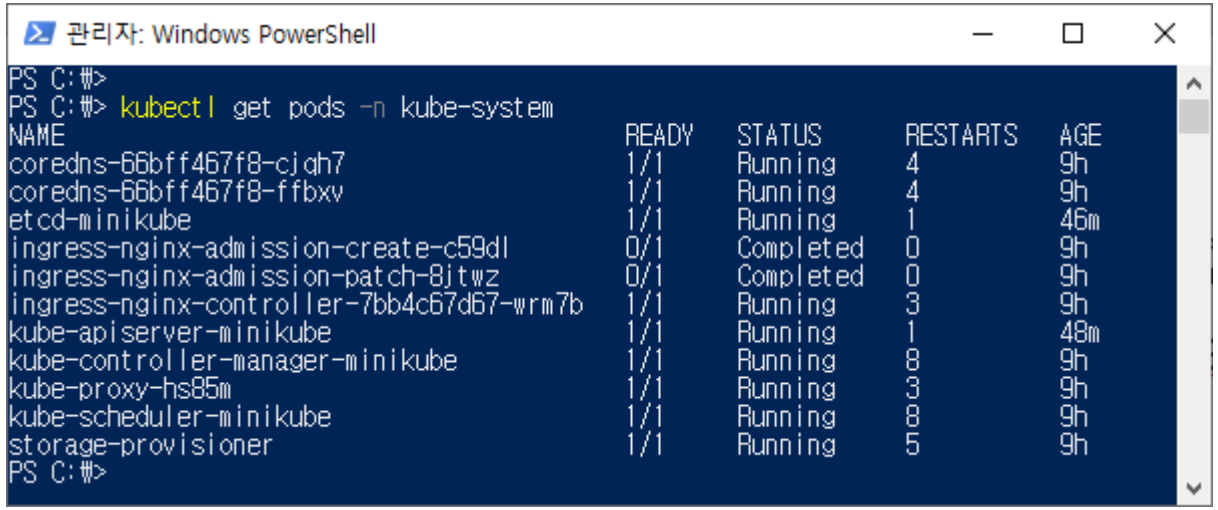
FEATURE STATE: `Kubernetes v1.1 [beta]`

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.



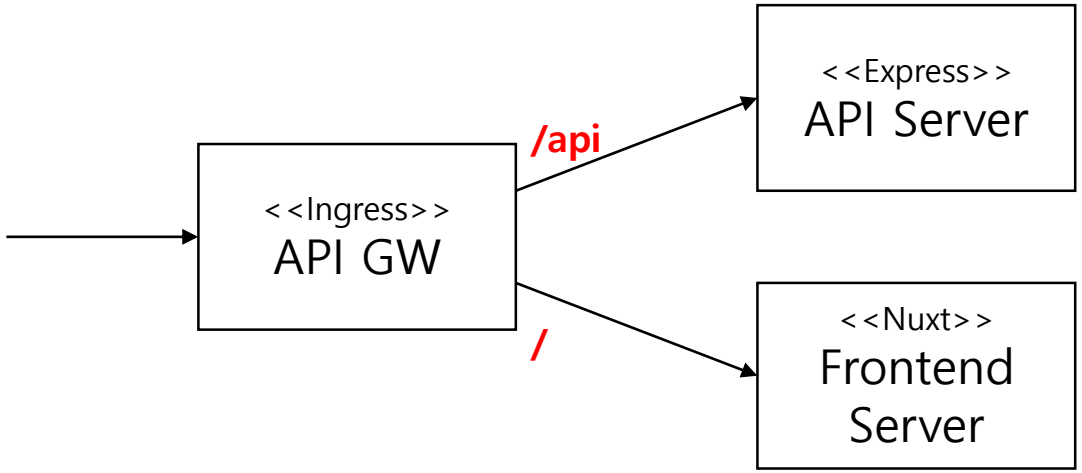
8. Map UI

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)

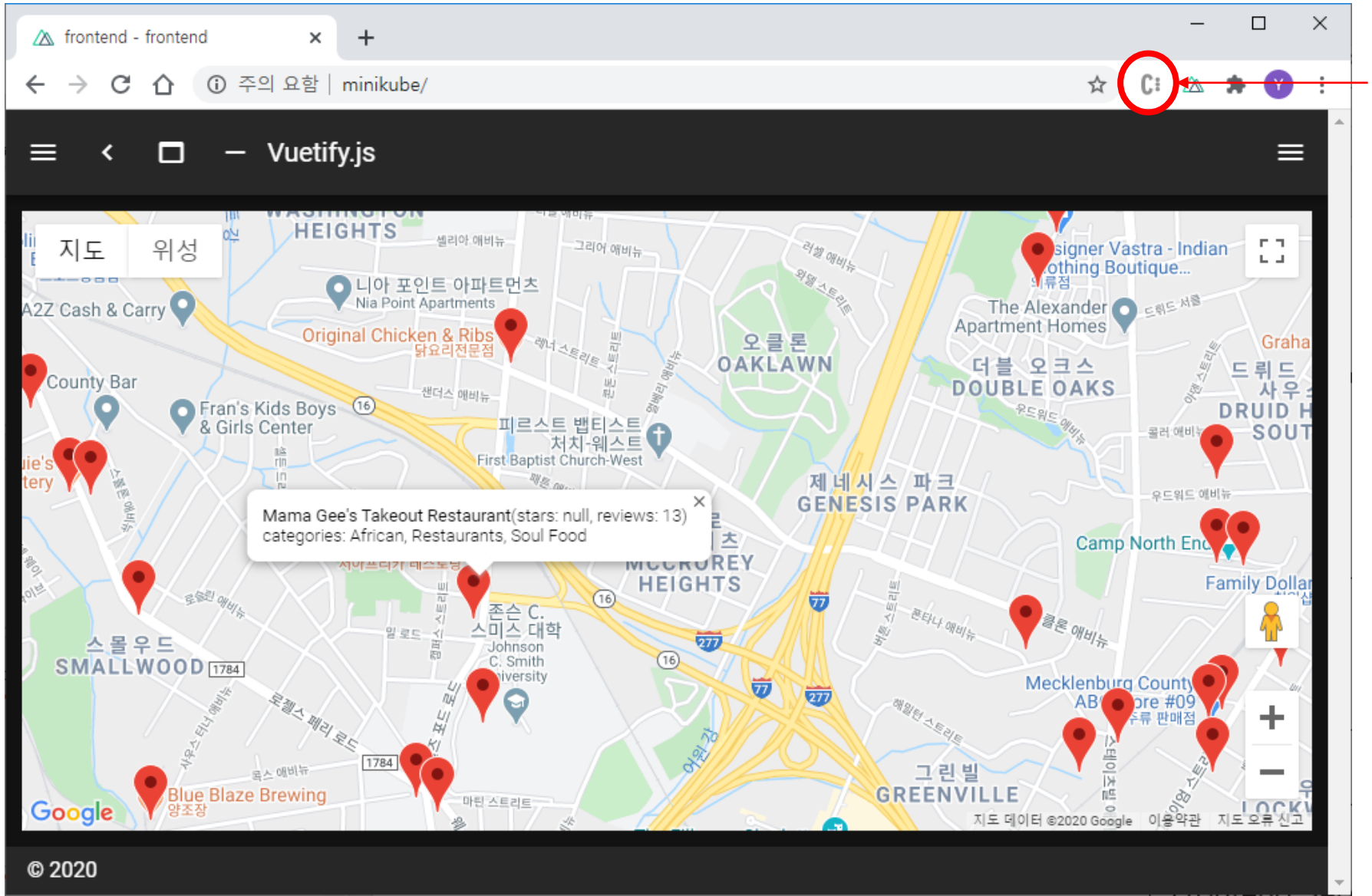
kubectl apply –f ingress_deployment.yaml로 인그레스 정책을 설정한다.

```
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

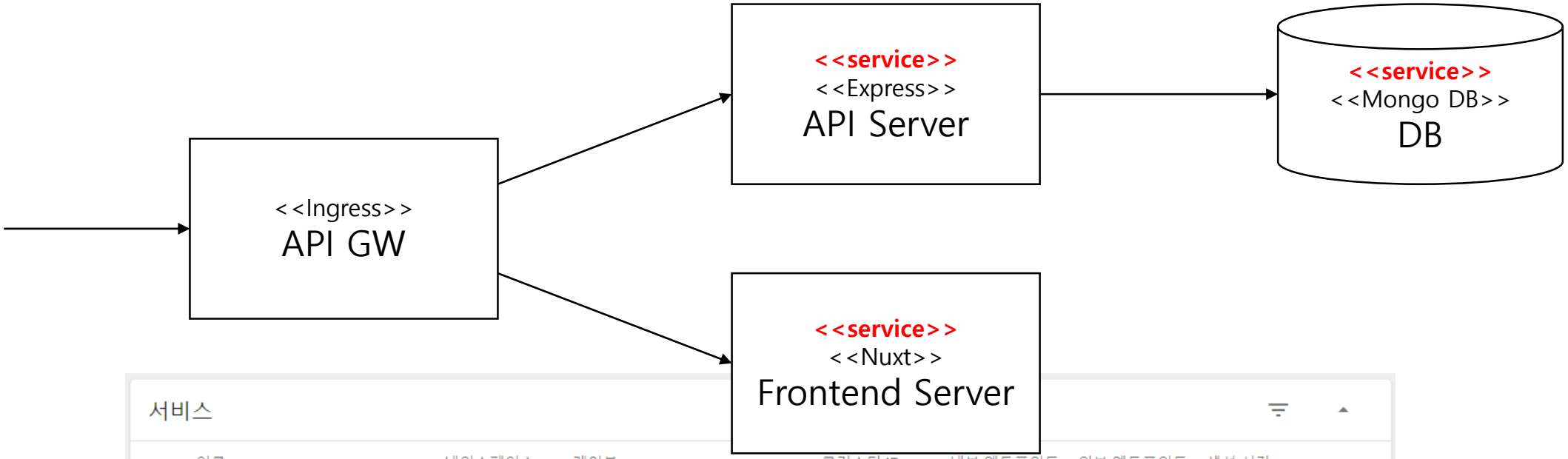
https://github.com/bosornd/map-ui



CORS 문제가 없음.

8. Map UI

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



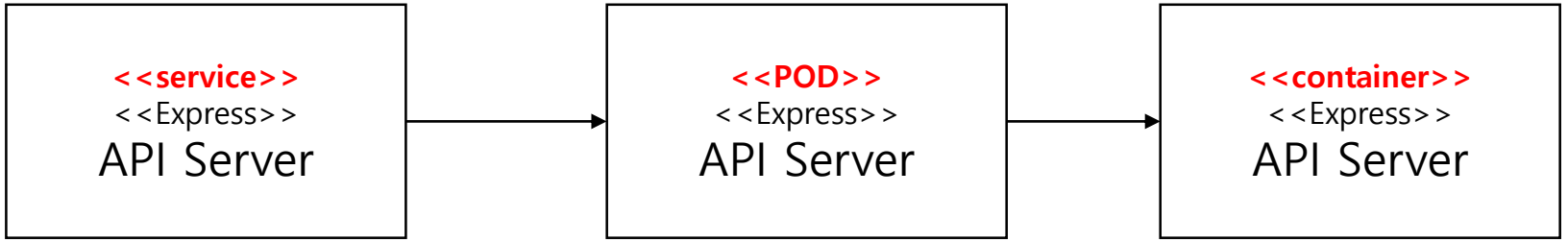
서비스

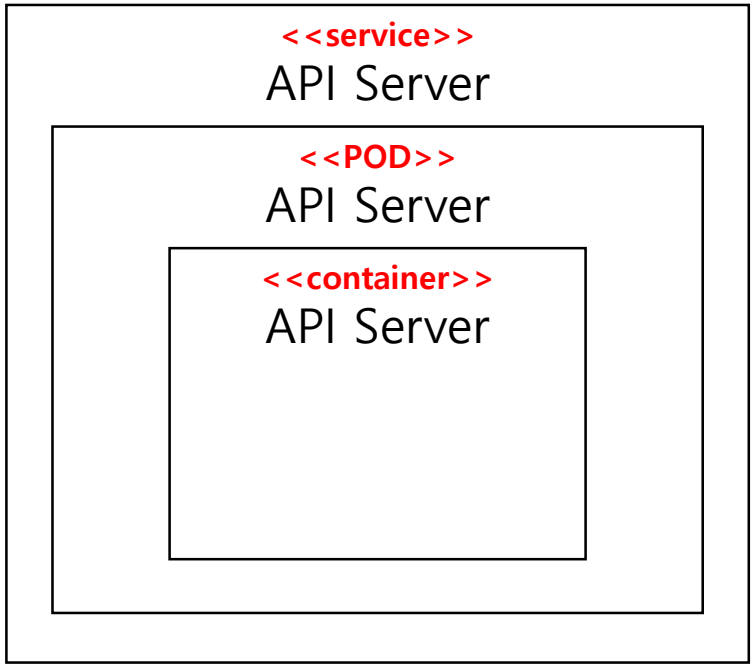
이름	네임스페이스	레이블	클러스터 IP	내부 엔드포인트	외부 엔드포인트	생성 시간
api-server	default	svc: api-server	10.104.61.72	api-server:8080 TCP api-server:0 TCP	-	10 minutes ago
frontend-server	default	app: frontend-server	10.102.146.244	frontend-server:8080 TCP frontend-server:0 TCP	-	8 minutes ago
kubernetes	default	component: 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

1 – 4 of 4

8. Map UI

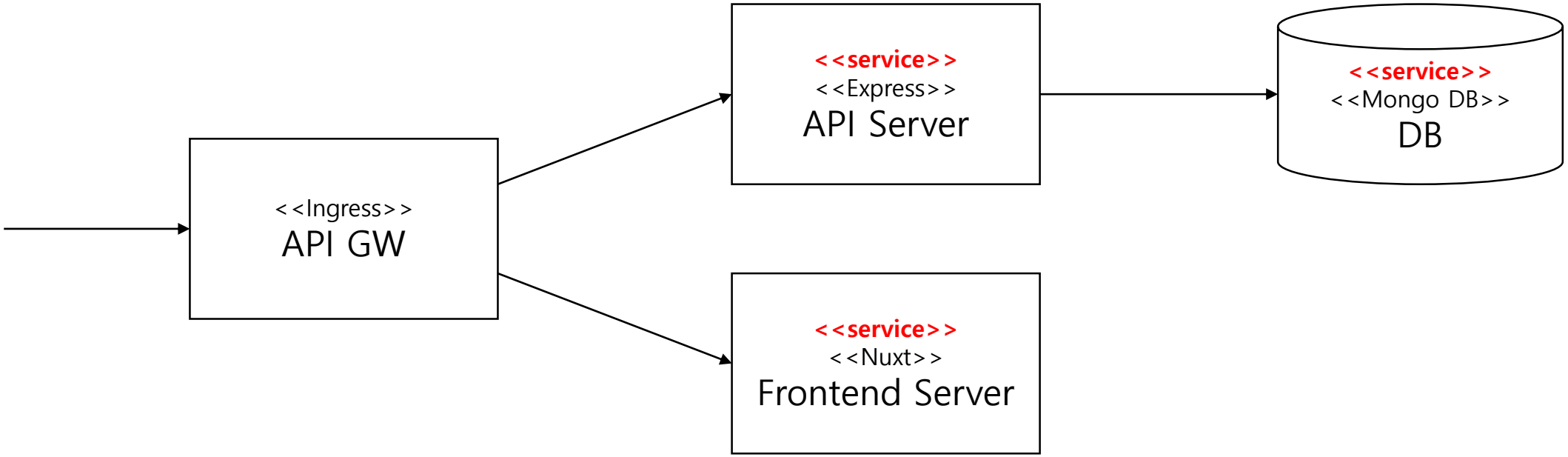
https://github.com/bosornd/map-ui





8. Map UI

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

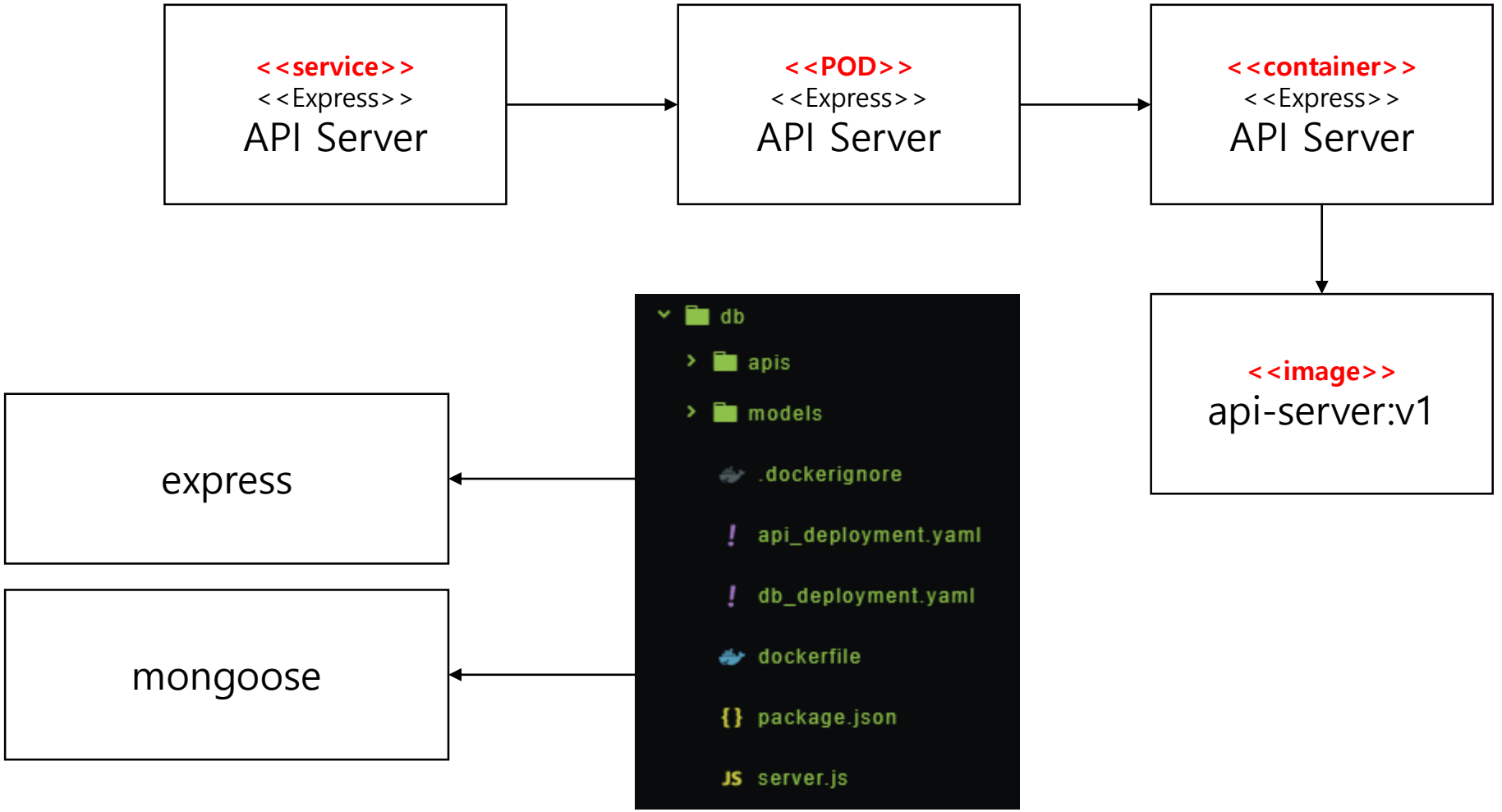


관리자: Windows PowerShell

```
PS C:₩>
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
jettech/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

- null != 0 ➔ true
- undefined != 0 ➔ true
- null == undefined ➔ true
- null !== undefined ➔ true

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

변수의 범위

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