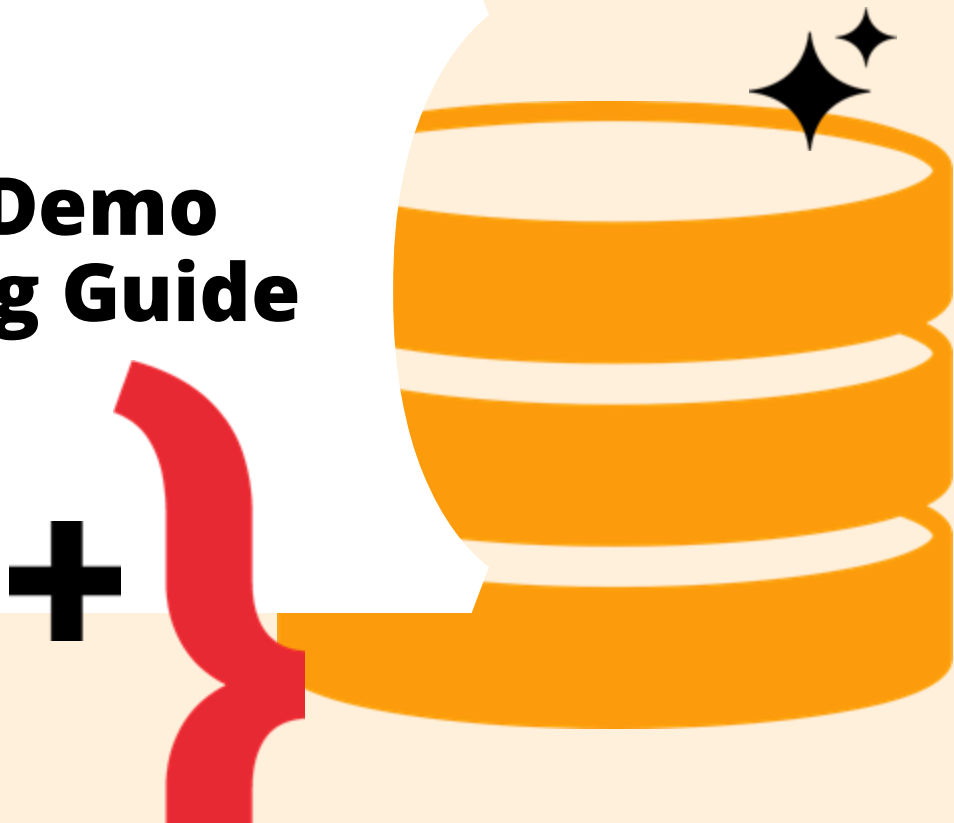


Couchbase Mobile Demo - Survey App Setting Guide

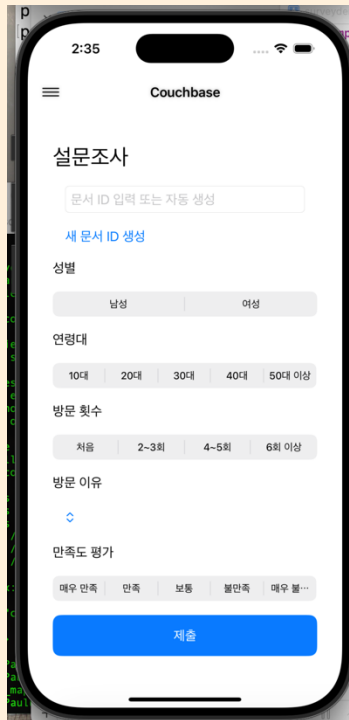
Jongho Park, Cloud Architect

Nov 2024



Agenda

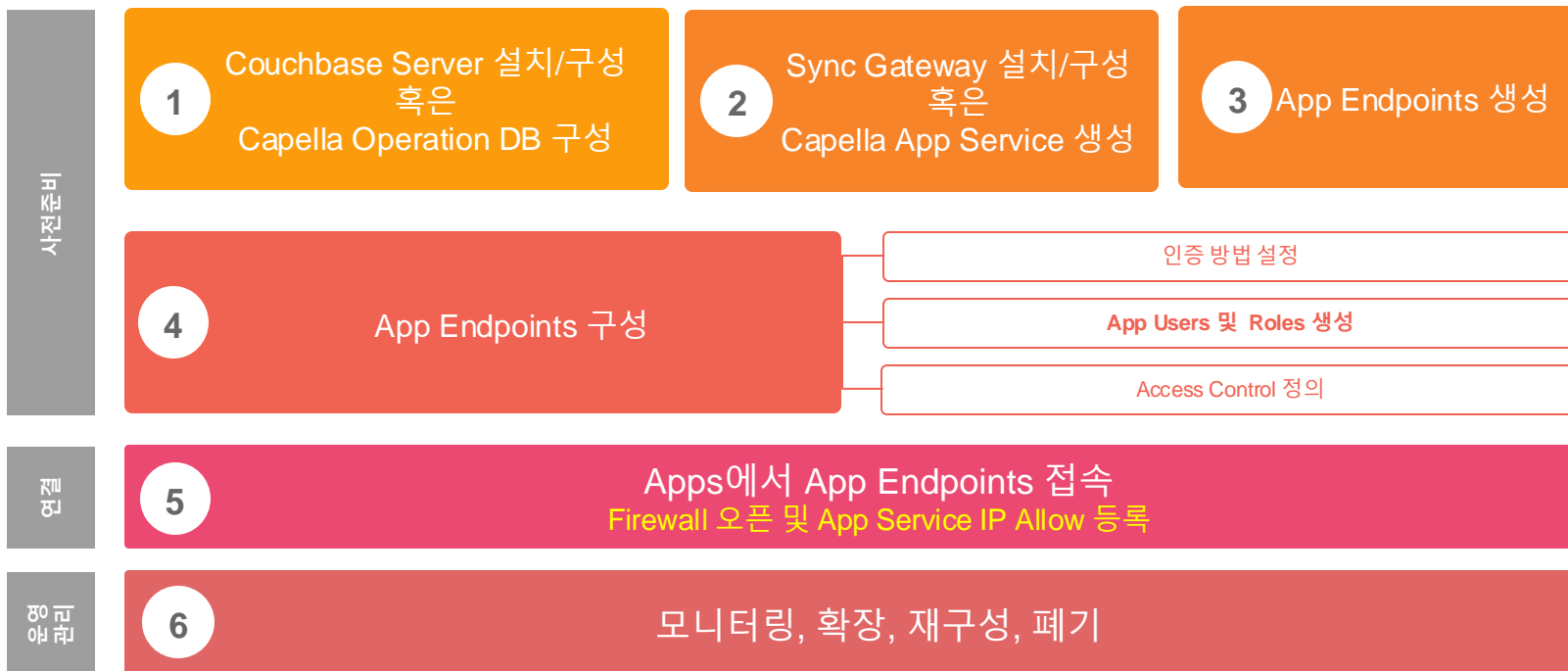
1. Couchbase Mobile 환경
2. 구축형 Sync Gateway 구성
3. 완전관리형 Capella APP Service 구성
4. Survey Demo App 실행



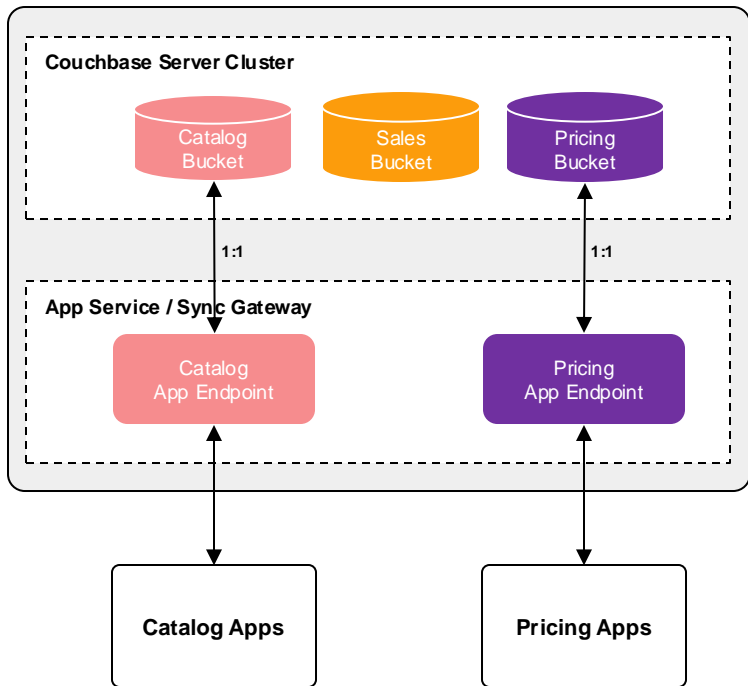
1-1. Couchbase Mobile 환경 > 구성 절차

구성 1안(설치형) : Couchbase Server ⇔ Sync Gateway ⇔ Couchbase Lite

구성 2안(관리형) : Capella (Operational DB ⇔ App Service) ⇔ Couchbase Lite

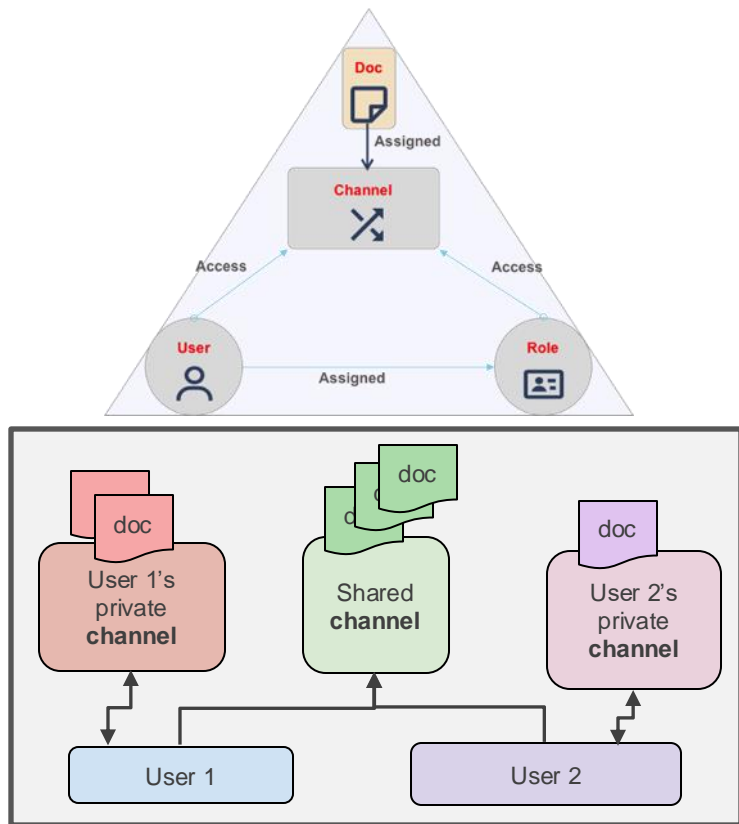


1-2. Couchbase Mobile 환경 > App Endpoints



- 특징
 - App에서 App Service를 접근할 수 있는 경로 제공
 - 1 개의 Bucket과 연결 설정 가능
- 참고 내용
 - 연결된 Bucket에 "Sync Gateway DB"가 생성되고 해당 Bucket 내에 데이터 저장
 - 구성 정보는 모든 App Service 노드에 적용

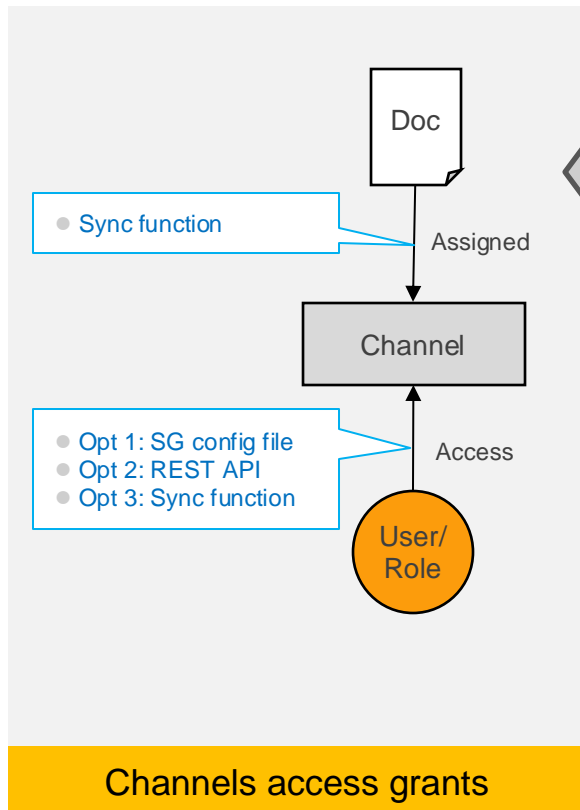
참고 > Access Control 아키텍처



- 문서를 채널로 라우팅하고 해당 유형의 문서에 액세스할 수 있도록 하려는 사용자 또는 역할이 해당 채널에 액세스할 수 있도록 하여 문서 액세스를 제어
> 접근 제어 구성 (채널, 사용자, 롤)
- 모든 다큐먼트를 1개 이상의 채널에 할당
- User에게 채널에 접근하기 위한 권한 부여
- Role은 유사한 접속 권한을 가진 User 그룹
 - Role에게 채널에 접근할 권한 부여
 - User는 특정 Role에 특정 등록

참고 : <https://docs.couchbase.com/sync-gateway/current/access-control-concepts.html>

참고 > Access Control 적용

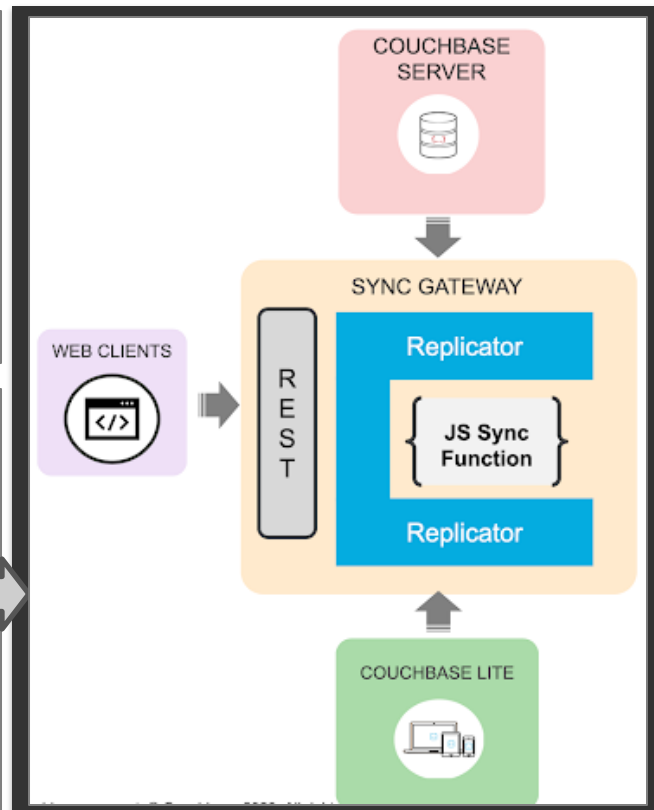


적용 방식

- Document를 Channel에 할당
 - Sync 함수
- User에 특정 Channel 접근 권한 부여
 - Configuration File
 - REST API
 - Sync 함수

Sync 함수

- Configuration File에 정의하는 JavaScript 함수
- Document가 변경되는 시점에 항상 호출
- 기능
 - Authorization
 - Validation
 - Access Grant
 - Channel Assignment



2-1. 구축형 Sync Gateway > 설치

- Download URL : <https://www.couchbase.com/downloads/?family=sync-gateway>

Sync Gateway **ENTERPRISE**

Enterprise-grade Sync Gateway builds on the strong foundations of the Community version and includes scaling, performance, availability, and security enhancements for business-critical mobile and edge computing applications. Features include delta sync for optimized data transfer, sharding of imports, tunable options for cache management, load balancing and HA of inter-Sync Gateway replications, and more.

Learn More

Before downloading Sync Gateway, see why customers are moving to Capella App Services, our fully hosted backend data sync solution. [Learn more about App Services](#)

Version

3.2.1 (Current) ▼

OS

Ubuntu ▼

Download

What's new

Release notes

Sync Gateway **COMMUNITY**



2-1. 구축형 Sync Gateway > 설치

- FTP or wget을 통한 설치 파일 다운로드

COUCHBASE SYNC GATEWAY 3.2.1 (CURRENT) ENTERPRISE FOR UBUNTU

[Download](#)[Copy Link](#)[Show checksum](#)

```
root@couchbase-139:~#
root@couchbase-139:~#
root@couchbase-139:~# wget https://packages.couchbase.com/releases/couchbase-sync-gateway/3.2.1/couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb
--2024-11-15 13:24:47-- https://packages.couchbase.com/releases/couchbase-sync-gateway/3.2.1/couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb
packages.couchbase.com (packages.couchbase.com) 해석 중... 13.226.61.22, 13.226.61.30, 13.226.61.105, ...
다음으로 연결 중: packages.couchbase.com (packages.couchbase.com) [13.226.61.22]:443... 연결했습니다.
HTTP 요청을 보냈습니다. 응답 기다리는 중... 200 OK
길이: 35498676 (34M) [application/vnd.debian.binary-package]
저장 위치: 'couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb'

couchbase-sync-gateway-enterprise_3.2.1_x 100%[=====] 33.85M 10.0MB/s / 4.5s

2024-11-15 13:24:53 (7.60 MB/s) - 'couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb' 저장함 [35498676/35498676]
```


2-1. 구축형 Sync Gateway > 설치

▪ Sync Gateway 설치

```
root@couchbase-139:~# dpkg -i couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb
Selecting previously unselected package couchbase-sync-gateway.
(데이터베이스 읽는 중 ...현재 218397개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb ...
Unpacking couchbase-sync-gateway (3.2.1-15) ...
couchbase-sync-gateway (3.2.1-15) 설정하는 중입니다 ...
Created symlink /etc/systemd/system/multi-user.target.wants/sync_gateway.service → /lib/systemd/system/sync_gateway.service.

You have successfully installed Couchbase Sync Gateway.

You can find sample sync_gateway configuration files in the /opt/couchbase-sync-gateway/examples folder.

You can control the Couchbase Sync Gateway service by using the following command:

    systemctl start sync_gateway

That's it! Sync Gateway is now running on port 4984. See https://docs.couchbase.com/sync-gateway/current/get-started-verify-install.html on how to get started.

By using this software you agree to the End User License Agreement.
See /opt/couchbase-sync-gateway/LICENSE.txt.
```

- 설치 CMD : `dpkg -i couchbase-sync-gateway-enterprise_3.2.1_x86_64.deb` (Ubuntu)
- 시작 : `systemctl start sync_gateway`
- 종료 : `systemctl stop sync_gateway`

- 설치 참고 URL : <https://docs.couchbase.com/sync-gateway/current/get-started-install.html>

2-2. 구축형 Sync Gateway > 사용 계정 생성

- User 생성 : syncgateway / password

The screenshot shows the Couchbase Security console interface. The left sidebar has a menu with 'Security' highlighted (1). The top navigation bar has 'ADD USER' highlighted (2). The 'Add New User' dialog is open, showing the following fields and options:

- Username:** syncgateway (3)
- Full Name (optional):** (empty)
- Password:** password (4)
- Verify Password:** password
- Roles:** Sync Gateway (5)
- Groups:** (empty)

The 'Add User' button is highlighted (5). A tooltip for the 'Sync Gateway' role is visible, stating: 'Full access to bucket data as required by Sync Gateway. This user cannot access the web console and is intended only for use by Sync Gateway. This user can read and write data, manage indexes and views, and read some cluster information.'

- 생성 참고 URL : <https://docs.couchbase.com/sync-gateway/current/get-started-prepare.html#configure-server>

2-2. 구축형 Sync Gateway > 설정 파일 생성(bootstrap)

▪ Sync Gateway config 파일 생성

- sg_config.json 생성

```
{
  "bootstrap": {
    "server": "couchbases://192.168.35.118",
    "username": "syncgateway",
    "password": "password",
    "server_tls_skip_verify": true,
    "use_tls_server": true
  },
  "api": {
    "admin_interface": ":4985"
  },
  "logging": {
    "console": {
      "enabled": true,
      "log_level": "info",
      "log_keys": ["*"]
    }
  }
}
```

Couchbase server
connect string

couchbase username

couchbase password

2-2. 구축형 Sync Gateway > 구동

- Sync Gateway config 파일 반영하여 재시작

- `systemctl stop sync_gateway`
- `/opt/couchbase-sync-gateway/bin/sync_gateway sg_config.json`

```
root@couchbase-139:/opt/couchbase-sync-gateway# /opt/couchbase-sync-gateway/bin/sync_gateway config.json
2024-11-15T14:04:10.248+09:00 ==== Couchbase Sync Gateway/3.2.1(15;a90d17c) EE ====
2024-11-15T14:04:10.249+09:00 [INF] Loading content from [config.json] ...
2024-11-15T14:04:10.259+09:00 [INF] Config: Starting in persistent mode using config group "default"
2024-11-15T14:04:10.260+09:00 [INF] Logging: Console to stderr
2024-11-15T14:04:10.260+09:00 [INF] Logging: Files disabled
2024-11-15T14:04:10.260+09:00 [ERR] No log_file_path property specified in config, and --defaultLogFilePath command line flag was not set. Log files required for
duct support are not being generated. -- base.InitLogging() at logging_config.go:78

2024-11-15T14:04:10.260+09:00 [INF] Logging: Console level: info
2024-11-15T14:04:10.260+09:00 [INF] Logging: Console keys: [*]
2024-11-15T14:04:10.260+09:00 [INF] Logging: Redaction level: partial
2024-11-15T14:04:10.262+09:00 [INF] Logging stats with frequency: &{1m0s}
2024-11-15T14:04:10.262+09:00 [INF] Initializing server admin connection...
2024-11-15T14:04:10.729+09:00 [INF] Config: Successfully initialized cluster agent
2024-11-15T14:04:10.729+09:00 [INF] Finished initializing server admin connection
2024-11-15T14:04:10.729+09:00 [INF] Initializing bootstrap connection..
2024-11-15T14:04:11.034+09:00 [WRN] Config: No database configs for group "default". Continuing startup to allow REST API database creation -- rest.(*ServerCont
initializeBootstrapConnection() at server_context.go:2098
2024-11-15T14:04:11.034+09:00 [INF] Config: Starting background polling for new configs/buckets: 10s
2024-11-15T14:04:11.034+09:00 [INF] Finished initializing bootstrap connection
2024-11-15T14:04:11.034+09:00 [INF] Diagnostic API not enabled - skipping.
2024-11-15T14:04:11.034+09:00 [INF] Starting metrics server on 127.0.0.1:4986
2024-11-15T14:04:11.034+09:00 [INF] Starting admin server on :4985
2024-11-15T14:04:11.034+09:00 [INF] Starting server on :4984 ...
```

2-2. 구축형 Sync Gateway > Endpoint 생성

- Couchbase Server와 Sync 할 Bucket/Scope/Collection 설정
 - Couchbase Server 인증 변수 설정 : `DIGEST=`echo -n syncgateway:password | base64``
 - Sync Gateway에 REST API 로 Couchbase Server의 **Endpoint** 구성, sync function 안에 channel로 이 Collection의 Documentation에 할당

```
$ DIGEST=`echo -n syncgateway:password | base64`  
$ curl --location --request PUT 'http://127.0.0.1:4985/survey/' \  
  --header "Authorization: Basic $DIGEST" \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "bucket": "travel-sample",  
    "scopes": {  
      "mobile": {  
        "collections": {  
          "survey": {  
            "sync": "function(doc){channel(\"survey_ch\");}"  
          }  
        }  
      }  
    },  
    "num_index_replicas": 0  
  }'
```

Endpoint Name

※ Scope, Collection 작성시 Lite DB와 동일한 이름

Bucket name

Scope name

Collection name

Channel name

Server	Gateway	Lite
<i>Bucket</i>	<i>Endpoint</i>	<i>DB</i>
Scope		Scope
Collection		Collection

2-2. 구축형 Sync Gateway > Role 생성

▪ Role 생성

- Sync Gateway의 Endpoint에 접근 할 수 있는 Role 생성

```
curl --location --request PUT 'http://127.0.0.1:4985/survey/_role/survey_rw_role' \
--header "Authorization: Basic $DIGEST" \
--header 'Content-Type: application/json' \
--data '{
  "name": " survey_rw_role ",
  "admin_channels": [" survey_adm_ch"]
}'
```

①

②

③

1. 작업할 DB 이름과 Role, Role 이름 추가
2. Role명 설정
3. 채널명 설정
 - 채널은 데이터베이스 데이터를 나눠서 어떤 데이터에 접근 할 수 있는지 정의하는 그룹
 - 역할에 채널을 설정하면, 그 역할을 가진 사용자는 해당 채널의 데이터에 접근할 수 있습니다.

2-2. 구축형 Sync Gateway > User 생성

■ User 생성

- Sync Gateway의 Endpoint에 접근 할 수 있는 User 생성
- 사용자는 개별인증과 데이터 접근 제어를 위해 필요
- 역할은 효율적인 권한 관리와 보안 강화를 위해 추가

```
curl --location -g --request POST 'http://localhost:4985/survey/_user/' \ ①
--header 'Content-Type: application/json' \
--header "Authorization: Basic $DIGEST" \ ②
--data '{
  "name": "investigator ", ③
  "password": "Passw0rd!",
  "admin_roles": ["survey_rw_role"], ④
  "admin_channels": ["survey_ch"] ⑤
}'
```

1. 작업할 Endpoint(Bucket) 에 접근 가능한 User 이름 추가, API Endpoint
2. 관리자 User (Couchbase Server에 등록 되어 있는 사용자 Slide 9)
3. 새로운 User 정보
4. 기존에 생성 한 Role 할당
5. (옵션) Role에서 할당된 채널 이외 추가 채널 지정

2-2. 구축형 Sync Gateway > Role/User 생성 예시

- Role, User 생성 예시.

```
root@couchbase-139:~# curl --location --request PUT 'http://127.0.0.1:4985/sync_test/_role/stdrole' \
  --header "Authorization: Basic $DIGEST" \
  --header 'Content-Type: application/json' \
  --data '{
    "name": "stdrole",
    "admin_channels": ["newrolechannel"]
  }'
root@couchbase-139:~#
root@couchbase-139:~# curl --location -g --request POST 'http://localhost:4985/sync_test/_user/' \
  --header 'Content-Type: application/json' \
  --header "Authorization: Basic $DIGEST" \
  --data '{
    "name": "sgwuser1",
    "password": "passwordstring",
    "admin_roles": ["stdrole"],
    "admin_channels": ["public"]
  }'
```


3-1. Capella App Service 생성

- Capella App Service 생성 : [survey_app](#)

Create App Service

App Service Name *

[survey_app](#)

You can rename this at any time.

8/128

Linked Cluster

Choose the cluster for your App Service. Single Node clusters can only use Single Node App Services, for development and testing purposes. Your chosen cluster must not already have a linked App Service and must meet the [specified restrictions](#). Linking an App Service to a cluster will see a slight increase in data storage and index sizes.[Learn More](#)

Only clusters deployed in [AWS](#), [Azure](#), and [GCP](#) regions supported by App Services are listed below.

Linked Cluster *

[my_capella_cluster_name](#)

You can only select a cluster that can be linked to an App Service.

ⓘ

Your **Linked Cluster** is a free tier cluster. Your App Service will use the available free tier configuration. You cannot change this configuration.

Plan

Your App Service will use the same plan as your linked cluster. App Services adds additional costs to your plan [Learn More](#)

App Service Details

Name: [app_test](#)
Project: [My first project](#)
Cluster: [couchbase_capella](#)

Configuration

Nodes: Free Tier

Estimated Cost*

[FREE](#)

연결 할 클러스터 선택

※ App Service 생성 전 Lite 동일한 이름의 Scope, Collection 생성

3-2. Capella App Service > Endpoint 생성

- Endpoint 생성 (Endpoint : [survey](#), Bucket : [travel-sample](#), Scope : [mobile](#), Collection : [survey](#))

[← Back to App Service](#)

Create App Endpoint

i By default, **Delta Sync** and **Import Filter** are turned off and your **Authentication Provider** is set to Basic Authentication. Change these settings after you create your App Endpoint.

Name your App Endpoint

App Endpoint Name *

[survey](#)

⚠ You cannot change the App Endpoint name later.

4/128

Choose a bucket and scope

Choose a bucket and scope to link collections to your App Endpoint. You can only select a **Memory** and **Disk** bucket.

Bucket *

[travel-sample](#)

⚠ You cannot change the linked bucket later.

Scope *

[mobile](#)

⚠ You cannot change the linked scope later.

Choose collections to link

Link collections to your App Endpoint to use them in your applications. Link collections are linking. **You can link a maximum of 250 collections at once.** Endpoint.

COLLECTION	LINK OR UNLINK
survey	Link Unlink

Summary

App Endpoint Name
demo

Bucket
sync_test

Scope
test

Collections
1 Linked

[Create App Endpoint](#)

Capella기준으로 Lite와 연결 시킬
Bucket, Scope, Collection 선택 (※ Lite DB와 동일한 이름)

3-2. Capella App Service > Endpoint > Role 생성

▪ Role 생성

[← Back to App Roles List](#)

Create App Role

App Role Name *

stdrole

Name cannot be edited after creating the App Role.

7/128

Assign Channels From Linked Collections

Enter the Channel names for each linked collection that you want this App User to access. You can enter multiple Channel names for a linked collection. App Users can only access documents in their assigned Channels from each linked collection. [Learn More](#)

linked bucket [lite_sync](#) / linked scope [lite](#)

Admin Channels

LINKED COLLECTIONS	ADMIN CHANNELS
<div>survey</div>	<div>newrolechannel x</div>

Showing 1 of 1 results

Save

Cancel

3-2. Capella App Service > Endpoint > User 생성

- Endpoint 에 접속할 User 생성 : Username : **investigator** , password : **Passw0rd!**

Create App User

Credentials

App User Name *

investigator

12/128

Password *

.....

Minimum 8 characters. Use at least 1 uppercase, 1 lowercase, 1 number and 1 symbol.

☐ Disable App User
Disabled App Users cannot access App Endpoints.

[Configure Access Grants](#) ^

Assign App Roles (Optional)

You can use App Roles to grant access to Channels and group together similar App Users. An App User can access any Channels assigned to their App Role, or you can assign Channels for each App User. [Learn More](#)

Assigned App Roles

stdrole x

Assign Channels (Optional)

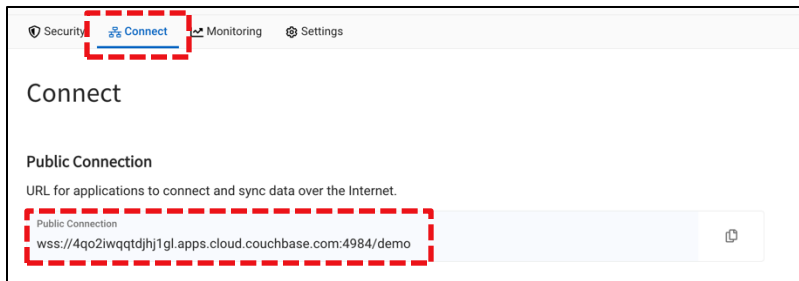
Enter the Channel names for each linked collection that you want this App User to access. You can enter multiple Channel names for a linked collection. App Users can only access documents in their assigned Channels from each linked collection. [Learn More](#)

Bucket **sync_test** / Scope **test**

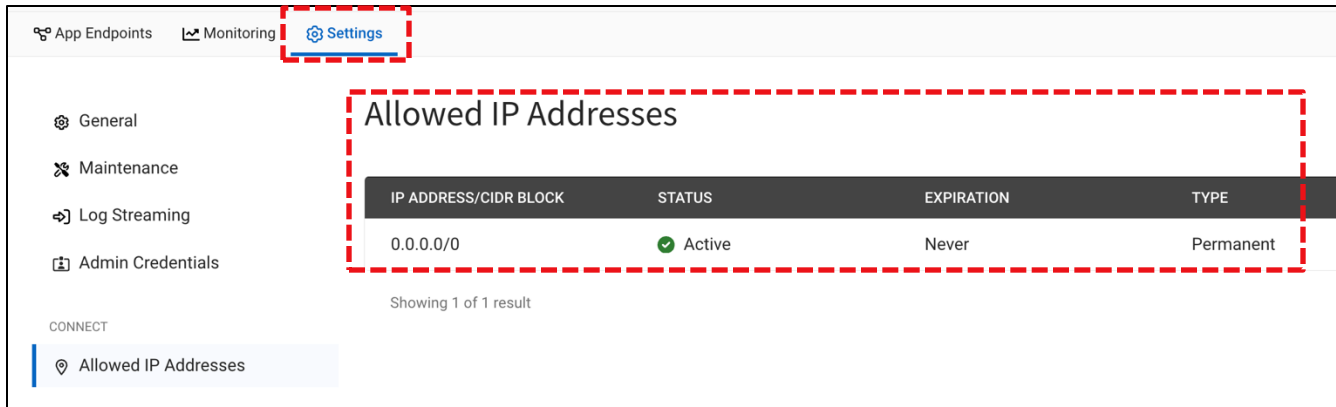
LINKED COLLECTIONS	ADMIN CHANNELS
survey	public x

3-3. Capella App Service > Allowed IP Addressess 등록

- AppService IP and 접속 허용

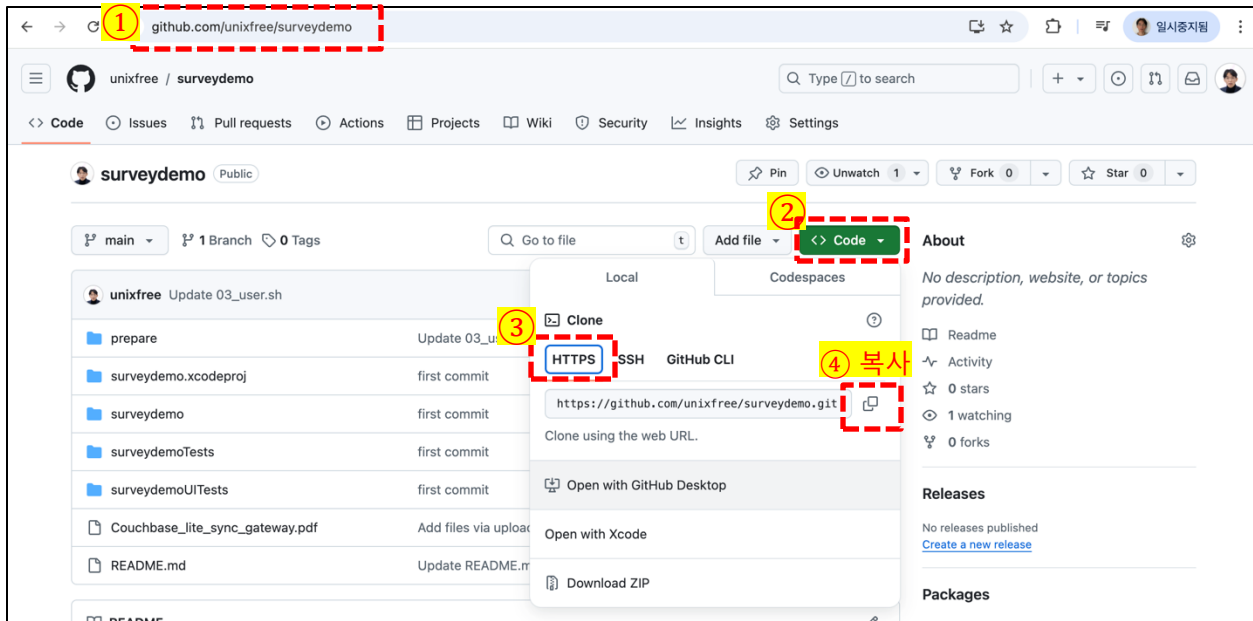


1. AppService 연결시 주소창의 IP가 아닌 Connect Tab > Connection IP 필요
2. Settings Tab -> IP 접속 허용 필요



4-1. App 소스 다운로드

- Survey 데모 앱 소스 다운로드
 - <https://github.com/unixfree/surveydemo>



5 `$ cd my_folder`
`$ git clone https://github.com/unixfree/surveydemo.git`

6 Xcode 실행 후, **File > Open... > my_folder > surveydemo > surveydemo.xcodeproj** 선택 후 > **Open** 클릭

4-2. Xcode에서 Couchbase Packages 설치

The screenshot illustrates the process of installing Couchbase Lite Swift EE as a package dependency in Xcode. The interface is divided into several panes:

- Left Pane (Project Navigator):** Shows the project structure for 'surveydemo'. A red dashed box with a yellow circle '1' highlights the 'surveydemo' project.
- Right Pane (Package Dependencies):** Shows the 'Package Dependencies' tab. A red dashed box with a yellow circle '3' highlights the 'Package Dependencies' tab. Below it, a table lists the installed packages:

Name	Location	Dependency Rule
CouchbaseLiteSwift	https://github.com/couchbase/couchbase-lite-swift-ee	Up to Next Major Vers

A red dashed box with a yellow circle '4' highlights the '+' button to add a new package.

A yellow box with black text contains the URL: <https://github.com/couchbase/couchbase-lite-swift-ee.git> 입력

The 'Recently Used' section shows two packages: 'couchbase-lite-vector-search-spm' and 'couchbase-lite-swift-ee'. A red dashed box with a yellow circle '5' highlights the search bar.

The 'CouchbaseLiteSwift-EE' package details are shown, including the repository URL and the dependency rule. A red dashed box with a yellow circle '6' highlights the 'Add Package' button.

The 'Choose Package Products for couchbase-lite-swift-ee.git' dialog is shown. A red dashed box with a yellow circle '7' highlights the 'surveydemo' package product. A red dashed box with a yellow circle '8' highlights the 'Add Package' button.

4-3. App Connect Code 설명

■ InsertDataView.swift

```
// 데이터 제출 함수
func submitData() {
    // ID 유효성 검사: 입력된 ID가 비어 있을 경우 경고 표시
    guard !documentID.trimmingCharacters(in: .whitespacesAndNewlines).isEmpty else {
        showIDWarning = true
        return
    }

    showIDWarning = false // ID가 비어 있지 않으면 경고 해제

    do {
        let database = try Database(name: "survey")
        let collection = try database.createCollection(name: "survey", scope: "mobile")

        // 기존 문서가 있으면 업데이트, 없으면 새로 저장
        let document = MutableDocument(id: documentID)
        document.setString(gender, forKey: "gender")
        document.setString(age, forKey: "age")
        document.setString(visit, forKey: "visit")
        document.setString(visitReason == "기타" ? otherReason : visitReason, forKey:
            "visitReason")
        document.setString(rating, forKey: "rating")
        document.setString(text, forKey: "text")
        document.setDate(Date(), forKey: "createdAt")

        // 데이터 저장
        try collection.save(document: document)
        print("데이터가 성공적으로 저장되었습니다.")

        // 폼 초기화
        resetForm()
        documentID = ""
    } catch {
        print("데이터 저장 실패: \(error.localizedDescription)")
    }
}
```

1) ID 확인

- 입력한 ID(문서의 이름 같은 것)가 빈칸인지 확인
 - > ID가 없으면 데이터를 저장할 수 없기 때문에 경고 메시지
 - > 예: "ID를 입력해주세요!"

2) 데이터베이스 준비

- **Mobile Database(name: "survey")**
 - > 이 부분은 "survey"라는 데이터베이스를 오픈
- **Collection(name: "survey", scope: "mobile")**
 - > 데이터베이스 안에서 "sync"라는 그룹(Collection)을 생성

3) 문서 작성

- **MutableDocument(id: documentID)**
 - ID를 사용해 **문서(Document)**를 생성
 - > 문서 = 데이터를 저장할 파일 같은 것
- 문서 안에 정보를 저장
 - > gender, age, visit, visitReason, rating, text, 그리고 작성 날짜(createdAt)

4) 데이터 저장

- **collection.save(document: document)**
 - 문서를 데이터베이스에 저장
 - > 새 문서면 추가하고,
 - > 기존 문서면 수정
 - > 결과: "데이터가 성공적으로 저장되었습니다!"

5) 폼 초기화

- 저장이 끝나면 입력했던 폼을 초기화해서 다음 입력을 준비
 - > 모든 입력란을 빈칸으로 초기화

4-3. App Connect Code 설명

■ SelectDataView.swift

```
// 데이터 조회 함수
func fetchData() {
    do {
        let database = try Database(name: "survey")
        let collection = try database.createCollection(name: "survey", scope: "mobile")

        var query: Query

        // 필터 옵션에 따른 쿼리 설정
        if selectedOption == "All" {
            query = QueryBuilder
                .select(SelectResult.all(), SelectResult.expression(Meta.id)) // Meta.id 포함
                .from(DataSource.collection(collection))
        } else {
            query = QueryBuilder
                .select(SelectResult.all(), SelectResult.expression(Meta.id)) // Meta.id 포함
                .from(DataSource.collection(collection))
                .where(Meta.id.equalTo(Expression.string(documentID)))
        }

        // 쿼리 실행 및 결과를 배열로 변환
        let result = try query.execute()
        documents = result.allResults().compactMap { result in
            let dict = result.dictionary(forKey: "survey")
            return MyData(
                id: result.string(forKey: "id") ?? "N/A", // Meta.id 값을 id 필드로 설정
                gender: dict?.string(forKey: "gender") ?? "N/A",
                age: dict?.string(forKey: "age") ?? "N/A",
                visit: dict?.string(forKey: "visit") ?? "N/A",
                visitReason: dict?.string(forKey: "visitReason") ?? "N/A",
                rating: dict?.string(forKey: "rating") ?? "N/A",
                text: dict?.string(forKey: "text") ?? "N/A"
            )
        }

        print("데이터가 성공적으로 조회되었습니다.")
    } catch {
        print("데이터 조회 실패: \(error.localizedDescription)")
    }
}
```

1) 데이터베이스 준비

- Mobile Database(name: "survey")
 - > "survey"라는 데이터베이스를 오픈
- Collection(name: "survey", scope: "mobile")
 - > 데이터베이스 안에서 "survey"라는 그룹(Collection)**을 준비
 - > 데이터를 저장한 그룹에서 데이터를 가져옴

2) 쿼리(Query) 준비

- 데이터를 어떻게 가져올지 쿼리(Query)를 생성
- 사용자가 선택한 옵션(selectedOption)에 따라 쿼리가 달라집니다.
 - > 모든 데이터를 가져올 때 : selectedOption이 "All"이면 저장된 모든 데이터 가져옴
 - > 특정 ID로 데이터를 가져올 때 : selectedOption이 "ID"이면 사용자가 입력한 documentID에 해당하는 데이터만 가져옴
 - > 쿼리는 QueryBuilder를 사용해 작성

3) 쿼리 실행

- query.execute()
 - > 준비한 쿼리를 실행해서 데이터를 가져옴
 - 가져온 데이터는 배열로 변환하여(allResults()) 사용하기 쉽게 정리

4) 결과를 객체로 변환

- 가져온 데이터는 MyData라는 객체로 변환
 - > 각 데이터의 필드 값을 읽어서 객체로 생성
 - > 필드 값 예시: id: 문서의 ID, gender: 성별, age: 나이, visit: 방문 여부, visitReason: 방문 이유, rating: 평가 점수, text: 추가 정보

5) 데이터 조회 결과 확인

- 데이터가 성공적으로 조회되면 콘솔에 "데이터가 성공적으로 조회되었습니다." 메시지가 출력됨
- 데이터 조회에 실패하면 오류 메시지를 출력됨

4-3. App Connect Code 설명

■ ServerSyncView.swift

```
// 데이터 동기화 함수
func syncData() {
    do {
        let database = try Database(name: "survey")
        let collection = try database.createCollection(name: "survey", scope: "mobile")

        // 서버 엔드포인트 설정 (요청하신 URL로 수정)
        let target = URLEndpoint(url: URL(string:
            "wss://u49geqy3uv9kf6.apps.cloud.couchbase.com:4984/survey")!) // Sync_gateway가
            설치된 서버 주소
        var replConfig = ReplicatorConfiguration(target: target)

        // 컬렉션 및 기본 충돌 해결 방식 추가
        replConfig.addCollection(collection)
        // Couchbase Server에서 생성한 User 정보 RestAPI "Authorization: Basic $DIGEST" 방식과 동일
        replConfig.authenticator = BasicAuthenticator(username: "investigator", password:
            "Password!")
        replConfig.replicatorType = .pushAndPull

        // 동기화 시작
        let replicator = Replicator(config: replConfig)
        replicator.start()

        replicator.addChangeListener { (change) in
            if let error = change.status.error as NSError? {
                print("동기화 오류 발생: \(error.localizedDescription)")
            } else {
                print("동기화 상태: \(change.status.activity) - 완료 문서 수:
                    \(change.status.progress.completed) / 총 문서 수:
                    \(change.status.progress.total)")
            }
        }

        // 동기화 완료 후 토스트 메시지 표시
        showToast = true
    } catch {
        print("동기화 실패: \(error.localizedDescription)")
    }
}
```

1) 데이터베이스와 컬렉션 준비

- **Mobile Database(name: "survey")**
 - > 로컬에서 사용할 데이터베이스를 오픈
- **Collection(name: "survey", scope: "mobile")**
 - > 데이터베이스 안의 특정 그룹(Collection)을 선택
 - > 동기화할 데이터가 이 컬렉션에 저장

2) 동기화 서버 설정

- **Sync Gateway 서버 주소**를 설정
- **URLEndpoint(url: "wss://u49geqy3uv9kf6.apps.cloud.couchbase.com:4984/survey")**
 - > "wss://"는 동기화를 위한 **WebSocket 프로토콜**
 - > **u49geqy3uv9kf6.apps.cloud.couchbase.com** 는 App Service 의 DNS 주소.
 - > "survey"는 동기화할 Endpoint(Bucket) 이름

3) 동기화 구성

- **ReplicatorConfiguration**으로 동기화 설정:
 - > **컬렉션 추가**: 동기화할 데이터를 정의
 - > **인증 추가**: 서버에 접근하기 위해 사용자 이름과 비밀번호 입력.
 - > BasicAuthenticator를 사용해 username과 password를 설정
 - > **동기화 타입**: .pushAndPull을 사용해 서버로 데이터 보내기와 서버에서 데이터 가져오기를 동시에 설정.

4) 동기화 상태 확인

- **addChangeListener**로 동기화 상태를 실시간으로 확인:
 - > 동기화 중 오류가 발생하면 오류 메시지를 출력됨
 - > 동기화 진행 상황(완료된 문서 수와 총 문서 수)을 출력합니다.
 - 예: "완료 문서 수: 5 / 총 문서 수: 10"

5) 동기화 완료 후 알림

동기화가 완료되면 화면에 알림 메시지(토스트)를 표시함

4-4. App 소스 Endpoint 설정

■ ServerSyncView.swift

```
// 데이터 동기화 함수
func syncData() {
    do {
        let database = try Database(name: "survey")
        let collection = try database.createCollection(name: "survey", scope: "mobile")

        // 서버 엔드포인트 설정 (요청하신 URL로 수정)
        let target = URLEndpoint(url: URL(string:
            "wss://u49geqy3uv9kf6.apps.cloud.couchbase.com:4984/survey")) // Sync_gateway가
            // 위치
        var replConfig = ReplicatorConfiguration(target: target)

        // 컬렉션 및 기본 충돌 해결 방식 추가
        replConfig.addCollection(collection)
        // Couchbase Server에서 생성한 User 정보 RestAPI "Authorization: Basic SECRET" 방식과 동일
        let authenticator = BasicAuthenticator(username: "investigator", password:
            "Password!")
        replConfig.authenticator = authenticator
        replConfig.replicatorType = .pushAndPull

        // 동기화 시작
        let replicator = Replicator(config: replConfig)
        replicator.start()

        replicator.addChangeListener { (change) in
            if let error = change.status.error as NSError? {
                print("동기화 오류 발생: \(error.localizedDescription)")
            } else {
                print("동기화 상태: \(change.status.activity) - 완료 문서 수:
                    \(change.status.progress.completed) / 총 문서 수:
                    \(change.status.progress.total)")
            }
        }

        // 동기화 완료 후 토스트 메시지 표시
        showToast = true
    } catch {
        print("동기화 실패: \(error.localizedDescription)")
    }
}
```

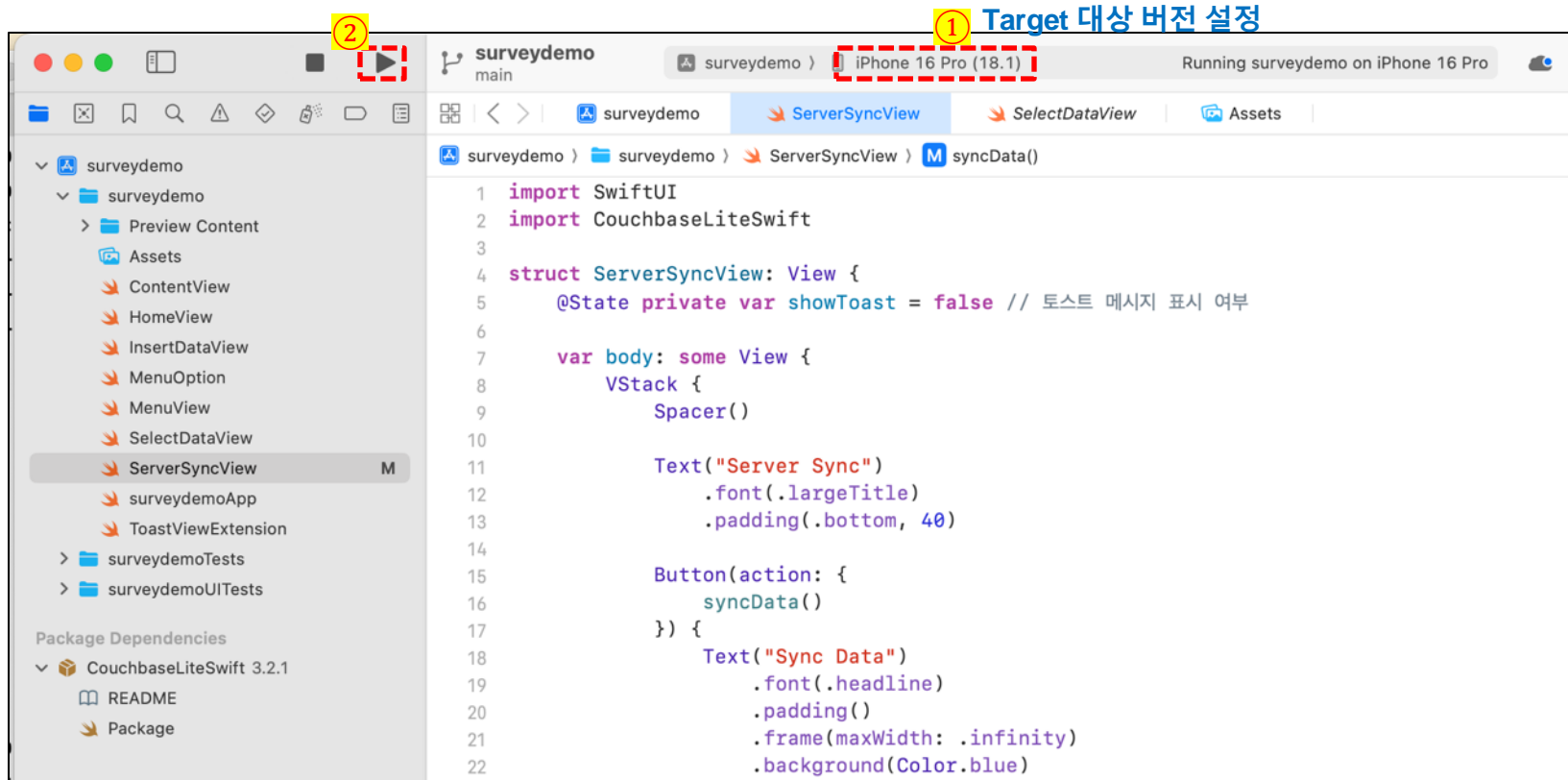
1) Sync_gateway를 통한 App 연결

- > Sync Gateway App IP 주소 + Endpoint(Bucket) 명
- > Sync_gateway와 Capella AppService는 비밀번호 요구사항이 다르기 때문에 Password 확인

2) AppService를 통한 App 연결

- > AppService Connect Tab에서 사용 되는 IP 사용
- > 허용 IP 설정
- > Sync_gateway와 Capella AppService는 비밀번호 요구사항이 다르기 때문에 Password 확인

4-5. App 빌드 및 실행





Thank you!



작성 : jong.park@couchbase.com
수정 : paul.son@couchbase.com



Couchbase