THINK 2020 Virtual Lab :

CareKit SDK + IBM Hyper Protect SDK for iOS

May 06th 2020, 3:00 - 5:00 PM EST

Moderators:

- Elton de Souza (Cloud Native Client Success - IBM)
- Mal Pattiaratchi (IBM Technical Leader - IBM)
- Ryley Wharton (Client Success and Developer Advocate - IBM)

Agenda:

- Account setup
- Deploy a Sample App with CareKit
- Deploy a Hyper Protect Virtual Server instance
- Deploy a IBM Hyper Protect DBaaS for MongoDB instance
- Integrate IBM Hyper Protect with the Sample App
    - IBM Hyper Protect MBaaS
    - Bootstrapping with Ansible
        - Bootstrapping Hyper Protect Virtual Server
        - Bootstrapping local dev environment
- Integrate IBM Hyper Protect SDK for iOS into the Sample App
    - Setup
- Troubleshooting

Useful links:

- Developer blog post
- Executive blog post
- CareKit
- IBM Cloud Hyper Protect Services

Account setup

Goal : At the end of this section, you should have pre-requisite accounts needed to successfully complete this lab

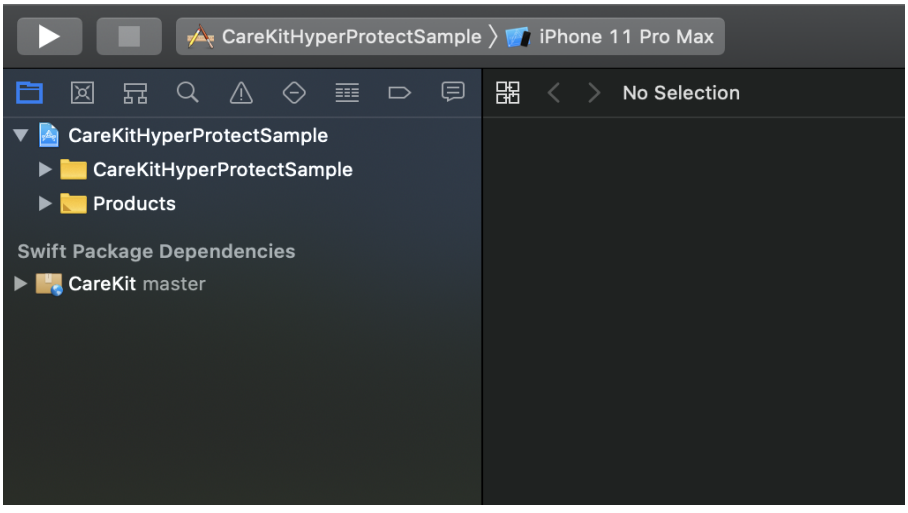
IBM Cloud Account:

An IBM Cloud Account is required for this lab. If you need to create one, please go to : https://cloud.ibm.com/registration and follow the prompts. Additional documentation can be found here

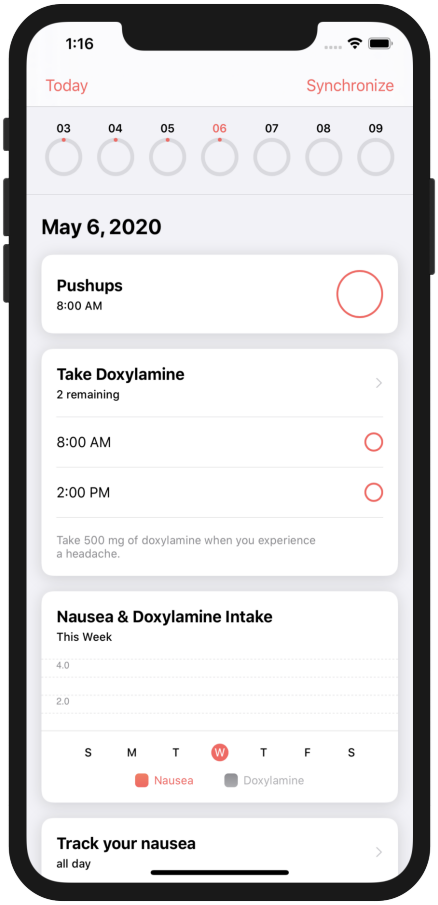
Deploy a Sample App with CareKit

> CareKit requires XCode 11.4+

1. First `git clone https://github.com/THINKLab2020/CareKitHyperProtectSample.git`
2. Click on 'CareKitHyperProtectSample.xcodeproj' to open up the project in XCode. Once the dependencies are downloaded, you should see this:



3. Hit Run to build the sample app. The first run might take a while since it's building everything from source, but subsequent runs will be much quicker.
4. You should see an app in the simulator:



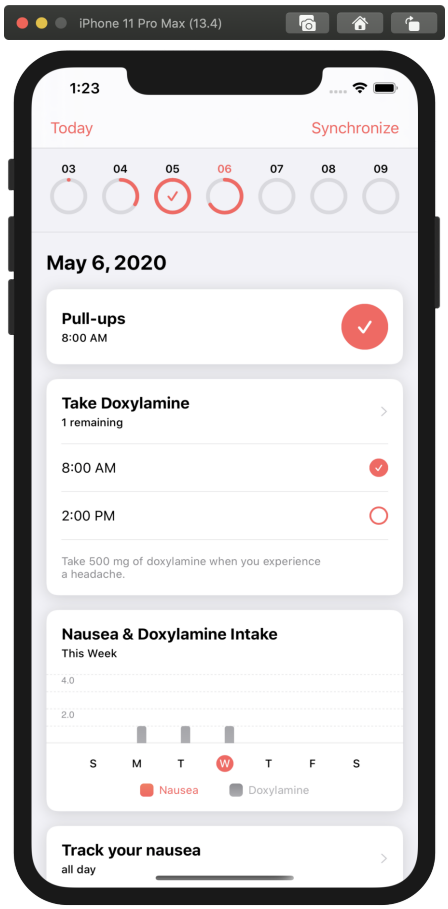
1. The first view you see are tasks that were programmatically set up using `OCKTasks`. Tasks look at the populateSampleData() in the AppDelegate.swift file to see how this was done.

```
        let thisMorning = Calendar.current.startOfDay(for: Date())
        let aFewDaysAgo = Calendar.current.date(byAdding: .day, value: -4,
to: thisMorning)!
        let beforeBreakfast = Calendar.current.date(byAdding: .hour,
value: 8, to: aFewDaysAgo)!
        let afterLunch = Calendar.current.date(byAdding: .hour, value: 14,
to: aFewDaysAgo)!

        let schedule = OCKSchedule(composing: [
            OCKScheduleElement(start: beforeBreakfast, end: nil,
                               interval: DateComponents(day: 1)),

            OCKScheduleElement(start: afterLunch, end: nil,
                               interval: DateComponents(day: 2))
        ])

        var doxylamine = OCKTask(id: "doxylamine", title: "Take
Doxylamine", carePlanUUID: nil, schedule: schedule)
```

6. Next, click on some tasks to see the graphs change. These are `outcomes` and are programmatically represented by `OCKOutcome`. Typically you would only explicitly define an outcome programmatically for testing purposes. In regular app operation, outcomes are created as a result of tasks being completed.

7. Change or add tasks in the populateSampleData().
    - Changing text will be the easiest. You can change the `title` or `instruction` of the existing `doxylamine` OCKTask.
    - If you do decide to create a new task, it will need a schedule. You could also re-use the original `doxylamine` tasks' `schedule` for your new task with some imaginative times.
    - If you change `id`, you'll need to change the array `identifiers` in `CareViewController.swift` too

8. Push build on your new app. In our case, we changed the text from pushups to "pull-ups".



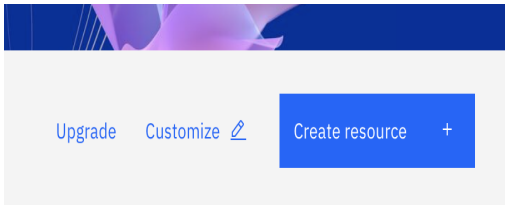
Deploy a Hyper Protect Virtual Server instance

Goal : At the end of this section, you should have a production ready MongoDB instance from IBM Hyper Protect DBaaS and a secure execution environment from IBM Hyper Protect Virtual Servers to deploy the IBM Hyper Protect MBaaS on.

1. Log into the IBM Cloud dashboard (https://cloud.ibm.com). Please refer to the documentation on the *README* page for further instructions on logging into IBM Cloud, or obtaining an IBM Cloud account

2. After the IBM Cloud homepage loads, click the blue 'Create Resource' button on the top right of the page, directly underneath the Cloud toolbar. Alternatively, the create resource function is accessible by selecting the 'Add Resource +' option underneath the Resource Summary table.

    - The desired destination is the Cloud catalog, where all current available Cloud offerings are listed
    - In order to view the full list of Cloud offerings, ensure that the 'Services' tab is selected on the left-hand side of the catalog



3. Within the catalog listings, locate and click on 'Hyper Protect Virtual Servers'.

    - The page defaults to the 'Create' tab, where the various Virtual Server options are able to be selected.

    - Choose the appropriate region to deploy the virtual server in, as well as a suitable pricing plan based on the technical requirements for the Virtual Server. *The 'free' payment option will be adequate for this lab*

    - Add any necessary tags for labeling the virtual server

        - Copy and paste your public ssh key (`cat ~/.ssh/id_rsa.pub` locally) into the last field. This is a required step as the Virtual server is only accessible via ssh key. If you do not know how/where to find your ssh key, please follow these instructions.
        - After all required fields have been properly filled out, click the create button on the bottom right hand side of this page to begin provisioning of the Hyper Protect Virtual Server instance

> As this will take some time, we will deploy a MongoDB instance in IBM Hyper Protect and then test the connection to this virtual server once that's complete.

Deploy a IBM Hyper Protect DBaaS for MongoDB instance

Goal : At the end of this section, you should have a production ready MongoDB instance for consumption by the IBM Hyper Protect MBaaS

---

1. Navigate back to the IBM Cloud catalog, and search for the 'Hyper Protect DBaaS for MongoDB' offering
2. Choose the appropriate region to deploy the virtual server in, as well as a suitable pricing plan based on the technical requirements needed for the MongoDB service.
3. There are several fields listed below the pricing plans, some of which are required, while other fields are optional. Follow the next steps for the required fields.
4. The non-essential fields will be left to the default values:
    - Create and add a 'Cluster Name' for the database. e.g. My_Mongo_Cluster
    - Choose an admin ID name for the MongoDB service, this ID will have full administrative access to the database e.g Example: admin
    - Generate a password for the associated database admin user
    - PLEASE TAKE NOTE OF USERNAME AND PASSWORD YOU WILL NEEED THEM LATER.

> Password requirements: 15 characters minimum, must include 1 capital letter, and 1 number, The use of non-alphanumeric symbols is not required but is recommended

After all required fields have been properly filled out, click the create button on the bottom right hand side of this page to begin provisioning of the Hyper Protect DBaaS for MongoDB

> **Note:** Please allow up to 10 minutes for both services to deploy after the process initiates. There are several security hardening processes and checks happen after you deploy your instances which is why these take longer than regular VM deployments.

Further Documentation:

- HPVS Documentation
- DBaaS Documentation

Connect to your Virtual Server (optional)

Now that both required Cloud services have been deployed, it is time to access the newly provisioned virtual server using ssh protocol

1. To obtain the public IP address of the virtual server, navigate to the Cloud dashboard homepage, and click on 'Services' in the Resource Summary box. Once the deployed services are listed, locate the Hyper Protect virtual server instance and click on the name. The public IP address will be displayed on the following screen.
2. After finding the public IP address, access a terminal and leverage the ssh protocol to navigate to the virtual server. Example ssh command: `ssh root@{public_IP}`. The ID required for a successful ssh connection must be 'root'

Integrate IBM Hyper Protect with the Sample App

Goal : At the end of this section, you should have a fully running CareKit based app end-to-end that that consumes IBM Hyper Protect Services

---

The SDK consists of 2 layers:

- The frontend Swift component that extends CareKit's functionality
- The backend server that acts as a mediator between the frontend and the database

---

TODO : Include Arch Diagram

IBM Hyper Protect MBaaS

Goal: At the end of this next section, the recently provisioned Hyper Protect Virtual server and Hyper Protect DBaaS Instance should be fully configured with the Backend SDK CareKit application. Follow the illustrated steps below to run the *ansible* playbook for automated setup.

Prerequisites

Note: if pip3 fails, re-run the commands using just pip

1. Install ansible via pip3 (or pip) with:

```
pip3 install ansible
``
```

2. Install docker-compose via pip3

```
pip3 install docker-compose
```

3. Install docker

```
pip3 install docker
```

Bootstrapping with Ansible

1. Clone the carekit-hyperprotect-lab repository on the local machine using `git clone https://github.com/e-desouza/carekit-hyperprotect-lab.git` This github repository

containing the ansible playbooks has been made public for accessibility.

2. After the Github repo has been cloned to the local machine, change directories to `carekit-hyperprotect-lab/ansible_setup`.

3. There are two available playbooks that can be run for bringing up the CareKit Backend SDK application. The first playbook uses the HPVS and HPDBaaS instances recently provisioned. While the second option will configure the Backend SDK app on the local machine instead. Choose the next set of instructions accordingly.

Bootstrapping Hyper Protect Virtual Server

> Pre-requisites : IBM Hyper Protect Virtual Servers and MongoDB in IBM Hyper Protect DBaaS

Two critical modifications need to be made to the playbook files in order to successfully run the hpvs_setup.yml playbook.

The first change is within the ansible.cfg file. Alter the value for environmental variable `ansible_ssh_private_key_file` with the location and name of your public ssh key. This particular ssh key **must** be the same key used when deploying the Hyper Protect Virtual Server, as illustrated in a previous section.

With Linux and Mac, the default public ssh key location is:

> ~/.ssh/id_rsa.pub

ansible.cfg:

```
$ cat ansible.cfg
[defaults]
inventory = inventory.yml
remote_user = root
deprecation_warnings=False

#add public ssh key directory and filename as private_key_file value
ansible_ssh_private_key_file = "{Public_SSH_Key}"
```

The second change occurs within the `inventory.yml` file. Underneath the commented line, add the public IP address of the Hyper Protect Virtual Server.

inventory.yml:

```
$ cat inventory.yml
[hosts]
# if running locally local_setup.yml is already pointed towards
'localhost' and no change needs to be made
#add HPVS public IP below this comment
{Public_IP_Address}
```

Please goto the MongoDB service you created on the IBM Cloud and then click on the copy icon next to where it says "To connect to your database(s) with Compass, use the URL below."

<>

Please edit the command below and replace the {mongoUrl} field with the URL you just copied.

```
ansible-playbook hpvs_setup.yml -e "db={mongoUrl}"
```

The command should look like (with your specific mongoUrl)

ansible-playbook hpvs_setup.yml -e "db=mongodb://dbaas31.hyperp-dbaas.cloud.ibm.com:28128,dbaas29.hyperp-dbaas.cloud.ibm.com:28239,dbaas30.hyperp-dbaas.cloud.ibm.com:28219/admin?replicaSet=mal_cluster"

Now, add the admin ID and password used when creating the HPDBaaS MongoDB instance (you will have noted it down prior - if you haven't - oops you will need to delete the MongoDB instance and create a new one!)

```
ansible-playbook hpvs_setup.yml -e "db=mongodb://{username}:
{password)@dbaas31.hyperp-dbaas.cloud.ibm.com:28128,dbaas29.hyperp-
dbaas.cloud.ibm.com:28239,dbaas30.hyperp-dbaas.cloud.ibm.com:28219/admin?
replicaSet=mal_cluster"
```

The command should look like (with your specific username, password and mongoUrl)

```
ansible-playbook hpvs_setup.yml -e
"db=mongodb://admin:password12345@dbaas31.hyperp-
dbaas.cloud.ibm.com:28128,dbaas29.hyperp-
dbaas.cloud.ibm.com:28239,dbaas30.hyperp-dbaas.cloud.ibm.com:28219/admin?
replicaSet=mal_cluster"
```

Now run the correctly formatted command.

Allow the playbook to run through it's designated tasks and configure the HPVS container.

```
PLAY [Configure Hyper Protect Virtual
Serve*****************************************************************
************************
TASK [Gathering
Fact*************************************************************\*********
*************************************************
ok: [169.63.212.6
TASK [Install required system
package*****************************************************\*\*\*\**********
*************************************
```

```
changed: [169.63.212.61] => (item=curl)
changed: [169.63.212.61] => (item=npm)
changed: [169.63.212.61] => (item=docker.io)
changed: [169.63.212.61] => (item=ufw)
changed: [169.63.212.61] => (item=docker-compose)
ok: [169.63.212.61] => (item=gi
TASK [Clone github repo (SDK
CareKit**************************************************\**************
**********************************
changed: [169.63.212.61]
.
PLAY
REC*****************************************************************\****
*************************************************************
169.63.212.61 : ok=9 changed=7 unreachable=0 failed=0 skipped=0 rescued=0
ignored=0
```

The HPVS configuration should now be complete, follow the validation test section listed below to confirm the setup worked as intended.

Bootstrapping local dev environment

> This section is only here for completeness. You will be using the services from IBM Cloud instead of localhost so you can skip this section for the lab

Please note that while the local setup does *not* require an IBM Cloud HPVS nor DBaaS instance, a few local

Unlike the 'hpvs_setup.yml' playbook, the local setup already has the correct host configuration written within the playbook itself, and does not require any additional file modifications. Please use the listed command below to run the ansible script.

Do note that a prompt will occur for the *BECOME* password due to the *-K* parameter being passed into the command. When prompted for the password, enter your local machine password, as the password is required when setting up the application.

```
$ ansible-playbook local_setup.yml -K
```

Allow the playbook to run and complete the predefined playbook tasks.

```
PLAY [Configure Hyper Protect Virtual
Server*****************************************************************
*****************************

TASK
[GatherinFacts*********************************************************
```

```
*************************************************************
ok: [localhost
TASK [Installing required 'pip'
modules*****************************************************************
**********************************
ok: [localhost]
...
TASK [Running setup via docker-
compose.yml]************************************************************
************************************
changed: [localhost]
PLAY
RECAP*******************************************************************
*****************************************************************
localhost                  : ok=7    changed=5    unreachable=0
failed=0    skipped=0   rescued=0    ignored=0
```

In order to ensure that the Backend SDK app was created properly, use the following docker commands to check for containers currently running.

```
docker ps -a
```

You should see two new containers as a result, one labeled as 'hyperprotectbackendsdk', and the other container as 'mongo'.

Copy the Container ID serial number for the 'hyperprotectbackendsdk' from the above output, and use the *docker logs* command to verify the logs. If the app is running properly from the local machine, you will see output stating that the example app is running on port 3000.

```
$ docker  ps -a
CONTAINER ID        IMAGE                                  COMMAND
CREATED              STATUS              PORTS
NAMES
1072137b9f78        hyperprotectbackendsdk-test_app   "docker-
entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3000-
>3000/tcp, 27017/tcp   app
fdd1014096ce        mongo                                  "docker-
entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:27017-
>27017/tcp            mongo
$ docker logs  1072137b9f78

> hyper-protect-sdk-backend@0.0.1 start /usr/app/carekit-hyperprotect
> set debug=* && ts-node-dev --respawn --transpileOnly ./src/index.ts

Using ts-node version 8.10.1, typescript version 3.8.
```

```
UUID : 7263588F-09C1-488C-9248-539060C0D124
Example app listening on port 3000! Go to https://localhost:3000/
```

Validation Test

To validate that the app is running properly, a simple curl command can be issued to for verification. Please make certain to change the `{HPVS_IP_or_locahost}` to either the HPVS public IP address, or *localhost* if the local_setup was run.

This curl command should be executed from the local machine, while pointing the *cacert* argument at the rootCA.crt file, as shown below.

Curl Command

```
curl --cacert rootCA.crt --location --request POST
'https://{HPVS_IP_or_locahost}:3000/revisionRecord' \
--header 'Content-Type: application/json' \
--data-raw verification.json
```

After the curl command has been issued, if successful you will see a `RevisionRecord stored` message.

Another verification check is to check the Docker container logs on the Virtual Server, or localhost. The docker logs can be checked as such:

1. Check docker for running containers

   ```
   docker ps -a
   ```

2. Locate the Docker container ID

   ```
   root@b4e8f18c497b:~# docker ps -a
   CONTAINER ID        IMAGE                                COMMAND
   CREATED             STATUS
   c876afbe4704        hyperprotectbackendsdktest_app   "docker-
   entrypoint.s…"   2 hours ago         Up 2 hours
   ```

3. Use container ID from previous step to check current log entries

   ```
   docker logs {Container_ID}
   ```

4. View the hyperprotectbackendsdk Docker logs, traces of the curl command should be present.

> **You now have the IBM Hyper Protect MBaaS running in IBM Hyper Protect Virtual Servers**

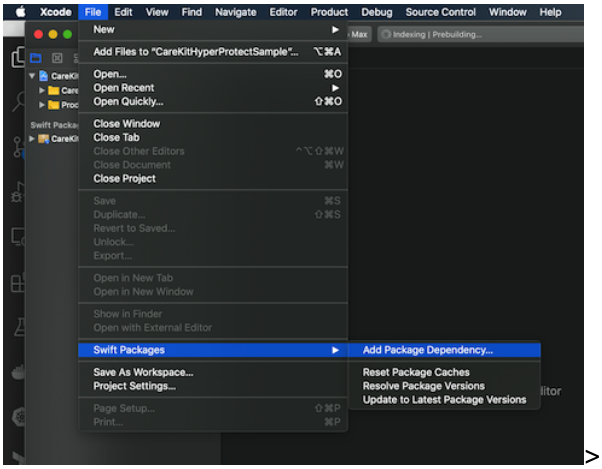
Integrate IBM Hyper Protect SDK for iOS into the Sample App

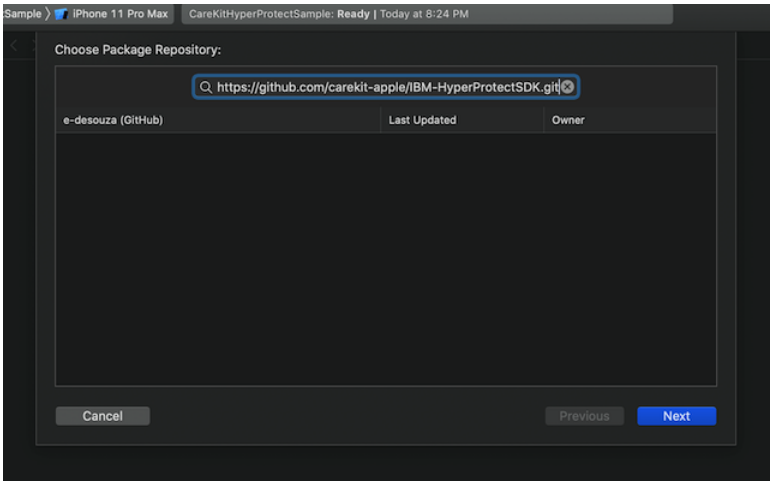
This SDK implements CareKit's Remote Synchronization API and must be coupled with the backend SDK IBM-HyperProtectMBaaS on the server side.

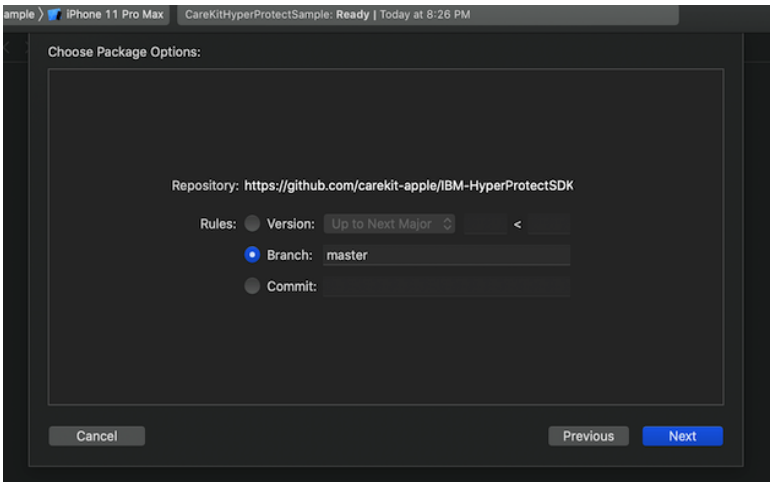
Note, this is a pre-1.0 release and is still in beta

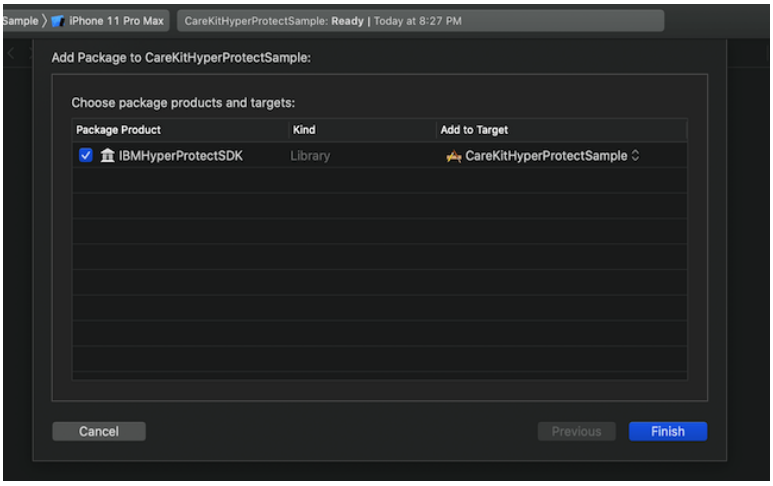
Setup

This package can be imported into XCode using Swift Package Manager:









Now import the package with

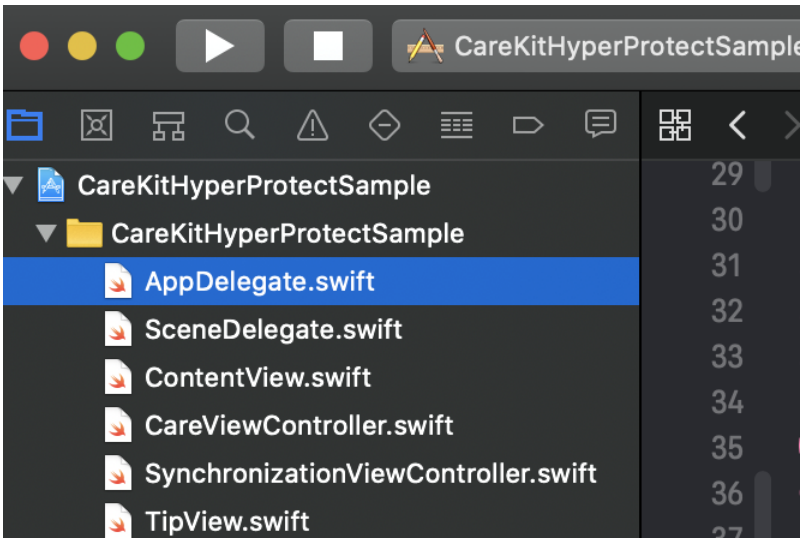
```
import IBMHyperProtectSDK
```

and pass it in to your OCKStore:

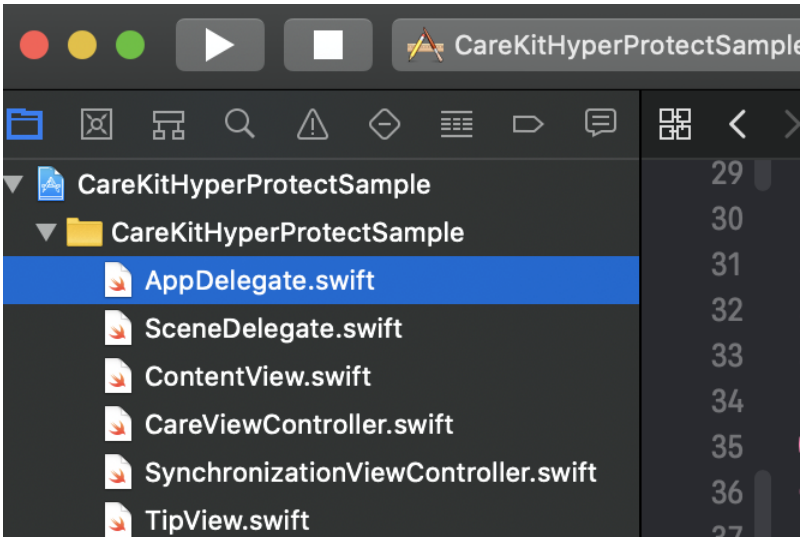
```
let remote = IBMMongoRemote()
let store = OCKStore(name: "SampleAppStore", type:
  inMemory, remote: remote)
```

By default if no backend API information is passed in, it will default to `https://localhost:3000` . Pass in the `apiLocation` parameter to point to your IBM Hyper Protect MBaaS deployed locally for development or in IBM Cloud.

To test synchronization with the MBaaS, run the app and select some outcomes:



Next, stop your app but clicking the square icon at the top left of XCode.



Comment out the programmatic generation of tasks on line 44 of AppDelegate.swift:

```
    // Manages synchronization of a CoreData store
    lazy var synchronizedStoreManager: OCKSynchronizedStoreManager = {
        let remote = IBMMongoRemote(apiLocation: [your MBaaS location],
apiTimeOut: 2.0)
        let store = OCKStore(name: "SampleAppStore", type: .inMemory,
remote: remote)
        //store.populateSampleData() // COMMENT THIS
        let manager = OCKSynchronizedStoreManager(wrapping: store)
        return manager
    }()
```

Start the app again and notice how it has no Tasks populated. Hit the Synchronize button and if everything worked according to plan, you should see a success message like:

If you swipe down on that message, you should now see the tasks and outcomes you entered during your last run! This can be done across multiple devices and conflicts are automatically resolved!

> **🎉 That's it ! In just two lines of code, your CareKit app has bi-directional synchronization where the data is stored in a zero-trust environment protected by industries only FIPS140-2 Level 4 compliant HSM in a public cloud! 🎉**

Troubleshooting

There are a few potential issues that could arise while running the Ansible automation.

Failure on docker-compose task

Error:

```
TASK [Running setup via docker-compose.yml]
*********************************************************************************
**************************
fatal: [169.63.212.61]: FAILED! => {"changed": false, "msg": "Failed to
import the required Python library (Docker SDK for Python: docker (Python
>= 2.7) or docker-py (Python 2.6)) on e7c65ce8b4cd's Python
/usr/bin/python3. Please read module documentation and install in the
appropriate location. If the required library is installed, but Ansible is
using the wrong Python interpreter, please consult the documentation on
ansible_python_interpreter, for example via `pip install docker` or `pip
install docker-py` (Python 2.6). The error was: No module named
'requests'"
```

Explanation and Resolution:

The Hyper Protect Virtual Server has just installed the docker daemon for the first time. This particular error can be somewhat misleading, as it appears to be missing a pip/python package. However, this error is due to the docker daemon being installed, but not enabling quick enough before docker-compose is called.

- Wait 1-2 minutes and rerun the ansible playbook, no changes are required!

Failure on validation steps using rootCA.crt

Error:

```
curl: (35) error:04FFF084:rsa routines:CRYPTO_internal:data too large for
modulus
```

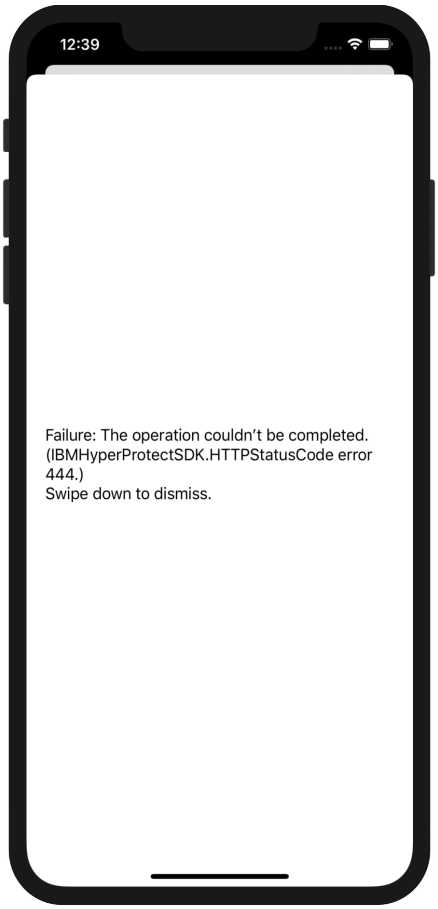
Resolution:

The SSL certificates were either created too quickly, causing the signing to fail, or the *rootCA.crt* file did not copy from the Virtual Server to the local machine properly.

```
scp root@{Public_IP_Address}:/root/HyperProtectBackendSDK/certs/rootCA.crt
{directory_for_ansible}
```

IBMHyperProtectSDK.HTTPStatusCodeError 444

Error:



Resolution:

444 stands for Connection Closed Without Response. This typically means your app couldn't connect to your MBaaS backend. It could also mean you are trying to make a HTTP connection instead of a HTTPS connection. You could add this to Info.plist

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
    <key>NSExceptionDomains</key>
    <dict>
        <key>example.com</key>
        <dict>
            <key>NSExceptionAllowsInsecureHTTPLoads</key>
            <true/>
            <key>NSIncludesSubdomains</key>
            <true/>
        </dict>
    </dict>
</dict>
```

Note, you should NEVER do this in production.