

# 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
  - Connect to your Virtual Server
- Deploy a IBM Hyper Protect DBaaS for MongoDB instance
- Integrate IBM Hyper Protect with the Sample App
  - IBM Hyper Protect MBaaS
  - Prerequisites
  - Initial setup to prepare for running ansible
  - IBM Hyper Protect Virtual Server Bootstrapping with Ansible
  - Bootstrapping your local dev environment
  - Validation Test
- 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:

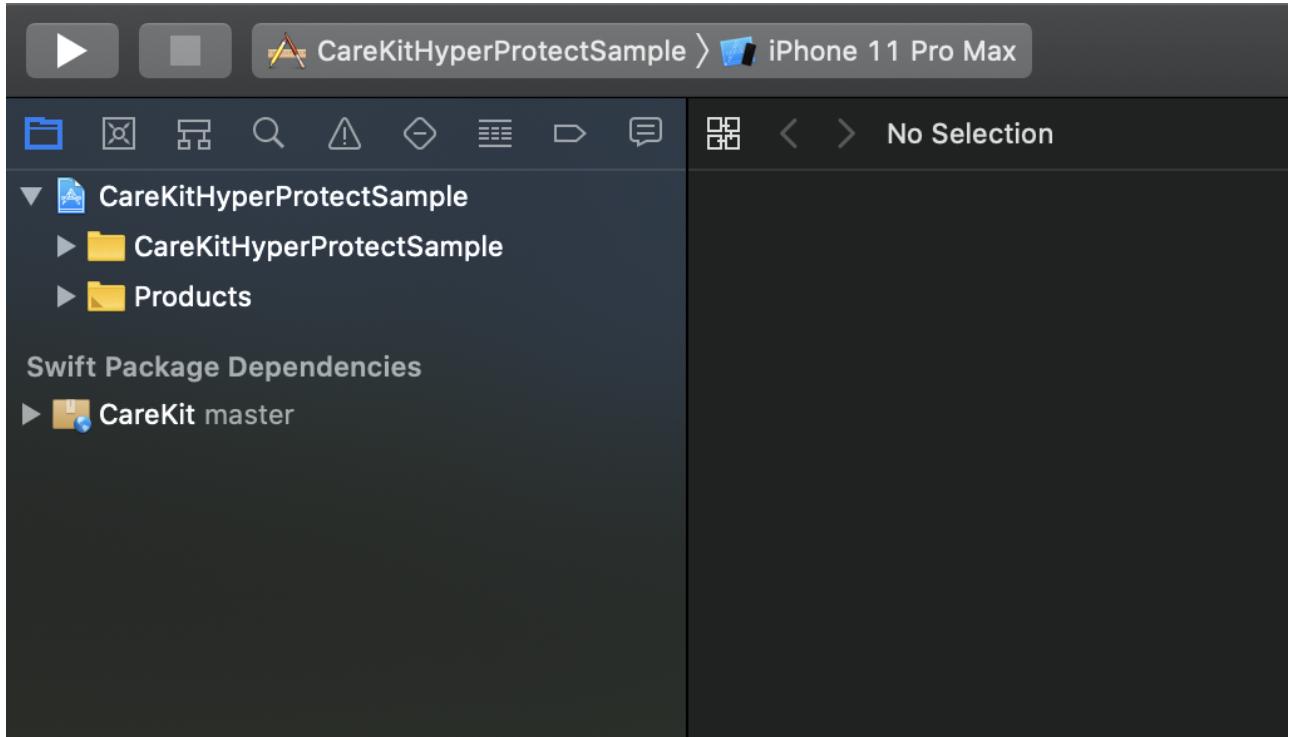
An IBM Cloud Account is required for this lab. If you need to create one, please go to :

<https://cloud.ibm.com/registration> and follow the prompts. Additional documentation can be found [here](#)

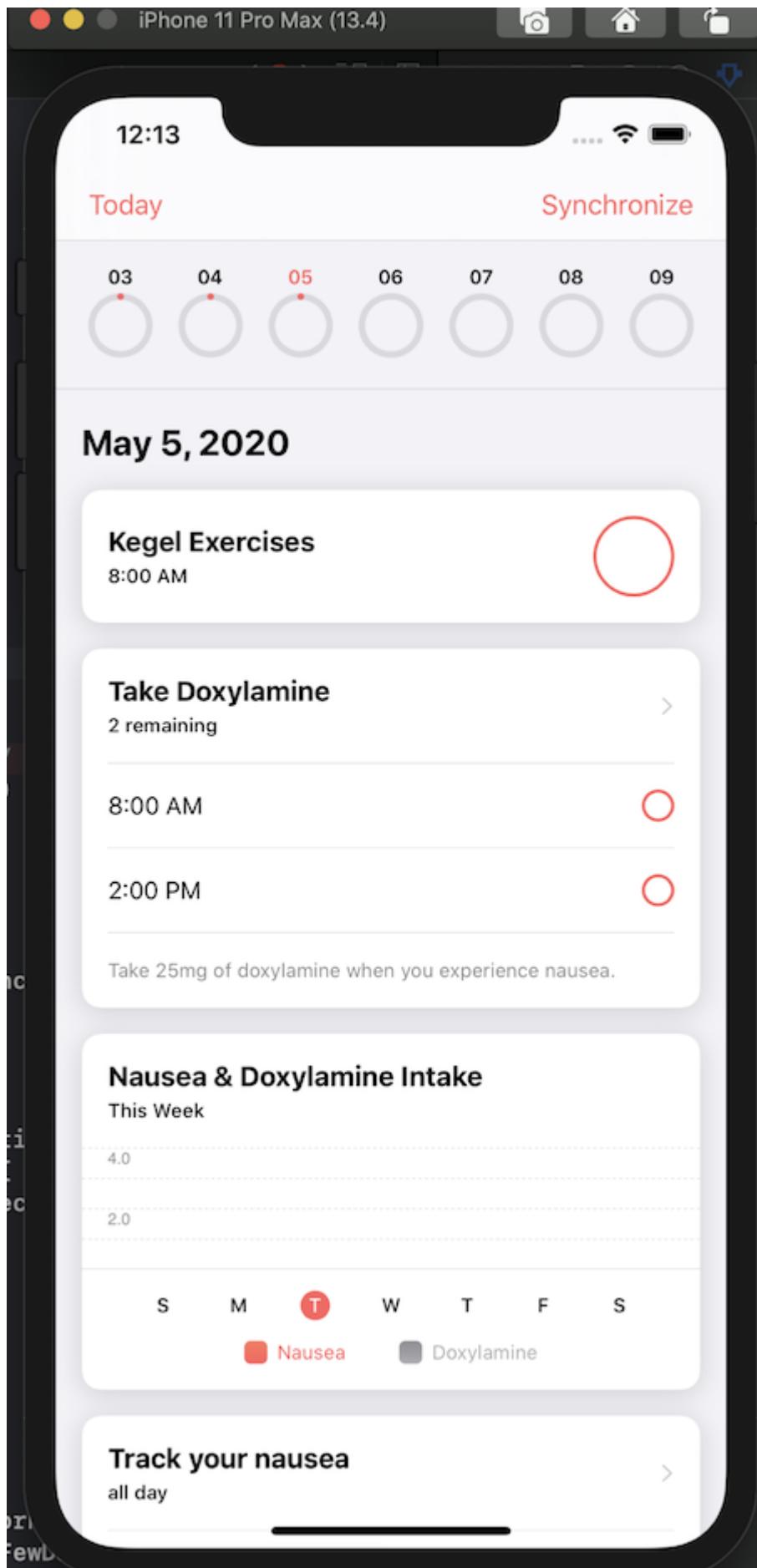
# Deploy a Sample App with CareKit

CareKit requires XCode 14.3+

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:



5. 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)!

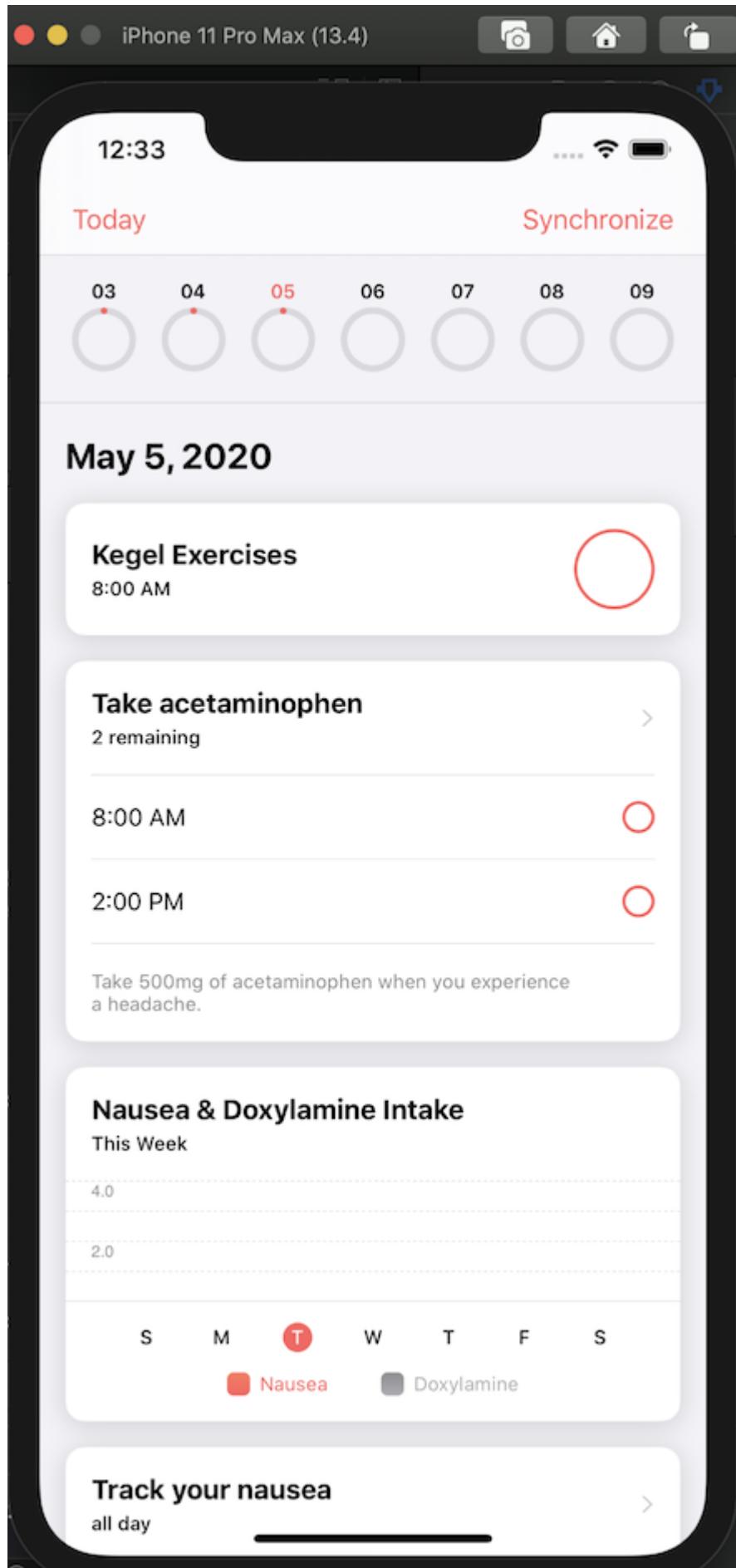
let schedule = OCKSchedule(composing: [
    OCKScheduleElement(start: beforeBreakfast, end: nil,
                        interval: DateComponents(day: 1)),
    OCKScheduleElement(start: afterLunch, end: nil,
                        interval: DateComponents(day: 2))
])

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()`.
  - o Changing text will be the easiest. You can change the **title** or **instruction** of the existing **doxylamine** **OCKTask**.
  - o 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.

8. Push build on your new app. In our case, we changed the text to acetaminophen (Tylenol)



# 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>)
  - o 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.
  - o The desired destination is the Cloud catalog, where all current available Cloud offerings are listed
  - o 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'.
  - o The page defaults to the 'Create' tab, where the various Virtual Server options are able to be selected.
  - o 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*
  - o Add any necessary tags for labeling the virtual server
    - Copy and paste your public ssh key 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

## Connect to your Virtual Server

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

- 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.
  - o 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'

# 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

>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](#)
- 

# 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

1. Install ansible via pip3 (or pip)
  - In order to run this playbook on the local machine *ansible* is a required package.
  - Command for installation: *pip3 install ansible*
    - Please note that this command uses pip3, if you still have python2 as the default python interpreter, use 'pip' in lieu of pip3
2. Install docker-compose via pip3
  - *pip3 install docker-compose*
3. Install docker
  - *pip3 install docker*

## Initial setup to prepare for running ansible

1. Clone the repository on the local machine `git clone https://github.com/e-desouza/carekit-hyperprotect-lab.git` This github repository containing the ansible playbooks has been made public in order to clone this repository
2. After the Github repo has been cloned to the local machine, go to the `carekit-hyperprotect-lab/ansible_setup` directory
3. There are two ansible playbooks that can be run for bringing up the CareKit Backend SDK application. The first uses a HyperProtect Virtual Server, and DBaaS MongoDB instance. While the second option will configure the Backend SDK app on the local machine, uses localhost. Choose the next set of instructions accordingly.

## IBM Hyper Protect Virtual Server Bootstrapping with Ansible

Requires HPVS and MongoDB IBM Cloud instances

1. Two critical additions to the yml files must be made in order to successfully run the `hpvs*setup.yml` playbook. Add the following values accordingly:

- ansible.cfg
- Within the ansible.cfg file, the value for environmental variable `_ansible*ssh_private_key_file*` is currently an empty string. Add the location and name of your public SSH key used to provision the Hyper Protect Virtual Server
  - Example: `ansible_ssh_private_key_file = "~/ssh/id_rsa.pub"`

```
[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 = "~/ssh/id_rsa.pub"
```

\_ inventory.yml \_ Underneath the commented line, add the public IP address of the Hyper Protect Virtual Server \_ Example:

```
[hosts]
#add HPVS public IP below this comment
169.63.212.70
```

2. One final preparation step is required prior to running the ansible playbook. The DBaaS MongoDB admin ID and password needs to be added to the mongo URI, as this value will be passed into the command line when invoking the playbook. \_ Ensure that the entire string including all 3 replica hosts are in the uri string, and also the Cluster name at the end. \_ Example:

```
mongodb://admin:password@dbaas30.hyperp-
dbaas.cloud.ibm.com:28162,dbaas29.hyperp-
dbaas.cloud.ibm.com:28130,dbaas31.hyperp-dbaas.cloud.ibm.com:28222/admin?
replicaSet=Wu-Tang_Clan * Replace 'admin' and 'password' with the proper admin ID and
correlated password into the MongoDB URI string provided after provisioning the DBaaS instance.
```

3. Now that the Public IP address has been added as a listed host, and the location of the public SSH key was specified, it is now time to run the ansible playbook for setup. Use the following command in order to run the playbook properly.

- Notice that the initial portion of the extra variable being passed to the playbook starts with `db=`. This is required to specify which variable is being passed to the ansible playbook.
- Command Example:

```
ansible-playbook hpus_setup.yml -e "db=mongodb://{{admin_ID}}:
{{Mongo_Password}}@DBaaS_Mongo_URI:port.../admin?
replicaSet=Cluster_Example"
```

```
Ryleys-MacBook-Pro:HyperProtectSDK_Ansible ryley.wharton1ibm.com$ ansible-playbook hpvs_setup.yml -e "db=mongodb://admin:MongoPassword@dbaas30.hyperp-dbaas.cloud.ibm.com:28162,dbaas29.hyperp-dbaas.cloud.ibm.com:28130,dbaas31.hyperp-dbaas.cloud.ibm.com:28222/admin?replicaSet=Cluster_1"
```

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

```
PLAY [Configure Hyper Protect Virtual Server] ****
TASK [Gathering Facts] ****
ok: [169.63.212.70]

TASK [Install required system packages] ****
changed: [169.63.212.70] => (item=curl)
changed: [169.63.212.70] => (item=npn)
changed: [169.63.212.70] => (item=docker.io)
changed: [169.63.212.70] => (item=ufw)
changed: [169.63.212.70] => (item=docker-compose)
ok: [169.63.212.70] => (item=git)

TASK [Clone github repo (SDK CareKit)] ****
changed: [169.63.212.70]

TASK [Create file on HPVS container for .env test] ****
changed: [169.63.212.70]

TASK [Copy 'cert.pem' file to the app's src/ directory] ****
changed: [169.63.212.70]

TASK [Running setup via docker-compose.yml] ****
changed: [169.63.212.70]

PLAY RECAP ****
169.63.212.70 : ok=6    changed=5    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 your local dev environment

This section is only here for completeness. You will be using the services from IBM Cloud instead of localhost. Feel free to skip this step and jump to [Validation Tests](#)

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

- Unlike the 'hpvssetup.yml' playbook, the local setup already has the correct \_localhost configurations written within the playbook itself, and does not require any other changes to the inventory file, nor ansible.cfg. Please use the listed command below to run the ansible script.
- Command: `ansible-playbook local_setup.yml -K`
  - The -K argument is required due to the playbook utilizing the "Become" parameter. Enter the local machine password (user profile credentials) when prompted.

```
Ryleys-MacBook-Pro:ansible_setup ryley.wharton1ibm.com$ ansible-playbook local_setup.yml -K
BECOME password: ?
```

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

```

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Installing required 'pip' modules] *****
ok: [localhost]

TASK [Clone github repo (SDK CareKit)] *****
changed: [localhost]

TASK [Create file on HPVS container for .env test] *****
changed: [localhost]

TASK [Copy 'cert.pem' file to the app's src/ directory] *****
changed: [localhost]

TASK [Creating SSL certificates] *****
changed: [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

```

3. In order to ensure that the Backend SDK app was created properly, use the following docker commands to check.

```
docker ps -a
```

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

4. Copy the Container ID serial number from the above output, and use the *docker logs* command to check 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.

```
Ryleys-MacBook-Pro:ansible_setup ryley.wharton1ibm.com$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
5f1068917cbd      hyperprotectbackendsdk-test_app   "docker-entrypoint.s..."   3 minutes ago     Up 3 minutes       0.0.0:3000->3000/tcp, 27
017/tcp           app                  "docker-entrypoint.s..."   3 minutes ago     Up 3 minutes       0.0.0:27017->27017/tcp
4560bafb5813      mongo                "docker-entrypoint.s..."   3 minutes ago     Up 3 minutes       0.0.0:27017->27017/tcp
mongo

Ryleys-MacBook-Pro:ansible_setup ryley.wharton1ibm.com$ docker logs 5f1068917cbd
> 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 3.8.3
(node:25) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
UUID : 0E11D8B9-9043-4A6A-85AF-2F8B7F508B62
Example app listening on port 3000! Go to https://localhost:3000/
```

## Validation Test

To validate that the app is running properly, and listening on port 3000, a simple curl command can be issued to for verification. Please make certain that the IP address is changed in the http address after the POST declaration, as the goal is to hit the running application using the pulic IP addresss of the Virtual Server.

- Copy the entire curl command below, after replacing the `{HPVS_IP_or_localhost}` parameter depending on which setup playbook was run.
  - If '`localsetup.yml`' was used for configuration, change the parameter to `_localhost`
  - If '`hpvs_setup.yml`' was used instead, simply change the parameter to the public IP address of the HPVS instance
- Run the curl command from local machine, this will validate the Backend SDK application is reachable, and taking requests.
- If the test is successful, a returned output of 'RevisionRecord stored' will populate after the curl command.
  - Also, within the docker container logs, the POST call will come through, and a `201` code will be returned.
  - Command to view docker logs: `docker logs`
    - ContainerID can be found by running `docker ps -a` and locating the HyperProtectBackendSDK container

### Curl Command

```
curl --cacert rootCA.crt --location --request POST  
'https://'{HPVS_IP_or_localhost}:3000/revisionRecord' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
    "entities": [  
        {  
            "type": "task",  
            "object": {  
                "schemaVersion": {  
                    "majorVersion": 2,  
                    "minorVersion": 0,  
                    "patchNumber": 4  
                },  
                "id": "nausea",  
                "uuid": "75EE244A-7303-43CF-9AA5-6CC3BB81210A",  
                "createdDate": 609212115.685683,  
                "updatedDate": 609212115.685702,  
                "title": "Track your nausea",  
                "notes": [],  
                "timezone": {  
                    "identifier": "America/Sao_Paulo"  
                },  
                "instructions": "Tap the button below anytime you  
experience nausea.",  
                "impactsAdherence": false,  
                "effectiveDate": 608785200,  
                "lastModified": 609212115.685702  
            }  
        }  
    ]  
}
```

```
"schedule": {
    "elements": [
        {
            "text": "Anytime throughout the day",
            "duration": {
                "isAllDay": true
            },
            "interval": {
                "minute": 0,
                "hour": 0,
                "second": 0,
                "day": 1,
                "month": 0,
                "year": 0,
                "weekOfYear": 0
            },
            "targetValues": [],
            "start": 608785200
        }
    ]
},
{
    "type": "task",
    "object": {
        "schemaVersion": {
            "majorVersion": 2,
            "minorVersion": 0,
            "patchNumber": 4
        },
        "id": "doxylamine",
        "uuid": "C0861A29-C726-4B58-B3AB-89CF3E3294F6",
        "createdDate": 609212115.696223,
        "updatedDate": 609212115.696224,
        "title": "Take Doxylamine",
        "notes": [],
        "timezone": {
            "identifier": "America/Sao_Paulo"
        },
        "instructions": "Take 25mg of doxylamine when you experience nausea.",
        "impactsAdherence": true,
        "effectiveDate": 608814000,
        "schedule": {
            "elements": [
                {
                    "duration": {
                        "seconds": 0,
                        "isAllDay": false
                    },
                    "interval": {
                        "minute": 0,
                        "hour": 0,
                        "second": 0
                    }
                }
            ]
        }
    }
}
```

```

        "second": 0,
        "day": 1,
        "month": 0,
        "year": 0,
        "weekOfYear": 0
    },
    "targetValues": [],
    "start": 608814000
},
{
    "duration": {
        "seconds": 0,
        "isAllDay": false
    },
    "interval": {
        "minute": 0,
        "hour": 0,
        "second": 0,
        "day": 2,
        "month": 0,
        "year": 0,
        "weekOfYear": 0
    },
    "targetValues": [],
    "start": 608835600
}
]
}
}
],
"knowledgeVector": {
    "processes": [
        { "id" : "1C43F648-D41A-4A5A-8708-15737425FA7C", "clock" : 10},
        { "id" : "2B43F648-D41A-4A5A-8708-15737425FA7C", "clock" : 4}
    ]
}
}'
```

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              PORTS
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 Docker logs, traces of the curl command should be present as depicted in the screenshot below.

```
start: 608814000 } ] } } } ],
knowledgeVector:
{ processes:
  [ { id: '1C43F648-D41A-4A5A-8708-15737425FA7C', clock: 10 },
    { id: '2B43F648-D41A-4A5A-8708-15737425FA7C', clock: 4 } ] } }
::ffff:174.109.82.91 - POST /revisionRecord HTTP/1.1 201 21 - 121.263 ms
root@b4e8f18c497b:~#
```

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

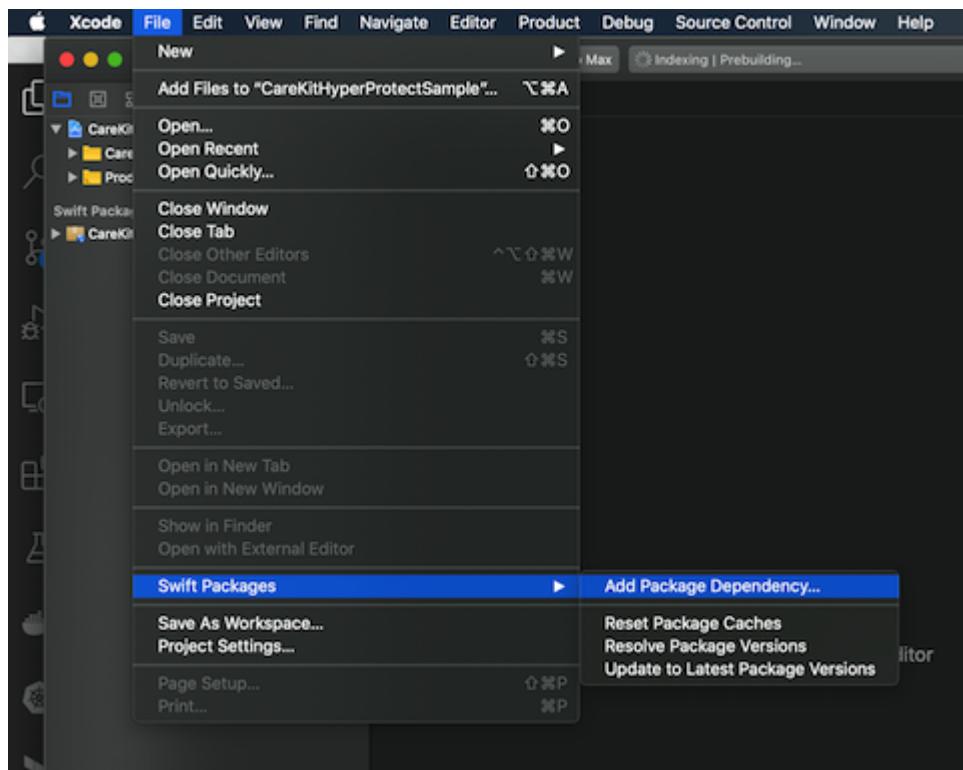
---

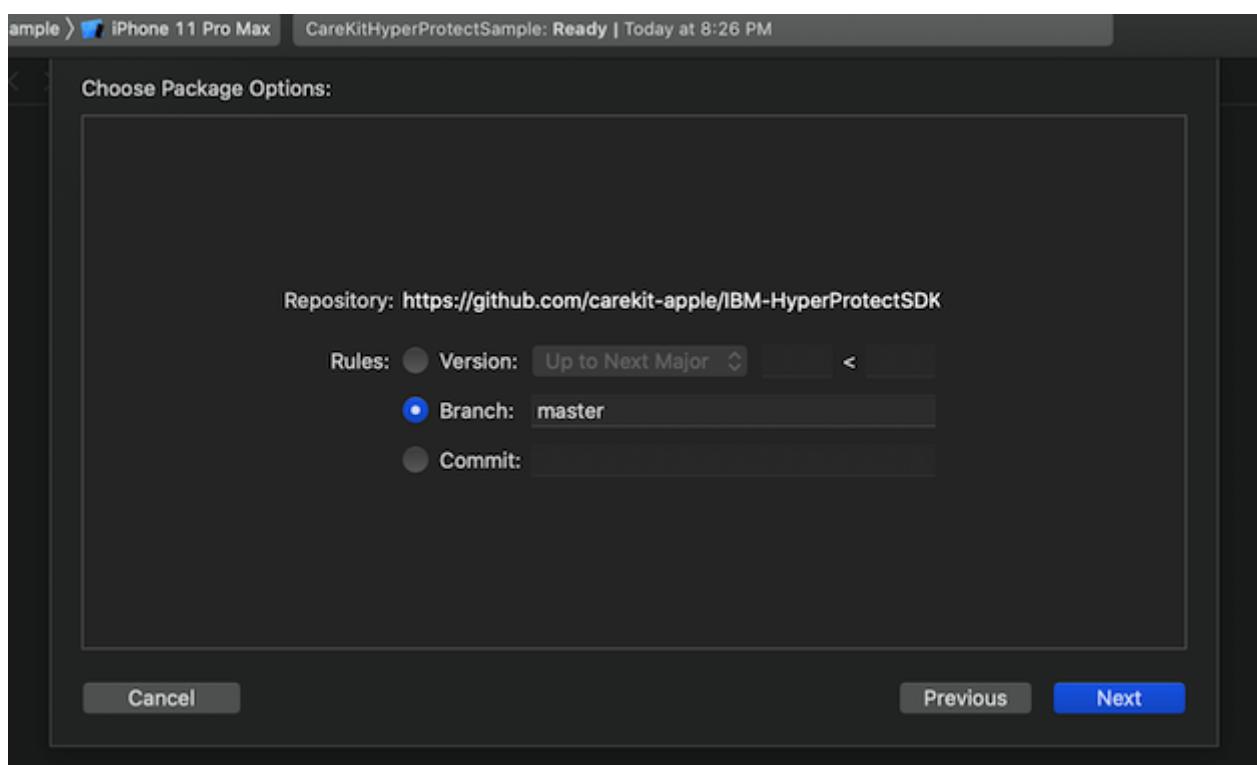
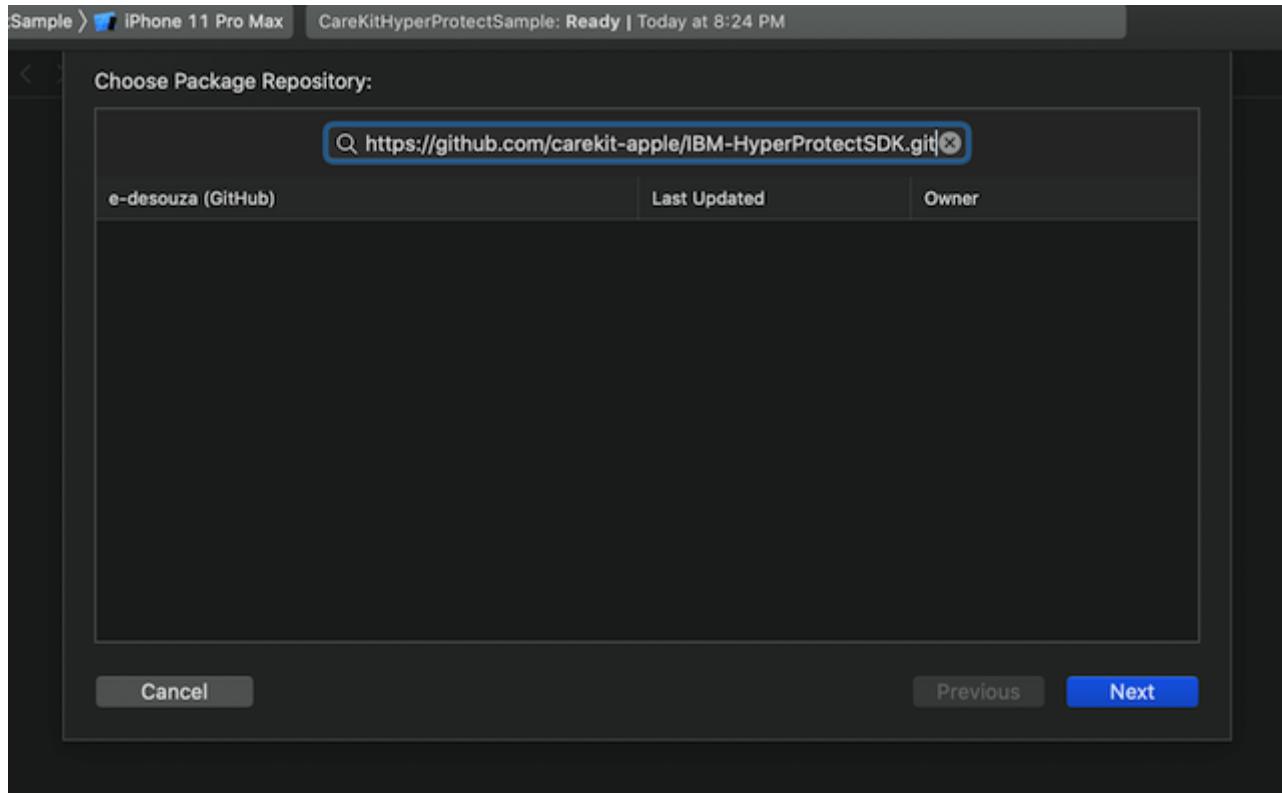
This SDK implements CareKit's Remote Synchronization API and must be coupled with the backend SDK [IBM-HyperProtectMBaaS](#) on the server side.

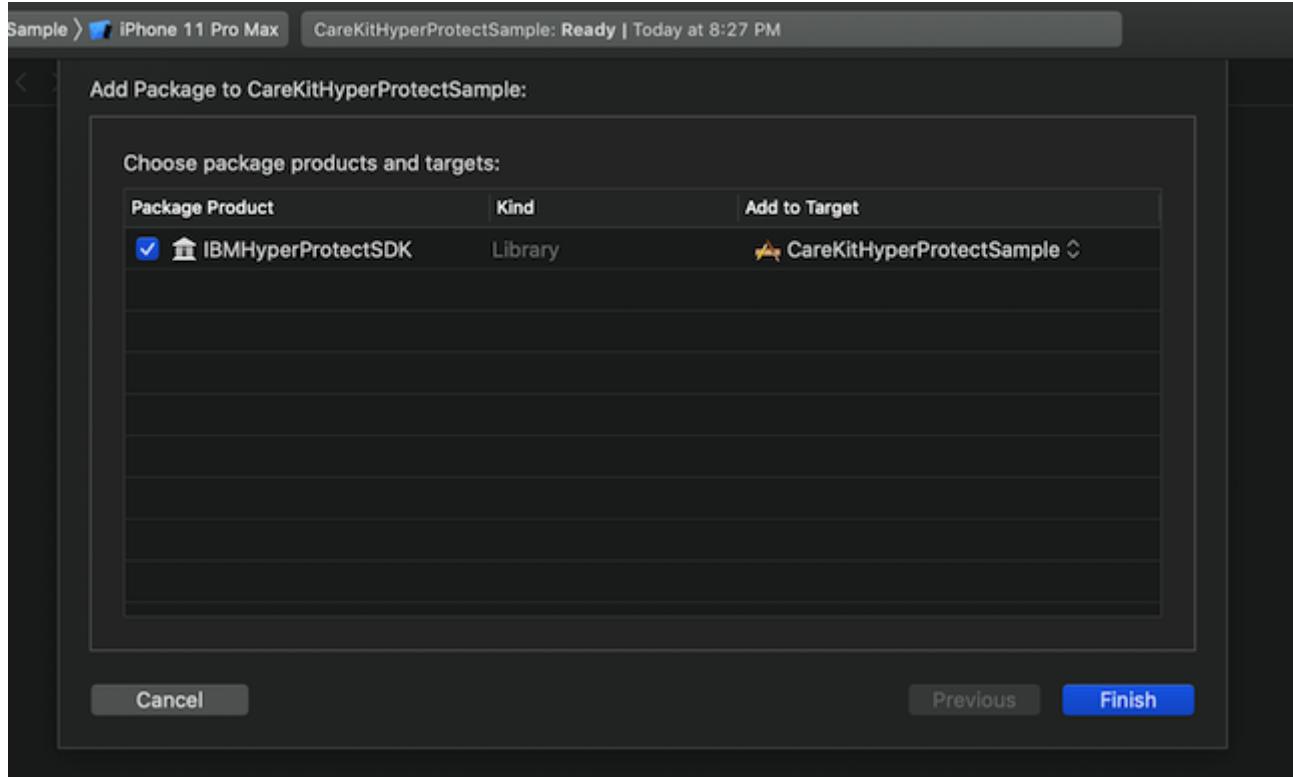
*Note, this is a pre-1.0 release and is still in beta*

## 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.

TODO

# 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]
*****
*****致命错误：[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}