

# Snappy Ubuntu Core

- Required software components
- Topology
- Configuring and building sample Kaa client
- Building snappy package
- Installing snappy Ubuntu
- Installing sample Kaa client on snappy Ubuntu
- Further reading

The following instructions explain how to build a Kaa endpoint SDK and integrate it with snappy Ubuntu Core.

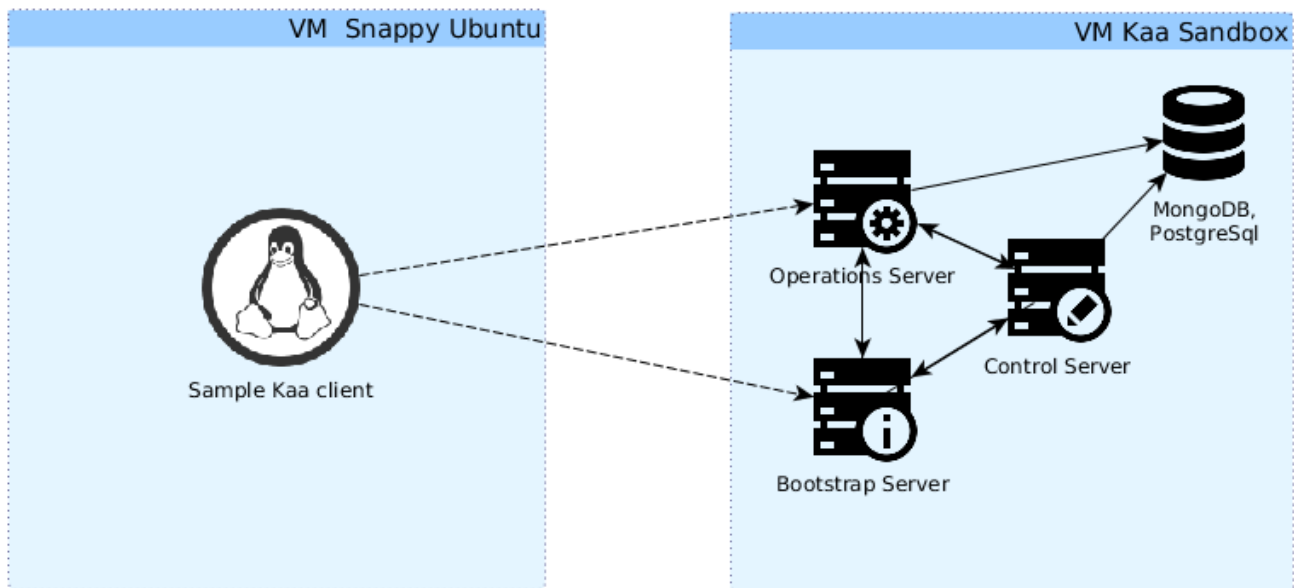
## Required software components

Ensure the following components are installed on your machine.

- VirtualBox
- Vagrant
- Latest Kaa sandbox

## Topology

To integrate Kaa with snappy Ubuntu Core, you need to use two virtual boxes: one to deploy snappy Ubuntu and a Kaa client and the other to deploy the Kaa sandbox.



## Configuring and building sample Kaa client

We assume that the [Kaa sandbox](#) is already started.

NOTE: Make sure that the Kaa client is able to connect to Bootstrap and Operations servers deployed on the Kaa sandbox.

To configure and build a sample Kaa client, proceed as follows:

1. Go to <http://localhost:8080/sandbox/>.
2. Enter the IP address of your Kaa sanbox VM into the text field and then click **Change** (in the following screenshot, the IP address is 192.168.100.100).

## Kaa Sandbox - Try Kaa in action!

### Main console

Go to [Kaa Administrative Web Console](#)

Go to [Avro UI Sandbox Web Console](#)

To change kaa services host/ip enter new host value in field below and click 'Change' button.

### Demo projects

#### Smart House Demo

Platform: ANDROID

Smart house application on android platform demonstrating event subsystem (IoT)

3. Wait till the service restarts.
4. Get the latest Kaa code from [github](#).

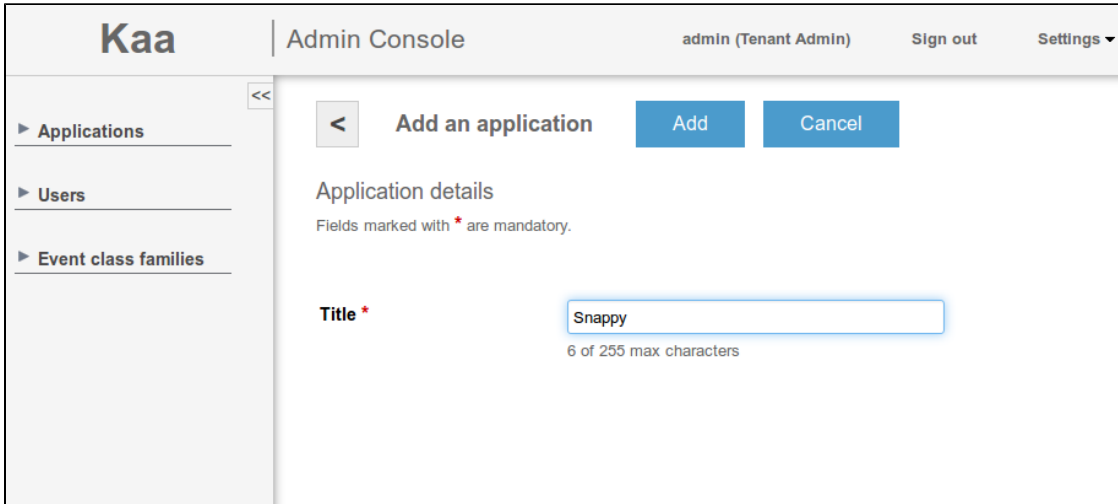
```
$ git clone git@github.com:kaaproject/kaa.git
$ cd kaa/examples/
```

5. Configure and generate C SDK for your Kaa client.  
NOTE: You can find sample Avro schemas in `c-sdk-sample/avro/path`.

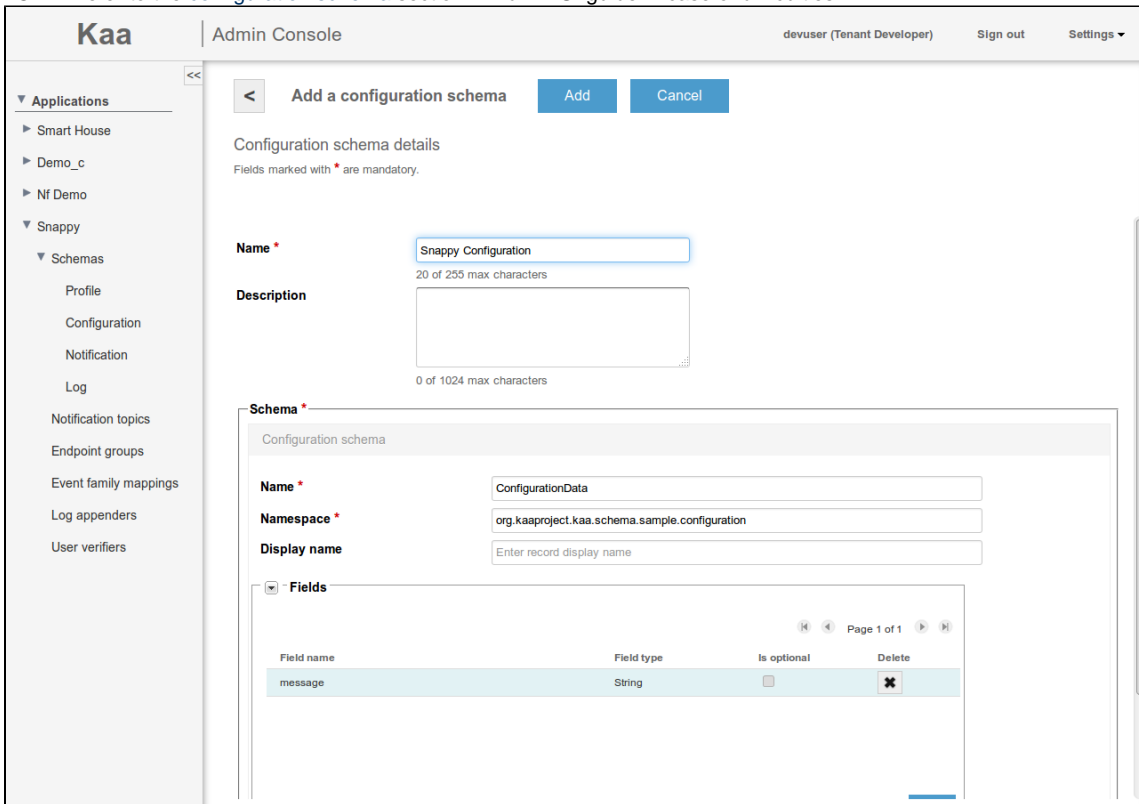
### Avro schemas

```
$ ls c-sdk-sample/avro/
configuration.avsc log.avsc profile.avsc
```

6. Create a snappy application in web UI.  
NOTE: Refer to [managing applications](#) in case of difficulties.



7. Add a new configuration schema for your application (look at [configuration.avsc](#)).  
NOTE: Refer to the [configuration schema](#) section in Admin UI guide in case of difficulties.



8. Add a new profile schema for your application (look at [profile.avsc](#)).  
NOTE: Refer to the [profile schema](#) section in Admin UI guide in case of difficulties.

Kaa Admin Console devuser (Tenant Developer) Sign out Settings

Applications

- Smart House
- Demo\_c
- Nf Demo
- Snappy
  - Schemas
    - Profile
    - Configuration
    - Notification
    - Log
    - Notification topics
    - Endpoint groups
    - Event family mappings
    - Log appenders
    - User verifiers

Add a profile schema Add Cancel

Profile schema details  
Fields marked with \* are mandatory.

Name \* Snappy Profile  
14 of 255 max characters

Description  
0 of 1024 max characters

Schema \*

Schema

Name \* Profile

Namespace \* org.kaaproject.kaa.schema.sample.profile

Fields

Field name	Field type	Is optional	Delete
id	String	<input type="checkbox"/>	✕
os	Enumeration	<input type="checkbox"/>	✕
os_version	String	<input type="checkbox"/>	✕
build	String	<input type="checkbox"/>	✕

Add

9. Add a new log schema for your application (look at [log.avsc](#)).

NOTE: Refer to the [log schema](#) section in Admin UI guide in case of difficulties.

Kaa Admin Console devuser (Tenant Developer) Sign out Settings

Applications

- Smart House
- Demo\_c
- Nf Demo
- Snappy
  - Schemas
    - Profile
    - Configuration
    - Notification
    - Log
    - Notification topics
    - Endpoint groups
    - Event family mappings
    - Log appenders
    - User verifiers

Add log schema Add Cancel

Log schema details  
Fields marked with \* are mandatory.

Name \* Snappy Log  
10 of 255 max characters

Description  
0 of 1024 max characters

Schema \*

Schema

Name \* LogData

Namespace \* org.kaaproject.kaa.schema.sample.logging

Fields

Field name	Field type	Is optional	Delete
level	Enumeration	<input type="checkbox"/>	✕
tag	String	<input type="checkbox"/>	✕
message	String	<input type="checkbox"/>	✕

Add

10. Generate the C SDK for your application.

NOTE: Refer to the [generating SDK](#) section in Admin UI guide in case of difficulties.

The screenshot shows the Kaa Admin Console interface. At the top, it says 'Kaa Admin Console' with a user 'devuser (Tenant Developer)' and options for 'Sign out' and 'Settings'. The left sidebar has a tree view under 'Applications' and 'Schemas'. The main area is titled 'Generate SDK' and contains a form for 'SDK details'. The form includes the following fields:

- Configuration schema version \* (dropdown: 2.0)
- Profile schema version \* (dropdown: 2.0)
- Notification schema version \* (dropdown: 1.0)
- Log schema version \* (dropdown: 2.0)
- Target platform \* (dropdown: C)
- Event class families (checkbox)
- Default user verifier (dropdown)

Buttons for 'Generate SDK' and 'Cancel' are visible at the top right of the form area.

11. Build a Kaa client with the generated C SDK. Remove everything in the c-sdk-sample/libs/kaa folder and unzip the generated SDK to that folder.

```
Build C Kaa client  
  
$ rm -rf kaa/examples/c-sdk-sample/libs/kaa/* # remove existing files  
$ tar -cvzf kaa-client-sdk-p2-c2-n1-l2.tar.gz  
kaa/examples/c-sdk-sample/libs/kaa/ # untar Kaa sdk  
$ ./build.sh clean build #build client
```

If the build is successful, you will get the result on your screen as follows:

## Build result

```
[100%] Built target kaac_s
-- The C compiler identification is GNU 4.8.2
-- The CXX compiler identification is GNU 4.8.2
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Found OpenSSL:
/usr/lib/x86_64-linux-gnu/libssl.so;/usr/lib/x86_64-linux-gnu/libcrypto.so
(found version "1.0.1f")
-- Configuring done
-- Generating done
-- Build files have been written to:
/home/igor/code/github/kaa/examples/c-sdk-sample/build
Scanning dependencies of target sample_c_client
[100%] Building C object CMakeFiles/sample_c_client.dir/src/kaa_demo.c.o
Linking C executable sample_c_client
[100%] Built target sample_c_client
```

## Building snappy package

To build a snap package, proceed as follows:

1. Install snappy Ubuntu tools by executing the following commands either from Ubuntu or from the Kaa sandbox (in case you are using another OS). Building inside Kaa sandbox is preferable, because system upgrade step may harm your system.

### Install Snappy tools

```
$ sudo add-apt-repository ppa:snappy-dev/beta
$ sudo apt-get update
$ sudo apt-get upgrade #do it for your own risk, you better to use virtual
machine
$ sudo apt-get install snappy-tools
```

2. Create the correct folder structure. Refer to the official snappy Ubuntu Core documentation for details.

```
$ mkdir snappy-kaa
$ mkdir snappy-kaa/meta
$ mkdir snappy-kaa/bin
```

3. Copy the previously built Kaa client to the snappy-kaa/bin directory.

```
$ cp kaa/examples/c-sdk-sample/build/sample_c_client snappy-kaa/bin/client #  
Copy with rename sample client to bin directory
```

4. Create the package.yaml file in the snappy-kaa/meta directory with the following context.

```
package.yaml  
name: kaa  
vendor: Vendor Name <info@kaaproject.org>  
icon: meta/kaa.png  
version: 0.0.1  
binaries:  
- name: bin/client  
  description: "snappy example: kaa c client"
```

5. Create the readme.md file in the snappy-kaa/meta directory with the following context.

```
readme.md  
Minimal C Kaa client for snappy  
  
Any other info about your application.
```

6. Build a snap package from the snappy-kaa directory.

```
Build snappy package  
$ snappy build snappy-kaa
```

As a result, you should get the kaa\_0.0.1\_all.snap file.

```
$ ls snappy-kaa  
bin kaa_0.0.1_all.snap meta
```

Known issue: your readme.md should contain at least two lines of text or package build will silently fail. this is some kind of undocumented behavior.

## Installing snappy Ubuntu

To locally deploy snappy Ubuntu, run the following commands.

## Deploy Snappy ubuntu

```
$ vagrant init ubuntu/ubuntu-core-devel-amd64
$ vagrant up
$ vagrant ssh
```

As a result, you should be connected to your snappy Ubuntu box.

```
Welcome to Ubuntu Vivid Vervet (development branch) (GNU/Linux 3.18.0-9-generic
x86_64)

* Documentation:  https://help.ubuntu.com/
Welcome to the Ubuntu Core rolling development release.

* See https://ubuntu.com/snappy

It's a brave new world here in snappy Ubuntu Core! This machine
does not use apt-get or deb packages. Please see 'snappy --help'
for app installation and transactional updates.

Last login: Tue Feb 24 15:32:09 2015 from 10.0.2.2
ubuntu@localhost:~$
```

## Installing sample Kaa client on snappy Ubuntu

To install the previously built sample Kaa client on snappy Ubuntu, proceed as follows:

1. From the snappy Ubuntu shell, copy kaa\_0.0.1\_all.snap from your host machine.

### Get and install snap package

```
$ scp username@remote_machine_ip:path/to/your/remote/snapp/file . # Copy
kaa_0.0.1_all.snap from your host machine or Kaa sandbox.
$ snappy install ./path/to/your/local/snapp/file --allow-unauthenticated #
Install kaa_0.0.1_all.snap
```

2. Start the Kaa client on the snappy Ubuntu box.

### Start Kaa client

```
$ kaa.client
```

As a result, you should get the logs similar to the following:



```

Initializing Kaa driver...Initializing Kaa SDK...
2015/02/25 14:14:40 [INFO] [kaa.c:200] (0) - Kaa SDK version 0.7.0-SNAPSHOT,
commit hash
2015/02/25 14:14:40 [DEBUG] [kaa_logging.c:110] (0) - Initialized log
collector with log storage {0x7f113400c70}, log upload strategy
{0x7f113400fd0}
2015/02/25 14:14:40 [TRACE] [kaa_demo.c:170] (0) - Creating endpoint profile
2015/02/25 14:14:40 [TRACE] [kaa_demo.c:178] (0) - Setting profile
2015/02/25 14:14:40 [INFO] [kaa_profile.c:406] (0) - Endpoint profile is
updated
2015/02/25 14:14:40 [WARNING] [kaa_channel_manager.c:357] (-6) - Failed to
find transport channel for service 1
2015/02/25 14:14:40 [TRACE] [kaa_demo.c:181] (0) - Adding transport channels
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:166] (0) - Kaa TCP channel
creating....
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:247] (0) - Kaa TCP channel
keepalive is 300
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:256] (0) - Kaa TCP channel
(protocol: id=0x56C8FF92, version=1)
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:294] (0) - Kaa TCP channel
created
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:166] (0) - Kaa TCP channel
creating....
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:247] (0) - Kaa TCP channel
keepalive is 300
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:256] (0) - Kaa TCP channel
(protocol: id=0x56C8FF92, version=1)
2015/02/25 14:14:40 [TRACE] [kaa_tcp_channel.c:294] (0) - Kaa TCP channel
created
2015/02/25 14:14:40 [INFO] [kaa_channel_manager.c:229] (0) - Bootstrap
transport channel [0x1f2082ee] added (protocol: id=0x56C8FF92, version=1)
...

```

## Further reading

To learn more about snappy Ubuntu Core, visit the official [Ubuntu web site](#).

To learn more about Kaa, look at the following guides and references.

Guide	What it is for
<a href="#">Getting started</a>	Use it to quickly create your first Kaa application.
<a href="#">Design reference</a>	Use this reference to learn about features and capabilities of Kaa ( <a href="#">Endpoint profiling</a> , <a href="#">Events</a> , <a href="#">Notifications</a> , <a href="#">Logging</a> , and other features).
<a href="#">Programming guide</a>	Use this guide to create your own Kaa applications.