Design reference

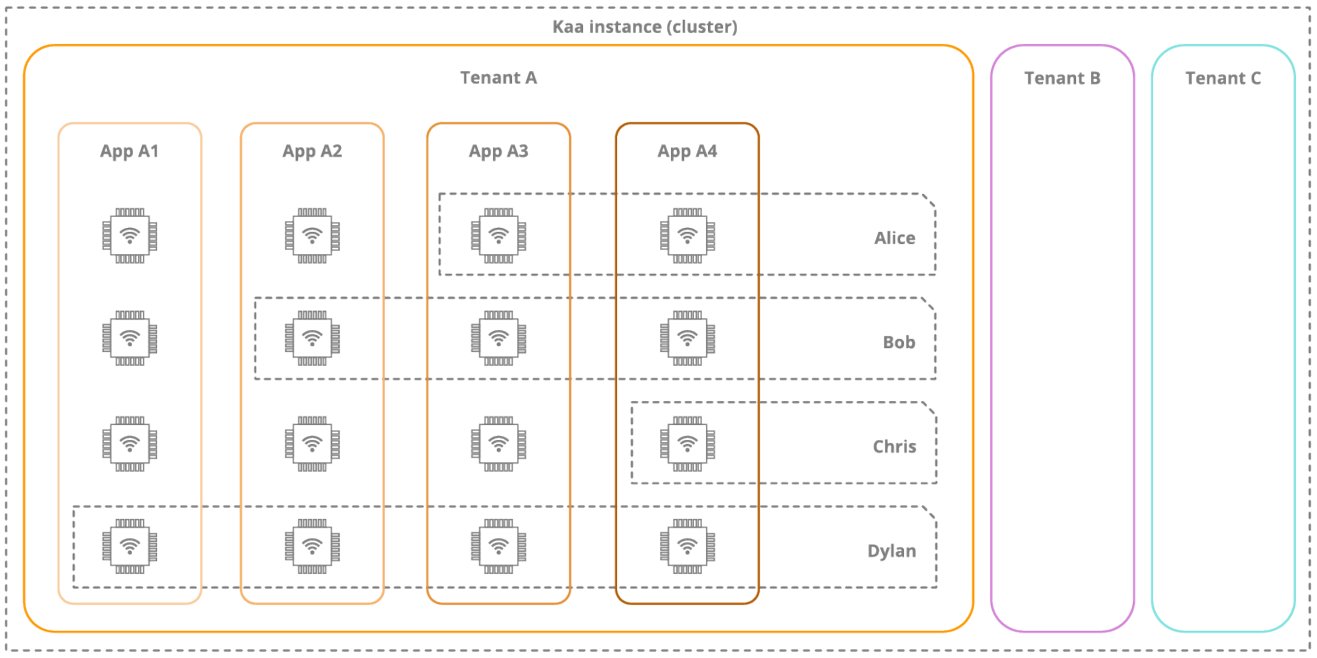
The *Kaa IoT platform* consists of the *Kaa server* and *endpoint SDKs*. The Kaa server implements the back-end part of the platform, exposes integration interfaces, and offers administrative capabilities. An endpoint SDK is a library which provides communication, data marshalling, persistence, and other functions available in Kaa for specific type of an endpoint (e.g. Java-based, C++-based, C-based, Objective-C-based). This SDK can be used to create *Kaa clients*, which are any pieces of software that utilize Kaa functionality and are installed on some connected devices. It is the responsibility of the Kaa client to process structured data provided by the Kaa server (configuration, notifications, etc.) and to supply data to the return path interfaces (profiles, logs, etc.).

A *Kaa instance* (interchangeable with *Kaa deployment*) is a particular implementation of the Kaa platform and it consists of a *Kaa cluster* and *endpoints* . A Kaa cluster represents a number of interconnected Kaa server *nodes*. An *endpoint* is an abstraction which represents a separate managed entity within a Kaa deployment. Practically speaking, an endpoint is a specific Kaa client registered (or waiting to be registered) within a Kaa deployment. For example, a news application installed on your mobile phone, the same news application installed on your tablet, and the same news application on your WiFi-enabled fridge would be considered three different endpoints in Kaa.

An *application* in Kaa represents a family of available implementations of a specific software application used by endpoints. For example, two versions of a sound frequency measuring application which differ by their implementation for, respectively, Arduino and STM32 platforms would be considered the same application in Kaa.

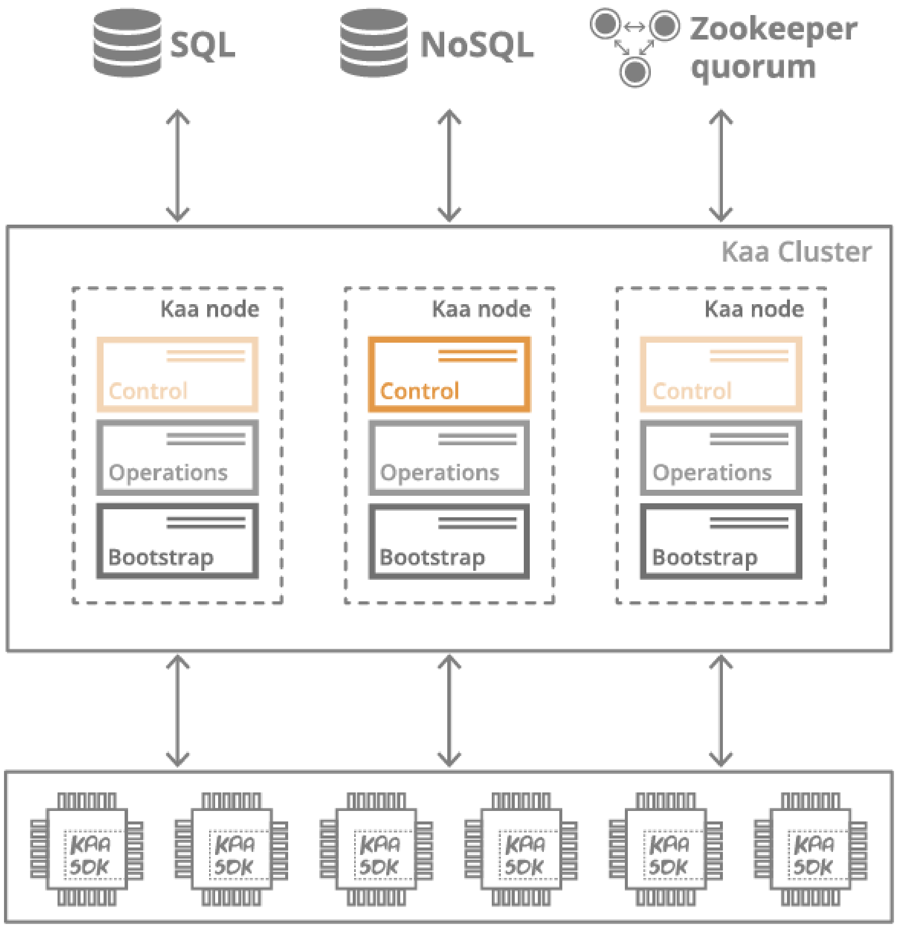
A *tenant* in Kaa is a separate business entity which contains its own endpoints, applications and users.

A single Kaa deployment is able to support multiple tenants, with multiple applications per each tenant. As illustrated in the following diagram, applications belong to tenants, while endpoints register within applications. In addition to tenants, applications and endpoints, there are also users and endpoint groups, which will be described in detail further in this reference.



High-level architecture

Kaa cluster consists of Kaa nodes that use Apache ZooKeeper for services coordination. Kaa cluster also requires SQL and NoSQL database instances.



Kaa cluster node

The Kaa cluster node is comprised of the Control, Operations, and Bootstrap services. Kaa administrator can enable or disable each of this services individually.

Control service

A Kaa Control service is responsible for managing overall system data, processing API calls from the web UI and external integrated systems, and delivering notifications to Operations servers. A Control service manages data stored in a database (independently for each tenant) and notifies every Operations server on most data updates via a Thrift-based protocol. A Control server maintains an up-to-date list of available Operations servers by continuously obtaining this information from ZooKeeper. In addition, Control service provides web UI, which is a standalone component that integrates with the Control server and allows users to create applications, register and configure endpoints, create endpoint groups, etc.

To support high availability, a Kaa cluster must include at least two nodes with control service enabled, with one of them being active and the other(s) being in a standby mode. In case of the active Control service failure, ZooKeeper notifies one of the standby Control service and promotes it to the active Control service.

Operations service

A Kaa Operations service is a “worker” service that is responsible for concurrently handling multiple requests from multiple clients. Most common Operations service tasks include endpoint registration, processing endpoint profile updates, configuration updates distribution, and notifications delivery.

Multiple nodes with Operations service enabled may be set up in a Kaa cluster for the purpose of horizontal scaling. In this case, all the Operations service will function concurrently. In case an Operations service outage happens, the corresponding endpoints switch to the other available Operations services automatically. A Kaa cluster provides instruments for the workload re-balancing at run time, thus effectively routing endpoints to the less loaded nodes in the cluster.

Bootstrap service

A Kaa Bootstrap service is responsible for directing endpoints to Operations services. On their part, Kaa endpoints have a built-in list of Bootstrap services set up in the given Kaa deployment. The endpoints use this list to query the Bootstrap services and retrieve a list of currently available Operations services from them, as well as security credentials. Bootstrap services maintain their lists of available Operations service nodes by coordinating with the ZooKeeper servers.

Endpoint

A Kaa endpoint is a particular application which uses the Kaa client SDK and resides on a particular connected device. The Kaa endpoint SDK provides functionality for communicating with the Kaa server, managing data locally in the client application, as well as provides integration APIs. The client SDK abstracts the communication protocol, data persistence, and other implementation details that may be specific for any concrete solution based on Kaa.

Third-party

Zookeeper

Apache ZooKeeper enables highly reliable distributed coordination of Kaa cluster nodes. Each Kaa node push information about connection parameters, enabled services and corresponding services load. Other Kaa nodes use this information in order to get list of their neighbors and communicate with them. Control service uses information about existing Bootstrap services and their connection parameters during SDK generation.

SQL database

SQL database instance is used to store metadata about tenants, applications, endpoint groups, etc. This information is shared between endpoints, thus it's volume does not scale and it can be efficiently stored in modern SQL databases. To support high availability of Kaa cluster, SQL database should be also deployed in cluster mode.

NoSQL database

NoSQL database instance is used to store information about endpoint profiles, notifications, configurations, etc. The volume of this information scales linearly with amount of endpoints that are managed using particular Kaa cluster instance. NoSQL database nodes can be co-located with Kaa nodes on the same physical or virtual machines. Kaa support Apache Cassandra and MongoDB as a NoSQL database at the moment.

Further reading

After you familiarize yourself with all the sections in Design reference, use the following guides and references to advance further into Kaa. We also assume that you're already through with Kaa installation as described in either Sandbox or Installation guide. Make sure you did not skip the Getting started page as it allows you to quickly create your first Kaa application.

Guide	What it is for
Administration UI guide	Use this guide to start working with the Kaa web UI.
Programming guide	Use this guide to create your own Kaa applications.
Contribute to Kaa	Use this guide to learn how to contribute to Kaa project and which code/documentation style conventions we use.