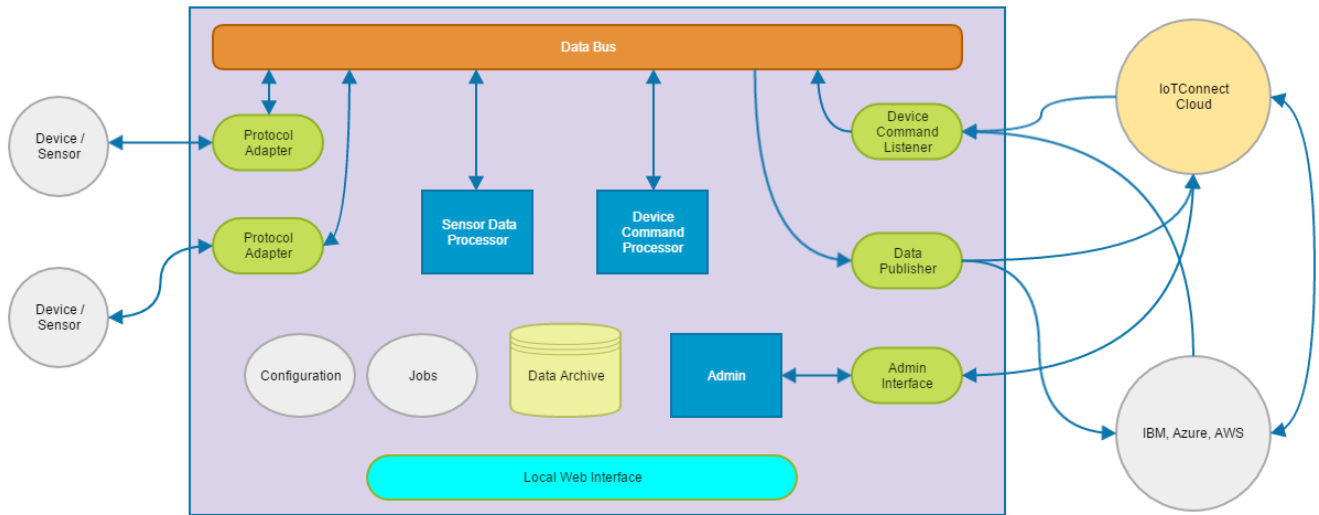# Gateway General Architecture

Android gateway design notes

- The gateway should be designed to run as an Android service.
- Protocol adapters should be designed as "plugins" for extensibility. An SDK should be provided for developers to build new plugins.
- There should be an SDK to help with building applications leveraging the gateway.
- There will not be a local web interface; however, we could develop an application (similar to MyIoT gateway) to configure and monitor the gateway.
- Leverage a lightweight open source database for data archive such as Realm, Couchbase Lite or UnQLite.
- Leverage a lightweight open source persistent queue library such as Tape or Priority Job Queue.

iOS gateway design notes

- The gateway should be designed to run in background mode
- Protocol adapters should be designed as "plugins" for extensibility. An SDK should be provided for developers to build new plugins
- There should be an SDK to help with building applications leveraging the gateway.
- There will not be a local web interface; however, we could develop an application (similar to MyIoT gateway) to configure and monitor the gateway.
- Leverage a lightweight open source database for data archive such as Realm, Couchbase Lite or UnQLite.
- Leverage a lightweight open source persistent queue library such as EDQueue or IPOfflineQueue

.Linux gateway design notes

- The gateway should be lightweight and designed to run as a system service at startup. Spring framework might not be a good choice here
- Protocol adapters should be designed as "plugins" for extensibility. An SDK should be provided for developers to build new plugins.
- There should be a local web interface for configuring and monitoring the gateway.
- Leverage a lightweight open source NoSQL database for data archive such as Couchbase Lite, UnQLite, MongoDB or an SQL database such as H2, HSQLDB, SQLite, MySQL
- Leverage a lightweight open source persistent queue component such as Redis