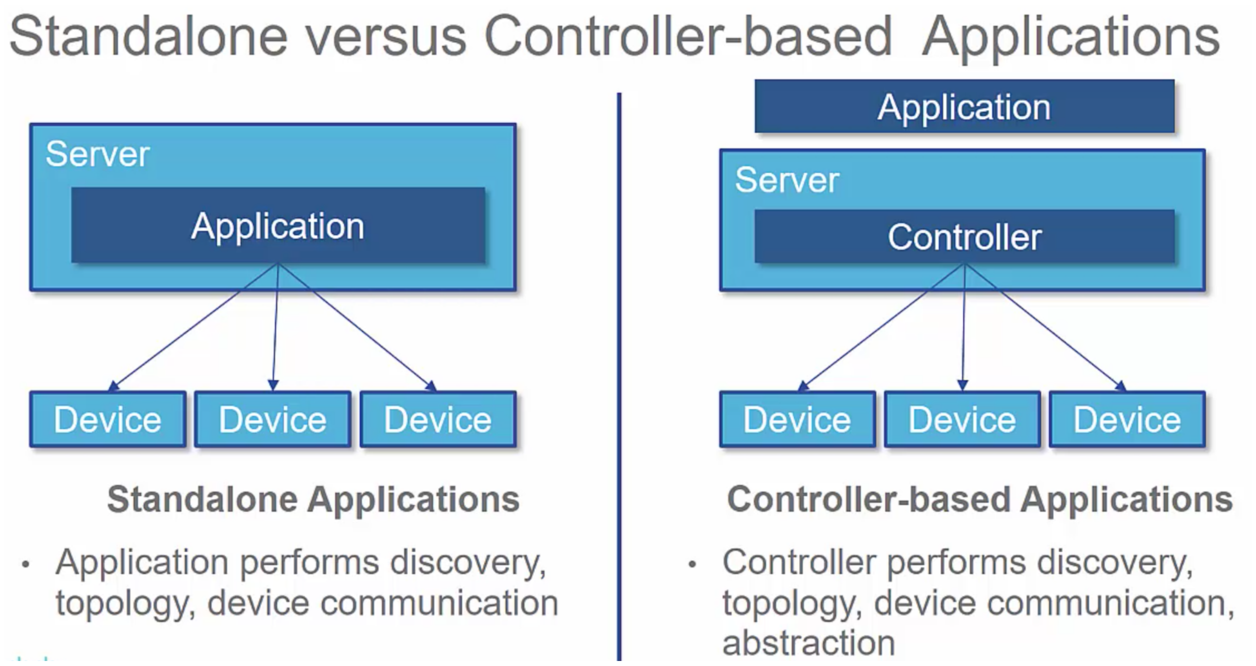
Standalone vs. Controller-based Applications



Standalone versus Controller-based Applications

Standalone Applications

- Application performs discovery, topology, device communication

Controller-based Applications

- Controller performs discovery, topology, device communication, abstraction

Standalone applications have the following characteristics and considerations:

Communication to each device is direct, from your application directly to the device. Compare with controller-based, where communication goes through a translation process as it goes through the controller, before being sent to the device.

Communication is individually to each device. In other words, communication is not done at a higher group or policy level. If you wish to communicate with a group of devices, that functionality will need to be implemented by your own application.

Discovery and topology are nominal features of every controller-based solution. Without a controller, your application will need to perform its own device and topology discovery.

Controller-based applications have the following characteristics and considerations:

Communication to each device is through the controller, which provides some level of translation. For example, in ODL translation is done from RESTCONF into a NETCONF request. Translation can also be done from JSON to XML. Controllers may also provide some level of abstraction. For example, APIC-EM provides abstraction from policies, users, and resources, to the actual CLI commands appropriate for each device. Thus your application is completely shielded from CLI details, by being able to use the policy-based abstraction provided by the APIC-EM controller.

Communication can be with groups of devices, rather than device-by-device. APIC-EM allows the programmatic application to apply policy across a set of devices in this manner. On the other hand, some controllers still require per-device communication; consider ODL's NETCONF API, which requires the application to communicate individually to each device.

Controllers often will discover devices and topology, so that your application does not need to provide this functionality. Be aware however that there may be limitations to this functionality provided by the controller; for example, with ODL, discovery of OpenFlow devices and their topology is automatic, but discovery of NETCONF devices requires individual mounting of each device by your application. In addition, with NETCONF no topology information is retrieved. The bottom line is that your application may or may not be able to benefit from device and topology discovery, depending on the environment in which your application is running.

External versus Internal (Controller-Based)
There are two distinct types of application with controller-based applications: external and internal.

Definitions:

External applications run outside the domain of the SDN controller, and communicate via RESTful APIs provided by the controller itself, or by other applications running inside the controller. External applications can thus be written in any language, including of course Python.

Internal applications run inside the domain of the SDN controller within the ODL controller's Java-based container, which means that internal applications for ODL must be written in Java, and must adhere to the MD-SAL structure for internal ODL applications.

External applications have the following characteristics and considerations.

Applications can be written in any language – Python, Java, Ruby, Go, or even in no language at all (Bash) - since they run outside the controller.

The API is via REST, which means that your application can actually be running remotely from the controller – on a different machine or even in the cloud.

Internal applications have the following characteristics and considerations.

They must be written in the native language of the controller, which in ODL's case is Java.

For ODL, they must adhere to the internal structure of that controller, which is MD-SAL.

The application can use internal Java APIs, but the REST APIs are also available if so desired.