Plugin Registration Service

This folder contains a utility, pluginwatcher, for Kubelet to register different types of node-level plugins such as device plugins or CSI plugins. It discovers plugins by monitoring inotify events under the directory returned by kubelet.getPluginsDir(). We will refer to this directory as PluginsDir.

Plugins are expected to implement the gRPC registration service specified in pkg/kubelet/apis/pluginregistration/v*/api.proto.

Plugin Discovery

The pluginwatcher service will discover plugins in the PluginDir when they place a socket in that directory or, at Kubelet start if the socket is already there.

This socket filename should not start with a '.' as it will be ignored.

gRPC Service Lifecycle

For any discovered plugin, kubelet will issue a Registration.GetInfo gRPC call to get plugin type, name, endpoint and supported service API versions.

If any of the following steps in registration fails, on retry registration will start from scratch:

- Registration.GetInfo is called against socket.
- Validate is called against internal plugin type handler.
- Register is called against internal plugin type handler.
- NotifyRegistrationStatus is called against socket to indicate registration result.

During plugin initialization phase, Kubelet will issue Plugin specific calls (e.g: DevicePlugin::GetDevicePluginOptions).

Once Kubelet determines that it is ready to use your plugin it will issue a Registration.NotifyRegistrationStatus gRPC call.

If the plugin removes its socket from the PluginDir this will be interpreted as a plugin Deregistration. If any of the following steps in deregistration fails, on retry deregistration will start from scratch:

- Registration.GetInfo is called against socket.
- DeRegisterPlugin is called against internal plugin type handler.

gRPC Service Overview

Here are the general rules that Kubelet plugin developers should follow:

- Run plugin as 'root' user. Currently creating socket under PluginsDir, a root owned directory, requires plugin process to be running as 'root'.
- The plugin name sent during Registration.GetInfo grpc should be unique for the given plugin type (CSIPlugin or DevicePlugin).
- The socket path needs to be unique within one directory, in normal case, each plugin type has its own sub directory, but the design does support socket file under any sub directory of PluginSockDir.
- A plugin should clean up its own socket upon exiting or when a new instance comes up. A plugin should NOT remove any sockets belonging to other plugins.

- A plugin should make sure it has service ready for any supported service API version listed in the PluginInfo.
- For an example plugin implementation, take a look at example_plugin.go included in this directory.

Kubelet Interface

For any kubelet components using the pluginwatcher module, you will need to implement the PluginHandler interface defined in the types.go file.

The interface is documented and the implementations are registered with the pluginwatcher module in kubelet.go by calling AddHandler(pluginType, handler).

The lifecycle follows a simple state machine:

```
           Validate -> Register -> DeRegister
              ^          +
              |          |
              +--------- +
```

The pluginwatcher calls the functions with the received plugin name, supported service API versions and the endpoint to call the plugin on.

The Kubelet component that receives this callback can acknowledge or reject the plugin according to its own logic, and use the socket path to establish its service communication with any API version supported by the plugin.