SIG CLI Deepdive

Overview of session

- Immediate and long term goals of SIG
- Initiatives in service of these goals
- Techniques and strategies for implementing

Goals

- Support client / server version skew
- Support API extensions
- Faster releases
- Faster development
- Consistency and composability of subcommands

InitiATives - Generic & Dynamic Commands

- Why
  - Work with version skew & API extensions
  - Can be released independent of the Kubernetes release
  - Can be developed outside kubernetes/kubernetes
- What are they
  - Do not reference specific API instances - instead use API metadata
- Examples
  - Kubectl create -f <file>
  - Kubectl scale
- Implemented using
  - Schema and structure (OpenAPI)
  - Subresources (discovery service)
  - RPC services in the cluster?

Initiatives - Plugins

- Why
  - Enable subcommands for API extensions (e.g. kubectl create foo)
  - Enable functionality to be built outside of core for scaling development
  - Enable functionality outside the scope of core to be made available in kubectl
- What are they
  - Executables that kubectl can exec if installed in specific locations
- Implemented using
  - Kubectl plugin discovery
- Future work
  - Discovery and distribution

CLI Libraries

- Facilitate building generic / dynamic commands in kubectl
- Facilitate building plugins
- Facilitate building other clis

Questions & comments