In kruise-tools, some sub commands are developed by import in cmd.go, such as:

```
scale          Set a new size for a Deployment, ReplicaSet or Replication Controller
autoscale      Auto-scale a Deployment, ReplicaSet, or ReplicationController

Cluster Management Commands: (Described in cmd.go)
  certificate    Modify certificate resources.
  cluster-info   Display cluster info
  top            Display Resource (CPU/Memory/Storage) usage.
  cordon         Mark node as unschedulable
  uncordon       Mark node as schedulable
  drain          Drain node in preparation for maintenance
  taint          Update the taints on one or more nodes

Troubleshooting and Debugging Commands:  (Described in cmd.go)
  describe       Show details of a specific resource or group of resources
  logs           Print the logs for a container in a pod
  attach         Attach to a running container
  exec           Execute a command in a container
  port-forward   Forward one or more local ports to a pod
  debug          Attach a debug container to a running pod

Advanced Commands: (Described in cmd.go)
  diff           Diff live version against would-be applied version
  apply          Apply a configuration to a resource by filename or stdin
  patch          Update field(s) of a resource using strategic merge patch
  replace        Replace a resource by filename or stdin
  wait           Experimental: Wait for a specific condition on one or many resources.
  kustomize      Build a kustomization target from a directory or a remote url.

Other Commands: (Described in cmd.go)
  api-resources Print the supported API resources on the server
  api-versions   Print the supported API versions on the server, in the form of "group/version"
  config         Modify kubeconfig files
  plugin         Provides utilities for interacting with plugins.
  version        Print the client and server version information

The other commands are developed in code, for example:
CloneSet Commands: (Developed as code)
kubectl-kruise rollout       Manage the rollout of a resource

# Check the history of deployments including the revision
kubectl-kruise rollout history deployment/frontend

# Rollback to the previous deployment
kubectl-kruise rollout undo deployment/frontend

# Rollback to a specific revision
```

kubectl-kruise rollout undo deployment/frontend --to-revision=2

Watch rolling update status of "frontend" deployment until completion
kubectl-kruise rollout status -w deployment/frontend

Rolling restart of the "frontend" deployment
kubectl-kruise rollout restart deployment/frontend

kubectl-kruise migrate # Migrate from K8s original workloads to Kruise workloads

For each rollout command, it calls NewCmdRollout() to return a pointer of cobra.Command, which includes the commands from the sub commands.

Firstly, for each rollout sub-command function, it returns an initialized options instance and then creates a variable validArgs to declare their supporting resource type. After these, each sub-command function initializes a new cobra.Command with the previous initialized options and validArgs, and adds a file name option with the file name usage. After this, in some conditions some flags are added to the command, and then each rollout sub-command function returns the command to function NewCmdRollout(),

When initializing each sub-command in the rollout sub-command function, there are some options of the function: Use, DisableFlagsInUseLine, Short, Long, Example, Run, ValidArgs. Use tells the format of the sub-command; DisableFlagsInUseLine is always true; Short and long arguments describe the usage of the command shortly; Example gives some examples for running these sub-commands.

For the run argument, it calls the option's three functions: Complete(), Validate() and Run(). The function Complete() completes all the required options; The function Validate() makes sure all the provided values for command-line options are valid; The Run() function performs the execution of 'rollout [sub-command]' sub-command. When executing the Run() function, the builder resource is dynamic at the beginning, and the decision of which type of resource to get is based on o.Resources. The rollback execution is in rollbacker.Rollback(...). The rollbacker is an interface type and its real implementation types include DaemonSetRollbacker and CloneSetRollbacker. This rollbacker object is generated by internalpolymorphichelpers.RollbackerFn according to the actual resource type.
To avoid the potential mistake, this argument calls cmdutil.CheckErr() function with the parameters from the returned value from Complete(), Validate() and Run(), to check if there is any returned error from these three functions.

For migrate command, its running process is similar to rollout command: Create a cobra.Command instance and return its pointer, which is initialized with a migrate option. The difference is that different commands have different flags, options and samples. Unlike the rollout command, the migrate command does not have the Validate() function because it does not have a sub-command. Moreover, kruise-tools considers the clonsets as a special case and at this time the Run() function will call migrateCloneSet() function to perform the migration execution.