Installing client-go

Using the latest version

If you want to use the latest version of this library, use go1.16+ and run:

```
go get k8s.io/client-go@latest
```

This will record a dependency on `k8s.io/client-go` in your go module. You can now import and use the `k8s.io/client-go` APIs in your project. The next time you `go build`, `go test`, or `go run` your project, `k8s.io/client-go` and its dependencies will be downloaded (if needed), and detailed dependency version info will be added to your `go.mod` file (or you can also run `go mod tidy` to do this directly).

Using a specific version

If you want to use a particular version of the `k8s.io/client-go` library, you can indicate which version of `client-go` your project requires:

- If you are using Kubernetes versions >= `v1.17.0`, use a corresponding `v0.x.y` tag. For example, `k8s.io/client-go@v0.20.4` corresponds to Kubernetes `v1.20.4`:

```
go get k8s.io/client-go@v0.20.4
```

- If you are using Kubernetes versions < `v1.17.0`, use a corresponding `kubernetes-1.x.y` tag. For example, `k8s.io/client-go@kubernetes-1.16.3` corresponds to Kubernetes `v1.16.3`:

```
go get k8s.io/client-go@kubernetes-1.16.3
```

You can now import and use the `k8s.io/client-go` APIs in your project. The next time you `go build`, `go test`, or `go run` your project, `k8s.io/client-go` and its dependencies will be downloaded (if needed), and detailed dependency version info will be added to your `go.mod` file (or you can also run `go mod tidy` to do this directly).

Troubleshooting

Go versions prior to 1.16

If you get a message like `module k8s.io/client-go@latest found (v1.5.2), but does not contain package k8s.io/client-go/...`, you are likely using a go version prior to 1.16 and must explicitly specify the k8s.io/client-go version you want. For example:

```
go get k8s.io/client-go@v0.20.4
```

Conflicting requirements for older client-go versions

If you get a message like `module k8s.io/api@latest found, but does not contain package k8s.io/api/auditregistration/v1alpha1`, something in your build is likely requiring an old version of

`k8s.io/client-go` like `v11.0.0+incompatible`.

First, try to fetch a more recent version. For example:

```
go get k8s.io/client-go@v0.20.4
```

If that doesn't resolve the problem, see what is requiring an `...+incompatible` version of client-go, and update to use a newer version of that library, if possible:

```
go mod graph | grep " k8s.io/client-go@"
```

As a last resort, you can force the build to use a specific version of client-go, even if some of your dependencies still want `...+incompatible` versions. For example:

```
go mod edit -replace=k8s.io/client-go=k8s.io/client-go@v0.20.4
go get k8s.io/client-go@v0.20.4
```

Go modules disabled

If you get a message like `cannot use path@version syntax in GOPATH mode`, you likely do not have go modules enabled.

Dependency management tools are built into go 1.11+ in the form of go modules. These are used by the main Kubernetes repo (>= `v1.15.0`) and `client-go` (>= `kubernetes-1.15.0`) to manage dependencies. If you are using go 1.11 or 1.12 and are working with a project located within `$GOPATH`, you must opt into using go modules:

```
export GO111MODULE=on
```

Ensure your project has a `go.mod` file defined at the root of your project. If you do not already have one, `go mod init` will create one for you:

```
go mod init
```