Dep : dependency management in Golang

* Dependency management is also known as vendorig in go

Steps:

1 . dep init

* dep will analyize your project, identify your dependencies, pick the highest compatible version for each dependency, generate a [Gopkg.toml](https://github.com/golang/dep/blob/master/docs/Gopkg.toml.md) manifest and Gopkg.lock files (see [here](https://github.com/golang/dep/issues/119#issuecomment-275467269) for background on the two file format), and install dependencies in vendor/.

* Gopkg.lock: it locks to version of the packages EXCEPT the version should be maintained in the Gopkg.toml

This file is autogenerated

It depends on the import statements in the source version controlled by Gopkg.toml.

Project Code 
(imports) 
Gopkg.lock 
Gopkg.toml 
Deps 
(vendor) 

* To update a package, we can update its version in the Gopkg.toml file and then then 'dep ensure' to update the package to the desired version.
* dep ensure -add : To add a new package.

-----------------------------------

Glide :

1. Glide create (init):

* Scans the code.
* Creates glide.yaml file with all the dependencies.
* The imports in the dependent packages are also scanned to determine dependencies of dependencies.
* If a dependent package contains glide.yaml, then that glide.yaml file is used to help determine the dependency rules.
* Then these dependencies are exported in the 'vendor' folder where go tools can find
* Glide.yaml:

1. Names the current package.
2. It declares external dependencies.

A brief glide.yaml file looks like this: 
package: github.com/Masterminds/g1ide 
import : 
package: github.com/Masterminds/semver 
package: github . com/Masterminds/cookoo 
version: 
Al.2.e 
repo: git@github . com:Masterminds/cookoo.git 

* Above tells us that:

1. Our package name is github.com/Masterminds/glide
2. This package depends on two libraries mentioned below.

1. Glide get <package name>:

* Downloads the package into the vendor directory.
* Adds the package info to the glide.yaml file.

1. Glide update or glide up

* Regenerates the dependency version using scanning and rules
* Downloads or updates all of the libraries from the glide.yaml file and put them in the vendor directory

1. Glide.list

* To list down the downloaded packages.

1. Glide install:

* Does not scan the project.
* Reads glide.lock file and install the commit id specific version there.

Glide.lock file:

* Its created when glide update command is run.
* It contains complete dependency tree (transitive ones)
* It also contains the revision (commit ID) in use.
* This file should not be edited.