# Development Environment

This section covers specific instructions to create a developer-friendly environment to contribute to the DAOS development. This includes how to regenerate the protobuf files or add new Go package dependencies, which is only required for development purposes.

## Building DAOS for Development

Prerequisite when built using --build-deps are installed in component specific directories under PREFIX/prereq/$TARGET\_TYPE. Initialize and update the git submodules:

$ git submodule init  
$ git submodule update

Run the following scons command:

$ scons PREFIX=${daos\_prefix\_path}  
 install  
 --build-deps=yes  
 --config=force

Installing the components into separate directories allow upgrading the components individually by replacing --build-deps=yes with --update-prereq={component\\_name}. This requires a change to the environment configuration from before. For automated environment setup, source utils/sl/utils/setup\_local.sh.

The install path should be relocatable with the exception that daos\_admin will not be able to find the new location of daos and dependencies. All other libraries and binaries should work without any change due to relative paths. Editing the .build-vars.sh file to replace the old with the new can restore the capability of setup\_local.sh to automate path setup.

To run daos\_server, either the rpath in daos\_admin needs to be patched to the new installation location of spdk and isal or LD\_LIBRARY\_PATH needs to be set. This can be done using SL\_SPDK\_PREFIX and SL\_ISAL\_PREFIX set when sourcing setup\_local.sh. This can also be done with the following commands:

source utils/sl/setup\_local.sh  
sudo -E utils/setup\_daos\_admin.sh [path to new location of daos]

This script is intended only for developer setup of daos\_admin.

With this approach, DAOS gets built using the prebuilt dependencies in ${daos\_prefix\_path}/prereq, and required options are saved for future compilations. So, after the first time, during development, only “scons --config=force” and “scons --config=force install” would suffice for compiling changes to DAOS source code.

If you wish to compile DAOS with clang rather than gcc, set COMPILER=clang on the scons command line. This option is also saved for future compilations.

Additionally, users can specify BUILD\_TYPE=[dev|release|debug] and scons will save the intermediate build for the various BUILD\_TYPE, COMPILER, and TARGET\_TYPE options so a user can switch between options without a full rebuild and thus with minimal cost. By default, TARGET\_TYPE is set to 'default' which means it uses the BUILD\_TYPE setting. To avoid rebuilding prerequisites for every BUILD\_TYPE setting, TARGET\_TYPE can be explicitly set to a BUILD\_TYPE setting to always use that set of prerequisites. These settings are stored in daos.conf so setting the values on subsequent builds is not necessary.

If needed, ALT\_PREFIX can be set to a colon separated prefix path where to look for already built components. If set, the build will check these paths for components before proceeding to build.

## Go dependencies

Developers contributing Go code may need to change the external dependencies located in the src/control/vendor directory. The DAOS codebase uses [Go Modules](https://github.com/golang/go/wiki/Modules) to manage these dependencies. As this feature is built in to Go distributions starting with version 1.11, no additional tools are needed to manage dependencies.

Among other benefits, one of the major advantages of using Go Modules is that it removes the requirement for builds to be done within the $GOPATH, which simplifies our build system and other internal tooling.

While it is possible to use Go Modules without checking a vendor directory into SCM, the DAOS project continues to use vendored dependencies in order to insulate our build system from transient network issues and other problems associated with nonvendored builds.

The following is a short list of example workflows. For more details, please refer to [one](https://github.com/golang/go/wiki/Modules#quick-start) of [the](https://engineering.kablamo.com.au/posts/2018/just-tell-me-how-to-use-go-modules/) [many](https://blog.golang.org/migrating-to-go-modules) resources available online.

# add a new dependency  
$ cd ~/daos/src/control # or wherever your daos clone lives  
$ go get github.com/awesome/thing  
# make sure that github.com/awesome/thing is imported somewhere in the codebase  
$ ./run\_go\_tests.sh  
# note that go.mod and go.sum have been updated automatically  
#  
# when ready to commit and push for review:  
$ go mod vendor  
$ git commit -a # should pick up go.mod, go.sum, vendor/\*, etc.

# update an existing dependency  
$ cd ~/daos/src/control # or wherever your daos clone lives  
$ go get -u github.com/awesome/thing  
# make sure that github.com/awesome/thing is imported somewhere in the codebase  
$ ./run\_go\_tests.sh  
# note that go.mod and go.sum have been updated automatically  
#  
# when ready to commit and push for review:  
$ go mod vendor  
$ git commit -a # should pick up go.mod, go.sum, vendor/\*, etc.

# replace/remove an existing dependency  
$ cd ~/daos/src/control # or wherever your daos clone lives  
$ go get github.com/other/thing  
# make sure that github.com/other/thing is imported somewhere in the codebase,  
# and that github.com/awesome/thing is no longer imported  
$ ./run\_go\_tests.sh  
# note that go.mod and go.sum have been updated automatically  
#  
# when ready to commit and push for review:  
$ go mod tidy  
$ go mod vendor  
$ git commit -a # should pick up go.mod, go.sum, vendor/\*, etc.

In all cases, after updating the vendor directory, it is a good idea to verify that your changes were applied as expected. In order to do this, a simple workflow is to clear the caches to force a clean build and then run the test script, which is vendor-aware and will not try to download missing modules:

$ cd ~/daos/src/control # or wherever your daos clone lives  
$ go clean -modcache -cache  
$ ./run\_go\_tests.sh  
$ ls ~/go/pkg/mod # ~/go/pkg/mod should either not exist or be empty

## Protobuf Compiler

The DAOS control plane infrastructure uses [Protocol Buffers](https://github.com/protocolbuffers/protobuf) as the data serialization format for its RPC requests. Not all developers will need to compile the \\*.proto files, but if Protobuf changes are needed, the developer must regenerate the corresponding C and Go source files using a Protobuf compiler compatible with proto3 syntax.

### Recommended Versions

The recommended installation method is to clone the git repositories, check out the tagged releases noted below, and install from source. Later versions may work, but are not guaranteed. You may encounter installation errors when building from source relating to insufficient permissions. If that occurs, you may try relocating the repo to /var/tmp/ in order to build and install from there.

* [Protocol Buffers](https://github.com/protocolbuffers/protobuf) v3.11.4. [Installation instructions](https://github.com/protocolbuffers/protobuf/blob/master/src/README.md).
* [Protobuf-C](https://github.com/protobuf-c/protobuf-c) v1.3.3. [Installation instructions](https://github.com/protobuf-c/protobuf-c/blob/master/README.md).
* gRPC plugin: [protoc-gen-go](https://github.com/golang/protobuf) is the version specified in [go.mod](https://github.com/daos-stack/daos/blob/master/src/control/go.mod). This plugin is automatically installed by the Makefile in $DAOSREPO/src/proto.

### Compiling Protobuf Files

The source (.proto) files live under $DAOSREPO/src/proto. The preferred mechanism for generating compiled C/Go protobuf definitions is to use the Makefile in this directory. Care should be taken to keep the Makefile updated when source files are added or removed, or generated file destinations are updated.

Note that the generated files are checked into SCM and are not generated as part of the normal DAOS build process. This allows developers to ensure that the generated files are correct after any changes to the source files are made.

$ cd ~/daos/src/proto # or wherever your daos clone lives  
$ make  
protoc -I /home/foo/daos/src/proto/mgmt/ --go\_out=plugins=grpc:/home/foo/daos/src/control/common/proto/mgmt/ acl.proto  
protoc -I /home/foo/daos/src/proto/mgmt/ --go\_out=plugins=grpc:/home/foo/daos/src/control/common/proto/mgmt/ mgmt.proto  
...  
$ git status  
...  
# modified: ../control/common/proto/mgmt/acl.pb.go  
# modified: ../control/common/proto/mgmt/mgmt.pb.go  
...

After verifying that the generated C/Go files are correct, add and commit them as you would any other file.