

Using gomote

Installing

```
$ GO111MODULE=on go get -u golang.org/x/build/cmd/gomote # up to Go 1.15
```

OR

```
$ go get golang.org/x/build/cmd/gomote@latest # Go 1.16 and latest (including current  
Go 1.16 dev tree)
```

Using

TODO: examples. For now:

The gomote command is a client for the Go builder infrastructure. It's a remote control for remote Go builder machines.

See <https://go.dev/wiki/Gomote>

Usage:

```
gomote [global-flags] cmd [cmd-flags]
```

For example,

```
$ gomote create openbsd-amd64-60
```

```
user-username-openbsd-amd64-60-0
```

```
$ gomote push user-username-openbsd-amd64-60-0
```

```
$ gomote run user-username-openbsd-amd64-60-0 go/src/make.bash
```

```
$ gomote run user-username-openbsd-amd64-60-0 go/bin/go test -v -short os
```

To list the subcommands, run "gomote" without arguments:

Commands:

create	create a buildlet; with no args, list types of buildlets
destroy	destroy a buildlet
gettar	extract a tar.gz from a buildlet
list	list active buildlets
ls	list the contents of a directory on a buildlet
ping	test whether a buildlet is alive and reachable
push	sync your GOROOT directory to the buildlet
put	put files on a buildlet
put14	put Go 1.4 in place
puttar	extract a tar.gz to a buildlet
rm	delete files or directories
run	run a command on a buildlet
ssh	ssh to a buildlet

To list all the builder types available, run "create" with no arguments:

```
$ gomote create
(list tons of buildlet types)
```

The "gomote run" command has many of its own flags:

```
$ gomote run -h
run usage: gomote run [run-opts] <instance> <cmd> [args...]
  -builderenv string
    Optional alternate builder to act like. Must share the same
    underlying buildlet host type, or it's an error. For
    instance, linux-amd64-race or linux-386-387 are compatible
    with linux-amd64, but openbsd-amd64 and openbsd-386 are
    different hosts.
  -debug
    write debug info about the command's execution before it begins
  -dir string
    Directory to run from. Defaults to the directory of the
    command, or the work directory if -system is true.
  -e value
    Environment variable KEY=value. The -e flag may be repeated
    multiple times to add multiple things to the environment.
  -path string
    Comma-separated list of ExecOpts.Path elements. The special
    string 'EMPTY' means to run without any $PATH. The empty
    string (default) does not modify the $PATH. Otherwise, the
    following expansions apply: the string '$PATH' expands to
    the current PATH element(s), the substring '$WORKDIR'
    expands to the buildlet's temp workdir.
  -system
    run inside the system, and not inside the workdir; this is implicit if cmd
    starts with '/'
```

Debugging buildlets directly

Using "gomote create" contacts the build coordinator (farmer.golang.org) and requests that it create the buildlet on your behalf. All subsequent commands (such as "gomote run" or "gomote ls") then proxy your request via the coordinator. To access a buildlet directly (for example, when working on the buildlet code), you can skip the "gomote create" step and use the special builder name "<build-config-name>@ip[:port>", such as "windows-amd64-2008@10.1.5.3".

Tricks

Windows

```
$ gomote run -path '$PATH,$WORKDIR/go/bin' -e 'GOROOT=c:\workdir\go' $MOTE
go/src/make.bat
```

```
$ gomote run -path '$PATH,$WORKDIR/go/bin' -e 'GOROOT=c:\workdir\go' $MOTE
go/bin/go.exe test cmd/go -short
```

Subrepos on Windows

```
$ tar --exclude .git -C ~/go/src/ -zc golang.org/x/tools | gomote puttar -
dir=gopath/src $MOTE
$ gomote run -e 'GOPATH=c:/workdir/gopath' $MOTE go/bin/go test -
run=TestFixImportsVendorPackage golang.org/x/tools/imports
```

If ssh'd into the machine, these envvars may be handy:

```
$ set GOPATH=c:\workdir\gopath
$ set PATH=%PATH%;c:\workdir\gopath\bin;c:\workdir\go\bin
$ set CGO_ENABLED=0
```

Subrepos on Unix

Testing golang.org/x/sys/unix on \$MOTE

```
$ tar -C $GOPATH/src/ -zc golang.org/x/sys/unix | gomote puttar -dir=gopath/src $MOTE
$ gomote run -e 'GOPATH=/tmp/workdir/gopath' -dir 'gopath/src/golang.org/x/sys/unix'
$MOTE go/bin/go test -v golang.org/x/sys/unix
```

(The GOPATH part is for GOPATH compatibility mode; the `-dir` is for modules mode, which looks in the working directory and up for `go.mod`)

Android

```
export MOTE=`gomote create android-arm64-wikofever`
gomote push $MOTE
gomote run $MOTE go/src/make.bash
```

PATH must contain the exec wrapper, `go_android*_exec`, built by `make.bash`.

```
gomote run -path '$PATH,$WORKDIR/go/bin' $MOTE go/bin/go test math/big
```

About Buildlets

<https://farmer.golang.org/builders> lists information about how each buildlet is deployed and configured. The information is from `golang.org/x/build/dashboard` and `golang.org/x/build/env`.

Access token

Note that as of May 2021, new gomote accounts are on hold while new infrastructure is built.

To get an access token, file an issue with the `access:` prefix and cc `@golang/release-team`, asking them to provide you with the hash reported by running `genbuilderkey user=$USER`, where `$USER` is your computer's username (as reported by `echo $USER`, or `echo $USERNAME` on Windows). Write the resulting token to the gomote config file, as in this hypothetical example:

```
$ echo d41d8cd98f00b204e9800998ecf8427e > $HOME/.config/gomote/user-$USER.token
```

gomote ssh

The `gomote ssh` command uses SSH keys associated with your GitHub account for authentication. After creating a gomote instance (which requires a Gomote access token described above), to use `gomote ssh` to connect to it, you should ensure that:

1. `gophers.GitHubOfGomoteUser` returns the correct GitHub account. If it needs to be modified, send a CL.
2. You've added an SSH key to your GitHub account. You can test this with `ssh -T git@github.com`. See [GitHub documentation](#) for more information.