Go Installation Instructions

Setting up a Go-Agent (for sponsoring companies or individuals)

Go Agent machines are the computers that do the heavy lifting. They are assigned jobs from the Go Server for compilation, test or other purposes.

You can add another agent to the common cloud by following the instructions here.

✅ RECOMMENDED METHOD – with Docker

Run the following commands:

```
git clone https://github.com/genivigo/gocd-setup
cd gocd-setup/docker/agent
<Edit some variables in Makefile -- see inside>
make build
make run
```

Details (do not use – use the Docker setup instead)

Most information is available in Go.CD standard documentation

---

The required setting in **/etc/default/go-agent**, for the most recent versions are:

```
GO_SERVER_URL=https://go.genivi.org:8154/go
```

⚠️ In older versions, format should still work but is deprecated:

```
GO_SERVER=go.genivi.org
The PORT should be set to the standard port:  8153
```

(8154 is used for HTTPS/SSL automatically once agent/server are acquainted with each other)

Once set up the agent will attempt to register with the Go Server (by contacting the go.genivi.org server), and it will be listed as waiting.
An **administrator** needs to **Enable** the agent using the Web interface on the Go Server.

To Administrator: When an agent is enabled make sure you take the time to define its resources - see Agent Resources

Installing prerequisites in go-agent

Go agents need to have the software installed that is necessary to at least bootstrap a building process.

⚠️ *As noted above – this is now all handled by scripts – in particular the Docker setup is guaranteed to have the right content.*

Yocto prerequisites

Larger machines may be assigned jobs that include full system builds using Yocto / bitbake. We expect the minimum requirements for Yocto to be installed in each agent.
The Yocto Project Quickstart currently recommends the following:

```
 $ sudo apt-get update
 $ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat
libsdl1.2-dev xterm
```

⚠️ For newer Yocto versions, this might be outdated – see the gocd-setup instead

```
 $ sudo apt-get install gettext
```

For non-Debian package names, refer to the Yocto Quickstart. The Yocto reference manual lists more detailed information for special needs. You might choose to install all of them.

Also install the following:

General prerequisites

Some general/minimum prerequisites(*) for our non-Yocto builds (components built natively). Please add to this list

- Git
- GNU make
- cmake (CommonAPI tools, DLT and others)
- autotools / automake
- boost
- gcc and gcc-c++ a.k.a. g++
- wget and/or curl
- python
- libtool (libdbus)
- libexpat devel/headers (libdbus)
- maven (CommonAPI tools)
- pkg-config (various components)
- doxygen (often used for documentation generation)

```
 $ sudo apt-get install zlib1g-dev libglib2.0-dev libsystemd-daemon-dev git wget gcc g++ intltool autotools-dev automake make cmake python libtool maven libexpat1-dev pkg-config libdbus-1-dev libdbus-c++-dev libboost-dev libboost-thread-dev libboost-system-dev libboost-log-dev doxygen sudo apt-get install docbook-xsl
```

UPDATE: I recommend you instead use the packages listed in this file. It may be more up to date, and also provides package names for RPM-based distros.

(*) For build recipes that do not build all their prerequisites from source, additional packages might be required on a build agent.
For example, it would be typical of a program to depend on something common like zlib, or libdbus.

LBS

For LBS builds libdbus C++ (added above) and the following unique requirements:

```
$ sudo apt-get install gpsd libgps-dev xsltproc
```

Setting up a Go-Server

This is not recommended since the the point of the GENIVI-Go project is to collect up necessary build jobs into a shared environment (one single go-server - multiple agents) that everyone benefits from. But if you want to try it locally and then transfer what you created, then documentation is available on the Go.CD website and the easy-genivigo project might help if you want to try out a complete local environment.

Recreating/installing go.genivi.org:

Documentation about how go.genivi.org server is configured (for reconstruction in case of failure, etc.)
These are the basic steps (some tweaking may be required):

- The physical (or VM) server shall run a modern Linux distro with Docker installed
- Clone the gocd-setup project and use the **docker/server** directory
- There is a README inside that directory
- Make sure the Go Server configuration is set up to use HTTPS (only)
- Set up a separate Web server that will redirect any requests to HTTP/port 80 to the HTTPS URL: https://go.genivi.org
- Install the certificates according to this instruction: SSL/TLS cert on HTTPS://go.genivi.org

Additional information in this ticket: **OSSINFR-157** - Getting issue details... STATUS