Go Continuous Integration Server

Working page for information about go.genivi.org

Current administrator: Gunnar Andersson *Please report any noticed problems immediately*.
We have had a few different secondary maintainers but right now that role is *vacant* - contact Gunnar if you want to be more involved.

Additional information (sub-pages)

How do I use it?

Q: Do I need a personal account?

A: You need a personal account if you want to edit/add jobs or to manually trigger the pipelines to run.
*The **guest account** (user: **guest**, password: **genivigo**) is sufficient to study the results of existing pipelines.*

If you want a personal account, please see:

- Go User Account Creation

Installation

1) To use go.genivi.org, you do not need to install anything, just use a web browser and use the server with credentials as above. Skip to the next section.

2) If you can provide a Go **Agent** machine (i.e. a "build slave" machine part of the shared build cloud), then refer to:

- Go Installation Instructions

3) If you are interested in running your own test environment (both server and agent) locally, you might try easy-genivigo project (easy)
Or modify the scripts and configuration used for the official GENIVI go-server and go-agents (a bit more to do)

If you are an IT administrator and need to reinstall the **go.genivi.org site** then instructions are at the end of Go Installation Instructions.

Organization (for developers and users)

Overview

You may want to watch or modify the queue of **JIRA tickets.** If you have a JIRA login then it's best to look at the real JIRA task filter which shows Go-related tickets.

You can get a login to the JIRA and/or Confluence infrastructure by checking the instructions on the right hand side of the Wiki starting page.

Agents

Go Agents are what other systems might call "build slaves". They do the actual compilation Jobs assigned by the Go Server.

List of Go Agents and maintainers:
(This list needs continuous maintenance - please notify the administrator if you see a mismatch)

Agent Name	Location, Company and Maintainer contact	Operating system & setup	Comments
vbox-agent	Gunnar Andersson	Custom, *temporarily unavailable*	Specialized agent for **Vagrant/VirtualBox** jobs
genivi-builder	Main builder machine **Run by GENIVI IT, rented at Hetzner data center** (Go issues : Gunnar Andersson) (Machine issues: GENIVI IT mgr Nicholas Contino)	Standard Docker setup	**256GB RAM** **6-core / 12-thread CPU (slightly older)** **~2 x 400GB SSD combined in RAID0**
docssite-agent	Agent running on docs.projects.genivi.org machine to handle automated testing preparation and Lava integration **Run by GENIVI IT (AWS)** *This is a special purpose agent used to deploy files on the "docs" local disk (to be able to put them into web serving directories, and another storage location where lava.genivi.org can fetch test artifacts.* "docs" is the generic download site and also hosts binaries. (Go issues : Gunnar Andersson) (Machine issues: GENIVI IT mgr Nicholas Contino)	Standard Docker setup + install custom credentials for lava.genivi.org	N.B. The credentials to log into lava.genivi.org are manually added to the docker container.
webblack	Temporary. Machine is run by Gunnar Andersson *This is the fastest build machine on average*	Standard Docker setup	**64 GB RAM, modern CPU & fast SSD**

Collaboration

Modifying and adding pipelines will require us to keep good communication.

- Maintainers (see above) have superuser (Go Administrator) privileges.
- All others that have editing permissions can also create and modify pipelines. You are encouraged to experiment in the **Experiments_and_Tests** pipeline group. If the pipeline starts working, it can then be moved to another group.
- One way to keep up-to-date with changes is to track the continuous backup of the go-server configuration on GitHub (If you clone and pull this repository regularly you will be helping to keep a backup copy of the most relevant settings). Every change to pipeline configuration is committed to git and the server pushes it to the backup remote on GitHub once per hour.
- When in doubt, ask, for example on genivi-projects@lists.genivi.org. If there is significant discussions we may consider a mailing list for coordination.

Creating Pipelines

Detailed information about each pipeline and some processes for creating **new** pipelines can be found in:

- Go Pipelines Information.

Agent Resources

Resources are a way to specify what an agent provides.

Jobs that run as part of a *pipeline* stage can require particular resources and will then *only* be assigned to an agent carrying that resource.

Note that *jobs* that don't ask for any special resource can be assigned to any agent.

First, every agent should be tagged with a resource equal to the agent's host name. This allows us to easily schedule certain administration jobs to specific agents.

Resource Name	Purpose and explanation	Example of Pipeline needing this resource	Example of Agent having this resource

renesas_binaries	This signifies an agent machine that has been entrusted to keep a copy of licensed binary drivers for Renesas hardware, typically graphics drivers. Without those binaries the build would fail but since they are only distributed after user license acceptance, there is no opportunity for an Agent to download them as part of the pipeline input material. Special build steps are included to copy the binaries from a *predefined location* on an agent machine carrying this resource.	All GDP / Baseline builds for Renesas Hardware	Most Yocto_build enabled agents – please refer to the actual agent list on server (might need admin credentials)
yocto_build	This is used to indicate the agent is capable of running full Yocto builds of GDP size.	Full GDP / baseline builds using Yocto.	Most agents, unless they are known to be small (slow, or with little disk space).
deploy_genivi	This means the agent is colocated or otherwise has the ability to copy files directly to the disks involved in other parts of GENIVI infrastructure - most notably to copy files to the web server area so that they show up on https://docs.projects.genivi.org/releases		Single special-purpose agent.
test_<architecture>	This resource is specified on an agent dedicated to control a board farm for <architecture>, i.e. testing jobs can be assigned to run on it.	GDP & Baseline automated tests	
arm (N/A)	This is an agent machine that runs an ARM processor appropriate for **native** compilation/execution on ARM.		~~codethink-genivigo-agent0~~ *currently none used*

Environments

Environments are similar but a more strict way to subdivide part of the Go work into sections. An Agent can belong to maximum one environment and if it does, it will only run pipelines that explicitly belong to that environment. That is typical for deployment where you want to direct certain pipelines strictly to a particular machine, and no other pipeline type should run on that machine.

No environments are used at this time.

Parameters

Parameters are used to parameterize some pipeline templates. Refer to Go.cd documentation for explanation.

Templates

Pipeline templates are used heavily and encouraged. This reduces repetition of the build steps for pipelines that only differ in Material (git repo or branch) or configuration (using Environment variables or Parameters). Go.cd documentation

GitHub integration

Two plugins are used: Build GitHub pull requests and GoCD build status notifier.

These will build Pull Requests as soon as they are created and indicate directly in the Pull Request if the build was successful (⚠️ limitations below).

Installation:

Most of the information is in the README of the plugins (linked above) and won't be repeated. Here follows information specific to our installation:

Authentication:

The plugin to build pull requests uses an OAuth token that it fetches from a file: $HOME/.github on the go-server. (In our instance $HOME is /var/go)
The token is valid for the **genivigo** login name.

The other plugin that updates Github's status has its configuration under the Go-Server Plugin Admin interface (web interface).
Also there the same **genivigo** user OAuth token is used.

The user that is allowed to write the status of checks (**genivigo**) must be a collaborator of the project, and it seems it must have **write** access role. (If you set this up on your own, remember to limit the OAUTH token access as much as possible, except for repository access)

⚠️ Bugs: It used to be that ony PRs for **master** worked. This is reportedly fixed now.

HOWTO for maintainers:

To build pull requests all you need is the first plugin. To also update PR status you need to add the GitHub user **genivigo** as collaborator in your repository as above.

Then set up a separate pipeline according to instructions in the Readme:

- As you can see in the README it's a little complicated to add the pipeline because you need to create a pipeline (which immediately forces you to add one Material but GitHub is **not** given as a choice there). After saving, you go back and edit the pipeline to add new material of type "GitHub". And then you finally remove the original material. There are some hoops to jump through (make sure a subdir is defined for each material during the time you have multiple materials, and then you may want to remove the subdir usage if only one Material at the end)

- Once that is done, run the pipeline once manually (it will then build master branch, see below). After that it should pick up any new pipelines. If it doesn't work, use the Readme or ask that project.

Some deeper information about how pipeline design affects GitHub reporting is available in: **TOOL-86** - Getting issue details... STATUS