Continuous Integration with GameTeX

This document assumes that you're using Git for version control[1] and GitHub (or Bitbucket) or GitLab for hosting. If that is not true, you will not be able to fully follow these instructions.

1 Why CI?

"Continuous integration" (CI) is the idea, now quite popular in software development, that any time somebody commits to a software project their change should be immediately, automatically built and tested. In a context like Guild game writing, this has two chief advantages:

Finding compile errors quickly GMs will sometimes make a change and accidentally make it so the game no longer fully compiles. Without CI, this will be discovered whenever another GM updates and happens to try to compile whatever broke. If the breakage only impacted a few files, this may take a while, so by the time the issue is discovered there might be several commits to go through to find the error. With CI, within minutes of pushing the commit, the CI system will attempt to build the commit, fail, and email the GMs. Whoever made the change (or is best at GameTeX) then has only a single commit to look over to find the cause.

Downloadable sheets without LaTeX Less important, but possibly still helpful, a CI system can save the generated sheets and allow GMs to download them with a browser but no LaTeX compiler. This could be mildly useful for a variety of reasons – quick changes on a borrowed computer (using GitHub's online editor), a GM who can't get LaTeX to work, linking to a specific version, etc..

Why CircleCI or GitLab? I went into this assuming that most GMs would have two requirements:

Private repos They must be able to use a private repo so that players can't happen across the game and spoil themselves.

Free Most GMs won't be willing to pay much, if anything. $10/month *might* be reasonable, but is distinctly on the high side.

It turns out that continuous integration systems are mostly sold to people writing software, for whom this is a productivity tool worth much more than $10/month. Many of them are sold on a "freemium" model: they'll let you use it for free for public repos (which presumably have lower commercial value, but can be a good way to get users hooked), but charge a significant amount for private repos (which are more likely to be commercial companies). Unfortunately, Guild games want the private repos, but have little commercial value.

I found four candidate CI systems from a small amount of research:

TravisCI $129/mo for private repos

SnapCI $30/mo for private repos

Solano CI $15/mo for anything

CircleCI free[a]

There are presumably other CI systems, but having found one that supported what I needed, I stuck with it.

I was later introduced to *GitLab CI*, which also works.

[a] for non-parallel Linux builds – OS X or building multiple commits at once aren't free, but Guild games probably don't need those

2 Prerequisites

Before you get started, you will need some familiarity with Git and a GitHub or GitLab repo for your game that you have admin access to[2]. If you are using GitHub, you'll use CircleCI; if you are using GitLab, you'll use GitLab CI.

[1] Instructions on using Git for version control are at `https://adehnert.gitlab.io/TestGame/guildcamp-git.pdf`

[2] CircleCI also supports Bitbucket, so that may also work. Bitbucket private repos are free as long as your team has at most five members, so if you can't use GitHub (likely due to the private repos) and don't like GitLab for some reason, Bitbucket is another choice. I haven't tried using Bitbucket or CircleCI's support for it, but they're presumably straightforward.

The most obvious advantage of using GitHub is that it is very common – odds are decent that some of your GMs may have used it before. However, it has various disadvantages:

Private repos You will almost certainly want a private GitHub repo, which requires somebody on your team either paying GitHub money or being a student. Even if one applies, you can probably only have private personal repos, which ties the repo pretty closely to the owner. Notably, much of the CircleCI setup will need to be done by them.

Permissions CircleCI requires a bunch of permissions. If you use GitHub for other things, using GitLab (or a second GitHub account) may be preferable.

GitLab strikes me as a better choice than GitHub for many GM teams.

3 Setup

You will only have to do this step once per game.

3.1 Build instructions: `vgametex.py` and `Makefile`

Your CI system will need to know how to build your game. I use two common prereqs for doing this:

vgametex.py This script eliminates the need to set up your `.bash_environment` to build the game. You can use it to run commands (such as latex), and it will set up the GameTeX variables automatically. Download it from `https://adehnert.gitlab.io/TestGame/vgametex.py`, and save it as `Extras/vgametex.py` (and, of course, commit and push it to your repo).

Makefiles The Makefiles encode how to actually build the game. We currently have two: one to build the full printable set of sheets (which should catch most errors), and one to build greensheets (on the theory that these are what you're most likely to want to stick on a website to hand out to players). It should be simple to add more if you want. Download them from `https://adehnert.gitlab.io/TestGame/Makefile-prod` (save to `Production/Makefile`) and `https://adehnert.gitlab.io/TestGame/Makefile-green` (save to `Greensheets/Makefile`).

You can also use these locally to build the game – just `cd` to the relevant directory and then run `make` to generate updated sheets. It's not perfect, but it'll usually recompile things when they need to be recompiled, and most GameTeX games won't need internal references, which can require multiple runs of LaTeX to get right. Note that CircleCI will be building from scratch each time, so Make's incremental build support isn't actually needed.

(There are certainly other ways to set up CI – `vgametex.py` could certainly be skipped, for example, at the cost of more per-game customization of the CI config. I think this is one of the simpler ways to do and to explain, though.)

3.2 Add CI build instructions

Whether you're using CircleCI or GitLab CI, you'll need to write a config file that installs the LaTeX packages needed to build the game, builds the game, and saves the artifacts (anything you'll want after – printable PDFs by type, publishable individual greensheet PDFs, etc.).

If you're using CircleCI, create a (hidden) `.circleci` directory in the root of your repository, and then save `https:`

`//adehnert.gitlab.io/TestGame/circleci-config.yml` to `.circleci/config.yml`. If you need to do further customization, more documentation is online.

If you're using GitLab CI, save `https://adehnert.gitlab.io/TestGame/gitlab-ci.yml` to `.gitlab-ci.yml` in the root of your repo. If you need to do more customization, more documentation is online.

If your game has GameTeX installed to the root of your repo (that is, `Production` and `.git` are in the same directory), the CircleCI and GitLab CI config files should just work. If not, everywhere `Production` or `Greensheets` appears in `config.yml` (CircleCI) or `.gitlab-ci.yml` (GitLab), prepend the path to your GameTeX install. For example, if your GameTeX install is at `GameTeX`, you would put `GameTeX/Production` in several locations.

4 Using your CI setup

4.1 CircleCI

Each GM, starting with the private repo's owner, will need to visit the CircleCI website (`https://circleci.com/`) and choose to log in. After authorizing CircleCI with GitHub, you can choose "Add Projects" to set up the project. (If you've already signed up, you can pick the "Projects" tab in the left side bar, and then "Add Projects" will be in the top right.) Pick your game's repo. The repo owner should see an option to "build project", after which other GMs should see an option to "follow project".

To view built files (that is, your compiled sheets), choose "Builds" in the left sidebar, pick a build (presumably the most recent one), and then choose "Artifacts". You'll be shown a hierarchy of built files (under a label `$CIRCLE_ARTIFACTS`), and can expand the hierarchy and download the files by clicking the links.

You can sign up for build emails under "User settings" (in the left side bar), followed by "notification settings". After that, the options should be fairly intuitive – I'd generally lean towards "Send me a personalized email every time a build on a branch I've pushed to fails; also once they're fixed.", but of course it's up to you.

For debugging, one thing to know is that CircleCI does allow you to ssh into the build VMs, which can be very useful. If the build doesn't just work out of the box (but is working locally), I recommend sshing in, installing missing packages and whatnot interactively, and only once you've gotten it to build that way updating `config.yml` with the corrected build recipe.

4.2 GitLab

GitLab seems to be easier to configure. By virtue of adding your GMs to the repo, they have access to the build results, and it appears that by default they will receive emails when a build fails. (That said, with a small GM team, changing the notification setting (`https://gitlab.com/profile/notifications`) to "Watch" rather than "Participate" (the default shows up as "Global", and the default global value is "Participate") may make sense – GMs are reasonably likely interested in what other GMs are changing. This or a custom setup may also be required to get notified of other GM's failed builds.)

To view built files (that is, your compiled sheets), choose "CI/CD" in the left sidebar, pick "Pipelines", and then in the "Stages" column click on the checkmark or X and then "build game" from the dropdown. The bulk of the page will show you output from the build process, and on the right side you can "Browse" the "Job artifacts". Most files can be viewed inline, and after viewing you can also click the cloud icon in the top right to download the file.

The sample GitLab CI configuration sets up "GitLab Pages" to publish the scenario and rules document – if you set up your game in a GitLab group named "groupname" and your project is "projectname", you (and anybody else you give the link to) should be able to view it at `https://groupname.gitlab.io/projectname/`. To publish public greensheets or other content,

you can modify the `.gitlab-ci.yml` to copy additional files into place. If you want the index page to link to those additional files, create `public/index.html` as an HTML file with appropriate links.

Unlike CircleCI, GitLab CI does not appear to allow ssh'ing in to build images – see this issue.

5 Afterword

Hopefully this is all straightforward and your game will build just fine. Feel free to contact me (adehnert) for help. I'd also love to hear from you if you tried to get this working and whether or not it proved useful.