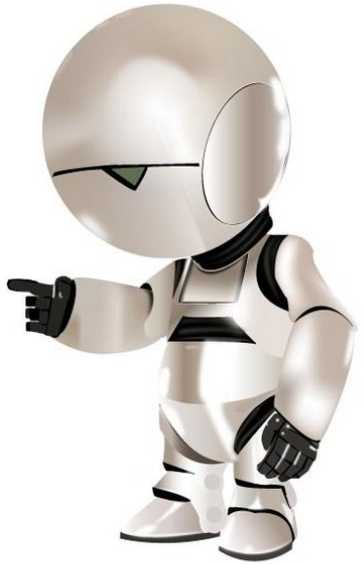
Bot to automate workflow of adding software to EESSI

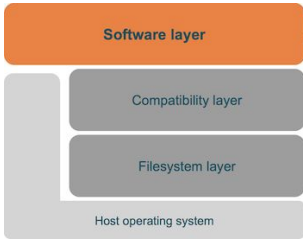
10 May 2022

Agenda

- Scope & goals
- Tasks for the bot in context of EESSI
- Exploratory work done during EESSI hackathons
- PyGHee as a “base” library for implementing the bot
- A bot for EasyBuild, EESSI, and beyond
- Development setup & plan (proposal)



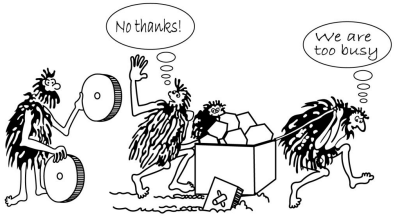
Scope & goals



EESSI

- Collection of software in EESSI pilot repository is still rather limited
- Mostly held back by **lack of automated building + deploying**
- We (also) want to open the door for community contributions
- We can’t expect that contributors will set up a build node for every target CPU
- Building + deploying should also be automated to be more secure

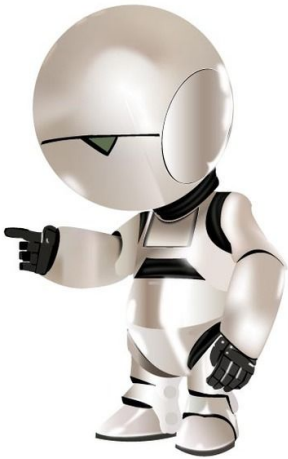
Scope & goals



- Current workflow to add software installations to EESSI is **too manual**
  - install_software_layer.sh script is started manually on build nodes
  - Tarball with new software installations is created with create_tarball.sh script
  - Tarball is uploaded manually to S3 bucket for ingestion on Stratum-0
    using `eessi-upload-to-staging` command
- PR #120 adds one-script-to-rule them all, but:
  - PR is old/stale, doesn't use existing `build_container.sh` script (but adds a new script)
  - Doesn't help much with building for different target CPUs
  - Still triggered by a human, so remains hurdle to facilitate community contributions…

Tasks for bot in context of EESSI

- React to pull requests that affect the EESSI software stack
- **Build** missing software installations
  - In a controlled sandbox environment (build container)
  - For all CPU targets
  - Only after a human reviews the PR?
- **Test** new software installations
  - In a sandbox environment different from build environment (different OS)
- **Deploy** software installations to EESSI repository
  - Only after human approval
- **Verify** that software was indeed deployed correctly
  - And then automatically merge the pull request
- Also update Lmod cache?



Designing a bot for EESSI

- Run somewhere with access to target CPU architectures?

  => Spawn VM / job on machine with target arch

- Listen to events from GitHub repo storing easystack file(s) for EESSI software stack

- Determine + implement actions to be performed based on events received:

  launch VM / job, set up build node / environment, build software,

  run tests, send new events (?), ingest built software into repository

Bot implementation

- Implemented in Python, on top of existing Python packages
  - Flask, waitress, PyGithub, PyGHee
- GitHub App (see docs)
  - Triggered by events coming from GitHub
  - Creating of PR, posting of a comment, adding a label, …
- Submits jobs to Slurm for different “stages” (build, test, deploy?, verify?)
  - So any Slurm cluster could become a “provider” for EESSI
- Stateless?
  - Try to avoid having to manage a database for ongoing work…

Exploratory work done during EESSI hackathons

- Mostly by Bob + Jörg (+ Kenneth)
- See hackathon notes + 03_workflow subdir in hackathon repo
- WIP bot implementation in eessi-bot-software-layer repo
- Scripts in hackathon repo
- Implemented functionality
  - Triggered when pull request is opened
  - Read configuration file
  - Obtain (temporary) GitHub token
  - Downloading easystack file from PR + submitting job to “build” missing software, incl. setting up a dedicated subdirectory per PR as working directory

PyGHee as a base library

- https://github.com/boegel/pyghee (pronounced as “piggy”)
- Small library that facilitates implementing a GitHub App in Python
- Takes care of:
  - Logging of incoming events
  - Verifying events (see docs)
  - Collect event information in Python dictionary
  - Calling event handler (if available)
- PyGHee is by no means mature yet…
  - Implemented in just two weekends
  - Could provide a lot more (like easy way to post comments back into PR)
  - Hasn’t been battle-tested at all, only basic test apps were implemented on top of it

Example of a bot implemented on top of PyGHee

Full code of a bot that logs posted comments in issues/PRs:

```
import waitress

from pyghee.lib import PyGHee, create_app
from pyghee.utils import log

class ExamplePyGHee(PyGHee):

    def handle_issue_comment_event(self, event_info, log_file=None):
        """
        Handle adding/removing of comment in issue or PR.
        """
        request_body = event_info['raw_request_body']
        issue_url = request_body['issue']['url']
        comment_author = request_body['comment']['user']['login']
        comment_txt = request_body['comment']['body']
        log("Comment posted in %s by @%s: %s" % (issue_url, comment_author, comment_txt))
        log("issue_comment event handled!", log_file=log_file)

if __name__ == '__main__':
    app = create_app(klass=ExamplePyGHee)
    log("App started!")
    waitress.serve(app, listen='*:3000')
```

A bot for EasyBuild, EESSI, and beyond

- A “build and deploy” bot could also be useful outside of EESSI
- HPC sites using EasyBuild may also want to start using a bot like this
- Implementation of build/test/deploy phases would be different
- But overall workflow would be very similar…
- Move as much generic functionality as possible to PyGHee
- Keep EESSI-specific stuff separate (shell scripts that bot just runs, derived class, …)
- Should bot be developed under `easybuilders` or `EESSI` GitHub organisation?

Development setup (proposal)

- Continue development in EESSI/eessi-bot-software-layer GitHub repo
  - Use separate branches to develop/test bot + open PRs when progress was made
  - Two-pairs-of-eyes principle: don't merge your own PRs
  - Test-driven development would be nice (write test along with implementing changes)
- Bot is running on EESSI CitC cluster (`eessi-bot` account)
  - Only small group of people have access to update & restart bot
- Use PRs to EESSI/hackathons repository for testing during development
  - Bot on CitC cluster is configured to obtain events from EESSI/hackathons repo via https://smee.io/7PIXBDoqczjEVXaf

Development plan (proposal)

- 0) Port existing implementation to leverage PyGHee
- 1) Rework “build” phase
  - Start with a single CPU target, then expand to multiple CPU targets
  - Let bot just submit a “build.sh” job script in prepared working directory to install software + create tarballs
  - Bot should “follow up” on running job(s), take action when ready (trigger event from job script?)
  - Bot should report back in PR by posting a comment
- 2) Also implement “deploy” phase
  - Bot should create tarballs and upload them to S3 bucket for ingestion (by running “deploy.sh” script?)
  - Trigger should be human action (adding a label? approved review?)
  - Bot should merge PR when deployment is complete & verified (by running “verify.sh” script?)
- 3) Update Lmod cache(s) after additional software installations were deployed
  - Triggered by merged PR event?
- Later: test phase, ask bot to give status update, …
- On the side: enhance PyGHee library with extra features to clean up actual bot implementation

Practical stuff

- Main communication channel: #software-layer-bot channel in EESSI Slack
- Joint HackMD note: https://hackmd.io/g-QbUpGqT6OUsLCEyIHIcw
- Quick meeting every couple of weeks to discuss progress + next steps?
- If you're actively working on something, let the others know
- If the running bot should be updated, ask Kenneth or Bob (via Slack)
- Set up your own bot in account on CitC cluster for easier testing/development?