

Software Planning: What to do about externally developed packages?

Stephen Geer, Michael Kagan, Jeremiah Mitchell,
Mandy Rominsky, Ariel Schwartzman, Dylan Temples
Offline Coordination Group

MAGIS-100 Science & Simulation Meeting,
March 19, 2024



MAGIS GitLab

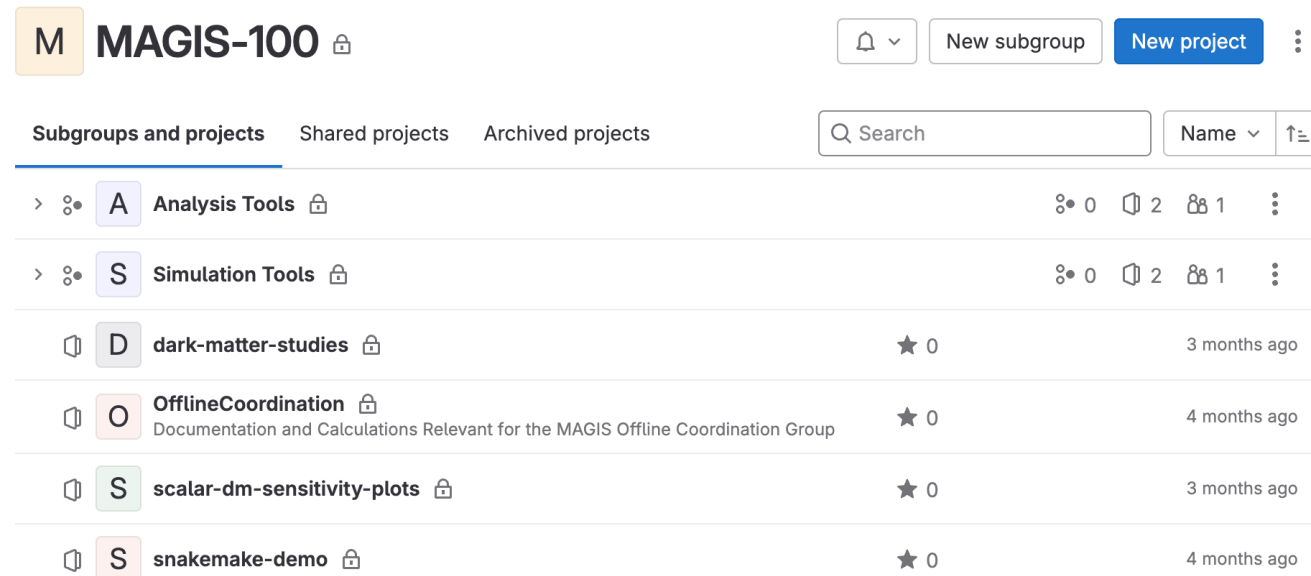
2

Thanks to the hard work of Dylan and Jeremiah, we now have a MAGIS-100 Gitlab

- Please contact Dylan if you'd like to start using this space before its opened to the whole collaboration
- Feedback on structure critical

Eventually we would like all production code for simulation, data processing, and official analysis to be here*

*But... what about externally developed packages?



The screenshot shows the GitLab interface for the 'MAGIS-100' group. At the top, there is a navigation bar with a bell icon, a 'New subgroup' button, and a 'New project' button. Below this, there are tabs for 'Subgroups and projects', 'Shared projects', and 'Archived projects'. A search bar and a 'Name' dropdown menu are also visible. The main content area lists several subgroups and projects:

Subgroup/Project Name	Stars	Members	Created
Analysis Tools	0	2	1
Simulation Tools	0	2	1
dark-matter-studies	0	-	3 months ago
OfflineCoordination Documentation and Calculations Relevant for the MAGIS Offline Coordination Group	0	-	4 months ago
scalar-dm-sensitivity-plots	0	-	3 months ago
snakemake-demo	0	-	4 months ago

External Packages Developed by MAGIS Collaborators

Several packages are being developed for general purpose, not necessarily specific for MAGIS but certainly useful / critical

- Stanford Timing Interface
- GradOptics
- ...

Some of these may be under active development outside MAGIS

Potentially may also benefit from developments for MAGIS specific purposes

Should we integrate (a version) of these packages in MAGIS?

How best to handle such software

1. Fork repos, bring inside MAGIS gitlab, develop inside MAGIS

- Cons:
 - Will drift from main repo, potentially not benefitting from outside development
 - Won't necessarily have same support, need to develop it
 - Won't benefit from open source developments, may need to do on our own
 - Unless... it makes sense for authors to move permanently into MAGIS
- Pros:
 - More control to implement developments we need
 - Develop more inside expertise
 - Can better guarantee long-term support by collaboration

Should we integrate (a version) of these packages in MAGIS?

How best to handle such software

2. Leave outside MAGIS, use a tagged version of code (e.g. from PyPI)

- Cons:
 - Less control over developments and when changes implemented / merged
 - May not be able to guarantee long-term maintenance from outside
- Pros:
 - More involvement of developers
 - More involvement of opensource community
 - Developers may prefer to keep code independent

Moving forward

Are there other options we didn't mention here?

What are your thoughts as collaborators on how to handle this SW?

This isn't meant to inspire people to write all their code externally

- Code developed for MAGIS should be inside MAGIS Gitlab
- What if it is of general use? → we should discuss this

Basically, we need to develop a policy on how we handle code written for MAGIS, and code useful for MAGIS but written externally

Question: Will all MAGIS code be public? Maybe this will alleviate some of the developer concern for moving code into MAGIS