Table of contents

[Generally how a hack/doc event is / should be run 1](#_Toc500238983)

[Pre-hack/doc for the project leads 1](#_Toc500238984)

[Best practices 2](#_Toc500238985)

[Tools to support technical collaboration 2](#_Toc500238986)

[A model for using github to work on a project 3](#_Toc500238987)

[What a collaborative dev process looks like 3](#_Toc500238988)

[Pre-hack/doc for everyone 3](#_Toc500238989)

[Hack/doc event - ideally 2 days 4](#_Toc500238990)

# Generally how a hack/doc event is / should be run

* the projects should be chosen and fully defined prior to the hack/doc - every effort should be made to minimize decisions (especially regarding spec and scope) that need to be made at the event itself
* open to and supportive of all technical levels and backgrounds - hack/docs are great places to learn and teach!
* expected/ideal outcome is running code and/or published documentation - NOTE: a tech-heavy project team still might do a documentation project (e.g. codifying best practices)
* it’s very useful/good to have at least one person at the event who is focused not on the projects themselves but instead on making sure everyone has working tools / environment (that person may also help with a project, but the project should not be their primary focus)
* ideally food during the work day should be provided so that flow is minimally disrupted

# Pre-hack/doc for the project leads

1. project(s) chosen - main projects, and smaller, secondary projects if the main projects are completed faster than expected
2. a project lead is chosen for each project
3. everyone participating in the hack/doc has github accounts and has communicated their username to the project leads
4. if the project doesn’t already have a main repo in the organization github space, the project lead creates one
5. the project lead clones the main repo into a dev repo for the hack/doc and sets up appropriate access

# Best practices

* A development environment(s) that replicate production but are independent of it (e.g. not just a hidden site on a production system)
* Are you working with a Git repo? If so, are you running from HEAD? If not, which branch?
  + We run HEAD during hack/docs.

# Tools to support technical collaboration

* Project management (PM) tool that has
  + issue tracking
  + discussion/planning
* Source control
  + github, because
    - free for open source work
    - supports organizations and users
    - has sufficient (for now, at least) built-in PM tools
    - has built-in wiki for documentation/discussion/planning
    - can easily be hooked up to travis for CI testing
    - widely used in the coding community
* Standardized development target / environment
  + Is there a vagrant box?
  + Generally we use the islandora community vagrant box
    - base vs …‘not-base’?
* Toolset that everyone can use
  + vagrant, virtualbox, git
  + text editor (vi, emacs, phpstorm, atom, jedit, notepad++, sublime text 2)
  + Testing environment
    - We use the automated testing provided by drupal’s simpletest framework
  + google docs

NOTE on ISLANDORA dev: work in the backend (e.g. solr config files) or within drupal (e.g. custom ingest forms) would still use a source control repository but might not contain directly runnable or embeddable code - e.g. might be ansible scripts, XML files, best practices docs, etc.

# A model for using github to work on a project

1. on github: start from the main/release/official project repo
2. on github: if the project doesn’t yet have a main repo, set up one in the organizational github
   1. We start at ICG <https://github.com/Islandora-Collaboration-Group>
3. on github: fork the main repo to a dev repo that you own
4. on github: set up any necessary additional access/collaborators for your dev repo
5. locally: set up your dev environment (eg. islandora vagrant box) (NOTE: might have to modify the vagrant file to make a given directory accessible using your normal OS tools)
6. locally: clone the dev repo from github to your local environment (in the directory you made accessible to your normal OS, if applicable)
7. locally: do work, committing locally often and pushing to github dev repo at appropriate points
8. locally: final commit and push to GH dev repo
9. on github: create a pull request from dev repo to main repo
10. main repo manager processes the pull request (test, merge in, review, reject, whatever)

# What a collaborative dev process looks like

* there’s a member organization that owns the code
* each project has a lead that does initial repository set-up and access management and such, as well as managing discussion/planning related to the project
* each project has its own repository
* any development work is done on/in a clone of the main repository, then a pull request is created to get code from the dev repo back into the main project repo
* ideally there’s an automated testing step that happens when dev work is pulled into the main project repo, and the pull is rejected if the testing fails
* NOTE: this organization can work well for non-code projects as well - it’s perfectly reasonable for a repo to contain just documentation / text

# Pre-hack/doc for everyone

1. verify that you have the necessary tool set (this might be a mini pre-hack meeting the night before or some such thing): git, vagrant, virtualbox, text editor
2. verify that you have a local dev environment set up (this should have been specified in advance by the hack/doc organizers and/or project leads
   1. OUR default is the Islandora Vagrant box
3. Clone the dev repo into your local dev environment
4. Verify that you have registered for wireless internet access
5. Double-check messages from organizers and project leads - is there anything they have asked you to do in advance that you have not yet done?

# Hack/doc event - ideally 2 days

1. day 0: pre-hack option for those that want/need help getting tool set etc. working
2. day 1 morning: general intro (who folks are (name, institution, relevant skills/background), event runner, event tool specialist, meal plans, facilities, etc.), event technical info (docs folders, wifi passwords, hashtags, back-channels, etc.), overview of each project (name, goal, expected skills / roles needed, etc.), set up teams if that’s not already figured out
3. day 1 most of day: project team(s) do work
4. day 1 end: quick report from each team - progress, roadblocks, needs, etc.
5. day 2 morning: set up new, small projects if anything was totally wrapped up the previous day
6. day 2 most of day: project team(s) do work
7. day 2: push work back up to the dev repo if it’s ready
8. day 2: when ready, create a pull request to get that work back into the main repo
9. day 2 end: project team(s) final report, projects pulled to main repo if ready, hack/doc event review\*\*\*\*
10. day 2 end: project team makes plans for any post-hack/doc work necessary\*\*\*\*

\*\*\*\*THIS may also happen post-hack/doc if times runs out!