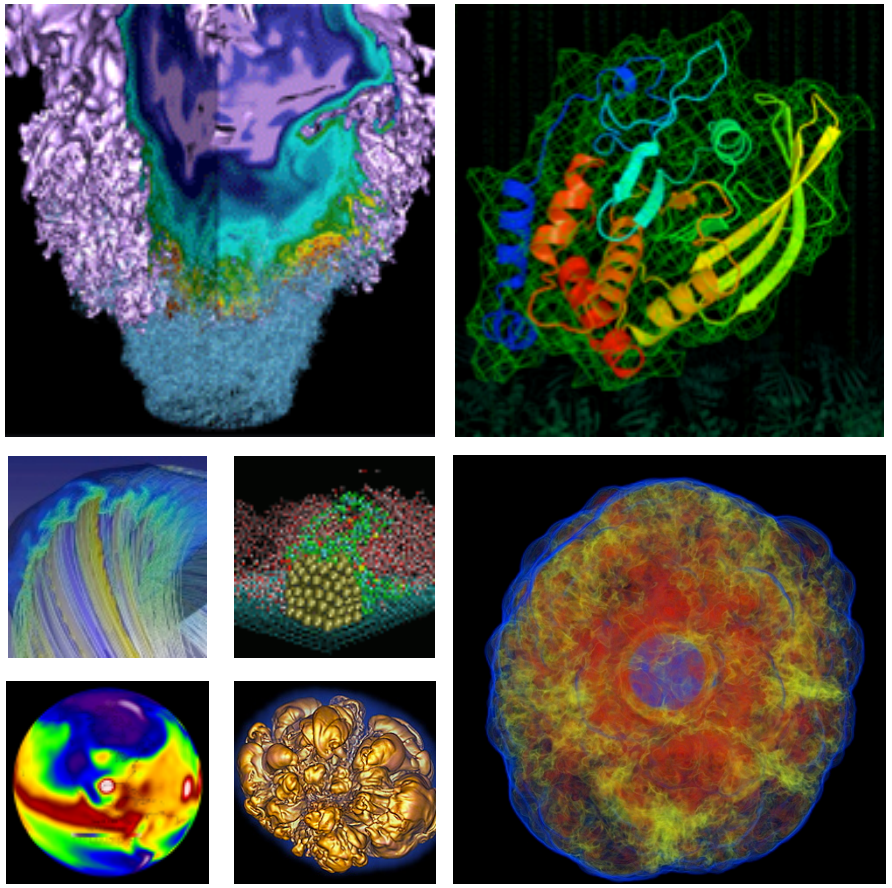
Advanced Git and Gitlab



Tony Wildish
Dan Udwary

May 30, 2017

Advanced Gitlab

- **Prerequisites**
- **Branching and Tagging**
- **Building multiple containers**
- Pushing images to multiple repositories
- **Using metadata in containers**
- Deploying runners on NERSC hosts
- **Best practices & recommendations**

- **=> Get the code for this tutorial:**
  - Fork the tutorial repository, then clone your fork to your laptop
  - https://gitlab.com/TonyWildish/gitlab-advanced/

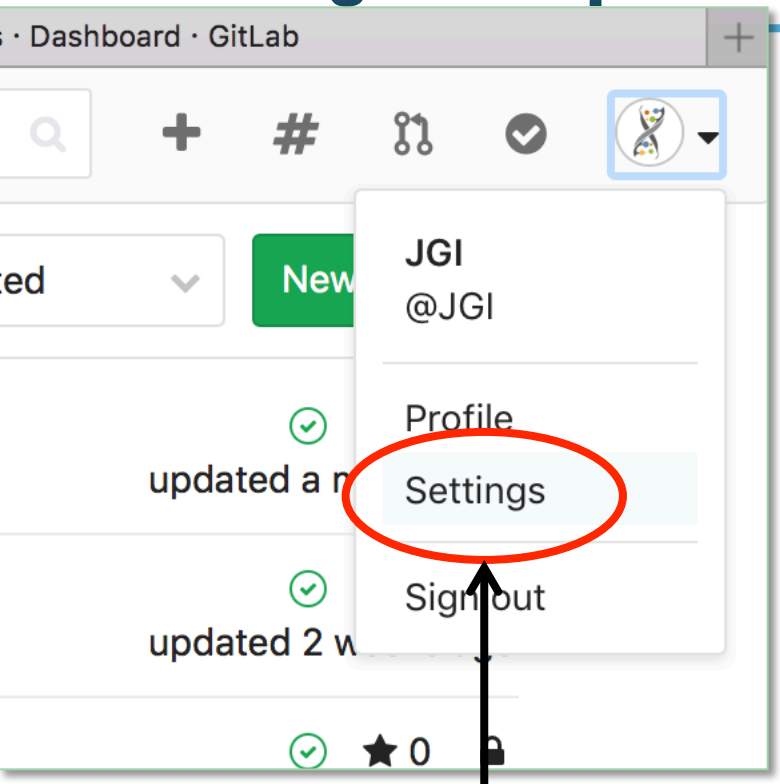
Prerequisites

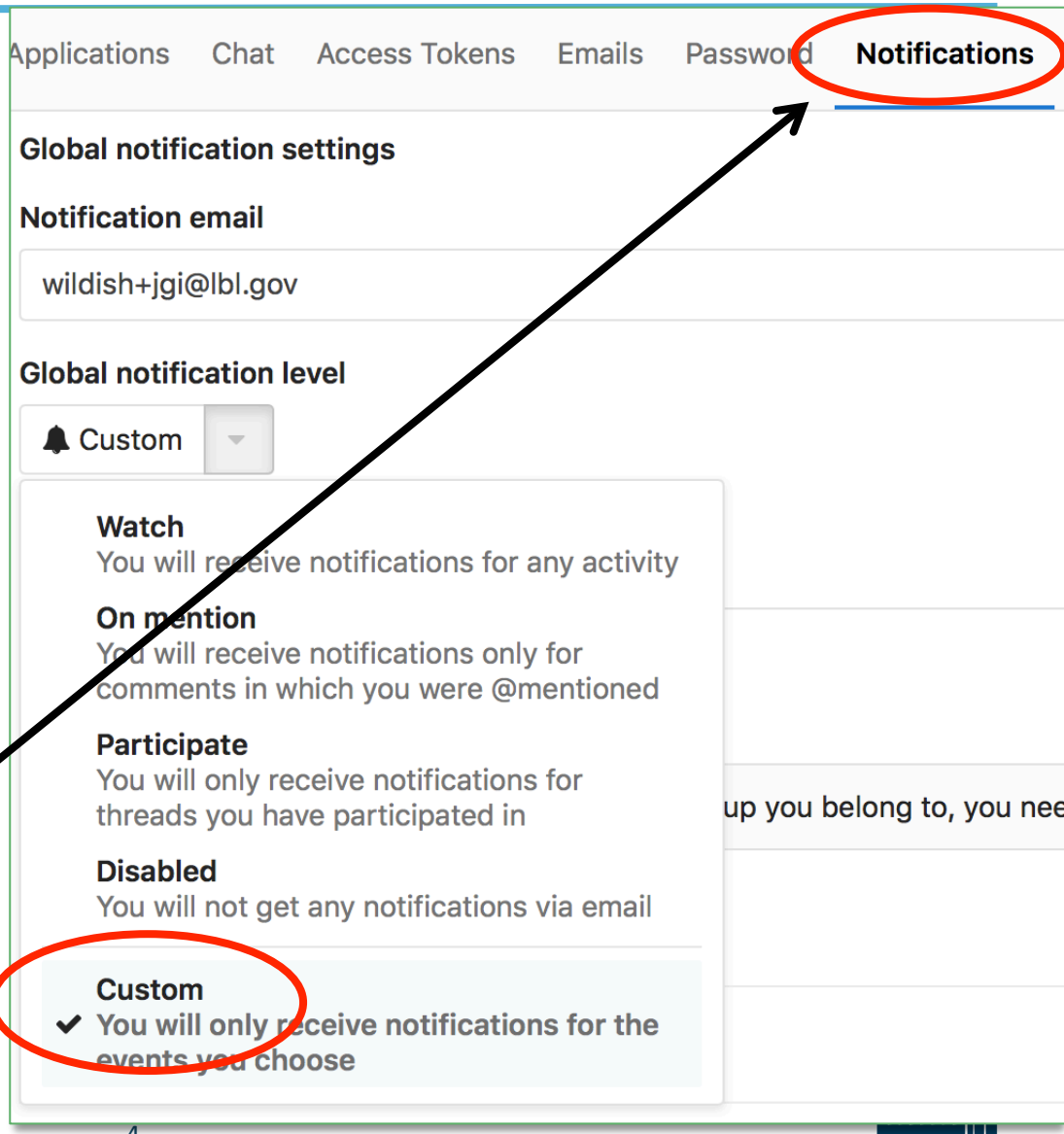
- **Familiarity with git, docker, gitlab**
  - Git, version 2.11 or higher
  - Docker, version 1.12.3 or higher
  - An account on gitlab.com

- **Earlier tutorials:**
  - https://www.nersc.gov/assets/Uploads/Git+Docker-Tutorial-Dec01-2016.pdf
    - Do exercises 4 and 5
  - https://www.nersc.gov/assets/Uploads/2017-02-06-Gitlab-CI.pdf
    - Do the first exercise

https://gitlab.com/TonyWildish/gitlab-advanced/

Bonus gitlab tip: Notification emails

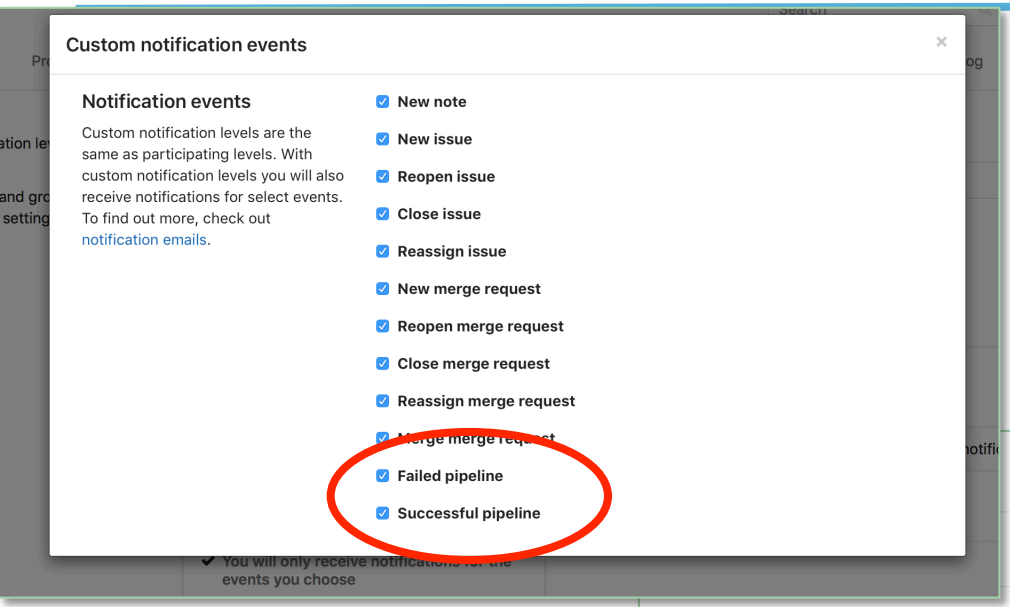




Go to your account settings,

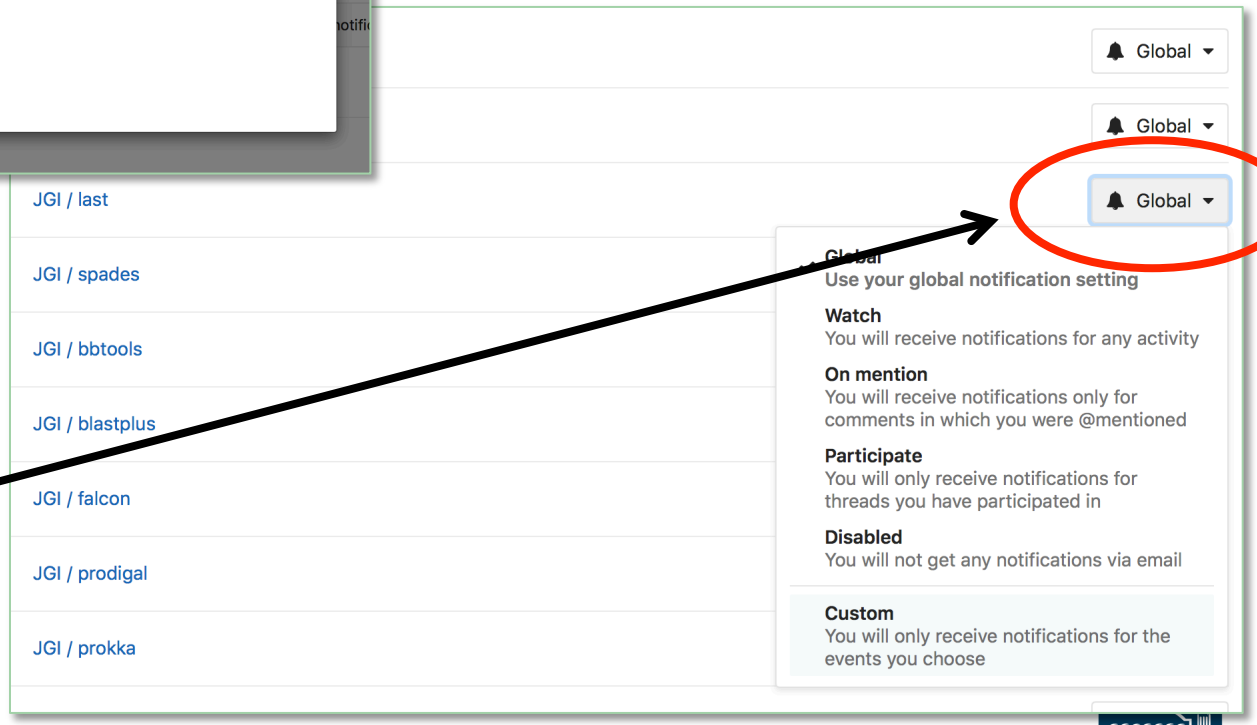
Notifications, Custom

Bonus gitlab tip: Notification emails



Choose any/all fields,

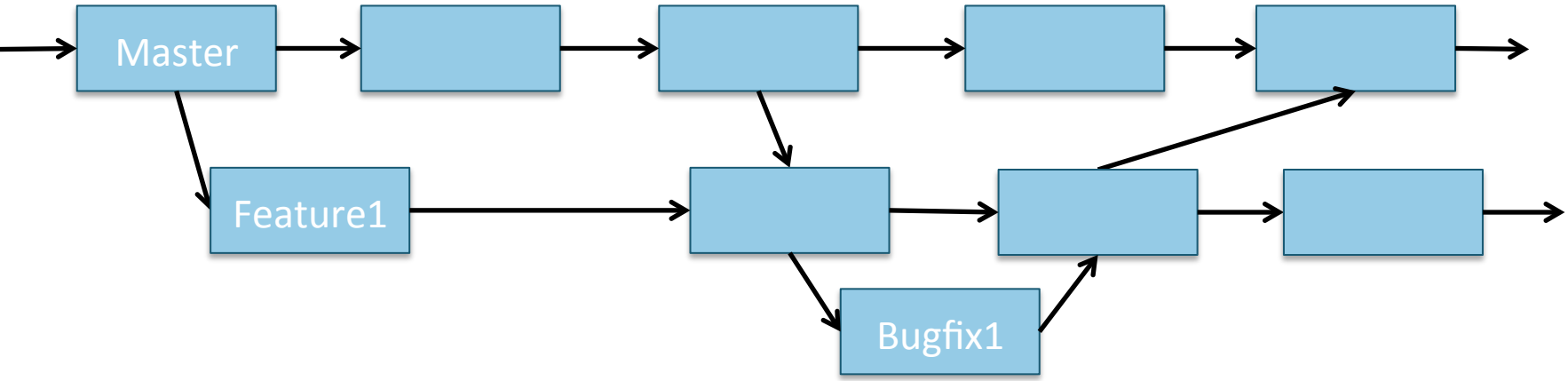
can set per-project too



Branching and tagging

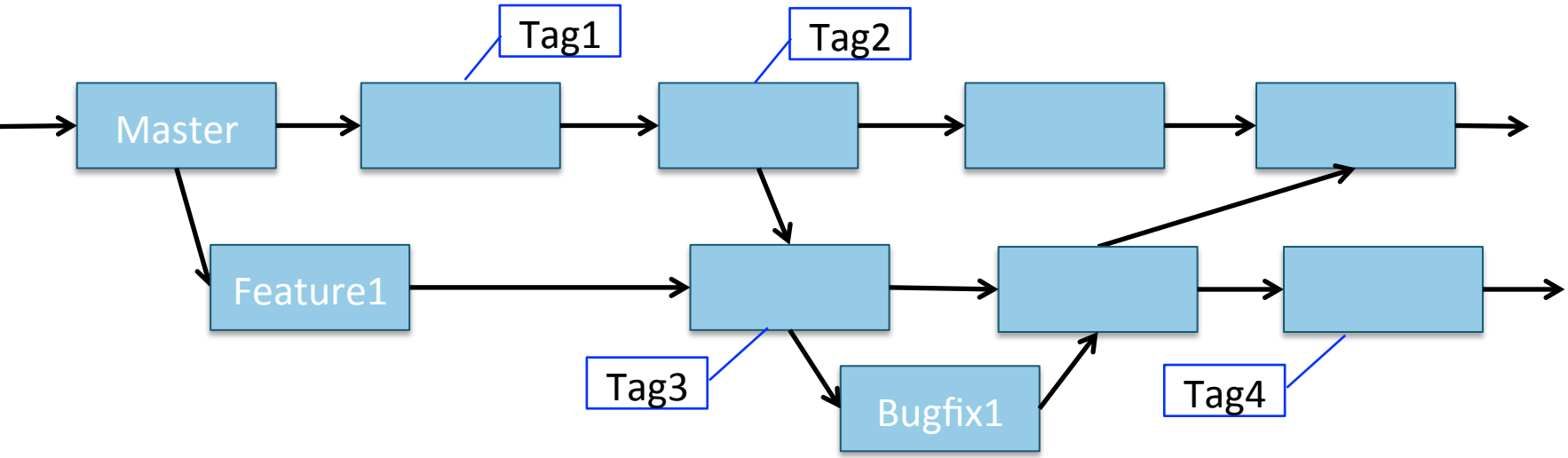
- **Branches**
  - Allow parallel development in a single repository
  - Create branches as needed, delete when obsolete
  - Can merge branches if you like, or keep forever
    - Bugfix branches: merge, delete the branch
    - Feature branches: keep forever.
      - Can merge back & forth to control divergence

"Pro Git", by Scott Chacon, Chapter 3



Branching and tagging

- **Tags**
  - Static label, identifies a particular commit
  - Easily recover particular version at any time in future
  - Once pushed, tags shouldn't be deleted or moved!



Branching and tagging

- **Tags and branches in gitlab**
  - Can be used to identify build products, label images etc
    - If there's a tag, use that
    - If not, use the branch name
    - 'master' branch -> 'latest' docker version (by convention)
  - Let's do exercise 01!

https://gitlab.com/TonyWildish/gitlab-advanced/

Working with forked repositories

- **How do you keep a forked repository up to date?**
  - Add the original source as another 'remote' repository

> **git clone git@bitbucket.org:TWildish/jgi-lapinpy.git**
Cloning into 'jgi-lapinpy'...
[…]

> **cd jgi-lapinpy/**
> **git remote add upstream git@bitbucket.org:berkeleylab/jgi-lapinpy.git**
> **git remote -v show**
origin git@bitbucket.org:TWildish/jgi-lapinpy.git (fetch)
origin git@bitbucket.org:TWildish/jgi-lapinpy.git (push)
upstream git@bitbucket.org:berkeleylab/jgi-lapinpy.git (fetch)
upstream git@bitbucket.org:berkeleylab/jgi-lapinpy.git (push)

> **git pull upstream master**
From bitbucket.org:berkeleylab/jgi-lapinpy
 * branch   master   -> FETCH_HEAD
Updating a3f5e1e..03943c8
Fast-forward

"Pro Git", by Scott Chacon, Section 2.5

Working with forked repositories

- **How do you keep a forked repo up to date?**
  - Add the original source as another 'remote' repository

```
> git clone git@bitbucket.org:TWildish/jgi-lapinpy.git
Cloning into 'jgi-lapinpy'...
[…]

> cd jgi-lapinpy/
> git remote add upstream git@bitbucket.org:berkeleylab/jgi-lapinpy.git
> git remote -v show
origin  git@bitbucket.org:TWildish/jgi-lapinpy.git (fetch)
origin  git@bitbucket.org:TWildish/jgi-lapinpy.git (push)
upstream  git@bitbucket.org:berkeleylab/jgi-lapinpy.git (fetch)
upstream  git@bitbucket.org:berkeleylab/jgi-lapinpy.git (push)

> git pull upstream master
From bitbucket.org:berkeleylab/jgi-lapinpy
 * branch            master     -> FETCH_HEAD
Updating a3f5e1e..03943c8
Fast-forward
```

"Pro Git", by Scott Chacon, Section 2.5

Building multiple containers

- **Suppose you have a particular package with:**
  - A few core dependencies, very small total
  - Several optional extras that add hundreds of MB
- **How do you build an optimal container?**
  - Include everything -> baggage that not all users need
  - Leave stuff out -> don't satisfy all users
- **Solution:**
  - Build two containers (or more) in the same repository

https://gitlab.com/TonyWildish/gitlab-advanced/

Building multiple containers

- **Gitlab supports building Docker images with names other than the repository name**
  - Default Docker name structure
    - $REGISTRY_USER/$APPLICATION:$RELEASE_TAG
  - Extended syntax:
    - $REGISTRY_USER/$APPLICATION/real-name:$RELEASE_TAG
  - Use extended syntax repeatedly in .gitlab-ci.yml, with different ‘real-name’s
  - “myapp-lite” & “myapp”, or “myapp” & “myapp-full”
  - See exercise 02!

https://gitlab.com/TonyWildish/gitlab-advanced/

Pushing images to multiple repositories

```
variables:
  GIT_STRATEGY: clone
  GITLAB: registry.gitlab.com
  SHIFTER: registry.services.nersc.gov:8443
  REGISTRY_USER: TonyWildish
  APPLICATION: gitlab-advanced
  RELEASE_TAG:
  DOCKER_DRIVER:


stages:
  - build

build:
```

GITLAB and SHIFTER variables point to different registry hosts

https://gitlab.com/TonyWildish/gitlab-advanced/

Pushing images to multiple repositories

```
script:
  # Log into the gitlab docker registry. Work out what the name
  - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $GITLAB
  - if [ "$RELEASE_TAG" == "master" ]; LEASE_TAG=
  - export DOCKER_IMAGE=`echo $REGISTRY TION:$RELE
  - echo "Build/deploy $DOCKER_IMAGE"

  # Build it and push it to gitlab
  - docker build -t $GITLAB/$DOCKER_IMAGE .
  - docker push $GITLAB/$DOCKER_IMAGE
  - echo "Pushed $GITLAB/$DOCKER_IMAGE"

  # Re-tag the same image and push it to NERSC
  - docker tag $GITLAB/$DOCKER_IMAGE $SHIFTER/$DOCKER_IMAGE
  - docker push $SHIFTER/$DOCKER_IMAGE
  - echo "Pushed $SHIFTER/$DOCKER_IMAGE"
```

Build and push to GITLAB

Re-tag, push to SHIFTER

https://gitlab.com/TonyWildish/gitlab-advanced/

Pushing images to multiple repositories

- **Caveat: Security!**
  - Gitlab hands you a login-token for every build
  - For shifter, once you're inside the firewall, there's no authentication needed, so no token
  - Anywhere else, you probably need a token or password, but where do you store it?
    - Can't be in the repository, is too visible 
    - Has to be in the runner runtime environment somehow
    - Can do this in SPIN, though not very securely at the moment
    - Can do it on your laptops
    - Want to do it elsewhere? come for a chat
  - Exercise 03, in your own time ☺

https://gitlab.com/TonyWildish/gitlab-advanced/

Using metadata in containers

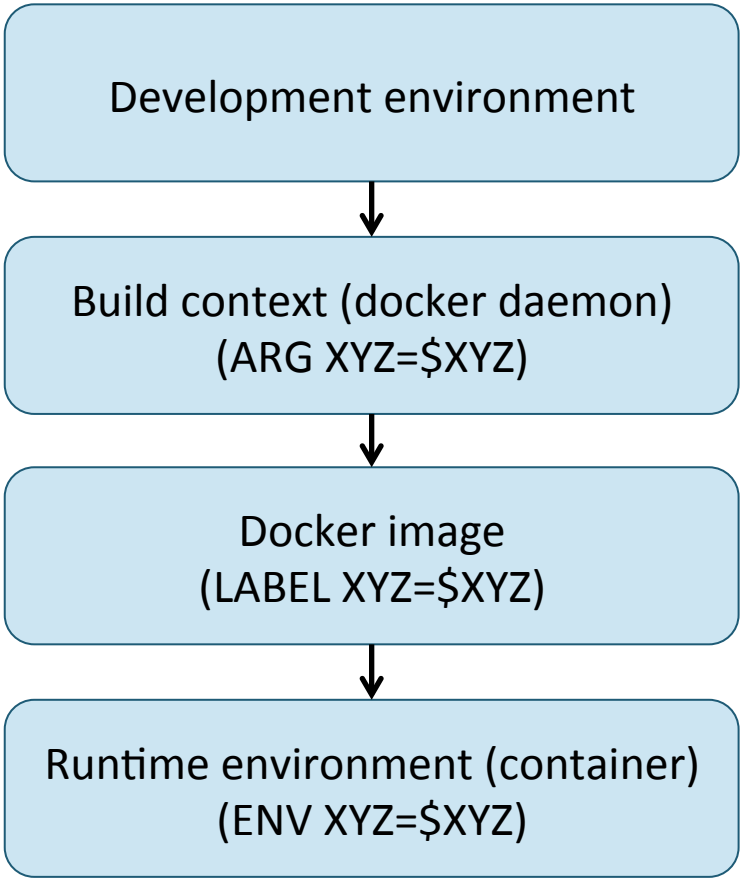
- **Pass information from the build environment**
  - To the image, or to the user at runtime

Thanks Michael, Alex

- **Tell the user anything they might want to know:**
  - What runtime environment the software needs
  - What level of testing, certification has been performed
  - Pointers to documentation, source code, maintainers…
  - Runtime details:
    - where the container looks for input
    - where it expects to be able to put output…

http://docs.master.dockerproject.org/v1.5/userguide/labels-custom-metadata/
https://speakerdeck.com/garethr/shipping-manifests-bill-of-lading-and-docker-metadata-and-container

Using metadata in containers



docker build … --build-args XYZ=123

How metadata goes from the build environment to the image, and to the running container

See **Dockerfile.metadata** in the repo

docker inspect … | grep XYZ

docker run… echo $XYZ

https://gitlab.com/TonyWildish/gitlab-advanced/

Using metadata in containers

- **How can we use metadata?**
  - E.g. defining a proper ontology
  - Automating pipelines, testing, discovery…
- **Working group(?) to investigate this**
  - Probably later in the year after the migration
  - Volunteers/suggestions gratefully accepted!
  - Exercise 04!

https://gitlab.com/TonyWildish/gitlab-advanced/

Deploying runners on NERSC hosts

- **A runner at NERSC with write-access to $HOME etc?**
- **You can do this, but there are serious risks involved!**

  - Don't share the runner registration token with anyone
    - ~= giving them your NERSC password
  - Don't give other users master-level access to your repository
  - Consider alternatives:
    - Use a Docker image, with your custom build environment, on SPIN
    - Use a VM somewhere…
  - Talk to a consultant before attempting this!
  - Some of these risks are gitlab-specific
  - Some are inherent in running any internet-enabled services

https://gitlab.com/TonyWildish/gitlab-advanced/

Deploying runners on NERSC hosts

- **Basic recipe**
  - Download the binary for a gitlab runner
  - Register it, give it a host-specific config file
  - Give it specific tags when registering, to identify it
  - Use those tags in your .gitlab-ci.yml file
  - Your pipeline can roam over the entire filesystem if you want, but it's up to you then to ensure the directories you use are clean
  - See exercise 05 for details – we won't do this today!

https://gitlab.com/TonyWildish/gitlab-advanced/

Other gitlab features

- **API, programmable interface to Gitlab**
  - https://docs.gitlab.com/ee/api/
    - See JGI/gitlab-cli-tools repo for some basic tools, contributions welcome!
- **Build hooks**
  - Trigger actions on external services other than gitlab
    - Similar capabilities on github, bitbucket
  - Trigger actions in gitlab from external service
    - E.g. nightly build, regardless of commits
- **Mirroring repositories**
  - Master repository in bitbucket/github?
  - Can mirror to gitlab, automatically, transparently
- **Issue-tracking, wiki**…
  - Other goodies come for free with gitlab, as with other hosting services

Best practices, recommendations

- **Git:**
  - Use the fork/pull-request model instead of granting people direct-commit access to your repository
  - Use branches to experiment, try out bugfixes etc
    - Merge long-lived branches frequently to control divergence
  - Use tags to identify stable versions, releases etc
  - Don't delete or move tags once they're pushed to the master repository

https://gitlab.com/TonyWildish/gitlab-advanced/

Best practices, recommendations

- **Gitlab:**
  - Build multiple Docker images if you have different use-cases to serve from the same code-base
  - Pushing to multiple registries lets users access your images from many places, easily
  - Use metadata in your containers!
    - Help us establish standards for JGI container metadata
  - Control access to your repositories
    - Don't give out the runner-registration token
    - Avoid giving others admin/developer-access to the project
    - Think twice before deploying runners on NERSC resources

https://gitlab.com/TonyWildish/gitlab-advanced/

Finally…

- **You're all experts now, so update your resumes!**
  - "experience building and optimizing Docker images for bioinformatic software"
  - "experience configuring and using continuous-integration platforms, such as gitlab, to automate building and deploying software"
  - "in-depth understanding of best-practices for software management, such as version control with git and use of metadata to describe Docker images"
  - "understanding of git workflow models for teams, including the use of branches, tags, and developer access-control"

https://gitlab.com/TonyWildish/gitlab-advanced/

National Energy Research Scientific Computing Center