Joint Institute for Nuclear Research

Software development workflow in BM@N: tools and features

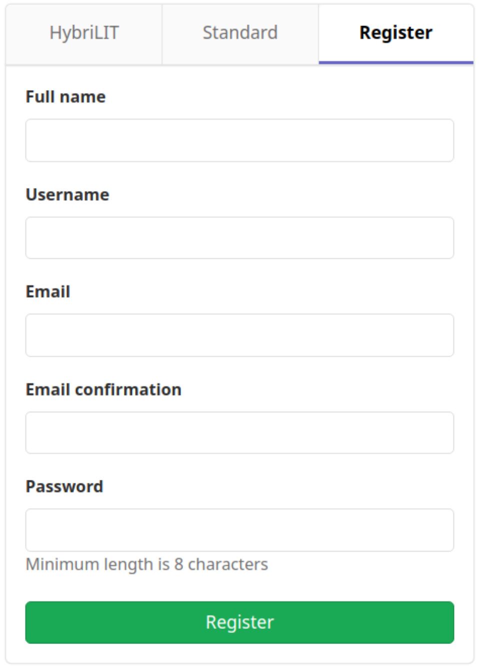
Nikita Balashov

GitLab Service at git.jinr.ru

- All of the most popular technologies for software development in one place
  - Version control system – Git
  - Continuous integration / continuous deployment – GitLab Runners
  - Issue tracker
  - Role-based access control to projects
  - Repository branch protection

Getting an Account

- If you have an email address in **jinr.ru** domain, you can register manually. Then ask the **project coordinator** to add you to the project.
- If you are an **“external”** user, request account from the coordinator. The account will be created for you.
- Use the **Standard** tab to login
- Additionally, if you have a JINR SSO account, you can link it to your git.jinr.ru profile (Profile Settings->Account)



Full name

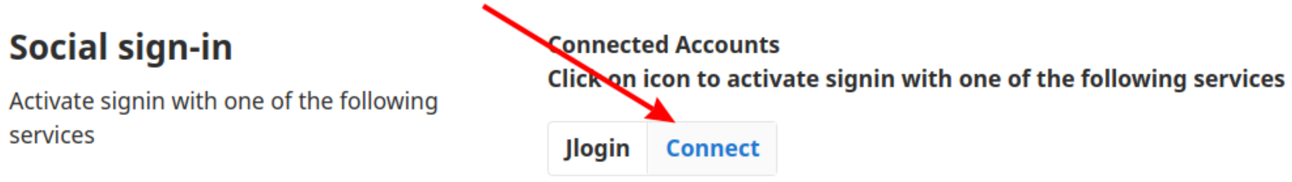
Username

Email

Email confirmation

Password

Minimum length is 8 characters


Social sign-in

Activate signin with one of the following services

Connected Accounts

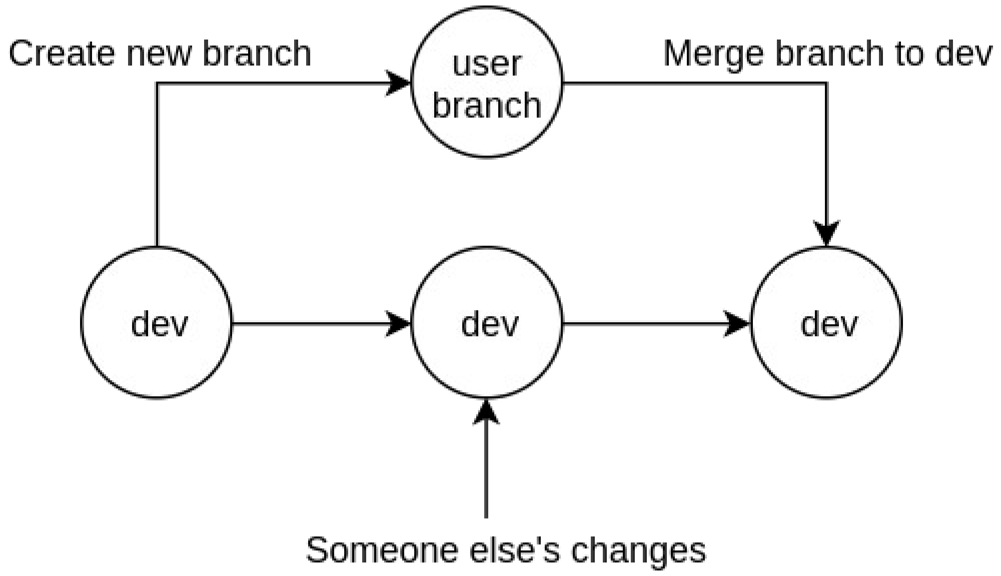
Click on icon to activate signin with one of the following services

Interacting with the Repository

- The project page is at https://git.jinr.ru/nica/bmnroot
- To access the git-repo over **https** use this link (you’ll need to provide the username/password): https://git.jinr.ru/nica/bmnroot.git
- Or add a public key to your GitLab profile settings to access the repo via **ssh**: git@git.jinr.ru:nica/bmnroot.git
- This is a standard git repo, so you can use any git tools
- Quick fixes can be done through the GitLab web-interface

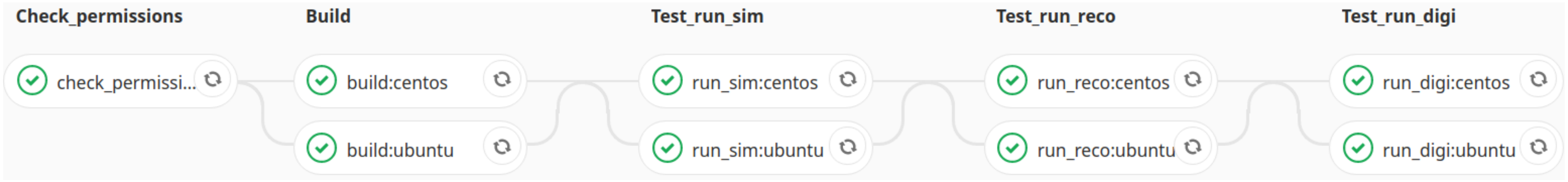
Branches and Basic Workflow

- There are two **protected** branches: **dev** and **pro**
- Only maintainers can push to protected branches directly
- Developers can only merge changes to **dev** branch from **other** branches
- Branches and merge requests prevent accidental overwrites of someone else’s changes

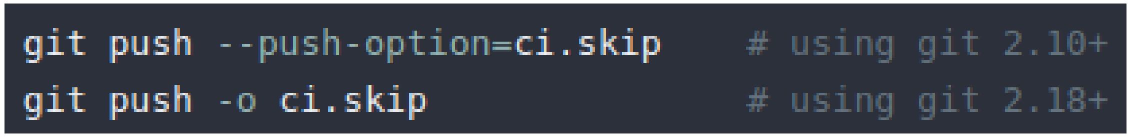


Automated Tests

- Two dedicated GitLab runners (Ubuntu and CentOS 7): 4 CPUs, 16 GB RAM **each**
- Runners are cloud virtual machines co-shared with other NICA projects (mpdroot, nicafemto)
- All the tests are defined in plain text file **.gitlab-ci.yml** at the root of the repo
- **Failed pipelines prevent** your changes from being merged into **dev** branch
- Tests should run on merge requests only



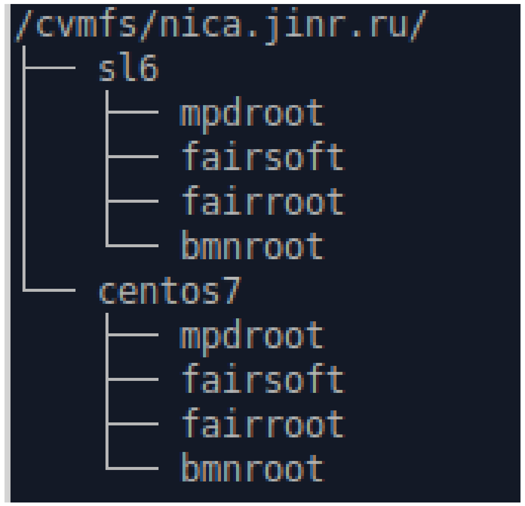
- If you want to skip the pipeline, add either add one of [skip ci]/[ci skip] to the commit message, or pass ci.skip push option



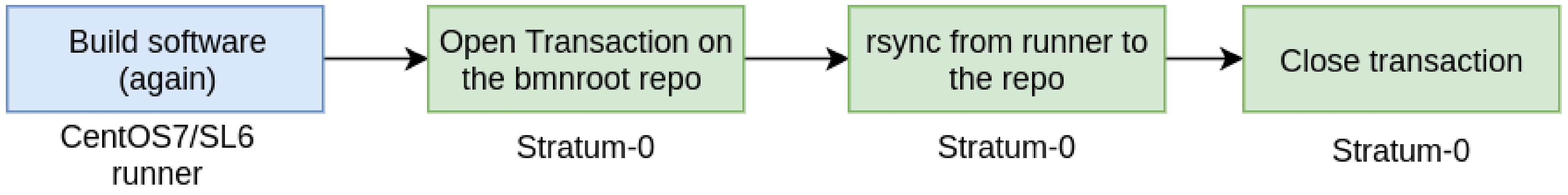
```
git push --push-option=ci.skip    # using git 2.10+
git push -o ci.skip               # using git 2.18+
```

Automated Deployment

- Two dedicated GitLab runners (CentOS 7 and SL 6): 4 CPUs, 16 GB RAM **each**
- Additional “**Deploy”** stage in the pipeline for the **dev** and **pro** branches only
- Defined in **.gitlab-ci.yml**, same as tests
- Software is stored in cvmfs, mounted on T1/T2 and potentially can be mounted on any computer in JINR network



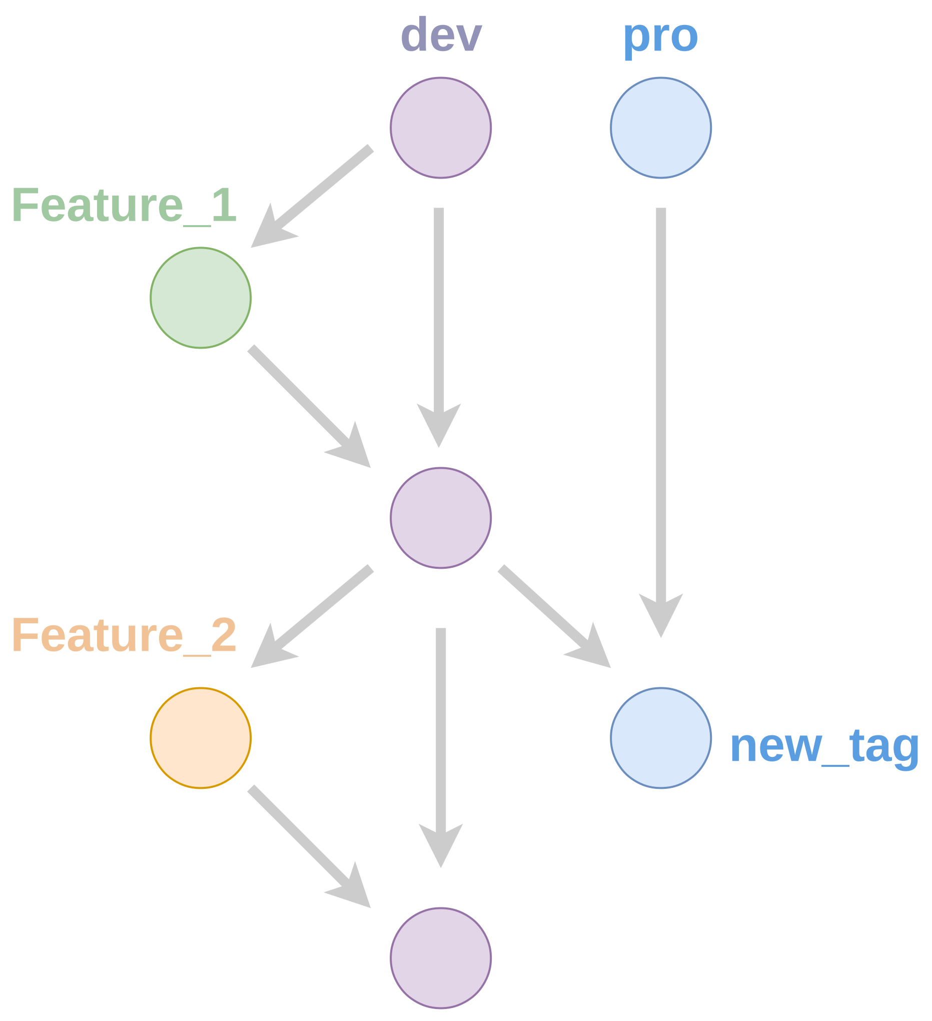
```
/cvmfs/nica.jinr.ru/
├── sl6
│   ├── mpdroot
│   ├── fairsoft
│   ├── fairroot
│   └── bmnroot
└── centos7
    ├── mpdroot
    ├── fairsoft
    ├── fairroot
    └── bmnroot
```



- Sometimes deploy jobs fail due to cvmfs is limited to only one open transaction at a time and we run deployment jobs in parallel
- Failed jobs restarted manually

Complete Workflow

- Don’t underestimate the **issue tracker**
- Start an issue
- Create a merge request right from the issue page, a corresponding git branch will be created also
- Pull the changes to your local repo copy and check out your feature branch
- Commit locally and push when you are ready to share the changes
- If automated tests fail – fix your code
- Merge the changes, close the issue
- When there’s enough changes to produce a new release, the **release manager** merges the **dev** branch into **pro** branch and gives it a new version tag



Future changes

- We could benefit parallel transactions from CVMFS **Gateway** and **Publisher** technologies (requires significant changes to the CVMFS infrastructure)
- Move tests into docker containers
  - Prepare a set of containers with all the required environments
  - Recreate and unify the tests runners
- It’s not quite clear how to run deploy jobs in docker
  - The deploy container needs to be accessible from Stratum-0 over ssh
- If test and deploy containers work out well on dedicated runners, we may try to get use of **generic** docker runners
- Release docker containers (anybody needs them?)

Thanks!