Gitlab CI/CD

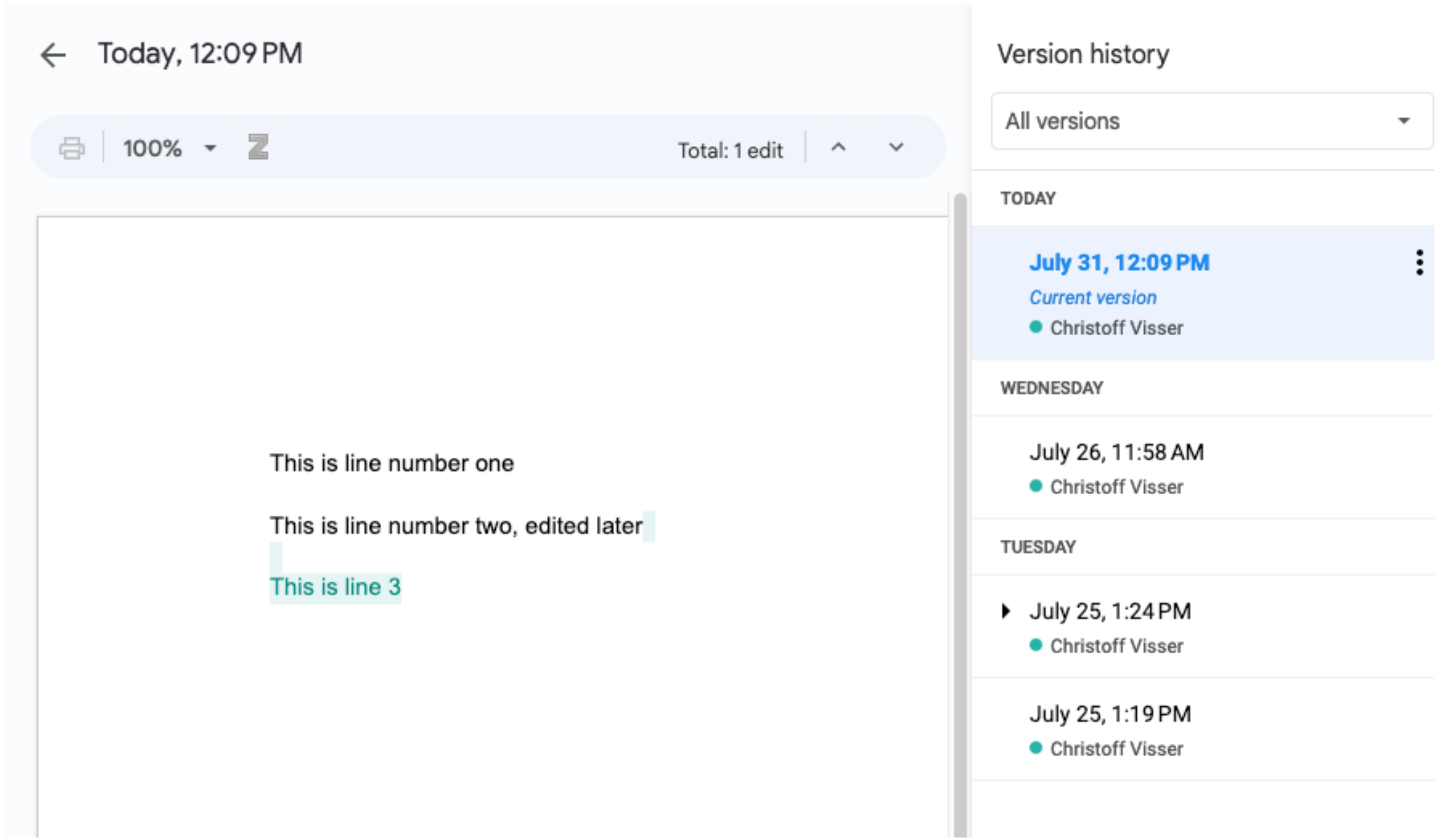
Christoff Visser

christoff@iij.ad.jp

Internet Initiative Japan

Introducing git

Introducing git



This is line number one

This is line number two, edited later

This is line 3

July 31, 12:09 PM
Current version
Christoff Visser

July 26, 11:58 AM
Christoff Visser

July 25, 1:24 PM
Christoff Visser

July 25, 1:19 PM
Christoff Visser

Distributed Workflows

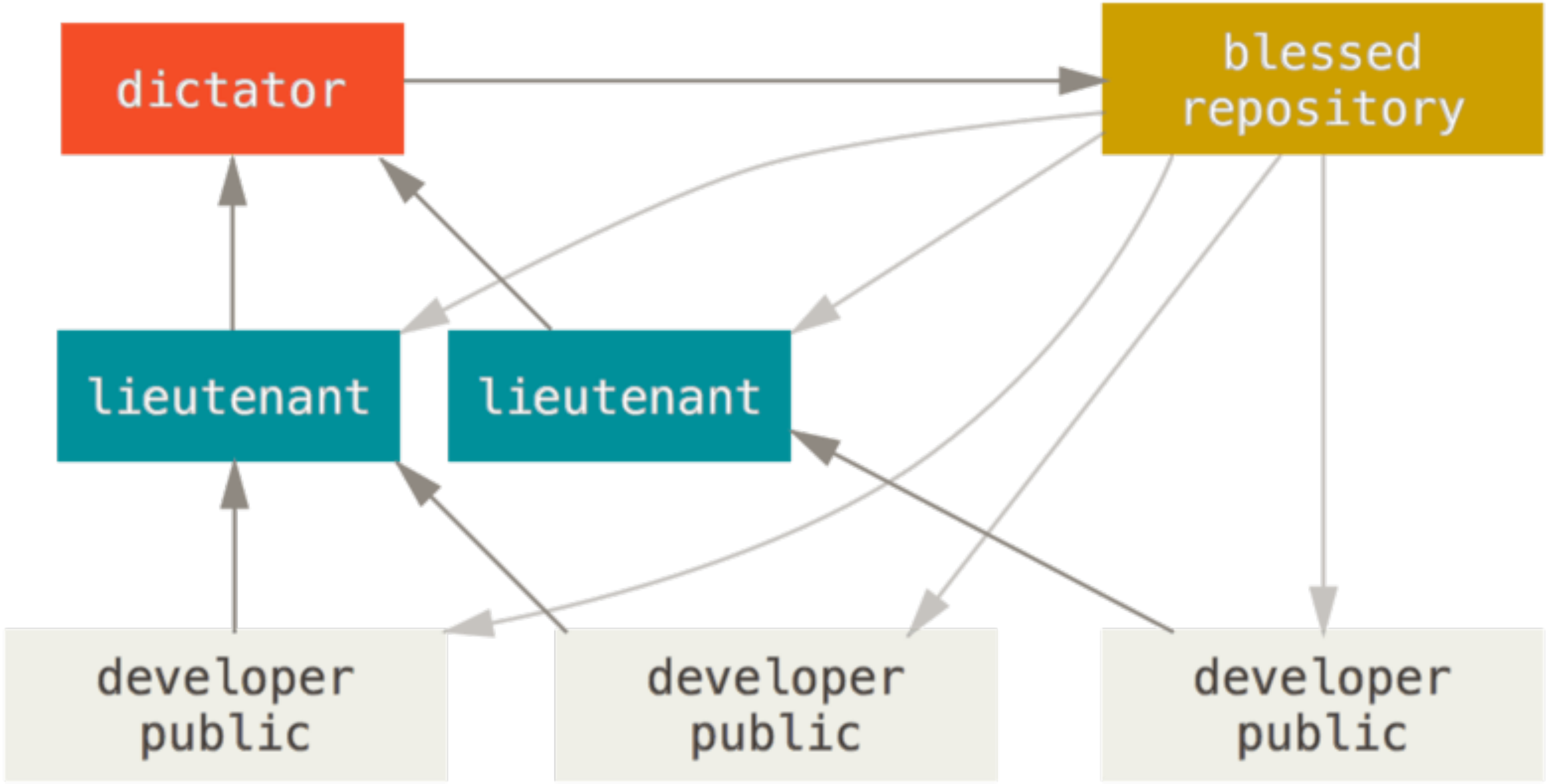


Image source: https://www.git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows

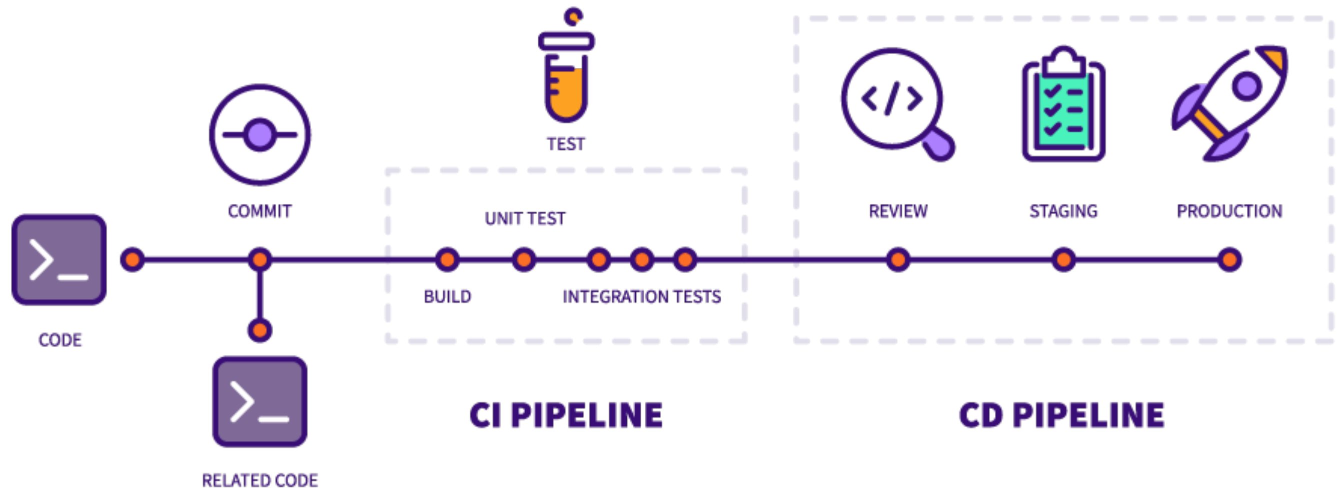
Where to store the repository?

- GitHub, Gitlab, Bitbucket, AWS CodeCommit

- Gitlab
  - Self-host your own instance
  - Some more freedom with the CI/CD

What is CI/CD

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Continuous Deployment (CD)



Where does the code run

- Code typically runs inside a docker container as a job
- One can use available popular containers like alpine, ubuntu, centos, or application specific containers
- One can also build own container using base image of any of the available containers if installing multiple packages

And where does the job run?

- Typically a runner - can be shared runner offered by popular hosted Git providers like Gitlab, Github etc and also dedicated runners which you can host on your machine (desktop/server)

- Runner can be a program installed & running on machine or simply a docker image with special permissions

- One can have multiple runners configured in a project & use them as needed across various tasks. E.g task 1 on runner on server1, task 2 on runner on server2 etc

- Good idea to have basic understanding of docker ecosystem to make efficient use of CI/CD

Key Objective

- Make use of extremely low code, fast to deploy tool like Ansible to automate or semi-automate repetitive tasks
- Trigger Ansible as a docker container running Ansible on runner of your choice
- Trigger (Ansible + Docker) via CI/CD pipelines

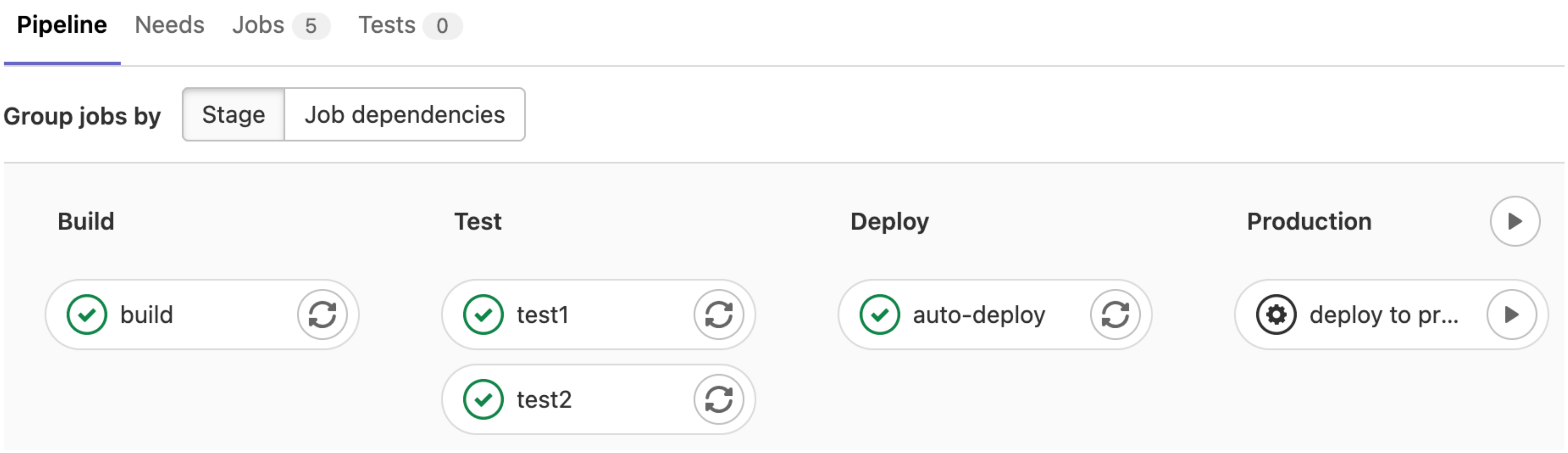
Stage & Jobs

- Config is divided in stages
- Each stage can have one or more jobs which run in parallel (by default)
- Stages run sequentially
- Any job can have dependency on any other job if needed

Jobs

- Each action is run as a job
- A job runs inside a docker container
- Job can have any script (bash, python etc) or Ansible Playbook or anything that is packed in container
- Job can have dependency on any other job:
  - run job 2 only when job 1 is finished
  - run job 3 only when job 1 has failed
- Jobs can be triggered automatically upon commit, via web UI, via scheduler & via API call
- Multiple jobs together is a pipeline

Typical design of pipeline



- Build containers
- Compile code

- Deploy containers
- Deploy code
- Test code in containers

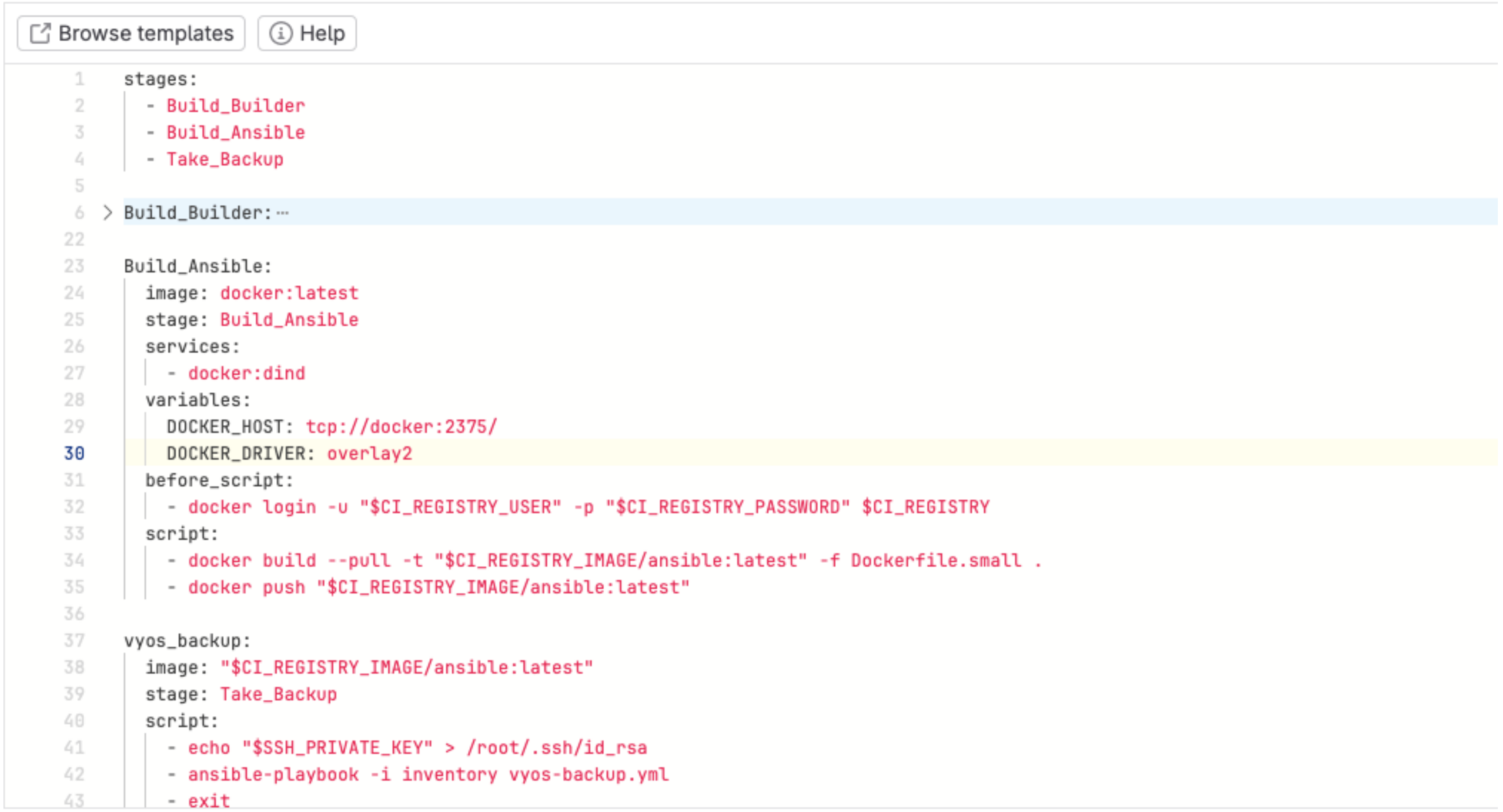
- Deploy to production

- Interact with production system
- e.g., Revert to previous state

Gitlab Artifacts

- (Reminder!) Containers by design are stateless.
  - State must be stored outside
- Store the output data from the job (if need to)
- Enable sharing of files between jobs
- Can be stored within Gitlab, S3 endpoint or a self hosted storage instance

Sample .gitlab-ci.yml



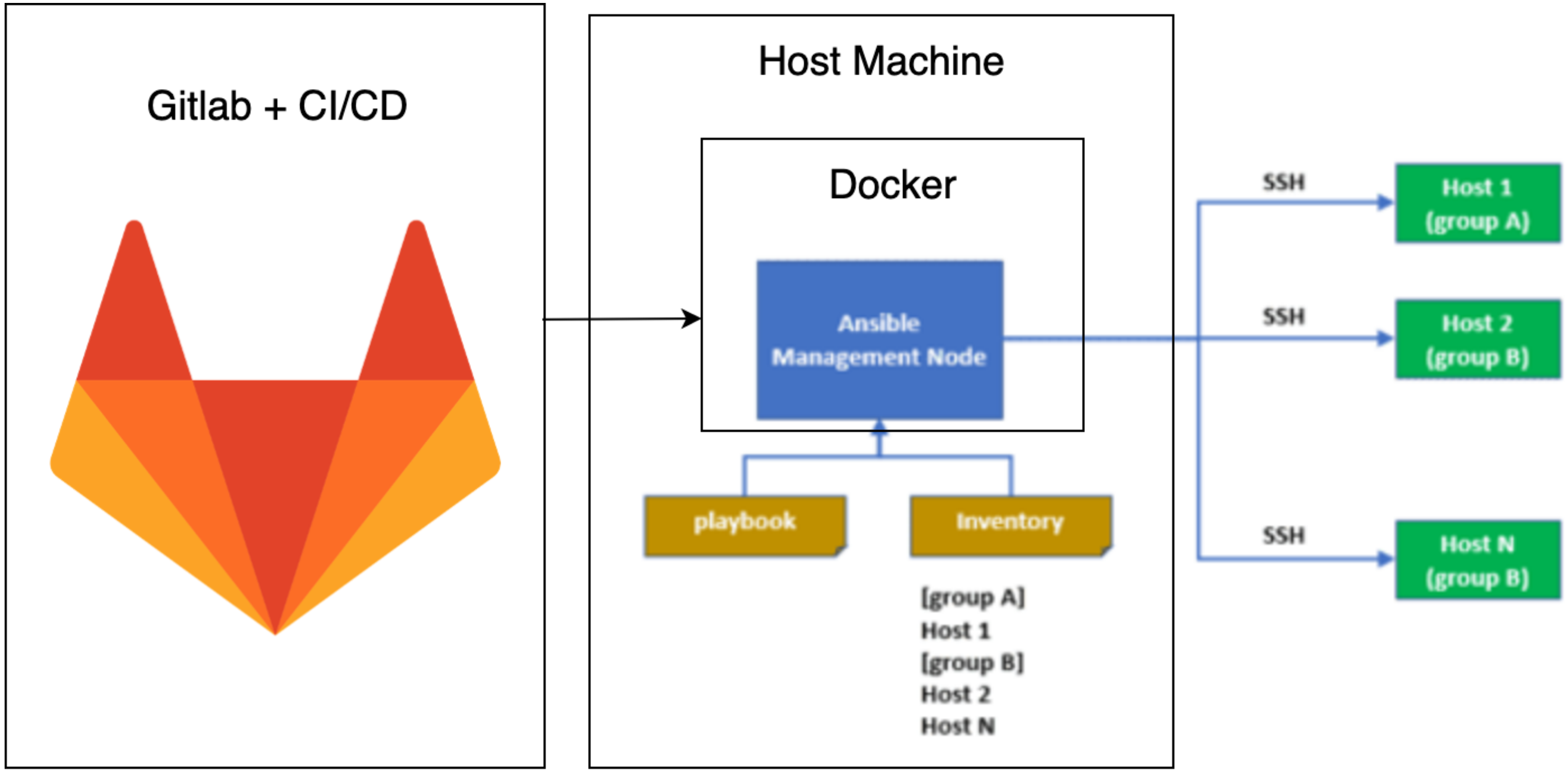
```
stages:
  - Build_Builder
  - Build_Ansible
  - Take_Backup

Build_Builder:···

Build_Ansible:
  image: docker:latest
  stage: Build_Ansible
  services:
    - docker:dind
  variables:
    DOCKER_HOST: tcp://docker:2375/
    DOCKER_DRIVER: overlay2
  before_script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE/ansible:latest" -f Dockerfile.small .
    - docker push "$CI_REGISTRY_IMAGE/ansible:latest"

vyos_backup:
  image: "$CI_REGISTRY_IMAGE/ansible:latest"
  stage: Take_Backup
  script:
    - echo "$SSH_PRIVATE_KEY" > /root/.ssh/id_rsa
    - ansible-playbook -i inventory vyos-backup.yml
    - exit
```

Final Workflow result



Questions?

christoff@iij.ad.jp

Internet Initiative Japan