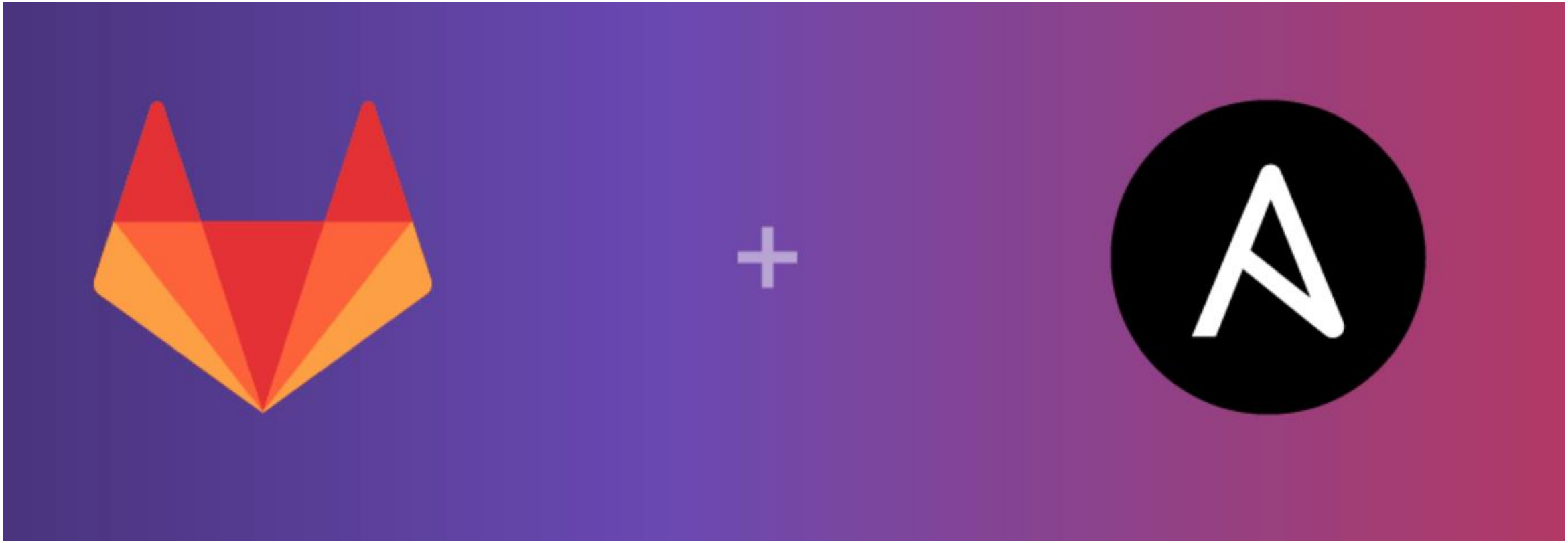
CI/CD utilizando GitLab CI e Ansible



Whoami?

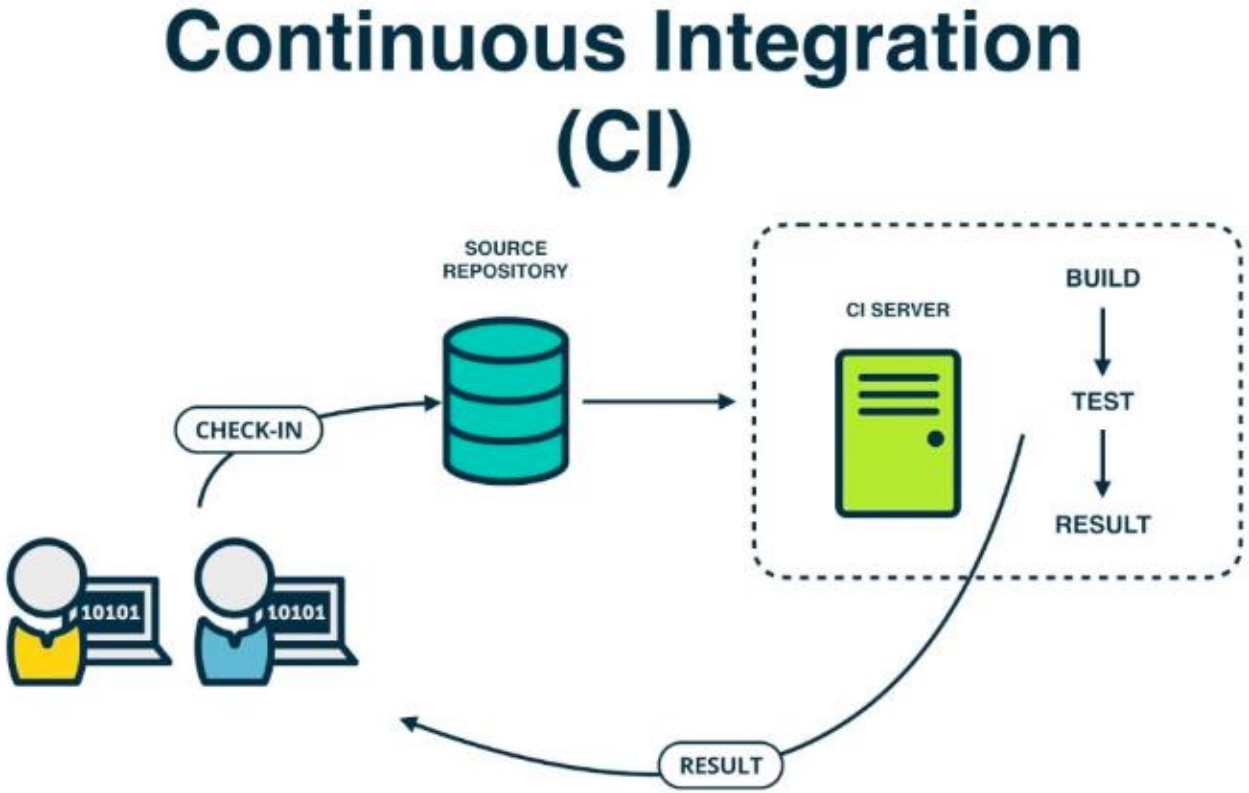
- Analista de suporte de TI – Assert
- Tecnólogo em Redes de Computadores - IFPB
- HCIA – R&S e Cloud
- NSE1 | NSE2
- DEPC | Fundation

Agenda

- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- O que é gitlab ?
- Gitlab CI, o que é ?
- Pipeline, Stages e Steps
- Runners
- CI as Code
- Gerência de Configuração
- Iac
- Ansible
- Estrutura Ansible
- Módulos
- Ad-hoc
- Playbooks
- Roles e Tags
- Tasks
- Mão na massa !!!!
  - Provisionando Infrastrutura de deploy
  - Meu primeiro pipeline
  - Criando steps e gerando artefatos
  - Meu primeiro playbook
  - Fazendo deploy on primsse
  - Fazendo deploy com docker
  - Monitorando a saude do meu código

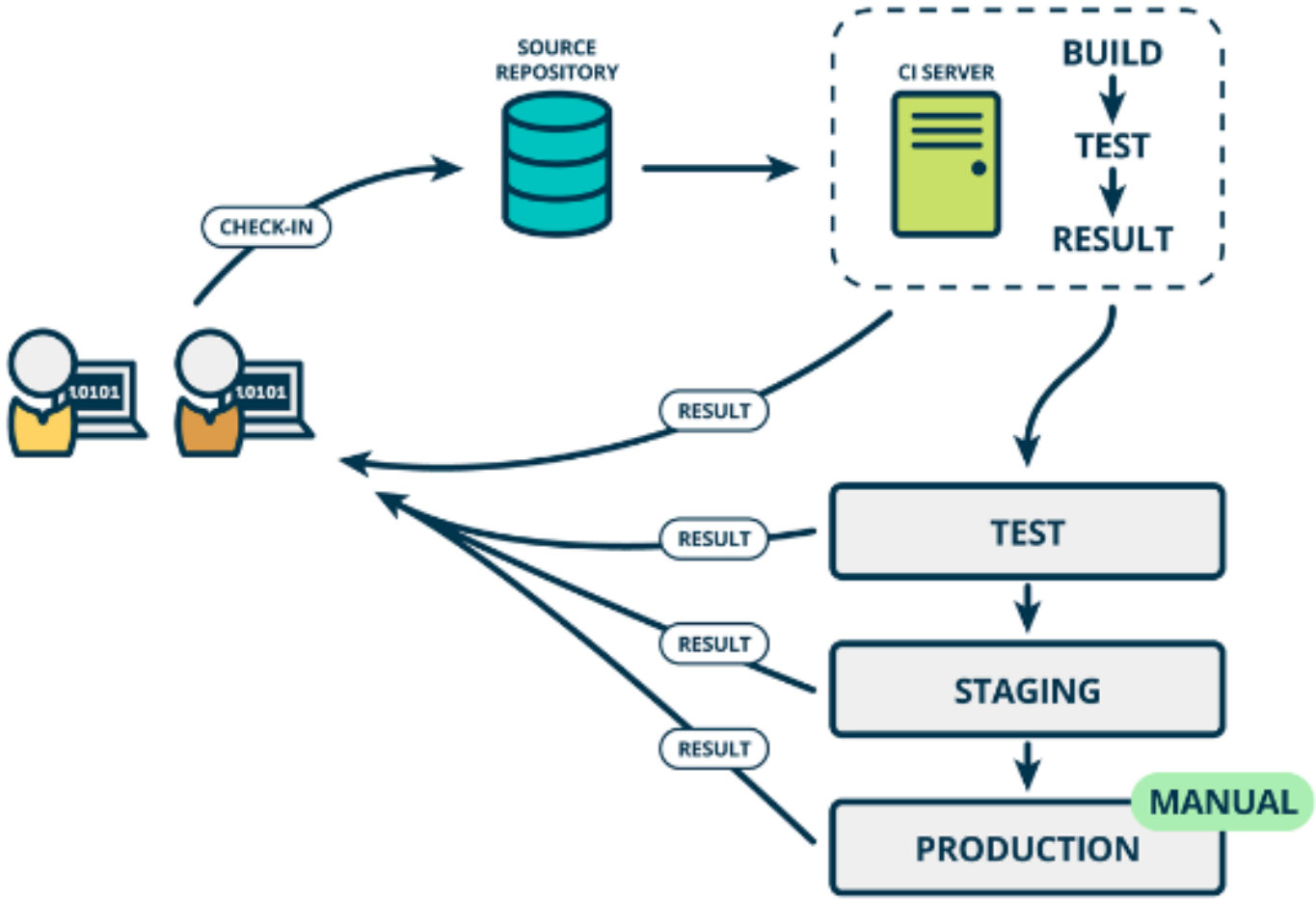
Continuous Integration

- Integração do código visando validar e testar modificações continuamente
- Eliminar bugs
- Mensurar qualidade e saúde do código



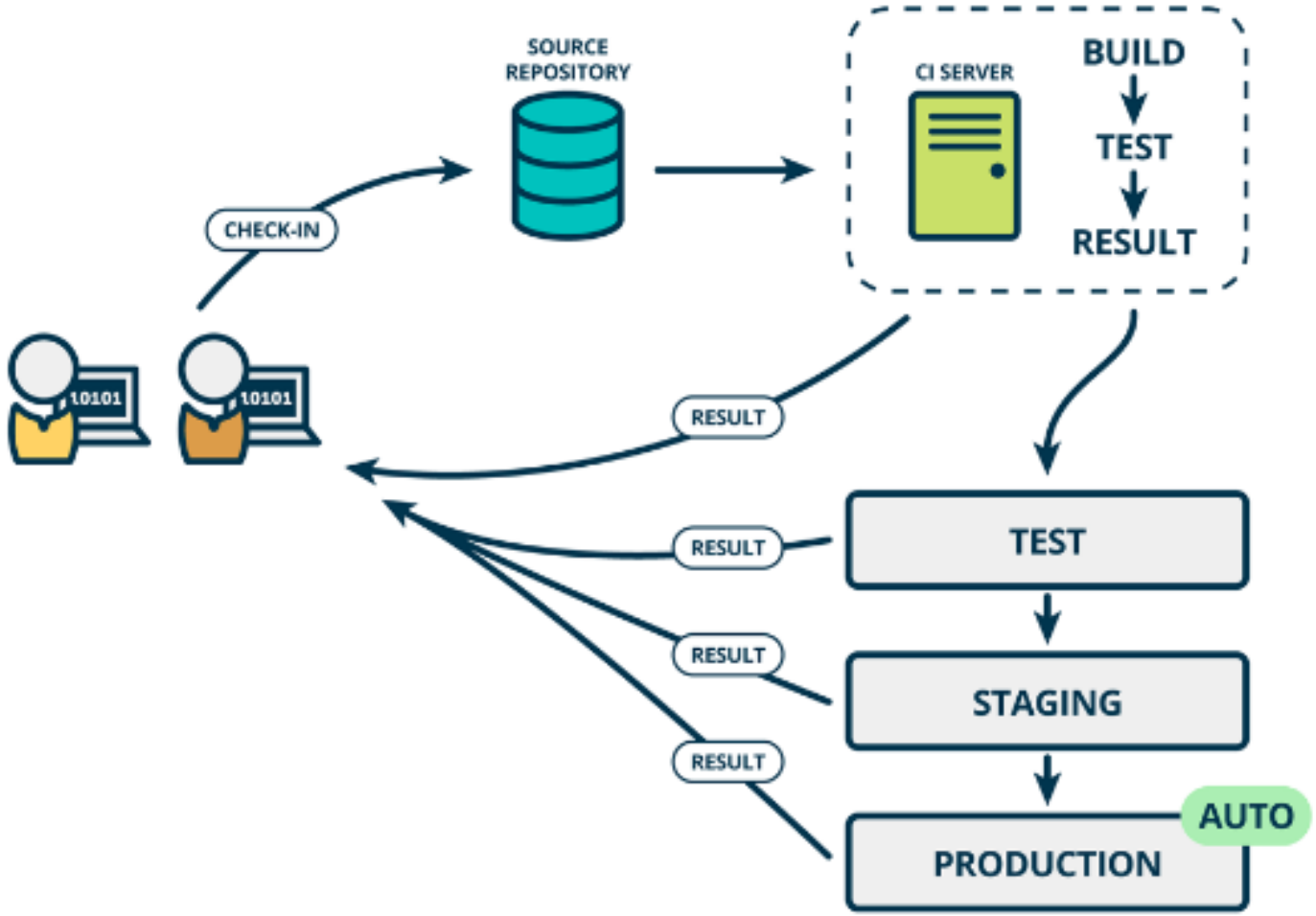
Continuous Delivery

- Disponibilizar uma versão da aplicação para realização de teste QA.
- Nesta fase feedbacks são fundamentais como rege a prática DevOps. Como na fase anterior, no Continuous Delivery a ideia é encontrar o percalços/bugs o mais cedo possível antes da aplicação seguir para o estágio final.



Continuous Deployment

- Continuous Deployment tem o objetivo de prover um deploy/implantação do projeto para ambiente de produção de forma fácil e automatizada (sem intervenção humana). Mas para que isso aconteça com sucesso é necessária uma ampla cobertura de testes na(s) aplicação(s) feita na etapa anterior, Continuous Delivery.



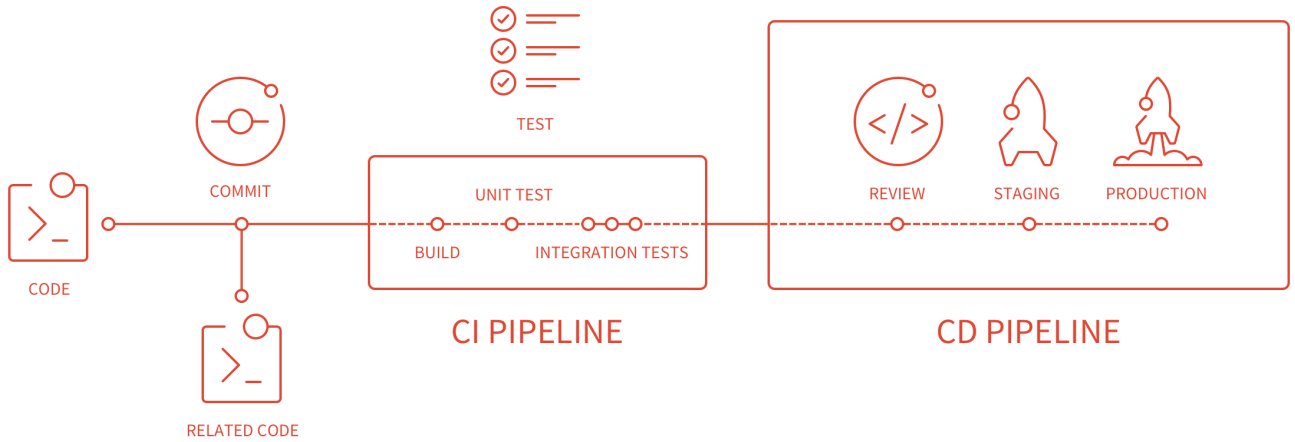
O que é gitlab ?

- Gerenciador de repositório de software baseado em git
- Open source

- Wiki, gerenciamento de tarefas, CI/CD, Container Registry

Gitlab CI, o que é ?

- Ferramenta Para criação de Pipelines completos de CI/CD
- Totalmente Integrado ao Gitlab
- Facil de usar
- Altamente escalavel e configuravel
- Ambiente de testes isolados



Pipeline, Stages e Steps/jobs

Steps/jobs

Etapa de construção, tarefa de compilação, testes e integrações

Stages

Conjunto de jobs que possuem relação e devem ser executados no mesmo passo

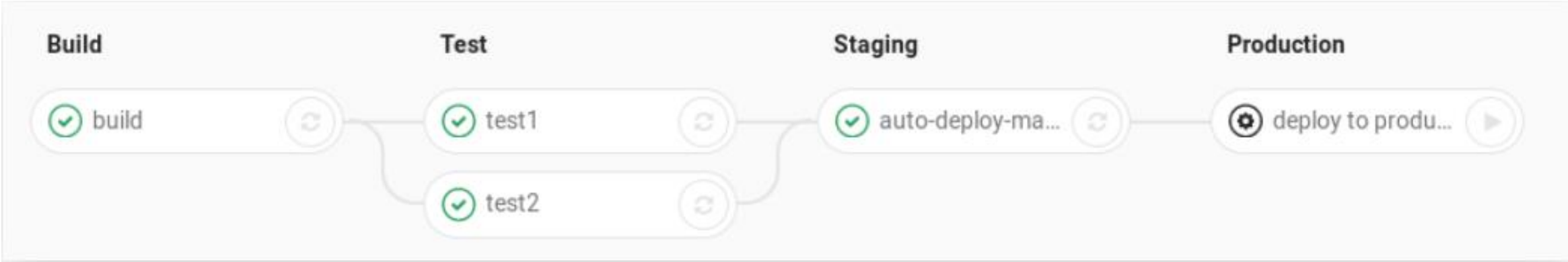
Pipelines

Orquestrando e reunindo todos os stages na ordem correta

Pipeline, Stages e Steps/jobs

Configuração do pipeline **.gitlab-ci.yml**

- Definição de tarefas
- imagens
- Variáveis
- Scripts
- branches



Runners

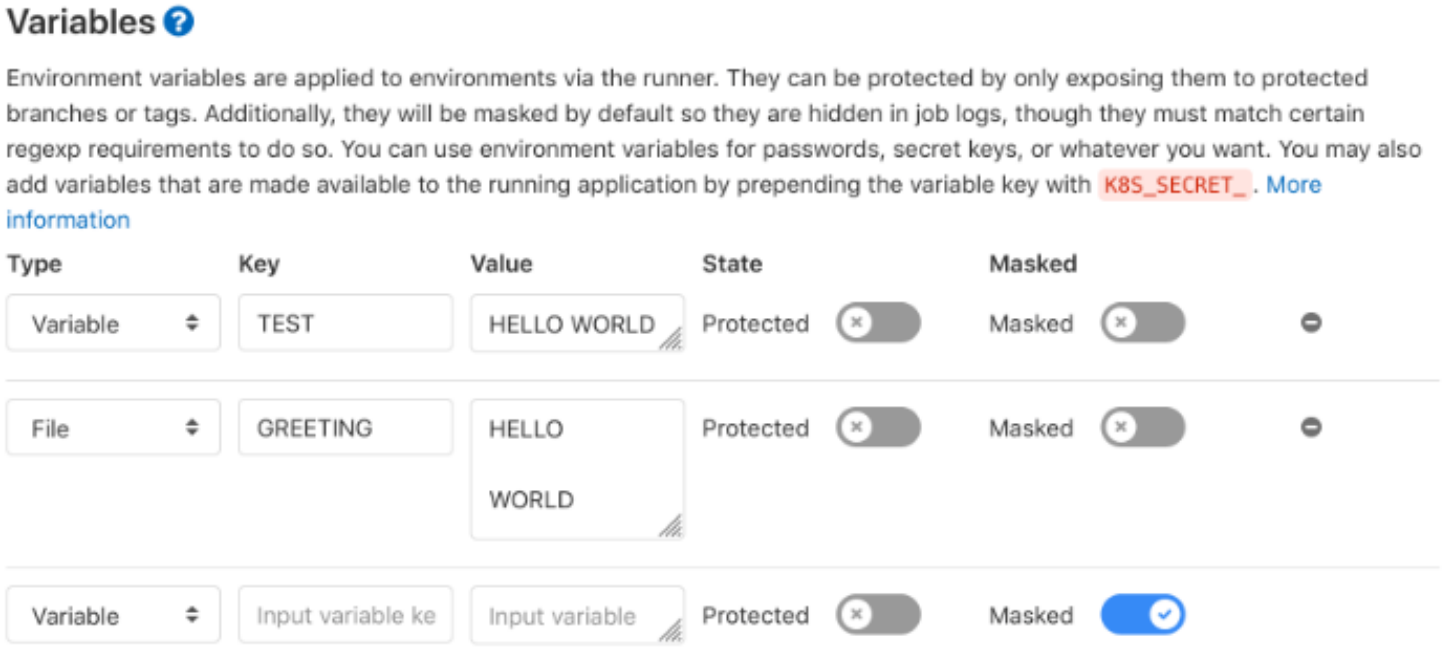
- Responsavel por processar os jobs
- Linux, Windows e OSX
- Docker
- Shared e Private
- Diferentes executores:
  - Docker
  - Shell
  - SSH
  - Virtualbox
  - Kubernetes

Runners

Executor	SSH	Shell	VirtualBox	Parallels	Docker	Kubernetes	Custom
Clean build environment for every build	✗	✗	✓	✓	✓	✓	conditional (4)
Reuse previous clone if it exists	✓	✓	✗	✗	✓	✗	conditional (4)
Migrate runner machine	✗	✗	partial	partial	✓	✓	✓
Zero-configuration support for concurrent builds	✗	✗ (1)	✓	✓	✓	✓	conditional (4)
Complicated build environments	✗	✗ (2)	✓ (3)	✓ (3)	✓	✓	✓
Debugging build problems	easy	easy	hard	hard	medium	medium	medium

Variáveis

- Nativas/predefinidas
  https://docs.gitlab.com/ce/ci/variables/predefined_variables.html
- Pipeline
- Privadas
  - Usuário
  - Grupo



CI as Code

.gitlab.ci.yml

```yaml
default:
  before_script:
    - ls

before_script:
  - ls
after_script:
  - ls

image: node:latest

stages:
  - test
  - sonar
  - build
  - deploy-dev

test-unitario-front:stable:
  stage: test
  script:
    - cd ./production/api
    - npm run test-unit
```

```yaml
test-api:stable:
  stage: test
  script:
    - cd ./stage/api
    - npm run api-test

sonarqube:publish:
  stage: sonar
  image: ciricihq/gitlab-sonar-scanner
  variables:
    SONAR_URL: "$SONAR_URL_PUB"
    SONAR_ANALYSIS_MODE: publish
  script:
    - gitlab-sonar-scanner -X
```

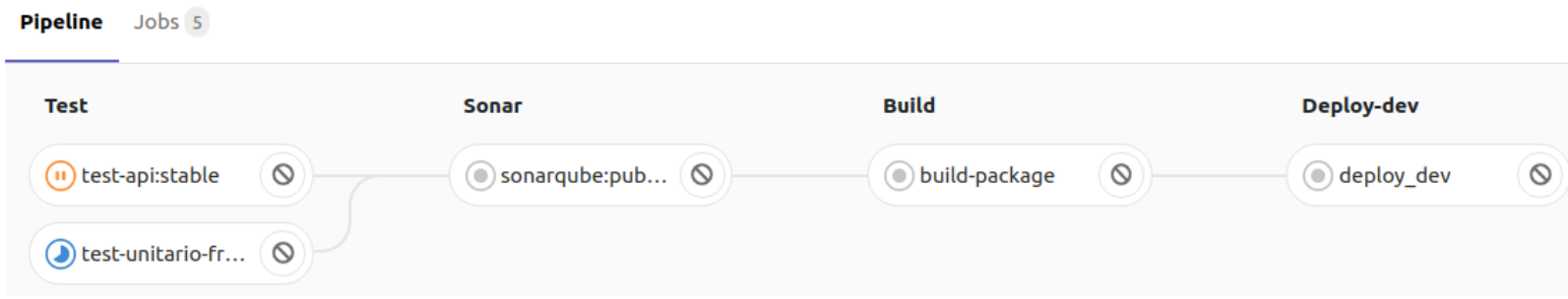
```yaml
build-package:
  stage: build
  script:
    - npm run build
  artifacts:
    paths:
    - dist/

build-image:
  stage: build
  script:
    - cd stage/web
    - docker ps
    - docker build -t testebuild .
  tags:
    - builddocker
```

```yaml
deploy_dev:
  stage: deploy-dev
  image: edsonlvalmeida/ansible:1.0
  script:
    - echo "Deploy to staging server"
    - echo "$private_key" > key && chmod 400 key
    - eval "$(ssh-agent -s)"
    - ssh-add key
    - export ANSIBLE_HOST_KEY_CHECKING=False
    - ansible-playbook -i hosts -u edson playbook.yml --tags "dev"
  environment:
    name: deploy-dev
    url: "$ENV_PROD"
  when: manual
  only:
  - master
```

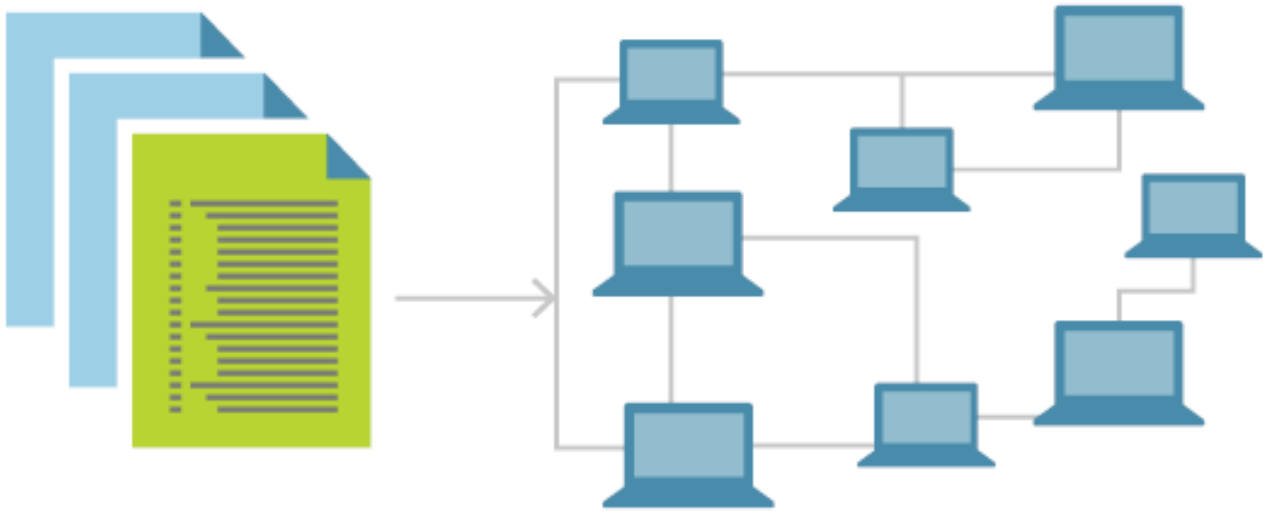
CI as Code

- .gitlab.ci.yml



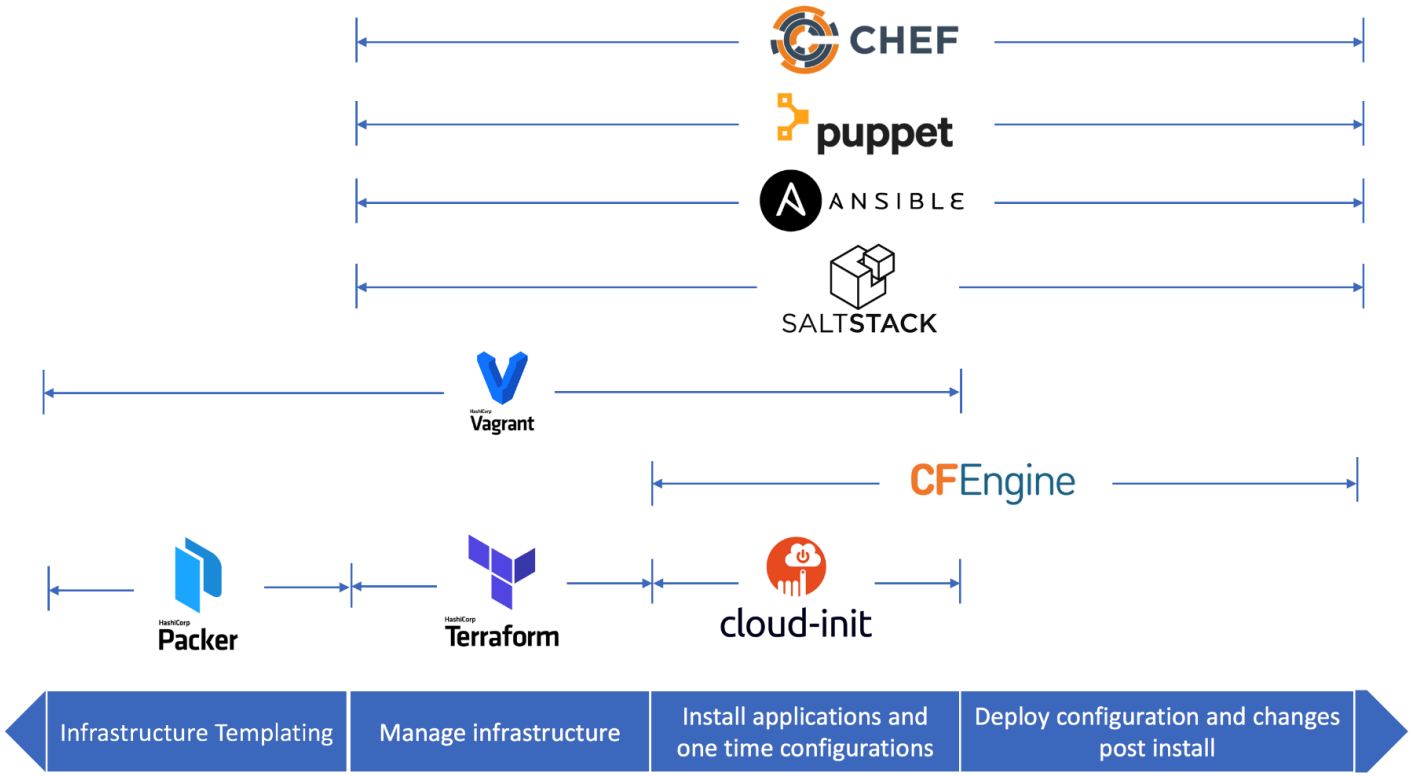
IaC

- Processo de gerenciamento e provisionamento por meio de arquivos
- Padronização em configurações e implantações
- Grande poder de escalabilidade
- Agilidade na implantação e no troubleshoot de problemas



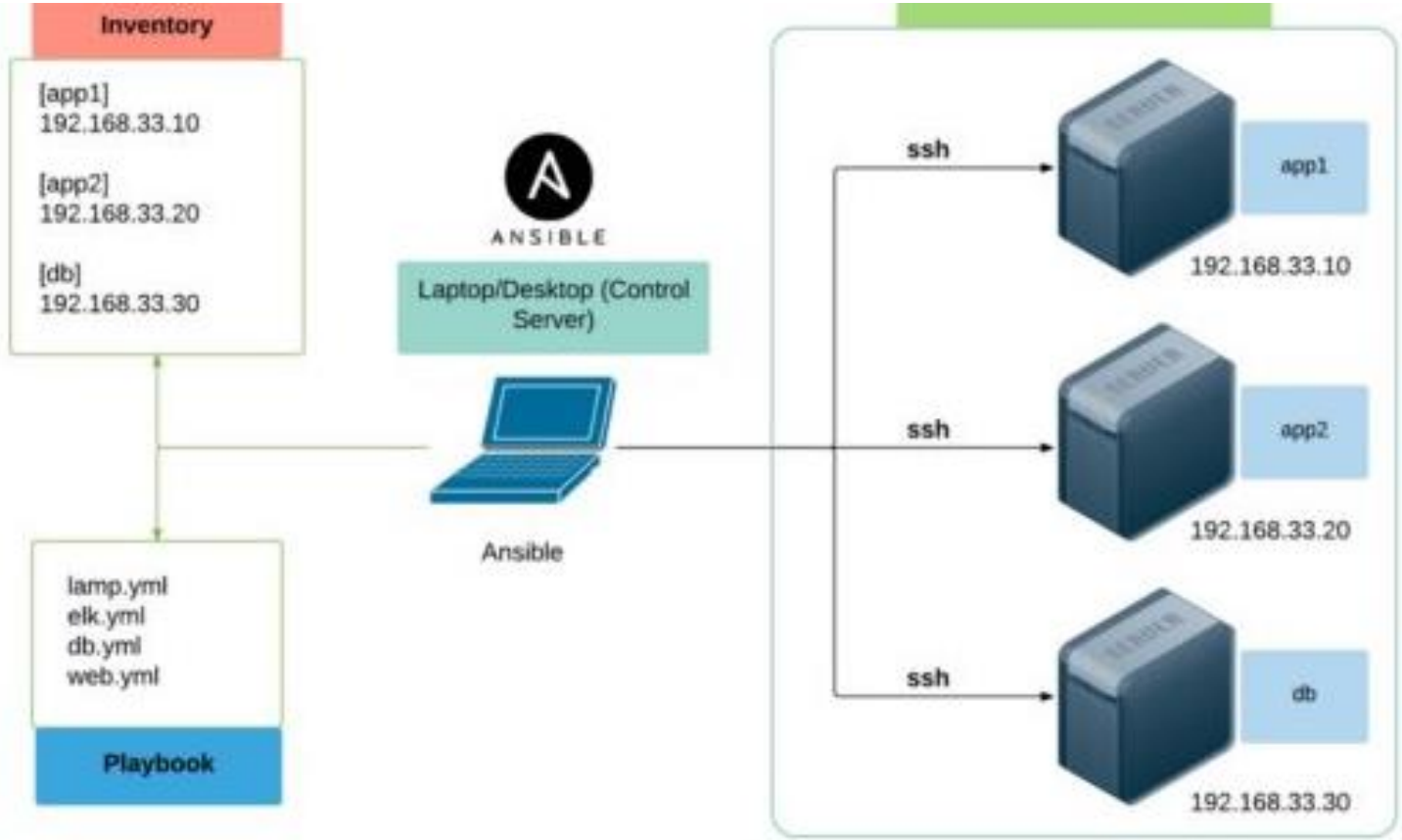
Gerência de Configuração

- “um conjunto de atividades de apoio que permitem controlar as mudanças que ocorrem no desenvolvimento de software, mantendo a estabilidade na evolução do projeto.”
  - Controle de Versão
  - Agilidade no provisionamento



Ansible

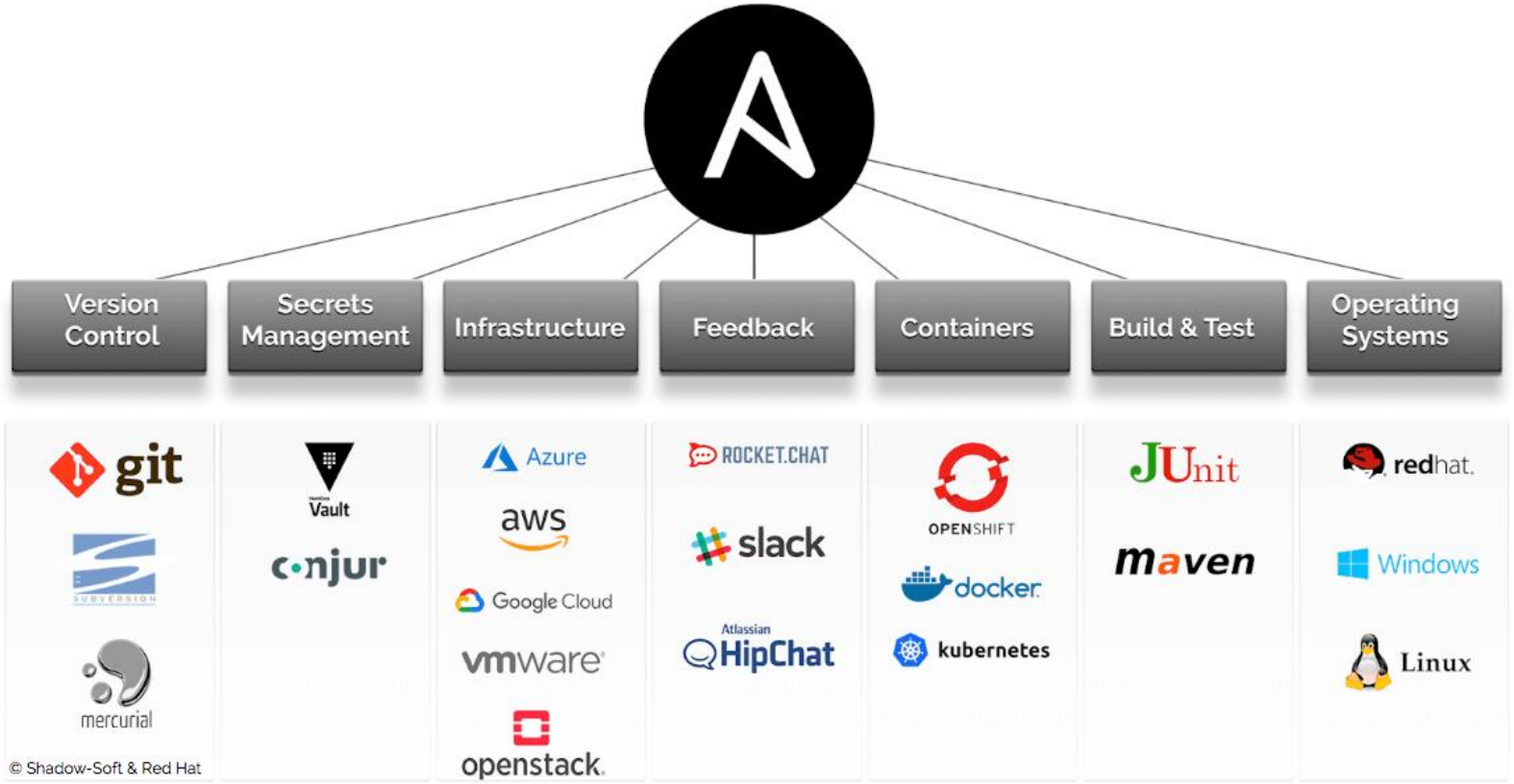
- "Ansible é uma ferramenta de automação de TI. Ele pode configurar sistemas, implantar software e orquestrar tarefas de TI mais avançadas, como implantações contínuas ou atualizações sem interrupção."
- Simples e facil de usar
- SSH e WinRM
- Inventario
- Playbook



Ansible

- Módulos
- Controlar recursos do sistema
  - Serviços
  - Pacotes
  - Arquivos
  - Códigos
  - Configurações
  - Usuários

https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html



Ad-hoc

- Comandos individuais
- Ideais para tarefas esporádicas

ansible –i ‘file’ ‘group’ -m **[**module**]** -a "[module options]“

ansible –i ‘hosts’ all -m ping –u “user” –k

```
root@ubuntu18:/home/assert# ansible -i 'hosts' web -m ping -u vagrant -k
SSH password:
10.0.60.54 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

facts

- Variáveis referentes ao sistema
- Pode usar em condicionais para o sistema
- Obter informações do host ad-hoc

ansible –i ‘hosts’ all -m setup –u “user” –k

```
root@ubuntu18:/home/assert# ansible -i 'hosts' web -m setup -u vagrant -a 'filter=ansible_os_family'

10.0.60.54 | SUCCESS => {
    "ansible_facts": {
        "ansible_os_family": "Debian",
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false
}
root@ubuntu18:/home/assert# ansible -i 'hosts' web -m setup -u vagrant -a 'filter=ansible_processor'

10.0.60.54 | SUCCESS => {
    "ansible_facts": {
        "ansible_processor": [
            "0",
            "GenuineIntel",
            "Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz",
            "1",
            "GenuineIntel",
            "Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz"
        ],
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false
}
root@ubuntu18:/home/assert#
```

tasks

- Executas na ordem da declaração
- Realiza uma ação no host via modulo
- Pode gerar novas variáveis a partir de resultados dos modulos

```
tasks:
 - name: make sure apache is running
   service:
     name: httpd
     state: started
```

tasks

```
TASK [docker : aplicando permissoes no compose  Compose (caso n?o esteja instalado)] ***
skipping: [10.0.60.159]

TASK [docker : link compose (caso noo esteja instalado)] **********************
ok: [10.0.60.159]

TASK [docker : verificar pip] *************************************************
changed: [10.0.60.159]

TASK [docker : Instalando pip] ************************************************
ok: [10.0.60.159]

TASK [docker : instalando pip-docker] *****************************************
changed: [10.0.60.159]

TASK [docker : verifica service] **********************************************
changed: [10.0.60.159]

TASK [docker : removendo service atual] ***************************************
changed: [10.0.60.159]

TASK [docker : enable environment] ********************************************
changed: [10.0.60.159]

TASK [docker : exec seeds] ****************************************************
changed: [10.0.60.159]

PLAY [Deploy test] ************************************************************

PLAY [Deploy production] ******************************************************

PLAY RECAP ********************************************************************
10.0.60.159                : ok=18   changed=11   unreachable=0    failed=0

Job succeeded
```

Roles e Tags

Se você tem um manual grande, pode ser útil poder executar apenas uma parte específica dele, em vez de executar *tudo* no manual. O Ansible suporta um atributo "tags:" por esse motivo.

- ansible-playbook example.yml --tags "configuration,packages“
- ansible-playbook example.yml --tags "packages“

```
tasks:
- yum:
    name:
    - httpd
    - memcached
    state: present
  tags:
  - packages

- template:
    src: templates/src.j2
    dest: /etc/foo.conf
  tags:
  - configuration
```

Roles e Tags

- Roles são maneiras de carregar automaticamente determinados arquivos, vars, tarefas e manipuladores com base em uma estrutura de arquivo conhecida. O agrupamento de conteúdo por funções também permite o fácil compartilhamento de funções com outros usuário

```
site.yml
webservers.yml
fooservers.yml
roles/
  common/
    tasks/
    handlers/
    files/
    templates/
    vars/
    defaults/
    meta/
  webservers/
    tasks/
    defaults/
    meta/
```

```
- hosts: webservers
  roles:
    - common
    - webservers

- hosts: fooservers
  roles:
    - common
    - fooservers
```

Playbooks

- Conjunto de tudo que foi visto + muitas outras coisas;
- Definição de variáveis a nível de roles e tags;
- Os playbooks são a base para um sistema de gerenciamento de configuração e implantação de várias máquinas;
- Os Playbooks podem declarar configurações, mas também podem orquestrar etapas de qualquer processo;
- A definição, escrita, organização e reutilização de playbooks pode seguir sua necessidade

Estrutura Ansible

- tasks - contém a lista principal de tarefas a serem executadas pela função.
- handlers - contém manipuladores, que podem ser usados por essa função ou mesmo em qualquer lugar fora dela.
- defaults- variáveis padrão para a função (consulte Usando variáveis para obter mais informações).
- vars- outras variáveis para a função (consulte Usando variáveis para obter mais informações).
- files - contém arquivos que podem ser implantados por meio dessa função.
- templates - contém modelos que podem ser implantados por meio dessa função.
- meta- define alguns metadados para esta função. Veja abaixo para mais detalhes.

```
webserver/
  tasks/
  handlers/
  files/
  templates/
  vars/
  defaults/
  meta/
```

Demonstração e Mão na massa !!

- https://www.mindtheproduct.com/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/
- https://pt.slideshare.net/davidhahn1004/continuous-integrationdeployment-with-gitlab-ci
- https://pt.slideshare.net/RahulBajaj94/ansible-92287889
- https://docs.gitlab.com/ee/ci/yaml/
- https://docs.gitlab.com/runner/executors/README.html
- https://docs.ansible.com/
- https://devopsideas.com/ansible-local-setup-using-vagrant-virtualbox/
- https://docs.microsoft.com/en-us/azure/devops/learn
- https://medium.com/cloudnativeinfra/when-to-use-which-infrastructure-as-code-tool-665af289fbde