github actions

NKLCB

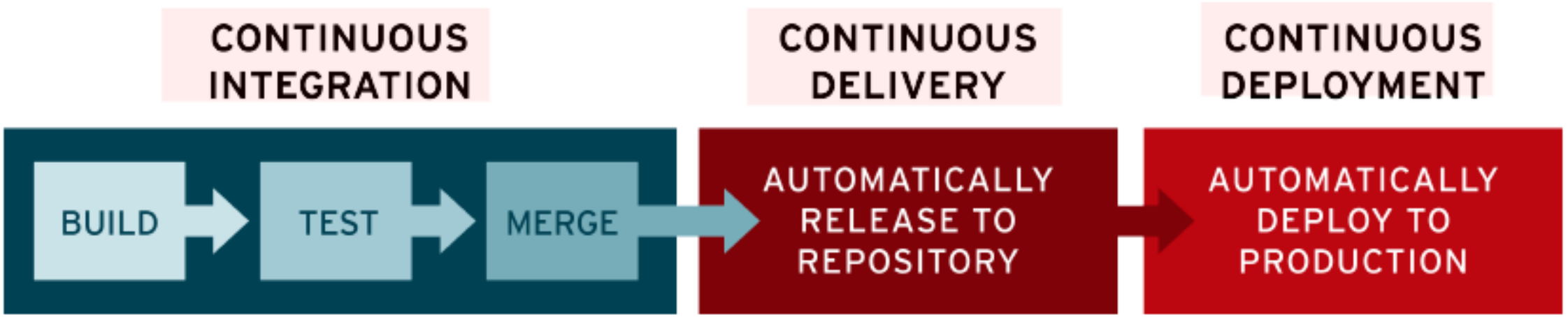
DevOps

github organization

DevOps = Development + Operations

- 개발과 운영의 합성어
- Dev: Plan - Code - Build - Test
- Ops: Release - Deploy - Operate - Monitor
- Cross Functional Team: 개발과 운영을 한 팀으로 묶어 프로세스의 자동, 단일화
- CI/CD Tool 이용하여 Build, Test, Deploy 자동화
- Pros
    - 커뮤니케이션 리소스 개선
    - 개발, 배포 속도가 빨라짐
    - 프로세스 간소화
    - 짧은 릴리즈 주기

CI/CD



- TravisCI, CircleCI, Jenkins

CI(Continuous Integration)

- 자동화된 프로세스
- 코드 변경사항의 정기적 빌드, 테스트 병합 자동화
- Pros
  - 빠른 디버깅
  - 코드 품질 개선
  - 검증 및 릴리즈 시간 단축

CD(Continuous Delivery(or Deployment))

- Continuous Delivery: 공유 저장소로 자동 Release(Test -> Staging)
- Continuous Deployment: Production Level까지 자동 Deploy(Test -> Staging -> Production)
- MSA(MicroService Architecture) + Agile 일 경우, 사용자에게 최대한 빠른 시간안에 Production 제공 필요

Issue & PR Templates

- configuration file을 만들어 issue와 PR 내용의 template 지정 가능
- `.github/ISSUE_TEMPLATE/*.md` 에 template 생성 가능
- `Settings` - Features_Set up templates 에서 손쉽게 생성 가능
- `.github/PULL_REQUEST_TEMPLATE/pull_request_template.md`

pull_request_template.md

```
Please ensure your pull request adheres to the following guidelines:

- [ ] Use the following format: `* [owner/repo](link)`
- [ ] Link additions should be added to the bottom of the relevant category.
- [ ] New categories or improvements to the existing categorization are welcome.
- [ ] Search previous suggestions before making a new one, as yours may be a duplicate.
- [ ] Sort by alphabetical order

Thanks for contributing!
```

CI/CD tools for github

GitHub Actions

github actions

- github에서 공식 제공하는 CI/CD Tools
- 개발 workflow 자동

github actions - core

- Workflow
- Event
- Job
- Step
- Action
- Runner

Workflow

- Job들로 구성. Event에 의해 트리거되는 자동화된 프로세스
- 최상위 개념
- YAML으로 작성되며, .github/workflows에 저장

Event

- Workflow를 실행하는 규칙
- Push, pull request, Cron, webhook으로 연결된 외부 이벤트에 의해 실행

Job

- Step들로 구성
- 가상환경의 인스턴스에서 실행
- 다른 Job에 의존관계를 가질 수 있고, 독립 병렬 실행도 가능

Step

- Task들의 집합으로 커맨드를 실행하거나 action을 실행

Action

- 가장 작은 단위
- Step을 연결해 Job을 구성
- 재사용 가능
- marketplace나 개인이 만든 action을 사용할 수 있음

Runner

- workflow가 실행될 인스턴스
- github-hosted runner와 self-hosted runner로 나뉨

echo hello

```yaml
name: Hello

on:
    push:
        branches:
        - main

jobs:
    hello:
        runs-on: ubuntu-latest
        steps:
          - name: Say Hello
            run: echo "hello, world!"
```

echo something

```yaml
name: CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main, develop ]

  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Run a one-line script
        run: echo Hello, world!

      - name: Run a multi-line script
        run: |
          echo Add other actions to build,
          echo test, and deploy your project.
```

Adding a workflow status badge

```
![example workflow](https://github.com/<OWNER>/<REPOSITORY>/actions/workflows/<WORKFLOW_FILE>/badge.svg)
```

Adding Another types of status badge

shields.io

규칙 정하기

```yaml
on:
    push:
        branches: [ main, develop ]
        tags: "v*"
    pull_request:
        branches: [ develop ]

on:
    schedule:
    - cron: "* * * * * *"
```

Don't reinvent the wheel

- CI/CD 파이프라인은 그 절차와 각 단계에서의 할 일이 정해져 있는 편
- 이미 만들어진 것에 커스터마이즈 하는 것이 CI/CD 스크립트 짜는 시간을 개발에 투입 가능

Wiki

- 팀 프로젝트에서 가장 중요한 것: 문서
- 프로젝트와 관련된 모든 문서를 wiki로 만들어 repo와 함께 쉽게 관리 가능
- repo_name.`wiki`.git 을 clone 하여 local에서 관리 가능