

커뮤니티의 개발 워크플로 자동화 적용기

CI/CD @ GitHub for Cloud-Barista Community

김윤곤
ETRI 연구원



CONTENTS

- 01 Cloud-Barista 커뮤니티 개요
- 02 CI/CD 및 GitHub Actions 소개
- 03 커뮤니티 개발 워크플로 자동화
- 04 맺음말





01

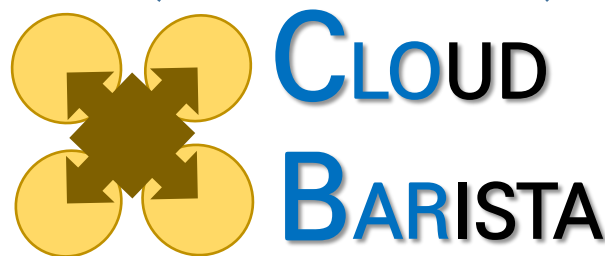
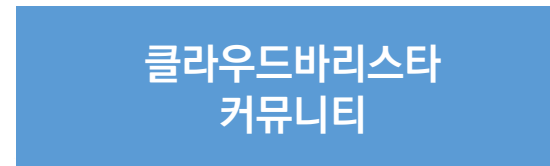
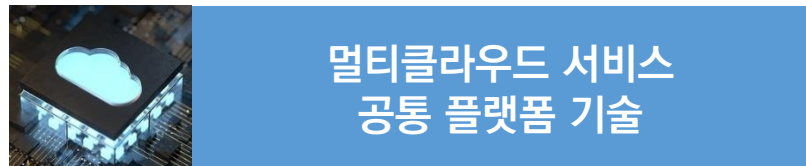
Cloud-Barista 커뮤니티 개요

01

클라우드바리스타(Cloud-Barista)란?

멀티클라우드 서비스/솔루션을 만드는데 반드시 요구되는
기반, 공통SW 기술

멀티클라우드 서비스/솔루션을 만드는
공개SW 커뮤니티



멀티클라우드 C.E.O SW 확보

- (Common) 멀티클라우드 서비스/솔루션에 공통 적으로 요구되는 핵심SW 개발
- (Efficient) 개별 기업/기관마다 중복개발의 비효율성을 제거
- (Open) 니즈가 있는 수요자라면, 누구나 자유롭게 사용할 수 있는 공개SW로 제공

수요자에 대한 기술 내재화

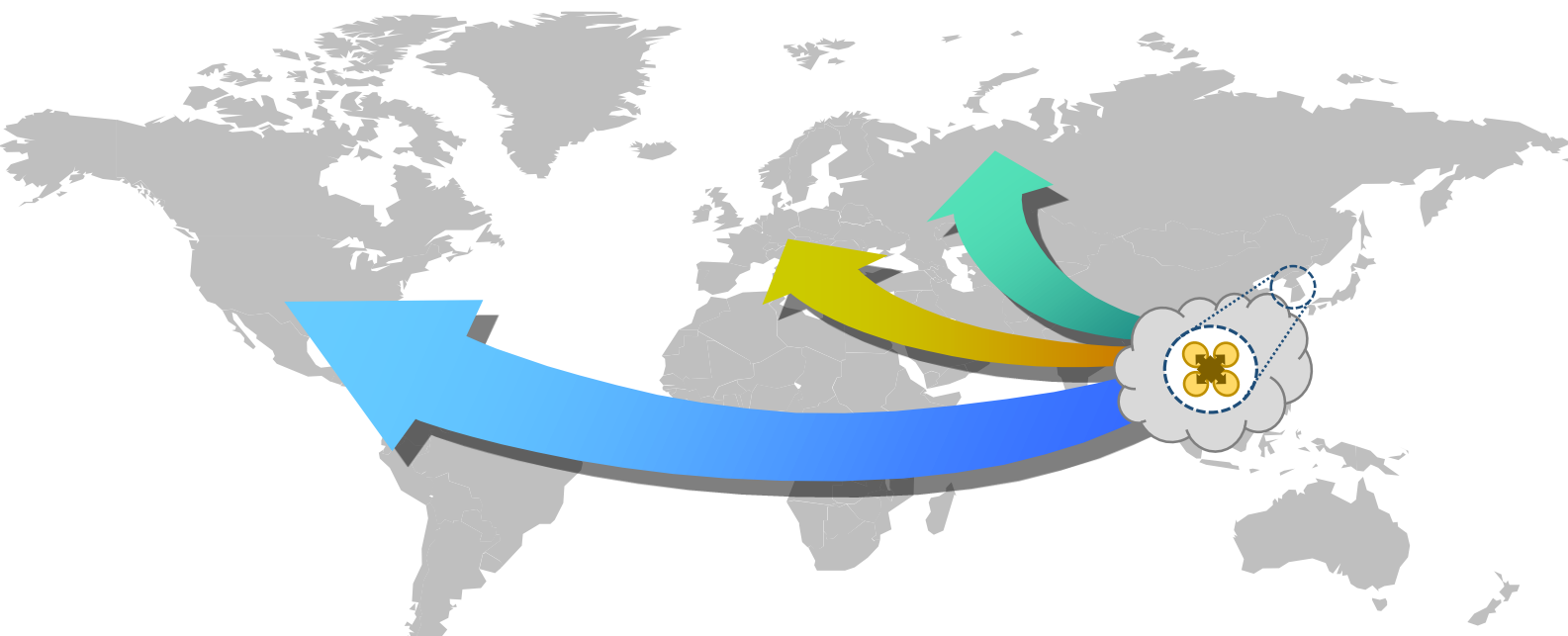
- (소통) Cloud-Barista 기술의 국내 내재화를 위한 소통 창구
- (공유) Cloud-Barista의 개발 결과물(소스코드, 문서, 노하우 등) 공유의 장
- (협업) 자발적 개발자 및 참여자와의 협업 장소

기술활용.확산의 GAP 해소

01

Cloud-Barista의 목표/지향점

그 자체가 글로벌 스케일인, 멀티클라우드를 기반으로
우리의 서비스를 세계 곳곳으로 보내는 그날까지...

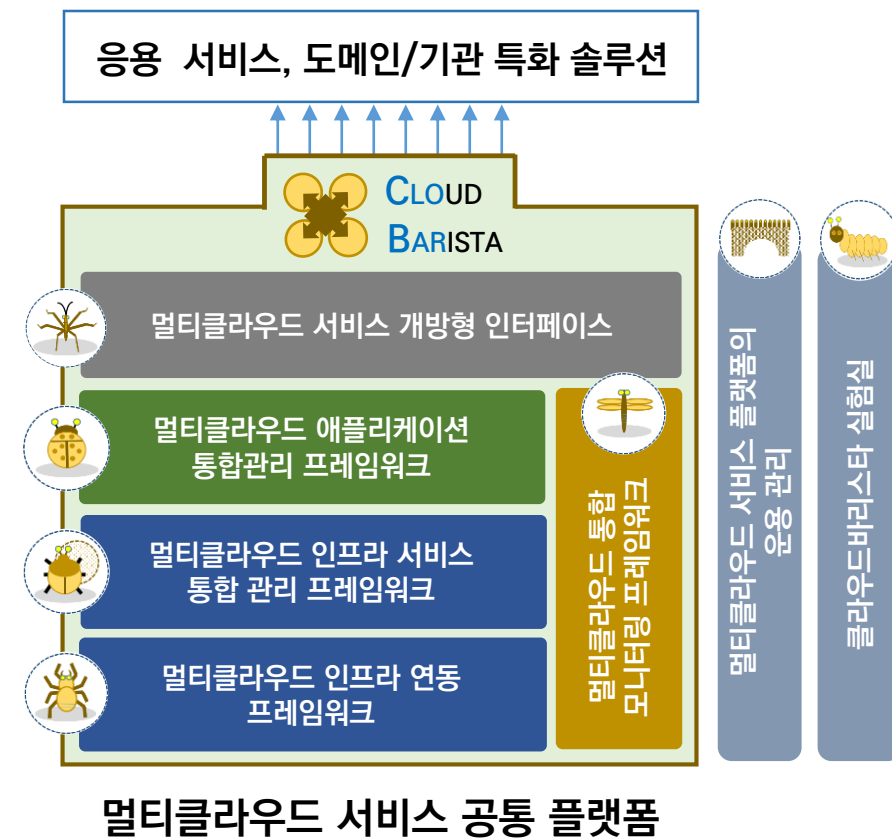


이제 까지는... 해외 글로벌 기술 기반의 국내 최적화 솔루션 개발
앞으로는... 국내 기술 기반으로 글로벌 솔루션을 개발 !

“Cloud-Barista Community의 공개SW 활동 비전 및 현황”이 궁금하시다면?

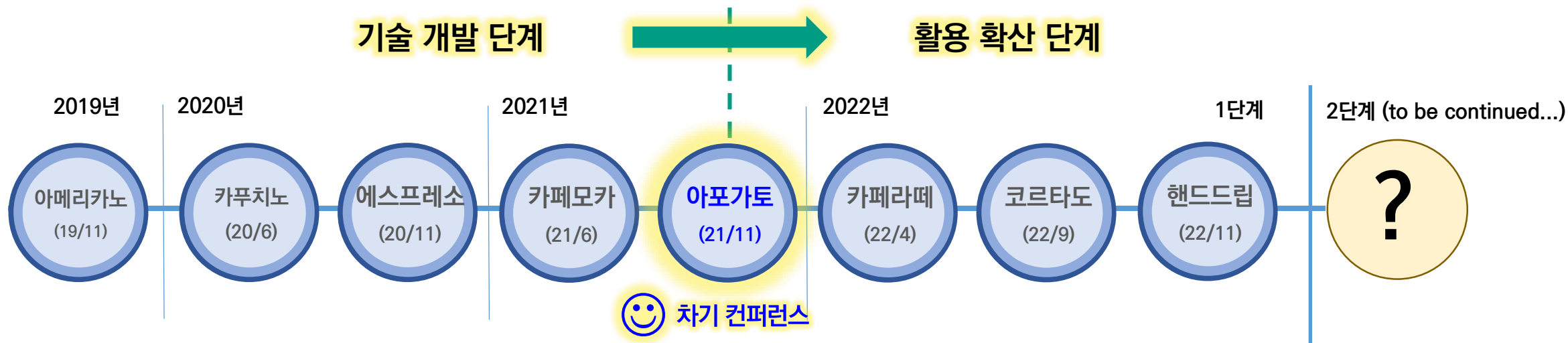
Click! → <https://youtu.be/JOwmFLMxc1w> YouTube

멀티클라우드 기술 내제화를 위해
굵직한 기술을 공개SW로...



01

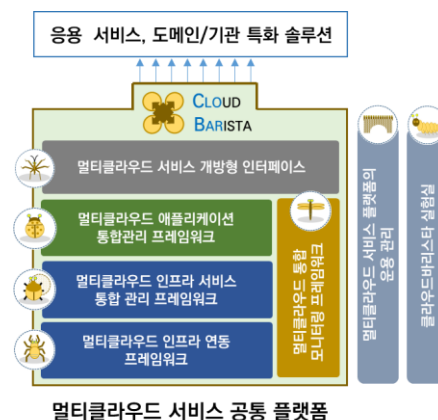
커뮤니티 차원의 CI/CD 체계 도입 필요성 대두



기술 성숙도
신규 컨트리뷰터



기여 내용 검토량 증가
손수 검토하기 어려움



업무 자동화 필요성 증가
CI/CD 체계 도입 필요성 대두





02

CI/CD 및 GitHub Actions 소개

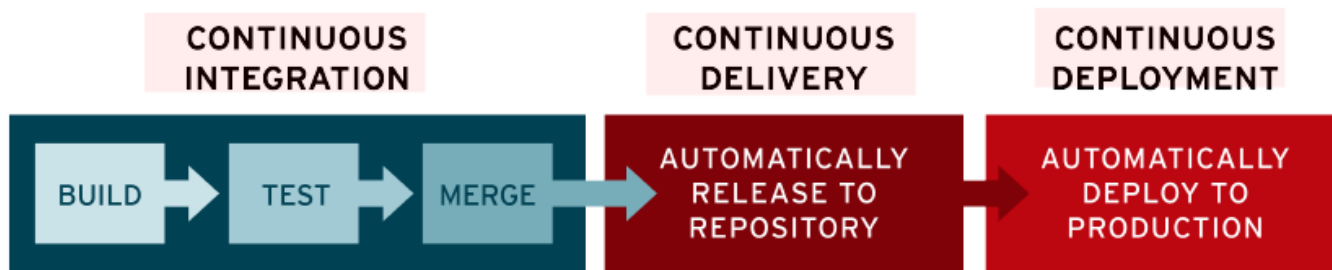
02

CI/CD 란?

- CI/CD: Continuous Integration / Continuous Delivery/Deployment
- CI/CD(방법/전략): 애플리케이션/소프트웨어 개발 단계를 자동화하여 보다 짧은 주기로 배포하는 방법 또는 전략

* DevOps를 실현을 위해 **지속통합(CI)**과 **지속배포(CD)** 자동화는 필수

CI/CD 파이프라인: 새 버전의 소프트웨어를 제공하기 위해 수행해야 할 일련의 단계



CI/CD (자동화) 도구



...



GitHub Actions 채택

(For free) Public repository는 무료 사용
(Customizing) 공개SW 관련 워크플로 자동화를 고려

02

GitHub Actions란?



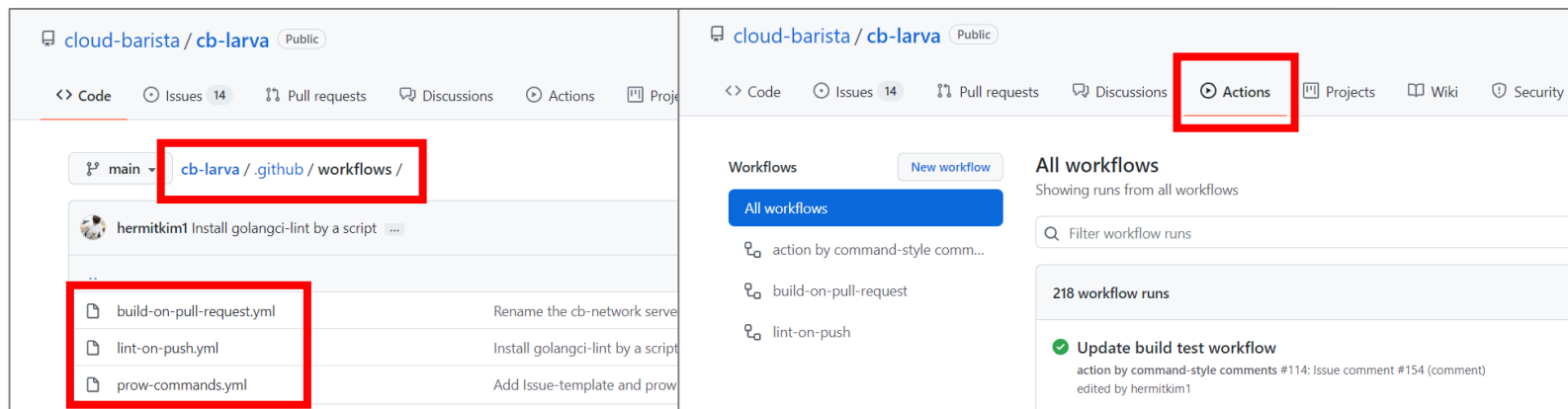
- GitHub에서 제공하는 워크플로(Workflow) 자동화 도구

워크플로 위치

워크플로 수행 결과



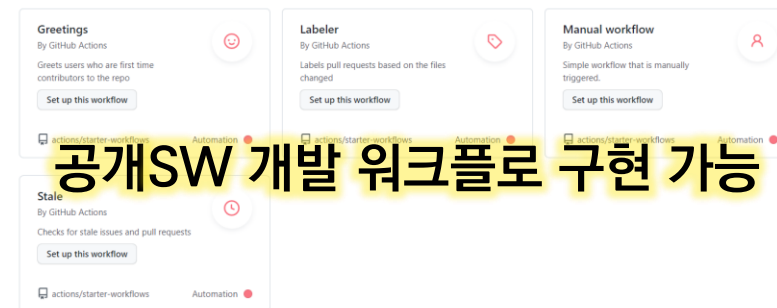
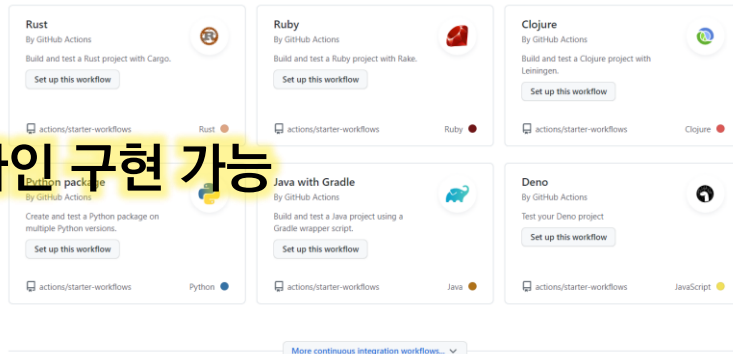
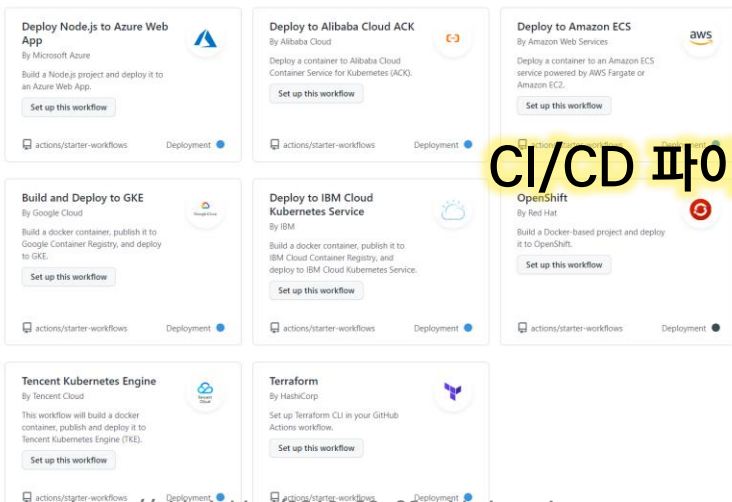
Source: <https://github.com/features/actions>



Deploy your code with the popular service

Continuous integration workflows

Automate every step in your process



CI/CD 파이프라인 구현 가능

공개SW 개발 워크플로 구현 가능

Source: <https://github.blog/2019-08-08-github-actions-now-supports-ci-cd/>

GitHub-hosted Runner

Supported runners and hardware resources

Each virtual machine has the same hardware resources available.

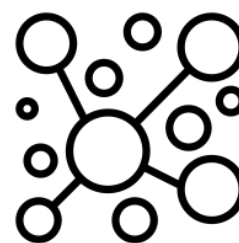
- 2-core CPU
- 7 GB of RAM memory
- 14 GB of SSD disk space

Virtual environment	YAML workflow label
Windows Server 2019	<code>windows-latest</code> or <code>windows-2019</code>
Ubuntu 20.04	<code>ubuntu-20.04</code>
Ubuntu 18.04	<code>ubuntu-latest</code> or <code>ubuntu-18.04</code>
Ubuntu 16.04	<code>ubuntu-16.04</code>
macOS Big Sur 11.0	<code>macos-11.0</code>
macOS Catalina 10.15	<code>macos-latest</code> or <code>macos-10.15</code>

Note: The Ubuntu 20.04 virtual environment is currently provided as a preview only. The `ubuntu-latest` YAML workflow label still uses the Ubuntu 18.04 virtual environment.

Note: The macOS 11.0 virtual environment is currently provided as a private preview only. Any users or organizations that are already using this runner can continue using it, but we're not accepting any further users or organizations at this time. The `macos-latest` YAML workflow label still uses the macOS 10.15 virtual environment.

Self-hosted Runner



Deploy Node.js to Azure Web App

By Microsoft Azure

Build a Node.js project and deploy it to an Azure Web App.

Set up this workflow

actions/starter-workflows Deployment

Deploy to Alibaba Cloud ACK

By Alibaba Cloud

Deploy a container to Alibaba Cloud Container Service for Kubernetes (ACK).

Set up this workflow

actions/starter-workflows Deployment

Deploy to Amazon ECS

By Amazon Web Services

Deploy a container to an Amazon ECS service powered by AWS Fargate or Amazon EC2.

Set up this workflow

actions/starter-workflows Deployment

Build and Deploy to GKE

By Google Cloud

Build a docker container, publish it to Google Container Registry, and deploy to GKE.

Set up this workflow

actions/starter-workflows Deployment

Deploy to IBM Cloud Kubernetes Service

By IBM

Build a docker container, publish it to IBM Cloud Container Registry, and deploy to IBM Cloud Kubernetes Service.

Set up this workflow

actions/starter-workflows Deployment

OpenShift

By Red Hat

Build a Docker-based project and deploy it to OpenShift.

Set up this workflow

actions/starter-workflows Deployment

Tencent Kubernetes Engine

By Tencent Cloud

This workflow will build a docker container, publish and deploy it to Tencent Kubernetes Engine (TKE).

Set up this workflow

actions/starter-workflows Deployment

Terraform

By HashiCorp

Set up Terraform CLI in your GitHub Actions workflow.

Set up this workflow

actions/starter-workflows Deployment

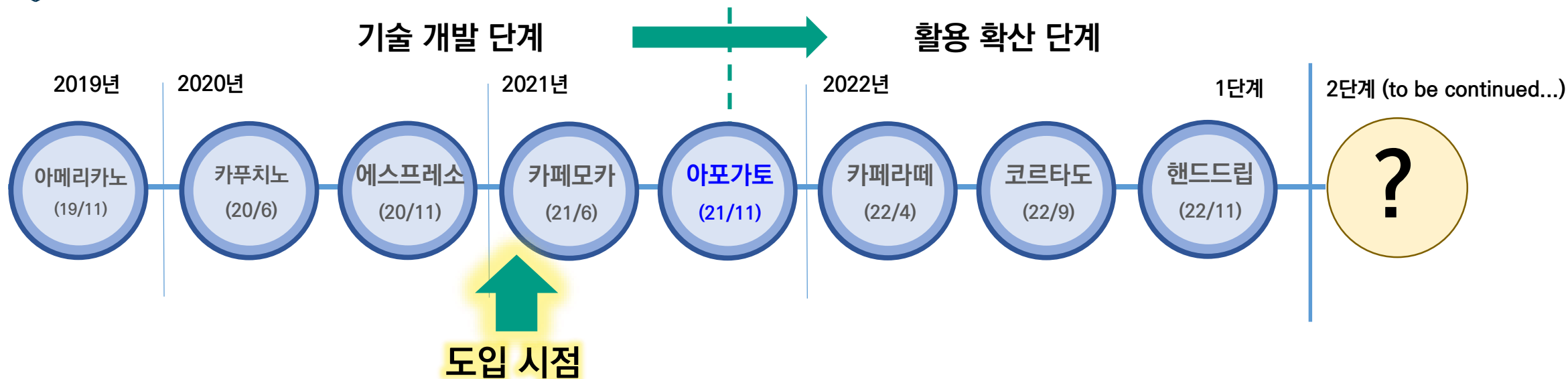


03

커뮤니티 개발 워크플로 자동화

03

커뮤니티의 CI/CD 도입 시점 및 상황

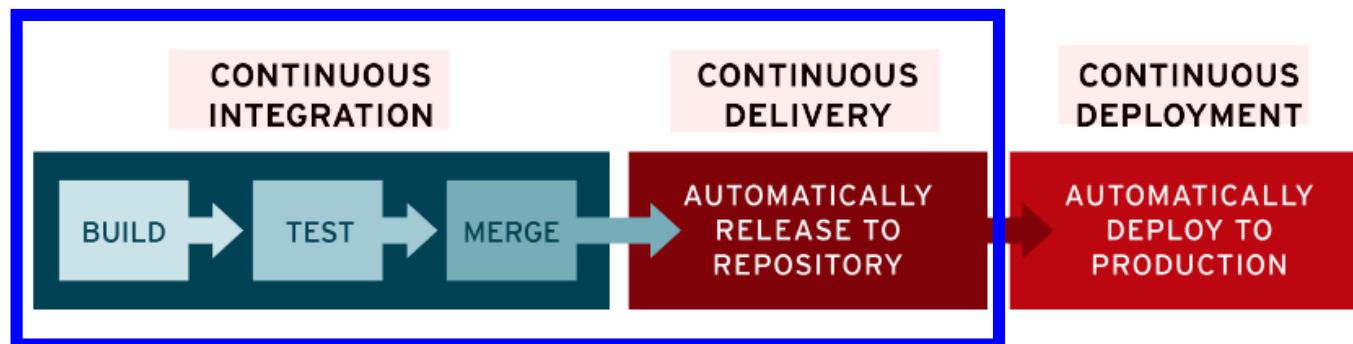


도입 상황(이슈)
(CI/CD 도입은 쉽지 않아요... π)

상당한 양의
기 개발된 소스코드 개선

앞으로 개발되는/기여되는 새로운 소스코드 통합으로 인하여
발생하는 이슈(소위: Integration hell) 대비

범위

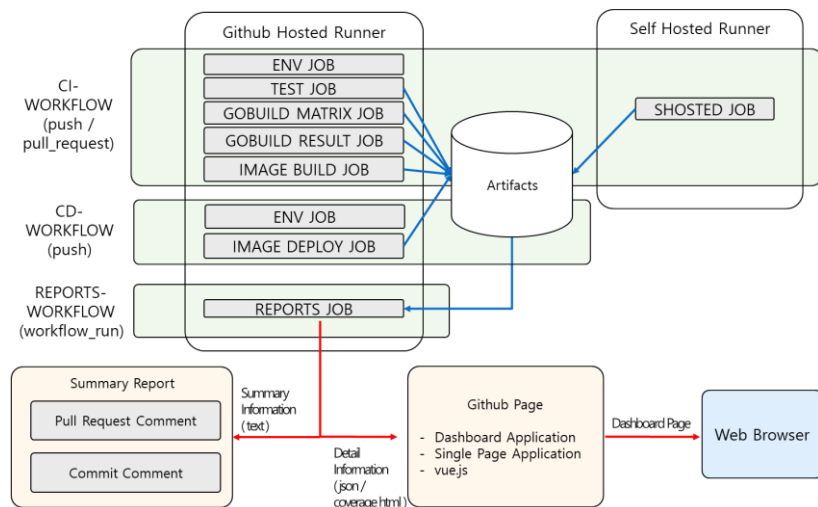


[GitHub Actions] Cloud-Barista의 두가지 워크플로

번거롭게 반복하고 있는 작업 있다면? 자동화 하기 위해 한번쯤 검토해 보시면 좋을 것 같습니다 ^^

CI/CD 자동화 관련 워크플로 및 Jobs

- CI
 - ✓ Build test
 - ✓ Lint test
 - ✓ Unit test/API test
- CD
 - ✓ Container image publish test
- Reports
 - ✓ Generate reports
 - ✓ Publish reports



(소개) Cloud-Barista 개발 워크플로 자동화 및 통합 리포트 체계

공개SW 개발 워크플로 및 Jobs

- ✓ Prow-command
- ✓ Bi-directional wiki sync
- ✓ contribute-list

그 밖에:

Automate every step in your process

Greetings
By GitHub Actions

Greets users who are first time contributors to the repo

Set up this workflow

actions/starter-workflows Automation

Labeler
By GitHub Actions

Labels pull requests based on the files changed

Set up this workflow

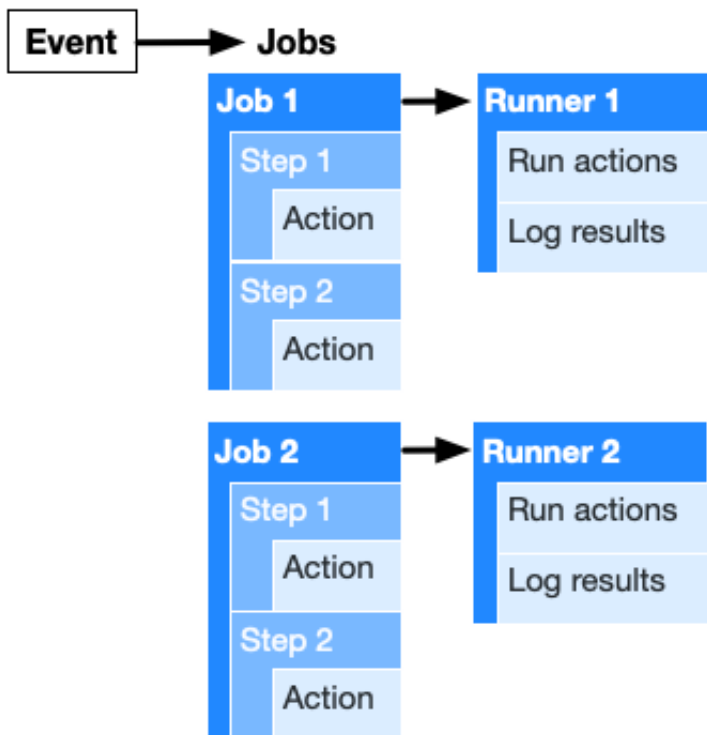
actions/starter-workflows Automation

03

GitHub Actions 동작 원리



- GitHub Actions 동작 원리: “Event” 기반으로 “Job”을 수행

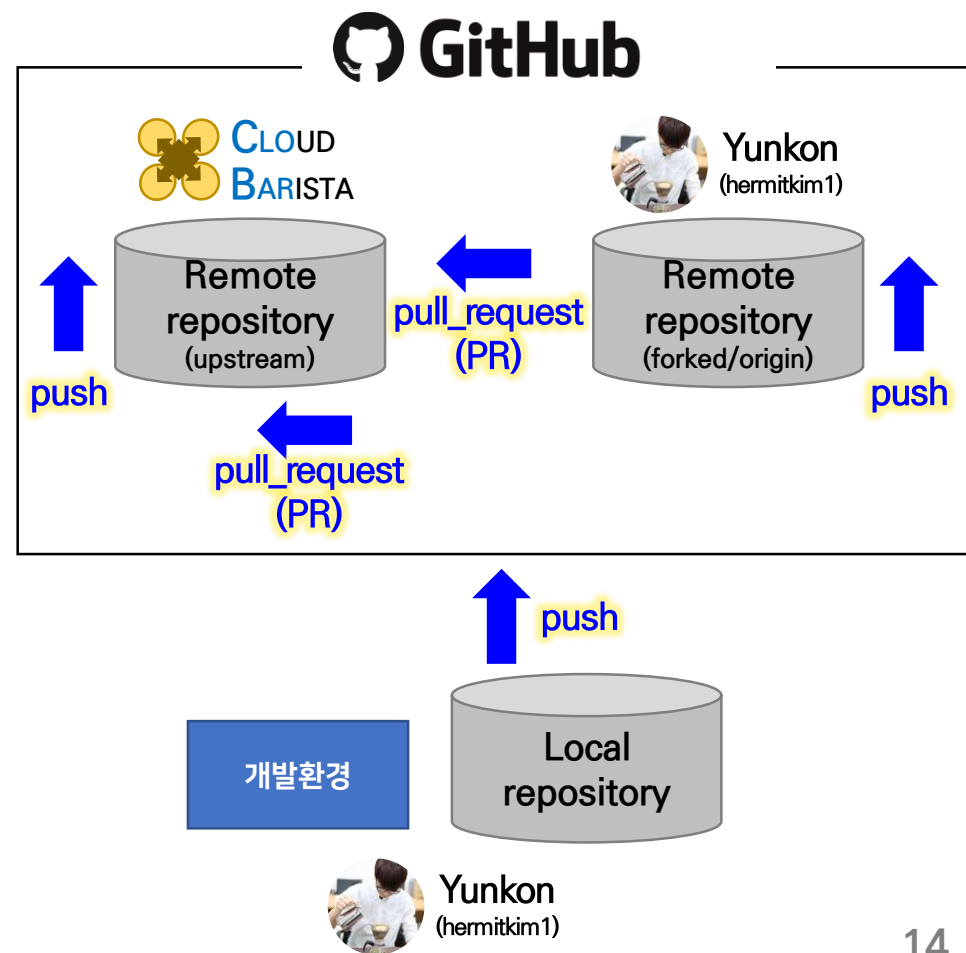


[An example workflow]

```

name: learn-github-actions
on: [push, pull_request]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - uses: actions/setup-node@v2
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
  
```

[The path of yaml files] {Repository}/.github/workflows
예) cb-larva/.github/workflows



03

[Build test] CI 워크플로

- Build test 목적: 소스코드의 빌드 성공/실패 여부 점검

Build test “자동화” 적용



“pull_request” event



GitHub Actions

“Build test” 워크플로 수행



```
strategy:
matrix:
  os: [ubuntu-18.04, ubuntu-20.04]
  go: ["1.16", "1.17"]
```



Review required

At least 1 approving review is required by reviewers with write access. [Learn more.](#)

[Show all reviewers](#)

1 pending reviewer



All checks have passed
1 successful check

[Hide all checks](#)

Build amd64 container image / Building (pull_request) Successful in 2m

[Details](#)

Merging is blocked

Merging can be performed automatically with 1 approving review.

Enable auto-merge

Automatically merge when all requirements are met. [Learn more](#)

번거로운 공수



Before

- ✓ 신규 소스코드 Merge 이후 빌드 에러 발생
- ✓ Maintainer가 신규 소스코드 Build test후 Merge

After



기술 개발/개선에 집중할 수 있음

- ✓ Build test 성공 시 → 코드리뷰
- ✓ Build test 실패 시 → 기여자에게 알리고 개선에 대해 논의

Cloud-Barista 커뮤니티의 build test 방법

방법 1: “go build” on Ubuntu 18.04

방법 2: Container image build by a Dockerfile

- Lint test 목적: 소스코드 품질 향상/유지

* 린트(lint) 또는 린터(linter): 소스 코드를 분석하여 프로그램 오류, 버그, 스타일 오류, 의심스러운 구조체에 표시(flag)를 달아놓기 위한 도구들

`golanci-lint` is a Go linters aggregator.

- deadcode - Finds unused code
- errcheck - Errcheck is a program for checking for unchecked errors in go programs. These unchecked errors can be critical bugs in some cases
- gosimple - Linter for Go source code that specializes in simplifying a code
- govet - Vet examines Go source code and reports suspicious constructs, such as Printf calls whose arguments do not align with the format string
- ineffassign - Detects when assignments to existing variables are not used
- staticcheck - Staticcheck is a go vet on steroids, applying a ton of static analysis checks
- structcheck - Finds unused struct fields
- typecheck - Like the front-end of a Go compiler, parses and type-checks Go code
- unused - Checks Go code for unused constants, variables, functions and types
- varcheck - Finds unused global variables and constants
- asciicheck - Simple linter to check that your code does not contain non-ASCII identifiers
- bodyclose - checks whether HTTP response body is closed successfully
- depguard - Go linter that checks if package imports are in a list of acceptable packages

...

- 여러 Linter가 있음
- 적용 목표/범위 설정 필요

03

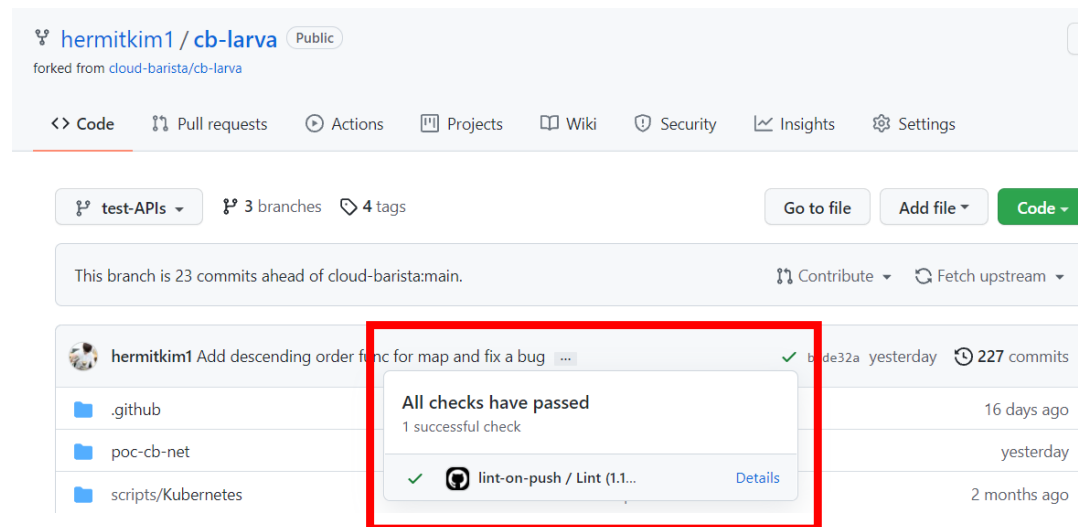
[Lint test] CI 워크플로

Lint test “자동화” 적용

Contributor/Developer
“pull_request” / “push” event



GitHub Actions
“Lint test” 워크플로 수행



Lint test 결과

- deadcode(lint) : 42 found
- errcheck(lint) : 164 found
- staticcheck(lint) : 60 found
- revive(lint) : 788 found
- gofmt(lint) : 118 found
- govet(lint) : 371 found
- gocyclo(lint) : 102 found
- golint(lint) : 629 found
- ineffassign(lint) : 50 found
- misspell(lint) : 4 found

✓ 기 개발된 코드에 적용시
항목별 Lint error를 발생 (다수..)

✓ (공수) 모두 해결 vs. 점차 해결

✓ Maintainer의 판단에 따라
Lint 도구 및 적용시기를 정함

[참고] 현재는 선택사항:

신규 기여 시 Lint test 수행 하더라도 기여자에게 수정을 요구하지 않는 상황

[의견]

- ✓ 처음에는 번거로운 작업 일 수 있음
- ✓ 코드 품질 향상 → 기술 개발 속도 향상

03

[Unit/API test] CI 워크플로

- Unit test 목적: 기능 단위로 정상/비정상 입력 값에 따라 상응하는 결과 출력 확인

Code Coverage (평가 지표)

- 소프트웨어의 테스트를 논할 때 얼마나 테스트가 충분한가를 나타내는 지표중 하나.
- 소프트웨어 테스트를 진행했을 때 코드 자체가 얼마나 실행되었는지 숫자로 볼 수 있다.

Statement
Coverage
예시

```

1: func Print(a int, b int) {
2:     sum := a + b
3:     if sum > 0 {
4:         fmt.Print("This is a positive result")
5:     } else {
6:         fmt.Print("This is negative result")
7:     }
8: }

```

$$6 / 8 * 100 = 75\%$$

화이트박스 테스트(White box test)

- 객체 내부를 확인하고 검증하는 테스트이다.
- 쓰이지 않은 변수는 없는지..? 특정 범위만 받는 함수가 있는지..? 확인하는 과정
- 코드 커버리지(Code Coverage)는 화이트박스 테스트의 일부**

코드 커버리지 기준

- 구문 (Statement Coverage): 코드 한 줄이 한번 이상 실행된다면 충족**
- 결정 (Decision Coverage)(Branch Coverage): 전체적인 결과가 참/거짓이면 충족**
- 조건 (Condition Coverage): 각 내부 조건이 참/거짓을 가지게 되면 충족**

03

[Unit/API test] CI 워크플로

- API test 목적: API 단위로 정상 입력 값에 따라 상응하는 결과 출력 확인

✓ 많은 기 개발 사항으로

✓ API test로 변경하여 테스트 진행

* 테스트 시나리오 작성 필요

API test “자동화” 적용



Contributor/Developer

“pull_request”



GitHub Actions

“API test” 워크플로 수행



[관련 이슈]

API 변경/추가 시 다량의 수정사항 발생

✓ API가 성숙한 후 적용

✓ Unit test 적용 검토

블랙박스 테스트(Black box test)

- 테스트 시 객체 내부에 무엇이 들어 있는지 알 수 없거나 알지 않아도 된다는 것을 가정하여 테스트하는 방법
- 객체 내부가 어떻게 변하던 상관없이 입력을 주었을때 원하는 결과값이 나오면 테스트는 통과함

CB-SPIDER CI/CD Dashboard

Summary

Lint / deadcode
42 found

[details](#)

Lint / errcheck
164 found

[details](#)

Lint / staticcl
60 found



github-actions bot commented on 12 Aug

Summary Result

- deadcode(lint) : 42 found
- errcheck(lint) : 164 found
- staticcheck(lint) : 60 found
- revive(lint) : 788 found
- gofmt(lint) : 118 found
- govet(lint) : 371 found
- gocyclo(lint) : 102 found
- golint(lint) : 629 found
- ineffassign(lint) : 50 found

Lint / gofmt
118 found

[details](#)

Lint / govet
371 found

[details](#)

Lint / gocyclo
102 found

Lint / ineffassign
50 found

[details](#)

Lint / misspell
4 found

[details](#)

Test / unit
0 failure

[436 cases](#) [details](#)

Test / coverage
43.0%

[details](#)

Test / go bui
Success

Deploy / docker build
Success

[details](#)

Deploy / pu
Skip

[\[details\]](#)

- coverage total : 43.0%
- unit test : 436 tests, 0 failure

- docker build : success
- deploy : -
- self host runner : success

03

[Container image publish test] CD 워크플로

- Container image publish test 목적:
Docker Hub / GitHub Container Registry (GHCR)에 이미지 배포 작업 수행에 따른 성공/실패 여부 확인

Container image publish test “자동화” 적용

Docker Hub 활용 ◀ Before

- ✓ 소스코드 push 발생 시 Webhook을 통해 Docker Hub에 새로운 이미지 빌드를 트리거링

After ▶ Docker Hub 및 GHCR 활용

- ✓ 소스코드 Merge시 Docker image publish 수행
- ✓ (적용시도) Multi-arch image build

Multi-arch image build
(선택과 집중을 위해 적용 보류)

```
name: Build and publish
id: docker_build
uses: docker/build-push-action@v2.7.0
with:
  builder: ${{ steps.buildx.outputs.name }}
  context: ./
  file: ./Dockerfile
  target: prod
  platforms: linux/amd64 # linux/arm/v7,linux/arm64,linux/386,linux/ppc64le,linux/s390x,linux/arm/v6
  ...
```

- Reports 워크플로: CI/CD 워크플로 수행 후 결과 리포트 작성 및 배포

CB-SPIDER Dashboard

Summary

Lint / deadcode 32 found details	Lint / errcheck 38 found details	Lint / staticcheck 27 found details	Lint / revive 50 found details
Test / unit 0 failure 31 cases details	Test / coverage 9.1% details	Test / go build Success details	
Deploy / docker build Skip details		Deploy / publish image Skip details	

Details

deadcode errcheck staticcheck revive unit go build docker build publish image

```

cloud-control-manager/cloud-driver/drivers/alibaba/main/old/t.go:10:6: `main` redeclared in this block (typecheck)
func main() {
^
cloud-control-manager/cloud-driver/drivers/alibaba/main/old/TR.go:333:6: other declaration of main (typecheck)
func main() {
^
cloud-control-manager/cloud-driver/drivers/alibaba/main/old/TR.go:47:34: PublicIPHandler not declared by package resources (typecheck)
handler := ResourceHandler.(irs.PublicIPHandler)
^
cloud-control-manager/cloud-driver/drivers/alibaba/main/old/TR.go:102:20: PublicIPReqInfo not declared by package resources (typecheck)

```

Pull request action test 1 #1

Merged jmleefree merged 1 commit into `cloud-barista:master` from `jmleefree:action-test` 8 days ago

Conversation 1 Commits 1 Checks 3 Files changed 1



jmleefree commented 8 days ago

Member ...

action test

Pull request action test 1

✓ 4cf68d5



github-actions bot commented 8 days ago

...

Summary Result

- deadcode(lint) : 32 found
- errcheck(lint) : 38 found
- staticcheck(lint) : 27 found
- revive(lint) : 50 found
- coverage total : 9.1%
- unit test : 31 tests, 0 failure
- go build : success
- docker build : -
- deploy : -

[\[details\]](#)

03

공개SW 개발 워크플로

“prow-commands” 워크플로:
command-style comments에 의해 워크플로 수행
(효율적인 소통)

* Prow is a Kubernetes based CI/CD system

ChobobDev commented 21 hours ago • edited ▾ Member Author ☺ ...

I have used VS Code Extensions to review my commit so I believe there are no more Typos on this document.

Again, Thank you for your hard work in finding my typos on this markdown file.

Thank you

☺ ❤️ 1

seokho-son commented 21 hours ago Member ☺ ...

/lgtn

github-actions (bot) added the **lgtn** label 21 hours ago

seokho-son commented 21 hours ago Member ☺ ...

/approve

github-actions (bot) approved these changes 21 hours ago View changes

seokho-son commented 21 hours ago Member ☺ ...

@ChobobDev Thank you~~!!

“bi-directional-wiki-sync” :
Docs/Wiki 업데이트시 양방향 동기화 수행(아이디어 기여)













main ▾ cb-coffeehouse / docs / Go to file Add file ▾ ...

PARKINHYO Fix typo of kubernetes deploy tutorial b300a66 18 days ago History

Ⓜ (Draft)-The-Cl-CD-workflows-and-guide-for-Clo...	Auto sync wiki to docs [skip actions]	3 months ago
Ⓜ (GitHub-Actions)-Skip-pull-request-and-push-w...	Auto sync wiki to docs [skip actions]	3 months ago
Ⓜ A-guide-to-the-development-environment-setu...	Create a guide to a dev env setup for Cloud-Barista	2 months ago
Ⓜ A-memo-to-create-MCIS-by-CB-Tumblebug-and...	Auto sync wiki to docs [skip actions]	3 months ago

“contribute-list” :
주기적(매일 자정)으로 Collaborators and contributors 업데이트

Collaborators and contributors

 Yunkon (Alvin) Kim	 K039_이도훈	 Eeeclipse	 INHYO	 Seokho Son	 Taegeon An
 Jae Hyeok Yu	 Jangh-lee	 KOHHyunkung	 Meet Gor	 Modney	 Sglim



04

맺음말

Earlier, Better

(개인적으로)

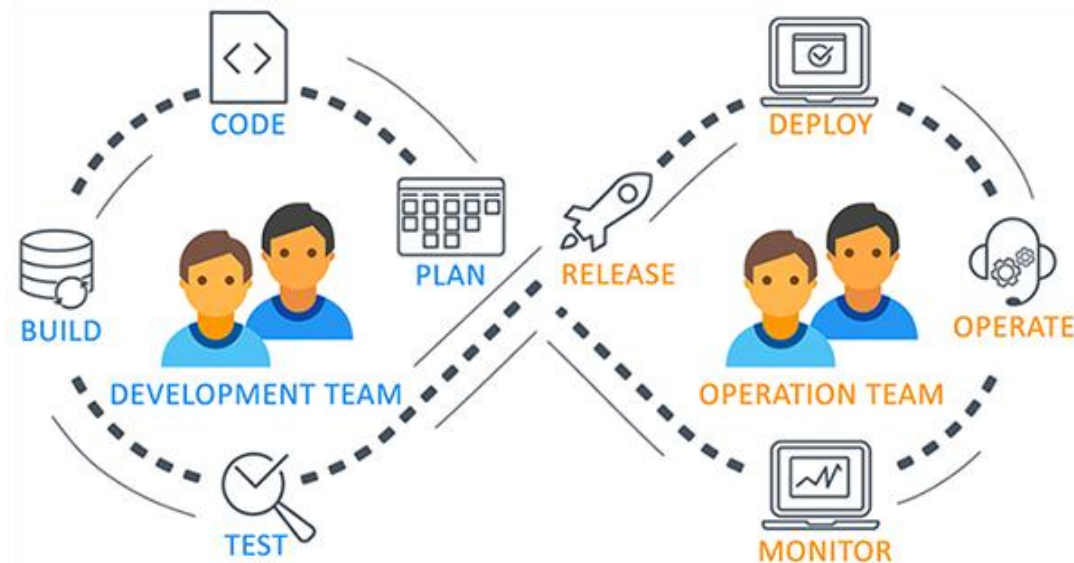
번거롭게 반복하고 있는 작업 있다면?
CI/CD 자동화 검토 추천!

공개SW 개발을 위한 CI/CD 자동화 도입 시
구성원 모두의 “발전적 이해관계 도출” 및 “역량 강화” 필요
(여러 사람이 함께 이해하고 만들어 나아가야 합니다)
(공수가 상당해요 ^^;;)

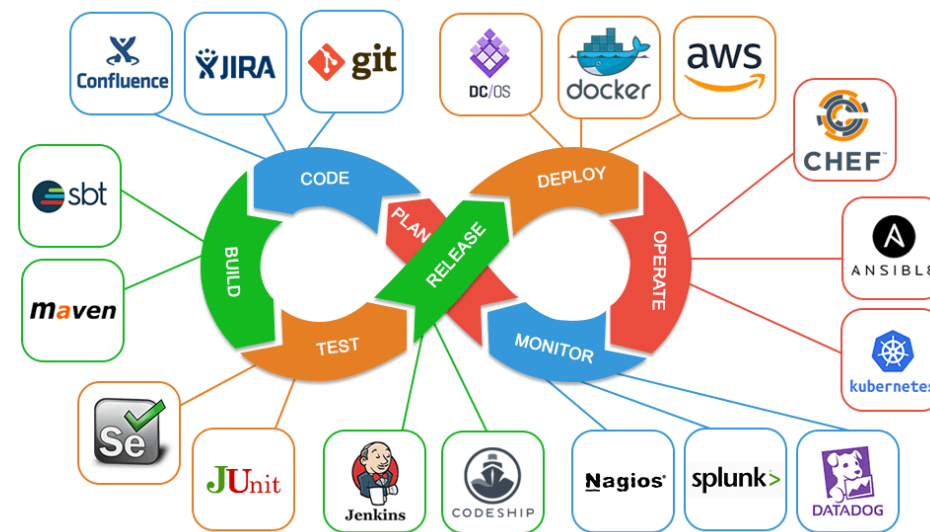
[참고] CI/CD 자동화는 DevOps의 핵심! (연계 가능)

- DevOps: Development(개발+테스트) + Operations(운영)
- DevOps(문화): 고객과 업무를 가장 잘 이해하는 비즈니스 조직원의 생각과 아이디어가 실제적인 IT서비스로 구현되고 계속 발전되는 "문화"를 만들자는 것

* DevOps문화에서 “속도”와 “반복”은 중요한 요소임 (비즈니스의 요구에 적시에 대처하고 지속적으로 개선 하기 위해)



Source: <https://www.dotnettricks.com/learn/devops/what-is-devops-and-devops-advantages>



Source: <http://nearcoding.com/what-is-devops/>

감사합니다

커뮤니티 개발 워크플로 자동화 적용기

