



IDC환경의 Gitlab CI를 AWS에 구성하기 (Feat, EKS & Pod Identity)

정영진 (LG U+)

간단한 자기소개



<https://linkedin.com/in/youngjin-jung>



LG U+ DevOps Engineer
AWS Community Builder

Backend Developer(APM)
→ DevOps Engineer

AWS/K8s

SRE

Migration/Modernization

Agenda

1. Why(왜 IDC의 Gitlab CI를 EKS환경에?)
2. What(어떤 걸 바꾸려고 했을까?)
3. How(어떻게 바꿨을까?)
4. Result(그래서 얼마나 좋아졌을까?)



1. Why

왜 IDC의 Gitlab CI를 EKS환경에?

이걸 하게 된 History

- 기존 SI 프로젝트로 구성된 파이프라인
- Public Cloud와 Private Cloud, Private Server 등 하이브리드로 구성된 환경
- Public Cloud에서 89개의 마이크로 서비스
- SI가 끝난 시점에 내재화를 통해 개발자들의 개발 효율성, 생산성을 재고

그렇다면 해야 할 것은...



그렇다면 해야 할 것은...

기존 파이프라인 이해와 분석!!

→ 하지만, 기존 Private 환경의 Jenkins + 권한도 요청...

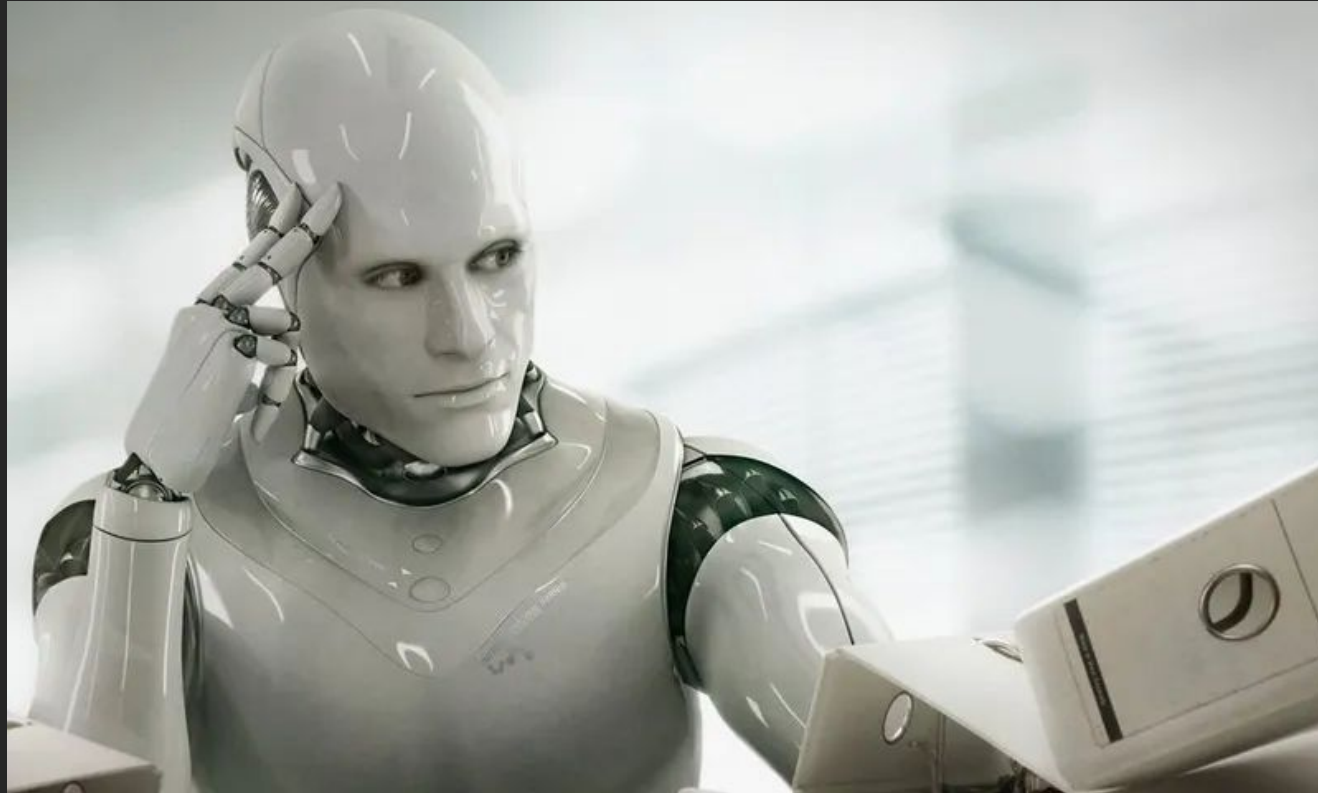
→ 빌드/배포 파이프라인 코드에 대한 History 파악의
어려움

그렇다면 해야 할 것은...



그렇다면 해야 할 것은...

지금 있는 코드를 분석해서 새로 만들자





2. What

어떤 걸 바꾸려고 했을까?

3가지의 변화

1. 개발자들의 개발 습관
2. IDC환경의 한정된 리소스로 구성된 빌드 환경과 속도
3. 확장성과 가독성을 겸비한 빌드 파이프라인

3가지의 변화

1. 개발자들의 개발 습관

Direct Push로 인해 Commit History 파악이 어려움

Commit Message가 무엇을 변경했는지 알 수 없음(퇴근 1트)

Branch 패턴이 제멋대로임(본인 이름으로 하는 경우도 있음)

3가지의 변화

2. IDC환경의 한정된 리소스로 구성된 빌드 환경과 속도

IDC에 위치한 Jenkins Build 서버로 병렬 빌드의 어려움

유지보수는 되고 있으나, 불필요한 태스크로 인해 빌드 시간 최대 2시간

Jenkins 서버가 장애난다면? SPOF

공용 계정

3가지의 변화

3. 확장성과 가독성을 겸비한 빌드 파이프라인

Jenkins로 구성되어 있었기 때문에 Groovy로 구성

하지만 Groovy가 제대로 관리되지 않아, 히스토리 파악이 어려움(이게 왜 추가됐지?)

기능 하나 수정 추가 하려면 SI로 1MM 산정 → 최소 1000만원을 들여야 함

3가지의 변화

3. 확장성과 가독성을 겸비한 빌드 파이프라인

Jenkins로 구성되어 있었기 때문에 Groovy로 구성

하지만 Groovy가 제대로 관리되지 않아, 히스토리 파악이 어려움(이게 왜 추가됐지?)

기능 하나 수정 추가 하려면 SI로 1MM 산정 → 최소 1000만원을 들여야 함

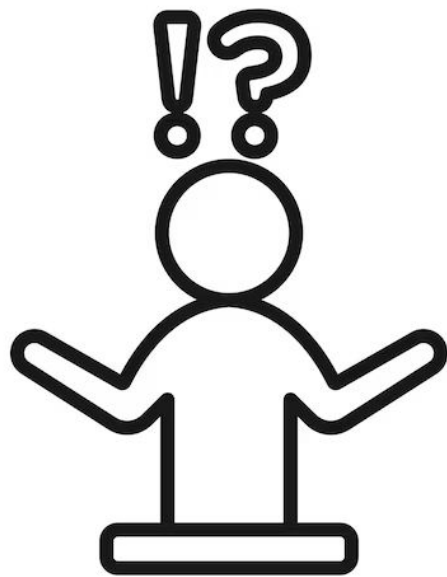


3. How

어떻게 바꿨을까?

고심 끝에...

Gitlab을 어떻게 잘쓰면 되지 않을까?



변화 요구사항에 대한 Gitlab 살펴보기

1. 개발자들의 개발 습관

→ **MR + Build Condition**

2. IDC환경의 한정된 리소스로 구성된 빌드 환경

→ **Gitlab Runner + EKS**

3. 확장성과 가독성을 겸비한 빌드 파이프라인

→ **Gitlab CI yaml을 공통화와 변수 사용**

변화 요구사항에 대한 Gitlab 살펴보기

1. 개발자들의 개발 습관

Merge requests

Tier: Free, Premium, Ultimate

Offering: GitLab.com, GitLab Self-Managed, GitLab Dedicated

History ...

A merge request (MR) is a proposal to incorporate changes from a source branch to a target branch.

When you open a merge request, you can visualize and collaborate on the changes before merge. Merge requests include:

- A description of the request.
- Code changes and inline code reviews.
- Information about CI/CD pipelines.
- A comment section for discussion threads.
- The list of commits.

rules

Use `rules` to include or exclude jobs in pipelines.

Rules are evaluated when the pipeline is created, and evaluated *in order*. When a match is found, no more rules are checked and the job is either included or excluded from the pipeline depending on the configuration. If no rules match, the job is not added to the pipeline.

`rules` accepts an array of rules. Each rules must have at least one of:

- `if`
- `changes`
- `exists`
- `when`

Rules can also optionally be combined with:

- `allow_failure`
- `needs`
- `variables`
- `interruptible`

변화 요구사항에 대한 Gitlab 살펴보기

2. IDC환경의 한정된 리소스로 구성된 빌드 환경과

속도



변화 요구사항에 대한 Gitlab 살펴보기

3. 확장성과 가독성을 겸비한 빌드 파이프라인

Optimize GitLab CI/CD configuration files

Tier: Free, Premium, Ultimate

Offering: GitLab.com, GitLab Self-Managed, GitLab Dedicated

You can reduce complexity and duplicated configuration in your GitLab CI/CD configuration files by using:

- YAML-specific features like [anchors \(&\)](#), aliases (*), and map merging (<<). Read more about the various [YAML features](#) ↗.
- The `extends` keyword, which is more flexible and readable. You should use `extends` where possible.

Gitlab Runner를 구성해보자!

Executors

Tier: Free, Premium, Ultimate

Offering: GitLab.com, GitLab Self-Managed, GitLab Dedicated

GitLab Runner implements different executors that can be used to run your builds in different environments.

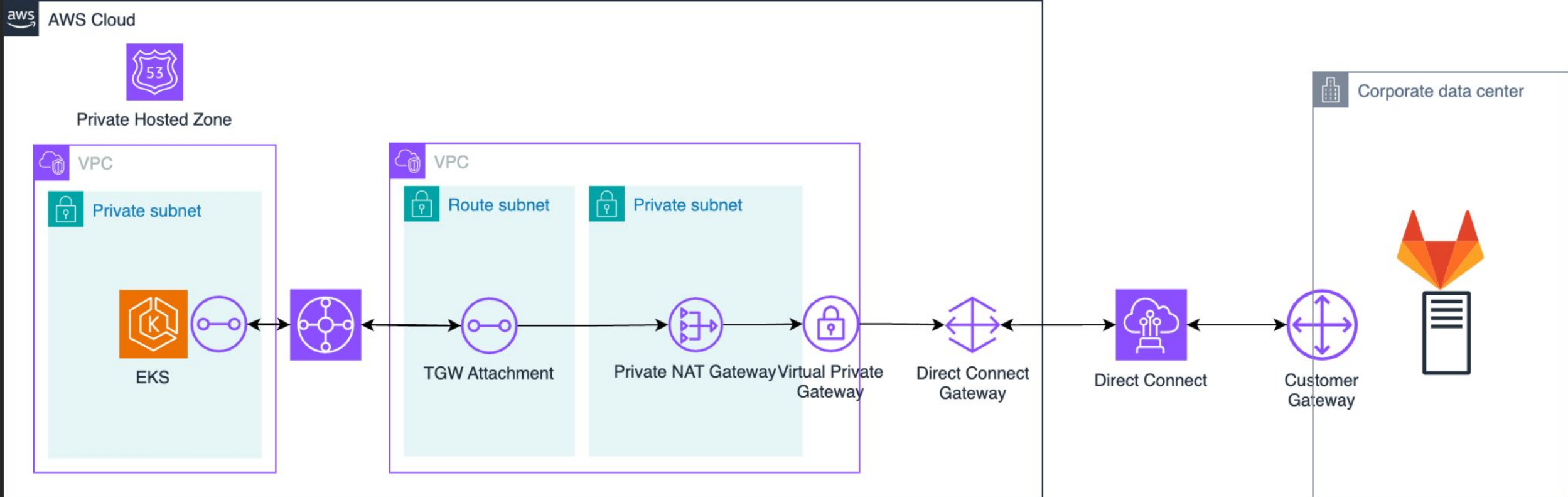
If you are not sure about which executor to select, see [Selecting the executor](#).

For more information about features supported by each executor, see the [compatibility chart](#).

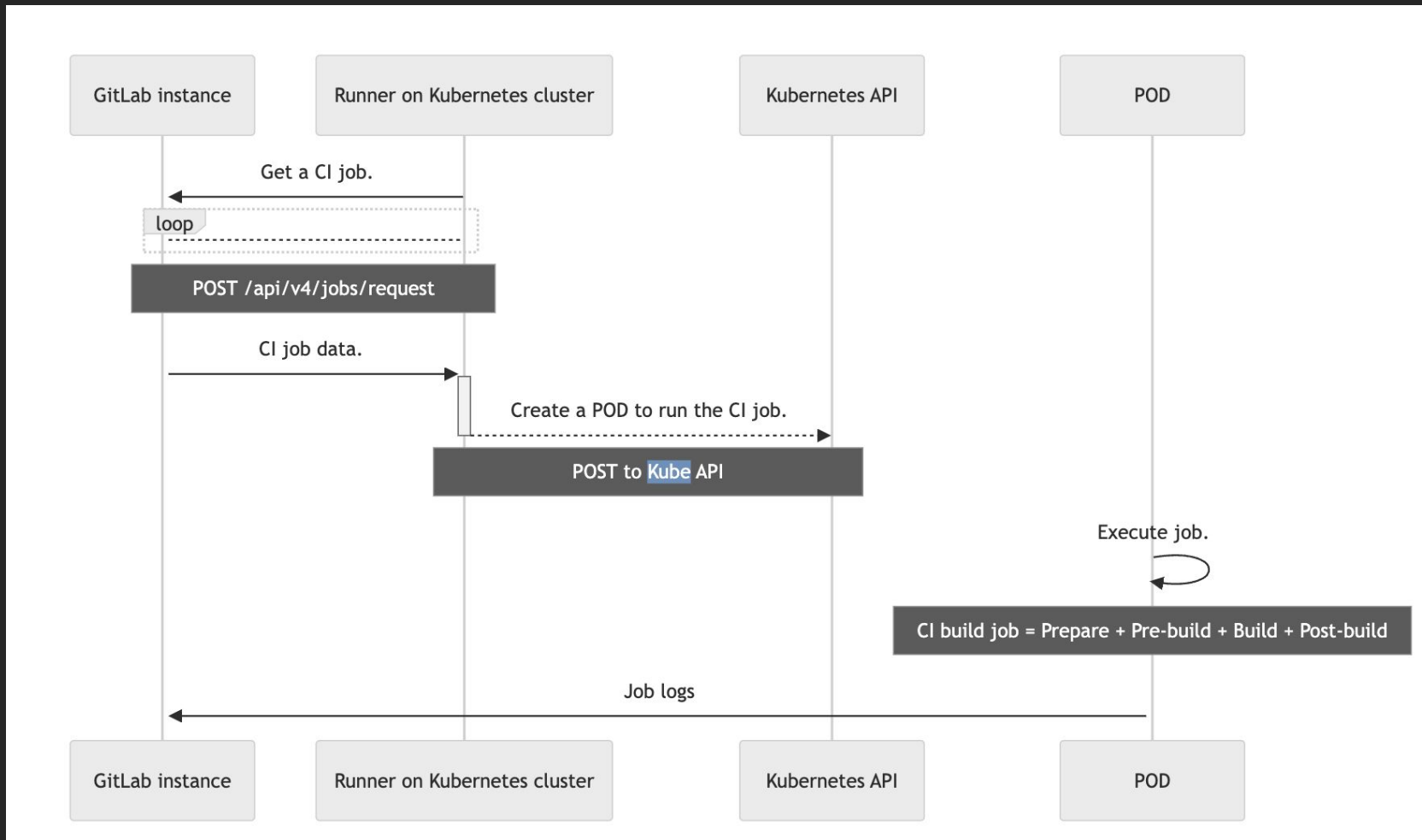
GitLab Runner provides the following executors:

- [SSH](#)
- [Shell](#)
- [Parallels](#)
- [VirtualBox](#)
- [Docker](#)
- [Docker Autoscaler](#)
- [Docker Machine \(auto-scaling\)](#)
- [Kubernetes](#)
- [Instance](#)
- [Custom](#)

Gitlab Runner on EKS 구성



Gitlab Runner on EKS 동작과정



이게 끝? ㅋㅋ 최적화

- SonarQube를 통한 정적 코드 분석
- 불필요한 테스트 Phase 제거
- Maven 성능 최적화

SonarQube를 통한 정적 코드 분석

기존(SI할 때)는 SonarQube의 CQ(코드 품질)로 개발사에게
Challenge

- 그럼 제거해도 되나? No, 코드 품질 유지가 안 될거 같음
- 현재는 SI가 끝났기 때문에 현재 코드 품질만 봐도 되지 않을까?
- 그럼 Daliy로 심야 시간에 돌리자!
- Schedule Job으로 풀어보자!

SonarQube를 통한 정적 코드 분석

Run jobs for scheduled pipelines

You can configure a job to be executed only when the pipeline has been scheduled. For example:

```
job:on-schedule:
  rules:
    - if: $CI_PIPELINE_SOURCE == "schedule"
  script:
    - make world

job:
  rules:
    - if: $CI_PIPELINE_SOURCE == "push"
  script:
    - make build
```

In this example, `make world` runs in scheduled pipelines, and `make build` runs in branch and tag pipelines.

Edit Pipeline Schedule

Description

SonarQube

Interval Pattern

- ☐ Every day (at 12:03pm)
☐ Every week (Tuesday at 12:03pm)
☐ Every month (Day 20 at 12:03pm)
☒ Custom

10 0 * * *

Set a custom interval with Cron syntax. [What is Cron syntax?](#)

Cron timezone

[UTC+9] Seoul

Select target branch or tag

develop

Variables

Variable

Input variable key

Input variable value

☐ Activated

Save changes

Cancel

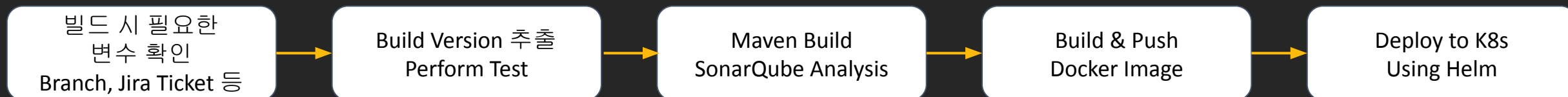
불필요한 테스트 Phase 제거

과연 이 테스트 코드가 의미가 있을까?

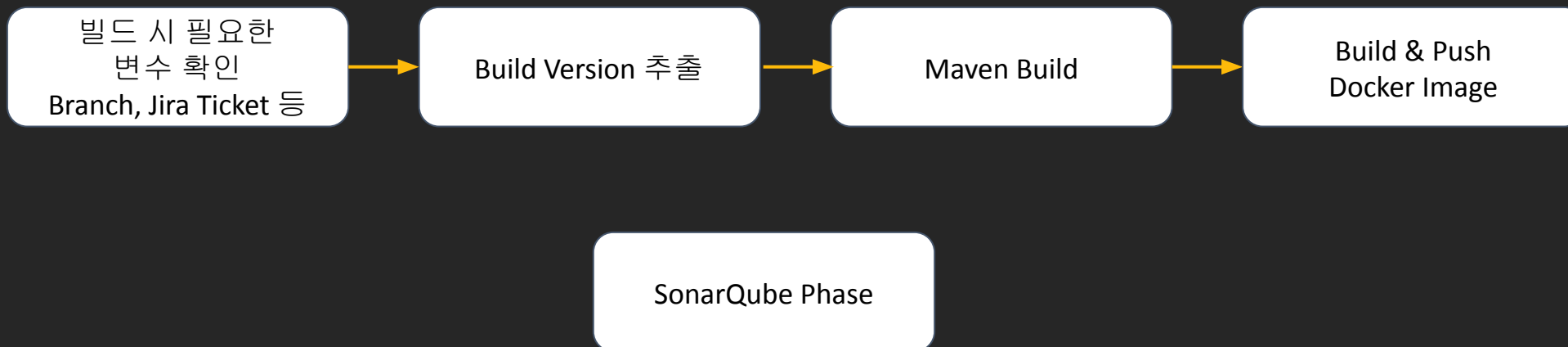
- Legacy 함수에 대한 테스트, 중복된 Test도 많음
- 어떤 서비스는 Test를 하고 어떤 서비스는 안함
- 정말 신뢰할 수 있는 테스트 코드인가? No
- 그럼 제거 하는게 맞지 않을까?

불필요한 테스트 Phase 제거

기존 빌드 파이프라인 (CD까지 이어짐)



변경 빌드 파이프라인 (CI만 하고, CD는 ArgoCD)



Maven 성능 최적화

Pod(Executor)에서 Maven을 통해서 빌드할 때 더 빠르게 하는 방법

→ Concurrent Option + Cache 적용

→ Cache는 local로 하면 날아가니까 AWS Native 기능 뭐 없나..?

→ S3를 활용해보자!

→ 그럼 Pod Identity를 활용해볼까!

Maven 성능 최적화

Maven Concurrent Build

Parallel builds in Maven 3

Created by Kristian Rosenvold, last modified by Benjamin Marwell on Dec 06, 2024

Overview

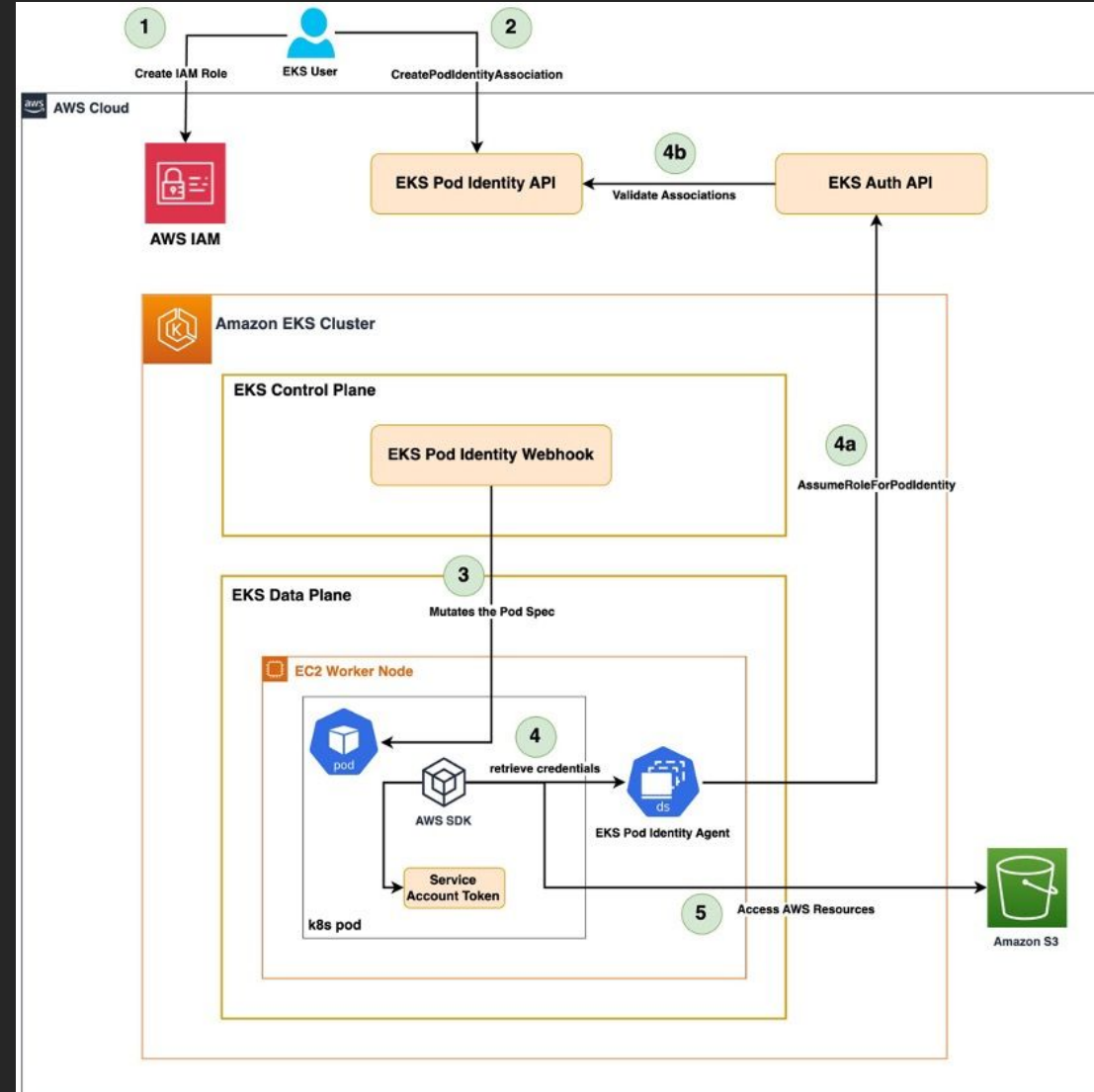
Maven 3.x has the capability to perform parallel builds. The command is as follows:

```
mvn -T 4 clean install # Builds with 4 threads  
mvn -T 1C clean install # 1 thread per cpu core  
mvn -T 1.5C clean install # 1.5 thread per cpu core
```

This build-mode analyzes your project's dependency graph and schedules modules that can be built in parallel according to the dependency graph.

The above mentioned thread per cpu core means [the number of cores is used as a multiplier](#)

EKS Pod Identity



Maven 성능 최적화

S3 Cache 설정

Example:

```
[runners.cache]
  Type = "s3"
  Path = "path/to/prefix"
  Shared = false
  [runners.cache.s3]
    ServerAddress = "s3.amazonaws.com"
    AccessKey = "AWS_S3_ACCESS_KEY"
    SecretKey = "AWS_S3_SECRET_KEY"
    BucketName = "runners-cache"
    BucketLocation = "eu-west-1"
    Insecure = false
    ServerSideEncryption = "KMS"
    ServerSideEncryptionKeyID = "alias/my-key"
```



If any of `ServerAddress`, `AccessKey` or `SecretKey` aren't specified and `AuthenticationType` is not provided, the S3 client uses the IAM instance profile available to the `gitlab-runner` instance. In an `autoscale` configuration, this is not the on-demand machine that jobs are executed on. If `ServerAddress`, `AccessKey` and `SecretKey` are all specified but `AuthenticationType` is not provided, `access-key` is used as the authentication type.

When you use Helm charts to install GitLab Runner, and `rbac.create` is set to true in the `values.yaml` file, a ServiceAccount is created. This ServiceAccount's annotations are retrieved from the `rbac.serviceAccountAnnotations` section.

For runners on Amazon EKS, you can specify an IAM role to assign to the service account. The specific annotation needed is: `eks.amazonaws.com/role-arn: arn:aws:iam::<ACCOUNT_ID>:role/<IAM_ROLE_NAME>`.

Maven 성능 최적화

Test GitLab Runner and GitLab CI on AWS EKS with AWS pod identity

 Open  Issue created 8 months ago by **Darren Eastman**

Overview

Amazon released [AWS EKS Pod identity](#) and customers using AWS to host runners are beginning to adopt this new feature.

However, in one case after enabling AWS Pod Identity a customer found an error with using S3 buckets.

```
ERROR: error while generating S3 pre-signed URL error=uri host is not a loopback address: http://169.254.170.23/
```

This came up in a [zendesk ticket](#) - internal only

Scope

- ☐ Test GitLab Runner on Kubernetes with AWS Pod Identify

Deliverables

- ☐ Update runner documentation as needed to include how to setup and use AWS EKS Pod identity.

0 of 2 checklist items completed · Edited 8 months ago by Christopher Mutua

Maven 성능 최적화

Upgrade github.com/minio/minio-go to v7.0.70

 Merged Mathieu Quesnel requested to merge  xmath279/gitlab-runner:U... into  main 8 months ago

Overview **6** Commits **2** Pipelines **5** Changes **2**

☐ Please check this box if this contribution uses AI-generated content (including content generated by GitLab Duo features) as outlined in the [GitLab DCO & CLA](#)

What does this MR do?

Upgrades minio library.

Why was this MR needed?

Adds support for EKS Pod Identities credentials

What's the best way to test this MR?

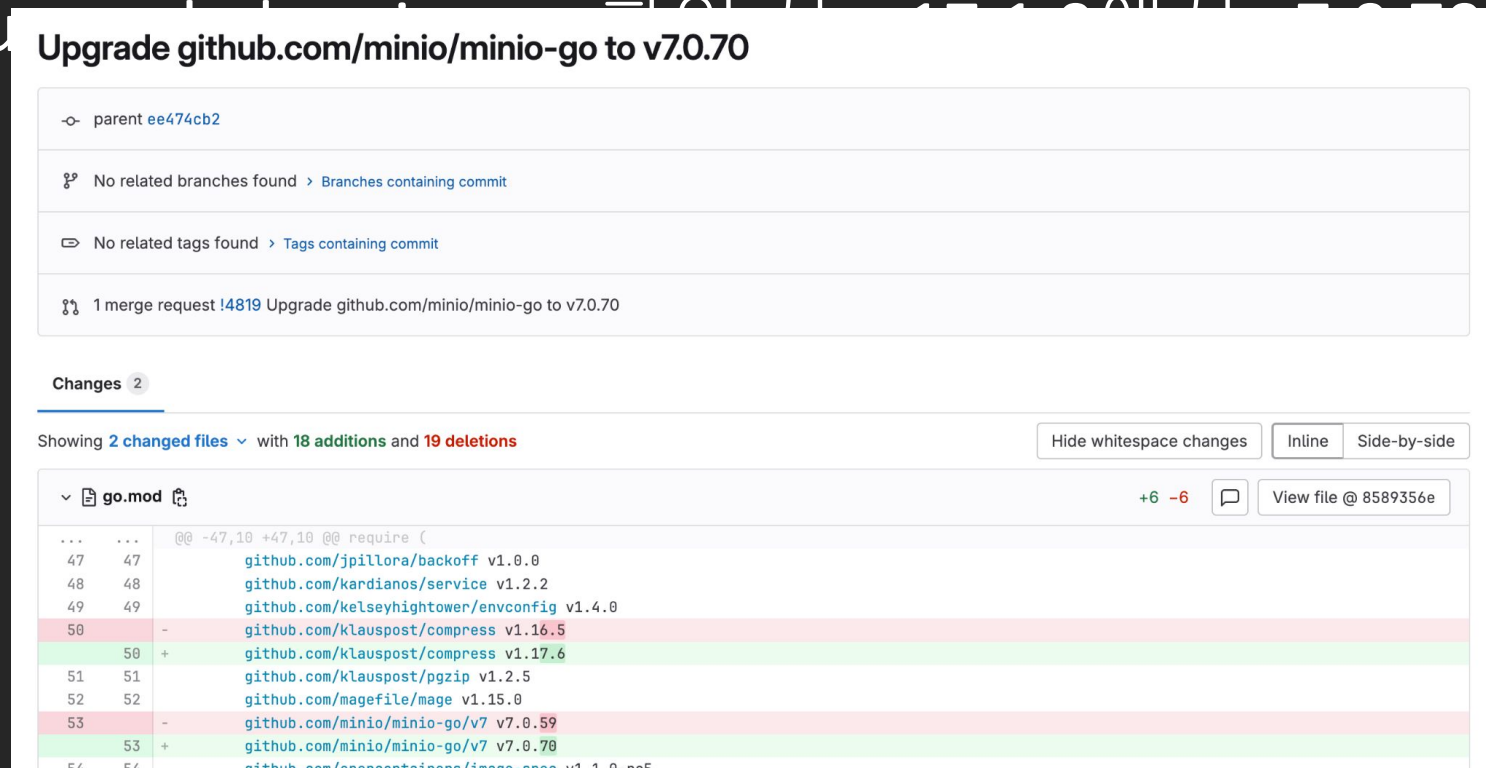
What are the relevant issue numbers?

[#37795](#)

Maven 성능 최적화

따라서 EKS Pod Identity를 적용하기 위해서는 Minio-go 버전 v7.0.69 이상

→ gitlab-runner를 사용하여 Minio-go v7.0.70 적용





4. Result

그래서 얼마나 좋아졌을까?

결과론적으로...

On-Prem Server 에서 돌렸었기 때문에 나가지 않던 Node + Network 사용료 발생

하지만, 성능/생산성은 향상됐습니다.

개발자들의 습관

본인들이 올린 MR로 빌드가 제대로 안되면, Gitlab에서
바로 확인 가능

→ Branch Condition과 Merge Request로 Build도 되기 때문에
Commit History를 이전보다 관리하게 쉽게 변경

빌드 속도의 향상

기존 파이프라인은 배포 시간 제외, 1시간 이상 걸림

→ 변경된 파이프라인에서는 빌드만 진행 + Phase 분리를 통해
최대 15분

→ CI 단계만 최대 75% 이상 단축

관리가 쉬운 CI 파이프라인

`include`

Use `include` to include external YAML files in your CI/CD configuration. You can split one long `.gitlab-ci.yml` file into multiple files to increase readability, or reduce duplication of the same configuration in multiple places.

You can also store template files in a central repository and include them in projects.

The `include` files are:

- Merged with those in the `.gitlab-ci.yml` file.
- Always evaluated first and then merged with the content of the `.gitlab-ci.yml` file, regardless of the position of the `include` keyword.

The time limit to resolve all files is 30 seconds.

앞으로 남은 부분

1. 시간대에 따른 HPA 조정(완료)

→ Spot Instance로 띄우지만, CronJob을 통해 심야 시간에 HPA 0

2. Runner 모니터링을 통해 Node Type 최적화(예정)

→ 12개의 Group안에는 40개 이상의 Project에 대한 Executor 동작

→ 따라서 모니터링을 통해 리소스 최적화로 Instance Spec

재산정



감사합니다!

QnA는 지금 해주셔도 좋고,
LinkedIn을 통해 추후에 주셔도 좋습니다.