



# 한 발자국 내딛기

---

정주리(무신사)

# 발표 주제

## 무신사 SRE팀이 Platform Engineering에 한 발자국 내딛게 된 이야기

### 이야기 하는 것

- 무신사 SRE팀이 일하는 방식
- 회사가 커지면서 벌어진 문제, 그리고 그걸 해결하기 위한 고민들
- 그래서 어떤 일을 했나?

### 이야기 하지 않는 것

- 디테일한 기술 이야기

# 무신사?

- 온라인 패션 커머스
- 프리챌 “무진장 신발 사진이 많은 곳”
- 일본기업 X
- 2003년 설립
- 무신사 스토어는 2009년 시작

The MUSINSA logo is displayed in white, bold, uppercase letters on a solid black rectangular background.

MUSINSA

무신사?

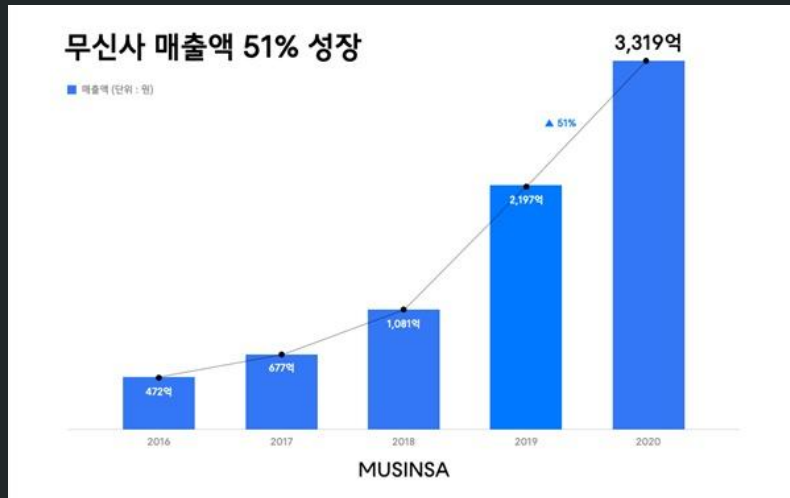
# 블랙 프라이데이

무신사, '무진장 블프' 판매액 3083억 돌파

2023.12.04

# 때는 바야흐로 2021년..

- 회사가 수직 성장 시작
- 개발 조직 확대
- AWS 이전 완료



# 눈에 보이기 시작한 문제들

- 개발 조직이 커지기 시작함
  - SRE가 응대해야 하는 사람의 수도 늘어남
- 신규 서비스가 마구마구 생겨남
  - 인프라를 일일이 손으로 만들어서는 하루 종일 인프라 프로비저닝만 해야 함

# 신규 인프라 생성의 고통

- 신규 서비스가 태어나려면, 개발/알파/운영 환경의 인프라 프로비저닝이 필요
- AWS 리소스는 Terraform으로, EKS 리소스는 2개의 저장소에서 yaml로 관리
- 일일이 수동으로 코드를 추가하면... 보통 3일 정도 소요됨
- 실제로는 장애대응, 문의대응, 그 외의 업무와 병행하면 더 늦어짐

앞으로 서비스는 더 커질텐데...



# 신규 인프라 생성의 고통

코드의 양을 줄이자!

Terraform > Terragrunt

> 하나의 인프라 패키지

= 1개의 terragrunt.hcl 파일

```
resource "aws_iam_policy" "devel_ranking_api_policy" {
  name = "devel_ranking_api_policy"
  assume_policy = jsonencode({
    "Statement": [
      {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Resource": "arn:aws:iam::${var.devel_aws_account_id}:role/${var.devel_ranking_api_role_name}"
      }
    ]
  })
  managed_by = "terraform"
}

resource "aws_iam_policy_attachment" "devel_ranking_api_policy_attachment" {
  name = "aws_iam_policy_attachment_devel_ranking_api_policy"
  policy_arn = aws_iam_policy.devel_ranking_api_policy.arn
  role = aws_iam_role.devel_ranking_api_role.name
}

resource "aws_lb_target_group" "devel_ranking_web_tg_ip" {
  name = "devel-ranking-web-tg"
  port = 80
  protocol = "HTTP"
  protocol_version = "HTTP1"
  target_type = "ip"
  vpc_id = var.devel_vpc_id
  deregistration_delay = 120
  slow_start = 30

  health_check {
    interval = 5
    path = "/search/monitor/healthcheck"
    healthy_threshold = 2
    unhealthy_threshold = 2
    protocol = "HTTP"
    timeout = 3
    matcher = "200"
  }
}

tags = {
  Name = "devel-ranking-web-tg"
  Environment = "Devel"
  Stage = "Devel"
  Project = "Ranking"
  Domain = "Ranking"
  Managed_by = "terraform"
}

resource "aws_security_group" "devel_ranking_api_sg" {
  name = "devel-ranking-api-devel-sg"
  description = "devel-ranking-api-devel-sg"
  vpc_id = var.devel_vpc_id

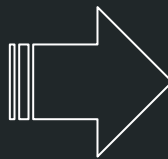
  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = [
      "0.0.0.0/0"
    ]
  }

  tags = {
    Name = "devel-ranking-api-devel-sg"
    Managed_by = "terraform"
    Environment = "Devel"
  }

  resource "aws_elastic_load_balancing_target_group" "devel_eks_cluster_ranking_api_target_group" {
    name = "devel_ranking_api_target_group"
    port = 80
    protocol = "HTTP"
    vpc_id = var.devel_vpc_id
    deregistration_delay = 120
    slow_start = 30
    health_check {
      interval = 5
      path = "/search/monitor/healthcheck"
      healthy_threshold = 2
      unhealthy_threshold = 2
      protocol = "HTTP"
      timeout = 3
      matcher = "200"
    }
  }

  resource "aws_elastic_load_balancing_target_group_attachment" "devel_eks_cluster_ranking_api_target_group_attachment" {
    target_group_arn = aws_elastic_load_balancing_target_group.devel_eks_cluster_ranking_api_target_group.arn
    instance_id = aws_instance.devel_ranking_api_instance.id
  }

  resource "aws_iam_policy" "devel_ranking_api_policy" {
    name = "devel_ranking_api_policy"
    description = "devel_ranking_api_policy"
    policy = jsonencode({
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::${var.devel_aws_account_id}:role/${var.devel_ranking_api_role_name}"
        }
      ]
    })
    managed_by = "terraform"
  }
```



```
resource "aws_lb_target_group" "devel_ranking_web_tg_ip" {
  name = "devel-ranking-web-tg"
  port = 80
  protocol = "HTTP"
  protocol_version = "HTTP1"
  target_type = "ip"
  vpc_id = var.devel_vpc_id
  deregistration_delay = 120
  slow_start = 30

  health_check {
    interval = 5
    path = "/search/monitor/healthcheck"
    healthy_threshold = 2
    unhealthy_threshold = 2
    protocol = "HTTP"
    timeout = 3
    matcher = "200"
  }
}

tags = {
  Name = "devel-ranking-web-tg"
  Environment = "Devel"
  Stage = "Devel"
  Project = "Ranking"
  Domain = "Ranking"
  Managed_by = "terraform"
}

resource "aws_security_group" "devel_ranking_api_sg" {
  name = "devel-ranking-api-devel-sg"
  description = "devel-ranking-api-devel-sg"
  vpc_id = var.devel_vpc_id

  ingress {
    from_port = 80
    to_port = 80
    protocol = "tcp"
    cidr_blocks = [
      "0.0.0.0/0"
    ]
  }

  tags = {
    Name = "devel-ranking-api-devel-sg"
    Managed_by = "terraform"
    Environment = "Devel"
  }

  resource "aws_elastic_load_balancing_target_group" "devel_eks_cluster_ranking_api_target_group" {
    name = "devel_ranking_api_target_group"
    port = 80
    protocol = "HTTP"
    vpc_id = var.devel_vpc_id
    deregistration_delay = 120
    slow_start = 30
    health_check {
      interval = 5
      path = "/search/monitor/healthcheck"
      healthy_threshold = 2
      unhealthy_threshold = 2
      protocol = "HTTP"
      timeout = 3
      matcher = "200"
    }
  }

  resource "aws_elastic_load_balancing_target_group_attachment" "devel_eks_cluster_ranking_api_target_group_attachment" {
    target_group_arn = aws_elastic_load_balancing_target_group.devel_eks_cluster_ranking_api_target_group.arn
    instance_id = aws_instance.devel_ranking_api_instance.id
  }

  resource "aws_iam_policy" "devel_ranking_api_policy" {
    name = "devel_ranking_api_policy"
    description = "devel_ranking_api_policy"
    policy = jsonencode({
      "Statement": [
        {
          "Action": "sts:AssumeRole",
          "Effect": "Allow",
          "Resource": "arn:aws:iam::${var.devel_aws_account_id}:role/${var.devel_ranking_api_role_name}"
        }
      ]
    })
    managed_by = "terraform"
  }
```



신규 인프라 생성의 고통

그래도 많다!



# 인프라 생성 작업



자동화!



권한 및 접근제어 수정

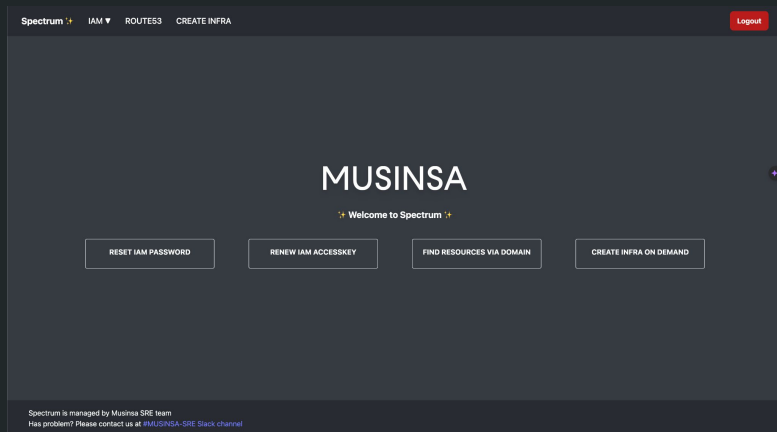
Terragrunt  
(AWS)

K8s  
(SRE)

# 신규 인프라 자동 생성

## LaunchPad(on Spectrum)

- v1(2021), v2(2023) 한 달 정도 급조개발
- AWS 인프라를 위한 Terragrunt 코드 및 인프라 자동생성, EKS YAML 자동 생성, 배포 파이프라인 자동 생성
- 모든 생성 인프라는 IaC로 관리되어 GitHub에 Push



# 신규 인프라 자동 생성



어? 어디서 많이 본 것 같은데...?

## Platform Engineering으로 가는 길

처음부터 Platform Engineering을 추구하지 않았으나...

개발조직은 커지고, 모두의 인프라를 관리해야 한다면,

자동화는 필수적인 요소

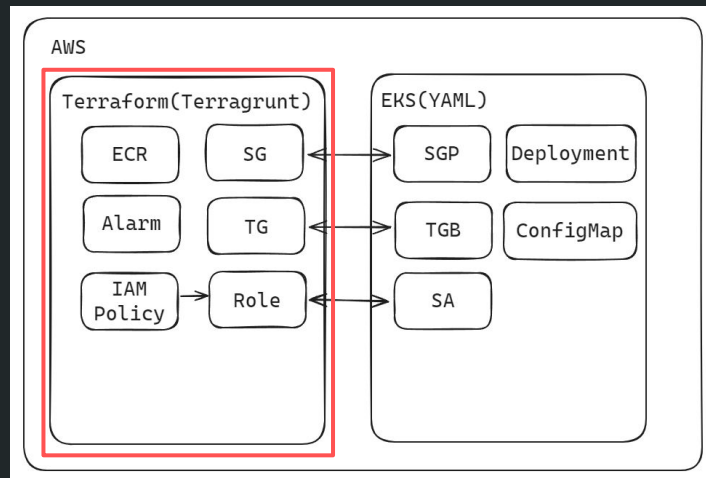
자동화된 도구를 사용해 개발자가 잘 관리되는 인프라를 Self-Service

Platform Engineering

# 신규 인프라 자동 생성

## AWS 인프라(Terragrunt 코드) 자동 생성

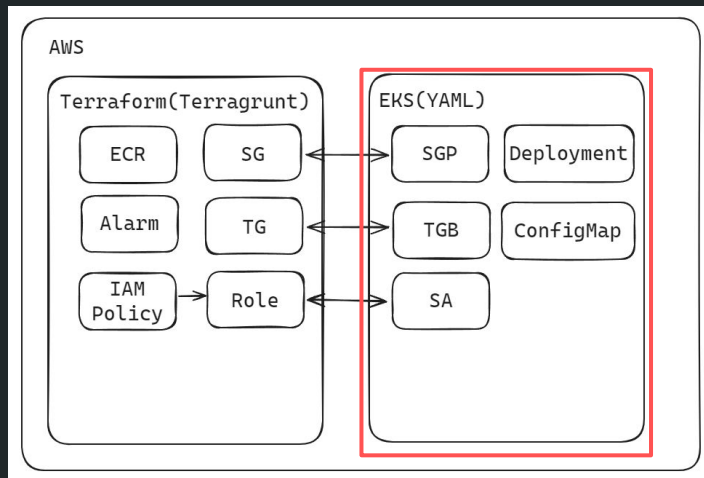
- 결국 Terragrunt 코드를 생성 후 worker에서 직접 cli로 apply까지 함
- 불안정함...  
cli 실행의 실패 원인을 알기 어려움



# 신규 인프라 자동 생성

## EKS Manifest(YAML) 자동 생성

- 개발팀에게 직접 수정할 수 있도록 서비스별 저장소마다 YAML 제공
- 생 YAML을 직접 생성
  - => 장점: 자유도 높은 설정
  - => 단점: 기존 Manifest의 일괄 작업이 어렵다



# 신규 인프라 자동 생성

## EKS Manifest(YAML) 자동 생성

- 개발팀에게 직접 수정할 수 있도록 서비스별 저장소마다 YAML 제공
- 대부분의 개발자들은 건드리고 싶지 않아함...
- SRE가 봐야 할 부분, 개발자가 봐야 할 부분이 섞여 있어 혼란
- Helm으로 만들어 SRE가 관리하는 부분을 분리하고, 개발자에게 values만 관리하도록 할 예정

deployment.yaml

filebeat-configmap.yaml

nginx-configmap.yaml

rbac.yaml

sgp.yaml

svc.yaml

tgb.yaml

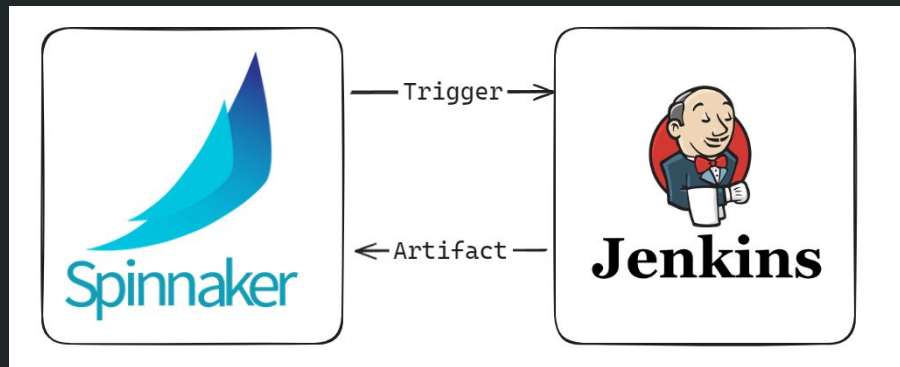


# 신규 인프라 자동 생성

## 배포 파이프라인 자동 생성

v1: Jenkins+Spinnaker

Spinnaker 파이프라인을 자동생성



- 하지만 팀이 생길 때마다 Jenkins를 만들어 달라고 요청함(관리도 SRE가..)
- 인프라 생성보다 Jenkins 생성이 더 오래 걸림
- Spinnaker에 Jenkins를 추가할 때마다 재시작
- Jenkins가 망가져도 SRE 출동(특: 자주 망가짐)
- Spinnaker 설정과 파이프라인은 IaC가 매우 어렵다
- Spinnaker는 업데이트를 할 때 마다 서비스 하나는 꼭 말썽을 일으키는데...

# 신규 인프라 자동 생성

## 배포 파이프라인 자동 생성

v2: ArgoCD

ArgoCD Application을 코드로 자동생성

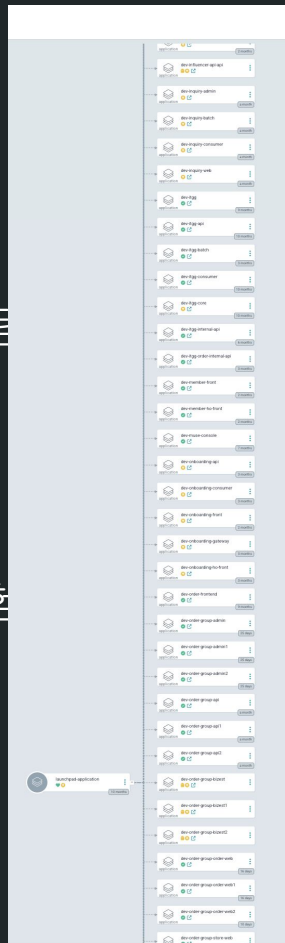
- 와! GitOps!
- Spinnaker 보다 훨씬 납득 가능한 API
- 배포 파이프라인도 IaC가 가능!
- 배포 전략에 대한 고민이 부족했음



# 신규 인프라 자동 생성

배포 전략에 대한 고민이 부족했음

- 하나의 Argo ApplicationSet에 모든 서비스의 Application이 등록 됨
- 등록된 서비스가 130개가 넘어감
- 스크롤이 끝나지 않아요
- 개발자가 실수로 다른 서비스의 Application을 지움(!)
- 모든 환경에 같은 토큰을 쓰는 바람에 GitHub API Rate Limit이 위험



# 다음 목표

## As-is

- 생성 도중 실패 시, 재시도가 어려움
- 삭제 기능이 없다(!)
- UI를 만들다 말았음(마치라익 지금의 PPT)
- 문서도 쓰다 말았음
- 결국 내가 만들다 만 탓

위와 같은 이유로 인해 자동화는 되었지만, 아직은 SRE가 직접 LaunchPad를 조작해 인프라를 Delivery하고 있음

# 다음 목표

## To-be

개발자가 자유롭게 임시로 인프라를 생성해보고, 자신이 만든 컨테이너를 시험해 본 뒤 접근제어와 권한 부여와 같은 디테일한 작업만 SRE의 손을 탈 수 있도록

**SRE는 중앙 관리만, 각 인프라는 개발팀의 Self Service!**

그러지 않으면 우리는 과로를 피할 수 없을 것이다

# 마치며

여기까지가 개발자의 업무! 여기까지가 SRE의 업무! 라고 선을 긋고 싶지는 않음

하지만 조직은 커졌고 우리는 덜 커졌음. 한계가 왔다.(꽤 오래 전에)

여러분들이 아는 좋은 툴, 좋은 아이디어, 좋은 고민, 그리고 경험한 모든 것이 궁금합니다!

마치며

감사합니다!

