Red Hat

Ansible을 통한 효율적인 자동화

목차

www.facebook.com/redhatkorea
구매문의 02-6105-4390
buy-kr@redhat.com

면책 조항

고객 사례를 기반으로 이 가이드는 작성자들 개개인의 개별 의견만을 기술하였습니다. 이 가이드에 제시된 정보는 어떠한 보증도 없이 '있는 그대로' 제공됩니다. 이 가이드를 읽는 시점에는 최신 내용이 아닐 수 있습니다. 작성자는 이 정보와 관련된 어떠한 책임도 지지 않습니다. 작성자들은 이 가이드가 일반적으로 도움이 될 수 있다고 ***생각***합니다.

버전 기록

버전	날짜	상세 정보	작성자 및 기여자
1.0	2023년 10월 12일	첫 공개 버전	**작성자:** 니콜라스 웡(Nicholas Wong), 기네시 마다파람바스(Gineesh Madapparambath), 마이클 크냐제프(Michael Knyazev), **기여자:** 도나 벤자민(Donna Benjamin), 제레미 베나즈라(Jeremie Benazra), 데니스 미칼킨(Denis Mikhalkin), 프레드릭 손(Frederick Son), 발렌타인 슈워츠(Valentine Schwartz)

1. 소개

고객 사례를 기반으로 한 이 가이드는 자동화 전문가와 실제로 자동화를 활용하려는 이들로 구성된 광범위한 커뮤니티를 지원하기 위해 작성되었습니다. DevOps 엔지니어와 아키텍트, 제품 소유자, IT 및 운영 관리자가 주요 대상입니다. 이 문서를 읽은 바로 다음 날부터 본문에 언급된 리소스와 새로운 지식을 사용할 수 있도록 내용을 구성하였습니다. 이 가이드는 실용적인 스타일입니다. 각 장마다 해당 사례에 대한 소개와 집중적인 권장 사항을 시작부터 최소 기능 제품(MVP) 제공까지 진행되는 실제 자동화 "샘플 프로젝트"와 번갈아 가며 제시합니다. 가능한 MVP 구현은 MIT 라이센스에 따라 게시된 다음 개인 리포지토리에서 찾을 수 있습니다.[1]
https://github.com/mikhailknyazev/automation-samples

이 가이드를 통해 누구나 자신의 환경과 관련된 고유한 요소를 찾을 수 있습니다. 이 가이드만의 장점은 엔지니어링, 제품 및 애자일 실무자가 업무 윤리를 지키고 고객에게 가치 있는 결과를 제공하면서, 체계적으로 자동화 프로젝트를 함께 진행하는 과정을 잘 보여주는 데 있습니다.

Red Hat에서 제공하는 관련 오퍼링은 마지막에 나온 "Red Hat의 고객 지원 사항은 무엇인가요?"라는 특별 섹션에 정리되어 있습니다.

[1] 가능한 MVP 구현(MIT 라이센스에 따라 게시된 개인 리포지토리) https://github.com/mikhailknyazev/automation-samples

2. 자동화를 위한 공통 언어로서의 Ansible

복잡한 IT 환경에서 오케스트레이션의 필요성은 늘 제기되어 왔으며, 이미 많은 에코시스템에서 자체 오케스트레이터를 사용하고 있습니다. OpenStack의 Heat, Amazon의 CloudFormation, Azure Resource Manager, Jenkins 또는 HashiCorp의 Packer, Terraform과 같은 툴은 모두 태스크를 오케스트레이션하고 "코드형 인프라(IaC)" 접근 방식을 실현하기 위한 것입니다.

하지만 고객이 단 하나의 환경으로 오케스트레이션을 제한할 수 있는 가능성은 얼마나 될까요? 여기서 Ansible®이 등장합니다. Ansible의 모듈 라이브러리와 손쉬운 확장성을 활용하면 하나의 구조화된 Ansible 구문(Syntax)을 사용하여 다양한 환경에서 다양한 "관리자"를 간단하게 오케스트레이션할 수 있습니다. 새로운 스크립트 기능을 구현하는 데 무엇을 사용할지 선택할 때, Ansible은 "베어" Bash/PowerShell/Python 스크립팅을 효율적으로 대체할 때가 많습니다.

Ansible은 고객의 기존 SSH 및 WinRM 인프라와 함께 작동하므로 추가로 열어야 할 네트워크 포트가 없습니다. Ansible은 조직의 IT 아티팩트를 연결하는 범용 "접착제" 역할을 합니다. 컴퓨팅 노드, 스토리지 기기, 네트워크, 로드 밸런서, 모니터링 시스템, 웹 서비스 및 기타 기기와 함께 작동할 수 있습니다. 예를 들어 부하 분산 풀에서 서버를 추가하거나 제거하고 업데이트 중인 각 머신에 대한 모니터링 알림을 비활성화할 수 있습니다.

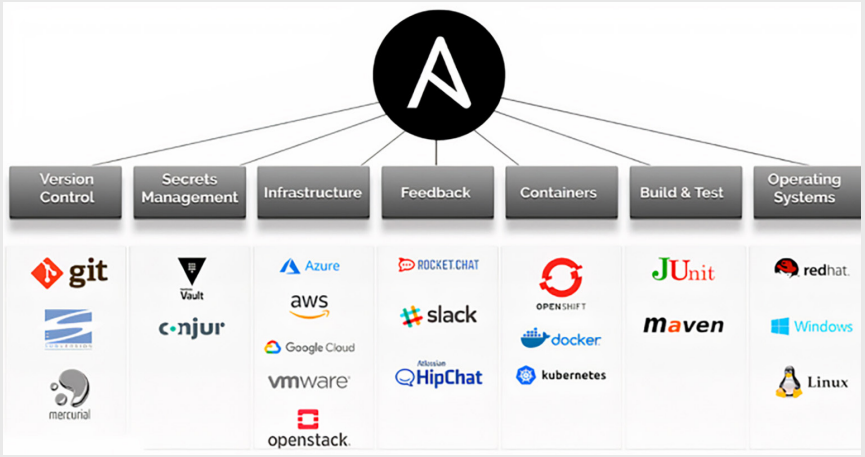


그림 출처: "Ansible을 사용한 업계의 자동화 활용 사례"[2]

Ansible 기반 인프라 프로비저닝, 배포 및 테스트는 물론 Red Hat® OpenShift® 및 쿠버네티스를 위한 운영 자동화 등 실제 프로젝트에서 수많은 사례를 확인할 수 있습니다. 원하는 대로 자동화를 트리거하기 위해 Event-Driven Ansible(EDA)에 쿠버네티스를 이벤트 소스로 추가할 수도 있습니다.[3] Red Hat OpenShift에 Ansible Automation Platform을 배포할 수 있습니다.[4]

다음은 고객의 조직에 적용할 수 있는 자동화 아이디어의 몇 가지 예입니다.

- 일관된 시스템 설정을 보장하여 드리프트, 중단 및 다운타임 감소
- 운영 태스크를 자동화하여 인적 오류 최소화
- 여러 도메인에 걸쳐 기술 요구 사항 감소
- 시스템에서 필요한 기준을 설정 및 유지하여 컴플라이언스 충족
- 패치 관리를 통해 보안 침해 및 기타 보안 관련 인시던트 위험 감소

[2] Ansible을 사용한 업계의 자동화 활용 사례(Industry use cases on automation using Ansible) https://www.linkedin.com/pulse/industry-use-cases-automation-using-ansible-anand-kumar/

[3] "쿠버네티스와 Event-Driven Ansible의 결합(Kubernetes meets Event-Driven Ansible)" https://www.ansible.com/blog/kubernetes-meets-event-driven-ansible

[4] "새로운 표준 아키텍처: Red Hat OpenShift에 Ansible Automation Platform 2 배포(New reference architecture: Deploying Ansible Automation Platform 2 on Red Hat OpenShift)" https://www.ansible.com/blog/new-reference-architecture-deploying-ansible-automation-platform-2-on-red-hat-openshift

특히 엔터프라이즈 보안은 동종 엔터티가 아닙니다. 일반적으로 분산되어 사일로화된 경우가 많은 팀에서 운영하는 여러 벤더 솔루션의 포트폴리오입니다. 이렇게 다양한 계층이 존재할 때 자동화는 보안 운영 팀이 책임을 통합하고 공유하며 여러 보안 기술을 함께 연결하도록 지원하는 데 효과적이라는 사실이 입증되고 있습니다.

IT 팀원들은 매일 같은 태스크를 반복하는 경우가 많습니다. 예를 들면 다음과 같습니다.

- 시스템 엔지니어는 서버와 가상 머신을 구축하고, 패키지를 설치하며, 오래된 시스템에 패치를 적용하고, 방화벽 기기를 구성하는 등의 작업을 합니다.
- 개발자는 프로그래밍 언어나 소프트웨어 라이브러리의 새 버전이 릴리스될 때마다 코딩 환경을 구축하는 데 어려움을 겪습니다. 또한 코드를 테스트하고 테스트 결과를 기다리느라 많은 시간을 보냅니다.
- 데이터베이스 관리자가 디스크 공간을 프로비저닝하고 스토리지 기기를 구성하는 데 귀중한 시간을 할애하는 동안 새로운 데이터베이스 서버 프로비저닝이 지연되거나 네트워크/시스템 준비 상태에 문제가 발생합니다.
- 운영 팀은 쏟아지는 이벤트와 알림으로 인해 어려움을 겪으며 오탐을 걸러내는 데 시간을 쓰느라 생산성이 저하됩니다.
- 보안 팀은 위반 사항을 수정하고 시스템이 관련 보안 표준을 준수하는지 확인하느라 바쁩니다.

3. 자동화를 통한 효율적인 개선 계획

더 적은 노력으로 더 많은 가치를 창출하는 것이 목표라고 할 때, 자동화를 구현하기에 앞서 잠시 중요한 질문을 해보는 것이 좋습니다.

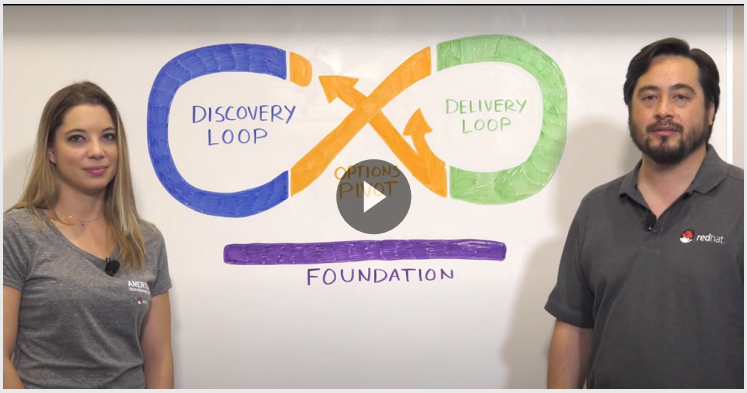
- 활용 사례가 얼마나 복잡한가?
- 인적 오류를 줄일 수 있는가?
- 배포 시간을 단축하고 태스크 속도를 높일 수 있는가?
- 이 태스크를 얼마나 자주 수행하는가?
- 이 태스크를 자동화하면 얼마나 많은 시간을 절약할 수 있는가?
- ROI(투자수익률)는 어떠한가? 비용을 절약할 수 있는가?

활용 사례를 찾아 자동화를 구현하는 것이 시스템 팀, 플랫폼 팀 또는 인프라 팀만의 책임이라 생각하는 것은 흔한 오해입니다. 업무 환경과 일상 태스크를 살펴보면 Ansible을 사용하여 자동화할 수 있는 태스크는 수천 가지입니다. 예를 들면 데이터베이스 서버와 인스턴스를 관리하는 데이터베이스 팀, 네트워크 운영을 처리하는 네트워크 팀, 애플리케이션 업데이트를 보다 효과적으로 배포하려는 애플리케이션 팀 등입니다. 환경 내 자동화 구현은 협업이 기반이 된 과정이므로 여러 팀의 지원과 안내가 필요합니다.

체계적인 방식으로 자동화를 도입하는 데 도움이 되는 잘 알려진 개방형 사례가 있습니다. 이 섹션에서는 다음과 같은 논리적 순서에 따라 사례와 예시를 소개하겠습니다.

- “검색 루프”
  - 동기 부여 매핑
  - 큰 그림 이벤트 스토밍
  - 도메인 스토리텔링
  - 메트릭 기반 프로세스 매핑(MBPM)
  - 사용자 스토리 매핑
  - 목표 성과
- “옵션 피벗” 단계
  - Kano 모델
  - 가치 분할
  - MVP

이 가이드에서 다룬 사례에 대한 더 큰 컨텍스트는 다음 짧은 동영상에서 설명합니다.



Open Practice Library로 성과 기반 제공[5]

Open Practice Library는 여기에서 확인할 수 있습니다.[6] https://openpracticelibrary.com
이 가이드에서 해당 섹션 등에 대한 많은 참고 자료를 찾을 수 있습니다.

다음 섹션에서는 다양한 전문성을 갖춘 여러 팀 구성원들이 서로의 지식을 공유하는 "구조화된 브레인스토밍" 접근 방식인 **이벤트 스토밍**을 소개합니다. 이는 자동화 컨텍스트의 큰 그림을 이해하는 데 필요합니다.

또한 이 가이드 전체에서 사용하는 **"샘플 프로젝트"**도 소개합니다.

3.1. 기존 환경 살펴보기

기술 전문가뿐만 아니라 비즈니스 담당자가 참여하는 종합적인 논의가 필요한 경우 가능한 한 원활한 커뮤니케이션이 이루어지게 해야 합니다. **이벤트 중심 포커스**는 비즈니스와 기술의 경계를 뛰어넘는 데 도움이 되는 것으로 나타났습니다. 해당 도메인과 관련된 모든 종류의 이벤트는 모든 당사자가 인식할 수 있습니다. 브레인스토밍을 할 때 늘 그렇듯이 정확성보다는 발전이 중요합니다.

3.1.1. 대표적인 이벤트 중심 사례

다음 섹션에서 설명하는 "**이벤트 스토밍**" 사례는 문제 영역을 검색하는 데 중점을 둡니다. 시스템이나 시간 기반 관점이 아닌 이벤트 중심 관점을 통합하는 또 다른 사례는 "이벤트 모델링"입니다.[7] 후자는 "명령 쿼리 책임 분리" 및 "이벤트 소싱"(CQRS 및 ES 설계 패턴) 개념을 기반으로 솔루션의 청사진을 만듭니다. 다시 말해, 이벤트 모델링은 일부 유형의 솔루션을 구축하는 데는 매우 효율적일 수 있지만 모두에게 적합한 것은 아닙니다.

다음은 "이벤트 스토밍"과 "이벤트 모델링"을 중심으로 한 이벤트 중심 사례 발전 과정을 요약한 것입니다.

[5] "Open Practice Library로 성과 기반 제공(Outcome-driven delivery with Open Practice Library)" https://www.youtube.com/watch?v=N4mBlZg8MnQ
[6] "Open Practice Library" https://openpracticelibrary.com/
[7] "이벤트 모델링(Event Modeling)" https://openpracticelibrary.com/practice/event-modeling/

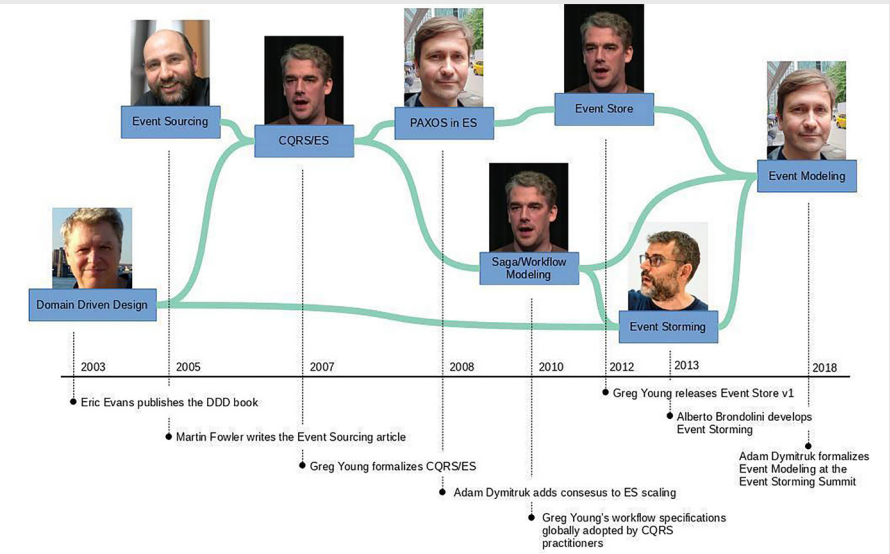


그림 출처: "이벤트 모델링 소개(About Event Modeling)"[8]

이 가이드에서는 "이벤트 스토밍" 사례에 중점을 둡니다. "이벤트 모델링"에 대한 자세한 내용은 작성자인 애덤 디미트룩(Adam Dymitruk)의 "심층 분석"인 "Event Modeling Deep Dive(이벤트 모델링 심층 분석)"가 도움이 될 것입니다.[9] 다음 이벤트 모델링 리소스도 유용합니다.

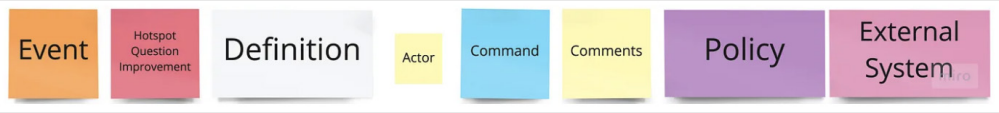
- 훌륭한 샘플 기반 설명[10]
- Miro 친화적인 리소스 일부는 여기에서 확인할 수 있습니다[11]
- "큰 그림 이벤트 스토밍"에서 "이벤트 모델링" 프로세스를 부트스트랩하는 방법[12]

3.1.1.1. "이벤트 스토밍" 검색 사례란?

이벤트 스토밍 워크샵에는 여러 유형이 있습니다. "큰 그림 이벤트 스토밍"은 일반적으로 비즈니스 도메인을 검색하고 지식을 공유하는 데 사용됩니다. "소프트웨어 디자인 이벤트 스토밍"은 시스템 설계에 더 중점을 둡니다.

"이벤트 스토밍" 사례는 게임스토밍[13]의 지원 그룹 학습 사례와 도메인 중심 설계(DDD) 원칙을 종합한 것입니다.

지식은 온라인 세션 중 가상으로 Miro[14]와 같은 툴을 사용해 스티커 메모, 또는 현장에서 8~10미터 높이의 벽이 있는 방에 붙은 여러 색상의 포스트 잇(클수록 좋음)에 작성되어 있습니다. 아래는 각각의 색상과 목적을 알 수 있는 그림입니다.



[8] "이벤트 모델링 소개(About Event Modeling)" https://eventmodeling.org/about/
[9] "본 버논과 애덤 디미트룩의 이벤트 모델링 심층 분석(Event Modeling deep dive with Vaughn Vernon and Adam Dymitruk)" https://www.youtube.com/watch?v=ufKgwjsD1l8
[10] "이벤트 모델링이란?(What is Event Modeling?), 예시 포함" https://www.goeleven.com/blog/event-modeling/
[11] "이벤트 모델링 치트 시트(Event Modeling cheat sheet)" https://eventmodeling.org/posts/event-modeling-cheatsheet/
[12] "이벤트 스토밍과 이벤트 모델링 비교, 라팔 맥시악, DDD 이스탄불(Event Storming vs Event Modeling by Rafal Maciag @DDD Istanbul)" https://youtu.be/LDPlvmD9upk?t=2274
[13] 위키피디아의 게임스토밍 항목(Gamestorming on Wikipedia) https://en.wikipedia.org/wiki/Gamestorming
[14] Miro https://miro.com/

여기서는 핵심 개념만 소개하며, 이 섹션에 있는 아래 링크를 따라가면 더 자세한 내용을 읽어보실 수 있습니다.

이벤트

모든 이벤트 스토밍은 비즈니스에 대한 지식을 수집하는 것으로 시작하여, 도메인 이벤트 형태로 표현되며, 왼쪽에서 오른쪽으로 시간순으로 (가상 또는 실제) 벽에 붙어 있습니다. 도메인 이벤트는 비즈니스에 명시적인 영향을 미치는 모든 중요 이벤트를 말합니다. 이벤트는 비즈니스 시스템 내부 또는 외부에서 발생할 수 있습니다. 관행상 각 이벤트는 주황색 스티커 메모에 적힌 과거형 동사로 표현됩니다.
예시: "예약이 요청됨", "예약이 수락됨", "여정 시작됨", "여정 완료됨".

커맨드

대부분의 도메인 지식이 벽에 붙어 있는 이벤트에 반영되어 있다고 생각되면 적절한 이벤트를 트리거하는 커맨드에 집중하세요. 비즈니스에 중요한 커맨드를 하늘색 메모지에 적고 해당 커맨드가 생성하는 이벤트 왼쪽에 배치하세요. 커맨드는 사용자의 의도를 나타내는 메시지로서, 작업으로 표현될 수 있습니다.
예시: "예약 요청", "예약 취소", "환불 요청".

정책

정책 아티팩트는 이벤트 발생에 대한 조건과 정책을 문서화하는 데 사용됩니다. 따라서 스토밍 벽에서 정책은 도메인 이벤트와 커맨드 사이에 위치합니다. 정책은 “X일 때마다 Y” 또는 “만약 X이면 Y”와 같이 공식화됩니다.
예시: "예약이 생성될 때마다 제안을 생성합니다."

참고:

- Open Practice Library의 이벤트 스토밍[15]
- Open Practice Library의 큰 그림[16]
- 큰 그림 이벤트 스토밍[17]

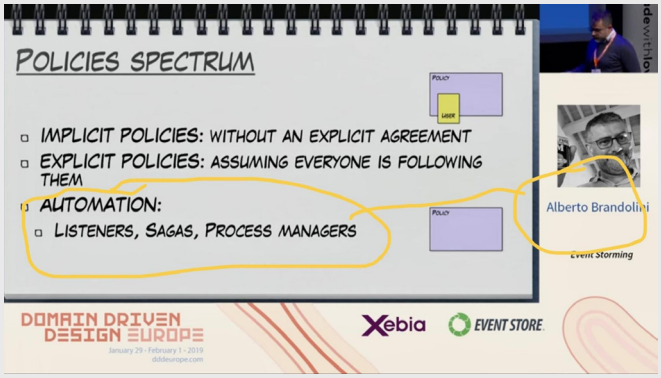
3.1.1.2. 이벤트를 통한 개념-기술 "확대"의 예

앞서 언급한 것처럼 이벤트 중심은 설계 워크샵과 그 이후의 구현 이터레이션 과정에서 비즈니스와 기술의 경계를 뛰어넘는 데 도움이 됩니다. 이 섹션에서는 사용자/비즈니스 컨텍스트의 이벤트, 즉 도메인 이벤트가 같은 "이벤트 언어"를 사용하면서 특정 기술 실현을 "확대"하는 데 어떤 도움이 되는지 알아봅니다.

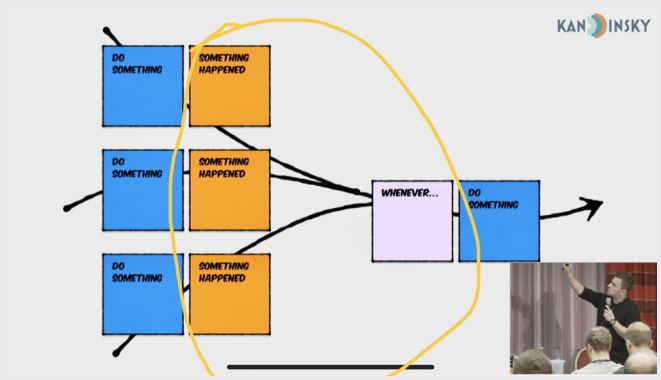
대표적인 이벤트 스토밍 사례를 처음 만든 알베르토 브란돌리니(Alberto Brandolini)는 자동화를 위한 ("이벤트 스토밍" 컨텍스트에서) 다음 유형의 "정책"을 언급합니다.

- 대상
- Saga
- 프로세스 관리자

[15] Open Practice Library의 이벤트 스토밍(Event Storming in Open Practice Library) https://openpracticelibrary.com/practice/event-storming/
[16] Open Practice Library의 큰 그림(The Big Picture in Open Practice Library) https://openpracticelibrary.com/practice/the-big-picture/
[17] 큰 그림 이벤트 스토밍(Big Picture Event Storming) https://medium.com/@chatuev/big-picture-event-storming-7a1fe18ffabb



참고: "이벤트 스토밍 - 알베르토 브란돌리니 - DDD 유럽 2019(Event Storming - Alberto Brandolini - DDD Europe 2019)"[18]



이 주제에 대한 또 다른 전문가인 마리우스 길(Mariusz Gil)도 "Saga/프로세스 관리자 패턴을 사용한 복잡한 프로세스 및 시간 모델링"이라는 강연에서 프로세스 관리자 패턴에 대해 자세히 설명합니다.[19] 특히 마리우스는 "프로세스 관리자" 자동화 패턴의 데모 구현을 보여줍니다. 간단히 말해, 도메인 이벤트의 세 가지 흐름을 하나의 흐름으로 병합하는 방법에 관한 것입니다. 이 가이드의 컨텍스트에서 기술적인 측면에 대해 말하자면, EDA(Ansible Rulebook)는 "프로세스 관리자" 패턴에 매우 잘 맞습니다. 다음은 그 작동 방식을 보여주는 다이어그램입니다.

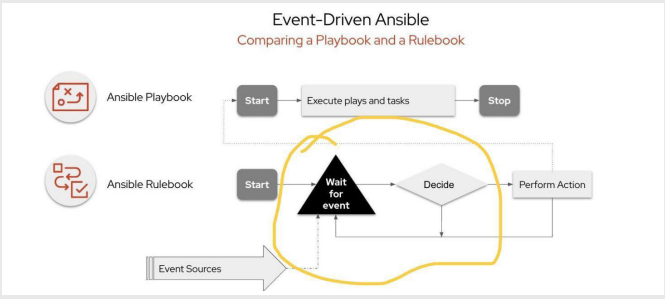


그림 출처: "Ansible Automation Platform 2.3의 새로운 기능"[20]

[18] 이벤트 스토밍 - 알베르토 브란돌리니 - DDD 유럽 2019(Event Storming - Alberto Brandolini - DDD Europe 2019) https://www.youtube.com/watch?v=mLXQIYEwK24

[19] Saga/프로세스 관리자 패턴을 사용한 복잡한 프로세스 및 시간 모델링 - 마리우스 길(Modeling complex processes and time with Saga/Process Manager patterns - Mariusz Gil) https://www.youtube.com/watch?v=WvjTCmeGIGA

[20] Ansible Automation Platform 2.3의 새로운 기능(What's new in Ansible Automation Platform 2.3) https://www.ansible.com/blog/whats-new-in-red-hat-ansible-automation-platform-2.3

3.1.2. 도메인 검색을 위한 전통적인 시각 "도메인 스토리텔링" 사례

프로젝트의 도메인을 그려보는 한 가지 방법은 "도메인 스토리텔링"을 사용하는 것입니다. 예를 들어, 프로세스에서 여러 사람이나 시스템이 적극적으로 관여하는 부분을 발견하게 될 수 있습니다. 이러한 부분이 도메인 분석에 중요하다면 프로세스에 대한 또 다른 관점을 얻을 수 있도록 이러한 부분을 도메인 스토리로 추가 모델링하는 것이 좋습니다.

그림을 보면 이해하기가 더 쉬우므로 도메인 스토리 "재고 소모품 주문"의 예를 살펴보겠습니다. 보시다시피 이 특정 예시의 그래픽 표기는 UML 구성 요소/배포 다이어그램과 유사합니다. 따라서 프로젝트 팀의 엔지니어에게 "친숙"할 수 있습니다.

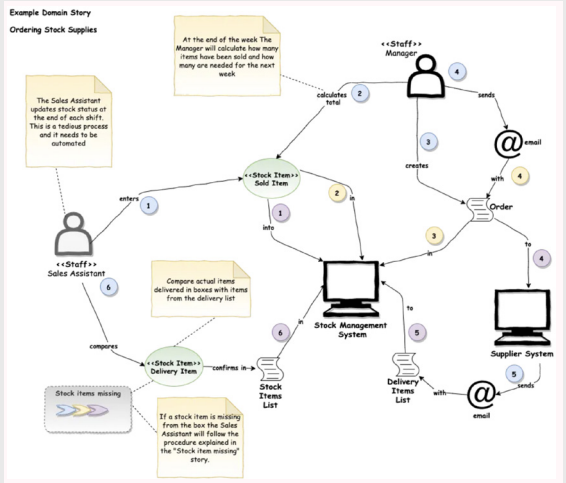


그림 출처: "도메인 스토리의 예"[21]

"도메인 스토리텔링"의 중요한 측면은 현재 상태와 원하는 상태를 연결한다는 점입니다 따라서 이 사례를 적용하면 프로젝트 목표를 더 자세히 설명하는 데 도움이 됩니다. 권장 단계는 다음과 같습니다.

1. 분야별 전문가(SME)가 비즈니스 프로세스(현재 상태)에 대한 설명을 시작하게 합니다.
2. 진행자 또는 지원 팀이 워크플로우 다이어그램에 각 단계를 기록합니다.
3. 기록된 각 단계에 번호를 매기고 아래에 자세한 설명을 해당 번호 옆에 추가합니다.
4. 원하는 상태가 될 때까지 이 과정을 반복합니다.

"도메인 스토리텔링"에 대한 자세한 내용은 여기에서 확인할 수 있습니다.[22]

3.1.3. "샘플 조직"을 위한 샘플 프로젝트: 동기 부여 매핑

아시다시피 비즈니스 과제는 계절에 따라 급격하고 반복적으로 변화할 수 있어 다이나믹한 경우가 많습니다. 이해관계자의 요구 사항을 항상 파악하는 한 가지 방법은 "동기 부여 매핑" 사례를 적용하여 그들의 동기를 깊이 이해하고 "매핑"하는 것입니다.[23] 이 섹션에서는 이러한 사례를 활용하여 이 가이드의 샘플 프로젝트에 구조를 부여합니다. 실제로 "동기 부여 매핑" 시각 캔버스를 채우면 프로젝트 팀과 모든 이해관계자에게 "동기 부여"가 전달됩니다. 그러면 모두가 자신의 실제 활동이 전략적으로 계획한 방향에 맞는지 수시로 확인할 수 있습니다.

[21] 도메인 스토리 예시(Example of a Domain Story) https://medium.com/domain-driven-stories/example-of-a-domain-story-d20ade05831e
[22] Open Practice Library의 도메인 스토리텔링(Domain Storytelling in Open Practice Library) https://openpracticelibrary.com/practice/domain-storytelling/
[23] Open Practice Library의 동기 부여 매핑(Motivation Mapping in Open Practice Library) https://openpracticelibrary.com/practice/motivation-mapping/

요약하자면, “동기 부여 매핑”은 간단한 캔버스를 사용하여 다음을 정리하는 것입니다.

- **컨텍스트 -** 우리 또는 고객을 둘러싼 환경은 무엇인가? 우리는 어떤 플랫폼에서 운영되고 있는가? 우리 환경에 영향을 미치는 요인은 무엇인가?
- **목표 -** 우리 또는 고객의 목표는 무엇인가? 우리는 무엇을 달성하고자 하는가?
- **목표의 이유 -** 이 목표나 목적을 설정한 이유를 파악합니다.
- **장애 요인 -** 목표 달성을 방해하는 요인은 무엇인가? 걸림돌은 무엇인가?
- **목표 성과 -** 목표를 달성했다는 것은 어떤 상태를 말하는가?

자동화 컨텍스트에서 다음과 같은 장점이 "동기 부여 매핑" 캔버스와 관련이 있는지 확인하는 것이 좋습니다.

- 자동화를 통해 실행의 일관성 강화
- 태스크 완료 시간 단축
- 팀 효율성 제고 및 더 중요한 태스크를 처리하는 역량 증대
- 취약점 관리 프로세스에 대한 신뢰 강화
- 환경 전반에서 배포 간소화
- 더욱 확장 가능한 프로세스
- Ansible & co.를 통해 "코드형 인프라(IaC)” 확대
- 향상된 기능 테스트
- 수동 작업/오버헤드 및 오류 감소
- 일부 자동화 콘텐츠는 다른 비즈니스 워크플로우에서 재사용 가능

이 가이드의 “샘플 프로젝트”는 가상의 "샘플 조직"을 대상으로 진행되며 이 조직은 최근 확장된 규정과 컴플라이언스 요건에 따라 달라질 수 있습니다. 이 회사가 금융 부문을 확대하고 새로운 지역에 진출하면서 이러한 상황이 발생하고 있습니다. 예를 들어 PCI-DSS 표준 세트[24]는 최신 패치를 설치하도록 요구하는 한편 중요 패치 설치에 허용되는 최대 시간을 설정하고 있습니다. 이 가이드의 "애플리케이션"은 이제 "샘플 조직" 작업에 매우 중요합니다. 이 애플리케이션은 "샘플 조직" 전반에서 액세스 권한을 요청하는 데 사용되는 기본 소프트웨어이므로 안정성과 가용성이 가장 중요합니다.

프로젝트의 초기 시점에 "샘플 프로젝트" 팀에 대해서도 언급하겠습니다. 다음과 같은 준비 활동이 적절합니다.

- 온보딩 활동, 필요한 원격 협업 툴 테스트, 주간 미팅 설정, 화이트보드 세션 준비(현장 활동이 계획된 경우)
- 소개 및 모든 구성원의 입장 확인
- 높은 성과를 내는 팀의 요건과 해당 팀의 장점 및 거쳐야 할 단계 논의(Tuckman의 "형성-스토밍-규범화-성과" 모델)
- 개별 및 팀 업무 방식에 대해 논의하고 합의("사회적 계약" 생성 및 합의)

다음은 이 가이드의 "샘플 프로젝트"에 대한 "동기 부여 매핑" 캔버스의 모습입니다. 이 캔버스는 "샘플 프로젝트" 팀과 "샘플 조직"에서 근무해 상황을 잘 아는 이해관계자들이 한 시간 동안 제작했습니다.

[24] 위키피디아의 PCI DSS 항목(PCI DSS on Wikipedia) https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

컨텍스트

카테고리 “애플리케이션”(비즈니스 애플리케이션에 중요)

> 조직 전반의 액세스 권한을 요청하는 데 사용되는 중요한 비즈니스 애플리케이션을 실행하는 데이터센터의 Linux® 장비. 최근 사용자 수가 증가하면서 불안정한 동작이 보고되었습니다. 이 애플리케이션은 컴플라이언스를 위해 Linux 장비에 있어야 합니다. 애플리케이션 배포는 현재 수동으로 관리되고 있습니다.

카테고리 "제품군 패치 적용"(Linux 제품군에 대한 패치 적용 및 보안 문제 해결)

> 회사는 새로운 시장으로 확장하고 있으며 이제 더 많은 용량이 필요합니다. 나머지 수동 단계는 매일 장애 요인이 되고 있습니다. 조직 내 전체 Linux 제품군에 대한 작업량과 완료 시한 요구 사항은 향후 몇 달 내에 급격히 증가할 것으로 예상됩니다.

목표

애플리케이션

> 비즈니스 사용자는 애플리케이션에 1년 365일 안정적으로 액세스할 수 있어야 합니다.

제품군에 패치 적용

> 패치 적용은 완전히 자동화되어야 합니다. 이러한 자동화를 통해 내부 고객에게 제공되는 업데이트의 속도와 품질이 크게 개선되며 전문가는 만족도가 높은 업무에 집중할 수 있습니다.

목표의 이유

애플리케이션

> 다양한 시간대의 새로운 시장으로 확장하는 비즈니스는 거의 1년 365일 운영됩니다. 특히 야간과 주말(원래 시간대)에 애플리케이션을 필요로 하는 사용자가 더 많아졌습니다. 애플리케이션 업그레이드/유지관리를 위한 "유지 관리 창구"를 찾기는 점점 더 어려워지고 있습니다. 또한 새로 합류한 경영진은 애플리케이션의 느린 응답 속도에 불만을 느끼고 있습니다.

제품군에 패치 적용

> Linux 장비에 대한 규제 컴플라이언스 관리가 점점 더 어려워지는 만큼, 완전한 자동화가 필요합니다. 이 프로세스로 인해 직원들이 더 중요한 업무에 할애할 수 있는 시간이 줄어듭니다. 수동 업무가 많아 인적 오류 가능성이 높습니다. 인프라 팀은 "사이트 신뢰성 엔지니어" 모델로 업무 방식을 전환하고자 합니다.

장애 요인

"애플리케이션"과 "제품군 패치 적용"이라는 두 가지 카테고리:

> 애플리케이션의 고가용성(HA)을 구현해야 하는 엔지니어의 온보딩, 패치 프로세스 개선은 보안상의 이유로 2~4주가 소요됩니다. 또한 프로젝트 팀이 사용할 수 있는 지역 데이터센터의 개발(프로덕션 제외) 환경을 위한 컴퓨팅 용량이 현재 부족합니다. 추가 용량은 3~6주 내에 프로비저닝할 수 있습니다.

목표 성과

애플리케이션

- 프로덕션 환경의 새로운 애플리케이션 버전 업데이트와 해당 Linux 시스템의 보안 패치 적용 및 유지관리는 비즈니스 사용자의 서비스 중단 없이 이루어져야 합니다.
- 애플리케이션은 사용량이 많은 시간대에도 신속해야 합니다. 엣지 환경에서 만족스러운 수준의 응답 시간은 최대 3초입니다. 그러나 사용자 요청의 95% 이상은 1초 이내에 완료되어야 합니다.

제품군에 패치 적용

- 미션 크리티컬 Linux 워크로드의 경우, 자동화된 패치 적용과 보안 문제 해결은 서비스 다운타임 없이 이루어져야 합니다. 애플리케이션의 경우 6주 내에, 기타 여러 미션 크리티컬 Linux 워크로드의 경우 2~3개월 내에 완료해야 합니다.
- 중요한 Linux 워크로드가 아닌 경우, 최대 3000대의 서버에 대해 24시간 이내에 자동으로 패치를 적용하고 보안 문제를 해결해야 합니다(계획된 다운타임 허용). 이 수준의 "패치 처리량"은 3개월 이내에 달성해야 합니다.

3.1.4. 샘플: 초기 검색

이벤트 스토밍 워크샵에는 적임자가 참석해야 합니다. 여기에는 어떤 질문을 해야 할지 아는 사람과 사정을 잘 아는 이해관계자(도메인 전문가, 제품 소유자)가 포함됩니다. 훌륭한 사회자와 진행자(스크럼 마스터, 애자일 코치)의 도움을 받으면 원활히 진행됩니다.

이벤트 스토밍 워크샵을 준비하는 동안 "샘플 프로젝트" 팀은 다음과 같은 지식 영역에 집중하기로 결정했습니다.

- 사용 중인 애플리케이션 - 액세스 권한 요청
- 애플리케이션 새 버전 배포
- 애플리케이션의 Linux 장비에 패치 적용

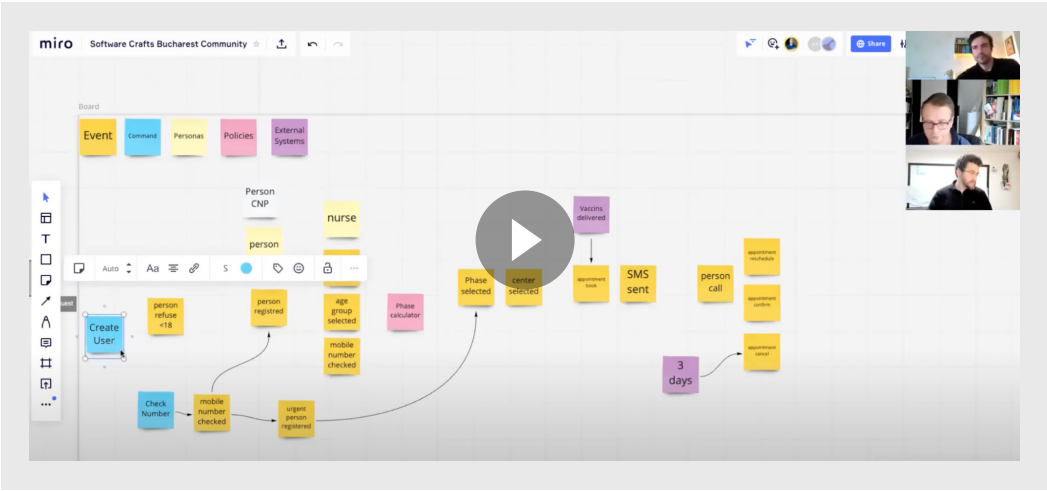
프로젝트 팀은 위에서 언급한 각 지식 영역에 대해 다음 단계를 거쳤습니다.

- 1단계 - 도메인 이벤트 수집(큰 그림)
- 2단계 - 도메인 이벤트 구체화(큰 그림)
- 3단계 - 원인 추적(프로세스 모델링)

이 가이드의 컨텍스트에서 가장 중요한 지식 영역은 "애플리케이션 새 버전 배포"입니다. 다음 섹션에는 위에서 언급한 세 단계 전체에 대한 "샘플 프로젝트" 팀의 이벤트 스토밍 결과가 나와 있습니다. 다른 두 영역에 대한 결과는 다음 부록에서 각각 확인할 수 있습니다.

- 부록 A. "사용 중인 애플리케이션 - 액세스 권한 요청"에 대한 이벤트 스토밍
- 부록 B. "애플리케이션의 Linux 장비 패치 적용"에 대한 이벤트 스토밍

처음부터 이벤트 스토밍 세션의 전체 예시를 확인하고 싶다면 다음 동영상이 도움이 될 수 있습니다 “이벤트 스토밍 워크숍, 부쿠레슈티 소프트웨어 장인정신 커뮤니티”[25]

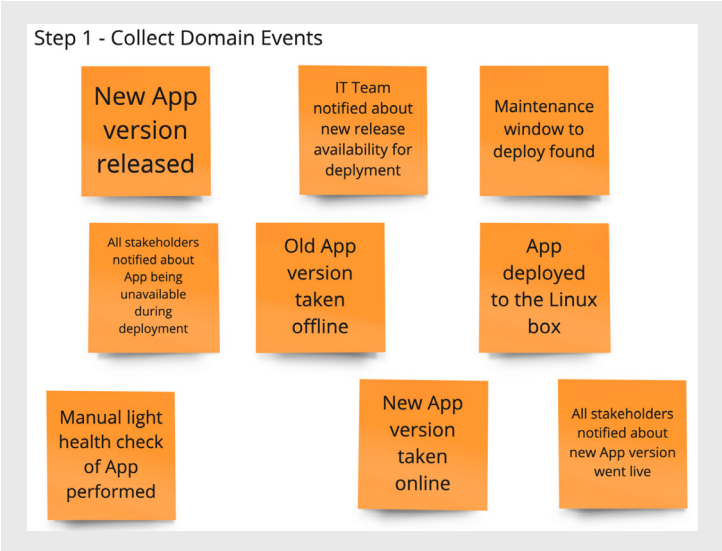


[25] 이벤트 스토밍 워크숍, 부쿠레슈티 소프트웨어 장인정신 커뮤니티(Event Storming Workshop @Bucharest Software Craftsmanship Community) https://www.youtube.com/watch?v=xVSaDdj3PVE

3.1.4.1. 샘플: "애플리케이션 새 버전 배포"를 위한 이벤트 스토밍

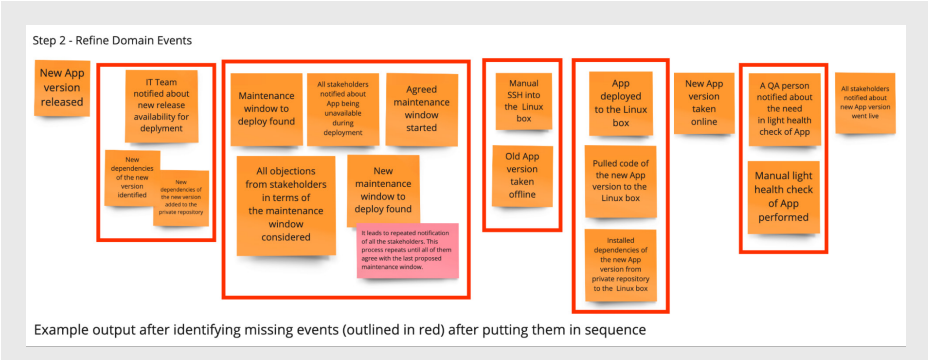
"1단계 - 도메인 이벤트 수집(큰 그림)" 단계에서는 각 참가자가 주황색 포스트잇만 사용합니다. 각각의 주황색 포스트잇은 전문 이벤트를 의미합니다. 전문 이벤트는 비즈니스 과정에서 발생한 기술적으로 관련된 사실입니다. 따라서 포스트잇의 동사는 과거에 일어난 일이어야 합니다.

첫 단계는 기존 도메인 이벤트에 대한 순수한 브레인스토밍 과정입니다. 사람들에게 사건 발생 순서대로 이벤트를 달도록 요청합니다.



"2단계 - 도메인 이벤트 구체화(큰 그림)"에서는 참가자와 함께 도메인 이벤트 포스트잇을 살펴봅니다. 참가자에게 각 이벤트의 의미를 설명해 달라고 요청합니다. 구문의 정확성을 확인합니다.

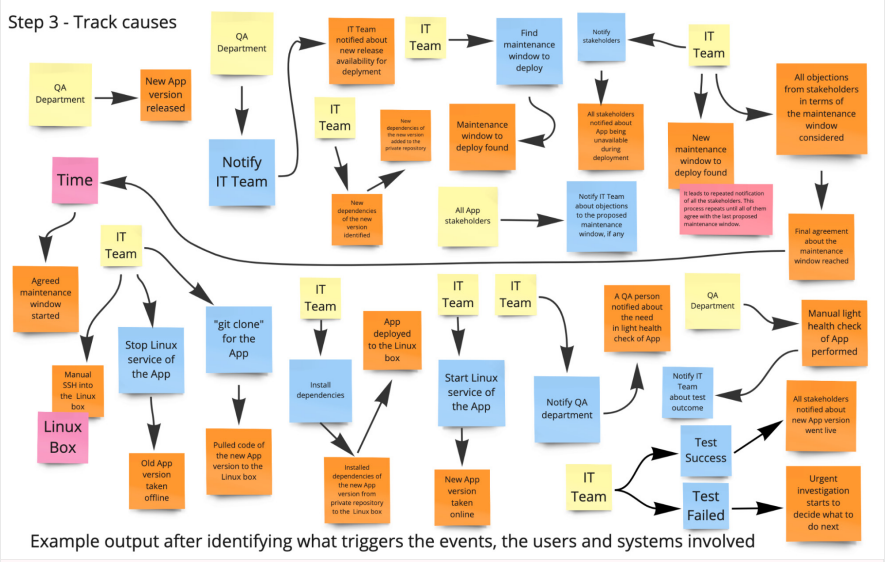
또한 이벤트가 올바른 시간 순으로 나열되었는지 다시 한번 논의합니다. 발생하는 동의어(같은 항목에 대한 다른 용어)를 통합하고, 같은 용어가 다른 것을 설명하는 데 사용되었다면 그 차이점을 명확히 합니다.



"3단계 - 원인 추적(프로세스 모델링)"에서는 원인 분석에 들어갑니다. 도메인 이벤트는 어디에서 발생할까요? 다음 네 가지 주요 원인이 있습니다.

- 사용자 작업(커맨드)
- 외부 시스템
- 시간(예: 약속 경과), 비즈니스 프로세스
- 기타 도메인 이벤트(자동 반응, 정책/비즈니스 룰을 통해)

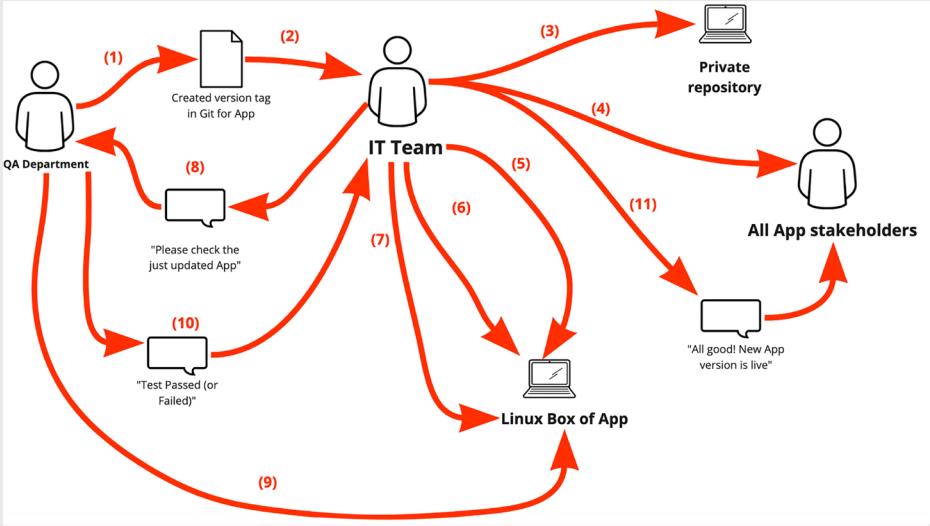
참가자들에게 도메인 이벤트의 트리거에 대해 물어봅니다.



3.1.4.2. 샘플: "애플리케이션 새 버전 배포"에 대한 도메인 스토리(현재 상태)

"샘플 프로젝트" 팀은 "애플리케이션 새 버전 배포" 프로세스에 대한 또 다른 관점을 얻기 위해 프로세스의 협업 부분을 도메인 스토리로 추가로 모델링했습니다. 팀은 아래의 다이어그램과 설명을 만들기 위해 다음 단계를 거쳤습니다.

- 분야별 전문가(SME)가 비즈니스 프로세스(현재 상태)에 대한 설명을 시작하게 합니다.
- 진행자 또는 지원 팀이 워크플로우 다이어그램에 각 단계를 기록합니다.
- 기록된 각 단계에 번호를 매기고 아래에 자세한 설명을 해당 번호 옆에 추가합니다.



1. QA 부서에서 Git 버전 제어 시스템에서 애플리케이션의 새 릴리스에 태그를 지정합니다.
2. QA 부서에서 IT 팀에 배포할 수 있는 새 릴리스 가용성에 대해 알립니다.
3. IT 팀이 새로운 종속성을 식별하고 해당 아티팩트를 비공개 리포지토리에 추가합니다.
4. IT 팀은 새 버전의 애플리케이션을 배포하기 위한 유지관리 기간에 대해 모든 애플리케이션 이해관계자와 합의합니다.
5. 합의된 유지관리 기간이 시작되면 IT 팀은 수동으로 Linux 장비에서 애플리케이션의 Linux 서비스(이전 버전)를 중지합니다.
6. IT 팀은 새 버전의 애플리케이션을 Linux 장비에 배포합니다. (a) 새 애플리케이션 버전의 코드를 Linux 장비로 가져오고, (b) 새 애플리케이션 버전의 종속성을 비공개 리포지토리에서 Linux 장비로 설치합니다
7. IT 팀이 애플리케이션의 Linux 서비스를 시작하여 온라인으로 전환합니다.
8. IT 팀은 방금 업데이트한 애플리케이션의 간단한 상태 점검의 필요성을 QA 부서에 알립니다.
9. QA 부서에서 방금 배포한 애플리케이션에 대해 수동으로 간단한 상태 점검을 수행합니다.
10. QA 부서에서 테스트 결과를 IT 팀에 알립니다.
11. 새 애플리케이션 버전을 간단히 테스트한 결과가 "통과"로 보고된 경우, IT 팀은 모든 이해관계자에게 새 애플리케이션 버전이 가동되었음을 알립니다.

3.2. 어떤 문제가 있나요? 프로세스의 장애 요인은 무엇인가요?

이제 당면한 쟁점, 문제점, 비효율성 및 위험을 보다 구체적으로 이해하고자 합니다. 현재 전략적으로 가장 큰 영향을 미칠 수 있는 개선 사항은 무엇인가요? “프로젝트 팀”을 프로세스 개선 작업에 투입하려 합니다. 프로세스 장애 요인을 명확히 이해하는 "프로젝트 팀"은 현재 전략적으로 가장 관련성이 높은 프로세스 개선에 집중할 수 있습니다. 다음 섹션에서는 MBPM 사례가 어떻게 도움이 될 수 있는지 살펴보고 지금까지 수집한 모든 관련 정보를 요약하겠습니다.

3.2.1. "메트릭 기반 프로세스 매핑" 사례란?

MBPM에 대해 소개하겠습니다. MBPM은 린(lean) 프로세스 개선 기법입니다. MBPM 실습은 가능한 프로세스 개선 사항과 그 영향을 파악하는 데 도움이 됩니다. MBPM 사례는 고객/이해관계자에게 성과/서비스를 제공하는 프로세스에 대한 상세한 맵을 만듭니다. 이 맵에는 프로세스 단계, 책임 있는 행위자, 리드 타임 메트릭 및 품질 메트릭이 표시됩니다. 이를 통해 아래에 제시된 프로세스의 잠재적 개선 사항을 파악할 수 있습니다.

따라서 프로젝트 참가자는 **프로세스**의 각 **단계**에 대해 "MBPM을 수행"하는 동안 다음을 파악하거나 추정할 수 있습니다.

- 프로세스 단계별 작업을 설명하는 **동사-명사 구문**. **예: "테스트 코드"**(필요한 경우 명확성을 위해 세부 사항 추가 가능)
- 해당 단계를 실행하는 **팀** 또는 **행위자**
- **프로세스 시간(PT)**, 즉 팀이나 행위자가 실제로 특정 단계에서 그에 해당하는 생산적인 업무를 실제로 수행하는 시간
- **리드 타임(LT)**, 또는 경과된 시간:
  - 특정 단계에서 팀/행위자가 업무를 시작할 준비가 된 시점**부터**(즉, 합리적인 종속성이 없는 경우, 예를 들어 백로그 항목에 대해 "정의가 준비 완료"되는 시점)
  - 이 특정 단계의 산출물/성과가 (그에 상응하는 입력으로) 다음 "업무 단계"로 전달되는 시점**까지**
- **"완료 및 정확도 비율"(PCA** 또는 **"%C&A")**, 즉 특정 단계에서 완벽하고 정확하게 완료된 산출물의 비율. 다운스트림 "업무 센터"(팀/행위자)가 다운스트림 업무를 시작하기 위해 기타 수정 사항, 추가 사항 또는 설명이 필요하지 않은 경우 산출물은 정확히 완료된 상태입니다.

예를 들어, 특정 프로세스의 작업 활동을 이러한 방식으로 "매핑"하면 다음을 파악하는 데 도움이 됩니다.

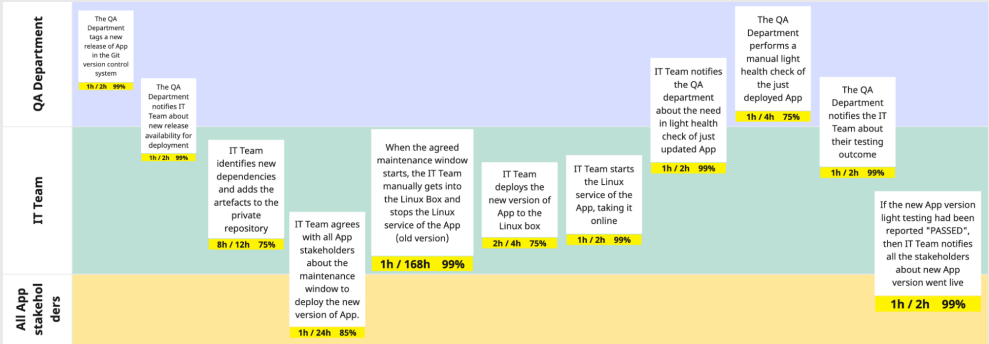
- 프로세스의 산출물인, 결함이 없는 결과물의 비율. 이러한 결과물에 대한 LT.
- 일부 중간 프로세스 단계에서 수정 또는 재작업이 필요하여 프로세스 속도 저하의 원인이 되는 결과물의 비율.
- PT가 짧으니 LT가 큰 단계(장애 요인이 있음을 나타냄). 이러한 단계는 상당한 정도의 개선이 필요할 수 있고, 일반적으로 PT/LT 비율이 높고 PT가 낮은 일부 대안을 통해 "리팩토링"할 수도 있습니다.

이 가이드에서 다루지 않는 MBPM의 다른 예가 궁금하다면 다음 동영상이 도움이 될 수 있습니다: "메트릭 기반 프로세스 매핑으로 프로세스 시각화, 측정 및 최적화"[26]

3.2.2. 샘플: MBPM - 현재 프로세스 흐름, 소유자, 메트릭

다음 다이어그램은 "애플리케이션 새 버전 배포"에 대한 MBPM의 가능한 예를 보여줍니다. 각 프로세스 단계의 아래쪽에는 다음과 같은 그림이 있습니다.

프로세스 시간(PT)/ 리드 타임(LT)　"완료 및 정확도 비율"(PCA %)



[26] 메트릭 기반 프로세스 매핑으로 프로세스 시각화, 측정 및 최적화(Visualizing, measuring and optimizing your processes with Metric Based Process Mapping) https://www.youtube.com/watch?v=g1XSbEwR3bU

보다 간단한 분석을 위해 수집/추정된 PT-LT-PCA 데이터를 표로 요약해 보겠습니다. 이 표는 MBPM 사례가 정량적인 방식으로 어떻게 프로세스 개선을 제안할 수 있는지 보여줍니다.

단계 이름	PT(시간)	LT(시간)	PT/LT%	PCA%
1. 새 릴리스 태그 지정	1	2	50%	99%
2. IT 팀에 새 릴리스 가용성 알리기	1	2	50%	99%
3. 새로운 종속성 식별 및 비공개 리포지토리 업데이트	8	12	67%	75%
4. 배포에 사용할 유지관리 기간에 대해 모두와 합의	1	24	4%	85%
5. 합의된 유지관리 기간이 시작될 때까지 기다렸다가 애플리케이션의 Linux 서비스 중지	1	168	0.5%	99%
6. 새 버전의 애플리케이션 배포	2	4	50%	75%
7. 애플리케이션의 Linux 서비스 시작	1	2	50%	99%
8. 간단한 상태 점검의 필요성을 QA 부서에 알리기	1	2	50%	99%
9. 방금 배포한 애플리케이션에 대해 수동으로 간단한 상태 점검 수행	1	4	25%	75%
10. IT 팀에 테스트 결과 알리기	1	2	50%	99%
11. 모든 이해관계자에게 새 애플리케이션 버전이 가동되었음을 알리기	1	2	50%	99%
요약	**요약 PT: 19시간**	**요약 LT: 224시간**	**요약 PT/LT: 8.5%** "요약 LT"로 나눈 "요약 PT"	**요약 PCA 33%** 모든 PCA% 값을 곱함

앞의 샘플 프로세스 맵과 표에는 다음과 같은 중요한 결과가 표시됩니다.

- 결과물의 33%가 프로세스를 "통과"하여 결함 없이 완전하고 정확하게 완성됩니다. 이러한 결과물의 리드 타임은 224시간입니다.
- 대부분의 결과물(67%)은 일부 중간 프로세스 단계에서 수정 또는 재작업이 필요하며, 이로 인해 리드 타임이 "예상한" 224시간보다 길어집니다.
- 4번째와 5번째 단계는 리드 타임이 길고 PT/LT 비율이 낮아 장애 요인이 있음을 나타냅니다. 이러한 단계를 개선하면 전체 프로세스 리드 타임을 최대로 개선할 수 있습니다.
- 9번째 단계는 리드 타임이 적당한 편이지만, PT/LT 비율이 여전히 너무 낮습니다. 또한 PCA가 75%라는 건 부정확성/불완전성으로 인해 LT가 증가할 위험이 상대적으로 높다는 의미입니다.

3.2.3. 샘플: 현재 문제와 위험

지금까지 "샘플 프로젝트" 팀이 수행한 작업은 다음과 같습니다.

- "동기 부여 매핑": 전반적인 목표, 컨텍스트, 목표의 이유, 장애 요인, 목표 성과, 그리고 "동기 부여 매핑"을 통해 정의된 컨텍스트에서 다음과 같은 특정 지식 영역을 추가로 분석했습니다.
- 사용 중인 애플리케이션 - 액세스 권한 요청
  - 이벤트 스토밍
- 애플리케이션 새 버전 배포
  - 이벤트 스토밍
  - 도메인 스토리(현재 상태)
  - MBPM
- 애플리케이션의 Linux 장비에 패치 적용
  - 이벤트 스토밍

그 후, "샘플 프로젝트" 팀은 컨텍스트의 쟁점, 문제점, 비효율성 및 위험을 요약하는 세션을 두어 차례 가졌습니다. 다음은 이러한 분석 세션에서 수집한 결과입니다.

(a) 문제 분석: "비즈니스 애플리케이션에 중요한 사항"

- 애플리케이션이 내부적으로 작동하는 방식에서 논리적 문제는 발견되지 않음. 단일 Linux 장비의 애플리케이션은 사용자 부하 증가로 인해 때때로 응답하지 않음. 단일 Linux 장비의 처리 능력이 부족하기 때문(CPU가 과도하게 사용되었을 가능성이 높음)
- 새 버전 배포가 쉽지 않음
- 애플리케이션의 상태를 적절히 파악하지 못했으며, 사용자 관련 오류는 사후에 발견됨
- 유지 관리 기간: 협상에 많은 시간이 소요되고, 시작까지 많은 시간이 소요됨
- 수동 테스트는 부서 간 조율이 필요하며 리드 타임이 프로세스 시간보다 상당히 오래 소요됨
- 위험: 수동 단계와 수동 테스트는 사람의 실수가 발생하기 쉬움

(b) 문제 분석: "Linux 제품군에 대한 패치 적용 및 보안 문제 해결"

패치 적용은 부분적으로만 자동화되어 있음. 단점:

- 수동 조작이 너무 자주 필요하며, 이러한 수동 개입의 정도는 Linux 장비 수에 비례하는 경우가 많음
- 완료 속도가 너무 느림
- 다운타임을 피할 수 없을 때도 있음
- 유지 관리 기간: 협상에 많은 시간이 소요되고, 시작까지 많은 시간이 소요됨
- 위험: 수동 단계와 수동 테스트는 사람의 실수가 발생하기 쉬움

요약하면, 위에서 확인된 문제와 위험은 프로세스 자동화 수준이 낮거나 프로세스 수행을 위해 할당된 컴퓨팅 용량이 불충분한 것과 관련이 있습니다.

그렇다면 이 모든 상황을 개선하기 위한 전략은 무엇인가요? 다음 섹션에서는 몇 가지 유용한 관련 사례를 소개하고 이에 따라 "샘플 프로젝트" 팀이 어떤 결정을 내리는지 살펴봅니다.

3.3. 실제 요구 사항에 따라 제공할 준비가 되어 있나요?

섹션 3.1.3. "샘플: 동기 부여 매핑..."에서는 샘플 프로젝트의 전반적인 목표 성과를 소개했습니다. 현재 상호 작용에 대한 검색 사례를 완료한 후에는 이러한 요구 사항이 점차 변경되고 방금 우선순위가 지정된 항목이 중복될 수도 있으므로 우선순위를 실제 비즈니스/고객 요구 사항과 다시 연계해야 합니다. 기본적으로 이터레이션마다 헛수고가 되지 않도록 확인하고 있습니다. "산출물"은 "성과"와 다릅니다. 즉, "실제 투입된 노력의 모든 결과물"과 "이해관계자에게 가치 있는 결과물/성취"는 동일하지 않습니다.

다시 말해, "검색/개발"의 초기가 아닌 이터레이션에서 "목표 성과" 세션이 열릴 때는 방금 완료된 이터레이션의 결과물/성취가 주된 초점입니다. 여기서는 가장 효율적인 방향으로 가고 있는지 비교 검토하고 확인하고자 합니다.

아래에 제시된 "목표 성과" 사례는 이러한 비교 검토를 체계적으로 수행하는 데 도움이 됩니다.

3.3.1. "목표 성과" 검색 사례란?

"목표 성과" 사례는 검색 이터레이션의 성과 요소의 주요 사례입니다. 다른 검색 사례에서 얻은 결과물과 학습 내용을 팀과 이해관계자들이 정기적으로 참조할 수 있게 공개적으로 표시할 수 있는 한정된 진술로 요약합니다. 참고: 목표 성과.[27]

이 시점에서 프로젝트 팀의 관점에서 의견을 말씀드리겠습니다. 다음 중 어떤 시나리오에서든 팀은 목표 성과를 설정함으로써 이점을 얻을 수 있습니다.

- 팀 결과물의 범위가 예기치 않게 변경되는 경우
- 고객이 제품을 사용하거나 제품 피드백을 제공하기 전에 팀이 서비스를 성공적으로 제공했음을 알리는 경우
- 일상적인 팀 상호 작용이 원하는 결과를 달성하는 기능의 버전을 완성하기보다 기능을 완성하는 데 중점을 두는 경우

3.3.2. 샘플: 첫 번째 제공 이터레이션을 위한 전략

"샘플 프로젝트"의 첫 번째 "제공 이터레이션"을 준비하고 있습니다. 이 단계에서는 MVP 구축에 집중하는 것이 일반적입니다. MVP란?

3.3.2.1. MVP란?

일반적으로 MVP는 다음과 같은 목표를 달성하는 것을 목표로 합니다.

- 사용자/이해관계자가 만지고 느끼고 상호작용할 수 있는 것을 제공
- 제품 아이디어를 실제 데모로 시연하고, 이해관계자가 프로젝트에 더 관심을 갖도록 하며, 더 많은 잠재적 사용자를 유치
- 가능한 한 빨리 제품 아이디어에 더 많은 노력과 시간을 투자할 가치가 있는지 검증. 그렇지 않은 경우 제품에 더 많은 리소스를 투입하는 것이 합리적이지 않음을 분명히 알림
- 실제 사용자/이해관계자로부터 귀중한 피드백 수집

간단히 말해, MVP는 "전체 여정을 완료하기 위한 최단 경로"이며, 여기서 "전체 여정"은 프로젝트의 전반적인 미래 성과입니다. 즉, MVP의 포커스는 "부가 기능"이 아닙니다.

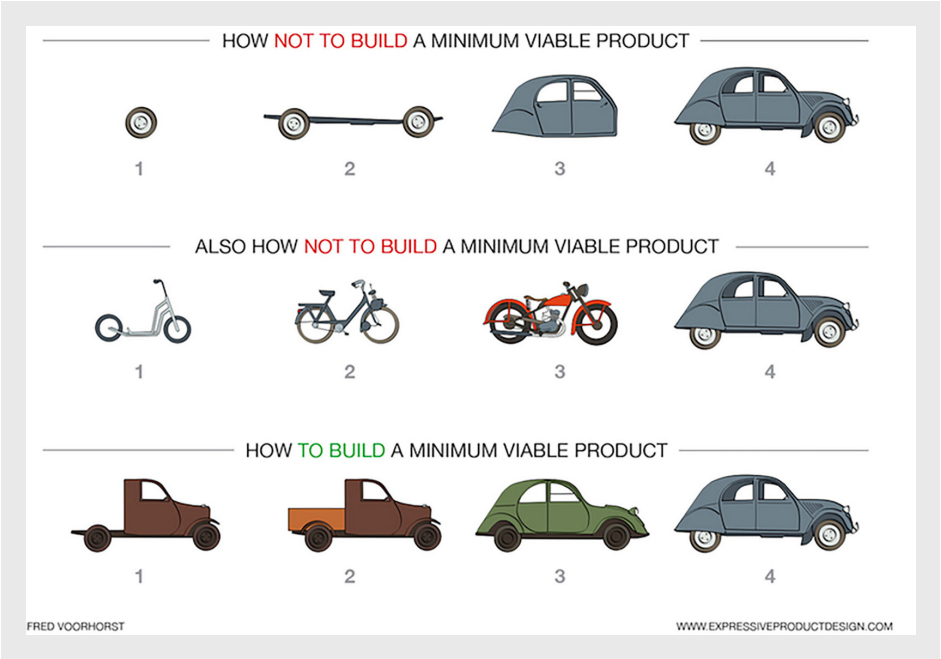
[27] Open Practice Library의 목표 성과(Target Outcomes in Open Practice Library) https://openpracticelibrary.com/practice/target-outcomes/

MVP 개발에는 다음 두 가지가 모두 필요합니다

A. 전체 프로젝트 계획 실행을 통해 전체적으로 이상적인 최종 프로젝트 성과가 되도록 효율적으로 작업

B. MVP 릴리스와 동시에 앞으로도 전체 프로젝트 성과의 최종 사용자에게 매우 가치 있고 사용 가능한 결과 제공

아래 그림에서 세 번째 줄은 위의 (A)와 (B)가 모두 충족되는 경우의 예시입니다. 그림의 다른 두 가지 예를 보면 두 번째 줄은 (A)에 맞지 않고, 첫 번째 줄은 위의 (B)에 맞지 않습니다.



다음 섹션에서는 샘플 프로젝트의 MVP에 대해 간략하게 설명합니다.

3.3.2.2. MVP의 개념과 옵션

일반적으로 샘플 프로젝트 팀은 샘플 프로젝트의 목표에 부합하는 MVP의 형태로 가치를 제공하는 것이 합리적이라는 데 동의했습니다. MVP가 합리적인 비용으로 이해관계자들에게 가치를 제공한다고 입증되면 샘플 프로젝트 팀은 나중에 팀 투표/우선순위에 따라 기능을 추가할 수 있습니다.

즉, 팀/이해관계자가 지금 어떤 기능을 제안하든 크게 다음 두 가지 카테고리로 분류할 수 있습니다.

- MVP 범위 내
- MVP 이후

한 가지 중요한 점은 "MVP 이후" 릴리스를 위해 사전 계획된 모든 기능/항목이 실제로 구현될 수도 있고 안될 수도 있다는 점입니다. MVP의 사용자/이해관계자의 피드백은 릴리스 직후 수집되며, 요청된 개선 사항은 기존의 "후보 항목"과 함께 다음 릴리스에 포함할지 여부를 고려할 가능성이 높습니다.

섹션 3.1.3. "샘플: 동기 부여 매핑..."에서는 샘플 프로젝트의 전반적인 목표와 기타 컨텍스트에 맞는 세부 사항을 소개했습니다. 다음은 해당 도메인별 목표입니다

(a) 비즈니스 애플리케이션에 중요한 사항

비즈니스 사용자는 애플리케이션에 1년 365일 안정적으로 액세스할 수 있어야 합니다.

(b) Linux 제품군에 대한 패치 적용 및 보안 문제 해결

패치 적용은 완전히 자동화되어야 합니다.

이러한 목표를 염두에 두고 프로젝트 팀은 "MVP 개요 정리" 세션을 갖기로 결정했습니다. 팀원들은 화이트보드/Miro 보드에 몇 가지 아이디어를 제안했습니다. 10분 정도의 시간이 걸렸습니다. 세션 진행자가 아이디어를 추가로 그룹화하고 조금 더 다듬었습니다. 이 그룹화 활동은 모든 참가자가 명확히 이해할 수 있도록 다소 편안한 분위기에서 요점을 중심으로 토론하는 방식으로 진행되었습니다. 25분이 더 걸렸습니다.

다음은 첫 번째 제공 이터레이션에서 MVP를 샘플 프로젝트에 제공할 수 있도록 개선된 아이디어의 결과물입니다

A. 다운타임이 있는 자동화된 패치 적용(중요하지 않은 Linux 장비용)

B. 쿠버네티스/Red Hat OpenShift로 애플리케이션 리플랫포밍

C. 애플리케이션 배포, 패치 적용, 관측성 및 성능 개선. (애플리케이션 패치 적용만 자동화할 예정이며 추후 HA 요구 사항이 있는 더 많은 Linux 장비로 이 아이디어를 확장하려고 합니다.)

위에서 언급한 항목에 대해 논의하는 동안 팀은 다음 사항을 강조했습니다.

- **아이디어(A)**는 "패치는 완전히 자동화되어야 한다"는 내용을 자세히 다루고 있지만 샘플 프로젝트의 가용성에 대한 기대치와 같이 애플리케이션에 대한 합리적인 주의를 기울이지는 않고 있습니다.
- **아이디어(B)**는 컨테이너화된 애플리케이션 오케스트레이션, 관측성 및 가용성의 최근 개발 방식을 활용하여 애플리케이션의 운영 준비도를 획기적으로 개선할 수 있는 잠재력이 있습니다. 그러나 여기에는 "Linux 제품군의 자동화된 패치 적용"이라는 측면이 빠져 있으며, 자체 Linux 제품군은 샘플 프로젝트의 성과를 활용하는 조직의 IT 자산 중 자연스럽고 중요한 부분입니다.
- **아이디어(C)**는 관련 애플리케이션 개선과 Linux 장비 패치 적용을 모두 다룹니다. 이 옵션의 단점은 현재 정확히 무엇을 해야 할지 명확하지 않다는 점입니다. 하지만 이 아이디어의 최초 작성자는 기술 설계에 대한 몇 가지 인사이트를 제공함으로써 이 아이디어가 괜찮다고 팀원들을 설득할 수 있었습니다.

진행자는 투표 세션을 진행하여 팀이 제출하기로 결정한 아이디어, 즉 MVP 아이디어에 대해 합의하기로 했습니다. 투표에 7분이 걸렸습니다. 투표 결과는 다음과 같습니다.

- **1위 – 아이디어(C):**애플리케이션 배포, 패치 적용, 관측성 및 성능 개선
- **2위 – 아이디어(B):** 쿠버네티스/Red Hat OpenShift로 애플리케이션 리플랫포밍
- **3위 – 아이디어(A):** 다운타임이 있는 자동화된 패치 적용(중요하지 않은 Linux 장비용)

앞서 언급했듯이, 1위를 차지한 아이디어(C)의 단점은 현재 MVP 결과를 달성하기 위해 정확히 무엇을 해야 할지 명확하지 않다는 점입니다. 다음 섹션에서는 필요한 명확성을 확보하는 데 도움이 되는 몇 가지 사례를 소개합니다.

3.4. 첫 번째 제공 이터레이션을 시작할 준비가 되었나요?

앞서 논의한 것처럼 일반적으로 샘플 프로젝트 팀은 샘플 프로젝트의 목표에 부합하는 MVP의 형태로 가치를 제공하는 것이 합리적이라는 데 동의했습니다. MVP의 특징은 다음과 같습니다.

애플리케이션 배포, 패치 적용, 관측성 및 성능 개선
(애플리케이션 패치만 자동화합니다. 추후 HA 요구 사항이 있는 더 많은 Linux 장비로 이 아이디어를 확장하려고 합니다.)

이러한 개선 사항은 다음과 같은 전체 프로젝트 목표와 함께 진행되어야 합니다.

- 비즈니스 사용자는 애플리케이션에 1년 365일 안정적으로 액세스할 수 있어야 함
- 패치 적용은 완전히 자동화되어야 함

이제 샘플 프로젝트 팀과 사정을 잘 아는 이해관계자가 간단한 프로젝트에서 제공할 특정 기능을 제안할 차례입니다. 그런 다음 그 기능을 크게 두 가지 카테고리로 분류해야 합니다.

- MVP 범위 내
- MVP 이후

MVP 범위 내 기능은 MVP의 작업 범위(SoW)를 효과적으로 구성합니다.

이제 팀은 "샘플 조직" 도메인을 잘 이해하고 있습니다. 큰 그림의 이벤트 스토밍 결과와 "도메인 스토리" 다이어그램이 있습니다. 이들은 MBPM 사례를 적용하여 도메인 프로세스의 "장애 요인"을 파악했습니다. 또한 "3.2.3.샘플: 현재 문제와 위험"에 설명된 대로 현재 문제를 요약했습니다.

여기에는 팀/제품 소유자가 원하는 특정 기능뿐만 아니라 비기능적 요구 사항도 모두 포함되어 있습니다. 다음 섹션에서는 몇 가지 관련 용어와 개념을 빠르게 수정한 다음 "샘플 프로젝트"의 첫 번째 제공 이터레이션을 위한 기능을 파악하는 데 집중합니다.

3.4.1. "사용자 스토리 매핑"과 "가치 분할" 사례란?

먼저 관련 용어를 모두 제대로 이해하고 있는지 다시 한번 확인해 보겠습니다.

3.4.1.1. 용어

일반적으로 "에픽"에는 중첩된 "기능"이 있을 수 있고, "기능"에는 중첩된 "사용자 스토리"가 있을 수 있습니다. 해당 용어는 모두 제품/시스템의 기능적 요구 사항뿐만 아니라 비기능적 요구 사항을 표현하는 데도 사용될 수 있습니다. "사용자 스토리"는 "대화별 요구 사항"을 위한 기법일 뿐입니다. 여기서 비기능적 요구 사항은 기능적 요구 사항과 함께 대화의 한 카테고리에 불과합니다. 이는 모두 제공 항목의 한 측면입니다.

에픽:

- 이해관계자/고객에게 새로운 제품, 솔루션, 서비스를 제공하는 대규모 이니셔티브
- 대규모 기능 모음으로 구성

기능:

- 제품 소유자가 관심을 갖는 역량
- 사용자/이해관계자에게 가치 제공
- 다수의 사용자 스토리에 의해 실현됨

사용자 스토리:

- 사용자/이해관계자의 개별적인 요구 사항을 나타냄
- 사용자/이해관계자에게 가치가 있는 기능/비기능적 측면을 설명함
- 프로젝트 계획의 세부 항목 역할을 함
- 프로젝트에서 최소 단위의 가치 증분을 나타냄
- 프로젝트 팀에서 목표 대화의 포커스로 사용하기에 편리함

3.4.1.2. "사용자 스토리 매핑" 및 "가치 분할"

"사용자 스토리 매핑"은 기존의 애자일 백로그 사례에서 발전된 방식입니다. 표준 애자일 제공 사례를 주도할 수 있는 경량 릴리스 계획을 만드는 데 효과적인 방법입니다. 올바르게 적용하면 다음과 같은 이점을 얻을 수 있습니다.

- 프로젝트 팀이 계획 기간 내에 제공할 수 있다고 생각하는 범위 항목의 백로그(스토리 또는 단순히 기능 제목으로 표현)
- 약 3개의 "제공 이터레이션"으로 "분할"되어 가장 가까운 배포 계획의 윤곽을 형성하는 백로그
- 계획의 첫 번째 "제공 이터레이션"이 작업을 시작하기에 충분한 세부 정보

참고: 사용자 스토리 매핑 및 가치 분할.[28]

다음은 "가치 분할"을 통해 식별된 초기(MVP) 제공 이터레이션이 포함된 사용자 스토리 매핑의 예입니다. **참고**: 아래 그림의 도메인은 가이드의 "샘플 프로젝트"와 다르며, 이 특정 예시는 가상의 현대적인 택시 서비스 도메인에 대한 것입니다. 다음 섹션의 "샘플 프로젝트"에 이 사례를 적용하겠습니다.

아래 매핑은 팀에서 "릴리스 1"과 "릴리스 2"의 이터레이션에 각각 미리 할당된 일부 "값 슬라이스"와 함께 식별한 것입니다.

그림 출처.[29]

위 그림의 녹색 카드는 제품의 특징(제품 기능)을 나타냅니다. 파란 카드는 제안된 사용자 스토리(프로젝트에서 최소 단위의 가치 증분)입니다.

일반적으로 애자일 백로그를 "분할(slice)"하는 방법에는 두 가지가 있습니다.

- **수평 분할**: 아키텍처 계층을 하나씩 작업
- **수직 분할**: 전체 아키텍처 스택을 가로지르는 엔드 투 엔드 기능별로 작업 분할

실제 가치 제공을 우선으로 한다면 일반적으로 "작업 단위"를 형성하면서 **수평 분할보다는 수직 분할을 사용하는 것이 좋습니다**.

[28] Open Practice Library의 목표 성과(Target Outcomes in Open Practice Library) https://openpracticelibrary.com/practice/target-outcomes/

[29] 스토리 맵, MVP, 우선순위 지정, 추정 - 6/7부| 애자일 제품 여정: 아이디어에서 시작까지(Story Map, MVP, Prioritization, Estimation – Part 6/7| Agile Product Journey, Idea to Inception) https://www.youtube.com/watch?v=D18HFmVLXQc

수평 분할은 일반적으로 대체 옵션으로 고려해야 합니다. 예를 들면 시스템의 단일 레이어에서만 작업할 수 있는 팀원만 다음번 제공 이터레이션에 투입할 수 있는 경우입니다. 예를 들면, 가상의 시스템의 비즈니스 로직 계층에서만 작업할 수 있어 다음 이터레이션에서 물리적으로 시스템의 프레젠테이션 계층뿐만 아니라 DB 계층도 개선할 수 없고, 이 작업을 수행할 전문가가 없는 경우입니다. 즉, 다른 두 계층에 변경이 필요한 "수직 분할"은 다음 이터레이션에서 작업할 수 없습니다. 이 경우 비즈니스 로직 계층과 관련된 수평 분할을 이터레이션에 포함시켜 프로젝트 작업을 계속 진행할 수 있습니다.

참고: 애자일을 위한 수평 또는 수직 분할.[30]

3.4.1.3. 사용자 스토리를 구현할 준비가 되었나요?

사용자 스토리를 구현할 준비가 되었다고 어떻게 확신할 수 있을까요? "택시비 지불" 기능 아래에 "현금"(사용자 스토리의 프로토타입)과 같은 항목이 있는 것만으로는 충분하지 않습니다. 프로젝트 팀이 다음과 같이 사용자 스토리를 자세히 설명할 때 다음 체크리스트 "준비 완료의 정의"(구현을 시작하는 데 사용할 사용자 스토리 설명에 대한 "준비 완료")를 유용하게 사용할 수 있습니다.

- 팀이 사용자 스토리를 이해함
- 사용자 스토리에 명확한 비즈니스 가치가 있음
- 사용자 스토리가 추정됨
- 사용자 스토리의 종속성이 식별됨
- 사용자 스토리가 작음
- 사용자 스토리의 승인 기준이 정의됨

"좋은 사용자 스토리"를 위한 또 다른 유명한 "세 가지 '**C**'의 기준은 다음과 같습니다.

Card(카드)

- 메모지 카드에 작성됨

- 예상치, 값, 메모 등으로 주석을 달 수 있음

Conversation(대화)

- 고객과의 대화를 통해 스토리의 세부 사항 파악

Confirmation(확인)

- 스토리 구현이 완료되었는지 확인하기 위해 승인 테스트가 정의됨

마지막으로 "좋은 사용자 스토리"의 기본 텍스트에 사용할 수 있는 몇 가지 합리적인 템플릿을 소개합니다.

- {역할}**로서** 저는 {비즈니스 목표를 달성}**하기 위해** {측정 가능한 자질을 가진 무언가를 할 수 있거나 가지고} **있습니다**.
- {비즈니스 목표를 달성}**하기 위해** {역할}은 {측정 가능한 자질을 가진 무언가를 하거나 가질 수} **있습니다**.

3.4.2. 샘플: 에픽, 기능 및 초기 사용자 스토리 맵

이전 섹션에서 설명한 대로 "샘플 프로젝트" 팀은 다음과 같은 MVP를 구축하기로 했습니다

애플리케이션 배포, 패치 적용, 관측성 및 성능 개선

이 MVP 아이디어의 원래 작성자는 팀에서 염두에 두던 기술 설계에 대한 인사이트를 제공했습니다. 샘플 프로젝트의 초기 범위로 다음 두 가지 에픽이 확인되었습니다

- **에픽: 애플리케이션의 롤링 업데이트**
  예를 들어, 새로운 애플리케이션 버전으로 Linux 장비를 롤링 업데이트하는 기능(프로세스 Linux 패치 적용에 통합)
- **에픽: 원격 상태 검사기**
  애플리케이션 상태를 관찰하는 상태 검사기 애플리케이션(사용자와 유사한 원격 구성 요소)

---

[30] 애자일을 위한 수평 또는 수직 분할(Horizontal or vertical slicing for agile?) https://www.youtube.com/watch?v=jQg27pFGmWA

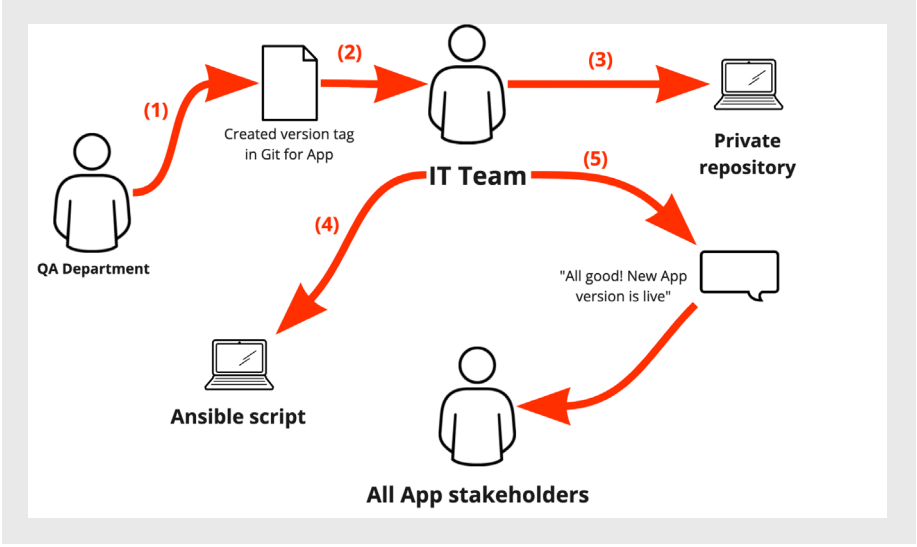
이 시점에서 "샘플 프로젝트" 팀은 원하는 상태에 대한 도메인 스토리텔링 세션 "라운드 2"를 가지는 것이 합리적이라 판단했습니다. 이 세션의 목적은 MVP 또는 그 이후에 어떤 기능을 추가해야 하는지/추가할 수 있는지를 더 명확히 하는 것입니다.

3.4.2.1. 샘플: "애플리케이션 새 버전 배포"에 대한 도메인 스토리 (원하는 상태)

섹션 "3.2.2. 샘플: MBPM - 현재 프로세스 흐름, 소유자, 메트릭"에서 설명한 대로 "샘플 프로젝트" 팀은 배포 프로세스에서 다음과 같은 "장애 요인"을 확인했습니다.

- ...
- 배포에 사용할 유지관리 기간에 대해 모두와 합의
- 합의된 유지관리 기간이 시작될 때까지 기다렸다가 애플리케이션의 Linux 서비스 중지
- ...
- 방금 배포한 애플리케이션에 대해 수동으로 간단한 상태 점검 수행
- ...

"샘플 프로젝트" 팀은 MVP 아이디어의 원래 작성자인 분야별 전문가(SME)를 초대했습니다. 무엇보다도 팀은 현재 확인된 "장애 요인"과 관련된 개선 사항에 집중하고자 합니다. 팀은 세션을 통해 원하는 프로세스 상태를 설명하는 다음 다이어그램을 제안했습니다.



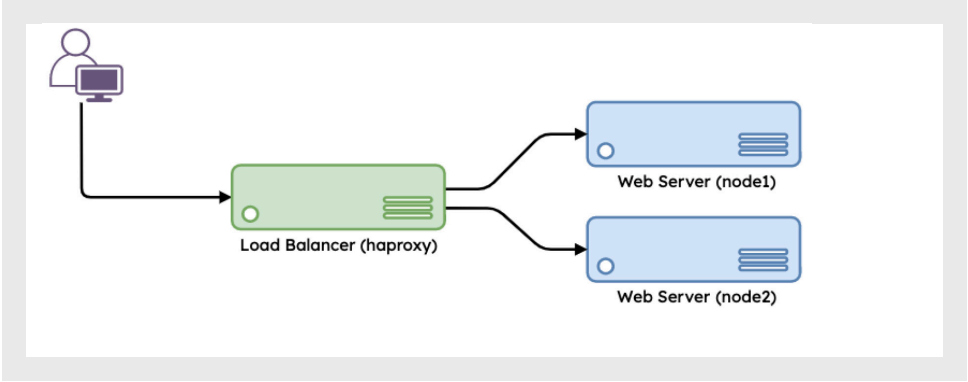
1. QA 부서에서 Git 버전 제어 시스템에서 애플리케이션의 새 릴리스에 태그를 지정합니다.
2. QA 부서에서 IT 팀에 배포할 수 있는 새 릴리스 가용성에 대해 알립니다.
3. IT 팀이 새로운 종속성을 식별하고 해당 아티팩트를 비공개 리포지토리에 추가합니다.
4. IT 팀이 새 애플리케이션 버전을 배포하기 위해 자동화 Ansible 스크립트를 실행합니다.
5. 스크립트가 새 애플리케이션 버전에 대한 자동화된 테스트를 "통과"로 보고한 경우, IT 팀은 모든 이해 관계자에게 새 애플리케이션 버전이 가동되었음을 알립니다.

보다시피 위의 새 다이어그램에는 이전에 식별된 "장애 요인"이 없습니다. 이는 아이디어 작성자가 다음과 같이 설명한 Ansible 스크립트와 일부 아키텍처가 변경되었기 때문입니다.

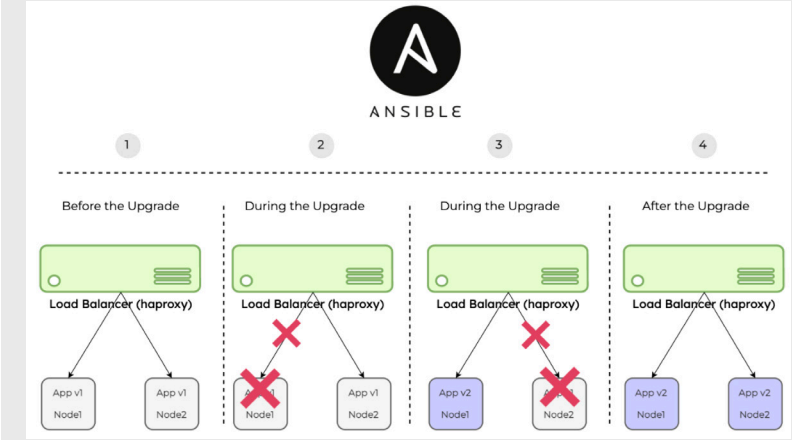
- 로드 밸런서(예: HAProxy) 뒤에는 해당 전용 Linux 장비에서 실행되는 애플리케이션의 여러 인스턴스가 있습니다.[31]
  - 모든 인스턴스가 사용자 요청을 처리하므로 사용자에 대한 서비스 가용성이 향상됩니다. 예를 들어, 인스턴스 중 하나가 다운되더라도 다른 인스턴스는 계속 요청을 처리합니다.
  - 더 많은 CPU가 사용되므로 사용자 요청 처리가 더 빠릅니다.
- 로드 밸런서 뒤에서 새 버전의 애플리케이션 인스턴스가 롤링 배포됩니다. 특히 인스턴스가 하나씩 오프라인 상태가 되었다가 다시 "활성" 상태로 전환되므로 전체 롤링 업데이트 프로세스 중에 다른 인스턴스가 사용자 요청을 처리할 수 있습니다.

다음은 전문가가 "샘플 프로젝트" 팀에 설명한 내용을 시각화한 몇 가지 다이어그램입니다(그림 출처: 실제 자동화를 위한 Ansible)[32]

로드 밸런서 뒤에서 여러 Linux 서버에 호스팅되는 애플리케이션



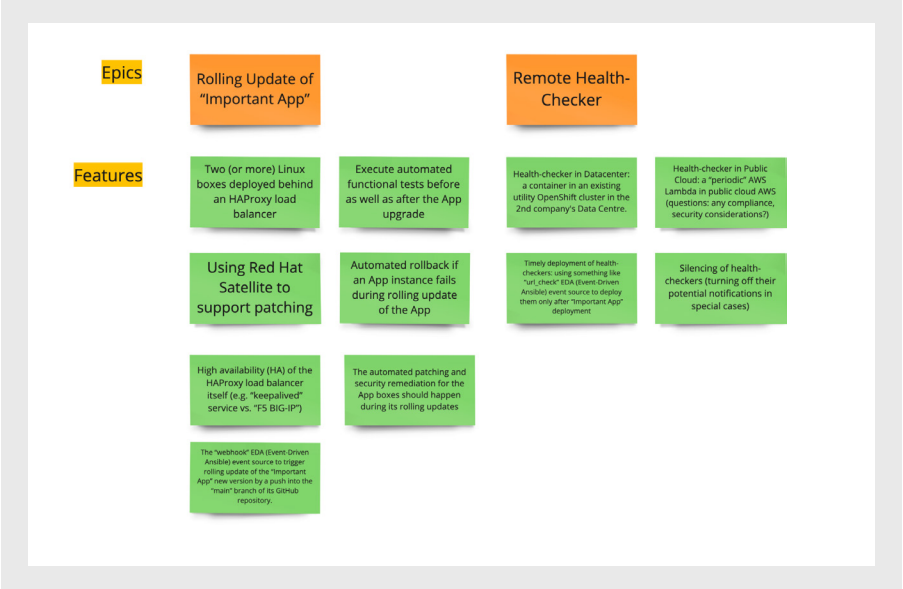
Ansible을 사용한 애플리케이션 롤링 업데이트



3.4.2.2. 샘플: 초기 사용자 스토리 맵

"샘플 프로젝트" 팀은 이전 섹션에서 논의한 개선 사항을 추가로 조사하는 데 하루를 보냈습니다. 그런 다음 다른 세션에서 다음과 같이 사용자 스토리 맵을 구축하기 시작했습니다

[31] 위키피디아의 HAProxy 항목(HAProxy on Wikipedia) https://en.wikipedia.org/wiki/HAProxy
[32] 기네시 마다파람바스의 "실제 자동화를 위한 Ansible(Ansible for Real-Life Automation by Gineesh Madapparambath") https://amzn.asia/d/flHSDjh



에픽: 애플리케이션의 롤링 업데이트
기능 (이해 관계자가 관심을 갖는 기능):

- 둘 이상의 Linux 장비를 HAProxy 로드 밸런서 뒤에 배포
- 애플리케이션 업그레이드 전과 후에 자동화된 기능 테스트 실행
- 애플리케이션 장비에 대한 자동화된 패치 적용과 보안 문제 해결은 롤링 업데이트 중에 이루어져야 함
- Satellite를 사용하여 패치 적용 지원
- 애플리케이션 롤링 업데이트 중 애플리케이션 인스턴스에 장애 발생 시 롤백 자동화
- GitHub 리포지토리의 "메인" 브랜치로 푸시하여 애플리케이션 새 버전의 롤링 업데이트를 트리거하는 "웹후크" EDA(Event-Driven Ansible) 이벤트 소스 참고:
  Event-Driven Ansible + Gitops.[33]
- HAProxy 로드 밸런서 자체의 HA(예: "keepalived" 서비스와 "F5 BIG-IP" 비교)

에픽: 원격 상태 검사기
기능 (이해 관계자가 관심을 갖는 기능):

실제 사용자가 영향을 받기 전에 사용자가 직면하는 문제를 감지하는 데 도움이 되는, 사용자와 유사한 (원격) 상태 검사기 구성 요소. 이는 New Relic의 "합성" 클라이언트 또는 AWS CloudWatch에서 지원되는 "카나리아"와 유사합니다.

- 데이터센터의 상태 검사기: 두 번째 회사의 데이터센터에 있는 기존 유틸리티 Red Hat OpenShift 클러스터의 컨테이너
- 퍼블릭 클라우드의 상태 검사기: 퍼블릭 클라우드 AWS의 "주기적" AWS Lambda(질문: 컴플라이언스, 보안 고려 사항이 있는가?)
- 상태 검사기의 적시 배포: "url_check" EDA 이벤트 소스 등을 사용하여 애플리케이션 배포 후에만 배포
- 상태 검사기 무음화(특별한 경우에 잠재적 알림 끄기)

[33] Event-Driven Ansible + Gitops https://www.youtube.com/watch?v=Bb51DftLbPE

3.4.3. 첫 번째 제공 이터레이션에 들어가는 기능은 무엇인가요?

자동화는 거의 항상 유용하지만 구현에는 비용이 들기 때문에 계획 항목의 우선순위를 정해야 합니다. 이러한 우선순위 지정은 다음과 같은 경우에 필요합니다.

- 투입된 노력의 **가치를 극대화**하기 위해
- **"미완료 작업"을 최대한 활용**하기 위해(즉, 무의미한 일에 애쓰지 않으려면)
- 실제로 무가치한 **산출물 "낭비"를 줄이기** 위해 "산출물"은 "성과"와 다릅니다. 일부 산출물은 시간이 지날수록 유지관리가 필요하지만 이해관계자/고객에게 실질적인 가치를 제공하지 못하기 때문에 실제로는 "낭비"입니다.

다음 섹션에서는 우선순위 결정을 지원하는 네 가지 사례를 다루며, 특히 "Kano 모델" 사례를 집중적으로 살펴봅니다. 여기에서 "샘플 프로젝트" 팀이 후자를 가장 고객 중심적인 프로젝트에 사용하기로 한 것을 볼 수 있습니다.

3.4.3.1. 우선순위 지정 사례

"영향 및 노력 우선순위 지정(매트릭스)" 사례는 **태스크의 영향과 구현에 필요한 노력을 평가**합니다. 팀의 필요에 적합한 경우 더 많은 노력이 들고 더 큰 영향을 미치는 개선 사항보다, 빠른 성과를 낼 수 있는 개선 사항을 선택할 수 있습니다. 참고: 영향 및 노력 우선순위 지정(매트릭스).[34]

"우선순위 슬라이더" 사례는 향후 활동에 집중하기 위해 **상대적인 우선순위**에 대한 대화를 지원하는 툴입니다. 참고: 우선순위 슬라이더.[35]

"How-Now-Wow" 매트릭스 기반 사례는 위의 방식과 도표상 유사합니다. 특히 **혁신적인 아이디어를 분류**하는 데 유용합니다. 참고: How-Now-Wow 우선순위 지정(매트릭스).[36]

좀 더 짜임새 있는 **"Kano 모델"** 사례는 여기에 언급된 네 가지 사례 중 **가장 고객 중심적인** 사례입니다. 아래에서 Kano 모델에 대해 자세히 설명하겠습니다.

도쿄과학대학의 명예 교수인 노리아키 카노(Noriaki Kano)는 제품의 필수 속성과 고객/이해관계자 차별화 속성을 구분하는 간단한 순위 체계를 고안했습니다. 카노 교수에 따르면 소비자 선호도는 다음 다섯 가지 카테고리로 분류할 수 있습니다:

- **반드시 필요한 것(일명 필수적인 기본 욕구로 부재 시 불만족)** - 잘 갖춰져 있으면 고객은 중립적인 태도를 취하지만, 부재 또는 부실한 경우 고객은 매우 실망
- **일차원적 특성(일명 욕구, 만족 요소, 놀라운 요소)** - 존재하면 만족, 존재하지 않으면 불만족으로 이어짐
- **매력 요소(일명 유희적 요소)** - 실현 시 만족감을 제공하지만, 실현되지 않더라도 불만족을 유발하지는 않음
- **무관심 요소(일명 중립 요소)** - 종종 간과되는 기능으로, 달성하기 위해 노력하지만 고객에게 중요하지는 않은 기능
- **역품질 요소** - 실현 시, 불필요하게 과한 복잡성 등 실제로는 일부 고객에게 불만을 야기

[34] Open Practice Library의 영향 및 노력 우선순위 지정 매트릭스(Impact & effort prioritization (Matrix) in Open Practice Library) https://openpracticelibrary.com/practice/impact-effort-prioritization-matrix/

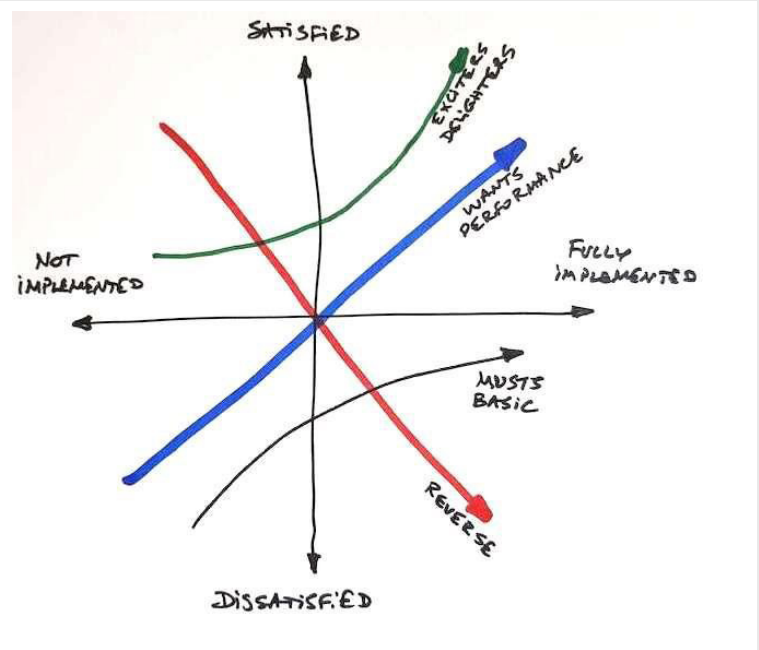
[35] Open Practice Library의 우선순위 지정 슬라이더(Priority Sliders in Open Practice Library) https://openpracticelibrary.com/practice/priority-sliders/

[36] Open Practice Library의 How-Now-Wow 우선순위 지정 매트릭스(How-Now-Wow Prioritization (Matrix) in Open Practice Library) https://openpracticelibrary.com/practice/how-now-wow-prioritization-matrix/

제품 기획 시 참고할 수 있는 일반적인 "Kano 기반" 권장 사항은 다음과 같습니다.

- **모든 필수 요구 사항을 포함**합니다. 이는 꼭 필요합니다.
- **만족스러운 수준의 "일차원적 특성"("욕구")과 "매력적"("유희적")** 기능을 포함하고 충분히 최적화합니다. 고객/이해관계자가 더 많은 것을 요구하더라도 괜찮습니다. 고객이 제품에서 가치를 발견한다면 이러한 기능을 바로 최적화할 필요는 없습니다. 다음 몇 번의 이터레이션에서 적절하게 최적화/추가할 수 있습니다.

다음 "Kano 매트릭스"를 사용하여 계획 중인 제품 기능을 표시하고 프로젝트 팀, 고객, 이해관계자와 기능에 맞는 카테고리를 논의할 수 있습니다. 그런 다음 위에서 언급한 "Kano 기반" 권장 사항에 따라 다음 "제공 이터레이션"에 적절한 기능을 포함시키면 됩니다. 참고: Kano 모델.[37]



3.4.3.2. 샘플: MVP 정의 범위, 가치 슬라이스

"샘플 프로젝트" 팀은 지금까지 파악한 초기 기능의 우선순위를 정하고 분류하기 위해 "Kano 모델" 방식을 적용하기로 했습니다. 팀은 진정한 고객 중심 방식을 마음에 들어했습니다. MVP에 어떤 요소가 포함되며 나중에 구현할 수 있는 것은 무엇일까요?

3.4.3.2.1. 샘플: Kano 모델 적용

특히 팀은 참가자들 의견을 암묵적으로 측정하기 위해 Kano의 표준화된 설문지를 사용하기로 했습니다. 이는 샘플 프로젝트의 이해관계자를 결정론적인 "유한한" 방식으로 이해하는 데 도움이 될 것입니다. 따라서 참가자는 각 제품 기능에 대해 두 가지 질문("긍정적" 방식으로 구성된 한 가지 질문과 "부정적" 방식으로 구성된 다른 한 가지 질문)에 답해야 합니다.

다음은 "3.4.2.2. 샘플: 초기 사용자 스토리 맵" 섹션에 설명된 각 기능에 대해 각 참가자에게 제공되는 설문지입니다.

[37] Open Practice Library의 Kano 모델(Kano Model in Open Practice Library) https://openpracticelibrary.com/practice/kano-model/

	좋다	기대된다	중립적이다	감수할 수 있다	싫다
"긍정적" 질문					
제품에 ...이 있다면 어떻게 느끼시겠습니까?					
...이 더 많다면 어떻게 느끼시겠습니까?					
"부정적" 질문					
제품에 ...이 *없다면* 어떻게 느끼시겠습니까?					
...이 더 적다면 어떻게 느끼시겠습니까?					

그런 다음 진행자가 이 설문지를 수거하고 Kano 교수가 제안한 다음 표에 따라 해당 기능을 "Kano 카테고리"에 배치하기 위한 참가자의 "투표"를 계산했습니다.

"긍정적" 질문		"부정적" 질문		카테고리
기대된다	+	싫다	→	필수
좋다	+	싫다	→	일차원적 특성(욕구, 만족 요소)
좋다	+	중립적이다	→	매력 요소
중립적이다	+	중립적이다	→	무관심 요소
싫다	+	기대된다	→	역품질 요소(불만족 유발)

다음 기능을 예로 들어보겠습니다.

- GitHub 리포지토리의 "메인" 브랜치로 푸시하여 애플리케이션 새 버전의 롤링 업데이트를 트리거하는 "웹후크" EDA 이벤트 소스.

참가자 중 한 명이 다음과 같이 설문지를 작성했습니다

	좋다	기대된다	중립적이다	감수할 수 있다	싫다
"긍정적" 질문					
제품에 ...이 있다면 어떻게 느끼시겠습니까?	✓				
"부정적" 질문					
제품에 ...이 *없다면* 어떻게 느끼시겠습니까?					✓

따라서 해당 참가자의 투표는 다음과 같이 계산되었습니다

"좋다"("긍정적" 질문에 해당) **+ "싫다"**("부정적" 질문에 해당) **=> 일차원적 특성(일명 욕구, 만족 요소, 놀라운 요소)** – 존재하면 만족, 존재하지 않으면 불만족으로 이어짐

설문조사 참가자들이 분류한 기능을 요약하면 다음과 같습니다.

에픽: 애플리케이션의 롤링 업데이트

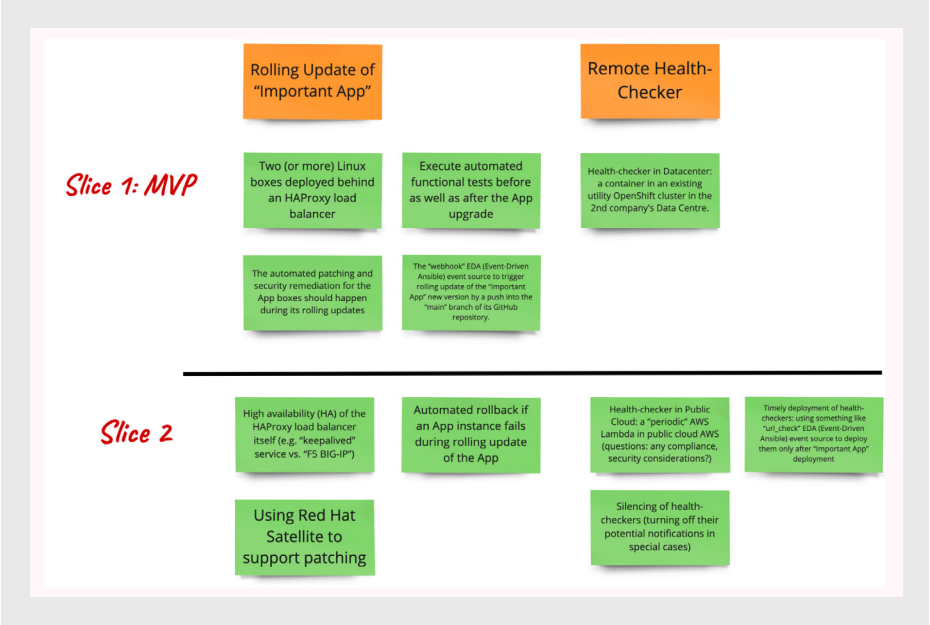
- **[필수]** 둘 이상의 Linux 장비를 HAProxy 로드 밸런서 뒤에 배포
- **[필수]** 애플리케이션 업그레이드 전과 후에 자동화된 기능 테스트 실행
- **[필수]** 애플리케이션 장비에 대한 자동화된 패치 적용과 보안 문제 해결은 롤링 업데이트 중에 이루어져야 함
- **[일차원적 특성]** Satellite를 사용하여 패치 적용 지원
- **[일차원적 특성]** 애플리케이션 롤링 업데이트 중 애플리케이션 인스턴스에 장애 발생 시 롤백 자동화
- **[일차원적 특성]** GitHub 리포지토리의 "메인" 브랜치로 푸시하여 애플리케이션 새 버전의 롤링 업데이트를 트리거하는 "웹후크" EDA 이벤트 소스
- **[매력 요소]** HAProxy 로드 밸런서 자체의 HA(예: "keepalived" 서비스와 "F5 BIG-IP" 비교)

에픽: 원격 상태 검사기

- **[필수]** 데이터센터의 상태 검사기: 두 번째 회사의 데이터센터에 있는 기존 유틸리티 Red Hat OpenShift 클러스터의 컨테이너
- **[무관심 요소]** 퍼블릭 클라우드의 상태 검사기: 퍼블릭 클라우드 AWS의 "주기적" AWS Lambda(질문: 컴플라이언스, 보안 고려 사항이 있는가?)
- **[매력 요소]** 상태 검사기의 적시 배포: "url_check" EDA 이벤트 소스 등을 사용하여 애플리케이션 배포 후에만 배포
- **[매력 요소]** 상태 검사기 무음화(특별한 경우에 잠재적 알림 끄기)

3.4.3.2.2. 샘플: 가치 슬라이스

다음은 "샘플 프로젝트" 팀과 관련 이해관계자가 점진적 릴리스 전략을 수립하기 위해 높은 가치를 분할한 방법입니다. 이들은 가치 있는 성과를 제공하는 데 계속 집중하고자 합니다.



4. 효율적인 자동화 제공

가치 있는 엔지니어링 솔루션의 핵심 측면은 다음과 같으며, 이는 항상 균형을 이루어야 합니다.

- 비즈니스 관점에서 고객에게 제공하는 솔루션의 가치
- 프로덕션 환경에서 솔루션의 상태
- 솔루션에 대한 요구 사항
- 솔루션의 설계
- 솔루션의 개발 및 제공
- 자동화 수준
- 보안 및 컴플라이언스
- 솔루션 비용

사이트 신뢰성 엔지니어링(SRE)과 같은 분야는 자동화를 통해 IT 솔루션을 관리하는 데 도움이 되는 일련의 원칙과 사례를 정의하며, 이는 반복적인 수동 이터레이션 작업보다 확장성과 지속 가능성이 높습니다. 특히 IT 솔루션의 다음 측면들 간에 원하는 수준의 균형을 유지하는 데 SRE를 적용하는 것이 유용합니다.

- 프로덕션 환경에서 솔루션의 상태
- 솔루션의 개발 및 제공

예를 들어, SRE는 시스템 안정성 위험을 관리하는 방법을 권장합니다. 이러한 위험은 IT 솔루션 자체의 계획된 변경 또는 솔루션이 실행되는 환경의 일부 변경으로 인해 발생할 수 있습니다. SRE는 위험을 어느 정도 허용할지를 결정할 때 운영 팀과 개발자 간 의견을 절충할 때 이해 관계를 배제하는 특별한 "오류 예산" 메트릭을 도입합니다. 참고: 위험 수용하기.[38]

원래 프로젝트 계획이 갑자기 변경되거나, 매우 혁신적이거나 다이나믹한 환경에서 프로젝트가 진행되는 경우 다음과 같은 불균형이 발생할 위험이 상당히 높습니다.

- 비즈니스 관점에서 고객에게 제공하는 솔루션의 가치
- 솔루션에 대한 요구 사항
- 솔루션의 설계
- 솔루션의 개발 및 제공

이러한 경우 안타깝게도 “원래 계획대로 모든 것을 구축한다”는 제공 방식은 장기적으로는 효율적이지 않을 수 있습니다.

프로젝트 팀의 입장에서는 프로젝트 성공을 위해 좀 더 적응력 있고 민첩하게 대응하는 것이 합리적입니다. 경험상 프로젝트의 약 80%가 이런 식입니다. 따라서 다음 섹션에서는 이 가이드의 컨텍스트를 간단히 되짚어본 후 "애자일"에 대한 체계적인 접근 방식을 소개합니다.

4.1. "제공 루프"와 애자일 사례 소개

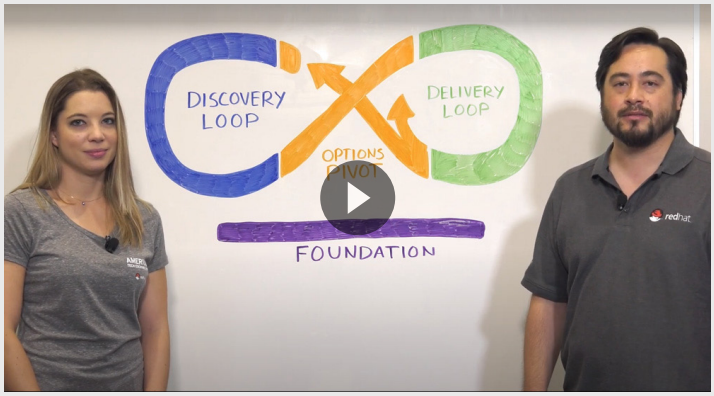
앞의 큰 섹션 3에서 다룬 사례는 다음과 같습니다.

- “검색 루프”
  - 동기 부여 매핑
  - 큰 그림 이벤트 스토밍
  - 도메인 스토리텔링
  - MBPM
  - 사용자 스토리 매핑
  - 목표 성과

[38] 위험 수용하기(Embracing risk) https://sre.google/sre-book/embracing-risk/

- “옵션 피벗” 단계
  - Kano 모델
  - 가치 분할
  - MVP

이 섹션에서는 앞서 언급한 같은 동영상의 두 번째 부분인 "Open Practice Library로 성과 기반 제공(Outcome-driven delivery with Open Practice Library)" 동영상에서 설명한 "제공 루프"와 관련된 몇 가지 사례에 초점을 맞춥니다.[39]



이 큰 섹션 4의 소개에서 언급한 것처럼, 프로젝트 성공을 위해 프로젝트 팀의 "민첩성(agile)”을 원하는 경우가 많습니다. 점차 민첩한 실무자들은 IT 솔루션의 애자일 개발 및 제공을 효과적으로 지원하기 위한 몇 가지 공통된 의식/사례를 확인했습니다. 이러한 사례가 "제공 루프"에 어떻게 매핑되는지는 다음을 통해 알 수 있습니다.

- 백로그 세분화(옵션 사례)
- "증분 계획" 또는 "스프린트 계획"(제공 사례)
- 일일 스탠드업 미팅(기본 사례)
- 쇼케이스 또는 데모(제공 사례)
- 후향적 평가(기본 사례)
- "완료의 정의"(기본 사례)

다음 섹션에서 사례에 대한 자세한 내용을 확인할 수 있습니다.

4.1.1. 백로그 세분화(옵션 사례)

"백로그 세분화"는 프로젝트 팀이 제품 소유자 또는 사정을 잘 아는 이해관계자와 협력하여 현재 증분(이터레이션)의 항목과 백로그의 최우선 순위 항목을 검토하는 활동으로, 정해진 시간 내에 완료해야 합니다. 백로그 세분화 중에 팀은 제공할 각 항목의 세부 사항과 승인 기준을 명확히 합니다. 참고: 백로그 세분화.[40]

4.1.2. "증분 계획" 또는 "스프린트 계획"(제공 사례)

증분 계획은 프로젝트 팀이 제품 소유자/사정을 잘 아는 이해관계자와 협업하여 일련의 명확한 작업을 수행하고 다음 증분(이터레이션)에 대한 목표를 설정하는 활동으로, 정해진 시간 내에 완료해야 합니다. 참고: 이터레이션(스프린트) 계획.[41]

[39] Open Practice Library로 성과 기반 제공(Outcome-driven delivery with Open Practice Library) https://www.youtube.com/watch?v=N4mBIZg8MnQ
[40] Open Practice Library에서 백로그 세분화(Backlog refinement in Open Practice Library) https://openpracticelibrary.com/practice/backlog-refinement/
[41] Open Practice Library에서 이터레이션(스프린트) 계획(Iteration (Sprint) Planning in Open Practice Library) https://openpracticelibrary.com/practice/iteration-planning/

4.1.3. 일일 스탠드업 미팅(기본 사례)

일일 스탠드업 미팅은 프로젝트 팀이 합의된 증분(이터레이션) 목표를 위한 활동을 조율하기 위해 시간을 할애하는 짧은 활동입니다. 팀원들이 모여 진행 중인 작업의 상태를 공유하고 진행에 방해가 되는 장애 요인에 집중합니다. 참고: 일일 스탠드업 미팅.[42]

4.1.4. 쇼케이스 또는 데모(제공 사례)

쇼케이스는 프로젝트 이해관계자 및 당사자에게 증분(이터레이션) 기간 동안 수행한 최근 작업의 데모를 보여주고 피드백을 수집하는 활동입니다. 쇼케이스(데모)는 제품/프로젝트 라이프사이클의 주요 이정표에서 수행하거나 필요에 따라 일정을 예약할 수도 있습니다. 참고: 쇼케이스.[43]

4.1.5. 후향적 평가(기본 사례)

증분(이터레이션) 막바지에 프로젝트 팀은 후향적 평가를 통해 협업 방식을 되짚어보고, 분석하고, 조정합니다. 후향적 평가는 프로젝트 팀의 성과에 영향을 미치는 사실, 관찰, 느낌을 드러내 보이는 데 도움이 됩니다. 이는 모두 관련 프로젝트 팀의 개선을 위한 아이디어를 제안하고, 수집하고, 합의하는 데 활용됩니다. 참고: 후향적 평가.[44]

4.1.6. "완료의 정의"(기본 사례)

"완료의 정의" 사례는 프로젝트 팀과 상황을 잘 아는 이해관계자 간의 이해와 공유된 기대치를 조정합니다. 예를 들어 "완료의 정의"에 대해 다음과 같이 합의할 수 있습니다.

- 사용자 스토리가 승인 기준에 따라 테스트됨
- 사용자 스토리가 관찰된 오류 없이 전달됨
- 사용자 스토리가 이해관계자에게 데모로 시연됨
- 사용자 스토리를 제품 소유자가 승인함
- 사용자 스토리와 관련된 문서가 업데이트됨

참고: "완료의 정의"[45]

4.2. 샘플: 첫 번째 이터레이션 계획에 따라 제공하기

"옵션 피벗" 단계의 성과에 이어 활동을 더욱 세분화하고 계획하기 위해 "샘플 프로젝트" 팀과 제품 소유자는 다음 두 가지 사례를 차례로 적용하기로 했습니다.

- 백로그 세분화(옵션 사례)
- "증분 계획" 또는 "스프린트 계획"(제공 사례)

예를 들어, 이들은 "샘플 프로젝트"에 다음과 같은 장애 요소가 있음을 발견했습니다(섹션 3.1.3: "샘플 조직을 위한 샘플 프로젝트: 동기 부여 매핑..." 참조).

> 애플리케이션에 HA를 구현해야 하는 엔지니어의 온보딩, 패치 프로세스 개선은 보안상의 이유로 2~4주가 소요됩니다. 또한 프로젝트 팀이 사용할 수 있는 지역 데이터센터의 개발(프로덕션 제외) 환경을 위한 컴퓨팅 용량이 현재 부족합니다. 추가 용량은 3~6주 내에 프로비저닝할 수 있습니다.

---

[42] Open Practice Library의 일일 스탠드업 미팅(Daily standup in Open Practice Library) https://openpracticelibrary.com/practice/daily-standup/

[43] Open Practice Library의 쇼케이스(Showcase in Open Practice Library) https://openpracticelibrary.com/practice/showcase/

[44] Open Practice Library의 후향적 평가(Retrospectives in Open Practice Library) https://openpracticelibrary.com/practice/retrospectives/

[45] Open Practice Library에서 완료의 정의(Definition of Done in Open Practice Library) https://openpracticelibrary.com/practice/definition-of-done/

따라서 "샘플 프로젝트" 팀과 제품 소유자는 다음과 같은 추가 작업 항목에 합의하기로 했습니다.

> 데이터센터의 현재 시스템에 영향을 주지 않고 MVP 배포를 지원하는 환경을 프로비저닝할 곳을 찾습니다. 엔지니어링 팀은 즉시 시작할 수 있어야 합니다.

요약하자면, 엔지니어링 팀은 몇 가지 의사 결정을 세분화하고 첫 번째 제공 이터레이션에 대한 실행 계획에 합의했습니다. 다음 두 섹션에서 이를 확인하시기 바랍니다.

4.2.1. 샘플: 이터레이션을 위한 의사 결정

MVP는 다음과 같이 합의됩니다.

- 애플리케이션을 위한 두 Linux 장비를 HAProxy 로드 밸런서 뒤에 배포
  - 이러한 컴퓨팅 리소스 구성으로 컴퓨팅 용량이 증가함
  - 이 때 새로운 애플리케이션 버전의 자동화된 롤링 업데이트를 지원해야 함
  - (롤링 업데이트 중에) 두 Linux 장비의 자동화된 패치 적용 및 보안 문제 해결을 지원해야 함
  - MVP의 범위에서는 HAProxy 뒤에서 새 버전의 애플리케이션을 롤링 배포/업데이트하는 GitOps 스타일의 시작을 지원합니다. 특히
    - EDA를 통해 애플리케이션 리포지토리에 대한 GitHub의 "푸시" 이벤트를 수신하는 표준 "ansible.eda.webhook" 이벤트 소스를 구현합니다. 이러한 이벤트가 수신되면 EDA는 새 애플리케이션 버전에 대한 자동화된 롤링 업데이트를 시작해야 합니다.
    - 롤백 지원은 구현되지 않음
- 애플리케이션 업그레이드 전과 후에 자동화된 기능 테스트 실행
- 애플리케이션에 대한 사용자와 유사한 상태 검사기:
  - 두 번째 회사의 데이터센터에 있는 기존 유틸리티 Red Hat OpenShift 클러스터의 컨테이너입니다. 상태 검사기 애플리케이션은 JavaScript/Typescript로 구현됩니다.
  - 이 애플리케이션은 적시에 배포되어야 하지만, 지금은 EDA에서 제공하는 "url_check" 이벤트 소스를 사용하지 않고 있습니다.

4.2.2. 샘플: 이터레이션을 위한 실행 계획

- HAProxy 로드 밸런서 구성 조사
- 애플리케이션에 대한 기능 테스트를 정의하고 Ansible에서 구현
- 다음을 수행하는 Ansible Playbook 설계 및 구현
  - 새 애플리케이션 버전의 자동화된 롤링 업데이트
  - (롤링 업데이트 중에) 두 Linux 장비의 자동화된 패치 적용 및 보안 문제 해결
  - 애플리케이션 업그레이드 전과 후에 자동화된 기능 테스트 실행
- 애플리케이션의 GitOps 스타일 롤링 배포를 위한 EDA 구성
  - Rulebook의 구현 및 배포.
    표준 "ansible.eda.webhook" 이벤트 소스는 EDA를 통해 애플리케이션 리포지토리에 대한 GitHub의 "푸시" 이벤트를 수신합니다. 이러한 이벤트가 수신되면 EDA는 위에서 언급한 Ansible Playbook을 사용하여 애플리케이션의 자동화된 롤링 업데이트를 시작해야 합니다.
- 데이터센터의 현재 시스템에 영향을 주지 않고 MVP 배포를 지원하는 환경을 프로비저닝할 곳을 찾습니다. 엔지니어링 팀은 즉시 시작할 수 있어야 합니다.

  참고: 이 가이드의 필요에 따라 AWS 퍼블릭 클라우드에 MVP용 Linux 장비를 배포합니다. 물론 모든 실제 요구 사항과 규정을 충족하는 경우 "샘플 프로젝트" 팀에서 "실제 결정"을 내릴 수도 있습니다.

  - 애플리케이션 인스턴스 및 HAProxy 로드 밸런서 인스턴스를 배포하기 위한 AWS 구성
  - 데이터센터와 같은 3개의 EC2 Linux 인스턴스를 배포하기 위한 Terraform 코드 구현

- 상태 검사기 애플리케이션의 경우:
  - (JavaScript/Typescript에서) 상태 검사기 애플리케이션의 설계 및 구현
  - Red Hat OpenShift와의 통합을 위한 API 조사

    참고: "샘플 프로젝트" 팀은 다음에 배포할 계획을 세우고 있습니다.
    "두 번째 회사의 데이터센터에 있는 기존 유틸리티 Red Hat OpenShift 클러스터"
    이 프로젝트는 가상의 "샘플 조직"을 위한 것이지만 배포 방법을 보여드리겠습니다. 두 번째 회사의 데이터센터에 다음이 있다고 가정해 보겠습니다.
    표준 "개발자 샌드박스" Red Hat OpenShift 환경
    참고: 개발자 샌드박스에서 무료로 탐색 시작하기.[46]
  - 두 번째 회사의 데이터센터에서 상태 검사기 배포를 위한 유틸리티 Red Hat OpenShift 클러스터 구성

    참고: 이 가이드에서는 실제로
    표준 "개발자 샌드박스" Red Hat OpenShift 환경 구성을 사용합니다.

4.2.3. 샘플: 이터레이션 도중

참고: 가능한 MVP 구현은 MIT 라이센스에 따라 게시된 다음 개인 리포지토리에서 찾을 수 있습니다.
https://github.com/mikhailknyazev/automation-samples

이터레이션 중에 "샘플 프로젝트" 팀은 다음과 같이 실행 계획을 따랐습니다.

- **HAProxy 로드 밸런서 구성 조사**
  유용한 링크:
  - Ansible 역할(Ansible Role) - https://github.com/geerlingguy/ansible-role-haproxy
  - Packt에서 게시한 "실제 자동화를 위한 Ansible"의 코드 리포지토리(Code repository for "Ansible for Real-Life Automation", published by Packt)
    https://github.com/PacktPublishing/Ansible-for-Real-life-Automation
  - HAProxy 커뮤니티 에디션(HAProxy community edition) https://www.haproxy.org

```
---
- name: Deploy Load Balancer using HAProxy
  hosts: loadbalancer
  become: yes
  vars:
    haproxy_frontend_name: 'hafrontend'
    haproxy_backend_name: 'habackend'
    haproxy_backend_servers:
     - name: node1
       address: node1:80
     - name: node2
       address: node2:80
  tasks:

    - name: Install haproxy
      include_role:
        name: geerlingguy.haproxy
```

[46] 개발자 샌드박스에서 무료로 탐색 시작(Start exploring in the Developer Sandbox for free)
https://developers.redhat.com/developer-sandbox

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/haproxy-plus-health-checker.yaml

- **애플리케이션에 대한 기능 테스트를 정의하고 Ansible에서 구현**

```yaml
- name: Verify deployment
  hosts: "web"
  become: no
  tasks:
    - name: Verify application health
      delegate_to: localhost
      ansible.builtin.uri:
        url: http://{{ inventory_hostname }}
        status_code: 200
        return_content: true
      register: response
    - name: Check if 'Serving from...' is in the response
      delegate_to: localhost
      ansible.builtin.assert:
        that: "'Serving from {{ inventory_hostname }}' in response.content"
        fail_msg: "The phrase 'Serving from {{ inventory_hostname }}' was not found in the response"
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/app-deploy.yaml

- **다음을 수행하는 Ansible Playbook 설계 및 구현:**
  - **새 애플리케이션 버전의 자동화된 롤링 업데이트**
  - **(롤링 업데이트 중에) 두 Linux 장비의 자동화된 패치 적용 및 보안 문제 해결**
  - **애플리케이션 업그레이드 전과 후에 자동화된 기능 테스트 실행**

```yaml
---
- name: Rolling Update
  hosts: "web"
  become: yes
  serial: 1
  vars:
    haproxy_backend_name: 'habackend'
    application_repo: 'https://github.com/mikhailknyazev/automation-samples'
    application_branch: main
    subfolder_path: sample-app
    application_path: /var/www/html

  tasks:

    - name: Preparing rolling deployment of sample App
      ansible.builtin.debug:
        msg: >
          Branch: {{ application_branch }}
          Subfolder: {{ subfolder_path }}
          Repo: {{ application_repo }}
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/rolling-update.yaml

- **애플리케이션의 GitOps 스타일 롤링 배포를 위한 EDA 구성**

```
yum update -y
yum install -y vim git wget java-17-amazon-corretto-devel sshpass

yum -y groupinstall "Development Tools"
yum -y install openssl-devel bzip2-devel libffi-devel

wget https://www.python.org/ftp/python/3.9.17/Python-3.9.17.tgz
tar xvf Python-3.9.17.tgz
cd Python-*/
./configure --enable-optimizations
make altinstall

cd ~
python3.9 --version
pip3.9 --version

/usr/local/bin/python3.9 -m pip install --upgrade pip
pip3.9 --version
pip3.9 install ansible-rulebook ansible ansible-runner wheel openshift

# Note: we are opening port 5000 for incoming webhook calls on port 5000
yum install -y firewalld
systemctl enable --now firewalld
firewall-cmd --add-port=5000/tcp --permanent
firewall-cmd --reload
firewall-cmd --list-ports

sudo -u devops /usr/local/bin/ansible-galaxy collection install community.general ansible.eda
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-infra/user-data-ansible-engine-with-eda.sh

- **Rulebook의 구현 및 배포.**
  표준 "ansible.eda.webhook" 이벤트 소스는 EDA를 통해 애플리케이션 리포지토리에 대한 GitHub의 "푸시" 이벤트를 수신합니다. 이러한 이벤트가 수신되면 EDA는 위에서 언급한 Ansible Playbook을 사용하여 애플리케이션의 자동화된 롤링 업데이트를 시작해야 합니다.

```
---
- name: Listen for events on a webhook
  hosts: all

  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
      filters:
        - ansible.eda.dashes_to_underscores:

  rules:
    - name: Handle Webhook event
      condition: >
        event.payload is defined and event.meta.headers.X_GitHub_Event == "push"
        and event.payload.ref == vars.application_branch_in_webhook_event
        and event.payload.repository.url == vars.application_repo
#     condition: event.payload is defined

      action:
        run_playbook:
          name: rolling-update.yaml
          extra_vars:
            application_repo: "{{ application_repo }}"
            application_branch: "{{ application_branch }}"
            subfolder_path: "{{ subfolder_path }}"
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/webhook-for-rolling-update.yaml

- **데이터센터의 현재 시스템에 영향을 주지 않고 MVP 배포를 지원하는 환경을 프로비저닝할 곳을 찾습니다. 엔지니어링 팀은 즉시 시작할 수 있어야 합니다.**
  - **애플리케이션 인스턴스 및 HAProxy 로드 밸런서 인스턴스를 배포하기 위한 AWS 구성**
  - **데이터센터와 같은 3개의 EC2 Linux 인스턴스를 배포하기 위한 Terraform 코드 구현**

유용한 링크: "Terraform을 사용하여 AWS에서 무료 Ansible Lab 만들기(Use Terraform to Create a FREE Ansible Lab in AWS)" https://www.techbeatly.com/use-terraform-to-create-a-free-ansible-lab-in-aws/

```
locals {
  connection = {
    type        = "ssh"
    user        = "ec2-user"
    private_key = file(pathexpand(var.ssh_pair_private_key))
  }
}

resource "aws_instance" "ansible-engine" {
  ami           = var.aws_ami_id
  instance_type = "t2.micro"
  key_name      = aws_key_pair.ec2loginkey.key_name
  security_groups = ["sample-mvp-sg", "sample-mvp-webhooks-sg"]
  user_data       = file("user-data-ansible-engine-with-eda.sh")

  // Copy the "get-automation-sample-playbooks.sh" script to the remote machine
  provisioner "file" {
    source      = "files-for-upload/get-automation-sample-playbooks.sh"
    destination = "/home/ec2-user/get-automation-sample-playbooks.sh"
    connection {
      type        = local.connection["type"]
      user        = local.connection["user"]
      private_key = local.connection["private_key"]
      host        = self.public_ip
    }
  }
```

참고: https://github.com/mikhailknyazev/automation-samples/tree/main/sample-infra

- **상태 검사기 애플리케이션의 경우:**
  - **(JavaScript/Typescript에서) 상태 검사기 애플리케이션의 설계 및 구현**
  - **Red Hat OpenShift와의 통합을 위한 API 조사**
  - **표준 "개발자 샌드박스" Red Hat OpenShift 환경 구성**

  유용한 링크:
  - 2부: Red Hat OpenShift용 개발자 샌드박스에 풀스택 JavaScript 애플리케이션 배포(Part 2: Deploying full-stack JavaScript applications to the Developer Sandbox for Red Hat OpenShift) https://developers.redhat.com/developer-sandbox/activities/deploying-full-stack-javascript-applications-to-the-sandbox/part2
  - 기네시 마다파람바스의 "실제 자동화를 위한 Ansible"에서 "Ansible을 사용한 쿠버네티스 관리"에 관한 장(Chapter "Managing Kubernetes Using Ansible" in "Ansible for Real-Life Automation" by Gineesh Madapparambath) https://amzn.asia/d/flHSDjh
  - 루카 버튼의 "예시별 쿠버네티스용 Ansible: Ansible로 쿠버네티스 클러스터 자동화하기"("Ansible for Kubernetes by Example: Automate Your Kubernetes Cluster with Ansible" by Luca Berton) https://a.co/d/7tJWteG

```
- name: Display list of all Pods from the current Namespace (Project)
  kubernetes.core.k8s_info:
    kubeconfig: "{{ openshift.kubeconfig_file }}"
    kind: Pod
    namespace: "{{ openshift.namespace_name }}"
  register: pod_list
- name: Print pod_list
  debug:
    var: pod_list

- name: Deploy the Health-Checker with OpenShift feature "Source-to-Image" (S2I)
  ansible.builtin.command: >
    oc --kubeconfig={{ openshift.kubeconfig_file }} new-app
    https://github.com/mikhailknyazev/automation-samples --context-dir=sample-health-checker --name={{ health_checker_label }}
    -e HAPROXY_PUBLIC_IP={{ haproxy_public_ip }}
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-playbooks/health-checker.yaml

```
async function fetchData() : Promise<void>  {
  const haproxy_public_ip = document.getElementById('haproxy_public_ip').value;
  const response : Response  = await fetch(`http://${haproxy_public_ip}`, {
    method: 'GET',
    headers: {
      'Cache-Control': 'no-cache'
    }
  });
  const data : string  = await response.text();

  let lines : string[]  = data.split('\n');
  let resultIndex : number  = lines.findIndex(e : string  => e.includes("Serving from"));

  let str;
  if (resultIndex === -1) {
    str = "No matches found";
  } else {
    str = lines.slice(resultIndex).join('\n').replace(/<[^>]+>/g, ' ');
  }

  const listItem : HTMLLIElement  = document.createElement('li');
  listItem.textContent = str;
  document.getElementById('resultList').appendChild(listItem);
}

// Call fetchData every 5 seconds
setInterval(fetchData, 5000);
```

참고: https://github.com/mikhailknyazev/automation-samples/blob/main/sample-health-checker/public/script.js

4.2.4. 샘플: 이터레이션을 위한 쇼케이스(데모)

참고: 일부 스크린샷은 MIT 라이센스에 따라 게시된 개인 리포지토리에서 찾을 수 있습니다. https://github.com/mikhailknyazev/automation-samples/blob/main/README.md

첫 번째 제공 이터레이션이 끝날 때 쇼케이스(데모)에서 "샘플 프로젝트" 이해관계자들은 증분(이터레이션) 동안 수행된 최근 작업에 대한 데모를 보고 피드백을 제공했습니다. 일반적으로 이들은 MVP의 목표인 "애플리케이션 배포, 패치 적용, 관측성 및 성능 개선"과 관련된 결과에 깊은 인상을 받았습니다.

특히

- 이해관계자들은 시연된 자동화 기술 중 일부를 조직 전체에 재사용할 수 있으며 점차 수동 IT 작업을 50~80% 줄일 수 있을 것으로 예상했습니다.
- 이제 애플리케이션 배포가 완전히 자동화되었습니다. 따라서 인프라 팀은 작업 방식을 효율적인 "사이트 신뢰성 엔지니어" 모델로 전환하는 계획을 실제로 세울 수 있습니다.
- 이해관계자들은 이제 부하 분산과 관련된 아키텍처 변경 덕분에 컴퓨팅 용량을 이전보다 더 쉽게 필요에 따라 애플리케이션에 추가할 수 있다는 점을 높이 평가했습니다.
- 시연한 "GitOps with Ansible" 기술은 애플리케이션 개발 생산성을 높이고 미션 크리티컬한 Linux 워크로드의 변경 사항 추적성을 개선할 수 있습니다.
- "상태 검사기" 원격 구성 요소는 애플리케이션에 대한 관측성을 개선하여 궁극적으로 사용자에게 제공되는 서비스의 안정성을 개선합니다. 이해관계자들은 프로젝트 목표를 달성하기 위해 Red Hat OpenShift의 Source-to-Image(S2I) 기능을 사용하는 것이 얼마나 간단한지에 감탄했습니다.

MVP를 위해 개선할 점으로 이해관계자들은 패치 적용 및 보안 문제 해결 데모를 더 구체적으로 다뤘으면 좋았을 것이라 언급했습니다. 그러나 애플리케이션 롤링 업데이트 중 Linux 패키지 업데이트 데모 덕분에 이전보다 훨씬 쉽게 개선할 수 있게 되었다는 부분은 높이 평가했습니다.

5. Red Hat의 고객 지원 사항은 무엇인가요?

이 섹션에서는 이 가이드 작성자들 개개인의 개별 의견만을 기술하였습니다. 이 가이드에 제시된 정보는 어떠한 보증도 없이 '있는 그대로' 제공됩니다. 이 가이드를 읽는 시점에는 최신 내용이 아닐 수 있습니다. 작성자는 이 정보와 관련된 어떠한 책임도 지지 않습니다. 작성자는 다른 관련 Red Hat 리소스 사용을 지원하는 것이 도움이 될 수 있다고 *생각*합니다.

이 섹션에서는 이 가이드의 컨텍스트와 관련된 Red Hat의 상용 제품 및 서비스 제공 중 일부에 대해 설명합니다.

5.1. Red Hat Open Innovation Labs를 통한 자동화 또는 컨테이너 도입 여정에 대한 신뢰 향상

Red Hat Open Innovation Labs는 자동화 또는 컨테이너 도입 여정을 시작하는 조직을 지원할 수 있습니다. 고객의 직원들은 오픈소스 방식으로 탁월한 제품을 구축하는 방법을 배우게 됩니다. Red Hat의 몰입형 레지던스 환경은 엔지니어와 오픈소스 전문가를 연결하여 고객의 아이디어를 비즈니스 성과로 발전시킵니다.

4~12주 동안 진행되는 Open Innovation Labs 레지던스 환경에서는 팀의 아이디어를 오픈소스 커뮤니티가 제공하는 최고의 기술과 결합하는 방법을 배웁니다. Red Hat 전문가는 Red Hat 기술, 오픈소스 커뮤니티, 고객 팀의 잠재력을 끌어내는 데 필요한 주요 혁신 사례를 심층적으로 파악하도록 지원합니다.[47]

세계보건기구(WHO)는 DevOps 플랫폼을 구축하기 위해 8주간의 Red Hat Open Innovation Lab 레지던스 환경을 가상으로 진행했습니다. 다음은 이를 요약한 짧은 동영상입니다.[48] 또한 전문가들이 어떤 방식으로 참여를 유도하는지 알아보려면 흥미로운 프레젠테이션인 도나 벤자민(Donna Benjamin)의 "스토리를 소프트웨어"로 전환하기(“Turning stories into software”)[49]를 확인해 보시기 바랍니다.

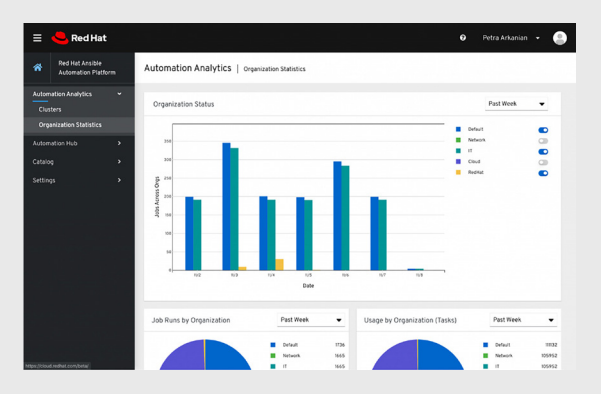
5.2. 구성 요소를 대규모로 이동하나요? Red Hat Ansible Automation Platform을 사용해보세요

Red Hat Ansible Automation Platform[50]은 멀티 티어 아키텍처, 이기종 IT 환경 및 사용자 권한을 관리하는 데 도움이 되는 제품입니다. 조직 수준의 IT 관리를 위해 설계되었으며, 관리되는 모든 호스트 정보를 간략하게 보여주는 편리한 UI 대시보드를 제공하고, 모든 종류의 자동화 기능을 쉽게 탐색할 수 있습니다. 조직의 유연한 거버넌스와 보안 요구 사항을 충족합니다. AAP는 인증 및 권한 부여를 위해 Active Directory를 사용하는 등 기존 표준 접근 방식에 높은 수준으로 "임베딩이 가능"합니다. 또한 IT 팀에 Ansible Playbook 실행을 위한 동급 최고의 관측성을 제공합니다.

예를 들어 Ansible Automation Platform이 제공하는 장점은 다음과 같습니다.

- 역할 기반 액세스 제어 및 변경 요청 관리를 통해 자동화 보호
- 유연한 단일 플랫폼에 통합하여 사일로화된 팀과 프로세스 제거
- 온프레미스, 클라우드, 네트워크 엣지 등 어디서나 자동화 확장

[47] Red Hat Open Innovation Labs https://www.redhat.com/ko/services/consulting/open-innovation-labs
[48] WHO의 새로운 플랫폼: 혁신적인 학습 플랫폼 구축(WHO built this: Creating an innovative learning platform together) https://www.youtube.com/watch?v=f69l6Vo9rGk
[49] 스토리를 소프트웨어로 전환하기(Turning stories into software) https://www.youtube.com/watch?v=J3VaaPYshek
[50] Red Hat Ansible Automation Platform https://www.redhat.com/ko/technologies/management/ansible



AAP의 워크플로우를 사용하면 서로 다른 인벤토리 사용에 관계 없이 원하는 수의 플레이북을 연결하고, 다양한 자격 증명을 활용하고, 여러 사용자가 실행할 수 있습니다. 참고: 23장. 워크플로우.[51]

AAP는 안전한 REST API와 해당 CLI 툴을 노출합니다. 때문에 ServiceNow와 같은 ITSM 툴 및 Jenkins와 같은 CI 시스템과의 통합이 쉽습니다. 예를 들어, API 또는 CLI를 통해 자동화 작업(관리형 플레이북)을 시작할 수 있습니다.

AAP는 머신 및 클라우드 시스템에 대한 모든 자격 증명을 안전하게 저장할 스토리지와 강력한 역할 기반 액세스 제어 엔진을 추가하여, 누가 어떤 환경에서 어떤 자동화를 실행할 수 있는지에 대한 정책을 쉽게 설정할 수 있으므로 올바른 사용자만 머신에 액세스하고 구성을 적용할 수 있게 합니다. AAP 자체는 분산형 시스템으로, 고수준의 엔지니어링 기법을 통해 고가용성 플랫폼으로 만들었습니다.

AAP는 활발히 개발 중이며, "Ansible Automation Platform 2.4의 새로운 기능"[52]에 따르면 AAP에 새로 추가되거나 수정된 기능은 다음과 같습니다.

EDA, 컬렉션 리포지토리 관리, 검증된 콘텐츠 통합, Ansible Builder 3.0, ARM용 플랫폼 설치 지원, IBM Watson Code Assistant가 통합된 Ansible Lightspeed 기술 프리뷰가 제공됩니다.

6. 결론

Ansible을 통한 효율적인 자동화에 관심을 가져 주셔서 감사합니다!

고객 사례를 기반으로 한 이 가이드의 초반부에서는 Ansible이 DevOps 툴과 같이 사용하기 좋으며 IT 아티팩트를 위한 범용 "접착제" 역할을 할 수 있는 이유를 설명했습니다. 또한 조직 차원의 자동화 아이디어와 자동화가 일상적인 반복 태스크에 어떤 도움이 되는지 소개했습니다.

가이드의 주요 부분에서는 "검색 루프", "옵션 피벗", "제공 루프"의 사례를 통해 자동화 프로젝트의 성공에 어떻게 도움이 되는지 설명하고 데모를 진행했습니다. 마지막으로 Red Hat에서 제공하는 관련 오퍼링 몇 가지를 설명했습니다.

도움이 되었기를 바랍니다!

[51] "23장. 워크플로우" https://docs.ansible.com/automation-controller/latest/html/userguide/workflows.html
[20] Ansible Automation Platform 2.4의 새로운 기능 https://www.ansible.com/blog/whats-new-in-red-hat-ansible-automation-platform-2.4

7. 작성자 소개

마이클 크냐제프
Red Hat

마이클 크냐제프(Michael Knyazev)는 OpenShift, Ansible, AWS, GCP, Kafka를 전문으로 하는 숙련된 핸즈온 아키텍트/DevOps 컨설턴트입니다.

마이클은 아시아 태평양 지역의 주요 Red Hat 고객에게 가치 있는 성과를 제공하는 데 핵심적인 역할을 해왔습니다. Red Hat에 입사하기 전에는 세계적인 수준의 여러 솔루션을 공동 설계하고 구축했습니다. 대표적으로 TPG Telecom과 Insurance Australia Group을 위한 쿠버네티스 기반 플랫폼, 호주 국영 코로나바이러스 헬프라인 클라우드 솔루션, Morningstar Tick Data 및 Beam Wallet FinTech 플랫폼 등이 있습니다.

마이클은 커뮤니티에 기여하기를 즐깁니다! 그는 Red Hat의 "Ansible을 통한 효율적인 자동화" e-book에 주요 기여자로 참여했으며, 인기 있는 Udemy 강좌인 "신뢰성을 위한 쿠버네티스 구성(Configuring Kubernetes for Reliability)"을 개설하고, 2006년에 IT 아키텍처의 자동화된 개발에 관한 박사 학위 논문으로 과학적 성취를 이루었습니다.

배움에 대한 열정이 넘치는 마이클은 15개 이상의 전문 자격증뿐 아니라 MBA도 보유하고 있습니다. 최근 몇 년은 글로벌 카오스 엔지니어링 컨퍼런스에서, 이전에는 JavaOne에 발표자로 참여했습니다.

현재 호주 시드니에서 가족과 함께 살고 있습니다.

기네시 마다파람바스
Red Hat

기네시 마다파람바스(Gineesh Madapparambath)는 Red Hat 싱가포르 지사의 플랫폼 및 DevOps 부문 컨설턴트입니다. 시스템 엔지니어부터 자동화 전문가 및 콘텐츠 작성자에 이르는 다양한 경력을 쌓았으며, 주로 Ansible 자동화, 컨테이너화(OpenShift 및 쿠버네티스), 코드형 인프라(Terraform)를 주로 담당하고 있습니다. 기네시는 “실제 자동화를 위한 Ansible(Ansible for Real-Life Automation)”이라는 책을 출간하기도 했습니다. 자동화 솔루션을 설계, 개발 및 배포한 풍부한 경험을 보유하고 있으며, Ansible과 Ansible Automation Platform(이전의 Ansible Tower)을 사용하고 있습니다. 이러한 솔루션은 베어 메탈/가상 서버 구축, 패치 적용, 라이센스 관리, 네트워크 운영 및 사용자 정의 모니터링과 같은 다양한 태스크를 포괄합니다. 또한 전 세계 데이터센터의 서버 설계 및 배포를 오케스트레이션하며 다양한 문화권에서 클래식, 프라이빗 클라우드(OpenStack, VMWare), 가상 및 퍼블릭 클라우드 환경(AWS, Azure, Google Cloud)에 대한 귀중한 경험을 쌓았습니다. 다양한 경력을 쌓으며 그는 시스템 엔지니어, 자동화 전문가, 인프라 디자이너, 콘텐츠 작성자 등 여러 직책을 맡아왔습니다.

니콜라스 웡
Red Hat

니콜라스 웡(Nicholas Wong)은 "더 나은 사회" 만드는 것을 인생의 사명으로 삼고 선한 영향력을 발휘하는 사람입니다. 개인, 팀, 기업과 협력하며 가이드, 코치, 멘토로서 자신의 역할에 진지하게 임합니다. 그는 다른 사람들과 의미 있게 소통하기 위한 핵심 능력을 키우며, 어디서나 아름다운 관계를 만들어가기 위해 노력합니다.

수석 제공 컨설턴트, 가치 흐름 전문가, 마스터 코치, 여러 분야의 공인 강사인 그는 품질 평가 컨설턴트, 시스템 관리자, 기술 전달 책임자로서 현장에서 쌓은 경험을 바탕으로 다양한 분야에서 활동하고 있습니다.

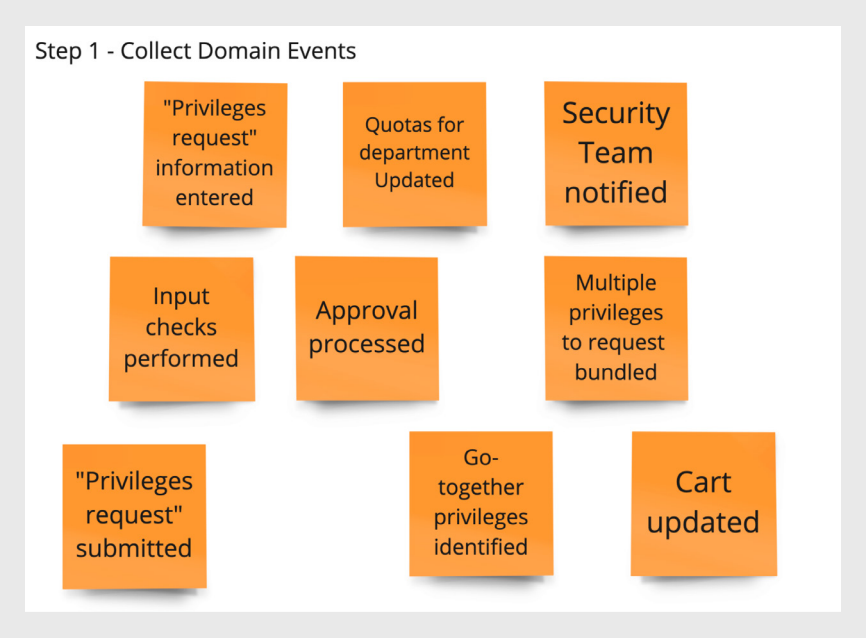
니콜라스는 다른 사람들이 잠재력을 최대로 실현하고, 새로운 도전을 받아들이고, 목표를 달성하도록 돕는 데 많은 시간을 보냅니다. 그는 학습자에게 무엇이 필요한지 잘 파악하여 그에 따른 맞춤형 서비스를 제공합니다. 업무 외에도 그는 경영진과 고성과자를 대상으로 일대일 코칭을 하기도 합니다.

니콜라스는 DevOps, 애자일 코칭, 기업/개인 코칭 커뮤니티에서도 활발히 활동하고 있습니다. 그는 지식과 전문성을 누구나 활용할 수 있어야 한다고 굳게 믿고 있으며, 모든 계층의 리더들과 협력을 통해, 의욕적으로 적극 참여하는 자율적인 팀을 미래 지향적인 방향으로 이끄는 것을 즐깁니다.

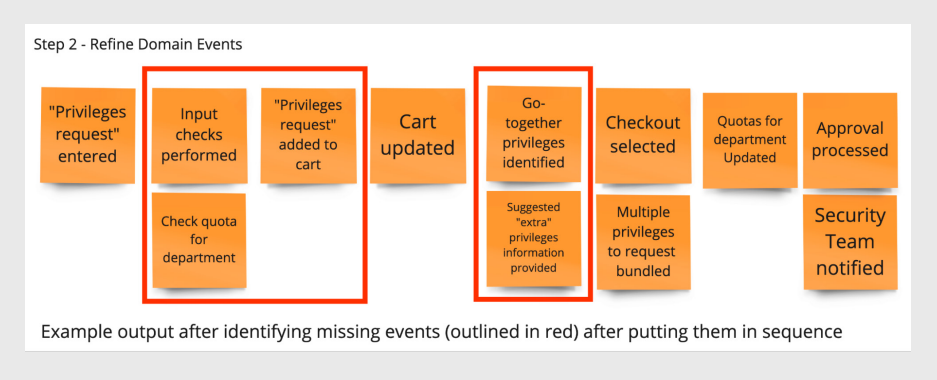
"구성원과 함께 성과를 내며 삶을 개선합니다" ~ 니콜라스 웡

부록 A. "사용 중인 애플리케이션 - 액세스 권한 요청"에 대한 이벤트 스토밍

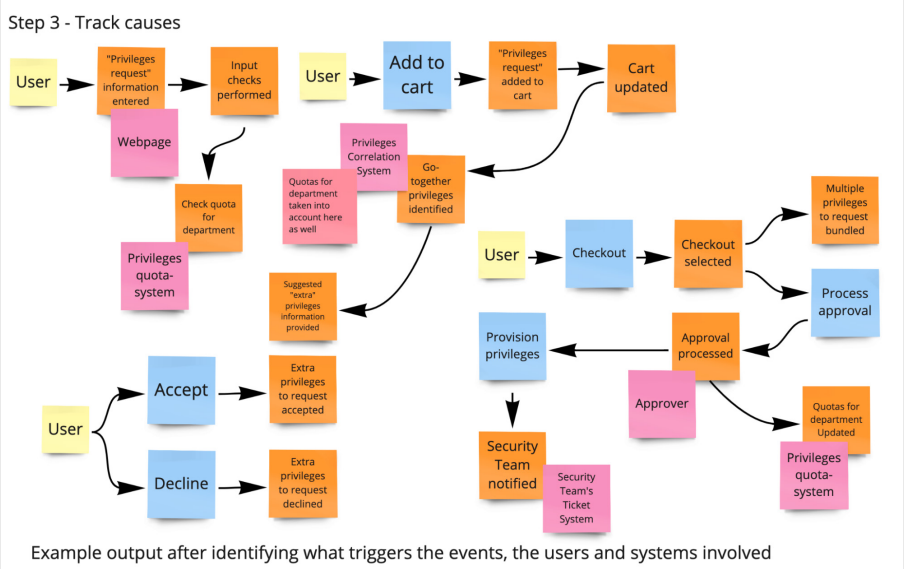
"1단계 - 도메인 이벤트 수집(큰 그림)" 단계에서는 각 참가자가 주황색 포스트잇만 사용합니다.



"2단계 - 도메인 이벤트 구체화(큰 그림)"에서는 참가자와 함께 도메인 이벤트 포스트잇을 살펴봅니다. 참가자에게 각 이벤트의 의미를 설명해 달라고 요청합니다. 구문의 정확성을 확인합니다. 또한 이벤트가 올바른 시간 순으로 나열되었는지 다시 한번 논의합니다. 발생하는 동의어(같은 항목에 대한 다른 용어)를 통합하고, 같은 용어가 다른 것을 설명하는 데 사용되었다면 그 차이점을 명확히 합니다.

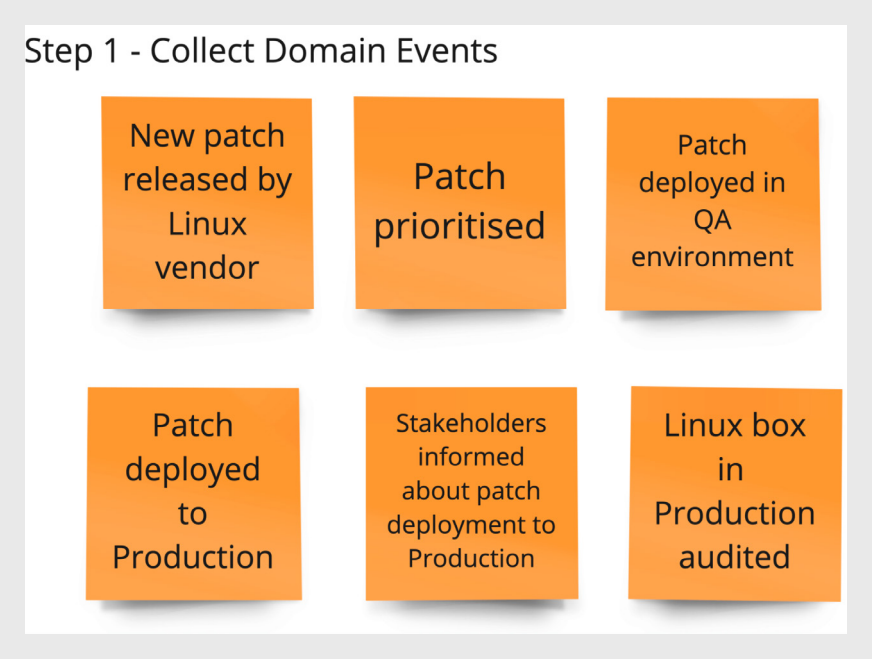


"3단계 - 원인 추적(프로세스 모델링)"에서는 원인 분석에 들어갑니다. 도메인 이벤트는 어디에서 발생할까요?

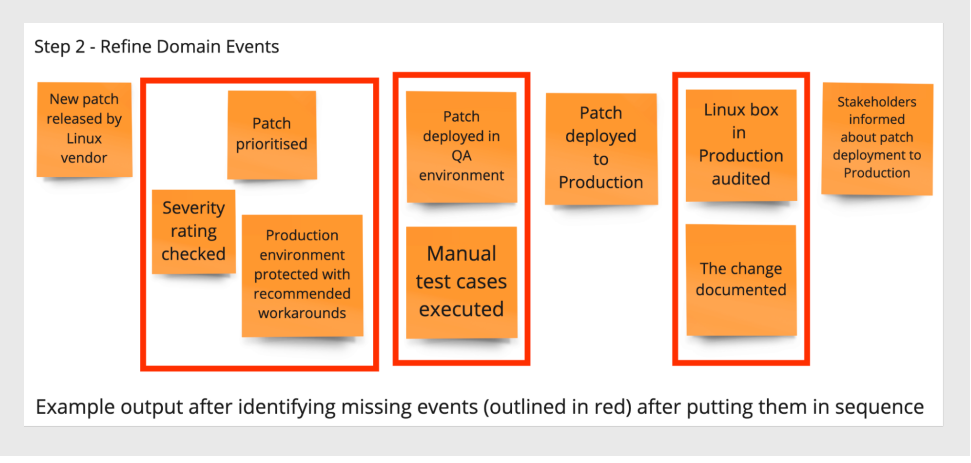

Step 3 - Track causes
User
"Privileges request" information entered
Input checks performed
Webpage
Check quota for department
Privileges quota-system
User
Add to cart
"Privileges request" added to cart
Cart updated
Privileges Correlation System
Quotas for department taken into account here as well
Go-together privileges identified
Suggested "extra" privileges information provided
User
Checkout
Checkout selected
Multiple privileges to request bundled
Process approval
Approval processed
Approver
Provision privileges
Security Team notified
Security Team's Ticket System
Quotas for department Updated
Privileges quota-system
User
Accept
Extra privileges to request accepted
Decline
Extra privileges to request declined
Example output after identifying what triggers the events, the users and systems involved

부록 B. "애플리케이션의 Linux 장비 패치 적용"에 대한 이벤트 스토밍

"1단계 - 도메인 이벤트 수집(큰 그림)" 단계에서는 각 참가자가 주황색 포스트잇만 사용합니다.

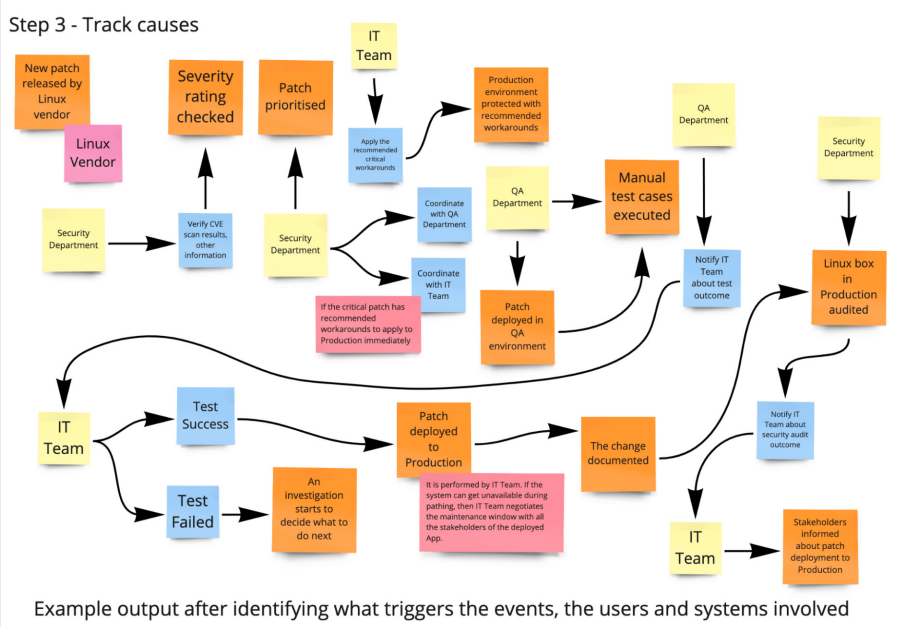


"2단계 - 도메인 이벤트 구체화(큰 그림)"에서는 참가자와 함께 도메인 이벤트 포스트잇을 살펴봅니다. 참가자에게 각 이벤트의 의미를 설명해 달라고 요청합니다. 구문의 정확성을 확인합니다. 또한 이벤트가 올바른 시간 순으로 나열되었는지 다시 한번 논의합니다. 발생하는 동의어(같은 항목에 대한 다른 용어)를 통합하고, 같은 용어가 다른 것을 설명하는 데 사용되었다면 그 차이점을 명확히 합니다.



Example output after identifying missing events (outlined in red) after putting them in sequence

"3단계 - 원인 추적(프로세스 모델링)"에서는 원인 분석에 들어갑니다. 도메인 이벤트는 어디에서 발생할까요?



Example output after identifying what triggers the events, the users and systems involved