DevOps 오픈소스 트랜드



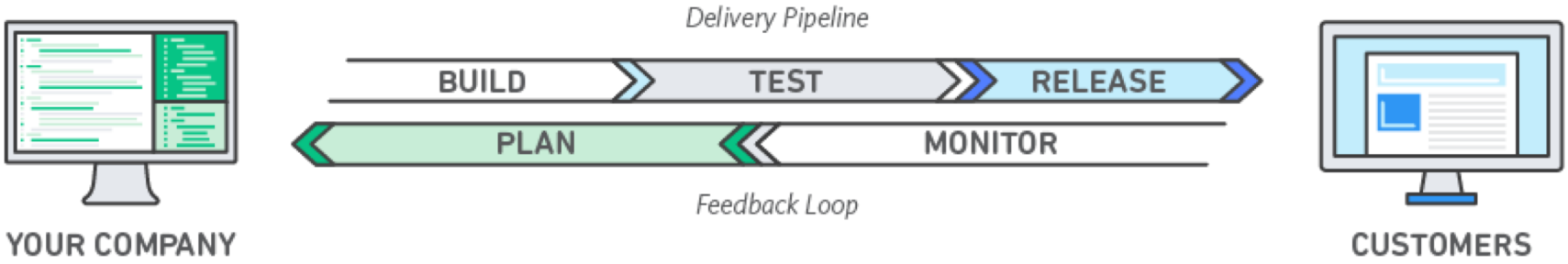
손 영수

ONYCOM

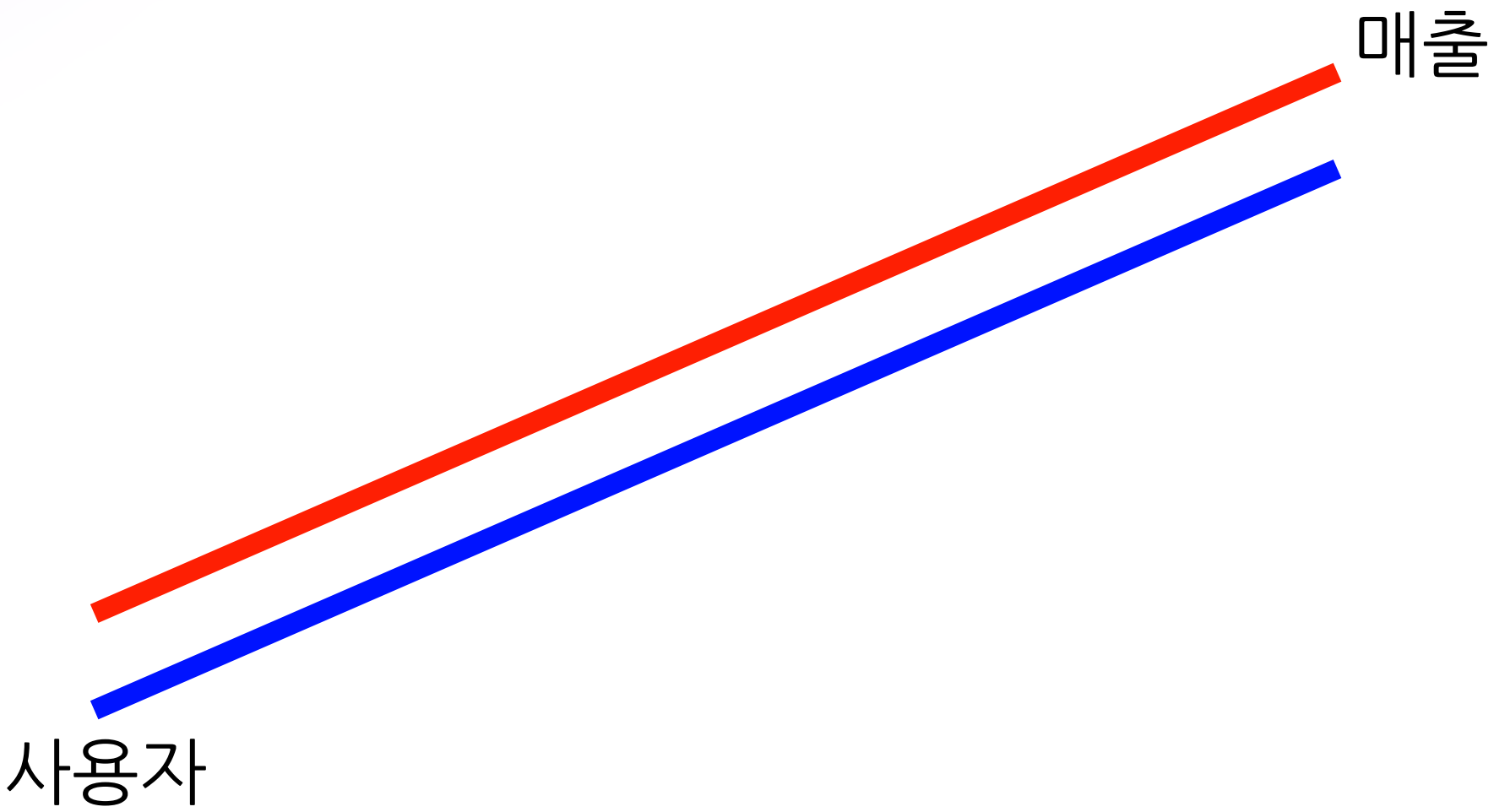
DevOps란? (AWS 사이트 참조)

애플리케이션과 서비스를 빠른 속도로 제공할 수 있도록 조직의 역량을 향상시키는 문화 철학, 방식 및 도구의 조합입니다.

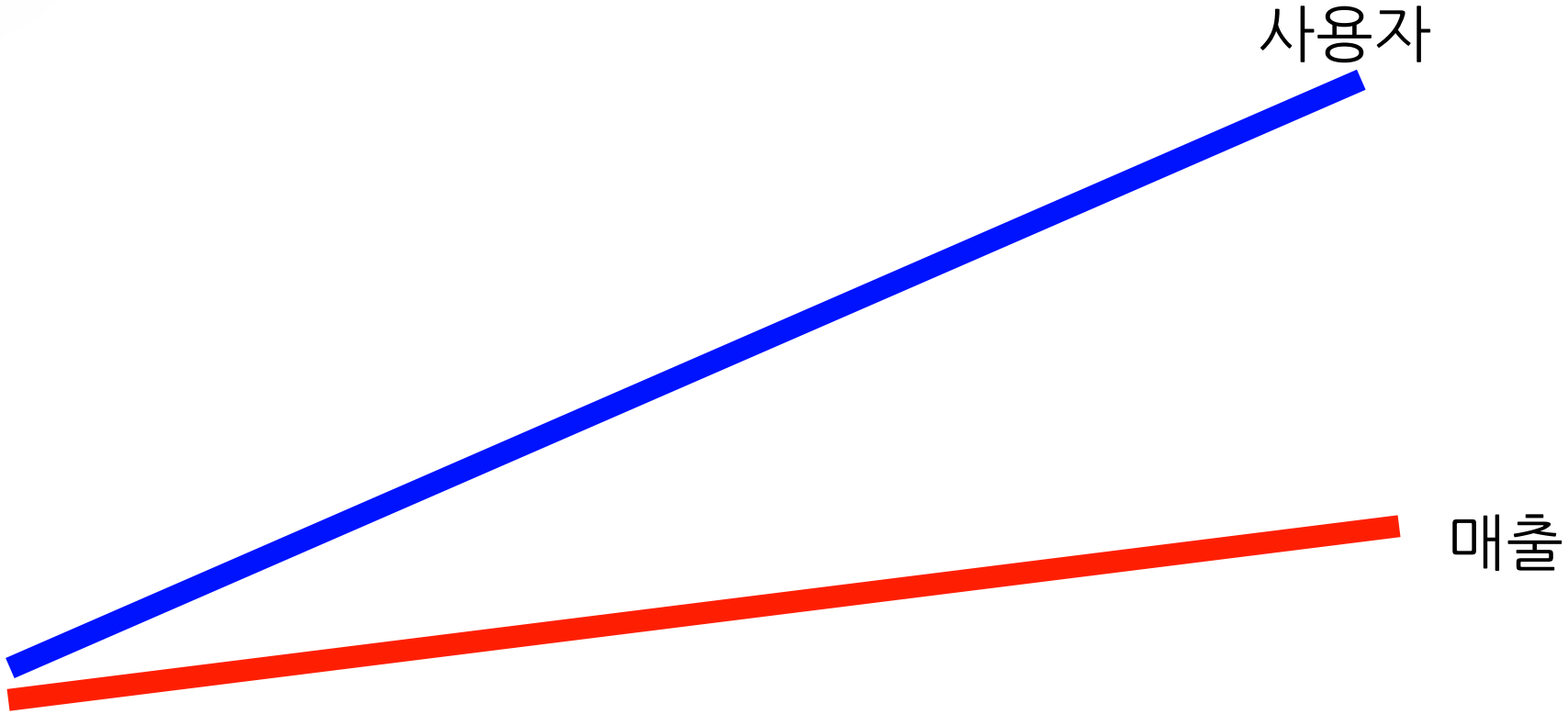
이러한 빠른 속도를 통해 조직은 고객을 더 잘 지원하고 시장에서 좀 더 효과적으로 경쟁할 수 있습니다.



DevOps가 나온 시대적 배경 (모바일 시대 이전)



DevOps가 나온 시대적 배경
(모바일 이후 - 사용자가 많다고 매출이 좋은 시대는 안녕..)




Part 1.
Cloud & DevOps

사용자가 많아도 수익이 별로라면..

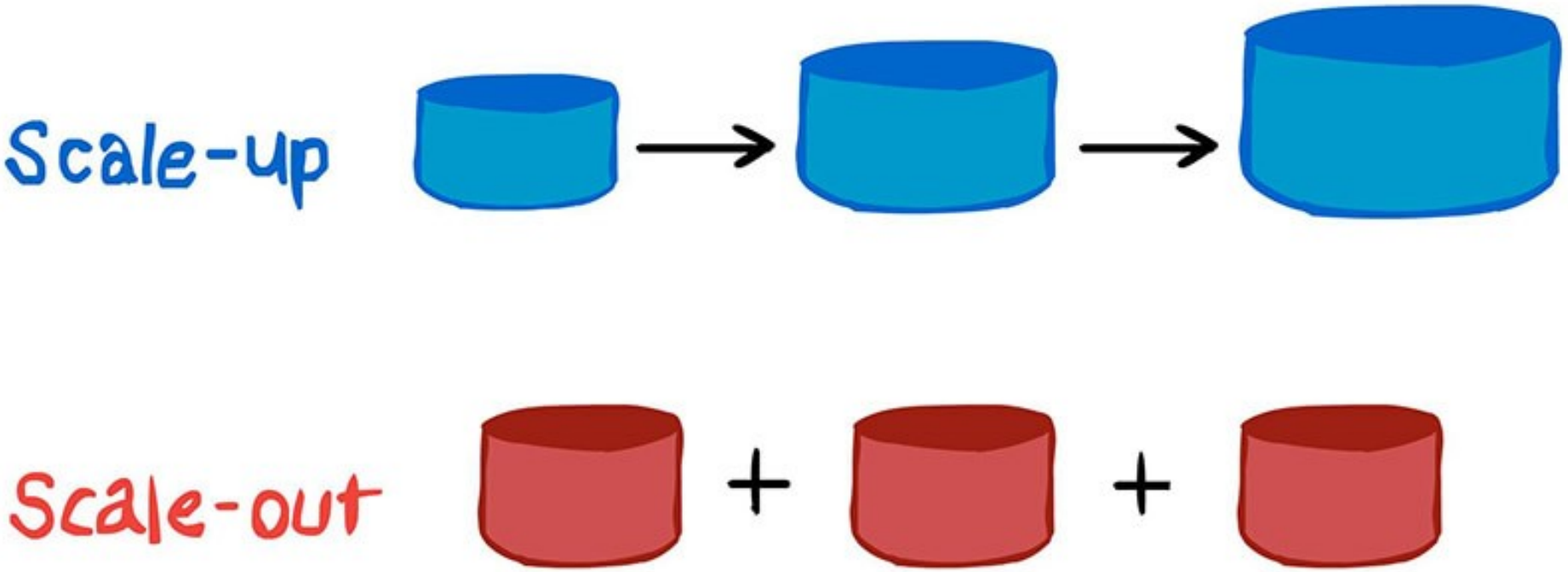
저렇게 많은 사용자를 어떻게 견디지??

값 비싼 하드웨어 장비, 오라클.. 을 살수 없는데..

고객이 한국에만 있나? 중국, 미국, 전 세계에 있다면..

대신 저렴하며 비슷한 효과를 낼 만한 것은?

클라우드 서버를 늘려가면서. (Scale Out) + 오픈소스 솔루션으로..

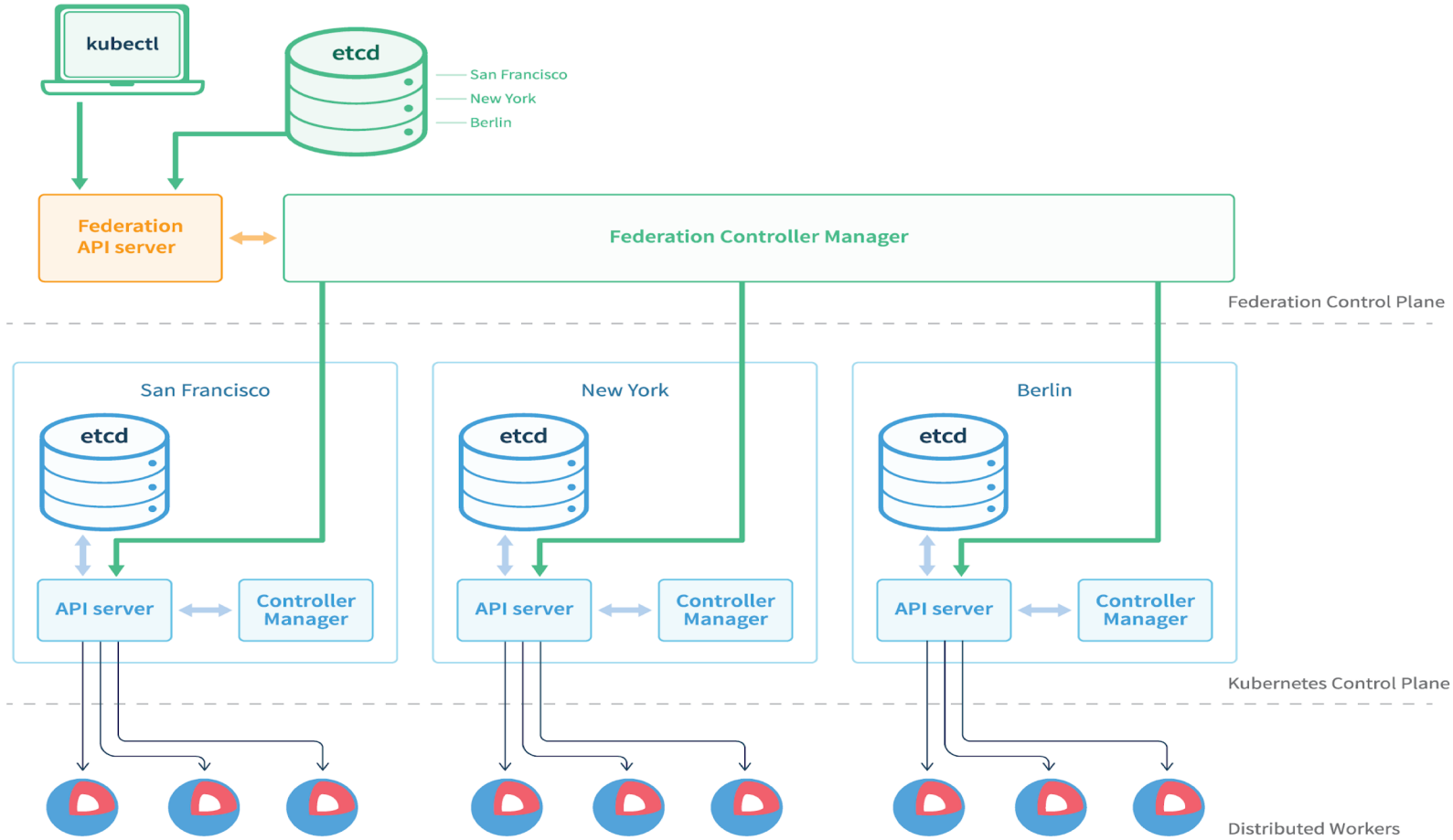


Scale Up **보다는** Scale Out **으로..**
비용이 더싸요.

왜 DevOps가 필요해 졌나. (클라우드..)

상대적으로 낮은 사양의 수 많은 서버들을 관리

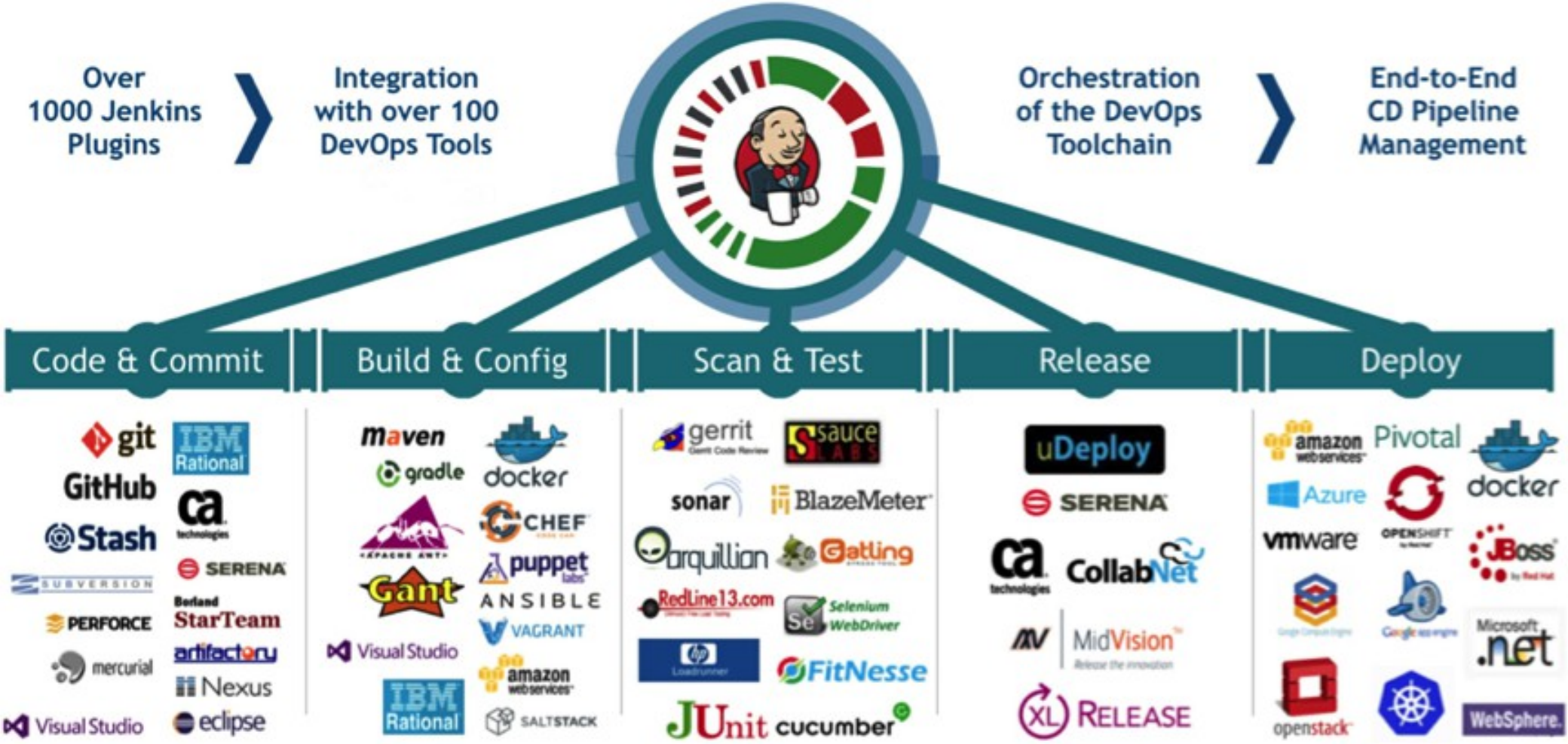
다양한 리전에 배포 하는 기술이 필요



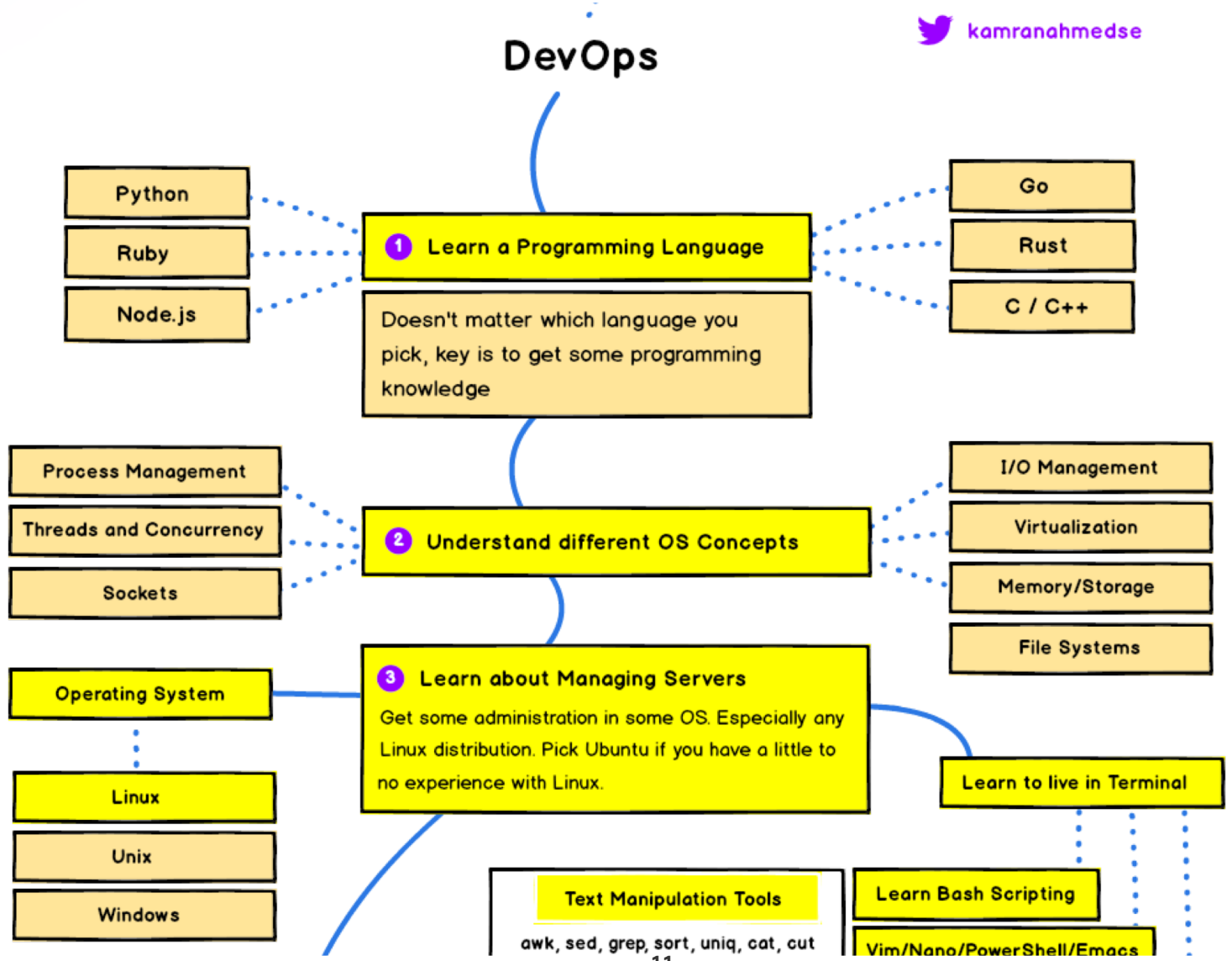
DevOps가 가져오는 의미 – 업무 간에 경계가 모호해 진다.



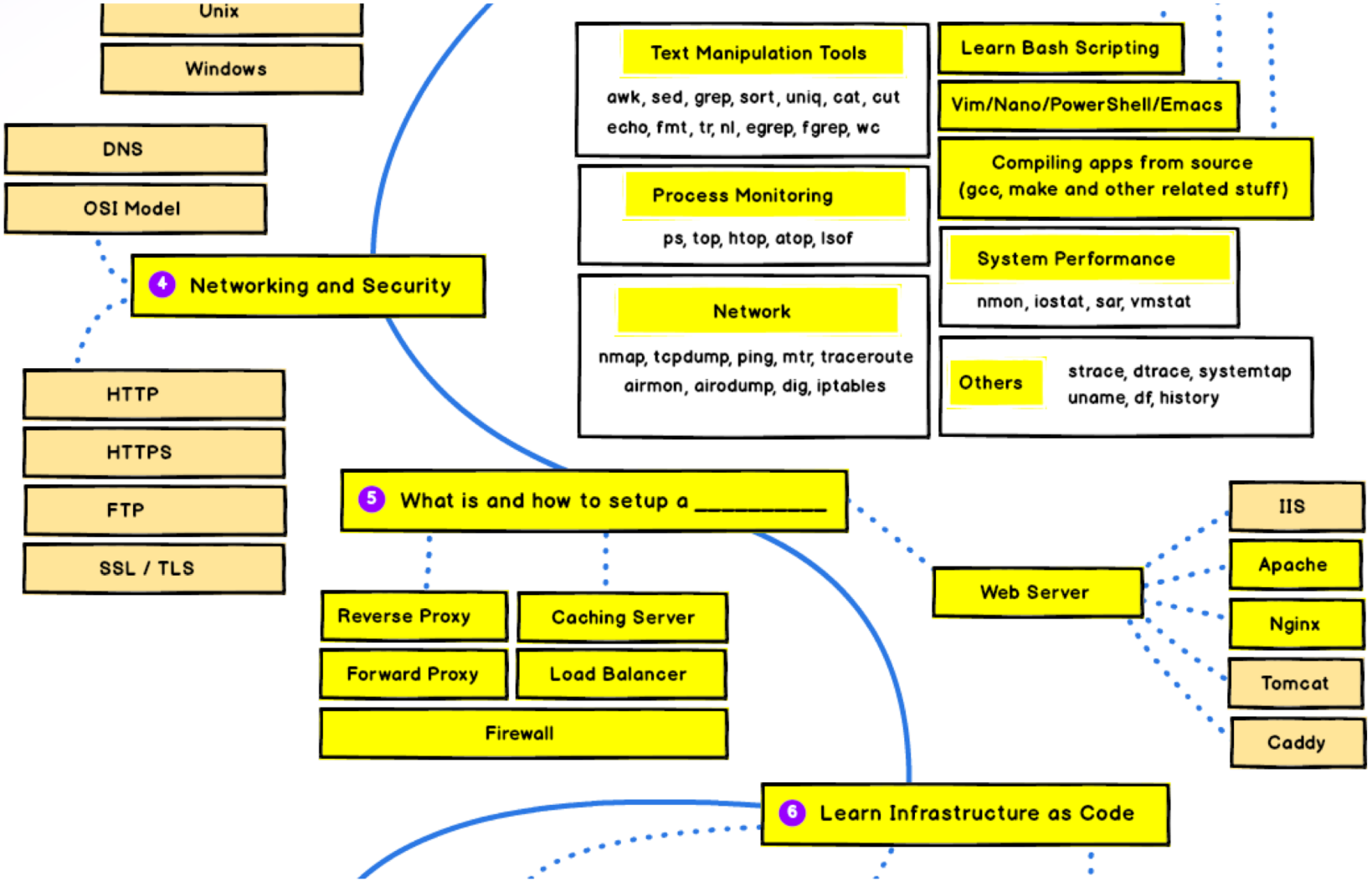
그래서 복잡한 DevOps 단계별 다양한 툴들..



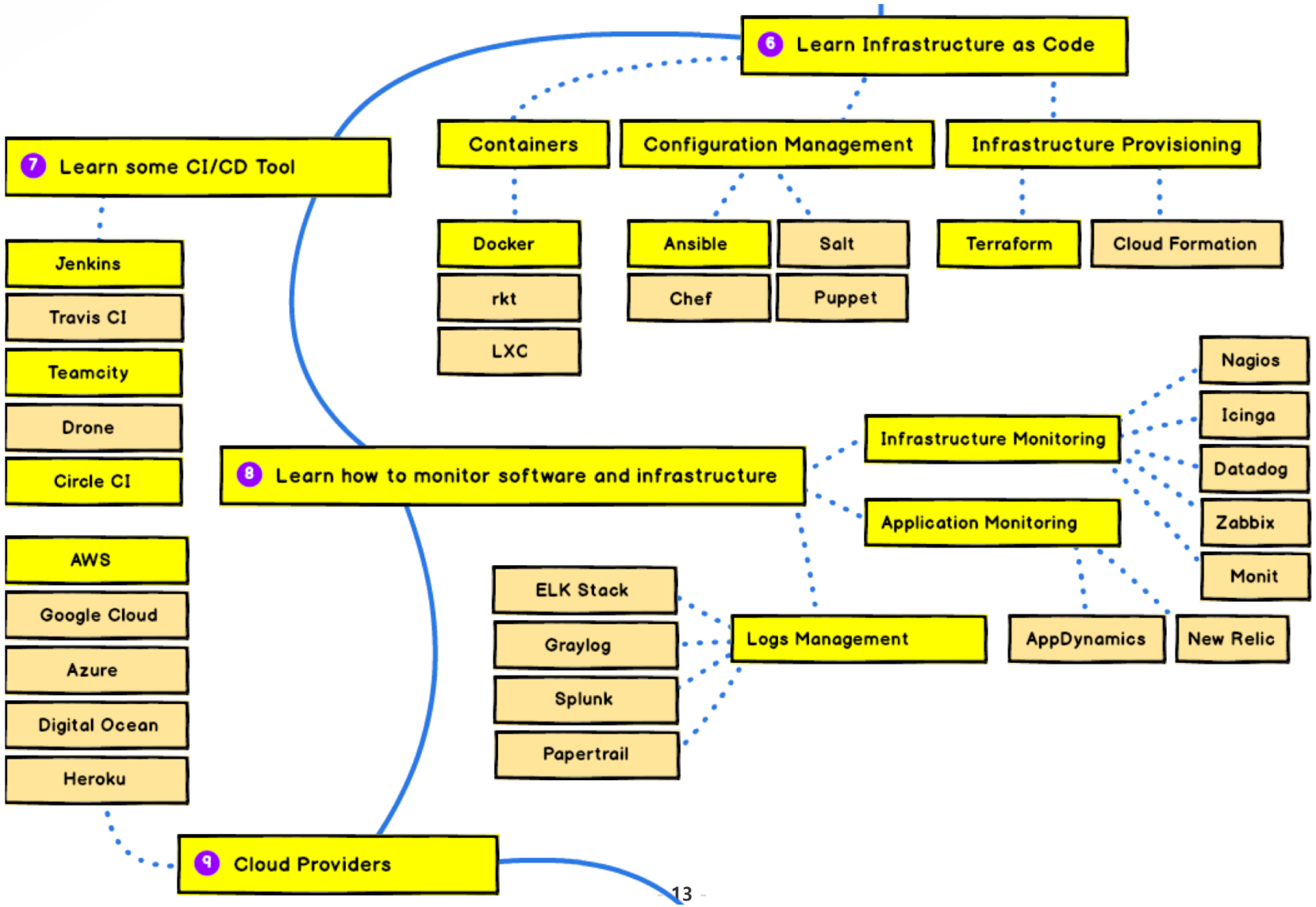
DevOps 엔진니어의 RoadMap
(https://github.com/kamranahmedse/developer-roadmap)



DevOps 엔진니어의 RoadMap
(https://github.com/kamranahmedse/developer-roadmap)



DevOps엔지니어 의 RoadMap
(https://github.com/kamranahmedse/developer-roadmap)



DevOps의 모범사례

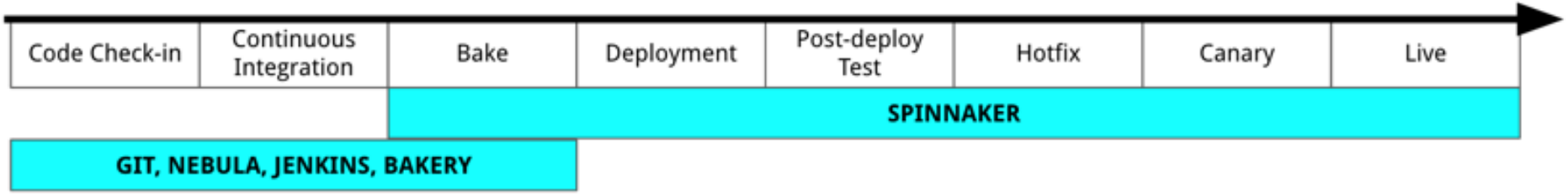
NETFLIX

Code Commit 부터 리그레이션 / 성능/ 부하 / 보안 테스팅을 거쳐

배포하는데 까지 걸리는 시간이 단지

16분

DevOps의 모범사례 – NETFLIX 사례 (2017년 기준)



- NEBULA - NETFLIX 빌드를 빠르고, 쉽게 테스트하기 위한 Gradle 플러그인들의 집합체들
- BAKE – Aminator 를 사용함 (커스텀 AMI 를 생성하기위한 툴을 사용)

  불변 인프라 (Immutable Infrastructure / server) 를 구성함 (AMI : Amazon Machine Image) 즉 이미지 형태로 구성되어 배포
- Deployment – Spinnaker 를 사용하여 배포 / 멀티 클라우드를 지원하는 Continuous Delivery Platform으로 오픈소스 하여 제공

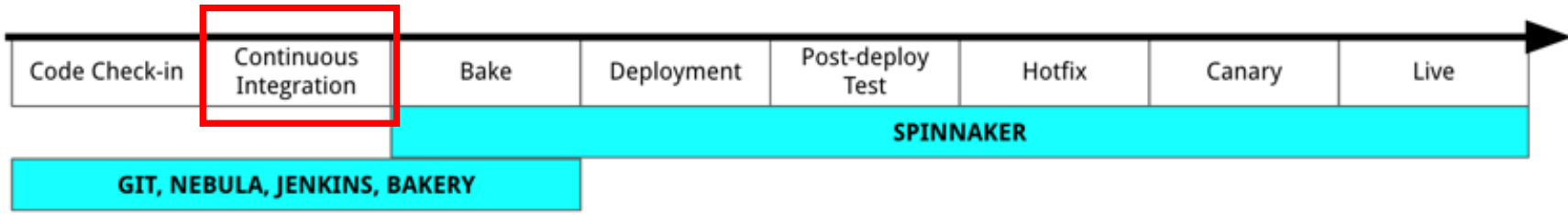
  Amazon, Azure, GCE, Kubernetes , OpenStack 과 같은 오픈소스 기반의 클라우드 또는 컨테이너 플랫폼을 동시에 지원하여 배포 가능함.

  또한 파이프라인을 통해 Smoke Test후 성공 하면 배포, 낮에는 업무를 봐야하니 Nightly Build등이 가능한 상황
- Canary - 전체 다 배포하지 않고, 일부 AMI만 배포를 해서 기존 것과 어떠한 성능 / 안정성 비교 평가 후 점진적 배포 , 아니면 롤백
- Live – 라이브 배포

DevOps의 모범사례 – CI 사례

NETFLIX

nebula



OPEN SOURCE

Jenkins

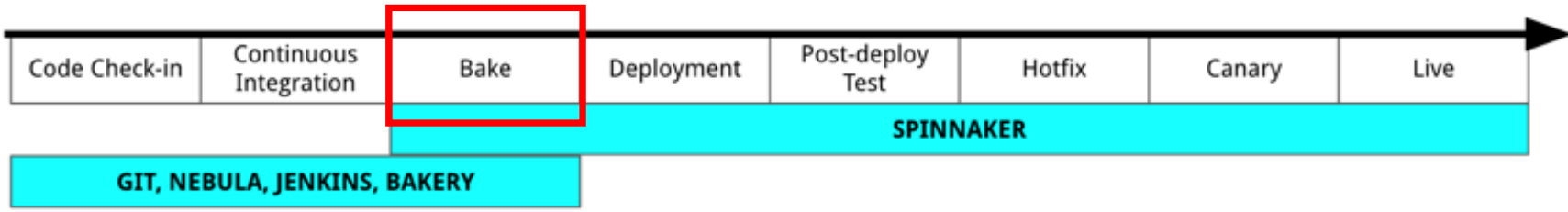
gradle

DevOps의 모범사례 – Bake 사례

Aminator (Amazon AMI Creator)

NETFLIX





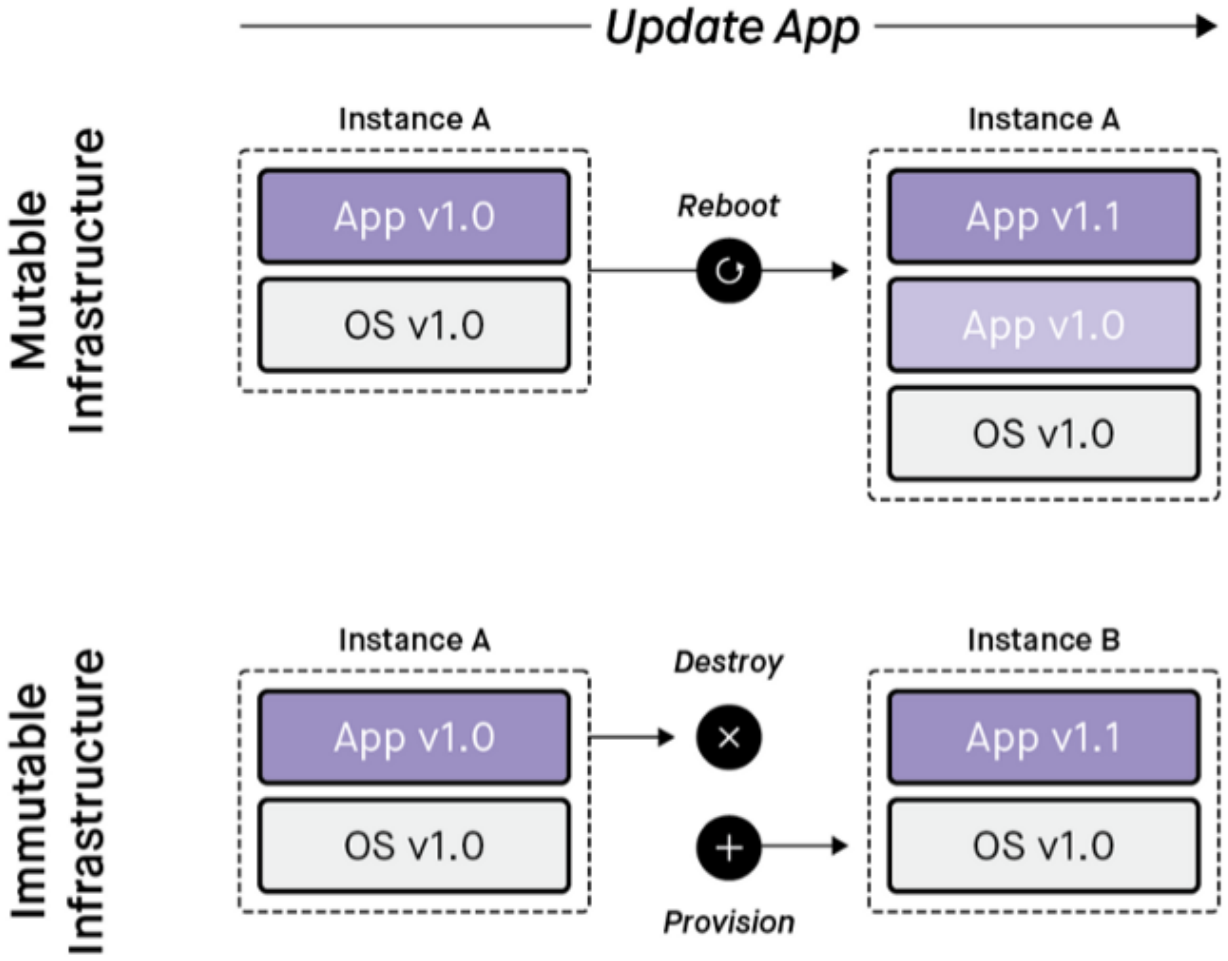
OPEN SOURCE



HashiCorp Packer

Bake 단계 – 불변 인프라 (Immutable Server – PhoenixServer)

Code 가 아닌 Image (VM, Docker) 를 배포함으로써, 버전 관리 Diff를 체크해 버전 별 패치하는 것으로 부터 자유로워짐.





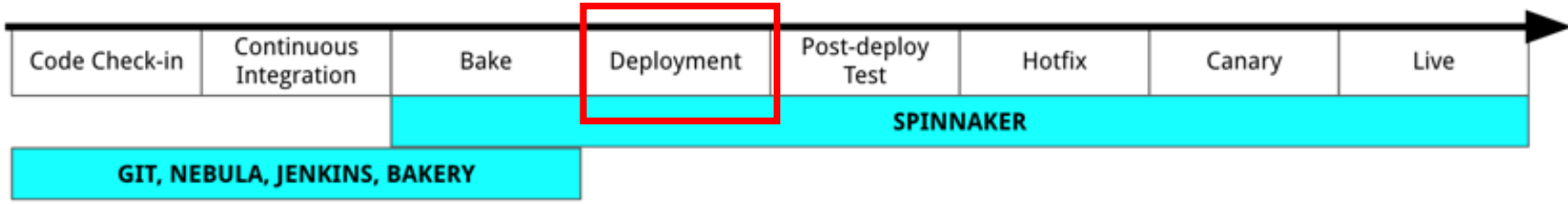
불변 인프라 (이미지 배포)의 장점 과 케이스 스터디

- 낮은 배포 실패 확률
  - 프로세스 다운 타임 없는 레드-블랙 배포와 릴리즈
  - 원자성 (배포 되거나 롤백되거나.)
- 일관된 STG 환경과 쉬운 Scale Out
  - 모든 서버가 동일한 생성 프로세스 사용 (특이 배포케이스가 없다)
  - Seamless 하게 서버를 추가하여 Scale Out이 매우 용이
- 간단한 롤백과 복구 프로세스
  - 이미지 이력을 보관하여 사용
- 적용 사례 (Netflix 외..)
  - CodeShip - https://blog.codeship.com/immutable-infrastructure/
  - Chef - https://blog.chef.io/2014/06/23/immutable-infrastructure-practical-or-not/
  - Koddi - https://www.koddi.com/developing-with-an-immutable-infrastructure/
  - Fugue - https://www.fugue.co/assets/docs/Immutable_Infrastructure_Fugue.pdf
- 참조 - https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure

DevOps의 모범사례 – Deployment 사례

NETFLIX

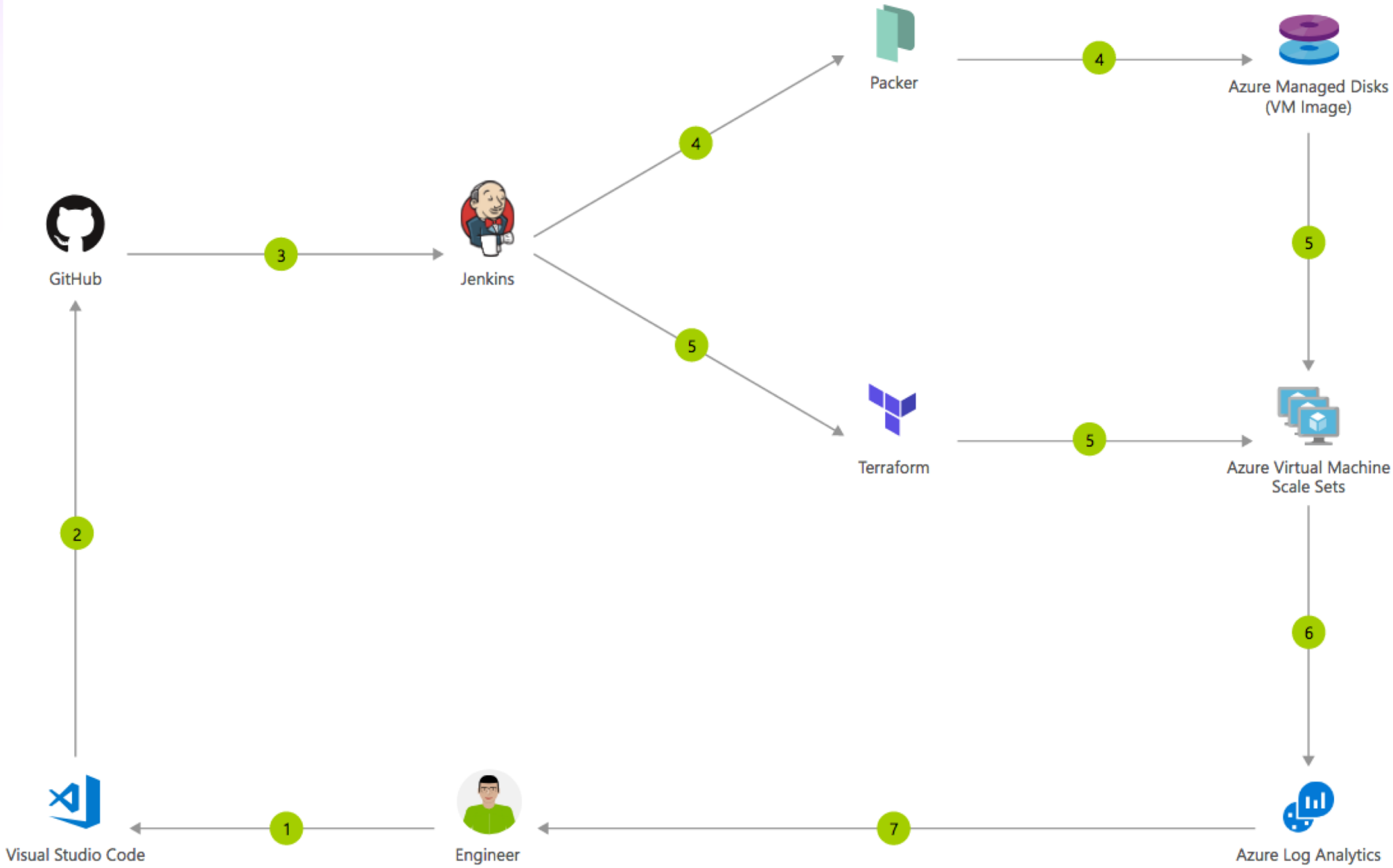
Spinnaker



OPEN SOURCE

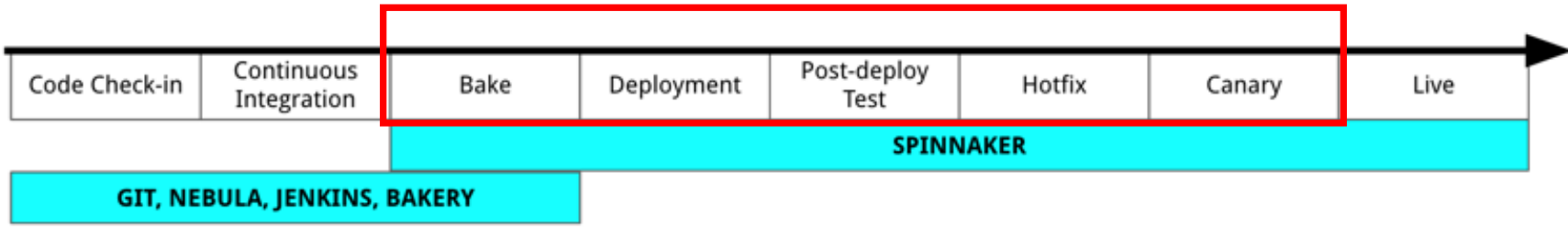
HashiCorp Terraform

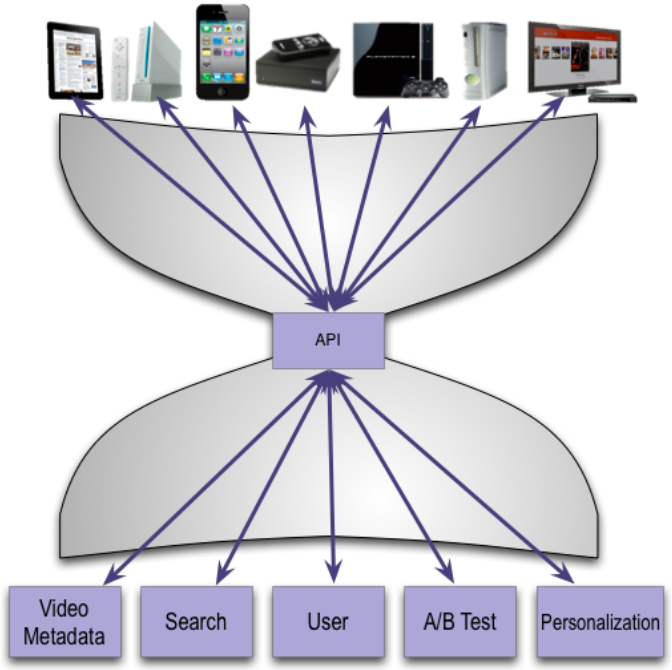
DevOps의 모범사례 – Deployment 사례 (ONYCOM - imqa 사례)



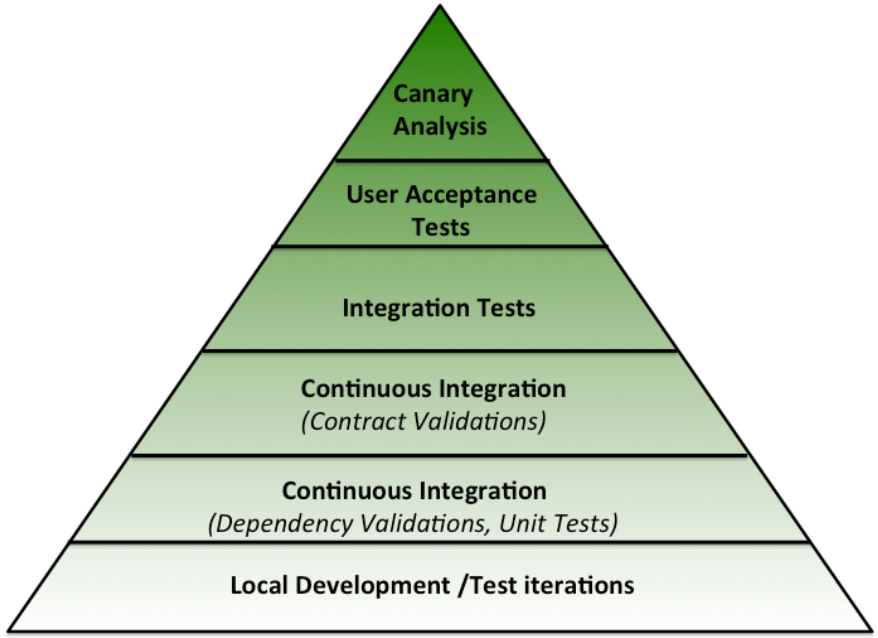
DevOps의 모범사례 – QA 확보 방안 (API Lifecycle Management + Test)

NETFLIX





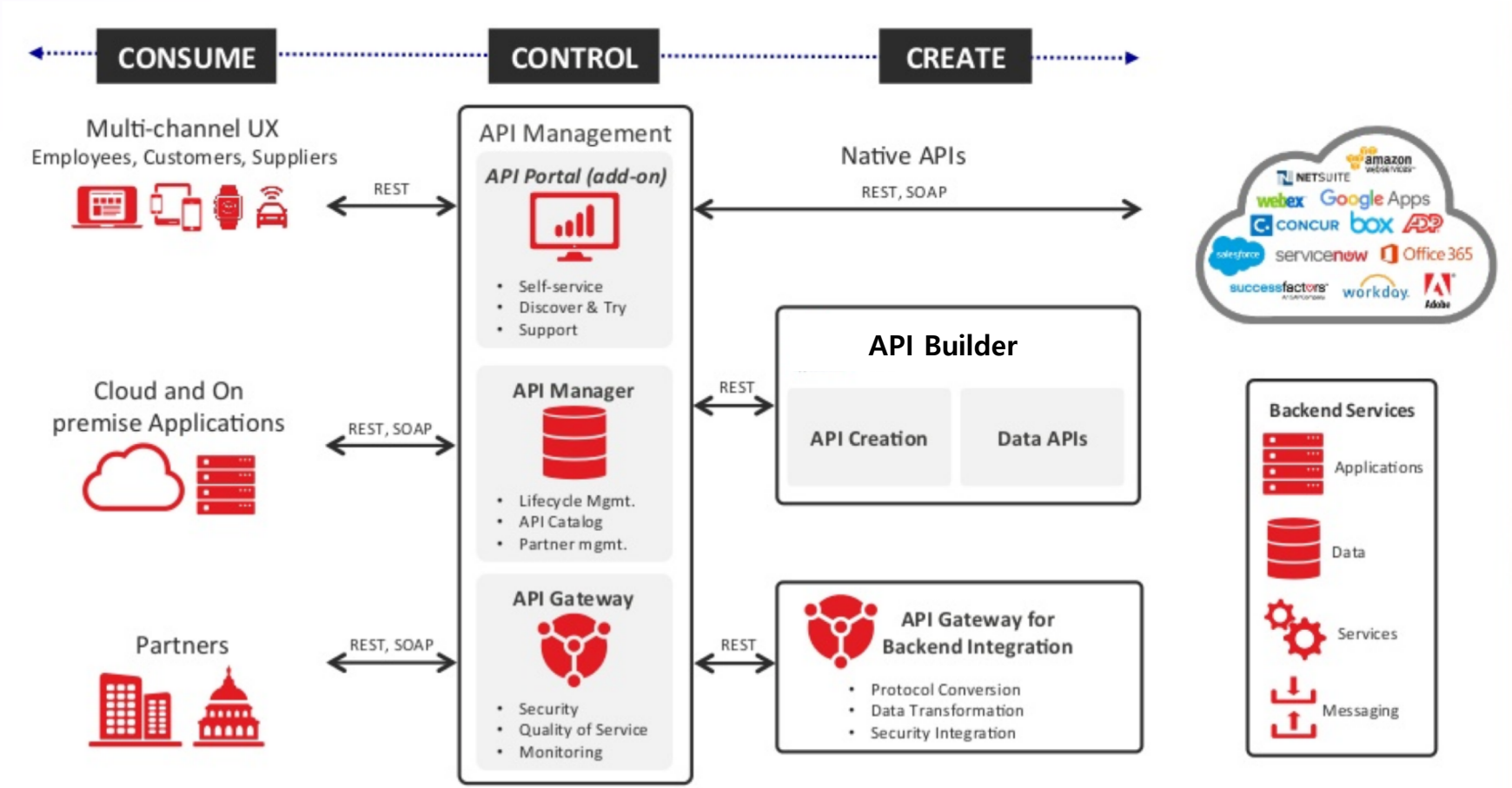
+



API Lifecycle Mgmt

Testing

API 시대 – 죽지 않는 여러 기법들 도입 (참조 – axway 제품)



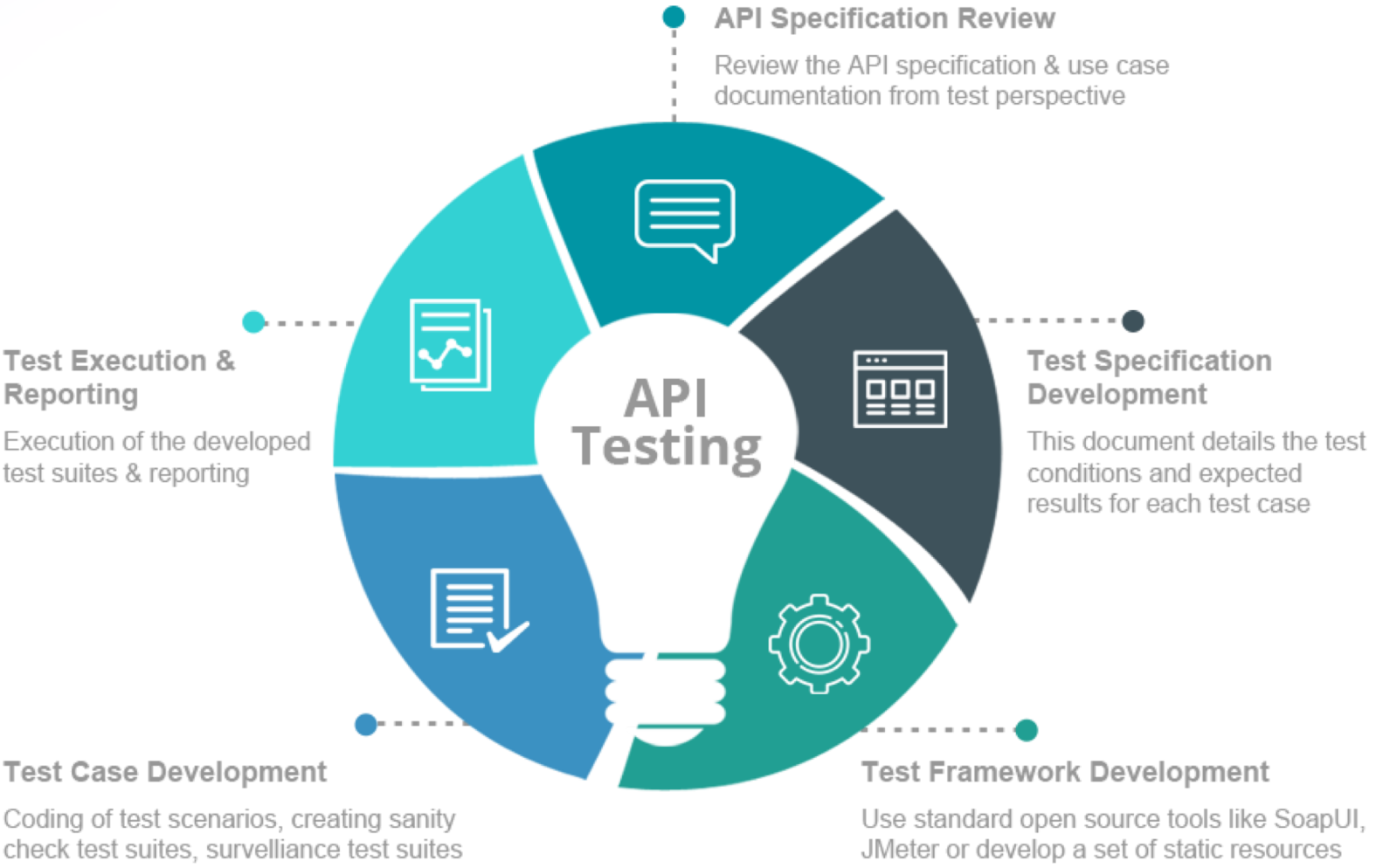
Testing

Testing

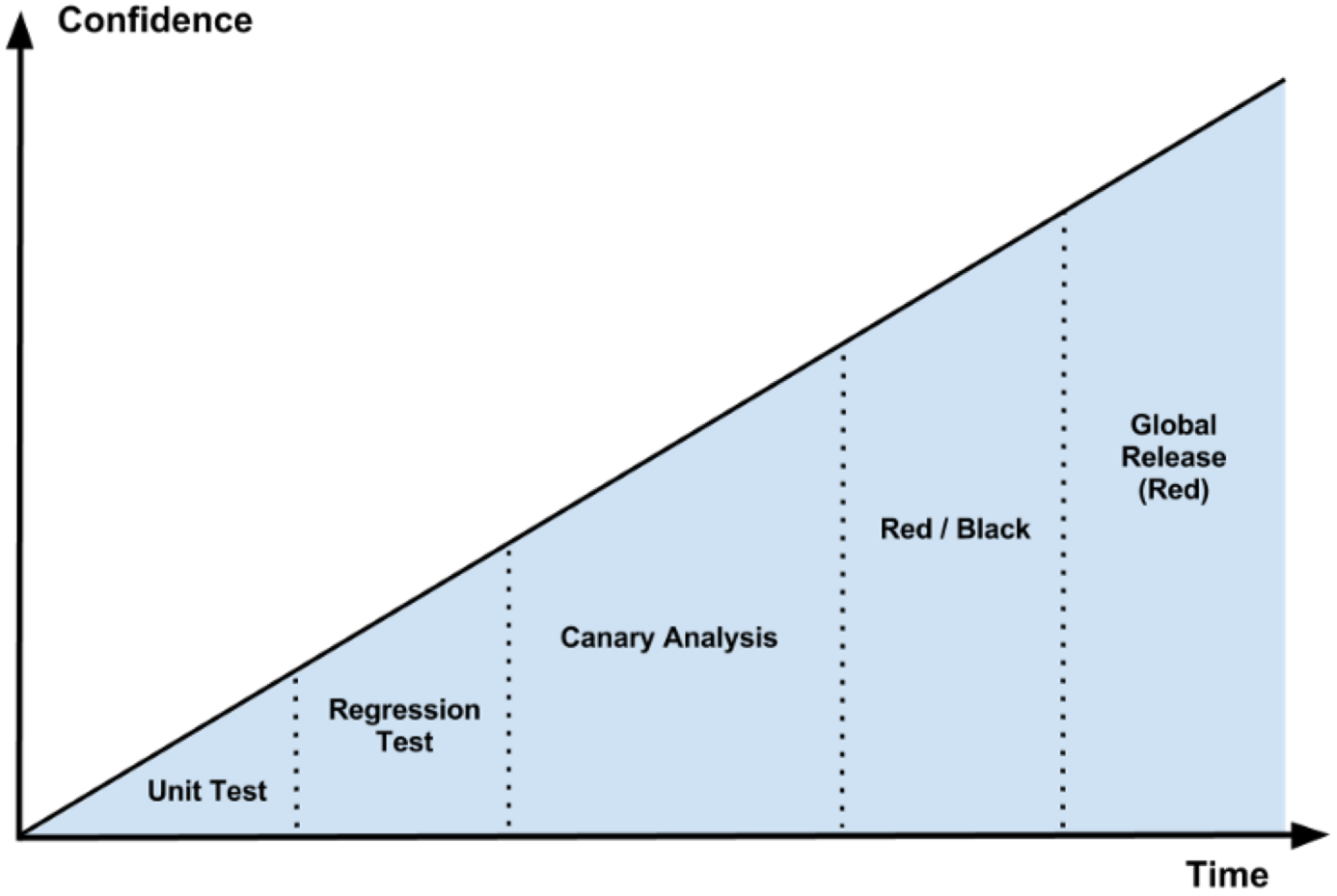
API 시대 – Netflix는 API Gateway 관련 기술을 다 내재화 시킴

Microservices Concern	Spring Cloud & Netflix OSS
Configuration Management	Config Server, Consul, Netflix Archaius
Service Discovery	Netflix Eureka, Hashicorp Consul
Load Balancing	Netflix Ribbon
API Gateway	Netflix Zuul
Service Security	Spring Cloud Security
Centralized Logging	ELK Stack (LogStash)
Centralized Metrics	Netflix Spectator & Atlas
Distributed Tracing	Spring Cloud Sleuth, Zipkin
Resilience & Fault Tolerance	Netflix Hystrix, Turbine & Ribbon
Auto Scaling & Self Healing	-
Packaging, Deployment & Scheduling	Spring Boot
Job Management	Spring Batch
Singleton Application	Spring Cloud Cluster

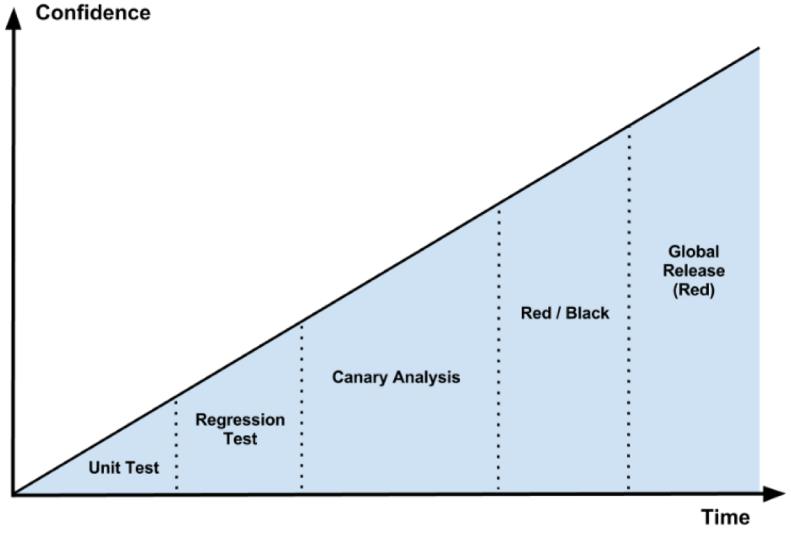
API Testing – 이 부분에 대해서는 내부 공개된 데이터가 없음

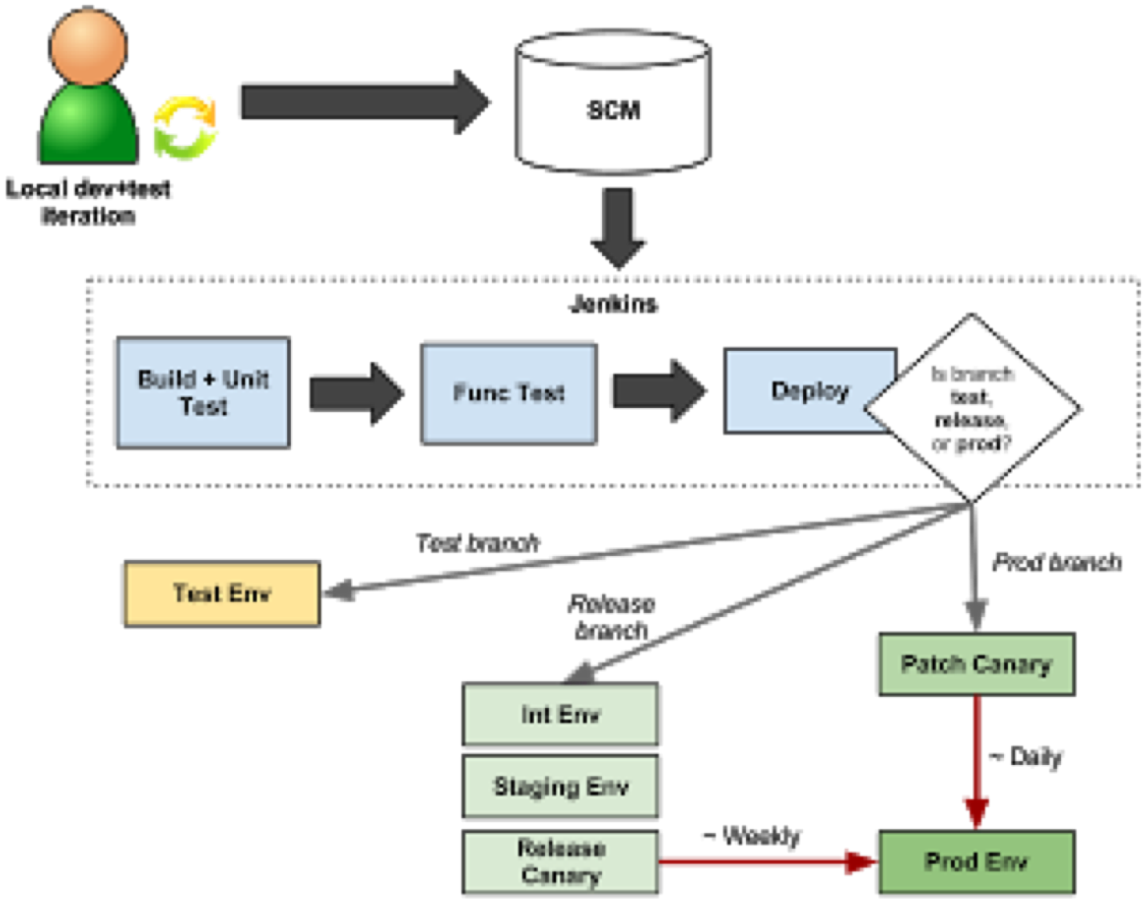


DevOps의 모범사례 – Netflix가 보는 Test의 흐름

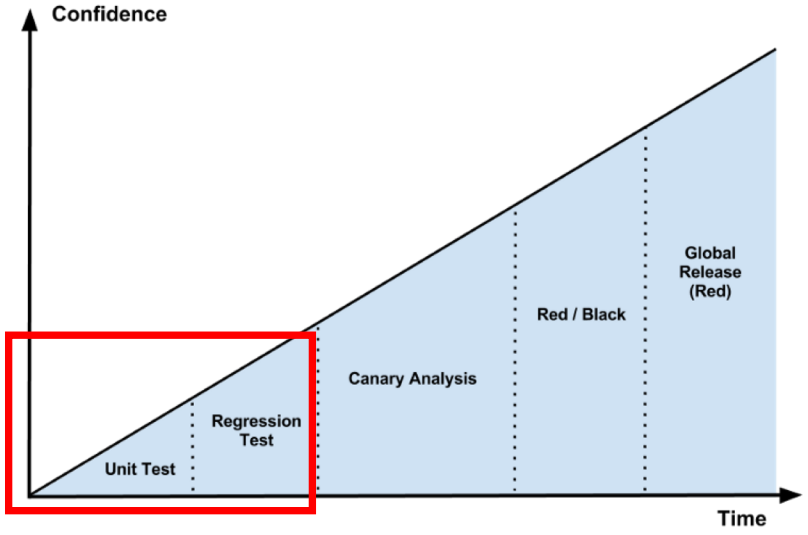


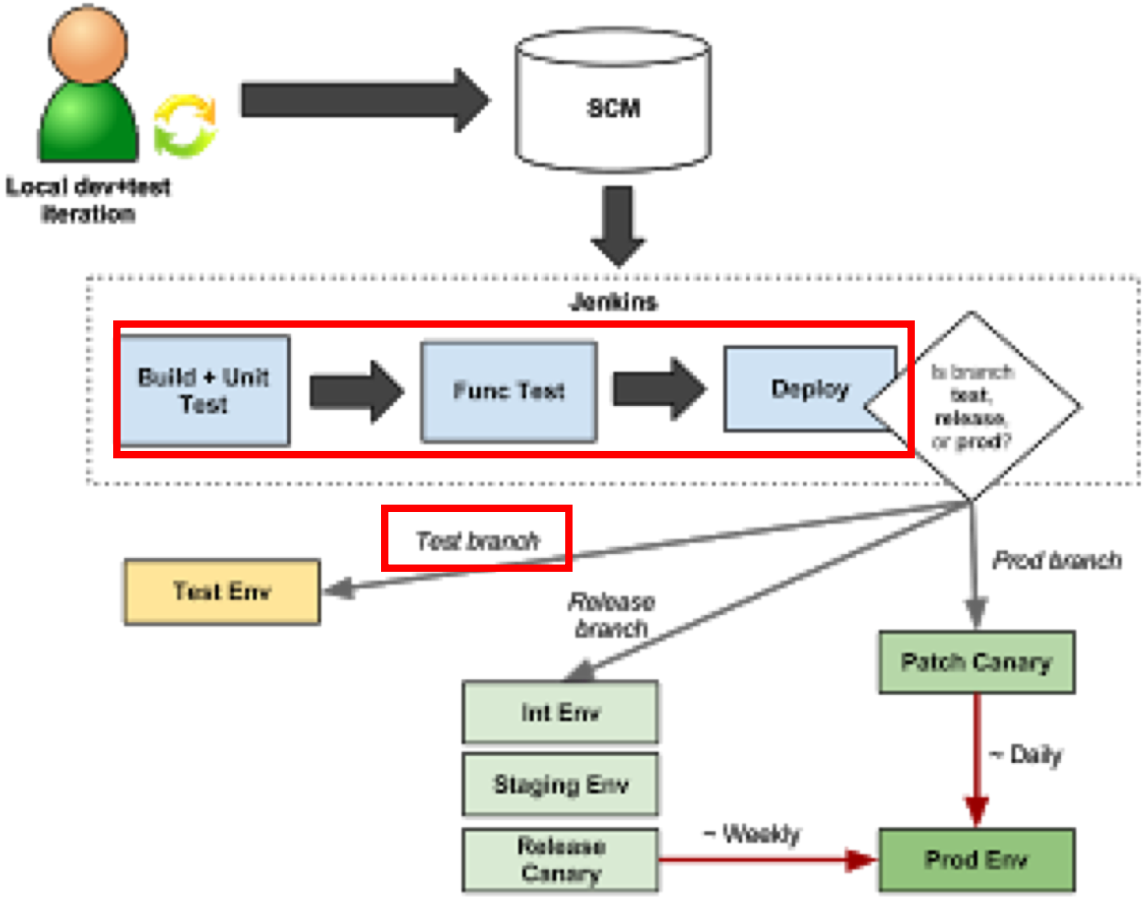
DevOps의 모범사례 – Branch(Test/Release/Prod) 별로 따로 관리



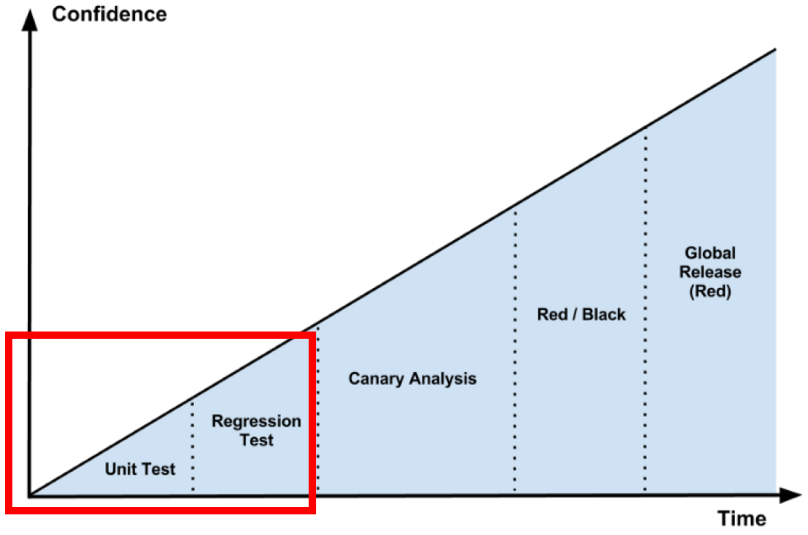


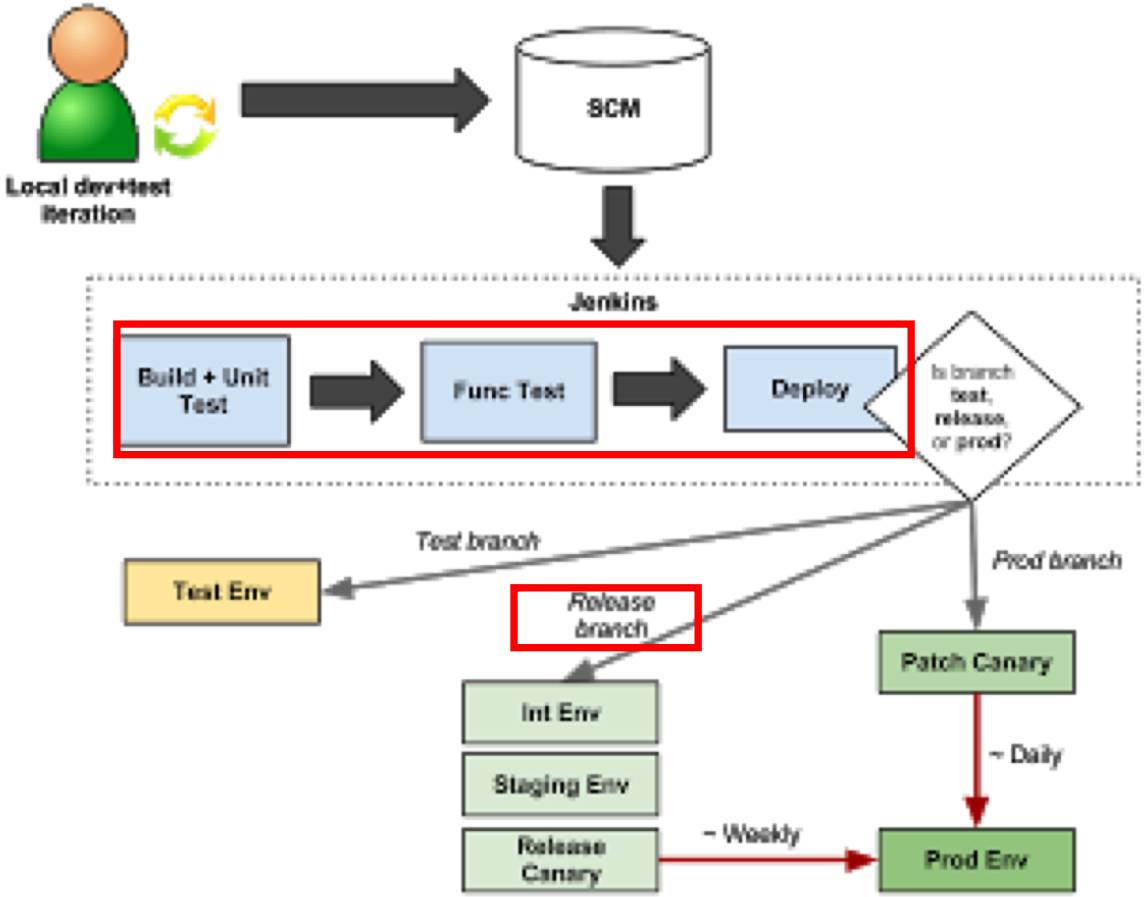
DevOps의 모범사례 – Unit / Func Test를 Test 브랜치에서 실행



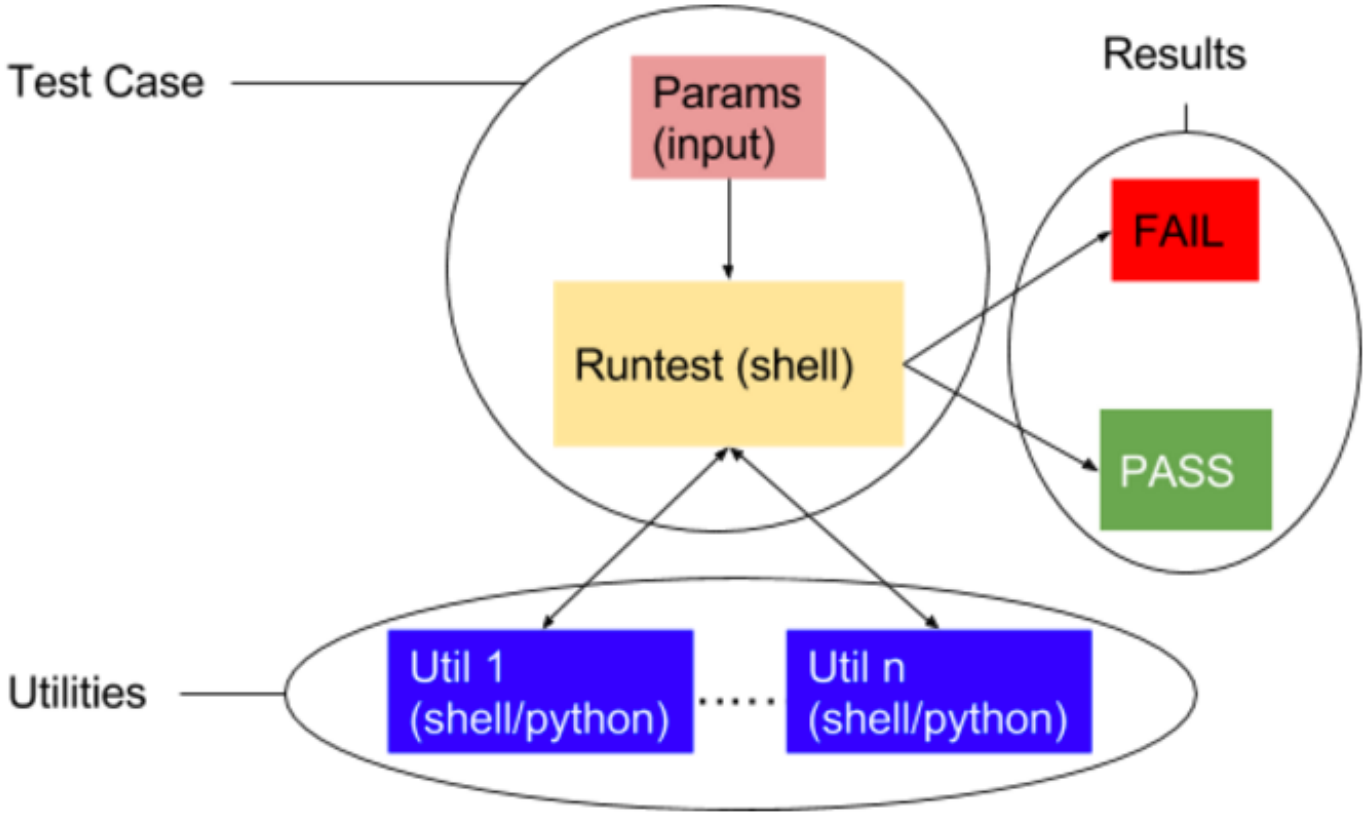


DevOps의 모범사례 – 성공하면 Release Branch에서 테스트

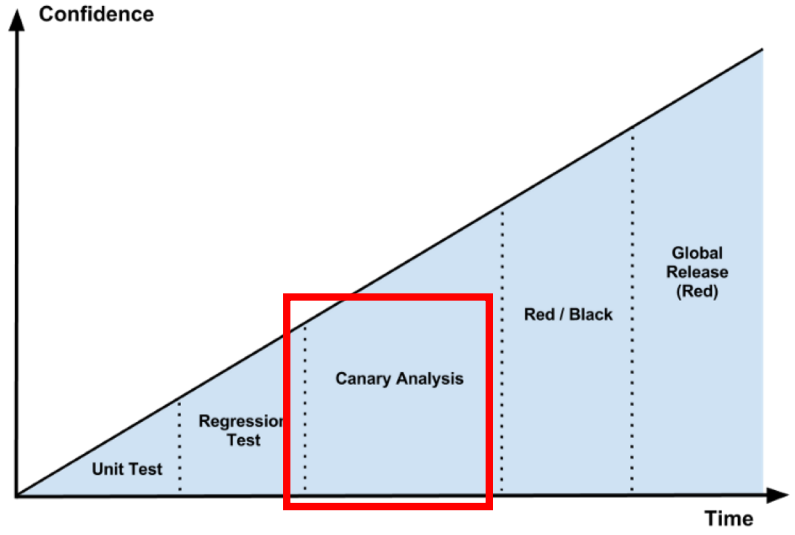


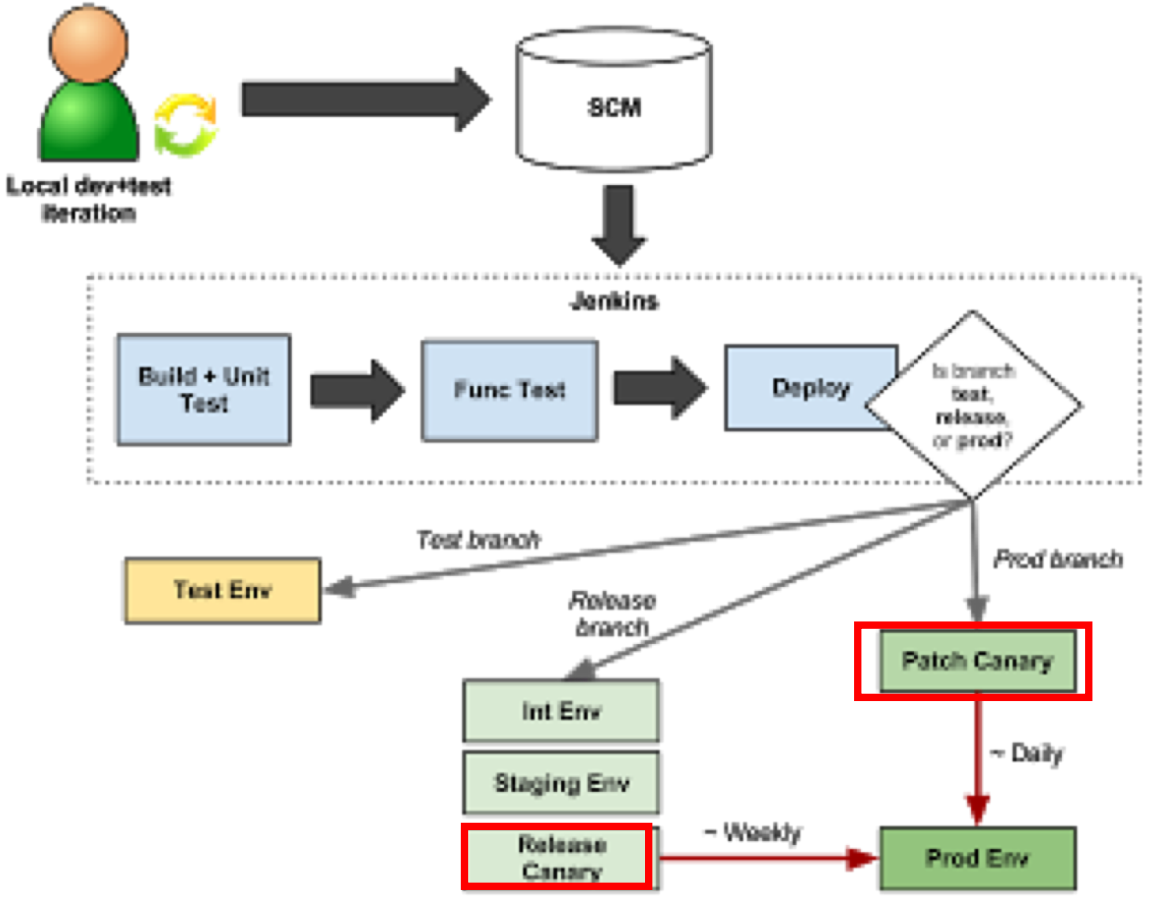


DevOps의 모범사례 – NETFLIX API Testing 을 Jenkins로 구축

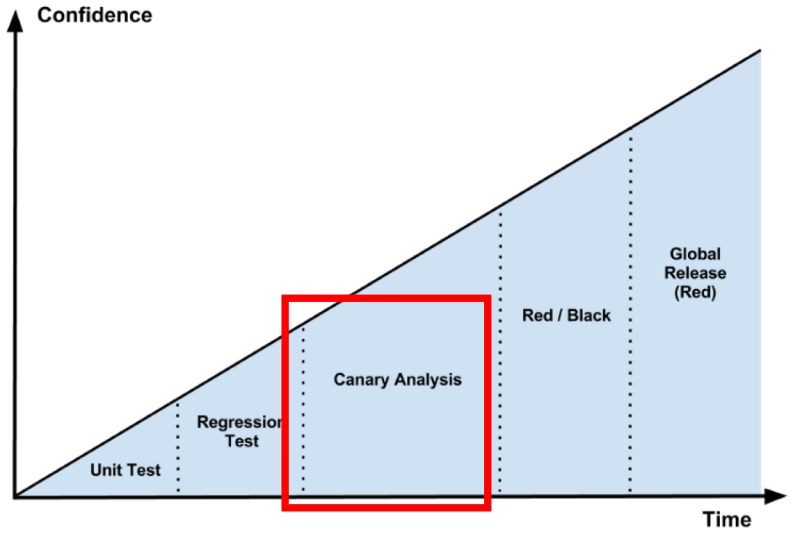


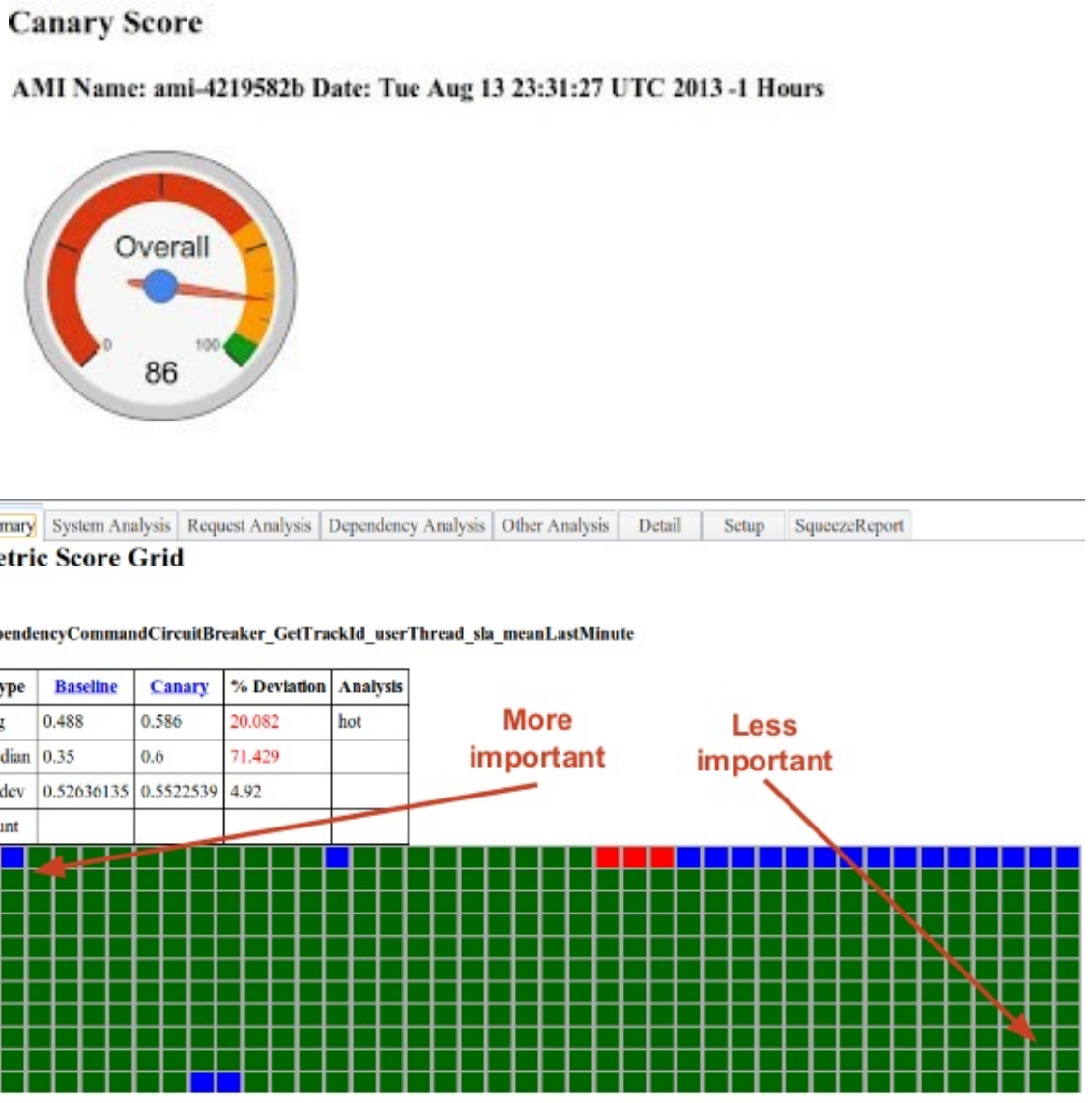
DevOps의 모범사례 – 또 성공하면 Canary를 날리자



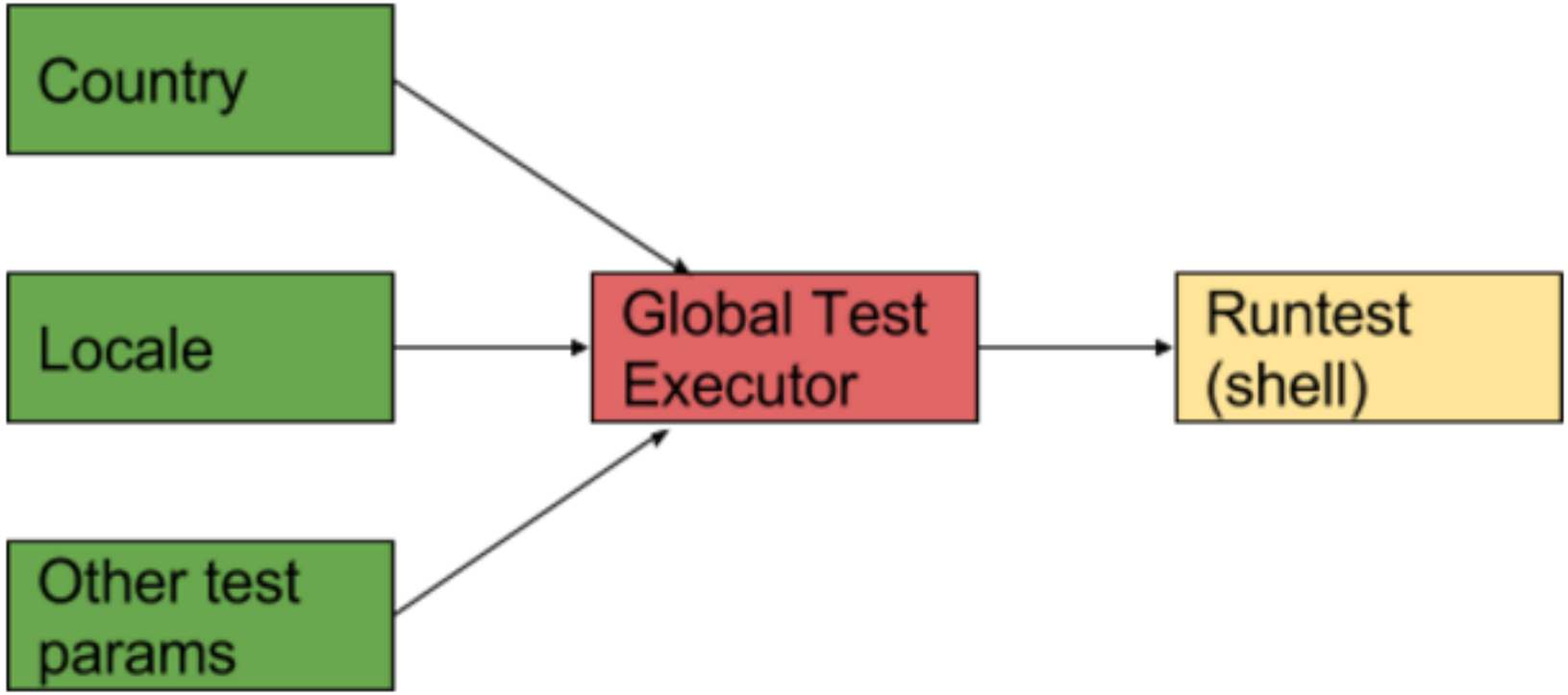


DevOps의 모범사례 – Baseline과 Canary에 대한 성능 비교 (기준치 도달 못하면 롤백)





Canary시 – 글로벌하게 테스트를 실행함.



점진적 배포 – Red / Black Push

API ASG
v1
AMI 123

1. Baseline Amazon Instance가 배포/작동중인 경우

API ASG
v1
AMI 123

API ASG
v2
AMI 456

2. Baseline AMI 의 10% 정도만 다음 버전 AMI Live 생성후 배포.
   새로운 버전의 안정성 체크.
   안정화 검증되면 새 버전 인스턴스 100대 만들기

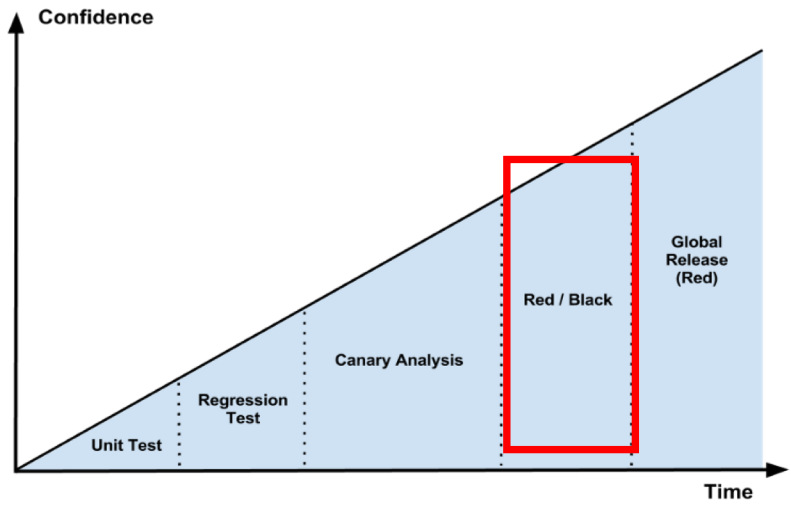
API ASG
v1
AMI 123

API ASG
v2
AMI 456

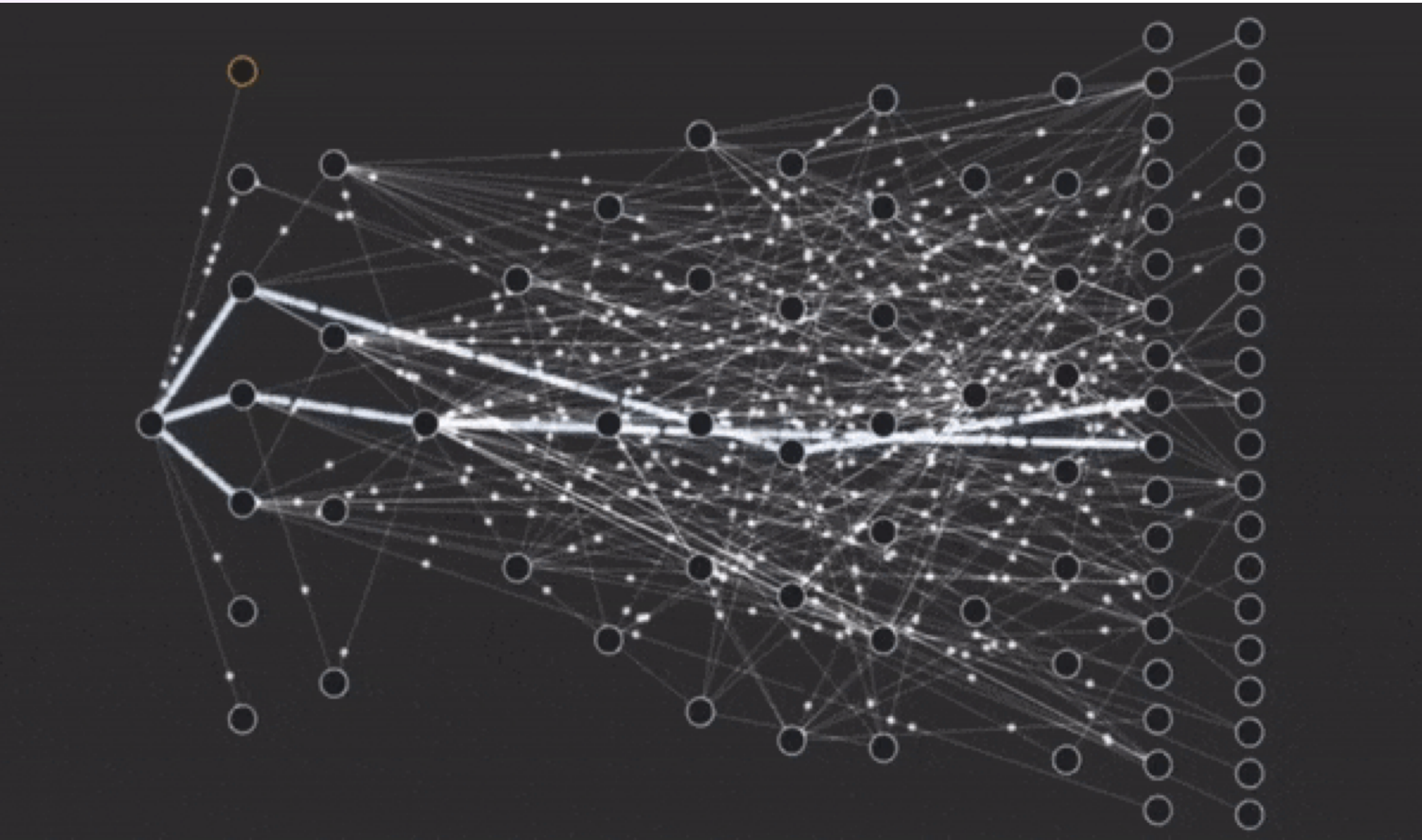
3. 기존 버전 (Baseline AMI) 정지 시킴 Black,
   그러니 새 버전만 작동 RED .
   이 상황이 RED/Black

API ASG
v2
AMI 456

4. 구 버전 AMI 내려버림
   새로운 버전의 AMI만 남게함

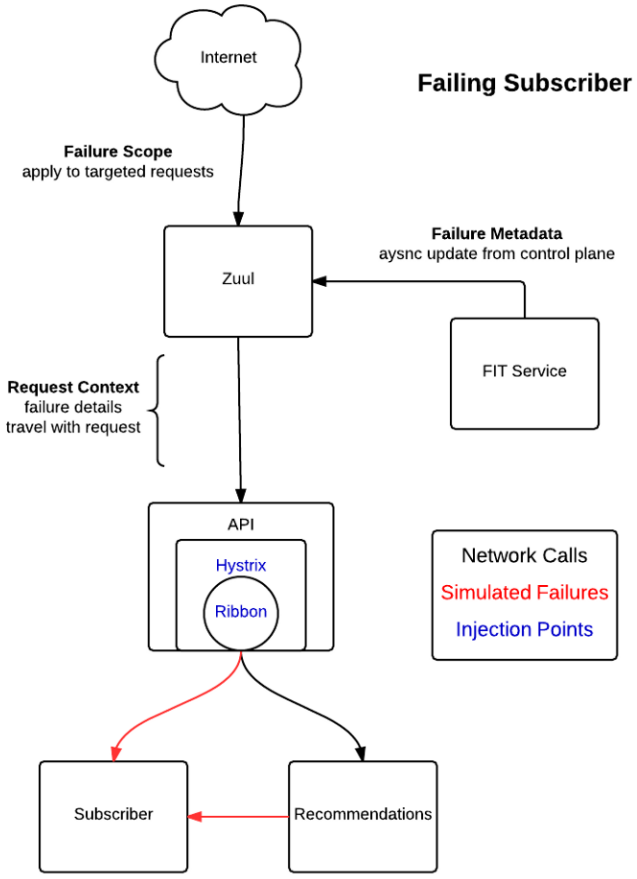


문제 없으면 - Global Release



Fault Injection Testing (Simian Army)

서버나 효율성이나 고객의 서비스의 안정성을 위해 가하는 테스팅 기법





Fault Injection Testing (Simian Army)

서버나 효율성이나 고객의 서비스의 안정성을 위해 가하는 테스팅 기법

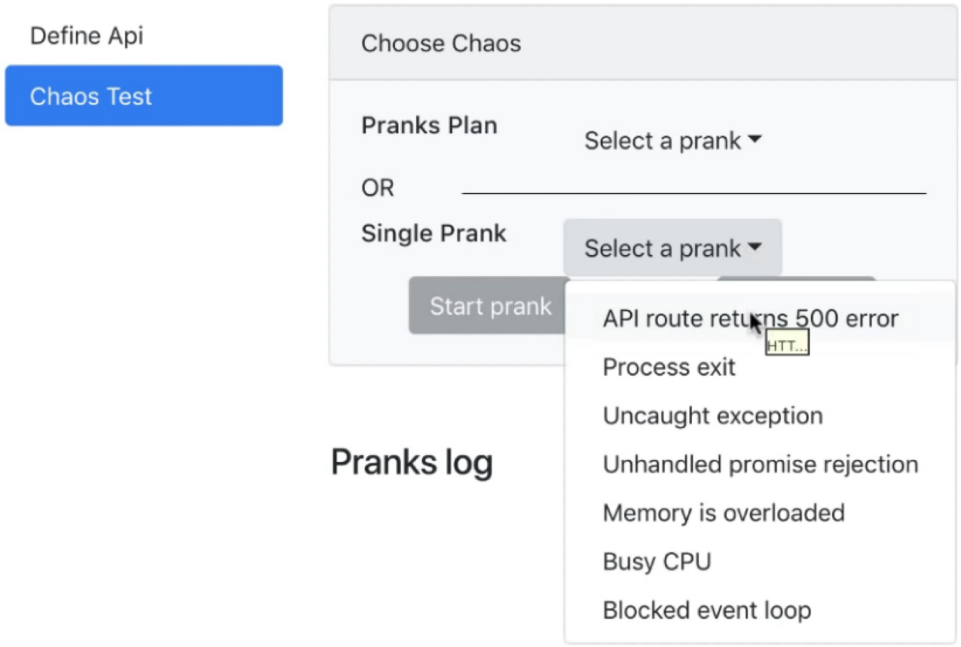
- Chaos Monkey
- Chaos Gorilla
- Chaos Kong
- Janitor Monkey
- Doctor Monkey
- Compliance Monkey
- Latency Monkey
- Security Monkey

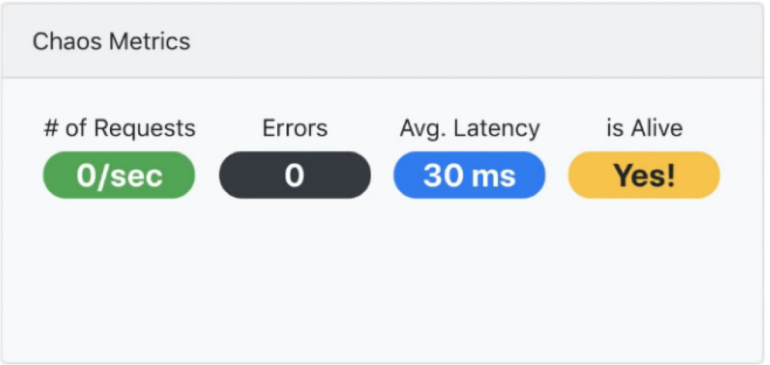


node-chaos-monkey

서버나 효율성이나 고객의 서비스의 안정성을 위해 가하는 테스팅 기법





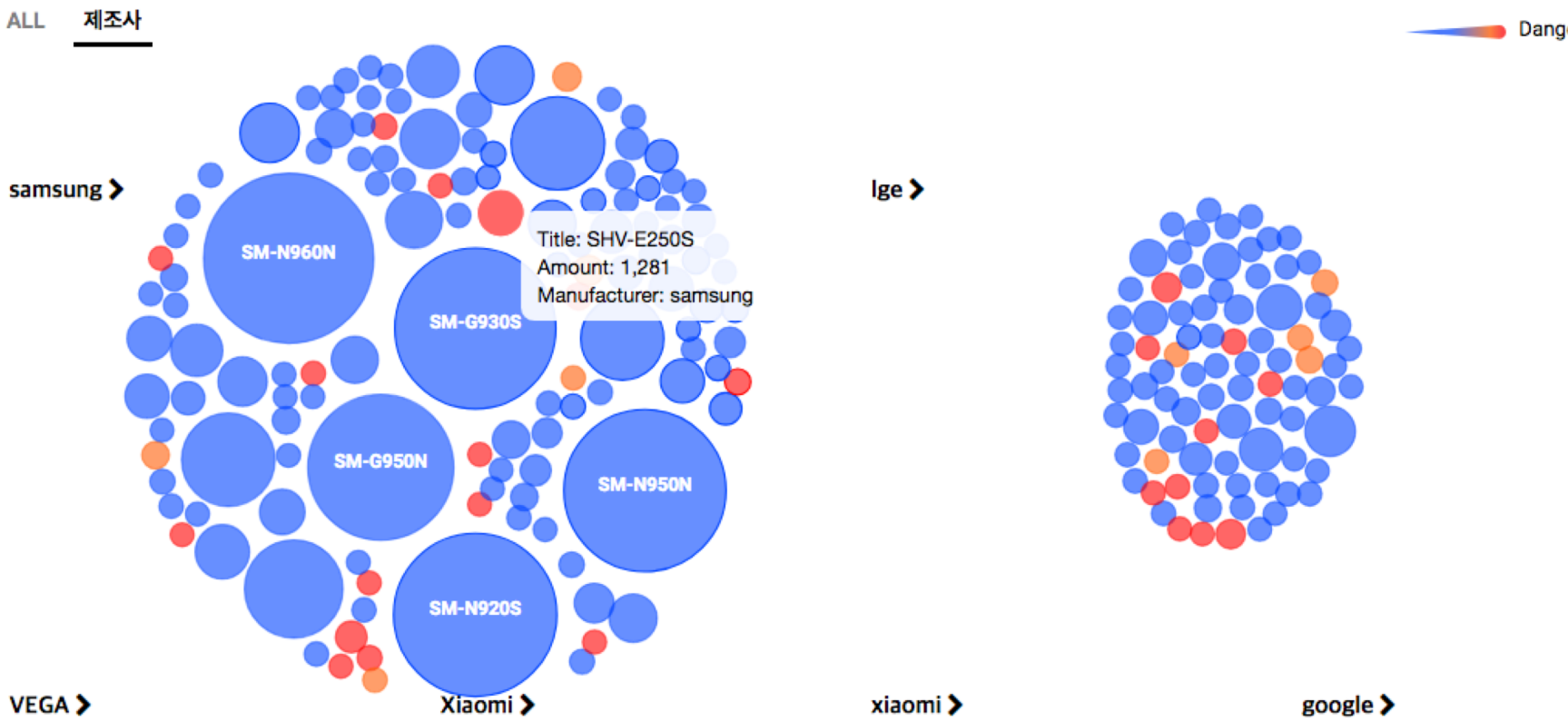




Part 2.
Mobile & DevOps

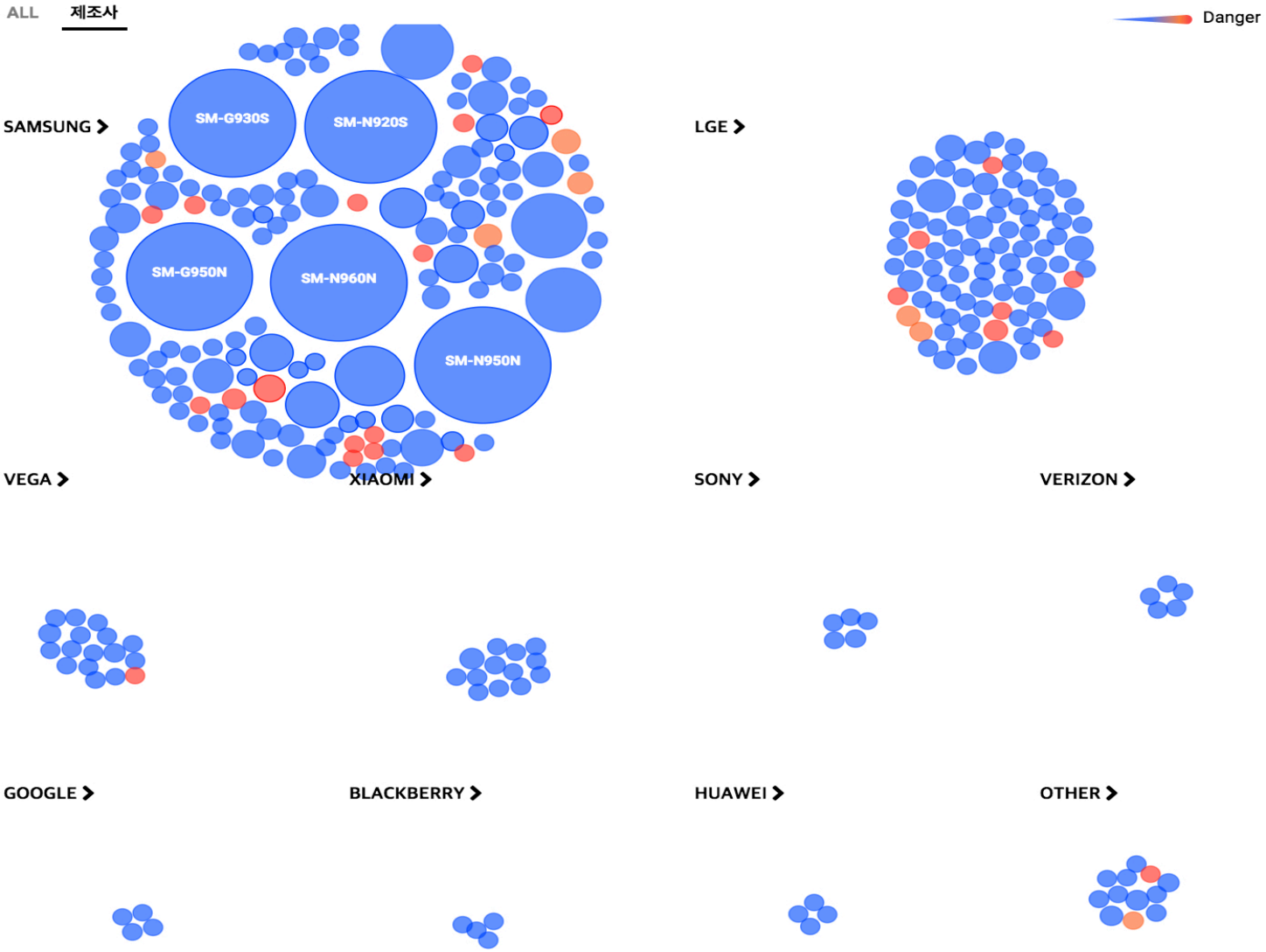
모바일 에서는 왜 DevOps가 필요해 졌나. (모바일)

파편화된 모바일 환경에서 어떻게 품질을 확보 해야지? (노트2의 점유율)

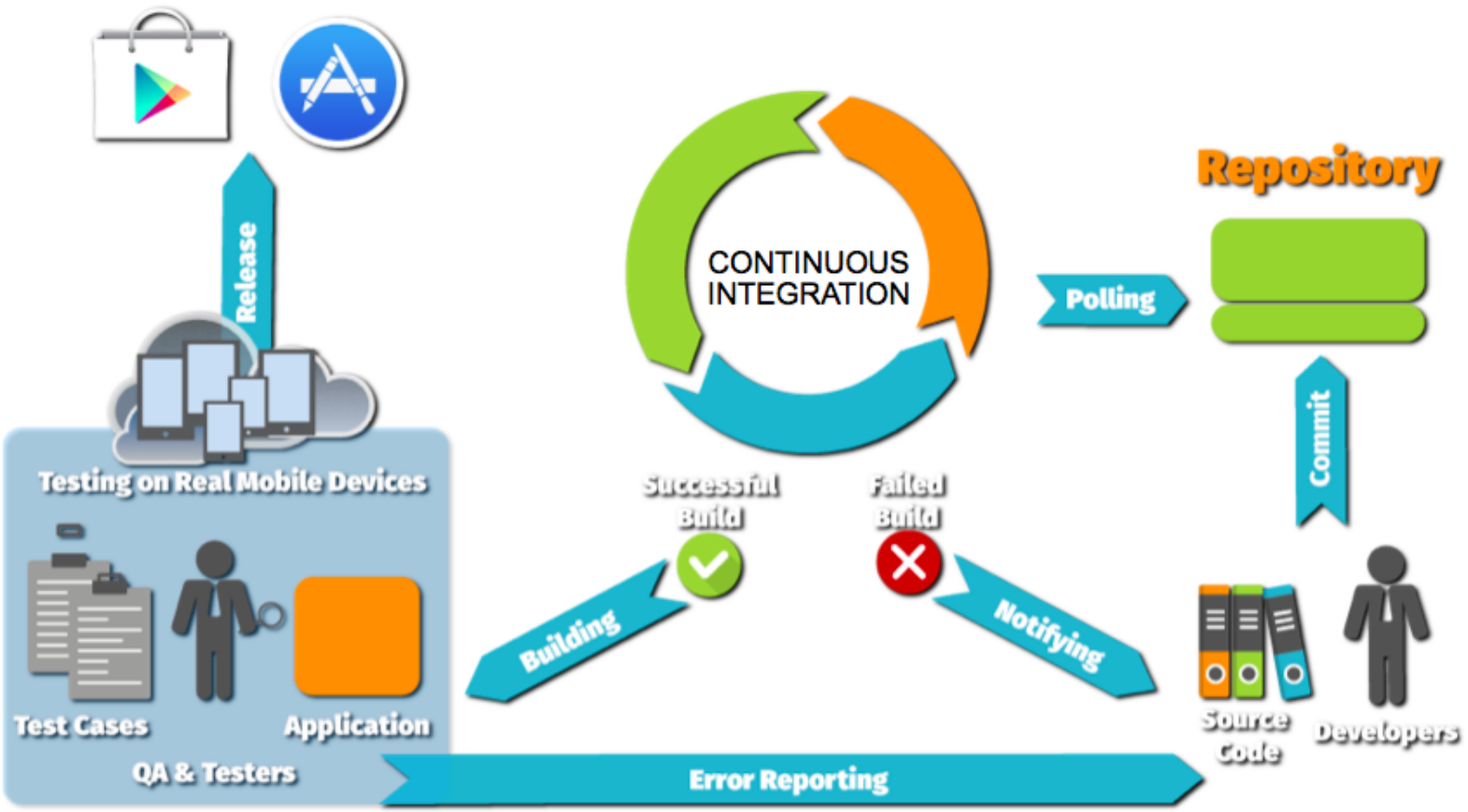


모바일 에서는 왜 DevOps가 필요해 졌나. (모바일)

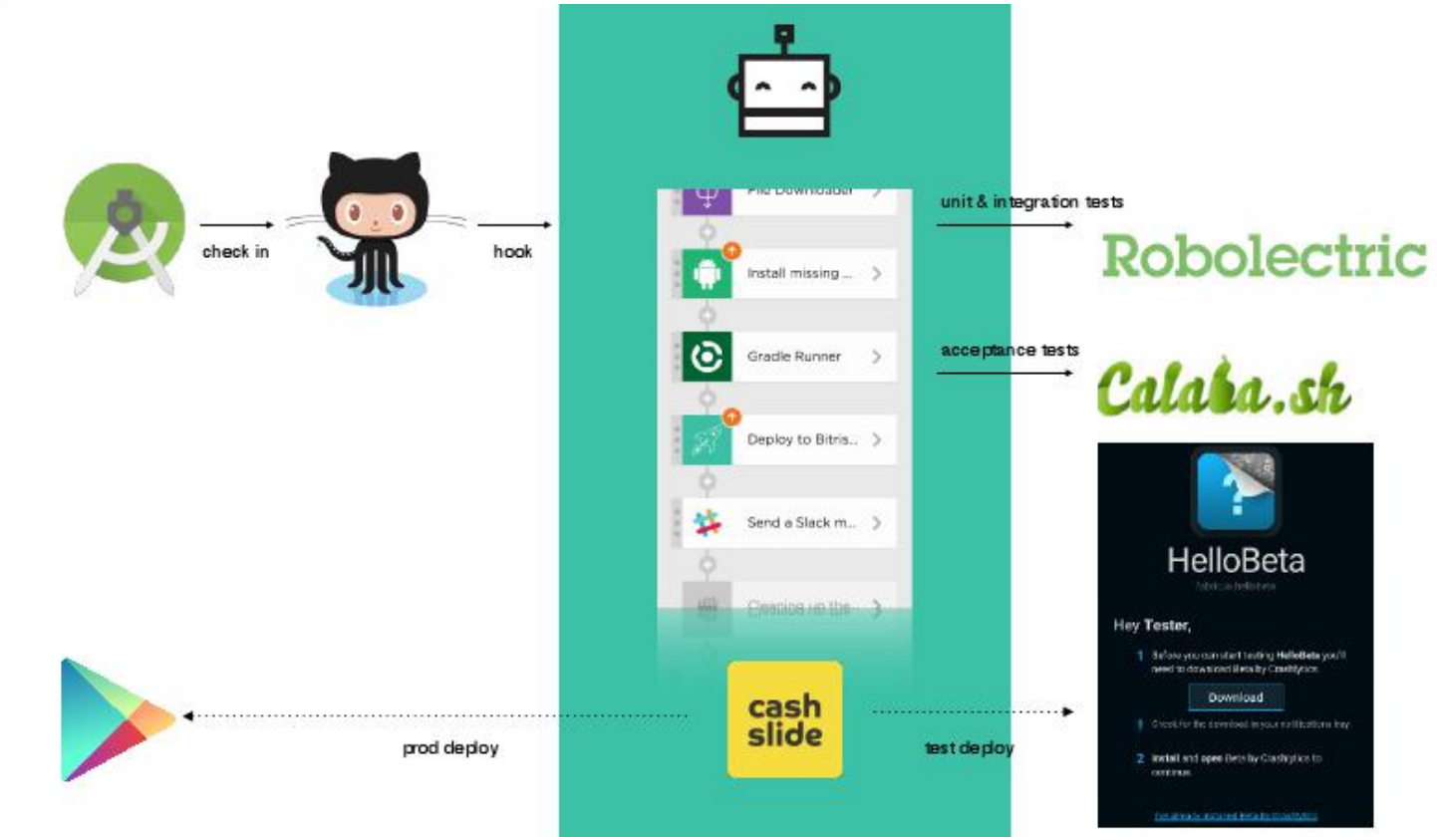
Xiaomi 폰의 국내 점유율 4위



모바일 DevOps 흐름도

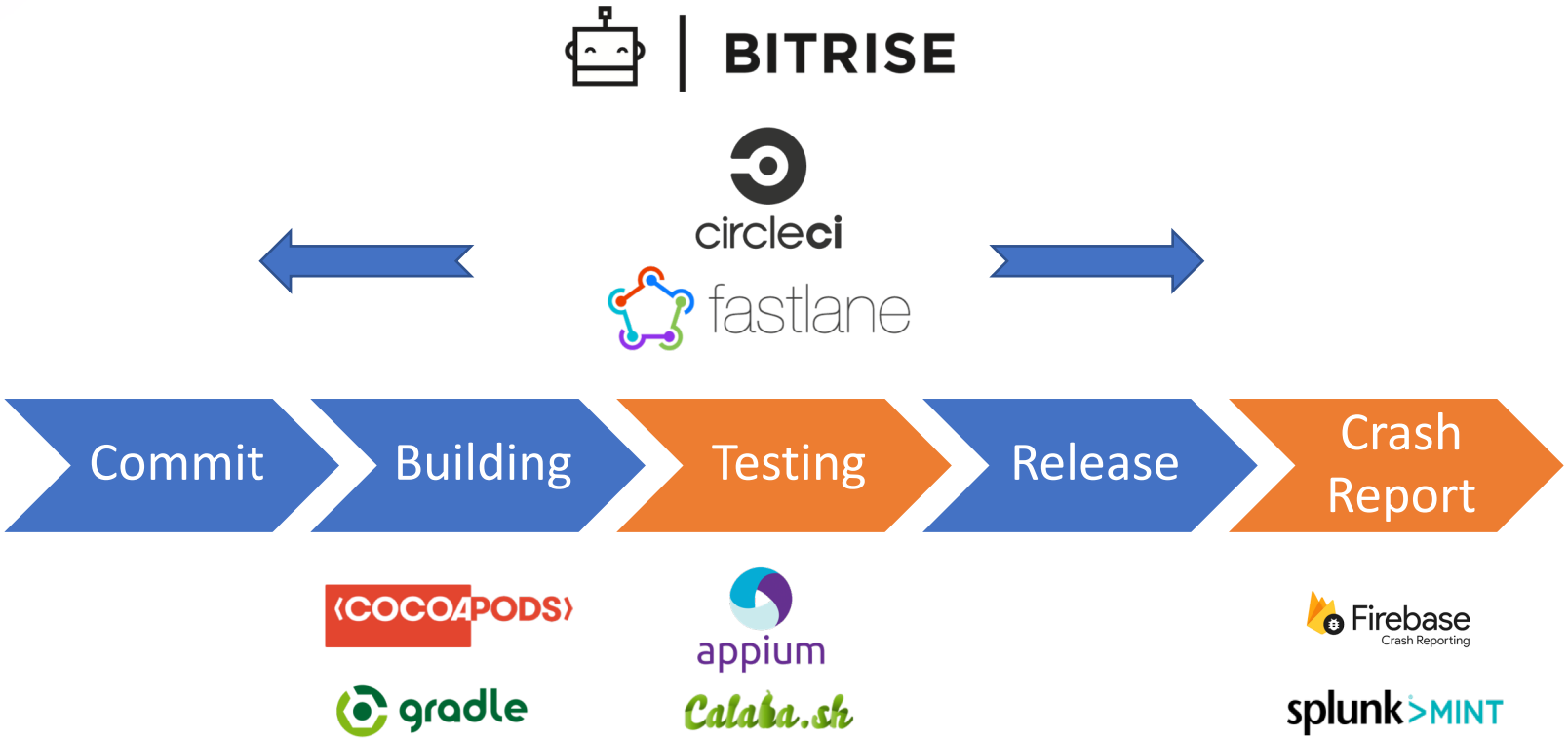


CashSlide 사례 - 모바일 DevOps 흐름도 (BITRISE 사용 사례)

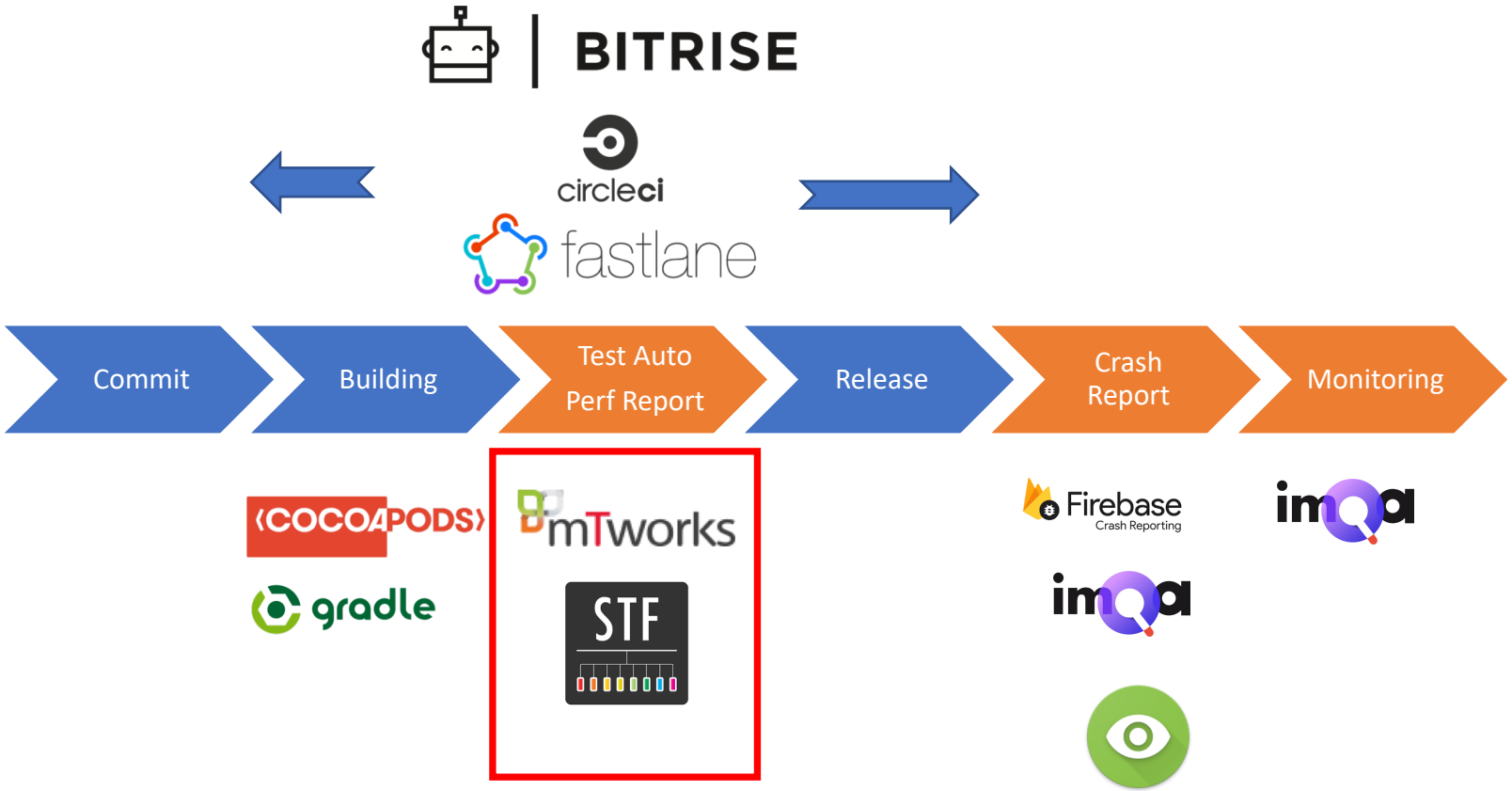


관련 자료 - https://academy.realm.io/kr/posts/continuous-delivery-for-android/

모바일 DevOps 현재 흐름도

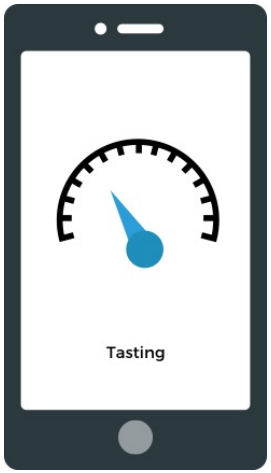


모바일 DevOps 제안 흐름도



더 나은 품질 확보를 위하여.

기존의 모바일 서비스는 출시 전 QA(Quality Assurance)를 통해 개발 관점에서의 사용자 만족도를 높여 왔습니다. 하지만 모바일 서비스의 각 사용자 별 환경에 따른 성능에 대해서는 현재까지도 스토어의 별점 등을 통해 피드백만 받아 온것이 현실입니다. 여기서 중요한 것은 성능의 문제점을 겪은 고객은 점점 해당 서비스의 이탈로 이어진다는 것입니다.



VS.



Testing

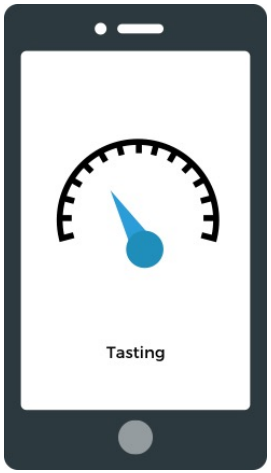
- 출시 전 오류 점검
- 기능 및 사용성에 대한 개발 관점
- 사용자의 만족도 예측

Performance Monitoring

- 출시 전, 후 성능 점검
- 성능 및 사용성에 대한 운영 관점
- 사용자의 만족도 실측

더 나은 품질 확보를 위하여.

기존의 모바일 서비스는 출시 전 QA(Quality Assurance)를 통해 개발 관점에서의 사용자 만족도를 높여 왔습니다. 하지만 모바일 서비스의 각 사용자 별 환경에 따른 성능에 대해서는 현재까지도 스토어의 별점 등을 통해 피드백만 받아 온것이 현실입니다. 여기서 중요한 것은 성능의 문제점을 겪은 고객은 점점 해당 서비스의 이탈로 이어진다는 것입니다.



VS.



Device Farm

Crash, Monitoring

mTworks

STF

Firebase Crash Reporting

imQa

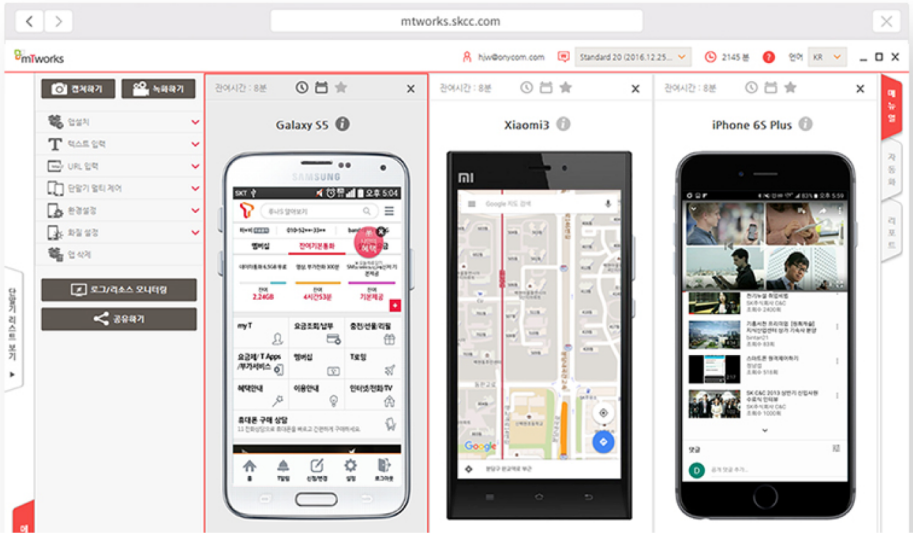
AndroidGodEye

mTworks - mtworks.skcc.com
(국내 최대 레퍼런스 디바이스 팜 – Android / iOS 140개의 디바이스 제공)

언제 어디서든
인터넷을 통해 단말기에 접속하는
새로운 차원의 모바일 테스트 서비스

인터넷을 통해 실제 단말기에 접속하여 테스트를 할 수 있으며,
자동화 툴을 활용하여 업무 생산성을 높일 수 있습니다.

mTworks 소개





Samsung Remote Test Lab
(삼성 단말만 가능 / 삼성 가전 제품도 일부 제공 – 완전 무료라는 이점)





Firebase Test Lab.
(google 이 30~40 개 종류의 디바이스 제공 / 북미 위주의 디바이스 편성)

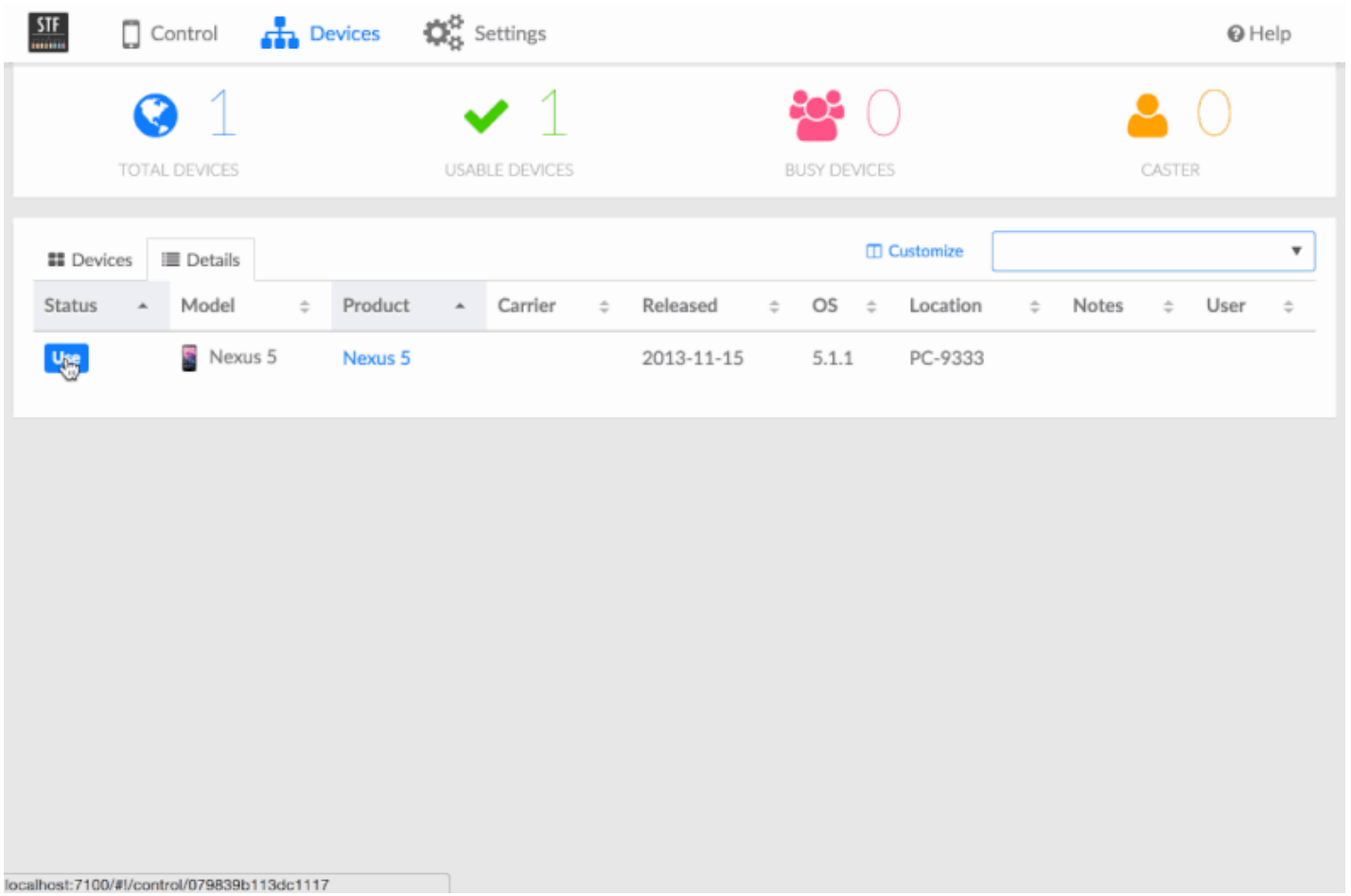
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\slava> gcloud firebase test android models list

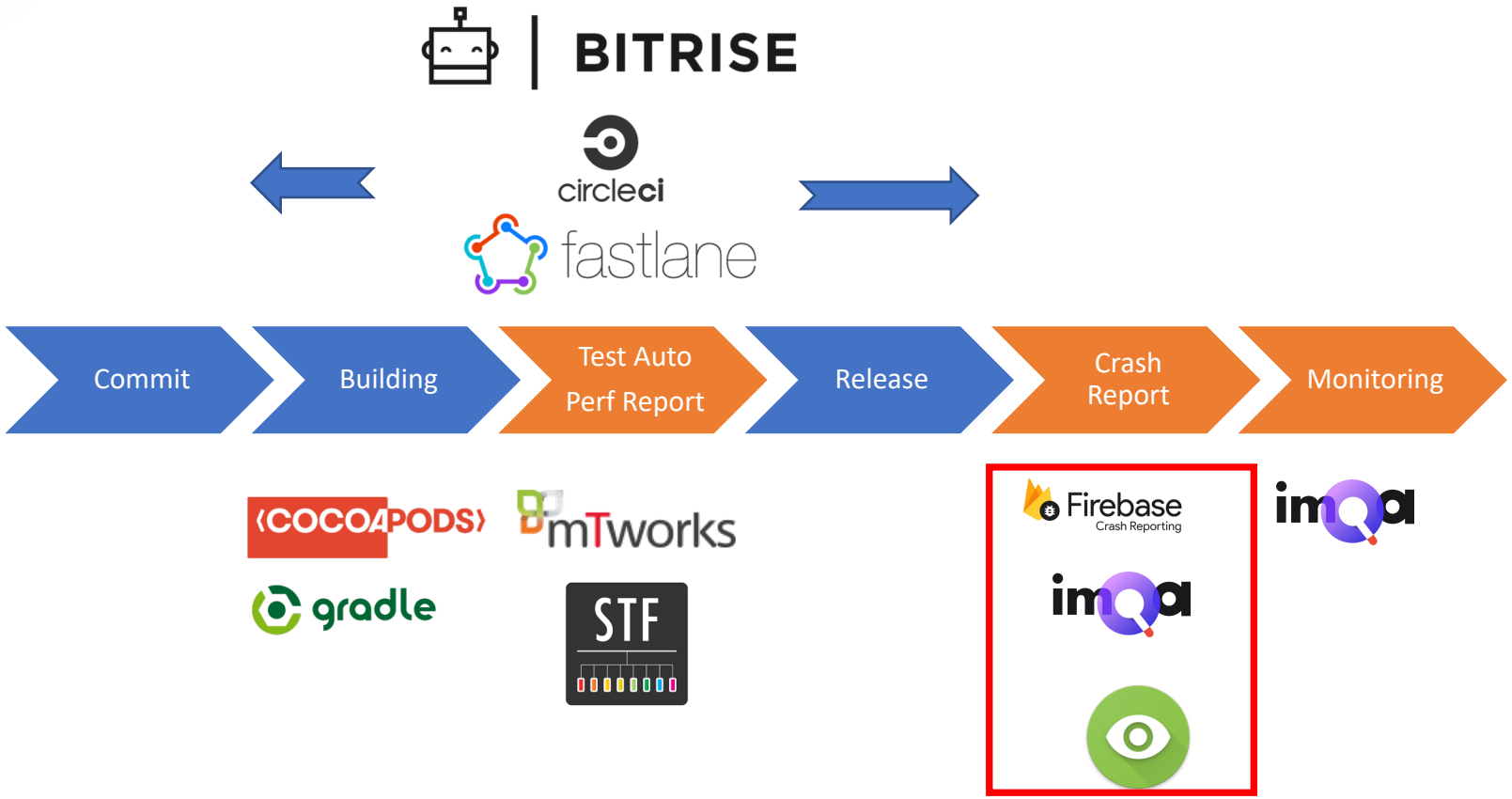
Have questions, feedback, or issues? Get support by visiting:
  https://firebase.google.com/support/
```

MODEL_ID	MAKE	MODEL_NAME	FORM	RESOLUTION	OS_VERSION_IDS	TAGS
A0001	OnePlus	OnePlus One	PHYSICAL	1920 x 1080	22	
D6503	Sony	Xperia Z2	PHYSICAL	1920 x 1080	21	
D6603	Sony	Xperia Z3	PHYSICAL	1920 x 1080	21	
E5803	Sony	Xperia Z5 Compact	PHYSICAL	1280 x 720	22	
F5121	Sony	Sony Xperia X	PHYSICAL	1920 x 1080	23	
Nexus10	Samsung	Nexus 10	VIRTUAL	2560 x 1600	19,21,22	
Nexus4	LG	Nexus 4	VIRTUAL	1280 x 768	19,21,22	
Nexus5	LG	Nexus 5	VIRTUAL	1920 x 1080	19,21,22,23	
Nexus5X	LG	Nexus 5X	VIRTUAL	1920 x 1080	23,24,25	
Nexus6	Motorola	Nexus 6	VIRTUAL	2560 x 1440	21,22,23,24,25	
Nexus6P	Google	Nexus 6P	VIRTUAL	2560 x 1440	23,24,25	
Nexus7	ASUS	Nexus 7 (2012)	VIRTUAL	1280 x 800	19,21,22	
Nexus9	HTC	Nexus 9	VIRTUAL	2048 x 1536	21,22,23,24,25	
NexusLowRes	Generic	Low-resolution MDPI phone	VIRTUAL	640 x 360	23,24,25	
SH-04H	SHARP	SH-04H	PHYSICAL	1920 x 1080	23	
athene	Motorola	Moto G4 Plus	PHYSICAL	1920 x 1080	23	
athene_f	Motorola	Moto G4	PHYSICAL	1920 x 1080	23	
condor_umts	Motorola	Moto E	PHYSICAL	960 x 540	19	
falcon_umts	Motorola	Moto G (1st Gen)	PHYSICAL	1280 x 720	19	
flo	ASUS	Nexus 7 (2013)	PHYSICAL	1920 x 1200	19,21	
flounder	HTC	Nexus 9	PHYSICAL	2048 x 1536	21	
g3	LG	LG G3	PHYSICAL	2560 x 1440	19	
hammerhead	LG	Nexus 5	PHYSICAL	1920 x 1080	19,21,22,23	default
hero2lte	Samsung	Galaxy S7 edge	PHYSICAL	1440 x 2560	23	
herolte	Samsung	Galaxy S7	PHYSICAL	1440 x 2560	23,24	
hlte	Samsung	Galaxy Note 3 Duos	PHYSICAL	1920 x 1080	19	
htc_m8	HTC	HTC One (M8)	PHYSICAL	1920 x 1080	19	
j5lte	Samsung	Galaxy J5	PHYSICAL	1280 x 720	23	
ltO2wifi	Samsung	Galaxy Tab 3	PHYSICAL	600 x 1024	19	
m0	Samsung	Samsung Galaxy S3	PHYSICAL	1280 x 720	18	
mako	LG	Nexus 4	PHYSICAL	1280 x 768	19,22	
osprey_umts	Motorola	Moto G (3rd Gen)	PHYSICAL	1280 x 720	22	
p1	LG	LG G4	PHYSICAL	1440 x 2560	22	
sailfish	Google	Pixel	PHYSICAL	1080 x 1920	25,26	
serranolte	Samsung	Galaxy S4 mini	PHYSICAL	960 x 540	19	
shamu	Motorola	Nexus 6	PHYSICAL	2560 x 1440	21,22,23	
t03g	Samsung	Galaxy Note 2	PHYSICAL	1280 x 720	19	
titan_umts	Motorola	Moto G (2nd Gen)	PHYSICAL	1280 x 720	19	
trelte	Samsung	Galaxy Note 4	PHYSICAL	1920 x 1080	22	
victara	Motorola	Moto X	PHYSICAL	1920 x 1080	19	
zeroflte	Samsung	Galaxy S6	PHYSICAL	2560 x 1440	22	
zerolte	Samsung	Galaxy S6 Edge	PHYSICAL	2560 x 1440	22	

OpenSTF.IO - https://github.com/openstf/stf
(회사내부에 직접 디바이스 Farm 을 구축하고 싶을때. – 안드로이드만 지원)



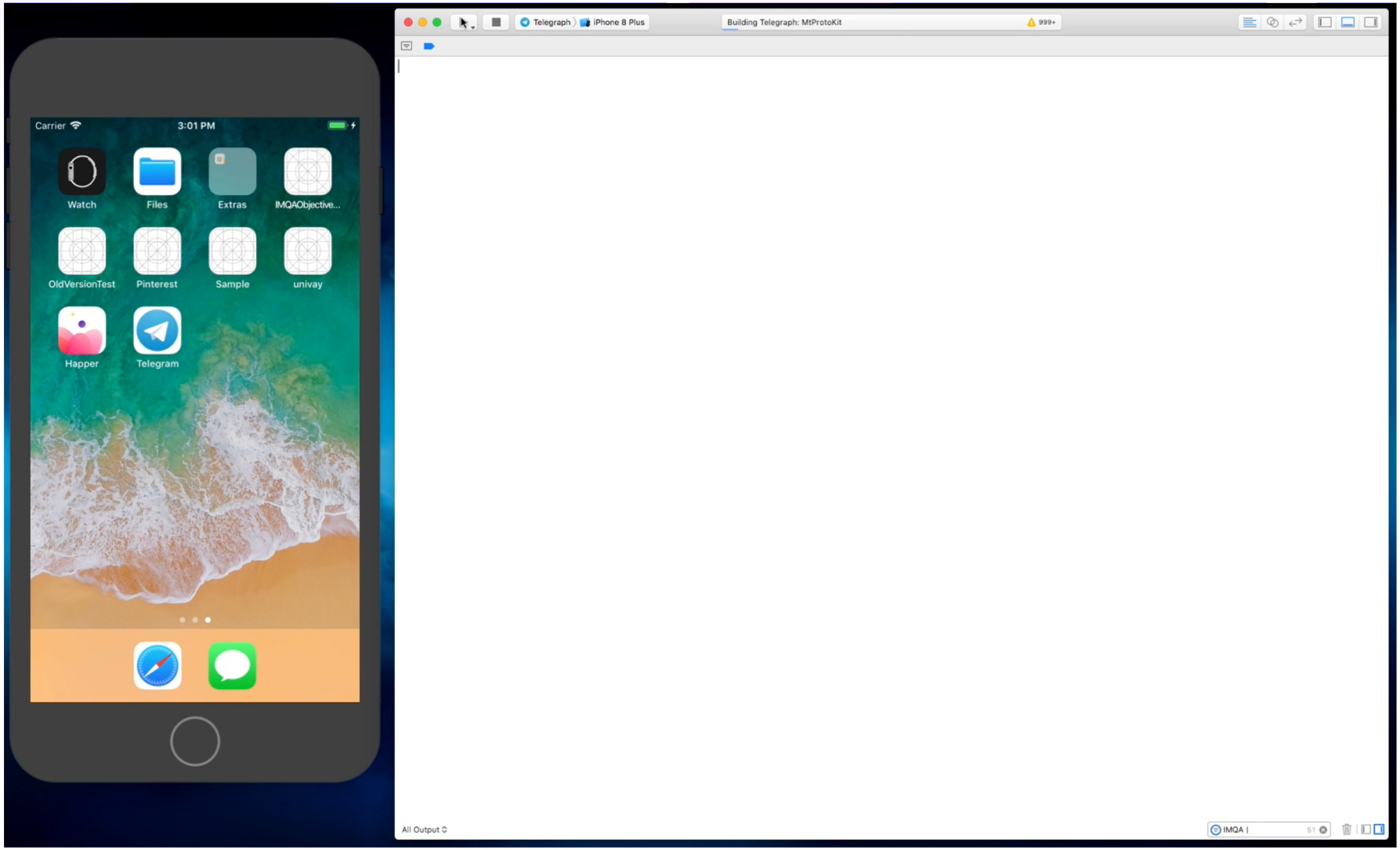
모바일 DevOps 제안 흐름도



IMQA 사례 - 실 사례로 보는 모바일 성능 확보 방안



데이터를 수집하는 방법

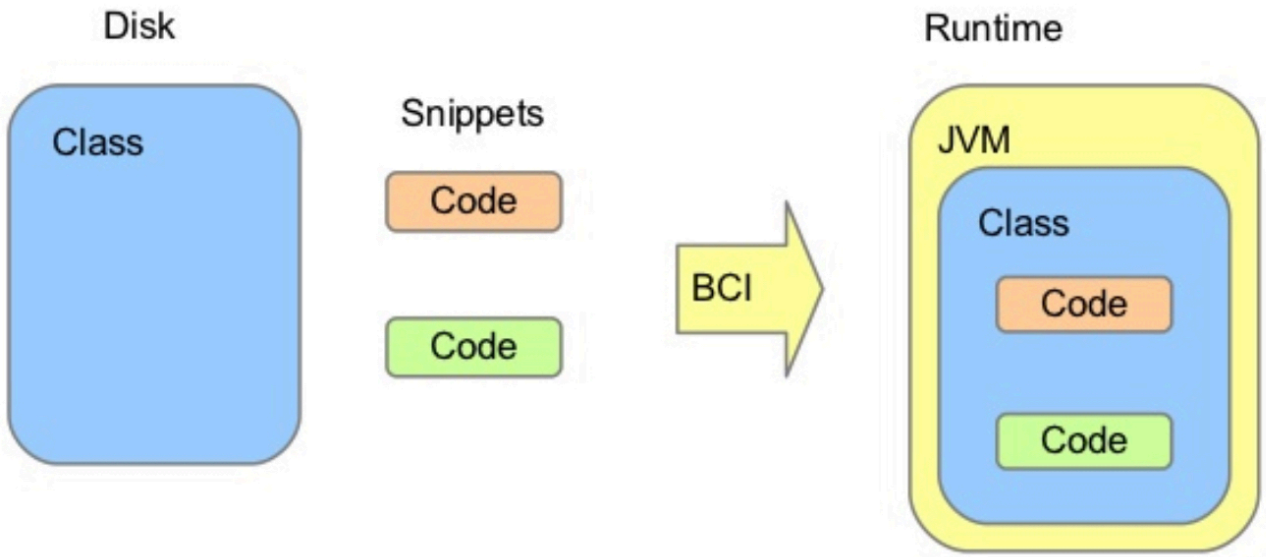

Carrier
3:01 PM
Watch
Files
Extras
IMQAObjective...
OldVersionTest
Pinterest
Sample
univay
Happer
Telegram
Telegraph
iPhone 8 Plus
Building Telegraph: MtProtoKit
All Output
IMQA

주요 수집 데이터

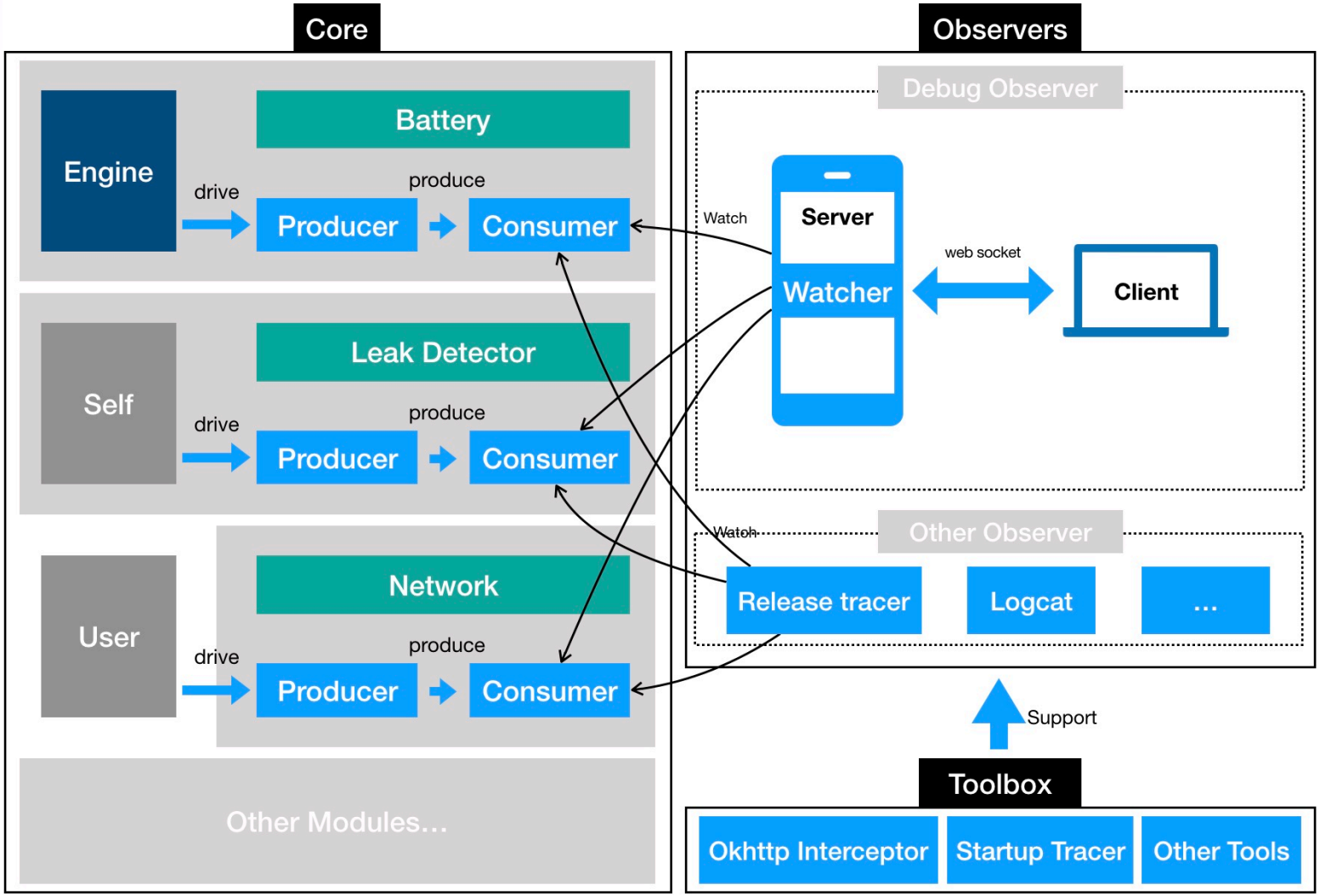
- 화면 이름 (Activity / View)
- UI Rendering Time (FPS 와 Lifecycle 다 고려)
- CPU (App / OS)
- Memory (App / OS)
- Network Response Time
- 코드 스택
- Battery Usage
- ...

어떻게 수집하나?

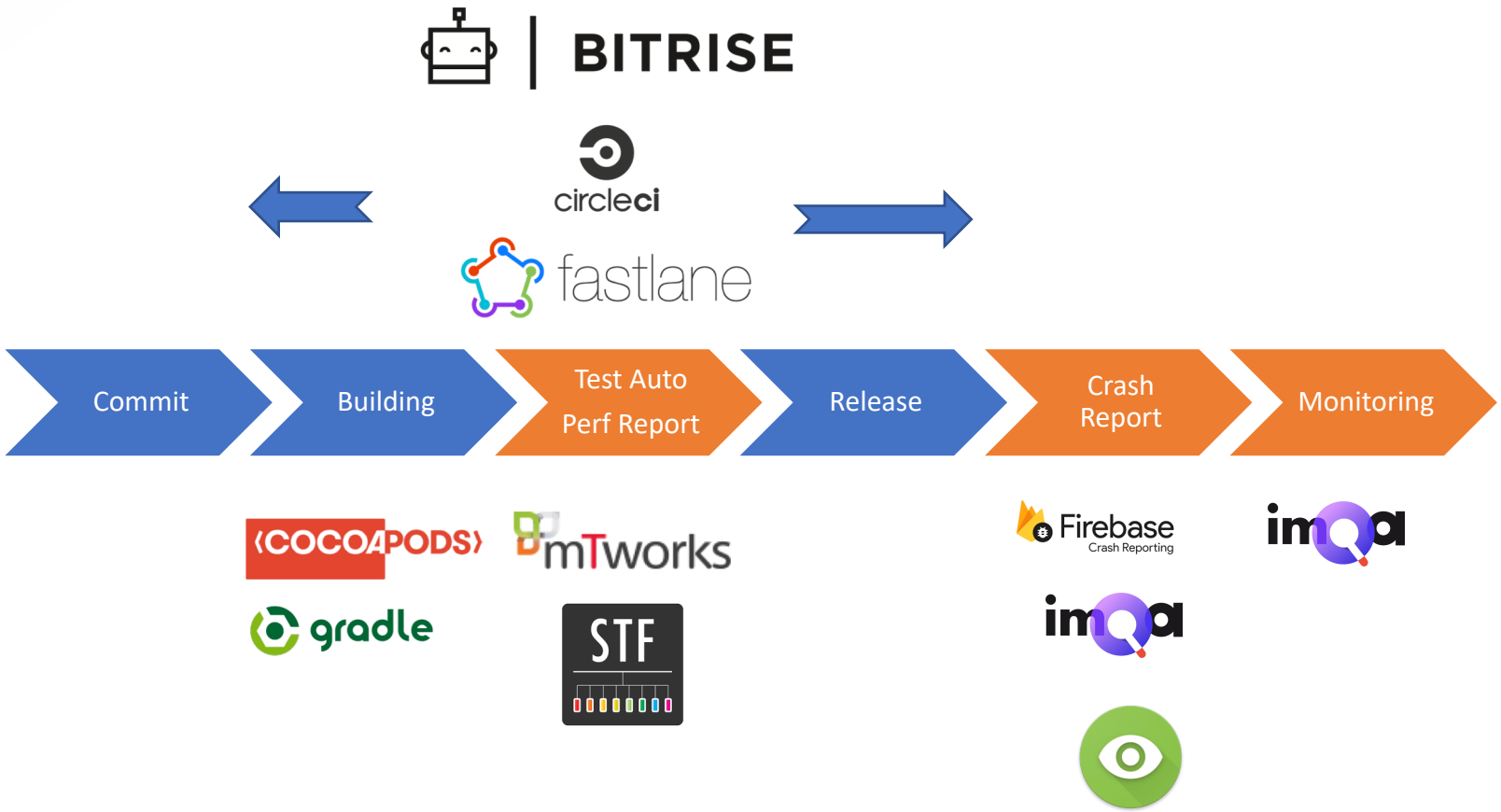
- Android. : Byte code Instrumentation 기법
- iOS : Swizzling (https://academy.realm.io/kr/posts/sash-zats-swift-swizzling/) + @가 많음



Android God Eye - https://github.com/Kyson/AndroidGodEye



모바일 DevOps 정리



감사합니다.

Thank you for your interesting.



IMQA 제품에 대한 자세한 문의는 아래로 주세요

손영수 상무 (총괄)
ysson@onycom.com

강성민 차장 (영업)
seankang@onycom.com

유영수 주임 (기술)
youngsoo@onycom.com