

# JBoss Troubleshooting Essential

장애 처리와 튜닝 프로세스(Thread Dump 분석)

## Application Performance Management

# 트러블 슈팅 절차 및 방법

**KHAN**  
[ a p m ]

# 시스템 구축 후 - 고민들



- ✓ 오늘 온라인방문자 수가 급격히 줄었는데요?
- ✓ 다음달에 온라인 광고를 해야 하는데 문제는 없겠죠?
- ✓ 주기적으로 장애가 발생하는데 대책은 없나요?
- ✓ 사용자가 조금 만 늘어도 시스템이 다운이니 불안해서 뭘 할 수가 없네요.

- ✓ 확인해 봤는데 소스는 문제가 없어요. WAS 문제 아닌가요?
- ✓ 느린 페이지에서 사용하는 DB 쿼리를 보고 싶으세요.
- ✓ 개발할 땐 빠르는데 운영시스템에서만 느린 이유가 무엇인가요?



- ✓ 서비스가 갑자기 느려졌는데 WAS 문제일까요?
- ✓ 실시간 동시 사용자수나 TPS 등의 기본 운영 정보를 알고 싶으세요.
- ✓ 서비스가 느려질 때 WAS를 Restart 하면 정상적으로 동작해요.
- ✓ 특정페이지 나 서비스가 느린 이유는 무엇일까요?



0:28 / 0:28



<https://www.youtube.com/watch?v=|YjGoNEV-E&index=12&list=PLaFP0KYzLL-QWKsbT40cxwr-cijU6L>

# 고객이 체험하는 것은?



KHAN [ a p m ]



<https://www.youtube.com/watch?v=jMUpM-1hx7Q&index=6&list=PLaFPOkYzLL-QWKsbT40cxwr-cjeU6L>

# 엔터프라이즈 모니터링에서는 문제 없음

네트워크, 서버 모두 잘 운영되고 있습니다.  
심지어 어제보다도 성능이 좋은데요.



0:23 / 0:25

<https://www.youtube.com/watch?v=eE7rPuksd1c&index=7&list=PLaFP0kYzLL-QWksbT40cxwr-cijeU6L>

# 각 분야에서는 전문가 – 코끼리 만지기

KHAN [ a p m ]



0:48 / 0:51



[https://www.youtube.com/watch?v=P5S0gl-4hZl&list=PLaFPOkYzLL-\\_QWKsbT40cxwr\\_-cijeU6L&index=10](https://www.youtube.com/watch?v=P5S0gl-4hZl&list=PLaFPOkYzLL-_QWKsbT40cxwr_-cijeU6L&index=10)

## Application Performance Management

# Java Trouble Shooting 프로세스

**KHAN**  
[ a p m ]

# Troubleshooting 공통 프로세스



장애발생

상황수집  
(긴급조치,로그수집)

진단&모니터링

원인분석&재현

조치

복구



증상과 원인



진단 및 모니터링



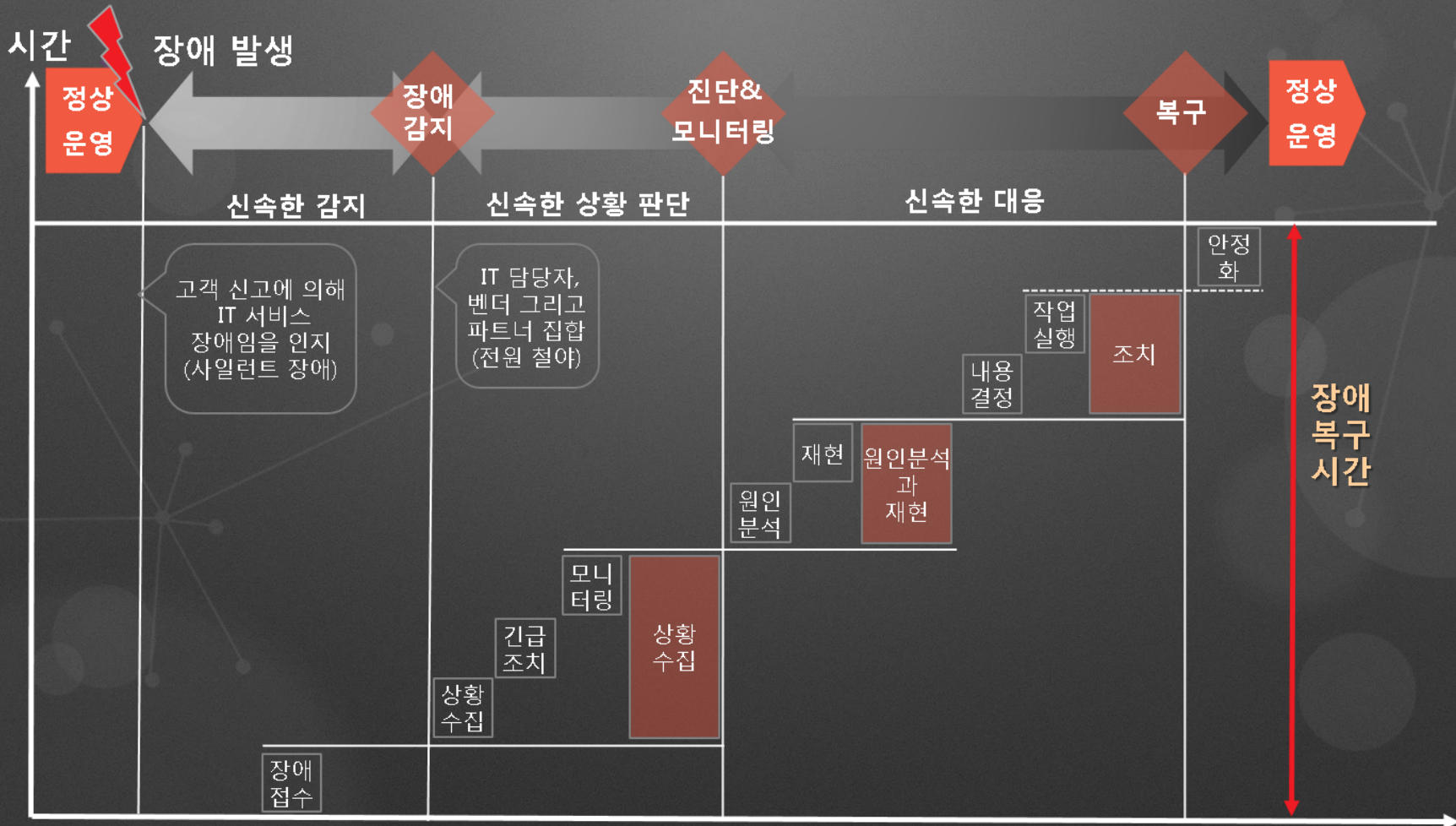
조치



방지대책

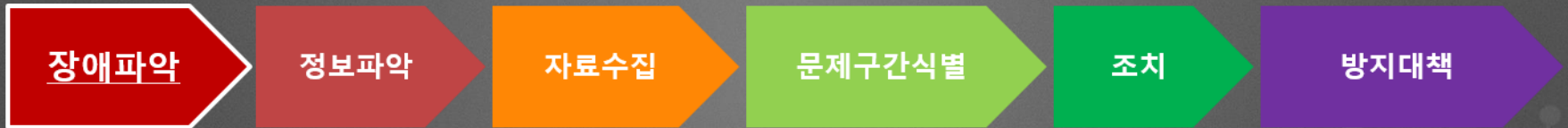
# Incident & Problem Management

- 장애복구시간을 단축할 수 있는 도구는 ?

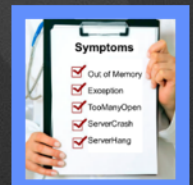


장애 복구 과정 ( Incident Lifecycle Management)

# 증상과 원인 - 장애 내용 파악

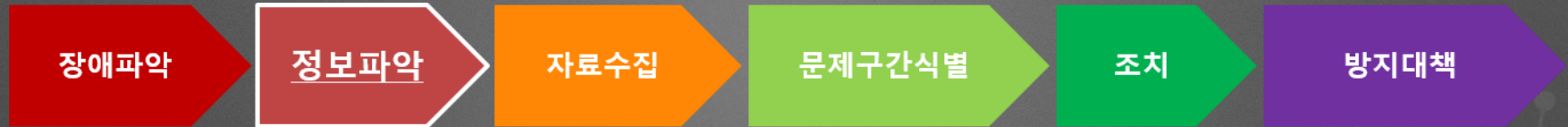


- 장애 내용 및 상황 확인
  - 장애 발생 과정
  - 시스템 정지 여부
  - 장애 발생 빈도
  - 장애 발생 조건
  - 장애 재현 가능 여부
  - 장애 재현 절차
  - 회피 방법 유무



증상과 원인

# 증상과 원인 - 현황 파악

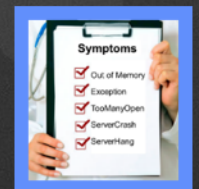


- 기본 정보 파악

- JBoss EAP 버전
- OS 종류/버전
- JVM 종류/ 버전
- 데이터베이스 종류/ 버전
- etc...

- 설정 파일 파악

- JBoss 설정 파일(domain.xml, host.xml, standalone.xml 등 )
- JBoss 시작 스크립트
- etc...

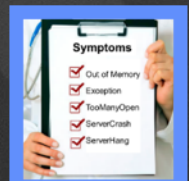


증상과 원인

# 증상과 원인 - 수집 자료는?

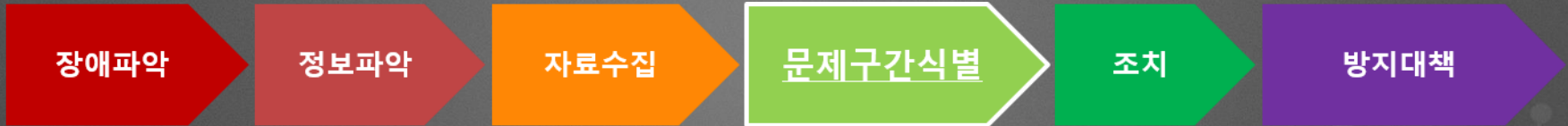


- **기본 수집 자료**
  - 로그!
  - Thread dump!
  - 기본 환경 설정 정보(버전, 설정, 스크립트)
- **관련된 상황은?**
  - 언제
  - 어떤 서버, 어느 인스턴스에서 정확한 OS, JVM, WAS 버전은?
  - 발생 주기는?
  - 재현은 되는지?
- **어떤 영향을 미치는지?**
  - Business 부서의 영향은?
- **관련된 네트워크, H/W 및 S/W는 어떤 것인지?**
  - 대상 시스템에 로그나 상황은 어떤지 체크



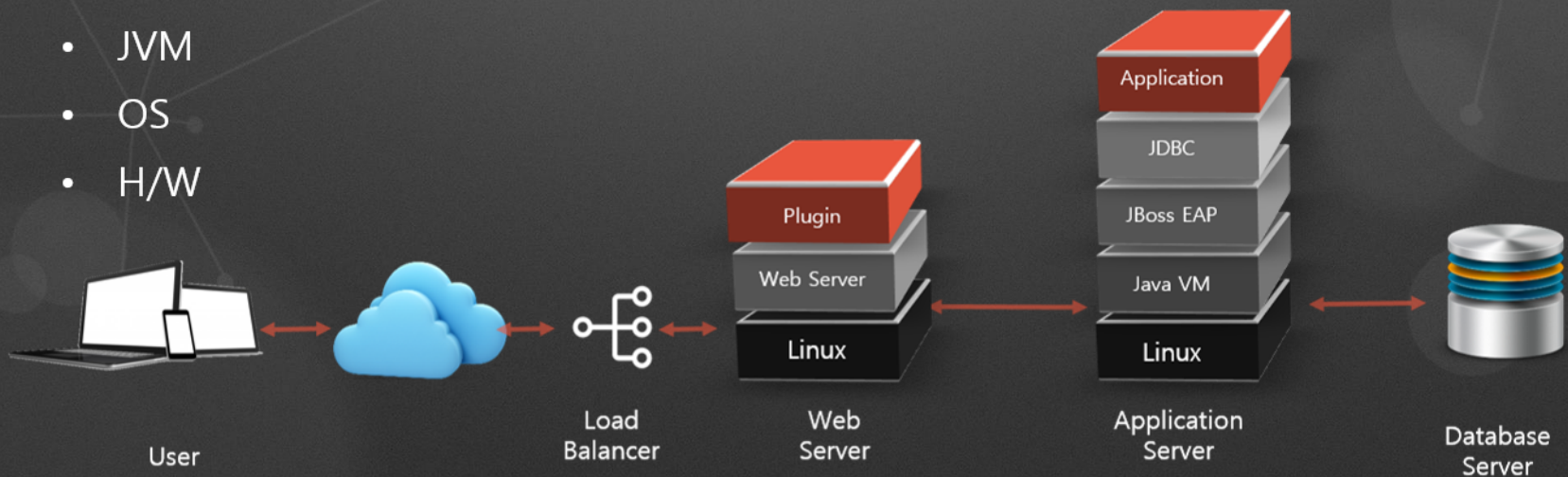
증상과 원인

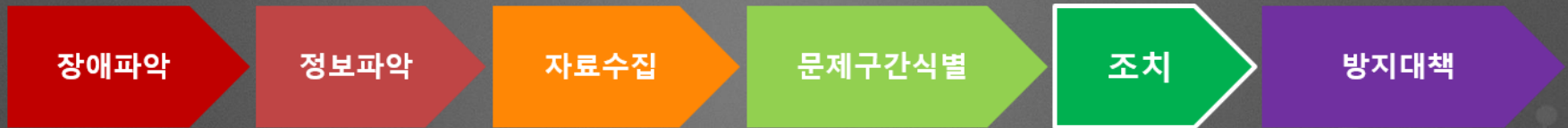
# 진단 및 모니터링 - 정보수집



- 문제가 발생 구간 파악 및 간단한 샘플 재현

- Application
- JBoss
- Database
- Web Server
- JVM
- OS
- H/W





- **조치 내용**

- 설정 변경
- 어플리케이션 수정
- 버전업
- Patch
- etc...

- **조사 방법**

- 재현 스텝
- 디버그 정보



조치

# 조치 - Patch

장애파악

정보파악

자료수집

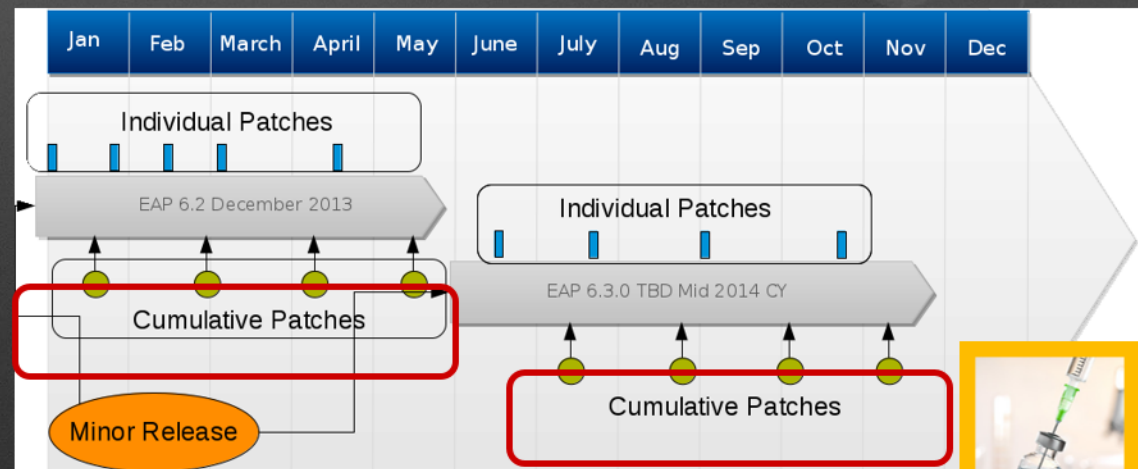
문제구간식별

조치

방지대책

- 패치 업데이트는 ZIP 및 EAP의 설치 기반의 RPM 형식으로 전달되며, zip 형태의 패치 파일은 JBoss CLI 의 Patch 명령어를 통해 적용
- 적은 개별 패치들은 관리자로 하여금 관리 부담 감소
- 심각도가 높고 시간이 급한 케이스에 대한 버그 픽스는 기존 방법대로 개별 패치들로 배포

- 여러 수정사항을 포함한 누적패치는 함께 테스트되어 제품의 질을 향상시키고, 위험성을 감소

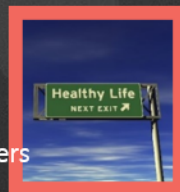
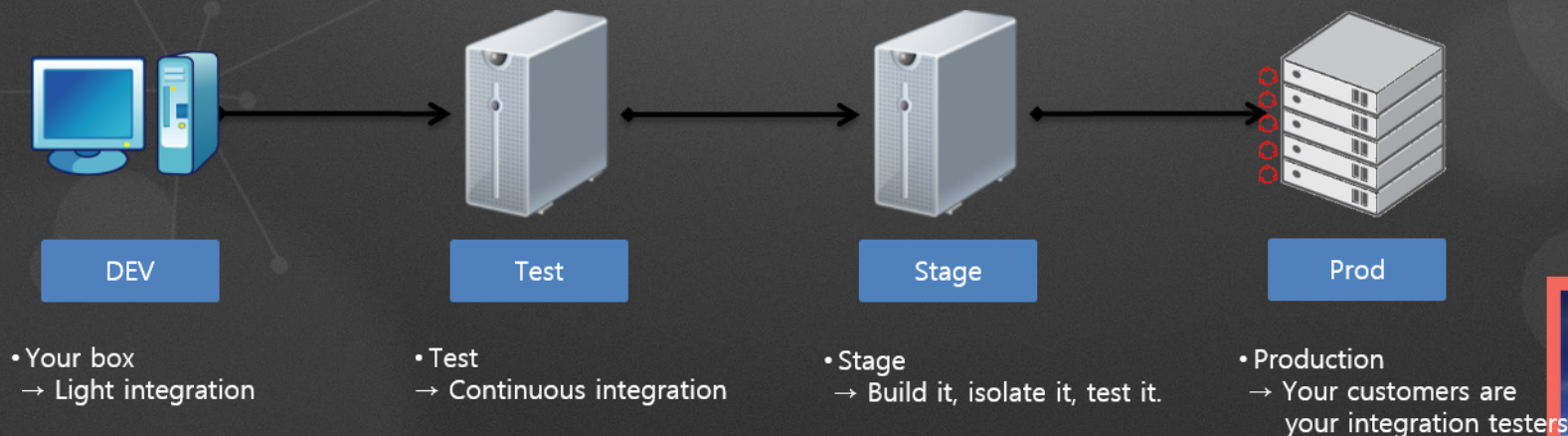


Source : EAP 6.2 이상부터 유지관리 배포방식 변경  
<https://access.redhat.com/site/ko/articles/645133>

조치



- 안정된 코드 테스트를 애플리케이션과 검증을 위한 환경 구축
- 서비스 품질을 높이기 위한 기본 프레임워크
- Subversion (SVN) 또는 GitHub
- 빈번한 커밋과 자동화된 빌드와 자동 피드백
- 테스트 프레임워크 적용



방지대책

# 트러블 슈팅의 기본

- **로그 등 관련 자료는 충분히 확보해야**
  - 로그는 첫 번째 문제부터 살펴봐야
- **선부른 추측은 금물**
  - 전해들은 말을 100% 믿지 말라
  - 증거에 의존하여 분석
  - 문서를 100% 믿지 말라
- **문제분석**
  - 문제를 분리하는 방법(Divide & Conquer)
  - 코드 샘플링을 통한 분리 / 재현
  - 재현 & 테스트 & 재현 & 테스트



Application Performance Management

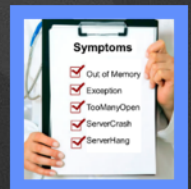
# 서버가 느려질 때 분석방법

## 스레드 덤프 분석

KHAN  
a p m

# Server Hang

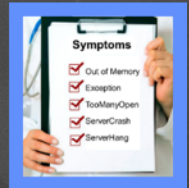
- **현상**
  - 운영 중인 서버의 애플리케이션에서 새로운 요청을 받지 못함
  - Request가 타임 아웃되며, 응답시간이 점점 느려짐.
- **주된 원인**
  - 애플리케이션의 설계나 구현의 버그
  - 잘못된 용량 산정(Capacity Planning)
  - 모든 스레드가 처리 중이어서, 새로운 요청을 받지 못하는 상태
    - 처리시간이 오래 걸리는 작업의 수행
    - 데이터베이스와 같은 외부 자원의 응답 대기
    - 자원의 경합
  - **Full GC 시간이 오래 걸림**
    - 튜닝 부족이나 적합치 않은 GC 알고리즘 선택
  - **JVM이나 미들웨어의 버그**



증상과 원인

# Server Hang이란?

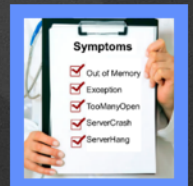
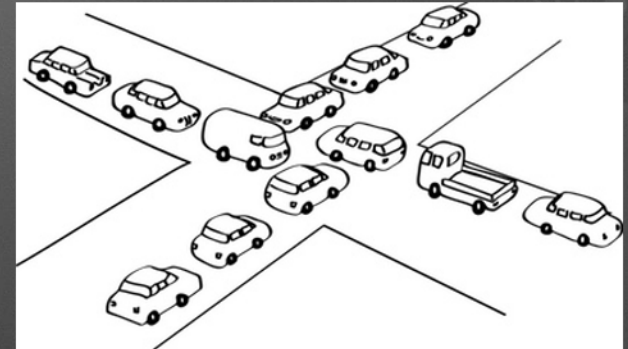
- 서버가 전혀 Response를 주지 못함 → 서버 Hang
- 느리더라도 Response를 준다 → 서버 Slowdown
- 서버 Hang의 증상
  - Request가 처리되지 않는다
  - 새로운 Request를 받지 못하고 기존 Request가 Timeout되거나 응답이 없는 상황
  - 서버가 아무 작업도 하지 않는 것 같다
  - 서버의 증상:
    - Hang이 오래 지속되면 JVM Crash될 수도 있다(항상은 아님)
    - 그냥 두면 Hang이 풀리는 경우도 있다(리소스 경합이 발생하여 Hang이 발생한 경우에는 Resource가 풀리면 Hang이 풀린다)
    - 대부분 Hang이 발생한 후에는 관리자의 조치가 없으면 Hang 상태로 무한이 남아있게 된다



증상과 원인

# Server Hang의 원인

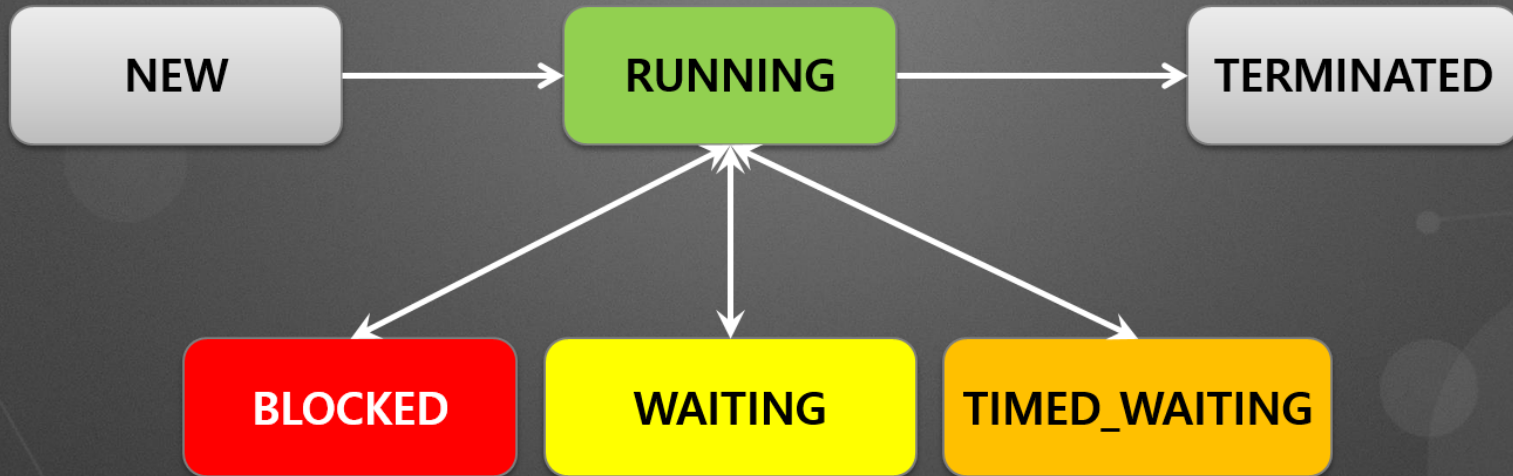
- 리소스 부족
  - 스레드나 메모리 부족 → CPU 과점유나 OutOfMemory 에러발생
  - 파일 핸들이 부족한 경우 → Too many Open Files 에러가 발생
  - 리소스 경합이나 DeadLock이 발생하는 경우
    - JDBC Dead Lock이 발생하는 경우
    - JNDI Lookup에서 경합이 발생하는 경우
    - Application Dead Lock
  - JSP 컴파일 주기가 짧아서 발생하는 Hang
- Hang 처럼 보이는 경우
  - 리소스 경합의 경우 리소스가 가용해지면 조금씩 풀리는 경우
  - Full GC 시간이 오래 걸리는 경우
  - JVM에서 Code Optimization시 Hang이 걸리는 경우



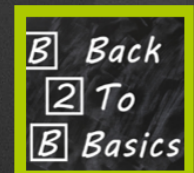
증상과 원인

# Thread 상태

- `java.lang.Thread.State`

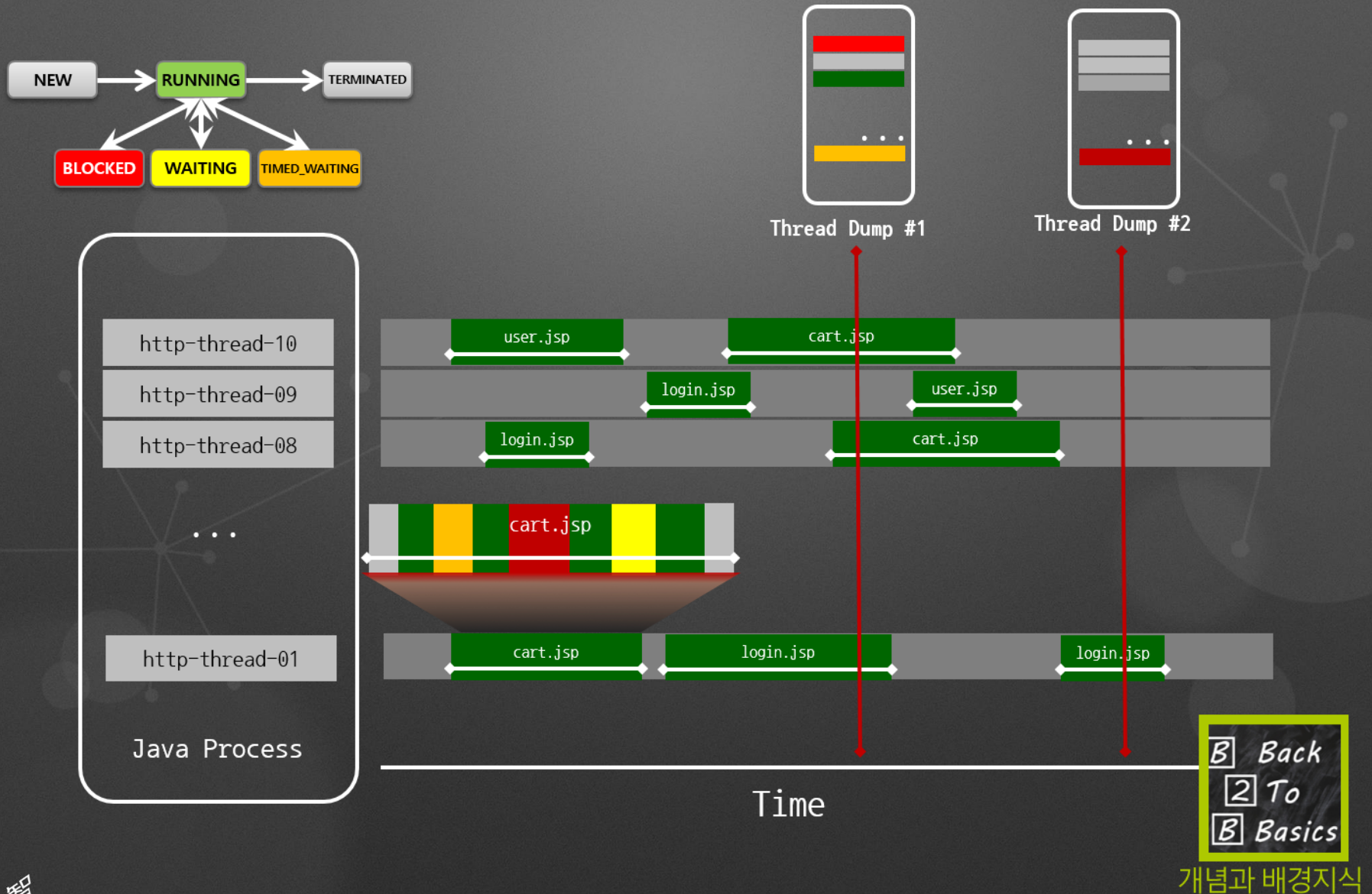


- NEW – 생성되어있지만 실행되지 않은 상태
- RUNNING – 작업 수행 중
- BLOCKED – 다른 스레드의 Lock 해제를 기다리는 중
- WAITING – `wait()`, `join()`, `park()` 메소드로 대기 중
- TIMED\_WAITING – WAITING과 같지만, 정해진 시간만 대기
- TERMINATED – 종료됨



개념과 배경지식

# Java Thread Dump의 이해



Back  
To  
Basics

개념과 배경지식

# 스레드 덤프의 정보 구성

- 스레드 덤프의 각 스레드의 정보 구성



스레드 이름

우선순위

스레드ID

OS 스레드ID

스레드 상태

```

"http-thread-pool-threads - 250" prio=0 tid=0x00007fab9c192800 nid=0xdbc9 waiting on condition [0x00007faa6ce2c000]
java.lang.Thread.State: WAITING (parking)
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <0x0000000078b428170> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:186)
    at java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2043)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1068)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1130)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
    at java.lang.Thread.run(Thread.java:744)
    at org.jboss.threads.JBossThread.run(JBossThread.java:122)

```

스레드 호출 Stack

[B Back](#)  
[2 To](#)  
[B Basics](#)

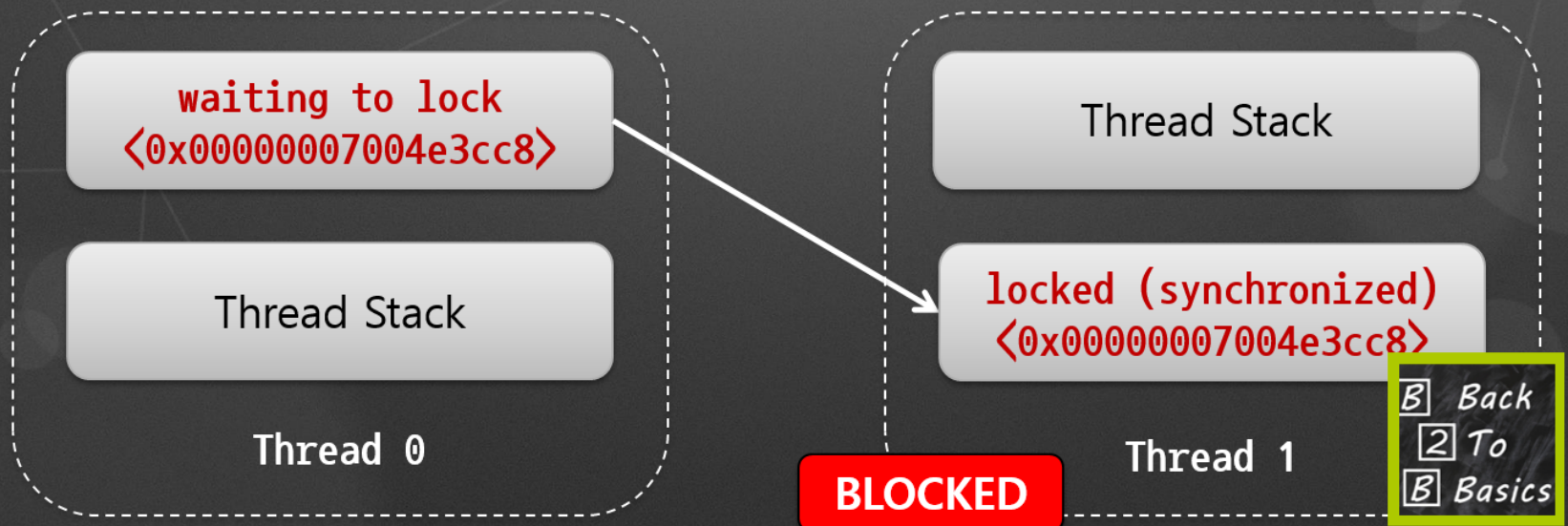
개념과 배경지식

# Thread Dump의 Blocked의 의미

- 여러 스레드에서 같은 변수나 메소드를 변경할 경우
- **synchronized**로 다른 스레드의 처리가 끝날 때 까지 제어가 넘어가지 않도록 보호

```
// 특정한 객체에 lock을 걸고자 할 때
synchronized(객체의 참조변수){
    // ...
}

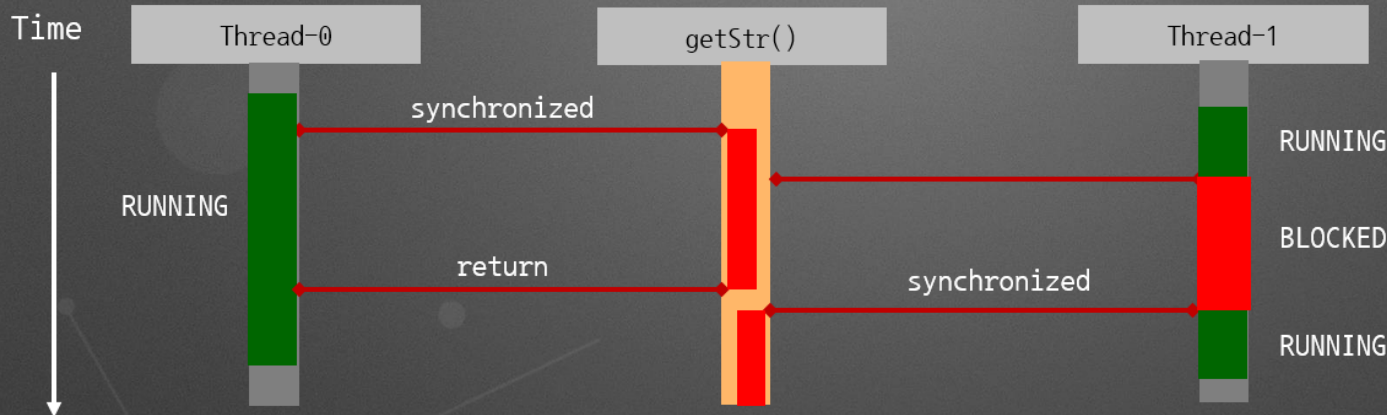
// 메서드에 lock을 걸고자할 때
public synchronized void methodName(){
    // ...
}
```



개념과 배경지식

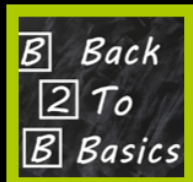
# Java Thread Dump의 BLOCKED의 의미

- waiting for monitor entry
  - 오브젝트에 대한 락의 해제를 기다리고 있는 상태



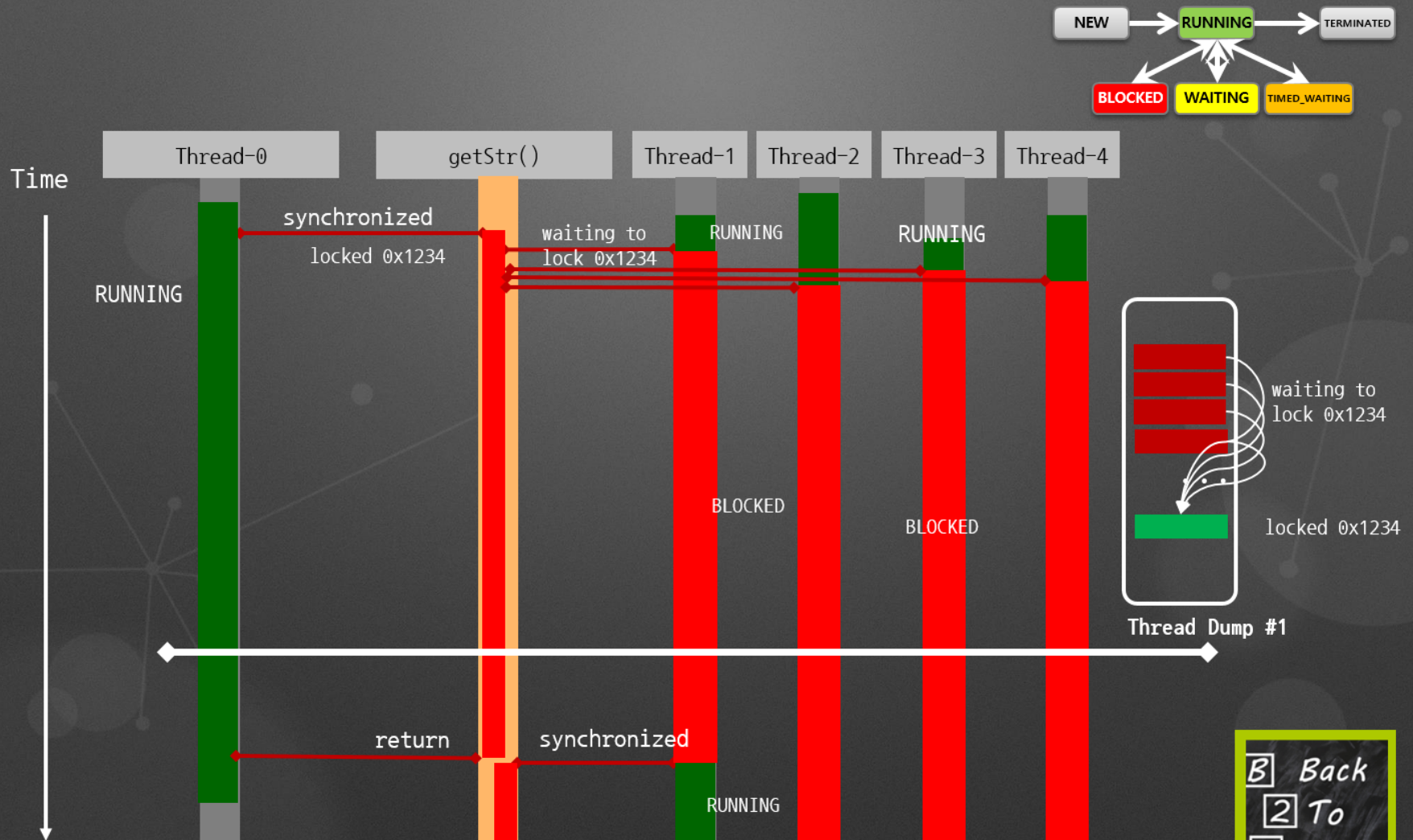
```

"Thread-1" prio=10 tid=0x00007f69fc0b8800 nid=0x7358 waiting for monitor entry [0x00007f6a0050a000]
  java.lang.Thread.State: BLOCKED (on object monitor)
    at sample.SampleMain.getStr(SampleMain.java:23)
      - waiting to lock <0x000000077acacf60> (a java.lang.Class for sample.SampleMain)
    at sample.SampleMain.run(SampleMain.java:17)
    at java.lang.Thread.run(Thread.java:679)
"Thread-0" prio=10 tid=0x00007f69fc0b6800 nid=0x7357 waiting on condition [0x00007f6a0060b000]
  java.lang.Thread.State: TIMED_WAITING (sleeping)
    at java.lang.Thread.sleep(Native Method)
    at sample.SampleMain.getStr(SampleMain.java:23)
      - locked <0x000000077acacf60> (a java.lang.Class for sample.SampleMain)
    at sample.SampleMain.run(SampleMain.java:17)
    at java.lang.Thread.run(Thread.java:679)
  
```



개념과 배경지식

# Thread BLOCKED 상태가 길어지면 발생하는 상황



Back  
2 To  
Basics

개념과 배경지식



# Thread Dump 분석 방법

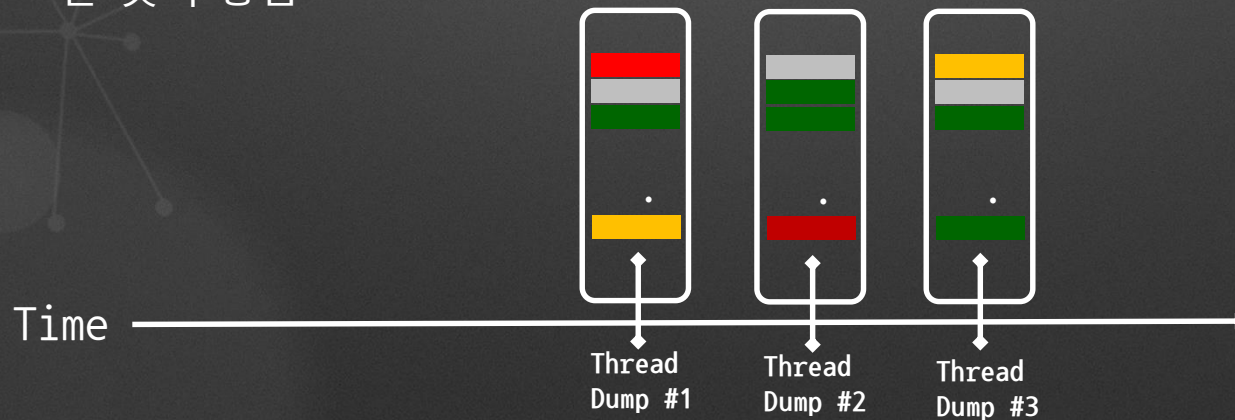
# 스레드 덤프

## • 스레드 덤프란?

- JVM 내부에서 현재 수행중인 스레드의 SnapShot을 텍스트로 출력
- 서버 Hang, 서버 Slowdown등 서버에서 발생하는 문제를 해결하는 가장 유용한 도구
- 약 3~5초간격으로 3~5회 연속해서 스레드 덤프를 받아야 분석이 가능하다

## • 언제 받아야 하나?

- 운영 중인 서버에서도 수행가능(많은 시간이 걸리지 않는 작업임)
- 서버 Hang, Slowdown시, 서버가 Crash되거나 OutOfMemory가 발생한 직후에 받는 것이 좋음



**B** Back  
**2** To  
**B** Basics

개념과 배경지식

# 스레드 덤프를 분석

Take at least 3 times, with 3 to 5 seconds interval



진단 및 모니터링

Thread 1  
Thread 2  
Thread 3  
Thread 4  
Thread 5  
Thread 6  
Thread 7  
Thread 8  
Thread 9

The image displays three sequential screenshots of a thread dump analysis. Each screenshot shows a terminal window with the title 'yusuke - bash - 216x'. The terminal output lists threads 1 through 9, with each thread's state and stack trace highlighted by a red box. The threads are listed in descending order of ID. The stack traces show the call stack for each thread, including the main method and various system calls. The three screenshots likely represent different points in time, showing the progression of the thread dump analysis.

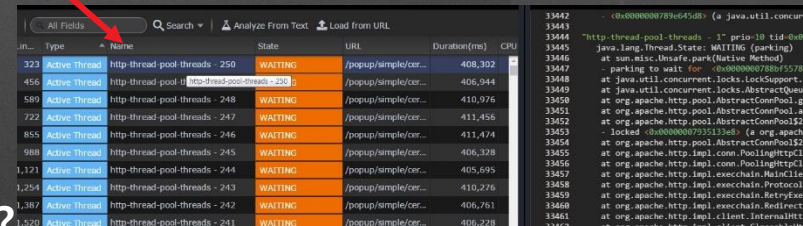
## 스레드 덤프 받는 방법은?

- Linux / UNIX 시스템에서 스레드 덤프
  - kill -3 <PID>
  - 일반적을 nohup 명령을 이용하여 시작하기 때문에 nohup.out파일에 stdout 로그가 남게 됨
  - 윈도우에서는 Ctrl + Break
  - KHAN [apm] 화면에서 버튼 클릭(장애시 자동 생성)



진단 및 모니터링

- Docker와 Openshift 같은 환경
  - 스레드 덤프를 어떻게 받나?

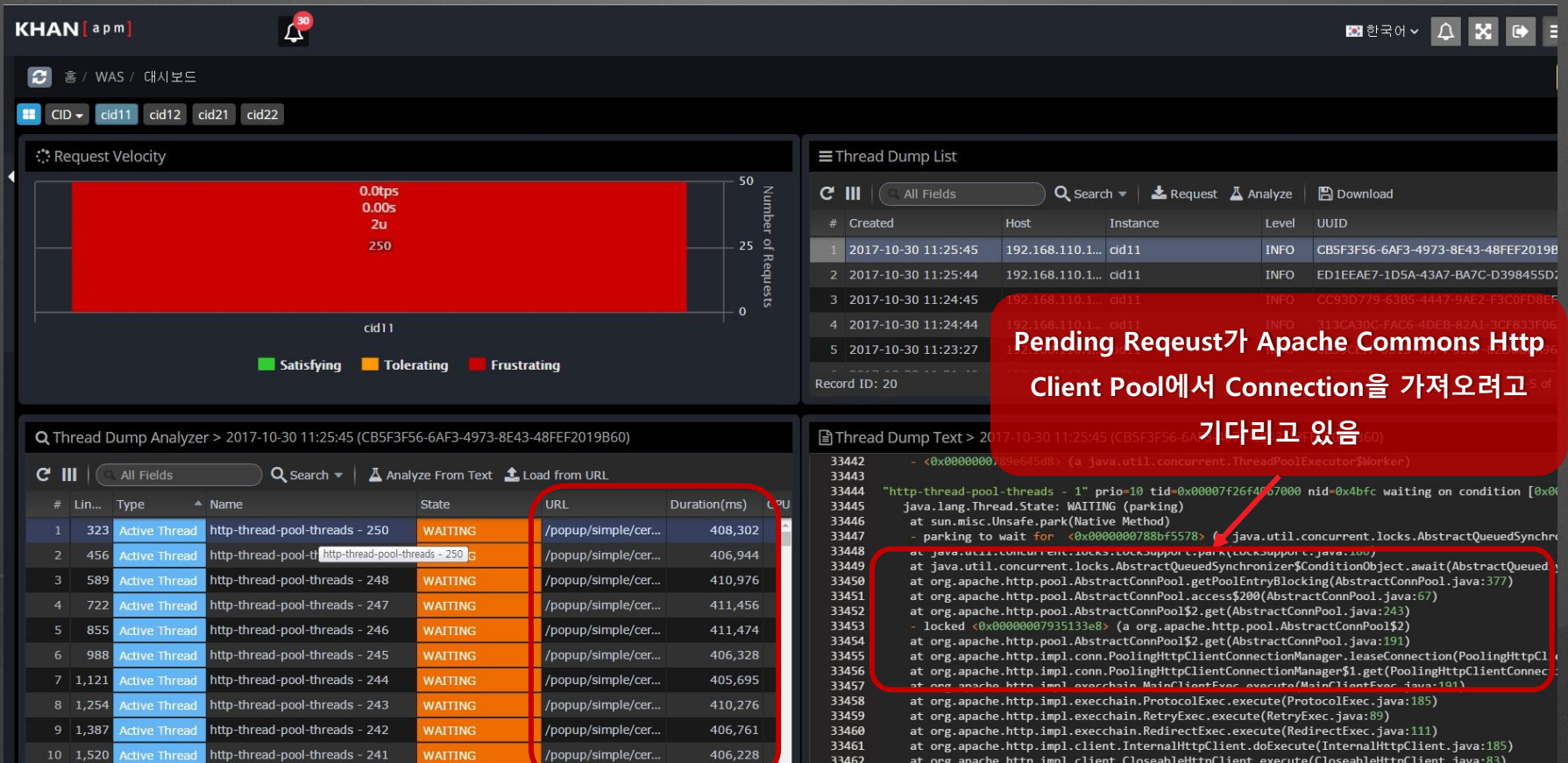
| in... | Type          | Name                           | State   | URL                  | Duration(ms) | CPU |
|-------|---------------|--------------------------------|---------|----------------------|--------------|-----|
| 323   | Active Thread | http-thread-pool-threads - 250 | WAITING | /popup/sample/cor... | 408,302      |     |
| 456   | Active Thread | http-thread-pool-threads - 250 | WAITING | /popup/sample/cor... | 406,944      |     |
| 569   | Active Thread | http-thread-pool-threads - 248 | WAITING | /popup/sample/cor... | 410,976      |     |
| 722   | Active Thread | http-thread-pool-threads - 247 | WAITING | /popup/sample/cor... | 411,456      |     |
| 855   | Active Thread | http-thread-pool-threads - 246 | WAITING | /popup/sample/cor... | 411,474      |     |
| 988   | Active Thread | http-thread-pool-threads - 245 | WAITING | /popup/sample/cor... | 406,328      |     |
| 1,121 | Active Thread | http-thread-pool-threads - 244 | WAITING | /popup/sample/cor... | 405,695      |     |
| 1,254 | Active Thread | http-thread-pool-threads - 243 | WAITING | /popup/sample/cor... | 410,276      |     |
| 1,387 | Active Thread | http-thread-pool-threads - 242 | WAITING | /popup/sample/cor... | 406,761      |     |
| 1,520 | Active Thread | http-thread-pool-threads - 241 | WAITING | /popup/sample/cor... | 406,228      |     |

- docker exec -it c4343003c1 /bin/bash  
~# kill -3 <PID>
- 내용을 어떻게 복사? nohup 로그는 남기나?
  - docker cp c4343003c1:/file/path/nohup.out /host/path/target
- KHAN [apm]은 Openshift, Docker를 지원 → 버튼 클릭 & 분석 UI 제공

# Server가 느려 지는 상황에서 Dashboard



# KHAN [apm] 스레드 덤프 분석 예시



Pending Request가 Apache Commons Http Client Pool에서 Connection을 가져오려고 기다리고 있음

스레드에서 어떤 URL이 얼마동안, CPU 자원을 사용하며 실행중인지 표시함



|    |       |             |                  |         |                      |        |       |               |          |
|----|-------|-------------|------------------|---------|----------------------|--------|-------|---------------|----------|
| 2  | 924   | Active Thre | default task-238 | BLOCKED | /cnsl/front/exame... | 33,986 | 693.4 | org.apache... | 0x000... |
| 3  | 1,061 | Active Thre | default task-236 | BLOCKED | /cnsl/front/exame... | 33,058 | 628.9 | org.apache... | 0x000... |
| 4  | 1,181 | Active Thre | default task-235 | BLOCKED | /cnsl/front/exame... | 39,877 | 745.3 | org.apache... | 0x000... |
| 5  | 1,588 | Active Thre | default task-215 | BLOCKED | /cnsl/front/exame... | 38,246 | 745.2 | org.apache... | 0x000... |
| 6  | 1,737 | Active Thre | default task-212 | BLOCKED | /cnsl/front/exame... | 32,150 | 637.7 | org.apache... | 0x000... |
| 7  | 1,983 | Active Thre | default task-204 | BLOCKED | /cnsl/front/exame... | 30,869 | 627.5 | org.apache... | 0x000... |
| 8  | 2,117 | Active Thre | default task-202 | BLOCKED | /cnsl/front/exame... | 38,711 | 757.2 | org.apache... | 0x000... |
| 9  | 2,314 | Active Thre | default task-196 | BLOCKED | /cnsl/front/exame... | 30,347 | 608.5 | org.apache... | 0x000... |
| 10 | 2,498 | Active Thre | default task-193 | BLOCKED | /cnsl/front/exame... | 38,861 | 757.6 | org.apache... | 0x000... |
| 11 | 2,626 | Active Thre | default task-191 | BLOCKED | /cnsl/front/exame... | 38,892 | 757.6 | org.apache... | 0x000... |
| 12 | 2,717 | Active Thre | default task-189 | BLOCKED | /cnsl/front/exame... | 38,892 | 757.6 | org.apache... | 0x000... |
| 13 | 3,117 | Active Thre | default task-187 | BLOCKED | /cnsl/front/exame... | 38,892 | 757.6 | org.apache... | 0x000... |
| 14 | 3,261 | Active Thre | default task-174 | BLOCKED | /cnsl/front/exame... | 33,674 | 599.3 | org.apache... | 0x000... |
| 15 | 3,419 | Active Thre | default task-172 | BLOCKED | /cnsl/front/exame... | 33,674 | 599.3 | org.apache... | 0x000... |
| 16 | 3,533 | Active Thre | default task-168 | BLOCKED | /cnsl/front/exame... | 33,674 | 599.3 | org.apache... | 0x000... |
| 17 | 4,007 | Active Thre | default task-165 | BLOCKED | /cnsl/front/exame... | 29,606 | 589.9 | org.apache... | 0x000... |
| 18 | 4,144 | Active Thre | default task-164 | BLOCKED | /cnsl/front/exame... | 29,606 | 589.9 | org.apache... | 0x000... |
| 19 | 4,278 | Active Thre | default task-163 | BLOCKED | /cnsl/front/exame... | 31,212 | 564.5 | org.apache... | 0x000... |
| 20 | 4,400 | Active Thre | default task-146 | BLOCKED | /cnsl/front/memb...  | 1,145  | 11.3  | org.apache... | 0x000... |
| 21 | 4,577 | Active Thre | default task-143 | BLOCKED | /cnsl/front/exame... | 37,405 | 694.4 | org.apache... | 0x000... |
| 22 | 5,179 | Active Thre | default task-110 | BLOCKED | /cnsl/front/exame... | 28,692 | 618.0 | org.apache... | 0x000... |
| 23 | 5,302 | Active Thre | default task-109 | BLOCKED | /cnsl/front/exame... | 12,069 | 282.6 | org.apache... | 0x000... |
| 24 | 5,589 | Active Thre | default task-98  | BLOCKED | /cnsl/front/exame... | 31,622 | 630.8 | org.apache... | 0x000... |
| 25 | 5,738 | Active Thre | default task-95  | BLOCKED | /cnsl/front/exame... | 38,526 | 743.0 | org.apache... | 0x000... |
| 26 | 5,917 | Active Thre | default task-90  | BLOCKED | /cnsl/front/exame... | 28,350 | 612.5 | org.apache... | 0x000... |
| 27 | 6,265 | Active Thre | default task-74  | BLOCKED | /cnsl/front/exame... | 37,662 | 743.6 | org.apache... | 0x000... |
| 28 | 6,496 | Active Thre | default task-66  | BLOCKED | /cnsl/front/exame... | 27,470 | 557.1 | org.apache... | 0x000... |

```

2769 at org.jboss.as.logging.stdio.log.ContextStdioContextSelector.getSelectedContextSelector(java.lang.String)
2770 at org.jboss.studio.StdioContext$1.getDelegate(StdioContext.java:148)
2771 at org.jboss.studio.StdioContext$ DelegatingPrintStream.write(StdioContext.java:264)
2772 at sun.nio.cs.StreamEncoderImpl.write(StreamEncoder.java:223)
2773 at sun.nio.cs.StreamEncoderImpl.flushBuffer(StreamEncoder.java:291)
2774 at sun.nio.cs.StreamEncoderImpl.flush(StreamEncoder.java:295)
2775 at sun.nio.cs.StreamEncoder.flush(StreamEncoder.java:143)
2776 at java.io.OutputStreamWriter.close(OutputStreamWriter.java:229)
2777 at java.io.OutputStreamWriter.close()
2778 at org.apache.log4j.AppendSkeleton.doAppend()
2779 - locked <0x00000006cf0ca580> (JvarkitImpl.java:67)
2780 at org.apache.log4j.helpers.AppenderAttachah
2781 at org.apache.log4j.Category.log(Category.java:1177)
2782 at org.apache.commons.logging.impl.Log4jLogger.debug(Log4jLogger.java:377)
2783 at org.springframework.data.jpa.repository.support.JpaRepositoryManager.delegate(DataSourceTransactionManager.java:221)
2784 at org.springframework.transaction.support.AbstractPlatformTransactionManager.getTransaction(AbstractPlatformTransactionManager.java:372)
2785 at org.springframework.transaction.interceptor.TransactionAspectSupport.createTransactionIfNecessary(TransactionAspectSupport.java:417)
2786 at org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:255)
2787 at org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.java:84)
2788 at org.springframework.reflect.methodinvocation.proceed(ReflectiveMethodInvocation.java:172)
2789 at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172)
2790 at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172)
2791 at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172)
2792 at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:172)
2793 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2794 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2795 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2796 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2797 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2798 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2799 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2800 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2801 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2802 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2803 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2804 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2805 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2806 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2807 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2808 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2809 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2810 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2811 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2812 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2813 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2814 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2815 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2816 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2817 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2818 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2819 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2820 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2821 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2822 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2823 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2824 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)
2825 at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invoke(CglibAopProxy.java:633)

```

## Opennaru Service Desk

고객께서 Quick Service를 통해  
전달해 주신  
화면과 Thread Dump 및  
메타 정보를 접수합니다.  
자동으로 케이스를 생성합니다.

# Opennaru Quick Service 란?



## 고객사

장애 분석  
보고서 전달

1 시스템  
이슈발생

2 스레드 덤프와  
모니터링 화면 전달



3 오픈나루  
서비스 데스크 접수

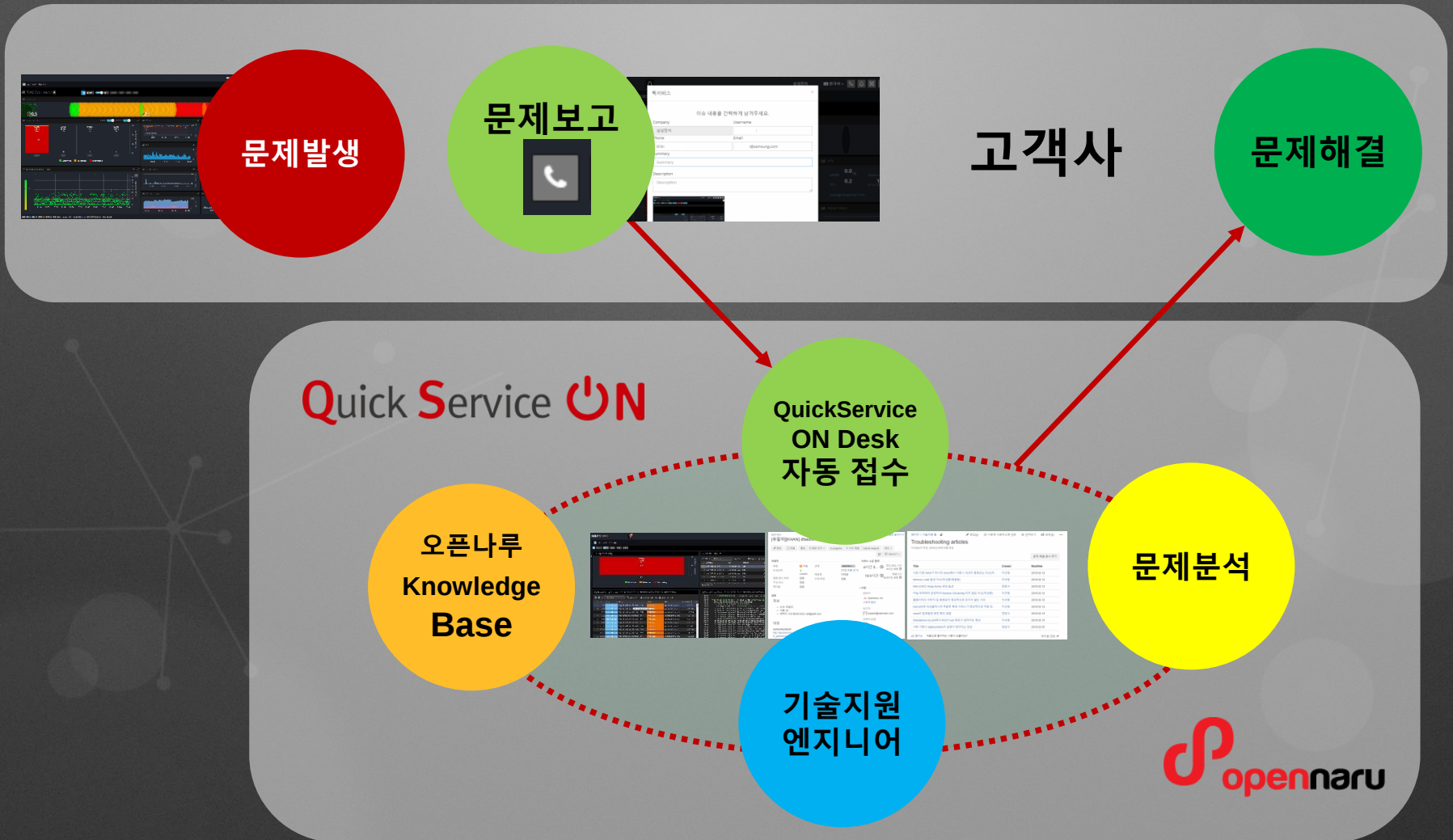
4 장애 분석



5

오픈나루

# Quick Service ON



# KHAN [apm] = Ah ! Provisioning + Monitoring



| 설치                                                                                               | 구성                                                                                                                              | 튜닝                                                                                   | 모니터링                                                                                             | 진단/조치                                                                                          |
|--------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>•리눅스만 설치되어 있으면 WAS 와 웹서버 등 웹시스템에 필요한 소프트웨어를 자동설치</li></ul> | <ul style="list-style-type: none"><li>•WAS 클러스터링, 데이터그리드, 필수 운영스크립트 등 난이도 높은 구성을 지원</li><li>•개발자/관리자를 위한 운영 가이드 자동 생성</li></ul> | <ul style="list-style-type: none"><li>•전문가 수준의 튜닝 (리눅스 커널, 웹서버/JVM/WAS 튜닝)</li></ul> | <ul style="list-style-type: none"><li>•시스템과 OS 에 대한 모니터링 뿐만 아니라 Java 나 WAS 까지 포괄적으로 지원</li></ul> | <ul style="list-style-type: none"><li>•Threaddump, 네트워크 상태, 웹서버 상태 등을 진단할 수 있는 도구 제공</li></ul> |

“살아 남는 종(種)은 강한 종이 아니고,  
또 우수한 종도 아니다.  
변화에 적응하는 종이다.”

- Charles Darwin, 1809



감사합니다.



제품이나 서비스에 관한 문의

콜 센터 : 02-469-5426 ( 휴대폰 : 010-2243-3394 )

전자 메일 : [sales@opennaru.com](mailto:sales@opennaru.com)