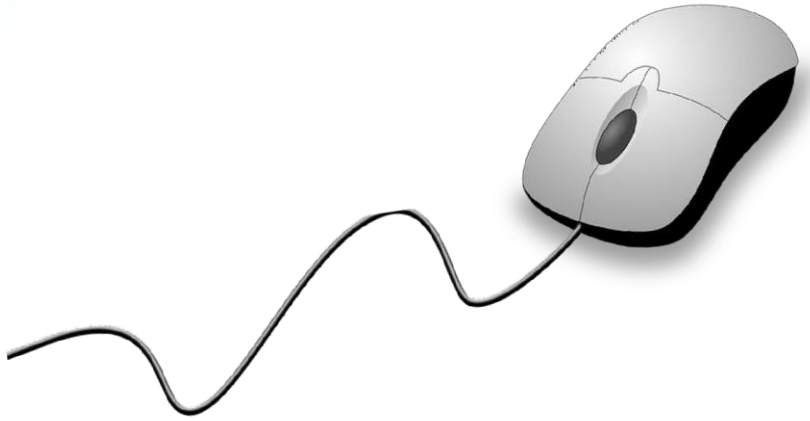
공개SW 솔루션 설치 & 활용 가이드

미들웨어 > WAS



WildFly

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

소개	• Java를 기반으로 하는 오픈소스 미들웨어로써 레드햇에서 상용제품인 Jboss EAP와 혼동을 막기위해 Jboss AS8 버전부터 WildFly라는 이름으로 변경 • 대표적으로 Java EE스펙을 지원하며 40개 이상의 다양한 프로젝트가 있으며 Jboss 커뮤니티에 의해 개발 및 운영		
주요기능	• domain mode 를 통한 중앙 집중 관리 용이 • 풍부한 관리 인터페이스 • 클러스터링을 통한 부하분산, 상태복제, 자동장애복구 등을 구현		
대분류	• 미들웨어	**소분류**	• WAS
라이선스형태	• GNU Lesser General Public License	**사전설치 솔루션**	• JDK 1.8 이상
운영제제	• Linux	**버전**	• 10.0
특징	• 서브시스템의 모듈화로 경량화, 확장성 • 최신 엔터프라이즈 Java 표준구현 • Web Console 기능 추가 및 보완, 관리 기능 강화		
보안취약점	• 취약점 ID : CVE-2018-10683 • 심각도 : 9.8 CRITICAL(V3) • 취약점 설명 : 보안 영역 참조가 없는 기본 설치의 경우 공격자는 인증없이 서버에액세스할수 있음 • 대응방안 : 기본적으로 각 관리 인터페이스에 인증 메커니즘을 추가하고 익명 사용자에 대한 사용 권한을 제한, 관리자/사용자만 .war 파일을 배포할 수 있어야하며 기본적으로 자동배포를사용중지 • 참고 경로 : https://github.com/kmkz/exploit/blob/master/CVE-2018-10682-CVE-2018-10683.txt		
개발회사/커뮤니티	• Jboss 커뮤니티		
공식 홈페이지	• https://www.wildfly.org		

2. 기능요약

• WildFly10 은 오픈소스이며 자바를 기반으로 하는 오픈소스 미들웨어이다.

주요기능	내 용
관리	웹브라우저를 통한WEB admin console 지원
클러스터링	다양한 클러스터링 지원으로 고가용성 및 서비스 유지
모드	단일 노드 인스턴스로 되어있는 standalone mode 다수의 노드 인스턴스들을 관리할 수 있는 domain mode

3. 실행환경

- WildFly10은 JDK 1.8 버전 이상의 최신버전을 사용하길 권장한다.
- 설치를 위하여 최소 200M의 용량이 필요하다.
- 메모리는 최소 64M가 필요하다.

Operatiing System	CPU Arch.	Java
Red Hat Enterprise Linux 6 Red Hat Enterprise Linux 7	X86_64	JDK 1.8.x
Micorsoft Windows 2008 Server	x86_64	JDK 1.8.x
CentOS 6.x CentOS 7.x	x86_64	JDK 1.8.x

4. 설치 및 실행

세부 목차

4. 설치 및 실행

4.1 설치(1/5)

4.1.1 OpenJDK 설치

- WildFly10을 사용하기 위해서는 JDK 1.8 버전 이상이 필요하다. 다음과 같이 설치 후 확인한다.

```
root@localhost~# yum install java-1.8.0-openjdk
root@localhost~# java -version
openjdk version "1.8.0_141"
OpenJDK Runtime Environment (build 1.8.0_141-b16)  Ope
nJDK 64-Bit Server VM (build 25.141-b16, mixed mode)
```

4.1.2 WildFly10 설치

- WildFly 10 의 배포본은 다음 링크에서 받을 수 있다. Zip 또는 tar 파일 형식으로 제공한다.

  http://www.wildfly.org/downloads/

- WildFly10은 인스톨러를 사용하지 않는다.
- 다운로드 받은 파일을 원하는 디렉토리에 압축해제만으로 설치가 가능하다.

```
root@localhost~# mkdir /usr/local/wildfly10
root@localhost~# cd /usr/local/wildfly10
root@localhost~# tar -xvzf wildfly-10.1.0.Final.tar.gz
```

4.1.3 시작하기(1/4)

- WildFly10은 관리콘솔을 사용하기 위해서 관리자 계정을 생성한다.
- $WildFly10_HOME/bin/add_user.sh 스크립트를 사용하여 추가한다.

```
[root@localhost bin]# ./add-user.sh

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
b)Application User (application-users.properties)
(a): a
```

- 서버관리를 위한 사용자 추가이기에 'a) Management User'를 선택하여 생성한다.

```
Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username :
Password :
Re-enter Password :
```

4. 설치 및 실행

4.1 설치(3/5)

4.1.3 시작하기(2/4)

• WilFly10이 설치된 디렉토리로 이동하여 다음 명령어를 입력한다.

•WildFly10은 standalone, domain 두 개의 모드로 동작이 가능하며 standalone mode는 기존 Jboss 이전 릴리즈에서 사용되었던 실행스크립트와 동일하다.

• standalone mode 로 실행

```
root@localhost~# cd $WILDFLY_HOME/bin
root@localhost~# ./standalone.sh
```

• domain mode 로 실행

```
root@localhost~# cd $WILDFLY_HOME/bin
root@localhost~# ./domain.sh
```

• -server-config 옵션을 이용하여 프로파일을 지정하여 서버를 시작할 수 있다.

• default 는 standalone.xml / domain.xml 이다.

```
root@localhost~# cd $WILDFLY_HOME/bin
root@localhost~# ./standalone.sh -server-config=standalone-ha.xml
```

4.1.3 시작하기(3/4)

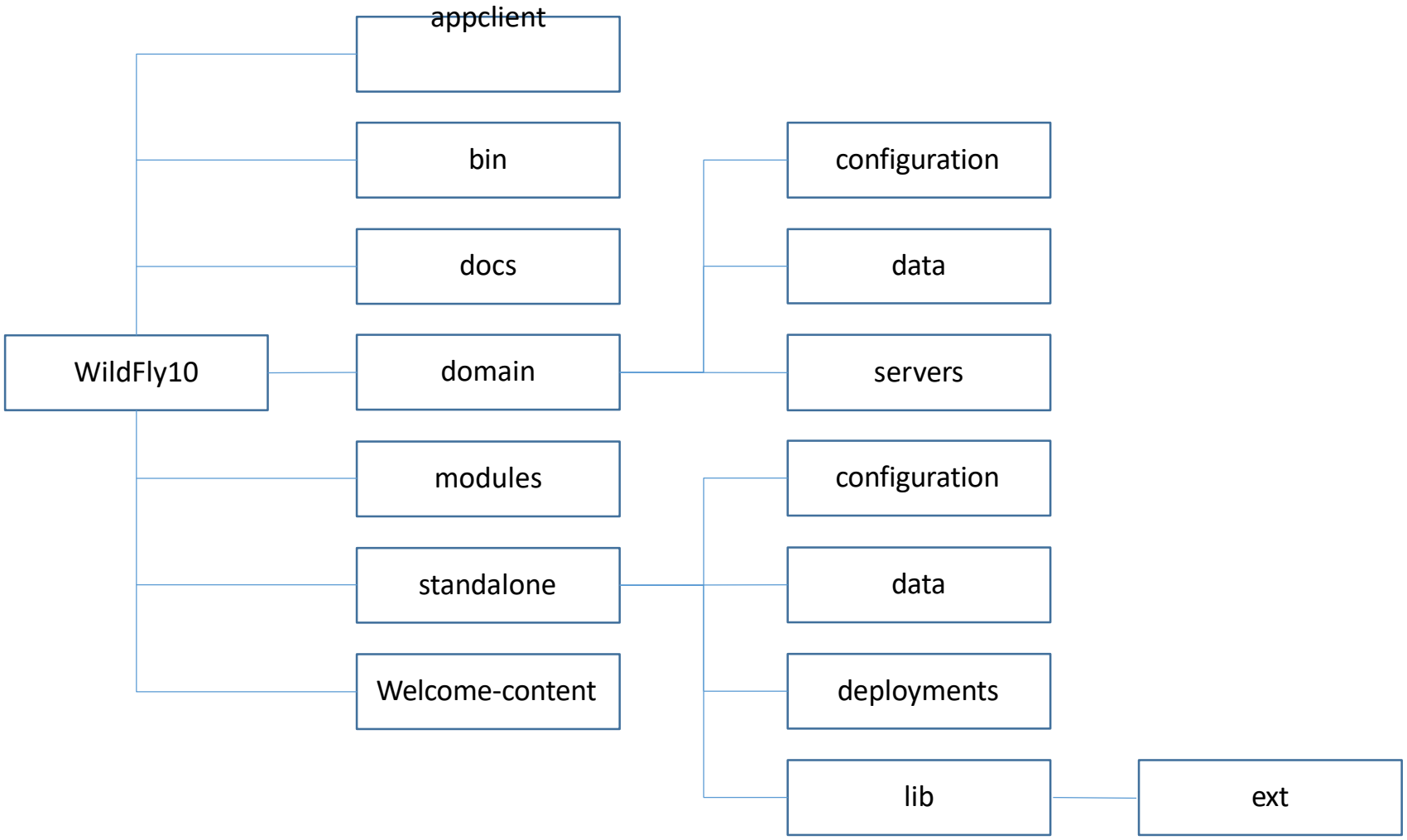
- standalone mode 구성에서 프로파일이 각각의 파일로 존재한다.
  - standalone.xml, standalone-ha.xml, standalone-full.xml, standalone-full-ha.xml
- domain mode 구성에서는 domain.xml 에서 모든 프로파일 정의

standalone 구성파일	standalone.xml	standalone-ha.xml	standalone-full.xml	standalone-full-ha.xml
domain 프로파일 명	default	ha	full	full-ha
jacorb	-	-	O	O
messaging	-	-	O	O
webservice	-	-	O	O
jgroups	-	O	-	O
modcluster	-	O	-	O
infinispan	O	O	O	O
ee	O	O	O	O
ejb3	O	O	O	O

4.1.3 시작하기(4/4)

- 시작페이지를 열어 서버가 제대로 작동하는지 확인할 수 있다.
- http://localhost:8080 에서 작동한다.

4.2.1 전체 디렉토리 구조 - $WildFly10/ (1/2)



4.2.1 전체 디렉토리 구조 - $WildFly10/ (2/2)

디렉토리	내 용
appclient	- 클라이언트용 홈 디렉토리
bin	- 시작 스크립트, 시작 설정파일
docs/schema	- xml schema 정의파일
docs/examples/configs	- 특정 사용 사례를 나타내는 구성파일의 예시
domain	- domain 홈 디렉토리
modules	- 추가 모듈 배포 디렉토리
standalone	- Standalone 홈 디렉토리
welcome-content	- http://localhost:8080 접속시 보여주는 기본설정페이지

4.2.2 standalone mode 디렉토리 구조 - $WildFly10/standalone/

- 이전 Jboss와 같은 단일 서버 인스턴스로 구성

디렉토리	내 용
configuration	- Standalone 구성 디렉토리
data	- Standalone server 실행 시 생성되는 파일들을 보관
deployments	- App 배포 디렉토리
lib/ext	- 자바 SE/EE 스타일 'extensions'를 지원, 설치된 라이브러리가 위치
log	- 로그 파일 기본 보관 디렉토리
tmp	- 임시파일 위치

4.2.3 domain mode 디렉토리 구조 - $WildFly10/domain/

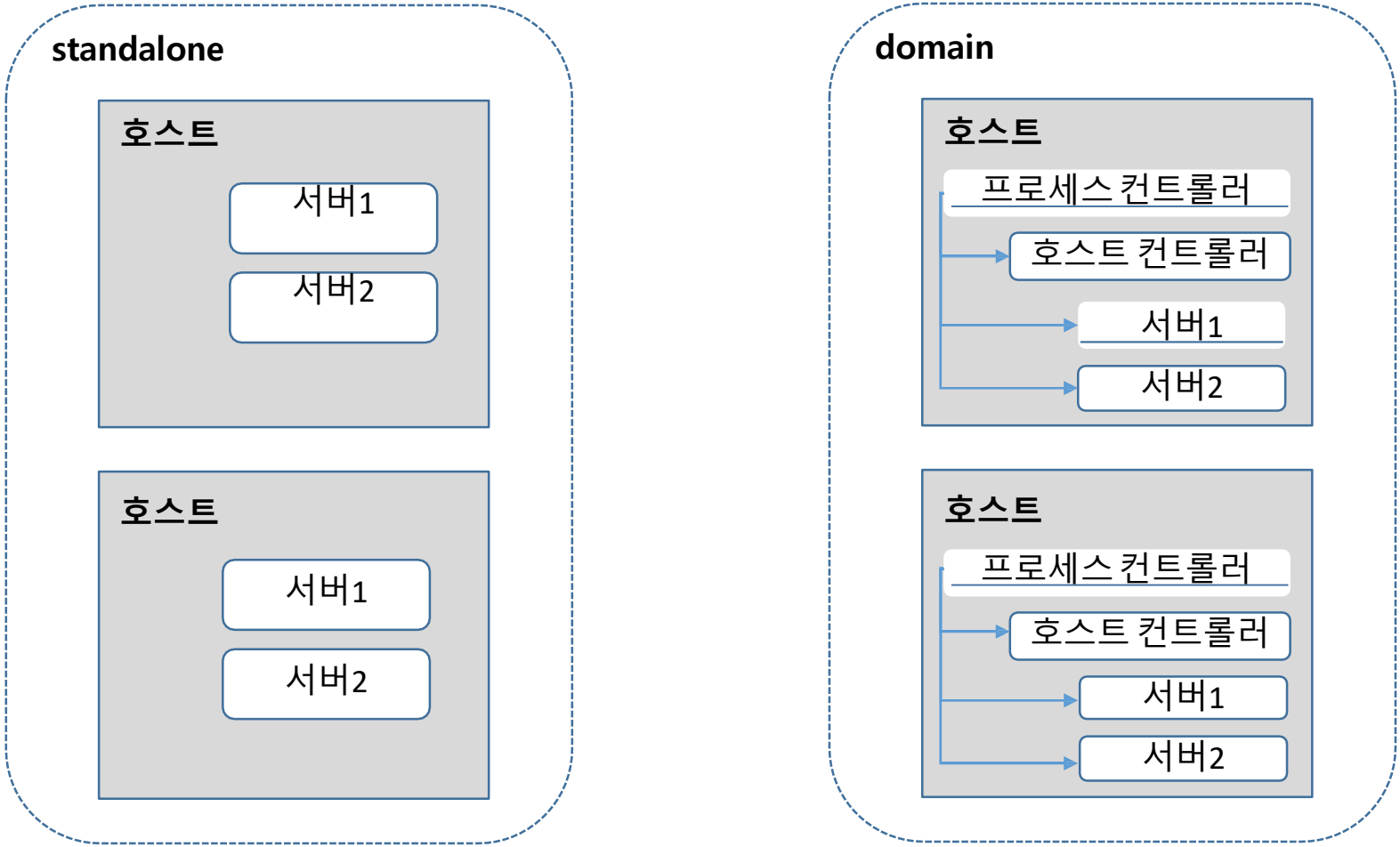
- 다중 서버 모드
- 도메인 컨트롤러로 여러 서버 인스턴스들을 관리

디렉토리	내 용
configuration	- domain 구성 디렉토리
data	- domain sever 실행 시 생성되는 파일들이 보관
cotent	- 배포된 모듈을 저장하기 위한 저장소로 사용
lib/ext	- 자바 SE/EE 스타일 'extensions'를 지원, 설치된 라이브러리가 위치
log	- domain server 관련 로그 파일 위치
servers	- 서버 인스턴스 별 디렉토리 - 하위 디렉토리에는 log. tmp 폴더가 위치
tmp	- 임시파일 위치

4.3.1 standalone mode와 domain mode비교(1/2)

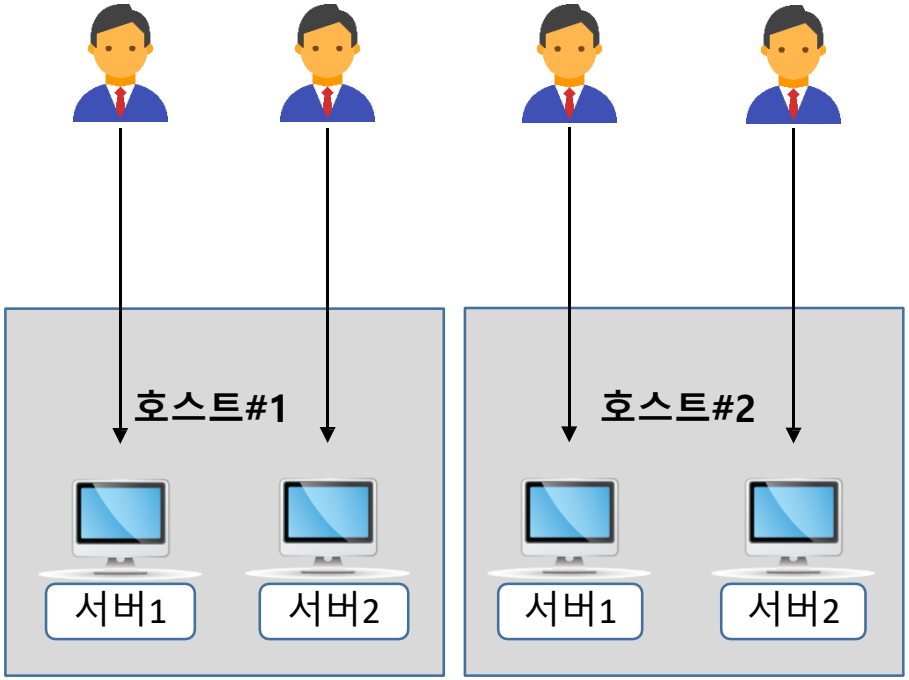
- **standalone mode**
  - 단일 인스턴스로 실행된다.
  - 인스턴스 별 개별관리가 필요하며 각 인스턴스별 프로파일 설정을 해줘야 한다.
- **domain mode**
  - 프로파일 설정은 도메인 컨트롤러에 포함된다.
  - 다수의 인스턴스가 그룹으로 구성되어 있다.
  - 인스턴스 별 그룹 지정이 가능하며 그룹별 설정이 가능하다.

4.3.1 standalone mode와 domain mode비교(2/2)

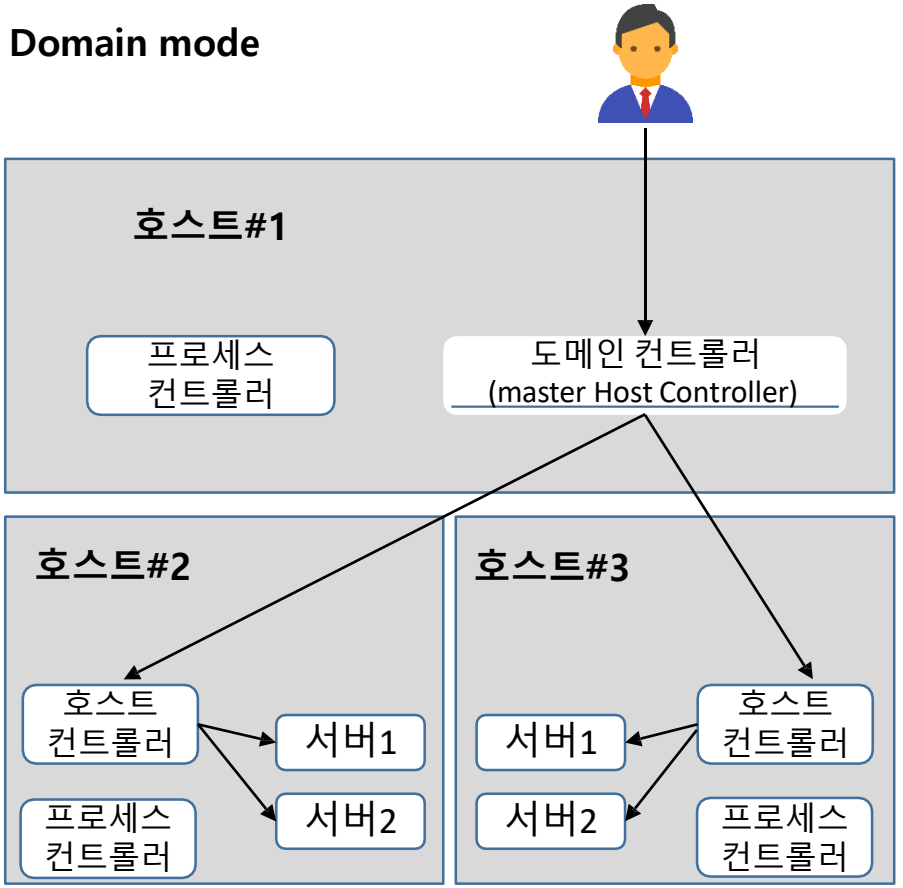


4.3.2 관리 방식

standalone mode



Domain mode



4.3.3 standalone mode 설정(1/5)

- 설정파일 : $WILDFLY_HOME/standalone/configuration/standalone.xml
- 환경설정은 다음과 같은 주요 요소로 구성되어 있다.
- **extensions**

  - 애플리케이션 서버는 모든 서비스에서 공유되는 확장이라는 기본적인 모듈 리스트를 포함한다. 애플리케이션의 기능을 확장하는데 사용되는 모듈로 볼 수 있으며 $WILDFLY_HOME/modules 폴더에 저장된다.

```
<extensions>
        <extension module="org.jboss.as.clustering.infinispan"/>
        <extension module="org.jboss.as.connector"/>
        <extension module="org.jboss.as.deployment-scanner"/>
</extensions>
```

- **paths (1/2)**

  - 시스템 경로들을 논리적으로 정의한 경로들을 확인할 수 있다.
  - $WILDFLY_HOME/standalone/log/server.log 경우 다음과 같다.
  - jboss.server.log.dir 는 log 디렉토리를 가리킨다.

```
<file relative-to="jboss.server.log.dir" path="server.log"/>
```

4.3.3 standalone mode 설정(2/5)

- **paths(2/2)**

  - WildFly는 여러가지 표준경로를 자동으로 제공한다.

경 로	내 용
Jboss.home.dir	WildFly 배포판의 루트 디렉토리
User.home	사용자의 홈 디렉토리
User.dir	사용자의 현재 작업 디렉토리
Java.home	자바가 설치된 디렉토리
Jboss.server.base.dir	개별 서버 인스턴스의 루트 디렉토리
Jboss.server.config.dir	서버가 구성 파일 저장에 사용할 디렉토리
Jboss.server.data.dir	서버에서 영구 데이터 파일 저장에 사용할 디렉토리
Jboss.server.log.dir	로그 파일 저장에 사용할 디렉토리
Jboss.server.temp.dir	임시 파일 저장에 사용할 디렉토리
jboss.domain.servers.dir	호스트 컨트롤러가 개별 서버 인스턴스에 대해 생성하는 작업공간이 해당 디렉토리 아래에 생성

4.3.3 standalone mode 설정(3/5)

- **Profiles**

- 프로파일은 서브시스템의 모음이라고 볼 수 있는데 각 서브시스템은 차례대로 애플리케이션 서버에서 사용하는 기능의 하위 집합으로 포함된다. 웹 서브시스템은 컨테이너에 사용되는 커넥터의 정의가 포함되고 메시지 서브시스템은 JMS설정과 애플리케이션 서버의 메시지 제공자가 사용하는 모듈이 정의된다.

- standalone mode와 domain mode 설정 파일의 차이는 포함하는 프로파일 개수이다. standalone mode 설정은 서브시스템이 포함된 단일 프로파일이 사용되어지는데 반해 domain mode 설정은 여러 개의 프로파일들을 가질 수 있다.

- **Interfaces(1/2)**

-애플리케이션 서버가 바인딩하는 네트워크 인터페이스 / IP주소 / 호스트명이 포함된다. standalone 애플리케이션 서버는 public 과 management 두개의 네트워크 인터페이스가 정의된다.

```
<interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:127.0.0.1}"/>
        </interface>
</interfaces>
```

4.3.3 standalone mode 설정(4/5)

- **Interfaces(2/2)**
  - public 네트워크 인터페이스는 애플리케이션 코어 서비스에 사용되기 위한 것이다.
  - management 네트워크 인터페이스는 AS 매니지먼트 인터페이스에 사용된다.
  - 두 네트워크 인터페이스는 기본적으로 127.0.0.1 주소로 자기자신을 가리킨다
  - inet-address value 값을 변경하면 원하는 장비 또는 다른 IP주소로 접근이 가능하다

4.3.3 standalone mode 설정(5/5)

- **Socket Binding Group**
  - 소켓 바인딩은 소켓 설정 이름으로 구성된다.
  - 포트를 열고 수신하도록 네트워크 포트를 구성한다.
  - 포트는 사용자가 원하는 포트로 변경이 가능하다.

```
"   port-offset="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="management-http" interface="management
"   port="${jboss.management.http.port:9990}"/>
        <socket-binding name="management-https" interface="management
"   port="${jboss.management.https.port:9993}"/>
        <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
        <socket-binding name="http" port="${jboss.http.port:8080}"/>
        <socket-binding name="https" port="${jboss.https.port:8443}"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

4.3.4 domain mode 주요 용어

- **도메인 컨트롤러 (메인 호스트 컨트롤러)**

  - domain 에 대한 중앙에서 집중화된 관리/통제를 위한 싱글 프로세스

  -호스트 컨트롤러 프로세스에 의해 관리되는 서버 인스턴스들에 대해서 domain 환경에서 구성을 관리하고 배포하는 프로세스

- **호스트 컨트롤러**

  - 도메인 컨트롤러 역할을 겸비한다

  - domain에 포함된 개별 호스트에서 실행되는 프로세스로 도메인 컨트롤러와 연결된 개개의 호스트

  -도메인 컨트롤러 프로세스와 연결된 호스트 컨트롤러에 의해서 각각의 서버 인스턴스에 대한 배포와 환경을 성정

- **프로세스 컨트롤러**

  - 서버 인스턴스 및 호스트 컨트롤러 프로세스의 기동/정지 및 모니터링

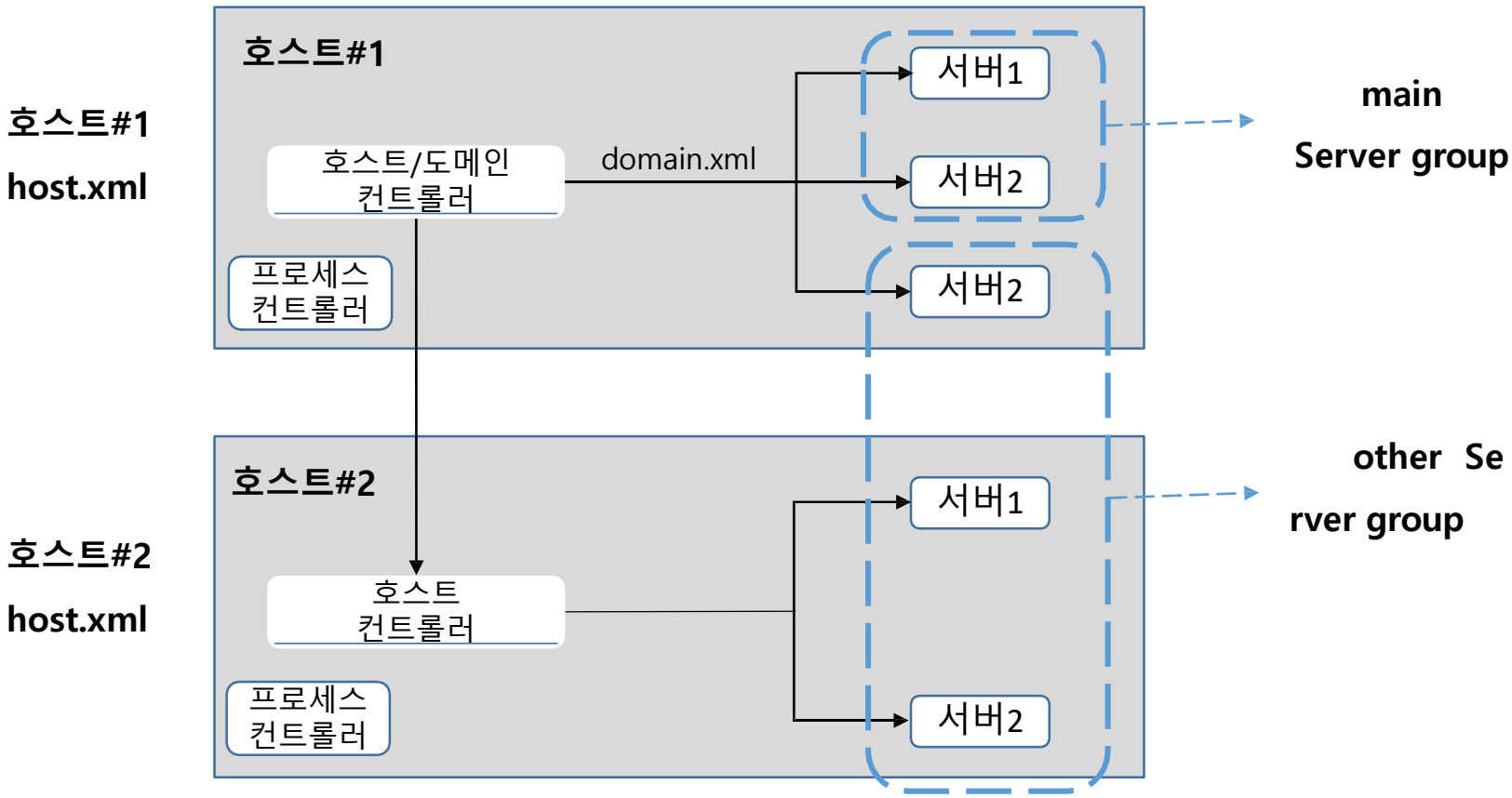
4.3.5 domain mode 설정(1/9)

- 기본적으로 domain server 에서 domain은 관리 단위이며 그 안에서 WildFly 서버들은 도메인 컨트롤러에 의해 관리된다.
- 설정파일 1 - $WildFly10_HOME/domain/configuration/domain.xml
  - domain server의 기능들을 나타내고, domain의 부분인 서버 그룹을 정의
- domain.xml 의 설정은 그 domain 내에서 모든 서버가 공유한다. 파일의 내용은 standalone와 동일한 구조를 따른다.
- 각 AS 노드들은 서버그룹으로 나뉠 수 있고 그룹별로 서로 다른 프로파일을 적용받는다. 서버그룹으로 인하여 노드들은 동일한 프로파일을 적용받거나 각기 특정 프로파일을 적용받을 수 있다.
- 하나의 서버그룹이 적용받을 수 있는 공통속성은 다음과 같다.
  - profiles, socket binding group, deployments, system properties, jvm setting

4.3.5 domain mode 설정(2/9)

- 설정파일 2 - $WildFly10_HOME/domain/configuration/host.xml
  - domain의 일부인 호스트 상에서 구동되는 서버 노드를 설정한다.
  - domain은 여러 개의 호스트(호스트#1, 호스트#2, ..)를 가지고 또한 여러 개의 서버그룹을 가질 수 있다.
  - 하나의 장비에 여러 개의 AS를 설치할 수 있어 동일 장비에 다중 호스트를 가질 수도 있다.
- 다음 장의 그림을 참고하여 이해하자.

4.3.5 domain mode 설정(3/9)



4.3.5 domain mode 설정(4/9)

- host.xml 설정에서 먼저 도메인 컨트롤러를 구성하는 방법을 먼저 알아보자.
- 도메인 컨트롤러를 구성하기 위해서는 다음 두 단계가 필요하다.
  - 호스트가 전체 도메인에 대한 도메인 컨트롤러로 작동하도록 구성
  - 호스트는 관리호스트가 접근할 수 있도록 관리 인터페이스를 설정
- 도메인 컨트롤러로 지정하여 도작하도록 설정한다.

```
<domain-controller>
        <local/>
</domain-controller>
```

4.3.5 domain mode 설정(5/9)

- 관리 인터페이스를 설정한다.
- 관리 인터페이스는 커맨드라인 인터페이스(CLI)와 http 인터페이스를 가지고 있다.
- 기본적으로 네트워크 관리 인터페이스에 바인딩 된다.
- CLI는 9999포트, http관리 인터페이스는 9990을 사용한다.

```
 <management-interfaces>
            <native-interface security-realm="ManagementRealm">
                <socket interface="management"
port="${jboss.management.native.port:9999}"/>
            </native-interface>
            <http-interface security-realm="ManagementRealm" http-upgrade-
enabled="true">
                <socket interface="management"
port="${jboss.management.http.port:9990}"/>
            </http-interface>
        </management-interfaces>
```

4.3.5 domain mode 설정(6/9)

- 네트워크 관리 인터페이스는 management 와 public 두개가 있다.
- inet-address 값을 변경함으로서 애플리케이션 서버의 주소를 설정할 수 있다.
- 다음은 managerment 인터페이스와 public 인터페이스가 루프백으로 설정된 경우이다.

```
    <interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:127.0.0.1}"/>
        </interface>
    </interfaces>
```

- master 도메인 컨트롤러는 인증을 요구하도록 되어있어 slave 용 계정을 만든다.
  (master 서버에서 생성해야만 한다.)
- 사용자 추가 스크립트인 $WildFly10_HOME/bin/add_user.sh 로 생성한다.

4.3.5 domain mode 설정(7/9)

- 도메인 컨트롤러를 구성하였다면 도메인에 속해있는데 호스트들의 호스트 컨트롤러를 구성한다.
  - 도메인 내에 속해있는 호스트들은 각자 고유의 이름으로 되어있어야 한다.
  - 호스트 컨트롤러는 도메인 컨트롤러 IP주소를 알아야 한다.
- 호스트에 대한 이름을 부여 합니다. 예시로 "slave"로 지정한다.

```
<host xmlns="urn:jboss:domain:4.2" name="slave">
….중략…
</host>
```

- 추후 관리 및 접근을 위한 보안영역을 정의한다.
  - secret value 값은 slave 용 계정 패스워드를 입력한다. (base64 타입)

```
<security-realm name="SlaveRealm">
  <server-identities>
    <secret value="cE3EBEkE=" />
  </server-identities>
</security-realm>
```

- 메인 도메인컨트롤러를 지정해준다.

```
<domain-controller>
   <remote protocol="remote" host="192.168.0.101" port="9999" username="slave" s
ecurity-realm="SlaveRealm"/>
</domain-controller>
```

4.3.5 domain mode 설정(8/9)

- 도메인컨트롤러는 하나 이상의 서버그룹에 속하게 된다.
- domain.xml 에서 서버그룹을 설정하며 host.xml 에서 속할 그룹 및 설정을 정의한다.
  - 서버그룹은 노드별 설정을 달리 적용할 수 있게 한다.
  - 각 서버 그룹은 JVM 셋팅, 소켓바인딩 인터페이스 설정을 적용시킬 수 있다.

```
<server-groups>
    <server-group name="main-server-group" profile="default">
        <jvm name="default">
           <heap size="64m" max-size="512m"/>
           <permgen size="128m" max-size="128m"/>
        </jvm>
        <socket-binding-group ref="standard-sockets"/>
    </server-group>
    <server-group name="other-server-group" profile="bigger">
        <jvm name="default">
            <heap size="64m" max-size="512m"/>
        </jvm>
        <socket-binding-group ref="bigger-sockets"/>
    </server-group>
</server-groups>
```

4.3.5 domain mode 설정(9/9)

- 각 노드의 host.xml 에서의 서버그룹 설정 내용이다.

```
<servers>
    <server name="server-one" group="main-server-group">
        <\!-\- server-one inherits the default socket-group declared in the ser
        v
er-group \-->
        <jvm name="default"/>
    </server>

    <server name="server-two" group="main-server-group" auto-start="true">
        <socket-binding-group ref="standard-sockets" port-offset="150"/>
        <jvm name="default">
            <heap size="64m" max-size="256m"/>
        </jvm>
    </server>

    <server name="server-three" group="other-server-group" auto-start="false">
        <socket-binding-group ref="bigger-sockets" port-offset="250"/>
    </server>
```

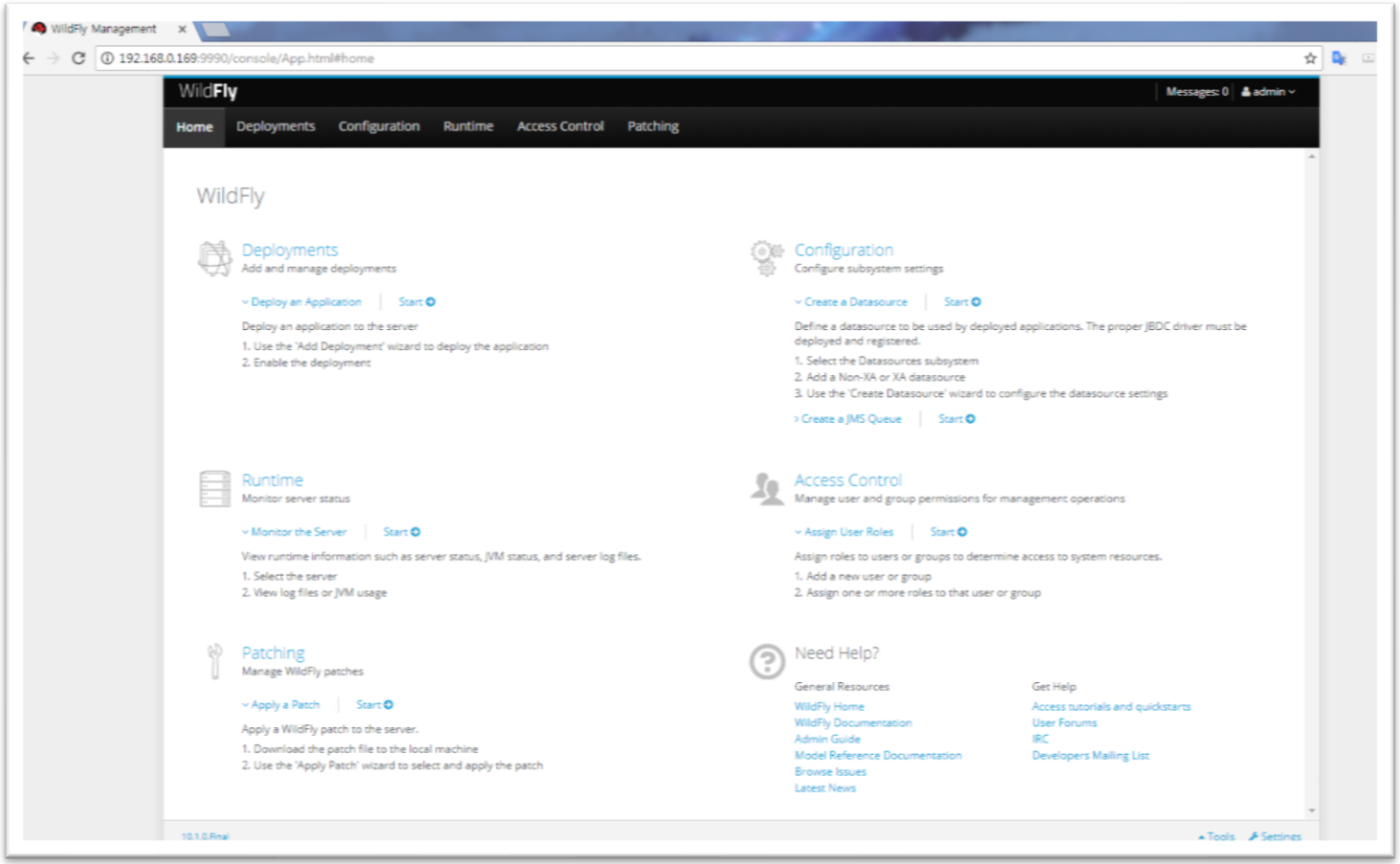
- JVM은 도메인컨트롤러 및 호스트컨트롤러에 설정이 가능하며 호스트컨트롤러의 설정을 우선시 한다.

5. 기능 소개

세부 목차

5.1.1 WEB admin console

- http://IP주소:9990/console 로 접속이 가능하다.
- 초기 생성한 계정을 통해 로그인 시 다음과 같은 관리화면을 볼 수 있다.



5.1.2 WEB admin console 메뉴소개

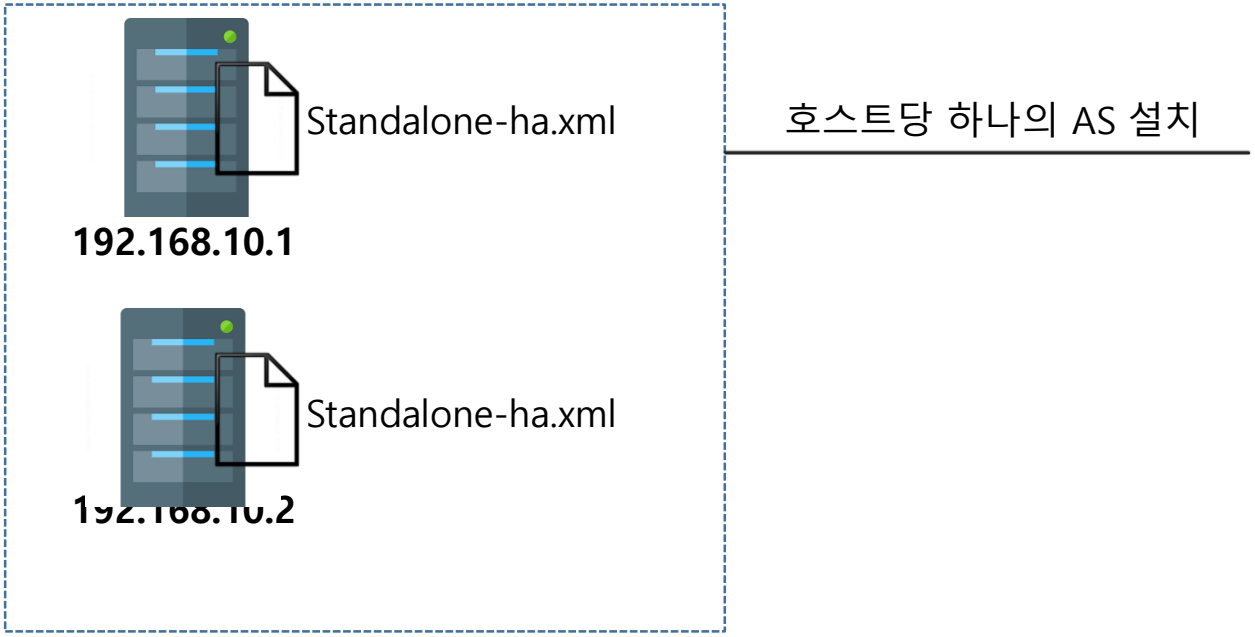
- **Home**
  - 초기 관리페이지 화면이다.
  - 각 Deployments, Configuration, Runtime, Access Control, Patching 메뉴로 이동할 수 있다.
- **Deployments**
  - 응용프로그램 및 기타 EE 리소스를 배포하고 관리한다.
- **Configuration**
  - 애플리케이션 서버 및 리소스 설정이 가능하다.
    (Subsystems, Interfaces, Socket Binding, Paths, System Properties)
- **Runtime**
  - 현재 구동중인 애플리케이션 서버에 대한 정보를 제공한다. (JVM, Environment, Log Files, Subsystems)
- **Access Control**
  - 사용자 및 그룹 생성 및 삭제, 접근권한 설정을 한다.
- **Patching**
  - 새로운 패치에 대한 적용 및 롤백이 가능하다. 패치파일은 사전에 다운로드 받아야 한다.

5.2.1 클러스터링

- 클러스터링은 확장성 및 고가용성을 위해 사용되어진다.
  - 대량 요청시 응답 지연 및 손실에 대비한다.
  - 단일지점 장애에 대비한다.
- 클러스터 빌딩 블록 (cluster-building block)을 설정한다.
  - 제이그룹스(Jgroups) 서브시스템은 노드들 사이의 통신을 위해 사용한다.
  - 인피니스팬(Infinispan) 서브시스템은 고급 데이터 그리드 플랫폼을 사용하여 클러스터의 일관성을 처리 한다.

5.2.2 standalone 클러스터링 설정 (1/5)

- standalone mode에서는 클러스터링을 위한 standalone-ha.xml 프로파일을 제공한다.
- standalone mode에서의 클러스터링은 다음과 두가지 경우가 있다.
  - 다른 장비에서 운영되는 AS 노드 클러스터
  - 동일 장비에서 운영되는 AS 노드 클러스터
- 다른 장비에서 운영되는 AS 노드 클러스터



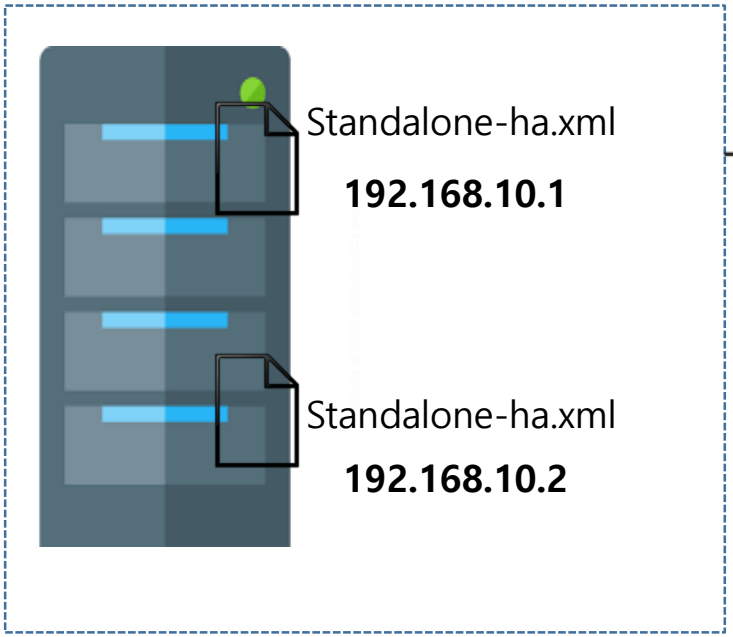
5.2.2 standalone 클러스터링 설정 (2/5)

- 두개의 노드를 구성시 각 노드의 standalone-ha.xml 파일을 열어 interfaces 부분을 열어 각 서버의 IP를 설정한다.

```
<interfaces>
        <interface name="management">
            <inet-address value="${jboss.bind.address.management:192.168.10.1}"/>
        </interface>
        <interface name="public">
            <inet-address value="${jboss.bind.address:192.168.10.1}"/>
        </interface>
</interfaces>
```

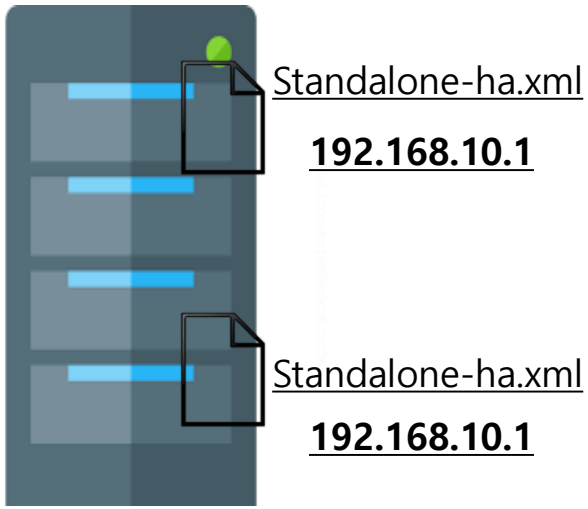
5.2.2 standalone 클러스터링 설정 (3/5)

- 동일 장비에서 운영되는 AS 노드 클러스터
- 동일 장비에서 서버 클러스터링을 구성하는 경우 다음 두가지 경우를 유의해야 한다.
  - 동일 장비에 다중 IP 주소를 정의
  - 각 서버를 위한 포트 정의 (포트 중복을 피하기 위해)
- 동일 장비에 다중 IP 구성
  - 각 standalone-ha.xml 에 interfaces 부분에 사용 IP를 설정한다.



5.2.2 standalone 클러스터링 설정 (4/5)

- 동일 장비에 포트 변경을 통한 구성



5.2.2 standalone 클러스터링 설정 (5/5)

- 동일 장비에 포트 변경을 통한 구성의 경우 interfaces 의 경우 동일 IP를 설정하며 port-offset을 설정하여 클러스터링을 할수 있다.
- 첫 번째 서버의 경우 기본 소켓 바인딩 포트를 사용한다.

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offs
et="${jboss.socket.binding.port-offset:0}">
```

- 두 번째 서버의 경우 150의 port-offset 를 설정해준다.
- 포트 충돌을 방지하기 위해 관리 인터페이스 포트도 변경해준다.
- 첫 번째 서버는 기본값(9990)으로 남겨두고 두 번째 서버의 포트를 변경한다.

```
<socket-binding-group name="standard-sockets" default-interface="public" port-offs
et="${jboss.socket.binding.port-offset:150}">
        <socket-binding name="management-http" interface="management" port="${jbos
s.management.http.port:19990}"/>
……
</socket-binding-group>
```

- 클러스터 설정이 완료됐다. 다음과 같이 설정 파일을 불러들여 실행한다.

```
root@localhost~# cd $WILDFLY_HOME/bin
root@localhost~# ./standalone.sh –server-config=standalone-ha.xml
```

5.2.3 domain 클러스터링 설치 (1/4)

- domain mode는 domain.xml에 클러스터링 환경을 위한 ha 설정내용이 포함되어있다.
- 클러스터링 사용을 위해서 ha 프로파일을 가리키는 서버그룹을 설정한다.

```
<server-groups>
        <server-group name="main-server-group" profile=" ha">
            <jvm name="default">
                <heap size="64m" max-size="512m"/>
            </jvm>
            <socket-binding-group ref=“ha-sockets"/>
        </server-group>
        <server-group name="other-server-group" profile="ha">
            <jvm name="default">
                <heap size="64m" max-size="512m"/>
            </jvm>
            <socket-binding-group ref="ha-sockets"/>
        </server-group>
    </server-groups>
```

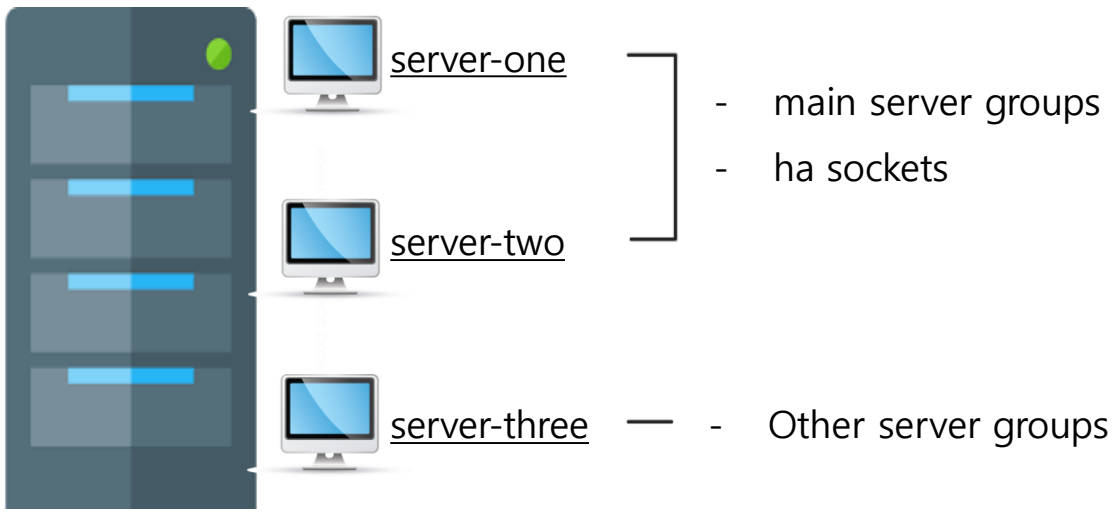
5.2.3 domain 클러스터링 설치 (2/4)

- socket-binding-group 도 ha-sockets을 참조한다.

```
<socket-binding-group name="ha-sockets" default-interface="public">
            <!-- Needed for server groups using the 'full' profile  -->
            <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
            <socket-binding name="http" port="${jboss.http.port:8080}"/>
            <socket-binding name="https" port="${jboss.https.port:8443}"/>
            <socket-binding name="iiop" interface="unsecure" port="3528"/>
            <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
            <socket-binding name="txn-recovery-environment" port="4712"/>
            <socket-binding name="txn-status-manager" port="4713"/>
            <outbound-socket-binding name="mail-smtp">
                <remote-destination host="localhost" port="25"/>
            </outbound-socket-binding>
        </socket-binding-group>
```

5.2.3 domain 클러스터링 설치 (3/4)

- 아래와 같은 구성이 설정되었다.



5.2.3 domain 클러스터링 설치 (4/4)

- 실행되는 각 서버(AS)의 host.xml 파일 내에 ha-sockets 를 참조하도록 설정한다.

```
<servers>
        <server name="server-one" group="main-server-group">
              <socket-binding-group ref="ha-sockets" port-offset="150">
              <jvm name="default">
 </server>
        <server name="server-two" group="main-server-group" auto-start="true">
              <socket-binding-group ref="ha-sockets" port-offset="250">
              <jvm name="default">
 </server>
        <server name="server-three" group="other-server-group" auto-start="false">
              <socket-binding-group ref="standard-sockets" port-offset="350">
              <jvm name="default">
</server>
 </servers>
```

- 실행준비가 완료되었다 domain.sh 를 사용하여 실행하면 서버그룹은 ha 설정파일을 참조하여 두개의 노드 클러스터를 만든다.

```
root@localhost~# cd $WILDFLY_HOME/bin
root@localhost~# ./domain.sh
```

5.2.4 세션 클러스터링 (1/3)

- 클러스터 노드들 간의 데이터 동기화 및 세션클러스터 기능을 수행한다.
- 한 인스턴스의 장애 발생 시 다른 인스턴스에서 세션을 유지한다.
- 설정은 standalone-ha.xml / domain.xml 에서 가능하다.

```
 <subsystem xmlns="urn:jboss:domain:infinispan:4.0">
            <cache-container name="server" aliases="singleton cluster" default-cache="default" module="org
.wildfly.clustering.server">
                <transport lock-timeout="60000"/>
                <replicated-cache name="default" mode="SYNC">
                    <transaction mode="BATCH"/>
                </replicated-cache>
            </cache-container>
            <cache-container name="web" default-cache="dist" module="org.wildfly.clustering.web.infinispan
">
            ……
            </cache-container>
            <cache-container name="ejb" aliases="sfsb" default-cache="repl" module="org.wildfly.clustering
.ejb.infinispan">
            ……
            </cache-container>
```

5.2.4 세션 클러스터링 (2/3)

- cache-container 요소는 하나 또는 여러 개의 캐시 정책이 있는데 이는 어떻게 캐시 컨테이너 대해 데이터를 동기화하는지에 대해 정의한다.
- 캐시 정책은 다음과 같다.
  - Local(로컬) : 클러스터가 만들어지는 것과 관계없이 값들은 로컬 노드에만 저장된다. 이 모드에서 인피니스팬은 일반적으로 로컬 캐시로 운영된다.
  - Replication(복제) : 모든 값은 모든 노드에 복제된다.
  - Distribution(분산) : 값들은 노드의 하위집합에 분산된다.
  - Invaildation(무효화) : 값들은 캐시 저장소(데이터베이스 같은)에 저장되고, 모든 노드로부터 무효화된다. 한 노드가 값이 필요할 때, 캐시 저장소에서 값을 로딩한다.

5.1.4 세션 클러스터링 (3/3)

- 다음은 캐시 설정을 위한 세부내용이다.

```
  <cache-container name="web" aliases="standard-session-cache" default-cache="repl"
module="org.wildfly.clustering.web.infinispan">
      <transport lock-timeout="60000"/>
      <replicated-cache name="repl" mode="ASYNC" batching="true">
          <locking isolation="REPEATABLE_READ"/>
          <file-store/>
      </replicated-cache>
       <distributed-cache name="dist" mode="ASYNC" batching="true" l1-lifespan="0">
          <file-store/>
      </distributed-cache>
  </cache-container>
```

- replicated-cache와 distributed-cache가 있으며 replicated-cache모드가 기본 설정되어 있다.
- 모드의 변경은 default-cache 값에 변경할 수 있다.
- 멤버들간의 데이터 동기화는 동기메세지(SYNC) 혹은 비동기 메시지(ASYNC)를 사용해서 수행한다.
- 동기 메시지 (SYNC)
  - 각 노드가 모든 클러스터 멤버로부터 메시지 확인을 기다린다.
  - 클러스터의 모든 노드가 높은 일관성을 요구하는 캐시데이터에 접근할 때 필요하다.
- 비동기 메시지 (ASYNC)
  - 일관성 보다는 속도를 강화하고 HTTP 세션 복제 같은 경우에 특히 유리하다.
  - 세션은 항상 같은 클러스터 노드에 접근하고 데이터 접근이 실패할 경우에만 다른 노드에 접근한다.

6. 활용 예제

세부 목차

6.1.1 WildFly 와 Apache 웹 서버 사용

- 실제 서비스에서 애플리케이션 서버의 앞 단에 Apache 웹 서버의 사용은 일반화 되어있다. 이는 다음과 같은 장점이 있다.
  - 속도 : Apache는 WildFly 웹 서버보다 정적인 컨텐츠를 서비스하는데 일반적으로 빠르다.
  - 보안 : 민감한 데이터를 가지고 있는 애플리케이션 서버는 보호구역에 두고 보안관점에서 Apache 웹 서버에 대해서만 걱정하면 된다.
  - 로드 밸런싱과 클러스터링 : 앞 단에 Apache를 사용함으로써 다중의 WildFly 웹 서버 인스턴스들에 트래픽을 처리할 수 있다. 애플리케리션 서버 중 하나에서 장애 발생 시 다른 노드에 통신을 유지한다.

6.1.2 mod_jk 설정(1/4)

- mod_jk는 Apache 웹 서버와 WildFly AS(애플리케이션 서버)를 연결할 때 가장 많이 사용하는 솔루션이다. 모든 요청은 처음에 Apache 웹 서버로 들어온다. Apache 웹 서버는 이미지 혹은 HTML 페이지들을 위한 요청과 같은 정적 자원들을 위한 요청을 수용하고 처리한다. 그리고 JSP 혹은 서블릿 컴포넌트들을 위한 요청들은 mod_jk 의 도움으로 WildFly 웹 서버 인스턴스로 보낸다. AJP 프로토콜을 사용해서 이러한 리다이렉션을 수행한다.
- mod_jk 설치하기 위해선 기본적으로 Apache 가 설치되어 있어야 한다.
- mod_jk 설치를 위해 http://tomcat.apache.org/download-connectors.cgi 에서 톰캣 커넥터를 다운받는다. 다운받은 파일을 압축해제후 native 디렉토리 이동 후 컴파일한다.

```
[root@localhost src]# tar -xvzf tomcat-connectors-1.2.42-src.tar.gz
[root@localhost src]# cd tomcat-connectors-1.2.42-src/native
[root@localhost src]# ./configure –with-apxs=/usr/sbin/apxs
[root@localhost src]# make
[root@localhost src]# make install
```

- 설치가 되었다면 /etc/httpd/modules/ 에서 mod_jk.so 파일을 확인한다.

6.1.2 mod_jk 설정(2/4)

- vi $APACHE_HOME/conf/httpd.conf 다음을 추가한다.

```
LoadModule jk_module    modules/mod_jk.so
<IfModule mod_jk.c>
        JkWorkersFile conf/workers.properties
        JkLogFile "logs/jk.log"
        JkLogLevel error
</IfModule>

JkMount /myapp/* loadbalancer
JkShmFile logs/jk.shm
```

- LoadModule 지시자는 mod_jk 라이브러리를 참조해야 한다.
- JkMount 지시자는 URL들을 mod_jk 모듈로 전달되어야 한다고 Apache에게 알려준다 위 설정파일레서는 URL경로 /myapp/*를 가지는 모든 요청은 mod_jk 커넥터로 보낸다. 모든 웹 애플리케이션에서 mod_jk를 사용하고자 한다면 /*를 사용해서 모든 URL을 전달할 수 있다.

6.1.2 mod_jk 설정(3/4)

- 다음은 conf/workers.properties을 설정한다. 이 파일은 각기 다른 웹 서버들이 있는 위치를 나타내고 어떻게 그 서버들로 로드 밸런싱 해야하는지를 표시한다. 단일노드를 위한 설정파일은 다음과 같다.

```
worker.list=tomcat1,tomcat2,loadbalance

############
## Was1 ##
############
## [Tomcat1]  worker.tomcat1.typ
e=ajp13  worker.tomcat1.host=192
.168.10.1  worker.tomcat1.port=8
009  worker.tomcat1.lbfactor=1

## [Tomcat2]  worker.tomcat2.typ
e=ajp13  worker.tomcat2.host=192
.168.10.2  worker.tomcat2.port=8
009  worker.tomcat2.lbfactor=1

worker.loadblance.balance_workers=tomcat1,tomcat2
worker.lb1.recover_time=600 #second default 60
```

6.1.2 mod_jk 설정(4/4)

- workers.properties 파일에서 각 노드는 worker.XXX 규칙을 사용해서 정의할 수 있다. 여기서 XXX는 각 웹 서버 컨테이너들을 위해 선택한 임의의 이름을 나타낸다. 각 worke를 위해서 호스트명(ip주소)과 웹 서버에서 구동하는 AJP13 커텍터의 포트를 명시해야 한다.
- WildFly 에서 standalone.xml 을 열어 해당 부분에 AJP 커텍터를 추가해준다.

```
            <buffer-cache name="default"/>
            <server name="default-server">
                <ajp-listener name="ajp" scheme="http" socket-binding="ajp"/>
                <http-listener name="default" socket-binding="http" redirect-
socket="https" enable-http2="true"/>
                <https-listener name="https" socket-binding="https" security-
realm="ApplicationRealm" enable-http2="true"/>
                <host name="default-host" alias="localhost">
                    <location name="/" handler="welcome-content"/>
                    <filter-ref name="server-header"/>
                    <filter-ref name="x-powered-by-header"/>
                </host>
            </server>
```

- WildFly 와 Apache 를 재시작 후 Apache에서 설정한 웹사이트를 접속하여 페이지를 확인한다.

Q Jboss와 WildFly의 차이는 무엇인가?

A JBoss AS는 JBoss Application Server의 약자이며 많은 경우 JBoss로 표기하여 Red Hat의 상용 제품인 JBoss EAP 와 혼돈을 야기했다. 이러한 혼란을 해소하기 위하여 JBoss AS8 버전부터는 기존의 JBoss라는 이름을 Wildfly라는 이름으로 변경하기로 결정했으며, 현재는 jboss.org 가 아닌 wildfly.org 커뮤니티에서 Wildfly이라는 이름으로 오픈소스 프로젝트를 진행하고 있다.

Q 상용과 오픈소스의 제품 차이가 있는가?

A 상용과 오픈소스는 차이가 없다. 상용은 subscription을 구매하면 되며 릴리즈가 오픈소스와 달리 공식적인 릴리즈 버전을 제공하고 이것에 따라서 hotfix나 patch도 릴리즈 된다.
버전업은 일반적으로 오픈소스가 더 빠르다. 따라서 오픈소스를 사용하더라도 버그가 패치되고 기능이 향상된 버전이 더 빨리 wildfly.org 에서 공개되기에 제품자체에 대한 차이는 없다. 상용과 오픈소스의 차이는 기술지원 여부이다.

Q 관리는 웹으로만 가능한가?

A WildFly 는 기본적으로 CLI 라는 커맨드라인 인터페이스와 웹으로 접근 가능한 WEB admin console 두 가지를 제공한다. 이전 버전까지는 web console이 Jboss의 가장 취약한 부분이었지만 WildFly10까지 버전업이 되며 많은 개선이 되었다.

Q standalone mode, domain mode ?

A 궁극적으로 standalone과 domain은 관리의 차이성 외에는 동일하다고 볼 수 있다.
다수의 standalone 서버를 운영한다면 각 서버 별 설정파일마다 관련 설정을 모두 해주어야 한다. 이런 경우는 domain을 사용하는 이점이 확실하다. 각 설정파일의 정의는 도메인 컨트롤러에 포함되고 중앙 집중화된 곳을 제공해주며 사용자들은 그곳을 통해 설정을 일관되게 유지하거나 여러 가지 방식으로 서버의 설정을 변경할 수 있다. 관리자에 의해 domain 설정에 준하는 시스템이 갖추어져 있다면 domain 설정은 바람직하지 않을 수 있다.

8. 용어정리

용어	설 명
서브 시스템	WildFly10에서 지원하는 모듈방식의 시스템. 여러 서브 시스템의 추가로 기능을 추가시킬 수 있다.
프로파일	각 서브 시스템 구성의 세부 사항과 서브 시스템의 이름이 지정된 목록. 설정파일이라 불리운다.
클러스터링	애플리케이션 서버의 가용성 보장을 위한 동작방식. 데이터의 동기화, 분산, 복제 등의 방식으로 fail-over, load balancing 기능을 지원하며 서비스의 연속성을 보장한다.

Open Source Software Installation & Application Guide