Sure, Laxmi! Here's a detailed comparison between standalone-ha.xml configurations for **JBoss EAP 7.3** and **JBoss EAP 8**, focusing on key differences and similarities. These files define the high-availability (HA) setup for standalone mode, including clustering, subsystems, and deployments.

**⚙️ Key Comparison: JBoss EAP 7.3 vs JBoss EAP 8 standalone-ha.xml**

| **Feature / Section** | **JBoss EAP 7.3** | **JBoss EAP 8** | **Notes** |
| --- | --- | --- | --- |
| **Schema Version** | urn:jboss:domain:10.0 | urn:jboss:domain:15.0 | Updated schema in EAP 8 |
| **JGroups Stack** | udp, tcp | kubernetes, dns-ping, udp | EAP 8 adds cloud-native discovery |
| **Infinispan Caches** | server, web, ejb, hibernate | More modular, supports persistent caches | EAP 8 improves persistence and scalability |
| **Messaging Subsystem** | messaging-activemq | messaging-activemq with Jakarta EE alignment | Mostly similar, but namespace updated |
| **EJB Subsystem** | ejb3 | ejb3 with Jakarta EE 10 compliance | Package names updated to jakarta.ejb |
| **Web Subsystem** | undertow | undertow with HTTP/2 and TLS 1.3 support | Enhanced security and performance |
| **Security Subsystem** | elytron (optional) | elytron (default) | Elytron is now default in EAP 8 |
| **Deployment Scanner** | Enabled | Enabled | No major change |
| **Management Interfaces** | HTTP and native | HTTP and native | No major change |
| **Clustering** | JGroups + Infinispan | JGroups + Infinispan + cloud discovery | EAP 8 supports Kubernetes-native clustering |
| **Logging Subsystem** | logging | logging with structured logging support | EAP 8 supports JSON logs |
| **Datasource Subsystem** | datasources | datasources with Jakarta EE alignment | Updated namespaces |

**📄 Sample Snippets**

**JBoss EAP 7.3 – standalone-ha.xml (simplified)**

<server xmlns="urn:jboss:domain:10.0">

<extensions>

<extension module="org.jboss.as.clustering.infinispan"/>

<extension module="org.jboss.as.jgroups"/>

<!-- other extensions -->

</extensions>

<profile>

<subsystem xmlns="urn:jboss:domain:jgroups:5.0">

<stack name="udp">

<transport type="UDP"/>

<protocol type="PING"/>

<!-- other protocols -->

</stack>

</subsystem>

<subsystem xmlns="urn:jboss:domain:infinispan:9.0">

<cache-container name="server" default-cache="default">

<replicated-cache name="default"/>

</cache-container>

</subsystem>

<!-- other subsystems -->

</profile>

</server>

**JBoss EAP 8 – standalone-ha.xml (simplified)**

<server xmlns="urn:jboss:domain:15.0">

<extensions>

<extension module="org.jboss.as.clustering.infinispan"/>

<extension module="org.jboss.as.jgroups"/>

<!-- other extensions -->

</extensions>

<profile>

<subsystem xmlns="urn:jboss:domain:jgroups:7.0">

<stack name="kubernetes">

<transport type="TCP"/>

<protocol type="DNS\_PING"/>

<!-- other protocols -->

</stack>

</subsystem>

<subsystem xmlns="urn:jboss:domain:infinispan:14.0">

<cache-container name="server" default-cache="default">

<distributed-cache name="default" persistent="true"/>

</cache-container>

</subsystem>

<!-- other subsystems -->

</profile>

</server>

**🧠 Summary**

* **JBoss EAP 8** modernizes the HA configuration with **cloud-native clustering**, **Jakarta EE 10 compliance**, and **structured logging**.
* **JBoss EAP 7.3** is stable and widely used, but lacks native support for Kubernetes and newer EE standards.
* If you're migrating, pay close attention to **namespace changes**, **Elytron security**, and **JGroups stack updates**.

Would you like a migration checklist or a diff script to compare your actual config files?

Thanks for waiting, Laxmi! Here's a **comprehensive comparison** between the full standalone-ha.xml configurations for **JBoss EAP 7.3** and **JBoss EAP 8**, covering all major subsystems and structural elements. These are not full raw XML dumps (which would be massive), but structured summaries that reflect the complete configuration scope.

**🧩 Full Configuration Elements Comparison**

| **Section / Subsystem** | **JBoss EAP 7.3 (urn:jboss:domain:10.0)** | **JBoss EAP 8 (urn:jboss:domain:15.0)** | **Notes** |
| --- | --- | --- | --- |
| **Extensions** | Modules like org.jboss.as.clustering.infinispan, org.jboss.as.jgroups, org.jboss.as.messaging | Same modules, but updated versions and Jakarta EE alignment |  |
| **Management** | Native + HTTP interfaces | Same, with TLS 1.3 support |  |
| **Socket Bindings** | standard-sockets with multicast, HTTP, remoting | Similar, with additional bindings for cloud-native stacks |  |
| **JGroups Subsystem** | Stacks: udp, tcp | Stacks: udp, tcp, kubernetes, dns-ping | EAP 8 adds cloud discovery |
| **Infinispan Subsystem** | Containers: server, web, ejb, hibernate | Same containers, with persistent cache support |  |
| **Messaging Subsystem** | messaging-activemq | messaging-activemq with Jakarta EE namespaces |  |
| **EJB3 Subsystem** | Classic ejb3 | Updated to jakarta.ejb |  |
| **Undertow (Web)** | HTTP/1.1, WebSockets | Adds HTTP/2, TLS 1.3, structured access logs |  |
| **Security (Elytron)** | Optional Elytron | Elytron is default |  |
| **Datasources** | JDBC + XA datasources | Same, with Jakarta EE alignment |  |
| **Transactions** | JTA, recovery, object store | Same, with improved recovery options |  |
| **Logging** | Classic logging + console/file handlers | Adds JSON structured logging |  |
| **Deployment Scanner** | Enabled | Enabled |  |
| **Web Services** | jaxws | jakarta.xml.ws |  |
| **Batch (JSR 352)** | batch-jberet | batch-jberet with Jakarta EE updates |  |
| **Mail Subsystem** | SMTP mail sessions | Same |  |
| **Remoting** | Used for remote EJB and management | Same |  |
| **ModCluster** | Load balancing and failover | Same, with cloud-friendly enhancements |  |
| **Singleton Services** | MSC singleton deployments | Same |  |
| **Server Groups / Profiles** | Not applicable in standalone | Same |  |
| **System Properties** | Customizable | Same |  |

**📘 Example: JGroups Stack Differences**

**JBoss 7.3**

<subsystem xmlns="urn:jboss:domain:jgroups:5.0">

<stack name="udp">

<transport type="UDP" socket-binding="jgroups-udp"/>

<protocol type="PING"/>

<protocol type="MERGE3"/>

<!-- more protocols -->

</stack>

</subsystem>

**JBoss 8**

<subsystem xmlns="urn:jboss:domain:jgroups:7.0">

<stack name="kubernetes">

<transport type="TCP" socket-binding="jgroups-tcp"/>

<protocol type="DNS\_PING"/>

<protocol type="MERGE3"/>

<!-- more protocols -->

</stack>

</subsystem>

**📘 Example: Elytron Security in EAP 8**

<subsystem xmlns="urn:jboss:domain:elytron:8.0">

<security-domains>

<security-domain name="ApplicationDomain" default-realm="ApplicationRealm">

<realm name="ApplicationRealm"/>

</security-domain>

</security-domains>

</subsystem>

**🔍 References**

You can explore full configuration guides and examples from Red Hat’s official documentation:

* [JBoss EAP 7.3 High Availability Configuration Guide](https://docs.redhat.com/en/documentation/red_hat_jboss_enterprise_application_platform/7.1/html/configuration_guide/configuring_high_availability)
* [JBoss EAP 8 High Availability Configuration Guide](https://docs.redhat.com/en/documentation/red_hat_jboss_enterprise_application_platform/8.0/html/configuration_guide/configuring_high_availability)

Would you like me to generate a diff tool or script to compare two actual XML files line-by-line? Or help you customize a standalone-ha.xml for your specific cluster setup?

import xml.etree.ElementTree as ET

from collections import defaultdict

def parse\_xml(file\_path):

tree = ET.parse(file\_path)

root = tree.getroot()

return root

def flatten\_xml(element, path=""):

items = defaultdict(list)

tag = element.tag.split("}")[-1] # Remove namespace

current\_path = f"{path}/{tag}"

items[current\_path].append(element.attrib)

for child in element:

child\_items = flatten\_xml(child, current\_path)

for k, v in child\_items.items():

items[k].extend(v)

return items

def compare\_dicts(dict1, dict2):

all\_keys = set(dict1.keys()).union(dict2.keys())

for key in sorted(all\_keys):

if key not in dict1:

print(f"🟢 Added in new XML: {key}")

elif key not in dict2:

print(f🔴 Removed in new XML: {key}")

elif dict1[key] != dict2[key]:

print(f"🟡 Modified: {key}")

print(f" Old: {dict1[key]}")

print(f" New: {dict2[key]}")

def compare\_xml\_files(file1, file2):

root1 = parse\_xml(file1)

root2 = parse\_xml(file2)

flat1 = flatten\_xml(root1)

flat2 = flatten\_xml(root2)

compare\_dicts(flat1, flat2)

# Example usage

compare\_xml\_files("jboss-7.3-standalone-ha.xml", "jboss-8-standalone-ha.xml")

Perfect! Since you're working in a modular, automation-friendly environment, here's a Python-based diff tool tailored for comparing two standalone-ha.xml files—ideal for JBoss EAP 7.3 vs 8. It highlights structural differences, subsystem changes, and attribute updates.

**🛠️ XML Diff Tool for JBoss standalone-ha.xml**

**🔧 Features**

* Compares two XML files element-by-element
* Highlights added, removed, and modified nodes
* Supports namespace-aware parsing
* Outputs a structured diff report

**📄 Python Script**

import xml.etree.ElementTree as ET

from collections import defaultdict

def parse\_xml(file\_path):

tree = ET.parse(file\_path)

root = tree.getroot()

return root

def flatten\_xml(element, path=""):

items = defaultdict(list)

tag = element.tag.split("}")[-1] # Remove namespace

current\_path = f"{path}/{tag}"

items[current\_path].append(element.attrib)

for child in element:

child\_items = flatten\_xml(child, current\_path)

for k, v in child\_items.items():

items[k].extend(v)

return items

def compare\_dicts(dict1, dict2):

all\_keys = set(dict1.keys()).union(dict2.keys())

for key in sorted(all\_keys):

if key not in dict1:

print(f"🟢 Added in new XML: {key}")

elif key not in dict2:

print(f🔴 Removed in new XML: {key}")

elif dict1[key] != dict2[key]:

print(f"🟡 Modified: {key}")

print(f" Old: {dict1[key]}")

print(f" New: {dict2[key]}")

def compare\_xml\_files(file1, file2):

root1 = parse\_xml(file1)

root2 = parse\_xml(file2)

flat1 = flatten\_xml(root1)

flat2 = flatten\_xml(root2)

compare\_dicts(flat1, flat2)

# Example usage

compare\_xml\_files("jboss-7.3-standalone-ha.xml", "jboss-8-standalone-ha.xml")

**📦 Requirements**

* Python 3.7+
* Two XML files: jboss-7.3-standalone-ha.xml and jboss-8-standalone-ha.xml

**🧪 Sample Output**

🟢 Added in new XML: /server/profile/subsystem/jgroups/stack/kubernetes

🟡 Modified: /server/profile/subsystem/infinispan/cache-container/server

Old: [{'default-cache': 'default'}]

New: [{'default-cache': 'default', 'persistent': 'true'}]

🔴 Removed in new XML: /server/profile/subsystem/jgroups/stack/udp

Would you like me to help you generate a baseline standalone-ha.xml for JBoss 8 based on your current 7.3 config? I can also modularize it for easier automation and version control.

Modular JBoss EAP 8 standalone-ha.xml (Baseline Template)

false



<server xmlns="urn:jboss:domain:15.0">

<!-- 🔌 Extensions -->

<extensions>

<extension module="org.jboss.as.clustering.infinispan"/>

<extension module="org.jboss.as.jgroups"/>

<extension module="org.jboss.as.messaging"/>

<extension module="org.jboss.as.ejb3"/>

<extension module="org.jboss.as.undertow"/>

<extension module="org.wildfly.extension.elytron"/>

<extension module="org.jboss.as.security"/>

<extension module="org.jboss.as.transactions"/>

<extension module="org.jboss.as.logging"/>

<extension module="org.jboss.as.naming"/>

<extension module="org.jboss.as.connector"/>

<extension module="org.jboss.as.webservices"/>

<extension module="org.wildfly.extension.batch.jberet"/>

<extension module="org.jboss.as.mail"/>

<extension module="org.jboss.as.remoting"/>

<extension module="org.jboss.as.modcluster"/>

</extensions>

<!-- ⚙️ Management -->

<management>

<security-realms>

<security-realm name="ManagementRealm">

<authentication>

<local default-user="$local" skip-group-loading="true"/>

</authentication>

</security-realm>

</security-realms>

<management-interfaces>

<http-interface security-realm="ManagementRealm">

<http-upgrade enabled="true"/>

<socket-binding http="management-http"/>

</http-interface>

</management-interfaces>

</management>

<!-- 🔗 Socket Bindings -->

<socket-binding-group name="standard-sockets" default-interface="public">

<socket-binding name="management-http" port="9990"/>

<socket-binding name="http" port="8080"/>

<socket-binding name="https" port="8443"/>

<socket-binding name="jgroups-tcp" port="7600"/>

<socket-binding name="messaging" port="5445"/>

<!-- Add more bindings as needed -->

</socket-binding-group>

<!-- 🧩 Profile -->

<profile>

<!-- 🧠 JGroups -->

<subsystem xmlns="urn:jboss:domain:jgroups:7.0" default-stack="kubernetes">

<stack name="kubernetes">

<transport type="TCP" socket-binding="jgroups-tcp"/>

<protocol type="DNS\_PING"/>

<protocol type="MERGE3"/>

<protocol type="FD\_SOCK"/>

<protocol type="FD\_ALL"/>

<protocol type="VERIFY\_SUSPECT"/>

<protocol type="pbcast.NAKACK2"/>

<protocol type="UNICAST3"/>

<protocol type="pbcast.STABLE"/>

<protocol type="pbcast.GMS"/>

<protocol type="MFC"/>

<protocol type="FRAG2"/>

</stack>

</subsystem>

<!-- 🗃️ Infinispan -->

<subsystem xmlns="urn:jboss:domain:infinispan:14.0">

<cache-container name="server" default-cache="default">

<distributed-cache name="default" persistent="true"/>

</cache-container>

<cache-container name="web" default-cache="passivation">

<replicated-cache name="passivation"/>

</cache-container>

<cache-container name="ejb" default-cache="clustered">

<replicated-cache name="clustered"/>

</cache-container>

<cache-container name="hibernate" default-cache="local-query">

<local-cache name="local-query"/>

</cache-container>

</subsystem>

<!-- 📬 Messaging -->

<subsystem xmlns="urn:jboss:domain:messaging-activemq:6.0">

<server name="default">

<http-connector name="http-connector" socket-binding="messaging"/>

<security-enabled>false</security-enabled>

</server>

</subsystem>

<!-- 🧱 EJB3 -->

<subsystem xmlns="urn:jboss:domain:ejb3:10.0">

<cluster-passivation-store name="default"/>

<thread-pools>

<thread-pool name="default"/>

</thread-pools>

</subsystem>

<!-- 🌐 Undertow -->

<subsystem xmlns="urn:jboss:domain:undertow:13.0">

<server name="default-server">

<http-listener name="default" socket-binding="http"/>

<host name="default-host" alias="localhost"/>

</server>

</subsystem>

<!-- 🔐 Elytron -->

<subsystem xmlns="urn:jboss:domain:elytron:8.0">

<security-domains>

<security-domain name="ApplicationDomain" default-realm="ApplicationRealm">

<realm name="ApplicationRealm"/>

</security-domain>

</security-domains>

</subsystem>

<!-- 🧾 Logging -->

<subsystem xmlns="urn:jboss:domain:logging:15.0">

<console-handler name="CONSOLE">

<level name="INFO"/>

<formatter>

<named-formatter name="PATTERN"/>

</formatter>

</console-handler>

<root-logger>

<level name="INFO"/>

<handlers>

<handler name="CONSOLE"/>

</handlers>

</root-logger>

</subsystem>

<!-- 🧪 Transactions -->

<subsystem xmlns="urn:jboss:domain:transactions:7.0">

<core-environment>

<process-id>

<uuid/>

</process-id>

</core-environment>

</subsystem>

<!-- 📡 Remoting -->

<subsystem xmlns="urn:jboss:domain:remoting:5.0">

<connector name="http-remoting-connector" socket-binding="http"/>

</subsystem>

<!-- 📨 Mail -->

<subsystem xmlns="urn:jboss:domain:mail:4.0">

<mail-session name="default" jndi-name="java:jboss/mail/Default">

<smtp-server outbound-socket-binding-ref="mail-smtp"/>

</mail-session>

</subsystem>

<!-- 🧮 Batch -->

<subsystem xmlns="urn:jboss:domain:batch-jberet:3.0">

<default-job-repository name="in-memory"/>

</subsystem>

<!-- 🧼 Web Services -->

<subsystem xmlns="urn:jboss:domain:webservices:4.0">

<endpoint-config name="Standard-Endpoint-Config"/>

</subsystem>

</profile>

<!-- 🔍 Deployment Scanner -->

<deployments>

<deployment-scanner path="deployments" relative-to="jboss.server.base.dir" scan-interval="5000"/>

</deployments>

</server>

🧠 Tips for Customization

* **Clustering**: If you're not on Kubernetes, switch jgroups stack to udp or tcp.
* **Security**: Customize Elytron realms and domains for your auth needs.
* **Messaging**: Add queues/topics under messaging-activemq if needed.
* **Datasources**: Add JDBC/XA datasources under datasources subsystem.
* **Monitoring**: Add metrics or microprofile subsystems if you're integrating with Prometheus/Grafana.

Would you like me to modularize this into separate YAML or JSON blocks for automation pipelines? Or generate a script to convert your existing 7.3 config into this 8.0 format?

JBoss 7.3 → 8.0 XML Conversion Script

🔧 Features

* Updates schema version and namespaces
* Replaces JGroups stack with Kubernetes-ready config
* Adds Elytron if missing
* Converts EJB and Web Services subsystems to Jakarta EE
* Preserves custom socket bindings, deployments, and datasources

import xml.etree.ElementTree as ET

from xml.dom import minidom

# Namespace mappings

NS\_MAP = {

"server": "urn:jboss:domain:15.0",

"jgroups": "urn:jboss:domain:jgroups:7.0",

"infinispan": "urn:jboss:domain:infinispan:14.0",

"elytron": "urn:jboss:domain:elytron:8.0",

"ejb3": "urn:jboss:domain:ejb3:10.0",

"webservices": "urn:jboss:domain:webservices:4.0",

"undertow": "urn:jboss:domain:undertow:13.0",

"logging": "urn:jboss:domain:logging:15.0",

"transactions": "urn:jboss:domain:transactions:7.0",

"mail": "urn:jboss:domain:mail:4.0",

"remoting": "urn:jboss:domain:remoting:5.0",

"batch": "urn:jboss:domain:batch-jberet:3.0",

"messaging": "urn:jboss:domain:messaging-activemq:6.0"

}

def update\_namespace(elem, new\_ns):

tag = elem.tag.split("}")[-1]

elem.tag = f"{{{new\_ns}}}{tag}"

def convert\_subsystems(profile):

for subsystem in profile.findall("./\*"):

tag = subsystem.tag.split("}")[-1]

if tag in NS\_MAP:

update\_namespace(subsystem, NS\_MAP[tag])

# Replace JGroups stack

if tag == "jgroups":

subsystem.set("default-stack", "kubernetes")

for stack in subsystem.findall(".//stack"):

stack.set("name", "kubernetes")

for protocol in stack.findall("protocol"):

if protocol.get("type") == "PING":

protocol.set("type", "DNS\_PING")

# Add Elytron if missing

if "elytron" not in [s.tag.split("}")[-1] for s in profile]:

elytron = ET.Element(f"{{{NS\_MAP['elytron']}}}subsystem")

sd = ET.SubElement(elytron, "security-domains")

domain = ET.SubElement(sd, "security-domain", {

"name": "ApplicationDomain",

"default-realm": "ApplicationRealm"

})

ET.SubElement(domain, "realm", {"name": "ApplicationRealm"})

profile.append(elytron)

def convert\_server(file\_in, file\_out):

tree = ET.parse(file\_in)

root = tree.getroot()

# Update root namespace

update\_namespace(root, NS\_MAP["server"])

# Convert subsystems

profile = root.find("./profile")

convert\_subsystems(profile)

# Pretty print and write

xml\_str = ET.tostring(root, encoding="unicode")

pretty\_xml = minidom.parseString(xml\_str).toprettyxml(indent=" ")

with open(file\_out, "w") as f:

f.write(pretty\_xml)

# Example usage

convert\_server("jboss-7.3-standalone-ha.xml", "jboss-8.0-standalone-ha.xml")

equirements

* Python 3.8+
* Input file: jboss-7.3-standalone-ha.xml
* Output file: jboss-8.0-standalone-ha.xml

🧪 What It Does

* Renames all subsystem namespaces to match EAP 8
* Replaces PING with DNS\_PING in JGroups
* Adds Elytron security domain if missing
* Leaves your socket bindings, datasources, deployments untouched