Running Oracle WebLogic on containers

Borja Aparicio Cotarelo¹, Antonio Nappi¹, Lukas Gedvilas², Luis Rodríguez Fernández¹, Theodoros Rizopoulos¹ and Artur Wiecek¹

¹IT Department – CERN – Meyrin – Geneva – Switzerland
²Servicio de Relaciones Internacionales – Universidad Politécnica de Madrid – Comunidad de Madrid – Madrid – España

Abstract. CERN IT department is providing production services to run containers. Given that, the IT-DB team, responsible to run the Java based platforms, has started a new project to move the WebLogic deployments from virtual or bare metal servers to containers: Docker on Kubernetes allow to improve the overall productivity of the team by reducing both operations time and time-to-delivery. Additionally, this framework allows us deploy our services in a fully-reproducible fashion. The scope of the project goes from the design and production of Docker images, deployment environments based on Kubernetes as well as all procedures and operations including the needed tools to hand over to users the management of their deployed applications. This article illustrates how at CERN we have faced all the technical and design aspects to run it in a production environment. That means the implementation of the solutions needed including monitoring, logging, high availability, security and traceability.

1 CERN Database and Application Server infrastructure

CERN, the European Laboratory for Particle Physics, concentrates thousands of scientists, engineers, technicians and administrative stuff among others in a highly technological environment. A natural consequence of this environment is the need of IT services to cover activities that go from the DAQ (Data Acquisition) from the physics experiments and post analysis, to handling invoices or producing the access badges for the authorized personnel requiring physical access into the CERN sites.

A key element in that catalogue of services are the web based services: There are over 800 servers that expose port 443 (HTTPS) to the Internet. These web services can be classified in two groups: small/medium-sized general purpose applications and critical enterprise applications that require highly-available platforms. Some statistics can be found in [1].

Focusing in the enterprise applications, most of them are the Java-based web applications deployed on top of the platform provided by “Database and Application Server infrastructure” team, part of the IT Department.

The majority of these applications are deployed on clusters of Oracle WebLogic Servers [2]. However, there are a number of applications with low resource requirements which are deployed on single-node Apache Tomcat-based application servers [3].

1.1 Delivery model

The delivery model for this service requires supporting the needs of both production and testing environments. Production environments need to be regularly secured and updated, while testing environments need to support the needs of developers. The service also allows developers and operators the interfaces to the environments to perform actions like restart,
deploy or un-deploy applications. Additionally, the service allows access to logs and metrics for auditing, debugging and monitoring purposes.

At CERN there are three main communities that provide enterprise Java applications. Those three communities are independent and serve both in-house developed applications and commercial products. The environments provided must be isolated so no information leaks can happen and must meet the requirements of each application.

Four different environments have to be provided:
- Development: reduced amount of resources used by developers to implement new applications, change requests or bug fixes.
- Test: reduced amount of resources used to validate and integrate the changes performed in the development phase. The test and development environments are the first to receive software updates.
- Preproduction: replica of production with a reduced amount of resources and dedicated to integration tests. Applications deployed here point to the same data sources as production and considered the latest step before a production deployment.
- Production: Provides the highest level of availability and the resources needed to deliver optimal performance.

Table 1 show the relation of URLs (applications) configured and served by the team classified by its flavour:

| Production | Preproduction | Test | Development |
|------------|---------------|------|-------------|
| ~160       | ~85           | ~100 | ~150        |

### 1.2 Deployment model

The CERN IT department offers a private cloud based on OpenStack and a centralized configuration management system based on Puppet. Both services offer tools for IT service managers to create and manage virtual machines (VMs).

The CERN cloud is organized in three availability zones (AZs) a service manager can select where a new VM instantiated. This allows to distribute services across AZs and to remain available during power, hardware or network failures.

The logical unit in the Oracle WebLogic application server [4] is the domain. A domain groups several WebLogic Server instances isolating resources (Figure 1).

Each domain is dedicated to a set of applications belonging a community of users with the same level of criticality, allowing separation of production and non-production applications. This enables the Oracle WebLogic administrators to perform interventions or updates at either host or domain level, affecting only a specific set of applications.

Within a single domain, applications runs in several WebLogic servers distributed across at least three VMs running in different AZs. The maximum number of WebLogic servers dedicated to run a single application depends on the resources needed for that application to cope with the load and community of users that it serves. The size (in terms of memory allocated) for each WebLogic server depends on the complexity of the application that it runs and its demands of resources.
deploy or un-deploy applications. Additionally, the service allows access to logs and metrics for auditing, debugging and monitoring purposes.

At CERN there are three main communities that provide enterprise Java applications. Those three communities are independent and serve both in-house developed applications and commercial products. The environments provided must be isolated so no information leaks can happen and must meet the requirements of each application.

Four different environments have to be provided:

- **Development**: reduced amount of resources used by developers to implement new applications, change requests or bug fixes.
- **Test**: reduced amount of resources used to validate and integrate the changes performed in the development phase. The test and development environments are the first to receive software updates.
- **Preproduction**: replica of production with a reduced amount of resources and dedicated to integration tests. Applications deployed here point to the same data sources as production and considered the latest step before a production deployment.
- **Production**: provides the highest level of availability and the resources needed to deliver optimal performance.

Table 1 shows the relation of URLs (applications) configured and served by the team classified by its flavour:

| Flavour  | URL Served |
|----------|------------|
| Production | ~160       |
| Preproduction | ~85       |
| Test       | ~100       |
| Development | ~150       |

1.2 Deployment model

The CERN IT department offers a private cloud based on OpenStack and a centralized configuration management system based on Puppet. Both services offer tools for IT service managers to create and manage virtual machines (VMs).

The CERN cloud is organized in three availability zones (AZs) a service manager can select where a new VM is instantiated. This allows to distribute services across AZs and to remain available during power, hardware or network failures.

The logical unit in the Oracle WebLogic application server [4] is the domain. A domain groups several WebLogic Server instances isolating resources (Figure 1).

Each domain is dedicated to a set of applications belonging to a community of users with the same level of criticality, allowing separation of production and non-production applications. This enables the Oracle WebLogic administrators to perform interventions or updates at either host or domain level, affecting only a specific set of applications.

Within a single domain, applications run in several WebLogic servers distributed across at least three VMs running in different AZs. The maximum number of WebLogic servers dedicated to run a single application depends on the resources needed for that application to cope with the load and community of users it serves. The size (in terms of memory allocated) for each WebLogic server depends on the complexity of the application it runs and its demands of resources.

![Fig. 1. Deployment model of a WebLogic domain on virtual machines](image)

The clusters are the WebLogic logical units, within a domain, that group together a set of WebLogic servers. A single cluster may run one or several applications. In this case, each cluster runs a single application and is distributed across multiple VMs to maximize availability. This model allows management of each application independently (deploy, undeploy, upgrade, downgrade) and delegate that management to developers that can work independently.

![Fig. 2. Example of representation of WebLogic domain in an LDAP directory](image)

The definition and specific configuration of the WebLogic domains are persistent in an LDAP directory (see Fig. 2). This information is converted into JSON format in order to be used to instantiate, recreate or update the configuration of the managed domains.

Finally, the last element is the web frontend which receives user requests and redirects them. These front-ends also use the same grouping criteria as WebLogic domains: different Apache web servers [5] are running in several VMs to avoid single points of failure. They are grouped together to serve applications of the same community and with the same criticality (see Table 1). The bridge between web server and application server is done with the WebLogic Apache plugin [6] which redirects each user request to the needed WebLogic cluster. The definition and configuration of the different Apache web servers is stored in LDAP. That information together with a set of scripts, is used to generate the server configurations.

1.3 Limitations
This model of service delivery achieves several optimizations and advantages. A single technical group is responsible for the platform where applications run, which is equivalent regardless the application. It also liberates the different groups of developers and responsible of applications from the delivery and maintenance of the underlying platform and allows them to focus only in the final application.

On the other hand, there are several factors that limit the capacity of the team to keep the service level agreement and at the same time, reply in time to the increasing number of requests of new environments: the process of delivering a new environment requires many semi-automated steps like allocation of VMs, definition and configuration of environments, etc, which are time consuming and error prone.

Once an environment delivered in this model is running, it is a difficult process to identify whether a given error is originated in the application or in the platform where it runs as environments degrade over time (updates, configuration changes, mistakes, etc).

Finally, the upgrade and patching process when an update or security patch is delivered is a very time consuming process that, with a growing number of environments, increases the delivery time of the team.

2 Containers-based deployment

With the environment described above, the team has looked for a technological evolution to achieve the following:

- Reduce the overall cost in time to deliver a new cluster(s)
- Improve the immutability of environments and facilitate the replication of environments
- Ease the patch and upgrade process
- Modular architecture with isolated modules
- Version-control for the system
- Change management workflow

Software containers have been strongly adopted by the industry in the last decade. Containers allow reproducible packaging by abstracting away the operating system layer, hence removing the dependency between the software and the platform where it runs, key fact to achieve portability.

2.1 Docker and Kubernetes

Docker is currently the reference in containers technology and has become an industry standard. As described in [7]: “A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.”. The portability aspect of the containers technology eliminates most of the deployment cost that currently means to provide a new WebLogic cluster. VMs do not have to be provisioned and configured to run WebLogic applications specifically. VMs become general worker nodes where Kubernetes clusters and Docker containers can run regardless what the container image implements.

In order to implement a full WebLogic domain based on container, several containers need to be running. Each container implements a specific function in a modular approach and all of them rely on an orchestration tool to work together.

Kubernetes [8] is the leading orchestration technology: “Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and
discovery”. Kubernetes provides the features to start the domains and restart automatically in case of failure. By using Kubernetes and Docker container images, a full system was designed to define and deploy WebLogic domains and handle them in all its life cycle.

2.2 CERN Infrastructure and Oracle vision

The CERN cloud offering was expanded in January 2018 [9] to offer a production Container-Engine-as-a-Service based on Openstack Magnum. This service allows CERN service managers to create, amongst others, Kubernetes clusters and to deploy services on them.

From January 2018, Oracle certifies the use of WebLogic service in Kubernetes and Docker containers [10] and provides examples and recipes on how to create WebLogic Docker images and deploy Kubernetes clusters that run WebLogic domains. This gave the go ahead for the team to develop the project and tools needed to port the managed set of WebLogic domains to Docker/Kubernetes.

3 WebLogic domains in Kubernetes

3.1 Service and deployment model

The design of the system and deployment method had to be done taken into account different factors to ensure that the team can achieve the defined scale out objectives.

The first design decision was to deploy one WebLogic domain with a single cluster in the different AZs for each application so customizations needed for each domain are minimized and the same template of WebLogic domain can be used for creating multiple instances therefore simplifying the deployment method. Then, Kubernetes namespaces [11] will be created to deploy each WebLogic domain. This allows to isolate the applications. Also, applications that belong to different communities of users or with different level of criticality will be deployed in separate Kubernetes clusters so they are kept private.

Another important factor in this design is to have the possibility port or migrate the WebLogic domains to different Kubernetes clusters provided by any cloud service. For this, cloud-specific features are avoided. For example, the need to persist the deployed application even if the domain is restarted: the applications, instead of stored in persistent storage, will be downloaded from an external REST service when the environment is starting up. Also the logs produced by the applications and monitoring metrics are shipped out of the cluster to external resources like ElasticSearch [12] for logs and InfluxDB [26] for metrics.

Finally, when working with containers, all the elements of the project, including tools, software and the infrastructure itself, can be defined as code. GitLab [13] is used for version control and deployment tools have been implemented to instantiate the environments. GitLab and its Continuous Integration (CI) pipelines are used for building Docker images. For issue tracking, Atlassian Jira [14] is used. Together these elements allow us to go from a change request or bug report, to a change in the code, review by peers, approval and production of new Docker images ready to be deployed.

Each Kubernetes namespace [11] has been designed with a modular approach. Each component behaves like a microservice providing a specific functionality. This allows each
component to be replaced without affecting the rest, and the overall functioning of the system. Below we list the different components that will run in each namespace describing its function:

- **Oracle WebLogic pods**: running Oracle WebLogic server. Apart from the server software, these pods also run a Filebeat [16] process that consumes the log output generated by the WebLogic server and send out of the pod to a Logstash [18] and a JVM [23] agent that export Java metrics.

- **Apache Derby [17]** pod: Is used as an RDBMS security store for session information.

- **Logstash pod**: gathers the logs generated by the rest of pods, shipped by Filebeat and sends them to ElasticSearch.

- **Telegraf [19]**: pulls from each WebLogic server pod the JMX metrics exposed for monitoring. It sends them on to InfluxDB time series database for later visualization with Grafana [24].

- **HAProxy Ingress [20]** controller: entry point for the namespace. It redirects the web request to the different pods and, using liveness probes, it will automatically blacklist a pod that is in error state.

### Oracle WebLogic docker image:

The main component of the described above is the Oracle WebLogic pod that runs an Oracle WebLogic docker image designed in layers. The build process of the image is automated using GitLab CI pipelines and driven by code changes pushed to GitLab repositories. The final design is organized in two images: base and server. The WebLogic base image is built from the default operating system image plus the needed administration tools and the WebLogic server software. It includes Jolokia [22], the Derby database client and Filebeat. The Dockerfile for this image uses environment variables to choose the version of WebLogic to build. This permits us to make several images from a single Dockerfile. Part of the build process consist in creating the WebLogic Pack [21] in an intermediate pipeline step. This object contains the structure of the WebLogic domain and used to build the server image. Finally, the WebLogic server image is built based on the WebLogic base image and includes the WebLogic pack object and a custom script to start the WebLogic server. This script runs when the container is started and sources the environment injected to customize WebLogic, deploys the pack and finally, starts the WebLogic application server.

### Logging and monitoring:

Logs and metrics have to be shipped out of the containers and made available to developers. Each pod running a WebLogic server runs Filebeat, responsible for sending the logs generated to the Logstash pod of the namespace. Logstash processes these logs and sends them to a specific tenant in an ElasticSearch instance. This ES instance is exposed to developers for visualizing the data using Kibana.

About metrics, a similar approach has been taken: each pod running WebLogic server also runs the Jolokia agent. The agent consumes JMX metrics that are produced by the JVM running WebLogic. Then, a Telegraf pod pulls those metrics and forwards them to an external instance of InfluxDB. Similar to the ElasticSearch case, the external InfluxDB is a central instance with independent schemas dedicated to specific community of users.
ensuring the privacy of the data collected. Then, using Grafana dashboards, developers are able to visualize the metrics generated by their applications.

**Platform deployment toolset:**

Each environment to be deployed is defined in the LDAP directory. Scripts using this definition, connect to the Kubernetes cluster and create the Kubernetes object descriptions needed for all elements of the WebLogic domain to run in the Kubernetes cluster.

**Web frontend:**

A highly available web front-end based on HAProxy [25], listens to the user requests and redirects them to the appropriate Kubernetes clusters where the requested application is running.

4 **Conclusions**

Virtualization technologies have given good results, stability and have free up platform providers from administering the hardware. However, this deployment model is still a limiting factor to scale while offering the same level of availability.

Containers are well stablish in the industry as technology to deploy applications, eliminates the overhead of provisioning and configuring virtual machines and has been adopted and is fully supported by Oracle, company owner of the WebLogic application server. In this context, the team has defined and implemented a whole new system based on containers to provide to the application developers the WebLogic environments where to deploy and run applications. At the time of writing this article, a number of development and test application have been already migrated and the system is in validation phase with the users.

The benefits of this new deployment method also requires an important change in the working habits of the IT specialists working with the systems: environments are no longer unique entities with a life cycle, but instances of a software defined system that can be destroyed, recreated, replicated or ported to new hardware.

**References**

1. https://webservices.web.cern.ch/webservices/ServiceStatus/ [accessed 2018-09-10]
2. Oracle WebLogic server, “WLS” [software], version 12.1.3 and 12.2.1.2, 2018. Available from https://www.oracle.com/middleware/technologies/weblogic.html [accessed 2018-09-10]
3. http://tomcat.apache.org/ [accessed 2018-09-10]
4. https://docs.oracle.com/cd/E13222_01/wls/docs81/adminguide/overview_domain.html [accessed 2018-09-10]
5. https://httpd.apache.org/ [accessed 2018-09-10]
6. Oracle WebLogic Server Proxy Plug-in for Apache HTTP Server, “WLS Apache plugin” [software], version 12.1.1, 2018. Available from https://docs.oracle.com/middleware/1221/webtier/develop-plugin/apache.html/ [accessed 2018-09-10]
7. https://www.docker.com/resources/what-container [accessed 2018-09-10]
8. Kubernetes orchestration engine, “Kubernetes” [software], version 1.7.6, 2018. Available from https://kubernetes.io [accessed 2018-09-10]

9. http://openstack-in-production.blogspot.com/2017/01/containers-on-cern-cloud.html [accessed 2018-09-10]

10. https://blogs.oracle.com/weblogicserver/weblogic-server-certification-on-kubernetes [accessed 2018-09-10]

11. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ [accessed 2018-09-10]

12. https://www.elastic.co/products/elasticsearch [accessed 2018-09-10]

13. https://about.gitlab.com/ [accessed 2018-09-10]

14. https://www.atlassian.com/software/jira [accessed 2018-09-10]

15. https://kubernetes.io/docs/concepts/workloads/pods/pod/ [accessed 2018-09-10]

16. Filebeat – Lightweight Shipper for Logs, “Filebeat” [software], version 6.5.0, 2018. Available from https://www.elastic.co/products/beats/filebeat [accessed 2018-09-10]

17. Apache Derby, “Derby” [software], version 10.10.1.1, 2018. Available from https://db.apache.org/derby/ [accessed 2018-09-10]

18. Logstash – Centralize, transform and & Stash Your Data, “Logstash” [software] version 6.5.0, 2018. Available from https://www.elastic.co/products/logstash [accessed 2018-09-10]

19. Telegraf agent, “Telegraf” [software] version 1.8.3, 2018. Available from https://www.influxdata.com/time-series-platform/telegraf/ [accessed 2018-09-10]

20. HAProxy Ingress Controller for Kubernetes, “HAProxy Ingress” [software] version 0.6.1, 2018. Available from https://www.haproxy.com/blog/haproxy_ingress_controller_for_kubernetes/ [accessed 2018-09-10]

21. https://docs.oracle.com/cd/E13179_01/common/docs92/pack/commands.html [accessed 2018-09-10]

22. Jolokia JMX-HTTP bridge, “Jolokia” [software] version 1.6.0, 2018. Available from https://jolokia.org/ [accessed 2018-09-10]

23. Java Virtual Machine, “JVM” [software] version 1.8.0_192, 2018. Available from https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html [accessed 2018-09-10]

24. https://grafana.com/ [accessed 2018-09-10]

25. http://wwwhaproxy.org/ [accessed 2018-09-10]

26. https://www.influxdata.com/time-series-database/ [accessed 2018-09-10]