A Business Guide To Cloud Deployment Options For Jakarta EE Applications

The Payara® Platform - Production-Ready,
Cloud Native and Aggressively Compatible.

User Guide

Contents

Guide Updated: **August 2023**

The Java Platform has been the platform of choice for enterprise application development for a lot of developers over the last two and half decades. There is no shortage of frameworks and platforms for developing all kinds of software applications using the Java Programming Language. One such platform that has stood the test of time is Jakarta EE.

This guide takes a look at various production deployment options for Jakarta EE applications. It starts off by taking a brief look at the theoretical foundations of Jakarta EE, followed by a look at the Jakarta EE programming model. It then takes a look at the various production deployment options available for a typical Jakarta EE application. By the end of this guide, you should have a good grasp of the deployment options available to you for your next Jakarta EE project.

What is Jakarta EE?

Jakarta EE is a set of community developed, abstract specifications that together form a platform for developing end-to-end, multi-tier enterprise applications. Jakarta EE is built on the Java Standard Edition, and aims to provide a stable, reliable and vendor neutral platform on which to develop cloud native applications.

Hitherto, Jakarta EE was called Java EE and was a property of Oracle Inc., evolved through the Java Community Process (JCP). However, in late 2017, Oracle decided to move the platform to an open foundation for a much broader community-led evolution. The Eclipse Foundation got chosen and Java EE, after the transfer, got rebranded to Jakarta EE.

What Is a Specification?

As stated in the above definition, Jakarta EE is made up of a set of specifications that each cover a specific API for solving a specific software development need. For example, the Jakarta Contexts and Dependency Injection (CDI) specification provides constructs for creating loosely coupled applications through dependency injection. These different specifications are combined into a single "umbrella" specification for each Jakarta EE release. As such, Jakarta EE 10 for instance, is released under the Jakarta EE 10 specification.

More technically, a specification is a formal proposal document made to the Jakarta EE Specification Committee through the Jakarta EE Specification Process (JESP) that outlines the functions of a given set of APIs. This document outlines what the expected behaviour should be for various invocations of the API. The specification then acts as the blueprint for the API.

What Is a Compatible Implementation?

As a specification is merely a document that outlines the behaviour of a given API, it needs an implementation that realises the actual outcomes for each invocation of the API. For instance, the Jakarta Persistence specification provides the EntityManager interface that has the persist() method. This method, when called and passed an instance of a Jakarta Persistence entity, persists that entity instance as a database row to the underlying database. The "library" that does the actual work of taking that instance and making sure it gets stored to the durable storage when the EntityManager#persist() method is invoked, is called a compatible implementation of the Jakarta Persistence specification.

Each specification that makes up the full Jakarta EE platform has an implementation. As a specification itself, the Jakarta EE platform also has an implementation in the form of compatible products. As the specifications are separated from their implementations, you as a developer will generally code against the API constructs of the specification, and are free to pick any compatible implementation of the platform. With this abstraction, Jakarta EE implementation vendors can collaborate on the base, standard specifications and compete through innovations on top of the base platform.

An example of such invocation is the Payara Cloud offering from Payara. This innovation helps you realise the dream of true separation of your business domain application and the runtime that powers it. With Payara Cloud, you simply upload your Jakarta EE application web archive (.war file) and have it automatically deployed to the cloud, just as Jakarta EE was envisaged to have separation of business domain from the runtime. Another example of custom features available on the Payara Platform is remote CDI events. This feature, built on the Jakarta CDI specification, allows the firing of CDI events that can be observed by any listener in a given Hazelcast cluster.

What is Eclipse MicroProfile?

The Jakarta EE Platform is a general purpose platform for developing all kinds of applications. As modern application development paradigms have changed a lot in the past years, there is a need to evolve the platform to meet such changes. One such paradigm is cloud-native software application development.

As the base Jakarta EE Platform has always been geared towards enterprises, it has historically evolved at a much slower pace than changes in the software development space. It is for this reason that the Eclipse MicroProfile project was created as an extension to the base platform to provide cloud-native APIs for developing modern cloud-based applications.

Eclipse MicroProfile, built upon Jakarta CDI, Jakarta REST and Jakarta JSON Processing, comes with the following APIs

- OpenTracing
- OpenAPI
- REST Client
- Config
- Fault Tolerance
- Metrics
- JWT Propagation
- Health

These APIs augment the much larger Jakarta EE Platform APIs to provide the developer with a cohesive set of APIs for developing, testing and deploying cloud-native modern enterprise applications.

Jakarta EE Programming Model

The programming model of Jakarta EE follows the general software development lifecycle process of requirements gathering and planning, design and/or prototyping, actual development, testing, deployment and maintenance. Different companies use different combinations of these steps. However, development, testing and deployment are steps that almost every company's application development process will entail.

Development

Where Jakarta EE differs significantly is the separation of runtime from application code. Historically, Jakarta EE has always been a platform that encourages the separation of platform runtime from business code. This way, your concern with using the platform should be delivering business value. This separation of runtime from application code is demonstrated in the universal way of adding the Jakarta EE dependency to your project as shown in Listing 1.

Listing 1

```
      <dependency>
                  <groupId>jakarta.platform</groupId>
                  <artifactId>jakarta.jakartaee-api</artifactId>
                  <version>10.0.0</version>
                  <scope>provided</scope>
            </dependency>
```

The scope of the dependency is set to provided. This means the application expects the runtime to provide implementations of all the specifications that are part of the given release, in this case version 10. The resulting project artefact will not include any implementations of the Jakarta EE specifications. This separation of the application runtime from business code allows for true separation of concerns, and also makes it easy to have different implementations of the specification without needing to change or repackage application code.

Testing

Every non-trivial enterprise application will require testing. There are different types and combinations of testing that can be employed to assure a certain level of quality for software applications. Most applications will employ a combination of unit, integration and acceptance testing.

Deployment

There are different options for deploying Jakarta EE applications, ranging the simplest to the most complicated. The next sections of this guide go into detail about the various options available to you as a Jakarta EE developer.

Jakarta EE Deployment Options

Application deployment can be defined as the process of making an application or collection of applications that solves a particular problem available to users with that problem. This definition may sound straightforward, but application deployment is a long process that starts with testing. A typical application will have a suite of tests that run on a CI/CD pipeline. This pipeline will then produce an artefact that, depending on the specific configuration, will be transformed into a container image or directly uploaded to the actual deployment server. Before we look at the various deployment options, let us first consider some application deployment techniques out there.

Application Deployment Techniques

An application deployment technique is the actual strategy used to roll out different versions of a released (or sometimes unreleased) application to users. Each company will have its own way of making releases available. But in general, the following five techniques are the most popular.

A/B Testing

This deployment technique entails releasing two versions of the application with likely varied functionalities to two different subsets of users under predefined conditions. This technique is mostly used to test which version of an application produces some required outcome. After the testing, the underperforming application will be undeployed.

Beta/Canary

This deployment technique is used to release a not fully ready version of the application to users that opt to use such version in return for feedback. Such users get to test the latest upcoming features first and in turn provide feedback that will be used to make the application fully ready for all the users.

Recreation

This deployment technique is perhaps the most straightforward one in which an existing version is supplanted by a new version. The old version is terminated and undeployed.

Ramped

This deployment technique is where a new version of the application is released and then it gradually replaces the previous version. The previous version will be in use until all users get updated/upgraded to the latest deployed version.

This is not an exhaustive list of deployment techniques, but are by far the most popular ones that are deployed in one form or the other across many companies.

Deployment Options

There are a good number of deployment options out there that can be used to deploy production Jakarta EE applications. The type of deployment option is mostly a combination of business and technical considerations that are particular to each organisation. The following are the major options available to you as a Jakarta EE developer.

Self Hosted Dedicated Servers

This used to be the most popular option for deploying Jakarta EE applications in the past. This option entails an organisation signing up for a virtual machine from a provider, downloading an application runtime like Payara on the machine, configuring it and then uploading an application binary to the configured server. This option may still be popular with solo/small organisations with very limited budgets.

Another option with dedicated servers is where an organisation leases dedicated servers, then uses in-house skills to provision all the infrastructure needed to deploy containerised Jakarta EE applications. So even though the application is containerised, the entire infrastructure for deployment is managed in-house by the organisation.

The cost of dedicated servers varies depending on the configured specification. However, this option is predictable in terms of cost because the cost of the server is known ahead of time. An organisation can plan and budget based on how many servers they will need and know exactly how much it will cost.

The simplicity and cost effectiveness of this deployment option is outweighed by all the plumbing that needs to be done to get the server ready to receive and serve traffic. The management of the application server and the containerised infrastructure are done by personnel of the organisation. With the rise of cheaper cloud offerings, this option isn't one I'd recommend as a first option to you.

Containerized Deployments

The rise of containerisation changed how software is developed, tested and deployed. Container platforms like Docker and Kubenetes have given rise to an almost infinite number of options for application deployment. For brevity, let us assess Jakarta EE containerised deployment options from the three big cloud providers - Amazon Web Services, Microsoft Azure and Google Cloud Platform.

Amazon Web Services

Amazon Web Services is by far the largest cloud provider in the world with a plethora of services for every software development concern. For containerized Jakarta EE deployment, AWS has the following options.

AWS Beanstalk

Beanstalk is a service that allows you to deploy your containerized Jakarta EE applications, with ancillary services automatically provisioned for you. You provision the service through a series of answers and then the platform deploys your containerised application on your behalf, with a public facing URL.

Amazon Elastic Container Service

Amazon Elastic Container Service is a fully managed infrastructure for running containerised workloads. It has integration with Amazon Container Registry for pulling your published images. This option is by far the most flexible but also likely the most complicated to fully grasp. It is flexible because it is general purpose for running any imaginable container workload. And naturally that much flexibility comes with some complexity that will need to be managed.

AWS App Runner

AWS App Runner is a simplified version of Amazon Elastic Container Service for deploying containerised web applications and API services. You upload your container image, or point the service to where that image can be pulled from, configure the service and have it deployed.

Microsoft Azure

Microsoft Azure is the second most popular cloud provider, after AWS. Given the maturity and enterprise experience of Microsoft, Azure equally has a plethora of services for deploying containerised workloads. For Jakarta EE applications, Azure has the Web App for Containers service.

Web App for Containers

Web App for Containers is a fully managed platform for deploying containerised web applications. All needed services are automatically provisioned for you. With Microsoft owning GitHub, it is much easier to set up a direct pipeline to the service from your GitHub repo. Azure does have other services that can be used to deploy Jakarta EE workloads. However, Web App for Containers is by far the most flexible and feature for containerised Jakarta EE applications.

Google Cloud Platform

Google Cloud Platform is the cloud offering from Google. It has a plethora of services for all kinds of industries and software needs. For Jakarta EE containerised workload, Google has Cloud Run.

Google Cloud Run

Google Cloud Run is a managed platform for deploying containerised applications. This is naturally suited to deploying Jakarta EE containerised workloads. Cloud Run has feature parity with the other options from the above discussed providers with regards to integrations with Git and container registries.

Cost Considerations

The containerised deployment options discussed above are not an exhaustive list of all that the cloud providers offer. However, one thing that is common among all of them is the pricing model. All of them use the pay as you go cost model for billing. This model can be both beneficial and challenging depending on a number of factors. Paying only for what you use is naturally a good thing since you save money when your application isn't serving requests. However, from a financial planning perspective, this model can be very challenging for budgeting and forecasting.

How much should we allocate to deployment costs? What happens if we have sudden, unexpected spikes? How much do we budget for that? These are some of the considerations that will go into planning for application deployment. However, given the popularity of these services and their pricing models, it is safe to say that for a large number of organisations, such a model works. But keep in mind what works for large organisations in terms of pricing may not necessarily work for small and medium organisations. Your mileage with regards to cloud cost models may vary.

Native Jakarta EE Deployment

The options for Jakarta EE application deployment discussed so far are much more generic and use containers to abstract your application. However, the developer still needs to take care of packaging the application as a container with all the setup needed for such. However, there is also a native Jakarta EE cloud deployment option that goes a step back by allowing you to upload your application binary (.WAR file) and have it automatically deployed to the cloud.

Payara Cloud

Payara Cloud is the first Jakarta EE native cloud deployment option that abstracts you from all the container infrastructure and information, freeing you to focus almost exclusively on your application code and business domain. It works by allowing you to upload your application binary, optionally configure your application data source and other resources and have the application automatically deployed to the cloud, with a URL for accessing it.

Each account has a namespace within which N-number of applications can be deployed. Each application has its unique URL to which you can direct application specific custom domains. Payara Cloud is a product from Payara, a leading and well known Jakarta EE community organisation. Payara Cloud is a new cloud native Jakarta EE application production deployment option that is worth considering.

The service has a simple pricing model where you pay for a pre-allocated block of compute resources. This gives you a clear way to budget for application deployment as you know ahead of time what compute resources you are paying for and how much it will cost you over a given period. Payara Cloud abstracts you from the low level infrastructure plumbing, relieving your team to focus on delivering business value.

Summary

There are a good number of options for deploying Jakarta EE applications to the cloud. Each option comes with its own costs and benefits. The choice of a deployment option is a business and technical decision. However, not all options are suitable for all organisations. As a developer, architect or CTO, your goal is to deliver business value to customers. And choosing the right option that frees as much time as possible to deliver value to customers must be the most important metric in selecting a cloud deployment option.

FREE TRIAL

If you liked this, you might like:

- Explaining Microservices: No Nonsense Guide for Decision Makers
- Jakarta EE 10: What Decision Makers Need to Know
- Dismiss The Myths: Open Source

sales@payara.fish

UK: +44 800 538 5490
Intl: +1 888 239 8941

www.payara.fish