UML Deployment Diagram

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.

Purpose of Deployment Diagram

The main purpose of the deployment diagram is to represent how software is installed on the hardware component. It depicts in what manner a software interacts with hardware to perform its execution.

Both the deployment diagram and the component diagram are closely interrelated to each other as they focus on software and hardware components. The component diagram represents the components of a system, whereas the deployment diagram describes how they are actually deployed on the hardware.

The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

Following are the purposes of deployment diagram enlisted below:

1. To envision the hardware topology of the system.
2. To represent the hardware components on which the software components are installed.
3. To describe the processing of nodes at the runtime.

How to draw a Deployment Diagram?

The deployment diagram portrays the deployment view of the system. It helps in visualizing the topological view of a system. It incorporates nodes, which are physical hardware. The nodes are used to execute the artifacts. The instances of artifacts can be deployed on the instances of nodes.

Since it plays a critical role during the administrative process, it involves the following parameters:

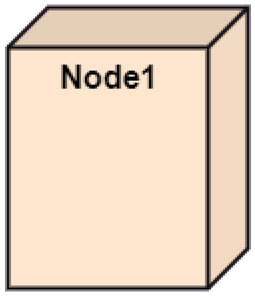
1. High performance
2. Scalability
3. Maintainability

4. Portability
5. Easily understandable

One of the essential elements of the deployment diagram is the nodes and artifacts. So it is necessary to identify all of the nodes and the relationship between them. It becomes easier to develop a deployment diagram if all of the nodes, artifacts, and their relationship is already known.

Example of a Deployment diagram

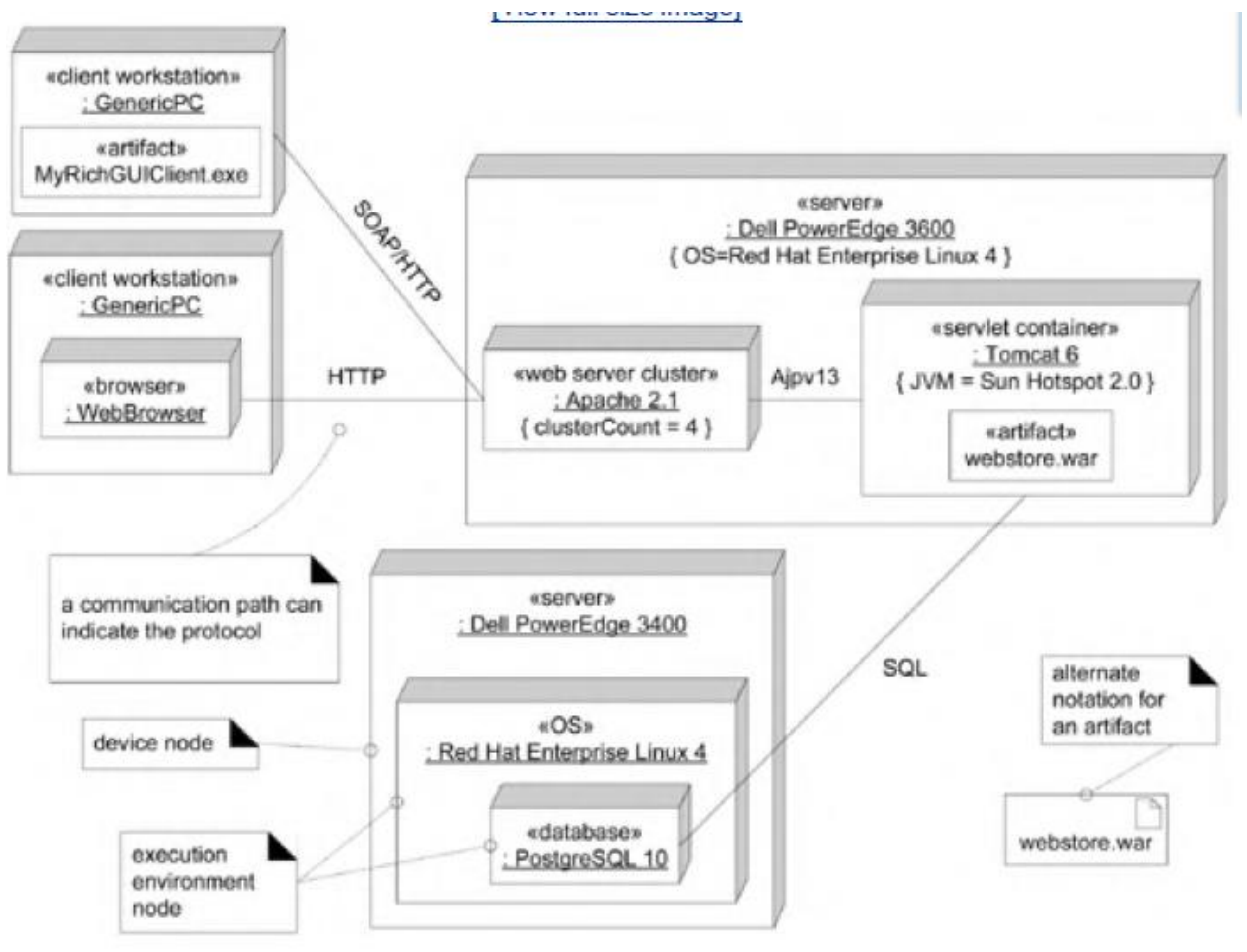
The basic element of a deployment diagram is a node, of two types:



device node (or device) A physical (e.g., digital electronic) computing resource with processing and memory services to execute software, such as a typical computer or a mobile phone.

execution environment node (EEN) This is a software computing resource that runs within an outer node (such as a computer) and which itself provides a service to host and execute other executable software elements. For example:

- an operating system (OS) is software that hosts and executes programs
- a virtual machine (VM, such as the Java or .NET VM) hosts and executes programs
- a database engine (such as PostgreSQL) receives SQL program requests and executes
- them, and hosts/executes internal stored procedures (written in Java or a proprietary language)
- a Web browser hosts and executes JavaScript, Java applets, Flash, and other executable technologies
- a workflow engine
- a servlet container or EJB container



When to use a Deployment Diagram?

The deployment diagram is mostly employed by network engineers, system administrators, etc. with the purpose of representing the deployment of software on the hardware system. It envisions the interaction of the software with the hardware to accomplish the execution. The selected hardware must be of good quality so that the software can work more efficiently at a faster rate by producing accurate results in no time.

The software applications are quite complex these days, as they are standalone, distributed, web-based, etc. So, it is very necessary to design efficient software.

Deployment diagrams can be used for the followings:

1. To model the network and hardware topology of a system.
2. To model the distributed networks and systems.
3. Implement forwarding and reverse engineering processes.
4. To model the hardware details for a client/server system.

5. For modeling the embedded system.