Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

The deployment diagram maps the software architecture created in design to the physical system architecture that executes it. In distributed systems, it models the distribution of the software across the physical nodes.

The software systems are manifested using various **artifacts**, and then they are mapped to the execution environment that is going to execute the software such as **nodes**. Many nodes are involved in the deployment diagram; hence, the relation between them is represented using communication paths.

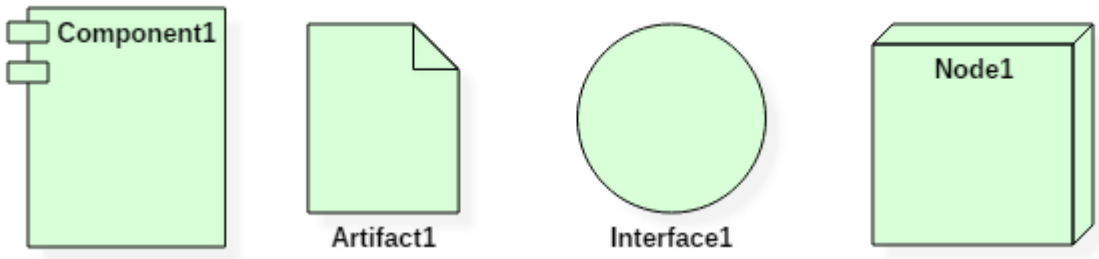
There are two forms of a deployment diagram.

- Descriptor form
  - It contains nodes, the relationship between nodes and artifacts.
- Instance form
  - It contains node instance, the relationship between node instances and artifact instance.
  - An underlined name represents node instances.

Purpose of a deployment diagram

Deployment diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality. It is used to describe software to hardware interaction and vice versa.

Deployment Diagram Symbol and notations



Deployment Diagram Notations

A deployment diagram consists of the following notations:

1. A node

2. A component
3. An artifact
4. An interface

What is an artifact?

An artifact represents the specification of a concrete real-world entity related to software development. You can use the artifact to describe a framework which is used during the software development process or an executable file. Artifacts are deployed on the nodes. The most common artifacts are as follows,

1. Source files
2. Executable files
3. Database tables
4. Scripts
5. DLL files
6. User manuals or documentation
7. Output files

Artifacts are deployed on the nodes. It can provide physical manifestation for any UML element. Generally, they manifest components. Artifacts are labeled with the stereotype **<<artifact>>,** and it may have an artifact icon on the top right corner.

Each artifact has a filename in its specification that indicates the physical location of the artifact. An artifact can contain another artifact. It may be dependent on one another.

Artifacts have their properties and behavior that manipulates them.

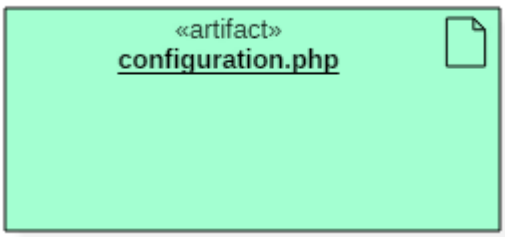
Generally, an artifact is represented as follows in the unified modeling language.

artifact

Artifact Instances

An artifact instance represents an instance of a particular artifact. An artifact instance is denoted with same symbol as that of the artifact except that the name is underlined. UML diagram allows this to differentiate between the original artifact and the instance. Each physical copy or a file is an instance of a unique artifact.

Generally, an artifact instance is represented as follows in the unified modeling language.



artifact instance

What is a node?

Node is a computational resource upon which artifacts are deployed for execution. A node is a physical thing that can execute one or more artifacts. A node may vary in its size depending upon the size of the project.

Node is an essential UML element that describes the execution of code and the communication between various entities of a system. It is denoted by a 3D box with the node-name written inside of it. Nodes help to convey the hardware which is used to deploy the software.

An association between nodes represents a communication path from which information is exchanged in any direction.

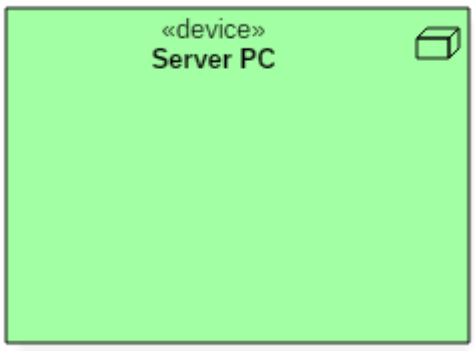
Generally, a node has two stereotypes as follows:

- **<< device >>**

  It is a node that represents a physical machine capable of performing computations. A device can be a router or a server PC. It is represented using a node with stereotype <<device>>.

  In the UML model, you can also nest one or more devices within each other.

  Following is a representation of a device in UML:

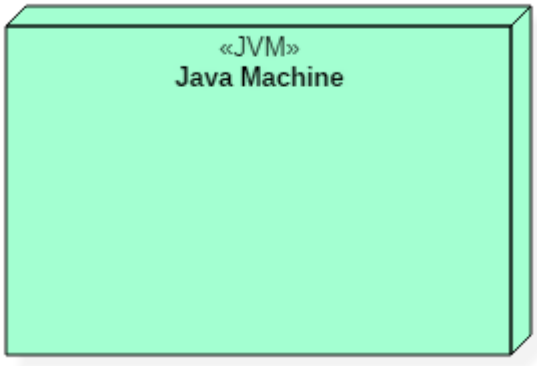


device node

- **<< execution environment >>**

  It is a node that represents an environment in which software is going to execute. For example, Java applications are executed in java virtual machine (JVM). JVM is considered as an execution environment for Java applications. We can nest an execution environment into a device node. You can net more than one execution environments in a single device node.

Following is a representation of an execution environment in UML:



execution environment node

How to draw a deployment diagram?

Deployment diagram visualizes the topological view of an entire system. It represents the deployment of a system.

A deployment diagram consists of nodes which describe the physical devices used inside the system. On these nodes, artifacts are deployed. We can also have node instances on which artifact instances are going to be implemented.

Node and artifacts of a system participate in the final execution of a system.

A deployment diagram plays a critical role during the administrative process, and it must satisfy the following parameters,

- High performance
- Maintainability
- Scalability
- Portability
- Easily understandable

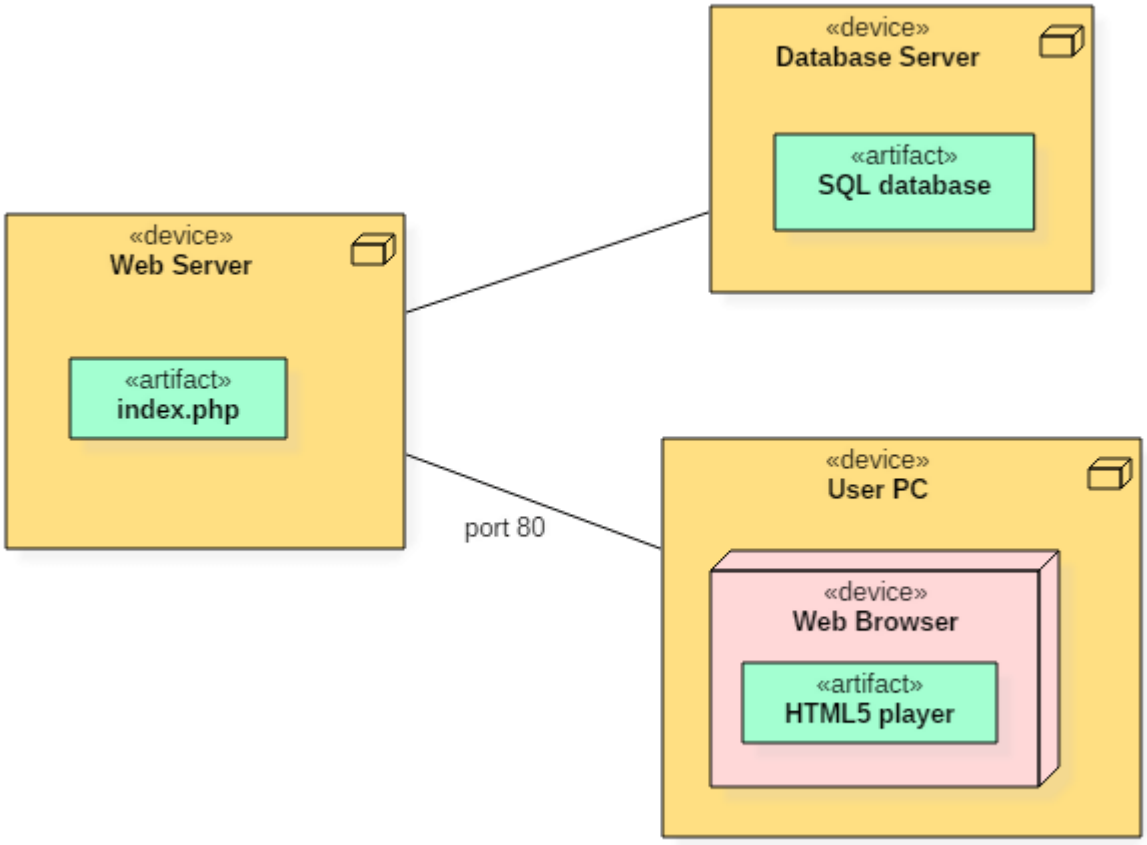
Nodes and artifacts are the essential elements of deployment. Before actually drawing the deployment diagram, all nodes and the relationship between every node of the system must be identified.

You must know the architecture of a system, whether an application is a web application, cloud application, desktop application, or a mobile application. All these things are critical and plays a vital role during the development of a deployment diagram.

If all the nodes, relations, and artifacts are known, then it becomes easy to develop a deployment diagram.

Example of a Deployment diagram

Following deployment diagram represents the working of HTML5 video player in the browser:



Deployment Diagram

When to use a deployment diagram?

Deployment diagrams are mostly used by system administrators, network engineers, etc. These diagrams are used with the sole purpose of describing how software is deployed into the hardware system. It visualizes how software interacts with the hardware to execute the complete functionality.

To make the software work efficiently and at a faster rate, the hardware also must be of good quality. It must be designed efficiently to make software work properly and produce accurate results in quick time.

Deployment diagrams can be used for,

1. Modeling the network topology of a system.
2. Modeling distributed systems and networks.
3. Forward and reverse engineering processes.