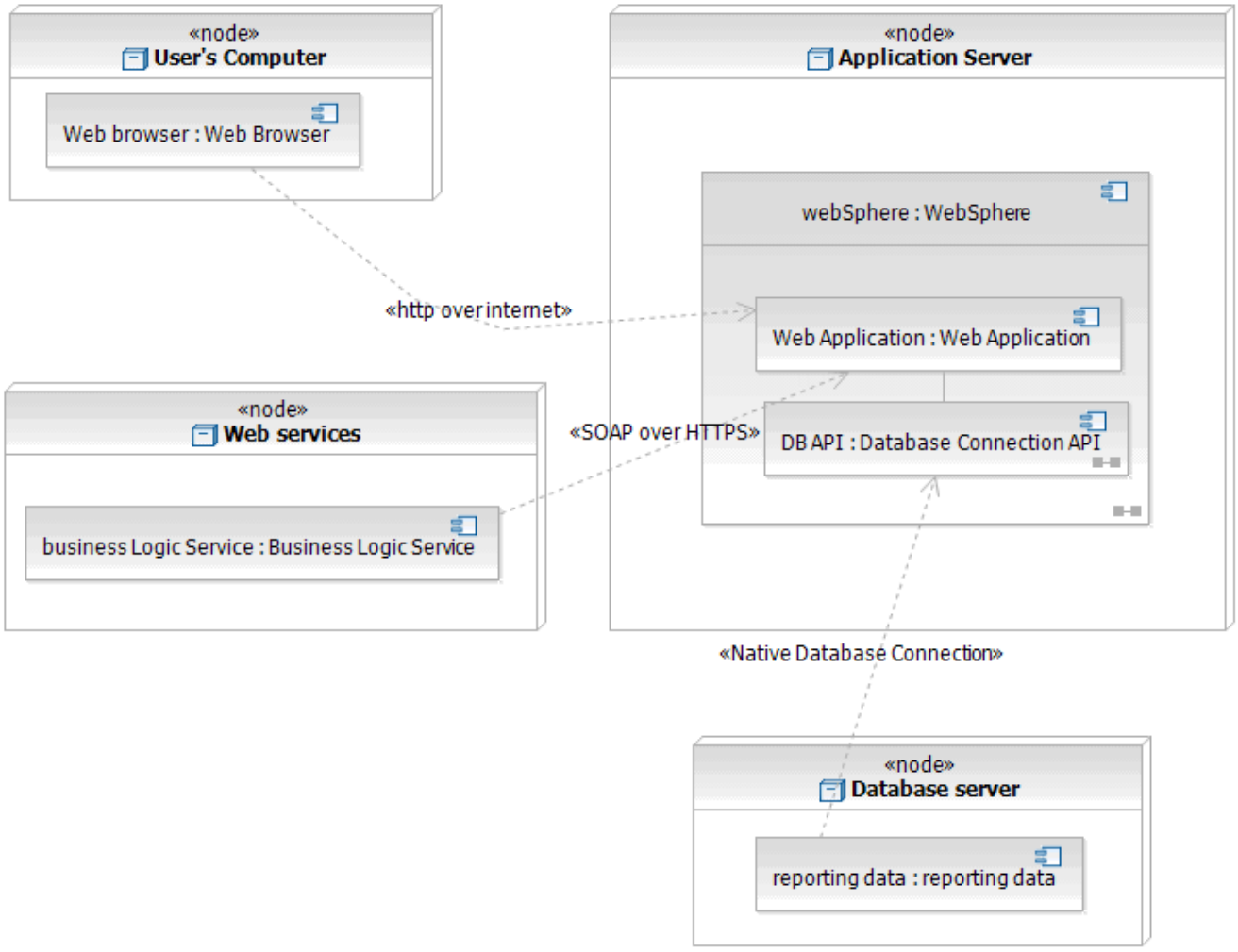
Deployment diagrams

Last Updated: 2023-09-21

In UML, deployment diagrams model the physical architecture of a system. Deployment diagrams show the relationships between the software and hardware components in the system and the physical distribution of the processing.
Deployment diagrams, which you typically prepare during the implementation phase of development, show the physical arrangement of the nodes in a distributed system, the artifacts that are stored on each node, and the components and other elements that the artifacts implement. Nodes represent hardware devices such as computers, sensors, and printers, as well as other devices that support the runtime environment of a system. Communication paths and deploy relationships model the connections in the system.



Deployment diagrams are effective for visualizing, specifying, and documenting the following types of systems:

- Embedded systems that use hardware that is controlled by external stimuli; for example, a display that is controlled by temperature change
- Client/server systems that typically distinguish between the user interface and the persistent data of a system
- Distributed systems that have multiple servers and can host multiple versions of software artifacts, some of which might even migrate from node to node

Because deployment diagrams focus on the configuration of the runtime processing nodes and their components and artifacts, you can use this type of diagram to assess the implications of distribution and resource allocations.

> **Note:** Deployment diagrams are distinct from component diagrams. A deployment diagram shows components and artifacts in relation to where they are used in the deployed system. A component diagram defines the composition of components and artifacts in the system.

> **Note:** Deployment diagrams are distinct from deployment topologies, a different type of model. For information on deployment topologies, see Modeling deployment topologies.

The following topics describe model elements in deployment diagrams:

- **Nodes in UML models**
  In UML models, *nodes* are model elements that represent the computational resources of a system, such as personal computers, sensors, printing devices, or servers. Nodes can be connected by communication paths to describe network structures.
- **Node instances**
  In UML modeling, a *node instance* is a model element that represents an instantiation, or actual occurrence, of a node. Node instances are based on existing nodes.
- **Execution environments**
  In UML modeling, an *execution environment* is a type of node that represents a particular execution platform, such as an operating system or a database management system. You can use execution environments to describe the context in which the execution of a model takes place.
- **Artifacts**
  In UML models, *artifacts* are model elements that represent the physical entities in a software syster Artifacts represent physical implementation units, such as executable files, libraries, software components, documents, and databases.
- **Artifact instances**
  In UML modeling, an *artifact instance* is a model element that represents an instantiation, or actual occurrence, of an artifact. Artifact instances are based on existing artifacts.
- **Devices**
  In deployment diagrams, a *device* is a type of node that represents a physical computational resourc in a system, such as an application server.
- **Deployment specifications**
  A *deployment specification* is essentially a configuration file, such as an XML document or a text file,

that defines how an artifact is deployed on a node.

– **Relationships in deployment diagrams**
In UML, a relationship is a connection between model elements. A UML relationship is a type of mode element that adds semantics to a model by defining the structure and behavior between model elements.

Related tasks:

→ Creating deployment diagrams

→ Specifying the deployment of artifacts within nodes

Feedback