Deployment Patterns

4 out of 6 rated this helpful

Retired Content

This content is outdated and is no longer being maintained. It is provided as a courtesy for individuals who are still using these technologies. This page
when originally published, but now link to sites or pages that no longer exist. Please see the patterns & practices guidance for the most current inform

Microsoft®
patterns & practices
proven practices for predictable results

Version 1.1.0

Complete List of patterns & practices

"What do you mean it doesn't run in production? It ran fine in the development environment." - Anonymous developer

Building enterprise class solutions involves not only developing custom software, but also deploying this software into a production server environment.
development efforts intersect with systems infrastructure efforts. Bringing these two disciplines together effectively requires a common understanding c
set of application and system infrastructure skills. The required skills are rarely found in a single team; therefore, deployment activities often involve the
each team contributing specialized skills. To simplify discussion, this chapter assumes that there are two teams: the application development team and t

Bringing Teams Together

The application development team is responsible for developing and maintaining a set of software components that fulfill the application's requirement
with meeting functional requirements quickly and flexibly. Its members seek to manage complexity by creating software abstractions that make the syst

The system infrastructure team is responsible for building and maintaining the servers and network infrastructure. Its members are primarily concerned
as security, availability, reliability, and performance. Stability and predictability are critical success factors, which are primarily addressed by controlling ch
good configurations.

The forces acting on the application development team are quite different from the forces acting on the system infrastructure team. The result is an inhe
teams. If this tension is not addressed, the resulting solution may be optimized for one team or the other, but it will not be an optimal business solution
element for delivering a holistic, software-intensive enterprise solution that is optimized for the overall needs of the business.

The patterns in this chapter help reduce the tension between the teams by offering guidance on how to optimally structure your applications and techni
the solution's requirements. The patterns then discuss how to map the software structure to the hardware infrastructure. Specifically, this chapter presen
to:

- Organize your software application into logical layers.
- Refine your logical layering approach to provide and consume services.
- Organize your hardware into physical tiers, so you can scale out.
- Refine your physical tier strategy in a three-tiered configuration.
- Allocate processes to processors with a deployment plan.

Patterns Overview

Although the concepts of layer and tier are often used interchangeably, the patterns in this chapter make a strong distinction between the two terms. A
mechanism for the elements that make up your software solution; a *tier* is a physical structuring mechanism for the system infrastructure. The first set of
the logical structuring of the software application into layers. The second set of patterns explores the physical structuring of the system infrastructure in
patterns and their interrelationships.

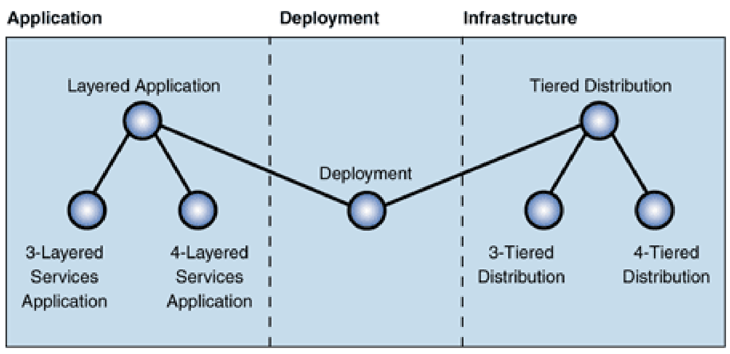

Figure 1: Deployment cluster

Application Patterns

The first pattern in this cluster, *Layered Application*, organizes a software application into a set of logical layers for the purpose of managing dependenci
components. The pattern defines exactly what a layer is and then describes how to define your own layers. It also describes some additional techniques
benefits of using a layered application. One of the key benefits of *Layered Application* is that the well-defined interfaces and strong dependency manage
flexibility in deploy an application. Although it is very hard to distribute a single-layered application across multiple servers, it is much easier to divide th
and distribute the different parts to multiple servers. Not every layer boundary makes a good distribution boundary, however, because the forces that m
different from the forces that shape your distribution decisions. For more information, see *Deployment Plan*.

Layered Application is applied extensively throughout the software development world. A common implementation of the pattern for enterprise applicat
This implementation defines three layers: presentation, business, and data. Although you can add more layers, these three layers are almost always neec
applications.

Most enterprise applications are now developed using a component-based approach. Although many definitions of a component exist, the simplest def
of self-contained software functionality that can be independently deployed. Components can be plugged into and out of an execution environment thа
interfaces at runtime. This pluggability offers a great deal of flexibility when it comes to deployment. The self-contained aspect of components makes th
deployment decisions can be made.

Three-Layered Services Application refines *Layered Application*to provide specific structuring guidance for enterprise applications that collaborate with ot
larger service-oriented architecture. It expands on the typical three layers described earlier and defines a set of component types for each layer.

Infrastructure Patterns

The next set of patterns in this cluster focuses on the physical infrastructure. The context for these patterns is an infrastructure that supports an applicati
server. Specifically, these patterns do not address mainframe or other large multiprocessor infrastructure configurations.

Tiered Distribution organizes the system infrastructure into a set of physical tiers to provide specific server environments optimized for specific operatior
resource usage. A single-tiered infrastructure is not very flexible; the servers must be generically configured and designed around the strictest of operati
support the peak usage of the largest consumers of system resources. Multiple tiers, on the other hand, enable multiple environments. You can optimize
of operational requirements and system resource usage. You can then deploy components onto the tier that most closely matches their resource needs
their operational requirements. The more tiers you use, the more deployment options you will have for each component.

Three-Tiered Distribution refines *Tiered Distribution* to provide specific guidance on structuring the infrastructure for Web applications with basic security
requirements. The pattern suggests that the solution's servers be organized into three tiers: client, Web application, and data. The client and data tiers a
application tier hosts application business components as well as the Web presentation components. For solutions with more stringent security and ope
want to consider moving the Web functionality into its own tier.

Bringing Applications and Infrastructure Together

The final pattern in the cluster is *Deployment Plan,* which describes a process for allocating components to tiers. During this process, it is critical to ensu
the application development and system infrastructure teams. All the previous patterns increase the deployment flexibility of the software application ar
Deployment Plan builds on this deployment flexibility, which provides more options for the teams to resolve conflicts of interest. Resolving these conflic

simple Web application, complex Web application, extended enterprise, and rich client.

Deployment Patterns

Table 1 lists the patterns included in the deployment cluster, along with the problem statements for each, which should serve as a roadmap to the patte

Table 1: Deployment Patterns

Layered Application	How do you structure an application to support such operational requirements as maintainability, reusability, sca
Three-Layered Services Application	How do you layer a service-oriented application and then determine the components in each layer?
Tiered Distribution	How should you structure your servers and distribute functionality across them to efficiently meet the operationa
Three-Tiered Distribution	How many tiers should you have, and what should be in each tier?
Deployment Plan	How do you determine which tier you should deploy each of your components to?

Microsoft® patterns & practices
proven practices for predictable results