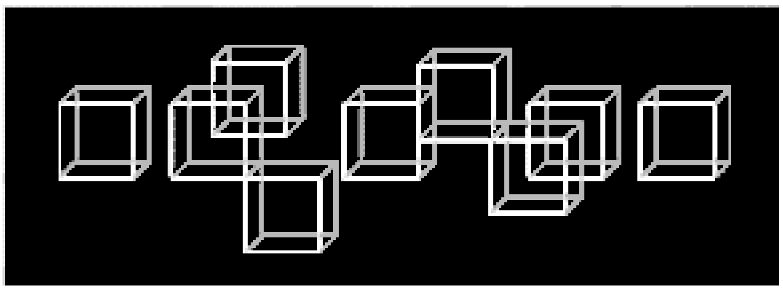
Building a Web Site:



A Developer's Guide

by James Mendelsohn

Table of Contents

Overview and First Principles

At first glance, nothing could make expert developers more impatient than a description of the basics of building a Web site. But time and again, our experts report that even the best developers feel that they are better in some areas of site creation than in others. Or they tend to neglect some of the more basic elements for building a good site. Instead, they focus with considerable brilliance on details but miss the larger view. They don't see the forest for the trees.

Hence the purpose of this guide, which provides a view of the forest for the aspiring and the expert developer. The point is to ensure that the fundamentals of creating a Web site are covered, freeing you to consider without worry the details — which other Dot-Com Builder articles address. This is a view from above: a comprehensive — but not excessively detailed — view.

This overview of the entire guide covers the following topics:

- Organization of the guide
- Reading this guide
- First principles for designing a Web site

The site as a system

- Critical objectives
- Practical design issues throughout the building process
  - Content on your site
- Sizing and anticipating growth
- Sequestering applications and segmenting content

Organization of the Guide

We proceed as if we were architects building a house, and we describe site building in eight discrete sections or "building blocks." First, we address the creation of a data center, whether in-house or outsourced. Then we move to the contents of "the house," from network infrastructure to server hardware, and so on. The building blocks are as follows:

- Data centers
- Network infrastructure
- Server hardware
- The OS and database layer

- Identity and security policy
- Application servers and content management
- Development and staging environments
- Metrics, monitoring, and performance

Reading This Guide

The eight blocks of the guide are designed to be read consecutively, but they do not exactly identify a step-by-step procedure for creating a Web site. Each block builds upon those before it, describing an order by which you might proceed. But that order should not be taken strictly. Nor would you make all of your decisions consecutively, in the order the building blocks are presented, because you should consider some elements of the later blocks from the beginning. For example, truly good performance and monitoring (the subject of the last block) must be built into the data center, the server hardware, and — where applicable — the application servers (the subjects of the first, third, and sixth blocks). You need to be aware of the contents of each building block as you plan a site. Thus you may find it wise to read through all of the blocks before acting.

If you are well-versed in all but a few of the subjects covered here, you can of course feel free to skip around. Each building block is also designed to be read as an independent unit.

First Principles for Designing a Web Site

While the eight building blocks address interrelated components, there are also principles, critical objectives, and practical design issues to bear in mind throughout the process of creating a Web site.

The Site as a System

A basic principle of site design underlies this guide: A Web site is a system. By its very nature, all the pieces of the site fit together like a puzzle, all of them interrelated. So the kind of system you decide upon in one area will determine the kinds of choices you make in another area. For example, the decisions you make about data centers will affect your choices in network infrastructure. If you decide to use a co-location facility, you will need a network to upload content and code to the site; but if not, you may not need a VPN. The choices you make in network infrastructure will determine the kinds of decisions you make in server hardware, and so on until, like a machine, all the parts build together into a site functioning as a whole — without incongruous or spare parts.

Critical Objectives

While we do not state it explicitly, every aspect of site building requires you to bear in mind these design objectives:

- Security. In every element of building and policy, you should protect the site and plan against the possibility of site failure.
- Reliability. Your site should be as continuously available to users as possible, as free of downtime as you can make it.
- Performance. Your site should respond to users as quickly and accurately as possible, and you should plan so that it continues to do so as it grows.
- Supportability. Your site should be as easy to maintain as possible, for which simplicity of design is an important principle.

Practical Design Issues Throughout the Building Process

From abstract objectives, we move to practical matters. As you design, a few considerations are so often influential that they should be before you throughout the process. Two are characteristics of your site, its content and its size. One is an element of design that promotes performance and security, namely sequestering applications and segmenting content.

Content on Your Site

Content determines the ways in which the site interacts with users, so content will strongly influence many of the decisions you make about your site, from hardware and software to policy. For example, the kind of content affects whether you have considerable demand for applications, which in turn affects network architecture, choices about server hardware, and the configurations of that hardware. So as you plan the site — as you read each section of this guide — bear in mind these questions:

- Is your site producing mostly static content? Is it a more personalized site, directed toward providing custom applications?
- Is it producing user-generated content? Is your site transaction-oriented, as is e-commerce?

Sizing and Anticipating Growth

Your decisions about nearly everything on the site vary according to its size — whether it is medium, large, or super-large — and the growth for which you need to plan. Scalability is but one element here. For your server hardware and database, for the architecture of your applications, for your data center and more, assess how much capacity you need at present and how to create the added capacity you will need in the future.

Sequestering Applications and Segmenting Content

As you create a network architecture, the security of your site and its performance are enhanced if you sequester applications and segment content:

- **Sequestering applications**. You should place each functional unit or application — such as the database, the Web server, and the mail server — in a chrooted environment on its own piece of hardware or the equivalent of such an environment on non-UNIX® systems. Hackers will have a much more difficult time infiltrating your system beyond the specific area they manage to compromise. The amount of damage a hacker might do is thereby contained.
- **Segmentation of content**. If you write your own applications so they are not separable, then your options for load balancing later will be reduced because you won't be able to balance according to the kind of application. In general, segmenting the content on your site so that its forums and various applications are all autonomous groups of files (so that you can navigate between them without breaking links) gives you the greatest flexibility for improving performance.

Building Block 1: A Data Center

You first need a physical home for your Web site, a location for its equipment and connections: a data center. And as with any housing, the choices you make are never simple. The most obvious and complex decision is whether to create a data center in-house or to outsource all or part of it. In this building block, we'll cover the basic issues you'll want to address:

- Budget issues for data centers
  - In-house or outsourced data center
  - Other budget factors
- General considerations for all data centers
  - Scalable space
  - Secure environment
  - Multiple data centers
  - Air conditioning
  - Redundant power supply
  - Sufficient and redundant bandwidth
  - Round-the-clock support
- Outsourcing issues for data centers
  - Degree of outsourcing
  - Firewalls
  - Scalable, redundant bandwidth

Budget Issues for Data Centers

You may want to think only in technical terms, but your budget is not going to let you do so. Budget issues will affect how you create a data center.

Data Center Decision Table				
Requirement	**In-House: Pro**	**In-House: Con**	**Outsourced: Pro**	**Outsourced: Con**
Power Backup		Requires owning an electric generator. Possible zoning issues.	Standard with vendors	
Internet Bandwidth		Can be expensive. Should use multiple vendors in case of failover. Can be difficult to add capacity during peak demands.	Vendor spreads cost over many customers. Has excess capacity for peak loads.	Vendor may have oversold the facility and have poor performance
Physical Security	Easy to limit access to trusted employees	Requires 24x7 monitoring	Vendors offer varied levels of security	Shared areas are less secure
Network Security		Requires in-house expertise that you may not have	Security experts available when needed	
24x7 Support		Requires in-house support staff. Expensive to provide full time on-site coverage. Companies more often use on-call support.	Standard with most vendors. Cost is shared over many customers.	Custom support can be expensive
Cost	Large up-front capital expense	Low recurring costs	Only pay for what you need	Can be high monthly expenses. Subject to vendor price increases.
Future Expansion	Easy if you build a large data center from the beginning	Very expensive once you outgrow your data center	Generally have additional space available when needed. Easy to add space in different geographic locations.	Hard to lock in additional space adjacent to your current computers

In-House vs. Outsourced

In-House or Outsourced Data Center

Budget issues are never simply a matter of how much money you have but of how you are financed and how you can best balance costs. While for some developers housing and creating an in-house data center sounds ridiculously expensive, for others outsourcing seems wildly extravagant. To call forth a bit of well-worn advice, only you know what best fits your organization.

It is not that one of these two solutions is more expensive than the other. They are differently expensive. On the one hand, building and maintaining a data center requires significant capital expenditures and resources; on the other hand, the monthly cost of outsourcing will astonish you. At a well-known co-location facility on the West Coast, the 2001 list price for floor space alone was $127 per square foot per month. While no one ever pays full list price, no one ever buys simply the floor space. In addition, outsourcing may mean the expense of providing and expanding a data center is subject to changes in the market. It's hard to lock in a cost with this solution.

As a general rule, if you have the money for capital expenditures and a well-developed IT staff, building in-house may be the best choice; if you don't have either the resources or the capital, outsourcing may be your best and certainly quickest option. But those of you with capital and resources may still find a co-location facility that is nearby and inexpensive; or you may have little capital and resources but find an inexpensive means of creating a data center in-house: You may well find a solution that breaks this general rule but works.

Other Budget Factors

Building in-house or outsourcing is just one budget decision to consider. So, too, are a number of factors that affect that decision, including space, security, and location.

Amount of Space

Once you decide how much space you need now and in the future, the costs of that space will strongly affect where you can locate your data center. If, for example, the prices for space in a co-location facility are high and skyrocketing as they did in 1998, you might lean toward an in-house center rather than lock yourself into a lease at an exorbitant rate. If those prices are dropping precipitously, as in 2001, a lease from a co-location facility may be more attractive to you. The cost of expansion may also affect you. If you will need *x* amount of square feet now but 10 times *x* in only a year, a co-location facility may be less costly in the long run than that much additional office space.

Degree of Security

The costs of security vary according to where the data center is located, the potential threats to your center, and the value of the data and the equipment. The costs of securing the site may strain the resources you have, leading you to outsource; or the level of security you demand may make you certain you want the center to be in-house. For example, your office space may be so open to all employees and visitors it is impossible to secure a site for the data center without hiring 24-hour security and retrofitting the space at an unreasonable expense.

Or you may find the opposite to be the case: Many co-location facilities provide you with a wire-mesh cage that you share with several other businesses. Your equipment is both visible to all within the facility and exposed to the personnel of those other firms with which you share a cage. If your security needs are greater, then you have to pay for added features, such as upgrading to a private wire-mesh cage, or still more secure, a private, enclosed room, often equipped with a biometric scanner. Measuring the costs of that added security against the costs of the existing or upgraded security in your office space may lead you to decide that in-house is ultimately less expensive.

Location(s)
You may need more than one center (see the section on general considerations below). The costs of building multiple centers can powerfully influence the solutions you devise.

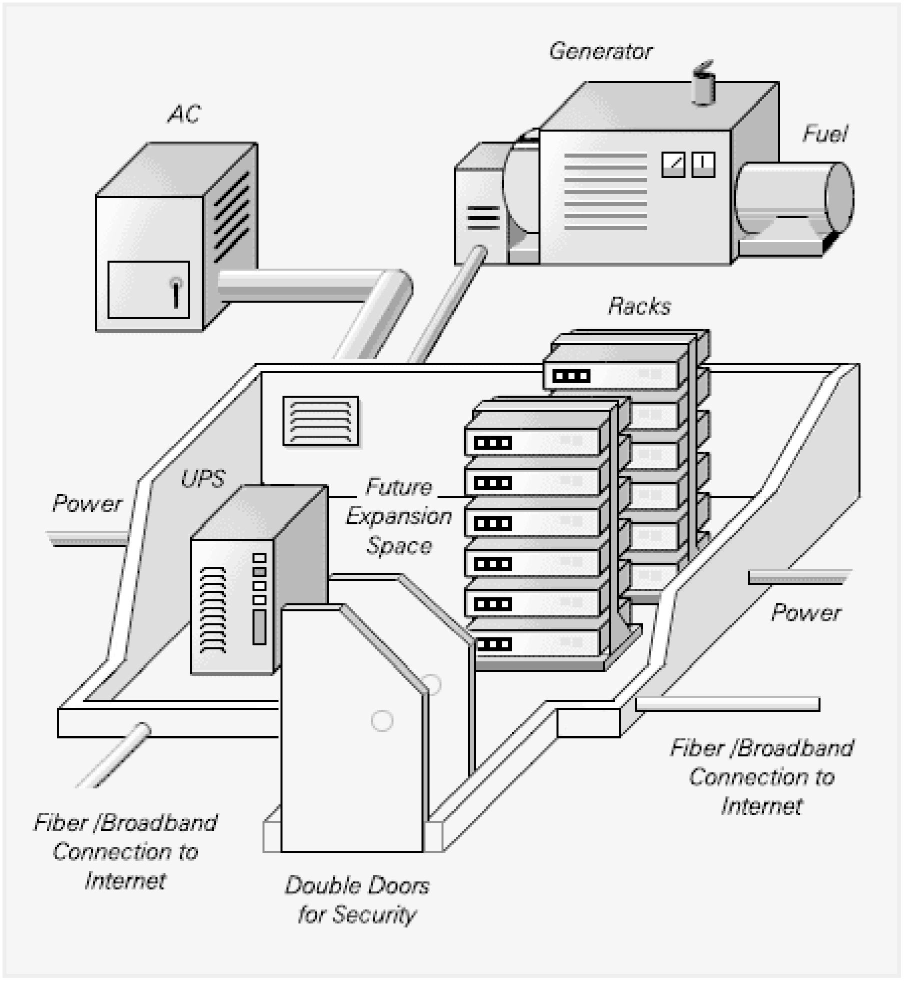
General Considerations for All Data Centers

Whether in-house or outsourced, your data center needs to consider the following issues to ensure your site functions well and securely.

Scalable Space
You need to judge how much physical space you require and also plan for additional space in the future, as you upgrade or expand. You should house your growing data center in one place, so be far-sighted: If in-house, make sure that the space you have is large enough to grow into, or that the areas adjacent to that space can be taken over. If outsourcing, make sure you have enough room for the present and that the space adjacent to what you've rented is also available. If the adjacent space isn't yet available, you want the right-of-first refusal when it becomes free.

Secure Environment
To guard against theft and loss, you need a secure space. For some developers, that may mean simply a partitioned section of an office; for others, it means a secure room or a building. The range of choices is immense, from KEVLAR-coated fortresses with state-of-the-art security to single rooms in a strip mall. In between these extremes are several possibilities. Many facilities require both digital badges and biometric scanners to enter. Some facilities not only use these badges and scanners but require advance notification and an authorization code before they will admit you to the facility. And still more secure facilities have 24-hour guards behind bullet-proof glass and pass you through a so-called "dead-man's chamber" — a room trapping visitors between full entrance and full exit — to ensure that the privileged visitor alone enters with each card and scanner authorization.

Whichever your choice of physical security, you also want a space that is equipped for a variety of natural disasters, from fires to floods to earthquakes (where applicable).


Generator
AC
Fuel
Racks
Power
UPS
Future Expansion Space
Power
Fiber /Broadband Connection to Internet
Fiber /Broadband Connection to Internet
Double Doors for Security

Physical Setup of Data Center

Multiple Data Centers
Ideally, to enhance security and ensure the best performance and availability, you should have more than one data center. Equipment in a data center can break down or be damaged. A second center guards against significant downtime. Data centers also perform better — they provide a faster user experience — if they are closer to the users they service. If, for example, your users are on the East and West Coasts, a data center on each coast will have a considerable effect on network performance, in particular on the speed with which Web pages load, and on overall site performance.

Air Conditioning
Data centers get very hot. You need to install central air conditioning to ensure your equipment runs properly unless you've outsourced, in which case the co-location facility will provide it. To meet your needs as you grow, be sure that the air-conditioning system you first purchase can handle an expanded center.

Redundant Power Supply
Ideally, you should have not only a regular but two kinds of backup power supply: First, an uninterruptible power supply (UPS), which automatically kicks in if the power goes out and doubles as continuous protection against power surges.

Typically, a UPS lasts only a few hours, so your goal is to have a second, more capable backup, a gasoline or diesel generator that can provide you with enough power for two to three days — power for running both your equipment and the air conditioning. (A cautionary note: Gas or diesel generators may conflict with zoning ordinances because many communities don't want that much gasoline or diesel fuel stored on site.)

Sufficient and Redundant Bandwidth
You should ensure that your supplier can provide you with enough bandwidth for now and greater bandwidth in the future. In addition, as with a power supply, you need redundancy, which means multiple lines. Those lines should enter into your data center at different locations, in case someone accidentally cuts a line in one location. You should have the bandwidth come from different suppliers in case one supplier suddenly goes down.

Round-the-Clock Support
Whether at your own facility or outsourced, your data center must be maintained and watched. You need a clear sense of who will monitor both your equipment and the bandwidth. You should know who is on call in case there is a problem in the middle of the night. The need for support should figure into your budget and administrative planning.

Outsourcing Issues for Data Centers

Virtually all of these general considerations apply to outsourcing. You must either provide for these concerns yourself or ensure that the co-location facility is doing so. But outsourcing has its own, special considerations, including the extent of outsourcing, firewalls, and bandwidth issues.

Degree of Outsourcing

You need to decide how much of the data center you wish to outsource. At a minimum, the co-location facility should provide you with physical security, power, and bandwidth. (Power and bandwidth are likely to be separate fees.)

At the maximum, for which you will pay dearly, the facility does nearly everything: You put the content on the machines; the facility runs and manages the machines as well as takes care of the firewalls and the backups. Then there's everything between these two extremes. For example, you may want the facility to monitor the machines, back them up, and when necessary, reboot them. A co-location facility should provide you with a menu of services and fees.

Firewalls

While the co-location facility is likely to have security, it's less likely to provide you with firewalls between the data center and any private connection you have to your office, such as a T1 line. You need to establish those firewalls.

Scalable, Redundant Bandwidth

You should determine if there is indeed enough megabits-per-second bandwidth available for you now and as you expand. If the co-location facility is so built out that it can't meet the bandwidth you will require in a year, then you have a serious problem. Our experts predict that it will be hard to get such information from a facility, but it is worth pursuing as best you can. Just as with your own data center, you should ask the co-location facility if they have redundant bandwidth connections and multiple suppliers of that bandwidth.

Building Block 2: The Network Infrastructure

From housing, we move to connections, a subject that experts know well. For them, this chapter will likely be a review of the basics. Much else could be said on this topic, and indeed, there are as many network infrastructure issues as there are thoughts in a day. But this building block provides a bird's eye perspective of the fundamental and readily identifiable areas you need to address.

With network infrastructure, we look at the communication layers between the components of your system, including those that connect to the Internet and those between components. To explain the basics, it is useful to work back from the Internet to the parts of your system, or "from the Internet in."

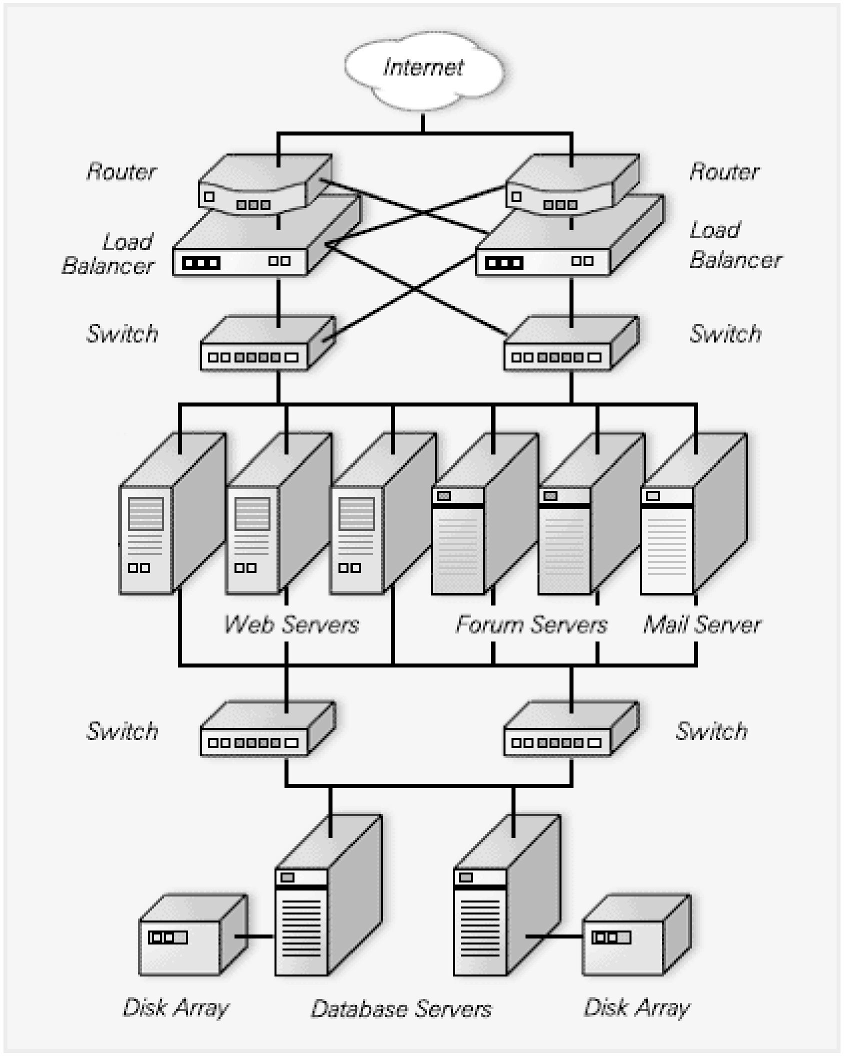
Network infrastructure divides into these topics:

- Connections needed for network infrastructure
- Sizing bandwidth
- Routers and switches
    - Router issues
- Network cards and connections
    - Virtual LANs
- Firewalls
- Load balancing

Connections Needed for Network Infrastructure

First and foremost, your infrastructure must connect multiple locations and parts of your system. In the most general terms, those locations and parts include:

- The data center or centers, where your production machines are located.
- The staging area machines, which may be where your production machines are, or just as typically, may be located with your development team.
- Your development team offices, which often are close to your development servers. Note well, however: Those offices could be in a number of nearby buildings or in a number of locations around the world.
- The various machines within your production environment, which may be in-house or at a co-location facility but could include database machines, Web servers, and dedicated application servers, as well as separate machines for LDAP, mail, and chat.

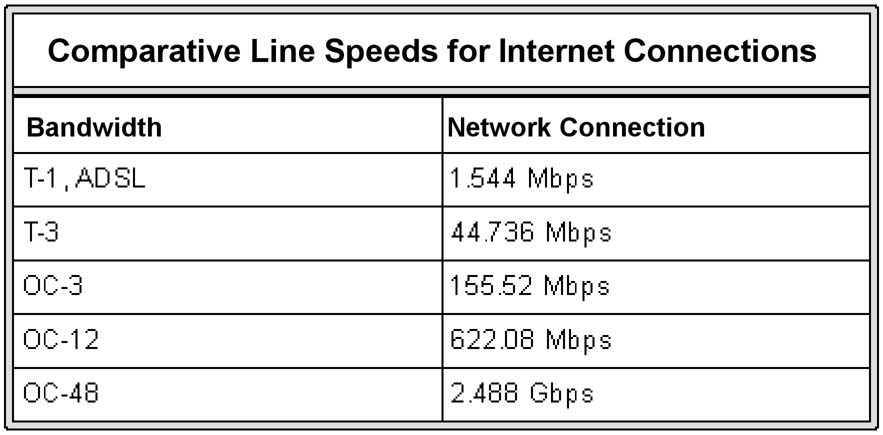


Basic Network Infrastructure

Sizing Bandwidth

We begin with the element of your network infrastructure that is closest to the Internet: bandwidth. Then we work “back” into your system, so to speak.

If your data center is in-house, you need first to determine how much bandwidth your site requires. That is likely to mean choosing whether your site needs one or more T1 lines, T3 lines, or DSL. While T1 and T3 lines are more typical, some symmetric DSL lines provide three to four times the bandwidth of a T1 line.



Comparative Line Speeds for Internet Connections

Bandwidth	Network Connection
T-1 , ADSL	1.544 Mbps
T-3	44.736 Mbps
OC-3	155.52 Mbps
OC-12	622.08 Mbps
OC-48	2.488 Gbps

As you size bandwidth, you should also consider sudden, predictable increases in usage. For example, a sports site might have such a spike at Super Bowl time; a golf site might have one during the US Open. If anything in your business model predicts a spike, factor it into your decision.

Many co-location facilities provide a 10-MB Ethernet drop. If that describes your situation, you should pursue an agreement with the facility that allows you to upgrade easily to a 100-MB or even a 1-Gb drop.

At the risk of being repetitive — see Building Block 1: A Data Center — be sure to include backup bandwidth for failover.

Routers and Switches

Your bandwidth settled, the first connections to your system are routers and switches. Routers link the system to the Internet and connect networks on your system; switches provide connections within the same network. Of these two kinds of connections, routers require more discussion.

Router Issues

To be sure, routers deserve several articles to address properly, but here are the fundamental issues, which apply whether you're handling the routing in-house or outsourcing its management.

Router Basics

You need routing capability if you have more than one network address. Most ISPs have routers — you can simply have the ISP drop a wire to your system and leave them to handle the basics — but that is likely to be insufficient for the complexity of your system.

The actual degree of routing capability you need varies with your system architecture. If you have a multitiered architecture and/or load balancing, for example, you probably need more sophisticated abilities.

Redundancy: Failover Philosophy and Logistics

Whether you rent a cage at a co-location facility or run a data center in your own building, you have to decide whether you want two different bandwidth feeds with two different routers.

Therein lies a question of failover philosophy: Routers are expensive, and don't fail as often as other devices, for example, disks. For failover, you may decide in favor of building two networks using different ISPs but decide against a second router. Ultimately, your decision depends on your level of comfort with redundancy and with risk, which should lead you to ask: "At what point do I stop adding equipment to this system — equipment that increases the complexity of the system, which in itself can cause failover?"

If you decide to have a second router with a second feed, one basic issue is determining how your routers handle failover, including whether or not they require additional network cards to enable failover.

Security: Static versus Dynamic Routing

You must choose between static and dynamic routing. Of course you want the router to act as dynamically as possible, to ensure that a packet gets routed to the proper part of the network. But spoofed routing is a concern, and static routing may well be a good means of avoiding that security problem.

While your decision depends on how you set up the router, you may find it best to use the default settings. A cautionary note: If you use static routing and have failover, you must check that the routers support a failover protocol that works with static routing.

Network Cards and Connections

The routers direct traffic to the networks on your system, which should distinctly separate public and private use. For that purpose, it is important that all machines have two Ethernet cards:

- The first card for public/front-end access, which should be well protected with a firewall and should allow truly limited access on specific ports.
- The second card for a back-end, private network, a discrete network that should not allow communication or traffic with the public network. The private network enables intercommunication between the various machines on the network as well as provides secure access for the developers. Access to it could be through a dedicated private line and/or a VPN tunneling through the Internet.

For better performance and security, it is especially effective to connect your database machine to the Web server on the back-end network. In this way, the public end has less network traffic and no access to the database.

An important note: However you back up data on your system, your concerns for good performance should lead you to use the back-end network to run the backup. Backing up data creates a lot of network traffic. You want this traffic to be on the back-end network rather than the public front-end network.

Virtual LANs
In some cases, you may have only one Ethernet card on your machines but create a second, virtual network on it, which will be non-routable. If you find it necessary to do so, you should expect that the V-LAN will decrease the performance of your system.

Firewalls

Next in, the firewalls. No one argues the importance of firewalls, which usually sit behind the router and in front of the load balancer. But people do argue about which firewalls to choose, especially the degrees of security and performance they provide.

Comparison of Software and Hardware Firewalls

Software Pro	**Hardware Pro**
Lower cost	Easier to install and administer
Easier to upgrade new versions	More throughput
Supports a greater number of dynamic protocols, such as RealAudio, NetMeeting	Greater scalability
Offers additional security applications, such as virus-scanning, intrusion detection, and content-monitoring	

Hardware firewalls commonly reduce performance unless you spend considerably more for a higher-end solution. You may therefore decide to create firewalls on the individual servers. Or you could choose still greater security, selecting a hardware solution as well as hardening — that is, securing via firewalls — the individual servers.

In addition, router configurations can add to the collective firewall capability by providing packet-level filtering to your network.

Load Balancing

If more than one machine is performing the same task on your system, load balancing is essential for scalability, reliability, and security. It is most common to load balance a Web server farm, for which you purchase a load balancer (a hardware or software solution). The process of load balancing, however, may take place on application servers and even between multiple database machines, which either have the ability to balance built into their software or require code to be written. For example, an online brokerage business would load balance a cluster of application servers to handle buy-and-sell transactions.

Load-balancing considerations include:

- **Kinds of load distribution**. You need first to decide whether to use load balancing to distribute overall load on your Web or application servers, or to distribute demand according to the kind of task to be performed. If, for example, you have a variety of dedicated applications and hence different application servers, you might load balance according to the kind of application the user requests.
- **Geographic load balancing**. If you have multiple data centers, you should consider an altogether different kind of load balancing, geographic load balancing. Geographic load balancing distributes load according to demand, site capacity, and whichever site is closest to the user. Moreover, if one center should go down, the geographic load balancer provides failover ability.
- **Location on your system**. But the most common form of load balancing — load balancers on Web farms — requires still more attention. In the sections on load balancing to follow, we focus exclusively on them, beginning with their location on your system. You place hardware load balancers in front of the servers and behind the routers because they direct routed traffic to the appropriate servers. Software solutions reside on the Web servers themselves. With software solutions, one of the servers typically acts as traffic scheduler.
- **Packet-reading abilities**. A good load-balancing solution is able to read the headers and the contents of incoming packets, enabling you to balance not only according to brute load but according to the information within the packet, including the kind of user and the type of request. That allows you to identify privileged users and to direct requests to servers handling specific tasks.
- **Performance**. As you evaluate load-balancing performance, be sure to investigate how dynamically the load balancer communicates with all the servers on your cluster — whether the scheduler pings each server or creates “live” agents that reside on the servers. You should also examine how the load balancer parses TCP packets, paying particular attention to how quickly it can process a packet. Some load balancers will be more efficient than others. That efficiency is typically measured in throughput.
- **Hardware or software**. Load-balancing solutions can be easily divided into software and hardware options, but these alternatives are less easily decided between.

Some experts argue that hardware solutions have better performance and reliability. Others note that the hardware solutions require you to buy a second load balancer if you want redundancy, while the software solutions automatically shift distribution to another server on the cluster if there is a failure.

Building Block Three: Server Hardware

And now the appliances: You've chosen your house, identified and established your connections; finally comes the hardware. If there are scads of issues about infrastructure, there are infinitely more when it comes to server hardware. But in this section, we provide a framework from which to evaluate those very issues. Once again, the idea is to offer a view from above, a guide that allows you to concentrate on the welter of details about server hardware as they arise, comfortable that the larger-scale concerns have been addressed.

In this block, those larger-scale issues divide into the following areas:

- Four areas of importance for all servers
  - Matching servers to content, recognizing the importance of size
  - Two-tiered versus three-tiered architecture
  - Matching servers to application architecture
  - Allocation, load balancing, and Web farms
- Sizing servers
  - Server configurations
  - Disk configurations
  - Memory
- Power supply for servers

Four Areas of Importance for All Servers

Before and while you explore hardware configurations, you should consider four factors that are important to all servers.

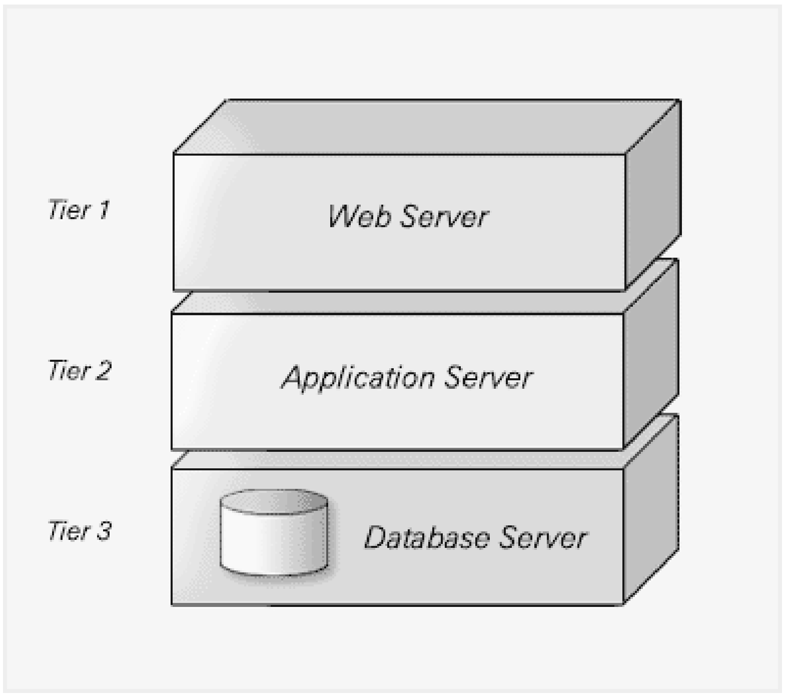
Matching Servers to Content, Recognizing the Importance of Size
Choosing servers first depends on what you are serving. Static content requires fewer CPUs than dynamic content, which uses such technologies as JavaServer Pages[tm] (JSP[tm]), Active Server Pages, and CGI scripts. The general rule — which will be implied or repeated throughout this building block — is that content underlies many of the choices you will make.

To be sure, the size of the site matters, especially for server configuration. Servers with single CPUs are inadequate. A single CPU can handle a decent amount of traffic, but with a spike in load, its performance becomes erratic. To gauge the effect of size, consider the following example. At Sun, the Java Developer Connection[sm], Solaris Developer Connection[sm], and Dot-Com Builder share engineering resources. The three sites have a large but not huge load. The group runs a dozen four-CPU servers delivering Web pages, chat, forums, mail, and search, as well as four six-CPU servers covering back-end operations, from database to search indexing. This server count includes failover machines for all servers.

Some super-large sites use as many as 200 CPUs on their application servers alone. (We define a medium-sized site as one that serves anywhere from 100,000 to 1 million page views per day. A large site serves from 1 to 10 million page views per day, and any site that serves above 10 million page views per day qualifies as super-large.)

But the size of your site is not the determining factor for server configuration. As the next two sections imply, content is often the most significant and underlying influence.

Two-Tiered versus Three-Tiered Architecture
Although multilayered systems are commonplace, it is worth considering what best fits your site rather than assuming you should create multiple tiers.



Three-Tiered Architecture

Occasionally, you will decide on a Web server alone. Most often, you will choose between these two options:

- A Web server or server farm with a database, where dynamic content is created by, say, JSP pages
- A three-tiered configuration of Web server, application server(s), and a database back end, which is typical when you need to integrate legacy systems or perform transactions. (For the purposes of our conversation, “three-tiered” represents any *n*-tiered network.)

You should determine whether to use multiple servers according to what gives you the most functionality, aids scalability, and facilitates administration and security: What works best now and allows you to grow in the future. That decision depends in part on whether your site is driven by content or by dynamic applications.

If you aren’t using application servers and don’t plan to, two tiers make sense. In general, however, three-tiered solutions are more scalable than two-tiered ones and are probably the best choice with complex applications or legacy systems.

Matching Servers to Application Architecture
The kind of servers you choose varies according to the architecture of your system and the needs of each tier. (And content is once again the underlying influence: It affects how tasks are divided on the site and how much load exists on the server, application, and database layers.) Architecture determines still more precisely the number of CPUs needed and the size of the server in each tier, including RAM, server configuration, and disk configuration.

At the most basic level, you should consider separate machines for the following:

- Multiple Web servers (static content, JSP pages, ASPs, and CGI scripts typically run on each machine.)
- A database machine, if you have a database, with the appropriate number of CPUs for the size of the database and your performance requirements
- Application server or servers in the middle, to run anything from e-commerce to bank balances, according to the kind of site you have
- LDAP server to perform user authentication
- Instant messaging server
- Forum/discussion group servers
- Personalization servers

Note well: you're not done buying machines once you make these decisions about your production environment. You need to create development and staging environments that replicate the production area (Building Block 7 addresses development and staging environments). Replicating, however, is unlikely to mean duplicating hardware because it's too expensive. Nonetheless, it is important to duplicate enough of the production environment that you can reliably test whether the site will work once you go live. The closer you come to exact duplication, the more you can predict the performance and scalability of the production environment.

Allocation, Load Balancing, and Web Farms
Your general considerations of server hardware must also include allocation, load balancing, and Web farms.

For load balancing and Web farms in particular, scalability is an important, underlying issue. Effectively, you must choose between vertical and horizontal scalability: whether to build more robust applications and to place them on bigger machines, or to worry less about the efficiency of each application and to scale by placing multiple copies of the application on separate, smaller machines.

Allocation
Certainly, the amount and kind of load on each tier requires different kinds of machines. The difference between what an instant messaging server and a Web server needs is a useful example. A Web server typically provides many, short-lived network connections, while an instant messaging server has fewer but longer connections. For the Web server, therefore, the number of connections per second is most important, while for an instant messaging server, the number of simultaneous connections is of greatest value.

Load Balancing
Decisions about load balancing become more concrete when you consider server hardware. Deciding if it makes sense to load balance your Web servers or to enable your application machines to load balance means deciding whether it is better to select a couple of larger machines or to load balance a group of smaller machines. The answer depends on what hardware is readily available to you and the kind of workload the machines will shoulder.

Web Farms
Your decision about load balancing is often a decision about creating a Web farm. Weigh the advantages and disadvantages of a Web farm versus a single machine (presumably with a number of CPUs equivalent to the sum of the CPUs in the farm). Multiple machines provide you with redundancy and protect you better against downtime. But a farm of less expensive machines requires greater investment in resources to maintain them, and if you choose too small a machine to farm, the result won't be

satisfying: What you save initially on less expensive machines will be noticeably smaller than what it costs you in added resources to maintain them. For this reason, we safely predict that few of you will select eight single-CPU machines over four two-CPU servers.

Sizing Servers

These general considerations in mind, you should match the function of the servers to the following hardware characteristics:

- Server configuration
- Disk configuration
- Memory

Server Configuration

Perhaps the most complex decision to be made about hardware is server configuration, and that depends on answering these two questions:

- What kind of machine(s) are you comfortable with?
- What are the machine(s) doing?

Comfort is an unspoken but enormously influential factor. You're likely to save a lot of time and resources working with machines you are familiar with. It should be an important part of your choice.

Yet matching server configuration to function requires a more detailed explanation. While servers can have up to 64 CPUs on one machine, cost and your precise need will probably result in your choosing anywhere from two to eight CPUs, including two to four CPUs on Web servers and four to eight CPUs on database machines. Server configurations require different considerations if they are database machines, Web servers, or application servers.

Database Machines

Unless you're a super-large site, where you might have a database farm, you're likely to have only one machine for your database layer because they're expensive. You can't split up a database over multiple machines, so your single machine must have multiple CPUs to do the job.

Web Servers

Web servers have the opposite need for configuration. You can get better performance from multiple Web servers with small numbers of CPUs rather than a single machine with many CPUs. A Web server farm increases the number of TCP/IP connections and lessens the amount of traffic on a single network card. Multiple Web servers can also have multiple disks, which will improve I/O performance. They are more scalable — which makes it likely you'll load balance — and they are less expensive to buy than a single machine.

Application Servers

Like database servers, application machines function as discrete logical units. On larger sites, you might have a number of (balanced) application servers.

Disk Configurations

Like server configurations, disk configurations vary greatly according to what gives you the best performance. The key criteria are how much data storage is needed, how best to access that data, and the kind of content being stored. This discussion of disk configurations has three parts:

- Storage solutions
- An instructive example
- RAID, striping, and mirroring issues

Storage Solutions

The storage configurations of your servers divide into three general categories:

- Large disk array. Larger machines have a separate box that handles multiple disks, a disk array that can run anywhere from gigabytes to terabytes. Physically connected to only that one server, a disk array offers maximum performance. It is the likely choice for a database machine.
- Network storage device. In this configuration, the disks reside on the network and are available for any machine to access their information. Writing to a network storage device is less efficient. This device works best where the stored information is read-only, so static content is served well by this solution.
- Multiple disks. Rather than a separate array, this configuration typically uses one to three disks in a machine.

An Instructive Example

While disk configurations are easier to select for a database machine than for most other servers, they are harder for a Web farm. Consider a Web farm comprising machines that have two CPUs serving mostly static content. For such a farm, you need to decide between the relative advantages and disadvantages of multiple disks on individual machines and a network storage device.

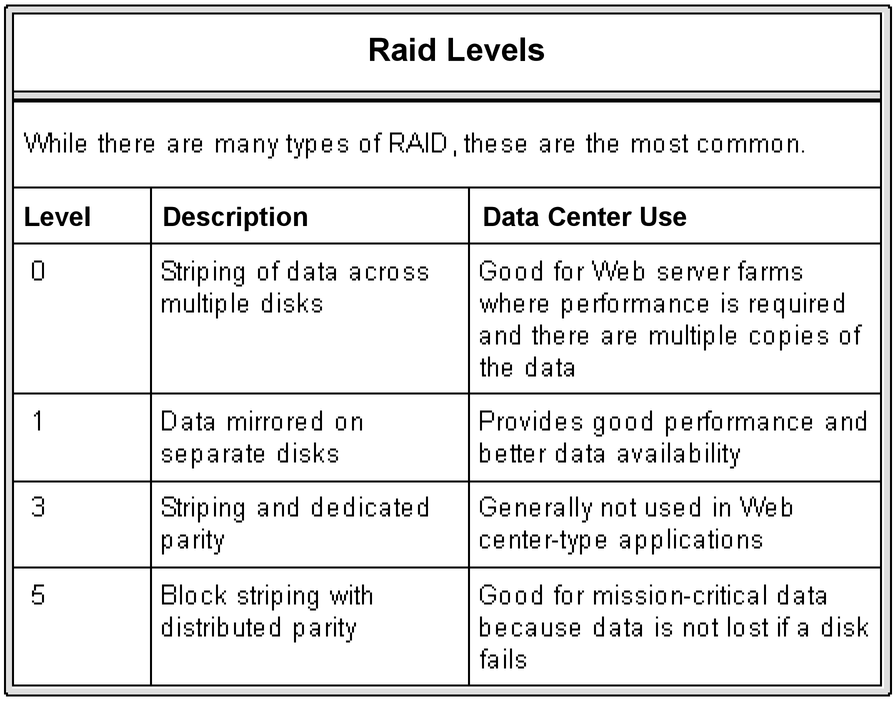
Weighing these two options reveals much about the compromise between performance and reliability:

- Multiple disks require you to replicate content on each machine, which gives you good performance but requires constant maintenance. You need to ensure that the content is the same on each machine, updated as updates occur. Therein lies an often-overlooked problem: This solution requires you to pay special attention to machines that are down for maintenance while a content change occurs. Offline machines need to be modified once brought back online.
- The alternative, a network storage device to which all machines have access, solves the problem of maintaining up-to-date content and is a little easier to maintain, but its performance is not quite that of disks on individual machines.

RAID, Striping and Mirroring

All three kinds of storage solutions require you to decide on:

- RAID states between 0 and 5.
- Striping data across several disks to improve read and write performance.
- Mirroring data to give you multiple copies of files on disks, which increases reliability and improves your ability to recover data but at a cost to performance.



Raid Levels

While there are many types of RAID, these are the most common.

Level	Description	Data Center Use
0	Striping of data across multiple disks	Good for Web server farms where performance is required and there are multiple copies of the data
1	Data mirrored on separate disks	Provides good performance and better data availability
3	Striping and dedicated parity	Generally not used in Web center-type applications
5	Block striping with distributed parity	Good for mission-critical data because data is not lost if a disk fails

RAID Levels and Corresponding Functionality

Once again, your choices balance performance and reliability, and these balancing acts vary according to the content and the machine. The key criteria will be whether the data being stored is read-only or write-modified, and whether or not the storage solution should be hot swappable.

The differences between database and Web farm disk configurations are instructive here. The disk array of your only database machine should be mirrored and hot swappable, allowing you to take a disk out of the array without turning off the machine. In contrast, a Web farm — on which machines have their own disks — should be striped, but it does not need to be mirrored because you have other copies of the data residing on the different machines of the Web farm.

Memory
RAM is simply one of the most important areas of your hardware. While it is moderately expensive, it powerfully affects performance on your servers. Therefore, the more memory, the better. Once you evaluate your needs, you should not be stingy in the amount of RAM you provide your servers. Typically, you want to pay close attention to the memory requirements of your application servers, where RAM especially influences performance. Later, once your site is up and running, you can add still more RAM if you find that your machines are using a lot of swap memory — that is, accessing the hard disk to perform.

Sample Output from the vmstat Command

```
procs      memory                 page
r b w      swap       free        re    mf    pi    po    fr
0 0 0      549664     20848       5     360   3     379   1027
0 0 0      549664     21760       2     0     17    0     0
0 0 0      549296     21384       0     126   0     1     1
0 0 0      549264     20616       0     140   750   33    33
0 0 0      548688     17368       6     262   25    56    56
0 0 0      549664     17800       5     0     0     0     0
0 0 0      549480     17616       0     125   0     8     8
```

Here are some definitions for the output of the vmstat command:

procs — Report the number of processes in each of the three following states:

r	in run queue
b	blocked for resources I/O, paging, and so forth
w	runnable but swapped

memory — Report on usage of virtual and real memory.

swap	amount of swap space currently available (Kbytes)
free	size of the free list (Kbytes)

page — Report information about page faults and paging activity. The information on each of the following activities is given in units per second.

re	page reclaims
mf	minor faults
p	kilobytes paged in
po	kilobytes paged out
fr	kilobytes freed
de	anticipated short-term memory shortfall (Kbytes)
sr	pages scanned by clock algorithm

Configuring your database is an art unto itself, but at the most basic level, simple decisions about RAM can be quite effective.

If, for example, your database is primarily read-only, you want the machine to have enough RAM to locate the database in it if that is practical (which it won't be if your database is huge). In such a circumstance, RAM optimizes the efficiency of your database machine. Because RAM is much faster than the hard disk, the database performance dramatically improves if it is located there.

If you write often to the database, then locating the database in RAM is unimportant and expensive.

Power Supply for Servers

Beyond sizing your servers, their power supplies deserve a brief comment. While thereis not a lot of variation in power supplies — your machine manufacturer will provide the appropriate unit for your machine — you should ensure that your power supply has two abilities:

- Redundancy. You don’t want a single transformer failure to force your system to go down.
- Hot-swappable supply. You should have the ability to exchange one power supply for another without having to turn off the machine.

Building Block Four: The OS and Database Layer

Software is its own house within a house of hardware, and it needs its own foundation: the operating system and, to a lesser extent, the database layer. Ultimately your choices of an OS and database will be strongly influenced by the ones with which you are most familiar and comfortable. But to state the obvious, there is no such thing as simple selection in this case. Choosing an OS and database is only the beginning of your efforts to establish these layers. Creating a software foundation is no less an art than selecting and configuring hardware.

Nonetheless, the operating system and database have readily identifiable requirements. Security is the paramount issue as you establish your OS and database, which should become abundantly clear in the following sections:

- Securing the operating system
  - Hardening the OS upon installation
  - Tracking and patching
  - Secure Telnet and FTP
- Securing and configuring mail
  - Securing your mail server
  - Configuring your mail server: mail-routing issues
- Selecting and configuring a platform
  - Selecting a platform
  - Configuring the platform
- Choosing a database
  - Integration
  - Redundancy
  - Databases with multiple data centers

Securing the Operating System

No matter what operating system you choose, security is the greatest concern. And so we begin there. The means for hardening your operating system depends, of course, on the OS. But we identify three principal ways in which you must both secure and remain vigilant about securing your operating system:

- Hardening the OS upon installation
- Tracking and patching
- Secure Telnet and FTP

Hardening the OS Upon Installation

As soon as you install the operating system, you'll need to improve its security. Any OS has security flaws that must be attended to. For example, with systems that use UNIX® or variations of UNIX, you should harden the OS by removing everything from it that is not essential for what you're doing on the site. In effect, you are ridding the OS of unnecessary applications that would give hackers tools with which to compromise your security. Among other things, you should remove compilers and strip out all components related to X Windows, the windowing system for the Solaris[tm] Operating Environment and other UNIX-based machines.

Tracking and Patching

Hardening the operating system also requires you to have all current patches and to remain vigilant about the latest security holes to be discovered in operating systems, applications, and protocols. You should regularly track CERT advisories on both patches and newly discovered holes.

Secure Telnet and FTP

Telnet and FTP are necessary but must be used in a secure way. You therefore should use SSH and secure FTP at all times, and, if possible, open ports for these services only on the private, secure network of your system.

Nonetheless, some sites use TCP wrappers, which are not the most secure option, but are worth considering as a means of using and restricting FTP and Telnet. TCP wrappers have the added advantage of reducing the number of rules your firewall must filter. With fewer rules, the firewall is less likely to slow the processing of packets and to degrade service for an end user. TCP wrappers also allow you to control access to such services on a per-host basis, which is a quick and easy means of adding or changing hosts.

Securing and Configuring Mail

Your ability to receive and to send mail is not only a vital function but also a means for someone to wreak havoc on your network.

Securing Your Mail Server

If you're serious about security, you should run a mail server only on a separate machine dedicated to mail serving. The reason for doing so is simple enough and an important first principle: All mail applications have holes; the point is to minimize the holes as well as their effect. We have already addressed the importance of sequestering each application on its own piece of hardware (and each application on a machine in a chrooted environment). This principle bears repeating here: If you sequester your mail server, then anyone who manages to hack into the mail server will have much more difficulty trying to damage the system beyond that server.

Configuring Your Mail Server: Mail-Routing Issues

At the same time you secure your mail server, you must enable the system to send and to receive mail for all of your applications. After all, users send mail and requests to various parts of your site while your system generates important mail for internal use. Make sure that:

- Applications that need to send, such as confirmations, are able to do so. All mail sent from an application server and returned bounced is routed not to the application server but to a mailbox where it can be read. To provide these abilities and to maintain security, you should forward all mail from each application to the mail server — which then connects to users rather than the individual applications doing so. To ensure proper routing, you should configure the "from" portion of the mailer headers so that all replies to mail sent from applications go to some place where they will be read.
- If you should use `sendmail` to enable your Web servers to mail, configure it so that it runs only when called upon to send mail, after which it immediately shuts down.
- If your system is UNIX-based, then system-generated mail such as `cron` errors or system messages should be routed not to the root on the individual machine — where no one would see it — but to some administrator.

Selecting and Configuring a Platform

Selecting a Platform

In the simplest of terms, to choose a platform is to choose between UNIX (or a UNIX variation such as Linux or Solaris) and Windows. In spite of the passion with which some people voice their preference, there is no wrong selection for a platform, though you should always carefully review current security guidelines and warnings in the course of the selection process. The most important factor in your decision-making should be which platform you are most expert and comfortable with. For example, you should consider which programming languages the platform uses — Java[tm], C or C++, Visual Basic, or Perl, among others — and with which of these programming languages you are comfortable.

But you should also consider the following as you make your choice:

- Software selection: What software is available for that platform, and what database servers, Web servers, chat servers, and application servers do you want to run? Windows will of course run different software than UNIX. But each UNIX variation will have different software available.
- Hardware selection: What hardware is available for your platform? You should investigate how expensive those machines are. You also should find out whether the platform runs on big or little machines, if it is scalable to large machines, and whether it clusters well.

Configuring the Platform
Your configuration of the platform should make security the first priority and performance nearly as important. Configuring a platform for security and performance is strongly platform-specific and hence resistant to generalization. But consider these properties and requirements as you configure:

- The number of available TCP connections
- The number of open file handles
- The memory allocation for each application (bearing in mind the large demands on memory your database will make)

Choosing a Database

The repository of information, database software is likely to be the centerpiece of any large, dynamic Web site. While you may have the apparent luxury of several Web servers, cost considerations will probably restrict you to only one database. Which database you choose — for example, Oracle, Sybase, SQL server, mySQL, Informix, or DB2 — should depend on the performance, reliability, and cost of the database. It should also depend on which database you have experience or expertise with and, where applicable, which databases your company is already using.

Integration

This last consideration is particularly important if you have legacy applications, such as HR, accounts payable, and financial databases. Your ability to integrate the database you choose with your existing applications may well be the determining factor. (For a few of you, the ability to migrate from one database to another may also be important, in which case you'll want to investigate how easily and at what cost it can be done. In general, migration requires rewriting a lot of code.)

Redundancy

Beyond selecting a database, you should also provide some failover ability or replicate the software and data.

Failover

With databases (and for that matter, with application servers, mail servers, and all other layers of your system), redundancy requires you to make a careful calculation: What amount of downtime is acceptable on your site? To answer that question is to decide among three types of standby:

- Cold: a manual process that takes hours to restore function
- Warm: a semiautomated process that takes minutes to restore function
- Hot: an automated process that is seamless or takes mere seconds to restore function

This decision is a calculation because cost factors will play an important part in your choice. To keep your network available 95 percent of the time will cost you so many dollars. To keep it available for 100 percent of the time — to achieve that final 5 percent — is likely to cost you 10 times that amount. You therefore need to measure the cost of downtime on your system against this exponential difference in the cost of providing standby. If you are an online brokerage firm, the cost of five minutes of downtime may be millions of dollars. In that case, you probably should choose hot standby. But if your business is likely to lose only $1,000 in those five minutes, you may want to consider a less expensive alternative.

Replication

While it is less likely, you might provide redundancy by creating a separate database. An additional database requires you to shoulder not only the additional hardware and software costs but the increased demand on resources for managing a more complex network.

Databases With Multiple Data Centers

Replication may also be something you consider if you have multiple data centers. There, too, you need a kind of redundancy — database ability at multiple sites. Most often, however, businesses with multiple data centers do not replicate under these circumstances but enable all of the data centers to draw from a single database machine at one of them. If you decide to buy additional databases, database makers have real-time replication software to synchronize them, but that software is likely to be expensive.

Building Block 5: Identity and Security Policy

From OS and database security, we move to the means by which you identify users and secure your system as they use it. Your site needs a general identity and security policy that focuses on the user because identity and security are obviously important and easily botched.

You should design that identity and security policy early in the process of building your Web site, so that all parts of your system work together to create a coherent and unified policy. Doing so reduces security risks. It also prevents a common problem: requiring users to repeatedly identify themselves as they navigate different parts of the site where no additional identification should be necessary.

Swiftly defined, the purposes of such a policy are to manage the user experience smoothly and to protect the system thoroughly. Those purposes translate into the following goals:

- To manage user identity, authorize access, and store access information, such as user passwords.
- To identify the types of information you have on your site, determine the value of that information, and create from that evaluation a thorough understanding of your security risks and an appropriate policy to address them.

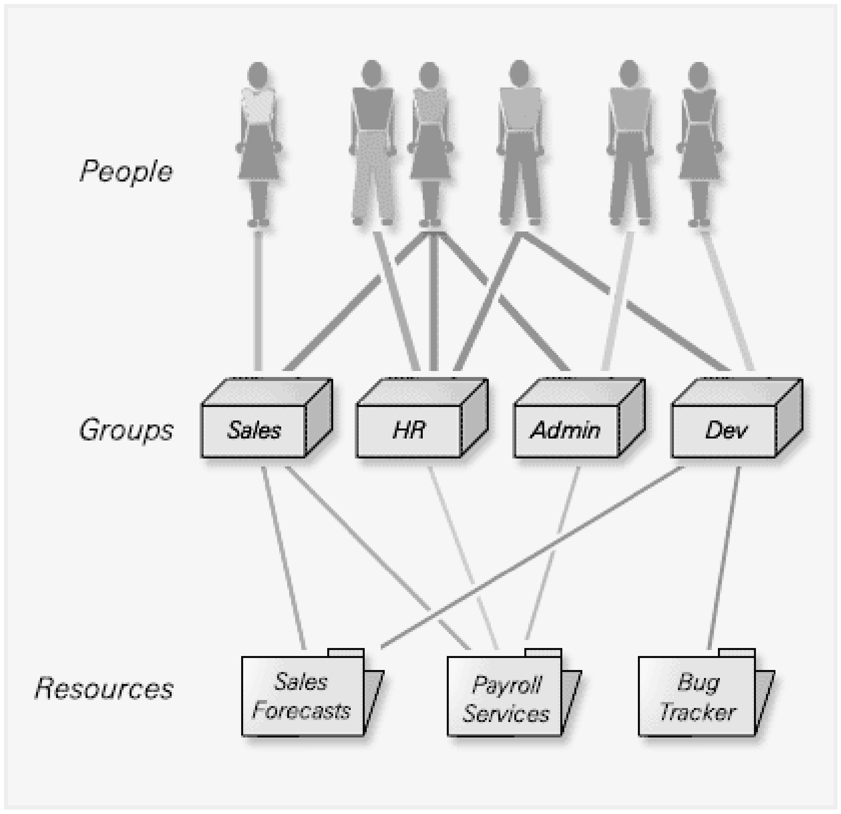
We consider identity and security policy in the following sections:

- General abilities of an identity and security policy
- Authenticating: passwords versus digital certificates
- Maintaining persistence and entitlements
- Integrating the policy: identity issues
- Integrating the policy: security issues
- Managing entitlements on your site

General Abilities of an Identity and Security Policy

For your identity and security policy to be effective, you need some basic abilities, which can be divided into three kinds:

- Identifying a user and providing critical access.
- Maintaining the user's identity throughout the session, which involves both creating a means for session persistence and repeatedly validating the user during the session.
- Using that validated identity to grant access to the appropriate parts of the site according to the privileges of that person.



Identity and Security, People, Groups, Resources

Authenticating: Passwords Versus Digital Certificates

To identify your users, you will probably choose between two forms of authentication:

- User names and passwords
- Digital certificates

User names and passwords are of course simpler to use; digital certificates are more secure but more technically complex for both users and administrators. Your decision between these two forms of authentication depends on the kind of site you have and the kind of users. For example, a bank Web site needs high security but serves a population of technically unsophisticated users: Bank sites usually opt for user names and passwords because of their users. A Web site for military intelligence is likely to choose digital certificates, which increases its security but requires resources for creating and managing the certificates as well as for training its users.

Maintaining Persistence and Entitlements

To establish persistence on your site, you should use cookies or session identifiers embedded within the URL. Whichever means you use, it is important to authenticate by the same mechanism on all parts of the site, so that session persistence is maintained and users do not have to authorize each time they use an application.

As users move around the site, you must identify them and allow them access to more secure areas according to their privileges. In effect, you must create entitlements, a means of providing privileges to the individual once he or she is authorized, whether those privileges are for reading sensitive content or for administrative functions. The practical means for managing entitlements is addressed in the last section of this building block.

Integrating the Policy: Identity Issues

Authenticating in the same way across your Web site is a start toward integrating identity policy throughout your network. You should also integrate that policy into the dynamic parts of your site, across all your machines and applications.

Your database, the repository of information for your site, deserves special attention. For integrating security policy with your database, two alternatives come readily to mind:

- You can use LDAP to identify, authenticate, and grant or deny access (effectively using LDAP as the data repository).
- You can use your database directly.

You must then evaluate how well each alternative integrates with the security policy you are trying to establish and when to use those alternatives.

Your decisions about LDAP or a database — where and when you use each to integrate your identity policy — should be strongly influenced by the kind of site you have. For example, LDAP performs well with data that changes infrequently, such as a phone book or log-in information. Directly accessing a database works well with data you modify often, such as shopping carts for e-commerce, transactions for a bank site, or online stock trades.

But the choices may not be so clear-cut. If you already use the database directly for looking up user authentication and storing access privileges, then continuing to do so may make more sense than implementing LDAP. That means you should also be influenced by which data repository you have the most experience with and how well each of these alternatives works with your legacy applications.

Integrating the Policy: Security Issues

Integrating your policy is a matter of not only identity but security. You want to guard against a hacker using the operating system and hardware of an individual server, such as an application server, to attack the network beyond that individual server. In practical terms, that means all applications should be sequestered on separate machines, and each application should run in a chrooted environment.

You also want to protect against a hacker taking on the identity of a specific user to steal or otherwise damage that user's account or the services available to that user. You should always encrypt user names and passwords with SSL to prevent hackers from using packet sniffers to examine user names and passwords sent in clear text. You should also monitor and protect carefully the database or LDAP server on which you store user names and passwords.

Managing Entitlements on Your Site

The previous sections addressed how users are authorized and gain access to data on your site. But to make entitlements work, you still need to keep track of what privileges each user is granted.

You need a software application to manage the information on your site and establish which pages and what applications require what kinds of privileges. There are essentially two means of acquiring this software:

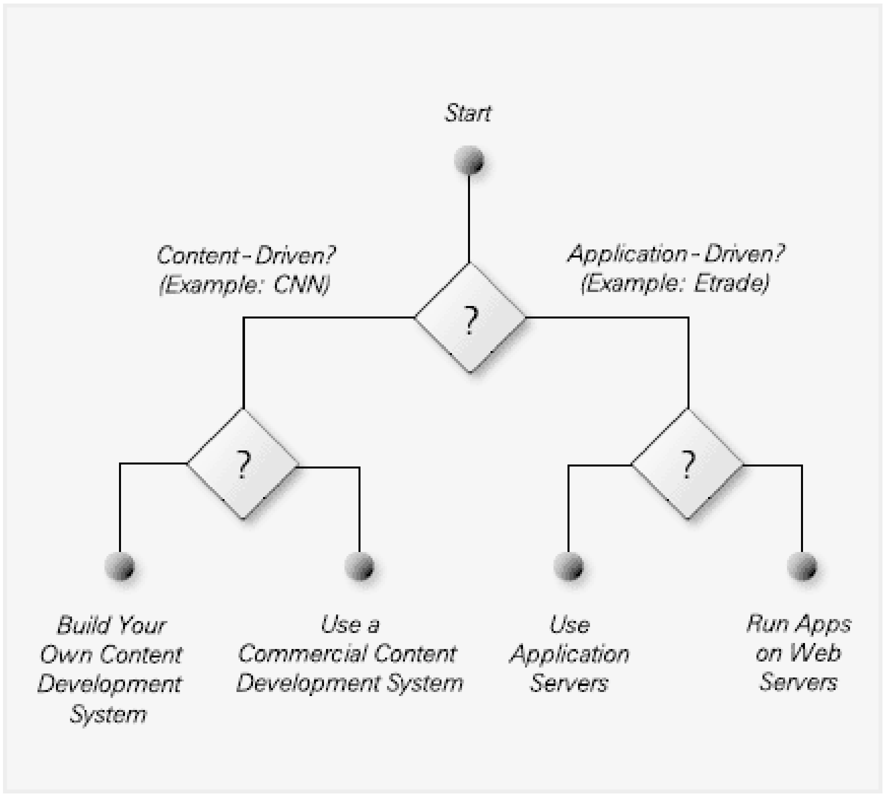
- You can build your own authorization model for applications, using JavaServer Pages[tm] technology or ASPs as you go along, or you can use `htaccess`, which is built into Web servers such as Apache.
- You can install a purchased application or customize it to your needs.

Your choice here depends on the complexity of your site and your users. A bank site has many users with approximately the same privileges. For such a site, the security portion of the application is fairly simple, and you're likely to build your own authorization model: It doesn't require a lot of resources or many changes once deployed, and site personnel aren't actively modifying the pages on the site.

A large corporate Web site, however, is a likely candidate for a purchased solution. Such a site has executives, salespeople, and other users, all with different privileges, many of them with the privilege of changing content. It requires a complex system of entitlements. Building an application for such a site goes beyond the resources of most businesses. A purchased application, whether or not customized, has an easy-to-use interface for managing such complexity.

Building Block 6: Application Servers and Content Management

You've established hardware and software foundations and created an identity and security policy. Now functions on your site need to be addressed. And as discussion turns toward the function of your site, the road divides. On one side are sites that are largely application-driven, on the other, sites that rely mostly on content. Some sites partake equally of both, and if so, the two parts of this building block will apply equally. But many Web sites consider themselves either application-driven or content-based — closer in kind to either E*TRADE or CNN. For the former, application servers are vital; for the latter, content management is essential.



Application-Driven Versus Content-Driven

Application servers and content management therefore demand separate attention, and we address them in the following sections:

- Application servers
  - Application servers vs. applications on Web servers
  - Choosing application servers
  - Building with application servers: consistency
  - Configuring application servers
- Content management
  - Workflow process: the template model
  - Creating documents
  - Regenerating and customizing documents
- A final note on XML

Application Servers

To use application servers is to build distributed applications. In the following sections, we discuss the case for distributed applications and the advantages of using application servers.

Application Servers vs. Applications on Web Servers

You can of course run applications without application servers, in which case each of your Web servers would have a copy of the application you want to make available to your users. If your applications are not terribly complex — if, for example, they are small and don't interact with other modules — then installing applications on your Web servers may be a reasonable solution for the moment. An application that fulfills a simple request for information, such as a simple look-up in a phone directory, is reasonably handled by an application on a Web server.

But for complex applications, application servers are especially valuable. By "complex," we mean applications that have many different modules with discrete functions that interact with each other and must integrate with legacy systems. They might also add functions over time. An online brokerage site uses complex applications with interacting modules for making trades, creating personal trade histories, and generating stock and bond information.

To no surprise, application servers are the wave of the future for large to super-large sites. The reasons are clear enough. Application servers not only distribute applications across multiple machines but also:

- Keep track of users and session states.
- Maintain the database connection pool.
- Provide transaction controls for multiple requests.
- Ensure that the data from different forms gets to the database accurately.
- Cache read-only information that is used repeatedly by different users.
- Coordinate development and deployment when the application requires multiple development teams.
- Facilitate deployment from the development server to multiple application servers on the front line.

Given these abilities, the advantages of an application server are easily summarized:

- Scalability: You have greater scalability for your application because you can distribute it across multiple servers.
- Performance: You can create caching schemes to reduce hits to the database, thereby optimizing database queries and schema designs. As a result, the database can be used more effectively for tables or queries whose results cannot be cached, such as personalized customer data.

If your goal is to create a scalable, professional, and dynamic site, one that frees you from writing unnecessary code for applications so that you can focus on creating the functions of the site, application servers are simply necessary.

Choosing Application Servers
Your choice of application servers may be based on your individual requirements, or it may be constrained by the servers available for your operating system. Where you have a choice between servers, consider performance and ease of implementation.

Application Servers Compliant* With J2EE[tm] 1.2 Technology

ATG Dynamo Application Server 5 for Solaris[tm] Operating Environment and Windows NT

BEA WebLogic Server 6.1 for Solaris, Windows NT/2000, HPUX, AIX, and Red Hat Linux

Borland AppServer 4.5 for Solaris and Windows 2000

Fujitsu INTERSTAGE for Solaris, Windows NT, and Linux

Hitachi Cosminexus Server Standard Edition for Windows NT

HP Bluestone Total-e-Server for Windows NT

IBM WebSphere Application Server 4.0 for Windows NT, Solaris, Linux, AIX,
HPUX Version A Release Level B.11.0, and iSeries

IBM WebSphere Application Server 4.0.1 for z/OS and OS/390

IONA iPortal Application Server 1.3 for Windows NT and Solaris

iPlanet[tm] Application Server 6.0 for Solaris

Macromedia JRun Server for Windows NT

Oracle9*i* Application Server for Windows NT and Solaris

Persistence PowerTier Application Server

SilverStream Application Server 3.7 for Solaris, Windows NT/2000, HPUX, AIX, and Linux

Sybase EAServer 3.6.1 for Solaris and Windows NT

TogetherSoft ControlCenter

Trifork Enterprise Application Server for Windows NT

* *(As of December 2001; to get the latest list, see http://java.sun.com/j2ee/compatibility.html)*

Performance
Some experts say performance should be the deciding factor. They argue as follows: All application servers conform to the specifications of the Java[tm] 2 Platform, Enterprise Edition (J2EE[tm]) standard or of its equivalent on other operating systems, such as Microsoft's COM model. Therefore, your applications run on any server, in which case performance is the most important decision you have to make.

That advice may be particularly well-received by large, well-financed sites, where performance is almost all that matters, and the difference between $10,000 and $17,000 for an application server is less important.

In any case, performance varies significantly between servers, especially in how they scale. For example, a noticeable difference exists in the performance of some application servers in clustered environments when they are sharing the session states of users.

Ease of Implementation
Still other experts argue that ease of implementation should be equally if not more important to your decision. You want to be able to use the server as easily as possible, with a minimum of training. That ease of use may come from the server itself, including the ease of server installation, application deployment, integration with development tools, and maintenance. But that ease may also be the result of your comfort and familiarity with the server.

Building With Application Servers: Consistency
As you begin to build functions on your application servers, commit all the way. That is, use the same programming architecture throughout your site. Don't create half of your functions by writing code in Perl script and the other half using JSP[tm] technology. Instead, develop applications on application servers throughout your site to avoid creating undue complexity in your system, which drains resources and makes managing your applications a nightmare.

Stated as a principle, you should create functions on your site in a consistent manner. That implies you should not only develop Web applications in the same way but use common tools for your system.

For example, use a single user-authentication and access-control module throughout the site. Develop the same method for connecting to the database throughout your system.

Configuring Application Servers
As you configure your application servers, you should simplify the process of developing your applications, freeing you to focus on creating functionality. To that end, follow the specifications of your application servers. Doing so reduces the amount of code you must write and in general minimizes the amount of work needed to develop applications. You aren't required to write code for generic communication issues with the database because the application server automatically handles the storing of data and the transfer of it to and from the database. As a result, you write only the programs for the specific functions you want the application to perform. (Because you cluster a group of application servers, they automatically communicate and load balance with each other, which requires you to write less code.)

Factors in Tuning an Application Server

Reactive

If	regular monitoring of the database server computer reveals that the CPU or memory is reaching maximum capacity —
Then	you should decrease the size of the database connection pools.
If	you notice the database server is showing abundant available capacity, while the application server is showing heavy CPU or memory usage —
Then	you should increase the database connection pool sizes, but only for the applications shown to be spending much time waiting for database connections.
If	HTTP requests to applications on front-end Web servers take too long to be serviced, and the application server computer is showing available capacity —
Then	one possibility is to *increase* the number of threads.
But	if the application server machine is at its maximum capacity, it may be better to *decrease* the number of threads serviced. There will be some delay for HTTP requests, but the application server will run more smoothly.

Proactive

When you anticipate needing increased capacity, you can proactively make tuning changes to the application-server software; for example, perhaps there has been a release of new software more computationally intensive, or you expect increased traffic as a result of a new marketing campaign.

A Final Note

These application-server tuning methods go hand-in-hand with the configuration of the computers. When you notice excessive CPU or memory usage, you must decide whether or not to tune the application-server software to alleviate the problem. Of course, there are other options, such as adding memory to the computer, modifying the amount of available swap space, or migrating the application server software to a more powerful computer.

Setting the Number of Threads

Your ultimate goal is to optimize the performance of the application servers while minimizing demand on the database. To that end, the settings for the number of threads and the database connection pool deserve particular attention. For each application server, you should set the number of threads — the number of requests it can take from the Web server — to optimize performance. You want to maintain the number of threads at the maximum figure that each application server can handle without impeding performance. Set too small, the number of threads creates a bottleneck, hindering each application server from handling the number of requests of which it is capable. Set too large, the number of threads will overwhelm the server with requests, slowing performance.

That maximum number varies according to the hardware of your server, especially the number of CPUs and other details specific to the hardware: The vendors of your application servers can help identify the appropriate number of threads.

Setting the Database Connection Pool
The database connection pool is an invaluable way to keep connections to the database open. Your goal is to configure it to preserve database resources while ensuring that the application server can efficiently process all the necessary calls to the database. You vary the database connection pool according to the number and the complexity of calls on each application server, whether it stands alone or is part of a cluster. A simple call may be a price look-up for inventory, while a complex call would be one that requires the merging of diverse database tables — personal information, address, financial information, and so on — for a large number of people.

Clearly, the demand of fewer complex database calls is equal to that of many more simple calls. To assess the demand on the database from each application server, you need to weigh the kind of calls in the pool and the number of calls, avoiding the obvious extremes. Set arbitrarily at the highest level for the number of concurrent calls, the database connection pool causes your application server to hoard database resources, taking memory and additional resources from the database that other application servers might need. If the pool is set too low, your application server's performance slows considerably.

In general, the database connection pool should be large enough to handle traffic on the application server but small enough to avoid placing unnecessary demands on the database. In practice, you determine that setting by monitoring the CPU memory and utilization rates of your database server.

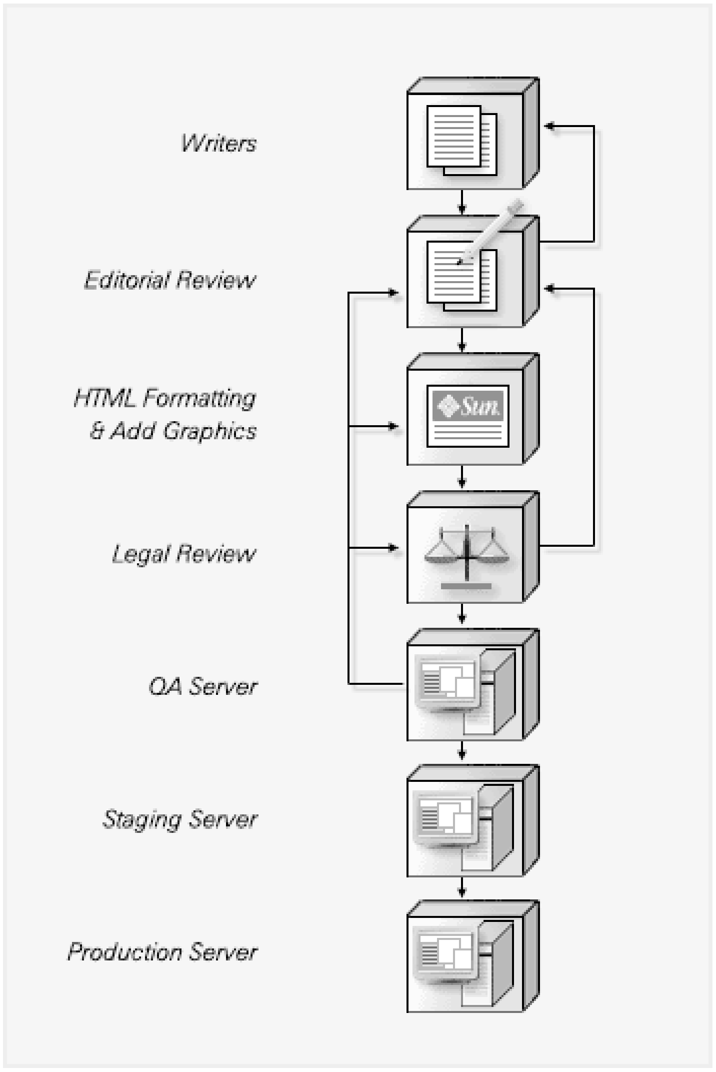
Content Management

Your site may be less centered on complex, interactive applications and more on the publishing of so-called "static content" to users, such as words and images that make up stories on a news site. For such a site, content management is important because it manages workflow, the process of authoring, editing, approving, staging, and publishing content. It is possible to build this software yourself; it is more likely that you purchase it. For that reason, the focus here is on how the software enables function on content-driven sites.

Your goal with content management software is to solve three problems inherent to content-driven sites:

- How to effectively manage hundreds if not thousands of documents that may require updating. You need to create and then control the version of a particular document that is available. You may have to coordinate the work of authors and editors scattered throughout the world, who may not be technically adept (or you can't rely on them being so).

- How to regenerate all the pages if you should decide to change the appearance of the site. It's not unusual to add, subtract, or modify a banner, logo, menu bar, or the layout of your pages. You need to be able to re-generate the content of these pages easily as you make these design changes.
- How to customize hundreds or thousands of pages, either to vary the language of the page or to allow for personalized views of the data.



Workflow from Author Through Production

Workflow Process: The Template Model
Content management software solves these problems through two important functions:

- Workflow process
- Page generation

Whichever software you buy, you want its workflow process abilities to provide a template model for framing and indeed containing the content.

Creating Documents
The workflow module facilitates the creation of content and its movement from the writer to the various editors, regardless of where they are. For each individual article produced as content, this module coordinates development from:

- Authors creating content
- Editors modifying the language of that content
- HTML editors adding graphics or pictures and making the necessary technical additions for Web viewing

The software should allow you to oversee the process by which the content is first written, next approved, and then scheduled for release with its associated graphics or pictures added to the template.

Re-Generating and Customizing Documents

The template model should also solve the problems of regenerating and customizing pages in the following ways:

- Whenever you create a new template, altering page appearance to suit your preferences for banners, logos, or layout, content is regenerated with the new look you seek.
- The template model allows for personalization because individuals or groups of users can have their own templates for the mix of content specific to them. In effect, the individual user or group of users can manage the content delivered to them within parameters you establish.

Page Generation
The second of the two modules in content management software, page generation, solves the problem of version control and coordinates the actual production of the document. The document is generated for review, whether that final review takes place on the development or staging server. If that review takes place on the development server, the document automatically is added to the staging server and — based on the publishing schedule — pushed automatically to the production side. Once at the production server, all versions are synchronized.

Personalization
Content management software can provide you with still more precise and robust personalization abilities: most systems allow you to offer users unique combinations of content based on one of two basic models. Both forms of personalization increase your costs and the demand on your system, but they are worth considering:

- Explicit personalization: Users personally log in and are delivered content tailored to their explicit demands, such as local weather or news.
- Implicit personalization: This delivers content based on the pages you select as you navigate the site. Commonly used in e-commerce, implicit personalization ranges from collaborative filtering to neural networking.

A Final Note on XML

For both application-driven and content-centric sites, XML may play an important function. With application servers, XML has two ready functions. If your site has data to share with application servers on another site or system, you need a way to move data between them. XML is an excellent way to do just this. In addition, XML is often used for deployment descriptors for the application servers — effectively as an installation guide for the application on the application server.

For content-driven sites, XML or customized variations of XML are potentially useful for content syndication — that is, for publishing content on both your own site and on other sites. Some content management systems have XML-enabled content syndication for when you want to license or distribute content.

Building Block 7: Development and Staging Environments

While we have focused attention on the production side of a Web site, there is another, equally important part of your network to build: the environments in which you produce your site. The development and staging environments require their own hardware and a carefully directed means by which all that you develop is deployed to the production servers, from where the site goes live.

In this building block, we address the development, staging, and production environments in the following sections:

- Goals for deployment
- Principles of deployment
- The development server
  - Back-end development software
  - Content management software revisited
- The staging server
  - QA and bugs
- The production server: going live
- Mirroring via manual FTP, Rsync and Perl scripts, or content management software
- The logistics of modifying content

Goals for Deployment

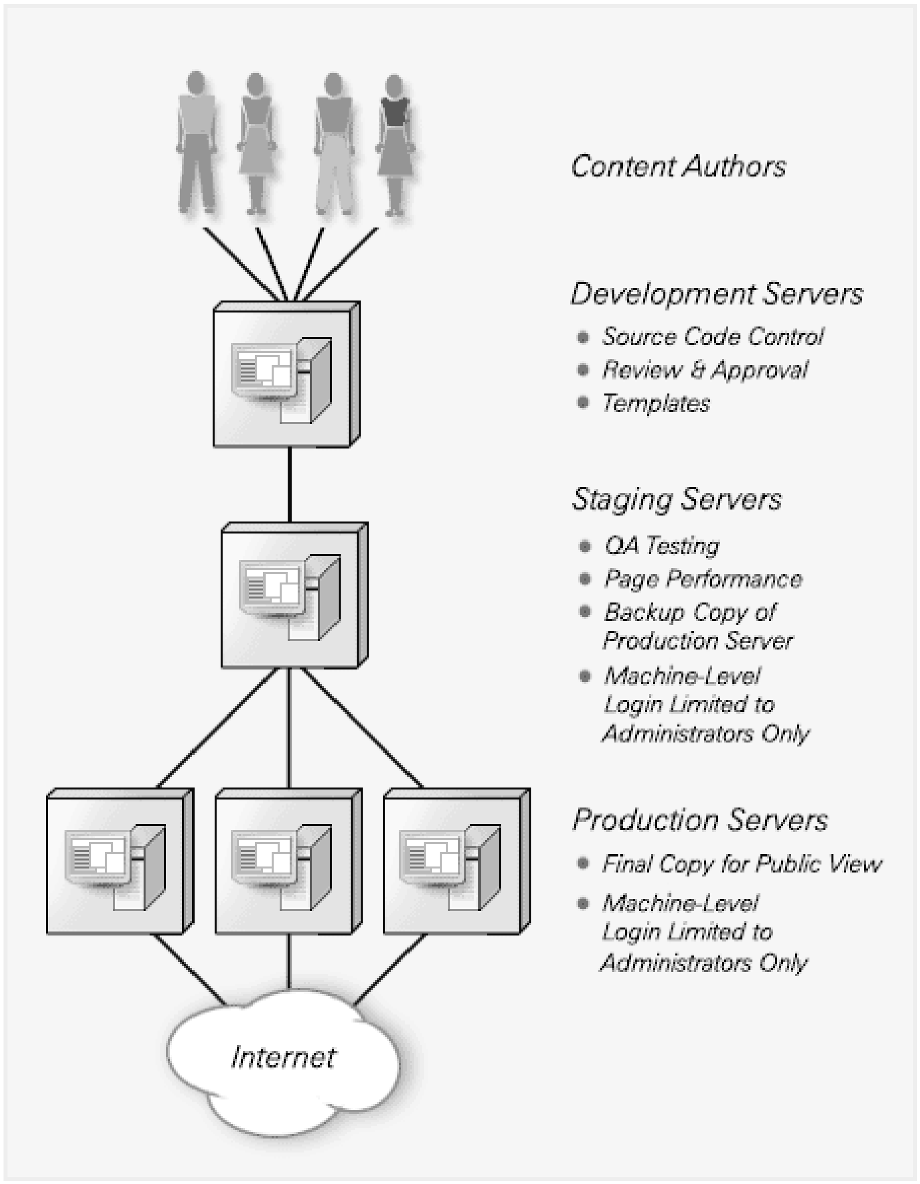
Deployment, which includes the development and staging environments, has an overwhelmingly important effect on the ability to create and to modify a site and on the likelihood a site performs according to your expectations: Without proper attention, the result can be disastrous.

To move from developing to staging to producing a Web site requires:

- Establishing a means and location for your authors to create and modify content — which involves not only deploying a content management system but maintaining the integrity of the production server.
- Ensuring that your development and staging environments lead to a site that appears and behaves as expected once it is live — which means comprehensively testing a site in advance.
- Producing and modifying the actual site — which means facilitating the rapid and easy modification of a site, from upgrading of features to changing, adding, or deleting content.

Principles of Deployment

- It is not unusual to use the same server for development and staging, but ideally you should have three separate pieces of hardware for each environment:
  - A development server
  - A staging server, which is identical in kind if not in size and performance to the production server
  - One or more production servers
- The scheme for replicating data to the production servers should be appropriate for the size of your site and the complexity of updating it.
- To maintain good security and avoid a logistical nightmare, limit access to the production servers to a few engineers only; content authors should be segregated from the production servers. While deploying new code, use only a secure connection to the production hardware, over a private network card.
- As you modify content, you should carefully manage the order in which those changes are pushed through to the production server.


Content Authors
Development Servers
Source Code Control
Review & Approval
Templates
Staging Servers
QA Testing
Page Performance
Backup Copy of Production Server
Machine-Level Login Limited to Administrators Only
Production Servers
Final Copy for Public View
Machine-Level Login Limited to Administrators Only
Internet

Relationship of Development and Staging Environments

The Development Server

Your development server is the shop in which you build your site. If the site is not too large, then you will probably use FTP to transfer data files from this server to the next. For the development server, you are certain to need back-end development software and likely to need content management software.

Back-End Development Software

For back-end development software, you must have source code control software, such as SCCS, RCS, or CVS, as well as performance testing software. In addition, an integrated development environment (IDE) is helpful.

Content Management Software Revisited

Content management has already been addressed in Block 6 as a necessary component of content-oriented sites. Here, however, we add greater detail and focus on its importance for development and deployment. The same basic goals are met by content management software. It facilitates:

- Publication of content
- Development or modifications by people who may not have technical backgrounds

Functions for the Less Technical: Authoring

For less technical contributors, your choice of content management software may be more useful if it includes presentation templates for converting and presenting content in formats such as HTML, XML, WML, and PDF.

The software should provide the ability to:

- Create presentation templates.
- Centralize control of site design elements.
- Easily define navigation rules and generate site navigation accordingly.
- Select content criteria.
- Define site hierarchy.

Functions for the Less Technical: Workflow

To segregate back-end issues and access from content publishing functions, the content management solution should offer:

- Ability to prepare and automatically post materials to the site (self-service authoring)
- Browser-based authoring (so desktop authoring software is unnecessary)
- Version archiving and audit trail

- Integrated workflow
- Content scheduling

Functions for the Administrator
To enable you to administer site development, content management software should include the capacity to:

- Manage users remotely
- Establish and monitor security
- Categorize content both automatically and according to in-house rules (metadata).
- Archive content, preferably in a repository that allows for its reuse.

The Staging Server

After development, your site needs to be moved to a staging server. Although this version of the site is not live, the staging server should be identical in kind (if not in size and performance) to the production server:

- The software on your staging server should be configured identically to the production machine, so it is a true basis for quality assurance testing.
- Your staging server hardware should be comparable to the production hardware. In that way, your site administrator can test performance on it, multiply for the number of machines deployed in production, and thus forecast site capacity accurately.

If you take a production Web server out of commission for maintenance...

Remember that new content still flows to the remaining production servers, so you must take care to resynch the machine that was off-line to “catch up” with the latest content.

QA and Bugs
On the staging server, you must test the site for QA and for bugs because it is that copy of the site you send to your production servers when you go live. In particular, you want your QA on the staging server to check the following:

- Validity of all hyperlinks.
- All other navigation on the site, including:
  - Landing pages for all sections of the site.
  - Links on each of the pages (including such standard sections as Help, Privacy Policy, Terms and Conditions, Advertising Sales, About Us, and Contact Us).
- Site search functions.
- All other components, special functions, or features, from calculators and polling features to applications.
- All live feeds.
- Appearance: All the pages look as you intend them to — including advertisements.
- Performance: All pages load quickly.
- Testing: The site passes any additional load testing, script testing, and database connection testing.
- Spelling on each page.
- Content: The content works well on the types of browsers you intend to support, including such elements as the format and page layout.

Bugs
Bugs are inevitable; the efficient treatment of bugs is not. You should centralize knowledge about all bugs, in effect creating a single database for your testers' reports on problems. If you have a large site that you administer in-house, you may want to assign a single person to oversee the record and verify that all bugs are fixed.

Your process for fixing bugs should include:

- Centralizing the reports of all fixes.
- Determining which bugs are highest priority and scheduling corrections accordingly.
- Meeting regularly about bugs.
- Separating fixes from added features and functionality, which means using a separate process and database for any modifications to the site.

The Production Server: Going Live

Mirroring via Manual FTP, Rsync and Perl Scripts, or Content Management Software
Finally you reach the production servers, on which the live site resides. Here, you need some kind of synchronizing protocol to mirror the site. Content management software often automates synchronization.

If you do not have automated synchronization, then you'll have to develop your own protocol. For modest changes to your site, you might be able to manually send an FTP version of each file from the staging to the production server as a means of replicating. But most sites are too sophisticated for such a treatment, and as the modifications increase, you will quickly find that manual replication is not practical. Rsync, a home-grown Perl script, or some other program, will be necessary to ensure that mirroring occurs fully.

Rsync and Perl scripts are more efficient means of synchronization than FTP. They can identify automatically the changes in the directories and files, so they more efficiently select what you need to synchronize than manual FTP does. File management will also be better with Rsync and Perl scripts because they will automatically delete old files, thereby reducing load size on the Web server.

The Logistics of Modifying Content

No matter how carefully you design your system, deployment and staging issues will arise each time you modify the site. You therefore need to pay special attention to the logistics of publishing to avoid a broken site. If, for example, the publication process takes half an hour to transfer all the files for a new article you are publishing, your Web site will have a broken link unless the Web page that contains that link is the last page published. You may want to begin with the graphics, follow with the article, and once everything else is synchronized, publish the Web page with the link.

Building Block 8: Metrics, Monitoring, and Performance

You've laid the foundations, added the functions, and created the means for developing and staging. But your site functions only as well as the ability you have to monitor its operation and performance.

The capacity for monitoring, metrics, and performance analysis should be not an afterthought but built into your site at each stage of its development. The more you do so, the more carefully you can observe the functioning of your site, which in turn affects your ability to improve performance and locate any problems precisely and quickly. Good monitoring and metrics save you considerable time, resources, and hence money.

In this final block, we examine metrics, monitoring, and performance in these sections:

- Goals of metrics and monitoring
- Host monitoring
- Application and database monitoring
- URL monitoring
- Traffic analysis
  - Number of raw hits and page views
  - Number of site visitors
  - Click-through analysis
- Backup systems and network monitoring
- Reports and management

Goals of Metrics and Monitoring

While you need to monitor all disks comprehensively, you also need to monitor and analyze what your data indicates about the current and future performance of your network. And performance analysis takes a number of forms — machine performance, application metrics, transaction rates, and stress testing.

Metrics and monitoring should not only enable you to administer your site efficiently and comprehensively but also contribute to building a strategy for the site. They should provide the following information:

- Whether or not any or all parts of your site are down — and precisely which parts
- Whether or not all parts of the site are popular and otherwise productive for you

- What your users are doing on the site
- Where your visitors are having problems
- What your future needs are

To acquire that information, you want metrics and monitoring ability in several areas, beginning with host monitoring.

Host Monitoring

Monitor each piece of hardware on your site. At the very least, perform heartbeat monitoring to ensure each machine is functioning. Of course, a more sophisticated form of this minimal monitoring would come from an internal agent. But it pays off in the end to monitor the hardware comprehensively, including:

- CPU utilization — the percentage of the CPU utilized
- Ethernet traffic — how much traffic is going in and out from each network interface
- Disk utilization — which includes disk capacity and I/O performance
- Memory utilization — which should include memory swapping

Sample Output from the mpstat Command

```
CPU  minf mjf  xcal  intr  ithr  csw icsw  migr  smtx srw  syscl  usr  sys  wt idl
  0   296   2   881     2     0  127    1    15    13   0    576    2    3   4  91
  1   221  32  1183   246    44  193    2    12    37   0    614    6    4  28  63
  2   473   6   796    40    39  147    1    16    27   0    909    3    6   9  82
  3   230   8   670   341   339  201    2    13    28   0    664    4    4  10  82

CPU  minf  mjf  xcal  intr  ithr  csw icsw  migr  smtx srw  syscl  usr  sys  wt idl
  0   165   70  1606     1     0  221    1     8     2   0    289    4    3  67  26
  1    37    6   668   242    42  105    1     7     2   0    411    6    2   9  83
  2     9    0   676    30    30   78    0     7     2   0    151    0    1   4  95
  3   234   23   363   475   473  101    2     6     2   0    631   12    3  26  59
```

The mpstat command reports the following information:

CPU	processor ID
minf	minor faults
mjf	major faults
xcal	interprocessor cross-calls
intr	interrupts
ithr	interrupts as threads (not counting clock interrupt)
csw	context switches
icsw	involuntary context switches
migr	thread migrations (to another processor)
smtx	spins on mutexes (lock not acquired on first try)
srw	spins on readers/writer locks (lock not acquired on first try)
syscl	system calls
usr	percent user time
sys	percent system time
wt	percent wait time

All of these areas are important to monitor, but disk utilization is particularly noteworthy. It is common for Web server disks to fill up rapidly and unexpectedly with log files, which may lead the site to crash. This sudden filling of the disk is the result of any process on the machine that writes to the file system. (How rapidly the disks fill up depends in part on how you have partitioned them.) For example, a forum server keeping track of messages contributes to this build-up.

Monitoring the disk utilization of the mail server and database server is critical for the same reason. On the database server, for example, transaction logs are notorious for quickly filling up disks.

Host monitoring is a common service of co-location facilities, so you may want to outsource it. But of course there are many products you can buy and many such products that come with the hardware. Many of you will decide to build the monitoring tools yourselves.

Application and Database Monitoring

This monitoring should indicate whether or not the database and all parts of the applications are working. It should also indicate what their performance is, from which you can better scale your applications.

As you build your applications, you want to design monitoring metrics into them that measure each part of the application. With this kind of instrumentation, you can pinpoint where you need to debug an application. That precision is particularly important because as you add functions to an application, some parts of the application may break.

Consider the instrumentation you want for an e-commerce application with modules for viewing products, shopping cart, order status, and credit card transactions. You want to know that each part of that application is functioning. You also want to know how well each part is functioning and ultimately how long it takes to execute transactions and queries.

Internal database monitoring varies in kind with the database you use, but it should include the following:

- Transaction rates
- Memory utilization rates (and other measurements of how the software is managing memory)
- Cache hit rates
- Amount of time for reading and writing to the database

Co-location facilities rarely provide application or database monitoring, so you must develop this ability yourself.

URL Monitoring

With URL monitoring, you should evaluate not simply the home page but a sampling of all the Web pages you produce. Many services will check all of your pages for broken links, but you want also to assess network performance, especially response time to requests for pages, from different locations.

Therefore, be sure that such monitoring is both external as well as internal: It is easy enough to check the URL from within your system, to see if the URL is working for you. But you must also check from outside the network, ideally from a number of remote locations. Many commercial products perform URL monitoring.

Traffic Analysis

Your analysis should tell you what users are doing on the site and how often they come back. The potential sources for traffic analysis are HTTP log analysis and session monitoring of usage, the latter often a result of cookies. Of these two sources, log analysis is largely focused on IP addresses and doesn't easily distinguish between individual users. It is useful albeit more expensive to have real-time monitoring and analysis of cookies or other header information.

Traffic analysis on your site falls into three kinds.

Number of Raw Hits and Page Views

Raw hits and page views are important information for assessing network bandwidth usage and predicting future traffic for capacity planning. Public relations and advertising people for your site will also want this information to prove just how big a site you have.

Number of Site Visitors

The number of different site visitors provides a useful metric for assessing stickiness, the length of time someone stays on your site for each visit. Your marketing people will want this information, but so will your site administrator because it may indicate the satisfaction level of your users as they visit or navigate your site.

Click-Through Analysis

Click-through analysis, the sequence of pages a visitor follows, tells you still more accurately how users navigate the site. Your analysis should provide you with information on the first and last pages users visit as well as where they go between the two.

Of this information, the last page is particularly important. From it, you can assess whether users have an easy navigation experience and other information about their site experience. For example, if the last page many users click to is the search results page or the site index page, then the search engine of the site or the index is likely to be frustrating rather than leading them to continue exploring the site. If many visitors exit at a page that is a sign-up form, it may tell you that the form is discouraging users. If you have an e-commerce site, the last page allows you to calculate how often people leave the site before completing transactions.

Backup Systems and Network Monitoring

Monitoring both your backup systems and your network should be part of your overall disaster recovery plan. This monitoring and analysis should ensure that all of your standard operating procedures — that is, your plans for how you do backups and monitor — actually function. (Many systems use their failover networks to perform metrics.) Be sure that you audit tape backups and other storage elements, which are sometimes forgotten elements in such monitoring. And include in your monitoring the people who are part of these procedures: Monitor the technician who must change the tapes every day.

Reports and Management

Just as important as the specific monitoring functions is what you do with all of this information. To that end, you want to create reports, centralize management, and establish clear alert and escalation procedures:

- Reports. Monitoring and metrics in all areas should lead to generating graphs, pictures, and reports for system administrators and other managers.
- Management. To make changes systemwide and to debug easily, handle error logging and session management in one place.
- Alert and escalation procedures. As part of management, these procedures should identify whom to alert and what else to do with each malfunction. If an event is repeated a second or third time, or if a repeat occurrence creates a more serious problem than the first, you want the chain of command to be clear.

Acknowledgments

This guide could not have been developed without the help of many technical experts. To them, a heartfelt thanks.

Sun experts provided the following assistance:

Felix Serna supplied valuable general commentary.

Ben Colborn gave critical explanations on selected OS security issues.

Eric D. Larson critiqued the general outline and offered important insights about identity policy.

Will Snow greatly enhanced the sections on server hardware; metrics, monitoring, and performance; and identity and security policy.

John Hoffmann and the java.sun.com engineering group provided critical understanding on development and staging environments as well as application servers and databases.

Warren Belfer provided insight and generosity (as well as patience beyond the call of duty) on server hardware; operating systems; metrics, monitoring and performance; and network infrastructure issues.

And **Charlotte Allen** effectively pioneered the entire outline for the project, fundamentally influencing its shape and providing all the other consultants with a set of ideas to which they reacted.

From outside of Sun, three experts provided invaluable help:

Rich Teer of Rite Online Inc. shepherded the writer through the intricacies of creating a data center.

Phil Bartholo of WaterWare Internet Services gave expert advice on application servers.

Finally, this project would not have happened without **Mark Waters** of WaterWare Internet Services, who is virtually a second author, having provided countless hours of explanation on every block of the primer.

Thanks to Ghostdog Design for their artistic direction and the Dot-Com Builder production team.